Outliers and data descriptions

David M.J. Tax, Robert P.W. Duin

Pattern Recognition Group, Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, The Netherlands
davidt@ph.tn.tudelft.nl

Keywords: pattern recognition, one-class problems, outlier detection, Support Vector Machines, Support Vector Data Description

Abstract

In previous research the support vector data description (SVDD) is proposed to solve the problem of one-class classification. In one-class classification, one set of data, called the target set, has to be distinguished from the rest of the feature space. In the original optimization of the support vector data description, two parameters have to be given beforehand by the user. In this paper a new, heuristic, error is defined. Minimizing this error, both free parameters in the SVDD can be determined without the use of example outlier objects. This paper shows under what circumstances the heuristic error correlates well with the true error.

1 Introduction

In one-class classification, the problem is to distinguish one class of data from the rest of the feature space. This situation can occur, for instance, when we want to monitor a machine. A classifier should detect when the machine is showing abnormal, faulty behavior. Measurements on the normal operation of the machine are easy to obtain, but in faulty situations, the machine might be destroyed completely. In these type of classification problems, one of the classes is characterized well, while for the other class (almost) no measurements are available. In one-class classification we assume that we have examples from just one class, called the target class and that all other possible objects, per definition the outlier objects, are uniformly distributed.

In general, the problem of one-class classification is harder than the problem of conventional two-class classification. For conventional classification the decision boundary is supported from both sides by examples of both classes. Because in the case of one-class classification only one set of data is available, only one side of the boundary is supported. It is hard to decide, on the basis of just one class, how strictly the boundary should fit around the data in each of the feature directions.

This one-class classification problem is often solved by estimating the target density [5], of by fitting a model to the data [4]. In this paper we use a method inspired by the support vector classifier [10] (or for a more simple introduction [7]). Instead of using a hyperplane to distinguish between two classes, a hypersphere around the target set is used. This method is called the support vector data description (SVDD)[8].

In the SVDD the volume of the hypersphere is minimized directly. It has the possibility to reject a fraction of the training objects, when it sufficiently decreases the volume of the hypersphere (a better explanation will be given in section 2). Unfortunately, the user has to supply a trade-off parameter C. Furthermore, the hypersphere model of the SVDD can be made more flexible by introducing kernel functions. Although it gives the possibility of making more flexible and better fitting descriptions, the user has to give the value of another parameter, the scale at which the important characteristics in the data are. The choice for these parameters is not directly intuitive, but they are important to find a good data description.

In this paper, we focus on the problem of the determination of the two variables, using only target objects (so no example outliers are required!). In section 2 we will explain the support vector data description, and introduce the free variables. In section 3 we show how s and C can be optimized and in section 4 this is applied to some classification tasks. We end with discussions in section 5.

2 Support vector data description

First we give a short derivation of the support vector data description. To describe the domain of a dataset, we enclose the data with a hypersphere with minimum volume. By minimizing the volume of the captured feature space, we hope to minimize the chance of accepting outlier objects. Assume we have a data set containing N data objects, $\{\mathbf{x}_i, i = 1, .., N\}$ and the hypersphere is described by center $\mathbf{a}$ and radius R.

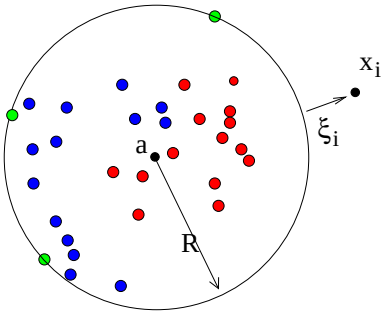

Figure 1: Graphical representation of the (hyper)sphere around some training data. One object is rejected by the description (i.e. an error).

A graphical representation is shown in figure 1. To allow the possibility of outliers in the training set, the distance from $\mathbf{x}_i$ to the center $\mathbf{a}$ should not be strictly smaller than R^2, but larger distances should be penalized. Therefore, we introduce slack variables ξ_i which measure the distance to the boundary, if an object is outside the description. An extra parameter C has to be introduced for the trade-off between the volume of the hypersphere and the errors.

