A classification scheme for applications with ambiguous data

Thomas P. Trappenberg
Centre for Cognitive Neuroscience
Department of Psychology
University of Oxford
Oxford OX1 3UD, England
Thomas.Trappenberg@psy.ox.ac.uk

Andrew D. Back
Katestone Scientific
64 MacGregor Tce
Bardon, QLD 4065 Australia
back@usa.net

Abstract

We propose a scheme for pattern classifications in applications which include ambiguous data, that is, where pattern occupy overlapping areas in the feature space. Such situations frequently occur with noisy data and/or where some features are unknown. We demonstrate that it is advantageous to first detect those ambiguous areas with the help of training data and then to re-classify those data in these areas as ambiguous before making class predictions on test sets. This scheme is demonstrated with a simple example and benchmarked on two real world applications.

Keywords: data classification, ambiguous data, probabilistic ANN, k-NN algorithm.

1. Introduction

Adaptive data classification is a core issue in data mining, pattern recognition, and forecasting. Many algorithms have been developed including classical methods such as linear discriminant analysis and bayesian classifiers, more recent statistical techniques such as k-NN (k-nearest-neighbors), MARS (multivariate adaptive regression splines), machine learning approaches for decision trees etc, including C4.5, CART, C5, Bayes tree and neural network approaches such as multilayer perceptrons and neural trees [1-8].

Most of these classification schemes work well enough when the classes are separable. However, in many real world problems the data may not be separable. That is, there may exist regions in the feature space that are occupied by more than one class. In many problems, this ambiguity in the data is unavoidable.

A similar problem occurs when the data are very closely spaced and a highly nonlinear decision boundary is required to separate the data. Accordingly, the aims of much recent work on classification has been to seek ways to find better nonlinear classifiers. Particularly notable in this area is the field of support vector machines [9, 10]. SVMs have captured much interest, as they are able to find nonlinear classification boundaries while minimizing the empirical risk of false classification.

However, it is important to consider what the data really means and what the practical, real world goals are. In many cases, it is desired to find a simple classifier which gives the user a rough, but understandable guide to what the data means. On the other hand, the data itself may be contaminated by noise, some input variables are completely missing (i.e. the data is then in a feature space which is implicitly too low) or when the data is flawed in other ways. This issue is commonly referred to in the context of robust statistics and outlier detection.

In this paper we propose a method for preprocessing ambiguous data. The basic idea is quite straight forward: rather than seek a complex classifier, we aim to first examine the data with the aim of removing any ambiguities. Once the ambiguous data is removed, we then apply whatever classifier is required, hopefully one which may lead to a much

simpler solution than would otherwise be obtained. In doing this we acknowledge that data in some regions of the state space can either not be classified at all or else our confidence in doing so is low. Hence those data points are labeled in a different manner to facilitate better classification.
Our proposed scheme is to identify ambiguous data and to re-classify those data with an additional class. We will call this additional class IDK (I don't know) to indicate that predicting the class of this data should not be attempted due to it's ambiguity. By doing so one looses some prediction of data. However, we will show that we can gain on the other hand a drastically increase in the confidence of the classification of the remaining data.
Our proposed scheme is outlined in the next section. There we will also introduce a particular implementation that will be used in the examples discussed in this paper. The synthetic example in section 3 is aimed to illustrate the underlying problem on the proposed solution in more detail. In section 4 we will report on the performance of our algorithms on two real world (benchmark) data sets, that of the classical Iris benchmark [11], and that of a medical data set [12].

2. Detection of Ambiguous Data

As stressed in the introduction, our scheme is to detect ambiguous state space areas and to re-classify data in these areas. Hence, our proposed scheme has the following steps:

1. Test all data points against a criteria of ambiguity.
2. Re-classify training data which are ambiguous.
3. Classify test data with algorithm trained on the re-classified data

