PRACTICAL ON ARTIFICIAL NEURAL NETWORKS

Amrender Kumar
Indian Agricultural Statistics Research Institute
Library Avenue, New Delhi – 110012
akjha@iasri.res.in

Neural networks, more accurately called Artificial Neural Networks (ANNs), are computational models that consist of a number of simple processing units that communicate by sending signals to one another over a large number of weighted connections. They were originally developed from the inspiration of human brains. In human brains, a biological neuron collects signals from other neurons through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. Like human brains, neural networks also consist of processing units (artificial neurons) and connections (weights) between them. The processing units transport incoming information on their outgoing connections to other units. The "electrical" information is simulated with specific values stored in those weights that make these networks have the capacity to learn, memorize, and create relationships amongst data. A very important feature of these networks is their adaptive nature where "learning by example" replaces "programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. These networks are “neural” in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. ANNs have powerful pattern classification and pattern recognition capabilities through learning and generalize from experience. ANNs are non-linear data driven self adaptive approach as opposed to the traditional model based methods. They are powerful tools for modelling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. These techniques are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology, physics, biology and agriculture. There are many different types of neural networks. Some of the most traditional applications include classification, noise reduction and prediction.

Various steps for designing a neural networks models

Step 1: Variable selection

The input variables important for modeling/ forecasting variable(s) under study are selected by suitable variable selection procedures.

Step 2: Data Preprocessing

Data preprocessing refers to analyzing and transforming the input and output variables to minimize noise, highlight important relationships, detecting trends and flatten the distribution of the variables to assist the neural network in learning the relevant patterns. This also can be achieved by normalization. The choice of the normalisation methods usually depends on the composition of the input vector. The following formulae are frequently used:

Linear transformation to [0,1]: $\frac{x_0 - x_{\min}}{x_{\max} - x_{\min}}$

statistical normalization: $\frac{x_0 - \bar{x}}{s}$

simple normalization: $\frac{x_0}{x_{\max}}$

Step 3: Partitions of data sets *viz.* training, testing and validation sets

The data set is divided into three distinct sets called training, testing and validation sets. The training set is the largest set and is used by neural network to learn patterns present in the data. The testing set is used to evaluate the generalization ability of a supposedly trained network. A final check on the performance of the trained network is made using validation set.

Step 4: Neural networks paradigms

Neural network architecture defines its structure including number of hidden layers, number of hidden nodes, activation function and number of output nodes etc.

(a) Number of hidden layers: The hidden layer(s) provide the network with its ability to generalize. In theory, a neural network with one hidden layer with a sufficient number of hidden neurons is capable of approximating any continuous function. In practice, neural network with one and occasionally two hidden layers are widely used and have to perform very well.

(b) Number of hidden nodes: There is no magic formula for selecting the optimum number of hidden neurons. However, some thumb rules are available for calculating number of hidden neurons. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters (1993). For a three layer network with n input and m output neurons, the hidden layer would have *sqrt(n*m)* neurons.

(c) Number of output nodes: Neural networks with multiple outputs, especially if these outputs are widely spaced, will produce inferior results as compared to a network with a single output.

(d) Activation function: Activation functions are mathematical formulae that determine the output of a processing node. Most units in neural network transform their net inputs by using a scalar-to-scalar function called an *activation function*, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called 'squashing functions'. Appropriate differentiable function will be used as activation function. Some of the most commonly used activation functions are :

(a) The sigmoid (logistic) function

$$f(x) = (1 + \exp(-x))^{-1}$$

(b) The hyperbolic tangent (tanh) function

$$f(x) = (\exp(x) - \exp(-x))/(\exp(x) + \exp(-x))$$

(c) The sine or cosine function

$$f(x) = \sin(x) \quad or \quad f(x) = \cos(x)$$

Activation functions for the hidden units are needed to introduce non-linearity into the networks. The reason is that a composition of linear functions is again a linear function. However, it is the non-linearity (i.e. the capability to represent nonlinear functions) that makes multilayer networks so powerful. Almost any nonlinear function does the job, although for back-propagation learning it must be differentiable and it helps if the function is bounded. Therefore, the sigmoid functions are the most common choices. There are some heuristic rules for selection of the activation function. For example, Klimasauskas (1991) suggests logistic activation functions for classification problems which involve learning about average behaviour, and to use the hyperbolic tangent functions if the problem involves learning about deviations from the average such as the forecasting problem.