Now, we minimize an error function L containing the volume of the hypersphere and the distance from the boundary of the outlier objects. We constrain the solution with the requirement that all data is within the hypersphere:

$$L(R, \mathbf{a}, \boldsymbol{\gamma}) = R^2 + C \sum_i \xi_i \tag{1}$$

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \qquad \forall_i \tag{2}$$

The constraints (2) can be incorporated in the error (1) by applying Lagrange multipliers [1]:

$$L(R, \mathbf{a}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\xi}) = R^2 + C \sum_i \xi_i \tag{3}$$
$$- \sum_i \alpha_i \{R^2 - (\mathbf{x}_i^2 - 2\mathbf{a} \cdot \mathbf{x}_i + \mathbf{a}^2)\} - \sum_i \gamma_i \xi_i$$

with Lagrange multipliers $\alpha_i \geq 0$ and $\gamma_i \geq 0$. This function has to be minimized with respect to $R, \mathbf{a}$ and ξ_i and maximized with respect to α_i and γ_i.

Setting the partial derivatives of L to R and $\mathbf{a}$ to zero, gives:

$$\sum_i \alpha_i = 1, \quad \mathbf{a} = \sum_i \alpha_i \mathbf{x}_i,$$
$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0, \quad \forall_i \tag{4}$$

From the last equation $\alpha_i = C - \gamma_i$ and because $\alpha_i \geq 0, \gamma_i \geq 0$, Lagrange multipliers γ_i can be removed when we demand that

$$0 \leq \alpha_i \leq C, \forall_i \tag{5}$$

Resubstituting these values in the Lagrangian (3) gives to maximize with respect to $\boldsymbol{\alpha}$:

$$L = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{6}$$

with $0 \leq \alpha_i \leq C$, $\sum_i \alpha_i = 1$.

This error function is in a standard quadratic form, and combined with the constraints, it gives a quadratic optimization problem. Error (6) should be maximized with respect to α_i. In practice it appears that a large fraction of the α_i becomes zero. For a small fraction, $\alpha_i > 0$ and the corresponding objects are called support objects. These objects appear to lie on the boundary (in figure 1 these are the three light gray objects on the boundary). We see that the center of the hypersphere depends just on the few support objects. The objects with $\alpha_i = 0$ can be disregarded in the description of the data.

Object $\mathbf{z}$ is accepted by the description when:

$$\|\mathbf{z} - \mathbf{a}\|^2 = (\mathbf{z} \cdot \mathbf{z}) - 2 \sum_i \alpha_i (\mathbf{z} \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2 \tag{7}$$

Radius R can be determined by calculating the distance from the center $\mathbf{a}$ to a support vector $\mathbf{x}_i$ on the boundary.

Here the model of a hypersphere is assumed and this will not be satisfied in the general case. Analogous to the method of Vapnik [10], we can replace the inner products $(\mathbf{x} \cdot \mathbf{y})$ in equations (6) and in (7) by kernel functions $K(\mathbf{x}, \mathbf{y})$ which gives a much more flexible method. When we replace the inner products by Gaussian kernels for instance, we obtain:

$$(\mathbf{x} \cdot \mathbf{y}) \rightarrow K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / s^2) \tag{8}$$

Equation (6) now changes into:

$$L = 1 - \sum_i {\alpha_i}^2 - \sum_{i \neq j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

The maximization of (9) gives $\boldsymbol{\alpha}$, which are used in the computation of the center $\mathbf{a}$. Now for a novel object $\mathbf{z}$ it can be checked if it is within the hypersphere (from (7)):

$$\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) \leq \frac{1}{2}\left(1 - R + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \tag{10}$$

This Gaussian kernel contains one extra free parameter, the width parameter s in the kernel (from definition (8)). For small values of s the SVDD resembles a Parzen density estimation, while for large s the original hypersphere solution is obtained [9]. As shown in [9] this parameter can be set by setting a priori the maximal allowed rejection rate of the target set, i.e. the error on the target set. Secondly, we also have the trade-off parameter C. We can define a new variable ν:

$$\nu = \frac{1}{NC} \tag{11}$$

Schölkopf [6] showed that this is an upper bound for the fraction of objects outside the description. The exact influence of s and ν (or C) on the SVDD is investigated in the next section.

3 Optimization of s and ν

When the user specifies beforehand a fraction of the target objects which can be rejected by the description, just one of the parameters s or ν can be determined. Unfortunately, both the values for s and ν have large influence on the final solution of the SVDD.

