Semi-Supervised Learning through Principal Directions Estimation

Olivier Chapelle, Bernhard Schölkopf, Jason Weston
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany
{*first.last*}*@tuebingen.mpg.de*

Abstract

We describe methods for taking into account unlabeled data in the training of a kernel-based classifier, such as a Support Vector Machines (SVM). We propose two approaches utilizing unlabeled points in the vicinity of labeled ones. Both of the approaches effectively modify the metric of the pattern space, either by using non-spherical Gaussian density estimates which are determined using EM, or by modifying the kernel function using displacement vectors computed from pairs of unlabeled and labeled points. The latter is linked to techniques for training invariant SVMs. We present experimental results indicating that the proposed technique can lead to substantial improvements of classification accuracy.

1 Introduction

In *semi-supervised* learning, one is usually given a relatively small number of labeled training examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \{\pm 1\} \tag{1}$$

and a large amount of unlabeled points

$$\mathbf{x}'_1, \ldots \mathbf{x}'_{n_u} \in \mathcal{X}. \tag{2}$$

Here, $\mathcal{X}$ is assumed to be a (nonempty) set.

A typical example of a semi-supervised problem is encountered in web page classification [1]. Here, the number of unlabeled examples is effectively unlimited, but labeled data sets are small since they require human work. A review of the field of semi-supervised learning has recently been given in [11].

The hope in semi-supervised learning is that the labeled examples provide information about the decision function, while the unlabeled examples help to reveal the structure of the data, giving us additional information on how to best deal with the labeled data. This hope has been corroborated by a number of experimental studies [1, 7, 8], where improved classification accuracies were obtained by adding unlabeled data.

One might argue that this is futile, since the unlabeled points contain no information about the conditional distribution $dP(y|\mathbf{x})$ of the labels given the inputs.

Without any assumption, it turns out that the information they do provide, about the marginal distribution $dP(\mathbf{x})$, is useless [11].

However, one should bear in mind that actually, even the labeled data alone do not allow us to *generalize* to previously unseen points (this is sometimes referred to as the "no-free-lunch-Theorem"). It is well known that for learning to be successful, we need both training data and a priori knowledge on the class of functions that the estimate is taken from (e.g., in the form of a structure on the function space, or via a prior distribution). This is where unlabeled data can help — if used wisely, they can help us determining the function space we are using.

In the present paper, this is done by effectively adapting the metric that is used in the pattern space (cf. also [4]), thus indirectly influencing the function class that is being used.

The remainder of the article is organized as follows: in the next section, we review the problem of risk minimization, and its connection to semi-supervised learning. In Section 3, we expand the ideas developed in the previous section by utilizing connections to invariant SVMs. The following section gives some algorithmic details on how we carry out the approach in feature spaces associated with SVM kernels. Section 5 summarized the practicalities of our algorithms. Finally, Section 6 reports experimental results, followed by a brief discussion.

2 Vicinal Risk

The *Vicinal Risk Minimization* (VRM) induction principle introduced in [3] is an extension of the *Empirical Risk Minimization* principle. It consists of two steps. First, perform a Parzen window estimate of the true underlying probability distribution,

$$dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} dP_{\mathbf{x}_i}(\mathbf{x}) \delta_{y_i}(y). \tag{3}$$

Here, $\mathbf{x}, \mathbf{x}_i \in \mathcal{X}$, and $dP_{\mathbf{x}_i}$ denotes a probability distribution on $\mathcal{X}$, parametrized by $\mathbf{x}_i$ (e.g. the distribution corresponding to a kernel density estimator). We consider the case of pattern recognition, where the outputs y, y_i are elements of ± 1.

The second step is to find the function (mapping into $\{\pm 1\}$) minimizing the *vicinal risk*,

$$R_{vic}(f) = \int \mathbb{I}_{f(\mathbf{x}) \neq y} \, dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} \int \mathbb{I}_{f(\mathbf{x}) \neq y_i} dP_{\mathbf{x}_i}(\mathbf{x}) \tag{4}$$

Spherical Gaussians $\mathcal{N}(\mathbf{x}_i, \sigma^2)$ are an obvious choice of the local estimate $dP_{\mathbf{x}_i}(\mathbf{x})$. If $\sigma = 0$, the vicinal turns out to be the empirical risk whereas the function minimizing the vicinal risk when $\sigma \to 0$ over the set of linear functions is the maximum margin solution [3].

