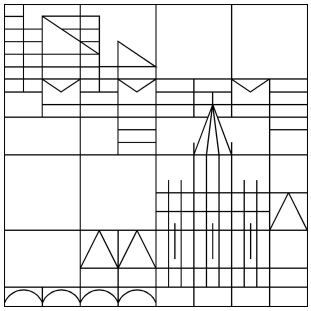


Universität Konstanz

An Adaptive Multi Objective Selection Strategy for Active Learning

Nicolas Cebron
Michael R. Berthold

Konstanzer Schriften in Mathematik und Informatik
Nr. 235, August 2007

ISSN 1430-3558

Konstanzer Online-Publikations-System (KOPS)
URL: http://www.ub.uni-konstanz.de/kops/volltexte/2007/3440/
URN: http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-34406

Universität Konstanz
Fach D 188, 78457 Konstanz, Germany
E-Mail: preprints@informatik.uni-konstanz.de
WWW: http://www.informatik.uni-konstanz.de/Schriften/

An Adaptive Multi Objective Selection Strategy for Active Learning

Technical Report

Nicolas Cebron and Michael R. Berthold
Nycomed Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
University of Konstanz
Box M 712, 78457 Konstanz, Germany

{nicolas.cebron, michael.berthold}@uni-konstanz.de

August 13, 2007

Classifying large datasets without any a-priori information poses a problem in numerous tasks. Especially in industrial environments, we often encounter diverse measurement devices and sensors that produce huge amounts of data, but we still rely on a human expert to help give the data a meaningful interpretation. As the amount of data that must be manually classified plays a critical role, we need to reduce the number of learning episodes involving human interactions as much as possible. In addition for real world applications it is fundamental to converge in a stable manner to a solution that is close to the optimal solution. We present a new self-controlled exploration/exploitation strategy to select data points to be labeled by a domain expert where the potential of each data point is computed based on a combination of its representativeness and the uncertainty of the classifier. A new Prototype Based Active Learning (PBAC) algorithm for classification is introduced. We compare the results to our previous approach and Active Learning with Support Vector Machines on several artificial and benchmark datasets.

1 Introduction

In a previous article [3], we have introduced the problem of mining cell assay images. To study the effects of drug candidates or more generally to obtain observations about how a biological entity reacts when it is exposed to a chemical compound, an array of cells is screened with a camera. This process is called High-Content-Screening and allows to collect a large amount of experimental data. Especially the development of new screening devices - with specialized robots creating the cell assay and automatically taking measurements and pictures - makes it possible to obtain hundreds of thousands of cell images in only a few days. In this work, we focus on classifying this large unlabeled dataset with the help of a biological expert who is able to provide us with class labels for few, selected examples.

In this work we extend earlier work [3] where we separately explore the data space by clustering and later adapt the learned policy with Learning Vector Quantization (LVQ). This approach was motivated by the idea that the dataset needs to be explored first to generate a coarse model and then the model can be adapted to further fine-tune the classification accuracy. However, this classification scheme had to be initialized with a fixed number of clusters, which influenced how much focus was laid on the exploration part. Each cluster was then split up into sub clusters in order to verify the current cluster classification hypothesis, which resulted in "unnecessary" queries, if all sub clusters already had the same class label.

Our new approach takes into account the density of the feature space and the uncertainty of the classifier. In contrast to our earlier approach, both criteria are combined to form one single criterion for the selection of unlabeled data. During each classification iteration the influence of the exploration part decreases whereas the influence of exploitation increases in a natural way. This allows for a smooth transition between these two opposing phases. In contrast to our old approach, the density is taken into account during the entire time and not only during the exploration phase. Furthermore, the use

of nearest prototype based classification (instead of labeling cluster prototypes and LVQ learning) enhances the stability of the classifier and makes it computationally feasible and robust even for large datasets. This completely changes the way that the data is classified, and enhances the classification accuracy drastically as we will demonstrate in the experimental section.

