Active Learning: Effects of query strategy and number of labeled samples on performance of classifiers

Authors

Connor Flynn
Dor Rubin
Jenna Warren

Abstract

This project examines the effect of training sample size and selection of training samples via active learning on the classification rate of generative and discriminant classifiers. Using random sampling as a baseline comparison, we examined the change in performance of several classifiers as we gradually increased the number of labeled training samples. We then compared the number of labeled training samples needed for each query strategy to match the classification performance of passive learning. We found that although performance is strongly tied to the dataset and the base classifier, one is able to gain up to a 90% reduction in the number of training samples needed.

1. Introduction

The influx of data has made machine learning applications ubiquitous. While data is often readily available, labeling said data can be inefficient and/or costly. This cost has given rise to what is called active learning. Active learning is a form of optimal experimental design in which one can optimize the training of a machine learning classifier with a training size smaller than the passive learning strategy. Using an intelligently selected set of labeled training points, one can maintain or surpass classification accuracy in terms of performance metrics such as correct classification rate (CCR).

As a starting point for our own study, we investigated the Pascal2 active learning challenge program hosted by Clopinet Labs. Contestants were asked to implement an algorithm using the classifier of their choice to predict labels on various datasets starting with only a single labeled training point [5]. Participants could “purchase” additional labels for a fixed cost using different active learning query strategies. It was found that 81% used uncertainty sampling, 47% used random sampling, and 38% used query-by-committee. Despite the ability to also use active learning in a semi-supervised approach, the majority of winning teams were able to achieve higher performance through a fully supervised learning approach. [5]

Using the competition as a launching point, our research explores some of the popular query strategies used and its affect on generative and discriminant classification performance. The goal in this experiment, is to investigate how the number of labeled data points as well as the strategy for selecting those points affects the classification rate versus a simply random selection. We hypothesized that more advanced querying techniques would improve performance with fewer labelled samples than a random querying technique. Likewise, we hypothesized that we would see a higher global maximum correct classification rate as the specific sample points selected would better capture the underlying patterns in the data and result in a more representative model. The challenging aspect of this work is the implementation of active learning on datasets in which the best classifier is unknown, as noted in the literature [8].

2. Literature Review

The primary area of research in active learning involves the methods and the metrics used for selecting which points to label for use in training the classifier. This is known as querying, and the entity that labels the selected points is known as the oracle. Depending on the application, the oracle may be a human, such as a medical professional labeling biomedical images. The primary querying strategies used in the project are random querying, uncertainty sampling, and query by committee. Querying based on expected model change, and variance reduction are summarized in the appendix to give examples of other strategies currently being pursued in the field but which were not the focus of the study.

2.1.1 Random Querying

Random querying is the simplest form of querying. Random querying involves randomly selecting unlabeled training samples to be labeled by the oracle and added to the training set. It was utilized to provide a baseline comparison of the performance of each query strategy that we implemented.

2.1.2 Uncertainty Sampling

The motivation of uncertainty sampling is, as the name implies, to query the label that the learner is the most uncertain about. One such query metric is to find the unlabeled training sample that for which the classifier is least confident. This is a fairly simple concept for probabilistic models. For example, in binary classification, using a generative classifier one would query the unlabeled point whose posterior distribution was closest to uniform across all classes. Using a geometric, discriminant model such as SVM, one would minimize the distance to the decision hyperplane (equation 4 in the appendix.)

For multi-class data, another query metric for uncertainty sampling that may be used is called margin sampling [8]. Margin sampling looks at the difference in the probability of selecting the two most probable classes given a particular unlabeled sample.

$$x_{query} = argmin_x[P(y_{best}|x) - P(y_{2nd\ best}|x)] \quad (1)$$

The training sample in which this difference is minimized (that is, the training sample in which the probability of the two most likely labels is the most similar) is the one that is chosen for labeling by the oracle.

A more standard metric used in uncertainty sampling is entropy.

$$x_{query} = argmax_x[-\sum_{i=1}^{k} P(y_i|x) * logP(y_i|x)\ ] \quad (2)$$

Using this metric, we look to query the data point that maximizes entropy.

