Invariances in Classification: an efficient SVM implementation

Gaëlle Loosli[1], Stéphane Canu[1], S.V.N. Vishwanathan[2], and Alex J. Smola[2]

[1] Laboratoire Perception, Systèmes, Information - FRE CNRS 2645
B.P. 08 - Place Emile Blondel
76131 - Mont Saint Aignan Cedex - France
(e-mail: gloosli@insa-rouen.fr, scanu@insa-rouen.fr)

[2] National ICT for Australia.
Canberra, ACT 0200 - Australia
(e-mail: vishy@axiom.anu.edu.au, alex.smola@anu.edu.au)

Abstract. Often, in pattern recognition, complementary knowledge is available. This could be useful to improve the performance of the recognition system. Part of this knowledge regards invariances, in particular when treating images or voice data. Many approaches have been proposed to incorporate invariances in pattern recognition systems. Some of these approaches require a pre-processing phase, others integrate the invariances in the algorithms. We present a unifying formulation of the problem of incorporating invariances into a pattern recognition classifier and we extend the SimpleSVM algorithm [Vishwanathan *et al.*, 2003] to handle invariances efficiently.

Keywords: SVM, Invariances, Classification, Active Constraints.

1 Introduction

The problem of invariances has been widely studied from a signal processing point of view for pattern recognition (see [Wood, 1996] for a review). Proposed methods in this area mainly consists in invariant features extraction before feature classification. To do so, Fourier transforms and similar transforms are used, as well as moment methods like Zernike moments [Wood, 1996]. In 1993 the invariances where taken into account in neural networks [Simard *et al.*, 1993] with the idea to modify the metric distance and use one that allows the variations of a pattern to be *close* the one from the others. In 1996 the invariances appeared for SVMs. In [Schölkopf *et al.*, 1996] the authors propose to generate some virtual examples to enlarge the dataset and thus make the algorithm learn invariances. Our approach is to provide a unifying framework for invariances in Support Vector Machines. First we define a general view of invariances in pattern recognition and show how to incorporate it in SVMs. The next part shows the connexion between our method and the existing ones. Finally we give details on *Invariant SimpleSVM* algorithm and some results obtained on the USPS database.

2 Invariant SimpleSVM

In this section we will propose a general formalisation of invariances in pattern recognition.

Definition. We need to define what a transformation is and how to apply it to any kind of patterns. A pattern x belongs to a level space $\mathcal{L}$ (for instance the contrast) and relies on its support space $\mathcal{S}$ (for instance the background). A pattern transformation is an application that maps a pattern and some parameters to a *transformed* pattern:

$$\begin{aligned} T : \mathcal{L} \times \Theta &\to \mathcal{L} \\ x, \theta &\mapsto T(x, \theta) \end{aligned}$$

Moreover we require $T(x, 0) = x$.

If we now consider the binary classification purpose, we define the decision function $D(x)$ as a mapping from $\mathcal{X}^d$ to $\{0, 1\}$ that maps x to $D(x)$. If we want the decision function to be invariant with respect to the rotation, we will require that it gives the same decision for an image and its rotations:

$$\begin{aligned} &D(\{I(\ell_i, L_i), i = [1, d]\}) = D(\{I(R_\theta(\ell_i, L_i)), i = [1, d], \forall\theta\}) \\ &D(x_0) = D(x_\theta) \end{aligned}$$

2.1 Formulation

Integrating invariances into SVMs requires us to distinguish between separable (with no error) and non separable (with errors) cases. The first case is quite easily solvable while the second requires some non trivial constraints to be satisfied.

