1

Pattern Recognition *via* Neural Networks

B. D. Ripley

Pattern recognition has a long history within electrical engineering but has recently become much more widespread as the automated capture of signals and images has become cheaper. Very many of the applications of neural networks are to classification, and so are within the field of pattern recognition. In this chapter we explore how neural networks fit into the earlier framework of pattern recognition, and show by some examples that that framework can help us to make better use of neural networks for classification.

1 What is Pattern Recognition?

Pattern recognition is concerned with making decisions from complex patterns of information. The goal has always been to tackle those tasks presently undertaken by humans, for instance to recognize faces, buy or sell stocks and to decide on the next move in a chess game, but simpler tasks have been considered which were within the reach of the hardware and software available. A few of the tasks which have been tackled are

Distinguishing a bicycle and a pedestrian on a level crossing [1].
Classifying galaxies (as spiral, elliptical and more finely).
Medical diagnosis.
Reading Zip codes (US postal codes) on envelopes [21].
Reading hand-written symbols on a pen-pad computer.
Predicting suitable habitat for Tsetse flies [29].
Financial trading [28].
Species and sex of *Leptograpsus* crabs [7].
Forensic studies of DNA, fingerprints [8], glass fragments [30, 32] ....
Identifying incoming missiles.
Credit allocation rules [10].
Damage to clothes by washing powders [9].

In all these of tasks there is a predefined set of classes of patterns which might be presented, and the task is to classify a future pattern as one of these classes. Such tasks are called *classification* or *supervised pattern recognition*[1]. Clearly

[1] The conference proceedings [14] provides a good view of pattern recognition applications prior to the current upsurge of neural networks.

someone had to determine the classes in the first place, and seeking groupings of patterns is called *cluster analysis* or *unsupervised* pattern recognition.

The patterns themselves can be of many different forms, as our list of examples shows (although images are a common source). In the parlance of the field, patterns are made up of *features*, which are measurements used as inputs to the classification system. For example, in the crabs task the features were four size measurements of the crab's carapace, plus its body depth. Where the tasks are images, the major part of the design of a pattern recognition system is to select suitable features; choosing the right features can be very much more important than what is done with them subsequently.

Cluster analysis has been applied to the pattern recognition field itself. One grouping which is widely recognized is into *'statistical'* and *structural* or *syntactic* pattern recognition. (The term 'statistical' comes from engineering, not from statisticians.) In statistical pattern recognition very little is assumed about the classes of patterns, all the information being *learned from examples*. In structural pattern recognition, qualitative information about the classes is used to structure the problem; in syntactic pattern recognition that information is provide by the grammar of a formal language. There are very few working examples of structural pattern recognition systems; an interesting demonstration system was provided by Chou [12] to convert typeset mathematical formulae into a typesetting language.

What is a neural network?

This is a much harder question to answer that might appear, since there a number of methods published in the neural networks literature for classification which seem to me to be neither neural nor networks, in that they have no biological motivation, and information is not just encoded in the connection strengths. One, the *probabilistic neural network* of [35, 36], causes considerable confusion, since it is also not probabilistic[2]. Kohonen's *learning vector quantization* (LVQ) methods [18, 19, 20] are commonly regarded as neural networks and have parameters called 'weights', yet these are not connection strengths, and the methods are much more closely related to classical *k-nearest neighbour methods.* (These connections are discussed in [34].)

To confine this chapter to a reasonable length, we will confine attention to feed-forward neural networks (also known as multilayer perceptrons). It would have been equally easy to use radial basis functions, since all we require is a flexible family of functions mapping multiple inputs to multiple outputs.

Learning from examples

Popular accounts of neural networks often stress their ability to learn from examples. For example

---

[2] It is an unacknowledged re-naming of the classical technique of kernel discriminant analysis; see for example [17]. To add yet more confusion, the 'probabilistic neural network' of [38] is a different classical statistical method, the use of mixtures of Gaussian distributions. Chou & Chen [13] call that a 'new fast algorithm for the effective training of neural classifiers'!

> Harmful emissions could become a thing of the past thanks to a computerised "car mechanic" that is being developed at Aston University's Neural Computing Research Group. ... Neural networks are computer programs with a brain-like ability to learn by example to solve problems. Such a system could be used in a car to optimise power and reduce emissions by adjusting ignition timing to match the circumstances such as, load, temperature, humidity and speed.
> (*Sunday Times*, 16 July 1995, page 2-9)

However, neural networks are only the latest of a long line of methods going back 200 years which 'learn from examples'. There have been many developments in the fields of statistics and statistical pattern recognition. There is a subfield of artificial intelligence research called *machine learning* which is concerned with learning logical concepts from examples. Consider the following anecdote which I heard from Professor Jim Alty:

> JA was visiting Cairo, and took a taxi with an English-speaking driver.
> JA noticed that the taxi driver did not stop at red traffic lights.
> Later, JA noticed that the driver *did* sometimes stop at red traffic lights, and that these traffic lights were also manned by policeman.
> This spawned a new hypothesis:
>
> > 'Taxi drivers in Cairo only stop at those red traffic lights which are manned by a policeman.'
>
> Later still, the taxi drove through a red traffic light which *was* manned by a policeman.
> Now JA can not fathom out the rule, and asks the driver for an explanation.
>
> > 'Ah, but that is obvious. He is my brother.'

