Pieter P. Jonker, Robert P.W. Duin, Dick de Ridder:

Pattern recognition for metal defect detection

This paper describes how to classify a data set containing features extracted from metal strips, using pattern recognition algorithms. In the first part, a short resume of pattern recognition principles and algorithms is presented, while in the second part the techniques are applied on steel samples obtained from the Anshan Steel Corporation, P.R. China. From the images made and pre-processed by the Institute of Bildsame Formgebung Aachen, Germany, features were extracted using ParsyVision from Parsytec GmbH Aachen, Germany. On these features we used several classifiers. The influence of the feature set size and sample size of the master set of samples was illuminated. Finally we established a checklist for pilot projects on automatic steel inspection systems.

Pattern recognition

Pattern recognition is the science of information processing procedures that are able to classify, describe and label measurements. Objects (e.g., a pit in a metal sheet) of unknown class are sensed (e.g., with cameras), pre-processed (e.g., with image processing algorithms) and observable features of the object are measured (e.g., area, perimeter). We will call this pre-processing system the *region of interest* (ROI) detector. The pattern recognition system is then able to estimate the class of the object, sometimes with a measure of confidence (e.g., the object belongs to a class with a confidence of: A: 80 %, B: 15 %, C: 5 %). In cases the system itself determines classes we usually refer to this as cluster algorithms; the algorithm discovers clusters in the data that can be thought of as object classes. Systems that can learn from examples that are classified by an expert are called supervised learning systems. In such a system, the master set of samples, which truly represents the statistics of the production facility, is classified by experts and then randomly divided in a *training set* and a *test set*. In training mode, the samples from the training set are fed into the system, whereupon the system produces a classification result that is likely to be wrong in first instances, but iteratively -in a learning or training procedure- the error between the class labelled by the expert and the class estimated by the system is minimised. To estimate the performance of the system, the test set should be used.

The key principle in such systems is that the extracted features of the object span a *feature space*. The dimensionality of this space is as large as the number of features that are used. By way of example, **fig. 1** shows a feature space. Each observed object (e.g. pits in a metal sheet), with measured features area and perimeter, is represented as a point P (perimeter, area) in a two dimensional space.

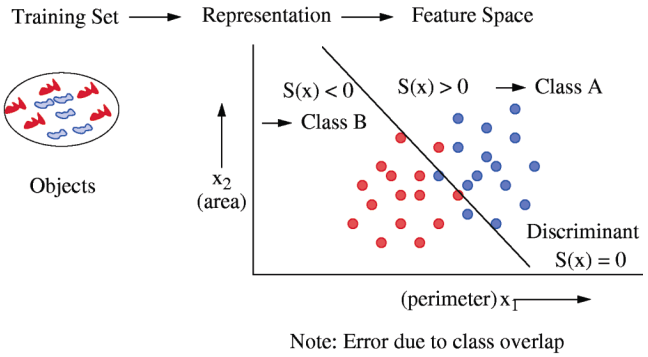


Fig. 1. Feature space and discriminant function

If we take a linear function (in 2D a straight line) as a discriminant function, we can use the training set to estimate the parameters of this line, and use the found equation of the line to classify the test set to measure the system's performance, or in production to classify the samples that our ROI detectors find. However, due to the linearity of the classifier we are bound to make errors if classes are entangled or, worse, overlap.

There are several methods to obtain the discriminant function or approximate it. The fastest method is to use the features in a decision tree as is shown in **fig. 2.** In this way the discriminant function is approximated with line segments parallel to the feature axes. This yields a fast but moderate performance. Moreover it is a difficult procedure to find the threshold values for each feature in the decision tree.

