ACTIVE LEARNING OF GAUSSIAN MIXTURE MODELS USING DIRECT ESTIMATION OF ERROR REDUCTION

By

Jeffry Gaston

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Approved by the
Examining Committee:

_____
Sanmay Das, Thesis Adviser

_____
Malik Magdon-Ismail, Member

_____
Mohammed Zaki, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2012
(For Graduation May 2012)

CONTENTS

LIST OF FIGURES

ABSTRACT

Traditional supervised learning for training a classifier involves a static set of fully-labeled data. However, labels can be costly to obtain. Active learning can provide lower misclassification rates for an equivalent number of labeled datapoints by allowing the learner to choose particularly important points for labeling. Standard paradigms for choosing points to label may be to request points near the decision boundary, or points where uncertainty is maximum. We show how to directly optimize the target criterion by estimating error reduction directly in a probabilistic framework (the Gaussian Mixture Model) and show that in several cases this can reduce the need for as many labeled training examples as would be required by the maximum-uncertainty approach.

1. Introduction

This thesis studies the problem of active learning. Traditional supervised learning for training a classifier involves a static set of fully-labeled data. However, labels can be costly to obtain. Active learning often provides lower misclassification rates for an equivalent number of labeled datapoints by allowing the learner to choose particularly important datapoints for labeling. Standard paradigms for choosing datapoints to label may be to request datapoints near the decision boundary, or datapoints where uncertainty is maximum. We do an explicit comparison between the maximum-uncertainty approach and another algorithm that optimizes the target criterion in a probabilistic framework based on the Gaussian Mixture Model. The primary flaw in the Max-Uncertainty algorithm we find to be that it does not consider the impact that newly acquired labels will have on the labels of the mixture components, which is most significant in early iterations when the labels of mixture components are highly unknown.

1.1 Semi-Supervised Learning

The machine learning community has studied the use of active learning to repeatedly query a dataset for the development of a classifier [1]. The active learner has two tasks: the classification of unlabeled datapoints, and the decision of which query to make next, both of which are dependent on the current set of both labeled and unlabeled examples.

At any or every iteration, an active learner can use a semi-supervised approach when required to actually compute the current classifier. Here we refer not to the clustering problem but to the classification problem, where there exists a set X of datapoints x, each having a d-dimensional feature vector, and for some datapoints their label (sometimes referred to as their "class") L_x is known, while for other datapoints the class is unknown. The goal in such a scenario is to properly classify as many datapoints as possible. The significance of semi-supervised learning is that often labels are costly to obtain, while unlabeled data are comparatively abundant;

therefore it is desirable to utilize semi-supervised learning to take advantage of labeled data to reduce human labor and improve accuracy if possible [2]. The request for the label of a datapoint x may represent a query, experiment, or action, depending on the particular research field and problem domain [3], but the exact interpretation does not affect the desire to minimize the number of such requests.

What we truly wish to estimate is the probability that a given datapoint corresponds to a given class, that is, $\Pr(y|x)$. However, using purely unlabeled data, the most information that we usually can hope to determine is $\Pr(x)$ [2], the relative frequency with which the feature vector x is expected to appear. Therefore, as a prerequisite for semi-supervised learning to provide any advantages over purely supervised learning, it is not only desirable but presumably essential that in our chosen model, $\Pr(x)$ shares some parameters with $\Pr(y|x)$ [2]. Fortunately, there exist practical classifiers, such as the Gaussian Mixture Model, for which $\Pr(y|x)$ and $\Pr(x)$ are not independent. Zhu and Goldberg [4], show that a classifier trained with both labeled and unlabeled data can attain a lower error rate than one trained using only the labeled data. Goldberg et al. [5] also analytically assert and empirically verify the existence of problems for which semi-supervised learning can lead to a lower error rate than learning on only the labeled examples.

1.2 Active Learning

The second job that an active learner is tasked with is to, at every iteration, generate a query about the dataset that is expected to "optimally" decrease the error rate of its classifier. Angluin [6] proposes and analyzes several types of queries, including membership queries for specific datapoints, in addition to queries of higher complexity, which, for example, may request a counterexample to the correctness of a given classifier. We impose essentially the same intuitive restriction as Cohn et al. (1994) [7], which is that the active learners we study will only have the opportunity to request the label associated with any particular datapoint x. McCallum et al. [8] agree that a human expert with appropriate domain knowledge may be able to aid the system in the classification of examples, and in some domains such as text classification, an overwhelming quantity of individuals posess the appropriate infor-

mation to aid the system.

The intuition behind the advantage of the active learner is that there must be some pattern in the data for learning to be possible, and if there is a pattern, then there is some degree of redundancy in the data. An active learner systematically selects the datapoint that provides the most information gain, whereas a passive learner merely receives whichever data is provided to it.

Cohn et al. [3] datapoint out that the importance in selecting an effective active learning scheme is quantified by not only the effort expended by those outside of the system, but also by the runtime as well. An active learner that selects datapoints more effectively will decrease the number of iterations for which it must run, thereby improving runtime, making doubly important the task of choosing datapoints appropriately.

Yet another advantage of some active learning systems is the ability to determine when the diminishing returns of learning warrant a termination of the learning process, to prevent excessive effort being expended in labeling additional datapoints. Premature termination requires that the classifier has a notion of its expected classification error rate. It also requires some relationship between classification error and labeling effort, which may be created by defining the overall cost at time T as a linear function of the number of misclassified datapoints and the number of labels requested.

Again, the goal of typical classifier design is essentially to produce the classifier with minimum error rate on the datapoints to which it will be applied [9], perhaps under some constraints on the amount of data or amount of execution time. In active learning, the classification problem is slightly less well-defined. The problem that is solved by the greedy approach is that the quality of each decision is based solely on the decrease in error rate observed in the subsequent iteration. In the context of a Support Vector Machine, Tong and Koller [10] implement and test a myopic (greedy) approach.

Yet another possible evaluation metric would be to evaluate the learner only on iteration k, where k is based on some estimate of how many trials it should require for the learner to attain a suitably low error rate.

The evaluation metric that seemingly mimics the real world would be applicable only for a learner that is able to automatically determine the datapoint at which it wishes to terminate learning. Given an appropriate cost function, it is possible to directly calculate the cost of using such a learner by substituting into the cost function the number of labels the learner requested before termination and the number of misclassified datapoints at termination. This would provide a value for the total cost associated with using the learner. However, because we observe that a slightly inappropriate model can lead to substantially inaccurate expected error rates, we do not yet perform a full investigation of learners that terminate their learning.

In this thesis, the primary evaluation metric that we use is a notion of how quickly the error rate decreases for a given active learner as it iteratively requests more labels. If for all (or more realistically, for most) k, the error rate of active learner A at iteration k is less than the error rate of active learner B, then learner A is superior to learner B for the given dataset. While this rule may often not pass judgment on the relative values of two different classifiers, it is extremely hard to dispute the correctness of this rule, and in many cases this rule is adequate for distinguishing between learners.

Myopic learners in the past have used several different algorithms to select datapoints for labeling. Cohn et al. [3] assumed that the bias of their learners was negligble, and that therefore their active learner would select the datapoint that minimized the expected variance, which would, in turn, minimize the expected error rate.

We propose directly calculating the expected error rate of the classifier by taking advantage of the probabilistic nature of the Gaussian Mixture Model. Once having a method to calculate the expected error rate, every iteration may consist of considering each datapoint independently and requesting the label of the one that is expected to lead to the lowest error rate.

1.3 Gaussian Mixture Models

As observed by Hsu et al. [11], a troublesome issue in many active learners is sampling bias. Fortunately, the fact that labeled examples are not representative of

the entire population is not a major concern in a Gaussian Mixture Model (GMM), because a GMM can use the unlabeled examples to properly estimate the population density functions.

The theory behind a GMM [12] is that each given datapoint in the dataset is generated as follows:

S is chosen at random from a fixed set of k Gaussian distributions, with respect to the relative weight of each distribution
x is chosen at random from S
return x

There are multiple variants on the above GMM [12]. Each covariance matrix may be constrained to be diagonal, which allows the variance of each dimension to be learned independently for a given Gaussian distribution but assumes that dimensions are independent. We do not see the need to enforce such a restriction here, and so we do not; in fact, preliminary tests showed higher error rates when enforcing diagonal covariance matrices, even though an arbitrary Gaussian can be approximated arbitrarily closely by a finite sum of Gaussians each having smaller weight and a diagonal covariance matrix.
The typical approach to learn the parameters of a GMM is to apply Expectation Maximization (EM). The center and covariance matrix of each distribution is initialized arbitrarily. Then, based on the current values of all parameters, for each datapoint and each Gaussian distribution, the datapoint is assigned a probability that it was generated by that distribution. Treating these probabilities as fractional counts, we update all parameters (centers and covariances) of the model. We alternate between these two steps until convergence. While it is possible that EM will converge to merely a local maximum or even merely a stationary datapoint [13], empirically the algorithm performs adequately for our purposes. If convergence to a local maximum becomes problematic, then other advanced techniques such as a split-and-merge algorithm [14] may prove useful.
Zhang and Scordilis [14] note the possibility of singularities in the Gaussian Mixture

Model, in which one covariance matrix may converge to zero if a small number of datapoints are located extremely close to the learned center of one Gaussian distribution. We claim that with tied covariance matrices, no covariance matrix will go to zero unless all go to zero, in which case the observed deviation is actually approximately zero and this is acceptable. Consequently, we enforce the constraint of tied (equal) covariance matrices for all Gaussians.

GMMs are used to model many types of systems, due to their ability to represent a large class of distributions [12]. This versatility of the Gaussian Mixture Model provides further justification for its use. Specifically, GMMs have been used to identify a speaker based on auditory data [12] or to classify Irises [15]. Here we will use them to classify Irises and estimate the age of abalone [16].

1.4 Contributions

- First, we show how to directly calculate the expected error of the classifier that uses the Gaussian Mixture Model. We use this error calculation to create a myopic algorithm that requests for labeling the datapoint that will minimize the expected error for the following iteration. We analyze this error and compare it to other current paradigms that active learners use to select the next datapoint for labeling, and we illustrate examples in which our myopic algorithm exhibits lower error rates than does the Maximum-Uncertainty algorithm.

- Second, we describe an algorithm known here as the Fast algorithm, intended to approximate the behavior of the myopic algorithm.