2.1.3 Query by Committee

Query by committee is a more complex querying strategy which involves an ensemble of two or more classifiers, called a committee. The goal of query by committee can be described as choosing the unlabeled data point of maximal disagreement or more technically, the unlabeled data that will best minimize the version space created by the committee. In other words, the goal of query by committee is to select unlabeled data points that will cause this committee of classifiers to agree on the label of potential test points after retraining. While there are no set rules on the number of committee members to use, it is generally common practice to use three [8].

There are two main approaches for measuring the level of disagreement in the committee. The first is an extension of information entropy as seen in uncertainty sampling. This metric is known as vote entropy. Vote entropy is like entropy, only the average number of votes across the committee takes the place of the probability distribution.

$$x_{query} = argmax_x[-\sum_{i=1}^{k} \frac{V(y_i)}{C} * log\frac{V(y_i)}{C}\ ] \quad (3)$$

Again, because the query strategy seeks to resolve the most uncertainty through gathering labels from the oracle, we seek to select a unlabeled data point which maximizes vote entropy.

Yet another query metric used in query by committee is the XOR heuristic. This metric selects all points of which at least two classifiers disagree. While colloquially this querying metric is known as "XOR," [11] it does have some differences to an XOR logic gate as seen in Table 1 of Appendix B.

Another approach to query by committee is one which utilizes KL divergence. In this case, the KL divergence is used to measure the difference in the posterior distribution of the committee as a whole (i.e. voting together) and the distribution of a single committee member.

$$x_{query} = argmax\ \frac{1}{C}\sum_{c=1}^{C} D(P_{\theta(c)}||P_C)$$

Where $P_{\theta(c)}$ is the label distribution of a given committee member, P_C is the aggregate label distribution of the full committee, C is the committee size. The point which maximizes the average of the KL divergence between all of the committee members and the committee as a whole is the sample that is labeled by the oracle.

2.2.1 Active Learning Scenarios

In addition to querying, the way data becomes available to the algorithm represents an important dimension of active learning. While this is not the focus of this study, it warrants some attention nonetheless.

The unlabeled data may be presented to the algorithm in three different scenarios. The first of these scenarios is a pool-based scenario. This simply means that all of the unlabeled data points are available from the start of active learning. In this scenario, the active learner can compare the utility of all unlabeled data points and assess the impact of each on the model. This sampling strategy appears to be the most common in the literature, likely due to its simplicity. The second scenario is known as stream-based active learning. This is when data is continuously fed to an active learner over time, thus the learner can only assess the utility of current and previously seen points, without knowing the utility of the unlabeled data it may see in the future. Finally, de novo sampling turns the constraint of querying points from a given distribution on its head and lets the active learner generate ideal points to be labeled. Notably, this scenario has received some criticism in that in a real world application, generating new data to query may confuse those who are labeling it with combinations of features that are not realistic [8].

3. Problem Formulation and Implementation

3.1 Performance Metrics

As stated in the introduction, the goal of the study was to track how well an intelligent query strategy performs relative to a random one. To quantify this, we used three different metrics. The first is correct classification rate (CCR). It was the primary metric for comparison since we were analyzing a classification problem. Second was the difference in area under the classification rate curve. Area calculation is a function of both CCR and number of samples. Though not a standard metric in research, this number gave us an idea of the aggregate learning especially as training size increased. The third and final metric was the sample savings. This was calculated by finding the number of samples used to reach the highest CCR with random selection subtracted by the minimum number of samples necessary to achieve the same CCR using a particular intelligent query strategy.

3.2 Base Classifiers

In many ways the classifiers used were abstracted away from our experimental design. Quadratic Discriminant Analysis (QDA) and Support Vector Machines (SVM) were used as representations for Generative and Discriminant models respectively. Since the goal was to assess the effects of active learning, optimizations to tuning parameters through cross validations to improve classification performance were not included. We used MATLAB's implementation of both classifiers as a black box with the following input parameters. To prevent runtime errors, the `pseudoQuadratic` input forced the algorithm to produce the `pseudoinverse` of the covariance matrix Σ_k .[11] For SVM, a radial basis function auto scaled kernel was used with standardized flag set to `true`.

3.2 Training , Testing, and Seeding

Before any query strategy was used, the datasets were partitioned into two randomly selected groups for training and testing. Both the training and testing groups remained constant for each trial. Additionally, the same test set was used at each iteration of incremental training size.