Step 5: Evaluation criteria

The most common error function minimized in neural networks is the sum of squared errors. Other error functions offered by different software include least absolute deviations, least fourth powers, asymmetric least squares and percentage differences.

Step 6: Neural networks training

Training a neural network to learn patterns in data involves iteratively presenting it with examples. The objective of training is to find the weights between the neurons that determine the global minimum of the error function. Multilayer feed forward neural network or multi layer perceptron (MLP), is very popular and is used more than other neural network type for a wide variety of tasks. Multilayer feed forward neural network learned by back propagation algorithm is based on supervised procedure, i.e., the network constructs a model based on examples of data with known output.

Step 7: Implementation

The developed neural networks can be implemented for the unseen part of data or future data for prediction as well as classification.

The neural networks models can be developed using Statistical Neural Networks Version 6.1 available at Indian Agricultural Statistics Research Institute, New Delhi. The details about this software are given below:

About STATISTICA Neural Networks (SNN)

STATISTICA Neural Networks (SNN) is a comprehensive, state-of-the-art, powerful, and extremely fast neural network data analysis package, featuring:

- Integrated pre- and post-processing, including data selection, nominal-value encoding, scaling, normalization, and missing value substitution, with interpretation for classification, regression, and time series problems.
- Exceptional ease of use coupled with unsurpassed analytic power; for example, a unique wizard-style Intelligent Problem Solver can guide you step by step through the procedure of creating a variety of different networks and choosing the network with the best

performance (a task that would otherwise require a lengthy "trial and error" process and a solid background in the underlying theory).

- Powerful exploratory and analytic techniques, including Input Feature Selection algorithms (choosing the right input variables in exploratory data analysis, which is a typical application of neural networks, is often a time-consuming process; *STATISTICA Neural Networks* can also do this for you).
- State-of-the-art, highly optimized training algorithms (including Conjugate Gradient Descent and Levenberg-Marquardt); full control over all aspects that influence the network performance such as activation and error functions, or network complexity.
- Support for combinations of networks and network architectures of practically unlimited sizes organized in Network Sets; selective training of network segments; merging, saving of network sets in separate files.
- Comprehensive graphical and statistical feedback that facilitates interactive exploratory analyses.
- Integration with the *STATISTICA* system, including direct transfer of data and graphs to *STATISTICA* for further analysis and customization of results (*STATISTICA Neural Networks* can also be used as a stand-alone package).
- API (Application Programming Interface) support for embedded solutions using Visual Basic, Delphi, C, C++ and other languages.

STATISTICA Neural Networks has the functionality to help you through the critical design stages, including not only state-of-the-art Neural Network Architectures and Training Algorithms, but also innovative new approaches to Input Parameter Selection and Network Design. Moreover, software developers and those users who experiment with customized applications will appreciate the fact that once your prototyping experiments are completed using *SNN*'s simple, intuitive interface, the *STATISTICA Neural Networks* API allows you to embed neural technology into your own specialized software simply and at low cost.

Input Data

STATISTICA Neural Networks stores its data in the *STATISTICA* data format; thus, your data can easily be transferred from *STATISTICA* or converted to the *STATISTICA* data format using any of the wide variety of *STATISTICA* data import facilities. Alternatively, you can create a data set in *STATISTICA Neural Networks* by entering the data into the editor, pasting them from the Clipboard, or importing from ASCII (text) files (both TAB-delimited and comma-delimited formats are supported). *SNN* automatically identifies nominal valued variables and missing values during Import. Once data are inside *STATISTICA Neural Networks*, they can easily be edited in *STATISTICA Neural Networks*' data set Editor, using a familiar spreadsheet-like interface; editing operations can include the labeling of cases and variables, adding and removing cases and/or variables, identification of input and output variables, or division of cases into Training, Verification, and Test Sets. You can also temporarily ignore selected cases and/or variables.

