DUKE
FUQUA
SCHOOL OF BUSINESS

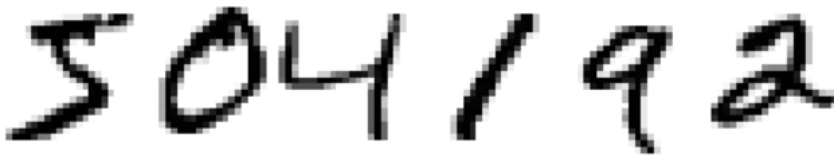
Sudipta Dasmohapatra
sd345@duke.edu

Neural Network and Deep Learning Workshop

Spring 2020

Neural Networks

Can you recognize these digits?



DUKE FUQUA SCHOOL OF BUSINESS

Neural Networks

Can you recognize these digits?

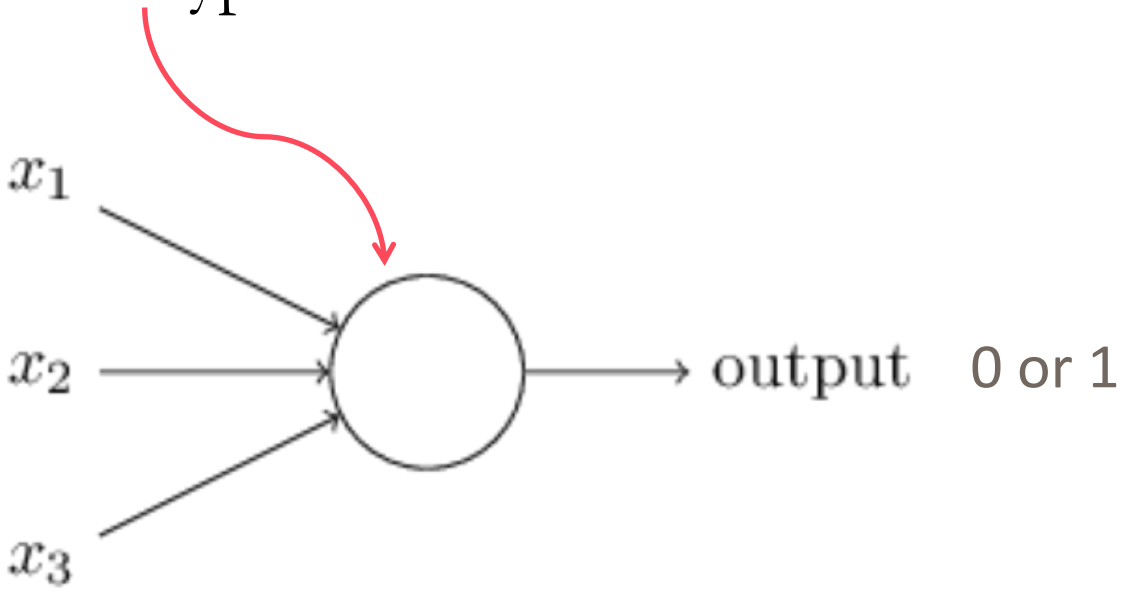
Training Examples

Neural Networks: Develops a system to to learn from these training examples: to automatically infer rules for recognizing these digits

DUKE FUQUA SCHOOL OF BUSINESS

What is a Neural Network?

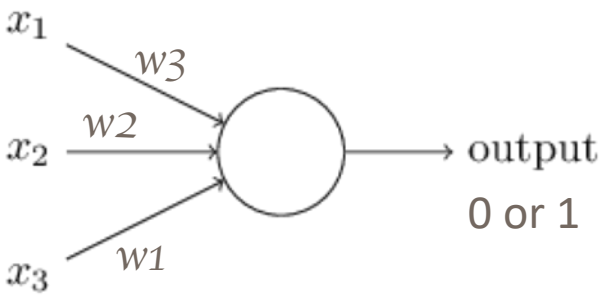
- Perceptron: Type of artificial neuron



DUKE FUQUA SCHOOL OF BUSINESS

What is a Neural Network?

- To compute output of a perceptron:



w1, w2 and w3 are weights (real numbers) that express importance of the respective inputs to the output

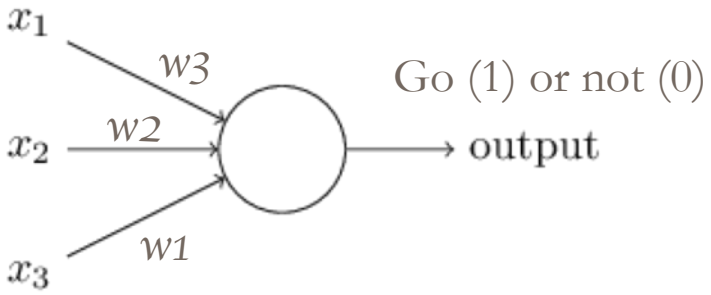
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

