Learned from Neural Networks

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

Abstract

Many problems in data analysis and pattern recognition may be attacked by neural networks. Sometimes this approach is better, sometimes it is worse than the use of alternatives. A general discussion is presented on possibilities, advantages and disadvantages of their use in comparison with more specific approaches. The study of neural networks, once almost a 'black box' has given a better understanding of the possibilities of pattern learning. The recent development of more specific methods has been strongly stimulated by this knowledge.

1. Introduction

Neural networks are now used for 10 to 15 years for adaptive data analysis. At their start they caused a large hype, attracting many new researchers to the field, fascinated by the new possibilities. For a number of scientists they are still the only way to do such an analysis. Some even simply call any approach in this field a neural network.

Much has been learned about neural networks in particular and about machine learning in general on the basis of a more than ten year world wide effort in experimenting, evaluating and analyzing neural networks. It is possible now, not only to look back and judge the early promises and claims, but moreover, to reformulate and explain the properties of neural network in present terms. To that end we will discuss a number of characteristic issues for neural networks that increased the understanding of learning in general. We will do this using the pattern recognition terminology as neural networks are well suited for finding and describing pattern classes in feature spaces.

This will not be an introductory paper to the field. It is assumed that the reader is generally familiar with the various approaches, training procedures and problems related to the neural networks. We will restrict ourselves to a global, verbal analysis, and to comments and opinions. In the context of this paper it is not possible to present very precise analyses end discussions. The author is aware of the fact that on some places more careful formulations and argumentations are desired for a more strict scientific discussion.

2. Architectures

As indicated above, there is no universally accepted definition of an artificial neural network. We will restrict to systems that are constituted by a possibly large set of identical simple building stones: neurons. Such a neuron has a set of inputs, a state vector (that may be independent of its actual input) and one or more outputs (dependent on state and input values). Neurons are usually connected: inputs may be fed by external measurements as well as by outputs of other neurons.

Neural networks can be flat: just a single layer of connected neurons which outputs is identical to the output of a 'winning' neuron. They also may be organized in a hierarchical way: a series of so called 'hidden' layers containing so called 'hidden neurons' and on top a visible layer of 'output' neurons.

The input of a neural network can be (a subset of) the raw measurement data or some feature representation of it. Two typical choices often made for the state of a neuron are a set of weights for their inputs (features) and/or a feature vector that may be interpreted as a position of the neuron in feature space. Three examples of neural network architectures are:

- The Feed-Forward Neural Network (FFNN) or multi-layer perceptron. There is at least one hidden layer. All neurons have a single sigmoid output. In classification applications its outputs may be interpreted as class confidences or posterior probabilities, provided the network is well trained.
- The Radial Basis Neural Network (RBNN) has a first hidden layer of neurons of which the outputs are radial basis functions of the inputs. This can be an exponential function (e.g. the Gaussian function) of the distance between the input and a 'position' of the neuron in the feature space. The neurons in the next layer may weight all the contributions of the radial basis functions. In total this can be interpreted as a weighted (not necessarily normalized) sum of Gaussians.
- The Self Organizing Map (SOM) consists also of a set of neurons 'positioned' in feature space. During training they are now connected in a low dimensional grid (1, 2 or possibly 3-dimensional). During test-

ing the 'winner takes all' strategy is used, i.e. the output of the network is determined by the neuron closest to the input.

In our discussion we will mainly refer to the FFNN as it is the most widely used architecture.

3. Training versus Implementing

There are many ways to estimate parameters (weights) for a given architecture. The architecture of the FFNN is so general that almost any function can be implemented by it. This is one of the basic statements in the neural network area [6]. Many papers have been devoted to the way a specific non-neural method can be implemented on a neural network, see [7] for an example.

In order to give the phrase 'neural network' a specific meaning we should specify both, the architecture as well the way its parameters are estimated. Therefore, neural network training has to be different from just estimating a function or a classifier in one way and mapping it on a neural network. Usually a true neural network training procedure is based on an iterative approximation in which the parameters are successively updated in numerous steps. Such a step can be based on a single data item, on a set of them, or an all available data points. In each step the desired outcome is compared with the actual one and, using the knowledge of the architecture, all parameters are changed slightly such that the error for the presented data points decreases.

