Empirical study on active learning

Robinson Luiz S. Garcia
Department of Computer Science
University of Toronto
Toronto, ON M5S 2E4
robinson.garcia18@rotman.utoronto.ca

Abstract

This work aims to study different active learning techniques known as uncertainty sampling and query by committee. These methods have important practical implications regarding a wider applicability of machine learning. In this paper, 8 datasets were used to test 3 uncertainty metrics with density weighting and without it. In addition, two ramifications of the original query-by-committee algorithm were evaluated and compared with uncertainty sampling. The results show that uncertainty sampling is an effective approach for selecting new samples. In most cases, uncertainty sampling outperformed random picking. In two datasets, query-by-boosting and query-by-committee had excellent performance. However, on average these methods, when applied without hyperparameter tuning, underperformed uncertainty sampling.

1 Introduction

Data is costly in some industries such as the oil and gas. The availability of information limits the application of machine learning models to enhance business results. It is in this scenario that active learning plays an important role supporting humans to make informative decisions regarding which samples must be obtained.

Settles [12] suggest different categories for active learning strategies. This paper focus on the performance of one set of strategies called uncertainty sampling and provide a brief view of a second set called query by committee. In addition, one technique for exploiting structure in data,, called density weighting, is tested with both strategies.

To evaluate uncertainty sampling 8 real datasets were studied (see table 1. For each dataset, a total of 15 trials was executed. At each trial, the data were randomly shuffled and split into train and test sets at a ration of 7 to 3. The only condition imposed by the process was that the train and test label arrays had each at least one sample from each category. This measure was necessary to avoid an error during the fitting process.

Settles and Craven [12] suggest a soft metric to evaluate the output of n committees in a query-by-committee procedure. Basically, the author recommends the application of uncertainty metrics to evaluate the average probability consensus of the committee of classifiers. In other words, a qbc with soft metric turns out to be an ensembling technique that averages the predictions of n classifier been used.

Mamitsuka [7] provides an alternative approach to the original query-by-committee algorithm. The original Query-by-committee proposition uses Gibbs algorithm that picks hypothesis from the hypothesis space and queries the sample in which the predictions are less certain [7]. According to Mamitsuka et al. [7], it is a computationally expensive algorithm. In [7], the authors proposed the use of boosting and bagging to address the efficiency issues.

Table 1: Datasets

Dataset	Classes	Samples	Dimensionality	Source
Wine	3	178	13	sklearn [9]
Digits	10	1797	64	sklearn [9]
Iris	3	150	4	sklearn [9]
Breast Cancer	2	569	2	sklearn [9]
crx	2	690	15	uci [6]
Tic-tac	2	958	9	uci [6]
Pima-indians	2	768	8	uci [6]
liver	2	345	7	uci [6]
*Synthetic	5	2000	2	sklearn [9]

Boosting is designed to be used with weak classifiers although there are no restrictions on using a Logistic Regression as the base classifier. However, in principle, we want to be able to build a large sequence of functions that can represent a complex model. If the functions are complex itself, such as a logistic regression, we may face overfitting or problems setting the models parameters which will yield a loss in performance.

For this reason, we will test boosting and bagging with stumpers (trees with depth of one) but at the end of the day, with a soft metric, the expectation is that the result will be only as good as the relative performance of the classifier being used in the model versus the classifier used in the ensemble.For boosting we use a stochastic gradient boosting tree and for bagging a random forest.

2 Project Diagram

2.1 General Approach to active learning

Figure 3 in the appendix provides a general view of a pool-based active learning procedure. In a nutshell, the program will set aside a pool of unlabelled samples (U). It starts by fitting a model to whatever labeled data is available and then perform a query for new data. The query can ask for a single sample or a group of samples. Now, the query step is where the active learning strategy comes into play. The question is what samples from the unlabeled set to choose? A user could pick samples at random or apply an active learning strategy to make an informed decision. Once a new sample is queried, the task now is to compute the accuracy and update the pools of labeled and unlabeled samples. This cycle is repeated until any particular threshold is reached.

Obviously it is desirable that the active learning strategy perform better than just random picking, otherwise, why bother doing this implementation and lose the fun of gambling with new data? In order to measure the relative performance, **accuracy gain** (τ)will be defined as the active learning accuracy relative to random picking accuracy.

$$\tau = \int_{L_0}^{L} (\mathbf{active}_{accuracy} - \mathbf{random}_{accuracy}) dL \tag{1}$$

On equation 1, L_0 refers to the initial pool of labeled samples, L the ending number of labeled samples and d_L is the size of the queried sample.

2.2 Main algorithim

Algorithm 1 is designed to test different combinations of active learning strategies (table 8) on separate datasets. It starts by picking one dataset from table 1 and splitting it into train and test sets. Each case is fully defined by the subcategory it belongs, the density weighting method and finally, for the query-by-committee family, the uncertainty metric.

The product of algorithm 1 is a list containing the accuracy curve for each case in table **??** given an initial set of labeled data and a constant number of queried samples. The same algorithm was applied for cases of random picking, which are not listed on table **??** but was essential to compute the accuracy gain. The post-processing to obtain the accuracy gain curves and tables are executed on other small scripts.