A popular but less understood classifier is the neural network with back propagation learning rule, see **fig. 3.**

In such a system each (feature) input is coupled to each of the units in the hidden layer with a certain weight factor, and each (class) output is coupled onto each hidden unit

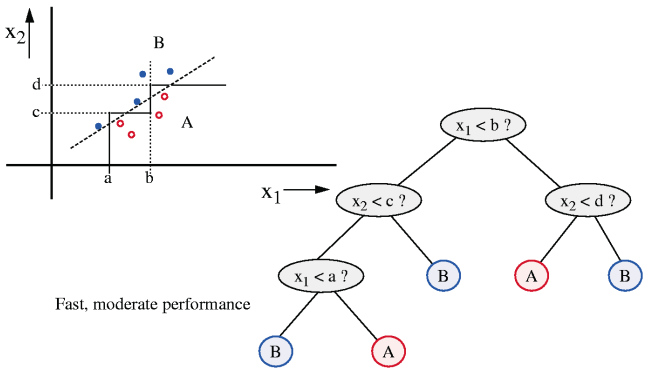


Fig. 2. Decision tree analysis

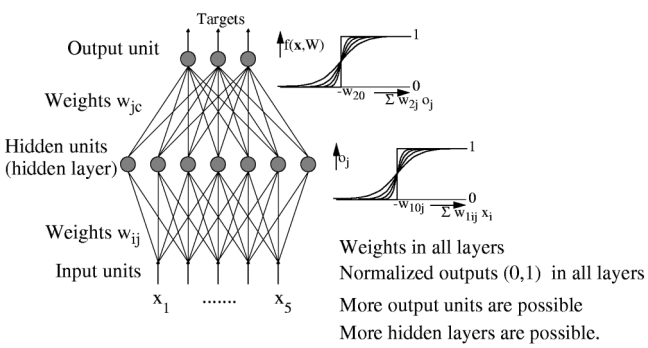


Fig. 3. Neural network with back propagation learning rule

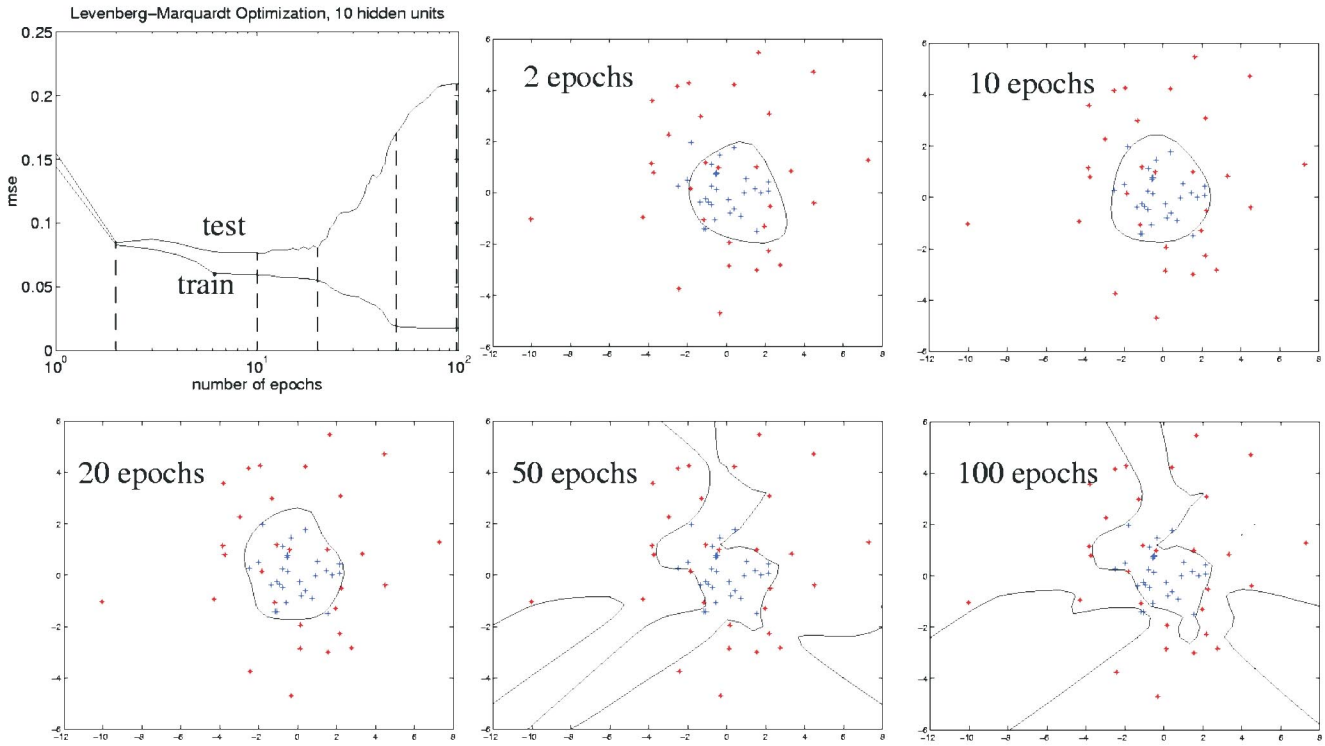


Fig. 4. Examples of training and overtraining

also with a certain weight factor. The transformation from input to hidden layer represents a remapping of the features to a transformed feature space with another dimensionality. Useful features are attenuated, whereas less important features are suppressed. The transformation from hidden layer to output layer forms the classification. In the learning phase a training set of data is repeatedly presented to the system, which results in updates of the weights. The error between the class labelled by the expert and the class estimated by the system will gradually diminish. However, the independent test set should be used to prevent *overtraining*. Overtraining occurs when a system learns the examples by heart and is not able to generalise any more, i.e. the system adapts itself to the noise in the training set. If the error in the test set goes up, training has gone too far and as the ideal trained network, a predecessor has to be taken. This is shown in **fig. 4,** a problem with two overlapping classes. Here a network and its discriminating curves are drawn after 2, 10, 20, 50 and 100 epochs. After 10 epochs, overtraining takes place and the system will not classify correctly anymore.

A recent, very promising development in pattern recognition is the *support vector classifier*. This classifier is based on the fact that a discriminating function can be obtained from the training samples that are the closest to the class boundaries. **Fig. 5** shows two examples in 2D, a linear discriminant function and a curved function, both set-up by their supporting training points (vectors). The benefits of this system is that it is insensitive to small samples sizes in high-dimensional feature spaces, and it is not so much influenced by outliers, as it seeks the boundary between the classes, disregarding the class average. The SV classifier trains slow but classifies fast.

In all trainable systems, there is a relation between the numbers of features used, the number of parameters of the system, the classification error on the test set and the number of samples of the training set. This relation is shown in **fig. 6** and is also known as the *peaking phenomenon*. Observing the curves, one can reason that above a certain number of features used, overtraining will be a problem. The lowest curve indicates the error on the training set, while the bundle of upper curves indicates the error on the test set. With an increasing number of features, the error on the test set diminishes, but if the number of features becomes too large, the error on the test set grows again, due to overtraining. We showed this in fig. 5. A new test error curve with a minimum further to the right is found, if a larger number of samples is used to train the system. This also leads to a lower possible minimum value of the error on the test set. So we have a problem if the number of

