Homogenised Virtual Support Vector Machines

Christian J. Walder[1,2]
[1]Max Planck Institute for Biological Cybernetics
Spemannstaße 38, 72076 Tübingen, Germany.

Brian C. Lovell[2]
[2]IRIS Research Group, EMI
School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, Queensland 4072, Australia.
christian.walder@tuebingen.mpg.de, lovell@itee.uq.edu.au

Abstract

In many domains, reliable a priori knowledge exists that may be used to improve classifier performance. For example in handwritten digit recognition, such a priori knowledge may include classification invariance with respect to image translations and rotations. In this paper, we present a new generalisation of the Support Vector Machine (SVM) that aims to better incorporate this knowledge. The method is an extension of the Virtual SVM, and penalises an approximation of the variance of the decision function across each grouped set of "virtual examples", thus utilising the fact that these groups should ideally be assigned similar class membership probabilities. The method is shown to be an efficient approximation of the invariant SVM of Chapelle and Schölkopf, with the advantage that it can be solved by trivial modification to standard SVM optimization packages and negligible increase in computational complexity when compared with the Virtual SVM. The efficacy of the method is demonstrated on a simple problem.

1 Introduction

In recent years Vapnik's Support Vector Machine (SVM) classifier [9] has become established among the most effective classifiers on real-world problems. As remarked in a comparison of classifiers applied to handwritten digit recognition written by LeCun *et al* in 1995, SVMs are capable of achieving high generalisation performance with none of the *a priori* knowledge that is necessary in order to achieve similar results with other methods [4].

In the years since this comparison was conducted, SVM researchers have managed to incorporate problem specific prior knowledge into the SVM algorithm. One very effective approach to date, as far as performance on the MNIST handwritten digit database is concerned, has been the Virtual SVM (V-SVM) method of Scholköpf *et al* [5]. Indeed, to the best of our knowledge this method has achieved the lowest recorded test set error on the MNIST set, with trivial data preprocessing [3]. The V-SVM method involves first training a normal SVM in order to extract the support vector set. A set of transformations that are known to have no effect on the likelihood of class membership are then applied to these vectors, producing "virtual" support vectors. For example, in the case of handwritten digits, new virtual examples can be created by randomly shifting and rotating (in the 2D image sense) the original training digits. A new machine is then trained on the union of the original support vector set and the synthetic virtual support vectors.

In V-SVM, when the machine is retrained on the augmented training set, no distinction is made between the "real" and virtual data vectors. As such, a potentially useful piece of information about the problem is being discarded, namely that we know the decision latent function itself should be invariant to the transformations applied to the support vectors not just the classifier output that corresponds to the sign of this latent function. Indeed, precisely this information has been incorporated in the "Invariant SVM" (I-SVM) method of Chapelle and Scholköpf [2], which has previously been implemented using ideas from kernel PCA [7]. The present work lies in between the I-SVM and V-SVM, in that the virtual examples are included as in V-SVM but the I-SVM like invariance of the latent function are imposed only on the support vectors. As we shall see however, the present approach represents a rather efficient approximation to the I-SVM as it can be imple-

mented by trivial modification to existing SVM optimization software such as *LIBSVM* [1], with the same computational cost as the V-SVM. The remainder of the paper is structured as follows: in Section 2 we briefly review the soft margin SVM, before introducing the I-SVM in Section 3. In Section 4 the main contribution of the paper begins with the derivation of the necessary equations for the new approach. In Section 5 we demonstrate the efficacy of the new approach on a simple toy problem, showing improved performance as compared to the state of the art and rather hard to beat V-SVM.

