Learning and Nonlinear Models (L&NLM) – Journal of the Brazilian Neural Network Society, Vol. 9, Iss.3, pp. 202-212, 2011.

An Overview on the Use of Neural Networks for Data Mining Tasks

Nelson F. F. Ebecken, D.Sc., nelson@ntt.ufrj.br COPPE/Federal University of Rio de Janeiro

Keywords: Neurocomputing, Neural Classifiers, Self-Organizing Maps, Nonlinear Classifiers, Time Series Data Mining

Abstract

This overview provides comments in the main aspects related to some of the most common neural network models in use today for data mining tasks. Despite of the different possibilities of learning paradigm, network architecture, network connectivity and learning algorithm the focus is concentrated in multilayer feed-forward networks with back propagation learning, radial basis function network and self organizing map network. The search for the more efficient models, the extraction of the hidden knowledge and the data mining application characteristics are examined.

Introduction

Neural networks have been successfully applied to solve data mining problems in several domains. An artificial neural network[1], often just called a neural network (NN), is a mathematical model or computational model inspired on biological neural networks, in other words, it is a simplified mathematical model from biological neuron model. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases a NN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. In practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Like most statistical models neural networks are capable of performing three major tasks: *regression*, *classification and clustering*. The performance of neural networks is measured by how well they can predict data not used during training. This is known as generalization. This issue of generalization is actually one of the major concerns when training neural networks. It is known as the tendency to *overfit* the training data accompanied by the difficulty in predicting new data. Cross-validation is a technique for assessing how the results of a statistical analysis will generalize to an independent data set[2]. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

The solution usually needs simultaneously train several network architectures and compute predictions from each. This may require significant computing times, but will automatically train and test a large number of different types of networks. In many cases an ensemble of models is necessary and voting approaches are often adopted: those that adaptively change the distribution of the training set based on the performance of previous models (as in boosting methods) and those that do not (as in Bagging)[3,4]. Using neural networks as a tool, data miners are extracting information and building predictive models from datasets. In this sense NN may achieve high accuracy, but the knowledge acquired by such neural networks is usually incomprehensible for humans. This fact can be a major obstacle in data mining applications, in which human-interpretable patterns describing the data, like symbolic rules or other forms of knowledge structure are important[5].

There are numerous publications that discuss neural networks and data mining[30,.38]. Generally they deal with particular aspects and specific applications. This paper intends to cover the following subjects:
- Which neural network models are commonly employed in the practice of data mining and should be known by data miners ?
- Which tools are most widely used commercially and academically and what resources are provided ?
- What needs to be improved in the current tools to meet the needs of data miners in practice ?

Multilayer perceptron

From 1958, when *Rosenblatt* introduced the *Perceptron,* to today, there has been a never-ending development of theoretical studies and applications of artificial neural systems. The perceptron is the simplest form of neural network and is used for the classification of a special type of patterns: linearly separable ones. The perceptron consists of a single neuron with adjustable synaptic weights and bias

(threshold), and it can be shown that if the patterns (vectors) used to train the perceptron are drawn from linearly separable classes, then the perceptron algorithm converges. The proof of convergence of the algorithm is known as the perceptron convergence theorem. One of the most important events in the field of NN was the introduction of the *Multi-Layer Perceptron* (MLP) structure in 1986, which was able to perform hard nonlinear mappings.

MLPs represent the most commonly used and extensively studied class of NNs in classification, implementing a feedforward, supervised and hetero-associative paradigm[1]. MLPs consist of different layers where the information flows only from one layer to the next. Layers between the input and output layer are called hidden layers. The input units play no active role in processing the information flow because they simply distribute the signals to the units of the first hidden layer. All hidden units work in an similar way, and the output units are simpler versions of the hidden units. In an MLP, each hidden unit transforms the signals from the former layer to one output signal, which is distributed to the next layer. Each hidden unit has an activation function, which is generally nonlinear. The activation function is in general a sigmoid or tanh function and is the same for all hidden units. The output of a hidden unit is determined by the weighted sum of the signals from the former layer. Figure 1 gives an example of a MLP with one layer topology.

