SHARP, RELIABLE PREDICTIONS USING SUPERVISED MIXTURE MODELS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Howard Scott Roy

March 1995

I certify that I have read this dissertation and that in my opinion it is fully adequate in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Michael Genesereth (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

David Rumelhart

I certify that I have read this dissertation and that in my opinion it is fully adequate in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Peter Cheeseman

Approved for the University Committee on Graduate Studies:

Abstract

This doctoral dissertation develops a new way to make probabilistic predictions from a database of examples. The method looks for regions in the data where different predictions are appropriate, and it naturally extends clustering algorithms that have been used with great success in exploratory data analysis. In probabilistic terms, the new method looks at the same models as before, but it only evaluates them for the conditional probability they assign to a single feature rather than the joint probability they assign to all features. A good models is therefore forced to classify the data in a way that is useful for a single, desired prediction, rather than just identifying the strongest overall pattern in the data.

The results of this dissertation extend the clean, Bayesian approach of the unsupervised AutoClass system to the supervised learning problems common in everyday practice. Highlights include:

- clear probabilistic semantics
- prediction and use of discrete, categorical, and continuous data

- priors that avoid the overfitting problem
- an explicit noise model to identify unreliable predictions
- the ability to handle missing data

A computer implementation, MultiClass, validates the ideas with performance that exceeds neural nets, decision trees, and other current supervised machine learning systems.

The dissertation is written for a general audience with many, many examples to motivate the new ideas. The scope of potential applications is very large, including problems like evaluating student admissions applications, assessing credit risk, and identifying customers likely to order from Tiffany's latest Christmas gift catalog.

Acknowledgements

"Hey Adam—it's done!"

. . . and what a ride it's been. I never would have made it without the love, encouragement, and ever so gentle prodding of all my wonderful family and friends. Few people seem as fortunate as I am with the people I'm surrounded with in my life. Mom and dad: I'll be home soon! Marc and Jodi, find some Bulls tickets somewhere so we can watch His Airness lead the Bulls past the Knicks yet again. And Sheryl, make sure that Josh remembers his uncle! I can't wait to get back to Chicago and see you all again, along with the entire "holiday gang" of aunts, uncles, and family friends. I've missed you all very much.

A special thanks to Adam and Becky, and to David and Laurel. Maybe now I'll actually get to spend some time with you, though I can't help but wish that we were all in the same city instead of scattered between Chicago and San Francisco as we are. And thanks to all my other close friends who have been just great when I needed them: Steve and Jennifer, Jon, Bobby (both of you), Jason, Maia, Wray, Linda, and LeAnn. Don and Narinder, I'll miss our daily lunch pilgrimage up to Tressider and the always entertaining lunchtime conversation. Oh, and Don:

I should be back in mid-April. Is Great America open then? :-) Seriously, finish the damn proof and graduate already. (Ha! It feels great to be on the other side of *that* fence.)

In a more serious vein, on the academic front I'd like to sincerely thank all my teachers through the years. This moment is the pinnacle of my life as a student, and every one of you has helped me at least a little bit along the way. A big thanks to Peter Cheeseman, who started me on the road to this dissertation with his memorable lecture five years ago. And another big thanks to my advisor, Mike Genesereth, for being helpful and patient despite all the bumps. And a third big thanks to David Rumelhart for taking the time and trouble to be on my reading committee. The members of the Logic Group have been the toughest sounding board I could possibly imagine, but I've enjoyed knowing every one of you. And thanks also to the Bots crowd and the PDP group. I've felt like a schizophrenic yo-yo caught between such different groups, but it's time to unwind at last and go relax in the bleachers at Wrigley.

Hey, you think they'll be playing baseball?

H. Scott Roy
March 17^{th}, 1995

Contents

List of Figures

1
The Problem and the Approach

I. The Problem

This dissertation describes how to make sharp, reliable inferences from a database of examples. Figure 1 presents a typical application, a fictitious database containing information about students and schools. The data is laid out in multiple tables in the standard format common to relational and object oriented databases. Given such a database, there are two problems this thesis looks to solve:

1) Predicting missing observations.
2) Validating recorded observations to identify those that are dubious, or unusual.

An observation is the information at a row and column intersection in one of the tables. For example: “The student with the name Woo scored 780 on his GRE,” or “It’s unknown how

STUDENTS

Name	Undergraduate School	GPA	GRE	References	Interest	···	Years	Papers
Woo	Harvard	3.8	780	Excellent	AI		6.5	1
Pippen	Chicago	3.5	800	Good	Theory		4.0	3
Jones	Iowa	3.2	?	Fair	Theory	···	5.5	5
Geddis	Stanford	4.0	780	Excellent	OS		?	0
Roy	Harvard	3.6	670	Poor	Graphics		?	?
Smith	?	3.1	610	Good	AI		?	?
							...	...

SCHOOLS

Name	Enrollment (k)	Public?	Average GPA
Harvard	6.4	-	2.8
Chicago	5	-	2.7
Iowa	25	X	2.1
Stanford	7	-	3.5
Columbia	10	-	3.0
...	...	...	...

Figure 1. *The Student Database*

long Roy took to earn his graduate degree." Observations can be missing for any number of reasons, like lost forms, forgotten test scores, or deliberate censorship. Quite often they simply indicate future, unknown events. Mr. Roy, for example, is likely still a student in the Stanford Ph.D. program, or else he is an aspiring applicant for whom the admissions committee is keenly interested in predicting how long he would take to finish if he were admitted.

The inference problem presented here includes the most familiar case of supervised machine learning: predicting a single column in a table from the other columns in that table. An admissions committee looking at the database of Figure 1, for example, might like to predict how long students like Mr. Roy will take to graduate on the basis of their admissions information. But the committee might like to answer any number of questions:

Do AI students take longer to graduate than theory students?

> Is Stanford losing top students to MIT?
>
> Are students that want to study software engineering avoiding Stanford?

... and so forth. All of these questions fall within the scope of the methods set forth in this dissertation.

II. The Approach

Probability theory formalizes the problem in an elegant way. Suppose we want to predict a particular observation in the student database, like Mr. Roy's time to graduate from Stanford. Then the mathematical question we seek to answer is,

> Given all the observations in the database, together with all our prior information regarding students and schools, what probability should we assign to the proposition, "Mr. Roy will graduate in n years or less?"

In symbols, our goal is to assess $\Pr(x \leq n \mid DI)$, where x is the number of years Mr. Roy will take to graduate, D is the evidence of the database, and I is our prior information. Prediction, validation, and all the other questions presented in the first section ultimately reduce to calculating one or more probabilities of this form, where the proposition "$x \leq n$" is replaced by one appropriate to the question. To validate that Mr. Woo's GRE score is reasonable, for example, we would imagine that it were missing and compute the probability distribution over the different GRE scores he might have obtained. We can then decide whether the actual score of 780 has an acceptably high probability.

Probabilities come from models. Just as a scientist builds a model to explain empirical facts about the world, so too one analyzes a database by building a model that explains the systematic patterns in the recorded observations. One then uses that model to make predictions. In the probabilistic framework the definition of a model is simple and precise: a model is any function that assigns a definite, computable probability to every possible database one might observe. We write this function using the conditional probability notation, $\Pr(D \mid MI)$.

In Bayesian induction, one computes the probability of a proposition Q by taking a weighted sum over as many different models M as one cares to consider, each of which makes its own predictions:

$$\Pr(Q \mid DI) = \int \Pr(Q \mid MDI) \Pr(M \mid DI)\, dM \tag{1}$$

Throughout this dissertation, the reader should understand that all integrals implicitly collapse to sums when the domain is discrete. The second factor in the integral of Equation 1, the posterior distribution $\Pr(M \mid DI)$, is the weighting factor that indicates how strongly we believe each model after having seen the database. We evaluate it using Bayes' theorem:

$$\Pr(M \mid DI) = \frac{\Pr(D \mid MI) \Pr(M \mid I)}{\Pr(D \mid I)} \tag{2}$$

Bayes' theorem shows how to update the prior distribution $\Pr(M \mid I)$ to the posterior one by incorporating the new information contained in the database. The factor in the denominator, $\Pr(D \mid I)$, is often called the evidence for the models, or the Bayes factor. It plays an important role when one allows the prior information I to vary, as occurs when considering different model space alternatives [38, 33], but for the purpose of this dissertation it is a fixed normalizing constant.

Together, Equations 1 and 2 completely specify Bayesian probabilistic induction. Jaynes's *Probability: the Logic of Science* is an exceptional introduction [28], as is the new book by Bernardo and Smith [2].

The posterior distribution, $\Pr(M \mid DI)$, indicates which models provide the best description of the database. More generally, as Equation 1 shows, we ideally make predictions by averaging the predictions of all the models we consider, weighted according to their posterior probabilities. In practice this approach is a pipe dream. The models considered in this dissertation are sufficiently complex that Equation 1 has no analytic solution, and we shall be forced to approximate it by making predictions using the model of maximum posterior probability, a process known as MAP induction. This approach works well as long as the posterior distribution is dominated by a single sharp peak, indicating that the combination of data and prior information overwhelmingly favors one model over the others. It is easy to replace MAP induction with more accurate but time consuming approximations to Equation 1, like Laplace's normal approximation [1] or Gibb's sampling [15], but this dissertation leaves such extensions to future work.

The complete MAP algorithm for the database modeling problem is as follows:

1) Choose a model space.
2) Assign a prior probability to every model in the space.
3) Find the model of maximum posterior probability.
4) Answer all questions using that model.

MAP induction fits nicely with a fundamental lesson of machine learning:

<u>Lesson #1</u>

Learning is search.

This search takes the form of graph search over the discrete parts of the model space, and continuous function optimization over the continuous parts. We have at our disposal all the tools of artificial intelligence on the one hand, and all those of operations research on the other.

Equation 1 tells us that Lesson #1 is wrong. Learning is not search at all, but rather summation. It is only when we reach the limit of our ability to do exact calculations that we fall back on approximations like MAP induction, in which we are simply finding the largest summand in Equation 1. Only when the posterior distribution is sharply peaked are the two approaches effectively equivalent.

Mitchell wrote down Lesson #1 for the machine learning community in 1977, in his work on version space learning [42]. It no doubt has a considerably longer history in the field of philosophy. But it is worth continually reemphasizing, since it seems so easy to conflate a model space with the algorithms that search it. It took years, for example, to firmly disentangle the BackProp algorithm from the idea of a feedforward neural network, and only recently have people begun to look at alternative optimization algorithms, such as conjugate gradient methods and quasi-Newton optimizers. The decision tree literature, similarly, seems to only periodically rediscover that one can search the space of decision trees using anything other than a greedy hill climbing algorithm. A 2-ply lookahead gives much better performance [5], and the problem seems natural for a general graph search algorithm like A*.

Some search algorithms are designed to work with particular model spaces, but for the most part the separation between model spaces and algorithms is clean, so that one has two orthogonal ways to improve the quality of data analysis: use a better search algorithm, or use a better model space. Improving the search algorithm is almost always much less important than improving the model space, since even the best model in the wrong space is likely to be much worse than a mediocre model in the right one. In describing the transition between successive versions of AutoClass, the program out of which this dissertation arises, Hanson and his coworkers relate that improving the model space brought orders of magnitude more improvement than tinkering with the search algorithm [19]. Chapter 2 will cement this lesson by showing numerous examples where an inadequate model renders sharp predictions impossible.

The wise practitioner never forgets:

Lesson #2

You find what you look for.

So you had better make certain you look for the right thing.

In Bayesian probability, Lesson #2 is reflected in the role of prior probabilities, which receive equal weight to the likelihood in Bayes' theorem. This dissertation focuses on setting up a good model space and prior with which to ask questions about large, poorly understood databases.

III. Organization and Contributions

The remainder of this thesis is as direct as possible, while attempting to remain accessible to the general reader who is not an expert in machine learning. Chapter 2 gives a progression of example databases to illustrate the major ideas and the shortcomings of the AutoClass system that motivate the dissertation. Chapters 3–6 develop model spaces and priors to address these shortcomings, and the last chapter wraps up with experimental results and future directions.

The major contributions of the dissertation are,

1) A new version of AutoClass, MultiClass, that corrects its limitations on supervised prediction problems while retaining its clear probabilistic semantics.
2) A mapping between hidden features and logical rules that provides insight into mixture models and delineates their expressive power.
3) A multiple hidden feature model space that generalizes unsupervised mixture models to account for all the useful ways to classify objects in a database.
4) A way to explicitly model noise and errors in a database to automatically recognize and eliminate unreliable predictions.

2
Example Databases

I. Understanding the Ideas

This chapter presents a series of example databases to illustrate the major ideas that motivate the dissertation. There is no math. The informal style is intended to convey the intuition and simplicity of the ideas without obscuring them behind formulas, or tying them to any one mathematical framework. The general themes that run through the chapter—hidden features, supervised vs. unsupervised analysis, and reliability—are easily understood without any greek magic.

This chapter is not a critique of existing methods. It is, instead, a map of the potentially treacherous terrain that all data analysis systems must navigate. Existing programs deal with many of the problems in this chapter just fine, but no current program handles all of them. The one system this chapter does frequently mention is AutoClass [9], since it is the limitations of AutoClass that this dissertation primarily looks to understand and overcome.

II. Iris Database

Chapter 1 described the ideal input to a machine learning algorithm, namely an existing object oriented or relational database laid out in multiple tables. The remainder of this dissertation focuses on the simpler, special case in which the data are stored in only a single table, the format universal to current machine learning research. The final chapter returns to the issue of multiple tables, showing both their importance and the ease with which the ideas in this dissertation can be extended to include them.

This dissertation begins with the idea of using hidden features to describe clusters and correlations in observable data. Figure 1 illustrates the intuition with a slice from the well known Iris data set of R. A. Fisher. The plot shows clear evidence for at least two kinds of flowers. There are actually three: the smaller Iris Setosa, and the larger Iris Virginica and Versicolor. Keep in mind that a computer, lacking eyes, sees only a table of numbers:

Sepal Length	Petal Length
5.1	1.4
5.4	1.7
7.7	6.9
6.7	4.7
4.8	1.6
. . .	. . .

As our eyes reveal from looking at the plot, it is natural to partition the flowers into two groups, one for the cluster in the lower left corner, and one for the cluster in the upper right. We record this classification by augmenting the original database with a new feature column indicating the cluster into which each flower falls:

Sepal Length	Petal Length	Cluster
5.1	1.4	Bottom
5.4	1.7	Bottom
7.7	6.9	Top
6.7	4.7	Top
4.8	1.6	Bottom
. . .	. . .	. . .

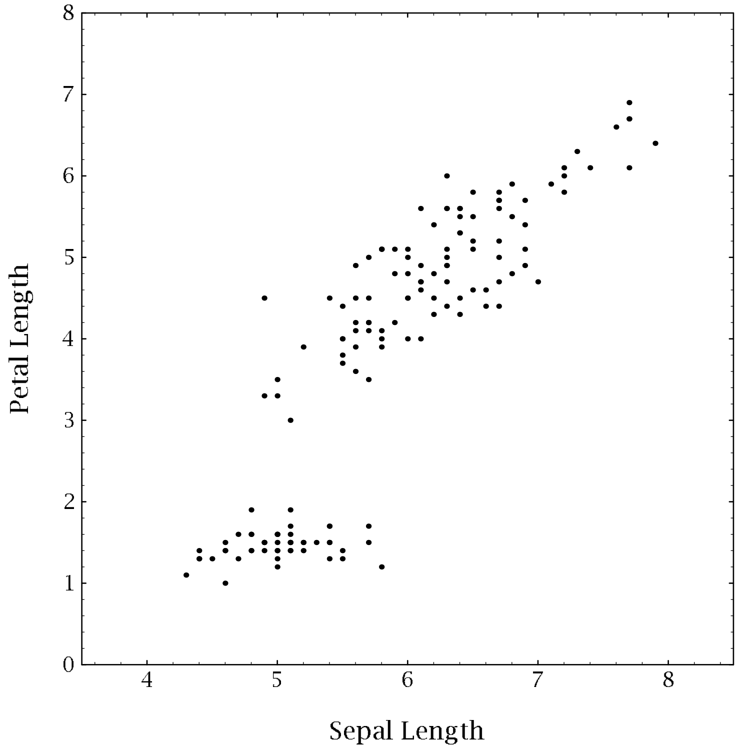


Figure 1. *The Iris Database*

We call the “Cluster” column a hidden feature, since although it is purely hypothetical and cannot be measured, one can imagine it as a real feature missing from the original database. In this case it indicates the species of flower: all the flowers in the lower left cluster are Iris Setosa, while those in the upper right are Virginica and Versicolor. If we apply the same analysis to the complete iris database, which includes width measurements in addition to the length ones, three distinct clusters emerge that let us recover all three kinds of flowers.

In general, a hidden feature indicates a causal effect inducing a pattern in the observable data. In a medical database, hidden features that describe clusters of symptoms may correspond to new syndromes or diseases, previously unknown to doctors. A famous example of machine discovery involved an analogous problem, in which the AutoClass program, performing an analysis much like the one suggested by the iris example, discovered new types of stars by spotting patterns in their spectral emissions [9].

III. Day Camp Database

Clusters can appear in discrete and categorical data as well as in real data. Here is an example, a database that shows the daily record of activities at a children's day camp:

Swimming	Whiffle	Coloring	Singing	Painting
×	×	–	–	–
–	–	×	–	×
×	–	×	×	–
×	×	–	–	–
–	–	–	×	×
–	–	×	×	×
...	...	...	...	...

Each row is a day, and each observation indicates whether the children undertook a particular activity. For example, on the first day there was swimming and whiffle ball, but no coloring or singing or painting. There is a clear pattern. The two outdoor activities, swimming and whiffle ball, occur together, as do the three indoor activities. The weather doubtless has a significant impact on the daily schedule, so we can improve our picture of the data by imagining the weather as an additional, hidden feature,

Swimming	Whiffle	Coloring	Singing	Painting	Weather
×	×	–	–	–	sunshine
–	–	×	–	×	rain
×	–	×	×	–	sunshine
×	×	–	–	–	sunshine
–	–	–	×	×	rain
–	–	×	×	×	rain
...	...	...	...	...	...

When a computer finds such clusters, of course, it still takes a human to assign a meaningful label like "Weather" to the patterns that it finds.

IV. United Nations Database

Many existing learning frameworks, including decision trees and mixture models, effectively model data using a single hidden feature, as in the iris and day camp examples. This approach is powerful, but one must be careful when applying such models to predictions and validations. There may be many equally good ways to classify the objects in a database, but not all of them are necessarily relevant to the questions one cares about.

Suppose, for example, that one were to collect the physical characteristics of delegates to a United Nations conference on world population. A small portion of the resulting database might look like this:

Height	Weight	Skin	Eyes	Hair
5'11"	180	white	brown	brown
4'10"	105	white	blue	blonde
5'4"	130	black	brown	black
5'3"	120	brown	brown	brown
6'2"	195	white	blue	red
. . .	. . .	. . .	. . .	. . .

There are at least two different ways to classify the people in this table. The height and weight measurements fall into two distinct, overlapping clusters, since men are both taller and heavier than women, so one way to classify people is according to sex. There are also strong correlations among skin color, eye color, and hair color, so a second way is according to ethnicity. It is therefore useful to imagine the database with two hidden features:

Height	Weight	Skin	Eyes	Hair	Sex	Ethnicity
5'11"	180	white	brown	brown	M	European
4'10"	105	white	blue	blonde	F	Scandinavian
5'4"	130	black	brown	black	F	African
5'3"	120	brown	brown	brown	F	African
6'2"	195	white	blue	red	M	Irish
. . .	. . .	. . .	. . .	. . .	. . .	. . .

The presence of two hidden features portends serious trouble for a program, like AutoClass, that hopes to describe all the effects in the database using only a single one. This database requires two hidden features, and a program that looks for only one will inevitably miss a large part of the structure in the data.

If one does use a program that looks for only a single hidden feature, it is imperative that it work in a supervised fashion where it knows which predictions are important, so that it can tell which of the many possible hidden features it should choose for its model. Lacking such detailed instructions, an unsupervised program will always describe the most dominant structure it can find. Only blind luck will decide whether that structure is relevant to the predictions one cares about. In the United Nations data, for example, an unsupervised program might easily decide to focus on sex when the goal is to predict eye color, or on ethnicity when the goal is to predict height. Either way, making predictions from the final model will be no better than throwing darts, since the structure it reveals will be irrelevant to the desired predictions.

It is possible, in principle, to combine multiple features like "Sex" and "Ethnicity" into a single hidden feature that embodies their crossproduct. But this solution quickly arrives at a severe overpartitioning problem as the number of useful classifications grows. A single crossproduct feature must enumerate an exponentially increasing number of classes, and there is almost never enough data to go around. Chapter 5 gives a full quantitative account of this phenomenon.

V. Thyroid Database

The original developers of AutoClass encountered the problem of multiple useful classifications on many data sets [8]. Quinlan's thyroid disease database is a good example. The goal with this database is to diagnose various thyroid diseases and prescribe treatment therapies. A very small portion of the data looks like this:

Sex	Pregnant?	⋯
M	–	
F	×	
F	–	
M	–	⋯
F	–	
F	×	
⋯	⋯	

There is a very strong connection between sex and pregnancy, since only women can be pregnant. AutoClass finds this pattern of great interest and settles on the rather dull hidden feature,

Sex	Pregnant?	⋯	Pregnant Woman?
M	–		–
F	×		×
F	–		–
M	–	⋯	–
F	–		–
F	×		×
⋯	⋯		⋯

Whatever one hopes to learn from this data set, this clearly is not it.

Unfortunately, all the classes that AutoClass finds in the thyroid data are of little importance or already known, like the pregnant women class, so that the final model performs no better than guesswork on the desired diagnoses. Cheeseman often tells a similar anecdote, in which AutoClass finds many fascinating social classes in a medical database, but ignores the much more interesting—albeit weaker—correlations among symptoms and diseases.

VI. Student Database

The thyroid database suggests that hidden features often have a natural interpretation as logical rules. Chapter 5 explores the connection and shows that this perception is correct. One common way for such rules to appear is when one flattens multiple tables into a single table by expanding functional relations. Unfortunately, one is often forced to follow this approach because of the inability of current machine learning systems to deal with multiple database tables.

Logical rules appear in the student data when one folds the information about schools directly into the student table like this:

School	Public?	Enrollment	···	GRE	Years	Papers
Harvard	–	6.4		780	6.5	1
Chicago	–	5.0		800	4.0	3
Iowa	×	25.0	···	?	5.3	5
Stanford	–	7.0		780	?	0
Harvard	–	6.4		670	?	?
...	...	...		...	...	...

This approach is less than perfect, since it fails to convey that some columns in the flattened database, like "Public?" and "Enrollment," are functionally dependent on others, in this case "School." Enrollment is always 6400, for instance, for every student from Harvard. The resulting perfect correlations can lead to trouble for an unsupervised system that becomes enamored rediscovering them.

VII. Prescription Database

Although a database can exhibit many, many patterns that are irrelevant to the predictions one cares about, even silly patterns, like the fact that only women can be pregnant, can be crucially important in spotting database errors that might lead to unreliable predictions.

Unreliability is an aspect of the traditional expert system problem known as brittleness, a sharp degradation of skill outside the limited range of expertise for which a system is designed. Brittleness can be either an annoyance, when a system cannot make use of common sense information, or a danger, when a system fails to recognize its own limitations, and users are either too naive or incautious to reject its conclusions. Lenat and Feigenbaum cite disturbing examples: a medical system issuing absurd prescriptions to a patient whose weight and age are accidentally interchanged, a loan authorization program approving an enterprising teenager claiming to have worked for over 20 years, and others [37].

Problems like these arise when an expert system fails to validate the things it is told. Fortunately, an expert system induced directly from a database of examples has access to all

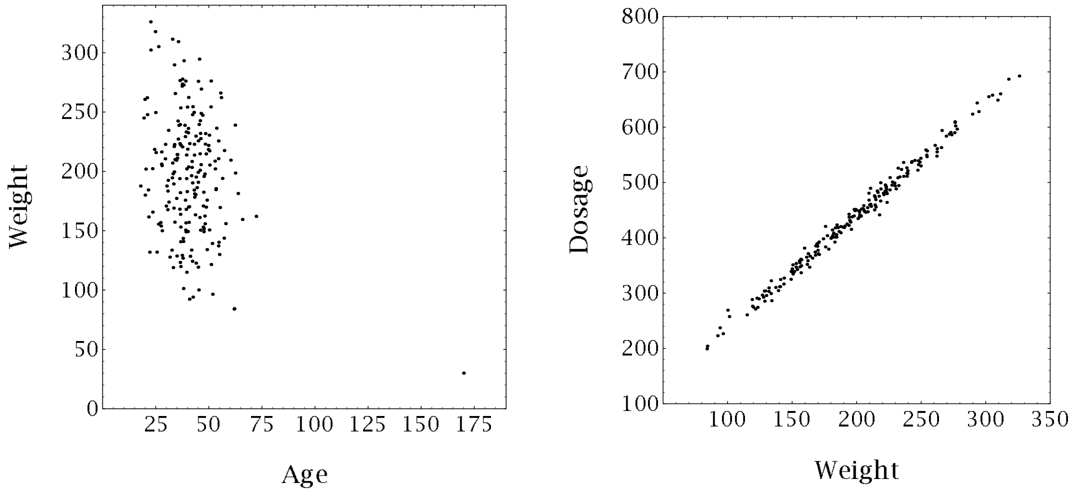


Figure 2. *The Prescription Database*

the information it needs to avoid such mistakes. The medical expert system described above, for example, might be created from a database that includes a fragment like the following:

Age	Weight	Dosage (mg)
20	180	400
34	210	460
55	130	300
28	150	340
170	30	?
...	...	...