DUKE FUQUA SCHOOL OF BUSINESS

Neural Network: Example

You would like to go to the Durham beer festival

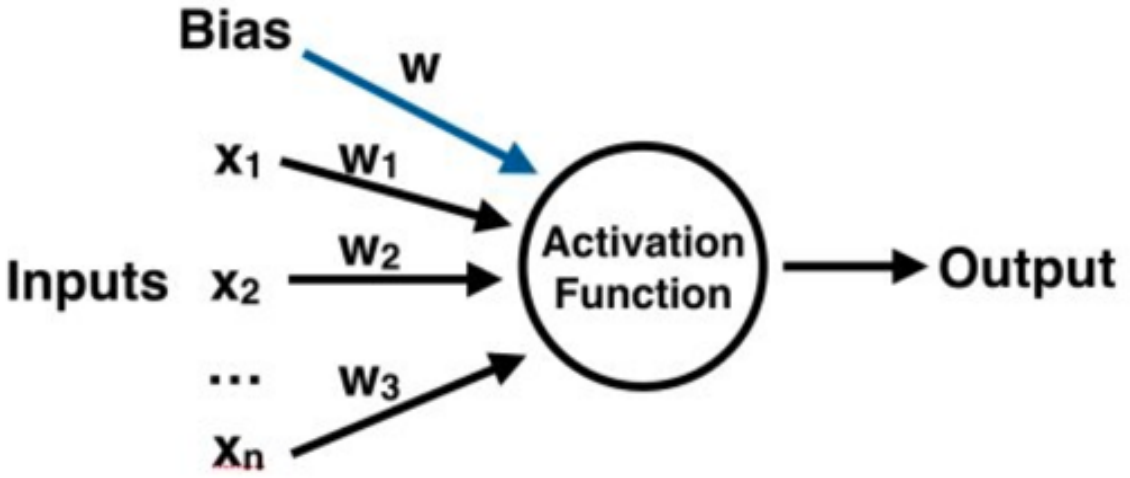
- You like beer and you want to decide whether or not to go to the festival. You might make your decision by weighting three factors:
  - Weather is good? (x1=1 if weather is good, otherwise=0)
  - Friend will go with you? (x2= 1 if your friend wants to go with you)
  - Festival near public transit? (you don't own a car) (x3=1 if festival near public transit)



Use weights w1, w2, w3 to indicate your preference for each of the inputs. If w1= larger then weather matter a lot to you much more than others. Choose a threshold for perceptron (e.g., 4 or 5) and then implement the desired decision model.

Neural Network: Perceptron

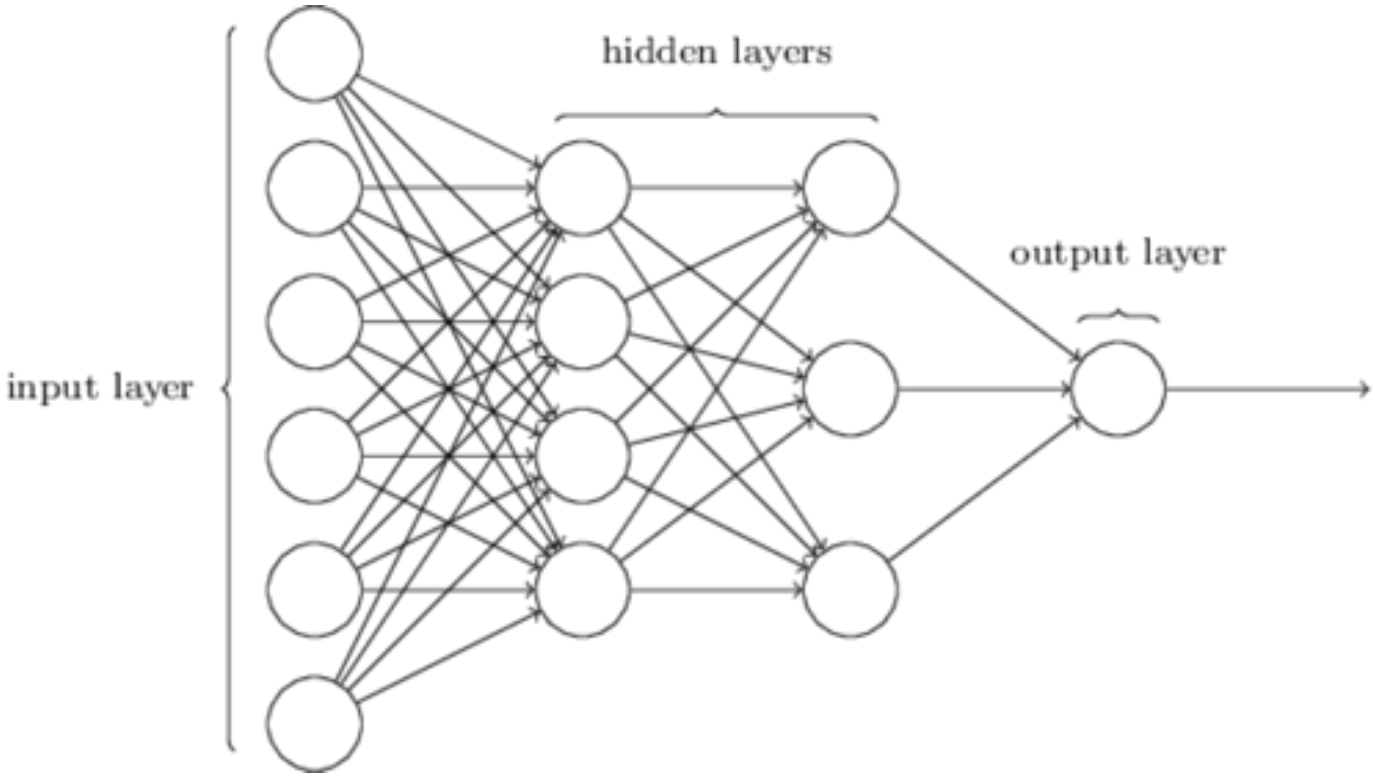
- A perceptron has one or more inputs, a bias, an activation function, and a single output
- The perceptron receives inputs, multiplies them by some weight, and then passes them into an activation function to produce an output