In figure 2 the decision boundaries of the SVDD for different choices of s and ν are shown. The figure shows that for smaller s, small details in the data are followed. On the other hand, when s is very large (in the order of the size of the complete dataset), only a spherical solution is found (for $s \to \infty$ the rigid hypersphere solution is obtained [9]). For $\nu = 0$ ($C = \infty$) no target objects are allowed outside the description. All objects (including the somewhat remote object at (10,0)) are accepted. When $\nu = \frac{1}{4}$ about one quarter of the target data can be outside the description. When enough data is available, a more robust estimate of the boundary is obtained.

For this 2-dimensional dataset it is possible to judge the boundary visually. For data in more than 3 dimensions this is impossible. In these cases other measures have to be invented to judge the quality of the solution.

For a good data description, two requirements have to be fullfilled: (1) a low target rejection rate and (2) a low outlier acceptance rate. When we are given only examples of the target set, the first term can be estimated by the number of support vectors that we obtain in the minimization of Lagrangian (6). It appears that the fraction of objects what become support vector is a leave-one-out estimate of the error on the target set:

$$E\left[P(\text{error target set})\right] = \frac{\#SV}{N} \tag{12}$$

Minimizing just the error on the target set is not sufficient. The optimal solution then would be to accept the complete feature space. To estimate the outlier acceptance rate without example outliers, we have to assume an outlier distribution. When we assume that the outliers are uniformly distributed in the part of the feature space we are interested in, we should minimize the volume occupied by the description. To minimize this, we have to estimate the volume of the description. This can be done by creating a large set of outlier objects around the target dataset. Unfortunately, this is very expensive in high dimensional feature spaces. It would be very convenient when the data description itself would offer an indication of the volume of the description.

When the Gaussian kernel is used (definition (8)), each object just has a limited support in the feature space. This is visible in the evaluation function (10), where the interaction between test object $\mathbf{z}$ and training objects $\mathbf{x}_i$ is only via the kernel (8). By this kernel, the influence of a particular support vector $\mathbf{x}_i$ is limited by a hypersphere with radius of (about) s in the feature space.

An heuristic approach for the SVDD with a Gaussian kernel would be to use the number of support vectors on the boundary. We then assume that each support object on the boundary accounts for a volume s^d in feature space. To be able to compare this with the error on the target set (which is a fraction of the target set which is rejected), we divide this with an estimate of the complete outlier volume $s_{\max}^d$ where $s_{\max}$ is the maximum distance present in the training set.

We now have an heuristic error to minimize:

$$\Lambda(s, \nu) = \frac{\#SV}{N} + \lambda \#SV_b \left(\frac{s}{s_{\max}}\right)^d \tag{13}$$

where λ regulates the trade-off between the error on the target data and on the outlier data. $\#SV$ indicates the total number of support vectors (both on the boundary as outside the description), $\#SV_b$ is the number of support vectors exactly *on* the boundary (i.e. $0 < \alpha_i < C$). In the

