3-2003

Inquisitive Pattern Recognition

Amy L. Magnus

AFIT/DS/ENG/03-09

INQUISITIVE PATTERN RECOGNITION

DISSERTATION

Amy L. Magnus
Major, USAF

AFIT/DS/ENG/03-09

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Research sponsored in part by the Air Force Research Laboratory, Air Force Materiel Command, USAF. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation thereon. The views and conclusions contained in this dissertation are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory, Department of Defense, or the United States Government.

AFIT/DS/ENG/03-09

INQUISITIVE PATTERN RECOGNITION

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Electrical Engineering

Amy L. Magnus, B.S.E.E., M.S.E.E.

Major, USAF

March 2003

AFIT/DS/ENG/03-09

INQUISITIVE PATTERN RECOGNITION

DISSERTATION

Amy L. Magnus, B.S.E.E., M.S.E.E.

Major, USAF

Approved:

Signed

Dr. Mark E. Oxley — Date
Dissertation Advisor

Signed

Dr. Gary B. Lamont — Date
Committee Member

Signed

Dr. Steven C. Gustafson — Date
Committee Member

Signed

Dr. Milton E. Franke — Date
Dean's Representative

Accepted:

Robert A. Calico, Jr. — Date
Dean, Graduate School of Engineering and Management

Preface

Inquisitiveness is marked by a drive to bound experience. The inquisitive computation achieves a primitive stage of critical thinking analogous to the Terrible 2s—the age where children are infamous for asking "Why? Why?" ad nauseam. This behavior is an attention-getting device, but the child is also acting on the recent revelation that their parents don't know everything. Children at this age aren't really interested in knowing exactly why the sky is blue; instead, they are attempting to bound the limits of what their parents know. If so motivated, this stage in child development is a striking early sign of critical thinking—the disciplined intellectual criticism that combines research, knowledge of historical context, and balanced judgement. The advancement of scientific knowledge often turns on asking the right question. We can expect the advancement of machine-based knowledge to turn on the same point.

Amy L. Magnus

Acknowledgements

Anything begun in vanity ends in humility.

Writing a dissertation is much like entering into a dream state for several years. Now, at its end, I find myself reawakening in awe—much like Dorothy fresh from Oz—and seeing my own dear people who were always there. I extend my deepest gratitude to you all: to Dr Mark Oxley, for his exceptional mind and vision; to Dr Gary Lamont, for his steady support and ready understanding; to Dr Steve Gustafson, for asking the probing questions; to Dr Steven Rogers and Dr Matthew Kabrisky, who never fail to inspire, to Lynda Myers, my best friend who cheered me on from the start; to Scott Harris, for showing me the way to the end; to Rob Pope and Leigh Method, the unstoppable team; and to Jeremy Holtgrave, we're crossing the finish line and together.

I could not have done this without you or my tremendous family: my father and grandfather who taught me the joy of engineering; my older sister who taught me the joy of music, drafting, and multiplication; my older brother whom I idolized from the first and followed through his ardent explorations of astronomy and science, and my little sister—fellow songwriter, yogi, and exceptional cookie maker—who, with my younger brother, continues to teach me about cool, my grandmothers, one who taught me generosity and ingenuity, the other who taught me perseverance and dedication; and my stepdad who taught me composition through quiet observation; my aunt who taught faith and her song "Ain't got no troubles" to hum through tough times, and my uncle who taught the key to invention is being the first to recognize that something we aspire to is at last attainable, and, finally...

I dedicate this research to my mother. She always found a connection with the work from suggesting the Prochaska/DiClemente change wheel to seeing the crazy quilts of Chapter VI. Mom, you are the best mentor a woman could hope for.

God bless.

Amy L. Magnus

Table of Contents

Page

List of Figures

AFIT/DS/ENG/03-09

Abstract

The Department of Defense and the Department of the Air Force have funded automatic target recognition for several decades with varied success. The foundation of automatic target recognition is based upon pattern recognition. In this work, we present new pattern recognition concepts—specifically in the area of classification—and propose new techniques that will allow one to determine when a classifier is being arrogant. Clearly, arrogance in classification is an undesirable attribute. A human is being arrogant when their expressed conviction in a decision overstates their actual experience in making similar decisions. Likewise, given an input feature vector, we say a classifier is arrogant in its classification if its veracity is high yet its experience is low. Conversely, a classifier is non-arrogant in its classification if there is a reasonable balance between its veracity and its experience. We quantify this balance and we discuss new techniques that will detect arrogance in a classifier.

Inquisitiveness is, in many ways, the opposite of arrogance. In nature, inquisitiveness is an eagerness for knowledge characterized by the drive to question, to seek a deeper understanding, and to challenge assumptions. The human capacity to doubt present beliefs allows us to acquire new experiences and to learn from our mistakes. Within the discrete world of computers, *inquisitive pattern recognition* is the constructive investigation and exploitation of conflict in information. This research defines inquisitiveness within the context of self-supervised machine learning and introduces mathematical theory and computational methods for quantifying incompleteness—that is, for isolating unstable, nonrepresentational regions in present information models.

The key methods of inquisitiveness presented are (1) falsification and (2) the classification of confusion in feature space. This work also introduces a functional model for persistent learning and a simplified model for data fusion tailored to the development of pattern recognition algorithms. Artificial neural network demonstrations are provided to illustrate inquisitive pattern recognition techniques.

Inquisitive pattern recognition (IPR) is a relational reasoning capability that allows computers to learn from imperfect decisions. IPR has immediate application to Air Force problems—specifically treaty and terrorist monitoring—where rare events must be separated from the mundane. This technology also represents a step forward in machine learning—toward more viable, flexible computer tasking in adaptive, collaborative environments.

INQUISITIVE PATTERN RECOGNITION

I. Introduction

The sophisticated computer should spur on, not squash, human innovation. Right down to its cold computational heart, a computer should be dedicated to its user's will, striving continuously to become more effective in its assigned tasks, to perfect its knowledge and performance in service of its user.

Over the years, computational machines have attained various levels of sophistication. Let us consider the elusive: intelligence. In a 1950 article "Computing Machinery and Intelligence" [99], Alan Turing introduced a test for computer intelligence—a concept that has become known as the Turing Test. For his problem formation, Turing proposed an imitation game in which, if a computer can fool a human interrogator, then the computer is intelligent. Imagine a game played with three people: a man, a woman, and an interrogator. The interrogator stays in a room apart from the other two but is allowed to submit questions to them via a messaging system. The interrogator's objective is to determine who is the woman. The role of the woman is to be the expert and to convince the interrogator that she is, indeed, the woman. The role of the man is to be the pretender—that is, to fool the interrogator that he is the woman. In the Turing Test, Turing proposed replacing the pretender with a computer—as seen in Figure 1—and suggested that an intelligent computer would be able to fool an interrogator into believing that it is human. [99]

One problem with the Turing Test is its subversive nature. It is ethically dubious to suggest that a computer must misrepresent itself in order to prove it is intelligent. We would prefer a computer that is capable of weeding out the pretender rather than a computer that aspires to be one. Consequently, we propose a modified Turing Test, which we shall call the Inquisitive Test.

The Inquisitive Test, shown in Figure 2, asserts that a computer is inquisitive if it is able to distinguish an expert from a pretender—for instance, if it is able to distinguish

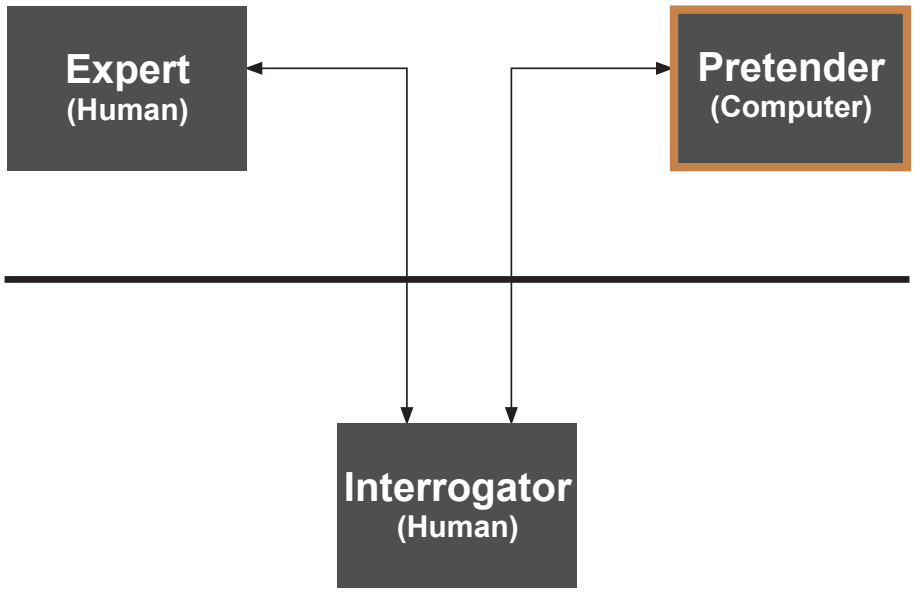


Figure 1. Turing's test for computer intelligence. If a computer is able to convince an interrogator that it is human, Turing suggested that this is a sign of intelligence.

appropriate expressions of expertise from expressions of arrogance[1]. In our form of the imitation game, we replace the human interrogator with a computer interrogator while the expert and the pretender may be people or computers. Automated interrogators have application in the evaluation of relative expertise and the promotion of intelligence amplification[2] among experts. Military applications for quantifying expertise abound, particularly in systems involving technologies such as Information Fusion, Intelligence Amplification, and Multi Agent Systems. One specific example is a new initiative from the Air Force Office of Scientific Research, the Hybrid Inferencing from Fused Information (HIFI) program; the goal of this program is to replicate expertise in human intelligence gathering through automation and to create and maintain flexible ontologies shared between human and automated systems. Another example of an Air Force application is sensor-based treaty monitoring where automated experts must confidently separate rare events from events that occur 1000+ times a day.

[1]Arrogance is a over-statement of expertise where the expert has no or insufficient experience. We define arrogance in classification formally in Section 5.1.2 and demonstrate an arrogant classifier—the multilayer perceptron—in Chapter VI.

[2]Intelligence Amplification (IA) is the enhancement of human intelligence through human-computer interaction [84].

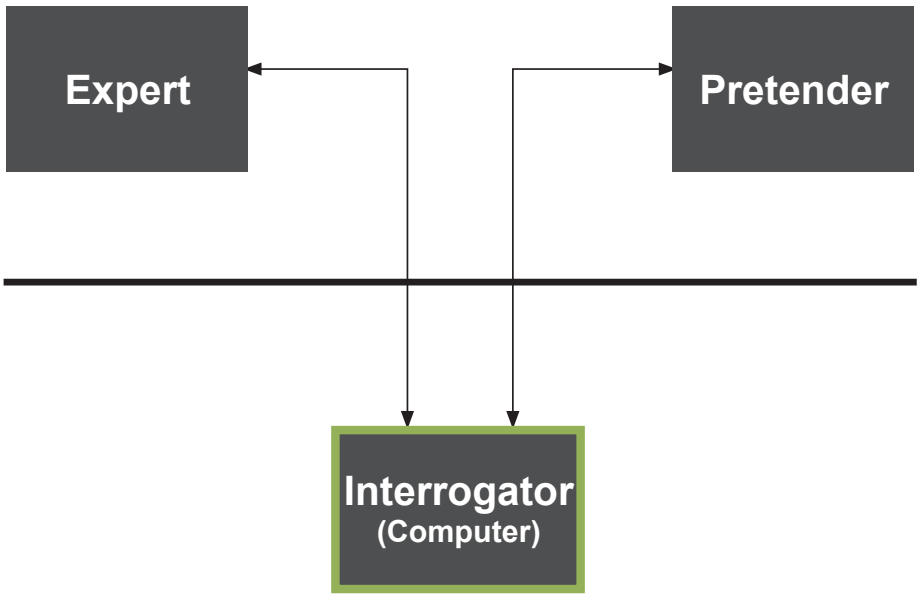


Figure 2. Our proposed test for inquisitiveness. We suggest that an inquisitive computer in its role as *interrogator* has the ability to distinguish an *expert* from a *pretender*. In the case where all three players—the interrogator, the expert and the pretender—are internal to an autonomous computer agent, the inquisitive test is an exercise in self-evaluation.

The Inquisitive Test—more so than the Turing Test—has real and immediate application to information management systems under development today. We see inquisitiveness as a necessary component of true intelligence—that is, to say, an intelligent computer must be inquisitive, but the inquisitive computer is not necessarily intelligent. Further, we believe that the Inquisitive Test is an important intermediate step in our quest to demonstrate and enhance intelligence and, to the benefit of Artificial Intelligence research, far more easily realized.

1.1 Problem Statement

This dissertation develops a new thrust in pattern recognition research—called inquisitive pattern recognition—and extends the discipline to address the hard problem [20] in computational intelligence: self-evaluation in machines. Traditional Pattern Recognition does not have a methodology for quantifying relative expertise. Inquisitive Pattern Recognition directly addresses this deficit. We propose the mathematical development of inquisitive pattern recognition as the constructive unification of multiple perspectives on a

single pattern recognition task. The challenge of inquisitiveness is negotiating among these varied perspectives—in particular, leveraging the conflict and confusion between "experts". To comprehend an expert's strengths and weaknesses, one must envision the expert's knowledge not as a seamless enigma but as the intermeshing of many beliefs tracing out a larger truth. It is the "seams"—the transitions—among beliefs that demand our most careful attention.

1.2 Scope

This dissertation introduces inquisitive pattern recognition (IPR), an investigative methodology that supports the evaluation of data generalizations trained via self-supervised learning[3]. Key to inquisitiveness is the isolation of those regions in a data generalization that may lead to poor decision making. In support of this methodology, several concepts are introduced and developed including the following:

1. Persistent learning—a process model for transitioning pattern recognition tasking from supervised experimentation to self-supervised customization in the design and maintenance of a pattern.
2. The theory of confusion—measure theory for quantifying incompleteness of information by contrasting peer experts.
3. Four-value logic—a label set for data generalization that expresses its relative experience level, i.e., {"extrapolation", "interpolation"}, as well as its veracity, i.e., {"false","true"} of experts.

Other topics include Expertise Logic—an extension of Fuzzy Logic—and arrogant classifiers. We show, for instance, that 4-value logic is a special case of expertise logic. The mathematical constructs of confusion theory and 4-value logic are applied to the multilayer perceptron. In doing so, we offer an appropriate new tool for evaluating multilayer perceptrons in data generalization applications.

[3]We define self-supervised learning as a training process that combines *a priori* knowledge and unlabeled *a posteriori* observations into a unified representation. See Section 2.1.2 for further detail.

The primary focus of this research is the development of novel techniques for evaluating learned patterns. As such, it is not our goal to build better data generalizations but, instead, to evaluate generalizations in order to (1) prevent instances of inappropriate application and to (2) communicate with reasoning engines—such as an interrogator—that benefit from explicit expressions of expertise.

1.3 Overview

The composition of this dissertation proceeds as follows. Chapter II discusses background on pattern recognition. Chapter III, Inquisitive Pattern Recognition, introduces a model for persistent learning—derived, in part, from human behavioral science—and defines inquisitive pattern recognition within this model—specifically, as the passive, investigative skills in persistent learning. Chapter IV presents the Theory of Confusion in terms of measure theory and introduces four-value logic. Chapter VI applies four-value logic to a combinatorial-geometry treatment of multilayer perceptrons. Finally, in Chapter VII, Conclusions and Recommendations, summaries the contributions of this research and indicates areas for further research.

II. Background

Pattern recognition faces a rich set of challenges. For many years, the discipline was insolated as its practitioners designed dedicated solutions for sensor-to-computer-to-user applications. More recently, pattern recognition research has focused on the sensor-computer interface leaving the human-computer interface to other disciplines. Applications in today's environment demand greater responsiveness, flexibility and explicit expression from the embedded pattern recognition algorithms particularly since the algorithms must be responsive to both the sensor and the human-computer interface. The discipline's new challenges come from the later interface where algorithms must ensure that—given present sensor evidence—queries posed by the human-computer interface are relevant.

Classical pattern recognition investigates trends and consistencies in data. We seek to flesh out the methodology of pattern recognition to include techniques that investigate the incompleteness of data. Such examinations tender clues over what is real, but generally unanticipated. Inquisitive pattern recognition is a method for solving knowledge aggregation problems[1] by facilitating the identification and classification of conflict and ambiguity in information models. Strategies for knowledge aggregation have broad appeal in collaborative software applications such as adaptive data fusion, negotiations between intelligent agents, and the automated maintenance of knowledge bases—applications where multiple information treatments must be allied to one purpose.

This chapter discusses trends and methods in pattern recognition, data fusion and computer-augmented environments and—through these discussions—builds the case for research into inquisitiveness.

2.1 Pattern recognition

Pattern recognition (PR) is the process of associating patterns to events of interest [38]. Pattern recognition operates in two modes: the design of a pattern and the application

[1]Knowledge aggregation is the accumulation of knowledge for application in new and unique contexts; problems in knowledge aggregation involve resolving value-added information from multiple, typically heterogeneous sources.

of the pattern. Patterns take two forms: data reductions and data generalizations. A data reduction is a relation or ordering reduced from a set of experimental observations. Contrarily, a data generalization[2] is a relation or ordering induced from a set of experimental observations.

In pattern recognition research, data reduction first garnered the bulk of attention mainly due to the importance of economy in communications, computer processing and media storage. Now, with boons in communication bandwidth, processing power and memory capacity, we can afford to invest in data generalizations. Unfortunately, the early emphasis on data reduction has biased research in data generalizations heavily and in some cases inappropriately.

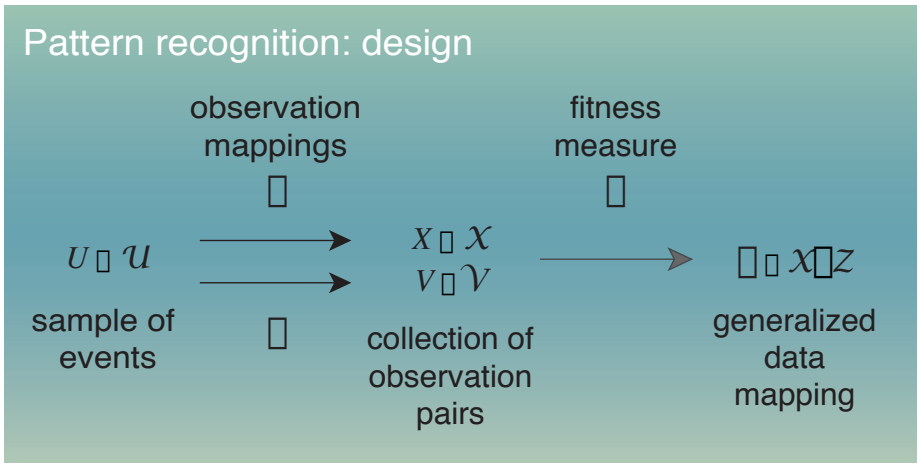
A paradigm shift is needed, and it is important to understand those forces that demand this shift. Accordingly, in this section we present background discussion on the training and application of data generalizations. Then, in Section 2.2 we discuss parallel research areas in information management that demand more from pattern recognition.

2.1.1 Generalized data derived from incomplete populations. Let us consider the data generalization. Often, a data generalization is a set of ordered pairs inferred from the set of multiple observations and a set of corresponding labels[3]. At design, patterns are generalized from sample sets of event data. A typical scheme is illustrated in Figure 3(a) where sampled observations are matched to a corresponding set of observed labels v. Here, the product of design is the relation θ, a data mapping from an observation set $\mathcal{X}$ to a decision set $\mathcal{Z}$. The mapping explicitly matches an observation $x \in \mathcal{X}$ to a decision $z \in \mathcal{Z}$. The relationship between labels and decisions is implied. In subsequent application as in Figure 3(b), the data mapping θ translates present evidence x to the decision z—a speculation on what event u likely occurred.

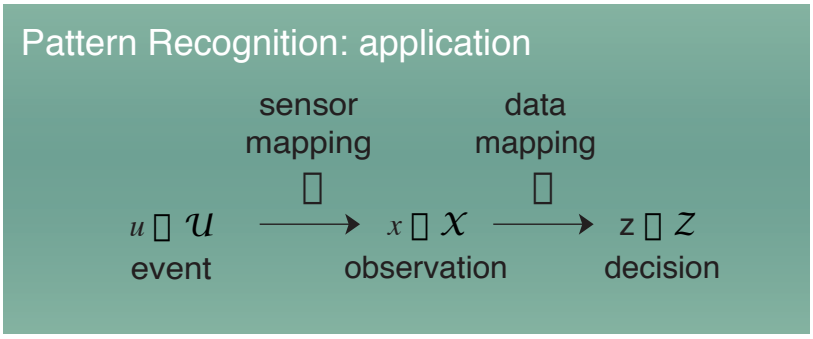
Given that a sample is a small (though hopefully representational) subset of a larger set, we define a data generalization as follows:

[2]Discussed further in Section 2.1.1 below.

[3]In pattern recognition, a label implies—explicitly or implicitly—a sensitivity, a quantitative assessment of the value of an observation—e.g. for example, the proportion of people who truly have a specific disease and are so identified by a clinical test [96]. Explicit representations of sensitivity often take the form of probability density functions (as in Bayesian networks) or membership functions (as in fuzzy logic).



(a) Design mode



(b) Application mode

Figure 3. Operating modes in Pattern Recognition: (a) At design, a data mapping θ is generalized from a set of supervised events based on the collected observations x and a corresponding set of labels v. The supervised collection of labels is denoted by observation mapping ζ. (b) In application, the data mapping θ transforms sensor observations x into decisions z relevant to an unsupervised event u of interest.

Definition 1 *(Data generalization). A data generalization is an explicit relation designed from a sample of experiences and extended to a larger set of new experiences as seen in the application, or operational use, of the generalization.* [62]

Let a population be an ordered set. Data—i.e., a sampled population—have meaning if the data are ordered. Quality training data faithfully convey the order of the entire population. One major challenge in pattern recognition is to capture quality training sets of minimal (or reasonable) cardinality. The next challenge is to generalize populations from the training data. Let PD represent the population. Given training data $TD \subset PD$, there exist several data GD that generalize TD such that the generalized data GD contains TD. The goal is to find a GD which maximizes the cardinality of $GD \cap PD$.

If generalized data GD is an ordered set containing the training sample ($TD \subset GD$), then let us define a data generalization as the ordering on GD. Because there exist several sets GD that contain the incomplete population TD, many data generalizations also exist. Any data generalization inferred from training data where $TD \neq PD$ is by definition an incomplete population (arguably even where $GD = PD$ as this equality is impossible to prove).

2.1.2 Training: distinctions in supervision. In the design and maintenance of data generalizations, there are three basic modes of training[4]. Figure 3(a) depicts the most common: supervised training. The other two are unsupervised and self-supervised. Training is often supervised during the design of a generalization; but in application, if learning continues, it must do so autonomously and, thus, either unsupervised or self-supervised. [5]

The three modes of training are separated by distinctions in data collection, specifically whether or how labeled training data are collected. A label is an abstract characterization of an event of interest. Supervised and self-supervised training fits observations x to labels, depicted as the decision z in Figure 3. Unsupervised training uses unlabeled observations x. Let x be an observation and v be a label selected from object sets $\mathcal{X}$ and $\mathcal{L}$, respectively. Label $v \in \mathcal{L}$ is either collected directly from the experiment, or it is derived

[4]Training modes for data reductions are supervised and unsupervised learning. There is little motivation for self-supervised learning in the production of data reductions.

[5]See Chapter 3.1 and Figure 6 for further discussion and an illustration.

after the fact. Given observation $x \in \mathcal{X}$ and label $v \in \mathcal{L}$, unsupervised, supervised, and self-supervised may be distinguished as follows [27, 13, 98, 30, 61]:

- Unsupervised learning

  — A training sample consists only of an observation x of present evidence.

  — Inferred patterns are drawn from the "natural order" of observations—for instance range, resolution, distribution, and correlations.

- Supervised learning

  — A training sample consists of an observation x matched to a collected label v, a label garnered from an authoritative source.

  — Patterns are inferred from the parallel trends among observations and labels.

- Self-supervised learning

  — A training sample consists of an observation x matched with a derived label v, a label generated by a source of questionable reliability.

  — Patterns are refined by contrasting the apparent natural order of observations with the derived expectations.

Collected labels versus derived labels differentiate the training sets of supervised and self-supervised learning, respectively. Collected labels are direct observations from the pattern recognition experiment, or they are provided by an authoritative expert. Derived labels are inferred *a posteriori* by artificial intelligence—either by the pattern recognition algorithm itself or by a peer algorithm. In general, collected labels are more reliable. However, *a priori* knowledge—such as domain context and assumptions—is more likely to be explicitly stated when labels are derived. This is advantageous as, when new training data conflict with derived labels, i.e., the explicit assumptions can be debugged and contextual awareness enhanced.

2.1.3 Hold-out validation. The worthiness of a data generalization is completely determined by the appropriateness of the generalization in analyzing new observations, i.e., by its competency in linking new observations to predicted outcomes [13, 83].

Hold-out validation is a popular supervised learning technique which attempts to select appropriate generalizations for a given data set. The hold-out method is used as a means to select an appropriate stopping point for training. For this process, one separates design exemplars into two sets: a training set and an evaluation set. A data generalization is induced from the training set while the evaluation set is held back. An error trend is drawn from the evaluation set and, when this trend begins to increase, training stops. As advertised, the hold-out method stops training before memorization occurs—halting the process when an appropriate level of complexity in the representation is achieved. Error can be tracked because learning is supervised, but error trends for the training set tend to be misleading. Given a convergent learning algorithm, the error from the training set will continue to decrease until the set is memorized. Memorization is undesirable; it defeats the purpose of training since the resultant knowledge representation does not generalize well in recognizing new data. [83]

Hold-out validation and its computationally-intensive extensions—cross validation and bootstrapping—are often successful at reducing the complexity of a trained generalization. In practice, however, these data-centric techniques are inadequate to prevent memorization. When any of these methods are used to select an appropriate generalization from several different architecture, the estimate of the generalization error of the "best" architecture will be optimistic [103, 44]. More reliable set-centric techniques—such as the method described in Chapter VI—resolve trends over significant regions of the domain[6] not just at easily isolated data points.

2.1.4 Validation after design. Besides preventing memorization of supervised training data, it is also desirable to be able to test the appropriateness of a data generalization beyond supervised design and into application. In application, additional observations are gathered and new experience may be gained. Hold-out validation tracks error trends for isolated labeled data, but what patterns can be tracked when the preponderance of training data is unlabeled? Error trends require supervision; we suggest other trends derived from uncertainty or apparent incompleteness can be utilized.

[6] In domains of low dimension, it is possible to resolve trends over the entire domain as we will demonstrate in Chapter VI.

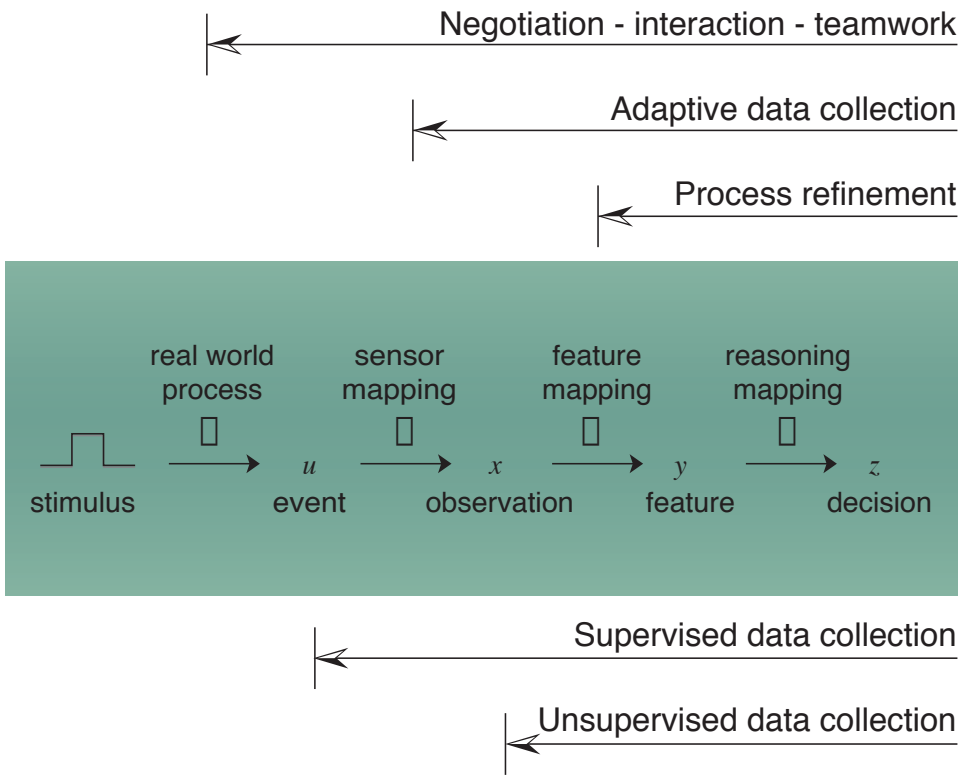


Figure 4. Justification for advanced investigative techniques in pattern recognition: Demands in adaptive information fusion and negotiating computer agents stipulate more dynamic mappings within a pattern recognition system while insight into the problem domain is diminished by an increased reliance on unsupervised data collection to manage these dynamics.

2.1.5 Pattern recognition summary. The rise of data generalization applications offers an opportunity for the pattern recognition community to take stock and reconsider the discipline in a new light. Among the general classes of new applications, collaborative systems offer fertile proving ground for inquisitive pattern recognition technologies. Multi-source, multi-modal data inherently contain conflicting information. As artificial intelligence researchers solve the problem of how to assimilate new information into existing knowledge representations, it behooves pattern recognition researchers to ensure that relevant new information is collected in an effective, manageable manner.

2.2 The demands of collaborative software applications

Computer agents, information fusion, and computer-augmented environments belong to a class of collaborative systems that use knowledge aggregation—the assimilation of multiple perspectives or data treatments to manage information. Demand for collaborative information systems is increasing [51, 69, 25, 1] and with it the demand for advances in pattern recognition. Besides more flexible techniques in information management, the artificial intelligence community is seeking on-line technologies to support algorithm maintenance, learning, and task negotiation and innovation.

In response, pattern recognition must prepare products of increased sophistication and expertise—expertise that must be hard-won. As depicted in Figure 4, pattern recognition algorithms will operate under demands for greater flexibility and responsiveness while at the same time with diminished insight on real world events. Negotiations with artificial or real world entities and subsequent adaptations will destabilize the innerworkings of the pattern recognition blackbox while the preponderance of data collected to manage adaptations will be unlabeled data.

2.2.1 Incompleteness and agency. In resolving learning problems, it is important for a computer agent to express its own value—having gauged the completeness of its information model. A computer agent is a program that is autonomous, adaptive and cooperative [71]. Given these characteristics, agency implies that an adaptive program is relatively free from human supervision but that it must cooperate with peer programs. In other words, computer agents must master self-supervised learning[7].

Recovery. Recognizing incomplete information enables a computer agent to label extrapolations—experiences outside the realm of normal operating modes. When an autonomous agent falls into a strange new world, its internal programs should recover functionality reasonably well and succeed in its principal task. Reminiscent of Dorothy's house landing in Oz, recovery is facilitated when the computer agent—like Dorothy—has the wherewithal to realize "...we're not in Kansas anymore".

[7] See Section 2.1.2.

Expressing expertise. Imagine two adaptive negotiating computer agents both let loose to gather experience on a domain. Since agents are autonomous, their experiences will differ. Each independently develops an internal domain representation based on distinct training sets with no common exemplars. The agents may even employ different sampling strategies so that the data distributions may also differ. After a time, the agents are given a joint task to negotiate. During this negotiation, the agents should pool their collective knowledge and resolve to a single solution. If one agent lacks experience in regard to the assigned task, it should defer to an agent that has more experience. Therefore it is useful for a negotiating agent to be able to express where its internal representation reasonably interpolates and where it is extrapolating.

The primary task of an intelligent agent is not to learn but to act in an informed manner[8]. Still, in its auxiliary role as "learner", an agent should exhibit an awareness of its strengths and shortcomings and be able to deduce the consequence thereof.

2.2.2 Information fusion. Information fusion is an information management technology that highlights the key relationships and suppresses the redundancies of data collected from multiple sensors [69]. Data fusion evolved out of the development of military weapon systems in the 1980's. Today, the majority of information fusion applications operate in stable domains where the collected data are expected to remain within predictable boundaries for the life of the system. Typically, human operators are presented with aligned data, and they, not the computer, are delegated the responsibility of providing situation assessment. Besides these operators, most information fusion systems require another room full of humans to provide off-line support. This group ensures the stability of the system by maintaining its algorithms and knowledge base. [41]

Air Force leadership has established clear goals for acquiring information fusion systems. Data fusion is listed at the top of the critical technology list in the Air University study *Air power in 2025* [51]. Proposed Air Force applications include:

- Networked battlefields, enhancing battlefield awareness and information dissemination [101],

[8]See Principal task discussion in Section 3.1.1, The change wheel.

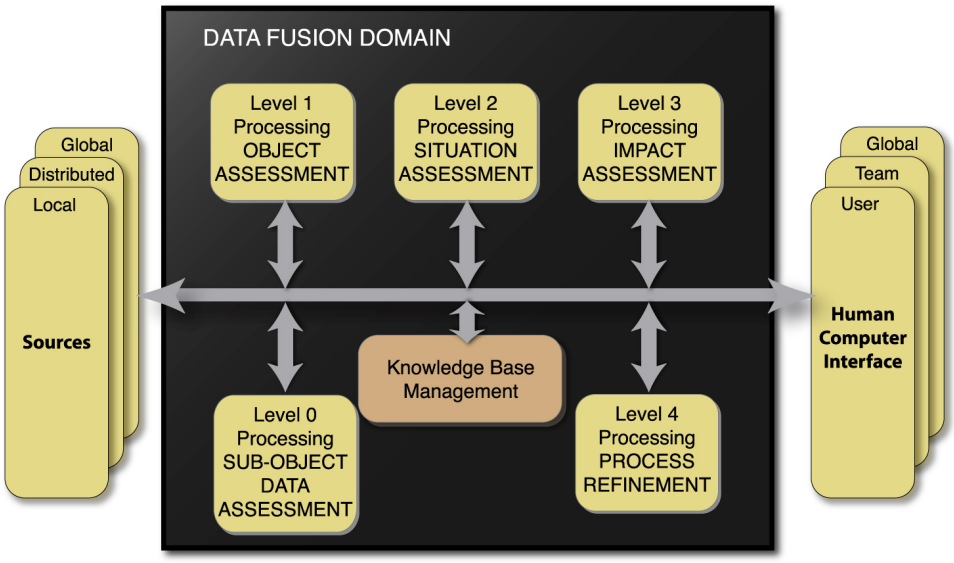


Figure 5. The Joint Directors of Laboratories standard for information fusion: This standard is a system level model.

- Interactive demonstrations of missions [36],
- Telemedicine, a medical delivery system, for battlefield triage [90],
- Medical training for rapid-response, disaster-level emergency procedures [35],
- Modeling of C4I systems [25].

Due to this push for more and varied systems, it becomes necessary to minimize the need for intervention from human operators and maintainers. To keep demand on human resources reasonable, system designers must delegate more fusion tasks to the computer and provide on-line capabilities for algorithm maintenance.

2.2.3 System-level functions of information fusion. The standard data fusion model was first developed in 1985 by the Joint Directors of Laboratories (JDL) Data Fusion Group [41]. This initial model and its subsequent revisions—the latest depicted in Figure 5—is the most widely used for categorizing data fusion-related functions in information management systems [92]. General categories include [92]:

- Level 0 - Sub-Object Data Assessment: estimation of observable entity states based on pixel/signal level data association and characterization;

- Level 1 - Object Assessment: estimation and prediction of entity states;
- Level 2 - Situation Assessment: estimation and prediction of relations among entities;
- Level 3 - Impact Assessment: estimation and prediction of effects on situations of planned or estimated/predicted actions;
- Level 4 - Process Refinement: adaptive data acquisition and processing to support mission objectives.

Level 1 and Level 0 are data or source management resources. Level 2 situational assessment is required for passive contextual awareness—nominally world modeling—the unified representation of the here and now. Both Level 3 and Level 4 functions are necessary for adaptive and interactive information fusion, respectively where fusion systems adjust to domain changes and where systems influence domain changes.

Progress in developing these functional areas has been mixed. The science of information fusion is mature for Level 0 and Level 1, immature in Level 2, and primordial in Levels 3 and 4 [43]. Early research efforts and most deployed data fusion capabilities centered efforts exclusively on Level 0 and Level 1 [41]. These efforts generally focused on early signal processing of multiple sources, often mono-modal sources. For example, two world class research sites—the MIT Media Lab and the Data Fusion Branch AFRL/IFE of the Air Force Research Laboratory—built substantive portions of their reputations on image fusion.

Systems incorporating automated Level 2-4 capabilities are much rarer [93, 92]. Most operational data fusion leaves the derivation of contextual information—situation and impact assessment—to human operators and support process refinement through dedicated maintenance teams. At the Information Fusion Workshop sponsored by the Information Institute on 22-24 May 2000, members concluded that many factors have stunted the automation of Levels 2 and above:

1. The exponential increase in the computational complexity of such functions,
2. The lack of standard guidelines,
3. The lack of common testbed problems for developing these standard guidelines,

4. The immaturity of pattern recognition in supporting interactive negotiation and persistent learning,

5. The immaturity of hybrid knowledge representations and inference engines that can move between the details of signal processing to the abstractions of natural language-based contextual inference.

Robotics can boast the most progress in Level 2-4 research but little of the formalism developed for robotics has been generalized and propagated to other disciplines [58]. For example, one fundamental flaw of the JDL fusion model is it does not include a cookbook for writing data fusion algorithms[9]; here, generalized guidelines for assimilating new observations into a robot's world model would have substantial impact.

At the Information Fusion Workshop, it was asserted by those who maintain the JDL fusion model that the biggest hurdle to Level 2-4 automation is the increased complexity of these functions. Crucial to the continued advancement of information fusion are the analytical and inference methods that master multimodal interaction within reasonable resources of time, memory, and computational might.

2.2.4 Computer-augmented environments. Research into virtual environments and augmented reality will also influence the development of information fusion. The technological demands of these highly interactive applications are destined to alter the course of information fusion research. Certainly, to move information fusion technology towards virtual environments is to move away from the specialized, narrowly tuned systems of today.

The military research establishment is motivated to ensure the success of computer-augmented environments [1, 90, 94]. Using highly-interactive, multi-user applications of virtual environments [91, 48] and augmented reality [95, 77], the military plans to improve mission execution through realistic training and increased accessibility to information [1]. There are extreme situations that—though rare—military teams must readily meet with practiced, certain skill. Real life enactments give such teams hands-on expertise; unfortu-

[9]This exclusion has been noted by the JDL model's architects as an intentional omission in a fledgling science and an attempt to keep the information fusion metaphor general—in the near term—encompassing both natural language and mathematical disciplines [93].

nately, these exercises are expensive and dangerous, and evaluating the training has proven difficult [1, 90, 94, 101]. The military sees virtual technologies as a means to improve command oversight of military exercises while increasing the safety of participants and decreasing the cost of the enactments [1, 94, 101].

Human dynamics are a threat to the stability of collaborative virtual environments (CVE) [7, 40, 102]. The success of these multi-user systems rely on the theories of computer-supported cooperative work (CSCW). Unfortunately, CSCW theories are not mature. Human dynamics—both social and political factors—have blocked the adoption of CSCW products in the workplace. The failure to garner group acceptance has led to many expensive failures impeding research in this area. [40]

Users want virtual spaces where interesting things happen [56, 52]; and, accordingly, the computer must allow for innovation from the human component. Jonathan Grudin in his paper "Groupware and social dynamics: eight challenges for developers" [40] states that a key contributing factor to past CSCW failures is the variability of group work. Groups—unlike larger parent organizations—tend to operate in informal structures. This informality lends flexibility so that group members continually renegotiate tasks of coordination, management, and allocation. Grudin stresses that flexibility does not necessarily mean malleability: A mature group expects new members—even new tools—to adjust to the established way of doing things. New groupware applications will likely be given the cold shoulder if they create extra work for members without an obvious return of investment, violate the expectations of members, or otherwise disrupt the group's social/political order [40, 49, 12].

As envisioned by Gloria Davenport, head of MIT's Interactive Cinema project, augmented environments are about "taking the computer out of the box and the media off the screen and burying them in the physical system [24]". A computer-augmented environment enhances the physical environment to allow users to interface with the computer in natural, intuitive ways [78, 95]. Creating virtual objects that respond naturally to real stimuli is no simple task [32, 11, 37, 26] and demands accurate, real-time recognition systems [91, 102, 11, 90]. Couple this interface response requirement with the requirement to allow for innovation in recognition tasks, and one can read the writing on the wall—the call

for intelligent interfaces and a disciplined pattern recognition approach that incorporates on-line, continuous learning and quality assessment [60, 15, 78, 24, 100, 32, 39].

2.3 Background summary

Customizing knowledge in collaborative systems requires discipline to sustain clear, complete, and timely representations. In keeping, pattern recognition must be augmented in both design and application methodology to ensure the collection of relevant new information in an effective manageable manner.

Classical pattern recognition is concerned—rightly—with representing knowledge fully and accurately in the computer, and traditional methods investigate trends and consistencies in data. Contrarily, inquisitive pattern recognition investigates the inconsistencies. It seeks to assign meaning to the incomplete, biased or volatile portions of data generalizations. The results of this investigation are then used to adapt a computer program's internal knowledge representation and to conform to an altered task.

III. Inquisitive pattern recognition in learning

In this introduction to inquisitive pattern recognition, we describe investigative methods supporting the selection of appropriate data generalizations during self-supervised learning—learning where the preponderance of training data is unlabeled. The objective of this work is to derive computational methods paralleling the logics of human doubt and intuition.

3.1 Persistent Learning: enabling online customization

The life of a pattern recognition algorithm unfolds in two main chapters: its design and its subsequent application. Let us consider the design of computer systems where learning must not end when the system is delivered to its end user but—as depicted in Figure 6—continues throughout its application. We assert such systems require multiple parallel investigations into the completeness of information models, at first offline during design and then online in application.

Inquisitiveness is part of an overall strategy to design persistent learning into computers. Persistent learning is the learning that continues after a knowledge representation transitions from design to application. Initial design-stage learning creates a generalized information model in a canned, offline environment. Then persistent learning customizes the model online—"on the fly"—using real world observations.

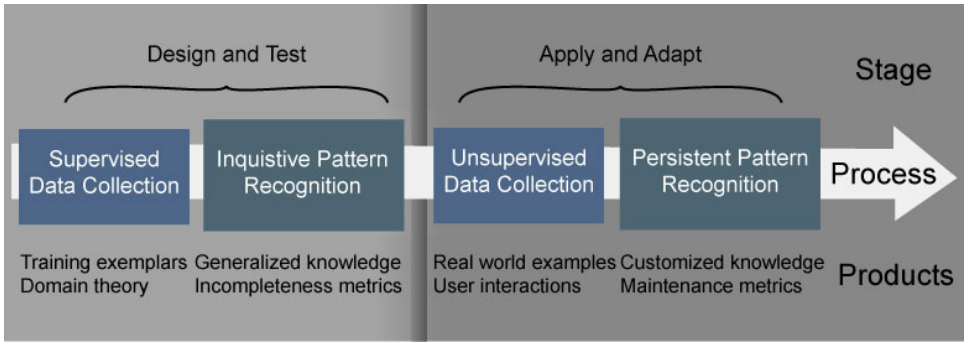


Figure 6. Designing intervention: maintenance and customization of data generalizations requires extending learning from initial design into online application.

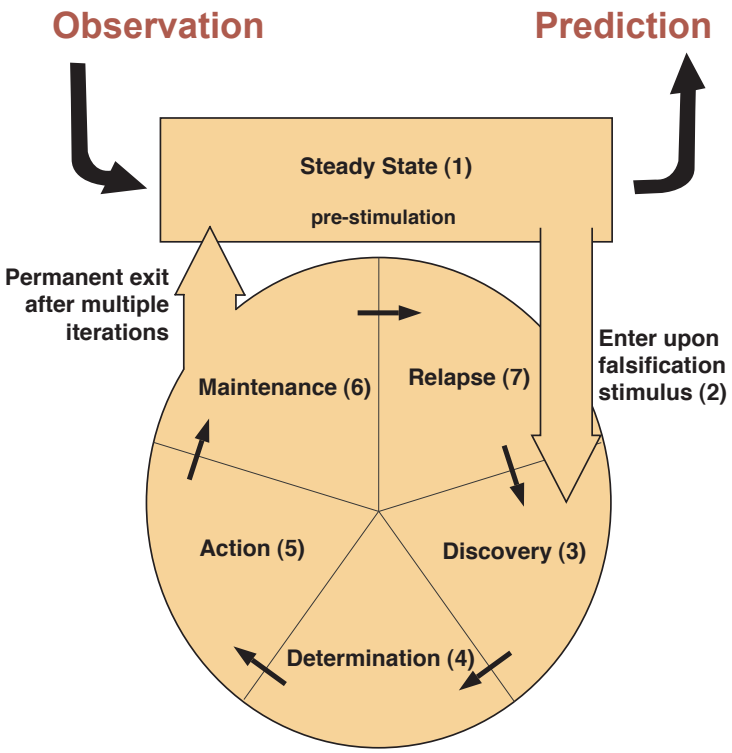


Figure 7. The change wheel—a model for persistent learning.

Successful persistent learning regulates the incremental, cumulative adaptations of data generalizations while ensuring the principal computer task remains viable and center stage. When training is self-supervised or autonomous, persistent learning schemes cannot simply resolve to a single optimal solution but must maintain and contrast competing solutions. Autonomous learning must also remain discretionary and stable.

3.1.1 The change wheel. In support of persistent learning, pattern recognition is not limited to training routines. Significant additional components of learning are falsification, discovery—the recognition of innovation—and consensus building. Our model for persistent learning—chosen for its completeness—is the Prochaska/DiClemente change wheel [81] shown in Figure 7. First developed to model human behavior modification, the change wheel is a feedback process with multiple stages. Persistent learning has seven stages: (1) steady state, (2) falsification, (3) discovery, (4) determination, (5) action, (6) maintenance, and (7) relapse.

Principal task:

- Steady state features the principal pattern recognition task. The principal task accomplishes prediction, and its resource demands dictate the pace of learning.

Passive learning stages:

- Falsification initiates learning. Upon a falsification stimulus—a contradiction to present beliefs—the change wheel is entered.
- Discovery is an investigative process. Tasks include the collection and evaluation of evidence that supports modifying the programs information model.
- Determination is a decision process. Here, recommendations are made regarding appropriate modifications to present beliefs.

Active learning stages:

- Action is the actual modification process where the information model is altered.
- Maintenance involves various sustainment actions. Tasks include minor adjustments to the information model, setting criteria for permanent exit from the wheel, and setting relapse criteria.
- Relapse handles the termination of an intervention cycle in preparation for a new cycle. Tasks may include archiving of data from the completed cycle and may also involve a partial or complete regression to the former information model.

Inquisitive pattern recognition operates in the passive stages of persistent learning—falsification, discovery, and determination—serving the investigative process and prioritizing opportunities for new experiences.

3.1.2 Designing intervention: persistent learning in phases. Inquisitiveness is part of an overall strategy to design intervention into computers—that is, to delegate the customization and maintenance of pattern recognition algorithms to the computer. Inquisitive pattern recognition is a transitional technology. The ultimate goal of on-line intervention is to construct robust PR algorithms that adapt autonomously.

Table 1. Phases of pattern recognition in relation to the stages of intervention.

	Phases of pattern recognition research		
Stage of Automated intervention	Stationary	Inquisitive	Persisent
Steady State	x	x	x
Discovery		x	x
Determination		x	x
Action			x
Maintenance			x
Relapse			x

In the interest of building stable systems, we assert it is best to design intervention into our pattern recognition algorithms in stages. The primary reason for phasing in persistent learning capabilities is to promote stability. It is counter productive to offer active intervention capabilities on-line without a formal discipline for recommending alterations to existing data generalizations. Inquisitive pattern recognition is to concentrate on the passive learning skills in order to minimize impact to the working order of the baseline recognition task. Consequently, we have grouped the intervention model into three phases of capability: respectively, stationary pattern recognition (SPR), inquisitive pattern recognition (IPR), and persistent pattern recognition (PPR). Each phase of pattern recognition capability incorporates ever more stages of intervention. See Table 1 for a summary. The classical SPR algorithm remains in stage 1, steady state—as depicted in Figure 8—so that all intervention processes must be performed off-line. Online intervention is designed into inquisitive and persistent PR algorithms though the IPR algorithm only implements the *passive* stages of the wheel, progressing only as far as stage 3, determination. Figure 9 illustrates inquisitive pattern recognition. The IPR algorithm responses to a stimulus for change and enters the intervention wheel. The remaining stages are delegated to off-line processes. The PPR algorithm implements both the passive and active stages of the change wheel as depicted in Figure 7 and, ideally, is capable of cycling through the change wheel multiple times.

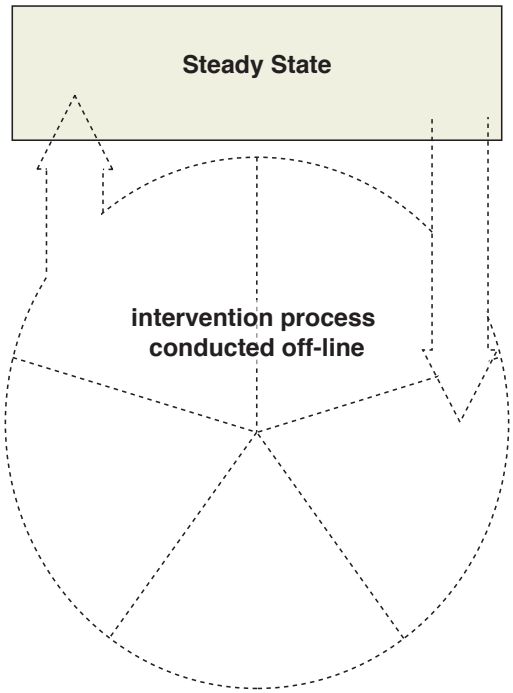


Figure 8. Off-line intervention in stationary pattern recognition.

Inquisitiveness is intended as a transitional capability between classical PR algorithms and persistent ones. The classical pattern recognition algorithm, or SPR algorithm, is assumed to operate within a stationary domain [27, 13]. Thus, the algorithm's knowledge model remains fixed over its lifetime. Conversely, an PPR algorithm should adjust its knowledge model in response to the dynamics of its domain [24]. In the transitional method of an IPR algorithm, intervention is merely "contemplated", that is to say information is collected and weighed to discover (1) if a domain shift has occurred and (2) if some adjustment to the PR algorithm is needed. The ultimate challenge of inquisitive pattern recognition is to determine—once a stimuli for change is detected—to what degree the algorithm should be augmented or modified.

3.1.3 Persistent Learning summary. The time is ripe for developing inquisitive pattern recognition. Besides the technological motivations discussed in Chapter II, the maturity of distributed processing [59] and parallel processing [45, 54] along with the low cost of memory offer researchers the greater computational resources in time and spaces. Addi-

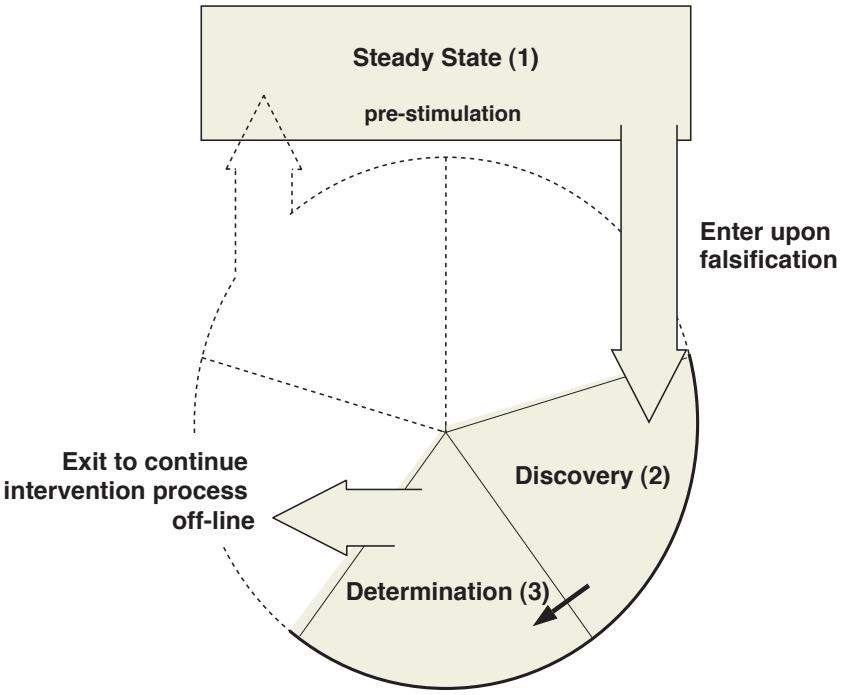


Figure 9. The passive modes of autonomous intervention incorporated by inquisitive pattern recognition.

tional resources will very likely be needed to fully develop persistent pattern recognition—particularly to enable the relapse capability—but for now we have sufficient resources to develop the passive intervention skills of inquisitive pattern recognition.

3.2 The concepts of Inquisitive Pattern Recognition

The goal of this research is to form a mathematical formalism for inquisitiveness, but such a formalism first requires a philosophy. The key components of inquisitive pattern recognition are falsification, confusion recognition, and relevancy testing. These three learning skills embody discretion, focus and prioritization—the virtues of doubt and intuition.

- Falsification identifies where in a feature set decision-making is known to be confused.
- Confusion recognition connects decision-related meaning to these regions in order to direct conflict resolution processes.
- Relevancy testing promotes important discoveries.

Together these learning skills compose the decision engine which proposes what unstable regions in an information model hold the most promise for yielding important new discoveries.

3.2.1 Background. When we first developed the concepts for inquisitive pattern recognition [61], we began by considering established models of human behavior modification with particular interest in what drives us to seek out new learning experiences. Here, we postulate that doubt and intuition are—arguably—key drives in humans. Together they spur on the innovative student, the learner who wishes a deeper understanding than his teachers have resources to give. As logical processes within critical thinking, doubt and intuition focus attention on resolvable trouble spots within an incomplete knowledge representation. A computational entity with the capacity to doubt is able to judge the incompleteness or inappropriateness of its prior knowledge. Intuition is fed bu the entity's resourcefulness at selecting viable opportunities to hone its knowledge while keeping its computation stable and relevant.

Doubt is a straightforward concept embodied in the cognitive science term falsification, the refutation of presently-held beliefs [9, 80]. Intuition is a fuzzier concept that balances two considerations in resolving new beliefs: (1) confusion recognition—the opportunity to define a beneficial pattern recognition experiment and (2) relevancy testing—the ability to identify evidence that is readily available to realize these benefits.

Doubt and intuition fit nicely into the change wheel: Doubt corresponds to falsification, and intuition corresponds to discovery and determination. In the next three sections, we will flesh out falsification, confusion recognition and relevancy testing as concepts and supported this discussion with a simple illustration of an object recognition problem.

3.2.2 Falsification. Doubt is the first step in unsupervised learning; it initiates the process. Cognitive science teaches us that humans predict through a set of beliefs, and we learn by first refuting present beliefs [9]. If a computer "doubts" itself, this means it has the capability of falsification—that is, the ability to autonomously recognize failings within its programs own knowledge representations. Consequently, learning benefits from having a focused goal and a definable context [30].

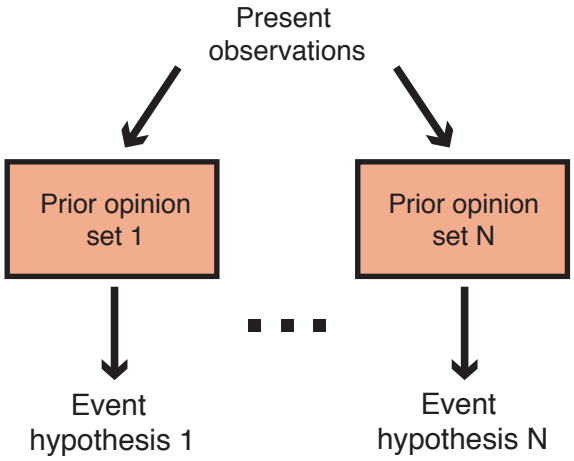


Figure 10. Event hypotheses are derived from disparate prior opinions. The hypotheses are then contrasted to flag apparent flaws in prior knowledge.

Falsification provides discretion. We envision persistent learning as a reaction to a stimulus, not a full-on full-time process. Falsification is intended to identify learning opportunities, concentrating on those portions of a program's knowledge representation in most need of attention.

3.2.2.1 Forming opinions for falsification.

Definition 2 *(Falsification). Falsification is a test that refutes a belief A through the use of the contrapositive* [80]*: Given beliefs A, B such that A implies B, if B is proven false then A is also false.*

Falsification is a test that requires prior opinions—generalizations, or restrictions, of a comprehensive opinion. The test refutes a comprehensive opinion by provong the simplier prior opinion false.

Definition 3 *(Comprehensive opinion). A comprehensive opinion is a pattern that can be restricted into prior opinions. Restrictions may affect prior knowledge or present observations or both.*

Definition 4 *(Prior opinion). Let A and B be patterns. If A implies B but B does not necessarily imply A, then, as we have defined, B is a* prior opinion *with respect to comprehensive opinion A.*

Note then if a prior opinion is proved false, by contrapositive logic, the comprehensive opinion is also proved false. Consider also that conflicting prior opinions apprehend weaknesses in the comprehensive—weaknesses that should be addressed through augmentation, either additional prior knowledge or supplementary observations or both.

3.2.2.2 Complexity testing. Falsification may be used to gauge the effective complexity of a data generalization. Complexity is a relative measure that conveys the ability of a knowledge representation to capture the underlying data structure of task-related observations. When a data generalization is too complex, it over-fits training samples and memorization occurs[1]. Falsification offers self-supervised complexity testing as unlabeled observations turn up in unanticipated regions of a feature set or in distributions other than what was hypothesized at design.

3.2.2.3 Belief testing. Falsification can also be used to test beliefs, or logical groupings within a knowledge model. As shown in Figure 10, new observations are collected and considered in the context of each prior opinion, forming respective event hypotheses. If hypotheses for an event vary significantly, we can reason that either certain beliefs are not appropriate to present observations or they are obsolete in light of recent evidence. Ideally, we want to be able to localize and replace inadequate beliefs—customizing knowledge without losing what still works.

3.2.2.4 An illustration of falsification. Falsification algorithms are knowingly naive—containing incomplete information structures that break conspicuously so that the computer can examine their fallacy in action. A falsification algorithm requires (1) disparate prior opinions and (2) a measure quantifying significant confusion among either prior opinions (for complexity testing) or event hypotheses (for belief testing). Falsification

[1]Recall from Section 2.1.3, memorization is undesirable as the resulting representation is conformed tightly (in the extreme, point-wise) to the training observations but rarely proves competent at recognizing new data (even those adjacent to the training observations) [83].

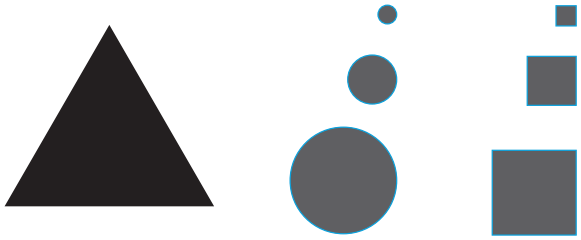


Figure 11. The target object and two representational sets—one comprised of circles, the other of squares.

is powerful in unsupervised learning because, though new observations are not labeled, the relationships between prior opinions can be specified.

Here is a simple illustration of an object reconstruction problem. For this illustration, prior opinions are contrasted in the falsification process. Consider the triangle of Figure 11. The triangle Y is a real-world object that cannot—for the purposes of this illustration—be represented completely. To reconstruct the triangle Y, we are given two separate basis sets—a set $\mathcal{A}$ of circles and a set $\mathcal{B}$ of squares—to fill in the two-dimensional space. As depicted in Figure 11, let the circle basis set $\mathcal{A}$ contain three circle sizes, and the square basis set $\mathcal{B}$ contain three square sizes.

$$\begin{aligned} \mathcal{A} &\rightarrow \{\text{small circle},\ \text{medium circle},\ \text{large circle}\} \\ \mathcal{B} &\rightarrow \{\text{small square},\ \text{medium square},\ \text{large square}\} \end{aligned}$$

The two basis sets are used to form two separate prior opinions $\theta_{\mathcal{A}}$ and $\theta_{\mathcal{B}}$. See Figures 12 and 13. The opinions of the triangle are estimated by filling the two-dimensional space—one with the circle basis set, the other with the square basis set[2]. An OR operation is performed on the two sets to merge the space filled by the circles so that $\theta_{\mathcal{A}} = \cup_{n=1}^{N} A_n$. The same operation is applied separately to the squares, $\theta_{\mathcal{B}} = \cup_{m=1}^{M} B_m$.

[2]For this example, we are restricted to using only one basis set at a time. This is a realistic limitation paralleling restrictions in imaging methods used to capture real world objects. In a simplistic way, the circles correspond to imaging onto film and the squares correspond to scanning the object and storing an image matrix.

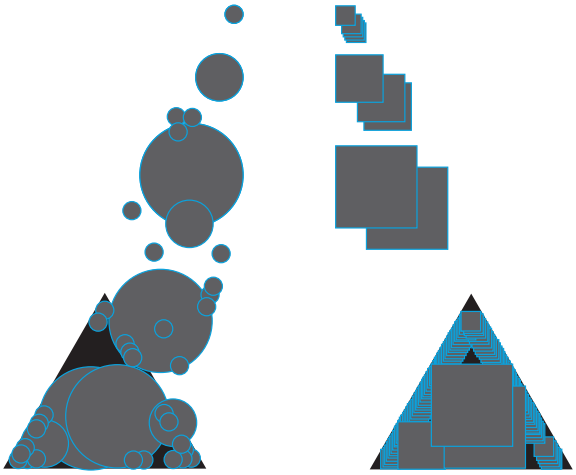


Figure 12. Designing the prior opinions.

Due to the limitations of the basis sets, neither the circle-based prior opinion nor the square-based prior opinion captures the entire essence of the triangle. The missing essence, as seen in Figure 13, is the error set. For this illustration, let us form each error set, $err_{\mathcal{A}} = Y \ominus \theta_{\mathcal{A}}$ and $err_{\mathcal{B}} = Y \ominus \theta_{\mathcal{B}}$, by applying an XOR operation to the triangle and each prior opinion in turn. Often, we do not have perfect knowledge of the real world object; so, normally, we cannot calculate the error set directly. Instead, here is where we perform falsification as the best we can do is evaluate the confusion between opinions. In this example, the prior opinions are contrasted by taking their symmetric difference[3] to form the falsification subset of the confusion set $\text{Falsify}(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}) = \mathbb{C}^{-,+}(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}) \cup \mathbb{C}^{+,-}(\theta_{\mathcal{A}}, \theta_{\mathcal{B}}) = \theta_{\mathcal{A}} \ominus \theta_{\mathcal{B}}$ as described in the Appendix Section A.1 and shown in Figure 14.