Input Selection and Dimensionality Reduction

Once you have a data set prepared, you will need to decide which variables to use in your neural network. Larger numbers of input variables require larger neural networks, with consequent increases in storage and training time requirements and the need for greater numbers of training cases. Lack of data and correlations between variables make the selection of important input variables, and the compression of information into smaller numbers of variables, issues of critical importance in many neural applications.

Input feature selection algorithms.
STATISTICA Neural Networks includes backward and forward stepwise selection algorithms. In addition, the Neuro-Genetic Input Selection algorithm uniquely combines the technologies of Genetic Algorithms and PNN/GRNNs (PNN stands for Probabilistic Neural Networks, and GRNN stands for Generalized Regression Neural Network) to automatically search for optimal combinations of input variables, even where there are correlations and nonlinear interdependencies. The near-instantaneous training time of PNN/GRNN not only allows the Neuro-Genetic Input Selection algorithm to operate, it also allows you, in conjunction with the *SNN* data set Editor's simple variable suppression facilities, to conduct your own input sensitivity experiments on a realistic time scale. *STATISTICA Neural Networks* also includes built-in Principal Components Analysis (PCA and Autoassociative networks for "nonlinear PCA") to extract smaller numbers of dimensions from the raw inputs. Note that a wide variety of statistical tools for data reduction are available in *STATISTICA*.

Data Scaling and Nominal Value Preparation
Data must be specially prepared for input into a network, and also it is important that the network output can be interpreted correctly. *STATISTICA Neural Networks* includes automatic data scaling (including Minimax and Mean/SD scaling) for both inputs and outputs; there is also automatic recoding of Nominal valued variables (e.g., Sex={Male,Female}), including one-of-N encoding. *STATISTICA Neural Networks* also has facilities to handle missing data. Normalization functions such as Unit Sum, Winner-takes-all, and Unit Vector are also supported. There are special data preparation and interpretation facilities for use with Time Series. A large number of relevant tools are also included in *STATISTICA*. For classification problems, you can set confidence limits, which *SNN* uses to assign cases to classes. In combination with *STATISTICA Neural Networks*' specialized Softmax activation function and cross-entropy error functions, this supports a principled, probabilistic approach to classification.

Selecting a Neural Network Model
The range of neural network models and the number of parameters that must be decided upon (including network size, and training algorithm control parameters) can seem bewildering (the wizard-style *Intelligent Problem Solver* is available to guide you through the selection process). *STATISTICA Neural Networks* supports the most important classes of neural networks for real world problem solving, including:
Multilayer Perceptrons (Feedforward Networks)
Radial Basis Function Networks
Kohonen Self-Organizing Feature Maps
Probabilistic (Bayesian) Neural Networks
Generalized Regression Neural Networks
Linear Modeling.

STATISTICA Neural Networks has numerous facilities to aid in selecting an appropriate network architecture. *STATISTICA Neural Networks* statistical and graphical feedback includes Bar Charts, Matrices and Graphs of individual and overall case errors, summaries of classification/misclassification performance, and vital statistics such as Regression Error Ratios - all automatically calculated.

STATISTICA Neural Networks automatically retains a copy of the best network found as you

experiment on a problem, which can be retrieved at any time. The usefulness and predictive validity of the network can automatically be assessed by including verification cases, and by evaluating the size and efficiency of the network as well as the cost of misclassification. *SNN*'s automatic Cross Verification and Weigend Weight Regularization procedures also allow you to quickly assess whether a network is overly or not sufficiently complex for the problem at hand. For enhanced performance, *STATISTICA Neural Networks* supports a number of network customization options. You can specify a Linear output layer for networks used in Regression problems, or Softmax Activation functions for probability-estimation in Classification problems. If your data suffers badly from outliers, you can replace the standard Error function used in training with the less sensitive City-Block error function. Cross-entropy error functions, based on information-theory models, are also included, and there are a range of specialized activation functions, including Step, Ramp and Sine functions.

The Intelligent Problem Solver (an Easy-to-Use Wizard for Network Creation)

Included with *STATISTICA Neural Networks* is an *Intelligent Problem Solver* (accessible via a toolbar button), which will guide you through the network creation process.

The *Intelligent Problem Solver* can create networks using data whose cases are independent (standard networks) as well as networks that predict future observations based on previous observations of the same variable (time series networks).

