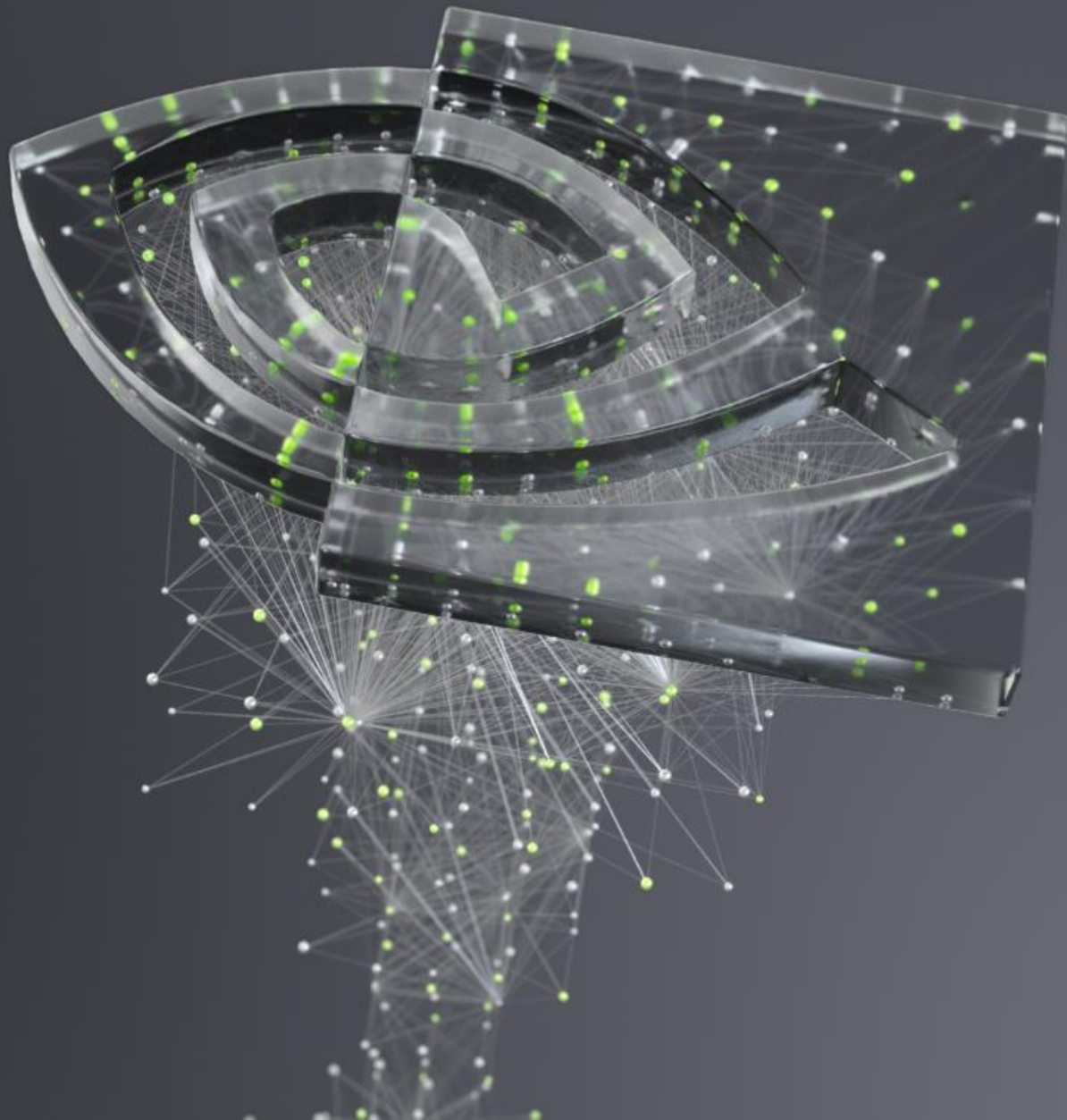




DEEP  
LEARNING  
INSTITUTE

# FUNDAMENTALS OF DEEP LEARNING

Part 2: How a Neural Network Trains



---

# AGENDA

Part 1: An Introduction to Deep Learning

Part 2: How a Neural Network Trains

Part 3: Convolutional Neural Networks

Part 4: Data Augmentation and Deployment

Part 5: Pre-trained Models

Part 6: Advanced Architectures

# AGENDA – PART 2

- Recap
- A Simpler Model
- From Neuron to Network
- Activation Functions
- Overfitting
- From Neuron to Classification


# RECAP OF THE EXERCISE

What just happened?


Loaded and visualized our data




Edited our data (reshaped, normalized, to categorical)



Created our model



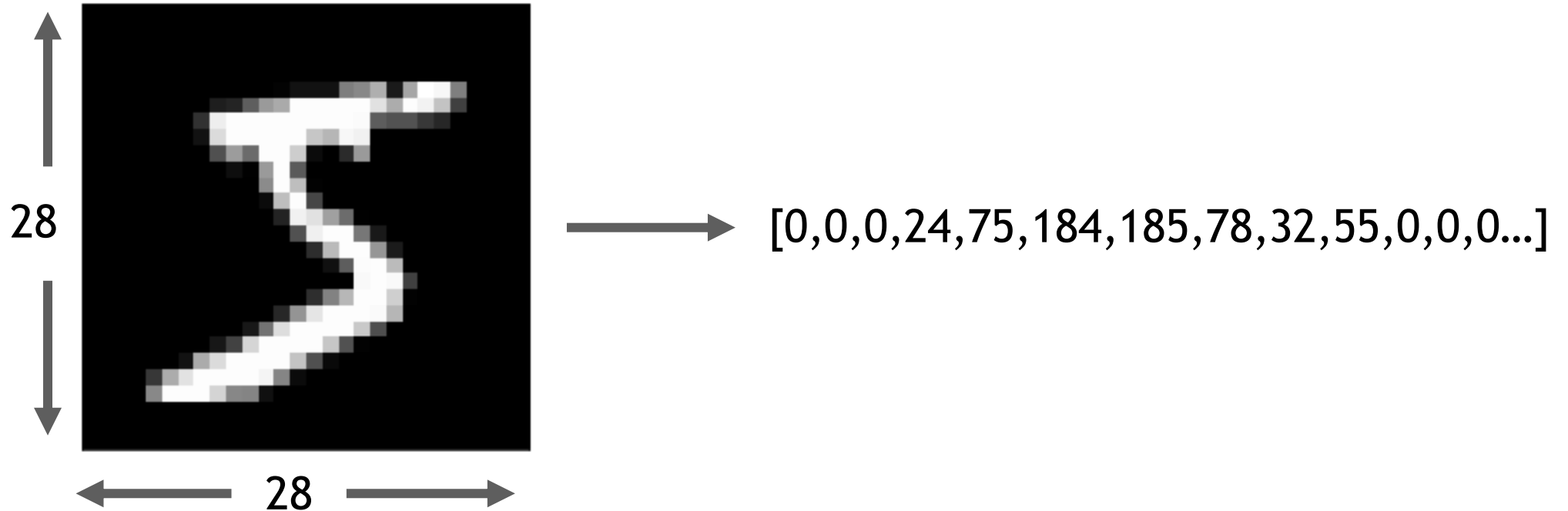
Compiled our model



Trained the model on our data

# DATA PREPARATION

Input as an array

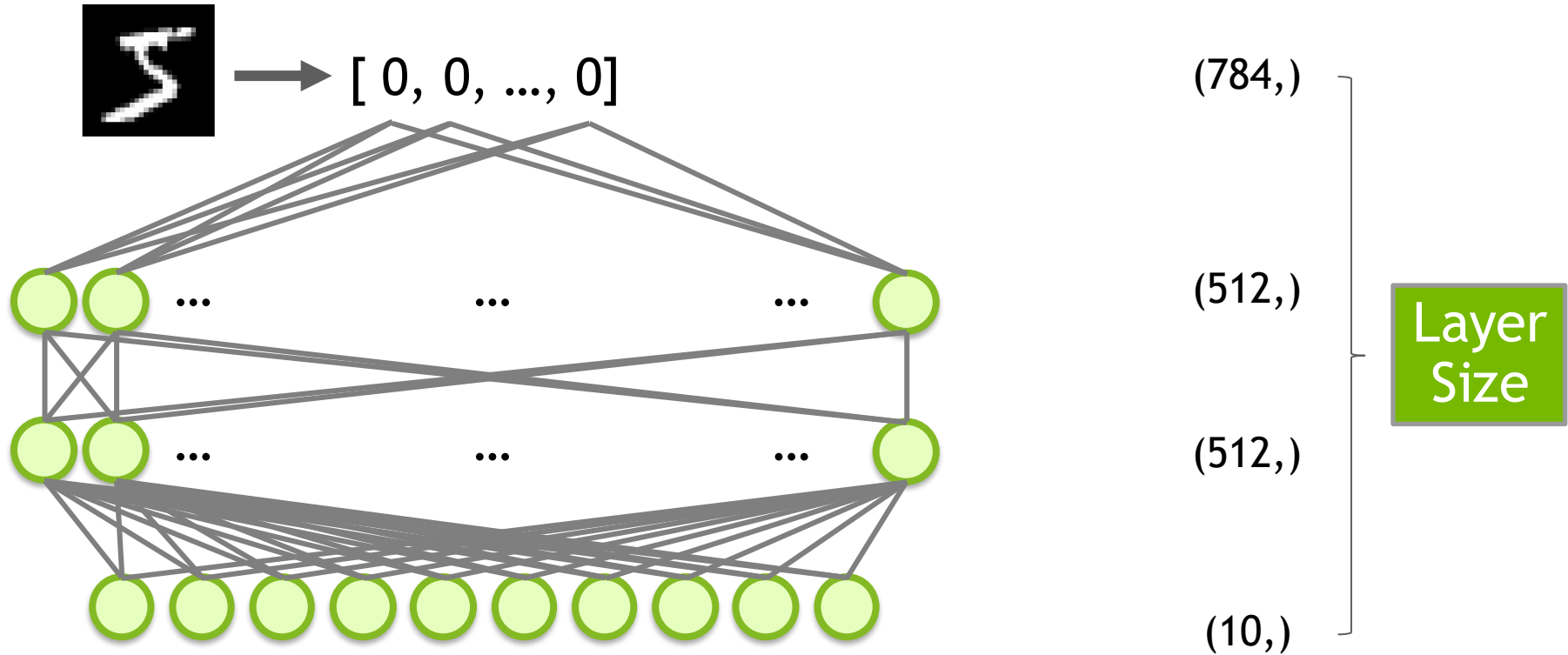


# DATA PREPARATION

Targets as categories

0	→	[1,0,0,0,0,0,0,0,0,0]
1	→	[0,1,0,0,0,0,0,0,0,0]
2	→	[0,0,1,0,0,0,0,0,0,0]
3	→	[0,0,0,1,0,0,0,0,0,0]
	•	
	•	
	•	

# AN UNTRAINED MODEL





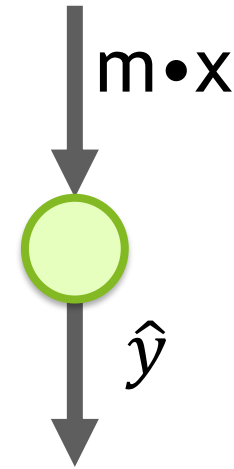
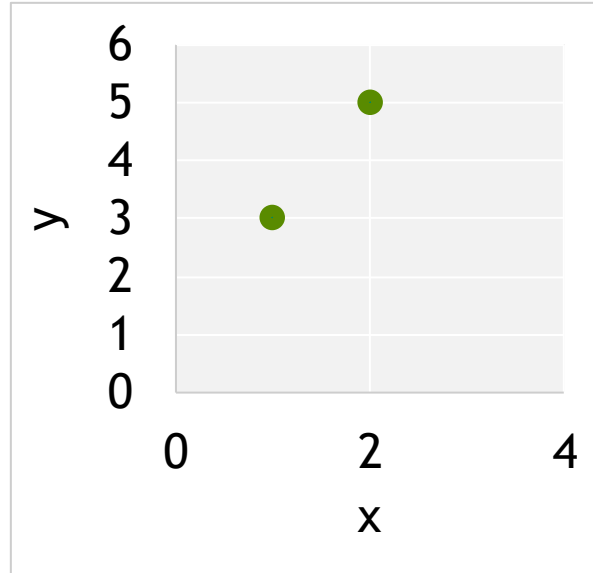
A SIMPLER MODEL