A kernel $k(x, y)$ is a positive and symmetric function of two variables (for more details see [Atteia and Gaches, 1999]) lying in a Reproducing Kernel Hilbert Space with the scalar product:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{k} \sum_{j=1}^{l} f_i g_j k(\mathbf{x}_i, \mathbf{x}'_j).$$

Hard-margins. In the separable case we can formulate the SVM problem with invariances as follows:

$$\begin{cases} \quad \min\limits_{f,b} \frac{1}{2} \|f\|^2_{\mathcal{H}} \\ \text{s.t.} \quad y_i(f(T(\mathbf{x}_i, \theta)) + b) \geq 1 \; i \in [1, m], \theta \in \Theta \end{cases} \tag{1}$$

where b is a scalar called bias. From this we can deduce the dual formulation (Wolfe's dual):

$$\begin{cases} \max\limits_{\boldsymbol{\alpha}} -\dfrac{1}{2}\sum\limits_{i,j=1}^{m}\displaystyle\int\int_{\Theta}\alpha_i(\theta_1)\alpha_j(\theta_2)y_iy_jk(T(\mathbf{x}_i,\theta_1),T(\mathbf{x}_j,\theta_2))d\theta + \sum\limits_{i=1}^{m}\int_{\Theta}\alpha_i(\theta)d\theta \\ \text{s.t. } \sum\limits_{i=1}^{m}\displaystyle\int_{\Theta}\alpha_i(\theta)y_id\theta = 0 \\ \text{and } \alpha_i(\theta) \geq 0 \qquad i \in [1,m], \theta \in \Theta \end{cases} \tag{2}$$

The Lagrange multipliers will be 0 for all points except the support vectors. Because of the nature of the hypothesis space, it is reasonable to assume that $\alpha_i(\theta)$ will have non-zero values only for a few finite number of parameters θ, thus we can simplify the writing:

$$\begin{cases} \max\limits_{\boldsymbol{\alpha}} -\dfrac{1}{2}\sum\limits_{i,j,\theta_1,\theta_2}\alpha_i(\theta_1)\alpha_j(\theta_2)y_iy_jk(T(\mathbf{x}_i,\theta_1),T(\mathbf{x}_j,\theta_2)) + \sum\limits_{i,\theta}\alpha_i(\theta) \\ \text{s.t. } \sum\limits_{i,\theta}\alpha_i(\theta)y_i = 0 \\ \text{and } \alpha_i(\theta) \geq 0 \qquad i \in [1,m], \theta \in \Theta \end{cases} \tag{3}$$

$$\begin{cases} \max\limits_{\boldsymbol{\gamma}\in\mathbb{R}^{m\times p}} & -\frac{1}{2}\boldsymbol{\gamma}^\top G\boldsymbol{\gamma} + \mathbf{e}^\top\boldsymbol{\gamma} \\ \text{s.t.} & \boldsymbol{\gamma}^\top\mathbf{y} = 0 \\ \text{and} & \gamma_i \geq 0 \qquad i \in [1,mp] \end{cases} \tag{4}$$

where $\boldsymbol{\gamma} = [\boldsymbol{\alpha}_1(\theta); \boldsymbol{\alpha}_2(\theta); \ldots; \boldsymbol{\alpha}_m(\theta)]$, G is the block matrix defined as $G_{IJ} = y_iy_jK^{ij}$ with $K^{ij}_{kl} = k(T(x_i,\theta_k),T(x_j,\theta_l))$ and p is the size of Θ.

Soft-margin. Considering the case of soft-margin, i.e. the non separable case, we face another problem. Adding a slack variable to allow errors in the solution makes the last condition of system 2 $\alpha_i(\theta) \geq 0$ become $0 \leq \int_{\Theta}\alpha_i(\theta)d\theta \leq C$ where C is a trade-off acting on the regularity of the decision function. This is quite similar to the hard-margin case. However we cannot make the assumption that we will end up with a finite number of non-zero valued Lagrange multipliers. Indeed if the trajectory of a transformation goes through the margin, then we have an infinite number of Lagrange multipliers bounded so that $\int_{\Theta}\alpha_i(\theta)d\theta = C$.

3 A Unifying Formulation

We can roughly identify three main streams in the algorithms dealing with invariances and support vector methods. Some are based on an artificial enlargement of the dataset, others rely on the modification of the cost function to incorporate the invariances (and thus using a different metric) and there are also methods using polynomial approximations to represent trajectories.

3.1 Enlarging artificially the dataset

A very intuitive way to learn invariances is to incorporate them in the training set. This operation can be processed before the use of the learning algorithm by artificially generating transformations of the sample data. The enlarged dataset (actual samples and virtual samples) contains thus the prior knowledge. Despite its simplicity, one major drawback of this method is the size of the resulting problem.

