Incremental Learning with Support Vector Machines[*]

Stefan Rüping[**]

[**]Department of Computer Science, AI Unit University of Dortmund, Germany. stefan.rueping@uni-dortmund.de

Abstract. Support Vector Machines (SVMs) have become a popular tool for learning with large amounts of high dimensional data. However, it may sometimes be preferable to learn incrementally from previous SVM results, as computing a SVM is very costly in terms of time and memory consumption or because the SVM may be used in an online learning setting. In this paper an approach for incremental learning with Support Vector Machines is presented, that improves existing approaches. Empirical evidence is given to prove that this approach can effectively deal with changes in the target concept that are results of the incremental learning setting.

Keywords. Support Vector Machines, Incremental Learning

1 Introduction

The ability to incrementally learn from batches of data is an important feature that makes a learning algorithm more applicable to real-world problems. Incremental learning may be used to keep memory and time consumption of the learning algorithm at a manageable level or because one needs to make predictions at a time when the whole data is not yet available (online setting).

The most important question of incremental learning is, whether the target concept may change between the learning steps or is assumed to be constant. The first case is called concept drift, the second case is true incremental learning. The practical difference between both kinds of learning is, that in the concept drift setting, old examples may be misleading, as they are examples of an old target concept, that may be quite different from the concept one is trying to learn. In the case of true incremental learning, all examples contain the same information about the target concept (more specific: one cannot judge the information of an example from its age). As a consequence, one can judge the performance of an incremental learning algorithm simply by comparing its results to the results of the learning algorithm trained an all data simultaneously as the gold standard. This paper will deal with incremental learning.

[*] A short version of this paper has appeared in the Proceedings of the ICDM 2001, San Jose, CA, USA

Support Vector Machines (SVMs) have been successfully used for learning with large and high dimensional data sets. This is because of the fact that the generalization property of an SVM does not depend on all the training data but only a subset thereof, the so-called Support Vectors [Vapnik, 1998]. Unfortunately, the training of Support Vector Machines itself can very time consuming, especially when dealing with noisy data.

As the number of Support Vectors typically is very small compared to the number of training examples, SVMs promise to be an effective tool for incremental learning by compressing the data of the previous batches to their Support Vectors. This approach to incremental learning with Support Vector Machines has been investigated in [Syed et al., 1999b] where it has been shown that incrementally trained Support Vector Machines compare very well to their non-incrementally trained equivalent.

The problem of drifting concepts in incremental Support Vector Machine learning has been addressed in [Syed et al., 1999a] where it has been experimentally validated that SVMs handle drifting concepts well with respect to the criteria of stability of the result during the learning steps, improvement of the prediction accuracy during the progress of the training and recoverability from errors resulting of the drifting concepts. Another approach to the handling of drifting concepts has been pursued in [Klinkenberg and Joachims, 2000], where a performance estimator [Joachims, 2000] has been used to detect whether a drift in the underlying concept did occur, at which point the old data was being discarded and training took place only on the new data.

This paper deals with the setting of incremental learning, i.e. the data is presented to the algorithm in several batches, such that the algorithm should produce a preliminary result after each training step and is not allowed to directly use all the data in the last training step, to keep down time and space consumption. Nevertheless, we will keep an eye on the idea of concept drift, because in real-life problems there may occur a change in the target concept between different batches of data, which is simply an artifact of the way the data is presented to the learning algorithm.

When, for example, the learning examples are being generated by a controlled experiment, the experimenter may find it favorable to follow a test plan which changes the process parameters successively. If these examples are used for incremental learning, earlier batches contain examples from very different regions of the parameter space than later batches. One may also observe changes in the training data which have no correspondence to controllable parameters of the experiment, e.g. if in applications in engineering the quality of a machine deteriorates over the course of its life-cycle or if in medical applications the algorithm learns on the data of one patient at a time. In all these examples, one is interested in a result which is optimal with respect to all data (all parameters / points in time / patients), i.e. although there are observable and significant changes in the data from batch to batch, no concept drift occurs.

Of course, all these problems could be effectively solved by randomly sampling the data, but in practice a main reason to use incremental learning in the first place is, that one does not have control over the way the examples are generated or cannot wait until all examples are ready. The question in this paper will be, how much the learning algorithm will be influenced by this effect, which we will call pseudo-concept drift.