Because active learning is incredibly sensitive to the initial seed used, we ensured consistency between the random and intelligent query strategies by using the same exact seed for the respective trials -- though the initial seed was randomized between trials to get a balanced representation of the types of seed distributions used to feed the algorithm.

The number of seeds used was derived from the Probably Approximately Correct learning model which states that given that the underlying data distribution can be perfectly classified by some hypothesis θ, then if ε is the maximum desired error rate, $O(1/\varepsilon)$ random labeled instances are needed. [7, 3] An extension of this theory is that using binary search in a perfectly separable dataset, a queried algorithm can find the optimal decision rule in $O(\log(1/\varepsilon))$ samples. Since the passive learning minimum error rate was 0.05, an optimal initial seed would have been roughly 20. We rounded up to the first power of 2 to a seed size of 32 training samples. This was consistent across the two datasets used.

Selected points are represented in a logical vector initialized to all 0. At each iteration the training sample size is doubled and indices of the selected data points are set to true. The base classifiers then use only those points to train the model.

3.3 Implementation of Query Strategies

The following sections are detailed accounts of the implementation of each query strategy. In this study uncertainty sampling was implemented in three ways: pure uncertainty sampling, uncertainty sampling scaffolded by randomly selected points, and density weighted uncertainty sampling. Query by committee was implemented in two different ways: pure query by committee and query by committee scaffolded by randomly selected points.

3.3.1 Uncertainty Sampling

In the purest sense, only uncertainty metrics were used to select points to be labeled. In the case of the SVM model, we determine uncertainty based on proximity to the decision hyperplane for classification, as was stated in the literature review. In the case of the QDA model, we implemented margin sampling. Specifically, we found the absolute difference in the posterior probability between the two classes and selected the point that minimized this value.

3.3.2 Uncertainty Sampling with Scaffolded Random Selections

In this implementation, on average, half of the points were selected randomly and half were selected by uncertainty sampling. Rather than selecting half of the samples to be random all the time, we implemented a heuristic that would front load the random selections so that a more stable, representative selection of points would be available early on. The point at which half of the available training size was included in the current training set, the proportion of selections based on uncertainty sampling to random selections would be flipped.

3.3.3 Density Weighted Uncertainty Sampling

The density weighted uncertainty sampling leveraged the underlying data distribution of the unlabeled training data. [11] First, a Gaussian similarity matrix was constructed. The standard deviation for the Gaussian similarity was the standard deviation of all the unlabeled training data. Next the distances for each point was summed and then averaged by the number of unlabeled samples. Finally, the uncertainty metric for a given point was scaled by its average distance to all other unlabeled training points.

3.3.4 Query by committee

In our query by committee strategies we used a committee of three classifiers, SVM, QDA, and Decision Tree. Each of these was trained using the training pool of the current iteration. Next, we used the model to predict the labels of the remaining unselected training samples. Label predictions were tallied for each of the unselected samples, and then using XOR and Voting Entropy techniques, samples were selected in the order of most disagreement. The points which exhibited the most disagreement were selected for labeling.

3.3.5 QBC with Scaffolded Random Selections

We also implemented a strategy identical to that used in uncertainty sampling with scaffolded random selections except that the selection consisted of random points and points selected by a query by committee strategy.

3.4 Datasets

We tested our implementation of the query strategies on several datasets that we generated ourselves. These test datasets consisted of two classes of 2,000 normally distributed points. We then formally tested our implementation on two larger more complex datasets. The spambase dataset of 4,601 samples required classifying whether or not a test sample (email) was spam. It had 58 features of continuous type and approximately 60/40 split in the class distribution. The largest dataset of 20,722 samples was Ibn Sina. It had 92 features of mixed type: binary, categorical, and continuous. This dataset involved identification of arabic characters in an ancient manuscript, and was also a binary classification problem. For both data sets 50% of the data was taken as a fully labeled test set and 50% reserved for training.

4. Experimental Results

For each intelligent query strategy implemented, its performance, given by CCR with respect to the number of labeled samples, was compared to that of a random query strategy with the same number of labeled samples.

Our initial findings were that uncertainty sampling was not performing better than random sampling. Uncertainty sampling has a tendency to select more outliers because they are naturally the points that a classifier may be most uncertain about. In addition, if the seed with which the initial classifier was trained did not contain a representative sample of the data, a downward spiral would ensue and uncertainty sampling would only negatively impact the classification rate until a very large portion of the data was selected, as demonstrated in the following plots.

