# A Simple Method For Estimating Conditional Probabilities For SVMs

**Stefan Rüping**
CS Department, AI Unit
Dortmund University
44221 Dortmund, Germany
rueping@ls8.cs.uni-dortmund.de

## Abstract

Support Vector Machines (SVMs) have become a popular learning algorithm, in particular for large, high-dimensional classification problems. SVMs have been shown to give most accurate classification results in a variety of applications. Several methods have been proposed to obtain not only a classification, but also an estimate of the SVMs confidence in the correctness of the predicted label. In this paper, several algorithms are compared which scale the SVM decision function to obtain an estimate of the conditional class probability. A new simple and fast method is derived from theoretical arguments and empirically compared to the existing approaches.

## 1 Introduction

Support Vector Machines (SVMs) have become a popular learning algorithm, in particular for large, high-dimensional classification problems. SVMs have been shown to give most accurate classification results in a variety of applications. Several methods have been proposed to obtain not only a classification, but also an estimate of the SVMs confidence in the correctness of the predicted label.

Usually, the performance of a classifier is measured in terms of accuracy or some other performance measure based on the comparison of the classifiers prediction $\hat{y}$ of the true class $y$. But in some cases, this does not give sufficient information. For example in credit card fraud detection, one has usually much more negative than positive examples, such that the optimal classifier may be to the default negative classifier. But then, still one would like to find out which transactions are most probably fraudulent, even if this probability is small. In other situations e. g. information retrieval, one could be more interested in a ranking of the examples with respect to their interestingness instead of a simple yes/no-decision. Third, one may be interested to integrate a classifier into a bigger system, for example a multi-classifier learner. To combine and compare the SVM prognosis with that of other learners, one would like a comparable, well-defined confidence estimate. The best method to achieve a confidence estimate that allows to rank the examples and gives well-defined, interpretable values, is to estimate the conditional class probability $P(y|x)$. Obviously, this is a more complex problem than finding a classification $cl(x) \in \{-1, 1\}$, as it is possible to get a classification function by comparing $\hat{P}(y|x)$ to the threshold $0.5$, but not vice versa.

For numerical classifiers, i. e. classifiers of the type $cl(x) = sign(f(x))$ with a numerical decision function $f$, one usually tries to estimation the conditional class probability from the decision function $\hat{P}(y|x) = \hat{P}(y|f(x))$. This reduces the probability estimation from a multi-variate to a one-dimensional problem, where one has to find a scaling function $\sigma$ such that $\hat{P}(Y = 1|x) = \sigma(f(x))$. The idea behind this approach is that the classification $cl(x)$ of examples that lie close to the decision boundary $\{x|f(x) = 0\}$ can easily change when the examples are randomly perturbed by a small amount. This is very hard for examples with very high or very low $f(x)$ (this argument requires some sort of continuity or differentiability constraints on the function $f$). Hence, the probability that the classifier is correct should be higher for larger absolute values of $f$. As was noted by Platt [10], this also means there is a strong prior for selecting a monotonic scaling function $\sigma$.

The rest of the paper is organized as follows: In the next section, we will shortly present the Support Vector Machine and Kernel Logistic Regression algorithm, as far as it is necessary for this paper. In Section 3, existing methods for probabilistic scaling of SVM outputs will be discussed and a new, simple scaling method will be presented. The effectiveness of this method will be empirically evaluated in Section 4.

## 2 Algorithms

### 2.1 Support Vector Machines

Support Vector Machines are a classification method based on Statistical Learning Theory [12]. The goal is to find a function $f(x) = w * x + b$ that minimizes the expected Risk

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

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

$$R\text{reg}[f] = \frac{1}{2}||w||^2 + C\sum_i |1 - y_i f(x_i)|_+ \qquad (1)$$

where $|\xi|_+ = \max(\xi, 0)$. This optimization problem can be efficiently solved in its dual formulation

$$-\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j x_i * x_j + \sum_{i=1}^{n} \alpha_i \to \min \qquad (2)$$

$$w.r.t. \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$
$$\forall i : 0 \leq \alpha_i \leq C$$

## 2.2 The Kernel Trick

The inner product $x_i * x_j$ in Equation 2 can be replaced by a kernel function $K(x_i, x_j)$ which corresponds to an inner product in some space, called feature space. That is, there exists a mapping $\Phi : X \to \mathcal{X}$ such that $K(x, x') = \Phi(x) * \Phi(x')$. This allows the construction of non-linear classifiers by an essentially linear algorithm.