Section 2 will present a brief introduction to Support Vector Machines and the their important properties for this paper. Section 3 will discuss the drawbacks of the existing approach to incremental SVM learning and develop an improved incremental learning algorithm. In section 4 the new algorithm will be experimentally evaluated. A summary and outlook in section 5 will conclude the discussion.

2 Support Vector Machines

Support Vector Machines (SVMs) are based on the work of Vladimir Vapnik in statistical learning theory [Vapnik, 1998]. Statistical learning theory deals with the question, how a function f from a class of functions $(f_\alpha)_{\alpha\in\Lambda}$ can be found, that minimizes the expected risk

$$R[f] = \int\int L(y, f(x))dP(y|x)dP(x) \tag{1}$$

with respect to a loss function L, when the distributions of the examples $P(x)$ and their classifications $P(y|x)$ are unknown and have to be estimated from finitely many examples $(x_i, y_i)_{i\in I}$.

The SVM algorithm solves this problem by minimizing the regularized risk $R_{\text{reg}}[f]$, which is the weighted sum of the empirical risk $R_{\text{emp}}[f]$ with respect to the data $(x_i, y_i)_{i=1\ldots n}$ and a complexity term $||w||^2$

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda||w||^2.$$

In their basic formulation, SVMs find a linear decision function $y = f(x) = \text{sign}(w \cdot x + b)$ that both minimizes the prediction error on the training set and promises the best generalization performance. Given the examples $(x_1, y_1), \ldots, (x_n, y_n)$ this is done by solving the following optimization problem:

$$\begin{aligned} \Psi(w, \xi, \xi^*) &= \frac{1}{2}(w^T w) + C\sum_{i=0}^{n} \xi_i \\ &\to \min \end{aligned} \tag{2}$$

subject to

$$y_i(w^T x_i + b) \leq 1 - \xi_i, i = 1, \ldots, n \tag{3}$$
$$\xi_i \geq 0, i = 1, \ldots, n \tag{4}$$

The hyperplane vector w has a representation in terms of the training examples $(x_i, y_i)_{i\in I}$ and their lagrangian multipliers $(\alpha_i)_{i\in I}$, that are calculated during the optimization process:

$$w = \sum_{i\in I} \alpha_i y_i x_i.$$

Examples with non-zero lagrangian multiplier are called Support Vectors. A key property of Support Vector Machines is, that training a SVM on the Support Vectors alone gives the same result as training on the complete example set.

The optimal constant C for the learning problem at hand is usually determined by some model selection technique, e.g. cross-validation.

In equation (2), the empirical risk of the SVM solution is measured with respect to a linear loss function. It is important to notice that the theory of the Support Vector algorithm is not restricted to the case of linear loss functions, but can be extended to broader classes of loss functions (e.g. see [Smola et al., 1998]).

3 Incremental Learning with Support Vector Machines

In principle, all working set methods to train SVMs, especially shrinking, (see [Osuna et al., 1997], [Joachims, 1999]) can be considered as incremental learning algorithms, as only a small part of the examples is actually used for optimization in each step. But this approaches are not useful to true incremental learning, as none of the examples are discarded during training and therefore will have to be reconsidered in each working set selection step. Therefore, no improvement in terms of space and time consumption can be expected here.

3.1 Existing Approaches

The approach to incremental learning with SVMs presented in [Syed et al., 1999b] is this: For each new batch of data a Support Vector Machine is trained on the new data and the Support Vectors from the previous learning step. In the following this approach will be called the SV-incremental algorithm.

The reasoning behind this is as follows: The resulting decision function of an SVM depends only on its Support Vectors, i.e. training an SVM on the Support Vectors alone results in the same decision function as training on the whole data set. Because of this, one can expect to get an incremental result that is equal to the non-incremental result, if the last training set contains all examples that are Support Vectors in the non-incremental case.

But how to decide whether an examples will end up as an Support Vector in the non-incremental case without actually training on all data? If a batch of data is a good sample, i.e. if the statistical properties of that batch and the whole data set do not differ very much, one can expect the resulting decision function to be roughly similar to the final decision function. Therefore, a Support Vector in the final set is likely to be a Support Vector in previous iterations too.

The problem with this approach is the assumption that the batch of data will be an appropriate sample of the data. While this may be likely if the data is collected beforehand and presented to the SVM in randomly drawn batches, there is no way to tell if this is the case if the examples are generated online during the training.

