



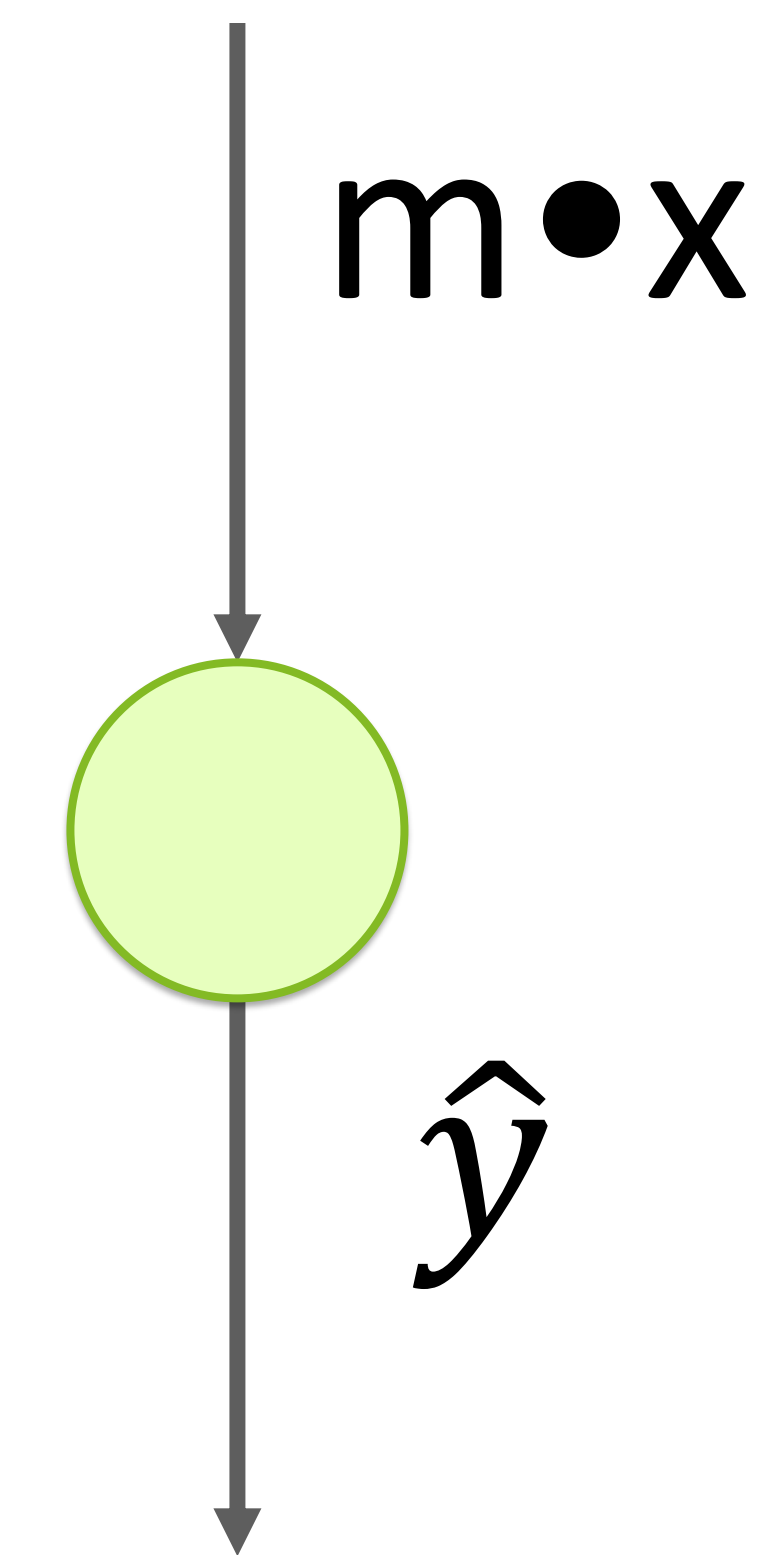
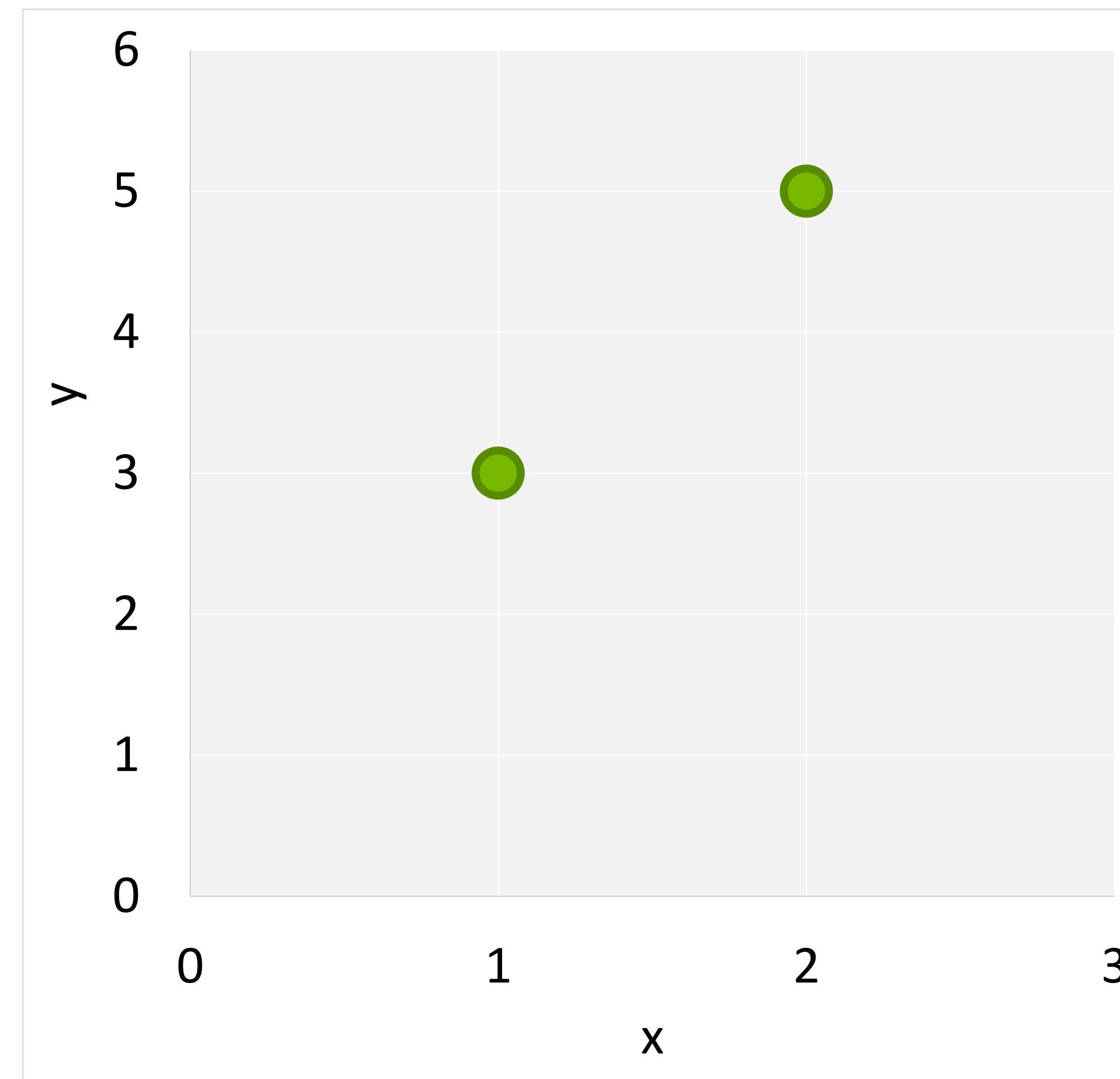
Agenda

- 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

A Simpler Model

$$y = mx + b$$

| x | y |
|---|---|
| 1 | 3 |
| 2 | 5 |



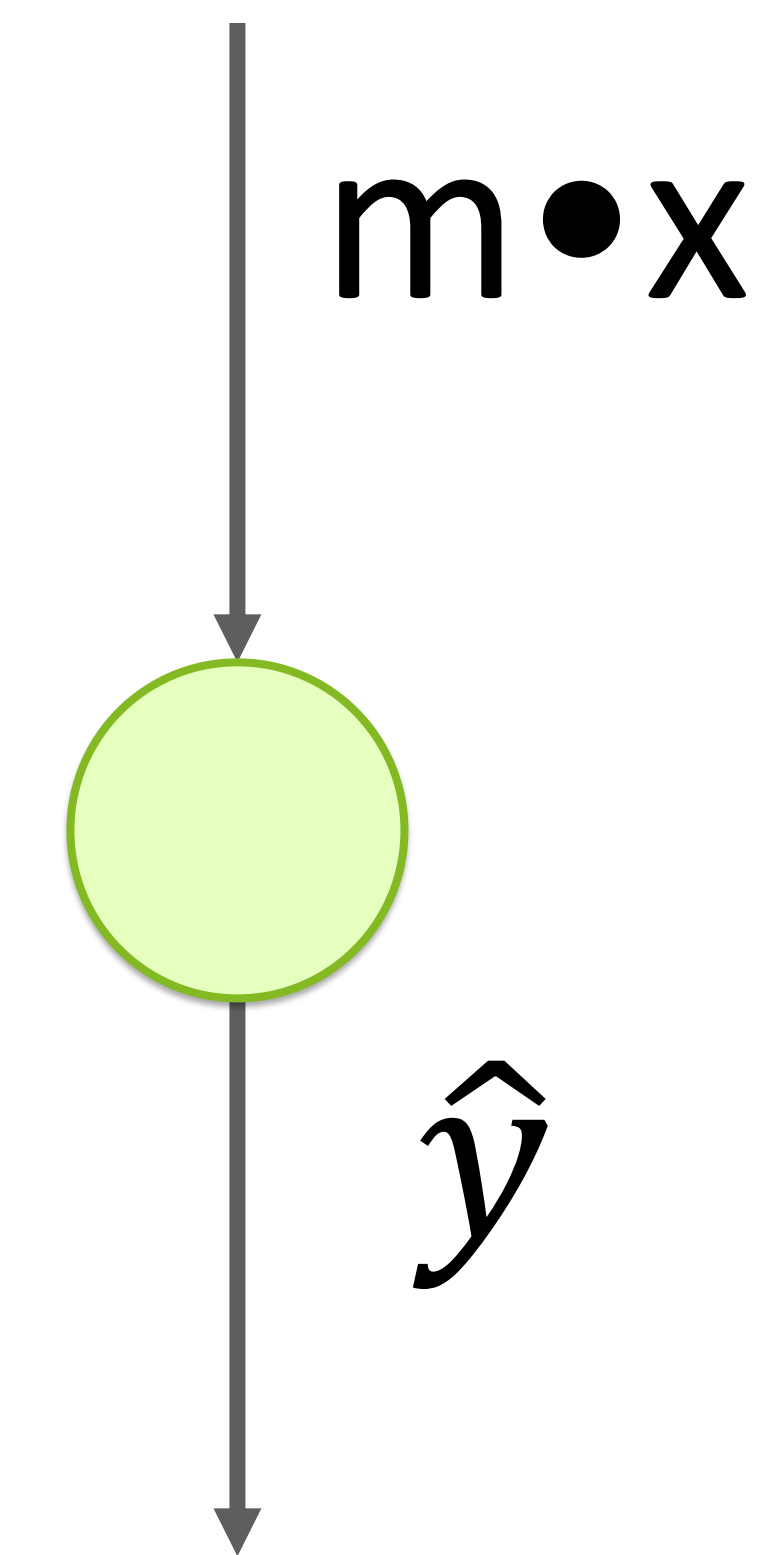
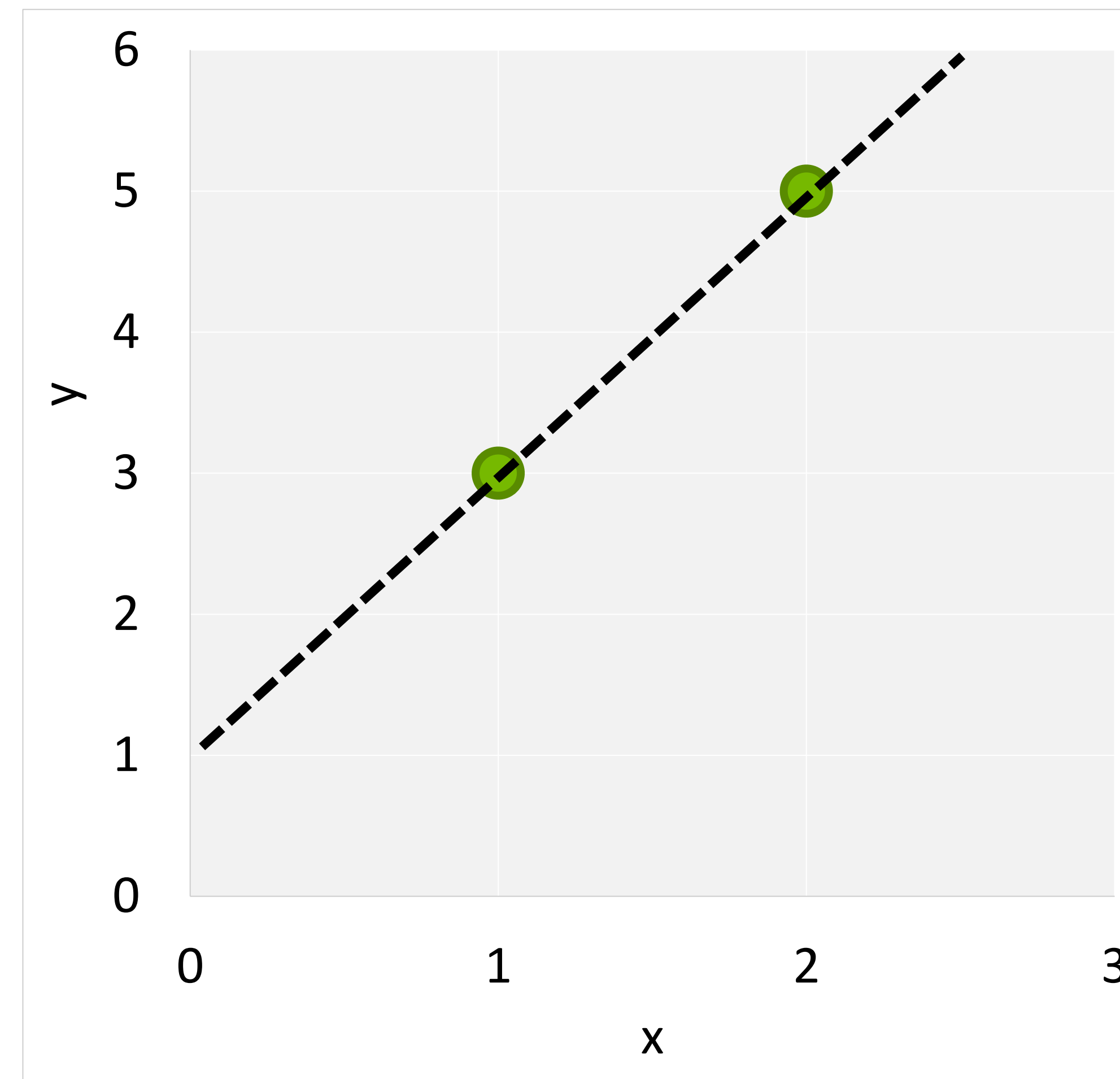
$$m = ?$$

$$b = ?$$

A Simpler Model

$$y = mx + b$$

| x | y |
|---|---|
| 1 | 3 |
| 2 | 5 |



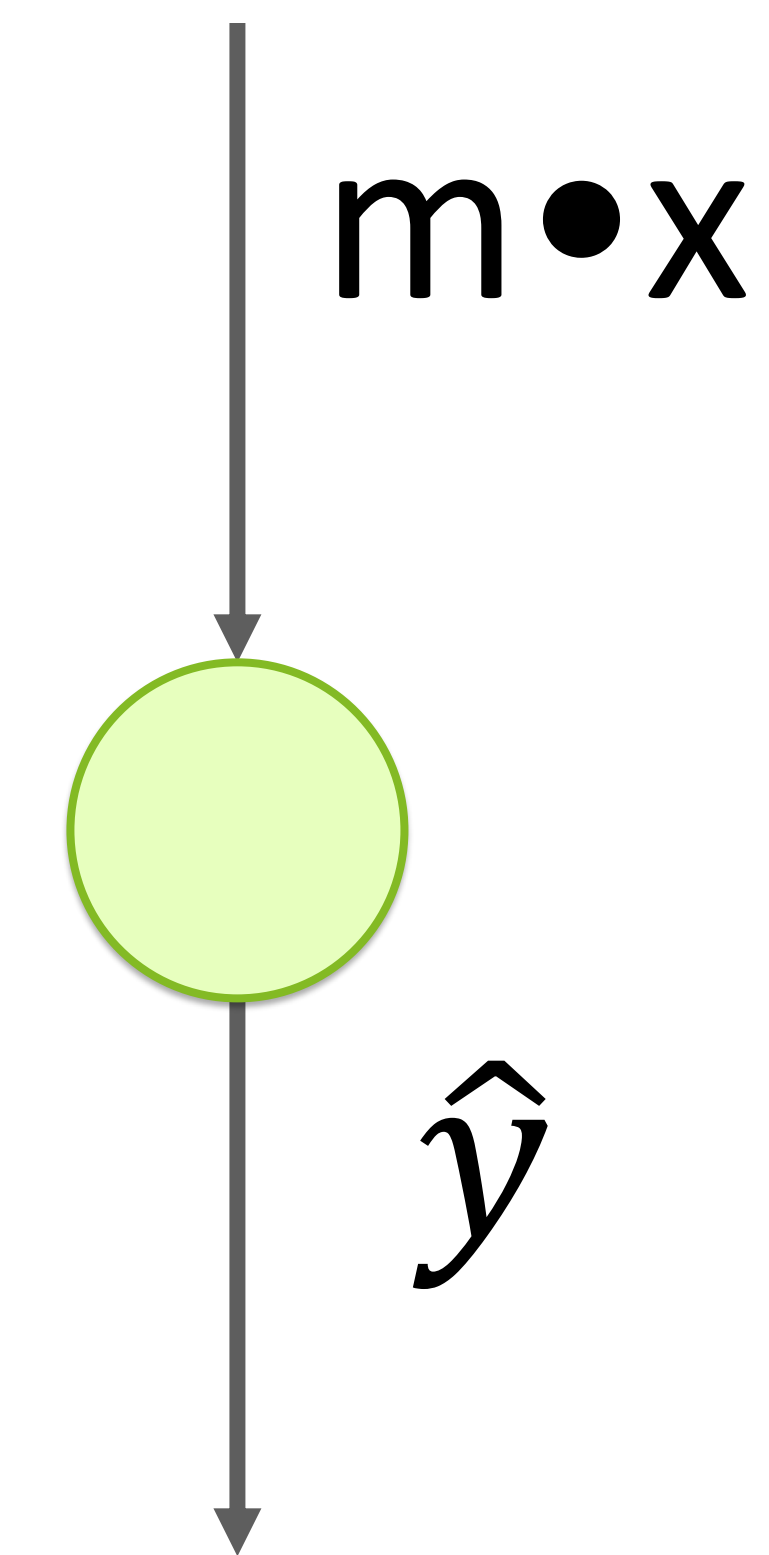
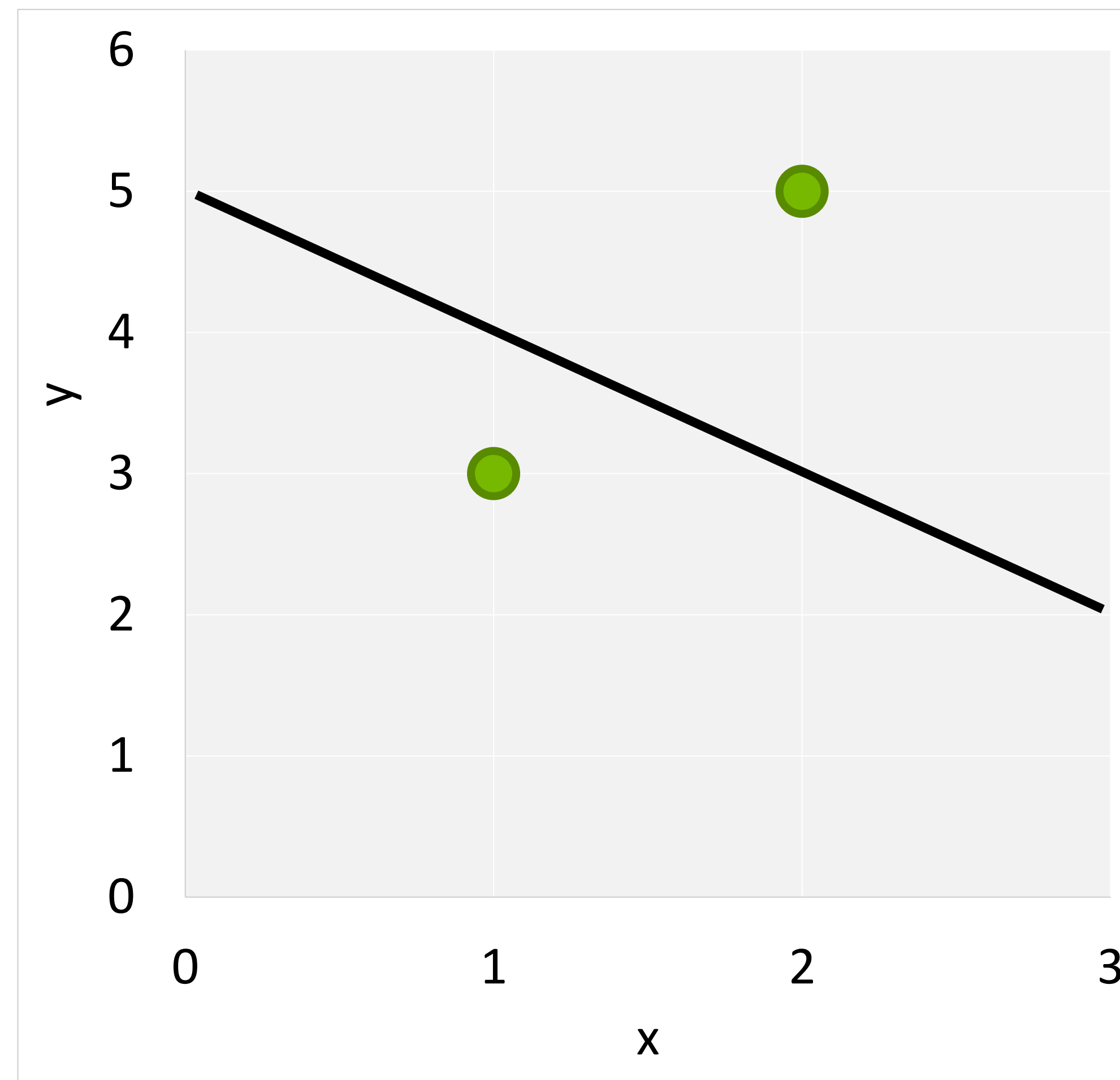
$$m = ?$$

$$b = ?$$

A Simpler Model

$$y = mx + b$$

| x | y | \hat{y} |
|-----|-----|-----------|
| 1 | 3 | 4 |
| 2 | 5 | 3 |



Start Random

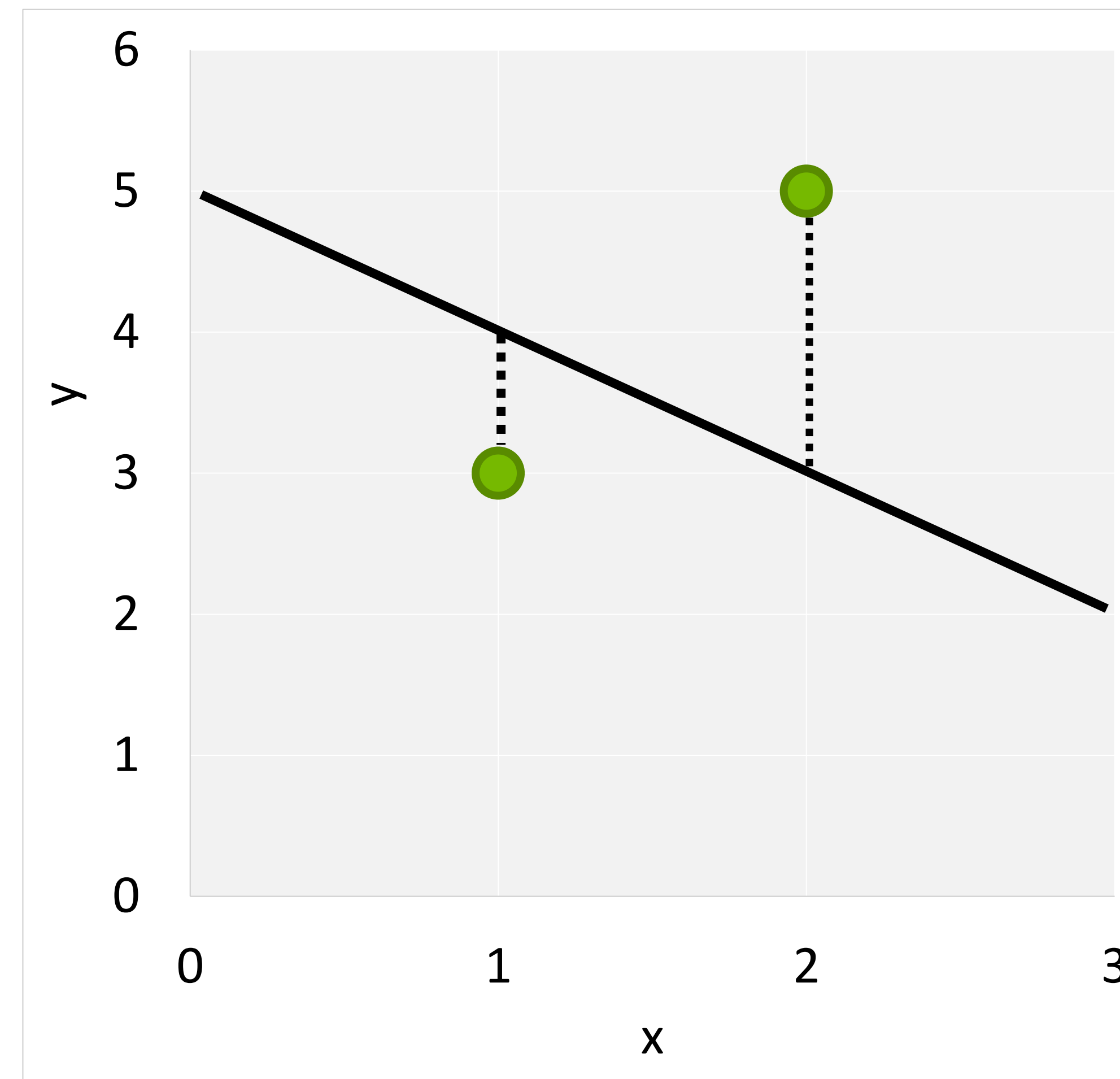
$$m = -1$$

$$b = 5$$

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 |



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

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

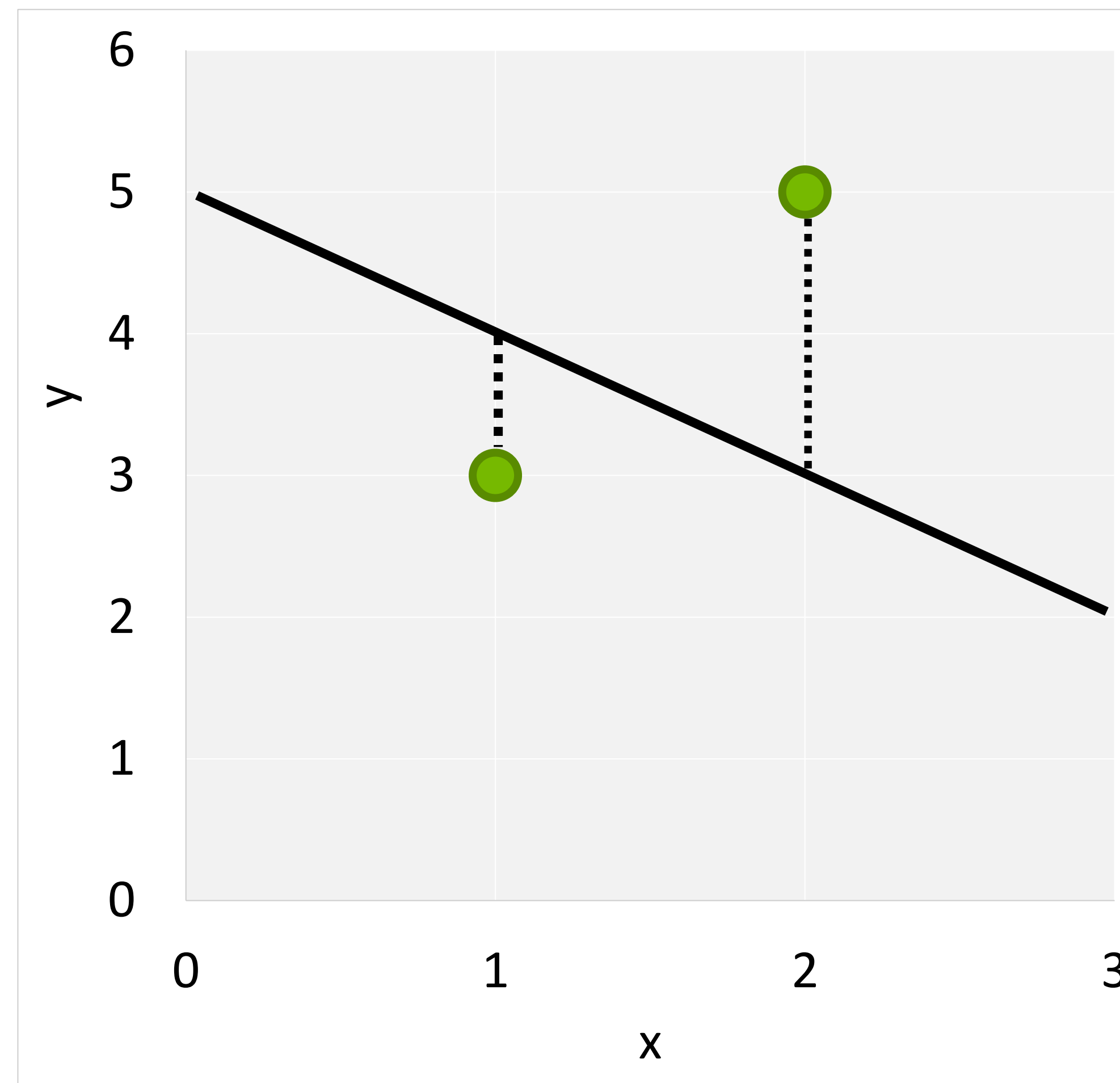
A Simpler Model

$$y = mx + b$$

| x | y | \hat{y} | err^2 |
|---|---|-----------|---------|
| 1 | 3 | 4 | 1 |
| 2 | 5 | 3 | 4 |

