Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods *

John C. Platt

March 26, 1999

John C. Platt
Microsoft Research
1 Microsoft Way
Redmond, WA 98052
jplatt@microsoft.com
http://research.microsoft.com/~jplatt

Abstract

The output of a classifier should be a calibrated posterior probability to enable post-processing. Standard SVMs do not provide such probabilities. One method to create probabilities is to directly train a kernel classifier with a logit link function and a regularized maximum likelihood score. However, training with a maximum likelihood score will produce non-sparse kernel machines. Instead, we train an SVM, then train the parameters of an additional sigmoid function to map the SVM outputs into probabilities. This chapter compares classification error rate and likelihood scores for an SVM plus sigmoid versus a kernel method trained with a regularized likelihood error function. These methods are tested on three data-mining-style data sets. The SVM+sigmoid yields probabilities of comparable quality to the regularized maximum likelihood kernel method, while still retaining the sparseness of the SVM.

1 Introduction

Constructing a classifier to produce a posterior probability $P(\text{class}|\text{input})$ is very useful in practical recognition situations. For example, a posterior probability allows decisions that can use a utility model [4]. Posterior probabilities are also required when a classifier is making a small part of an overall decision, and the classification outputs must be combined for the overall decision. An example of this combination is using a Viterbi search or HMM to combine recognition results from phoneme recognizers into word recognition [1]. Even in the simple case of a multi-category classifier, choosing the category based on maximal posterior probability over all classes is the Bayes optimal decision for the equal loss case.

However, Support Vector Machines [19] (SVMs) produce an uncalibrated value that is not a probability. Let the unthresholded output of an SVM be

$$f(\mathbf{x}) = h(\mathbf{x}) + b, \tag{1}$$

where

$$h(\mathbf{x}) = \sum_i y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) \tag{2}$$

*In *Advances in Large Margin Classifiers*, Alexander J. Smola, Peter Bartlett, Bernhard Schölkopf, Dale Schuurmans, eds., MIT Press, (1999), to appear.

lies in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{F}$ induced by a kernel k [22]. Training an SVM minimizes an error function that penalizes an approximation to the training misclassification rate plus a term that penalizes the norm of h in the RKHS:

$$C \sum_i (1 - y_i f_i)_+ + \frac{1}{2}||h||_{\mathcal{F}}, \tag{3}$$

where $f_i = f(\mathbf{x}_i)$. Minimizing this error function will also minimize a bound on the test misclassification rate [19], which is also a desirable goal. An additional advantage of this error function is that minimizing it will produce a sparse machine where only a subset of possible kernels are used in the final machine.

One method of producing probabilistic outputs from a kernel machine was proposed by Wahba [20, 22]. Wahba used a logistic link function,

$$P(\text{class}|\text{input}) = P(y = 1|\mathbf{x}) = p(\mathbf{x}) = \frac{1}{1 + \exp(-f(\mathbf{x}))}, \tag{4}$$

where f is defined as above, and then proposed minimizing a negative log multinomial likelihood plus a term that penalizes the norm in an RKHS:

$$-\frac{1}{m} \sum_i \left(\frac{y_i + 1}{2} \log(p_i) + \frac{1 - y_i}{2} \log(1 - p_i) \right) + \lambda ||h||^2_{\mathcal{F}}, \tag{5}$$

where $p_i = p(\mathbf{x}_i)$. The output $p(\mathbf{x})$ of such a machine will be a posterior probability. Minimizing this error function will not directly produce a sparse machine, but a modification to this method can produce sparse kernel machines [21].

This chapter presents modifications to SVMs which yield posterior probabilities, while still maintaining their sparseness. First, the chapter reviews recent work in modifying SVMs to produce probabilities. Second, it describes a method for fitting a sigmoid that maps SVM outputs to posterior probabilities. Finally, the SVM plus sigmoid combination is compared to a regularized likelihood fit using the same kernel on three different data-mining-style data sets.