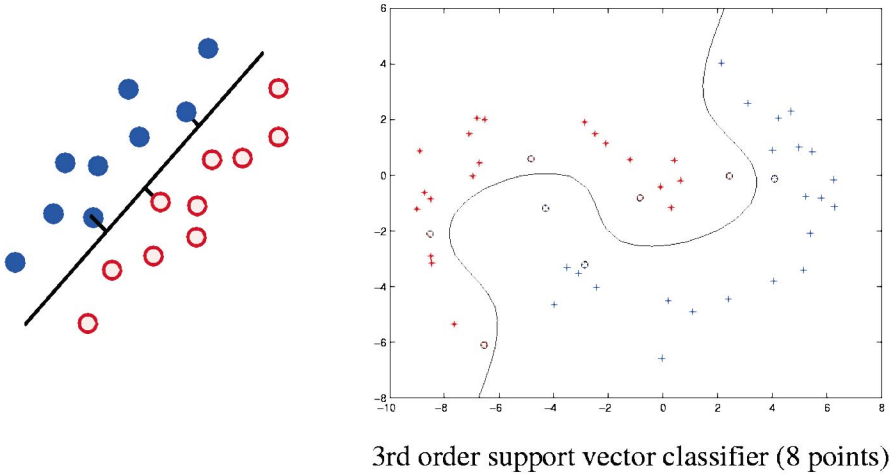


Fig. 5. Support vector classifiers

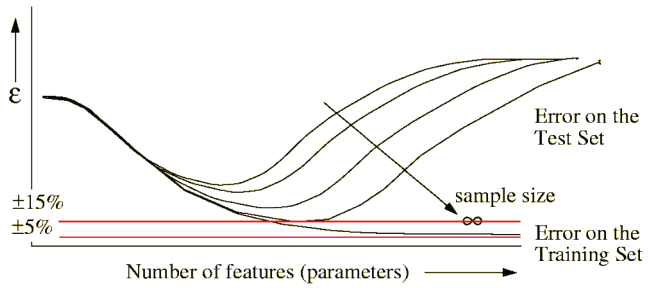


Fig. 6. The peaking phenomena

training samples is low. The reason for this can be partly understood by observing that the features span a high dimensional space, in which the training samples are points from which the discriminant function must be derived. No adequate structure can evolve if there are not enough sample points available. I.e., one needs to have at least 2 points to span a line in a 2D space ($y = \boldsymbol{a}x + \boldsymbol{b}$) and so we need to have at least N sample points in an N-dimensional feature space for a linear classifier. E.g., with 20 features and 4 classes a master set of samples (train plus test set) of at least $20 \cdot 4 \cdot 2 = 160$ samples is necessary. In the case of a linear classifier, (flat) hyperplanes separating the classes segment the hyperspace. (In case of a decision tree these planes are even always perpendicular to the main axes of the space.) Usually more sophisticated classifiers (e.g. neural networks or support vector classifiers) are necessary that can realise deformable surfaces in hyperspace to separate the classes. However, the number of necessary sample points grows with the number of free parameters of the classifier. Consequently, to get a low error on the test set, one needs many samples, especially when the number of features used is high. Often in inspection pilot projects this rule is violated.

Do's and Don'ts for automated defect detection

To elucidate further do's and don'ts for automated defect detection, we use data from the Anshan Steel Corporation[1] and a ParsyVision from Parsytec GmbH to acquire ROIs, as well as our own classification software PRTools, a Matlab toolbox for pattern recognition. As we noticed that the features for bright field and dark field were highly uncorrelated, we are in fact dealing with two independent three-class problems, see **table 1.** Analyzing all possible (79) features we manually selected 22 features of which we had indication that they were relevant for the problem. Since the feature values differed wildly, each feature was normalised by subtracting the mean value and dividing it by the standard deviation. The samples were split at random in a training set (67 %) and a test set (33 %). The experiments were performed 5 times to get an idea of the variance in results, so each time a new random selection of the training and test-set was made out of the Master set of samples. We found out that making for each set (BF and DF) a two-stage classifier, first sieving the pseudo defects and then classifying the remainder in separate defect classes was very beneficial. For both sets, linear classifiers were found to be insufficiently powerful. As a consequence of the fact that not all classes were represented well (the class *Coil Break* had even hardly any samples) a quadratic classifier could not be used. The best performing classifier was the support vector classifier, which can also form a linear classifier when insufficient data is available, see **fig. 7.** Table 1 shows the confusion matrix.

However, the error on the test set remained too high. Inspection using principal component analysis (PCA) revealed that the intrinsic dimensionality of the data sets was around 10; meaning that 95 % of the variance in the data is due to 10 features, and 75 % of the variance is due to 4 features only, being far less than the original number of 22 features, see **fig. 8.** We therefore applied a forward-selecting feature reduction mechanism.