MSE = 2.5

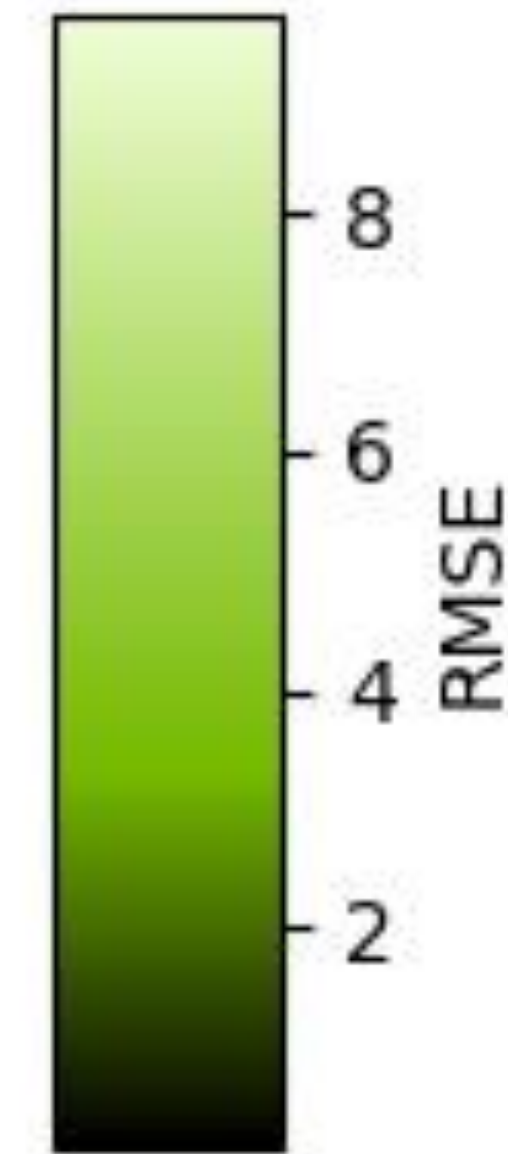
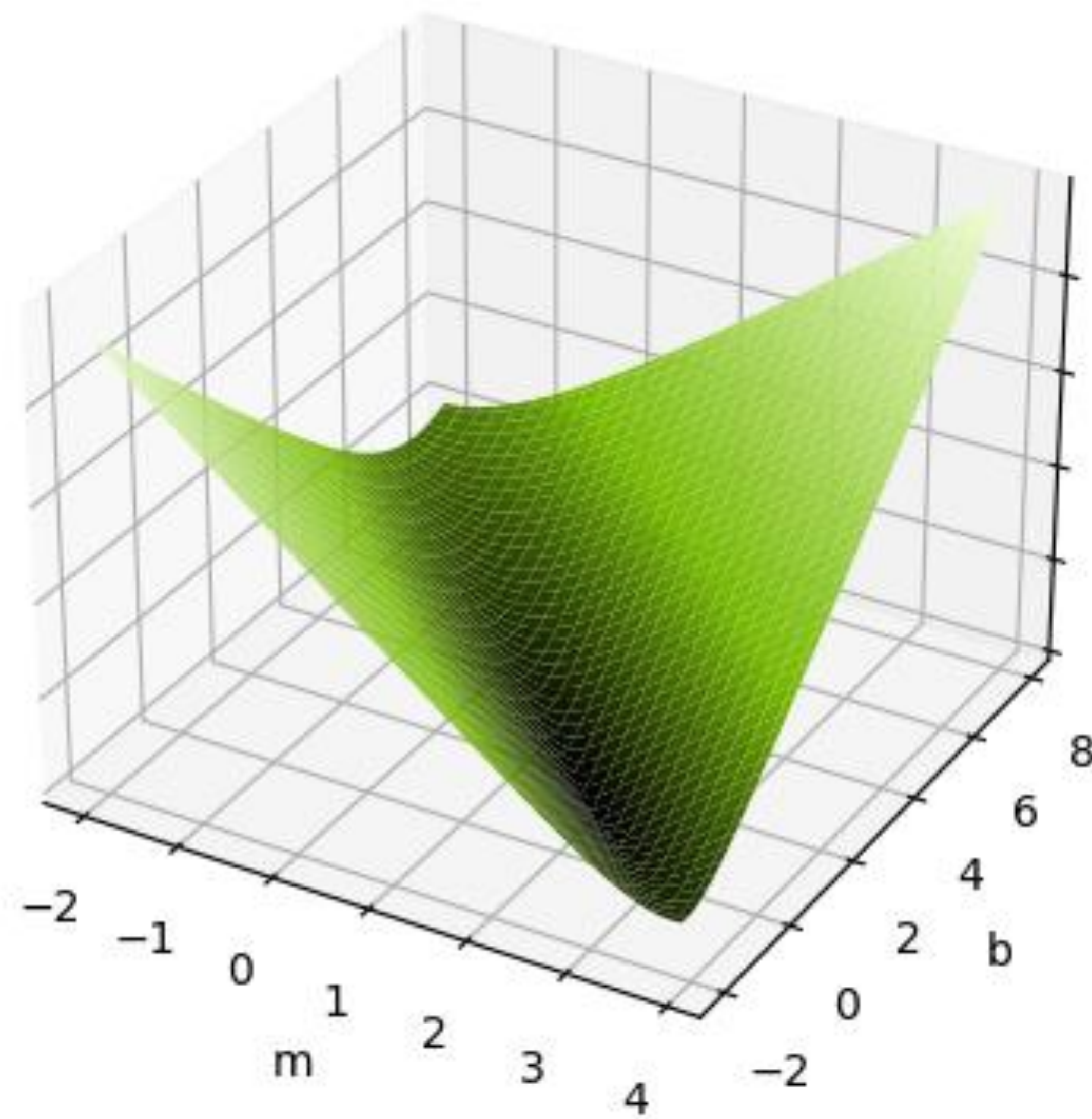
RMSE = 1.6



