Investigating Exploratory Capabilities of Uncertainty Sampling using SVMs in Active Learning

Dominik Lang Daniel Kottke Georg Krempl Myra Spiliopoulou

dominik.lang@st.ovgu.de, {daniel.kottke, georg.krempl}@ovgu.de, myra@iti.cs.uni-magdeburg.de
KMD Group, Otto von Guericke University Magdeburg, Germany

Abstract

Active learning provides a solution for annotating huge pools of data efficiently to use it for mining and business analytics. Therefore, it reduces the number of instances that have to be annotated by an expert to the most informative ones. A common approach is to use uncertainty sampling in combination with a support vector machine (SVM). Some papers argue that uncertainty sampling performs badly due to missing exploration, others report good results using an SVM. This paper investigates whether uncertainty sampling is able to explore the data space due to the kernel trick used by the SVMs. Hence, we evaluate this on multiple synthetic and real datasets and the effects of parameter tuning and kernel selection for different evaluation criteria.

1 Introduction

In classification tasks, active learning methods intelligently select unlabeled instances to be labeled by an expert. The aim of active learning is to request labels from those instances that improve the classifier's performance the most [22]. To select the most useful instances, active algorithms should (1) explore the data space to find regions with unexpected labels and (2) do exploitation, i.e., to refine the classifier's decision boundary [8, 18].

One of the most commonly used methods is uncertainty sampling (US) which preferably samples instances near the decision boundary. This behavior is often considered as pure exploitation, without exploration in the strict sense. Some articles argue that this lack of exploration is the main drawback of US [4] and they claim that this behavior of US may explain its inferiority to the purely exploratory random sampling [11, 22]. Some authors therefore propose exploratory components for US [2, 20], while others capitalize the exploratory behavior of SVMs, letting US sample instances near the SVM decision boundary [12]. These characteristics suggest that the combination of SVMs and US might be promising. The rationale behind this is, that since SVMs use the kernel trick to learn a linear separation of two classes, sampling close to this decision boundary should be sufficient and exploration gets unnecessary. But is this truly the case?

In this paper, we investigate to what extend US, when combined with differently tuned SVMs, covers the original data space to learn a classification model. Furthermore, we propose an evaluation framework to measure the influence of this coverage (exploration) on the classification performance.

To demonstrate the interplay of US and a classifier in Fig. 1, we depict an exemplary one-dimensional dataset, where we show how the behavior of US is affected by the behavior of the classification algorithm. We see a dataset with two classes distributed across three clusters. A Bayesian classifier or even a very simple linear classifier would probably detect the boundary, whereupon US would *not* select instances from the cluster at the right. An SVM may, depending on the fit of its hyperparamters, place instances from the cluster at the right

In: G. Krempl, V. Lemaire, E. Lughofer, and D. Kottke (eds.): Proceedings of the Workshop Active Learning: Applications, Foundations and Emerging Trends, AL@iKNOW 2016, Graz, Austria, 18-OCT-2016, published at http://ceur-ws.org

inside the decision area, whereupon US would readily consider them. Since the behavior of an SVM depends on the chosen kernel and the SVMs hyperparameters, we investigate the dependencies between the exploratory capabilities of US with SVMs and the chosen kernel, also w.r.t. its tuning.

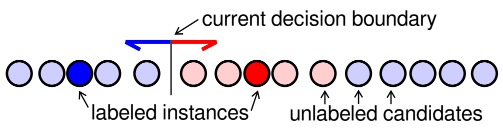


Figure 1: Two-class dataset with three clusters and a linear classifier.

The following section briefly summarizes the background and related work, followed by a description of the experimental framework in section 3 and the conducted experiments in section 4. Finally, we provide a discussion and conclude our results.

2 Background and Related Work

Active learning (AL) is a special area of machine learning, more precisely semi-supervised learning. A pool-based active learner successively selects and removes an instance $x \in \mathcal{U}$ from a large pool of unlabeled instances $\mathcal{U}$. An expert annotates this instance with a class label y. This information is added to the labeled set of instances $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, y)\}$ which forms the training basis for a classifier [22].

Different strategies exist to determine which instances are chosen. A simple but naive approach is selecting instances at random, which has the benefit of potentially sampling a well distributed set of instances. A common approach is called uncertainty sampling (US), its underlying rationale being that the learner should select those instances the classifier is most uncertain about. For SVMs this might be the distance to the decision boundary (simple margin [23]), probabilistic classifiers might use the posterior estimates.