2 Soft-Margin Support Vector Machines

In normal (squared loss) soft-margin SVM classification [9], we have a set of labelled points $\tilde{x}_i \in \mathbb{R}^d$, $i = 1 \dots N$, with associated class labels $y_i \in \{1, -1\}$. The standard SVM formulation solves the following problem:

$$\begin{array}{l} \underline{\text{Minimize}}_{\tilde{w},b} \\ \langle \tilde{w}, \tilde{w} \rangle + C \sum_{i=1}^{N} \xi_i^2 \\ \underline{\text{Subject To:}} \\ y_i \left(\langle \tilde{w}, \tilde{x}_i \rangle + b \right) \geq 1 - \xi_i, \quad i = 1 \dots N \\ \xi_i \geq 0, \quad i = 1 \dots N \end{array}$$

where C is a regularisation parameter. In the Lagrangian dual of the above problem, the data vectors appear only by way of their inner products with one another. This allows the problem to be solved in a possibly infinite dimensional feature space $\mathcal{H}$ by way of the "kernel trick", *i.e.,* the replacement of all inner products $\langle \tilde{x}_i, \tilde{x}_j \rangle$ by some Mercer kernel $k\left(\tilde{x}_i, \tilde{x}_j\right) = \phi\left(\tilde{x}_i\right).\phi\left(\tilde{x}_j\right)$, where $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ (see *e.g.* [8]). By dualising the above problem in this manner, we obtain the following final decision function:

$$g\left(\tilde{x}\right) = \text{sign}\left(f\left(\tilde{x}\right)\right), \text{where} f\left(\tilde{x}\right) = \sum_{i=1}^{N} \alpha_i^0 y_i k\left(\tilde{x}_i, \tilde{x}\right) + b$$

where the α_i^0 are obtained by maximising the dual objective function:

$$W\left(\tilde{\alpha}\right) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k\left(\tilde{x}_i, \tilde{x}_j\right)$$

subject to the constraints $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $\alpha_i \geq 0$.

3 Invariant Support Vector Machines

Previously, Chapelle and Schölkopf [6] incorporated domain specific prior knowledge into the linear SVM framework by minimising the objective function:

$$\left(1 - \gamma\right) \langle \tilde{w}, \tilde{w} \rangle + \gamma \sum_{i} \left(\langle \tilde{w}, d\tilde{x}_i \rangle \right)^2$$

subject to the usual SVM constraints (see Section 2). The tangent vectors $d\tilde{x}_i$ are directions in which *a priori* knowledge tells us that the functional value at $\tilde{x}_i$ should not change — the sense of the optimization can be seen from the equality:

$$f\left(\tilde{x} + d\tilde{x}_i\right) - f\left(\tilde{x}\right) = \langle \tilde{w}, d\tilde{x}_i \rangle$$

It turns out that the above problem is equivalent to the normal SVM after linearly transforming the input space by $\tilde{x} \rightarrow C_\gamma^{-1} \tilde{x}$ where the matrix C_γ is defined by:

$$C_\gamma \quad = \quad \left(\left(1 - \gamma\right) I + \gamma \sum_{i} d\tilde{x}_i d\tilde{x}_i \right)^{\frac{1}{2}}$$

which was shown to be equivalent to using the following kernel function in the otherwise unchanged SVM framework:

$$k_\gamma\left(\tilde{x}_i, \tilde{x}_j\right) = \tilde{x}_i^\top {C_\gamma}^{-2} \tilde{x}_j \tag{1}$$

However in order to use the kernel trick to perform this algorithm in the feature space $\mathcal{H}$ induced by $k\left(\cdot, \cdot\right)$, as in the previous Section, one must replace the term $d\tilde{x}_i d\tilde{x}_i$ with $d\Phi\left(\tilde{x}_i\right) d\Phi\left(\tilde{x}_i\right)$ in the expression for C_γ. Since $\mathcal{H}$ is typically very high dimensional, it is impossible to do this directly. A solution to this problem using kernel PCA [7] is given in [2], which takes advantage of the fact that the set $\{\phi\left(\tilde{x}_i\right)\}_{1 \leq i \leq N}$ spans a subset of $\mathcal{H}$ whose dimension is no greater than N. Unfortunately however, computing the modified kernel function in this manner does introduce computational disadvantages, and the need for a large scale version of the algorithm was noted by its authors in [2].

