Data Min Knowl Disc (2009) 18:283–299
DOI 10.1007/s10618-008-0115-0

Active learning for object classification: from exploration to exploitation

Nicolas Cebron · Michael R. Berthold

Received: 18 August 2007 / Accepted: 4 July 2008 / Published online: 27 July 2008

Abstract 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 other active learning approaches on several benchmark datasets.

Keywords Active learning · Data mining · Subtractive clustering · Exploration · Exploitation · Prototype classification

1 Introduction

In a previous article (Cebron and Berthold 2006), 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

Responsible editor: Pierre Baldi.

N. Cebron (✉) · M. R. Berthold
Department of Computer and Information Science, University of Konstanz, Konstanz, Germany
e-mail: nicolas.cebron@uni-konstanz.de

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 such a large unlabelled datasets with the help of a biological expert who is able to provide us with class labels for few, selected examples.

Similar use cases can be found in the field of text, music and image mining (McCallum and Nigam 1998; Mandel et al. 2006; Wang et al. 2003), where a large library of documents, music or images are available and we want to categorize this library with the help of the user. In these situations, the concept of active learning is applied, where the learning algorithm has control over which parts of the input domain it receives information about from the user.

In this work, we further develop the earlier work of Cebron and Berthold (2006) 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 datasets 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 unlabelled 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 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 Sect. 2, we will shortly revise the concept of active learning which leads us to a selection strategy in Sect. 3. In Sect. 4 we introduce the prototype based classification scheme (PBAC) with some examples. Results on different benchmark datasets are presented in Sect. 5 before drawing conclusions in Sect. 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 $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ lying in $\Re^d$. The training set consists of a large set of unlabelled 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 $Y = \{y_1, \ldots, y_m\}$: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\} \in X \times Y$.

In many classification tasks it is common that a large pool of unlabelled examples U is available whereas the cost of generating a label for an example is high. The concept of active learning (Cohn et al. 1994a) tackles this problem by enabling a learner to pose specific queries, chosen from an unlabelled 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 Cohn et al. (1994a), 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 Schohn and Cohn (2000), the query function chooses the next unlabelled data point closest to the decision hyperplane in the kernel induced space. In a more recent approach (Xu et al. 2004), the unlabelled 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. SVM with active learning have been widely used for image retrieval problems (Luo et al. 2005; Wang et al. 2003) and in the drug discovery process (Warmuth et al. 2003).

In Kang et al. (2004), 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 Buhmann and Zöller (2000). 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 McCallum and Nigam (1998), an EM approach is used to integrate the information from unlabelled 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 Nguyen and Smeulders (2004). 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.

There have been several attempts to balance between exploration and exploitation in active learning. In the work of Baram et al. (2004), a combination of different active learning algorithms that focus either on the current classification boundary or

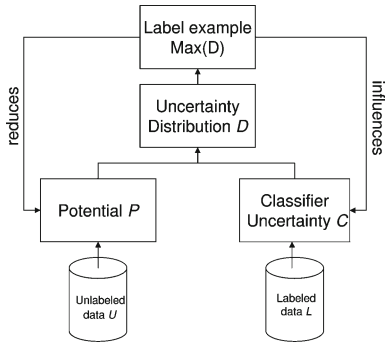


Fig. 1 Interaction of unlabelled and labeled data, potential and classifier uncertainty

on exploration have been used together with an algorithm for the multi-armed bandit problem and a novel performance evaluation measure. The decision whether to choose an algorithm for exploration or exploitation depends on the performance measure and does not take into account the distribution of the underlying input data.

In the work of Osugi et al. (2005), a Kernel-Farthest-First algorithm is used for exploration in active learning with SVM. The choice of an exploration step depends on the change that is induced with the newly labeled example on the hypothesis space. Results on benchmark data sets show a slight improvement of the classification accuracy with this technique.

