Bibliography

Barron, A. (1991), Universal approximation bounds for superpositions of a sigmoid function, Technical report, Univ. of Illinois- Urbana-Champaign.

Bishop, C. (1995), *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

Friedman, J. & Stuetzle, W. (1981), 'Projection pursuit regression', *Journal of the American Statistical Association* **76**, 817–823.

Hastie, T. & Tibshirani, R. (1990), *Generalized Additive Models*, Chapman and Hall.

Ripley, B. D. (1996), Pattern recognition and neural networks, Cambridge University Press.

1.1.2 Discussion

The *field* of neural networks is large, and there are many large and important conferences each year, which at their inception in the mid 1980's were devoted to various aspects of the model described in this chapter. The annual NIPS *Neural Information Processing Conference* in Denver, Colorado is one such venue. The mid 1990s versions of these conferences would be more aptly described as applied statistics than neural networks. Neural network models tend to be currently viewed as one of a number of flexible regression models.

There have been many ingenious modifications and restrictions to the neural network model to broaden its range of applications. Two examples are *bottle-neck* networks for nonlinear principal components (the inputs are also the outputs), and networks with duplicated weights to mimic autoregressive models.

While this field is strongly application driven, some interesting theory has been produced (Barron 1991). For example, neural networks are *universal approximators* — given enough parameters they can approximate smooth functions arbitrarily well.

This author has seen many examples of inappropriate use of neural network models. For example, their use is unlikely to be appropriate in small biostatistical binary regression problems, where the primary goal is to understand the effect of the inputs on the event represented by the binary response. The converse problem, of course, abounds as well. How often do we struggle to fit large traditional parametric models to huge datasets with our inappropriate software. When prediction is a goal, the *black box* neural network model will often deliver close to the best fit with very few tears.

There are many commercial software packages available for fitting neural networks, and many free packages as well. A SAS module exists, and functions for Splus contributed by Brian Ripley are available from the Statlib archive `http://lib.stat.cmu/S`. For statisticians interested in reading more about neural networks, I strongly recommend the books by Ripley and Bishop (Bishop 1995, Ripley 1996). Both of these synthesize the vast neural network literature, contain key references, and use a style and language familiar to statisticians.

- Batch algorithms typically use second derivative information as well as gradients.
- Online algorithms are useful for very large datasets, since changes are made continuously without having to cycle through the entire training data set. The name *online* refers to their use in dynamical systems which respond continuously to a changing environment.
- When the responses are categorical, a different criterion such as likelihood can be used for fitting, but the basic algorithm is the same.

Neural network models are typically over-parametrized, often with more parameters than observations. Furthermore, the parameters are intrinsically aliased — in fact, the model is perfectly symmetrical with respect to the hidden units. The symmetry is resolved by using random starting values, similar in flavor to the fitting of mixture models. Although the over-parametrization can be controlled somewhat by restricting the number of hidden units, other strategies for regularization are popular:

Early Stopping: Gradient descent will converge slowly to a local minimum, which in a saturated model will fit the training data perfectly. If an independent validation set is available, one can monitor the performance of the model on this validation set after each one or set of updates. The usual bias-variance tradeoffs will cause this validation error to eventually increase as training error approaches zero, and will provide an optimal place to stop.

Weight Decay: Similar to ridge regression, various proposals exist for shrinking the weights towards zero. As a simple example, consider adding a penalty of the form $\lambda(\sum \alpha_{kj}^2 + \sum \beta_{j\ell}^2)$ to the criterion R. This introduces a hyper-parameter λ, which shrinks between the unrestricted model ($\lambda = 0$) and the constant model ($\lambda = \infty$). This approach is similar in flavor to cubic smoothing splines (Hastie & Tibshirani 1990). A large basis of cubic splines is created, with a knot at each data point, but then the large number of coefficients is fit subject to rather stringent smoothness constraints of the derived function.

Model Averaging: Even when the model is not overfit, the criterion R has multiple minima, and the random starting weights will lead to one. One can restart the algorithm many times, and then combine the solutions in some way. Simple averaging works well; Ripley (1996) proposes an approximate form of Bayesian posterior averaging.