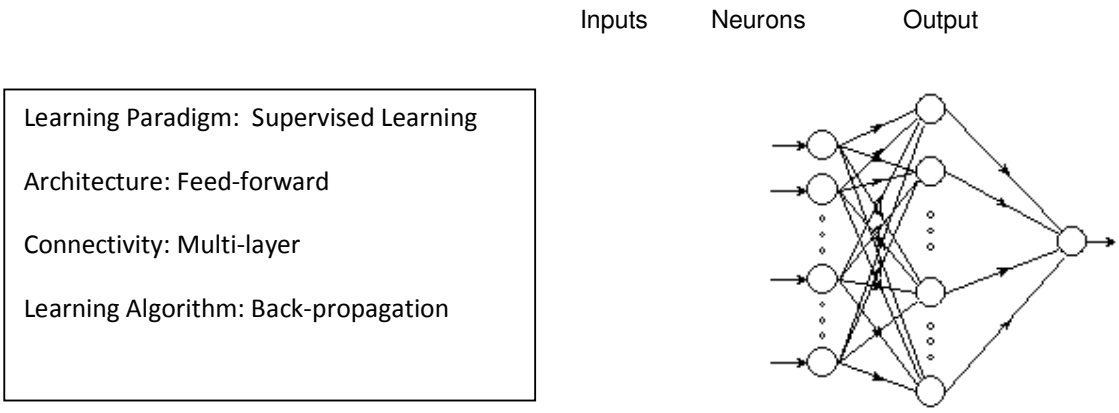


Figure 1: Architecture of a MLP with one layer.

MLPs and many other neural networks learn using an algorithm called backpropagation. With backpropagation, the input data are repeatedly presented to the neural network. With each presentation, the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (backpropagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as “training”.

After selected the training data set, the following sequence of steps are performed:

1) Present the network with an input-target (output) pair.
2) Compute the predictions of the network for the targets.
3) Use the error function to calculate the difference between the predictions (output) of the network and the target values.
4) Continue with steps 1 and 2 until all input-target pairs have been presented to the network.
5) Use the training algorithm to adjust the weights of the networks so that it gives better predictions for each input-target pair.

   The above steps correspond to one training *epoch*. The number of epochs needed to train a neural network model is not known, but can be obtained as part of the training process.

6) Repeat steps 1 to 5 again for a number of training epochs until the network starts producing sufficiently accurate outputs (according a specified tolerance).

Steps 1 through 5 constitute one training epoch. A typical neural network training process needs hundreds of epochs.

Neural networks are highly nonlinear tools that are usually trained using iterative techniques. The most commonly recommended techniques for training neural networks are the *BFGS* and *Scaled Conjugate*

Gradient algorithms[6]. These methods perform significantly better than the more traditional algorithms such as Gradient Descent, but they are, generally speaking, more memory-intensive and computationally demanding. Nonetheless, these techniques may require a smaller number of iterations to train a neural network given their fast convergence rates and more intelligent search criteria.

The problem of lengthy training time can be overcome either by devising faster learning algorithms or by implementing the existing algorithms on parallel computing architectures. The improvement of learning algorithms is an active area of research, and an improved algorithm can be further sped up by parallel implementation.

The performance of neural networks is greatly affected by their design, yet the question of finding optimal designs remains open and inspires a considerable amount of research. Most recent studies have focused on developing automatic algorithms for neural network configurations.

The choice of which network architecture to employ is an important decision that can dramatically alter the results of the analysis. A variety of strategies are used to select the network architecture. The features of network architecture that are most commonly optimized are the number of hidden layers and the number of nodes in the hidden layer. Many of these approaches use a prediction error fitness measure, such that they select an architecture based on its generalization to new observations, while others use the classification error or training error. The starting point is a very small network, and several parameters are optimized to obtain an appropriate architecture for each data set; these include the number of hidden layers, the number of nodes in the hidden layers, and the fraction of the previous change in a weight that is added to the next change (called the learning momentum). This trial-and-error approach, using some empirical rules obtained from experience, is the most commonly used approach for architecture optimization. The genetic algorithm theory can also be used to optimize neural networks automatically, but may result in a computationally expensive solution.

Multilayer perceptrons adjust their internal parameters by performing vector mappings from the input to the output space. Although they may achieve high accuracy, the knowledge acquired by such neural networks is usually incomprehensible to humans. This is a major obstacle in data mining applications, in which ultimately understandable patterns (e.g., classification rules) are important. Therefore, many algorithms have been developed for extracting rules from neural networks.

Neural networks (NNs) learn by adjusting their connection weights, which somehow reflect the statistical properties of the data. Thus, the knowledge acquired by an NN is codified in its connection weights, which in turn are associated to both its architecture and activation functions[7]. In this context, the process of decoding knowledge from NNs usually implies the use of algorithms based on the values of either the connection weights or the hidden unit activations. The algorithms designed to perform such tasks are generally called algorithms for rule extraction from neural networks. The task of rule extraction from NNs is a computationally difficult problem[8], and heuristics have been developed to overcome its combinatorial complexity[9]. A clustering algorithm is employed for rule extraction from MLPs. The method is based on the hidden unit activation values and consists of two main steps.

