Article type: Advanced Review

Active Learning with Support Vector Machines **Article ID**

Jan Kremer Kim Steenstrup Pedersen Christian Igel

University of Copenhagen

Keywords

active learning, SVM, support vector machine, version space, uncertainty sampling

Abstract

In machine learning, *active learning* refers to algorithms that autonomously select the data points from which they will learn. There are many data mining applications in which large amounts of unlabeled data are readily available, but labels (e.g., human annotations or results from complex experiments) are costly to obtain. In such scenarios, an active learning algorithm aims at identifying data points that, if labeled and used for training, would most improve the learned model. Labels are then obtained only for the most promising data points. This speeds up learning and reduces labeling costs. Support vector machine (SVM) classifiers are particularly well-suited for active learning due to their convenient mathematical properties. They perform linear classification, typically in a kernel-induced feature space, which makes measuring the distance of a data point from the decision boundary straightforward. Furthermore, heuristics can efficiently estimate how strongly learning from a data point influences the current model. This information can be used to actively select training samples. After a brief introduction to the active learning problem, we discuss different query strategies for selecting informative data points and review how these strategies give rise to different variants of active learning with SVMs.

Introduction

In many applications of supervised learning in data mining, huge amounts of unlabeled data samples are cheaply available while obtaining their labels for training a classifier is costly. To minimize labeling costs, we want to request labels only for potentially informative samples. These are usually the ones that we expect to improve the accuracy of the classifier to the greatest extent when used for training. Another consideration is the reduction of training time. Even when all samples are labeled, we may want to consider only a subset of the available data because training the classifier of choice using all the data might

be computationally too demanding. Instead of sampling a subset uniformly at random, which is referred to as *passive learning*, we would like to select informative samples to maximize accuracy with less training data. *Active learning* denotes the process of autonomously selecting promising data points to learn from. By choosing samples actively, we introduce a selection bias. Thus, we violate the assumption underlying most learning algorithms that training and test data are identically distributed. This must be addressed if we want to avoid detrimental effects on the generalization performance.

In theory, active learning is possible with any classifier that is capable of passive learning. This review focuses on the support vector machine (SVM) classifier. It is a state-of-the-art method, which has proven to give highly accurate results in the passive learning scenario and which has some favorable properties that make it especially suitable for active learning: (i) SVMs learn a linear decision boundary, typically in a kernel-induced feature space. Measuring the distance of a sample to this boundary is straightforward and provides an estimate of its informativeness. (ii) Efficient online learning algorithms make it possible to obtain a sufficiently accurate approximation of the optimal SVM solution without retraining on the whole dataset. (iii) The SVM can weight the influence of single samples in a simple manner. This allows for compensating the selection bias that active learning introduces.

Active Learning

In the following we focus on supervised learning for classification. There also exists a body of work on active learning with SVMs for other settings, such as regression[15] or ranking.[8;48] A discussion of these is, however, beyond the scope of this article.

The training set is given by $\mathcal{L} = \{(x_1, y_1), ..., (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathcal{Y}$. It consists of ℓ labeled samples that are drawn independently from an unknown distribution $\mathcal{D}$. This distribution is defined over $\mathcal{X} \times \mathcal{Y}$, the cross product of a feature space $\mathcal{X}$ and a label space $\mathcal{Y}$, with $\mathcal{Y} = \{-1, 1\}$ in the binary case. We try to infer a hypothesis $f : \mathcal{X} \to \mathcal{Z}$ mapping inputs to a prediction space $\mathcal{Z}$ for predicting the labels of samples drawn from $\mathcal{D}$. To measure the quality of our prediction, we define a loss function $L : \mathcal{Z} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$. Thus, our learning goal is minimizing the expected loss

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[L(f(x), x, y)\Big] \ , \tag{1}$$

which is called the *risk* of f. We call the average loss over a finite sample $\mathcal{L}$ the *training error* or *empirical risk*. If a loss function does not depend on the second argument, we simply omit it.

In sampling-based active learning, there are two scenarios: stream-based and pool-based. In *stream-based active learning*, we analyze incoming unlabeled samples sequentially, one sample at a time. Contrary, in *pool-based active learning* we have access to a pool of unlabeled samples at once. In this case, we can rank samples based on

a selection criterion and query the most informative ones. Although, some of the methods in this review are also applicable to stream-based learning, most of them consider the pool-based scenario. In the case of pool-based active learning, we have, in addition to the labeled set $\mathcal{L}$, access to a set of m unlabeled samples $\mathcal{U} = \{x_{\ell+1}, ..., x_{\ell+m}\}$. We assume that there exists a way to provide us with a label for any sample from this set (the probability of the label is given by $\mathcal{D}$ conditioned on the sample). This may involve labeling costs, and the number of queries we are allowed to make may be restricted by a budget. After labeling a sample, we simply add it to our training set.

In general, we aim at achieving a minimum risk by requesting as few labels as possible. We can estimate this risk by computing the average error over an independent test set not used in the training process. Ultimately, we hope to require less labeled samples for inferring a hypothesis performing as well as a hypothesis generated by passive learning on $\mathcal{L}$ and completely labeled $\mathcal{U}$.

In practice, one can profit from an active learner if only few labeled samples are available and labeling is costly or where using the whole labeled data is computationally infeasible. A list of real-world applications is given in the general active learning survey[39] and in a review paper which considers active learning in the context of natural language processing.[29]

Active learning might also be employed to benefit transfer learning.[41] In this setting, samples from the unlabeled target domain are selected for labeling and included in the source domain. A classifier which is trained on the augmented source dataset can then utilize the added samples to increase its accuracy in the target domain. This technique was used successfully, for example, in an astronomy application.[34] There it enabled overcoming a so-called *sample selection bias*, which causes source and target probability distributions to mismatch.[33]

Support Vector Machine

Support vector machines (SVMs) are state-of-the-art classifiers.[6;12;38;40] They have proven to provide well-generalizing solutions in practice and are well understood theoretically.[42] The *kernel trick*[38] allows for an easy handling of diverse data representations (e.g., biological sequences or multimodal data). Support vector machines perform linear discrimination in a kernel-induced feature space and are based on the idea of large margin separation: they try to maximize the distance between the decision boundary and the correctly classified points closest to this boundary. In the following, we formalize SVMs to fix our notation, for a detailed introduction we refer to the recent WIREs articles.[26;36]

An SVM for binary classification labels an input x according to the sign of a decision function of the form

$$f(x) = \langle \mathbf{w}, \phi(x) \rangle + b = \sum_{i=1}^{\ell} \alpha_i y_i \kappa(x_i, x) + b \ , \tag{2}$$

where κ is a positive semi-definite kernel function[1] and $\phi(x)$ is a mapping $\mathcal{X} \to \mathcal{F}$ to a kernel-induced Hilbert space $\mathcal{F}$ such that $\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. We call $\mathcal{F}$ the feature space, which includes the weight vector $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \phi(x_i) \in \mathcal{F}$. The training patterns x_i with $\alpha_i > 0$ are called *support vectors*. The decision boundary is linear in $\mathcal{F}$ and the offset from the origin is controlled by $b \in \mathbb{R}$.

