Improving Model Accuracy using Optimal Linear Combinations of Trained Neural Networks*

Sherif Hashem† Bruce Schmeiser‡

Abstract

Neural network (NN) based modeling often requires trying multiple networks with different architectures and training parameters in order to achieve an acceptable model accuracy. Typically, only one of the trained networks is selected as "best" and the rest are discarded.

We propose using optimal linear combinations (OLCs) of the corresponding outputs of a set of NNs as an alternative to using a single network. Modeling accuracy is measured by mean squared error (MSE) with respect to the distribution of random inputs. Optimality is defined by minimizing the MSE, with the resultant combination referred to as MSE-OLC.

We formulate the MSE-OLC problem for trained NNs and derive two closed-form expressions for the optimal combination-weights. An example that illustrates significant improvement in model accuracy as a result of using MSE-OLCs of the trained networks is included.

I. INTRODUCTION

Constructing neural network (NN) based models often involves training a number of networks. The creation of these networks may result during the search for a mixture of network architecture and training parameters that yields an "acceptable" model performance [1, 2]. Typically, the "best" performer is selected while the rest are discarded. In other situations, a number of "small" networks may be individually trained and then combined, instead of training one "larger" network for a given task. Although motivated by the former, the analysis and derivations presented in this paper may be applied under both scenarios.

We propose using a linear combination of the corresponding outputs of a (possibly screened) set of trained NNs, rather than using only the single best trained NN. Combining the trained NNs may help improve the resultant model accuracy and may be superior to the best NN. The combination-weights may be chosen according to any optimality criterion. We minimize mean squared error (MSE) over observed data, a criterion commonly used by both the neural network and statistics communities.

In Section II, related research on combining estimators is briefly discussed. In Section III, we formulate the optimal linear combination problem for trained NNs and derive two expressions for the optimal combination-weights. Estimating the optimal combination-weights using observed data is also discussed. An illustrative example is given in Section IV. The conclusions are summarized in Section V.

*Technical Report SMS 92–16, School of Industrial Engineering, Purdue University. *IEEE Transactions on Neural Networks,* forthcoming. This work was supported by PRF Research Grant 6901627 from Purdue University and by NSF Grant DMS–8717799.

†Pacific Northwest Laboratory, 906 Battelle Boulevard, P.O. Box 999, MSIN K1–87, Richland, WA 99352. Internet: s_hashem@pnl.gov

‡School of Industrial Engineering, Purdue University, West Lafayette, IN 47907–1287. Internet: schmeise@ecn.purdue.edu

II. COMBINING ESTIMATORS

Linear combinations of estimators have been used by the statistics community for a long time [3]. Clemen [4] cites more than 200 studies in his review of the literature related to combining forecasts, including contributions from forecasting, psychology, statistics, and management science literatures. Averaging a set of estimators is frequently compared to the individual estimators, and in many cases performs better [2, 3, 4].

Our approach is similar to the approaches adopted in the forecasting literature (for example see [5]), with differences primarily in the problem formulation and the definitions discussed in Section III.

III. OPTIMAL LINEAR COMBINATIONS OF NEURAL NETWORKS

In this section, we formulate the optimal linear combination (OLC) problem for a set of p trained NNs, and derive two expressions to compute the optimal combination-weights.

The mapping being approximated may be a multi-input-multi-output mapping. However, for the sake of simplicity of the analysis and the derivations, we focus on multi-input-single-output mappings. One possible treatment of the former case is to compute an optimal combination-weights vector for each output separately. Such independent treatment is straightforward and minimizes the total MSE for multi-input-multi-output mappings. In other contexts, a multivariate analysis may be more appropriate, however, is not considered here.

A trained NN accepts a vector-valued input $\vec{x}$ and returns a scalar output (response) $y(\vec{x})$. The approximation error is $\delta(\vec{x}) = t(\vec{x}) - y(\vec{x})$, where $t(\vec{x})$ is the true answer for a given input $\vec{x}$. A linear combination of the outputs of p NNs returns the scalar output $\tilde{y}(\vec{x};\vec{\alpha}) = \sum_{j=1}^{p} \alpha_j y_j(\vec{x})$, with corresponding error $\tilde{\delta}(\vec{x};\vec{\alpha}) = t(\vec{x}) - \tilde{y}(\vec{x};\vec{\alpha})$; where $y_j(\vec{x})$ is the output of the jth network, and α_j is the combination-weight associated with $y_j(\vec{x})$; $j = 1, 2, \ldots, p$.