- Once we have the output from this model, it is compared to a known label and weights are adjusted accordingly
- Process is repeated multiple times (until an acceptable error rate)

Neural Networks

Neural Networks: Layers of perceptrons added together

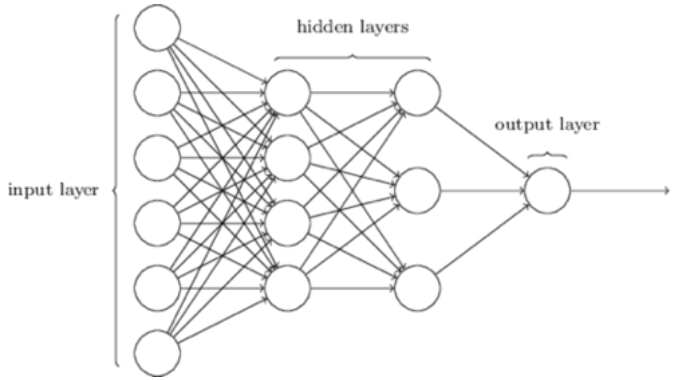


DUKE FUQUA SCHOOL OF BUSINESS

Neural Networks

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j + b \leq 0 \\ 1 & \text{if } \sum_j w_j x_j + b > 0 \end{cases}$$



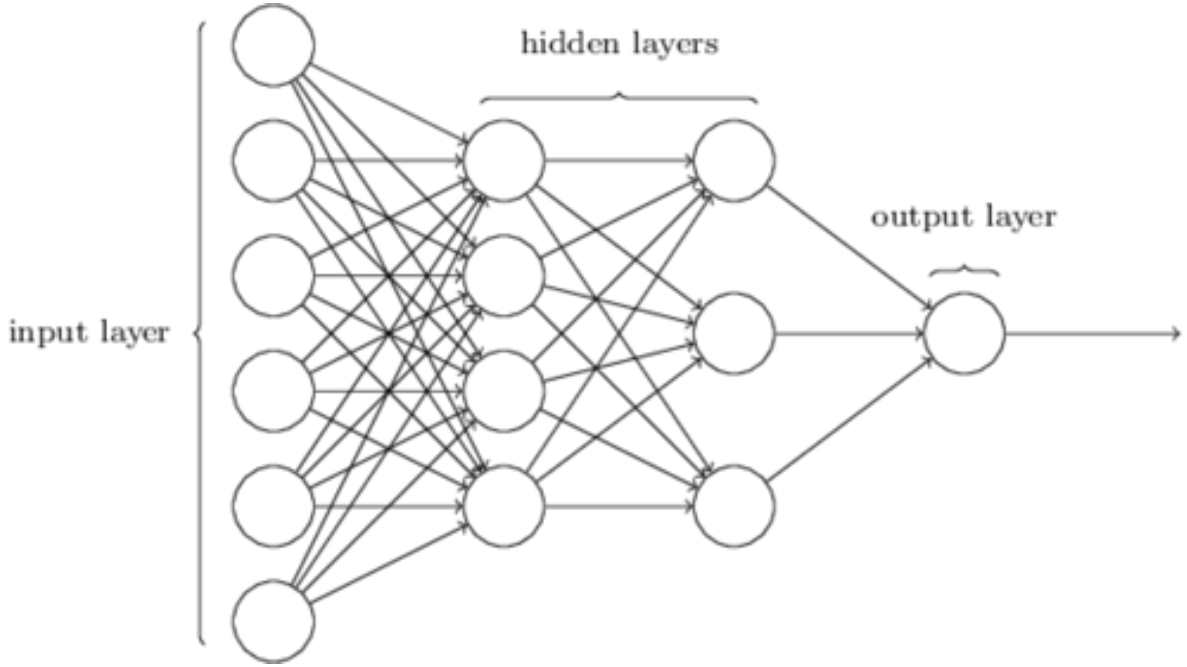
b≡−threshold

Bias is a measure of how easy it is to get the perceptron to *fire or get an output of 1* For a perceptron with a really big bias, it's extremely easy for the perceptron to output a 1. But if the bias is very negative, then it's difficult for the perceptron to output a 1.

