Appears in
From Statistics to Neural Networks: Theory and Pattern Recognition Applications,
V. Cherkassky, J.H. Friedman and H. Wechsler (eds.), NATO ASI Series F, Springer-Verlag 1994.

Prediction Risk and Architecture Selection for Neural Networks

John Moody

Oregon Graduate Institute, Department of Computer Science and Engineering, P.O. Box 91000, Portland, OR 97291–1000, Email: moody@cse.ogi.edu

Abstract. We describe two important sets of tools for neural network modeling: prediction risk estimation and network architecture selection. Prediction risk is defined as the expected performance of an estimator in predicting new observations. Estimated prediction risk can be used both for estimating the quality of model predictions and for model selection. Prediction risk estimation and model selection are especially important for problems with limited data. Techniques for estimating prediction risk include data resampling algorithms such as *nonlinear cross–validation (NCV)* and algebraic formulae such as the *predicted squared error (PSE)* and *generalized prediction error (GPE)*. We show that exhaustive search over the space of network architectures is computationally infeasible even for networks of modest size. This motivates the use of *heuristic* strategies that dramatically reduce the search complexity. These strategies employ directed search algorithms, such as selecting the number of nodes via *sequential network construction (SNC)* and pruning inputs and weights via *sensitivity based pruning (SBP)* and *optimal brain damage (OBD)* respectively.

Keywords. prediction risk, network architecture selection, cross–validation (CV), nonlinear cross–validation (NCV), predicted squared error (PSE), generalized prediction error (GPE), effective number of parameters, heuristic search, sequential network construction (SNC), pruning, sensitivity based pruning (SBP), optimal brain damage (OBD).

1 Introduction and Motivation

This paper describes two important sets of tools for neural network modeling: prediction risk estimation and network architecture selection. Prediction risk is defined as the expected performance of an estimator in predicting new observations. While estimating prediction risk is important in its own right for providing a means of estimating the expected error for predictions made by a network, it is also an important tool for model selection.

In this section, we motivate the need for neural network architecture selection techniques and survey various selection criteria. Section 2 introduces prediction risk,

while sections 3 and 4 present prediction risk estimation techniques, including test set validation, *nonlinear cross-validation (NCV)*, the *predicted squared error (PSE)*, and the *generalized prediction error (GPE)*. Section 5 describes architecture selection for perceptron architectures and the complexity and impossibility of exhaustive search over the space of architectures. Section 6 describes some computationally feasible heuristic search strategies, including determining the number of nodes via *sequential network construction (SNC)*, *sensitivity based pruning (SBP)* of inputs, and *optimal brain damage (OBD)* pruning of weights.

1.1 Nonparametric Modeling with Limited Data

Many data modeling problems are characterized by two difficulties: (1) the absence of a complete *a priori* model of the data generation process (such as the models frequently available in physics) and (2) by a limited quantity of data. When constructing statistical models for such applications, the issues of model selection and estimation of generalization ability or *prediction risk* are crucial and must be addressed in order to construct a near optimal model.

When a complete *a priori* model for the data generation process does not exist, one often adopts a *nonparametric modeling* approach. In nonparametric modeling, elements of a class of functions known to have good approximation properties, such as smoothing splines (for one or two dimensional problems) or neural networks (for higher dimensional problems), are used to fit the data. An element of this class (e.g., a particular neural network) is then chosen which "best fits" the data.

The notion of "best fits" can be precisely defined via an objective criterion, such as *maximum a posteriori probability (MAP)*, minimum *Bayesian information criterion (BIC)* (Akaike, 1977; Schwartz, 1978), *minimum description length (MDL)* (Rissanen, 1978), or *minimum prediction risk (P)*. In this paper, we focus on the prediction risk as our selection criterion for two reasons. First, it is straightforward to compute, and second, it provides more information than selection criteria like MAP, BIC, or MDL, since it tells us how much confidence to put in predictions produced by our best model.

The restriction of limited data makes the model selection and prediction risk estimation problems more difficult. A limited training set results in a more severe bias / variance (or underfitting vs overfitting) tradeoff (see for example Geman, Bienenstock and Doursat (1992)), so the model selection problem is both more challenging and more crucial. In particular, it is easier to overfit a small training set, so care must be taken not to select a model which is too large.

The expected training and test errors and the bias / variance tradeoff for finite training samples as a function of model size are illustrated in Figure 1.

Also, limited data sets make prediction risk estimation more difficult if there is not enough data available to hold out a sufficiently large independent test sample. In these situations, one must use alternative approaches that enable the estimation of prediction risk from the training data alone, such as data resampling and algebraic estimation techniques. Data resampling methods include nonlinear refinements of *v–fold cross–validation (NCV)* and *bootstrap estimation* (see section 3.3),

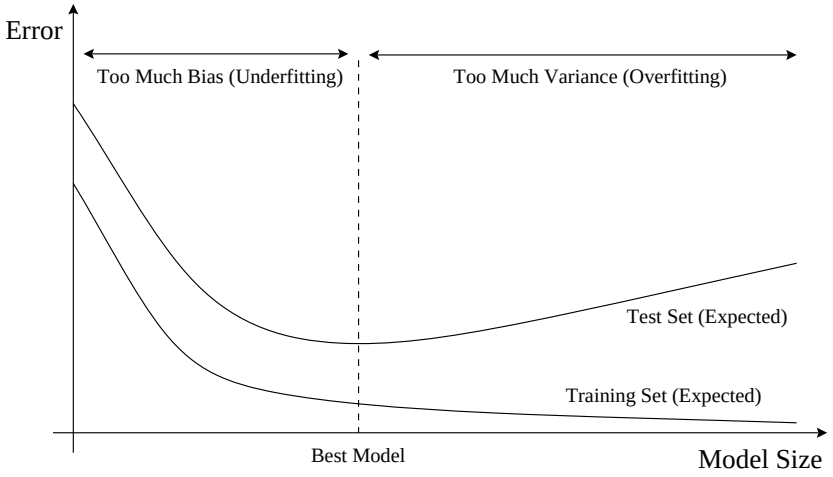

Figure 1: Idealized depiction of the expected training error and expected test error (prediction risk) versus model size for models trained on a fixed finite training sample. Note the regions of underfitting (high model bias) and overfitting (high model variance). Model selection techniques attempt to find the optimal tradeoff between bias and variance. The optimal model corresponds to a global minimum of the expected test error or prediction risk curve. Since the prediction risk can not be computed directly, it must be estimated. Examples of actual (not expected) training error, test error, and estimated prediction risk curves can be found in Utans and Moody (1991), Moody and Utans (1992), and Moody and Yarvin (1992).