In Section 2, we will shortly revise the concept of Active Learning which leads us to a selection strategy in Section 3. In Section 4 we introduce the prototype based classification scheme with some examples. Results on artificial and different benchmark datasets are presented in Section 5 before drawing conclusions in Section 6.

2 Active Learning

We now describe and formalize the central part of this work, the active learning. Consider a set X of n feature vectors $\{x_1, x_2, \ldots, x_n\}$ lying in $\Re^d$. The training set consists of a large set of unlabeled examples $U \subseteq X$ and a set of labeled examples L, which contains examples from X and their corresponding labels from a set of possible class labels Y: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \in X \times Y$.

In many classification tasks it is common that a large pool of unlabeled examples U is available whereas the cost of generating a label for an example is high. The concept of active learning [6] tackles this problem by enabling a learner to pose specific queries, chosen from an unlabeled dataset. In this setting, we assume that we have access to a noiseless oracle that is able to predict the class label of a sample. We can describe an active learner by its underlying classifier and a query function. The classifier f is trained on L and sometimes also on U. The query function q makes a decision based on the current model f which examples from U should be chosen for labeling. In pool-based active learning, a new classifier f' is generated after a fixed number of queries.

Many active learning strategies for different kinds of algorithms exist. In [6], a selective sampling is performed in areas where the most general and the most specific hypotheses disagree. The hypotheses were implemented using feed-forward neural networks with backpropagation.

Active Learning with Support Vector Machines (SVM) has also become very popular. The expensive learning process for the SVM can be reduced by querying examples with a certain strategy. In [12], the query function chooses the next unlabeled data point closest to the decision hyperplane in the kernel induced space. In a more recent approach [15], the unlabeled data located in the margin of the SVM is clustered using the k-means algorithm to choose representative samples to query next. This approach is more similar to our work as it is attempting to take into account the distribution of the input data. However, this approach is still based on a discriminative model and does not take the data distribution into account as good as a generative model. Support Vector Machines with active learning have been widely used for image retrieval problems [9] [13] or in the drug discovery process [14].

In [8], k-means clustering has been used to select an initial training set for active learning in text classification. However the problem of choosing the correct number of clusters in order to have a representative from each available class is not addressed. Active learning in hierarchical pairwise data clustering has been proposed in [2]. The active

data selection aims to reduce the number of samples needed in pairwise data clustering to reduce the computational load. This approach focuses on clustering and does not incorporate class labels. In [10], an EM approach is used to integrate the information from unlabeled data. In that work, the active learning is based on the Query-by-Committee algorithm, normally used in stream-based learning. This involves having a set of classifiers to measure the disagreement between committee classifiers. Active Learning for 2-class problems and Clustering have been combined in [11]. The dataset is preclustered with the K-medoid algorithm before additional samples enhance the classification accuracy. The clustering is adapted during the learning epochs. A combination of density estimation and classifier uncertainty is used to select new examples. However, the density of each data point is estimated only with the current set of clusters.

Our approach differs from the others in the way that we combine the *potential* of each data point (which is based on density estimates on the unlabeled data) and the *classifier uncertainty* (based on the labeled data) in one single criterion which we call *uncertainty distribution* (see Figure 1). Instead of preclustering the dataset, we use the