Uncertainty sampling is solely focused on exploitation of the data [4], i.e. it acquires the labels of instances that are useful to refining the decision boundary already assessed by the classifier. If the decision boundary proposed by the model is close to the actual decision boundary for the data, this process of refinement is likely to perform well. However, if there are unexplored regions in the data space with wrongly predicted labels far from the decision boundary, they will not be found. One approach to overcome this problem is to alternate between US for exploitation and an exploration component like random sampling [18, 16] or a semi-supervised learning method[13, 12]. Another approach is to improve the US selection criterion either by improving the uncertainty measure [23] or adding additional components to the criterion like expected model change [3, 12], representativeness and diversity [6, 10, 12] or confidence [15].

A Support Vector Machine (SVM) is a supervised learning model performing binary classification in a kernel-induced feature space [12]. For the binary classification problem (+/-), an SVM learns a decision hyperplane that separates the classes in the kernel-induced topological space by a maximal margin. Therefore, an SVM defines an objective function, where the sign indicates the predicted class of an instance x. The absolute value of this objective function represents the distance of an instance x to the SVM decision hyperplane. Some of the most commonly used kernel functions for support vector machines are the Polynomial, RBF, Sigmoid and Laplacian kernels [21].

In this context, the goal of an SVM is to achieve an optimal separation in the kernel-induced space. As the hyperplane is gradually 'refined', exploration becomes less and less important. Indeed, Tong and Koller [23] assume a quick reduction of the version space size using US since the current hypothesis of an SVM is roughly in the center of the version space. Hence, choosing an instance near the center should approximately split the version space in halves [23, 18]. There have been arguments against this, suggesting that the simple margin implementation of US fails to achieve this approximate halving of the version space in some cases [17, 12]. An aspect that has to be considered in the discussion about US with SVMs and exploration is the question of how the SVM hyperparameters are chosen, since it has been shown that they strongly influence the performance of the active learner [5, 14]. A problem with using common methods to determine appropriate SVM hyperparameters, such as tuning them through grid search, in the context of AL is that the required labeled data needed for such approaches is not available at the start of the AL process. However the most commonly used kernel function for SVM AL of those mentioned above is RBF [6, 7, 13, 16, 18], which requires only one parameter. This popularity might be due to the fact that the RBF kernel has been shown to learn concepts well even with few available training examples [17] (as is usually the case in active learning scenarios) and tends to explore the feature space more than its alternatives, although it can be more sensible to noise in the data [5]. Lin and Lin recommend using the RBF kernel for most applications, but also suggest that the Sigmoid kernel can behave similar to RBF given certain hyperparameters [14].

3 Evaluation Framework

In our experiments, we use pool-based active learning as described in Sec. 2 and stop learning after 50 label acquisitions (budget $B = 50$). For uncertainty sampling (US), we use the most commonly used simple margin (sm) implementation for SVMs by [23]. To compare the exploratory behavior, we additionally use a random sampler with each classifier (SVM+kernel+tuning). Since an SVM requires at least one instance of each of the two classes for training the labeled set $\mathcal{L}$ set was initialized accordingly.

In Alg. 1, we show our framework to evaluate our active learning experiments. For a given data model $\mathcal{M}$, we perform US using SVMs with different kernels as defined in [21], namely a polynomial kernel, a radial basis function kernel, a sigmoid kernel, and a Laplacian kernel.

To tune the hyperparameters of each classifier (SVM+kernel), we perform a grid search on a separate tuning set D_{tune}. This is generated by selecting B labeled instances from the data model $\mathcal{M}$, to optimize the classifier according to the final number of labels. The labels in the D_{tune} set are used solely for tuning and validating the hyperparameters. To avoid a bias of the model due to overfitting, the labels in D_{tune} are not used in the training or testing of the active learner. Based on the hyperparameter optimization, we select for each kernel (1) the best parameter setting and (2) a parameter setting that achieved medium results. Surely, the classification performance will probably decrease by selecting a non-optimal parameter setting. Here, we want to investigate the influence of the parameter choice for doing exploitation. Hence, we get two different classifiers for each kernel $C_{k,t}$. The search space of the hyperparameter optimization is given in Tab. 1: γ denotes the kernel coefficient, C is the penalty parameter of the error term of the SVM, d is the degree of the polynomial kernel and $coef_0$ is the independent term of the kernel function [19].

Parameters	Value Space
γ	$\{1e^{-5}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 0.2, 0.4\}$
C	$\{1, 10, 100, 200, 400, 1000\}$
d	$\{1, 2, 3, 4\}$
$coef_0$	$\{0.0, 0.1, 0.2, 0.3\}$

Table 1: Search space for hyperparameter tuning

The experiments are conducted based on a set of 200 different seeds that were used for generating resp. splitting the datasets. For each seed, we generate a test set D_{test} which we use for evaluation and a training set $\mathcal{U}$ consisting of solely unlabeled instances. As mentioned above, the labeled set $\mathcal{L}$ is initialized with one instance from each of the two classes. Then, the active learner chooses B instances successively to be labeled. The unlabeled instance is removed from $\mathcal{U}$ and added (with the corresponding label) to $\mathcal{L}$.