The task is to assess the dosage for the last patient. Figure 2 plots this data so that one can see what is happening. Weight and dosage have a tight linear relation, while age and weight form a visible cluster centered around 45 years and 150 pounds. A traditional expert system, or a supervised learning system that only cares about patterns useful for predicting dosage, will assess a dosage for the final patient somewhere around 100mg, without any warning bells or whistles.

But the database obviously contains all the information needed to recognize the unreliability of this prescription, since the final row is so different from all the others. In the joint plot of age and weight, the final entry falls far outside the envelope of the others. A system trained to look for all the patterns in the data will notice this fact and sound an alarm, or at the very least refuse to make a sharp prediction.

Reliability is complimentary to accuracy, robustness, sensitivity, and other usual ways of measuring the quality of a prediction. It looks at how similar conditioning information is to the data from which a model was constructed. A model is unreliable whenever it has the potential to make sharp predictions in situations that are highly unlike any it has seen.

VIII. Movie Database

In many real world problems missing data is the norm rather than the exception. A doctor, for example, will typically only run a tiny fraction of the lab tests he has at his disposal when diagnosing a patient, so his medical records will be filled with missing observations. As a more fun example, a video rental store might try to recommend new movies based on questionaires in which customers list movies they like and dislike.

Star Wars	ET	Alien	Singing in the Rain	Sound of Music
9	8	10	?	3
?	7	9	?	?
?	?	2	?	?
2	?	?	9	10
?	3	4	8	9
...	...	...	...	...

Each row gives a person's ratings for movies he has seen. The first person, for example, enjoys science fiction but is less excited by musicals. The goal is to predict whether or not *Singing in the Rain* is something he would like. There is no hope of acquiring complete data in this domain, even for customers that are dedicated moviegoers. Applications like this one are impossible unless a machine learning program can handle missing data.

IX. Sharp, Reliable Predictions

The example databases reveal the challenges that confront a system hoping to make sharp, reliable predictions from the evidence contained in a database. To summarize:

- Hidden features classify data in ways that explain clusters and correlations.
- There are often many interesting classifications.
- Supervised learning can focus a system on the relevant ones.
- All patterns become important when trying to identify unreliable predictions.
- Missing data is a reality.

This dissertation grows out of the AutoClass system, which performs an unsupervised analysis using a single hidden feature, as suggested by the iris and day camp examples. The United Nations database and those following it reveal why this simple approach falls. A single hidden feature model is too weak to account for all the interesting classifications, and an unsupervised approach gives a system no way to decide which classifications will be useful. The intent of this dissertation is to correct these problems in a clear probabilistic way.

3

Mathematical Preliminaries

I. Formulation

This chapter fleshes out the principles of Bayesian induction, sets up the database analysis problem as one of density estimation, and precisely defines the terms “sharpness” and “reliability.” At this stage the formulas are all universal, independent of any specific model. Later chapters specialize them to the hidden feature models that are the focus of this dissertation.

The presentation skips any pedantic discussion along the lines of, “What is a database?”, trusting that the reader’s understanding is sufficient to reconstruct the obvious formalization. There are only two details that are worth emphasizing. First, features come in all types: continuous, like the width of a flower petal; discrete, like the number of years a student takes to graduate; and categorical, like a person’s eye color. One of the main design goals of the models in this dissertation is to treat every feature for what it is, without being forced to lump them all together into a single universal type.

Second, there is no restriction that the rows of a table are unique. Although each row corresponds to a distinct entity in the world, two entities can easily be indistinguishable based only on the observations in a database. Two students, for example, might coincidentally submit applications that are identical when reduced to the few features recorded in the student table.

II. Density Estimation

The rows of a database table are naturally viewed as points in an n dimensional space, with the i^{th} component drawn from the possible observations for the i^{th} feature together with the symbol '?'. We will always use the small letter $\mathbf{z}$ for points in this space, and the letters $\mathbf{x}$ and $\mathbf{y}$ for points in subspaces and projections. D, the totality of observations in a database, is equivalent to the set $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, where N is the number of rows in the database.

The models in this dissertation all treat the rows of a table as independent samples from a fixed distribution. This independence lets one write the likelihood of a database as a product of individual factors, one for each row:

$$\Pr(D \mid MI) = \prod_{i=1\ldots N} \Pr(\mathbf{z}_i \mid MI) \tag{1}$$

Treated as a function of $\mathbf{z}$, $\Pr(\mathbf{z} \mid MI)$ gives the density with which each possible row of observations appears.

In the student database, the likelihood of seeing the observed table is the product of separate likelihoods for Woo, Roy, Geddis, and the others. The order is irrelevant. Under the hypothesis of any particular model, each student is independent from all the others, so that the likelihood of student 1000 is unaffected by the previous 999. Learning is possible because we do not assume a fixed model, but instead maintain a distribution over a set of models. This distribution adjusts as we consider each new student so that it favors models that give a good description of the data. As the distribution becomes more and more sharply peaked, we learn less and less from each new student we see.

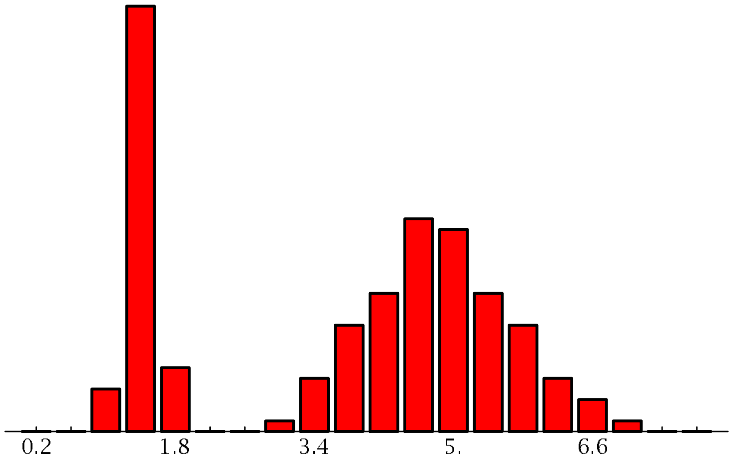


Figure 1. *Petal Lengths in the Iris Data*

For instance, we might decide to model GRE scores using a normal distribution with a flat, uninformative prior over the mean μ. After seeing 1000 students, the posterior distribution will be tightly centered on models for which μ agrees with the sample mean $\overline{\text{GRE}}$ measured from the data. Although the data picks out a very precise model in the space of normal distributions, that model may be a terrible fit to the actual data. Figure 1 plots the histogram of petal lengths in the iris data, for which one gets a sharp posterior in the space of normal distributions, but a very poor model.

In the general case we search a model space parameterized by ϕ, a multidimensional quantity that can contain any number of continuous, discrete, and categorical parameters, so that every choice of ϕ denotes a distinct model. It is perfectly acceptable for two different values of ϕ, denoting different models, to nevertheless generate identical probability distributions. If a row in a database table contains no '?', then its joint probability conditioned on a model M_ϕ is,

$$
\begin{aligned}
\Pr(\mathbf{z} \mid M_\phi I) &= \frac{1}{Z(\phi)} k(\mathbf{z} \mid \phi) \\
Z(\phi) &= \int k(\mathbf{z} \mid \phi)\, d\mathbf{z}
\end{aligned}
\tag{2}
$$

The kernel function $k(\mathbf{z}|\phi)$ gives the probability distribution its shape, while the partition function $Z(\phi)$ normalizes everything to one.

A normal distribution over GRE scores has the form,

$$N(\text{GRE} \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{\text{GRE}-\mu}{\sigma})^2}$$

Mapping onto the symbols of Equation 2,

$$\phi = \{\mu, \sigma\}$$
$$k(\mathbf{z} \mid \phi) = e^{-\frac{1}{2}(\frac{\text{GRE}-\mu}{\sigma})^2}$$
$$Z(\phi) = \sqrt{2\pi}\sigma$$

For a larger section of the student database, we might decide to model observations using a separate point mass distribution for each discrete component, and a single multivariate normal for all the continuous ones. The likelihood function for the observations in a single row is,

$$\begin{aligned} \Pr(\text{recommendations}, \text{GRE}, \text{GPA}, \text{years}, \text{school} \mid M_\phi I) = \\ & M(\text{recommendations} \mid \mathbf{w}_1) \times \\ & M(\text{school} \mid \mathbf{w}_2) \times \\ & N(\text{GRE}, \text{GPA}, \text{years} \mid \boldsymbol{\mu}, \Sigma) \end{aligned}$$

The function M denotes a point mass distribution parameterized by the weight vector $\mathbf{w}$, while N is a multivariate normal with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ.

Linking Equation 2 to the current example, $\mathbf{z}$ is the vector of a student's recommendations, GRE score, GPA, years to graduate, and undergraduate school, ϕ is the union of the parameters $\mathbf{w}_1$, $\mathbf{w}_2$, $\boldsymbol{\mu}$, and Σ, and the partition function Z is absorbed into the component distributions so that it does not appear explicitly. In this particular case all the parameters are continuous, though the $\mathbf{w}_i$ must be nonnegative and Σ must be positive definite to ensure the overall distribution is well defined. The components of each $\mathbf{w}_i$ must also sum to 1.

One of the elegant aspects of density estimation is that it explains how to handle missing data. If a row in a database table contains a '?', we can marginalize to obtain the joint probability of all the observations that are not missing. Letting $\mathbf{x}$ be the part of $\mathbf{z}$ that is known, and $\mathbf{y}$ the part that is unknown, the joint probability is,

$$\Pr(\mathbf{z} \mid MI) = \int \Pr(\mathbf{yx} \mid MI)\, d\mathbf{y} \tag{3}$$

A flaw in this method is that it does not account for the possibility of censorship. If students deliberately withhold low test scores, for example, the fact that a score is missing is cogent information. It is easy to account for this possibility by adding features to the database that make missing observations explicit. With the student data, for example,

GRE	Missing GRE?	⋯
780	–	
800	–	
?	×	⋯
780	–	
670	–	
⋯	⋯	

Now we can marginalize away missing GRE scores and still notice if a missing score is significant.

Equations 2 and 3 specify $\Pr(\mathbf{z} \mid M_\phi I)$ regardless of whether $\mathbf{z}$ contains missing values. Plugging Equation 2 into Equation 1 gives the likelihood of a database table with no missing values in parameterized form:

$$\Pr(D \mid M_\phi I) = \frac{1}{Z(\phi)^N} \prod_{i=1\ldots N} k(\mathbf{z}_i \mid \phi) \tag{4}$$

The partition function $Z(\phi)$ factors out of the product since it is the same for each row of the database.

When the MAP model cannot be found by an exact analytic calculation, the search inevitably reduces to repeatedly evaluating trial models using Equation 4 and Bayes' theorem. The time complexity of calculating $Z(\phi)$ almost always dominates the computations, as it is a complex multidimensional integral for all but the simplest of kernel functions. It is only a slight exaggeration to say that all the difficulty of density estimation is tied to the difficulty of computing the partition function.

III. Database Queries

Chapter 1 presented the two chief formulas of Bayesian induction, the weighted sum rule that combines different predictions, and Bayes' rule that lets one evaluate models:

$$\Pr(Q \mid DI) = \int \Pr(Q \mid MDI)\Pr(M \mid DI)\, dM$$
$$\Pr(M \mid DI) = \frac{\Pr(D \mid MI)\Pr(M \mid I)}{\Pr(D \mid I)}$$

A minor detail that Chapter 1 omitted was how to evaluate $\Pr(Q \mid MDI)$, the probability of a query given a specific model and the recorded observations in the database.

A query is any statement whose truth could be determined from a complete database with no missing entries. For example, "Mr. Roy graduated in less than 5 years," or "Mr. Woo scored 780 on his GRE." Given a model M, the probability of a query Q is found by summing over all possible ways to fill in the missing entries in the database to yield a complete database D' in which Q is true:

$$\Pr(Q \mid MDI) = \int \Pr(D' \mid MI)\, dD' \tag{5}$$

Equation 5 is general and covers any possible model. In the case of density estimation, if the query Q refers to just a single row, as do the examples above, then all the other rows in the database are irrelevant and one obtains a simpler formula. The integral now spans $\mathbf{z}'$, all the completions of the query row $\mathbf{z}$ for which Q is true:

$$\begin{aligned}\Pr(Q \mid MDI) &= \Pr(Q \mid \mathbf{z}MI)\\ &= \int \Pr(\mathbf{z}' \mid MI)\, d\mathbf{z}'\end{aligned}$$

The dimensionality of this integral is equal to the number of missing values in $\mathbf{z}$. It presents no computational difficulties in the most common case where $\mathbf{z}$ contains only a single missing value, the feature we are trying to predict.

IV. Sharpness and Reliability

In supervised machine learning, the joint feature vector $\mathbf{z}$ breaks up into two parts: the known features $\mathbf{x}$, and the unknown features $\mathbf{y}$ that one is trying to predict. When evaluating student applications, for example, $\mathbf{x}$ is the information on the application and $\mathbf{y}$ is the unknown number of years the student will take to graduate and the unknown number of papers he will publish. The joint density factors in a meaningful way to reflect this breakdown:

$$\begin{aligned} \Pr(\mathbf{z} \mid MI) &= \Pr(\mathbf{yx} \mid MI) \\ &= \Pr(\mathbf{y} \mid \mathbf{x}MI)\Pr(\mathbf{x} \mid MI) \end{aligned} \tag{6}$$

The factor on the left, $\Pr(\mathbf{y} \mid \mathbf{x}MI)$, is the only part relevant to making predictions. Its uncertainty is a measure of how precise of a prediction the model can make, so it gives a crisp definition to the term "sharpness":

$$\text{Sharpness} \equiv \frac{1}{H\left(\Pr(\mathbf{y} \mid \mathbf{x}MI)\right)}$$

H is the entropy function. The sharpness of a model's predictions depends on the conditioning information $\mathbf{x}$. A model may make sharp predictions in one part of the input space and noncommittal predictions in another. Figure 2 plots examples of continuous probability distributions in the direction of increasing sharpness. Figure 3 does the same for discrete distributions.

The term "reliability" has two senses: one refers to the reliability of conditioning information, and one refers to the reliability of a model's predictions. The factor on the right of Equation 6, $\Pr(\mathbf{x} \mid MI)$, assesses the reliability of the conditioning information. It is most meaningful on a relative scale. When $\Pr(\mathbf{x} \mid MI)$ is large, it means that $\mathbf{x}$ closely resembles rows in the database, and one can have confidence when the model makes sharp predictions because they are backed up by recorded data. When $\Pr(\mathbf{x} \mid MI)$ is small the model is extrapolating and one should worry, since $\mathbf{x}$ is likely to be very different from the data from which the model was constructed. We therefore define,

$$\text{Raw Reliability} \equiv \Pr(\mathbf{x} \mid MI)$$

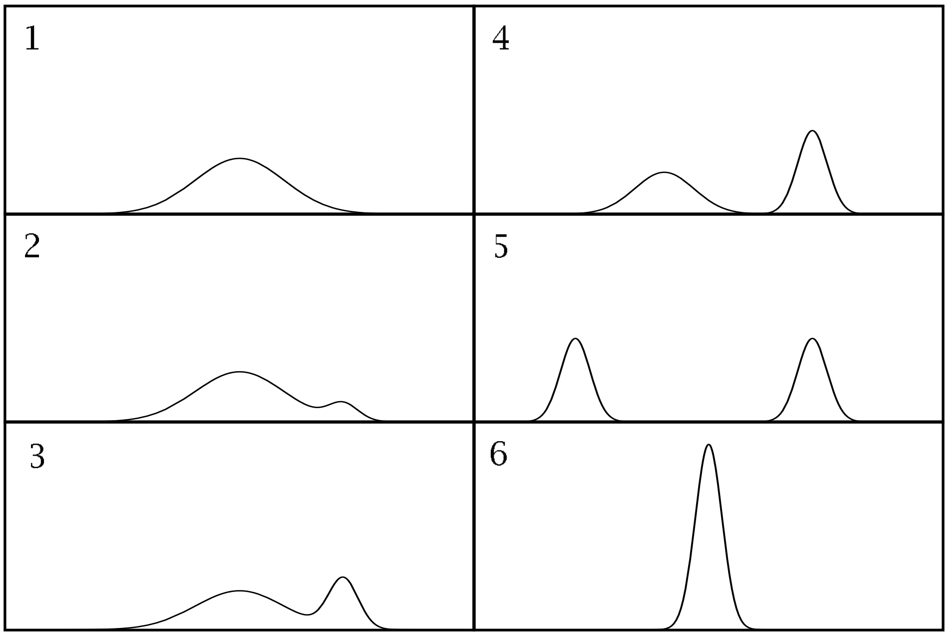


Figure 2. *Increasingly Sharp Continuous Distributions*

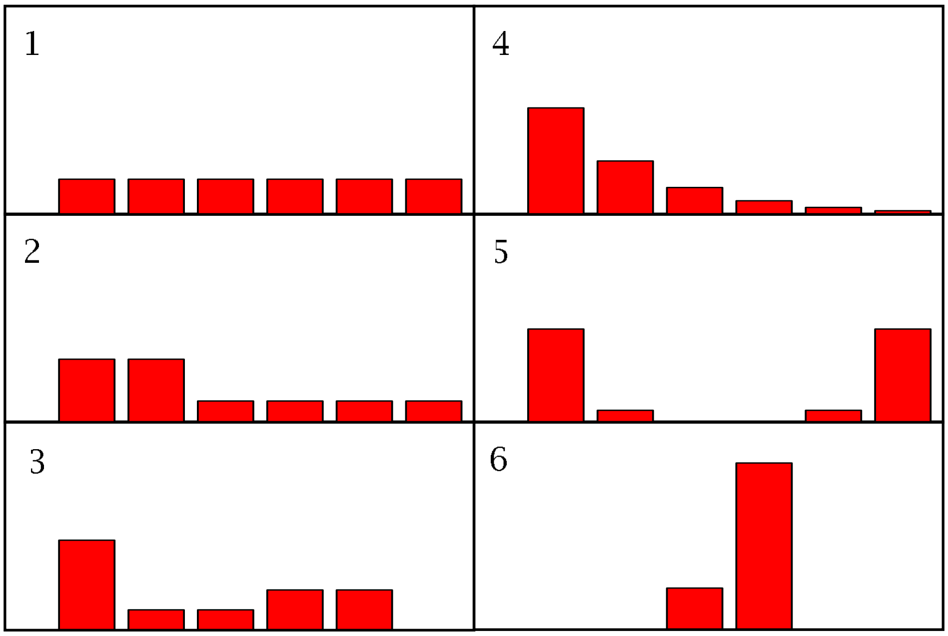


Figure 3. *Increasingly Sharp Discrete Distributions*

The appropriate relative scale resembles the P-value scale in classical statistics. We are after the probability of seeing conditioning information at least as improbable as $\mathbf{x}$, so we sum the probabilities of all the points that are less probable. We also need to account for the fact that as the amount of data increases, we expect to see more and more improbable events. The formula is,

$$\text{Reliability} \quad \equiv \quad N \int_S \Pr(\mathbf{x}' \mid MI)\, d\mathbf{x}' \qquad S = \{\mathbf{x}' \mid \Pr(\mathbf{x}' \mid MI) \leq \Pr(\mathbf{x} \mid MI)\}$$

This function varies on a scale between 0 and N. It has a natural interpretation as the expected number of points in the database with conditioning information less probable than $\mathbf{x}$. A reliability of 3, for instance, indicates that one should expect to find 3 rows in the database rarer than $\mathbf{x}$. The models in this dissertation explicitly account for noise to ensure that sharp predictions do not arise from unreliable conditioning information. There is no quantitative law, but as reliability decreases, sharpness also decreases to generate more conservative predictions.

V. Conditional Estimation

Density estimation is difficult because it considers both sharpness and reliability. Often, practical considerations make it possible to dispense with reliability and model $\Pr(\mathbf{y} \mid \mathbf{x}MI)$ directly. Chapter 6 justifies this approach and points out its limitations. Where justified, it saves tremendous time and trouble by ignoring all the complex structure that may appear in $\Pr(\mathbf{x} \mid MI)$.

In the student data, when trying to predict years to graduate we can factor the joint density,

$$\begin{aligned} &\Pr(\text{recommendations}, \text{GRE}, \text{GPA}, \text{years}, \text{school} \mid I) = \\ &\qquad\qquad \Pr(\text{years} \mid \text{recommendations}, \text{GRE}, \text{GPA}, \text{school}\, I) \times \\ &\qquad\qquad \Pr(\text{recommendations}, \text{GRE}, \text{GPA}, \text{school} \mid I) \end{aligned}$$

The application information may be useless for making predictions, but a density model must painstakingly describe all the correlations among a student's recommendations, GRE scores, GPA, and undergraduate school. All the elaborate structure it finds will simply disappear when making predictions.

The parameterized form of the conditional distribution $\Pr(\mathbf{y} \mid \mathbf{x} M_\phi I)$ is almost identical to the joint distribution in Equation 2. The kernel that gives the distribution its shape is the same, but the partition function now depends on the conditioning information $\mathbf{x}$:

$$\begin{aligned}\Pr(\mathbf{y} \mid \mathbf{x} M_\phi I) &= \frac{1}{Z(\mathbf{x},\phi)} k(\mathbf{y} \mid \mathbf{x}\phi) \\ Z(\mathbf{x},\phi) &= \int k(\mathbf{y} \mid \mathbf{x}\phi)\, d\mathbf{y}\end{aligned} \tag{7}$$

The function $k(\mathbf{y} \mid \mathbf{x}\phi)$ is identical to $k(\mathbf{z} \mid \phi)$. We write it differently, with $\mathbf{z}$ separated into $\mathbf{x}$ and $\mathbf{y}$, simply as a notational convenience to indicate that some of the variables are known. Equation 7 is more general than Equation 2, since it reduces to it when we let $\mathbf{y}$ contain all the features and $\mathbf{x}$ none of them. Equation 7 expresses how to seamlessly shift observations from one side of the conditioning bar to the other.

In the admissions data, the sum needed to compute $Z(\mathbf{x},\phi)$ is taken over the different lengths of time a student might take to graduate, conditioned on the known data $\mathbf{x}$ in his application and the model parameters ϕ. In the previous case where we modeled each row using point mass distributions for the discrete components and a multivariate normal for the continuous ones, all the discrete components vanish, since they are independent of years to graduate. The conditional probability of years to graduate given all the admissions information is,

$$\begin{aligned}&\Pr(\text{years} \mid \text{recommendations}, \text{GRE}, \text{GPA}, \text{school}, \text{papers}\, M_\phi I) = \\ &\qquad\qquad N(\text{years} \mid \text{GRE}, \text{GPA}, \boldsymbol{\mu}, \Sigma)\end{aligned}$$

The kernel of the multivariate normal is computed using the same mean and covariance matrix as before, but now it is treated as a function of only one variable, years, rather than of the three variables years, GRE, and GPA. The normalizing factor changes to reflect the GRE and GPA scores. The model used in this example is obviously too weak for serious data analysis, since it fails to account for correlations between the discrete observations and time to graduate, but it forms the building block for the mixture and product models that arise in models with hidden features.

The conditional likelihood of a database table is,

$$\Pr(D \mid M_\phi I) = \prod_{i=1\ldots N} \frac{1}{Z(\mathbf{x}_i, \phi)} k(\mathbf{y}_i \mid \mathbf{x}_i \phi) \tag{8}$$

Comparing to Equation 4, which gave the analogous formula for the joint likelihood, in this case the partition function $Z(\mathbf{x}, \phi)$ does not factor out of the product, because it is different for each row of the database. Nevertheless, Equation 8 may be much easier to evaluate than Equation 4, since $Z(\mathbf{x}, \phi)$ integrates over fewer dimensions than $Z(\phi)$. In the most common case where $\mathbf{y}$ is only a single feature, $Z(\mathbf{x}, \phi)$ is a one dimensional sum or integral, which in the worst case we can numerically evaluate in time proportional to that needed for $k(\mathbf{y} \mid \mathbf{x}\phi)$. This speedup over density estimation, where the time complexity of $Z(\phi)$ can grow exponentially in the number of features, is one of the primary lures of conditional estimation.

As a caveat, the exposition has been a little too slick in moving from the joint likelihood function to the conditional one. As Jaynes points out, conditioning on continuous quantities like GRE and GPA is really a limit operation, and in general the kernel function may change depending on how the limit is approached [28]. However, the models in this dissertation are all members of the well behaved exponential family, and the limiting processes are the natural ones that lead to Equation 7. Interested readers are encouraged to read Jaynes' entertaining discussion on paradoxes in probability theory.

4
Mixture Models

I. AutoClass

The goal of this dissertation is to improve AutoClass, a Bayesian database analysis system developed over the last six years at NASA's Ames Research Center [9, 10, 19]. AutoClass traces its ancestry to the long history of finite mixture distributions in statistics [53] and to an earlier program, SNOB, developed by Wallace in the early 1970's [3]. It is unsupervised, working without any prediction or validation tasks in mind. Its goal is to find a good model of all the effects in a database, thereby providing insight into large, poorly understood domains. It has been successfully applied to many problems, including the IRAS astronomical data [9], protein amino acid sequences [23], and raw English text[51].