As the number of samples, e.g., for coil break was below a minimum, we artificially created new samples using

Table 1: The confusion matrices as result on the test-set on the bright field and dark field problems

True class → classified as: ↓	em. mark	rust	scale	pseudo
em. mark	129	1	0	23
rust	0	183	1	31
scale	0	0	546	186
pseudo	1	6	26	167

true class → classified as: ↓	coil break	edge crack	roll imprint	pseudo
coil break	0	0	0	2
edge crack	0	24	0	5
roll imprint	0	0	5	3
pseudo	1	7	6	131

Gaussian noise. Moreover, when the distribution of the training samples does not reflect the relative severity of the defects (a lot of rust, no coil break), an a-priority defect distribution matrix should be made, that makes it possible to compensate for this during training.Prior probabilities are very important, e.g., if defect A occurs 4 times as much as defect B, this can lead to a choice for different classifier than if we assume equal class probabilities.

And finally, if the cost of one defect is higher than that of another, this should be used during training, using a cost matrix. Misclassifications have different costs. The mill

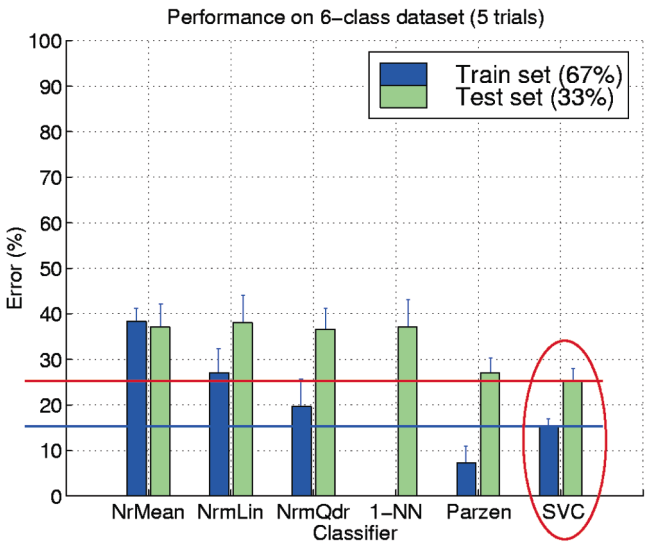


Fig. 7. The performance of various classifiers

[1] derived from *De Ridder et al.:* WP2.2.1. classification of steel samples, QC-SIASIS, INCO-DC project 961895

might be stopped for maintenance if (false) roll-imprints occur.

If companies have no data on this, assumptions might be used to correct the result. Finally, after removal of the pseudo defects and with own assumptions on a-priority defect distributions and cost matrices, we could come to results that had errors on the test set below 2 ± 1 % for the bright field 3 class problem and 7 ± 2% for the dark field 3 class problem using a support vector classifier.

Conclusions

In this paper we briefly reviewed the research area of pattern recognition with respect to current defect detection systems for metal strip production. We then provided considerations, based on a pilot in a EU-project, on do's and don'ts for pilot projects. We conclude:

⇒ a pre-processing system is a first sieve. It selects regions of interest with candidate defects. All defects not detected here will not be detected at all. With multiple cameras, defects might be split in separate ROIs (e.g., long scratches perpendicular to the belt), which may lead to wrong results;

⇒ one has to establish which defect classes one wants to distinguish, and which ones are really relevant;

⇒ establish a master set of samples that represents the statistics of the production system;

⇒ manually select a reasonable number (± 20) of the (± 300) available features of the inspection system;

⇒ use principal component analysis on the master set of samples to detect how many of those reasonable number of features are really relevant;

⇒ apply an automatic feature selection mechanism to select features that are effective and orthogonal;

⇒ separate the data in an *independent training and test set.* Let relative class frequencies mirror the probability of occurrence of an error;

⇒ most defects are either based on bright field or on dark field features, so the classification problem can be split in a bright field and a dark field problem;

⇒ it is beneficial to separate the system in an outlier detector that prunes away the pseudo defects, followed by a subsequent defect classifier;

