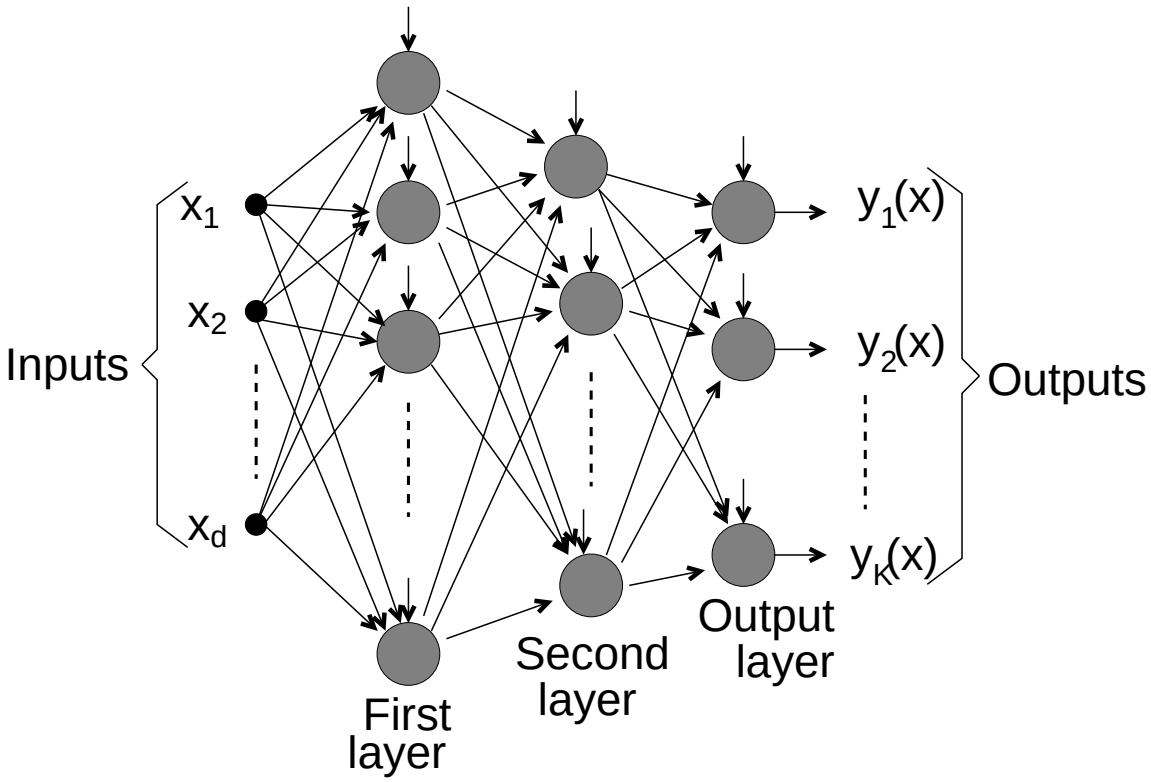
University of Cambridge
Engineering Part IIB