When a lot of unlabeled data are available, it can be worth trying to estimate more accurately the marginal distribution. For instance, let us suppose that to each labeled sample $\mathbf{x}_i$, we associate a local covariance matrix Σ_i and let us rewrite equation (3) as

$$dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{x}_i)^\top \Sigma_i^{-1} (\mathbf{x} - \mathbf{x}_i) \right) \delta_{y_i}(y). \tag{5}$$

Summing equation (5) over y gives an estimate of the marginal distribution $dP(\mathbf{x})$. The matrices Σ_i can be estimated with EM in order to maximize the likelihood of

the unlabeled data with respect to this estimate of the marginal distribution. If the data lie on a manifold of small dimension, Σ_i will be flat and equation (5) can be seen as a special case of a Mixture of Factor Analyzers [5] with centers $\mathbf{x}_i$ being the training points and with equal minxing parameters $1/n$.

3 The Connection to Invariant SVMs

Let us add the additional constraint that in equation (5) all the covariances matrices are equal $\Sigma_1 = \cdots = \Sigma_n = \Sigma$. This constraint might seem very restrictive, but as we shall see, doing so in feature space yields already significant improvements.[1] Now let us consider the following linear transformation of the input space: $\tilde{\mathbf{x}} = \Sigma^{-1/2}\mathbf{x}$. After this transformation, the ellipsoidal Parzen window estimate (5) of the density becomes spherical. But remember that in the case of a spherical Parzen window with a bandwidth going to 0, the linear function minimizing the vicinal risk is the maximum margin one (provided that the training set is linearly separable) [3]. For this reason, we suggest to run a linear SVM on the transformed patterns $\tilde{\mathbf{x}}_i$.

This kind of preprocessing for linear SVMs has already been used to train invariant SVMs [9]. Suppose that one knows a set of transformations $\mathcal{L}_t^1, \ldots, \mathcal{L}_t^p$, such that for any k and for any small value of the transformation parameter t, an input pattern $\mathbf{x}$ should belong to the same class as its transformed version $\mathcal{L}_t^k(\mathbf{x})$. For instance, on a digit recognition task, the invariance transformations include horizontal and vertical translations, rotations or dilatations. The vector $d_k\mathbf{x}_i = \partial\mathcal{L}_t^k(\mathbf{x}_i)/\partial t$ is called a *tangent vector* and represents a direction of invariance. In [9], the authors propose to compute the tangent covariance matrix,

$$B = \frac{1}{np}\sum_{i=1}^{n}\sum_{k=1}^{p} d_k\mathbf{x}_i \; d_k\mathbf{x}_i^\top,$$

and to perform the following linear transformation of the input space before training a linear SVM,

$$\mathbf{x} \leftarrow (\gamma B + (1-\gamma)I)^{-1/2}\mathbf{x}, \tag{6}$$

where γ is a parameter between 0 and 1. Performing such a preprocessing is equivalent to finding the hyperplane which tries to maximizing the margin, while being parallel to the tangent vectors (this trade-off being controlled by the value of γ).

The common point between the approach proposed in this paper for semi-supervised learning and invariant SVMs is that in both cases, we try to take into account the principal directions of the data, or in other words, the orientation of the manifold on which the data lie. The only difference is that the local covariance matrix Σ is not computed from known invariances as $\gamma B + (1-\gamma)I$, but is estimated directly from the set of unlabeled data.

4 Kernelization

Up to now, we only considered linear decision boundaries. It is possible to extend the algorithm presented in this paper to the non-linear case by using the so-called "kernel trick": the data are mapped implicitely to a high-dimensional feature space through a mapping function $\Phi : \mathcal{X}^d \mapsto \mathcal{H}$ in which the linear algorithm is carried

[1] Essentially, the restriction is similar to the one underlying SVMs, i.e., that the data are usually fairly well separable by a hyperplane in the feature space, provided a "sufficiently nonlinear" kernel is chosen. In both cases, a seemingly severe restriction is made, but the space of possible solutions is extended by the freedom of using kernels.

out. If the algorithm depends only on dot products, the mapping Φ does not need to be carried out explicitly; instead only the dot products in feature space $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$ have to be computed through the *kernel* function K.

