Multi-Layer Neural Networks

LIN 313 Language and Computers
UT Austin Fall 2025

(***let any math today just wash over you)

Admin

- HW 3 due tonight at midnight
- extra credit opportunities

Overview 10/14

- limitations of single layer neural networks: the XOR problem
- multi-layer perceptrons (MLP)
- training large multi-layer networks
  - gradient descent
  - backpropagation

What can the perceptron do?

- "which side of a piece of paper is the X on?"
- logical functions

it can draw a line in space

Perceptron AND gate

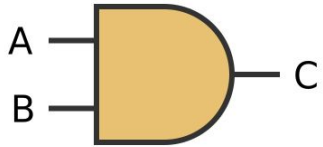
lookup table

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

4 different representations of logical AND

Perceptron AND gate

physical circuit



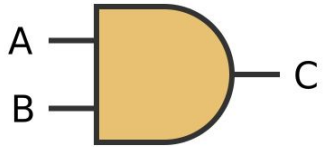
lookup table

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

4 different representations of logical AND

Perceptron AND gate

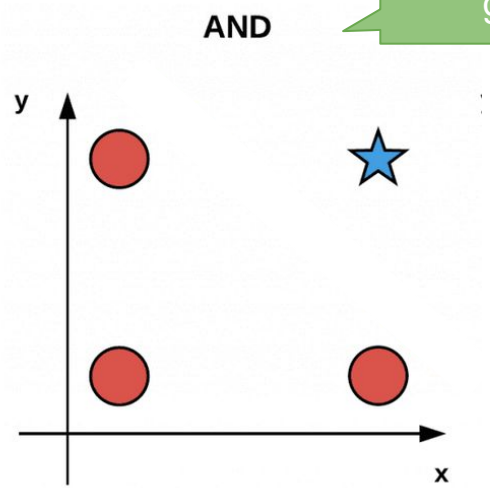
physical circuit



lookup table

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

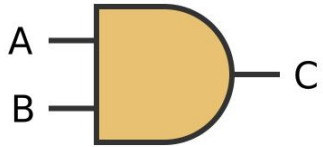
geometric



4 different representations of logical AND

Perceptron AND gate

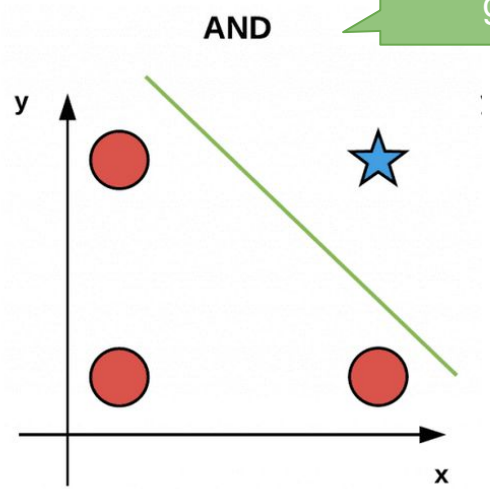
physical circuit



lookup table

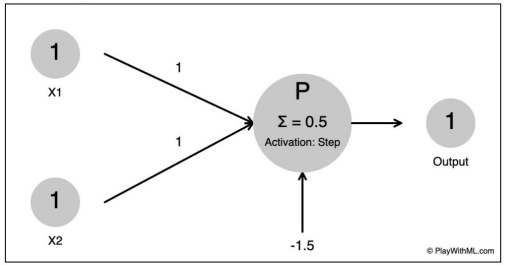
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

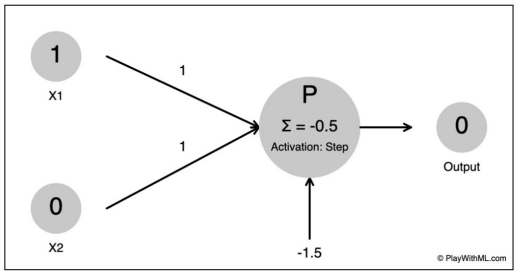
geometric

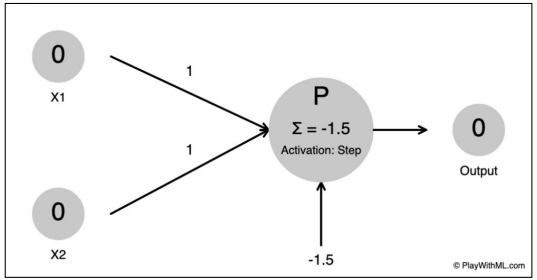


4 different representations of logical AND

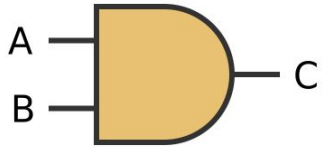
Perceptron AND gate







physical circuit

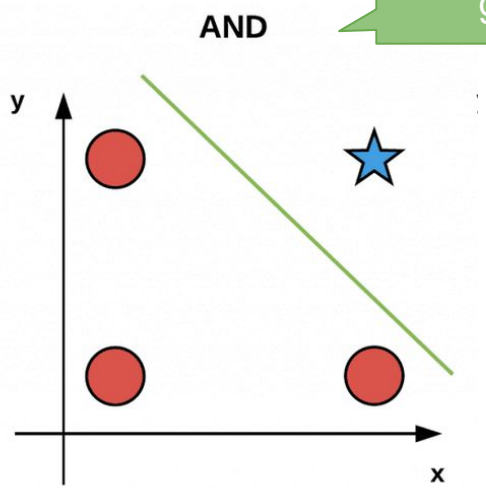


lookup table

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

perceptron

geometric



4 different representations of logical AND

Perceptron OR gate

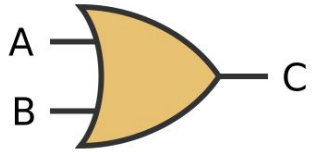
lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

4 different representations of logical OR

Perceptron OR gate

physical circuit



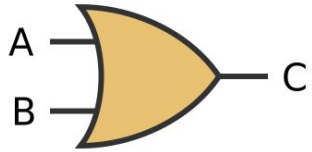
lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

