Stopping Rules and Estimation of the Generalization Error In Neural Classification to Multiple Categories

Ashot Chilingaryan and Ararat Vardanyan

Yerevan Physics Institute, Cosmic Ray Division, Armenia

Abstract:

The most common drawback in Feed-Forward Neural Networks (FFNN) performance is the limited number of training and test samples. Usually, in many NN applications we can't simulate enough simulation trials, therefore, we never can be sure that we use sufficient number of examples to learn a general problem and not the specific training data set. As we are not sure that training samples used reflect all variability of the considered phenomenon, learning of specific selected examples too well is not desirable. What we need is to generalize from the used training set to entire problem. Therefore, the strategy, checking the expected performance of FFNN during training is of crucial importance. The strategy connected with the Prediction Risk estimation, that reuse data and gives unbiased estimate ever for small sample sets, is generalization of one-leave-out-for-the-time estimate. Estimates of the prediction risk offer a sound basis for assessing the generalization performance of the model and can be used as a tool for architecture selection and constructing of the stopping rule. Therefore, it is important to check the training results not with the "training error", but with the "generalization error", represented by the prediction risk. In present paper the over-training effects are investigated and the possibility of selecting the most appropriate NN architecture for a problem solution using the Final Prediction Risk (FPE) estimate is demonstrated.

Introduction:

The classification mode of the neural mapping and the recovering of the unknown functional dependence are the main realizations of neural mapping possibilities implemented by FFNN. We will consider the classification problem bellow. A primary advantage of mapping networks over classical statistical analysis methods is that the FFNN have more general (algorithmic) functional forms than classical statistical methods can effectively deal with [5]. The FFNN are free from depending on linear superposition or orthogonal functions and can mimic sophisticated stochastic mechanism whereby the Nature generate the data. Therefore, in contrast with classical classification problem, we've to specify not the particular member of known analytic family of functions, rather the non-parametric algorithm (classificator), which generalizes the unknown mapping rule, implementing learning strategy on the training sample. The classification learning strategy will be based on the fundamental notion of the generalization. As we are not sure that training samples used reflect all variability of the considered phenomenon, learning of specific selected examples too well will effect the so called over-training, when the NN performance on training sample is much better than on the independent (test) sample. What we need is to generalize from the used training set to entire problem. Therefore, the strategy checking the expected performance of FFNN during training is of crucial importance. The strategy, proposed in [5] is connected with the Prediction Risk as a performance measure. In general, the particular FFNN model can be specified (indexed) by the λ parameter:

$$\lambda \subset \Lambda \equiv (V, G, W), \tag{1}$$

- Where $V \subset v$ notes a chosen subset of variables from the set of all possible variables V;
- G is a selected architecture from the class of possible architectures ς;
- and W is the set of net parameters (weights).

The prediction risk $P(\lambda)$ is defined as expected net performance on a finite test set:

$$P(\lambda) \approx E\{\frac{1}{M}\sum_{i=1}^{M}(t_j^* - 0_\lambda(u_j^*)^2\}, \tag{2}$$

where the u_j^* is the vector of input parameters, t_j^* is the corresponding true output and (t_j^*, u_j^*) were not used in training O_λ - is the trained network output. The strategy consists in the selection of the particular λ from the model space Λ, which minimizes an estimate of the prediction risk.

The procedure of the prediction risk estimation, that reuse data and gives unbiased estimate even for small sample sets, is connected with the generalization of one-leave-out-for-the-time estimate. The *k-fold cross-validation,* introduced by Geisser [2] and Wahba [4], instead of leaving only one event, delete larger subsets from

training sample. Let the training sample $\tilde{p} \equiv (t_j, u_j), j = 1, M)$ be divided into K randomly selected disjoint subsets of the equal sizes $M_k = M / k$, denoted by $\tilde{p}$. And the $\tilde{P}_i$ with denote the training sample with deleted i-th sub sample $\tilde{P}_i$. Then the cross-validation mean error (MSE) for the selected subset $\tilde{P}_i$ is defined as:

$$MSE_{\tilde{p}_i}(\lambda) = \frac{1}{M_k} \sum_{(t_j, u_j) \subset \tilde{p}_i} (t_j - o_{\lambda, \tilde{p}_i}(u_j))^2, \tag{3}$$

and

$$MSE(\lambda) = \frac{1}{k} \sum_{j=1}^{k} MSE_{\tilde{p}_i}(\lambda). \tag{4}$$

Typical choices of K are 5 and 10 depending on the training sample size. An useful modification of cross-validation mean square error, penalizing complicated networks comprising many hidden units, is the Akaike's final prediction error (FPE) [5]. For large enough training sets it takes the following form:

$$P(\lambda) \equiv MSE(\lambda)\left(1 + 2\frac{NTOT}{M}\right), \tag{5}$$

where NTOT is the total number of networks weights. Just this expression is recommended by authors of [3] as an estimate of prediction risk. Estimates of the prediction risk is directly connected with generalization performance of the model λ and can be used as a tool for architecture selection when the V (the training pattern set) is fixed. It is important to check the training results not with the "training error", but with the "generation error", represented by the prediction risk for constructing the stopping rule.

