in: C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, & B. Sendhoff (eds.): Artificial Neural Networks --- ICANN'96. Springer Lecture Notes in Computer Science, Vol. 1112, Berlin, 1996, 47 --52

Incorporating Invariances in Support Vector Learning Machines

Bernhard Schölkopf[1 2], Chris Burges[2], and Vladimir Vapnik[2]

[1] Max–Planck–Institut für biologische Kybernetik, Tübingen, Germany, E–mail bs@mpik-tueb.mpg.de
[2] AT&T Bell Laboratories, Holmdel, NJ, USA

Abstract. Developed only recently, support vector learning machines achieve high generalization ability by minimizing a bound on the expected test error; however, so far there existed no way of adding knowledge about invariances of a classification problem at hand. We present a method of incorporating prior knowledge about transformation invariances by applying transformations to support vectors, the training examples most critical for determining the classification boundary.

1 Incorporating Invariances

In many applications of learning procedures, prior knowledge about properties of the function to be learned is available (for a review, see Abu–Mostafa, 1995). For instance, certain transformations of the input could be known to leave function values unchanged. Mostly, two different ways of exploiting this knowledge have been used: either the knowledge is directly incorporated in the algorithm, or it is used to generate artificial training examples ("virtual examples") by transforming the training examples accordingly.

In the first case, an additional term in an error function can force a learning machine to construct a function with the desired invariances (Simard et al., 1992); alternatively the invariance can be achieved by using an appropriate distance measure in the pattern space (Simard, Le Cun, and Denker, 1993). The latter is akin to changing the representation of the data by first mapping them into a more suitable space; an approach pursued for instance by Segman, Rubinstein, & Zeevi (1992), or Vetter & Poggio (1996).

In the second case, it is hoped that given sufficient time, the learning machine will extract the invariances from the artificially enlarged training data. Figure 1 contains illustrations of the different approaches.

Simard et al. (1992) compare the two techniques and find that for the considered problem — learning a function with three plateaus where the function values are locally invariant — training on the artificially enlarged data set is significantly slower, due to both correlations in the artificial data and the increase in training set size. Moving to real–world applications, the latter factor becomes even more important. If the size of a training set is multiplied by a number of desired invariances (by generating a corresponding number of artificial examples for each training pattern), the resulting training set can get rather large (as the ones used by Drucker, Schapire, and Simard, 1993). On the other hand, the method of generating virtual examples has the advantage of being readily implemented for all kinds of learning machines and symmetries. If instead of Lie groups of symmetry transformations one is dealing with discrete symmetries, such as the bilateral symmetries of Vetter, Poggio, & Bülthoff (1994), derivative–based methods such as the ones of Simard et al. (1992) are not applicable. In conclusion, it would be desirable to have an intermediate method which has the advantages of the virtual examples approach without its computational cost. In this paper, we will try to convince the reader that such a method can be realized if we build on a type of learning machine to be described in the following section.