Consequently, a neural network is defined by an architecture based on many simple neurons *and* a training procedure in which these neurons are slowly adapted such that the overall performance increases. A number of problems related with this adaptation procedure will be discussed below. The main problem, however, can be stated here.

The parameters of the individual neurons are in a nonlinear way related to the performance of the entire network. Moreover, their influence is highly dependent. Optimization procedures are based on a simplified relationship: linear or quadratic approximation, steps computed for a subset of the data. This works if the steps are small. In fact, the larger the network and the more complex the data set, the smaller the steps should be in order to find a path in a good direction. Consequently, neural network training is very, very slow.

4. The Universal Approximator

In theory, almost any well-behaved function can be realized by a neural network of sufficient size [6]. For some functions this size may be large. This property, however, holds for the architecture. There is no guarantee that the parameters of such a system can be found by a training procedure based on examples. In practice, it appears to be very difficult or impossible to find strongly nonlinear networks describing the data, even if it is known to exist [13].

A serious problem caused by the property that sufficiently large network can implement any function, is that they are also able to follow the noise in the data. This prevents large networks to generalize. Instead they overtrain, as will be discussed section 6.

5. Nonlinearity

An important step made by the introduction of neural networks is the possibility of creating controlled, moderately nonlinear systems. The above discussed problem, that the theoretical possibility of implementing arbitrary functions cannot be realized in practice, is in fact an advantage. In many applications, the way the data is given by the sensors and by the measurements (features) obtained from the sensor data is such that it has some meaning to the experimentator. Consequently, a natural metric for the data representation (e.g. Euclidean distances in the feature space) offers already a first approximation to the structure in the data. Linear descriptions may thereby yield a good performance. Moderately nonlinear descriptions may improve this where the natural representation is good but not optimal.

At this point it is good to realize that the common initialization of a FFNN by small weights generates an almost linear network. At the start of training this linear function adapts itself to the linear structures in the data. Next, when the weights increase, it gradually becomes nonlinear. If training is proceeded too far, the weights may become large and the network becomes strongly nonlinear, see section 6.

If a highly nonlinear description fits, however, this indicates that a completely different representation than the original data description may be better. We are then in the area where the experimenter has no idea what is going on in the data (his representation is wrong) and all knowledge has to be found from a statistical analysis of the data.

We believe that such a desperate attempt should only be made in the very rare situation that there is no good prior knowledge, but one is convinced that there is something hidden in the data. Sufficient resources should be available to collect a large dataset and to perform an extensive nonlinear analysis. For the large majority of applications a moderately nonlinear approach is just perfect: it finds in the data a small improvement over the available knowledge.

6. Training and Overtraining

A neural network is trained in order to establish a fit to a set of examples, i.e. the network should output values close to the target values assigned by an expert-teacher to the training set of objects. Well

trained networks generalize in the sense that their outputs are (close to) correct for objects not available in the training set.

If the training set is noisy or contains errors (like overlapping classes in case of a classification problem), a too heavily trained network may be adapted to the noise and does not generalize. It is overtrained and produces random outputs for unseen input objects.

Overtraining can be detected during training by the use of a test set. The disadvantage is that such a set reduces the size of the training set and thereby decreases the final performance. A good trick is to rotate (parts of) the training set and the test set. Overtraining is sometimes related to strong nonlinearity. This can be detected by the size of the weights. Only for large weights the network is able to have a strong nonlinear functionality.

Overtraining may be avoided by:

- Early stopping, i.e. stopping the training in time, before an adaptation to the noise starts. It is difficult to detect an optimal stopping moment without the use of a test set.
- Small networks. If the network has much less parameters than objects in the training set, it cannot be overtrained.
- Pruning, i.e. reducing the size of a network after it is trained. In this way unimportant neurons, causing noise, are deleted. It is necessary to retrain the network after pruning.
- Large training sets. Networks of a fixed size cannot be overtrained for large training sets.
- Data enrichment, i.e. artificially enlarging the training set by objects that 'smooth out' the noise.
- Regularization. One way to do this is to add a penalty term to the optimization criterion for networks with large weights, as these are related to strong nonlinearity, section 5.