1.1 Recent Work

In [19, sec. 11.11], Vapnik suggests a method for mapping the output of SVMs to probabilities by decomposing the feature space $\mathcal{F}$ into a direction orthogonal to the separating hyperplane, and all of the $N-1$ other dimensions of the feature space.The direction orthogonal to the separating hyperplane is parameterized by t (a scaled version of $f(\mathbf{x})$), while all of the other directions are parameterized by a vector $\mathbf{u}$. In full generality, the posterior probability depends on both t and $\mathbf{u}$: $P(y = 1|t, \mathbf{u})$. Vapnik proposes fitting this probability with a sum of cosine terms:

$$P(y = 1|t, \mathbf{u}) = a_0(\mathbf{u}) + \sum_{n=1}^{N} a_n(\mathbf{u}) \cos(nt). \tag{6}$$

The coefficients of the cosine expansion will minimize a regularized functional [19, eqn. 7.93], which can be converted into a linear equation for the a_n that depends on the value of $\mathbf{u}$ for the current input being evaluated.

Preliminary results for this method, shown in [19, fig. 11.8], are promising. However, there are some limitations that are overcome by the method of this chapter. For example, the Vapnik method requires a solution of a linear system for every evaluation of the SVM. The method of this chapter does not require a linear system solver call per evaluation because it averages the $P(y = 1|f)$ over all $\mathbf{u}$. The price of this efficiency is that dependencies of $P(y = 1|f)$ on $\mathbf{u}$ cannot be modeled. Another interesting feature of the Vapnik method is that the sum of the cosine terms is not constrained to lie between 0 and 1, and is not constrained to be monotonic in f. See, for example, [19, fig. 11.8]. There is a very strong prior for considering the probability

$P(y=1|f)$ to be monotonic in f, since the SVM is trained to separate most or all of the positive examples from the negative examples.

Another method for fitting probabilities to the output of an SVM is to fit Gaussians to the class-conditional densities $p(f|y=1)$ and $p(f|y=-1)$. This was first proposed by Hastie and Tibshirani in [7], where a single tied variance is estimated for both Gaussians. The posterior probability rule $P(y=1|f)$ is thus a sigmoid, whose slope is determined by the tied variance. Hastie and Tibshirani [7] then adjust the bias of the sigmoid so that the point $P(y=1|f)=0.5$ occurs at $f=0$. This sigmoid is monotonic, but the single parameter derived from the variances may not accurately model the true posterior probability.

One can also use a a more flexible version of the Gaussian fit to $p(f|y=\pm 1)$. The mean and the variance for each Gaussian is determined from a data set. Bayes' rule can be used to compute the posterior probability via:

$$P(y=1|f) = \frac{p(f|y=1)P(y=1)}{\sum_{i=-1,1} p(f|y=i)P(y=i)}, \tag{7}$$

where $P(y=i)$ are prior probabilities that can be computed from the training set[1]. In this formulation, the posterior is an analytic function of f with form:

$$P(y=1|f) = \frac{1}{1+\exp(af^2+bf+c)}. \tag{8}$$

There are two issues with this model of SVM outputs. First, the posterior estimate derived from the two-Gaussian approximation violates the strong monotonic prior mentioned above: the function in (8) is non-monotonic. Second, the assumption of Gaussian class-conditional densities is often violated (see Figure 1).

2 Fitting a Sigmoid After the SVM

2.1 Motivation

Instead of estimating the class-conditional densities $p(f|y)$, we use a parametric model to fit the posterior $P(y=1|f)$ directly. The parameters of the model are adapted to give the best probability outputs.

The form of the parametric model can be inspired by looking at empirical data. Figure 1 shows a plot of the class-conditional densities $p(f|y=\pm 1)$ for a linear SVM trained on a version of the UCI Adult data set (see [15]). The plot shows histograms of the densities (with bins 0.1 wide), derived from threefold cross-validation. These densities are very far away from Gaussian. There are discontinuities in the derivatives of both densities at both the positive margin $f=1$ and the negative margin $f=-1$. These discontinuities are not surprising, considering that the cost function (3) also has discontinuities at the margins. Theory to explain the form of these class-conditional densities is currently under development.

The class-conditional densities between the margins are apparently exponential. Bayes' rule (7) on two exponentials suggests using a parametric form of a sigmoid:

$$P(y=1|f) = \frac{1}{1+\exp(Af+B)}. \tag{9}$$

This sigmoid model is equivalent to assuming that the output of the SVM is proportional to the log odds of a positive example. This sigmoid model is different from that proposed in [7] because it has two parameters trained discriminatively, rather than one parameter estimated from a tied variance.

[1] This model for SVM output probabilities was independently proposed and used for speaker identification in a talk by C. J. C. Burges at the 1998 NIPS SVM workshop.