The problem is to find "good" values for the combination-weights $\alpha_1, \alpha_2, \ldots, \alpha_p$. One approach is to select one of the p networks as "best", say k, set $\alpha_k = 1$ and set the other combination-weights to zero. Using a single network has the advantage of simplicity, but the disadvantage of ignoring the (possibly) useful information in the other $p-1$ networks. Another approach, which is widely used by the forecasting community [4], is to use equal combination-weights (simple averaging). Simple averaging is straightforward but assumes that all the component networks are equally good.

Think of the input $\vec{x}$ as an observation of a random variable $\vec{X}$ from a (usually unknown) multivariate distribution function $F_{\vec{X}}$. Then the true answer is $t(\vec{X})$, the output of the jth network is $y_j(\vec{X})$, and the associated approximation error is $\delta_j(\vec{X})$; $j = 1, 2, \ldots, p$. The linear-combination output is the random variable $\tilde{y}(\vec{X};\vec{\alpha}) = \sum_{j=1}^{p} \alpha_j y_j(\vec{X})$, and the linear-combination error is the random variable $\tilde{\delta}(\vec{X};\vec{\alpha}) = t(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha})$. The OLC is defined by the optimal combination-weights vector $\vec{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_p^*)$ that minimizes the expected loss

$$\int_{\mathcal{S}} \ell(\tilde{\delta}(\vec{X};\vec{\alpha}))\, dF_{\vec{X}}\,,$$

where $\mathcal{S}$ is the support of $F_{\vec{X}}$ and ℓ is a loss-function.

Although various loss functions could be pursued, here we restrict attention to squared-error loss, $\ell(\tilde{\delta}) = \tilde{\delta}^2$. The objective is then to minimize the mean squared error (MSE),

$$\mathrm{MSE}\left(\tilde{y}(\vec{X};\vec{\alpha})\right) = \mathrm{E}\left(\left(\tilde{\delta}(\vec{X};\vec{\alpha})\right)^2\right)\,,$$

where E denotes expected value with respect to $F_{\vec{X}}$. The resultant optimal linear combination is referred to as the MSE-optimal linear combination (MSE-OLC).

A. *Deriving the MSE-OLC Weights*

Differentiating the MSE with respect to $\vec{\alpha}$ leads to the unconstrained optimal combination-weights vector,

$$\vec{\alpha}^* = \Phi^{-1}\vec{\Theta}\,, \tag{1}$$

where $\Phi = [\phi_{ij}] = \left[\mathrm{E}\left(y_i(\vec{X})\, y_j(\vec{X})\right)\right]$ is a $p \times p$ matrix, and $\vec{\Theta} = [\theta_i] = \left[\mathrm{E}\left(t(\vec{X})\, y_i(\vec{X})\right)\right]$ is a $p \times 1$ vector.

Consider also the case of constrained combination-weights, where $\sum_{j=1}^{p} \alpha_j = 1$. In such case, if the y_j's are unbiased (in a statistical sense), then $\tilde{y}$ will also be unbiased. Under the condition that $\sum_{j=1}^{p} \alpha_j = 1$, the optimal combination-weights vector is

$$\vec{\alpha}^* = \Omega^{-1}\,\vec{1}\,/\,(\vec{1}^t\,\Omega^{-1}\,\vec{1}\,)\,, \tag{2}$$

where $\Omega = [\omega_{ij}] = \left[\mathrm{E}\left(\delta_i(\vec{X})\, \delta_j(\vec{X})\right)\right]$ is a $p \times p$ matrix, and $\vec{1}$ is a $p \times 1$ vector with all components equal to one. Equations 1 and 2 are derived in the appendix. The General Ensemble Method [6], developed independently, is similar to our constrained MSE-OLC.

B. *Estimating the MSE-OLC Weights*

In practice, one seldom knows the multivariate distribution $F_{\vec{X}}$. Thus, Φ, Θ, and Ω in Equations 1 and 2 need to be estimated.