The falsification set is where contrapositive logic allows us to state with certainty that the object is not well understood. Again, falsification identifies the regions of feature set where decision making is known to be poor as identified by $\text{Falsify}(\theta_{\mathcal{A}}, \theta_{\mathcal{B}})$, and in these regions, both decision-making models $\theta_{\mathcal{A}}, \theta_{\mathcal{B}}$ are incomplete.

Confusion is not error. Nor is it a complete measure of uncertainty, as confusion set does not include all subsets where it is likely that decision making is poor. But it is a good

[3]Symmetric difference is a binary set operation that selects elements that belong to only one of the two sets. Logically the operation is expressed $A \ominus B = (A \cap \neg B) \cup (\neg A \cap B)$.

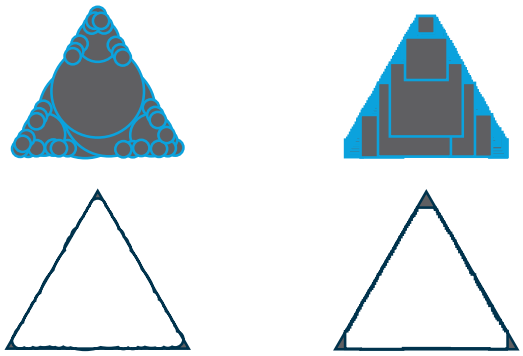


Figure 13. The two prior opinions and their respective error sets. An XOR operation is applied to the triangle and each prior opinion in turn to form the error sets.

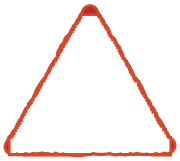


Figure 14. The falsificaltion subset of the confusion set is formed by applying an XOR to the prior opinions.

starting point, and with a little more manipulation, a better estimate of the error set and thus a better understanding of the original object can be formed.

The tools for this additional manipulation are confusion recognition and relevancy testing. Respectively, these persistent learning skills partition the confusion set and prioritize the refinement of the partitions.

3.2.3 Confusion recognition. Confusion recognition interprets features in confusion space by leveraging the known idiosyncrasies of the applied basis sets. The recognition process divides confusion space and associates meaning with each piece.

3.2.3.1 Bounding the learning opportunity.

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."

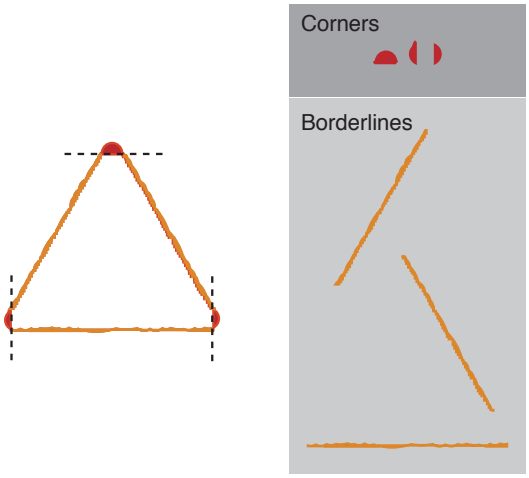


Figure 15. Confusion recognition. The falsification set of Figure 14 can be partitioned into six separate regions. These regions have then been classified into two categories, corners and borderlines.

Sir Arthur Conan Doyle
The Sign of Four [1890]

Inquisitive pattern recognition that stops at falsification is rather weak. Doubt may initiate the learning process but, if the cause of confusion is not known, then context for customization is unclear and learning flounders unregulated. As a result, the conclusions that can be drawn from mere falsification are severely limited and often regressive [80].

When forming new experiences, the cause of confusion is decidedly not known [30]. Instead, one might go through the process—as the fictional Sherlock Holmes does in The Sign of Four—of eliminating potential causes. Take, for example, the three-class problem of separating apples from oranges from tangerines. If we can eliminate the choice of—say—apples as contributing to the confusion of one region, we can narrow down the context to a choice between oranges and tangerines and bound the learning opportunity that region constitutes.

3.2.3.2 The association of decision-related meaning. The true power in confusion recognition is not found in associating task-related meaning, as in our apples-to-oranges-to-tangerines example, but in associating decision-related meaning to confusion

space. In the modeling of decision boundaries, confusion occurs in transitional regions between what is believed to be true and what is believed false [97, 50]. Transitional regions do not all look alike and (This is key!) are as much a function of the basis sets we choose to model a decision as the form of the decision itself. If this were not true, the confusion space of Figure 14 would be null space.

Confusion recognition separates confusion space into distinct regions that may benefit from being resolved in different ways. For instance, in Figure 15 we have divided regions of confusion space into two categories—corners and borderlines—and assert that the confusion associated with corners is significantly different from the confusion associated with borderlines. From this assertion, we can infer that any given strategy to resolve the confusion within these disparate regions will yield disparate results.

Confusion recognition enables the capture of new experiences. It serves to bound unsupervised learning by associating meaning to new information that is either task-related or—more generally—decision-related. When learning is properly bounded, it is more efficient as training requires fewer exemplars and fewer features. This lessens demands placed on the collection of new training data. Moreover, fewer features enhance the probability of reducing confusion. [13, 83]

3.2.4 Relevancy testing. The last and crucial step of inquisitive pattern recognition is relevancy testing—assigning priority to learning opportunities. Before attempting to resolve confusion, it is best to weigh the benefits and risks posed by each opportunity—pursuing only those where there is both sufficient evidence and justifiable need. In the end, relevancy testing determines what, if any, action should be taken to refine a knowledge model. Relevancy testing calls for an excellent understanding of how the limitations of the applied basis sets hinder the goals of the computer task. Again, take our object reconstruction illustration: Note that the corners of the triangle are not engulfed by the confusion space but that the borderlines essentially are. Here it is more important to determine that the corners are poorly modeled. This declaration requires not only an understanding of representational shortcomings—acute corners are difficult to fill—but also an appreciation for the task at hand, i.e., object reconstruction.

Relevancy testing must cross-index representational limitations with task goals, and this is where our effort to build task-savvy, decision-oriented knowledge bases comes in. Within these knowledge bases, we are endeavoring to capture the idiosyncrasies of our favorite representational tools—tools that include data pyramids, wavelets, and neural networks. These knowledge bases provide the reasoning resources (1) to cross-index specific task-related knowledge with decision-making expertise and (2) to infer from this mapping which learning opportunities constitute important discoveries.

3.2.5 IPR summary. Falsification triggers learning when the refutation of one belief can be combined with other knowledge to further the task at hand [80]. Inquisitive pattern recognition seeks to harness the power of falsification by augmenting the process with confusion recognition and relevancy testing. These additional skills provide a means to form new experiences by bounding and regulating unsupervised learning opportunities. Supporting research involves the generation of knowledge bases that grasp the mechanics of mathematical representations and capture inherent implications to decision making.

3.3 Summary

Automated intervention strategies delegate customization and maintenance tasks to the computer. Designing intervention opens up the computer's potential for innovation—giving the computer the capacity to absorb new experiences, to learn practical knowledge on its own—awakening the computer to it's role as team member, not just team tool. Toward designing a computer capable of innovation, concepts of inquisitiveness have been derived from models of human behavior modification with particular interest in what drives humans to seek new experiences. This chapter has asserted that doubt and intuition are key incentives spurring on the innovative student. As logical processes, doubt and intuition focus attention on resolvable trouble spots within an incomplete knowledge representation.

- Doubt is the capacity of a computational entity to judge the incompleteness or inappropriateness of its prior knowledge.
- Intuition is the entity's resourcefulness at selecting viable opportunities to hone its knowledge while keeping its computation stable and relevant.

Customizing data generalizations requires discipline to sustain clear, complete, and timely representations. In keeping, pattern recognition must be augmented in both design and application methodology to ensure the collection of relevant new information in an effective manageable manner. This dissertation advocates a shift in the PR discipline away from training single "optimal" solutions from static data sets. For one, the paradigm that the single best solution is found at design does not serve dynamic domains. Secondly, too many training/evaluation methods fail to explicitly state the range and resolution of pattern recognition solutions, to investigate potential flaws in interpolations or to post warnings on extrapolations. It behooves the pattern recognition to consider the design and maintenance of multiple working "opinions", strategies to recognize and resolve incompleteness in representation and to affirm enduring trends throughout a program's lifecycle and domain.

IV. Measuring the incompleteness of information

This chapter presents the Theory of Confusion—quantifying and qualifying incompleteness of information. This theory handles different types of confusion among classifiers and yields the regions in a feature set where confusion occurs. Using measure theory, we propose confusion set and 4-value logic in order to characterize the apparent incompleteness of information—particularly to separate the extrapolations of an empirically-derived function from its interpolations.

In this chapter, we present formal definitions of data, information and knowledge. In the development of these definitions, we draw from the writings of Charles S. Peirce and Richard T. Cox—respected figures in logic and entropy theory. Next, we discuss the difference between error and confusion and formalize this difference via measure theory. Finally, we conclude the chapter by introducing and demonstrating 4-value logic.

4.1 Experience and information

In pattern recognition, there is a vague consensus that knowledge is something more than information, and information is something more than data. To develop a formal definition of information, let us consult Charles S. Peirce[1] and his later writings on the nature of experience. In his 1902 manuscript "MS L75: Logic, Regarded As Semeiotic" [76], Peirce lumped experience into three categories—fact, representation, and essence.

1. Fact is the external experience—stimulus or action—an ordered, windowed observation or stimulant on the real world [76].
2. Representation is an internalization of experience unique to each computational entity, one internal conceptualization in an infinite realm of possibilities [76].
3. Essence is the communal or shared experience—a symbol affirmed by many and enduring [76].

[1]Charles Sanders Peirce (1839-1914) was a engineer whose contributions to logic include 3-valued logic [29], Boolean algebra [57], and quantification logic [75]. A chemist and geodesist by profession, his life's work was devoted to the study and research of logic, including the theory of signs [16] from which we draw for our discussion of experience.

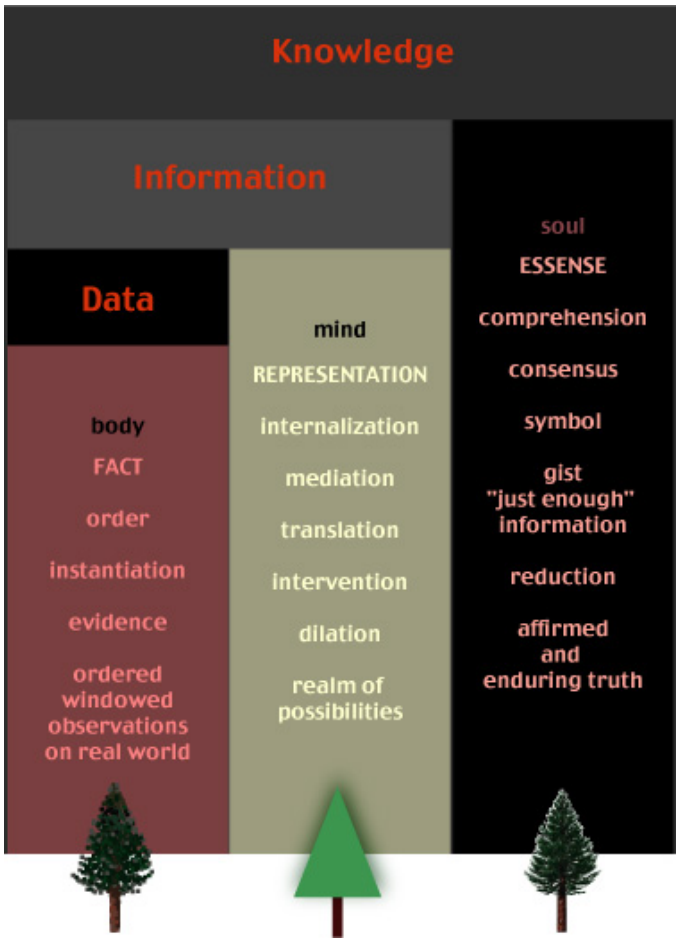


Figure 16. The categories of experience.

Figure 16 conveys the concepts behind these categories. The figure also illustrates how we believe Peirce's categories of experience relate to data, information and knowledge. Data are a collection of facts, information is a collection of facts and representations, and knowledge is a collection of facts, representations, and symbols.

4.1.1 Quantifying experience via probability and bearing. An experience arises in the presence of an event, an observer and an internal response to the event by the observer [76]. To represent a sign, i.e. a fundamental element of observer's experience, we propose the triad $\{x, q, \ell\}$ where x is the event, q is a query representing a perspective of the observer, and ℓ is an assertion, the observer's internal response to the query q given event x.

Definition 5 *(Event). An event x is an instantiated element from the set of all possible events.*

Definition 6 *(Query). A query q asks something about an event and is represented by the set of assertions that answer that query over the set of all possible events.* [21]

Definition 7 *(Assertion). An assertion ℓ says something about an event in response to a query and is represented by the set of events for which the assertion gives a truthful response to the query.* [21]

This triad formalizes Peirce's notion of a sign by drawing on Richard Cox's essay "On inference and inquiry, an essay in inductive logic" [21] in which Cox describes the parallel logics of queries and assertions. The important connection we make here is that an observer can be represented by a set of queries.

From Cox, we learn that a query may assigned a bearing, i.e., a measure of relevancy, with respect to an assertion or event. Let bearing map to the interval $b \in [0, 1]$ where $b(q) = 0$ for a query asked in vain and $b(q) = 1$ for a query that is real, or wholly relevant. Assertions are assigned a probability on the interval $p \in [0, 1]$ where $p(\ell) = 0$ for an assertion that is false and $p(\ell) = 1$ for an assertion that is true. [21]

Thus, given a set of queries $\mathcal{Q}$ and a set of assertions $\mathcal{L}^*$, there exists a mapping $\zeta : \mathcal{Q} \times \mathcal{L}^* \to [0, 1] \times [0, 1]$ from the 2-tuple of a query and an assertion to a 2-tuple of a bearing and a probability, $\zeta(q, \ell) = \{b, p\}$.

One way to look at the differences between bearing and probability is to consider decision boundaries. Probability is constant along a decision boundary while bearing varies. As we move away from the decision boundary, bearing remains constant while probability varies.

According to Cox, bearing and probability are quantities that have the same properties only the former is used to order queries and the latter orders assertions [21]. A query may be represented by a set of assertions [33], and probability allows us to order the set of assertions that respond to the query. For instance, if I was a hat check attendant, over the course of the work day I might ask many people what color their coat is. Based on the apparent charateristics of each person, I form a list of likely responses, a set of colors where each color is assigned a probability. If the color pink is assigned a high probability, then likely I am querying a young girl. We can make this inference because, in our culture, the color pink is associated with clothing of young girls more so that with the clothing of young women, mature women, and males in general. An assertion may also be represented by a set of queries [33], and that set may be ordered by bearing. Pink is an asssertion associated with a host of questions including "What is the color of your coat?" and "What is your favorite color?" each of which assigned a bearing again based on context. Context, then, can be represented by a ordering of queries and an ordering of assertions. The ordering of queries captures mainly the interrogator's point of view while the ordering of assertions captures the responder's point of view.

In Chapter IV, we present an example of a probability-bearing pair, the quantities of veracity and experience which we use to explore certain dynamics between an interrogator and an expert under evaluation. Veracity is a special quantity of an assertion's probability, expressing an expert's conviction or belief in an assertion. Experience is a special quantity of a query's bearing, denoting the accumulation of experiences relevant to a query and expressing the fitness of an expert to answer the query in light of those experiences. To ask an expert a question where the expert has no experience is to ask that question in vain. Questions that asked in vain are important tools of an interrogator. If an interrogator knowingly asks a question in vain and the responder strongly asserts an answer, then the interrogator can reasonably infer that the responder is an arrogant pretender and not a

true expert. Note that the interrogator makes this inference of arrogance based on both veracity and experience (i.e., the interrogator has discerned that veracity is represented as high where experience is low). A judgement of arrogance cannot be made without both quantities.

4.1.2 Defining experience in terms of set theory. Let us present the relationships of event, assertion, and query sets using the formal definitions of relations and orderings.

Definition 8 *(Relation). Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets and let $\mathcal{X} \times \mathcal{Y}$ be their Cartesian product. Any subset $\mathsf{R} \subset \mathcal{X} \times \mathcal{Y}$ is called a* relation. *If R is a relation, the set of all elements x that occur as first members of pairs $(x, y) \in \mathsf{R}$ is called the domain of R, denoted $\mathcal{D}(\mathsf{R})$. The set of second members y is called the range of R, denoted by $\mathcal{R}(\mathsf{R})$.* [47]

Definition 9 *(Ordering). The special relation $\mathsf{R} \in \mathcal{X} \times \mathcal{X}$ is called a* ordering *of a set $\mathcal{X}$ if and only if, given $a, b, c \in \mathcal{X}$, (1) R is reflexive ($a\mathsf{R}a$ for every $a \in \mathcal{X}$) and (2) R is transitive (whenever $a\mathsf{R}b$ and $b\mathsf{R}c$, then $a\mathsf{R}c$).* [47]

Definition 10 *(Partial Ordering). The special relation $\mathsf{R} \in \mathcal{X} \times \mathcal{X}$ is called a* partial ordering *of a set $\mathcal{X}$ if and only if, given $a, b, c \in \mathcal{X}$, (1) R is reflexive ($a\mathsf{R}a$ for every $a \in \mathcal{X}$), (2) R is transitive (whenever $a\mathsf{R}b$ and $b\mathsf{R}c$, then $a\mathsf{R}c$), and (3) R is antisymmetric (if $a\mathsf{R}b$ and $b\mathsf{R}a$, then $a = b$).* [47]

Definition 11 *(Total Ordering). The special relation $\mathsf{R} \in \mathcal{X} \times \mathcal{X}$ is called a* total ordering *of a set $\mathcal{X}$ if and only if, given $a, b, c \in \mathcal{X}$, (1) R is reflexive ($a\mathsf{R}a$ for every $a \in \mathcal{X}$), (2) R is transitive (whenever $a\mathsf{R}b$ and $b\mathsf{R}c$, then $a\mathsf{R}c$), (3) R is antisymmetric (if $a\mathsf{R}b$ and $b\mathsf{R}a$, then $a = b$), and (4) Relation is for any two elements $a, b \in \mathcal{X}$, either $a\mathsf{R}b$ or $b\mathsf{R}a$.* [70]

Let $\mathcal{X}$ be a nonempty event set, let $\mathcal{L}$ be a nonempty label set, and let $\mathcal{Q}$ be a nonempty set of queries. Given the formalism we've discussed above, we can now define the three levels of experience—fact, representation and essence—starting with fact.

Definition 12 *(Fact). A fact is an ordered pair $(x, \ell) \in \mathcal{X} \times \mathcal{L}$. The first object $x \in \mathcal{X}$ is an observed or enacted event; the second object $\ell \in \mathcal{L}$ is a label.*

Let $\mathcal{L}^* \supset \mathcal{L}$ be a superset of $\mathcal{L}$. We make a mild distinction here between a label and an assertion: A label remains linked with the instantiated event while an assertion is assigned a probability and projected forward in the decision process.

Definition 13 *(Representation) A* representation R *is a subset of* $(\mathcal{X} \times \mathcal{L}) \times (\mathcal{X} \times \mathcal{L}^*)$ *such that representation* R *is the explicit relation that matches an event to an assertion.*

Experiences of general utility evolve into symbols which are represented as the ordered 3-tuple $(x, \ell, s) \in (\mathcal{X} \times \mathcal{L}^*) \times \mathcal{Q}$.

Definition 14 *(Essence) Essence* Ω *is the explicit relation defined on a set of symbols and matches a set of assertions to a query.*

From the three levels of experience described above, we form the definitions of data, information and knowledge.

Definition 15 *(Data) Data* $\mathfrak{D}$ *is a set of facts such that* $\mathfrak{D} \subset \mathcal{X} \times \mathcal{L}$*. Data are a sample of ordered pairs, not a relation.*

Definition 16 *(Information) Information* $(\mathfrak{D}, \mathcal{R})$ *is a collection of facts and representations where* $\mathfrak{D}$ *is a set of facts and* $\mathcal{R}$ *is a set of representations.*

Definition 17 *(Knowledge) Knowledge* $(\mathfrak{D}, \mathcal{R}, \Omega)$ *is a collection of facts, representations, and essence.*

Ordering a set of queries imposes an ordering on a set of labels or assertions; and, in turn, the ordering on a set of labels implies an ordering on a set of events. This property allows us to form knowledge from a set of facts.

4.1.3 Affirmation in the accumulation of experience. Multiple observers or competing hypotheses are fundamental to the accumulation of experience. Essence in experience arises from the affirmation of persistent truths, and affirmation explicitly implies the consultation of a self and an other: two (or more) computations that uniquely interpret a shared experience.

A consultation requires at least two discernible entities, each capable of internalizing the experience and providing independent feedback about that experience. For there to be a

true self and a true other, each computational engine must employ an internal representation that is innovative—unique from its peers in patterning the same experience. If I see a shooting star streak across the night sky, my first instinct is to turn to my neighbor and say "Did you see that?" A positive response lends greater confidence to my observation of a fleeting, unexpected stimulus and the moment passes. A negative response, however, forces me to pause, take stock of the situation and discern the cause of the discrepancy. Say the fault was in one's own observation: for instance, a perceived streak of light could be the manifestation of a migraine headache. One learns upon negative reinforcement to be more cautious in categorizing similar stimuli.

The recognition and advancement of innovation is an important part of affirmation. Majority rule dumbs down the recognition process by degrading into group think—the tendency to gloss over what is not easily understood and cling to long-held or majority-sponsored beliefs—slowing down the group's responsiveness to domain changes. It is constructive to explore other, more expressive forms of consensus building.

4.1.4 Summary. Learning is the active, cumulative, incremental process of converting observations into experience, and a learning algorithm masters new experiences through training, innovation, and consensus building. From facts, a training algorithm derives representations. Learning can stop there, but a training algorithm—unaware of its own consequence—must remain under close scrutiny to prevent the indiscriminate application of its derived representations. A learning algorithm—in the full sense of "learning"—seeks more than information; it seeks knowledge, or signs, the common threads that link over multiple observations or among multiple observers.

An inquisitive learning algorithm strives to capture the apparent context in which the trained generalization appropriately serves. One important aspect of context hinges on the interplay of assertions and queries. Cox foot-stomped the connection in his essay "On inference and inquiry" [21], and Turning noted the connection himself in pondering the role of the interrogator [99]: An interrogator must judge whether a query is a relevant in addition to judging whether an expert's answer to the question is true [99].

We will expound on dual measures of experience ourselves in the next chapter on expertise logic. For now, we shall continue our discussion of incompleteness in information and the presentation of the Theory of Confusion.

4.2 The Theory of Confusion

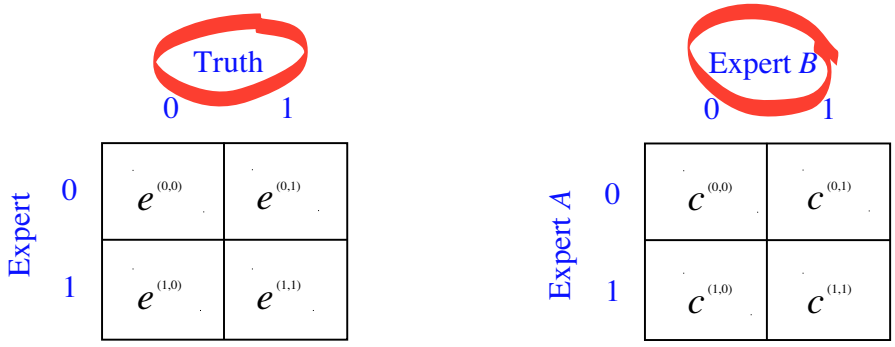
The Theory of Confusion measures the degrees to which information is substantiated by an authoritative source, prior experience or reasonable induction. To define incomplete information, let us start from the axiom that facts[2] are useful only if ordered and ordered appropriately. This ordering—a relation in the strict sense—may be explicit or implicit. If the relation is implicit, the collection of facts constitutes data. If the relation is explicit, the collection constitutes information. Thus, information is a collection of facts and an explicit relation defined on those facts. [62]

Incomplete information implies a lack of experience with the facts and/or a lack of expertise such as a poorly rendered relation. When the ordering of an incomplete set of facts is generalized, the resulting relation is shaped by an odd mix of experience, induction and chance. To quantify the incompleteness of information it is important to (1) quantify the extent to which a collection of facts adequately constrains the induction of an explicit relation and (2) identify the subset of unsubstantiated facts (experiences that have not been truthed by an authoritative expert) within the domain of an underconstrained relation. Additionally, it is instructive to bound what information is currently missing but would be particularly useful if it were fleshed out.

4.2.1 Distinguishing confusion from error. In pattern recognition literature, the term confusion matrix is used interchangeably with error matrix [55]. Here—as illustrated in Figure 17—we make a distinction:

Definition 18 *(Error).* Error *quantifies the discrepancy between truth and an expert's opinion.*

[2]See Section 4.1 for a discussion of facts and their relationship to information.



(a) Error matrix (b) Confusion matrix

Figure 17. The structure of an error matrix and a confusion matrix for two-class classifiers.

Definition 19 *(Confusion).* Confusion *quantifies the difference between the opinions of experts.*

We assert there are benefits to delineating error and confusion. In strict interpretation, measures of error require supervision by an omnipotent authority. As such, error measures are fully realizable in applications where truth is known completely—for instance, in certain compression and reconstruction applications. Measures of confusion are less rigorous as they involve consultations among peer experts. Confusion measures are realizable in autonomous systems featuring collaborative computations and—through disciplined interpretation—edge pattern recognition closer to the elusive capability of self-evaluation. [61]

4.2.2 Quantifying incompleteness in information. The strategy administered for collecting data (both the sampling rate and observation field) and succeeding training methods limit the range, resolution and complexity of resolved data generalizations. It is important to express these consequences in order to speak to the incompleteness of a data generalization and to ensure appropriate application by its end user. [62, 61]

A data generalization, as defined in Section 2.1.1, is an explicit relation or ordering designed from a sample of experiences to cover a larger set of new experiences. When forming a data generalization, instances of truth are collected and—through training—

broadened to cover other instances. By interpolating between all that is known, a data generalization offers an opinion on what is unknown. [62]

The success of a data generalization hinges on its ability to interpolate operational data accurately. Interpolations occur near training data where there is reason to presume operational data will be collected. When new data are collected where a mapping is hypothesized to be interpolating, that mapping is suspect but may be trusted if its results are not reasonably contradicted by other expert opinions. If interpolation is risky, extrapolation is downright dangerous. Extrapolations occur "far" from training data where new data are not expected. If operational data are collected where a mapping is hypothesized to be extrapolating, the mapping is not representative of any real trend and computational results are based capriciously on chance. [62]

A data generalization is best evaluated in consideration of its operational set, as opposed to its training set. Such consideration must balance the two key uncertainties in generalizations: (1) the *completeness* of the training set in representing the operational set and (2) the *fitness* of a generalization in ordering the hypothesized operational set. [62]

To quantify incompleteness in a data generalization, confusion is a more appropriate measure than error. Confusion measures are particularly useful for investigating the interpolations and extrapolations of a data generalization. Interpolation implies the consideration of an unlabeled data point between labeled data. Therefore, as defined above, error measures may not express discrepancies in the interpolated regions of a feature set, but confusion measures may. Certainly, the success of a data generalization is directly proportional to its behavior in interpolated regions, and—by default—confusion becomes the appropriate measure to convey the stability and completeness of the model within these regions. [62]

4.2.3 Background in measure theory. The incompleteness of information is defined formally using measure theory. As presented in Section 4.1, we have defined information as a collection of facts and representations. To quantifying confusion, one identifies apparent "holes" in this collection and determines whether the incompleteness of information is due to inadequate facts or inadequate representations.

4.2.3.1 Measure Theory. Let $\mathcal{X}$ be a nonempty feature set. A σ-algebra $\mathfrak{B}$ is a collection of subsets of $\mathcal{X}$ that forms a Boolean algebra with the added property that, for each countable collection of subsets $\{S_n\} \subset \mathfrak{B}$ then

$$\bigcup_n S_n \in \mathfrak{B}.$$

The pair $(\mathcal{X}, \mathfrak{B})$ is called a *measurable space* [42, 88]. A subset $E \subset \mathcal{X}$ is called a measurable set (i.e., measurable with respect to $\mathfrak{B}$) if E is an element of $\mathfrak{B}$. [88]

A *measure* is a non-negative set function[3] defined on a σ-algebra of subsets that must satisfy the two properties

$$\mu(\emptyset) = 0 \quad (1)$$

$$\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mu(E_i) \quad (2)$$

for any countable collection $\{E_i\} \in \mathfrak{B}$ of disjoint measurable sets. The second property says that a measure is countably additive. It follows that a measure is also finitely additive in that

$$\mu(\bigcup_{i=1}^{N} E_i) = \sum_{i=1}^{N} \mu(E_i) \quad (3)$$

for all disjoint measurable subsets E_i belonging to $\mathfrak{B}$ since we may set $E_i = \emptyset \ \ \forall i > N$. [88]

A measurable space $(\mathcal{X}, \mathfrak{B})$ together with the measure μ defined on $\mathfrak{B}$ yields the triple $(\mathcal{X}, \mathfrak{B}, \mu)$ which is called a *measure space*. If $\mu(\mathcal{X})$ is finite, then $(\mathcal{X}, \mathfrak{B}, \mu)$ is called a finite measure space. If $\mu(\mathcal{X}) = 1$, then the finite measure space is called a probability space. A measure space $(\mathcal{X}, \mathfrak{B}, \mu)$ is called a σ-finite measure space if there is a sequence $\{\mathcal{X}_n\}$ of sets in $\mathfrak{B}$ such that

$$\mathcal{X} = \bigcup_{n=1}^{\infty} \mathcal{X}_n \quad (4)$$

and $\mu(\mathcal{X}_n) < \infty$ for each n. Hence, every finite measure space is a σ-finite measure space (but not vice versa). [88]

[3] A set function is a function whose input is a set and whose output is a real number.

4.2.4 Integral measure. Consider the special case of the measure generated from an integral. Let $\mathcal{X} = \mathbb{R}^d$ and let m be the Lebesgue measure on $\mathbb{R}^d$. Let $f \in \mathsf{L}^1(\mathcal{X}, \mathbb{R})$ so that f is a Lebesgue integrable function. Given any measurable subset $E \in \mathcal{X}$ define the measure μ in terms of an integral

$$\mu(E) = \int_E f(x)\, m(dx). \tag{5}$$

It can be shown that μ satisfies the conditions in equations (1) and (2) above. [88]

4.2.5 Classifiers. Over the next several sections, we shall define incomplete information in terms of classifiers. A classifier is a special data generalization, that is, a relation designed from a labeled training set (i.e., a sample of experiences) to cover a larger operational set (i.e., new experiences seen in the application, or operational use, of the generalization) [62].

In measure theory, a classifier is a special relation. Let us formally define a relation and then define the terms function and classifier.

Definition 20 *(Relation). Let X and Y be two sets and let $X \times Y$ be the Cartesian product. Any subset $\mathsf{R} \subset X \times Y$ is called a* relation. *If R is a relation, the set of all elements x that occur as first members of pairs $(x, y) \in \mathsf{R}$ is called the domain of R, denoted $\mathcal{D}(\mathsf{R})$. The set of second members y is called the range of R, denoted by $\mathcal{R}(\mathsf{R})$.*

Definition 21 *(Function). A* function *is a special relation. Define function F as a set of ordered pairs (x, y) where no two distinct pairs have the same first member. That is, given $x \in \mathcal{D}(F)$ and $y, z \in \mathcal{R}(F)$, if $(x, y) \in F$ and $(x, z) \in F$ then $y = z$.*

Definition 22 *(Classifier). Given a nonempty feature set $\mathcal{X}$ and a nonempty label set $\mathcal{L}$, a classifier C is defined as a correspondence between $\mathcal{X}$ and $\mathcal{L}$, and we write this as $C : \mathcal{X} \to \mathcal{L}$.*

4.2.6 Measurable classifiers. Let C be a classifier defined on a feature set $\mathcal{X}$ with output label set $\mathcal{L}$. If an input/output pair $(x_k, \ell_k) \in C$ then the feature vector $x_k \in \mathcal{X}$ and label $\ell_k \in \mathcal{L}$. Assume $(\mathcal{X}, \mathfrak{B})$ is measurable space.

Definition 23 *(Measurable classifier). We say classifier C is a measurable classifier if the inverse image of each label $\ell \in \mathcal{L}$, denoted $C^{-1}[\{\ell\}] = \{x \in \mathcal{X} : C(x) = \ell\}$, is a measurable set in algebra $\mathfrak{B}$.*

Let $\mathcal{L}$ be a finite set of N labels. We assume there exists at least one feature vector that maps to each label.

Assuming crisp logic, classifier C partitions $\mathcal{X}$ into N disjointed, nonempty subsets $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$. Each subset $\mathcal{X}_i$ corresponds to a label ℓ_i. Additional properties for the N-element collection of subsets in $\mathcal{X}$ are: (1) every subset is measurable with respect to $\mathfrak{B}$, i.e., $\mathcal{X}_i \in \mathfrak{B}$, and; (2) the collection is exhaustive, i.e.,

$$\bigcup_{i=1}^{N} \mathcal{X}_i = \mathcal{X} \tag{6}$$

and (3) mutually disjoint, i.e., $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for all $i \neq j$.

Given $\mathcal{X} = \mathbb{R}^d$, a measurable space $(\mathcal{X}, \mathfrak{B})$, a density function $f \in \mathsf{L}^1(\mathcal{X}, \mathbb{R})$ and a measurable classifier C, define a measure μ as in Equation 5. Assume that f is a density function, i.e., f is nonnegative and

$$\mu(\mathcal{X}) = \int_{\mathcal{X}} f(x) m(dx) = 1.$$

Therefore, given a label $\ell \in \mathcal{X}$, the measure $\mu(C^{-1}[\{\ell\}])$ takes the form

$$\mu(C^{-1}[\{\ell\}]) = \int_{C^{-1}[\{\ell\}]} f(x) m(dx). \tag{7}$$

In this treatment, density function f is the weighing on the population of feature set $\mathcal{X}$; i.e., it bounds and weighs the relevant elements of that feature set with respect to the population.

4.2.6.1 Error rates between truth and a measurable classifier. Truth mappings are definitive classifiers, the ultimate authority on the partitioning and labeling of a feature set. Let truth mapping T be defined on $\mathcal{X}$ with the same output label set $\mathcal{L}$. In-

dividually, the inverse images of classifier C and the truth mapping T effectively partition feature set $\mathcal{X}$ into N subsets. (Assume that T takes on each label for now.) Classifier C and truth mapping T can be used to partition $\mathcal{X}$ into N^2 subsets.

To compare and contrast a classifier to the authoritative truth mapping T, construct the measurements of sets for all $i, j = 1, \ldots, N$ such that

$$\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) = \int_{T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]} f(x)dx. \tag{8}$$

Consider the 2-class, 2-value logic crisp classification problem. In this case, the output label set $\mathcal{L}$ has two elements, for example {false,true}, {f,t} or {0,1}. Here we will use $\mathcal{L} = \{0, 1\}$ where 0 denotes false and 1 denotes true. As a result, there are $N^2 = 4$ disjoint subsets of feature set $\mathcal{X}$ and

$$\begin{aligned} \mathcal{X} &= \left(T^{-1}[\{0\}] \cap C^{-1}[\{0\}]\right) \cup \left(T^{-1}[\{1\}] \cap C^{-1}[\{0\}]\right) \\ &\quad \cup \left(T^{-1}[\{0\}] \cap C^{-1}[\{1\}]\right) \cup \left(T^{-1}[\{1\}] \cap C^{-1}[\{1\}]\right). \end{aligned}$$

Precision sets $T^{-1}[\{\ell_i\}] \cap C^{-1}[\{\ell_i\}]$ denote where the truth mapping and the classifier agree. When $i \neq j$, omission/commission sets denote which elements $x \in \mathcal{X}$ were incorrectly parsed into subset $C^{-1}[\{\ell_i\}]$ (commission set) instead of into the appropriate subset $C^{-1}[\{\ell_j\}]$ (omission set). For $\mathcal{L} = \{0, 1\}$, there are four measurements in the collection $\{\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])\}$ and—in the tradition of error matrices [14, 55]—we call these 4 measured values *error rates*.

Definition 24 *(Error rates). Let truth mapping T and classifier C be 2-class, measurable classifiers defined on measure space $(\mathcal{X}, \mathfrak{B}, \mu)$ where $\mathcal{X} = \mathbb{R}^d$ and measure μ is the integral measure defined in Equation 7 such that $\mu(\mathcal{X}) = 1$. Then, the four error rates of classifier C follow.*

Measurement of Class 0 *precision set*

$$\mu(T^{-1}[\{0\}] \cap C^{-1}[\{0\}]) = \int\limits_{T^{-1}[\{0\}] \cap C^{-1}[\{0\}]} f(x)dx$$

Measurement of Class 0 *omission (or Class* 1 *commission) set*

$$\mu(T^{-1}[\{1\}] \cap C^{-1}[\{0\}]) = \int\limits_{T^{-1}[\{1\}] \cap C^{-1}[\{0\}]} f(x)dx$$

Measurement of Class 1 *precision set*

$$\mu(T^{-1}[\{1\}] \cap C^{-1}[\{1\}]) = \int\limits_{T^{-1}[\{1\}] \cap C^{-1}[\{1\}]} f(x)dx$$

Measurement of Class 1 *omission (or Class* 0 *commission) set*

$$\mu(T^{-1}[\{0\}] \cap C^{-1}[\{1\}]) = \int\limits_{T^{-1}[\{0\}] \cap C^{-1}[\{1\}]} f(x)dx$$

These error rates quantify how often over a weighted feature space the classifier agrees or does not agree with truth. The weighting of feature space must be representative of the sampling of the feature space. Precision rates $\mu(T^{-1}[\{\ell_i\}] \cap C^{-1}[\{\ell_i\}])$ quantify how often over a weighted feature space the classifier agrees with truth. Measurements $\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])$ for all $i \neq j$ quantify the rate of omission/commission errors.

As defined, we can show the following theorems regarding the error rates of classifier C hold true.

Theorem 1 *The summation of a class's precision and commission rates is the measure* $\mu(C^{-1}[\{\ell_i\}])$

$$\begin{aligned} \sum_{j=1}^{N} \mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) &= \int\limits_{C^{-1}[\{\ell_i\}]} f(x)dx \\ &= \mu(C^{-1}[\{\ell_i\}]). \end{aligned}$$

Theorem 2 *The summation of the precision and omission rates for a class is the measure* $\mu(T^{-1}[\{\ell_i\}])$

$$\begin{aligned} \sum_{i=1}^{N} \mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) &= \int\limits_{T^{-1}[\{\ell_j\}]} f(x)dx \\ &= \mu(T^{-1}[\{\ell_j\}]). \end{aligned}$$

Theorem 3 *The summation of the rate measures is equal to one*

$$\begin{aligned} \sum_{i=1}^{N}\sum_{j=1}^{N}\mu(T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}]) &= \int_{\mathcal{X}} f(x)dx \\ &= \mu(\mathcal{X}) = 1. \end{aligned} \tag{9}$$

These assertions are easily determined since we know from the properties of a measure that

$$\sum_{i=1}^{N}\mu(\mathcal{X}_i) = \mu(\bigcup_{i=1}^{N}\mathcal{X}_i) \tag{10}$$

where $\{\mathcal{X}_i \subset \mathcal{X} : i = 1,\ldots,N\}$ are mutually disjoint subsets. It follows that

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\mu(T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}]) = \mu(\bigcup_{i=1}^{N}\bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}])$$

where

$$\begin{aligned} \bigcup_{i=1}^{N}\bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}] &= \left(\bigcup_{i=1}^{N}C^{-1}[\{\ell_i\}]\right)\bigcap\left(\bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\right) \\ &= \mathcal{X}\cap\mathcal{X} = \mathcal{X} \end{aligned}$$

and

$$\sum_{j=1}^{N}\mu(T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}]) = \mu(\bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}])$$

where

$$\begin{aligned} \bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}] &= \left(C^{-1}[\{\ell_i\}]\right)\bigcap\left(\bigcup_{j=1}^{N}T^{-1}[\{\ell_j\}]\right) \\ &= C^{-1}[\{\ell_i\}]\cap\mathcal{X} = C^{-1}[\{\ell_i\}] \end{aligned}$$

and also

$$\sum_{i=1}^{N}\mu(T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}]) = \mu(\bigcup_{i=1}^{N}T^{-1}[\{\ell_j\}]\cap C^{-1}[\{\ell_i\}])$$

where

$$\begin{aligned}\bigcup_{i=1}^{N} T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}] &= \left(T^{-1}[\{\ell_j\}]\right) \bigcap \left(\bigcup_{i=1}^{N} C^{-1}[\{\ell_i\}]\right) \\ &= T^{-1}[\{\ell_j\}] \cap \mathcal{X} = T^{-1}[\{\ell_j\}].\end{aligned}$$

4.2.6.2 Error rates per class. Error matrices traditionally present precision and omission rates per class [17]. To capture the error rate per class, a variation on the measurement $\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])$ is used. Define the error rates for class j as

$$\varepsilon^{(j,i)}(C^{-1}[\{\ell_i\}]) = \frac{\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])}{\mu(T^{-1}[\{\ell_j\}])} \tag{11}$$

for all $i \in \{1, \ldots, N\}$. The measure ε is defined on the inverse image of the truth mapping for class j so that the measure space for class j is the triple $(T^{-1}[\{\ell_j\}], \mathfrak{B}^{(j)}, \varepsilon^{(j,\cdot)})$ where $\mathfrak{B}^{(j)} = \{S \cap T^{-1}[\{\ell_j\}] : S \in \mathfrak{B}\}$.

Typically, the error matrix is set up as follows

$$\begin{bmatrix} \varepsilon^{(1,1)} & \ldots & \varepsilon^{(1,N)} \\ \vdots & \ddots & \vdots \\ \varepsilon^{(N,1)} & \ldots & \varepsilon^{(N,N)} \end{bmatrix} \tag{12}$$

where the columns of the matrix list the error rates for each class.

4.2.6.3 A partial ordering of classifiers. Now, having constructed the measure space $(\mathcal{X}, \mathfrak{B}, \mu)$, we define a relation $\succeq$ to gauge whether or not classifier A is more authoritative than classifier B. Recall, a set of ordered pairs is a relation. Relation $\succeq$ on a set S is a partial ordering of S if and only if (1) $\succeq$ is reflexive ($a \succeq a$ for every $a \in S$), (2) $\succeq$ is transitive (if $a \succeq b$ and $b \succeq c$, then $a \succeq c$), and (3) $\succeq$ is antisymmetric (if $a \succeq b$ and $b \succeq a$, then $a = b$).

Definition 25 *(Partially ordered set). A partially ordered set (or poset) is an ordered pair consisting of a set S and a relation $\succeq$ defined on the Cartesian set $S \times S$. A poset is denoted $(S, \succeq)$.* [47]

Definition 26 *(Dual of Poset). Given the poset $(S, \succeq)$, $(S, \preceq)$ is called the dual of $(S, \succeq)$ where the relation $\preceq$ is defined such that, for all $a, b \in \mathcal{C}$, $a \preceq b$ iff $b \succeq a$.* [47]

Let $\mathcal{C}$ be a collection of classifiers and the authoritative relation $\succeq$ be defined on the Cartesian map $\mathcal{C} \times \mathcal{C}$ to form the poset $(\mathcal{C}, \succeq)$. We define relation $\succeq$ such that, given classifiers $A, B \in \mathcal{C}$, classifier A is more authoritative than B (written as $A \succeq B$) if and only if

$$\mu(T^{-1}[\{\ell_j\}] \cap A^{-1}[\{\ell_i\}]) \leq \mu(T^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]) \qquad \forall i \neq j. \tag{13}$$

Note, it is necessary for poset $(\mathcal{C}, \succeq)$ to be defined on measure space $(\mathcal{X}, \mathfrak{B}, \mu)$ instead of the measure spaces $(T^{-1}[\{\ell_j\}], \mathfrak{B}^{(j)}, \varepsilon^{(j,\cdot)})$ in order to keep the relation $\preceq$ meaningful across feature set $\mathcal{X}$.

4.2.6.4 Estimation of error rates. In practice, the expression of measurement $\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])$ contains several unknowns. Among them are truth mapping T and density function f. Throughout the pattern recognition literature, a common strategy for overcoming this ignorance is to sample pairs from the truth mapping T. An estimate of f is implied by the sampling scheme. Let D be the collection of sampled pairs $D = \{(x_k, \ell_k) : k = 1, \ldots, K\}$. The implied estimate of f has the general form $f_D(x) = \sum_{k=1}^{K} \delta(x - x_k)$ where δ is the Dirac delta function. Then, error rates can be approximated as

$$\begin{aligned} \mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) &\approx \int\limits_{T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]} f_D(x) m(dx) \\ &\approx \int\limits_{T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]} \sum_{k=1}^{K} \delta(x - x_k) m(dx) \\ &\approx \frac{card(D \cap T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])}{K} \end{aligned} \tag{14}$$

such that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) \approx \frac{card(D)}{K} = \frac{K}{K} = 1. \tag{15}$$

The precision of the error rates is known to be highly susceptible to the sampling strategy [17] and, thus, highly susceptible to estimates of f.

There are other unknowns in the expression $\mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}])$. Specifically, it is unclear whether the complete feature set $\mathcal{X}$ and the complete label set $\mathcal{L}$ have been specified. Incomplete feature and label sets contribute significantly to confusion in classification. Overpopulated feature/label sets also contribute to confusion in classification, but this is due to shortcomings in training where, due to the higher ratio of parameters to training points, it is more difficult to generalize models of the appropriate form and complexity. [18, 73]

4.2.7 Confusion defined for measurable classifiers. Confusion measures compare and contrast the opinions of experts. Given this definition, error is a special instance of a confusion measure where one of the experts is the all-knowing authority on truth. Confusion merely involves a consultation among peers. In that sense, measuring confusion is less demanding. Interpreting confusion, however, is less straightforward.

Partitioning a feature set $\mathcal{X}$ using two peer classifiers A and B—a la $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]$—does not yield subsets that can be defined as measures of precision or commission omission rates but rates of different meaning and utility. This different meaning is captured in 4-value logic and utilized in the proposed confusion measure.

4.2.7.1 Class overlap in an N^3 partitioning of feature space. The severity of separation issues in a classification problem may be gauged by considering the nonempty partitions $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]$ where $j \neq i$. Given a classifier C defined on measure space $(\mathcal{X}, \mathfrak{B}, \mu)$ with output label set $\mathcal{L} = \ell_1, \ldots, \ell_N$, recall from Theorem 1 that the summation of a class's precision and commission rates is equal to the measurement $\mu(C^{-1}[\{\ell_i\}])$

$$\begin{aligned} \sum_{j=1}^{N} \mu(T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) &= \mu(\bigcup_{j=1}^{N} T^{-1}[\{\ell_j\}] \cap C^{-1}[\{\ell_i\}]) \\ &= \mu(C^{-1}[\{\ell_i\}]) \end{aligned} \tag{16}$$

From this theorem, we can prove the following:

Theorem 4 *With respect to measurable classifiers A, B, and the truth mapping T, there is a total of a N^3 summations within the measurement of sets $\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}])$ for all $i, j \in \{1, \ldots, N\}$.*

Proof. From Theorem 1, we can rewrite the value $\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}])$ as

$$\begin{aligned} &\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]) \\ &= \mu\left(\left\langle \bigcup_{k=1}^{N} T^{-1}[\{\ell_k\}] \cap A^{-1}[\{\ell_j\}] \right\rangle \bigcap \left\langle \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \cap B^{-1}[\{\ell_i\}] \right\rangle\right) \end{aligned}$$

Let $i, j \in \{1, \ldots, N\}$ then feature set $\mathcal{X}$ is comprised of N^2 disjoint subsets of the form $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]$. Each of these subsets can be partitioned further into N disjoint subsets as seen by manipulating the subset $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]$. First, consider

$$\begin{aligned} A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}] &= \left\langle \bigcup_{k=1}^{N} T^{-1}[\{\ell_k\}] \cap A^{-1}[\{\ell_j\}] \right\rangle \bigcap \left\langle \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \cap B^{-1}[\{\ell_i\}] \right\rangle \\ &= \left\langle \bigcup_{k=1}^{N} T^{-1}[\{\ell_k\}] \right\rangle \cap A^{-1}[\{\ell_j\}] \bigcap \left\langle \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \right\rangle \cap B^{-1}[\{\ell_i\}] \\ &= \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \cap \left\langle \bigcup_{k=1}^{N} T^{-1}[\{\ell_k\}] \right\rangle \cap B^{-1}[\{\ell_i\}] \cap A^{-1}[\{\ell_j\}]. \end{aligned}$$

Next, since $T^{-1}[\{\ell_n\}] \cap T^{-1}[\{\ell_k\}] \cap B^{-1}[\{\ell_i\}] \cap A^{-1}[\{\ell_j\}] = \emptyset$ for all $k \neq n$, it follows that

$$\begin{aligned} A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}] &= \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \cap B^{-1}[\{\ell_i\}] \cap T^{-1}[\{\ell_n\}] \cap A^{-1}[\{\ell_j\}] \\ &= \bigcup_{n=1}^{N} T^{-1}[\{\ell_n\}] \cap B^{-1}[\{\ell_i\}] \cap A^{-1}[\{\ell_j\}]. \end{aligned}$$

Therefore, the measurement $\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}])$ can be expressed as the summation of N measurements

$$\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]) = \sum_{n=1}^{N} \mu(T^{-1}[\{\ell_n\}] \cap B^{-1}[\{\ell_i\}] \cap A^{-1}[\{\ell_j\}]).$$

and the calculation of all N^2 measurements $\mu(A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}])$ thus involves N^3 summations. ■

The amount of confusion among classifiers A and B may be gauged—in part—by assessing the non-empty partitions $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_i\}]$ where $j \neq i$. A full set of subsets implies interpolation issues as it suggests considerable overlap among classes. Unfortunately, this limited analysis leaves out the confusion that may lie in the partitions $A^{-1}[\{\ell_j\}] \cap B^{-1}[\{\ell_j\}]$. Confusion in these partitions is generally caused not by overlapping class data

but instead by a lack of data. In order to capture both forms of confusion (uncertainty due to interpolation or extrapolation issues), we turn to four-value logic.

4.2.7.2 Four-value logic. Four-value logic addresses ambiguities in 2-value and 3-value logics in representing experience. Two-value logic implements the rudimentary binary decision set: "false" and "true". Three-value logic allows for the possibility of a non-decisive computation and implements "false", "uncertain", and "true" decision sets. Along this vein, 4-value logic resolves an ambiguity in 3-value logic. The label set of 3-value logic—{false, uncertain, true}=$\{f, u, t\}$—provides no distinction between vague experience and the absence of experience. Four-value logic splits the "uncertain" label in two resulting in the label set {false, true, uncertain interpolation, uncertain extrapolation}=$\{f, t, i, e\}$. We have summarized the distinctions of 4-value logic from 2-value and 3-value logic treatments in Figure 18. [62]

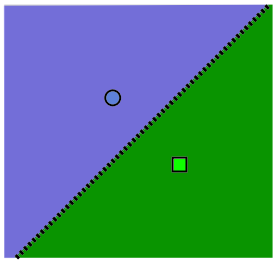
Four-value logic interprets classifiers as class experts—classifiers which isolate class data rather than merely separate.

Definition 27 *(Class expert). A class expert is a classifier designed to isolate features near the training data of a specific class from other data.*

For instance, in a 4-value interpretation of a two-value classifier, we might replace the label set $\mathcal{L}_{2v} = \{f, t\}$ with the label set $\mathcal{L}_{2v} = \{\{f, e\}, \{t, i\}\}$. The label set $\mathcal{L}_{2v} = \{\{f, e\}, \{t, i\}\}$ implies that the 2-value classifier was trained based on a strategy to isolate elements that are considered near or similar to Class "true" data. An isolation strategy is motivated to capture the sensitivity[4] of a classification, more so than capturing its separability from other classes.

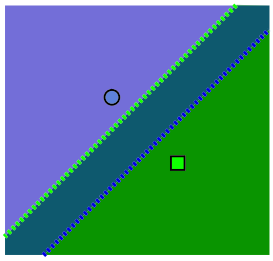
Define a classifier B on the Cartesian set $\mathcal{X} \times \mathcal{L}_{2v}$. Let label set $\mathcal{L}_{2v} = \{\{f, e\}, \{t, i\}\}$ so that classifier B dichotomizes the feature set X into two subsets $B^{-1}[\{f, e\}]$, $B^{-1}[\{t, i\}]$. Subset $B^{-1}[\{t, i\}]$ contains "true"-labeled data and interpolations between "true" data. Alternately, $B^{-1}[\{f, e\}]$ contains "false"-labeled data and extrapolations from "true" elements. Now define a different 2-value classifier A with a label set $\mathcal{L}'_{2v} = \{\{t, e\}, \{f, i\}\}$

[4]Sensitivity of classifier B is the probability $p(x \in \mathcal{X}|B(x) = \ell)$ that a feature $x \in \mathcal{X}$ is assigned a particular label $\ell \subset \mathcal{L}$.



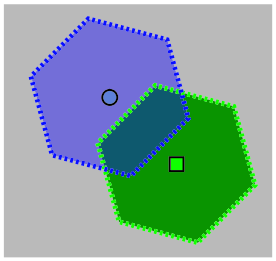
$\mathcal{A} = \neg\mathcal{B}$ == "false"
$\mathcal{B} = \neg\mathcal{A}$ == "true"

(a) Two-value logic



$\mathcal{A} \cap \neg\mathcal{B}$ == "false"
$\mathcal{B} \cap \neg\mathcal{A}$ == "true"
$(\mathcal{A} \cap \mathcal{B}) \cup (\neg\mathcal{A} \cap \neg\mathcal{B})$ == "uncertain"

(b) Three-value logic



$\mathcal{A} \cap \neg\mathcal{B}$ == "false"
$\mathcal{B} \cap \neg\mathcal{A}$ == "true"
$\mathcal{A} \cap \mathcal{B}$ == "interpolated"
$\neg\mathcal{A} \cap \neg\mathcal{B}$ == "extrapolated"

(c) Four-value logic

Figure 18. Class arrangements in 2-value, 3-value and 4-value logic where "◯"∈ Class False and "□"∈ Class True.

to isolate those elements considered near "false" data, so that subset $A^{-1}[\{f,i\}]$ contains "false"-labeled data and interpolations between those false elements and subset $A^{-1}[\{t,e\}]$ contains "true"-labeled data and extrapolations from "false" data.

In the next section, we derive a 4-value classifier from a pair of classifiers—the first classifier with output label set $\mathcal{L}_{2v} = \{\{f,e\},\{t,i\}\}$, the second with output label set $\mathcal{L}'_{2v} = \{\{t,e\},\{f,i\}\}$—using a construction called the confusion set.

4.2.7.3 Four-value logic confusion set for the two-class problem. Let A and B be classifiers defined on feature set $\mathcal{X}$ with output label set $\mathcal{L} = \{f,t,i,e\}$. For each pair of subsets of labels $L, M \subset \mathcal{L}$, define the set-valued mapping $\mathbb{C}^{(L,M)}$ defined on classifiers A and B to be

$$\mathbb{C}^{(L,M)}(A,B) = A^{-1}[L] \cap B^{-1}[M]. \tag{17}$$

Assume classifier A is an expert on class "false" and classifier B is an expert on class "true". Classifier A, then, is an expert on the features in the set $A^{-1}[\{f,i\}] \subset \mathcal{X}$. Similarly, B is an expert on the set $B^{-1}[\{t,i\}]$. Define the *expertise sets* of A and B to be, respectively, $\mathcal{A} = A^{-1}[\{f,i\}]$ and $\mathcal{B} = B^{-1}[\{t,i\}]$. For special choices of the label subsets $L = \{f,i\}$, $L = \{t,e\} = 1 - \{f,i\}$ and $M = \{t,i\}$, $M = \{f,e\} = 1 - \{t,i\}$, one gets the *confusion set*, a 4-tuple partition of feature set X, in terms of classifiers A and B

$$\begin{aligned}
\mathbb{C}^{(\{f,i\},\{t,i\})}(A,B) &= A^{-1}[\{f,i\}] \cap B^{-1}[\{t,i\}] = \mathcal{A} \cap \mathcal{B} \\
\mathbb{C}^{(\{t,e\},\{f,e\})}(A,B) &= A^{-1}[\{t,e\}] \cap B^{-1}[\{f,e\}] = \neg\mathcal{A} \cap \neg\mathcal{B} \\
\mathbb{C}^{(\{f,i\},\{f,e\})}(A,B) &= A^{-1}[\{f,i\}] \cap B^{-1}[\{f,e\}] = \mathcal{A} \cap \neg\mathcal{B} \\
\mathbb{C}^{(\{t,e\},\{t,i\})}(A,B) &= A^{-1}[\{t,e\}] \cap B^{-1}[\{t,i\}] = \neg\mathcal{A} \cap \mathcal{B}.
\end{aligned}$$

Observe since, by definition, the classifiers' domains are all $\mathcal{X}$ then, indeed, the set complements are correct since

$$\begin{aligned}
\mathcal{X} &= A^{-1}[\{f,t,i,e\}] \\
&= A^{-1}[\{f,i\} \cup \{t,e\}] \\
&= A^{-1}[\{f,i\}] \cup A^{-1}[\{t,e\}].
\end{aligned}$$

Therefore,

$$\begin{aligned} A^{-1}[\{t,e\}] &= \mathcal{X} - A^{-1}[\{f,i\}] \\ &= \mathcal{X} - \mathcal{A} \\ &= \neg\mathcal{A}. \end{aligned}$$

For clarification, consider the table:

Table 2. The confusion set as a confusion matrix of sets.

	M = False ∪ Extrapolated	M = True ∪ Interpolated
L = False ∪ Interpolated	$\mathcal{A} \cap \neg\mathcal{B}$	$\mathcal{A} \cap \mathcal{B}$
L = True ∪ Extrapolated	$\neg\mathcal{A} \cap \neg\mathcal{B}$	$\neg\mathcal{A} \cap \mathcal{B}$

Table 2 denotes the *confusion set* in matrix form—that is, the table presents the confusion matrix of sets. Thus given classifier A, an expert on the set $\mathcal{A}$ (that corresponds to the first label, i.e., f in this case), and the classifier B, an expert on the set $\mathcal{B}$ (that corresponds to the second label, i.e., t in this case), define the *confusion mapping* that produces the confusion matrix of sets to be

$$\mathbf{C}^{(L,M)}(A,B) = \begin{bmatrix} \mathcal{A} \cap \neg\mathcal{B} & \mathcal{A} \cap \mathcal{B} \\ \neg\mathcal{A} \cap \neg\mathcal{B} & \neg\mathcal{A} \cap \mathcal{B} \end{bmatrix}.$$

To construct the measure of the confusion sets, we define the matrix-valued measure $\overrightarrow{\mu}$ to be

$$\overrightarrow{\mu}(\mathbf{C}^{(L,M)}(A,B)) = \begin{bmatrix} \mu(\mathcal{A} \cap \neg\mathcal{B}) & \mu(\mathcal{A} \cap \mathcal{B}) \\ \mu(\neg\mathcal{A} \cap \neg\mathcal{B}) & \mu(\neg\mathcal{A} \cap \mathcal{B}) \end{bmatrix}.$$

4.2.7.4 Constructing a 4-value classifier. A 4-value classifier C defined on feature set $\mathcal{X}$ with output label set $\mathcal{L} = \{f, t, i, e\}$ may be derived from the confusion set.

$$\begin{aligned} C^{-1}[\{f\}] &= \mathcal{A} \cap \neg\mathcal{B} \qquad & C^{-1}[\{i\}] &= \mathcal{A} \cap \mathcal{B} \\ C^{-1}[\{e\}] &= \neg\mathcal{A} \cap \neg\mathcal{B} \qquad & C^{-1}[\{t\}] &= \neg\mathcal{A} \cap \mathcal{B} \end{aligned}$$

True set $C^{-1}[\{t\}]$ includes those features hypothesized far from "false" and near "true"; therefore, we believe these features to be "true" and easily separated from "false" data. The false set $C^{-1}[\{f\}]$ includes those features we believe to be "false" and easily separated from "true" data. Together, the true set $C^{-1}[\{t\}]$ and false set $C^{-1}[\{f\}]$ comprise the *confirmation set*

$$\begin{aligned} \text{Confirm}(A,B) &= C^{-1}[\{f,t\}] \\ &= C^{-1}[\{f\}] \cup C^{-1}[\{t\}] \\ &= \mathcal{A} \ominus \mathcal{B}. \end{aligned} \tag{18}$$

where $\ominus$ denotes the XOR operation. The confirmation set affirms the expertise of A and B in those regions of feature set $\mathcal{X}$ where the class experts are not in conflict. The interpolation set $C^{-1}[\{i\}]$ specifies features hypothesized to be similar to data of both class—regions of $\mathcal{X}$ where experts A and B are in direct conflict. The extrapolation set $C^{-1}[\{e\}]$ encompasses features considered far from all classes. Recall, a primary motivation in 4-value logic treatments is to identify the operational set—the set of largely unrealized, but anticipated data in the operational use of a classifier. The *operational set* of 4-value classifier C is the subset $C^{-1}[\{f,t,i\}] \subset X$. This subset excludes the extrapolation set $C^{-1}[\{e\}]$ so that the operational set encompass only those features hypothesized to be similar to training data or previous experiences.

$$\begin{aligned} \text{Operational}(A,B) &= C^{-1}[\{f,t,i\}] \\ &= C^{-1}[\{f\}] \cup C^{-1}[\{t\}] \cup C^{-1}[\{i\}] \\ &= \mathcal{X} - C^{-1}[\{e\}] \end{aligned} \tag{19}$$

Finally let us note, for a collection of 4-value classifiers $\mathcal{C}$ on measure space $(\mathcal{X}, \mathfrak{B}, \overrightarrow{\mu})$, we can define a relation $\succeq_{4v}$ to gauge whether or not classifier C_1 is more general than classifier C_2 given $C_1, C_2 \in \mathcal{C}$. This relation contrasts the cardinality of the classifiers' confirmation and operational sets since their memberships decrease as data generalizations range from oversimplification to memorization. We anticipate as operational experience is accumulated and set memberships are more fully realized, the appropriate size and complexity of the operational set will become clear. [18, 73]

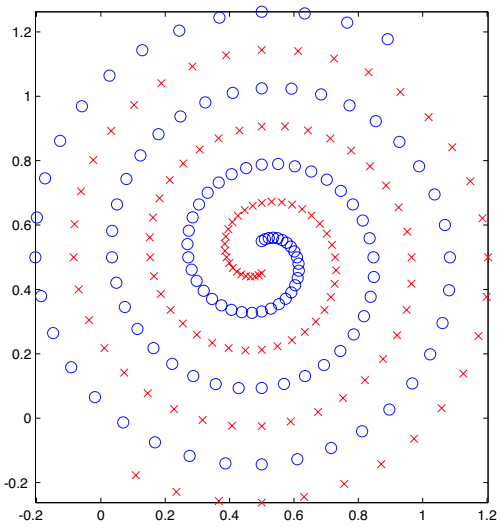


Figure 19. Data set for interlocking spirals: Class True == 'o', Class False == 'x'.