The resulting decision function is given by

$$\begin{aligned} f(x) &= w * \Phi(x) + b \\ &= \sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b. \end{aligned}$$

The actual SVM classification is given by $sign(f(x))$. It can be shown that the SVM solution depends only on its support vectors SV = $\{x_i | \alpha_i \neq 0\}$. See [12; 2] for a more detailed introduction on SVMs.

## 2.3 Kernel Logistic Regression

Kernel Logistic Regression [13; 5; 14; 11] is the kernelized version of the well-known logistic regression technique. The optimization problem is similar to the SVM problem in Equation 1 except that an exponential loss function is used instead of the L1 loss:

$$\frac{1}{2}||w||^2 + C \sum_i g(-y_i(w * x_i - b)) \to \min$$

where

$$g(\xi) = log(1 + e^{\xi})$$

As for the SVM, the problem can be solved in its dual formulation [6]:

$$\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) + C \sum_i g(\xi_i) \to \min .$$

In contrast to the SVM, Kernel Logistic Regression directly models the conditional class probability, i. e. $P(Y = 1|x)$ can be estimated via

$$P(y|x) = \frac{1}{1 + e^{-y(w * x - b)}}.$$

The drawback of KLR is that typically all $\alpha_i$ are nonzero, as all examples play a role in estimating the conditional class probability, whereas in the SVM only a small number of support vectors are needed to classify the examples. Hence, KLR is computationally much more expensive than the SVM.

## 3 Probabilistic Scaling of Support Vector Machines

One can easily see that the SVM decision function $f(x) = w * \Phi(x) + b$ gives the feature space distance of the transformed example $\Phi(x)$ to the hyperplane defined by $(w, b)$. Assuming that $P(Y = 1|x)$ is continuous in $x$, it seems reasonable that examples lying closer to the hyperplane have a larger probability of being misclassified than examples lying far away (the closer the example is to the hyperplane, the smaller changes have to be to produce a different classification). Hence, it seem suitable to model the conditional class probability $P(y|x)$ as a function of the value of the SVM decision function, i. e. $\hat{P}(Y = 1|x) = \sigma(f(x))$ with an appropriate scaling function $\sigma$.

There are several ad-hoc scaling functions, e. g. the softmax scaler

$$\sigma_{\text{softmax}}(z) = \frac{1}{1 + e^{-2z}},$$

which monotonously maps the decision functions value $z = f(x)$ to the interval $[0, 1]$. The scaler assumes that for the decision function is of the type $sign(z)$ and hence for $z = 0$ the classifiers class decision is smallest such that $z$ is mapped to the conditional class probability $0.5$. This allows to view $\sigma_{\text{softmax}}(z)$ as a probability. However, this mapping is not very well founded, as the scaled values are not justified from the data.

To justify the interpretation $\hat{P}(Y = 1|x) = \sigma(f(x))$, it is better to use data to calibrate the scaling. One can use a subset of the data which has not been used for training (or use a cross-validation-like approach) and optimize the scaling function $\sigma$ to minimize the error between the predicted class probability $\sigma(f(x))$ and the empirical class probability defined by the class values $y$ in the new data. There are two error measures which are usually used, cross-entropy and mean squared error. Cross-entropy is defined by

$$CRE = \sum_i y_i log(z_i) + (1 - y_i) log(1 - z_i)$$

(where $z_i = \sigma(f(x_i))$), which is the Kullback-Leibler distance between the predicted and the empirical class probability. For comparison of different data sets it is better to divide the cross-entropy by the number of examples and work with the mean cross-entropy mCRE. The mean squared error is defined by

$$MSE = \frac{1}{n} \sum_i (y_i - p_i)^2.$$

It is an appropriate error measure because for a binary random variable $Y \in \{0, 1\}$, the expected value of $(Y - p)^2$ is minimized by $p = P(Y = 1)$. Hence, the task of estimating the conditional class probability becomes a regression task. The open question is, what types of scaling functions should be fitted to the data.