while algebraic estimates in the regression context include various formulae derived for linear models, for example *generalized cross–validation (GCV)* (Craven and Wahba, 1979; Golub, Heath and Wahba, 1979), Akaike's *final prediction error (FPE)* (Akaike, 1970), Akaike's *information criterion A (AIC)* (Akaike, 1973), and *predicted squared error (PSE)* (see discussion in Barron (1984)), and the recently proposed *generalized prediction error (GPE)* for nonlinear models (Moody (1991; 1992; 1995)).

2 Prediction Risk

2.1 Prediction Risk for Squared Error Loss

The notion of generalization ability can be defined precisely as the *prediction risk*, the expected performance of an estimator is predicting new observations. We present here a brief, simplified description for the most typical case: a signal plus noise data generation model with a squared error loss function. (We present more general formulations in Moody (1992) and Moody (1995).) For comprehensive discussions

of the standard approaches to prediction risk estimation, see Eubank (1988), Hastie and Tibshirani (1990), and Wahba (1990).

Consider a set of observations $D = \{(\vec{x}_j, t_j); j = 1 \ldots N\}$ that are assumed to be generated as

$$t_j = \mu(x_j) + \epsilon_j \tag{1}$$

where $\mu(x)$ is an unknown function, the inputs x_j are drawn independently with an unknown stationary probability density function $p(x)$, the ϵ_j are independent random variables with zero mean ($\overline{\epsilon} = 0$) and variance σ_ϵ^2, and the t_j are the observed target values.

The learning or regression problem is to find an estimate $\hat{\mu}_\lambda(x; D)$ of $\mu(x)$ given the data set D from a class of predictors or models $\mu_\lambda(x)$ indexed by λ. In general, $\lambda \in \Lambda = (S, A, W)$, where $S \subset X$ denotes a chosen subset of the set of available input variables X, A is a selected architecture within a class of model architectures $\mathcal{A}$, and W are the adjustable parameters (weights) of architecture A. Here we make explicit the dependence of $\hat{\mu}$ on the available data D used for training. $\hat{\mu}$ also depends on the architecture class $\mathcal{A}$ which is chosen independently of the data set D and its size N.

The *prediction risk* $P(\lambda)$ is defined as the expected performance on future data:

$$P(\lambda) = \int dx\ p(x)[\mu(x) - \hat{\mu}(x)]^2 + \sigma_\epsilon^2\ . \tag{2}$$

(Here, we have used the squared error, but $P(\lambda)$ can be defined for other loss functions as well.) This can be approximated by the expected performance on a finite test set:

$$P(\lambda) \approx E\{\frac{1}{N}\sum_{j=1}^{N}(t_j^* - \hat{\mu}_\lambda(x_j^*))^2\}\ , \tag{3}$$

where (x_j^*, t_j^*) are new observations that were not used in constructing $\hat{\mu}_\lambda(x)$. In what follows, we shall use $P(\lambda)$ as a measure of the generalization ability of a model.

Our strategy is to choose an architecture λ in the model space Λ which minimizes an estimate of the prediction risk $P(\lambda)$. The set of networks considered is determined by the heuristic search strategies described in Section 6.

2.2 Prediction Risk for Nonlinear Models

The training error functions of nonlinear models, such as two layer perceptrons, often contain many local minima (see Figure 2). Each minimum should be viewed as defining a different predictor. In order to seek out good local minima, a good learning procedure must therefore include both a gradient-based optimization algorithm and a technique like random restart which enables sampling of the space of minima.

Since the models we use as predictors are trained to a particular "good" local minimum in weight space, we are interested in estimating the generalization ability

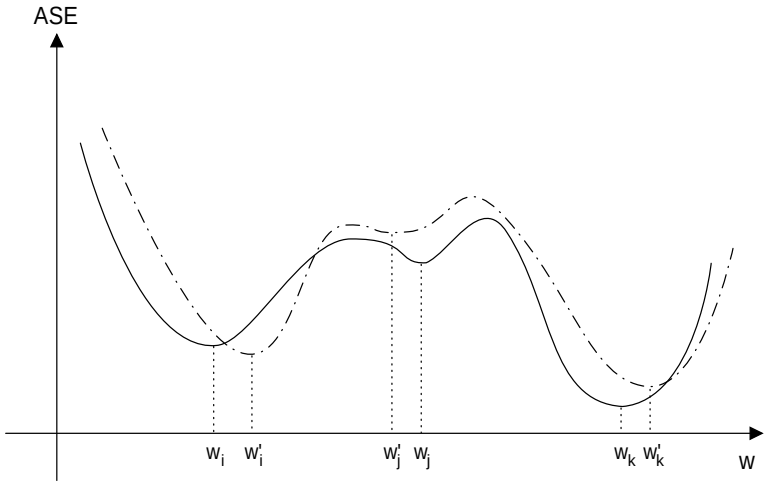

Figure 2: A nonlinear model can have many local minima in the error function. Each local minimum $\mathbf{w}_i$, $\mathbf{w}_j$ and $\mathbf{w}_k$ (solid curve) corresponds to a different set of parameters and thus to a different model. Training on a different finite sample of data or retraining on a subsample, as in nonlinear cross-validation, gives rise to a slightly different error curve (dashed) and perturbed minima $\mathbf{w}_i'$, $\mathbf{w}_j'$ and $\mathbf{w}_k'$. Variations due to data sampling in error curves and their minima are termed *model variance*.

or prediction risk associated with that minimum and not others. This point is important for the estimation procedures described in the next section.

Note that different finite training samples of fixed size will result in slightly different error surfaces in weight space. This is illustrated in Figure 2 by the difference between the solid and dashed curves. Note that the minima of the dashed curve differ slightly from the minima of the solid curve. This effect gives rise to *model variance*.