A significant amount of time during the design of a neural network is spent on the selection of appropriate variables, and then optimizing the network architecture by heuristic search. *STATISTICA Neural Networks* takes the pain out of the process by automatically conducting a heuristic search for you.

Specifically, the *Intelligent Problem Solver* is an extremely effective tool that uses sophisticated nonlinear optimization techniques (including Simulated Annealing) to search automatically for an optimal network architecture. Why labor over a terminal for hours, when you can let *STATISTICA Neural Networks* do the work for you?

The *Intelligent Problem Solver* can also be used in a process of model building when *STATISTICA Neural Networks* is used in conjunction with some modules of the main *STATISTICA* system to identify the most relevant variables (e.g., the best predictors to be included and then tested in some Nonlinear Estimation model).

Training a Neural Network

As you experiment with architectures and network types, you rely critically on the quality and speed of the network training algorithms. *STATISTICA Neural Networks* supports the best known state-of-the-art training algorithms.

For Multilayer Perceptrons, *SNN* naturally includes Back Propagation with time-varying learning rate and momentum, case-presentation order shuffling, and additive Noise for robust generalization. However, *STATISTICA Neural Networks* also includes two fast, second-order training algorithms: Conjugate Gradient Descent and Levenberg-Marquardt. Levenberg-Marquardt is a very powerful, modern nonlinear optimization algorithm, and it is strongly recommended. However, as Levenberg-Marquardt is limited in application to fairly small networks with a single output variable, *STATISTICA Neural Networks* also includes Conjugate-Gradient Descent for more difficult problems. Both of these algorithms typically converge far more quickly than Back Propagation, and frequently to a far better solution.

STATISTICA Neural Networks' iterative training procedures are complemented by automatic tracking of both the Training error and an independent Verification error, including a Graph

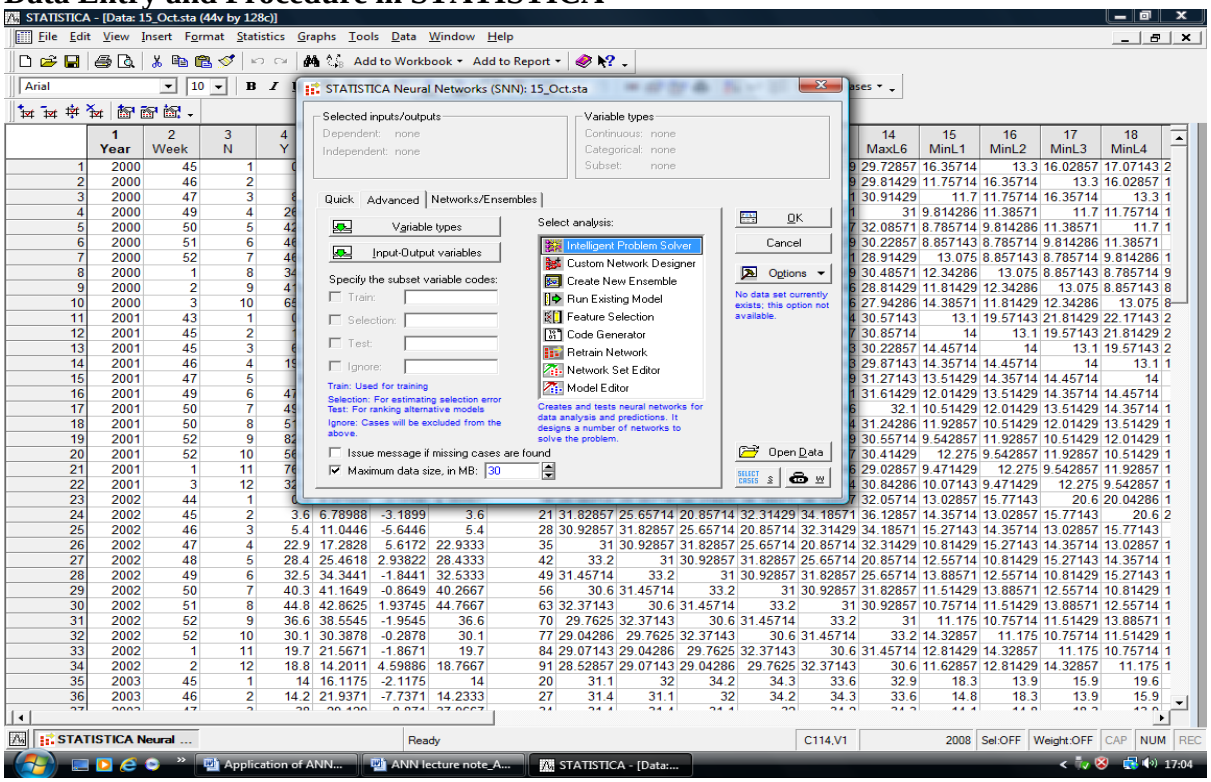
of the overall errors and a Bar Chart for individual case errors. Training can be aborted at any point by the click of a button, and you can also specify Stopping Conditions when training should be prematurely aborted, for example, when a target error level is reached or when the Verification error deteriorates over a given number of epochs (indicating Over-learning). If over-learning occurs, you needn't worry; *SNN* automatically retains a copy of the Best Network discovered, which can be retrieved at the click of a button. When training is finished, you can check performance against the independent Test Set.
STATISTICA Neural Networks also includes a range of training algorithms for other network architectures. Radial Basis Function and Generalized Regression networks can have Radial exemplar units and smoothing factors assigned by a variety of algorithms, including Kohonen training, Sub-Sampling, K-Means, Isotropic, and Nearest Neighbor techniques. The Linear output layers of Radial Basis Function networks can be fully optimized using Singular Value Decomposition, as can Linear networks.

