Deep Learning

Assignment 3

Back-Propagation

Flavie Vampouille MSc in DS&BA

CentraleSupelec

ESSEC Business School

Training Neural Networks

Understanding the code

Associating the nnet.m script with the course number 5 (slide 59) I obtain:

- d_output_d_activation = $\frac{\partial y}{\partial b}$ (gradient of the output with respect to the activation)
- d_loss_d_activation = $\frac{\partial L}{\partial b}$ (gradient of the loss with respect to the activation)
- d_loss_d_output = $\frac{\partial L}{\partial y}$ (gradient of the loss with respect to the output)

When we compute “ 2*lambda * Weights{l}(:, 2:end); ” we started at two because the first term in Weights: Weights{l}(:,1) is the bias.

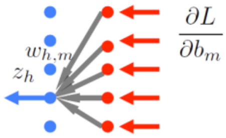
We are doing :

d_loss_d_output_below = Weights{l}' * d_loss_d_activation;

because it is a backward node (like in lecture 5 slide 65). We do the sum on weigths * gradient of loss with respect to activation.

Linear layer in backward mode: one from all

$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$

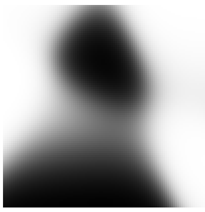


$$\frac{\partial L}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \cdot \frac{\partial b_c}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} w_{h,c}$$

Extension: using softmax

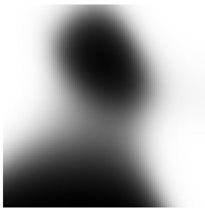
Using the sigmoid method alone I obtain:

Relative Difference: 1.31281e-10



Using the softmax method on the top layer neuron I obtain:

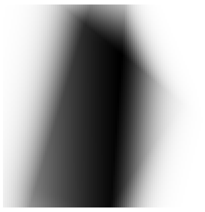
Relative Difference: 6.73298e-11.



Extension: Using ReLUs

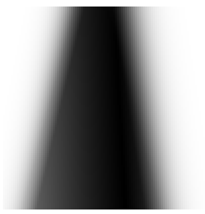
Using the sigmoid method on the top layer neuron and ReLU method on the hidden layer I obtain:

Relative Difference: 1.36326e-10



Using the softmax method on the top layer neuron and ReLU method on the hidden layer I obtain:

Relative Difference: 1.01353e-10



Experiment

Not done.