Neural Networks

Until now, we focused on linear learning techniques. Even when we appended non-linear features, the weights still remained linear. We will now discuss a natively non-linear learning technique called neural networks. In other words, this technique will be non-linear both in terms of features and weights. Despite the underlying non-convexity, neural networks perform remarkably well and often outperform the linear techniques discussed until now.

Feedforward Neural Network

In the class, we focused on the feedforward neural network with a single hidden layer. Here we consider a slightly more general case in which we have d features, L hidden layers, and a single output at the output layer. We will use the following notation in this note.

- Input features: $\boldsymbol{x} \in \mathbb{R}^d$.
- Feature vector that includes the bias feature: $\mathbf{x} = \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} \in \mathbb{R}^{d+1}$.
- Number of neurons of the l^{th} hidden layer is $n(l)$.
- Activation computed at the k^{th} node of the l^{th} hidden layer: $a_k^{(l)}$.
- Response of the k^{th} node of the l^{th} hidden layer: $h_k^{(l)} = g(a_k^{(l)})$.
- Weight from the j^{th} node of the $(l-1)^{th}$ hidden layer to the k^{th} node of the l^{th} hidden layer: $\beta_{kj}^{(l-1)}$.
- Weight vector from the $(l-1)^{th}$ hidden layer to the k^{th} node of the l^{th} hidden layer: $\boldsymbol{\beta}_k^{(l-1)} = \begin{bmatrix} \beta_{k0}^{(l-1)} \\ \beta_{k1}^{(l-1)} \\ \dots \\ \beta_{kn(l-1)}^{(l-1)} \end{bmatrix} \in \mathbb{R}^{n(l-1)+1}$.

- Weight matrix from the $(l-1)^{th}$ hidden layer to the l^{th} hidden layer:

$$\boldsymbol{\beta}^{(l-1)} = \begin{bmatrix} \boldsymbol{\beta}_{\mathbf{1}}^{(l-1)^T} \\ \boldsymbol{\beta}_{\mathbf{2}}^{(l-1)^T} \\ \vdots \\ \boldsymbol{\beta}_{n(l)}^{(l-1)^T} \end{bmatrix}^T = \begin{bmatrix} \beta_{10}^{(l-1)} & \beta_{20}^{(l-1)} & \cdots & \beta_{n(l)0}^{(l-1)} \\ \beta_{11}^{(l-1)} & \beta_{21}^{(l-1)} & \cdots & \beta_{n(l)1}^{(l-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{1n(l-1)}^{(l-1)} & \beta_{2n(l-1)}^{(l-1)} & \cdots & \beta_{n(l)n(l-1)}^{(l-1)} \end{bmatrix}. \tag{8.1}$$

- Activation of the output node: a_o. Response of the output node: $\hat{y} = g(a_o)$.

- Weight vector from the L^{th} hidden layer to the output node: $\boldsymbol{\beta}^{(L)} = \boldsymbol{\beta}_o = \begin{bmatrix} \beta_{o0} \\ \beta_{o1} \\ \cdots \\ \beta_{on(L)} \end{bmatrix} \in \mathbb{R}^{n(L)+1}$.

- Clearly, the vector of activations at the l^{th} hidden layer is:

$$\mathbf{a}^{(l)} = \boldsymbol{\beta}^{(l-1)^T} \bar{\mathbf{h}}^{(l-1)} = \begin{bmatrix} a_1^{(l)} \\ a_2^{(l)} \\ \vdots \\ a_{n(l)}^{(l)} \end{bmatrix}, \text{where } \bar{\mathbf{h}}^{(l-1)} = \begin{bmatrix} h_0^{(l-1)} \\ h_1^{(l-1)} \\ \vdots \\ h_{n(l-1)}^{(l-1)} \end{bmatrix}. \tag{8.2}$$

- Similar to the activations, the vector of the responses at the l^{th} hidden layer can be expressed as:

$$\mathbf{h}^{(l)} = \mathbf{g}\left(\mathbf{a}^{(l)}\right) = \begin{bmatrix} g(a_1^{(l)}) \\ g(a_2^{(l)}) \\ \vdots \\ g(a_{n(l)}^{(l)}) \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{\beta}_1^{(l-1)^T} \bar{\mathbf{h}}^{(l-1)}) \\ g(\boldsymbol{\beta}_2^{(l-1)^T} \bar{\mathbf{h}}^{(l-1)}) \\ \vdots \\ g(\boldsymbol{\beta}_{n(l)}^{(l-1)^T} \bar{\mathbf{h}}^{(l-1)}) \end{bmatrix} \tag{8.3}$$

