Efficient Cross-Validation for Feedforward Neural Networks

Tin-Yau Kwok (jamesk@cs.ust.hk) and Dit-Yan Yeung (dyyeung@cs.ust.hk)
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

ABSTRACT

In this paper, we study the use of cross-validation for estimating the prediction risk of feedforward neural networks. In particular, the problem of variability due to the choice of random initial weights for learning is addressed. We demonstrate that nonlinear cross-validation may not be able to prevent the network from falling into the "wrong" perturbed local minimum. A modified approach that reduces the problem to a linear problem is proposed. It is more efficient and does not suffer from the local minimum problem. Simulation results for two regression problems are discussed.

1. Prediction of Generalization Performance

In recent years, multi-layer feedforward neural networks have been popularly used in many pattern classification, function approximation and regression problems. After the network has been designed and trained, one would naturally like to have an estimate of the network's performance on future observations. This is useful in ensuring that the network is good enough before submitting it to the customer, or in situations when we want to pick the "best" network among a number of alternatives.

Consider a set D of N observations $\{(x_j, y_j)\}$, with input x_j and output y_j, generated by $y_j = f(x_j) + \epsilon_j, j = 1, \ldots, N$, where f is the unknown underlying function, and ϵ_j is the independent and identically distributed (iid) error with mean zero and variance σ^2. A neural network constructs an estimator $\hat{f}$ of the underlying function f from the training set D. Generalization performance can be measured by the *prediction risk* [4, 7], which is the expected error on the entire input domain:

$$P(\hat{f}; D) = \int [f(x) - \hat{f}(x)]^2 p(x)\, dx + \sigma^2,$$

where $p(x)$ is the unknown probability density function of the input x. Obviously, $P(\hat{f}; D)$ can be calculated only when exact knowledge of f, p and σ is available. In reality, one can at best obtain an estimate of $P(\hat{f}; D)$. A simple estimate is the mean squared error on the training data, $E_{tr}(\hat{f}; D) = \frac{1}{N}\sum_{j=1}^{N}[y_j - \hat{f}(x_j)]^2$. However, it is well known that this estimate is biased.

In the following, we study the use of cross-validation for estimating the prediction risk of feedforward neural networks. An overview of cross-validation is in Section 2. Problems particular to nonlinear models, of which neural networks form a subclass, are discussed in Section 3. Section 4 discusses a modified approach to performing cross-validation. Simulation results are presented in Section 5, followed by some concluding remarks in the last section.

2. Cross-Validation

2.1. Data Splitting

Cross-validation (CV) is an old idea. For reviews, see [2, 9]. The simplest form of cross-validation is *data splitting*, which is commonly used in the neural network community [10]. This consists of dividing the available training data into two sets. The first set is used to train the network, while the second one is for evaluating the performance of the trained network. This procedure is simple, but with three main limitations. Firstly, when too small a portion of the available data are used for training, the network's prediction risk increases and thus its performance degrades. Secondly, the ability to measure the prediction risk suffers when too small a portion of the data are used for testing. Hence, when the data available are scarce, such splitting may not be possible. Thirdly, the estimated prediction risk obtained by this procedure is sensitive to the specific way of splitting [11].

2.2. Leave-One-Out Cross-Validation

Under this scheme [2], all observations except the jth data point (x_j, y_j) are used in the training. $\hat{f}_{(-j)}(x)$ is chosen such that it minimizes

$\frac{1}{N-1}\sum_{i\neq j}[y_i - \hat{f}_{(-j)}(x_i)]^2$. The jth data point, being a singleton set, is used for validation. This process is repeated for all j. The estimated prediction risk is:

$$P_{CV}(\hat{f}; D) = \frac{1}{N}\sum_{j=1}^{N}[y_j - \hat{f}_{(-j)}(x_j)]^2.$$

The main objection to this approach is that training of the network has to be repeated N times. As each such training process involves a nonlinear optimization problem, it is very computationally expensive for use in neural networks.

