Classifier problem archetypes

Robert P.W. Duin, David M.J. Tax
Delft University of Technology, The Netherlands
{r.p.w.duin,d.m.j.tax}@tudelft.nl

Anil K. Jain
Michigan State University, East Lansing, USA
jain@cse.msu.edu

Abstract

Every automatic classification procedure, defined by a classifier and an optimization procedure, has (a set of) problems for which it performs better than other procedures. The most simple of these problems we call the problem archetype of the classification procedure. In this paper we present 2D examples of these problem archetypes for 14 standard classifiers. It gives insight in the focus and robustness of the classification procedures[1].

1. Introduction

The analyst who has to design a pattern recognition system for a given application has to make a choice from the large set of available classification procedures. These procedures differ in the classification model and the optimization routine. Some of them are entirely automatic, but many require the adjustment of parameters that can not be optimized fully automatically from the data. Still, most software packages offer default choices. We will call these standard classifier procedures.

The question of what is the best classifier procedure for a given problem cannot be answered straightforwardly. Some procedures focus on a particular type of problems and show a bad performance on many others. Other procedures are more robust and show an acceptable performance on a wide variety of datasets.

The pragmatic solution to compare classifiers is to run them over a set of standard problems. This produces tables with performances and shows the variability of classifier behavior over the chosen set. Some insight may be gained on classifier performances w.r.t. problem characteristics like sample size, dimensionality, number of classes and multimodality. Examples of such studies are [4, 5]. Another way to study the behavior of classification procedures is to analyze their assumptions and compare these over the set of procedures. The way some classifiers are defined, however, does not always map on a well defined model of problem characteristics. Still there is a need to build knowledge on what classifiers to choose for what type of datasets.

In this paper we try to achieve insight in classifier behavior by an entirely different approach, based on artificially generated problems only. First we state the following conjecture:

For every useful classifier procedure there exists at least a single classification problem, possibly artificially generated, for which it outperforms all other classifiers.

This is a consequence of the fact that any training procedure is explicitly or implicitly based on assumptions on the data distribution. A problem for which that distribution exactly matches the assumptions is perfectly suited for the corresponding classifier. If the assumptions used by classifier A are more strict than the assumptions for classifier B, then classifier A will perform better on a problem that just fulfills assumptions A. Classifier B will perform better if the assumptions B hold and the assumptions A do not.

A possible exception might be that a classifier C assuming 'B and not A' will do better than the more general classifier B. Consequently, there would be no problem for which classifier B is best. We think, however, that this situation is very rare. Often classifier A is a particularly biased version of B (for instance using an independence assumption), filling a subspace in the version space of A. This causes that classifier C becomes practically equal to classifier B.

The most simple classification problems for which a particular classifier outperforms all other classifiers is archetypical for that classifier. It fulfills its basic assumptions and nothing more. It will thereby clearly illustrate the classifier characteristics.

In this paper we set out to construct archetypical problems for each of 14 standard classifiers. We restrict ourselves to two-class problems with equal class prior probabilities in two dimensions and given by 50 training objects per class each. This is done for simplicity and for illustrative purposes. We realize that these restrictions may cause difficulties as some classifiers might be particular good for multi-class problems, high dimensionalities or for skewed priors.

[1] Unpublished paper, 2006

In the next section the classifiers and their proposed archetypical problems are presented as well as the results of the comparative experiments. The paper concludes with a discussion.

2. Classifiers and their archetypical problems

Here we present a set of 14 artificial two-class problems that illustrate the particular qualities of each of the 14 standard classifiers that we took from the PRTools toolbox [3]. We use default settings. Space does not allow to present the classifiers in mathematical terms or by pseudo code. Interested readers can download the software and the experiments from the author's website. General references for most of the classifiers we used can be found in [1, 2].

Our experiments are based on 25 training sets of 50 objects per class generated for each of the problems. Classifiers are tested on a test set of 1000 objects per class. Figure 1 shows the results. The datasets are presented by scatterplots of 200 objects per class, to give a better impression of the distributions. The classifier for which each dataset is designed is listed above the plot. On the left of the plots the averaged errors ($\times 1000$) are listed.