3 Formal Desciption

3.1 Multinomial Logistic Classifier

A multinomial logistic classifier was used in conjunction with the active learning techniques. The logistic model takes the form:

$$P(Y_n = c|x_n) = \frac{exp(w_c.x_n)}{\sum_{c'} exp(W_{c'}.x_n)} \tag{2}$$

[8] denotes $\mu_i c = p(y_i = c|x_i, W) = S(\eta_i)_{c'}$, where $\eta_i = W^T x_i$ is a $Cx1$ vector. The author let $y_i c = I(y_i = c)$ be the one-of-C encoding of y_i. Finally, set $w_c = 0$, to ensure identifiability, and define $w = vec(W(:, 1 : C - 1))$ to be a D x $(C - 1)$ column vector.

which leads to the following negative log likelihood (NLL):

$$l(W) = log \prod_{i=1}^{N} \prod_{c=1}^{C} \mu_{ic}^{y_{ic}} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} log \mu_{ic} \tag{3}$$

The NLL function can then be writen as:

$$\sum_{i=1}^{N} [(\sum_{c=1}^{C} y_{ic} w_c^T x_i) - log(\sum_{c'=1}^{C} exp(w_{c'}^T x_i))] \tag{4}$$

A l_2 regularization is applied trough all simulations. No hyperparameter search is executed in order to keep conditions between experiments as close as possible. Furthermore, the goal is to measure relative accuracy, thus optimizing the regularization parameter would have little effect in the desired results. With this setting, the solution can be obtained using a gradient based optimizer. The optimization requires the objective, gradient and Hessian, which take the following form:

$$f'(w) = NLL(w) + \lambda w^T w \tag{5}$$

$$g'(w) = g(w) + \lambda w \tag{6}$$

$$H'(w) = H(w) + \lambda I \tag{7}$$

For more details on the implementation of a logistic regression model see[8]

3.2 Uncertainty Sampling

Uncertainty sampling is a technique that selects the next sample based on how uncertain the model is about a given unlabeled sample. To do so, a set of uncertainty metrics can be used. Three metrics that are commonly used are:

- Least confident, measures outliers, samples that the model is lest certain about. In other words, this metric will select samples that the most probable associated category is than the others.
- Margin is the difference in probabilities of the most likely class and the second most likely one. It will steer to samples in which the model is showing a certain level of doubt. The lower the margin, the more likely it is that the prediction is wrong.
- Entropy is a metric that derives from information theory. It is a function that has a maximum when the probability distribution is most uniform. It is similar to margin but differs when the model has more than two classes.

Table 2: Uncertainty Metrics [12]

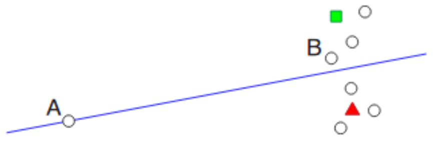
Metric	**Formulation**	$\phi(P_\theta(\hat{y}\|x))$
Least confident	$x^*_{LC} = argmax_x 1 - P_\theta(\hat{y}\|x)$	$1 - P_\theta(\hat{y}\|x)$
Margin	$x^*_M = argmax_x[P_\theta(\hat{y_2}\|x) - P_\theta(\hat{y_1}\|x)]$	$P_\theta(\hat{y_2}\|x) - P_\theta(\hat{y_1}\|x)$
Entropy	$x^*_H = argmax_x \sum_y -P_\theta(y\|x) log P_\theta(y\|x)$	$\sum_y -P_\theta(y\|x) log P_\theta(y\|x)$

Density weighing is a combination of a similarity and an uncertainty metric. The idea is that the model should place more emphasis on samples that are clustered, closer to others, in order to avoid uninformative outliers. The following equation synthesize the concept of weighting

$$x^*_{ID} = argmax_x \phi_A(x)(\frac{1}{U}\sum_{x\prime \in U} sim(x, x'))^\beta \tag{8}$$

We assume a beta of 1 to simplify, but different values could be explored on datasets with many outliers. Sim is a similarity metric and could be cosine similarity, Euclidean etc. We shall use euclidean from here forward. $\phi_A(x)$ is a utility function, which is one of the uncertainty metrics from table 2

Figure 1: Illustration from [11] showing an example of an uninformative outlier that lies on the decision boundary but carries little information about the class it belongs to.



3.3 Gradient Boosting

The generalization of the original AdaBoost algorithimis known as Gradient Boosting requires solving the following objevtive function:

$$\sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \omega(f_t) \tag{9}$$

where $g_i = \partial_{\hat{y}^{(t-1}} l(y_i, \hat{y}^{(t-1)}), h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

According to [8] The proper loss function is a log-loss function given by:

$$L_m(\phi) = \sum_{i=1}^{N} log[1 + exp(-2\hat{y}_i(f_{m-1}(x) + \phi(x_i)))] \tag{10}$$

This optimization procedure is know as logitBoost and according to [8] it can be generalized to the multi-class setting. Freund et al. [5] provides a detailed framework to solve the multiclass problem. Both implementations for Random Forest and Gradient Boosting used on this paper were provided by [9].

