Classifiers in Almost Empty Spaces

Robert P.W. Duin
Pattern Recognition Group, Department of Applied Physics
Delft University of Technology, The Netherlands
duin@ph.tn.tudelft.nl

Abstract

Recent developments in defining and training statistical classifiers make it possible to build reliable classifiers in very small sample size problems. Using these techniques advanced problems may be tackled, such as pixel based image recognition and dissimilarity based object classification. It will be explained and illustrated how recognition systems based on support vector machines and subspace classifiers circumvent the curse of dimensionality, and even may find nonlinear decision boundaries for small training sets represented in Hilbert space.

1. Introduction

The way of thinking, within the pattern recognition community, on the sample-size / feature-size problem has been dramatically changed during the last decade. Traditionally it was thought that, due to the curse of dimensionality (Rao [1]) it is necessary to 'fill' the feature space with more objects than its dimensionality in order to obtain a classifier which generalizes well. Recently it has become clear, however, that there are several ways to construct good classifiers in almost empty spaces: for small datasets in very high-dimensional spaces, or even in Hilbert space. In this paper we will try to give some understanding on how this is achieved and present some examples. We will argue that the curse of dimensionality still exists, but that there are now ways to avoid it. Moreover, it will be pointed out how it can be distinguished from the related effects: the peaking phenomenon and overtraining.

For many traditional classifiers, trained by m objects, the generalization error $\varepsilon(k)$ shows, for increasing feature sizes k, a minimum at about $k = m/\alpha$ (the peaking phenomenon, see Jain et al. [3]). The generalization error is usually estimated by an independent test set. In fig. 1 an example is shown based on Fisher's Linear Discriminant (FLD). The data set used here consists of 2000 samples for each of the characters '3' and '8' in one of the NIST databases. They are, after a normalizing preprocessing stage, represented in 16x16 grey value images, see also [16]. A random subset of the data is used for training (equally sized parts for both classes) and the remaining set is used for testing. Results averaged over 50 experiments are shown. Unless mentioned otherwise, all other figures, shown below, are based on the same dataset, using a similar procedure.

Usually, optimal values for α are found between 2 and 10. In a feature space with $k = m$ all m training objects lie in

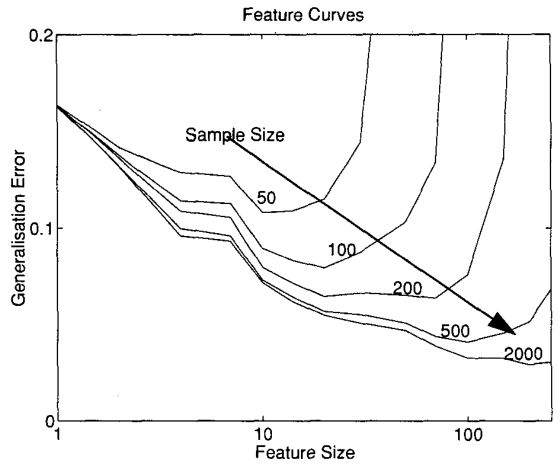

Fig. 1. Peaking of the generalization error of FLD as a function of the feature size for various sample sizes.

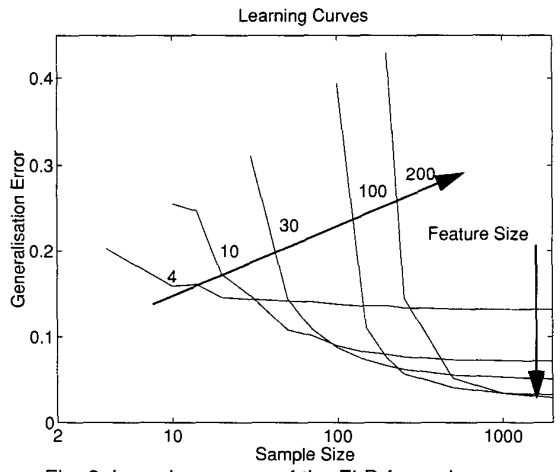

Fig. 2. Learning curves of the FLD for various feature sizes.

a m-1-dimensional linear hyperplane. A linear classifier can be found that correctly classifies all training objects, even if they are randomly labeled, see Cover [2]. As a consequence, it was believed that no guarantee can be found that classifiers optimized by $m < k$ samples have any generalization capability. Therefore the "small sample size problem" ($m \downarrow k$) should be treated very carefully, Raudys and Jain [4]. In fig. 2 some learning curves for the FLD are given. This classifier is not defined for $m \leq k$, as the sample scatter matrix becomes singular and cannot be inverted. Consequently, for low sample sizes low feature sizes are needed and for large sample sizes larger feature sizes are possible and asymptotically yield better performances.