2.3 Estimates of the Prediction Risk

Since it is not possible to exactly calculate the prediction risk P_λ given only a finite sample of data, we have to estimate it. The standard method based on test–set validation is not advisable when the data set is small.

Cross-validation (CV) is a sample re–use method for estimating prediction risk; it makes maximally efficient use of the available data. We have developed a nonlinear refinement refinement of CV called NCV. Algebraic estimates, such as generalized cross–validation (GCV), the final prediction error (FPE), the predicted squared error (PSE), and the generalized prediction error (GPE), combine the average training squared error (ASE) with a measure of the model complexity. These will be discussed in the next sections.

3 Test-Set Validation and Cross-Validation

3.1 Test-Set Validation

If enough data is available, it is possible to use only part of the data for training the network. The remaining exemplars form a test-set that can be used to estimate the prediction risk. The obvious disadvantage of this method is that not all data is used for training. Even in cases where the data set is large, one would like to use as much data as possible for training, since the estimation error associated with model variance becomes worse as the training set size is reduced. However, if the test-set is too small, an accurate estimate of the prediction risk cannot be obtained. Test-set validation becomes practical only if the data-sets are very large or new data can be generated cheaply.

3.2 Cross-Validation: General Formulation

Cross–Validation is a method that makes minimal assumptions on the statistics of the data. The idea of cross-validation can be traced back to Mosteller and Tukey (1968). For reviews, see Stone (1974; 1978), Geisser (1975), Eubank (1988), Hastie and Tibshirani (1990), and Wahba (1990). For our presentation of the general method in this section, we follow Eubank. We then present a refinement of the method for nonlinear models called NCV in Section 3.3.

Denoting a predictor trained on all N data samples by $\hat{\mu}_\lambda(x)$, let $\hat{\mu}_{\lambda(j)}(x)$ be a predictor trained using all observations except (x_j, t_j) such that $\hat{\mu}_{\lambda(j)}(x)$ minimizes

$$ASE_j = \frac{1}{(N-1)} \sum_{k \neq j} \left(t_k - \hat{\mu}_{\lambda(j)}(x_k)\right)^2 \quad . \tag{4}$$

For $\hat{\mu}_{\lambda(j)}(x)$ we can treat the j^{th} omitted datum as a test set, and repeat this for all j. Then, an asymptotically unbiased estimator for the prediction risk $P(\lambda)$ of the model trained on *all* N *exemplars* $\hat{\mu}_\lambda(x)$ is the *cross-validation average squared error*

$$CV(\lambda) = \frac{1}{N} \sum_{j=1}^{N} \left(t_j - \hat{\mu}_{\lambda(j)}(x_j)\right)^2 \quad . \tag{5}$$

This form of $CV(\lambda)$ is known as *leave–one–out* cross–validation.

However, $CV(\lambda)$ in (5) is expensive to compute for neural network models; it involves constructing an additional N networks, each trained with $N-1$ patterns. For the work described in this paper we therefore use a variation of the method, *v-fold cross–validation*, that was introduced by Geisser (1975) and Wahba and Wold (1975). Instead of leaving out only one observation for the computation of the sum in (5), we delete larger subsets of D.

Let the data D be divided into v randomly selected disjoint subsets D_j of roughly equal size: $\cup_{j=1}^{v} D_j = D$ and $\forall i \neq j,\ D_i \cap D_j = \emptyset$. Let N_j denote the number of observations in subset D_j. Let $\hat{\mu}_{\lambda(D_j)}(x)$ be an estimator trained on all data except

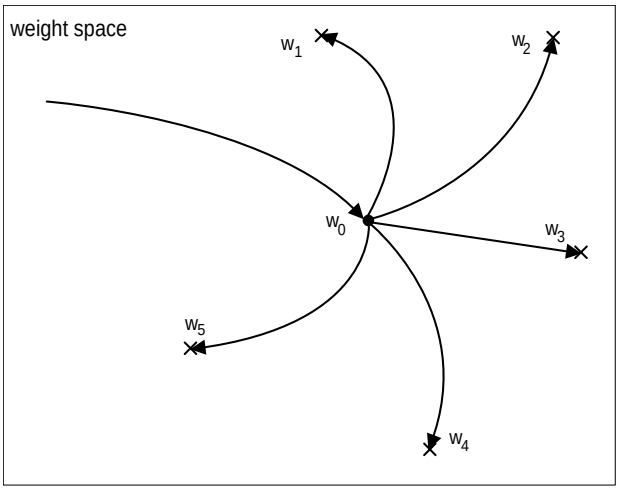

Figure 3: Illustration of the computation of 5–fold *nonlinear cross-validation (NCV)*. First, the network is trained on all data to obtain $\mathbf{w}_0$ which is used as starting point for the cross-validation. Each subset $D_i, i = 1 \ldots 5$ is removed from the training data D in turn. The network is trained, starting at $\mathbf{w}_0$, using the remaining data. This "perturbs" the weights to obtain $\mathbf{w}_i$. The test error of the "perturbed model" w_i is computed on the hold-out sample D_i. The average of these errors is the 5-fold CV estimate of the prediction risk for the model with weights w_0.

for $(x,t) \in D_j$. Then, the cross-validation average squared error for subset j is defined as

$$CV_{D_j}(\lambda) = \frac{1}{N_j} \sum_{(x_k,t_k)\in D_j} \left(t_k - \hat{\mu}_{\lambda(D_j)}(x_k)\right)^2 \quad . \tag{6}$$

These are averaged over j to obtain the v-fold cross-validation estimate of prediction risk:

$$CV(\lambda) = \frac{1}{v} \sum_j CV_{D_j}(\lambda) \quad . \tag{7}$$

Typical choices for v are 5 and 10. Note that leave–one–out CV is obtained in the limit $v = N$. Note that CV is a nonparametric estimate of the prediction risk that relies only on the available data.

3.3 NCV: Cross-Validation for Nonlinear Models

The frequent occurrence of multiple minima in nonlinear models (see Figure 2), each of which represents a different predictor, requires a refinement of the cross-validation procedure. This refinement, *nonlinear cross-validation (NCV)*, was implemented by

Utans and Moody (1991) and Moody and Utans (1992) and is illustrated in Figure 3 for $v = 5$.