## A SIMPLER MODEL

$$y = mx + b$$

x	y
1	3
2	5



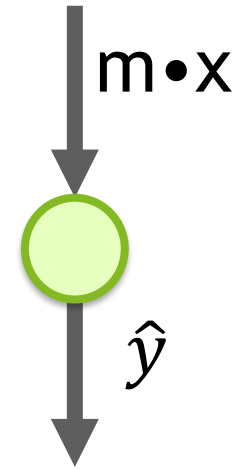
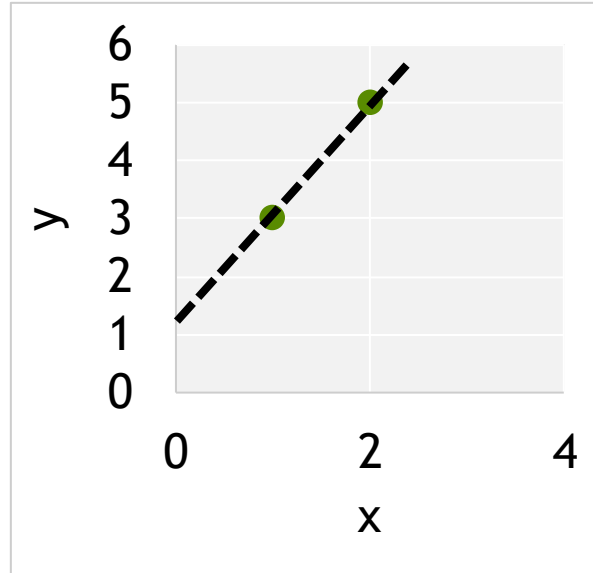
$$m = ?$$

$$b = ?$$

## A SIMPLER MODEL

$$y = mx + b$$

x	y
1	3
2	5



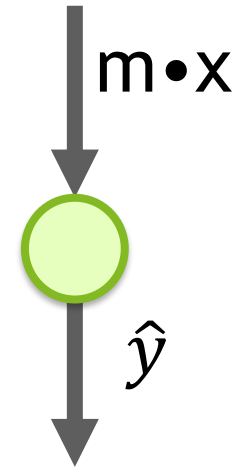
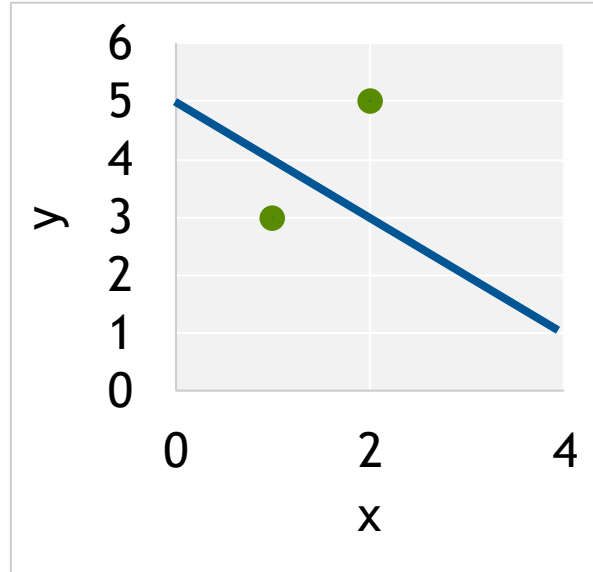
$$m = ?$$

$$b = ?$$

## A SIMPLER MODEL

$$y = mx + b$$

x	y	$\hat{y}$
1	3	4
2	5	3



Start  
Random

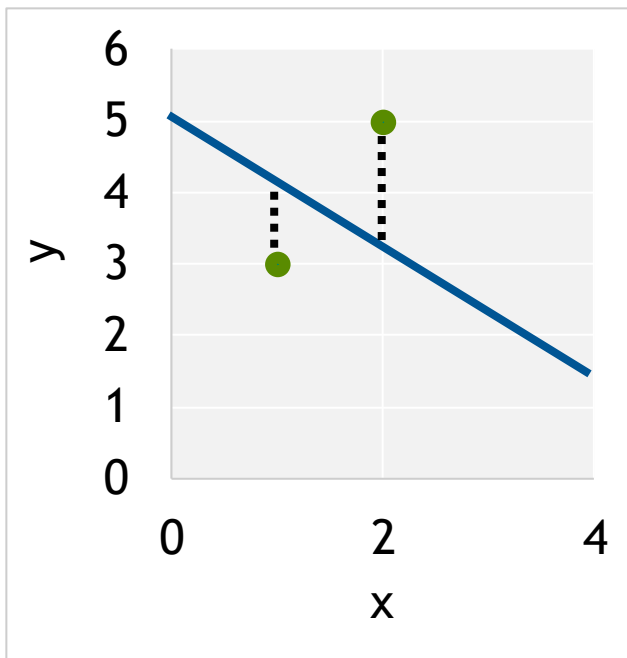
$$m = -1$$

$$b = 5$$

## A SIMPLER MODEL

$$y = mx + b$$

x	y	$\hat{y}$	$err^2$
1	3	4	1
2	5	3	4
MSE =			2.5
RMSE =			1.6



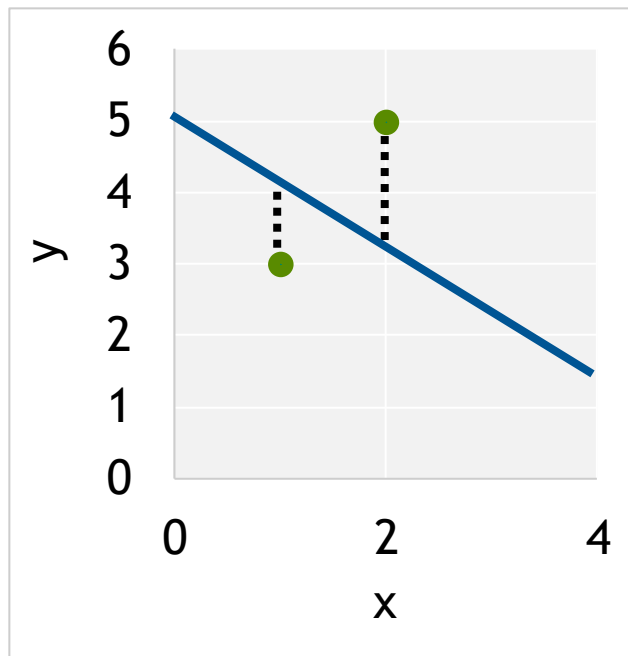
$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

## A SIMPLER MODEL

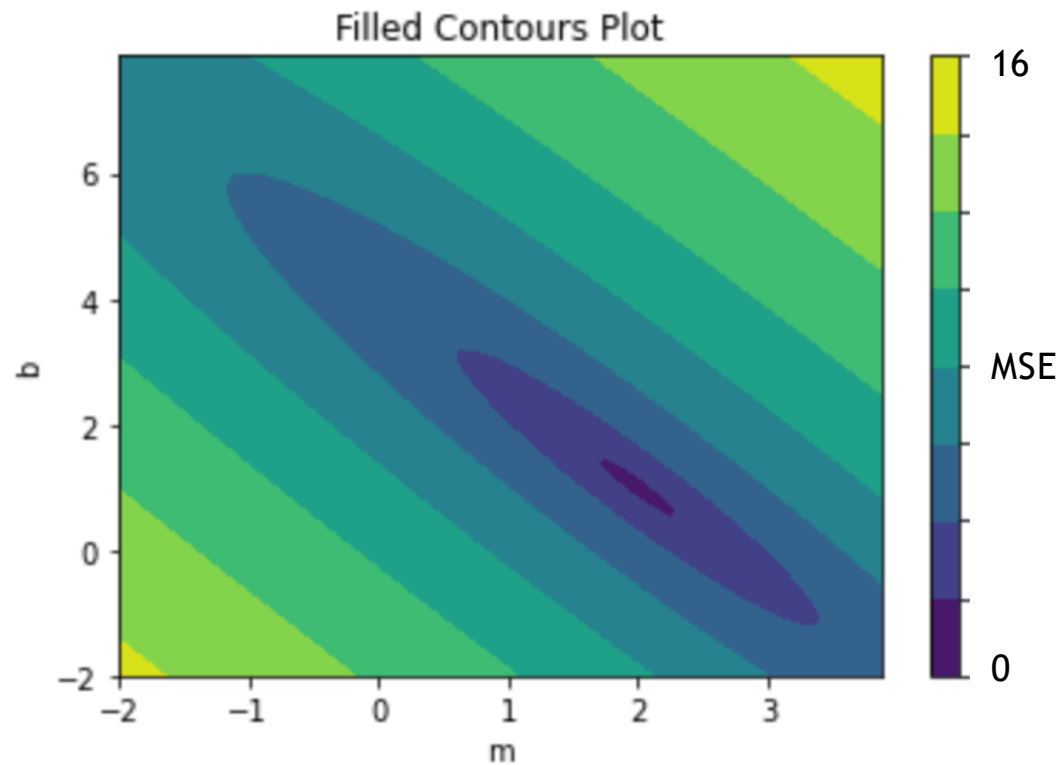
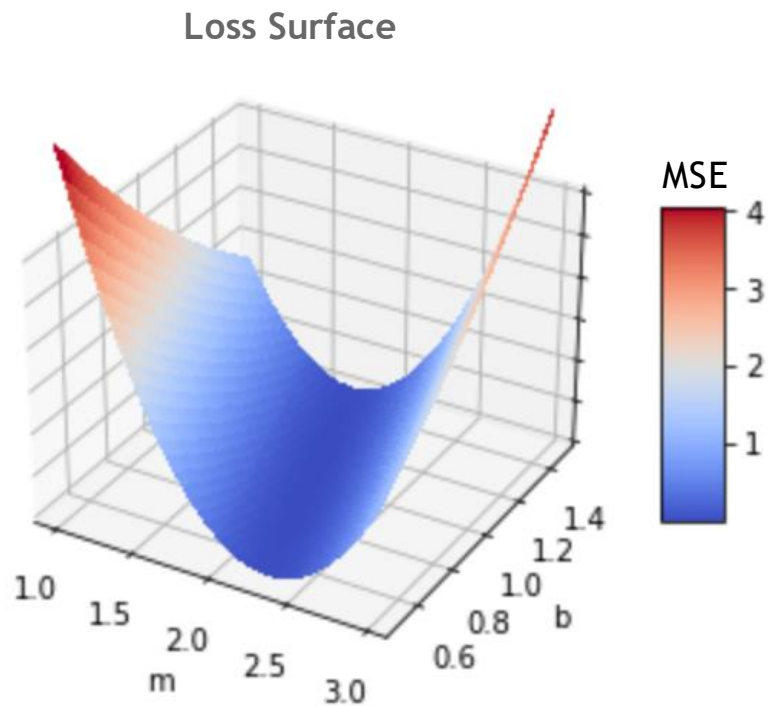
$$y = mx + b$$

x	y	$\hat{y}$	$err^2$
1	3	4	1
2	5	3	4
MSE =			2.5
RMSE =			1.6

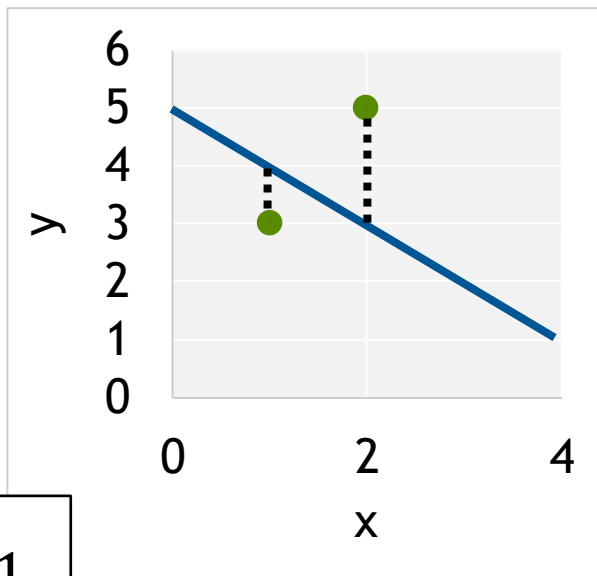


```
1 data = [(1, 3), (2, 5)]
2 m = -1
3 b = 5
4
5
6 def get_rmse(data, m, b):
7     """Calculates Mean Square Error"""
8     n = len(data)
9     squared_error = 0
10    for x, y in data:
11        # Find predicted y
12        y_hat = m*x+b
13        # Square difference between
14        # prediction and true value
15        squared_error += (
16            y - y_hat)**2
17    # Get average squared difference
18    mse = squared_error / n
19    # Square root for original units
20    return mse **.5
21
```