As the model $\mathcal{M}$ is not explicitly given for the real world datasets, we split B instances from the real datasets for hyperparameter tuning. Then, half of the remaining instances is used for testing, the other half for training.

In the evaluation step, we focus on two aspects: (1) the classifier's performance, and (2) the amount of exploration. To address the classifier's performance, we determine the hold-out accuracy on the test set D_{test} for each budget step resulting in a learning curve for each active learner and each classifier. The exploration is addressed by a score motivated by [6]. Here, we determine the average euclidean distance from each instance $x \in D_{test}$ to the nearest labeled instance $x_l \in \mathcal{L}$ (see Eq. 1). This score (avg. min. distance) is determined after each acquisition, which again results in a learning curve. A low value indicates a good coverage of the data space, high values indicate that labeled instances are far away. Per definition, this score is decreasing over time, as more labeled instances are added subsequently to the labeled set.

$$\frac{1}{|D_{test}|} \sum_{x \in D_{test}} \min_{x_l \in L} ||x - x_l||_2 \tag{1}$$

Regarding the expected results of the experiments, it is difficult to make predictions as to the differences in performance and exploration between the different kernel functions. Generally, it is to be expected that the rbf and laplacian kernel will show a similar but not identical behavior based on their similarity. Among the artifical datasets the ones referred to as 'Three Cluster Datasets' (Sec. 4.1) are designed with an expected result in mind:

```
Data: data model $\mathcal{M}$, seeds $S$, budget $B = 50$, SVM $\mathcal{C}$, active method US
begin
    for $k \in \{\text{RBF}, \text{Poly}, \text{Sigm}, \text{Laplace}\}$ do
        for $t \in \{\text{best}, \text{middle}\}$ do
            $D_{tune} \leftarrow \text{getLInst}(\mathcal{M}, B)$;
            $p_{k,t} \leftarrow \text{gridSearch}(\mathcal{C}(k), D_{tune}, t)$;
            $C_{k,t} \leftarrow \text{initializeClassifier}(\mathcal{C}(k), p_{k,t})$;
            for $s \in S$ do
                $D_{test} \leftarrow \text{getLInst}(\mathcal{M}, 350, s)$;
                $\mathcal{L} \leftarrow \text{getOneLInstPerClass}(\mathcal{M}, s)$;
                $\mathcal{U} \leftarrow \text{getUInst}(\mathcal{M}, 350, s)$;
                for $i \in \{1, \dots, B\}$ do
                    $C^* \leftarrow \text{trainClassifier}(C, \mathcal{L})$;
                    $x^* \leftarrow \text{selectInst}(\text{US}, \mathcal{U}, C^*)$;
                    $y^* \leftarrow \text{getLabel}(\mathcal{M}, x^*)$;
                    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{x^*\}$;
                    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x^*, y^*)\}$;
                    $\text{evaluate}(C^*, \mathcal{L}, D_{test})$
                end
            end
        end
    end
end
```

Algorithm 1: Evaluation framework

since one of the three clusters is unknown to the active learner at the beginning because only two labels are given in the initial L set, we expect that learners using US with a linear classifier will find this cluster very late. However, as these clusters are generated with Gaussian distributions, it is likely that both the rbf and laplacian kernel will perfom better by using this implicit information.

4 Experimental Evaluation

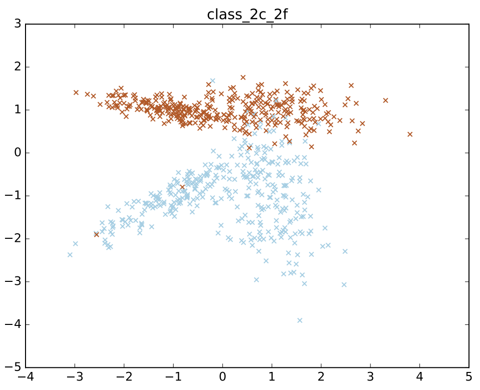
In the experimental evaluation, we use 10 different data models resp. datasets to investigate the exploratory capabilities of uncertainty sampling and SVMs (see Fig. 2a-2f). Therefore, we used a cluster system running the NeuroDebian system [9]. First, we discuss the results on an artificial dataset consisting of three clusters and investigate, if uncertainty sampling acquires labels in every cluster. We then try to generalize the findings on further artificial datasets from a standard library and on real data.