- Third, we provide a variation of the Expectation Maximization algorithm, which approximates its results to any arbitrary accuracy but executes more quickly when there are large numbers of Gaussian distributions in the mixture model.

1.5 Problem Summary

Let us now concisely state the classification task being considered:

Suppose that there are k distributions $S_1 ... S_k$, where each datapoint x_j is a vector of length d, and each x_j belongs to some S_i. For each x_j let there be a corresponding label L_{x_j}, and also let there be a label L_{S_i} for each S_i. Let the label of each datapoint match the label of its corresponding distribution; that is, let $L_{x_j} = L_{S_i}$ when $x_j \in S_i$. Each L_{S_i} remains unknown, as is the information as to which x_j was generated by which S_i. Let $S_i(x_j)$ denote the probability density function of S_i evaluated at x_j. Finally, let m denote the number of distinct labels that may be found in the dataset. Note that for binary classification problems, $m = 2$.

We wish to estimate $L_{x_1} ... L_{x_n}$.

Therefore, we make the reasonable assumption that there exists an expert we may ask to determine any L_{x_j}, but that the expert's time is costly. So, it is desirable to minimize the misclassification rate using only a limited number of labeled datapoints. In this thesis, we focus primarily on the creation of a greedy approach, that is to say, given a set X of datapoints x_i, and a (possibly empty) subset of $\{L_{x_i}\}$, we wish to identify x such that knowing L_x minimizes the expected number of mislabeled datapoints x_i: $\sum_{i=1}^{n} (1 - \max_j \Pr(L_{x_i} = j))$.

2. Active Learning of GMMs in One - & Two - Dimensions

The underlying intuition that we present here resembles that given by Roy & McCallum [1]; however, we apply it in the context of a Gaussian Mixture Model rather than a Naive Bayes Classifier. We wish to request the label of the datapoint that is expected to minimize the expected error rate that we explicitly calculate.

In order to gain intuition, we consider a simple active learning setting of the following form.

1. The data is 1-dimensional.
2. There are exactly two distributions, each having a distinct label.

For the simplified problem, the algorithm we apply to choose x is as follows:

Initialize μ_1 to the average of $\{x_i \text{ s.t. } L_{x_i} = 1\}$, or $\max(x)$ by default if there are no such x_i

Intialize μ_2 to the average of $\{x_i \text{ s.t. } L_{x_i} = 2\}$ or $\min(x)$ by default if there are no such x_i

{Here we explain the Expectation Maximization algorithm, which uses these initial guesses to calculate μ_1 and μ_2.}

while μ_1 or μ_2 have not converged **do**

  for all unlabeled x_j **do**

    Compute $\Pr(x_j \in S_i)$ based on the locations of x_j and all S_i.

  end for

  for all labeled x_j **do**

    {set $\Pr(x_j \in S_i)$ to 1 or 0 for all known labels}

    $\Pr(x_j \in S_i) \leftarrow \Pr(L_{x_j} = L_{S_i})$ {L_{x_j} is known and we assume for the moment $\Pr(L_{S_i} = i) = 1$.}

  end for

  Using $\Pr(x_j \in S_i)$, update μ_1 and μ_2 as is usual in an Expectation Maximization (EM) algorithm. {The remaining details of EM will be clarified later in case

the reader is unfamiliar}

end while

{Here we explain the myopic active-learning approach}

for all x_j **do**

Calculate the expected value of what the error will be after determining L_x, which is calculated as an appropriately weighted average of the expected error after assigning either possible value to L_x. {Note that in the interest of runtime, during this step we do not recompute the means using Expectation Maximization; however, for each datapoint we do re-estimate the probability with which it has any particular label, as will be explained below}

end for

return the x_j that for which expect minimum error after receiving its label.

The expected error for any such possible labeling can be computed as follows:
Let A denote the event that the labels given to the known datapoints match their current known labels. Let B denote the event that the true labels of the distributions match the assumed labels.
From Bayes' Theorem we have $\Pr(B|A) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A)}$. We assume that either distribution has equal prior probability of having any particular label, that is, $\Pr(B) = \frac{1}{2} = \Pr(\neg B)$. So $1 = \Pr(B) + \Pr(\neg B) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A)} + \frac{\Pr(A|\neg B)\Pr(\neg B)}{\Pr(A)} = \frac{\Pr(A|B)+\Pr(A|\neg B)}{2\Pr(A)}$ and then $\Pr(A) = \frac{\Pr(A|B)+\Pr(A|\neg B)}{2}$. Therefore, $\Pr(B|A) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A)} = \frac{2\Pr(A|B)\Pr(B)}{\Pr(A|B)+\Pr(A|\neg B)}$.
Now, using $\Pr(B|A) = \Pr((L_{S_1} = 1)|A)$, $\Pr(A) = 1$, and $\Pr(x_j \in S_i)$, we can compute $\Pr(L_{x_j} = 1) = \Pr(x_j \in S_1)\Pr(L_{S_1} = 1) + \Pr(x_j \in S_2)\Pr(L_{S_2} = 1)$. From this we can compute the probability that any particular datapoint will get any particular label. Finally, the expected total error is $\sum_{j=1}^{n}(1 - \max_i \Pr(L_{x_j} = i))$, for the scenario where x was assigned label L. These expected errors are multiplied by their corresponding probabilities and summed over all L. The x with minimum expected error is the datapoint whose label is to be requested next.

2.1 Predicted Behavior

Here we analyze the expected behavior for the first iteration in the 1 dimensional case.

Suppose $\mu_1 < \mu_2$ are the centers of Gaussians S_1 and S_2, each 1-dimensional and having standard deviation σ. Suppose that exactly one datapoint, x, has a known label of 1. Our assumption on the prior probabilities is that it is equally likely that $L_{S_1} = 1$ and $L_{S_2} = 2$ as $L_{S_2} = 1$ and $L_{S_1} = 2$. Then, we update our probabilities as follows: $\Pr(L_{S_1} = 1|x) = \frac{\Pr(x|L_{S_1}=1)\Pr(L_{S_1}=1)}{\Pr(x)} = \frac{S_1(x)\frac{1}{2}}{\frac{1}{2}S_1(x)+\frac{1}{2}S_2(x)} = \frac{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}+e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}$. Because the reciprocal function is monotonically decreasing for positive inputs, and this value is strictly positive, we will show that it is maximized when its reciprocal is minimized: $\frac{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}+e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}} = 1 + \frac{e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}$ which is minimized for the same x as $\frac{e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}$.

Because log is an increasing function, the above expression is minimized when its log is minimized: $\log(\frac{e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}) = -\frac{(x-\mu_2)^2}{2\sigma^2}+\frac{(x-\mu_1)^2}{2\sigma^2} = \frac{1}{2\sigma^2}(2(\mu_2-\mu_1)x+(\mu_1^2-\mu_2^2))$ Clearly, this value decreases as x approaches $-\infty$.

As x approaches $-\infty$, $\Pr(L_{S_1} = 1|x) = \frac{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}+e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}} = \frac{1}{1+e^{\frac{-(x-\mu_2)^2+(x-\mu_1)^2}{2\sigma^2}}}$ approaches 1.

Stated simply, as x approaches $-\infty$, we become increasingly certain that $x \in S_1$. Therefore, we expect the first datapoint that the algorithm selects for labeling to be the point at one extreme of the domain, because that will maximize the certainty with which it knows which distribution has which label.

2.2 Results for Preliminary Algorithm

We expected that the optimal first datapoint chosen for this simplified problem would be either $\max(x_j)$ or $\min(x_j)$. We created a synthetic dataset by drawing 10 points at random from each of two Gaussian distributions, one with mean 0 and the other with mean 1, and each with standard deviation of 1. Observe that the results support the intuition. The first datapoint chosen by the algorithm (the green

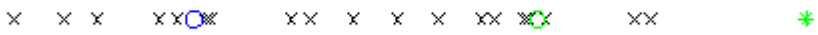


Figure 2.1: The first datapoint chosen in the one-dimensional case is at one extreme.

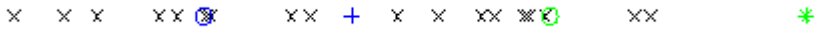


Figure 2.2: The second datapoint chosen in the simple, one-dimensional case is in the middle.

asterisk) is at one extreme, which minimizes the expected total error by maximizing the dominant contribution to the error: the uncertainty in knowing which distribution gets which label. Subsequent datapoints chosen are near the middle, when the dominant contribution to the error is the uncertainty in which distribution generated the datapoints in the middle. Note that with an arbitrarily large quantity of unlabeled data from each distribution, the acquisition of one label near the boundary will make an arbitrarily small change to the classifier. The significance of the acquisition of a label near the boundary is that it removes the uncertain datapoint from the error estimate.

Also note that this algorithm assumes that each Gaussian distribution receives a different label. Therefore, after having received a single label, the algorithm is sufficiently confident in the labels of each distribution that the second label requested is near the middle, where uncertainty is highest.

2.3 Heat Map

For the remainder of this thesis, let us now remove the assumptions that:

1. The data is one-dimensional, and
2. There are exactly two different distributions, each with different labels.

We do still assume, however, that the probability density function for each distribution, S_i, is a Gaussian function with mean μ_i and covariance matrix Σ_i. Note that it is permissible for several distributions to receive the same label.

We will explain in a subsequent chapter our exact algorithm for calculating the expected number of errors on the dataset. For now, to better visualize the intended results, we show a heat map depicting the expected information gain from labeling

a datapoint as a function of the location of the datapoint. Each blue "*" represents a datapoint known to have been drawn from the blue distribution, and each blue circle is suspected to be the mean of a blue distribution. Each green "+" or circle is the analogue of a blue "*" or circle, respectively. Each black "x" is a datapoint with unknown label.

The background color in the heat map is based on the expected information gain from acquiring the label of a datapoint at any particular location. This expected decrease in error counts is calculated by this algorithm:

for all locations on the image **do**
  add an unlabeled x to the set X of datapoints
  $b \leftarrow -$(the expected number of errors on the dataset, including x)
  for all label L **do**
    $L_x \leftarrow$ unknown
    $p \leftarrow \Pr(L_x = L)$
    $L_x \leftarrow L$
    $c \leftarrow$ (the expected number of errors on the dataset X, including the labeled x)
    $b \leftarrow b + cp$
  end for
  remove x from the set X of datapoints
  use b to determine the color of the rectangle at the current location
end for

The reddest rectangle is expected to provide the least information and the grayest rectangle the most. The colors are scaled based on percentile, such that a rectangle that is halfway between red and grey has the median expected information gain.