Virtual-SV. This idea was applied in SVM in [Schölkopf *et al.*, 1996] with the V-SV. The authors propose a way to reduce the size problem. Knowing that all the information needed for the classification task in SVM is contained in the support vectors, the authors make the assumption that one do not need to take into account the non-support vector's variations, since they are supposed to be far from the frontier between the classes. So basically the idea is to run a first time a classical SVM to retrieve the support vectors. Then the virtual vectors are generated from these support vectors and another SVM is run on the enlarged database. Experiments show that applying transformations on support vectors only gives at least as good results as enlarging the complete dataset.

Invariant SimpleSVM can achieve the same task if the transformation $T(x, \theta)$ is approximated by a finite number of point by fixing a finite number of values for θ.

3.2 Adapting the distance metric to invariances

Introduced in 1993 in [Simard *et al.*, 1993] and referred as the tangent distance, the motivation was to find a better distance measure than the Euclidean distance for the purpose of invariances treatment. In the field of support vector algorithm this idea has been used in various methods and we briefly describe some of them in the following parts.

Invariant SVM Let's now introduce the tangent vectors. The idea is to associate each training vector with one or several tangent vectors and to incorporate the invariances in the cost function [Chapelle and Schölkopf, 2002]. Optimising this cost function turns out to be equivalent to run a classic SVM with pre-processed data with a particular linear mapping. In the non-linear case, the results are similar except one would train a linear SVM on pre-processed data with a non-linear mapping.

Tangent Distance and Tangent Vector Kernels. Following the idea of taking a tangent measure (TD-measure), kernels embedding the invariances has been proposed. In [Pozdnoukhov and Bengio, 2003] the authors propose kernel functions between trajectories rather than between points. They define a function that measures the proximity between a point and the transformation trajectory of another point.

Invariant SimpleSVM is similar to these methods if the transformation is approximated by a first order polynomial ($T(x, \theta) \simeq x + \nabla_\theta T(x, 0)\theta$).

3.3 Polynomial Approximation

Semi-Definite Programming Machines.

Presenting the SDPM [Graepel and Herbrich, 2004], the authors are trying to learn data that are trajectories instead of the usual points. The aim is then to separate trajectories that represent the (differentiable) transformations of the original training points.

They show that this problem is solvable for transformations that can be represented or approximated by polynomials. Basing their approach on Nesterov's theorem they formulate the problem of learning a maximum margin classifier with an SDP formulation under polynomial constraints. Using the SD-representability of non negative polynomials they replace the usual non-negativity constraints in SVM by positive semi-definite constraints. Doing so they show that it is possible to learn to classify trajectories. However this approach is rather intractable since it requires to solve large SDP.

Invariant SimpleSVM also contains this approach if the transformation is approximated by a second order polynomial ($T(x,\theta) \simeq x + \nabla_\theta T(x,\theta)\theta + \frac{1}{2}\theta^\top H_\theta \theta$).

In the separable case, we can solve directly the problem and our solution is thus more tractable than the SDPM. Nevertheless we are penalised in the non separable case since we need to discretise the parameter space. Despite the complexity of SDPM, it always works in the original space Θ.

4 Algorithm and applications

We present in this section the SimpleSVM algorithm. We extend this method for invariances because of its structure. Briefly, SimpleSVM adds points to the solution one by one and this let us incorporate some treatment to each point separately. This strong property breaks down the computing time that would occur if one would apply the equivalent treatment to the whole database. The main idea is to transform points only when adding them to the solution, which excludes from this treatment all the points that are far from the frontier between classes.

4.1 SimpleSVM

The SimpleSVM algorithm [Vishwanathan *et al.*, 2003] is based on the decomposition of the database into three groups (the *working* set, the *inactive* set and the *bounded* set). Assuming the groups are known, it solves the SVM optimisation problem on the working set only. Having a solution, it checks whether the group repartition is relevant. If not the groups are updated (by adding a violator point in the working set) and it iterates over these two steps (see algorithm 1). A detailed explanation can be found in [Loosli *et al.*, 2004].