DUKE FUQUA SCHOOL OF BUSINESS

Neural Networks

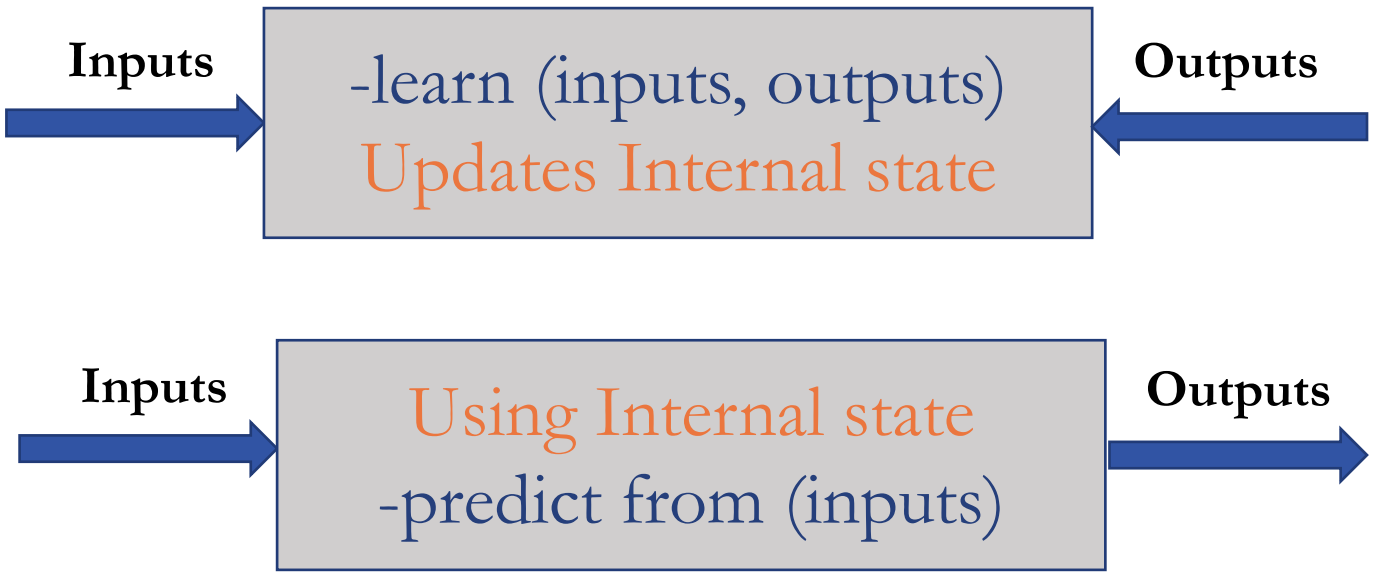
Neural networks, if sufficiently wide and deep can approximate any function



A hidden layer that is fully connected to the preceding layer is designated dense.

DUKE FUQUA SCHOOL OF BUSINESS

Supervised Neural Network



DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Desired Output
0	0
1	2
2	4
3	6
4	8

$$y = W * x$$

W is called the ***weights*** *of the network*

Model form = Linear (layers)

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Desired Output
0	0
1	2
2	4
3	6
4	8

1. Model Initialization: random

Random Initialization $y = 3 * x$
(3 is generated at random initially)

$$y = W * x$$

W is called the ***weights*** *of the network*

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Random Initialization $y = 3 * x$

Input	Actual Output of Model
0	0
1	3
2	6
3	9
4	12

$$y = W * x$$

W is called the ***weights*** *of the network*

2. Forward Propagate: Start from input, pass them through network layer and calculate the actual output (prediction)

Forward Propagation
Input>Network>Output

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Actual Output of Model	Desired Output
0	0	0
1	3	2
2	6	4
3	9	6
4	12	8

3. Loss Function: Performance metric on how the NN manages to reach its goal of generating outputs close to desired values

Loss = Desired output – Actual output (predicted)

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Actual Output of Model	Desired Output	Absolute Error
0	0	0	0
1	3	2	1
2	6	4	2
3	9	6	3
4	12	8	4
Total:	..	..	10

Loss = Absolute value (Desired output – Actual output)

Loss = How much precision do we lose (minimize this)

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Actual Output of Model	Desired Output	Absolute Error	Squared Error
0	0	0	0	0
1	3	2	1	1
2	6	4	2	4
3	9	6	3	9
4	12	8	4	16
Total:	..	..	10	30

Loss = Sum of squares of absolute errors

Loss = How much precision do we lose when the real output is replaced by the actual predicted?