AutoClass operates within the density estimation framework of Chapter 3, treating each row in the database as an independent sample from a fixed distribution. Its goal is to reconstruct that distribution. Conceptually, AutoClass partitions the rows into K disjoint classes, models each class independently of the others, and then puts the classes together in a mixture model that combines them all.

The iris database from Chapter 1 provides a good illustration. The model that AutoClass finds splits the data of Figure 2-1 into two classes like this:

Sepal Length	Petal Length	Cluster
5.1	1.4	Bottom
5.4	1.7	Bottom
7.7	6.9	Top
6.7	4.7	Top
4.8	1.6	Bottom
...	...	...

Each class is described using a bivariate normal distribution over petal length and sepal length. Figure 1 plots the final model using ellipses to indicate each class. Each ellipse is two standard deviations in radius and centered at the mean of the class distribution. The numbers indicate a class's weight in the mixture. The numerical parameters are,

	Classes #1	*Classes* #2
	0.33	0.67
sepal length	5.01 ± 0.36	6.26 ± 0.66
petal length	1.46 ± 0.17	4.91 ± 0.82
ρ	0.28	0.83

Conceptually, AutoClass finds this model by doing continuous function optimization over parameterized mixtures of two bivariate normals:

$$\Pr(\mathbf{z} \mid M_{\phi} I) = w_1 N(\mathbf{z} \mid \boldsymbol{\mu}_1 \Sigma_1) + w_2 N(\mathbf{z} \mid \boldsymbol{\mu}_2 \Sigma_2)$$

AutoClass uses the EM optimization algorithm [11, 45], but any good optimization algorithm will work, and it is easy to visualize the dynamics of a general gradient search. Changing $\boldsymbol{\mu}$ causes an ellipse to move while keeping its shape and orientation; changing Σ causes it to

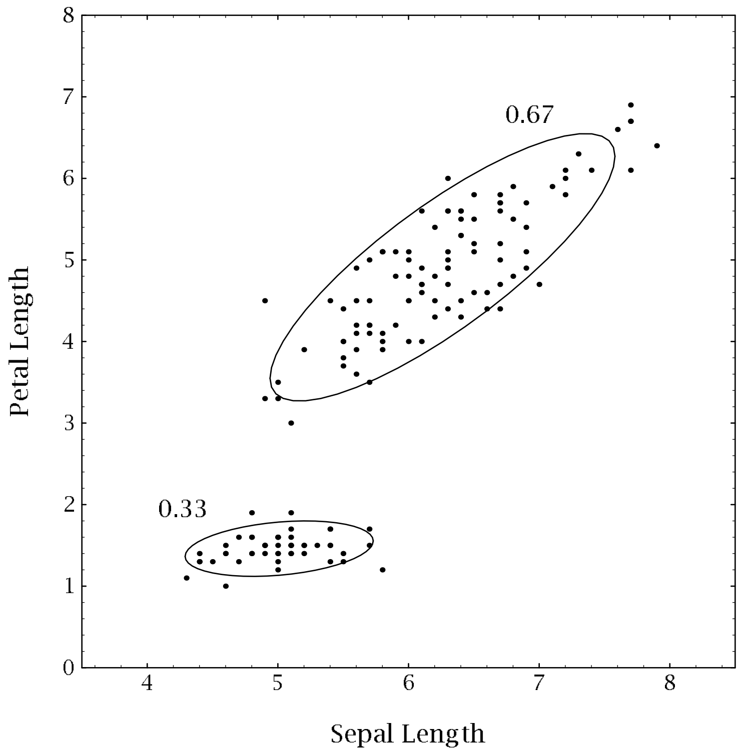


Figure 1. *Model of the Iris Data*

stretch or shrink or rotate; changing **w** gives one class more weight, and the other less. Finding the optimal model amounts to adjusting the ellipses until they fit the data well. The final distribution is not quite the maximum likelihood solution, since AutoClass incorporates priors that influence the values slightly.

AutoClass works much the same way in the day camp database, where it classifies the days according to weather:

Swimming	Whiffle	Coloring	Singing	Painting	Weather
×	×	–	–	–	sunshine
–	–	×	–	×	rain
×	–	×	×	–	sunshine
×	×	–	–	–	sunshine
–	–	–	×	×	rain
–	–	×	×	×	rain
...	...	...	...	...	...

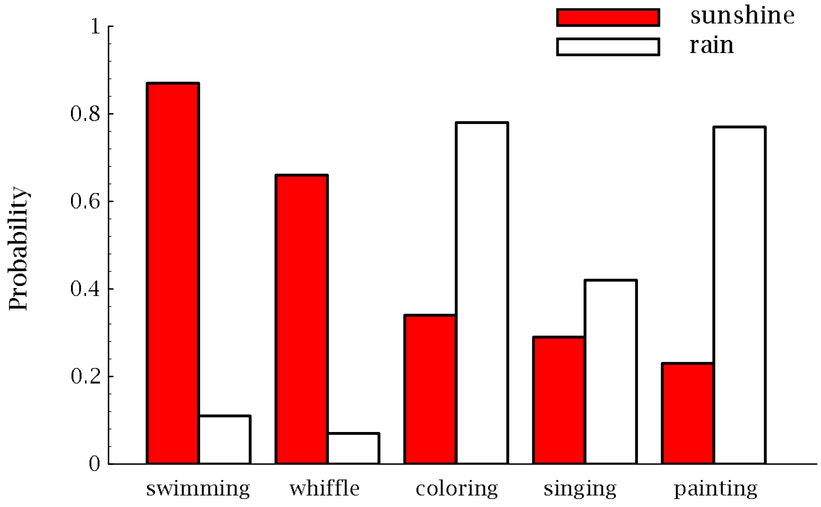


Figure 2. *Model of the Day Camp Data*

Its model uses a Bernoulli distribution to describe the probability of each activity given rain or sun. Figure 2 plots the final model with a bar graph showing the probability of each activity in the two kinds of weather. The probability values are,

	Classes	
	sun	*rain*
	0.62	0.38
swimming	0.87	0.11
whiffle ball	0.66	0.07
coloring	0.34	0.78
singing	0.29	0.42
painting	0.23	0.77

Notice that under the hypothesis of this model, once the weather is known the probability of any activity happening is independent of whether or not any other occurs. A search for the optimal model proceeds much as it did for the iris data. In this case, the search algorithm adjusts the heights of the bars in Figure 2 and the weight of each class until it gets a good fit.

The parameterized function it is trying to optimize is,

$$\Pr(\mathbf{z} \mid M_{\phi} I) = w_1 f(\mathbf{z} \mid \mathbf{p}_1) + w_2 f(\mathbf{z} \mid \mathbf{p}_2)$$

$$f(\mathbf{z} \mid \mathbf{p}) = \prod_i \nu_i \qquad \nu_i = \begin{cases} p_i & \text{if } z_i = \times \\ 1 - p_i & \text{if } z_i = - \end{cases}$$

II. General Mixture Models

In the general case, AutoClass follows the MAP algorithm detailed in Chapter 1, searching the space of mixture models to find the optimal number of classes, the optimal description of each class, and the optimal frequency with which each class appears. The general likelihood function for its mixture models is,

$$\Pr(\mathbf{z} \mid \mathbf{w}\theta K I) = \sum_{i=1 \ldots K} w_i f(\mathbf{z} \mid \theta_i) \tag{1}$$

where the parameters satisfy the constraints,

$$\sum w_i = 1$$
$$\int f(\mathbf{z} \mid \theta_i)\, d\mathbf{z} = 1$$

The function f, called the class description function, controls what each class can look like. The next section goes through the range of possibilities that AutoClass allows. Each term in Equation 1 corresponds to the probability of a class, and the conditional probability of observing different values of $\mathbf{z}$ given that class:

$$\Pr(\text{class} = i \mid \mathbf{w}\theta K I) = w_i$$

$$\Pr(\mathbf{z} \mid \text{class} = i, \mathbf{w}\theta K I) = f(\mathbf{z} \mid \theta_i)$$

Equation 1 is just the marginal probability of $\mathbf{z}$ summed over all the classes.

An important point about the way AutoClass works is that it never categorically assigns objects to classes; Equation 1 computes probabilities taking all the classes into account. The first day of the day camp data, for example, is clearly sunny, whereas the third day might be either sunny or rainy. Using the final model and Bayes' theorem we can compute exact numbers:

$$\begin{aligned}
\Pr(3^{\text{rd}}\text{ day} \mid \text{sun}\, MI) &= (0.87)(0.66)(1-0.34)(1-0.29)(1-0.23) = 0.0225 \\
\Pr(3^{\text{rd}}\text{ day} \mid \text{rain}\, MI) &= (0.11)(0.07)(1-0.78)(1-0.42)(1-0.77) = 0.0077 \\
\text{Odds(sun)} &= \frac{0.62 \times 0.0225}{0.38 \times 0.0077} = 4.77 \\
\Pr(\text{sun} \mid 3^{\text{rd}}\text{ day}\ MI) &= \frac{4.77}{1+4.77} = 0.83
\end{aligned}$$

We can draw the augmented day camp table to explicitly show the probability of each kind of weather:

Swimming	Whiffle	Coloring	Singing	Painting	Pr(sun)	Pr(rain)
×	×	–	–	–	1.00	0.00
–	–	×	–	×	0.01	0.99
×	–	×	×	–	0.83	0.17
×	×	–	–	–	1.00	0.00
–	–	–	×	×	0.02	0.98
–	–	×	×	×	0.01	0.99
...	...	...	...	...	...	...

The extension to non-binary hidden features is obvious. The examples in this dissertation all show only the most probable class assignments, rather than the individual class probabilities. For problems like the iris data where the classes are well separated, the two displays are roughly equivalent, but writing out the class probabilities always exposes the entries for which no single class is clearly indicated.

The K-means clustering algorithm [12, 53] is closely related to AutoClass. It differs in that it uses hard classifications where every object is assigned to its most probable class. This approach works fine for problems like the iris data, but leads to known biases when the classes are not well separated. The trouble is that K-means optimizes the wrong quantity. It finds a peak in the joint likelihood of the model parameters and hidden class feature, instead of the desired peak in the posterior distribution of the model parameters alone.

III. Class Description Functions

AutoClass lets the user choose many different possible class description functions, and the selection has grown with each new version of the program. AutoClass II and III use the simplest possible function, one that treats each observation independently from the others. Continuous, real valued features are modeled using a normal distribution; discrete and categorical features are modeled using a point mass distribution. Within a class, every feature is independent of all the others.

Letting θ denote all the parameters in the class description function, for the iris data AutoClass II uses,

$$\begin{aligned} f(\text{petal length}, \text{sepal length} \mid \theta) = & \\ & N(\text{petal length} \mid \mu_1 \sigma_1) \times \\ & N(\text{sepal length} \mid \mu_2 \sigma_2) \end{aligned}$$

This function differs from the one in Figure 1, since it does not allow a correlation between petal length and sepal length within a class. Figure 3 shows the optimal mixture model using this f. The means and variances of the two classes are identical to those in Figure 1, but the correlation within each class is zero. The contrast between Figures 1 and 3 emphasizes the importance of choosing a good f. With equal prior probabilities, the model in Figure 1 is 10^{26} times more probable than the model in Figure 3. The odds ratio more than doubles with every new data point.

A more complete example including categorical features is the United Nations data from Chapter 1:

$$\begin{aligned} f(\text{height}, \text{weight}, \text{eyes}, \text{hair}, \text{skin} \mid \theta) = & \\ & N(\text{height} \mid \mu_1, \sigma_1) \times \\ & N(\text{weight} \mid \mu_2, \sigma_2) \times \\ & M(\text{eyes} \mid \mathbf{p}_1) \times \\ & M(\text{hair} \mid \mathbf{p}_2) \times \\ & M(\text{skin} \mid \mathbf{p}_3) \end{aligned}$$

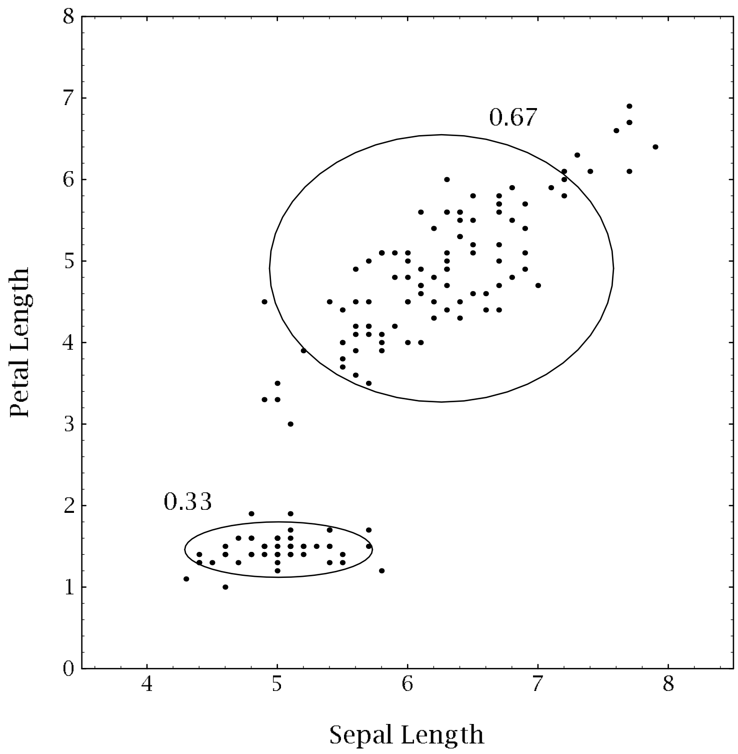


Figure 3. *Model of the Iris Data with no Correlations*

Although features are all independent within a single class, they are likely to be correlated in the overall mixture. In the model of Figure 3, if we know that a flower belongs to the bottom class, then learning its petal length tells us nothing about its sepal length. But if we have no idea which class a flower is in, then any estimate of its sepal length depends very strongly on its petal length.

AutoClass IV extends the model space of AutoClass II and III to a multivariate normal distribution over continuous components. We have already seen this model for the iris data. Chapter 2 presented another example with the student data:

$$
\begin{aligned}
f(\text{recommendations}, \text{GRE}, \text{GPA}, \text{years}, \text{school} \mid \theta I) = \\
& M(\text{recommendations} \mid \mathbf{p}_1) \times \\
& M(\text{school} \mid \mathbf{p}_2) \times \\
& N(\text{GRE}, \text{GPA}, \text{years} \mid \boldsymbol{\mu}, \Sigma)
\end{aligned}
$$

The user has the option of breaking the continuous features into disjoint sets, creating a block diagonal covariance structure in which the different sets are independent. In the limiting case where all the continuous features are treated separately, this model space reduces to the one in AutoClass II and III. AutoClass IV incorporates a number of additional possibilities for the class description function. For instance, it provides a way to specify bound constraints on continuous variables by modeling them with a truncated normal or an exponential distribution, and it lets the user use a Poisson distribution to treat discrete data distinctly from categorical.

All the versions of AutoClass require the user to choose a class description function by hand, but in principle the system could easily conduct an automated search across all the possibilities.

The results of this dissertation apply using any f that is simple to compute and normalize, like those presented here. Functions in the exponential family are particularly advantageous, since the presence of sufficient statistics lets one use the fast EM optimization algorithm employed by AutoClass. Where there is significant domain expertise, one should choose an f that meshes with the domain knowledge. Figure 4 shows a hypothetical database listing the recovery locations of meteorite fragments. For this database, one would choose a ring shaped class description function:

$$f(x, y \mid x_0, y_0, \mu, \sigma) \propto N(\sqrt{(x - x_0)^2 + (y - y_0)^2} \mid \mu, \sigma)$$

For poorly understood data, one should use a general f like those in AutoClass. MultiClass, which implements the ideas in this dissertation, currently uses point mass, normal, and multivariate normal distributions. Its design lets one easily extend the model space by adding new ones.

IV. Belief Nets

Bayesian belief nets are a good language for understanding AutoClass, and for describing hidden feature models in general. A complete account of belief nets can be found in many places,

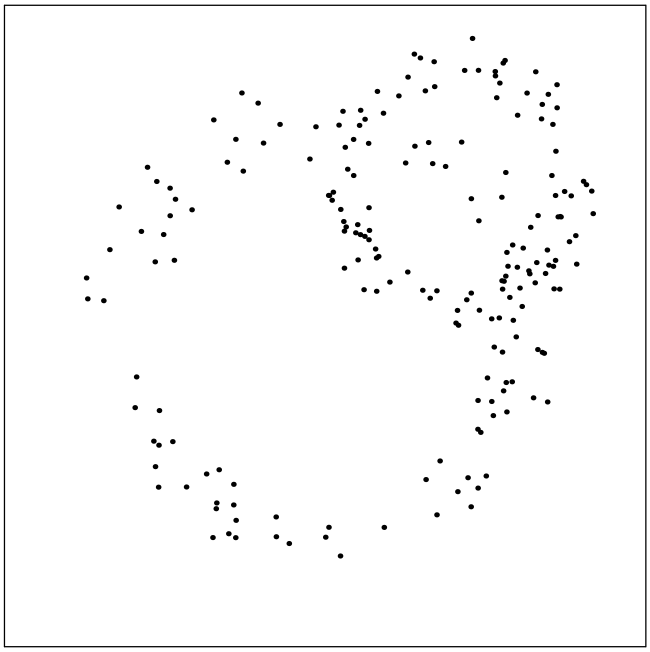
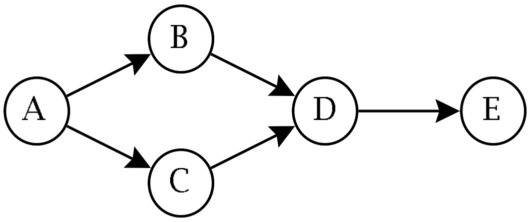


Figure 4. *The Meteorite Database*

most notably Pearl's book [47]. For the reader that needs a refresher, here is a simple directed belief net over the propositional observations A–E:



The graph structure encodes independence relations that let one express a joint probability distribution over the five propositions compactly:

$$\Pr(ABCDE) = \Pr(A)\Pr(B \mid A)\Pr(C \mid A)\Pr(D \mid BC)\Pr(E \mid D)$$

This factorization requires only 22 parameters to characterize the joint probability, rather than the $2^5 = 32$ that would be needed without any independence assumptions. The savings grow

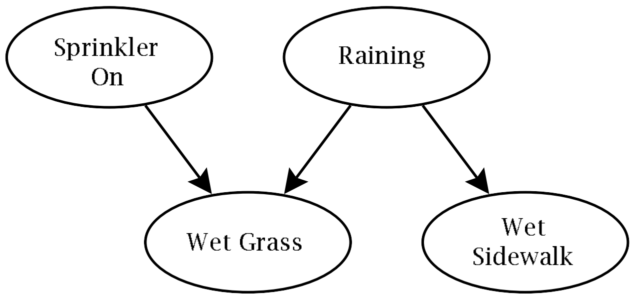


Figure 5. *Wet Grass*

exponentially as the number of variables increases.

A great attraction of belief nets is that the arcs in the graph often have a natural interpretation as causal influences. This property makes them easy for people to write down and understand. Figure 5 shows a popular example involving wet grass and lawn sprinklers, while Figure 6 illustrates a small fragment of a real belief net used to diagnose gas turbine trouble [20]. Although the engine domain contains an impressive number of variables, the independence relationships let one express the joint probability of all of them using relatively few numbers.

AutoClass implements the simple belief net in the left half of Figure 7. The single conditional arc corresponds to the joint probability factorization,

$$\Pr(\mathbf{z}\,\text{class} \mid I) = \Pr(\mathbf{z} \mid \text{class}\,I)\,\Pr(\text{class} \mid I)$$

Summing over the unknown class variable generates the mixture models used in AutoClass:

$$\Pr(\mathbf{z} \mid I) = \sum \Pr(\mathbf{z} \mid \text{class} = i, I)\,\Pr(\text{class} = i \mid I)$$

This belief net can be further refined if the class description function includes additional independence assumptions. In AutoClass, independence within each class results in what Heckerman calls an “idiot Bayes” model, which is shown in the right half of Figure 7 for the United Nations Data.

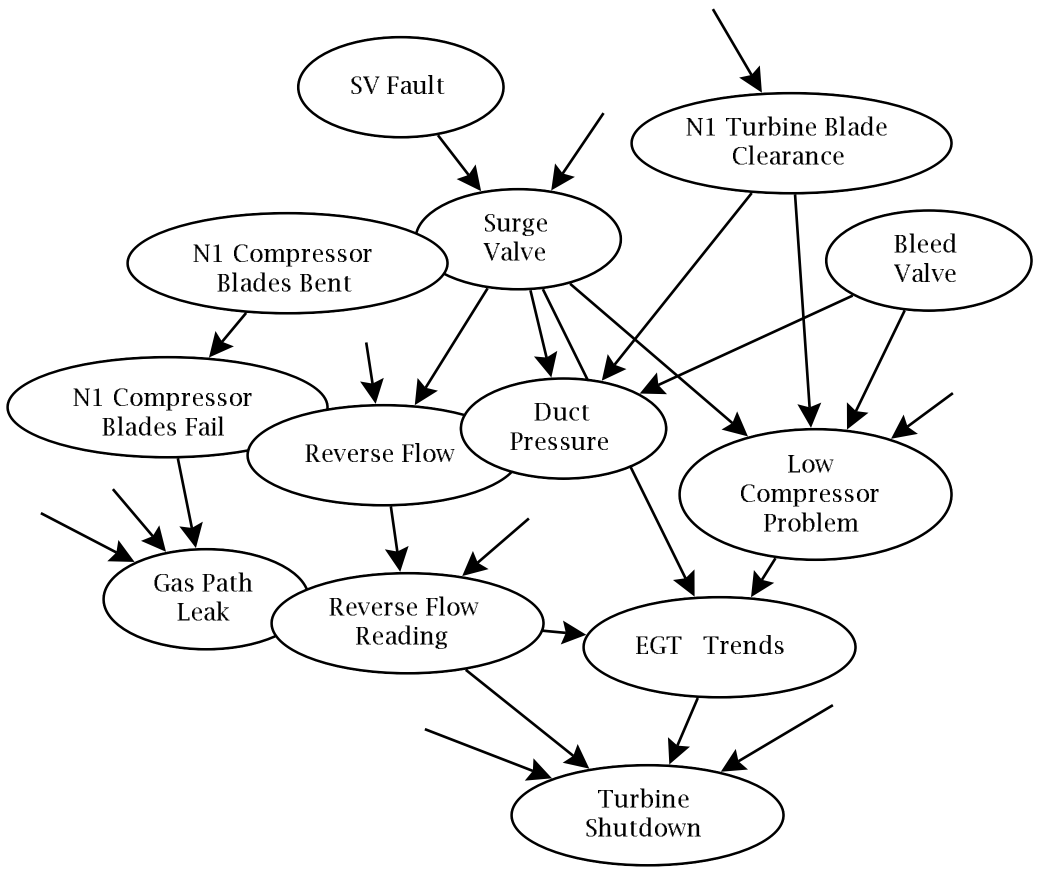


Figure 6. *Gas Turbine Diagnosis*

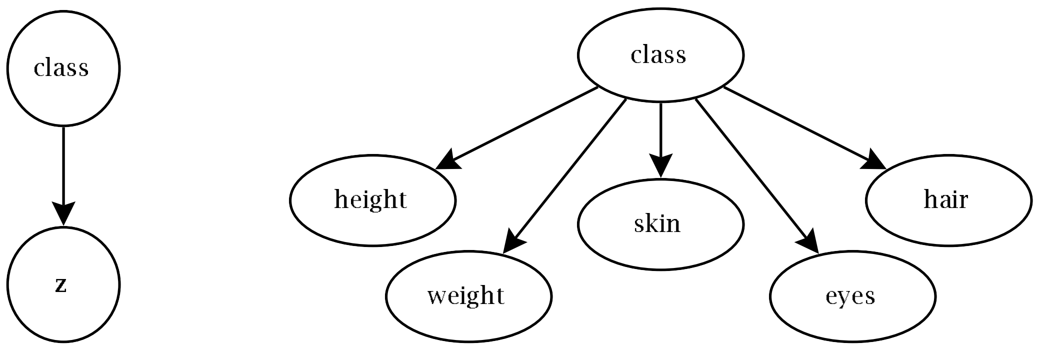


Figure 7. *The AutoClass Belief Nets*

The obvious difference in complexity between the expert system in Figure 6 and the belief nets in Figure 7 lets one appreciate why AutoClass might not capture all the systematic effects in a database. Nevertheless, fielded expert systems, like the medical diagnosis system PathFinder [20], have demonstrated success using models close to the idiot Bayes one. The complexity of Figure 6 should leave one skeptical that AutoClass will find all the effects in a database, but there is no doubt that whatever structure it does find is very real.

Belief nets like those in Figure 7 have a much wider scope than the way that AutoClass uses them. A naive Bayesian classifier, for example, is nothing more than an idiot Bayes model applied to data where the class variable is known rather than hidden [36, 13].