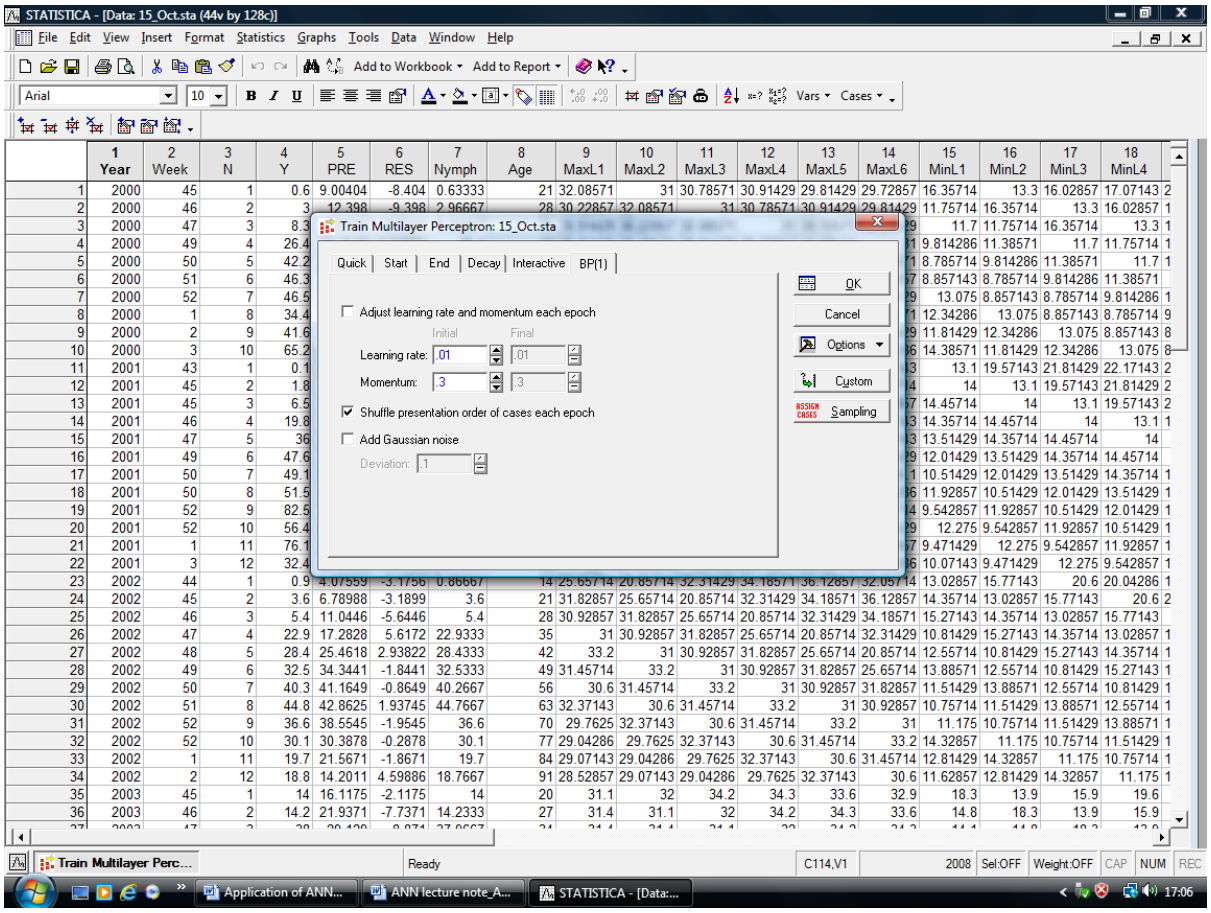
Hybridization of Network Structures.
STATISTICA Neural Networks also supports hybridization of network structures; for example, a modified Radial Basis Function network could have a first layer trained by Kohonen's algorithm, and a nonlinear second layer trained by Levenberg-Marquardt.