First, clustering is employed to find clusters of hidden unit activation values. Then, these clusters are translated into logical rules. Andrews et al.[7] suggested a classification scheme for rule extraction algorithms. The proposed scheme is based on four aspects: (i) the form and quality of the extracted rules; (ii) the necessity of specific neural network training algorithms; (iii) the complexity of the rule extraction algorithm; and (iv) the translucency of the neural network. According to this scheme, the method provides propositional rules of the form “If y Then x,” and does not require any specific MLP training algorithms.

This method can be additionally applied, for example, to classification problems involving discrete and continuous attributes. The complexity of the rule extraction algorithm is based on the clustering strategy employed. As far as the translucency of the NN is concerned, there are three approaches: decompositional, pedagogical and eclectic. Decompositional approaches involve rule extraction at the level of hidden and output units, which are mapped in a binary form. Pedagogical approaches try to map inputs directly to outputs using machine-learning techniques. When hidden unit activation expressions are employed to get classification rules by means of a clustering algorithm the approach can be classified as eclectic because it is based on both decompositional and pedagogical approaches.

Radial Basis Function Network

Radial basis function (RBF) networks emerged as a variant of artificial neural networks in the late 1980's, and they are effective computational tools that have attracted great interest in the areas of systems modeling and data mining. The main advantages of RBF networks are the simplicity of their structure and the speed of their learning algorithms.

RBF networks are commonly trained following a hybrid procedure that operates in two stages, where the center locations (hidden nodes) are computed first and the connection weights are calculated in the second stage.

Radial functions are simply a class of functions. In principle, they could be employed in any sort of linear or nonlinear models, and any sort of single-layer or multilayer networks. However RBF networks have traditionally been associated with radial functions in a single-layer network such as that shown in Figure 2.

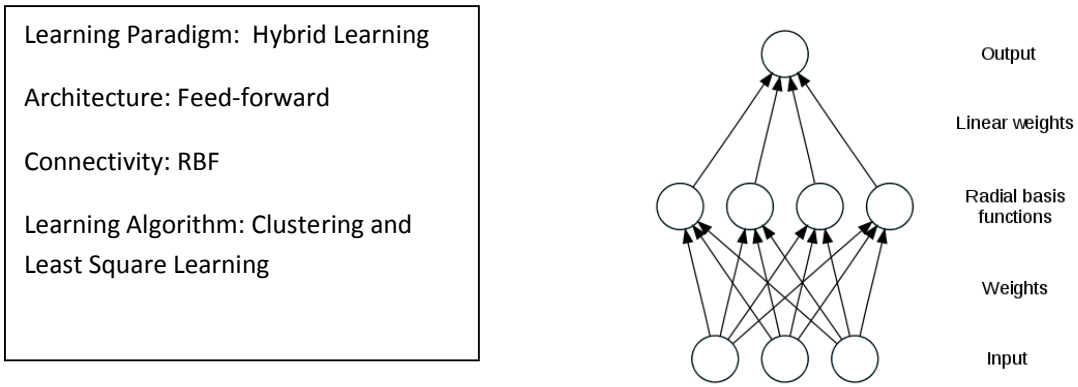


Figure 2: Architecture of a radial basis function network.

The methods used to train radial basis function networks are fundamentally different from those employed for MLPs. This mainly is due to the nature of the RBF networks, with their hidden neurons (basis functions) forming a Gaussian mixture model that estimates the probability density of the input data[6]. For RBF networks with linear activation functions, the training process involves two stages. In the first part, we fix the location and radial spread of the basis functions using the input data (no targets are considered at this stage). In the second stage, we fix the weights connecting the radial functions to the output neurons. When the output activation function is the identity, this second stage of training involves a simple matrix inversion. Thus, it is exact and does not require an iterative process. In summary, the most popular training technique is based on a two-stage approach, using k-means clustering to determine the center locations, while the weights of the output layer are trained by a single-shot process using pseudo-inverse matrices.

The linear training, however, is only valid when the error function is sum-of-squares and the output activation functions are the identity. In the case of cross-entropy error functions and output activation functions other then the identity, an iterative algorithm such as BFGS is necessary to fix the hidden-output layer weights in order to complete the training of the RBF neural network.

Besides the k-means-based approach, a variety of algorithms have been suggested to determine the configuration of RBF networks automatically, including the orthogonal least squares algorithm[10], constructive methods[11] (in which the network is incrementally built), pruning methods[12] (in which a large number of centers is reduced as the algorithm proceeds) and optimization methods based on genetic algorithms[13].

The users are almost entirely excluded from the RBF network configuration process when applying these automatic algorithms, which reduce the user's workload but also restrict process manipulation. Usually, the user sets the parameters before algorithm execution and repeats the entire process if the results are inadequate.