This state of affairs, a deteriorating performance by an increasing feature size, is as a rule counterintuitive. It does not allow the use of image objects represented by all, e.g. 256^2 pixels as features, as there is no way to collect a dataset of sufficient size. Still, there are applications for which this representation works, e.g. a simple classifier like the nearest mean method for recognizing character with a fixed font. Applied on images, this method is also called template matching.

In this paper classifier modifications are discussed that generalize in high-dimensional feature spaces, even if they are *almost empty*, i.e. filled with a training set that is (much) smaller than the dimensionality of the space. These classifiers may be described as, possibly nonlinear, functions of (dis)similarities, correlations or distances between the templates and the object to be classified. This might be an attractive solution if the number of templates necessary for a good performance is small. Our main theme is to give some understanding how this works in a feature space representation.

In the next section we will present the use of training sets, representation sets and kernel mappings for handling large feature sizes and for constructing nonlinear classifiers. First we will show how the FLD can be used to that purpose. Next we will discuss the support vector classifier, the subspace classifier and dissimilarity based classifiers.

2. Representation sets and kernel mapping

Throughout the paper we will use the fact that any k-dimensional feature space R_k can be mapped on an n-dimensional space using a *representation set* S and a kernel function $K(S,\boldsymbol{x})$. The set $S = \{ \boldsymbol{x}_1^S, \boldsymbol{x}_2^S, ..., \boldsymbol{x}_n^S \}$ contains n vectors which represents in one way or another the dataset of interest. The kernel function computes a similarity or dissimilarity between an arbitrary feature vector $\boldsymbol{x} \in R_k$ and each of the n vectors in the representation set. The output vector $\boldsymbol{u}$ then contains n components. So by kernel mapping $\boldsymbol{u} = K(S,\boldsymbol{x})$ an arbitrary object $\boldsymbol{x} \in R_k$ is mapped on an object $\boldsymbol{u} \in R_n$ in the kernel space, i.e. the output space of the kernel function for a given representation set. Typical choices for the kernel function $K(\boldsymbol{y}, \boldsymbol{x})$ are:

$$\text{polynomials: } K(\boldsymbol{y}, \boldsymbol{x}) = (\boldsymbol{x} \bullet \boldsymbol{y} + 1)^p \tag{1}$$

$$\text{Gaussians: } K(\boldsymbol{y}, \boldsymbol{x}) = \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right) \tag{2}$$

Note that the polynomial kernel is linear for $p = 1$. Although $\boldsymbol{y}$ always has k components, it may be embedded in a linear subspace of dimensionality $k' < k$ if the intrinsic dimensionality of S is k'. The polynomial kernel increases the dimensionality of a subspace to at most $\binom{k'+1+p}{p}$. The dimensionality of the kernel space for the Gaussian kernel, however, is unbounded for increasing n. One may thereby state that such a kernel maps the vectors into Hilbert space. For introduction to the use of kernel mapping techniques, see [5], [12] and [13].

The interesting application of kernel mapping is that it facilitates the computation of nonlinear classifiers in the feature space by constructing a linear classifier in the kernel space. We will restrict ourselves here to two-class problems with class labels $\lambda = \pm 1$:

$$C(\boldsymbol{x}) = \boldsymbol{w} \bullet \boldsymbol{u} + w_0 = \boldsymbol{w} \bullet K(S, \boldsymbol{x}) + w_0$$

$$C(\boldsymbol{x}) < 0 \dashrightarrow \hat{\lambda} = -1 \text{ else } \hat{\lambda} = 1 . \tag{3}$$

This classifier always has, independent of the nonlinearity, $n+1$ free parameters, stored in $[\boldsymbol{w},w_0]$. A training set $L = \{ \boldsymbol{x}_1^L, \boldsymbol{x}_2^L, ..., \boldsymbol{x}_m^L \}$ with given labels $\Lambda = \{\lambda_1, \lambda_2, ... ,\lambda_m\}$, in which $\lambda_i = \pm 1$, can be used for estimating $\boldsymbol{w}$ after it is mapped in the kernel space. The representation set S may or may not be a subset of the training set L. In some procedures they coincide. In other procedures a subset of the original training set is used for representation and the remaining part, or all, is used for training.