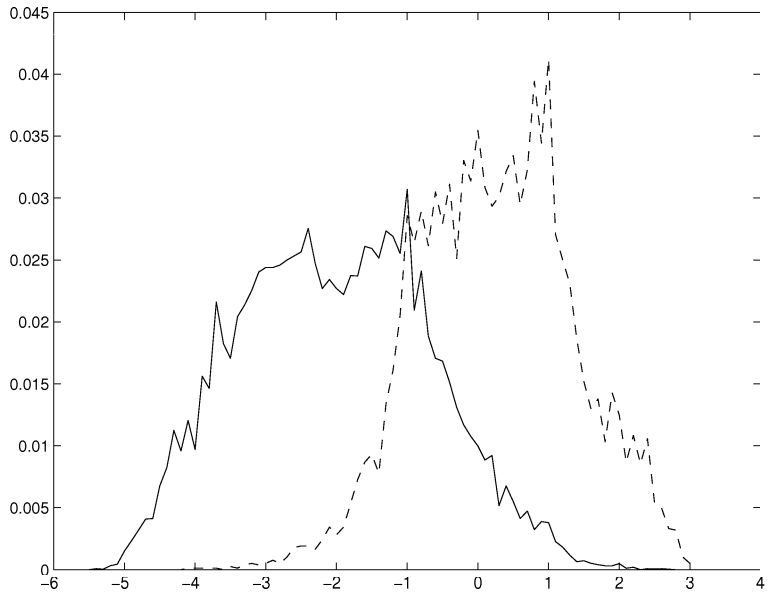

Figure 1: The histograms for $p(f|y = \pm 1)$ for a linear SVM trained on the Adult data set. The solid line is $p(f|y = -1)$, while the dashed line is $p(f|y = 1)$. Notice that these histograms are not Gaussian.

The sigmoid fit works well, as can be seen in Figure 2. The data points in Figure 2 are derived by using Bayes' rule on the histogram estimates of the class-conditional densities in Figure 1. For a linear SVM trained on the Adult data set [15], the sigmoid fits the non-parametric estimate extremely well, even beyond the margins. On the other sets and other kernels described in this chapter, the sigmoid fits reasonably well, with a small amount of bias beyond the margins. The non-parametric model of posterior probability for handwritten digits shown in [19, fig. 11.8] is also very close to a sigmoid. Therefore, the sigmoid posterior model seems to be close to the true model.

One can also view the sigmoid function as a linearization (in log-odds space) of the posterior in (8). As long as $A < 0$, the monotonicity of (9) is assured. Even if, in some cases, the class-conditional densities are close to Gaussian, the sigmoid fit is still appropriate and valuable.

2.2 Fitting the Sigmoid

The parameters A and B of (9) are fit using maximum likelihood estimation from a training set (f_i, y_i). First, let us define a new training set (f_i, t_i), where the t_i are target probabilities defined as:

$$t_i = \frac{y_i + 1}{2}. \tag{10}$$

The parameters A and B are found by minimizing the negative log likelihood of the training data, which is a cross-entropy error function:

$$\min -\sum_i t_i \log(p_i) + (1 - t_i)\log(1 - p_i), \tag{11}$$

where

$$p_i = \frac{1}{1 + \exp(Af_i + B)}. \tag{12}$$

The minimization in (11) is a two-parameter minimization. Hence, it can be performed using any number of optimization algorithms. For robustness, the experiments in this paper were

