NNIGnets, Neural Networks Software

Tânia Fontes, Vânia Lopes, Luís Silva, Jorge Santos, Joaquim Marques de Sá

NNIGnets, Neural Networks Software

Tânia Fontes[1], Vânia Lopes[1], Luís M. Silva[1], Jorge M. Santos[1,2], Joaquim Marques de Sá[1]

[1] INEB - Instituto de Engenharia Biomédica, Campus FEUP (Faculdade de Engenharia da Universidade do Porto), Rua Dr. Roberto Frias, s/n, 4200-065 Porto, Portugal

[2] ISEP - Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal
{trfontes, vaniadlopes, lmsilva, jmfs, jmsa} @ fe.up.pt

Abstract. NNIGnets is a freeware computer program which can be used for teaching, research or business applications, of Artificial Neural Networks (ANNs). This software includes presently several tools for the application and analysis of Multilayer Perceptrons (MLPs) and Radial Basis Functions (RBFs), such as stratified Cross-Validation, Learning Curves, Adjusted Rand Index, novel cost functions, and Vapnik–Chervonenkis (VC) dimension estimation, which are not usually found in other ANN software packages. NNIGnets was built following a software engineering approach which decouples operative from GUI functions, allowing an easy growth of the package. NNIGnets was tested by a variety of users, with different backgrounds and skills, who found it to be intuitive, complete and easy to use.

Keywords: artificial neural networks, software application, freeware tool.

1 Introduction

NNIGnets includes several state-of-art tools of Artificial Neural Networks (ANNs) training and analysis, namely stratified Cross-Validation [1] with train/test statistics, Learning Curves, novel cost functions such as Generalized Exponential [2], Cross Entropy [3] and Shannon Entropy of the Error [4], [5], Adjusted Rand Index [6], [7] and Vapnik–Chervonenkis (VC) dimension estimation [8], [9], among others that are not usually covered by other software implementations. NNIGnets can be used for teaching, research or, under certain conditions, business applications. Over the last five years, this software has been used as a support tool for the practical sessions of the Neural Networks Summer School yearly held at Porto (http://www.isep.ipp.pt/nn/) and was tested by many users with different backgrounds and skills, who showed great interest in the functionalities available and agreed that it was a very complete package and easy to use. In the following sections we describe NNIGnets in detail.

2 Model description

NNIGnets is a freeware computer program developed in C# at INEB (PSI/NNIG) - *Instituto de Engenharia Biomédica* and is available for download from http://paginas.fe.up.pt/~nnig. It is composed by a number of menus that allow ANN (presently only Multilayer Perceptrons (MLPs) and Radial Basis Functions (RBFs)) configuration and evaluation. The Settings menu is used to set the essential information for ANN design by specifying the training and test data, the network architecture and the learning algorithm type. On the other hand, the Results menu provides access to the design outcomes such as tables and graphs that help the user to interpret the ANN performance. Although NNIGnets presently only provides MLPs and RBFs, the software structure was designed having in view an easy insertion of other ANN types within an uniform philosophy.

2.1 ANN Inputs – Settings Menu

The Settings menu is divided into three main submenus: Data, Architecture and Learning Algorithm. These three submenus are related to each other and several definitions affect one another.

The ANN inputs are specified through an input file chosen on the Data panel. The user uploads data through a graphical user interface, by selecting a text file (.txt). Each text file line must comply with either of the following formats: tab separated columns or semi-colon separated columns (possibly with spaces in between).

After data file uploading, the graphical interface shows the respective variable names and values. Each data column (variable) can be used as one of four types:

a) input: if the variable is to be used as input to the neural network (NN);
b) nominal output: if the variable is to be used as target in classification mode;
c) continuous output: if the variable is to be used as target in regression mode;
d) Ignore: variable to be discarded from the NN design.