2.3. v-Fold Cross-Validation

This approach [1] is aimed at reducing the computation involved in the simple leave-one-out method. Assume that $N = vd$ for some integers v and d. Instead of deleting one observation at a time, more than one observation are deleted. The available data are divided randomly into v groups $G_1, G_2, \ldots, G_v$, each of size d. Without loss of generality, suppose that the division is as follows:

$$\overbrace{1,\ldots,d}^{G_1},\ \overbrace{d+1,\ldots,2d}^{G_2},\ \ldots,\ \overbrace{(v-1)d+1,\ldots,vd}^{G_v}.$$

$\hat{f}_{(-j)}(x)$ is chosen by leaving out group G_j and minimizing the mean squared error for the remaining $N - d$ training examples. This is repeated for all the groups. The estimated prediction risk is

$$P_{CV}(\hat{f}; D) = \frac{1}{N}\sum_{j=1}^{v}\sum_{(x_k,y_k)\in G_j}[y_k - \hat{f}_{(-j)}(x_k)]^2. \quad (1)$$

Obviously, when $v = N$, it reduces to the leave-one-out form. When v is small, Burman [1] suggested the addition of two correction terms:

$$\begin{aligned} P^*_{CV}(\hat{f}; D) &= P_{CV}(\hat{f}; D) + E_{tr}(\hat{f}; D) \\ &\quad -\frac{1}{v}\sum_{j=1}^{v}\sum_{k=1}^{N}[y_k - \hat{f}_{(-j)}(x_k)]^2. \quad (2) \end{aligned}$$

There are other approaches closely related to v-fold cross-validation [1, 12]. But they are more computationally expensive and hence will not be addressed in this paper.

3. Problems Encountered

3.1. Excess Error and Expected Excess Error

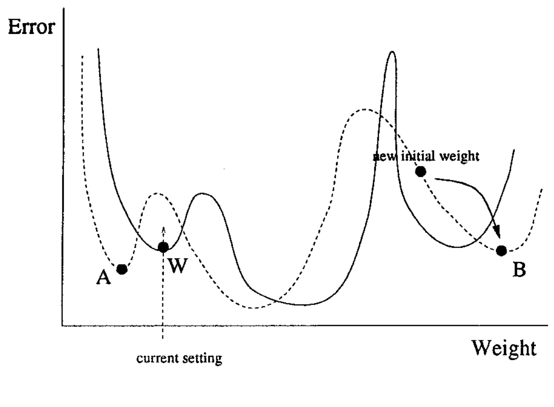
Define *excess error* $R = P(\hat{f}; D) - E_{tr}(\hat{f}; D)$, and *expected excess error* $\langle R\rangle$ as the expectation of R over all possible realizations of the training set D [2]. After training on set D, the practitioner obtains a set of network weights W, and is interested in the generalization performance (i.e. prediction risk $P(\hat{f}; D)$) of the network with that particular weight configuration. This may be obtained once R is known. Cross-validation, however, actually estimates $\langle R\rangle$ rather than R. [2] showed that the CV estimate of prediction risk is an unbiased estimate of $\langle R\rangle + E_{tr}(\hat{f}; D)$ for linear models. [8] also proved similar unbiasedness results for neural networks. In other words, the CV estimate is averaged over the many training sets that the practitioner might have observed and, therefore, over many realizations of the network model, rather than for that particular network he has obtained. Hence, the CV estimate is only an approximation to R. Nevertheless, this is almost inevitable [5] and the approximation is usually good [3].

3.2. Nonlinear Cross-Validation

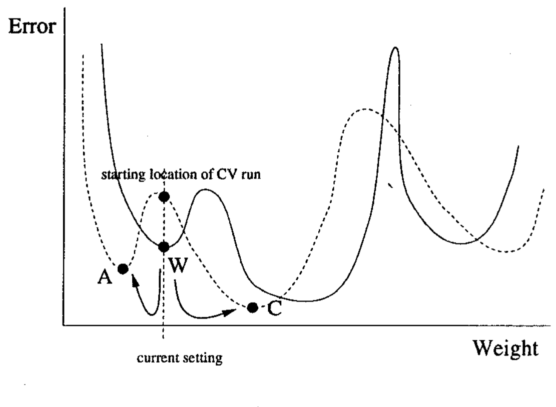
However, CV becomes more problematic for neural networks [7]. For linear models, the optimal model parameters can be uniquely determined from the training set D. Similarly, in [8], the network weights are assumed to be at the global optimum for D. But this is not true in practice. The training error surface typically contains multiple local minima, and the globally optimal weights are unlikely to be obtained. If the network is started with random initial weights for each subset of the data during cross-validation, the network will typically end up in different locally optimal weight configurations (Figure 1). Hence, the estimate suffers from an additional variability due to the choice of the random initial weights.

Nonlinear cross-validation (NCV) [7] is suggested to alleviate this problem. The network is first trained using all available data. Network weights are then saved and used as initial values in all subsequent CV runs. In so doing, the algorithm attempts to avoid going to different local optimal weight configurations in different runs. It is also expected to reduce the computational load that would be incurred by retraining from random initial weights.