⇒ select a classifier that does the job good and fast and can handle the available training samples in the training set and the number of features that are used;

⇒ look at the *errors in the test set,* not in the training set. Manufacturers have to specify their performance based on an independent test set, *not* on a training set;

⇒ are there enough samples in each defect class? Even if not enough samples per class can be obtained to reflect the prior probabilities, artificial generation of additional data (e.g., using Gaussian noise) can be used to enlarge certain class sample sizes;

⇒ what are the prior probabilities of the defects? This knowledge may be used to unbias the master set of samples;

⇒ it is beneficial if the client could specify costs of misclassification per class in an award or penalty matrix;

⇒ train the pseudo defects, test to prevent over-training;

⇒ train the defect classes, test to prevent over-training;

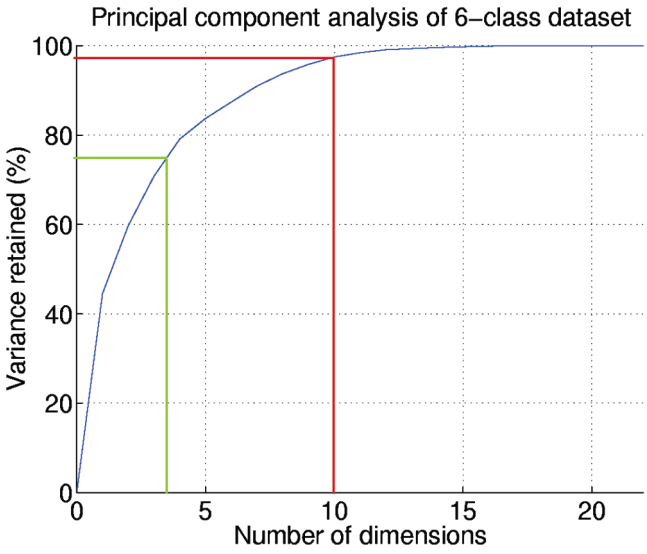


Fig. 8. Principal component analysis

⇒ *support vector classifiers* are a good alternative for neural networks. They hook onto class boundaries, so are insensible for outliers;

⇒ *start collecting data.* Make use of the inspection machine first to collect a master set of samples that is representative for the production facility. Several hundred samples per class (200 - 500) are usually necessary to obtain good insight into the problem.

References

[1] *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.

[2] *K. Fukunaga:* Introduction to statistical pattern recognition, 2nd edn., Academic Press, New York, 1990.

[3] *S.J. Raudys and A.K. Jain:* IEEE Trans. Pattern Anal. Mach. Intellig. 13 (1991) No. 3, p. 252/64.

[4] *J.R. Quinlan:* C4.5: Programs for machine learning, Morgan Kaufmann Publishers, San Mateo, California, 1993.

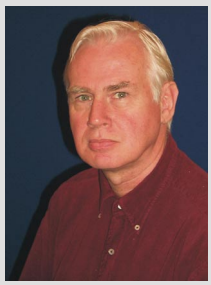
[5] *C.M. Bishop:* Neural networks for pattern recognition, Clarendon Press, Oxford, 1995.

[6] *T. Kohonen:* Self-organizing maps, Springer Series in Information Sciences, Vol. 30, Berlin, 1995.

[7] *V.N. Vapnik:* The nature of statistical learning theory, Springer Verlag, Berlin, 1995.

[8] *E. Gose, R. Johnsonbaugh, and S. Jost:* Pattern Recognition and Image Analysis, Prentice-Hall, Englewood Cliffs, 1996.

[9] *R.P.W. Duin:* PRTools, a Matlab Toolbox for Pattern Recognition, http://www.ph.tn.tudelft.nl/~bob/PRTOOLS.html



Dr. ir. Pieter P. Jonker *Dr. ir. Robert P.W. Duin* *Dr. ir. Dick de Ridder*

Pattern Recognition Group, Faculty of Applied Sciences, Delft University of Technology, The Netherlands.