Suppose we use least-squares on a sample of *training data* to learn the parameters or *weights* in (1.1):

$$R(\alpha, \beta) = \sum_{i=1}^{N} \sum_{k=1}^{K} (y_k^i - \hat{y}_k^i)^2,$$

a criterion nonlinear in the parameters. Since we anticipate using gradient descent algorithms, we require derivatives. All the derivatives will be sums over the N observations, and we display only the ith component (denoted by superscript i.)

$$\begin{aligned}
\frac{\partial R^i}{\partial \beta_{kj}} &= -2(y_k^i - \hat{y}_k^i) f_k'(\beta_k^T \mathbf{z}^i) z_j^i \\
\frac{\partial R^i}{\partial \alpha_{j\ell}} &= -\sum_{k=1}^{K} 2(y_k^i - \hat{y}_k^i) f_k'(\beta_k^T \mathbf{z}^i) \beta_{kj} \sigma'(\alpha_j^T \mathbf{x}^i) x_\ell^i
\end{aligned}$$

Given these derivatives, a gradient update at the $r+1$st iteration has the form

$$\begin{aligned}
\beta_{kj}^{(r+1)} &\leftarrow \beta_{kj}^{(r)} - \gamma_r \sum_i \frac{\partial R^i}{\partial \beta_{kj}^{(r)}} \\
\alpha_{j\ell}^{(r+1)} &\leftarrow \alpha_{j\ell}^{(r)} - \gamma_r \sum_i \frac{\partial R^i}{\partial \alpha_{j\ell}^{(r)}}
\end{aligned}$$

where γ_r is the *learning rate* which can change with iteration number r. The standard initialization is to use *random* (Gaussian) starting values for the parameters.

Online learning refers to a similar gradient descent algorithm, but where we take a separate gradient step in response to each observation one at a time (as opposed to the *batch* mode as presented). The algorithm then has two distinct phases of operation in response to each new training pair $(\mathbf{x}^i, \mathbf{y}^i)$:

- the *feed-forward* phase in which $\mathbf{x}^i$ is filtered up through the network to produce a prediction $\hat{\mathbf{y}}^i$.
- the *backpropogation* phase in which the error $(\mathbf{y}^i - \hat{\mathbf{y}}^i)$ is filtered back and apportioned to each of the coefficients, which are modified in turn in a small way to reduce such a future error.

There are many different variants of this basic approach — some examples are:

– $f_k(\mathbf{v}) = e^{v_k}/\sum_\ell e^{v_\ell}$: inverse multiple logit, used for k-class classification. Note here that each f_k requires the entire vector of outputs $\mathbf{v}$.

Neural networks can have more than one hidden layer of units. *Multilayer perceptrons (MLP)* simply repeat the hidden layer several times, creating even more complex models. In this chapter we focus on the SLP.

Each hidden unit can be thought of as a nonlinear basis function which creates a new derived variable z_j from a linear combination of the inputs. The responses are then regressed on these transformed data z_j either linearly, or via logistic regression. When the model is *learned* or fit to the data, an optimal set of basis functions is learned at the same time.

Early *perceptron* models used *hard or binary thresholding* functions $\sigma(z) = 1(z > 0)$ at the hidden units, with strong neuro-physiological implications — the hidden unit (neuron) either fires or does not in response to its inputs. This biological interpretation is not taken too seriously today, and soft threshold functions such as the sigmoid permit differentiation and smoother learning-algorithms.

Versions of neural networks look like familiar statistical models:

- linear regression, when $\sigma(z) = z$, as long as $m \geq p$.
- logistic regression with 2 or more classes; again, $\sigma(z) = z$ and the appropriate versions of f_k are used.
- series and basis expansion models, where we regress the response on a few appropriately placed basis functions.