V. Prior Probabilities

Priors are important to avoid overfitting and ensure that one arrives at sensible, conservative answers when data is scarce. Bayesian philosophy emphasizes that no prior is universally correct, but the mixture prior developed here benefits from a sound rationale and excellent empirical results. It breaks up the complete prior for a mixture model, $\Pr(\phi \mid I)$, into independent parts, one for each set of class parameters θ_i, and one for the class weights w:

$$\Pr(\phi \mid I) = \Pr(\mathbf{w} \mid I) \prod_{i=1\ldots K} \Pr(\theta_i \mid I)$$

Since the order of the classes is interchangeable, all the $\Pr(\theta_i \mid I)$ are given by a single function $\Pr(\theta \mid I)$. This function quantifies what one expects a class to look like before seeing any observations that fall into it.

The insight behind $\Pr(\theta \mid I)$ is that when a class contains little data, it most plausibly resembles the database as a whole. Figure 8 illustrates this effect using the GRE scores from the student database. The observed mean and standard deviation for all students is 680 ± 50, and these values act as a magnet tugging on the estimated distribution for any subset, like those attending a particular school. The top panel shows the Harvard students, who have an empirical mean GRE of 748; the middle panel shows Berkeley, at 609; the bottom panel shows the lone student from Swarthmore, at 800. In all three cases, the estimated distribution of GRE

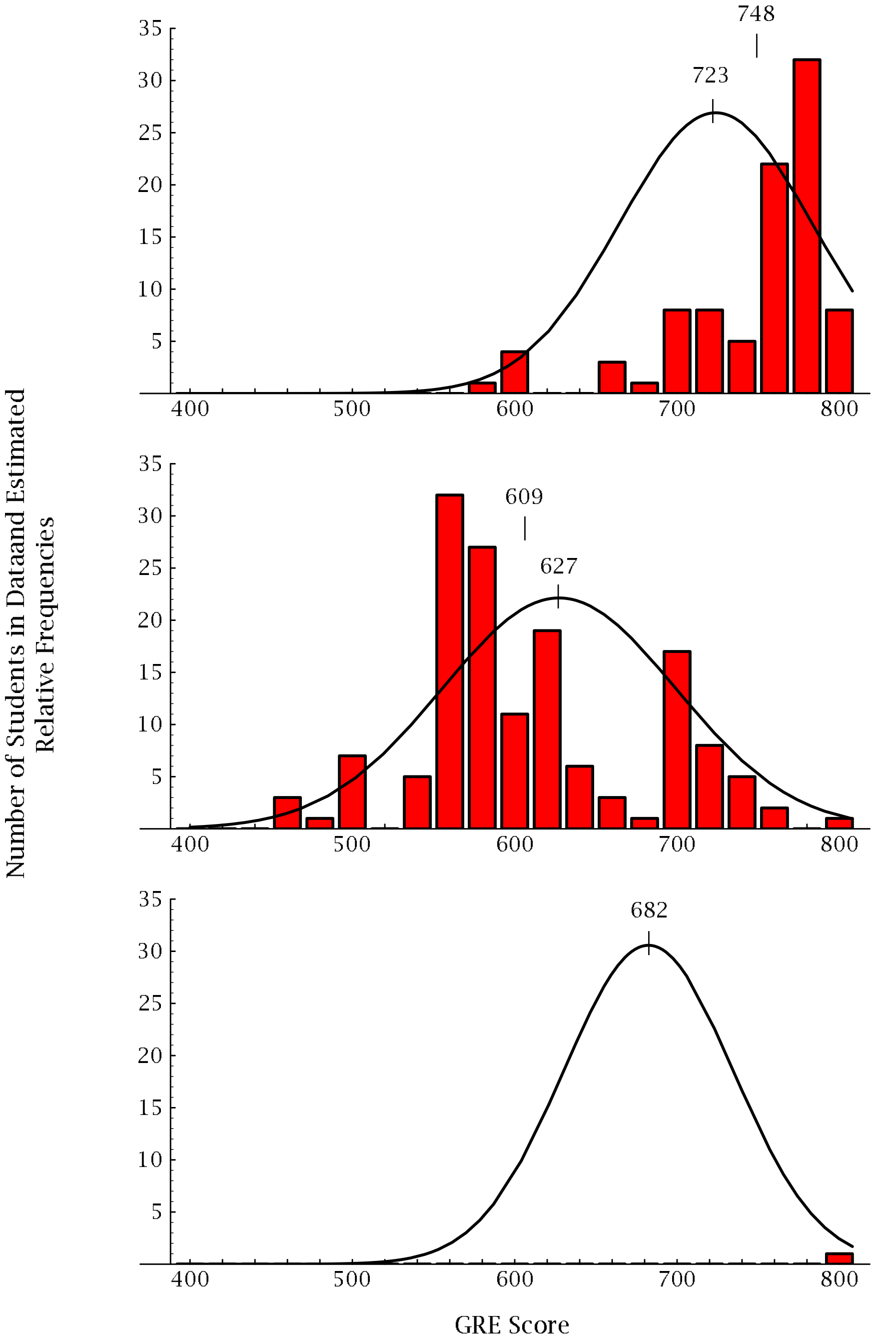


Figure 8. *Prior Influence on Estimated GRE Scores*

scores is shifted towards 680 ± 50, the distribution one expects before seeing any students from a school. The estimate for Swarthmore has hardly budged despite the magnificent score of 800, because a single student is far too few to have a serious impact on prior expectations. The net effect of this prior is to avoid sharp spikes in the final joint density that are supported by only one or two observations.

The categorical case is no different. If we know that children go swimming on 80% of the days at the day camp, then our prior should pull classes towards that value until there is enough data to justify a different conclusion. Quinlan has noted the importance of a prior like this for decision trees, and he suggests that the leaf predictor nodes should relax towards the base prediction rate in the data, rather than the 50-50 guess that is common.

One achieves the effect of these examples by using an informative, data driven prior that takes the original database, reduces it to an effective size of P entries, and feeds it as an additional input to each class estimate:

$$\begin{aligned}\Pr(\theta \mid PI) &\propto \Pr(D \mid \theta I)^{\frac{P}{N}} \\ &= \left(\prod_{i=1\ldots N} f(\mathbf{z}_i \mid \theta)\right)^{\frac{P}{N}}\end{aligned} \tag{2}$$

P is a hyperparameter that controls the weight of the prior. The normalizing constant is of no importance in MAP induction, since it is the same for all θ and cancels out in the search for the optimal model. When the amount of data in a class is small, so that the prior dominates the likelihood, Equation 2 forces θ towards noncommittal values that fit the database as a whole.

In cases where the class description function is in the exponential family, the presence of sufficient statistics lets one eliminate the product and obtain a simple, conjugate prior. With the student GRE scores, one ends up with the following prior over μ and σ:

$$\Pr(\mu, \sigma \mid PI) \propto \left(\frac{1}{\sigma} e^{-\frac{1}{2\sigma^2}\left((\mu-680)^2+50^2\right)}\right)^P$$

This prior corresponds to the probability of observing P additional GRE scores with the same mean and variance, 680 ± 50, as the original N. In practice, one would normalize the GRE scores to mean zero and variance one before doing any analysis leaving an even simpler equation:

$$\Pr(\mu, \sigma \mid PI) \propto \left(\frac{1}{\sigma} e^{-\frac{1}{2\sigma^2}(\mu^2+1)}\right)^P$$

In the discrete day camp data where children go swimming on 80% of the days, the parameter q that gives the probability of going swimming in a subclass of days has the prior,

$$\Pr(q \mid PI) \propto \left(q^{0.8}(1-q)^{0.2}\right)^P$$

which corresponds to the probability of observing P additional days in which the children go swimming 80% of the time.

It is straightforward to derive a conjugate prior for any probability distribution in the exponential family. The formulas for normal, multivariate normal, and point mass distributions are given here for completeness. For a normal distribution with an empirical mean $\overline{x}$ and variance s^2:

$$\begin{aligned} \Pr(\mu\sigma \mid \overline{x}s^2\, PI) &\propto \left(\frac{1}{\sigma} e^{-\frac{1}{2\sigma^2}((\mu-\overline{x})^2+s^2)}\right)^P \\ \overline{x} &= \frac{1}{N}\sum x_i \\ s^2 &= \frac{1}{N}\sum\left(x_i - \overline{x}^2\right) \end{aligned} \tag{3}$$

For a multivariate normal distribution with an empirical mean $\overline{\mathbf{x}}$ and covariance matrix S^2:

$$\begin{aligned} \Pr(\boldsymbol{\mu}\Sigma \mid \overline{\mathbf{x}}S^2\, PI) &\propto \left(\frac{1}{|\Sigma|^{1/2}} e^{-\sum \frac{1}{2\lambda_{ij}^{-1}}\left((\mu_i-\overline{x}_i)(\mu_j-\overline{x}_j)+s_{ij}^2\right)}\right)^P \\ \overline{x}_i &= \frac{1}{N}\sum x_i \\ s_{ij}^2 &= \frac{1}{N}\sum (x_i - \overline{x}_i)(x_j - \overline{x}_j) \end{aligned} \tag{4}$$

For a point mass distribution with an empirical frequency vector $\mathbf{f}$:

$$\begin{aligned} \Pr(\mathbf{p} \mid \mathbf{f}\, PI) &\propto \left(\prod p_i^{f_i}\right)^P \\ f_i &= \frac{1}{N}\sum_j \delta(x_j = i) \end{aligned} \tag{5}$$

In all these priors, one must either give P a definite numerical value, or else step back and introduce a new prior, $\Pr(P \mid I)$, to reflect a range of possibilities. Setting P to a small value, around 3 or 4, is reasonable and gives a broad, weakly informative prior. AutoClass uses the conjugate priors shown here, although the AutoClass papers do not tie the parameters to the data as in Equation 2 and give no rationale for how to set them. The complete prior $\Pr(\theta \mid PI)$ is the product of one instance of Equation 3, 4, or 5 for each of the independent components in the class description function.

Turning to the class weights, it is natural to treat the hidden class feature just like any other categorical feature and use the Dirichlet prior of Equation 5 for the prior $\Pr(\mathbf{w} \mid I)$. In this case we are ignorant of how often each class might appear, so we use the symmetric prior,

$$\Pr(\mathbf{w} \mid kI) \propto \prod_{i=1\ldots n} w_i^{k-1}$$

The constant k is another hyperparameter and can take on any positive value. Figure 9 plots the symmetric Dirichlet distribution for three different values of k when $n = 2$. The plot shows there are three distinct shapes: when $k < 1$ the distribution is a horseshoe favoring the extremes; when $k = 1$ the distribution is uniform; when $k > 1$ the distribution is a bell favoring the middle. The same pattern holds in higher dimensions:

$$\text{Dirichlet shape} = \begin{cases} \text{horseshoe}, & k < 1 \\ \text{uniform}, & k = 1 \\ \text{bell}, & k > 1 \end{cases}$$

Uninformative priors can be logically deduced in many ways, including group invariance [25], marginalization [26], and arguments that seek to maximize the information content of the data [2]. In this instance one can use a marginalization argument, which this dissertation omits, to conclude that k should depend on a deeper hyperparameter α according to the relation $nk = \alpha$. The overall prior is therefore,

$$\Pr(\mathbf{w} \mid \alpha I) = \prod_{i=1\ldots n} w_i^{(\alpha/n)-1}$$

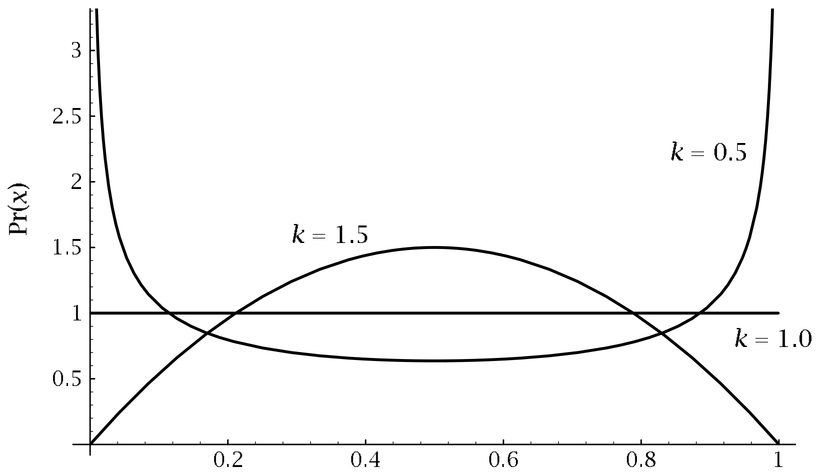


Figure 9. *Dirichlet Shapes*

The idea is for α to remain fixed as we vary the number of classes, so that we gradually shift from a horseshoe prior through a uniform prior to a bell prior as the number of classes increases. We can naturally interpret α as the number of classes known to be in the data. If we set $\alpha = 4$, for example, and look for a two class model so that $n = 2$, then the prior has a bell shape, disallows classes with zero weight, and forces us towards models that use both classes rather that just one. If we look for a six class model the situation is reversed. Now the prior has a horseshoe shape and pushes us towards models in which some classes disappear. Only when we look for a four class model, so that $n = \alpha = 4$, are the weights free to follow the likelihood function. AutoClass makes the logical choice $\alpha = 1$, so that even two classes are not guaranteed apriori.

It may seem that a horseshoe prior would always eliminate all but one class by forcing weights to zero, but it is not so. Figure 10 shows a possible marginal posterior $\Pr(w \mid \alpha DI)$ for a two class model, where $n = 2$ and $\alpha = 1$. Although the poles have infinite density, all the volume is concentrated in the middle near $w = 0.5$. The bizarre shape of Figure 10 unfortunately portends pragmatic difficulties for gradient based optimization algorithms that look for peaks in the posterior density. Chapter 7 contains more details.

Note that we set out to find an unconditional prior $\Pr(\phi \mid I)$, and instead ended up with $\Pr(\phi \mid \alpha PI)$, which depends on two hyperparameters. At present, there does not seem to be either a good way or a compelling reason to get rid of them. The combined effect of the class

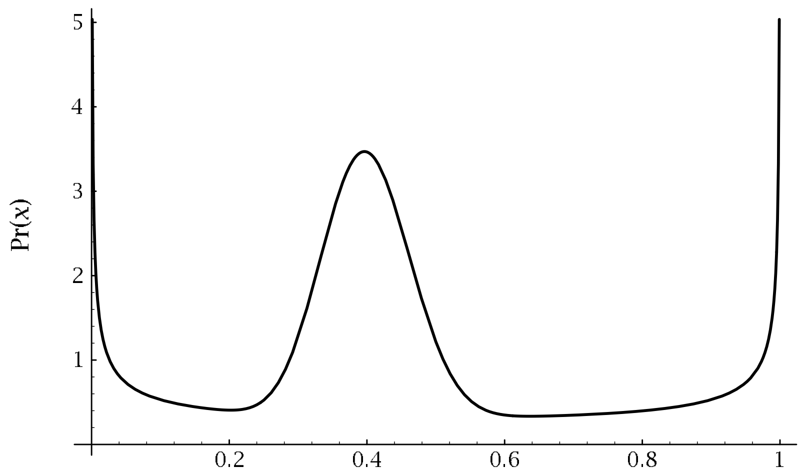


Figure 10. *Posterior Distribution for Class Weights*

prior, $\Pr(\theta \mid PI)$, and the weight prior, $\Pr(\mathbf{w} \mid \alpha I)$, is to favor models with fewer classes and to eliminate unjustified sharp spikes in the final density estimate. This bias is the Bayesian solution to the well known overfitting problem of Ockham's razor.

VI. Noise and Reliability

Using a mixture model, one can extend any joint density to account explicitly for noise and errors. If M is a density model, then a model M' allowing for noise is,

$$\Pr(\mathbf{z} \mid M'I) = (1-\epsilon)\Pr(\mathbf{z} \mid MI) + \epsilon \Pr(\mathbf{z} \mid SI)$$

S models the density of noisy observations. The constant ϵ, the noise probability, controls how often they occur. If the original joint density is itself a mixture model, as in AutoClass, then one can envision the complete model, including noise, as either a hierarchical mixture, or else as a flat mixture in which S is just one additional class.

The noise class S can potentially encode detailed knowledge about the possible sources of noise and errors. A common mistake when entering data, for example, is to exchange adjacent features, as in the prescription database from Chapter 2. M' can describe such occurrences by letting S be identical to the normal joint density model M, but with age and weight interchanged. The extension to other features is obvious, and in general S can be a mixture model that gives a case by case account of how questionable observations arise. Such a detailed noise model lets a system not only recognize discrepant entries, but also automatically diagnose and correct them.

In practice, a simple, uniform background noise model is sufficient to recognize unreliable observations and avoid overly sharp, unjustified predictions. It is easy to construct such an S. If we normalize the student data, for example, we can use,

$$\begin{aligned}\Pr(\mathbf{z} \mid SI) = \\ & M(\text{recommendations} \mid \mathbf{p}_1 = \text{uniform}) \times \\ & M(\text{school} \mid \mathbf{p}_2 = \text{uniform}) \times \\ & N(\text{GRE}, \text{GPA}, \text{years} \mid \boldsymbol{\mu} = 0, \Sigma = \delta I)\end{aligned}$$

This density function is flat and assigns no pattern to the noise. The constant δ controls how spread out the continuous observations are. The generalization to an arbitrary, normalized database is clear: we use a uniform point mass distribution for each discrete and categorical feature, and a very wide normal distribution for each continuous one.

Figure 11 shows how such a noise class claims points in the valleys of a univariate mixture of two normal densities. Points in the shaded region fall into the noise class. The odds that any point $\mathbf{z}$ will be considered noise can be computed from Bayes' theorem:

$$\frac{\Pr(S \mid \mathbf{z}I)}{\Pr(M \mid \mathbf{z}I)} = \frac{\epsilon \Pr(\mathbf{z} \mid SI)}{(1-\epsilon)\Pr(\mathbf{z} \mid MI)}$$

$\Pr(\mathbf{z} \mid SI)$ is essentially constant, so whenever $\Pr(\mathbf{z} \mid MI)$ is sufficiently small, as shown in the Figure, the noise class provides the best description and dominates predictions.

In the prescription database, the noise class explains the outlier in the lower right corner of Figure 12, with the result that the dosage prediction for this patient is a very broad "I don't know." The left half of Figure 13 plots how predicted dosage varies as a function of weight; the right half plots it as a function of age for a patient whose weight is fixed at 150 pounds. The

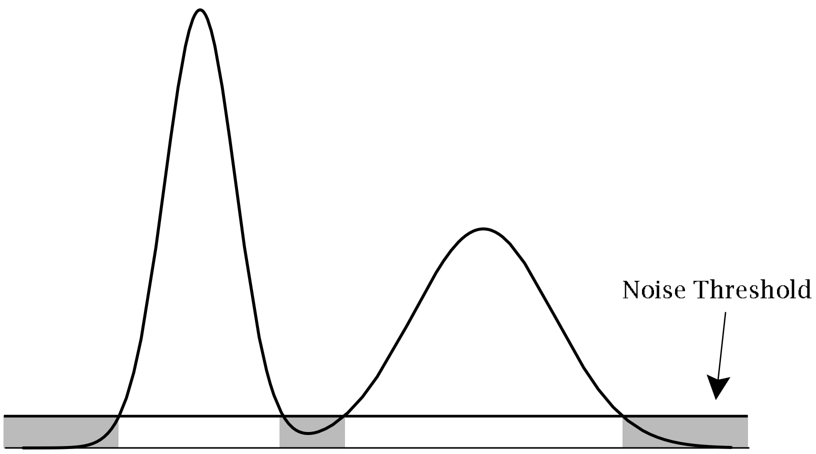


Figure 11. *Noise Claims the Valleys*

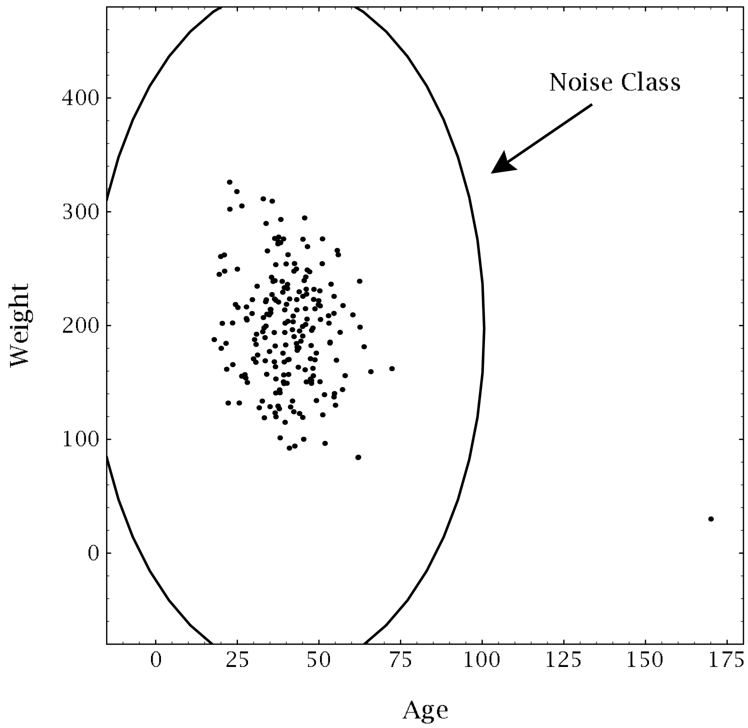


Figure 12. *Noise Class in the Prescription Database*

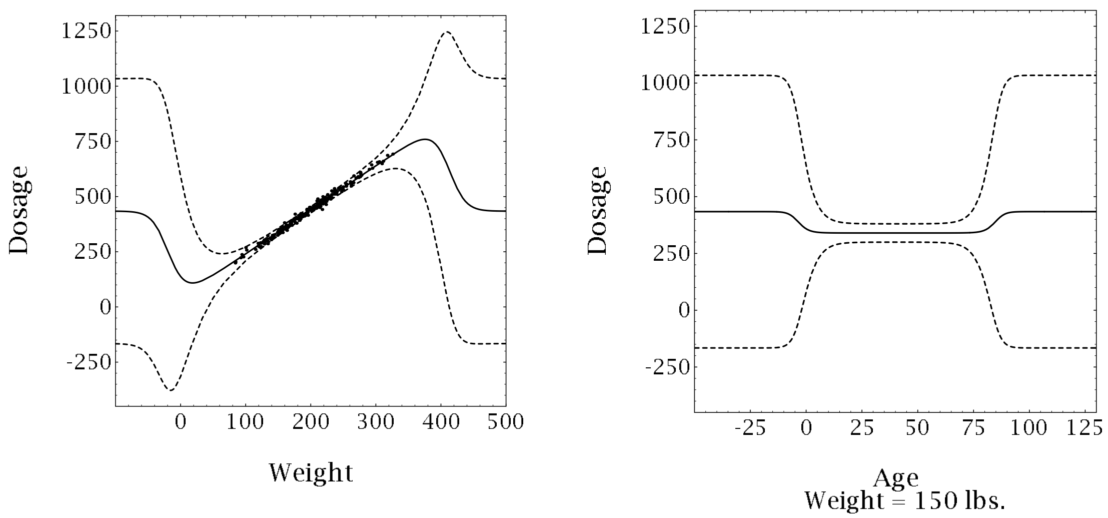


Figure 13. *Predicted Drug Dosage*

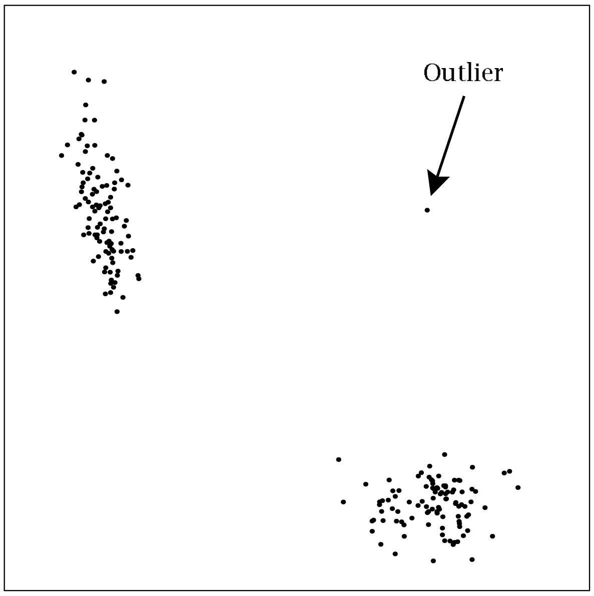


Figure 14. *A Non-Trivial Outlier*

solid line is the mean prediction, and the dashed lines give the two standard deviation span. The model makes sharp dosage predictions whenever weight and age are reasonable, but very noncommittal predictions otherwise.

The right half of Figure 13 is interesting, since it reveals how the model refuses to make sharp predictions even when it is only an irrelevant feature, in this case age, that is doubtful. One can debate whether this behavior is correct, but it errs on the side of caution and lets one spot all discrepancies in the data. The conditional models of Chapter 6 go to the other extreme: irrelevant features are always ignored, no matter how crazy their values may seem.

It is important to realize that a diffuse noise class does much more than just check each feature individually for problems. In Figure 14, for example, both the x and y coordinates of the outlier point are reasonable alone, but the combination is highly suspect. Proper reliability checking requires us to look at the complete joint density, and not just at marginal or conditional projections.