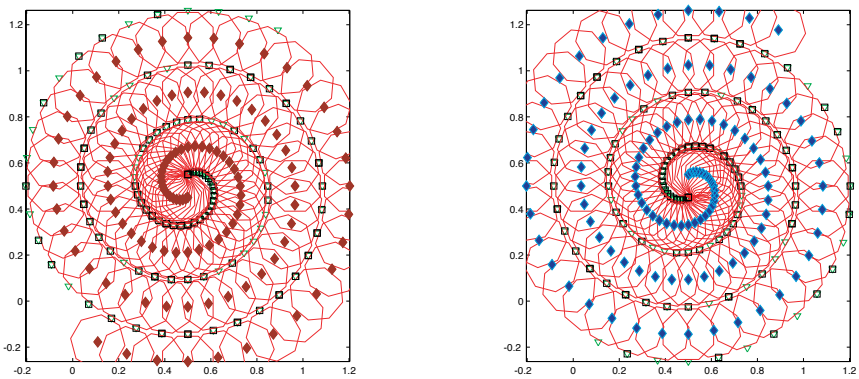
4.3 Illustration of the 4-tuple confusion set

To illustrate the construction of a 4-value classifier, we separate two interlocking spirals using a special treatment of multilayer perceptrons. The interlocking spirals are partitioned using two class experts: Class expert A isolates *false* points (the x's in Figure 19) from all other points, and class expert B isolates *true* points (the o's in Figure 19). Both class experts use a 2-hidden-layer MLP architecture. Each classifier has memorized the training set by employing a nearest neighbor rule base, but the classifiers differ in that each is more generous in corralling neighborhoods for the particular class it isolates.

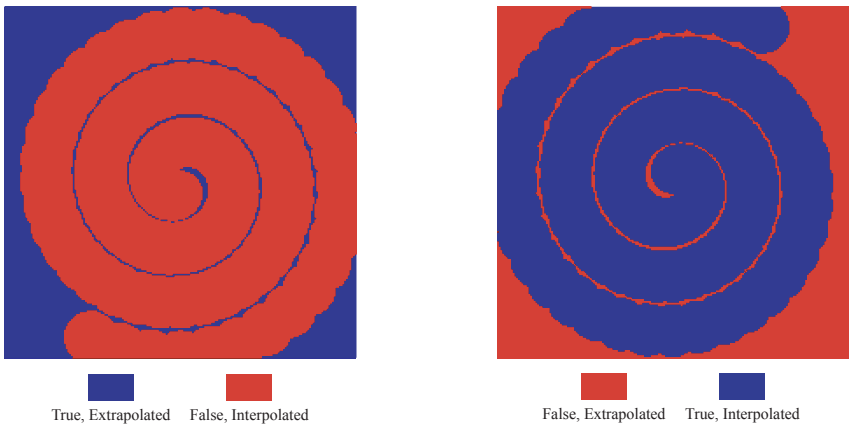
Let A, B and C be classifiers defined on the $\mathcal{X} = \mathbb{R}^2$ with output label set $\mathcal{L} = \{\mathrm{x}, \mathrm{o}, i, e\}$. Classifier A is illustrated in Figure 20 (a) in terms of its set of hyperplanes and (c) with respect to its partitioning of $\mathbb{R}^2$. Classifier B is illustrated in Figure 20 (b) and (d). Figure 21 depicts 4-value classifier C, the 4-value logic interpretation of classifiers A and B as given in the confusion set $\mathbb{C}^{L,M}(A, B)$ for the two-class problem.

$$C^{-1}[\{\mathrm{x}\}] = \mathbb{C}^{L_+,M_-}(A,B) = \mathcal{A} \cap \neg\mathcal{B} \qquad C^{-1}[\{i\}] = \mathbb{C}^{L_+,M_+}(A,B) = \mathcal{A} \cap \mathcal{B}$$
$$C^{-1}[\{e\}] = \mathbb{C}^{L_-,M_-}(A,B) = \neg\mathcal{A} \cap \neg\mathcal{B} \qquad C^{-1}[\{\mathrm{o}\}] = \mathbb{C}^{L_-,M_+}(A,B) = \neg\mathcal{A} \cap \mathcal{B}$$

where $L \in \{L_-, L_+\} = \{\{\mathrm{o}, e\}, \{\mathrm{x}, i\}\}$ and $M \in \{M_-, M_+\} = \{\{\mathrm{x}, e\}, \{\mathrm{o}, i\}\}$.



(a) Arrangement A (b) Arrangement B



(c) Output of MLP implementing A (d) Output of MLP implementing B

Figure 20. Two multilayer perceptrons—respectively class experts A and B—implement overlapping generalizations of the two-class spiral data shown in Figure 19.

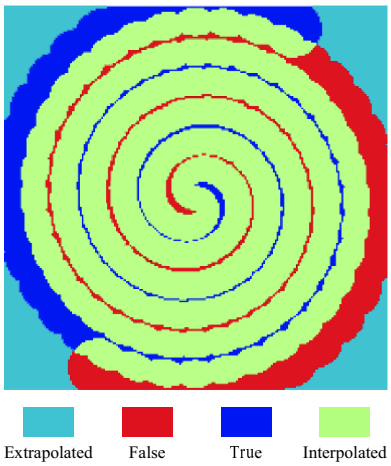
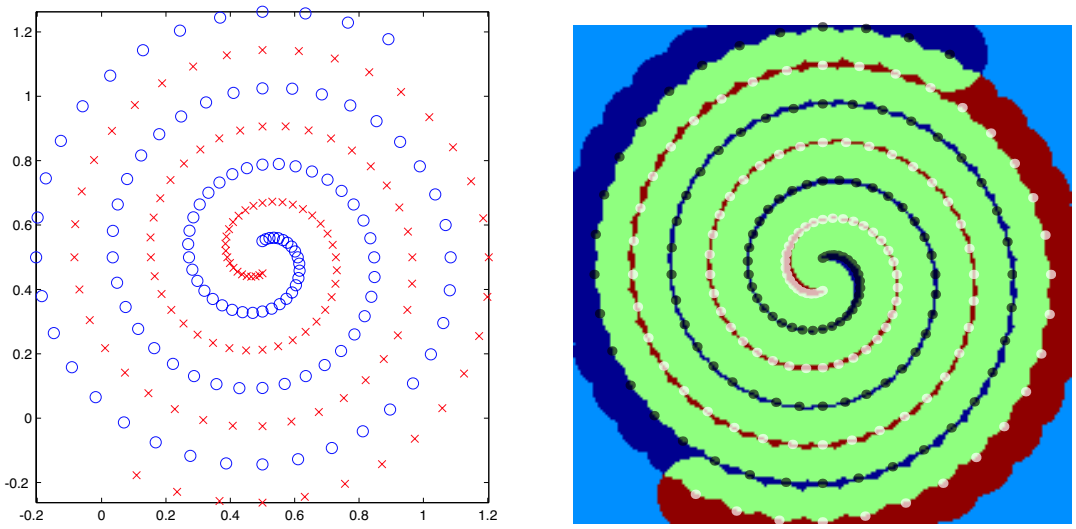


Figure 21. Four-value logic generalization of spiral data.



(a) Original data set (b) Four-value logic generalization

Figure 22. Four-value logic generalization with overlay of the original data.

The overlay of original data in Figure 22(b) illustrates that, even though the original data set was memorized, training has not led to poor generalization in the final solution. Since the training data was sampled at an appropriate resolution, new information within the range of the original training set is classified correctly while points outside the range of training data are properly labeled as extrapolations in the 4-value classifier. In the mean-squared-error sense, all three classifiers A, B and C perform equally for training data and any new points generated by the original functions $x_{\mathrm{O}}(z) = [(1 - \frac{0.95}{144}z).*cos(\frac{\pi}{2} + \frac{\pi}{18}z); (1 - \frac{0.95}{144}z).*sin(\frac{\pi}{2} + \frac{\pi}{18}z)]$ and $x_{\mathrm{X}}(z) = -x_{\mathrm{O}}(z)$ within the range $z \in [33, 144]$. This holds because there is ample experience within this range to resolve $x_{\mathrm{O}}(z)$ and $x_{\mathrm{X}}(z)$. The 4-value classifier C is more desirable because it expresses the conditions where the original function was not sampled adequately and no certain solution can be tendered.

4.4 The 4-value classifier

In summary, let us review the construction of a four-value classifier as first discussed in Section 4.2.7.4. A crisp-set classifier partitions a feature set and tags each partition with a particular label. Let classifier C be a function defined on the Cartesian product $\mathcal{X} \times \mathcal{L}$. Feature set $\mathcal{X}$ and label set $\mathcal{L}$ form an input/output pair $(x_k, \ell_k) \in C$ where feature

$x_k \in \mathcal{X}$ and label $\ell_k \in \mathcal{L}$. Assuming crisp logic, classifier C partitions $\mathcal{X}$ into N disjointed, nonempty subsets $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$. Let each subset $\mathcal{X}_i$ correspond to one label ℓ_i and each label correspond to at least one feature $x_k \in \mathcal{X}$. The collection $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$ is then exhaustive, i.e., $\bigcup_{i=1}^{N} \mathcal{X}_i = \mathcal{X}$, and mutually disjoint, i.e., $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for all $i \neq j$, and the inverse image of any label $\ell \in \mathcal{L}$ denotes a subset of $\mathcal{X}$, i.e., $C^{-1}[\{\ell\}] \subset \mathcal{X}$.

A 4-value classifier is constructed by creating expert classifiers for each class in a training set. Let 2-class dataset $\mathfrak{D}$ be a subset of the Cartesian product $\mathcal{X} \times \mathcal{L}$ where label set $\mathcal{L} = \{false,\ true\} = \{f, t\}$. The class expert in *false* is prepared by isolating domain regions near Class *false* points. Let the class expert A partition $\mathcal{X}$ into the signed set $(\mathcal{A}, \mathcal{A}^c)$ where subsets

$$\begin{aligned} \mathcal{A} &= A^{-1}[\{f, i\}] = A^{-1}[\{f\}] \cup A^{-1}[\{i\}], \\ \mathcal{A}^c &= A^{-1}[\{t, e\}] = \mathcal{X} - \mathcal{A}. \end{aligned}$$

Partition $\mathcal{A}$ is the inverse image of the false and interpolation labels—the subset of $\mathcal{X}$ that contains both truthed *false* points and those untruthed points considered near *false* points. Complement $\mathcal{A}^c$ contains all other points. Similarly, the expert in *true* isolates regions near Class *true* points. Let class expert B partition $\mathcal{X}$ into the signed set $(\mathcal{B}, \mathcal{B}^c)$ such that $\mathcal{B} = B^{-1}[\{t, i\}]$ and $\mathcal{B}^c = B^{-1}[\{f, e\}] = \mathcal{X} - \mathcal{B}$. The 4-value logic interpretation of class experts A and B is expressed as four disjoint partitions of $\mathcal{X}$. Let classifier C be a function defined on Cartesian product $\mathcal{X} \times \mathcal{L}_{4v}$ where $\mathcal{L}_{4v}$ = {False, True, Interpolation, Extrapolation} $= \{f, t, i, e\}$ such that

$$\begin{aligned} C^{-1}[\{f\}] &= \mathcal{A} \cap \mathcal{B}^c \qquad C^{-1}[\{i\}] = \mathcal{A} \cap \mathcal{B} \\ C^{-1}[\{t\}] &= \mathcal{A}^c \cap \mathcal{B} \qquad C^{-1}[\{e\}] = \mathcal{A}^c \cap \mathcal{B}^c. \end{aligned}$$

Classifier C defines the inverse image of label False as $C^{-1}[\{f\}] = \mathcal{A} \cap \mathcal{B}^c$ where expert A isolates regions near *false* points and is not contradicted by expert B. The inverse image of label True is $C^{-1}[\{t\}] = \mathcal{A}^c \cap \mathcal{B}$ where expert B isolates regions near *true* points and is not contradicted by expert A. The inverse image of Interpolation $C^{-1}[\{i\}] = \mathcal{A} \cap \mathcal{B}$ specifies where the class experts contradict each other—both asserting "nearness"—while

the inverse image of Extrapolation $C^{-1}[\{e\}] = \mathcal{A}^c \cap \mathcal{B}^c$ denotes where neither expert asserts prior experience. [47, 62]

4.5 Summary

Formal confusion measures edge us closer to the elusive capability of self-evaluation. The power of the Theory of Confusion pivots on the assertion that a data generalization embodies a hypothesis of what is considered near training data and what is considered far.

By contrasting the hypotheses of multiple classifiers, one can partition a feature space into regions of confidence (i.e., the confirmation set), regions of uncertainty due to interpolation confusion, and regions of high uncertainty due to extrapolation confusion. This partition constitutes a more analytical, expressive declaration of confidence than can be offered in the sample mean error rate tabulated by an error matrix and is valuable in selecting a single classifier or committee of classifiers to process particular regions of feature space with realistic expectations of performance.

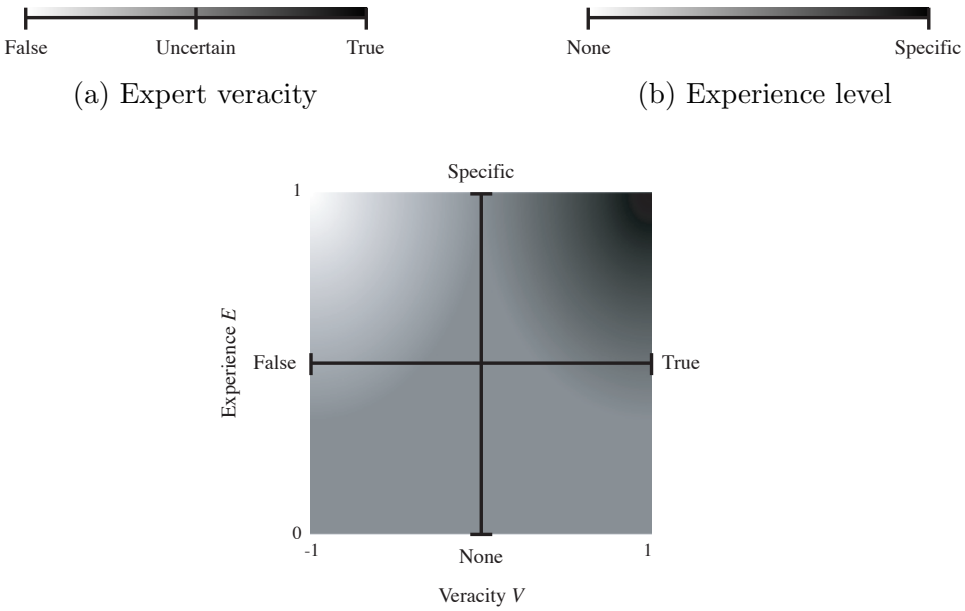
V. Expertise logic

The goal of this chapter is the quantification of expertise in classification. Quantifying expertise allows one to separate expert classifiers from "pretenders". The proficiency of a classifier hinges on its ability to interpolate operational data accurately. Success is measured by quantifying (1) the fitness of the classifier in ordering the hypothesized operational set and (2) the completeness of the training set in representing the operational set. When the operational set is largely untruthed, error measures—model versus truth comparisons—have limited utility in gauging a classifier's competency. To compensate, we propose the use of confusion measures—model versus model comparisons—to evaluate expert classifiers over untruthed, but interesting, portions of the operational set.

We begin this chapter with an introduction to expertise logic, its elements and its utility in recognizing arrogance in classification; we wrap up with proposed fusion rules for expert classifiers. In the next section, we define the expert classifier and propose a figure of merit—the ordered veracity-experience response curve—to judge appropriate expressions of expertise over the domain of such a classifier. We base our concept of computational expertise on the premise that learning agents do not need to be authoritative experts in a task to be able to recognize confusion among themselves. As a motivation to this conceptualization, consider that students in a classroom can advise each other of subjects in a curriculum where retraining with additional resources and special care is required. In other words, the students do not need to be experts in a subject to make reasonable judgements on whether their teachers' instruction needs to be augmented. We would like computer agents to be able to exhibit similar reasoning skill in self and peer-to-peer evaluation. Toward this end, we construct a confusion measure by contrasting a classifier's veracity to its apparent skill. We have deemed this construction *expertise logic*.

5.1 Expertise logic and arrogance in classification

Expertise logic orders the opinions of experts. An *expert* is a special classifier—a relational computation with not only a mechanism for decision making but also a quantifiable skill level. We propose expertise logic as a means of conveying relative experience



(a) Expert veracity

(b) Experience level

(c) Expert veracity versus experience

Figure 23. The uncertainty of an expert is quantified in two intervals: (a) Expert veracity, (b) Experience level. Uncertainty is conceptualized as a function of grayscale on (c) the experience level versus veracity set.

among experts and avoiding expressions of arrogance. Expertise logic is derived from the two quantities called veracity and experience as shown in Figures 23(a) and (b). In this section, we shall define these quantities and how they relate to arrogance in classification.

5.1.1 Veracity versus Experience. Let us define veracity and experience. Let A be a classifier defined on feature set $\mathcal{X}$ with label set $\mathcal{L}$. Assume that classifier A was trained on subset $\mathcal{A} \subset \mathcal{X} \times \mathcal{L}$. We consider the pair $(A, \mathcal{A})$ together. Given a feature vector $x \in \mathcal{X}$, one wishes to derive an expert—a relation that maps the information $(x, A, \mathcal{A})$ to a decision $(\ell, \mathfrak{e}, \mathfrak{v})$. The 3-tuple decision of the expert is comprised of the label $\ell = A(x) \in \mathcal{L}$ and two additional outputs $\mathfrak{e}$ and $\mathfrak{v}$. The output $\mathfrak{e}$ quantifies the experience associated with the input x and training set $\mathcal{A}$, and the output $\mathfrak{v}$ quantifies the veracity of the classifier towards its choice of label $\ell = A(x)$.

5.1.1.1 Veracity of a Classifier. Veracity[1] is the classifier's internal conviction towards truthfulness of a label $\ell \in \mathcal{L}$. There are established methods for quantifying veracity. Veracity is a special quantity of an assertion's probability and may be derived from confidence, certainty or uncertainty scores such as fuzzy logic, Bayesian error bars, error matrices, and entropy. Here, it is not our intention to derive new or improved veracity quantifiers but to make a subtle distinction in definition.

Definition 28 *(Veracity) The veracity of a classifier A at feature x is a quantity of the strength of conviction in the classification $\ell = A(x)$. Let $\mathfrak{v} = V_A(x) \in [0,1]$ denote the veracity of classifier A at x. If $V_A(x) = 0$ then the classifier has no conviction toward the chosen label ℓ. If $V_A(x) = 1$ then the classifier is fully convicted towards its choice of label.*

In this definition, veracity must not be misinterpreted as a measure of correctness. A classifier may assign an incorrect label but still have a sizable strength of conviction towards its labeling. Given the 2-class problem with label set $\mathcal{L} = \{false, true\}$, veracity is mapped into two different intervals $V_{false}(x) \in [0,1]$ and $V_{true}(x) \in [0,1]$. For illustration purposes, we have depicted the two veracities in a single interval in Figure 23(a) by mapping $-V_{false}(x) \in [-1,0]$ and $V_{true}(x) \in [0,1]$ to form the interval $V(x) \in [-1,1]$, ranging then from $false$ to *uncertain* to $true$. Note if the conviction for a *true* classification is small, it does not necessarily follow that the classification is *false*. This is an important distinction, because otherwise we assume the classifier is always interpolating. To assume a classifier is always interpolating is to assume that if a feature is not near training data of one class then it is near training data of another class. This assumption leads to false positives.

5.1.1.2 Experience of a Classifier. Experience is a special quantity of a query's bearing, expressing the fitness of an expert to answer a query based on what the expert knows *a priori*. The experience of a classifier depends on the data set used to train the classifier.

Definition 29 *(Experience) The experience of a classifier A at feature x is a quantity of closeness of feature x to training data $\mathcal{A}$. Let experience $\mathfrak{e} = E_A(x) \in [0,1]$ denote the*

[1]For further motivation on veracity, see the paper by Alsing et al [3].

experience of classifier A at x. If $E_A(x) = 0$ then the classifier has no experience with data near x. If $E_A(x) = 1$ then the classifier has all-knowing experience with data near x.

Of the two quantities, experience is more often ignored in classification problems so let us spend time in its discussion. To understand the concept, let us start with some background material concerning metric spaces. Let ρ be a metric defined on the feature set $\mathcal{X}$ so that $(\mathcal{X}, \rho)$ is a metric space. Recall the definition of a metric.

Definition 30 *(Metric [5]). Let $\mathcal{X}$ be a nonempty set. A metric is a real-valued function ρ defined on all of $\mathcal{X} \times \mathcal{X}$ such that the following properties hold true:*

1. *$\rho(x, y) \geq 0$ for all $x, y \in \mathcal{X}$. (nonnegativity)*
2. *$\rho(x, y) = \rho(y, x)$ for all $x, y \in \mathcal{X}$. (symmetry)*
3. *$\rho(x, y) \leq \rho(x, z) + \rho(z, y)$ for all $x, y, z \in \mathcal{X}$. (triangle inequality)*
4. *$\rho(x, y) > 0$ if and only if $x \neq y$. (positive definiteness)*

Definition 31 *(Metric Space [5]). Let $\mathcal{X}$ be a nonempty set and let ρ be a metric defined on $\mathcal{X}$, then the pair $(\mathcal{X}, \rho)$ is said to be a metric space.*

Definition 32 *(Ball). Let $(\mathcal{X}, \rho)$ be a metric space. An open ball, or open hypersphere, at x with radius r is defined to be the set $\mathsf{B}(x, r) = \{y \in \mathcal{X} : \rho(x, y) < r\}$. A closed ball is the set $\overline{\mathsf{B}(x, r)} = \{y \in \mathcal{X} : \rho(x, y) \leq r\}$.*

Definition 33 *(Distance to a Set). Given a subset $\mathcal{S} \subset \mathcal{X}$ and a feature $x \in \mathcal{X}$, we define the distance from x to $\mathcal{S}$ to be* $\operatorname{dist}(x, \mathcal{S}) = \inf\{\rho(x, y) : y \in \mathcal{S}\}$.

There are many ways to construct the experience quantifiers; but, before we present an example, we should discuss a few axioms that we can assume about experience. Let A be a classifier defined on $\mathcal{X}$ trained on data $\mathcal{A}$, and let the set of features from the data be denoted as the set of first components,

$$\mathcal{A}_1 = \{x \in \mathcal{X} : (x, \ell) \in \mathcal{A}\}.$$

Our first three axioms establish experience $E_A(x)$ as a normalized quantity inversely proportional to $\operatorname{dist}(x, \mathcal{A}_1)$.

Axiom 1 *Given a feature $x \in \mathcal{X}$, if x is a training datum, then experience $E_A(x) = 1$.*

Axiom 2 *Given a feature $x \in \mathcal{X}$, if there are no training data near x, then the experience $E_A(x)$ is small (close to 0).*

Axiom 3 *Given a feature $x \in \mathcal{X}$, if there are training data near x, then the experience $E_A(x)$ is large (close to 1) .*

In considering the distribution of experience values over a domain, with respect to density we formulate the following axioms.

Axiom 4 *Given a feature $x \in \mathcal{X}$, if there are few training data near x, then the experience $E_A(x)$ is likely to be small but is not necessarily small.*

Axiom 5 *Given a feature $x \in \mathcal{X}$, if there are many training data near x, then the experience $E_A(x)$ is likely to be large but is not necessarily large.*

These last two axioms say that $E_A(x)$ is proportional to $\text{card}(\mathcal{A})$ though other forces—such as the closeness of a training datum or the local complexity of the data distribution—can supercede. To construct the example of an experience mapping, let us first consider a definition based directly on the density at x with respect to $\mathcal{A}_1$,

$$\text{den}(x, \mathcal{A}_1) = \lim_{r \to 0^+} \frac{\text{card}(\mathcal{A}_1 \cap \mathsf{B}(x, r))}{\text{vol}(\mathsf{B}(x, r))}.$$

Of course, for finite sets $\mathcal{A}_1$ with $x \notin \mathcal{A}_1$ then $\text{den}(x, \mathcal{A}_1) = 0$. If $x \in \mathcal{A}_1$ then $\text{den}(x, \mathcal{A}_1) = \infty$ since $\text{card}(\mathcal{A}_1 \cap \mathsf{B}(x, r)) \geq 1$ for all $r > 0$. Therefore, we modify the definition to work for finite sets. Choose the radius to be $r_x = 2\ \text{dist}(x, \mathcal{A}_1)$. Now determine the volume of the ball with radius r_x. We know that the set $\mathcal{A}_1 \cap \mathsf{B}(x, r_x)$ is nonempty, hence the cardinality is nonzero. Consider

$$\frac{\text{card}(\mathcal{A}_1 \cap \mathsf{B}(x, r_x))}{\text{vol}(\mathsf{B}(x, r_x))}$$

which approximates the density at x. But, this quantity is not bounded between 0 and 1, nor does it satisfy Axiom 1. We apply the function $z/(z+1)$ to force the values to lie in

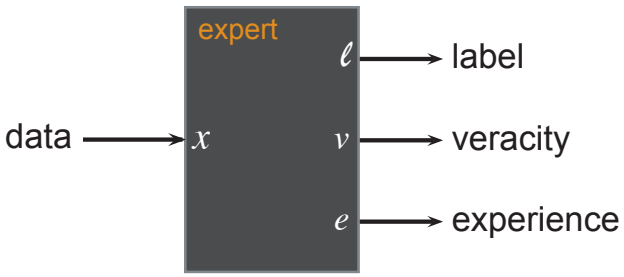


Figure 24. The expert classifier with a 3-tuple output representing a decision—the label ℓ, a veracity score $\mathfrak{v}$ and a quantification of experience $\mathfrak{e}$—in relation to a given feature x.

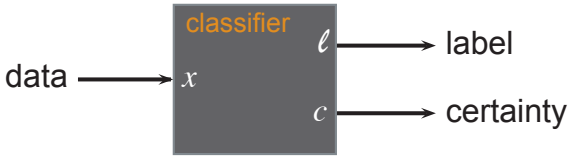


Figure 25. A classifier with a 2-tuple output representing a decision—the label ℓ and a confidence score $\mathfrak{c}$—in relation to a given feature x.

$[0, 1]$ and be increasing. Hence, consider

$$E_A(x) = \frac{\operatorname{card}(\mathcal{A}_1 \cap \mathsf{B}(x, r_x))}{\operatorname{card}(\mathcal{A}_1 \cap \mathsf{B}(x, r_x)) + \operatorname{vol}(\mathsf{B}(x, r_x))}.$$

Observe that this satisfies Axiom 1, since if $x \in \mathcal{A}_1$ then $r_x = 2\operatorname{dist}(x, \mathcal{A}_1) = 0$, thus, $\operatorname{vol}(\mathsf{B}(x, r_x)) = 0$ and $E_A(x) = 1$. Also, the other axioms hold true by construction.

5.1.1.3 Mapping expertise logic. Given the definitions of veracity and experience, we may now formally define the expert classifier.

Definition 34 *(Expert classifier) Given a classifier $A \subset \mathcal{X} \times \mathcal{L}$, its training data $\mathcal{A} \subset A$, and feature vector $x \in \mathcal{X}$, the expert classifier is the mapping $\widetilde{A}(x, A, \mathcal{A}) = (\ell, \mathfrak{e}, \mathfrak{v})$ where the 3-tuple $(\ell, \mathfrak{e}, \mathfrak{v})$ represents the expert's decision—the label $\ell \in \mathcal{L}$, the experience $\mathfrak{e}$, and the veracity $\mathfrak{v}$.*

It is possible to map the veracity and experience of an expert classifier (per Figure 24) to a single quantity of certainty $\mathfrak{c}$ (per Figure 25). Unfortunately, there is significant expres-

sion lost in this reduction. Such a mapping is not one-to-one as several veracity-experience pairs map to a certainty of $\mathfrak{c} = 0$. That is, there are many conditions where certainty $\mathfrak{c} = 0$, and predominantly among of them are expressions of arrogance.

For each class label $\ell \in \mathcal{L}$, we can relate a quality of certainty to corresponding veracities and experience. Figure 23 conceptualized this relation in veracity-experience space for the 2-class problem. Given $\mathcal{L} = \{false, true\}$, we mapped the veracity scores along the horizontal axis: The veracity $V_{true}(x)$ of the *true* class maps into the interval $[0, 1]$, and the veracity of the *false* class maps into the interval $[-1, 0]$ so that $-V_{false}(x) \in [-1, 0]$. Let $\mathfrak{c}$ denote certainty. The range of certainty is depicted within bright-to-dark grayscale in Figure 23(c) corresponding to the interval $[-1, 1]$ so that "Certain False"$= -1$, "Uncertain"$= 0$, and "Certain True"$= 1$ respectively. The exact form of certainty $\mathfrak{c}(x) = \mathfrak{c}(V_\ell(x), E_\ell(x))$ is domain dependent, but we can form the following axioms.

Axiom 6 *Certainty* $\mathfrak{c}$ *approaches* 0 *as experience* $\mathfrak{e}$ *approaches* 0.

Axiom 7 *Certainty* $\mathfrak{c}$ *approaches* 1 *as both veracity* $\mathfrak{v}_{true} \to 1$ *and experience* $\mathfrak{e}_{true} \to 1$ *for class true.*

Axiom 8 *Certainty* $\mathfrak{c}$ *approaches* -1 *as both veracity* $\mathfrak{v}_{false} \to 1$ *and experience* $\mathfrak{e}_{false} \to 1$ *for class false.*

Following these axioms, we have conceptualized a generic form of certainty in Figure 23(c) such that certainty $\mathfrak{c} = 0$ for the entire veracity interval where experience $\mathfrak{e} = 0$, then $\mathfrak{c} = 0$ for an increasingly smaller and smaller subinterval of the veracity interval as experience increases until, finally, $\mathfrak{c} = 0$ is a point at experience $\mathfrak{e} = 1$ and veracity $\mathfrak{v} = 0$. When mapped into this conceptualization, we see fuzzy logic corresponds to the veracity interval where $\mathfrak{e} = 1$ as shown in Figure 26(a). Thus, fuzzy logic presumes an expert encapsulates specific experience, and we conclude that fuzzy logic is most appropriate for data compression applications where this assumption holds true.

When we project data generalizations onto the 2-dimensional veracity-experience set, care must be taken to map onto the "V" illustrated in Figure 26(b). This restriction in expertise logic ensures veracity does not overstate skill where there is little or no justification.

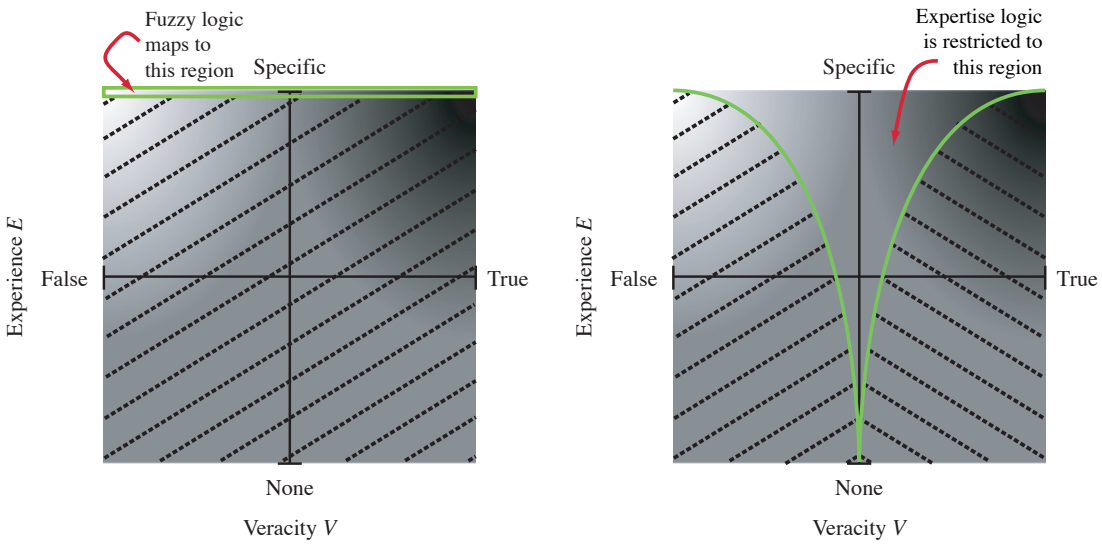


Figure 26. Mapping (a) fuzzy logic and (b) expertise logic onto the uncertainty set.

As we shall demonstrate in Chapter VI, the multilayer perceptron is a prime example of a data generalization that tends to overstate experience. A single-hidden-layer MLP compactly represents high order logic by overly partitioning a feature space. Methods such as back propagation and cross validation do nothing to regulate the ordering of partitions that do not contain training data. This leads to arbitrary and often misleading representations of veracity in regions where there are no training data. Where an unpopulated region is arbitrarily assigned a strong veracity ($|\mathfrak{v}| \to 1$), we say the classifier expresses arrogance.

5.1.2 Arrogance in Classification. Expert classifiers may be too bold or too timid in extending their training data to the larger operational set. Traditionally, we judge classifiers by assessing the amount of apparent memorization in classification. When a classifier memorizes—or overfits—training data, it is too timid in associating data near training data with the training data's label. Classifiers may also be too bold in assigning labels to data far from training data. We call this arrogance in classification. When a classifier has a tendancy to express an internal conviction toward certain answers without the necessary collaborating experience, one must trade off veracity versus experience before accepting a classifier's answer. We use an *arrogance curve* in veracity-experience space to

do this. To select an arrogance curve, we wish to define a function $g : [0,1] \to [0,1]$ such that the following properties hold true:

(a) g is defined on all of $[0,1]$;

(b) g is non-decreasing on $[0,1]$;

(c) $g(0) \geq 0$;

(d) $g(1) \leq 1$.

Examples of functions g are:

1. $g(s) = s$;
2. $g(s) = \sqrt{s}$;
3. $g(s) = s^q$ for some $q \in (0,1]$;
4. $g(s) = \log(s+1)/\log 2$;
5. $g(s) = \begin{cases} 0 & \text{for} \quad 0 \leq s < b \\ c & \text{for} \quad b \leq s \leq 1. \end{cases}$

Let function g define an arrogance curve such that, if experience $\mathfrak{e}$ is less than $g(\mathfrak{v})$ for veracity $\mathfrak{v}$ then the classifier expresses a certain amount of veracity without sufficient experience to back it up. The classifier is an *arrogant classifier* in this case. A decision maker should choose a threshold of arrogance that they can "live with"; thus, they choose a function g.

Definition 35 *(Arrogant Classifier) Let g be a function that satisfies the properties (a)-(d) above. Let $A : \mathcal{X} \to \mathcal{L}$ be a classifier with training set $\mathcal{A}$. Given a vector $x \in \mathcal{X}$ we say A is arrogant at x if $E_A(x) < g(V_A(x))$.*

Let us consider the arrogance curve given above in Example 5. We use this particular curve in the construction of 4-value logic [64], a crisp-set interpretation of expertise logic. We use expertise logic to represent 4-value logic in crisp sets in much the same way as fuzzy logic represents 3-value logic, but take note: As shown in Figure 27(b), 4-value logic is actually a 6-value logic with two forbidden values. To clarifiy, let us first examine fuzzy logic. As illustrated in Figure 26(a), fuzzy logic maps expert opinions into the interval $[0,1]$ along the veracity dimension. Given two thresholds $a, b \in [0,1]$ where $a < b$, fuzzy

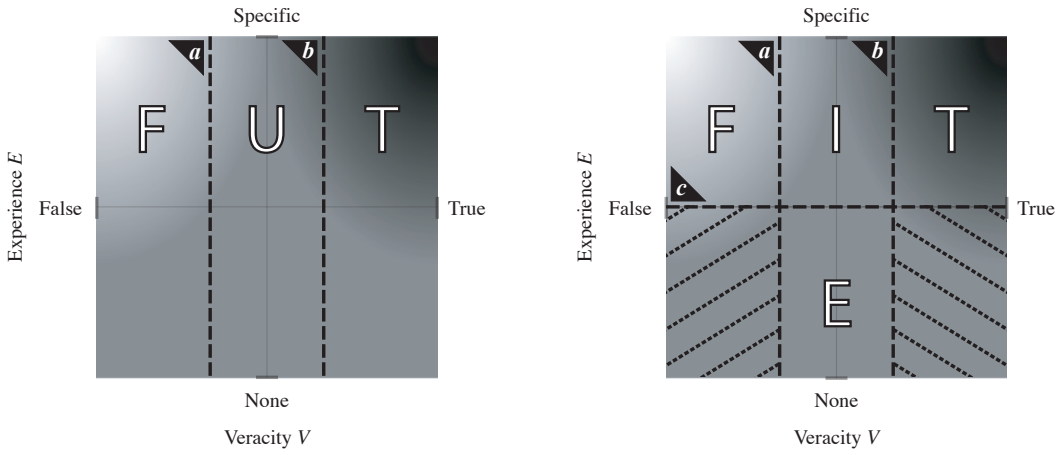


Figure 27. Three-value and four-value logic projected onto the uncertainty set. (a) Using two thresholds a, b, three-value logic divides the uncertainty set into 3 disjoint partitions. (b) Using three thresholds a, b, c, four-value logic divides the uncertainty set into 6 disjoint partitions: 4 subsets corresponding to 4-value logic and 2 subsets (the crossed-out regions above) corresponding to arrogance.

logic represents 3-value logic as relaxed intervals: False=$[0, a]$, Uncertain=(a, b), True=$[b, 1]$; these relaxed intervals and the thresholds a and b are depicted in Figure 27(a). Figure 27(b) shows a similar treatment for expertise logic with the following three exceptions: Expertise logic maps into two dimensions instead of one, (2) the veracity scores are mapped in to a $[-1, 1]$ interval instead of $[0, 1]$, and (3) our crisp logic treatment prescribes an additional threshold, c, applied along the second dimension. Given $a, b \in [-1, 1]$ where $a < b$ and the third threshold $c \in [0, 1]$, 4-value logic is represented in the relaxed intervals

$$\begin{aligned} \text{False} &= [-1, a] \times [c, 1], \\ \text{True} &= [b, 1] \times [c, 1], \\ \text{Uncertain Interpolation} &= (a, b) \times [c, 1], \text{ and} \\ \text{Extrapolation} &= [-1, 1] \times [0, c]. \end{aligned}$$

The Extrapolation interval splits further into 3 partitions: $[-1, a] \times [0, c]$, $(a, b) \times [0, c]$, and $[b, 1] \times [0, c]$. The second partition $(a, b) \times [0, c]$ expresses Uncertain Extrapolation (an appropriate expression) as shown in Figure 27(b); the first and third partition convey

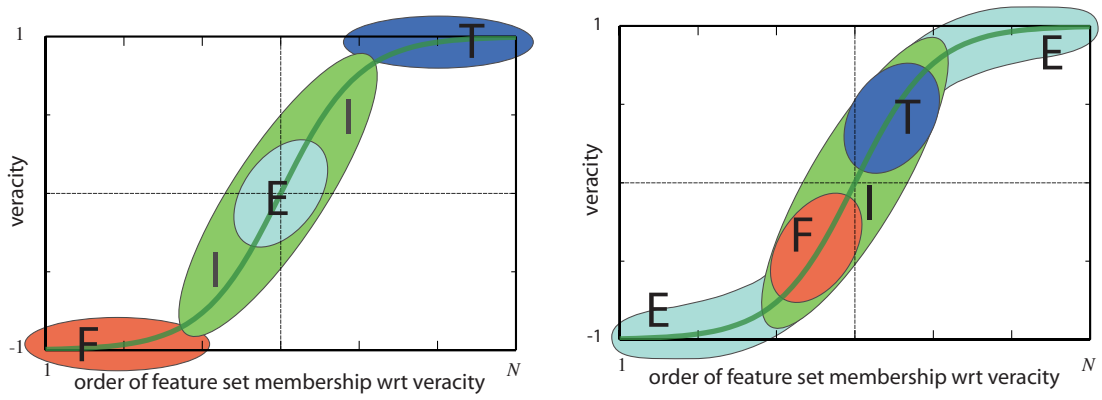


Figure 28. The above figures of merit illustrate appropriate and inappropriate expressions of expertise respectively. (a) The appropriate expression of expertise: Specific experience clusters in the asymptotic regions such where $|\mathfrak{v}(x)| \to 1$, while extrapolations cluster tightly where $\mathfrak{v}(x) = 0$. (b) Expressions of arrogant classification: Extrapolations dominate the asymptotic regions pushing specific experience away from the asymptotes toward $\mathfrak{v}(x) = 0$.

Arrogance. By forbidding a generalization to map into intervals that convey Arrogance, we restrict the mapping's output to 4-value logic: {False, True, Uncertain Interpolation, Uncertain Extrapolation} or $\{f, t, i, e\}$.

5.1.3 Depicting expertise in a figure of merit, the OVER curve. We assert that a good generalization isolates data, allowing extrapolations and uncertain interpolations to be easily separated from class data in a feature set. To illustrate good isolation, we recommend a figure of merit in which we order the feature set of a classifier in terms of its veracity scores and, then, cross reference this ordering to the classifier's experience scores. A figure of merit is "a parameter or characteristic of a machine, component, or instrument that is used as a measure of its performance" [86]. Here, we are using the term in the characteristic sense, and we shall call this characteristic the ordered veracity-experience response curve, or OVER curve.

For good separation and appropriate expression of expertise, we prefer a classifier which orders its domain similar to the OVER curve in Figure 28(a). Note, it is desirable for feature vectors near "false" training points to score veracities approaching -1 and to have

features near "true" points to score veracities approaching 1. To provide good isolation of classes, it is further desirable for feature vectors near both "true" and "false" exemplars to score veracities approaching 0 (that is, along the non-linear transition between asymptotes) and for features far from any exemplar to score veracities of 0.

Some classifiers merely separate training data from disparate classes and do not isolate the classes. These classifiers arbitrarily order extrapolated data (and may erratically order interpolated data depending on the classifier's complexity). In the arrogant classifier, significant regions of extrapolated regions receive veracity scores that are higher than scores assigned to truthed data. Instead of producing the desired domain ordering in Figure 28(a), the arrogant classifier orders the domain as depicted in the OVER curve of Figure 28(b). We will demonstrate instantiations of such an ordering using a 4-node-input/3-node-output multilayer perceptron in Chapter VI. For this demonstration, we chose an MLP trained using back propagation—a popular optimization technique—which separates class data without isolating the classes. Consequently, we expect the MLP to exhibit some degree of arrogance.

5.1.4 Summary of arrogance in classification. The training of good generalizations must mitigate both memorization and arrogance. Memorization is characterized as being too timid in associating new observations with previous experience. Arrogance is being too bold. In classification problems, memorization is traditionally assessed via error matrices and iterative error-based techniques such as cross validation. These techniques, however, do nothing to assess arrogance in classification. To identify arrogant classifications, we have proposed a confusion-based figure of merit called the ordered veracity-experience response curve, or OVER curve. To produce the OVER curve, one must employ expert classifiers. In this section, we defined the elements of both the expert classifier and the OVER curve and, in Chapter VI, we shall demonstrate their utility using the multilayer perceptron. But, for now, we shall complete this chapter with a presentation of fusion rules for expert classifiers.

5.2 Fusion rules for expert classifiers

Given a finite collection of classifiers trained on n-class data, one wishes to fuse the classifiers to form a new classifier with improved performance. Typically, the fusion is

performed on the output level using logical ANDs and ORs. We propose a fusion method based on the mitigation of arrogance among expert classifiers and the location of the feature vector with respect to training data. Given a feature x, if any one of the classifiers is a true expert on x, then that classifier should dominate the fusion. If the classifiers are confused at x, then the fusion rule should be defined in such a way to reflect this confusion. If a classifier is arrogant, then its results should not be considered and, thus, filtered out from the fusion process. We give this fusion rule based upon the metrics of veracity and experience.

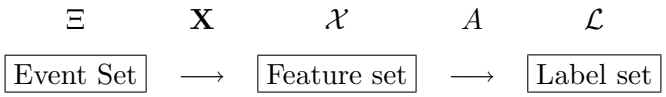
5.2.1 Classifier Theory. Classifiers categorize interesting events by mapping certain event observations to a predetermined class. Let Ξ be a nonempty event set. Let $\mathbf{X}$ be a random variable representing a sensor (with no noise) whose image is contained in a set of observations, i.e., the feature set $\mathcal{X}$. Thus, we can represent the data collection of a sensor as the function $\mathbf{X} : \Xi \to \mathcal{X}$ as illustratrated in the block diagram below.

$$\begin{array}{ccc} \Xi & \mathbf{X} & \mathcal{X} \\ \boxed{\text{Event Set}} & \longrightarrow & \boxed{\text{Feature Set}} \end{array}$$

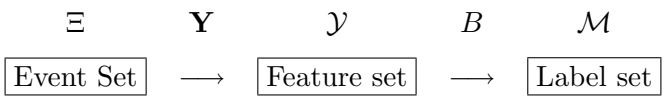
Formally, a classifier is a special relation that maps from a feature set to a decision set. Elements of a decision set may include class labels, situational qualifiers, and measures quantifying the ease of assigning a given decision. Consider the simple case where the decision set is a label set such that each classification—or element of the decision set—is a class label. Let classifier A be a function that maps a feature vector $x \in \mathcal{X}$ to a label $\ell \in \mathcal{L}$ where $\mathcal{L}$ is a label set. For a 2-class problem, the label set could be $\mathcal{L} = \{$false, true$\}$, $\mathcal{L} = \{$f,t$\}$, $\mathcal{L} = \{0, 1\}$ or $\mathcal{L} = \{-1, 1\}$. For other problems, the label set might be a continuum, e.g., $\mathcal{L} = \mathbb{R}$, or $\mathcal{L} = [-1, 1]$. We want a classifier to be a function so that an input feature vector maps to a single output label (possibly a vector). The corresponding block diagram follows.

$$\begin{array}{ccc} \mathcal{X} & A & \mathcal{L} \\ \boxed{\text{Feature set}} & \longrightarrow & \boxed{\text{Label set}} \end{array}$$

Composing the two functions yields the block diagram below.



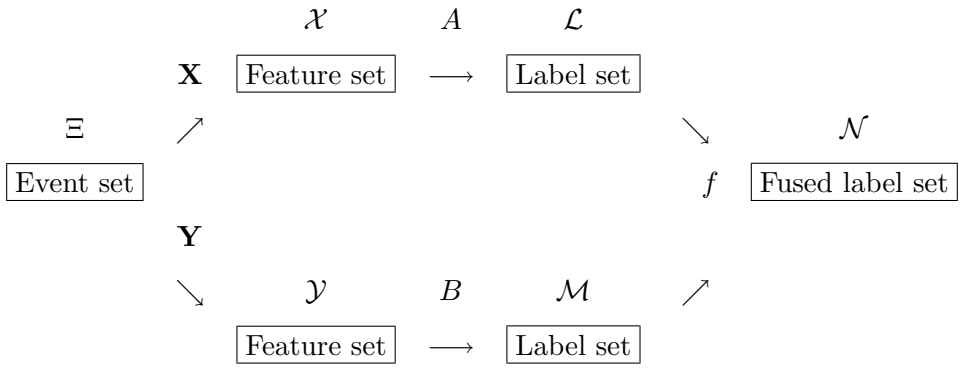
Suppose there is another random variable (or sensor) $\mathbf{Y}$ that also observes instantiations from the event set Ξ and produces a feature vector $y \in \mathcal{Y}$. In general, $\mathcal{Y}$ may be a different feature set from $\mathcal{X}$. Assume there is another classifier $B : \mathcal{Y} \to \mathcal{M}$, where $\mathcal{M}$ is some other (possibly different) label set. The composition block diagram is given similarly.



5.2.2 Classifier Fusion. Consider the case where the two sensors $\mathbf{X}$ and $\mathbf{Y}$ observe events occurring in the same event set Ξ as above. Assume they produce feature vectors in different feature sets $\mathcal{X}$ and $\mathcal{Y}$. In particular, assume $\mathbf{X} : \Xi \to \mathcal{X}$ and $\mathbf{Y} : \Xi \to \mathcal{Y}$. How can one combine or "fuse" two different classifiers designed from different training sets to produce results better than the individual classifiers separately? This is a basic question in classifier fusion theory. Usually, the fusion occurs on the outputs of the classifiers. To understand this, form Cartesian products and define the feature set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and label set $\mathcal{N} = \mathcal{L} \times \mathcal{M}$. Define the "system" classifier $C : \mathcal{Z} \to \mathcal{N}$ by

$$C(z) = C(x, y) = f(A(x), B(y))$$

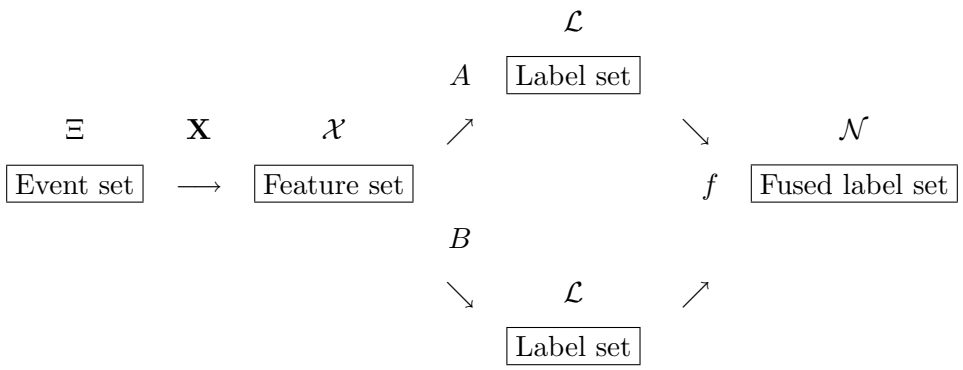
for $z = (x, y) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Thus, the block diagram



simplifies to the block diagram below.

$$\begin{array}{ccccccc} \Xi & & (\mathbf{X}, \mathbf{Y}) & & \mathcal{Z} = \mathcal{X} \times \mathcal{Y} & C = f(A, B) & \mathcal{N} = \mathcal{L} \times \mathcal{M} \\ \boxed{\text{Event set}} & \longrightarrow & & & \boxed{\text{Fused feature set}} & \longrightarrow & \boxed{\text{Fused Label set}} \end{array}$$

Assume that the feature sets are the same for the remainder of this paper, that is, $\mathcal{X} = \mathcal{Y}$. Given two classifiers A and B defined on the same feature set $\mathcal{X}$ and label set $\mathcal{L}$, we wish to fuse them into a new classifier C also defined on $\mathcal{X} \times \mathcal{L}$. Let f denote the fusion rule, or mapping. We wish this rule to be a function, that is, there is a unique output given the input classifiers A and B. Hence, we denote classifier C by $C = f(A, B)$ and the following block diagram.



One way to produce a fused classifier C is given by

$$C(x) = f_1(A, B)(x) = A(x) \wedge B(x) \text{ for every } x$$

where $\wedge$ denotes logical AND operation. That is, if classifiers A and B agree on a label ℓ, then the label for classifier C is ℓ. Another fusion example is

$$C(x) = f_2(A, B)(x) = A(x) \vee B(x) \text{ for every } x$$

where $\vee$ denotes the logical OR operation. That is, if A or B says that x is a target, then C will declare it a target. But these two fusion examples are based upon the labels only. We seek a fusion rule—to be denoted f_3—that not only uses the output labels but also considers the feature vector x as well.

5.2.3 Fusion rule for expert classifiers. Given two classifiers A and B, we wish to produce a fused classifier C based upon the veracities and experiences of both A and B. Let A and B be defined on feature set $\mathcal{X}$ with output label set $\mathcal{L}$, with respective training data $\mathcal{A}$ and $\mathcal{B}$. Given $x \in \mathcal{X}$, we define the fused classifier $C = f_3(A, B)$ as follows

$$
\begin{aligned}
&C(x) = f_3(A,B)(x) = \\
&\begin{cases}
A(x) & \text{if } A, B \text{ agree} \\
A(x) & \text{if } [A, B \text{ are confused}] \wedge [A \text{ not arrogant}] \wedge [B \text{ is arrogant}] \\
B(x) & \text{if } [A, B \text{ are confused}] \wedge [A \text{ is arrogant}] \wedge [B \text{ is not arrogant}] \\
A(x) & \text{if } \begin{cases} [A, B \text{ are confused}] \wedge [A \text{ is not arrogant}] \wedge [B \text{ is not arrogant}] \wedge \\ [A \text{ has more experience than } B] \end{cases} \\
B(x) & \text{if } \begin{cases} [A, B \text{ are confused}] \wedge [A \text{ is not arrogant}] \wedge [B \text{ is not arrogant}] \wedge \\ [B \text{ has more experience than } A] \end{cases} \\
A(x) & \text{if } \begin{cases} [A, B \text{ are confused}] \wedge [A \text{ is not arrogant}] \wedge [B \text{ is not arrogant}] \wedge \\ [A \text{ has the same experience as } B] \wedge [A \text{ has more veracity than } B] \end{cases} \\
B(x) & \text{if } \begin{cases} [A, B \text{ are confused}] \wedge [A \text{ is not arrogant}] \wedge [B \text{ is not arrogant}] \wedge \\ [A \text{ has the same experience as } B] \wedge [B \text{ has more veracity than } A] \end{cases} \\
0 & \text{if } \begin{cases} [A, B \text{ are confused}] \wedge [A \text{ is not arrogant}] \wedge [B \text{ is not arrogant}] \wedge \\ [A \text{ has the same experience as } B] \wedge [A \text{ has the same veracity as } B] \,. \end{cases}
\end{cases}
\end{aligned}
$$

Given g_A as the arrogance curve for classifier A, define arrogance indicator $\chi_A(x)$ such that

$$
\chi_A(x) = \begin{cases} 1 & \text{if } E_A(x) \geq g(V_A(x)) \\ 0 & \text{otherwise.} \end{cases}
$$

Similarly, given g_B as the arrogance curve for classifier B, let $\chi_A(x) = 1$ if $E_B(x) \geq g(V_B(x))$, otherwise $\chi_B(x) = 0$. Therefore, the fused classifier at x becomes

$$
\begin{aligned}
C(x) &= f_3(A,B)(x) \\
&= \operatorname{sign}\left[A(x)V_A(x)\chi_A(x) + B(x)V_B(x)\chi_B(x)\right. \\
&\qquad \left. -(A(x) \wedge B(x))V_A(x)V_B(x)\chi_A(x)\chi_B(x)\right] .
\end{aligned}
$$

Define the training set for the fused classifier to be $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$. Then we define the veracity and experience of the fused classifier C by the following:

$$\begin{aligned} V_C &= |\ell_A V_A \chi_A + \ell_B V_B \chi_B - (\ell_A \wedge \ell_B) V_A \chi_A V_B \chi_B| \\ E_C &= E_A \chi_A + E_B \chi_B - E_A \chi_A E_B \chi_B. \end{aligned}$$

Notice experience E_C is larger than $\max\{E_A, E_B\}$ for nonarrogant classification, even if the classifiers are confused. Veracity—in contrast—will decrease if there is confusion.

Table 3. Fusing experts A and B. Let a = arrogant, n = nonarrogant, $\alpha = v + \mu - v\mu$, $\beta = |v - \mu + v\mu|$, $\gamma = |-v + \mu + v\mu|$, $\delta = e + \varepsilon - e\varepsilon$. The symbol * denotes any value. The symbol ** denotes that the formula above will produce the label. The symbol *** that it depends on the values, specifically, if $|v + \mu - v\mu| > g(e + \varepsilon - e\varepsilon)$ then $\chi_C = 1$, otherwise $\chi_C = 0$.

	Aa Ba	Aa Bn	An Ba	An, Bn agree	An, Bn disagree	An, Bn disagree
ℓ_A	*	*	ℓ	ℓ	1	-1
V_A	*	*	v	v	v	v
E_A	*	*	e	e	e	e
χ_A	0	0	1	1	1	1
ℓ_B	*	ℓ	*	ℓ	-1	1
V_B	*	μ	*	μ	μ	μ
E_B	*	ε	*	ε	ε	ε
χ_B	0	1	0	1	1	1
ℓ_C	0	ℓ	ℓ	ℓ	**	**
V_C	0	μ	v	α	β	γ
E_C	0	ε	e	δ	δ	δ
χ_C	0	1	1	1	***	***

There exists a mapping $\mathbf{M}$ that maps the feature vector x, a classifier A, and its corresponding training set $\mathcal{A}$ to the 3-tuple $(\ell, \mathfrak{v}, \mathfrak{e})$. That is,

$$\begin{aligned} \mathbf{M}(x, A, \mathcal{A}) &= (\ell, \mathfrak{v}, \mathfrak{e}) \\ &= (\ell_A(x), V_A(x), E_A(x)). \end{aligned}$$

Given another classifier B with its corresponding training set $\mathcal{B}$ then

$$\mathbf{M}(x, B, \mathcal{B}) = (\ell_B(x), V_B(x), E_B(x)).$$

In Table 3, we give the values for the 3-tuple $\mathbf{M}(x, C, \mathcal{C}) = \mathbf{M}(x, f_3(A, B), \mathcal{A} \cup \mathcal{B}) = (\ell_C(x), V_C(x), E_C(x))$ given the values for $(\ell_A(x), V_A(x), E_A(x))$ and $(\ell_B(x), V_B(x), E_B(x))$. For brevity of notation, we have suppressed the x dependence in the table.

5.2.4 Measure of Performance. It is possible to quantify the improvement of the fusion rule f_3 over standard rules like f_1 and f_2 by viewing their corresponding Receiver Operator Characteristic (ROC) curves [2, 4]. However, ROC curves are based on error, not confusion. Thus, a ROC curve relates a classifier's performance with respect to its training set and without particular respect to its operational set. For a confusion based technique, we recommend the figure of merit first discussed in Section 5.1.3 and which we shall demonstrated in Chapter VI.

5.3 Summary

The goal of expertise logic and 4-value logic is to construct expert classifiers that realistically model experience across an entire domain. Expertise logic gives us the degree of freedom to separate uncertain decisions based on vague experience from decisions based on no experience. For data generalizations, we must anticipate encountering regions of the domain where experience is incomplete and, in these regions, avoid making arrogant classifications. The power of expertise logic rests in the assertion that a classifier embodies a hypothesis of what is considered near training data and what is considered far. By contrasting the hypotheses of multiple experts, feature space can be partitioned into regions of certainty, regions of uncertainty due to interpolation conflicts, and regions of high uncertainty due to extrapolation.

VI. Arrogance in the multilayer perceptron

So far in this dissertation, we have presented new pattern recognition concepts—specifically in the area of classification—and proposed a new diagnostic technique, the OVER curve, that allows one to determine when a classifier is being arrogant. We proposed the OVER curve—a confusion-based figure of merit—to augment established error-based metrics in order to properly regulate generalizations safely between the extremes of arrogant classification and memorization.

Arrogance in classification is clearly an undesirable attribute. In this chapter, we demonstrate that single-hidden layer perceptrons trained by back propagation are arrogant in their classification of certain regions in a feature space. When a classifier has been optimized to perform well on truthed data, one hopes that the classifier will perform well on operational (unknown) data. In turn, if a classifier performs well on operational data, then we say that the classifier generalizes well. For a multilayer perceptron (MLP), we detail where in a feature set the classifier generalizes poorly due to arrogance. Our analysis yields new evaluation tools for the MLP based upon combinatorial geometry techniques that have existed for more than a decade but are just now being exploited in the computational intelligence community.

Quantifying MLP generalization performance has several approaches, as the literature shows, including cross validation and bootstrapping. In a marked departure from these stochastic approaches, we use 4-value logic to quantify generalization performance. Four-value logic is a simplified expression of both veracity and experience. The benefits of our 4-value logic technique over other iterative and stochastic methods include the following: (1) Our technique evaluates the generalization of an MLP after model selection, requiring only the classifier's weights and biases and the data used to train these parameters. Iterative methods require that the evaluation of generalization take place over multiple training runs prior to the selection of a specific model; then, once a model is selected, the estimation of generalization for the selected model is optimistic. (2) The results of our evaluation do not change based on the order that data are presented. (3) Stochastic techniques exercise very

little use of the domain—that is, the measure of the set of feature vectors used is zero. For our technique, the measure of feature sets used is positive.

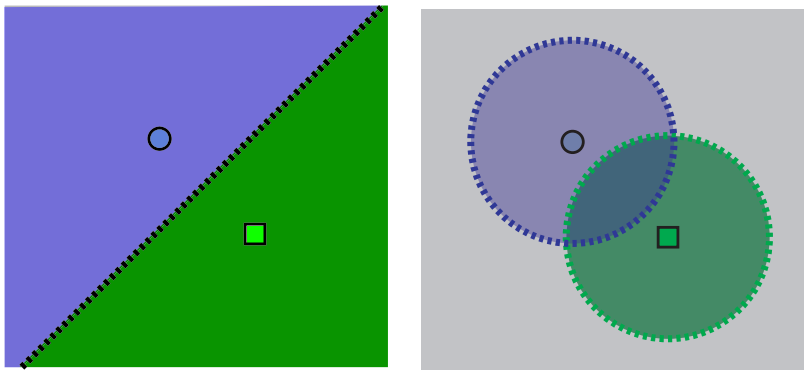
In general, we expect arrogant classification to occur in classifiers that separate but do not isolate data. The multilayer perceptron is one such classifier, and in this chapter we shall present the mathematical theory and algorithms necessary to identify arrogance in a an multilayer perceptron. We shall also provide demonstrations of our evaluation algorithms using the classic XOR and Fisher iris problems. To mitigate arrogance in classification, we advocated the use of isolation strategies which strike a reasonable balance between being too timid and too bold in assigning class labels, and we outline an isolation strategy for the multilayer perceptron in the next section.

6.1 Observations on the multilayer perceptron

We have selected the multilayer perceptron trained by backpropagation as an example of an arrogant classifier. The first reason should be obvious from our discussions in the previous chapter. That is, we believe arrogance in classification is likely to occur when training criteria do not require isolation of class data, and successful training in backpropagation need only separate class data. Thus, we suspect that backpropagation must be augmented as a training technique for generalizations in order to prevent arrogant expressions in the final solution. A second reason stems from a new observation we have made in this research regarding the multilayer perceptron: Overfitting, or memorization, in the MLP corresponds to the over-isolation of training data. It falls out from this second point that it is easy to apply the OVER curve to the multilayer perceptron—easy, that is, relative to other classifiers.