4 different representations of logical OR

Perceptron OR gate

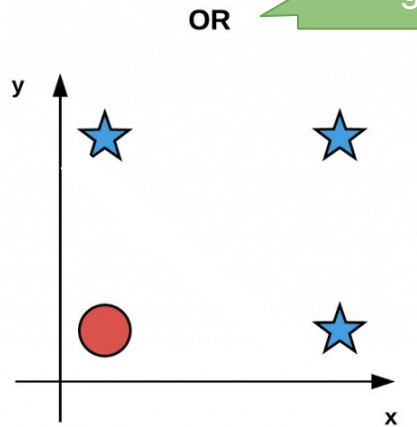
physical circuit



lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

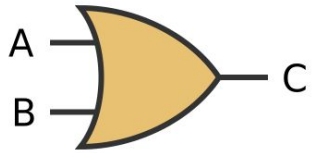
geometric



4 different representations of logical OR

Perceptron OR gate

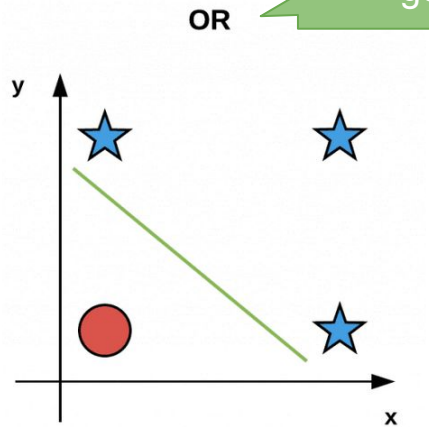
physical circuit



lookup table

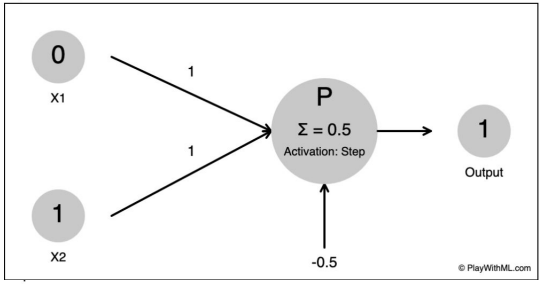
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

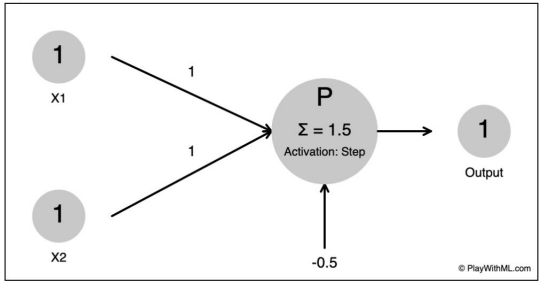
geometric

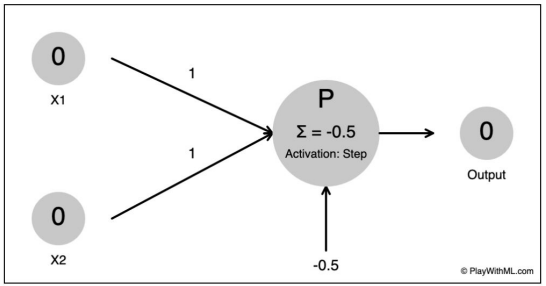


4 different representations of logical OR

Perceptron OR gate

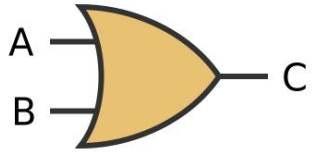






perceptron

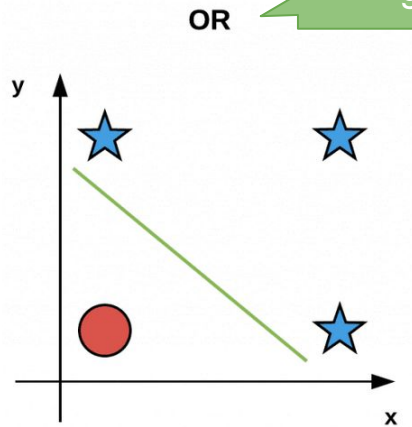
physical circuit



lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

geometric



4 different representations of logical OR

Perceptron XOR gate

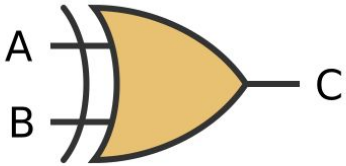
lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

different representations of logical XOR

Perceptron XOR gate

physical circuit



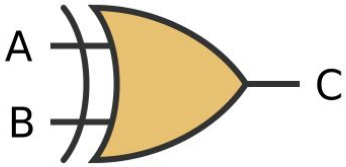
lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

different representations of logical XOR

Perceptron XOR gate

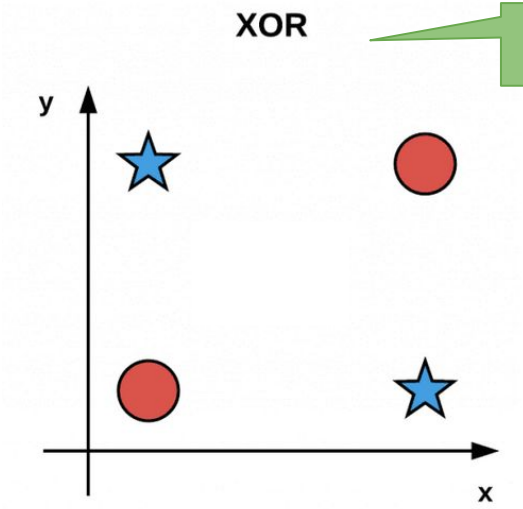
physical circuit



lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

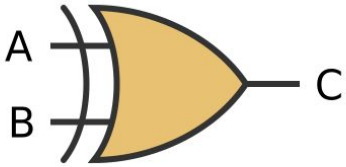
geometric



different representations of logical XOR

Perceptron XOR gate

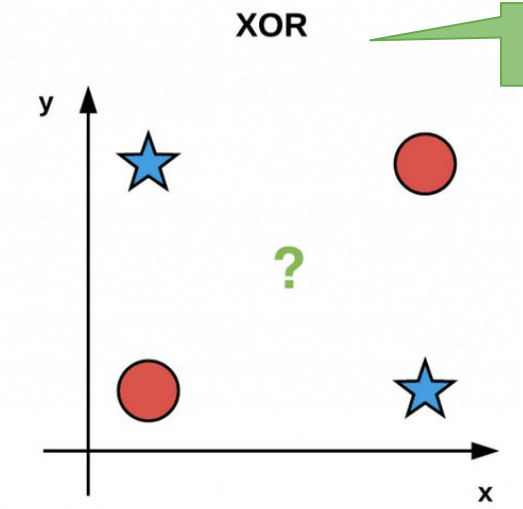
physical circuit



lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

geometric

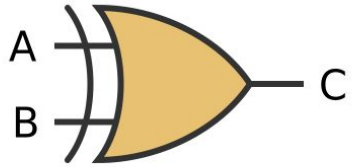


different representations of logical XOR

Perceptron XOR gate

XOR is NOT linearly separable. We can't draw a line in space to solve this problem, so we can't design (or learn) a solution with the perceptron