The theoretical problem behind this is, that the Support Vectors are a sufficient description of the decision boundary between the examples, but not of the examples themselves. As there are typically only very few Support Vectors, their influence on the decision function in the next incremental learning step may be very small if the new data is distributed differently.

See for example the 200 examples in figure 1 and imagine the data is presented to the SVM in two batches, such that the first batch contains all the examples with $x < 0$. Training an SVM on the first batch of data will lead to relatively few Support Vectors (18 for $C = 1000$). Now train an SVM with the Support Vectors of the first batch and the rest of the examples. Figure 2 shows the resulting decision function, together with the decision function learned on all data.

It is obvious that SVM results largely ignores the old Support Vectors and almost corresponds to the decision function that would have been learned on the second batch of examples alone. Usually this is a desired property of the SVM algorithm, because it means that the SVM is robust against outliers in the data - only in this case the outliers are the old Support Vectors one would wish to take into account for the new decision function.

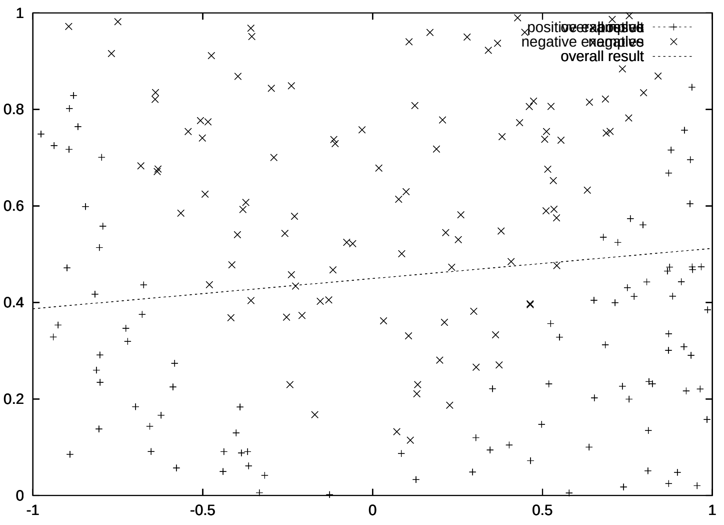

Figure 1 Example data set (Plusses = positive examples, crosses = negative examples) with SVM decision function (C=1000).

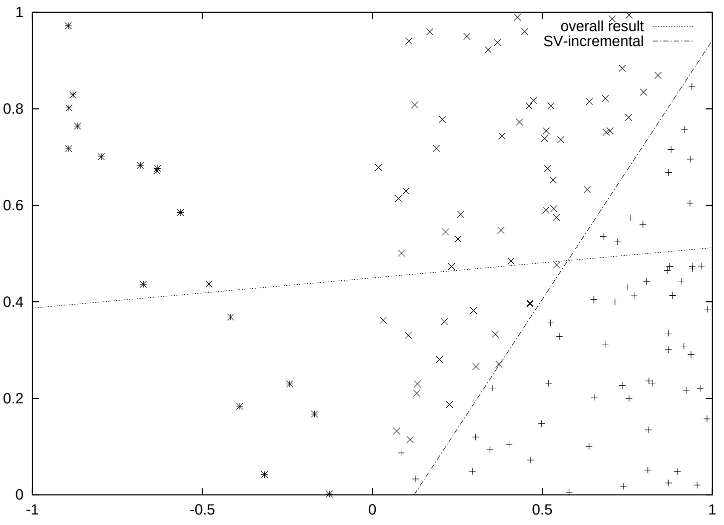

Figure 2 Second batch of data including old SVs (= stars), resulting decision function and desired overall decision function.

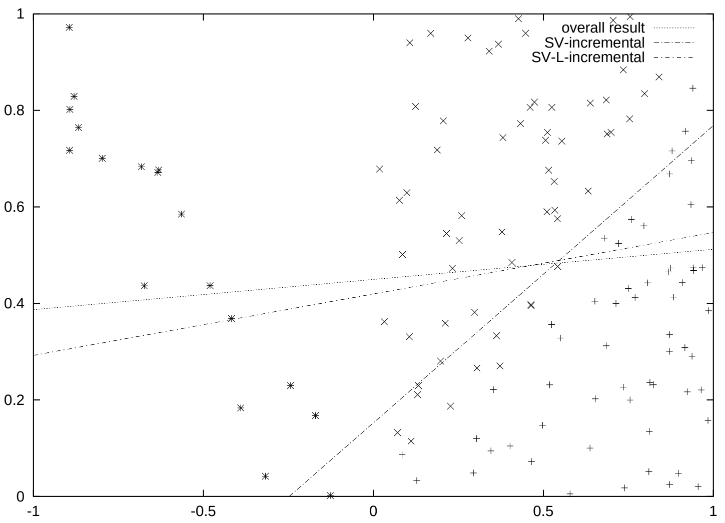

Figure 3 Comparison of the results of the SV-incremental algorithm and the SV-L-incremental algorithm on the example problem.