4.1 Three Cluster Dataset

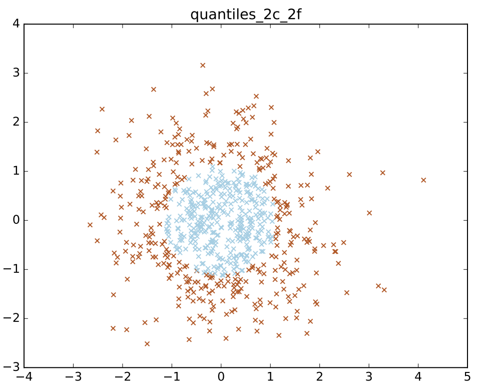
The first model consists of three Gaussian clusters, two clusters of class one and the other cluster of the second class. To get more expressive results, we use three different standard deviations (equal for all clusters) to have well separated clusters as well as overlapping ones (high standard deviation (overlapping): gaussian_2dp0, medium: gaussian_2dp1, low (well-separated): gaussian_2dp2). Since 2 of the 3 clusters are sampled in the initialization step, the learner's task is to discover the remaining cluster in order to perform well.

If a learner-classifier combination was able to find the remaining cluster is summarized in Tab. 2. In each cell, we show the percentage that the active learner found the unknown cluster within the first B label acquisitions across all 200 trials. Every classifier has a specific kernel and a set of hyperparameters coming from the grid search tuning. Here, we choose the best performing hyperparameters (best) as well as non-optimal ones (middle).

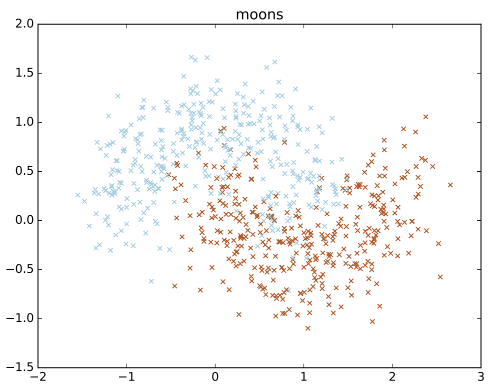
Choosing the best set of parameters, the SVMs with a laplacian and an rbf kernel are able to discover the remaining cluster reliably, whereas the discovery rate of those using a polynomial or sigmoid kernel is less than 25%. Applying not-optimal parameters, the detection rate of an SVM with an rbf decreases, but the rate of polynomial SVMs increases to approx. 50% and of sigmoid SVMs to higher than 80%. This exploratory behavior is also indicated by the average minimal distance score in Fig. 3b, 3d. Here, a high discovery rate correlates with a low value for the avg. min. distance. In this specific dataset, there is a clear correlation of the exploration score, resp. the discovery rate, and the learning curves in Fig. 3a. The previously mentioned expectation of the



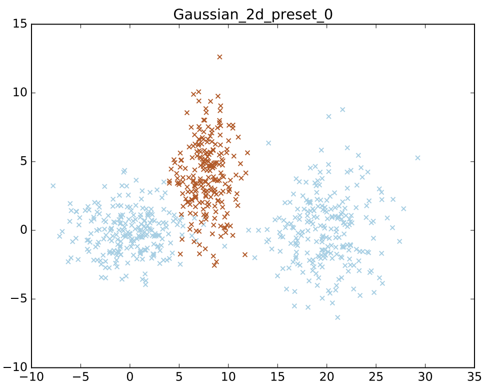
(a) Example of the 'classification' dataset