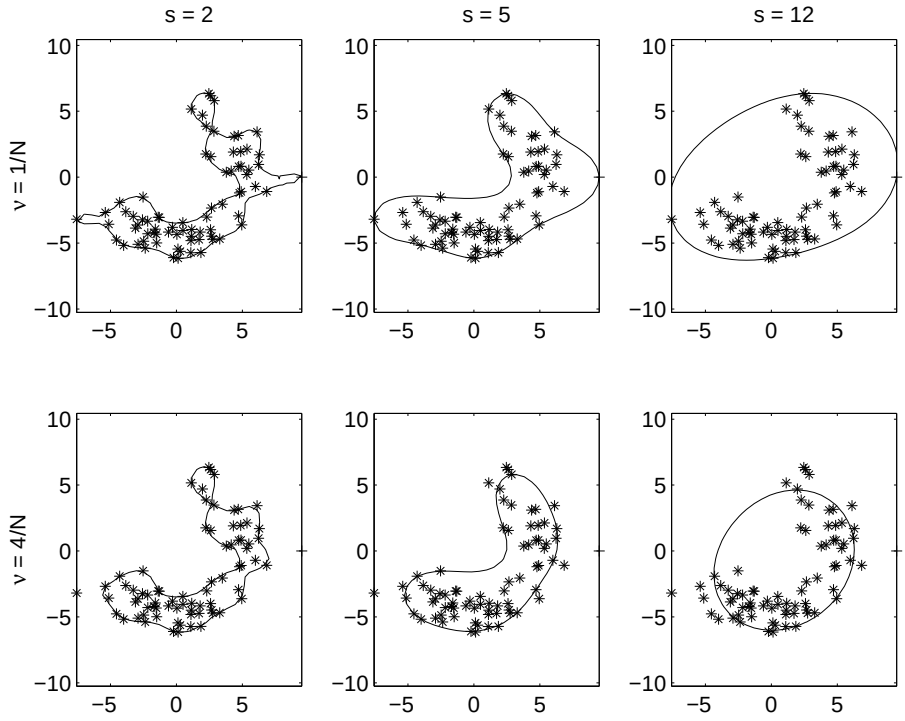

Figure 2: Decision boundaries of a simple 2-dimensional dataset, with different choices of s and ν.

experiments we will consider the variables

$$f_T = \frac{\#SV_b}{N} \tag{14}$$

and ν, defined by (11) (now both f_T and ν are parameters between 0 and 1).

This estimate is very rough. It disregards the fact that several support vectors on the boundary may support the same part of the data description. The effective volume $(\frac{s}{s_{\max}})^d$ per boundary support vector might, therefore, be substantially smaller. In these cases the trade-off parameter λ should be adapted, or the method will minimize the number of objects at the boundary further than is desirable. The advantage is, that no example outliers are required in the optimization.

For the data shown in figure 2 the s and ν are found by minimizing error (13). For the minimal error $s = 5.8, \nu = 0.16$, and the resulting data description is shown in figure 3. Although it does not detect the banana shape of the dataset, it rejects the remote outliers. So with an assumption on the volume of the data description (that it can be estimated by the number of support vectors on the boundary) we replaced the variables s and ν by the new variable λ. In the experiments we will investigate how well these assumptions hold.

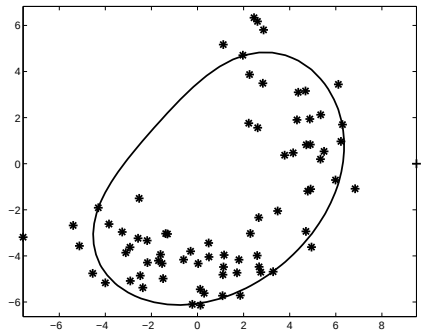

Figure 3: Solution of the data description for the dataset given in figure 2 and $\lambda = 1$.

4 Experiments

In this section we want to investigate if the optimization of (13) results in reasonable values of s and ν and thus reasonable data descriptions.

4.1 Handwritten digits

In the first experiment, we trained a SVDD on a handwritten digits dataset (the Concordia dataset, already used in [2]). This dataset contains handwritten digits (size 32×32), with 400 objects per class in a training set and 200 objects per class in a testing set. The data dimensionality was reduced from 1024 to about 30 using PCA, retaining 75% of the variance in the data. A SVDD

was trained on the class representing the threes. The s and ν were optimized simultaneously by minimizing (13).

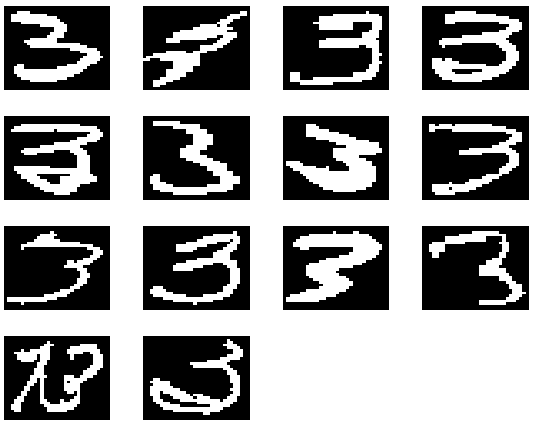

Figure 4: Testing target objects rejected by the data description.

In figure 4 the objects are shown, which are rejected by the SVDD. Surprisingly, some pretty reasonable threes are rejected (the upper left three, for instance), but also a real segmentation error was detected (the thirteen in the lower left). Note that only information about the target class is used, and that no extra preprocessing was done (except for using 75% PCA). These results show that reasonable results can be obtained with this automatic optimization, but it is hard to judge whether these results are good, because here no ground truth is available (i.e. an object is a genuine three, or an outlier three). In the next section we investigate an example where a ground truth is available.