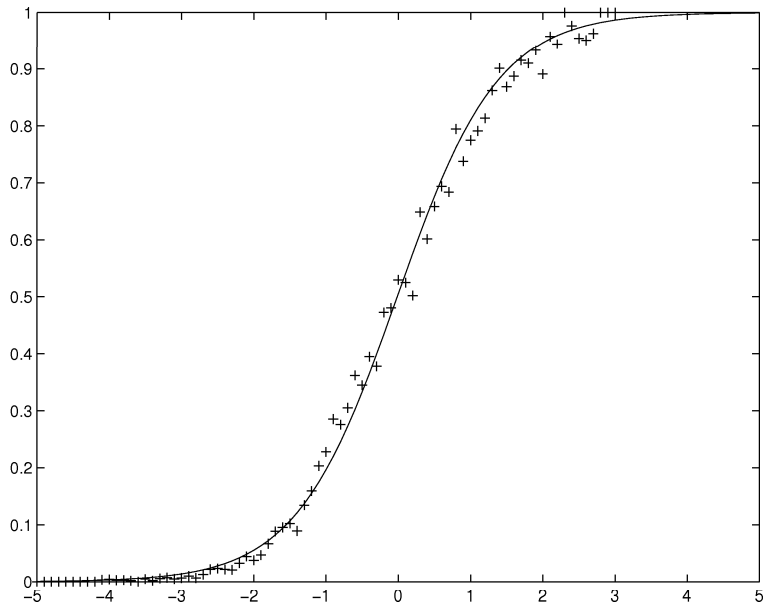

Figure 2: The fit of the sigmoid to the data for a linear SVM on the Adult data set (as in Figure 1). Each plus mark is the posterior probability computed for all examples falling into a bin of width 0.1. The solid line is the best-fit sigmoid to the posterior, using the algorithm described in this chapter.

performed using a model-trust minimization algorithm [6], whose pseudo-code is shown in Appendix 5.

Two issues arise in the optimization of (11): the choice of the sigmoid training set (f_i, y_i), and the method to avoid over-fitting this set.

The easiest training set to use is simply the same training examples used to fit the SVM. That is, $f_i = f(x_i)$, where x_i is the ith training example. However, the training of the SVM causes the SVM outputs f_i to be a biased estimate of the distribution of f out of sample. For examples at the margin, the f_i are forced to have absolute value exactly 1, which certainly will not be a common value for test examples. The training examples that fail the margin $(1 - y_i f_i > 0)$ are also subtly biased, since the f_i are pushed towards the margin by the corresponding α_i. Only the f_i that are beyond the margin are substantially unbiased.

For linear SVMs, the bias introduced by training usually is not severe. In almost all cases, a maximum of $N + 1$ support vectors will lie on the margin (for an input dimensionality of N), which is usually a small fraction of the training set. Also, for many real-world problems that use linear SVMs, optimal performance is reached for small C, which causes the bias on the margin failures to become small. Therefore, for linear SVMs, it often possible to simply fit the sigmoid on the training set.

For non-linear SVMs, the support vectors often form a substantial fraction of the entire data set, especially when the Bayes error rate for the problem is high [19]. Through empirical experiments, fitting a sigmoid to the training set of non-linear SVMs sometimes leads to disastrously biased fits. Therefore, we must form an unbiased training set of the output of the SVM f_i.

One method for forming an unbiased training set is to approximate leave-one-out estimates of f_i, as described in [this volume, chapter by Weston]. However, this either requires the solution of a linear system for every data point in the training set, or a re-run of an SVM solver at every data point, which can be computationally expensive.

There are two computationally inexpensive methods for deriving an unbiased training set: generating a hold-out set and cross-validation. To use a hold out set, a fraction of the training

set (typically 30%) is not used to train the SVM, but is used to train the sigmoid. This same hold-out set can be used to estimate other parameters of the system, such as kernel choice, kernel parameters, and C. Once A, B, and all of the other system parameters are determined from the hold out set, the main SVM can be re-trained on the entire training set. If SVM training scales roughly quadratically with training set size [16, 9], then the hold-out set will be only 1.5 times slower than simply training on the entire data set. Because determining the system parameters is often unavoidable, determining A and B from the hold-out set may not incur extra computation with this method.

Cross-validation is an even better method than a hold-out set for estimating the parameters A and B [10]. In three-fold cross-validation, the training set is split into three parts. Each of three SVMs are trained on permutations of two out of three parts, and the f_i are evaluated on the remaining third. The union of all three sets of f_i can form the training set of the sigmoid (and also can be used to adjust the SVM system parameters). Cross-validation produces larger sigmoid training sets than the hold-out method, and hence gives a lower variance estimate for A and B. Three-fold cross-validation takes approximately 2.2 times as long as training a single SVM on an entire training set. All of the results in this chapter are presented using three-fold cross-validation.

Even with cross-validated unbiased training data, the sigmoid can still be overfit. For example, in the Reuters data set [5, 8], some of the categories have very few positive examples which are linearly separable from all of the negative examples. Fitting a sigmoid for these SVMs with maximum likelihood will simply drive the parameter A to a very large negative number, even if the positive examples are reweighted. There can be an infinite number of solutions with infinitely steep sigmoids when the validation set is perfectly separable. Therefore, we must add a regularization term to prevent overfitting to a small number of examples.

