Robust Bayes Point Machines

Ralf Herbrich, Thore Graepel, Colin Campbell*
Computer Science Department
Technical University of Berlin
10587 Berlin, Germany

Abstract. Support Vector Machines choose the hypothesis corresponding to the centre of the largest hypersphere that can be inscribed in version space. If version space is elongated or irregularly shaped a potentially superior approach is take into account the whole of version space. We propose to construct the *Bayes point* which is approximated by the centre of mass. Our implementation of a *Bayes Point Machine* (BPM) uses an ergodic billiard to estimate this point in the kernel space. We show that BPMs outperform hard margin Support Vector Machines (SVMs) on real world datasets. We introduce a technique that allows the BPM to construct hypotheses with non–zero training error similar to soft margin SVMs with quadratic penelisation of the margin slacks. An experimental study reveals that with decreasing penelisation of training error the improvement of BPMs over SVMs decays, a finding that is explained by geometrical considerations.

1. Introduction

Recently there has been considerable interest in the theory and application of Support Vector Machines (SVMs) [3]. SVMs construct the hypothesis using the centre of the largest inscribable hypersphere in *version space*, i.e. the space of all hypotheses consistent with the training data. Boundaries of version space tangentially contacting the hypersphere correspond to *support vectors*.

If version space is elongated a potentially superior approach is to take into account the exact geometrical structure of version space for defining the hypothesis. In this paper we will consider learning machines based on finding the midpoint of the region of intersection of all hyperplanes which divide version space into two halves of equal volume: the Bayes point. The approach can be motivated from a Bayesian perspective: if we consider a new test point $\mathbf{x}$, the set of Bayes–optimal decision functions is given by those weight vectors $\mathbf{w}$ whose posterior on a binary decision at $\mathbf{x}$ is greater than 0.5. As in general the intersection of Bayes–optimal decision functions for all $\mathbf{x}$ is empty we could approximate it by the function $\mathbf{w}_{\text{Bayes}}$ closest to the Bayes–optimal decision having knowledge of the input distribution. This hypothesis is called the Bayes

*Permanent Address: Department of Engineering Mathematics, Bristol University, Bristol BS8 1TR, UK.

point. It was shown elsewhere [4] that, assuming a spherical input distribution, $\mathbf{w}_{\text{Bayes}}$ converges to the centre of mass of version space. In this paper we will sketch an algorithmic approach for estimating the Bayes point in kernel space in Section 2. The full description of this algorithm has been presented in a technical report [1]. We extend this approach to handling admission of training errors in Section 3 and experimentally compare its performance to soft margin SVMs.

2. Estimating the Bayes Point in Kernel Space

From the theory of reproducing kernel Hilbert spaces it is known that it is possible to perform a mapping $\boldsymbol{\phi} : \mathcal{X} \mapsto \mathcal{F}$ from input space $\mathcal{X}$ to a potentially high-dimensional Hilbert space $\mathcal{F}$ called *feature space* such that a linear function f can be expressed as an inner product between the mapped point $\mathbf{x}$ and a vector $\mathbf{w} \in \mathcal{F}$ in terms of a *kernel function* $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, i.e.

$$f(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle_{\mathcal{F}} = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \qquad \mathbf{w} \in \mathcal{F}, \quad \boldsymbol{\alpha} \in \mathbb{R}^m \,. \tag{1}$$

Since a multiplication of $\mathbf{w}$ by a positive number would not change its classification we assume $\|\mathbf{w}\|_{\mathcal{F}} = 1$. Suppose we are given a training set $Z = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in (\mathcal{X} \times \{-1, +1\})^m$ then version space is defined by:

$$V(Z) = \left\{ \mathbf{w} \;\middle|\; \forall (\mathbf{x}_i, y_i) \in Z : y_i \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle_{\mathcal{F}} = y_i \sum_{j=1}^{m} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) > 0 \right\} . \tag{2}$$

We now outline an algorithm for estimating the Bayes point by the centre of mass in version space [1]. This approach develops a method presented by Pal Ruján [2]. In order to obtain the centre of mass of $V(Z)$ we uniformly generated random points (hyperplanes in input space) and average over them. Since it is difficult to generate hyperplanes consistent with Z we average over the trajectory of a ball which is placed inside $V(Z)$ and bounced like a billiard ball. The boundaries constraining the billiard are given by the hyperplanes with normal vectors $y_i \boldsymbol{\phi}(\mathbf{x}_i)$. This process converges to the centre of mass under the assumption of *ergodicity* with respect to the uniform distribution in $V(Z)$.

