Kernel whitening for one-class classification

David M.J. Tax[1] and Piotr Juszczak[2]

[1] Fraunhofer Institute FIRST.IDA,
Kekuléstr.7, D-12489 Berlin, Germany
davidt@first.fraunhofer.de
[2] Pattern Recognition Group
Faculty of Applied Science, Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, The Netherlands
piotr@ph.tn.tudelft.nl

Abstract. In one-class classification one tries to describe a class of target data and to distinguish it from all other possible outlier objects. Obvious applications are areas where outliers are very diverse or very difficult or expensive to measure, such as in machine diagnostics or in medical applications. In order to have a good distinction between the target objects and the outliers, good representation of the data is essential. The performance of many one-class classifiers critically depends on the scaling of the data and is often harmed by data distributions in (non-linear) subspaces. This paper presents a simple preprocessing method which actively tries to map the data to a spherical symmetric cluster and is almost insensitive to data distributed in subspaces. It uses techniques from Kernel PCA to rescale the data in a kernel feature space to unit variance. This transformed data can now be described very well by the Support Vector Data Description, which basically fits a hypersphere around the data. The paper presents the methods and some preliminary experimental results.

1 Introduction

In almost all machine learning and pattern recognition research, it is assumed that a (training) dataset is available which reflects well what can be expected in practice. On this data a classifier or regressor should be fitted such that good generalization over future instances will be achieved [1]. Unfortunately, it is very hard to guarantee that the training data is a truly identically distributed sample from the real application. In the data gathering process certain events can easily be missed, because of their low probability of occurance, their measuring costs or because of changing environments.

In order to detect these 'unexpected' or 'ill represented' objects in new, incoming data, a classifier should be fitted, which detects the objects that do not resemble the bulk of the training data in some sense. This is the goal of one-class classification [2, 3], novelty detection [4], outlier detection [5] or concept learning [6]. Here, one class of objects, the *target* class, has to be distinguished from all other possible objects, the *outlier* objects.

A common solution for outlier or novelty detection is to fit a probability density on the target data [7, 5, 8], and classify an object as outlier when the object falls into a region with density lower than some threshold value. This works well in the cases where the target data is sampled well. That means, the sample size is sufficient and the distribution is representative. But density estimation requires large sample sizes. When the boundary of the target class with limited sample size is to be estimated, it might be better to directly fit the boundary instead of estimating the complete target density. This is Vapnik's principle to avoid solving a more general problem than what is actually needed to solve [9]. Using this principle, the problem is changed from density estimation to domain description.

The support vector data description (SVDD, [2]) is a method which tries to fit directly a boundary with minimal volume around the target data without performing density estimation. It is inspired by the (two-class) support vector classifier [9]. All objects inside the hypersphere will be 'accepted' and classified as target objects. All other objects are labeled outliers. By minimizing the volume of the hypersphere, it is hoped that the chance of accepting outliers is minimized. In [10] a linear one-class classifier is presented, based on the idea to separate the data with maximal margin from the origin. In [11] again a linear classifier was used, but here the problem was posed as a linear programming problem, instead of a quadratic progamming problem.

In general, the hypersphere model is not flexible enough to give a tight description of the target class and analogous to the Support Vector Classifier (SVC), the SVDD is made more flexible by transforming the objects from the input space representation to a representation in kernel space. It appears that not all kernels that were proposed for the SVC can be used by the SVDD. In most cases the data classes are elongated, which is useful for discrimination between two classes, but is harmful for one-class classification. An exception is the Gaussian kernel, where good performances can be obtained.

Unfortunately, even using the Gaussian kernel still an homogeneous input feature space is assumed, which means that distances in all directions in the space should be comparable. In practice, data is often distributed in subspaces, resulting in very small typical distances between objects in directions perpendicular to the subspace. Moving inside the subspace will change the objects just slightly, but moving out of the subspace will result in an illegal object, or an outlier. Although comparable distances are traveled, the class memberships of the objects differ drastically. This homogeneity of the distances does not just harm the SVDD, but in principle all one-class methods which rely on distances or similarities between the objects.

In this paper we propose a rescaling of the data in the kernel feature space, which is robust against large scale differences in scaling of the input data. It rescales the data in a kernel space such that the variances of the data are equal in all directions. We will use the techniques of Kernel-PCA [12]. In section 2 we will present the SVDD, followed by an example where it fails. In section 3 the rescaling of the data is presented, followed by some experiments and conclusions.