The distance of a pattern (x, y) from the decision boundary is given by $|f(x)/\|\mathbf{w}\||$. We call $yf(x)$ the *functional margin* and $yf(x)/\|\mathbf{w}\|$ the *geometric margin*: a positive margin implies correct classification. Let us assume that the training data $\mathcal{L}$ is linearly separable in $\mathcal{F}$. Then $m(\mathcal{L}, f) = \min_{(x,y)\in\mathcal{L}} yf(x)$ defines the margin of the whole data set $\mathcal{L}$ with respect to f (in the following we do not indicate the dependency on $\mathcal{L}$ and f if it is clear from the context). We call the feature space region $\{x \in \mathcal{X} \mid |f(x)| \leq 1\}$ the *margin band*.[9]

A *hard margin SVM* computes the linear hypothesis that separates the data and yields a maximum margin by solving

$$\begin{aligned} \max_{\mathbf{w},b,\gamma} \quad & \gamma \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \phi(x_i) \rangle + b) \geq \gamma \quad , \quad i = 1, \ldots, \ell \\ & \|\mathbf{w}\| = 1 \end{aligned} \tag{3}$$

with $\mathbf{w} \in \mathcal{F}$, $b \in \mathbb{R}$ and $\gamma \in \mathbb{R}$.[40] Instead of maximizing γ and keeping the norm of $\mathbf{w}$ fixed to one, one can equivalently minimize $\|\mathbf{w}\|$ and fix a *target margin*, typically $\gamma = 1$.

In general, we cannot or do not want to separate the full training data correctly. *Soft margin SVMs* mitigate the concept of large margin separation. They are best understood as the solutions of the regularized risk minimization problem

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} C_i L_{\text{hinge}}(\langle \mathbf{w}, \phi(x_i) \rangle + b, y_i) \ . \tag{4}$$

Here, $L_{\text{hinge}}(f(x_i), y_i) = \max(0, 1 - y_i f(x_i))$ denotes the *hinge loss*. An optimal solution $\mathbf{w}^* = \sum_{i=1}^{\ell} \alpha_i^* y_i \phi(x_i)$ has the property that $0 \leq \alpha_i^* \leq C_i$ for $i = 1, \ldots, \ell$. For soft margin SVMs, the patterns in $\mathcal{L}$ need not be linearly separable in $\mathcal{F}$. If they are, increasing the C_i until an optimal solution satisfies $\alpha_i^* < C_i$ for all i leads to the same hypotheses as training a hard margin SVM. Usually, all samples are given the same weight $C_i = C$.

Uncertainty Sampling

It seems to be intuitive to query labels for samples that cannot be easily classified using our current classifier. Consider the contrary: if we are very certain about the class of a sample, then we might regard any label that does not reflect our expectation as noise.

On the other hand, uncovering an expected label would not make us modify our current hypothesis.

Uncertainty sampling was introduced by Lewis and Gale.[23] The idea is that the samples the learner is most uncertain about provide the greatest insight into the underlying data distribution. Figure 1 shows an example in the case of an SVM. Among the three different unlabeled candidates, our intuition may suggest to ask for the label of the sample closest to the decision boundary: the labels of the other candidates seem to clearly match the class of the samples on the respective side or are otherwise simply mislabeled. In the following, we want to show how this intuitive choice can be justified and how it leads to a number of active learning algorithms that make use of the special properties of an SVM.

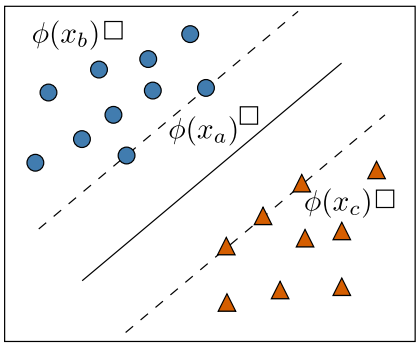


Figure 1: The three rectangles depict unlabeled samples while the blue circles and orange triangles represent positively and negatively labeled samples, respectively. Intuitively, the label of the sample x_a might tell us the most about the underlying distribution of labeled samples, since in the feature space, $\phi(x_a)$ is closer to the decision boundary than $\phi(x_b)$ or $\phi(x_c)$.

Version Space

The *version space* is a construct that helps to keep track of all hypotheses that are able to perfectly classify our current observations.[27] Thus, for the moment, we assume that our data are linearly separable in the feature space. The idea is to speed up learning by selecting samples in a way that minimizes the version space rapidly with each labeling.

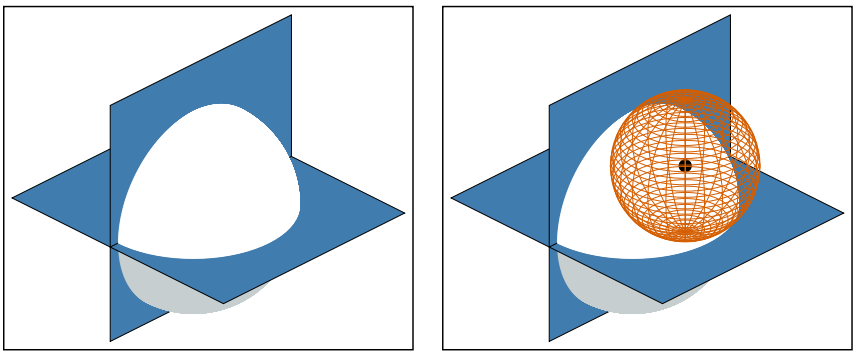
We can express the hard margin SVM classifier (3) in terms of a geometric representation of the version space. For this purpose, we restrict our consideration to hypotheses $f(x) = \langle \mathbf{w}, \phi(x) \rangle$ without bias (i.e., $b = 0$). The following results, however, can be extended to SVMs with $b \neq 0$.

The version space $\mathcal{V}(\mathcal{L})$ refers to the subset of $\mathcal{F}$ that includes all hypotheses consistent with the training set $\mathcal{L}$:[27]

$$\mathcal{V}(\mathcal{L}) := \left\{ \mathbf{w} \in \mathcal{F} \mid \|\mathbf{w}\| = 1,\ yf(x) > 0, \forall (x, y) \in \mathcal{L} \right\} \tag{5}$$

In this representation, we can interpret the hypothesis space as the unit hypersphere given by $\|\mathbf{w}\| = 1$. The surface of the hypersphere includes all possible hypotheses classifying samples that are mapped into the feature space $\mathcal{F}$. We define $\Lambda(\mathcal{V})$ as the area the version space occupies on the surface of this hypersphere. This is depicted in Figure 2. The hypothesis space is represented by the big sphere. The white front, which is cut out by the two hyperplanes, depicts the version space.