4 Homogenised Virtual SVM

In the I-SVM method of the previous Section, the tangent vectors $d\tilde{x}_i$ are usually not available directly, and so a finite difference approximation is used. In this case the term $\sum_i \left(\langle \tilde{w}, d\tilde{x}_i \rangle \right)^2$ can be written:

$$\sum_{i} \left(\langle \tilde{w}, \tilde{x}_i + \nabla\tilde{x}_i \rangle - \langle \tilde{w}, \tilde{x}_i \rangle \right)^2 \tag{2}$$

where the vector $\tilde{x}_i + \nabla\tilde{x}_i$ is equivalent to a "virtual" vector of the V-SVM approach that has been derived from $\tilde{x}_i$.

In the V-SVM method, the virtual vectors are derived from the real vectors by some group of "invariant" transformations, that is transformations that should not affect the likelihood of class membership. For example, in handwritten digit recognition the group of one-pixel image translations have been used to good effect [3] – in this case, for each of the original training patterns an additional 8 "virtual" vectors can be derived, one for each possible single pixel translation.

We will presently consider a combination of the I-SVM and the V-SVM, which we dub the "Homogenised Virtual SVM" (HV-SVM). To begin we combine the inclusion of the virtual examples as in the V-SVM method, with the invariance term (2), allowing as well a soft margin with parameter C as in Section 2. Altogether this leads to the objective function:

$$(1-\gamma)\left(\langle\tilde{w},\tilde{w}\rangle + C\sum_i \xi_i^2\right) + \gamma\frac{1}{2}\sum_{n=1}^{P}\sum_{\forall i,j\in\mathcal{S}_n}\left(\langle\tilde{w},\tilde{x}_i\rangle - \langle\tilde{w},\tilde{x}_j\rangle\right)^2 \qquad (3)$$

where each of the P sets $\mathcal{S}_n$ contain the subscripts of those vectors that are invariant transformations of one another. This effectively extracts a finite difference approximation of an invariant direction for each pair of vectors in the same set $\mathcal{S}_p$. The term $\frac{1}{2}$ is included for an equivalence with I-SVM, since here each invariant direction is effectively included twice. The objective function must be minimized subject to the normal SVM constraints of Section 2. It is well known by the SVM community that these constraints lead to the following optimality conditions:

$$\alpha_i\left(y_i\left(\langle\tilde{w},\tilde{x}_i\rangle + b\right) - 1 + \xi_i\right) = 0$$

which imply that for those vectors that have $\alpha_i > 0$ (the support vectors), then $\xi_i = 1 - y_i\left(\langle\tilde{w},\tilde{x}_i\rangle + b\right)$. This means that if $\tilde{x}_i$ and $\tilde{x}_j$ are support vectors belonging to the same set $\mathcal{S}_p$, then $y_i = y_j \in \{1,-1\}$ and therefore $\langle\tilde{w},\tilde{x}_i\rangle - \langle\tilde{w},\tilde{x}_j\rangle = \xi_i - \xi_j$. Assuming that all vectors are support vectors, the objective function (3) can therefore be rewritten:

$$(1-\gamma)\left(\langle\tilde{w},\tilde{w}\rangle + C\sum_i \xi_i^2\right) + \gamma\frac{1}{2}\sum_{n=1}^{P}\sum_{\forall i,j\in\mathcal{S}_n}\left(\xi_i - \xi_j\right)^2 \qquad (4)$$

In reality however, not all of the vectors will be support vectors, and thus we effectively have an approximation that becomes more ideal as the number of support vectors increases. At this point we should note that the V-SVM method usually retains only the support vectors from a preliminary training iteration before deriving and retraining with the virtual examples, and that by a similar process we can reasonably expect a high proportion of support vectors. Moreover, the approximation is roughly equivalent to minimising the invariance term of only those vectors that lie near the decision boundary (the support vectors), which also seems reasonable since those are the vectors that are widely believed to contain the most important information.