If $\mathcal{D}$ is a set of independent random observations from $F_{\vec{X}}$, then each vector $\vec{x} \in \mathcal{D}$ may be treated as equally likely. Hence, Φ, Θ, and Ω may be estimated using

$$\widehat{\phi}_{ij} = \sum_{k=1}^{|\mathcal{D}|} \left(y_i(\vec{x}_k)\, y_j(\vec{x}_k)\right) / |\mathcal{D}| \quad \forall\, i,j\,; \tag{3}$$

$$\widehat{\theta}_i = \sum_{k=1}^{|\mathcal{D}|} \left(t(\vec{x}_k)\, y_i(\vec{x}_k)\right) / |\mathcal{D}| \quad \forall\, i\,; \tag{4}$$

and

$$\widehat{\omega}_{ij} = \sum_{k=1}^{|\mathcal{D}|} \left(\delta_i(\vec{x}_k)\, \delta_j(\vec{x}_k)\right) / |\mathcal{D}| \quad \forall\, i,j; \tag{5}$$

respectively; where $|\mathcal{D}|$ denotes the cardinality of $\mathcal{D}$.

IV. EXAMPLE AND DISCUSSION

Consider the problem of approximating the function

$$t(X) = 0.02\left(12 + 3X - 3.5X^2 + 7.2X^3\right)(1 + \cos 4\pi X)(1 + 0.8\sin 3\pi X)$$

over the interval $[0, 1]$, reported in [7]. The range of $t(X)$ is $[0, 0.9)$. We train three 2-hidden-layers NNs with 5 hidden units in each hidden layer (NN1, NN2, and NN3); and three 1-hidden-layer NNs with 10 hidden units (NN4, NN5, and NN6) using the error-backpropagation algorithm [8, pp. 115–130]. The networks are initialized with independent random connection-weights uniformly-distributed in [-0.3, 0.3]. Each network has one input unit and one output unit. The activation function for the hidden units as well as the output units is the logistic sigmoid function $g(s) = (1 + e^{-s})^{-1}$. A set of 200 independent uniformly-distributed points is used in training all the networks and in estimating the optimal combination-weights as well. Except for the structural differences and the different initial connection-weights, the six networks are trained in the same manner. NN4, which is the best NN in the MSE sense, yields a true — computed with respect to the true (known) function $t(X)$ — MSE of 0.000137, and simple averaging of the outputs of the six trained NNs yields a true MSE of 0.000396.

Using Equations 3 and 4 in Equation 1 yields the unconstrained MSE-OLC with a resultant true MSE of 0.000017, which is 88% less than the MSE produced by the best NN (NN4); and 96% less than the MSE produced by the simple averaging of the six NNs. The resultant root mean squared error (RMS) is 0.004, which is small compared to the range of $t(X)$, indicating that the combined model accurately approximates $t(X)$. For the constrained MSE-OLC, the above results differ by about 1–3%. Such a small difference arises because unconstrained combination-weights for accurate component networks (such as seen in this example) tend to automatically sum to one. The empirical comparisons in [9] show that the effect of constraining the weights is greater for less-accurate component networks.

Thus in this example, using MSE-OLCs of the trained NNs significantly improves model accuracy compared to using the single best NN or using the simple averaging of all the trained NNs.

V. CONCLUSIONS

MSE-OLCs allow straightforward integration of multiple trained networks. Since multiple trained networks are often available as a byproduct of the modeling process, the additional computational effort required to create an MSE-OLC is essentially that of estimating the optimal combination-weights, which is mainly a matrix inverse. The gain in accuracy, compared to using the single best network or the simple averaging of the trained networks, is substantial in our example.

APPENDIX
MSE-OPTIMAL COMBINATION-WEIGHTS

We derive the optimal combination-weights given in Equations 1 and 2 in Section III.

A.1 The Unconstrained Case

In Section III, the MSE is given by $\text{MSE}\left(\tilde{y}(\vec{X};\vec{\alpha})\right) = \text{E}\left(\left(\tilde{\delta}(\vec{X};\vec{\alpha})\right)^2\right)$; where $\tilde{\delta}(\vec{X};\vec{\alpha}) = t(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha})$, and $\tilde{y}(\vec{X};\vec{\alpha}) = \vec{\alpha}^t\,\vec{y}(\vec{X})$. Hence,

$$\begin{aligned}\text{MSE}\left(\tilde{y}(\vec{X};\vec{\alpha})\right) &= \text{E}\left(\left(\tilde{\delta}(\vec{X};\vec{\alpha})\right)^2\right) = \text{E}\left(\left(t(\vec{X}) - \vec{\alpha}^t\,\vec{y}(\vec{X})\right)^2\right)\\ &= \text{E}\left(t^2(\vec{X}) - 2\,t(\vec{X})\,\vec{\alpha}^t\,\vec{y}(\vec{X}) + \left(\vec{\alpha}^t\,\vec{y}(\vec{X})\right)^2\right)\\ &= \text{E}\left(t^2(\vec{X})\right) - 2\,\vec{\alpha}^t\,\vec{\Theta} + \vec{\alpha}^t\,\Phi\,\vec{\alpha}\ ;\end{aligned}$$