2 SVDD

To describe the domain of a dataset, we enclose the data by a hypersphere with minimum volume (minimizing the chance of accepting outlier objects). Assume we have a d-dimensional data set containing n data objects, $\mathcal{X}^{tr} : \{\mathbf{x}_i, i = 1, .., n\}$ and the hypersphere is described by center $\boldsymbol{a}$ and radius R. We will assume throughout the paper that a sum $\sum_i$ will sum over all training objects, which means $\sum_{i=1}^{n}$.

To allow the possibility of outliers in the training set, the distance from $\mathbf{x}_i$ to the center $\boldsymbol{a}$ need not be strictly smaller than R^2, but larger distances should be penalized. An extra parameter ν is introduced for the trade-off between the volume of the hypersphere and the errors. Thus, an error function L, containing the volume of the hypersphere and the distances, is minimized. The solution is constrained with the requirement that (almost) all data is within the hypersphere. The constraints can be incorporated in the error function by applying Lagrange multipliers [1]. This yields the following function to maximize with respect to $\boldsymbol{\alpha}$ (for details [2]):

$$L = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad \text{with} \quad 0 \leq \alpha_i \leq \frac{1}{n\nu}, \quad \sum_i \alpha_i = 1 \tag{1}$$

and

$$\boldsymbol{a} = \sum_i \alpha_i \mathbf{x}_i \,. \tag{2}$$

The last constraint in (1) influences the effective range of hyperparameter ν. For $\nu > 1$ this constraint cannot be met, and therefore in practice ν is $0 \leq \nu \leq 1$. (This hyperparameter plays the same role of ν in an comparable one-class classifier, the ν-SVC [10].)

Now (1) is in a standard quadratic optimization problem. By the box constraints, the free parameters α_i after optimization can be in two situations. Most objects $\mathbf{x}_i$ will satisfy $\|\mathbf{x}_i - \boldsymbol{a}\|^2 < R^2, \alpha_i = 0$ and for just a few objects $\mathbf{x}_i$ $\alpha_i > 0$. Analogous to [9] these objects are called the *support objects*, because they determine the (center of the) hypersphere via (2). A new object $\mathbf{z}$ is accepted by the description (or classified as target object) when:

$$\begin{aligned} f(\mathbf{z}) &= \|\mathbf{z} - \boldsymbol{a}\|^2 \\ &= (\mathbf{z} \cdot \mathbf{z}) - 2 \sum_i \alpha_i (\mathbf{z} \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2 \,. \end{aligned} \tag{3}$$

The radius R is determined by calculating the distance from the center $\boldsymbol{a}$ to any support vector $\mathbf{x}_i$ on the boundary.

The hyperspherical shape for the boundary of a dataset is very restricting and will not be satisfied in the general case. Analogous to the method of Vapnik [9], we can replace the inner products $(\mathbf{x} \cdot \boldsymbol{y})$ in Equations (1) and in (3) by kernel functions $K(\mathbf{x}, \boldsymbol{y}) = \Phi(\mathbf{x}) \cdot \Phi(\boldsymbol{y})$ (where K is a positive definite kernel, or Mercer kernel). By this replacement of the inner product by K, the data is implicitly

mapped to a new feature space. Ideally, this mapping would map the data into a spherical constrained domain, such that the assumptions for the SVDD are fulfilled.

Several kernels have been proposed [9], mainly in the application of Support Vector Classifiers. A popular choice is the polynomial kernel $(\mathbf{x} \cdot \boldsymbol{y}) \rightarrow K(\mathbf{x}, \boldsymbol{y}) = (\mathbf{x}\boldsymbol{y} + 1)^p$, which maps the data to a feature space spanned by all monomial features up to p. For one-class classification this kernel works poorly, because it tends to transform the data into elongated, flat structures instead of spherical clusters. Especially for larger degrees p, taking the power will stress the differences in the variances in different feature directions. For large p the direction with largest variance in input space will overwhelm all smaller variances in kernel space.

For another popular kernel, the Gaussian kernel, this is not the case:

$$(\mathbf{x} \cdot \boldsymbol{y}) \rightarrow K(\mathbf{x}, \boldsymbol{y}) = \exp(-\|\mathbf{x} - \boldsymbol{y}\|^2/\sigma^2)\,. \tag{4}$$

The width parameter σ in the kernel (from definition (4)) determines the scale or resolution at which the data is considered in input space. Although here the data is implicitly mapped to an infinitely dimensional space $\mathbb{F}$ [13], the inner products (or the kernel outputs) are between 0 and 1. Furthermore, $K(\mathbf{x}, \mathbf{x}) = 1$ indicating that all objects have length 1, placing the objects effectively on a hypersphere with radius 1.