Nonlinear system identification via artificial neural networks consists of adjusting the network in such a way that it approximately describes the input-output mapping of the system. In its complete form, network

induction involves both *parametric* and *structural* learning[1]. The two tasks can be seen as optimization problems, and as such, they may present problems with multiple local minima, non-differentiability and discontinuities.

As a result, the idea of applying *genetic algorithms* (GA) to the problem of complete network acquisition comes naturally. In summary, an RBF network involves the following adjustable parameters: (1) the number m of RBFs units in the hidden layer; (2) the positions of the RBF centers; (3) the widths of the RBFs; (4) the output weights.

There are several possible ways to use a GA to configure a RBF network. A straightforward approach is to fix a topology and use the GA as an optimization tool to compute all the free parameters. This has been done for time-series forecasting[14]. In Mak[15], the number of hidden neurons was also fixed, and the GA optimized only the locations of the centers. The widths of the centers and output weights were computed by the k-nearest neighbor heuristic and the singular value decomposition, respectively. Whitehead and Choate[16] also fixed the number of centers and evolved their locations and widths, but using a completely different — and interesting— approach: instead of encoding a network in each individual, they combined the entire set of chromosomes to constitute a RBF network, each one being a center.

The most common and most promising approach is to use a GA to set the network's topology and centers' locations and widths, while using an algebraic or gradient-descent method to compute the weights in the output layer are computed by This approach was adopted by several authors. Naturally, there are many differences between them: for example, in Whitehead and Choate[17], an indirect representation is used; the locations of the centers are governed by space-filling curves whose parameters evolve genetically. Another example is the work of Maillard and Gueriot[18], in which the basis functions are not only restricted to Gaussians but also subjected to evolution.

It is possible to interpret the hidden layer configuration of a RBF network as a subset selection problem. Given a fixed set of candidate RBFs, one can compare different network configurations, each with a subset of the original population as its hidden units. For real-world applications, finding the best possible subset is often an intractable task. Therefore, it is necessary to consider heuristic procedures to explore this search space rationally. One idea is to start with an empty hidden layer, to which new RBF networks are successively added. The choice of which function to select can be made based on a simple criterion: choose the one that reduces some error measure the most, e.g., the sum of square errors (SSE).

Among the approaches for dealing with the latter problem, two algorithms stand out for different reasons. The orthogonal least squares algorithm (OLS)[19] is extremely computationally efficient, but tends to generate suboptimal solutions. On the other hand, genetic algorithms (GAs) usually produce excellent solutions for the topological definition of RBF networks, but the associated computational cost is excessive. It is better to adopt a hybrid algorithm that assimilates the qualities of both of these approaches. The genetic orthogonal least squares algorithm (GOLS)[19] is capable of generating solutions that are substantially better than those produced by the OLS method, without incurring in the computational cost of the standard GA.

It is also possible to implement an approach, based on a visual technique called Star Coordinates[20], that allows the user to be involved in the RBF network configuration process via interactive visualization. The Star Coordinates technique renders a visual RBF network design from the algorithmic results and allows the user to participate interactively in the design evaluation and refining. The proposed technique may be useful for academic and business areas, although it is expected to inspire greater interest among students, educators and researchers once they have the appropriate knowledge to enable wider and more efficient utilization of the available resources.

Jang and Sun[21] have shown that radial basis function RBF networks and a simplified class of fuzzy systems are functionally equivalent under some mild conditions. This functional equivalence has made it possible to combine the features of these two systems, which have been developed into powerful "neurofuzzy" systems[22]. However, a fuzzy system that has been trained using learning algorithms may lose its interpretability or transparency, which is one of the most important features of fuzzy systems. The extraction of interpretable fuzzy rules from trained RBF networks may be possible by the application of regularization techniques.

Self-Organizing Map

The Kohonen Self-Organizing Map (SOM) network[23] is probably the best-known of the unsupervised neural network methods, and it has been used in a wide variety of applications. The main function of SOM

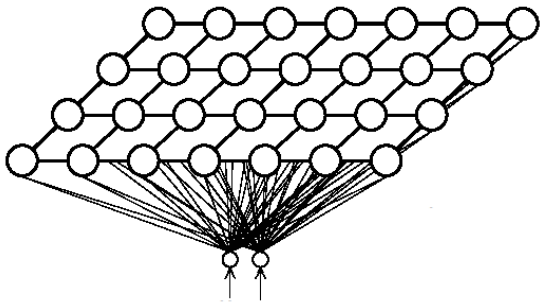
networks is to map the input data from an n-dimensional space to a lower dimensional plot while maintaining the original topological relations (Figure 3).

