CPH 636 — Spring 2009 — Dr. Charnigo

Lecture 9

Neural networks: regression problems

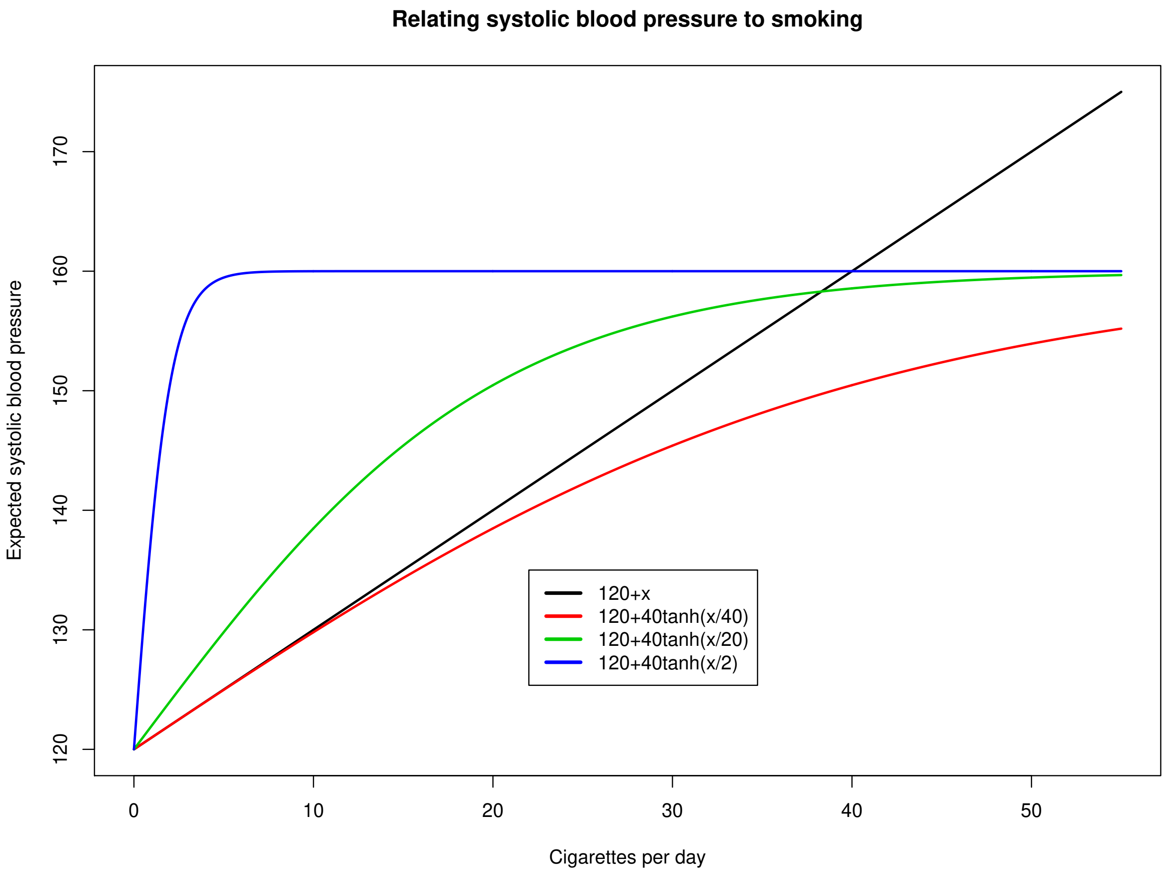
Scenario and motivation for neural networks. We have a continuous response variable Y and several explanatory variables $X_1, \ldots, X_k$ that we may assume are some mix of continuous and dichotomous. The linear regression model and the regression tree are often useful in this scenario, especially if we are concerned about the interpretability and transparency of our results.