We now find the Lagrangian dual of this approximation to I-SVM. The algebra is less of a burden if we assume to begin with that all of the vectors are in the same set $\mathcal{S}_1$. Dividing the resultant objective function by $(1-\gamma)$ gives $\langle\tilde{w},\tilde{w}\rangle + C\sum_i \xi_i^2 + \frac{\gamma}{2(1-\gamma)}\sum_{i,j}\left(\xi_i - \xi_j\right)^2$ (the summations run from $i = 1\ldots N$, $j = 1\ldots N$, as shall be assumed for the remainder of this section). Note that:

$$\begin{aligned} & C\sum_i \xi_i^2 + \tfrac{\gamma}{2(1-\gamma)}\sum_{i,j}\left(\xi_i - \xi_j\right)^2 \\ = \;& C\sum_i \xi_i^2 + \tfrac{\gamma}{2(1-\gamma)}\sum_{i,j}\left(\xi_i^2 + \xi_j^2 - 2\xi_i\xi_j\right) \\ = \;& \left(C + \tfrac{\gamma N}{(1-\gamma)}\right)\sum_i \xi_i^2 - \tfrac{\gamma}{(1-\gamma)}\sum_{i,j}\xi_i\xi_j \end{aligned}$$

so that if we let $F = C + \frac{\gamma N}{(1-\gamma)}$ and $G = -\frac{\gamma}{(1-\gamma)}$ the objective function can be rewritten as:

$$\langle\tilde{w},\tilde{w}\rangle + F\sum_i \xi_i^2 + G\sum_{i,j}\xi_i\xi_j$$

The Lagrangian function with Lagrange multipliers α_i is then:

$$\begin{aligned} L\left(\tilde{w},b,\tilde{\xi},\tilde{\alpha}\right) = \;& \tfrac{1}{2}\langle\tilde{w},\tilde{w}\rangle + \tfrac{1}{2}F\sum_i \xi_i^2 + \tfrac{1}{2}G\sum_{i,j}\xi_i\xi_j \\ & - \sum_i \alpha_i\left(y_i\left(\langle\tilde{w},\tilde{x}_i\rangle + b\right) + \xi_i - 1\right) \end{aligned}$$

and the stationarity conditions are:

$$\frac{\partial L}{\partial\tilde{w}} = \tilde{0} = \tilde{w} - \sum_i \alpha_i y_i\tilde{x}_i \qquad (5)$$

$$\frac{\partial L}{\partial b} = 0 = \sum_i \alpha_i y_i \qquad (6)$$

$$\frac{\partial L}{\partial\xi_i} = 0 = F\xi_i + G\sum_j \xi_j - \alpha_i \qquad (7)$$

so $\tilde{w}$ has the usual "support vector expansion" $\tilde{w} = \sum_i \alpha_i y_i\tilde{x}_i$. Substituting (5) and (6) into the Lagrangian yields:

$$\begin{aligned} L = \;& -\tfrac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\langle\tilde{x}_i,\tilde{x}_j\rangle + \sum_i \alpha_i - \\ & \sum_i \alpha_i\xi_i + F\frac{1}{2}\sum_i \xi_i^2 + G\frac{1}{2}\sum_{i,j}\xi_i\xi_j \\ = \;& -\tfrac{1}{2}\tilde{\alpha}^\top H\tilde{\alpha} + \tilde{e}^\top\tilde{\alpha} - \tilde{\alpha}^\top\tilde{\xi} + \tfrac{1}{2}F\tilde{\xi}^\top\tilde{\xi} + \tfrac{1}{2}G\tilde{\xi}^\top E\tilde{\xi} \end{aligned}$$

where H is the usual Gram matrix, $[H_{i,j}] = y_i y_j\langle\tilde{x}_i,\tilde{x}_j\rangle$, E is a matrix of ones and I the identity matrix. Rewriting