7. Redundancy

Neural networks are often larger than necessary. The final network, after training, may be mapped on a smaller system without a significant change in behavior. The question arises: wouldn't it be advantageous to scale down the networks to a size that just fits the problem?

It is interesting to note that this is not true. The performance of a neural network shows, as a function of its size an optimum for sizes larger than necessary. This is directly related to the difficulty to exploit the possible nonlinearity as discussed above. The conclusion that can be drawn is that some redundancy in a network is advantageous for good training.

This observation contradicts the traditional opinion in pattern recognition that redundant systems should be avoided in order to circumvent effects like the curse of dimensionality and the peaking phenomenon [8]. Neural networks have shown that a detour by oversized systems may be worthwhile.

8. Speed and memory

Neural networks are large systems. As a consequence they are slow and need a lot of memory. This holds for testing and in particular for training. The update of large sets of parameters by an iterative procedure based on all training samples is computationally intensive. For larger networks some optimization algorithms, e.g. the ones based on second derivatives, are even computationally prohibitive.

In the early days of neural networks, their speed was sometimes advocated as an advantage. This is based on the possibility of using parallel hardware. Many neurons having the same architecture make it possible to build cheap special devices that efficiently evaluate an entire network. It appears to be difficult, however, to integrate such devices with various training rules and to maintain the flexibility in training that is necessary for the application over a large set of problems. Moreover, it has to be realized that many other pattern recognition algorithms, like the Parzen classifier and the nearest neighbor rule, may be speeded up similarly by such devices.

9. What one needs to know

Retrieving knowledge from observations cannot start from nothing. What may be hoped is that existing knowledge grows from observations. The question arises: what prior knowledge is needed for what learning system? This may differ considerably over the various procedures for pattern learning. Bayes methods need knowledge over distributions and prior probabilities. Most traditional classification procedures prefer a small set of good features.

The type of knowledge needed for neural networks is somewhat different. As they adaptively adjust to moderately nonlinear data structures, probabilistic information is hard to use. Moreover, to some respect neural networks may combine feature reduction and classification. In a FFNN the input layer can be considered as a mapping to a subspace represented by the hidden layer. If the number of neurons in this layer is smaller than the number of inputs, this is a dimension reduction.

So what is needed to make a neural network successful? Let us reconsider the FFNN. To some respect the number of parameters in the input layer is not so important. What is essential for a good generalization, however, is the number of parameters in the output layer relative to the sample size. This output layer constitutes a linear classifier and determines directly the performance. Even if all weights below this layer are random, a good accuracy may be reached if the proportion in the order of 1:10. Here we see the rela-

tion between sample size and nonlinearity. More samples enable a larger hidden layer and thereby a larger nonlinearity.

What is needed for a neural network to be successful is that the sample size fits the demands of the problem in the above described way. The number of samples should allow for an architecture that can be adapted to the structure of the data. Moreover, everything, data and network and training procedure, should fit into the computer at hand. So the data representation, the data size and the network architecture together should match the data structure to be adapted.

In general, there is no guaranteed way to solve this. In practice one hopes that the data representation is sufficiently well to make the problem solvable (at most moderately nonlinear without introducing to much noise) by the architecture that is allowable for the given size of the training set.

10. What one may learn

A successfully trained neural network maps inputs on outputs such that these are close to the targets. As such it can be applied. In some applications one likes to achieve more. For instance, the actually important features, or an explanation for a certain outcome. This appears to be very difficult. All information is distributed over the neurons. Just by advanced experiments one may extract such information [14].

A well trained multi-layer network represents in its hidden layer a space from which the following layers construct the final output. If this space has a lower dimensionality than the input space (less neurons than inputs) then the neural network represents from input to hidden layer a useful nonlinear map on which also other classifiers or data analysis techniques may be applied.

An important property that is returned by a neural classifier is that such a system, trained for hard classifications, may still return confidence values that approximate for some problems the posterior probabilities [15]. This makes a neural networks suitable for integration with expert knowledge or for combining classifiers [16].