Motivated by an empirical analysis, Platt [10] uses scaling functions of the form

$$\sigma_{a,b}(z) = \frac{1}{1 + e^{-az + b}}$$

with $a \geq 0$ to obtain a monotonically increasing function. The parameters $a$ and $b$ are found by minimization of the cross-entropy error over a test set $(x_i, y_i)$ with $z_i = f(x_i)$. For an efficient implementation, see [8].

Garczarek [4] proposes a method which scales classification values by

$$\sigma(z) = \mathcal{B}^{-1}_{\alpha 1, \beta 1} \mathcal{B}_{\alpha 1, \beta 1}(z)$$

where $\mathcal{B}_{\alpha,\beta}$ is the Beta distribution function with parameters $\alpha$ and $\beta$. The parameters $\alpha 1, \beta 1, \alpha 2$ and $\beta 2$ are selected such that over a test set $(x_i, y_i)$

1. the average value of $\sigma(f(x))$ for each class is identical to the classification performance of the classifier $f$ in this class and
2. the mean square error $(y - \sigma(f(x)))^2$ is minimized.

Originally, the algorithm is designed for multiclass problems and computes an individual scaler for each predicted class. For binary problems, it is better to modify this approach such that only one scaler is generated. This avoids

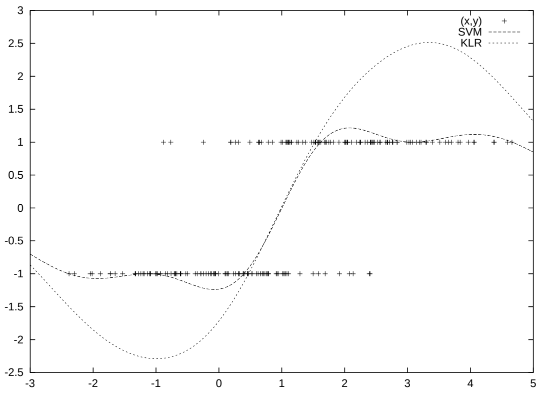

Figure 1: One-dimensional comparison of SVM and KLR predictions. Negatives examples are drawn from N(0,1) (dots at y=-1), positive examples from N(2,1) (dots at y=1). Both methods find the class border at x=1, but the SVM prediction is essentially constant for y outside [-1,1]. KLR correctly estimates higher confidences for points nearer to class centers.

discontinuities in $\hat{P}(Y = 1|x)$ when the prediction changes from one class to the other.

Binning has also been applied to this problem [3]. The decision values are discretized into several bins and one can estimate the the conditional class probability by counting the class distribution in the single bins. Other, more complicated approaches also exists, see e. g. [7] or [12], Ch. 11.11.

### 3.1 Theoretical Limitations

Bartlett and Tewari [1] show that there is a tradeoff between sparseness of a classifier and the ability to estimate conditional probabilities. Their result says, in short, that if one is able to estimate $P(Y = 1|x)$ on some interval, sparseness is lost in that region. Hence, the question arises in how far the decision function of the SVM, which generally produces sparse classifiers, can approximate the true conditional density or the estimate of the non-sparse KLR, respectively.