For good performance of the SVDD with the Gaussian kernel, still properly scaled distances are required. The new inner product (4) now depends on distance $\|\mathbf{x} - \boldsymbol{y}\|^2$. Very inhomogeneous distances will still result in elongated clusters and large empty areas around the target class in input feature space that are still accepted.

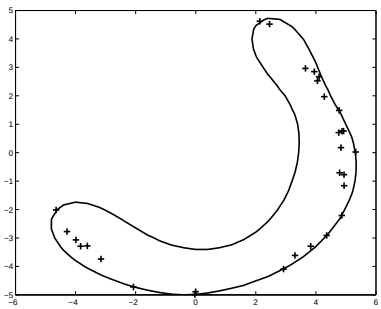

Fig. 1. Decision boundary of an SVDD trained on an artificial 2D dataset.

In figure 1 a scatterplot of an artificial 2-dimensional dataset is show. The SVDD is trained to fit a boundary around it such that about 25% of the target data is on the boundary. Although the SVDD follows the curve in the data, it does not fit tightly the subspace structure in the data. A large strip inside the

curve is classified as target object, but does not contain target training objects. This is caused by the large scale difference of the data parallel and perpendicular to the subspace.

In the approach of [10] a linear hyperplane instead of an hyperspherically shaped boundary is used. This plane should separate the target data with maximal margin from the origin of the feature space. Although in input space this is incomparable with the hypersphere approach, the method can be 'kernalized" and using the Gaussian kernel this method appears to be identical to the SVDD [2].

3 Kernel whitening

Instead of directly fitting a hypersphere in the kernel space, we propose to rescale the data to have equal variance. Fitting a hypersphere in the rescaled space $\mathbb{F}$ will be identical to fitting an ellipsoid in the original kernel space. The rescaling is easily done, using the derivation of the Kernel PCA [12]. The data is basically mapped onto the principal components (or the largest eigenvalues) of the data covariance matrix and then rescaled by the corresponding eigenvalues. Therefore the eigenvectors and eigenvalues of the covariance matrix in the kernel space have to be estimated. The eigenvectors with eigenvalues close or equal to zero will be disregarded.

Assume the data $\mathcal{X}^{tr}$ is mapped to the kernel space $\mathbb{F}$ by some (possibly non-linear) mapping $\Phi : \mathbb{R}^d \to \mathbb{F}$. When we also assume that the data is centered in this space, i.e. $\sum_i \Phi(\mathbf{x}_i) = 0$, the covariance matrix C of the mapped dataset can be estimated by:

$$C = \frac{1}{n} \sum_i \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T \,. \tag{5}$$

The eigenvectors $\boldsymbol{v}$ and eigenvalues λ satisfy:

$$C\boldsymbol{v} = \frac{1}{n} \sum_j \left(\Phi(\mathbf{x}_j) \cdot \boldsymbol{v}\right) \Phi(\mathbf{x}_j) = \lambda \boldsymbol{v} \,. \tag{6}$$

Equation (6) shows that the eigenvectors with non-zero eigenvalue must be in the span of the mapped data $\{\Phi(\mathbf{x}_i)\}$, which means that $\boldsymbol{v}$ can be expanded as:

$$\boldsymbol{v} = \sum_i \alpha_i \Phi(\mathbf{x}_i) \,. \tag{7}$$

Multiplying Equation (6) from the left with $\Phi(\mathbf{x}_k)$ and using (7) gives:

$$\frac{1}{n} \sum_j \left(\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_j)\right) \left(\Phi(\mathbf{x}_j) \cdot \sum_i \alpha_i \Phi(\mathbf{x}_i)\right) = \lambda \sum_i \alpha_i \left(\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i)\right) \quad \forall k \,. \tag{8}$$

When again the kernel matrix $K_{ij} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ is introduced, it appears that the coefficients $\boldsymbol{\alpha}$ from Equation (7) can directly be obtained by solving the eigenvalue problem:

$$\lambda \boldsymbol{\alpha} = K \boldsymbol{\alpha} \,. \tag{9}$$

For normal kernel-PCA the eigenvectors should be normalized to unit length, and this means that for each eigenvector $\boldsymbol{v}^k$ the $\boldsymbol{\alpha}^k$ are rescaled to:

$$\lambda_k \left(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k \right) = 1 \,. \tag{10}$$

We assumed that the data is centered in $\mathbb{F}$. This can be done by transforming the original kernel matrix. Assume K is the $n \times n$ kernel matrix of the training data and K^{tst} the $m \times n$ matrix of some new data (or possibly the same training data). The centered kernel matrix is computed by:

$$\tilde{K} = K^{\text{tst}} - 1_n^* K - K^{\text{tst}} 1_n + 1_n^* K 1_n \tag{11}$$

where 1_n is an $n \times n$ matrix and 1_n^* is an $m \times n$ matrix both with all entries $1/n$ [12]. We will assume that we always have centered the kernel matrices using (11).