This was given as an example of a task for a machine learning system. It shows some new features:

(i) It includes 'learning by query'. Human learning allows us to ask for explanations, to ask for the classes of examples we construct (a classic paradigm of science) and to ask for counter-examples to current hypotheses [3].

(ii) The induced rule has considerable *generalization* — to all taxi drivers from one, for example. (Opinion polls generalize from a sample to the population of voters.)

(iii) Explanation is important — a 'black box' solution such as a neural network would be unacceptable.

(iv) A precise rule may need unobserved information.

The third point can be very important, for example in getting a medical diagnosis system to be accepted by medical practitioners. We found that the zoologists involved in the project on Tsetse fly habitat adopted a classification tree [29] rather than a neural network solution even though the latter was more accurate, because they could interpret the classification tree. Many classification tasks have an element of *discrimination*, describing the distinguishing features of the classes. Neural networks are not usually suitable for such tasks.

A medical diagnosis example

We will illustrate pattern recognition by data on diagnostic tests on patients with Cushing's syndrome, a hyper-sensitive disorder associated with over-secretion of cortisol by the adrenal gland, taken from [2]. This dataset has three recognised types of the syndrome represented as a, b, c in Figure 1. (These encode 'adenoma', 'bilateral hyperplasia' and 'carcinoma'.)

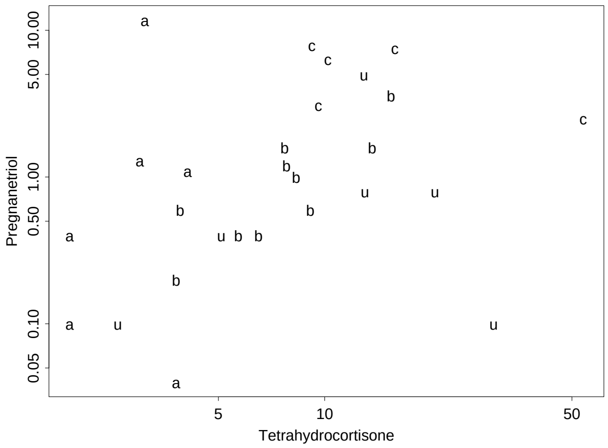

FIG. 1. Results of two diagnostic tests on patients with Cushing's syndrome.

The observations are urinary excretion rates (mg/24h) of the tetrahydrocortisone and pregnanetriol, and are considered on log scale. One of the patients of unknown type (marked u) was later found to be of a fourth type of the syndrome, and another was measured faultily. Such *outliers* are a common part of a pattern recognition task, and should be reported as outliers, not assigned to one of the classes.

This is a very small example, with few examples and only two features. It was chosen because it is easy to illustrate. Yet it is not atypical, since other tasks have many more examples and many more features, so the examples are often equally sparse in the feature space.

2 Statistical Pattern Recognition

We now formalize the pattern recognition task. The examples belong to one of K classes, and the allowed decisions are to declare an example to belong to one of these classes or 'doubt' $\mathcal{D}$ or 'outlier' $\mathcal{O}$. The motivation behind the 'doubt' decision is to allow difficult examples to be passed to a more expensive classifier (say a human expert), possibly after making further measurements. This can be a very helpful strategy, to allow a machine to deal with routine cases and let the experts deal with those where their expertise is required. A Zip-code reader will inevitably have difficulty with some people's writing, but a human may be able to understand the words in the overall context, especially if (s)he knows that Mrs Jones lives at 2034 Sunset Boulevard.

The *training set* $\mathcal{T}$ is a set of n correctly classified examples; future examples present a pattern $\mathbf{X}$ for a decision about its class. The training set is assumed to be a random sample from the same population as future examples.

We need to assign costs C_{jk} to making the wrong decision k when j is the true class, and this should be done carefully, as different mistakes can have quite different costs. (For example, the cost of refusing credit to a shopper who would have been a good payer is quite different to the cost of giving credit to a poor payer.) Decision theory assumes[3] that everything can be reduced to costs (or *utilities*) and that rational decision making will take the decision which minimizes the expected cost. Not everyone will be happy with this approach, not least the patient who is mis-diagnosed; the goal is long-run performance, not to do the best for each individual.

The rule which minimizes the expected cost of the decision is called the *Bayes rule.* It minimizes

$$\sum_j C_{jk}\, p(j \mid \mathbf{x}).$$

over decisions k (one of the classes or doubt $\mathcal{D}$). The *posterior probabilities* $p(k \mid \mathbf{x})$ are central to the theory; they give the probability that an example with features $\mathbf{x}$ has true class k. On the other hand, the *prior probabilities* π_k describe abundance of the classes in the whole population, before the features are given.