Unfortunately, this approach cannot be implemented directly for our algorithm since it is not expressed only in terms of dot products: for instance, after having mapped the data in feature space, the covariance matrix Σ has the size of the dimension of the feature space and cannot be inverted if this latter is very high. For this reason, we propose an alternative method. The idea is to use directly the so-called *kernel PCA map*, first introduced in [10] and extended in [12]. It was also used in [2] to extend linear invariant SVMs to the non-linear case and in [6] to construct an algorithm for nonlinear ICA.

This map is based on the fact that even in a high dimensional feature space $\mathcal{H}$, a training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of size n when mapped to this feature space spans a subspace $E \subset \mathcal{H}$ whose dimension is at most n. More precisely, if $(\mathbf{v}_1, \ldots, \mathbf{v}_n) \in E^n$ is an orthonormal basis of E with each $\mathbf{v}_i$ being a principal axis of $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)\}$, the kernel PCA map $\psi : \mathcal{X} \to \mathbb{R}^n$ is defined coordinatewise as

$$\psi_p(\mathbf{x}) = \Phi(\mathbf{x}) \cdot \mathbf{v}_p, \quad 1 \leq p \leq n.$$

Each $\mathbf{v}_i$ has a linear expansion in terms of the training points $\{\Phi(\mathbf{x}_i)\}$ and the coefficients of this expansion are obtained using kernel PCA [10]. Writing the eigendecompostion of the kernel matrix K as $K = U\Lambda U^\top$, with U an orthonormal matrix and Λ a diagonal one, it turns out that the the kernel PCA map reads

$$\psi(\mathbf{x}) = \Lambda^{-1/2} U^\top \mathbf{k}(\mathbf{x}), \tag{7}$$

where

$$\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n))^\top.$$

Note that by definition, for all i and j, $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ lie in E and thus $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \psi(\mathbf{x}_i) \cdot \psi(\mathbf{x}_j)$. This reflects the fact that if we retain all principal components, kernel PCA is just a basis transform in E, leaving the dot product of training points invariant.

As a consequence, training a nonlinear SVM on $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is equivalent to training a linear SVM on $\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)\}$ and thus, thanks to the nonlinear mapping ψ, we can work directly in the linear space E and use exactly the method described for the *linear* case in the previous sections. Note however that if only the labeled points are used to compute the kernel PCA map, the unlabeled points do not necessarily belong to E and by projecting them onto E some information might be lost. Adding some additional unlabed points for the computation of the empirical kernel PCA map increases the computational complexity (since the eigendecompostion of the Gram matrix scales as the cube of the number of points), but also enlarges the space E and less information is lost by projecting the points onto E. In our experiments, a good trade-off consists in selecting around 500 points to compute the map.

5 Practical Implementation

In summary, the proposed algorithm is the following,

1. Fix a kernel function K
2. Compute the empirical kernel PCA map ψ as described in section 4 using p randomly selected labeled or unlabeled points. For computational reasons, p should not be much more than 1000.

3. For each labeled, unlabeled and test point, compute $\psi(\mathbf{x}) \in \mathbb{R}^p$. Now, linear techniques can be used on these transformed points in $\mathbb{R}^p$.
4. Use the EM algorithm to find the matrix $\Sigma \in \mathbb{R}^{p \times p}$ which maximizes the likelihood of the unlabeled data under the model (5).
5. Train a linear SVM on the labeled points $\Sigma^{-1/2}\psi(\mathbf{x}_i)$ and get $\mathbf{w} \in \mathbb{R}^p$ and b the hyperplanes parameters.
6. A test point is classified as the sign of $\mathbf{w} \cdot \Sigma^{-1/2}\psi(\mathbf{x}_i) + b$.

We give the details of step 4 and then propose a fast approximation of it.