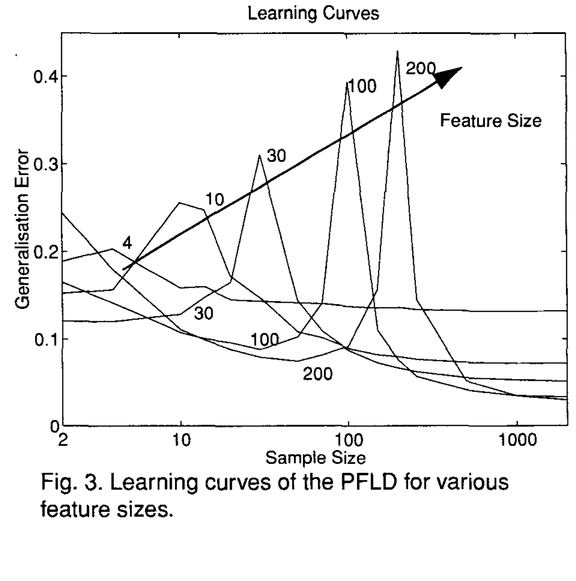

Fig. 3. Learning curves of the PFLD for various feature sizes.

If the linear kernel is used (polynome with $p = 1$) and $S = L$, so $n = m$, effectively a linear classifier in R_k is computed even if $n < k$ using thereby a training set with $m < k$ objects. A straightforward (but not optimal) way to do that is the Pseudo-Fisher Linear Discriminant. It is discussed in section 3 to gain a better understanding of the feature-size / sample-size problem.

3. The Pseudo-Fisher Linear Discriminant

If $m = n$ (e.g. if $S = L$), an exact solution is found for $\boldsymbol{w}$ in (3), provided that no objects are dependent. It is defined by

$$\lambda_i C(\boldsymbol{x}_i) = 1, \forall x_i \in S \qquad (4)$$

For a linear kernel this is a classifier $C(\boldsymbol{x})$ perpendicular to the hyperplane defined by the training set. The weight vector $\boldsymbol{w}$ is located in this hyperplane. All objects have the same distance to the decision boundary, since $C(\boldsymbol{x}) = \pm 1$ $\forall \boldsymbol{x} \in S$. Elsewhere ([14], [18]) we named this the Pseudo-Fisher Linear Discriminant (PFLD), as it is equivalent to the use of pseudo-inverse for the singular scatter matrix. In fig. 3 the surprising learning curves for this classifier are shown, theoretically explained in [18]. As predicted, generalization is bad when the sample size equals the feature size. For *smaller* sample sizes, however, a *better generalization* appears to be possible. The best results, however, are always found for an asymptotically increasing samples size. The overall results for the (P)FLD are shown in a sample-size-feature-size diagram, see fig. 5.

The PFLD is of theoretical interest as it is identical to the FLD for sample size larger than the dimensionality and as it clearly shows the generalization result of a complete adaptation of the classifier to the data below that point. It has the same behavior as the regularized FLD with an asymptotically small regularization term [22].

The interesting area for this paper is the bottom-right part of fig. 5, for which the sample size is smaller than the feature size (the dimensionality), so $m < k$. It appears that performances can be found for low sample sizes and high

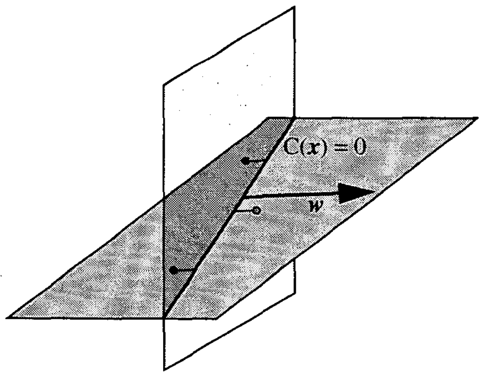

Fig. 4. The Pseudo-Fisher Linear Discriminant as it is constructed in the subspace defined by the training data.