By combining randomly sampled points along with the points determined to be the most uncertain, we were able to lessen the impact of sampling outliers and surpass the correct classification rates of randomly sampling the data.

Another way to modify uncertainty sampling such that your model is not overwhelmed by outliers is to use density weighted uncertainty sampling [1]. In this approach, we multiply the uncertainty metric by a density metric so that we are ultimately selecting points that the classifier is uncertain about, but which are also representative of the unlabeled data.

Though this strategy also has a tendency to select outliers as the next point for querying, we found that the performance in this scenario was much better than pure uncertainty sampling. This is demonstrated in the following plots.

We found that for the spam recognition dataset, though we didn't see sample savings for any of the strategies, we did see a modest performance boost from mixed uncertainty sampling and query by committee for the QDA classifier. However, for Ibn Sina we did see a 5% reduction in samples needed for training by using the query by committee strategy, though other strategies may yield a better maximum performance given more labeled samples.

SVM saw the most improvement, up to 11% reduction in the number of training samples for spam recognition and up to 90% for Ibn Sina, using uncertainty sampling with scaffolded random selections.

Tables 3 and 4 in Appendix B summarize the results of the query strategies that were implemented with the two base classifiers that were used in our experiment.

5. Conclusion

The difficulty of implementing active learning has been noted before in the literature [2]. In most cases, when working on active learning algorithms, researchers will know which classifiers best fit the data and perhaps even a good seed to start their active learning algorithm with in advance. This is because the quality of the data selected by an active learner is dependent on the current state of the active learner. Thus, a poorly initialized active learner can lead to disastrous performance, referred to as a "cold start." Indeed, it has been said that if the best model and feature class are not understood in advance, one may be better off by simply randomly sampling the data, abandoning more complex learning strategies altogether [8]. That said, when implemented properly we can see that some query strategies do perform better than simply randomly querying.

6. Description of Individual Effort

Connor Flynn: Primary researcher
Dor Rubin: Primary implementer of the code
Jenna Warren: Primary writer of the reports

7. References

[1] Attenberg, J. & Ertekin, S. Class imbalance and active learning. Imbalanced Learning: Foundations, Algorithms, and Applications. First Edition, 2013.

[2] Attenberg, J., & Provost, F. Inactive learning? : difficulties employing active learning in practice. ACM SIGKDD Explorations Newsletter, 12(2): 36-41, 2011

[3] Dasgupta, S., Langford, J. A tutorial on Active Learning. UC San Diego, Yahoo Labs (Doctoral dissertation), 2007

[4] Douak, F., Melgani, F. and Benoudjit, F. Kernel ridge regression with active learning for wind speed prediction. Applied Energy, 103: 328-340, 2013

[5] Guyon, I., Cawely, G., Dror, G., and Lemaire, V. Results of the Active Learning Challenge. JMLR: Workshop and Conference Proceedings, 16: 19-45, 2011

[6] Mamitsuka, N. A. H. Query learning strategies using boosting and bagging. Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98). Vol. 1, 1998

[7] Peng, J., Zhang, P. & Riedel. Discriminant Learning Analysis. IEEE Transaction on Systems, Man, and Cybernetics - Part B: Cybernetics, 38(6), 2008

[8] Settles, B. Active Learning Literature Survey. Computer Sciences Technical Report 1648, 2010

[9] Seung, H. S., Opper, M., and Sompolinsky, H. Query by committee. COLT '92 Proceedings of the fifth annual workshop on Computational learning theory (pages 287 - 294), 1992

[10] Tong, S. Active Learning: Theory and Applications. (Doctoral dissertation). 2001

[11] Singh, A. Active Learning. Carnegie Mellon School of Computer Science. Retrieved from: http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture25.pdf, 2010

[12] Yang, Y., Ma, Z., Nie, F., Chang, X., & Hauptman, A. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization. International Journal of Computer Vision, 113(2): 113-127, 2014

8. Appendix

A. Equations

1. Margin Sampling:

$$x_{query} = argmin_x[P(y_{best}|x) - P(y_{2nd\ best}|x)]$$

Where y_{best} is the label with the highest posterior probability given a generative classifier.

