Soft Margin Bayes-Point-Machine Classification via Adaptive Direction Sampling

Karsten Vogt and Jörn Ostermann

Institut für Informationsverarbeitung,
Leibniz Universität Hannover, Germany
(vogt, office)@tnt.uni-hannover.de

Abstract. Supervised machine learning is an important building block for many applications that involve data processing and decision making. Good classifiers are trained to produce accurate predictions on a training set while also generalizing well to unseen data. To this end, Bayes-Point-Machines (BPM) were proposed in the past as a generalization of margin maximizing classifiers, such as Support-Vector-Machines (SVM). For BPMs, the optimal classifier is defined as an expectation over an appropriately chosen posterior distribution, which can be estimated via Markov-Chain-Monte-Carlo (MCMC) sampling. In this paper, we propose three improvements on the original BPM classifier. Our new statistical model is regularized based on the sample size and allows for a true soft-margin formulation without the need to hand-tune any nuisance parameters. Secondly, this model can handle multi-class problems natively. Finally, our fast adaptive MCMC sampler uses Adaptive Direction Sampling (ADS) and can generate a sample from the proposed posterior with a runtime complexity quadratic in the size of the training set. Therefore, we call our new classifier the Multi-class-Soft-margin-Bayes-Point-Machine (MS-BPM). We have evaluated the generalization capabilities of our approach on several datasets and show that our soft-margin model significantly improves on the original BPM, especially for small training sets, and is competitive with SVM classifiers. We also show that class membership probabilities generated from our model improve on Platt-scaling, a popular method to derive calibrated probabilities from maximum-margin classifiers.

1 Introduction

Models of statistical learning and classification methods are vital components in many current applications, such as autonomous driving [13], natural language processing [2] and game AI [23]. A challenging aspect of machine learning concerns the balancing of classification accuracy and generalizability on unseen data, especially if only few training examples are available.

For classification problems, supervised learning has the aim to derive a decision function $y = h(\boldsymbol{x})$ from a labeled training set $Tr = (\boldsymbol{x}_i, y_i)_{i=1}^N$, where $\boldsymbol{x} \in \mathbb{R}^F$ are feature vectors from an F-dimensional feature space and $y \in C$ are labels chosen from a finite set of class labels. Different theoretical models and

learning algorithms have been proposed in the past. Support-Vector-Machines (SVM), originally developed by Cortes & Vapnik [5], have retained widespread usage due to their excellent theoretical underpinnings and their competitive performance on many datasets. Other vector machine approaches were later proposed to alleviate some of the shortcomings of SVMs. These include, for example, extensions for multi-class problems [1, 26], probabilistic decision functions [26, 25] and highly sparse solutions [25].

Herbrich et al. [7] presented their own take of a vector-machine classifier based on the concept of a Bayesian point estimate of the optimal parametrized decision plane. This Bayesian-Point-Machine (BPM) ties maximum-margin classification into a larger framework of Bayesian decision making. As a nontrivial byproduct, learning a BPM constructs an approximation of the Bayesian posterior over all classification models. This posterior distribution can, for example, be used to inexpensively derive various statistics for use in more complex decision models or to compute calibrated class membership probabilities. In this paper, we propose three improvements to the BPM classifier. Firstly, the BPM is based on a regularized hard-margin model. Although the BPM has been proven to have good generalization capabilities for the hard-margin case, this was never conclusively shown for the soft-margin variant. Our experiments in Section 5 show that this may not be the case. The regularization also introduces an additional hyperparameter into the model, which must be carefully tuned. To solve these problems, we will substitute the statistical data model with a true soft-margin model that contains no nuisance parameters. Secondly, we extend this new formulation to handle multi-class problems natively. These changes necessitate the development of a new sampling approach. Therefore, we introduce a novel sampling algorithm that can create a sample from our posterior with a runtime complexity of $O(N^2|C| + N|C|^2)$. Our statistical classifier will subsequently be called the Multi-class-Soft-margin-Bayes-Point-Machine (MS-BPM).

Our paper is composed as follows. Section 2 provides a brief introduction to BPMs. Sections 3 and 4 then introduce our new soft-margin model and a fast multi-class sampling algorithm. We evaluate the generalization capabilities and class membership probabilities of the MS-BPM in Section 5 and conclude with Section 6.