We continued to iterate over more trials just for this set. Above each scatterplot the number of trials is given for which the difference between the winning classifier and the next one is significant.

QDA. This is the quadratic Bayes classifier assuming Gaussian distributions. Two normal distributions with different, non-diagonal covariance matrices are generated.

UDA. As QDA, but assuming uncorrelated covariance matrices. So we aligned the distributions with the feature axes. QDA performs just slightly worse here, as it estimates full covariance matrices.

LDA. As QDA, but assuming equal covariance matrices. Other linear classifiers like the Logistic classifier and SVM-1 perform slightly worse here as they do not use the Gaussian model.

Nearest-Mean. The nearest mean classification rule is optimal for spherical classes that only differ in mean. So we generated two spherical Gaussian distributions. Other linear classifiers, not based on the Gaussian model, perform slightly worse.

Parzen. Our Parzen classifier uses a spherical Gaussian kernel for which the width is optimized for the classification performance on the training set using leave-one-out cross-validation. This non-parametric density estimation rule is similar to the K-NN classifier. The use of a Gaussian kernel, however, has an advantage if in the overlap region the tails of the class distributions have also a Gaussian shape. In order to make this problem difficult for the Bayes Gaussian classifiers like QDA we generated two 'banana'-shaped classes (uniformly sampled sinusoidal curves) convoluted with a spherical normal distribution. Consequently, neural network based classifiers constructing a general non-linear decision boundary also perform well on this data.

K-NN. Our implementation of the K-Nearest Neighbor rule optimizes K by a leave-one-out cross validation procedure over the training set. The K-NN rule effectively adapts the size of the neighborhood to the local density. We generated the same banana-shaped classes as used for the Parzen classifier problem, but additionally transformed the features by an exponential scaling. This deteriorates the performance of the Parzen classifier (using equal width kernels) more than that of the K-NN rule.

1-NN. The 1-Nearest Neighbor rule is sub-optimal for overlapping classes. However, various attempts to use non-overlapping two-dimensional domains for the classes (e.g. as used for the Decision Tree and the Radial Basis Support Vector Machine, RB-SVM) failed, as either a neural network or the RB-SVM always performed better. Therefore, a rather artificial spiral problem was selected, sampled uniformly with the polar angle. The other classifiers, except K-NN perform bad for this problem. The difference in performance with K-NN was after 1000 trials not significant.

Naive-Bayes. The Naive-Bayes rule assumes independent features. It has shown a surprisingly good performance in high-dimensional problems for which standard density estimations are problematic. The presented 2D-example is based on a uniform distribution along one axis and a two-mode Gaussian distribution along the other. These two axes are reversed for the two classes.

RB-Neural-Net. The implementation of the Radial Basis Neural classifier is based on Matlab's Neural Network Toolbox. In this implementation the neural network is grown by adding hidden units, till a maximum of 100 units. The second layer of the network has two sigmoidal output units. This classifier often performs well when general non-linear decision functions are demanded, but it is usually in competition with Parzen and K-NN. In case classes hardly overlap, it wins as it emphasizes geometrical discrimination over modelling of distributions.

Dec-Tree. The decision tree we implemented is based on a CART-like procedure, maximizing the purity. As it treats features separately, the effective decision function is piecewise linear, parallel to the feature axes. So we shaped the classes aligned with the features and prevented class overlap in order to profit optimally from the purity strategy.

LM-Neural-Net. Again Matlab's Neural Network Toolbox was used for training a feed-forward neural network with 5 hidden units using the Levenberg-Marquardt rule. As can be observed, this classifier is able to implement rather sharp non-linearities. The example is the same as for the decsion tree, but now rotated.

Logistic. The logistic classifier is a linear, robust procedure that concentrates on the area of overlap and is in-

sensitive to the shapes of the class distributions outside this area. The Gaussian density based classifiers suffer from the additional modes we generated for the classes. We had to make the overlap area small, aligned with the classes and had to use uniform instead of Gaussian distributions in order to achieve a better performance than the linear Support Vector Machine SVM-1.