DUKE FUQUA SCHOOL OF BUSINESS

Loss Function in Neural Networks

- Goal is minimizing the loss function (to reach as close as possible to 0) when we take the total of the sum of squared errors

We can transform this to an optimization process that aims to minimize the loss function

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

4. Optimization: We can use any optimization technique that modifies the internal weights of neural networks to minimize loss function

$$y = W * x$$

One Method: Take values of W from -100 to +100 at a step of 0.001 and run model to get which W has the smallest sum of squared errors over the dataset

What if the model has large number of parameters? How about precision?

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

4. Optimization: We can use any optimization technique that modifies the internal weights of neural networks to minimize loss function

Differentiation: Derivative of the loss function (gives the rate at which this function changes its values at a point)

Effect of Derivative: How much the total error will change if we change the internal weight of the neural network with a certain small value (δW).

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Desired Output	W=3 Actual	sq error(3)	W=3.0001 Actual	sq error (3.001)
0	0	0	0	0	0
1	2	3	1	3.0001	1.0002
2	4	6	4	6.0002	4.0008
3	6	9	9	9.0003	9.0018
4	8	12	16	12.0004	16.0032
Total:	..	..	30	..	30.006

$$\delta W = 0.0001$$

Error is increased by 0.006

We really care about the rate of change of error relative to change in weight

DUKE FUQUA SCHOOL OF BUSINESS

Simple Example: Neural Network

Input	Desired Output	W=3 Actual	sq error(3)	W=3.0001 Actual	sq error (3.001)
0	0	0	0	0	0
1	2	3	1	3.0001	1.0002
2	4	6	4	6.0002	4.0008
3	6	9	9	9.0003	9.0018
4	8	12	16	12.0004	16.0032
Total:	..	..	30	..	30.006

$$\delta W = 0.0001$$

Derivative = rate of change in total error due change in weight
= 0.006/0.0001 = 60 times increase with increase in W by 0.0001

DUKE FUQUA SCHOOL OF BUSINESS

Loss Function: Derivative

Derivative: rate of change of error due to change in weight

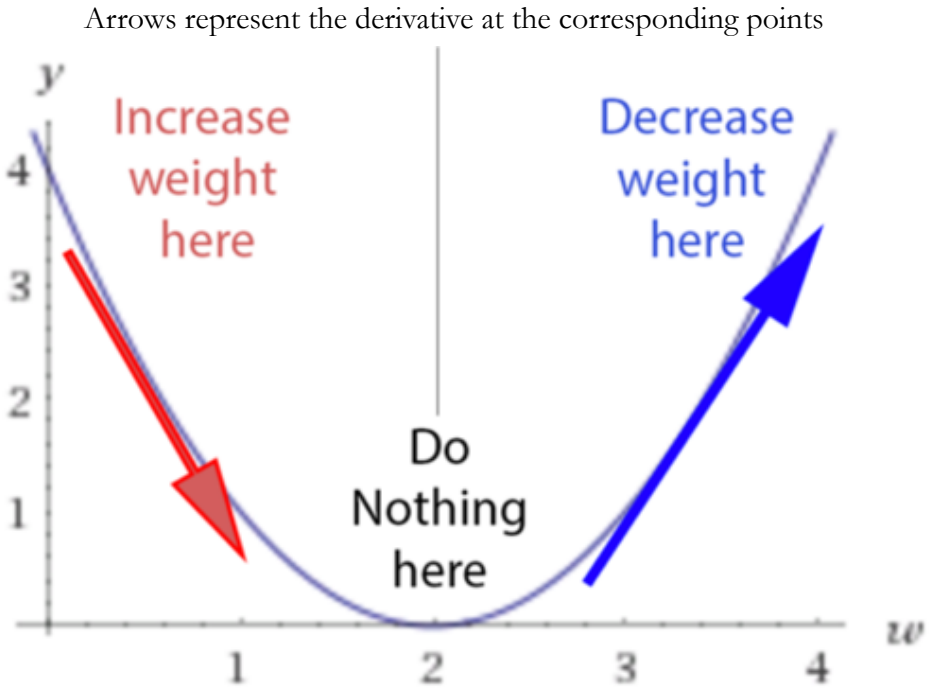
Loss function:

- If w=2, we have a loss of 0 (loss =0, derivative =0)= perfect model
- If w>2 (positive loss function), derivative = +ve, increase in weight will increase loss even more
- If w<2 (positive loss function), derivative = -ve, increase in weight will decrease the loss function