Each sample x can be interpreted as defining a hyperplane through the origin of $\mathcal{F}$ with the normal vector $\phi(x)$. Each hyperplane divides the feature space into two half-spaces. Depending on the label y of the sample x, the version space is restricted to the surface of the hypersphere that lies on the respective side of the hyperplane. For example, a sample x that is labeled with $y = +1$, restricts the version space to all $\mathbf{w}$ on the unit hypersphere for which $\langle \mathbf{w}, \phi(x) \rangle > 0$, i.e., the ones that lie on the positive side of the hyperplane defined by the normal vector $\phi(x)$. Thus, the version space is defined by the intersection of all half-spaces and the surface of the hypothesis hypersphere. Figure 2a illustrates this geometric relationship.



(a) The sphere depicts the hypothesis space. The two hyperplanes are induced by two labeled samples. The version space is the part of the sphere surface (in white) that is on one side of each hyperplane. The respective side is defined by its label.

(b) The center (in black) of the orange sphere depicts the SVM solution within the version space. It has the maximum distance to the hyperplanes delimiting the version space. The normals of these hyperplanes, which are touched by the orange sphere, correspond to the support vectors.

Figure 2: Geometric representation of the version space in 3D following Tong.[43]

If we consider a feature mapping with the property $\forall x, z \in \mathcal{X} : \|\phi(x)\| = \|\phi(z)\|$, such as normalized kernels, including the frequently used Gaussian kernel, then the SVM solution (3) has a particularly nice geometric interpretation in the version space. Under this condition, the decision function $f(x) = \langle \mathbf{w}, \phi(x) \rangle$ maximizing the margin $m(\mathcal{L}, f)$ (i.e., the minimum $y\langle \mathbf{w}, \phi(x) \rangle$ over $\mathcal{L}$) also maximizes the minimum distance between $\mathbf{w}$ and any hyperplane defined by a normal $\phi(x_i)$, $i = 1, \ldots, \ell$. The solution is a point within the version space, which is the center of a hypersphere, depicted in orange in Figure 2b. This hypersphere yields the maximum radius possible without intersecting the hyperplanes that delimit the version space. The radius is given by

$r = \frac{y\langle \mathbf{w}, \phi(x) \rangle}{\|\phi(x)\|}$, where $\phi(x)$ is any support vector. Changing our perspective, we can interpret the normals $\phi(x_i)$ of the hyperplanes touching the hypersphere as points in the feature space $\mathcal{F}$. Then, these are exactly the support vectors since they have the minimum distance to our decision boundary defined by $\mathbf{w}$. This distance is $m(\mathcal{L}, f)$, which turns out to be proportional to the radius r.

Implicit Version Space

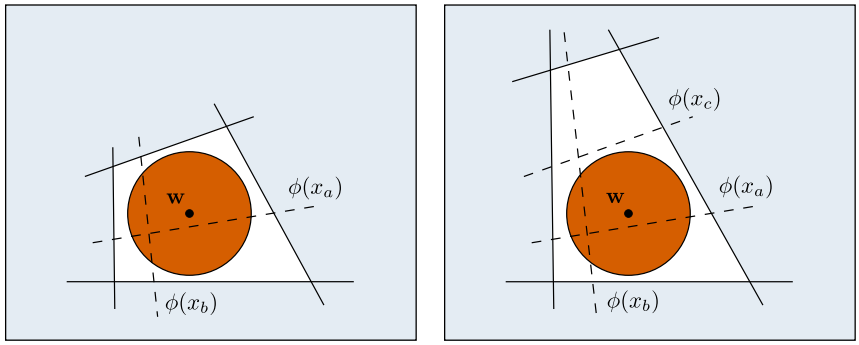
An explicit version space, as defined above, only exists if the data are separable, which is often not the case in practice. The Bayes optimal solution need not have vanishing training error (as soon as under $\mathcal{D}$ we have $p(y_1|x) \geq p(y_2|x) > 0$ for some $x \in \mathcal{X}$ and $y_1, y_2 \in \mathcal{Y}$ with $y_1 \neq y_2$). Thus, minimizing the version space might exclude hypotheses with non-zero training error that are in fact optimal. In agnostic active learning,[2] we do not make the assumption of an existing optimal zero-error decision boundary. An algorithm that is theoretically capable of active learning in an agnostic setting is the A^2-algorithm.[2] Here, a hypothesis cannot be deleted due to its disagreement with a single sample. If, however, all hypotheses that are part of the current version space agree on a region within the feature space, this region can be discarded. For each hypothesis the algorithm keeps an upper and a lower bound on its training error (see the work by Balcan et al.[2] for details). It subsequently excludes all hypotheses which have a lower bound that is higher than the global minimal upper bound. Despite being intractable in practice, this algorithm forms the basis of some important algorithms compensating the selection bias as discussed below.

Uncertainty-Based Active Learning

Although the version space is restricted to separable problems, it motivates many general active selection strategies. Which samples should we query to reduce the version space? As we have seen previously, each labeled sample that becomes a support vector restricts the version space to one side of the hyperplane it induces in $\mathcal{F}$. If we do not know the correct label of a sample in advance, we should always query the sample that ideally halves the version space. This is a safe choice as we will reduce it regardless of the label. Computing the version space in a high-dimensional feature space is usually intractable, but we can approximate it efficiently using the SVM. In the version space, the SVM solution $\mathbf{w}$ is the center of the hypersphere touching the hyperplanes induced by the support vectors. Each hyperplane delimits the version space. Assuming that the center of this hypersphere is close to the center of the version space, we can use it as an approximation. If we now choose a hyperplane that is close to this center, we approximately bisect the version space. Therefore, we want to query the sample $\hat{x}$ that induces a hyperplane as close to $\mathbf{w}$ as possible:

$$\hat{x} = \underset{x \in \mathcal{U}}{\operatorname{argmin}} \, |\langle \mathbf{w}, \phi(x) \rangle| = \underset{x \in \mathcal{U}}{\operatorname{argmin}} \, |f(x)| \tag{6}$$

This strategy queries the sample closest to the current decision boundary and is called *Simple Margin*.[43] Figure 3 shows this principle geometrically, projected to two dimensions.



(a) *Simple Margin* will query the sample that induces a hyperplane lying closest to the SVM solution. In this case, it would query sample x_a.

(b) Here the SVM does not provide a good approximation of the version space area. *Simple Margin* would query sample x_a while x_c might have been a more suitable choice.