3.4 The query algorithim

Algorithm 2 executes the task of sampling the pool of unlabelled data as illustrated in figure 3 and described in the algorithm 1. Given that this procedure is testing different cases, the query algorithm is a conditional that applies the formulations on table 2 and any case-specific task. The variable ϕ represents a utility function that expresses the uncertainty metric used (least confident, margin or entropy) and was mentioned before in subsection3.2.

For the general uncertainty sampling case, the algorithm will simply measure the uncertainty of all samples in the unlabelled pool. If density weighting was defined as an option, the uncertainty measurement is weighted based on similarity weights computed in algorithm 1 (lines 8 to 12). A model fitting step is included for query-by-boosting and query-by-bagging (lines 6 and 11 in algorithm 2).

4 Related work

4.1 Active learning

Settles and Craven [12] described several active learning strategies that can be implemented with low computational costs. The authors claim that the comparative effectiveness of these method has not been studied and point to limitations concerning the effect of noise on techniques based on uncertainty sampling and query-by-committee.

Roy and McCallum pointed these two methods can be limited in the presence of outliers. By outliers, they are referring to samples that the learner considers informative but that contains little information regarding the distribution of instances. Amongst the methods presented by Settles and Craven, Information Density is the one that tries to address the effect of the presence of outliers on the results of uncertainty sampling and query-by-committee techniques. The goal is to reduce the influence of uninformative samples by introducing a weighting function based on some similarity metric (eg. cosine similarity or Euclidean distance).

Abe and Mamitsuka proposed a practical way to implement a query-by-committee algorithm utilizing the ideas of bagging and boosting. The authors compare their implementation with the original query-by-committee algorithm but there isn't evidence to show that the proposed method performs better than other learning strategies such as uncertainty sampling.

Schein [10] defends an experimental design technique over empirical methods like uncertainty sampling or query-by-committee. In [10] the author formulates the A-optimality approach using logistic regression. The results were close to random picking, which can be fairly achieved by simple uncertainty metrics as shown in the next subsections. In his experimental results, however, Schein shows two cases in which uncertainty sampling performed better than A-optimality and two cases were it did not.

4.2 Gradient Boosting and Random Forest

A decision tree is an algorithm that can be used for classification tasks but is limited due to high prediction variance. Breiman [1] proposed the concept of bagging which consists in averaging predictions of several classifiers in order to reduce the prediction variance. It is an effective way to improve the performance of a classifier, and this concept has been used to enhance the power of decision trees.

However, it had been noticed that the correlation between the trees was also a factor that contributed to prediction error. One idea proposed by Breiman [2] to solve this, was decorrelating the decision trees by sampling, for each tree in the ensemble, the features that would be used in the training process. The fewer features that were randomly selected for training each tree in the ensemble, the better the results. The idea is that the introduction of additional noise can reduce the prediction error.

Freund and Schapire [4] proposed the algorithm called AdaBoost that later on was generalized by Breiman [3] to what is called Gradient Boosting. Gradient Boosting Tree is a particular case of gradient boosting in which decision tree stumps are used as the weak classifier. Generally speaking, Boosting is an additive modeling procedure where each weak learner will be trained on the residual errors of the previous classifiers. This procedure forms a prediction function that can assume complex shapes as the number of weak learners is increased.

Table 3: Average standardized accuracy gain across 8 real datasets: L_0: 5% of the Trainning set, L: 70% of the trainning set, and $d_L = round(\frac{(L-L_0)}{20})$: 20 equal samples, starting from 5 percent initial pool of labelled until 70 percent of the data is reached

method	**mean**	**std**
entropy-none-none	0.5221287106722681	0.4022554332616668
least_confident-none-none	0.579799763303847	0.37708362980659504
margin-none-none	0.5502602896196355	0.3626491227603235
entropy-euclidean-none	0.3167165071770323	0.33777368259008594
least_confident-euclidean-none	0.43792686623721055	0.29469467052008474
margin-euclidean-none	0.7125000000000004	0.4243754679704351

Table 4: Average real accuracy gain across 8 real datasets: L_0: 5% of the Trainning set, L: 70% of the trainning set, and $d_L = round(\frac{(L-L_0)}{20})$: 20 equal samples, starting from 5 percent initial pool of labelled until 70 percent of the data is reached

method	**mean**	**std**
entropy-none-none	0.34299531282263174	0.7709184629661522
least_confident-none-none	0.4399376613124248	0.8109318344088152
margin-none-none	0.44692105307985147	0.8073215876372373
entropy-euclidean-none	0.31685336450728363	0.7771600926967595
least_confident-euclidean-none	0.4177943322853162	0.7964679767759761
margin-euclidean-none	0.46458054253304654	0.7927593115998557

5 Comparsion

5.1 Uncertainty samppling

Each dataset responded to the active learning strategy according to its own characteristics. (see figures 4a and 4b). In order to choose which method performs best on average, the results on the scale of each dataset, which differ considerably, were standardized. The final metric gave a score from zero to one for each method on each dataset. Table 3 shows the average standardized accuracy gain and standard deviation for each metric. This is the product of 120 simulations corresponding to 6 cases (table 8 items 1 to 6), 8 datasets and 15 trials for each pair case dataset. The trials were necessary because the results varied depending on the split between the train and test data. Averaging 15 cases provided a better way to measure the expected gain in a more constant way.