4.2 Texture segmentation

In this application we try to distinguish between two types of textures. These textures are taken from [3], and we considered textures 1, 3 and 4 (texture 2 appeared to be very simple to separate, this is not shown here). We know beforehand the ground truth of both textures. In figure 5 the ground truth and the three texture images are shown. To classify a pixel an image patch containing $16 \times 16 = 256$ pixels is drawn around this pixel. The texture is therefore characterized by a 256-dimensional feature vector.

In the first experiment, no feature reduction is performed and the SVDD is trained on a training set, which is sampled from the complete target region (i.e. the dark region in the upper left picture of figure 5). Four hundred objects are used for training. When a patch in the neighborhood of the boundary of the target class is used, a part of

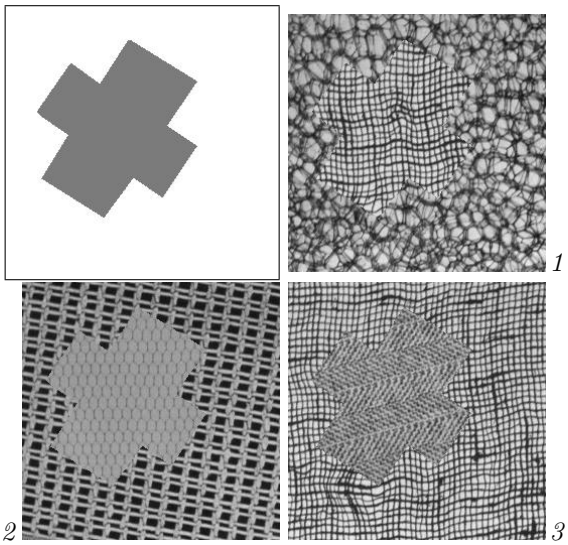

Figure 5: Texture segmentation. The texture in the cross shaped region in the middle is used as target class (upper left figure). The next three images are examples of textures used.

the outlier texture can be included in the patch. In that case the objects is not a pure target object and can be considered an outlier object for the target class. First we consider the third image (lower right image in figure 5) because this problem is neither very simple, not extremely hard. Next we will discuss the other two problems.

In table 1 the results are shown for the third image. We compare the performances of the methods by directly minimizing Λ (in the first two lines) and by varying f_T and ν by hand (last six lines). The third column shows the minimal error Λ^* (13), where numbers between brackets show the standard deviation over five runs. The fourth and fifth column give the error on the target class $\mathcal{E}_{\text{tar}}$ and the error on the outlier class $\mathcal{E}_{\text{out}}$ respectively (the percentage of pixels which is erroneously classified). This shows which trade-off between the target and outlier errors is made. The last column gives the total error $\mathcal{E}_{\text{tot}}$, the percentage of pixels that is erroneously classified.

In the comparison between the total error $\mathcal{E}_{\text{tot}}$ of the hand-picked s and ν and that of the automatic optimization shows that the automatic optimization comes close to the best hand-picked configuration (although it is not completely optimal). The heuristic error has a strong correlation with the true total error.

It appears that the minimization of Λ is relatively insensitive for the value of λ. For $\lambda = 1$, a comparable solution as for $\lambda = 1000$ is obtained. In the optimization of Λ, solutions with low num-

Table 1: Performances on image *3* using a trainingset of 400 patches (size 16×16). All 256 features are used. Results are averaged over five runs and multiplied by 100.

f_T	ν	Λ^*	$\mathcal{E}_{\text{tar}}$	$\mathcal{E}_{\text{out}}$	$\mathcal{E}_{\text{tot}}$
Well-sampled data, image *3*					
$\lambda = 1$		8.05 (1.43)	6.45 (1.78)	0.67 (0.41)	2.47 (0.36)
$\lambda = 1000$		5.80 (1.18)	6.41 (2.08)	0.70 (0.41)	2.48 (0.46)
0.00	0.00	5.40 (1.04)	6.28 (0.95)	0.64 (0.20)	2.40 (0.20)
0.00	0.10	11.40 (0.52)	12.06 (1.63)	0.10 (0.04)	3.82 (0.52)
0.00	0.20	21.15 (0.29)	19.67 (3.20)	0.00 (0.00)	6.12 (1.00)
0.10	0.00	6.25 (1.26)	7.67 (1.23)	0.45 (0.12)	2.70 (0.32)
0.10	0.10	11.55 (0.57)	10.93 (1.19)	0.18 (0.09)	3.52 (0.33)
0.10	0.20	21.20 (0.33)	21.63 (2.35)	0.00 (0.00)	6.73 (0.73)