A network is trained on the entire data set D to obtain a model $\hat{\mu}_\lambda(x)$ with weights $\mathbf{w}_0$. These weights are used as the starting point for the v-fold cross–validation procedure. Each subset D_j is removed from the training data in turn. The network is re-trained using the remaining data starting at $\mathbf{w}_0$ (rather than using random initial weights). Under the assumption that deleting a subset from the training data does not lead to a large difference in the locally-optimal weights, the retraining from $\mathbf{w}_0$ "perturbs" the weights to obtain $\mathbf{w}_i, i = 1 \ldots v$. The $CV(\lambda)$ computed for the "perturbed models" $\hat{\mu}_{\lambda(D_j)}(x)$ according to Equation (7) thus estimates the prediction risk for the model with locally-optimal weights $\mathbf{w}_0$ as desired, and not the performance of other predictors at other local minima.

If the network is trained from random initial weights for each subset, it could converge to a different minimum corresponding to $\mathbf{w}_i$ different from the one corresponding to $\mathbf{w}_0$. This would correspond to a different model. Thus, starting from $\mathbf{w}_0$ assures us that the cross-validation estimates the prediction risk for a particular model in question corresponding to $\mathbf{w} \approx \mathbf{w}_0$.

In Figure 2, the unperturbed model could be associated with one of the minima of the solid error curve (say w_k), and a perturbed model would be associated with the corresponding minimum of the dashed curve with weights w'_k. Our NCV algorithm attempts to avoid finding a "wrong" perturbed model (e.g., with weights w'_i). NCV has the additional benefit of having much less computational load than would be incurred by retraining from random initial weights.

Note that the same perturbation procedure described here yields nonlinear refinements of the bootstrap and jackknife algorithms as well. (See for example Efron and Gong (1983).)

4 Algebraic Estimates of Prediction Risk

4.1 Predicted Squared Error for Linear Models

For linear regression models with the squared error loss function, a number of useful algebraic estimates for the prediction risk have been derived. These include the well known *generalized cross–validation (GCV)* (Craven and Wahba, 1979; Golub *et al.*, 1979) and Akaike's *final prediction error (FPE)* (Akaike, 1970) formulas:

$$GCV(\lambda) = \mathit{ASE}(\lambda) \frac{1}{\left(1 - \frac{Q(\lambda)}{N}\right)^2} \qquad FPE(\lambda) = \mathit{ASE}(\lambda) \left(\frac{1 + \frac{Q(\lambda)}{N}}{1 - \frac{Q(\lambda)}{N}}\right) \quad . \quad (8)$$

$Q(\lambda)$ denotes the number of weights of model λ. Note that although GCV and FPE are slightly different for small sample sizes, they are asymptotically equivalent for large N:

$$GCV(\lambda) \approx FPE(\lambda) \approx \mathit{ASE}(\lambda) \left(1 + 2\frac{Q(\lambda)}{N}\right) \quad (9)$$

A more general expression of *predicted squared error (PSE)* is:

$$PSE(\lambda) = ASE(\lambda) + 2\widehat{\sigma}^2 \frac{Q(\lambda)}{N}, \tag{10}$$

where $\widehat{\sigma}^2$ is an estimate of the noise variance in the data. Estimation strategies for (10) and its statistical properties have been analyzed by Barron (1984). FPE is obtained as special case of PSE by setting $\widehat{\sigma}^2 \equiv ASE(\lambda)/(N-Q(\lambda))$. See Eubank (1988), Hastie and Tibshirani (1990) and Wahba (1990) for tutorial treatments.

It should be noted that that PSE, FPE and GCV are asymptotically unbiased estimates of the prediction risk for the neural network models considered here under certain conditions. These are: (1) the noise ϵ_j in the observed targets t_j is independent and identically distributed, (2) the resulting model is unbiased, (3) weight decay is not used, and (4) the nonlinearity in the model can be neglected. For PSE, we further require that an asymptotically unbiased estimate of $\widehat{\sigma}^2$ is used. In practice, however, essentially all neural network fits to data will be biased and/or have significant nonlinearity.

Although PSE, FPE and GCV are asymptotically unbiased only under the above assumptions, they are much cheaper to compute than NCV since no retraining is required.

4.2 Generalized Prediction Error (GPE) for Nonlinear Models

The predicted squared error PSE, and therefore the final prediction error FPE, are special cases of the *generalized prediction error GPE* (Moody (1991; 1992; 1995)). We present an abbreviated description here.

GPE estimates prediction risk for biased nonlinear models which may use general loss functions and include regularizers such as weight decay. The algebraic form is

$$GPE(\lambda) \equiv \mathcal{E}_{\text{train}}(\lambda) + \frac{2}{N} \operatorname{tr} \widehat{V} \widehat{G}(\lambda) \ , \tag{11}$$

where $\mathcal{E}_{\text{train}}(\lambda)$ is the training set error (average value of loss function on training set), $\widehat{V}$ is a nonlinear generalization of the estimated *noise covariance matrix* of the observed targets, and $\widehat{G}(\lambda)$ is the estimated *generalized influence matrix*, a nonlinear analog of the standard influence or hat matrix.

GPE can be expressed in an equivalent form as:

$$GPE(\lambda) = \mathcal{E}_{\text{train}}(\lambda) + 2\ \widehat{\sigma}^2_{eff} \frac{\widehat{Q}_{eff}(\lambda)}{N}, \tag{12}$$

where $\widehat{Q}_{eff} \equiv \operatorname{tr} \widehat{G}$ is the estimated *effective* number of model parameters, and $\widehat{\sigma}^2_{eff} \equiv (\operatorname{tr} \widehat{V} \widehat{G})/(\operatorname{tr} \widehat{G})$ is the estimated effective noise variance in the data. For nonlinear and/or regularized models, $\widehat{Q}_{eff}(\lambda)$ is generally not equal to the number of weights $Q(\lambda)$.