Learning Paradigm: Unsupervised Learning

Architecture: Recurrent

Connectivity: Self Organized Maps

Learning Algorithm: Kohonen



Inputs

Figure 3: Architecture of a SOM network.

SOM networks can be used for clustering, classification, visualization and modeling. In particular, their visualization capabilities are useful for providing informative pictures of the data space and exploring data vectors or whole data sets. The versatile properties of SOM networks make them valuable tools in data mining and knowledge discovery, and they have been successfully applied in a variety of areas.

A basic SOM network is composed of an input layer and a Kohonen layer. The input layer contains neurons for each element in the input vector. The Kohonen layer is formed by neurons that are located on a regular, usually two-dimensional grid and are fully connected to those at the input layer. The neurons in the map are connected to adjacent ones by a neighborhood relationship dictating the topological structure of the neurons. The whole process is driven by repeated representations of the input vectors and the applied learning rule. The process begins with all network weights initialized to a small random value. Training proceeds by repeated exposure of the network to the entire set of input vectors. The Kohonen layer computes the distances between the weight vector for each of the neurons and the input pattern. The neuron that is closest (minimum distance) to the input vector in terms of some measure is the ‘‘winner’’. When an input vector is *presented* to the network, the neurons in the map compete with each other to be the winner, which is the closest to the input vector in terms of some kind of dissimilarity measure. Thereafter, similar input vectors are grouped into a single neuron or neighboring ones in the map when learning is accomplished.

Despite the large number of research reports on the use of self-organizing maps, some significant difficulties remain. The determination of a suitable number of neurons requires some insight into the structure of the data. This cannot be assumed in all cases, and it might be helpful if the neural network could determine this number itself during its learning process.

SOMs have great powers of visualization, but they cannot provide a full explanation of their structure and composition without further detailed analysis. One step towards filling this gap is provided by the unified distance matrix or U-matrix technique of Ultsch[24]. The U-matrix technique calculates the weighted sum of all Euclidean distances between the weight vectors for all output neurons. The resulting values can be used to interpret the clusters created by the SOM.

In terms of the interpretability of a trained SOM, it is not easy to read and interpret the structure and the characteristics learned during the training process from the map display without extensive manual interaction. While the map is able to learn the distinctions between various clusters, the exact extent of the clusters as well as their characteristics cannot be identified from the standard SOM representation. This problem has led to the development of a number of enhanced visualization techniques supporting the interpretation of the self-organizing map. However, while these enhanced visualization techniques may provide assistance in identifying the cluster structure and cluster boundaries on manual inspection, they still do not provide any information about the characteristics of the clusters. Thus, it still remains a tedious task to label the map manually, or to determine the features that are characteristic of a particular cluster. Given the mapping of an unknown data set onto a self-organizing map, even with the visualization of clear cluster boundaries it remains a nontrivial task to elicit the features that are the most relevant and determining for a

group of input data to form a cluster of its own. It is also difficult to determine which features the clusters share and which features distinguish them from other clusters.

It is necessary to develop a method that automatically labels a self-organizing map based on the features learned during the training process. To achieve this, every unit of the map is labeled with the features that best characterize all data points mapped onto that particular unit. This is accomplished by using a combination of the quantization error of every feature and the relative importance of each feature in the weight vector of the unit.

Data Mining Applications

Data mining consists of the algorithms employed for extracting patterns from data. In general, data mining tasks can be classified into two categories: descriptive and predictive. The descriptive techniques provide a summary of the data and characterize its general properties. The predictive techniques learn from the current data to make predictions about the behavior of new data sets. The most common tasks in data mining are association rule mining, classification, regression and cluster analysis. The last 3 tasks are in many cases performed with very high performance using neural network models, such that NNs have become an obligatory approach in the most commonly used data mining systems and database management systems today. The focus of this survey was on MLP, RBF and SOM networks. Many other types of neural networks could be also described. Bayesian networks are at least worth mentioning.

A Bayesian network (BN) is a directed acyclic graph in which the nodes represent the variables and the arcs represent causal relationships among the variables they connect. The conditional probability table gives the strengths of such relationships. The variables that are not connected by an arc can be considered as having no direct causal influence. Bayesian networks and Bayesian classifiers are usually employed in data mining tasks, mainly because they may deal with incomplete data sets straightforwardly, can learn causal relationships, may combine prior knowledge with patterns learnt from data and can help to avoid overfitting.