3.2 A New Incremental Learning Algorithm

As stated above, the SV-incremental algorithm suffers from the problem, that the Support Vectors do not describe the whole set of data but only the decision function induced by it. To make up for this problem in the incremental learning algorithm, one needs to make an error on the old Support Vectors (which represent the old learning set) more costly than an error on a new example.

Fortunately, this can easily be accomplished in the Support Vector algorithm. Let $(x_i, y_i)_{i \in S}$ be the old Support Vectors and $(x_i, y_i)_{i \in I}$ be the new examples. Then replace definition (2) by

$$\Psi(w, \xi, \xi^*) = \frac{1}{2}(w^T w) + C \left(\sum_{i \in I} \xi_i + L \sum_{i \in S} \xi_i \right).$$

This modification of the SVM problem can be viewed as training the SVM with respect to a new loss function.

A natural choice for L is to let L be the number of examples in the previous batch divided by the number of Support Vectors.

$$L = \frac{\#\text{examples}}{\#\text{SVs}}$$

This comes from the idea to approximate the average error of an arbitrary decision function over all examples by the average error over just the Support Vectors. In other words: Every Support Vector stands for a constant fraction of all examples. This algorithm will be called the SV-L-incremental algorithm.

Figure 3 shows the result of the SV-L-incremental algorithm on the example problem of the last section. It is obvious that its result lies much closer to the overall result than the SV-incremental algorithm.

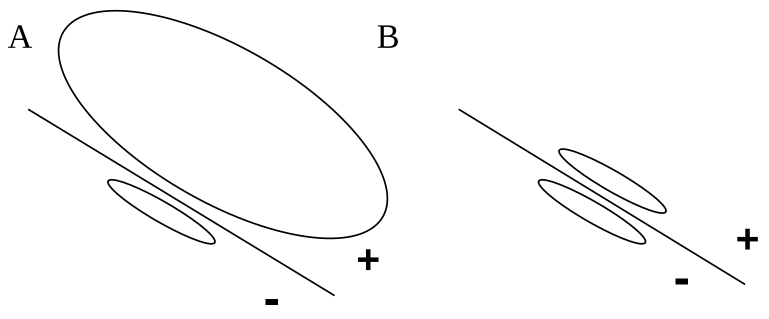

Figure 4 Two example set A and B with equal $P(y|x)$ but different $P(x)$.

3.3 Interpretability of Support Vectors

The results so far give interesting insight into the question, how Support Vectors can be interpreted. It is often argued, that the Support Vectors provide a sufficient representation of the examples for the given classification task, because training an SVM on the Support Vectors gives the same decision function as training on the whole data set. But of course, this only means that the Support Vectors are a sufficient representation of the *decision function* on the examples, not the *examples* themselves.

The important difference is, that in terms of the Statistical Learning Principle as formalized in equation 1, the Support Vectors provide an estimate of $P(y|x)$, but not of $P(x)$. In figure 4, both set A and B have the same $P(y|x)$, but different $P(x)$. If the Support Vectors are used to represent the entire data set, like in the incremental learning algorithm, this has to be taken into account.

4 Experiments

To compare both algorithms, some experiments, both on artificial data and on real-world data sets were made. A version of mySVM [Rüping, 2000], that was modified to handle loss functions defined per example, was used [1].

4.1 Artificial Data

To compare the performance of the SV-incremental and the SV-L-incremental algorithm, two experiments with artificial data sets were made. The performance of both algorithms were compared to that of non-incremental learning in the settings with and without concept changes. As the principle practicability of incremental learning with SVMs has already be en shown in [Syed et al., 1999b], the experiments here were restricted to the case of incremental learning in two steps.

Each training set consisted of three subsets, each of which had a size of 100 examples (50 positive and 50 negative). The first two subsets were used to incrementally train the SVM and the third subset was used as a test set to determine the accuracy of the resulting classifiers. All

[1]The modified version of mySVM is available from the author on request

the results printed are the averages of 10-fold cross-validation.