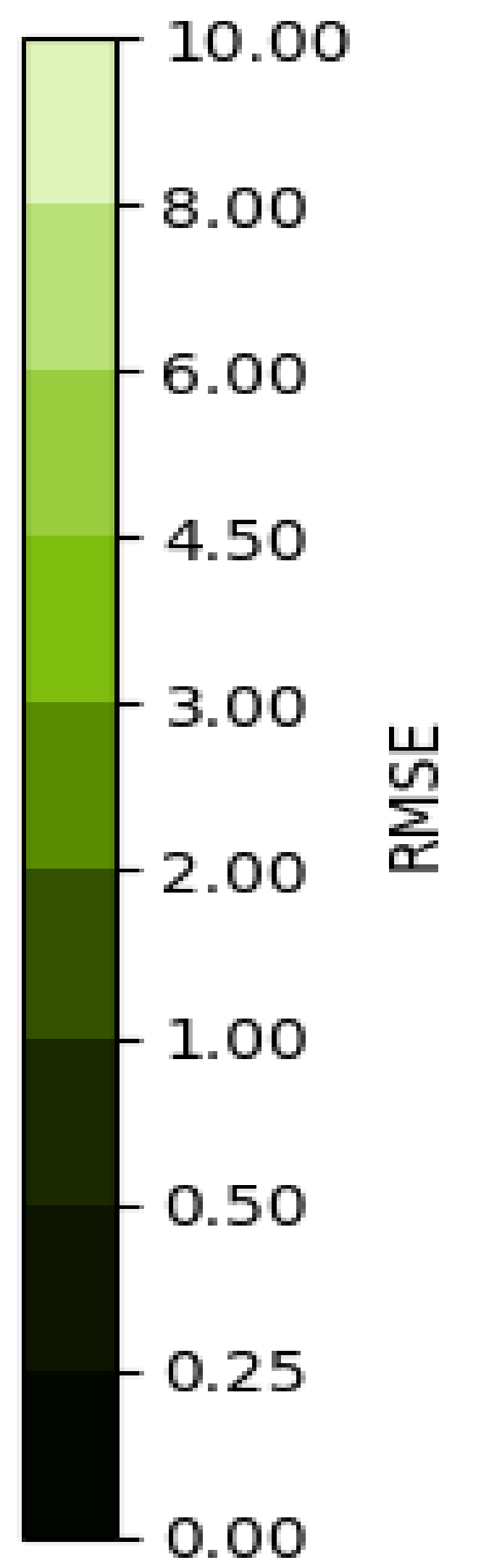
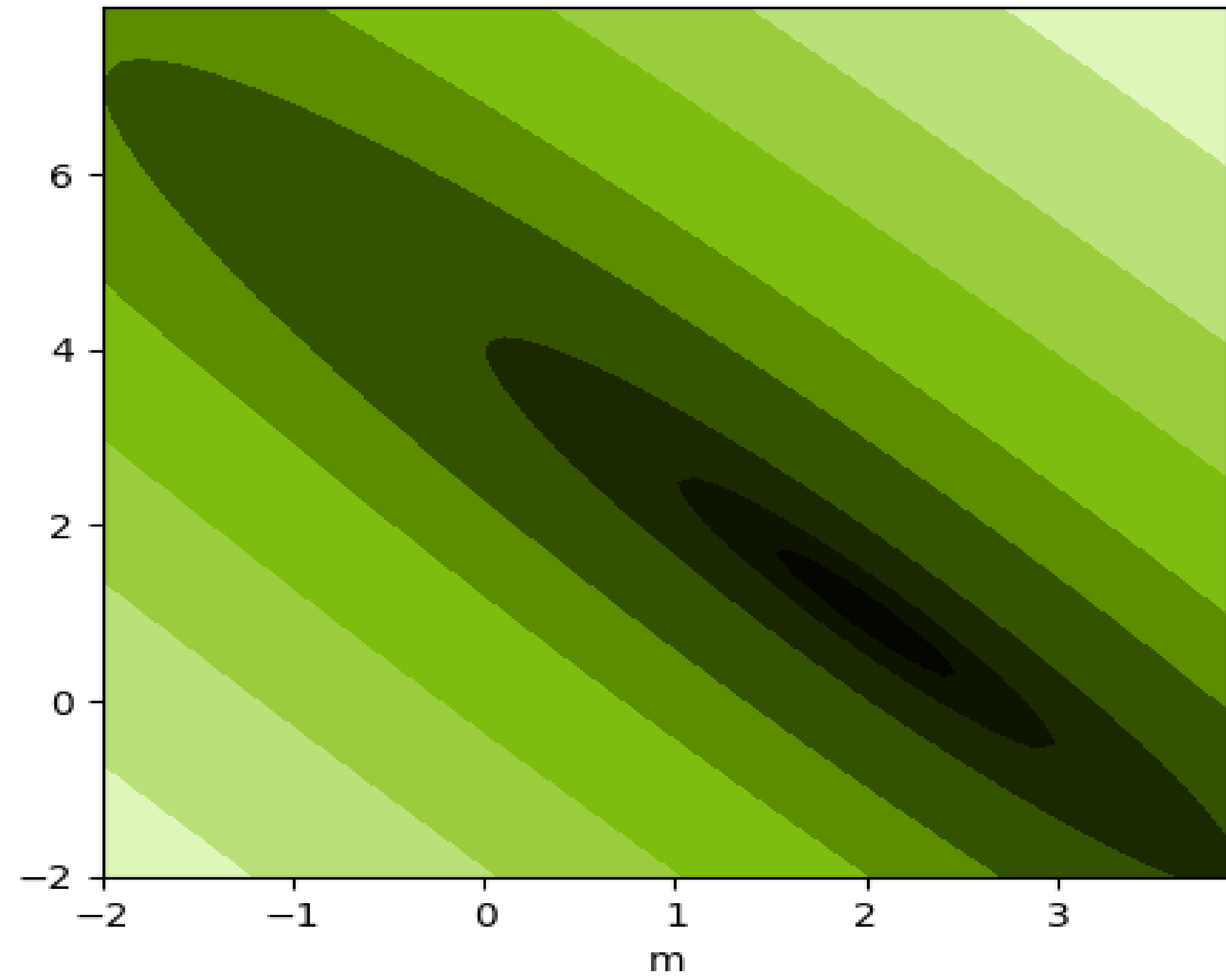
```
1 data = [(1, 3), (2, 5)]
2 m = -1
3 b = 5
4
5
6 def get_rmse(data, m, b):
7     """Calculates Mean Square Error"""
8     n = len(data)
9     squared_error = 0
10    for x, y in data:
11        # Find predicted y
12        y_hat = m*x+b
13        # Square difference between
14        # prediction 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

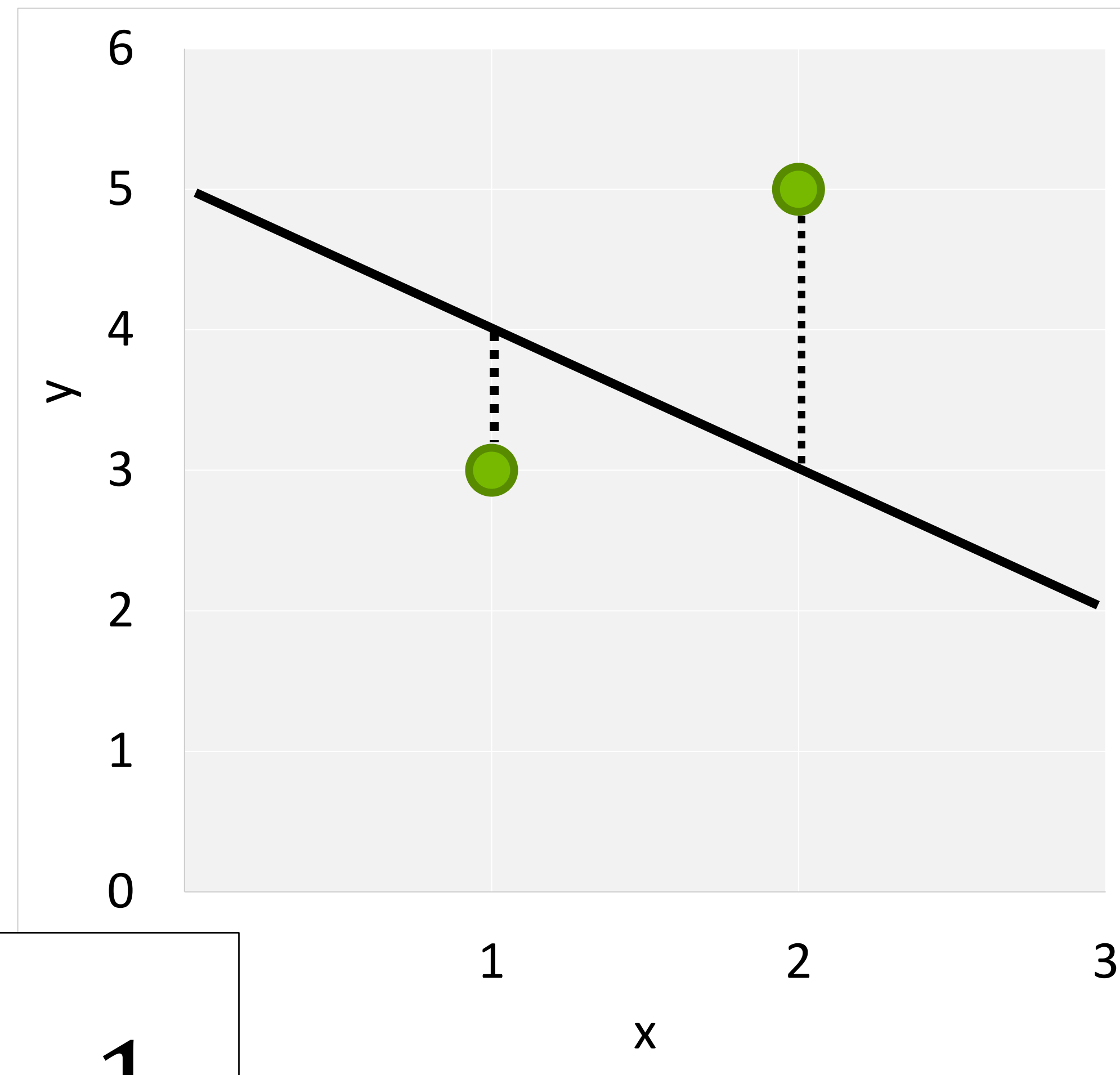
Loss Surface



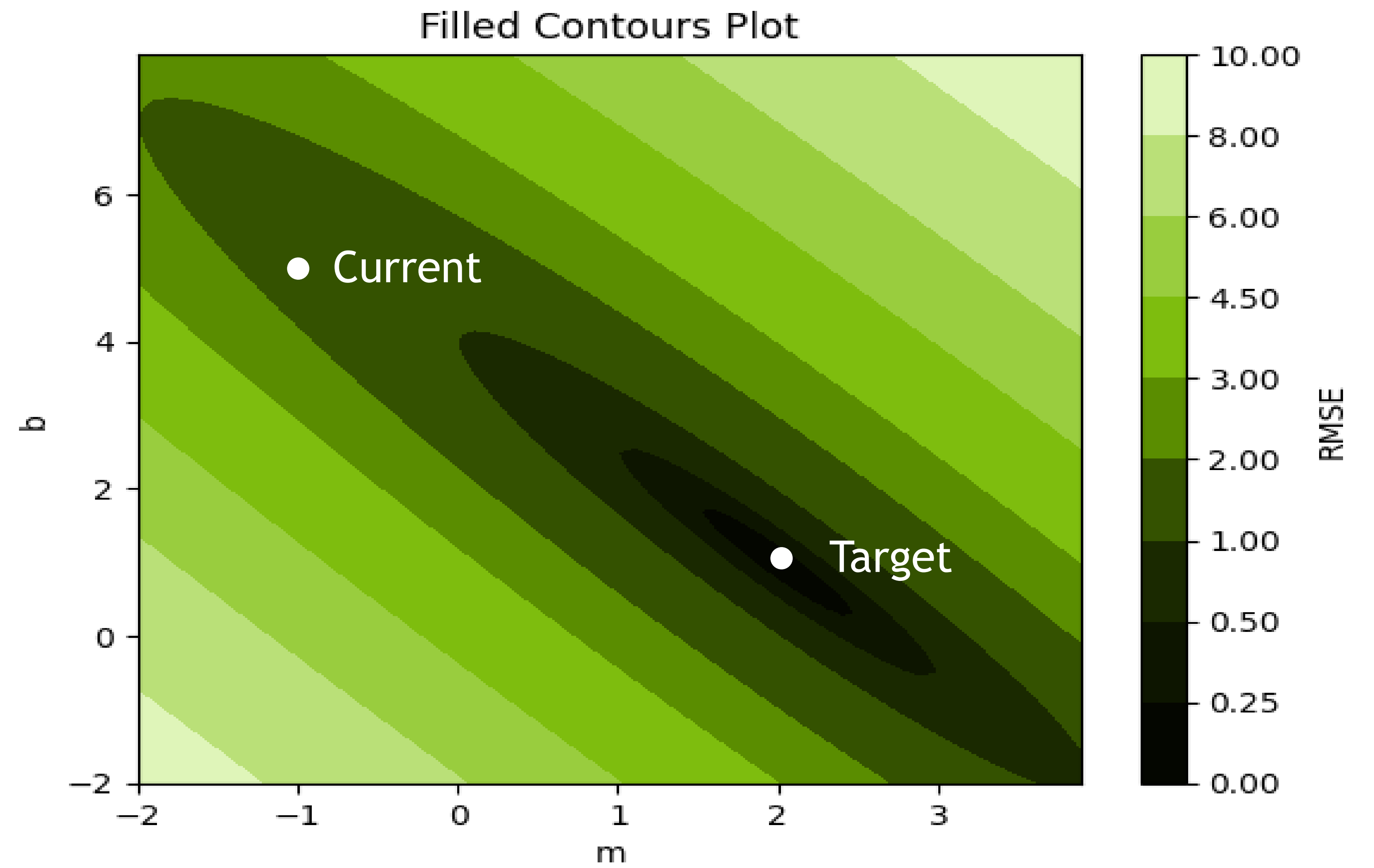
Filled Contours Plot



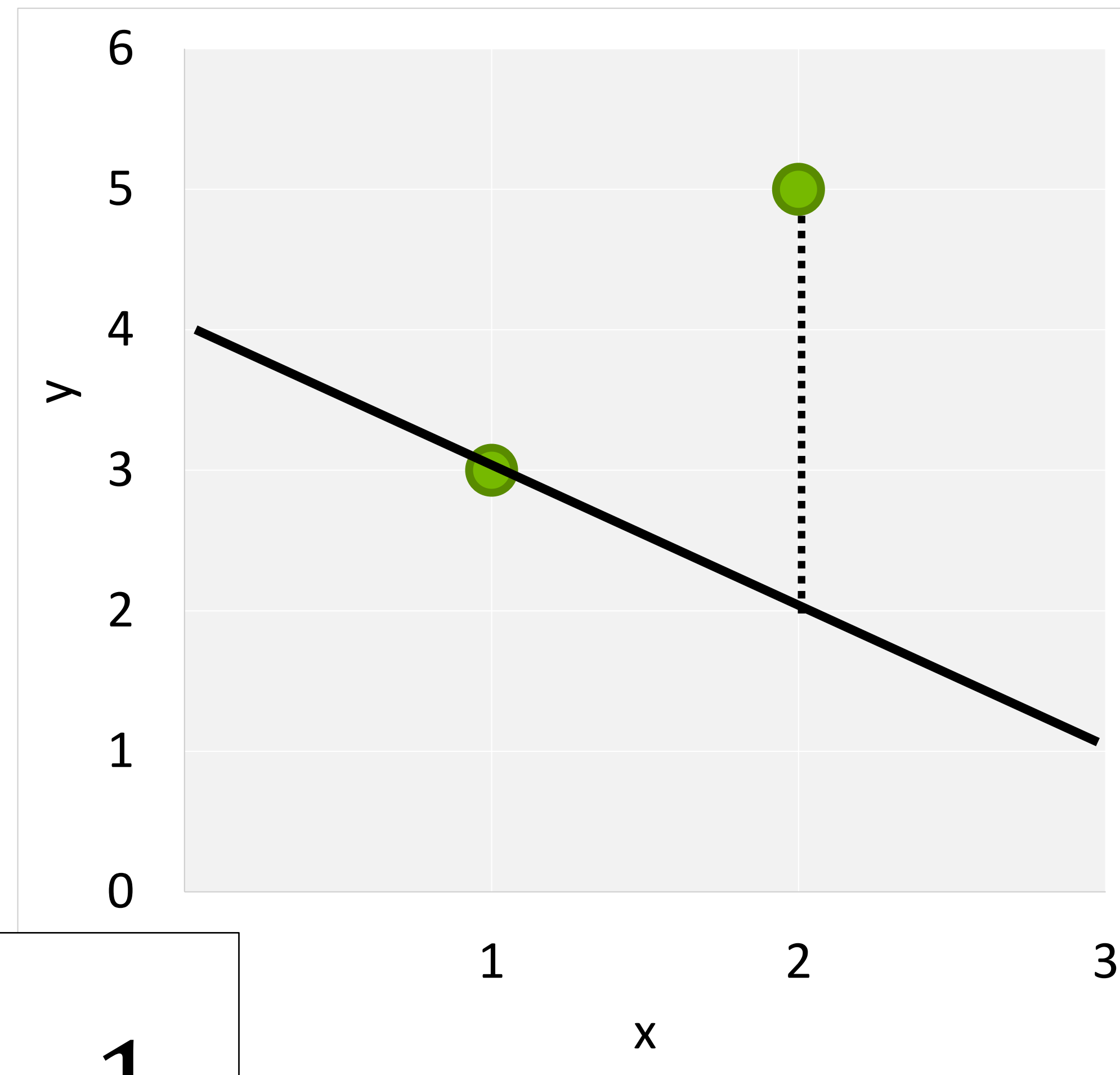
The Loss Curve



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

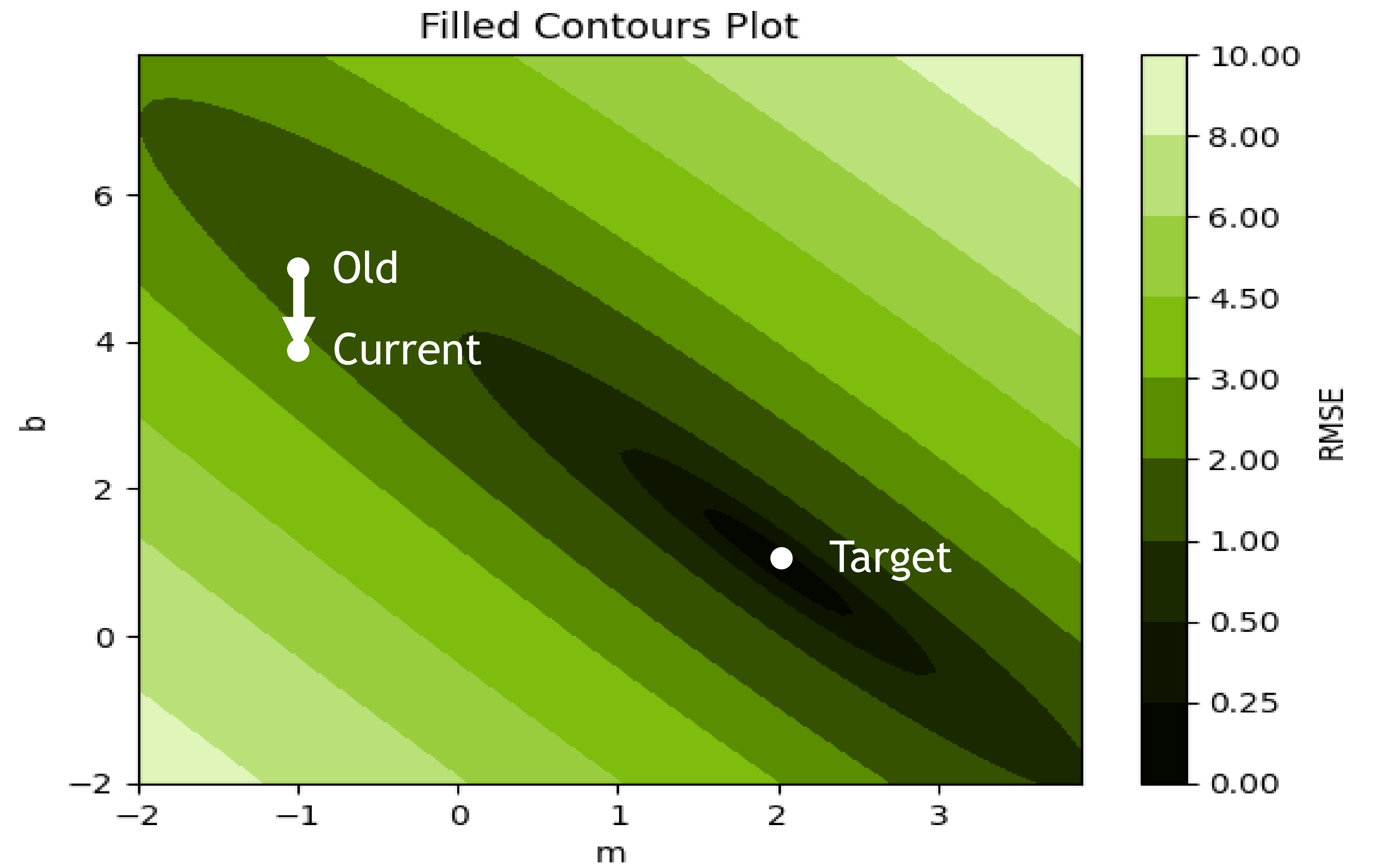


The Loss Curve

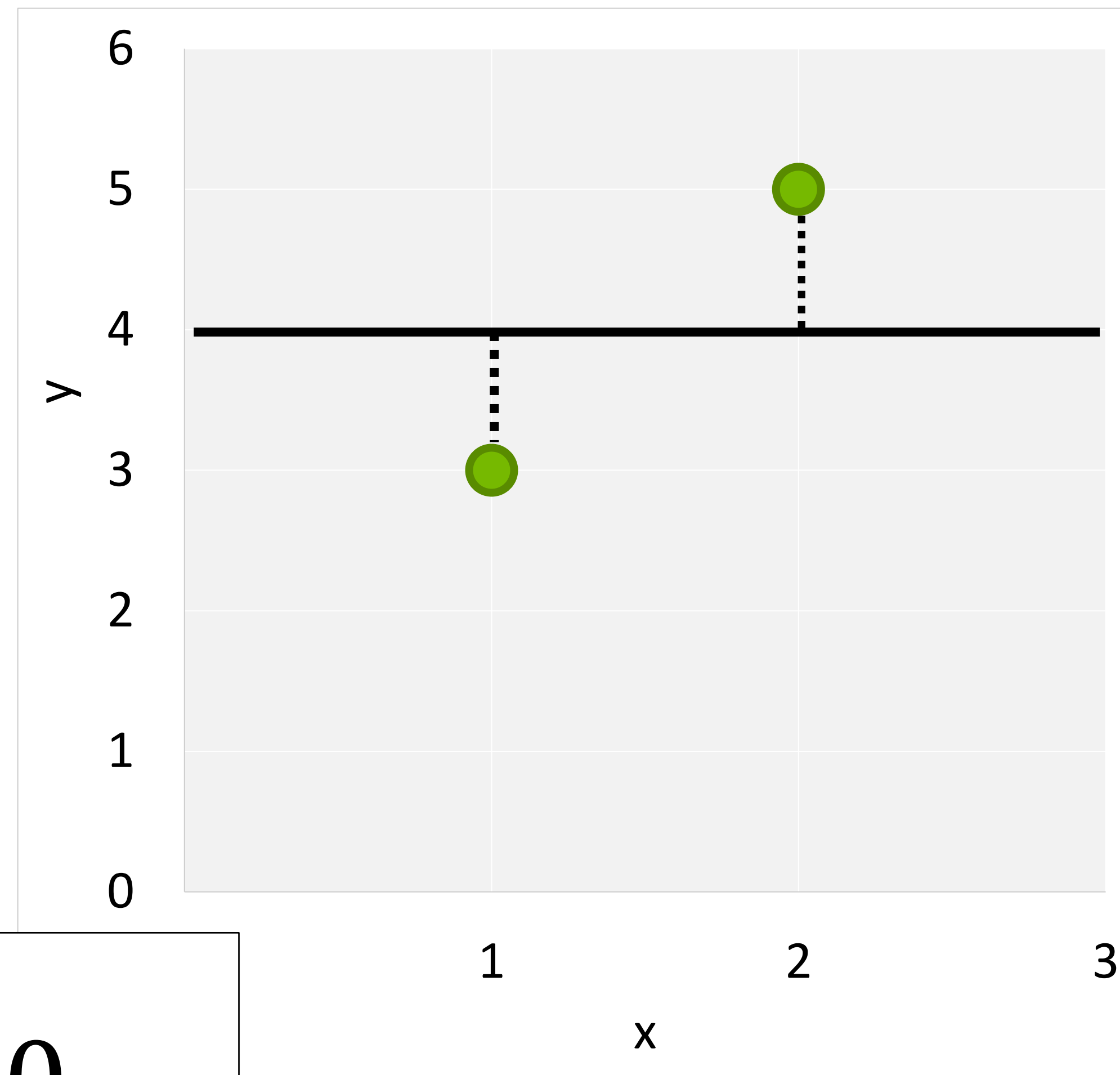


$$m = -1$$

$$b = 4$$

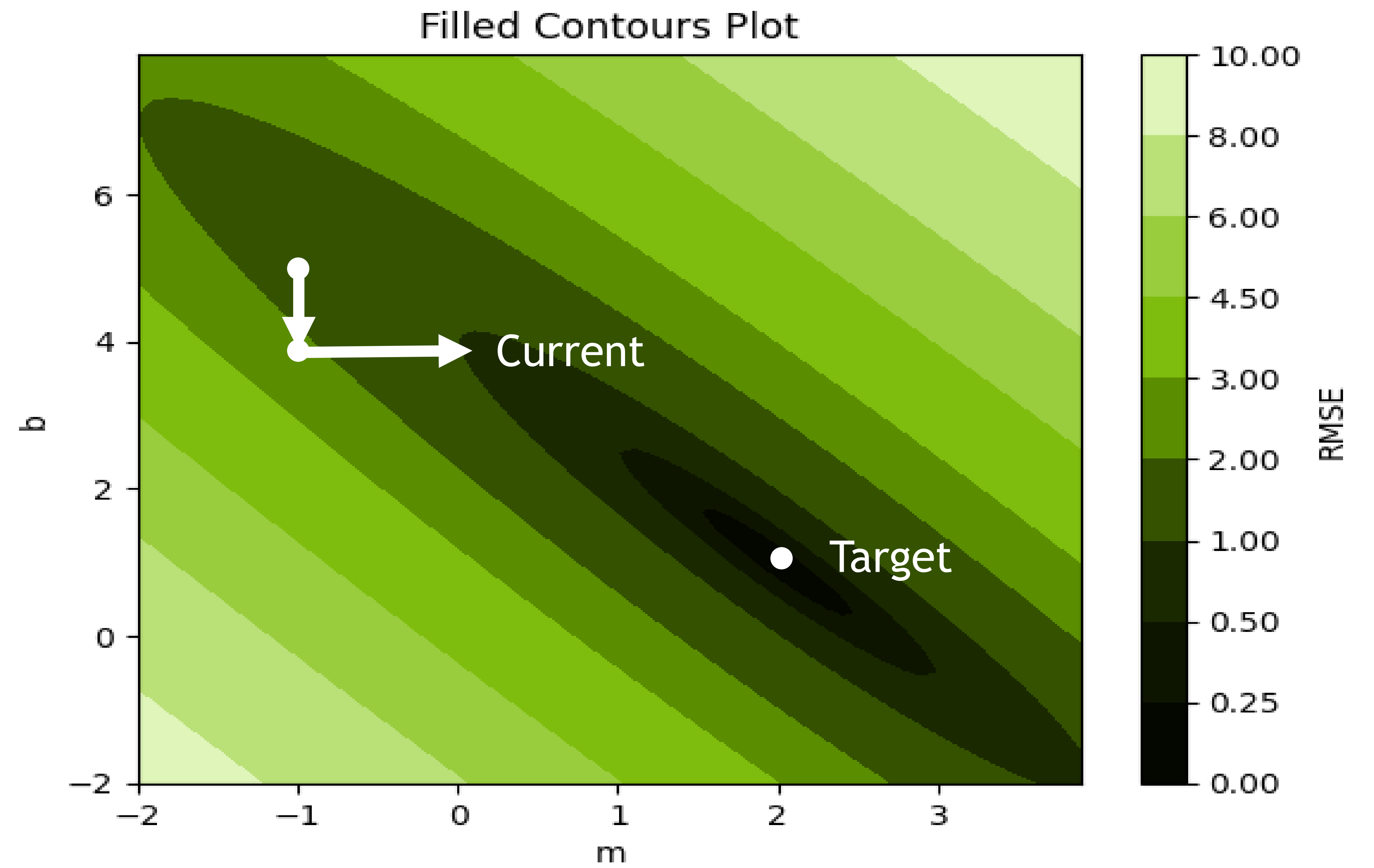


The Loss Curve



$$m = 0$$

$$b = 4$$



The Loss Curve

The Gradient

Which direction loss decreases the most

λ : The learning rate

How far to travel

Epoch

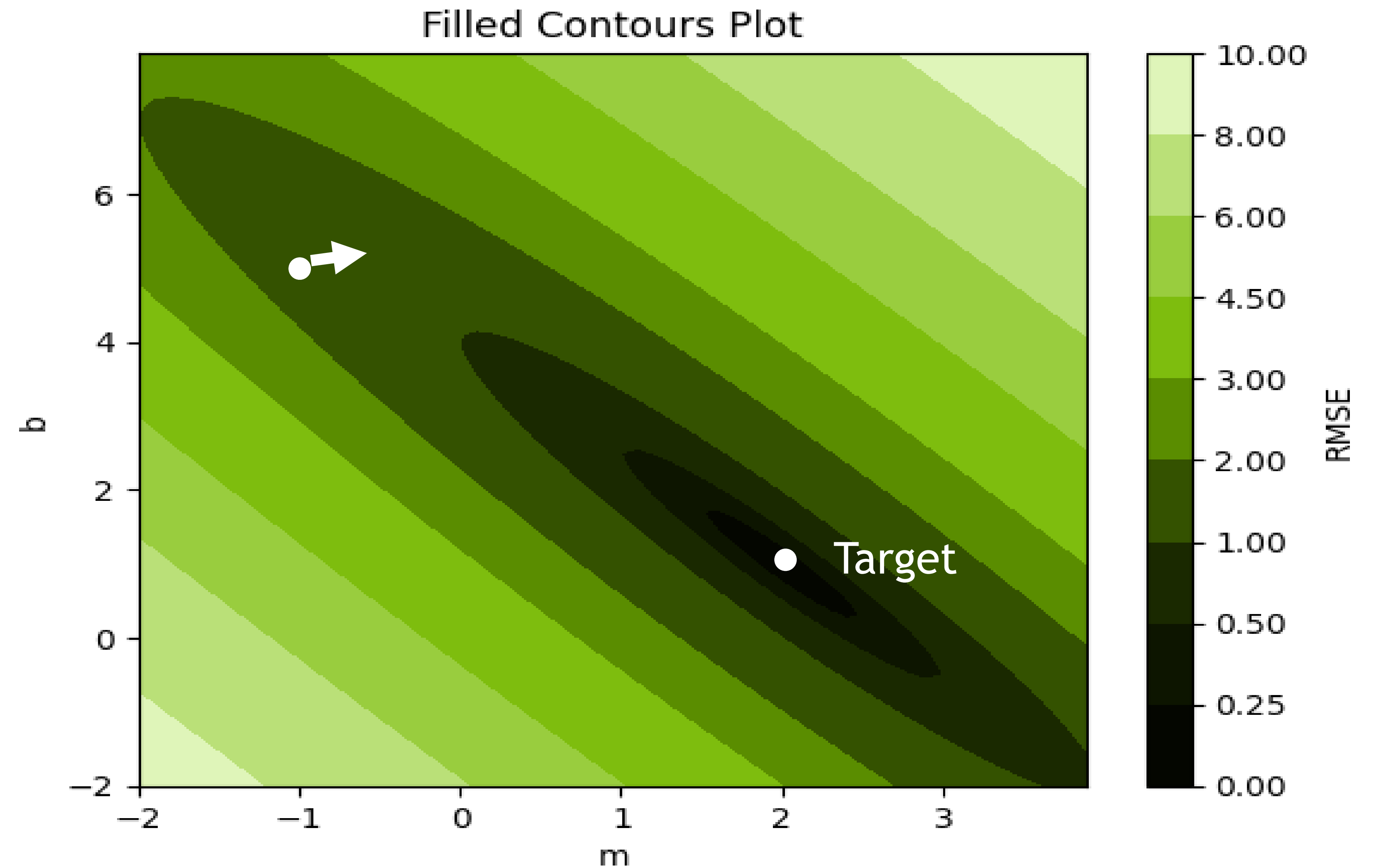
A model update with the full dataset

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

λ : The learning rate

How far to travel

Epoch

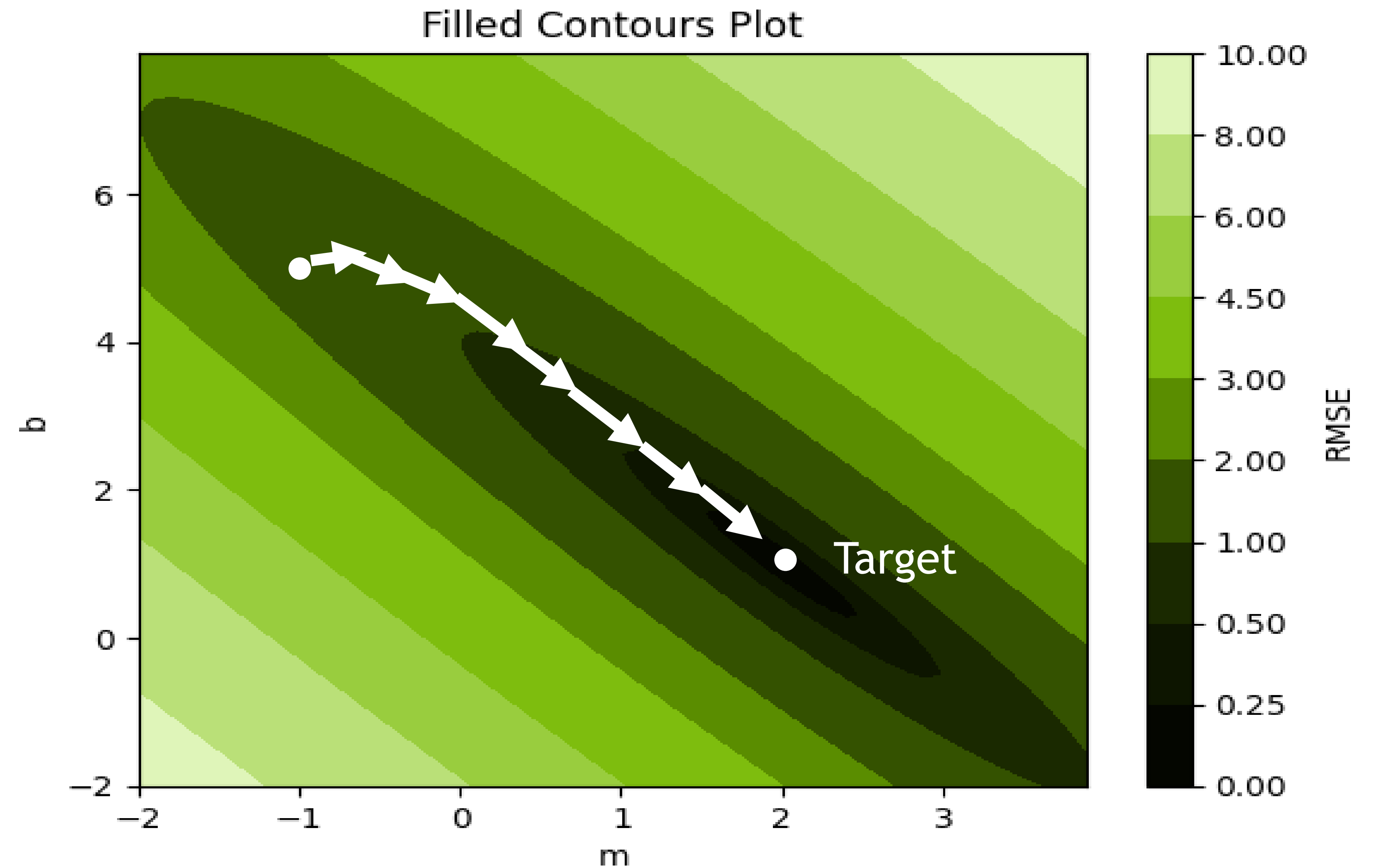
A model update with the full dataset

Batch

A sample of the full dataset

Step

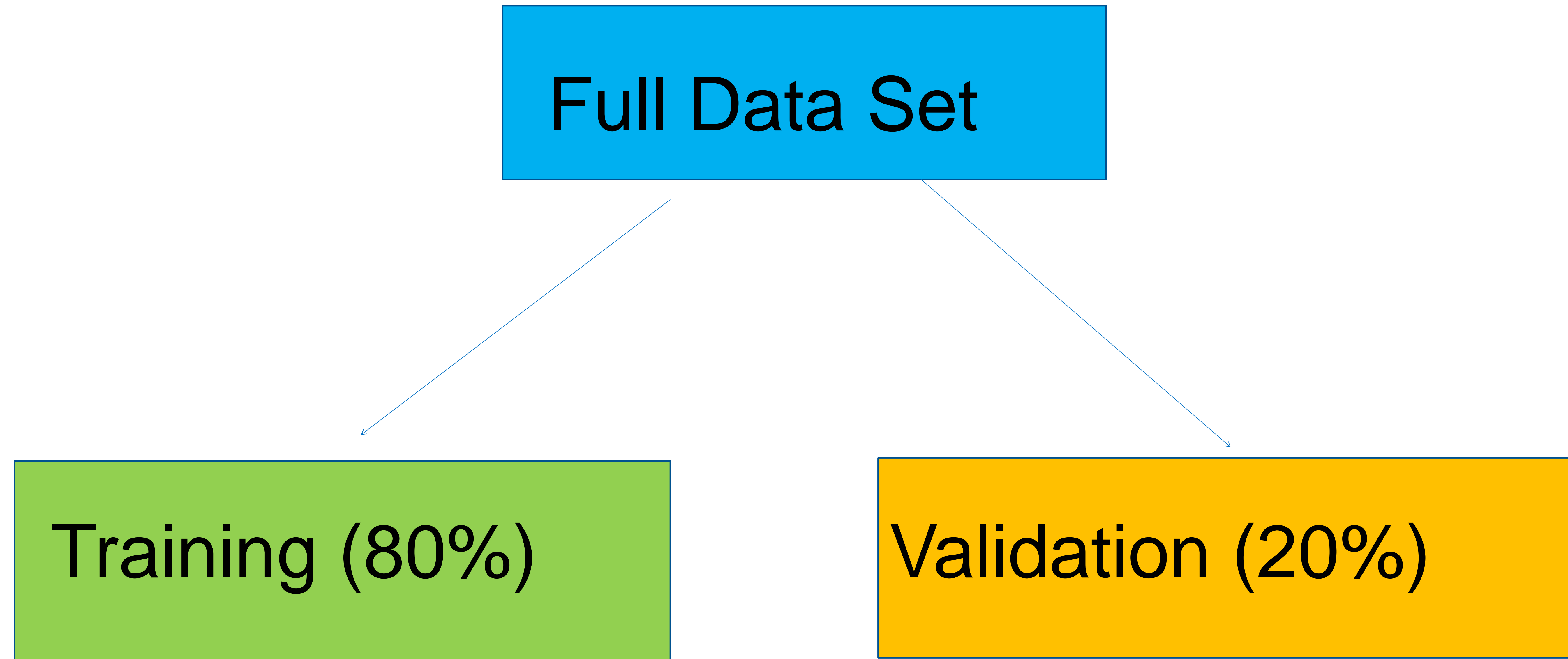
An update to the weight parameters



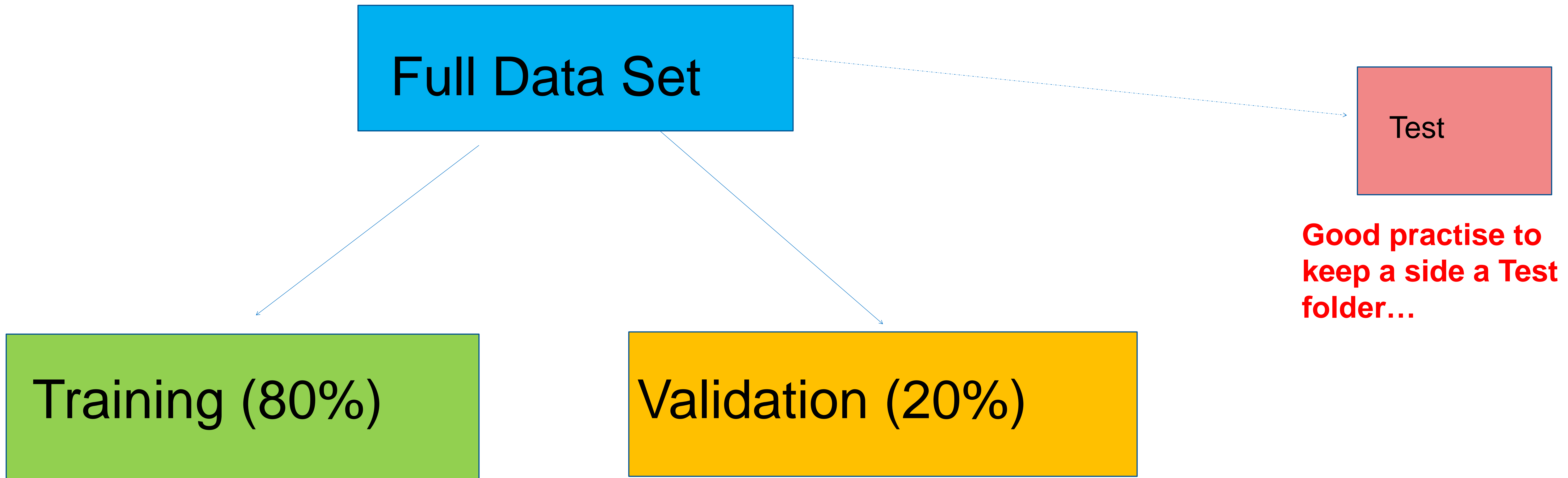
Definitions 1

- **1 Iteration:** 1 Forward pass + 1 Backward pass
- **1 Epoch:** 1 Forward pass + 1 Backward pass for **ALL** training samples.
- **Batch Size:** Number of training samples
- **Mini-batch** (often called Batch Size!): Number of training samples used in 1 iteration
- **Learning rate:** determines to what extent newly acquired information overrides old information
- **Loss:** difference between prediction and correct output
- **Accuracy:** $\frac{\text{number of correct predictions}}{\text{total number of predictions}}$

Definitions 2



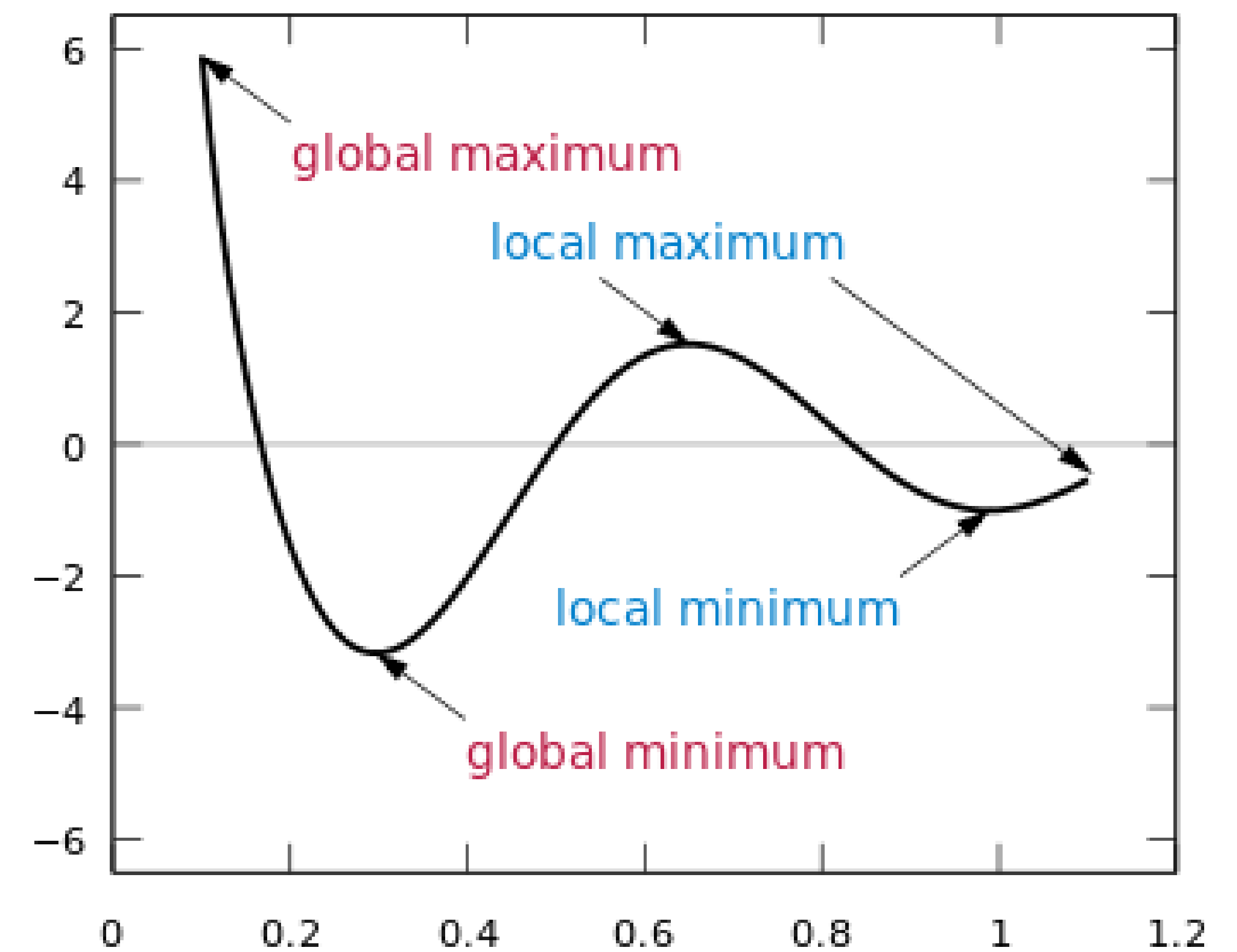
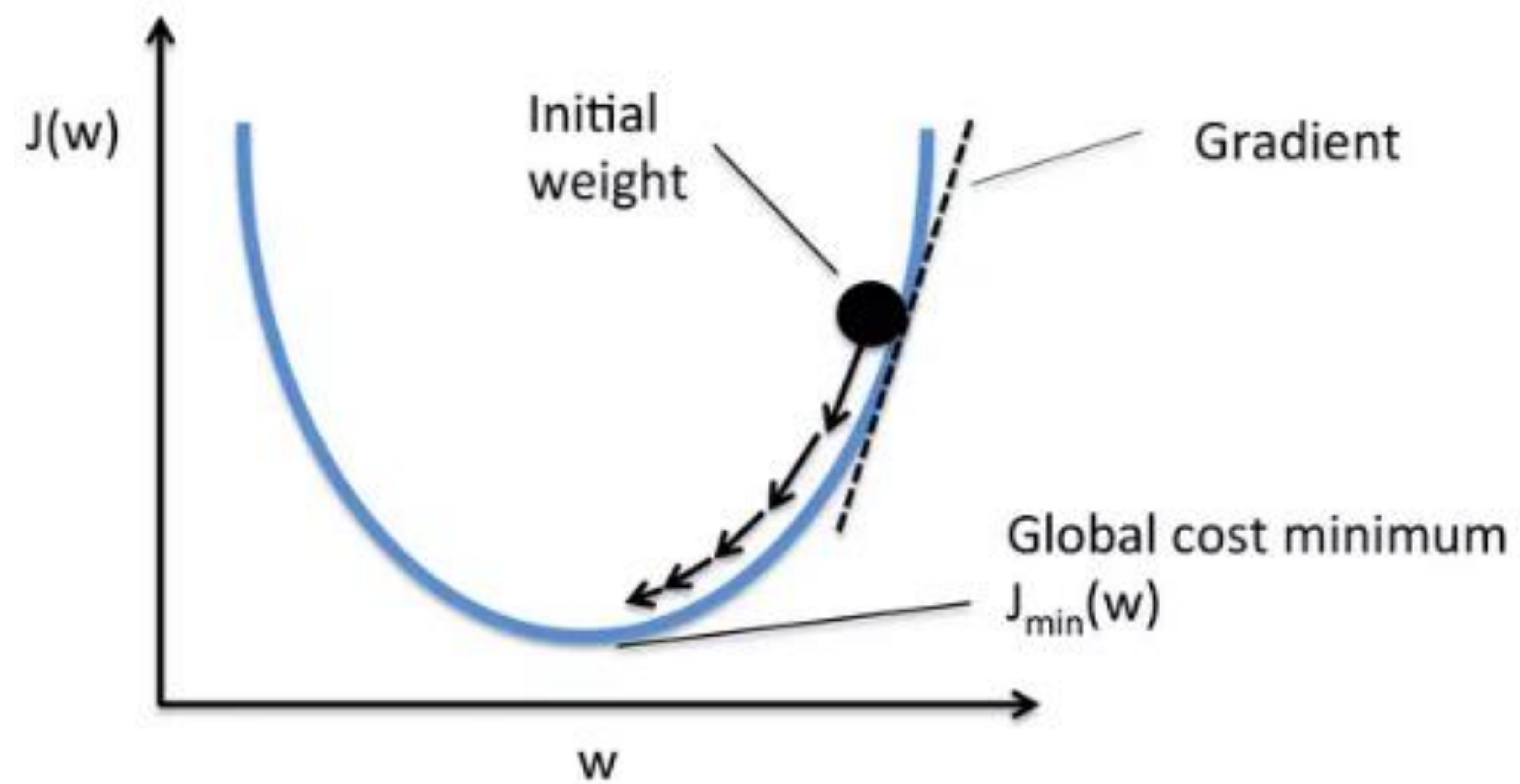
Definitions 2



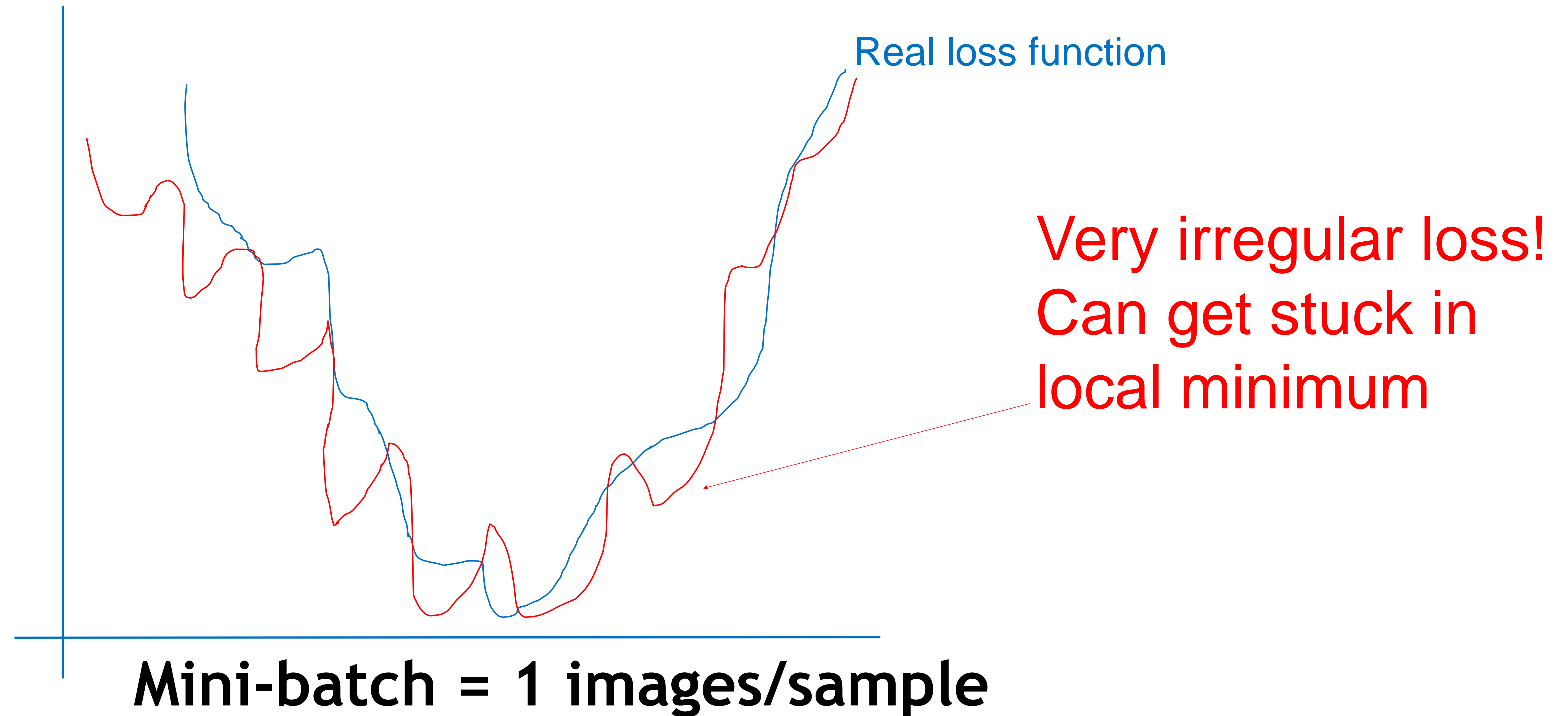
Definitions 3

- **Parameters:** weights of the Neural Network. Their values are derived via training
- **Hyper-parameters:** are values set before the **learning** process begins.
 - Ex: number of epochs
 - mini-batch size
 - learning rate

Non convex Cost functions

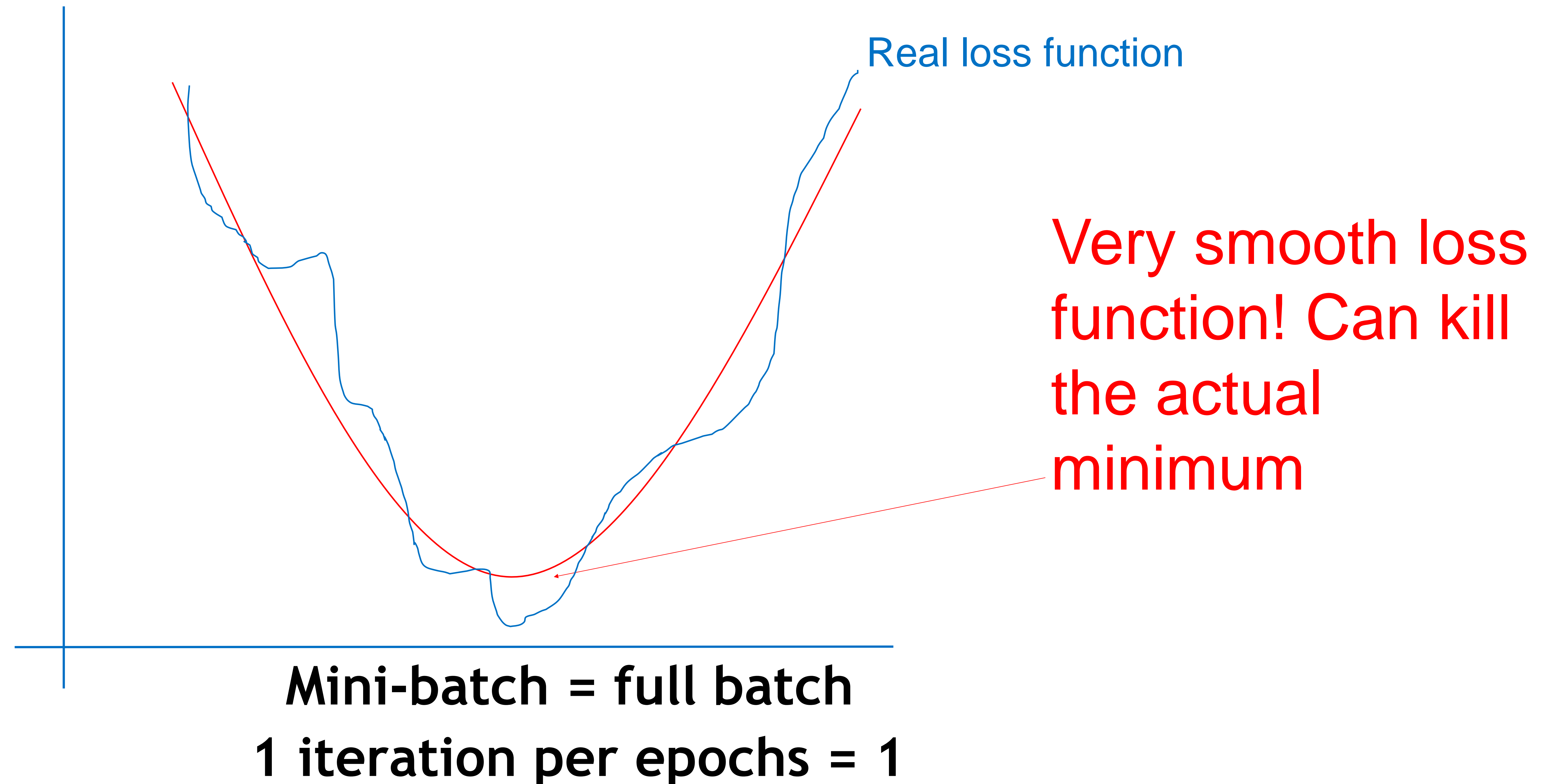