The best performance was achieved using *margin* combined with euclidean density weighting. The worst performance was with entropy combined with density weighting. It was a systemic poor performance on the Iris dataset that drove entropy down the rank. It was expected that entropy could give poor results in some cases given that it has been shown that this metric can get stuck in specific regions causing the learning to deteriorate. Thus when density weighting is added to the equation, the metric becomes even more resistant to exploring different regions of the unlabelled pool, which yields in poor results.

Figures 9a and 9b in the appendix where generated using a **synthetic dataset** (see the appendix figures 9 and 8 . The density plots show the labeled set at a given iteration step. It is possible to visualize that in9b the algorithm stayed fixated on a central region while in 9a the algorithm expanded the search to outer regions. The difference in accuracy between both methods is also provided in the image. When the labelled set had 111 samples, 9a had a 74% accuracy and 9b had 64%. When the labeled set reached 211 samples, 9b had an accuracy even lower than what it was at previous steps. This toy exercise provides an analogy to explain why entropy with euclidean weights performed so poorly on the Iris dataset. Figures 6a and 6a show that the algorithm kept selecting weak samples, thus the accuracy remained at very low levels while random picking was improving the model faster.

Figures 5, 6 and 7 exemplifies the benefits of uncertainty sampling methods. The "barebone" metrics (uncertainty without weighting) had an average standardized accuracy gain of 54 and very little

Table 5: Gradient Boosting and Random Forest results

method	**mean**	**std**
boosting-euclidean-margin	0.5615807209855465	0.846499680342924
bagging-euclidean-margin	0.4579592071866214	0.5597376853473798

Table 6: Boosting and Bagging per dataset

data	boosting-euclidean-margin	bagging-euclidean-margin
breast_cancer	0.0048448981	-0.0454437248
crx	0.2612218773	0.0811343118
digits	0.3280781059	0.4029983113
iris	1.4562550893	1.3563632197
liver	-0.2015290044	0.1948462555
pima	-0.0969089124	-0.0173005747
tictac	1.9963089691	0.9938097367
wine	0.744374745	0.6972661219

variation across them. The standard deviation of the entropy bare case is higher most likely because of the problem mentioned before. Figures The real gains are depicted in Table 4.

5.2 Boosting and Bagging

Table 5 contains the mean and the standard error for the real accuracy gain on the same conditions as in table 4 but for the boosting and bagging cases. The standard deviation is high due to the contribution of the tictac dataset. The boxplot on figure 4b illustrates that the averages were either close to zero or significant gain was achieved.

Boosting and Bagging outperformed uncertainty sampling on the iris and wine dataset. However, when compared across all methods and datasets in a standardized way, it is expected that bosting or bagging will perform around 50% of the best methods. Table 7 provide a comparison of all methods across datasets. Margin with and without euclidean weights had the best performance while boosting, bagging and entropy with euclidean weights were the worst performers.

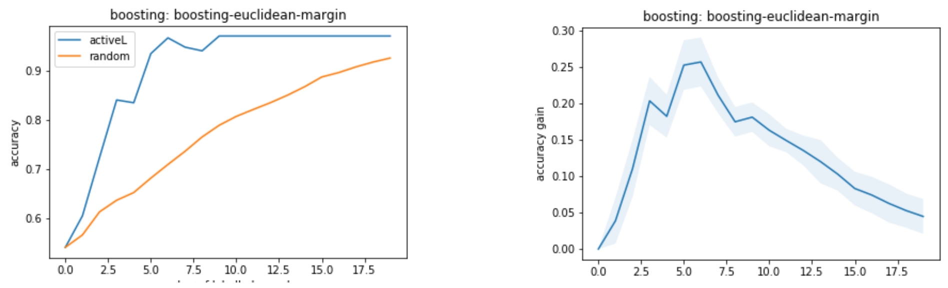


Figure 2: Results of boosting on the tic-tac dataset

6 Limitations

These results can not be used to make scientific generalizations about the efficacy of active learning methods. Here we only analyze empirical evidence on a limited number of datasets. In addition, comparing the ensemble methods with uncertainty sampling is not so straightforward. With the ensembles, the fitted model tends, or at least we expect so, to be better than the base model (here the logistic model). Thus when qbc is applied, the true gain in accuracy may correspond uniquely to the nature of the ensemble that will be predicting the committee's choices. Whereas with uncertainty sampling all predictions are done directly with the base classifier without additional help. That being

Table 7: Standardized accuracy gain accross all methods and dataset

case	standard performance
least_confident-none-none	0.7313935295605224
entropy-none-none	0.5961618324098878
entropy-euclidean-none	0.517255185909933
least_confident-euclidean-none	0.643322021085969
margin-none-none	0.7273159621362791
margin-euclidean-none	0.7649154410092884
boosting-euclidean-margin	0.5266668013637996
bagging-euclidean-margin	0.48587148836259053

said, the results here indicated that the performance of boosting and bagging was inferior to the other methods. But is important to recall that no hyperparameter optimization method was used. Thus, we can not assume that bagging and boosting here are better than uncertainty sampling.