When the noise in the target variables is assumed to be independent with uniform variance and the squared error loss function is used, (12) simplifies to:

$$GPE(\lambda) = ASE(\lambda) + 2\widehat{\sigma}^2 \frac{\widehat{Q}_{eff}(\lambda)}{N} \ . \tag{13}$$

Note that replacing $\widehat{Q}_{eff}(\lambda)$ with $Q(\lambda)$ gives the expression for PSE. Various other special cases of (11) and (13) have been derived by other authors and can be found in Eubank (1988), Hastie and Tibshirani (1990) and Wahba (1990). Larsen (1992) has extended (11) to autoregressive time series models, and Liu (1993) and Moody (1995) have shown that N-fold cross validation is equivalent to $O(1/N)$ to the general form of GPE in equation (11).

5 Neural Network Architecture Selection

For the discussion of architecture selection in this paper, we focus on the most widely used neural network architecture, the two-layer *perceptron* (or *backpropagation*) network. The response function $\hat{\mu}_\lambda(x)$ for such a network with I_λ input variables, H_λ internal (hidden) neurons, and a single output neuron is:

$$\hat{\mu}_\lambda(x) = f(\, v_0 + \sum_{\alpha=1}^{H_\lambda} v_\alpha \, g(w_{\alpha 0} + \sum_{\beta=1}^{I_\lambda} w_{\alpha\beta} \, x_\beta)) \; . \tag{14}$$

Here, x_β are the input variables, f and g are typically sigmoidal nonlinearities, the $w_{\alpha\beta}$ and $w_{\alpha 0}$ are input weights and thresholds, the v_α and v_0 are the second layer weights and threshold, and the index λ is an abstract label for the specific two layer perceptron network architecture. While we consider for simplicity this restricted class of perceptron networks in this paper, our approach can be easily generalized to networks with multiple outputs and multiple layers.

For two layer perceptrons, the *architecture selection problem* is to find a good (hopefully near-optimal) architecture λ for modeling a given data set. The architecture λ is characterized by

- the number of hidden units H_λ,
- the number of input variables I_λ,
- and the subset of weights v_α and $w_{\alpha\beta}$ that are non-zero.

If all of the v_α and $w_{\alpha\beta}$ are non-zero, the network is referred to as *fully connected*. If many of the input weights $w_{\alpha\beta}$ are zero, the network is *sparsely connected*, while setting to zero one of the v_α (or all $w_{\alpha\beta}$ for fixed α) corresponds to removing a hidden unit.

Note that for brevity of exposition in the sequel, we denote weights and biases in either layer of the network generically by w.

5.1 Complexity of Exhaustive Search

If we consider a set of two layer perceptrons up to some maximal size determined by $I_{\max}$ and $H_{\max}$, then the maximal fully connected network has

$$M_{\max} = 1 + H_{\max}(I_{\max} + 2)$$

weights. All smaller two layer networks can be obtained from the maximal network by setting weights and thresholds to zero. The total number of resulting network topologies (treating the ordering of the hidden units as unique and thus not considering permutations of them) is

$$N_{\max} = 2^{M_{\max}}$$

For example, a set of networks with $I_{\max} = 10$ and $H_{\max} = 12$ yields $N_{\max} = 2^{145} \approx 4.46 \times 10^{43}$. Even though the maximal network in this example is modest in size, $N_{\max}$ is a prohibitively large number of network topologies to search exhaustively. If each of the $N_{\max}$ networks could be trained and tested in one microsecond of CPU time, the total time required to train and test all $N_{\max}$ networks in this example would be 1.41×10^{30} years or roughly 10^{20} times the age of the universe.

Even if exhaustive search for an optimum (as measured by a selection criterion) were computationally feasible, it would likely prove to be pointless, since based on our experience, many network architectures are likely to provide similar performance. (See, for example, Figure 3 of Moody and Yarvin (1992).) Moreover, the model variance for networks trained on small, noisy data sets will make the near-optimal models indistinguishable from the asymptotically optimal model.[1]

Thus, we are compelled to consider restricted search strategies over the space of network architectures. This is the subject of Section 6.

6 Heuristic Search over the Space of Perceptron Architectures

Given the futility of exhaustively sampling the space of possible networks in search of an optimum, we present efficient heuristic search algorithms. These were developed while building a model for corporate bond rating prediction. Brief descriptions of the original work can be found in Utans and Moody (1991) and Moody and Utans (1992). A more detailed description of the bond rating problem and our empirical results is presented in Moody and Utans (1994).

First, a locally-optimal number of internal units is selected from a sequence of fully connected networks with increasing number of hidden units. An efficient algorithm, *sequential network construction (SNC)*, is used for training the networks in this sequence. Then, using the optimal fully connected network, input variables are pruned via *sensitivity based pruning (SBP)* and weights are pruned via *optimal brain damage (OBD)*.

Figure 4 depicts two search strategies. After selecting the number of hidden units H_λ, input removal and weight elimination can be carried out in parallel (A) or sequentially (B). The first strategy (A), which was used for the bond rating simulations described in Utans and Moody (1991) and Moody and Utans (1992), allows a comparison of the results of the input and weight pruning steps but requires a final step of combining the resulting networks to obtain the final network architecture. On the other hand, the sequential approach (B) is less costly to implement since for

[1] The term *asymptotic* refers to the limit of infinite training set size.

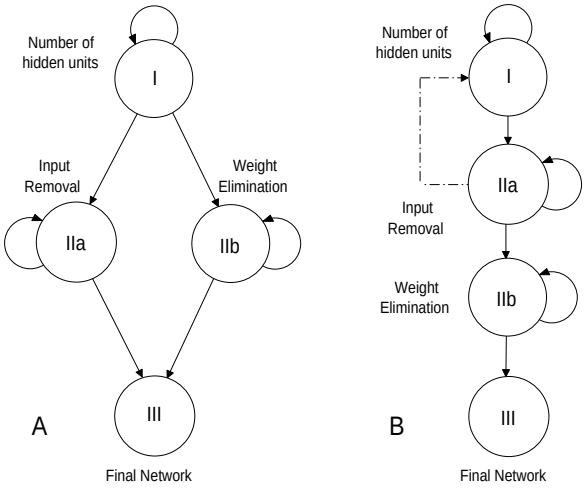

Figure 4: Heuristic Search Strategies: After selecting the number of hidden units H_λ, the input removal and weight elimination can be carried out in parallel (A) or sequentially (B). In (B), the selection of the number of hidden units and removal of inputs may be iterated (dashed line).

the weight elimination stage, the network is already reduced in size by removing inputs. This alternative corresponds to a *coarse to fine* strategy: pruning inputs removes groups of weights while the final weight elimination is concerned with individual weights. A refinement of the sequential approach is to re-determine the optimal number of hidden units after a good set of input variables has been selected or after the removal of each unnecessary input variable. This results in an iterative procedure (dashed line in Figure 4 (B)).