As noted in Lecture 8, however, a difficulty with the linear regression model is that rather strong assumptions must be made. These are not just assumptions about the error terms but also assumptions about the mathematical form of the conditional expectation $E[Y|x_1, \ldots, x_k]$. A difficulty with the regression tree, on the other hand, is its sensitivity to the observed responses in the training data set. In particular, a small alteration to the observed responses in the training data set can change the initial split at the top of the tree, which can dramatically affect the rest of the tree.

Neural networks provide an alternative approach if we are concerned about our predictions being undermined by an incorrectly specified model or by excess sensitivity to the observed responses in the training data set. In fact, neural networks are regarded as being among the best tools for prediction, especially in "high noise" settings (i.e., the response variable exhibits considerable variation even when the explanatory variables are fixed).

Ideas behind neural networks: development with one explanatory variable. Suppose for now that we have a single continuous explanatory variable X. For concreteness, suppose that X is the number of cigarettes smoked per day and that Y is systolic blood pressure. Figure 1 shows four versions of a mean response function.

Figure 1:



Version 1 (in black) is a straight line, which assumes that the expected impact of smoking 20 cigarettes per day is exactly twice the expected impact of smoking 10 cigarettes per day, the expected impact of smoking 30 cigarettes per day is exactly three times the expected impact of smoking 10 cigarettes per day, and so forth. In particular, there is no limit to the expected impact of smoking: the mean response can be made arbitrarily large by continually increasing the number of cigarettes smoked per day.

But what if there should be a limit to the expected impact of smoking, say 40 mm Hg? Versions 2 (in red), 3 (in green), and 4 (in blue) respect this limit. Version 2 approaches the limit most slowly, while Version 4 approaches it most quickly. In fact, with Version 4 the limit is essentially attained at 5 cigarettes per day, so Version 4 dichotomizes smoking: nonsmokers have mean response 120 and smokers have mean response 160.

The mean response functions in Versions 2, 3, and 4 all have the form

$$120 + \beta_1 \tanh(\alpha_1 x),$$

where x is the number of cigarettes smoked per day and $\tanh(\cdot)$ is the hyperbolic tangent function defined by

$$\tanh(x) := \frac{\exp[2x] - 1}{\exp[2x] + 1}.$$

The number β_1 determines the limit to the expected impact of smoking, while the number α_1 determines how quickly that limit is approached.

One can show mathematically that

$$120 + \beta_1 \tanh(\alpha_1 x) \approx 120 + x\beta_1\alpha_1$$

as long as $\alpha_1 x$ is not too large. This is illustrated by Version 2, in which $\beta_1 = 40$ and $\alpha_1 = 1/40$, so that

$$120 + 40 \tanh(x/40) \approx 120 + x$$

until x is about 15. If $\beta_1 = 160$ and $\alpha_1 = 1/160$ (not shown in Figure 1), then

$$120 + 160 \tanh(x/160) \approx 120 + x$$

until x is about 60. In fact, Version 1 is essentially a limiting case of $120 + \beta_1 \tanh(\alpha_1 x)$ as α_1 decreases and β_1 increases with $\beta_1\alpha_1$ constrained to equal 1.

A more general form for the mean response function is

$$\beta_0 + \beta_1 \tanh(\alpha_0 + \alpha_1 x). \tag{1}$$

In my example above, I had fixed $\beta_0 := 120$ and $\alpha_0 := 0$. Relation (1) defines a single-layer one-hidden-node neural network when there is a single explanatory variable, once we specify how the actual responses differ from the expected responses. A common choice is to assume, as in linear regression, that these differences are independent and normally distributed with common but unknown variance.

A single-layer one-hidden-node neural network imposes an upper limit of $\beta_0+|\beta_1|$ on the mean response and a lower limit of $\beta_0-|\beta_1|$, uses α_1 to control how quickly the mean response changes with x, and uses α_0 to determine the x at which the mean response equals β_0. The upper and lower limits for the mean response may not be approachable if the range of x is restricted. In my example above, the mean response never falls below 120 — and never approaches the lower limit of 80 — because x is restricted to be nonnegative.