A very common choice of costs is $C_{jj} = 0$, $C_{jk} = 1$ for definite errors, and $d < 1$ for declaring 'doubt'. Then Bayes rule is to declare the action

$$c(\mathbf{x}) = \begin{cases} k & \text{if } p(k \mid \mathbf{x}) = \max_j p(j \mid \mathbf{x}) > 1 - d \\ \mathcal{D} & \text{if } \max_j p(j \mid \mathbf{x}) \leqslant 1 - d \end{cases}.$$

If two classes both have the highest posterior probability, either can be chosen (as the expected cost is the same under either choice).

If the posterior probabilities were known, there would be nothing more to be said. They are almost always unknown, and the task is to learn them from the training set. There are two approaches to learning probabilities.

Sampling paradigm

In this approach we describe the observations from examples of class k, for example by a Gaussian distribution. Let $p_k(\mathbf{x})$ denote the probability density

[3] but justifies this assumption by deriving it from slightly simpler axioms.

function[4] of examples from class k. Bayes' formula gives

$$p(k \mid \mathbf{x}) = \frac{p_k(\mathbf{x})\pi_k}{\sum_j p_j(\mathbf{x})\pi_j}$$

so the rule is to pick the class which maximizes $p_k(\mathbf{x})\pi_k$. Most of these quantities will depend on parameters (such as the mean and variance of a Gaussian distribution) and these are estimated from the training set. Figure 2 shows a classifier for the Cushing's syndrome task based on Gaussian distributions for each of the three classes.

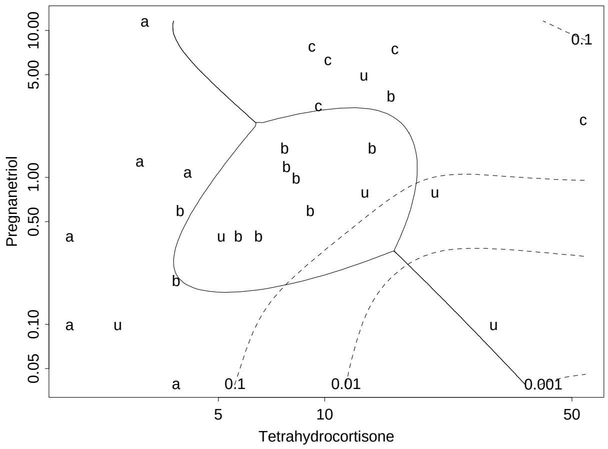

FIG. 2. A classification rule for the Cushing's syndrome task. The decision boundaries are shown; the left, central and right regions are classified as a, b and c respectively. Shown dashed are contours of $p(\mathbf{x})$ relative to the average on the training set.

This has been the main approach in statistics and pattern recognition. Notice that we have to learn more than we need from the examples, as Figure 3 shows. We end up knowing the joint distribution of features and classes, with density

$$p(\mathbf{x}, k) = p(k \mid \mathbf{x})p(\mathbf{x}) = p_k(\mathbf{x})\pi_k$$

where $p(\mathbf{x}) = \sum_k \pi_k p_k(\mathbf{x})$ is the density of the distribution of the features over all examples irrespective of class.

In the Cushing's syndrome task we also need to consider potential outliers, so contours of $p(\mathbf{x})$ are shown relative to the average value for the training set. Thus two of the unclassified examples have $p(\mathbf{x})$ less than 10% of the average for the training set, one much less than 1%. Using low values of $p(\mathbf{x})$ to declare outliers is reasonable if outliers can be thought to occur uniformly over the figure.

[4] For examples from class k, the probability of the observed features lying in a small set of volume A centred on $\mathbf{x}$ is about $p_k(\mathbf{x})A$.

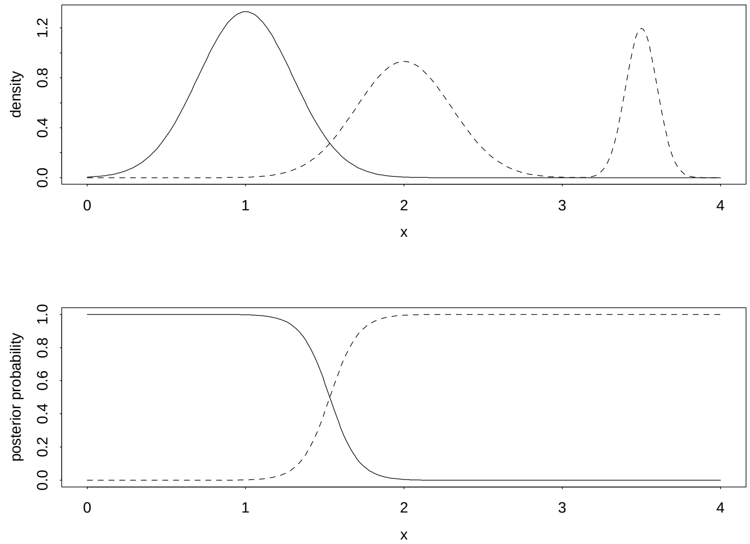

FIG. 3. It is not always necessary to model the class-conditional densities $p_k(\mathbf{x})$ (upper figure) accurately, as the posterior probabilities $p(k \mid \mathbf{x})$ in the lower figure are effectively unchanged by most aspects of modelling the right peak of the density shown dashed. Only the densities in the interval $[1,2]$ matter.