DUKE FUQUA SCHOOL OF BUSINESS

Loss Function

Let's check the derivative

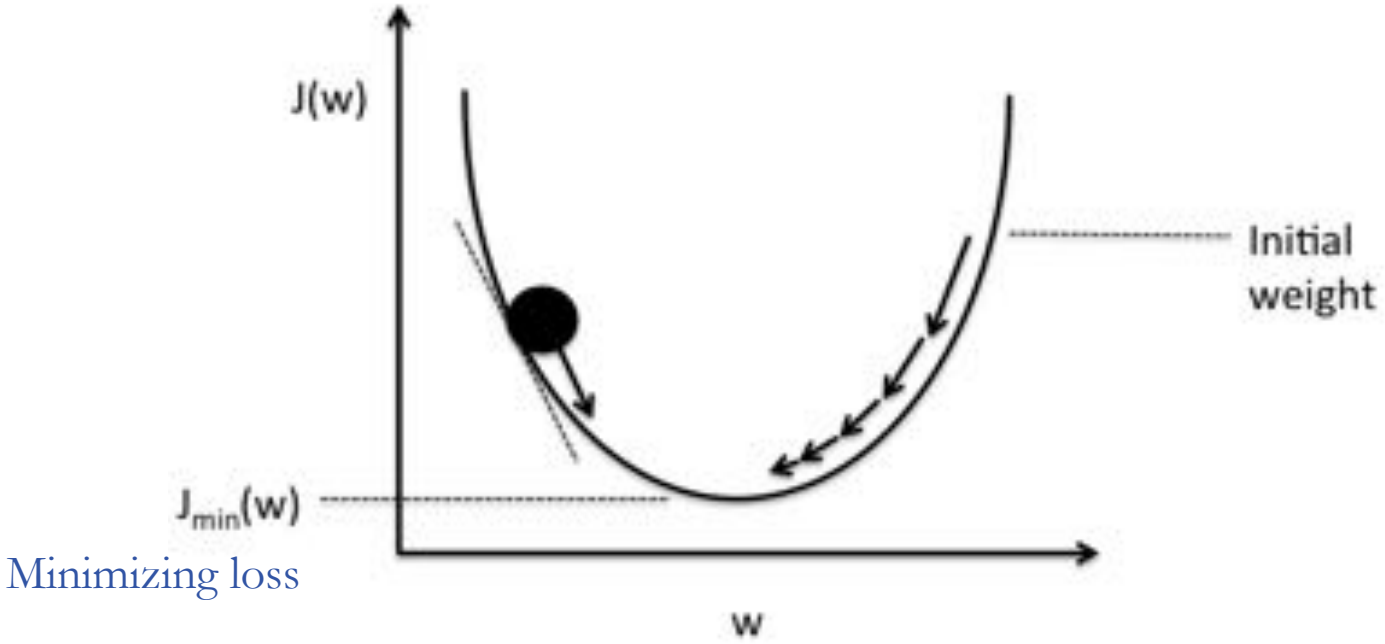


- If it is positive, meaning the error increases if we increase the weights > then we should decrease the weight (w)

- If it's negative, meaning the error decreases if we increase the weights, then we should increase the weight (w)
- If it's 0, we do nothing, we reach our stable point