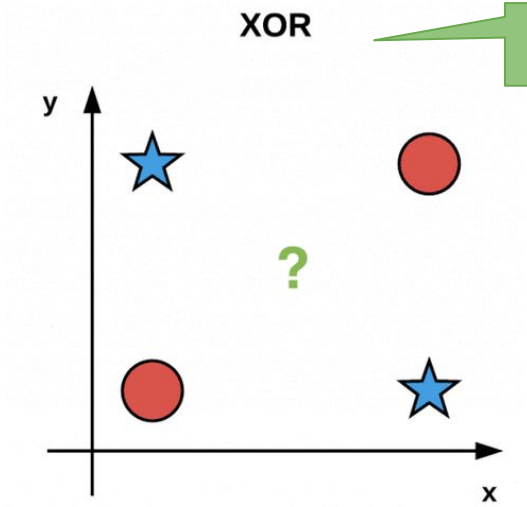
physical circuit



lookup table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

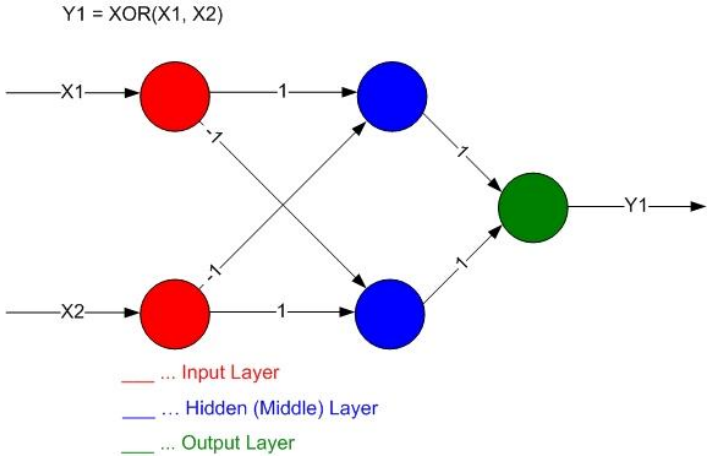
geometric

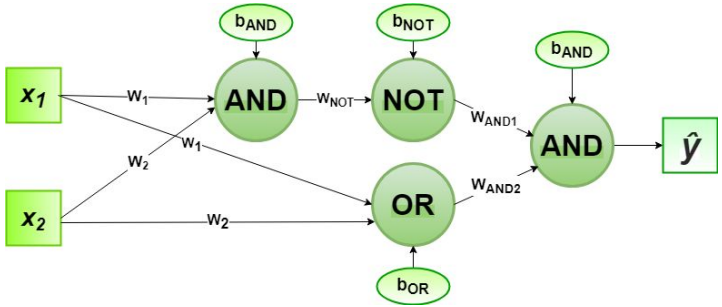


different representations of logical XOR

Rosenblatt was not a dummy

- Perceptrons are based off the brain
- Brains aren't a single neuron
- Even Minsky and Papert, the haters who caused the first AI winter, knew this was possible.

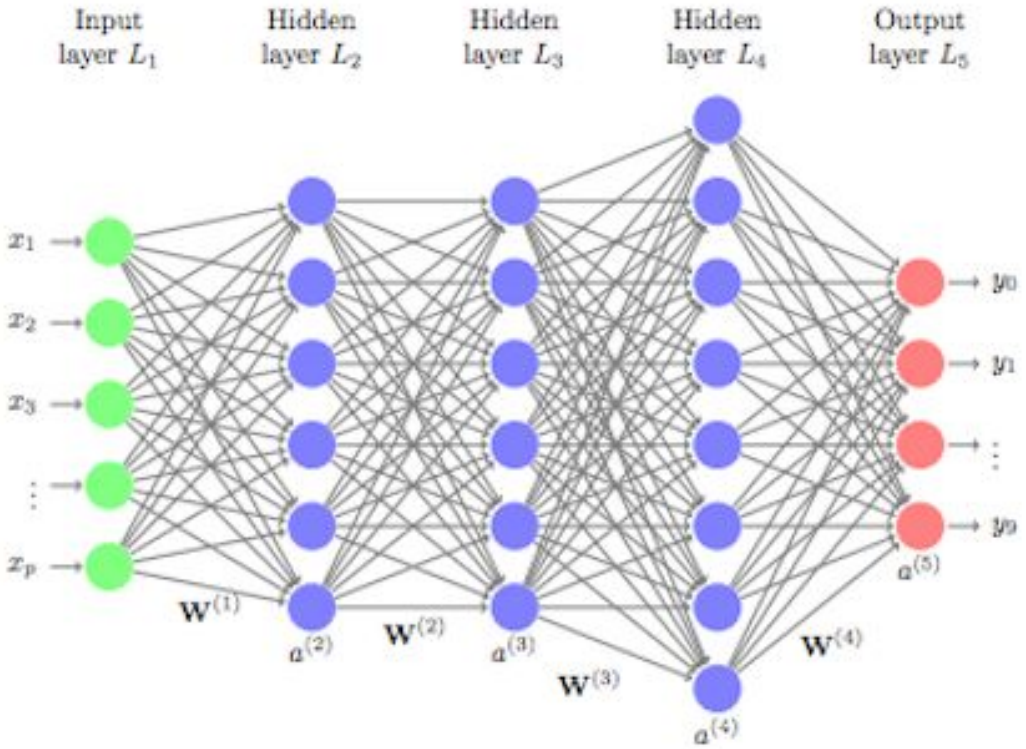




Multi-layer Perceptrons

A fully connected, feed-forward neural network:

Each weight in layer i sends its output to every weight in the next layer i+1

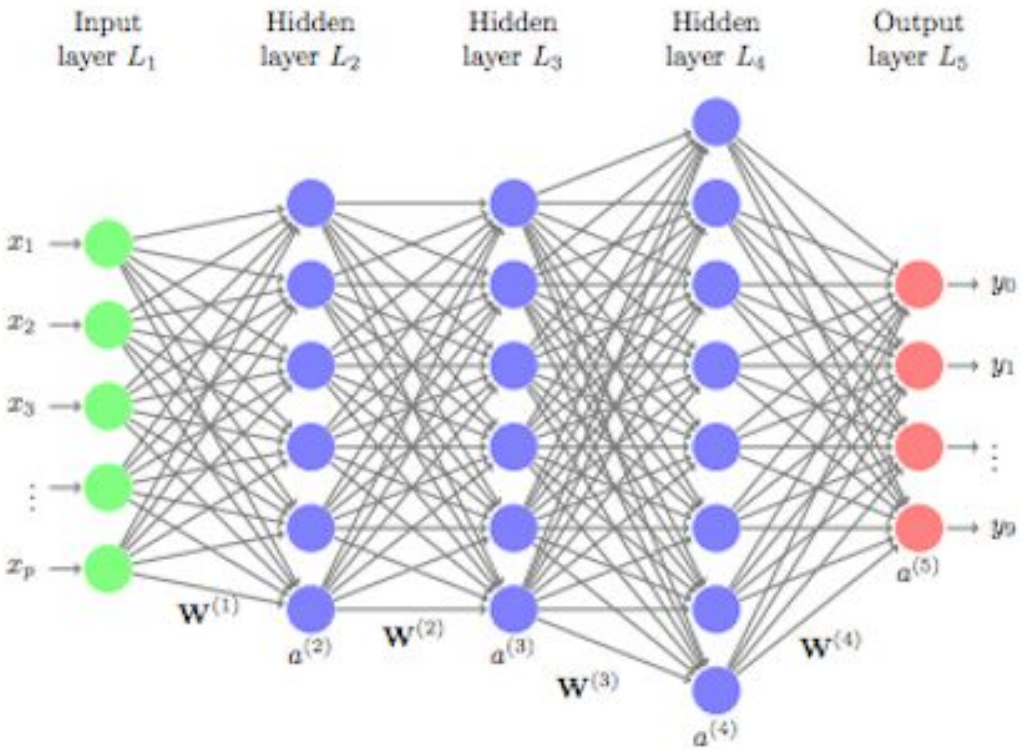


Multi-layer Neural Networks

Frank Rosenblatt, the inventor of the Perceptron, anticipated multi-layer perceptrons.

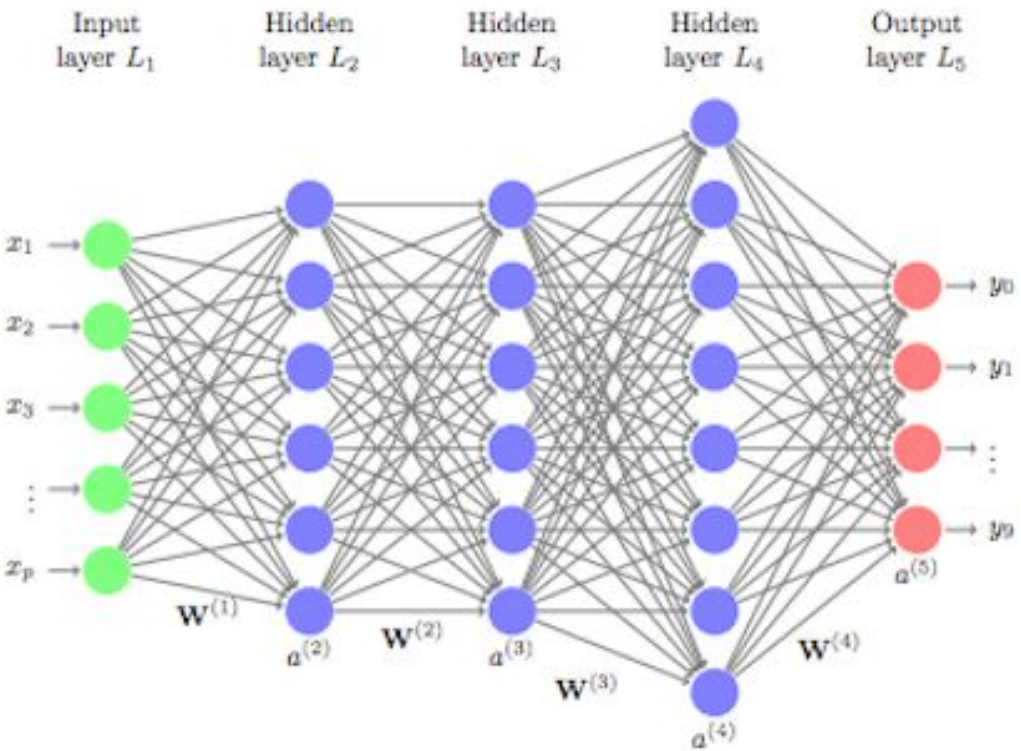
The problem wasn't designing complex networks.

It was training them



Training Large Neural Networks

- The perceptron rule goes weight by weight and then changes it a little bit towards predicting the right answer
- We can't use the Perceptron rule to train this network.



Loss

On ML, loss is a measure of how far your guess is from the right answer.

In other words, it's a metric for how much *error* the model produces. We talk about it as a *cost function* or a *loss function*:

$$E = (y - \hat{y})$$

The error (loss) is equal to the difference between the real value and the predicted value.

"But wait! E is a number, not a function!"

E is a function of our inputs: when the inputs change, the error changes.