ber of objects outside the boundary are favored (ν is small). The results for $\lambda = 1$ or $\lambda = 1000$ are comparable with the results of $f_T = 0, \nu = 0$ or $f_T = 0.1, \nu = 0$. The low value for ν indicates that the number of outlier objects in the data is low, and that we can use the dataset with confidence.

In table 2 the dimensionality of the objects was reduced from 256 to 67 or to 35, retaining about 90% or 75% of the variance in the training set. Although it might be expected that by reducing the dimensionality, the boundary of the target class can be determined better (using a limited training sample), the classification performance actually did not improve significantly. It appears that extra overlap between the classes is introduced. This is visible in the second experiment where 75% of the variance is retained. When 0.66% of the target class is rejected, already 20.2% of the outlier data is accepted.

The minimization of the heuristic error again results in almost the same solution as in the minimization of the true total error, although for the 75%-case a better correlation than for the 90% case is obtained. The results show that the heuristic error Λ has a preference for low values of ν. The minimal values of Λ^* (third column) often occur for $\nu = 0$. This shows that the first term in error (13) is weighted harder than the second term. Only for lower dimensionalities, larger values of ν are preferred. Also note, that when f_T and ν are optimized, large variation in Λ^* appears (for instance, $f_T = 0, \nu = 0.1$ at 75% PCA, $\Lambda^* = 77 \pm 143$!).

In table 3 the same experiment as in 1 is shown, except that in this experiment the training data is drawn from the boundary of the dataset. It means that all training objects, all the 16×16 image patches, somewhere touch or cross the boundary between the two classes. About half of the data is actually an outlier object (its true label is 'outlier'). The total error $\mathcal{E}_{\text{tot}}$ dramatically increases (from 3% error to about 30%), but the error on the target set $\mathcal{E}_{\text{tar}}$ actually decreases. By using the boundary objects, a much broader data description is obtained. The error on the target set completely vanishes, but a large fraction of the outlier data is accepted (and thus the error on the outlier data $\mathcal{E}_{\text{out}}$ is large). These results show, that when only boundary objects are available, it is still possible to find a one-class classifier but that it is not wise to use the heuristic error to optimize the parameters f_T and ν.

Similar results can also be observed for the second image (not shown here). The results on image *1*, are completely different though. The results on the first image show very poor performance (not shown in a table). It appears that when about 13% of the target class is rejected, 95% of the outlier class is accepted(!). Both the minimization of Λ as the hand-optimized f_T and ν totally fail. The fact that such a high fraction of the outlier obects is accepted, shows there is large overlap between the two classes. It appears that the target class is distributed over a wide range in the feature space, while the outlier class is much tighter clustered.

In figure 6 the resulting segmentations on the three images is shown. The upper left picture shows the output of the SVDD trained on the target class of image *1*. It suggest that the outlier class is even better characterized than the target class. The upper right shows the output on the SVDD trained on the outlier data of image *1*, and reasonable results are obtained. This result confirms the idea that the target data is at the extremities of the feature space, while the outlier data is clustered at one point (almost). Finally, the lower two images are the outputs for image *2* and *3*, and here good classification results are obtained.

To investigate the quality of the results ob-

Table 2: Performances on image *3* using a trainingset of 400 patches (size 16×16). The number of features was reduced by PCA. Results are averaged over five runs and multiplied by 100.