2 Bayes-Point-Machines

The BPM utilizes a very simple statistical model. In the hard-margin case, all classifiers that manage to perfectly separate a training set Tr receive a uniform likelihood, while classifiers that generate at least one training error are discarded. The set of valid classifiers can be described by a convex polytope called the version space. Using a Bayes estimator with an assumed L_2-loss, the point estimate of an optimal decision plane is simply the center of mass of this version space. This point is also called the Bayes-point classifier. It was shown that the BPM generalizes the concept of maximum-margin classification and will often generalize at least as well as the SVM [7]. The soft-margin case, where some margin is

sacrificed to mitigate the effects of outliers and overlapping class distributions, was handled for the kernelized version of the algorithm by regularizing the Gram matrix. In effect, this allows for some misclassified training examples near the decision boundary. This approach introduces a tunable dataset-dependent hyperparameter whose value must be optimized, e.g. using cross-validation.

Since the Bayes-point can be formulated as an expectation over the posterior distribution of classification models, sampling methods based on the Markov-Chain-Monte-Carlo (MCMC) methodology can be an effective way of estimation [21]. In the original works, a billiard scheme was proposed to generate a sample from the uniformly distributed version space [7, 22]. Later works improved the computational efficiency using the Expectation Propagation algorithm by approximating the posterior under the assumption of local Gaussianity [14].

In the next section, we present our new statistical model that directly models the soft-margin case without introducing an additional hyperparameter. Furthermore, our model can be straightforwardly extended to multi-class problems.

3 Statistical Model of Soft Margin Classification

Sampling from the distribution of decision boundaries requires the definition of a posterior distribution $p(\boldsymbol{\beta}|\, Tr)$. This section therefore introduces and elucidates the required components of our statistical multi-class soft-margin model. This includes a parametrization $\boldsymbol{\beta}$ of the decision boundaries, a data-dependent likelihood term $l(\, Tr|\boldsymbol{\beta})$ and a prior distribution $p(\boldsymbol{\beta})$ for the model parameters.

3.1 Parametrization

A non-probabilistic classifier can be parametrized using any partitioning function that subdivides the feature-space into $|C|$ partitions. To simplify the sampling, we will focus on linear partitionings. Non-linear decision boundaries can then be modeled via non-linear projections of the feature-space, e.g. using the kernel trick [8]. Following the example of generalized linear models [10], each class c has an associated linear predictor $f_c(\boldsymbol{x})$. Given a feature-vector $\boldsymbol{x}$, we always choose the class which produces the maximum response:

$$h(\boldsymbol{x}) = \underset{c \in C}{\arg\max}\ f_c(\boldsymbol{x}) = \underset{c \in C}{\arg\max}\ \boldsymbol{\beta}_c^T \boldsymbol{x} + \beta_{c,0}\ . \tag{1}$$

The parameters $\boldsymbol{\beta}_c$ are the importance weights of the linear predictor for class c and $\beta_{c,0}$ its intercept. We will further call a specific instantiation parametrized by the vector $\boldsymbol{\beta}$ a configuration. Furthermore, the parameters of this model can be reduced by subtracting $\boldsymbol{\beta}_1^T \boldsymbol{x} + \beta_{1,0}$ from all predictor functions. In this formulation, the anchor class $c = 1$ will always produce a zero response, while the remaining functions model the relative predictions for each class compared to the anchor class.

The remaining model parameters are still redundant in regard to uniform scaling. Herbich et al. [7] solved this problem for the two-class case by reparameterizing the model using a hyperspherical coordinate transform and normalizing

the radius to 1. We argue that the original cartesian parametrization allows for a simpler MCMC sampling algorithm. We solve the redundancy in a more classical fashion by introducing appropriate priors on the model parameters.