2. Information Entropy:

$$x_{query} = argmax_x[-\sum_{i=1}^{k} P(y_i|x) * logP(y_i|x)\ ]$$

3. Voting Entropy:

$$x_{query} = argmax_x[-\sum_{i=1}^{k} \frac{V(y_i)}{C} * log\frac{V(y_i)}{C}\ ]$$

Where C is the committee size and $V(y_i)$ is the number of votes for a given class

4. Distance to the hyperplane:

$$x_{query} = argmin_x[\frac{w^Tx+b}{||w||}]$$

5. KL Divergence for Committee Querying:

$$x_{query} = argmax\ \frac{1}{C}\sum_{c=1}^{C} D(P_{\theta(c)}||P_C)$$

Where $P_{\theta(c)}$ is the label distribution of a given committee member, P_C is the aggregate label distribution of the full committee, C is the committee size.

6. Density Weighted Uncertainty Sampling

$$x_{query} = argmax_x[Uncertainty(h(x), x) * Density(x)]$$

$$Density(x) = \left(\frac{1}{U}\sum_{i=1}^{U} S(x, x_i)\right)^b$$

$$S(x_i, x_j) = exp\{\frac{-\|x_i - x_j\|_2^2}{2\sigma^2}\}$$

B. Tables

Table 1: “XOR” Query Strategy

Classifier			Value
c1	c2	c3	
1	0	0	1
0	1	0	1
0	0	1	1
1	1	0	1
1	0	1	1
0	1	1	1
0	0	0	0
1	1	1	0

Table 2: Query Strategy Summary

Query strategy		**Main idea**
Random		Randomly sample from the unlabeled set
Uncertainty Sampling	Pure uncertainty sampling	Pick the point of which the classifier is most uncertain
	Mixed uncertainty and random	Sampling both randomly and using uncertainty sampling; Adds a robust initialization
	Density weighted uncertainty sampling	Adds a density bias to uncertainty sampling; Samples points that are more representative of the unlabeled training set
Query by Committee (Ensemble)	Vote Entropy	Pick points which maximize the entropy of an average number of votes for a given label
	XOR	Pick points on which at least two classifiers in a

		committee disagree
	Average KL Divergence	Pick points which maximize the average KL divergence of committee members
Expected model change		Pick points which will influence the model the most
Variance reduction		Pick points which reduce the variance of the training data the most

Table 3: QDA with various query strategies summary

	Spam Recognition		Ibn Sina	
Query Strategy	Difference in Area under under CCR curves	Training samples savings (Percent of Total Training Size)	Difference in Area under under CCR curves	Training samples savings (Percent of Total Training Size)
US	-131.8476	N/A	38.6849	N/A
mixedUS	40.8716	N/A	542.7505	0
DWUS	-777.2191	N/A	-1.1828e+04	N/A
QBC	7.6787	N/A	96.3332	512 (5%)
mixedQBC	26.4981	N/A	545.6649	0

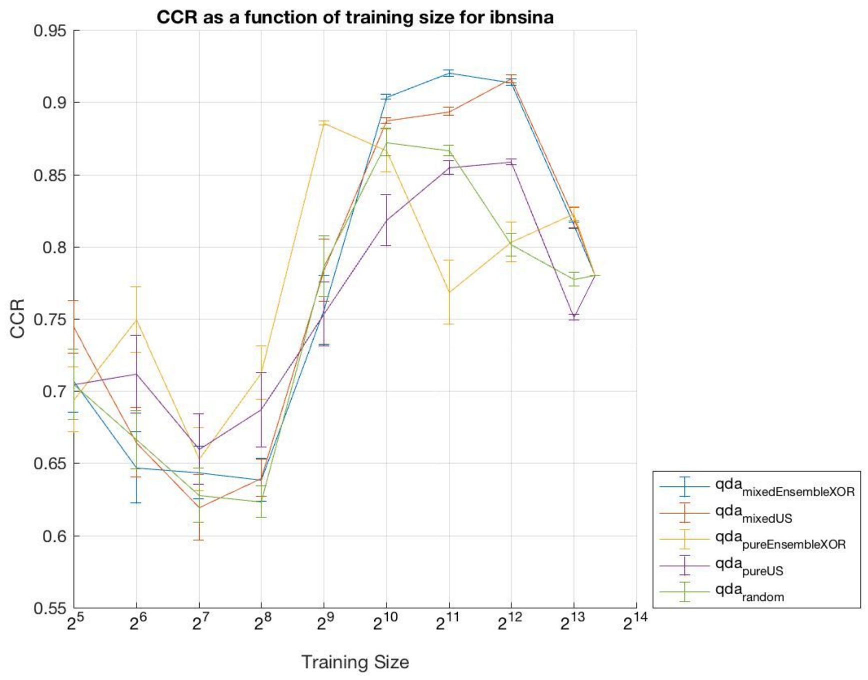
Table 4**:** SVM with various query strategies summary