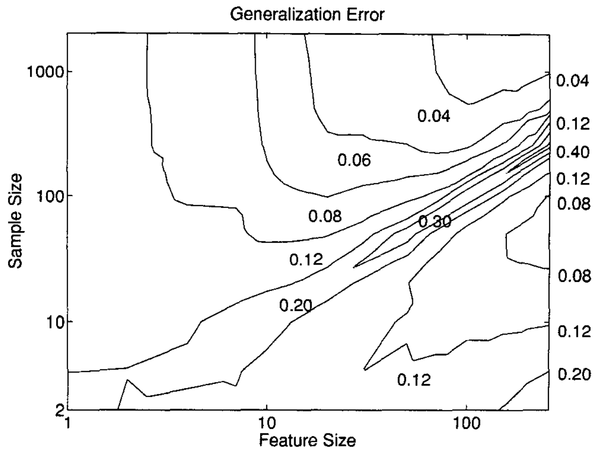

Fig. 5. The generalization error of the PFLD as a function of feature size and sample size.

feature sizes (deep in the region doomed by the curse of dimensionality) that are much better than obtained for 'sound' feature sizes. For larger feature sizes, much higher sample sizes are necessary. This is illustrated by the feature curves in fig. 6. These show that after the peaking phenomenon, the deterioration for increasing feature size, the error rates shows a maximum, followed by another area of good performances, with even lower error rates than the initial ones.

The above shows that generalizing classifiers may be found in almost empty spaces. The learning curves of the PFLD follow the trend as presented in fig. 3. First performance increases when the sample size grows. Then the space becomes more and more filled, in the sense that the set of objects shows an increasing dependency (i.e. objects can with an increasing accuracy be described as linear combinations of other objects), either caused by the finite feature set, or by the intrinsic dimensionality of the data. In this region it is still possible, however, to construct a classifier that perfectly separates all training objects. When the sample size approaches the dimensionality the classifier becomes overtrained as it still absorbs all noise, but hardly gains any new separating possibilities. After the sample size passes the dimensionality perfect solutions on the training set are not possible anymore. It is not possible anymore to absorb all noise, instead, it is gradually 'averaged out' using the least square criterion used for finding the coefficients.

It can be concluded that the generalization is determined by two clearly separated processes. As a function of the sample size, first an increasing structural adaptation is made, perfectly fitting the classifier to the training set. This is followed by a statistical adaptation in which a set of parameters of constant size is asymptotically better estimated. In between there is a 'dimensionality resonance', a region in which the structure is too complex (contains too

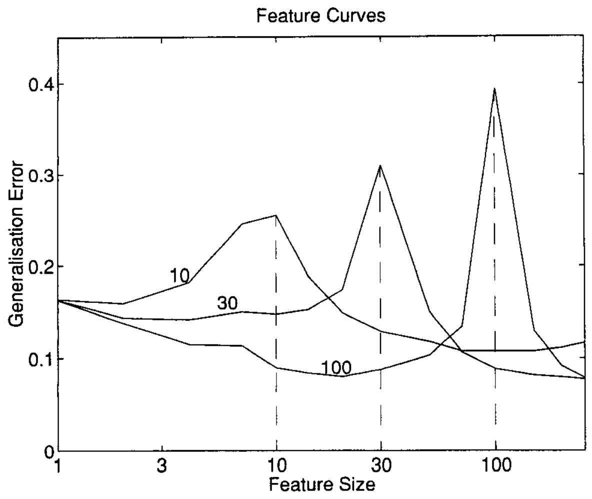

Fig. 6. The generalization error of the PFLD as a function of the feature size *k* for sample sizes of *m* = 10, 30 and 100. It appears that error (*m*=30, *k*=256) < error(*m*=30, *k*<30).

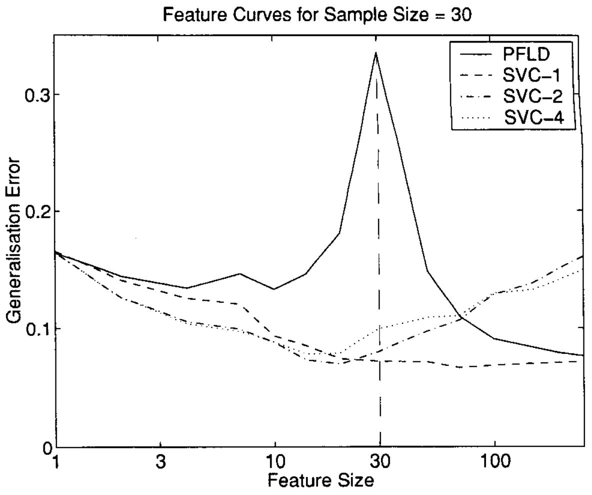

Fig. 7. The generalization errors of the PFLD and the SVC as a function of the feature size for a sample size of 30. In the SVC polynomial kernels are used of the orders 1,2 and 4.