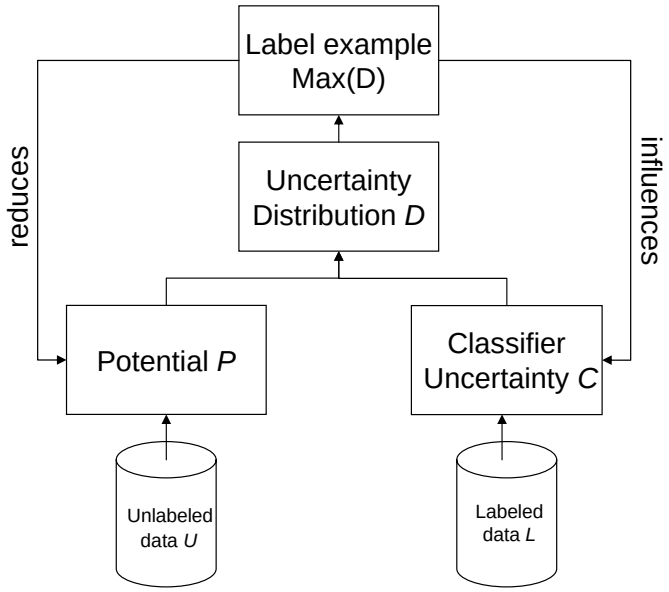


Figure 1: Interaction of unlabeled and labeled data, potential and classifier uncertainty.

uncertainty distribution to choose examples for a prototype based classification. As potentials of selected points and their neighboring points are reduced, a smooth transition between exploration and exploitation takes place since the labeled points will gradually reduce the potentials over the entire feature space. We will work out the details of this estimation technique and how this transition takes place in the next sections.

3 Selection Strategy

We assume that the data space X is normalized in the interval $[0, 1]$. In our special setting, we assume that we do not have any labeled instances at the beginning. Therefore

we focus on two interleaving aspects: exploration (finding representative samples in the dataset that are useful to label) and exploitation (adapting the classification boundaries).

In [5], having an input distribution $P(x)$, it is proposed to minimize the expected error:

$$\int_x E_T \left[(\widehat{y}(x;D) - y(x))^2 | x \right] P(x) dx \tag{1}$$

where $E_T[.]$ denotes the expectation over training sets D, $\widehat{y}(x;D)$ is the learner's output and $y(x)$ is the true label of x. Rather than trying to minimize the value of Equation 1 by minimizing the learner's variance (as proposed in [5]), we try to select examples with the biggest contribution to the current error of the classifier, similar to the approach in [11]. Equation 1 shows that we need to weight the uncertainty of the classifier with the input data distribution $P(x)$. In the next sections we will introduce two measures to estimate - for each data point - its density and the uncertainty of the classifier.

3.1 Exploration

At the beginning of model learning we assume that we do not have any labeled examples $|L| = 0$, so we need a strategy to pick examples to label based on the unlabeled samples from U. From the exploration point of view we want to explore unknown regions and find possible classes in a dataset. One criterion for data selection is the representativeness of a data point. Rare or borderline cases that do not occur very often are not interesting for a classification[1]. In order to estimate the representativeness of a data point we compute the *potential* P of each data point x_i, according to the subtractive clustering approach from [4] as:

$$P(x_i) = \sum_{j=1}^{n} e^{-\alpha d(x_i, x_j)^2}, \ \alpha = \frac{4}{r_a^2} \tag{2}$$

where r_a is a positive constant defining a neighborhood. $d()$ is a distance measure, usually the euclidean distance is used. All points x_j that lie within this neighborhood have a large influence on the potential of data point x_i. Unlike the original algorithm we do not need to calculate the total similarity for all data points. To reduce the computational load we only compute the bounded total potential for the data within the radius r_a, as the data points outside the boundary will have little effect on the total potential. An efficient way to find the nearest neighbors within a given distance is to use KD-Trees [1] as underlying data structure.

After the potential of each data point has been computed the data point x_k^* with the highest potential $P(x_k^*)$ is selected. In order to avoid that another data point near the chosen point is selected in the next iteration, the potential of this point and the surrounding points in the neighborhood are reduced:

$$P(x_i) \Leftarrow P(x_i) - P(x_k^*) e^{-\beta d(x_i, x_k^*)^2}, \ \beta = \frac{4}{r_b^2} \tag{3}$$

[1] In a real world application one could show those examples as potentially interesting outliers to the user but for the construction of a global model they do not carry much information.

where r_b is a positive constant defining the neighborhood in which the potentials are reduced. In the work of [4], r_b has been set to $1.25r_a$. Note that from the algorithmic point of view any density estimation technique can be used. However, the reduction of the potentials plays an important role as it reduces the overall exploration potential in each iteration and thus gradually allows a transition to the exploitation step, which will be described in the next section.

