# Cross Validation and MLP Architecture Selection

Tim Andersen and Tony Martinez
tim@axon.cs.byu.edu, martinez@cs.byu.edu
Computer Science Department, Brigham Young University

## Abstract

*The performance of cross validation (CV) based MLP architecture selection is examined using 14 real world problem domains. When testing many different network architectures the results show that CV is only slightly more likey than random to select the optimal network architecture, and that the strategy of using the simplest available network architecture performs better than CV in this case. Experimental evidence suggests several reasons for the poor performance of CV. In addition, three general strategies which lead to significant increase in the performance of CV are proposed. While this paper focuses on using CV to select the optimal MLP architecture, the strategies are also applicable when CV is used to select between several different learning models, whether the models are neural networks, decision trees, or other types of learning algorithms. When using these strategies the average generalization performance of the network architecture which CV selects is significantly better than the performance of several other well known machine learning algorithms on the data sets tested.*

## 1. Introduction

This paper examines the performance of cross validation (CV) as an MLP (multi-layer perceptron) architecture selection strategy. A primary advantage of CV is that only the data is used to determine which architecture is appropriate, without the requirement for user intervention or the setting of any adjustable parameters. Unfortunately, for a variety of reasons CV does not always perform as well as desired. The purpose of this paper is to determine empirically whether or not the expectation that CV based architecture selection will generally perform well on real world problems is justified. We also explore empirically and discuss general strategies for increasing the likelihood that CV will select a good architecture.

One of the major difficulties with MLPs lies in the selection of the optimal network architecture for a given problem. MLP architecture selection is concerned with the number of layers in the network, the number of nodes in each layer, the interconnections between the nodes, and so forth. For any given learning problem there is an essentially infinite number of possible MLP network architectures, but only a small subset of these exhibit good performance in general. A great deal of effort has been devoted towards MLP architecture selection, and several different methods which seek to automate (more or less) MLP architecture selection are now available. These methods include network construction, network pruning, information based criteria such as MDL and MML, and cross validation. In addition to architecture selection strategies, there are regularization methods such as weight decay, stopped training techniques, and bayesian techniques which all seek to obviate the need to select an optimal network architecture, instead using the most complex architecture which can be practically implemented and then using some other strategy to avoid overfitting. However, no one of these methods has yet proven to perform well on a large variety of problem domains.

We define the "optimum" network architecture to be the simplest network architecture which is capable of representing the underlying function which generated the training data. However, architecture selection strategies are rarely if ever concerned with identifying the "optimum" network architecture. A more pressing concern is the probability that a given MLP architecture will perform well after training. We define the network architecture which is the most likely to perform well after training on the available training data as the "optimal" network architecture. The determination of the optimal network architecture is thus highly dependent upon the available training data and the idiosyncracies of the training algorithm. Finding the optimal network architecture is the goal of most (if not all) architecture selection strategies.

This paper provides insight into the empirical performance of CV on a variety of real world problem domains. To date, there have been few studies which have focused on the empirical performance of CV based MLP architecture selection on a large number of real world problems. One reason for this may be the enormous amount of computation required for such a study. This study, which applies CV to 14 different real world problems, utilized 74 unix workstations running continuously over a period of approximately two and a half months. The studies in the literature which specifically examine the performance of CV and compare it with that of other methods [8][10][3] analyze performance using only a few (1 or 2) data sets, and so cannot be considered conclusive. A realistic evaluation of the performance of CV based MLP architecture selection on real world problems, including strength and weaknesses, needs to be established. This paper also examines the conditions which can affect the performance of CV, such as the number of architectures tested, the similarities between the architectures, the degree of difference in CV holdout scores, the amount of available training data, etc. It is important to be aware of these items and how they can affect the performance of CV in order to design a system which has a high probability of finding an optimal architecture.

The results in this paper, which are presented in detail in section 4, show that, at least on the real world data sets tested in this paper, CV is on average only slightly better than random architecture selection when choosing from among a large number of potential architectures. The main benefit of CV in this case is to decrease the likelihood of choosing an extremely sub-optimal architecture. Any potential increase in generalization accuracy obtainable through CV based architecture selection drops off rapidly as the number of tested architectures increases. This is particularly true when the architectures being compared are similar in their structure. This means that using CV to compare several similar network architectures, is not only wasteful of computational resources but can also degrade the performance of CV. However, if a reasonable difference between network architectures is maintained, then more architectures can be compared before the performance of CV begins to degrade. Also, the probability that CV will choose the optimal architecture is lower when the difference between CV scores is small, and significant improvement to generalization accuracy can be made by only accepting a particular network architecture if all other simpler architectures have significantly worse CV scores.