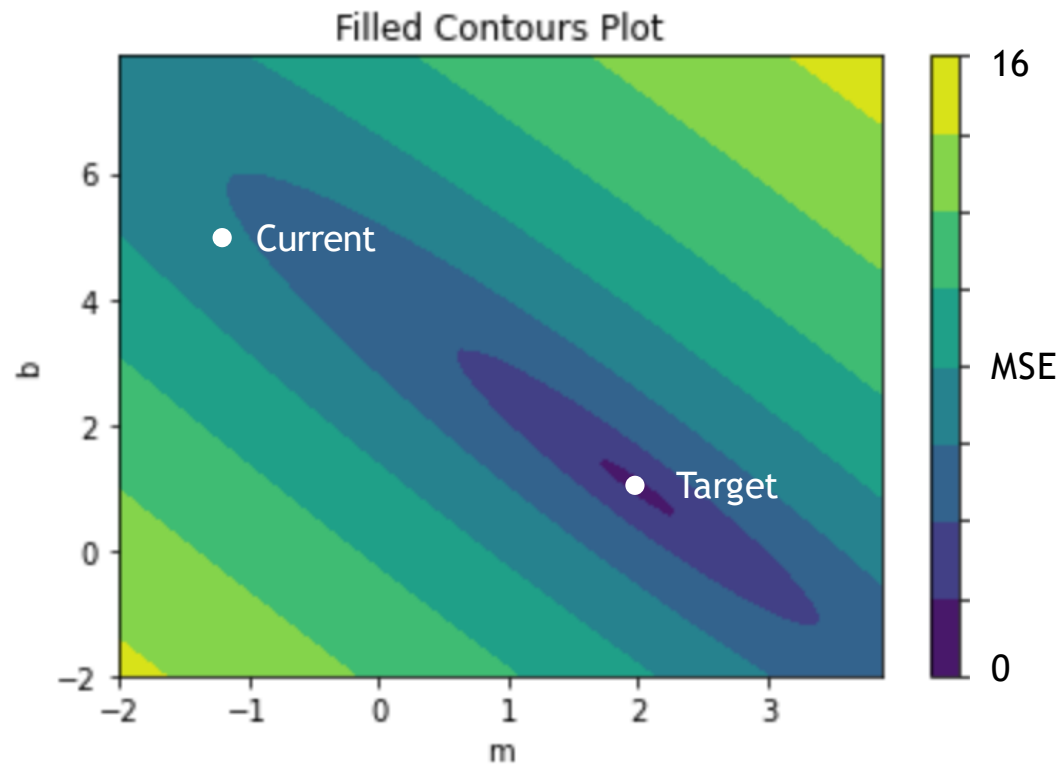
# THE LOSS CURVE



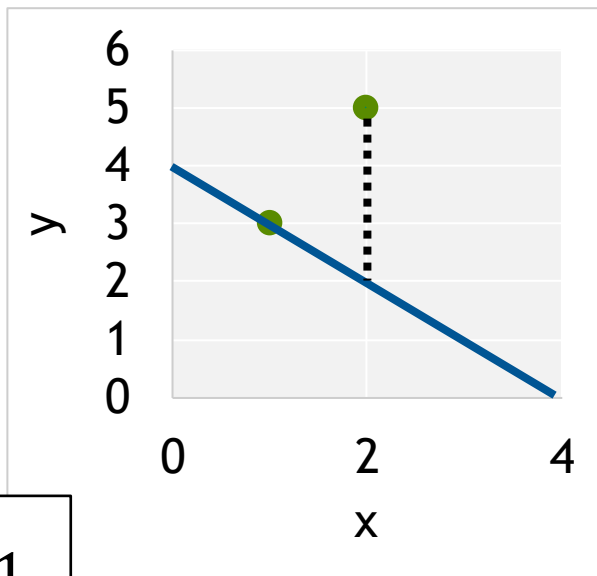
# THE LOSS CURVE



$$m = -1$$
$$b = 5$$

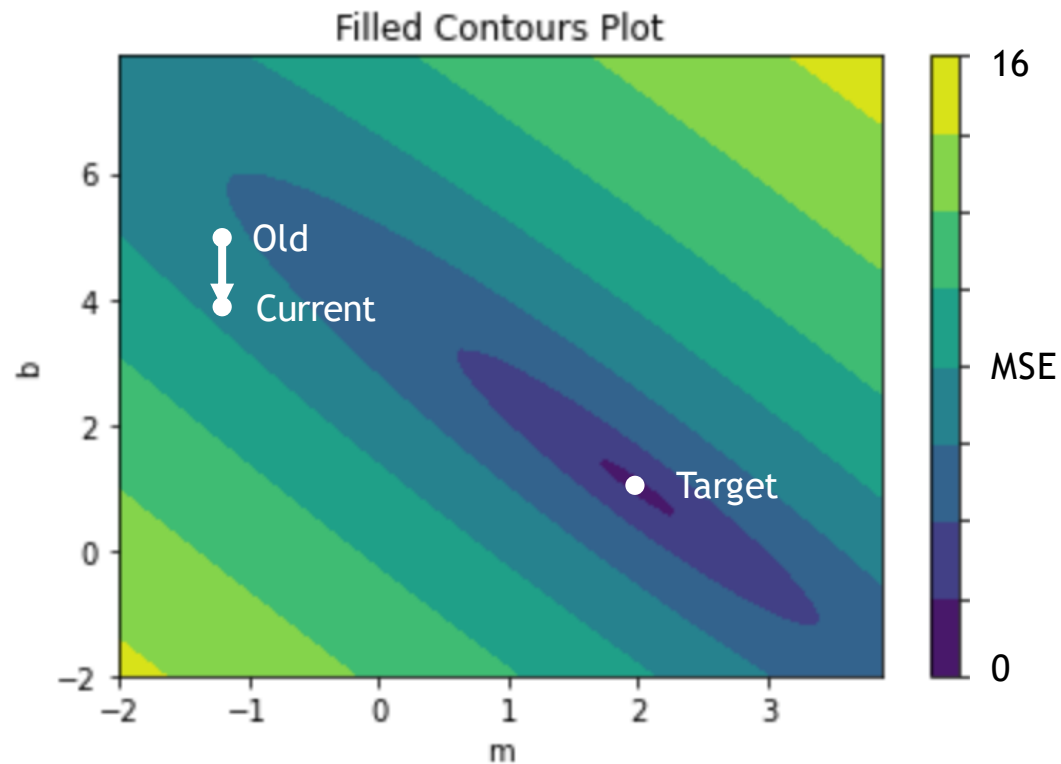


# THE LOSS CURVE



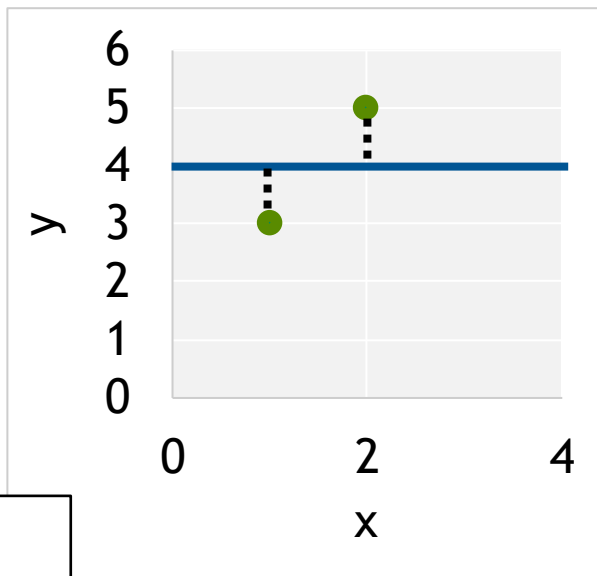
$$m = -1$$

$$b = 4$$



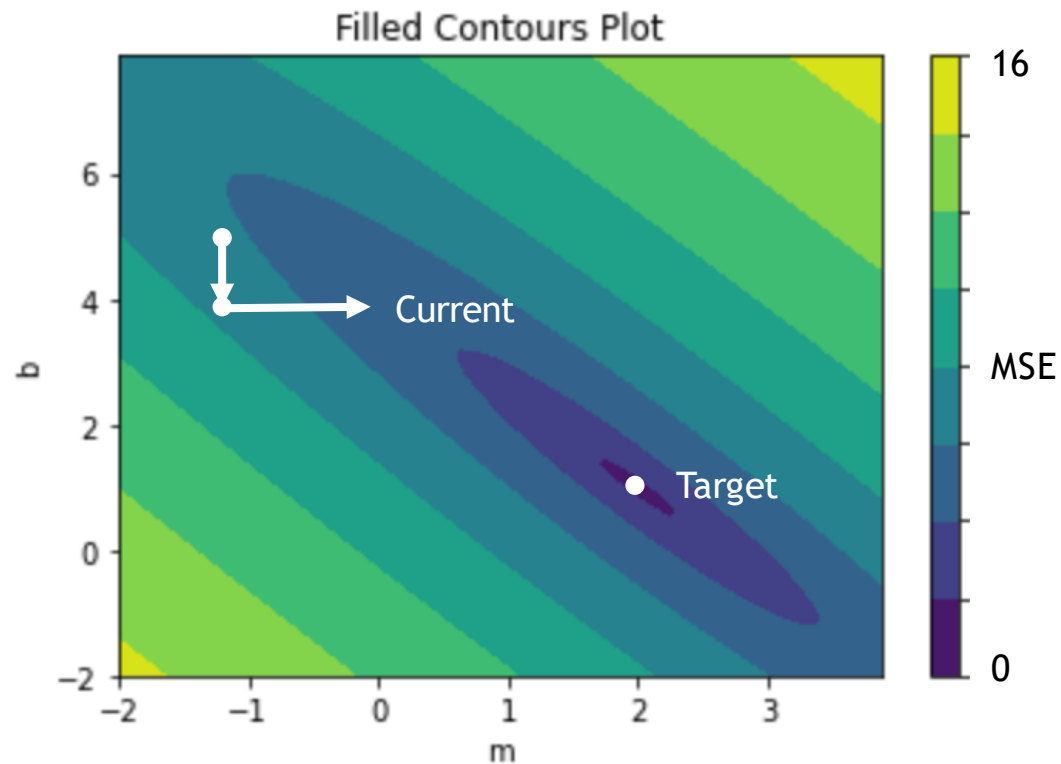


# THE LOSS CURVE



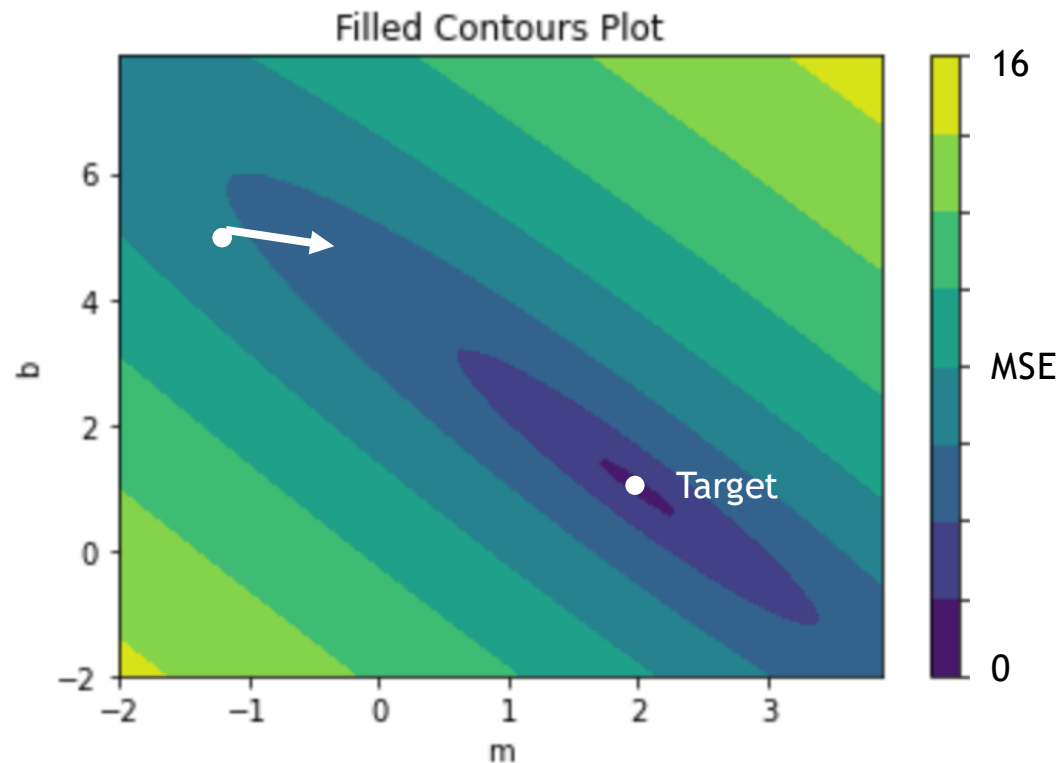
$$m = 0$$

$$b = 4$$



# THE LOSS CURVE

The Gradient	Which direction loss decreases the most
$\lambda$ : The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters

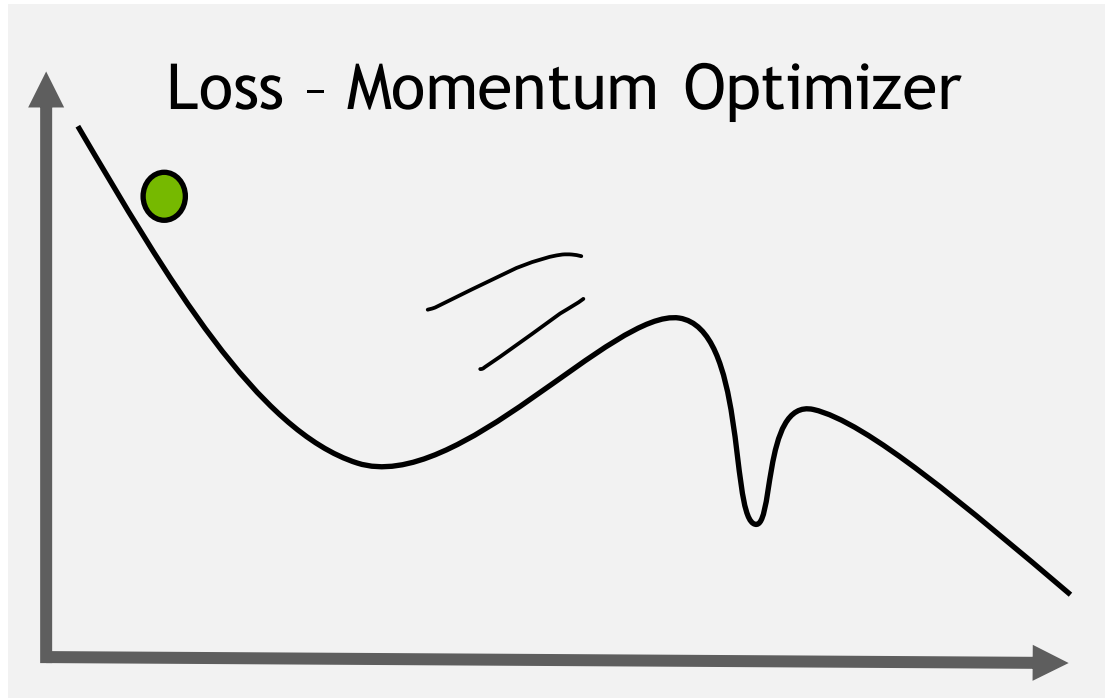


# THE LOSS CURVE

The Gradient	Which direction loss decreases the most
$\lambda$ : The learning rate	How far to travel
Epoch	A model update with the full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters



# OPTIMIZERS

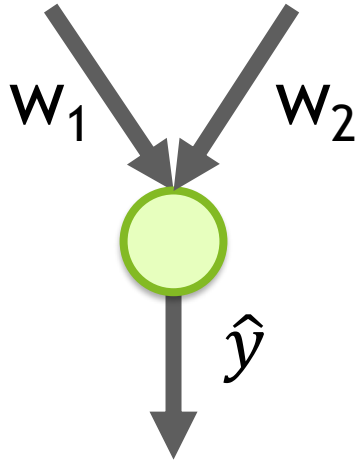


- Adam
- Adagrad
- RMSprop
- SGD



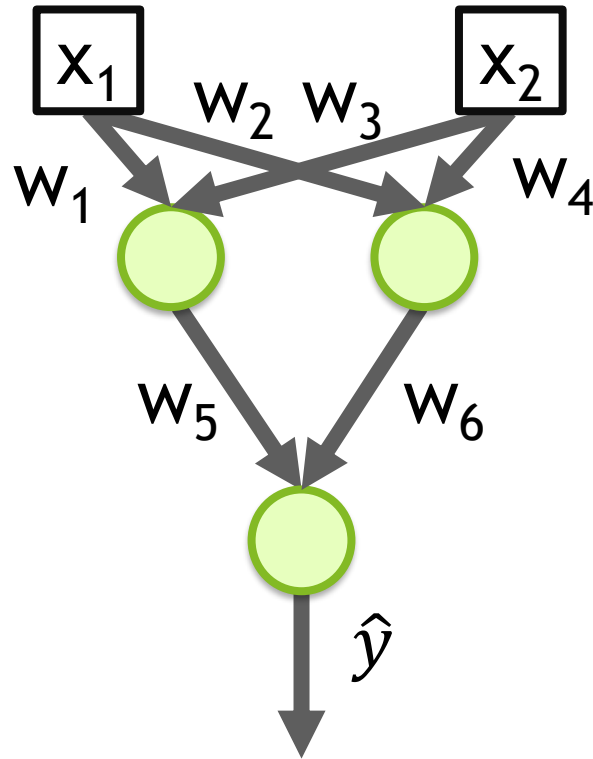
# FROM NEURON TO NETWORK

# BUILDING A NETWORK



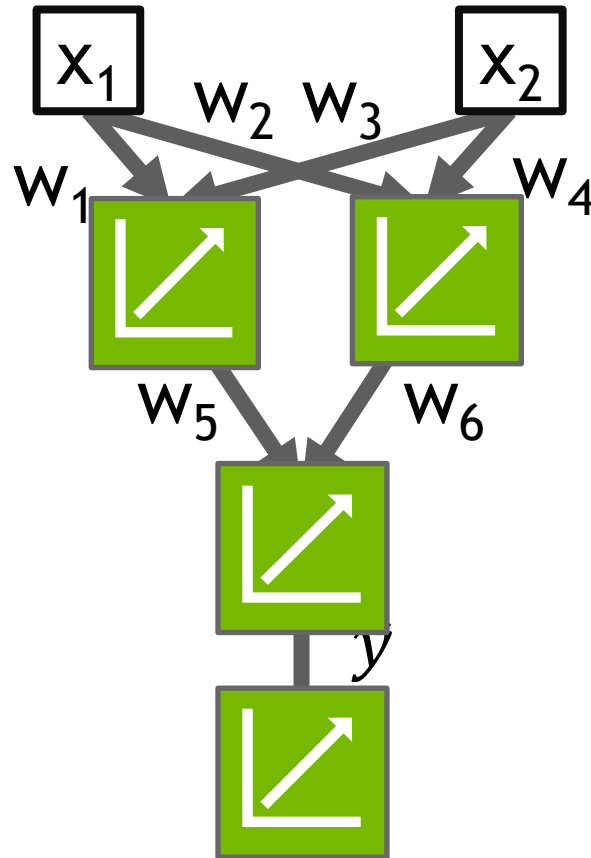
- Scales to more inputs

# BUILDING A NETWORK



- Scales to more inputs
- Can chain neurons

# BUILDING A NETWORK



- Scales to more inputs
- Can chain neurons
- If all regressions are linear, then output will also be a linear regression