3.2 Data Likelihood

The likelihood used by Herbrich et al. [7] is based on a simplified data model that is only valid for the hard-margin case. All configurations that achieve zero empirical training errors have a constant likelihood, while all configurations that produce at least one error are discarded. In case of outliers and overlapping class distributions, it may prove beneficial to admit at least some errors. In the original formulation, this is achieved by ignoring training errors that are geometrically close to the decision plane. For our soft-margin model, we would like to derive a likelihood that is more closely related to a well-defined data generating process. The likelihood of the entire training dataset Tr is usually defined by its log-loss:

$$l(Tr|\boldsymbol{\beta}) = \exp(\text{LogLoss}(Tr, \boldsymbol{\beta})) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{\beta}) \ . \tag{2}$$

The logistic regression [10], for example, substitutes the class label probabilities $p(y_i|\boldsymbol{x}_i, \boldsymbol{\beta})$ with the logistic function $(1 + \exp(\boldsymbol{x}_i^t \cdot \boldsymbol{\beta}))^{-1}$. The BPM, on the other hand, assumes a 0–1 loss. We define $p(y_i|\boldsymbol{x}_i, \boldsymbol{\beta}) = 1_{y_i = h(\boldsymbol{x}_i)}$. It can be easily seen that even a single misclassified example pulls the entire likelihood down to zero, which is highly problematic for the non-separable case. Intuitively, this can be interpreted as the BPM model placing infinite confidence on the decisions of the learned classifier. In order to handle overlapping class distributions, we propose to regularize the model by additionally estimating the classification confidences from the data. Our modified likelihood reads as

$$l(Tr|\boldsymbol{\beta}, \boldsymbol{\pi}) = \prod_{i=1}^{N} \pi_{y_i, h(\boldsymbol{x}_i)} \ , \tag{3}$$

where $\pi_{c,p} \in (0, 1)$ is the probability that an example x_i with a true class label $c = y_i$ is classified as class $p = h(\boldsymbol{x}_i)$. These parameters would require dataset-dependent tuning. We can improve the robustness of our model in regard to the parameters $\boldsymbol{\pi}$ by placing an appropriate prior distribution on them, thus creating a hierarchical model. In the Bayesian spirit, we then marginalize these parameters. Assuming Dirichlet priors with parameters $\boldsymbol{\alpha}$, this produces the likelihood

$$\begin{aligned} l_{\text{dm}}(Tr|\boldsymbol{\beta}, \boldsymbol{\alpha}) &= \int l(Tr|\boldsymbol{\beta}, \boldsymbol{\pi}) \cdot p_{\text{Dir}}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \ \mathrm{d}\boldsymbol{\pi} \\ &\propto \prod_{c=1}^{|C|} \frac{\prod_{p=1}^{|C|} \Gamma(M_{c,p} + \alpha_{c,p})}{\Gamma(\sum_{p=1}^{|C|} M_{c,p} + \alpha_{c,p})} \ , \end{aligned} \tag{4}$$

where $\Gamma(.)$ is the Gamma function and $M_{c,p}$ are the counts of how many training examples from class c were assigned to partition p. This model is also called a Dirichlet-multinomial or multivariate Pólya distribution [15]. In our model, the confidence we place on a classifier is largely based on the number of training examples it was derived from. In the separable case our regularized model will tend towards the BPM model for large N. Yet we still require a principled way of tuning the α parameters. General pointers of parametrizing Dirichlet distributions can be gleaned from the statistical literature. Generally, we get an uninformative flat prior by setting $\alpha_{c,p} = 1$. It turns out that this is not a sensible choice for classification models. As can be seen in Fig. 1, such a prior would place too much weight on models that exhibit high empirical errors. We need to guarantee that reductions in error always corresponds with increases in likelihood. This property trivially holds for the weakly informative prior with $\alpha_{c,p} = 1$, $c \neq p$ and $\alpha_{c,c} = 1 + N$. Furthermore, we will introduce the regularization parameter ν by setting $\alpha_{c,c} = 1 + \frac{N}{\nu}$. This way, setting $\nu \to \infty$ produces the uninformative prior while $\nu \to 0$ strongly penalizes misclassifications and corresponds in the limit with the original BPM model. Sensible choices for ν lie in the interval $(0, 1]$, but our model is largely robust to the specific choice of ν. For all experiments, we simply set it fixed to $\nu = 1$.