The Pre-processing frame holds two options that provide two different types of data pre-processing: data randomization and data standardization. For data randomization, data rows are randomly shuffled. For data standardization, three methods are provided: range scaling (1), where data is scaled to an user-specified real interval $\lfloor a, b \rfloor$; mean centering (2), where data is centered around the mean; and standardization (3), corresponding to the usual zero mean and unit variance standardization. Let x_i be the $i-th$ original variable where $i = 1, \dots, i_{max}$ and $\tilde{x}_i$ the standardized x_i; a and b the lower and upper interval respectively; μ the mean value of x_i ; and σ the standard deviation value of x_i. Then we have:

$$\tilde{x}_i = \frac{b-a}{x_{i_{max}} - x_{i_{min}}}\left(x_i - x_{i_{min}}\right) + a \ . \quad (1)$$

$$\tilde{x}_i = x_i - \mu \ . \quad (2)$$

$$\tilde{x}_i = \frac{x_i - \mu}{\sigma} \ . \quad (3)$$

Another option available is the evaluation frame. Here the user can choose the type of performance evaluation between train/test evaluation, Cross-Validation evaluation and Learning Curves. In train/test evaluation data is divided into two subsets, train

and test. The subset sizes can be specified as percentages of the whole dataset. In Cross-Validation, the data is divided into a number of approximately equal sized subsets. The user defines the number of subsets and decides whether or not the class ratio of the data should be kept (the so-called S*tratified Cross-Validation*).

The Learning Curves allow the user to analyze ANN convergence and generalization abilities by performing multiple designs with incremental data sizes. The user defines the starting and ending sizes of the training set, the increment, the number of experiments in each subset and the format to be carried out. The format can be one of the following: i.i.d., where the instances for the test sets are randomly picked up from the data, and off-train-sets: the same process as i.i.d. but discarding instances already present in the training set.

The Architecture panel allows the specification between a MLP or a RBF network. Notice that we consider that each neuron in a given layer is connected to every neuron on the next layer and there is a weight assigned to each of these connections and a bias in each layer. The number of neurons in the input layer depends on the number of inputs chosen from the data and the number of neurons in the output layer depends on whether the user chooses classification or regression mode. In regression mode only one neuron is needed, since the network output is regarded as the expected value of the model at a given point in input space, whereas in classification mode the number of neurons depends on the number of classes: 1 output for two-class problems, c outputs for c > 2 classes problems, using 1-out-of-c coding.

For each ANN architecture, the user can choose different activation functions: Sigmoid, Hyperbolic Tangent or Linear Heaviside for MLPs and Gaussian or Multiquadric for RBFs. By default, the initial weights of the MLP architecture are set to zero, but the user can specify their initial values in two ways: by initializing the weights randomly within a specify range, or by specifying values using a text file. In latter the file must follow the format specified for input data. For RBFs the weights of neurons in the hidden layer are set according to x and y centroid coordinates. In both ANN architectures the user can change the network properties by inserting and removing layers or neurons and by changing the layer's activation function (i.e. setting weights). There is also an option for viewing the connections of a specific neuron. To ease the visualization, neurons are identified using a color scale: black for bias, red for input neurons, and blue for the remaining neurons.

The selection of the learning algorithm is available on the Learning Algorithm window. For MLP networks, either the Batch Back-Propagation or the Sequential Back-Propagation algorithm can be chosen, while for RBF networks a hybrid learning algorithm is available.

Unlike other ANN programs where only the Mean Squared Error (MSE) cost function is provided (4), NNIGnets avails (presently) three further possibilities: Exponential (Exp) cost function [2] (5), Cross-Entropy (CE) cost function [3] (6) and Shannon Entropy of the Error (HS) cost function [4], [5], (7), which have the following definition for two-class problems:

$$E_{MSE} = \sum_{i=0}^{n} (t_i - y_i)^2 \,, \tag{4}$$

$$E_{Exp} = \exp\left(\frac{\sum_{i=0}^{n} (t_i - y_i)^2}{k}\right) * k \,, \tag{5}$$

$$E_{CE} = -\sum_{i=0}^{n} t_i \ln y_i + (1 - t_i) \ln(1 - y_i) \,, \tag{6}$$

$$E_{HS} = -\frac{1}{n}\sum_{i=0}^{n} \log \hat{f}(t_i - y_i) \,, \tag{7}$$

where n is the number of patterns, t_i and y_i are the target and network output (respectively) values for pattern i, $k \neq 0$ is the Exponential cost function parameter and $\hat{f}(x)$ is an estimate of the error density obtained with the Parzen window method.