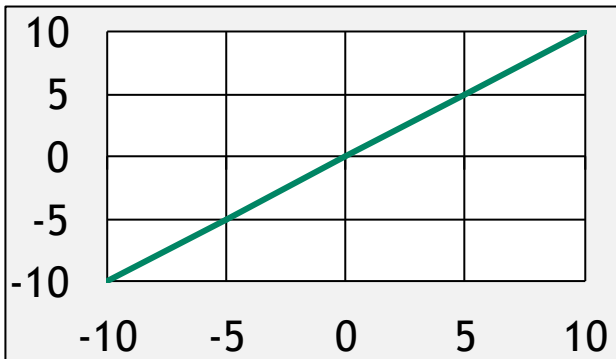
# ACTIVATION FUNCTIONS

# ACTIVATION FUNCTIONS

## Linear

$$\hat{y} = wx + b$$

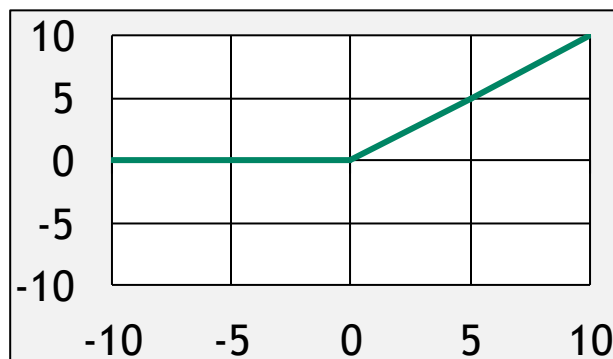
```
1 # Multiply each input
2 # with a weight (w) and
3 # add intercept (b)
4 y_hat = wx+b
```



## ReLU

$$\hat{y} = \begin{cases} wx + b & \text{if } wx + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

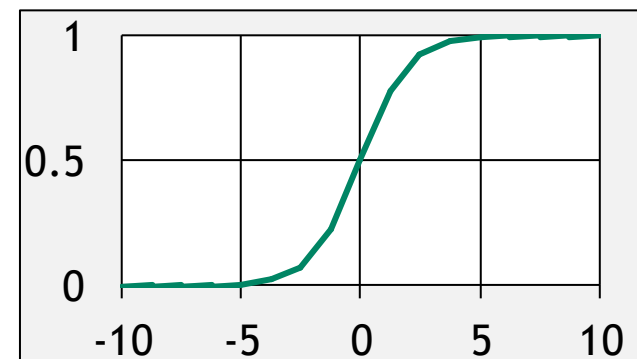
```
1 # Only return result
2 # if total is positive
3 linear = wx+b
4 y_hat = linear * (linear > 0)
```