Stochastic Gradient Descent (SGD) method

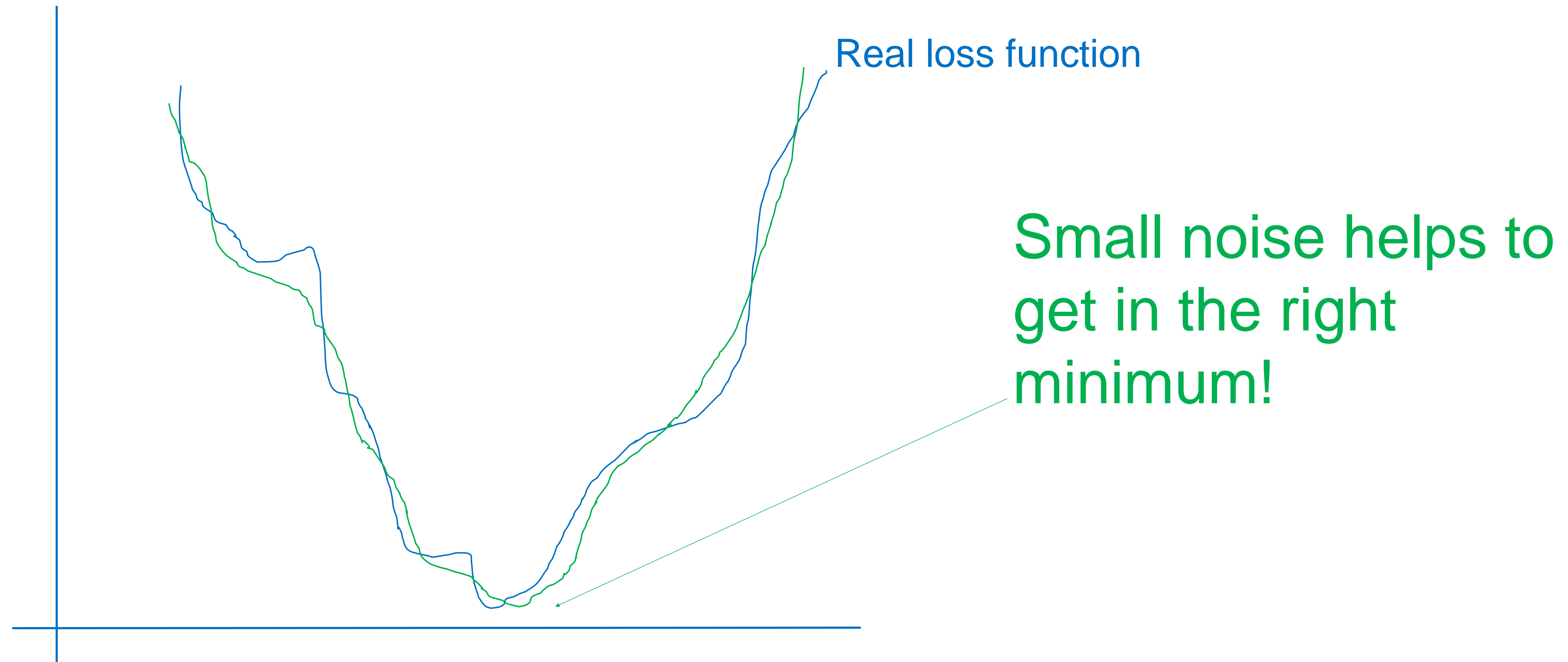


N iterations per epochs = n samples in the batch

Batch Gradient Descent (BGD) method



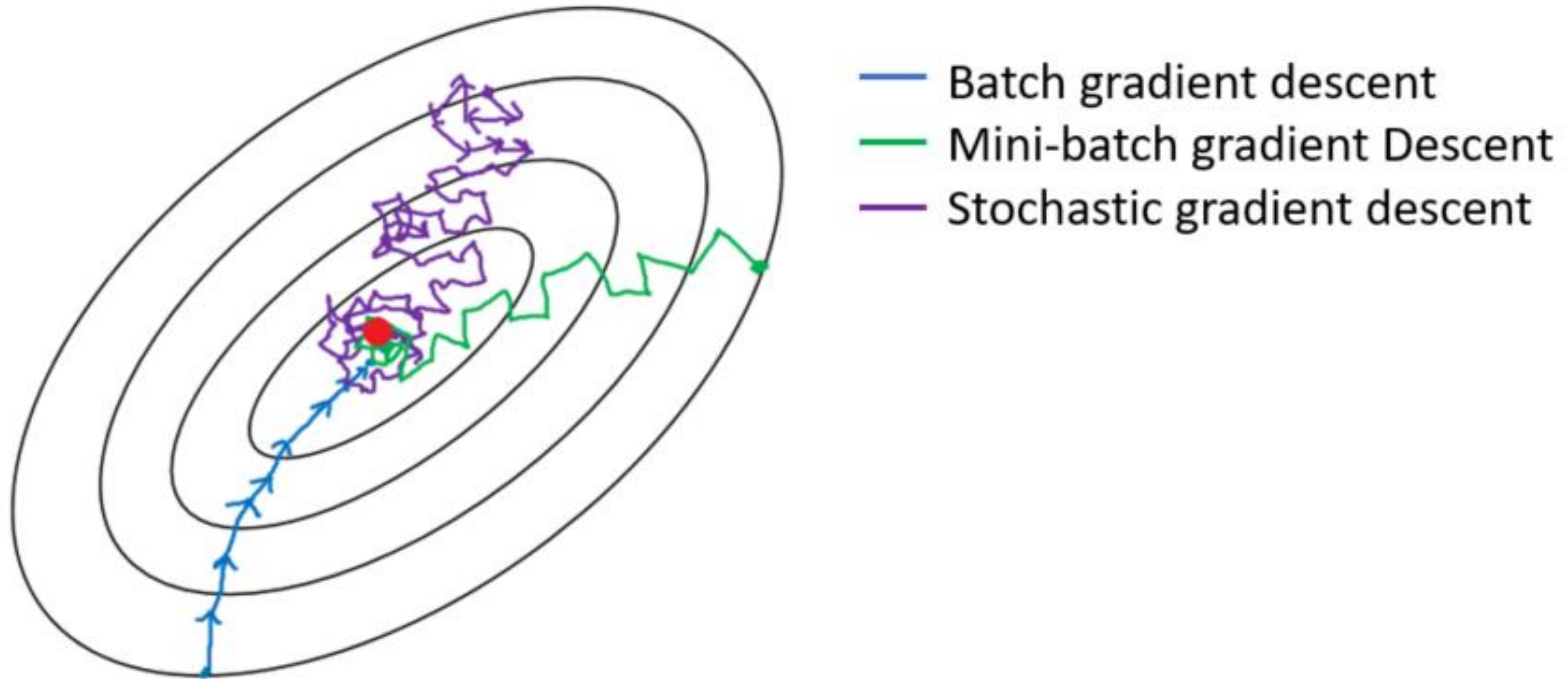
Mini-batch Stochastic Gradient Descent (mSGD) method



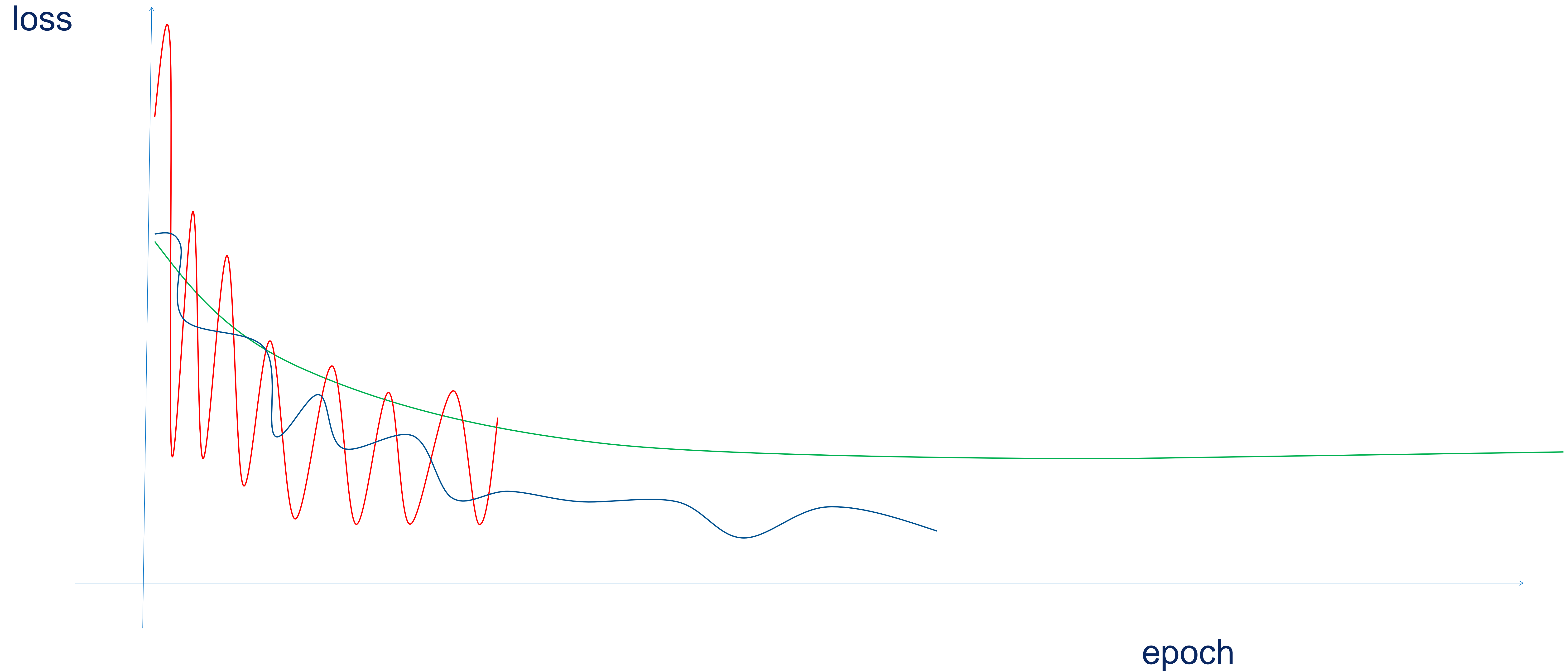
The mini-batch is equal to a subset of the batch size (usually 32/64/128...)

$$N \text{ iterations per epochs} = N \text{ samples in the batch} / \text{mini-batch size}$$

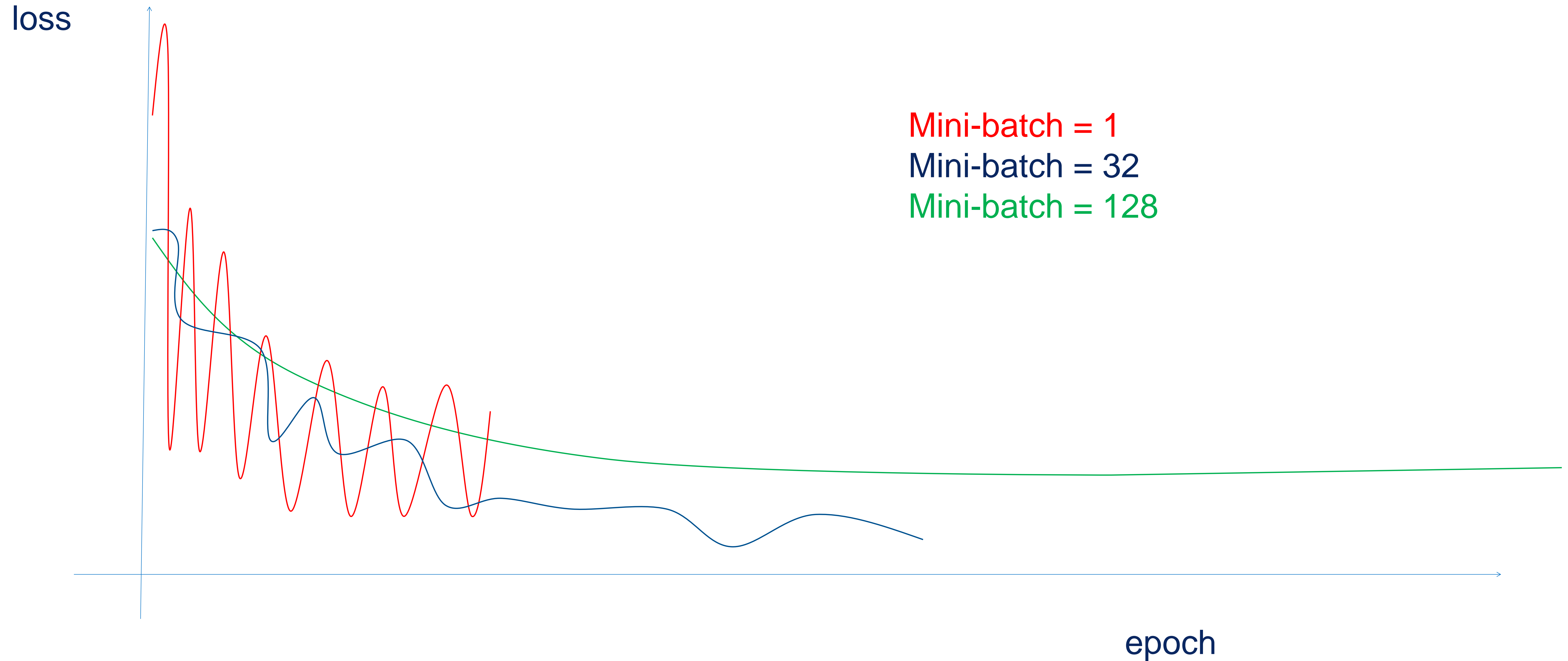
Effect of different mini-batch size



Which curve has the largest batch size?

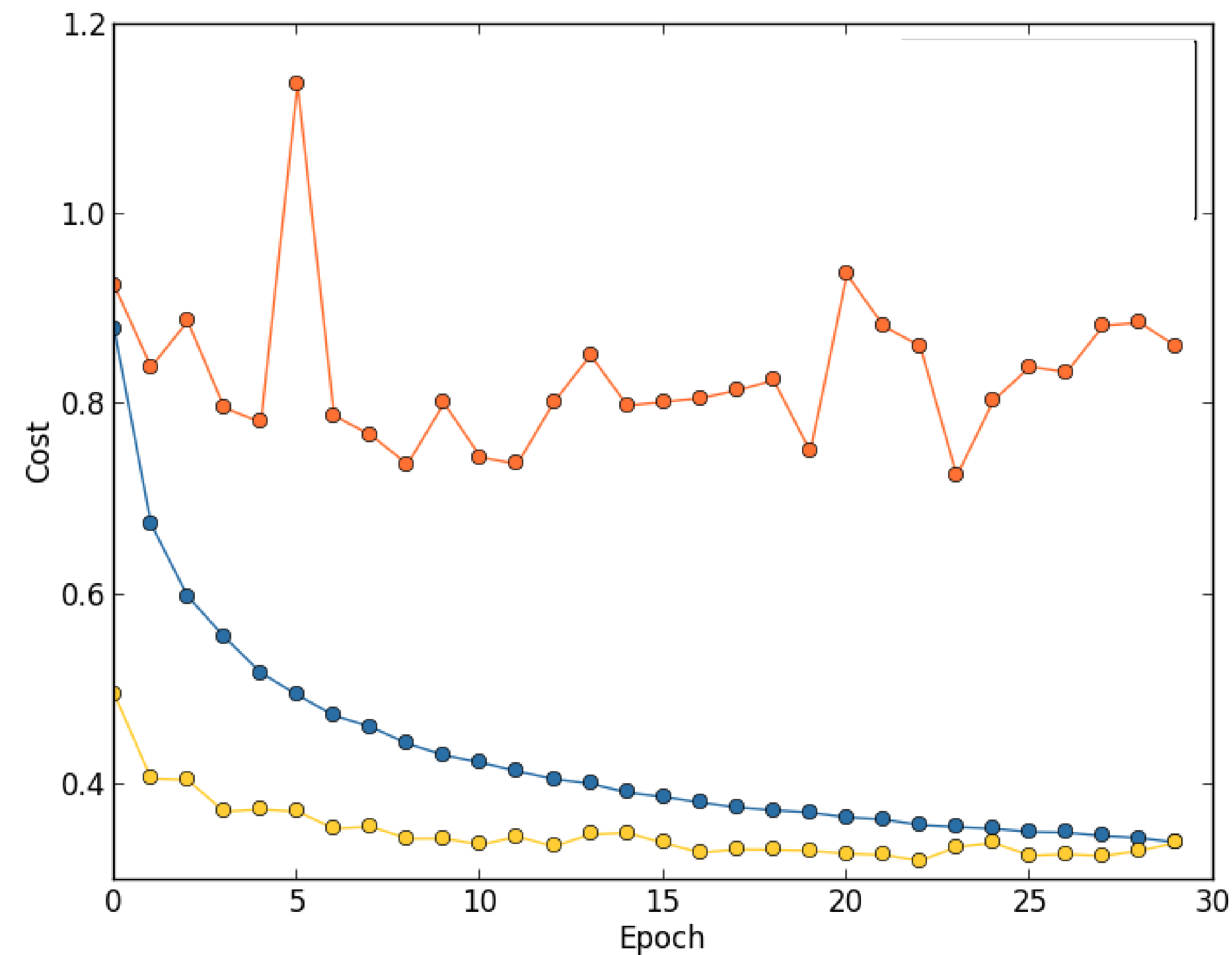


Which curve has the largest batch size?



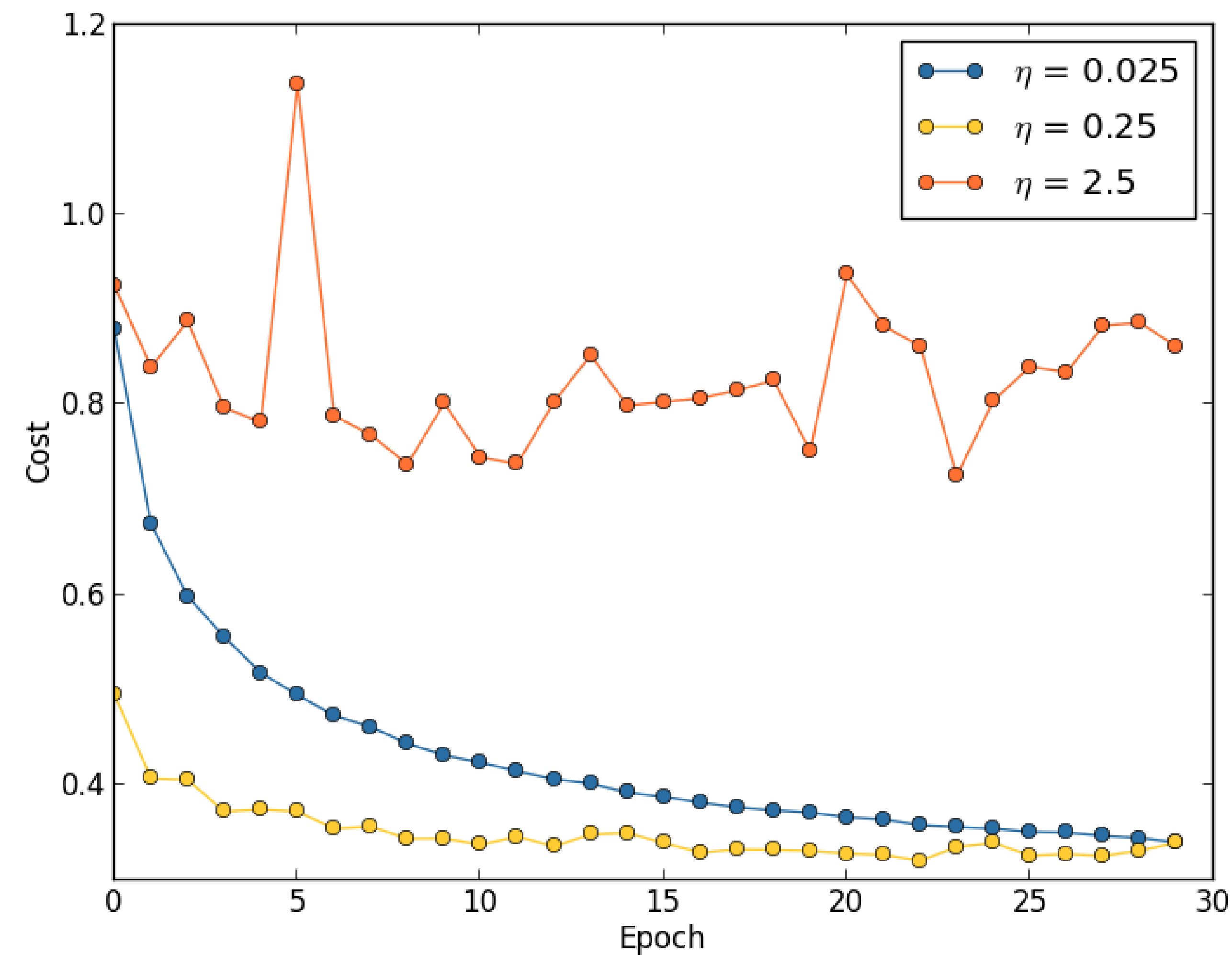
Which curve has the smallest learning rate?

$$w_t = w_{t-1} + \eta \frac{dE}{dw_t}$$

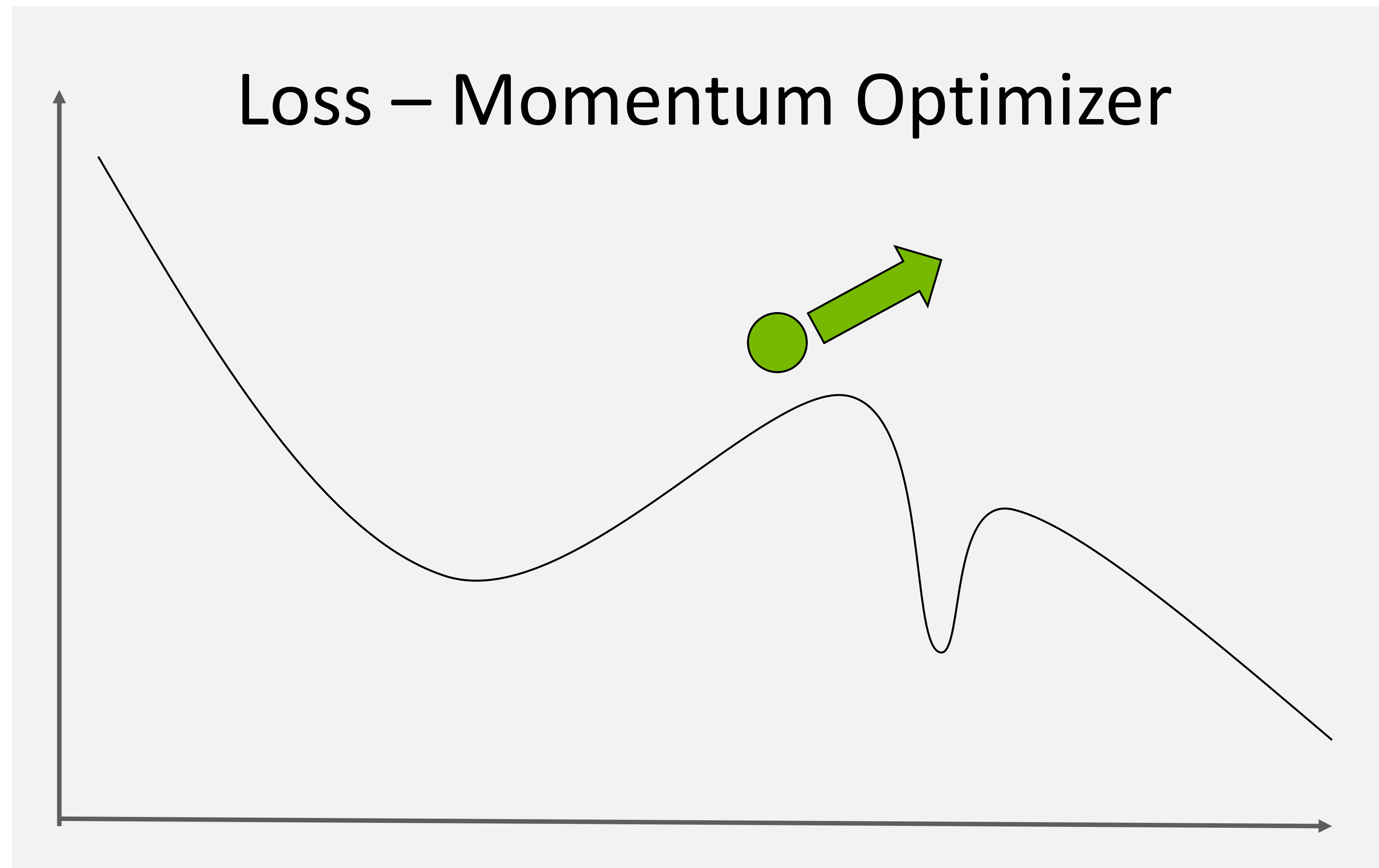


Which curve has the smallest learning rate?

$$w_t = w_{t-1} + \eta \frac{dE}{dw_t}$$



Optimizers

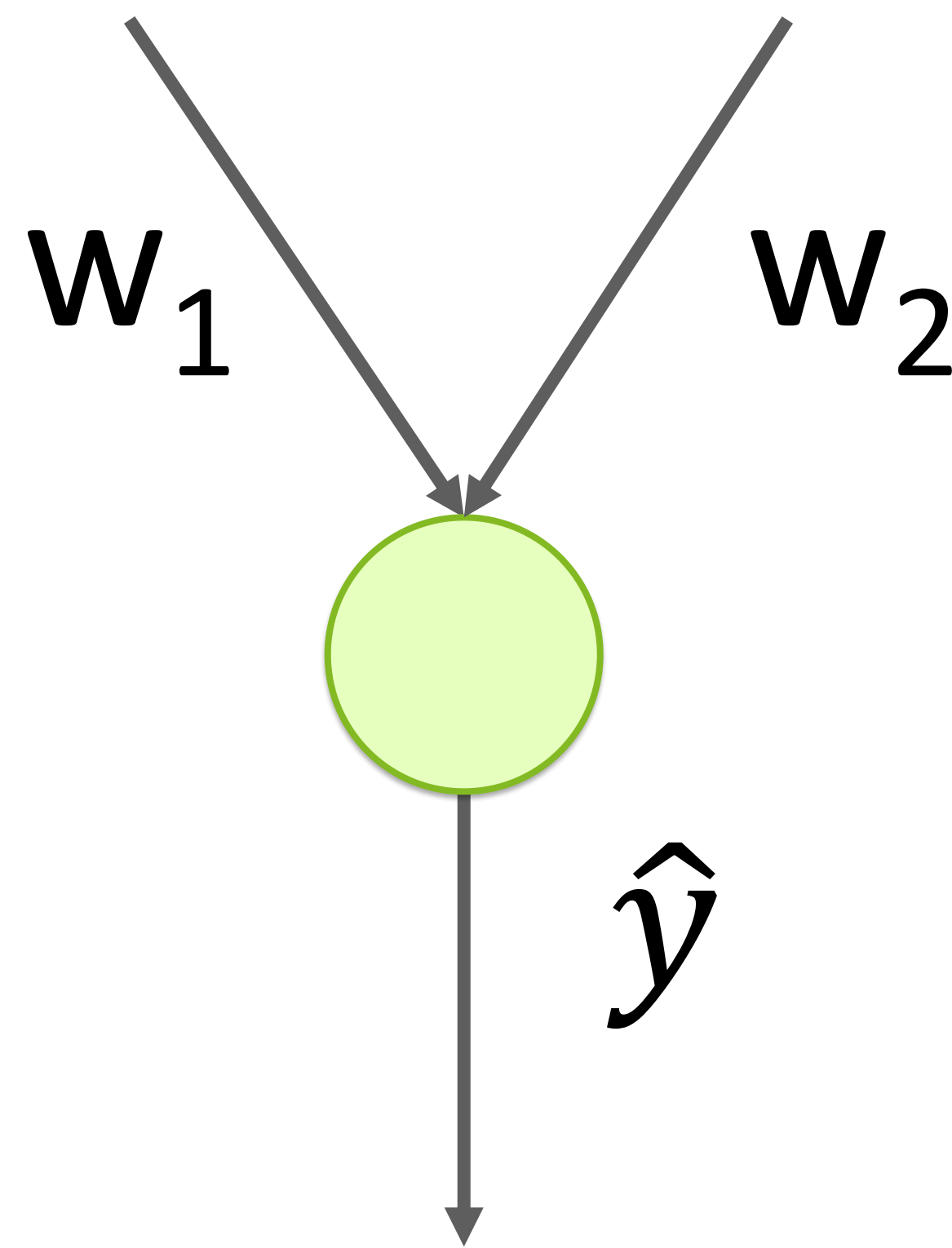


- Adam
- Adagrad
- RMSprop
- SGD



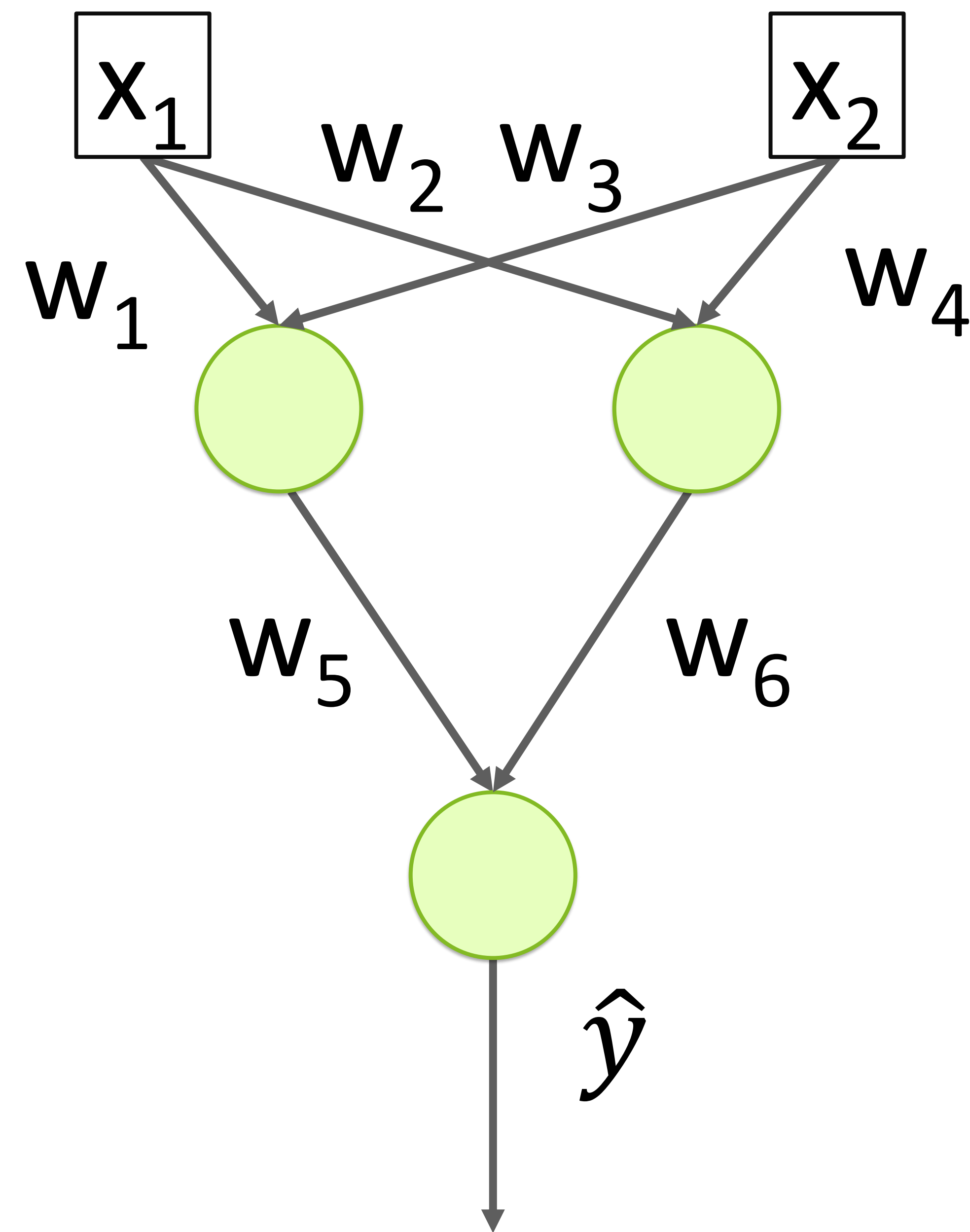
From Neuron to Network

Building a Network



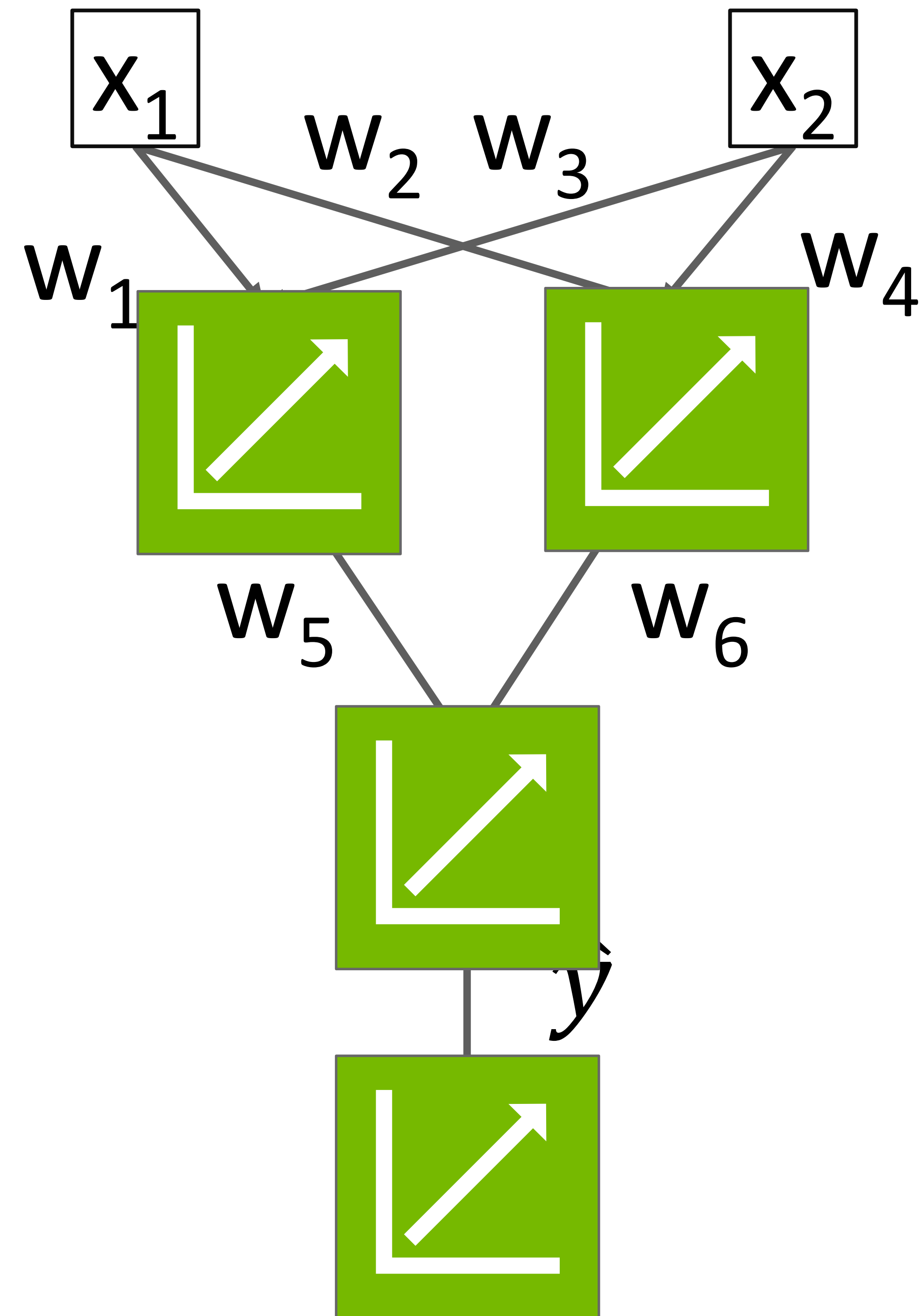
- Scales to more inputs

Building a Network



- Scales to more inputs
- Can chain neurons

Building a Network



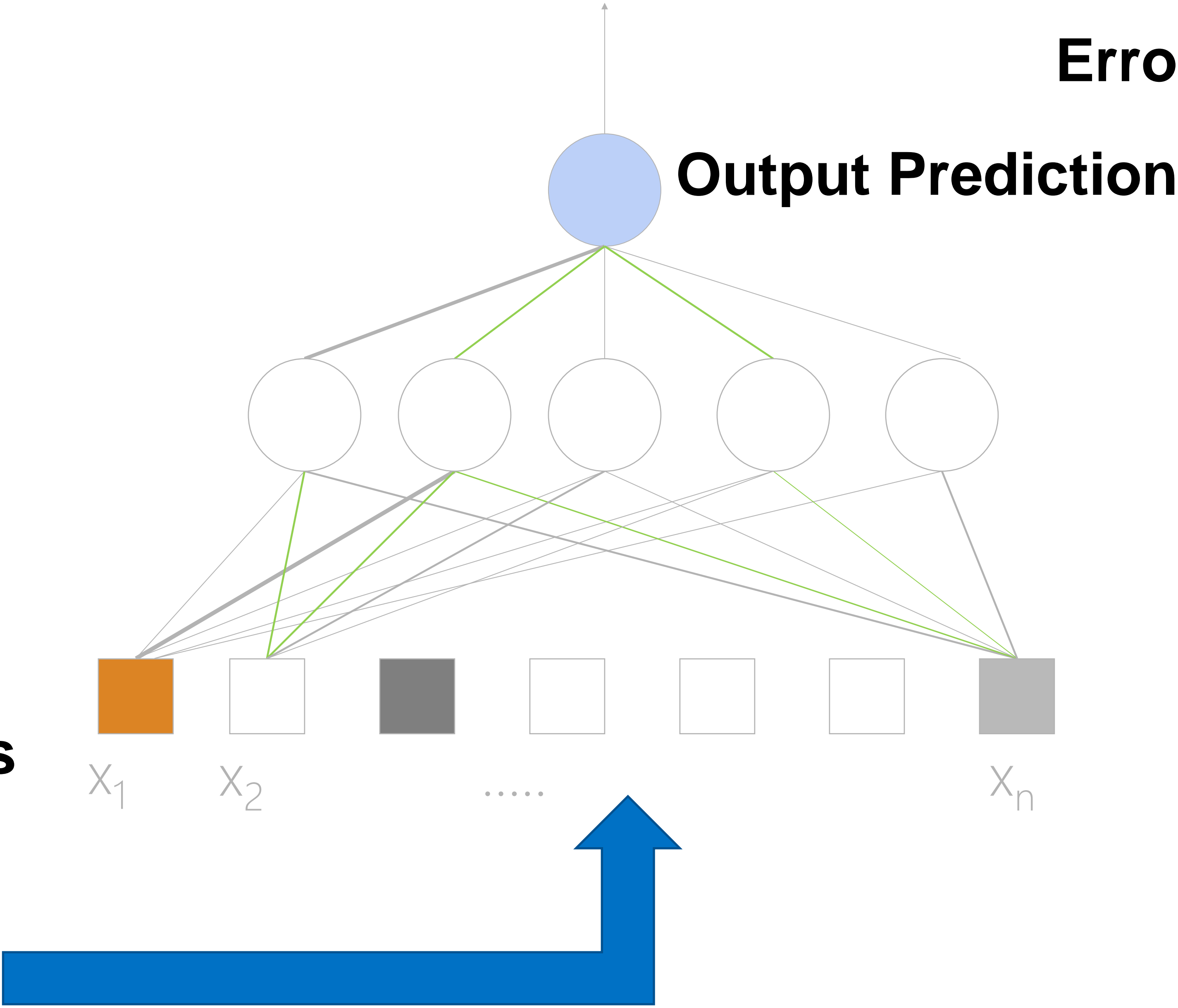
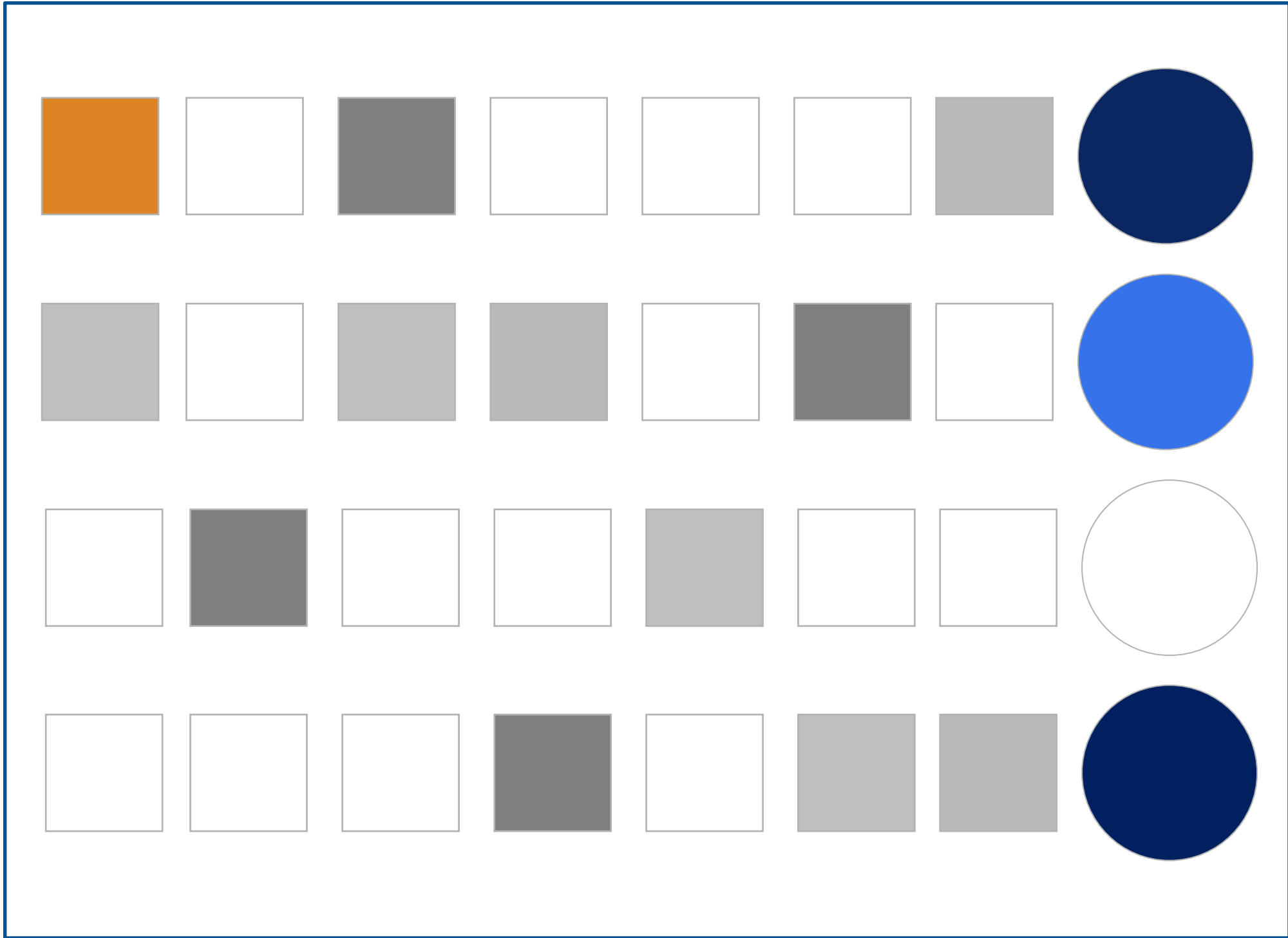
- Scales to more inputs
- Can chain neurons
- If all regressions are linear, then output will also be a linear regression

Learning Principle

Dataset

Inputs

Outputs

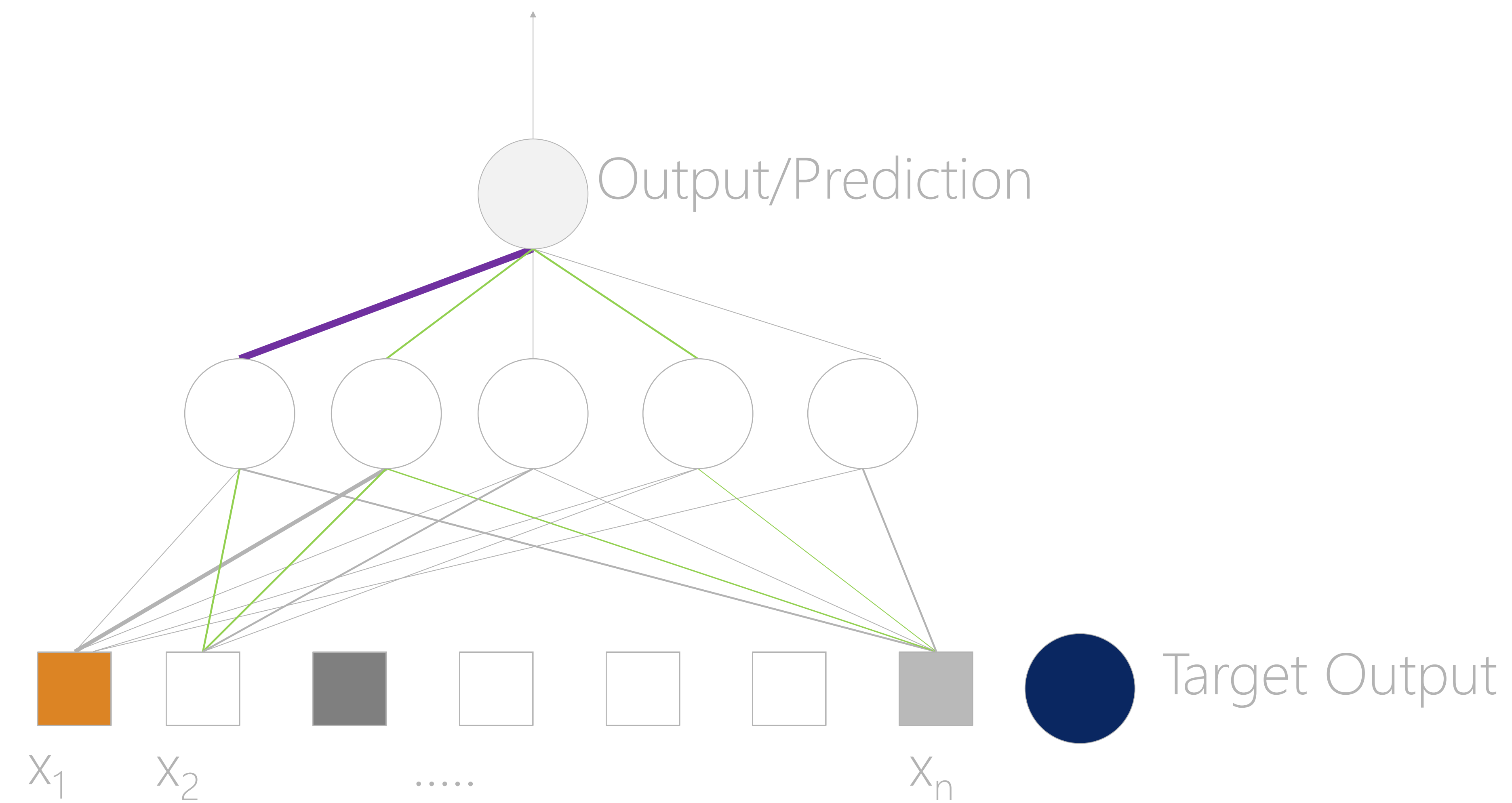



Error:

 -  = 5

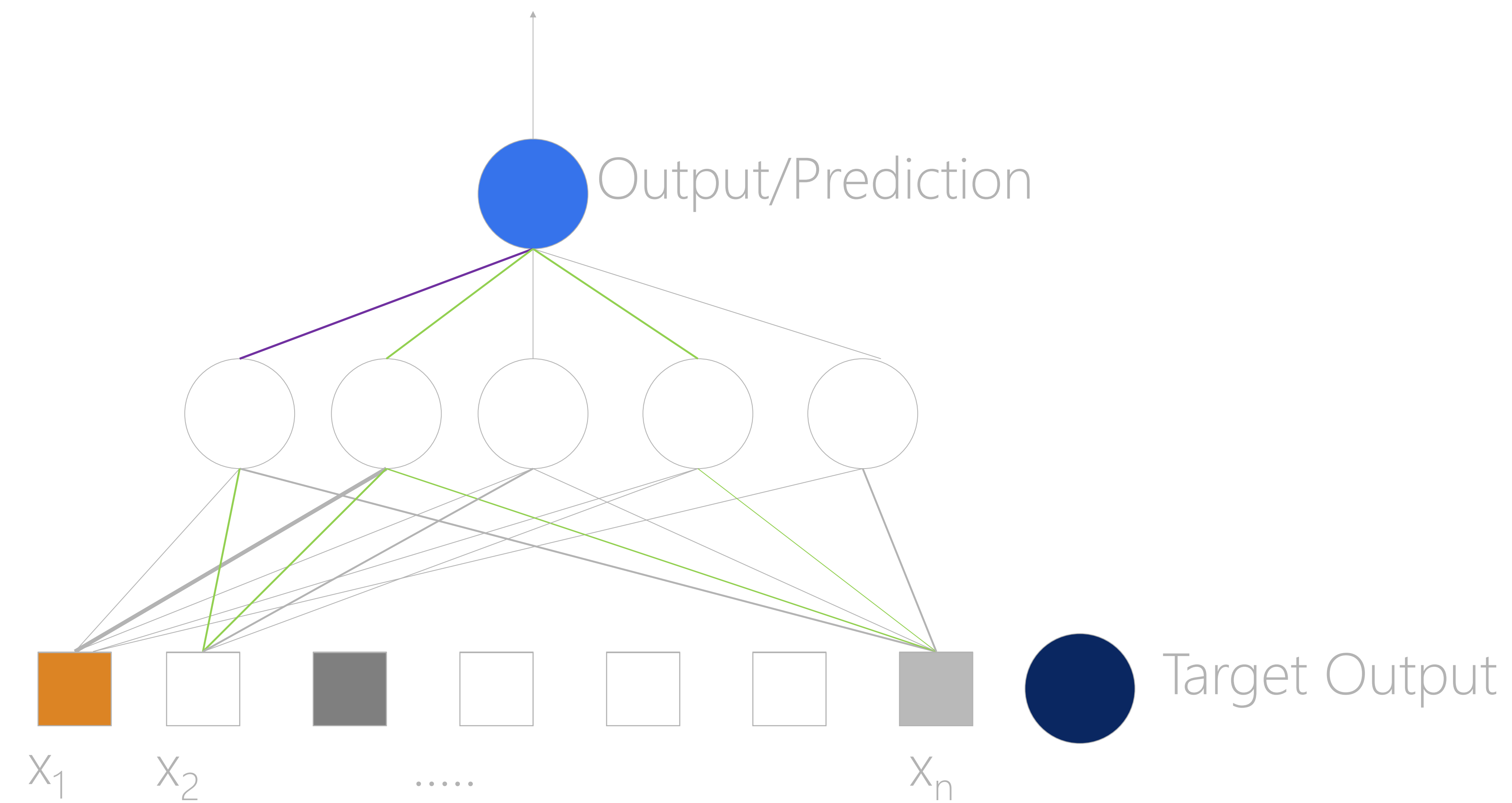
 **Target Output**



Learning Principle: adjust weights



Error:  -  = 15

Learning Principle: adjust weights



Error:  -  = 2.5

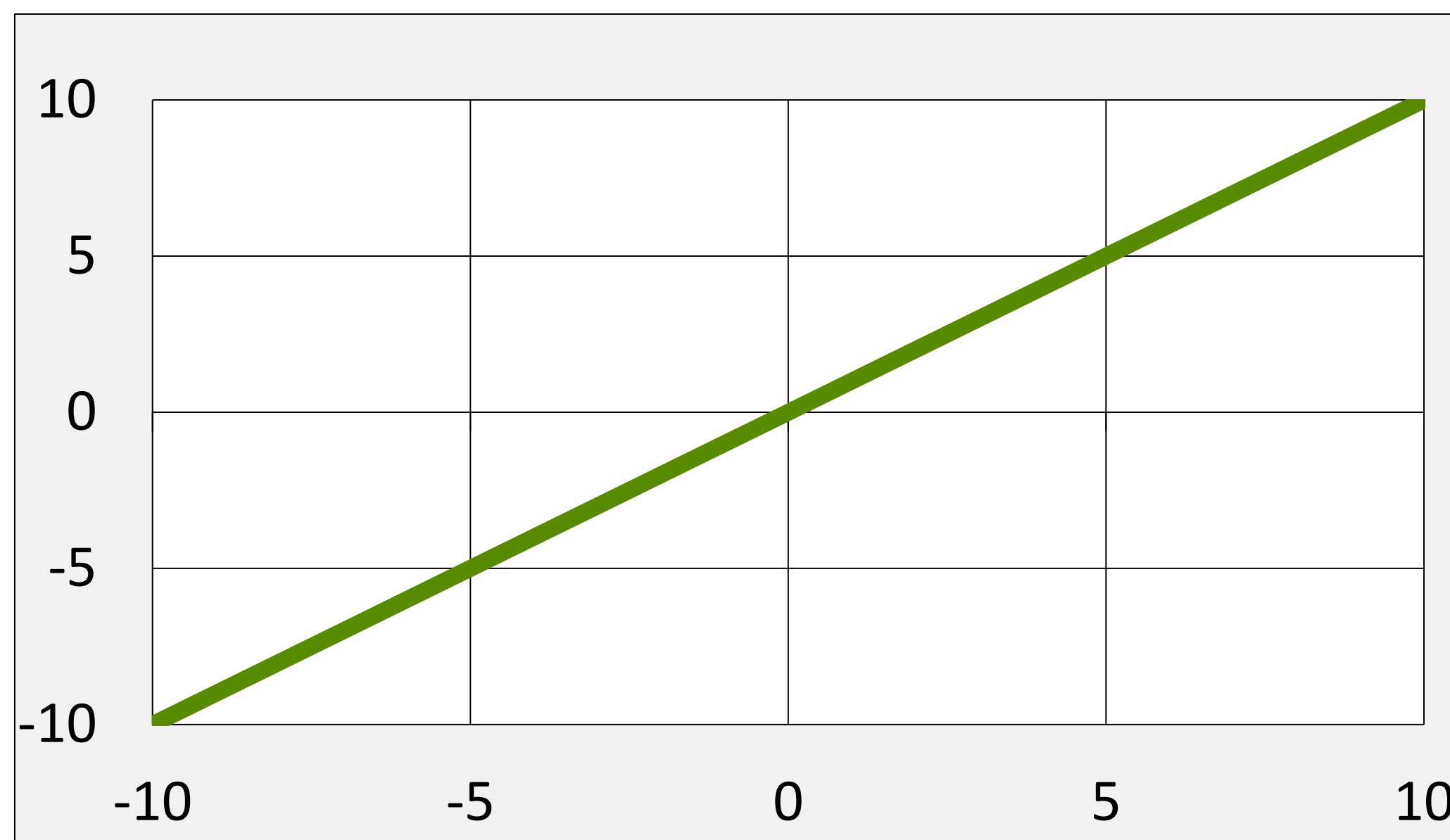
Activation Functions

Activation Functions

Linear

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

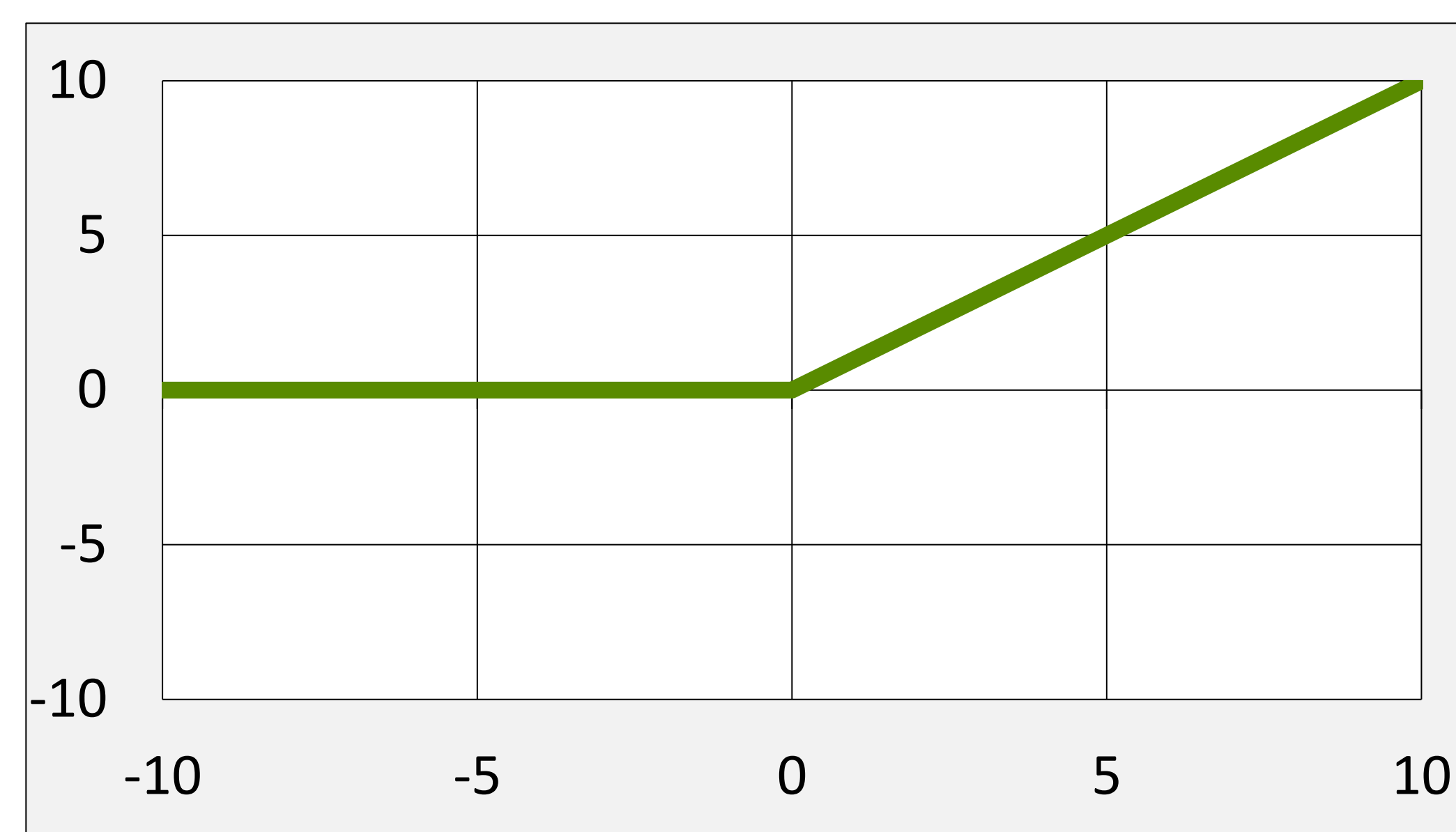
```
1 # Multiply each input
2 # with a weight (w) and
3 # add intercept (b)
4 y_hat = wx+b
```



ReLU

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

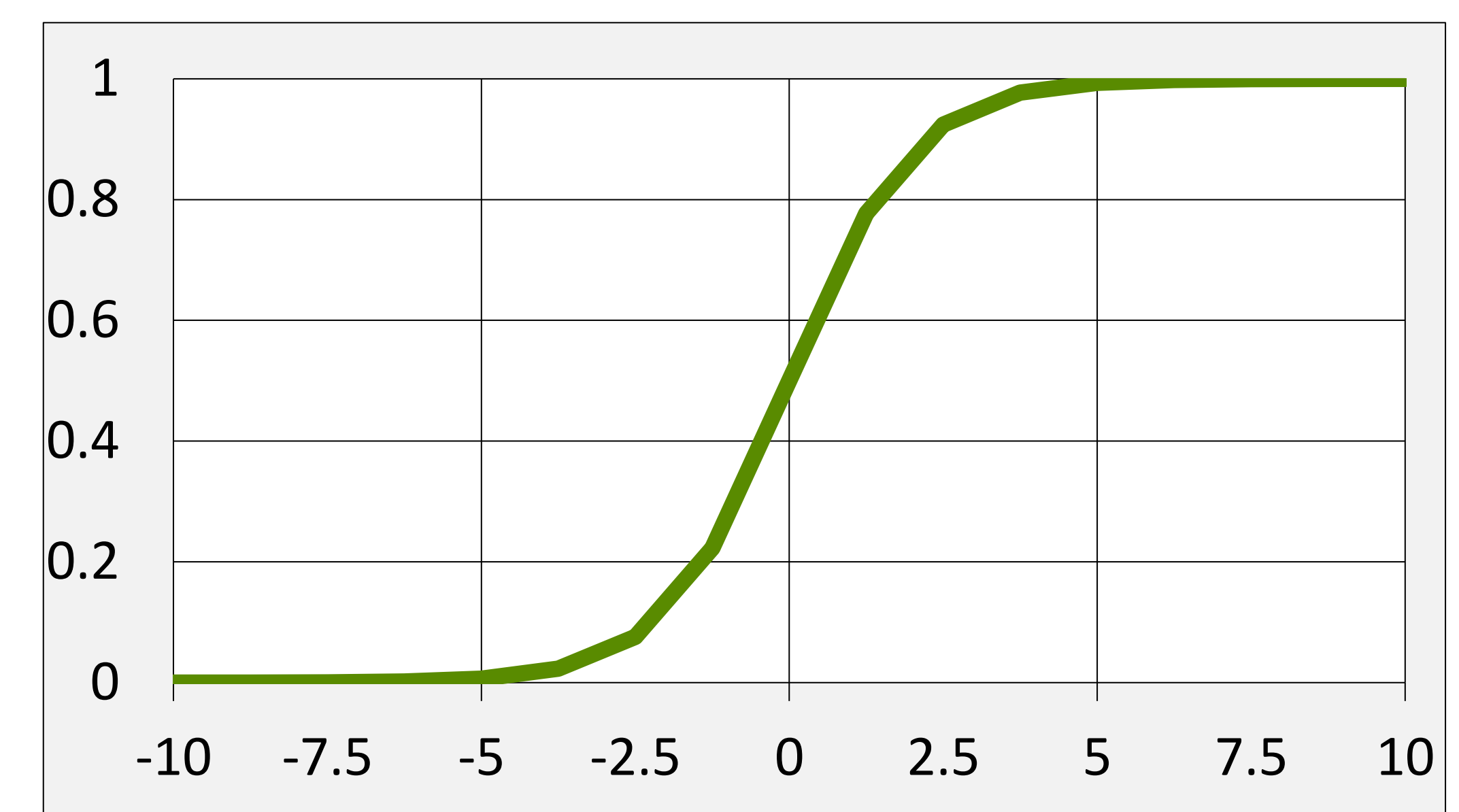
```
1 # Only return result
2 # if total is positive
3 linear = wx+b
4 y_hat = linear * (linear > 0)
```



Sigmoid

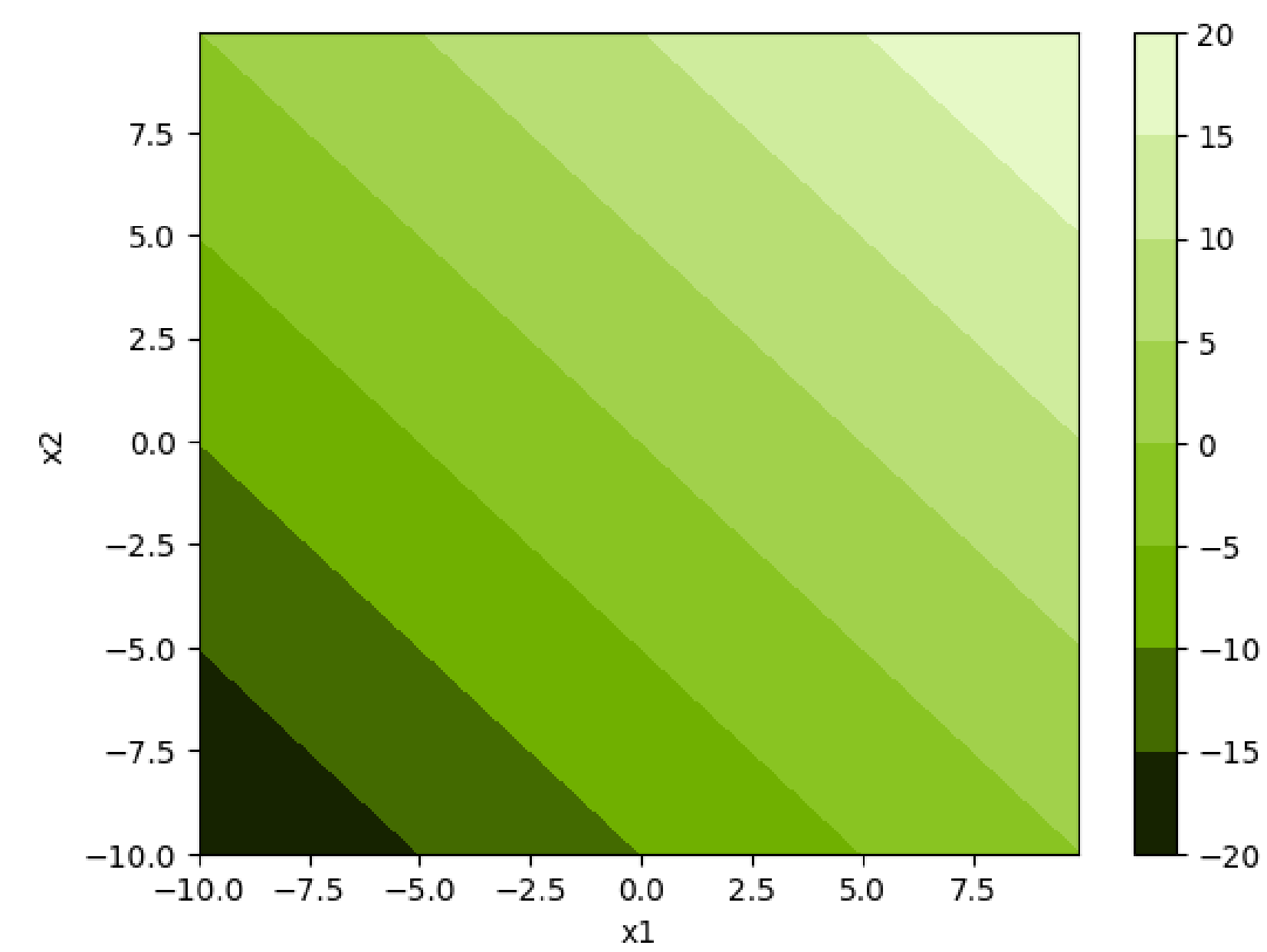
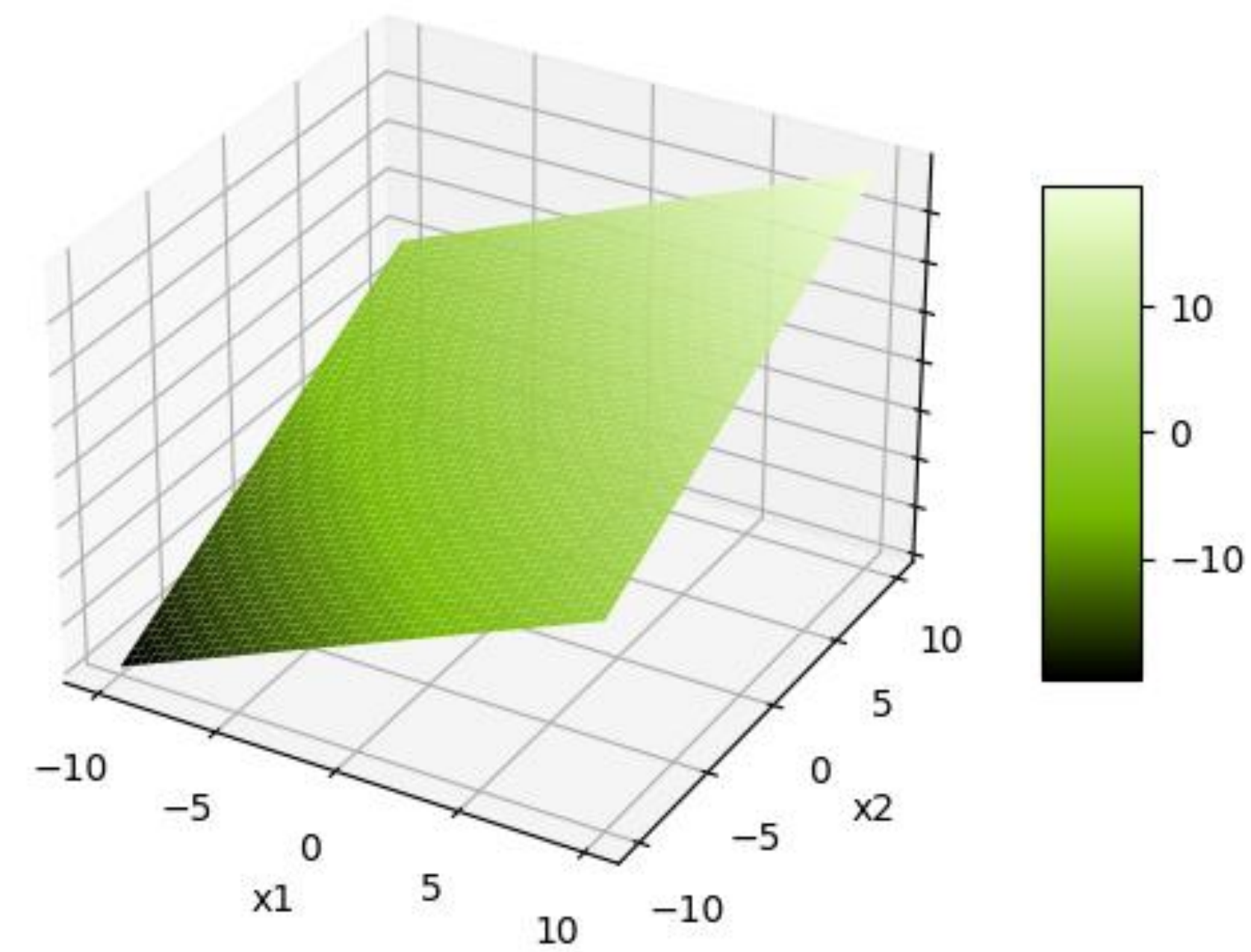
$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
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)
```