Section 2 discusses the problem of model selection, CV, and real world problems. Section 3 gives the data sets and methods used in this paper, and section 4 details the results. The conclusion is given in section 5.

## 2. Model Selection and Real World Problems

One of the primary goals of machine learning is to produce a general, automated learning algorithm which performs well for all types of learning problems. This has been proven to be an unattainable goal [7][9]. However, it is possible to develop a learning algorithm that will perform provably well for a particular problem or type of problems. For the most part we are not interested in all types of learning problems but are primarily interested in the "real world" learning problems. To the extent that all real world learning problems are similar, it should be possible to develop a general learning algorithm which performs well on them.

CV is an oft used method for comparing two or more learning models to estimate which model will perform the best on the problem at hand. With $n$-fold CV, the available training data is partitioned into $n$ disjoint subsets, the union of which is equal to the original training set. Each learning model is trained on $n$-1 of the available subsets, and then tested on the one subset which was not used during training. This process is repeated $n$ times, each time using a different test set chosen from the $n$ available partitions of the training data, until all possible choices for the test set have been exhausted. The $n$ test set scores for each learning model are then averaged (or summed), and the model with the highest average test set score is chosen as the most likely to perform well on unseen data. The standard practice for MLP model selection is to use 10-fold CV, and this is the type of CV which is tested in this paper.

The advantage of CV over other model selection strategies is that in its basic form it is entirely data driven. But in practice CV suffers from two major drawbacks. The first drawback is that when it is used to select between two or more models the estimate on model accuracy which CV provides tends to be higher than the true model accuracy, and this tendency becomes more pronounced as the number of models tested increases. The second and related problem is that, in general, the more models that are tested the higher the probability that CV will fail to select the best available model.

Research that has been done on CV based MLP architecture selection includes a recent paper by Schenker and Agarwal [10] where CV was found to be the better than a few other architecture selection strategies at choosing the optimal network architecture. However, the comparison was based on only a single type of artificial data and did not look at any real world problem domains, and so these can not be considered conclusive. Another paper by Kearns et. al. found that CV performs significantly better than Minimum Description Length (MDL) and Guaranteed Risk Minimization (GRM) [11] on the intervals model selection problem [3]. Unfortunately, the empirical results in this paper were also limited to a single type of artificial data, and did not explore any real world problem domains. Schaffer has also studied CV in [7] and [8].

CV is also employed in stopped training, weight decay, network construction algorithms, and network pruning methods.

## 3. Data and Methods

The main intent of this paper is to examine the performance of CV based MLP architecture selection on real world problems, and so 14 real world problems were selected from the UCI machine learning database repository as a basis for the experiments. The choice of which data sets to use was restricted to the binary classification (two output) problems for the sake of simplicity. The names and a short description of the 14 data sets are given in table 1.

The first column gives the name (or tag) used to identify the data set throughout the rest of this paper. The total number of attributes is listed in the third column, and the fourth column gives the total number of examples contained in the data set.

| tag | full name | attributes | instances |
|---|---|---|---|
| bc | breast cancer | 9 | 286 |
| bcw | breast cancer wisconsin | 10 | 699 |
| bupa | bupa liver disorders | 7 | 345 |
| credit | credit approval | 15 | 690 |
| echo | echocardiogram | 13 | 132 |
| sickeu | sick-euthyroid | 26 | 3163 |
| hypoth | hypothyroid | 26 | 3123 |
| ion | ionosphere | 35 | 351 |
| promot | promoter gene sequences | 57 | 106 |
| sick | sick | 30 | 3772 |
| sonar | sonar | 61 | 208 |
| stger | german credit numeric | 24 | 1000 |
| sthear | statlog heart | 13 | 270 |
| voting | house votes 1984 | 16 | 435 |

Table 1. Data sets.