6.1 Selecting the Number of Hidden Units

For determining the number of hidden units, we construct and train a nested set of models, using an efficient algorithm which we call *sequential network construction (SNC)*.

Before proceeding, we would like to note that many authors have independently proposed iterative network construction algorithms. Probably the best known of these is the cascade correlation algorithm (Fahlman and Lebiere, 1990). Cascade correlation was preceded by Ash (1989); see also Moody (1989). We have not attempted to exhaustively review this area, nor do we claim that SNC is necessarily unique or optimal.

6.1.1 The SNC Algorithm

The SNC algorithm constructs a sequence of networks, each of which is fully connected and uses all input variables, differing only in the number of hidden units. The sequence is built as follows: First a network with a small number of hidden units is trained, using random initial weights. Larger networks are obtained iteratively, by adding units in clumps of size C to the largest net in the sequence. This continues until a network of some predetermined maximum size has been produced.

When a larger network, of size $H_\lambda + C$, is constructed, it is trained as follows:

1. The weights of the previous network are used as initial weights for the first H_λ units of the new network. The weights of the newly added clump of C units are initialized to small random values.

2. The weights of the new clump are trained to a local optimum while keeping the weights of the first H_λ units fixed.

3. Finally, all weights are allowed to vary, and are trained until a local optimum for the entire network is reached.

The SNC algorithm is designed to solve two problems. First, it substantially reduces computation time compared to what would be required if larger networks were trained from scratch. Secondly, the sequence of networks constructed comprise a *nested set*.

6.1.2 Nested Models and Inverse Pruning

In general, a nonlinear network has many local minima, and each local minimum corresponds to a different set of parameters and thus to a different model (see Figure 2). The SNC algorithm, on the other hand, provides some continuity in model space by constructing a nested sequence in which larger networks contain the smaller networks.

The sequential construction accomplishes this nesting as follows. As larger networks are constructed, the features discovered by the smaller network's hidden units will continue to be used by the larger networks. In effect this means that as the sequence grows, the networks are trained to learn corrections to the mappings discovered so far. Moreover, we have found that the effect on existing hidden units of adding a new hidden unit once the model is big enough is likely to be small.

Because of the resulting continuity in model space, SNC can be thought of as an *inverse pruning* procedure. Since the most recently added clump of units serves only to correct a mapping learned already, it is highly likely that a node pruning method would prune these units first. Also, it is clearly less expensive computationally to construct a sequence of networks from small to large than it is to first train large networks and prune from large to small.

Three advantages are obtained by constructing a nested set of models. First, the sequence will have monotonically decreasing (or non-increasing) training error. Secondly, the sequence is likely to have an easily identifiable minimum of prediction

risk. Thirdly, architecture selection via prediction risk has a formal connection to the hypothesis testing approach to pruning when the set of models is nested (see for example the discussion in Akaike (1974)). The *inverse pruning* approach allows the theory and techniques of hypothesis testing to be applied to models trained via SNC.

Note that in spite of these advantages of SNC, it is possible that pruning nodes from a larger network may give rise to a better fit to the data and better generalization for a given final number of nodes. This is not to say, however, that a larger network obtained via SNC might not perform as well or better. Resolving the issue of whether SNC or node pruning is more effective in general requires a systematic study.

Finally, for each network in the sequence produced by SNC, an estimate of the prediction risk is computed. The network selected at this stage for further refinement via input and weight pruning is the one with the smallest estimated prediction risk.

6.2 Pruning Inputs and Weights via Directed Search

After the number of hidden units of the fully connected network with all available input variables is determined, the next step of our heuristic search is to select input variables and remove individual weights from the network.

As before, we evaluate a candidate network architecture by computing an estimate of the prediction risk. In order to avoid searching over the very large range of architectures obtained by considering all possible combinations of inputs and all possible connection structures, we propose a directed search strategy using the *sensitivity-based input pruning (SBP)* and *optimal brain damage (OBD)* algorithms.

With these algorithms, candidate architectures are constructed by evaluating the effect of removing an input variable or an individual weight from the fully connected network. These are ranked in order of increasing training error. Inputs and weights are then removed following a "Best First" strategy, i.e. selecting the input or weight that, when removed, increases the training error least. The candidate architecture is obtained by retraining the remaining weights, starting from their previous values, to find a new minimum in weight space. Note that we assume that the weights obtained after making a small modification to the network, such as removing a single input variable or a single weight, brings us closer to a good solution.

Note that the weight elimination stage can reduce the number of hidden units if all weights to or from a particular hidden unit are eliminated. For example, for the model considered here with a single output unit, eliminating a weight from a hidden unit to the output unit effectively removes that hidden unit from the network. Pruning units this way is more costly computationally than choosing a good number of hidden units in the first place (as is the purpose of our SNC and unit selection algorithm). However, both methods can complement each other.

6.2.1 Pruning of Input Variables via Sensitivity Analysis

In Moody and Utans (1992) and Utans and Moody (1991), we proposed a simple *sensitivity-based pruning* method for input variables *(SBP)*. The SBP algorithm computes a *sensitivity measure* S_i to evaluate the change in training error that would

result if input x_i were removed from the network. The sensitivity of the network model to variable i is defined as:

$$S_i = \frac{1}{N} \sum_j S_{ij} \quad , \tag{15}$$

where S_{ij} is the sensitivity computed for exemplar x_j. Since there are usually many fewer inputs than weights, a direct evaluation of S_i is feasible:

$$S_{ij} = SE(\overline{x}_i, w_\lambda) - SE(x_{ij}, w_\lambda) \qquad \text{with} \qquad \overline{x}_i = \frac{1}{N} \sum_{j=1}^{N} x_{ij} \ . \tag{16}$$

S_i measures the effect on the training squared error (*SE*) of replacing the i^{th} input x_i by its average $\overline{x}_i$ for all exemplars. Replacement of a variable by its average value removes its influence on the network output.