Regularization requires either a prior model for the parameter space (A, B), or a prior model for a distribution of out-of-sample data. One can imagine using a Gaussian or Laplacian prior on A. However, there is always one free parameter in the prior distribution (e.g., the variance). This free parameter can be set using cross-validation or Bayesian hyperparameter inference [11], but these methods add complexity to the code.

A simpler method is to create a model of out-of-sample data. One model is to assume that the out-of-sample data is simply the training data perturbed with Gaussian noise. This is the model behind Parzen windows [4, 19]. However, this model still has a free parameter.

The sigmoid fit in this chapter uses a different out-of-sample model: out-of-sample data is modelled with the same empirical density as the sigmoid training data, but with a finite probability of opposite label. In other words, when a positive example is observed at a value f_i, we do not use $t_i = 1$, but assume that there is a finite chance of opposite label at the same f_i in the out-of-sample data. Therefore, a value of $t_i = 1 - \epsilon_+$ will be used, for some ϵ_+. Similarly, a negative example will use a target value of $t_i = \epsilon_-$. Using a non-binary target does not require any modification to the maximum likelihood optimization code. Because (11) is simply the Kullback-Liebler divergence between f_i and t_i, the function is still well-behaved, even for non-binary t_i.

The probability of correct label can be derived using Bayes' rule. Let us choose a uniform uninformative prior over probabilities of correct label. Now, let us observe N_+ positive examples. The MAP estimate for the target probability of positive examples is

$$t_+ = \frac{N_+ + 1}{N_+ + 2}. \tag{13}$$

Similarly, if there are N_- negative examples, then the MAP estimate for the target probability of negative examples is

$$t_- = \frac{1}{N_- + 2}. \tag{14}$$

These targets are used instead of $\{0, 1\}$ for all of the data in the sigmoid fit.

These non-binary targets value are Bayes-motivated, unlike traditional non-binary targets for neural networks [18]. Furthermore, the non-binary targets will converge to $\{0, 1\}$ when the training set size approaches infinity, which recovers the maximum likelihood sigmoid fit.

The pseudo-code in Appendix 5 shows the optimization using the Bayesian targets.

3 Empirical Tests

There are at least two experiments to determine the real-world performance of the SVM+sigmoid combination. First, the SVM+sigmoid can be compared to a plain SVM for misclassification rate. Assuming equal loss for Type I and Type II errors, the optimal threshold for the SVM+sigmoid is $P(y = 1|f) = 0.5$, while the optimal threshold for the SVM is $f = 0$. This first experiment checks to see if the 0 threshold is optimal for SVMs.

The second experiment is to compare the SVM+sigmoid with a kernel machine trained to explicitly maximize a log multinomial likelihood. For the linear kernel case, this is equivalent to comparing a linear SVM to regularized logistic regression. The purpose of the second experiment is to check the quality of probability estimates by the SVM+sigmoid hybrid combination, and see if the error function (3) causes fewer misclassifications than (5). Three different classification

Task	Training Set Size	Testing Set Size	C	Number of Inputs	Number of SVMs
Reuters Linear	9603	3299	0.08	300	118
Adult Linear	32562	16282	0.05	123	1
Adult Quadratic	1605	16282	0.3	123	1
Web Linear	49749	21489	1.0	300	1
Web Quadratic	2477	21489	10.0	300	1

Table 1: Experimental Parameters

tasks were used. The first task is determining the category of a Reuters news article [5, 8]. The second task is the UCI Adult benchmark of estimating the income of a household given census form data [13], where the input vectors are quantized [15]. The third task is determining the category of a web page given key words in the page [15]. The Reuters task is solved using a linear SVM, while the Adult and Web tasks are solved with both linear and quadratic SVMs. The parameters of the training are shown in Table 1. The regularization terms are set separately for each algorithm, via performance on a hold-out set. The C value shown in Table 1 is for the SVM+sigmoid. The sigmoid parameters are estimated using three-fold cross-validation. The quadratic kernel for the Adult task is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left(\frac{\mathbf{x}_i \cdot \mathbf{x}_j + 1}{14} \right)^2, \tag{15}$$

while the quadratric kernel for the Web task is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left(\frac{\mathbf{x}_i \cdot \mathbf{x}_j + 1}{12} \right)^2, \tag{16}$$

The constants 12 and 14 are taken from the average over each data set of the dot product of an example with itself. This normalization keeps the kernel function in a reasonable range.