Our approach differs from the others in the way that we use an integrated approach with a classification model that combines the *potential* of each data point (which is based on density estimates on the unlabelled data) and the *classifier uncertainty* (based on the labeled data) in one single criterion which we call *uncertainty distribution* (see Fig. 1). Instead of preclustering the datasets, we use the 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 the work of Cohn et al. (1994b), having an input distribution $p(\boldsymbol{x})$, it is proposed to minimize the expected error weighted by the input distribution:

$$\int_{\boldsymbol{x}} E_T \left[(\widehat{y}(\boldsymbol{x}) - y(\boldsymbol{x}))^2 | \boldsymbol{x} \right] p(\boldsymbol{x}) dx \tag{1}$$

where $E_T[.]$ denotes the expectation over training set L, $\widehat{y}(\boldsymbol{x})$ is the learner's output and $y(\boldsymbol{x})$ is the true label of $\boldsymbol{x}$. Rather than trying to minimize the value of Eq. 1 by minimizing the learner's variance (as proposed in Cohn et al. 1994b), we try to select examples with the biggest contribution to the current error of the classifier, similar to the approach in Nguyen and Smeulders (2004). Equation 1 shows that we need to weight the uncertainty of the classifier with the input data distribution $p(\boldsymbol{x})$. 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 unlabelled 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 $\boldsymbol{x}_i$, according to the subtractive clustering approach from Chin (1997) as:

$$P(\boldsymbol{x}_i) = \sum_{j=1}^{n} e^{-\alpha d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}, \ \alpha = \frac{4}{r_a^2} \tag{2}$$

where r_a is a positive constant defining a neighborhood. $d()$ is a distance measure, usually the euclidean distance is used. All points $\boldsymbol{x}_j$ that lie within this neighborhood have a large influence on the potential of data point $\boldsymbol{x}_i$. 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 resulting potential. An efficient way to find the nearest neighbors within a given distance is to use KD-Trees (Bentley 1975) as underlying data structure.

After the potential of each data point has been computed the data point $\boldsymbol{x}_k^*$ with the highest potential $P(\boldsymbol{x}_k^*)$ 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:

$$P(\boldsymbol{x}_i) \Leftarrow P(\boldsymbol{x}_i) - P(\boldsymbol{x}_k^*) e^{-\beta d(\boldsymbol{x}_i, \boldsymbol{x}_k^*)^2}, \ \beta = \frac{4}{r_b^2} \tag{3}$$

where r_b is a positive constant defining the neighborhood in which the potentials are reduced. In the work of Chin (1997), r_b has been set to $1.25 r_a$. In this work, we set the value of r_b in the same way. Note that from the algorithmic point of view any density

[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.

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 K nearest prototypes of a query point $\boldsymbol{x}_q$ are denoted by $\boldsymbol{p}_k, k = 1, \ldots, K$. The class label $\widehat{f}(\boldsymbol{x}_q)$ is determined by taking the class label of the K nearest prototypes $f(\boldsymbol{p}_k)$ with the largest prototype weight.

$$\begin{aligned} \widehat{f}(\boldsymbol{x}_q) &= \max_{v \in Y} \sum_{k=1}^{K} w_k \delta(v, f(\boldsymbol{p}_k)) \\ \delta(a, b) &= \begin{cases} 1, & \text{if } a = b \\ 0 & \text{else} \end{cases} \end{aligned} \tag{4}$$

where

$$w_k = \frac{1}{d(\boldsymbol{x}_q - \boldsymbol{p}_k)^2} \tag{5}$$

Regarding all currently labeled examples as prototypes, we can calculate for all data points their class probabilities $c(y_i)$ for all classes $y_1, \ldots, y_m$:

$$c(y_i) = \sum_{k=1}^{K} w_k \delta(y_i, f(\boldsymbol{p}_k)) \tag{6}$$

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 probabilities.

We compute the entropy of the histogram of the class probabilities and refer to it as *classifier uncertainty* C:

$$C(\boldsymbol{x}_q) = H(c(y_1), \ldots, c(y_m)) = -\sum_{j=1}^{m} c(y_j) \log_2(c(y_j)) \tag{7}$$

The resulting entropy value must be normalized with the number of classes in the current iteration $H_{\max} = \log_2 m$. 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, inversely related to the voting confidence of the nearest neighbor classifier.