11. Are neural networks better?

A question that has been addressed may times is whether neural networks are any better than traditional techniques. This is in fact an unanswerable question as it relates two types of systems for which no objective evaluation exists [11]. This can be understood as follows.

Traditional pattern recognition systems usually have a small set of free parameters, like the number of neighbors in the k-NN rule. Such a parameter can be optimized over the training set. The performance of the classifier follows from a test set. On the whole, such an evaluation is objective as another data-analyst will find the same performance for the same classifier using the same data set.

A neural network has many user adjustable parameters, some determining its architecture, others are related to the training rule. It is very common that the data analyst chooses good values for these parameters on the basis of some initial experiments and his experience. Moreover, a neural network is randomly initialised. As a consequence, different data analysts, or even the same data analyst on different days, will find different networks and thereby different performances. What effectively is evaluated is not the neural network as such, but the skills of the data analyst in using the given neural network toolbox.

An evaluation over neural networks and traditional classifiers thereby compares analysts with fixed procedures. It is, of course, possible to freeze a neural network procedure by estimating its parameters from the data where possible and by making constant choices on other places. Such an automatic neural classifier can be compared objectively with a fixed traditional procedure. It is, however, not representative for the neural network possibilities as a whole.

12. Conclusions

What we have learned from neural networks is that it is possible to train large systems with many inputs on the basis of relatively small data sets. The resulting systems usually have a moderately nonlinear structure. These results are of significant scientific interest as they contradict the earlier state of knowledge.

Neural networks offer an interesting toolbox that may be powerful in the hands of an experienced data analyst. In theory they are more general and have potentially a better performance than any other learning system, as the result of any other system can be implemented on a neural network. In practice the training procedures for obtaining such results directly do not exist. In fact they cannot exist because a more general system will always perform less than an optimally chosen (using prior knowledge) specific system. This is reflected by the common saying that a neural network usually offers ‘the second best solution’.

Given the knowledge obtained from experiments and studies on neural networks one may aim at building better, more specific systems. Examples of new learning systems that have benefitted in this way are the technique of combining classifiers [5], the support vector machine [9] and the use of (non)linear subspaces [12].

13. References

[1] B.Cheng and D.M. Titterington, Neural Networks: A review from statistical perspective, *Statistical Science*, vol. 9, no. 1, 1994, 2-54.

[2] B.D. Ripley, Neural networks and related methods for classification, *Journal of the Royal Statistical Society B*, vol. 56, no. 3, 1994, 409-456.

[3] A.K. Jain, J. Mao, and K.M. Mohiuddin, Artificial Neural Networks: A Tutorial, *Computer*, vol. 29, no. 3, 1996, 31-44.

[4] J.C. Mao and A.K. Jain, Artificial neural networks for feature extraction and multivariate data projection, *IEEE Transactions on Neural Networks*, vol. 6, no. 2, 1995, 296-317.

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

[6] K. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks*, vol. 2, 1989, 183-192.

[7] I.K. Sethi, Neural implementation of tree classifiers, *IEEE Transactions on Systems Man and Cybernetics*, vol. 25, no. 8, 1995, 1243-1249.

[8] 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.

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

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

[11] R.P.W. Duin, A note on comparing classifiers, *Pattern Recognition Letters*, vol. 17, no. 5, 1996, 529-536.

[12] R.P.W. Duin, Classifiers in almost empty spaces, Proc. *ICPR2000*, 2000, accepted.

[13] D. de Ridder, R.P.W. Duin, P.W. Verbeek, and L.J. van Vliet, The applicability of neural networks to non-linear image processing, *Pattern Analysis and Applications*, vol. 2, no. 2, 1999, 111-128.

[14] M. Egmont-Petersen, J.L. Talmon, A. Hasman, and A.W. Ambergen, Assessing the importance of features for multi-layer perceptrons, *Neural Networks*, vol. 11, no. 4, 1998, 623-635

[15] P. Muller and D.R. Insua, Issues in Bayesian analysis of neural network models, *Neural Computation*, vol. 10, no. 3, 1998 Apr. 1, 749-770.

[16] S. Hashem, Optimal linear combinations of neural networks, *Neural Networks*, vol. 10, no. 4, 1997, 599-614.