Back Propagation
study note

J.F.

February 13, 2017

1 Mechanism

The deep learning mechanism is deeply influenced by neural network, in which model the neuron is the most basic component of the whole system. Accordingly, in deep learning, the neutron is used to emulate the neuron. To make the system capable of learning the very complex cases, several neutrons are combined to form a layer, and several layers together make up the whole network. A neutron is basically a homogeneous equation set, and the process of *training* the neural network is actually finding the coefficient matrix of the neutron that minimize the output error. The whole process can be broke down in 2 parts - forward pass and backpropagation.

1.1 Forward pass

In forward pass, neutrons are using existing coefficient matrix, aka weights, to predict the output. Each neutron gets the input from the previous layer or input data, and then it calculates the homogeneous equation with existing coefficient, finally it maps the result to a reasonable range by activate function to match the domain of the next neutron.

An example of activate funtion is sigmoid function.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$\text{sigmoid}'(x) = \text{sigmoid}(x)\left[1 - \text{sigmoid}(x)\right] \tag{2}$$

1.2 Backpropagation

The traning process is a loop of using the existing weights to forward calculating the prediction and backpropagate to revise each of the weights. To get the error of a specific weight, we need to calculate the derivative of the general output error. This is a multi-variable derivative problem, and there will be countless possible results, but not all of them are equally meaningful. We want to decrease the error as quickly as possible. Thus we should calculate its gradient. The thought of calculating the gradient and descend the error as quickly as possible is called *Gradient Descend*.

2 Variables

2 layers Formula	implication	Var Name
W	hidden-output weight	`h2o_weights`
w	input-hidden weight	`i2h_weights`
η	learning rate	`lr`
x	input data	`inputs`
$h = \Sigma_i w_i x_i$	signals into hidden layer	`hidden_inputs`
$a = f_h(h)$	signals from hidden layer	`hidden_outputs`
$H = W \cdot a$	signals into final layer	`final_inputs`
$\hat{y} = f_f(H)$	signals from final layer	`final_outputs`
$E = y - \hat{y}$	output error	`output_err`
$G = f'_f(H)$	output layer gradients	`output_grad`
$\delta^o = EG$	output layer error	`output_grad_err`
$g = f'_h(h)$	hidden layer gradients	`hidden_grad`
$\delta^h = W\delta^o g$	hidden unit error	`hidden_grad_err`
$\Delta W = \eta\delta^o a$	hidden-output GD step	`h2o_weights_step`
$\Delta w = \eta\delta^h x_i$	input-hidden GD step	`i2h_weights_step`