7 Conclusion

Uncertainty sampling has shown to be an interesting approach to solve active learning without much computational effort. The results indicate margin is a safe metric to use and will provide a good result in almost all cases. Entropy may fail on specific cases such as Iris, locking the search to a specific region of the feature space. In general, density weighting had a positive effect thus it should be considered in further applications and studies.

Bagging and boosting are excellent methods to improve classification. However, for active learning, they require constant fine-tuning. The first observation is that since the size of the labeled set is continuously changing so is the model. Thus for active learning, one must adjust the hyperparameters of the ensemble routinely. These models are complex, and tunning have a considerable impact on performance. Especially in the case of boosting. That being said, in this case, we kept a unique set of parameters and the results were great for some dataset and poor for others. Using an ensemble without the proper care can be worst than a simplistic but effective uncertainty sampling technique.

The positive aspect of uncertainty sampling is that none of this model tunning is necessary. Uncertainty sampling works out of the box with the same classifier being used in the model.

There are two new approaches that could be studied in the future. First, rather than searching the version space by using a committee of hypothesis limited to unique criteria, one could build a committee that works with different uncertainty metrics on a voting system. Second, a backtest using different metrics to decide which method to pursue next can also have interesting results. Usually, the trend tells if the method will go downhill or if it will continue to perform well. On that note, a backtest on the known data using cross-validation could indicate which active learning strategy will perform better for the next step. Finally, a combination of methods at different stages of the process may enhance overall performance. For instance, there was an indication that the failure of the ensembles was related to either a poor initial set or to the fact that the sample size was big and few queries were made. If the firsts predictions are wrong, the classifier tend to perform poorly for a while until it starts improving again, which may be to late. For this case, an uncertainty sampling procedure at the early stages could provide a good basis for better results using the ensemble methods at a later stage

References

[1] Leo Breiman. Bias, variance, and arcing classifiers. 1996.

[2] Leo Breiman. Random forests. *UC Berkeley TR567*, 1999.

[3] Leo Breiman et al. Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849, 1998.

[4] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of online learning and an application to boosting. 1995. In *European Conference on Computational Learning Theory*.

[5] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2):337–407, 2000.

[6] M. Lichman. UCI machine learning repository, 2013.

[7] Naoki Abe Hiroshi Mamitsuka et al. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML'98)*, volume 1, 1998.

[8] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[9] 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 E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Andrew Ian Schein and Lyle H Ungar. *Active learning for logistic regression*. University of Pennsylvania, 2005.

[11] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[12] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, 1998.

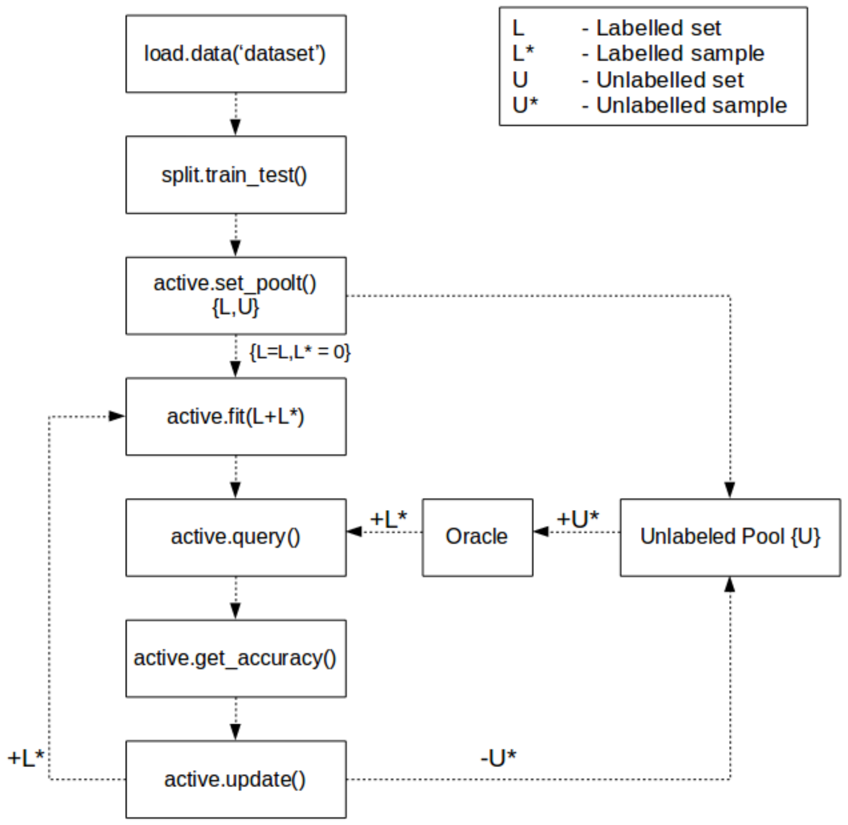


Figure 3: General flow of an active learning program [12]