- The vector of the responses at the l^{th} hidden layer including the bias response can be expressed as:

$$\bar{\mathbf{h}}^{(l)} = \begin{bmatrix} h_0^{(l)} \\ h_1^{(l)} \\ \vdots \\ h_{n(l)}^{(l)} \end{bmatrix} = \begin{bmatrix} h_0^{(l)} \\ \\ \mathbf{h}^{(l)} \end{bmatrix}. \tag{8.4}$$

Neural Network Training: Backpropagation

In this section, we will describe how a neural network is trained. We consider a generic loss function $L(\boldsymbol{\theta})$ in our derivation, where $\boldsymbol{\theta}$ is a vector which contains all the weights for all

layers. In particular, $L(\boldsymbol{\theta})$ can be expressed as

$$L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L^{(i)}(\boldsymbol{\theta}),$$

where $L^{(i)}(\boldsymbol{\theta})$ denotes the value of the loss function corresponding to the i^{th} training example. Now, the objective is to compute the derivatives of the loss function with respect to the weight parameters. Due to the structure of the loss function, it suffices to compute the derivative for a single training example. The derivative of the loss function will simply be the average of derivatives for all the training examples. Therefore, we will first focus on the i^{th} training example. The goal is to compute $\frac{\partial L^{(i)}}{\partial \boldsymbol{\beta}_o}$ and $\{\frac{\partial L^{(i)}}{\partial \boldsymbol{\beta}^{(l)}}\}$, $l \in \{0, \cdots, L-1\}$. These derivatives can be computed in the following reverse order (to apply chain rule):

$$\mathbf{x} \to \boldsymbol{\beta}^{(0)} \to \mathbf{a}^{(1)} \to \mathbf{h}^{(1)} \to \boldsymbol{\beta}^{(1)} \to \cdots \to \boldsymbol{h}^{(L-1)} \to \boldsymbol{\beta}^{(L-1)} \to \mathbf{a}^{(L)} \to \mathbf{h}^{(L)} \to \boldsymbol{\beta}_o \to a_o \to \hat{y}_i = g(a_o),$$

where the above reverse order also leads to the name *back propagation*.

Now, we discuss how to compute the derivative of $L^{(i)}$ with respect to the weights from the $(t-1)^{th}$ layer to the t^{th} layer, i.e., the weights in $\boldsymbol{\beta}^{(t-1)}$. Clearly, the output of the neurons of the t^{th} layer is $\mathbf{h}^{(t)} = \mathbf{g}(\mathbf{a}^{(t)})$, where $\mathbf{a}^{(t)} = \boldsymbol{\beta}^{(t-1)^T}\bar{\mathbf{h}}^{(t-1)}$. Note that the j^{th} element of $\mathbf{h}^{(t)}$ is $h_j^{(t)} = g(a_j^{(t)})$. Hence, $L^{(i)}(\mathbf{h}^{(t)})$ can be expressed as

$$L^{(i)}(\mathbf{h}^{(t)}) = L^{(i)}\left(\mathbf{g}\left(\mathbf{a}^{(t)}\right)\right) = L^{(i)}\left(\mathbf{g}\left(\boldsymbol{\beta}^{(t-1)^T}\bar{\mathbf{h}}^{(t-1)}\right)\right). \tag{8.5}$$

Let $\mathbf{w}^{(t-1)}$ be the column vector obtained by concatenating the columns of $\boldsymbol{\beta}^{(t-1)}$. Also, define by $\mathbf{H}^{(t-1)}$ the $n(t) \times [n(t)\,(n(t-1)+1)]$ matrix:

$$\mathbf{H}^{(t-1)} = \begin{bmatrix} \bar{\mathbf{h}}^{(t-1)^T} & 0 & \dots & 0 \\ 0 & \bar{\mathbf{h}}^{(t-1)^T} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{\mathbf{h}}^{(t-1)^T} \end{bmatrix}.$$

Since $\boldsymbol{\beta}^{(t-1)^T}\bar{\mathbf{h}}^{(t-1)} = \mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}$, (8.5) can be rewritten as a function of $\mathbf{w}^{(t-1)}$ as

$$L^{(i)}(\mathbf{h}^{(t)}) = L^{(i)}\left(\mathbf{g}\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)\right). \tag{8.6}$$

