Leren Written assignment 3 2016-2017

Micha de Groot

November 2016

1

(a)

The values of θ will be lower. The regularization penalizes the values of θ by reducing them in the update rule.

(b)

Increasing the number of training examples will, if it is good data, improve the values of θ. Whether θ will increase or decrease depends on the case. The value of λ has to be higher since the part of the equation $\alpha\frac{\lambda}{m}$ will have to be roughly the same.

(c)

If there is data with less noise the learning algorithm will be able to compute better values of θ. Therefore it is expected that the values of θ will change. With less noisy data there will be less chance of overfitting the outliers in the data, so it is expected that less regularization is required. Therefore the optimal λ will probably be lower.

2

The video of Andrew Ng on Coursera shows that it is possible to learn a XOR function with a neural net with one hidden layer. The input layer consists of a bias, x_1 and x_2. The hidden layer has a bias node and two regular nodes. The weights are chosen in such a way that the activation of the first node in the hidden layer is equivalent to $x_1 AND x_2$ and the second node to $(NOT x_1) AND (NOT x_2)$. The output node has the weight in such a way that it is an OR of the hidden layer nodes.

This can easily be changed to XOR by changing the last component to $NOT(x_1 OR x_2)$, which is the same as $(NOTx_1)AND(NOTx2)$. Merging the input layer and hidden layer in one layer to makes it an original logistic regression function.
However, this gives a quite complicated function, because the result is the sigmoid function of two other sigmoid functions. If the sigmoid function is shortened to $g(x)$ the resulting function will be:
$g(-20g(-30 + 20x_1 + 20x_2) - 20g(10 - 20x_1 - 20x_2) + 10)$
It is clear that for more input variables the complexity of similar functions will expand. It also means that the difficulty of deriving the update rule will increase. This might make it too complex for efficient logistic regression.

3

1

To calculate the hypothesis of this network with input -5 we apply forward propagation with -5
Activation of Z will be: $g(0.2 * -5) = 0.27$. The activation of Y is then: $g(0.27 * 0.1) = 0.50$, which is the hypothesis.

2

The value computed by the neural network is lower than the actual value, intuitively this might mean that the parameters should be higher, in which case the input for the sigmoid function will be higher and the result will also be higher. But since there is more than one layer and the output of each layer is the input of the next, it might be the case that it is not necessary for eacht parameter to be higher.

3

$\alpha = 0.1$
First the δ values of the backward propagation have to be calculated.
$\delta^{(3)} = Act_Y - Y = 0.5 - 1 = -0.5$

The value of $\delta^{(2)}$ will be calculated next, which is defined by: $\Theta(2)^T * \delta^{(3)} .* g'(z^{(2)})$
$\delta^{(2)} = (0.1 * -0.5) .* (Act_Z .* (1 - Act_Z)) = (0.1 * -0.5) * (0.27 * 0.73) = -0.001$

After all δ values of this training example have been calculated they are multiplied by the activation values and summed in the Δ matrices, which were initialized at 0. After all training examples have been propagated forward and backward through the network all Δ matrices are divided by the data size to

make the $D^{(l)}$ matrices. In this case 1. This makes $D^{(1)}$ and $D^{(2)}$ the same as $\delta^{(2)} * Act_X^T$ and $\delta^{(3)} * Act_Z^T$ respectively.
The only step remaining is the actual updating of the Θ matrices. For this we use: $\Theta^{(l)} := \Theta^{(l)} - \alpha D^{(l)}$
$\Theta^{(1)} = 0.2 - 0.1 * (-0.001 * -5) = 0.2$
$\Theta^{(2)} = 0.1 - 0.1 * (-0.5 * 0.27) = 0.11$