3.3 Combination

Based on the *potentials* P that we compute on the unlabelled 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.

$$D(\boldsymbol{x}_i) = (1 - \epsilon)P(\boldsymbol{x}_i) + \epsilon C(\boldsymbol{x}_i) \tag{8}$$

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 Sect. 3, we outline our new 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.

The performance of the PBAC algorithm depends on its two parameters: the value of r_a the radius of the neighborhood and ϵ the value that controls the influence of $C(\boldsymbol{x}_i)$ (both values are in the range of [0, 1]). The radius r_a controls how many points in the neighborhood are considered in the estimation of the potential. A large radius provides a smoother estimate, but as this radius is also used for the reduction of potentials around a data point, this can result in too much reduction—leaving some areas in the datasets unexplored. If the value of the radius is too small, too many high peak values for exploration will be obtained which will result in too many unnecessary queries. The second parameter ϵ controls the point of time when the exploitation takes place. As the potentials are reduced in each iteration, the focus will be laid on the classification uncertainty $C(\boldsymbol{x}_i)$ in subsequent iterations in a natural way. But a higher ϵ value will result in more exploitation in the first steps. The choice of ϵ depends on the underlying datasets but should not be too high in order to allow for some exploration in the beginning. We analyze the influence of the parameters in Sect. 5.1.

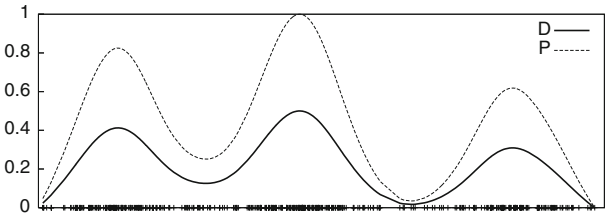
Algorithm 1 Prototype Based Active Classification

Require: Threshold t
1: GlobalPotential $\Leftarrow 0$
2: **for all** $\boldsymbol{x}_i \in U$ **do**
3: Compute the potential $P(\boldsymbol{x}_i)$ according to Equation 2.
4: GlobalPotential $\Leftarrow$ GlobalPotential + $P(\boldsymbol{x}_i)$
5: **end for**
6: **while** GlobalPotential $>$ t **do**
7: **for all** $\boldsymbol{x}_i \in U$ **do**
8: Compute the classifier uncertainty $C(\boldsymbol{x}_i)$ according to Equation 7.
9: Compute the uncertainty distribution $D(\boldsymbol{x}_i)$ according to Equation 8.
10: **end for**
11: Select the data point $\boldsymbol{x}_t$ with the highest uncertainty distribution.
12: Obtain a class label y_j for $\boldsymbol{x}_t$.
13: Create a new prototype with values $\boldsymbol{x}_t$ and class label y_j.
14: Classify the datasets with the current set of prototypes.
15: Reduce the potentials according to Equation 3.
16: **end while**

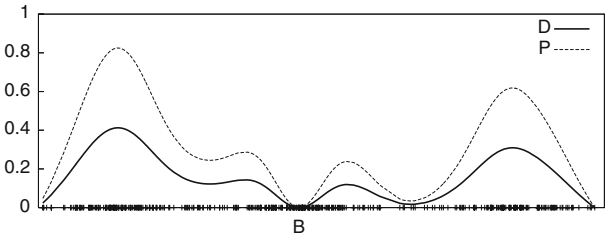
We demonstrate the mode of operation of the PBAC algorithm on a 1-dimensional artificial example with three classes (A, B and C) in Fig. 2. In Fig. 2a, the potentials for all data points (plotted on the x-axis) indicate that there are three distributions that need to be explored. The exploitation factor 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 Fig. 2b). After choosing the next point with class label A with a high potential in step 2 (Fig. 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 (Fig. 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 datasets 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