The first heat map depicts the desirability of each location before any datapoints are labeled.

The rightmost distributions in the first heat map appear to have the most ownership over the datapoints, so knowing their labels is of the most value.

After choosing one datapoint and receiving its label, the second heat map adjusts to show that additional nearby labels are less desirable than labels elsewhere.

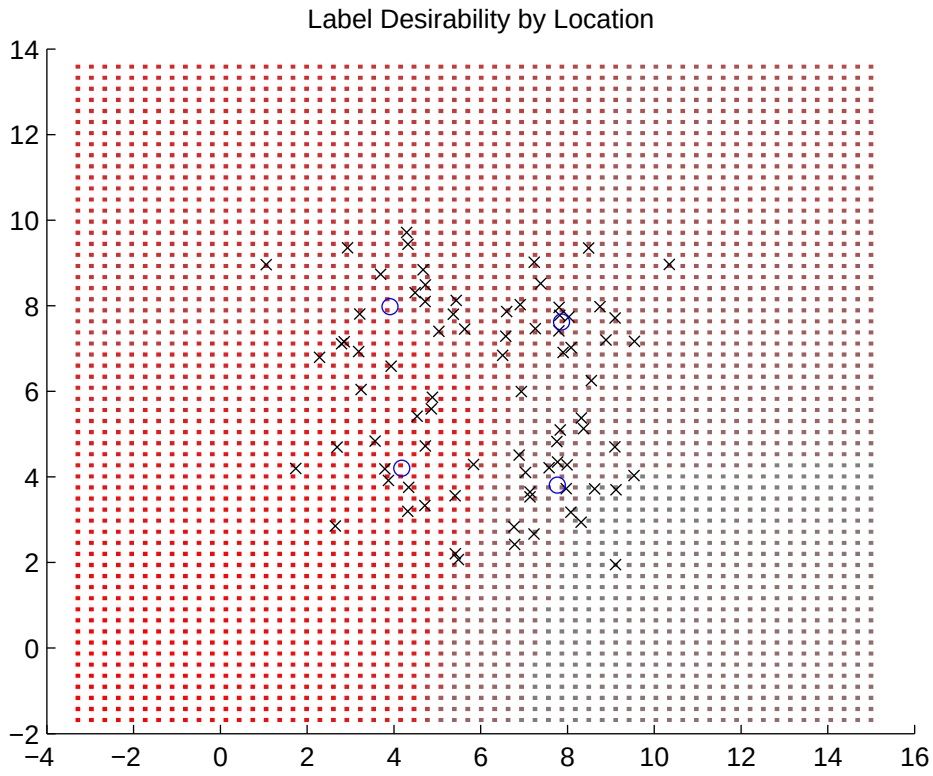


Figure 2.3: The desirability of each location before the acquisition of any labels.

Note that the first datapoint to be labeled by the Max-Uncertainty algorithm would be chosen arbitrarily. When no labels are known, we assume that each distribution has a $\frac{1}{2}$ probability of having either label, which implies that each datapoint also has a $\frac{1}{2}$ probability of having either label.

Also note that on later iterations, a datapoint halfway between the centers of two distributions of different labels will also have approximately a $\frac{1}{2}$ probability of having either label (the maximum amount of uncertainty possible in the label). However, such a datapoint would lower the overall error rate by a smaller amount than a datapoint that helps to find the label of a distribution whose label has high uncertainty. If the label of each datapoint is sufficiently expensive to obtain (i.e. each datapoint represents an airplane wing that must be built and tested), then even small decreases in the number of labels required become significant.

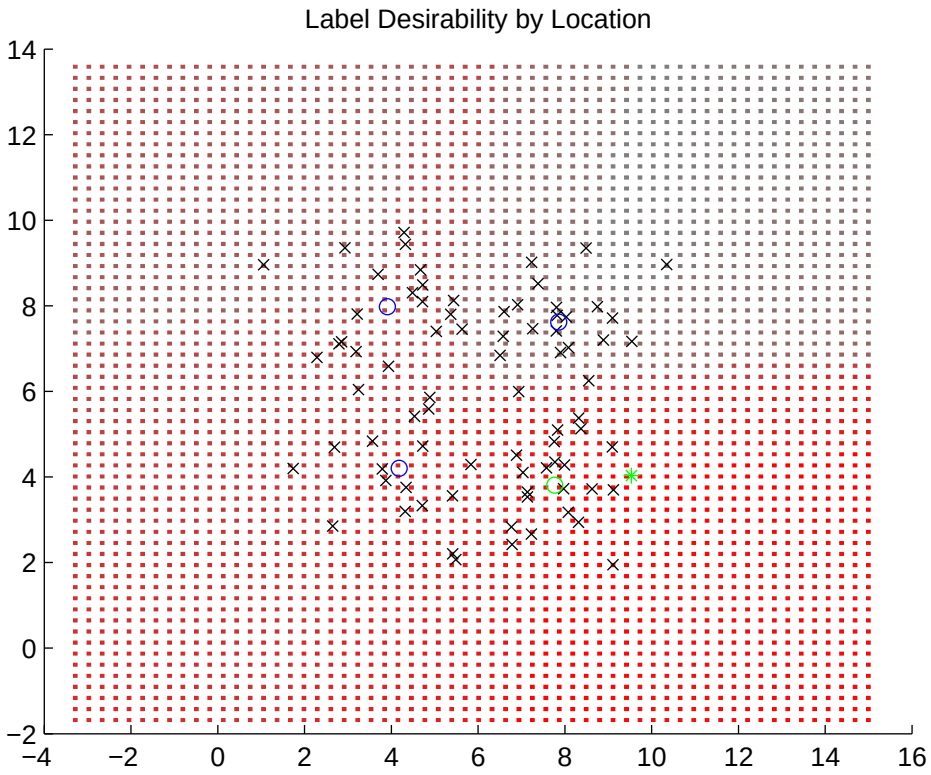


Figure 2.4: The desirability of each location after the acquisition of one label.

3. The Generalized GMM Active Learning Algorithm

Central to many of the algorithms is a method to calculate μ_i and σ_i given a set X of datapoints x_j, and an arbitrary subset of L_{x_i}. We propose an Expectation Maximization algorithm that is implemented as follows:

Step 1) Initialization.

{this is a recursive function that puts the datapoints X into a k-d tree to form the initial guesses for μ}

$i \leftarrow 0$

{Identify the dimension with the largest spread}

$d \leftarrow \text{argmax}_i \, (\max_j x_{j_i} - \min_j x_{j_i})$

Sort $\{x\}$ by increasing x_{j_d}

{Identify the two consecutive datapoints with largest difference between them}

$i \leftarrow \text{arg}max_j (x_{(j+1)_d} - x_{j_d})$

{Determine the number of distributions expected to be to have lower d-coordinate than these two datapoints, and snap this value to an integer}

$numLowerDistributions = \lfloor k\frac{i}{n} + \frac{1}{2} \rfloor$

if $numLowerDistributions = 0$ or $numLowerDistributions = k$ **then**

$numLowerDistributions \leftarrow \lfloor \frac{k}{2} \rfloor$

end if

{Split dimension d}

$numLeftPoints = \lfloor n * k / numLowerDistributions \rfloor$

$left \leftarrow x_1 \ldots x_{numLeftPoints}$

$right \leftarrow x_{(numLeftPoints+1)} \ldots x_n$

create a k-d tree, A, from $left$

create a k-d tree, B, from $right$

return a k-d tree having A and B as its immediate subtrees

We then invoke this function to create a k-d tree using the input datapoints X, and ensure that the number of leaf branches in the k-d tree equals k, the number

of Gaussian distributions being modeled. Once the k-d tree is formed, for each leaf box B_i, calculate the bounding box C_i of all datapoints that fall into B_1. Initialize each μ_i to the center of the corresponding C_i.
Also initialize σ to the identity matrix.

Step 2) Expectation.
Step 2a) For each datapoint x_j and each distribution S_i, compute $\Pr(x_j = S_i)$, based solely on the locations of x_j and S_i. Let $S = S_1 \cup S_2 \cup \ldots S_k$. Using Bayes' Theorem, we calculate: $\Pr(x_j \in S_i | x_j \in S) = \frac{\Pr(x_j \in S | x_j \in S_i) \Pr(x_j \in S_i)}{\Pr(x_j \in S)} = \frac{\Pr(x_j \in S_i)}{\Pr(x_j \in S)} = \frac{S_j(x_i)}{\sum_{m=1}^{k} S_m(x_j)}$.

Step 2b) For each distribution S_i and label L_j, calculate $\Pr(L_S = L)$
For each distribution S_i and datapoint x_j, we computed $\Pr(x_j \in S_i)$ in step 2a.
Now we iterate over each combination C of labelings of each S_i.
For each C_i, we calculate $\Pr(C_i | \{L_{x_j}\}) = \Pr(\{L_{x_j}\} | C_i) \frac{\Pr(C_i)}{\Pr(\{L_{x_j}\})} = \Pr(\{L_{x_j}\} | C_i) c$ for a constant c. Using $\sum \Pr(C_i | L_{x_j}) = 1$, we get $\Pr(C_i | \{L_{x_j}\}) = \dfrac{\Pr(\{L_{x_j}\} | C_i)}{\sum_i \Pr(\{L_{x_j}\} | C_i)}$.
Then for each S_i and L_j we calculate $\Pr(L_{S_i} = L_j)$ based on the probability that the correct scenario includes $L_{S_i} = L_j$.

Step 2c) For each S_i and each known L_{x_j}, compute $\Pr(x_j \in S_i)$, using information about their labels.
That is, $\Pr(x_j \in S_i | \{L_{x_j}\}) = \Pr(\{L_{x_j}\} | x_j \in S_i) \frac{\Pr(x_j \in S_i)}{\Pr(\{L_{x_j}\})} = \frac{\Pr(L_{x_j} = L_{S_i}) S_i(x_j)}{\sum \Pr(L_{x_j} = L_{S_i}) S_i(x_j)}$.