However, the effectiveness of NCV in alleviating the problem is unclear (Figure 1). Determining whether the weights obtained in each CV run correspond to the "correct" perturbed weight configuration is by itself difficult, as the error surface for a particular subset of data may be very different from that for all data. Moreover, the weights obtained depend on both the shape of the perturbed error surface and the learning algorithm. Using the saved configuration as initial weights does not guarantee finding the "correct" location. On the computational aspect, it is also doubtful whether anything can be saved, as the whole NCV procedure still involves the same number of nonlinear optimization



(a) starting from random weights



(b) NCV

Fig. 1: For illustration purpose, assume that the weight vector of the network is one-dimensional. The solid line is the (training) error surface corresponding to all available data, and the weight setting obtained during training is W, while the dotted line is that for a particular subset of data. As shown, both the error surface and W are perturbed. A is the "correct" perturbed weight corresponding to W. If the network ends up at B or C, then this corresponds to a different local optimum.

problems.

4. Modified NCV

The idea is simple. After training the network with all available data, as in NCV, all the weights except those connecting the final hidden layer to the output layer are held fixed. Cross-validation then proceeds as usual. The rationale is as follows. Hidden units in the final hidden layer produce transformed variables of the input by summarizing all transformations by previous hidden layers. For linear output units, the network is a linear model of these transformed variables (and similarly a logistic model for sigmoidal outputs). The neural network obtained by the practitioner is a particular linear (or logistic) model with a particular set of regression coefficients (which correspond to the hidden-to-output weights in the network). This expected prediction risk can then be approximated quite satisfactorily using the usual cross-validation technique. Moreover, this method has the following advantages over NCV:

1. In the case of linear output units and quadratic error criterion, optimization for each CV run is reduced to a linear problem, with a unique globally optimal solution. The problem of falling into different local minima no longer exists. Computationally, only pseudo-inverses are required, or may be solved rapidly by descent algorithms (especially those with second-order information). Leave-one-out cross-validation also becomes feasible, and in this case the estimate for each CV run can be obtained by simple algebraic equation.
2. Even when the output units are not linear or the quadratic error criterion is not used, optimization is still more efficient because of the reduction in the number of parameters to be optimized.
3. Results in [8] assume that the network weights obtained are always globally optimal, which is not practical. With modified NCV, the problem reduces to one for linear models when the output units are linear. Classical results on unbiasedness can be directly applied.

5. Simulation

All networks under investigation have one hidden layer. Accuracy of the estimated prediction risk is measured by:

$$\text{relative deviation} = \frac{P_{CV}(\hat{f};D) - P(\hat{f};D)}{P(\hat{f};D)}.$$

For simplicity, the "true" prediction risk $P(\hat{f};D)$ is approximated by the mean squared error on a large, independent test set. Moreover, experiments indicated that the correction terms in (2) are not significant, hence we compute P_{CV} as in (1).

5.1. Univariate Function

For ease of visualization, we first study the following simple regression problem:

$$y_j = \sin(2x_j)+\epsilon_j, \quad x_j \in U[-\pi,\pi], \epsilon_j \sim N(0, 0.1^2).$$

The training and test sets have sizes of 100 and 6282 respectively. 5-fold cross-validation is used. 20 independent trials, each using random initial weights, are performed.

We first illustrate a situation when the network falls into a different local minimum under NCV. Figure 2 compares the family of functions obtained from the 5 CV runs with that learned using all available data, in a particular trial using both 5-fold NCV and 5-fold modified NCV. Each curve in the figure corresponds to a function learned in a CV run. There is noticeable difference between the function learned using all available data and the family of functions learned in the CV runs for NCV. The resulting relative deviations for NCV and modified NCV are -0.684 and -0.066 respectively.

Figure 3 compares the relative deviations for the various estimators over 20 trials. Each jointed line corresponds to a weight configuration for the trained network in each trial. There is a clustering of the jointed lines, causing by the fact that the weight configurations have clustered around the local minima in the error surface. As expected, the MSE on the training set generally gives poor estimate of the prediction risk. Estimation by NCV can sometimes be quite good, but is frequently plagued by having very bad estimates, because of the problem of falling into the wrong local minimum mentioned earlier.

We have also used different noise levels, different numbers of folds and different sizes of the training set, and similar results are obtained.

5.2. Complicated Interaction Function

$$\begin{aligned} y_j &= 1.9(1.35 + e^{x_{j1}} \sin(13(x_{j1} - 0.6)^2) \\ &\quad e^{-x_{j2}} \sin(7x_{j2})) + \epsilon_j, \\ &\quad x_j \in U[0,1]^2, \epsilon_j \sim N(0, 0.25^2). \end{aligned}$$

This function is used in [6]. A total of 225 pairs of training data points are generated, and a test set of size 10000 is generated on a regularly spaced grid on $[0,1]^2$. Figure 4 shows a plot of the function.