Diagnostic paradigm

In this approach we learn $p(k \mid \mathbf{x})$ directly, by looking at examples with similar $\mathbf{x}$ to the new example, and using the distribution of classes amongst those examples to predict $p(k \mid \mathbf{x})$. The simplest such approach is that of *k-nearest neighbours*, in which the k closest examples in the training set are found, and the distribution of their classes provides the estimate of $p(k \mid \mathbf{x})$.

In the diagnostic paradigm it is usual *not* to estimate $p(\mathbf{x})$, as this is not needed for classification. Something of the sort is however needed to detect outliers, which are normally ignored in the diagnostic paradigm. This can be very dangerous, not least because high posterior probabilities $p(k \mid \mathbf{x})$ can lead to unwarranted confidence in the classification rule.

The major approach in the diagnostic paradigm within statistics has been *logistic discrimination*. First consider just two classes. Then the log-odds on class 2 are

$$\operatorname{logit} p(2 \mid \mathbf{x}) = \log \frac{p(2 \mid \mathbf{x})}{p(1 \mid \mathbf{x})}$$

for the logit function $\operatorname{logit} p = p/(1-p)$ whose inverse is the logistic function $\ell(x) = e^x/(1+e^x)$. Now suppose the log-odds are a linear function of the features, so

$$\operatorname{logit} p(2 \mid \mathbf{x}) \quad = \quad \alpha + \beta^T \mathbf{x},$$

$$p(2 \mid \mathbf{x}) \;=\; \ell(\alpha + \boldsymbol{\beta}^T \mathbf{x}), \qquad p(1 \mid \mathbf{x}) = 1 - p(2 \mid \mathbf{x}). \tag{2.1}$$

This may start to look familiar; it is a feed-forward neural network with no hidden layer and just one output unit. Think of this in two parts, a linear function plus a logistic output stage. To achieve a more flexible model we can replace the linear function by a non-linear function $f(\mathbf{x}) = \operatorname{logit} \mathrm{p}(2 \mid \mathbf{x})$ and approximate that by a neural network with hidden layer(s). Although this *is* a feed-forward neural network, the occurrence of the logistic function in the output stage is coincidental. With a radial basis function network, we would still apply a logistic function to the output. Thus it is appropriate to think of a feed-forward neural network with linear output units as a flexible non-linear function, and then consider what to do with the outputs.

Now suppose we have more than two classes, which are not ordered in any way (such as the digits in Zip code recognition). The natural generalization is

$$f_k(\mathbf{x}) = \log p(k \mid \mathbf{x}) = \alpha_k + \boldsymbol{\beta}_k^T \mathbf{x}$$

known as a *log-linear model* or *multiple logistic regression.* Note that we can take $\alpha_1 = \boldsymbol{\beta}_1 = 0$ since only ratios of probabilities matter (since they sum to one). The probabilities are given by

$$p(k \mid \mathbf{x}) = \frac{\exp f_k(\mathbf{x})}{\sum_j \exp f_j(\mathbf{x})} \tag{2.2}$$

Once again we can replace the linear part by a non-linear function, this time a neural network with hidden layer(s) and linear output units. In the neural network literature this procedure is known as *softmax* [5] although it is has been known for many years.

Softmax is appropriate if we want to classify into one of K classes, and an example must belong to exactly one. A different problem sometimes occurs in medical diagnosis. We have K diseases, and the patient may have none, one or more diseases. The natural approach is to build a logistic model for each disease; this corresponds to a neural network with a separate logistic output unit for each class. This is much more widely used than softmax, even though it is rarely appropriate. The difference is often small, but can be significant when more than one class is given appreciable posterior probability.

If the K classes are ordered other methods (Mathieson, 1995) are more appropriate. Examples from our list in section 1 are the degree of risk in granting credit to a shopper, the amount of damage to clothes, and the degree of ellipticity of a galaxy. Softmax may be sufficiently accurate if the costs C_{jk} are chosen to reflect the structure in the classes.

3 Fitting the Neural Network

We have now seen that neural networks provide a natural generalization of logistic discrimination to estimate posterior probabilities $p(k \mid \mathbf{x})$ within the diagnostic paradigm. How should the parameters $\mathbf{w}$ (the weights in neural network jargon) be chosen? We have a function $\mathbf{f}(\mathbf{x};\mathbf{w}) = (f_k(\mathbf{x};\mathbf{w}))$, the fitted neural network, that determines the posterior probabilities by (2.1) or (2.2).

The most common answer is to choose $\mathbf{w}$ to maximize the log-likelihood, which is the sum of $\log p(k \mid \mathbf{x};\mathbf{w})$ over the examples of the training set (with $\mathbf{x}$ the observed features and k the given class for the example). This is negative, and it is equivalent to minimizing

$$E(\mathbf{w}) = \sum_{\mathcal{T}} -\log p(k \mid \mathbf{x})$$

This is sometimes known as logarithmic scoring, since it penalizes the procedure by the logarithm of the probability of the event which occurred. Small probabilities lead to large surprise and hence a large penalty. Maximizing the likelihood has been suggested many times from several different viewpoints in the neural networks literature, but has been used for decades in the logistic discrimination literature.