### 3.1 Experiments

The MBP neural network simulator [1], which implements a fast conjugate gradient descent training algorithm, was used to train the various network architectures due to its speed of training and relative ease of use. Since there is a limited amount of available data for the real world data sets, the accuracy of the model which CV chooses must be estimated using CV. This implies that within each CV split used to estimate the accuracy of the chosen model, a secondary CV split must be performed in order to facilitate the choice of the MLP architecture. A formal explanation of this process follows.

Each real world data set is first divided into 10 disjoint test (validation) sets of equal size (or as equal in size as possible). Let $D$ be the entire set of available labeled data. We define $V_i$ (the ith test set) to be the $i$th subset of $D$ such that the following hold:

$$(\forall i)(1 \le i \le 10 \rightarrow V_i \subset D) \quad (1)$$

$$D = \bigcup_{i=1}^{10} V_i \quad (2)$$

$$(\forall i,k)(1 \le i,k \le 10 \wedge i \ne k \rightarrow V_i \cap V_k = \varnothing \wedge \left\| V_i \right| - \left| V_k \right\| \le 1) \quad (3)$$

Simply stated, equations 1 through 3 partition $D$ into 10 non-overlapping subsets any two of which differ in size by at most one element, and the union of which equals $D$. For each test set $V_i$ we define an associated training set $T_i$ as follows:

$$\text{let } T_i = D - V_i \quad (4)$$

Each $T_i$ is further subdivided into 10 disjoint holdout sets $H_{ij}$ in precisely the same way as was done with the data set $D$.

$$(\forall i,j)(1 \le i,j \le 10 \rightarrow H_{ij} \subset T_i) \quad (5)$$

$$T_i = \bigcup_{j=1}^{10} H_{ij} \quad (6)$$

$$(\forall i,j,k)(1 \le i,j,k \le 10 \wedge j \ne k \rightarrow H_{ij} \cap H_{ik} = \varnothing \wedge \left\| H_{ij} \right| - \left| H_{ik} \right\| \le 1) \quad (7)$$

For each holdout set $H_{ij}$ we define an associated sub training set $T_{ij}$ as follows:

$$\text{let } T_{ij} = T_i - H_{ij} \quad (8)$$

Let $\lambda$ be a function which takes as inputs a network architecture $\varphi$ and a set of labeled training examples $T$ and returns a fully trained network. The general format for this function is then

$$\lambda(\varphi, T) \quad (9)$$

Where $\lambda$ is the training algorithm, $\varphi$ is the network architecture, and $T$ is the training set. For the network architectures tested in this paper it is sufficient to differentiate between them by expressing $\varphi$ as an integer which is equal to the number of hidden nodes in the network, since the network architecture is restricted to be fully connected with a single hidden layer. Let $\rho$ be a function which takes as arguments a fully trained network and a labeled data set and returns the performance of the network on that data set. There are several different error functions which can be used to measure the performance of a network. For this paper we use the percentage of correct predictions. The CV based procedure for choosing a network architecture is then for each $T_i$ choose $\varphi$ which

$$\text{maximizes } \sum_{j=1}^{10} \rho\left(m\left(\varphi, T_{ij}\right), H_{ij}\right) \quad (10)$$

For a given $T_i$ we define the network architecture chosen by CV to be $\varphi_i$. The actual performance of $\varphi_i$ is then estimated using the test set $V_i$. There are several ways which this can be done. One way is to retrain $\varphi_i$ using the entire training set $T_i$, in other words use $\rho(m((\varphi_\iota, T_{\iota\varphi}), V_{ij})$ as the estimate for the actual performance of $\varphi_i$. Another way is to combine the 10 separate networks obtained from training $\varphi_i$ on the 10 different sub training sets $T_{ij}$ with some type of voting scheme. The method which was used to estimate the performance of a particular architecture $\varphi$ is to average the test set performance of the 10 networks trained on the 10 sub training sets, as shown in equation 11.

$$\sum_{j=1}^{10} \rho\left(m\left(\varphi_i, T_{ij}\right), V_{ij}\right) \quad (11)$$