By applying chain rule to (8.6), we obtain

$$\begin{aligned} \frac{\partial L^{(i)}}{\partial \mathbf{w}^{(t-1)}} &= \frac{\partial L^{(i)}}{\partial\left(\mathbf{g}\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)\right)} \times \frac{\partial \mathbf{g}\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)}{\partial\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)} \times \frac{\partial\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)}{\partial \mathbf{w}^{(t-1)}}, \\ &\overset{(a)}{=} \frac{\partial L^{(i)}}{\partial \mathbf{h}^{(t)}} \times \frac{\partial \mathbf{g}\left(\mathbf{a}^{(t)}\right)}{\partial \mathbf{a}^{(t)}} \times \mathbf{H}^{(t-1)}, \\ &\overset{(b)}{=} \frac{\partial L^{(i)}}{\partial \mathbf{h}^{(t)}} \times \mathrm{diag}\left(\mathbf{g}'\left(\mathbf{a}^{(t)}\right)\right) \times \mathbf{H}^{(t-1)}, \end{aligned} \tag{8.7}$$

where step (a) follows from $\mathbf{h}^{(t)} = \mathbf{g}\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)$, $\mathbf{a}^{(t)} = \mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}$, and the fact that $\frac{\partial\left(\mathbf{H}^{(t-1)}\mathbf{w}^{(t-1)}\right)}{\partial \mathbf{w}^{(t-1)}} = \mathbf{H}^{(t-1)}$. Step (b) follows since the derivative of $g(a_i^{(t)})$ with respect to $a_j^{(t)}$, $j \neq i$, is zero. Hence, we have

$$\frac{\partial \mathbf{g}\left(\mathbf{a}^{(t)}\right)}{\partial \mathbf{a}^{(t)}} = \begin{bmatrix} \frac{\partial g(a_1^{(t)})}{\partial a_1^{(t)}} & \frac{\partial g(a_1^{(t)})}{\partial a_2^{(t)}} & \cdots & \frac{\partial g(a_1^{(t)})}{\partial a_{n(t)}^{(t)}} \\ \frac{\partial g(a_2^{(t)})}{\partial a_1^{(t)}} & \frac{\partial g(a_2^{(t)})}{\partial a_2^{(t)}} & \cdots & \frac{\partial g(a_2^{(t)})}{\partial a_{n(t)}^{(t)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g(a_{n(t)}^{(t)})}{\partial a_1^{(t)}} & \frac{\partial g(a_{n(t)}^{(t)})}{\partial a_2^{(t)}} & \cdots & \frac{\partial g(a_{n(t)}^{(t)})}{\partial a_{n(t)}^{(t)}} \end{bmatrix} = \begin{bmatrix} g'(a_1^{(t)}) & 0 & \ldots & 0 \\ 0 & g'(a_2^{(t)}) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & g'(a_{n(t)}^{(t)}) \end{bmatrix}$$
$$= \operatorname{diag}\left(\mathbf{g}'\left(\mathbf{a}^{(t)}\right)\right),$$

where $\mathbf{g}'(\mathbf{a}^{(t)})$ is given by

$$\mathbf{g}'(\mathbf{a}^{(t)}) = \begin{bmatrix} g'(a_1^{(t)}) \\ g'(a_2^{(t)}) \\ \vdots \\ g'(a_{n(t)}^{(t)}) \end{bmatrix}.$$

By defining $\boldsymbol{\delta}^{(t)} = \frac{\partial L^{(i)}}{\partial \mathbf{h}^{(t)}}$, (8.7) can be further rewritten as

$$\frac{\partial L^{(i)}}{\partial \mathbf{w}^{(t-1)}} = \boldsymbol{\delta}^{(t)} \times \operatorname{diag}\left(\mathbf{g}'\left(\mathbf{a}^{(t)}\right)\right) \times \mathbf{H}^{(t-1)}. \tag{8.8}$$

From (8.8), we note that in order to compute $\frac{\partial L^{(i)}}{\partial \mathbf{w}^{(t-1)}}$, we just need to evaluate $\boldsymbol{\delta}^{(t)}$. Clearly, we have

$$L^{(i)}\left(\mathbf{h}^{(t)}\right) = L^{(i)}\left(\mathbf{h}^{(t+1)}\right) = L^{(i)}\left(\mathbf{g}\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)\right). \tag{8.9}$$

Hence, by applying chain rule to (8.9), we get

$$\begin{bmatrix} \frac{\partial L^{(i)}}{\partial h_0^{(t)}} \\ \boldsymbol{\delta}^{(t)^T} \end{bmatrix}^T = \begin{bmatrix} \frac{\partial L^{(i)}}{\partial h_0^{(t)}} \\ \left(\frac{\partial L^{(i)}}{\partial \mathbf{h}^{(t)}}\right)^T \end{bmatrix}^T = \frac{\partial L^{(i)}}{\partial \bar{\mathbf{h}}^{(t)}} = \frac{\partial L^{(i)}}{\partial\left(\mathbf{g}\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)\right)} \times \frac{\partial \mathbf{g}\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)}{\partial\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)} \times \frac{\partial\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)}{\partial \bar{\mathbf{h}}^{(t)}},$$
$$= \frac{\partial L^{(i)}}{\partial\left(\mathbf{g}\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)\right)} \times \operatorname{diag}\left(\mathbf{g}'\left(\boldsymbol{\beta}^{(t)^T}\bar{\mathbf{h}}^{(t)}\right)\right) \times \boldsymbol{\beta}^{(t)^T},$$
$$= \frac{\partial L^{(i)}}{\partial \mathbf{h}^{(t+1)}} \times \operatorname{diag}\left(\mathbf{g}'\left(\mathbf{a}^{(t+1)}\right)\right) \times \boldsymbol{\beta}^{(t)^T},$$
$$= \boldsymbol{\delta}^{(t+1)} \times \operatorname{diag}\left(\mathbf{g}'\left(\mathbf{a}^{(t+1)}\right)\right) \times \boldsymbol{\beta}^{(t)^T}. \tag{8.10}$$

