Dynamic topology adjustment algorithm for MLP networks

Robert Sulej[1)], Krzysztof Zaremba[1)] , Krzysztof Kurek[2)]

[1)] Institute of Radioelectronics, Warsaw University of Technology, Warsaw, Poland
[2)] Soltan Institute for Nuclear Studies, Warsaw, Poland

http://www.ire.pw.edu.pl/~rsulej/NetMaker
e-mail: rsulej@ire.pw.edu.pl

Method

Choosing the network topology (e.g. number of hidden units) is a crucial part of the network preparation. In virtually all real-live cases there is no clue what number of hidden units is the minimum that allows to solve the problem and when it becomes too large and causes overtraining effect.

The goal of our algorithm is to establish the proper number of neurons in each hidden layer of the MLP network by adding and removing neurons during the network training. Between the structure modifications network is trained with a back-propagation algorithm (we use quick-prop, but other algorithms are also applicable). Proposed techniques of the structure modifications are not harmful to the current state of the network and the training progress. Only useless neurons are removed and new neurons may only improve the network performance. Thus, network structure may be modified at any time of the training. Additionally, structure modifications change the weight space which is favorable for back-propagation algorithms giving a chance of escaping from local minimums.

Insertion of new hidden units

The idea of a network growth is borrowed from the Cascade-Correlation network. Like in this model, we use a pool of neuron candidates (initial weights are randomized, but we plan to use some heuristics for at least one neuron in the pool). Their weights are trained while the rest of the network weights are frozen. When training is completed the best candidate (resulting with the smallest network error) is chosen to extend the network structure (Fig.1). It is done only if this neuron decreases the network error, otherwise the network remains unchanged. New neurons in Cascade-Correlation model become a part of the fixed structure. In contrary, in our algorithm, all network weights are retrained after successful insertion of new neuron. New neurons are not fixed on particular features of the training data; they are just pre-trained to fit into the existing structure in a best possible way. This allows a better integration of the new neuron with the network structure and results in a smoother network response.

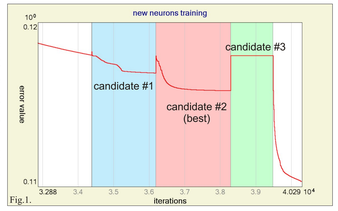


Fig.1.

Pruning redundant neurons

Most of the network training algorithms start from a random neuron weights. We observed that during the training process not all of the hidden units are fully utilized. Also some neurons responsible for overtraining effect may be recognized and removed from the network structure. We propose three types of redundant neurons that can be safely removed.

- joining *twin* neurons

Neurons that respond to any excitation with nearly the same output value likely can be joined into a new single unit. Such a situation often corresponds to presented in Fig.2, where class borders l_1 and l_2 may be replaced with l_{12}.

Weights connecting the new neuron with the following layer should be calculated as sums of corresponding weights of the removed neurons. A measure of similarity between neurons can be used as a parameter for the pruning procedure; for this purpose we use following simple formula:

$$d_{twin} \quad \frac{\|\mathbf{w}_a \quad \mathbf{w}_b\|}{\|\mathbf{w}_a \quad \mathbf{w}_b\|}$$

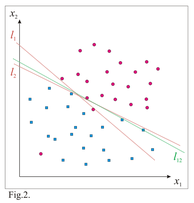


Fig.2.

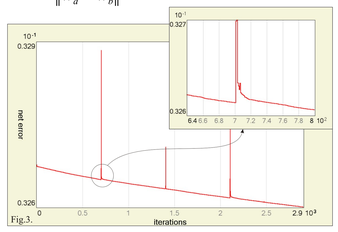


Fig.3.

where $\mathbf{w}_a$ and $\mathbf{w}_b$ are the vectors of the input weights of the tested neurons.

Fig.3 shows an example of joining *twin* neurons. Network in this example was trained with a large, fixed structure until error value has stabilized. Then *twins* pruning was enabled. Error value returns near to its previous value in a few iterations after each joining of neurons.

- removing *constant* neurons

Because of random character of initial weights some neurons learn irrelevant statistical fluctuations (like in the example in Fig.4, where neuron responsible for border *l* has no chance of escaping from that state), or just produce nearly constant output for all training vectors. These neurons are detected by a small standard deviation of their activation for training vectors when compared to full possible range of activations. When *constant* neuron is removed, its mean output value is included in a bias of neurons in a following layer. Fig.5 shows a fragment of the training process, where *constant* neuron is removed (the same oversized network is used as in the joining of *twin* neurons example).

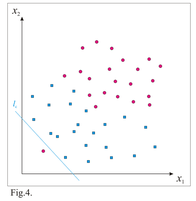


Fig.4.

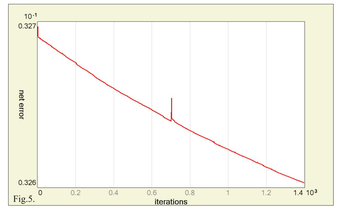


Fig.5.