When the coefficients $\boldsymbol{\alpha}$ are obtained, a new object $\mathbf{z}$ can be mapped onto eigenvector $\boldsymbol{v}^k$ in $\mathbb{F}$ by:

$$(\hat{\mathbf{z}})_k = (\boldsymbol{v}^k \cdot \Phi(\mathbf{z})) = \sum_i \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z})) = \sum_i \alpha_i^k K(\mathbf{x}_i, \mathbf{x}) \tag{12}$$

where $(\hat{\mathbf{z}})_k$ means the k-th component of vector $\hat{\mathbf{z}}$.

To transform the data into a representation with equal variance in each feature direction (for directions with $\lambda^k > 0$) the normalization from Equation (10) has to be slightly adapted. The variance of the mapped data along component $\boldsymbol{v}^k$ is:

$$\text{var}(\mathcal{X}^{tr}) = \frac{1}{n} \sum_j (\hat{\mathbf{x}}_j)^2 = \frac{1}{n} \sum_j \left(\sum_i \alpha_i^k k(\mathbf{x}_i, \mathbf{x}_j) \right)^2 = \frac{1}{n} (\boldsymbol{\alpha}^k)^T K K \boldsymbol{\alpha}^k \,. \tag{13}$$

Using Equation (9) this is constant for all features when instead of (10) we use the normalization:

$$\lambda_k^2 \left(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k \right) = 1 \quad \text{for all considered components } k \,. \tag{14}$$

The dataset $\hat{\mathcal{X}^{tr}}$, transformed using the mapping (12) with normalization (14), can now be used by *any* one-class classifier. The dimensionality of this dataset depends on how many principal components $\boldsymbol{v}^k$ are taken into account. Not only do all the features have equal variances, by the fact that the data is mapped onto the principal components of the covariance matrix, the data is also uncorrelated. The fact that the data is now properly scaled, makes it ideal to estimate a normal distribution, or to use the SVDD, which in the linear case just fits a hypersphere.

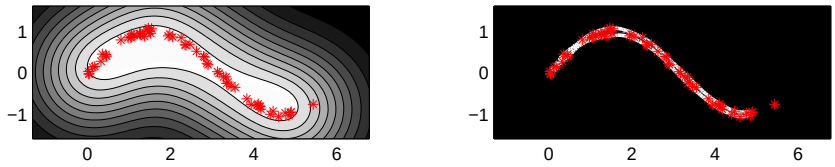

Fig. 2. The data description of a sinusoidal distributed dataset. Left shows an SVDD trained in the input space, the right shows the decision boundary of the hypersphere in kernel space. In both cases a Gaussian kernel with $\sigma = 4$ is used.

In figure 2 an artificial 2D dataset is shown, where the data is distributed in a sinusoidal subspace. In the left subplot the output of the SVDD is shown, in the right subplot the output of the SVDD with the data scaled to unit variance. In order to model this data well, a one-class classifier has to be very flexible, and large amounts of data should be available to follow both the large sinusoidal structure and be tight around the subspace. The SVDD are optimized to have about 30% error on the target set. The decision boundary are given by the white line. It is clear that it does not model the subspace structure in the data.

4 Characteristics whitening

How efficient the mapping of the data to the new representation with unit variance is, depends on the choice of the kernel and parameters. When this feature extraction captures the data structure, it is easy to train a one-class classifier on this data and obtain good classification performance. In table 1 decision boundaries for the artificial data for different choices of the kernels are shown. The left column shows the results for the polynomial kernel of degree $d = 1, d = 2$ to $d = 3$ (from top to bottom). The left column shows the results for the Gaussian kernel, for $\sigma = 5, 15$ and 50. Results show large dependence on the choice of the free parameter. The rescaling tend to overfit for high values of the degree d and low values of σ. Visually it can be judged that for the polynomial kernel $d = 3$ is reasonable, for the Gaussian kernel a σ between 15 and 50 can be used. Applying an ill-fitting kernel results in spurious areas in the input space.