5

Multiple Hidden Features

I. Warning

This chapter develops density estimation models that use multiple hidden features, easily capable of describing all the example databases to this point. The reader should be warned, however, that they are little more than an intellectual diversion. They have not been implemented. Computational feasibility limits their utility, though there is always hope that future generations will unravel the mathematical knots that remain. The most important section is the last, which describes a strong correspondence between hidden features and logical rules, and which demonstrates that a multiple hidden feature model naturally generalizes a set of logical rules in DNF form.

II. Limitations of AutoClass

AutoClass breaks on problems like the United Nations database, where it must guess which of several interesting classifications to use. The problem is that a mixture model makes too strong an assumption about the world when it assumes that all patterns in the data can be traced to a single root cause. An unsupervised mixture model is fine for exploratory data analysis, but inappropriate when the final goal is an accurate joint density that allows targeted predictions.

With enough data, AutoClass will start combining multiple classifications into a single hidden feature that represents their crossproduct. In the United Nations data, for example, AutoClass will find classes like African men, Irish women, and so forth. Eventually, with enough data, it can include all the classifications that are useful. Combining classifications like this requires exponentially more data for each new one that is added. Here is an abstract example that strips the problem to its essentials:

x_1	y_1	x_2	y_2	$\cdots$	x_n	y_n
0	0	1	1		1	1
1	1	0	0	$\cdots$	1	1
1	1	1	1		0	0
0	0	0	0		0	0
1	1	0	0	$\cdots$	0	0
1	1	0	0		1	1
...	...	...	...		...	...

This database is constructed so that the columns x_i and y_i are identical, but all other pairs of columns are independent. One can imagine that the individual bits are generated by coin flips. The actual proportion of ones and zeros is unimportant.

With a class description function like the ones used in AutoClass, a two class mixture model can capture the constraint between one pair of columns:

x_1	y_1	Class
0	0	1
1	1	2
1	1	2
0	0	1
1	1	2
1	1	2
...	...	...

The classes are equally weighted and have the parameters,

	Classes #1	#2
	0.50	0.50
x_1	0.00	1.00
y_1	0.00	1.00

To capture the same structure with two pairs of columns requires a four class model:

x_1	y_1	x_2	y_2	Class
0	0	1	1	2
1	1	0	0	3
1	1	1	1	4
0	0	0	0	1
1	1	0	0	3
1	1	0	0	3
...	...	...	...	...

The classes are again equiprobable and have the parameters,

	Classes #1	#2	#3	#4
	0.25	0.25	0.25	0.25
x_1	0.00	0.00	1.00	1.00
y_1	0.00	0.00	1.00	1.00
x_2	0.00	1.00	0.00	1.00
y_2	0.00	1.00	0.00	1.00

The pattern is clear. With $2n$ columns the single hidden feature must allow 2^n possibilities. Justifying such a model requires enough data so that each class appears in the data, at least $O(2^n)$ rows. One actually needs even more. It is a standard combinatorics problem to show that if all classes are equiprobable, one needs $O(n2^n)$ entries before one can expect to see them all. A simple computer program confirms that when $n = 10$, one needs about 7700 data points before all 1024 classes appear.

A far better model uses n hidden features. With four columns, for example,

x_1	y_1	x_2	y_2	Class_1	Class_2
0	0	1	1	1	2
1	1	0	0	2	1
1	1	1	1	2	2
0	0	0	0	1	1
1	1	0	0	2	1
1	1	0	0	2	1
...	...	...	...	...	...

With this model the pattern is perfectly clear after only six rows. The model will have overwhelmingly probability after fifteen or sixteen—irrespective of the number of columns. The important point is that each classification is justified independently of the others, so the exponential blowup never materializes. One can readily construct similar examples with continuous data.

III. Improving the Class Description Function

The goal of this dissertation is to correct this situation so that one can use a Bayesian system like AutoClass for supervised learning tasks. We wish to stay within the framework of density estimation, since it provides intuitive semantics, the ability to make reliability estimates, and a coherent way to deal with missing data. There are two approaches this chapter considers:

1) Improving the class description function.
2) Using a more powerful, multiple hidden feature model space that can capture all the systematic effects in the data.

With a sufficiently powerful class description function one can dispense with hidden features altogether. A class description function that allows direct correlations among discrete features, for example, can describe the identical columns database using a single class. Here is such a class description function:

$$f(\mathbf{z} \mid \phi) = \frac{1}{Z(\phi)} e^{-\sum w_{ij} z_i z_j - \sum w_i z_i} \tag{1}$$

This function fits the identical columns database when $z_{2i} \equiv x_i$, $z_{2i+1} \equiv y$, and the weights depend on a large positive constant C,

$$w_{ij} = \begin{cases} +2C, & \text{if } j = i+1 \\ 0, & \text{if } j \neq i+1 \end{cases} \qquad w_i = -C$$

With these parameters, the exponent in Equation 1 receives a factor of $-C$ every time $x_i \neq y_i$; it is identically zero only if equality holds for all the x_i. The above parameter values therefore lead to a uniform distribution over the possible database rows in which adjacent pairs of columns are identical.

Equation 1 is how a Hopfield net models data [22]. It is also how a Boltzmann machine works [21], with the additional complication that some of the z_i can be hidden. Unfortunately, there is a reason why AutoClass uses the class description functions it does: they are already about as complicated as they can be while remaining computationally feasible. The difficulty is that nobody knows how to calculate the partition function $Z(\phi)$, which is needed to compare different possible values for the parameters w_{ij} and w_i. The Boltzmann machine resorts to numeric integration by stochastic simulation to estimate $Z(\phi)$, although Peterson has developed a much improved algorithm that uses a different technique, the mean field approximation of statistical mechanics [48].

Equation 1 is easy to normalize when all the z_i are continuous, in which case $f(\mathbf{z}|\phi)$ is just an unusual way to parameterize a multivariate normal distribution. In this case continuous integration is considerably easier than discrete summation. In view of the difficulty evaluating $Z(\phi)$ in the discrete case, we shift direction and abandon the idea of using more complicated class description functions than those in AutoClass.

IV. Multiple Hidden Features

The natural way to overcome the limitations of AutoClass is to use a model space that permits multiple hidden features, each of which gives a different, useful way of classifying the data. Such models readily describe all the example databases to this point. In the United Nations database, for example, a multiple hidden feature model could include two hidden features, one for sex and one for ethnicity, rather than being forced to choose one or the other.

The belief net in Figure 1 is one way to turn the idea of multiple hidden features into a definite, mathematical expression for the joint likelihood $\Pr(\mathbf{z} \mid MI)$. It is the obvious extension to the belief nets from the previous chapter, where there are now multiple hidden causes that impact our expectation of the observable features $\mathbf{z}$. The belief net corresponds to the joint probability factorization,

$$\Pr(\mathbf{z}, \text{class}_1 \ldots \text{class}_n \mid I) = \Pr(\mathbf{z} \mid \text{class}_1 \ldots \text{class}_n\, I) \prod \Pr(\text{class}_i \mid I)$$

Unfortunately, this factorization leaves the problematic term $\Pr(\mathbf{z} \mid \text{class}_1 \ldots \text{class}_n\, I)$. In the case of a single hidden feature we introduced a class description function parameterized by the different values the hidden feature could adopt:

$$\Pr(\mathbf{z} \mid \text{class} = i, MI) = f(\mathbf{z} \mid \phi_i)$$

In the case of multiple hidden features we might try to do the same:

$$\Pr(\mathbf{z} \mid \text{class}_1 = h_1, \ldots, \text{class}_n = h_n, MI) = f(\mathbf{z} \mid \phi_{\mathbf{h}})$$

where $\mathbf{h} = h_1 \times \cdots \times h_n$ indexes all the possible combinations of the classes. We see at once, though, that this equation is no different than using a single hidden feature that is a crossproduct of all the distinct ones, so clearly we want something different. One possibility is a noisy-or model [47], which unfolds the belief net of Figure 1 into the more manageable model of Figure 2 using auxiliary propositional variables. Ghahramani has proposed an additive model for the case in which z consists of only continuous features [16], and Musick has written his thesis about using a general purpose conditional estimation model, like a neural net, to represent the troublesome probability [44].

This dissertation takes a different approach. It keeps the graph structure of Figure 1, but drops the directionality of the arrows to give the undirected belief net of Figure 3, known as a

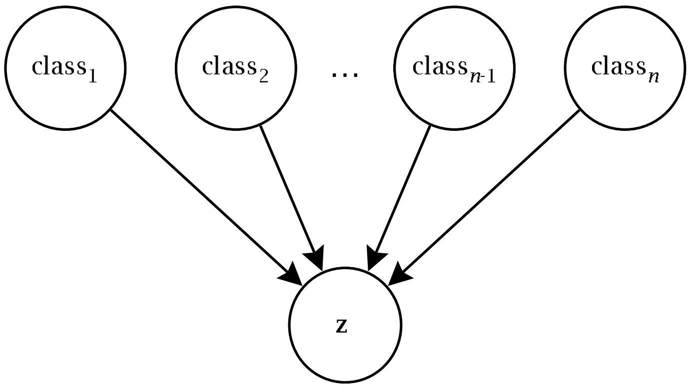


Figure 1. *Multiple Cause Belief Net*

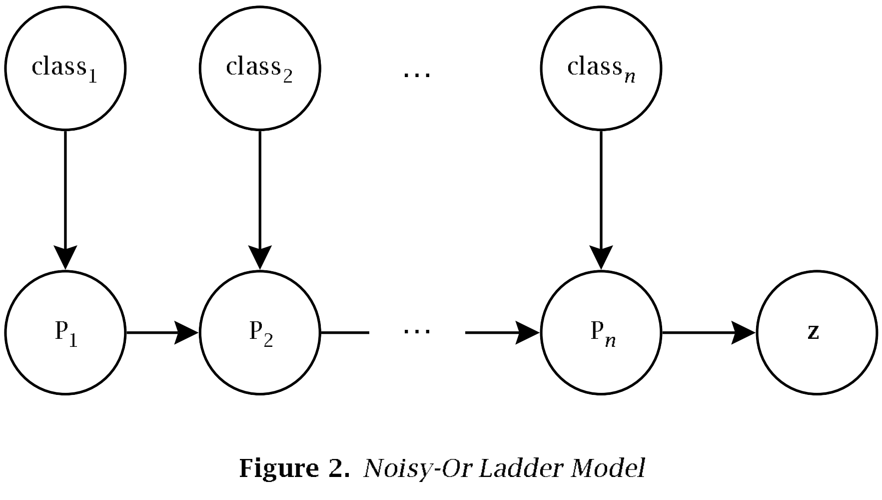


Figure 2. *Noisy-Or Ladder Model*

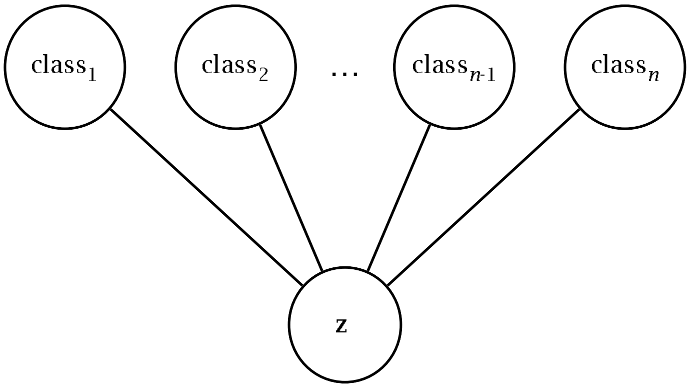


Figure 3. *Multiple Cause Undirected Belief Net*

Markov random field. This model is a maximum entropy model that takes into account correlations among the hidden features and the observables. Pearl's book gives a good introductory description of undirected belief nets [47].

The directed and undirected graphs encode different independence relations. In the directed belief net, the hidden classes are all marginally independent, but conditionally dependent given **z**. This type of graph is typified by the wet grass example of Chapter 4. The chance of rain is independent of the chance the lawn sprinkler is on, but as soon as we see wet grass the two possible causes become negatively correlated. In the undirected graph the situation is reversed: the hidden classes may be marginally dependent, but they are conditionally independent given **z**. This situation resembles the directed net of Figure 4, in which the arrows are reversed from Figure 1. We cannot, however, use Figure 4 directly, since it corresponds to the factorization,

$$\Pr(\mathbf{z}, \text{class}_1 \ldots \text{class}_n \mid MI) = \Pr(\mathbf{z} \mid MI) \prod \Pr(\text{class}_i \mid \mathbf{z}MI)$$

This equation takes us in a circle. It requires us to specify the very quantity we hope to model, $\Pr(\mathbf{z} \mid MI)$, before any calculation can take place. Martin works from this equation in his OLOC system [40], but he only avoids circularity by abducting the most probable class assignments

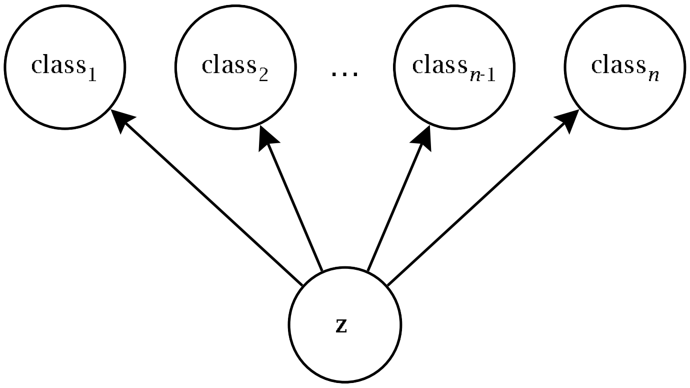


Figure 4. *Reversing the Arcs*

for each **z**, rather than summing over all of them.

The joint probability distribution for an undirected belief net is given by a product of individual factors, one for each maximal clique in the graph. In Figure 3 the maximal cliques correspond to the arcs, so the density function is,

$$\Pr(\mathbf{z}, \mathrm{class}_1 \ldots \mathrm{class}_n \mid MI) = \frac{1}{Z_g} \prod g_i(\mathbf{z}, \mathrm{class}_i) \tag{2}$$

Z_g is a normalizing constant that depends on the functions g_i, which can in general be distinct. In our case we have no reason to treat the hidden features non-uniformly, and also no reason to treat different possible values of a hidden feature non-uniformly. We therefore make the inspired choice,

$$g_i(\mathbf{z}, \mathrm{class}_i = j) = w_{ij}\, f(\mathbf{z} \mid \phi_{ij})$$

The function f is our familiar class description function. Equation 2 now factors nicely to give the marginal probability of $\mathbf{z}$:

$$\begin{aligned}
\Pr(\mathbf{z} \mid MI) &= \sum_{\mathbf{h}} \Pr(\mathbf{z}, \text{class}_1 = h_1, \ldots, \text{class}_n = h_n \mid MI) \\
&= \frac{1}{Z(\phi)} \sum_{\mathbf{h}} \left(\prod_i w_{ih_i} f(\mathbf{z} \mid \phi_{ih_i}) \right) \\
&= \frac{1}{Z(\phi)} \prod_i \left(\sum_j w_{ij} f(\mathbf{z} \mid \phi_{ij}) \right) \qquad (3)
\end{aligned}$$

Equation 3 is a product of individual AutoClass mixture models. There is one term in the product for each hidden feature, and one term in each interior sum for every value a hidden feature can adopt. The sums can be of different lengths, since each hidden feature can partition the data into a different number of classes. Equation 3 collapses to a simple mixture when there is only one hidden feature, and it is therefore a natural way to extend AutoClass to simultaneously allow multiple, independent ways of classifying the data.

Comparing with our original attempt to use the directed belief net in Figure 1, Equation 3 gives the conditional probability,

$$\Pr(\mathbf{z} \mid \text{class}_1 = h_1, \ldots, \text{class}_n = h_n, MI) = \frac{1}{Z(\phi, \mathbf{h})} \prod w_{ih_i} f(\mathbf{z} \mid \phi_{ih_i})$$

This expression is, loosely speaking, what one might expect from trying to combine the effects of different, independent predictions, one for each separate classification, by multiplying their individual predictions together.

The multiple hidden feature model is exactly right for problems like the United Nations data, where the final model is a product of two mixtures, one for sex and one for ethnicity:

$$\begin{aligned}
\Pr(\mathbf{z} \mid MI) = (1/Z(\phi)) &\times (0.6 f(\mathbf{z} \mid \phi_{\text{men}}) + 0.4 f(\mathbf{z} \mid \phi_{\text{women}})) \\
&\times (0.2 f(\mathbf{z} \mid \phi_{\text{African}}) + \cdots + 0.05 f(\mathbf{z} \mid \phi_{\text{Irish}}))
\end{aligned}$$

The weights represent the proportion with which each class appears in the data. The sex classes make strong predictions about height and weight, but are uniform over eye, hair, and skin color. The ethnicity classes are the opposite. When these two mixtures are multiplied together, the peaks combine to predict that there will be African men, Irish women, and all the other possible combinations. This example is far too primitive to illustrate the full expressive power of multiple hidden feature models, since sex and ethnicity are marginally independent. Equation 3 requires no such thing, and in general it will not happen.

V. $Z(\phi)$

Unfortunately, the only known way to evaluate the partition function $Z(\phi)$ exactly is to multiply out the product in Equation 3 at exponential cost. If we cannot evaluate the partition function, we cannot normalize models to compare them and see which is best. The difficulty in computing the partition function was mentioned in Chapter 3. $Z(\phi)$ is a multidimensional integral that has no convenient simplification. We shall presently see a three line proof that it is, in general, NP-hard to evaluate.

There are, of course, many practical problems that are NP-hard, and one dives in with the best methods available. Multidimensional integration is of keen interest to many fields, and there is currently a great deal of research directed towards finding better techniques. Physicists have been attacking partition function problems for over a century, and they have developed many ingenious methods, like the mean field approximation mentioned earlier. Ghahramani reports good results using the mean field approximation in his directed graph system [16], so there is a chance it applies to Equation 3. Finally, multiplying out the product in Equation 3 is feasible if there are few enough terms in the resulting sum. If there are many, a form of beam search that multiplies one factor at a time and discards all but the W most significant classes may give a practical approximation.

Freund and Haussler have explored a special case of Equation 3 they call the harmonium [14]. This model arises when all inputs and hidden features are binary, in which case it reduces to the Boltzmann machine shown in Figure 5. They have no silver bullet to evaluate the partition function, but instead approximate it with the $O(n^k)$ algorithm that sums over the states in which no more than k out of the n hidden features are active. They obtain promising results in a handwritten digit recognition problem, where each hidden feature identifies a line or

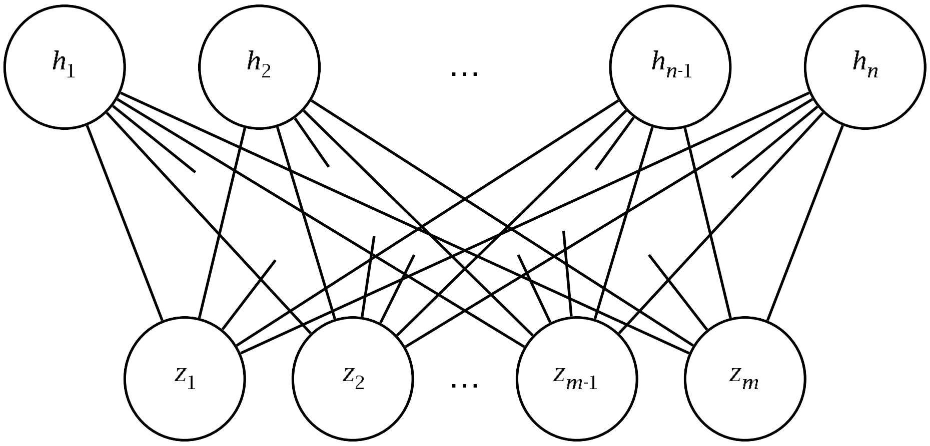


Figure 5. *Harmonium*

stroke, as opposed to a complete digit. Saund has looked at similar letter recognition problem using a different multiple cause model [50]. His model is less firmly grounded in probability theory, but against that he obtains good results and has no partition function to evaluate.

VI. Logical Rules

This section shows how hidden feature models naturally extend logical rules. The end result is a simple, mechanical translation from DNF rules to mixture models, and from sets of DNF rules to multiple hidden feature models. The translation preserves truth and helps clarify what hidden features are doing. Throughout this section, all propositional symbols should be understood as predicates that apply to the objects in a database. The explicit dependence on $\mathbf{z}$ is usually dropped to avoid cluttering the notation.

As a simple example, the thyroid data obeys the rule,

$$\mathsf{Pregnant} \Longrightarrow \mathsf{Woman}$$

If 75% of the people are women, and a third of them are pregnant, then the joint distribution of sex and pregnancy is,

	man	*woman*
pregnant	0.00	0.25
¬pregnant	0.25	0.50

A mixture model that describes this distribution using two equiprobable classes is,

	Classes #1	#2
	0.50	0.50
woman	0.50	1.00
pregnant	0.00	0.50

A natural parallel emerges when we compare the density function of this model to the logical rule written as a disjunction,

$$\Pr(\mathbf{z} \mid MI) = 0.25(1 - \text{pregnant}) + 0.25\,\text{woman}$$

$$\mathsf{R} \iff \neg\mathsf{Pregnant} \vee \mathsf{Woman}$$

The two disjuncts mirror the two classes. A woman that is not pregnant satisfies both halves of the disjunction, just as she can be placed in either class of the mixture model.

In the day camp data, there may not be enough hours in the day for children to play whiffle ball, go swimming, and still have time to do any of the other activities. If not, then the data will obey the logical rule,

$$\mathsf{Swimming} \wedge \mathsf{Whiffle\text{-}Ball} \Longrightarrow \neg\mathsf{Coloring} \wedge \neg\mathsf{Painting} \wedge \neg\mathsf{Singing}$$

This rule expands into a DNF formula with three disjuncts. It maps onto the three class the mixture model,

	Classes #1	#2	#3
	0.33	0.33	0.33
swimming	0.00	0.50	0.50
whiffle ball	0.50	0.00	0.50
coloring	0.50	0.50	0.00
singing	0.50	0.50	0.00
painting	0.50	0.50	0.00

The model satisfies the logical constraint for any class weights. In fact, only the zero probabilities are significant, and all the others and can be freely set to any values between 0 and 1. The correspondence with the logical rule is again most apparent when we write out the probability density next to the expanded rule:

$$\begin{aligned}\Pr(\mathbf{z} \mid MI) = {}& {}^1/_3\,{}^1/_{2^4}\,(1 - \text{swimming}) \\ &+ {}^1/_3\,{}^1/_{2^4}\,(1 - \text{whiffle ball}) \\ &+ {}^1/_3\,{}^1/_{2^2}\,(1 - \text{coloring})(1 - \text{singing})(1 - \text{painting})\end{aligned}$$

$$\mathsf{R} \iff \neg\mathsf{Swimming} \vee \neg\mathsf{Whiffle\text{-}Ball} \vee (\neg\mathsf{Coloring} \wedge \neg\mathsf{Painting} \wedge \neg\mathsf{Singing})$$

In general, one can convert any DNF formula into a mixture model by reinterpreting all the logical elements as arithmetic elements.

Logical Rule $\longmapsto$	Mixture Model
T	1
F	0
$\vee$	$+$
$\wedge$	$\times$
$\neg$	$1 - x$

Each disjunct then becomes a class in which every proposition is either true, false, or unknown, in which case it is equally likely to be either true or false. The mapping preserves the algebraic structure, so the zeros of each class in the mixture model coincide with the zeros of the disjunct from which it arises. Each class assigns a uniform density over the states that satisfy the disjunct. The coefficients on the classes can be chosen arbitrarily to give a properly normalized mixture. The two examples to this point use the coefficients,

$$c_i = \frac{1}{K} \frac{1}{2^{n-m_i}}$$

where K is the number of disjuncts in the logical rule, n is the number of feature columns in the database, and m_i is the number of propositions that appear in the i^{th} disjunct. These coefficients give equiprobable classes.

Letting R be a DNF rule and M_{R} the model that results from the above mapping, a point $\mathbf{z}$ has probability zero if and only if it fails to satisfy any of the disjuncts in R:

$$\Pr(\mathbf{z} \mid M_{\mathsf{R}} I) = 0 \quad \Longleftrightarrow \quad \neg\mathsf{R}(\mathbf{z})$$

One can create a multiple hidden feature model for a set of rules Δ by multiplying the models that arise from each individual rule. The identical columns database satisfies the rules $\Delta = \{\mathsf{X}_i \iff \mathsf{Y}_i\}$, so the corresponding multiple hidden feature model is,

$$\Pr(z \mid MI) = \frac{1}{Z} \prod (x_i y_i + (1 - x_i)(1 - y_i))$$

A zero probability in a multiple hidden feature model M_Δ indicates that at least one of the rules in Δ is not satisfied. If C represents the conjunction of all the rules in Δ,

$$\Pr(\mathbf{z} \mid M_\Delta I) = 0 \quad \Longleftrightarrow \quad \neg\mathsf{C}(\mathbf{z}) \tag{4}$$

The partition function Z_Δ sums the unnormalized probabilities of M_Δ over all possible $\mathbf{z}$. Equation 4 implies that C is satisfiable if and only if this value is non-zero. Z_Δ must therefore be NP-hard to evaluate, since we could otherwise answer 3-SAT questions by converting logical formulae into multiple hidden feature models and calculating partition function values. Equation 4 also shows that M_Δ preserves logical truth and is a consistent probabilistic extension of Δ. For any query Q,

$$\Pr(Q \mid DM_\Delta I) = 1 \quad \Longleftrightarrow \quad D \cup \Delta \models Q$$

M_Δ assigns a probability between 0 and 1 if neither Q nor $\neg Q$ follows from the ground facts in D and the rules in Δ. These probabilities are simple to interpret, but only the person writing down the rules can decide whether they are meaningful. They exhibit some strange, at first counterintuitive properties. Rules are no longer idempotent, for example. The joint density of the model for $\mathsf{A} \vee \mathsf{B}$ is,

	A	¬A
B	0.50	0.25
¬B	0.25	0.00

While the density for $(\mathsf{A} \vee \mathsf{B}) \wedge (\mathsf{A} \vee \mathsf{B})$ is,

	A	¬A
B	0.66	0.17
¬B	0.17	0.00

So the probabilities preserve logical truth, but the exact numbers are sensitive to the syntactic form of the rules.