In the case without concept changes, all the positive examples were drawn randomly from a two-dimensional normal distribution with center $(1, 1)$ and variance 1. The negative examples were drawn from a similar normal distribution with center $(-1, -1)$. The following table shows the accuracy of the training with different values of C.

C	non-inc.	SV-inc.	SV-L-inc.
1e-06	0.893	0.896	0.895
1e-05	0.916	0.787	0.796
1e-04	0.919	0.799	0.839
0.001	0.926	0.928	0.878
0.01	0.919	0.799	0.795
0.1	0.926	0.927	0.805
1	0.916	0.882	0.916
10	0.925	0.921	0.923
100	0.938	0.932	0.937
1000	0.921	0.896	0.923
1e04	0.920	0.881	0.904
1e05	0.922	0.878	0.874
1e06	0.909	0.892	0.783

To simulate a concept change in the data, another set of data was created. This time the first training set consisted of positive examples drawn randomly from a two-dimensional normal distribution with center $(1, 1)$ and negative examples normally distributed with center $(-1, -1)$. The second training set had positive examples centered at $(1, -1)$ and negative examples centered at $(-1, 1)$. The variance was 1 in all cases. The test set consisted of 50 variables from the distribution of the first training set and another 50 variables from the distribution of the second training set, such that the distribution of the variables in the test set and the distribution in the union of the first and second training set were identical. The next table shows the accuracy of the resulting classifiers with different values of C.

C	non-inc.	SV-inc.	SV-L-inc.
1e-06	0.835	0.555	0.812
1e-05	0.823	0.636	0.749
1e-04	0.852	0.599	0.812
0.001	0.835	0.553	0.784
0.01	0.854	0.562	0.844
0.1	0.835	0.684	0.814
1	0.836	0.758	0.837
10	0.834	0.755	0.819
100	0.845	0.743	0.759
1000	0.844	0.689	0.734
1e04	0.834	0.689	0.788
1e05	0.834	0.678	0.746
1e06	0.847	0.620	0.684

As can be seen, the SV-incremental and the SV-L-incremental algorithm perform comparable in the absence of a concept change, whereas in the setting with concept changes the SV-L-incremental algorithm performs better than the original SV-incremental algorithm.

4.2 Real-World Data

4.2.1 Data Set To check the performance on real-world data sets, several data sets from the UCI Repository of Machine learning Databases [Murphy and Aha, 1994] were used. Before the training, nominal variables have been binarized and the attributes have been scaled to expectancy 0 and variance 1. All results have been obtained by 10-fold cross-validation, i.e. or each test set, a classifier has been trained incrementally on the 9 other batches of data and the performance of the final classifier on the test set has been recorded.

For each data set, the optimal kernel function has been determined for the non-incremental case by 10-fold cross-validation on the complete data set. In all experiments, the value $C = 1$ has been used. The following table shows these data sets and the kernel function that were used in the experiments.

Name	Dim.	Examples	Kernel
australian	38	690	RBF, $\gamma = 0.0005$
diabetes	8	768	RBF, $\gamma = 0.01$
german	24	1000	RBF, $\gamma = 0.0005$
heart	20	270	RBF, $\gamma = 0.0005$
ionosphere	34	351	RBF, $\gamma = 0.1$
liver-disorders	6	345	RBF, $\gamma = 0.1$
monks-1	15	556	RBF, $\gamma = 0.1$
monks-2	15	601	RBF, $\gamma = 1$
monks-3	15	554	RBF, $\gamma = 0.001$
mushrooms	112	8124	linear
promotor-gene	228	106	linear
sonar	60	208	RBF, $\gamma = 0.01$

4.2.2 Comparison of the SV-incremental and SV-L-incremental Algorithm The following table compares the Accuracy of the classifier trained on all available data (non-inc.), the SV-incremental algorithm and the SV-L-incremental algorithm.