3.2 Exploitation

The idea of exploitation in a classification task is to take into account information of the current classifier model in order to find new points that help to enhance the classification. We use the weighted k-nearest neighbor classifier based only on the examples that have been labeled so far. The class label $\widehat{f}(x_q)$ is determined by taking the class label of the k nearest prototypes with the largest weight.

$$\widehat{f}(x_q) = \max_{v \in Y} \sum_{c=1}^{k} w_c \delta(v, f(x_q)) \;\; \delta(a,b) = 1 \text{ if } a = b, 0 \text{ otherwise} \tag{4}$$

where

$$w_c = \frac{1}{d(x_q - p_c)^2} \tag{5}$$

Regarding all currently labeled examples as prototypes, we can calculate for all other data points their class probabilities. Having assigned a class label to each prototype, we can classify all data points by assigning them the label of the prototype with the highest probability. The class probabilities also provide us with information about the uncertainty of the classifier. We focus on data points that have an almost equal probability to different classes. These points can be found easily with an one-pass scan through the class weights. We compute the entropy of the histogram of the class weights and denote it as *classifier uncertainty* C:

$$C(x_i) = H(w_1, ..., w_k) = -\sum_{c=1}^{k} w_c \log_2(w_c) \tag{6}$$

The resulting entropy value must be normalized with the number of classes in the current iteration $H_{max} = \log_2 |Y|$. Intuitively, a very sharply peaked distribution has a very low entropy, whereas a distribution that is spread out over many bins has a very high entropy. Therefore, we take the entropy as an uncertainty measurement, reversely related to the voting confidence of the Nearest-Neighbor classifier.

3.3 Combination

Based on the *potentials* P that we compute on the unlabeled data and the *classification uncertainty* C which is based on the labeled data, we form a new data selection criterion which is called *Uncertainty Distribution* D.

$$D(x_i) = (1 - \epsilon)P(x_i) \cdot \epsilon C(x_i) \tag{7}$$

where $\epsilon \in [0, 1]$ controls the influence of the exploitation in the first iterations. As the potentials are reduced in each step, the classifier uncertainty becomes more and more important. However, the remaining potential on the data point still prevents unrepresentative samples from being chosen. This helps to prevent selection of rare or borderline cases. The reduction of potentials also provides a useful measure to induce a stopping criterion. If the total sum of all potentials drops under a predefined value t, we can stop the classification process.

4 Prototype Based Active Classification

Based on the idea developed in Section 3, we outline our new prototype based classification scheme (PBAC) in Algorithm 1. The potentials of all data points can be precomputed offline in order to speed up the interactive classification process. We start by calculating the uncertainty distribution for all data points and selecting the point with the highest estimate as the first prototype. In every iteration, the potentials for the chosen data point and its neighbor points are reduced, which also causes an overall reduction of the global potential. Each selected data point gets labeled by the expert and is added as a prototype. Based on the current set of prototypes, all data points are classified with the weighted k-nearest neighbor algorithm.

Algorithm 1 Prototype Based Active Classification

Require: Threshold t

GlobalPotential $\Leftarrow 0$
for all $x_i \in U$ **do**
  Compute the potential $P(x_i)$ according to Equation 2.
  GlobalPotential $\Leftarrow$ GlobalPotential + $P(x_i)$
end for
while GlobalPotential $>$ t **do**
  for all $x_i \in U$ **do**
    Compute the classifier uncertainty $C(x_i)$ according to Equation 6.
    Compute the uncertainty distribution $D(x_i)$ according to Equation 7.
  end for
  Select the data point x_t with the highest uncertainty distribution.
  Obtain a label l for x_t.
  Create a new prototype with values x_t and class label l.
  Classify the dataset with the current set of prototypes.
  Reduce the potentials according to Equation 3.
end while