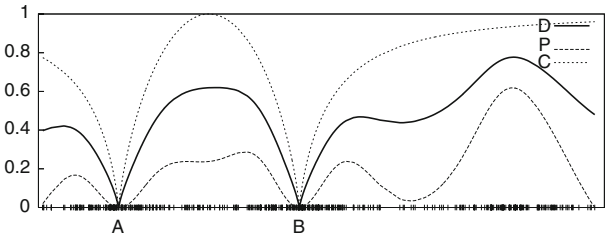
We have chosen different datasets from the UCI Machine Learning Repository (Asuncion and Newman 2007) to demonstrate the effectiveness of our active learning algorithm. In Sect. 5.1, we analyze the influence of the parameters of the PBAC algorithm on the segment data, before comparing our algorithm against a margin-based Active SVM with randomly chosen examples for initialization. In subsequent experiments on the segment data, the satimage data and the pendigits data from the UCI Machine Learning Repository (Asuncion and Newman 2007), we compare the PBAC algorithm against our own Active LVQ classification scheme (Cebron and Berthold 2006) and an Active Representative SVM, that uses clustering to select representative examples.



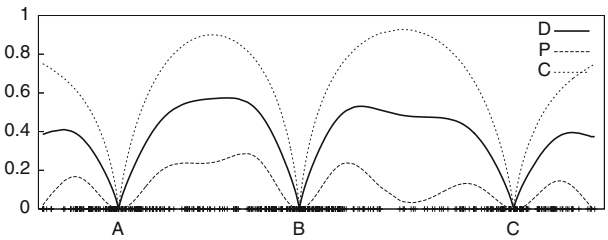
(a) Step 0



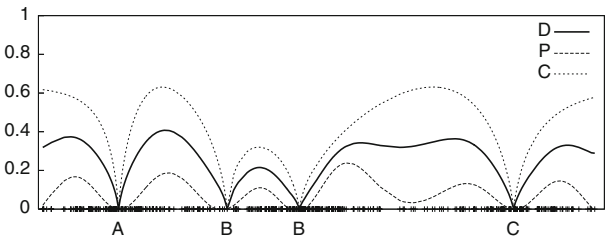
(b) Step 1



(c) Step 2



(d) Step 3



(e) Step 4

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

We have evaluated the performance of three prominent kernels (polynomial, RBF and hyper-tangent kernel) that are used by the support vector machine on the whole classified datasets. Then we choose the best performing kernel and tried out different reasonable values for its parameters and the overlapping penalty. All training instances are first assumed to be unlabelled. The performance is always measured on a separate test data set. In all figures we show the number of samples that are classified by the expert on the x-axis versus the resulting classification accuracy on a separate test set in percent on the y-axis.

5.1 Segment data

The segment data from the UCI Machine Learning Repository (Asuncion and Newman, 2007) consists of numerical features of different outdoor images. The images were handsegmented to create a classification for every pixel.

Before comparing the different active learning methods with each other, we want to analyze the influence of the parameters r_a and ϵ in the PBAC algorithm on the segment data. In Fig. 3, the ϵ-value has been set to zero to show the influence of the radius r_a. A larger radius seems to be beneficial in the first iterations, whereas a small radius leads to more regions with high potential. This causes more exploration and leads to a more detailed (but slower) exploration of the datasets, which proves beneficial in later iterations.

In Fig. 4, the value of r_a has been fixed to 0.4 to analyze the influence of the parameter ϵ. A too high value of ϵ obviously leads to a bad and instable classification performance. A moderate value in the range of 0.4 shows a little improvement with respect to a small value in classification accuracy. This demonstrates that it may be