Maximum likelihood is the commonest theoretical answer, but not the most commonly used method, which is to fit the posterior probabilities by least squares, for example for two classes to minimize

$$E(\mathbf{w}) = \sum_{\mathcal{T}} \left[y - p(2 \mid \mathbf{x})\right]^2$$

where y is one if class two was the true class, otherwise zero. This has no merits except widely-available software.

There are more complicated but better answers. The approach so far adjusts the weights $\mathbf{w}$ of the network to fit the posterior probabilities to the training set $\mathcal{T}$, then acts as if $p(k \mid \mathbf{x};\mathbf{w})$ are the true posterior probabilities. The more flexible the function $\mathbf{f}(\mathbf{x};\mathbf{w})$ (for example with more hidden units) the better we will be able to fit to the training set, but not necessarily to the true posterior probabilities. Thus we will normally find that as we increase the number of hidden units, the performance on future examples at first improves and then becomes steadily worse. This can be overcome within the *predictive approach* [2, 32, 34] which takes the uncertainty in the fitted weights into account, and is discussed in the chapter by Bishop in this volume. (This was used for Figure 2.)

An easier (but less theoretically satisfying) way to avoid over-fitting the training set is known as *regularization* (see, for example, [27]). This penalizes 'wiggly' functions $\mathbf{f}(\mathbf{x};\mathbf{w})$, often for neural networks by adding λ times the sum of the squared weights to $E(\mathbf{w})$. We can see the effect of this on a synthetic example from [30]. There are 125 examples from each of two classes. Since the example is synthetic, all the probabilities are known, hence the Bayes rule; it make 8.0% errors. As Figure 4 shows, the (linear) logistic discriminant is inadequate. The

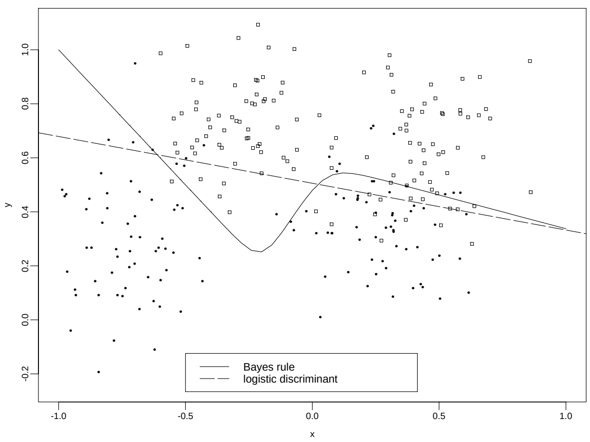

FIG. 4. A synthetic example with classes marked by open squares and filled circles.

neural networks come quite close to the Bayes rule:

Bayes rule	8.0%
Logistic discrimination	11.4%
Neural network with 3 hidden units	11.1%
ditto with weight decay	9.4%
ditto with 6 hidden units	9.5%

Note the effect of weight decay in Figure 5. When weight decay is used adding more units makes relatively little difference to the solution.

One difficulty which is often overlooked is that the algorithms to fit neural networks only find a local minimum of the fit criterion. There will normally be several different local minima[5], even when weight decay is used, as Figure 6 shows. Even if we can afford the CPU time to find many local minima, it is far from clear how we should choose between them, as the best fit on the training set often does not work best to predict future examples. If more CPU time can be used, the different local minima can be combined in some way. Theory ([34], §2.6) suggests the optimum way is to average the posterior probabilities, that is to use

$$p(k \mid \mathbf{x}) = \sum_m \alpha_m p(k \mid \mathbf{x}; \mathbf{w}_m)$$

where the $\mathbf{w}_m$ are the weights corresponding to different local minima, and the

[5] All the examples presented here used a general-purpose optimization algorithm to fit the weights; in my experience this works a great deal faster and more reliably than the on-line modifications of gradient descent which are commonly described in the neural network literature. Except for Figure 6, the first solution found was used.

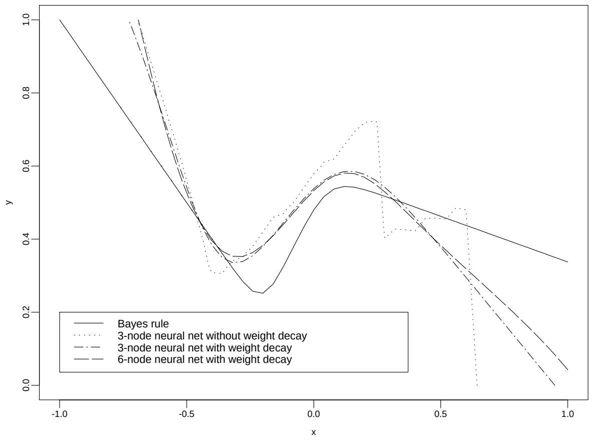

FIG. 5. Neural network rules for the synthetic example.