Step 3: Maximization of the centers
After having computed the probability of any particular distribution generating any particular data datapoint, it is straightforward to update the expected centers, μ_i. For each μ_i, set μ_i to the weighted average of all x_j, using weights equal to $\Pr(x_j \in S_i)$ as calculated in step 2.

Step 4: Expectation again.
Repeat step 2 to update the expected probabilities.

Step 5: Maximization of the covariance matrix.
Using the calculated values of $\Pr(x_j \in S_i)$, compute the expected deviation of x_j from the center of its corresponding distribution, and update σ accordingly. Note that there is only one σ that must simultaneously estimate the covariance matrices of each of the k Gaussian distributions.

Step 4: Termination. If the change in each μ_j is below a sufficiently small threshold, then stop. Otherwise, jump to Step 2.

This marks the end of the Expectation Maximization algorithm.

3.1 The Importance of Combinatorial Priors

It is important to note that the labels assigned to each distribution are not independent. If, for example, there are two Gaussian distributions and only one datapoint whose label is known, then the label of at least one distribution much match the label of the labeled datapoint. That is, if either distribution's label does not match the known label, then the label of the other distribution must. The lack of independence in the labels of each distribution creates additional complexity that must be handled. Essentially, our approach is to consider each combination C of labelings of all distributions, and compute a probability for each C.

If there are k Gaussian Distributions and m possible label types, we assign equal prior probability to each of the m^k possible labelings of the k distributions. Thus, with a binary classification problem, there are 2^k such possible labelings considered. In some cases it may be acceptable to only consider the labeling that is most likely, however, there are cases in which this approach does not work:

Suppose that no datapoints have known labels. In this case, each labeling has an equal posterior probability, and it is not possible to select the most likely labeling of the distributions. If we instead choose one labeling arbitrarily, then clearly there is a loss of information.

Another pathological example occurs when exactly one datapoint has been labeled. For each distribution S_i, there is a nonzero probability that the labeled datapoint was drawn from S_i, which implies that for every S_i, the most likely label of S_i is the same as the label of the one known datapoint. Clearly, if we simply assume that the most likely labeling is the correct one, then we always come to the unreasonable conclusion after labeling one datapoint that the labels of all distributions must certainly match the given label. In general, this confusion can lead to problems when few examples are labeled, which is the key domain of interest for us.

We conclude that there are cases in which we must consider several, if not all, possible labelings of the given distributions.

3.2 The Implementation of Combinatorial Priors

The expected number operations required to run this Expectation Maximization algorithm is exponential in the number of Gaussian distributions to be used in the mixture model. For some applications this is prohibitively slow, so we propose an improvement that considers a set of labelings whose total posterior probability is at least $1 - \epsilon$, and decreases the expected runtime substantially.

Let each $\{C_i\}$ be the current hypothetical label for distribution S_i. The simple, slower, algorithm to calculate $\Pr(S_i = L)$ is:

{Let $P(x_j, S_i)$ be the previously computed probability that distribution S_i generated x_j, using information only from locations and not from labels}
for all $C_1 \in \{1 \dots m\}$ **do**
  for all $C_2 \in \{1 \dots m\}$ **do**
    for all $C_3 \in \{1 \dots m\}$ **do**
      {And so on, where the number of loops equals the number of Gaussian distributions, k}
      {In actuality, this is not hardcoded as a fixed number of loops, but it is easier to interpret as such}
      $p \leftarrow 1$.

{Compute the probability that we would observe these labels on these datapoints, given that the distributions are labeled as we assume}
for all $x \in X$ **do**
$q \leftarrow 0$
for $i = 1$ to k **do**
if $L_x = C_i$ **then**
$q \leftarrow q + P(x, C_i)$
end if
end for
$p \leftarrow pq$
end for
{For each distribution, update the probability that it would receive the current label}
for $i = 1$ to k **do**
$\Pr(S_i = C_i) \leftarrow \Pr(S_i = C_i) + p$
end for
end for
end for
{Normalize the probabilities because we left out the constant factors in Bayes' Theorem}
for $i = 1$ to k **do**
$p \leftarrow \sum_j \Pr(S_i = C_j)$
for $j = 1$ to m **do**
$\Pr(S_i = C_j) \leftarrow \Pr(S_i = C_j)/p$
end for
end for

Unfortunately, this algorithm is extremely slow if the number of distributions, k, is large, even if only 2 classes exist, because the vector C takes a total of 2^k different values overall. Note that this step must be run for every iteration of Expectation Maximization, which means that any decrease in the execution time for this step

may create a great improvement to the overall performance. Even more importantly, the myopic algorithm must also run this step nm times at every iteration, where n is the number of datapoints that remain unlabeled and m is the number of distinct label types.

The intuition for the improvement that we suggest is that each distribution S_i will be located near some datapoints x_j, and for many such distributions, the nearby datapoints will likely all have the same label. If we are considering the a labeling C that includes a distribution whose label does not match the labels of its nearby datapoints, then the scenario is very unlikely and perhaps not worth considering.

For every combination of labelings C, which assigns a label to each distribution, there is some probability $\Pr(C)$ that C is the correct labeling. We choose small $\epsilon = 0.0001$ and ensure that the sum of $\Pr(C)$ over all C that we incorporate into our average will be at least $1 - \epsilon$. Here is our the fast, approximate algorithm to compute $\Pr(C)$:

{Let $P(x_j, S_i)$ be the previously computed probability that distribution S_i generated datapoint x_j, using information only from locations and not from labels}
{First, we attempt to guess the correct label for each distribution}
for all distribution S **do**
  for all label l **do**
    $W_{S,l} \leftarrow \sum_x P(x, S) \Pr(L_x = l)$
  end for
  $L_S \leftarrow \text{argmax}_l(W_{S,l})$
end for
{Next, we calculate the probability that the labeled datapoints would have received the labels that they received, given that the labels of each distribution are the labels that we currently assume they have}
$p \leftarrow 1$
for all $x \in X$ **do**

$q \leftarrow 0$
for $i = 1$ to k **do**
if $L_x = C_i$ **then**
$q \leftarrow q + P(x, C_i)$
end if
end for
$p \leftarrow pq$
end for
$p_{max} \leftarrow p$
{Now we begin to iterate over possible labelings and compute a probability for each}
$\epsilon \leftarrow 0.0001$
$p_{sum} \leftarrow 0$
for all $C_1 \in \{1 \dots m\}$ **do**
$r_1 \leftarrow 1$
for $i = 1$ to n **do**
if $L_i \neq C_1$ **then**
$r_1 \leftarrow r_1(1 - P(i, C_1))$
end if
end for
{Verify that the probability of this labeling is large enough to be worth considering}
if $rm^1 \geq \max(p_{max}, p_{sum})\epsilon$ **then**
for all $C_2 \in \{1 \dots m\}$ **do**
$r_2 \leftarrow r_1$
for $i = 1$ to n **do**
if $L_i \neq C_2$ **then**
$r_2 \leftarrow r_2(1 - P(i, C_2))$
end if
end for

{Verify that the probability of this labeling is large enough to be worth considering}

if $rm^2 \geq \max(p_{max}, p_{sum})\epsilon$ **then**

  for all $C_3 \in \{1 \dots m\}$ **do**

    $r_3 \leftarrow r_2$

    for $i = 1$ to n **do**

      if $L_i \neq C_3$ **then**

        $r_3 \leftarrow r_3(1 - P(i, C_3))$

      end if

    end for

    {And so on, where the number of loops equals the number of Gaussian distributions, k}

    {In actuality, this is not hardcoded as a fixed number of loops, but it is easier to interpret as such}

    $p \leftarrow 1.$

    {Compute the probability that we would observe these labels on these datapoints, given that the distributions are labeled as we assume}

    for all $x \in X$ **do**

      $q \leftarrow 0$

      for $i = 1$ to k **do**

        if $L_x = C_i$ **then**

          $q \leftarrow q + P(x, C_i)$

        end if

      end for

      $p \leftarrow pq$

    end for

    $p_{max} = \max(p, p_{max})$

    $p_{sum} = p_{sum} + p$

    {For each distribution, update the probability that it would receive the current label}

    for $i = 1$ to k **do**

$\Pr(S_i = C_i) \leftarrow \Pr(S_i = C_i) + p$

end for

end for

end if

end for

end if

end for

{Normalize the probabilities because we left out the constant factors in Bayes' Theorem}

for $i = 1$ to k **do**

$p \leftarrow \sum_j \Pr(S_i = C_j)$

for $j = 1$ to m **do**

$\Pr(S_i = C_j) \leftarrow \Pr(S_i = C_j)/x$

end for

end for

We assert r_i is an upper bound on the probability that the true labelings of $\{S_1 \ldots S_i\}$ match the candidate labeling $\{C_1 \ldots C_i\}$.

Note that:

$$\begin{aligned} \Pr(x \notin \{S_1 \cup S_2 \ldots S_i\}) &= 1 - \Pr((x \in \{S_1 \cup S_2 \ldots S_{i-1}\})) - \Pr(x \in S_i) \\ &\leq 1 - \Pr(x \in \{S_1 \cup S_2 \ldots S_{i-1}\}) - \Pr(x \in S_i) \\ &\quad + \Pr(x \in \{S_1 \cup S_2 \ldots S_{i-1}\}) \Pr(x \in S_i) \\ &= (1 - \Pr(x \in \{S_1 \ldots S_{i-1}\}))(1 - \Pr(x \in S_i)) \quad (3.1) \end{aligned}$$

By induction, we get

$$\Pr(x \notin \{S_1 \cup S_2 \cup \ldots S_i\}) \leq \prod_{j=1}^{i} (1 - \Pr(x \in S_j)) \quad (3.2)$$

Because $L_x \neq L_{S_i}$ implies $x \notin S_i$, we conclude

$$\Pr(L_x \notin \{L_{S_1} \dots L_{S_i}\}) \leq \prod_{j=1}^{i} (1 - \Pr(x \in S_j)) \tag{3.3}$$

Then, the probability that we would observe the given label L_x, assuming that the labels $L_{S_1} \dots L_{S_i}$ match $C_1 \dots C_i$ is no more than

$$\prod_{S_i \in \{S_1 \dots S_i\} | L_x \neq C_i} (1 - \Pr(x \in S_i))) \tag{3.4}$$

We conclude that the overall probability that we would observe the given labels L_{x_j}, assuming that the labels $L_{S_1} \dots L_{S_i}$ match $C_1 \dots C_i$ is no more than

$$\prod_{x} \prod_{S_i \in \{S_1 \dots S_i\} | L_x \neq C_i} (1 - \Pr(x \in S_i))) = \prod_{S_i \in \{S_1 \dots S_i\}} \prod_{x | L_x \neq C_i} (1 - \Pr(x \in S_i))) \tag{3.5}$$

To ensure that the total probabilities of scenarios rejected does not exceed ϵ, we must ensure that the number of scenarios times the probability of each does not exceed the remaining probability ϵ. Because both p_{max} and p_{sum} were generated by sums of weights of actual candidate labelings, we know that each one is a lower bound on the total weight over all labelings. Therefore, it is acceptable to check whether $rm^i \leq max(p_{max}, p_{sum})\epsilon$. If we ignore all scenarios where this inequality holds, then the overall error will be no more than ϵ, while we will observe large gains in execution speed.