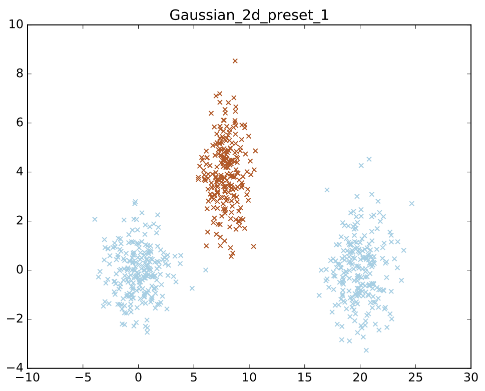
(b) Example of the 'quantiles' dataset



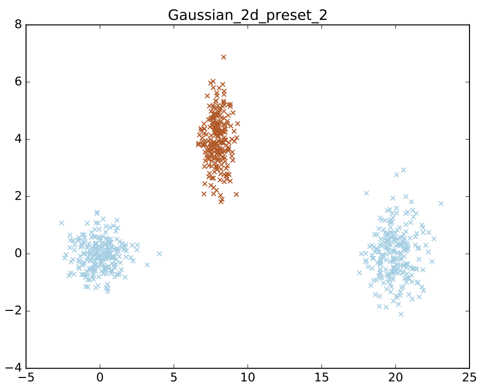
(c) Example of the 'moons' dataset



(d) Example of the 3 cluster 'Gaussian' dataset, preset 0

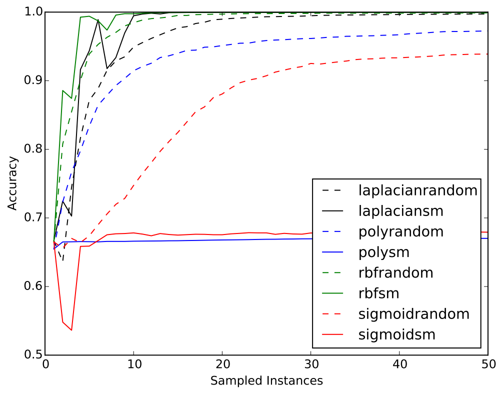


(e) Example of the 3 cluster 'Gaussian' dataset, preset 1

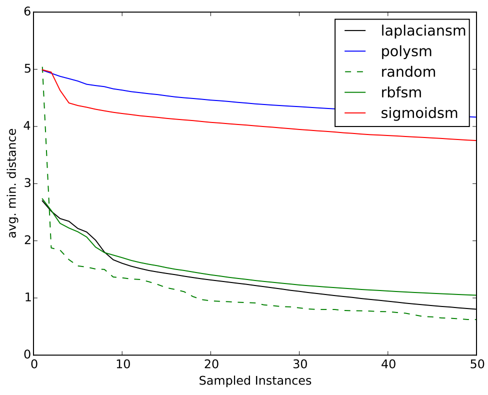


(f) Example of the 3 cluster 'Gaussian' dataset, preset 2

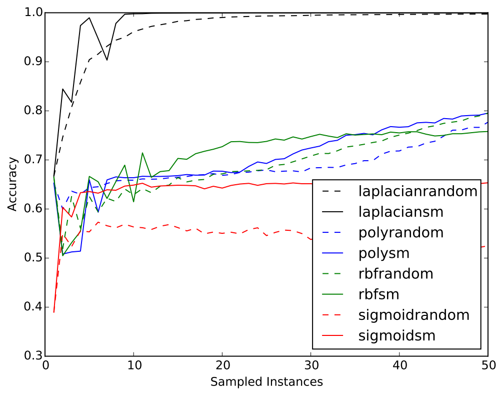
Figure 2: Examples for the various artificial datasets