Figure 3: The version space area is shown in white, the solid lines depict the hyperplanes induced by the support vectors, the center of the orange circle is the weight vector $\mathbf{w}$ of the current SVM. The dotted lines show the hyperplanes that are induced by unlabeled samples. This visualization is inspired by Tong.[43]

By querying the samples closest to the separating hyperplane, we try to minimize the version space by requesting as few labels as possible. However, depending on the actual shape of the version space, the SVM solution may not provide a good approximation and another query strategy would have achieved a greater reduction of the version space area. This is illustrated in Figure 3b. The strategy of myopically querying the samples with the smallest margin may even perform worse than a passive learner.

Therefore, we can choose a different heuristic to approximate the version space more accurately.[37;45] For instance, we could compute two SVMs for each sample: one for the case we labeled it positively and one assuming a negative label. We can then, for each case, compute the margins $m^+ = +\langle \mathbf{w}, \phi(x) \rangle$ and $m^- = -\langle \mathbf{w}, \phi(x) \rangle$. Finally, we query the sample which gains the maximum value for $\min(m^+, m^-)$. This quantity will be very small if the corresponding version spaces are very different. Thus, we take the maximum to gain an equal split. This strategy is called *MaxMin Margin*[43] and allows us to make a better choice in case of an irregular version space area, as we can see in Figure 4. This, however, comes with the additional costs of computing the margin for each potential labeling.

Uncertainty sampling can also be motivated by trying to minimize the training error directly.[9] Depending on the assumptions made, we can arrive at different strategies. Considering the classifier has just been trained on few labeled data, we assume the prospective labels of the yet unlabeled data to be uncorrelated with the predicted labels.

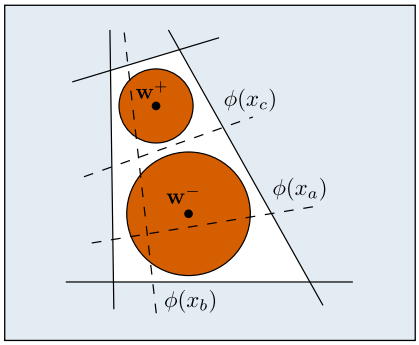


Figure 4: In this case, the *MaxMin Margin* strategy would query sample x_c. Each of the two orange circles correspond to an SVM trained with a positive and a negative labeling of x_c.[43]

Therefore, we want to select the sample for which we can expect the largest error, namely

$$\hat{x} = \underset{x \in \mathcal{U}}{\operatorname{argmax}} \frac{1}{2}\Big[\max(0, 1 - f(x)) + \max(0, 1 + f(x))\Big] \ , \tag{7}$$

where we assume the hinge loss and $f(x)$ as defined in (2). Assuming a separable dataset, we are only interested in uncertain samples, i.e., those within the margin band. Under these constraints, any choice of x leads to the same value of the objective (7): we select the sample at random in this case. However, as soon as some labeled samples are available for SVM training, the prediction of the SVM for an unlabeled point x is expected to be positively correlated with the label of x. Thus, we assume correct labeling and look for each sample at the minimum error we gain regardless of the labeling. We want to find the sample that maximizes this quantity, i.e.,

$$\hat{x} = \underset{x \in \mathcal{U}}{\operatorname{argmax}} \min\Big\{\max(0, 1 - f(x)), \max(0, 1 + f(x))\Big\} = \underset{x \in \mathcal{U}}{\operatorname{argmin}} |f(x)| \ , \tag{8}$$

which gives us the same selection criterion as in (6), the *Simple Margin* strategy. If all unlabeled samples meet the target margin (i.e., the hinge loss of the samples is zero) and if we assume the SVM labels them correctly (i.e., $|f(x)| \geq 1$), it seems that we have arrived at a hypothesis that already generalizes well. Both, picking samples near or far away from the boundary appears to be non-optimal. Therefore, we simply proceed by choosing a random sample from the unlabeled data.

In practice, we may start by training on a random subset and perform uncertainty sampling until the user stops the process or until all unlabeled samples meet the target margin. In this case, we query another random subset as a validation set and estimate the error. We may repeat the last step until we reach a satisfactory solution.

Expected Model Change

Instead of trying to minimize an explicit or implicit version space, we can find the most informative sample by selecting it based on its expected effect on the current model,

the *expected model change*. In gradient-based learning, this means selecting the sample $x \in \mathcal{U}$ that, if labeled, would maximize the expected gradient length, where we take the expectation $\mathbb{E}_y$ over all possible labels y.

Non-linear SVMs are usually trained by solving the underlying quadratic optimization problem in its Wolfe dual representation.[12;38] Let us assume SVMs without bias. If we add a new sample $(x_{\ell+1}, y_{\ell+1})$ to the current SVM solution and initialize its coefficient with $\alpha_{\ell+1} = 0$, the partial derivative with respect to $\alpha_{\ell+1}$ of the dual problem $W(\boldsymbol{\alpha})$ to be maximized is

$$g_{\ell+1} = \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_{\ell+1}} = 1 - y_{\ell+1} \sum_{i=1}^{\ell} \alpha_i y_i \kappa(x_i, x_{\ell+1}) = 1 - y_{\ell+1} f(x_{\ell+1}) \ . \tag{9}$$

As α_i is constrained to be non-negative, we only change the model if the partial derivative is positive, that is, if $y_{\ell+1} f(x_{\ell+1}) < 1$. Note that $y_{\ell+1} f(x_{\ell+1}) > 1$ implies that $(x_{\ell+1}, y_{\ell+1})$ is already correctly classified by the current model and meets the target margin.

Let us assume that our current model classifies any sample perfectly and that the dataset is linearly separable in the feature space:

$$p(y|x) = \begin{cases} 1 & \text{if } y\, f(x) > 0 \\ 0 & \text{otherwise} \end{cases} \ . \tag{10}$$

If we just consider the partial derivative $g_{\ell+1}$ in the expected model change selection criterion, we arrive at selecting

$$\begin{aligned} \hat{x} &= \operatorname*{argmax}_{x \in \mathcal{U}} \Big(p(y=1|x)|1 - f(x)| + p(y=-1|x)|1 + f(x)| \Big) \\ &= \operatorname*{argmax}_{x \in \mathcal{U}} \Big(p(y=1|x)(1 - f(x)) + p(y=-1|x)(1 + f(x)) \Big) \\ &= \operatorname*{argmax}_{x \in \mathcal{U}} \begin{cases} 1 - f(x) & \text{if } f(x) > 0 \\ 1 + f(x) & \text{if } f(x) < 0 \end{cases} = \operatorname*{argmin}_{x \in \mathcal{U}} |f(x)| \ . \end{aligned} \tag{11}$$

Thus, uncertainty sampling can also be motivated by maximizing the expected model change. Next, we want to have a look at approaches that try to exploit the uncertainty of samples and simultaneously explore undiscovered regions of the feature space.