where $\Phi = [\phi_{ij}] = \left[\text{E}\left(y_i(\vec{X})\,y_j(\vec{X})\right)\right]$ is a $p \times p$ matrix, and $\vec{\Theta} = [\theta_i] = \left[\text{E}\left(t(\vec{X})\,y_i(\vec{X})\right)\right]$ is a $p \times 1$ vector. Taking the derivative of the MSE with respect to $\vec{\alpha}$, then equating it with zero, leads to the optimal combination-weights vector $\vec{\alpha}^* = \Phi^{-1}\vec{\Theta}$, of Equation 1.

A.2 The Constrained Case

Under the condition that $\vec{\alpha}^t\,\vec{1} = 1$,

$$\tilde{\delta}(\vec{X};\vec{\alpha}) = t(\vec{X}) - \tilde{y}(\vec{X};\vec{\alpha}) = t(\vec{X})\,\vec{\alpha}^t\,\vec{1} - \vec{\alpha}^t\,\vec{y}(\vec{X}) = \vec{\alpha}^t\left(t(\vec{X})\,\vec{1} - \vec{y}(\vec{X})\right) = \vec{\alpha}^t\,\vec{\delta}(\vec{X})\,.$$

Hence,

$$\text{MSE}\left(\tilde{y}(\vec{X};\vec{\alpha})\right) = \text{E}\left(\left(\tilde{\delta}(\vec{X};\vec{\alpha})\right)^2\right) = \text{E}\left(\left(\vec{\alpha}^t\,\vec{\delta}(\vec{X})\right)^2\right) = \vec{\alpha}^t\,\Omega\,\vec{\alpha}\ ,$$

where $\Omega = [\omega_{ij}] = \left[\text{E}\left(\delta_i(\vec{X})\,\delta_j(\vec{X})\right)\right]$, is a $p \times p$ matrix.
Forming the Lagrangian function, $\mathcal{L} = \vec{\alpha}^t\,\Omega\,\vec{\alpha} + 2\,\lambda\,(\vec{\alpha}^t\,\vec{1} - 1)$, then differentiating $\mathcal{L}$ w.r.t. $\vec{\alpha}$, $\partial\mathcal{L}/\partial\vec{\alpha} = 2\,\Omega\,\vec{\alpha} + 2\,\lambda\vec{1} = 0$, yields $\vec{\alpha}^* = -\lambda\,\Omega^{-1}\,\vec{1}$.
Premultiplying by $\vec{1}^t$, and using the condition, $\vec{\alpha}^t\,\vec{1} = \vec{1}^t\,\vec{\alpha} = 1$, yields

$$\lambda = \frac{-1}{\vec{1}^t\,\Omega^{-1}\,\vec{1}} \qquad \text{and} \qquad \vec{\alpha}^* = \frac{\Omega^{-1}\,\vec{1}}{\vec{1}^t\,\Omega^{-1}\,\vec{1}} \quad \text{of Equation 2.}$$

ACKNOWLEDGEMENT

We thank the referees for their helpful comments.

The software used in training and testing the neural networks was partially developed while the first author was a Research Intern at the Sensor and System Development Center, Honeywell Inc., Minneapolis, MN.

References

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[2] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[3] C. W. J. Granger, "Combining forecasts — twenty years later," *Journal of Forecasting*, vol. 8, pp. 167–173, 1989.

[4] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *International Journal of Forecasting*, vol. 5, pp. 559–583, 1989.

[5] C. W. J. Granger and R. Ramanathan, "Improved methods of combining forecasts," *Journal of Forecasting*, vol. 3, pp. 197–204, 1984.

[6] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing* (R. J. Mammone, ed.), Chapman & Hall, 1993. Forthcoming.

[7] A. Namatame and Y. Kimata, "Improving the generalising capabilities of a back-propagation network," *The International Journal of Neural Networks Research & Applications*, vol. 1, no. 2, pp. 86–94, 1989.

[8] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[9] S. Hashem, *Optimal Linear Combinations of Neural Networks.* PhD thesis, School of Industrial Engineering, Purdue University, Dec. 1993. (Technical report SMS 94–4).