The cross- validation procedure and estimation of the EPE:

Described above technique with appropriate defined error function was used for the different NN training for the 3-way classification problem and the FPE estimation of these networks for different sample sizes. The simulated toy-problem was to classify the events from two dimensional Gaussian population with means 1,2 and 3 for the first second and third classes respectively:

$$f(\vec{x}_j) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2}\left(\frac{\vec{x}_j - \mu}{\sigma}\right)^2} \tag{6}$$

where $\vec{x}_j$ is the input vector, $\vec{x} = [x_1, x_2]; j = 1...50000, \sigma = 1$ for all three classes.

Six sample with different sizes were used to train the NN to generalize the 3-way classification of random events from Gaussian populations. Particularly, samples

containing 100, 200, 500, 1000, 2000, and 5000 events per class were used. Five different NN architectures were constructed using 2 input nodes and one output node for all configurations, and the number of hidden nodes varying from 2 to 10 nodes. The FPE of NN with different architectures were estimated for these sample sizes and compared with the training error.

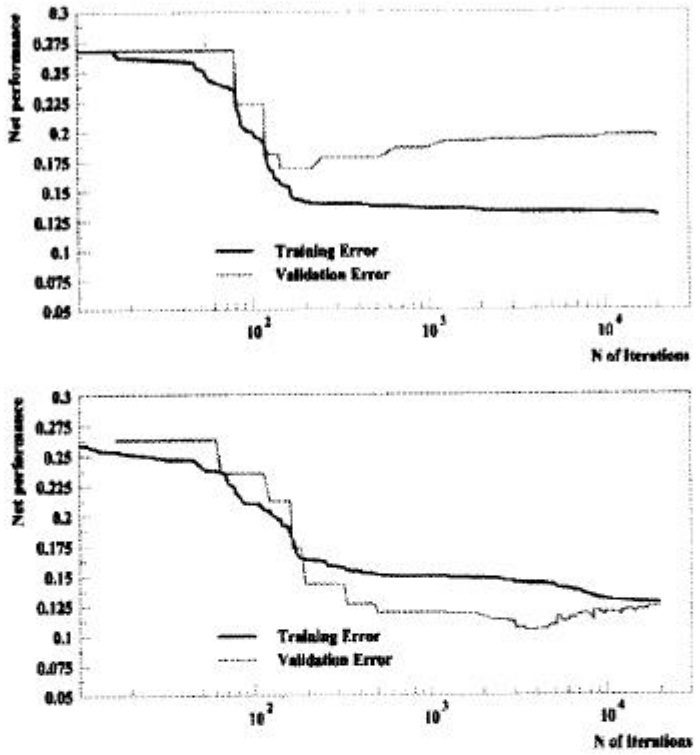

Figure1: The training process of the NN with architecture 2:2:1,
TS consists of 300 events (30 validation events, 10-fold cross-validation)

For large enough training sets the usage of the one-leave-out cross-validation can be very much time consuming procedure, therefore we have used a 10-fold cross-validation for each sample size. Particularly, we remove 10% of the training events form TS and use them as a validation sample (VS).

After the training process is finished (the maximum number of iterations is reached or some stopping criteria is applied), the removed 10% of events are taken for the validation purposes. So, each sample is reused for the training process 10 times until all separate 10% of the sample are subsequently removed and used as a validation sample. Note, that in each stage of the cross-validation the VS events are disjoint. The MSE in each stage is calculated by the formula 4, and the expected MSE for the given sample is calculated by formula 5 for both training and validation samples. Then the formula?? Is used to calculate the PR of NN on validation sample, this is used as the Final Prediction Error.

