Lecture 9

March 29, 2016

1 Nerual Network

input is the same as last time except now we are going to add many computing units. so that there are more nuerons in the network. In this case there is a single layer of compute units. In this case as well you can calculate many outputs for the classification. As in the case of the digits you may have 10 outputs signifying which digit you predict. This is a situation of multi-input ==¿ multi-output. This could lead to a situation where multiple outputs have some level of confidence in a given digit.

Training set now takes the form of:

$$(\vec{x_1}, \vec{t_1}) \rightarrow (2, [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]) \tag{1}$$

During the first interation we perform the feed forward phase and measure the error. If the error is not zero we perform the backpropagation step.

$$\vec{\nabla E} = (\frac{\partial E_1}{\partial \omega_{11}^1}, \ldots, \frac{\partial E_1}{\partial \omega_{k+1}^2}) \tag{2}$$

2 Generate Derivative

You can compute a constant times a variable and run the network backwork to get the derivative.

if you have a simple function within the network for the computational node. If you can compute the derivative you store this within the network as well. This allows you to acquire the derivative in reverse.

This is in chapter 7 in the text for this class.

we want to minimize the error for the complete data set ran on the network.

$$\bar{W_1} = \begin{bmatrix} w_{11} & w_{12} & \ldots \\ w_{21} & w_{22} & \ldots \\ \vdots & \vdots & \ldots \\ w_{n+1,1} & w_{n+1,2} & \ldots \end{bmatrix} \tag{3}$$

$$\vec{o}^{(1)} = s(\vec{o}^{(0)}\bar{W_1})\vec{o}^{(2)} = s(\vec{o}^{(1)}\bar{W_2}) \tag{4}$$