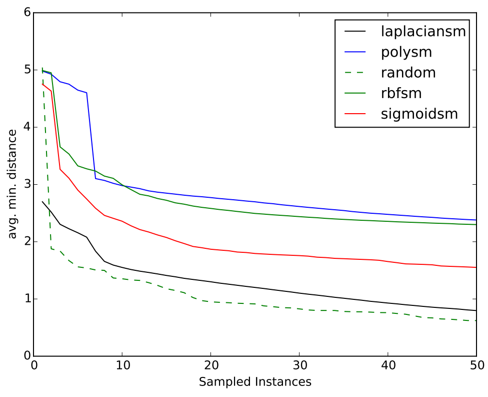
(a) Learning curve with best parameters



(b) Avg. min. dist curve with best parameters

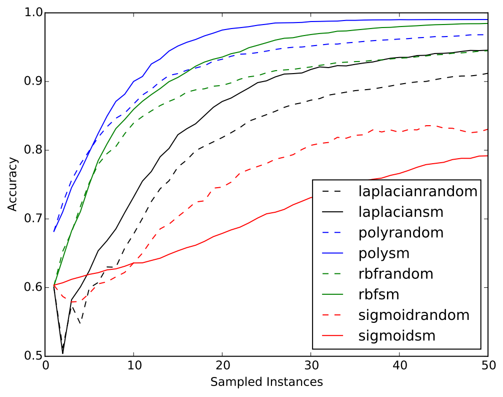


(c) Learning curve with middle parameters

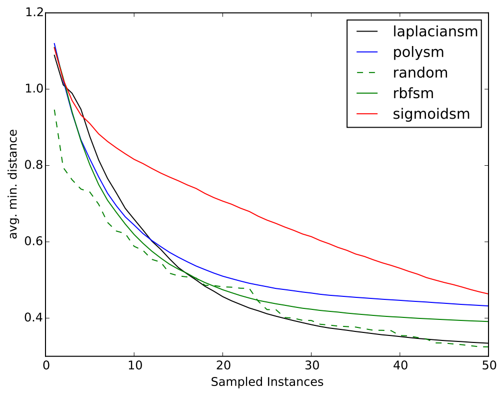


(d) Avg. min. dist curve with middle parameters

Figure 3: Results for the Gaussian_2dp1 dataset

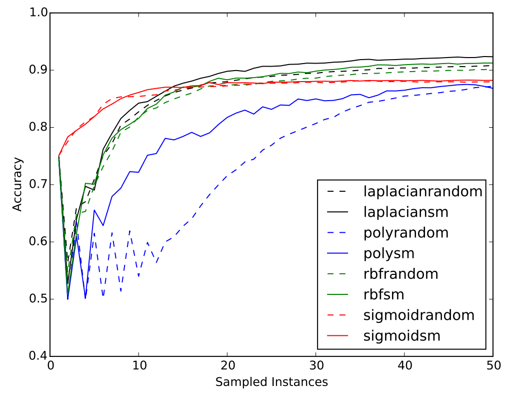


(a) 'Quantiles' learning curve

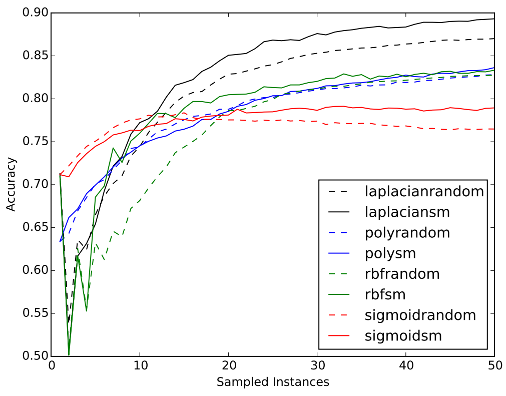


(b) 'Quantiles' avg. min. dist curve

Figure 4: Results using best parameters on the generated data using the scikit-learn functions

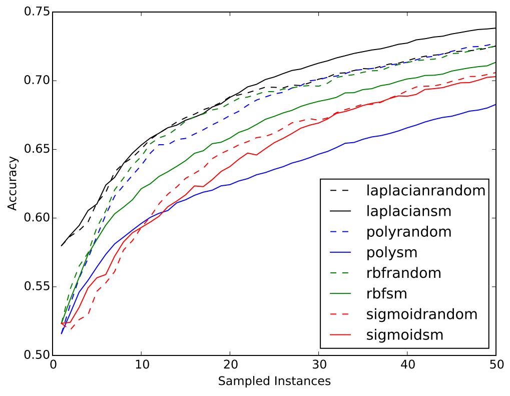


(a) 'Classification' learning curve

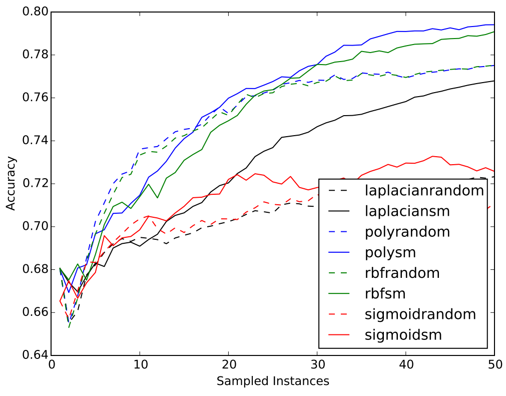


(b) 'Moons' learning curve

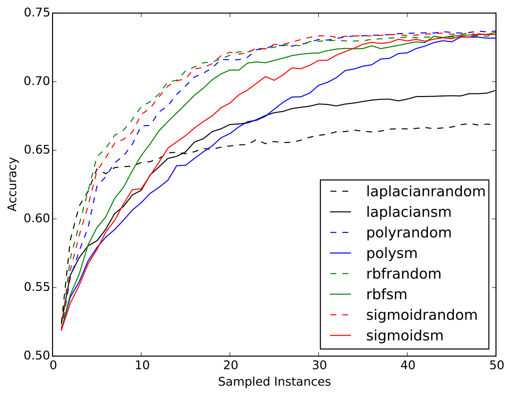
Figure 5: Results using best parameters on the generated data using the scikit-learn functions



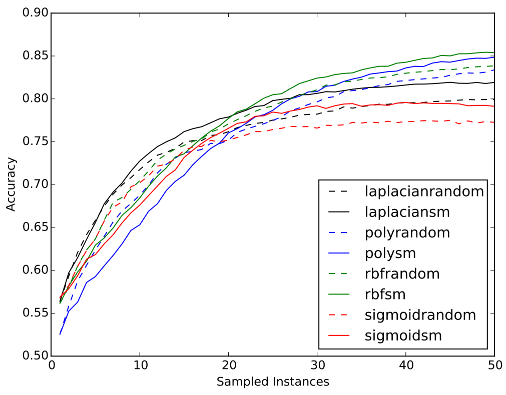
(a) Abalone



(b) Mammo



(c) Haberman



(d) Vertebral

Figure 6: Learning curves on the real-world datasets with best parameters

Params	Kernel/Data	p0	p1	p2
best	laplace	100%	100%	100%
	poly	23%	1%	0%
	rbf	100%	100%	100%
	sigmoid	18%	7%	22%
middle	laplace	100%	100%	100%
	poly	67%	49%	56%
	rbf	64%	56%	56%
	sigmoid	98%	94%	82%