	Spam Recognition		Ibn Sina	
Query Strategy	Difference in Area under under CCR curves	Training samples savings (Percent of Total Training Size)	Difference in Area under under CCR curves	Training samples savings (Percent of Total Training Size)

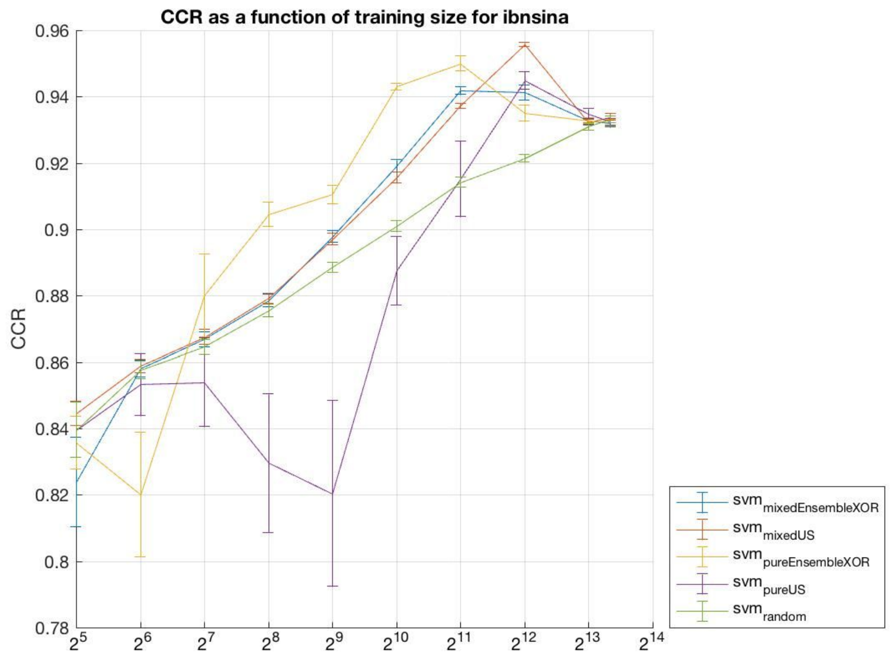
US	6.1992	252 (11%)	38.4494	6265 (60%)
mixedUS	18.9765	252 (11%)	165.1277	8313 (90%)
DWUS	-3.5971e+03	0	-2.3818e+04	N/A
QBC	-137.5868	252 (11%)	146.4670	9337 (90%)
mixedQBC	9.5417	252 (11%)	125.5880	8313 (80%)