6.1.1 Arrogance in MLP simple solutions. In preparing MLP classifiers, there are many techniques that seek to limit a network's complexity[1] in an effort to avoid memorization. The implication is that a generalization should only be as complex as it needs to be to implement a training set and no more complex. However, if we apply the principles of Ockham's razor to multilayer perceptrons, we can run into problems. Ockham's razor is

[1]We define complexity as the total variation of a function.



(a) (b)

Figure 29. Separation versus isolation in 2-dimensional space (a) Two points may be separated by a line (parameter set: 1 slope and bias). (b) The same points are isolated by two circles (parameter set: 2 radii and 2 biases).

the "scientific rule that simple explanations should be preferred to more complicated ones, and that the explanation of a new phenomenon should be based on what is already known" [86]. One can be led astray by taking the first part of this rule too much to heart as we find the simplest solution is not necessarily the best solution for generalization purposes. A simple solution that implements a training set may still be a poor generalization if that solution expresses arrogance. Recall, as discussed in Chapter V, poor generalization occurs as a result of memorization and/or arrogance. Other researchers have found a strong correlation between complexity and memorization [68] but, empirically, we have not found the same correlation between complexity and arrogance. Nor do we expect such a correlation. Instead, we believe that arrogance occurs in architectures that separate training data but do not isolate the data. Since a classifier of high complexity may fail to isolate data, it is possible to generate classifiers that exhibit both memorization and arrogance.

If we apply a technique—such as pruning—to reduce the complexity of a neural network, it may reduce memorization in the solution but do nothing to address arrogance and, in fact, may exacerbate such expression. Architectures that merely separate data tend to be much less complex than architectures that isolate data. Figure 29 gives a rudimentary 2-dimensional example where two points may be separated by a line (two parameters) but are isolated by two circles (four parameters). When pruning alters an isolation architecture to a separation architecture, arrogance will result and generalization degrades.

6.1.2 Convergence speed versus generalization. Two key aspects of training an MLP are (1) the ability of the solution to generalize well and (2) the convergence speed—i.e., the time or computational effort it takes to converge to a solution. Unfortunately, certain conditions that serve to speed up convergence have negative effects on the generalization capability of the attained solution. For example, separation architectures in general have faster convergence speeds than isolation architectures with respect to the size of the training set. The reasons for the disparity in speed is that the training rules that generate separation architectures tend to be less complex than training techniques for isolation architectures and more conducive to optimization.

The benefits of converging quickly to a separation architecture that implements a dataset can be significant, but the danger of arrogance is real if nothing further is done to the architecture. To optimize both convergence speed and generalization ability of a neural network solution, we advocate a 3-fold learning technique:

1. The training of a complex separation architecture,
2. A modification step to prune the separation architecture to simpler model, and
3. The conversion of the simplified model to an isolation architecture.

When building a multilayer perceptron solution, the third step can be achieved by (1) finding the populated polytopes which are unbounded, which can be shown to be a polynomial time algorithm[2] of quadratic degree; (2) converting the unbounded, populated polytopes to bounded polytopes, a well-understood problem in combinatorial geometry; and (3) adjusting the network's bias. Alternatively, the conversion to an isolation architecture can be achieved, as we shall demonstrate in Section 6.3.1.3, by wrapping the entire training set in a convex hull and adding each hyperplane of the convex hull as a strongly weighted perceptron in the first hidden layer.

[2]See the discussion on local max and local min chambers in Section 6.3.2 and Theorem C.6 on unbounded central chambers in Appendix C. The gist of the proof is based on the fact that, if an arrangement of hyperplanes forms an unbounded central chamber, the arrangement contains only one local min chamber. We show in this chapter that any chamber can easily be converted to a central chamber. From there, it is possible to write a simple routine to walk from the central chamber to any local min chamber using the adjacency rules discussed in Section 6.3.1.2. If the discovered local min chamber tests true to being the only local min chamber in the arrangement (see Lemma C.3), then the central chamber is unbounded.

The necessity of the second step in our 3-fold learning technique—which calls for pruning the complex separation architecture—stems from the fact that training an MLP is a constraints problem. In Constraints Programming, we find the simplest solution is among the hardest solutions to find. Imagine trying to cast 25 roles in a play from a pool of temperamental actors. Some of the actors hate one another and refuse to be in the same production. Others will participate only if their friends are hired. This problem formulation is called a *satisfiability problem* [46]. Now, imagine that you are the casting director and the play's producer has preselected actors to fill certain roles. You find the more actors the producer preselects, the less degree of freedom you have in filling the vacant roles from the remaining pool of actors and the cleverer you must be in order to retain a workable mix of temperaments. The satisfiability problem has now become a *constraints problem* as some of the problem's variables come preselected in the given instantiation of the problem formulation. In the example above, the variables of this problem formation are the roles in the play, and the constraints are the actors that must fill certain roles as stipulated by the producer. Note that the constraints that must be met (i.e., exactly how many and which ones) are known only when an instantiation of the constraints problem is formed.

Constraints problems have a range of difficulty depending on the set of constraints that must be met [67]. Let γ be the ratio of constraints to variables. Having observed various instantiations of a constraints problem (say your production has gone through a lot of producers, each with a different idea on how various roles should be cast), we can plot the time required to find a solution versus the ratio γ. From the empirical evidence collected by scientists on constraints problems, we can expect our plot to manifest an apparent phase shift—such as in Figure 30—exhibiting a behavior such that, as the ratio γ is increased, the problems suddenly change from easy to impossible.

The training of a multilayer perceptron qualifies as a constraints problem. An MLP chunks a feature set into multiple polytopes. Some of these polytopes contain training data; others do not. The polytopes that are populated with training data are the constraints of the MLP training problem. Given a training set, as the ratio of populated chambers to unpopulated polytopes increases, the number of potential solutions that implement the training set decreases until there are no solutions that can implement the training set.

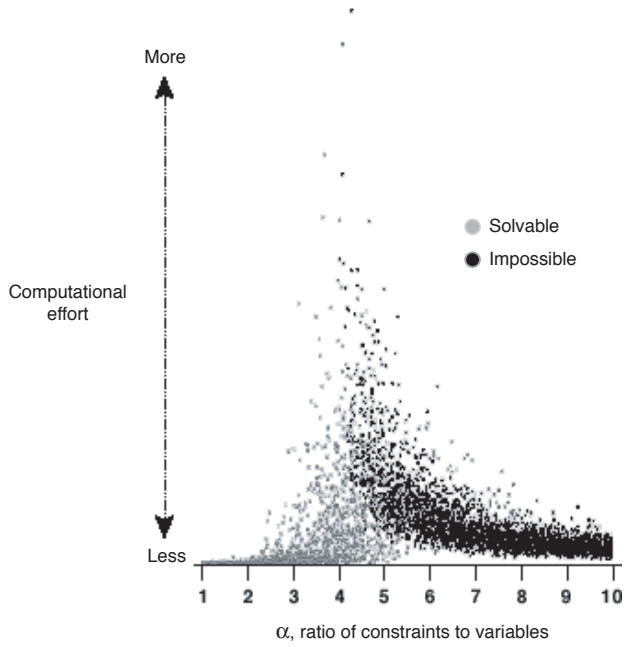


Figure 30. An example of the phase shift manifested in a Constraints Problem as reported in an article on the research of Remi Monasson et al [46].

In order to converge to a MLP solution in a reasonable amount of time using back propagation, we find that one may have to risk a complex architecture. For our discussion, let a complex architecture be one that can be simplified and still implement the training set. If the added complexity is not managed effectively, then memorization and, thus, poor generalization are likely to occur. The greater the ratio γ of unpopulated polytopes to populated polytopes there are, the more solutions exist for a given MLP architecture. There are two ways of increasing this ratio:

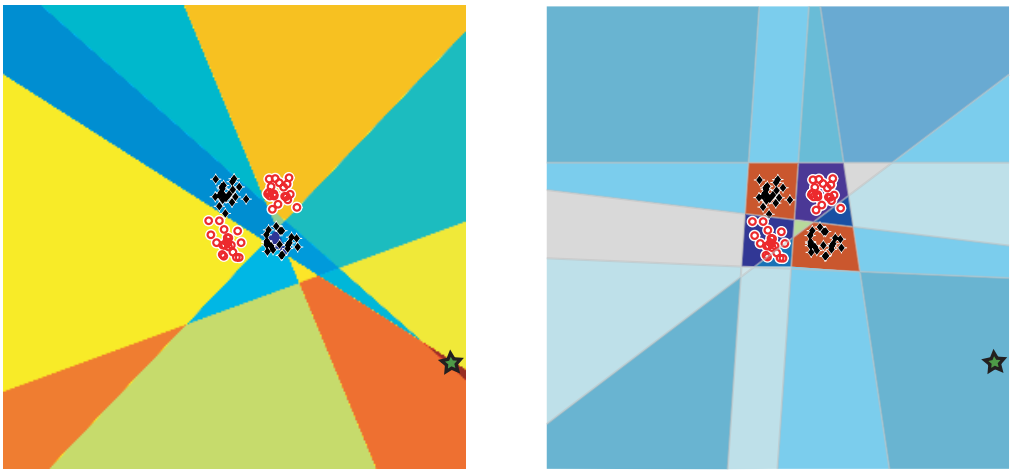
1. For a given architecture, select a set of parameters that clusters more points in fewer polytopes, or
2. Increase the number of total polytopes by choosing a more complex architecture (i.e., adding more first-hidden layer nodes).

The first method corresponds to an initialization problem[3]; the second method is, unfortunately, the one more often used. We say unfortunately because this method can exacerbate memorization by creating smaller polytopes which hold less training points—thus, over-isolating training points. In a architecture that over-isolates, it is easier to find the solutions that result in over-fitting, or memorization, as they outnumber the solutions do not over-fit. When an MLP is over-populated with first-hidden layer nodes, memorization can be so bad that every data point in the training and evaluation set is isolated in its own polytope. Selecting a good generalization is highly unlikely in such a case.

6.1.3 Overfitting in the MLP. The multilayer perceptron is capable of compactly representing high order logic, but it does so at the expense of generality. To represent functions of high complexity, a multilayer perceptron overly partitions a feature space creating disjoint subsets, many unpopulated by training data. The unpopulated subsets can be assigned arbitrary values—whatever works to fit the overall function to populated partitions—without affecting optimization objectives that rely on sampled data to regulate training. Consequently, optimization routines such as back propagation and cross validation promote single hidden-layer MLPs that form functions of far greater complexity than the training data warrants and with no special claim to good generalization. Cross validation, for one, promises to find appropriate generalizations by preventing memorization, i.e., the over-fitting of training points. Cross validation is a data-centric method based on the hold-out method:

1. Split training data into two sets—a training set and an evaluation set.
2. Fit training set using an iterative optimization routine (e.g., back propagation) with a slow learning rate.
3. Compute the validation error trend from the evaluation set.
4. Stop fitting when the validation error starts to increase for the evaluation set.

[3]By initializing a multilayer perceptron such that the class points are dispersed among only a few populated polytopes, we expect—on average—to improve convergence speed. This improvement cannot be guaranteed especially for simple architectures where there are few if any solutions.



(a) (b)

Figure 31. The two multilayer perceptrons trained to 100% classification, and their solutions are mapped over a portion of the 2-dimensional feature set. (a) A single hidden-layer MLP trained by back propagation and regulated by cross validation rules, and (b) a three hidden-layer MLP constructed using a modified support vector machine rule base in support of four-value logic. At the extrapolated evaluation point denoted by the star $\star$, the single hidden-layer MLP does not map onto the "V" of the expertise map while the three hidden-layer MLP models experience more appropriately.

Proponents of cross validation promise the technique will promote good generalization, but in multilayer perceptrons—where over-fitting equates to over-isolating training—it cannot. This is because the method has inadequate controls over memorization and no control over expressions of arrogance. Indeed, the method is doomed to fail on both counts when a complex architecture is being tested because (1) cross validation does not optimize the membership of populated polytopes implemented by the architecture and has no mechanism to recognize over-isolation, and (2) it does not regulate the ordering of unpopulated polytopes or provide a mechanism to select an appropriate generalization with respect to these unpopulated polytopes.

Because of its propensity to over-isolate training data, the multilayer perceptron must be carefully schooled into expertise logic. Unfortunately, current "state of the art" training techniques do not properly restrict the unpopulated partitions of the multilayer percep-

tron and expressions of arrogance result. To illustrate, note the renderings of two MLPs in Figure 31. Both have been trained to 100% classification: (a) A single hidden-layer MLP is trained by back propagation and regulated by cross validation rules, and (b) a 3-hidden-layer MLP is constructed using a modified support vector machine rule base for 4-value logic treatments. In these figures, veracity is depicted as color and experience is gauged by distance from the depicted training data. At an extrapolated evaluation point denoted by the star $\star$ in the figure, the single hidden-layer MLP maps to "False" in a region where experience is rated as "None" while the 3-hidden-layer MLP models veracity more appropriately in relation to its experience. Due to its blatantly unbalanced representation of veracity and experience, we say the single-hidden-layer MLP expresses arrogance.

6.1.4 Summary. Arrogance in classification is one of three new observations we have made in this research regarding the multilayer perceptron. The other two observations are as follows:

1. Overfitting in the MLP corresponds to the over-isolation of data points.
2. The training of an MLP is a constraints problem.

These two points follow from an understanding that the MLP partitions a feature space via the arrangement of hyperplanes implemented in the network's first hidden layer, and it falls out from the first point that the application of the OVER curve (introduced in Chapter V) is relatively easy to implement with respect to other classifiers. Using a confusion-based measure to delineate the experience of the MLP across its domain, our analysis yields new evaluation tools for the MLP and paves the way for other classifiers. The specific measure of experience for the multilayer perceptron is based upon combinatorial geometry techniques which we discuss the next three sections. Combinatorial geometry enable us to observe the ordering of a feature set as implemented by a multilayer percetron and allow us to prune and otherwise manipulate the network in an informed manner. Finally, we shall present how to include an ordered veracity-experience response treatment as part of an investigation into memorization and arrogance in specific MLP solutions, not just architectures.

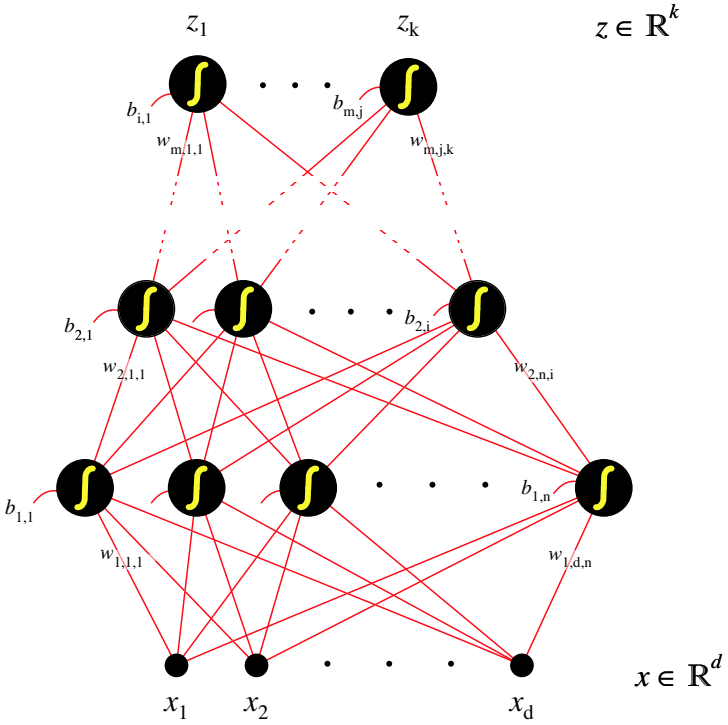


Figure 32. A multilayer perceptron.

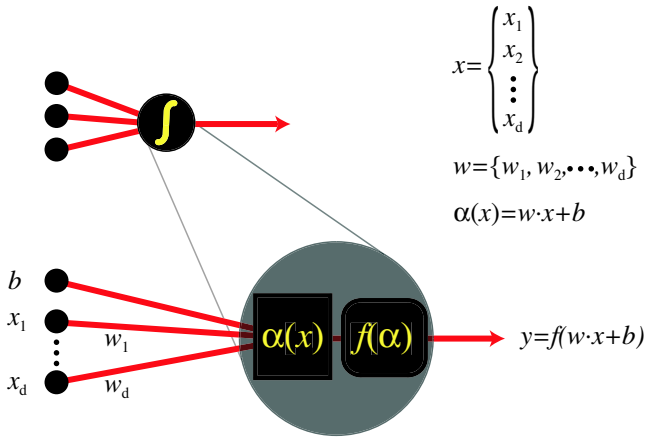


Figure 33. A perceptron. Each input x_i is multiplied by a synaptic weight w_i. The weighted inputs are summed together, and a bias term b is added. Finally, a non-linear activation function f is applied to the sum.

6.2 The multilayer perceptron as ordered arrangement of hyperplanes

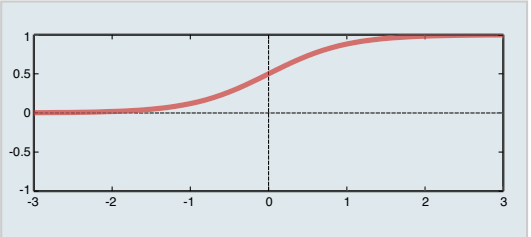
For this application of the ordered veracity-experience response curve, we shall derive an original algorithm that orders the domain of a classification problem in response to a multilayer perceptron (MLP) solution. First, we construct a partially ordered set over the problem domain based on the arrangement of hyperplanes implemented by the MLP's first layer of weights and biases. Then, we map the partially ordered set against a 4-value logic representation derived in response to the MLP's training set. Constructing the partially ordered set from the MLP's weights and biases is the first order of business; and, for this, we enter into a lengthy discussion on the multilayer perceptron and its manipulation of a domain in terms of combinatorial geometry.

The multilayer perceptron[4] is a feed-forward artificial neural network composed of perceptrons. First proposed in 1943 by McCulloch and Pitts [66], the perceptron[5] is a generalization capable of discriminating two linearly separable classes. Figure 33 illustrates a perceptron's components. Given feature set $\mathcal{X} = \mathbb{R}^d$ and output set $\mathcal{Y} = \mathbb{R}$, these components are:

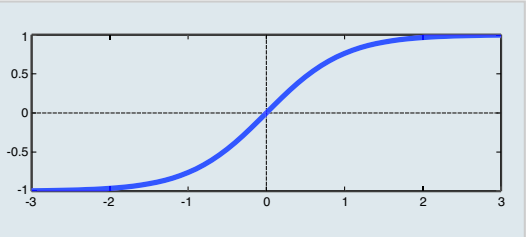
1. Input vector $x \in \mathcal{X}$,
2. Weight vector $w \in \mathbb{R}^d$,
3. Bias $b \in \mathbb{R}$,
4. Activation function $f : \mathbb{R} \to \mathcal{Y}$, and
5. Output vector $y \in \mathcal{Y}$.

A perceptron transforms its input vector $x = (x_1, \ldots, x_d)^T$ into a scalar via a composite of two transformations. The first transform is the affine map $\alpha : \mathcal{X} \to \mathbb{R}$. Given the input column vector $x \in \mathcal{X}$, a weight row vector $w \in \mathbb{R}^d$, and a scalar bias $b \in \mathbb{R}$, the affine mapping is $\alpha(x) = w \cdot x + b$. The second transform is a nonlinear activation function $f : \mathbb{R} \to \mathcal{Y}$ that is composed with α so that output $y = f(\alpha(x))$. Forms of this composition include

[4]See Figure 32.
[5]See Figure 33.



(a) Sigmoid



(b) Hyperbolic Tangent

Figure 34. The Sigmoid and Hyperbolic Tangent transformation functions.

- Sigmoid $f(\alpha) = \frac{1}{1+e^{-\alpha}}$,
- Hyperbolic tangent $f(\alpha) = \tanh(\alpha)$, or
- Tansig $f(\alpha) = \frac{2}{1+e^{-2\alpha}} - 1$.

Two popular activation functions—the sigmoid and hyperbolic tangent—are shown in Figure 34. The tansig function is mathematically equivalent to the hyperbolic tangent but has the advantage of being computationally faster when implemented on most systems. The perceptron separates a feature set into two halves along the hyperplane $\{x \in \mathcal{X} : \alpha(x) = 0\}$. Separation is achieved by selecting a threshold $y_0 = f(0)$ and, then, assigning "true" to outputs $y \geq y_0$ and "false" to $y < y_0$. As seen in Figure 34, the choice of activation function affects the range of y and the value of threshold y_0. When the activation function is the sigmoid function, output $y \to 1$ in the "true" halfspace and $y \to 0$ in the "false" halfspace; threshold $y_0 = 0.5$ separates the halfspaces. Alternately, if the activation function is the hyperbolic tangent or tansig function, $y \to 1$ in the "true" halfspace, $y \to -1$ in the "false" halfspace, and threshold $y_0 = 0$. [13, 66, 87, 85]

The MLP organizes its perceptrons, or nodes, into layers so that the nodes in one layer connect forward only to nodes occupying the next layer. Layering adds order to the mapping. At a minimum, an MLP requires an input layer, one hidden layer and an output layer. Additional hidden layers may be added. In 1989, Cybenko proved that a single-hidden-layer multilayer perceptron is a universal mapping capable of estimating any function in the mean-squared sense given a sufficient number of hidden layer nodes [23]. Since this proof, several respected researches [8, 10] have supposed that Universal Mapping = Valid Generalization, but we can challenge this assumption by considering the fact that

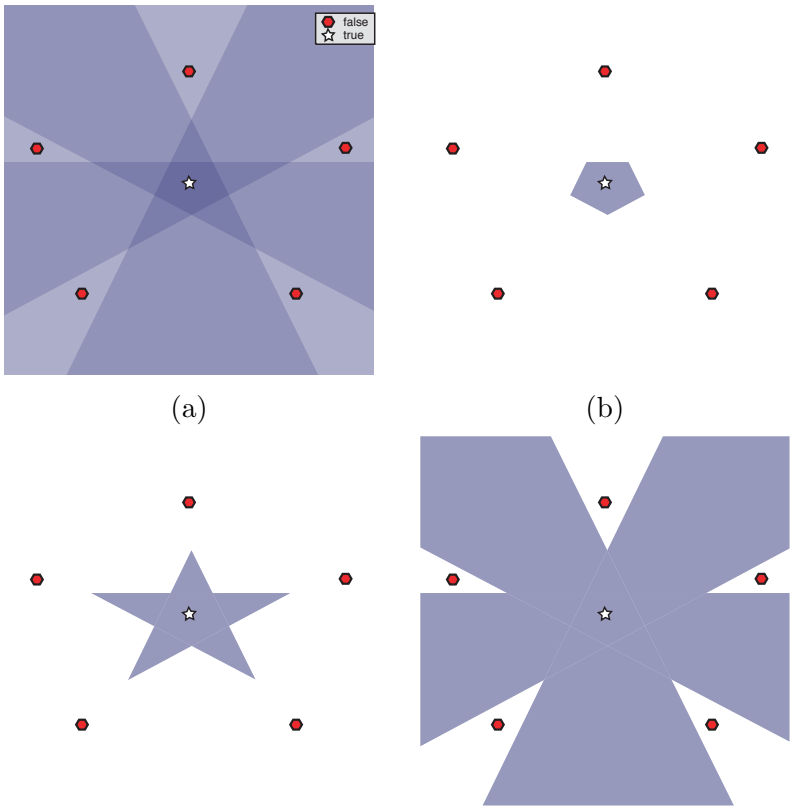


Figure 35. Three variations of an MLP based on thresholding. The superposition of the MLP's sigmoidal nodes (a) without thresholding applied, (b) with the highest threshold applied, (c) with a lower threshold applied, (d) with the lowest. Note how the logical complexity of the neural network varies significantly with only simple adjustments to the output node's bias.

variations in an MLP's thresholds may alter its functional form without affecting its mean-squared error.

Simple variations in an MLP's thresholds may yield significant variations in overall complexity even while the function adheres to certain localized constraints. It is difficult to select a valid generalization from the MLP solutions that implement a dataset in the mean-squared sense given the wide range of complexity among these solutions. Consider a single-hidden-layer MLP for a 2-dimensional problem. We wish to dichotomize a simple dataset and view manipulations of the network by projecting its solution set back onto its 2-dimensional domain. Let the multilayer perceptron have 5 hidden nodes which form a

pentagon-shaped convex hull that contains a "true" point. Let the parameter set of the hidden-layer nodes, $[W_{1,n} : b_{1,n}]$, remain fixed such that the nodes' outputs—generating the five halfspaces $\{H_1, H_2, H_3, H_4, H_5\}$—project into the domain in relation to the "true" point and five "false" points as depicted in Figure 36. Given that each output-layer weight $W_{2,n} = 1$ and the output bias $b_2 \in \mathbb{R}$, there are essentially 3 solutions that implement the simple dataset and yet uniquely partition the feature set. Each unique solution can be implemented by a particular range of biases

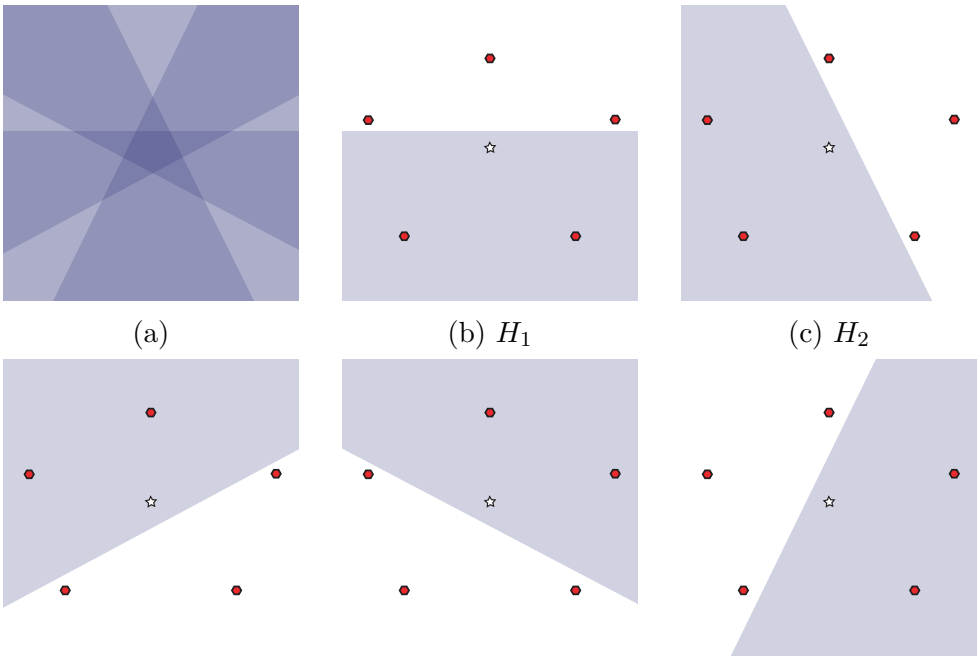
$$b_{2,1} \in (-5, -4],$$
$$b_{2,2} \in (-4, -3],$$
$$b_{2,3} \in (-3, -2].$$

We select a bias from each interval to construct 3 output weights

$$[W_{2,1} : b_{2,1}] = [1, 1, 1, 1, 1, -4.5]$$
$$[W_{2,2} : b_{2,2}] = [1, 1, 1, 1, 1, -3.5]$$
$$[W_{2,3} : b_{2,3}] = [1, 1, 1, 1, 1, -2.5]$$

such that each implements a unique partitioning of the feature set. The solutions implemented by these output vectors are shown in Figure 35(b), (c), and (d) respectively. In Figure 35(b), the weight vector with the highest-magnitude bias produces the simplest and most reasonable decision boundary, a convex hull. In Figure 35(c), the next lower-magnitude bias produces a star-shaped decision boundary, a complex interpolation of "true". In Figure 35(d), the lowest magnitude bias produces a complex decision boundary—shaped like a shining star—yielding unreasonable, arrogant classifications.

For regularization purposes, let us impose Ockham's razor to whittle down the 3 solutions above to 1. The solutions portrayed in Figure 35 differ only in the choice of threshold, but these slight adjustments in the output bias dramatically alter the complexity of the multilayer perceptron's decision boundary. Below, we represent the 3 variant solutions in terms of the intersects $\cap$ and unions $\cup$ of the outputs $\{H_1, \ldots, H_5\}$ from the MLP's five



(a) (b) H_1 (c) H_2

(d) H_3 (e) H_4 (f)H_5

Figure 36. We are given the outputs of the fixed hidden nodes are given as five halfspaces superimposed in (a) and illustrated separately in (b)-(f).

hidden nodes.

$$
\begin{aligned}
g([w_{2,1} : b_{2,1}]) &= H_1 \cup H_2 \cup H_3 \cup H_4 \cup H_5 \\
g([w_{2,2} : b_{2,2}]) &= (H_1 \cap H_2) \cup (H_2 \cap H_3) \cup (H_3 \cap H_4) \cup (H_4 \cap H_5) \cup (H_5 \cap H_1) \\
g([w_{2,3} : b_{2,3}]) &= (H_1 \cap H_3) \cup (H_2 \cap H_4) \cup (H_3 \cap H_5) \cup (H_4 \cap H_1) \cup (H_5 \cap H_2)
\end{aligned}
$$

From the above equations, we see that the first output weight vector $[w_{2,1} : b_{2,1}]$ implements a first order logic solution and the second and third output weight vectors implement second-order logic solutions. So, again we see that the first output weight vector implements the simplest solution of the three. Since it is the simplest and appears to be reasonably based on known data, the first solution is the preferred solution per Ockham's razor. The second weight vector $[W_{2,2} : b_{2,2}]$ yields a solution that is arguably too complex for its purpose, while it is clear that the third weight vector $[W_{2,3} : b_{2,3}]$ produces an overly complex solution that is excessively generous in its association of new data to the known "true" datum.

When training an MLP using data-centric techniques, we must carefully select biases to properly regulate the final solution. Notice, for instance, that the magnitude of the augmented vector $[W_{2,1} : b_{2,1}]$—the one that implements the preferred solution—is largest of the 3 vectors. The magnitude of the weight vector is of interest because backpropagation tends to incrementally increase the magnitude of an MLP's weight vectors during training. This means, when backpropagation is used, training is apt to fix on a output weight vector with a low-magnitude bias before it finds a similar vector with a higher-magnitude bias to implement the training set. As a result, backpropagation training tends to favor MLP solutions with overly generous thresholds over preferable solutions with more conservative thresholds.

Using combinatorial geometry, it can be shown that the shattering capability[6] of a multilayer perceptron with a single hidden layer is limited [18, 73]. This assertion appears contrary to Cybenko's proof that a single-hidden layer MLP is a universal mapping capable of estimating any function given a sufficient number of hidden layer nodes. Though Cybenko is technically correct, it is important to understand that the MLP approximates a function in the mean-squared error sense and, as such, the approximation is highly susceptible to memorization and extrapolation errors as illustrated above. In addition to the above commentary, it has been noted [82, 28] as a practical matter that a single-hidden layer MLP requires an unreasonable number of hidden layer nodes to implement datasets of second-order logic or greater—such as the spiral data problem in Chapter IV—resulting in functions of unwieldy geometric complexity at least within today's computational resources. Bottom line: Cybenko's proof that a single hidden layer MLP with sufficient nodes approaches any function in the mean-squared error sense holds true at the expense of generalization.

In summary, due to variations caused by simple changes in thresholding, MLP generalizations fit by mean-squared error techniques are suspect. Remedies to such techniques may be achieved by moving from data-centric evaluations of data generalizations to evaluations that are set-centric such as the confusion-based techniques we propose.

[6]Shattering is the ability of a logical architecture to reorder a dataset into all possible partial orderings.

6.2.1 The first hidden layer's arrangement of hyperplanes. The set of all nodes in the first hidden layer of a multilayer perceptron corresponds to an arrangement of hyperplanes in the feature set. Each perceptron $f \circ \alpha_{m,n}$ in an MLP's first hidden layer $m = 1$ represents a hyperplane in $\mathcal{X}$ defined by $h_n = \{x \in \mathcal{X} : \alpha_{1,n}(x) = 0\}$. Given feature set $\mathcal{X} = \mathbb{R}^d$ and output set $\mathcal{Z} = \mathbb{R}^k$, let $\mathcal{F}$ be the set of all multilayer perceptrons whose domain is $\mathcal{X}$ and range is $\mathcal{Z}$. Let the set $\mathbb{A}$ represent all hyperplane arrangements in the domain set $\mathcal{X}$. Define the mapping $\aleph_1 : \mathcal{F} \to \mathbb{A}$. Given a multilayer perceptron $F \in \mathcal{F}$, the mapping $\aleph_1$ erects an arrangement of hyperplanes with one hyperplane h_n for each perceptron $f(\alpha_{1,n})$ in F.

6.2.1.1 Hyperplanes. A hyperplane linearly partitions a d-dimensional space. Hyperplane $h = \{x \in \mathcal{X} : w \cdot x + b = 0\}$ is a translated subspace of dimension $(d-1)$ defined by two fixed parameters: a normal vector $w \in \mathbb{R}^d$ and a bias term $b \in \mathbb{R}$. The normal w is a row vector denoting the slope of the hyperplane; bias b is a scalar denoting the translation of the hyperplane in d-space. [72, 89]

Let h be the set of all real hyperplanes in $\mathbb{R}^d$. Given hyperplane $h \in \mathsf{h}$, there exists a mapping $\Lambda : \mathsf{h} \to \mathbb{R}^{d+1}$ such that $\Lambda(h) = (w, b)$. Parameter vector $(w, b) \in \mathbb{R}^{d+1}$ specifies the normal $w \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$ of hyperplane h, respectively. The mapping $\Lambda : \mathsf{h} \to \mathbb{R}^{d+1}$ is not one-to-one. For a one-to-one mapping, let us restrict the range of Λ. Let $\|\cdot\|$ be the Euclidean norm. A hyperplane can be expressed various ways such as

$$\begin{aligned} h_i &= \{x \in \mathcal{X} : \tfrac{w_i}{\|w_i\|} \cdot x + \tfrac{b_i}{\|w_i\|} = 0, w_i \neq 0\} && \text{so} \quad \Lambda(h_i) = (\tfrac{w_i}{\|w_i\|}, \tfrac{b_i}{\|w_i\|}) \\ &= \{x \in \mathcal{X} : \tfrac{w_i \cdot x}{\|(w_i,b_i)\|} + \tfrac{b_i}{\|(w_i,b_i)\|} = 0, \|(w_i, b_i)\| \neq 0\} && \Lambda(h_i) = (\tfrac{w_i}{\|(w_i,b_i)\|}, \tfrac{b_i}{\|(w_i,b_i)\|}). \end{aligned}$$

Therefore, let us restrict our choice of parameters to the "unit-vector" normal and scaled bias $(\frac{w_i}{\|w_i\|}, \frac{b_i}{\|w_i\|})$. Hence, let

$$\Lambda(h) = (w, b) \text{ such that } \|w\| = 1 \text{ and } b \in \mathbb{R}^1. \tag{20}$$

The range of function Λ is a d-dimensional manifold S, a cylinder in $\mathbb{R}^{d+1}$ defined by

$$S = \left\{ (w, b) \in \mathbb{R}^{d+1} \;\middle|\; \|w\| = 1 \ \wedge \ b \in \mathbb{R}^1 \right\}. \tag{21}$$

In this way, we defined $\Lambda : \mathsf{h} \to S$, a one-to-one mapping from any hyperplane $h \in \mathsf{h}$ to parameter vector $(w, b) \in S$. It is now possible to define the inverse mapping $\Lambda^{-1} : S \to \mathsf{h}$. Given a parameter vector $(w, b) \in S$, then $\Lambda^{-1}[(w, b)]$ returns hyperplane $h = \{x \in \mathcal{X} | \; w \cdot x + b = 0\}$.

6.2.1.2 An arrangement of hyperplanes. An *arrangement of hyperplanes* is a finite set of hyperplanes that collectively partition a d-dimensional space. Given a nonempty arrangement $\mathrm{A} = \{h_1, h_2, \ldots, h_N\}$ in $\mathcal{X}$, define mapping $\Lambda : \mathsf{h}^N \to S^N$ by

$$\begin{aligned} \Lambda(\mathrm{A}) &= \{\Lambda(h_1), \Lambda(h_2), \ldots, \Lambda(h_N)\} \\ &= \{(w_1, b_1), (w_2, b_2), \ldots, (w_N, b_N)\}. \end{aligned}$$

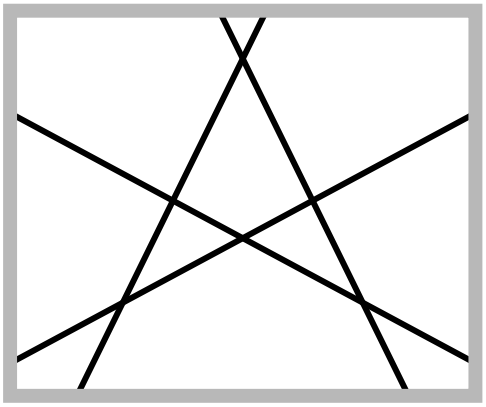
Alternately, we can express the range of Λ in matrix form

$$\Lambda(\mathrm{A}) = (\overline{W} : \overline{b}) \tag{22}$$

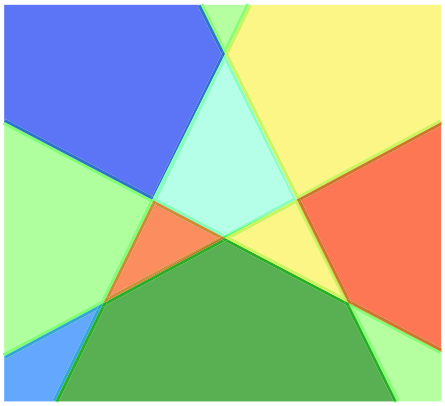
where $\overline{W}$ is a $N \times d$ matrix formed by stacking row vectors w_n and $\overline{b}$ is a $N \times 1$ matrix of bias terms b_n.

$$\overline{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \qquad \overline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}. \tag{23}$$

We designate the augmented matrix $(\overline{W} : \overline{b})$ as the parameter set of the arrangement A. Given a multilayer perceptron, we leverage Equations 22 and 23 to specify the parameters of the hyperplane arrangement implemented by the MLP. In Section 6.2.1, we defined mapping $\aleph_1 : \mathcal{F} \to \mathbb{A}$ to construct an arrangement of hyperplanes $\aleph_1(F) \in \mathbb{A}$ from the first-hidden-layer perceptrons of network $F \in \mathcal{F}$. Each hyperplane $h_n = \{x \in \mathcal{X} : w_n \cdot x + b_n = 0\}$ in the arrangement $\aleph_1(F)$ derives its parameters from the activation of perceptron $f(\alpha_{1,n})$. Given the set of affine activations $\{\alpha_{1,n}(x) = w_{1,n} \cdot x + b_{1,n} : n = 1, 2, \ldots, N\}$, we write the



(a) Arrangement A of hyperplanes



(b) Set of chambers Cham(A)

Figure 37. An arrangement of hyperplanes and the set of chamber formed by the arrangement. A set of 4 two dimensional hyperplanes arranged in general position form a set of $q = 11$ chambers.

parameter set of arrangement $\aleph_1(F)$ in matrix form

$$\Lambda(\aleph_1(F)) = (\overline{W}_1 : \bar{b}_1)$$

where

$$\overline{W}_1 = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ \vdots \\ w_{1,N} \end{bmatrix} \quad \bar{b}_1 = \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,N} \end{bmatrix} . \tag{24}$$

6.2.1.3 On cardinality, subarrangements and chambers. Now, we define a few important terms for hyperplane arrangements. Let $\mathrm{A} = \{h_1, h_2, \ldots, h_N\}$ be a nonempty hyperplane arrangement.

Definition 36 *The number of hyperplanes in arrangement* A *is the* cardinality *of* A*, denoted* card(A).

Definition 37 *If* $\mathrm{B} \subset \mathrm{A}$*, then* B *is called a* subarrangement *of* A *and* $\mathrm{card}(\mathrm{B}) \leq \mathrm{card}(\mathrm{A})$.

Definition 38 *The* intersection *of* A *is given by*

$$a(\mathrm{A}) = \bigcap_{h \in \mathrm{A}} h.$$

The intersection of an arrangement is often the null set when $\text{card}(\text{A}) > d$, the dimensionality of the feature set. However, an arrangement implies a set of nonempty intersections specified by its subarrangements. Given the set of all subarrangements, an intersection $a(\text{B}_i)$ is unique within the arrangement A if $a(\text{B}) \neq a(\text{B}')$ for all $\text{B} \neq \text{B}'$ and $\text{B}, \text{B}' \subset \text{A}$.

The set complement of a hyperplane arrangement forms a collection of disjointed subsets called *chambers*. Each chamber is an open convex polytope. A closed chamber includes the polytope and the convex hull that encloses it. Consider the set of closed chambers. Let $\text{int}(V)$ denote the interior of the set V and $\text{cl}(V)$ denote the set closure of V. For an open set, $V = \text{int}(V)$; for a closed set, $V = \text{cl}(V)$. [22, 53]

Definition 39 *Let* A *be a hyperplane arrangement. The* chamber set Cham(A) *denotes the set of closed chambers formed by* A*. That is,*

$$\text{Cham}(\text{A}) = \{C \subset \mathbb{R}^d : C \neq \text{int}(C), \text{int}(C) \cap h = \emptyset \ \forall h \in \text{A}\}.$$

Given an arrangement of N hyperplanes in d-dimensional space, the maximum number of chambers, q, is realized where

$$q = \sum_{i=0}^{\min\{N,d\}} \binom{N}{i}. \tag{25}$$

To achieve the maximum number of chambers, the hyperplanes of A must be arranged in general position.

Definition 40 *An arrangement* A *is in* general position *if, given any subarrangement* $\text{B} \subset \text{A}$ *where* $\text{card}(\text{B}) \leq d$*,*

1. *The dimension of the intersection of the subarrangement* B *equals* $d - \text{card}(\text{B})$*, and*
2. *The intersection of the subarrangement* B *is unique.*

Figure 37 illustrates an arrangement of 4 hyperplanes partitioning a two-dimensional space. Here, the general position assumption holds true. Therefore, given $d = 2$ and $N = 4$, we can compute from Equation 25 that the number of chambers formed by this arrangement is, in fact, the maximum $q = 11$. [72, 18, 19]

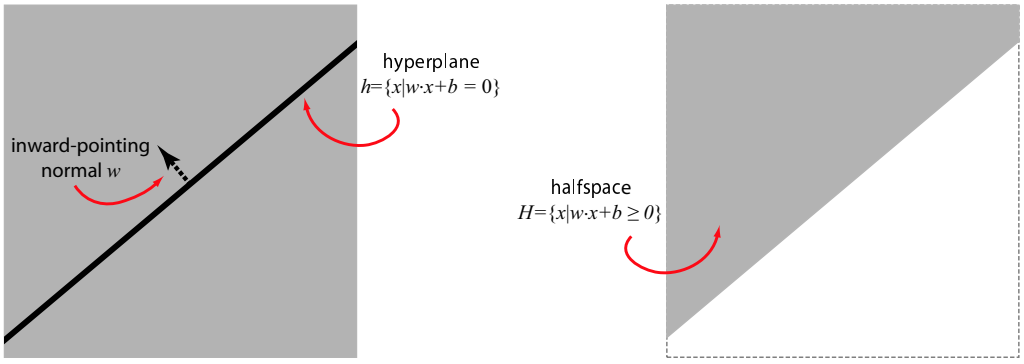


Figure 38. Halfspace H as designated by hyperplane h.

6.2.2 An arrangement of signed hyperplanes. A multilayer perceptron implements a special hyperplane arrangement called an arrangement of signed hyperplanes. The signed hyperplanes $h_+ = \{x \in \mathcal{X} : w \cdot x + b = 0\}$ and $h_- = \{x \in \mathcal{X} : -w \cdot x - b = 0\}$ are not considered the same hyperplane due to the fact $\Lambda(h_+) \neq \Lambda(h_-)$. An arrangement of signed hyperplanes is especially interesting because it implies an *arrangement of halfspaces*—a finite set of halfspaces partitioning the same feature set $\mathcal{X}$—and implements an ordering of chambers.

6.2.2.1 Halfspaces. A halfspace is the set of points to one side of a hyperplane. Given a hyperplane $h = \{x \in \mathcal{X} : w \cdot x + b = 0\}$, it is possible to define two halfspaces: $H = \{x \in \mathcal{X} : w \cdot x + b \geq 0\}$ and $\overline{H^c} = \{x \in \mathcal{X} : w \cdot x + b \leq 0\} = H^c \cup h$ where H^c denotes the set complement of H, and the overbar denotes set closure[7]. Hyperplane h and its normal w suggest the first halfspace, H, as shown in Figure 38, but we have occasion to specify the halfspace $\overline{H^c}$. For such occasions, let us define a directed halfspace $\mathcal{H}$ such that, given a point $x \in \mathcal{X}$ and a halfspace H,

$$\mathcal{H}(x, H) = \begin{cases} H & \text{if } x \in H \\ \overline{H^c} & \text{if } x \notin H. \end{cases} \tag{26}$$

[7]Note, $\overline{H^c} \cup H = \mathcal{X}$ and $\overline{H^c} \cap H = h$.

Let H be the set of all halfspaces in $\mathbb{R}^d$. Given the halfspace $H \in \mathsf{H}$, we wish to define a mapping $\Lambda_* : \mathsf{H} \to S$ much as we defined Λ in Equation 20. Let

$$\Lambda_*(H) = (w, b)$$

where, recall, output parameter vector $(w, b) \in S$ is scaled so that $\|w\| = 1$ and $b \in \mathbb{R}^1$. Mapping $\Lambda_* : \mathsf{H} \to S$ is one-to-one and onto, so the inverse mapping $\Lambda_*^{-1} : S \to \mathsf{H}$ exists. Hence, given a vector $(w, b) \in S$, then $\Lambda_*^{-1}[(w, b)]$ returns a unique halfspace $H = \{x \in \mathcal{X} : w \cdot x + b \geq 0\}$.

6.2.2.2 An arrangement of halfspaces. An arrangement of signed hyperplanes implies a unique arrangement of halfspaces. Given a nonempty feature set $\mathcal{X}$ and a finite halfspace arrangement $\mathcal{A} = \{H_1, H_2, ..., H_N\}$ in $\mathcal{X}$, define mapping $\Lambda_* : \mathsf{H}^N \to S^N$

$$\Lambda_*(\mathcal{A}) = (\overline{W} : \overline{b}),$$

similar to Equation 22. For every arrangement of signed hyperplanes $\mathrm{A} \in \mathsf{h}^N$, there is a unique halfspace arrangement $\mathcal{A} \in \mathsf{H}^N$ such that $\Lambda_*(\mathcal{A}) = \Lambda(\mathrm{A})$. Let inverse mapping $\Lambda_*^{-1}[(\overline{W} : \overline{b})]$ return a set of halfspaces so that, given the signed hyperplane arrangement A, a unique arrangement of halfspaces can be expressed as

$$\mathcal{A} = \Lambda_*^{-1}[\Lambda(\mathrm{A})]. \tag{27}$$

Subarrangement and intersection operations readily apply to halfspace arrangements. Let A be an arrangement of N signed hyperplanes $\{h_1, h_2, \ldots, h_N\}$ and $\mathcal{A}$ be the unique arrangement of halfspaces $\{H_1, H_2, \ldots, H_N\}$ where $\Lambda_*(\mathcal{A}) = \Lambda(\mathrm{A})$. The cardinality of the halfspace arrangement is equal to the number of halfspaces in the arrangement, $\mathrm{card}(\mathcal{A}) = N$, and equal to the cardinality of hyperplane arrangement. A subarrangement of halfspaces behaves like an subarrangement of hyperplanes—for example, if $\mathcal{B} \subset \mathcal{A}$ then $\mathrm{card}(\mathcal{B}) \leq$

card($\mathcal{A}$). The intersection of a halfspace arrangement $\mathcal{A}$ is given by

$$a(\mathcal{A}) = \bigcap_{H \in \mathcal{A}} H. \tag{28}$$

Definition 41 *If an intersection of halfspaces, $a(\mathcal{A}) \neq \emptyset$, is non-empty, the intersection forms a single chamber (closed set). This chamber is called the* central chamber *of chamber set* Cham(A) *where hyperplane arrangement* $\mathrm{A} = \Lambda^{-1}[\Lambda_*(\mathcal{A})]$.

Given an arrangement A of signed hyperplanes and a point $x \in \mathcal{X}$, we can define the chamber that contains x by combining Equation 28 with Equations 26 and 27 above.

Theorem 5 *Given the hyperplane arrangement A and feature $x \notin \cup h, h \in A$, let $C(x)$ be the closed chamber in* Cham(A) *that contains x. Then, chamber $C(x)$ is the intersection of directed halfspaces derived from hyperplane arrangement* A *and the point x.*

A set of directed halfspaces $\{\mathcal{H}(x, H_1), \mathcal{H}(x, H_2), \ldots, \mathcal{H}(x, H_N))\}$ always forms a non-empty intersection since, by definition, they all contain point x.

$$C(x) = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \mathcal{H}(x, H). \tag{29}$$

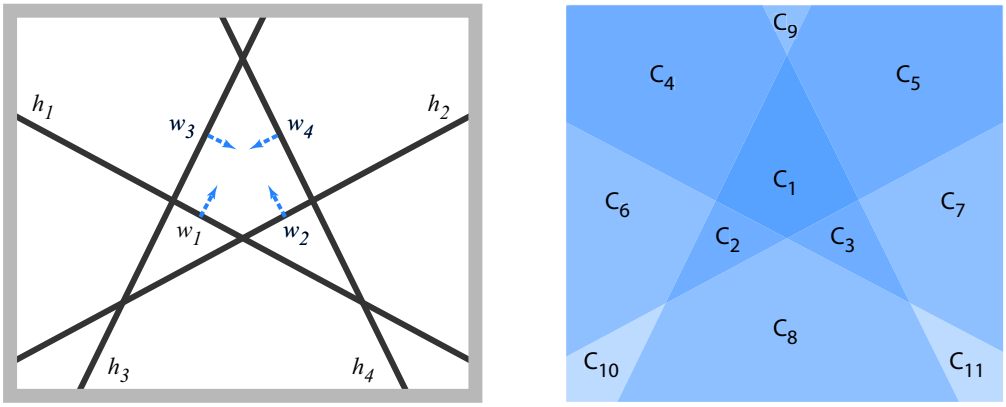
6.2.2.3 The partial ordered set of chambers. An arrangement of signed hyperplanes represents a partial ordering of chambers.

Definition 42 *Let* A *be an arrangement of signed hyperplanes. Given a chamber $C \in$* Cham(A)*, define the mapping $\rho :$ Cham(A) $\rightarrow \mathbb{R}$ so that*

$$\rho(C, \mathrm{A}) = \mathrm{card}(\{H \in \mathcal{A} : C \subset H\}). \tag{30}$$

where $\mathcal{A} = \Lambda_*^{-1}[\Lambda(\mathrm{A})]$. Thus, $\rho(C, \mathrm{A})$ is the number of halfspaces in arrangement $\mathcal{A}$ that contain the chamber C.

To define a relation $\preccurlyeq$ on Cham(A) , we say chamber $C \in$ Cham(A) "precedes" chamber $B \in$ Cham(A) and write $C \preccurlyeq B$ if and only if $\rho(C, \mathrm{A}) \leq \rho(B, \mathrm{A})$. From this definition, we form the theorem below.



(a) Signed arrangement (b) Ordered set of chambers

Figure 39. An ordered set of chambers as designated by a signed arrangement of 4 hyperplanes. The ordered set is $(\{C_{10}, C_{11}\}, \{C_6, C_7, C_8, C_9\}, \{C_2, C_3, C_4, C_5\}, C_1)$.

Theorem 6 *Let* A *be a hyperplane arrangement, then* ρ *defines an ordering* $\preccurlyeq$ *on chamber set* Cham(A)*. Thus,* $(\text{Cham(A)}, \preccurlyeq)$ *is an ordered set.*

Proof of Theorem 6. The partially ordered set $(\mathbb{R}, \leq)$ is well-defined. Clearly, $\preccurlyeq$ is reflexive over Cham(A). Also, transitivity is clear. Antisymmetry does not hold true, since if $\rho(C, \text{A}) = \rho(B, \text{A})$ then there is no reason to conclude that $C = B$. Hence, $\preccurlyeq$ defines an ordering on Cham(A), though not a partial ordering. ■

In order to make a partially ordered set, we create equivalence classes of chambers. Define the collection of equivalence classes for Cham(A) to be Class(Cham(A)). Define the class $[C]_\rho = \{C' \in \text{Cham(A)} : \rho(C', \text{A}) = \rho(C, \text{A})\}$. We define the relation $\preccurlyeq$ on Class(Cham(A)) to be $[C] \preccurlyeq [B]$ if and only if $\rho(C', \text{A}) \leq \rho(B', \text{A})$ for all $C' \in [C]$ and $B' \in [B]$. Now, reflexivity, transitivity, and antisymmetry hold true, and $(\text{Class(Cham(A))}, \preccurlyeq)$ is a partially ordered set. Thus, the following theorem holds true.

Theorem 7 *Let* A *be a hyperplane arrangement, then* ρ *defines an ordering* $\preccurlyeq$ *on the collection of equivalence classes for* Cham(A)*. Thus,* $(\text{Class(Cham(A))}, \preccurlyeq)$ *is a partially ordered set.*

Figure 39(a) illustrates an arrangement of four hyperplanes $\text{A} = \{h_1, h_2, h_3, h_4\}$ and their respective set of normals $\{w_1, w_2, w_3, w_4\}$. Figure 39(b) represents the ordering of Cham(A) with respect to ρ as a function of grayscale (from light to dark). Note $C_2 \preccurlyeq C_1$ is

true as $\rho(C_1, \mathrm{A}) = 4$, $\rho(C_2, \mathrm{A}) = 3$ and, thus, $\rho(C_2, \mathrm{A}) < \rho(C_1, \mathrm{A})$. Also, $\rho(C_{10}) = \rho(C_{11})$, therefore $C_{11} \in [C_{10}]_\rho$. In fact, $[C_{10}]_\rho = \{C_{10}, C_{11}\}$. The ordering of the chamber set with respect to ρ summarizes as $\rho(C_i, \mathrm{A}) < \rho(C_j, \mathrm{A}) < \rho(C_k, \mathrm{A}) < \rho(C_1, \mathrm{A})$ where $i \in \{10, 11\}$, $j \in \{6, 7, 8, 9\}$, $k \in \{2, 3, 4, 5\}$. Given the definition of equivalence classes above, the set $(\mathrm{Class}(\mathrm{Cham}(\mathrm{A})), \preccurlyeq) = \{\{C_{10}, C_{11}\}, \{C_6, C_7, C_8, C_9\}, \{C_2, C_3, C_4, C_5\}, \{C_1\}\}$ is partially ordered. In this example, chamber set Cham(A) has a central chamber, C_1. If a central chamber exists, it supersedes all other chambers—that is, it ranks last and exclusively last in the ordered set as chamber C_1 does in $(\mathrm{Class}(\mathrm{Cham}(\mathrm{A})), \preccurlyeq)$ above.

By changing "signs" on an arrangement's parameter vectors, additional orderings of a chamber set may be explored. Let P be an $N \times N$ diagonal matrix whose diagonal elements are restricted to -1 or 1. Given a hyperplane arrangement A with parameter set $\Lambda(\mathrm{A}) = (\overline{W} : \bar{b}) \in S$ and chamber set Cham(A) , let $P\Lambda(\mathrm{A}) = (P\overline{W} : P\bar{b})$ so that $P\Lambda(\mathrm{A}) \subset S$. Thus, we can specify a new arrangement $\mathrm{A}_P = \Lambda_*^{-1}[P\Lambda(\mathrm{A})]$ with the same chamber set, $\mathrm{Cham}(\mathrm{A}) = \mathrm{Cham}(\mathrm{A}_P)$, but a different ordering on that set, $(\mathrm{Cham}(\mathrm{A}), \preccurlyeq) \neq (\mathrm{Cham}(\mathrm{A}_P), \preccurlyeq)$.

6.2.3 Summary. In this section, we derived an ordering of a feature set for use in the application of the ordered veracity-experience response curve, or OVER curve. The ordering of the feature set is derived from an ordering of the chamber set formed by the hyperplane arrangement implemented in an MLP's first hidden layer. The chamber set allows us to order an infinitely large feature set using a finite number of elements. From this ordering, we can derive the first part of the OVER curve which is an ordering of a domain based on a classifier's veracity over the features of that domain. We will demonstrate this capability in the next section. Also, we shall demonstrate the derivation of the second part of the OVER curve which requires us to cross-reference a classifier's apparent experience with a feature to the veracity assigned to that feature. We shall derive and demonstrate algorithms that allow us to investigate the memberships of the ordered chambers in relation to the MLP's training set and, from these investigations, to assign a quantification of apparent experience to each chamber in the domain. Finally, we shall demonstrate OVER curves for several MLPs used to implement a classic XOR dataset.

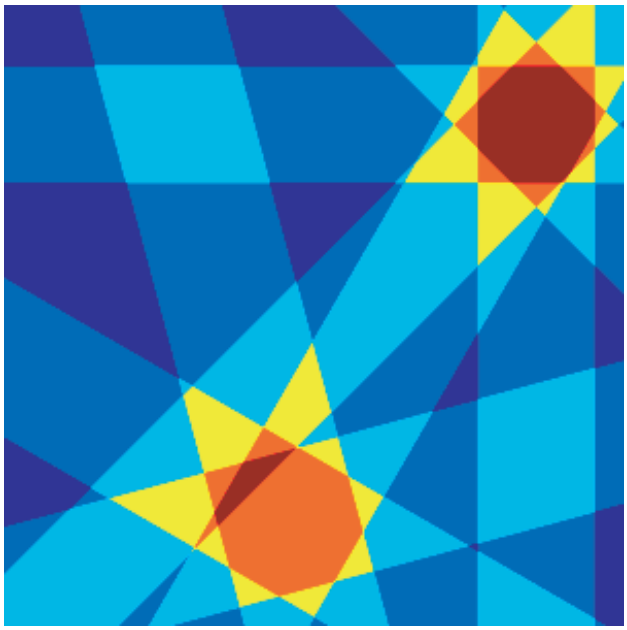


Figure 40. An arrangement A of 16 hyperplanes forms a set of chambers Cham(A).

6.3 The unconstrained multilayer perceptron

When designing the multilayer perceptron, today's pattern recognition community relies on training tools that ignore the partial ordering of chambers. Resulting information models are biased toward memorization and littered with false positives as they arbitrarily order unpopulated chambers, i.e., chambers that do not contain training data. When governed by training tools such as mean-squared error measures, backpropagation and cross validation, the multilayer perceptron is ill suited for generalization tasks as design solutions prove to be poor interpolators and unregulated extrapolators.

6.3.1 Ordered chambers. To explore the ordered set $(\text{Cham(A)}, \preccurlyeq)$, we begin by resolving all populated chambers. Define a populated chamber as a closed chamber that contains training data. Given a hyperplane arrangement $\text{A} \in \mathsf{h}$ and a dataset $\mathfrak{D} \subset \mathcal{X}$, let $Q_{\mathfrak{D}} \subset \text{Cham(A)}$ be the set of populated chamber—that is,

$$Q_{\mathfrak{D}} = \{C \in \text{Cham(A)} : x \in C \text{ for some } x \in \mathfrak{D}\}.$$

Assume we have trained a multilayer perceptron to partition a signed, truthed dataset $\mathfrak{D} = \mathfrak{D}^+ \cup \mathfrak{D}^-$. Given the set $\mathcal{F}$ of all MLPs whose range is $\mathbb{R}$, we say a classifier $F \in \mathcal{F}$ implements $\mathfrak{D}$ if

$$F(x) = \begin{cases} +1 & \text{if } x \in \mathfrak{D}^+ \\ -1 & \text{if } x \in \mathfrak{D}^-. \end{cases} \tag{31}$$

Let $\mathcal{F}_{\mathfrak{D}} = \{F \in \mathcal{F} : F \text{ implements } \mathfrak{D}\}$ be the set of all MLPs that implement the dataset $\mathfrak{D}$. Given a multilayer perceptron $F \in \mathcal{F}_{\mathfrak{D}}$, there is a hyperplane arrangement $\mathrm{A} = \aleph_1(F)$ whose collection of populated chambers $Q_{\mathfrak{D}}$ we now define.

6.3.1.1 Defining a populated chamber. From Section 6.2.2.1 we know, given an arrangement of hyperplanes and any point $x \in \mathcal{X}$, we can define a closed chamber $C(x)$ per Equation 29. Given a point $x \in \mathfrak{D}$ and a hyperplane arrangement $\mathrm{A} = \aleph_1(F)$, we can express a populated chamber as the intersection of directed halfspaces

$$C(x) = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\aleph_1(F))]} \mathcal{H}(x, H).$$

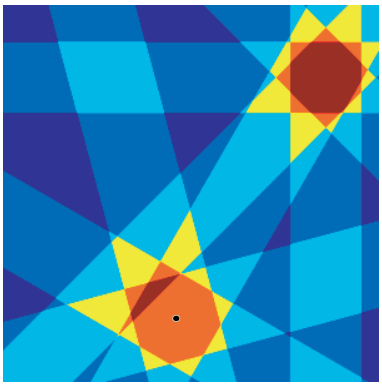
Alternatively, we can define chamber $C(x)$ in terms of an N-tuple set of labels $\overrightarrow{\phi} = (\phi_1, \ldots, \phi_N) \in \mathcal{L}^N$ where the label set $\mathcal{L} = \{-1, 1\}$. Given some x' in $C(x) \in \mathrm{Cham}(\aleph_1(F))$ and a halfspace $H \in \Lambda_*^{-1}[\Lambda(\aleph_1(F))]$, define the label mapping $\phi(x', H)$

$$\phi(x', H) = \begin{cases} 1 & \text{if } x' \in H \\ -1 & \text{if } x' \notin H. \end{cases} \tag{32}$$

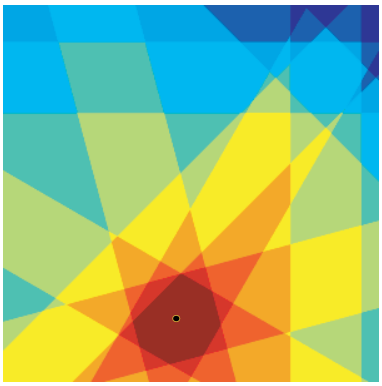
Label $\phi(x', H)$ represents the condition for the directed halfspace $\mathcal{H}(x', H)$ given a point $x' \in C(x)$.

