MCB128

Quiz b0

September 11, 2026

Name: _____ **Score:** _____

Instructions:

- Show all work for full credit.
- Can check notes; no prompts to chatbot allowed

---

1. (**2/10 pts**) Why is there an activation function in the single neuron? in other words, which property do the logistic, the sigmoid (tanh) and the step function share in common? and why is that useful/necessary in a perceptron?

   The activation function introduces a non-linearity in the relationship of the inputs ($\mathbf{x}$) to the outputs (y) through the weights ($\mathbf{w}$).

   Calling the linear combination of inputs and weights, the activation a,

   $$a = W_0 + \sum_i x_i W_i$$

   Then the perceptron introduces the output using an activation function f (here the logistic function)

   $$y = f(a) = \frac{1}{1+e^{-a}} = \frac{1}{1+e^{-(W_0+\sum_i x_i W_i)}}$$

   In the absence of the activation function, the relationship of inputs to outputs is just linear in the weights

   $$y = a = W_0 + \sum_i x_i W_i$$

   which is too simple, and will make adding layers of more neurons to the system pointless.

2. (**2/10 pts**) Backpropagation changes the parameters of the model in a way that is proportional to the change of the error ($G(\mathbf{w})$) with that parameters $\mathbf{w}$).

   $$\mathbf{w}^{new} = \mathbf{w}^{0ld} - \eta \frac{\delta G(\mathbf{w})}{\delta \mathbf{w}} \mid_{\mathbf{w}=\mathbf{w}^{0ld}} .$$

   The proportionality constant $\eta > 0$ is called the "learning rate". The learning rate is a meta parameter, which means, that it is not learned by the model (as the weights $\mathbf{w}$ are), but it has to be provided as an input, and it can take any arbitrary value.

Why do you think it has that name? on other words, what would happen if we use a very large learning rate? and if we use a very small one?

The learning rate determines how large are the changes introduced in the parameters at each step of the gradient descent process. If η is large, the weights updates are going to swindle around a lot and we many never reach a minimum for the loss. If η is very small, the optimization will proceed securely to a minimum of the loss, but it may take a long long time to get there.

There is no theoretical way to set the learning rate, it is a matter of trial and error.

3. (**6/10 pts**) We have introduced an error (or loss) function of the form

$$G(\mathbf{w}) = \sum_n [t_n \log y_n + (1 - t_n) \log(1 - y_n)]$$

(a) (**2/10 pts**) Where is the dependence on the weights $\mathbf{w}$ hiding?
The dependency is in y_n

$$y_n = f(a) \quad a = W_0 + \sum_i x_n^i W^i$$

(b) (**2/10 pts**) Here is a different loss function that we could use

$$G1(\mathbf{w}) = \sum_n (t_n - y_n)^2$$

Is this a reasonable loss function? Name at least two properties of G1 that makes it a good substitute of G

Yes. (1) it is always semi-definite positive. (2) It goes to zero as the output y_n approach the actual values t_n. (3) It is differentiable.

(c) (**2/10 pts**) How about a loss function of the form

$$G2(\mathbf{w}) = \sum_n (t_n - y_n).$$

Is this a good loss function? How would you fix it? No. It has properties (2) and (3), but it is not always positive, the loss will go up and own in value depending on whether the outputs go above of below the labels.

A valid related loss function uses the norm to satisfy point (1).

$$G3(\mathbf{w}) = \sum_n \mid t_n - y_n \mid .$$