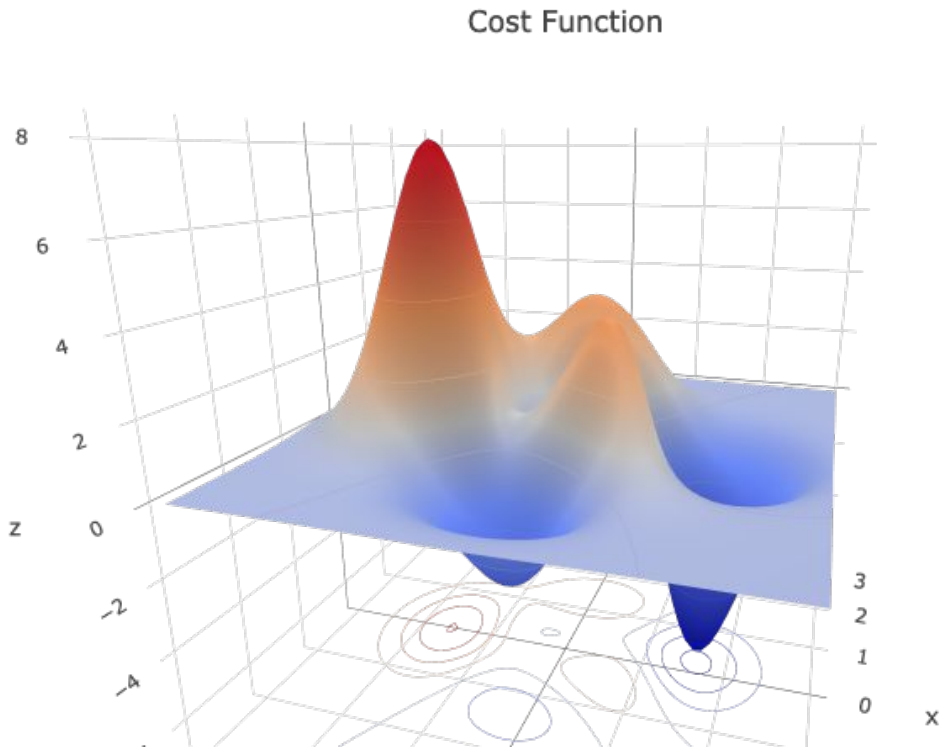
$$E = (y - (\boldsymbol{wx} + \ b) $$

The Loss Landscape

https://blog.skz.dev/gradient-descent

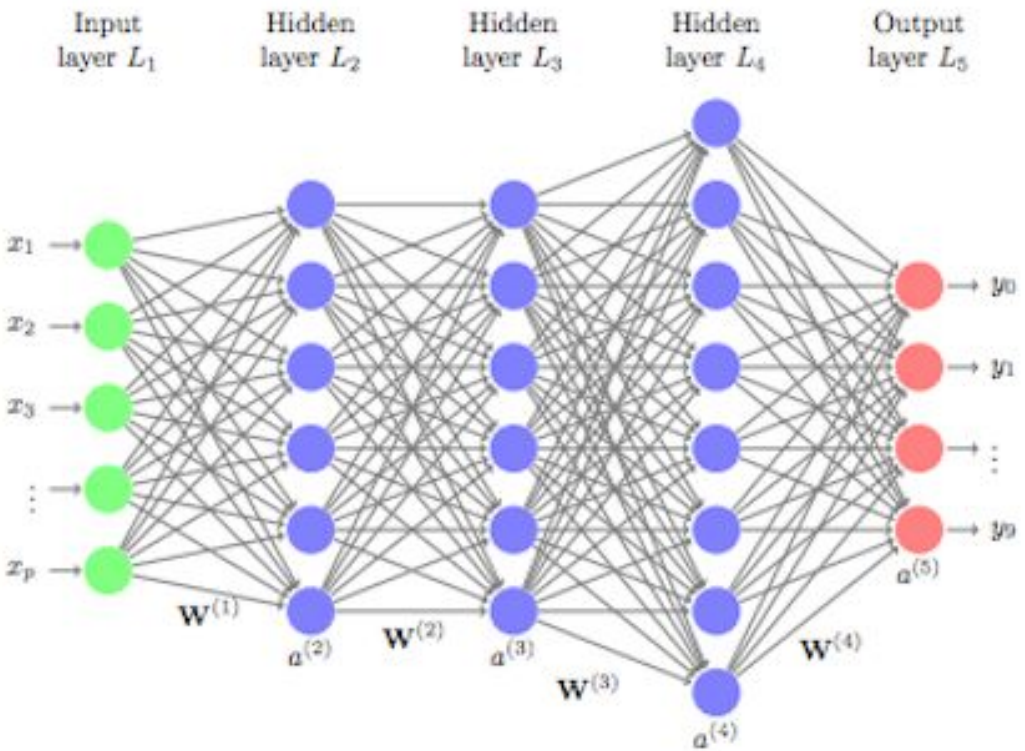
We can visualize a loss landscape for a model with three inputs $x=[x_1,x_2,x_3]$

The three axes are our inputs, and the graph shows how the loss changes with different inputs.



Back to Multi-layer Neural Networks

- Why can't we use the Perceptron rule to train this network?



Tuning weights by hand

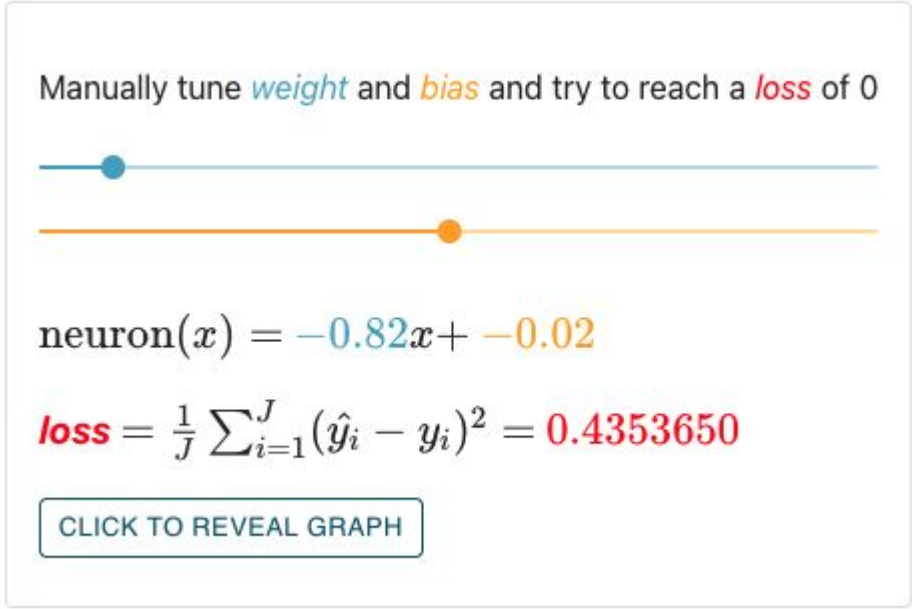
Open [Instapoll]

Navigate to https://xnought.github.io/backprop-explainer/

DON'T CLICK TO REVEAL GRAPH.