10 independent trials of 15-fold cross-validation are performed. Figure 5 compares the boxplots of relative deviations for the two approaches. Again, as is usually the case, when the number of hidden units increases, the training error surface becomes increasingly complex and the chance of falling into "wrong" perturbed local minima also increases, leading to a broader range and more outliers in the boxplot.

6. Conclusion

In this paper, we study the use of cross-validation for estimating the prediction risk of feedforward neural networks. We demonstrate that NCV may not be able to prevent the network from falling into "wrong" perturbed local minima, while modified NCV does not suffer from this problem and is also more efficient. Future directions include applying this to classification problems and also for bootstrapping.

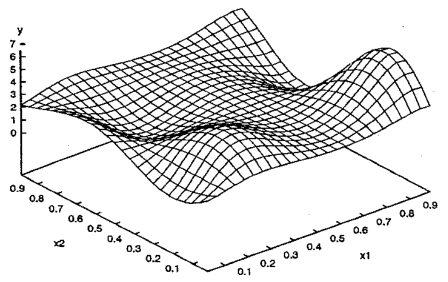


Fig. 4: A plot of the complicated interaction function.

Acknowledgments

This research has been partially supported by the Hong Kong Research Grants Council under grants RGC/HKUST 15/91 and RGC/HKUST 614/94E and the Hong Kong Telecom Institute of Information Technology under grant HKTIIT 92/93.002 awarded to the second author. The first author is supported by the Sir Edward Youde Memorial Fellowship.

References

[1] P. Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76:503–514, 1989.

[2] B. Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1982.

[3] B. Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81(394):461–470, June 1986.

[4] R.L. Eubank. *Spline Smoothing and Nonparametric Regression.* Marcel Dekker, Inc., 1988.

[5] G. Gong. Cross-validation, the jackknife, and the bootstrap: Excess error estimation in forward logistic regression. *Journal of the American Statistical Association*, 81(393):108–113, May 1986.

[6] J.N. Hwang, S.R. Lay, M. Maechler, D. Martin, and J. Schimert. Regression modeling in back-propagation and projection pursuit learning. *IEEE Transactions on Neural Networks*, 5(3):342–353, May 1994.

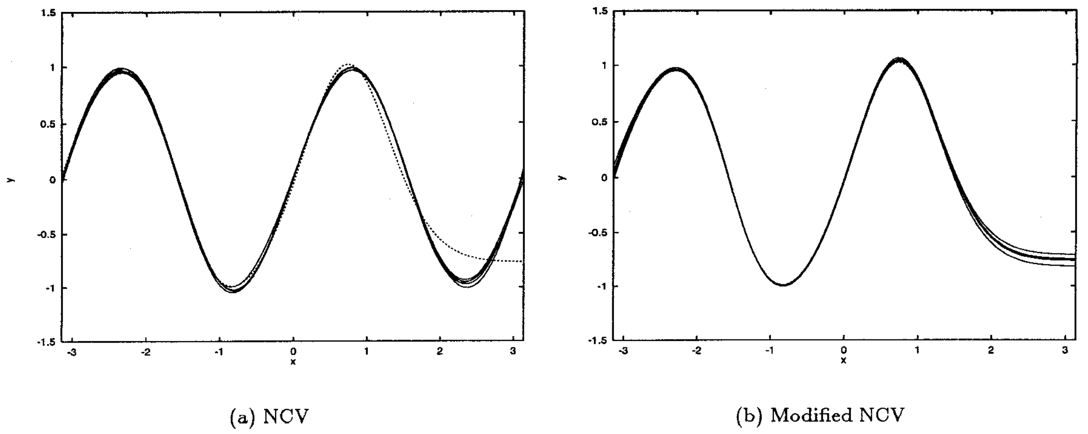


Fig. 2: The problem of falling into another local minima by NCV. In figure (a), the dotted line is the function learned using all available data and the solid lines is the family of functions learned in 5 CV runs. The difference is obvious towards the right-hand side of the plot. In figure (b), there is no noticeable difference between that learned by all data and each CV run.

[7] J. Moody. Prediction risk and architecture selection for neural networks. In V. Cherkassky, J.H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, volume 136 of *NATO ASI Series F*, pages 147–165. Springer-Verlag, 1994.