Discussion questions. Which of Versions 2, 3, and 4 can be well approximated by a quadratic function over the range of x displayed in Figure 1? Generally speaking, what kinds of mean responses (in one explanatory variable) can be described by a single-layer one-hidden-node neural network but not a quadratic function? What kinds of mean responses can be described by a quadratic function but not a single-layer one-hidden-node neural network?

A statistical model for neural networks with multiple explanatory variables. Now we allow for multiple explanatory variables $X_1, \ldots, X_k$ that can be any mix of continuous and dichotomous. To adhere to Enterprise Miner conventions, we will assume that the continuous explanatory variables have been standardized and that the dichotomous explanatory variables have been coded with values $+1$ and -1. Enterprise Miner performs these operations automatically, so we do not need to standardize continuous explanatory variables or recode dichotomous explanatory variables before using Enterprise Miner.

The statistical model for a single-layer neural network is as follows. For a given positive integer j, we define $H_1, H_2, \ldots, H_j$ by

$$H_1 := \tanh(\alpha_{0,1} + \alpha_{1,1}X_1 + \alpha_{2,1}X_2 + \cdots + \alpha_{k,1}X_k),$$

$$H_2 := \tanh(\alpha_{0,2} + \alpha_{1,2}X_1 + \alpha_{2,2}X_2 + \cdots + \alpha_{k,2}X_k), \cdots,$$

$$H_j := \tanh(\alpha_{0,j} + \alpha_{1,j}X_1 + \alpha_{2,j}X_2 + \cdots + \alpha_{k,j}X_k).$$

We refer to $H_1, H_2, \ldots, H_j$ as "hidden nodes", "hidden units", or "hidden neurons". We refer to the intercept-like parameters $\alpha_{0,1}, \alpha_{0,2}, \ldots, \alpha_{0,j}$ as "bias" terms because they represent contributions to $H_1, H_2, \ldots, H_j$ that do not arise from $X_1, \ldots, X_k$. This use of the word "bias" is not related to the bias/variance tradeoff discussed in Lecture 2.

We finish by expressing the mean response as

$$\beta_0 + \beta_1 H_1 + \cdots + \beta_j H_j$$

and specifying how the actual responses differ from the expected responses. Again, a common choice is to assume that these differences are independent and normally distributed with common but unknown variance.

What I presented in (1) was the special case with $k = 1$ and $j = 1$.

Discussion question. Let us try to better understand the role of j. Suppose for simplicity that $k = 1$. What does taking $j > 1$ allow that taking $j = 1$ did not?

Further remarks on neural networks. There is no universal rule for deciding how many hidden nodes to include. Since more hidden nodes entail more parameters in our statistical model, we are inclined to include more hidden nodes when the number of explanatory variables is large, the size of the training data set is large, and the noise level is low. Conversely, we are inclined to include fewer hidden nodes when the number of explanatory variables is modest, the size of the training data set is modest, and the noise level is high. Enterprise Miner will suggest how many hidden nodes to include based on a user's declaration that the noise level is low, moderate, or high. You can accept Enterprise Miner's suggestion or set the number of hidden nodes yourself. [[**Discussion question:** Supposing for simplicity that $k = 1$, explain why we may be more comfortable having more parameters in our statistical model when the noise level is low than when the noise level is high.]]

What I have described above is called a single-layer neural network because there is only one layer of intervening quantities between $X_1, \ldots, X_k$ and Y. If I had prescribed that Y should be defined in terms of $S_1, S_2, \ldots, S_{j_2}$, that $S_1, S_2, \ldots, S_{j_2}$ should be defined in terms of $F_1, F_2, \ldots, F_{j_1}$, and that $F_1, F_2, \ldots, F_{j_1}$ should be defined in terms of $X_1, \ldots, X_k$, then I would have had a two-layer neural network. A three-layer neural network, a four-layer neural network, etc., would be defined in the obvious manner. In practice, most users employ a single-layer neural network.