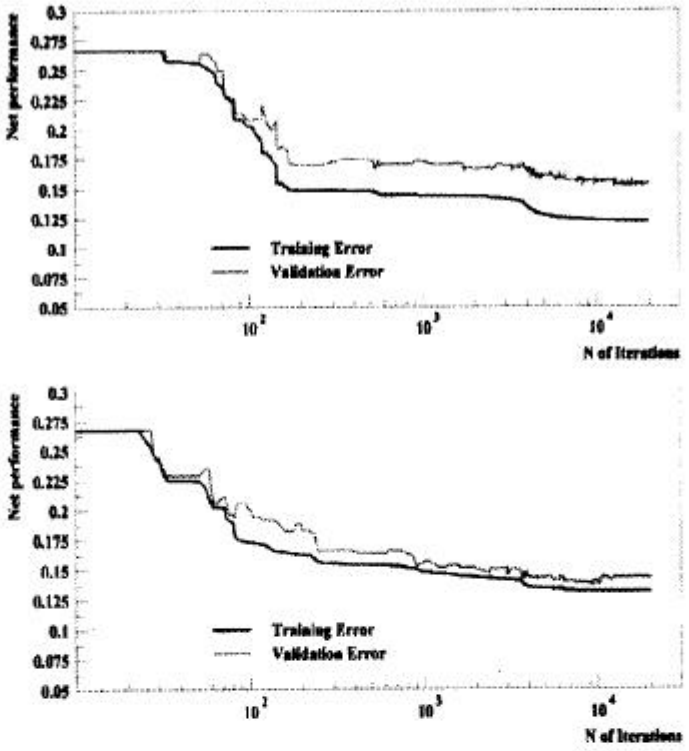

Figure2: The training process of the NN with architecture 2:6:1, TS consists of 600 (above part) and 1500 (bellow part) events (60(150)-validation events)

Figures 5 and 6 display the training process development for different sample sizes. From Figure 1 one can see that when the TS size is very small the over-training occurs even with the smallest NN architecture (the training error is decreasing during further training, while the error on independent validation sample is reached to some local minima and then is increasing).

That means having very small TS one could not expect to obtain a generalized NN, because the net weights are easily tuned to classify rather well the small number of training patterns only. Therefor, in such cases, the NN should not be trained with many training epochs (presenting the training patterns to the NN many times). But for the question, when to stop the training process, the common (general) answer has not found yet, which could be applied for any model λ,being independent of the problem to be solved, sample size and NN architecture. As it is Easy to see from upper and lower boxes of the Figure 2, using the same sample and the same NN architecture, in different stages of the cross-validation procedure the over-training has occurred in completely different ways in terms of number of training iterations and MSE on training and validation samples.

one possibility is to stop the training when the validation error starts increasing (or this increase is larger than some threshold value), but this criteria is not desirable due to the random character of the over-training effect. As it is demonstrated on the

Figure 2, the random increase of the validation error is occurred rather often. The curve of the validation error has many local minima, but during further training the MSE is decreasing and the best point in multidimensional space of the NN weights is obtained at the end of training process.
The acceptable technique for this purpose is not to stop the learning process at all, but to apply the following procedure:

- After each successful iteration of the learning process the net error (MSE) is calculated for the test sample
- If the test error value is less than the value in previous step the NN weights obtained at the current training step are stored
- Else, the NN weights obtained at the previous step are kept
- At the end of the training process the weights which give a best results (minimal MSE) on the test sample are found and used a final best weights for NN.

Discussion of the Results:

In previous section the procedure of the NN FPE estimation was described. This procedure gives an estimate of the FPE of the NN on some TS, but does not give any information about the error bounds of the estimated FPE. It is important to obtain not only unbiased estimate of the FPE but to estimate the errors of this estimation as well. With aim of this we have applied the cross-validation procedure for each of the 6 different samples 10 times, of course using 10 different (from the same general population, but completely disjoint) samples for each size. So, having 10 different estimates of the FPE for each network architecture we calculate the mean and the standard deviation of FPE by the following formulas:

$$\mathrm{E}(\mathrm{FPE}(\lambda)) = \frac{1}{N}\sum_{i=1}^{N} FPE_i(\lambda) \tag{7}$$

$$\sigma(FPE(\lambda)) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\{FPE_i(\lambda) - E(FPE(\lambda))\}^2} \tag{8}$$

where N is the number of cross-validation procedure for the given sample size and NN architecture, and i=1, N.