The mapping in this section may be a good way to incorporate domain knowledge into a probabilistic model. One can imagine a human expert writing down logical rules that are approximately correct, and then having a modeling system use them as a starting point in its search for the optimal model. The search will automatically adjust probabilities away from zero for rules like "birds fly" that are not absolute. The result would be much like Towell's KBANN system [54], but with the potential for considerably greater flexibility, since a multiple hidden feature model can so naturally incorporate a complete set of DNF rules.

6

Supervised Mixture Models

I. MultiClass

This Chapter describes a new modeling system, MultiClass, that acts as a supervised version of AutoClass. MultiClass circumvents the impracticalities of density estimation by performing conditional estimation instead. In this way it focuses on finding the most useful classification for a desired prediction, rather than simply the strongest overall pattern in the data. MultiClass and AutoClass use identical models, class description functions, priors, and noise terms. Only the evaluation criterion used to compare models is different. Whereas AutoClass looks at the joint probability of all the features, MultiClass only considers the conditional probability of features one wants to predict:

$$\begin{aligned} \Pr(\mathbf{y} \mid \mathbf{x} M_{\phi} I) &= \frac{1}{Z(\mathbf{x}, \phi)} \sum_{i=1 \ldots K} w_i\, f(\mathbf{y} \mid \mathbf{x} \theta_i) \\ Z(\mathbf{x}, \phi) &= \sum_{i=1 \ldots K} \int w_i\, f(\mathbf{y} \mid \mathbf{x} \theta_i)\, d\mathbf{y} \end{aligned} \tag{1}$$

The output $\mathbf{y}$ ordinarily consists of a single feature, but it can contain any number of them. MultiClass reduces to AutoClass in the limiting case where it contains them all. Equation 1 implicitly assumes the same joint density whether MultiClass is predicting continuous, discrete, or categorical features. Regression problems and classification problems are treated uniformly. Only the normalizing function $Z(\mathbf{x}, \phi)$ changes to reflect the conditional probability one wants to optimize. Notice that this function normalizes the entire mixture as a whole, rather than each component individually. If it were to normalize each class individually, the conditioning information $\mathbf{x}$ would cancel out of the equations, and one would compute the marginal probability $\Pr(\mathbf{y} \mid M_\phi I)$ instead of the conditional probability $\Pr(\mathbf{y} \mid \mathbf{x} M_\phi I)$.

MultiClass resembles the hierarchical mixture of experts model of Jordan and Jacobs [31, 30] and the earlier mixture of experts model on which it is based [24], but it is distinguished by its Bayesian methodology and roots in density estimation. It is unique in its priors, noise class, ability to handle missing data, and ability to deal with categorical, discrete, and continuous features.

This chapter is the heart of the dissertation, though it builds on everything that has come before. The first part uses examples to show how MultiClass works. The second discusses how the prior and noise class smooth out irregularities in the conditional likelihood function, which is very ill behaved. The last part explains the rationale for MultiClass and the assumptions that underlie conditional estimation.

II. Examples

When predicting sepal length from petal length in the iris data, Equation 1 reduces to a weighted mixture of linear regressions. The left half of Figure 1 shows the optimal two class model, and the right half does the same for the simpler case that ignores intraclass correlations between the two measurements. The weight of each regression is determined by a flower's petal length, so the predicted sepal length follows a smooth transition from one class to the other as petal length increases. The predicted distribution in the transition region is a bimodal mixture of

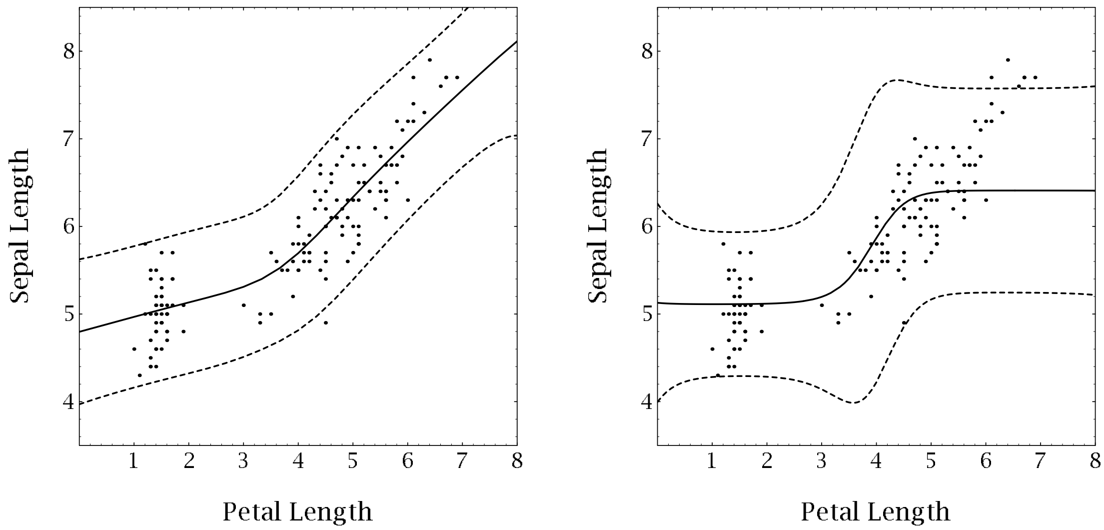


Figure 1. *Two Class Iris Regression*

two gaussians. The numerical parameters of the model without correlations are,

	Classes	
	#1	*#2*
	0.44	0.56
sepal length	5.11 ± 0.40	6.41 ± 0.58
petal length	2.57 ± 0.86	5.59 ± 1.10

Like AutoClass, MultiClass finds this model by doing continuous function optimization over parameterized mixtures of two uncorrelated bivariate normals. But instead of evaluating the joint probability of both measurements, it only calculates the conditional probability of sepal length given petal length:

$$\Pr(y \mid x M_\phi I) = w_1 N(y \mid x \boldsymbol{\mu}_1 \Sigma_1) + w_2 N(y \mid x \boldsymbol{\mu}_2 \Sigma_2)$$

The normalizing function is easy to evaluate in the case where the class description function is a product of independent distributions. It is also easy in the more general case that takes

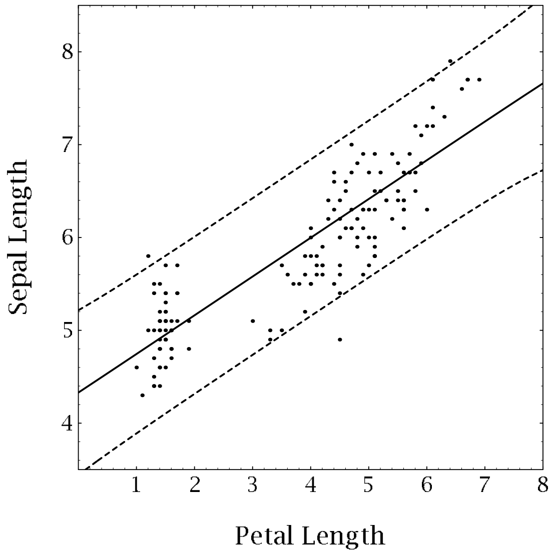


Figure 2. *One Class Iris Regression*

into account correlations among continuous features, since the conditional projection of a multivariate normal distribution is itself a multivariate normal distribution.

Figure 2 demonstrates that a one class model fits the iris data well as long as correlations are considered, and it helps emphasize how easy it is for a database to contain patterns that are unimportant to the predictions one cares about. In this case, although the two visible clusters affect the marginal distribution of both petal length and sepal length, they have only a minor impact on the conditional probability of one given the other.

MultiClass finds the following two class model in the day camp data to predict whether children will go swimming:

	Classes	
	sun	*rain*
	0.61	0.39
swimming	0.88	0.12
whiffle ball	0.67	0.08
coloring	0.33	0.79
singing	0.31	0.39
painting	0.22	0.76

The day camp data is generated from a two class mixture, so it is not surprising that this model is virtually indistinguishable from the one that AutoClass finds, presented in Chapter 4. If the children paint and play whiffle ball, but do not color or sing, then the probability that they will also go swimming is,

$$\begin{aligned}
\Pr(\text{day} \mid \text{sun}\, MI) &= (0.67)(1-0.33)(1-0.31)(0.22) = 0.0681 \\
\Pr(\text{day} \mid \text{rain}\, MI) &= (0.08)(1-0.79)(1-0.39)(0.76) = 0.0078 \\
\text{Odds}(\text{sun}) &= \frac{0.61 \times 0.0681}{0.39 \times 0.0078} = 13.68 \\
\Pr(\text{sun} \mid \text{day}\, MI) &= \frac{13.68}{1+13.68} = 0.93 \\
\Pr(\text{swimming} \mid \text{day}\, MI) &= (0.93)(0.88)+(0.07)(0.12) = 0.83
\end{aligned}$$

Each observation is as an independent test that divides its vote among the two classes. Whiffle ball is a strong test, because it has a very different probability in each class; singing is a weak test, because its probability is almost the same in each class. Picturing each observation as an independent test is a good way to understand how a supervised mixture model works. In the general case of K classes, each test splits up its vote among them all.

AutoClass is designed for problems like the iris and day camp data, and the advantage of MultiClass only materializes when there are multiple useful classifications, like in the United Nations data or the thyroid data. Here is a hypothetical completion of the thyroid data that lets one predict which patients suffer from hypothyroidism:

Sex	Pregnant?	Lab Test	Hypothyroid?
M	–	98.6	+
F	×	30.1	–
F	–	56.4	+
M	–	90.2	+
F	–	21.1	–
F	×	23.9	–
...	...	...	...

Sex and pregnancy are independent from the lab test and hypothyroid features. A normal thyroid usually gives a low lab test result, a deficient thyroid usually a high one. MultiClass

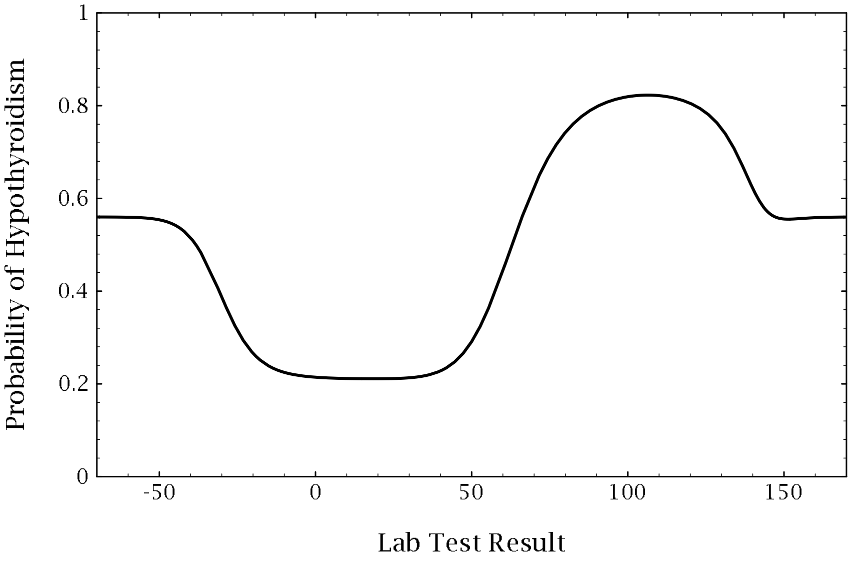


Figure 3. *Hypothyroid Predictions*

classifies people according to either sex or their lab test result depending on whether it is asked to predict pregnancy or hypothyroidism. The model it finds to predict hypothyroidism is,

	Classes #1	#2
	0.47	0.53
male	0.51	0.53
pregnant	0.27	0.26
lab test	54.2 ± 20.4	81.6 ± 14.1
hypothyroid	0.21	0.89

This model ignores the logical rule connecting sex and pregnancy, and neither feature influences a patient's classification. Figure 3 shows how hypothyroid predictions vary smoothly between a probability of 0.21 and 0.89 as the lab test result increases. The noise class leads to a conservative prediction when the lab value is unreliable. The model MultiClass finds to predict pregnancy is,

	Classes #1	#2
	0.62	0.38
male	0.84	0.02
pregnant	0.01	0.66
lab test	69.7 ± 21.2	71.1 ± 22.4
hypothyroid	0.59	0.52

Now the lab test and hypothyroid features are irrelevant. Ignoring them, one can calculate the probability of pregnancy using a computation like the one shown in full for the day camp data. The result is,

$$\Pr(\text{pregnant} \mid \text{sex} = \text{male}, MI) = 0.02$$

$$\Pr(\text{pregnant} \mid \text{sex} = \text{female}, MI) = 0.51$$

The empirical frequencies are 0% and 51%. The probability of seeing a pregnant man differs from zero due to the priors. In general, although a database may contain many interesting classifications, MultiClass focuses on the one that is most useful for making a given prediction.

III. Missing Data and Reliability

One can interpret a supervised mixture model as a density model that ignores unimportant patterns. This interpretation allows it to handle missing data and avoid unreliable predictions in a way that an entirely conditional model, like a decision tree or a feed forward neural network, cannot.

MultiClass marginalizes missing data as described in Chapter 3. For the class description functions that it uses the result is elegant: missing features are simply erased from the model before making predictions. In the day camp data, if there is a day for which the coloring and singing features are missing, MultiClass will eliminate them and use the model,

	Classes	
	sun	*rain*
	0.61	0.39
swimming	0.88	0.12
whiffle ball	0.67	0.08
painting	0.22	0.76

If one views each feature as an independent test, the tests for missing features are simply not included. This approach is very different from assigning a missing feature its most probable value, or even its expected value.

Purely conditional models cannot assess the reliability of conditioning information, leaving them susceptible to making unreliable predictions. MultiClass uses the noise class described in Chapter 4 to alleviate this problem. In the iris data, an imaginary flower with a petal length of 30 cm is caught by the noise class, resulting in a diffuse prediction of its sepal length. Since MultiClass ignores parts of the joint structure, its reliability estimates are less conservative than those that arise from a complete joint density. MultiClass will only notice an unreliable observation that impacts its predictions. When predicting which patients suffer from hypothyroidism, for instance, it will not even blink at a pregnant man.Figure 4 indicates what to expect. In the data on the left, x is irrelevant to y, and MultiClass finds a one class model that catches unreliable x values at the extremes, but misses the dead region in the center. In the data on the right, MultiClass finds a two class model that notices all the places where x is unreliable. A true density model would describe both data sets using two classes and spot all the unreliable observations. MultiClass derives its strength from dismissing parts of the joint structure that are irrelevant to its predictions. This focus results in sharper predictions, but only at the expense of decreased reliability.

IV. Smoothing the Conditional Density

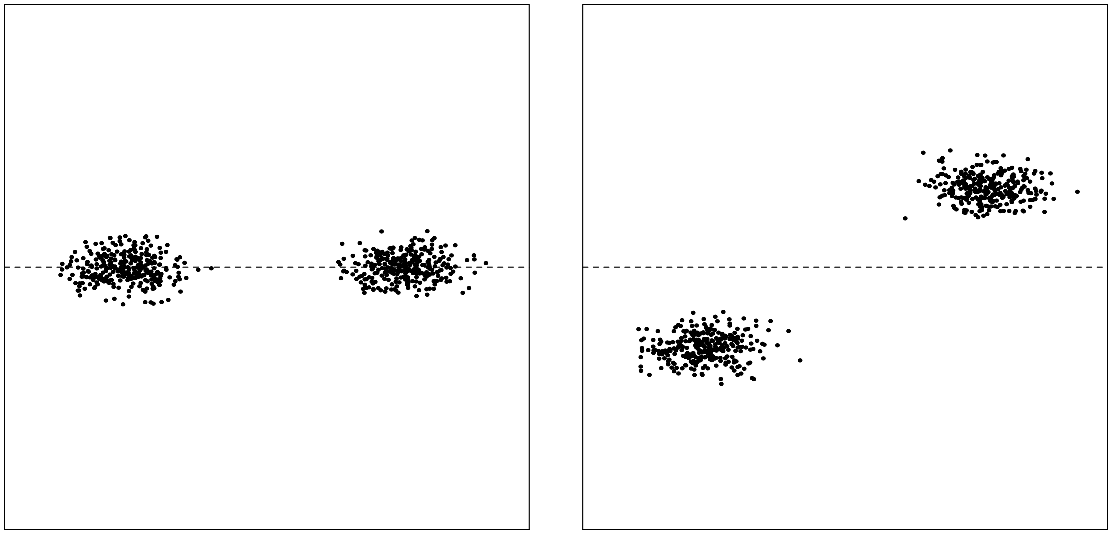


Figure 4. *Reliability in a Supervised Mixture Model*

The greatest challenge that MultiClass faces are the irregularities in the conditional likelihood function of Equation 1. There are many joint models that give rise to the same conditional probability distribution, so a joint model is badly overparameterized when optimized for a conditional probability. Figure 5 illustrates using a bivariate normal density in two dimensions. The model can be shifted along the regression line of y from x, or appropriately stretched and rotated, without affecting $\Pr(y \mid xMI)$. The conditional distribution has only three degrees of freedom, whereas the joint distribution has five, so optimizing the conditional likelihood leads to a family of solutions lying along a two dimensional manifold. This ridge confuses second order optimization algorithms, which wander along it afraid to terminate. Much worse, the lack of determinate parameter values casts doubt on our ability to interpret the final model as a legitimate joint density, which is necessary to handle missing data and reliability correctly.

The hypothyroid model characterizes a common way that unwanted parameters arise. There is no reason why sex must be 50-50 within each class, since it will cancel from the conditional likelihood as long as it has the same distribution in both of them. The ratio could just as easily be 70-30 or 20-80. Any irrelevant feature follows this same pattern: the conditional distribution is unaffected as long as the feature has the same distribution in each class.

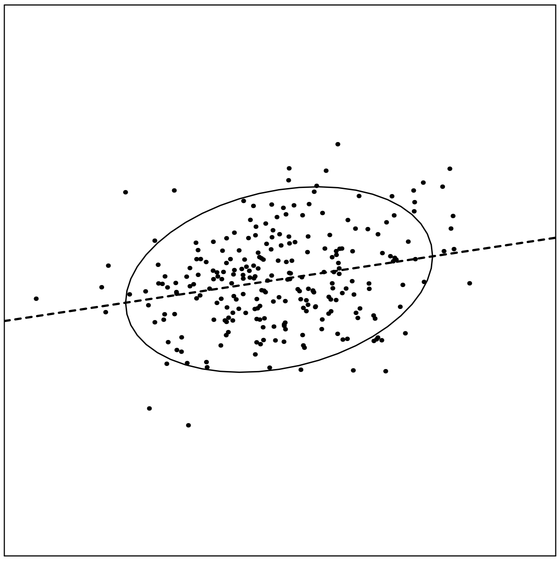


Figure 5. *An Overparameterized Model*

The noise class and priors in MultiClass are essential to overcome underdetermined parameters. They give shape to regions where the conditional likelihood is flat and force unused parameters to take on meaningful values. The location of the noise class is fixed by the empirical data, so it establishes an absolute coordinate system for models that could otherwise be shifted arbitrarily. A mixture class becomes useless if it wanders too far away from the points it is intended to describe. In Figure 5, for example, the noise class governs the predictions if the normal density is shifted too far in either direction, so the optimal solution is the one shown, centered over the data, minimizing the effect of the noise class. The noise class also fulfills its usual role of capturing outliers, which are particularly nasty in conditional estimation and lead to singularities in the conditional likelihood function when $Z(\mathbf{x}, \phi)$ goes to zero.

The prior has an even stronger effect. The prior presented in Chapter 4 for the class parameters θ is,

$$\Pr(\theta \mid PI) \propto \Pr(D \mid \theta I)^{\frac{P}{N}}$$

This function still treats each class as a density model. So while MultiClass optimizes conditional probabilities, its prior optimizes joint probabilities and forces extraneous parameters towards sensible values. In the normal density of Figure 5, the prior alone is enough to guide MultiClass to the natural solution centered on the empirical mean and variance of x. In the thyroid model, the prior leads to a distribution of sex within each class that matches the 50-50 ratio of men to women in the database.

The prior works well for the class parameters, but it is ineffective for the class weights. In AutoClass the weights naturally settle on the proportion with which each class appears in the data. In MultiClass they are underdetermined and may converge to strange values, particularly if the classes are well separated. When predicting the first column of the identical columns database, for example, MultiClass finds the obvious model:

	Classes #1	#2
	?	?
x_1	0.00	1.00
y_1	0.00	1.00

The class weights are irrelevant, because the x measurement is always perfectly informative. The likelihood function for the weights is therefore flat, and one needs either an informative prior or some other technique to set the weights to appropriate values. In the iris data, the version of MultiClass described to this point finds the following model:

	Classes #1	#2
	0.76	0.24
sepal length	5.12 ± 0.40	6.42 ± 0.58
petal length	2.35 ± 0.81	5.21 ± 1.16

Notice the strange class weights. They do not reflect the proportion with which each class occurs, so this model generates an inaccurate marginal distribution of sepal length that does not correctly handle a missing petal length. MultiClass corrects the problem of ill-determined class weights by running a two pass optimization algorithm:

1) Find the optimal model.
2) Calculate how often each class appears in the data.
3) Fix the weights to the observed proportions, and reoptimize the remaining parameters.

In the iris data, the first step finds the above model with class weights $w_1 = 0.76$ and $w_2 = 0.24$; the second step applies this model to the data and computes the frequency with which each class appears, $f_1 = 0.44$ and $f_2 = 0.56$; the third step fixes the weights to these frequencies and reruns the optimizer, finding the model at the beginning of the chapter. The effect of this algorithm is to choose the model along the optimal ridge where the weights reflect the empirical frequency with which each class occurs.

V. Supervised Mixtures vs. Multiple Hidden Features

Accurate density estimation requires multiple hidden features, so it is natural to wonder if a supervised mixture model, which uses only a single hidden feature, is powerful enough to make good predictions. The correspondence between logical rules and hidden features, presented in Chapter 5, reveals that MultiClass captures an impressive scope of predictions. Suppose one is analyzing a propositional database with the goal of predicting a boolean feature P roughly described by the logical rule,

$$\mathsf{P} \iff \mathsf{R}$$

where R is a conjunction of n observable features. This rule expands into the DNF formula,

$$(\mathsf{P} \wedge \mathsf{R}) \vee (\neg \mathsf{P} \wedge \neg \mathsf{R}) \tag{2}$$

The strategy is to express Equation 2 as a two class supervised mixture model. One class will act as a default capturing the situations where P is false, and the other will describe the cases where both R and P are true. The desired probabilistic model M_R is,

	Classes #1	#2
	0.50	0.50
R	1.00	0.50
P	1.00	0.00

In the day camp data, if it were the case that children paint if and only if they color and sing, then a model to predict painting would be,

	Classes #1	#2
	0.50	0.50
swimming	0.50	0.50
whiffle ball	0.50	0.50
coloring	1.00	0.50
singing	1.00	0.50
painting	1.00	0.00

A day on which the children do not color and sing can only fall into the second class, so they will not paint either. This model only imperfectly describes the logical rule in the other direction, and the generic probabilistic model does not encode Equation 2, but rather its close approximation,

$$(P \wedge R) \vee \neg P \tag{3}$$

The advantage of dropping $\neg R$ is that it is a disjunction, and therefore hard to express using the conjunctive class description functions in MultiClass. The conjunction R is easy. The model M_R always predicts $\neg P$ if R is false, since class #1 then has zero probability, and it assigns a high probability to P if R is true:

$$\Pr(\mathsf{P} \mid \neg\mathsf{R} M_{\mathsf{R}} I) = 0$$
$$\Pr(\mathsf{P} \mid \quad \mathsf{R} M_{\mathsf{R}} I) = \frac{1}{1 + 2^{-n}}$$

This model always predicts P correctly, and its predictions become increasingly sharp as n increases. Moreover, we can make the predictions arbitrarily sharp by allowing non-uniform class weights, where class #2 has a very low one. In the day camp example with equal weights, $n = 2$ and the model predicts there is an 80% chance that children will paint if they color and sing. This probability increases to 99% if the weight of class #2 is decreased from 0.50 to 0.04.

More generally, if R expands into a DNF formula with k disjuncts, then Equation 3 expands into a disjunction with $k + 1$ terms that maps onto a supervised mixture model with $k + 1$ classes. The first k classes cover the cases where R and P are both true, and the last class acts as a default that catches the remaining situations where P is false. By lowering the weight on the default class, the model can approximate Equation 2 as closely as one desires.