For additional theoretical and empirical speed, we implement a pre-processing step for this algorithm. We re-order each distribution S by decreasing certainty, $\max_l(W(S, l))$, which is explained by the following intuition. It is when we consider an unlikely label for the distributions for which we have strong suspicions about their labels that we are likely to be able to terminate our search early. By placing near the beginning of the list those distributions, we decrease the expected number

of calculations that will be required.

Also, note that because the probabilities in question can become very small, we take the standard approach where most likelihoods are actually stored as the logarithm of the corresponding likelihood, to avoid underflow.

Now we wish to verify empirically that this approximation of disregarding any combination of labels having probability less than ϵ does not make an appreciable difference in the values to which EM converges.
To this end, we tested it on the Ionosphere dataset [16]. In this test, all 351 datapoints in the Ionosphere dataset were used, but only the labels for 100 of them (chosen randomly) were provided to EM. We informed the algorithm to model the mixture with 12 Gaussian distributions. We ran the typical EM algorithm ($\epsilon = 0$), which did not disregard any combinations of labels, and then we ran a variation ($\epsilon = 1$) that aggressively disregarded any combination with probability less than the most likely combination observed so far. Note that the termination criteria in EM was to stop whenever 0.001 was greater than the sum of all components of all movement vectors of all centers of all distribution. When both algorithms terminated, the largest difference between corresponding components of the covariance matrices was 1.1325e-009, and the largest difference between corresponding components of the centers of corresponding Gaussian distributions was 3.5074e-008. This shows that the most-likely combination of labels contributes a large majority of the total weight in calculating the means of the Gaussians and the covariance matrix.

We then ran the same test again, except that we set $\epsilon = 0.0001$ for the second run. The largest difference between corresponding components of the covariance matrices was 6.9389e-015, and the largest difference between corresponding components of the centers of corresponding Gaussian distributions was 2.1030e-013.These errors were much smaller than the termination threshold (0.001) used in EM, which means that the error caused by disregarding substantially unlikely scenarios is substantially less than the error caused by terminating after the total movement of all

centers of all distributions totals less than 0.001.

3.3 Selection Algorithms

With the Expectation Maximization portion covered, we now discuss the algorithms being used to choose the next x_j to request for labeling.

There are four algorithms being considered.
The first algorithm we shall refer to as the Myopic Algorithm, even though it is not the only one that takes a greedy approach. It requests for labeling the x_j that will minimize the expected error rate. Its pseudocode is as follows:

Run EM
for all datapoints x, labels L **do**
  $P(x, L) \leftarrow \Pr(L_x = L)$
end for
$minError = \infty$
for all unlabeled datapoints x **do**
  $errorRate = 0$
  for all labels L **do**
    $L_x \leftarrow L$
    for all x **do**
      for all J **do**
        recompute $\Pr(L_x = j)$
      end for
    end for
    $currentErrorRate \leftarrow \sum_{i=1}^{n} (1 - \max_j \Pr(L_x = j))$
    $errorRate \leftarrow errorRate + P(x, L) currentErrorRate$
    $L_x \leftarrow unlabeled$
  end for
  if $errorRate < minError$ **then**
    $minError \leftarrow errorRate$

$chosenPoint = x$
end if
end for
return $chosenPoint$

We desire the classifier with minimum error without spending an exorbitant amount of execution time. We present other, simpler algorithms against which to compare.

The second algorithm, which we call the Max-Uncertainty Algorithm, requests for labeling the x_j that has highest uncertainty. It executes more quickly than the previous algorithm, and its pseudocode is as follows:

Run EM
$minUncertainty = \infty$
for all unlabeled datapoints x **do**
$uncertainty \leftarrow (1 - \max_j \Pr(L_x = j))$
if $uncertainty < minUncertainty$ **then**
$minUncertainty \leftarrow uncertainty$
$chosenPoint = x$
end if
end for
return $chosenPoint$

The third algorithm, known here as the Fast Algorithm, attempts to approximate the Myopic Algorithm while executing much more quickly. Its pseudocode is as follows:

Run EM
$bestScore \leftarrow 0$
{Calculate the expected number of unlabeled datapoints belonging to each distribution}

for all distributions S **do**
  $W_S \leftarrow 0$
  for all datapoints x whose label is unknown **do**
    $W_S \leftarrow W_S + \Pr(x \in S)$
  end for
end for
{Get the probability that the label of any particular distribution does not match what we predict}
for all distributions S **do**
  $E_S \leftarrow (1 - \max_l(\Pr(L_S = l)))$
end for
{Estimate the decrease in error rate caused by choosing any particular datapoint}
for all datapoint x whose label is unknown **do**
  {calculate the probability that this datapoint is mislabeled}
  $currentScore \leftarrow (1 - \max_l(\Pr(L_x = l)))$
  {add an estimate the change in expected error rate brought on by the acquisition of the label of this this datapoint}
  $currentScore \leftarrow currentScore + \Pr(x \in S) W_S E_S$
  if $currentScore > bestScore$ **then**
    $bestScore \leftarrow tempScore$
    $bestPoint \leftarrow x$
  end if
end for
return $bestPoint$

The fourth algorithm is the Random algorithm, and it is used as a control algorithm to which to compare. An active learner that randomly selects an unlabeled datapoint is equivalent to a passive learner that merely receives a randomly selected training set. The pseudocode for the Random algorithm is:

return a random unlabeled datapoint

4. Experimental Results

4.1 Synthetic Datasets

To strengthen the proof of concept, we tested the above algorithms on synthetic datasets. The first synthetic dataset consisted 20 datapoints drawn at random from each of four Gaussians, each with covariance matrix equal to the identity. The four Gaussians formed a square, with 4 standard deviations orthogonally between their centers. The two Gaussians with largest y-coordinates were assigned label 2, while the other two were assigned label 1. We also generated a test set at random (not shown) containing 200 datapoints from each distribution.

Observe that the error rates converge to zero for the unlabeled portion of the training set but not for the held-out test set. The active learner is able to systematically select for labeling those points on which it is uncertain, thereby removing them from the error calculation for the provided dataset. It does not have the opportunity to do this for points in the test set, however.

The error rate of the Myopic algorithm reliably falls quickly in the first few iterations because most early labels provides a good estimate of the label of one particular distribution. the Max-Uncertainty algorithm does not necessarily estimate the labels of each distribution as quickly, and its average error rate requires more labels before reaching an equally small level. The Max-Uncertainty algorithm chooses an arbitrary first datapoint, which may be considered a random choice, and its next few choices may also be suboptimal. Also note that the Random algorithm converges most slowly.
Also observe that as expected, the error rate on the test set does not converge to zero. Even if the classifier learns the correct model parameters, there are still regions of high uncertainty between nearby distributions.

To see that the model is appropriate for the synthetic dataset, observe below

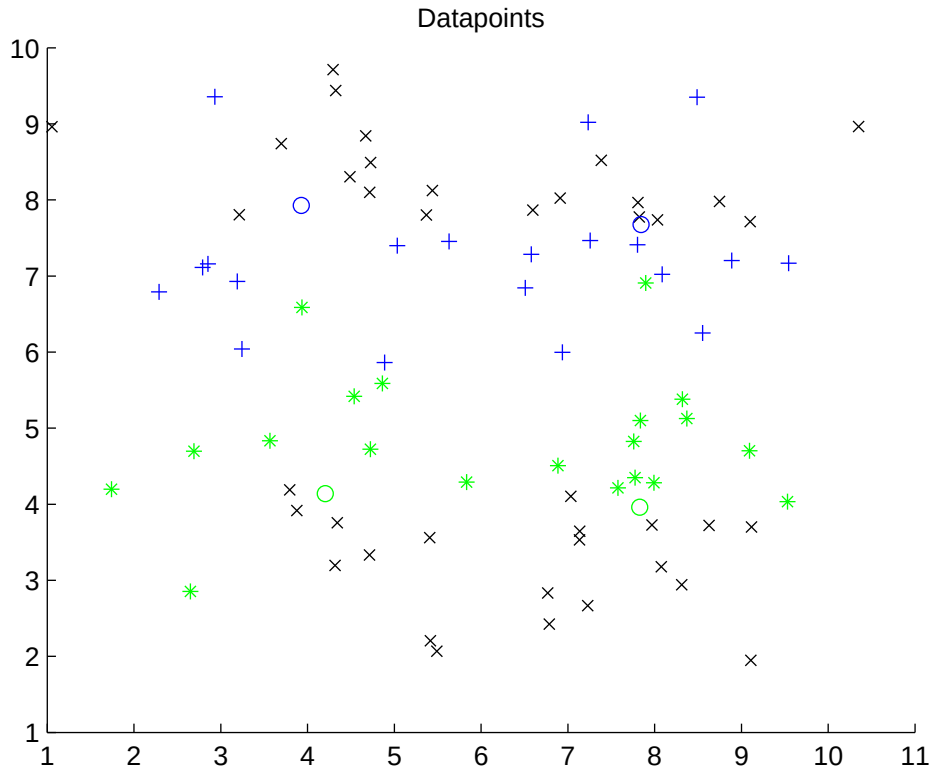


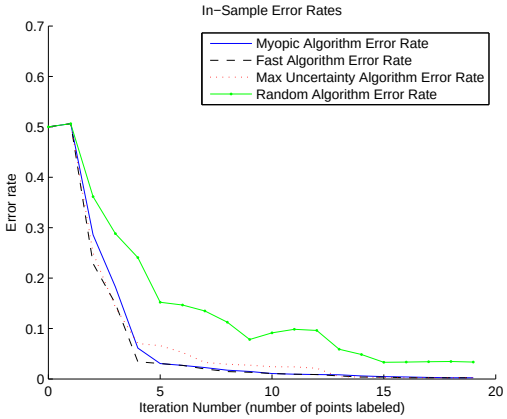
Figure 4.1: The 2x2 synthetic dataset.

the average actual and average expected error rates for the myopic algorithm as a function of the number of iterations.