Probing and Testing a Neural Network
Once you have trained a network, you'll want to test its performance and explore its characteristics. *STATISTICA Neural Networks* uses a range of on-screen statistics and graphical facilities. All statistics are generated independently for the Training, Verification, and Test Sets. You can view the individual weights and activations in the network in convenient data sheet format; one click of a button can also transfer them into *STATISTICA* as spreadsheets. Copying these data to the Clipboard or saving them to a file can also be accomplished by clicking a button. A range of output types can be viewed: post-processed outputs, output neuron activations, and codebook vectors. Activations can also be viewed in Bar Chart form. The results of running individual cases, or the entire set, can also be viewed in data sheet form. Overall statistics calculated include mean network error, the so-called Confusion matrix for Classification problems (which summarizes correct and incorrect classification across all classes), and the Regression Error Ratio for Regression problems - all automatically calculated. Kohonen networks include a Topological Map window, which allows you to visually inspect unit activations, and to relabel cases and units during data analysis. There is also a Win Frequencies window to instantly locate clusters in the Topological Map. Cluster analysis can also be conducted using conventional networks together with *STATISTICA Neural Networks*' Cluster Diagram (shown below). For example, you can train a Principal Components Analysis network, and plot data through the first two Principal Components.

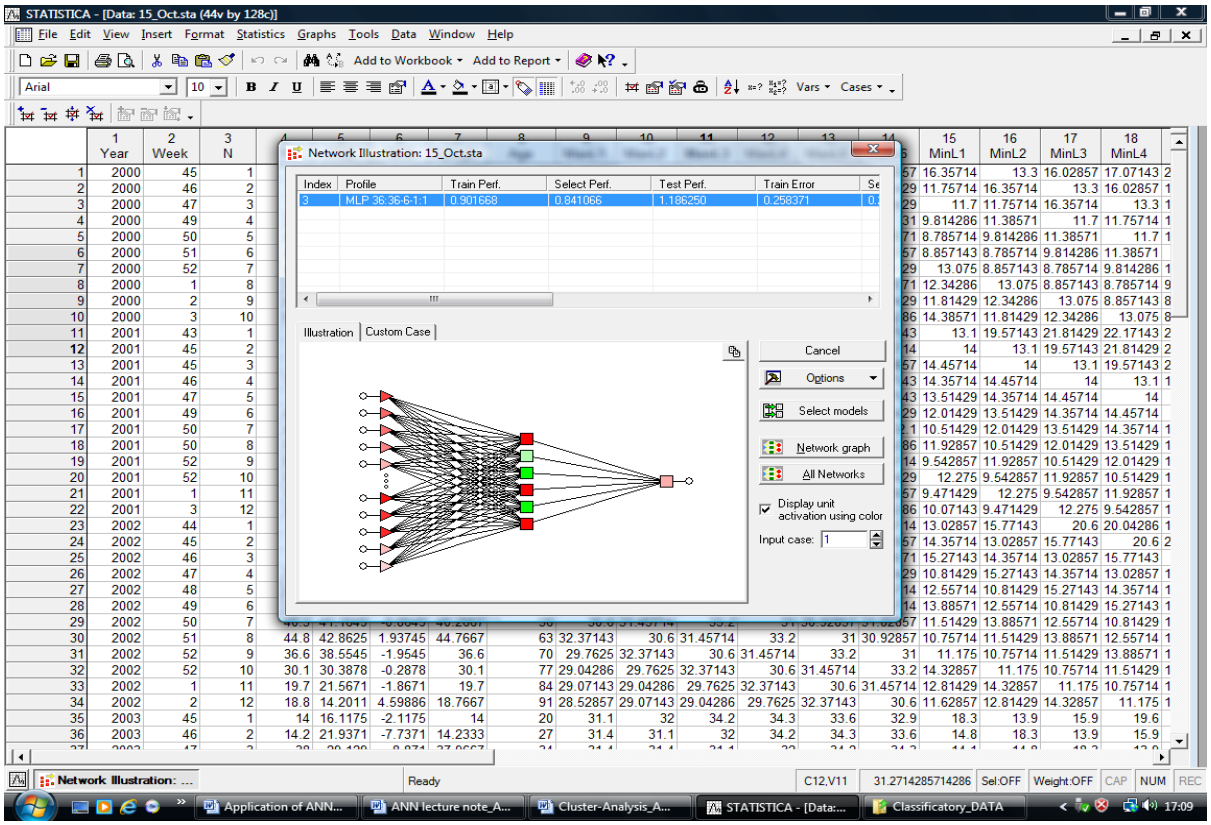
Network Editing and Modification
STATISTICA Neural Networks includes intelligent facilities to prune existing networks,and to join networks together. Entire layers can be deleted, networks with compatible numbers of Inputs and Outputs can be pipelined together, and individual neurons can be added or removed. These facilities allow *SNN* to support Dimensionality Reduction (for preprocessing) by the use of Autoassociative networks, and Loss Matrices (for minimum-cost decision making). Loss matrices are automatically included with Probabilistic Neural Networks.