We can use an equivalent construction if R is a disjunction of n observable features rather than a conjunction. In this case $\neg$R is a conjunction, so one drops R from Equation 2 rather than $\neg$R and follows the same steps as before, only with P and R negated. The resulting model always correctly predicts P is true if R is true, and it assigns a low probability to P if R is false:

$$\Pr(\mathsf{P} \mid \quad \mathsf{R} M_{\mathsf{R}} I) = 1$$
$$\Pr(\mathsf{P} \mid \neg\mathsf{R} M_{\mathsf{R}} I) = \frac{2^{-n}}{1 + 2^{-n}}$$

If R expands into a CNF formula with k disjuncts, then it maps onto a $k + 1$ class supervised mixture model in which the default case is for P to be true. By lowering the weight on the default class, this model too can approximate Equation 2 as closely as one desires.

The conclusion is that a supervised mixture model can compactly express any predictive rule that has either a simple DNF or CNF representation. It is very hard to construct natural examples that do not have this form. Compressing the notation a bit, any such example will have a form like,

$$\mathsf{P} \iff \mathsf{AB} \vee \mathsf{CD} \vee \mathsf{EF} \vee (\mathsf{G} \vee \mathsf{H})(\mathsf{I} \vee \mathsf{J})(\mathsf{K} \vee \mathsf{L})$$

The reader is invited to imagine a real application where such a rule would arise.

A decision tree has trouble with disjunctive rules, because it must replicate each disjunct down multiple paths in the tree [46]. MultiClass is closer to a feed forward sigmoid neural net with two hidden layers, where each node in the second layer corresponds to a class in the mixture model.

VI. Density vs. Conditional Estimation

This section derives the implicit assumption of conditional estimation. A little new notation is required. For any density model M over $\mathbf{xy}$, $M_{\langle \mathbf{y}|\mathbf{x} \rangle}$ will denote the projected model that makes conditional predictions of $\mathbf{y}$ from $\mathbf{x}$. Similarly, $M_{\langle \mathbf{x} \rangle}$ will denote the model that makes marginal predictions of $\mathbf{x}$ alone. The three models are linked by the defining equation,

$$\Pr(D \mid MI) = \Pr(D \mid M_{\langle \mathbf{y}|\mathbf{x} \rangle} I) \Pr(D \mid M_{\langle \mathbf{x} \rangle} I) \tag{4}$$

The first factor on the right gives the probability of the $\mathbf{y}$ values in the data. The second gives the probability of the $\mathbf{x}$ values.

With this notation in hand, Figure 6 contrasts the two alternatives of density estimation and conditional estimation. Both start in the upper left corner with joint models over $\mathbf{x}$ and $\mathbf{y}$ and work towards a model in the lower right corner that predicts $\mathbf{y}$ from $\mathbf{x}$. Density estimation follows the top path. It first finds the optimal joint model M^*, and then it conditions on $\mathbf{x}$ to make predictions using $M^*_{\langle \mathbf{y}|\mathbf{x} \rangle}$. Conditional estimation follows the bottom path. It first conditions on $\mathbf{x}$ so that it can make predictions in the space $M_{\langle \mathbf{y}|\mathbf{x} \rangle}$, and then it finds the optimal conditional model $M_{\langle \mathbf{y}|\mathbf{x} \rangle}{}^*$ from the data. Exchanging the induction and conditioning steps like this strips away unimportant details at the beginning of the problem, before doing any analysis.

Both paths in Figure 6 are valid, but conditional estimation discards information, so the two paths generally arrive at different final answers. Symbolically, $M^*_{\langle \mathbf{y}|\mathbf{x} \rangle} \neq M_{\langle \mathbf{y}|\mathbf{x} \rangle}{}^*$. Conditional estimation neglects the value of learning about the structure of $\mathbf{x}$. Depending on one's prior information, that knowledge may help reveal details about $\mathbf{y}$ and about how $\mathbf{x}$ and $\mathbf{y}$ are related.

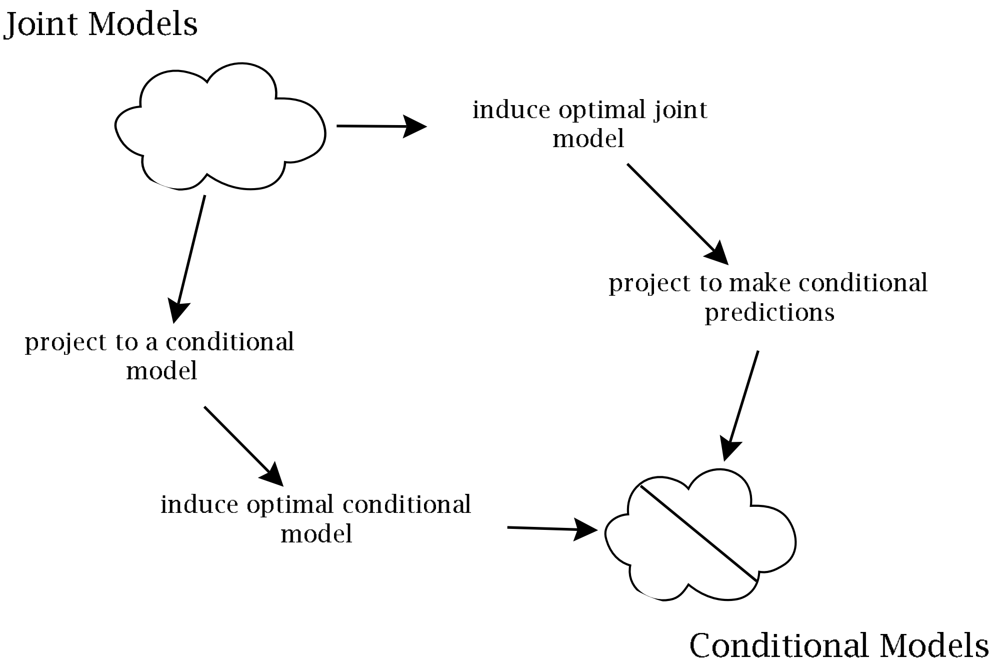


Figure 6. *Two Approaches to Learning*

Suppose, for instance, one looks at the daily record of a major league baseball team for the first few weeks of the season*,

Game Time	Result
1:20	W
1:20	W
1:20	L
7:30	W
1:20	W
. . .	. . .

The team is unknown. The goal is to predict how many games it will win and how likely it is to win the World Series.

Although the data show no connection between a game's starting time and its outcome, the preponderance of day games is a valuable clue. Most baseball games are at night. Only

*12 March 1995: 213 days into the strike and counting.

the Chicago Cubs regularly play in the afternoon, so they are therefore the unknown team the database describes. Pity the poor Cub fans! The Cubs have not won a World Series in 90 years, and one can safely predict that they will not win it again this year, despite their promising start. So while a game's starting time is irrelevant to its outcome, the structure in the starting time data has a noticeable impact on predictions.

Density estimation catches the structure in the baseball data. Conditional estimation misses it. In general, the two approaches give identical predictions as long as the posterior distribution in the joint model space factors along the lines of **x** and **y**:

$$\Pr(M \mid DI) = \Pr(M_{\langle \mathbf{y}|\mathbf{x} \rangle} \mid DI)\, \Pr(M_{\langle \mathbf{x} \rangle} \mid DI) \tag{5}$$

Equation 5 states that $\Pr(M \mid DI)$ is the product of two independent functions. If it holds, then one can find the optimal joint model $M^*$by optimizing each of these functions separately. Equality holds:

$$M^*_{\langle \mathbf{y}|\mathbf{x} \rangle} = M_{\langle \mathbf{y}|\mathbf{x} \rangle}{}^*$$

$$M^*_{\langle \mathbf{x} \rangle} = M_{\langle \mathbf{x} \rangle}{}^*$$

Equation 5 is a sufficient condition for conditional estimation and density estimation to give identical answers. Graphically, the two paths in Figure 6 reach the same final model whenever the overall problem of finding M^* breaks up into two independent halves: finding the conditional model $M_{\langle \mathbf{y}|\mathbf{x} \rangle}{}^*$, and finding the density model $M_{\langle \mathbf{x} \rangle}{}^*$. The condition under which Equation 5 holds can be found by applying Bayes' theorem to each part of Equation 4 and simplifying. The result is,

$$\frac{\Pr(M \mid DI)}{\Pr(M \mid I)} = \frac{\Pr(M_{\langle \mathbf{y}|\mathbf{x} \rangle} \mid DI)}{\Pr(M_{\langle \mathbf{y}|\mathbf{x} \rangle} \mid I)}\, \frac{\Pr(M_{\langle \mathbf{x} \rangle} \mid DI)}{\Pr(M_{\langle \mathbf{x} \rangle} \mid I)}$$

Equation 5 implies equality in the numerators, which occurs if and only if there is also equality in the denominators,

$$\Pr(M \mid I) = \Pr(M_{\langle \mathbf{y}|\mathbf{x} \rangle} \mid I)\, \Pr(M_{\langle \mathbf{x} \rangle} \mid I) \tag{6}$$

Equation 6 is the general assumption of conditional estimation. The baseball data is one example where it does not hold. The two halves of the problem are not independent, since learning about the starting times gives information about the team, which in turn influences how many games one expects the team to win. Equation 3 is an ignorance condition, and it seems reasonable to assume that it is roughly true in any database that is not perfectly understood. For a poorly understood database, the direct evidence of the data should always quickly overwhelm the indirect evidence of the prior and any effect of learning $M_{\langle \mathbf{x} \rangle}$.

VII. Generalizing MultiClass

Supervised mixture models are simple, powerful, and easy to use and understand. They have a coherent probabilistic semantics, and they naturally generalize existing, well understood ideas related to mixture models. But Figure 6 suggests that MultiClass is only an instance of a much more complete algorithm for data analysis. Namely,

1) Write down a parameterized joint probability model for a domain.
2) Write down priors for the parameters.
3) Optimize just the parts of the model that are needed to make predictions.

This algorithm follows the bottom path in Figure 6 rather than the top path, so that one conditions and marginalizes before doing any work. As long as the quantity to be predicted is one dimensional, the resulting conditional estimation is straightforward, as described in Chapter 3, and it eliminates all the practical problems of density estimation while retaining its clear semantics.

7

Experimental Results and Future Work

I. Hyperparameters

Chapter 4 finessed the question of how to set the four hyperparameters that appear in MultiClass's prior and noise class. The user must pick values. The defaults are,

$P = 1$	– an effective prior size of one data point
$\alpha = K$	– a uniform prior over the K classes
$\epsilon = 0.001$	– probability of noise
$\delta = 3.0$	– standard deviation of continuous features in the noise class

These values give good performance. In principle, one could provide priors for the hyperparameters and optimize them as part of the search. In practice, one can use a wrapper algorithm like C4.5-AP [34] to automatically find their optimal values using cross validation. [4].

The user must tell MultiClass how many classes to look for. The program often eliminates unwanted classes by reducing them to zero weight, but this procedure is unreliable. The current priors are much too weak, and MultiClass often settles on a model with too many classes. A horseshoe shaped prior with $\alpha < K$ would help reduce this problem, but it has proven impractical in practice due to the infinite poles that occur at extreme weights. A complete look at overfitting appears at the end of this chapter. MultiClass should be able to find the optimal number of classes with improved priors, but at present it is best to determine the right number through cross validation.

II. Optimization

MultiClass uses a general purpose quasi-Newton optimization package, NPSOL, developed by the Stanford Systems Optimization Lab [17]. This algorithm improves upon gradient descent by incrementally building a Hessian approximation [18] from the gradient information, and its only real disadvantage is that it requires $O(n^2)$ memory, where n is the number of parameters in the model. This restriction limits MultiClass to no more than a few hundred parameters, though the program is written to allow alternative optimization algorithms, like conjugate gradient, that do not have this problem. Jordan and Jacobs have developed an EM algorithm for their HME architecture [31, 32] which it would also be desirable to implement.

Rumelhart finds that second order optimization algorithms, like NPSOL, are sometimes ironically hindered by their far sighted view of the local topology [49]. They can descend into local minima that a less powerful algorithm, like gradient descent, would never see. Priors smooth the terrain and eliminate most of this effect, but one must still be careful. Figure 1 illustrates a common danger. A quasi-Newton optimizer starting from this model will immediately shrink the top left class to zero weight. That class describes none of the data, so getting rid of it is the quickest way to improve the likelihood function.

Multiclass avoids this pitfall by running its optimizer twice. The first pass uses $\alpha > n$, giving a bell shaped prior on the class weights that forces MultiClass to use all of its classes. After the search converges, the second pass continues with $\alpha = n$, giving a uniform prior that lets MultiClass eliminate unnecessary classes. In situations like Figure 1, this optimization

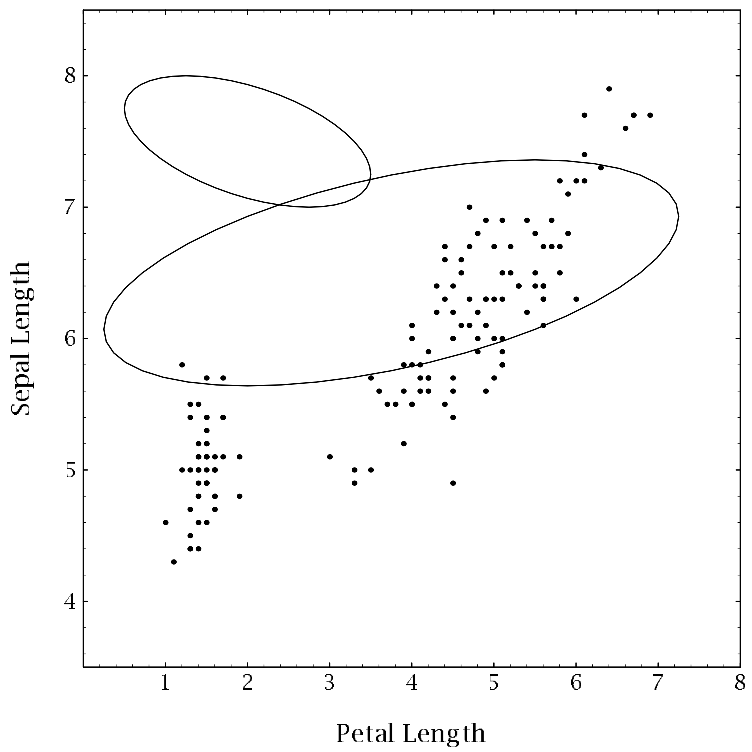


Figure 1. *Optimization Pitfalls*

strategy forces the program to first move all its classes to useful locations before deciding which ones to keep.

III. Evaluating Learning Programs

Programs that make categorical predictions are currently compared based on classification accuracy. This measure has the advantage of being a single number that is easy to understand: "Hey Ronny! I just got 84% on the Pima diabetes database!" Unfortunately, it can fail to distinguish guesswork from confident predictions. Imagine a doctor diagnosing the dreaded purple polka-dot syndrome. Half the time he is sure his patients are healthy. The other half he frankly has no idea and flips a coin. The joint distribution of his predictions and results is,

	sick	*healthy*
definitely healthy	0.00	0.50
unknown	0.25	0.25

The doctor is never wrong if he says a patient is definitely healthy, and he is right half the time when he flips a coin. His overall success rate is 75%. A charlatan across the street charges half the price but always tells his patients they are healthy. Amazingly, he too is right 75% of the time, since as the joint distribution above shows, 75% of the patients are healthy. Who should get more business?

Accuracy cannot distinguish the doctor from the charlatan, but a probability measure easily tells them apart. In a typical test set of 100 patients, the doctor will diagnose 50 of them as definitely healthy and guess on the remaining 50. The charlatan, who says all patients are 75% likely to be healthy, will be right on 75 of them and wrong on the other 25. The probability that each assigns to the test set is,

$$\begin{aligned}
\log \Pr(T \mid \text{Doctor } I) &= 50 \log 1.00 + 50 \log 0.50 = -34.7 \\
\log \Pr(T \mid \text{Charlatan } I) &= 75 \log 0.75 + 25 \log 0.25 = -56.2
\end{aligned}$$

The doctor's predictions are much sharper. By this measure he clearly understands much more than the charlatan.

The log probability of a test set is a natural metric with which to compare machine learning programs, but there is an even better one. The charlatan that always guesses provides a natural baseline against which to compare different programs. Taking the ratio of the charlatan's score to a program's score factors out both the size of the test set and the logarithm base, and it gives an indication of how much useful information a program provides. We call this number the "compression ratio," since it roughly measures the factor by which a program's model could compress the predicted feature relative to a naive program that always guesses. For the doctor,

$$\text{compression ratio} = \frac{-56.2}{-34.7} = 1.62$$

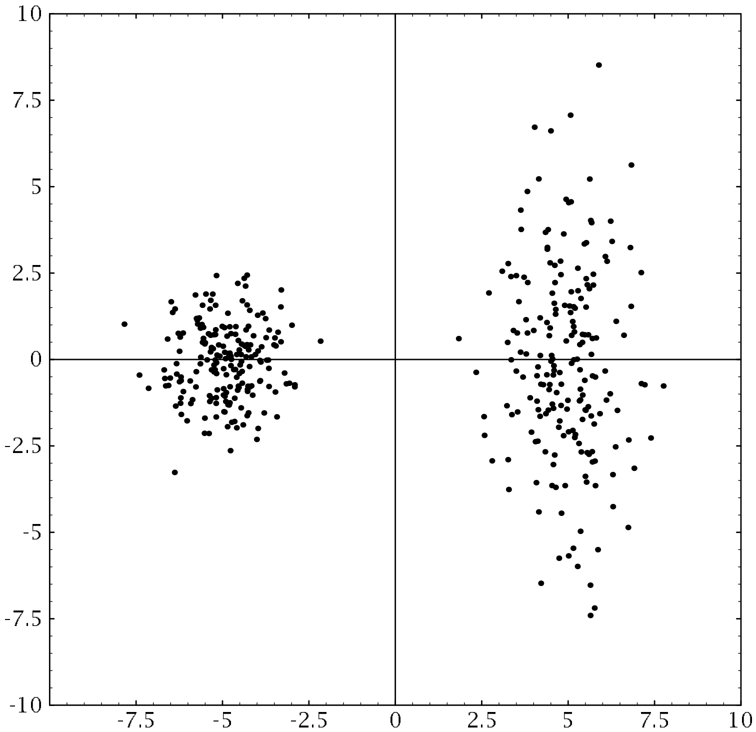


Figure 2. *Failures of Mean Squared Error*

Higher numbers are better. A compression ratio less than one indicates a program is performing worse than random chance.

Programs that make continuous predictions suffer from a similar reporting problem. The usual comparison metric is mean squared error, but Figure 2 shows a simple example where any reasonable program will guess $y = 0$ for all x, but some are really much better than others because they dynamically adjust the variance of their predictions. Like accuracy, mean squared error fails to consider how sharp of a prediction a model is willing to make. The compression ratio corrects this problem just as it does for classification. For continuous predictions, the baseline model is a normal distribution with the empirical mean and variance of the feature to be predicted. MultiClass achieves a compression ratio of 1.78 for the data in Figure 2, despite generating the same mean squared error as the baseline program.

The distinction that currently prevails between classification problems and continuous prediction problems seems artificial. For probabilistic models there is no difference, and MultiClass works identically no matter what feature in the data it is trying to predict. One of the nice properties of the compression ratio is that it provides a single, common, easy to understand number that unifies all types of prediction problems.

IV. MLC++ Data Sets

The experiments tested MultiClass on the collection of databases distributed with MLC++, a machine learning toolbox being developed at Stanford University [35]. The data sets include most of those from the UC Irvine repository [43], the Statlog project [41], and Thrun's monk data sets [52]. They are all classification problems. There are no experiments involving continuous predictions due to a lack of well established benchmarks. No features were added to explicitly indicate missing features, as described in Chapter 3, so none of these experiments account for the possibility that '?' is itself an informative value. This omission may be significant in problems like the diabetes and horse-colic databases where a sizable fraction of the entries are missing.

Three changes were made to the data distributed with MLC++. The pima database clearly contains missing values that have been replaced by zeros, and MultiClass was given the undoctored data where the zeros were changed back to '?'. The new-thyroid database was randomly split into a training and a test set, since none were provided, and the lenses database was resplit. This database is tiny, and the original training and test sets in MLC++ are highly dissimilar.

Figure 3 summarizes the results across two pages. The number of classes was increased for each data set until the compression ratio on the test set began to rise, indicating that performance was growing worse. The reported results are for the optimal number of classes. Cross validation on the training set is sufficiently bullet proof that it could find this number automatically. On data sets with few features, like glass, MultiClass automatically eliminates unwanted classes and converges to the optimal number.

The monk1 and corral data sets are artificial and correspond to simple DNF rules. MultiClass finds the exact models given by the construction in Chapter 6, so the observed compression ratios are artifacts of the prior and can be made arbitrarily large by lowering P and ϵ.

Database	% Accuracy C4.5-AP	% Accuracy MultiClass	Compression Ratio Train	Compression Ratio Test
australian	85.1	86.5	2.52	1.83
balance-scale	—	97.6	7.99	7.18
breast-cancer	73.9	73.7	1.24	1.10
breast	96.3	95.7	10.03	5.25
chess	99.6	99.6	31.11	22.89
cleve	75.6	81.1	3.66	1.50
corral	100.0	100.0	4.05	4.00
diabetes	75.3	74.2	1.55	1.27
flare	—	84.0	1.30	1.09
german	72.7	77.5	1.62	1.21
glass	68.2	68.1	1.92	1.56
glass2	74.9	80.0	1.97	1.35
hayes-roth	—	85.7	5.00	4.43
heart	82.2	87.8	2.94	1.80
hepatitis	84.5	80.8	8.84	0.66
horse-colic	84.2	83.8	3.16	1.21
iris	95.3	96.0	14.41	10.38
labor-neg	80.1	94.1	10.16	2.45
lenses	—	75.0	2.08	1.67
lymphography	74.1	84.0	6.77	2.03
monk1	100.0	100.0	9.71	8.10
monk2	62.5	93.5	2.93	2.25
monk3	97.2	97.2	5.77	5.57
new-thyroid	—	98.6	9.45	10.10
segment	96.8	95.7	18.38	12.60
shuttle-small	—	99.4	31.05	14.42
soybean-small	100.0	100.0	11.44	9.88
tic-tac-toe	93.7	97.9	6.47	5.37
tutorial	—	91.7	2.59	1.98
vehicle	72.4	77.7	4.24	2.76
vote-irvine	95.4	95.2	18.01	6.49
vote	—	99.3	13.07	15.87
vote1-irvine	87.6	91.0	5.93	2.78
vote1	—	95.6	4.98	4.05
xd6	—	100.0	22.5	22.69
zoo	—	70.6	3.24	1.99

Figure 3. *Experimental Results*

Database	*Objects in Database* *Train*	*Test*	*Classes Found*	*Range of Output*
australian	460	230	2	2
balance-scale	416	209	3	3
breast-cancer	191	95	2	2
breast	466	233	2	2
chess	2130	1066	4	2
cleve	202	101	3	2
corral	32	128	3	2
diabetes	512	256	2	2
flare	710	356	3	2
german	666	334	3	2
glass	142	72	4	6
glass2	108	55	3	2
hayes-roth	132	28	3	3
heart	180	90	3	2
hepatitis	103	52	3	2
horse-colic	300	68	2	2
iris	100	50	3	3
labor-neg	40	17	2	2
lenses	16	8	3	3
lymphography	98	50	3	4
monk1	124	432	4	2
monk2	169	432	3	2
monk3	122	432	3	2
new-thyroid	143	72	3	3
segment	1540	770	7	7
shuttle-small	3866	1934	5	6
soybean-small	31	16	4	4
tic-tac-toe	638	958	7	2
tutorial	9	24	2	2
vehicle	564	282	4	4
vote-irvine	300	135	2	2
vote	290	145	2	2
vote1-irvine	300	135	2	2
vote1	290	145	2	2
xd6	461	512	4	2
zoo	67	34	4	7

Figure 3. *Experimental Results (cont.)*

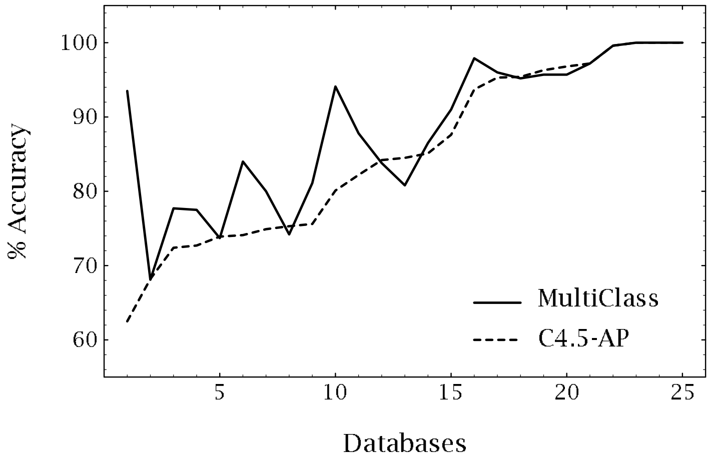


Figure 4. *Accuracy Comparison to C4.5-AP*

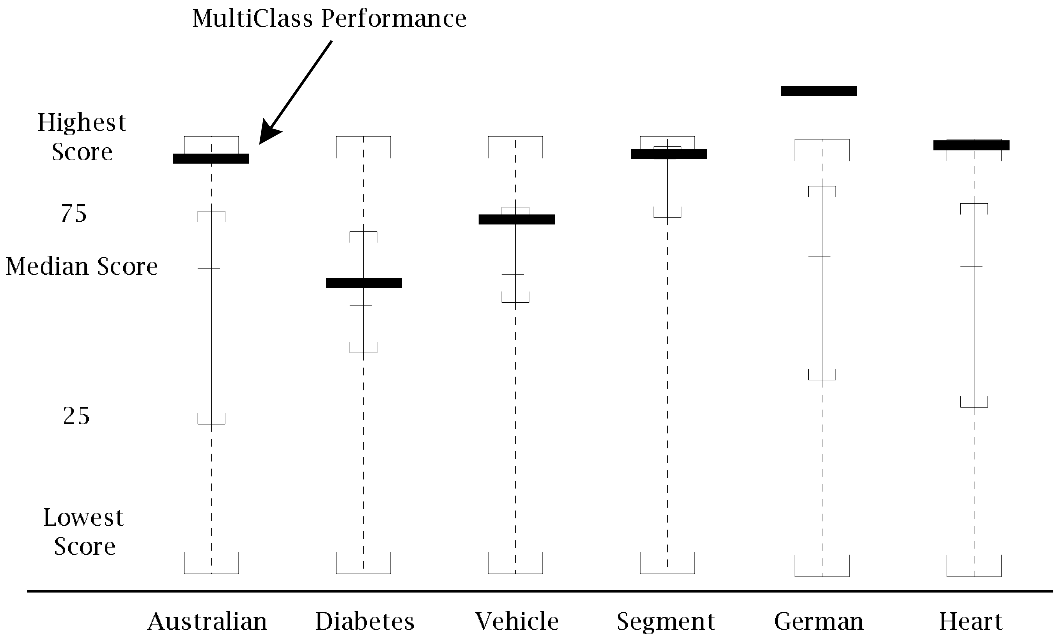


Figure 5. *Statlog Performance*

All of the problems, with the exception of balance-scale, were run without taking into account correlations among the continuous features within a class. The balance-scale model with correlations is almost twice as good as the model without. The tic-tac-toe and segment models were still improving when the experiments concluded at 7 classes.