(7) in matrix form and left-multiplying by $\tilde{\xi}^\top$ implies that $F\tilde{\xi}^\top\tilde{\xi} + G\tilde{\xi}^\top E\tilde{\xi} - \tilde{\alpha}^\top\tilde{\xi} = \tilde{0}$. The Lagrangian can therefore be written:

$$L\left(\tilde{w}, b, \tilde{\xi}, \tilde{\alpha}\right) \quad = \quad -\tfrac{1}{2}\tilde{\alpha}^\top H\tilde{\alpha} + \tilde{e}^\top\tilde{\alpha} - \tfrac{1}{2}\tilde{\alpha}^\top\tilde{\xi}$$

Finally, we substitute the expression for $\tilde{\xi}$ obtained from (7), $\tilde{\xi} = (FI + GE)^{-1}\tilde{\alpha}$, which leads to the final form of our dual objective function:

$$W'(\tilde{\alpha}) \quad = \quad -\tfrac{1}{2}\tilde{\alpha}^\top\left(H + (FI + GE)^{-1}\right)\tilde{\alpha} + \tilde{e}^\top\tilde{\alpha}$$

At this point one can show that the matrix $(FI + GE)^{-1}$ has entries equal to $\frac{F+(N-1)G}{F(F+GN)} = \frac{C(1-\gamma)+\gamma}{C(C(1-\gamma)+\gamma N)}$ on the diagonal and $\frac{-G}{F(F+GN)} = \frac{\gamma}{C(C(1-\gamma)+\gamma N)}$ elsewhere. Using these expressions we can now compactly write down the final form of the dual problem for (4). In this case one can readily verify that the sets $\mathcal{S}_n$ are treated similarly to the previous case (in which it was assumed that there is one single all encompassing set $\mathcal{S}_1 = \{1 \ldots N\}$), resulting in a dual problem that consists of minimising the dual objective function $\frac{1}{2}\tilde{\alpha}^\top(H + B)\tilde{\alpha} - \tilde{e}^\top\tilde{\alpha}$ subject to the normal SVM constraints (see Section 2). Referring to the $\mathcal{S}_n$ as groups, the matrix B is defined on the diagonal by:

$$[B]_{i,i} = \begin{cases} \frac{C(1-\gamma)+\gamma}{C(C(1-\gamma)+\gamma N_i)} & \text{(if } \tilde{x}_i \text{ is in a group)} \\ \frac{1}{C(1-\gamma)} & \text{(otherwise)} \end{cases} \tag{8}$$

where N_i is the total number of vectors in $\tilde{x}_i$'s group. Off the diagonal, B is defined by:

$$[B]_{i,j} = \begin{cases} \frac{\gamma}{C(C(1-\gamma)+\gamma N_i)} & \text{(if } \tilde{x}_i \text{ and } \tilde{x}_j \text{ are same group)} \\ 0 & \text{(otherwise)} \end{cases} \tag{9}$$

Note that if $\gamma = 0$ (or if there are no non-empty groups), then B simplifies to a diagonal matrix with entries $\frac{1}{C}$, recovering the soft-margin SVM of Section 2. The similarity with normal SVM allows the problem to be solved by trivial modification to existing SVM optimization packages such as *LIBSVM* [1] – one need simply replace the typical diagonal bias terms with the B term above.

4.0.1 Hybrid HV/I - SVM:

Note that it is also possible to derive a hybrid method that is closer to the I-SVM, in the sense that it ignores only the invariance terms of those entire sets $\mathcal{S}_n$ that contain no support vectors. If $\mathcal{V}_n$ are the support vectors, and $\overline{\mathcal{V}}_n$ the non support vectors in group $\mathcal{S}_n$, the invariance term of the HV-SVM method for that group is:

$$\gamma\tfrac{1}{2}\left(\sum_{\forall i,j\in\mathcal{V}_n}\left(\langle\tilde{w},\tilde{x}_i\rangle - \langle\tilde{w},\tilde{x}_j\rangle\right)^2 + 2\sum_{\forall i\in\mathcal{V}_n, j\in\overline{\mathcal{V}}_n}\xi_i^2\right)$$