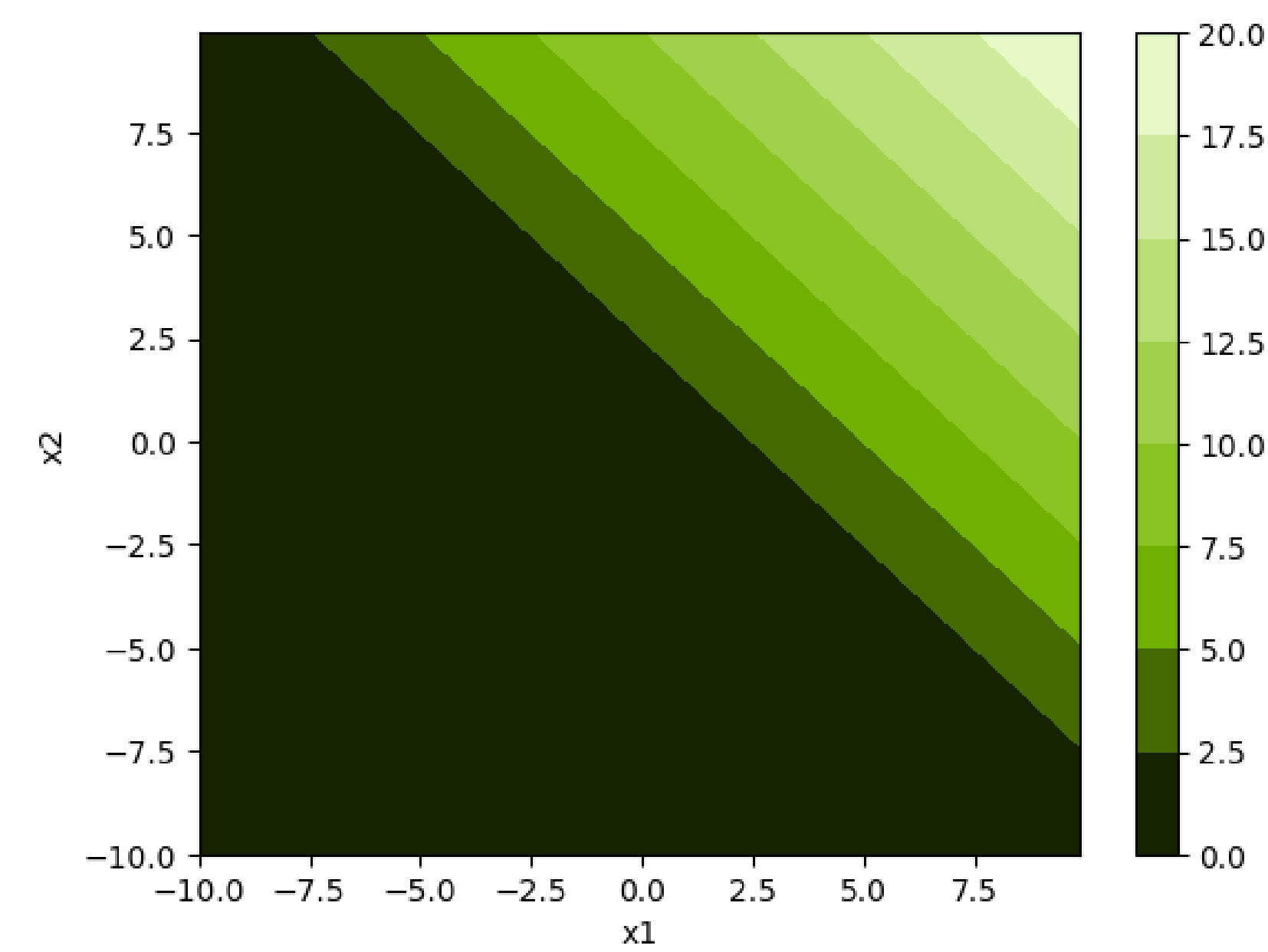
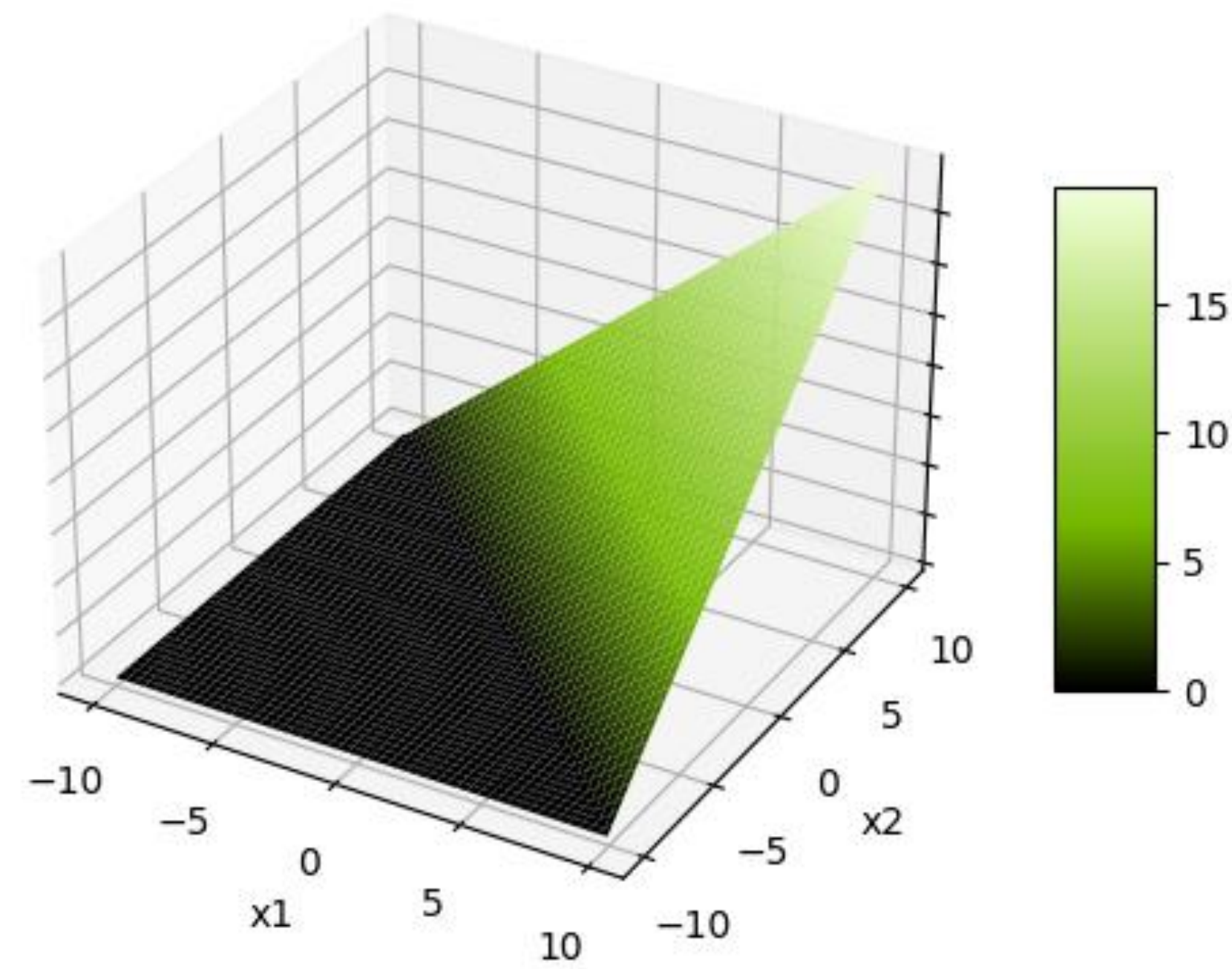