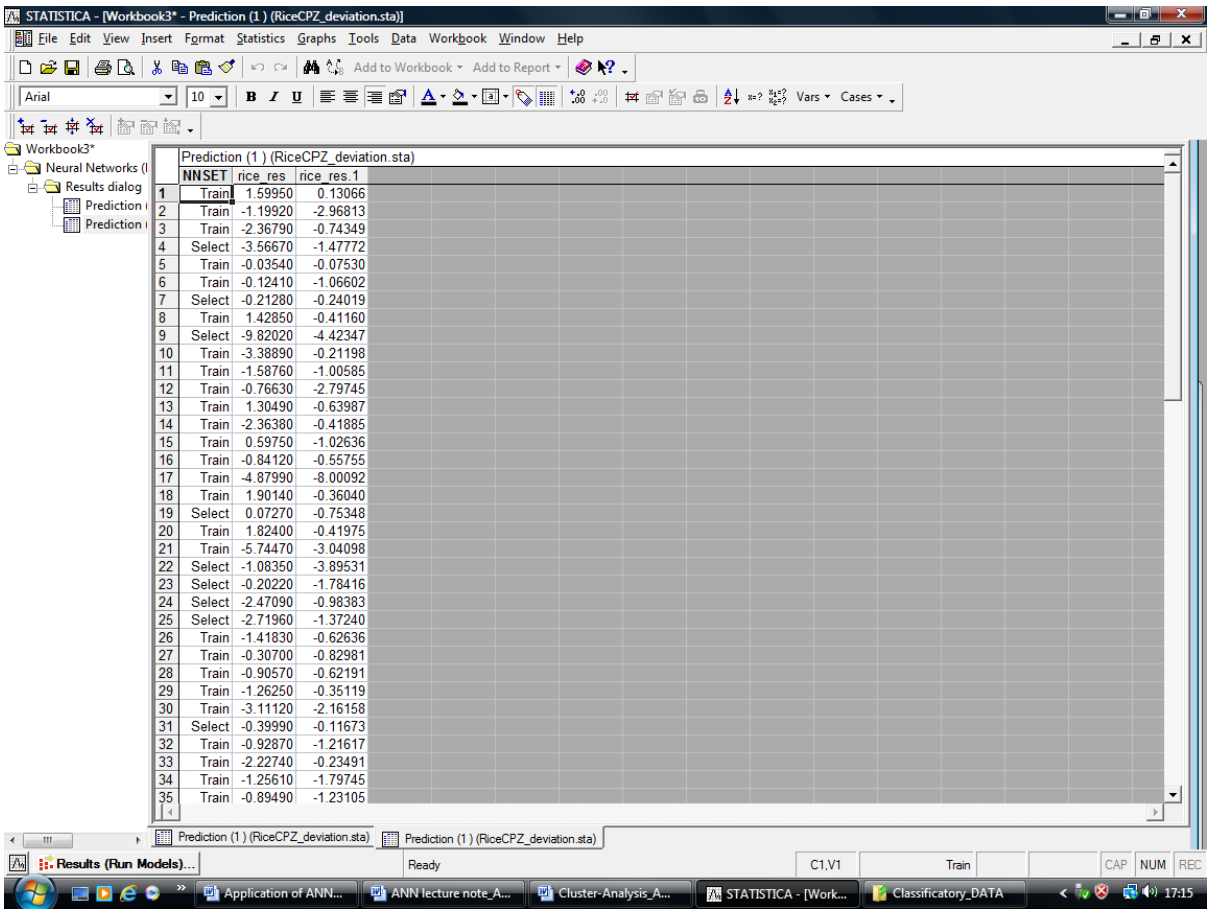
Data Entry and Procedure in STATISTICA





Output of STATISTICA





	NNSET	rice_res	rice_res.1
1	Train	1.59950	0.13066
2	Train	-1.19920	-2.96813
3	Train	-2.36790	-0.74349
4	Select	-3.56670	-1.47772
5	Train	-0.03540	-0.07530
6	Train	-0.12410	-1.06602
7	Select	-0.21280	-0.24019
8	Train	1.42850	-0.41160
9	Select	-9.82020	-4.42347
10	Train	-3.38890	-0.21198
11	Train	-1.58760	-1.00585
12	Train	-0.76630	-2.79745
13	Train	1.30490	-0.63987
14	Train	-2.36380	-0.41885
15	Train	0.59750	-1.02636
16	Train	-0.84120	-0.55755
17	Train	-4.87990	-8.00092
18	Train	1.90140	-0.36040
19	Select	0.07270	-0.75348
20	Train	1.82400	-0.41975
21	Train	-5.74470	-3.04098
22	Select	-1.08350	-3.89531
23	Select	-0.20220	-1.78416
24	Select	-2.47090	-0.98383
25	Select	-2.71960	-1.37240
26	Train	-1.41830	-0.62636
27	Train	-0.30700	-0.82981
28	Train	-0.90570	-0.62191
29	Train	-1.26250	-0.35119
30	Train	-3.11120	-2.16158
31	Select	-0.39990	-0.11673
32	Train	-0.92870	-1.21617
33	Train	-2.22740	-0.23491
34	Train	-1.25610	-1.79745
35	Train	-0.89490	-1.23105

References

Anderson, J. A. (2003). An Introduction to neural networks. Prentice Hall.

Cheng, B. and Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. Statistical Science, 9, 2-54.

Gaudart, J. Giusiano, B. and Huiart, L. (2004). Comparison of the performance of multi-layer perceptron and linear regression for epidemiological data. Comput. Statist. & Data Anal., 44, 547-70.

Hassoun, M. H. (1995). Fundamentals of Artificial Neural Networks. Cambridge: MIT Press.

Hebb,D.O. (1949) The organization of behaviour: A Neuropsychological Theory, Wiley, New York

Kaastra, I. and Boyd, M.(1996): Designing a neural network for forecasting financial and economic time series. Neurocomputing, 10(3), pp 215-236 (1996)

Patterson, D. (1996). Artificial Neural Networks. Singapore: Prentice Hall.

STATISTICA 6.1 StatSoft, Inc. USA

Warner, B. and Misra, M. (1996). Understanding neural networks as statistical tools. American Statistician, 50, 284-93.

Yegnanarayana, B. (1999). Artificial Neural Networks. Prentice Hall

Zhang, G., Patuwo, B. E. and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting,14, 35-62.