f_T	ν	Λ^*	$\mathcal{E}_{\text{tar}}$	$\mathcal{E}_{\text{out}}$	$\mathcal{E}_{\text{tot}}$
PCA to 90% of variance, image *3*					
$\lambda = 1$		7.70 (2.92)	2.46 (1.76)	4.60 (3.20)	3.94 (1.69)
$\lambda = 1000$		7.00 (1.14)	2.19 (0.52)	3.91 (1.01)	3.38 (0.56)
0.00	0.00	9.89 (0.82)	1.30 (0.79)	5.35 (0.82)	4.09 (0.46)
0.00	0.10	11.23 (3.30)	4.88 (1.54)	1.78 (0.75)	2.75 (0.16)
0.00	0.20	21.54 (1.60)	11.50 (1.60)	0.40 (0.19)	3.85 (0.43)
0.10	0.00	10.76 (6.54)	0.93 (0.36)	6.49 (2.27)	4.76 (1.50)
0.10	0.10	11.61 (0.28)	4.82 (1.64)	1.83 (0.71)	2.76 (0.08)
0.10	0.20	21.40 (0.34)	13.32 (1.56)	0.13 (0.06)	4.24 (0.46)
PCA to 75% of variance, image *3*					
$\lambda = 1$		23.90 (4.75)	12.52 (4.26)	1.86 (1.05)	5.18 (0.79)
$\lambda = 1000$		21.68 (8.94)	11.73 (5.98)	4.03 (5.80)	6.43 (2.21)
0.00	0.00	21.97 (23.49)	0.66 (0.31)	20.15 (3.91)	14.09 (2.60)
0.00	0.10	77.42 (143.65)	3.39 (1.00)	7.12 (1.61)	5.96 (0.97)
0.00	0.20	21.18 (0.45)	11.23 (1.32)	2.64 (0.77)	5.31 (0.35)
0.10	0.00	8.65 (4.36)	0.70 (0.49)	20.94 (5.15)	14.65 (3.41)
0.10	0.10	12.28 (1.63)	3.89 (0.49)	6.67 (0.71)	5.81 (0.46)
0.10	0.20	22.02 (1.61)	10.76 (1.79)	2.88 (0.81)	5.33 (0.26)

Table 3: Performances on first image using 400 training objects drawn from the boundary of the object. The complete 256-dimensional feature space is used.

f_T	ν	Λ^*	$\mathcal{E}_{\text{tar}}$	$\mathcal{E}_{\text{out}}$	$\mathcal{E}_{\text{tot}}$
Boundary data, image *3*					
$\lambda = 1$		8.40 (1.13)	0.00 (0.00)	58.97 (2.83)	40.63 (1.95)
$\lambda = 1000$		8.36 (1.42)	0.00 (0.00)	56.64 (3.44)	39.03 (2.37)
0.00	0.00	8.15 (1.72)	0.00 (0.00)	58.40 (4.06)	40.24 (2.80)
0.00	0.10	13.10 (0.52)	0.00 (0.00)	45.64 (2.83)	31.45 (1.95)
0.00	0.20	22.45 (0.76)	0.07 (0.09)	30.78 (1.13)	21.23 (0.78)
0.00	0.30	32.00 (0.68)	0.33 (0.29)	20.84 (2.53)	14.46 (1.71)
0.10	0.00	7.10 (0.88)	0.00 (0.00)	59.87 (3.74)	41.25 (2.57)

tained by the SVDD, we compared it with a simple unsupervised clustering on features obtained by Gabor filtering [3]. The images were filtered using twelve Gabor filters. For each filter an energy was computed and thus each pixel was characterized by twelve measurements. Pixels from the complete image were clustered in two classes using unsupervised k-means clustering. In figure 7 the labeling results are shown. Except for the first image, the SVDD (trained on just the target set) shows comparable or better results than the k-means clustering.

5 Conclusion

The Support Vector Data Description is made to separate one class of data from the rest of the feature space. In the basic form, the user has to supply two parameters, indicating the important scale in the data and the fraction of objects which is expected to be outlier, or which are not representative for the description of the data. In this paper we investigated if it would be possible to automate the optimization of the two parameters *without* the use of example outlier objects. For this the heuristic error criterion (13) is defined.

The heuristic error works when a reasonable training distribution is used and the fraction of outlier objects is not too large. When the fraction of outliers, or what appears to be outlier, is too large, this will be considered target data. Without further knowledge, this is therefore just accepted as target data. The results are also very poor, when the target data is scattered over the