Given the arrangement of halfspaces $\mathcal{A} = \{H_1, H_2, \ldots, H_N\}$ and a point $x \in \mathbb{R}^d$, define the mapping for a label vector as $\overrightarrow{\phi}(x, \mathcal{A})$ to be

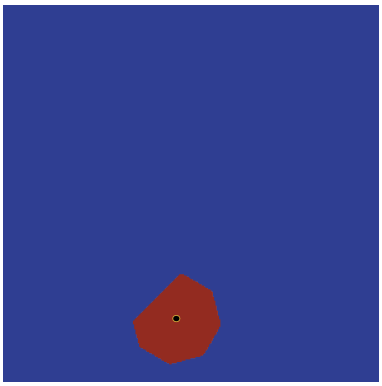
$$\overrightarrow{\phi}(x, \mathcal{A}) = (\phi(x, H_1), \phi(x, H_2), \ldots, \phi(x, H_N)) \tag{33}$$



(a) Select a point x.



(b) Arrangement A' of directed halfspaces $\mathcal{H}(x, H_n)$



(c) Chamber $C(x)$

Figure 41. A point x is selected within one of the chambers. The set of halfspaces are directed toward the point; their intersect specifies the chamber containing x.

such that $\overrightarrow{\phi} \in \{-1, 1\}^N$. We call label vector $\overrightarrow{\phi}(x, \Lambda_*^{-1}[\Lambda(\mathrm{A})])$ the *signed diagonal* of chamber $C(x) \subset \mathrm{Cham(A)}$ as the vector $\overrightarrow{\phi}$ constructs a scaling matrix to transform parameter set $\Lambda(\mathrm{A}) = [\overline{W} : \bar{b}]$ per Equation 22. Let the signed diagonal $\overrightarrow{\phi} \in \{-1, 1\}^N$ form the diagonal of an $N \times N$ scaling matrix $P = \mathrm{diag}(\overrightarrow{\phi})$. Given $P\Lambda(\mathrm{A}) = [P\overline{W} : P\bar{b}]$, the signed diagonal $\overrightarrow{\phi}$ corresponds to chamber $C(x)$ if

$$C(x) = \bigcap_{H \in \Lambda_*^{-1}[P\overline{W}:P\bar{b}]} H.$$

Recall from Section 6.2.2.1, a central chamber is the non-empty intersection of a set of halfspaces. Therefore, chamber $C(x)$ is the central chamber of arrangement $\Lambda^{-1}[P\overline{W} : P\bar{b}]$, and is formed by the intersection of halfspace arrangement $\Lambda_*^{-1}[P\overline{W} : P\bar{b}]$.

As an example of calculating the signed diagonal $\overrightarrow{\phi}$ of a chamber, consider Figures 40 and 41. Figure 40 illustrates an arrangement A of 16 signed hyperplanes forming two octagons. The coloring assigned to each chamber in this figure indicates its ordering in $(\mathrm{Cham(A)}, \preccurlyeq)$. We can resolve a chamber within the chamber set Cham(A) by selecting an arbitrary point x the centroid to the larger octagon as shown in Figure 41(a). Let the parameter set of arrangement A be the augmented matrix $[\overline{W} : \bar{b}] = \Lambda(\mathrm{A})$. To determine the signed diagonal $\overrightarrow{\phi}$ for chamber $C(x)$, we evaluate the product $[\overline{W} : \bar{b}] \cdot [x; 1]$ such that, given any $x' \in C(x)$,

$$\overrightarrow{\phi} = \overrightarrow{\phi}(x', \mathcal{A}) = ([\overline{W} : \bar{b}] \cdot [x'; 1] \geq 0) - ([\overline{W} : \bar{b}] \cdot [x'; 1] < 0).$$

In this expression, each element of the signed diagonal is derived as

$$\phi_i = \phi(x', H_i) = \begin{cases} 1 & \text{if } w_i \cdot x' + b_i \geq 0 \\ -1 & \text{if } w_i \cdot x' + b_i < 0 \end{cases}$$

where halfspace $H_i = \{x \in \mathcal{X} : w_i \cdot x + b_i \geq 0\} \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]$. Once we have the signed diagonal $\overrightarrow{\phi}(x', \mathrm{A})$, we can define the arrangement A' of directed hyperplanes in which chamber $C(x)$ is the central chamber: Let scaling matrix $P = \mathrm{diag}(\overrightarrow{\phi})$, then we can express arrangement $\mathrm{A}' = \Lambda^{-1}[P\overline{W} : P\bar{b}]$ as shown in Figure 41(b). Finally, from the

```
% populatedQ.m
% define a set of populated chambers
 function Phi_Q=populatedQ(dataSet,W,b)
% Get data set
 store dataSet: matrix of data points
 store ones: row vector of ones
% Get parameter set for arrangement
 store W: matrix of inward normals
 store b: column vector of biases
% Find populated chambers
 D=[dataSet; ones];
 chamberLabels=([W b]*D>=0)-([W b]*D<0);
% Eliminate redundant label vectors
 Phi_Q = [];
 phiLeft = chamberLabels;
 while ~isempty(phiLeft)
  phi=phiLeft(:,1);
  Phi_Q=[Phi_Q phi];
  rho=phi'*phiLeft;
  rhoMax=phi'*phi;
  indexMembersLeft = find(rho~=rhoMax);
  phiLeft=phiLeft(:,indexMembersLeft);
 end %while
% Return label vectors specifying Q,
%    the set of populated chambers
 return Phi_Q: set of signed diagonals
% end populatedQ.m
```

Figure 42. Pseudo code describing Algorithm *populatedQ*.

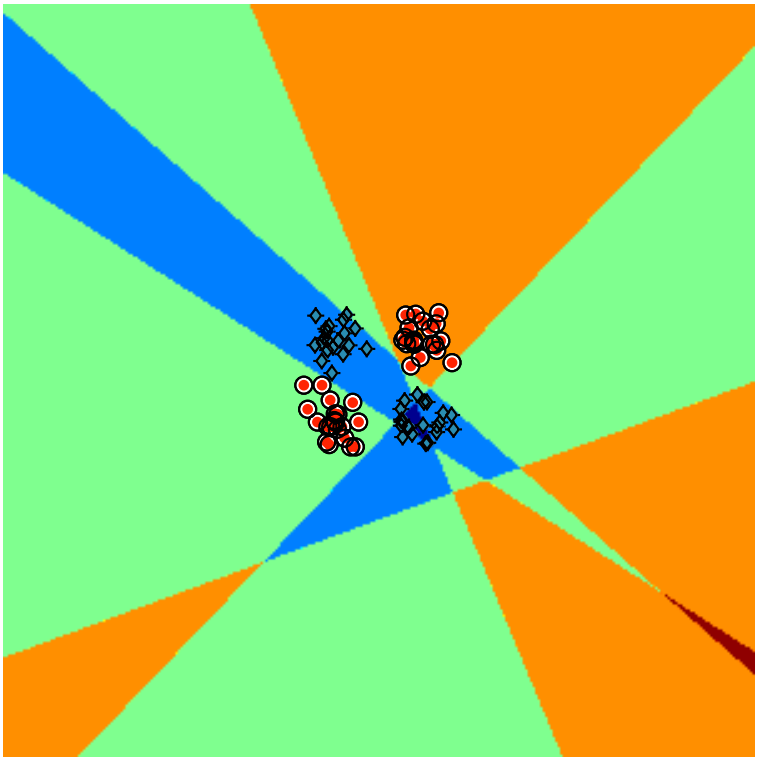
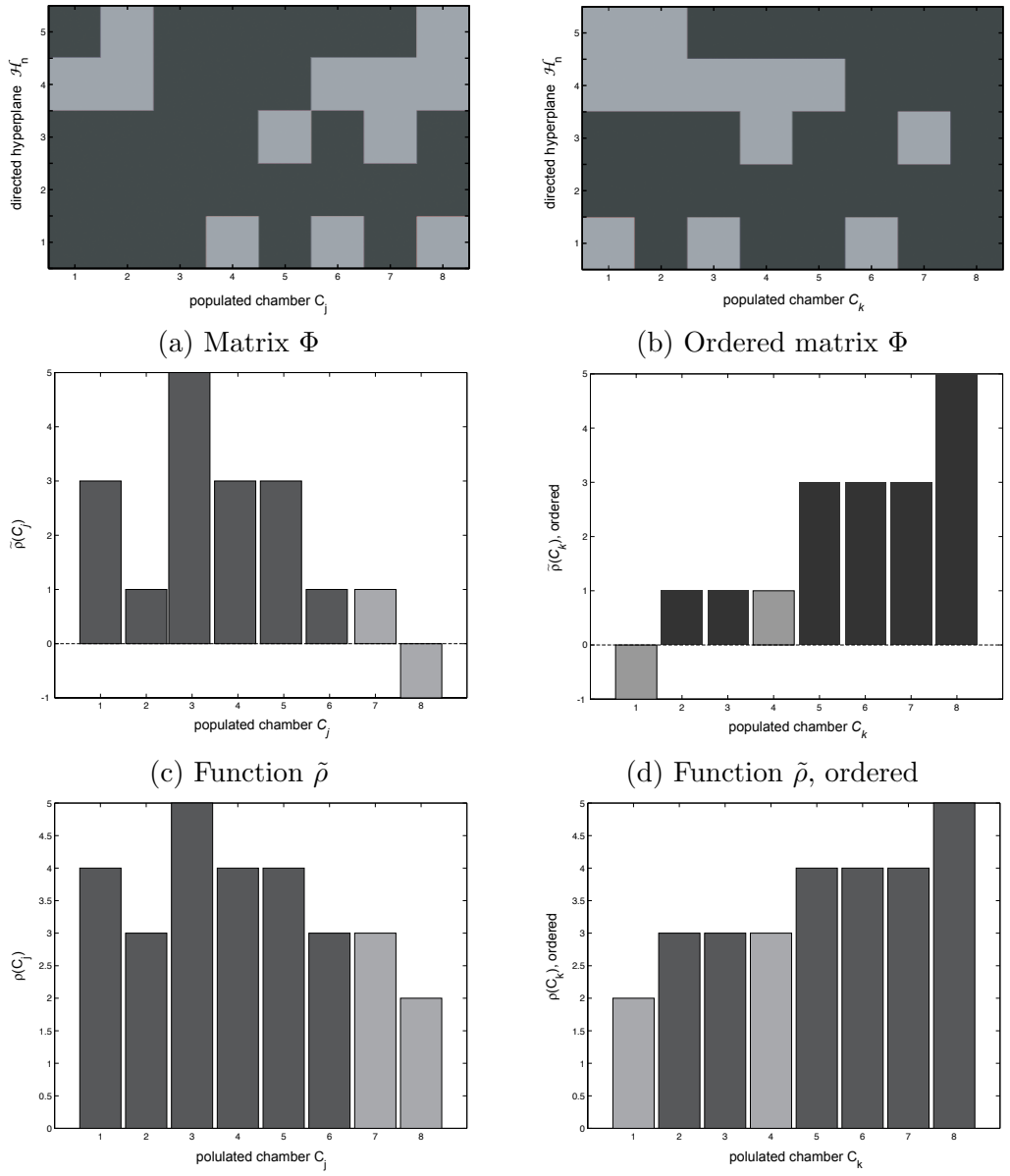


Figure 43. A multilayer perceptron is trained by back propagation to implement a solution for the XOR problem.

directed halfspace arrangement $\mathcal{A}' = \Lambda_*^{-1}[\Lambda(\mathrm{A}')] = \Lambda_*^{-1}[P\overline{W} : P\bar{b}]$, we specify chamber $C(x)$—depicted in Figure 41(c)—as the intersection $C(x) = a(\mathcal{A}') = \bigcap_{H \in \mathcal{A}'} H$.

Let us define a matrix Φ of signed diagonals where each column defines the signed diagonal of a unique populated chamber in Cham(A). Given a dataset $\mathfrak{D} \subset \mathcal{X}$ and a hyperplane arrangement A, matrix elements are $\phi_{i,j} = \phi(x_j, H_i)$ and matrix columns are $\overrightarrow{\phi}_j = \overrightarrow{\phi}(x_j, \Lambda_*^{-1}[\Lambda(\mathrm{A})])$ for each $x_j \in \mathfrak{D}$. When we create Φ using every point in $\mathfrak{D}$, the matrix reflects the cardinality of the dataset and not necessarily $Q_{\mathfrak{D}}$. Each column in Φ represents a chamber and, since multiple points may populate a chamber, it is necessary to eliminate redundant columns. The Algorithm *populatedQ*[8] describes one technique

[8]See Figure 42 for the pseudo code that summarizes a technique for reducing redundant columns. This technique relies on the property that a central chamber (as specified by the directed arrangement $\mathrm{A} = \Lambda^{-1}[(\mathbf{P}\overline{W} : \mathbf{P}\bar{b})]$ will uniquely supersede all other chambers in the ordering $(\mathrm{Cham}(\mathrm{A}), \preccurlyeq)$ described in Equation 30.



(a) Matrix Φ (b) Ordered matrix Φ

(c) Function $\tilde{\rho}$ (d) Function $\tilde{\rho}$, ordered

(e) Set function ρ (f) Set function ρ, ordered

Figure 44. Ordering $Q_{\mathfrak{D}}$, the set of populated chambers, via (a) Φ, the matrix of signed diagonals. The columns of matrix Φ are summed to create bar charts (c) $\tilde{\rho}$ and (e) $\rho = \frac{N+\tilde{\rho}}{2}$. (b) Set $(Q_{\mathfrak{D}}, \preccurlyeq)$ is ordered based on (f) the sort of set function ρ or, equivalently, (d) the sort of function $\tilde{\rho}$.

for reducing redundant columns. Once redundant columns $\overrightarrow{\phi}_j$ have been eliminated, the cardinality of $Q_{\mathfrak{D}}$ is equal to the number of remaining columns in Φ.

```
% orderQ.m
%   order set of chambers
 function posetPhi_Q=orderQ(Phi_Q)
% Get signed diagonals specifying Q,
%     the set of populated chambers
 store Phi_Q: set of signed diagonals
% Derive partial ordering of set Q
 rho_tilda=sum(Phi_Q);
 [dum,order]=sort(rho_tilda);
 posetPhi_Q=Phi_Q(:,order);
% Return ordered set of signed diagonals
%     specifying poset {Q,<=}
 return posetPhi_Q: ordered chamber set
% end orderQ.m
```

Figure 45. Pseudo code describing Algorithm *orderQ*.

Given we have the set of signed diagonals designating the populated chambers of an arrangement, we can order these chambers by summing each column vector in Φ and then sorting the sums. The sum of a signed diagonal is linearly proportional to set function ρ defined in Equation 30. Given an arrangement A of N hyperplanes and point $x \in \mathcal{X}$, we can rank the chamber $C(x)$ via

$$\begin{aligned} \rho(C(x), \mathrm{A}) &= \sum_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \frac{\phi(x, H) + 1}{2} \\ &= \frac{N}{2} + \frac{1}{2} \sum_{i=1}^{N} \phi(x, H_i) \end{aligned} \tag{34}$$

or

$$\tilde{\rho}(C(x), \mathrm{A}) = \sum_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \phi(x, H_i). \tag{35}$$

Given the transformation from $\tilde{\rho}$ to ρ is a positive scaling and a translation, $\tilde{\rho} = 2\rho - N$ can be used to rank $C(x)$ in the ordered set $(Q_{\mathfrak{D}}, \preccurlyeq)$.

Let matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$ be the set of non-redundant signed diagonals representing the populated chambers of $Q_{\mathfrak{D}}$, and let matrix $\Phi((Q_{\mathfrak{D}}, \preccurlyeq), \mathrm{A})$ be the set of signed diagonals for the ordered set $(Q_{\mathfrak{D}}, \preccurlyeq)$. For a demonstration of deriving $\Phi((Q_{\mathfrak{D}}, \preccurlyeq), \mathrm{A})$ from $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$, let us consider the dataset $\mathfrak{D}$ and the arrangement A of $N = 5$ hyperplanes depicted in Figure 43. In Figure 43, we depict Class True points as diamonds and Class False points as circles. Using Algorithm *populatedQ*, the Class True points were tested resulting in the discovery of 6 unique populated chambers. Next, a test of the Class False points resulted in the discover of two more populated chambers. Thus, we found a total of eight populated chambers and specified the signed diagonal of each in a column of matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$ below.

$$\Phi(Q_{\mathfrak{D}}, \mathrm{A}) = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

Figure 44(a) also portrays matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$ where, again, we ranked the signed diagonals in the order the test points $x \in \mathfrak{D}$ were presented to Algorithm *populatedQ*.

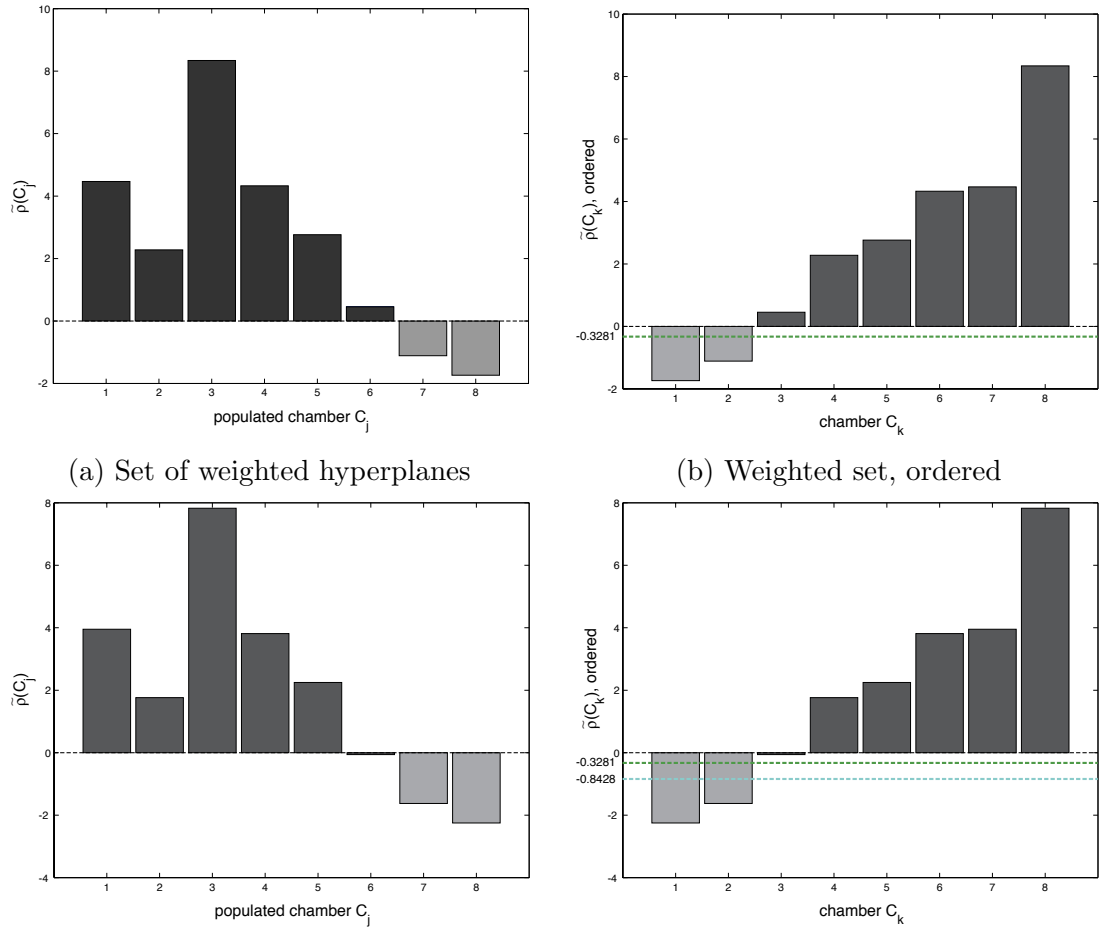
To order of the chambers of $Q_{\mathfrak{D}}$, we sum the columns of matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$ per Algorithm *orderQ*[9] such that

$$\tilde{\rho}(Q_{\mathfrak{D}}, \mathrm{A}) = [\ 3\ \ 1\ \ 5\ \ 3\ \ 3\ \ 1\ \ 1\ \ -1\ ],$$
$$\rho(Q_{\mathfrak{D}}, \mathrm{A}) = [\ 4\ \ 3\ \ 5\ \ 4\ \ 4\ \ 3\ \ 3\ \ 2\ ].$$

The bar charts of Figure 44(c) and Figure 44(e) depict $\tilde{\rho}$ and $\rho = \frac{N+\tilde{\rho}}{2}$, respectively. Finally, $(Q_{\mathfrak{D}}, \preccurlyeq)$ is realized by ranking the chamber set based on Figure 44(f) the sort of set function ρ or, equivalently, Figure 44(d) the sort of function $\tilde{\rho}$. Figure 44(b) portrays the result of this ranking where

$$\Phi((Q_{\mathfrak{D}}, \preccurlyeq), \mathrm{A}) = \begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 \end{bmatrix}.$$

[9] See Figure 45.



(a) Set of weighted hyperplanes

(b) Weighted set, ordered

(c) Pruned set of hyperplanes

(d) Pruned set, ordered

Figure 46. Reordering populated chambers (a)-(b) using a set of weighted, signed hyperplanes and (c)-(d) using a pruned set of weighted, signed hyperplanes.

With matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$, we can evaluate how adding or removing perceptrons from an MLP affects the ordering of chambers. To prune a perceptron is to prune one hyperplane, so we simply remove the hyperplane's row from matrix $\Phi(Q_{\mathfrak{D}}, \mathrm{A})$ before calculating $\tilde{\rho}$. For instance, we can remove the fourth hyperplane in arrangement A so that our pruned set of

signed hyperplanes looks like

$$\Phi(Q_{\mathfrak{D}}, \mathrm{A} - h_4) = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

and $\rho(Q_{\mathfrak{D}}, \mathrm{A} - h_4) = [\ 3\ \ 2\ \ 4\ \ 3\ \ 3\ \ 2\ \ 2\ \ 1\ ] = \rho(Q_{\mathfrak{D}}, \mathrm{A}) - 1$. Note, removing the fourth hyperplane does not affect the cardinality of $Q_{\mathfrak{D}}$ or the order of $(Q_{\mathfrak{D}}, \preccurlyeq)$. However, if we were to prune the first hyperplane h_1,

$$\Phi(Q_{\mathfrak{D}}, \mathrm{A} - h_1) = \begin{bmatrix} -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

the number of non-redundant populated chambers decreases to 6.

$$\Phi(Q'_{\mathfrak{D}}, \mathrm{A} - h_1) = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

and $\rho(Q'_{\mathfrak{D}}, \mathrm{A} - h_4) = [\ 3\ \ 4\ \ 3\ \ 3\ \ 2\ \ 2\ ]$.

We can also evaluate the chamber set ordering implemented by the weighted outputs of an MLP's hidden perceptrons. Here, we transform Φ via the scaling matrix P, then sum the columns of matrix $P\Phi$. In this application, the diagonal elements of P may be any real number. The transformed parameter set $P\Phi$ is not restricted to the manifold S, but such a restriction is unnecessary to order the chamber set. Figure 46 illustrates an ordering using the weighted perceptrons implemented in Figure 31(a). From this illustration, we can choose an appropriate bias (or threshold) for the complete set of weighted perceptrons and for the pruned set.

```
% psiChamber.m
% find minimum arrangement for chamber
 function psi_C=psiChamber(W,b,phi_C)
% Get parameter set for arrangement
%  of N hyperplanes
 store W: matrix of inward normals
 store b: column vector of biases
 store N: number of hyperplanes
 store d: dimension of space
% Get specification of chamber C
 store phi_C: signed diagonal of C
% Eliminate redundant hyperplanes
 psi_C=phi_C;
 P=diag(phi_C);
 V=P*[W b];
 for k=1:N
   kCompliment=find([1:N]~=k);
   g=V(k,1:d);
   t=V(k,d+1);
   U=V(kCompliment,1:d);
   c=V(kCompliment,d+1);
   xMax=linearProgram(-g,U,c);
   isHyperplane=g*xMax-t >= 0;
   psi_C(k)=isHyperplane*psi_C(k);
 end % for
% Return label vector specifying minimum
%  arrangement with central chamber C
 return psi_C: signed aliasing diagonal
% end psiChamber.m
```

Figure 47. Pseudo code describing Algorithm *psiChamber*.

6.3.1.2 Defining unpopulated chambers via Linear Programming. To order an entire chamber set, we resolve adjacent chambers and work our way out. Given the signed diagonal $\overrightarrow{\phi}_C$ of a populated chamber C, the signed diagonal of an adjacent chamber C_{adj} is exactly the same except for the sign of one of the vector's elements. That element corresponds to the only directed halfspace that must "flip" differently in Equation 29 when evaluating points in chamber C versus points in chamber C_{adj}. Given a nonempty arrangement A of hyperplanes, every chamber $C \in \text{Cham(A)}$ has a nonempty set of adjacent chambers $[C]_{adj} \subset \text{Cham(A)}$. The cardinality of $[C]_{adj}$ is equal to the number of hyperplanes that bound C.

We say a hyperplane h bounds a chamber if $h \cap C \neq \emptyset$, i.e. some subset of points on the hyperplane h are members of the closed chamber C. Given an arrangement A of hyperplanes and chamber $C \subset \text{Cham(A)}$, there exists a subarrangement B $\subset$ A that includes all the hyperplanes $h \in$ A where $h \cap C \neq \emptyset$ and none of the hyperplanes $h \in$ A where $h \cap C = \emptyset$. Specifying the subarrangement B suggests another label besides "-1" and "1"—the label "0" denoting the condition $h \cap C = \emptyset$. Given a halfspace $H \in \Lambda_*^{-1}[\Lambda(\text{A})]$ and a chamber $C \in \text{Cham(A)}$, define the mapping $\psi(C, H) \in \{-1, 0, 1\}$ such that

$$\psi(C,H) = \begin{cases} 1 & \text{if } (H \cap \overline{H^c}) \cap C \neq \emptyset \text{ and } C \subset H \\ 0 & \text{if } (H \cap \overline{H^c}) \cap C = \emptyset \\ -1 & \text{if } (H \cap \overline{H^c}) \cap C \neq \emptyset \text{ and } C \not\subset H \end{cases} \tag{36}$$

where $h = H \cap \overline{H^c}$. Given arrangement of halfspaces $\mathcal{A} = \{H_1, H_2, \ldots, H_N\}$, define the label vector mapping $\overrightarrow{\psi}(C, \mathcal{A}) = (\psi(C, H_1), \psi(C, H_2), \ldots, \psi(C, H_N)) \in \{-1, 0, 1\}^N$. We call $\overrightarrow{\psi}$ the *signed aliasing diagonal* as it is used to construct an aliasing matrix M. An aliasing matrix M is a matrix formed by removing every row from a $N \times N$ identity matrix that corresponds to a $\psi(C, H) = 0$ label in the vector $\overrightarrow{\psi}$. Let M_n denote an identity matrix with the n^{th} row removed. This aliasing matrix removes one non-bounding hyperplane if

$$C(x) = \bigcap_{H \in \Lambda_*^{-1}[M_n \overline{W} : M_n \overline{b}]} \mathcal{H}(x, H).$$

Definition 43 *(Minimum Arrangement). Given the signed hyperplane arrangement* A *and a chamber* $C \in \mathrm{Cham(A)}$, $\mathrm{B} \subset \mathrm{A}$ *is the minimal subarrangement with respect to* C *if*

1. $C \in \mathrm{Cham(B)}$,
2. $C \notin \mathrm{Cham(B')}$ *for all* $\mathrm{B'} \subset \mathrm{B}$ *with* $\mathrm{B'} \neq \mathrm{B}$.

The chamber C is a convex polytope and, accordingly, we use Linear Programming (LP) to resolve the minimum arrangement that defines C [6, 34]. Linear programming is a problem formation in parametric optimization. The form is used to find a parameter vector defined as optimal with respect to a linear objective function. The objective function is minimized or maximized subject to equality constraints $U_i(x) = 0$ and inequality constraints $U_j(x) \leq 0$. The Linear Programming problem is stated formally as

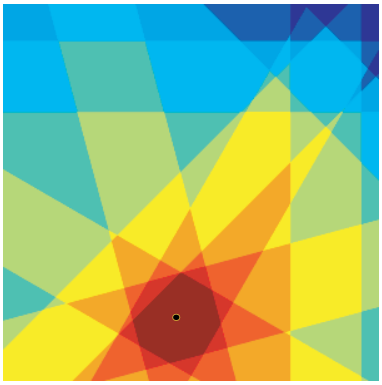
$$\begin{aligned} &\min \quad g^T \cdot x \quad \text{subject to:} \quad Ux \leq c \\ &x \in \mathbb{R}^d \end{aligned} \tag{37}$$

where x, g and c are column vectors and U is a matrix. Both equality and inequality constraints are defined using U and c; the convention is to list the equalities first in the rows of matrix U and vector c. To realize maximization in linear programming, supply $-g^T \cdot x$ to the minimization routine. [65, 34]

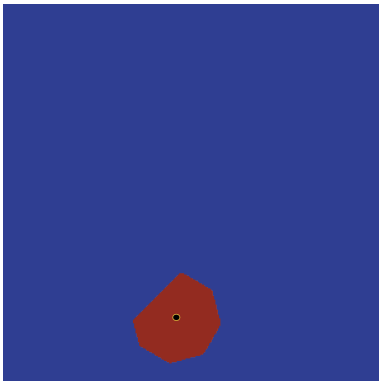
$$\begin{aligned} &\max \quad -g^T \cdot x \quad \text{subject to:} \quad Ux \leq c \\ &x \in \mathbb{R}^d \end{aligned} \tag{38}$$

Given a central chamber, the Linear Programing test checks each signed hyperplane in the arrangement to see if it is or is not a member of the chamber's minimal subarrangement. Given the signed diagonal $\overrightarrow{\phi} = (\phi_1, \ldots \phi_N)$ and a scaling matrix P constructed from the signed diagonal, the linear programming test states $\psi_n = 0$ if and only if $\phi_n w_n \cdot x_{\max} - \phi_n b_n < 0$ where the optimized point $x_{\max}$ is derived from the inequality constraints of the linear programming problem formation

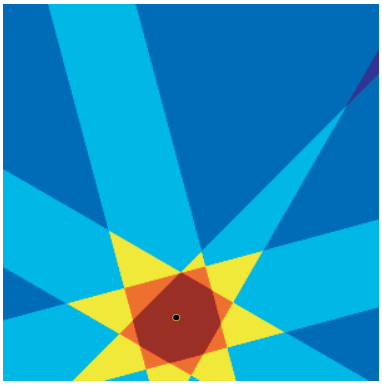
$$\begin{aligned} &\max \quad -\phi_n w_n \cdot x \quad \text{subject to:} \quad M_n P \overline{W} x \leq M_n P \overline{b}. \\ &x \in \mathbb{R}^d \end{aligned} \tag{39}$$



(a) Set of directed halfspaces



(b) Central chamber



(c) Minimal subarrangement

Figure 48. Given an arrangement with a central chamber C, linear programming determines the minimum subarrangement B in which chamber $C \in \text{Cham(B)}$ but the chamber is not in the chamber set of any subarrangement of B.

Recall, M_n is an aliasing matrix—an identity matrix with the n^{th} row removed—and strips parameters w_n, b_n from the parameter set $[\overline{W} : \bar{b}]$. Consider again the arrangement first depicted in Figure 40 as it has been manipulated in Figures 48(a)-(c). Figure 48(a) depicts the halfspace arrangement $\mathcal{A}' = \Lambda_*^{-1}[P\overline{W} : P\bar{b}]$ in which chamber C is the central chamber and Figure 48(b) renders chamber C as the intersection of the halfspace arrangement $\mathcal{A}'$.

Algorithm *psiChamber*[10] calculates the signed aliasing diagonal $\overrightarrow{\psi}(C, \mathrm{A})$ once the signed diagonal $\overrightarrow{\phi}$ for any chamber $C \in \mathrm{Cham(A)}$ is derived. Each non-zero element of $\overrightarrow{\psi}$ corresponds to a hyperplane in the minimum arrangement $\mathrm{B} \subset \mathrm{A}$ that implements chamber $C(x)$. Let matrix M be a signed aliasing matrix such that $M = \mathrm{diag}(\overrightarrow{\psi})$. Figure 48(c) depicts the set of directed hyperplanes for the minimum arrangement $\mathrm{B} \subset \mathrm{A}$ with respect to chamber C.

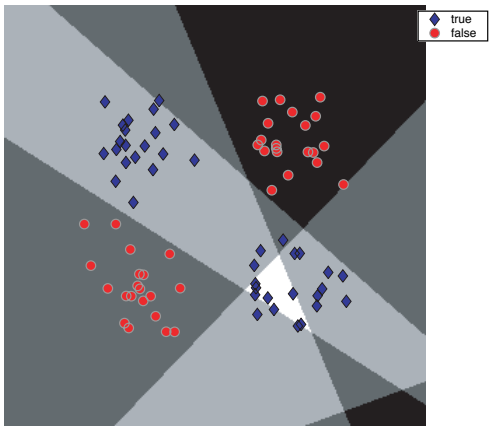
Once the set of signed aliasing diagonals is determined and stored in a matrix, an augmented set of signed diagonals can be determined to specify both the set of populated chambers $Q_{\mathfrak{D}}$ and those chambers adjacent to the populated chambers $[Q_{\mathfrak{D}}]_{adj}$. Vector $\overrightarrow{\psi}$ is calculated via Algorithm *psiChamber*. Given $\overrightarrow{\psi}$ and $\overrightarrow{\phi}$ for a chamber C, Algorithm *adjChambers*[11] computes the signed diagonals for the adjacent chambers to C and augments matrix Φ accordingly. Let matrix Ψ store the set of known signed aliasing diagonals. To resolve the chamber set Cham(A), we iterate between augmenting the matrix Φ of signed diagonals and resolving the matrix Ψ of signed aliasing diagonals via Linear Programming. Ordering Cham(A) via set function $\tilde{\rho}$ is then trivial.

6.3.1.3 Ordering a set of chambers. For a demonstration of the ordered veracity-experience curve, let us order a chamber set implemented by the weights and biases of a single-hidden layer MLP. The architecture of the multilayer perceptron is set so that each hidden-layer perceptron uses a tansig activation function and the output-layer perceptrons implement a linear activation function allowing us to visualize the ordering of each chamber. Let the MLP's unbiased output take the form $z = \sum_{n=1}^{N=5} u_n f(\alpha_n(x))$ where input feature $x \in \mathcal{X}$, output $z \in \mathbb{R}$, and vector $u \in \mathbb{R}^N$ specifies the output layer

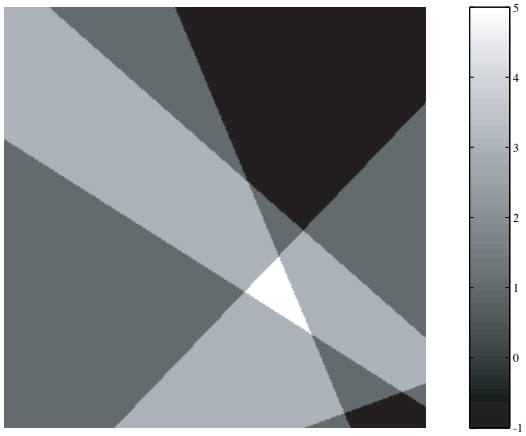
[10] See Figure 47.
[11] See Figure 49.

```
% adjChambers.m
% find set of adjacent chambers
 function adj_C=adjChamber(phi_C,psi_C)
% Get specifications for chamber C
 store phi_C: signed diagonal
 store psi_C: signed aliasing diagonal
% Index bounding hyperplanes
 indexFlippers=find(psi_C~=0);
% Specify one chamber for each
%   bounding hyperplane
 adj_C=[];
 for k=1:length(indexFlippers)
  h=indexFlippers(k);
  adjacentChamber=phi_C;
  % flip sign of one bounding hyperplane
  adjacentChamber(h)=-phi_C(h);
  adj_C=[adj_C adjacentChamber];
 end % for
% Return set of adjacent chambers
 return adj_C: adjacent chamber set
% end adjChambers.m
```

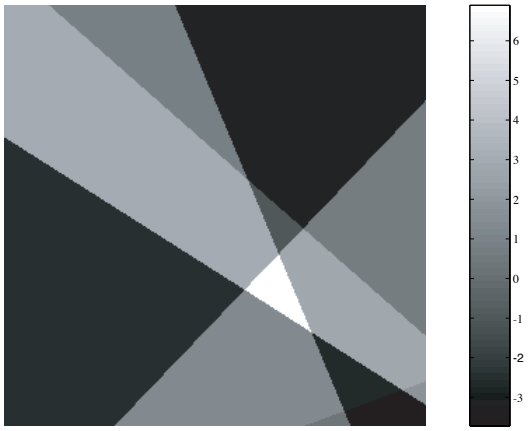
Figure 49. Pseudo code describing Algorithm *adjChambers*.



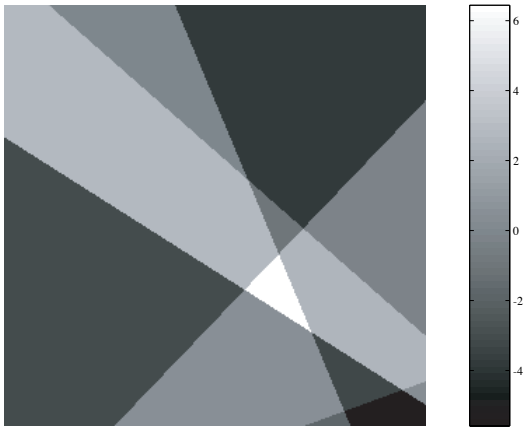
(a) MLP output with data overlay



(b) sum of an MLP's first-hidden-layer perceptrons $\sum_{n=1}^{N} f(\alpha_n(x))$



(c) weighted summation $\sum_{n=1}^{N} u_{1,n} f(\alpha_n(x))$



(d) weighted summation $\sum_{n=1}^{N} u_{2,n} f(\alpha_n(x))$

Figure 50. The ordering of a chamber set implemented by an MLP with $N = 5$ first-hidden layer perceptrons and three different sets of output layer weights.

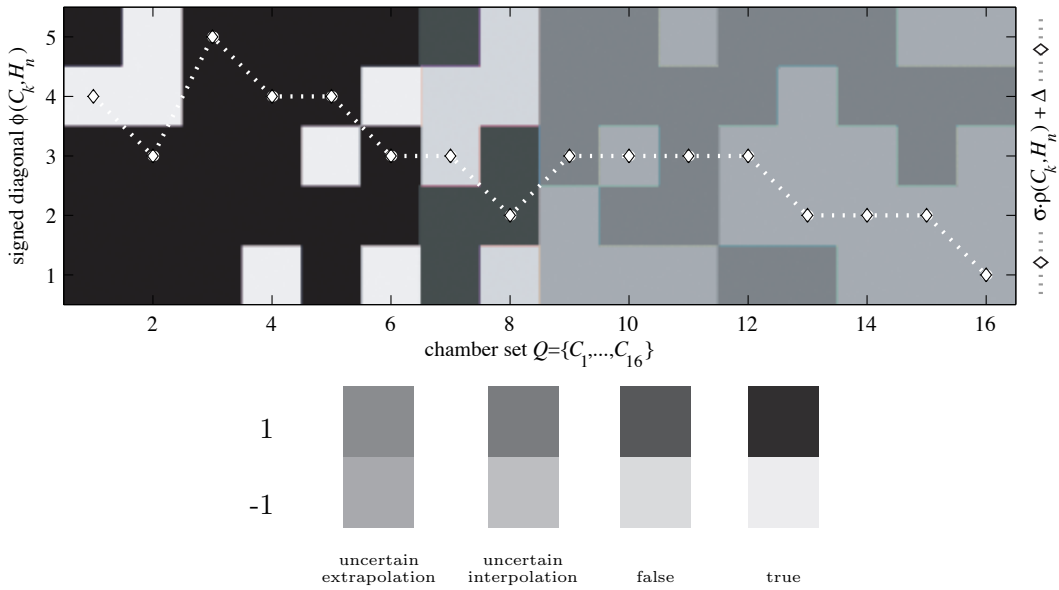
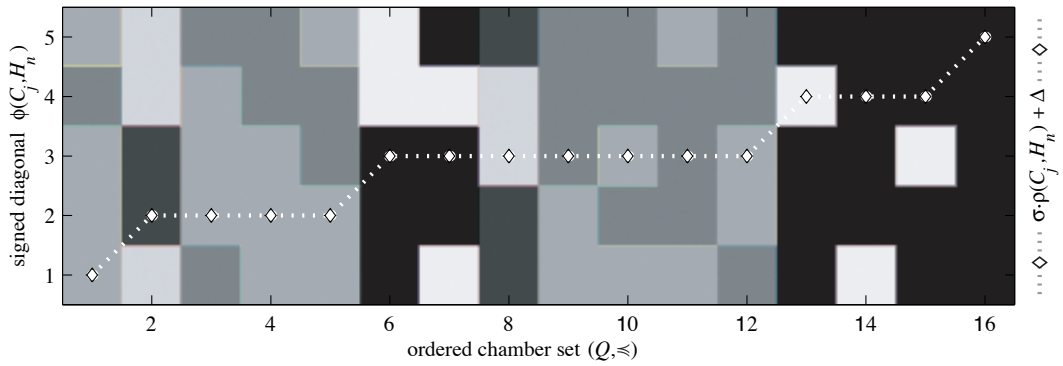
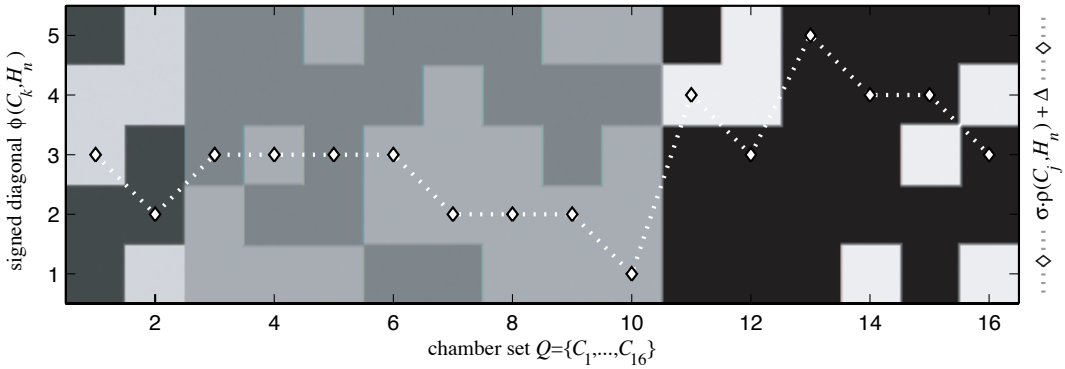


Figure 51. Given an arrangement $\mathrm{A} = \{h_1, h_2, \ldots, h_N\}$ of signed hyperplanes, the complete set of chambers—populated and unpopulated—is presented in the order each chamber was resolved. The graph includes an overlay of each chamber's value of set function $\rho(C_k, \mathrm{A})$.

weights (sans bias). We trained a 5-hidden node, 2-output node MLP that implements the XOR dataset given in Table 5 of Appendix D. The MLP's parameter sets are given in Equations 50 and 51, Appendix D. Solution sets from this MLP are presented in Figure 50 such that the ordering of the chamber set is depicted in grayscale over an interesting subset of the 2-dimensional feature set. Solution (b) illustrates the natural ordering of the first hidden layer's arrangement of signed hyperplanes where the weight vector $u = (u_1, \ldots, u_N)$ is set so that each output layer weight $u_n = 1$ for all $n \in \{1, \ldots, 5\}$. Figures 50(c) and (d) respectively show two output-layer parameter sets that successfully dichotomize the XOR training data. Figure 51 presents the complete set of signed diagonals designating all chambers in Cham(A)—populated and unpopulated—in the order each chamber was resolved using Algorithms *populatedQ* and *adjChambers*. The graph includes an overlay of each chamber's value of set function $\rho(C_k, \mathrm{A})$. For display purposes, the overlay of the function has been scaled and shifted such that $\sigma > 0$ and $\Delta \in \mathbb{R}$. Note, this figure is not an OVER curve as the chambers have not been reordered to reflect an ordering of $\rho(C_k, \mathrm{A})$. For a sample OVER curve, Figure 52(a) presents the natural ordering of chambers where

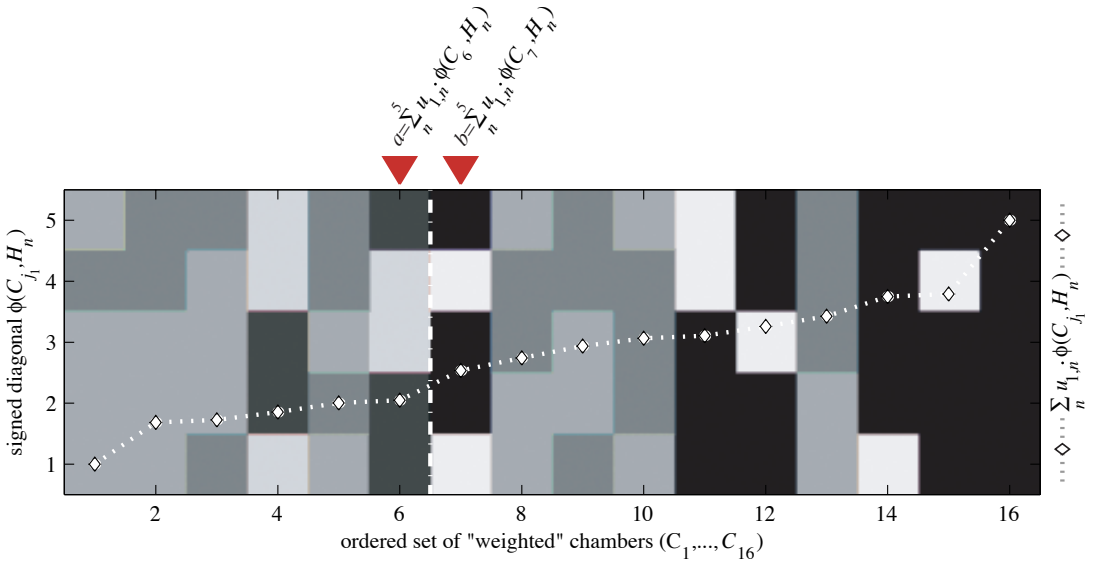


(a)

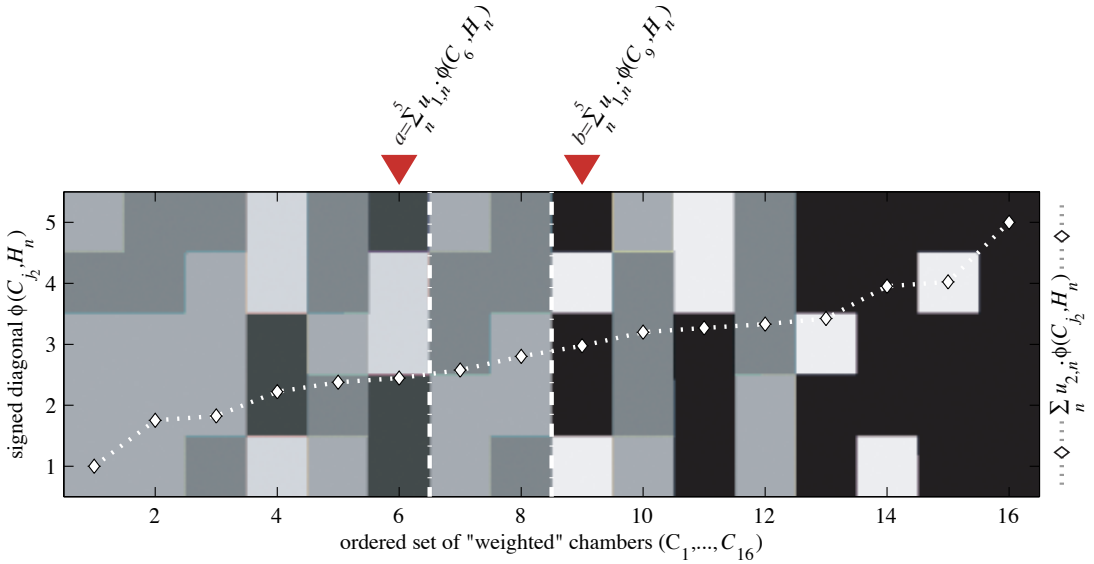


(b)

Figure 52. Given an arrangement $\mathrm{A} = \{h_1, h_2, \ldots, h_N\}$ of signed hyperplanes, (a) the natural ordering of chambers versus (b) the preferred order.



(a)

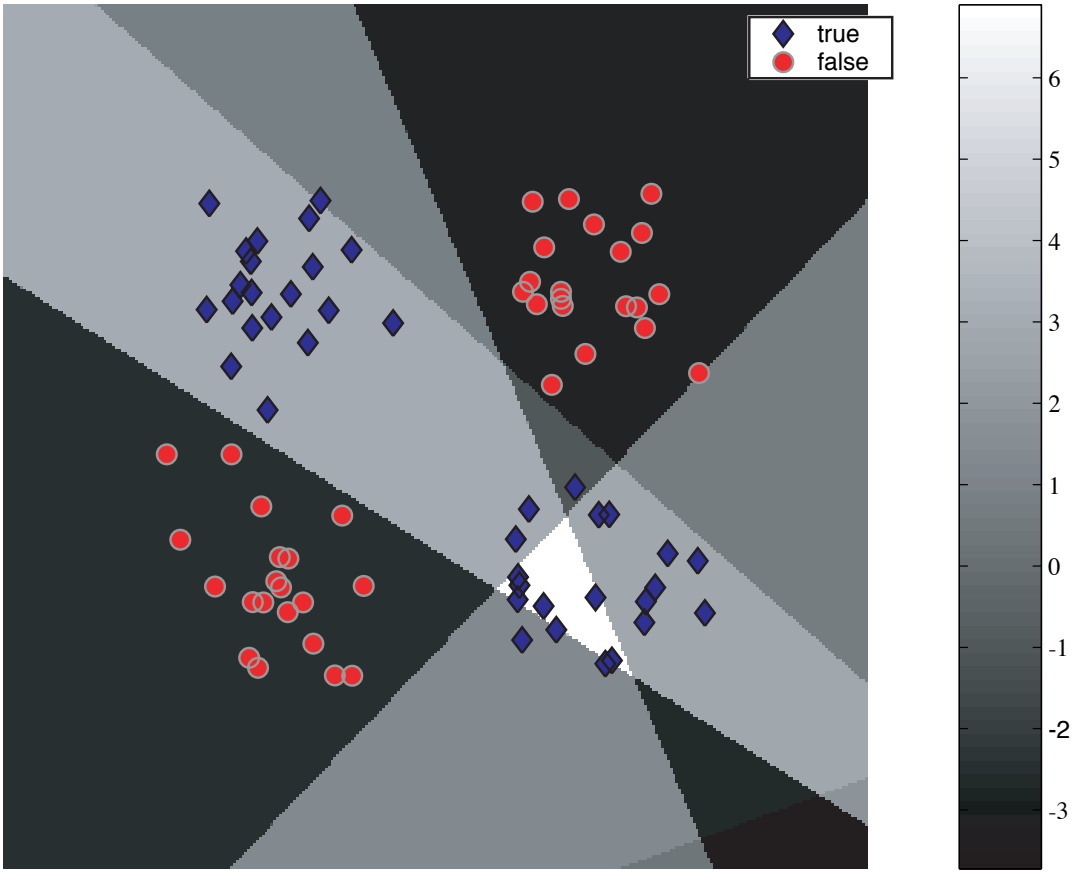


(b)

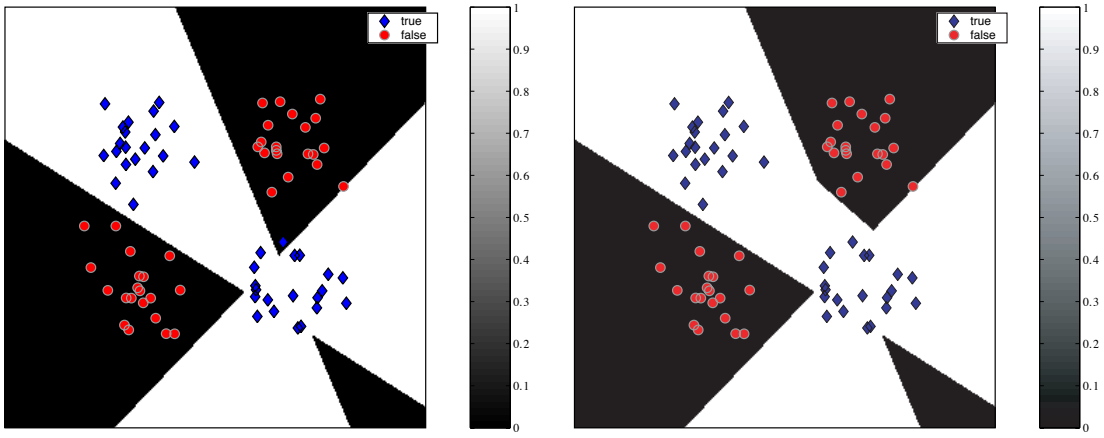
Figure 53. Given an arrangement $\mathrm{A} = \{h_1, h_2, \ldots, h_N\}$ of signed hyperplanes and two weight vectors $u_1 = (u_{1,1}, \ldots, u_{1,N})$ and $u_2 = (u_{2,1}, \ldots, u_{2,N})$, the complete set of chambers—populated and unpopulated—is weighted by u_1 and u_2 respectively and reordered. Note, using either weight vector, the populated chambers can be separated by class using a single threshold.

the outputs of the MLP's hidden-layer perceptrons are summed without weighting those outputs. Note, the natural ordering does not provide class separation. Figure 52(b) presents the preferred order of the chambers where "true" and "false" class data are separable and, additionally, unpopulated chambers are easily separable from populated chambers. OVER curves based on the two sets of output layer weights are depicted in Figure 53. Given weight vector $u_i = (u_{i,1}, \ldots, u_{i,N})$ from the i^{th} output node, it is possible to reorder the complete set of chambers—populated and unpopulated—to reflect the ordering of the MLP's output by vector multiplication $\rho_{u_i}(C, \mathrm{A}) = u_i \cdot \phi(C, \mathrm{A})$. Using either output-layer weight vector u_1 or u_2, the populated chambers can be separated by class using a single threshold t. Potential threshold values $t \in (a, b)$ are noted in Figures 53 and implemented in Figures 54 and 55. Two-valued separation {false,true} is achieved using the single threshold $t \in (a, b)$; three-valued separation {false,uncertain,true} is achieved using two thresholds a, b. Evaluated in terms of data reduction, both solutions perform well as both implement the training set. However, in terms of the generalization, both do poorly as noted by the arbitrary ordering of unpopulated chambers. Note, all or most of the unpopulated chambers fall outside of the "uncertain" interval (a, b).

In Figures 54 and 55, we have applied the orderings of a chamber set implemented by an MLP with $N = 5$ first-hidden layer perceptrons and two different sets of output layer weights to the solution set of the MLP by using the orderings to select appropriate threshold values for a and b. Threshold a gives the upper bound for the false class such that output $z \leq a$ is associated with false training data. Similarly, threshold b gives the lower bound for the true class such that $z \geq b$ is associated with true training data. Given a multilayer perceptron $F \in \mathcal{F}$ that implements the hyperplane arrangement $\mathrm{A} = \aleph_1(F)$, threshold a is found by determining the value of $a = \rho(C_{f,last}, \mathrm{A})$ where $C_{f,last}$ is the last chamber with false points in the set of ordered chambers. Threshold b is found by determining the value of $\rho(C_{t,first}, \mathrm{A})$ where $C_{t,first}$ is the first chamber with true points in the set of ordered chambers. These thresholds can be used to implement 2-valued separation in the MLP's solution set where thresholding with $t \in (a, b)$ results in a binary image as illustrated Figures 54(b) and 55(b). The thresholds can also be used to implement 3-valued separation as in Figures 54(c) and 55(c). Note, for weights give for this particular multilayer perceptron, there

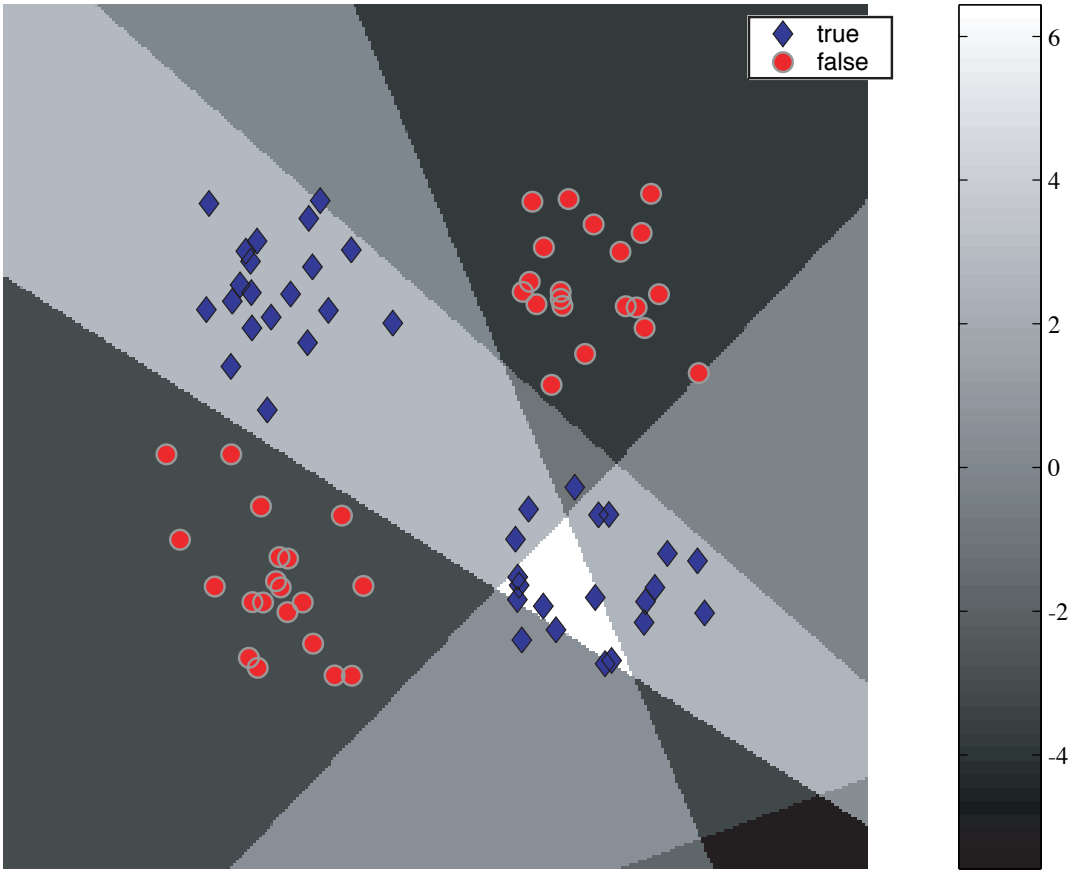


(a) weighted summation $\sum_{n=1}^{N} u_{1,n} f(\alpha_n(x))$

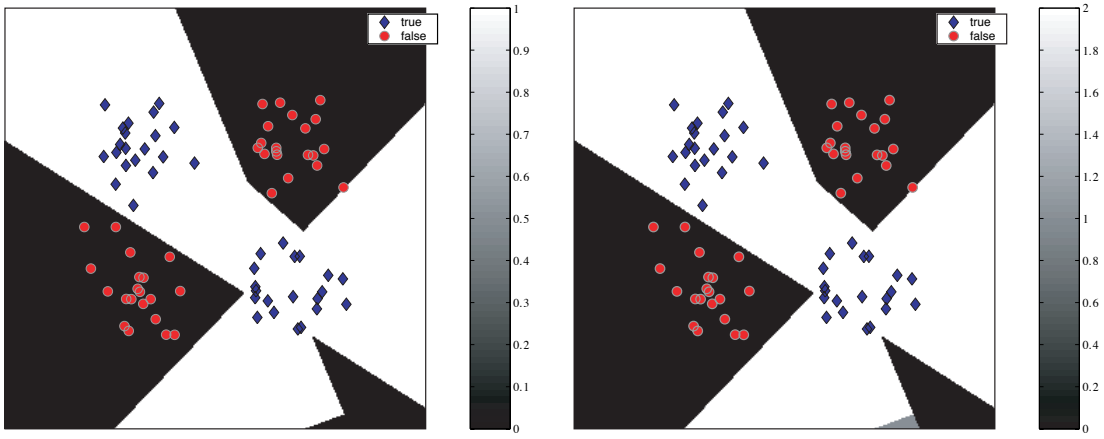


(b) two-valued separation (c) three-valued separation

Figure 54. Two-valued and 3-valued treatments for a 5-hidden-node MLP trained by backpropagation.



(a) weighted summation $\sum_{n=1}^{N} u_{2,n} f(\alpha_n(x))$



(b) two-valued separation (c) three-valued separation

Figure 55. Two-valued and 3-valued treatments for a 5-hidden-node MLP with different output weights.

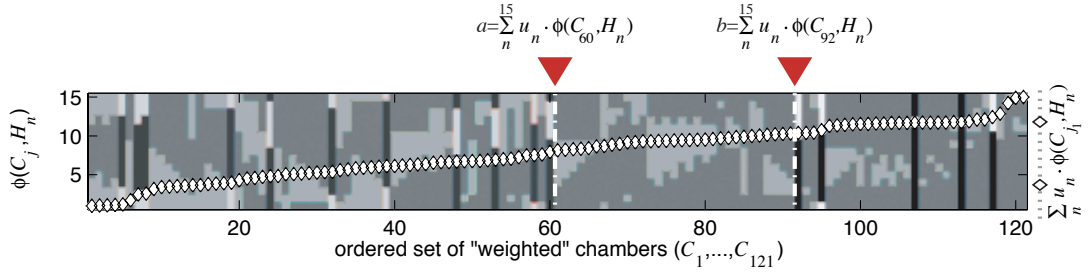
is little difference in the 2-valued separation and 3-valued separation images; this similarity is expected since there were no unpopulated chambers listed between chambers $C_{f,last}$ and $C_{t,first}$ in the ordered chamber set for the first weight vector u_1 and only two unpopulated chambers listed between chambers $C_{f,last}$ and $C_{t,first}$ in the ordered chamber set for weight vector u_2. Three-value logic adds little expression in these cases. We expect 3-value logic to produce confusing results in more complex solutions as is apparent when we apply the OVER curve to several multilayer perceptrons of greater complexity. Figure 56 portrays a 3-valued treatment for a complex network, a 15-hidden-layer-node MLP also trained by backpropagation. Chambers fall in the uncertain range in an arbitrary manner that is difficult to predict when the MLP architecture was initialized. We see similar confusion as we explore the orderings of the chamber sets implemented by MLPs as shown in Figure 57, where the number of first-hidden-layer nodes are $N = 3, 5, 10$ as specified. Figure 58 contains the respective 3-valued treatments of these multilayer perceptrons. Considering the populated chambers, the complex MLPs (where $N = 10$ or more) produce solution sets in which most training points have been isolated in finite chambers. There is no guaranteed, however, that all populated chambers are finite and, when they are, in many case the finite, populated chambers are smaller than is desirable leading to memorization. The association of unpopulated chambers to populated chambers is arbitrary leading to jagged, overly complex decision boundaries. Considering the populated chambers in the simpler MLPs ($N = 5$ or less), many of the populated chambers are infinitely large—i.e., unbounded—resulting in arrogant classification in the extremes of the unbound chambers.