Combining Informativeness and Representativeness

By performing mere uncertainty sampling, we may pay too much attention to certain regions of the feature space and neglect other regions that are more representative of the underlying distribution. This leads to a sub-optimal classifier. To counteract this effect, we could sample close to the decision boundary, but also systematically include samples that are farer away.[18;20]

In Figure 5, we see an example where uncertainty sampling can mislead the classifier. Although $\phi(x_a)$ is closest to the separating hyperplane, it is also far away from all other

samples in feature space and thus may not be representative of the underlying distribution. To avoid querying outliers, one would like to select samples not only based on their informativeness, but also based on representativeness.[13] In our example, selecting sample x_b would be a better choice, because it is located in a more densely populated region where a correct classification is of more importance to gain an accurate model.

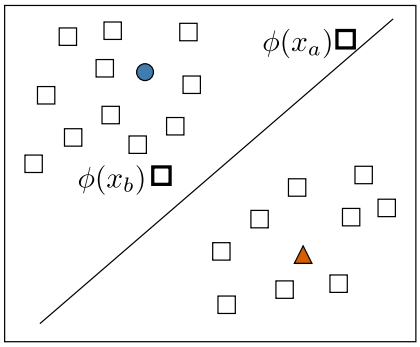
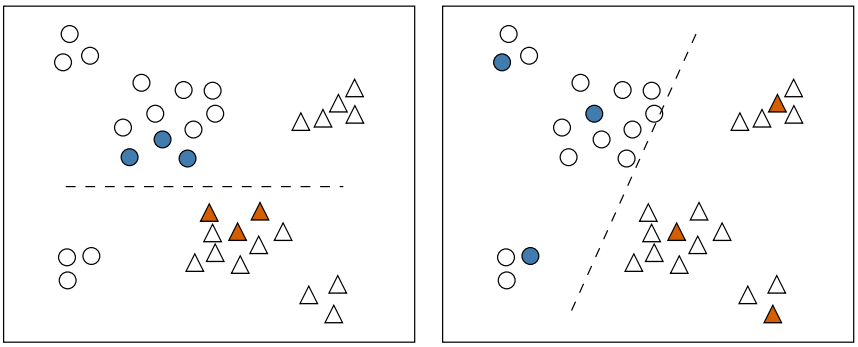


Figure 5: The white rectangles depict unlabeled samples. The blue circle and the orange triangle are labeled as positive and negative, respectively. In feature space, $\phi(x_a)$ lies closer to the separating hyperplane than $\phi(x_b)$, but is located in a region, which is not densely populated. Using pure uncertainty sampling, e.g., *Simple Margin*, we would query the label of sample x_a.

Informativeness is a measure of how much querying a sample would reduce the uncertainty of our model. As we have seen, uncertainty sampling is a viable method to exploit informativeness. Representativeness measures how well a sample represents the underlying distribution of unlabeled data.[39] By using a selection criterion that maximizes both measures, we try to improve our models with less samples than a passive learner while carefully avoiding a model that is too biased. In Figure 6 we can see a comparison of the different strategies using a toy example where we sequentially query six samples. Figure 6a shows a biased classifier as the result of uncertainty sampling. While the solution in Figure 6b is closer to the optimal hyperplane, it also converges slower, as it additionally explores regions where we are relatively certain about the labels. Combining both strategies, as shown in Figure 6c, yields a decision boundary that is close to the optimal (Figure 6d) with fewer labels.

Semi-Supervised Active Learning

To avoid oversampling unrepresentative outliers, we can combine uncertainty sampling and clustering.[47] First, we train an SVM on the labeled samples and then apply k-means clustering to all unlabeled samples within the margin band to identify k groups. Finally, we query the k medoids. As only samples within the margin band are considered, they are all subject to high uncertainty. The clustering ensures that the informativeness of our selection is increased by avoiding redundant samples.



(a) Uncertainty sampling.

(b) Selecting representative samples.

(c) Combining informativeness and representativeness.

(d) Optimal hyperplane, obtained by training on the whole dataset.

Figure 6: Binary classification with active learning on six samples and passive learning on the full dataset.

However, when using clustering, one has to decide what constitutes a cluster.[14] Depending on the scale and the selected number of clusters, different choices could be equally plausible. To avoid this dilemma, we can choose another strategy to incorporate density information.[22] We build on the min-max formulation[21] of active learning and request the sample

$$\hat{x} = \underset{x_s \in \mathcal{U}}{\operatorname{argmin}} \max_{y_s \in \{-1,+1\}} \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{(x,y)\in\mathcal{L}_s} L(f(x), x, y) \tag{12}$$

where $\mathcal{L}_s = \mathcal{L} \cup (x_s, y_s)$. We take the minimum regularized expected risk when including the sample $x_s \in \mathcal{U}$ with the label y_s that yields the maximum error. Selecting the sample $\hat{x}$ minimizing this quantity can be approximated by uncertainty sampling (e.g., using *Simple Margin*).

In this formulation, however, we base our decision only on the labeled samples and do not take into account the distribution of unlabeled samples. Assuming we knew the labels for each sample in $\mathcal{U}$, we define the set $\mathcal{L}_{su}$ containing the labeled samples (x, y)

for $x \in \mathcal{U}$ and the training set $\mathcal{L}$. We select the sample

$$\hat{x} = \underset{x_s \in \mathcal{U}}{\operatorname{argmin}} \min_{\mathbf{y}_u \in \{\pm 1\}^{n_u - 1}} \max_{y_s \in \{-1,+1\}} \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{(x,y) \in \mathcal{L}_{su}} L(f(x), x, y) \ , \tag{13}$$

where $n_u = |\mathcal{U}|$ and $\mathbf{y}_u$ is the label vector assigned to the samples $x \in \mathcal{U}$ without the label for x_s. Thus, we also maximize the representativeness of the selected sample by incorporating the possible labelings of all unlabeled samples. By using a quadratic loss-function and relaxing $\mathbf{y}_u$ to continuous values, we can approximate the solution through the minimization of a convex problem.[22]

Both clustering and label estimation are of high computational complexity. A simpler algorithm, which the authors call *Hinted SVM*, considers unlabeled samples without resorting to these techniques.[24] Instead, the unlabeled samples are taken as so-called hints that inform the algorithm of feature space regions which it should be less confident in. To achieve this, we try to simultaneously find a decision boundary that produces a low training error on the labeled samples while being close to the unlabeled samples, the hints. This can be viewed as semi-supervised learning.[10] It is, however, in contrast to typical semi-supervised SVM approaches that push the decision boundary away from the pool of unlabeled samples.

The performance of this algorithm depends on the hint selection strategy. Using all unlabeled samples might be too costly for large datasets while uniform sampling of the unlabeled pool does not consider the information provided by labeled samples. Therefore, we can start with the pool of all unlabeled samples and iteratively drop instances that are close to already labeled ones. When the ratio of hints to all samples is below a certain threshold, we can switch to uncertainty sampling.