Table 2 shows the results of these experiments. The table lists the number of errors for a raw SVM, an SVM+sigmoid, and a regularized likelihood kernel method. It also lists the negative log likelihood of the test set for SVM+sigmoid and for the regularized likelihood kernel method. McNemar's test [3] was used to find statistically significant differences in classification error rate, while the Wilcoxson signed rank test [14] is used to find significant differences in the

log likelihood. Both of these tests examine the results of a pair of algorithms on every example in the test set. In Table 2, underlined entries are pairwise statistically significantly better than all non-underlined entries, while not statistically significantly better than any other underlined entry. A significance threshold of $p = 0.05$ is used.

Task	Raw SVM Number of Errors	SVM + Sigmoid Number of Errors	Regularized Likelihood Number of Errors	SVM + Sigmoid $-\log(p)$ Score	Regularized Likelihood $-\log(p)$ Score
Reuters Linear	1043	<u>963</u>	1060	<u>3249</u>	3301
Adult Linear	2441	2442	2434	5323	<u>5288</u>
Adult Quadratic	2626	<u>2554</u>	2610	<u>5772</u>	5827
Web Linear	260	265	<u>248</u>	1121	<u>958</u>
Web Quadratic	<u>444</u>	<u>452</u>	507	1767	<u>2163</u>

Table 2: Experimental Results

3.1 Discussion

Three interesting results were observed from these experiments. First, adding a sigmoid sometimes improves the error rate of a raw SVM: a zero threshold is not necessarily Bayes optimal. For the Reuters Linear and Adult Quadratic tasks, the sigmoid threshold was significantly better than the standard zero threshold. For both of these tasks, the ratio of the priors $P(y = -1)/P(y = 1)$ is far from one, which will tend to push the Bayes optimal threshold away from zero. For example, on the Adult Quadratic task, the threshold $P(y = 1|f) = 0.5$ corresponds to a threshold of $f = -0.1722$, which is simply a more optimal threshold than zero. The VC bounds on the generalization error [19] do not guarantee that the zero threshold is Bayes optimal.

The second interesting result is that adding the sigmoid produces probabilities of roughly comparable quality to the regularized likelihood kernel method. For three of the five tasks, the regularized likelihood yields significantly better probabilities. For the Web Quadratic task, the SVM+sigmoid has a better overall log likelihood, but the Wilcoxon rank test prefers the regularized likelihood kernel method because more data points are more accurate with the latter method.

The third interesting result is that neither the SVM+sigmoid nor the regularized likelihood kernel machine is a completely dominant method for either error rate or log likelihood. The SVM+sigmoid makes fewer errors than the regularized likelihood kernel method for three out of five tasks, while the regularized likelihood method makes fewer errors for one out of five tasks. This result is somewhat surprising: the SVM kernel machine is trained to minimize error rate, while the regularized likelihood is trained to maximize log likelihood. These experiments indicate that, when all other factors (e.g., kernel choice) are held constant, the difference in performance between (3) and (5) is hard to predict *a priori*.

Finally, it is interesting to note that there are other kernel methods that produce sparse machines without relying on an RKHS. One such class of methods penalize the ℓ_1 norm of the function h in (3), rather than the RKHS norm [12, 2] (see, for example, [this volume, chapter by Mangasarian]). Fitting a sigmoid after fitting these sparse kernel machines may, in future work, yield reasonable estimates of probabilities.

4 Conclusions

This chapter presents a method for extracting probabilities $P(\text{class}|\text{input})$ from SVM outputs, which is useful for classification post-processing. The method leaves the SVM error function (3) unchanged. Instead, it adds a trainable post-processing step which is trained with regularized binomial maximum likelihood. A two parameter sigmoid is chosen as the post-processing, since it matches the posterior that is empirically observed. Finally, the SVM+sigmoid combination is compared to a raw SVM and a kernel method entirely trained with regularized maximum likelihood. The SVM+sigmoid combination preserves the sparseness of the SVM while producing probabilities that are of comparable quality to the regularized likelihood kernel method.

Acknowledgements

I would like to thank Chris Bishop for valuable advice during the writing of the paper.