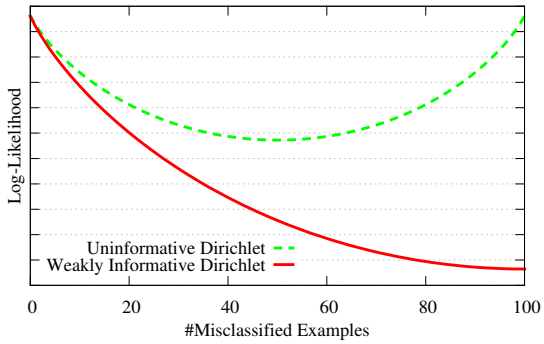


Fig. 1: Comparison of the log-likelihoods for a small training set ($N = 100$) plotted over the number of misclassified examples. Models under comparison are the Dirichlet-multinomial model with an uninformative Dirichlet prior and a weakly informative Dirichlet prior.

3.3 Feature Weight Prior

The parametrization introduced in Section 3.1 is redundant in regard to uniform scaling; that is $\boldsymbol{\beta} \equiv t \cdot \boldsymbol{\beta}$ for $t > 0$. This has the consequence that, given a uniform prior over the weights, the resulting posterior distribution will be improper. The typical solution involves replacing the uniform priors with proper ones. We

would expect a good prior distribution to be zero-centered, symmetrical, weakly-informative and simple to compute. In past works, the normal distribution and the Laplace distribution have been used frequently, especially since they have strong ties to L_2 and L_1 regularization, respectively.

$$p(\boldsymbol{\beta}) \propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{\beta}\|_2^2\right) \qquad \text{Normal Prior} \tag{5}$$

$$p(\boldsymbol{\beta}) \propto \exp\left(-\frac{1}{\sigma}\|\boldsymbol{\beta}\|_1\right) \qquad \text{Laplace Prior} \tag{6}$$

We can reduce the informativeness of the prior by increasing the scale parameter σ. The main difference between the two models is that the normal prior produces more dense solutions, while the Laplace prior prefers sparsity in the weight parameters. In more recent works, even more sparsity inducing prior distributions have been used [25]. The original BPM approach is restricted to dense models. Although, for computational reasons, our current implementation only uses dense normal priors, the MS-BPM method could be used in conjunction with any of these sparsity inducing priors.

4 Fast Multi-Class MCMC Sampler

In this section, we will introduce an efficient sampling scheme for our proposed statistical model. Multivariate sampling is achieved by performing fast univariate sampling along randomized search directions. Quick convergence can then be reached by adapting the distribution of search directions to the local properties of the posterior distribution.

4.1 Univariate vs Multivariate Sampling

The optimization of such classification problems based on a $0-1$ loss is known to be NP-hard [17]. Each training example splits the likelihood along $|C|-1$ half-spaces. As such, the potential number of equivalence-classes of different solutions can be stated as $2^{N\cdot(|C|-1)}$. Direct sampling from the multivariate posterior distribution quickly becomes prohibitively expensive even for small training sets. As can be seen in Fig. 2, the posterior distribution tends to be highly discontinuous, which is a direct result of our choice of the $0-1$ loss. The lack of useful gradient information also diminishes the effectiveness of a large class of MCMC algorithms, such as billiard schemes [16], Hamiltonian Monte Carlo [9] and covariance adaptive slice sampling [24]. One important observation is that arbitrary univariate sampling paths can only intersect at most $N \cdot (|C|-1)$ discontinuities. This implies that a univariate sampling algorithm could be implemented with a much lower computational complexity than a multivariate one. We describe such a sampling method in Section 4.2. To facilitate fast convergence of the Markov chain, it is essential to select useful search directions with a high probability. Our univariate sampler can be directly embedded in a number of higher-level sampling methods, such as Gibbs-sampling [21], Hit-and-Run [4] and Adaptive Direction Sampling (ADS) [6].

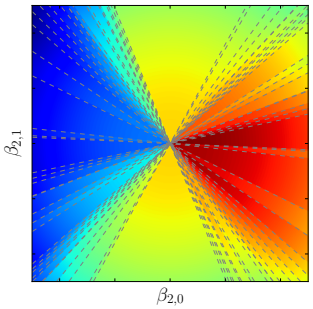


Fig. 2: A heatmap of the posterior distribution for a two-class toy problem with a 1D feature-space. Discontinuities are represented by gray stippled lines.