For MLP algorithms the following parameters can be adjusted: the learning rate which can be adaptive [10] or specified, the momentum, the number of epochs in one run and the minimum squared error. The hybrid learning algorithms only uses the number of weights which depends on the number of neurons on the hidden layer.

2.2 ANN Outputs – Results Menu

The Results menu has several submenus: the Classification Matrix, the Error Graph, the Error Surface and the Decision Border.

The Classification Matrix menu provides the classification error matrices resulting from the train/test learning process. The classification results can be adjusted to several types of priors: equal priors, with an equal weighting factor for every class, structured priors, with weights proportional to the set class sizes, and specified priors with weights specified by the user. The output varies according to the evaluation type selected in the Data panel: train classification matrix, whenever the user chooses 100% of the available data for the training set; train and test classification matrices, Cross-Validation, or the Learning Curves. NNIGNets software provides an estimate for the classification error rate (Error mean). Several performance indicators for two-class problems are also included (Sensitivity and Specificity, Balanced Error Rate and Adjusted Rand Index [6], [7]). Let c be the number of classes, j the class index, m_j the number of misclassified cases in class j, n_j the number of cases in class j and p_j the prior of class j; let fp be the number of false positives, fn the number of false negatives, tp the number of true positives and tn the number of true negatives. Then we have:

$$\text{Error Mean} = \sum_{j=0}^{c} \frac{m_j}{n_j} p_j \,. \tag{8}$$

$$\text{Sensitivity} = \frac{tp}{tp + fn} \,. \tag{9}$$

$$\text{Specificity} = \frac{tn}{tn + fp} \,. \tag{10}$$

$$\text{Balanced Error Rate (BER)} = \frac{\frac{fp}{tn+fp} + \frac{fn}{tp+fn}}{2} \,. \tag{11}$$

$$\text{Adjusted Rand Index} = \frac{\binom{n}{2}(tp+tn) - [(tp+fn)(tp+fp)+(fp+tn)(fn+tn)]}{\binom{n}{2}^2 - [(tp+fn)(tp+fp)+(fp+tn)(fn+tn)]} \,. \tag{12}$$

The outputs resulting from the training process are shown on the Output panel, and include the ANN output, the target value, the deviation value and the class confidence value (H). The misclassified instances are represented in dark grey. On the options frame the user can choose which output layer will be displayed. For two-class problems the target value can be either in [0, 1] or [-1, 1] depending on the activation function chosen. For multi-class problems the targets are coded in a 1-out-of-c (where c is the number of classes) mode. The class confidence value H, as a function of each pattern is given by [11]:

$$H = -\sum_{\Omega} P(\omega_i|x)\ln(\omega_i|x)\,, \tag{13}$$

where:

$P(\omega_1|x) = y_i(x)$, $P(\omega_0|x) = 1 - y_i(x)$, for two-class problems;

$P(\omega_i|x) = \frac{y_i(x)}{\sum_{\Omega} y_i(x)}$, for multi-class problems.

For classification problems, the confidence level CC is given by:

$$CC = \begin{cases} \ln(c) - H, & \text{if well classified} \\ -\ln(c) - H\,, & \text{otherwise} \end{cases}, \tag{14}$$

where c is the number of classes.

This Output panel also shows the Class Confidence Mean (Mean H) which is the average value for all cases, such that $-\ln(c) \leq H \leq \ln(c)$.

The Error Graph panel is one of the most useful tools available in this software. This panel is composed by a graph that shows the evolution of the mean classification error for the select cost function along the train/test epochs and helps the user to identify the minimum number of epochs for a specified ANN configuration.