## 4. Results

### 4.1 Cross Validation and Real World Problems

Table 2 reports the average generalization accuracy of CV based architecture selection on the 14 real world data sets introduced in section 4.1. Each data set was tested on network architectures with a single hidden layer containing from 2 to 20 hidden nodes. Equation 10 was used to select the winning network architecture. The first column of table 2 lists the names for the data sets tested, and the third column (labeled CV) gives the average accuracy of the CV selected architecture on the test set for each of the data sets. The table also reports the best and worst possible scores, where the 'best' column is the average test set accuracy obtained by choosing $\phi$ which maximizes 11, and the 'worst' column reports the average test set accuracy obtained by choosing $\phi$ which minimizes 11. The best column is an upper bound on the performance which can be achieved with the architectures and training techniques used in this paper, and the worst column gives a lower bound. The 'avg' column reports the average score of all architectures tested for each data set, which is essentially the score that would be expected if an architecture was chosen at random for each training set. The last row of the table reports the average of each column.

| data set | n=2 | CV | best | worst | avg |
|---|---|---|---|---|---|
| bc | 69.14 | 66.30 | 70.90 | 59.54 | 64.65 |
| bcw | 95.38 | 94.92 | 96.05 | 93.41 | 94.61 |
| bupa | 71.37 | 72.64 | 74.34 | 70.15 | 72.12 |
| credit | 84.45 | 84.13 | 85.06 | 80.17 | 82.13 |
| echo | 86.60 | 86.51 | 89.42 | 84.14 | 86.71 |
| hypoth | 98.19 | 98.17 | 98.46 | 97.90 | 98.21 |
| ion | 87.36 | 88.76 | 89.87 | 86.40 | 88.25 |
| promot | 90.60 | 90.32 | 93.08 | 87.39 | 90.56 |
| sick | 97.49 | 97.53 | 97.66 | 97.26 | 97.49 |
| sickeu | 96.64 | 96.80 | 96.93 | 96.46 | 96.75 |
| sonar | 78.57 | 79.18 | 80.49 | 76.79 | 78.52 |
| stger | 74.06 | 72.88 | 74.51 | 68.45 | 70.82 |
| sthear | 78.93 | 77.07 | 80.52 | 73.93 | 76.85 |
| voting | 94.24 | 94.74 | 95.22 | 93.84 | 94.58 |
| AVG | 85.93 | 85.71 | 87.32 | 83.27 | 85.16 |

Table 2. Test results for CV.

The average of all architectures across all data sets is 85.16%, which is only slightly lower than the average score of the CV chosen architectures. This means that CV is on average only slightly better than random at choosing between the available network architectures, and is 1.61 percentage points below the upper bound on performance. However, CV does appear to provide some insurance against the possibility of particularly poor performance by almost always scoring at or slightly above the average architecture score for each data set. When CV did score below the average architecture score, as it did with echo, hypoth and promot, it was at most 2 tenths of a percentage point lower than the average, but when it scored above the average it was as much as 2 percentage points higher. Interestingly, CV does not on average perform any better than the simplest (2 hidden nodes) network architecture tested. The second column of table 2 reports the average test set results of the 2 hidden node network on each of the data sets. The 2 hidden node network outperforms CV by 0.22 percentage points on average at the 0.9 confidence level.

### 4.2 Improving CV

This poor showing by CV is surprising, but there are areas where improvement can be made. The standard approach of choosing the architecture which maximizes the CV score may be overly optimistic in its trust of the scores which CV produces. A very slight difference in holdout scores is probably not much better than zero difference in determining the best architecture. Rather than selecting the network which maximizes the holdout set score as with equation 10, it may be better to accept a network of size $n$ only if it significantly outperforms all other smaller networks. We consider a score to be significantly better if, using the Student T-test, it can be said to be better at the 0.9 confidence level. This approach does offer significant improvement over standard CV, with an average generalization accuracy of 86.03% (versus 85.71% for CV) on the data sets tested.

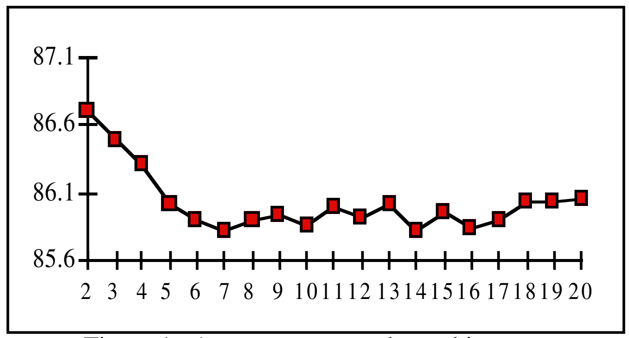


Figure 1. Average accuracy by architecture.