Note that the scheme is independent of any particular classification algorithm. In practice it might be critical to choose appropriate algorithms for each of these steps. As a means of illustrating the proposed method, we use some particular algorithms which are described below.
For the first step we employ a k-NN algorithm [3]. This algorithm takes the k closest data points to a data point in question into account to decide on the new class of this data point in question. If an overwhelming majority of neighboring data is of one particular class, then this class is taken to be
the class of this data point. If no overwhelming majority can be reached then this data point is declared as ambiguous and classified as member of the class IDK.
The next step requires a classification method for the predictive classification of further data (test data). While any type of adaptive classifier could be chosen, in the following test we use a feedforward neural network Hidden Layer:

$$\text{Hidden Layer}: a_j^{(1)} = \sum_k w_{jk}^{(1)} x_k + \theta_j^{(1)}; \quad h_j = f^{(1)}(a_j^{(1)})$$

$$\text{Output Layer}: a_i^{(2)} = \sum_j w_{ij}^{(2)} h_j + \theta_i^{(2)}; \quad y_i = f^{(2)}(a_i^{(2)})$$

with a softmax output layer defined by the normalized transfer function

$$y_i = \frac{\exp(a_i^{(2)})}{\sum_k \exp(a_k^{(2)})}$$

so that the outputs can be treated as probabilities, or confidence values, that the input data belongs to a class indicated by the particular output node. This network is trained on the negative cross entropy

$$E = -\sum_\mu \sum_i t_i^\mu y_i(\mathbf{x}^\mu; \mathbf{w})$$

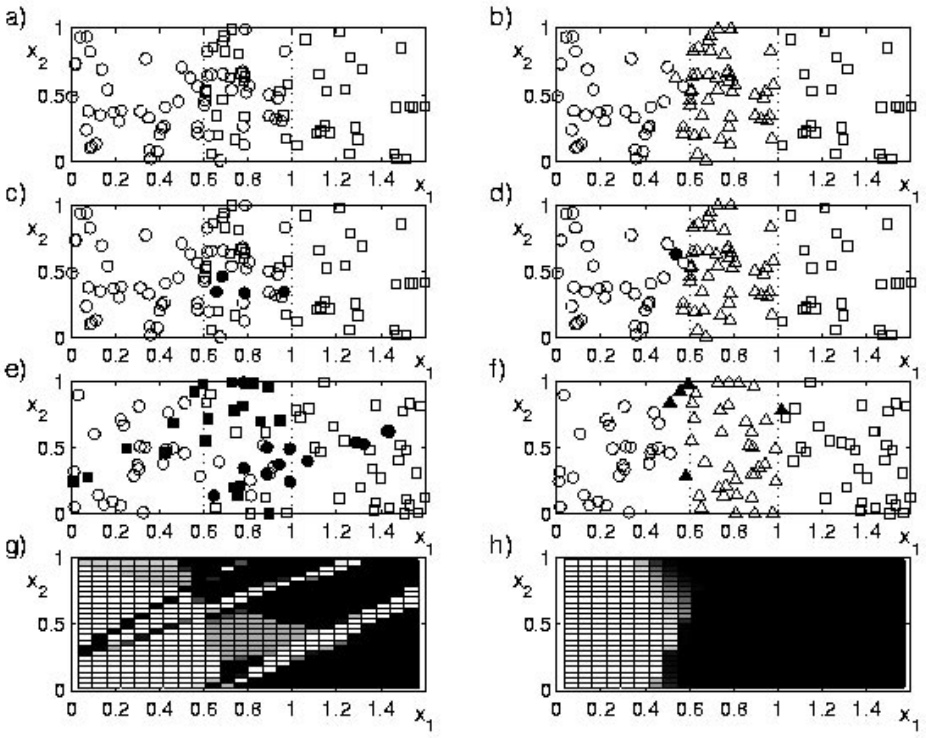

Figure 1: *Example with overlapping data. The left column shows examples within a standard classification procedure, whereas the right column shows examples with the proposed re-classification scheme. (a) The raw input data (training set) with data from class a (circles) and class b (squares). (b) Re-classified training set including ambiguous data in class IDK (triangles). (c) Classification of the original training data after training with a probabilistic MLP. False classifications are marked with solid symbols. (d) Classification of the re-classified training set. (e,f) Performance on a test set. (g,h) Probability surface of class a generated by the two networks.*

which is appropriate to give the network outputs the probabilistic interpretation [13]. t_i^{μ} is thereby the target vector of training example μ with component $t_i = 1$ if the example belongs to the class i. In the examples below we use the Levenberg-Marquardt algorithm (LM) [14] to train the network on the training data set.