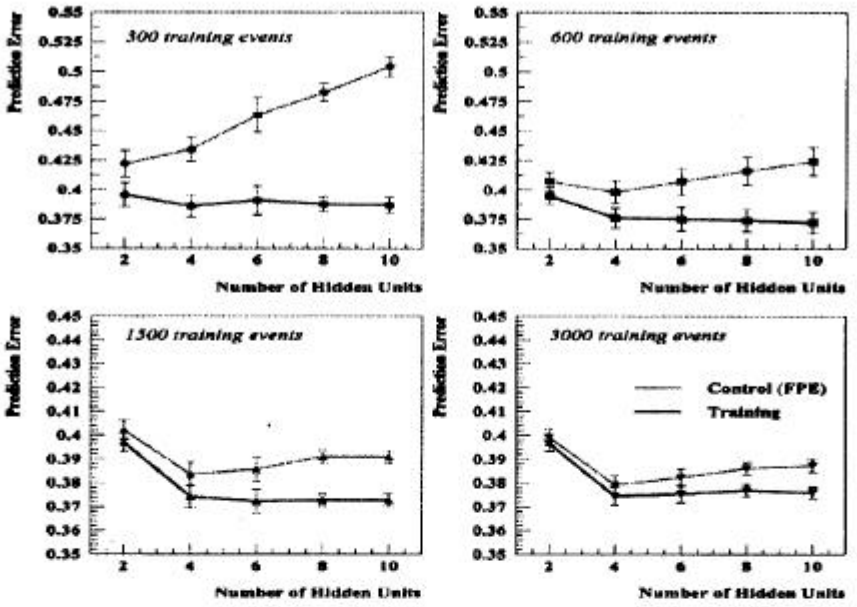

Figure 3: The dependence of the FPE and training errors on the number of hidden nodes in NN for different sample sizes

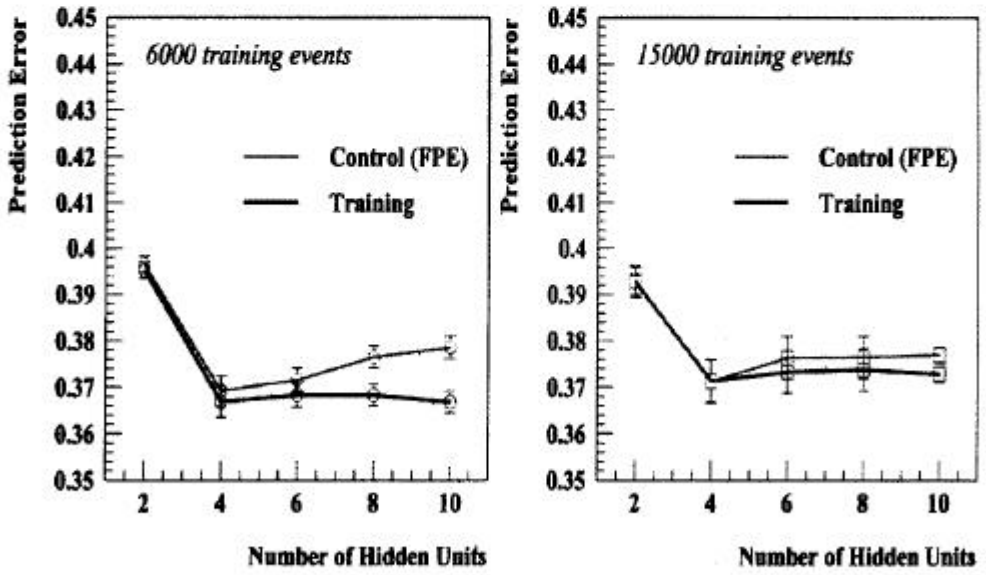

Figure 4: The dependence of the FPE and training errors on the number of hidden nodes in NN for different sample sizes

Figure 4 displays the dependence of the FPE on the used NN architecture for different sample sizes. As one can see from this figure for the smallest sample size (300 training events) the best NN configuration turns to be the net with 2 hidden nodes, because the training error is decreasing with increase of the number of hidden nodes, but the FPE is increasing dramatically. The remaining graphs on this figure show that when the sample size varies from 600 to 3000 the best configuration turns to be the net with 4 hidden nodes.

On figure?? the same dependence as on Figure?? is plotted for large sample sizes. It is easy to see that the best architecture is again 2:4:1. It is also easy to recognize that the FPE and training error values became closer in contrast with the Figure ??, which means that the NN trained with the large number of training patterns achieves better generalization. Although the best FPE is obtained using the net with 4 hidden units, from the right graph of this figure one can see that in case of very large training sample (15000 events) all networks (except the smallest) give very near FPE and within the error bounds all networks have the same performance.

References:

1. Hecht-Nielsen, R., *Neurocomputing*, Addison-Westly Publishing Company, Reding MA (1990).
2. Geisser, S., *The Predictive Sample Reuse Method with Applications*, JASA 70 (1975), 350.
3. Barron, A., *Predicted Squared Error: a Criterion for Automatic Model Selection, in Self Organizing Methods in Modeling*, ed. S. Farlow, Marsel Dekker, New York, (1984).
4. Wahba, G., *Spline Models for Observational Data*, Regional Conf. Series in Appl. Math. SIAM press, 59 (1990).
5. Akaiki, H., *Statistical Predictor Identification*, Ann. Inst. Statist. Math. 22(1970), 203.