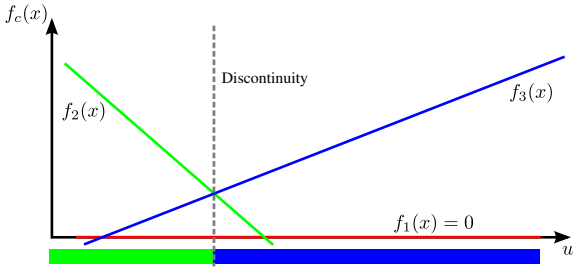


Fig. 3: Example of a three-class problem. The upper envelope of the linear predictor functions defines the partition membership intervals for one individual training example $\boldsymbol{x}$ along the search path. In this particular case, class 2 is selected left of the discontinuity and class 3 otherwise. Class 1 is never selected.

4.2 Efficient Univariate Sampling along Arbitrary Search Paths

Our fast univariate sampler will start at a configuration $\boldsymbol{\beta}_t$ and be given a search direction $\boldsymbol{d}$. The set of possible configurations

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + u \cdot \boldsymbol{d} \tag{7}$$

forms our search path for the next configuration. It is important to realize that $\boldsymbol{\beta}_t$ and $\boldsymbol{d}$ are fixed. As such, u is the random variable that we are actually sampling from. The task can also be stated as a problem of sampling from the posterior distribution conditioned on the search path in Eq. (7).

Our approach to this problem can be broken down into the following four steps:

1. Find all discontinuities along the search path.
2. Construct a discrete distribution of all intervals spanned by two consecutive discontinuities.
3. Draw an interval from this distribution.
4. Finally, draw a new configuration from the selected interval.

The first step can be directly tackled by substituting (7) into (1) as follows:

$$h(\boldsymbol{x}) = \underset{c \in C}{\arg\max} \ (\boldsymbol{\beta}_{t,c} + u \cdot \boldsymbol{d}_c)^T \boldsymbol{x}_i + (\beta_{t,c,0} + u \cdot d_{c,0}) \ .$$

Each example $\boldsymbol{x}_i$ in the training set will generate at most $|C| - 1$ discontinuities. These are situated at values for u, where the predictor functions of two classes become equal. By computing the upper envelope of all $|C|$ predictor functions,

e.g. using the convex-hull trick [18], we can find the partition assignment intervals for all configurations on the search path. The discontinuities actually mark the transitions between equivalence classes of solutions. Fig. 3 shows an example for a three-class problem.

Next for step 2, we store the discontinuities for all training examples in a list and sort them in ascending order. Our aim is to visit all discontinuities in a successive order. This allows us to efficiently update the counts $M_{c,p}$, which are required to evaluate the likelihood. We will start by initializing the counts for $u = -\infty$. Each discontinuity along the search path marks a point, where a training example switches from being classified as $p = p'$ to $p = p''$. We update the counts accordingly:

$$\begin{aligned} M_{y_i,p'} &= M_{y_i,p'} - 1 \\ M_{y_i,p''} &= M_{y_i,p''} + 1 \ . \end{aligned}$$

To compute the interval probability for the discretized sampling problem, we have to integrate over the conditional posterior density:

$$\begin{aligned} p_j &= \int_{u_j}^{u_{j+1}} l_{\mathrm{dm}}(M|\boldsymbol{\alpha}) \cdot p(\boldsymbol{\beta}_t + u \cdot \boldsymbol{d}) \ \mathrm{d}u \\ &= l_{\mathrm{dm}}(M|\boldsymbol{\alpha}) \cdot \int_{u_j}^{u_{j+1}} p(\boldsymbol{\beta}_t + u \cdot \boldsymbol{d}) \ \mathrm{d}u \ . \end{aligned} \tag{8}$$

Notice that the likelihood remains constant over the entire interval, since it only depends on the counts M. Integrating over the prior distribution is also trivial for the case of an isotropic normal distribution.