Activation Functions

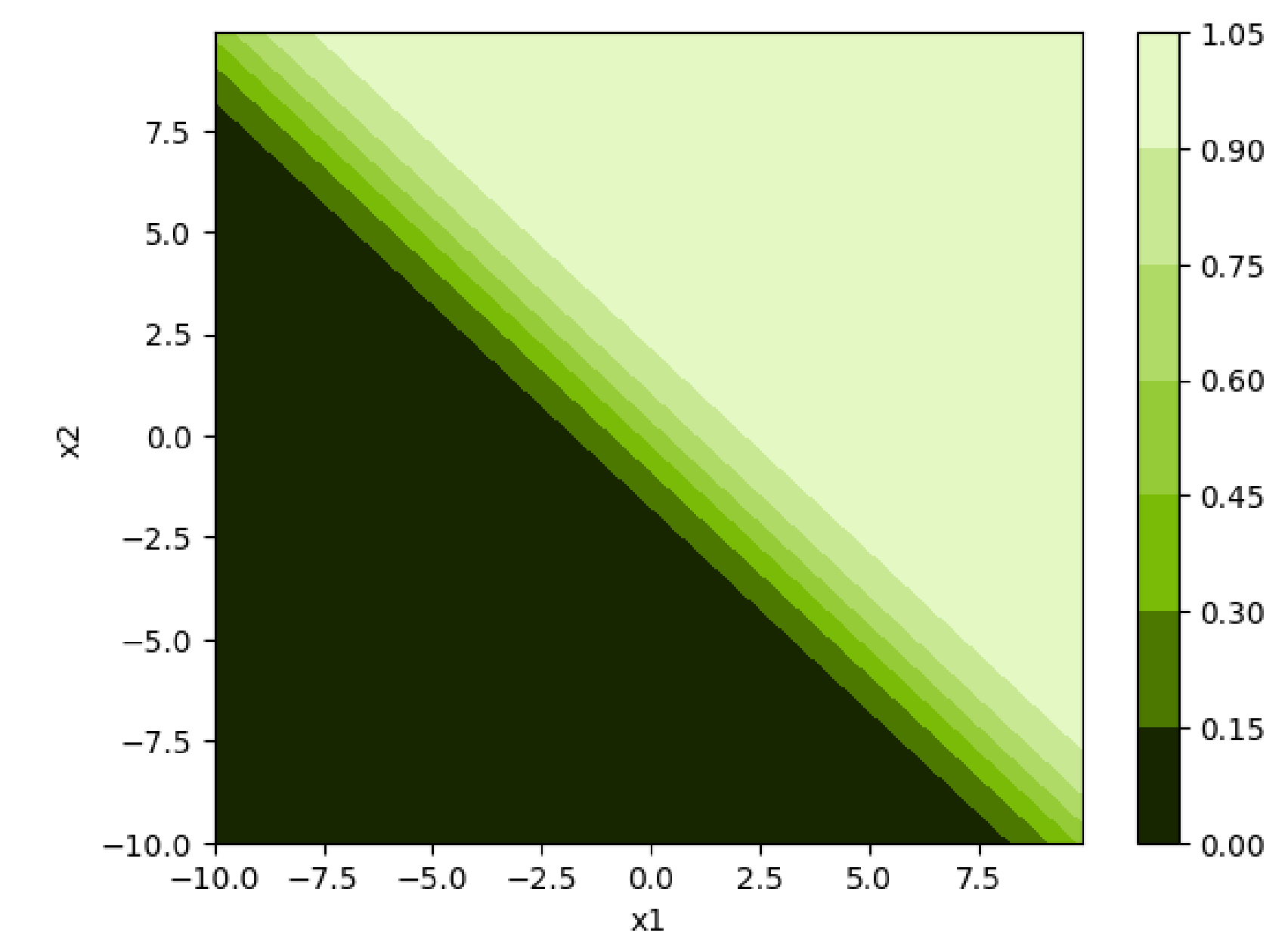
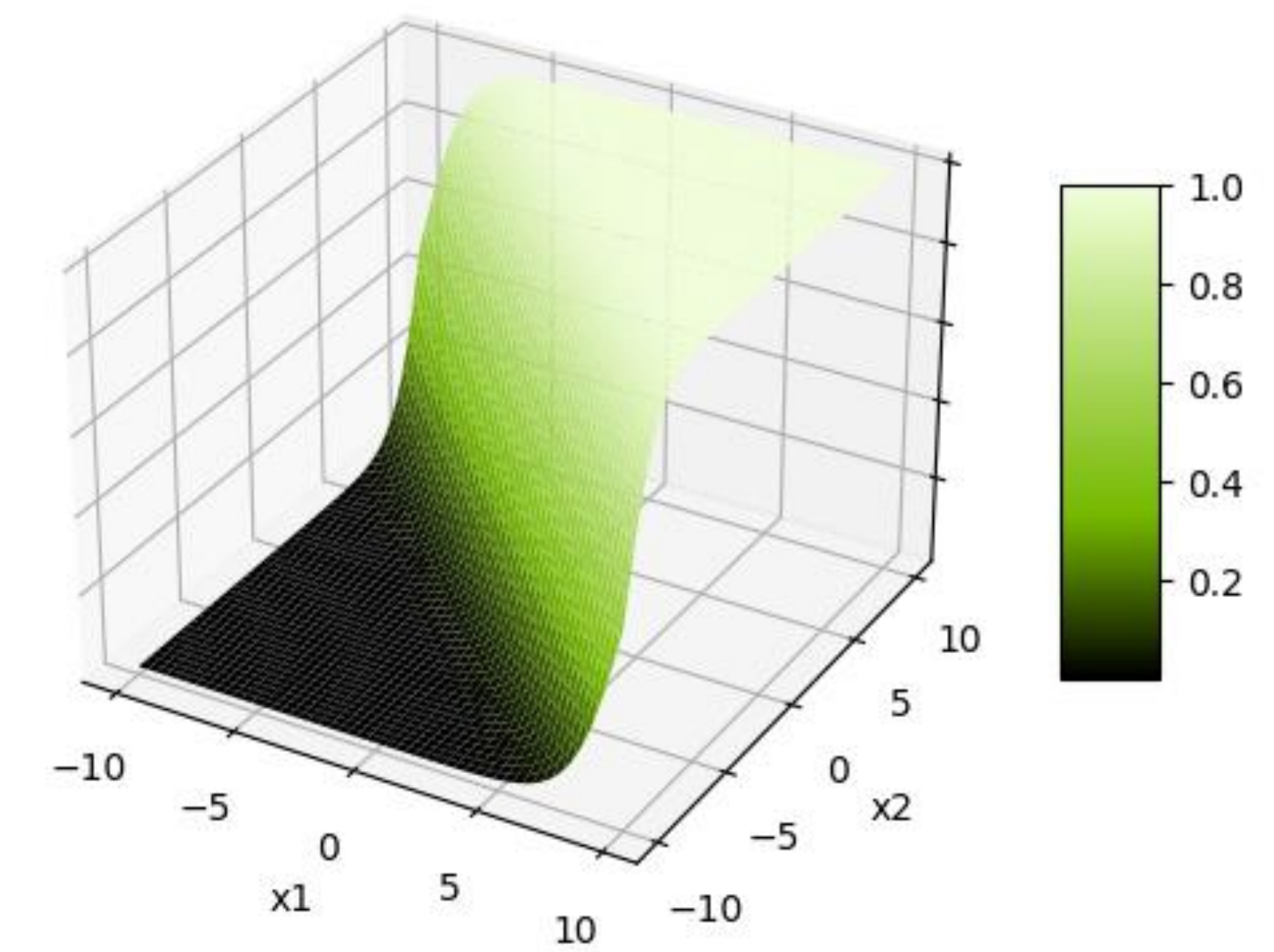
Linear



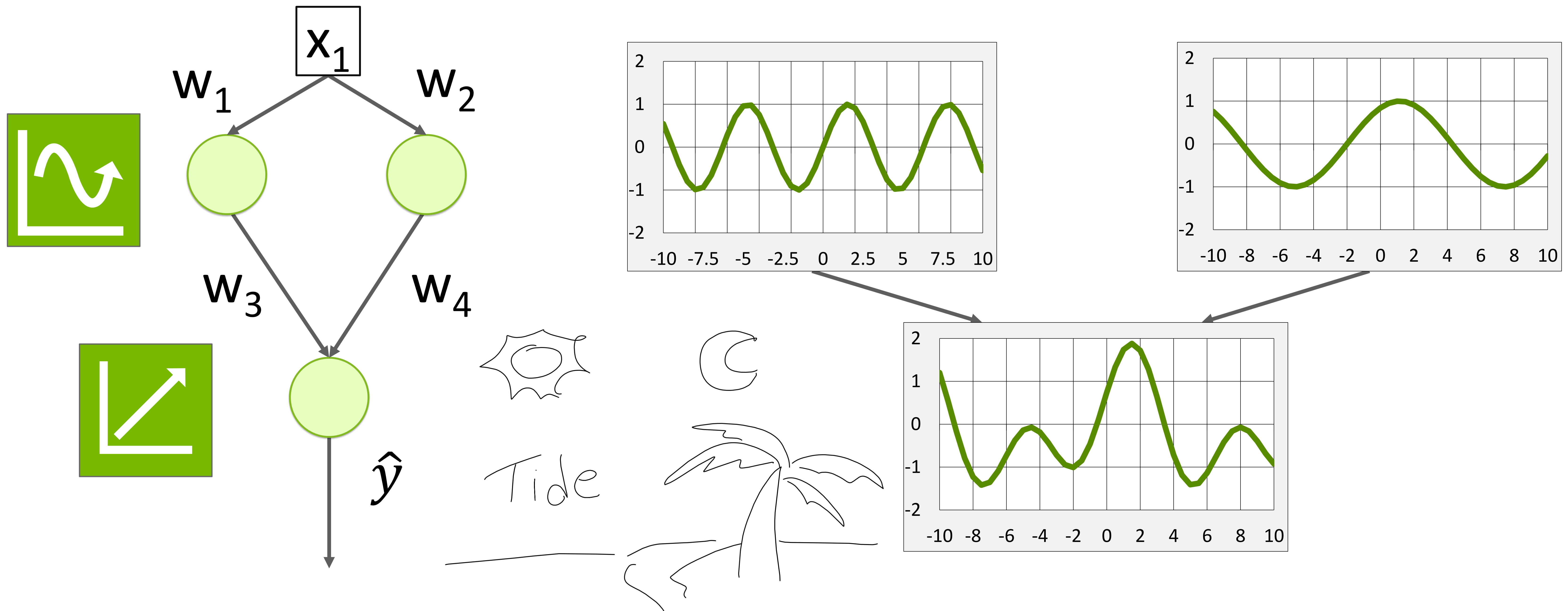
ReLU



Sigmoid



Activation Functions



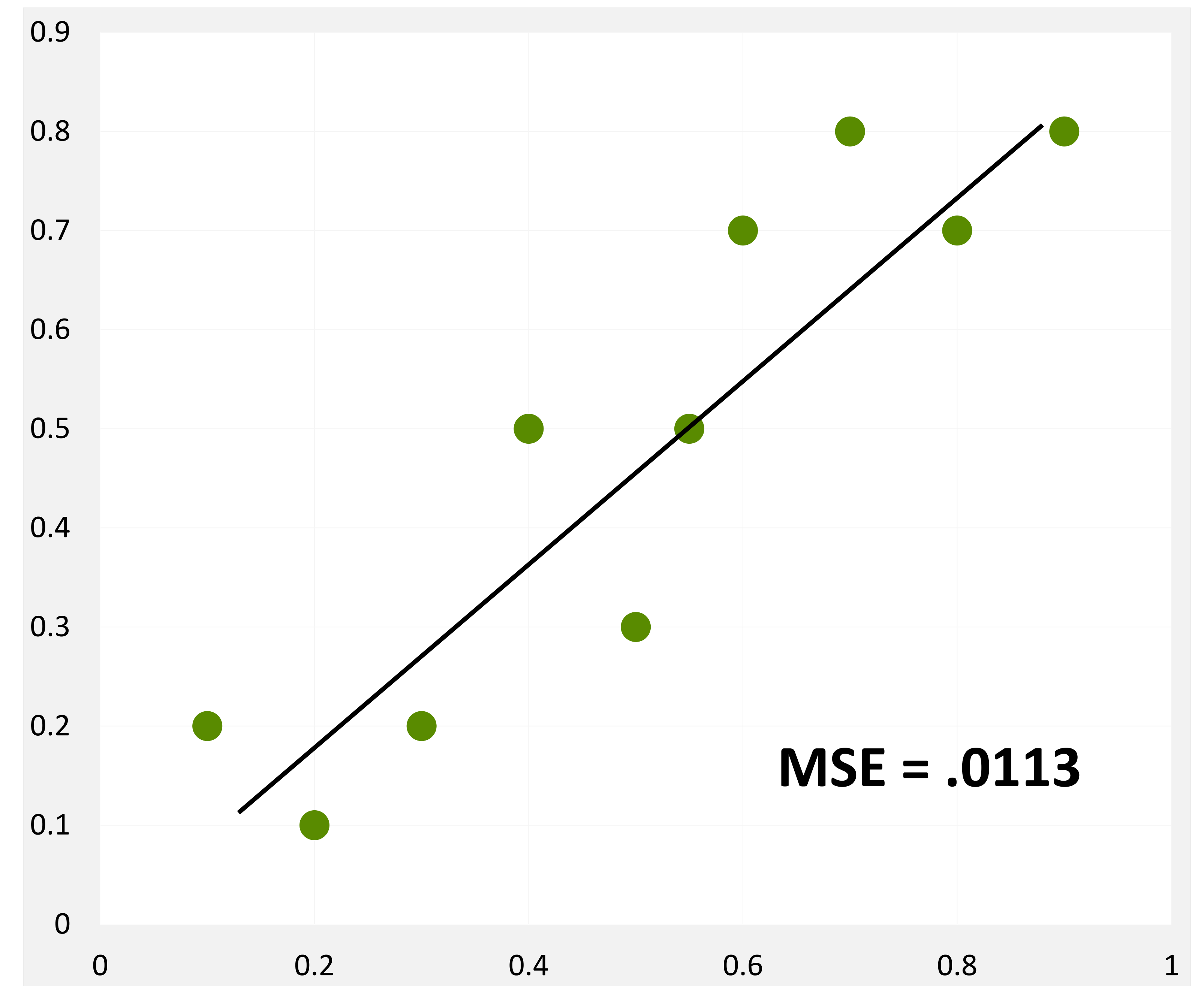
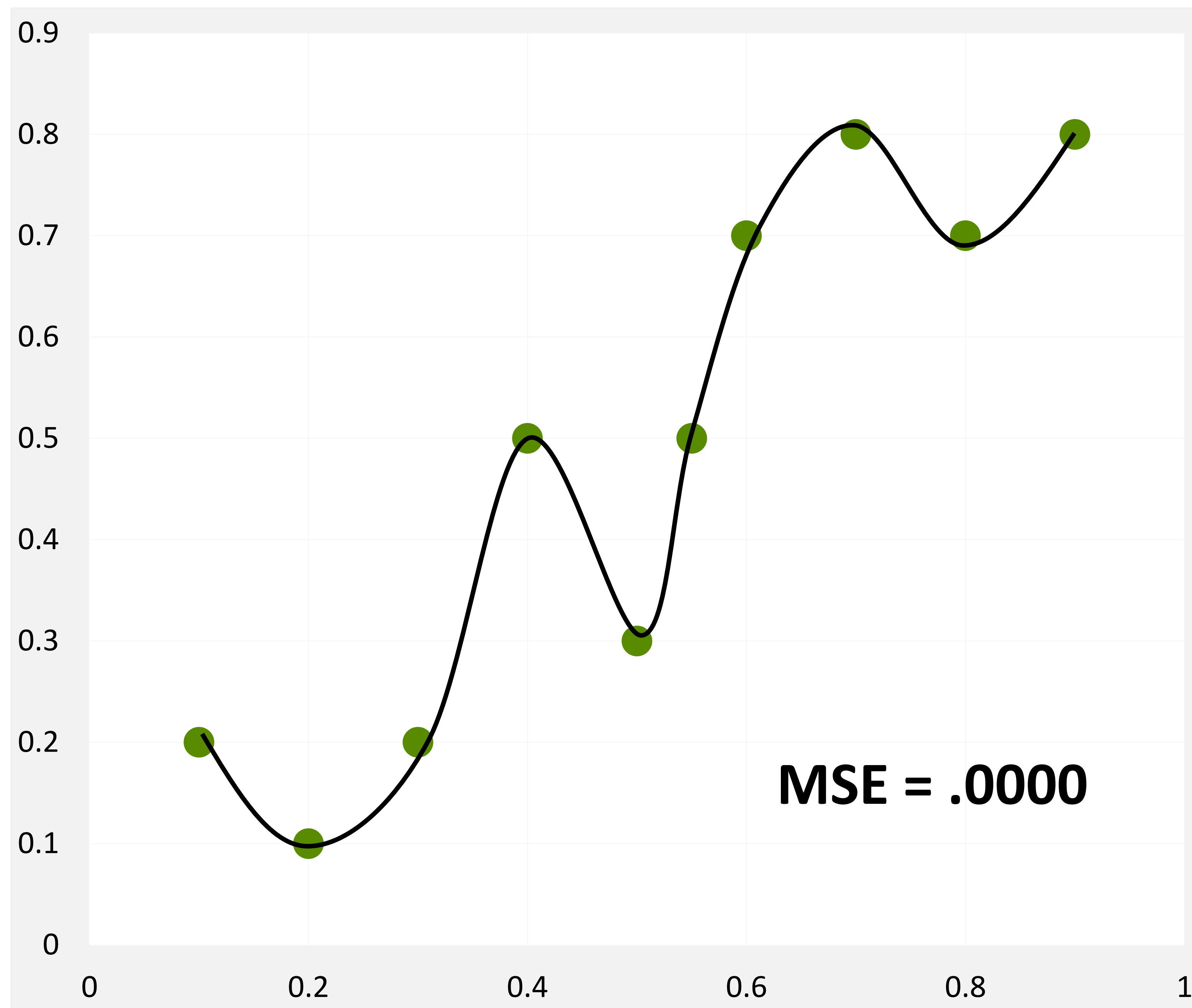
Overfitting

Why not have a super large neural network?