## Sigmoid

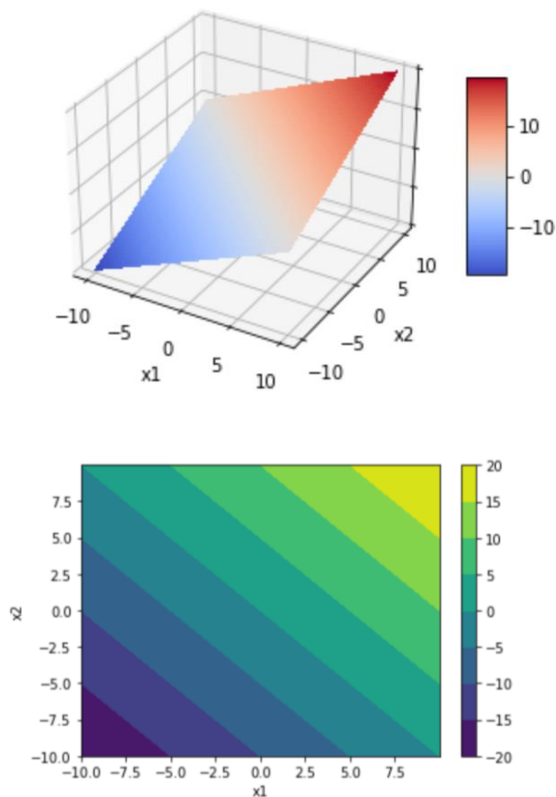
$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
1 # Start with line
2 linear = wx + b
3 # Warp to - inf to 0
4 inf_to_zero = np.exp(-1 * linear)
5 # Squish to -1 to 1
6 y_hat = 1 / (1 + inf_to_zero)
```

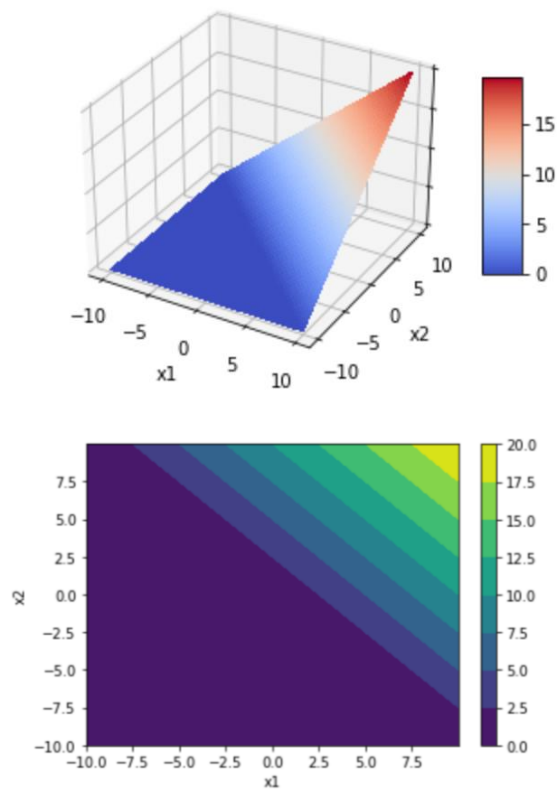


# ACTIVATION FUNCTIONS

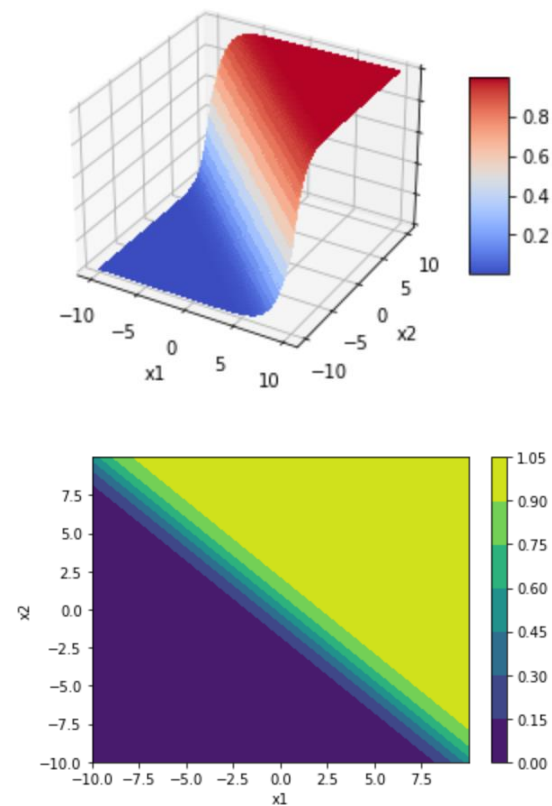
Linear



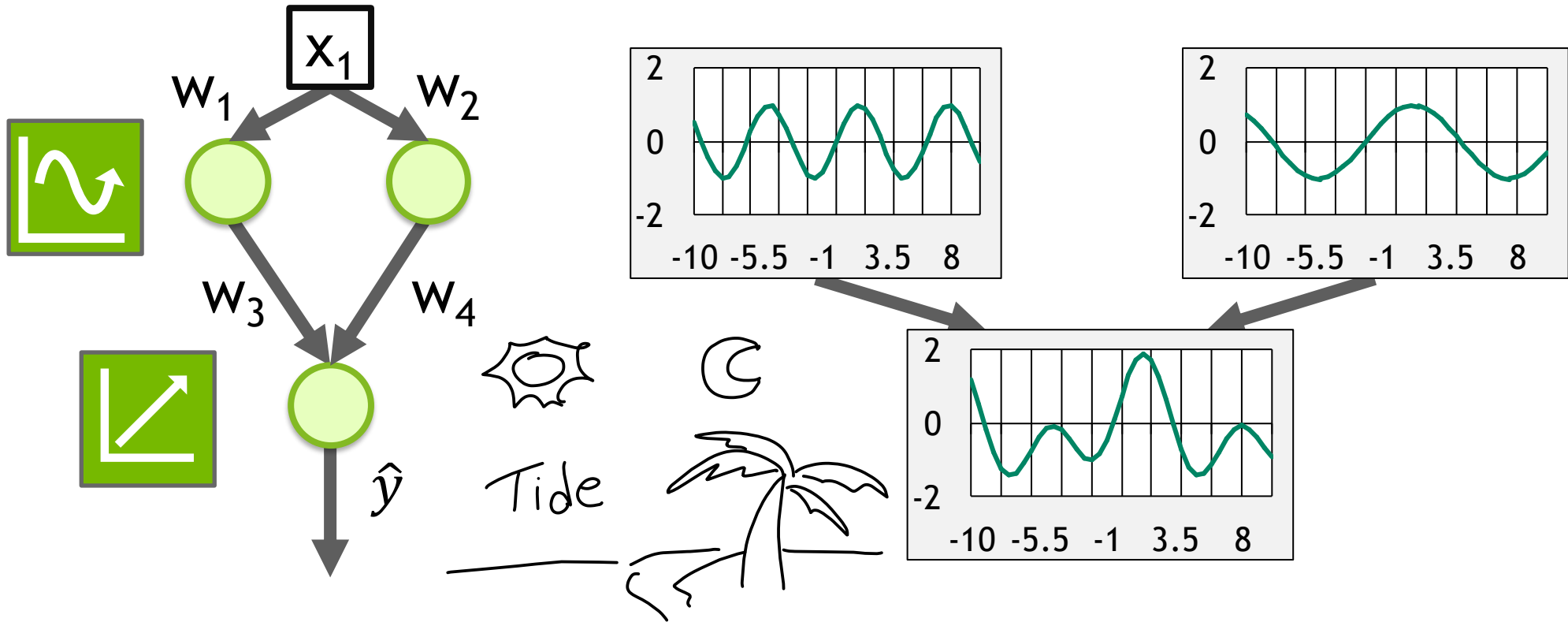
ReLU



Sigmoid



# ACTIVATION FUNCTIONS





OVERFITTING

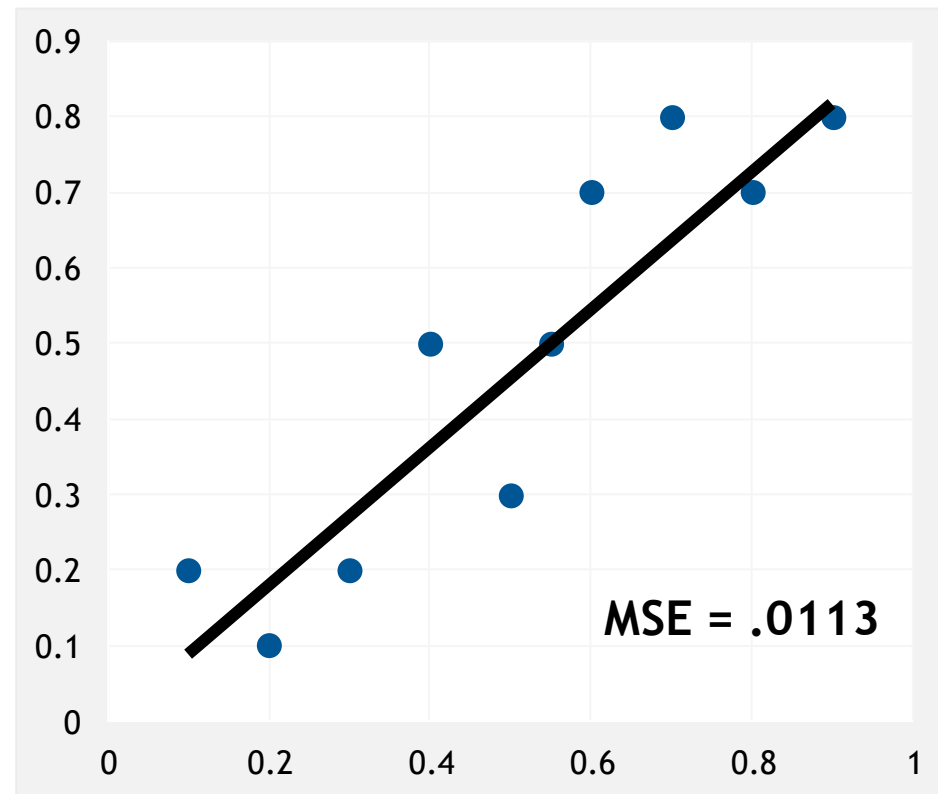
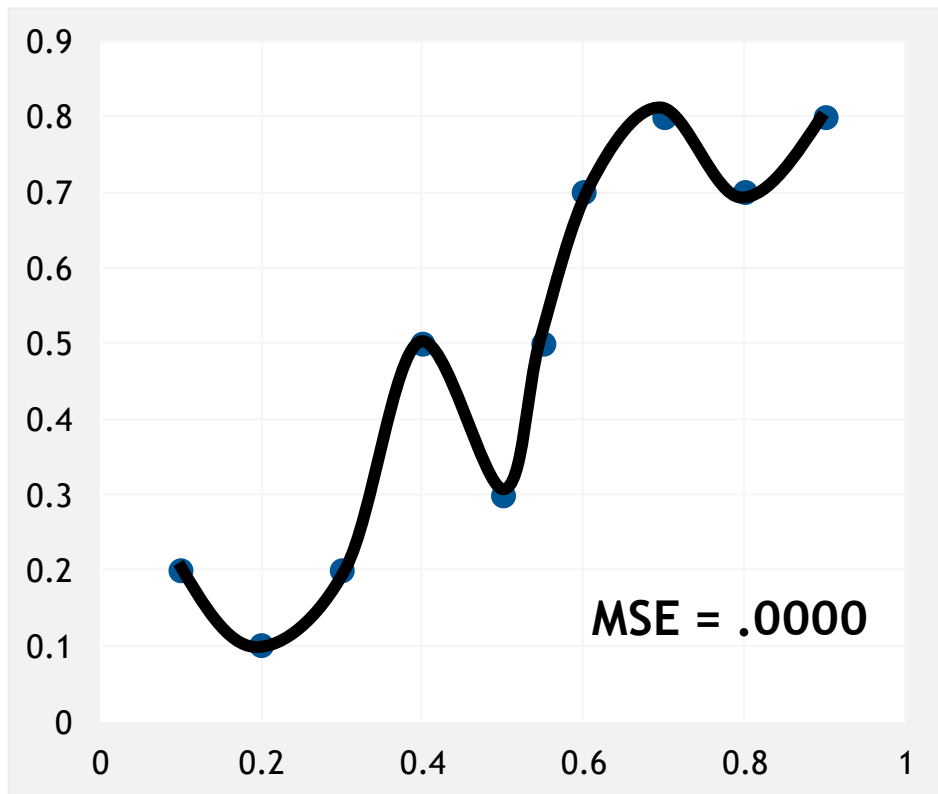
# OVERFITTING

Why not have a super large neural network?



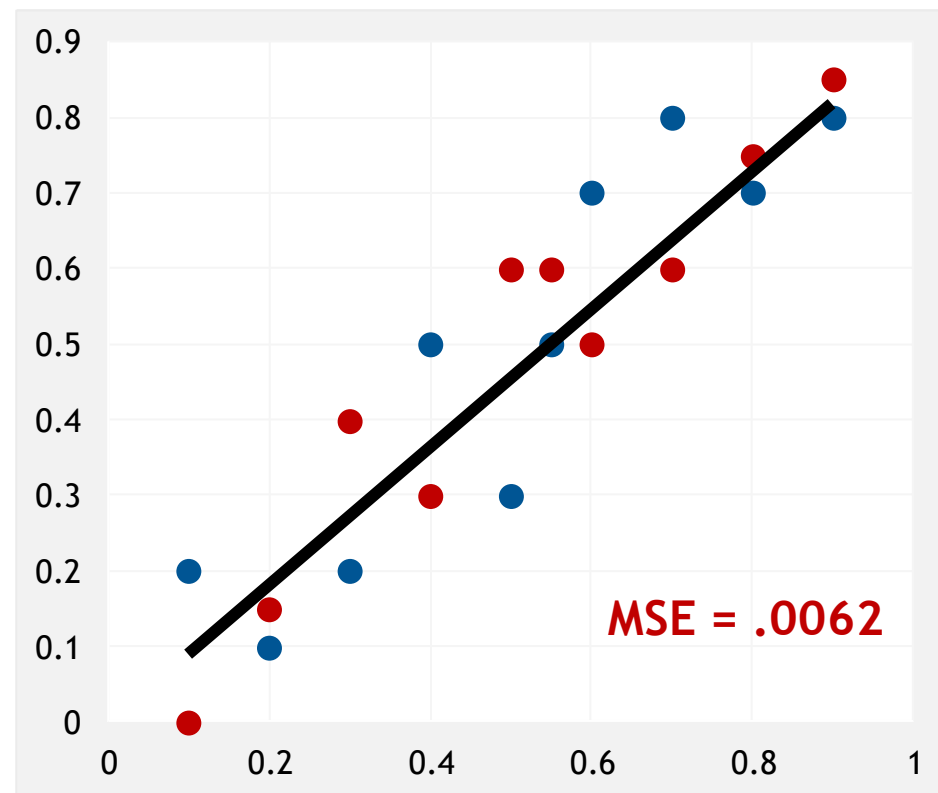
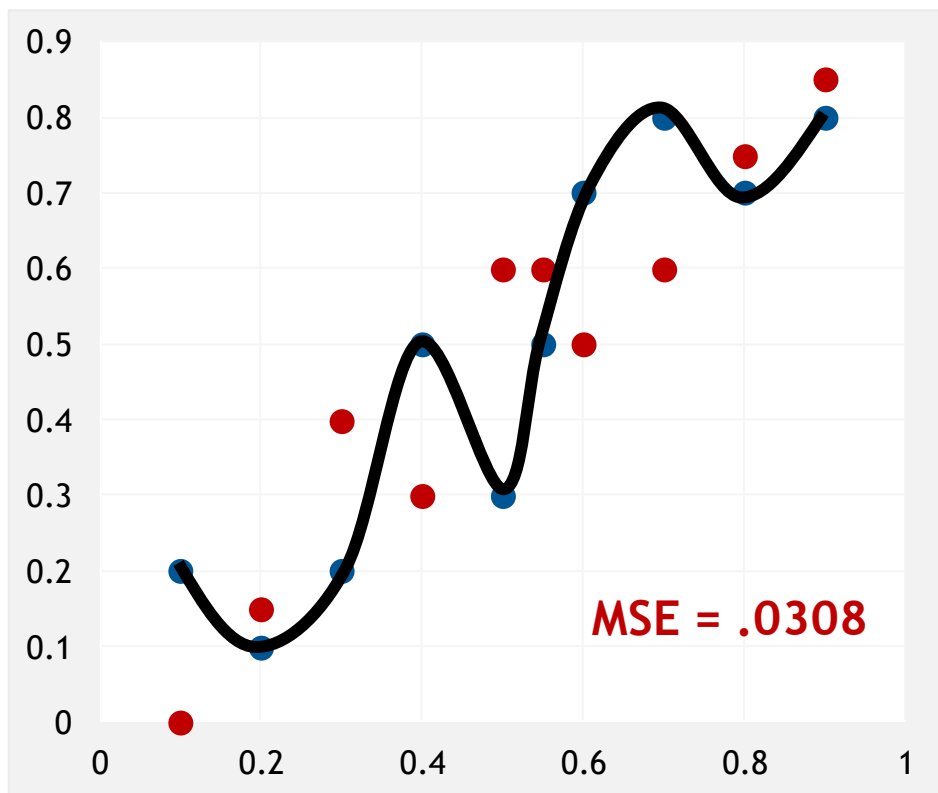