DUKE FUQUA SCHOOL OF BUSINESS

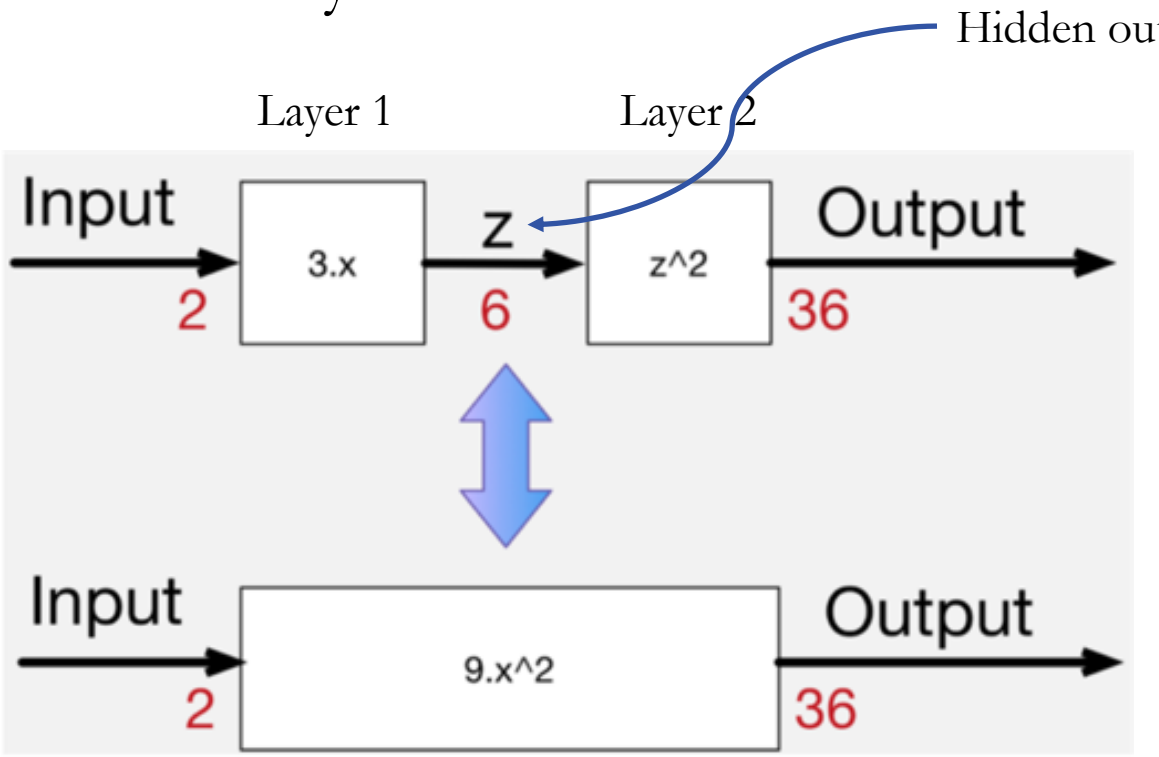
Gradient Descent



Minimizing loss

Back Propagation

5. More layers = Reaches more variations in the functionality of the network

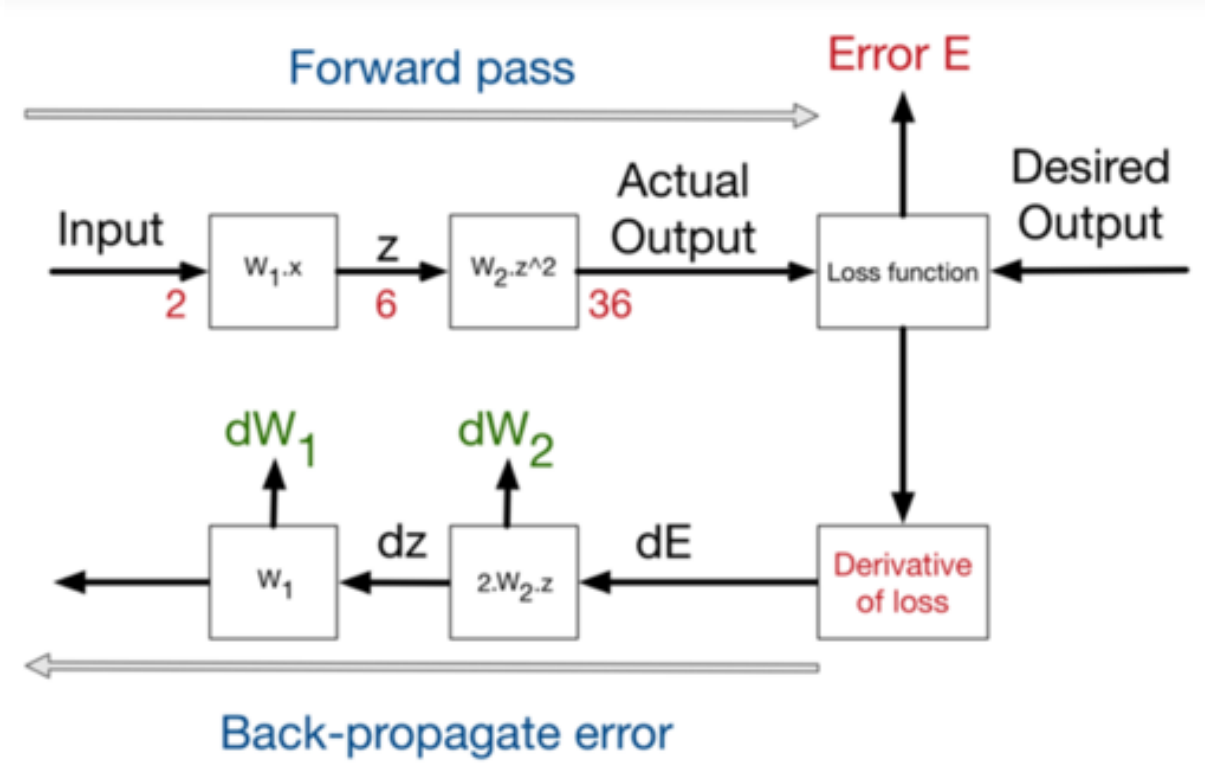


Composed functions are difficult =
Derivatives have to be calculated for every composition (**not scalable and error prone**)

DUKE FUQUA SCHOOL OF BUSINESS

Back Propagation

Input > Forward calls > Loss function > Derivative > Backpropagation



DUKE FUQUA SCHOOL OF BUSINESS

Weight Update

6. Weight update: The derivative is the rate of which the error changes relative to the weight changes

New weight = old weight – Derivative Rate * Learning Rate

Learning rate: constant (usually very small) in order to force the weight to get updated very smoothly and slowly

DUKE FUQUA SCHOOL OF BUSINESS

Weight Update

New weight = old weight – Derivative Rate * Learning Rate

Learning rate: constant (usually very small) in order to force the weight to get updated very smoothly and slowly