C. Figures

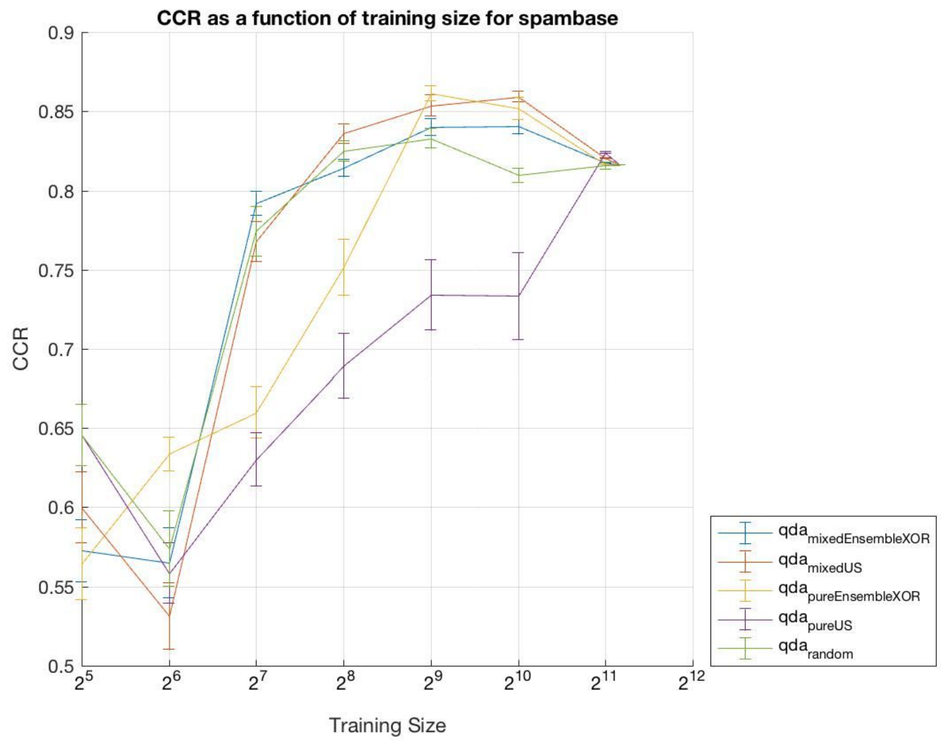
This plot shows CCR as a function of training size for Ibn Sina using the QDA as the base classifier.



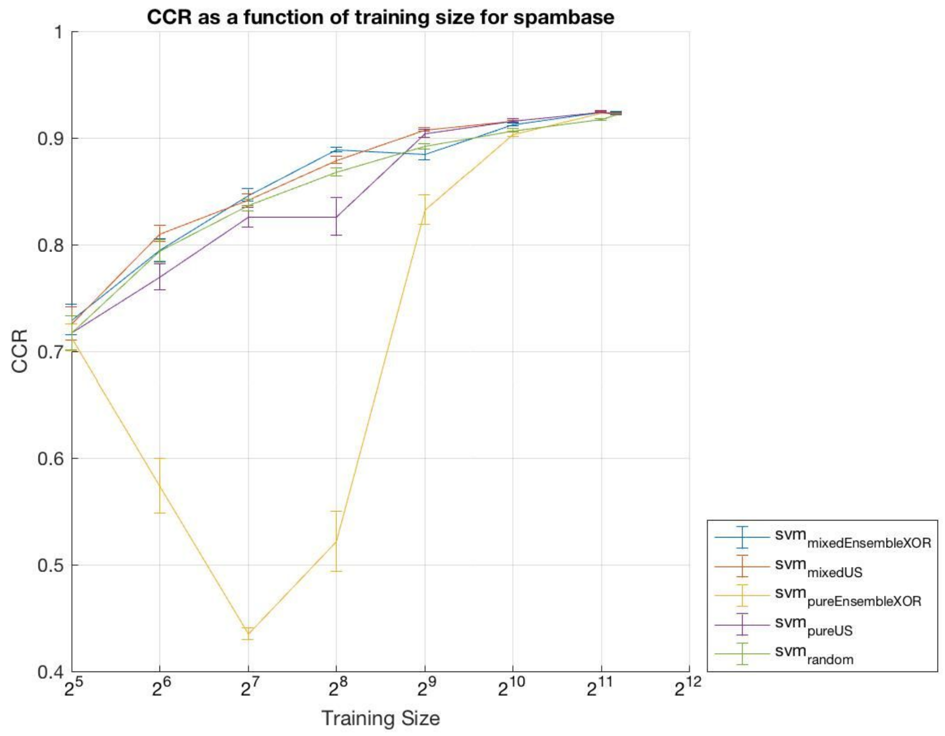
This plot shows CCR as a function of training size for Ibn Sina using the SVM as the base classifier.



This plot shows CCR as a function of training size for Spambase using the QDA as the base classifier



This plot shows CCR as a function of training size for Spambase using the SVM as the base classifier


CCR as a function of training size for spambase
CCR
Training Size
2^5
2^6
2^7
2^8
2^9
2^{10}
2^{11}
2^{12}
$svm_{mixedEnsembleXOR}$
$svm_{mixedUS}$
$svm_{pureEnsembleXOR}$
svm_{pureUS}
svm_{random}