# OVERFITTING

Which Trendline is Better?



# OVERFITTING

Which Trendline is Better?





# TRAINING VS VALIDATION DATA

Avoid memorization

## Training data

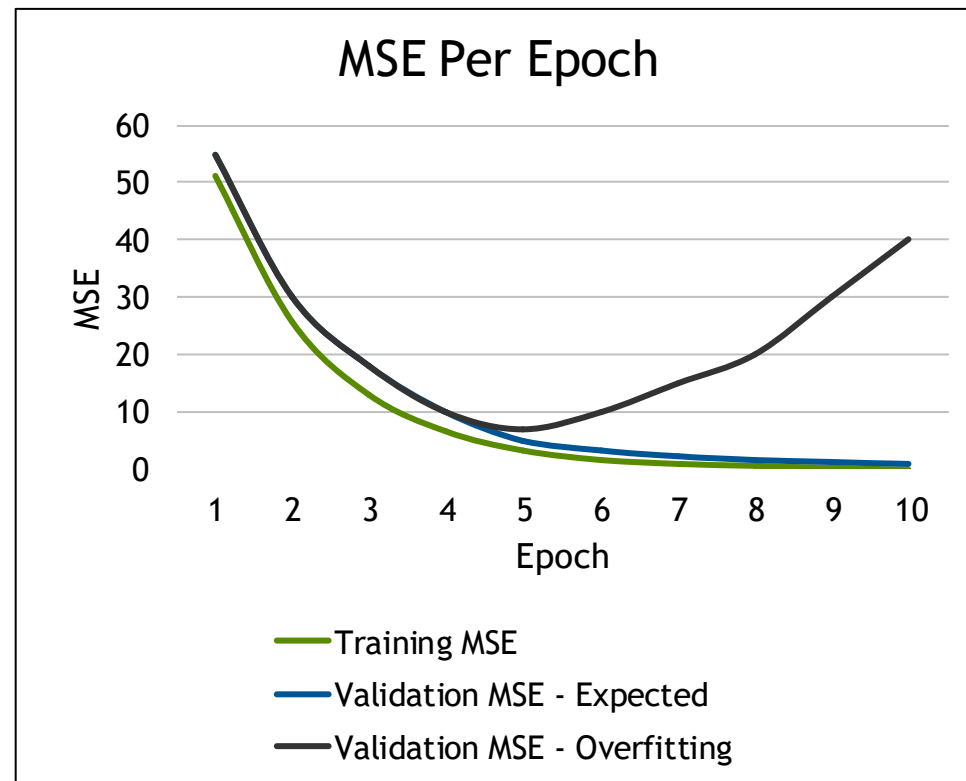
- Core dataset for the model to learn on

## Validation data

- New data for model to see if it truly understands (can generalize)

## Overfitting

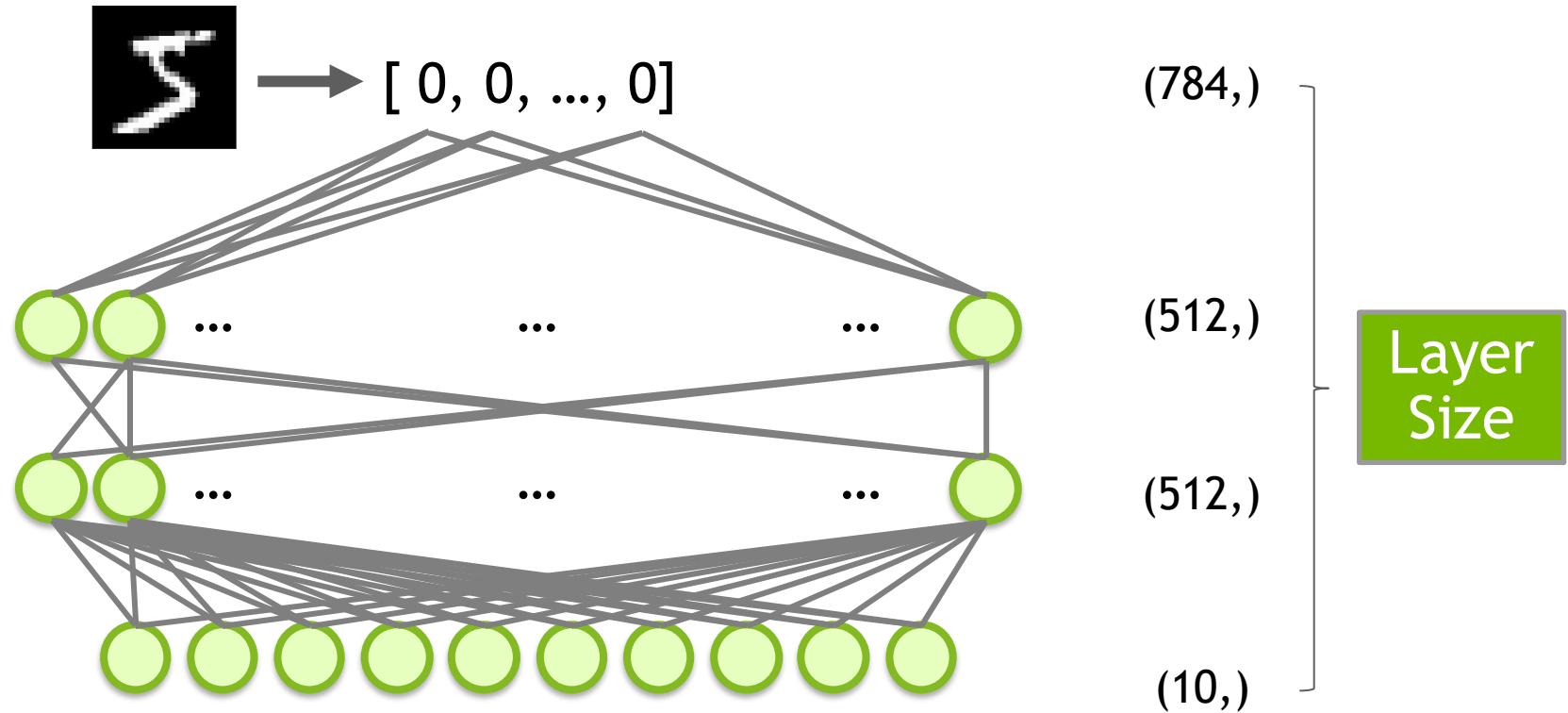
- When model performs well on the training data, but not the validation data (evidence of memorization)
- Ideally the accuracy and loss should be similar between both datasets



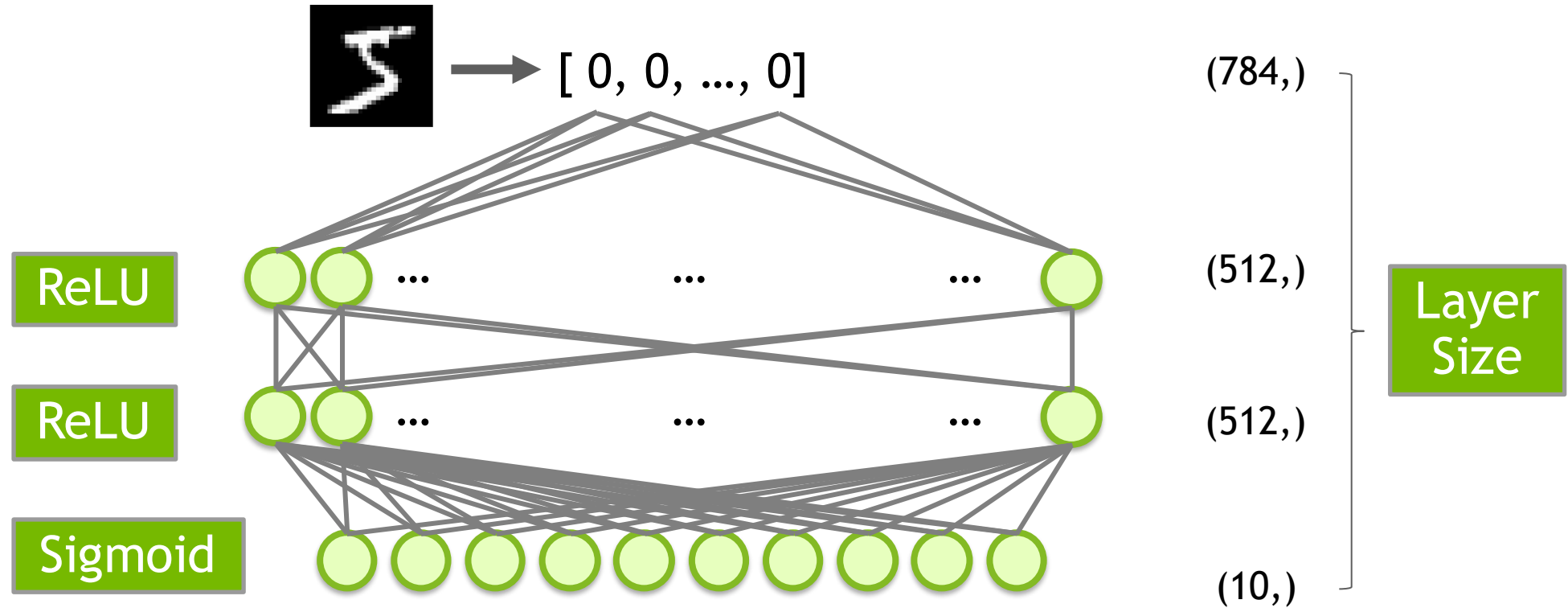


# FROM REGRESSION TO CLASSIFICATION

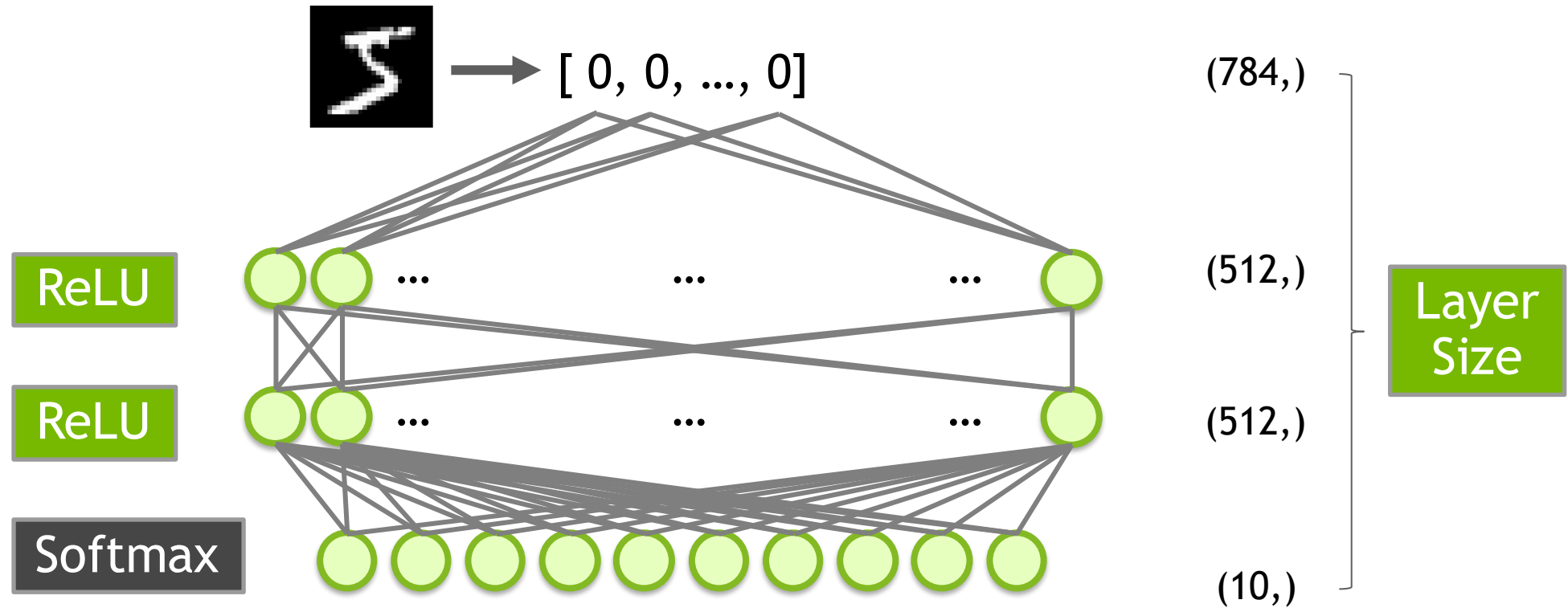