Based on the fact that we play billiards only in $V(Z)$ each position $\mathbf{b}$, direction vector $\mathbf{v}$ of the ball and estimate $\mathbf{w}_n$ of the centre of mass of $V(Z)$ can be expressed as linear combinations of the mapped input points, i.e. $\mathbf{w}_n = \sum_{i=1}^m \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$, $\mathbf{v} = \sum_{i=1}^m \beta_i \boldsymbol{\phi}(\mathbf{x}_i)$, and $\mathbf{b} = \sum_{i=1}^m \gamma_i \boldsymbol{\phi}(\mathbf{x}_i)$, where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are real vectors with m components and fully determine the state of the billiard. Using this notation inner products and norms in $\mathcal{F}$ become, e.g.

$$\langle \mathbf{b}, \mathbf{v} \rangle_{\mathcal{F}} = \sum_{i,j=1}^{m} \gamma_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) \qquad \|\mathbf{b}\|_{\mathcal{F}}^2 = \langle \mathbf{b}, \mathbf{b} \rangle_{\mathcal{F}} = \sum_{i,j=1}^{m} \gamma_i \gamma_j k(\mathbf{x}_i, \mathbf{x}_j) \,, \tag{3}$$

and similarly for $\mathbf{v}$ and $\mathbf{w}_n$. Before generating a billiard trajectory in version space we first run an algorithm to find an initial starting point $\mathbf{b}_0$ inside version space. Then the algorithm can be subdivided into three steps (for a detailed derivation see [1]):

1. Determine the closest boundary starting from current position $\mathbf{b}$ in direction $\mathbf{v}$. Since it is computationally very demanding to calculate the flight time of the ball *on* geodesics of the hypersphere we make use of the fact that the shortest distance in Euclidean space is also the shortest distance on the hypersphere. Thus, we have for the flight time τ_i of the ball at position $\mathbf{b}$ in direction $\mathbf{v}$ to the hyperplane with normal vector $y_i\boldsymbol{\phi}(\mathbf{x}_i)$

$$\tau_i \quad = \quad -\frac{y_i\langle \mathbf{b}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle_{\mathcal{F}}}{y_i\langle \mathbf{v}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle_{\mathcal{F}}} \stackrel{\text{def}}{=} -\frac{d_i}{\nu_i}\,. \tag{4}$$

After computing all m flight times, we look for the smallest positive, i.e.

$$c = \text{argmin}_{i:\tau_i>0}\tau_i\,.$$

If $\tau_c \to \infty$ we randomly generate a direction vector $\mathbf{v}$ pointing *towards* version space. Assuming that the last bounce took place at the hyperplane having normal $y_l\boldsymbol{\phi}(\mathbf{x}_l)$ this can easily be checked by

$$y_l\langle \mathbf{v}, \boldsymbol{\phi}(\mathbf{x}_l)\rangle_{\mathcal{F}} > 0\,. \tag{5}$$

2. Update the ball's position $\mathbf{b}' = \sum_{i=1}^m \gamma_i'\boldsymbol{\phi}(\mathbf{x}_i)$ and the new direction vector $\mathbf{v}' = \sum_{i=1}^m \beta_i'\boldsymbol{\phi}(\mathbf{x}_i)$ according to $\gamma_i' = \gamma_i + \tau_c\beta_i$ and $\beta_i' = \beta_i - 2\delta_{ic}\nu_i y_i$. To satisfy the uniqueness constraint the position $\mathbf{b}'$ and the direction vector $\mathbf{v}'$ need to be normalised. This can easily be achieved using equation (3).

3. Update the centre of mass $\mathbf{w}_n$ of the trajectory by the new line segment from $\mathbf{b}$ to $\mathbf{b}'$ calculated on the hypersphere. Since the solution $\mathbf{w}_\infty$ exists on the hypersphere we cannot simply update the centre of mass using a weighted vector addition. Having the midpoint

$$\mathbf{m} = \frac{\mathbf{b}+\mathbf{b}'}{\|\mathbf{b}+\mathbf{b}'\|_{\mathcal{F}}} = \sum_{i=1}^m \zeta_i\boldsymbol{\phi}(\mathbf{x}_i)\,, \qquad \zeta_i = \frac{\gamma_i+\gamma_i'}{\|\mathbf{b}+\mathbf{b}'\|_{\mathcal{F}}}\,,$$

