Quiz 1

Started: Jan 18 at 11:34pm

Quiz Instructions

Question 1 — 1 pts

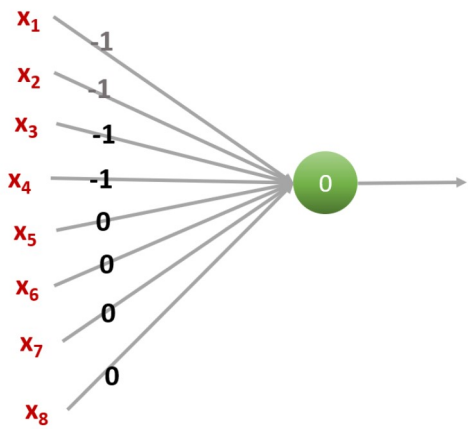
Which of your quiz scores will be dropped?

- ○ Lowest 3 quiz scores
- ○ Lowest 2 quiz scores
- ○ Lowest 1 quiz score
- ○ None of them

Question 2 — 1 pts

Under which conditions will the perceptron graph below fire, assuming binary inputs? (select all that apply).

In the figure the number within the circle is the threshold. The numbers on the edges are weights. The perceptron fires if the total incoming signal is at least equal to the threshold.

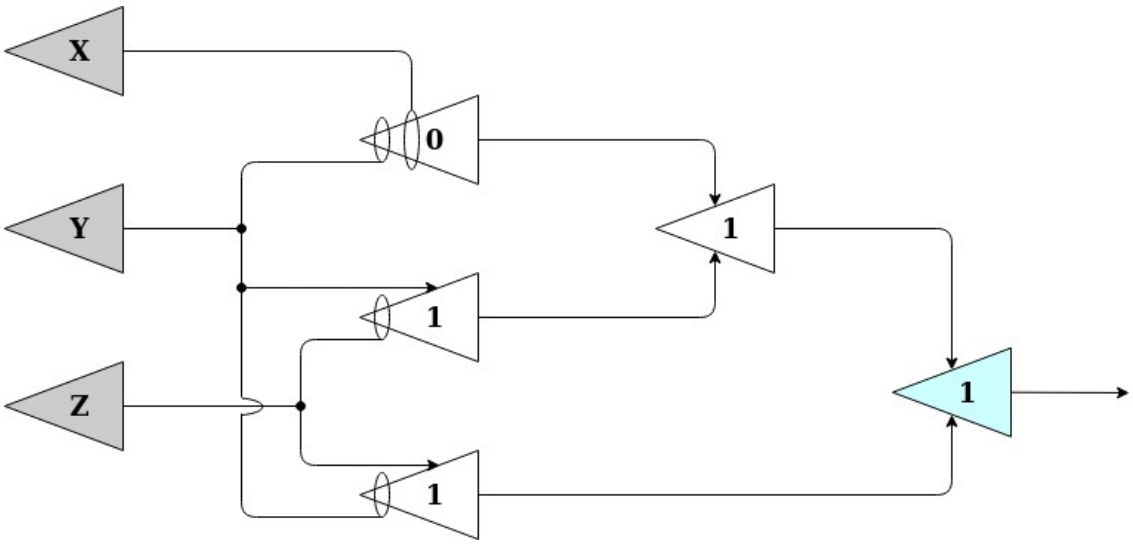


- ☐ $\bar{x_1}, \bar{x_2}, \bar{x_3}, \bar{x_4}, \bar{x_5}, \bar{x_6}, \bar{x_7}, \bar{x_8}$
- ☐ x_1, x_2, x_3, x_4
- ☐ Never fires

☐ $\bar{x_1}, \bar{x_2}, \bar{x_3}, \bar{x_4}, x_5, x_6, x_7, x_8$

Question 3 1 pts

Which of the following boolean functions does the McCullouch and Pitts network given below represent? (The gray neurons provide the binary input signals and the blue neuron represents the output neuron. The number inside each neuron is the threshold, **T**. For a neuron to send a pulse, it should represents the minimum number of input signals required for that neuron to fire it should receive at least **T** excitory pulses and no inhibitory pulses)