5 Appendix: Pseudo-code for the Sigmoid Training

This appendix shows the pseudo-code for the training is shown below. The algorithm is a model-trust algorithm, based on the Levenberg-Marquardt algorithm [17].

```
Input parameters:
  out = array of SVM outputs
  target = array of booleans: is ith example a positive example?
  prior1 = number of positive examples
  prior0 = number of negative examples
Outputs:
  A, B = parameters of sigmoid

A = 0
B = log((prior0+1)/(prior1+1))
hiTarget = (prior1+1)/(prior1+2)
loTarget = 1/(prior0+2)
lambda = 1e-3
olderr = 1e300
pp = temp array to store current estimate of probability of examples
set all pp array elements to (prior1+1)/(prior0+prior1+2)
count = 0
for it = 1 to 100 {
  a = 0, b = 0, c  = 0, d = 0, e = 0
// First, compute Hessian & gradient of error function
// with respect to A & B
  for i = 1 to len {
     if (target[i])
        t = hiTarget
     else
        t = loTarget
     d1 = pp[i]-t
     d2 = pp[i]*(1-pp[i])
     a += out[i]*out[i]*d2
     b += d2
     c += out[i]*d2
     d += out[i]*d1
     e += d1
```

```
  }
// If gradient is really tiny, then stop
  if (abs(d) < 1e-9 && abs(e) < 1e-9)
     break
  oldA = A
  oldB = B
  err = 0
// Loop until goodness of fit increases
  while (1) {
    det = (a+lambda)*(b+lambda)-c*c
    if (det == 0) {  // if determinant of Hessian is zero,
     // increase stabilizer
      lambda *= 10
      continue
    }
    A = oldA + ((b+lambda)*d-c*e)/det
    B = oldB + ((a+lambda)*e-c*d)/det
    // Now, compute the goodness of fit
    err = 0;
    for i = 1 to len {
      p = 1/(1+exp(out[i]*A+B))
      pp[i] = p
      // At this step, make sure log(0) returns -200
      err -= t*log(p)+(1-t)*log(1-p)
    }
    if (err < olderr*(1+1e-7)) {
      lambda *= 0.1
      break
    }
    // error did not decrease: increase stabilizer by factor of 10
    // & try again
    lambda *= 10
    if (lambda >= 1e6) // something is broken. Give up
      break
  }
  diff = err-olderr
  scale = 0.5*(err+olderr+1)
  if (diff > -1e-3*scale && diff < 1e-7*scale)
    count++
  else
    count = 0
  olderr = err
  if (count == 3)
    break
}
```

References

[1] H. Bourlard and N. Morgan. A continuous speech recognition system embedding MLP into HMM. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 186–193. Morgan Kaufmann, San Mateo, CA, 1990.

[2] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, May 1995.

[3] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1924, 1998.

[4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[5] S. Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems*, 13(4), 1998. In: M.A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt: Trends and Controversies — Support Vector Machines.

[6] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, 1981.

[7] T. Hastie and R. Tibshirani. Classification by pairwise coupling. Technical report, Stanford University and University of Toronto, 1996. http://www-stat.stanford.edu/ trevor/Papers/2class.ps.

[8] T. Joachims. Text categorization with support vector machines. In *European Conference on Machine Learning (ECML)*, 1998.

[9] T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

[10] M. Kearns. A bound on the error of cross validation using the approximation and estimation rates, with consequences for the training-test split. *Neural Computation*, 9(5):1143–1161, 1997.

[11] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[12] O. L. Mangasiarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[13] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[14] F. Mosteller and R. Rourke. *Sturdy Statistics.* Addison-Wesley, Reading, MA, 1973.

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

[16] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.).* Cambridge University Press, Cambridge, 1992.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, Cambridge, MA, 1986.

[19] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998. forthcoming.

[20] G. Wahba. Multivariate function and operator estimation, based on smoothing splines and reproducing kernels. In M. Casdagli and S. Eubank, editors, *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Proc. Vol XII*, pages 95–112. Addison-Wesley, 1992.

[21] G. Wahba. The bias-variance tradeoff and the randomized GACV. In D. A. Cohn M. S. Kearns, S. A. Solla, editor, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, Cambridge, MA, 1999. To appear.

[22] G. Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized GACV. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 69–88, Cambridge, MA, 1999. MIT Press.