Estimation of neural network parameters. The neural network parameters are estimated iteratively. This means that Enterprise Miner makes initial estimates, revises the initial estimates, revises the revised estimates, and so forth. With each iteration, the estimates approach numbers that minimize the average squared error on the training data set. The iterations continue until the average squared error on the training data set is so close to its minimum that further iterations are of no practical worth.

Iterative estimation is the norm rather than the exception for nonlinear statistical models. Even when we speak of maximum likelihood estimation in logistic regression, the maximum likelihood estimates are obtained iteratively. [[The maximum likelihood estimates in linear regression can be obtained in closed form — i.e., without iterations — because the partial derivatives of the log likelihood are linear in the parameters being estimated, yielding a system of linear equations for the parameter estimates that can be solved with a matrix inversion. Obviously, the same is not true for logistic regression.]]

As with a regression tree, having the smallest possible average squared error on the training data may diminish our ability to predict the response in other data sets. With a regression tree, we addressed this issue by "pruning" to get an intermediate version of the tree based on fewer than the maximum number of leaves. In particular, we sought an intermediate version of the tree that minimized average squared error on the validation data.

With a neural network, we address this issue by adopting an intermediate version of the neural network based on fewer than the maximum number of iterations. In particular, we seek an intermediate version of the neural network that minimizes average squared error on the validation data.

Illustrative example. Refer to {em_report.html} in the {NeuralNet} folder on my home page. I fit a single-layer one-hidden-node neural network to the diabetes data set from Written Assignment 1, taking GLU as the response variable and NPREG, BMI, BP, AGE as explanatory variables.

The optimization plot shows that the average squared error on the training data decreased rapidly until about the eighth iteration and then held fairly steady, although 23 iterations were required for the average squared error to stabilize enough for iterations to stop. On the other hand, the average squared error on the validation data was minimized at the fifth iteration. The version of the neural network with which predictions will be made is the one corresponding to the fifth iteration.

Looking at the "Fit Statistic" section, we see that the average squared errors are 826.79 on the training data, 539.57 on the validation data, and 1451.34 on the test data. Taking the square root of 1451.34, we conclude that a typical prediction error on the test data has magnitude 38.1 mm Hg, which suggests that NPREG, BMI, BP, AGE are not too useful in predicting GLU (at least in this population). Because 1451.34 is so much larger than 539.57, there is also a hint that a few unusual observations in the test data may be driving up the average squared error. In any case, the 1451.34 is not an artifact of the neural network. One actually fares slightly worse with a regression tree (1474.23).

Clicking on "Datastep Score Code" displays code that reveals all of the parameter estimates and that can be used to make predictions on GLU for specific individuals in any data set with NPREG, BMI, BP, AGE.

Neural networks: classification problems

Scenario and motivation for neural networks. We have a categorical response variable Y and several explanatory variables $X_1, \ldots, X_k$ that we may assume are some mix of continuous and dichotomous. Just as we may have misgivings about using a linear regression model or a regression tree in some regression problems, we may have misgivings about using a logistic regression model or a classification tree in some classification problems. Once again, neural networks provide an alternative approach if we are concerned about our predictions being undermined by an incorrectly specified model or by excess sensitivity to the observed responses in the training data set.

A statistical model. As before, we will assume that the continuous explanatory variables have been standardized and that the dichotomous explanatory variables have been coded with values $+1$ and -1. Moreover, we will assume that the categories of Y are designated $0, 1, \ldots, C-1$, where C is the number of categories.

With a categorical response variable, the statistical model for a single-layer neural network is as follows. First, for a given positive integer j, we define $H_1, H_2, \ldots, H_j$ by

$$H_1 := \tanh(\alpha_{0,1} + \alpha_{1,1}X_1 + \alpha_{2,1}X_2 + \cdots + \alpha_{k,1}X_k),$$

$$H_2 := \tanh(\alpha_{0,2} + \alpha_{1,2}X_1 + \alpha_{2,2}X_2 + \cdots + \alpha_{k,2}X_k), \cdots,$$

$$H_j := \tanh(\alpha_{0,j} + \alpha_{1,j}X_1 + \alpha_{2,j}X_2 + \cdots + \alpha_{k,j}X_k).$$

This is identical to what we did before, when we had a continuous response variable.