Table 8: Cases evaluated on this work

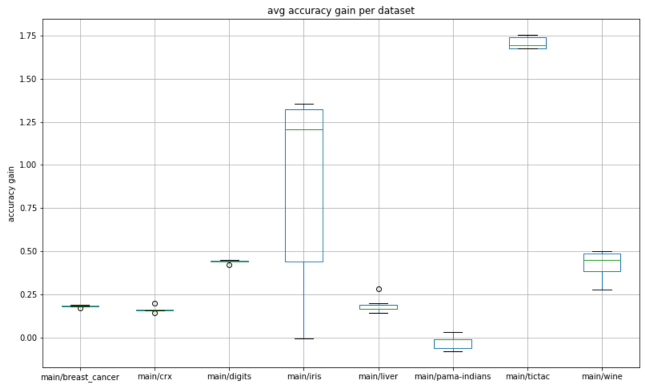
case	subcategory	density method	voting method
Entropy-none-none	entropy	none	none
Least confident-none-none	least confident	none	none
Margin-none-none	margin	none	none
Entropy-euclidean-none	entropy	euclidean	none
Least confident-euclidean-none	least confident	euclidean	none
Margin-euclidean-none	margin	euclidean	none
boosting-euclidean-none	boosting	euclidean	margin
Bagging-euclidean-none	bagging	euclidean	margin

Algorithm 1 Test activel learning methods on a dataset

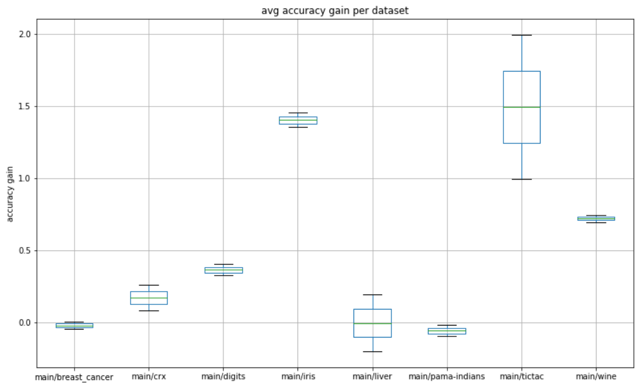
```
 1: procedure TESTCASES($X, y$)
 2:     load dataset from table 1
 3:     split data into 70% train and 30% test
 4:     size $\leftarrow$ size of 70% of train
 5:     for each case in table 8  do
 6:         cat $\leftarrow$subcategory ; dm $\leftarrow$ density method; vote $\leftarrow$ voting method
 7:         $L \leftarrow$ 5% of train set and $U \leftarrow$ size $- L$ = SETPOOL(train)
 8:         if dm = 'euclidean' then
 9:             $w \leftarrow \sum_c euclideanDist(U, U)$/size(U)
10:         else if dm = 'none' then
11:             $w \leftarrow ones(size(U))$
12:         end if
13:         for $i \leftarrow$ 0 to size  do                                   ▷ initiate fitting process
14:             LogisticClassifier.fit(L)                                      ▷ train logistic classifier
15:             sample L* $\leftarrow$ QUERY(($cat, w, voteM$),$L$,$U$)              ▷ active learning strategy
16:             accuracy[i]=getAccuracy($L, test$)
17:             $L \leftarrow L + L*$
18:             $U \leftarrow U - L*$                         ▷ update pools of labelled and unlabelled data
19:             w $\leftarrow$ remove weights relative to L*
20:         end for
21:         caseResults[case]=accuracy
22:     end for
23: end procedure
```

Algorithm 2 query($cat,w,voteM,L,U$)

```
 1: if cat = 'least confident', or 'margin',or 'entropy then           ▷ see table 2 for definitions
 2:     $P_\theta(\hat{y}|x)$ = LogisticClassifier.getProb(U)              ▷ classifier trained on TestCases
 3:     $vote \leftarrow w. * \phi(P_\theta(\hat{y}|x))$                  ▷ $\phi$ is a utility function (uncertainty metric)
 4: return $argmax_x(vote)$
 5: else if cat = 'Boosting' then
 6:     GradientBoosting(max-depth=1,n-estimators=1000,subsample=0.3,warm-
    start=True,learning-rate=0.1) ).fit($L$);
 7:     $P_\theta(\hat{y}|x)$ = GradientBoosting.getProb($U$)
 8:     $vote \leftarrow w. * \phi(P_\theta(\hat{y}|x))$
 9: return $argmax_x(vote)$
10: else if cat = 'Bagging' then
11:     RandomForest(n-estimators=500,max-depth=3).fit($L$)
12:     $P_\theta(\hat{y}|x)$ = RandomForest.getProb($U$)
13:     $vote \leftarrow w. * \phi(P_\theta(\hat{y}|x))$
14: return $argmax_x(vote)$
15: else if cat = 'Random' then
16: return random sample
17: end if
```

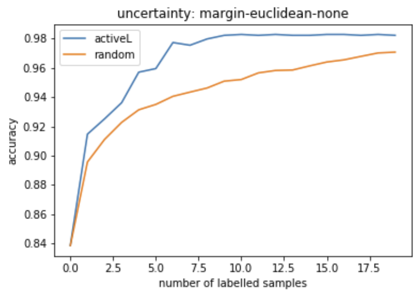


(a) Accuracy gain per dataset with uncertainty sampling

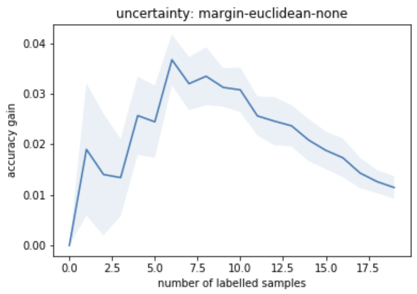


(b) Accuracy gain per dataset with boosting and bagging

Figure 4: Though there is a slight difference in magnitude of the average gain, the datasates responded on a similar to active learning. Tic- tac responded the best. Iris had a high average, except for one entropy case mentioned in 8.