Many one-class classifiers rely on the distances between the objects in the input space. When the data is whitened in the kernel space, and all significant eigenvectors are taken into account, the influence of rescaling (one of the) features is eliminated. In table 2 the results of rescaling one of the features is shown. In the middle row a scatterplot of the original data is shown. On this dataset an SVDD, an SVDD on the whitened data with all non-zero principal components, and an SVDD using just the first 5 principal components are trained. It appears that for this data there are just 8 non-zero principal components.

In the upper row of the table, the horizontal feature was rescaled to 10% of the original size, while on the lower row the data the feature was 10 times enlarged. The SVDD on the (kernel-) whitened data not only gives a tight description,

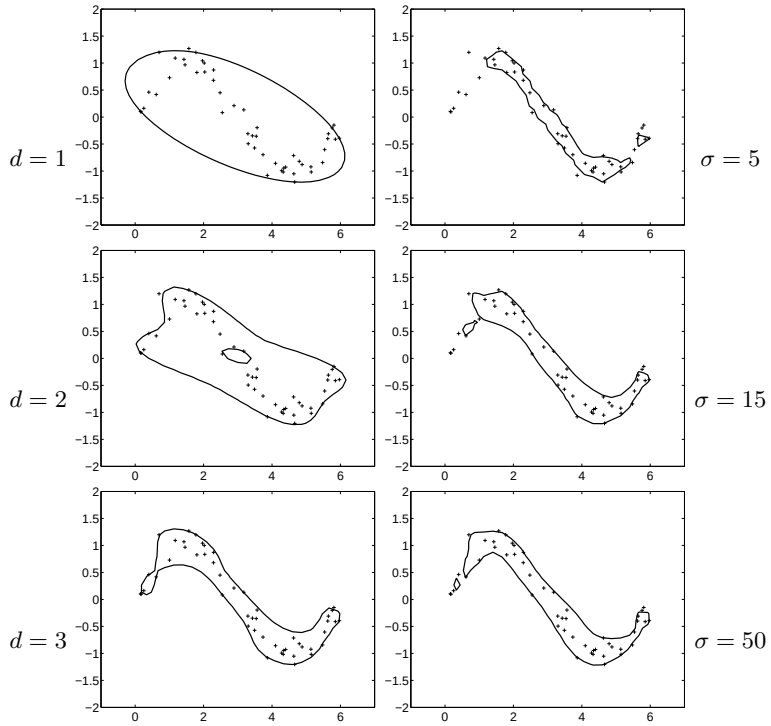

Table 1. The influence of the choice of the kernel. The left column shows the results using the polynomial kernel with varying degrees, the right column the Gaussian kernel with varying σ.

but is also robust against rescaling the single feature. The SVDD in input space heavily suffers from rescaling the data. Using just a few principal components from the mapped data also results in poorer results and in spurious areas.

The fact that the data is unit variance with uncorrelated features makes the normal distribution a good choice for describing the dataset in the kernel space. In figure 3 again the sinusoidal data set is shown, now with one prominent outlier present. Furthermore, typical decision boundaries of the fitted normal distribution (left) and the support vector data description (right) are shown. In most cases the difference in decision boundary between the SVDD and the Gaussian model are minor. In case the training data contains some significant outliers, The SVDD tends to be obtain tighter descriptions, because it can effectively ignore prominent outliers in the data. The normal distribution is still influenced by it, and starts to accept superfluous areas in feature space. This is also visible in figure 3. In both cases the decision boundary was optimized such that 10% of the training data is rejected.