In addition, there are interesting comparisons with the *projection pursuit* model of Friedman & Stuetzle (1981):

$$\hat{y}_k = \sum_j f_{kj}(\alpha_{kj}^T \mathbf{x}).$$

where the functions f_j and the directions α_{kj} are fit simultaneously. The projection pursuit model typically invests many parameters on each function f_{kj} for a given direction α_{kj}, while the neural network can be seen to use exactly 2.

1.1.1 *Learning* Neural Network Models

The computational paradigm central to most neural network systems is *backpropogation*, which is essentially gradient descent using the chain rule plus a few bells and whistles.

in *italics*.

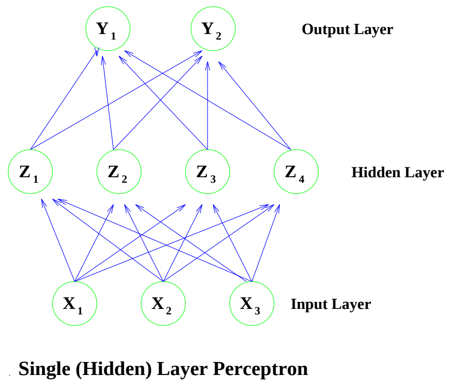

Figure 1.1: *A network diagram represents a neural network model with 3 inputs (predictors), 4 hidden units, and two outputs (responses).*
This configuration is often referred to as a *single layer perceptron.*

Figure 1.1 depicts a neural network with three predictors or *inputs*, a single hidden layer of four hidden units, and an output layer of two responses or *output* units.

If we denote the vector of inputs by $\mathbf{x}$, the outputs by $\mathbf{y}$, then this model can be written more traditionally as

$$\begin{aligned} z_j &= \sigma(a_{0j} + a_j^T \mathbf{x}), \ j = 1, \dots, m = 4 \\ \hat{y}_k &= f_k(\beta_{0k} + \beta_k^T \mathbf{z}), \ k = 1, \dots, q = 2) \end{aligned} \tag{1.1}$$

- The *activation function* σ is used to introduce a nonlinearity at the hidden layer, and is often taken to be the sigmoid $\sigma(z) = 1/(1+e^{-z})$.
- The parameters $\alpha_{j\ell}$ and β_{kj} are known as *weights*, and define linear combinations of the input vector $\mathbf{x}$ and hidden unit output vector $\mathbf{z}$ respectively.
- The intercepts α_{0j} and β_{0k} are known as *biases*.
- The function f_k permits a final transformation of the output, and the typical choices are:
  - $f_k(v) = v$: identity, suitable for regression with quantitative responses.
  - $f_k(v) = 1/(1 + \exp(-v))$: inverse logit, suitable when responses should lie in $[0, 1]$ (as in 2 class nonparametric logistic regression problems.)

Chapter 1

Computation

1.1 Neural Networks *by Trevor Hastie*

[1] Modern regression and classification techniques have seen a rapid development over the last 15 years. We hear of names such as smoothers, projection pursuit, additive models, CART, MARS, belief networks, and many more, each designed to accommodate some of the deficiencies of our more traditional linear-model-based regression techniques. Then along come *neural networks* in the early 80's with the flamboyant claim of being the ultimate *black-box* regression technique to supplant all the above. The name conjures up images of the computer mimicking the human brain, and the popular press soaks up the claims all too readily. To make matters even more offensive, neural networks are a product of Computer Science, not Statistics.

Statisticians rise to the defense, and denounce this intruder as reinventing the wheel, unprincipled, and unlikely to last the test of time. At the time of this writing neural networks have flourished for 13 years, and are used successfully in a large variety of applications including face recognition, handwriting and speech recognition, stock market prediction, medical diagnosis, system control, and genetic modelling.

While neural networks probably get more attention than they deserve in the scientific community at large, they in turn get less attention than they deserve from statisticians. In this chapter we give an overview of the technology, and attempt to place it in the broad context of flexible regression and classification. Along with these models comes a lot of new, redefined, and often attractive terminology, which we emphasize in the text

[1] To appear in *Encyclopedia of Biostatistics*, John Wiley.

Encyclopedia of Biostatistics

Trevor Hastie

November 15, 1996