Module 4F10: Statistical Pattern Processing

Deep Learning: Error Back Propagation

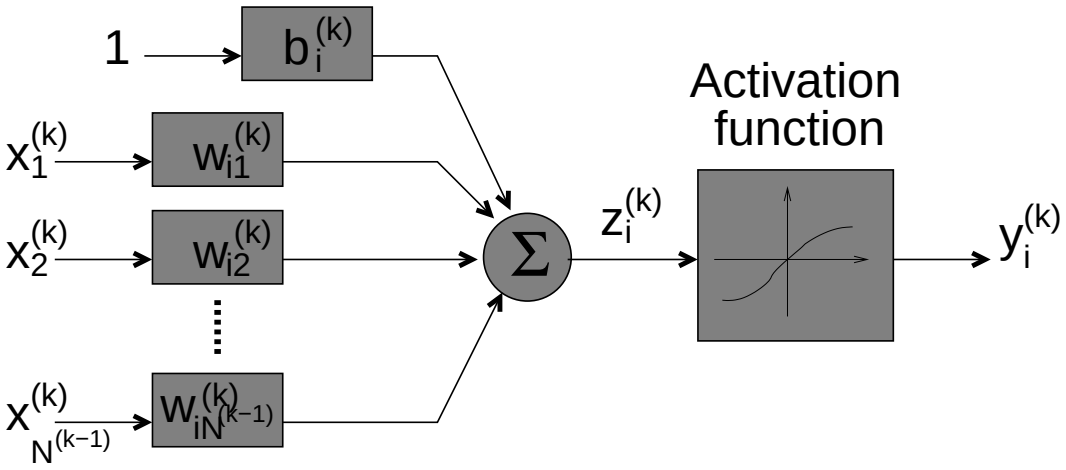


Mark Gales
mjfg@eng.cam.ac.uk
Michaelmas 2016

Reference - Error Back Propagation

Consider a multi-layer perceptron with:

- d-dimensional input data;
- L hidden layers ($L+1$ layer including the output layer);
- $N^{(k)}$ units in the k^{th} level;
- K-dimensional output.



The following notation will be used

- $\boldsymbol{x}^{(k)}$ is the input to the k^{th} layer
- $\tilde{\boldsymbol{x}}^{(k)}$ is the extended input to the k^{th} layer

$$\tilde{\boldsymbol{x}}^{(k)} = \begin{bmatrix} \boldsymbol{x}^{(k)} \\ 1 \end{bmatrix}$$

- $\mathbf{W}^{(k)}$ is the weight matrix of the k^{th} layer. By definition this is a $N^{(k)} \times N^{(k-1)}$ matrix.

Notation (cont)

- $\tilde{\mathbf{W}}^{(k)}$ is the weight matrix including the bias weight of the k^{th} layer. By definition this is a $N^{(k)} \times (N^{(k-1)}+1)$ matrix.

$$\tilde{\mathbf{W}}^{(k)} = \left[\begin{array}{cc} \mathbf{W}^{(k)} & \boldsymbol{b}^{(k)} \end{array} \right]$$

- $\mathbf{z}^{(k)}$ is the $N^{(k)}$-dimensional vector defined as

$$\mathbf{z}^{(k)} = \tilde{\mathbf{W}}^{(k)} \tilde{\boldsymbol{x}}^{(k)}$$

- $\boldsymbol{y}^{(k)}$ is the output from the k^{th} layer, so

$$y_j^{(k)} = \phi(z_j^{(k)})$$

All the hidden layer activation functions are assumed to be the same $\phi()$. Initially we shall also assume that the output activation function is also $\phi()$.

The following matrix notation feed forward equations may then used for a multi-layer perceptron with input $\boldsymbol{x}$ and output $\boldsymbol{y}(\boldsymbol{x})$.

$$\begin{aligned} \boldsymbol{x}^{(1)} &= \boldsymbol{x} \\ \boldsymbol{x}^{(k)} &= \boldsymbol{y}^{(k-1)} \\ \mathbf{z}^{(k)} &= \tilde{\mathbf{W}}^{(k)} \tilde{\boldsymbol{x}}^{(k)} \\ \boldsymbol{y}^{(k)} &= \boldsymbol{\phi}(\mathbf{z}^{(k)}) \\ \boldsymbol{y}(\boldsymbol{x}) &= \boldsymbol{y}^{(L+1)} \end{aligned}$$

where $1 \leq k \leq L+1$.

The target values for the training of the networks will be denoted as $\boldsymbol{t}$ for training example $\boldsymbol{x}$.

Error Back Propagation Algorithm

Need to calculate $\frac{\partial E}{\partial \tilde{w}_{ij}^{(k)}}$ for all layers, k, and all rows and columns of $\tilde{\mathbf{W}}^{(k)}$. Applying the chain rule

$$\frac{\partial E}{\partial \tilde{w}_{ij}^{(k)}} = \frac{\partial E}{\partial z_i^{(k)}} \frac{\partial z_i^{(k)}}{\partial \tilde{w}_{ij}^{(k)}} = \delta_i^{(k)} \tilde{x}_j^{(k)}$$

where

$$\frac{\partial E}{\partial z_i^{(k)}} = \delta_i^{(k)}$$

and the δ's are sometimes known as the individual "errors" (that are back-propagated).
For the output nodes the evaluation of δ_i is straightforward as we saw for the single layer perceptron.
To evaluate the δ_i's for hidden layers

$$\delta_i^{(k)} = \sum_m \left[\frac{\partial E}{\partial z_m^{(k+1)}} \frac{\partial z_m^{(k+1)}}{\partial z_i^{(k)}} \right]$$

where it is assumed that only the units in layer $k+1$ are connected to units in layer k, or

$$\delta_i^{(k)} = y_i^{(k)}(1 - y_i^{(k)}) \sum_m \tilde{w}_{mi}^{(k+1)} \delta_m^{(k+1)}$$