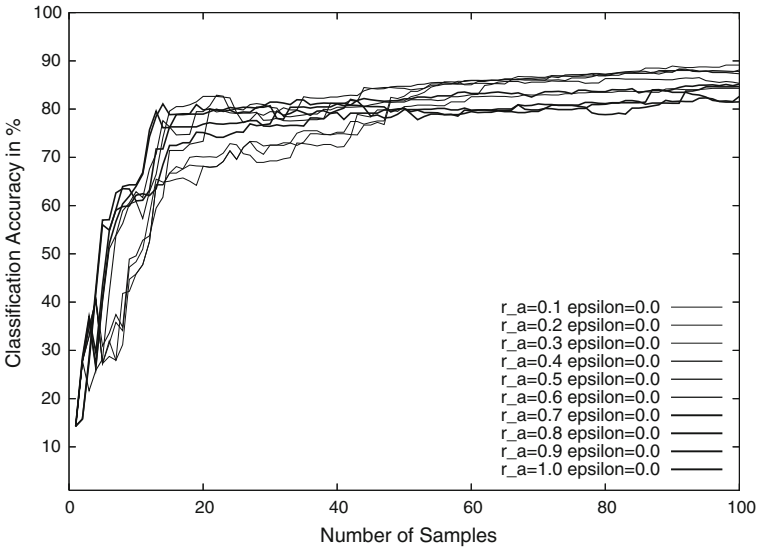


Fig. 3 Segment data: influence of the parameter r_a

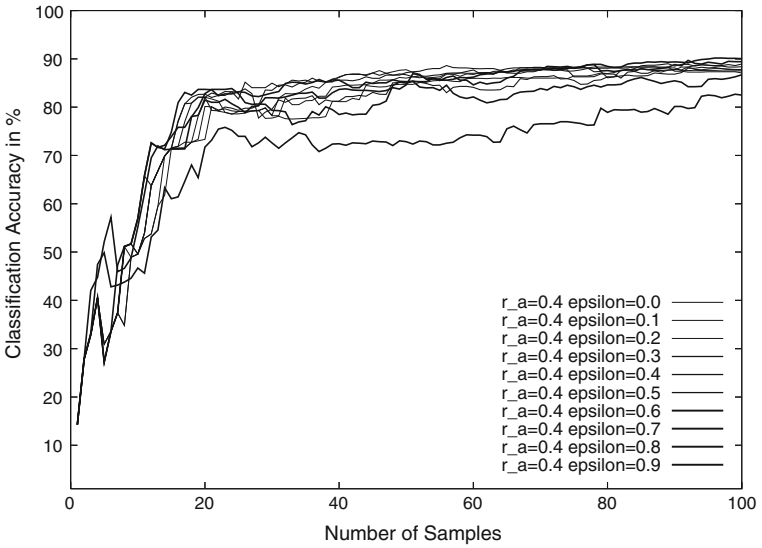


Fig. 4 Segment data: influence of the parameter ϵ

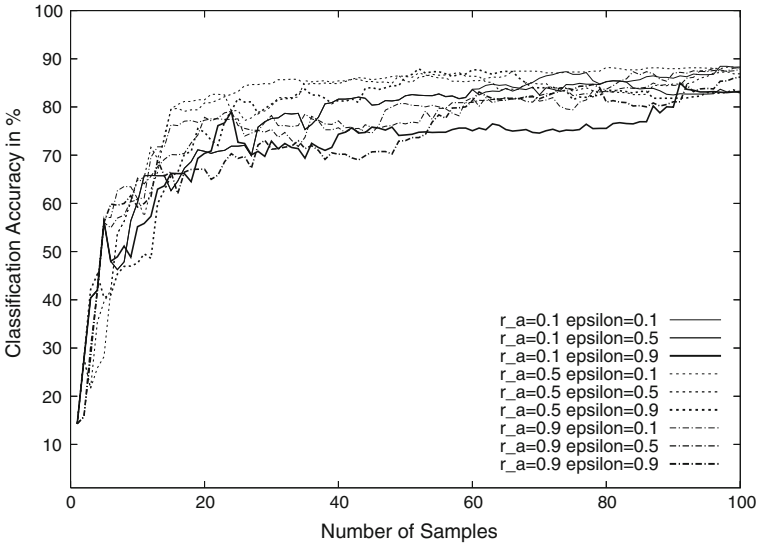


Fig. 5 Segment data: influence of the parameters r_a and ϵ

sometimes useful to lay the focus on exploitation even before the potentials have been totally reduced.