Figure 59 illustrates the most promising chamber set ordering implemented by a simple 3-hidden-layer-node MLP whose solution set is first shown in Figure 57(a). Note the sole unpopulated chamber is ordered between the chambers $C_{f,last}$ and $C_{t,first}$ that provide our thresholds $a = \rho(C_{f,last}, \mathrm{A})$ and $b = \rho(C_{t,first}, \mathrm{A})$. Given this ordering where the true, false, and unpopulated chambers are grouped nicely, we can easily convert the MLP to an isolation architecture by adding 4 addition perceptrons to the MLPs hidden layer. These additional perceptrons serve a special purpose in that they wrap the entire training set in a bounded convex hull. These perceptrons are assigned large magnitude output weights (as recorded in Equation 53, Appendix D) to ensure that the chambers contained within the bounded

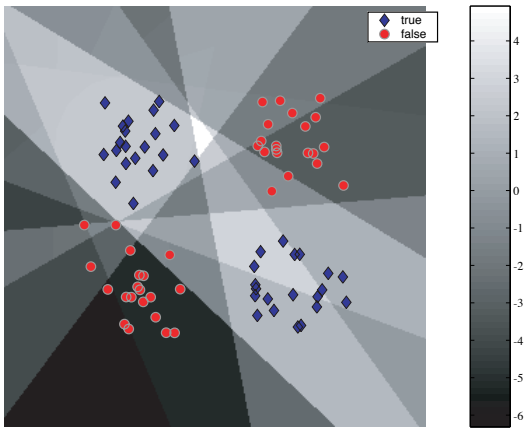
convex hull supersede all chambers outside the convex hull in the chamber ordering. Once we add the additional nodes to the first hidden layer, we recalculate thresholds a and b for the new chamber set and derive a third threshold $c = \rho(C_{f,first}, \mathrm{A})$ where $C_{f,first}$ is the first chamber with false points in the new set of ordered chambers. Figure 60 presents the results of a 4-valued isolation treatment to the augmented MLP such that, given the MLP output $z \in \mathbb{R}$ for a chamber $C \in \mathrm{Cham(A)}$, chambers where $z(C) < c$ are labeled as extrapolations, chambers where $c \leq z(C) \leq a$ are labeled as false, chambers where $a < z(C) < b$ are labeled as interpolations, and chambers where $z(C) \geq b$ are labeled as true.

Often, it is not possible to order all of the unpopulated chambers together as in the case above. Such cases require additional hidden layers to implement proper 4-valued isolation. To produce an alternate 4-value treatment for the multilayer perceptron, we must isolate the chambers to be labeled true, false, extrapolations and interpolations separately. The extrapolation class is determined by taking the set complement of the convex hull used to wrap all of the training data. Figure 61 presents an isolation treatment for "true" points in the 3-hidden-layer-node MLP. Figure 62 presents the isolation treatment for "false" points. The membership of the interpolation class are all those chambers not included in the membership of the true, false or extrapolation classes and does not need to be determined directly. Thus, the membership of the true, false and extrapolated class can be captured in a second hidden layer and the output node sums the weighted output from these class nodes. Figure 63(d) shows the solution set from this output node which combines the output of the class nodes given respectively in Figures 63(a), (c) and (d) to produce the alternate 4-value treatment for the multilayer perceptron.

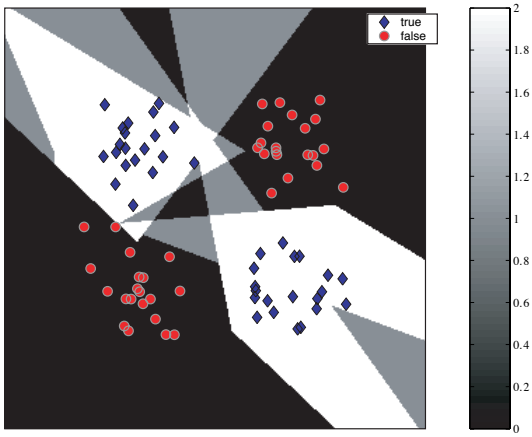
Thus, we have presented an application of the ordered veracity-experience curve and demonstrated its utility in understanding the manipulations of an multilayer perceptron over an entire feature set. Though our algorithms are designed to work in any dimension that is computationally feasible, our visualization of the OVER curve must be simplified for larger chamber sets. Later in the chapter, we shall present a simplified OVER curve and demonstrate arrogance in an MLP trained to solve a 4-dimensional problem.



(a) ordered chamber set

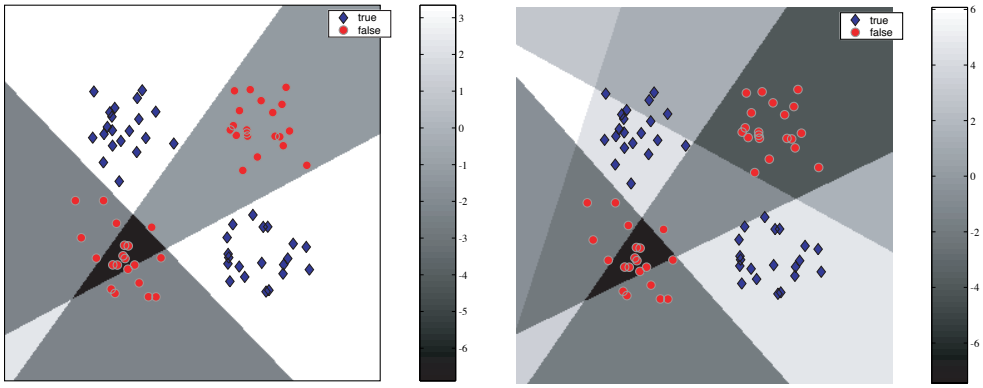


(b) weighted summation $\sum_{n=1}^{N} u_n f(\alpha_n(x))$

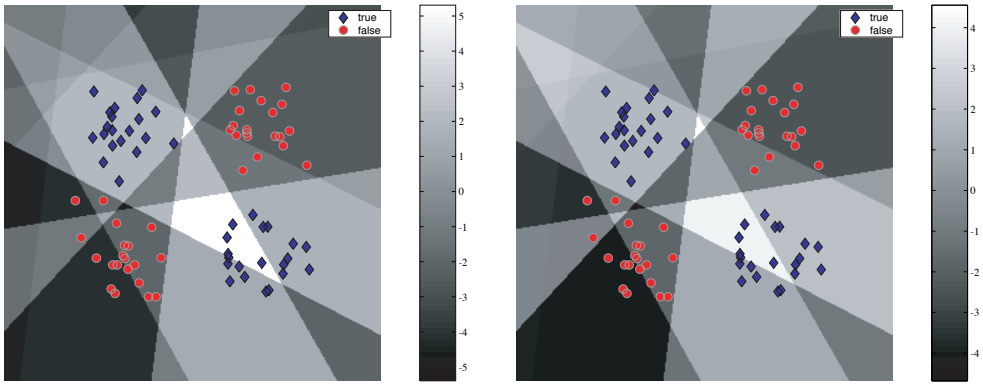


(c) three-valued separation

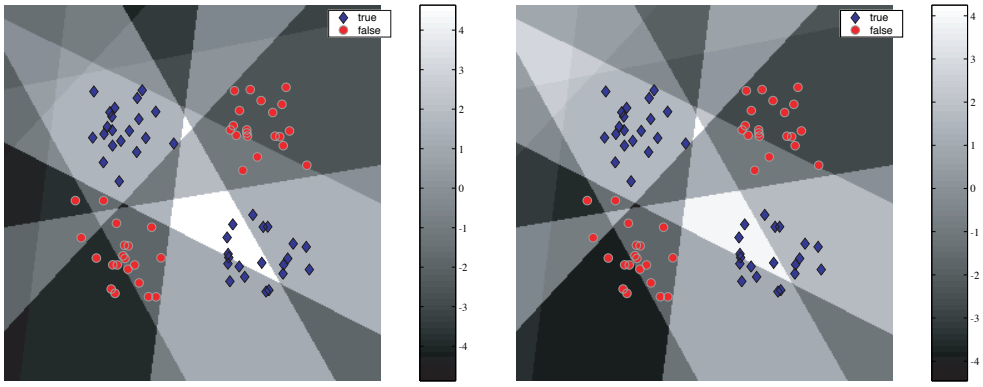
Figure 56. A 3-valued treatment for a 15-hidden-node MLP trained by backpropagation.



(a) $N = 3$ (b) $N = 5$

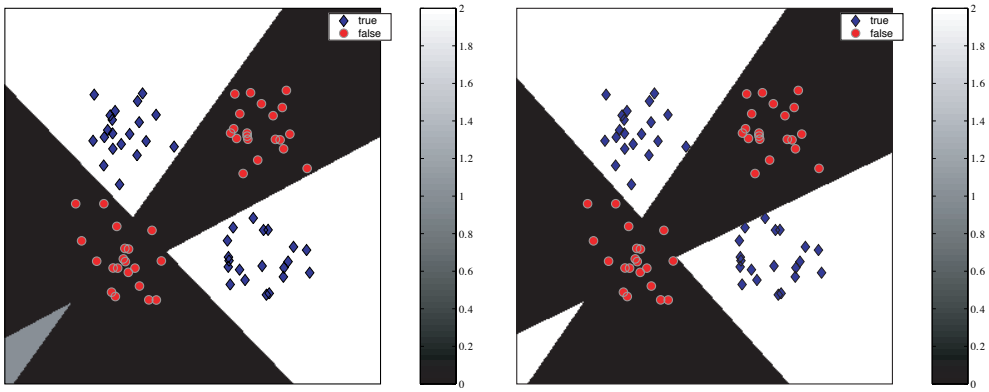


(c) $N = 10$ (d) $N = 10$



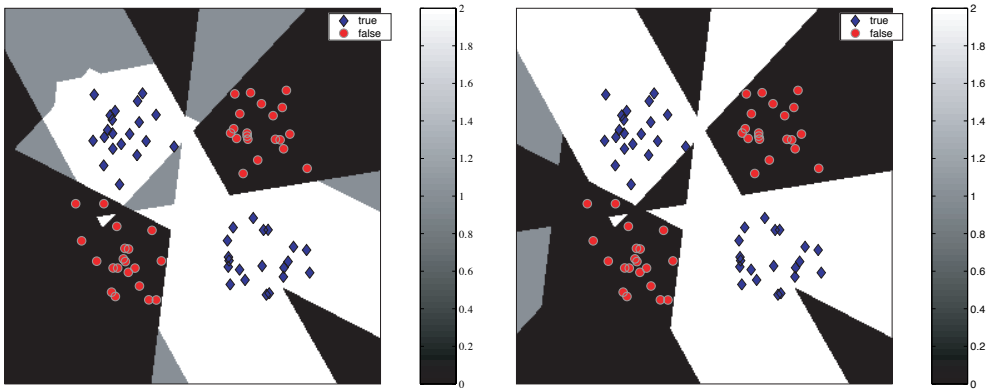
(e) $N = 10$ (f) $N = 10$

Figure 57. The ordering of various chamber sets implemented by MLPs with $N = 3, 5, 10$ first-hidden layer perceptrons. The weights and biases used in these MLPs are listed in Appendix D.



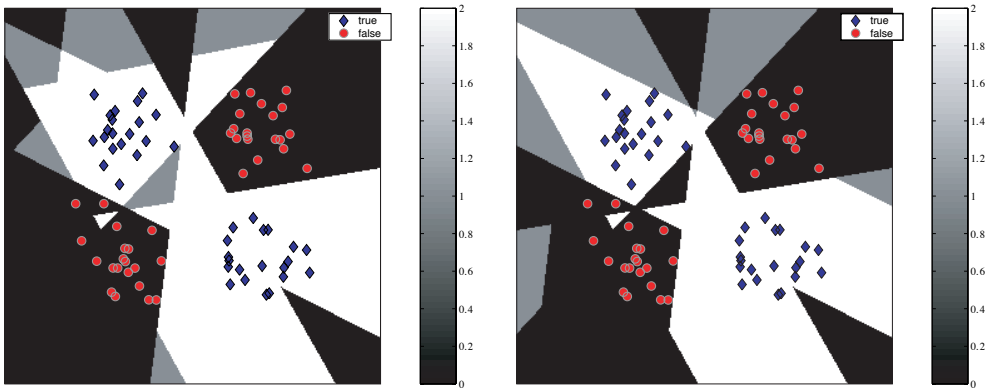
(a) $N = 3$

(b) $N = 5$



(c) $N = 10$

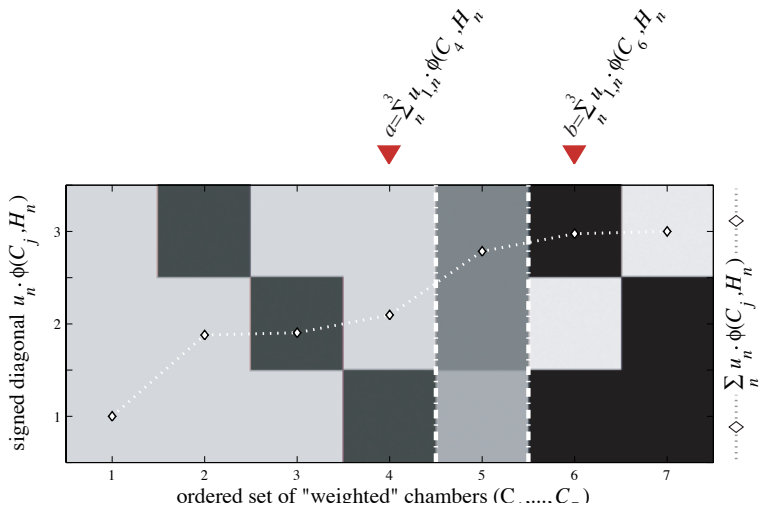
(d) $N = 10$



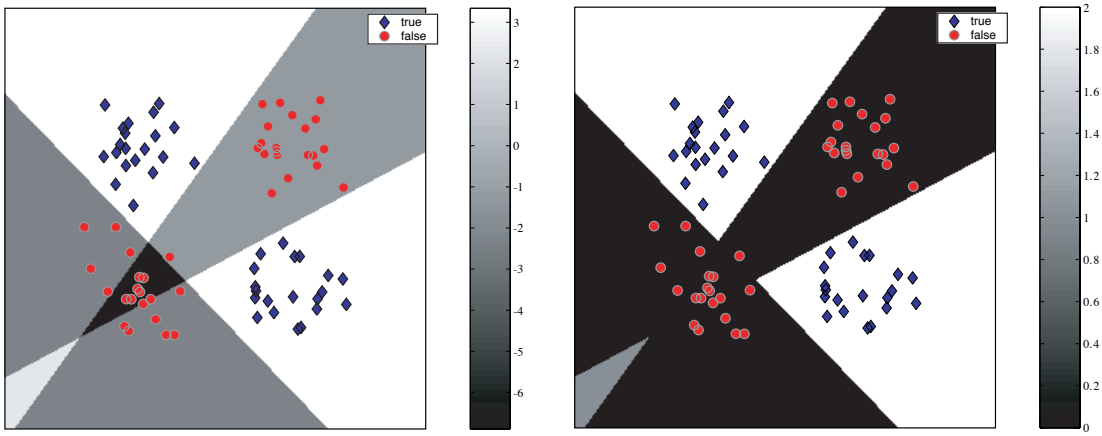
(e) $N = 10$

(f) $N = 10$

Figure 58. Three-valued treatments of various MLP with $N = 3, 5, 10$ first-hidden layer perceptrons.

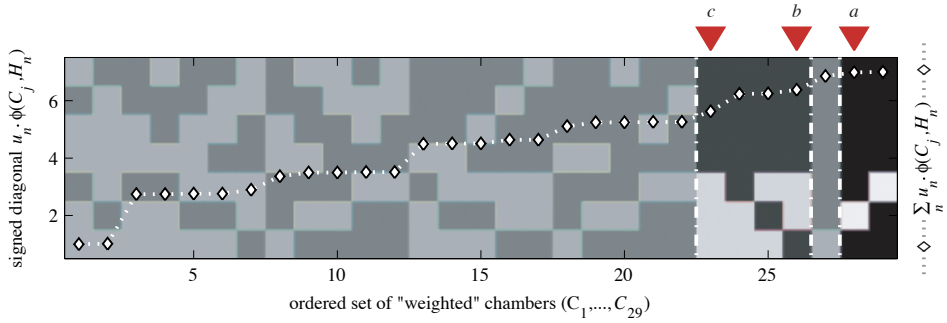


(a) ordered chamber set

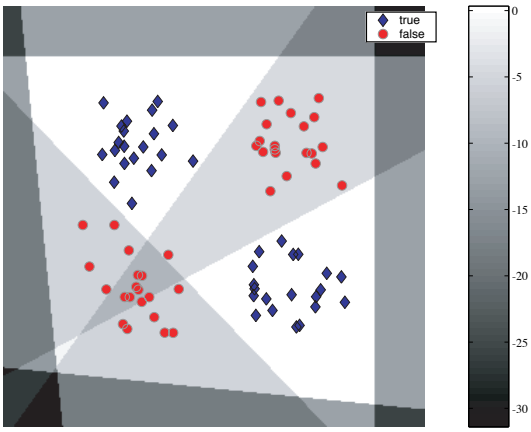


(b) weighted summation $\sum_{n=1}^{N} u_n f(\alpha_n(x))$ (c) three-valued separation

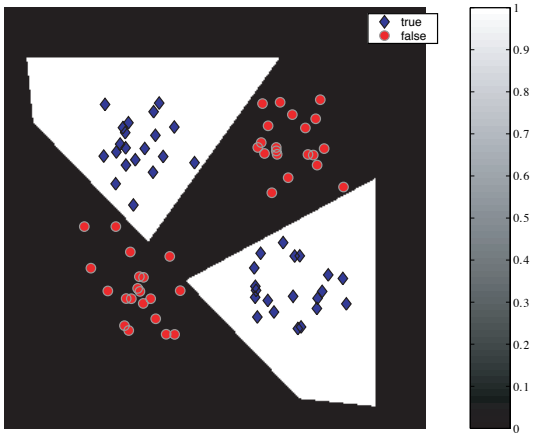
Figure 59. A 3-value treatment for a 3-hidden-node MLP.



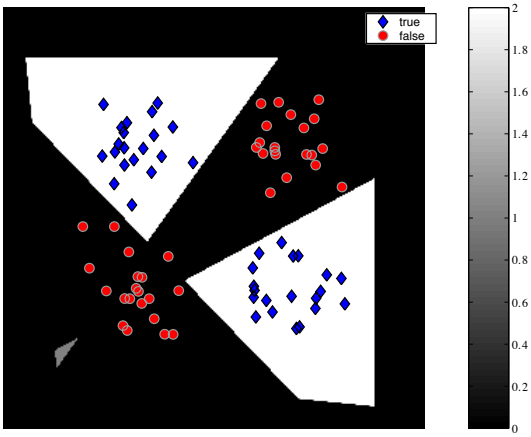
(a) ordered chamber set



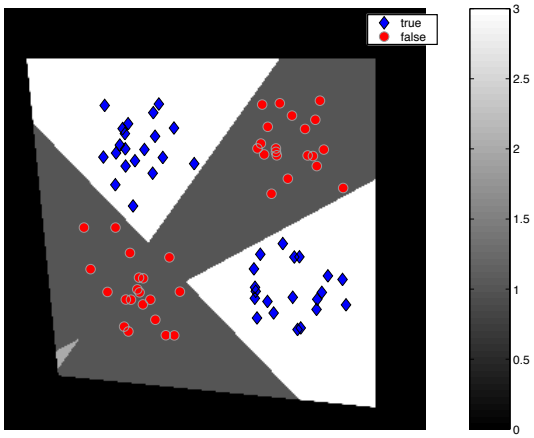
(b) chambers ordered by grayscale



(c) two-valued separation

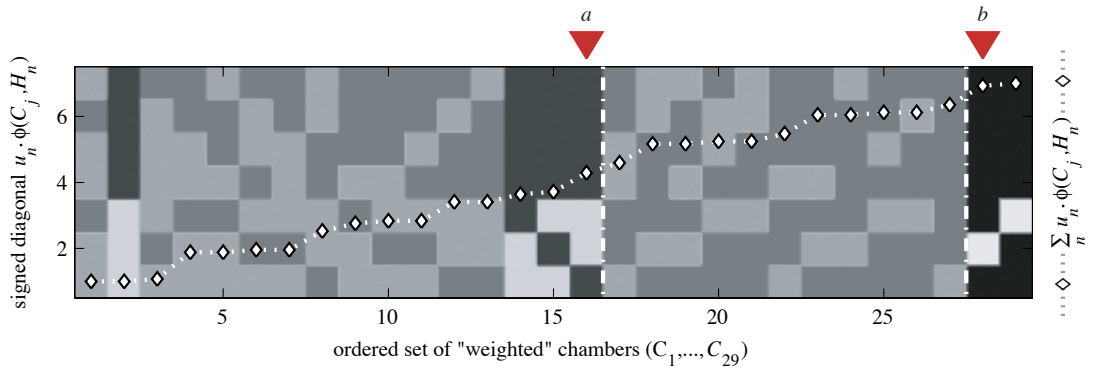


(d) three-valued separation

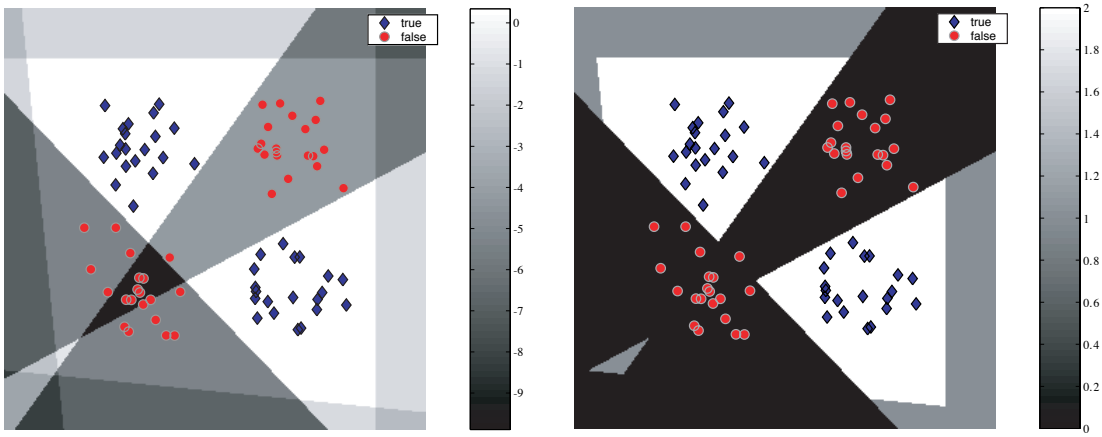


(e) four-valued isolation

Figure 60. A 4-valued treatment for a 3-hidden-node MLP augmented with 4 additional hidden nodes.

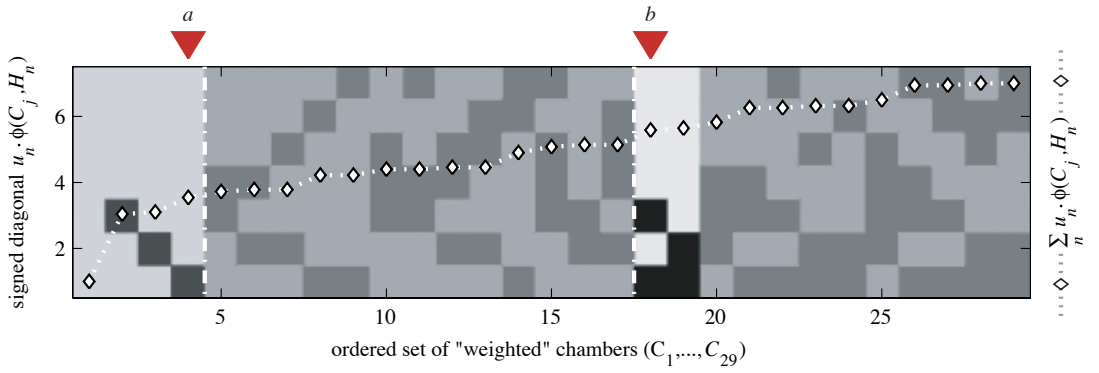


(a) ordered chamber set

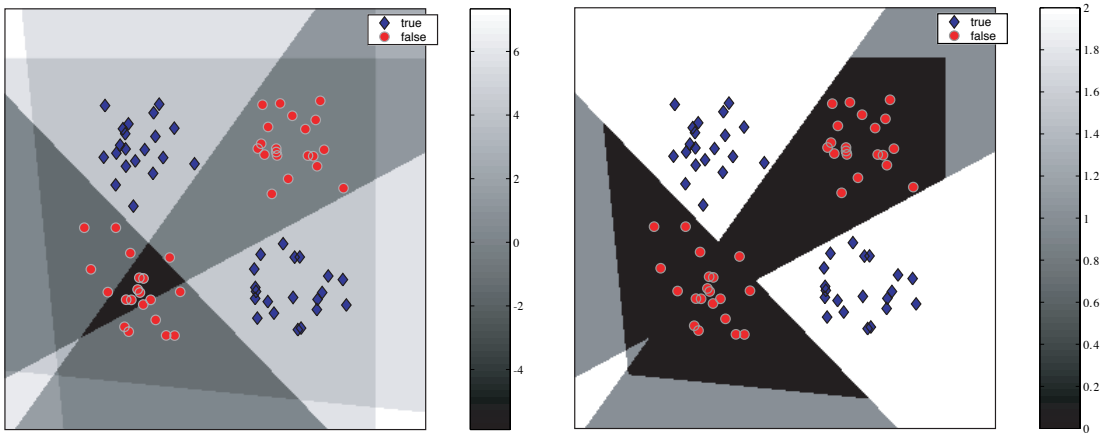


(b) weighted summation $\sum_{n=1}^{N} u_n f(\alpha_n(x))$ (c) three-valued separation

Figure 61. An isolation treatment for "false" points in the 3-hidden-node MLP.

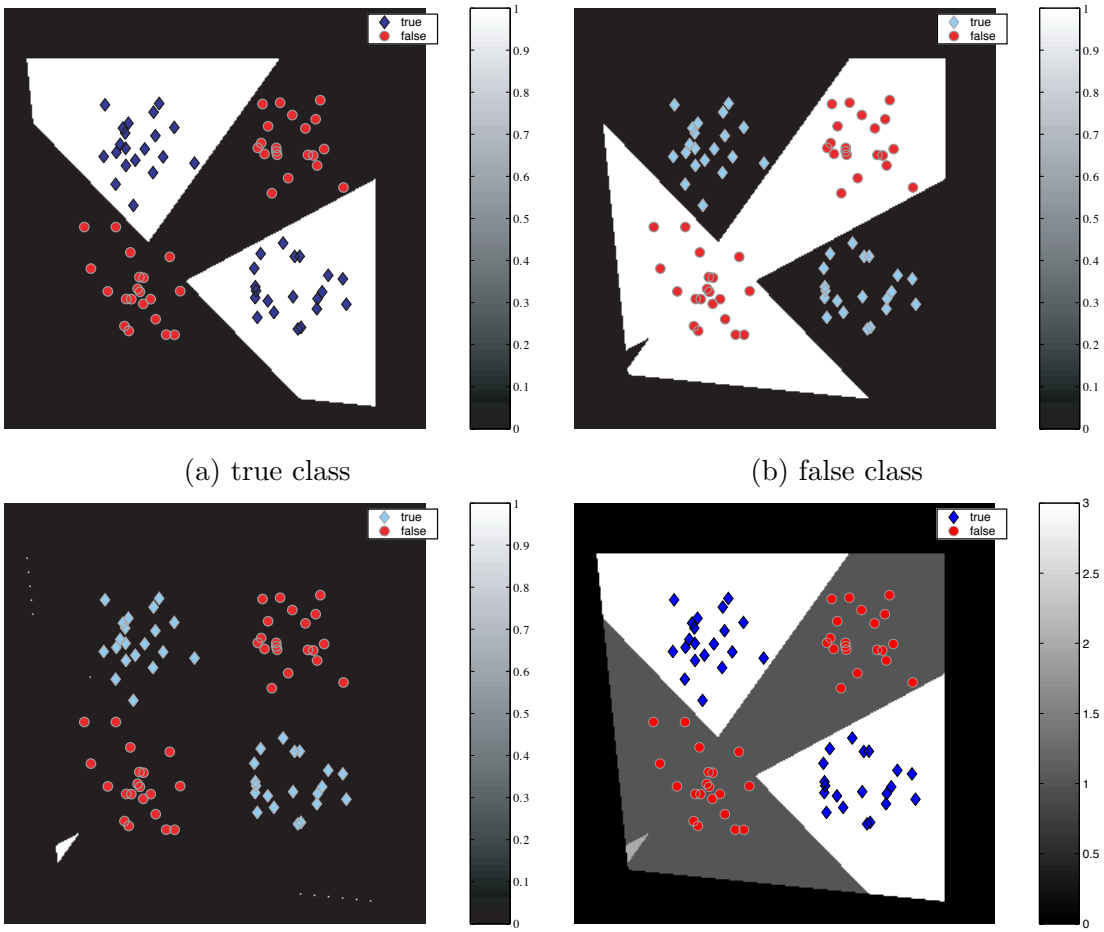


(a) ordered chamber set



(b) weighted summation $\sum_{n=1}^{N} u_n f(\alpha_n(x))$ (c) three-valued separation

Figure 62. An isolation treatment for "false" points in the 3-hidden-node MLP.



(a) true class

(b) false class

(c) uncertain class

(d) four-valued isolation

Figure 63. An alternate 4-valued treatment for a 3-hidden-node MLP.

6.3.2 Local max and local min chambers. As the dimension of a feature set increases, it quickly becomes implausible to resolve an entire chamber set. To keep computations to a minimum, it is wise to use perceptrons sparingly and implement architectures where first hidden-layer perceptrons merely separate class data in the feature set and additional layers isolate and order the partitions. This keeps the number of chambers implemented by the first layer of the MLP down. It is a good rule of thumb to limit the number of bounded chambers[12] implemented by a multilayer perceptron to the number of training points.

To extend the ordered-veracity scheme into higher dimensions, we shall define a much smaller subset of "interesting" chambers. The size of a chamber set grows exponentially in relation to d, the dimension of the feature set, and polynomially in relation to N, the cardinality of the hyperplane arrangement as seen in Equation 25. Thus, the computational complexity of any algorithm that derives an entire chamber set grows exponentially.

As d and N increase, determining the ordering of an entire chamber set becomes burdensome both in terms of CPU time and memory space. To work on more complex problems in a reasonable amount of memory, we propose an interesting subset of the chamber set: the set of local min chambers and local max chambers. In the remaining sections of this chapter, we define these interesting chambers and explain why they are so interesting. Appendix C presents some of the characteristics and behaviors of these chambers for use in developing prunable search routines that resolve local max and local min chambers in a reasonable amount of time.

Definition 44 *(Local Max Chamber). Given a signed hyperplane arrangement* A *and a chamber* $C \in \text{Cham(A)}$ *with minimal subarrangement* $\text{B} \subset \text{A}$ *per Definition 43, we say chamber* C *is a local max chamber of* A *if*

$$C = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H.$$

[12] See Section C.2 for a definition of a bounded chamber.

Definition 45 *(Local Min Chamber). Given a signed hyperplane arrangement* A *and a chamber* $C \in \text{Cham(A)}$ *with minimal subarrangement* $\text{B} \subset \text{A}$ *per Definition 43, we say chamber* C *is a local min chamber of* A *if*

$$C = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}.$$

Ordering the local max and min chambers of a chamber set simplifies the evaluation of single-hidden layer MLP solutions as local max and min chambers correspond respectively to the local peaks and valleys of the MLP's output. Given a threshold ρ_0, if a local max chamber $C_{\max}$ does not survive the threshold—i.e., $\rho(C_{\max}, \text{A}) \leq \rho_0$—then from Theorem 42 it follows that none of the chambers adjacent to $C_{\max}$ survive the threshold—i.e., $\rho(C', \text{A}) \leq \rho(C_{\max}, \text{A}) \leq \rho_0$ for all $C' \in [C_{\max}]_{adj}$. It also follows from Theorem 42 that if a local min chamber $C_{\min}$ does survive the threshold—i.e., $\rho(C_{\min}, \text{A}) \geq \rho_0$—then all chambers adjacent to $C_{\min}$ also survive the threshold—i.e., $\rho(C'', \text{A}) \geq \rho(C_{\min}, \text{A}) \geq \rho_0$ for all $C'' \in [C_{\min}]_{adj}$.

```
% localChamber.m
% determine if a chamber C is a local
% max or min chamber
 function local_C=localChamber(phi_C)
% Get specifications for chamber C
 store psi_C: signed aliasing diagonal
% Test for local chamber behavior
 localMax=sum(abs(psi_C))==sum(psi_C);
 localMin=sum(abs(psi_C))==□sum(psi_C);
 local_C=localMax-localMin;
% Return state of chamber C
%   local_C=1 if local max chamber
%   local_C=-1 if local min chamber
%   local_C=0 if neither
 return local_C: state of chamber C
% end localChamber.m
```

Figure 64. Pseudo code describing Algorithm *localChamber*.

6.3.2.1 Selecting local max and min chambers. Local max and min chambers are distinguishable by their signed aliasing diagonals $\overrightarrow{\psi}$. Recall from Section 6.3.1.2 that, given a signed hyperplane arrangement A, a chamber $C \in \text{Cham(A)}$ has a signed aliasing diagonal $\overrightarrow{\psi}(C, \Lambda_*^{-1}[\Lambda(\text{A})]) \in \{-1, 0, 1\}^N$. A local max chamber $C_{\max} \in \text{Cham(A)}$ has a signed aliasing diagonal such that $\overrightarrow{\psi}(C_{\max}, \Lambda_*^{-1}[\Lambda(\text{A})]) \in \{0, 1\}^N \subset \{-1, 0, 1\}^N$. Similarly, a local min chamber $C_{\min} \in \text{Cham(A)}$ has a signed aliasing diagonal such that $\overrightarrow{\psi}(C_{\min}, \Lambda_*^{-1}[\Lambda(\text{A})]) \in \{-1, 0\}^N$.

Let the set of local max and min chambers be denoted $Q_o \subset \text{Cham(A)}$. We define $Q_o = Q_{\min} \cup Q_{\max}$ where the set of local min chambers is $Q_{\min} = \{C \in \text{Cham(A)} : \gamma(\overrightarrow{\psi}(C, \text{A})) = -1\}$ and the set of local max chambers is $Q_{\max} = \{C \in \text{Cham(A)} : \gamma(\overrightarrow{\psi}(C, \text{A})) = 1\}$. Figure 65 shows the chamber set depicted in Figure 50(a) contains one local max chamber and three local min chambers. Chamber subset Q_o was derived by testing each signed aliasing diagonal in matrix Ψ via Algorithm *localChamber*[13]. Given a hyperplane arrangement A and a chamber $C \in \text{Cham(A)}$, the Algorithm *localChamber* can be summarized in the mapping γ as

$$\gamma(\overrightarrow{\psi}) = \begin{cases} 1 & \text{if } \sum_n^N |\psi(C, h_n)| = \sum_n^N \psi(C, h_n) \\ -1 & \text{if } \sum_n^N |\psi(C, h_n)| = -\sum_n^N \psi(C, h_n) \\ 0 & \text{otherwise} \end{cases} .$$

where $h_n \in \text{A}$.

In the example of Figure 65, the overlaid plot of $\rho(Q_o, \text{A})$ shows that, for a particular set of weights, the local max chamber survives threshold $\rho_0 = 0$ while the local min chambers do not. As such, our evaluation of the MLP's generalization focuses on the local max chamber $C_{\max}$ and resolving those chambers near $C_{\max}$. Of particular interest are the set of chambers near the decision boundary between what is considered near $C_{\max}$ and what is near the local min chambers—i.e., those chambers that just survive the threshold (i.e., $\rho_0 + \epsilon \geq \rho(C, A) \geq \rho_0$ for small $\epsilon > 0$) and those that just do not (i.e., $\rho_0 \geq \rho(C, A) \geq \rho_0 - \epsilon$ for small $\epsilon > 0$).

[13] See Figure 64.

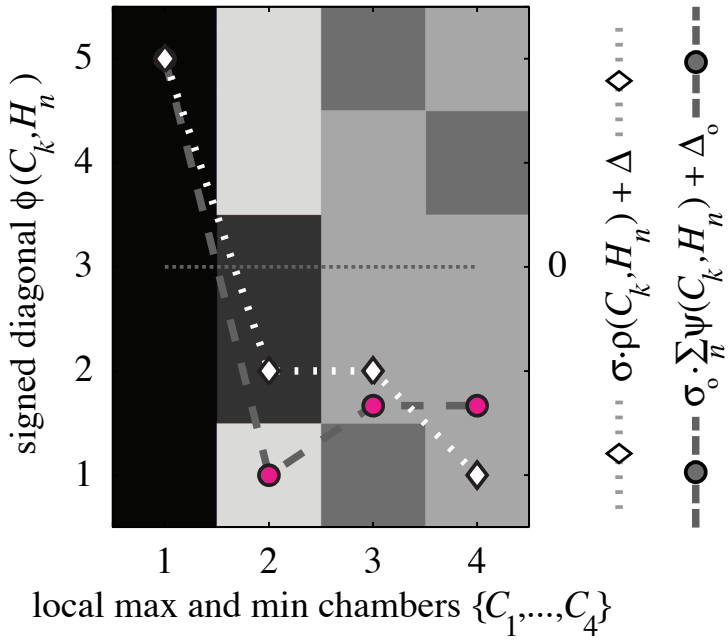


Figure 65. Given the arrangement A in Figure 50(a), the subset of local max and local min chambers. The arrangement implements one local max chamber whose $\rho(C, \mathrm{A}) > 0$ and 3 local max chambers whose $\rho(C, \mathrm{A}) < 0$.

6.3.3 Summary. In this section, we presented the means to succinctly view the partial ordering imposed by a multilayer perceptron over a feature set. This representation of $(\mathrm{Class}(\mathrm{Cham}(\mathrm{A})), \preccurlyeq)$ allows for informed manipulation of the partial ordering over both populated and unpopulated disjoint subsets. The representation $(\mathrm{Class}(\mathrm{Cham}(\mathrm{A})), \preccurlyeq)$ also accomplishes the first part of the OVER curve—that is, the ordered veracity. Further, we presented simple OVER curve representations by overlaying the ordered veracity scores over grayscale-coded representations of each chamber such that the grayscale-coding portrayed the chamber's experience (i.e., whether a chamber contained true, false, both or no training points); however, this form of visualization will not be appropriate for problems whose chamber sets have a cardinality $N > 100$. In the next section, we shall develop an OVER curve representation appropriate for larger chamber sets and apply the curve to a 4-dimensional feature set, the Fisher iris dataset.

6.4 The arrogant multilayer perceptron

As the literature shows, quantifying generalization performance has several approaches based on error measures. Besides being computationally expensive, training methods such as cross validation and bootstrapping do not support persistent learning where it is desirable to test the appropriateness of a data generalization beyond supervised design and into unsupervised application. In application, additional observations will be gathered and new experience may be gained. Hold-out validation tracks error trends of isolated labeled data; but, after training is complete and a model solution is selected for application, we wish to continue tracking patterns in the operational data. Error trends require supervision; we need other trends to investigate the apparent incompleteness in the training set with respect to the unlabeled operational set. The ordering of chambers constructed by an MLP is such a trend—that is, one that does not require additional supervision past the design stage.

For a multilayer perceptron, we detail where in a feature set the classifier is arrogant and non-arrogant as we quantify the overall generalization performance. In this treatment, we use the output of a MLP for the veracity measure and a four-value logic interpretation of the MLP's arrangement of hyperplanes to express relative experience over an entire feature set. To demonstrate arrogance, we shall first define our measures that map a multilayer perceptron's veracity and experience across partitions of the feature set. Then, using the veracity scores we order the MLP's domain based on the veracity measures and cross reference the experience measures to form an ordered veracity-experience response curve. Finally, we compare the OVER curve to Figures 28(a) and 28(b) from Chapter V—respectively, the ordering we desire for a classifier with good isolation and the ordering we expect from an arrogant classifier.

As previously listed, the benefits of our approach over other iterative and stochastic methods can be summed up in the following: (1) Our technique evaluates the generalization of an MLP after model selection, requiring only the classifier's weights and biases and the data used to train these parameters. (2) The results of our evaluation do not change based on the order that data are presented. (3) Stochastic techniques exercise little of the domain—that is, the measure of the set of feature vectors evaluated is zero whereas, for our technique, the measure of such sets used is positive.

6.4.1 Ordering the domain of a multilayer perceptron. Our new method of resolving and ordering populated chambers allows one to evaluate the quality of a training set in view of the MLP's representation of the operational set. Let us define *unpopulated chambers of interest* as the chambers unpopulated by training data but populated by operational data. To evaluate the effectiveness of an existing MLP design, we employ the steps below.

1. Gauge memorization: Track the operational data "hits" in populated chambers versus hits in unpopulated chambers. If the ratio of hits in the populated chambers to hits in unpopulated chambers of interest is near 0, then the MLP trained by back propagation has memorized the data. If the ratio of hits in the populated chambers to hits in unpopulated chambers of interest is 1, then the MLP has not memorized the data.
2. Apply a consistency test: Compare the adjacency of populated chambers to unpopulated chambers of interest and test if the local ordering of chambers make sense.

From here, we can determine a desired re-ordering of the local chambers and update the MLP architecture and parameters appropriately. We can also determine if the network architecture should be simplified or expanded to better achieve the desired ordering. Let $\mathcal{D}$ be two-class truthed data. Thus, $\mathcal{D}$ can be partitioned into two disjoint subsets $\mathcal{D}^+$ and $\mathcal{D}^-$ where $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$ and $\mathcal{D}^+ \cap \mathcal{D}^- = \emptyset$. A multilayer perceptron F trained on the data $\mathcal{D}$ yields a hyperplane arrangement A. From this arrangement, one forms the associated chamber set Cham(A). Some of the chambers are populated with the "+" class data vectors; other chambers are populated with the "-" class data vectors. Some chambers—and, often, most—are not populated at all. If the MLP has done an excellent job of separating the data, then no chamber exists that contains both "-" class and "+" class vectors.

Based upon the truthed data $\mathcal{D}$, we can define three subsets of the chamber set Cham(A) as

$$\begin{aligned}
\text{Cham(A)}^+ &= \{C \in \text{Cham(A)} : x \in C \text{ for some } x \in \mathcal{D}^+\} \\
\text{Cham(A)}^- &= \{C \in \text{Cham(A)} : x \in C \text{ for some } x \in \mathcal{D}^-\} \\
\text{Cham(A)}^u &= \text{Cham(A)} - (\text{Cham(A)}^+ \cap \text{Cham(A)}^-).
\end{aligned}$$

Since $\mathcal{D}^+ \cap \mathcal{D}^- = \emptyset$, one hopes that the MLP has done an excellent job of separating the data into separates chambers, such that $\text{Cham(A)}^+ \cap \text{Cham(A)}^- = \emptyset$. When the MLP has not separated the data then $\text{Cham(A)}^+ \cap \text{Cham(A)}^- \neq \emptyset$. Often, the MLP creates more chambers than needed and several chambers have no data in them; thus, in general, we have that the set of unpopulated chambers $\text{Cham(A)}^u \neq \emptyset$ or

$$\text{Cham(A)}^+ \cup \text{Cham(A)}^- \neq \text{Cham(A)}.$$

6.4.2 Ordering chambers by relative veracity. Consider a single-hidden-layer multilayer perceptron constructed using hard-limiter sigmoids in the hidden layer nodes and tansig sigmoids in the output layer. Thus, given a chamber C and a point $x \in \text{int}(C)$, the MLP output yields a veracity $\mathfrak{v}(x) \in [-1, 1]$ and, in fact, this veracity is the same value for all $x \in C$. Therefore, we consider the entire chamber C and define the veracity of the chamber $\mathfrak{v}(C) = \mathfrak{v}(x)$. Since the space $\mathbb{R}^d$ is partitioned into the disjoint subsets of Cham(A) by the MLP, then we need only to consider each chamber separately.

6.4.3 Ordering chambers by relative experience. We seek to identify each chamber $C \in \text{Cham(A)}$ with a label representing experience. Define the experience mapping $\mathfrak{e} : \text{Cham(A)} \to \{t, f, i, e\}$. Populated chambers shall be identified as containing specific experience. Let the true label t denote specific experience with "+"class, and let false label f denote specific experience with "-" class data. Unpopulated chambers shall be identified as either interpolations or extrapolations of training data. The interpolation label i implies marginal experience in training data, and the extrapolation label e implies insignificant-to-no experience with class data.

Our experience mapping $\mathfrak{e}$ utilizes a distance metric between chambers. For a simple metric, we determine whether or not a chamber intersects with any populated chamber. Let us construct a mapping that convenes this notion. Define the distance mapping σ on

chambers in Cham(A) to be

$$\sigma(C', C'') = \begin{cases} 0 & \text{if} \quad C' = C'' \\ 1 & \text{if} \quad C' \cap C'' \neq \emptyset \\ 2 & \text{if} \quad C' \cap C'' = \emptyset \end{cases}$$

for every $C', C'' \in \text{Cham(A)}$. Recall, chambers are closed sets so if $C' \cap C'' \neq \emptyset$ then $C' \cap C''$ is contained in one of the hyperplanes in the arrangement A.

We have the result that the above mapping is a metric.

Theorem 8 *The mapping σ is a metric on* Cham(A)*, and* $(\text{Cham(A)}, \sigma)$ *is a metric space.*

Proof. Note that $\sigma(C', C'')$ is defined for every possible choice of $C', C'' \in \text{Cham(A)}$.

1. Clearly, $\sigma(C', C'') \geq 0$ for all $C', C'' \in \text{Cham(A)}$.
2. Clearly, $\sigma(C', C'') = \sigma(C'', C')$ for all $C', C'' \in \text{Cham}(A)$.
3. Given $C', C'', C''' \in \text{Cham(A)}$ then, enumerating the different cases, we see

$$\sigma(C', C''') \leq \sigma(C', C'') + \sigma(C'', C''').$$

4. If $C' \neq C''$ then $\sigma(C', C'') = 1$ or 2, thus $\sigma(C', C'') \neq 0$ so σ is positive definite.

Therefore, σ is a metric defined on Cham(A). It follows that $(\text{Cham(A)}, \sigma)$ is a metric space. ■

Now, suppose $\mathbb{C}$ is a subset of chambers in Cham(A). We define the distance from a chamber C to a set of chambers $\mathbb{C}$ by

$$\text{dist}(C, \mathbb{C}) = \min\{\sigma(C, C') : C' \in \mathbb{C}\}.$$

Observe that if $\text{dist}(C, \mathbb{C}) = 0$ then chamber $C \in \mathbb{C}$. We may choose special subsets for $\mathbb{C}$ among our populated chambers. Let us assume that $\text{Cham(A)}^+ \cap \text{Cham(A)}^- = \emptyset$. Then,

we can produce a mapping based upon the data set $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$

$$\mathfrak{e}(C) = \begin{cases} t & \text{if} \quad \text{dist}(C, \text{Cham(A)}^+) = 0 \\ f & \text{if} \quad \text{dist}(C, \text{Cham(A)}^-) = 0 \\ i & \text{if} \quad \text{dist}(C, \text{Cham(A)}^u) = 1 \\ e & \text{if} \quad \text{dist}(C, \text{Cham(A)}^u) = 2. \end{cases}$$

Thus, populated chambers are labeled either t or f, and unpopulated chambers are labeled either i or e. If we allow the case that $\text{Cham(A)}^+ \cap \text{Cham(A)}^- \neq \emptyset$, then chambers containing both true and false data are labeled as i, or uncertain interpolations.

6.4.4 Mapping veracity versus expertise for the MLP. We can associate each chamber $C \in \text{Cham(A)}$ with a 2-tuple $(\mathfrak{v}(C), \mathfrak{e}(C))$. To visualize how the MLP will map every chamber (and, thus, every vector in $\mathbb{R}^d$) we consider the set

$$\{(\mathfrak{v}(C), \mathfrak{e}(C)) : C \in \text{Cham(A)}\}.$$

Since the chamber set Cham(A) is finite, we can order the chambers using their veracity values to get the ordered set $(\text{Cham(A)}, \preccurlyeq) = \{C_1, C_2, ..., C_q\}$ where $q = \text{card(Cham(A))}$ and

$$\mathfrak{v}(C_1) \leq \mathfrak{v}(C_2) \leq \cdots \leq \mathfrak{v}(C_q).$$

Thus, we have defined the relation $\preccurlyeq$ on Cham(A) such that chamber $C_i \in \text{Cham(A)}$ "precedes" chamber $C_j \in \text{Cham(A)}$ and write $C \preccurlyeq B$ if and only if $\mathfrak{v}(C_i) \leq \mathfrak{v}(C_j)$.

Now consider the set of 3-tuples

$$\{(i, \mathfrak{v}(C_i), \mathfrak{e}(C_i)) : i = 1, 2, ..., q\}.$$

If the MLP F isolates all the data perfectly, then the graph of this set yields plots like Figure 66. However, if the MLP F merely separates training data perfectly but arbitrarily orders extrapolated data, then the graph of the 3-tuple set yields plots like Figure 67.

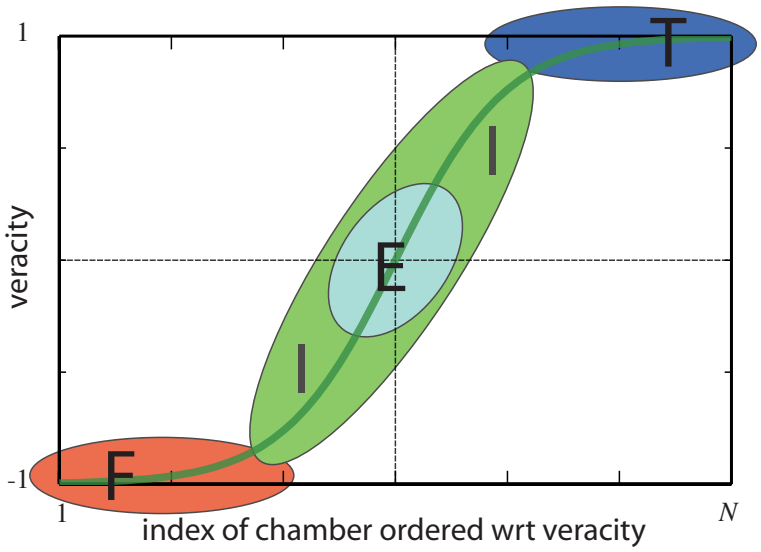


Figure 66. An appropriate expression of expertise: The desired clustering of experience based on the ordered veracity of a multilayer perceptron. Experience is represented in terms of 4-value logic. Specific experience should cluster in the appropriate asymptotic regions such where $|\mathfrak{v}(x)| \to 1$, while extrapolations should cluster tightly where $\mathfrak{v}(x) = 0$.

6.4.5 Desired and expected chamber orderings of the MLP. We assert that a good generalization isolates data, allowing extrapolations and uncertain interpolations to be easily separated from class data in a feature set. Unfortunately, as a classifier, the multilayer perceptron merely separates data from disparate classes and does not isolate the classes. To provide good separation, it is desirable to have chambers containing false data points to score veracities near -1 and to have chambers containing true points to score veracities near 1. To provide good isolation of classes, it is further desirable for a chambers near populated chambers to score veracities close to 0 (that is, along the non-linear transition) and for chambers far from populated chambers to score veracities of 0.

Unfortunately, we have found that the ordering of the chamber set constructed by a multilayer perceptron trained by back propagation rarely achieves the ordering shown in Figure 66. Instead, the expected veracity is more closely reflexed in Figure 67—a result that means that MLP mapping contains arrogant classifications. A multilayer perceptron that maps in this way gives interpolated and extrapolated regions stronger veracity scores than regions containing true and false training data.

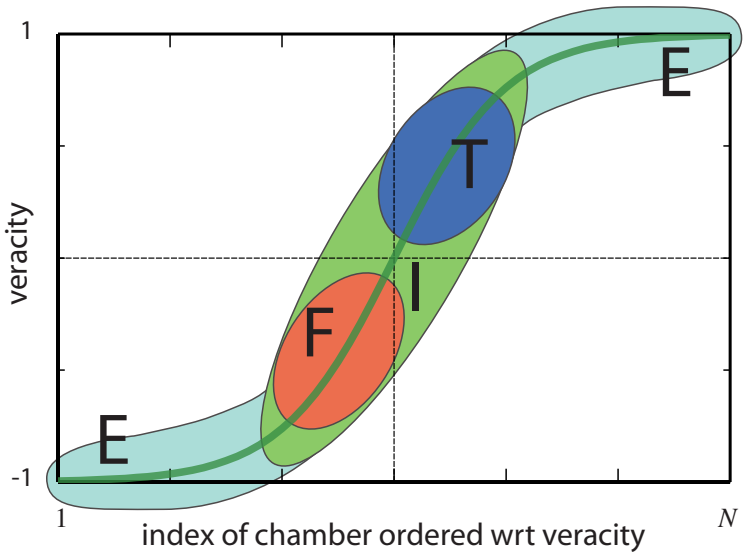


Figure 67. Expressions of arrogance in the multilayer perceptron: Our expected clustering of experience based on the ordered veracity of a multilayer perceptron trained by backpropagation.

The expectation of veracity shown in Figure 67 assumes that the chambers assigned as true, false or interpolated are bounded, i.e., chambers of finite volume. This is a reasonable assumption given $d \ll N$ where d is the dimension of the feature set and N is the number of first-hidden-layer nodes.

6.4.6 Resolving confusion in a Fisher iris solution. For a specific illustration of an arrogant classifier, let us investigate an MLP solution for the Fisher iris classification problem. The Fisher iris dataset [31] represents 150 observations of iris flowers, 50 each from 1 of 3 species of iris: Iris Setosa, Versicolor, and Virginica. Each observation is characterized by a vector capturing 4 numerical attributes: sepal length, sepal width, petal length, and petal width. We have selected to investigate a multilayer perceptron with 14 hidden layer nodes and 3 output nodes trained via back propagation to separate the first class of iris from the other two classes. This particular MLP was chosen because we had the necessary elements to do a complete analysis: the MLP's weights and biases, the network's training set and, additionally, its evaluation set). Additionally, the MLP is an interesting solution on two accounts: (1) The MLP is complex (14 hidden nodes is excessive to solve the Fisher

iris problem) and, yet, it has no significant issues with memorization; and (2), though the MLP does not appear to have memorized its data, the solution still represents a poor generalization because the MLP generates a significant number of arrogant classifications.

For the MLP, we were given the following: the weights and biases of a single-hidden layer perceptron, two datasets (one used to train the MLP and the other used to evaluate the MLP's performance), and the error curve generated in the training of the MLP. We were not privy to the training of this network, but we were supplied with the final parameter set of the design—the hidden layer weights $[\overline{W}_h : \overline{b}_h]$ and the output weights $[\overline{W}_o : \overline{b}_o]^T = [u_1, u_2, u_3]$.

$$[\overline{W}_h : \overline{b}_h] = \begin{bmatrix} 0.2599 & 0.3348 & -0.3876 & -0.2779 & 0.4585 \\ -0.3184 & 0.1027 & 0.3266 & -0.3127 & -0.1405 \\ -0.2844 & -0.0509 & -0.0714 & 0.3860 & 0.0187 \\ 0.1902 & 0.1249 & -0.3105 & -0.3983 & -0.0660 \\ -0.1328 & 0.6630 & 0.0140 & 0.3430 & 0.3249 \\ -0.2037 & 0.1119 & 0.0675 & -0.0988 & 0.1614 \\ 0.3562 & -0.2434 & -0.3651 & -0.3105 & 0.1878 \\ -0.4330 & 0.1389 & -0.1011 & -0.3456 & 0.5038 \\ -0.0740 & -0.2961 & -0.1801 & 0.2190 & -0.0622 \\ 0.2210 & -0.1963 & -0.3508 & -0.0272 & 0.0245 \\ -0.3331 & -0.4145 & 0.1865 & 0.1978 & -0.3176 \\ 0.1955 & -0.5565 & -0.2773 & -0.6217 & -0.5234 \\ 0.6625 & 0.4879 & -0.6841 & -0.1655 & 0.0317 \\ -0.1643 & 0.1604 & 0.2078 & 0.1187 & -0.0467 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.8781 & 17.3599 & 3.2325 \\ 3.8707 & -20.5670 & -12.0076 \\ 0.9027 & -15.2658 & -12.6445 \\ 6.5404 & -40.9967 & -7.3975 \\ 1.1803 & -21.1980 & 9.7689 \\ 2.0187 & -17.7602 & -4.9986 \\ 0.7589 & -4.0878 & -0.9049 \\ 2.4011 & 40.4554 & -20.8312 \\ 0.5458 & 3.1439 & -11.4082 \\ 1.0027 & -4.8418 & -6.2945 \\ 0.7954 & 9.6175 & -18.5234 \\ 4.9099 & -8.9478 & -0.7643 \\ 3.5889 & 19.2883 & -39.8599 \\ -0.6176 & 3.6268 & 3.6068 \\ -4.6405 & -16.0082 & -25.2557 \end{bmatrix}$$

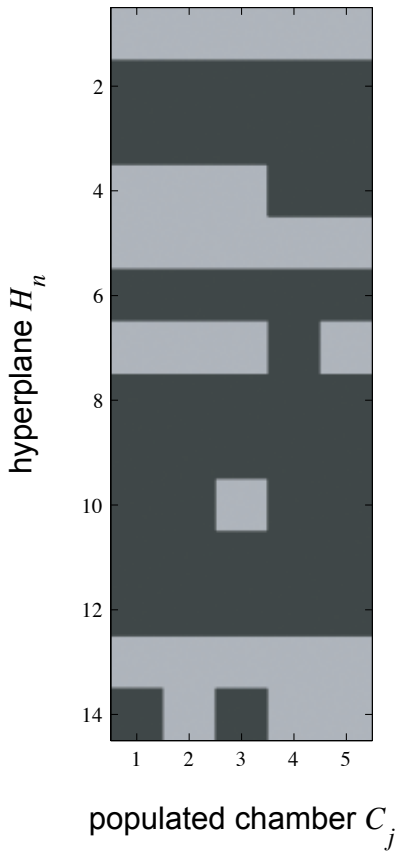


Figure 68. The populated chambers for a multilayer perceptron trained by back propagation to solve the Fisher iris problem. The first four chambers were the chambers populated by both the training set and test set. The fifth chamber was populated by a single datum from the test set.

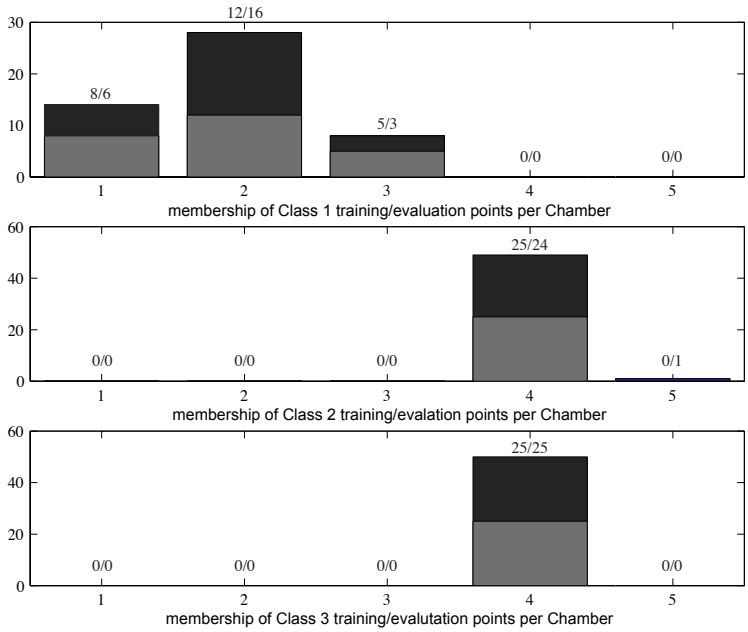


Figure 69. The membership of the populated chambers by class. The bar charts represent the sum of the membership by training set (lower) and test set (upper) respectively.

First, let us investigate whether or not the neural network is memorizing. It is simple to show using principle component analysis [38] that the problem of separating the first Fisher iris class from the other two classes is a linear problem. Thus, a successful solution to the problem requires only a single perceptron. Since the MLP under investigation uses a 14-hidden-node architecture, rules of thumb [68] tell us that the solution is unnecessarily complex and the MLP has likely memorized the training data. However, an analysis of the populated chambers and unpopulated chambers of interest belies this intuition. Instead of constructing an error matrix, we use our method [63] of resolving populated chambers. Figures 68 and 69 shows the results: out of a possible 1471 chambers[14], only 4 chambers are populated with training data and 5 chambers (the 4 populated chambers and one additional chamber) contain data from the evaluation set—i.e., labeled data available at design that is not used to train or select a model but is used to estimate the error in the solution. From Figure 69, we see little evidence of memorization. Only a single evaluation exemplar fell

[14]Using Equation 25 with $d = 4$ and $N = 14$ and assuming the hyperplane arrangement is in general position, then the cardinality of the arrangement's chamber set is q=1471.