The learning of a Bayesian network from data has become a highly active research topic in the last decade[25], and there are two main classes of methods to perform this task: methods based on heuristic searching[26] and methods based on the definition of conditional independence (CI)[27] to generate the network structure. There are also methods that combine these two strategies[28]. In a process of BN learning from data, the variables of the BN represent the dataset attributes. When using algorithms based on a heuristic search, the initial order of the dataset attributes may be considered an important issue. Some of these algorithms depend on this ordering to determine the direction of the arcs such that only an earlier attribute (in an ordered list) is a possible parent of a later one. On the other hand, conditional independence methods try to find the directions of the arcs without the need to order the attributes. However, even the CI methods can be improved when the ordering is known. Today, it is also recognized that the relationships extracted from complex network theory generate valuable knowledge to add to our understanding of interdisciplinary activities.

There is a wide variety of commercial tools available as complete systems, along with countless others currently under development. These tools range in scope from small-scale systems developed for particular problem domains to general-purpose systems that can be used for almost any data mining problem. There is great variance in terms of cost, platform independence, requirements and visualization facilities.

Nowadays, data mining tools are fully developed, very powerful and ready to be used. Data Mining finds applications in many areas. These areas can be grouped according to their objectives as follows:

- *Data Mining for Competitive Intelligence*: New Product Ideas, Retail Marketing and Sales Pattern, Competitive Decisions, Future Trends and Competitive Opportunities, etc.
- *Data Mining on Finance*: Consumer Credit Policy, Portfolio Management, Bankruptcy Prediction, Foreign Exchange Forecasting, Derivatives Pricing, Risk Management, Price Prediction, Forecasting Macroeconomic Data, Time Series Modeling, etc.
- *Scientific Models from Data*: Applications of Data Mining in Science, Engineering, Medicine, Global Climate Change Modeling, Bioinformatics, Ecological Modeling, etc.
- *Data Mining on Government*: Detection and Prevention of Fraud and Money Laundering, Criminal Patterns, Health Care Transactions, Counterterrorism, etc.

Many reasons can be listed for the wide range kind of applications encountered in data mining practice, including the following:

1. High Accuracy: Neural networks are able to approximate complex non-linear mappings.

2. Noise Tolerance: Neural networks are very flexible with respect to incomplete, missing and noisy data.

3. Independence from prior assumptions: neural networks do not make a priori assumptions about the distribution of the data, or the form of interactions between variables.

4. Ease of maintenance and exporting: Neural networks can be updated with fresh data, making them useful for dynamic environments.

5. Neural networks can be implemented in parallel hardware.

Although neural networks do seem to be able to solve many problems, it is important to pay attention to their limitations and problem characteristics. Overconfidence in neural networks can result in costly mistakes.

The evolution of data mining algorithms is closely connected to the development of other particular areas. Machine learning is traditionally a key component of artificial intelligence, and it provides innovative ideas for learning and adaptation utilized in neural network models. Soft computing and hybrid algorithms combine neural, fuzzy and evolutionary approaches to computational intelligence; recent developments include artificial immune systems and quantum computing. Hybrid methods significantly contribute to the robustness of intelligent technologies and have many applications. Intelligent signal processing is a major field where neural networks have clearly demonstrated their advantages over traditional statistical techniques. Neural networks have been successful in solving difficult signal processing and pattern recognition problems.

Nowadays a wide variety of data mining platforms are available. Some are very specialized, some can be used for development purposes, and several are complete data mining suites. Data mining software suites are complete tools or environments that support the entire data mining process, including data preparation, classification, clustering, and visualization. In general, the user looks for the tool suite that is easy to use and provides the necessary data with acceptable accuracy and is able to perform all the common tasks in a data mining project. Low costs and database integration are also desirable (Microsoft SQL Server, Oracle Data Mining). In Figure 4, some data mining suites are classified according to the MLP, RBF and SOM neural network availability. The preference for those models is quite evident.