The interaction of the two parameters r_a and ϵ is more complex. We try to get an insight by showing combinations of values in a low, medium and high range in Fig. 5. In this figure, especially the graphs with high ϵ stand out from the average performance. In order to gain an average performance in the tests, we set the value of

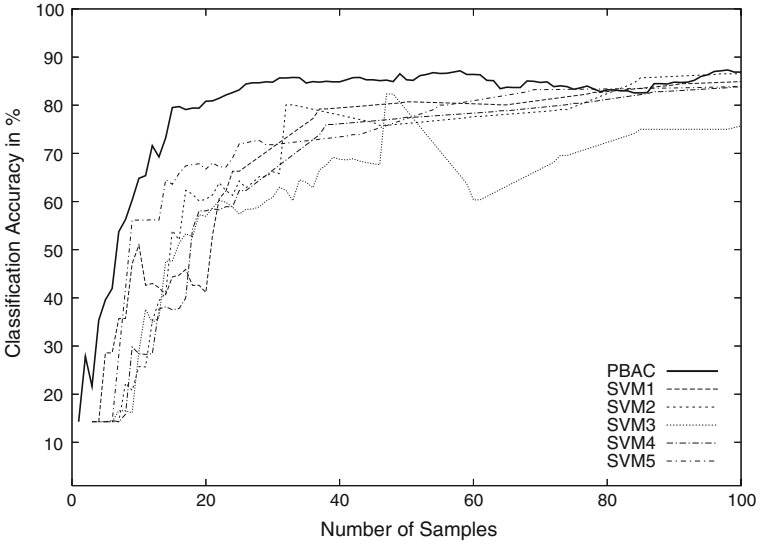


Fig. 6 Segment data: stability of the PBAC classifier against several runs of an active SVM

r_a and ϵ to 0.4 in all comparisons of the PBAC algorithm. This does not guarantee the best performance on all datasets, but as the differences are not that severe, a fixed parameter combination may be the best choice.

Another issue that we want to take a look at is the stability of our approach compared to an Active Margin SVM (Schohn and Cohn 2000) that is initialized with random examples and queries new samples at the decision hyperplane, see Fig. 6. The SVM was trained with a RBF kernel with $\sigma = 1.0$. One can see the effect of choosing representative examples in the stability of our PBAC algorithm against random selection in the SVM for the first queries. The random selection SVM technique does not reach the same performance and is, especially during earlier epochs, severely unstable. Performance can drop easily during subsequent queries. Since our potential 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 datasets is clustered to find representative examples for the initialization of the SVM, similar to Kang et al. (2004) and then examples inside the SVM's margin are clustered to find representative examples according to the approach in Xu et al. (2004). We call this procedure Active Representative SVM.

We now compare our PBAC algorithm against our own former Active LVQ classification scheme (Cebron and Berthold 2006) and the Active Representative SVM in Fig. 7. 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 examples and a weighted KNN classifier on all examples. Both the PBAC algorithm and the Active LVQ algorithm initialize in a stable way and gain a high classification accuracy with very few examples. The clustering

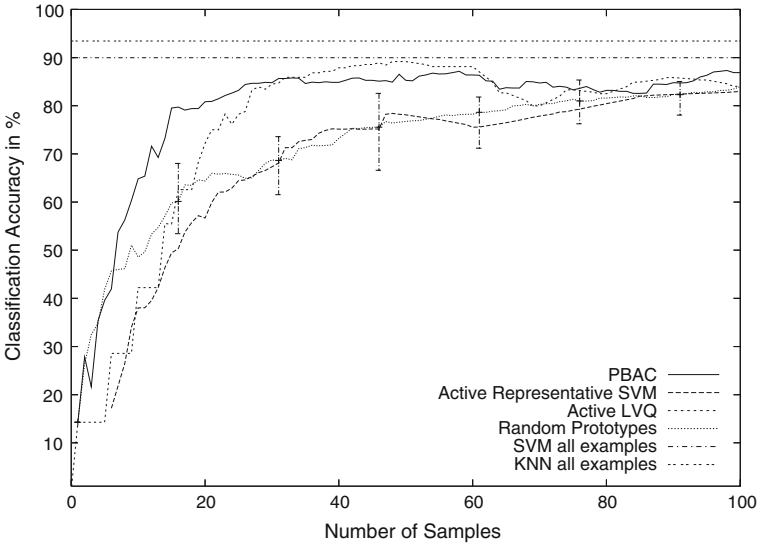


Fig. 7 Segment data: comparison of different classifiers

seems to be beneficial for the stability of the active SVM, but the performance on this datasets is below the other active learning schemes.