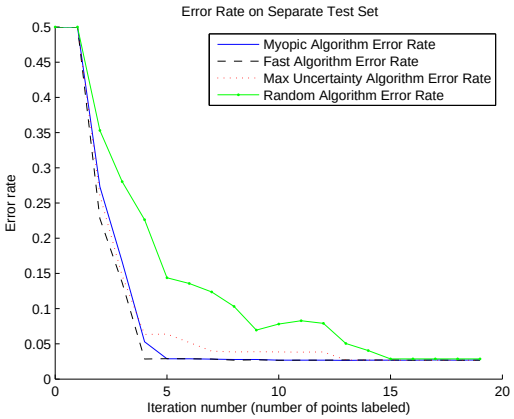
The second synthetic dataset consisted of 20 datapoints drawn at random from each of of nine Gaussians, each with covariance matrix equal to the identity. The nine Gaussians formed a grid, with 4 standard deviations orthogonally between their centers. During ten of the twenty trials, five alternating Gaussians were assigned label 0, while the other four were assigned label 1. During the remaining ten trials, the polarities were reversed.

We also generated a test set at random (not shown) containing 200 datapoints from each distribution.

Here we show the error rates, averaged over 20 trials, for each algorithm tested on the 3x3 synthetic dataset.



Error rates on the training set



Error rates on the test set

Figure 4.2: Error rates on the unlabeled portion of the 2x2 synthetic dataset, along with error rates on a held-out test set, averaged over 20 trials.

When observing one trial of the Max-Uncertainty algorithm, it was clear that the Max-Uncertainty algorithm waited many iterations before obtaining a high degree of certainty of the label of the distribution on the right-middle, the distribution at $(x, y) = (12, 8)$. This distribution made up $\frac{1}{9}$ of the entire dataset, and was not labeled correctly for approximately 80 iterations. Presumably, similar effects occurred in other trials.

To see that the model is appropriate for the synthetic dataset, observe below the average actual and average expected error rates for the myopic algorithm as a function of the number of iterations.

4.2 Real-World Datasets

Next, we applied the described active learner to the Iris dataset [16]. The Iris dataset contains three flower types, each corresponding to a different class, and 4 numerical dimensions associated with each datapoint. Three tests were run on this dataset. The first test was to classify the first distribution as distinct from the other two.

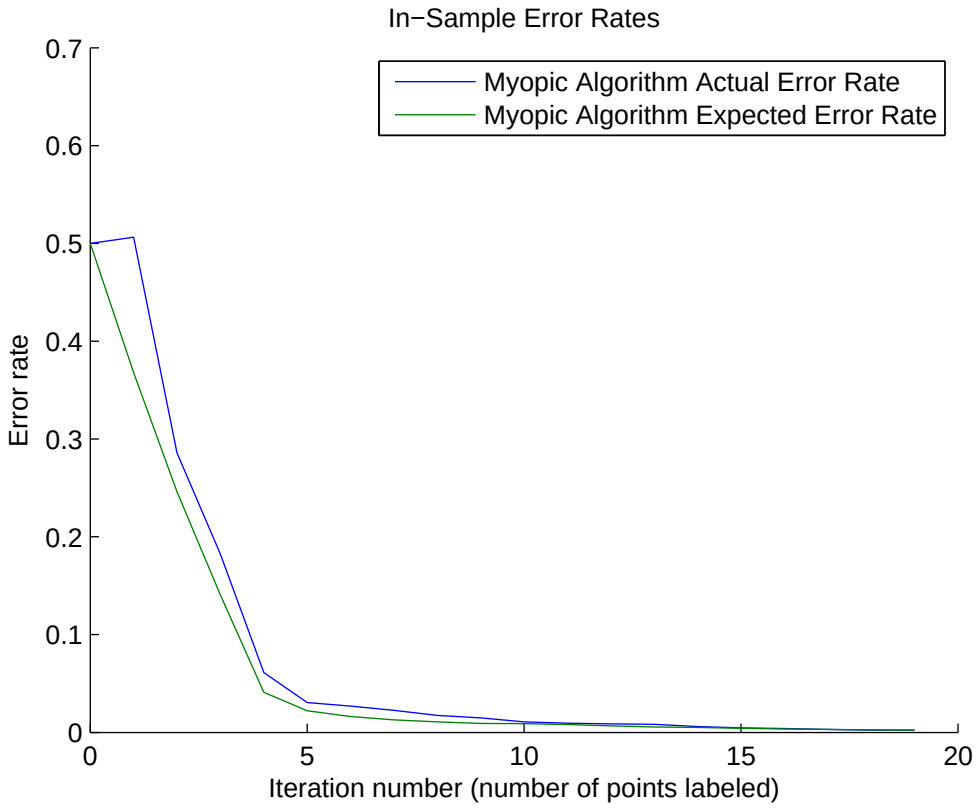


Figure 4.3: Actual and expected in-sample error rates on the synthetic 2x2 dataset, averaged over 20 trials.

When the learner has no data, it must assume the label of all datapoints, which in this case, will yield an error rate of $\frac{1}{3}$ or $\frac{2}{3}$, depending on which label type we arbitrarily choose to assume. So, we calculate and plot the error rate averaged over both cases.

Also note that for the Random algorithm, we averaged the error rate over 20 trials, 10 of each polarity.

The second test was to classify the second distribution as distinct from the other two. We average over 20 trials, 10 of each polarity, as in the previous test.

The third test was to classify the third distribution as distinct from the other two. We average over 20 trials, 10 of each polarity, as in the previous test.

We also applied all four active learners to a dataset containing measurements about abalone [16]. This 8-dimensional dataset contains 1 categorical attribute

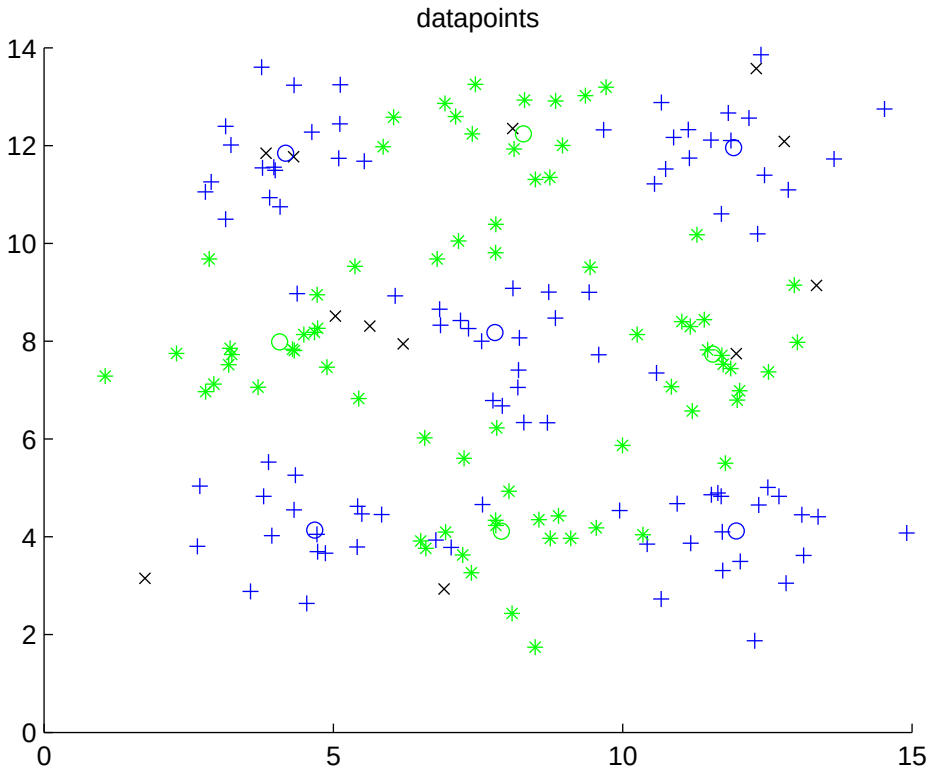
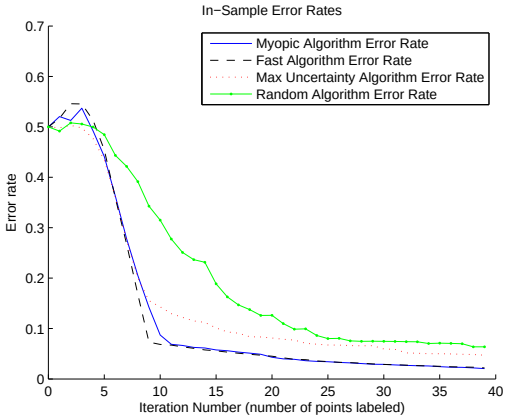


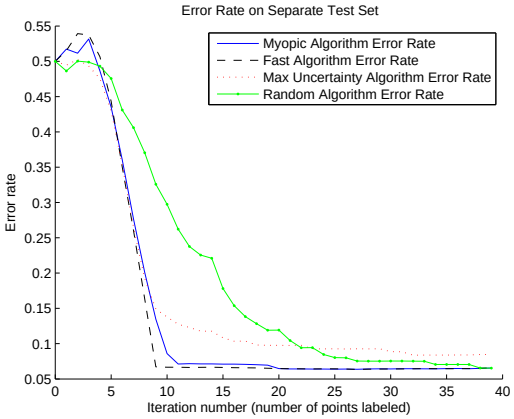
Figure 4.4: The 3x3 synthetic dataset.