We demonstrate the mode of operation of the PBAC algorithm on a 1-dimensional artificial example with three classes (A,B and C) in Figure 5. In Figure 2a, the potentials for all data points (plotted on the x-axis) indicate that there are three distributions that need to be explored. The exploitation factor ϵ has been set to 0.5, therefore the maximal uncertainty distribution value at the beginning is 0.5. In the first step, a prototype of class B is created and the potentials around it are reduced (see Figure 2b). After

choosing the next point with class label A with a high potential in step 2 (Figure 2c), the uncertainty distribution does no longer depend on the potential only but also on the classifier uncertainty (dotted line). The decisive factor for choosing the next prototype of class C is primarily due to a high potential, but is also a little bit amplified by the high classifier uncertainty for the data points at the right side. In step 3 (Figure 2d), one can observe the combination of potentials and classifier uncertainty. The fourth prototype of class B is chosen by a combination of classifier uncertainty between class A and B and the remaining potential between A and B. It can be clearly seen how the dataset is explored in the first iterations, finding all three possible classes. In subsequent steps, the focus is laid on the classification boundaries between class A and B and class B and C.

5 Results

We have chosen two datasets from the UCI Machine Learning Repository [7] to demonstrate the effectiveness of our active learning algorithm. In Section 5.1, we first compare the proposed algorithm (PBAC) against a margin-based Active SVM [12] on the satimage dataset [7]. We then compare the algorithm against our own Active LVQ classification scheme [3] and an Active Representative SVM on the same dataset. Analysis on the pendigits dataset [7] in Section 5.2 compares our algorithm against our old Active LVQ Algorithm and the Active Representative SVM.

5.1 Satimage data

For comparison, we used the satimage data from the UCI Machine Learning Repository [7] which consists of 4435 training instances which describe satellite image data by their multispectral values. All training instances are first assumed to be unlabeled. The performance was measured on a separate test data set containing 2000 instances. We compare our PBAC algorithm (bold line) against several runs of an Active Margin SVM [12] that is initialized with random examples and queries new samples at the decision hyperplane, see Figure 3. We set the search radius and the exploitation factor for our PBAC algorithm to 0.1. The SVM was trained with a linear kernel and overlapping penalty 2.0. One can see the stability of our PBAC algorithm against random selection in the SVM for the first queries. The accuracy reaches approx. 83.2% after only 100 queries, which means that we reach 93% of the possible classification accuracy with 2.26% of the available training data. The random selection SVM technique does not reach the same performance and is, especially during earlier epochs, severely unstable. Performance can drop easily 7% during subsequent queries. PBAC's performance increases almost monotonically.

Since our density estimation has such drastic effects on the stability and performance of the active learning scheme, we tried to provide the SVM with the possibility to choose more representative examples and give it a stable initialization. We implemented a combination of two techniques: First, the dataset is clustered to find representative examples for the initialization of the SVM, similar to [8] and then examples inside the SVM's margin are clustered to find representative examples according to the approach in [15]. We call this procedure Active Representative SVM.

We compare our PBAC algorithm against our own former Active LVQ classification scheme [3] and the Active Representative SVM. The PBAC algorithm and the Active Representative SVM have been initialized with the same parameters as mentioned above. The Active LVQ algorithm has been trained with an iteration size of 5 and a learning rate of 0.1. As a frame for the different active learning schemes, we plot the worst-case classification with randomly chosen prototypes (with variance) and the performance of a SVM trained on all 4435 examples with 85.1% and a weighted KNN classifier on all examples with an accuracy of 89.4%. The number of queries used on the x-axis versus the classification accuracy on the test data set on the y-axis for the first 250 queries is shown in Figure 4. It can be observed that random selection performs very unstable in the first iterations and has a large variance. Choosing representative examples for all different active classification schemes proves to be highly beneficial in the initialization phase. The PBAC algorithm outperforms both our own Active LVQ classification scheme as well as the Active Representative SVM approach with respect to accuracy. After 250 queries the PBAC algorithm has reached an accuracy of 85.3%, the Active Representative SVM 79.7% and the Active LVQ 74.1%.