5.1 Details of the EM step

The EM algorithm yields a sequence a covariances matrices $\Sigma^{(1)}, \ldots, \Sigma^{(q)}, \ldots$ computed as follows. At the q-th iteration, the E step consists of estimating the posterior probabilities

$$P(\mathbf{x}|\mathbf{x}_i) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma^{(q)}|}} \exp\left(-\frac{1}{2}(\psi(\mathbf{x}) - \psi(\mathbf{x}_i))^\top (\Sigma^{(q)})^{-1}(\psi(\mathbf{x}) - \psi(\mathbf{x}_i))\right),$$

for each labeled training point $\mathbf{x}_i$ and each unlabeled point $\mathbf{x} = \mathbf{x}'_1, \ldots, \mathbf{x}'_{n_u}$. And the M step is

$$\Sigma^{(q+1)} = \frac{1}{n_u}\sum_{i=1}^{n}\sum_{j=1}^{n_u} P(\mathbf{x}_i|\mathbf{x}'_j)(\psi(\mathbf{x}_i) - \psi(\mathbf{x}'_j))(\psi(\mathbf{x}_i) - \psi(\mathbf{x}'_j))^\top, \tag{8}$$

where $P(\mathbf{x}_i|\mathbf{x}'_j)$ is computed using Bayes rule as

$$P(\mathbf{x}_i|\mathbf{x}'_j) = \frac{P(\mathbf{x}'_j|\mathbf{x}_i)P(\mathbf{x}_i)}{\sum_k P(\mathbf{x}'_j|\mathbf{x}_k)P(\mathbf{x}_k)} = \frac{P(\mathbf{x}'_j|\mathbf{x}_i)}{\sum_k P(\mathbf{x}'_j|\mathbf{x}_k)}. \tag{9}$$

The last equality comes from the fact that we decided to set the mixing coefficients between the different Gaussians to $P(\mathbf{x}_i) = 1/n$.

A possible improvement is the following: modify the calculation of $P(\mathbf{x}_i|\mathbf{x}'_j)$ (equation 9) by adding a constant ε in the denominator. By doing so, the unlabeled points $\mathbf{x}'_j$ which are far away from the labeled ones (i.e. for which $P(\mathbf{x}'_j|\mathbf{x}_i) \ll \varepsilon,\ \forall i$) will give $P(\mathbf{x}_i|\mathbf{x}'_j) \approx 0,\ \forall i$, and will be neglected in the calculation of the covariance matrix. This is a desirable feature since we want to estimate a local covariance matrix and unlabeled points which are away from all the labeled ones should not be taken into account.

This modification can be seen as the solution the EM algorithm applied to the following model of the input distribution,

$$P(\mathbf{x}) = \frac{1-\varepsilon'}{n}\sum_{i=1}^{n} \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\psi(\mathbf{x}) - \psi(\mathbf{x}_i))^\top \Sigma^{-1}(\psi(\mathbf{x}) - \psi(\mathbf{x}_i))\right) + \varepsilon' P_0,$$

where P_0 is a uniform density and $\varepsilon' = \varepsilon / P_0(\mathbf{x})$.

5.2 Estimation Using Nearest Neigbors

Actually, we did not estimate Σ (step 4) using the EM algorithm because of compuational reasons: the sum (8) has $n \times n_u$ terms and the compuational cost of using EM can can be quite large.