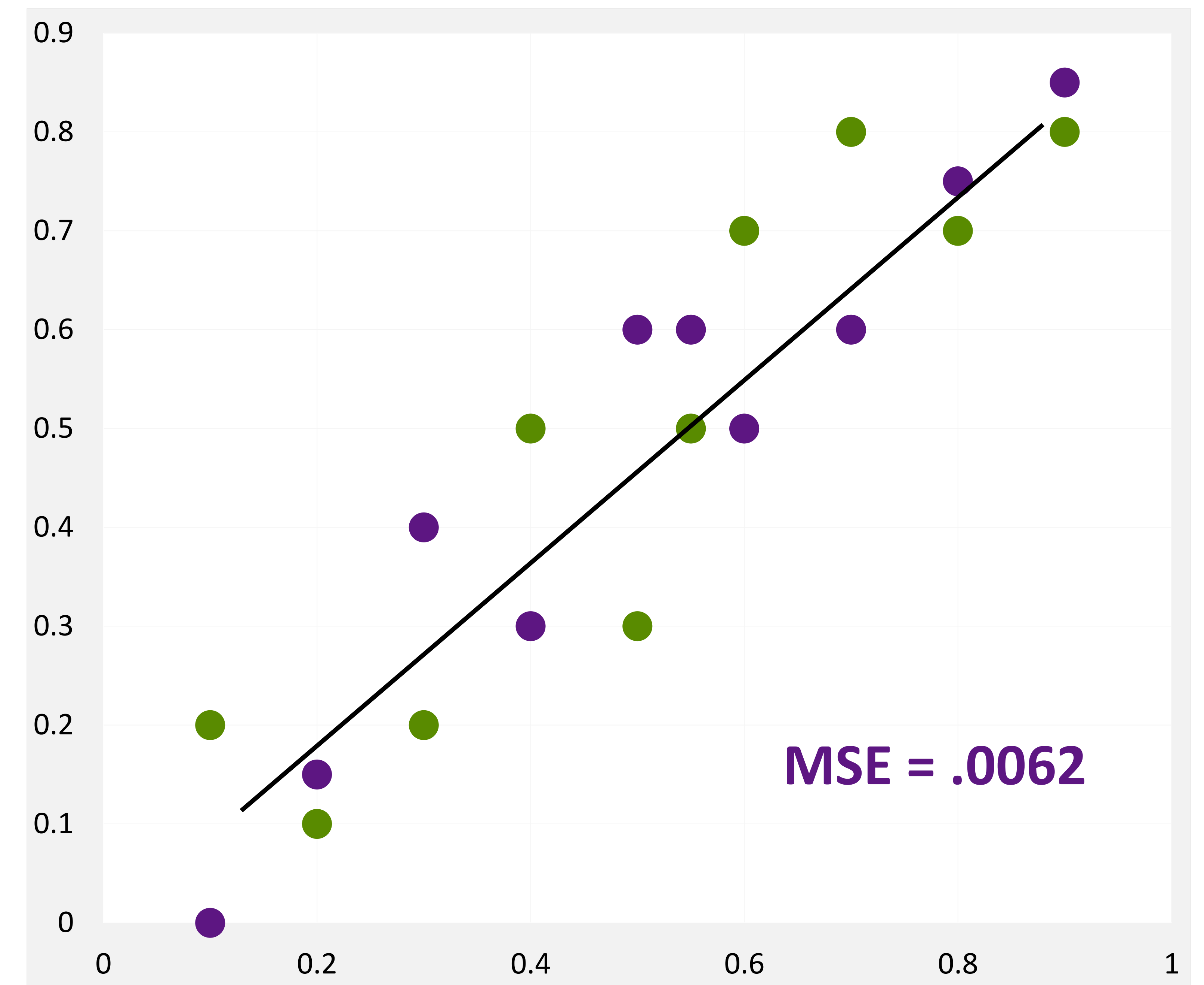
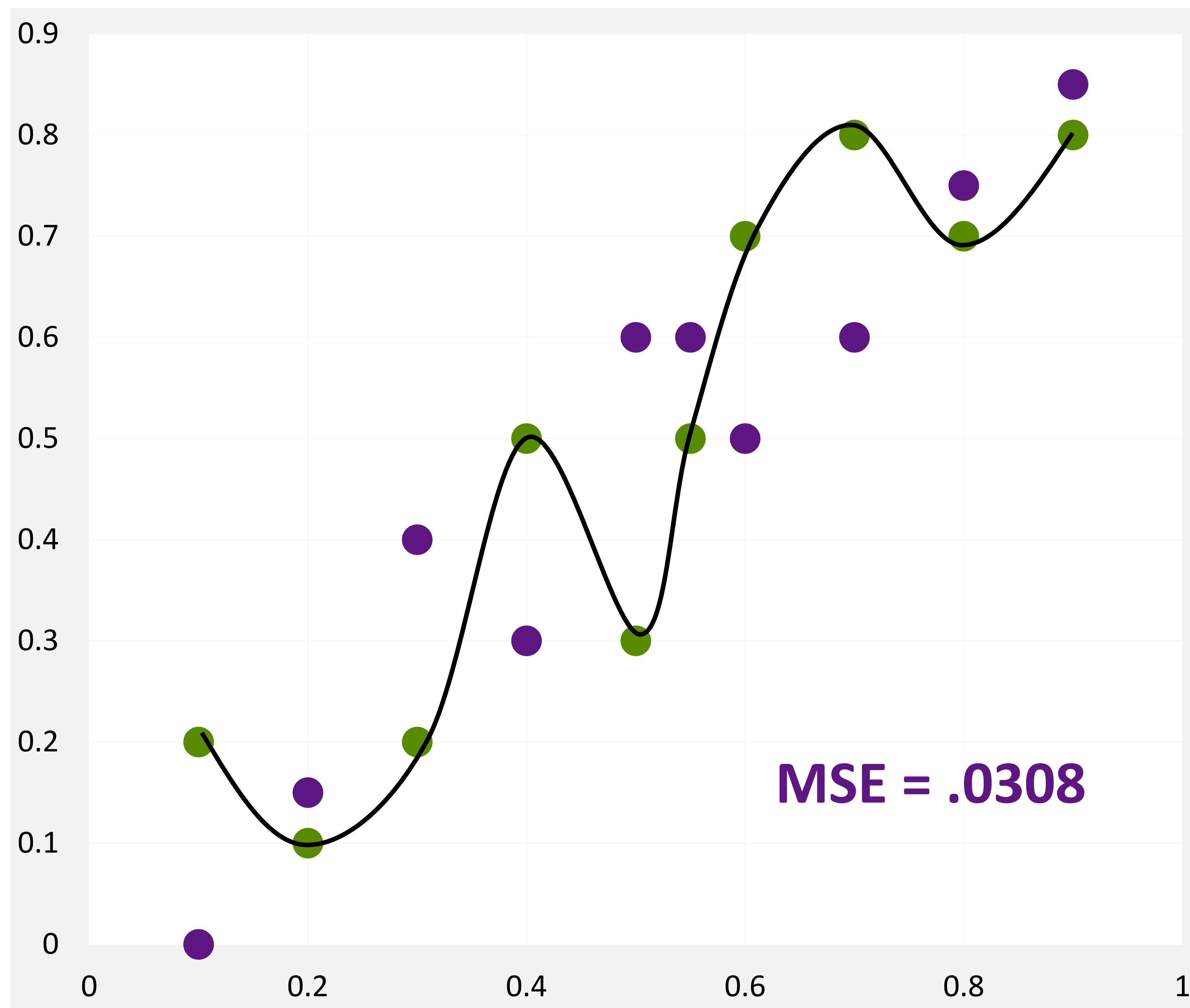
Overfitting

Which Trendline is Better?



Overfitting

Which Trendline is Better?



Training vs Validation Data

Avoid memorization

Training data

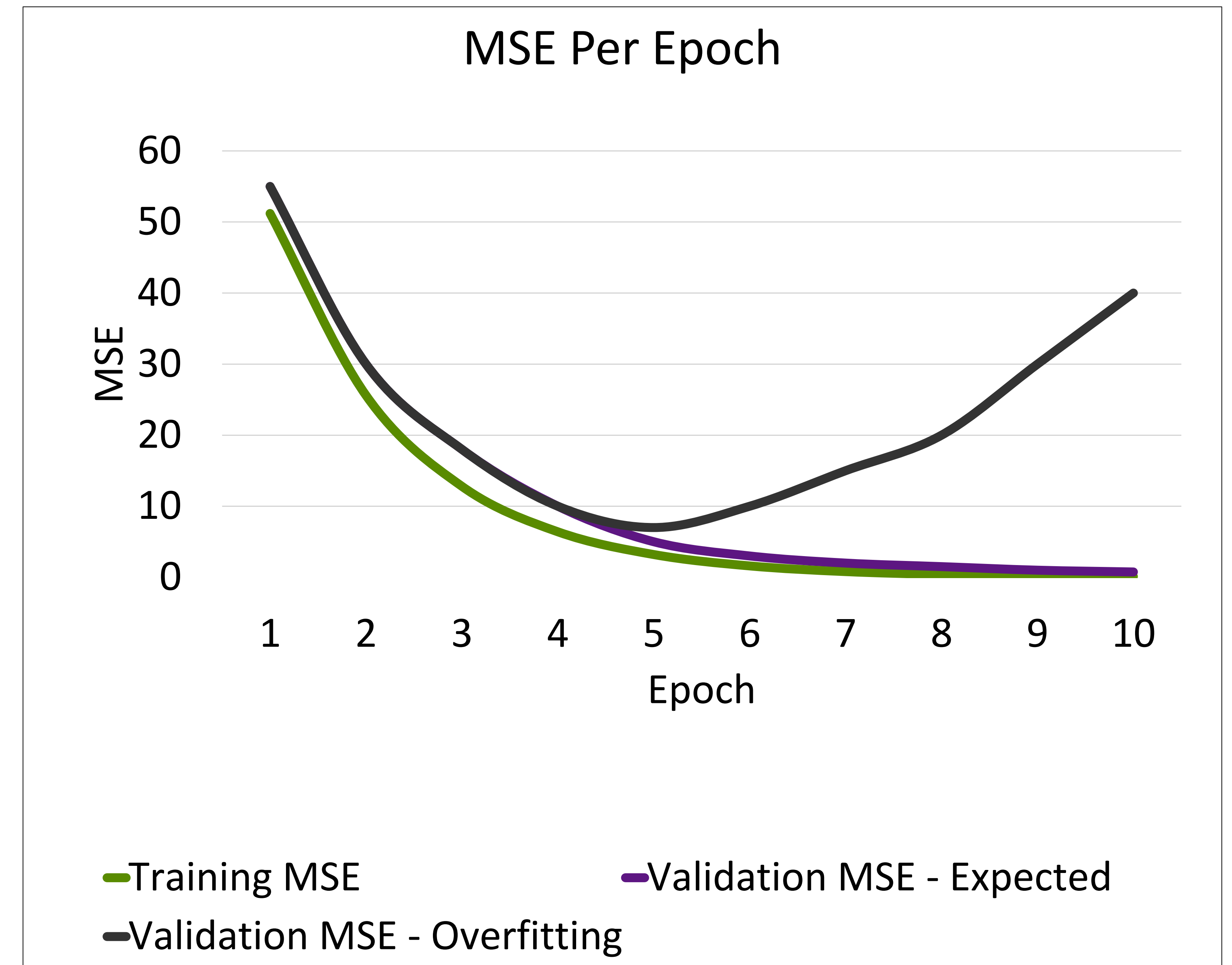
- Core dataset for the model to learn on

Validation data

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

Overfitting

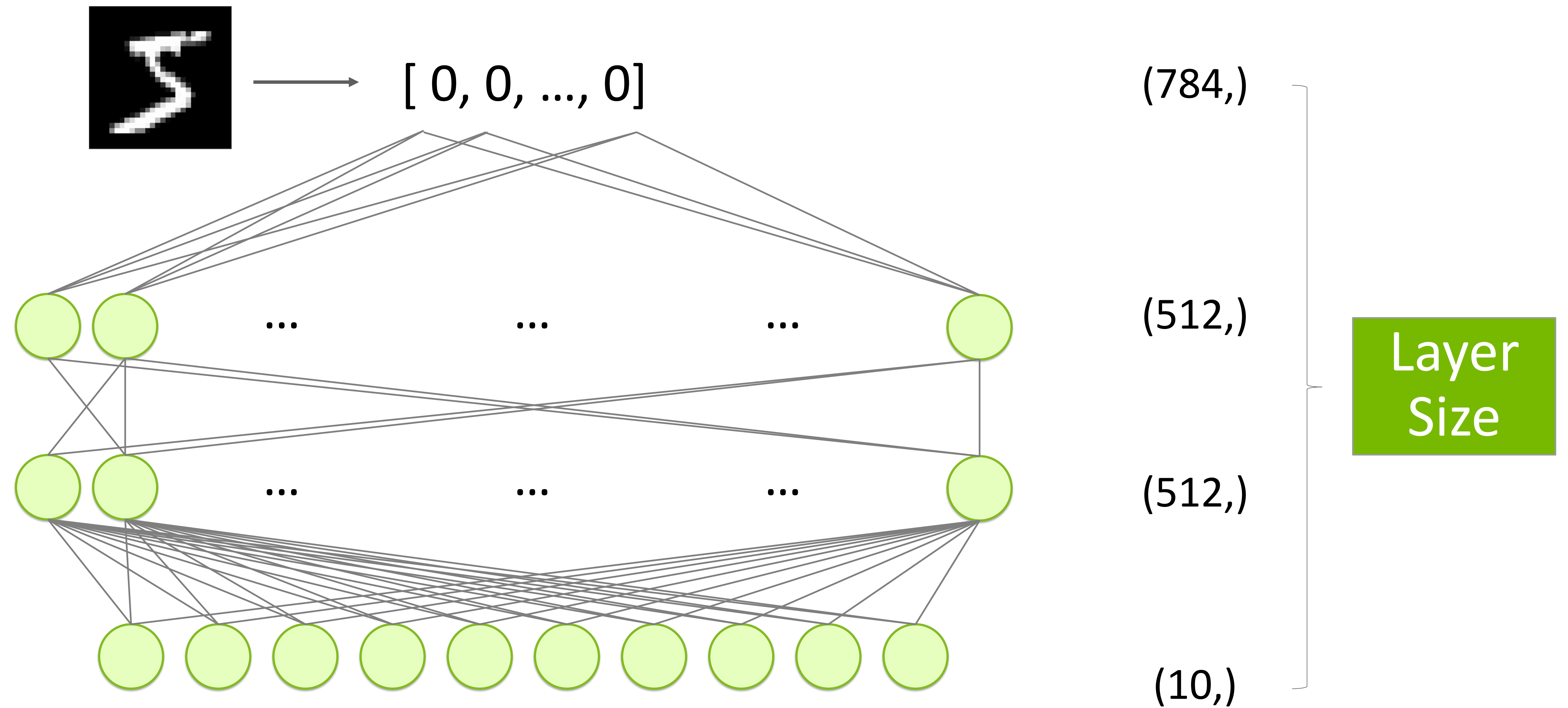
- When model performs well on the training data, but not the validation data (evidence of memorization)
- Ideally the accuracy and loss should be similar between both datasets