we use the following update formula

$$\alpha_i' = \varrho_1\left(\mathbf{w}_n, \mathbf{m}, \frac{\Lambda_n}{\Lambda_n+\lambda_n}\right)\alpha_i + \varrho_2\left(\mathbf{w}_n, \mathbf{m}, \frac{\Lambda_n}{\Lambda_n+\lambda_n}\right)\zeta_i\,,$$

$$\begin{aligned}
\varrho_1(\mathbf{w}_n, \mathbf{m}, \mu) &= \mu\sqrt{-\frac{\mu^2-\mu^2\langle \mathbf{w}_n, \mathbf{m}\rangle_{\mathcal{F}} - 2}{\langle \mathbf{w}_n, \mathbf{m}\rangle_{\mathcal{F}} + 1}}\,,\\
\varrho_2(\mathbf{w}_n, \mathbf{m}, \mu) &= -\varrho_1(\mathbf{w}_n, \mathbf{m}, \mu)\langle \mathbf{w}_n, \mathbf{m}\rangle_{\mathcal{F}} \pm [\mu^2(1-\langle \mathbf{w}_n, \mathbf{m}\rangle_{\mathcal{F}}) - 1]\,,
\end{aligned}$$

where we choose the $\pm$ sign such that ϱ_2 is positive. Here we have used $\lambda_n = \|\mathbf{b}-\mathbf{b}'\|_{\mathcal{F}}$ for the length of the trajectory at the n–th step and $\Lambda_n = \sum_{i=1}^n \lambda_i$

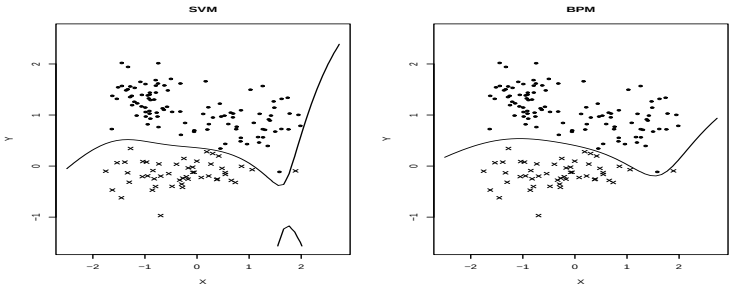

Figure 1: Decision boundaries for a 2D toy problem of a SVM (left) and BPM (right) using hard margins ($\lambda = 0$) and RBF kernels $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$ ($\sigma = 1$).

for the accumulated length of the trajectory up to the n–th step. This rather arcane definition implements a weighted addition of $\mathbf{w}_n$ and $\mathbf{m}$ such that μ is the fraction between the current and the total chord length. The weighting factor ϱ_2 of the new part of the trajectory decreases as the length of the total trajectory increases. Consequently a good stopping criterion is to terminate the algorithm when this weighting factor falls below a prespecified tolerance. Given $n_{\max}$ bounces the complexity of the algorithm is $\mathcal{O}(m^2 n_{\max})$. With BPMs, the appealing SVM feature of sparseness of the solution in the α's is lost, a problem that can be overcome by reduced rank approximations.

For illustration of the effective difference between SVM and BPM solutions consider Figure 1: the BPM trades margin for smoothness. To investigate the performance on real world datasets we compared BPMs constructed using the above algorithm against SVMs with hard margins. We studied performance on five standard benchmarking datasets from the UCI Repository[1], and `banana` and `waveform`, two toy datasets. In each case the data was randomly partitioned into 100 training and test sets in the ratio 60%:40%. The means and standard deviations of the average test set errors are presented as percentages in the columns headed SVM (hard margin) and BPM ($\lambda = 0$) in Table 1. The BPM outperforms SVMs on almost all datasets at a statistically significant level.

3. Bayes Point Machines with Soft Boundaries

To allow for training errors we will introduce the following version space conditions in place of those in (2):

$$y_i \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle_{\mathcal{F}} = y_i \sum_{j=1}^{m} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \geq -\lambda y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_i)\,, \tag{6}$$

[1] Publically accessible at http://www.ics.uci.edu/ mlearn/MLSummary.html.