(α_m) form a convex combination[6]. Such ideas go back at least to Stone (1974), but have been widely suggested more recently, for example by [22, 6, 39, 40, 26]. The (α_m) may be chosen by the methods below, but often taking equal weights over the local minima is better than selecting any one. Note that this averaging procedure needs more CPU time both to train the network and to predict future examples, but is trivial to parallelize if multiple CPUs are available.

Network complexity

We do have to choose the complexity of the fitted neural network, which is controlled both by the number of hidden units and the weight decay parameter λ. In fact the number of hidden units (provided it is enough) is not very important in the presence of weight decay as Figures 5 and 7 show. In fact the value of $\lambda = 0.01$ used was chosen from some Bayesian heuristics [31] and is often suitable for classification problems if the inputs are roughly scaled to the interval $[0, 1]$. More sophisticated methods of choosing complexity are the subject of current research [33].

The other way to choose complexity is based on estimating future performance. Suppose we have another set of classified examples $\mathcal{V}$ called the *validation set*. We then fit many different networks, evaluate their performance on $\mathcal{V}$, and choose the best. Finally, yet another set of examples, the *test set*, can be used to measure the performance of the chosen network. (These terms are commonly muddled in the neural networks literature.)

Quite often (as in the examples here) we do not have enough examples to keep separate training, validation and test sets. We can squeeze more out of small sets of examples by *cross-validation* [37]. This divides the training set $\mathcal{T}$ into V

[6] Each $\alpha_m \geqslant 0$ and $\sum \alpha_m = 1$.

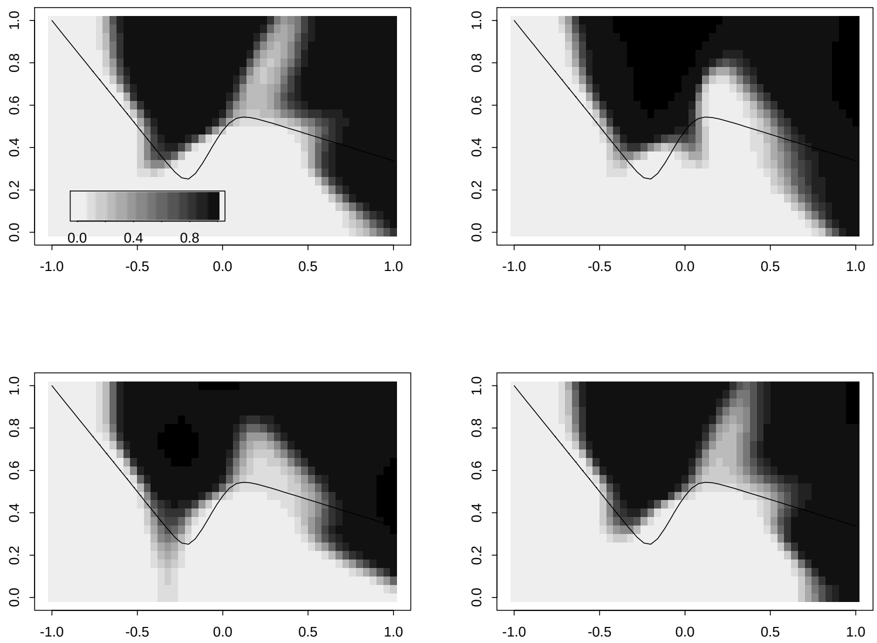

FIG. 6. Four solutions for the synthetic example based on a neural network with six hidden units. Dark values indicate a high posterior probability for class two, shown by open squares on earlier figures.

parts. Then each part in turn is kept as the validation set, and the network(s) are fitted to the remaining examples and used to predict those in the validation set. When this has been done V times, every example has been predicted, and the performance of the network(s) computed. There are two common choices of V. One is to leave each example out at a time, and train the network(s) n times. This is slow, and not particularly desirable theoretically. It is better to take V as 5 or 10.

Cross-validation validates performance. The same procedure can be used to test the performance of the chosen classifier, but note that we need to cross-validate the whole procedure, including the choice of network. This often results in cross-validation within cross-validation, and V^2 fits.

4 Forensic glass

Our main example comes from forensic testing of glass collected by B. German on 214 fragments of glass, and taken from [25]. Each case has a measured refractive index and composition (weight percent of oxides of Na, Mg, Al, Si, K, Ca, Ba and Fe). The fragments were originally classed as seven types, one of which was

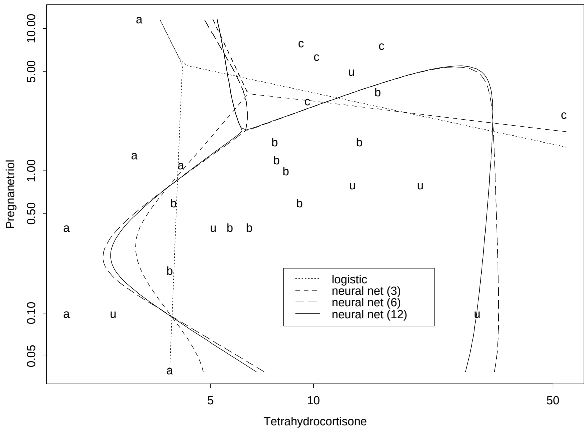

FIG. 7. Neural network rules for the Cushing's syndrome task, with 3, 6 and 12 hidden units, fitted with weight decay. The logistic discriminant is also shown.

absent in this dataset. The categories which occur are window float glass (70), window non-float glass (76), vehicle window glass (17), containers (13), tableware (9) and vehicle headlamps (29). The composition sums to around 100%; what is not anything else is sand.

