MTAT.03.227 Machine Learning
Practice session 9

November 11-13, 2019

1. **Backpropagation**

   Backpropagation is a common method for training a neural network. The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

   To go through the backpropagation process in this practice session, we're going to use a neural network with one input X, two layers each consisting of two hidden neurons, one output neuron which yields a prediction $\hat{y}$. Additionally, the hidden and output neurons will include a bias.

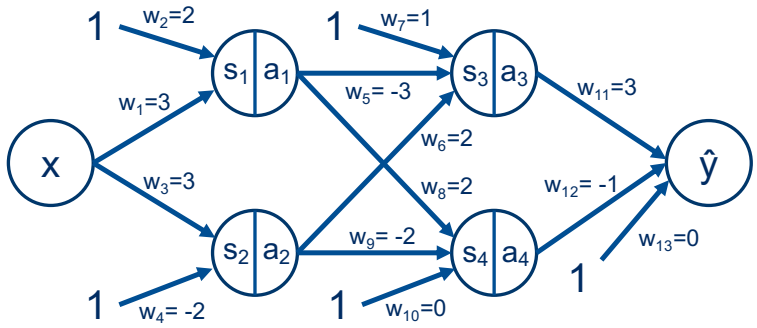

Figure 1: Simple neural network with one input, one output and two hidden layers

We are given input $x = 0.5$, we want the neural network to output $y = 0.25$. We will set the learning rate to be 0.1

(a) **Forward pass** To begin, lets see what the neural network currently predicts given the weights and biases above and the input. To do this we'll feed those inputs forward through the network.

The process looks like this: We figure out the total net input to each hidden layer neuron, squash the total net input using an activation function (here we use the logistic function), then repeat the process with the output layer neurons.

i. The total net input for the hidden neurons in the first hidden layer:

$$s_1 = w_1 \cdot x + w_2 \cdot 1 = ??? \tag{1}$$

$$s_2 = ??? \tag{2}$$

ii. Squash the total net input using an activation function to get the output of neuron. Let's use logistic function:

$$a_1 = \frac{1}{1 + e^{-s_1}} = ??? \tag{3}$$

$$a_2 = ??? \tag{4}$$

iii. We repeat this process for next hidden layer as well.

$$s_3 = w_5 \cdot a1 + w_6 \cdot a_2 + w_7 = ??? \tag{5}$$
$$s_4 = ??? \tag{6}$$

Again, using logistic function:

$$a_3 = \frac{1}{1 + e^{-s_3}} = ??? \tag{7}$$
$$a_4 = ??? \tag{8}$$

iv. Calculate the output.
We use linear activation function here for the regression task (consider it as we don't do anything with the output of the last neuron)

$$\hat{y} = w_{11} \cdot a_3 + w_{12} \cdot a_4 + w_{13} = ??? \tag{9}$$
$$\tag{10}$$

v. Now we calculate the error for the output neuron using the squared error function. Here we have only one output, so no need to sum up errors.

$$E = \sum (target - output)^2$$

$$E = (y - \hat{y})^2 = ??? \tag{11}$$
$$\tag{12}$$

vi. Are we far from the expected output?

(b) **Backward pass** Our goal is to now modify the weights so that the error decreases. Consider w_5. We want to know how much a change in w_5 affects the total error E, aka $\frac{\partial E}{\partial w_5}$. To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate η):

$$w_{5new} = w_5 - \eta \frac{\partial E}{\partial w_5}$$

i. What does this remind you?

ii. Recall useful formulae: derivative chain rule (13), derivative of logistic function (14)

$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial u} \tag{13}$$
$$\frac{d\sigma}{dz}(z) = \sigma(z)(1 - \sigma(z)) \tag{14}$$

Backpropagation is a way of computing gradients of expressions through recursive application of chain rule.

$$\frac{\partial E}{\partial w_5} = ??? \tag{15}$$

iii. Do the same for w_9 (*Hint: you can use the below table for the partial derivatives*)

$$\frac{\partial E}{\partial w_9} = ??? \tag{16}$$

Derivative	$\frac{\partial E}{\partial \hat{y}}$	$\frac{\partial \hat{y}}{\partial w_{11}}$	$\frac{\partial \hat{y}}{\partial w_{12}}$	$\frac{\partial a_3}{\partial s_3}$	$\frac{\partial a_4}{\partial s_4}$
Value	-0.5968	0.2392	0.7661	0.1820	0.1792

iv. Now let's look w_1. Find

$$\frac{\partial E}{\partial w_1} = ??? \tag{17}$$

v. Are we there yet? What do we do with these derivatives now?

vi. Calculate new values for w_5 and w_1 given that $\eta = 0.1$.

vii. What about the bias terms w_2, w_4, w_7, w_{10} and w_{13}?

viii. Are we there yet?

2. **Softmax function**

Softmax function - a generalization of the logistic function for K classes

$$\sigma_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

It takes a vector of arbitrary real-valued scores (in $\mathbf{z}$) and squashes it to a vector of values between $[0, 1]$ that sum to one.

In the below table you are given the output of a scoring function f for four classes considering the given image.

Class	Score	e^{z_j}	$\sigma_j(\mathbf{z})$
Dog	-3.44		
Cat	1.16		
Boat	-0.81		
Airplane	3.91		

Cross-entropy loss:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{18}$$

M – number of classes (dog, cat, boat, airplane)
log – the natural log
y – binary indicator (0 or 1) if class label c is the correct classification for observation o
p – predicted probability of observation o to belong to class c

(a) Can these scores be used in the calculations of cross-entropy loss? Why?
(b) Fill the 3rd column in the table.
(c) Fill the 4th column in the table
(d) Why can't we just use the values in the 3rd column?
(e) Find the cross-entropy loss.
(f) What is the softmax function in case of 2 classes? *Hint: Look at the sigmoid function.*

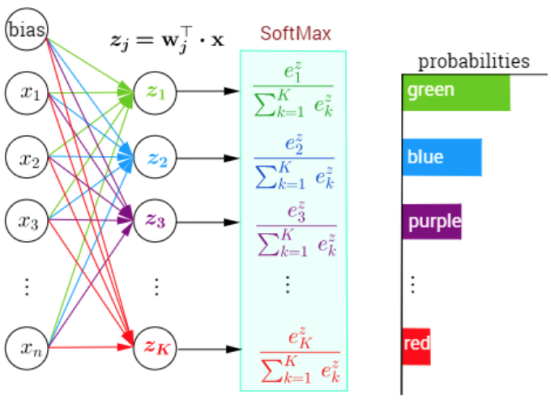

Figure 2: Softmax classifier