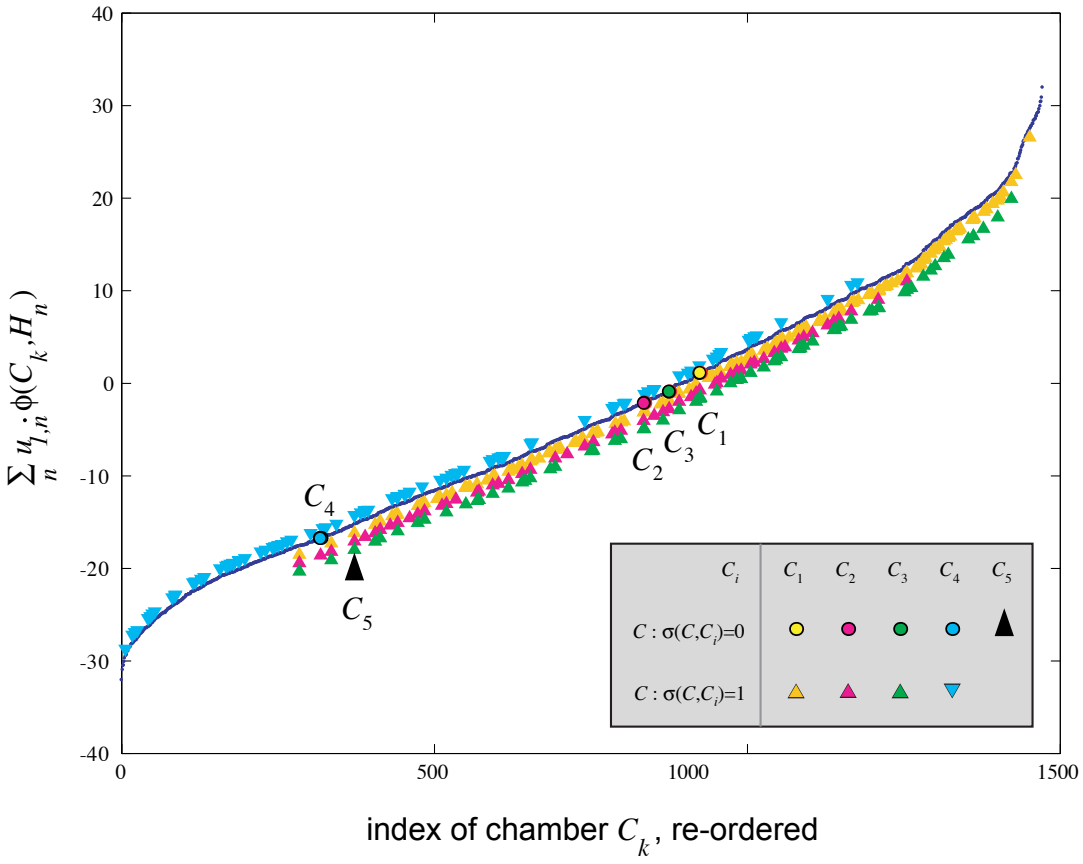


Figure 70. Weighted ordering of chambers in the Fisher iris feature domain via output node 1 of MLP F.

outside of the 4 populated chambers; and, though it is arguably excessive to separate Class 1 data into 3 separate chambers, the ratio of training data to evaluation data membership in each of these chambers is relatively even.

There is very little evidence for memorization in this particular MLP solution; but, now, let us consider arrogance. Recall, in Section 5.1.3 we asserted that a good generalization may be illustrated by ordering the feature set of a classifier in terms of its veracity scores and, then, cross referencing this ordering to the classifier's experience scores. Further, we defined measures to specify the veracity and experience scores over the chambers of an MLP given the network's weights and biases and the data used to train the network. Now, we have applied these measure to the 3-output-node MLP solution above and generated

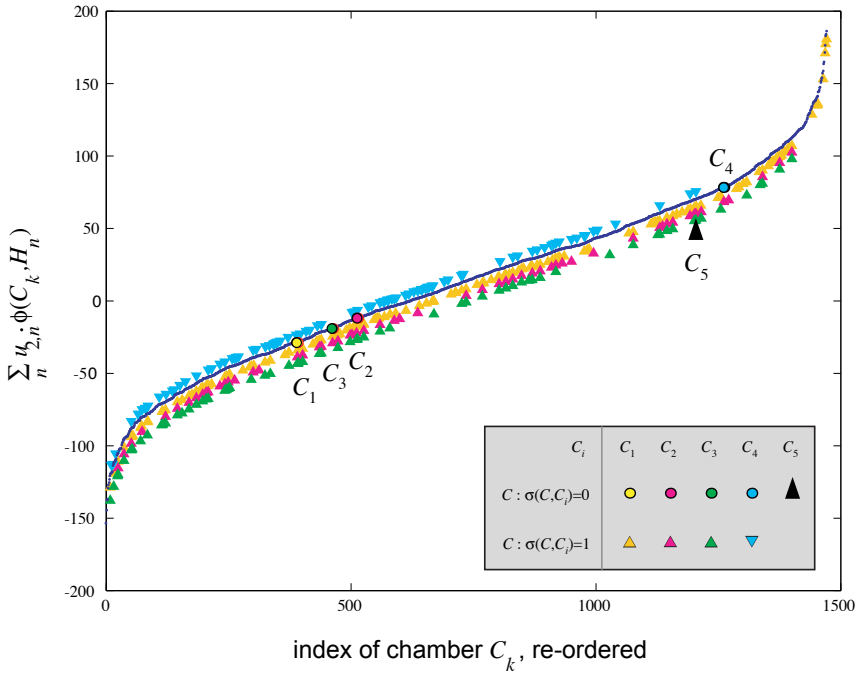


Figure 71. Weighted ordering of chambers in the Fisher iris feature domain via output node 2 of MLP F.

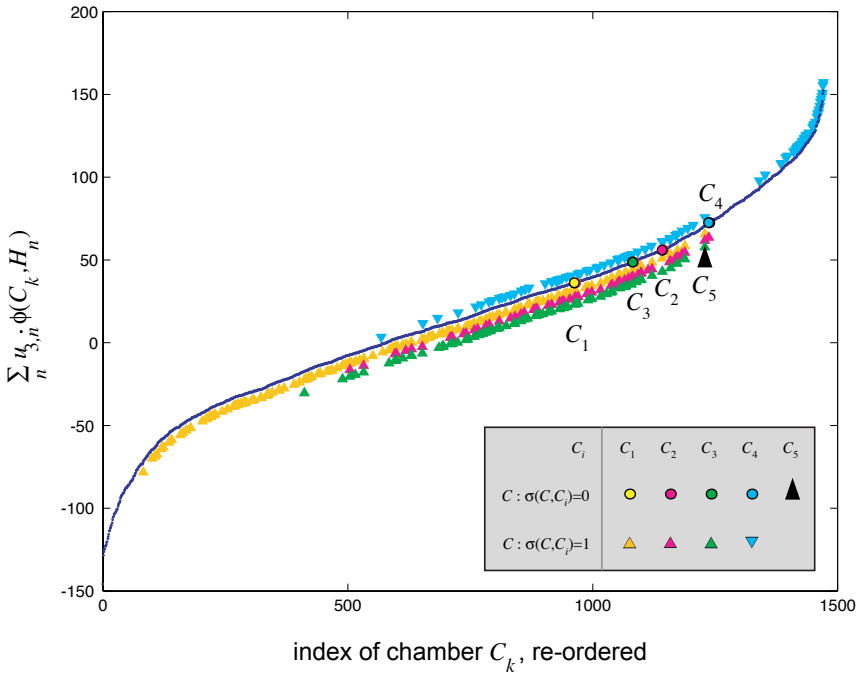


Figure 72. Weighted ordering of chambers in the Fisher iris feature domain via output node 3 of MLP F.

Figures 70, 71, and 72 to show the domain ordering specified by each output node. We used unnormalized veracity scores to simplify the calculation of the plots as they are merely scaled and shifted versions of the normalized graphs. In each plot, a point on the S-curve specifies a chamber. Specially noted chambers include populated chambers (denoted as circles labeled C_1, C_2, C_3, C_4), the unpopulated chamber of interest (the larger triangle labeled as C_5), and unpopulated chambers labeled as interpolations $\mathfrak{e}(C) = i$ (smaller triangles). The remaining points are unpopulated chambers labeled as extrapolations, i.e., designated by $\mathfrak{e}(C) = e$. Note in each graph, the veracity scores of populated chambers are not ordered at the extremes of the S-curve but instead along the linear portion of the curve. Extrapolated chambers appear arbitrarily along the entire length of the curve. These graphs each reflect a significant number of arrogant classifications with Figure 72 representing the worst offender, output node 3.

We have demonstrated a strong example of arrogance here. In further empirical experiments, we have found that the ordering of the chamber set implemented by a multilayer perceptron trained by back propagation rarely achieves the ordering of a good generalization shown in Figure 66. Instead, the ordered veracity-experience pairs more closely reflex those illustrated in Figure 67—a result that means that the classifier generates arrogant classifications. A multilayer perceptron that maps in this way gives interpolated and extrapolated regions stronger veracity scores than regions containing true and false training data.

When applied to the multilayer perceptron, distribution of the ordered veracity-experience pairs shown in Figure 67 assume that (1) the hyperplane arrangement A implemented by the MLP does not define a convex hull, (2) the complement of the hyperplane arrangement A also does not define a convex hull, and (2) the chambers that are assigned as true, false and interpolated are bounded, i.e., chambers of finite volume. These are reasonable assumptions given $d \ll N$ where d is the dimension of the feature set and N is the number of first-hidden-layer nodes. Note we are also relying on the fact that backpropagation tends to iterate to an MLP solution where the hyperplane arrangement of the first hidden layer is not symmetrically balanced as in a regular polygon.

6.4.7 Improved isolation in a Fisher iris solution. We can use the MLP above for guidance in preparing networks with better performance and architectures that more appropriately represent the experience of the training set. We have included the results for two new sets of input layer weights. Both sets of weights nearly implement the training set, i.e., separate all three iris classes. In both solutions, we have allowed one chamber to include the data of iris class B and C that appears to overlap. The first revised solution represents a simplified MLP that implements a solution for the three classes using only modified biases from the first MLP and a separation architecture. The second solution augments the simplified MLP with nodes that ensure the isolation of training data. Thus, we generated the new parameter sets following the 3-fold procedure for training and converting multilayer perceptrons from a separation architecture to an isolation architecture outlined in Section 6.1.2 and first demonstrated with a 3-hidden-node MLP at the end of Section 6.3.1.3.

1. Start with a complex multilayer perceptron, previously trained.
2. Prune the MLP to only those nodes required to implement the separation of class data into distinct chambers.
3. Augment the simplified MLP to ensure populated chambers are reasonably finite.

The MLP that implements simplified separation has the following parameter set.

$$[\overline{W}_h : \overline{b}_h] = \begin{bmatrix} -0.4330 & 0.1389 & -0.1011 & -0.3456 & 2.5000 \\ 0.2210 & -0.1963 & -0.3508 & -0.0272 & 0.4236 \\ 0.2599 & 0.3348 & -0.3876 & -0.2779 & -0.2219 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 0.1806 & -0.3106 & -0.0180 \\ 0.1212 & -0.3212 & -0.1212 \\ 0.0311 & 0.3110 & -0.3110 \\ 0.0398 & 0.3975 & -0.3975 \\ 0 & -0.7500 & 0 \end{bmatrix}$$

The chamber set implemented by the MLP above is shown in Figure 73. Figure 73(a) lists the populated chambers in terms of their signed diagonal. After studying the memberships in Figure 74, we label these 4 chambers, respectively, Class A, Class B, Uncertain Interpolation (between Class B and C), and Class C. The output weights and biases were chosen to separate the populated chambers in terms of class as shown in Figure 75. Note

that chamber C_3—the chamber populated with Class B and C training data—marginally survives thresholding for the orderings favoring Iris Class B and C. This selection of biases was done purposely per confusion set procedures discussed in Section 4.2.7.3.

Given that we have $N = 4$ nodes in a 4-dimensional space, the chamber set has a cardinality of $2^4 = 16$. Therefore, we generate OVER curves in Figure 76 much as we did for the XOR problem. Note though separation has been achieved, we see that, for these sets of output weights, isolation has not. There is significant mixing of unpopulated chambers among the populated chambers for the orderings of Class B and Class C that due to adjacencies cannot be overcome in a single-hidden layer architecture. Therefore, we need to add an additional layer of nodes to implement a true isolation architecture.

Isolation can be improved in a single hidden layer MLP by augmenting the above MLP with nodes that serve to wrap the entire training set. We have constructed an augmented MLP with the following parameter set.

$$[\overline{W}_h : \overline{b}_h] = \begin{bmatrix} -0.4330 & 0.1389 & -0.1011 & -0.3456 & 2.3000 \\ 0.2210 & -0.1963 & -0.3508 & -0.0272 & 0.4236 \\ 0.1902 & 0.1249 & -0.3105 & -0.3983 & 0.6014 \\ 0.1806 & 0.1212 & -0.3110 & -0.3975 & 0.8445 \\ -1.0000 & 0.0838 & 0.0305 & 0.0378 & 9.0000 \\ 0.0932 & -1.0000 & 0.0190 & 0.0860 & 6.0000 \\ 0.0466 & 0.0681 & -1.0000 & 0.0854 & 8.0000 \\ 0.0419 & 0.0379 & 0.0682 & -1.0000 & 4.0000 \\ 0.0525 & 1.0000 & 0.0542 & 0.0900 & 1.0000 \\ 0.0203 & 0.0709 & 1.0000 & 0.0822 & 0.3529 \\ 2.0672 & 0.2429 & 3.0698 & 1.2000 & 1.1010 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 0.1806 & -0.3106 & -0.0180 \\ 0.1212 & -0.3212 & -0.1212 \\ 0.0311 & 0.3110 & -0.3110 \\ 0.0398 & 0.3975 & -0.3975 \\ 0.5000 & 0.5000 & 0.5350 \\ 0.5050 & 0.5050 & 0.5300 \\ 0.5100 & 0.5100 & 0.5250 \\ 0.5150 & 0.5150 & 0.5200 \\ 0.5250 & 0.5250 & 0.5100 \\ 0.5300 & 0.5300 & 0.5050 \\ 0.5350 & 0.5350 & 0.5000 \\ -9.0000 & -10.0000 & -9.5000 \end{bmatrix}$$

Figure 77 represents the signed diagonals of the populated chambers for the MLP with an isolation architecture. Note, from Figure 73(a), the number of populated chambers has not changed; also, the membership of the chambers as listed remains the same as Figure 74. What has changed is that the populated chambers are now bounded, or finite, chambers; and the biases had to be adjusted to maintain appropriate thresholding per Figure 78.

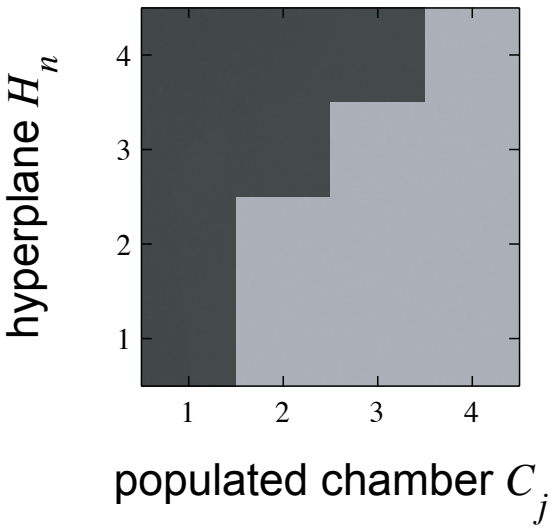
In the OVER curves illustrated in Figure 79, 80 and 81, we were able to use a simple interpretation of what is an interpolated chamber among the unpopulated chambers. We base this interpretation on the nodes that wrap the data set, defining interpolated chambers as those unpopulated chambers contained within the intersection of the augmented nodes. In Figures 82, 83 and 84, we tested a second set ofoutput weights

$$[\overline{W}_o : \bar{b}_o]^T = \begin{bmatrix} 0.1806 & -0.3106 & -0.0180 \\ 0.1212 & -0.3212 & -0.1212 \\ 0.0311 & 0.3110 & -0.3110 \\ 0.0398 & 0.3975 & -0.3975 \\ 1.2500 & 1.2500 & 1.3375 \\ 1.2625 & 1.2625 & 1.3250 \\ 1.2750 & 1.2750 & 1.3125 \\ 1.2875 & 1.2875 & 1.3000 \\ 1.3125 & 1.3125 & 1.2750 \\ 1.3250 & 1.3250 & 1.2625 \\ 1.3375 & 1.3375 & 1.2500 \\ -9.0000 & -10.0000 & -9.5000 \end{bmatrix}.$$

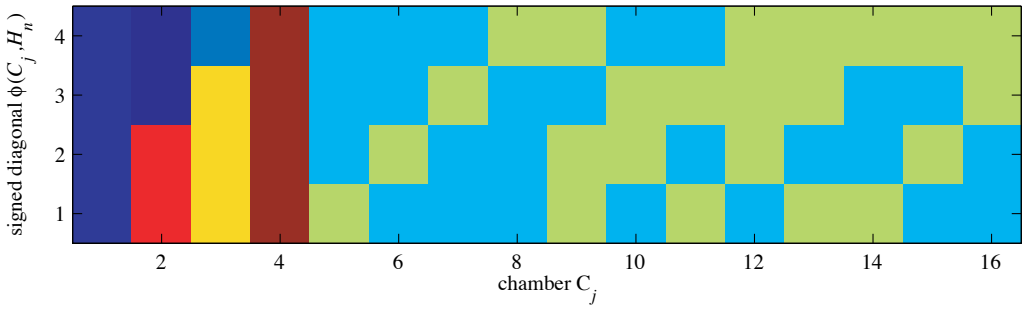
From this alternate set of output weights, we realized improved separation—though not complete isolation—of extrapolated chambers and interpolated chambers.

6.4.8 Benefits of the OVER characterization. Benefits of the ordered veracity-experience characterization over other iterative and stochastic methods include (1) model selection, (2) data presentation, and (3) domain coverage. In case (1), we feel strongly that a generalization evaluation technique should determine the appropriateness of a selected model. However, iterative methods such as cross validation and bootstrapping require that the evaluation of generalization take place over multiple training runs[15] prior to model

[15]In each run, the training dataset, or augmented dataset, is split into two: one subset is used to train a solution iteratively, and the second (typically smaller) subset is used to calculate the classification error for each training epoch. Training of the current model continues until the classification error stabilizes



(a)



(b)

Figure 73. The chamber sets for a simplified MLP solution to the Fisher iris problem, (a) the set of populated chambers and (b) the full set of chambers presented in the order of discovery. The first four chambers listed in (b) are coded respectively for Class A, Class B, Uncertain Interpolation between Class B and C, and Class C.

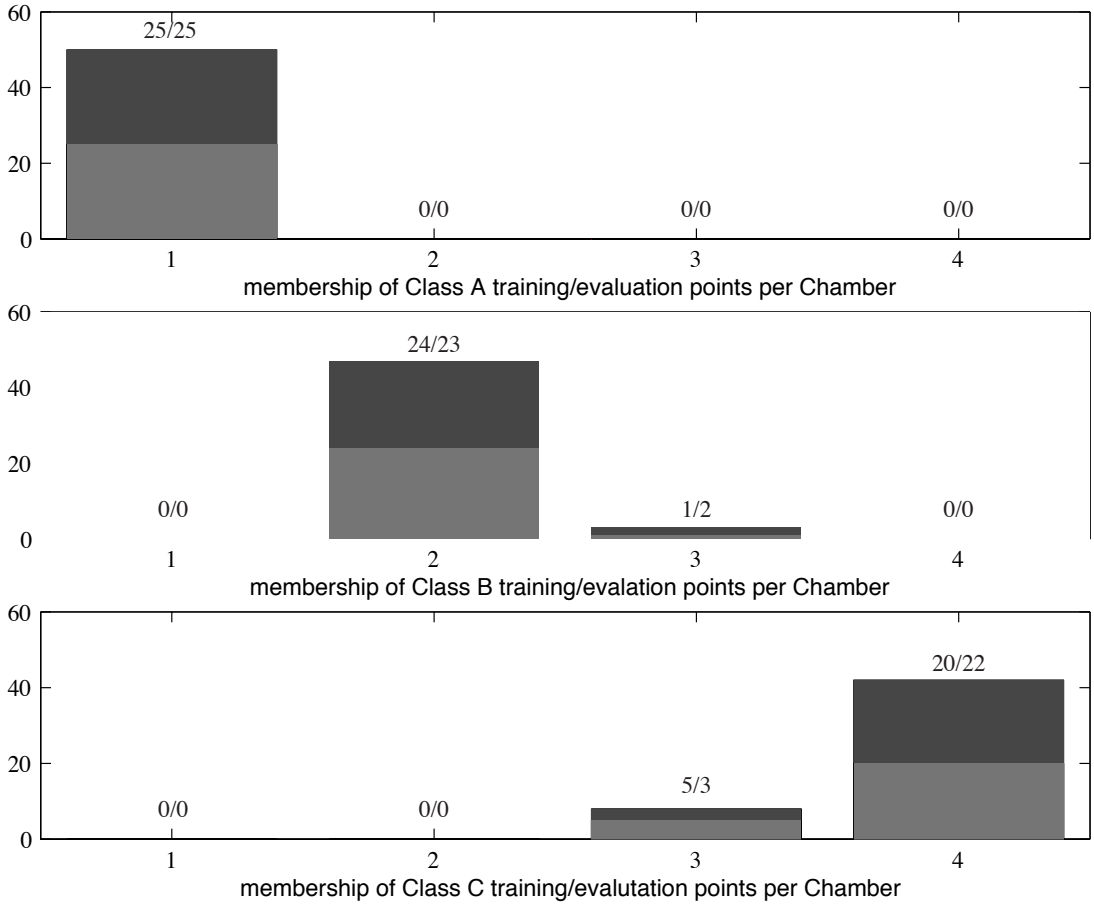


Figure 74. The membership of the populated chambers by class. The bar charts represent what portion of the membership of the training set (lower) and evaluation set (upper) occupy each populated chambers, respectively.

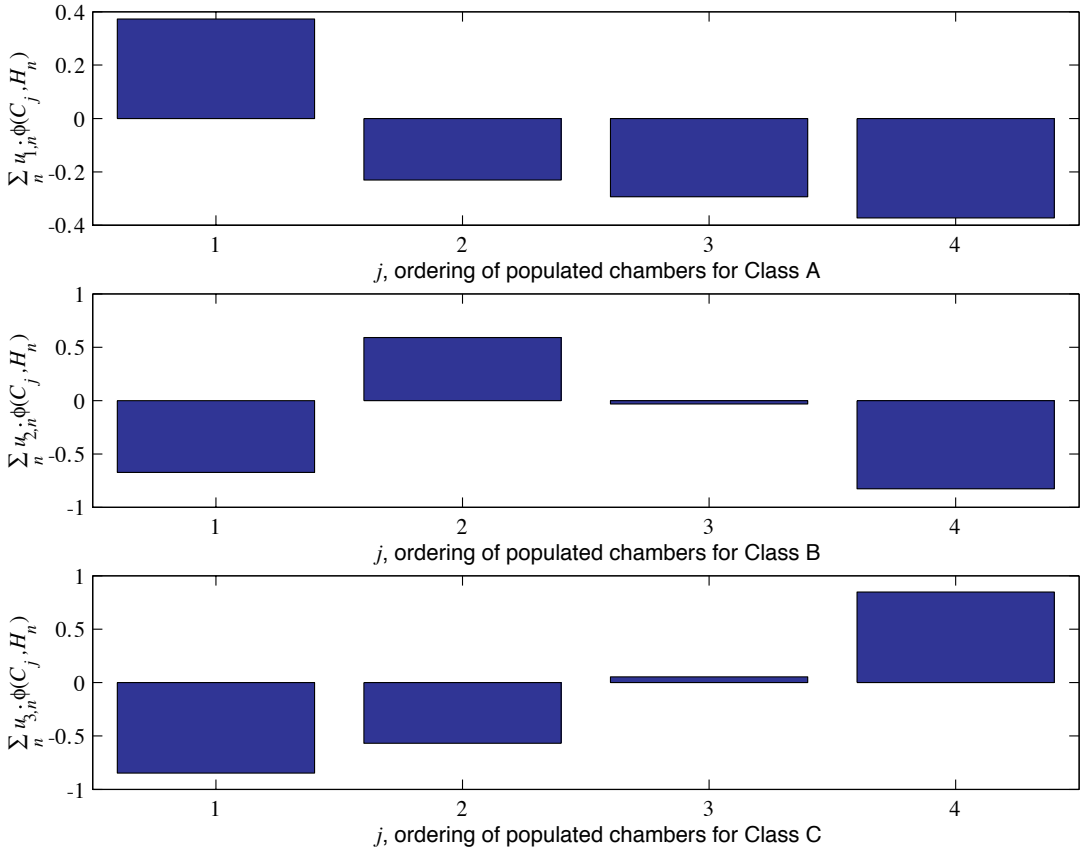
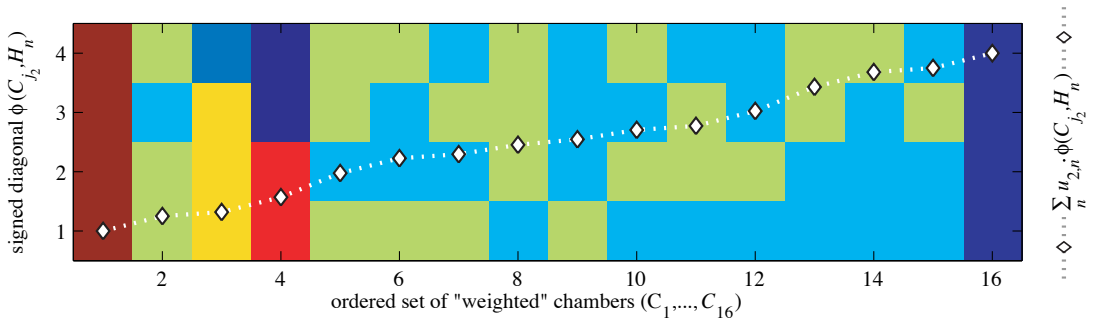
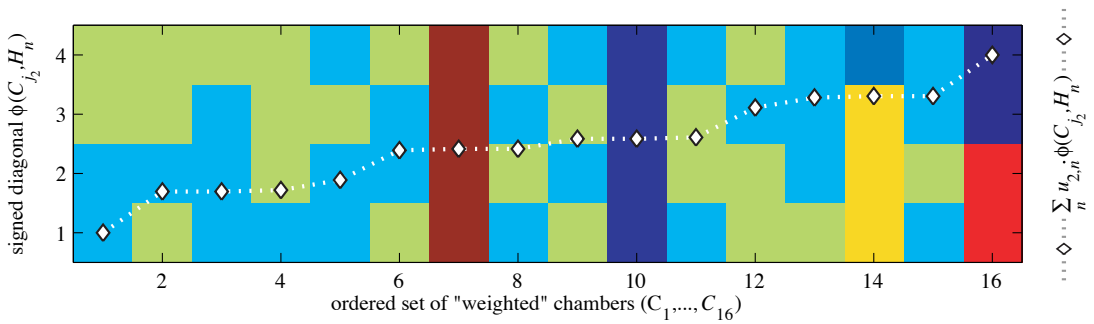


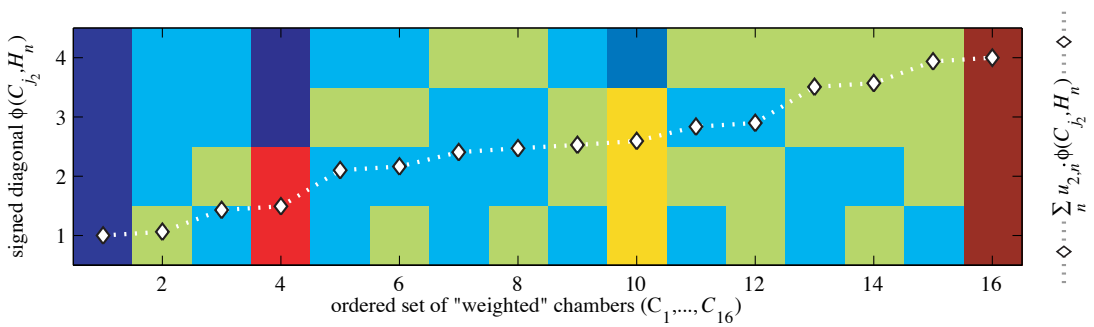
Figure 75. Three weighted orderings of the populated chambers favoring Class A, B, and C, respectively.



(a)



(b)



(c)

Figure 76. The 3 sets of weighted signed diagonals for the simplified MLP, favoring Class A, Class B, and Class C, respectively.

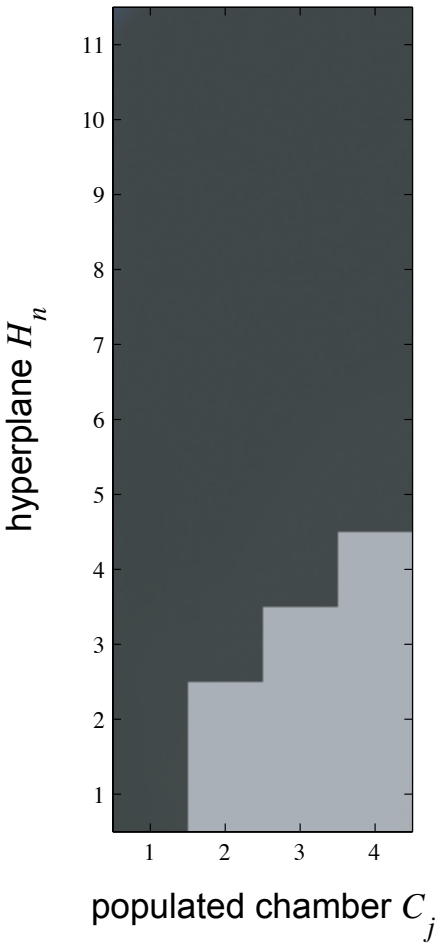


Figure 77. The populated chambers for the simplified MLP solution augmented to create an isolation architecture.

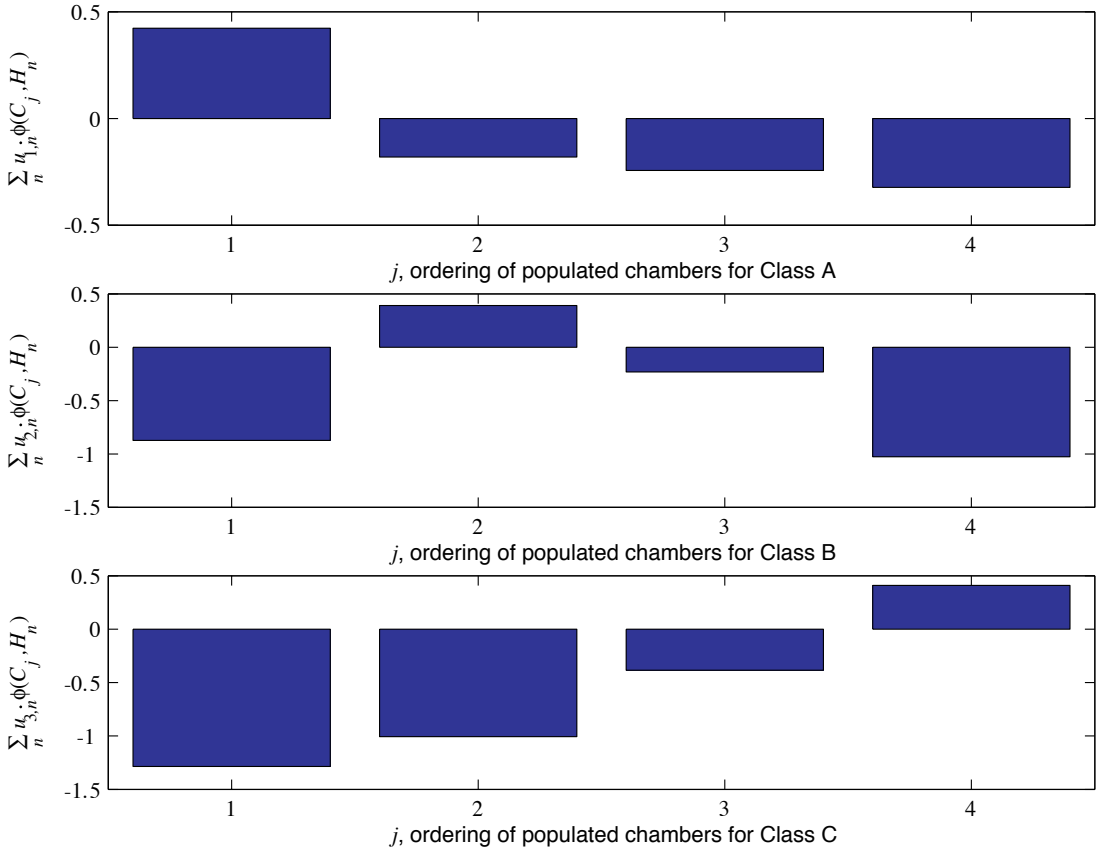
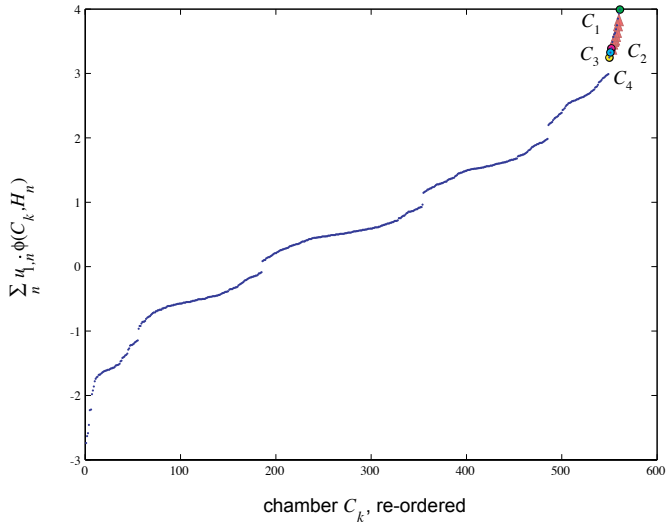
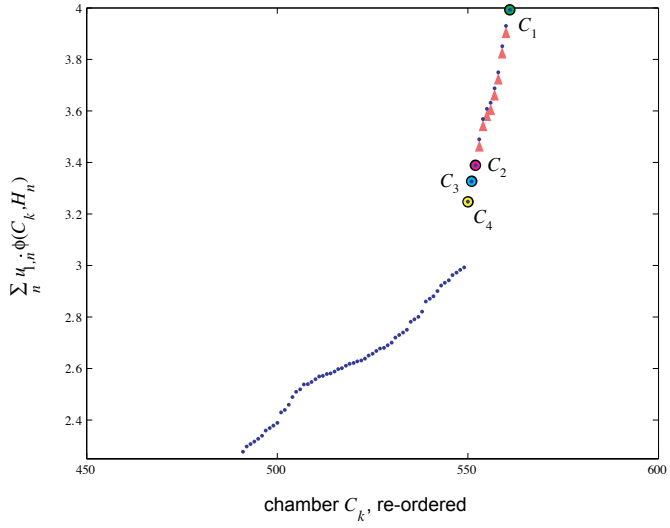


Figure 78. Three re-weighted orderings of the finite populated chambers favoring Class A, B, and C respectively.

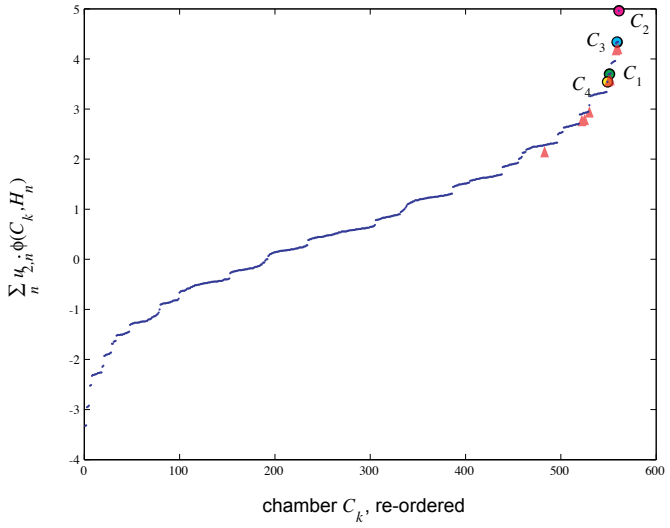


(a)

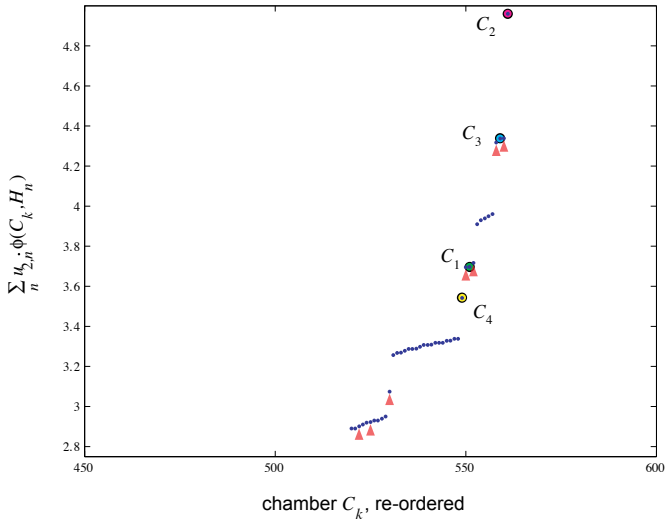


(b)

Figure 79. Weighted ordering of chambers in the Fisher iris feature domain via the output node for class A of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering.

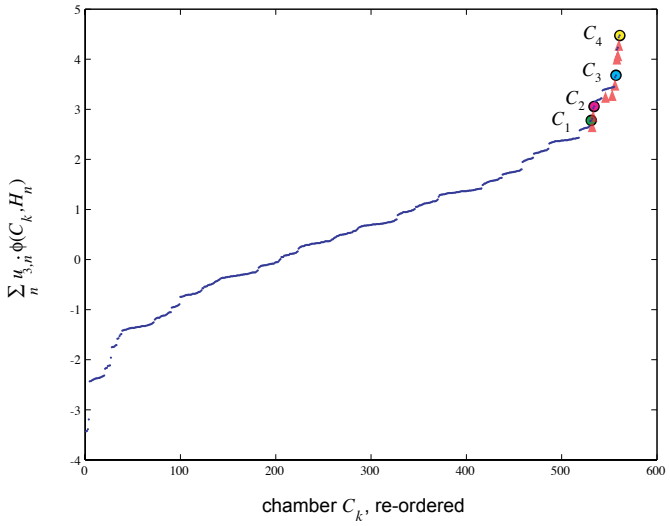


(a)

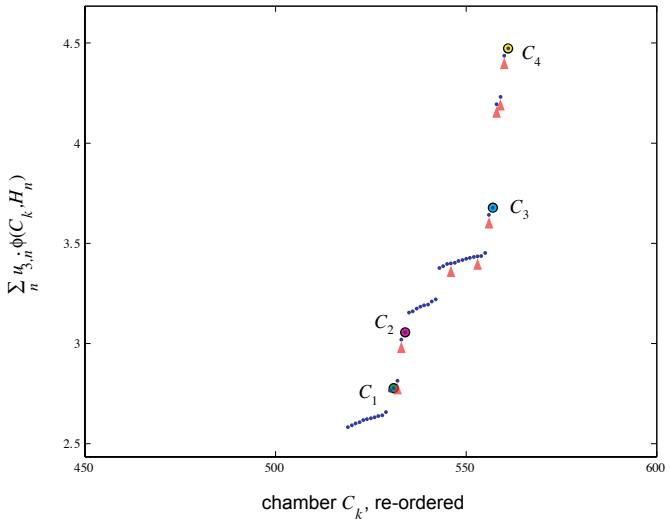


(b)

Figure 80. Weighted ordering of chambers in the Fisher iris feature domain via the output node for class B of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering.

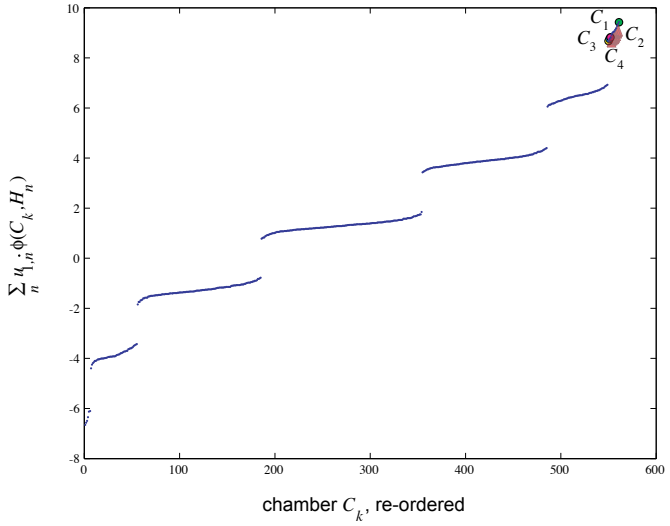


(a)

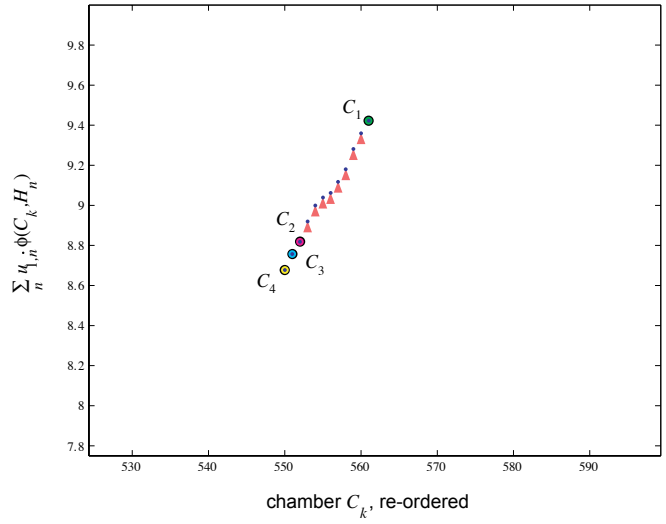


(b)

Figure 81. Weighted ordering of chambers in the Fisher iris feature domain via the output node for class C of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering.

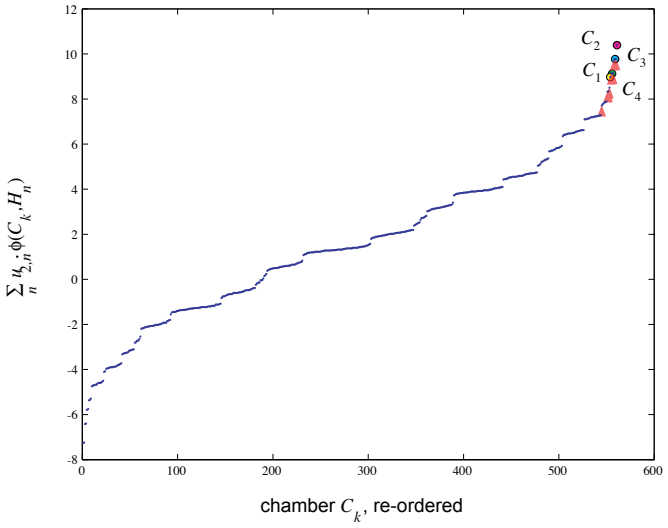


(a)

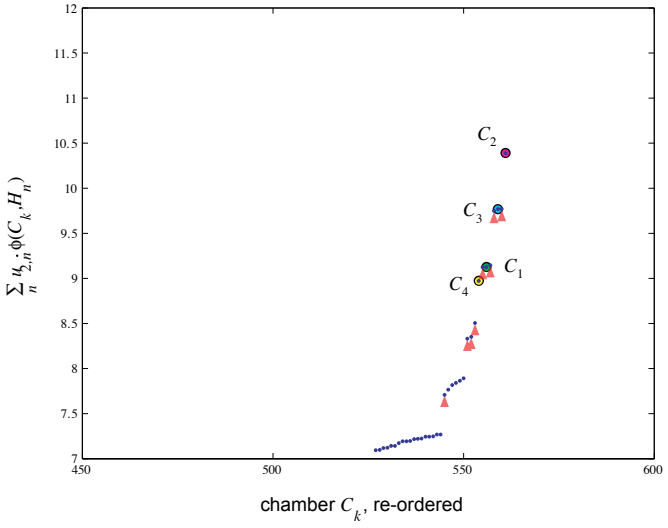


(b)

Figure 82. Improved ordering of chambers in the Fisher iris feature domain via the output node for class A of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering. The improvement is achieved by increasing the output weights evenly on the augmented nodes.

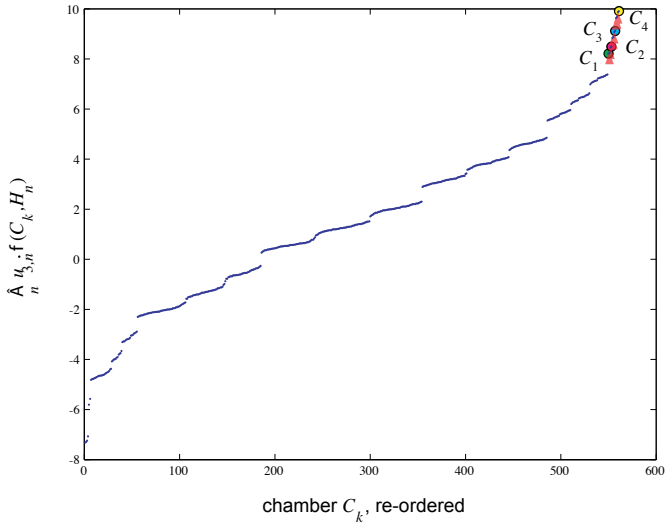


(a)

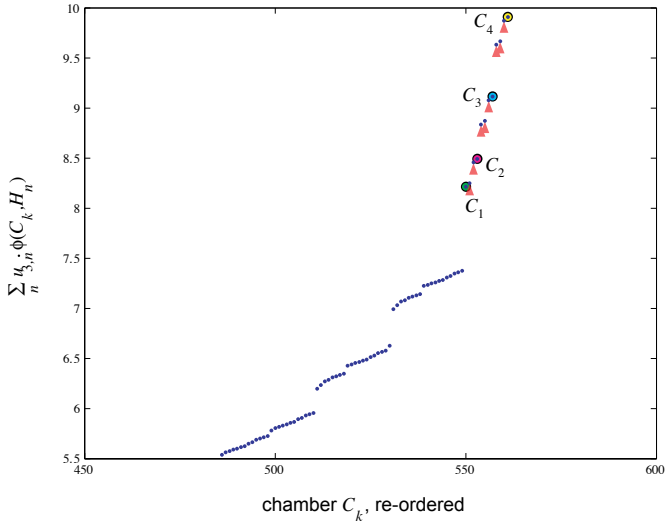


(b)

Figure 83. Improved ordering of chambers in the Fisher iris feature domain via the output node for class B of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering.



(a)



(b)

Figure 84. Improved ordering of chambers in the Fisher iris feature domain via the output node for class C of modified MLP $\widehat{F}$, (a) ordering of entire domain and (b) closeup in the superseding chambers of the ordering.

selection. These iterative techniques serve to prevent memorization (and memorization only); but, at best, they are appropriate for selecting an architecture for potential solutions. Cross validation and bootstrapping are not appropriate for identifying the solution with the best potential; that is because, in cross validation and bootstrapping, model selection is delayed until after the generalization evaluation is complete and after many solutions have been discarded arbitrarily. Once a model is selected, the estimation of generalization determined for the model's architecture is optimistic for the model itself. In contrast, our technique evaluates the generalization of an MLP after model selection. The evaluation requires only the classifier's weights, biases and the data used to train these parameters. Our technique demonstrates the weaknesses of a specific solution via deterministic measures—not through estimates—uncovering existing problems in both memorization and arrogance.

One test of the effectiveness of a generalization evaluation technique is its sensitivity to data presentation. A good technique should not be sensitive to changes in the order that training data are presented. Estimates should vary with changes to the membership of the training set, but not to mere changes in order of presentation. It should be a red flag that cross validation and bootstrapping demonstrate sensitivity to data presentation. The results of our evaluation do not change based on the order that data are presented. This is because our technique is based on the membership of chambers, and a training datum—no matter its order in the training set—is either a member of a particular chamber or it is not. This leads to much more consistent estimates of generalization based directly on the training set used to generate a particular solution and unbiased by alternative training sets used to generate alternative solutions not currently under consideration.

Another simple test of the effectiveness of a generalization evaluation technique is its domain coverage. We assert that, if the measure of the hypothesized operational set is positive, then the measure of domain coverage for a generalization evaluation should also be positive and preferably at least equal to the measure of the hypothesized operational set. Stochastic techniques exercise very little use of the domain, that is, the measure of the set of feature vectors used in the evaluation of generalization is zero since they are typically

and then starts to increase. In subsequent runs, a new solution is trained using a different subset of data. The classification error curve calculated for each run is fused with the other curves to form an estimate of generalization "error". [13]

finite. For our technique, the measure of domain coverage is positive; in low-dimensional cases such as the Fisher iris dataset above, we are able to exercise the entire domain.

6.5 Summary

In this chapter, we discussed the difference between preventing memorization in the multilayer percepton and the goal of our analysis which is to identify arrogance and memorization in classification. To balance evaluations of generalizations, we have proposed the ordered veracity-experience response (OVER) curve as a figure of merit which characterizes arrogance in classification. In addition to this new construct, we also briefly described a method similar (and superior) to error matrices tailored to identifying memorization in the MLP where memorization equates to over-isolation of training data. This method compares the membership of populated chambers (i.e., chambers containing training data) to the membership of unpopulated chambers of interest (i.e., chambers that contain data from an evaluation set or operational set but not data from the original training set). Where there is little correlation between these two memberships, there is memorization.

Expert classifiers realistically model experience across an entire domain. The goal of expertise logic and 4-value logic is to delineate appropriate expression of expertise from expressions of arrogance. Using expert classifiers, we have presented the mathematics to quantify expertise in a multilayer perceptron and the appropriate tools to resolve non-arrogant MLP classifiers from arrogant ones.

VII. Conclusions and Recommendations

This chapter summaries the contributions of this dissertation and indicates areas for further research.

7.1 Conclusions

Alan Turing first introduced the concept of "computer as pretender" in his 1950 article "Computing Machinery and Intelligence" [99]. In this dissertation, we have fleshed out the concept of "computer as interrogator". The autonomous interrogator is capable of quantifying experience and able to separate expert classifiers from "pretenders". Traditionally, we judge classifiers by assessing the amount of apparent memorization in classification. When a classifier memorizes—or overfits—training data, it is too timid in associating data near training data with the training data's label. Classifiers may also be too bold in assigning labels to data far from training data. We call this arrogance in classification, an over-statement of expertise where a classifier has no or insufficient experience.

We defined arrogance in classification formally in Chapter V and proposed a new diagnostic technique that allows one to determine when a classifier is being arrogant. We introduced the concept of the expert classifier as a special classifier that has a quantifiable skill level. Given this quantification of skill level, it is possible to determine whether a classifier is too bold or too timid in extending its training data to the larger operational set. A human is being arrogant when their expressed conviction in a decision overstates their actual experience in making similar decisions. Likewise, given an input feature vector, we say a classifier is arrogant in its classification if its veracity is high yet its experience is low. Conversely, a classifier is non-arrogant in its classification if there is a reasonable balance between its veracity and its experience. It is possible to quantify this balance, and we have discussed and demonstrated a new technique that will detect arrogance in a classifier.

We demonstrated an arrogant classifier—the multilayer perceptron—in Chapter VI, and further postulated that single-hidden-layer MLPs tend to be arrogant in their classification of significant regions in a feature space. This hypothesis is difficult to prove using first principles due to the fact that fitting a single-hidden layer perceptron to a dataset amounts

to—as we discussed in Chapter VI—a constraints problem. However, with our understanding of local min and local max chambers and the demonstrable difficulty in regulating these chambers in an arrangement of hyperplane, we go a long way in supporting our argument that the single-hidden-layer MLP is not the flexible data generalization we often need it to be.

When a classifier has been optimized to perform well on truthed data, one hopes that the classifier will perform well on operational (unknown) data. If a classifier performs well on new, previously-unseen data then we say that the classifier generalizes well. Quantifying this generalization performance has several approaches including cross validation and bootstrapping methods. In a departure from these stochastic approaches, we introduced the ordered veracity-experience curve as a means to characterize generalization performance. The OVER curve is a simplified expression of both veracity and experience and allows us to detail where in feature space the classifier is arrogant and where it is not. The benefits of our technique over other iterative and stochastic methods include the following: (1) Our technique evaluates the generalization of an MLP after model selection, requiring only the classifier's weights and biases and the data used to train these parameters. Iterative methods require that the evaluation of generalization take place over multiple training runs prior to the selection of a specific model; then, once a model is selected, the estimation of generalization for the selected model is optimistic. (2) The results of our evaluation do not change based on the order that data are presented. (3) Stochastic techniques exercise very little use of the domain, that is, the measure of the set of feature vectors used is zero. For our technique, the measure of such sets used is positive.

This work is part of a larger research effort addressing the hard problem in computational intelligence: self-evaluation. Inquisitive pattern recognition (IPR) is a reasoning capability that allows peer computer programs to compare relative skill levels and to learn from imperfect decision making. With the Theory of Confusion, expertise logic and 4-value logic, we have formed a mathematical basis for quantifying the expertise of data generalizations. This basis edges us closer to the elusive capability of self-evaluation by contrasting the opinions of peer experts. Further, we have developed confusion measures that serve to gauge the incompleteness of an data generalization beyond its initial design phase and into

its application. By cumulatively collecting evidence which supports or refutes the quality of the generalization's training set, inquisitive pattern recognition enables persistent pattern recognition, i.e., the continuous customization of the generalization specifically within its end users' application domain.

7.1.1 Contributions. The contributions presented in this dissertation are:

1. A process model for persistent learning[1]: transitioning pattern recognition tasking from supervised experimentation to self-supervised customization in the design and maintenance of a knowledge representation;
2. A simplified data fusion process model tailored for algorithm development (as opposed to the JDL standard model which is system oriented and does not scale easily);
3. The development of the Theory of Confusion: measure theory for identifying and labeling incomplete portions of information models;
4. The introduction and conceptualization of Expertise Logic and the proposed function for fusing classifiers using expertise logic,
5. The development and application of 4-value logic, the crisp set interpretation of expertise logic;
6. The ordered veracity-experience curve, a novel characterization tool for classifiers that, through investigating the ordering of a feature set, tests the generalization capability[2] of a solution; and
7. Algorithm development enabling the application of the OVER curve to multilayer perceptrons.

7.2 Recommendations for future research

The fundamentals addressed in this dissertation have broad application and are especially ripe for application to sensor-based classification problems. The military has several

[1]Recall, inquisitive pattern recognition corresponds to the passive stages of persistent learning.

[2]Appropriate data generalizations are determined by selecting the simplest architectures that successfully separate a training set while demonstrating low degrees of data memorization and inappropriate extrapolation.

applications for autonomous interrogators including automatic target recognition, treaty monitoring, battle damage assessment, and intelligence gathering. Treaty monitoring has proven an especially promising domain, and we developed the bulk of the theories and concepts contained within this work to improve pattern recognition specifically for this military application. Treaty monitoring—along with automatic target recognition, battle damage assessment and intelligence gathering—typically requires the classification of unbalanced training sets, i.e., the separation of rare events from the everyday; and the IPR methodology shows promise in resolving the special issues posed by these training sets.

Immediate goals for future research include the following:

- The application of the ordered veracity-experience curve and 4-value logic treatments to other classifiers,
- The application of 4-value logic to the selection of quality training and test sets in order to improve the convergence rates of general training methods,
- The refinement of rules that convert separation architectures to isolation architectures for the multilayer perceptron,
- Refinements to the conceptualization of the automated interrogator, and
- Demonstrations of the Inquisitive Test.

Of the items above, we find the demonstration of the Inquisitive Test the most compelling. This dissertation has contributed to this ambitious charge by developing a computational basis for doubt and intuition. This is the easy part. The much bigger challenge is to foster symbiotic interactions between artificial experts, to formalize the mathematics necessary to rigorously model and optimize intelligence amplification [84] between cooperating experts in a multi-agent system. When we achieve such capabilities, the autonomous interrogator will not only be able tell the difference between the pretender and the expert but will also be able to identify opportunities for bringing certain experts together to the benefit of a complicated task.

7.3 Summary

Inquisitive pattern recognition is a set of investigative skills that support persistent, self-supervised learning capability. It harnesses the power of deductive analysis within a computer program by augmenting decision-making processes with confusion recognition and relevancy testing. These skills provide a means to bound and regulate self-supervised learning. Applications for Inquisitive Pattern Recognition abound. The discipline is suited for generalization tasks, particularly classification problems with unbalanced training sets and a preponderance of untruthed data. In conclusion, this new direction in pattern recognition represents an important step forward toward the ultimate goal of machine learning—where a computer identifies a learning problem, defines a self-supervised experiment, interprets the results of that experiment, and finally applies these results in a practical manner.

Appendix A. Confusion among experts

We propose measures of confusion as a means to resolve the incompleteness of information among experts. Confusion is the state of uncertainty due to a dispute or disagreement over the proper interpretation of facts. In contrast to error and its truth-versus-classification evaluation, we express confusion as mapping-versus-mapping, a form based on logical expressions that can be evaluated, manipulated, and simplified over any part of feature space.

In this section, we shall define the confusion set—a construct to supplant the error matrix in the evaluation of data generalizations. In pattern recognition, the error matrix is a common tool used to evaluate an expert. Error is a state of certainty in which an expert is know to hold incorrect opinions. Typically, an error matrix is constructed by comparing the labels selected by an autonomous classifier against "truth". The error matrix (sometimes referred to as the confusion matrix in the literature) has distinct limitations. The form obscures innovation as (1) it does not communicate data trends in feature space, (2) there is no explicit comparison between the truth mapping and the autonomous mapping, only a comparison drawn from a small, potentially biased sampling of facts, and (3) there is no accounting for uncertainty in "truth" labeling. In remedy, we present an interpretation of the confusion matrix where confusion is considered within the reduced syntax of expert mappings and not on a point by point basis. For distinction sake only, we shall refer to this mapping-versus-mapping confusion matrix as the confusion set.

A.1 The confusion set

Confusion contrasts the assertions of disparate experts none of which may be authoritative. The confusion set is a logical partitioning of a feature set. The power of this form comes from comparing explicit representations of expert opinion such as data generalizations. Error is a special case of confusion where one expert knows truth; but, realistically,"truth" is difficult to obtain as an explicit mapping. Confusion may be expressed in the syntax of set theory while error is crudely sketched on a point by point basis. Formally-defined mappings readily partition feature space in ways sampled truth can not.

Unlike the number-crunched error matrix, embedded expressions within confusion set can be manipulated, simplified, and evaluated over any part of a feature set.

A.1.1 The 2×2 confusion set for crisp logic. Let a d-dimensional feature space be defined by set $\mathcal{Y} = \mathbb{R}^d$. Also, let classifiers A and B be crisp subsets of set $\mathcal{Y}$ where each is a function $\mathcal{Y} \to \{0, 1\}$. For two crisp subsets mapping to two-value logic, denote the partition mapping $\mathbb{C}^{i,j}$ of confusion set as

$$\begin{aligned} \mathbb{C}^{-,-}(A, B) &= \neg A \cap \neg B \\ \mathbb{C}^{+,+}(A, B) &= A \cap B \\ \mathbb{C}^{-,+}(A, B) &= \neg A \cap B \\ \mathbb{C}^{+,-}(A, B) &= A \cap \neg B \end{aligned}$$

where $\neg A = \mathcal{Y} - A$ and $\neg B = \mathcal{Y} - B$ are the complements of A and B, respectively. The superscripts $\mathbb{C}^{i,j}$ "$-$" and "$+$" designate the partition of confusion set with respect to A and B: For example, given an element $y \in \mathcal{Y}$, if $A(y) = 1$, point y is labeled "$+$" or true. If $A(y) = 0$, point y is labeled "$-$" or false. The superscripts i, j denote the partition of A and B, respectively; in the case of $\{i, j\} = \{+, -\}$, $A(y) = 1$ and $B(y) = 0$.

There are four partitions to the confusion set $\mathbb{C}$.

	False	True
False	$\neg A \cap \neg B$	$\neg A \cap B$
True	$A \cap \neg B$	$A \cap B$

Let us define the falsification subset Falsify(A, B) as the subset of confusion set where classifiers A and B disagree: $A(y) \neq B(y)$. The falsification subset is given by

$$\begin{aligned}\text{Falsify}(A,B) &= \mathbb{C}^{-,+}(A,B) \cup \mathbb{C}^{+,,-}(A,B) \\ &= (A \cap \neg B) \cup (\neg A \cap B) \\ &= A \ominus B\end{aligned}$$

where $\ominus$ is defined as the symmetric difference operator. Symmetric difference is a binary set operation that selects elements that belong to only one of the two sets.

Also let us define the confirmation subset $\text{Confirm}(A,B)$ as the subset of confusion set where classifiers A and B are in agreement—i.e., $A(y) = B(y)$. The confirmation subset is given by

$$\begin{aligned}\text{Confirm}(A,B) &= \mathbb{C}^{-,-}(A,B) \cup \mathbb{C}^{+,+}(A,B) \\ &= \mathcal{Y} - (A \ominus B).\end{aligned}$$

A.1.2 The 2×2 confusion set for fuzzy logic. Let a fuzzy classifier A be defined as a fuzzy subset of a set $\mathcal{Y}$ where the fuzzy classifier is a function $\mathcal{Y} \to [0,1]$. Given fuzzy classifiers A and B, the partitions of confusion set $\mathbb{C}^{i,j} \in [0,1]$ are given by:

	False	True
False	$\neg A \wedge \neg B$	$\neg A \wedge B$
True	$A \wedge \neg B$	$A \wedge B$

Notice the notation for fuzzy logic is much the same but instead of using intersect $\cap$ and union $\cup$, fuzzy logic requires the use of meet $\wedge$ and join $\vee$.

A.1.3 The 2^n confusion set for the n-class problem. Given a set of n crisp classifiers $\{A_1, A_2, \ldots, A_n\}$ that are defined on the feature set $\mathcal{Y}$ and such that each is a mapping

$\mathcal{Y} \rightarrow \{0, 1\}$, then the confusion set is given by

$$\mathbb{C}^{\mathbf{a}}(A_1, A_2, \ldots, A_n) = \wedge_{i=1}^{n} A_i^{a_i} \tag{40}$$

where $\mathbf{a}$ is a vector of length n whose elements $a_i \in \{-, +\}$ and classifiers $A_i^{a_i}$ are designated so that

$$A_i^{a_i} = \begin{cases} A_i^- & : \quad a_i = \text{“}-\text{”} \\ A_i^+ & : \quad a_i = \text{“}+\text{”} \end{cases} \tag{41}$$

and

$$A_i^-(y) = \begin{cases} 1 & : \quad y \in \neg A \\ 0 & : \quad y \notin \neg A \end{cases} \tag{42}$$

$$A_i^+(y) = \begin{cases} 1 & : \quad y \in A \\ 0 & : \quad y \notin A \end{cases} \tag{43}$$

A.1.4 Gleaning meaning from the confusion set. Confusion set has advantages over the traditional confusion matrix in that it provides a capability for completely modeling classification confidence over the entire feature set, not just at truthed samples. We must stress, though, confusion set is no silver bullet. Over the life cycle of a pattern recognition algorithm, only a small subset of the feature set will ever be relevant. Additional processing—the clustering of training data first gathered under supervision at design, then unsupervised during application—is necessary to focus on those pertinent regions of the feature set. This is where discussions on the need for intuition-based strategies—i.e. confusion recognition and relevancy testing—come to bear.

In information fusion systems, it is important to note that confusion may as likely result from poor alignment models as from noisy data. Noise is best discarded, but data that are merely poorly aligned are still of use. When applying inquisitive pattern recognition techniques to fusion, our goal is to model multi-source information in ways that facilitate the identification and classification of conflict between sources. Such methods culminate in the recognition of misaligned data and also in the identification of value-added information from single sources.

B.1 The three basic processes of information fusion

Information fusion aligns redundant information from disparate sources and then refines this information further through the assimilation of source data that is both unique and relevant. From an information management point of view, information fusion is beneficial in two ways [69]: (1) It improves signal to noise ratios, and (2) it increases the information bandwidth of a multi-source system—allowing more unique information to be captured and processed simultaneously.