# AN MNIST MODEL



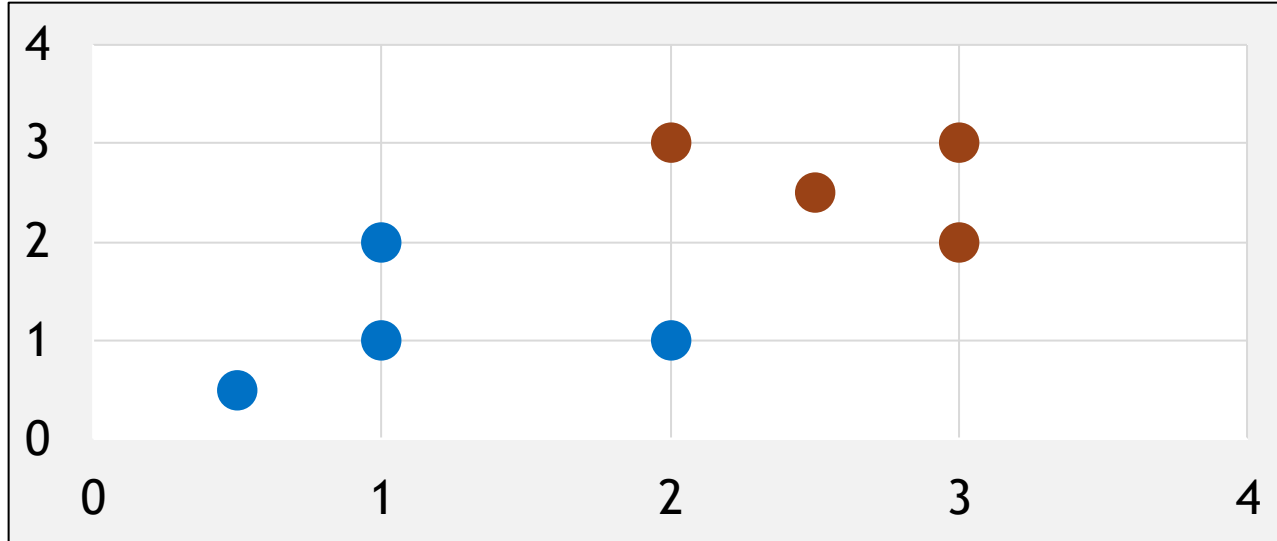
# AN MNIST MODEL



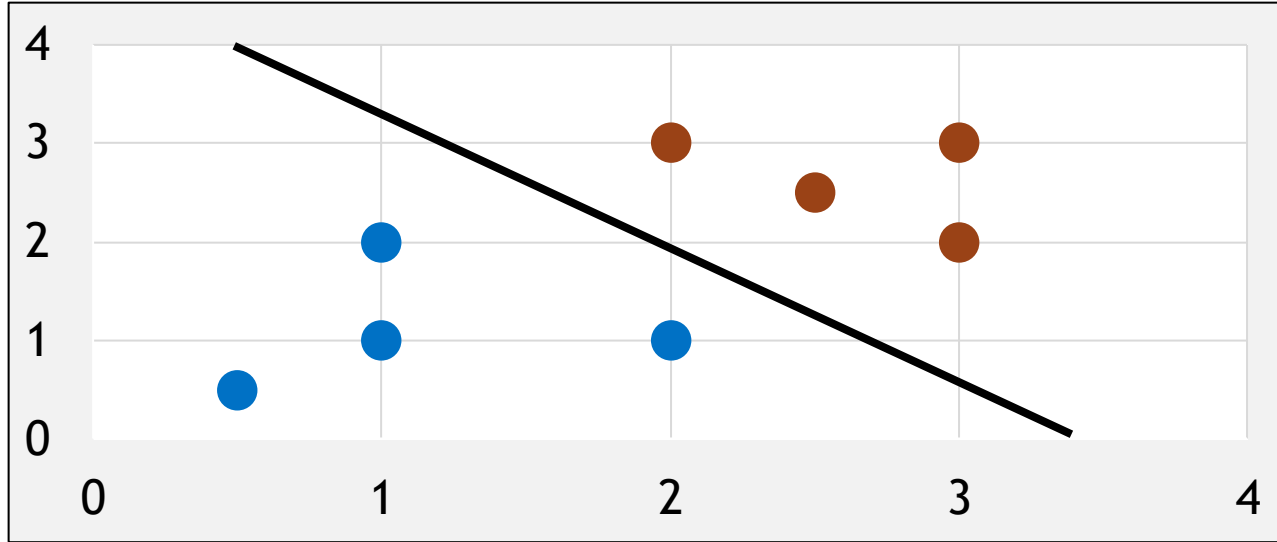
# AN MNIST MODEL



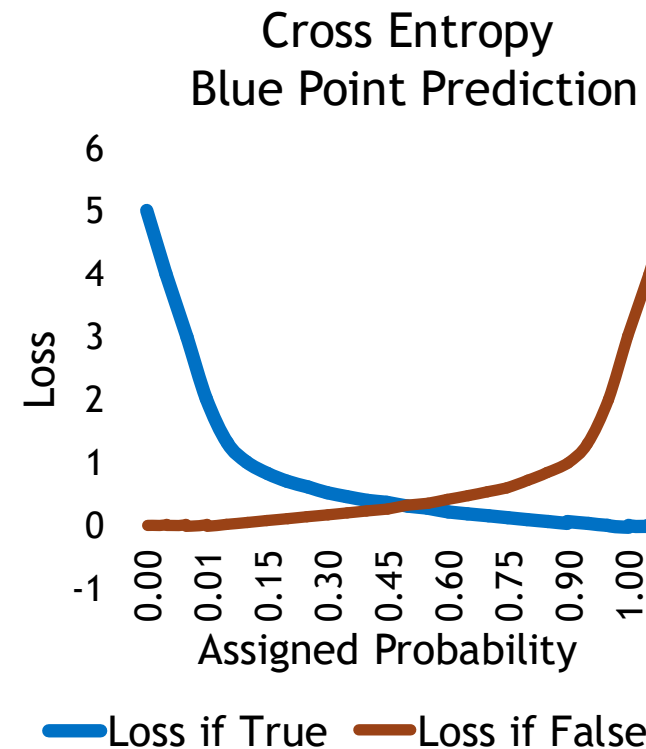
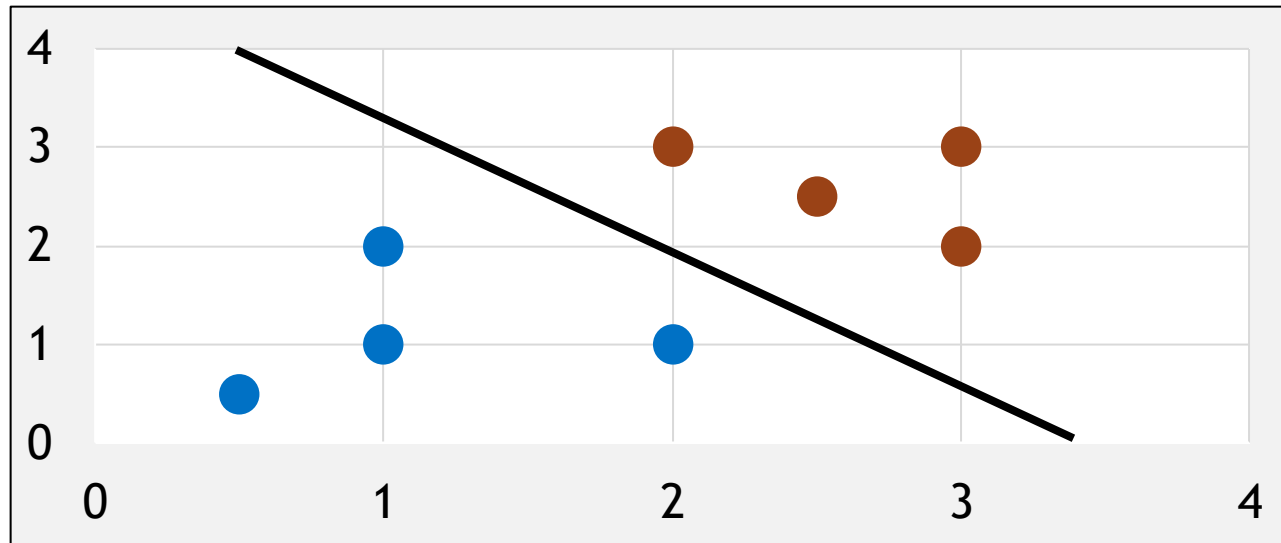
# RMSE FOR PROBABILITIES?



# RMSE FOR PROBABILITIES?

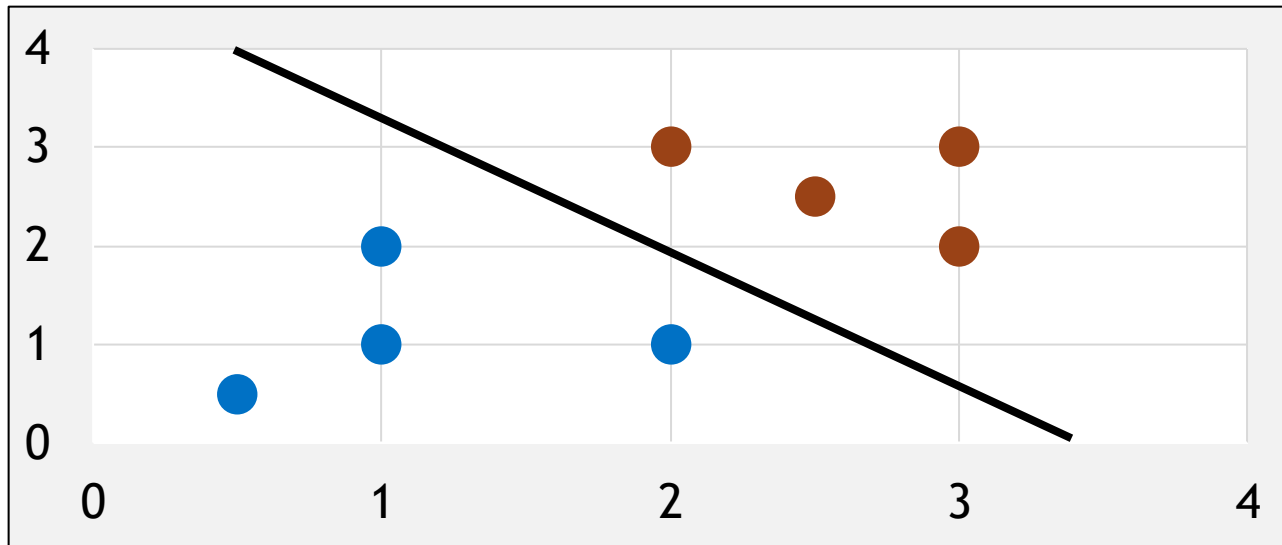


# CROSS ENTROPY





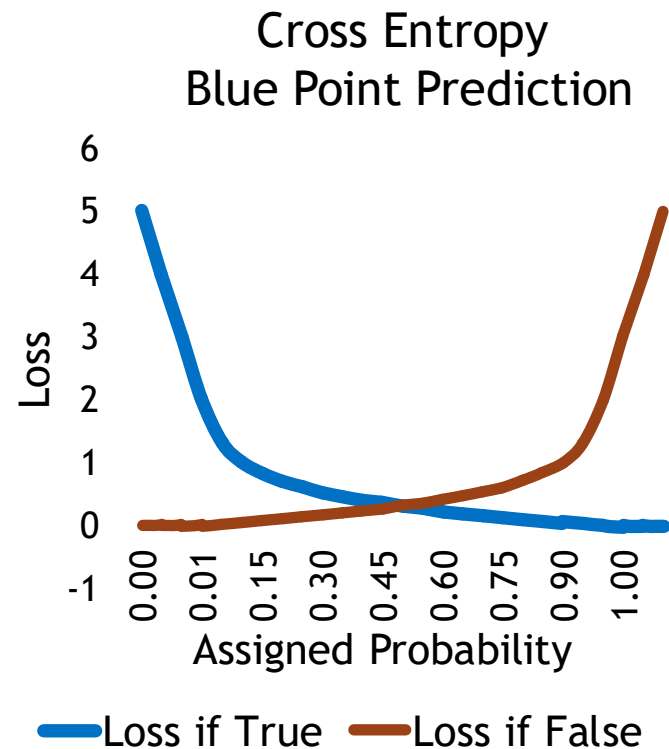
# CROSS ENTROPY



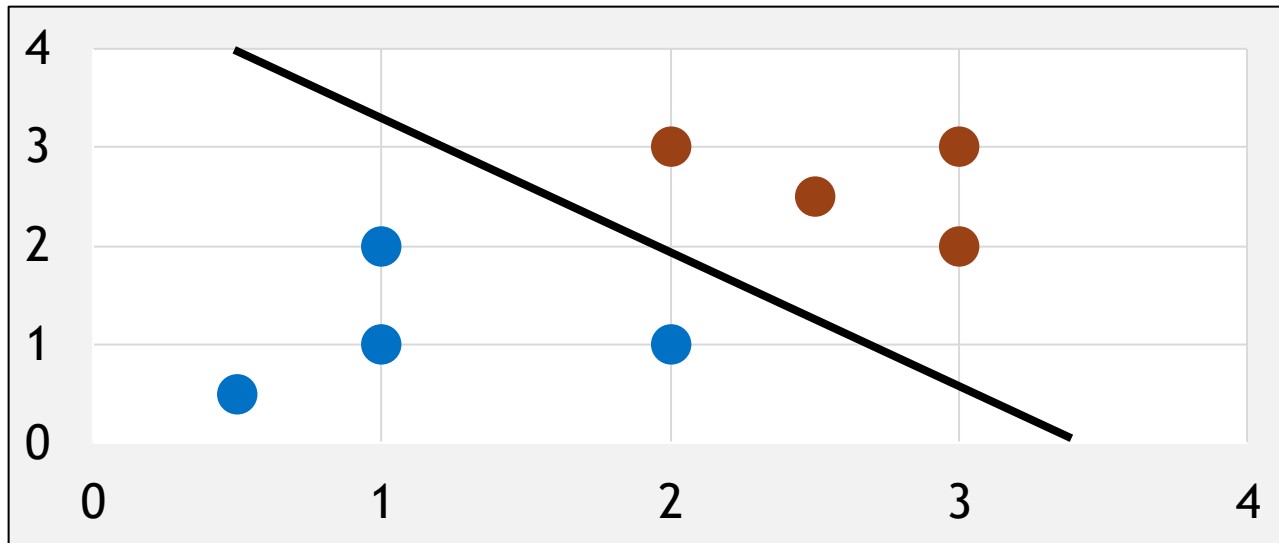
$$Loss = -((t(x) \cdot \log(p(x))) + (1 - t(x)) \cdot \log(1 - p(x)))$$

$t(x)$  = target (0 if False, 1 if True)

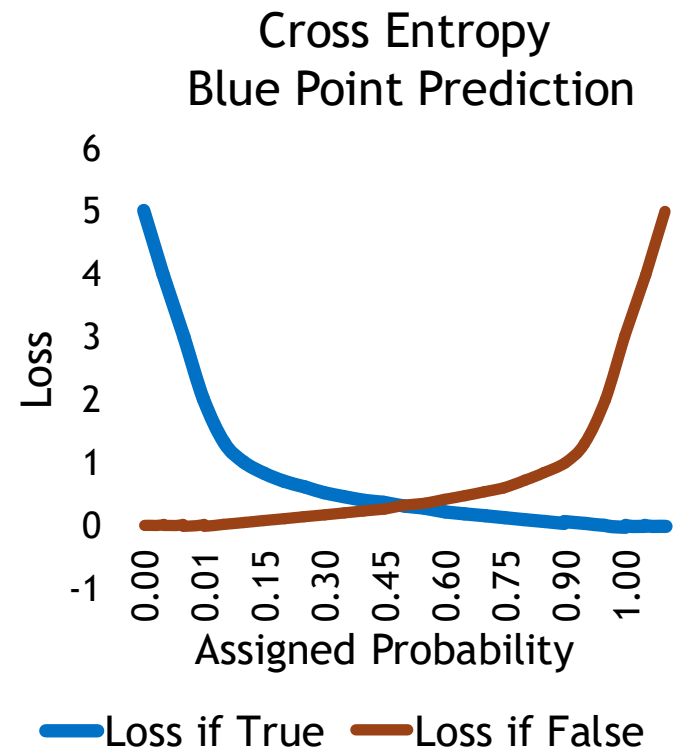
$p(x)$  = probability prediction of point  $x$



# CROSS ENTROPY