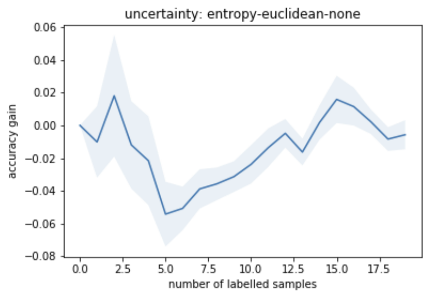


(a) digits:Test Accuracy

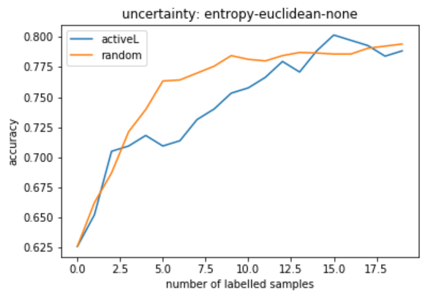


(b) digits:Accuracy gain

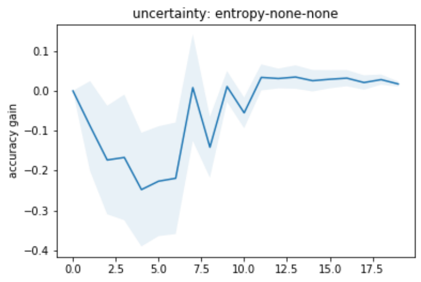
Figure 5: (5a) Digits dataset: active learning implemented with uncertainty sampling with margin and euclidean density weighting. (5b) Average accuracy gain and standard deviation.



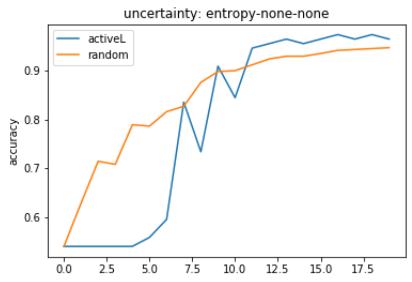
(a) pima:Test Accuracy



(b) pima:Accuracy gain

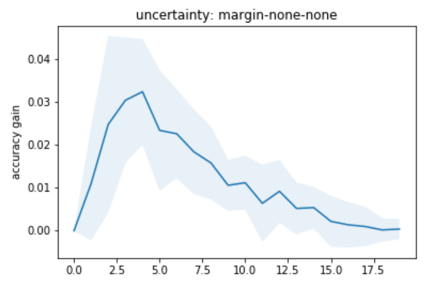


(c) iris:Test Accuracy

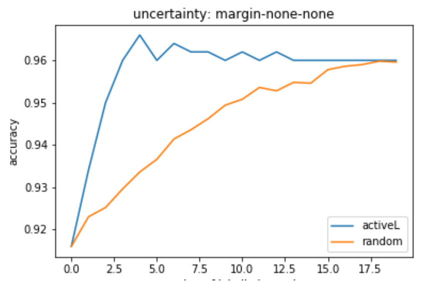


(d) iris:Accuracy gain

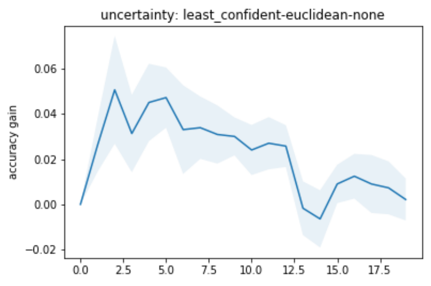
Figure 6: (5a) Pima dataset: active learning implemented with uncertainty sampling with entropy and euclidean density weighting. (5b) Average accuracy gain and standard deviation.6c Iris dataset: active learning implemented with uncertainty sampling with entropy. 6d Accuracy "stuck" during the first iterations"