	SVM (hard margin)	BPM (hard boundary)	σ	p-value
Heart	25.4±0.40	**22.8±0.34**	10.0	1.00
Thyroid	5.3±0.24	**4.4±0.21**	3.00	1.00
Diabetes	33.1±0.24	**32.0±0.25**	5.0	1.00
Waveform	13.0±0.10	**12.1±0.09**	20.0	1.00
Banana	16.2±0.15	**15.1±0.14**	0.5	1.00
Sonar	**15.4±0.37**	15.9±0.38	1.0	0.01
Ionosphere	11.9±0.25	**11.5±0.25**	1.5	0.99

Table 1: Experimental results on seven benchmark datasets. Shown are mean and standard deviation obtained on 100 different runs. The final column gives the p-values of a paired t-test indicating the improvement is statistically significant.

where $\lambda \geq 0$ is an adjustable parameter. Equation (6) can be interpreted as allowing the ball to penetrate walls. Since the decision function based on (1) is invariant under any positive rescaling of the α's it is necessary to have an α_j on the right hand side to make λ scale–invariant as well. This formulation gives a simple modification of the algorithm described in Section 2. To see this we note that equation (6) can be rewritten as

$$y_i \left[\sum_{j=1}^{m} \alpha_j (1 + \lambda \delta_{ij}) k(\mathbf{x}_j, \mathbf{x}_i) \right] \geq 0 \,, \tag{7}$$

Hence we can use the above algorithm but with an additive correction to the diagonal terms of the kernel matrix computed at the start of the algorithm $k(\mathbf{x}_i, \mathbf{x}_i) \leftarrow k(\mathbf{x}_i, \mathbf{x}_i) + \lambda$. This additive correction to the kernel diagonals is similar to the L_2 error norm used to introduce a soft margin during training of SVMs. Another insight into the introduction of soft boundaries comes from noting that the distance between two normalised points $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ can be written

$$||\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)||^2 \quad = \quad 2(1 + \lambda - k(\mathbf{x}_i, \mathbf{x}_j)) \,.$$

Thus, if we add λ to the diagonal elements of the kernel matrix, the points become equidistant for $\lambda \to \infty$. This gives the resulting version space a more isotropic shape. Hence, the centre of the largest inscribable sphere (SVM solution) tends towards the centre of mass of version space.

In order to investigate the effect of λ (soft boundaries) we trained a BPM with soft boundaries and compared it to the results of training an SVM with soft margin using the same kernel matrix (see equation (7)). In Figure 2 we plotted the generalisation error as a function of λ's for a toy problem and the `heart` dataset. We observe that the SVM with an L_2 soft margin achieves a minimum of the generalisation error which is close to, or just above, the minimum error which can be achieved using a BPM with positive λ's. This

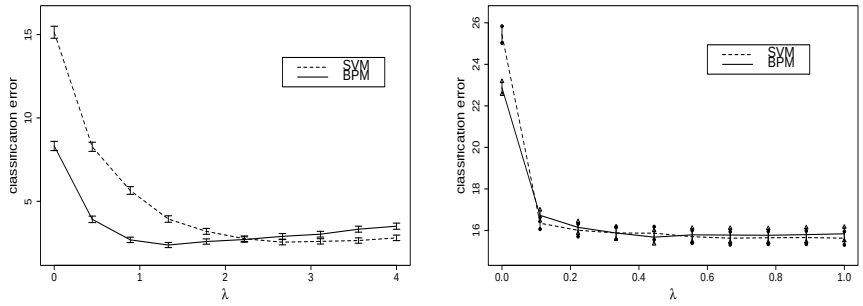

Figure 2: Generalisation error versus λ for a toy problem using linear kernels (left) and for the `heart` dataset using RBF kernels with $\sigma = 10.0$ (right). The error bars indicate one standard deviation of the estimated mean.

may not be too surprising taking the change of geometry into account. Thus soft margin SVMs approximates BPMs with soft boundaries.

4. Discussion and Conclusion

Our results indicate that hard boundary BPMs have an edge over hard margin SVMs for the type of data we investigated. We have introduced one mechanism for admitting classifiers of non–zero training error but we expect others are possible based on more refined noise models not easily implementable in the optimisation framework of SVMs. More importantly, theoretical results in the PAC–Bayesian framework (see [1]) indicate that the observed superiority of BPMs has a sound basis as measured by PAC type bounds on the generalisation error.

References

[1] R. Herbrich, T. Graepel and C. Campbell. Bayesian learning in reproducing kernel Hilbert spaces. TR 99-11, Technical University, Department of Computer Science, Berlin, 1999.

[2] P. Rujan. Playing billiard in version space. *Neural Computation*, 9:99–122, 1997.

[3] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[4] T. Watkin. Optimal Learning with a Neural Network. *Europhysics Letters* 21:871–877, 1993.