(gender $\in$ {male, female, infant}) and 7 numerical real attributes, in addition to the value to be predicted: the number of rings in the abalone (which equals the age of the abalone in years, minus 1.5). We used only the numerical values and ignored the categorical variable, which is not currently compatible with our algorithm. We set to 1 the label of any abalone with 14 or fewer rings, and set to 2 the label of any other abalone. We instructed each learner to use 10 Gaussian distributions with which to model. We ran 20 trials, during each of which we randomly selected 100 datapoints with label 1 and 100 with label 2, to form the dataset for the trial. We chose an additional 200 datapoints of each label type to form the test set. The values plotted are the average error rate over all trials for the given algorithm at the given iteration.

Early on in the training process, it is beneficial to request the label of a datapoint that will provide high certainty in the label of one particular distribution. Later on, it is preferable to choose a datapoint whose label is unknown. The ad-



Error rates on the training set



Error rates on the test set

Figure 4.5: Error rates on the unlabeled portion of the 3x3 synthetic dataset, along with error rates on a held-out test set, averaged over 20 trials.

vantage of the Myopic algorithm over the Max-Uncertainty algorithm, therefore, is primarily in the early iterations, and is greater when there are many distributions whose labels must be learned.

To quantify the level of appropriateness of the model to the Abalone dataset[16], observe below the average actual and average expected error rates for the myopic algorithm as a function of the number of iterations. There is a larger discrepancy between actual and expected error rates for the real-world dataset than for the synthetic datasets.

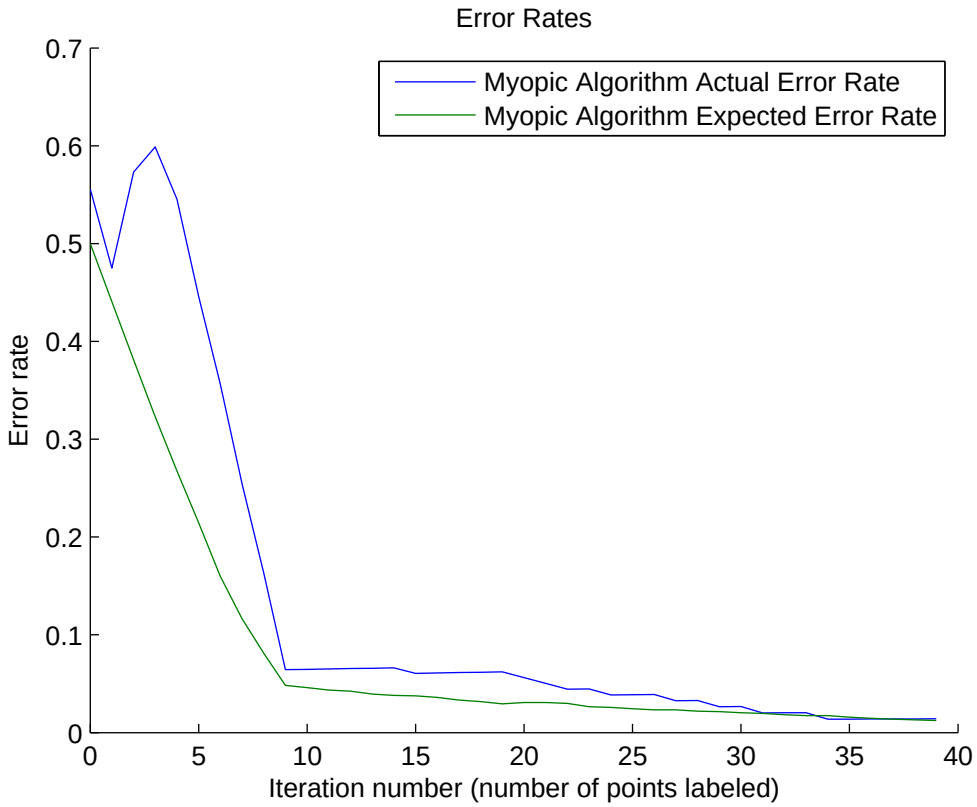


Figure 4.6: Actual and expected in-sample error rates on the synthetic 3x3 dataset, averaged over 20 trials.

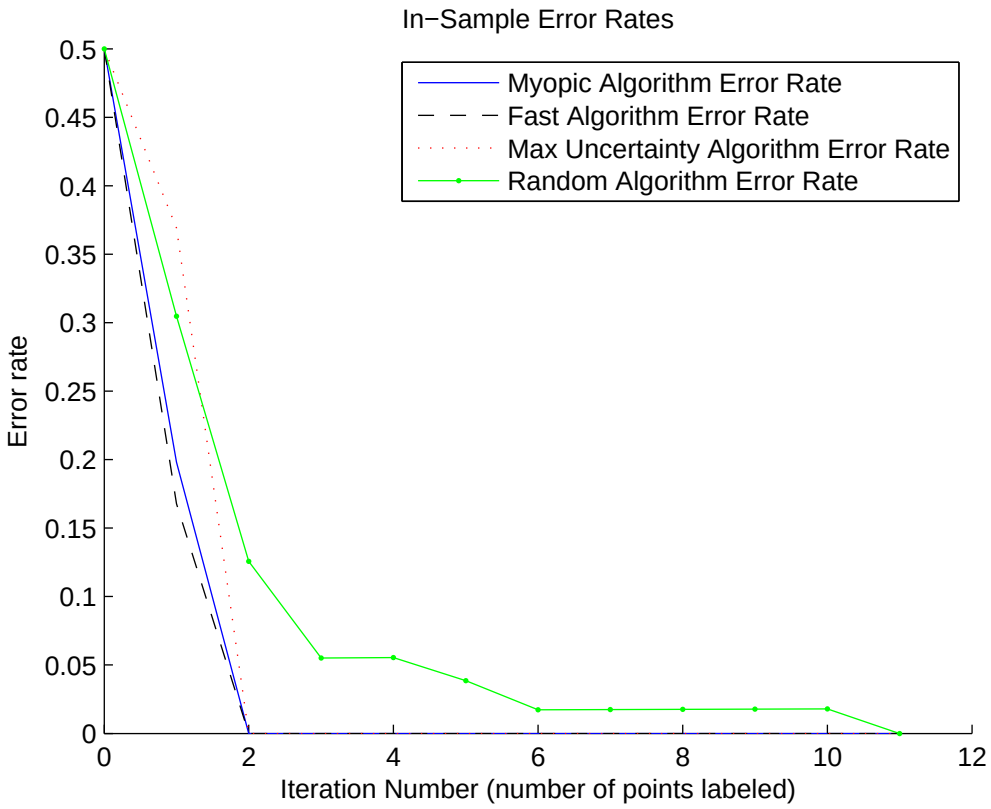


Figure 4.7: Error rates on the Iris dataset when datapoints of the first flower type are assigned label 1 and remaining datapoints are assigned label 2.

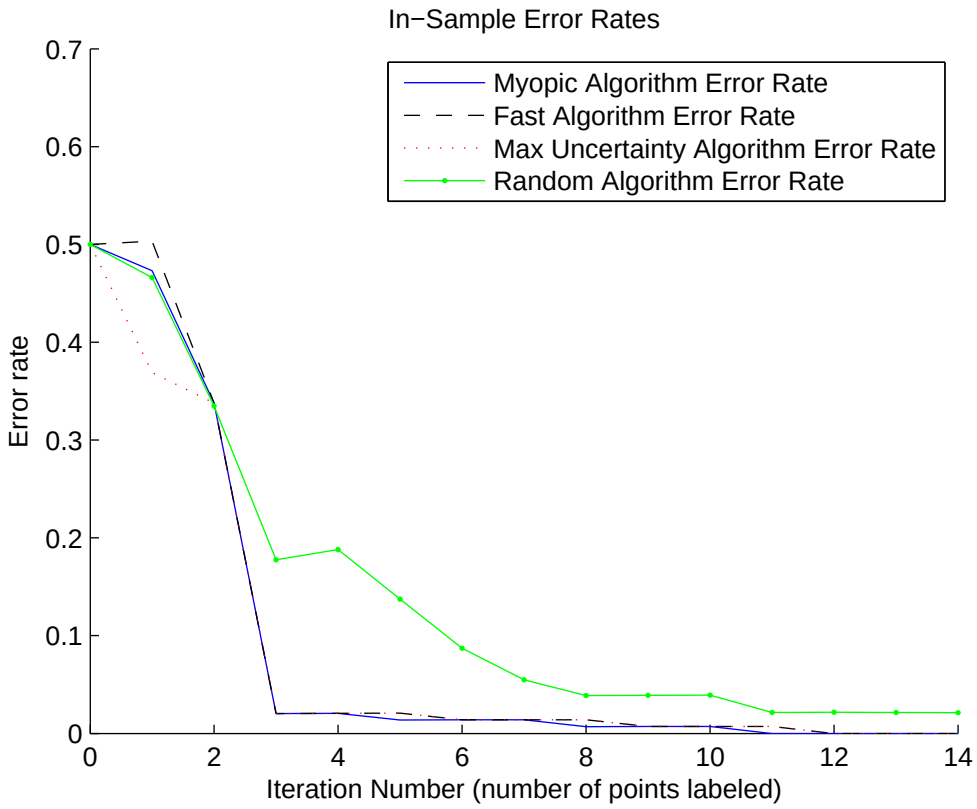


Figure 4.8: Error rates on the Iris dataset when datapoints of the second flower type are assigned label 1 and remaining datapoints are assigned label 2.