Another problem with uncertainty sampling is that it assumes our current hypothesis to be very certain about regions far from the decision boundary. If this assumption is violated, we will end up with a classifier worse than obtained using passive learning. One way to address this issue is to measure the uncertainty of our current hypothesis and to adjust our query strategy accordingly.[28] To achieve this, we compute a heuristic measure expressing the confidence that the current set of support vectors will not change if we train on more data. It is calculated as

$$c = \frac{2}{|\mathcal{L}_{\text{SV}}| \cdot k} \sum_{(x,y) \in \mathcal{L}_{\text{SV}}} \min(k_x^+, k_x^-) \ , \tag{14}$$

where $\mathcal{L}_{\text{SV}}$ are the support vectors and k_x^+ and k_x^- are the number of positively and negatively labeled samples within the k nearest neighbors of $(x, y) \in \mathcal{L}_{\text{SV}}$. In the extremes, we get $c = 1$ if $k_x^+ = k_x^-$ and $c = 0$ if $k_x^+ = 0 \vee k_x^- = 0$ for all $(x, y) \in \mathcal{L}_{\text{SV}}$. We can use this measure to decide whether a labeled data point (x, y) should be kept for training by adding it with probability

$$p(x) = \begin{cases} c & \text{if } yf(x) \leq 1 \\ 1 - c & \text{otherwise.} \end{cases} \tag{15}$$

to the training data set. This means that more samples are queried within the margin if we are very confident that the current hyperplane represents the optimal one. In the following, we discuss a related idea, which not only queries samples with a certain probability, but also subsequently incorporates this probability to weight the impact of the training set samples.

Importance-Weighted Active Learning

When we query samples actively instead of selecting them uniformly at random, the training and test samples are not independent and identically distributed (i.i.d.). Thus, the training set will have a sample selection bias. As most classifiers rely on the i.i.d. assumption, this can severely impair the prediction performance.

Assume that we sample the training data points from the biased sample distribution $\tilde{\mathcal{D}}$ over $\mathcal{X} \times \mathcal{Y}$, while our goal is minimizing the risk (1) with respect to the true distribution $\mathcal{D}$. If we know the relationship between $\tilde{\mathcal{D}}$ and $\mathcal{D}$, we can still arrive at an unbiased hypothesis by re-weighting the loss for each sample.[11;49] We introduce the weighted loss $L_w(z, x, y) = w(x, y)L(z, x, y)$ and define the weighting function

$$w(x, y) = \frac{p_{\mathcal{D}}(x, y)}{p_{\tilde{\mathcal{D}}}(x, y)} \tag{16}$$

reflecting how likely it is to observe (x, y) under $\mathcal{D}$ compared to $\tilde{\mathcal{D}}$ under the assumption that the support of $\mathcal{D}$ is included in $\tilde{\mathcal{D}}$. This leads us to the basic result

$$\begin{aligned} \mathop{\mathbb{E}}_{(x,y)\sim\tilde{\mathcal{D}}}\Big[L_w(f(x), x, y)\Big] &= \int_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_{\tilde{\mathcal{D}}}(x, y)\frac{p_{\mathcal{D}}(x, y)}{p_{\tilde{\mathcal{D}}}(x, y)}L(f(x), y)\mathrm{d}(x, y) \\ &= \int_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p_{\mathcal{D}}(x, y)L(f(x), y)\mathrm{d}(x, y) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[L(f(x), x, y)\Big] \ . \end{aligned} \tag{17}$$

Thus, by choosing appropriate weights we can modify our loss-function such that we can compute an unbiased estimator of the generalization error $R(f)$. This technique for addressing the sample selection bias is called *importance weighting*.[3;4]

We define a weighted sample set $\mathcal{L}_w$ as the training set $\mathcal{L}$ augmented with non-negative weights $w_1, \ldots, w_\ell$ for each point in $\mathcal{L}$. These weights are used to set $w(x_i, y_i) = w_i$ when computing the weighted loss. For the soft margin SVM minimizing the weighted loss can easily be achieved by multiplying each regularization parameter C_i in (4) with the corresponding weight, i.e., $C_i = w_i \cdot C$.[49] While the weighting gives an unbiased estimator, it may be difficult to estimate the weights reliably and the variance of the estimator may be very high. Controlling the variance is a crucial problem when using importance weighting.

The original importance-weighted active learning formulation works in a stream-based scenario and inspects one sample x_t at each step $t > 1$.[3] Iteration t of the algorithm works as follows:

1. Receive the unlabeled sample x_t.
2. Choose $p_t \in [0, 1]$ based on all information available in this round.
3. With probability p_t, query the label y_t for x_t, add (x_t, y_t) to the weighted training set with weight $w_t = 1/p_t$, and retrain the classifier.

In step 2 the query probability p_t has to be chosen based on earlier observations: this could be, for instance, the probability that two hypotheses disagree on the received sample x_t.

This algorithm can also be adapted to the pool-based scenario.[16] In this case, we can simply define a probability distribution over all unlabeled samples in the pool. We set the probability for each point in proportion to its uncertainty, i.e., its distance to the decision boundary. This works well if we assume a noise-free setting. Otherwise, this method suffers from the same problems as other approaches that are based on a version space. Given a mislabeled sample (i.e., the label has a very low probability given the features), the active learner can be distracted and focus on regions within the hypothesis space which do not include the optimal decision boundary.

One way to circumvent these problems is to combine importance weighting with ideas from agnostic active learning.[50] We keep an ensemble of SVMs $\mathcal{H} = \{f_1, ..., f_K\}$ and train each on a bootstrap sample subset from $\mathcal{L}$, which may be initialized with a random subset of the unlabeled pool $\mathcal{U}$ for which labels are requested. After the initialization, we choose points $x \in \mathcal{U}$ with selection probability

$$p_t(x) = p_{\text{threshold}} + (1 - p_{\text{threshold}})(p_{\text{max}}(x) - p_{\text{min}}(x)) \ , \tag{18}$$

where $p_{\text{threshold}} > 0$ is a small minimum probability to ensure that $p_t(x) > 0$. Using Platt's method,[31] we define $p_i(x) \in [0, 1]$ to be the probabilistic interpretation of a SVM f_i with $p_{\text{min}} = \min_{1 \leq i \leq K} p_i(x)$ and $p_{\text{max}} = \max_{1 \leq i \leq K} p_i(x)$. Thus, p_t is high if there is a strong disagreement within the ensemble and low if all classifiers agree. This allows to deal with noise, because no hypothesis gets excluded forever.