Use the sliders to try to minimize the loss.

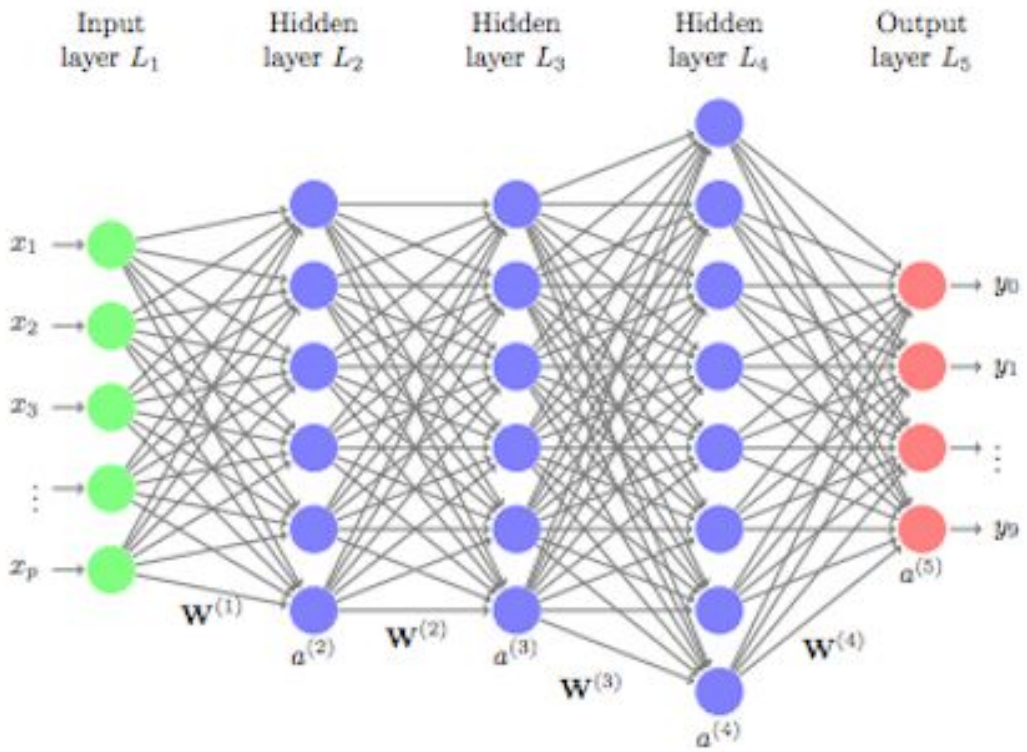
How many times did you have to move



Back to Multi-layer Neural Networks

Q: Why can't we use the Perceptron rule to train this network?

A: Imagine we update a weight on layer 4. then we go update a weight on layer 3. This messes up our optimization for layer 4!



Model Optimization

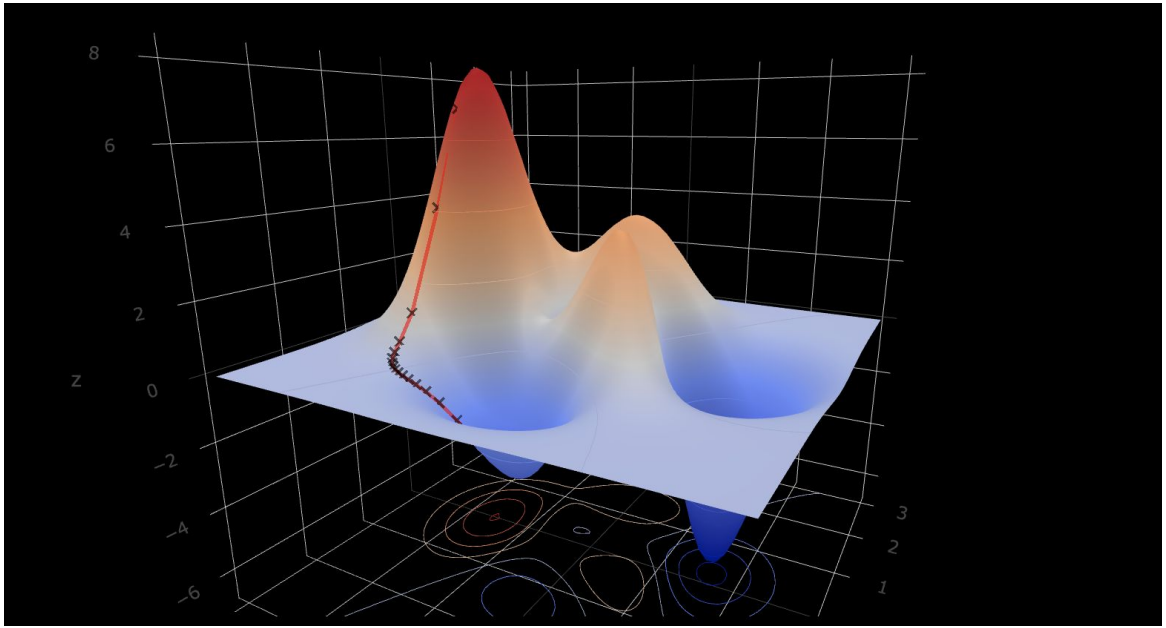
The goal in machine learning is to **minimize the loss function.**

Imagine you are lost in a mountain range without GPS or a compass. How do you get out?

