Deep Learning Problem Set 2

June 2017

Problem 1

Prove that the derivative of the sigmoid function satisfies the equality

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

This formula will be important should you choose to implement the error back-propagation algorithm yourself.

Problem 2

Explain why the time complexity of processing of one training sample is $O(|W|^2)$. What do you think is the space complexity of the backprop (on one sample)?

Problem 3

Imagine you are training a network using the error back-propagation and you noticed your error function stuck at some relatively large value (i.e. your accuracy is low). The learning continues but the accuracy does not improve. Provide two or more possible explanations of this behaviour and explain what can be done to avoid the causing issue in each case.

Problem 4

One can think that looking for a solution for a specific problem might be very hard due to the size of the input space. However there is an alleviating factor, called weight-space symmetry.

Two networks have the same network function if they produce exactly the same output for each possible input (in other words, they model the same function). There are many alternative weight assignments that produce exactly the same network function. Consider a network with single hidden layer having N neurons with *tanh* activation functions. Note that this function satisfies $tanh(-x) = -tanh(x)$. Argue that for (ALMOST!) any weight assignment W there are at least $2^N - 1$ other assignments producing exactly the same network function.

In fact there are many more, in general case. Hence, (ALMOST!) any solution generates a whole family of identical solutions.

Hint: think of the symmetries.

Problem 5

Show that the any multilayer fully-connected feedforward network with all the layers being linear (i.e., each neuron just outputs its weighted input: $Wx + b$) can be fully represented with a one-layer network with a single linear layer. In other words, stacking linear layers does not improve the representative power of the network.

Hint: think of the network as a composite function, where each layer represents one component, i.e., $\hat{y} = f_n(...f_2(f_1(x))...)$.

Problem 6

The backpropagation algorithm is a truly universal technique that can be extended beyond computing the gradients. Hessians, or the second derivative tensors, play a significant role in more advanced, higher-order optimization algorithms, that analyze behaviour of the gradient vector to guide optimization process. Here we will try to compute a fragment of the Hessian for a multilayer network. In general, computing $H_{lkji} = \frac{\partial^2 E}{\partial w_{lk} \partial w_{ji}}$ is a complicated dynamic programming problem. Here we just take a glimpse at it, by considering a simpler case, when both weights belong to the same layer. Recall that we defined $\delta_j = \frac{\partial E}{\partial z_j}$ and let's introduce $\delta_{jj'} = \frac{\partial^2 E}{\partial z_{j'} \partial z_j}$ for all pairs (j, j') of some layer l of the network. Prove that

$$\frac{\partial^2 E}{\partial w_{j'i'} \partial w_{ji}} = z_{i'} z_i z'_{j'} (\delta_{jj'} z_j + I_{jj'} \delta_j),$$

where $I_{jj'}$ is the corresponding element of the identity matrix.