After selecting an interval from the discretized interval distribution for step 3, all that remains is to draw a new configuration from the selected interval in step 4. Once again, we make use of the fact that the likelihood is constant. Therefore, the problem reduces to sampling from the prior distribution, conditioned on the selected interval. In our case, this means to draw a configuration $\boldsymbol{\beta}_{t+1}$ from an appropriately parametrized truncated normal distribution.

The runtime complexity of our sampling algorithm can be stated as $O(N^2|C| + N|C|^2)$ for the kernelized version. For typical datasets, where $N \gg |C|$, this is equivalent to the fast approximated BPM approach in [7] ($O(N^2|C|)$) and compares favorably with support-vector-machines ($O(N^3|C|)$), relevance-vector-machines ($O(N^3|C|)$) and import-vector-machines ($O(N^2q^2|C|)$) (we assumed a o-vs-r scheme for the non multi-class methods). Of course, sampling based methods will usually also incur a much higher constant factor compared to optimization based learning algorithms, so this advantage may only play out for very large datasets.

4.3 Choosing Good Search Directions

Reliably choosing good search directions is of great importance. Two non-adaptive methods are Gibbs sampling [21] and Hit-and-Run sampling [4]. Gibbs sampling proposes to only use search directions that are parallel to the axes of the

parameter space. The sampler alternates between these directions using either a predefined schedule or random schedule. In the case that two or more of the parameters are highly correlated, the Markov chain may be required to temporarily assume a low-probability state in order to reach more promising parts of the posterior distribution. This property may cause slow convergence. Hit-and-Run sampling, on the other hand, chooses a uniformly sampled random search direction at each iteration. It is more robust and can often show surprisingly fast convergence [12]. An adaptive sampling scheme, which exploits knowledge about the local correlation structure between parameters, is expected to significantly improve convergence in most cases. One simple adaptive sampling method is the ADS scheme [6]. ADS works by sampling multiple Markov chains in parallel. At each step, one chain is randomly chosen to be iterated on. In contrast to Hit-and-Run sampling the search direction is however not chosen uniformly. Information from two other randomly selected chains is utilized in order to steer the search along the principal directions of the parameter-space. To avoid the sampler getting stuck in a particular subspace of the parameter-space, some precautions have to be made. Following the findings of Gilks et al. [6], it has proven effective to occassionaly use a search direction generated by a non-adaptive method. The sampling behavior is typically very robust in regard to this selection probability. In our implementation, e.g., we arbitrarily fixed it to select ADS with 85% probability and Hit-and-Run sampling with 15% probability.

5 Evaluation

In this section, we evaluate our proposed MS-BPM method. We use the original BPM model with soft-margin regularization and the SVM as baseline methods. For all experiments, we simulated 200 independent MCMC chains of length 50, using 1000 iterations for the ADS sampler. We set $\nu = 1$ as described in Section 3.2. The hyperparameters for the SVM and BPM classifiers were optimized using a grid-search approach. All methods use the same RBF kernel using the kernel γ that was selected during the grid-search for the SVM runs. The kernel parametrizations for all methods were implemented exactly as in [3].

5.1 UCI Datasets

Our main evaluation is based on the commonly used supervised learning datasets from the UCI database [11]. These seven datasets cover a range of different classification problems of varying size, feature space and number of classes. We show the validity of our method for small and large training sets by training on 10% and 50% bootstrap samples for each dataset. The presented values in Table 1 show the out-of-bag accuracies for 100 independent runs and their standard deviations. As can be seen, our MS-BPM method displays similar performance characteristics as the baseline SVM classifier, yet it does not require hand-tuning of any regularization hyperparameters. The original BPM approach for soft-margin classification regularized the Gram-matrix to allow for some empirical errors on

the training set. Our experiments show that this approach is not competetive with our improved statistical model on most datasets, and especially for small training sets. The large standard deviations also indicate some robustness problems that are not observable in our method. A Wilcoxon signed rank test shows with a 97.5% confidence level that our MS-BPM classifier significantly improves on the BPM classifier. The same test is inconclusive when used to compare the results of the MS-BPM and SVM classifiers.