- removing *dead* neurons

When initial weights of a hidden unit cannot be utilized by the network during the training, one of possible scenarios is that weights connecting such neuron with neurons in following layer are suppressed. If the norm of neuron output weights vector is much below the mean value for all neurons in the layer, tested neuron can be removed without influence on functioning of the whole network. In our tests we observed that this type of redundant neurons appears most frequently. Fig.6 shows an example of removing dead neuron (again, the same oversized network is used as in previous examples).

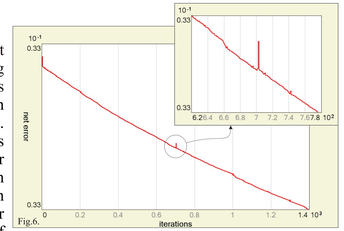


Fig.6.

Tests

Various tests and benchmarks have been performed using proposed algorithm. We developed our own simulation environment (Fig.7), which was used for testing the algorithm and also for designing networks for real-life applications (all plots on the poster were generated with this tool; we develop it as a freeware and it is available on our web site). To measure the network quality in classification tasks we use purity and efficiency of selection defined as:

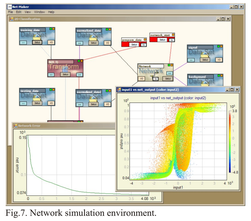


Fig.7. Network simulation environment.

$$purity = 100\ N_{sig}(OutputSet) \,/\, (\ N_{bkg}(OutputSet) + N_{sig}(OutputSet)\ )$$
$$efficiency = 100\ N_{sig}(OutputSet) \,/\, N_{sig}(InputSet)$$

where $N_{sig}(Set)$ and $N_{bkg}(Set)$ are the numbers of the signal and background events in the *Set*; *InputSet* is the set of classified events and *OutputSet* is the set of events with the network answer above a given threshold.

In this section we present two examples of class separation. First example (Fig.8) compares three networks: 1) network obtained with dynamic size adjustments, initiated with minimal structure; 2) network with a fixed, oversized structure; 3) oversized network with redundant neurons removed. This shows that however classification results are almost identical, using growth and reduction techniques simultaneously gives smallest network size and smoothest network output. Second example (Fig.9) is the two-spiral benchmark in a fuzzy version. MLP network trained with proposed algorithm is compared to Cascade-Correlation network model, which was the origin of the idea of the network growth in our model. This example shows that retraining the network after successful insertion of the new neuron improves its integration with the existing structure.

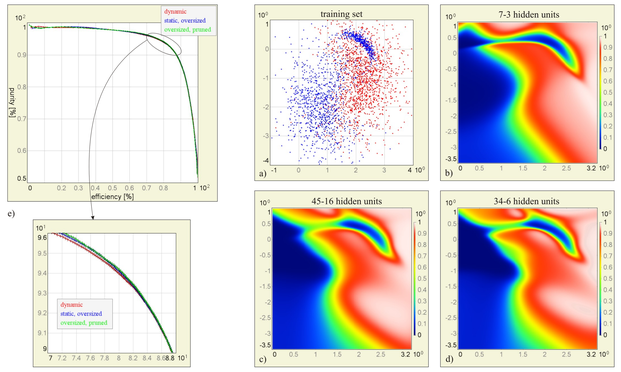


Fig.8. Simple class separation example: a) training set; network output: b) network obtained with dynamic size adjustments; c) network trained with fixed, oversized structure; d) oversized network with redundant neurons removed; e) purity-efficiency plot obtained for testing set.

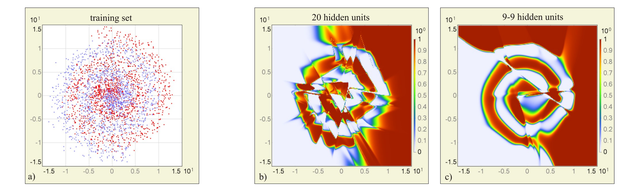


Fig.9. Two-spiral separation (fuzzy version): a) training set; network output: b) Cascade-Correlation network; c) dynamic MLP network.

Applications

Proposed algorithm have been successfully examined on various tasks of a data analisys of high-energy and neutrino physics experiments (SMC, COMPASS, ICARUS). In this work we present two application examples - classification of particle interaction type and estimation of one of the interaction parameters.