The problem can be seen in Equation 1. To obtain a maximally accurate classifier, the SVM contains $|1-y_i f(x_i)|_+$ in its objective function, i. e. the classifier is punished if $y_i f(x_i) < 1$ (it becomes a support vector). In this case, this forces an ordering on the values $y_i f(x_i)$ where the value is the higher, the more similar the example is to the rest of the examples in its class in feature space. Consequently, an estimation of $P(y_i|x_i)$ can be constructed from $y_i f(x_i)$. When the example is classified correctly with sufficient margin, i. e. $y_i f(x_i) > 1$, this example generates no loss and hence no specific order is enforced on these examples. For the SVM, all the examples on the right side of the margin have the same probability $P(y|x)$. This behavior can be seen in Figure 1.

What can be said about the support vectors? In the previous section we already saw that minimizing the mean squared error between the estimation function $\sigma(f(x))$ and $y$ gives a proper estimate of $P(y|x)$, as for a fixed $x$ the MSE is minimized for $\sigma(f(x)) = P(Y = 1|x)$. However, the error criterion in the SVM is the absolute error, not the squared error, and one can show that for a fixed $x$ the absolute error is minimized at $\sigma(f(x)) = 1$ iff $P(Y = 1|x) > 0.5$ and $\sigma(f(x)) = 0$ otherwise. What comes to the rescue is that $f(x)$ is not determined for each $x$ independently, but for all $x$ together. Hence, if not overfitting occurs, at least a value of $f(x) = 0$ is an indicator of $P(Y = 1|x) = 0.5$ and it seems plausible that $f(x)$ contains some useful information about $P(y|x)$.

### 3.2 A Simple Estimation Method

From the previous discussion we know that decision function value with $|f(x)| > 1$ are unreliable for estimating the conditional class probability. Values with $|f(x)| \leq 1$ directly optimize the order of the examples with respect to $P(y|x)$. Hence, the question arises if it is possible to estimate $P(Y = 1|x)$ by the following trivial procedure

$$\sigma_{01}(f(x)) = \begin{cases} p_+ & \text{iff} f(x) > 1 \\ \frac{1}{2}(1 + f(x)) & \text{iff} f(x) \in [-1, 1] \\ p_- & \text{iff} f(x) < -1 \end{cases}$$

where $p_+$ is the fraction of positive examples with $f(x) > 1$ and $p_-$ is the fraction of positive examples with $f(x) < -1$. For $f(x) \in [-1, 1]$, the SVM function is simply linearly scaled to $[0, 1]$. Similarly, one can define $\sigma_{PP}$ by clipping $f(x)$ at $p_-$ and $p_+$ instead of 0 and 1.

The advantage of this method compared to the existing approaches is that it requires almost work when training or applying the classifier, except counting the probabilities $p_+$ and $p_-$ and still gives reasonable, empirically founded probability estimates.

## 4 Experiments

The experiments were conducted on 11 data sets, including 7 data sets from the UCI Repository [9] (covtype, diabetes, digits, digits, ionosphere, liver, mushroom, promoters) and 4 other real-world data sets: a business cycle analysis problem (business), an analysis of a direct mailing application (directmailing), a data set from a life insurance company (insurance) and intensive care patient monitoring data (medicine). Prior to learning, nominal attributes were binarised and the attributes were scaled to expectancy 0 and variance 1. Multi-class-problems were converted to two-class problems by arbitrarily selecting two of the classes (covtype and digits) or combining smaller classes into a single class (business, medicine). For the covtype data set, a 1% sample was drawn. The following table sums up the description of the data sets:

| Name | Size | Dimension |
|---|---:|---:|
| covtype | 4951 | 48 |
| diabetes | 768 | 8 |
| digits | 776 | 64 |
| ionosphere | 351 | 34 |
| liver | 345 | 6 |
| mushroom | 8124 | 126 |
| promoters | 106 | 228 |
| business | 157 | 13 |
| directmailing | 5626 | 81 |
| insurance | 10000 | 135 |
| medicine | 6610 | 18 |

Experiments were made with Support Vector Machines and Kernel Logistic Regression with both linear and radial basis kernel. The parameters of the algorithms were selected in a prior step to optimize accuracy. The following algorithms were compared in the experiments:

**KLR:** Kernel Logistic Regression, used as the baseline.

**SVM-Platt:** SVM using Platt's scaling.

**SVM-Beta:** SVM using Garczarek's beta scaling.

**SVM-Beta-2:** SVM using binary beta scaling.

**SVM-Bin:** SVM and binning.

**SVM-Softmax:** SVM and softmax scaling.

**SVM-01:** SVM and output $f(x)$ clipped between 0 and 1.

**SVM-PP:** SVM and output $f(x)$ clipped between $P(Y = 1|f(x) < -1)$ and $P(Y = 1|f(x) > 1)$.

All reported results were 10-fold cross-validated. For the linear SVM and KLR, the following results were obtained:

| Method | MSE | mCRE |
|---|---|---|
| KLR | 0.1000 | 0.0332 |
| SVM-Platt | 0.0912 | 0.0291 |
| SVM-Beta | 0.5966 | $\infty$ |
| SVM-Beta-2 | 0.0915 | 0.0301 |
| SVM-Bin (10 bins) | 0.1201 | 0.0384 |
| SVM-Bin (50 bins) | 0.1301 | 0.0415 |
| SVM-Softmax | 0.0975 | 0.0343 |
| SVM-01 | 0.0970 | 0.0317 |
| SVM-PP | 0.0933 | 0.0296 |

With respect to the mean squared error, we get the following ranking: SVM-Platt < SVM-Beta-2 < SVM-PP < SVM-01 < SVM-Softmax < KLR < SVM-Bin-10 < SVM-Bin-50 << SVM-Beta. Sorting by mean cross-entropy, SVM-Beta-2 and SVM-PP change places, as well as SVM-Softmax and Bin-10.

The RBF kernel gave the following results:

| Method | MSE | mCRE |
|---|---|---|
| KLR | 0.0748 | 0.0242 |
| SVM-Platt | 0.0770 | 0.0250 |
| SVM-Beta | 0.6009 | $\infty$ |
| SVM-Beta-2 | 0.0819 | 0.0278 |
| SVM-Bin (10 bins) | 0.0939 | 0.0305 |
| SVM-Bin (50 bins) | 0.1106 | 0.0356 |
| SVM-Softmax | 0.0946 | 0.0327 |
| SVM-01 | 0.0916 | 0.0307 |
| SVM-PP | 0.0904 | 0.0289 |

This gives the following ranking for MSE: KLR < SVM-Platt < SVM-Beta-2 < SVM-PP < SVM-01 < SVM-Bin-10 < SVM-Softmax < SVM-Bin-50 << SVM-Beta.

A close inspection reveals that these results do not give the full picture, as the error measures reach very different values for the individual data sets. E. g. , the MSE for Kernel Logistic Regression with radial basis kernel runs from $10^{-7}$ (mushroom) to $0.191$ (liver). To allow for a better comparison, the methods were ranked according to their performance for each data set. The following table gives the average rank of each of the methods for the linear kernel:

| | avg. rank from | |
|---|---|---|
| Method | MSE | mCRE |
| KLR | 3.18 | 3.09 |
| SVM-Platt | 3.18 | 3.45 |
| SVM-Beta | 9.00 | 9.00 |
| SVM-Beta-2 | 3.27 | 3.45 |
| SVM-Bin (10 bins) | 5.18 | 5.55 |
| SVM-Bin (50 bins) | 6.55 | 6.45 |
| SVM-Softmax | 5.18 | 5.36 |
| SVM-01 | 4.91 | 5.09 |
| SVM-PP | 3.45 | 3.55 |

The corresponding table for the radial basis kernel:

| | avg. rank from | |
|---|---|---|
| Method | MSE | mCRE |
| KLR | 1.82 | 1.55 |
| SVM-Platt | 2.82 | 2.64 |
| SVM-Beta | 9.00 | 9.00 |
| SVM-Beta-2 | 4.27 | 4.27 |
| SVM-Bin (10 bins) | 4.82 | 4.36 |
| SVM-Bin (50 bins) | 6.73 | 6.73 |
| SVM-Softmax | 5.73 | 5.91 |
| SVM-01 | 5.36 | 5.64 |
| SVM-PP | 3.64 | 4.73 |

