Detecting Concept Drift with Support Vector Machines

Ralf Klinkenberg
klinkenberg@ls8.cs.uni-dortmund.de
Department of Computer Science,
University of Dortmund

Thorsten Joachims
joachims@ls8.cs.uni-dortmund.de
Department of Computer Science,
University of Dortmund

For tasks in which data is collected over a large period of time its underlying distribution is likely to change, this change in data is known as concept drift in the field of machine learning.

This paper proposes a new method to recognize and handle concept changes with support vector machines (SVM). The method maintains a window on the training data. The key idea is to automatically adjust the window size so that the estimated generalization error is minimized. The new approach is both theoretically well-founded as well as effective and efficient in practice. Since it does not require complicated parameterization, it is simpler to use and more robust than comparable heuristics. The paper includes an experiment which shows the efficiency of this method on real world concept drift scenario of text data.

For windows of fixed size, the choice of a "good" window size is a compromise between fast adaptivity (small window) and good generalization in phases without concept change (large window). On the other hand, the basic idea of adaptive window management is to adjust the window size to the current extent of concept drift.

The window adjustment approach described in this paper uses support vector machines (Vapnik, 1998) [2] as their core learning algorithm. Support vector machines are based on the structural risk minimization principle (Vapnik, 1998) [2] from statistical learning theory. In their basic form, SVMs learn linear decision rules as:

$$h(\vec{x}) = sign\{\vec{w}\cdot\vec{x}+b\} = +1 \ if \ \vec{w}\cdot\vec{x}+b>0 \ and \ -1 \ if \ \vec{w}\cdot\vec{x}+b\leq 0 \quad (1)$$

described by a weight vector $\overrightarrow{w}$ and a threshold b. The idea of structural risk minimization is to find a hypothesis h for which one can guarantee the lowest probability of error. For SVMs, Vapnik (1998) [2] shows that this goal can be translated into finding the hyperplane with maximum soft-margin.

The key idea is to select the window size so that the estimated generalization error on new examples is minimized. To get an estimate of the generalization error we use a special form of $\xi\alpha$-estimates (Joachims, 2000) [1]. $\xi\alpha$-estimates are a particularly efficient method for estimating the performance of a SVM. $\xi\alpha$-estimators are based on the idea of leave-one-out estimation. While the leave-one-out estimate is usually very accurate, it is very expensive to compute. $\xi\alpha$-estimators overcome this problem using an upper bound on the number of leave-one-out error. It owes its name to the two arguments they are computed from. ξ is the vector of training losses at the solution of the primal SVM training problem. α is the solution of the dual SVM training problem. With n as the total number of training examples, the $\xi\alpha$-estimators of the error rate is

$$Error = (|\{i: \ (\alpha_i+\xi_i)>1\}|)/n \quad (2)$$

The theoretical properties of this $\xi\alpha$-estimator are discussed in Joachims (2000) [1]. It can be shown that the estimator is pessimistically biased, overestimating the true error rate on average. Experiments show that the bias is acceptably small for text classification problem.

Now coming to the algorithm, the algorithm can be summarized as follows:

- Input: S training sample consisting of t batches containing m example
- For h $\in \{0,...,t-1\}$
  - train SVM on examples $\vec{z}_{(t-h,1)},...,\vec{z}_{(t,m)}$
  - compute $\xi\alpha$-estimator on examples $\vec{z}_{(t,1)},...,\vec{z}_{(t,m)}$
- Output: window size which minimizes $\xi\alpha$-estimate

The experiments use a subset of 2600 documents of the data set of TREC consisting of English text. The texts are randomly split into 20 batches of equal size containing 130 documents each. Each text is assigned some categories from 1 to 6. To perform the actual experiment, we will take four data management approaches:

	Full Memory	No Memory	Fixed Size	Adaptive Size
Scenario A:				
Error	20.36% (4.21%)	7.30% (1.97%)	7.96% (2.80%)	5.32% (2.29%)
Recall	51.69% (8.37%)	74.42% (4.61%)	77.64% (6.07%)	85.35% (4.93%)
Precision	64.67% (8.38%)	91.29% (5.10%)	87.73% (5.93%)	91.61% (5.11%)
Scenario B:				
Error	20.25% (3.56%)	9.08% (1.57%)	8.44% (2.00%)	7.56% (1.89%)
Recall	49.35% (7.01%)	67.22% (5.04%)	73.85% (5.51%)	76.70% (5.42%)
Precision	65.09% (6.80%)	88.86% (3.67%)	87.19% (4.18%)	88.48% (3.89%)
Scenario C:				
Error	7.74% (3.05%)	8.97% (2.84%)	10.17% (3.30%)	7.07% (3.16%)
Recall	76.54% (6.26%)	63.68% (5.27%)	68.18% (7.05%)	78.17% (6.34%)
Precision	83.15% (6.69%)	87.67% (7.06%)	79.00% (8.09%)	87.38% (6.99%)

Figure 1: Error, accuracy, recall, and precision of all window management approaches for all scenarios averaged over 10 trials with 20 batches each (standard sample error in parentheses).

- Full memory
- No memory
- Window of fixed size
- Window of adaptive size

And three concept drift scenarios:

- In scenario A, first documents of category 1 are considered relevant. This changes abruptly in batch 10, where documents of category 3 becomes relevant.
- In scenario B, first documents of category 1 are considered relevant. This changes slowly (concept drift) from batch 8 to batch 12, where documents of category 3 are relevant.
- In the scenario C, there is an abrupt concept shift from category 1 to category 3 in batch 9 and back to category 1 in batch 11.

In figure 1, we can see that the adaptive window size algorithm achieves a low average error rate on all three scenarios. Similarly, precision and recall are consistently high. The method directly implements the goal of discarding irrelevant data with the aim of minimizing generalization error. Unlike for the conventional heuristic approaches, this gives the new method a clear and simple theoretical motivation.

[1] T. Joachims. Estimating the generalization performance of a svm efficiently. *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[2] V. Vapnik. The support vector method of function estimation. *Statistical learning theory*, 1998.