C O M P A S S

(***CO**mmon **M**uon **P**roton **A**pparatus for **S**tructure and **S**pectroscopy*)

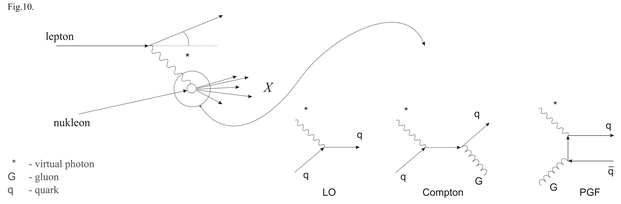
COMPASS is a high-energy physics experiment at the Super Proton Synchrotron (SPS) at CERN in Geneva, Switzerland. The purpose of this experiment is the study of hadron structure and hadron spectroscopy with high intensity muon and hadron beams. This experiment continues program that was initiated by observation (EMC experiment) that only a small fraction of the proton spin is carried by the spin of the quarks. One of the hypothesis to be tested is that a significant fraction of the nucleon spin comes from polarized gluons*.

*Gluons and quarks are the elementary particles which form the structure of nucleon (protons and neutrons).

Selection of photon-gluon fusion events

A process that involves gluons is required to measure the gluon polarization. Such a process, so-called photon-gluon fusion (PGF), occurs in a deep inelastic lepton-nucleon scattering among the other processes that we consider as a background. Diagrams of the PGF and two lowest order processes (most probable) are presented in Fig.10. As we cannot observe directly quarks and gluons, it is not straight-forward to distinguish which process was the origin of final particles (*X*) that appear in the detectors after scattering. Two approaches are used to select PGF from background. First is looking for pairs of hadrons with high transverse momentum (p_t) which likely appear as a product of q-$\bar{q}$ pairs produced in PGF. This provides a good separation from the virtual photon absorption process (LO), where the only source of p_t is small momentum of quarks inside the nucleon. Another approach is to look for charm particles (containing c quark) which may appear in PGF and not in LO and Compton processes as they are interactions with nucleon quarks (practically only u and d quarks). Unfortunately, heavy c quarks are also rarely produced.

Fig.10.



In this work we present selection of PGF events from high-p_t sample. As an input for network we use 11 kinematical variables describing scattered muon and produced hadrons. Network is trained with our algorithm to give high output (0.95) for PGF events and low (0.05) for LO and Compton events. Distribution of the network output (Fig.11) shows that PGF process is relatively easy to separate from LO events, while Compton events are much more difficult to classify correctly due to the similarity of process products to products of PGF.

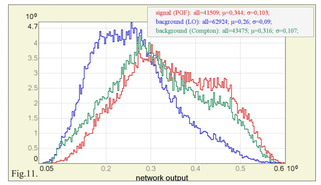


Fig.11.

Fig.12 shows purity-efficiency curves obtained for network classification and classification based on manually optimized cuts (though obvious limitations this technique still is popular due to possible connection between chosen cuts and knowledge about the interaction model).

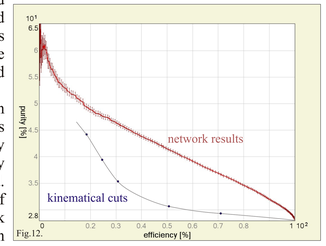


Fig.12.

Applying automatic hidden neuron insertion and pruning to the training process showed that networks with manually adjusted size of hidden layers (previously used for this task) tend to be oversized. Significant reduction of the number of attempts has been achieved as network results are less dependent on the initialization than in case of fixed sized networks.

Estimation of a_{LL} parameters

Another application of the presented algorithm is the estimation of the so-called „analyzing power" a_{LL} needed in the data analysis in COMPASS experiment. Gluon polarization is proportional to a_{LL} and depends on the measured asymmetry. Analyzing power cannot be directly calculated from the measured quantities (full information set contains variables describing interaction kinematics at the quark level) and some kind of approximated formula (parameterization) based on Monte-Carlo simulation is needed.

For test purposes network was trained with all variables required to calculate a_{LL}. Achieved high correlation between network output and true a_{LL} value (R_{xy} coefficient in Fig.13) proves that network is capable of learning the formula. When input variables are limited to measured quantities only, correlation coefficient drops (Fig.14), but still is higher than in case of a_{LL} parameterized in a standard way (Fig.15).

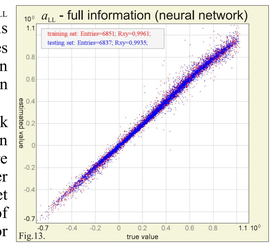


Fig.13.

Training algorithm allowed to keep the network sizes very compact (5 and 2 units in 2 hidden layers in presented case; 5 inputs and 1output neuron are determined by the task). Another advantage over standard parameterization is smaller training set required to prepare the network (about an order of magnitude) and therefore much shorter estimator preparation time.

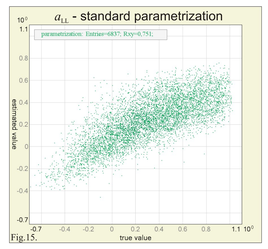


Fig.15.

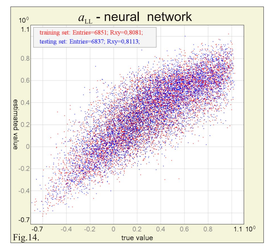


Fig.14.