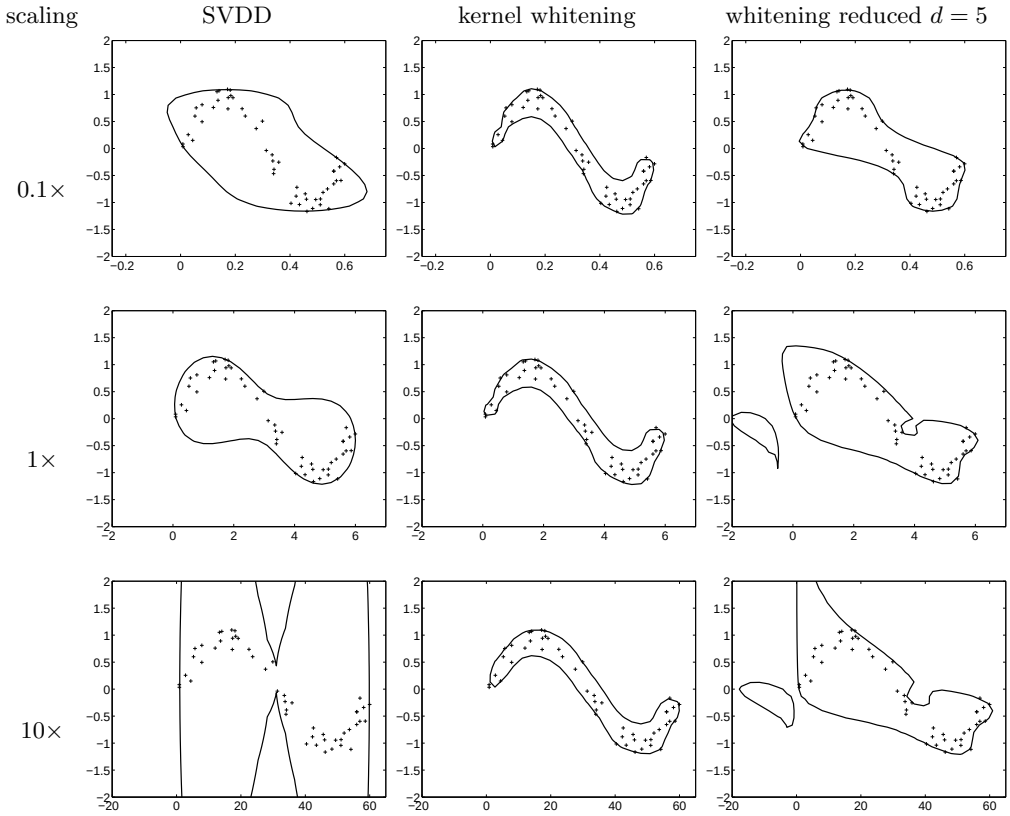

Table 2. Influence of the scaling of the features. The left column shows the decision boundary of the SVDD, the middle column the results of the data description using the whitening with all non-zero variance directions and the right column shows the output using the five principal components. The middle row shows the original data. In the upper row the horizontal feature is 10 times shrinked, the lower the horizontal feature is 10 times enhanced. For display purposes the data is scaled to show comparable scales.

5 Experiments

To show the results on real world datasets, we use the standard Concordia dataset [14], in which the digits are stored in 32×32 black-and-white images. Each of the digit classes can be designated to be the target class. Then all other digits are considered outliers. For training 400 objects per class, and for testing 200 objects per class are available.

In figure 4 typical images of rejected objects are shown. The one-class classifier was trained on class '2' and '3' respectively. The 32×32 images are first preprocessed to retain 80% of the variance, to remove all pixels with (almost) zero variance over the whole dataset. Then the data was (kernel-) whitened using

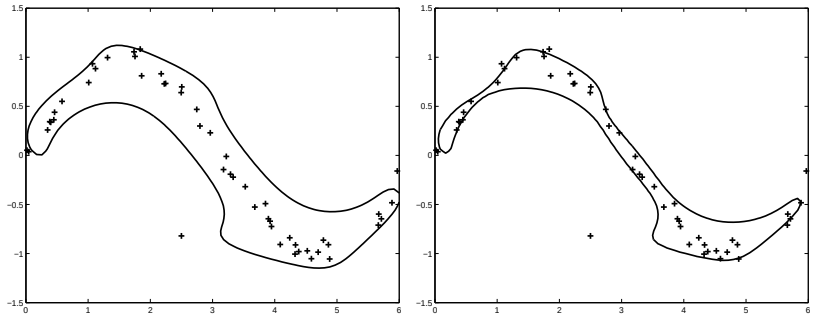

Fig. 3. Typical decision boundaries of the normal distribution (left) and the support vector data description (right), trained on the normalized data in $\mathbb{F}$. The SVDD tends to be tighter, especially when some outliers are present in the training data.

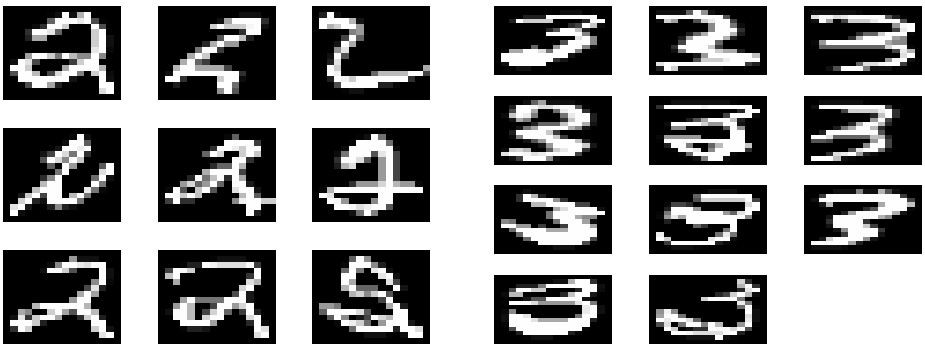

Fig. 4. Examples of rejected handwritten digits from the Concordia dataset. An SVDD trained on class 2 and 3, kernel whitened using a polynomial kernel with $d = 2$.

a polynomial kernel, degree 2. The first 20 principal components were chosen, the eigenvalues of the other principal components always were a factor 10^{-6} or more smaller than the largest eigenvalue. On this data a normal SVDD was fitted, such that about 5% of the target data is rejected. The results show that rejected objects are often skewed or are written very fatly, or contain big curls.