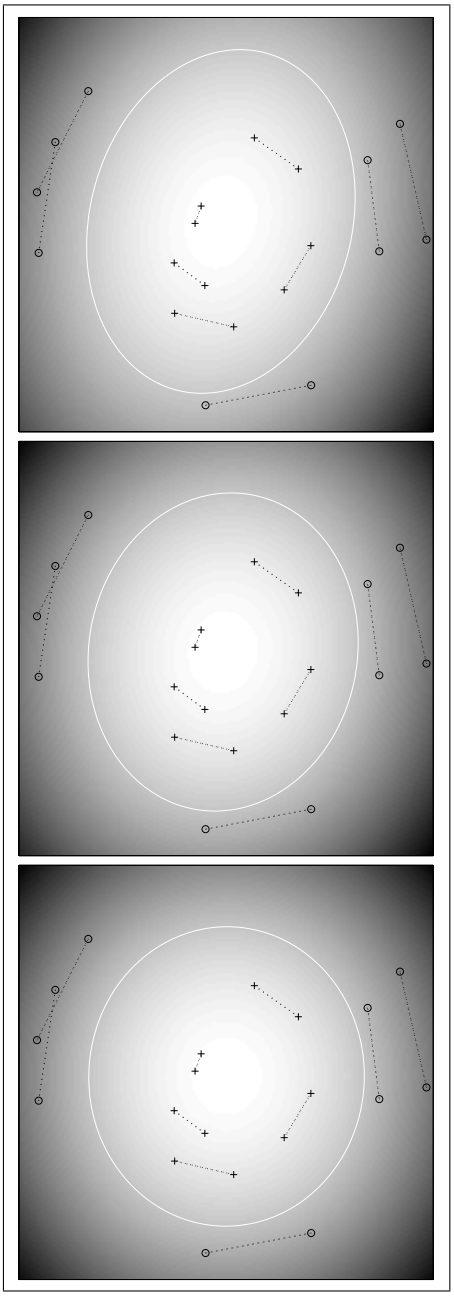

Figure 1. An illustration of the effect of the parameter γ in separating pluses from circles in 2D. An invariance to rotations is encoded by choosing virtual vectors that are rotations of the training data, each invariant group $\mathcal{S}_n$ being indicated by a dotted line. The gray level indicates the decision function value and the white line the decision boundary. ***Left*****: Virtual SVM,** ***Middle*****: HV-SVM with medium γ** ***Right*** **HV-SVM with γ close to one.**

which, assuming that we are aiming for the same result as the I-SVM, is not what is required unless $\overline{\mathcal{V}}_n$ is emtpy. However if we had known $\overline{\mathcal{V}}_n$ *a priori* and set $[B]_{i,j} = 0$ if either $i \in \overline{\mathcal{V}}_n$ or $j \in \overline{\mathcal{V}}_n$, then the second term in the above invariance term would equal zero. Thus, had we also modified the kernel function as per the I-SVM method with invariance directions $\left\{\forall\,(\tilde{x}_i - \tilde{x}_j) : \left(i \in \overline{\mathcal{V}}_n\right) \vee \left(j \in \overline{\mathcal{V}}_n\right)\right\}$ then we would have established once again the correct I-SVM invariance term for $\mathcal{S}_n$. This would need to have been done for all the $\mathcal{S}_n$, but as we do not know the $\mathcal{V}_n$ *a priori* it would be necessary to use for example approach of Algorithm 1, below, for determining which invariance directions need to be accounted for by the I-SVM kernel function, rather than by the more efficient HV-SVM bias terms $[B]_{i,j}$. In the algorithm, the invariance terms associated with non support vectors are ignored. Note that it should usually be possible to perform the retraining with fewer iterations than the initial pass, by using the previous solution as the starting point for the optimization.