4. Partially Overlapping Classes: An Example

Here we illustrate the proposed scheme using two overlapping classes in a two dimensional state space. In the following we define two classes *a* and *b*. The features x_1 and x_2 of the data from each class are thereby drawn from an equal distribution with overlapping areas:

$$\text{class a}: \quad x_1 \in [0,1]; \quad x_2 \in [0,1]$$
$$\text{class b}: x_1 \in [0.6,1.6]; \quad x_2 \in [0.6,1.6]$$

An example of 100 data points of these two classes is shown in Figure 1(a), data from class *a* as circles and data from class *b* as squares. For comparison we trained a classification network directly on these training data. The networks had always 10 hidden nodes and were trained with 100 LM training steps. The classifications of the training data with this network after training are shown in Figure 1(c). Only 4 data points have not been classified correctly. The network even learned to classify most of the training data in the ambiguous area.
The re-classification of these training with the KNN algorithm described above is shown in Figure 1(b). Data with components $0.6 < x_1 < 1$ are ambiguous. K = 10 nearest neighbors (including the data point itself) were taken into account when choosing a new class structure for this data set. The class of the data point was set to the majority of data if 80% or more of the neighboring data (including the data point itself) were of this majority. If such a majority could not be reached the data were classified as class IDK and symbolized with open triangles in the figures.
These re-classified data were used as training data for a second classification network similar to the previous network. We only added one output node to account for the additional class IDK. The classification of the re-classified training data with the classification network after training is shown in Figure 1(d). Only one data point was not correctly classified.
More important than the performance of the classification network on the training data is that of the performance on test data. An example is shown in Figure 1(e) and 1(f) for the two separate classification networks respectively. The network that was trained with ambiguous data (Figure 1(e)) falsely classified 1/3 of the test data corresponding to a standard performance value of $P' := n_c/n = 0.67$, where n_c are the number of correct classifications and n are the number of data. As might be expected there are of course numerous false classifications in the area with ambiguous data. However, what is even more disturbing is that there are a lot of false classifications of data far away from this area. This can also clearly be seen from the prediction surface of this network, which is illustrated in Figure 1(g) with gray values. White correspond thereby to a confidence (value of the first output node) of 1 of predicting class a with this network, whereas black correspond to a confidence of 0 of predicting class a (which correspond to a confidence of 1 to predict class *b* in this example). It can clearly be seen that the attempt of the network to find a classification scheme in the area with ambiguous data let to the proposal of structure that does not correspond to the underlying problem. This structure is extrapolated to areas without ambiguous data leading to the pure performance on data in these areas of the input space.
The situation is much better with the re-classified data. The results of the classifications of the same test data with the network trained on the re-classified data are shown in Figure 1(f). Only five pattern have been falsely classified, all of which are close to the boundaries of the area with ambiguous data. This correspond to a standard performance of $P' = 0.95$ (compared to 0.67) when taking all classes into account. The improvement comes largely from the fact that the underlying problem has not any longer ambiguous data, so that perfect classifications can be expected in the limit of infinite data. This is contrary to the original problem which includes ambiguous data and hence a perfect classification can not be archived in the limit of infinite training data.
Indeed, in the example shown in Figure 1(f), no data have been classified wrongly when taking only predictions of class *a* and *b* into account. This will not always be the case but will be true in the infinite data limit. Moreover, the areas far away from the ambiguous area can be predicted with high confidence, and the false classifications will have a much lower confidence value.

4. Real World Data: Some Benchmark Examples

The previous example was intended to describe our scheme and to illustrate why this scheme should be useful. However, only data taken from real-world examples can tell if this scheme is useful in practice. Hence, we will report in the following on some initial study of the application of this scheme to some real world data. The following examples are all taken from the UCI repository of machine learning databases [15] which can be accessed via the Internet.

4.1. Iris Dataset

We tested our scheme first on the classical Iris-data benchmark. The dataset contains 150 examples with 4 physical properties of 3 members of the family of iris flowers. The dataset was first used by Fischer in 1936 [11] to illustrate multidata discriminant analysis techniques in taxometric problems.
We divided the dataset evenly into a training set and a test set by taking 25 examples from each class into each subset. The training dataset was re-classified with the same procedure as in the example of section 3 without

adjusting any parameters. This re-classification run classified 8 examples of the training data set as class IDK. The class label of all members of the first class were preserved, which showed that the training set of this class did not include ambiguous data and was easily separable from the other classes. This finding is in accordance with similar findings of Duch et al. [16].
We used a similar network as in the previous example for the classification task itself; only the number of output nodes were adjusted to represent the required number of classes. After 100 train steps the network was able to represent all data in the training sets, of the original data as well as of the re-classified data. The network made 4% false classifications (3 examples) on the test set with the network trained on the original data. However, no false classifications were made in the classification task with the network trained on the re-classified data when only taking classifications of iris types into account. The price to pay was that 11 examples of the test set were labeled as IDK.