Dataset	non-inc.	SV-inc.	SV-L-inc.
australian	85.21	85.50	85.50
diabetes	70.94	70.80	70.42
german	74.90	75.80	76.70
heart	77.03	75.92	79.62
ionosphere	94.02	94.02	94.88
liver-disorders	69.61	69.92	71.05
monks1	100.00	100.00	100.00
monks2	85.03	85.03	85.03
monks3	96.38	96.38	96.38
mushroom	100.00	100.00	100.00
promoter-genes	93.63	93.63	93.63
sonar	85.59	85.59	86.07

4.2.3 Choice of L In section 3.2, the constant L in the SV-L-incremental algorithm has been heuristically determined as

$$L = \frac{\text{\#examples}}{\text{\#SVs}}.$$

To check the influence that this value has on the learning results, some other tests were made where half, double and five times this value was used. The next table shows, that in most cases the value

$$L = 2\frac{\text{\#examples}}{\text{\#SVs}}$$

shows even better results.

Dataset	$L * 0.5$	L	$L * 2$	$L * 5$
australian	74.63	85.50	86.66	85.36
diabetes	66.64	70.42	70.55	69.64
german	73.70	76.70	77.50	76.40
heart	55.55	79.62	83.33	84.07
ionosphere	94.59	94.88	95.45	94.87
liver-disorders	68.74	71.05	70.78	69.93
monks1	100.00	100.00	100.00	100.00
monks2	65.72	85.03	85.03	85.03
monks3	62.46	96.38	96.38	96.38
mushroom	100.00	100.00	100.00	100.00
promoter-genes	93.63	93.63	93.63	93.63
sonar	83.16	86.07	85.07	88.88

4.2.4 Experiments with Pseudo-Concept Drift So far, in all experiments in this section, the examples were distributed to the batches randomly. This is quite fortunate for an incremental algorithm, as the chance of a pseudo-concept drift occurring in the data is small. Now we want to see what happens, if the examples are distributed into the batches in an ordered way.

To do that, the examples have been sorted with respect to their first attribute value and were presented to the incremental algorithm in that order. As a result of the last experiment, the value of L has been doubled.

Dataset	non-inc.	SV-inc.	SV-L-inc.
australian	85.21	85.65	86.37
diabetes	70.94	71.09	70.69
german	74.90	75.89	77.89
heart	77.03	69.25	83.33
ionosphere	94.02	94.58	95.15
liver-disorders	69.61	70.48	70.75
monks1	100.00	100.00	100.00
monks2	85.03	84.02	84.02
monks3	96.38	96.38	96.38
mushroom	100.00	100.00	100.00
promoter-genes	93.63	93.45	90.72
sonar	85.59	80.85	84.21

5 Summary and Future Work

In this paper, a new algorithm for incremental learning with Support Vector Machines was presented. The method was especially designed to deal with problems that arise when the training data contains changing concepts, either real ones or ones that are artifacts arising from the specific order of the examples. It was experimentally shown that the performance of the new algorithm is comparable to an existing approach in the case of learning without concept changes and performs significantly better in the case of changing concepts in the training data.

An interesting question for future work is to find out, if the trick of individually assigning a weight to the examples can also be applied to the problem of learning in situations with real concept drift. Instead of increasing the weights for old Support Vectors, one could also decrease the weight of the old examples to make the SVM more sensitive to changes in newer examples.

It also remains to be investigated, whether this algorithm also works for regression SVMs.

Acknowledgments

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, “Reduction of Complexity for Multivariate Data Structures”) is gratefully acknowledged.

References

[Joachims, 1999] Joachims, T. (1999). Making Large-Scale SVM Learning Practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169 – 184. MIT Press, Cambridge.

[Joachims, 2000] Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference on Machine Learning*, San Francisco. Morgan Kaufman.

[Klinkenberg and Joachims, 2000] Klinkenberg, R. and Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco. Morgan Kaufmann.

[Murphy and Aha, 1994] Murphy, P. M. and Aha, D. W. (1994). UCI repository of machine learning databases.

[Osuna et al., 1997] Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In Principe, J., Giles, L., Morgan, N., and Wilson, E., editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York. IEEE.

[Rüping, 2000] Rüping, S. (2000). *mySVM-Manual*. Universität Dortmund, Lehrstuhl Informatik VIII. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

[Smola et al., 1998] Smola, A., Schölkopf, B., and Müller, K.-R. (1998). General cost functions for support vector regression. In Niklasson, L., Boden, M., and Ziemke, T., editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*.

[Syed et al., 1999a] Syed, N., Liu, H., and Sung, K. (1999a). Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, San Diego, CA, USA.

[Syed et al., 1999b] Syed, N., Liu, H., and Sung, K. (1999b). Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99)*, Stockholm, Sweden.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, Chichester, GB.