Figure 1 gives the average generalization (test set) accuracy over all of the data sets tested for each network architecture. As the complexity of the architectures increases the average generalization accuracy decreases rapidly until it levels off at the 7 hidden node architecture. It is interesting to look at the performance of CV (given in table 8) when it is limited to choosing between only those architectures which have either 2 or 20 hidden nodes (the maximum difference possible for the architectures tested), hereafter referred to as CV(2,20). Intuitively, the results given in figure 1 would seem to imply that the poor average generalization performance of the 20 hidden node network will cause CV(2,20) to perform worse than the simple 2 hidden node network. However, there is a higher probability that CV will choose the best architecture for CV(2,20) than for any other possible comparison due to the fact that CV is better at distinguishing between highly disimilar architectures than it is at distinguishing between similar architectures. In fact, CV(2,20) does have a higher average generalization accuracy than the 2 hidden node network as shown in table 3. The improvement is significant at the .95 confidence level.

| | |
|---|---|
| CV(2,20) | 86.07 |
| CV(2,3,4,5) | 85.81 |
| CV(2,3,4,5,20) | 85.80 |
| CV(2,3) | 85.97 |
| CV(2,6,10,15,20) | 86.01 |

Table 3. Average accuracy

The performance of CV quickly drops off when more than 2 or 3 similar networks are tested. But when testing networks that differ somewhat in their structure, more networks can be tested before it degrades the performance. For example, it would appear that restricting CV to the simplest 4 network architectures should produce good results, since the vast majority of significantly high test set scores occur with the 4 simplest architectures. The second row of table 2 (CV(2,3,4,5)) gives the results for restricting CV to the 4 simplest architectures tested. The confidence that this result is worse than CV(2,20) is .975. Adding the 20 hidden node network to the mix, CV(2,3,4,5,20), does not improve the average score of CV(2,3,4,5). Once too many similar networks have been included in the CV comparison the addition of more network architectures does not generally improve performance. Dropping the 4 and 5 hidden node networks (row 4 of table 3) leads to significant improvement. CV(2,20) still has a higher generalization accuracy on these data sets than CV(2,3), but the confidence that CV(2,20) is better than CV(2,3) is only 0.8. The results for CV(2,6,10,16,20) show that if a reasonable difference between network architectures is maintained, more architectures can be tested before performance degrades.

### 4.3 CV vs Other Learning Algorithms

Table 4 compares the average generalization accuracy of CV(2,20) on the 14 data sets tested in this paper against several other well-known learning algorithms. The comparison shows that CV and MLPs are capable of performing better than many of the learning algorithms which are frequently employed in the fields of machine learning and neural networks. The other learning methods compared against are c4 [4][12], c4.5 [2], ib1[3][6], mml [4][12], and cn2 [5][10]. The results for these algorithms are taken from [13]. The average generalization accuracy for CV is better than any of the other learning algorithms compared against (> .95 confidence level).

| c4 | c45 | ib1 | mml | cn2 | CV(2,20) |
|---|---|---|---|---|---|
| 84.57 | 84.68 | 84.00 | 85.85 | 80.74 | 86.07 |

Table 4. CV vs other learning algorithms

## 5. Discussion and Conclusion

There are three general strategies that can be applied to CV based architecture selection to significantly improve its performance. Through applying these strategies, CV based MLP architecture selection outperforms several other learning algorithms which are commonly used in the machine learning and neural network communities. These strategies are:

- Only choose a more complex network architecture if all simpler network architectures perform significantly worse.
- Restrict the set of networks which CV is choosing from to only the 2 or 3 simplest possible networks.
- Restrict the set of networks so that none of the networks in the set are too similar in their structure.

Each of these strategies individually produces significant improvement in the generalization accuracy of the network architectures which CV selects. Various combinations of these strategies were tested, but for the data sets and architectures tested in this paper none of the combinations improved the performance over individual application of the strategies.