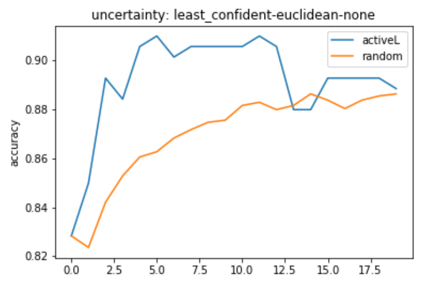
(a) breast cancer:Test Accuracy



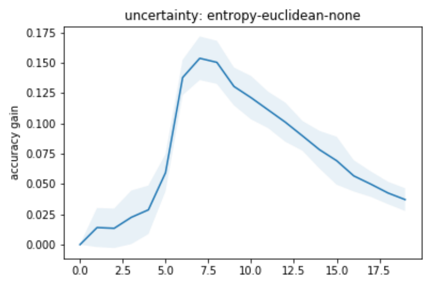
(b) breast cancer:Accuracy gain



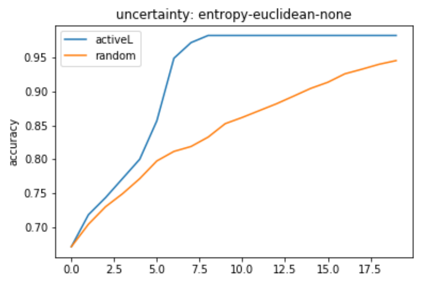
(c) crx:Test Accuracy



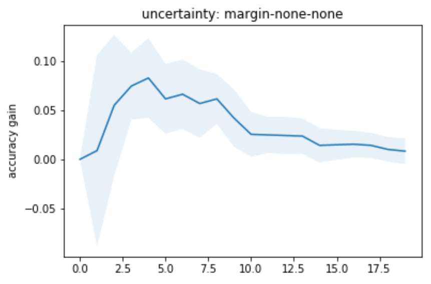
(d) crx:Accuracy gain



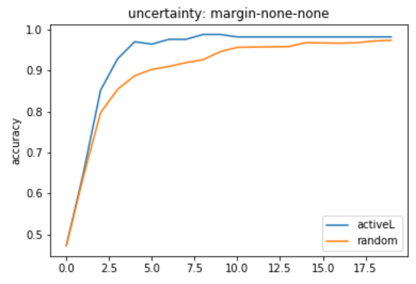
(e) tic-tac:Test Accuracy



(f) tic-tac:Accuracy gain

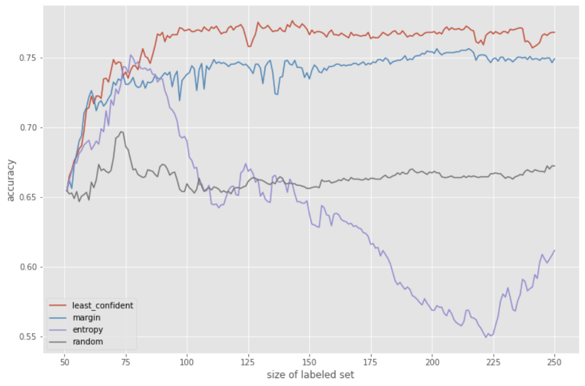


(g) wine:Test Accuracy

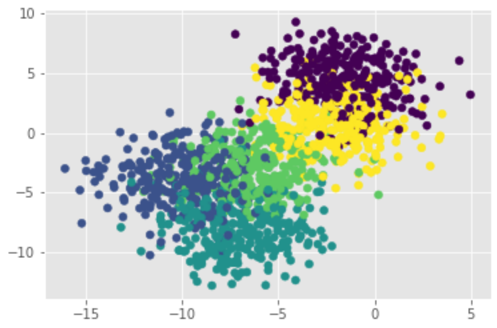


(h) wine:Accuracy gain

Figure 7: Sample results, on the left is a curve that shows the difference between the active learning procedure and the average of 100 trials of random picking. The boundaries of the curve are represented given the standard deviation of the random samples. On the right, typical accuracy curve.

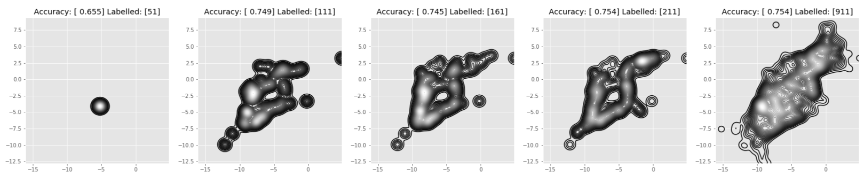


(a) Accuracy test

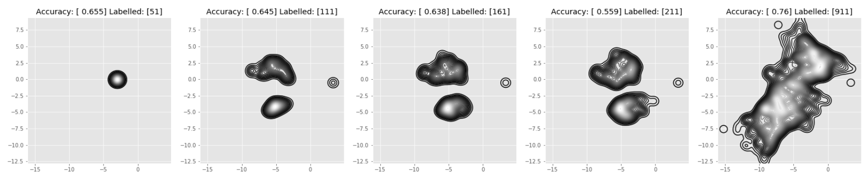


(b) 2d plot of a synthetic dataset

Figure 8: 8a Accuracy results for three uncertainty metrics compared to random sampling. 8b 2d plot of the synthetic dataset. Normally distributed clusters, 5 classes, 2 dimensions and standard deviation of 2 The sample generator used for the example on figure 9 is provided by *sklearn* [9]. It generates normally distributed data in clusters while giving control on the center and standard deviation of the normal distributions.



(a) uncertainty sampling with margin



(b) uncertainty sampling with entropy

Figure 9: Results on a synthetic two dimensional dataset. 9a Density plot of the labelled set at different stages of active learning with ***margin*** on a synthetic dataset.9b Density plot of the labelled at different stages of active learning with ***entropy*** on a synthetic dataset.