To validate the significance of the results, a paired t-test ($\alpha = 0.05$) was run over the cross-validation runs. The following table shows the comparison of the cross-entropy for the linear kernel of the best five of the scaling algorithms. Each row of the table shows how often the hypothesis that the estimation in that row is better than the estimation in the corresponding column was rejected. E. g. , the 6 in the last row and first column shows that the hypothesis that softmax scaling is better than KLR was rejected for 6 of the data sets. The contrary hypothesis was rejected on 2 data sets (first row, last column).

| | KLR | Platt | Beta2 | PP | Bin10 | Soft |
|---|---|---|---|---|---|---|
| KLR | 0 | 2 | 2 | 2 | 2 | 2 |
| Platt | 4 | 0 | 0 | 1 | 1 | 0 |
| Beta2 | 4 | 3 | 0 | 2 | 1 | 0 |
| PP | 6 | 4 | 3 | 0 | 2 | 0 |
| Bin10 | 7 | 6 | 6 | 5 | 0 | 3 |
| Soft | 6 | 8 | 8 | 6 | 4 | 0 |

These are the results for cross-entropy and the radial basis kernel:

| | KLR | Platt | Beta2 | PP | Bin10 | Soft |
|---|---|---|---|---|---|---|
| KLR | 0 | 0 | 0 | 0 | 0 | 0 |
| Platt | 6 | 0 | 0 | 1 | 0 | 0 |
| Beta2 | 7 | 6 | 0 | 4 | 1 | 0 |
| PP | 7 | 5 | 4 | 0 | 2 | 0 |
| Bin10 | 8 | 3 | 3 | 3 | 0 | 2 |
| Soft | 9 | 9 | 7 | 9 | 6 | 0 |

The corresponding tables for MSE show similar results. Summing up, we see that

- Kernel Logistic Regression give the best estimation of the conditional class probability (with some outliers in the linear case).
- The best scaling for the SVM is obtained by Platt's method and binary Beta Scaling.
- The trivial PP-scaling performs comparable to the much more complicated techniques.
- Multiclass Beta scaling gives by far the worst results (which was expected from the non-continuicity of its method of scaling each predicted class on its own).

## 5 Summary

The experiments in this paper showed that a trivial method of estimating the conditional class probability $P(y|x)$ from the output of a SVM classifier performs comparably to much more complicated estimation techniques.

## Acknowledgments

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

## References

[1] Peter L. Bartlett and Ambuj Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. submitted, 2004.

[2] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] Joseph Drish. Obtaining calibrated probability estimates from support vector machines. Technical report, University of California, San Diego, June 2001.

[4] Ursula Garczarek. *Classification Rules in Standardized Partition Spaces*. PhD thesis, Universität Dortmund, 2002.

[5] T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[6] S. S. Keerthi, K. Duan, S. K. Shevade, and A.N. Poo. A fast dual algorithm for kernel logistic regression. Submitted for publication in Machine Learning.

[7] James Tin-Yau Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10(5):1018–1031, September 1999.

[8] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on platt's probabilistic outputs for support vector machines, May 2003.

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

[10] John Platt. *Advances in Large Margin Classifiers*, chapter Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. MIT Press, 1999.

[11] Volker Roth. Probabilistic discriminative kernel classifiers for multi-class problems. In B. Radig and S. Florczyk, editors, *Pattern Recognition–DAGM'01*, number 2191 in LNCS, pages 246–253. Springer, 2001.

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

[13] Grace Wahba. *Advances in Kernel Methods - Support Vector Learning*, chapter Support Vector Machines, Reproducing Kernel Hilbert Spaces and the Randomized GACV, pages 69–88. MIT Press, 1999.

[14] Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Neural Information Processing Systems*, volume 14, 2001.