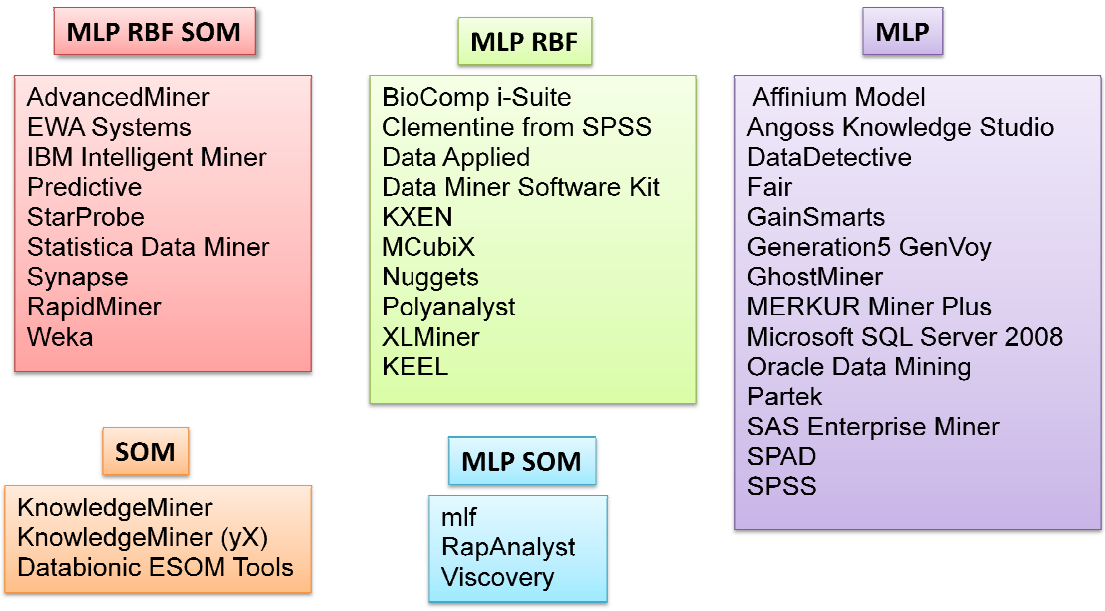


Figure 4: Data Mining Suites[29]

Finally, it must be stressed that 70-90 percent of the time required to perform a data mining project is spent in data preparation for modeling, for the following reasons:

- Data in commercial databases are collected from transactional systems to serve querying and reporting purposes, not analytical purposes.
- Data in commercial databases are rather dirty; i.e., databases often contain inappropriate data, improperly input data, etc.
- Most data mining algorithms require clean and complete data records as input. Missing data in some fields, outliers, etc. can invalidate the method.

Neural networks are powerful tools for improving data preparation tasks. A good data mining suite will provide tools for performing all of these operations. Some data mining suites are better than others in this aspect, which may be crucial for the success of a project.

Conclusion

It is impossible to comprehensively cover all the developments in the neural network field and all activities leading to real world applications of data mining[39,.43]. MLP, RBF and SOM networks are wide-range tools that represent a significant advance in data modeling. They have efficient learning and knowledge-generating algorithms that can be optimized and easily exported as functions to represent complex phenomena.

It is recognized that the success of data mining lies in its predictive analytics, and neural networks are one of the most important approaches to this problem. This trend began in the business intelligence world but has now invaded all scientific activities and become completely pervasive. Because most information held within business and science today is in unstructured format, the integration of NN data mining and text mining complete this dissemination.

References

1 Haykin S., Neural Networks and Learning Machines, Pearson Education, Inc.,New Jersey, 2009.

2 Krogh A.and Vedelsby J., Neural network ensembles, cross-validation and active learning, in Tesauro G, Touretzky D and Lean T, eds., *Advances in Neural Information Processing Systems*, MIT Press, 1995, 7:231-238.

3 Breiman L., Bagging predictors, Machine Learning, 1996, 24(2):123-140.

4 Freund Y. and Schapire R., Experiments with a new boosting algorithm, In Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy,1996, 148-156.

5 Hruschka E.R. ; Ebecken, N.F.F., Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach, Neurocomputing, Amsterdam, ELSEVIER, 2006,70:384-397.

6 Bishop C.M., Neural Networks for Pattern Recognition, Oxford University Press,1995.

7 Andrews R, Diederich and, Tickle A.B., A. Survey, Critique of techniques for extracting rules from trained artificial neural networks, Knowledge Based Systems. 1995, 8(6): 373–389.

8 Golea M, On the complexity of rule-extraction from neural networks and network-querying, in: Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, University of Sussex, Brighton, UK, 1996, 51–59.

9 Vahed A, Omlin CW, Rule extraction from recurrent neural networks using a symbolic machine learning algorithm, in: Proceedings of the Sixth International Conference on Neural Information Processing, Perth, Australia, 1999.

10 Chen S, Billings SA, Cowan CFN, Grant PW. Practical identification of NARMAX models using radial basis functions. Int J Control 1990,;52:1327–50.

11 Fritzke B. Fast learning with incremental RBF networks. Neural Process Lett 1994;1:2–5.

12 Musavi MT, Ahmed W, Chan KH, Farms KB, Hummels DM. On the training of radial basis function classifiers. Neural Netw 1992;5:595–603.

13 Barreto A, Barbosa H, Ebecken N. Growing compact RBF networks using a genetic algorithm. In: Proceedings of the VII Brazilian symposium on neural networks, 2002.

14 Sheta A.F. and Jong K.D., Time-series forecasting using GA-tuned radial basis functions, *Information Sciences*, Special issue, 2001.