5.2 Satimage data

The satimage data from the UCI Machine Learning Repository (Asuncion and Newman 2007) consists of 4435 training instances which describe satellite image data by their multispectral values. The performance was measured on a separate test data set containing 2000 instances.

The PBAC algorithm and the Active Representative SVM have been initialized with the same parameters as mentioned above. 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 Fig. 8. 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%.

5.3 Pendigits data

The Pendigits datasets from the UCI Machine Learning Repository (Asuncion and Newman 2007) 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 datasets with

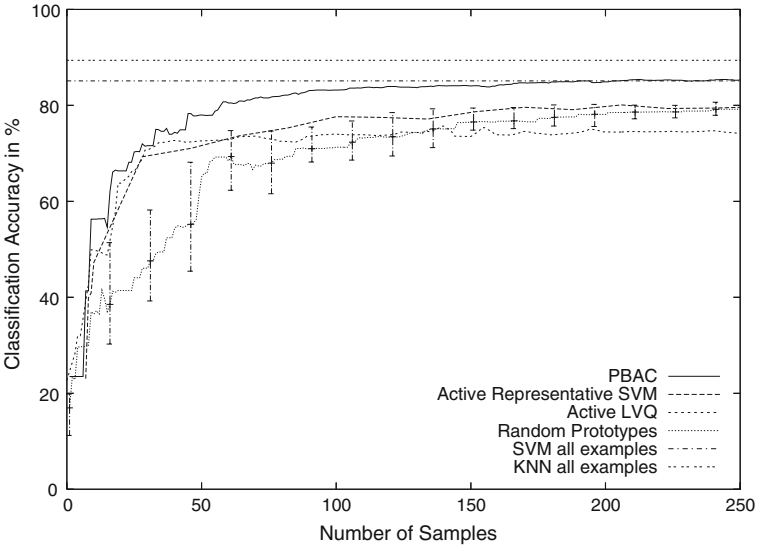


Fig. 8 Satimage data: comparison of different classifiers

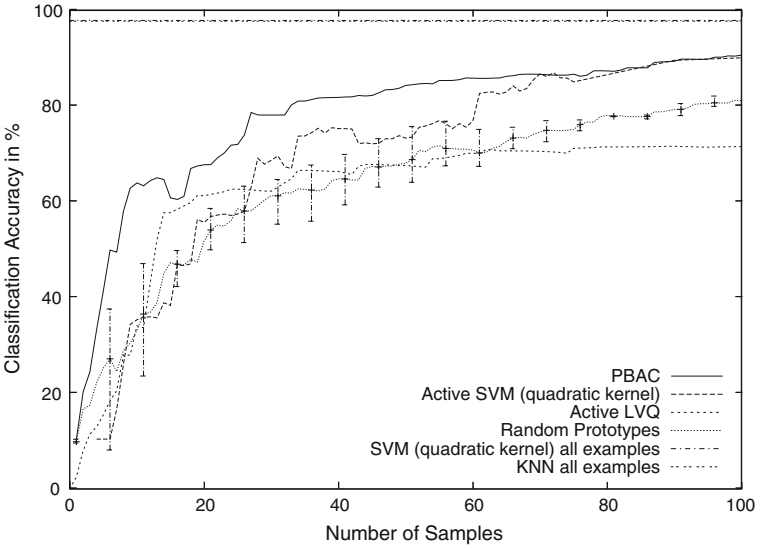


Fig. 9 Pendigits data: comparison of different classifiers

7,494 examples for training the classifier and a separate test datasets containing 3,498 examples for measuring the classification accuracy. Figure 9 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 7,494 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. 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 stably at the beginning but performs worse than all other learning schemes in subsequent learning iterations. The PBAC algorithm performs stably 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.