many parameters) and the statistical adaptation is too small (too less noise averaging) for a good performance. The maximum is where the sample size equals the dimensionality is in fact determined by the number of free parameters in the classifier. Note that in the feature curve plots (for constant sample size), see fig. 6, the same phenomenon but in reversed order is visible as in the learning curves, see fig. 3. First there is a statistical generalization, until the feature size equals the sample size, then, right of the maximum error, better results may be found, due to the structural adaptation. In [23] it is shown that this can even be reached by the addition of redundant (random) features!

The first maximum in the feature curve, where the feature size equals the sample size, is caused by the curse of dimensionality. Here the increasing dimensionality causes a problem as it is monotonically related to the number of free parameters. After this peak, the number of free parameters is fixed at the sample size. Where for very high feature sizes the error increases again, e.g. for m = 10 in fig. 6, this is caused by the increasing noise and has nothing to do which the dimensionality as such.

For sample sizes in the order of the dimensionality, the paradoxical situation exists that the expected performance can be improved by deleting a random subset of the samples. This points to the possibility that better classifiers may be found by first reducing the complexity of the structural adaptation. Next, some heuristic combination of the two criteria (complexity constraining and error minimization) may improve the generalization further.

The next sections describe examples of such classifiers.

4. Support Vector Classifiers

The Support Vector Classifier (SVC) offers a clear and systematic procedure for reducing the number of objects in the training set that are used for defining the classifier (3). This is done by optimizing the criterion:

$$\min_w(\boldsymbol{w} \bullet \boldsymbol{w}) \text{ while } \lambda_i C(x_i) \geq 1,\ \forall x_i \in S \qquad (5)$$

resulting in a quadratic optimization procedure (see [5], [6], [7]) that, as a side effect, maximizes the number of components $w_i = 0$ in $\boldsymbol{w}$. Consequently the number of objects x_i that participate in (3) is minimized. These objects are called support objects, as they are the only ones that take part in the definition of the classifier. Objects that are not needed, in the sense that they will be classified correctly anyhow, are ignored. This reduces the noise and thereby improves the generalization capability. The set of support objects can now be treated, in our terms, as the representation set *S*. As we are interested in classification problems in almost empty spaces, we restrict the treatment of the SVC to the situation of separable classes. The SVC has a number of very interesting properties:

- If the SVC fully correctly classifies all *m* samples in the training set and thereby needs *n* support objects, its estimated performance is bounded by $\frac{n}{2m}$. This directly follows from a leave-one-out observation: non-support objects are correctly classified if they are removed from the training set, while the support objects have at most a probability of 50% of being correctly classified after removal. So, the less support objects that are needed, the better the classifier (in expectation).

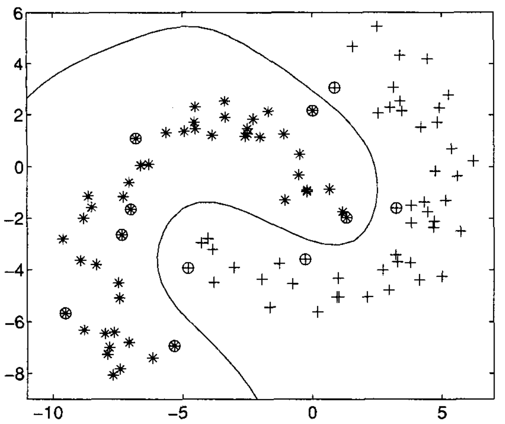

Fig. 8. Nonlinear example of a SVC based on a Gaussian kernel, separating two 2D classes. Support vectors are encircled.

- The SVC has no direct dimensionality problems, as it is entirely defined by the kernel values. If additional features add noise and no information, however, a deteriorating performance remains possible. In fig. 7 some feature curves are shown, illustrating this point as the performance of higher order polynomial kernels deteriorates. The number of support vectors needed in these experiments with in total 30 objects in the training set ranges from about 10 for the linear kernel to 30 for many of the classifiers based on the 4-th order polynomial kernel.
- Other, nonlinear kernels as discussed in section 2, may be used as well (provided that they fulfill Mercer's theorem [5]). If they need less support vectors, the expected performance is lower. Consequently, nonlinear classifiers, constructed in this way are not necessarily more complex, and do not need large training sets. A 2D-example is shown in fig. 8.