4.2. Wisconsin Breast Cancer data

The second test was made on medical data compiled by Dr. William H. Wolberg at the University of Wisconsin Hospitals, Madison. A smaller database of these data was initially studied in [12]. The version of the dataset we used contained data from 699 patients with 9 predictive attributes used in breast cancer diagnosis. The data were classified in two classes: benign (458 instances, 65.5%) and malignant (241 instances, 34.5%). Data from 16 patients were incomplete. We ignored these records in the following test. However, it should be stressed that incomplete information should lead to more ambiguous data and should favor our approach. The effect of incomplete data will be discussed in more detail elsewhere.
We again used the same KNN re-classification algorithms as in the previous example without adjusting any parameters. The data were randomly divided into 340 training data and 343 test data. 14 data points were classified as IDK with the KNN re-classification algorithm. All data points of the training sets, the original on the re-classified, were classified correctly after training the networks. However, the network trained on the original data classified 23 instances (6.7%) of the test data incorrectly, whereas the second network made only 6 mistakes (1.75%). 22 instances of the test data were classified as IDK.

5. Conclusion and Outlook

In this paper we have proposed a scheme to solve classification problems with ambiguous data. We showed that classification problems with ambiguous data can lead to pure classification results, not only for data in the ambiguous areas but also for data in the areas which should have a much better predictability. We showed that the identification of ambiguous input areas and the use of re-classified data for the training of the classification algorithms can lead to an drastic reduction of false predictive classifications. Hence one should consider avoiding predictions of some data in areas which are highly ambiguous. We think that this approach is very suitable when predictions have to be made with particular caution.
There are many issues within this scheme that have to be discussed in the future. In particular we used only a simple k-NN re-classification scheme to identify ambiguous areas in the input space. We neither studied systematically the dependence on the parameters of this algorithm, nor did we explore which algorithms might be best suited for a particular problem. There are also a variety of classification algorithms available, each which might have advantages for particular applications. Our network can be improved with a Bayesian regularization scheme, which can also provide additional information on the complexity of the underlying problem. Some work in this direction is in progress. Also other advanced classification methods such as SVM can be used.

Acknowledgment: We would like to thank Wlodek Duch for the discussions of his results on rule extraction during his visit in Japan.

References

[1] Breitman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

[2] Buntine, W.L., Learning classification trees, Statistics and Computing 2, 63-73, 1992.

[3] Cover, T.M., Hart, P.E. Nearest neighbor pattern classifications, IEEE Transaction on Information Theory 13, 21-27, 1967.

[4] Duda, R.O., Hart, P.E.m Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[5] Hanson R., Stutz, J., Cheeseman, P. Bayesian classification with correlation and inheritance,Proceedings of the 12th International Joint Conference on Artificial Intelligence 2, Sydney, Australia,Morgan Kaufmann, 692-698,1991.

[6] Michie, D., Spiegelhalter, D.J., Taylor, C.C., (editors), Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1994.

[7] Richard, M.D., Lippmann, R.P., Neural network classifiers etimate Bayesian a-posteriori probabilities, Neural Computation 3, 461-483, 1991.

[8] Tsoi, A.C., Pearson, R.A., Comparison of three classification techniques, CART, C4.5, and multilayer perceptrons, in Advances in Neural Information Processing Systems 3 , 963-969, 1991.

[9] Vapnik V., The Nature of Statistical Learning Theory, Springer Verlag, New York, 1995.

[10] Vapnik, V., Golowich, S., Smola, A., Support vector method for function approximation, regression estimation, and signal processing, in Advances in Neural Information Processing Systems 9, 1997.

[11] Fischer R., The use of multiple measurements in taxonomic problems, Annals of Eugenics 7, pp.179-188, 1936

[12] O. L. Mangasarian and W. H. Wolberg, Cancer diagnosis via linear programming, SIAM News, Volume 23, Number 5, September 1990, pp 1-18.

[13] Amari, S., Backpropagation and stochastic gradient desent methods, Neurocomputing 5, 185-196, 1993.

[14] Hagan, M.T., Menhaj, M., Training feedforward networks with the Marquardt algorithm, IEEE Transactions in Neural Networks, vol.5, no.6, pp.989-993, 1994

[15] Mertz, C.J., Murphy, P.M., UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-databases

[16] Duch W, Adamczak R, Grabczewski K, Zal G (1999) Hybrid neural-global minimization method of logical rule extraction, Int. Journal of Advanced Computational Intelligence (in print).