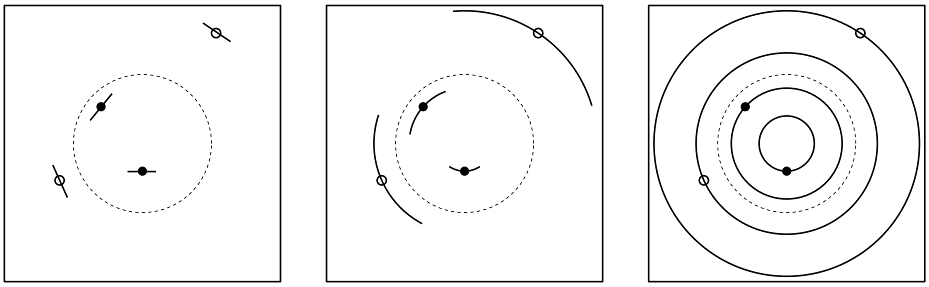

Fig. 1. Different ways of incorporating invariances in a decision function. The dotted line marks the "true" boundary, disks and circle are the training examples. We assume that prior information tells us that the classification function only depends on the norm of the input vector (the origin being in the center of each picture). Solid lines crossing an example indicate the type of information conveyed by the different methods of incorporating prior information. Left: incorporating a regularizer to learn tangent values (cf. Simard et al., 1992); middle: generating virtual examples in a localized region around each training example, right: changing the representation of the data by first mapping each example to its norm. If feasible, the latter method yields the most information. However, if the necessary nonlinear transformation cannot be found, or if the desired invariances are of localized nature, one has to resort to one of the former techniques. Finally, the reader may note from the right hand side picture that in particular, examples close to the boundary allow us to exploit prior knowledge very effectively: given a method to get a first approximation of the true boundary, the examples closest to it would allow good estimation of the true boundary. A similar two–step approach shall be pursued in this paper.

2 Support Vector Learning Machines

The support vector algorithm (Vapnik, 1995, Boser, Guyon & Vapnik, 1992, Cortes & Vapnik, 1995) uses the Structural Risk Minimization principle (Vapnik, 1979) to construct decision rules that generalize well. In doing so, they extract a small subset of the training data. Space does not permit to explain this algorithm in detail; we thus will merely outline its main ideas. The Method of Structural Risk Minimization is based on the fact that the test error rate is bounded by the sum of the training error rate and a term which depends on the so–called VC(Vapnik–Chervonenkis)–dimension of the learning machine. By minimizing the sum of both quantities, high generalization performance can be achieved. For linear hyperplane decision functions

$$f(\mathbf{x}) = \mathrm{sgn}\left((\mathbf{w} \cdot \mathbf{x}) + b\right), \tag{1}$$

the VC–dimension can be controlled by controlling the norm of the weight vector $\mathbf{w}$ (Vapnik, 1995). Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$, $\mathbf{x}_i \in \mathbf{R}^N, y_i \in \{\pm 1\}$, a separating hyperplane which generalizes well can be found by minimizing (Cortes & Vapnik, 1995)

$$||\mathbf{w}||^2 + \gamma \cdot \sum_{i=1}^{\ell} \xi_i \tag{2}$$

subject to

$$\xi_i \geq 0, \quad y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) \geq +(1 - \xi_i) \quad \text{for } i = 1, \ldots, \ell \tag{3}$$

(γ is a constant which determines the trade–off between training error and VC–dimension). The solution of this problem can be shown to have an expansion

$$\mathbf{w} = \sum_{i=1}^{\ell} \lambda_i \mathbf{x}_i, \tag{4}$$

where only those λ_i are nonzero which belong to an $\mathbf{x}_i$ precisely meeting the constraint (3) — these $\mathbf{x}_i$ lie closest to the decision boundary, they are called *Support Vectors*. The λ_i are found by solving the quadratic programming problem defined by (2) and (3).

Finally, this method can be generalized to nonlinear decision surfaces by first mapping the input nonlinearly into some high–dimensional space, and finding the separating hyperplane in that space (Boser, Guyon & Vapnik, 1992). This is achieved implicitely by using different types of symmetric functions $K(\mathbf{x}, \mathbf{y})$ instead of the ordinary scalar product $(\mathbf{x} \cdot \mathbf{y})$. This way one gets as a generalization of (1) and (4)

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{\ell} \lambda_i \cdot K(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{5}$$

3 Virtual Support Vectors

The choice of K determines the type of classifier that is constructed; possible choices include polynomial classifiers ($K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^n$), neural networks ($K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \cdot (\mathbf{x} \cdot \mathbf{x}_i) - \Theta)$) and radial basis function classifiers ($K(\mathbf{x}, \mathbf{y}) = \exp\left(-||\mathbf{x} - \mathbf{y}||^2/\sigma^2\right)$). Already without the use of prior knowledge, these machines exhibit high generalization ability;[3] moreover, they extract almost identical sets of support vectors (cf. Eq. 4). It is possible to train any one of

[3] In a performance comparison, Bottou et al. (1994) note that the support vector machine "has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers, it does not include knowledge about the geometry of the problem."