Algorithm 3 : *simpleSVM*

```
1. $(I_s, I_0, I_c) \leftarrow$ initialise
while minimumReached=FALSE
    2. $(\alpha,\lambda) \leftarrow$ solve the system without constraints($I_s$)
    if $\exists \alpha_i \leq 0$ or $\exists \alpha_i \geq C$
        3.1 project $\alpha$ inside the admissible set
        3.2 transfers the associated point from $I_s$ to $I_0$ or $I_c$
    else
        4. look for the best candidate $x_{cand}$ in $I_c$ and $I_0$
        if $x_{cand}$ is found
            5. transfer $x_{cand}$ to $I_s$
        else
            6. minimumReached $\leftarrow$ TRUE
        end if
    end if
end while
```

The Matlab implementation of this algorithm as well as the *invariant SimpleSVM* are available at:
`http://asi.insa-rouen.fr/~gloosli/simpleSVM.html` [Loosli, 2004].

4.2 Invariant SimpleSVM

Invariant SimpleSVM integrates invariances like virtual vectors, first order polynomials and so on. In the implementation we present here we chose to deal with virtual vectors. The idea is to add virtual vectors that are derived from potential support vectors only. Doing so we can achieve the same task as V-SV in only one run of the algorithm. Compared to SimpleSVM, only the step 4 in algorithm 1 is modified. While in SimpleSVM the best candidate is the point that violates the most the constraints in the dual space (or is the worst classified in the primal space), for *Invariant SimpleSVM* the best candidate is chosen among the transformations of one vector. Here we can come up with several heuristics, depending on how the transformations are represented. Let's take the case we choose to discretize the space of parameters Θ:

- complete search: each step considers only one point and all its transformations and searches whether one violates the constraints,
- incomplete search: each step considers a group of points and searches whether one violates the constraints. If so, it also considers all the transformations and looks if one is worse than the original point,
- random search: can be applied to both of the previous heuristics. Instead of taking all the transformations, pick randomly one or several transformations. This way is faster but does not necessarily reach the best solution.

4.3 Application to character recognition

It is known that incorporating invariances improves the results of a recognition task. In our experiments we first to get an idea of the efficiency of the method, in other words to monitor the actual improvements. We present the results for the complete USPS database in order to compare our method to the published results.

Experiments settings. All the results here are obtained on the USPS database. This database contains 7291 training pictures 16×16 pixels, valued in $[-1, 1]$ and 2007 test pictures. Pictures represent digits from 0 to 9 collected from handwritten postcodes. This dataset is widely used to benchmark recognition methods and is known as a difficult set. Indeed the human performance is 2.5% of error.

The nature of the data induces the choice of the transformations to apply. A digit means the same regardless of translations, small rotations, line thickness for instance. Hence the transformation used for experiments are vertical and horizontal translation, rotation with angle $10°$ clockwise and anti-clockwise, line thinning and thickening. These transformations are computed on-the-fly for any point point that is about to reach the working set. The main advantage of this choice is that we do not need to store all the transformations of all the points. However it increases the training time.

The experiments on the USPS database were done with several objectives. The first one was to show our algorithm was efficient and fast. The second one was to explore different combinations of transformations (for instance published results with SVM methods are applied with only the translation of one pixel). The results are shown in table 1. The parameters are obtained from a cross-validation. In table 2 we give the main published results on USPS.

Kernel	bandwidth	C	Transformation	Error	Time (train and test)
Poly 5	0.1	10^-5	none	4.09	235 sec
Poly 5	0.1	10^-4	tr	3.44	1800 sec
Poly 5	0.1	10^-4	tr+rot	3.19	3200 sec
Poly 5	0.1	10^-4	tr+er	3.14	-
Poly 5	0.1	10^-4	tr+er+dil	3.24	2400 sec
Poly 5	*0.1*	10^-4	*tr+rot+er*	*2.99*	*2300 sec*
Poly 5	0.1	10^-4	tr+rot+er+dil	3.24	4800 sec

Table 1. results on USPS: Here we present results obtained with *Invariant SimpleSVM.* The applied transformations are translation of 1 pixel (*tr*), rotation (*rot*), erosion and dilatation (respectively *er* et *dil*). The best result is obtained in less than 40 minutes. Note that the computation time depends on the number of support vectors, thus adding a transformation may improve computing time if it generates good support vectors that are eliminating many candidates (for instance this happens between *tr + rot* and *tr + rot + er*, where erosion clearly gives good support vectors and the algorithm converges faster).