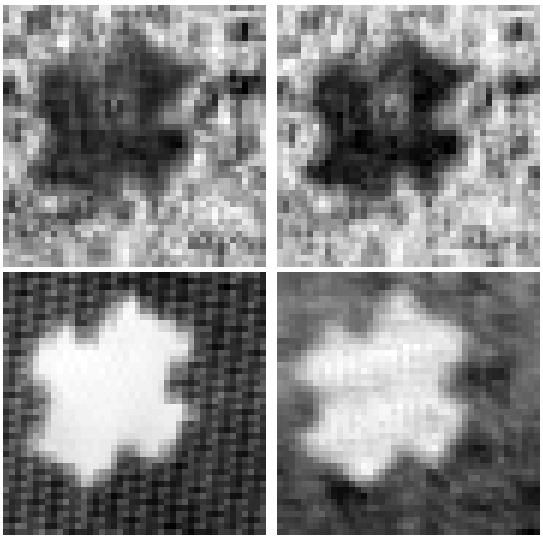

Figure 6: Output of SVDD with optimized s and ν for (from upper left to lower right) image *1*, image *1* trained on the outlier class, image *2* and image *3*. A light grayvalue indicates that the pixel is near the center $\mathbf{a}$ of the SVDD hypersphere (and thus classified as target object by the SVDD).

complete feature space and is not clustered. This is not a failure of the heuristic error, but because the data does not fit the basic assumption by the SVDD (namely that similar objects are near in the feature space). We can therefore conclude, that when relatively good target data is present (representative training set with a small fraction of outlier objects) this heuristic error gives a good data description, without the optimization of free parameters.

6 Acknowledgments

This work was partly supported by the Foundation for Applied Sciences (STW) and the Dutch Organization for Scientific Research (NWO).

References

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1995.

[2] Sung-Bae Cho. Recognition of unconstrained handwritten numerals by doubly self-organizing neural network. In *ICPR*, 1996.

[3] D. de Ridder, J. Kittler, O. Lemmers, and R.P.W. Duin. The adaptive subspace map for texture segmentation. In A. Sanfeliu, J.J. Villanueva, M. Vanrell, R. Alquzar, J.-O. Eklundh, and Y. Aloimonos, editors, *proceedings of the 15th IAPR international conference on pattern recognition*, pages 216–220. IAPR, 2000.

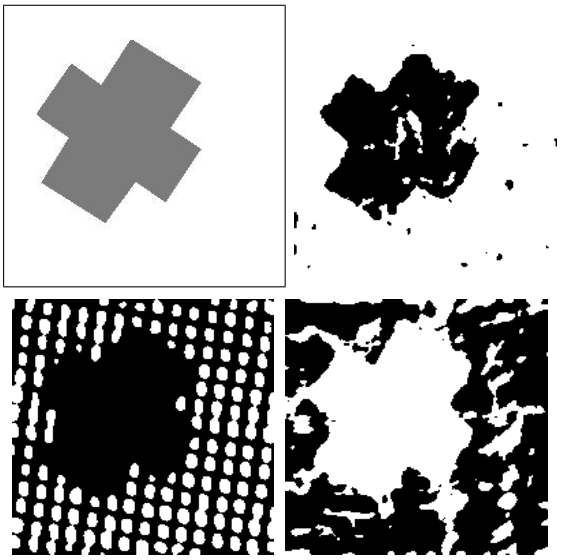

Figure 7: Texture segmentation by using k-means clustering on twelve Gabor filters.

[4] N. Japkowicz. *Concept-Learning in the absence of counter-examples: an autoassociation-based approach to classification*. PhD thesis, New Brunswick Rutgers, The State University of New Jersey, 1999.

[5] M.R. Moya, M.W. Koch, and L.D. Hostetler. One-class classifier networks for target recognition applications. In *Proceedings world congress on neural networks*, pages 797–801, Portland, OR, 1993. International Neural Network Society, INNS.

[6] B. Schölkopf, P. Bartlett, A.J. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, Advances in Neural, 1999.

[7] D.M.J. Tax, D. de Ridder, and R.P.W. Duin. Support vector classifiers: a first look. In *Proceedings ASCI'97*. ASCI, 1997.

[8] D.M.J. Tax and R.P.W Duin. Outlier detection using classifier instability. In Amin, A., Dori, D., Pudil, P., and Freeman, H., editors, *Advances in Pattern Recognition, Lecture notes in Computer Science*, volume 1451, pages 593–601, Berlin, August 1998. Proc. Joint IAPR Int. Workshops SSPR'98 and SPR'98 Sydney, Australia, Springer.

[9] D.M.J. Tax and R.P.W Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, December 1999.

[10] V.N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.