Table 2: Discovery rate of the unknown cluster on the gaussian_2d dataset for the different presets and learner configurations using uncertainty sampling.

rbf and laplacian kernel learners to perform better based on the Gaussian distributions of the data was shown to be correct. The accuracy curves of the random learners can also be used to find out, if the tuned SVM is able to fit the decision boundary. Using the best hyperparameters, all random active learners achieve a performance higher than 0.9 accuracy which means that these are generally able to perform well. Here, a lack of exploration also leads to bad performance.

Using non-optimal parameters, the discovery rate might increase, but the ability to find an appropriate boundary decreases (except for the laplacian kernel), which is not surprising. Hence, an increase of exploration by changing the kernel does not necessarily increase the accuracy of a classifier, as the classifier is not able anymore to fit the decision boundary accordingly. Interestingly, uncertainty sampling now performs better with non-optimal hyperparameters compared to random in general. This observation could be critical to active learning research: In active learning research, the main focus is to compare different active learning algorithms. Hence, they fix a classifier and a hyperparameter setting (however this is determined) and compare the active methods with each other. This experiment shows that random and US methods using the same SVM with the same kernel can change their order by just varying their hyperparameters.

Note that these experiments are based on normal distributions. We expect circular shapes which might favors the performance of rbf or laplace kernels. Hence, we perform more experiments using less structured data.

4.2 Synthetic Datasets

The results of the synthetic datasets are given in Fig. 4a-4b and Fig. 5a-5b. All three datasets are provided by the scikit-learn library [19], namely 'quantiles', 'classification' and 'moons'. Here, we only show the results for the best tuning hyperparameters.

In the exemplary Fig. 4a-4b, a fast decreasing avg. min. distance (high exploration) in the early steps indicates a fast improvement in terms of accuracy. This tendency is also visible on the other datasets which we provide at our companion website[1]. This is also indicated by the observation that the random sampler is similar or better in the early steps (except for sigmoid on Quantiles) but beaten later by the uncertainty sampling (simple margin sm) method. This has also been shown by various other authors that an exploration phase in the beginning is beneficial.

Finding the best SVM+AL combination remains difficult: On Quantiles, the polynomial SVM in combination with simple margin is superior. On Classification and Moons, the winner is the laplacian SVM + simple margin.

4.3 Real-world datasets

The real-world datasets are chosen from the UCI machine learning repository [1] and include the 'Abalone', 'Haberman', 'Mammo' and 'Vertebral'. Their characteristics are summarized in Tab. 3. We transformed nominal attributes into multiple binary attributes, numerical attributes were normalized to $[0, 1]$.

On these datasets, we choose to show the learning curves of the well-tuned SVMs in Fig. 6a-6d as one would do this in practice. The winning kernel differs very much across the datasets. On Abalone, the laplacian kernel was superior; on Mammo it was the polynomial kernel and on Haberman and Vertebral, the rbf kernel was best.

On Mammo and Vertebral, we observe the same situation as in Sec. 4.2: random sampling outperforms simple margin in the early learning stage and the latter catches up or surpasses random sampling later on. The results on Abalone in Fig. 6a show that the laplacian kernel is beneficial. Even more interesting is that random outperforms

[1]`http://kmd.cs.ovgu.de/res/explore-us/`

Name	Attributes	Size
Abalone	8	4177
Haberman	3	306
Mammo	11	830
Vertebral	6	310

Table 3: Characteristics of the real-world datasets

the simple margin strategy for every other kernel. The same appears on Haberman in Fig. 6c. Only the laplacian kernel single margin strategy outperforms its random competitor, but the performance of both are far less then all others.

To summarize, there are quite a lot of cases where a solely exploratory strategy (random) outperforms the uncertainty sampling approach which was combined with a pre-tuned SVM to potentially add some exploration.

5 Discussion and Conclusion

In this paper, we investigated the exploratory capabilities of uncertainty sampling (US) in combination with different support vector machines (SVMs). We described an evaluation framework and tested multiple synthetic and real datasets using this framework. Furthermore, we proposed to use the average minimum distance as an indicator for exploration.