If Cross-Validation method is selected, this graph also shows the MSE standard deviation for train and test over the number of epochs. Right clicking on the graph, a pop-ups menu provides several options such as saving the graph as an image on a pre-specified location, or showing x and y coordinates (visualization mode) when the user moves the mouse over the graph line.

From the Results menu the user can also choose to display the Decision Border and the Error Surface. The first is a two dimensional representation of the ANN behavior for a given range of values for two variables/characteristics specified by the user. Patterns are represented in different colors depending on the class and the border curve, which divides the classes, is represented in black. The user can set the grid resolution, the dots size and can also choose to display a grid with class labels. The Error Surface is a three dimensional plot of the ANN cost function for any pair of weights. Each ANN output can be regarded as a function of n input variables, which form a response surface on an n+1-dimensional space.

Another aspect that is not usual to find in other similar software is the estimation of the VC-dimension, which is part of the VC-theory [8]. VC-dimension assesses the model complexity and the generalization ability of a learning machine. Phatak demonstrates the inter-relationships between generalization and the VC-dimension of feed-forward ANNs [12] and a practical setup to estimate the VC-dimension of a

binary classifier is proposed by Vapnik et al. [8]. NNIGnets uses the optimized design proposed by Shao and Li [9] to obtain a more accurate estimate of the VC-dimension. The user selects the option VC-dimension from the Actions menu, which is only available for two class datasets, since the empirical process proposed in [9] is defined only for binary classifiers. On the VC Dimension window two inputs are defined: the number of experiments to be performed and the initial guess of the VC-dimension (h), which is a non negative number.

NNIGnets also includes a help tool that explains the software usage, describing each menu option, as well as the meaning of the fields on each panel.

3 Conclusions

The NNIGnets software has been used as a teaching tool for the practical sessions of the last editions of the Neural Networks Summer School, held in Oporto. The School involved over thirty participants of diverse backgrounds and practical interests from several countries. NNIGnets raised great interest among the participants who found it very intuitive, pleasant and easy to use, even for an inexperienced user. Since it offers a large set of options, including new functionalities that are not usually found in other similar software, and participants agreed it to be very complete and appealing.
New tools and ANN types are planned to be included in NNIGnets in the near future.

4 References

1. Hastie, T., Tibshirani, R., Friedman, J.: The Element of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2.º edition (2009)
2. Silva, L., Marques de Sá, J., Alexandre, L.A.: Data Classification with Multilayer Perceptrons using a Generalized Error Function. Neural Networks 21(9), 1302-1310 (2008)
3. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, (1995)
4. Silva, L., Marques de Sá, J., Alexandre, L.A.: The MEE Principle in Data Classification: A Perceptron-Based Analysis. Neural Computation 22, 2698-2728 (2010)
5. Silva, L.: Neural Networks with Error-Density Risk Functionals for Data Classification. PhD thesis, University of Porto (2008)
6. Hubert, L., Arabie, P.: Comparing Partions. J. of Classification 2, 193-218 (1985)
7. Santos, J.M., Ramos, S.: Using a Clustering Similarity Measure for Feature Selection in High Dimensional Data Sets. In: Innovation and Sustainable Development in Agriculture and Food 2010, vol. 1, pp. 900-905. IEEE Computer Society Press, Montpellier (2010)
8. Vapnik, V., Levin, E., Le Y.C.: Measuring the VC-Dimension of a Learning Machine, Neural Computation 6(5), 851-876 (1994)
9. Shao, X., Li, W.: Measuring the VC-Dimension using optimized experimental design. Neural Computation 12, 1969-1986 (2000)
10. Santos, J.M.: Data classification with neural networks and entropic criteria. PhD thesis, University of Porto (2007)
11. Wan, E.A.: Neural Network Classification: A Bayesian Interpretation. IEEE Tr NN (1990)
12. Phatak, D.: Relationship between fault tolerance, generalization and the Vapnik-Chervonenkis (VC) dimension of feed-forward ANNs. Proceedings of the International Joint Conference on Neural Networks (IJCNN) (1999)