In figure 5 the results are shown to compare the outliers obtained by using normal PCA and the kernel whitening, using the polynomial kernel with degree 3. In the normal PCA 12 objects are rejected. Some of them look reasonable by human interpretation. In the kernel whitening processing, 10 objects are rejected. Some of the objects are rejected in both methods, for instance the upper left object in PCA and the second object in the kernel whitening. Other objects are specifically rejected because the do not fit the particular model, for instance the lower right object in both the PCA and kernel whitening.

In figure 6 results on all Concordia digit classes are shown. On each of the digit classes one-class classifiers are trained and the ROC curve is computed.

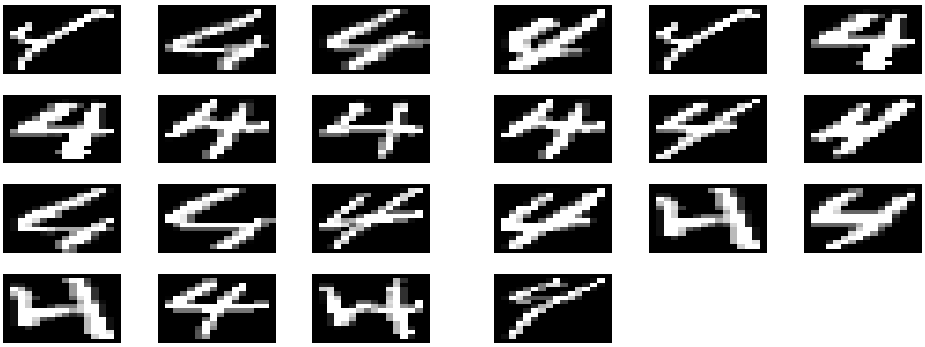

Fig. 5. An SVDD trained on digit class '4'. On the left the data was preprocessed using normal PCA, on the right kernel whitening with polynomial $d = 3$ is used.

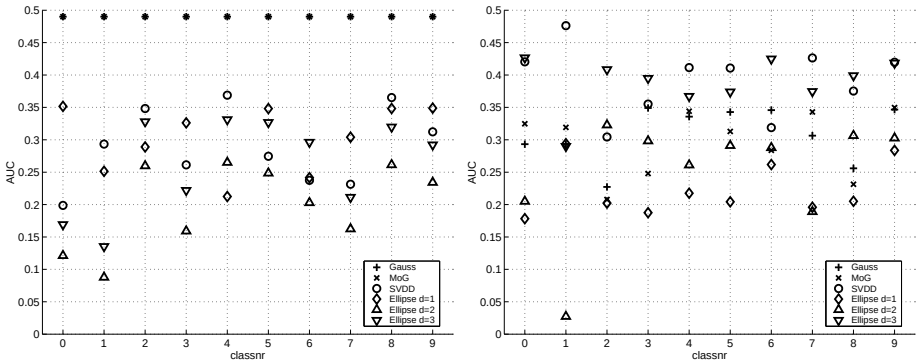

Fig. 6. AUC errors on the 10 classes of the Concordia handwritten digits. Left shows the AUC error on all 10 classes for a simple Gaussian density, a Mixture of Gaussians ($k = 5$), SVDD, whitening with polynomial degree 1, 2 and 3.

The ROC gives the error on the outlier data given varying values for the error on the target class [15]. From the ROC curves a error is derived, called the Area Under the ROC curve (AUC). Low values of the AUC indicates a good separation between the target and outlier data.

On each of the classes 6 one-class classifiers have been trained. The first two methods are density models: the Normal Density and the Mixture of Gaussians (with 5 clusters). The third is the basic SVDD directly trained in the input space, optimized such that about 10% of the target class is rejected. In the last three classifiers the data is mapped using the kernel whitening (polynomial kernel, $d = 1, 2$ and 3). Again the first 20 principal components were considered, retaining about 75% of the variance.