Figure 4 presents a graphical comparison to the C4.5-AP results of Kohavi and John [34]. C4.5-AP improves upon C4.5 by finding the optimal combination of input parameters to the program. It therefore represents the best that one can expect C4.5 to do and is an ideal benchmark against which to compare. The graph in Figure 4 plots the accuracy scores of the two programs for the common data sets on which they were run. The scores have been sorted to make the graph easier to read. One thing to note is that these scores are generated using different methodologies. The MultiClass results come from a train/test experiment using the training and test set divisions in MLC++. The C4.5-AP numbers are averaged from a 10-fold cross validation combining all the data. The numbers should be roughly comparable despite this difference.

The graph shows a clear performance edge for MultiClass, though there are many data sets for which the two programs are indistinguishable. On the other hand, a simple decision tree program like C4.5 finds its models at least an order of magnitude faster than MultiClass. MultiClass runs in minutes on small data sets and hours on large data sets. Here is a sampling of how long it needs to find the models in Figure 3 on a relatively slow 16.3 SPECmark machine:

iris	1 minute
breast-cancer	7 minutes
vote	18 minutes
australian	1.5 hours
german	2.5 hours
chess	18 hours

These times could be improved by using a stochastic gradient approximation [39], by giving MultiClass successively larger portions of a database until its model stabilizes, or even by simply easing the optimality tolerance of NPSOL. The time cost seems relatively unimportant in most applications, where training time is not a significant issue, but it does make it more painful

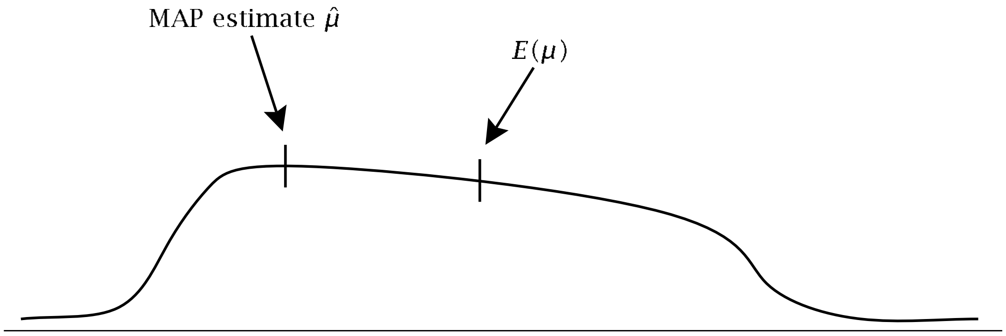


Figure 6. *Limitations of MAP Induction*

to run experiments. Both C4.5-AP and MultiClass make virtually instantaneous predictions from their final models.

Figure 5 graphs MultiClass's performance on the Statlog data sets against the results obtained in the Statlog experiment. Each entry shows the median, quartiles, and extremes of the Statlog entries. MultiClass does well, though the same methodology difference exists as with the C4.5-AP results.

V. Overfitting

Many of the compression ratios in Figure 3 indicate that MultiClass is overfitting. A model should generate equal ratios for the training and test sets, and the table shows many problems for which the training score is substantially higher. Overfitting occurs for two reasons. One is that MAP induction is inadequate, but the much more important reason is that the priors in MultiClass are too weak and do not accurately reflect the likelihood of irrelevant features.

Figure 6 illustrates why MAP induction breaks. It shows an imaginary distribution on the mean μ of a normal distribution. The MAP estimate $\hat{\mu}$ would lead to terrible predictions, since the vast majority of the probability is skewed to one side. A much better estimate of μ would be its expected value, and the best way to make predictions is to average over all possible values as described in Chapter 1. This example suggests that MultiClass might do better looking for the mean parameter values rather than the MAP values. Unfortunately, this solution is difficult to implement, ill-defined in the presence of linear constraints, and not terribly meaningful unless the posterior distribution is unimodal, but it does provide another way to think about summarizing the posterior distribution in a single model. The literature contains many ways to improve on MAP induction, like Laplace's approximation [1] or Gibb's sampling [15], that could potentially increase MultiClass's performance at the expense of computation time.

The most severe flaw in MultiClass is that its priors do not allow for the possibility that a feature is irrelevant. Here is a random database with 500 rows of coin flips to demonstrate the problem:

x_1	x_2	x_3	$\cdots$	x_9	x_{10}
H	T	H		H	H
T	H	T		H	H
T	T	T		T	T
T	T	H	$\cdots$	T	H
H	H	H		T	T
H	T	H		H	H
H	T	T		T	T
$\cdots$	$\cdots$	$\cdots$		$\cdots$	$\cdots$

There is no pattern, except any produced by blind luck, but MultiClass nevertheless finds the following two class model to predict the first flip from the others:

	Classes #1	#2
	0.49	0.51
x_1	0.67	0.36
x_2	0.53	0.46
x_3	0.51	0.49
x_4	0.49	0.57
x_5	0.45	0.56
x_6	0.55	0.41
x_7	0.55	0.43
x_8	0.48	0.46
x_9	0.58	0.41
x_{10}	0.52	0.55

Each feature in the model is only weakly informative, but many weak tests can equal one very strong one. If all the coin flips align with the first class, for instance, MultiClass will make an extreme prediction that heads should occur with probability 0.67. The trouble is that even though the columns in the database are independent, the correlation between any two is never exactly zero, so MultiClass finds that every column is slightly informative in predicting the first coin flip. The pairwise count in the data between the first column and the ninth column, which is the most informative, is,

$$\begin{array}{cc|cc|} & & \multicolumn{2}{c}{x_9} \\ & & H & T \\ \cline{3-4} x_1 & H & 134 & 123 \\ & T & 113 & 130 \\ \cline{3-4} \end{array}$$

MultiClass does not consider the possibility that these numbers arise from random chance, so it naturally concludes that the ninth coin flip contains a fair amount of useful information. The problem of extraneous correlations grows worse as the number of columns increases, since the probability of observing a lucky high correlation increases.

The horse-colic data provides a real example of how irrelevant features hurt MultiClass. The program's performance is much better when all but the four most relevant features are eliminated:

	Compression Ratio		
	Train	Test	Test Accuracy
All Features	3.16	1.21	57/68
Relevant Features	1.66	1.72	58/68

Now there is no evidence of overfitting, and the compression ratio on the test set is greatly improved. Notice that the testing accuracy cannot distinguish these two models.

A feature becomes irrelevant when it has the same distribution across multiple classes. MultiClass needs to be modified so that it assigns a prior to this possibility and explicitly searches for it. The strength of the prior should grow with the number of features to offset the increased likelihood of random high correlations. When predicting pregnancy in the thyroid data of Chapter 6, this change would let MultiClass conclude that the lab test and hypothyroid features are truly irrelevant. The model it would find is,

	Classes #1	#2
	0.62	0.38
male	0.85	0.01
pregnant	0.00	0.66
lab test	70.2 ± 21.7	70.2 ± 21.7
hypothyroid	0.56	0.56

This model improves on the one in Chapter 6, in which the lab test and hypothyroid features are slightly informative. A feature may be relevant part of the time, in which case it will vary in some classes and not in others. If there were nine classes, for example, hypothyroid might occur 20% of the time in seven of them and 67% in the other two.

The process of looking for completely irrelevant features is feature subset selection [29], while partial irrelevance is akin to weight sharing in neural networks. In either case, MultiClass must use graph search to decide which parts of its model to prune. A hill climbing algorithm that iteratively chooses a single feature and checks to see whether it should be the same in two classes seems logical, but this area is definitely one for future work.

VI. Programming with Probabilities

Implementing MultiClass was much harder than expected. There is currently a gaping chasm between the model level at which one naturally describes a data analysis problem, and the coding level at which one programs. Wray Buntine and I are developing a general purpose scientific modeling system to bridge this gap [7]. When finished, it will allow researchers to create programs like MultiClass in an afternoon.

The research effort, *Programming with Probabilities*, combines Buntine's graphical modeling language [6] with data flow and object oriented modeling lessons learned from the MultiClass implementation. The goal is to let one specify a mathematical model by drawing graphs, linking together components from a palette, and writing down equations. A symbolic package like *Mathematica* then analyzes the model, performs any required integral and derivative

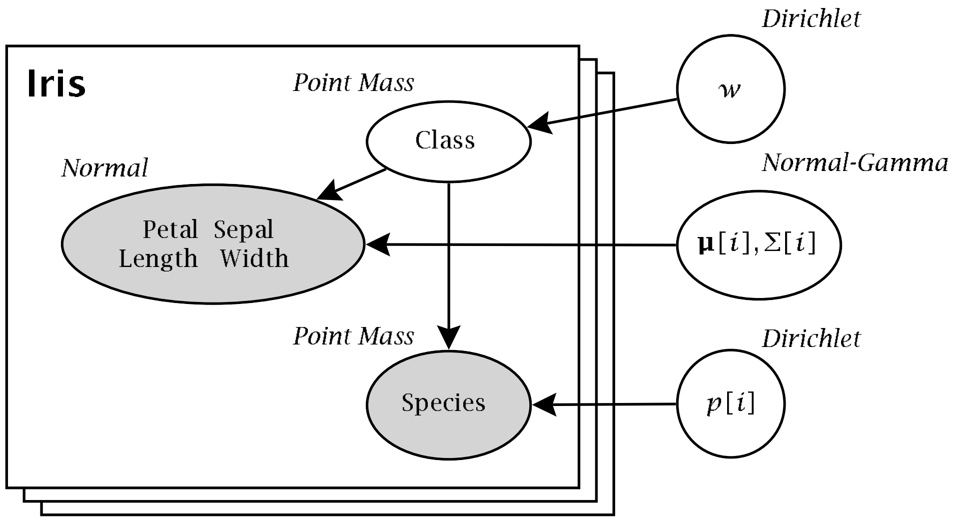


Figure 7. *Iris Model*

calculations, and compiles the result into efficient code using a data flow analysis to optimize function evaluations.

Figure 7 shows a mixture model for the iris data in this system. The picture generalizes the belief net structure from Chapter 4 by explicitly including the model parameters. Shaded variables are observable; unshaded ones are hidden. The stacked boxes, called plates, indicate structure that is replicated for each flower in the database. Each plate represents a flower, and the complete stack represents the entire table. Using this system, one would implement MultiClass by drawing the graph, asking the system to compile a module that calculates the conditional probability of petal length and its gradient with respect to the model parameters, and linking the final module to an optimizer like NPSOL that searches for the MAP parameter estimates. The system would symbolically evaluate the partition function integral and partial derivatives during the compilation.

Programming with Probabilities lets one move past the simplistic view that a database is a single, homogenous table. Figure 8 shows a model for \$/DM exchange rates, in which every bank is associated with a variable number of bid and ask price quotes. This model classifies banks according to their geographic location and how they behave in the market. Figure 9 shows a model for the student data, in which a program assessing student applications will

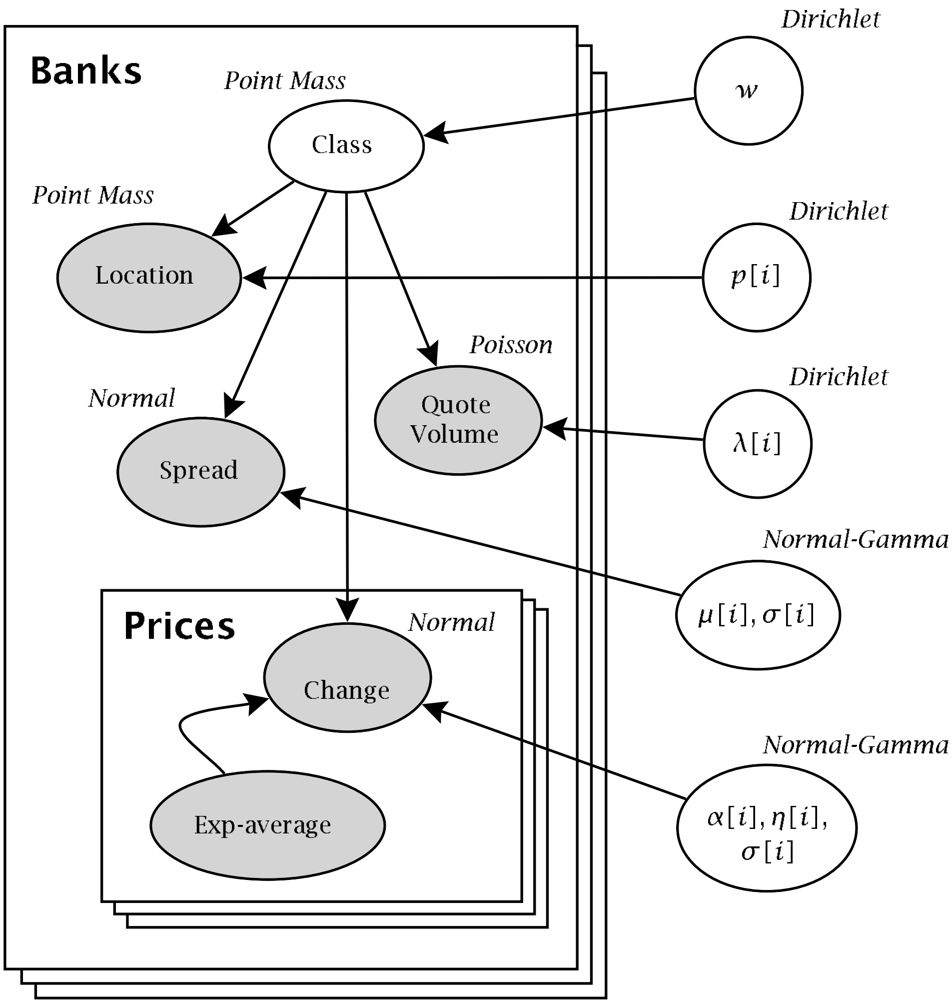


Figure 8. *Bank Model*

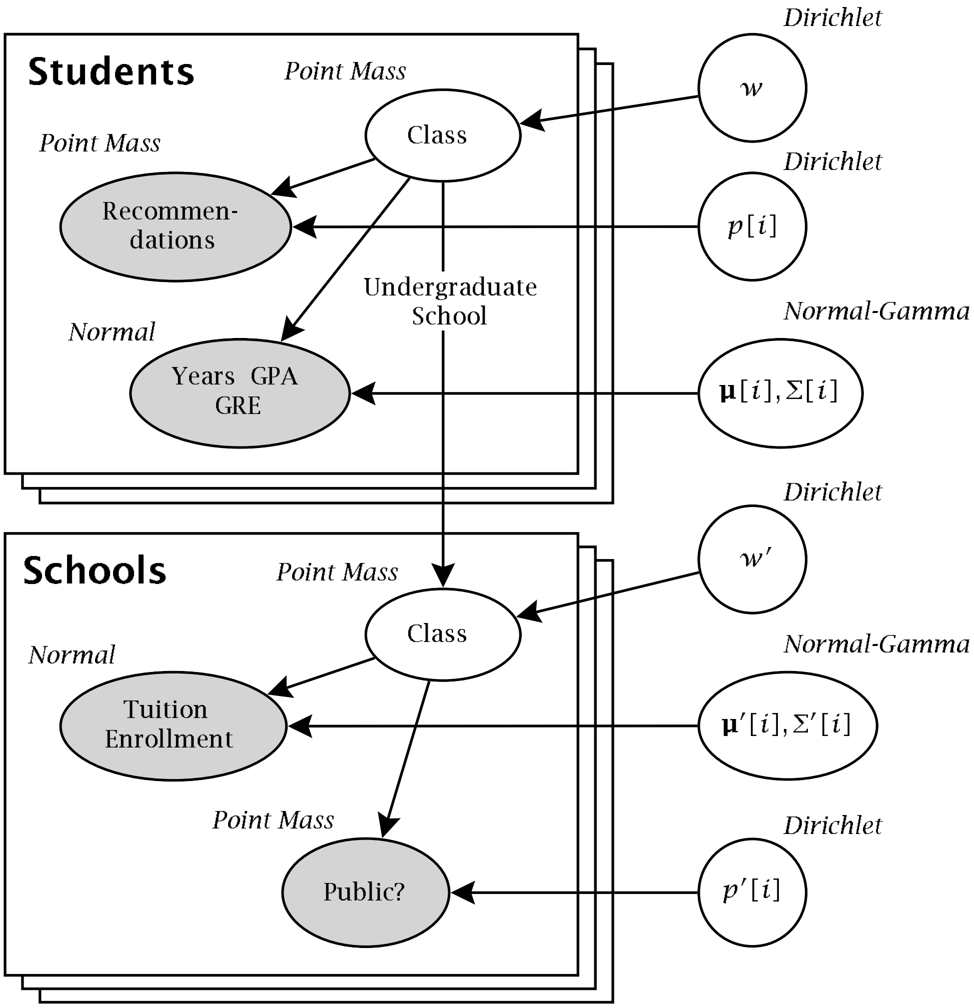


Figure 9. *School and Student Model*

simultaneously divide both students and schools into useful categories. It may discover, for instance, that private schools do a much better job than public schools of preparing students to earn Ph.D. degrees quickly. In these examples, the graphical language lets one express a probability distribution across multiple tables, each of which can contain both observed and hidden features. Using the general purpose algorithm described at the end of Chapter 6, one could optimize the parameters to make any desired joint or conditional prediction. The graphical language is exceptionally powerful, and the final system promises to be a great advance that overcomes the current engineering bottleneck in data analysis and opens the door to future research.

References

[1] A. Azevedo-Filho and R. Shachter. Laplace's method: Approximations for probabilistic inference in belief networks with continuous variables. In R. L. de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, pages 28–36, Seattle, WA, 1994.

[2] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory.* John Wiley & Sons, Chichester, 1994.

[3] D. M. Boulton and C. S. Wallace. A program for numerical classification. *The Computer Journal*, 13(1):63–69, 1970.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, 1984.

[5] W. L. Buntine. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, pages 182–201. Chapman & Hall, London, 1991.

[6] W. L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[7] W. L. Buntine and H. S. Roy. Software for data analysis with graphical models. To appear in the proceedings of the International Workshop on Artificial Intelligence and Statistics, 1995.

[8] P. Cheeseman. Personal communications.

[9] P. Cheeseman et al. AutoClass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, 1988.

[10] P. Cheeseman et al. Bayesian classification. In *Seventh National Conference on Artificial Intelligence*, pages 607–611, 1988.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[12] W. Dillon and M. Goldstein. *Multivariate Analysis: Methods and Applications.* John Wiley & Sons, 1984.

[13] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[14] Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. In *Proceedings of the 1991 Conference on Neural Information Processing Systems*. Morgan Kaufmann, 1992.

[15] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[16] Z. Ghahramani. Factorial learning and the EM algorithm. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS*93)*. Morgan Kaufmann, 1994. `ftp://psyche.mit.edu/pub/zoubin/factorial.ps.Z`.

[17] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for NPSOL (version 4.0): A Fortran package for nonlinear programming. Technical Report SOL 86-2, Stanford Systems Optimization Lab, Stanford, CA, 1986.

[18] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press ltd., London, 1981.

[19] R. Hanson, J. Stutz, and P. Cheeseman. Bayesian classification with correlation and inheritance. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1991.

[20] M. Henrion, J. S. Breese, and E. J. Horvitz. Decision analysis and expert systems. *AI Magazine*, 12(4):64–91, Winter 1991.

[21] G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 7, pages 282–317. MIT Press, 1986.

[22] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences, USA*, volume 79, pages 2554–2558, 1982.

[23] L. Hunter and D. States. Applying Bayesian classification to protein structure. In *IEEE Conference on Applications of AI*, 1991.

[24] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991.

[25] E. T. Jaynes. Prior probabilities. In *IEEE Transactions on Systems Science and Cybernetics*, pages 227–241. IEEE, 1968. Also appears in Jaynes [27], chapter 7.

[26] E. T. Jaynes. Marginalization and prior probabilities. In A. Zellner, editor, *Bayesian Analysis in Econometrics and Statistics.* North-Holland Publishing Company, Amsterdam, Holland, 1980. Also appears in Jaynes [27], chapter 12.

[27] E. T. Jaynes. *Papers on Probability, Statistics, and Statistical Physics.* D. Reidel Publishing Co., Dordrecht, Holland, 1989.

[28] E. T. Jaynes. *Probability Theory: The Logic of Science.* Fragmentary version of June, 1994. `http://omega.albany.edu:8008/JaynesBook.html`.

[29] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference.* Morgan Kaufmann, 1994. `ftp://starry.Stanford.EDU:pub/ronnyk/ml94.ps`.

[30] M. Jordan and R. Jacobs. Supervised learning and divide-and-conquer: A statistical approach. In P. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 159–166. Morgan Kaufmann, 1993.

[31] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[32] M. I. Jordan and L. Xu. Convergence properties of the EM approach to learning in mixtures-of-experts architectures. Technical Report 9301, MIT Computational Cognitive Science, Cambridge, MA, 1993.

[33] R. E. Kass and A. E. Raftery. Bayes factors and model uncertainty. Technical Report #571, Department of Statistics, Carnegie Mellon University, PA, 1993. To appear in *Journal of the American Statistical Association.*

[34] R. Kohavi and G. John. Automatic parameter selection by minimizing estimated error. `ftp://starry.Stanford.EDU:pub/ronnyk/c45ap.ps`, 1995.

[35] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743. IEEE Computer Society Press, 1994. `ftp://starry.Stanford.EDU:pub/ronnyk/mlc/toolsmlc.ps`.

[36] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Tenth National Conference on Artificial Intelligence*, 1992.

[37] D. B. Lenat and E. A. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47:185–250, 1991.

[38] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992. `ftp://131.111.48.8/pub/mackay/README.html`.

[39] O. L. Mangasarian and M. V. Solodov. Backpropagation convergence via deterministic nonmonotone perturbed minimization. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1994.

[40] J. D. Martin and D. O. Billman. Acquiring and combining overlapping concepts. *Machine Learning*, 16:1–37, 1994. `ftp://ai.iit.nrc.ca/pub/joel/nov.dvi.ps`.

[41] D. Mitche, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, 1994.

[42] T. M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pages 305–310, 1977.

[43] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases, 1995. `ftp://ics.uci.edu:pub/machine-learning-databases`.

[44] C. R. Musick Jr. *Belief Network Induction*. PhD thesis, University of California at Berkeley, 1994.

[45] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Submitted to *Biometrika*. `ftp://cs.toronto.edu/pub/radford/www/publications.html`, 1993.

[46] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning Journal*, 5:71–99, 1990.

[47] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[48] C. Peterson and J. R. Anderson. A mean field theory learing algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[49] D. Rumelhart. Personal communications.

[50] E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7:51–71, 1995.

[51] H. Schütze. Dimensions of meaning. In *Proceedings of Supercomputing '92*, 1992. `ftp://csli.stanford.edu/pub/prosit/papers`.

[52] S. B. Thrun et al. The monk's problems—a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, CMU School of Computer Science, 1991.

[53] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, Chichester, 1985.

[54] G. G. Towell, J. W. Shavlik, and M. O. Noordewier. Refinement of approximately correct domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, 1990.