Next, and here there is a difference from what we did before, we put

$$T_0 := 0, \quad T_1 := \beta_{0,1} + \beta_{1,1}H_1 + \beta_{2,1}H_2 + \cdots + \beta_{j,1}H_j,$$

$$T_2 := \beta_{0,2} + \beta_{1,2}H_1 + \beta_{2,2}H_2 + \cdots + \beta_{j,2}H_j, \cdots,$$

$$T_{C-1} := \beta_{0,C-1} + \beta_{1,C-1}H_1 + \beta_{2,C-1}H_2 + \cdots + \beta_{j,C-1}H_j.$$

Finally, we postulate that

$$P[Y = 0|X_1, \ldots, X_k] = \frac{\exp[T_0]}{\exp[T_0] + \exp[T_1] + \cdots + \exp[T_{C-1}]},$$

$$P[Y = 1|X_1, \ldots, X_k] = \frac{\exp[T_1]}{\exp[T_0] + \exp[T_1] + \cdots + \exp[T_{C-1}]}, \cdots,$$

$$P[Y = C-1|X_1, \ldots, X_k] = \frac{\exp[T_{C-1}]}{\exp[T_0] + \exp[T_1] + \cdots + \exp[T_{C-1}]}.$$

If $P[Y = 0|X_{1,i}, \ldots, X_{k,i}]$ is larger than any of the other conditional probabilities, we predict that $Y_i = 0$; if $P[Y = 1|X_{1,i}, \ldots, X_{k,i}]$ is larger than any of the other conditional probabilities, we predict that $Y_i = 1$; and so forth. Actually, since all of the conditional probabilities have the same denominator, and because the exponential function is strictly increasing, our prediction is really determined by which of $T_{0,i}, T_{1,i}, \ldots, T_{C-1,i}$ is largest.

Estimation of neural network parameters. As before, the neural network parameters are estimated iteratively. However, instead of reducing average squared error on the training data, Enterprise Miner now reduces the "Multiple Bernoulli" error on the training data.

[[The Multiple Bernoulli error is

$$-2\sum_{i=1}^{n}\{1_{Y_i=0}\log P_0(\mathbf{X}_i) + 1_{Y_i=1}\log P_1(\mathbf{X}_i) + \cdots + 1_{Y_i=C-1}\log P_{C-1}(\mathbf{X}_i)\},$$

where 1_A is an indicator that equals 1 only if A is true and $P_q(\mathbf{X}_i)$ is shorthand for the estimate of $P[Y = q|X_{1,i}, \ldots, X_{k,i}]$ induced by the parameter estimates. **Discussion question**: What is the rationale for reducing the

Multiple Bernoulli error on the training data?]]

We adopt an intermediate version of the neural network based on fewer than the maximum number of iterations. In particular, we seek an intermediate version of the neural network that minimizes the misclassification rate on the validation data.

Illustrative example. Refer to {em_report.html} in the {NeuralNet2} folder on my home page. We continue with the diabetes data set from Written Assignment 1, but this time GLU is among the explanatory variables and DIAB is the response variable.

The optimization plot shows that the Multiple Bernoulli error on the training data decreased rapidly until about the twelfth iteration and then held fairly steady, although 29 iterations were required for the Multiple Bernoulli error to stabilize enough for iterations to stop. On the other hand, the misclassification rate on the validation data was minimized at the third iteration. The version of the neural network with which we will make predictions is the one corresponding to the third iteration.

Looking at the "Fit Statistic" section, we see that the misclassification rates were 0.240 on the training data, 0.200 on the validation data, and 0.240 on the test data.

Clicking on "Datastep Score Code" displays code that reveals all of the parameter estimates and that can be used to make predictions on DIAB for specific individuals in any data set with NPREG, BMI, BP, AGE, GLU.