Lecture Outline

Reading: Chapter 5 Neural Networks

This lecture covers

- Introduction
- Feedforward functions (5.1)

Introduction

Introductory remarks on *neural networks*:

- The linear models for regression and classification (Chapters 3 and 4) are based on linear combinations of *fixed* nonlinear basis functions. Although these models have useful analytical and computational properties, their practicality is limited due to the *curse of dimensionality*. In order to apply such models to large-scale problems, it is necessary to adapt the basis functions to the data.
- The idea in the *feed-forward neural network* (NN), is to *fix* the number of basis functions but allow them to be *adaptive*. This requires a *parametric* form of the basis function in which the parameters are adapted during training.
- For many applications, the resulting model can be significantly more compact, and hence faster to compute. The price paid for this compactness is that the likelihood function, which forms the basis for network training, is no longer a convex function of the model parameters and hence may require considerable computational resources during the training phase.
- The problem of determining the network parameters within the maximum likelihood framework involves the solution to a nonlinear optimization problem. This requires evaluation of derivatives of the log likelihood function with respect to the network parameters. However, we shall see that these can be efficiently obtained using a technique known as *error backpropagation*.

5.1 Feedforward Network Functions

The linear models for regression and classification (Chapters 3 and 4) are based on linear combinations of fixed, nonlinear basis functions $\phi_j(\mathbf{x})$ and take the form

$$\textbf{Regression}: \quad y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \tag{1}$$

$$\textbf{Classification}: \quad y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) \right) = f\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \right) \tag{2}$$

where $f(\cdot)$ is a nonlinear activation function so that the output is in the range $[0, 1]$, e.g. logistic sigmoid, softmax.

Goal: Extend the fixed-basis, linear model by making the basis functions $\phi_j(\mathbf{x})$ depend on parameters and then allow these parameters to be adjusted, along with the coefficients $\{w_j\}$, during training.

Neural networks use basis functions that follow the same form as (2), so that each basis function is itself a nonlinear function of a linear combination of the inputs, where the coefficients in the linear combination are adaptive parameters.

This leads to the basic *neural network* model, which can be described by a series of functional transformations:

First, we construct M linear combinations of the input variables $x_1, \ldots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad j = 1, \ldots, M \tag{3}$$

where the superscript $^{(1)}$ indicates that the corresponding parameters are in the first 'layer' of the network. We shall refer to the parameters $w_{ji}^{(1)}$ as *weights* and the parameters $w_{j0}^{(1)}$ as *biases*. The quantities a_j are known as *activations* and for each j, is a linear combination of the input variables.

The activations are then transformed using a differentiable, nonlinear *activation function* $h(\cdot)$ to give

$$z_j = h(a_j). \tag{4}$$

These quantities are called *hidden units* and correspond to the outputs of the basis functions, $\phi_j(\mathbf{x})$ in the linear model. These are clearly nonlinear, parametric functions of the input. The functions $h(\cdot)$ are generally chosen to be sigmoidal functions such as the logistic sigmoid or the 'tanh' function,

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{\sinh(x)}{\cosh(x)} \tag{5}$$

or the Rectified Linear Unit (ReLU), $h(x) = \max(0, x)$.

Figure 1: nonlinear activation functions

Second, the hidden units are linearly combined (as in the linear model) to give *output unit activations*

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \quad k = 1, \ldots, K \tag{6}$$

where $k = 1, \ldots, K$ and K is the total number of outputs. In a classification application, K is the number of classes and for a regression application, K is the dimension of the regressor. This transformation corresponds to the second layer of the network and again $w_{k0}^{(2)}$ are bias parameters. [Note: I believe better notation would be $a_j^{(1)}$ for (3) and $a_k^{(2)}$ for (6)].

Finally, the output unit activations are transformed using an appropriate activation function:

Regression: Identity, $\sigma(a) = a$

Binary Classification: Logistic signmoid, $\sigma(a) = \dfrac{1}{1 + \exp(-a)}$

Multiclass Classification: Softmax, $\sigma(a_k) = \dfrac{\exp(a_k)}{\sum_j \exp(a_j)}$

We can combine these various stages to give the overall network function that, for sigmoidal output unit activation functions, takes the form

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{7}$$

where the set of weight and bias parameters have been grouped together to denote $\mathbf{w}$ on the left. Thus the neural network model is simply a nonlinear function from a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ controlled by a vector $\mathbf{w}$ of adjustable parameters.

This function can be represented in the form of a network diagram as shown in Fig. 2. The process of evaluating (7) (as in the test stage) can be interpreted as a *forward propagation* of information through the network.

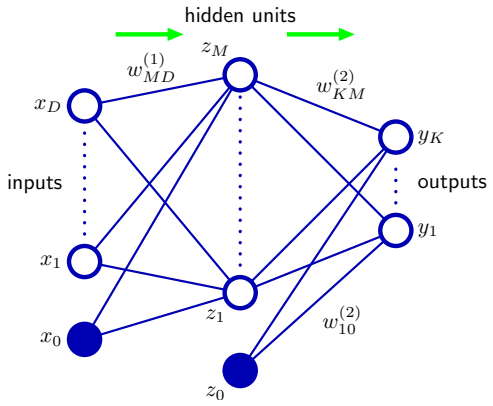

Figure 2:

The bias parameters in (3) and (6) can be absorbed into a set of weight parameters by defining an additional input variable $x_0 = 1$ and $z_0 = 1$ resulting in

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i \quad \text{and} \quad a_k = \sum_{j=0}^{M} w_{ji}^{(2)} z_j. \tag{8}$$

There the overall network function is simplified to

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right). \tag{9}$$

Notes:

- The neural network is also known as the *multilayer perceptron* or MLP. A key difference compared to the perceptron, however, is that the neural network uses continuous sigmoidal nonlinearities in the hidden units, whereas the perceptron uses step-function nonlinearities. This means that the neural network function is differentiable with respect to the network parameters, and this will play a central role in network training.
- There is some confusion in the literature regarding the terminology for counting the number of layers in the network. We recommend a terminology in which Fig. 2 is called a two-layer network, because it is the number of layers of adaptive weights that is important for determining the network properties.

- Neural networks are said to be *universal approximators*. For example, a two-layer network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units (see text Fig. 5.3).

Figure 3: Text Figure 5.3

The key problem is how to find suitable parameter values given a set of training data. There exist (as we will see) effective solutions to this problem based on both maximum likelihood and Bayesian approaches.