Aligning redundant information improves signal to noise ratios [74], and the assimilation of unique data increases the information bandwidth of an information fusion system [79]. To realize both benefits, fusion algorithms should include three basic processes[1]: (1) the alignment of redundant data, (2) the assessment of unique information, and (3) the assimilation of all relevant information—unique and redundant—into an exhaustive world model. [74, 41, 92]

Definition B.1 *Alignment is a process that orders redundant data from disparate sources in relation to a common reference.*

Definition B.2 *Assessment is a process that evaluates aligned information to identify confusion in the world model and unique data from single or minority sources.*

[1] See Figure 85 and Table 4.

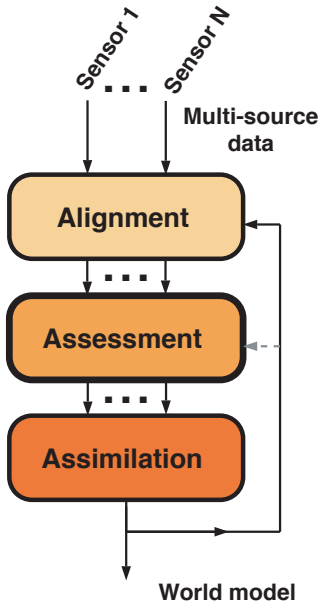


Figure 85. A flow diagram of the basic processes of an information fusion algorithm.

Table 4. The chief benefits of each information fusion process.

Data fusion process	Key data property	Benefits
Alignment	Redundancy	SNR improvement Frame of reference Data reduction
Assessment	Uniqueness	Bandwidth augmentation Falsification Conflict recognition
Assimilation	Value-added	Adaptation Conflict resolution Unified world model

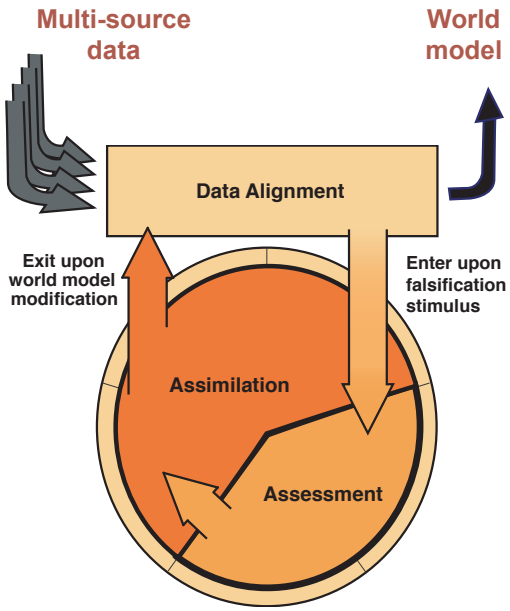


Figure 86. The processes of an information fusion algorithm depicted within the change wheel.

Definition B.3 *Assimilation is the process that modifies the world model melding together redundant and value-added unique information.*

Information fusion is a manifestation of a persistent learning process which periodically updates a world model—a unified representation of multi-source information [74]. Figure 86 matches persistent learning skills to the three basic processes of information fusion algorithms presented above. To summarize:

- Alignment is the steady state task and is not associated with the learning stages as it leverages redundant information—information that should be well understood.
- Assessment corresponds to the passive stages of the wheel.
- Assimilation encompasses the active stages.

Multi-source systems with overlapping mono-modal sensors focus on the alignment process—reducing signal to noise ratios to maintain high confidence in *target detection*. In systems with multi-modal sensors or sensors with little overlap, assessment becomes the

dominant processsiphoning value-added data from each sensor to form an understanding of *target behavior*. [41]

As information fusion systems become more sophisticated, the reasoning to tracking *target behavior* proves indispensable [92]. For efficiencys sake, it becomes crucial to recognize the shared properties of uniqueness and relevancythat is, to recognize value-added information: Unique + Relevant = Value-added.

B.2 Summary

As information fusion researchers solve the problem of how to assimilate new observations into existing knowledge representations, it behooves pattern recognition researchers to ensure that relevant new information is collected in an effective, manageable manner.

Inquisitive pattern recognition examines the conflicting observations from disparate sources in reference to the natural order of the data collection. Such an examination tenders clues over what is real, but generally unanticipated. The contributions of inquisitive pattern recognition are particularly suited to the assessment of unique information—addressing where is the value-added information and evaluating what opportunities this information offers.

Appendix C. Properties of local max and min chambers

Given the definition for a complement arrangement and the definitions for bounded and unbounded chambers, then we can demonstrate the certain interdependent properties for local max and local min chambers. Definitions for the complement arrangement and bounded and unbound chambers follow.

Definition C.1 *Let* A *be a signed hyperplane arrangement. The complement arrangement of* A *is defined to be* $\mathrm{A}^c = \Lambda^{-1}[-\Lambda(\mathrm{A})]$.

Definition C.2 *Given a feature set* $\mathcal{X} = \mathbb{R}^d$*, an arrangement* A *of hyperplanes in that feature set, and chamber* $C \in \mathrm{Cham}(\mathrm{A})$*, we say* C *is a* bounded chamber *if it is a polytope—that is, if there exists a point* $x_0 \in \mathcal{X}$ *and a radius* $r \in \mathbb{R}^+$ *such that the ball* $\mathsf{B}(x_0, r) \supset C$.

Definition C.3 *Given a feature set* $\mathcal{X} = \mathbb{R}^d$*, an arrangement* A *of hyperplanes in that feature set, and chamber* $C \in \mathrm{Cham}(\mathrm{A})$*, we say chamber* C *is an* unbounded chamber *if it is not a polytope–that is, if there does not exist a point* $x_0 \in \mathcal{X}$ *and a radius* $r \in \mathbb{R}^+$ *such that the ball* $\mathsf{B}(x_0, r) \supset C$.

Armed with these definitions, we shall prove the following theorems within the next several sections. These theorem detail key interdependencies of local max chambers and local min chambers.

Theorem C.1 *If an arrangement* A *has a central chamber then the central chamber is the only local max chamber in chamber set* $\mathrm{Cham}(\mathrm{A})$.

Theorem C.2 *If there are more than one local max chambers, the chamber set contains no central chamber.*

Theorem C.3 *If the complement arrangement of an arrangement* A *has a central chamber, then that chamber is the only local min chamber in* $\mathrm{Cham}(\mathrm{A})$.

Theorem C.4 *If an arrangement produces a central chamber, all local min chambers are unbounded.*

Theorem C.5 *If the central chamber is unbounded, there is only one local min chamber.*

Theorem C.6 *If a local max chamber is bounded, there must be more than one local min chambers.*

Theorem C.7 *If an chamber set contains only one local min chamber and that chamber is bounded, then there are multiple local max chambers and all local max chambers are unbounded.*

C.1 Complement arrangements

Given a complement arrangement $\mathrm{A}^c = \Lambda^{-1}[-\Lambda(\mathrm{A})]$, the following properties hold true.

1. For every signed hyperplane $h = \{x \in \mathbb{R}^d : w \cdot x + b = 0\}$ in A, there exists a signed hyperplane $h' = \{x \in \mathbb{R}^d : -w \cdot x - b = 0\}$ in A^c. Note $\Lambda(h) \neq \Lambda(h')$.
2. For every halfspace $H = \{x \in \mathbb{R}^d : w \cdot x + b \geq 0\}$ in $\mathcal{A} = \Lambda_*^{-1}[\Lambda(\mathrm{A})]$, there exists a hyperplane $H' = \{x \in \mathbb{R}^d : -w \cdot x - b \geq 0\}$ in $\mathcal{A}^c = \Lambda_*^{-1}[\Lambda(\mathrm{A}^c)$. Note $H' = \overline{H^c}$.
3. The chamber sets of the arrangements are the same, $\mathrm{Cham}(\mathrm{A}) = \mathrm{Cham}(\mathrm{A}^c)$.
4. The ordering of chamber sets $\mathrm{Cham}(\mathrm{A})$ and $\mathrm{Cham}(\mathrm{A}^c)$ are not the same—i.e., their ordered sets are not equal, $(\mathrm{Cham}(\mathrm{A}), \preccurlyeq) \neq (\mathrm{Cham}(\mathrm{A}^c), \preccurlyeq)$.

Consequently, the local max chambers of an arrangement A become the local min chambers of the complement arrangement A^c. If the chamber set of an arrangement has a bounded central chamber, then the chamber set of the complement arrangement does not contain a central chamber. However, if the chamber set of an arrangement has a unbounded central chamber, then the chamber set of the complement arrangement also contains an unbounded central chamber. This is proved true in Theorem C.3.

C.2 Bounded and unbounded chambers

Given a d-dimensional space, we know from geometry that a polytope is formed by at least $d+1$ hyperplanes. Therefore, if an arrangement A of hyperplanes has a cardinality

$N = \text{card(A)}$ less than the dimensional space, i.e., $N \leq d$, all of the chambers are unbounded and each chamber intersects N distinct hyperplanes. When $\text{card(A)} \leq d$, the chamber set contains one local max chamber—an unbounded central chamber—and one local min chamber.

If we have an arrangement of hyperplanes in general position where $\text{card(A)} = d + 1$, the chamber set contains one bounded chamber and $q = \sum_{i=1}^{d} \binom{d+1}{i}$ unbounded chambers. The bounded chamber—our polytope—is bounded by $d + 1$ hyperplanes; each unbounded chamber intersects d distinct hyperplanes. If there is one local max chamber and it is unbounded, again there is only one local min chamber also unbounded[1]. However, if the local max chamber is the bounded chamber, there are $\binom{d+1}{d} = d + 1$ local min chambers. This number is derived from the binomial coefficient $\binom{N}{d}$ which gives the number of unique subarrangements $\text{B} \subset \text{A}$ that can be formed where the $\text{card(B)} = d$. When $N = \text{card(A)} = d + 1$ and a local min chamber is the bounded chamber, it is the only local min chamber and there are $d + 1$ local max chambers.

An arrangement in general position whose cardinality is greater than $d + 1$ has less predictable relationship with respect to its local max and local min chambers.

C.3 Proofs

C.3.1 Central chambers.

Lemma C.1 *A central chamber is a local max chamber.*

Proof of Lemma C.1. Let $C_{\max} \in \text{Cham(A)}$ be a local max chamber of arrangement A. Then, there exists a minimal subarrangement $\text{B} \subset \text{A}$ such that

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H.$$

[1]This is proven true in Theorem C.4.

Since only chamber boundaries may intersect, no other chamber in Cham(A) exists that is contained in $C_{\max}$. That is, if $C \in \text{Cham(A)}$ and $C \neq C_{\max}$ then

$$C \not\subset \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H.$$

Observe that B $\subset$ A implies that $\Lambda_*^{-1}[\Lambda(\text{B})] \subset \Lambda_*^{-1}[\Lambda(\text{A})]$ hence

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H \quad \supseteq \quad \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H.$$

If Cham(A) contains a central chamber C_{cen}, then we know $C_{\text{cen}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H$. Therefore,

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H \quad \supseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H = C_{\text{cen}}. \tag{44}$$

Assume for contradiction purpose, that C_{cen} is a proper subset of $C_{\max}$. This is a contradiction, since $C_{\max}, C_{\text{cen}} \in \text{Cham(A)}$ and $C_{\max} \neq C_{\text{cen}}$ imply that $C_{\text{cen}} \not\subset C_{\max}$. Consequently, Equation 44 holds true only if $C_{\max} = C_{\text{cen}}$ and, thus, $C_{\max}$ is a central chamber of A. ■

The above proof is fairly trivial but it establishes the key arguments for the proof of Theorem C.1—that shows that a central chamber is the only local max chamber of its arrangement—and the proof of Theorem C.3.

Proof of Theorem C.1. Let A be a hyperplane arrangement with a central chamber $C_{\text{cen}} \in \text{Cham(A)}$. Then,

$$C_{\text{cen}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H.$$

Since no other chamber in Cham(A) exists that contains C_{cen}, then $C \in \text{Cham(A)}$ and $C \neq C_{\text{cen}}$ implies

$$C \not\supseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H. \tag{45}$$

Let arrangement B be any subarrangement of A. Observe that $\mathrm{B} \subset \mathrm{A}$ implies that $\Lambda_*^{-1}[\Lambda(\mathrm{B})] \subset \Lambda_*^{-1}[\Lambda(\mathrm{A})]$ hence

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B})]} H \quad \supseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} H$$

or

$$C_{\mathrm{cen}} \subseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B})]} H. \tag{46}$$

We know from Theorem C.1 that C_{cen} is also a local max chamber. Let chamber $C_{\max} \in \mathrm{Cham(A)}$ be local max chamber. Then, there exists a minimal subarrangement $\mathrm{B}' \subset \mathrm{A}$ such that

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B}')]} H.$$

Assume for the sake of contradiction that $C_{\max} \neq C_{\mathrm{cen}}$. This implies per Equation 45 that

$$C_{\mathrm{cen}} \not\subseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B}')]} H,$$

a contradiction to Equation 46. Therefore, $C_{\max} = C_{\mathrm{cen}}$. We conclude that if an arrangement has a central chamber then there is no other local max chamber besides the central chamber. ■

Of course the contrapositive of Theorem C.1 is true: if there are more than one local max chambers, the chamber set contains no central chamber.

Proof of Theorem C.2. Let $C_{\max}, C'_{\max} \in \mathrm{Cham(A)}$ be two distinct local max chambers, then there exists minimal subarrangements $\mathrm{B}, \mathrm{B}' \in \mathrm{A}$ such that

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B})]} H$$

and

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B}')]} H$$

Suppose, for contradiction, that a central chamber C_{cen} exists in Cham(A). That is,

$$C_{\text{cen}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H$$

Since B $\subset$ A then $\Lambda_*^{-1}[\Lambda(\text{B})] \subset \Lambda_*^{-1}[\Lambda(\text{A})]$. So,

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H \supset \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H$$

and $C_{\max} \supset C_{\text{cen}}$. Similarly, $C'_{\max} \supset C_{\text{cen}}$. Hence, $C_{\max} \cap C'_{\max} \supset C_{\text{cen}}$, and so $C_{\max} \cap C'_{\max} \neq C_{\text{cen}}$. This is a contradiction to original assumption that $C_{\max}$ is distinct from $C'_{\max}$. Thus, no C_{cen} exists. ■

C.3.2 Sink chambers. We can produce similar arguments to demonstrate that if there is a chamber $C \in$ Cham(A) that takes the form

$$C = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}.$$

then C is a local min chamber and the only local min chamber in Cham(A). First, let us define this special chamber.

Definition C.4 *Given a halfspace arrangement $\mathcal{A}$, if the intersection of all closed complement halfspaces is non-empty then the intersection forms a chamber called the* sink chamber *of the chamber set* Cham(A) *where* $\text{A} = \Lambda^{-1}[\Lambda_*(\mathcal{A})]$. *That is,* $C_{\text{sink}} \in$ Cham(A) *is a sink chamber if*

$$\begin{aligned} C_{\text{sink}} = a(\text{A}^c) &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A}^c)]} H \\ &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c} \\ &\neq \emptyset \end{aligned}$$

where $\text{A}^c = \Lambda^{-1}[-\Lambda(\text{A})]$.

Lemma C.2 *A sink chamber is a local min chamber.*

Proof of Lemma C.2. Let $C_{\min} \in \text{Cham(A)}$ be a local min chamber. Then, there exists a minimal subarrangement $\text{B} \subset \text{A}$ such that

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}.$$

Since only chamber boundaries may intersect, no other chamber in Cham(A) exists that is contained in $C_{\min}$. That is, if $C \in \text{Cham(A)}$ and $C \neq C_{\min}$ then

$$C \not\subset \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}.$$

Observe that $\text{B} \subset \text{A}$ implies that $\Lambda_*^{-1}[\Lambda(\text{B})] \subset \Lambda_*^{-1}[\Lambda(\text{A})]$ hence

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c} \quad \supseteq \quad \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}.$$

If Cham(A) contains a central chamber C_{sink}, then we know $C_{\text{sink}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}$. Therefore,

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c} \quad \supseteq \quad \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c} = C_{\text{sink}}. \tag{47}$$

Assume for contradiction purpose, that C_{sink} is a proper subset of $C_{\min}$. This is a contradiction, since $C_{\min}, C_{\text{sink}} \in \text{Cham(A)}$ and $C_{\min} \neq C_{\text{sink}}$ imply that $C_{\min} \not\subset C_{\text{sink}}$. Consequently, Equation 47 holds true only if $C_{\min} = C_{\text{sink}}$ and, thus, if chamber set Cham(A) contains a sink chamber, the local min chamber $C_{\min}$ must be that chamber. ■

Now that we have defined a sink chamber we can rewrite Theorem C.3 in a more straight forward way.

Lemma C.3 *If an arrangement* A *has a sink chamber then that chamber is the only local min chamber in* Cham(A).

Proof of Lemma C.3. Let A be a hyperplane arrangement with a sink chamber $C_{\text{sink}} \in \text{Cham(A)}$. Then,

$$C_{\text{sink}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}.$$

Since no other chamber in Cham(A) exists that contains C_{sink}, then $C \in \text{Cham(A)}$ and $C \neq C_{\text{sink}}$ then

$$C \not\supseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}. \tag{48}$$

Let arrangement B be any subarrangement of A. Observe that B $\subset$ A implies that $\Lambda_*^{-1}[\Lambda(\text{B})] \subset \Lambda_*^{-1}[\Lambda(\text{A})]$ hence

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c} \quad \supseteq \quad \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c}$$

or

$$C_{\text{sink}} \subseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}. \tag{49}$$

We know from Lemma C.2 that C_{sink} is also a local min chamber. Let chamber $C_{\min} \in$ Cham(A) be local min chamber. Then, there exists a minimal subarrangement B$'$ $\subset$ A such that

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B}')]} \overline{H^c}.$$

Assume for the sake of contradiction that $C_{\min} \neq C_{\text{sink}}$. This implies per Equation 48 that

$$C_{\text{sink}} \not\subseteq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B}')]} \overline{H^c},$$

a contradiction to Equation 49. Therefore, $C_{\min} = C_{\text{sink}}$. We conclude that if an arrangement has a sink chamber then there is no other local min chamber besides the sink chamber.

■

Now we can use Lemma C.3 to demonstrate Theorem C.3 proving that if the complement arrangement of arrangement A has a central chamber then, not only is that chamber

the only local max chamber in Cham(A^c), but it is also the only local min chamber in Cham(A).

Proof of Theorem C.3. Let A be a hyperplane arrangement with a $C \in \mathrm{Cham(A)}$ of the form

$$C = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \overline{H^c}.$$

This chamber is, by definition, the sink chamber of Cham(A) and, by Lemma C.3, the only local min chamber of Cham(A).

There exists a complement arrangement $\mathrm{A}^c = \Lambda^{-1}[-\Lambda(\mathrm{A})]$. Consequently, for every halfspace $H = \{x \in \mathbb{R}^d : w \cdot x + b \geq 0\}$ in $\mathcal{A} = \Lambda_*^{-1}[\Lambda(\mathrm{A})$, there exists a hyperplane $H' = \{x \in \mathbb{R}^d : -w \cdot x - b \geq 0\}$ in $\mathcal{A}^c = \Lambda_*^{-1}[\Lambda(\mathrm{A}^c)$ so that $H' = \overline{H^c}$. Thus, chamber C can be rewritten

$$\begin{aligned} C &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \overline{H^c} \\ &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A}^c)]} H. \end{aligned}$$

such that it is, by definition, a central chamber with respect to arrangement A^c. ■

Next, we must demonstrate that a chamber set with more than one local min chamber does not contain a sink chamber.

Lemma C.4 *If there are more than one local min chambers, the chamber set contains no sink chamber.*

Proof of Lemma C.4. Let $C_{\min}, C'_{\min} \in \mathrm{Cham(A)}$ be two distinct local min chambers, then there exists minimal subarrangements $\mathrm{B}, \mathrm{B}' \in \mathrm{A}$ such that

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B})]} H$$

and

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B}')]} H$$

Suppose, for contradiction, that a sink chamber C_{sink} exists in Cham(A). That is,

$$C_{\text{sink}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H$$

Since B $\subset$ A then $\Lambda_*^{-1}[\Lambda(\text{B})] \subset \Lambda_*^{-1}[\Lambda(\text{A})]$. So,

$$\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H \supset \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H$$

and $C_{\text{min}} \supset C_{\text{sink}}$. Similarly, $C'_{\text{min}} \supset C_{\text{sink}}$. Hence, $C_{\text{min}} \cap C'_{\text{min}} \supset C_{\text{sink}}$, and so $C_{\text{min}} \cap C'_{\text{min}} \neq C_{\text{sink}}$. This is a contradiction to original assumption that C_{min} is distinct from C'_{min}. Thus, no C_{sink} exists. ■

C.3.3 Bounded and unbounded chambers. There is one circumstance in which there is only one local min chamber and only one local max chamber. This circumstance is discussed in the proof of Theorem C.5. First, however, we shall show that the existence of a central chamber in a chamber set means that all local min chambers are unbounded.

Proof of Theorem C.4. Given a signed hyperplane arrangement A, assume a central chamber C_{cen} exists in Cham(A). Therefore,

$$C_{\text{cen}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H \neq \emptyset.$$

Let $C_{\text{min}} \in$ Cham(A) be a local min chamber, then there exists a minimal subarrangement B $\subset$ A such that

$$C_{\text{min}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}.$$

Assume for contradiction that C_{min} is bounded. Let point $x_0 \in C_{\text{cen}}$ then there exists a radius $r_0 > 0$ such that the ball $\mathsf{B}(x_0, r_0) \supset C_{\text{min}}$. For each hyperplane $h \in$ B, let $r_h = \text{dist}(x_0, h)$. Of these, define the largest distance between point x_0 and a hyperplane in B as

$$r_l = \max_{h \in \text{B}} r_h$$

and observe that $r_l < r_0$. Therefore, ball $\mathsf{B}(x_0, r_l)$ is supported by hyperplane $h_l \in$ B. By definition of C_{min} then $C_{min} \subset \overline{H_l^c}$ where $H_l = \Lambda_*^{-1}[\Lambda(h_l)]$. Therefore, the orientation of hyperplane h_l (and moreover the sign of it normal) is such that $\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} H = \emptyset$, a contradiction to the original hypothesis that the Cham(A) contains a central chamber. Therefore, in conclusion, there are no bounded local min chambers in a chamber set that contains a central chamber. ■

Now we can prove that if C_{cen} is the central chamber of A such that C_{cen} is unbounded, then there is only one local min chamber C_{min} and it is an unbounded sink chamber.

Proof of Theorem C.5. Let C_{cen} be the central chamber of Cham(A) such that C_{cen} is unbounded. Because the chamber set contains a central chamber, we know from Theorem C.4 that all local min chamber are unbounded.

Assume for contradiction that there are more than one local min chambers $C_{\text{min}}, C'_{\text{min}}$.

Let us define a ball $\mathsf{B}(x_0, r_h)$ such that $x_0 \in C_{\text{min}}$ and the radius $r_h = \text{dist}(x_0, h)$ where h is any hyperplane in A. Note by definition of C_{cen} halfspace $H = \Lambda_*^{-1}[\Lambda(h)]$ shall contain the central chamber but, by definition of C_{min}, not the ball or the local min chamber. As the local min chamber is not in any of the implied halfspaces, it is a sink chamber as defined in Definition C.4. Thus, from Theorem C.3 there is no other local min chamber and $C_{\text{min}} = C'_{\text{min}}$. ■

If a local max chamber is bounded, there must be more than one local min chambers.

Proof of Theorem C.6. Let C_{max} be bounded local max chamber. Then, there exists a minimal subarrangement B $\subset$ A such that

$$C_{\text{max}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{B})]} H.$$

Assume, for contradiction, that there is only one local min chamber C_{min} in Cham(A). We know from Lemma C.3 that if a chamber set contains only one local min chamber, then

that chamber is a sink chamber of the form

$$\begin{aligned} C_{\min} &= C_{\text{sink}} \\ &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \overline{H^c}. \end{aligned}$$

If $C_{\max}$ is a bounded chamber, there exists a ball $\mathsf{B}(x_0, r_0)$ where $x_0 \in C_{\min}$ and the radius $r_0 > 0$ such that $\mathsf{B}(x_0, r_0) \supset C_{\max}$. For each hyperplane $h \in \mathrm{B}$, let $r_h = \mathrm{dist}(x_0, h)$ and let the largest distance be

$$r_l = \max_{h \in \mathrm{B}} r_h.$$

Thus, there exists a ball $\mathsf{B}(x_0, r_l)$ is supported by hyperplane $h_l \in \mathrm{B}$.

By definition of $C_{\max}$, the halfspace $H_l = \Lambda_*^{-1}[\Lambda(h_l)]$ must contain the local max chamber and thus the ball which contains points in the local min chamber. We conclude $C_{\min}$ is not a sink chamber as $C_{\min} \subset H_l$ and, thus,

$$C_{\min} \neq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\mathrm{A})]} \overline{H^c}.$$

As the chamber set does not contain a sink chamber, there must be more than one local min chamber in Cham(A). ■

Since a central chamber is a local max chamber Theorem C.6 can be extended easily to Corollary 1.

Corollary 1 *If a central chamber is bounded, there must be more than one local min chambers.*

Finally, Theorem C.7 is demonstrated via the following two Lemmas.

Lemma C.5 *If an arrangement produces a sink chamber, all local max chambers are unbounded.*

Proof of Lemma C.5. Given a signed hyperplane arrangement A, assume a sink chamber C_{sink} exists in Cham(A). Therefore,

$$C_{\text{sink}} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c} \neq \emptyset.$$

Let $C_{\max} \in \text{Cham(A)}$ be a local max chamber, then there exists a minimal subarrangement $\text{B} \subset \text{A}$ such that

$$C_{\max} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} H.$$

Assume for contradiction that $C_{\max}$ is bounded. Let $x_0 \in C_{\text{sink}}$ then there exists a radius $r > 0$ such that $B(x_0, r) \supset C_{\max}$. Therefore, there must exist a smaller radius r_0 ($0 < r_0 < r$) such that the radius is reduced until the ball $\mathsf{B}(x_0, r_0)$ is supported by some hyperplane $h \in \mathcal{B}$. This hyperplane implies a halfspace $H = \Lambda_*^{-1}[\Lambda(h)]$ such that due to the ball's orientation, $H \cap \mathsf{B}(x_0, r_0) = \mathsf{B}(x_0, r_0)$. This requires that $H \supset C_{\text{sink}}$ and $\bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} \overline{H^c} = \emptyset$, a contradiction to the original hypothesis that the Cham(A) contains a sink chamber. Therefore, in conclusion, there are no bounded local max chambers in a chamber set that contains a sink chamber. ■

Lemma C.6 *If a local min chamber is bounded, there must be more than one local max chambers.*

Proof of Lemma C.6. Let $C_{\min}$ be bounded local min chamber. Then, there exists a minimal subarrangement $\text{B} \subset \text{A}$ such that

$$C_{\min} = \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{B})]} \overline{H^c}.$$

Assume, for contradiction, that there is only one local max chamber $C_{\max}$ in Cham(A). We know from Theorem C.1 that if a chamber set contains only one local max chamber, then

that chamber is a central chamber of the form

$$\begin{aligned} C_{\max} &= C_{\text{cen}} \\ &= \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H. \end{aligned}$$

If $C_{\min}$ is a bounded chamber, there exists a ball $\mathsf{B}(x_0, r_0)$ such that the center point $x_0 \in C_{\max}$ and the radius $r_0 > 0$ is large enough so that $\mathsf{B}(x_0, r_0) \supset C_{\min}$. Then there exists a smaller r $(0 > r > r_0)$ where the radius has been reduced until the ball is supported by a hyperplane $h \in \text{B}$ and the implied halfspace $H = \Lambda_*^{-1}[\Lambda(h)]$. Due to the orientation of ball $\mathsf{B}(x_0, r)$, this halfspace H does not contain the local min chamber nor the ball which contains points in the local max chamber. Therefore, $C_{\max} \not\subset H$. We conclude $C_{\max}$ is not a central chamber as

$$C_{\max} \neq \bigcap_{H \in \Lambda_*^{-1}[\Lambda(\text{A})]} H.$$

Per Theorem C.2, as the chamber set does not contain a central chamber, there must be more than one local max chamber in Cham(A). ■

Since a sink chamber is a local min chamber Lemma C.6 can be extended easily to Corollary 2.

Corollary 2 *If a sink chamber is bounded, there must be more than one local max chambers.*

Other proofs in development shall demonstrate properties of adjacent chambers such as the conjecture that, given a d-dimensional space, an unbounded chamber C_{un} bounded by d hyperplanes is adjacent to only unbounded chambers such that $\text{card}([C]_{adj}) = d$. In a related conjecture, an unbounded chamber bounded by $d + k$ chambers is adjacent to d unbounded chambers and k bounded chambers.

C.4 Summary

In this appendix, we have demonstrated that local max and local min chambers have the following general properties:

1. A central chamber is a local max chamber.

(a) If an arrangement produces only one local max chamber, it is a central chamber.

(b) If an arrangement produces a central chamber, all local min chambers are unbounded.

(c) If the central chamber is unbounded, there is only one local min chamber.

(d) If the central chamber is bounded, there must be more than one local min chambers.

2. If an chamber set contains only one local min chamber and that chamber is bounded then there are multiple local max chambers and those local max chambers are unbounded.

3. If there are more than one local max chambers, the chamber set contains no central chamber and one or multiple local min chambers.

Using backpropagation, we derived several single-hidden-layer MLPs that implemented the XOR dataset given at the end of this appendix in Table 5 and illustrated in Figure 87. Below, we list the parameter sets of the multilayer perceptrons.

The first MLP we investigated was a 5-hidden-node multilayer perceptron. We used this MLP to demonstrate many of the algorithms and results of its investigation are given in Figures 43, 44, 46, 50-55, and 65. The network's hidden-layer weights $[\overline{W}_h : \overline{b}_h]$ and sets of output weights $[\overline{W}_o : \overline{b}_o]^T$ follow:

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} -4.2524 & -1.7684 & 3.8185 \\ -1.1778 & 3.2085 & 2.1537 \\ 2.7051 & 4.2477 & -2.5031 \\ 3.3261 & -3.1723 & -1.4307 \\ -2.3225 & -2.6394 & 3.1606 \end{bmatrix}, \tag{50}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.0067 & 1.9402 \\ 0.5147 & 1.1525 \\ 2.7893 & 2.9208 \\ 1.9374 & 1.8115 \\ 1.0946 & 1.3958 \\ -1.4431 & -2.7789 \end{bmatrix}. \tag{51}$$

We derived a second 5-hidden-node multilayer perceptron whose results are given in Figures 57(b) and 58(b). The network's hidden-layer weights $[\overline{W}_h : \overline{b}_h]$ and sets of output weights $[\overline{W}_o : \overline{b}_o]^T$

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 3.8553 & 3.4229 & -2.1822 \\ 1.8157 & -3.7489 & 0.1545 \\ -0.7123 & -1.2915 & 1.2724 \\ -3.6947 & 2.5422 & -0.2209 \\ -0.9548 & 0.3104 & -0.3053 \end{bmatrix} \quad [\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 3.2875 \\ 2.7693 \\ 1.5929 \\ 2.6929 \\ 0.7763 \\ 0.4950 \end{bmatrix} \tag{52}$$

For the same XOR dataset, networks trained by backpropagation include a 15-hidden-node MLP whose results are illustrated in Figure 56. The network's parameter set is

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 10.0544 & 26.7693 & -13.6248 \\ 19.1789 & 20.3360 & -11.6964 \\ 26.0609 & 10.9662 & -4.4204 \\ 28.3137 & -0.5089 & 4.6333 \\ 25.6589 & -11.9810 & 13.9636 \\ 18.5674 & -21.3816 & 22.2659 \\ 8.2655 & -27.0851 & 28.1048 \\ -3.5449 & -28.2024 & 30.3699 \\ -15.0252 & -24.6709 & 28.3869 \\ -22.7099 & -16.1526 & 24.3250 \\ -27.8009 & -5.3887 & 15.9506 \\ -27.7016 & 6.2911 & 6.3172 \\ -22.6675 & 16.9213 & -2.6535 \\ -13.9875 & 24.3797 & -10.1279 \\ -2.3890 & 28.3996 & -13.3388 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 0.6777 & -0.6777 \\ 2.0362 & -2.0362 \\ 0.5655 & -0.5655 \\ 0.0256 & -0.0256 \\ 0.0252 & -0.0252 \\ 0.0252 & -0.0252 \\ 0.0252 & -0.0252 \\ 0.0984 & -0.0984 \\ 1.2626 & -1.2626 \\ 0.8205 & -0.8205 \\ -1.2976 & 1.2976 \\ 0.0395 & -0.0395 \\ 1.1361 & -1.1361 \\ 0.8494 & -0.8494 \\ -0.9806 & 0.9806 \\ -2.9748 & 2.9748 \end{bmatrix}.$$

We also derived two 10-hidden node MLPs. The first network's results were illustrated in Figures 57(c), 57(d), 58(c), and 58(c).

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 12.6617 & 24.4726 & -13.4804 \\ 23.9651 & 13.1706 & -6.8326 \\ 27.3270 & -3.3697 & 6.5654 \\ 20.1274 & -18.7884 & 19.5908 \\ 5.2401 & -27.0295 & 28.0299 \\ -12.5842 & -25.3943 & 27.7972 \\ -23.3114 & -13.2302 & 22.1696 \\ -26.9902 & 3.4370 & 9.6167 \\ -20.1681 & 18.6688 & -4.7401 \\ -4.5758 & 28.0143 & -10.9840 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 1.9578 & 1.5083 \\ 1.1228 & 1.2153 \\ 0.3181 & -0.1331 \\ 0.3180 & -0.1331 \\ 0.3170 & -0.1335 \\ 0.5488 & 0.7376 \\ 1.7557 & 1.0301 \\ -1.6335 & -1.3748 \\ 1.5422 & 1.7320 \\ -1.5221 & -1.5108 \\ -2.6820 & -1.1331 \end{bmatrix}$$

The second 10-hidden node MLPs results were illustrated in Figures 57(e), 57(f), 58(e), and 58(e).

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 12.6360 & 24.6478 & -13.4388 \\ 24.0057 & 13.2309 & -6.5619 \\ 27.3270 & -3.3696 & 6.5654 \\ 20.1274 & -18.7884 & 19.5907 \\ 5.2399 & -27.0301 & 28.0292 \\ -12.3563 & -25.1863 & 28.0536 \\ -23.3784 & -13.2448 & 22.0867 \\ -27.0213 & 3.4345 & 9.5005 \\ -20.1669 & 18.6652 & -4.7403 \\ -4.6328 & 27.9094 & -11.1840 \end{bmatrix}$$

$$[\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 1.6751 & 1.4599 \\ 1.0303 & 1.1061 \\ 0.2797 & -0.2008 \\ 0.2796 & -0.2008 \\ 0.2788 & -0.2010 \\ 0.7410 & 0.8945 \\ 1.5217 & 1.0832 \\ -1.4895 & -1.3491 \\ 1.3387 & 1.5408 \\ -1.2799 & -1.3831 \\ -2.7204 & -1.2008 \end{bmatrix}$$

We chose the following 3-hidden-node solution for further manipulation. See Figures 57(a), 58(a), and 59.

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 3.5553 & 3.4229 & -2.1822 \\ 2.0157 & -3.7489 & 0.1545 \\ -3.6947 & 2.6422 & -0.2209 \end{bmatrix} \quad [\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.8015 \\ 2.3128 \\ 2.2482 \\ 0.4765 \end{bmatrix}$$

This particular network is a simplification of the 5-hidden-layer MLP listed in Equation 52. After its simplification, the 3-hidden-layer network was augmented with 4 additional hidden layer nodes. An isolation solution was derived from the resulting 7-hidden node network, and results are illustrated in Figure 60.

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 3.5553 & 3.4229 & -2.1822 \\ 2.0157 & -3.7489 & 0.1545 \\ -3.6947 & 2.6422 & -0.2209 \\ 1.0000 & 0.1000 & 0.1500 \\ 0.1000 & 1.0000 & 0.1500 \\ -1.0000 & 0 & 1.1500 \\ 0 & -1.0000 & 1.1500 \end{bmatrix} \quad [\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.8015 \\ 2.3128 \\ 2.2482 \\ 6.5000 \\ 6.5000 \\ 6.5000 \\ 6.5000 \\ -28.5235 \end{bmatrix} \tag{53}$$

Alternatives to the above architecture were suggested and utilized an MLP which isolates the true class,

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 3.5553 & 3.4229 & -2.1822 \\ 2.0157 & -3.7489 & 0.1545 \\ -3.6947 & 2.6422 & -0.2209 \\ 1.0000 & 0.1000 & 0.1500 \\ 0.1000 & 1.0000 & 0.1500 \\ -1.0000 & 0 & 1.1500 \\ 0 & -1.0000 & 1.1500 \end{bmatrix} \quad [\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.8015 \\ 2.3128 \\ 2.2482 \\ 0.7500 \\ 0.7500 \\ 0.7500 \\ 0.7500 \\ -5.5235 \end{bmatrix}$$

and an MLP which isolates the false class

$$[\overline{W}_h : \overline{b}_h]^T = \begin{bmatrix} 3.5553 & 3.4229 & -2.1822 \\ 2.0157 & -3.7489 & 0.1545 \\ -3.6947 & 2.6422 & -0.2209 \\ -1.0000 & -0.1000 & -0.1500 \\ -0.1000 & -1.0000 & -0.1500 \\ 1.0000 & 0 & -1.1500 \\ 0 & 1.0000 & -1.1500 \end{bmatrix} \quad [\overline{W}_o : \overline{b}_o]^T = \begin{bmatrix} 2.8015 \\ 2.3128 \\ 2.2482 \\ 0.7500 \\ 0.7500 \\ 0.7500 \\ 0.7500 \\ 4.4765 \end{bmatrix}.$$

The results from these two MLP are presented in Figures 61 and 62 respectively; the combined result of the two networks is presenting in Figure 63.

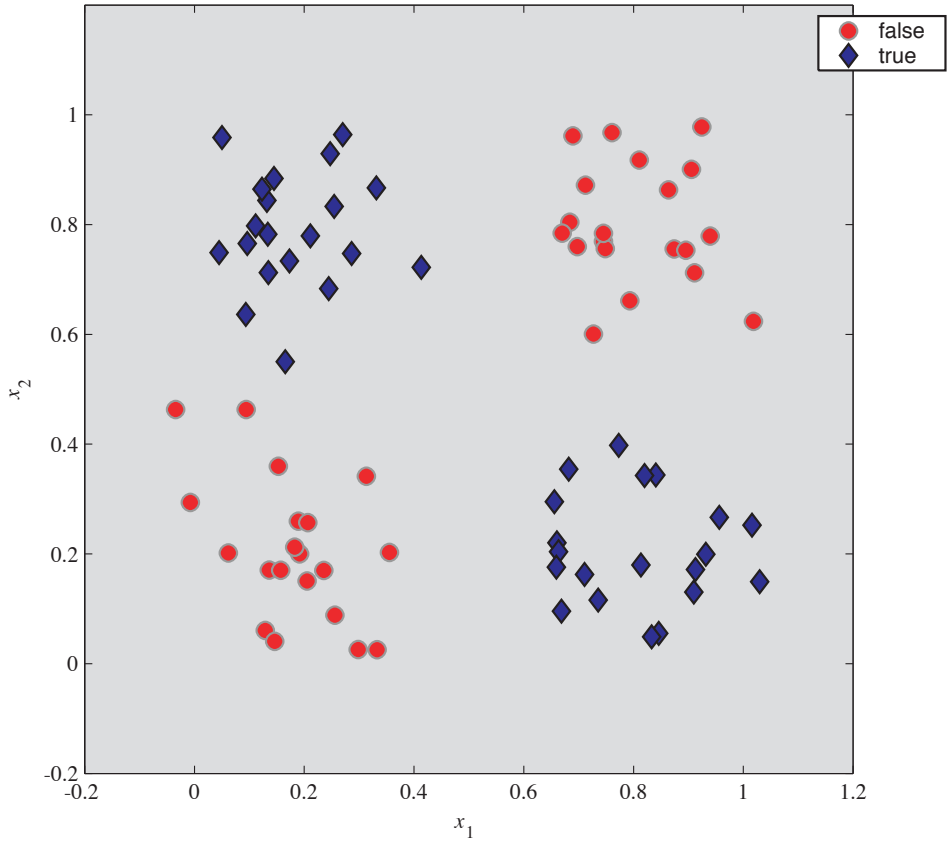


Figure 87. A plot of the 2-class dataset used in the XOR problem.

Table 5. The training set $\mathfrak{D} = \mathfrak{D}_+ \cup \mathfrak{D}_-$ used for the XOR problem. Dataset $\mathfrak{D}_+$ contains the "true" training vectors, and dataset $\mathfrak{D}_-$ contains the "false" training vectors.

$\mathfrak{D}_+$	
x_1	x_2
0.0503	0.9586
0.1336	0.7824
0.1319	0.8441
0.1230	0.8646
0.0935	0.6364
0.1654	0.5503
0.1348	0.7124
0.3314	0.8669
0.2112	0.7794
0.2700	0.9640
0.4133	0.7220
0.0962	0.7655
0.2473	0.9290
0.1449	0.8841
0.2863	0.7472
0.0448	0.7489
0.2546	0.8333
0.1113	0.7976
0.1730	0.7338
0.2447	0.6834
0.7107	0.1627
0.6603	0.2202
0.8407	0.3438
0.6632	0.2041
0.6818	0.3544
0.9560	0.2667
1.0156	0.2523
1.0296	0.1492
0.7357	0.1159
0.6594	0.1758
0.9316	0.1994
0.6558	0.2951
0.8199	0.3429
0.9129	0.1716
0.8458	0.0554
0.6685	0.0956
0.7729	0.3977
0.9099	0.1303
0.8329	0.0489
0.8134	0.1798

$\mathfrak{D}_-$	
x_1	x_2
0.0503	0.9586
0.1336	0.7824
0.1319	0.8441
0.1230	0.8646
0.0935	0.6364
0.1654	0.5503
0.1348	0.7124
0.3314	0.8669
0.2112	0.7794
0.2700	0.9640
0.4133	0.7220
0.0962	0.7655
0.2473	0.9290
0.1449	0.8841
0.2863	0.7472
0.0448	0.7489
0.2546	0.8333
0.1113	0.7976
0.1730	0.7338
0.2447	0.6834
0.7107	0.1627
0.6603	0.2202
0.8407	0.3438
0.6632	0.2041
0.6818	0.3544
0.9560	0.2667
1.0156	0.2523
1.0296	0.1492
0.7357	0.1159
0.6594	0.1758
0.9316	0.1994
0.6558	0.2951
0.8199	0.3429
0.9129	0.1716
0.8458	0.0554
0.6685	0.0956
0.7729	0.3977
0.9099	0.1303
0.8329	0.0489
0.8134	0.1798

Bibliography

1. Air Force Strategy Division, Office of the Secretary of the Air Force, Washington, D.C., *Air Force Executive Guidance*, January 1996.

2. S. G. Alsing, *The Evaluation of Competing Classifiers*. Ph.D. dissertation, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 2000.

3. S. G. Alsing, K. W. Bauer, Jr., and J. O. Miller, "A multinomial selection procedure for evaluating pattern recognition algorithms," *Pattern Recognition* **35**, pp. 2397–2412, November 2002.

4. S. G. Alsing, K. W. Bauer, Jr., and M. E. Oxley, "Convergence for receiver operating characteristic curves & the performance of neural networks," in *Intelligent Engineering Systems through Artificial Neural Networks*, *Proceedings of Artificial Neural Networks in Engineering International Conference* **9**, pp. 947–952, (St Louis, MO), Nov 1999.

5. T. M. Apostol, *Mathematical Analysis*, Addison-Wesley, Reading, MA, second ed., 1974.

6. D. Avis, "lrs: A revised implimentation of the reverse search vertex enumeration algorithm." World Wide Web, January 1999. Available at *http://cgm.cs.mcgill.ca/˜avis/doc/avis/Av98a.ps.gz*.

7. R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, *Readings in Human-Computer Interaction: Toward the Year 2000*, Morgan Kaufmann Publishers, San Francisco, second ed., 1995.

8. P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, eds., **9**, pp. 134–140, MIT Press, Cambrideg, MA, 1997.

9. W. W. Bartley, *The Retreat to Commitment*, Chatto & Windus, 1964.

10. E. B. Baum and D. Haussler, "What size net gives valid generalization?," *Neural Computation* **1**(1), pp. 151–160, 1989.

11. R. Baumann, D. Glauser, D. Tappy, C. Baur, and R. Clavel, "Force feedback for virtual reality based minimally invasive surgery simulator," in *Medicine Meets Virtual Reality 4: Health Care in the Information Age*, pp. 564–579, IOS Press, (Washington, D.C.), 1996.

12. M. Billinghurst, J. Savage, P. Oppenheimer, and C. Edmond, "The expert surgical assistant: An intelligent virtual environment with multimodal input," in *Medicine Meets Virtual Reality 4: Health Care in the Information Age*, pp. 590–607, IOS Press, (Washington, D.C.), 1996.

13. C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

14. Y. M. M. Bishop, S. E. Fienberg, and P. W. Holland, *Discrete Multivariate Analysis: Theory and Practice*, MIT Press, Cambridge, MA, 1975.

15. S. M. Brown, E. Santos, Jr., S. B. Banks, and M. E. Oxley, "Using explicit requirements and metrics for interface agent user model correction," in *Second International Conference on Autonomous Agents (Agents '98)*, (St. Paul, Minnepolis), May 1998.

16. R. Burch, "Charles Sandford Peirce," in *Stanford Encyclopedia of Philosophy*, E. N. Zalta, ed., World Wide Web, June 2001. Available at *http://plato.stanford.edu/entries/peirce/*.

17. J. B. Campbell, *Introduction to Remote Sensing*, Guilford Press, 1987.

18. M. A. Carter and M. E. Oxley, "Evaluating the vapnik-chervonenkis dimension of artificial neural networks using the Poincare polynomial," *Neural Network Letters* **12**, pp. 403–408, 1999.

19. M. A. Carter and M. E. Oxley, "Erratum: Evaluating the vapnik-chervonenkis dimension of artificial neural networks using the Poincare polynomial," *Neural Network Letters* **14**, pp. 1469–1470, 2001.

20. D. J. Chalmers, *The Conscious Mind: In Search of a Fundamental Theory*, Oxford University Press, New York, 1997.

21. R. T. Cox, "Of inference and inquiry, an essay in inductive logic," in *Maximum Entropy Formalism*, R. D. Levine and M. Tribus, eds., pp. 119–167, MIT Press, Cambridge, MA, 1978.

22. H. Croft, K. Falconer, and R. Guy, *Unsolved Problems in Geometry*, Springer-Verlag, New York, 1991.

23. G. Cybenko, "Approximation by superpositions of a sigmoid function," *Mathematics of Control Signals and Systems* **2**, pp. 303–314, 1989.

24. G. Davenport, "Seeking dynamic, adaptive story environments," *IEEE MultMedia* **1**(3), pp. 9–13, 1994.

25. Defense Information Systems Agency, *Command, Control, Communications and Computers and Intelligence (C4I) Modeling, Simulation and Assessment Directorate (D8)*, July 1996. Available at *http://www.disa.mil/info/pao04v.html*.

26. J. Driver, "Enhancement of selective listening by illusory mislocation of speech sounds due to lip-reading," *Nature* **381**, pp. 66–68, 9 May 1996.

27. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

28. S. E. Fahlam and C. Leviere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, ed., **2**, pp. 524–532, Morgan Kaufmann Publishers, San Francisco, 1990.

29. M. Fisch and A. Turquette, "Peirce's triadic logic," in *Transactions of the Charles S. Peirce Society*, **11**, pp. 71 – 85, 1966.

30. D. Fisher, M. Pazzani, and P. Langley, eds., *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufmann, 1991.

31. R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics* **7**, pp. 179–188, 1936.

32. J. Flanagan and I. Marsic, "Issues in measuring the benefits of multimodal interfaces," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Munich, Germany), April 1997.

33. R. L. Fry, "Cybernetic systems based on inductive logic," in *Maximum Entropy and Bayesian Methods, Paris 2000*, A. Mohammad-Djafari, ed., Kluwer Academic Publishers, Dordrecht, 2000.

34. K. Fukuda, "Frequently asked questions in polyhedral computation." World Wide Web Page, 2000. Available at *http://www.ifor.math.ethz.ch/~fukuda/polyfaq/polyfaq.html*.

35. B. W. Garcia, "Design and prototype of the afit virtual emergency room: a distributed virtual environment for emergency medical simulation," Master's thesis, Air Force Institute of Technology, 1996.

36. M. T. Gardner, "A distributed interactive simulation based remote debriefing tool for red flag missions," Master's thesis, Air Force Institute of Technology, 1993.

37. R. H. Gilkey and J. M. Weisenberger, "Sense of presence for the suddenly deafened adult: Implications for virtual environments," *Presence, the Journal of Teleoperators and Virtual Environments* **4**(4), pp. 357–363, 1995.

38. R. C. Gonzales and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

39. W. E. L. Grimson, G. J. Ettinger, S. J. White, T. Lozano-Pérez, W. M. Wells III, and R. Kikinis, "An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization," *Transactions on Medical Imaging* **15**(2), pp. 129–140, 1996.

40. J. Grudin, "Groupware and social dynamics: eight challenges for developers," in *Readings in Human-Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, eds., pp. 762–774, Morgan Kaufmann Publishers, San Francisco, second ed., 1995.

41. D. L. Hall, *Mathematical techniques in multi-sensor data fusion*, Artech House, Boston, 1992.

42. P. R. Halmos, ed., *Measure Theory*, Springer Graduate Texts in Mathematics, New York, 1950.

43. M. L. Hinman, Program Manager, Adaptive Sensor Fusion, Air Force Research Laboratory. Interview, 22 May 2000.

44. J. S. U. Hjorth, *Computer Intensive Statistical Methods: Validation, Model Selection, and Bootstrap*, Chapman & Hall, London, 1994.

45. K. Hwang and Z. Xu, *Scalable parallel computing: technology, architecture, programming*, WCB/McGraw-Hill, Boston, 1998.

46. G. Johnson, "Separating insolvable and difficult." *New York Times*, Science Section, 13 July 1999.

47. R. E. Johnson, *University Algebra*, Prentice Hall, Englewood, NJ, 1966.

48. R. Johnston, "Do you need VR? Implementing a needs assessment for virtual environments," *Virtual Reality Special Report* **2**, pp. 29–33, September/October 1995.

49. S. Kaufman, I. Poupyrev, E. Miller, M. Billinghurst, P. Oppenheimer, and S. Weghorst, "New interface metaphors for complex information space visualization: an ECG monitor object," in *Medicine Meets Virtual Reality: Global Healthcare Grid*, K. S. Morgan, H. M. Hoffman, D. Stredney, and S. J. Weghorst, eds., **39**, IOS Press OHMSHA, Amsterdam, 1997. Available at *http://www.hitl.washington.edu/projects/limit/papers.html*.

50. S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, Upper Saddle River, NJ, 1993.

51. J. W. Kelley, "Air power in 2025: executive summary." Air University, Maxwell AFB AL, August 1996. Available at *http://www.au.af.mil/au/2025/*.

52. M. Kelso, P. Weyhrauch, and J. Bates, "Dramatic presence," *Presence, the Journal of Teleoperators and Virtual Environments* **2**(1), pp. 1–15, 1993.

53. S. Krantz, *Handbook of Complex Analysis*, Birkhäuser, Boston, 1999.

54. V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of parallel algorithms*, Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

55. S. W. Laffan, "Spatially assessing model error using geographically weighted regression," in *IV International Conference on GeoComputation*, (Fredericksburg, VA), July 1999.

56. B. Laurel, *Computers as theatre*, Addison-Wesley, New York, 1991.

57. C. I. Lewis, *A Survey of Symbolic Logic*, University of California Press, Berkeley, 1918.

58. J. Llinas, Technical Advisor to the DoD's Joint Directors of Laboratories (JDL) Data Fusion Group. Interview, Dec 1999.

59. N. A. Lynch, *Distributed algorithms*, Morgan Kaufmann Publishers, San Francisco, 1996.

60. P. Maes, "Modeling adaptive autonomous agents," *Artificial Life Journal* **1**(1 & 2), pp. 135–162, 1994.

61. A. L. Magnus and S. C. Gustafson, "Inquisitive pattern recognition," in *Proceedings of 3rd International Conference on Information Fusion*, **1**, (Paris, France), July 2000.

62. A. L. Magnus and M. E. Oxley, "The theory of confusion," in *Proceedings of SPIE, Applications and Science of Neural Networks, Fuzzy Systems, & Evolutionary Computation IV*, pp. 105–116, (San Diego, CA), August 2001.

63. A. L. Magnus and M. E. Oxley, "Generalization tools for multilayer perceptrons," in *Proceedings of 2002 World Congress on Computational Intelligence*, (Honolulu, HI), May 2002.

64. A. L. Magnus and M. E. Oxley, "Quantifying the expertise of classifiers using 4-value logic," in *Proceedings of SPIE, Applications and Science of Computational Intelligence V*, **4739**, (Orlando, FL), April 2002.

65. The Mathworks, Natik, MA, *MATLAB Optimization Toolbox, User's Guide, Version 5*, May 1997.

66. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics* **5**, pp. 115–133, 1943.

67. R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, "Determining computational complexity from characteristic 'phase transitions'," *Nature* **400**, pp. 133–137, 8 July 1999.

68. L. R. Myers, *Radial complexity estimation for improved generalization in artificial neural networks*. PhD dissertation, Air Force Institute of Technology, 1998.

69. National Critical Technologies Panel, Arlington, VA, *National Critical Technologies Report,* PB95-255758GAR, March 1995. Available at *http://www1.whitehouse.gov/WH/EOP/OSTP/CTIformatted/index.html*.

70. A. W. Naylor and G. R. Sell, eds., *Linear Operator Theory in Engineering and Science*, Springer-Verlag, New York, 1971.

71. H. S. Nwana, "Software agents: An overview," *The Knowledge Engineering Review* **11**, pp. 1–40, September 1996.

72. P. Orlik, *Introduction to arrangements*, no. 72 in Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics, American Mathematical Society, Providence, RI, 1980.

73. M. E. Oxley and M. A. Carter, "Capability measures of artificial neural network architectures based on soft shattering," in *Proceedings of SPIE, Applications and Science of Computational Intelligence IV*, **4390**, (Orlando, FL), April 2001.

74. R. Parra-Loera, W. E. Thompson, and A. P. Salvi, "Adaptive selection of sensors based on individual performances in a multisensor environment," in *Data Structures and Target Classification, Proceedings of the SPIE*, **1470**, pp. 30–36, (Orlando,FL), 1991.

75. C. S. Peirce, "On the algebra of logic; a contribution to the philosophy of notation," *American Journal of Mathematics* **7**, pp. 180–202, 1885.

76. C. S. Peirce, "Logic, regarded as semeiotic." Manuscript L75, Cambridge Institute, 1902. Available at *http://members.door.net/arisbe/menu/library/bycsp/L75/L75.htm*.

77. A. Pentland, "Smart rooms, desks, and clothes," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Munich, Germany), April 1997.

78. A. P. Pentland, "Smart rooms," *Scientific American* **274**, pp. 68–76, April 1996.

79. M. J. Petrovsky, *Optimizing Bandwidth*, McGraw-Hill, New York, 1998.

80. K. R. Popper, *The Logic of Scientific Discovery*, Hutchinson, London, 1959.

81. J. O. Prochaska and C. C. DiClemente, "Stages of change in the modification of problem behaviors," *Progress in behavior modification* **28**, pp. 183–218, 1992.

82. T. F. Rathbun, *Autonomous construction of multi layer perceptron neural networks*. PhD dissertation, Air Force Institute of Technology, 1997.

83. B. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1995.

84. S. K. Rogers, M. Kabrisky, K. Bauer, and M. E. Oxley, "Computing machinery and intelligence amplification," in *Proceedings of 2002 World Congress on Computational Intelligence*, (Honolulu, HI), May 2002.

85. S. K. Rogers, M. Kabrisky, D. W. Ruck, and G. L. Tarr, *An introduction to biological and artificial neural networks*, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1990.

86. K. Rooney, ed., *Encarta World English Dictionary*, Bloomsbury Publishing, London, 1999.

87. F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Tech. Rep. 85-400-1, Project PARA, Cornell Aeronautical Lab, Ithaca, NY, 1957.

88. H. L. Royden, *Real Analysis*, MacMillan Publishing Company, New York, second ed., 1968.

89. D. W. Ruck, *Characterizations of Multilayer Perceptrons and their Application to Multisensor Automatic Target Detection*. PhD dissertation, Air Force Institute of Technology, 1990.

90. R. M. Satava, "Medical virtual reality: The current status of the future," in *Medicine Meets Virtual Reality 4: Health Care in the Information Age*, pp. 100–106, IOS Press, (Washington, D.C.), 1996.

91. D. Schmalstieg and M. Gervautz, "Implementing Gibsonian virtual environments," in *Cybernetics and Systems '96. Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research*, (Vienna, Austria), 1996.

92. A. Steinberg, C. Bowman, and F. White, "Revisions to the JDL data fusion model." presented at the Joint NATO/IRIS Conference, October 1998.

93. A. N. Steinberg, Technical Director for Data Fusion at Verdian ERIM International Inc. Interview, 24 May 2000.

94. M. R. Stytz, G. M. Godsell-Stytz, and E. G. Block, "Requirements and design of a virtual environment for emergency department physicians and personnel," in *IEEE Dual-Use Technologies and Application Conference*, pp. 19–27, (Utica, NY), May 1994.

95. Z. Szalavari, D. Schmalstieg, A. Fuhmann, and M. Gervautz, "Studierstube: an environment for collaboration in augmented reality," Tech. Rep. TR-186-2-96-17, Vienna University of Technology, 1996.

96. C. L. Thomas, ed., *Taber's Cyclopedic Medical Dictionary*, F. A. Davis, Philadelphia, 17th ed., 1993.

97. J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, MA, 1974.

98. J. Triesch and C. von der Malsburg, "Self-organized integration of adaptive visual cues for face tracking," in *Proceedings of SPIE, Sensor Fusion: Architectures, Algorithms, and Applications IV*, **4051**, pp. 397–406, (Orlando, FL), April 2000.

99. A. M. Turing, "Computing machinery and intelligence," *Mind* **59**(236), pp. 433–460, 1950.

100. A. Waibel, B. Suhm, M. T. Vo, and J. Yang, "Multimodal interfaces for multimedia information agents," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Munich, Germany), April 1997.

101. W. D. Wells, "Collaborative workspaces within distributed virtual environments," Master's thesis, Air Force Institute of Technology, 1996.

102. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(7), pp. 780–785, 1997.

103. H. Zhu and R. Rohwer, "No free lunch for cross-validation," *Neural Computation* **8**, pp. 1421–1426, 1996.

Vita

Amy L. Magnus was born in Morristown, New Jersey. Raised in Upstate New York, she attended Tioga Central High School where she graduated valedictorian in 1985. Amy attended the Rochester Institute of Technology (RIT), Rochester, New York for her undergraduate studies where—as part of her cooperative studies—she was an Engineering Assistant at RIT's Center of Imaging Science supporting graduate research in remote sensing. Receiving the Bachelors of Science degree in Electrical Engineering, Amy graduated magna cum laude from RIT on 19 May 1990, the same day she was commissioned into the United States Air Force. On 15 January 1991—one day before the Gulf War began—Amy entered active duty as a lieutenant working for the Defense Support Program (DSP) at the Space and Missile Systems Center, Los Angeles Air Force Base, CA. Highlights of her first assignment included the launching of a DSP satellite from a NASA space shuttle and the delivery of a new information fusion system Multi Message Fusion, the software that enabled Space Command to detect scud missiles during the Gulf War.

In 1994—the year she pinned on Captain—Amy was assigned to Wright-Patterson AFB to attend the Air Force Institute of Technology. She receive the Masters of Science in Electrical Engineering in December 1995 and was invited to extend into the institute's PhD program. Accepting AFIT's invitation, Amy built on her experience in image processing and information fusion and, in 1997, began her research into inquisitiveness. Outside of her research, Amy took the opportunity to work on two dream projects—first, as Art P.A. on the feature length film "The Dream Catcher" and, secondly, as singer-songwriter in Huge, an acoustic band with fellow AFIT PhD candidates Rob Pope and Jeremy Holtgrave. The year 1999 was a particularly exciting time for these projects. The film garnered Best Director at the Los Angeles Independent Film Festival, and the band Huge recorded an 15-track album (and got airplay on the band's favorite station WYSO).

In 1999, Amy was assigned to the Information Directorate, Rome Research Site, Rome, New York. Here, she continued her research for the Information Technology Branch as Chief of Research for Information Superiority; and, as her work garnered interest at the laboratory and beyond, she was offered a research position as Visiting Scientist at Cornell University's

Computer Science Department. In 2001, Amy pinned on Major and accepted a regular commission into the Air Force. As of August 2002, Major Amy Magnus is the Program Manager for Weapon Systems Survivability at the Defense Threat Reduction Agency where she oversees research efforts in information assurance and fault tolerance for low-powered integrated circuits, devices sensitive to single event upset. Following the completion of the PhD program, Major Magnus plans to teach a course on nuclear effects and to continue her research into arrogance and the Inquisitive Test through her affiliations with AFIT and Cornell University.

Permanent address: 399 Roki Blvd
Nichols, NY 13812

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
14-03-2003	Doctoral Dissertation	May 1998 - Mar 2003

4. TITLE AND SUBTITLE
Inquisitive Pattern Recognition

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Amy L. Magnus, Major, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Dept of Electrical and Computer Engineering
2950 P Street, Bldg 640
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/DS/ENG/03-09

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Information Directorate
525 Brooks Rd
Rome, NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/IFT

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

New pattern recognition concepts are presented in the area of classification, and new techniques are given that allows one to determine when a classifier is being arrogant. We quantify the balance between arrogant and non-arrogant classification and show how to detect arrogance. Inquisitive pattern recognition (IPR) is the constructive investigation and exploitation of conflict in information. This research defines inquisitiveness within the context of self-supervised machine learning and introduces mathematical theory and computational methods for quantifying incompleteness, that is, for isolating unstable, nonrepresentational regions in present information models. The key methods of inquisitiveness presented are (1) falsification and (2) the classification of confusion in feature space. This work also introduces a functional model for persistent learning and a simplified model for data fusion tailored to the development of pattern recognition algorithms. Artificial neural network demonstrations are provided to illustrate inquisitive pattern recognition techniques. IPR is a relational reasoning capability that allows computers to learn from imperfect decisions. Air force applications are discussed.

15. SUBJECT TERMS
Inquisitive pattern recognition, arrogant classification, confusion, intelligent computing, generalization capability, combinatorial geometry

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mark E. Oxley, AFIT/ENC Mark.Oxley@afit.edu
U	U	U	UU	237	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636 ext 4515