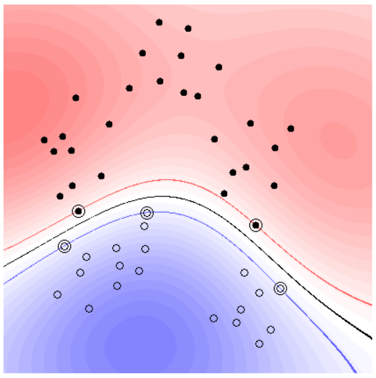

Fig. 2. Example of a support vector classifier found by using a radial basis function kernel $K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2)$. Both coordinate axes range from -1 to +1. Circles and disks are two classes of training examples; the middle line is the decision surface; the outer lines precisely meet the constraint (3). Note that the support vectors found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task.

these machines solely on the support vector set extracted by another machine, with a test performance not worse than after training on the full data base (Schölkopf, Burges, and Vapnik, 1995). Using this finding as a starting point, we investigated the question whether it might be sufficient to generate virtual examples from the support vectors only. After all, one might hope that it does not add much information to generate virtual examples of patterns which are not close to the boundary. In our experiments, we proceeded as follows:

1. Train a support vector machine to extract the support vector set.
2. Generate artificial examples by applying the desired invariance transformations to the support vectors. In the following, we will refer to these examples as *virtual support vectors*.
3. Train another support vector machine on the generated examples.

The first set of experiments was conducted on a US postal service data base of handwritten digits, containing 7291 training examples and 2007 test examples. This data base has been used extensively in the literature, with a LeNet1 Convolutional Network achieving a test error rate of 5.0% (Le Cun et al., 1989). In the experiments, we used $\gamma = 10$ (cf. (2)), and a smoothing of the data with a Gaussian kernel of width 0.75.

To get a ten–class classifier for digit recognition, we combine ten binary classifiers of the described type (cf. Vapnik, 1995). In this case, virtual support vectors are generated for the set of all different support vectors of the ten classifiers. Alternatively, one can carry out the procedure separately for the ten binary classifiers, thus dealing with smaller training sets during the training of the second machine. Table 1 shows that incorporating only translational invariance already improves performance significantly, from 4.0% to 3.2% error rate.[4]

[4] It should be noted that the used test set is rather difficult — the human error rate is 2.5% (for a discussion, see Simard, Le Cun, and Denker, 1993).

Table 1. Comparison of support vector sets and performance for training on the original data base and training on the generated virtual support vectors. In both training runs, we used a polynomial classifier of degree 3. Virtual support vectors were generated by simply shifting the images by one pixel in the four principal directions. Adding the unchanged support vectors, this leads to a training set of the second classifier which has five times the size of the first classifier's overall support vector set (i.e. the union of the 10 support vector sets of the binary classifiers — note that due to some overlap, this is smaller than the sum of the ten support set sizes).

classifier trained on	size	average no. of SVs	no. of different SVs	test error
full training set	7291	274	1677	4.0%
overall SV set	1677	268	1582	4.1%
virtual SV set	8385	686	4535	3.2%

For other types of invariances, we also found improvements, albeit smaller ones: generating virtual support vectors by rotation or by the line thickness transformation of Drucker, Schapire, and Simard (1993), we constructed polynomial classifiers with 3.7% error rate (in both cases).

The larger a database, the more information about invariances of the decision function is already contained in the differences between patterns of the same class. To show that it is nevertheless possible to improve classification accuracies with our technique, we applied the method to the MNIST database of 60000+10000 handwritten digits. This database has become the standard for performance comparisons in our department; the error rate record of 0.7% is held by a boosted LeNet4 (Bottou et al. 1994). Using virtual support vectors generated by 1–pixel translations, we improved a degree 5 polynomial classifier from 1.4% to 1.0% error rate. In this case, we applied our technique separately for all ten support vector sets of the binary classifiers (rather than for their union) in order to avoid getting overly large support vector sets after retraining.