This example is really too small to divide, so methods have been assessed by 10-fold cross-validation. Linear logistic discrimination was assessed as having an error rate of about 36.0%, whereas choosing the class of the closest match had an error rate of about 23.4%. To choose a neural network, we took a different 10-fold cross-validation split, and averaged the predictions over all the local minima which were found. The cross-validated error rates were

	#(hidden units)		
λ	2	4	8
0.0001	30.8	23.8	27.1
0.001	30.4	26.2	26.2
0.01	31.8	29.9	29.9

where λ is the weight-decay parameter. This suggests that we choose 4 hidden units and $\lambda = 10^{-4}$, and this was assessed to have an error rate of 23.8% on the cross-validated test set.

This is a salutary example. After expending about 2 hours' CPU time on a Sparc 20/612, we have achieved comparable performance to the simple rule of quoting the class of the nearest example in the training set, and both are 'black boxes' with no explanatory power. This is not an unusual conclusion, as *used well* nearest-neighbour methods are often amongst the best performers. In

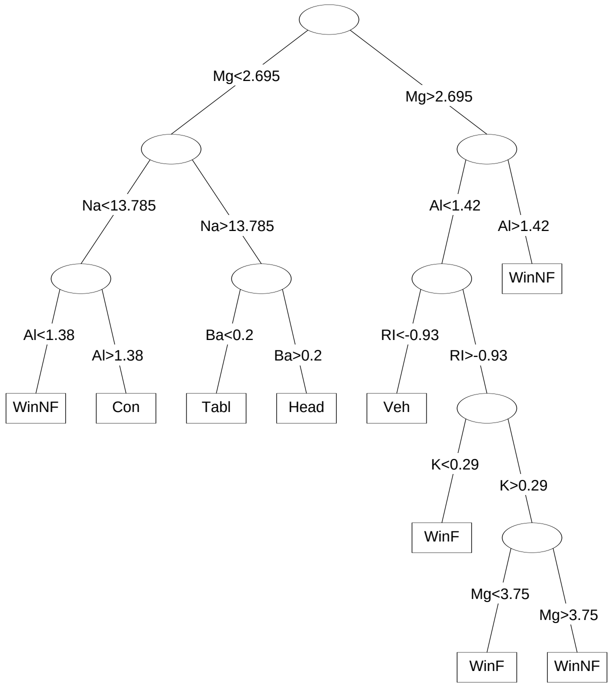

FIG. 8. Classification tree for the forensic glass task.

contrast, the classification tree shown in Figure 8 had an assessed error rate of 32.2%, but can be interpreted rather easily.

5 Further Reading

The classic book on pattern recognition is Duda & Hart [15]; although over twenty years old is still contains much relevant material. There have been surprisingly few more recent books, but Fukunaga [16] updates some of the material. Books which include a treatment of pattern recognition using neural networks are just beginning to appear. Bishop [4] is principally on mainstream neural network methods for classification, whereas Ripley [34] considers neural networks as one of many tools for pattern recognition. The collections edited by Cherkassky *et al.* [11] and Michie *et al.* [24] contain some excellent overview articles, and the second also contains a careful comparative study on (mainly) real problems.

Bibliography

1. Advisory Council on Science and Technology (1992) *Artificial Neural Networks.* Her Majesty's Stationary Office.
2. Aitchison, J. and Dunsmore, I. R. (1975) *Statistical Prediction Analysis.* Cambridge University Press, Cambridge.
3. Angulin, D. (1993) Learning with queries. In *Computational Learning and*

Cognition ed. E. B. Baum, pp. 1–28. SIAM, Philadelphia.