Note that in computing S_i, no retraining is done in evaluating $SE(\overline{x}_i, w_\lambda)$. Also note that it is not sufficient to just set $x_{ij} = 0 \ \forall \ j$, because the value of the bias of each hidden unit was determined during training and would not be offset properly by setting the input arbitrarily to zero. Of course, if the inputs are normalized to have zero mean prior to training, then setting an input variable to zero is equivalent to replacing it by its mean.

If the number of input variables I_λ is large, S_i is expensive to compute and can be approximated. For a small change of the i^{th} input of the j^{th} exemplar, dx_{ij},

$$dSE = \sum_i \frac{\partial SE}{\partial x_{ij}} dx_{ij} + \frac{1}{2} \sum_{i_1 i_2 j} \frac{\partial^2 SE}{\partial x_{i_1 j} \partial x_{i_2 j}} dx_{i_1 j} dx_{i_2 j} + O(||x_{ij}||^3) \quad . \tag{17}$$

The linear term in Equation (17) approximates S_{ij} as

$$\hat{S}_{ij} = \frac{\partial SE}{\partial x_{ij}} dx_{ij} \qquad \text{with} \qquad dx_{ij} = \overline{x}_i - x_{ij} \quad , \tag{18}$$

and the derivative information can be efficiently computed as a minor extension to the backpropagation algorithm used for training.

Note that the SBP algorithm neglects possible correlations between inputs. An improvement of the method would be to first orthogonalize the inputs (for example by a KL transform) and perform the sensitivity analysis using the transformed inputs. The pruning would be performed on input eigennodes rather than the actual input variables. This approach has been explored recently by Levin, Leen and Moody (1994).

A related input pruning algorithm based on a different approximation of the sensitivity measure has been proposed by Refenes, Francis and Zapranis (1994).

6.2.2 Weight Pruning via "Optimal Brain Damage"

Optimal Brain Damage (OBD) was introduced by LeCun, Denker and Solla (1990) as a method to reduce the number of weights in a neural network to avoid overfitting.

OBD is designed to select those weights in the network whose removal will have a small effect on the training average squared error (ASE). Assuming that the original network is too large, removing these weights and retraining the now smaller network should improve the generalization performance.

The method approximates ASE at a minimum $\mathbf{w} = \mathbf{w}^*$ in weight space by a diagonal quadratic expansion. The change in ASE caused by a small change $d\mathbf{w}$ from $\mathbf{w}^*$ is

$$dASE = \sum_k \frac{\partial ASE}{\partial w_k} dw_k + \frac{1}{2} \sum_{kl} \frac{\partial^2 ASE}{\partial w_k \partial w_l} dw_k dw_l + O(||w_k||^3) \quad , \tag{19}$$

where for simplicity of notation w_k represents any weight in any layer of the network.

Since the approximation is taken at a minimum in weight space, the first term in Equation (19) vanishes. In addition, terms higher order than quadratic are ignored. Thus, Equation (19) reduces to

$$dASE = \frac{1}{2} \sum_k \frac{\partial^2 ASE}{\partial w_k \partial w_k} dw_k dw_l \quad . \tag{20}$$

The evaluation of the Hessian becomes prohibitively expensive for large networks. By approximating the Hessian by considering only diagonal terms $\partial^2 ASE / \partial w_k^2$, we assume that weights can be removed individually without influencing each other. The *saliency* defined as

$$s_k = \frac{1}{2} \frac{\partial^2 ASE}{\partial w_k \partial w_k} w_k^2 \tag{21}$$

is a measure (in the diagonal approximation) of the change of ASE when weight w_k is removed from the network by setting its value to zero. Note that the saliency must be computed after training has stopped at a local minimum. The second derivatives required for s_k can be efficiently computed by a method similar to the backpropagation of first derivatives for weight updates during training (LeCun *et al.*, 1990).

The procedure for eliminating weights as described by LeCun *et al.* (1990) consists of ranking the weights in the network according to increasing s_k, removing first one weight or a few weights, then retraining the network, and repeating an arbitrary number of times. In contrast, we accept a network modification only if the expected performance on the test set is improved as measured by a decrease in the estimate of the prediction risk $\widehat{P}(\lambda)$.

6.3 Final Remarks on Pruning

6.3.1 Multiple Minima of Prediction Risk

Note that since the networks are trained to minimize the training error ASE and not $P(\lambda)$, the prediction risk need not be a monotonically decreasing function of the number of inputs or weights eliminated, and there can be multiple minima in the sequence of networks considered. Since the SBP and OBD procedures start with a

fully connected network, it is not sufficient to stop removing weights when $\widehat{P}(\lambda)$ first increases. This is particularly true given that both SBP and OBD are greedy algorithms, and they neglect interactions between multiple inputs and weights. Thus, removing additional inputs or weights even after $\widehat{P}(\lambda)$ first increases can lead to a further reduction of the prediction risk and thus yield a smaller final network.

6.3.2 Other Pruning Methods

A number of other pruning methods besides those we have described here are potentially effective and should be considered when constructing neural network models. These include the *irrelevant hidden unit* and *irrelevant input* hypothesis tests (White, 1989), pruning of units via skeletonization (Mozer and Smolensky, 1990), *optimal brain surgeon OBS* (Hassibi and Stork, 1993), and *principal components pruning PCP* (Levin *et al.*, 1994). It is important to note that all these methods, along with OBD and our method of input pruning via SBP, are closely related to the Wald hypothesis testing procedure (see for example Buse (1982)). In fact, the saliencies used in OBD, OBS, and SBP are special cases of the Wald test statistic.

Acknowledgements

The author thanks Joachim Utans for help in preparing this manuscript and Steve Rehfuss and Eric Wan for careful proof readings. The author is responsible for any remaining errors. This work was supported by the Advanced Research Projects Agency under grant N00014-92-J-4062.