5.2 Pendigits data

The Pendigits dataset from the UCI Machine Learning Repository [7] consists of (x, y) coordinates of hand-written digits. Each digit is represented as a vector in a 16 dimensional space. We have used the training dataset with 7494 examples for training the classifier and a separate test dataset containing 3498 examples for measuring the classification accuracy. Figure 5 shows the performance of the different classification schemes. We have used again the Active Representative SVM algorithm from the previous section and the Active LVQ algorithm for comparisons. A weighted KNN classifier initialized with all 7494 examples from the training set reaches an accuracy of 97.7%, a SVM classifier with a quadratic kernel trained on all examples 97.9%. As a worst-case classifier, we plot the mean performance of a classifier with randomly chosen prototypes and its variance. Our PBAC algorithm has been initialized with a search radius of 0.1 and an exploitation factor ϵ of 0.3. The SVM was trained with a quadratic kernel, bias 2.0 and overlapping penalty 2.0. The Active LVQ algorithm has been initialized with an iteration size of 5 and a learning rate of 0.1. After having labels for just approx. 1% of the available data, the PBAC algorithm reaches an accuracy of 90.6%, the Active Representative SVM 90.4% and the Active LVQ 71.5%. Our old Active LVQ algorithm performs stable at the beginning but performs worse than all other learning schemes in subsequent learning iterations. The PBAC algorithm performs stable at the beginning and equal to a Active Representative SVM with quadratic kernel in the following iterations. For clarity we omitted the plot of the performance of a SVM with a linear kernel which is worse, reaching an accuracy of 80.1% after 100 iterations.

5.3 Prototype based classification against SVM classification

One can not expect that our PBAC algorithm always performs better than a SVM with representative selection. As the performance of the introduced Active Representative SVM seems to be comparable to our PBAC algorithm, one might ask why we are not

using this technique instead. There are several reasons to favor a prototype based approach as presented in this paper: First of all, the performance of a support vector machine depends heavily on the kernel which is used. Finding a good kernel requires either labeled training data or a multitude of experimental evaluation, which will be impossible to achieve in real world applications where no labeled examples are available initially. Furthermore, SVM's are binary classifiers with computational complexity $O(n^2)$. In multi class problems, a SVM has to be trained for each class which requires that labels are queried for each SVM which can increase the number of total queries and the training time. And finally, prototypes provide the possibility to communicate the learned concept to a user in a way that can be grasped easily as opposed to support vectors, which tend to be hard to interpret.

6 Conclusions

In this article we have addressed the problem of classifying a large unlabeled cell assay image dataset with the help of a human expert in as few labeling steps as possible. We have proposed a new prototype based active learning scheme in which a new, labeled prototype is added in each learning iteration to fine-tune the classification of the dataset. Based on this growing pool of labeled prototypes, class probabilities are calculated for all data points in order to calculate estimates for their classification uncertainty. Together with a density estimation technique, a new criterion for prototype selection has been developed. Results revealed that this new approach enhances the classification accuracy significantly and is more stable - especially in subsequent iterations - compared to our old approach. The classification accuracy and stability of the PBAC algorithm proved to be better than classic Active Learning with Support Vector Machines with random initialization and closest-to-boundary selection. Choosing representative samples in the initialization can improve the performance of the SVM as well as choosing representative examples in the margin of the classifier. However, these techniques do not have the same advantages as our selection technique: we select representative samples first and focus on examples at the classification boundary when it becomes necessary in an automatic fashion. On several benchmark datasets we demonstrated stable performance for our algorithm, which reaches levels of accuracy close to the final one after only few iterations. This plays a crucial role in this application and is essential for user acceptance of such an interactive learning system.