Algorithm 1 HV-SVM/I-SVM Hybrid

1: $\mathcal{W} \leftarrow \emptyset$
2: $[B]_{i,j} \leftarrow 0, \forall i, j$
3: **for all** $\left\{(i,j) : \left(i \in \overline{\mathcal{W}}\right) \vee \left(j \in \overline{\mathcal{W}}\right)\right\}$ **do**
4: assign $[B]_{i,j}$ according to equations (8) and (9)
5: **end for**
6: Define k by equation (1) using the tangent vectors associated with $\mathcal{W}$
7: Optimise using k and B, and let $\mathcal{V}$ be the resultant support vector set
8: $\mathcal{R} \leftarrow \bigcup_{\forall n: \mathcal{S}_n \cap \mathcal{V} \neq \emptyset} \mathcal{S}_n$
9: **if** $\mathcal{R} \subseteq \mathcal{V} \cup \mathcal{W}$ **then**
10: finish
11: **else**
12: $\mathcal{W} \leftarrow \mathcal{W} \cup (\mathcal{R} - \mathcal{V})$
13: Go to line 2
14: **end if**

5 Experiments

Our tests on real world problems are the subject of ongoing work. In the present section we instead consider the following toy problem: the data are uniformly distributed over the interval $[-1, 1]^2$ and the true decision boundary is a circle centered at the origin, namely $f(\tilde{x}) = \text{sign}\,(||\tilde{x}|| - 1/2)$. The invariance we wish to encode is *local invariance under rotations*, and so we derive from each training vector a single virtual vector by applying a rotation of 0.5 radians about the origin. A training set of 15 points and test set of 2000 points were generated independently in 100 individual trials. For the kernel function we chose

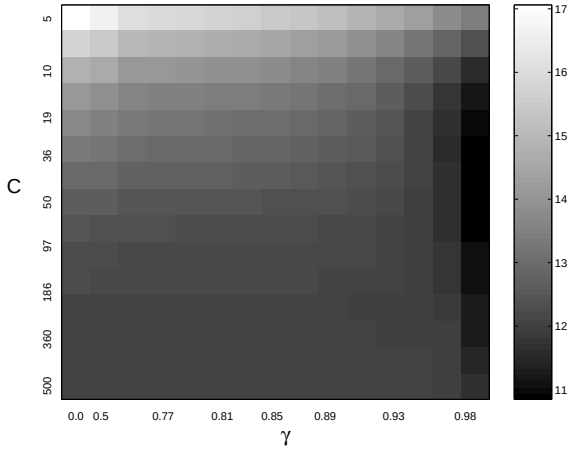

Figure 2. Cross validation performance for the circle toy problem, averaged over 100 trials. The gray-scale indicates mean test error percentage. The C scale is logarithmic, and the γ scale is logarithmic with the exception of the first value which is zero. Note that $\gamma = 0$ is equivalent to the Virtual SVM method, and that the invariances are enforced more as $\gamma \to 1$. Very little performance change is found by extending the plot to greater C values.

the second order polynomial kernel $k(\tilde{x}, \tilde{y}) = \left(\tilde{x}^\top \tilde{y} + 1\right)^2$. The results of the experiment are shown in Figure 2, which indicate that for any given value of the margin softness parameter C, the more the invariance is enforced ($\gamma \to 1$) the better the test set performance becomes. Moreover it can be seen from the example in Figure 1 that as expected, the invariance term leads to decision boundaries of a more circular nature.

6 Conclusion

We have presented a generalisation of the Virtual SVM in which the support vectors that are invariant transformations of one another are constrained to have similar decision function values. We have demonstrated that the method is superior to the state of the art Virtual SVM in a toy problem involving invariances under rotations (see Figure 2). Moreover, the algorithm is easy to implement, as the final optimisation problem is very similar to that of the standard SVM, and incurs no additional computational penalty in comparison with the Virtual SVM. We plan to perform more real-world tests – since the Virtual SVM represents the state of the art in digit recognition it seems interesting to apply the

method to this problem.

References

[1] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear support vector machines, 2001.

[3] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

[4] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition, 1995.

[5] B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In J. V. C. von der Malsburg, W. von Seelen and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, volume 1112, pages 47–52, Berlin, 1996. Springer Lecture Notes in Computer Science.

[6] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 640–646. MIT Press, 1998.

[7] B. Schölkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem, 1998.

[8] B. Scholköpf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[9] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.