Derivative rate =+ ve (an increase in weight will increase the error) > new weight = smaller

Derivative rate = – ve (an increase in weight will decrease the error) > new weight = larger

Derivative = 0 (we are in a stable minimum) >No update on weights

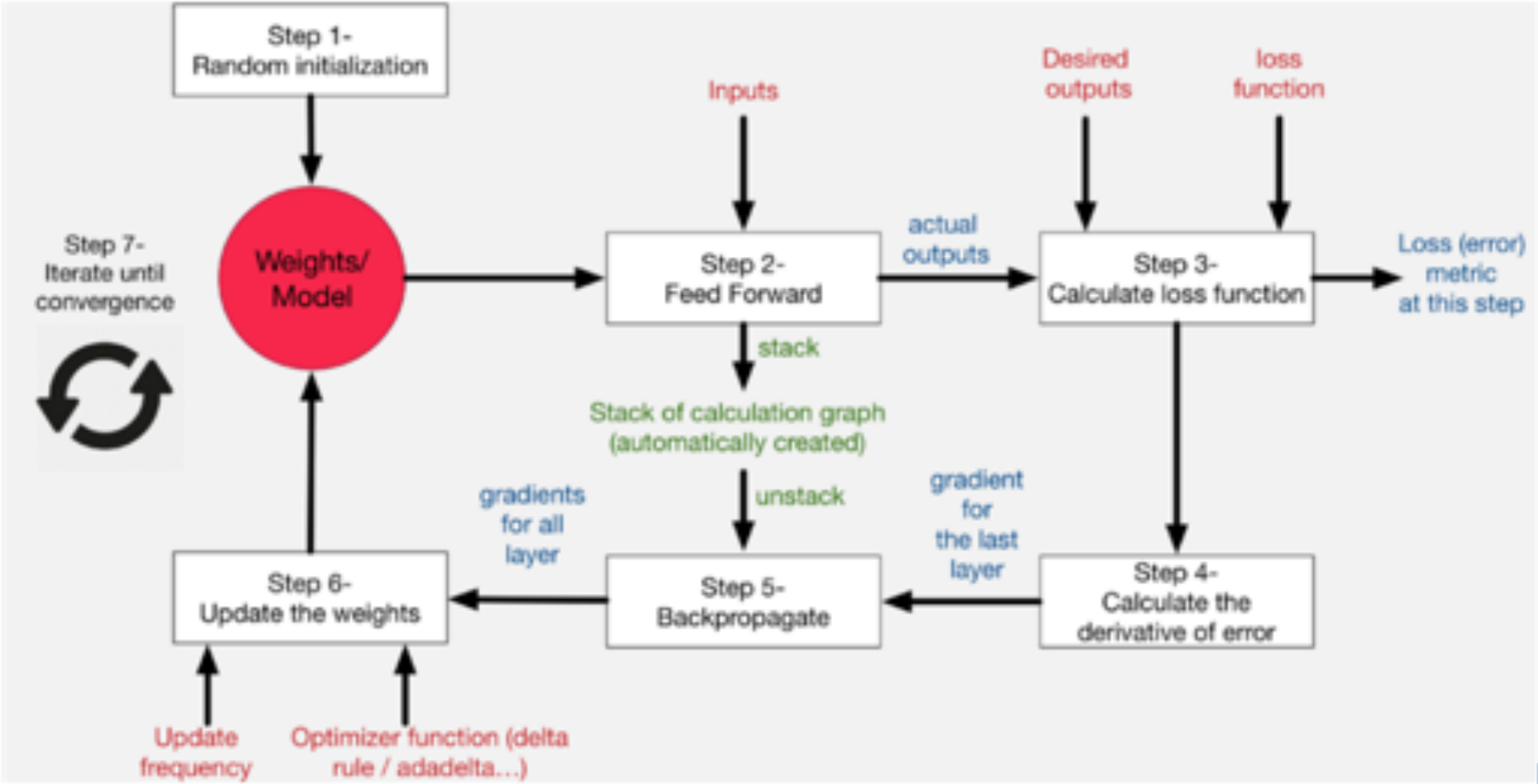
DUKE FUQUA SCHOOL OF BUSINESS

Iterate until Convergence

7. Iteration until convergence: Since weights are updated with small delta at a time, it takes several iterations in order to learn

- How many iterations are needed to converge (depends on)
  - How strong is the learning rate? High learning rate = faster learning, but with higher chance of instability
  - Meta-parameters of the network (how many layers, how complex the non-linear functions are). The more variables = the more it takes time to converge (but higher precision)
  - Optimization methods used: some weight update rules are proven to be faster than others
  - Random initialization of network
  - Quality of the training set. If input and output has no correlation, the NN will not learn a random correlation

Learning Process on Neural Networks



DUKE FUQUA SCHOOL OF BUSINESS

Thanks

sd345@duke.edu

DUKE FUQUA SCHOOL OF BUSINESS