Surprisingly, there is another strategy that performs almost as well on the real world data sets as the three listed above, which is to just use the simplest architecture. The simplest network tested had a single hidden layer containing 2 hidden nodes. This architecture had an average generalization accuracy of 85.93%. Of the various combinations tested, the best result obtained with CV was 86.12% for CV(2,6,10,15). The confidence that CV(2,5,10,15) is better than the simplest network is relatively high at 0.975, but with an improvement of only 0.19% the large amount of extra computation required by CV might not be worth it for many problems.

The low correlation of the CV and test set scores, and the low probability that CV will choose the best architecture are causes for concern. Experiments on artificial data support the notion that one reason for this poor performance may be that there is simply not enough available data to reliably train and/or determine the optimal network architecture for the data sets tested in this paper. In such a case, the simplest network architectures tend to perform as well or better than the more complex network architectures.

There are several promising areas for future work. One of these is the choice of which network architectures to include in the CV comparison. For this study, the network architectures that were tested, which were fully connected with a single layer of 2 to 20 hidden nodes, are relatively similar in structure. It should be advantageous to use CV to test network architectures which exhibit even greater diversity between them, such as architectures with many more hidden nodes, or multiple hidden layers. It would also be informative to extend the study to much larger data sets. Another area which we plan to explore is the question of what to do once an architecture has been selected. It is common practice to retrain the architecture with the entire available data set, but this approach runs the risk of generating a weight setting with poor generalization performance. A better approach might be to use all of the 10 trained copies of the network architecture that CV produces in some sort of voting scheme such as Bagging.

In conclusion, using the strategies proposed in this paper, CV based MLP architecture selection performed significantly better on average than several other learning algorithms. From the analysis of the results on both the real world and the artificial data sets it appears that many of the real world data sets tested have insufficient numbers of training data, which undermines the reliability of the CV holdout set scores. On larger data sets with adequate numbers of training instances it is likely that the correlation of the CV holdout set score with the true generalization performance will be even greater, and that CV will exhibit an even greater performance improvement over other learning models.

## 6. Bibliography


[1] Anguita, D., G.Parodi, R.Zunino - An efficient implementation of BP on RISC-based workstations. Neurocomputing, in press.

[2] Bartlett, P.L. (1997), For valid generalization, the size of the weights is more important than the size of the network. *Advances in Neural Information Processing Systems 9*, pp. 134-140 Cambrideg, MA: The MIT Press.

[3] Kearns, Michael, Yishay Mansour, Åndrew Ng, and Dana Ron (1997), "An Experimental and Theoretical Comparison of Model Selection Methods," *Machine Learning*, vol 27, pp 7-50.

[4] Krogh, Anders, and John Hertz. "A Simple Weight Decay Can Improve Generalization," In Moody, J.; Hanson, S.; and Lippmann, R., eds., Advances in Neural Information Processing Systems, volume 4, 950-957. San Mateo, CA: Morgan Kauffmann Publishers.

[5] Prechelt, Lutz (1998), Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, **11**, 761-767.

[6] Rissanen, Jorma (1986), Stochastic Complexity and Modeling. *The Annals of Statistics*, vol 14, no 3, pp 1080-1100.

[7] Schaffer, C. (1993), Overfitting avoidance as bias. *Machine Learning*, **10**, 153-178.

[8] Schaffer, Cullen. (1993), Selecting a classification method by cross-validation. *Machine Learning*, **13**, 135-143.

[9] Schaffer, C. (1994), "A conservation law for generalization performance," *Proceedings of the Eleventh International Conference on Machine Learning*. 259-265, Morgan Kaufman, 1994.

[10] Schenker, B, and M Agarwal. (1996), Cross-validated structure selection for neural networks. *Computers chem. Engng*, vol 20, no 2, 175-186.

[11] Vapnik, V. (1982), *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag.

[12] Weigend, D. H., D. E. Rumelhart, and B. A. Huberman (1989), Generalization by weight-elmination with application to forecasting. *Advances in Neural Information Processing Systems*, pp 875-882, Morgan Kaufman, San Mateo, 1991.

[13] Zarndt, F (1995), A Comprehensive Case Study: *An Examination of Machine Learning and Connectionist Algorithms.* Masters Thesis Brigham Young University.