Table 1: Out-of-bag accuracy estimates for the original regularized BPM,the SVM and our MS-BPM classifier on UCI datasets. Estimates were averaged over 100 bootstrap runs for simulated training sets of small (10%) and large (50%) size. For each experiment, the best result is printed in bold. Ties with the first place, as determined by a two-sample t-test with a 97.5% confidence interval, are also printed in bold.

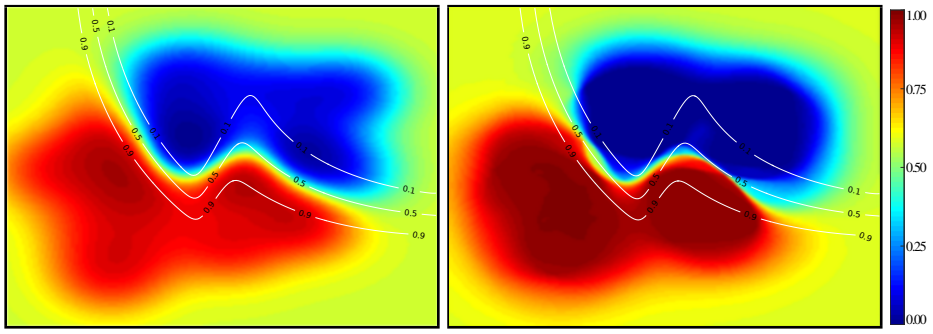
	10% Bootstrap			50% Bootstrap		
Dataset	BPM	MS-BPM	SVM	BPM	MS-BPM	SVM
Diabetes	69.2 ± 4.5	$\mathbf{71.3 \pm 3.8}$	$\mathbf{70.5 \pm 4.0}$	71.5 ± 4.0	$\mathbf{73.4 \pm 3.4}$	$\mathbf{73.5 \pm 3.5}$
Ecoli	75.2 ± 14.4	$\mathbf{80.4 \pm 3.2}$	$\mathbf{80.8 \pm 3.7}$	76.1 ± 16.7	83.8 ± 2.6	$\mathbf{85.4 \pm 2.3}$
Image	78.2 ± 22.6	90.2 ± 1.6	$\mathbf{92.1 \pm 1.5}$	93.1 ± 5.8	94.9 ± 1.2	$\mathbf{96.0 \pm 0.6}$
Ionosphere	82.2 ± 10.3	$\mathbf{87.6 \pm 4.1}$	$\mathbf{86.2 \pm 5.3}$	92.9 ± 1.7	$\mathbf{93.8 \pm 1.5}$	$\mathbf{93.5 \pm 1.7}$
Sat-Images	86.5 ± 0.7	87.4 ± 0.6	$\mathbf{88.0 \pm 0.5}$	88.8 ± 0.3	90.4 ± 0.4	$\mathbf{90.8 \pm 0.4}$
Sonar	62.2 ± 8.7	$\mathbf{68.9 \pm 4.6}$	$\mathbf{69.7 \pm 4.7}$	78.7 ± 5.2	$\mathbf{80.5 \pm 3.6}$	$\mathbf{81.3 \pm 3.4}$
Votes	83.0 ± 12.0	$\mathbf{90.2 \pm 2.3}$	$\mathbf{90.6 \pm 2.7}$	$\mathbf{92.0 \pm 6.3}$	$\mathbf{93.0 \pm 1.3}$	$\mathbf{93.4 \pm 1.4}$

5.2 Class Membership Probabilities

The class membership probabilities generated by our model often tend to better represent the true probabilities than classifiers that were calibrated subsequently after training, e.g. using Platt scaling [19]. This difference only gets amplified for small training sets, as any post-hoc calibration has to be based on a subsampling method, such as cross-validation. Figure 4 compares the membership probabilities for an SVM model and our classifier on the Ripley synthetic dataset [20]. This dataset features a two-class problem in a two-dimensional feature space. Both classes are mixtures of two Gaussians with distinct modes. This difference can be measured by comparing the log loss ($E[-\log(p(y_i|\boldsymbol{x}_i))]$) as estimated from a test sample. Our experiment gave the following results:

$$\text{LogLoss}_{SVM} = 0.15$$
$$\text{LogLoss}_{MS\text{-}BPM} = 0.08$$

Thus, our MS-BPM model improves over the SVM by approximately 53%. Most of the gains come from the improved estimation of membership probabilities in the higher-density regions of the dataset.