4. Bishop, C. (1995) *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford.
5. Bridle, J. S. (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neuro-computing: Algorithms, Architectures and Applications* eds F. Fogleman Soulié and J. Hérault, pp. 227–236. Springer, Berlin.
6. Bridle, J. S. and Cox, S. J. (1991) RecNorm: simultaneous normalisation and classification applied to speech recognition. In *Advances in Neural Information Processing Systems 3,* eds R. P. Lippmann, J. E. Moody and D. S. Touretzky, pp. 234–240. Morgan Kaufmann, San Mateo, CA.
7. Campbell, N. A. and Mahon, R. J. (1974) A multivariate study of variation in two species of rock crab of genus *Leptograpsus. Australian Journal of Zoology.* **22**, 417–425.
8. Candela, G. T. and Chellappa, R. (1993) Comparative performance of classification methods for fingerprints. US National Institute of Standards and Technology report NISTIR 5163.
9. Carstensen, J. M. (1992) *Description and Simulation of Visual Texture.* Ph.D. thesis 59, Institute of Mathematical Statistics and Operations Research (IMSOR), Technical University of Denmark, Lyngby.
10. Carter, C. and Catlett, J. (1987) Assessing credit card applications using machine learning. *IEEE Expert* **2(3)**, 71–79.
11. Cherkassky, V., Friedman, J. H. and Wechsler, H. (eds) (1994) *From Statistics to Neural Networks. Theory and Pattern Recognition Applications.* Springer, Berlin.
12. Chou, P. A. (1989) Recognition of equations using a two-dimensional stochastic context-free grammar. In *Visual Communications and Image Processing IV* ed. W. A. Pearlman. SPIE Proceedings Series **1199**, 852–863.
13. Chou, W.-S. and Chen, Y.-C. (1992) A new fast algorithm for the effective training of neural classifiers. *Pattern Recognition* **25**, 423–429.
14. Devijver, P. A. and Kittler, J. V. (eds) (1987) *Pattern Recognition Theory and Applications.* Springer, Berlin.
15. Duda, R. O. and Hart, P. E. (1973) *Pattern Classification and Scene Analysis.* Wiley, New York.
16. Fukunaga, K. (1990) *Introduction to Statistical Pattern Recognition.* Second edition. Academic Press, Boston.
17. Hand, D. J. (1982) *Kernel Discriminant Analysis.* Research Studies Press, Chichester.
18. Kohonen, T. (1988) Learning vector quantization. *Neural Networks* **1**(suppl 1), 303.
19. Kohonen, T. (1990) The self-organizing map. *Proceedings of the IEEE* **78**, 1464–1480.

20. Kohonen, T. (1995) *Self-Organizing Maps.* Springer, Berlin.
21. Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D. (1989) Backpropagation applied to handwritten zip code recognition. *Neural Computation* **1**, 541–551.
22. Lincoln, W. P. and Skrzypek, J. (1990) Synergy of clustering multiple backpropagation networks. In *Advances in Neural Information Processing Systems 2,* ed. D. S. Touretzky, pp. 650–657. Morgan Kaufmann, San Mateo, CA.
23. Mathieson, M. J. (1995) Ordinal models for neural networks. In *Neural Networks in Financial Engineering.* eds A. P. Refenes, Y. Abu-Mostafa and J. Moody. World Scientific, Singapore.
24. Michie, D., Spiegelhalter, D. J. and Taylor, C. C. (eds) (1994) *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, New York.
25. Murphy, P. M., and Aha, D. W. (1995). *UCI Repository of Machine Learning Databases* [Machine-readable data repository]. Irvine, CA: University of California, Dept. of Information and Computer Science. Available by anonymous ftp from `ics.uci.edu` in directory `pub/machine-learning-databases`.
26. Perrone, M. P. and Cooper, L. N. (1993) When networks disagree: Ensemble methods for hybrid neural networks. In *Artificial Neural Networks for Speech and Vision* ed. R. J. Mammone, pp. 126–142. Chapman & Hall, London.
27. Poggio, T. and Girosi, F. (1990) Networks for approximation and learning. *Proceedings of the IEEE* **78**, 1481–1497.
28. Refenes, A.-P. (1995) *Neural Networks in the Capital Markets.* Wiley, Chichester.
29. Ripley, B. D. (1993) Statistical aspects of neural networks. In *Networks and Chaos – Statistical and Probabilistic Aspects* eds O. E. Barndorff-Nielsen, J. L. Jensen and W. S. Kendall, pp. 40–123. Chapman & Hall, London.
30. Ripley, B. D. (1994) Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society series B* **56**, 409–456.
31. Ripley, B. D. (1994) Neural networks and flexible regression and discrimination. In *Statistics and Images 2* ed. K. V. Mardia. *Advances in Applied Statistics* **2**, pp. 39–57. Carfax, Abingdon.
32. Ripley, B. D. (1994) Flexible non-linear approaches to classification. In Cherkassky *et al.* (1994), pp. 105–126.
33. Ripley, B. D. (1995) Statistical ideas for selecting network architectures. In *Neural Networks: Artificial Intelligence and Industrial Applications*, eds B. Kappen & S. Gielen, pp. 183–190. Springer, London.
34. Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge.
35. Specht D. F. (1990a) Probabilistic neural networks. *Neural Networks* **3**,

109–118.

36. Specht D. F. (1990b) Probabilistic neural networks and the polynomial Adaline as complementary techniques for classification. *IEEE Transactions on Neural Networks* **1**, 111–121.
37. Stone, M. (1974) Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society series B* **36**, 111–147.
38. Streit, R. L. and Luginbuhl, T. E. (1994) Maximum likelihood training of probabilistic neural networks. *IEEE Transactions on Neural Networks* **5**, 764–783.
39. Wolpert, D. H. (1992) Stacked generalization. *Neural Networks* **5**, 241–259.
40. Xu, L., Kryzak, A. and Suen, C. Y. (1992) Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics* **22**, 418–435.