○ $\bar{Y}Z + Y\bar{Z} + \bar{X}\bar{Y}$

○ $\bar{Y}Z + Y\bar{Z} + XY$

○ $\bar{Y}Z + Y\bar{Z} + X\bar{Y}$

○ $\bar{Y}Z + Y\bar{Z} + \bar{X}Y$

Question 4 1 pts

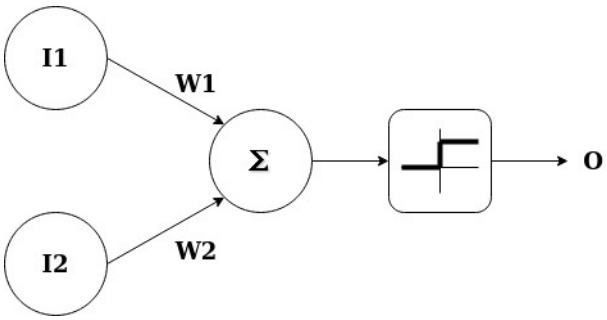
We sometimes say that neural networks are connectionist machines as opposed to Von Neumann machines. Which of the following describe why we make this distinction? (select all that apply)

☐ Because of its flexibility, a Von Neumann machine is capable of computing any Boolean function of a given number of Boolean inputs, whereas connectionist machines, no matter how complex, are fundamentally unable to model certain types of Boolean functions.

- ☐ It is possible to create hardware implementations of Von Neumann machines (e.g. CPU's) as well as software implementations (e.g. virtual machines). However, connectionist machines can only be implemented in software (e.g. neural networks in Python).
- ☐ A Von Neumann machine can be used for general-purpose computing by simply providing a different program, without changing the machine itself. A connectionist machine implements a specific program, and changing the program requires changing the machine.
- ☐ A Von Neumann machine has a general purpose architecture with a processing unit that is distinct from the memory that holds the programs and data. A connectionist machine makes no distinction between processing unit and the program.

Question 5 1 pts

John has learned that the Hebbian learning rule is broken and leads to unstable learning. The reason, he has learned, is that the Hebbian learning rule always increments weights in an unbounded manner, without ever decrementing them. To convince himself that this is true, he trains the following perceptron with Hebbian learning:



Here, **I1** and **I2** are two Boolean inputs, **W1** and **W2** are their corresponding weights, and **O** is the output after applying a threshold activation $\phi : \mathbb{R} \rightarrow \mathbb{R}$ to the weighted sum of the inputs (**W1xI1+W2xI2**) such that $\phi(x) = \begin{cases} 0, \text{if } x \leq 0 \\ 1, \text{otherwise} \end{cases}$.

He uses the Hebbian learning rule that we saw in slide 71 of lecture 1 (as posted in the slides on the course website). He trains the network with the following input table:

I1	I2	O
1	0	1
1	1	0

and cycles through it 100 times. He initializes **W1** and **W2** with 0 (Note that he doesn't use a bias term). To keep things balanced, in odd numbered iterations (where iteration numbers are numbered from 0 to 99), he updates the weights first with training instance 1, and then with training instance 2. In even numbered iterations he goes through the table in reverse order, first updating with training instance 2 and then with training instance 1. He repeats the experiment with learning rates η of 0.01, 0.1, and 1.

Subsequently, in each case, he inspects the weights and tests the network on the following test instances:

I1	I2	O
0	0	1
0	1	1
1	0	1
1	1	0

Which of the following does he find to be true? (You might want to run a simulation of John's experiments)

- ☐ The output of the learned model with $\eta = 0.1$ for **I1**=0 and **I2**=0 is **O**=1.
- ☐ The output of the learned model with $\eta = 0.01$ for **I1**=1 and **I2**=1 is **O**=1.
- ☐ The weights learned after 100 epochs with $\eta = 0.1$ are **W1**=10 and **W2**=10.
- ☐ The weights, as a function of the iteration number, are non-decreasing.
- ☐ The weights learned after 100 epochs with $\eta = 1.0$ are **W1**=0 and **W2**=100.
- ☐ The accuracy of the model learned with a $\eta = 0.1$ is 50% on the test instances.

Question 6

1 pts

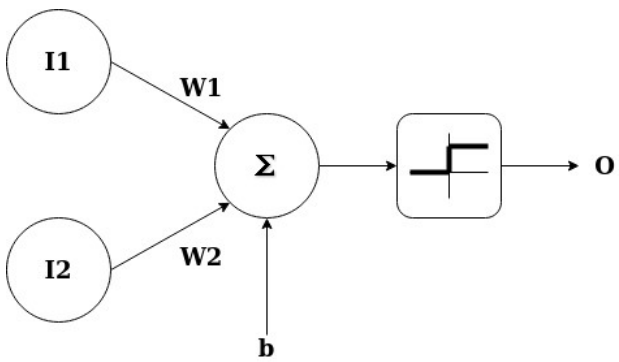
John has learned that the Hebbian learning rule is broken and leads to unstable learning. The reason, he has learned, is that the Hebbian learning rule always increments weights in an unbounded manner, without ever decrementing them, as a result of which the weights can become arbitrarily large.