Multi-Class Active Learning

The majority of research on active learning with SVMs focuses on the binary case, because dealing with more categories makes estimating the uncertainty of a sample more difficult. Furthermore, multi-class SVMs are in general more time consuming to train. There are different approaches to extend SVMs to multi-class classification. A popular way is to reduce the learning task to multiple binary problems. This is done by using either a one-vs-one[19;32] or a one-vs-all[35;46] approach. However, performing uncertainty sampling with respect to each single SVM may cause the problem that one sample is informative for one binary task, but bears little information for the other tasks and, thus, for the overall multi-class classification.

It is possible to extend the version space minimization strategy to the multi-class case.[43] The area of the version space is proportional to the probability of classifying the training set correctly, given a hypothesis sampled at random from the version space. In the one-vs-all approach for N classes, we maintain N binary SVM models $f_1, \ldots, f_N$ where the ith SVM model is trained to separate class i from the other classes. We consider minimizing the maximum product of all N version space areas to select

$$\hat{x} = \underset{x \in \mathcal{U}}{\operatorname{argmin}} \max_{y \in \{-1,1\}} \prod_{i=1}^{N} \Lambda(\mathcal{V}_{x,y}^{(i)}) \tag{19}$$

where $\Lambda(\mathcal{V}_{x,y}^{(i)})$ is the area of the version space of the i-th SVM if the sample $(x, y) : x \in \mathcal{U}$ was included in the training set. To approximate the area of the version space, we can use *MaxMin Margin* for each binary SVM. The margin of a sample in a single SVM only reflects the uncertainty with respect to the specific binary problem and not in relation to the other classification problems. Therefore, we have to modify our approximation if we want to extend the *Simple Margin* strategy to the multi-class case.

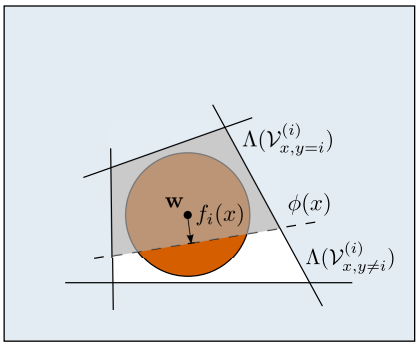


Figure 7: Single version space for the multi-class problem with N one-vs-all SVMs.[43] The area $\Lambda(\mathcal{V}_{x,y=i}^{(i)})$ corresponds to the version space area if the label $y = i$ for the sample we picked. The area $\Lambda(\mathcal{V}_{x,y\neq i}^{(i)})$ corresponds to the case where $y \neq i$. In the multi-class case, we want to measure both quantities to approximate the version space area.

We can interpret each $f_i(x)$ as a quantity that measures to what extent x splits the version space. As we have N different classifiers that influence the area of the version space, we have to quantify this influence for each one. In Figure 7, we can see the version space for one single binary problem where we want to discriminate between class i and the rest. Thus, we want to approximate the area $\Lambda(\mathcal{V}_{x,y=i}^{(i)})$. If we choose a sample x where $f_i(x) = 0$, we approximately halve the version space, for $f_i(x) = 1$, the area approximately stays the same and for $f_i(x) = -1$, we gain a zero area;

similarly for the area $\Lambda(\mathcal{V}^{(i)}_{x,y\neq i})$. Therefore, we can use the approximation

$$\Lambda(\mathcal{V}^{(i)}_{x,y}) = \begin{cases} 0.5 \cdot (1 + f_i(x)) \cdot \Lambda(\mathcal{V}^{(i)}) & \text{if } y = i \\ 0.5 \cdot (1 - f_i(x)) \cdot \Lambda(\mathcal{V}^{(i)}) & \text{if } y \neq i \end{cases} . \tag{20}$$

We can also employ one-versus-one multi-class SVMs.[25] Again, we use Platt's algorithm[31] to approximate posterior probabilities for our predictions. We simultaneously fit the probabilistic model and the SVM hyperparameters via grid-search to derive a classification probability $p_k(x)$, $k = 1, \ldots, K$, for each of the K SVMs given the sample x. We can use these probabilities for active sample selection in different ways. A simple approach is to just select the sample with the least classification confidence. This corresponds to the selection criterion

$$\hat{x}_{\text{LC}} = \operatorname*{argmin}_{x \in \mathcal{U}} \min_{k \in \{1,\ldots,K\}} p_k(x) \ . \tag{21}$$

This approach suffers from the same problem we mentioned earlier: the probability is connected only to each single binary problem instead of providing a measure that relates it to the other classifiers. To alleviate this, we can choose another approach, called *breaking ties*.[25] Here, we select the sample with the minimum difference between the highest class confidences, namely

$$\hat{x}_{\text{BT}} = \operatorname*{argmin}_{x \in \mathcal{U}} \min_{k,l \in \{1,\ldots,K\}, k \neq l} p_k(x) - p_l(x) \ . \tag{22}$$

This way, we prefer samples which two classifiers claim to be certain about, and thus, avoid considering the uncertainty of one classifier in an isolated manner.

Efficient Active Learning

In passive learning, selecting a training set comes almost for free, we just select a random subset of the labeled data. In active learning, we have to evaluate whether a sample should be added to the training set based on its ability to improve our prediction. A simple strategy to reduce computational complexity is to not only select single samples, but to collect them in batches instead. However, to profit from this strategy, we have to make sure to create batches that minimize redundancy.

Another consideration that makes efficient computation necessary is that for many algorithms we have to retrain our model on different training subsets. Usually, these subsets differ only by one or the few samples we consider for selection. Thus, we can employ online learning to train our model incrementally. In particular, this makes sense if we analyze the effect that adding a sample has on our model.

Online Learning

After we have selected a sample to be included in the training set, we have to retrain our model to reflect the additional data. Using the whole dataset for retraining is computationally expensive. A better option would be to incrementally improve our model with

each selected sample through online learning. LASVM[5;17] is an SVM solver for fast online training. It is based on a decomposition method[30] solving the learning problem iteratively by considering only a subset of the α-variables in each iteration. In LASVM, this subset considers one unlabeled sample (corresponding to a new variable) in every second iteration, which can, for instance, be picked by uncertainty sampling.[5]

Batch-Mode Active Learning

When confronted with large amounts of unlabeled data, estimating the effect of single samples with respect to the learning objective is costly. Besides online learning, we can also gain a speed-up by labeling samples in batches. A naive strategy is to just select the n samples that are closest to the decision boundary.[44] This approach, however, does not take into account that the samples within the batch might bear a high level of redundancy.