```
1 def cross_entropy(y_hat, y_actual):  
2     """Infinite error for misplaced confidence."""  
3     loss = log(y_hat) if y_actual else log(1-y_hat)  
4     return -1*loss
```

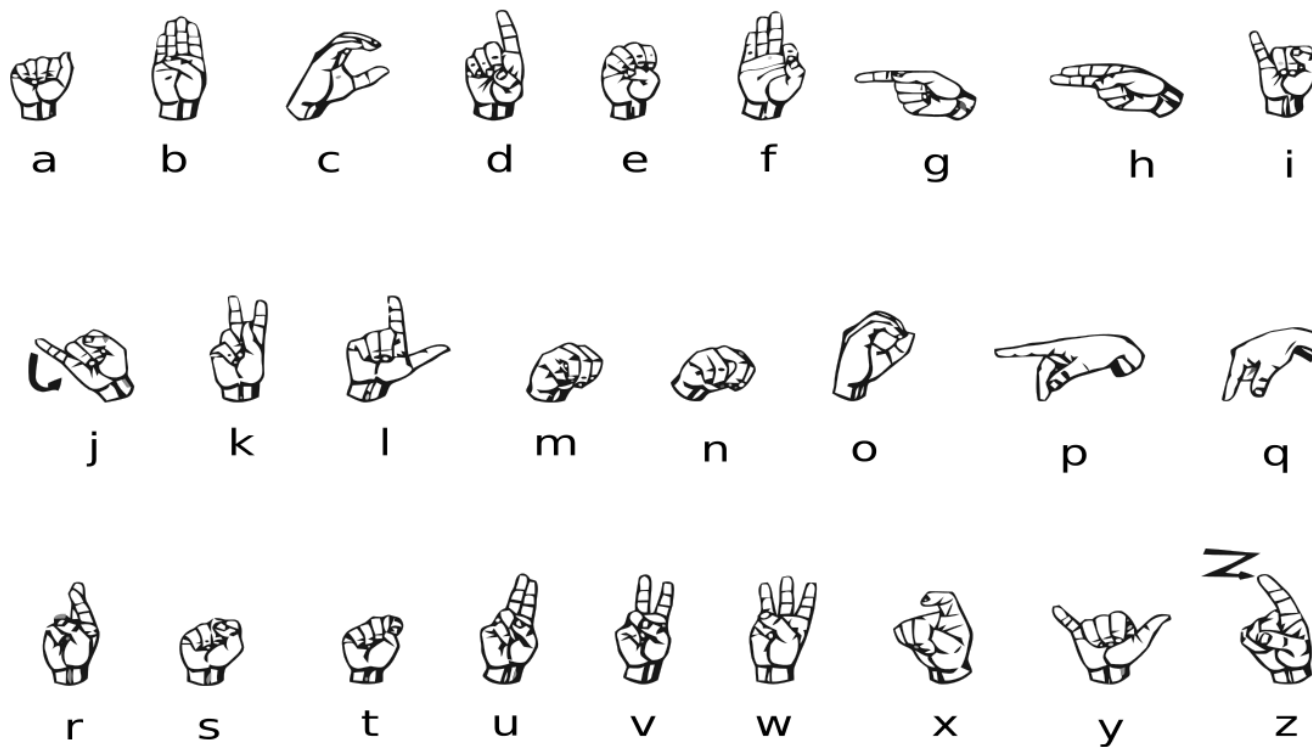




BRINGING IT TOGETHER

# THE NEXT EXERCISE

## The American Sign Language Alphabet





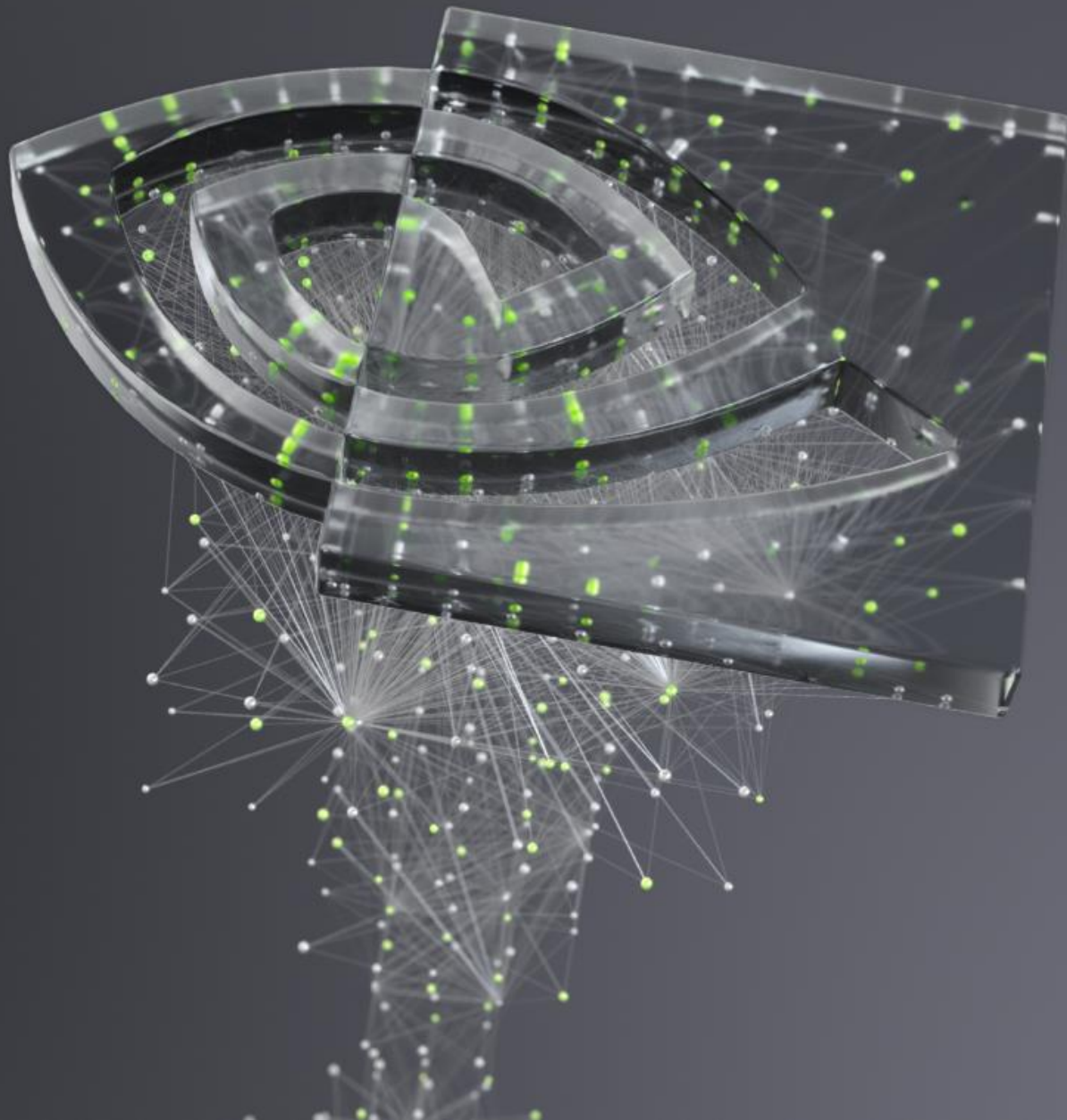
LET'S GO!



DEEP  
LEARNING  
INSTITUTE

# APPENDIX: GRADIENT DESCENT

HELPING THE COMPUTER CHEAT CALCULUS



# Learning From Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2 = \frac{1}{n} \sum_{i=1}^n (y - (mx + b))^2$$

$$MSE = \frac{1}{2} ((3 - (m(1) + b))^2 + (5 - (m(2) + b))^2)$$

$$\frac{\partial MSE}{\partial m} = 5m + 3b - 13$$

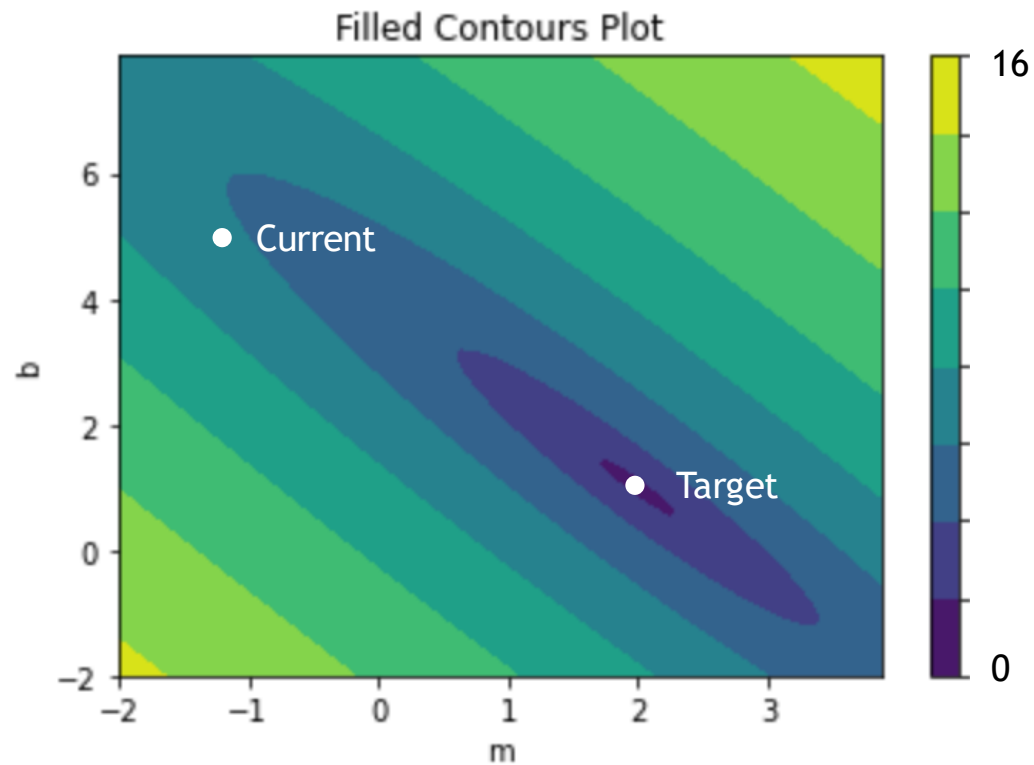
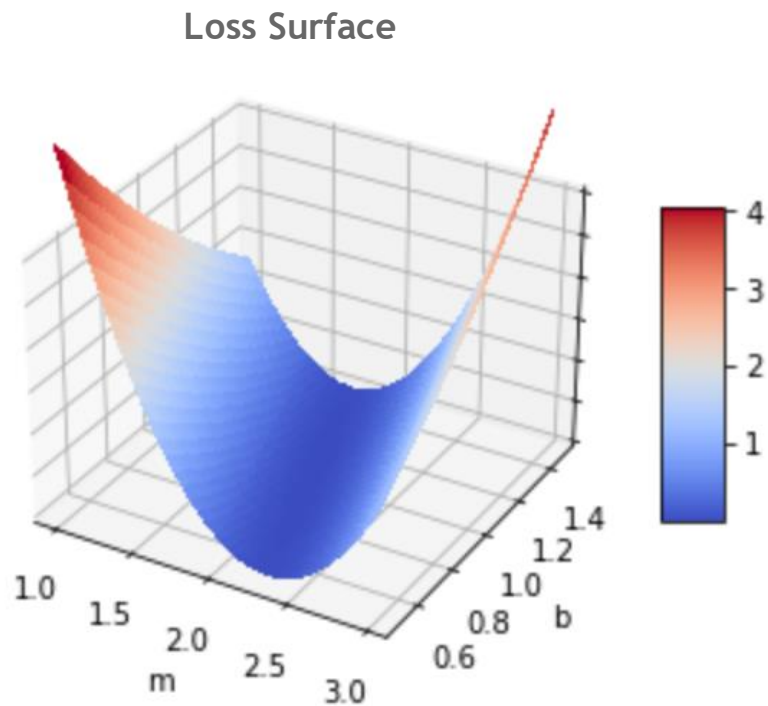
$$\frac{\partial MSE}{\partial b} = 3m + 2b - 8$$

$$\frac{\partial MSE}{\partial m} = -3$$

$$\frac{\partial MSE}{\partial b} = -1$$

$$m = -1$$
$$b = 5$$

# THE LOSS CURVE





# THE LOSS CURVE

$$\frac{\partial MSE}{\partial m} = -7$$

$$\frac{\partial MSE}{\partial b} = -3$$

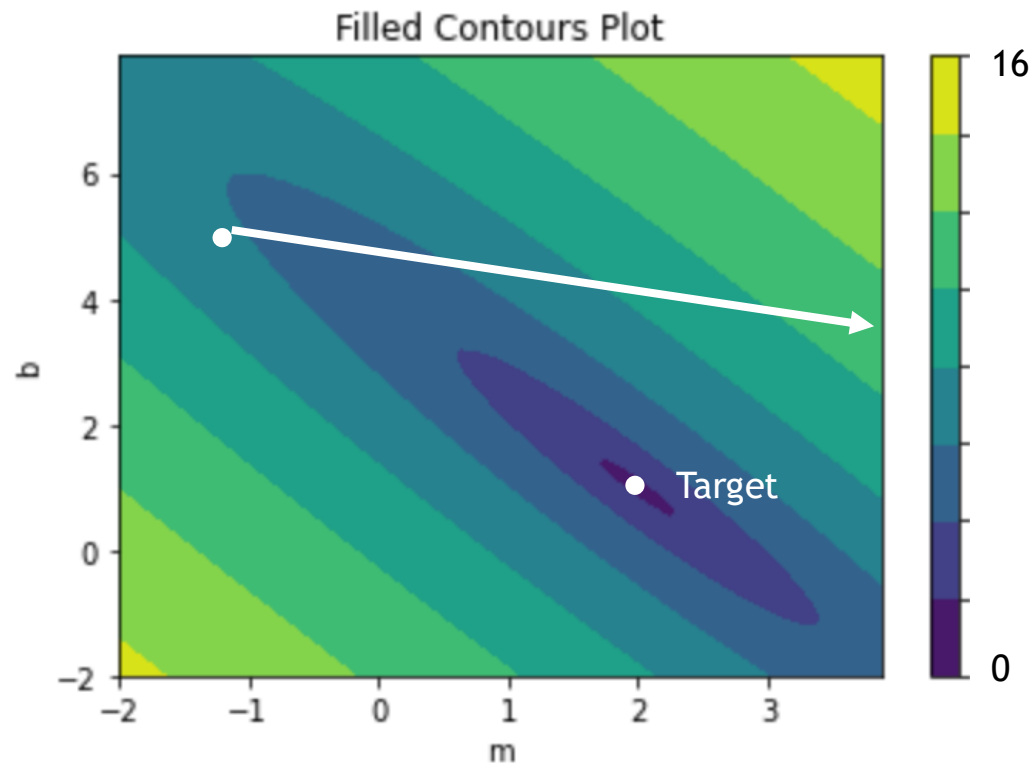


# THE LOSS CURVE

$$\frac{\partial MSE}{\partial m} = -7 \quad \frac{\partial MSE}{\partial b} = -3$$

$$m := m - \lambda \frac{\partial MSE}{\partial m}$$

$$b := b - \lambda \frac{\partial MSE}{\partial b}$$



# THE LOSS CURVE

$$\frac{\partial MSE}{\partial m} = -7 \quad \frac{\partial MSE}{\partial b} = -3$$

$$m := m - \lambda \frac{\partial MSE}{\partial m} \quad \lambda = .6$$

$$b := b - \lambda \frac{\partial MSE}{\partial b}$$



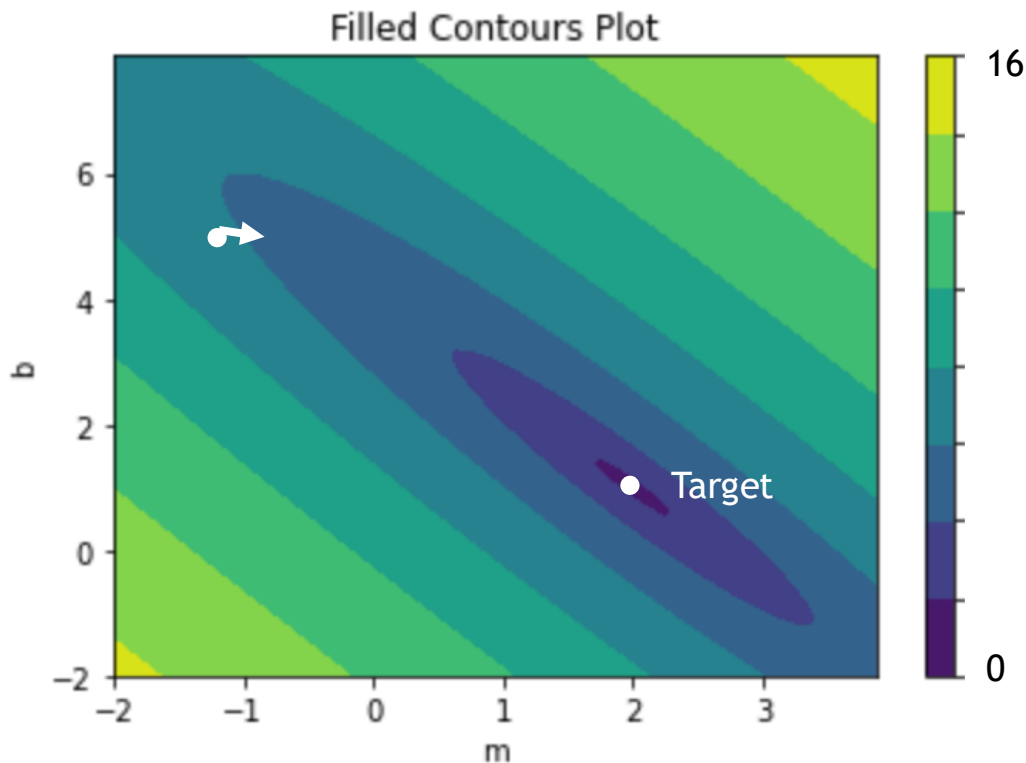
# THE LOSS CURVE

$$\frac{\partial MSE}{\partial m} = -7 \quad \frac{\partial MSE}{\partial b} = -3$$

$$m := m - \lambda \frac{\partial MSE}{\partial m}$$

$$\lambda = .005$$

$$b := b - \lambda \frac{\partial MSE}{\partial b}$$



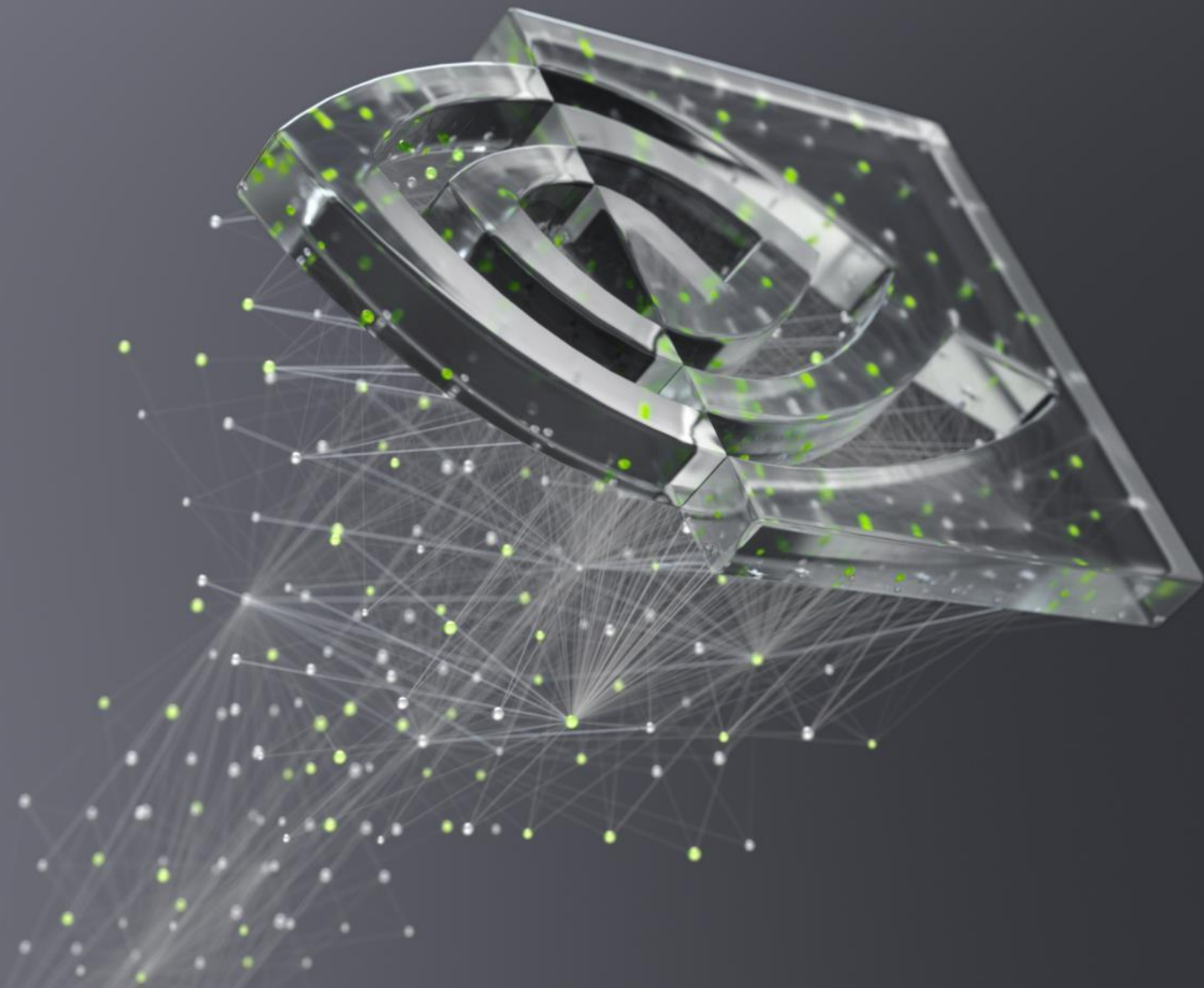
# THE LOSS CURVE

$$\lambda = .1$$

$$m := -1 + 7\lambda = -0.3$$

$$b := 5 + 3\lambda = 4.7$$





DEEP  
LEARNING  
INSTITUTE