Gradient Descent

https://blog.skz.dev/gradient-descent

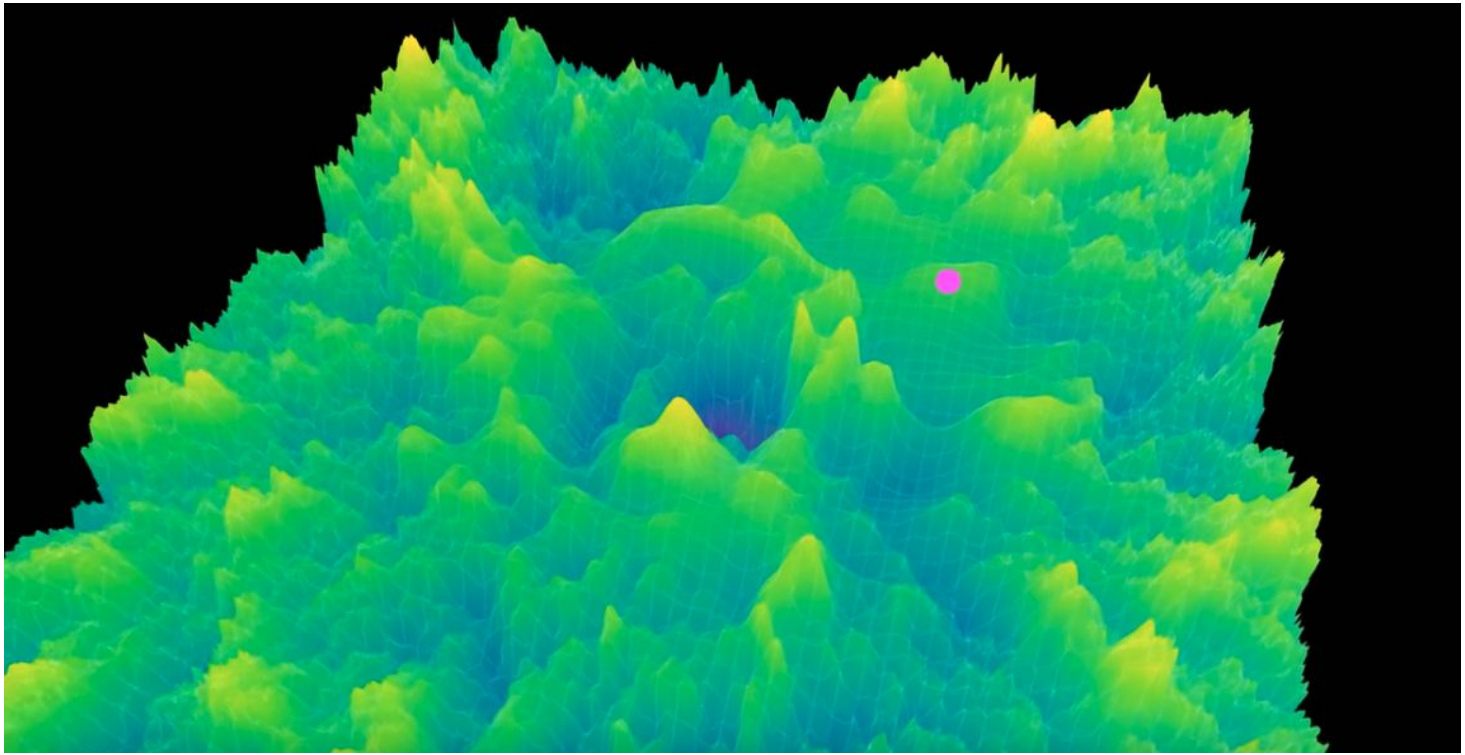
Training deep NNs works the same way.



Gradient Descent

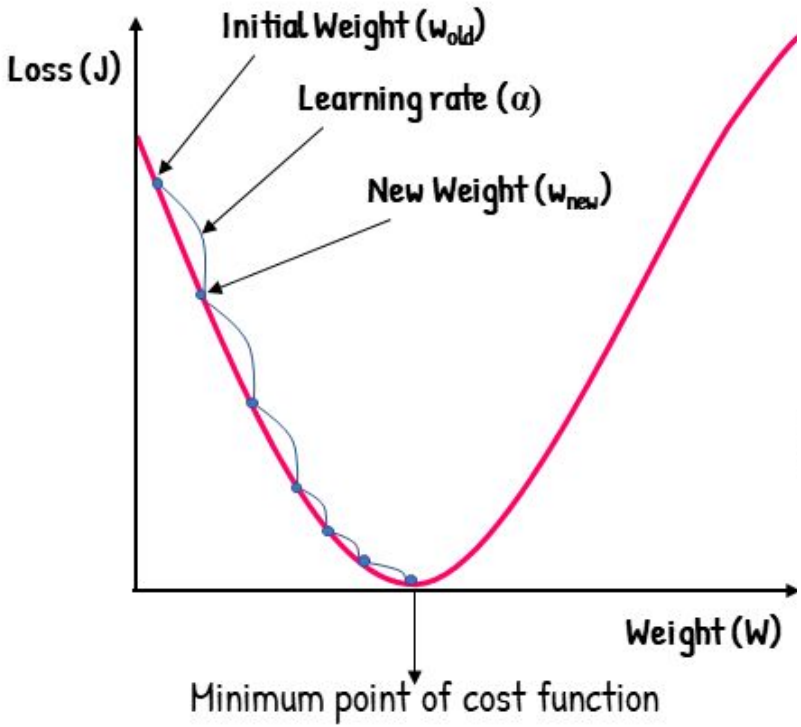
2-D visualization of the loss landscape for LLAMA-2