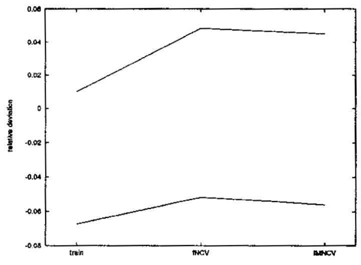
[8] M. Plutowski, S. Sakata, and H. White. Cross-validation estimates IMSE. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 391–398. Morgan Kaufmann, 1994.

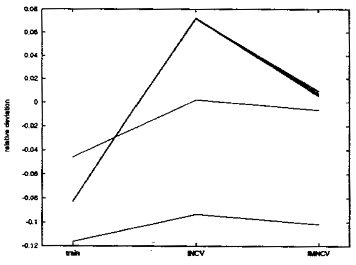
[9] M. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society Series B*, 36:111–147, 1974.

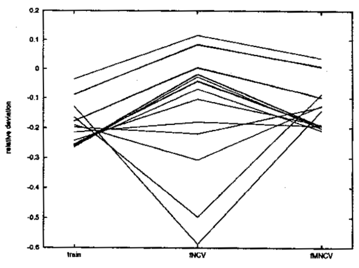
[10] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1(3):193–209, 1990.

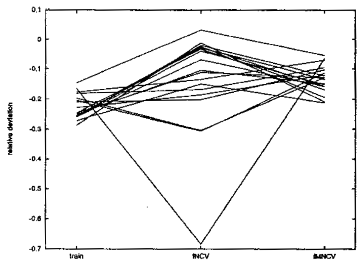
[11] A.S. Weigend and B. LeBaron. Evaluating neural network predictors by bootstrapping. In *Proceedings of International Conference on Neural Information Processing*, volume 2, pages 1207–1212, Seoul, Korea, October 1994.

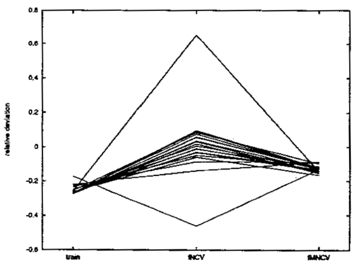
[12] P. Zhang. Model selection via multifold cross validation. *The Annals of Statistics*, 21(1):299–313, 1993.


(a) #hid=1


(b) #hid=2


(c) #hid=3


(d) #hid=4


(e) #hid=5

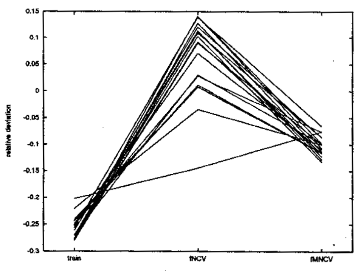
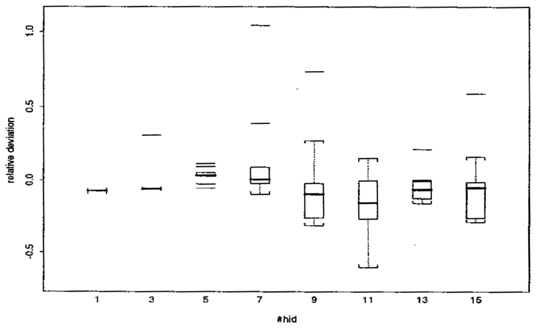
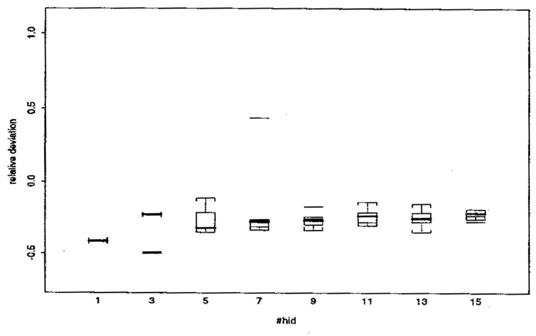

(f) #hid=6

Fig. 3: Comparison of the relative deviations for the various estimates in the univariate function problem. Here train denotes the training MSE, fNCV is 5-fold NCV, fMNCV is the 5-fold modified NCV.



(a) NCV



(b) Modified NCV

Fig. 5: Comparison of relative deviations for the complicated interaction function. The horizontal line in the interior of the box is located at the median. The height of the box is equal to the interquartile distance, or IQD, which is the difference between the third quartile of the data and the first quartile. The *whiskers* (the dotted lines extending from the top and bottom of the box) extend to the extreme values or a distance $1.5 \times IQD$ from the center, whichever is less. Data points which fall outside the whiskers are indicated by horizontal lines.