(a) SVM (b) MS-BPM

Fig. 4: Posterior class membership probabilities for the Ripley synthetic dataset. Both classifiers use an RBF kernel with $\gamma = 7.5$ as selected via grid search. The heatmaps show the posterior plots of the learned classification models while the overlaid contour plots depict the true probabilities. The MS-BPM classifier (right) tends to generate more confident predictions than the SVM classifier (left) and produces smoother class boundaries.

6 Conclusion

In this paper, we presented our proposed improvements to the BPM classifier. The experiments demonstrated that our MS-BPM model exhibits similar performance to SVMs and significantly improves on the original BPM, especially for small training sets. Yet it requires less hand-tuning of hyperparameters while also supporting multi-class problems natively. We also showed that the class membership probabilities generated by our model are superior to post-hoc calibrated probabilities for maximum-margin models. The algorithmic complexity of our learning algorithm ($O(N^2|C| + N|C|^2)$) also compares favorably to other kernelized vector-machine classifiers.

Acknowledgments This work was supported by the German Science Foundation (DFG) under grant OS 295/4-1.

References

1. Bordes, A., Bottou, L., Gallinari, P., Weston, J.: Solving multiclass support vector machines with larank. In: ICML. pp. 89–96. ACM (2007)
2. Cambria, E., White, B.: Jumping nlp curves: a review of natural language processing research. IEEE Computational Intelligence Magazine 9(2), 48–57 (2014)
3. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3), 27 (2011)
4. Chen, M.H., Schmeiser, B.W.: General hit-and-run monte carlo sampling for evaluating multidimensional integrals. Operations Research Letters 19(4), 161–169 (1996)

5. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
6. Gilks, W.R., Roberts, G.O., George, E.I.: Adaptive direction sampling. The statistician pp. 179–189 (1994)
7. Herbrich, R., Graepel, T., Campbell, C.: Bayes point machines. The Journal of Machine Learning Research 1, 245–279 (2001)
8. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. The annals of statistics pp. 1171–1220 (2008)
9. Homan, M.D., Gelman, A.: The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. The Journal of Machine Learning Research 15(1), 1593–1623 (2014)
10. Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X.: Applied logistic regression, vol. 398. John Wiley & Sons (2013)
11. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
12. Lovász, L., Vempala, S.: Hit-and-run from a corner. SIAM Journal on Computing 35(4), 985–1005 (2006)
13. Luettel, T., Himmelsbach, M., Wuensche, H.J.: Autonomous ground vehiclesconcepts and a path to the future. Proceedings of the IEEE 100(Special Centennial Issue), 1831–1839 (2012)
14. Minka, T.P.: Expectation propagation for approximate bayesian inference. In: UAI. pp. 362–369. Morgan Kaufmann Publishers Inc. (2001)
15. Mosimann, J.E.: On the compound multinomial distribution, the multivariate β-distribution, and correlations among proportions. Biometrika 49(1/2), 65–82 (1962)
16. Neal, R.M.: Slice sampling. Annals of statistics pp. 705–741 (2003)
17. Nguyen, T., Sanner, S.: Algorithms for direct 0–1 loss optimization in binary classification. In: ICML. pp. 1085–1093 (2013)
18. PEGWiki: Convex hull trick (2016)
19. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers 10(3), 61–74 (1999)
20. Ripley, B.D.: Pattern recognition and neural networks. Cambridge university press (2007)
21. Robert, C., Casella, G.: Monte Carlo statistical methods. Springer Science & Business Media (2013)
22. Ruján, P.: Playing billiards in version space. Neural Computation 9(1), 99–122 (1997)
23. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016)
24. Thompson, M., Neal, R.M.: Covariance-adaptive slice sampling. arXiv preprint arXiv:1003.3201 (2010)
25. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. The journal of machine learning research 1, 211–244 (2001)
26. Zhu, J., Hastie, T.: Kernel logistic regression and the import vector machine. In: NIPS. pp. 1081–1088 (2001)