Further improvements can possibly be achieved by combining different types of invariances, and choosing the kind of transformations applied to individual support vectors according to whether they actually do provide new information about the decision boundary (c.f. (3)). Another intriguing extension of the scheme would be to use techniques based on image correspondence (e.g. Vetter & Poggio, 1996) to extract transformations from the training examples. Those transformations can then be used to generate virtual support vectors.

4 Discussion

We have shown that for support vector learning machines, invariances can readily be incorporated by generating virtual examples from the support vectors, rather than from the whole training set. The method yields a significant gain in classification accuracy at a moderate cost in time: it requires two training runs (rather than one), and it constructs classification rules utilizing more support vectors, thus slowing down classification speed (cf. Eq. 5) — in our case, both points amounted to a factor of about 2.

Given that support vector machines are known to allow for short training times (Bottou et al., 1994), the first point is usually not critical. Certainly, training on virtual examples generated from the whole data base would be significantly slower. To compensate for the second point, one could use the reduced set method of Burges (1996) to increase the speed.

Acknowledgements The ideas presented here were influenced by discussions with F. Girosi, P. Niyogi, T. Poggio, and K. Sung during a stay of B. S. at the Massachusetts Institute of Technology.

References

Abu–Mostafa, Y. S.: Hints. Neural Computation **7** (1995) 639–671

Boser, B. E., Guyon, I. M., Vapnik, V.: A training algorithm for optimal margin classifiers. Fifth Annual Workshop on Computational Learning Theory, Pittsburgh ACM (1992) 144–152.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., Le Cun, Y., Müller, U. A., Säckinger, E., Simard, P., Vapnik, V.: Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem (1994)

Burges, C.: Simplified support vector decision rules. 13th International Conference on Machine Learning (1996)

Cortes, C., Vapnik, V.: Support Vector Networks. Machine Learning **20** (1995) 1–25

Drucker, H., Schapire, R., Simard, P.: Boosting performance in neural networks. International Journal of Pattern Recognition and Artificial Intelligence **7** (1993) 705–719

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. J.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1** (1989) 541–551

Schölkopf, B., Burges, C., Vapnik, V.: Extracting support data for a given task. In: Fayyad, U. M., Uthurusamy, R. (eds.): Proceedings, First International Conference on Knowledge Discovery & Data Mining, AAAI Press, Menlo Park, CA (1995)

Segman, J., Rubinstein, J., Zeevi, Y. Y.: The canonical coordinates method for pattern deformation: theoretical and computational considerations. IEEE Transactions on Pattern Analysis and Machine Intelligence **14** (1992) 1171–1183

Simard, P., Le Cun, Y., Denker, J.: Efficient pattern recognition using a new transformation distance. In: Hanson, S. J. , Cowan, J. D., Giles, C. L. (eds.): Advances in Neural Information Processing Systems 5, Morgan Kaufmann, San Mateo, CA (1993)

Simard, P., Victorri, B., Le Cun, Y., Denker, J.: Tangent Prop — a formalism for specifying selected invariances in an adaptive network. In: Moody, J. E., Hanson, S. J., Lippmann, R. P.: Advances in Neural Information Processing Systems 4, Morgan Kaufmann, San Mateo, CA (1992)

Vapnik, V.: Estimation of Dependences Based on Empirical Data. [in Russian] Nauka, Moscow (1979); English translation: Springer Verlag, New York (1982)

Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag, New York (1995)

Vetter, T., Poggio, T.: Image Synthesis from a Single Example Image. Proceedings of the European Conference on Computer Vision, Cambridge UK, in press (1996)

Vetter, T., Poggio, T., Bülthoff, H.: The importance of symmetry and virtual views in three–dimensional object recognition. Current Biology **4** (1994) 18–23