5. Subspace Methods

In the SVC the number of support vectors, and thereby the classifier complexity, is a result of complexity constraining and cannot be preset. Subspace methods constitute a way to control the complexity directly. Each class is approximated by subspace of the feature space using an eigenvector analysis. Either the number of eigenvalues, or the desired accuracy (explained variance) can be set. New objects are assigned to the class of the nearest subspace. See Oja [10], [8], [9]. Originally, subspaces are defined in the feature space. Here they will be defined more generally on the kernel space. The representation set S is split into c subsets $S = \{S_j, j=1, ..., c\}$ according to the c classes. Each of these subsets constitutes a subspace in the kernel space according to:

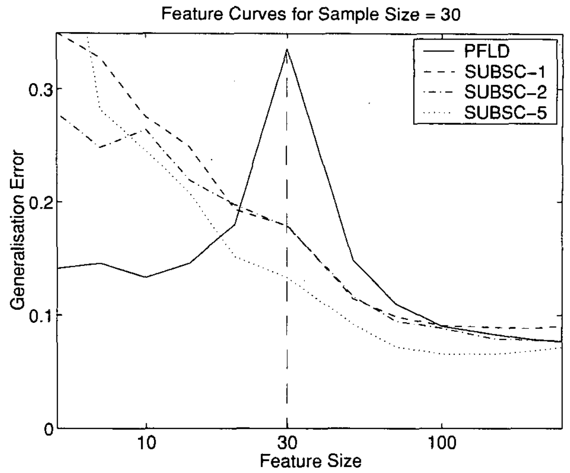

Fig. 9. The generalization errors of the PFLD and the subspace classifier (SUBSC) as a function of the feature size k for a sample size of m = 30. For SUBSC three subspace dimensionalities per class are used: 1, 2 and 5

$$R_j(x) = K(S_j, x) \tag{6}$$

in which x is an arbitrary object to be mapped. The dimensionality of R_j is equal to the number of vectors of class S_j. A classifier can now be defined by selecting the subspace that contains the largest projection of a new object:

$$\text{class}(x) = \text{argmax}_j\{R_j(x) \bullet R_j(x)\} \tag{7}$$

A better generalization may be obtained be reducing the dimensionalities of the subspaces, e.g. by an eigenvector analysis based on a principal component analysis. In relation with the use of kernels, this is sometimes called kernel-PCA [12]. One may choose for either, a fixed dimensionality (a preset number of eigenvectors) or for a fixed accuracy (selection of the minimum number of eigenvectors that explains more than a fraction α, e.g. 95%, of the variance). Some feature curves are presented in fig. 9 . They show that for low feature sizes the subspace method is not very useful, as it is difficult to characterize classes by subspaces. For high feature sizes, however, the method may have a performance that approaches the SVC, but has the possibility of a user defined complexity. Classification by 5-dimensional subspaces in a two-class problem needs the same amount of computations as a SVC based on 10 support objects.

6. Dissimilarity Based Object Representations

The previous sections show that the kernel representation is a powerful tool for building classifiers in high-dimensional spaces. It reduces the problem of m points in a k-dimensional space ($m < k$) to m points in an m-dimensional space. The SVC further reduces the set of m training objects to a smaller representation set by optimizing the set of support objects. The subspace method reduces the dimensionality below m, keeping the training set size at m.

Instead of constructing kernels in a feature space, describing object relations, such relations may be given externally and used as a starting point. In this way a distance matrix $K(Y, X)$ between two sets of objects X and Y may play a similar role as the kernel functions. E.g.:

$$K(y, x) = \|x - y\|^2 \tag{8}$$

In general, $K(S,x)$ is now the set of dissimilarities between an object x and the objects in the representation set S. The symbol x, however, may not refer anymore to just a point in a feature space but it may also point to the original object. Relational discriminant analysis based on dissimilarities now works as follows [19], [20], [21]:

1. Select some representation set.
2. Define some dissimilarity measure. This may be based, either on a feature representation, either on the raw data (e.g. images or contours).
3. Train a classifier on a training set represented by the dissimilarities to the representation set.
4. Use this classifier for new objects represented by their dissimilarities to the representation set.