Method	Error
Tangant Vector and Local Rep. [Keysers *et al.*, 2002]	2.0 %
Virtual SVM [Schölkopf *et al.*, 1996]	3.2 %
Invariance Hyperplane + V-SV	3.0 %
Invariant SimpleSVM (this paper)	3.0 %
Human performance	2.5 %

Table 2. Some published results on USPS

4.4 Discussion

We show here that *Invariant SimpleSVM* is efficient. Let's note that we have implemented the transformations with the discrete point of view, which is equivalent to the V-SV method. However we achieve a better performance on USPS. This can be explained by the fact we method is more flexible concerning the points which generates virtual vectors. Indeed we consider the transformations of each point that could be support vector, but not necessarily is support vector in the end. That way we consider more transformations. Taking into account the invariances considerably increases the training time (from less than 6 min without transformations up to 1 hour if we consider all the listed transformations) but it is still very fast compared to other methods. As for the effect of the different transformations, it is hard to conclude. We noticed that the translation is the most influential one. The others have small effects and the differences between different combinations are not significant enough.

5 Conclusion

Based on the unifying approach for invariances with SVM proposed, an efficient implementation for the virtual vector case has been developed. This implementation is an interesting evolution of the SimpleSVM algorithm and is available on our website [Loosli, 2004]. The efficiency of our method has been illustrated on the USPS database. Our results outperform the equivalent algorithm Virtual-SVM in a significantly shorter computational time. We are now carrying on a deeper study of the comparison with the SDPM.

This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[Atteia and Gaches, 1999]Marc Atteia and Jean Gaches. *Approxiation Hilbertienne.* Presses Universitaires de Grenoble, 1999.

[Chapelle and Schölkopf, 2002]O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear SVMs. In Dietterich T. G.and Becker S. and Ghahramani Z., editors, *Advances in Neural Information Processing Systems*, volume 14, pages 609–616, Cambridge, MA, USA, 2002. MIT Press.

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

[Graepel and Herbrich, 2004]Thore Graepel and Ralf Herbrich. Invariant pattern recognition by semi-definite programming machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[Keysers *et al.*, 2002]D. Keysers, R. Paredes, H. Ney, and E. Vidal. Combination of tangent vectors and local representation for handwritten digit recognition. In *Lecture Notes in Computer Science*, volume LNCS 2396, pages 538–547. SPR2002, International Workshop on Statistical Pattern Recognition, Windsor, Ontario, Canada, springer-vertag edition, Aug 2002.

[Loosli *et al.*, 2004]G. Loosli, S. Canu, S.V.N. Vishwanathan, Alexander J. Smola, and Monojit Chattopadhyay. Une boîte à outils rapide et simple pour les SVM. In Michel Liquière and Marc Sebban, editors, *CAp 2004 - Conférence d'Apprentissage*, pages 113–128. Presses Universitaires de Grenoble, 2004.

[Loosli, 2004]G. Loosli. Fast SVM toolbox in MATLAB based on SimpleSVM algorithm, 2004. `http://asi.insa-rouen.fr/~gloosli/simpleSVM.html`.

[Pozdnoukhov and Bengio, 2003]A. Pozdnoukhov and S. Bengio. Tangent vector kernels for invariant image classification with SVMs. IDIAP-RR 75, IDIAP, Martigny, Switzerland, 2003. Submitted to International Conference on Pattern Recognition 2004.

[Schölkopf *et al.*, 1996]B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In J.C. Vorbrüggen 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.

[Simard *et al.*, 1993]P. Simard, Y. LeCun, and Denker J. Efficient pattern recognition using a new transformation distance. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann, 1993.

[Vishwanathan *et al.*, 2003]S. V. N Vishwanathan, A. J. Smola, and M. Narasimha Murty. SimpleSVM. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[Wood, 1996]Jeffrey Wood. Invariant pattern recognition: A review. *Pattern Recognition*, 29 Issue 1:1–19, 1996.