In the left subplot, the data is not preprocessed. The density methods are not capable in estimating the density and give the highest AUC error of 0.5. In most

cases the best performances is obtained by applying the whitening procedure with $d = 2$. The SVDD can perform poorly, due to the relative low sample size and the complexity of following the boundary in the high dimensional feature space. Whitening with higher polynomial degrees also suffers from low sample size effects.

In the right subplot, the data is preprocessed by basic to retain again 80% of the variance. By the reduction of the dimensionality, in some cases some overlap between the classes is introduced and the performance of the best whitening procedures deteriorate. The density methods now work well and often outperform the poorer whitening versions. The actual performance increase or decrease is mainly determined by how well the model fits the data. That means for the whitening procedure that good performance is obtained when the data is distributed in some (nonlinear) subspace.

6 Conclusions

This paper presents a simple whitening preprocessing for one-class classification problems. It uses the idea of Kernel PCA to extract the non-linear principal features of the dataset. After mapping the data to this new feature space (implicitly defined by the kernel function), feature directions with (almost) zero variance are removed and the other features are rescaled to unit variance. By the Kernel PCA and rescaling, the resulting data is zero mean with an identity covariance matrix. Finally, this data can now in principle be described by *any* one-class classifier.

By this preprocessing step one-class classifiers can be trained which contain large differences in scale in the input space. In particular, data in (non-linear) subspaces can be described well. For most one-class classifiers data distributed in subspaces are problematic, because the data contains large differences in typical scale *within* the subspace and *perpendicular* to the subspace. By using a suitable kernel in the kernel PCA, these scale differences in the data are recognized and modeled in the mapping. The transformed data now has equal variance in each feature direction.

This subspace modeling comes at a price though. The mapping requires a reasonable sample size, in order to extract the more complex non-linear subspaces. Using too complex mappings and too many principal components in combination with small sample sizes will result in overfitting on the data and in poor results on independent test data.

A drawback of this rescaling on the Kernel PCA basis is that the expansion in (13) is in general not sparse. This means that for each projection of a test point onto a principal direction, all training objects have to be taken into account. For large training sets this can become very expensive. Fortunately, approximations can be made, which reduce the number of objects in the expansion (7) drastically [10].

Finally, the problem how to choose the kernel function and values for the hyperparameters is still open. When test data is available, both from the target

as the outlier class, this can be used for evaluation of the model (which then includes both the whitening and the classifier in the kernel space). In the general case of one-class classification, we have just a very poorly represented outlier class, and estimating the performance on this dataset will give a bad indication of the expected performance. In these cases we have to rely on, for instance, artificially generated outlier data.

7 Acknowledgements

This research was supported through a European Community Marie Curie Fellowship. The author is solely responsible for information communicated and the European Commission is not responsible for any views or results expressed.

References

1. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Walton Street, Oxford OX2 6DP (1995)
2. Tax, D.: One-class classification. PhD thesis, Delft University of Technology, http://www.ph.tn.tudelft.nl/˜davidt/thesis.pdf (2001)
3. Moya, M., Hush, D.: Network contraints and multi-objective optimization for one-class classification. Neural Networks **9** (1996) 463–474
4. Ritter, G., Gallegos, M.: Outliers in statistical pattern recognition and an application to automatic chromosome classification. Pattern Recognition Letters **18** (1997) 525–539
5. Bishop, C.: Novelty detection and neural network validation. IEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks **141** (1994) 217–222
6. Japkowicz, N., Myers, C., Gluck, M.: A novelty detection approach to classification. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. (1995) 518–523
7. Tarassenko, L., Hayton, P., Brady, M.: Novelty detection for the identification of masses in mammograms. In: Proc. of the Fourth International IEE Conference on Artificial Neural Networks. Volume 409. (1995) 442–447
8. Surace, C., Worden, K., Tomlinson, G.: A novelty detection approach to diagnose damage in a cracked beam. In: Proceedings of SPIE. (1997) 947 – 943
9. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
10. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J.: SV estimation of a distribution's support. In: Advances in Neural Information Processing Systems. (1999)
11. Campbell, C., Bennett, K.P.: A linear programming approach to novelty detection. In: NIPS. (2000) 395–401
12. Scholkopf, B., Smola, A.J., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10** (1998) 1299–1319
13. Smola, A.: Learning with kernels. PhD thesis, Technischen University Berlin (1998)
14. Cho, S.B.: Recognition of unconstrained handwritten numerals by doubly self-organizing neural network. In: International Cconference on Pattern Recognition. (1996)
15. Metz, C.: Basic principles of ROC analysis. Seminars in Nuclear Medicine **VIII** (1978)