To counteract this redundancy, we can select samples not only due to their individual informativeness, but also if they maximize the diversity within each batch.[7] One heuristic is to maximize the angle between the hyperplanes that the samples induce in version space:

$$|\cos(\angle(\phi(x_i), \phi(x_j))| = \frac{|\langle \phi(x_i), \phi(x_j) \rangle|}{\|\phi(x_i)\| \|\phi(x_j)\|} = \frac{|\kappa(x_i, x_j)|}{\sqrt{\kappa(x_i, x_i)\kappa(x_j, x_j)}} \tag{23}$$

Let $\mathcal{S}$ be the batch of samples, which we initialize with one sample $x_{\mathcal{S}}$. We subsequently add the sample $\hat{x}$ whose corresponding hyperplane minimizes the maximum angle between any other hyperplane induced by a sample in the batch. It is computed as

$$\hat{x} = \operatorname*{argmin}_{x \in \mathcal{U} \setminus \mathcal{S}} \max_{z \in \mathcal{S}} |\cos(\angle(\phi(x), \phi(z))| \ . \tag{24}$$

We can form a convex combination of this diversity measure with the well-known uncertainty measure (distance to the hyperplane) with trade-off parameter $\lambda \in [0, 1]$. Then, samples that should be added to the batch are iteratively chosen as

$$\hat{x} = \operatorname*{argmin}_{x \in \mathcal{U} \setminus S} \left(\lambda |f(x)| + (1 - \lambda) \max_{z \in S} |\cos(\angle(\phi(x), \phi(z))| \right) \ . \tag{25}$$

We can choose $\lambda = 0.5$ to give equal weight to the uncertainty and diversity measure. An optimal value, however, might depend on how certain we are about the accuracy of the current classifier.

Conclusion

Access to unlabeled data allows us to improve predictive models in data mining applications. If it is only possible to label a limited amount of the available data due to labeling costs, we should choose this subset carefully and focus on patterns carrying the information most helpful to enhance the model. Support

vector machines (SVMs) have convenient properties that make it easy to evaluate how unlabeled samples would influence the model if they were labeled and included in the training set. Therefore, SVMs are particularly well-suited for active learning. However, there are several challenges we have to address, such as efficient learning, dealing with multiple classes, and that actively choosing the training data introduces a selection bias. Importance weighting seems to be most promising to counteract this bias, and it can be easily incorporated into an active SVM learner. Devising parallel algorithms for sample selection can speed up learning in many cases. Most of the research in active SVM learning so far has focused on binary decision problems. A challenge for future research is to develop efficient active learning algorithms for multi-class SVMs that address the nature of the multi-class decision in a more principled way.

Acknowledgments

The authors gratefully acknowledge support from The Danish Council for Independent Research (FNU 12-125149).

References

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[2] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 65–72. ACM Press, 2006.

[3] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 49–56. ACM Press, 2009.

[4] A. Beygelzimer, J. Langford, Z. Tong, and D. Hsu. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems (NIPS)*, pages 199–207. MIT Press, 2010.

[5] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research (JMLR)*, 6:1579–1619, 2005.

[6] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Workshop on Computational Learning Theory (COLT)*, pages 144–152. ACM Press, 1992.

[7] K. Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 59–66. AAAI Press, 2003.

[8] K. Brinker. Active learning of label ranking functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 129–136. ACM Press, 2004.

[9] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 111–118. Morgan Kaufmann, 2000.

[10] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[11] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In N. Bshouty, G. Stoltz, N. Vayatis, and T. Zeugmann, editors, *Algorithmic Learning Theory*, pages 38–53. Springer, 2008.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[13] S. Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 412(19):1767–1781, 2011.

[14] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 208–215. ACM Press, 2008.

[15] B. Demir and L. Bruzzone. A novel active learning method for support vector regression to estimate biophysical parameters from remotely sensed images. In *Proceedings of SPIE Image and Signal Processing for Remote Sensing*. International Society for Optics and Photonics, 2012.

[16] R. Ganti and A. Gray. Upal: Unbiased pool based active learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 422–431, 2012.

[17] T. Glasmachers and C. Igel. Second-order SMO improves SVM online and active learning. *Neural Computation*, 20(2):374–382, 2008.

[18] I. Guyon, G. Cawley, G. Dror, and V. Lemaire. Results of the active learning challenge. *Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings*, 16:19–45, 2011.

[19] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *Annals of Statistics*, 26(2):451–471, 1998.

[20] C.-H. Ho, M.-H. Tsai, and C.-J. Lin. Active learning and experimental design with SVMs. *Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings*, 16:71–84, 2011.

[21] S. Hoi, R. Jin, J. Zhu, and M. Lyu. Semi-supervised SVM batch mode active learning for image retrieval. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2008.

[22] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems (NIPS)*, pages 892–900. MIT Press, 2010.

[23] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12. ACM Press, 1994.

[24] C.-L. Li, C.-S. Ferng, and H.-T. Lin. Active learning with hinted support vector machine. *Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings*, 25:221–235, 2012.

[25] T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 3, pages 478–481. IEEE, 2004.

[26] A. Mammone, M. Turchi, and N. Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289, 2009.

[27] T. M. Mitchell. Generalization as search. *Artificial intelligence*, 18(2):203–226, 1982.

[28] P. Mitra, C. Murthy, and S. Pal. A probabilistic active support vector learning algorithm. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(3):413–418, 2004.

[29] F. Olsson. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science, 2009.

[30] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

[31] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[32] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems (NIPS)*, 12(3):547–553, 2000.

[33] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, 2009.

[34] J. W. Richards, D. L. Starr, H. Brink, A. A. Miller, J. S. Bloom, N. R. Butler, J. B. James, J. P. Long, and J. Rice. Active learning to overcome sample selection bias: Application to photometric variable star classification. *The Astrophysical Journal*, 744(2), 2012.

[35] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research (JMLR)*, 5:101–141, 2004.

[36] S. Salcedo-Sanz, J. L. Rojo-Álvarez, M. Martínez-Ramón, and G. Camps-Valls. Support vector machines in engineering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):234–267, 2014.

[37] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 839–846. Morgan Kaufmann, 2000.

[38] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[39] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[40] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[41] X. Shi, W. Fan, and J. Ren. Actively transfer domain knowledge. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 342–357. Springer, 2008.

[42] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.

[43] S. Tong. *Active learning: Theory and applications*. PhD thesis, Stanford University, 2001.

[44] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the International Conference on Multimedia (MM)*, pages 107–118. ACM Press, 2001.

[45] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research (JMLR)*, 2:45–66, 2002.

[46] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[47] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative sampling for text classification using support vector machines. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 393–407. Springer, 2003.

[48] H. Yu. SVM selective sampling for ranking with application to data retrieval. In *Proceedings of the International Conference on Knowledge Discovery in Data Mining (SIGKDD)*, pages 354–363. ACM Press, 2005.

[49] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 435–442. IEEE, 2003.

[50] L. Zhao, G. Sukthankar, and R. Sukthankar. Importance-weighted label prediction for active learning with noisy annotations. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 3476–3479. IEEE, 2012.

Cross-References

Machine learning, Data mining, Support vector machines