Although SVM and US are seen as a promising combination for active learning, it does not mitigate the lack of exploration to which the inferior performance of US is accredited to. The exploiting behavior of US in the kernel-induced space ends up looking similar to exploratory behavior in the feature space, yet it does not perform actual exploration in a strict sense.

If non-optimal SVM hyperparameters are used, the exploitation that US performs in the kernel-induced space becomes less precise which can lead to the behavior showing more exploratory characteristics. However, in a strict sense this is more a misbehavior of the exploitation than real exploration of the data space. Hence, we conclude that merely choosing a SVM to perform US does not replace a dedicated exploratory component.

The experimental results affirm that exploration in the beginning of the active learning process is indeed beneficial to the classification performance. Furthermore, they indicate that the hyperparameter tuning is critical to classification performance. We propose to validate multiple hyperparameters in an evaluation of active methods to get rid of the bias induced.

Acknowledgements

We thank Michael Hanke and Alex Waite for support by providing the cluster for our computations and Pawel Matuszyk for the discussions on that topic.

References

[1] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2015.

[2] Christian Beyer, Georg Krempl, and Vincent Lemaire. How to select information that matters: A comparative study on active learning strategies for classification. In *Proc. of the 15th Int. Conf. on Knowledge Technologies and Data-Driven Business (i-KNOW 2015)*. ACM, 2015.

[3] Wenbin Cai, Ya Zhang, Siyuan Zhou, Wenquan Wang, Chris Ding, and Xiao Gu. Active learning for support vector machines with maximum model change. In *Machine Learning and Knowledge Discovery in Databases*, pages 211–226. Springer, 2014.

[4] Gavin C Cawley. Baseline methods for active learning. In *Active Learning and Experimental Design@ AISTATS*, pages 47–57, 2011.

[5] Nicolas Cebron. *Aktives Lernen zur Klassifikation großer Datenmengen mittels Exploration und Spezialisierung.* PhD thesis, 2008.

[6] Gang Chen, Tian-jiang Wang, Li-yu Gong, and Perfecto Herrera. Multi-class support vector machine active learning for music annotation. *International Journal of Innovative Computing, Information and Control*, 6(3):921–930, 2010.

[7] Husheng Guo and Wenjian Wang. An active learning-based svm multi-class classification model. *Pattern Recognition*, 48(5):1577–1597, 2015.

[8] Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov, editors. *Active Learning Challenge*, volume 6 of *Challenges in Machine Learning*. Microtome Publishing, 2011.

[9] Yaroslav O Halchenko and Michael Hanke. Open is not enough. let's take the next step: an integrated, community-driven computing platform for neuroscience. *Frontiers in neuroinformatics*, 6, 2012.

[10] Tianxu He, Shukui Zhang, Jie Xin, Pengpeng Zhao, Jian Wu, Xuefeng Xian, Chunhua Li, and Zhiming Cui. An active learning approach with uncertainty, representativeness, and diversity. *The Scientific World Journal*, 2014, 2014.

[11] Daniel Kottke, Georg Krempl, Dominik Lang, Johannes Teschner, and Myra Spiliopoulou. Multi-class probabilistic active learning. In *ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 586 – 594, 2016.

[12] Jan Kremer, Kim Steenstrup Pedersen, and Christian Igel. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):313–326, 2014.

[13] Yan Leng, Xinyan Xu, and Guanghui Qi. Combining active learning and semi-supervised learning to construct svm classifier. *Knowledge-Based Systems*, 44:121–131, 2013.

[14] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, pages 1–32, 2003.

[15] Pabitra Mitra, CA Murthy, and Sankar K Pal. A probabilistic active support vector learning algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(3):413–418, 2004.

[16] Nir Nissim, Mary Regina Boland, Robert Moskovitch, Nicholas P Tatonetti, Yuval Elovici, Yuval Shahar, and George Hripcsak. An active learning framework for efficient condition severity classification. In *Artificial Intelligence in Medicine*, pages 13–24. Springer, 2015.

[17] Nir Nissim, Robert Moskovitch, Lior Rokach, and Yuval Elovici. Detecting unknown computer worm activity via support vector machines and active learning. *Pattern Analysis and Applications*, 15(4):459–475, 2012.

[18] Thomas Osugi, Deng Kim, and Stephen Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[19] 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.

[20] Tobias Reitmaier and Bernhard Sick. Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4ds. *Information Sciences*, 230:106 – 131, 2013. Mobile and Internet Services in Ubiquitous and Pervasive Computing Environments.

[21] Matthias Rupp. Machine learning for quantum mechanics in a nutshell. *International Journal of Quantum Chemistry*, 115(16):1058–1073, 2015.

[22] Burr Settles. *Active Learning*. Number 18 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[23] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.