SVM-1. We used a ν-SVM procedure [6], estimating ν by the 1-NN error. It appeared to be very difficult to find an example for which the linear support vector machine SVM-1 is better than the logistic classifier. This may be caused by the fact that the weights for the erroneously classified objects are limited by the sigmoid function in the logistic classifier while in the SVM they are weighted linearly with the distance to the classifier. In the constructed example the SVM is slightly better as the logistic classifier is somewhat unstable for small class overlaps.

RB-SVM. The Radial Basis Support Vector Machine, has a Gaussian shaped kernel. Again, we used a ν-SVM machine [6], estimating ν by the 1-NN rule. The width of the kernel was optimized in 10 steps using 10-fold cross validation. Consequently 101 classifiers have to be computed, which makes the learning procedure very time consuming. However, this classifier often performs well, especially for non-linearly separable problems. Class overlap should be avoided to make it better than Parzen and the K-NN rule.

3. Discussion

We successfully created for every standard classifier in the used toolbox an example for which that classifier outperforms all the others. Especially for the Naive-Bayes classifier and the SVM-1, this appeared to be difficult, mainly as a consequence of the limitations in 2D feature spaces.

The presented examples clearly show the specific classifier characteristics like linearity, emphasis on class overlap or model robustness. As we restricted ourselves, due to space limitations, to problems with a fixed training set size and dimensionality, issues like overtraining, stability and dimensionality sensitivity have been neglected.

A further analysis of the data shows that some classifiers are very similar, e.g. K-NN and Parzen, or Logistic and SVM-1. The Nearest Mean rule is an outlier in the sense that it is only good for its own problem. The radial basis rules for the SVM and the neural network are good, robust performing classifiers, directly followed by the LM-Neural-Net, Parzen and K-NN. These latter two rules seem to perform better for overlapping problems and the neural networks, SVMs and 1-NN rule for almost separable classes. The logistic classifier is an overall robust linear classifier.

It does not make much sense to average classifier performances over different problems as the presented set of examples will not be representative, in distribution, of any set of practical problems. The worst case performances of classifiers, however, may indicate the risk (in terms of the worst thing that may happen) of using a particular classifier for an arbitrary problem. If we exclude the very artificial spiral problem (1-NN) and normalize by the best performance for a particular problem, then the best classifiers ranked according to their worst relative performance are (i). RB-SVM (1.68), (ii). RB-Neural-Net (1.74) and (iii). 1-NN (1.97), indicating that the RB-SVM was at most 68% worse than the best classifier for the same problem (which was for this classifier the UDA problem: 89/53 = 1.68) .

We think that the methodology presented in this paper may be worthwhile for discussing new classifier proposals. They should be accompanied with their own archetypical problems and may be compared with those of the standard classifiers to judge their characteristics. An extension of this methodology to multi-class problems and multi-dimensional feature spaces may increase its value for real world applications.

References

[1] R.O. Duda, P.E. Hart and D.G. Stork, Pattern classification, Second Edition, Wiley, New York, 2001.

[2] A.K. Jain, R.P.W. Duin, and J. Mao, Statistical Pattern Recognition: A Review, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, 2000, 4–37.

[3] R.P.W. Duin, P. Juszczak, D. de Ridder, P. Paclik, E. Pekalska, and D.M.J. Tax, PRTools, a Matlab toolbox for pattern recognition, http://prtools.org, 2004.

[4] D. Michie, D.J. Spiegelhalter and C.C Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, 1994.

[5] T.S. Lim, W.Y. Loh, Y.S. Shih, A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms, Machine Learning, Vol. 40, 203-228, 2000.

[6] B. Scholkopf, A.J. Smola, R.C. Williamson, P.L. Bartlett, New Support Vector Algorithms, Neural Comp. Col. 12, 1207–1245, 2000.