A direct use of the SVC for reducing a training set to the representations set S, treating (dis)similarities as kernel outcomes, may often not be possible. Arbitrary similarity measures often do not fulfill Mercer's conditions [5], while for dissimilarity measures this is theoretically impossible. Therefore other techniques have to be used to reduce the size of the representation set [20]. In a given kernel space, however, arbitrary classifiers may be applied. In [21] it is argued that many distance measures are constructed by summations over differences in feature values or pixel values. As a consequence they are often approximately normally distributed by which Bayes normal classifiers perform well.

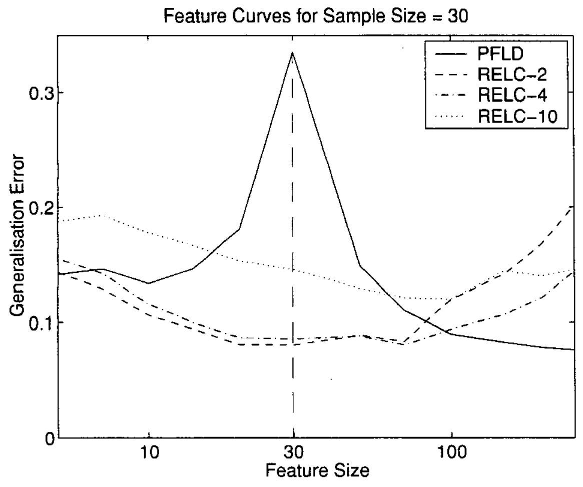

Fig. 10. The generalization errors of the PFLD and the linear relational classifier (RELC) as a function of the feature size k for a sample size of $m = 30$. For RELC three sizes of the representation set are used: 2, 4 and 10.

This approach may be useful for applications in which (dis)similarities between objects can be directly measured, but also in case no natural feature set can be found and instead a distance measure between the raw measurements is defined. Some initial studies on this featureless, relational representation have been presented by us [15], [17], [19], [20] and [21], showing that there are various ways to build classifiers for a dissimilarity based object representation. The example presented in fig. 10 is based on squared Euclidean distances (8). For the selection of the representation sets, having sizes of 2, 4 and 10 objects, some type of vector quantization [24] has been used. The remaining objects in the training set of size $m = 30$, were used for training the classifier, here the FLD. In studying fig. 10 in comparison with fig. 7 it can be noted that RELC-2 defined on 30 features (computational complexity of $O(2x30)$) performs almost as good as FLD for 256 features ($O(1x256)$) and just slightly worse than the linear SVC which needs on the average about 10 support vectors ($O(10x256)$). This illustrates the computational advantage of the relational classifier.

7. Discussion

The training of classifiers in almost empty feature spaces yields a problem if the classifiers are directly represented on the features. The number of free parameters will then increase by an increasing dimensionality. This results in a deteriorating performance, the 'curse of dimensionality', which is due to the mismatch of the training size and the number of free parameters. This can be solved by representing the objects on themselves, using kernels or dissimilarities. In a next step the dimensionality of this representation can be reduced further. The SVC finds a minimum support set that classifies the data correctly. By the subspace method linear subspaces in the kernel space are determined. The relational classifier uses all training objects for finding a classifier based on a small representation set. In all cases new objects are classified using a function of kernels or dissimilarities with the representation set. These three methods, although closely related, show large computational differences.

The SVC needs a computational extensive training procedure in optimizing the support set. Especially for training sets larger than about 1000 objects specific adapted procedures may be necessary to make the handling of 1000x1000 matrices feasible. As the size of the support set cannot be preset by the user the computational burden of the resulting classifier cannot be controlled.

The subspace method and the relational classifier are, both, fast and simple to use. Moreover, the complexity can be set by choosing the dimensionality of the representation. The drawback of this is that the complexity is not optimized automatically. In case of linear subspaces the resulting eigenvectors are weighted versions of all the original objects and may be stored as a new object. For kernel PCA (nonlinear subspaces) this is not possible. Consequently all objects in the representation set should be stored and are needed in each classification.

The additional advantages of these approaches are that nonlinear classifiers can be constructed by using nonlinear kernels or dissimilarity measures. As the number of parameters is defined by the size of the representation set, this increased discrimination potency is not affected by an increased noise sensitivity.

All methods described in this paper, can be used in many versions. Several choices can be made for object normalization, selection of representation sets, dimensionality reduction, etcetera. They constitute a large family that, by the nature of the object representation, is able to construct good performing classifiers in almost empty spaces.

8. Acknowledgement

The author gratefully acknowledges his colleagues Pavel Paclik, Elżbieta Pękalska, Dick de Ridder, Marina Skurichina, David Tax and Alexander Ypma for stimulating discussions and support.