Acknowledgement

This work was supported by the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces".

References

[1] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[2] J. Buhmann and T. Zoller. Active learning for hierarchical pairwise data clustering. *icpr*, 02:2186, 2000.

[3] Nicolas Cebron and Michael R. Berthold. Adaptive active classification of cell assay images. In *Knowledge Discovery in Databases: PKDD 2006 (PKDD/ECML)*, volume 4213, pages 79–90. Springer Berlin / Heidelberg, 2006.

[4] Stephen L. Chin. An efficient method for extracting fuzzy classification rules from high dimensional data. In *Journal of Advanced Computational Intelligence and Intelligent Informatics*, volume 1, pages 31–36. Fuji Technology Press Ltd, Tokyo, 1997.

[5] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Advances in Neural Information Processing Systems*, 7:705–712, 1995.

[6] David A. Cohn, Les Atlas, and Richard E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[7] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[8] Jaeho Kang, Kwang Ryel Ryu, and Hyuk-Chul Kwon. Using cluster-based sampling to select initial training set for active learning in text classification. In *Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 384–388. Springer Berlin / Heidelberg, 2004.

[9] T. Luo, K. Kramer, D.B. Goldgof, L.O. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. *Journal of Machine Learning Research*, pages 589–613, 2005.

[10] Andrew Kachites McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *Proc. 15th International Conf. on Machine Learning*, pages 350–358. Morgan Kaufmann, San Francisco, CA, 1998.

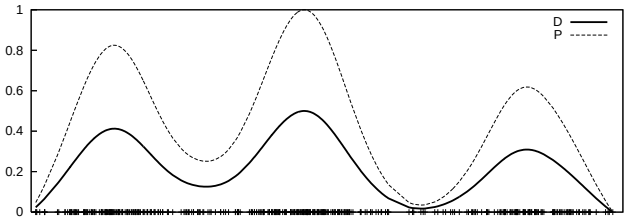
[11] H.T. Nguyen and A. Smeulders. Active learning using pre-clustering. *ICML*, 2004.

[12] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. *ICMLProceedings, 17th International Conference on Machine Learning*, pages 839–846, 2000.

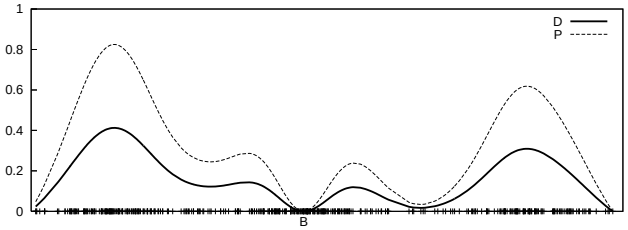
[13] L. Wang, K. L. Chan, and Z. h. Zhang. Bootstrapping svm active learning by incorporating unlabelled images for image retrieval. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:629–634, 2003.

[14] M. K. Warmuth, G. Raetsch, M. Mathieson, J. Liao, and C. Lemmen. Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, pages 667–673, 2003.

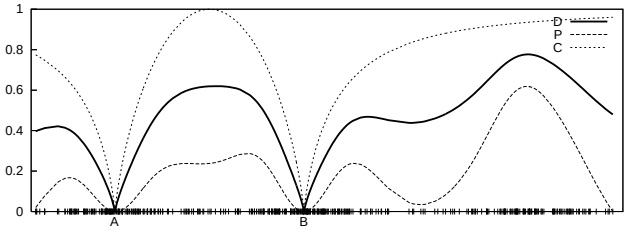
[15] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *ECIR 2003*, volume 2633, pages 393–407. Springer Berlin / Heidelberg, 2004.