He realizes that the problem is that the weights are always updated by $\eta x_i y$, (where x_i is the i-th input and y is the output) *even when the neuron has already obtained the correct output* y. This perceptual reinforcement of already-learned concepts seems to him to be the source of the instability.

To fix this, he modifies the Hebbian rule to update the weights, not by the target output y, but by the **error** between the target output and what the unit actually outputs currently.

Instead of updating the weights by $\eta x_i y$, he updates the weights by $\eta x_i (y - t)$, where t is the output of the neuron using the *current* set of weights, and y is what the output must ideally be according to the training data. Note that if the neuron outputs the correct value for the input, then $y = t$ and the weights no longer get updated. They only get updated if $y \neq t$. To test his new rule, he does the following:

He trains the following perceptron with his new rule:



Here, **I1** and **I2** are two Boolean inputs, **W1** and **W2** are their corresponding weights, **b** is a bias term, and **O** is the output after applying a threshold activation $\phi : \mathbb{R} \rightarrow \mathbb{R}$ to the weighted sum of the inputs (**W1xI1+W2xI2 + b**) such that $\phi(x) = \begin{cases} 0, \text{if } x \leq 0 \\ 1, \text{otherwise} \end{cases}$. He trains the network with the following input table:

I1	I2	O
0	0	1
0	1	1
1	0	1
1	1	0

and cycles through it 100 times. He initializes **W1**, **W2** and **b** with 0. To keep things balanced, in odd numbered iterations (where iteration numbers are numbered from 0 to 99), he iterates through the training instances in order 1,2,3,4, and in even numbered iterations he iterates through them in order 4,3,2,1. He repeats the experiment with learning rates η of 0.01, 0.1, and 1.

He then inspects the weights of the network in each case and tests the network on all 4 combinations of inputs.

Which of the following does he find to be true? (You might want to run a simulation of John's experiments)

- ☐ The output of the model trained with η =0.1 for **I1**=1 and **I2**=1 is **O**=1.
- ☐ The output of the model trained with η =0.1 for **I1**=0 and **I2**=0 is **O**=1.
- ☐ The bias of the model trained with η =0.01 is **b**=0.03.
- ☐ The weights, as a function of the iteration number, are non-decreasing.
- ☐ The weights of the model trained with η =0.01 are **W1**=0.01 and **W2**=0.02.
- ☐ The accuracy of the model trained with η =0.01 is 75% over the 4 combinations of inputs.

Question 7

1 pts

For this question, please read the paper titled "**Multilayer Feedforward Networks are Universal Approximators** (http://deeplearning.cs.cmu.edu/document/readings/Hornik_Stinchcombe_White.pdf)" by Kur Hornik, Maxwell Stinchcombe and Halber White. According to the paper, which of the following statements are true? Here the term “accuracy” refers to the ℓ_∞ error between the function and its approximation.

- ☐ An MLP with squashing-activation units can model any function exactly over any set of N points using only N hidden units.
- ☐ A “sum” function of the kind $\sum_g g(x)$ can approximate a “sum-product” function of the kind $\sum_g \prod_i g_i(x)$ with arbitrary accuracy on any compact set.

- ☐ A smooth continuous function cannot be modelled to arbitrary precision on any compact set by MLPs with the threshold function, which is discontinuous and undefined at 0.
- ☐ An MLP can model any smooth, continuous function to arbitrary precision over any input space if the activation functions of the neurons are also continuous.
- ☐ An MLP can model any function to any arbitrary degree of accuracy.

Question 8 — 1 pts

How does the number of weights in an XOR network with 1 hidden layer grow with the number of inputs to the network?

- ○ Between polynomial and exponential
- ○ Exponential or faster
- ○ Polynomial but faster than linear
- ○ Linear

Question 9 — 1 pts

An activation function must always be a squashing function.

- ○ True
- ○ False

Question 10 — 1 pts

For any Boolean function on the real plane, which outputs a 1 within a single connected but **non-convex** region in the plane (*i.e.* not a convex polygon), you need at least three layers (including the output layer) to model the function **exactly**.

- ○ True
- ○ False

Not saved Submit Quiz