9. References

[1] C.R. Rao, The utilization of multiple measurements in problems of biological classification (with discussion), *JRSSB*, vol. 10, 1948, 159-203.

[2] T.M. Cover, Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition, *IEEE Transactions on Electronic Computers*, vol. EC-14, June 1965, 326-334.

[3] A.K. Jain and B. Chandrasekaran, Dimensionality and Sample Size Considerations in Pattern Recognition Practice, in: P.R. Krishnaiah and L.N. Kanal (eds.), *Handbook of Statistics*, vol. 2, North-Holland, Amsterdam, 1987, 835 - 855.

[4] S.J. Raudys and A.K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252-264, 1991.

[5] V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, 1998.

[6] B. Schölkopf, *Support vector learning*, Oldenbourg Verlag, Munich, 1997.

[7] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 1997.

[8] M. Prakash and M.N. Murty, Growing subspace pattern recognition methods and their neural-network models, *IEEE Transactions on Neural Networks*, vol. 8, no. 1, 1997, 161-168.

[9] S. Watanabe and N. Pakvasa, Subspace method in pattern recognition, Proc. Int. Joint Conf. Pattern Recognition, 1973, pp.25-32

[10] E. Oja, *The Subspace Methods of Pattern Recognition*, Wiley, New York, 1984.

[11] A.K. Jain and D. Zongker, Representation and recognition of handwritten digits using deformable templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 12, 1997, 1386-1391.

[12] B. Schölkopf, A. Smola, and K.R. Muller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Computation*, vol. 10, no. 5, 1998, 1299-1319.

[13] N. Cristinanini and J. Shawe-Taylor, *Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, Cambridge, UK, 2000.

[14] R.P.W. Duin, Small sample size generalization, in: G. Borgefors (eds.), *SCIA'95, Proc. 9th Scandinavian Conf. on Image Analysis*, Vol. 2 (Uppsala, Sweden, June 6-9), 1995, 957-964.

[15] R.P.W. Duin, D. de Ridder, and D.M.J. Tax, Experiments with object based discriminant functions; a featureless approach to pattern recognition, *Pattern Recognition Letters*, vol. 18, no. 11-13, 1997, 1159-1166.

[16] R.P.W. Duin and D. de Ridder, Neural network experiences between perceptrons and support vectors (invited paper), in: A.F. Clark (eds.), *Proc. of the 8th British Machine Vision Conference*, volume 2 (Colchester, UK, Sep.8-11), University of Essex, Colchester, UK, 1997, 590-599.

[17] R.P.W. Duin, D. de Ridder, and D.M.J. Tax, Featureless Pattern Classification, *Kybernetika*, vol. 34, no. 4, 1998, 399-404.

[18] S. Raudys and R.P.W. Duin, On expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix, *Pattern Recognition Letters*, vol. 19, no. 5-6, 1998, 385-392.

[19] R.P.W. Duin, Relational Discriminant Analysis and Its Large Sample Size Problem, in: A.K. Jain, S. Venkatesh, B.C. Lovell (eds.), *ICPR'98, Proc. 14th Int. Conference on Pattern Recognition* (Brisbane, Aug. 16-20), IEEE Computer Society Press, Los Alamitos, 1998, 445-449.

[20] Robert P.W. Duin, Elżbieta Pękalska, and Dick de Ridder, Relational discriminant analysis, *Pattern Recognition Letters*, vol. 20, no. 11-13, 1999, 1175-1181.

[21] Elżbieta Pękalska, Robert P.W. Duin, Classifiers for dissimilarity-based pattern recognition, *ICPR2000, Proc. 15th Int. Conf. on Pattern Recognition* (Barcelona, 3-8 Sept. 2000).

[22] M. Skurichina, Ph.D. Thesis, Delft University of Technology, 2000, to appear

[23] M. Skurichina and R.P.W. Duin, Regularization of linear classifiers by adding redundant features, *Pattern Analysis and Applications*, vol. 2, no. 1, 1999, 44-52

[24] A. Ypma and R.P.W. Duin, Support objects for domain approximation, in: L. Niklasson, M. Boden, T. Ziemke (eds.), *ICANN'98, Proceedings of the 8th International Conference on Artificial Neural Networks* (Skovde, Sweden, 2-4 September 1998), Springer, Berlin, 1998, 719-724.