(LLAMA is Meta's open source LLM)



https://www.youtube.com/watch?v=NrO20Jb-hy0

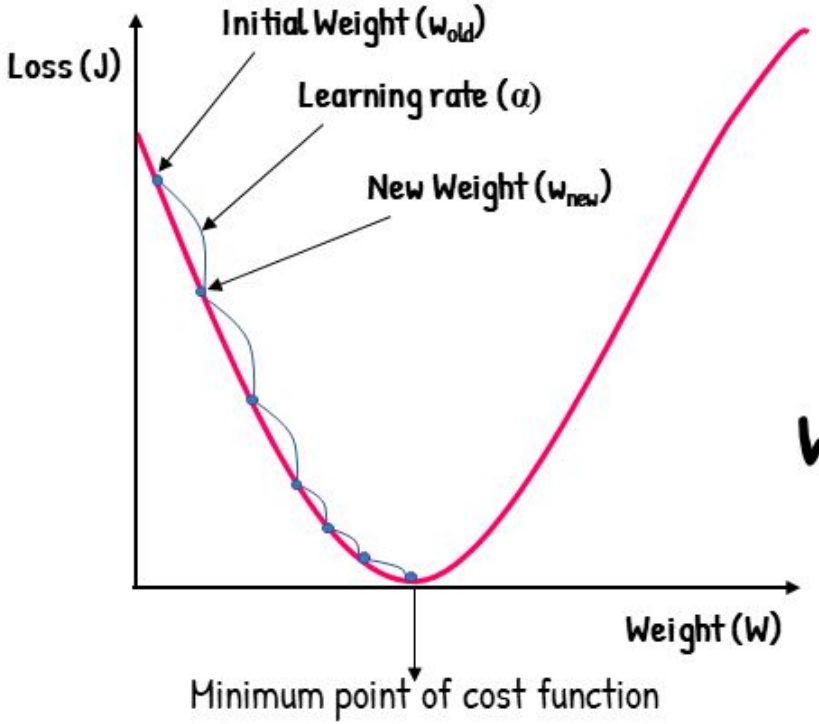
Gradient Descent



In practice, we can't see the whole loss curve at once. We can only see where we are.

Gradient descent uses the **derivative** (aka calculus) to find the slope of the curve at that point, so we can head downhill.

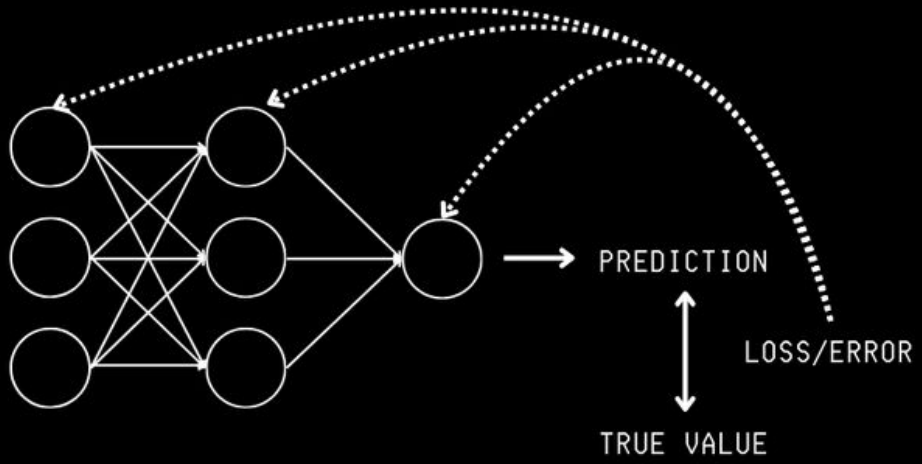
Gradient Descent



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

BACKPROPAGATION

AN ALGORITHM THAT COMPUTES THE GRADIENT OF THE LOSS FUNCTION WITH RESPECT TO EACH PARAMETER IN A NEURAL NETWORK AND UPDATES THEM TO MINIMIZE THE ERROR BY PROPAGATING THE ERRORS BACKWARDS FROM THE OUTPUT TO THE INPUT LAYER.



Backpropagation Intuition

Letter | Published: 09 October 1986

Learning representations by back-propagating errors

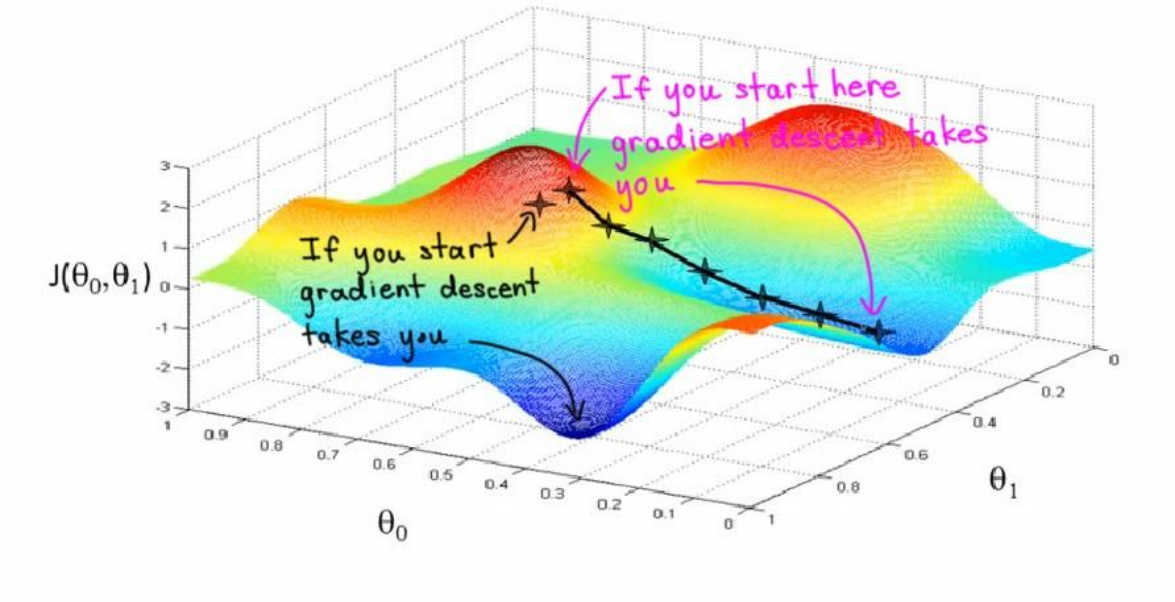
David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams

Nature **323**, 533–536 (1986) | Cite this article

Back propagation is a way of breaking up error to see the contribution of each weight in the network to the overall error. Maybe one weight decreased the error! don't change it. Maybe another weight increased the error a lot (and had a lot of downstream effects!) Change that one a lot.

Local Minima

https://www.youtube.com/watch?v=NrO20Jb-hy0



Linear Regression

The only difference is the activation function that we choose!

Linear regression uses the **sigmoid** activation function