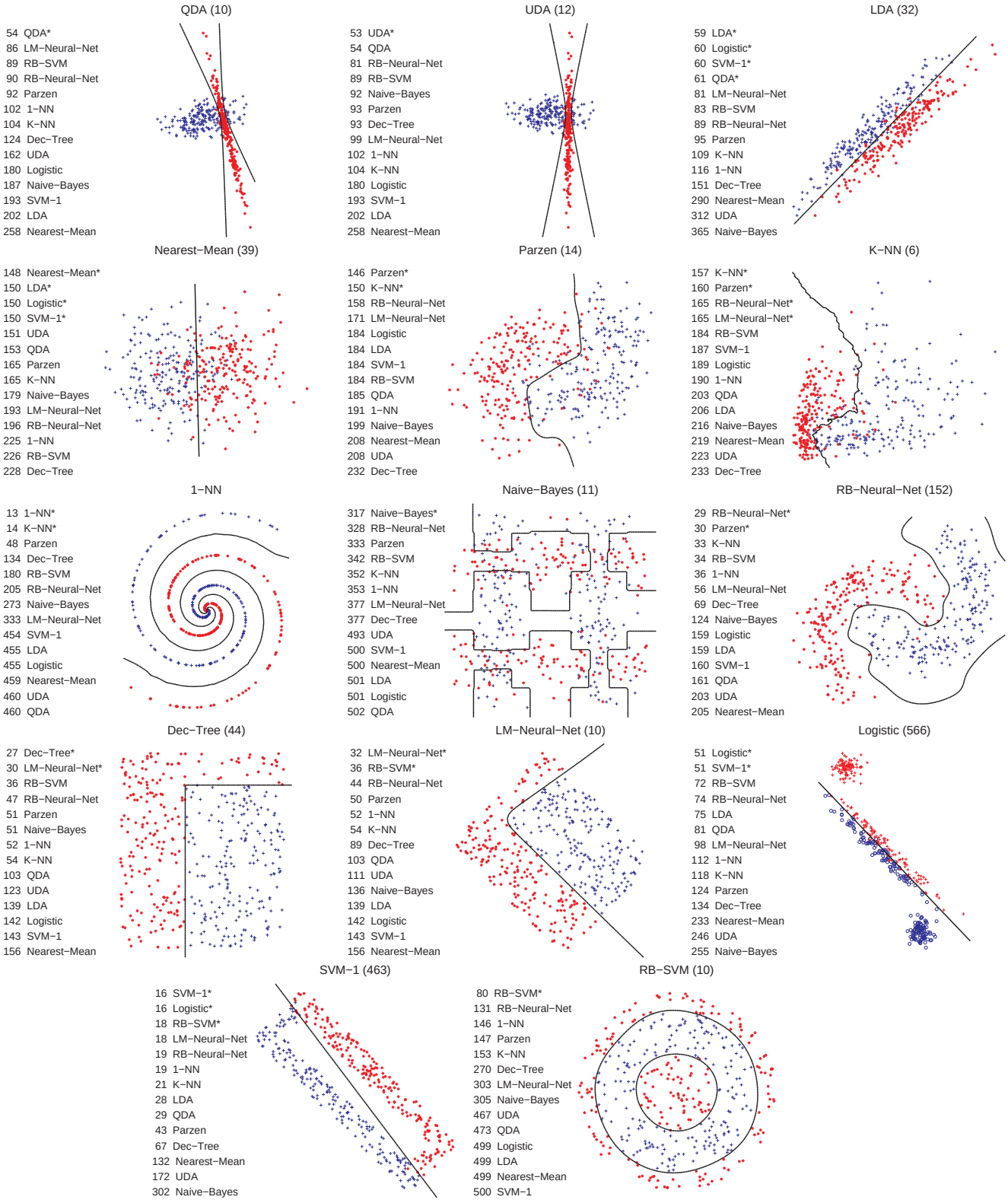

Figure 1. Archetypical problems for the 14 standard classifiers. Mean errors ($\times 1000$, averaged over 25 experiments) are listed left of scatterplots based on 200 objects per class. Standard deviations are estimated from the 25 experiments. Performances that are not significantly different from the best results are indicated by *. The number next to the title indicates how many experiments are required to obtain a significant performance difference between the best and second best classifier. The higher the number, the smaller the difference in statistical terms between the best classifiers.