15 Mak M.W. and Cho K.W., Genetic evolution of radial basis function centers for pattern classification. In *Proc. Of The 1998 IEEE International Joint Conference on Neural Networks*, 1998,1:669–673.

16 Whitehead B.A. and Choate T.D., Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Trans. on Neural Networks*, 1995, 7(4):869–880.

17 Whitehead B.A. and Choate T.D., Evolving space-filling curves to distribute radial basis functions over an input space. *IEEE Trans. on Neural Networks*, 1993, 5(1):15–23.
18 Maillard E.P. and Gueriot D., RBF neural network, basis functions and genetic algorithm. In *Int. Conf. on Neural Networks*, IEEE,1997, 4: 2187–2192.

19 Barreto A.M.S., Barbosa H.J.C, Ebecken N.F.F., GOLS - Genetic orthogonal least squares algorithm for training RBF networks. Neurocomputing (Amsterdam), Elsevier, 2006, 69:2041-2064.

20 Otto M.F. Ebecken, N.F.F., Visual RBF network design based on Star Coordinates. Advances in Engineering Software, 2009, 913-919.

21 Jang J.S. R. and Sun C.T., Functional equivalence between radial basis functions and fuzzy inference systems, IEEE Trans. on Neural Networks, 1993,4:156–158.

22 Jang, J.S.R and Sun C.T., Neuro-fuzzy modeling and control, Proceedings of the IEEE, 1995, 83:378–405.

23 Kohonen T., Self-Organizing Maps, Springer-Verlag Berlin Heidelberg Germany 1995.
24 Ultsch A., Mantyk R., Halmans G., Connectionist knowledge acquisition tool: CONKAT. In: Hand J (ed) Artificial intelligence frontiers in statistics: AI and statistics III. Chapman and Hall, 1993, 256–263.

25 Gelman A., Carlin J.B., Stern H.S. and Rubin D.B., Bayesian Data Analysis, Chapman and Hall, London 1995.

26 Acid S. and Campos L.M., Searching for Bayesian Network structures in the space of restricted acyclic partially directed graphs, *Journal of Artificial Intelligence Research*, 2003, 18:445–490.

27 Campos M., Independency relationships and learning algorithms for singly connected networks, *Journal of Experimental and Theoretical Artificial Intelligence*, 1998, 10:511–549.

28 Acid S., Campos L.M., Benedict: an algorithm for learning probabilistic belief networks, in: Proceedings of the IPMU-96 Conference, 1996, 979–974.

29 http://www.kdnuggets.com (accessed November 30 2010)

30 Singh Y.and Chauhan A.S.,Neural Networks in Data Mining, http://www.jatit.org/volumes/research-papers/Vol5No1/1Vol5No6.pdf

31 Krieger C., Neural Networks in Data Mining,1-23,
http://www.cs.uml.edu/~ckrieger/user/Neural_Networks.pdf

32 Vesely A., Neural Networks in data mining,427-431,
http://wps.prenhall.com/wps/media/objects/4242/4344809/turban_online_ch06.pdf

33 Craven M.W.and Shavlik J.W., Using neural networks for data mining, Future Generation Computer Systems 13,1997 21:1-229.

34 How Neural Networks are used in Data Mining,
http://members.cox.net/john.stanton/Neural_Networks.html

35 Nirkhi S., Potential use of Artificial Neural Network in Data Mining**,** The 2nd International Conference on Computer and Automation Engineering (ICCAE), 2010 , Singapore , IEEE digital Library.

36 Myllymäki P., Advantages of Bayesian Networks in Data Miningand Knowledge Discovery, http://www.bayesit.com/docs/advantages.html

37 Laxman S. and Sastry P.S., A survey of temporal data mining, *Sadhana*, Part 2, April 2006, 31:173–198, http://www.ias.ac.in/sadhana/Pdf2006Apr/173.pdf

38 Pedrycz W., Handbook of data mining and knowledge discovery, Oxford University Press, Inc. New York, NY, USA 2002.

39 Kargupta H., Han J., Yu P.S., Motwani R. and Kumar V. (Editor), Next Generation of Data Mining (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series) 2009.

40 Howlett R.J., Jain L.C.(Editor), Radial Basis Function Networks, Recent Developments in Theory and Applications (Studies in Fuzziness and Soft Computing) , Physica-Verlag HD; December, 2010.

41 David V.K. and Rajasekaran S.,Pattern Recognition Using Neural and Functional Networks Studies in Computational Intelligence,Springer; 2011.

42 Kwon S.J.(Editor), Artificial Neural Networks, Nova Science Pub Inc.,2011.

43 Du, K.L., Clustering: A neural network approach, Neural Networks, 2010, 23:89-107.