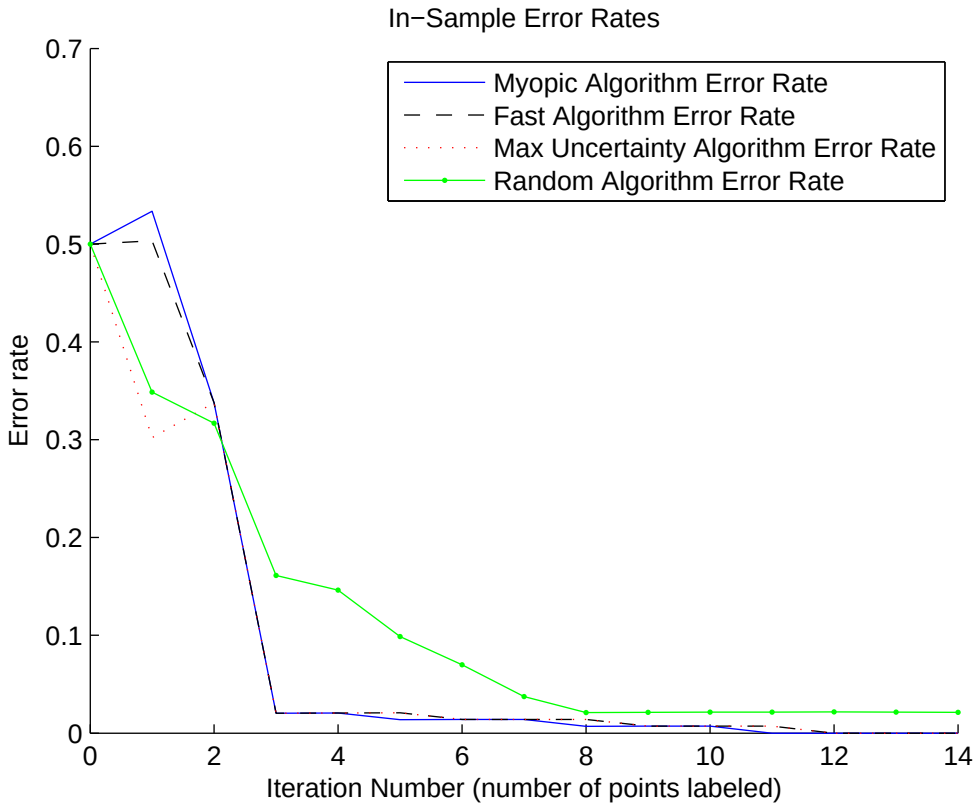
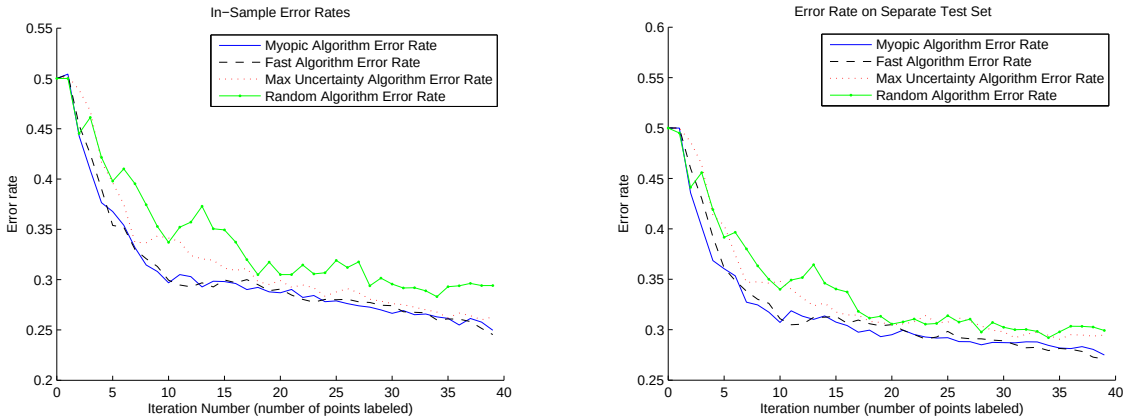


Figure 4.9: Error rates on the Iris dataset when datapoints of the third flower type are assigned label 1 and remaining datapoints are assigned label 2.



Error rates on the Abalone training set

Error rates on the Abalone test set

Figure 4.10: Error rates on the Abalone dataset, averaged over 20 trials.

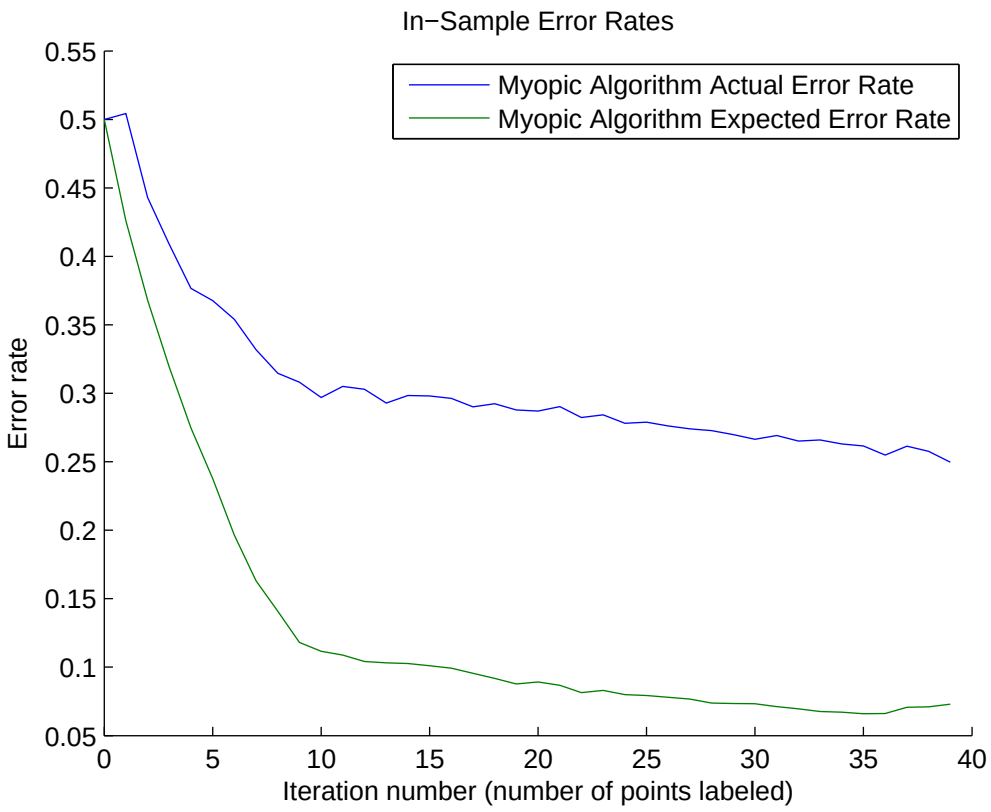


Figure 4.11: Expected and actual in-sample error rates on the Abalone dataset, averaged over 20 trials.

5. Conclusions

In this thesis, we explore methodologies for active learning. In particular, we develop an algorithm that directly (myopically) minimizes expected future error for Gaussian Mixture Models, in contrast to the more typical methods that focus on picking points of maximum uncertainty.

Experimentally, we observe that on a classifier having many parameters (e.g. a Gaussian Mixture Model involving many Gaussian mixture components), a myopic approach directly minimizing the expected error can outperform the common Max-Uncertainty approach, with the majority of the improvement occurring early in the learning process, while there is still high uncertainty in some model parameters (the labels of each mixture component). Evidence confirms that the Myopic algorithm outperforms the Max-Uncertainty algorithm by a larger margin when there are greater numbers of model parameters to be learned. If each label is costly to obtain, then any such improvement corresponds to a substantial real-world savings.

We even find that an approximate estimate of the change in values in model parameters due to newly acquired labels can be adequate to attain a lower error rate than the Max-Uncertainty algorithm. In fact, a heuristic based on learning one mixture component at a time may yield a learner more efficient overall than the Myopic algorithm.

After the acquisition of several labels, the dominating factor in the choice of the next label to request is that the chosen label is removed from the error calculation. Thus, datapoints having high uncertainty become desirable candidates for labeling. Labels received in these conditions need not change the classifier substantially or alter the error rate on the test set significantly.

Consistent with the observation that the majority of the learning occurs dur-

ing early iterations, the error rate of the classifier on a held-out test set does not converge to zero. Datapoints near the decision boundary are expected to have highly uncertain labels.

The implementation of a Gaussian Mixture Model appears to require the acknowledgment of, if not necessarily the evaluation of, an exponential number of priors with respect to the number of mixture components. If we will tolerate an arbitrarily small amount of error, then we usually can greatly reduce the runtime requirements when there are many mixture components in the model.

Due to the inaccurate assumption that the centers of all distributions are equal to their assumed values when calculating error rates, the calculated expected error rate is often lower than the actual error rate. When mixture components are not Gaussian in nature, the expected error rate appears to often be lower than the actual error by an even larger margin. However, even when mixture components are not Gaussian distributions, the Myopic algorithm can still outperform the Max-Uncertainty algorithm.

LITERATURE CITED

[1] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. 18th Int. Conf. Mach. Learning*, pp. 441–448, Morgan Kaufmann Publishers Inc., 2001.

[2] X. Zhu, "Semi-supervised learning literature survey," 2005.

[3] D. Cohn, Z. Ghahramani, and M. Jordan, "Active learning with stat. models," *J. Artificial Intell. Research*, vol. 4, pp. 129–145, 1996.

[4] X. Zhu and A. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures Artificial Intell. and Mach. Learning*, vol. 3, no. 1, pp. 1–130, 2009.

[5] A. Goldberg, X. Zhu, A. Singh, Z. Xu, and R. Nowak, "Multi-manifold semi-supervised learning," in *12th Int. Conf. Artificial Intell. and Stat. (AISTATS)*, pp. 169–176.

[6] D. Angluin, "Queries and concept learning," *Mach. Learning*, vol. 2, no. 4, pp. 319–342, 1988.

[7] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[8] K. Nigam, A. McCallum, and T. Mitchell, "Semi-supervised text classification using EM," *Semi-Supervised Learning*, pp. 33–56, 2006.

[9] B. Juang, W. Hou, and C. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. Speech and Audio Process*, vol. 5, no. 3, pp. 257–265, 1997.

[10] S. Tong and D. Koller, "Support vector mach. active learning with applicat. to text classification," *J. Mach. Learning Research*, vol. 2, pp. 45–66, 2002.

[11] S. Dasgupta and D. Hsu, "Hierarchical sampling for active learning," in *Proc. 25th Int. Conf. Mach. Learning*, pp. 208–215, ACM, 2008.

[12] D. Reynolds and R. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Trans. Speech and Audio Process.*, vol. 3, no. 1, pp. 72–83, 1995.

[13] C. Wu, "On the convergence properties of the EM algorithm," *Ann. Stat.*, vol. 11, no. 1, pp. 95–103, 1983.

[14] Y. Zhang and M. Scordilis, "Optimization of GMM training for speaker verification," in *ODYSSEY04-Speaker and Language Recognition Workshop*, pp. 231–236.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and D. E., "Scikit-learn: Mach. Learning in Python ," *J. Mach. Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] A. Frank and A. Asuncion, "UCI mach. learning repository," 2010. Available: http:archive.ics.uci.edu/ml [Apr. 3, 2012].