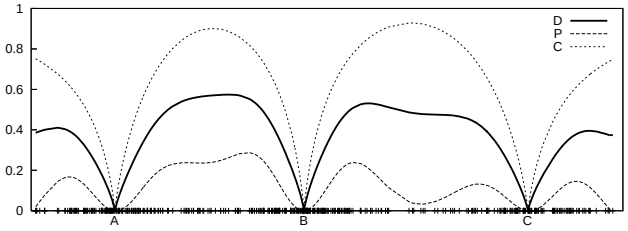
(a) Step 0



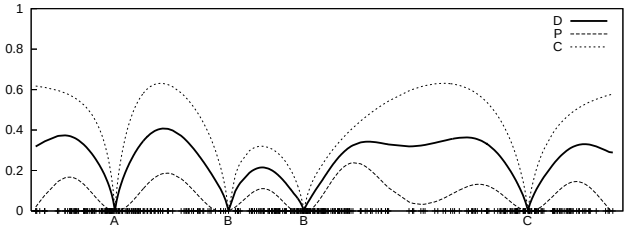
(b) Step 1



(c) Step 2



(d) Step 3



(e) Step 4

Figure 2: Characteristics of potential P, classifier uncertainty C and the resulting uncertainty distribution D in consecutive steps.

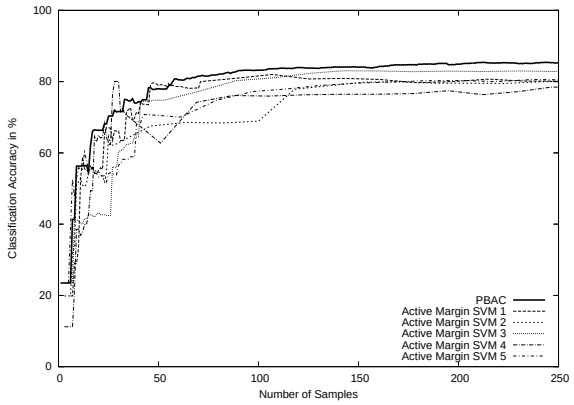


Figure 3: PBAC algorithm vs. Active SVM on the satimage dataset. The plot shows the number of requested labels on the x-axis versus the classification accuracy on the y-axis.

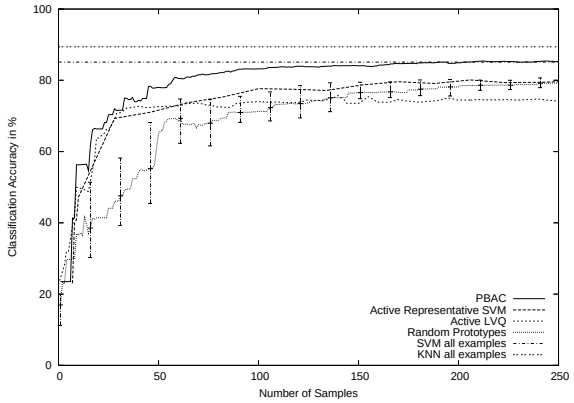


Figure 4: Performance of different classification schemes on the satimage dataset

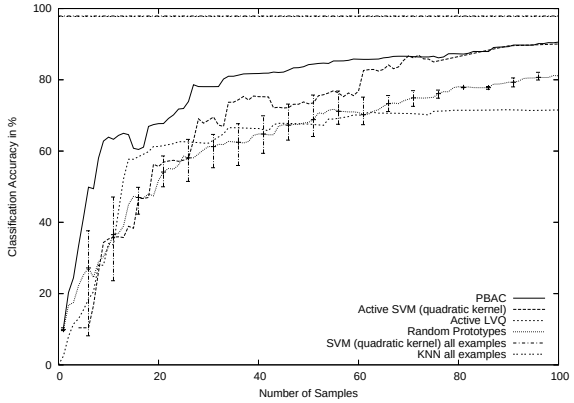


Figure 5: Performance of different classification schemes on the Pendigits dataset