References

Akaike, H. (1970), 'Statistical predictor identification', *Ann. Inst. Statist. Math.* **22**, 203–217.

Akaike, H. (1973), Information theory and the extension of the maximum likelihood principle, Akademia Kiado, Budapest, pp. 267–810.

Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* **AC-19**(6), 716–723.

Akaike, H. (1977), On entropy maximization principle, *in* P. R. Krishnaiah, ed., 'Applications of Statistics', North-Holland Publishing Company, pp. 27–41.

Ash, T. (1989), 'Dynamic node creation in backpropagation neural networks', *Connection Science* **1**(4), 365–375.

Barron, A. (1984), Predicted squared error: a criterion for automatic model selection, *in* S. Farlow, ed., 'Self–Organizing Methods in Modeling', Marcel Dekker, New York.

Buse, A. (1982), 'The likelihood ratio, wald, and lagrange multiplier test: An expository note', *The American Statistician* **36**(3), 153–157.

Craven, P. and Wahba, G. (1979), 'Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation', *Numer. Math.* **31**, 377–403.

Efron, B. and Gong, G. (1983), 'A leisurely look at the bootstrap, the jackknife and cross–validation', *The American Statistician* **37**(1), 36–48.

Eubank, R. L. (1988), *Spline Smoothing and Nonparametric Regression*, Marcel Dekker, Inc.

Fahlman, S. E. and Lebiere, C. (1990), The cascade-correlation learning algorithm, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 2', Morgan Kaufmann Publishers, San Mateo, CA, pp. 525–532.

Geisser, S. (1975), 'The predictive sample reuse method with applications', *Journal of The American Statistical Association* **70**(350).

Geman, S., Bienenstock, E. and Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural Computation* **4**(1), 1–58.

Golub, G., Heath, H. and Wahba, G. (1979), 'Generalized cross validation as a method for choosing a good ridge parameter', *Technometrics* **21**, 215–224.

Hassibi, B. and Stork, D. G. (1993), Second order derivatives for network pruning: Optimal brain surgeon, *in* S. J. Hanson, J. D. Cowan and C. L. Giles, eds, 'Advances in Neural Information Processing Systems 5', Morgan Kaufmann Publishers, San Mateo, CA, pp. 164–171.

Hastie, T. J. and Tibshirani, R. J. (1990), *Generalized Additive Models*, Vol. 43 of *Monographs on Statistics and Applied Probability*, Chapman and Hall.

Larsen, J. (1992), A generalization error estimate for nonlinear systems, *in* 'Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing', IEEE Service Center, Piscataway, NJ, pp. 29–38.

LeCun, Y., Denker, J. S. and Solla, S. A. (1990), Optimal brain damage, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 2', Morgan Kaufmann Publishers.

Levin, A. U., Leen, T. K. and Moody, J. E. (1994), Fast pruning using principal components, *in* J. Cowan, G. Tesauro and J. Alspector, eds, 'Advances in Neural Information Processing Systems 6', Morgan Kaufmann Publishers, San Francisco, CA.

Liu, Y. (1993), Neural network model selection using asymptotic jackknife estimator and cross-validation method, *in* S. J. Hanson, J. D. Cowan and C. L. Giles, eds, 'Advances in Neural Information Processing Systems 5', Morgan Kaufmann Publishers, San Mateo, CA, pp. 599–606.

Moody, J. E. (1989), Fast learning in multi-resolution hierarchies, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 1', Morgan Kaufmann Publishers, San Mateo, CA.

Moody, J. E. (1991), Note on generalization, regularization and architecture selection in nonlinear learning systems, *in* B. H. Juang, S. Y. Kung and C. A. Kamm, eds, 'Neural Networks for Signal Processing', IEEE Signal Processing Society, pp. 1–10.

Moody, J. E. (1992), The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 847–854.

Moody, J. E. (1995), 'The *effective* number of parameters and generalized prediction error for nonlinear regression.', *Manuscript in preparation* .

Moody, J. E. and Utans, J. (1992), Principled architecture selection for neural networks: Application to corporate bond rating prediction, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 683–690.

Moody, J. E. and Yarvin, N. (1992), Networks with learned unit response functions, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 1048–55.

Moody, J. and Utans, J. (1994), Architecture selection strategies for neural networks: Application to corporate bond rating prediction, *in* A. N. Refenes, ed., 'Neural Networks in the Captial Markets', John Wiley & Sons.

Mosteller, F. and Tukey, J. W. (1968), Data analysis, including statistics, *in* G. Lindzey and E. Aronson, eds, 'Handbook of Social Psychology, Vol. 2', Addison–Wesley. (first edition 1954).

Mozer, M. C. and Smolensky, P. (1990), Skeletonization: A technique for trimming the fat from a network via relevance assessment, *in* D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems 1', Morgan Kaufmann Publishers, San Mateo, CA.

Refenes, A. N., Francis, G. and Zapranis, A. D. (1994), 'Stock performance modeling using neural networks: A comparative study with regression models', *Neural Networks* . accepted June 1993.

Rissanen, J. (1978), 'Modeling by shortest data description', *Automatica* **14**, 465–71.

Schwartz, G. (1978), 'Estimating the dimension of a model', *Ann. Stat.* **6**, 461–4.

Stone, M. (1974), 'Cross–validatory choice and assessment of statistical predictions', *Roy. Stat. Soc.* **B36**, 111–147.

Stone, M. (1978), 'Cross–validation: A review', *Math. Operationsforsch. Statist., Ser. Statistics* **9**(1).

Utans, J. and Moody, J. (1991), Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction, *in* 'Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street', IEEE Computer Society Press, Los Alamitos, CA.

Wahba, G. (1990), *Spline Models for Observational Data*, Vol. 59 of *Regional Conference Series in Applied Mathematics*, SIAM Press, Philadelphia.

Wahba, G. and Wold, S. (1975), 'A completely automatic french curve: Fitting spline functions by cross–validation', *Communications in Statistics* **4**(1), 1–17.

White, H. (1989), 'Learning in artificial neural networks: A statistical perspective', *Neural Computation* **1**, 425–464.