Note that all that is being done here is evaluating the differentials of the error at the output with respect to the weights throughout the network by using the chain rule for partial derivatives.

Matrix Formulation

In matrix notation we can write

$$\frac{\partial E}{\partial \tilde{\mathbf{W}}^{(k)}} = \boldsymbol{\delta}^{(k)} \tilde{\boldsymbol{x}}^{(k)\prime}$$

We need to find a recursion for $\boldsymbol{\delta}^{(k)}$.

$$\begin{aligned}
\boldsymbol{\delta}^{(k)} &= \left(\frac{\partial E}{\partial \mathbf{z}^{(k)}} \right) \\
&= \left(\frac{\partial \mathbf{z}^{(k+1)}}{\partial \mathbf{z}^{(k)}} \right) \left(\frac{\partial E}{\partial \mathbf{z}^{(k+1)}} \right) \\
&= \left(\frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{z}^{(k)}} \right) \left(\frac{\partial \mathbf{z}^{(k+1)}}{\partial \mathbf{y}^{(k)}} \right) \boldsymbol{\delta}^{(k+1)}
\end{aligned}$$

But we know from the forward recursions

$$\frac{\partial \mathbf{z}^{(k+1)}}{\partial \mathbf{y}^{(k)}} = \frac{\partial \mathbf{z}^{(k+1)}}{\partial \boldsymbol{x}^{(k+1)}} = \mathbf{W}^{(k+1)\prime}$$

This yields the recursion

$$\boldsymbol{\delta}^{(k)} = \boldsymbol{\Lambda}^{(k)} \mathbf{W}^{(k+1)\prime} \boldsymbol{\delta}^{(k+1)}$$

Matrix Formulation (cont)

Define the activation derivative matrix for layer k as

$$\mathbf{\Lambda}^{(k)} = \frac{\partial \mathbf{y}^{(k)}}{\partial \mathbf{z}^{(k)}} = \begin{bmatrix} \frac{\partial y_1^{(k)}}{\partial z_1^{(k)}} & \frac{\partial y_2^{(k)}}{\partial z_1^{(k)}} & \cdots & \frac{\partial y_{N^{(k)}}^{(k)}}{\partial z_1^{(k)}} \\ \frac{\partial y_1^{(k)}}{\partial z_2^{(k)}} & \frac{\partial y_2^{(k)}}{\partial z_2^{(k)}} & \cdots & \frac{\partial y_{N^{(k)}}^{(k)}}{\partial z_2^{(k)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1^{(k)}}{\partial z_{N^{(k)}}^{(k)}} & \frac{\partial y_2^{(k)}}{\partial z_{N^{(k)}}^{(k)}} & \cdots & \frac{\partial y_{N^{(k)}}^{(k)}}{\partial z_{N^{(k)}}^{(k)}} \end{bmatrix}$$

This has given a matrix form of the backward recursion for the error back propagation algorithm.

We need to have an initialisation of the backward recursion. This will be from the output layer (layer $L+1$)

$$\begin{aligned} \boldsymbol{\delta}^{(L+1)} &= \frac{\partial E}{\partial \mathbf{z}^{(L+1)}} \\ &= \left(\frac{\partial \mathbf{y}^{(L+1)}}{\partial \mathbf{z}^{(L+1)}}\right)\left(\frac{\partial E}{\partial \mathbf{y}^{(L+1)}}\right) \\ &= \mathbf{\Lambda}^{(L+1)}\left(\frac{\partial E}{\partial \mathbf{y}(\boldsymbol{x})}\right) \end{aligned}$$

$\mathbf{\Lambda}^{(L+1)}$ is the activation derivative matrix for the output layer.