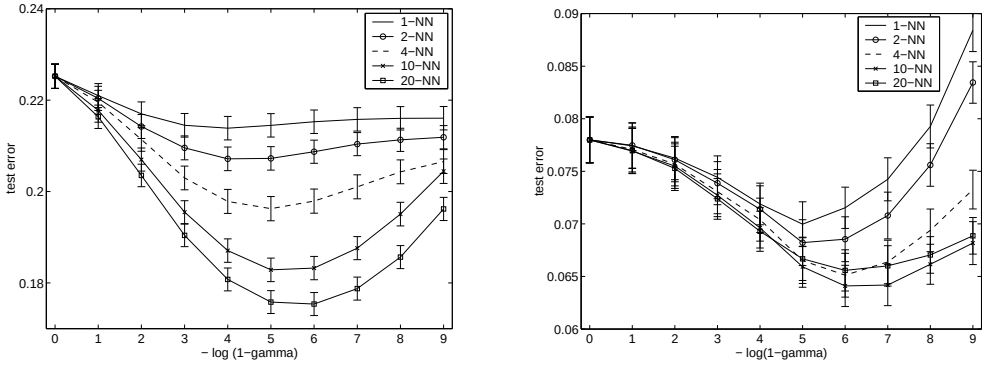

Figure 1: Test error on the USPS database using invariant SVMs and where invariances are computed from the k nearest neighbors ($k = 1, 2, 4, 10$ or 20. Results averaged over different training sets of size 30 (left) or 317 (right). The value $\gamma = 0$, i.e. the left end of the x-axis, corresponds to no usage of unlabeled data (cf. (6)); as one moves to the right, more and more emphasis is places on utilizing the tangent vectors computed from unlabeled points.

Insteas, we derived a more direct approach involving the nearest neighbors. For each labeled point $\mathbf{x}_i$, let $\mathbf{x}_{unl(i,1)}, \ldots \mathbf{x}_{unl(i,k)}$ be the k nearest unlabeled points from $\mathbf{x}_i$ in the kernel PCA space. Then we defined

$$\Sigma \equiv \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} (\psi(\mathbf{x}_i) - \psi(\mathbf{x}_{unl(i,j)}))(\psi(\mathbf{x}_i) - \psi(\mathbf{x}_{unl(i,j)}))^\top. \tag{10}$$

Note that equation (10) and (8) are very similar: the only difference is that $P(\mathbf{x}_i|\mathbf{x}'_j)$ is replaced by 0 or 1 according to whether or not the unlabeled point $\mathbf{x}'_j$ is a nearest neigbor of $\mathbf{x}_i$.

In high dimensions, if n and k are small, Σ might not be invertible. As in the case of invariant learning (cf equation (6)), we actually considered a regularized version of Σ, $\Sigma_\gamma = \gamma\Sigma + (1-\gamma)I$.

Estimating Σ using equation (10) instead of EM gives a much more direct interpretation in terms of invariant learning: since it is likely that the nearest neighbors of a labeled points are of the same class, the vectors $\psi(\mathbf{x}_i) - \psi(\mathbf{x}_{unl(i,j)})$ can be interpreted as *tangent vectors* and the invariant SVM algorithm [2] coincides with the one presented in this paper.

6 Experiments

Experiments have been carried on the USPS database using a polynomial kernel of degree 3 and the task being to discriminate digits 0 to 4 from 5 to 9. The kernel PCA map has been computed using a random set of $p = 500$ examples. The training set of 7291 has been divided in either 23 subsets of 317 examples or 243 of size 30. For each subset of $n = 30$ or 317 labeled examples, the $7291 - n$ remaining examples form the unlabeled set. Results are presented in figure 1 and seem very promising.

On this problem, we also tried the Transductive SVM algorithm [7] which finds the hyperplane maximizing the margin on *both* labeled and unlabeled data, but we did

not succeed in obtaining an improvement with this algorithm. eventhough it demonstrated significant improvements in test error when applied to semi-supervised text classification tasks [7].

7 Discussion

We have proposed an approach for incorporating unlabeled data into kernel methods by effectively changing the metric of the pattern space. Our initial experimental results are rather promising, and indicate that the present approach may well be a good candidate solution for two problems that have recently received significant attention in the machine learning community: the problem of utilizing unlabeled data, and the problem of choosing a kernel function suitable for a given task.

References

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1998.

[2] O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear Support Vector Machines. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[3] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

[4] C. Domeniconi and D. Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[5] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Departement of Computer Science, University of Toronto, 1997.

[6] S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel feature spaces and nonlinear blind source separation. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[7] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999.

[8] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

[9] B. Schölkopf, P. Y. Simard, A. J. Smola, and V. N. Vapnik. Prior knowledge in support vector kernels. In MIT Press, editor, *NIPS*, volume 10, 1998.

[10] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[11] M. Seeger. Learning with labeled and unlabeled data. Technical report, Edinburgh University, 2001.

[12] K. Tsuda. Support vector classifier with asymmetric kernel function. In M. Verleysen, editor, *Proceedings of ESANN'99*, pages 183–188, 1999.