From (8.10), we notice that $\{\boldsymbol{\delta}^{(t)}\}$ can be obtained in a recursive manner. In particular, we can first evaluate $\boldsymbol{\delta}^{(L+1)}$. Then, using $\boldsymbol{\delta}^{(L+1)}$, the vectors $\{\boldsymbol{\delta}^{(t)}\}$, $t \in \{1, 2, \cdots, L\}$, can be

Algorithm 1 Back propagation algorithm

Given: a training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$ and a learning rate α.
Initialize the learning parameters $\{\boldsymbol{\beta}^{(t)}\}$ and construct the weight vectors $\{\mathbf{w}^{(t)}\}$.
Repeat:
  Set $\boldsymbol{\psi}^{(t)} = \mathbf{0}$, $t \in \{0, 1, \cdots, L\}$.
  Set $i = 1$.
  Repeat:
    Set $\mathbf{x} = \begin{bmatrix} 1 \\ \boldsymbol{x}_i \end{bmatrix}$.
    Perform forward propagation to compute $\{\mathbf{a}^{(t)}\}, \{\mathbf{h}^{(t)}\}, a_o$, and $\hat{y} = g(a_o)$.
    Compute $\{\boldsymbol{\delta}^{(t)}\}$ using (8.10).
    Compute $\{\frac{\partial L^{(i)}}{\partial \mathbf{w}^{(t-1)}}\}$ using (8.8).
    $\boldsymbol{\psi}^{(t-1)} := \boldsymbol{\psi}^{(t-1)} + \frac{\partial L^{(i)}}{\partial \mathbf{w}^{(t-1)}}$, $t \in \{1, 2, \cdots, L+1\}$.
    $i := i + 1$.
  Until $i > n$.
  $\mathbf{w}^{(t)} := \mathbf{w}^{(t)} - \frac{\alpha}{n}\boldsymbol{\psi}^{(t)}$, $t \in \{0, 1, \cdots, L\}$.
  Construct $\{\boldsymbol{\beta}^{(t)}\}$ using $\{\mathbf{w}^{(t)}\}$.
Until the termination condition of the gradient descent algorithm is met.

evaluated recursively using (8.10). Afterwards, the derivative of the loss function with respect to the weights in each layer can be computed using (8.8). The steps of the backpropagation algorithm are summarized in Algorithm 1.

The choice of the loss function depends on the type of the learning problem. In particular, for regression problems, one potential choice is the squared loss function, where $L^{(i)}(\boldsymbol{\theta})$ can be expressed as

$$L^{(i)}(\boldsymbol{\theta}) = \frac{1}{2}\left(y_i - \hat{y}_i\right)^2,$$

On the other hand, for classification problems, the log likelihood function (equivalently, cross entropy loss function) that we derived for logistic regression can be used. In that case, $L^{(i)}(\boldsymbol{\theta})$ can be expressed as

$$L^{(i)}(\boldsymbol{\theta}) = -\left(y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log\left(1 - \hat{y}_i\right)\right).$$

As an example, we now evaluate $\frac{\partial L^{(i)}}{\partial \mathbf{w}^{(L)}}$ for the squared loss function (i.e., consider a regression problem). First, $\boldsymbol{\delta}^{(L+1)}$ can be evaluated as

$$\boldsymbol{\delta}^{(L+1)} = \frac{\partial L^{(i)}}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}\left[\frac{1}{2}\left(y_i - \hat{y}_i\right)^2\right] = \hat{y}_i - y_i = g(a_o) - y_i. \tag{8.11}$$

Hence, from (8.8), the derivative of the loss function with respect to the output layer can be evaluated as

$$\frac{\partial L^{(i)}}{\partial \mathbf{w}^{(L)}} = \boldsymbol{\delta}^{(L+1)} g'(a_o) \bar{\mathbf{h}}^{(L)^T} \overset{\text{(a)}}{=} \left(g(a_o) - y_i\right)\bar{\mathbf{h}}^{(L)^T}, \tag{8.12}$$

where (a) follows from (8.11) and $g(a_o) = a_o$ for regression problems.