5.4 Prototype based classification vs. SVM classification

One cannot expect that our PBAC algorithm always performs better than a SVM with representative selection. Prototype based classification can suffer from the "curse of dimensionality", whereas a SVM is better suited for high-dimensional data (e.g. in text classification). In our case (classification of cell assay images) this problem is not so important. 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, SVMs 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 unlabelled datasets 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 datasets. 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 SVM 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 such applications and is essential for user acceptance of such an interactive learning system. Future work can be done by tuning the parameters of the PBAC algorithm to a specific problem. One possibility may be to fine-tune the ϵ value by measuring the change in the expected classification error.

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

References

Asuncion A, Newman D (2007) UCI machine learning repository. http://mlearn.ics.uci.edu/mlrepository.html

Baram Y, El-Yaniv R, Luz K (2004) Online choice of active learning algorithms. J Mach Learn Res 5:255–291

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Buhmann JM, Zöller T (2000) Active learning for hierarchical pairwise data clustering. In: International conference on pattern recognition (ICPR), Barcelona, Spain, vol II, pp 2186–2189

Cebron N, Berthold MR (2006) Adaptive active classification of cell assay images. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) PKDD, vol 4213 of lecture notes in computer science. Springer, Berlin, pp 79–90

Chin SL (1997) An efficient method for extracting fuzzy classification rules from high dimensional data. JACIII 1(1):31–36

Cohn DA, Atlas L, Ladner RE (1994a) Improving generalization with active learning. Mach Learn 15(2):201–221

Cohn DA, Ghahramani Z, Jordan MI (1994b) Active learning with statistical models. In: Tesauro G, Touretzky DS, Leen TK (eds) NIPS. MIT Press, Cambridge, MA, pp 705–712

Kang J, Ryu KR, Kwon H-C (2004) Using cluster-based sampling to select initial training set for active learning in text classification. In: Advances in knowledge discovery and data mining, vol 3056. Springer, Berlin, pp 384–388

Luo T, Kramer K, Goldgof DB, Hall LO, Samson S, Remsen A, Hopkins T (2005) Active learning to recognize multiple types of plankton. J Mach Learn Res 6:589–613

Mandel MI, Poliner GE, Ellis DPW (2006) Support vector machine active learning for music retrieval. Multimedia Syst 12(1):3–13

McCallum A, Nigam K (1998) Employing em and pool-based active learning for text classification. In: Shavlik JW (ed) Proceedings of the fifteenth international conference on machine learning (ICML 1998), Madison, WI, July 24–27, 1998. Morgan Kaufmann, San Fransisco, CA, pp 350–358

Nguyen HT, Smeulders AWM (2004) Active learning using pre-clustering. In: Brodley CE (ed) Machine learning, proceedings of the twenty-first international conference (ICML 2004), Banff, Alberta, Canada, July 4–8, 2004. ACM

Osugi T, Kun D, Scott S (2005) Balancing exploration and exploitation: a new algorithm for active machine learning. In: ICDM '05: proceedings of the fifth IEEE international conference on data mining. IEEE Computer Society, Washington, DC, pp 330–337

Schohn G, Cohn D (2000) Less is more: active learning with support vector machines. In: Langley P (ed) Proceedings of the seventeenth international conference on machine learning (ICML 2000), Stanford University, Stanford, CA, June 29–July 2, 2000. Morgan Kaufmann, San Fransisco, CA, pp 839–846

Wang L, Chan KL, Zhang Z (2003) Bootstrapping SVM active learning by incorporating unlabelled images for image retrieval. In: IEEE Computer Society conference on computer vision and pattern recognition (CVPR 2003), Madison, WI, June 16–22, 2003. IEEE Computer Society, pp 629–634

Warmuth MK, Liao J, Rätsch G, Mathieson M, Putta S, Lemmen C (2003) Active learning with support vector machines in the drug discovery process. J Chem Inf Comp Sci 43(2):667–673

Xu Z, Yu K, Tresp V, Xu X, Wang J (2004) Representative sampling for text classification using support vector machines. In: ECIR 2003, vol 2633. Springer, Berlin, pp 393–407