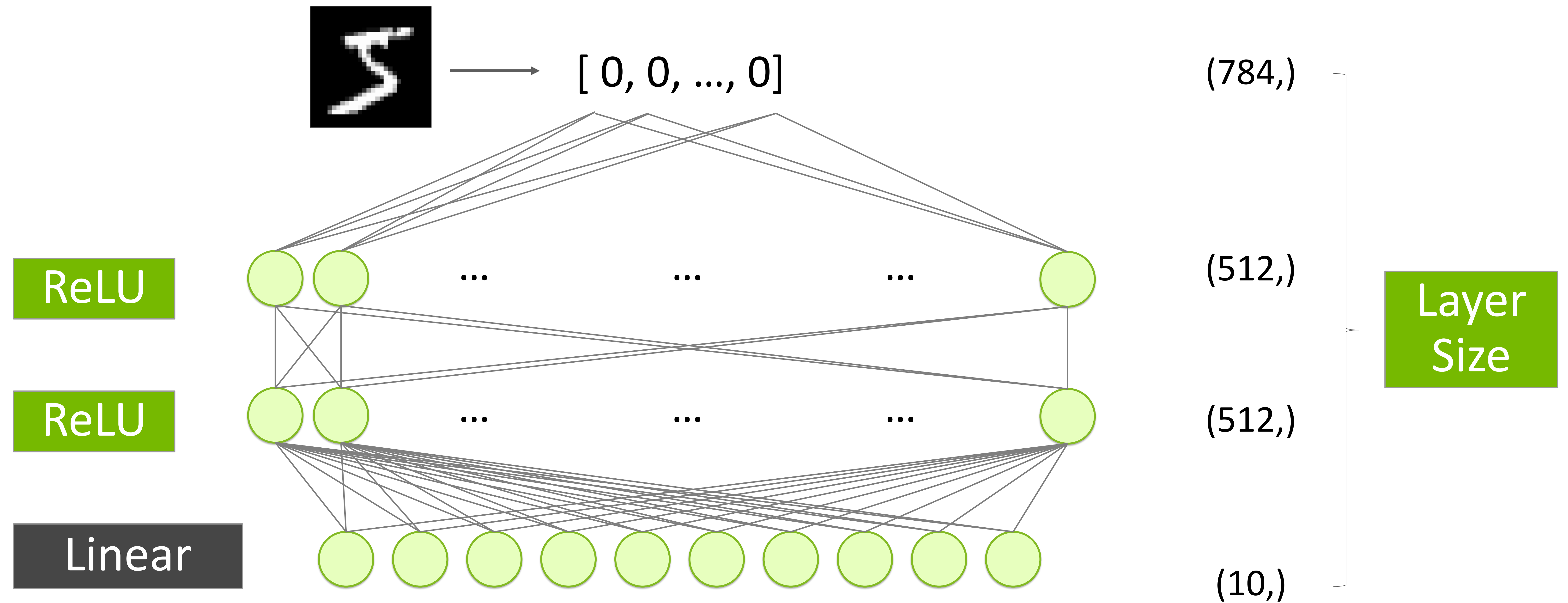
From Regression to Classification

An MNIST Model



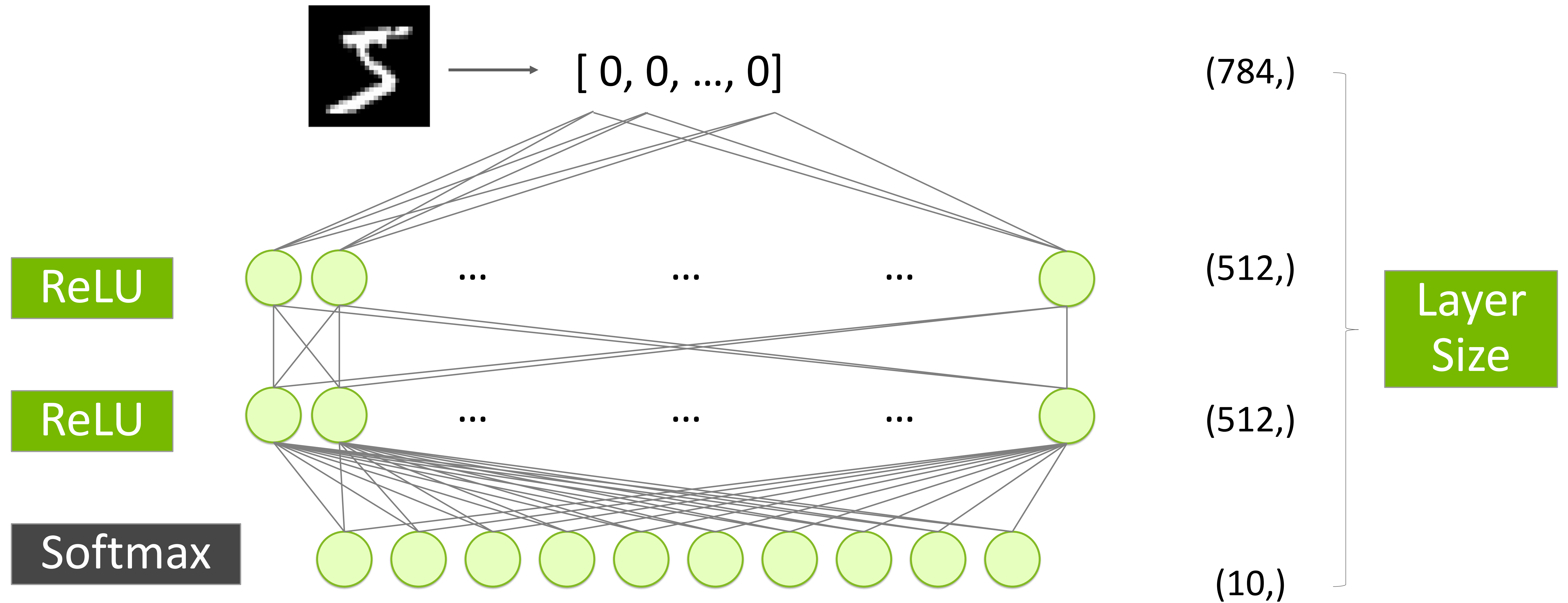
An MNIST Model

During Prediction

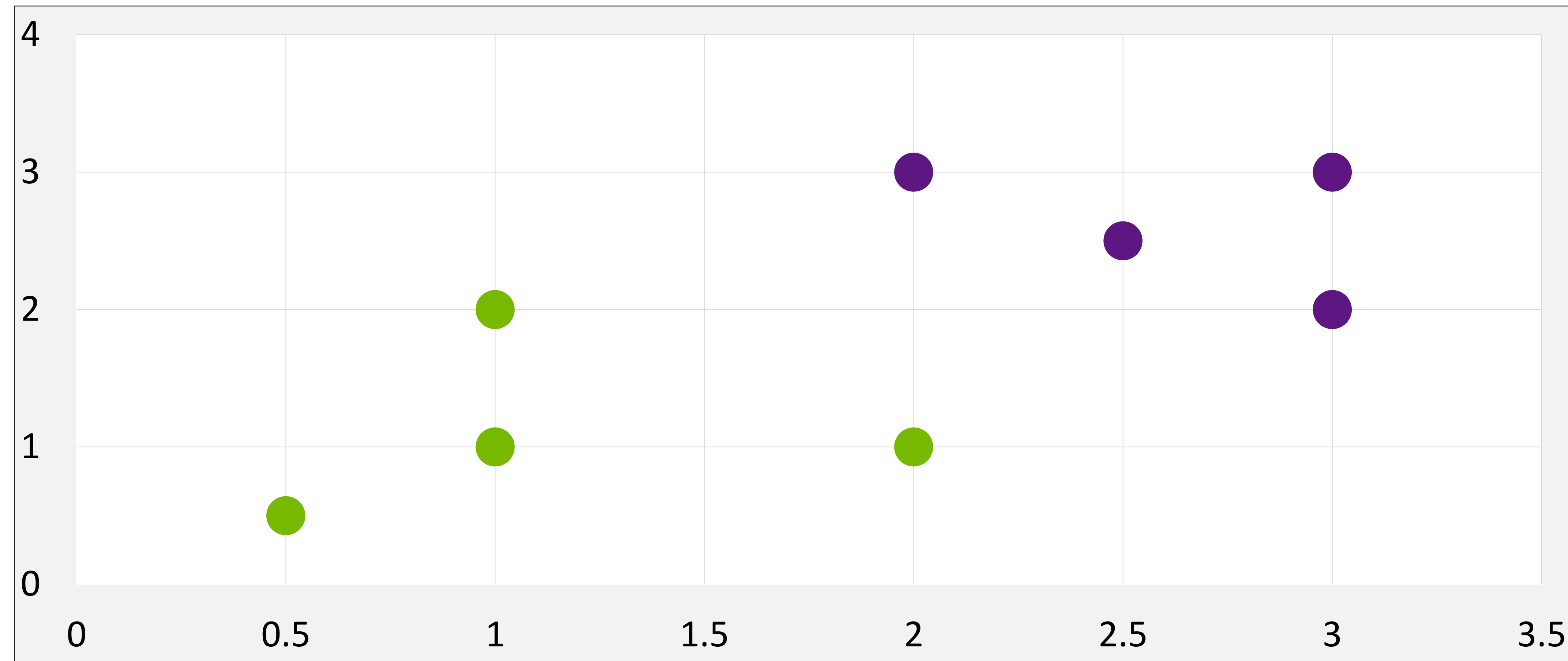


An MNIST Model

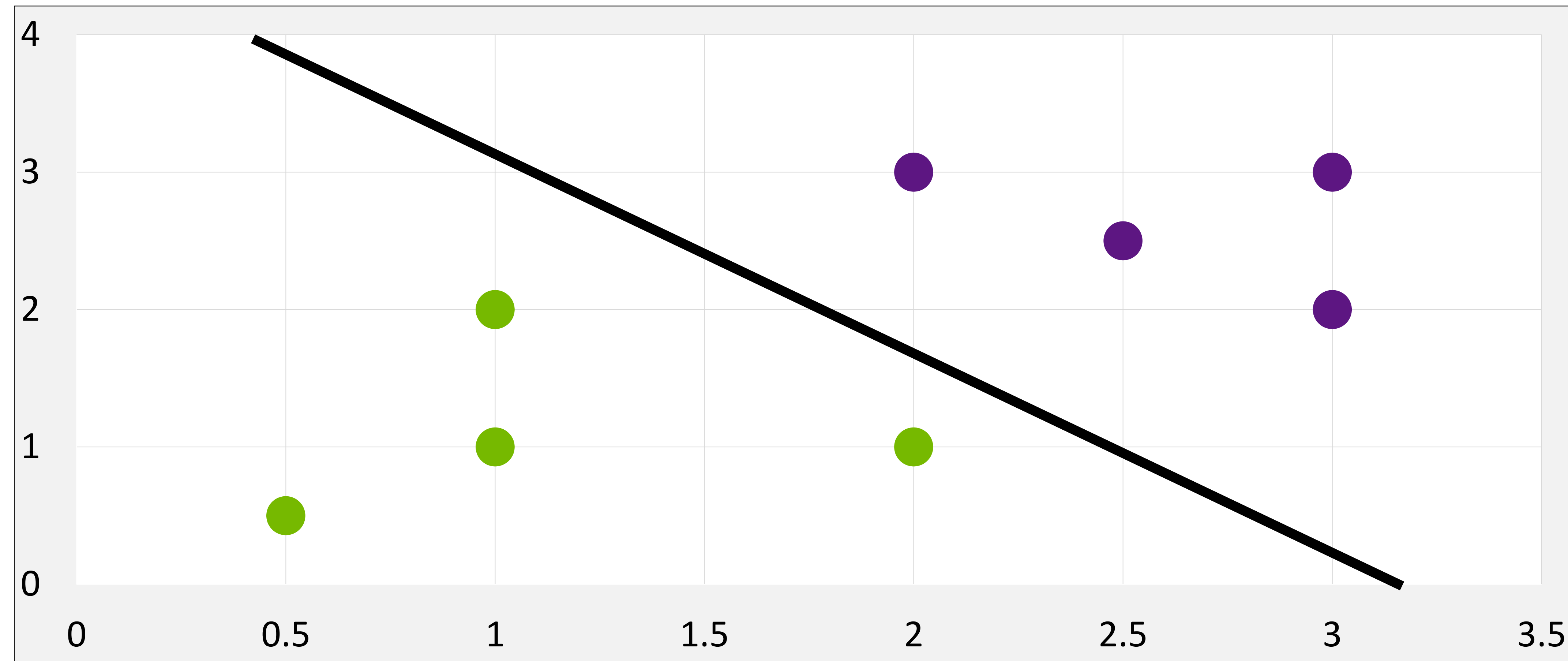
During Training



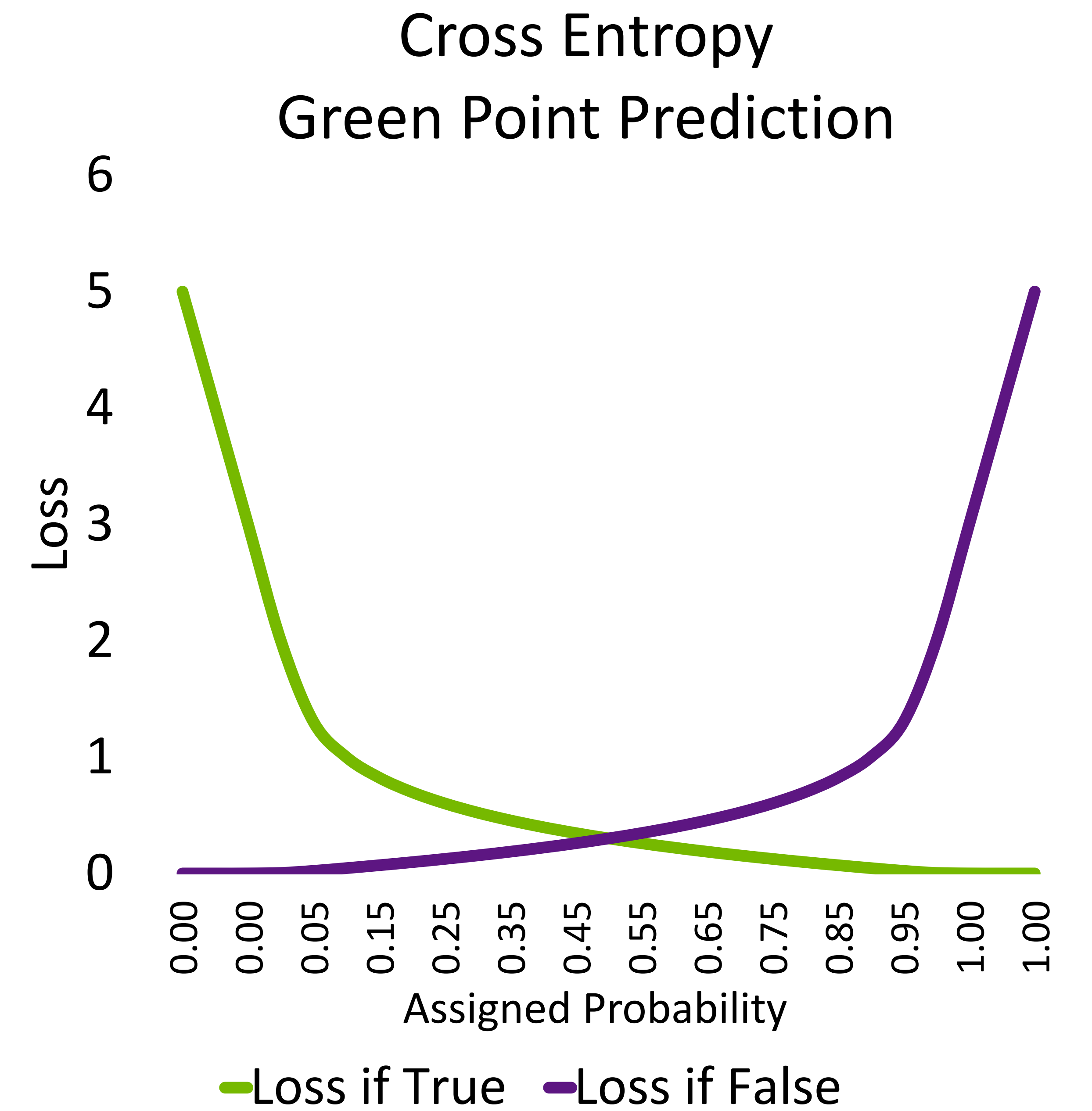
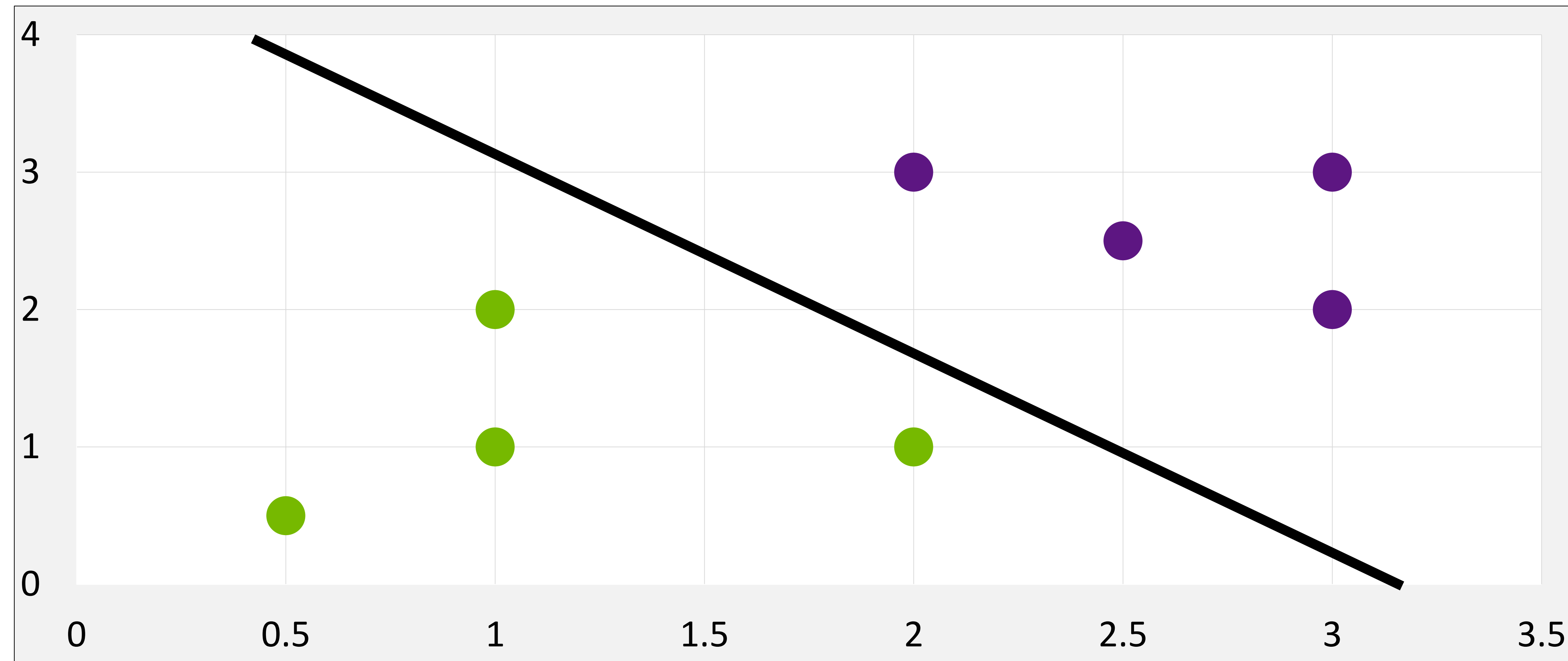
RMSE For Probabilities?



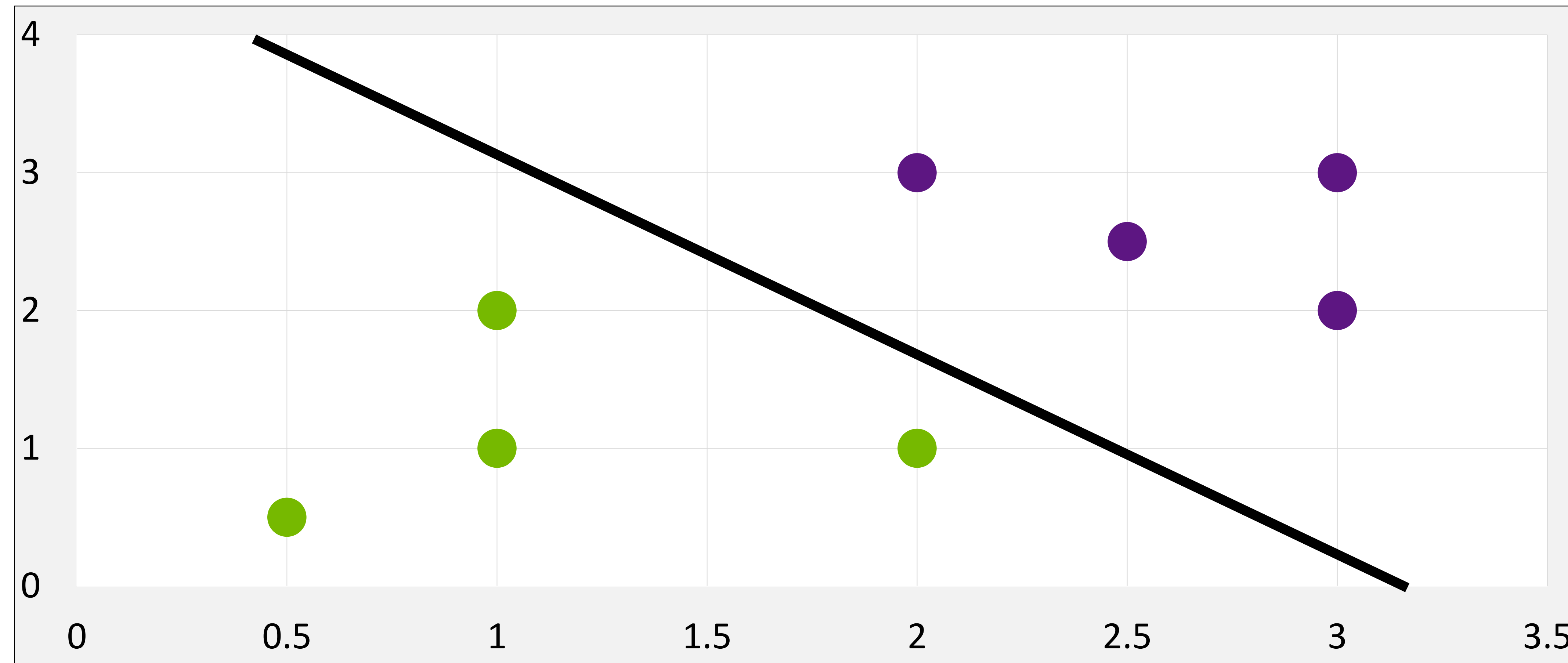
RMSE For Probabilities?



Cross Entropy



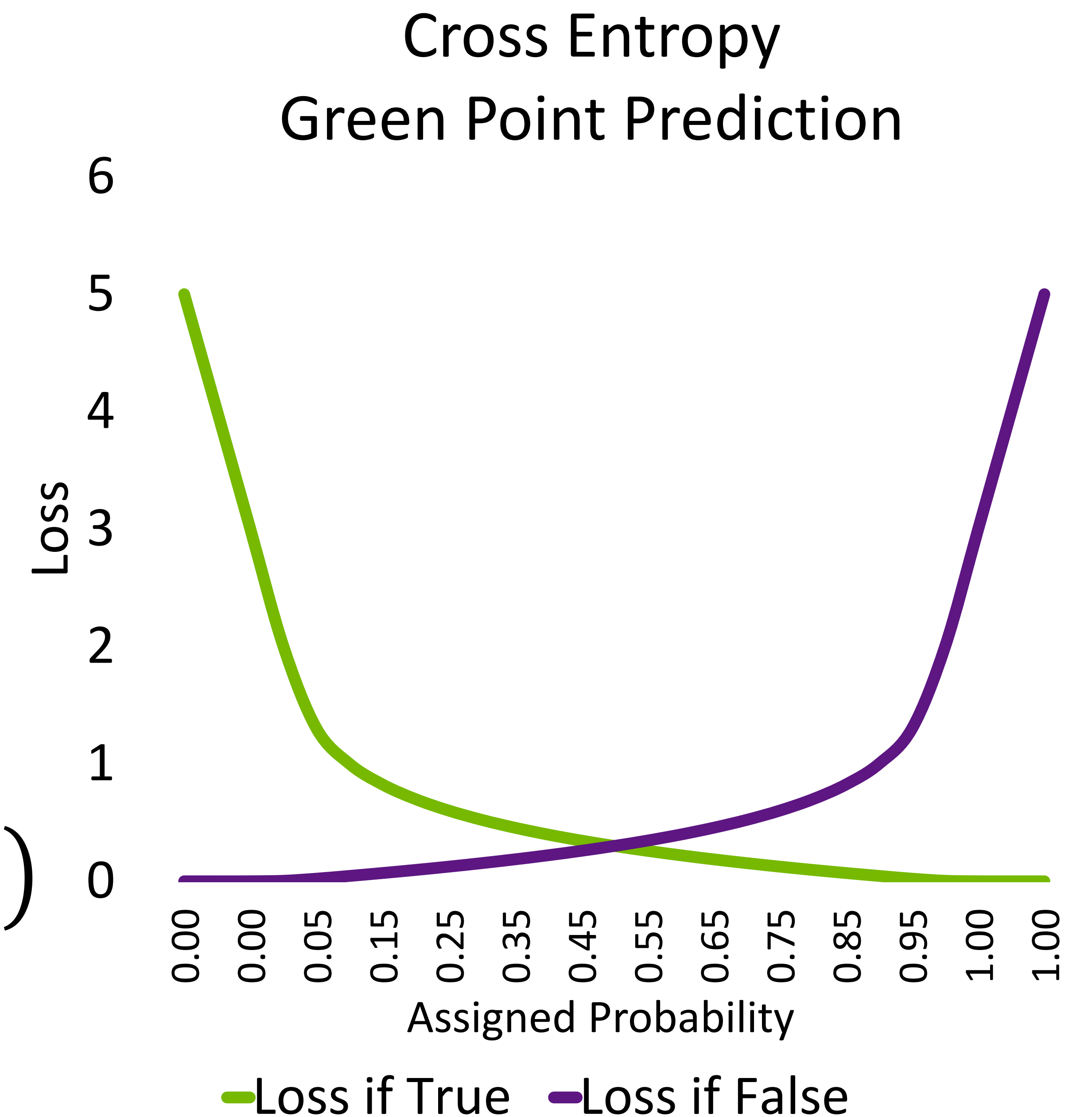
Cross Entropy



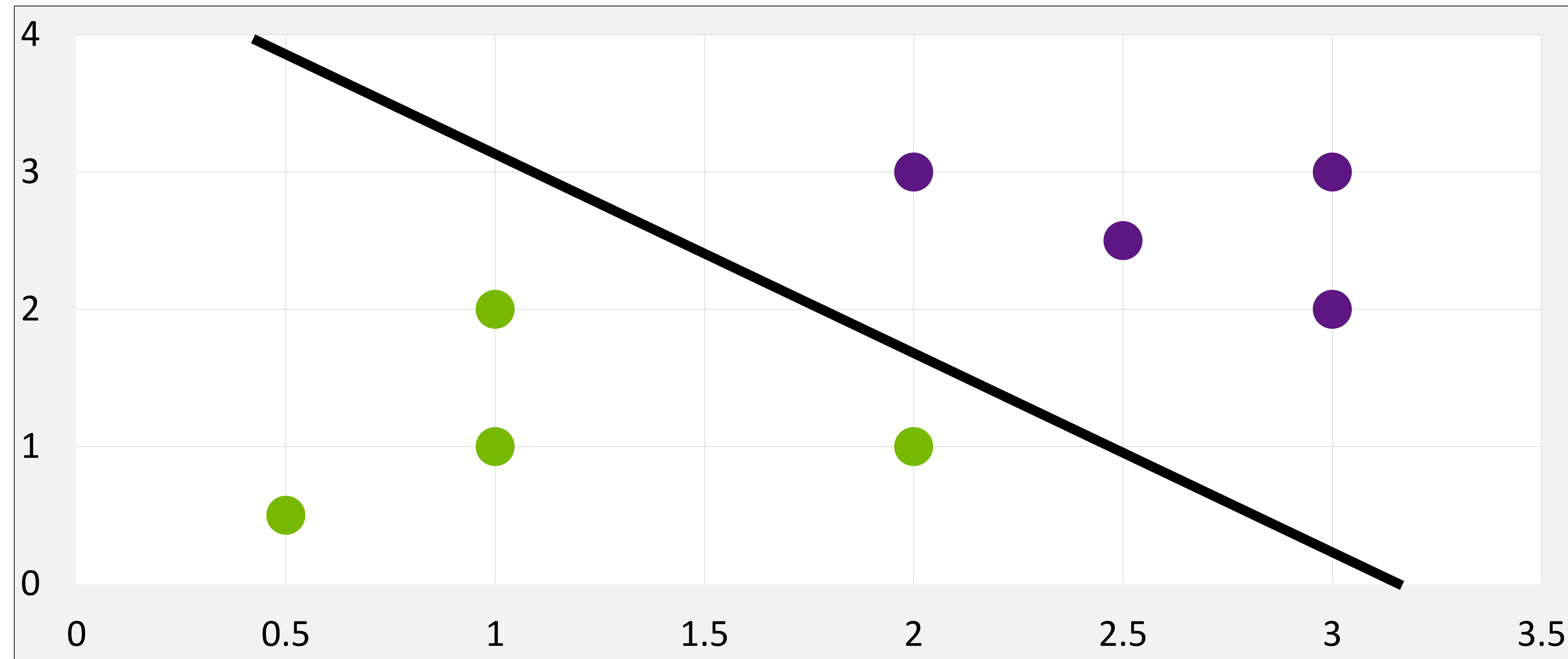
$$Loss = -((t(x) \cdot \log(p(x))) + (1 - t(x)) \cdot \log(1 - p(x)))$$

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

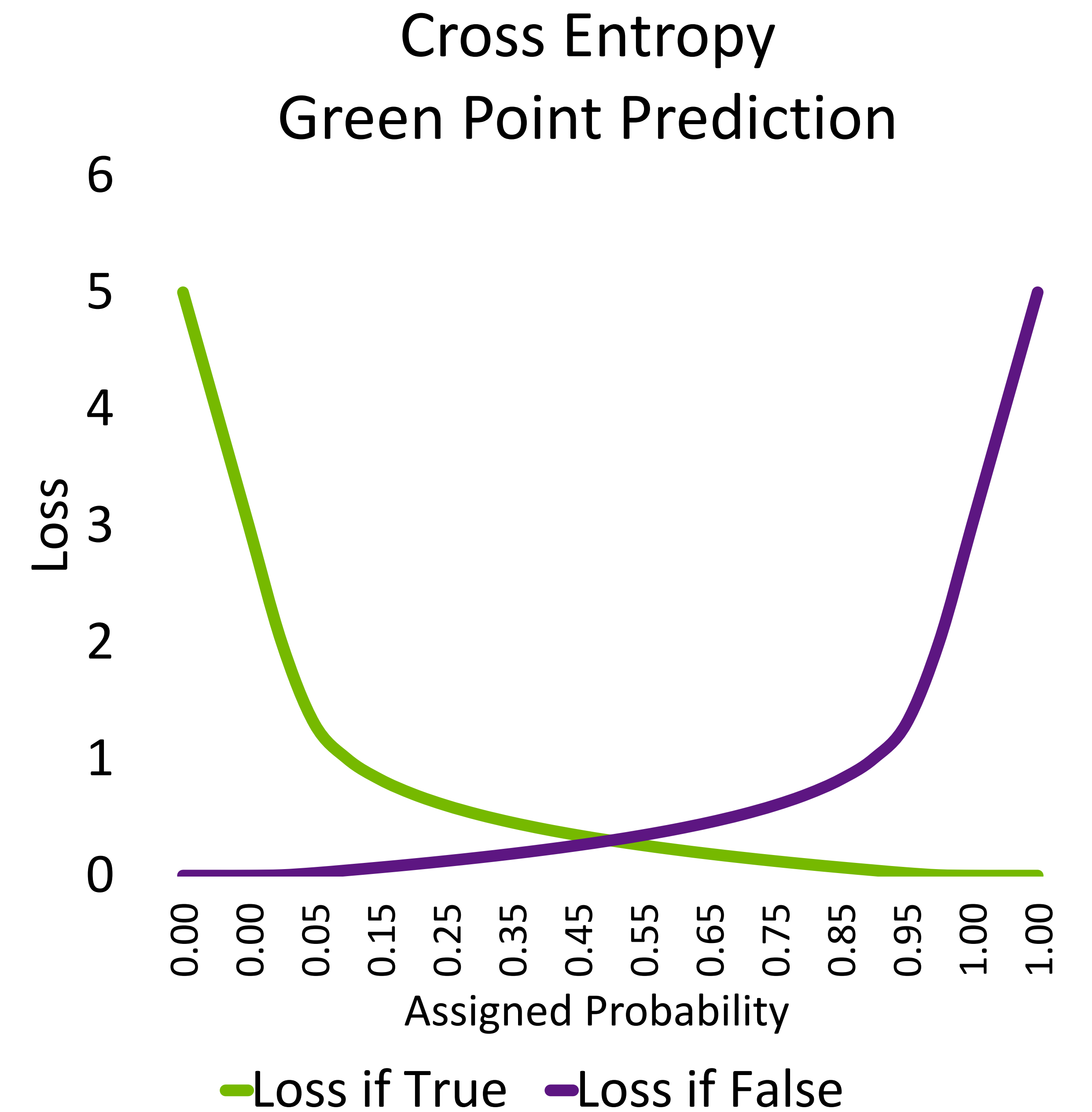
$p(x)$ = probability prediction of point x



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
```

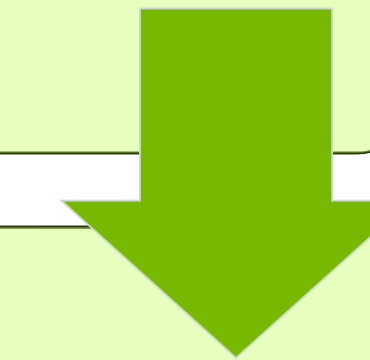




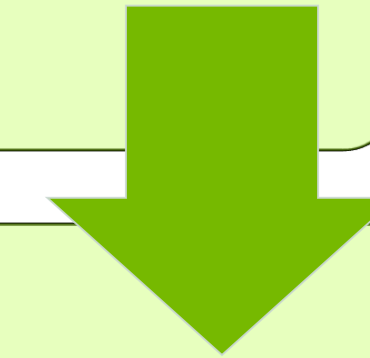
Overview of the Course

Structure of the Course

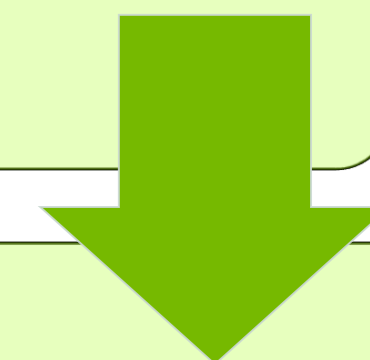
“Hello World” of Deep Learning



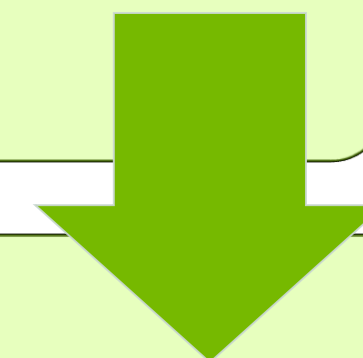
Train a more complicated model



New architectures and techniques to improve performance



Pre-trained models



Transfer learning

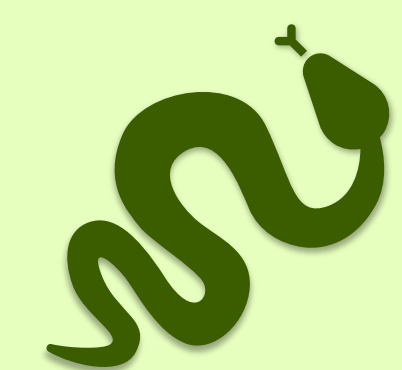
Platform of the Course



GPU powered cloud server



JupyterLab platform



Jupyter notebooks for interactive coding

Software of This Course

- Major deep learning platforms:
 - TensorFlow + Keras (Google)
 - PyTorch (Meta)
 - MXNet (Apache)
- We'll be using PyTorch
- Good idea to gain exposure to others moving forward



Hands on Exercises

- Get comfortable with the process of deep learning
- Exposure to different models and datatypes
- Get a jump-start to tackle your own projects
- Have fun!!



Hands on: work through Introduction Section

1. WIFI Info: eduroam
2. Browser Recommendation: Chrome
3. websockettest.courses.nvidia.com
4. <https://learn.nvidia.com/dli-event>
5. create an Account (if you have not done yet!)
6. event code:
7. work through the Introduction Section and 'Start' launching your first GPU task

First exercise: MNIST dataset classification (Hello Neural Networks)

Train a network to correctly classify handwritten digits

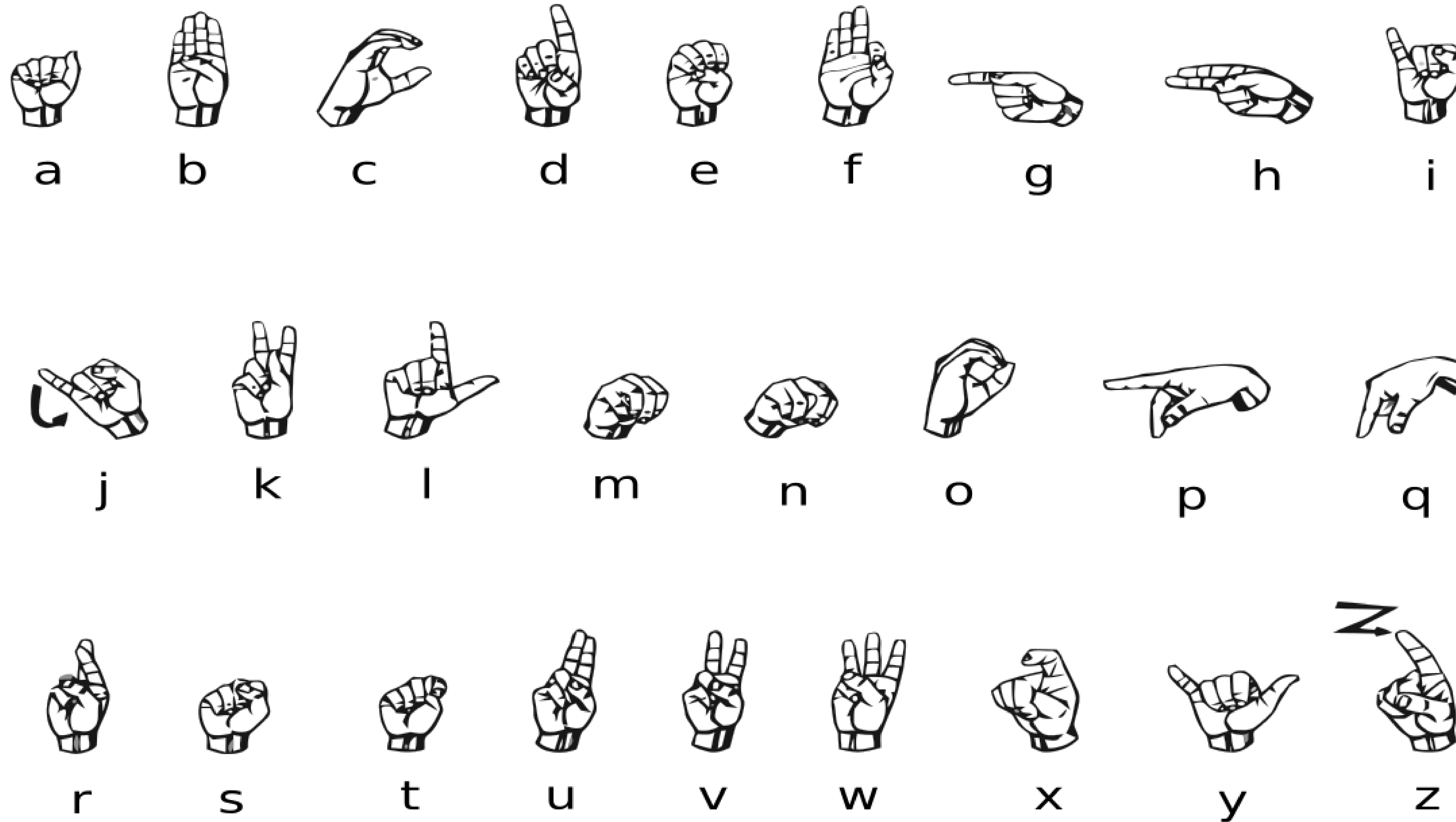
Historically important and difficult task for computers

Try learning like a Neural Network

Get exposed to the example, and try to figure out the rules to how it works

Second Exercise

The American Sign Language Alphabet





Questions?



Appendix: Gradient Descent

Helping the Computer Cheat Calculus

Learning from Error

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

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

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

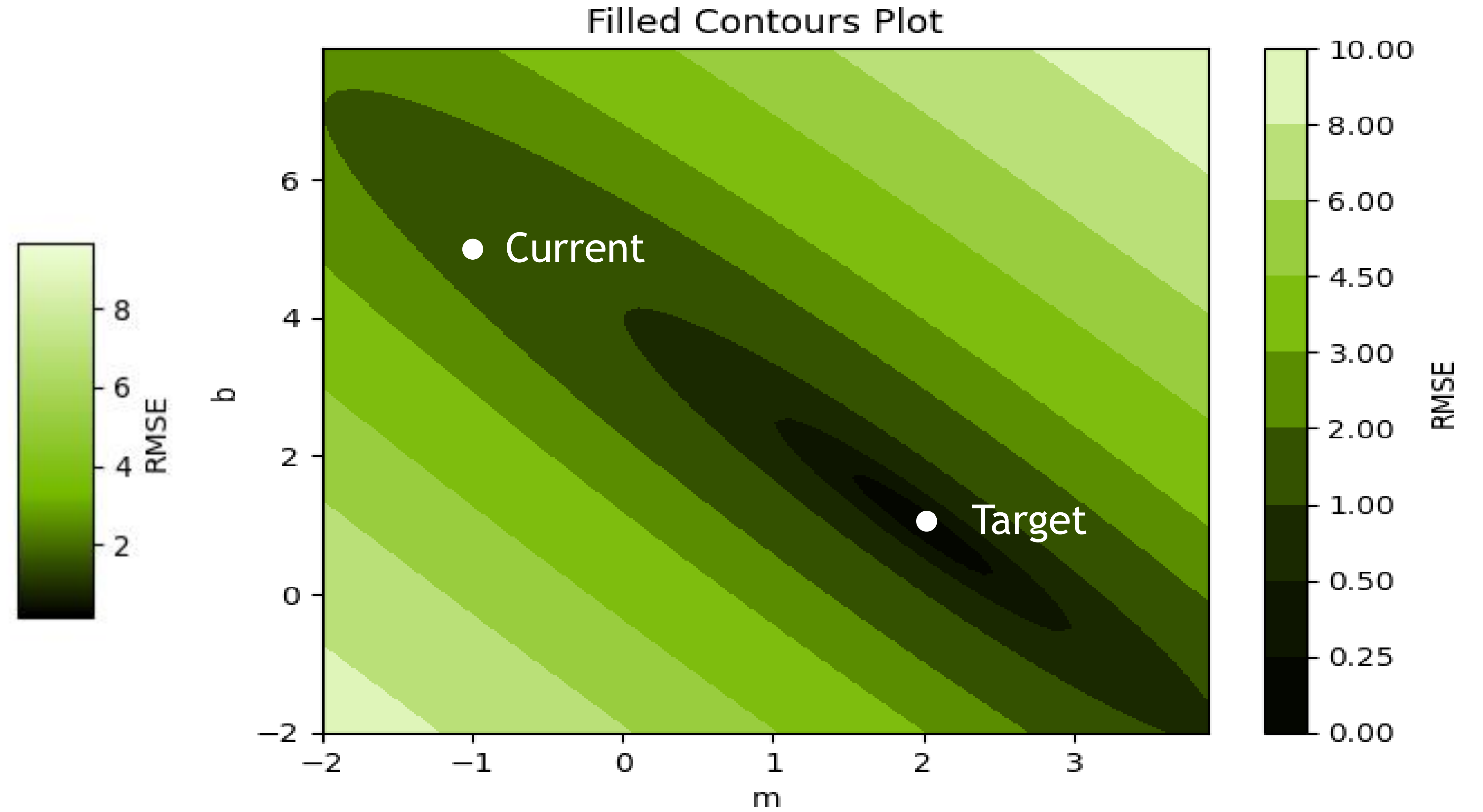
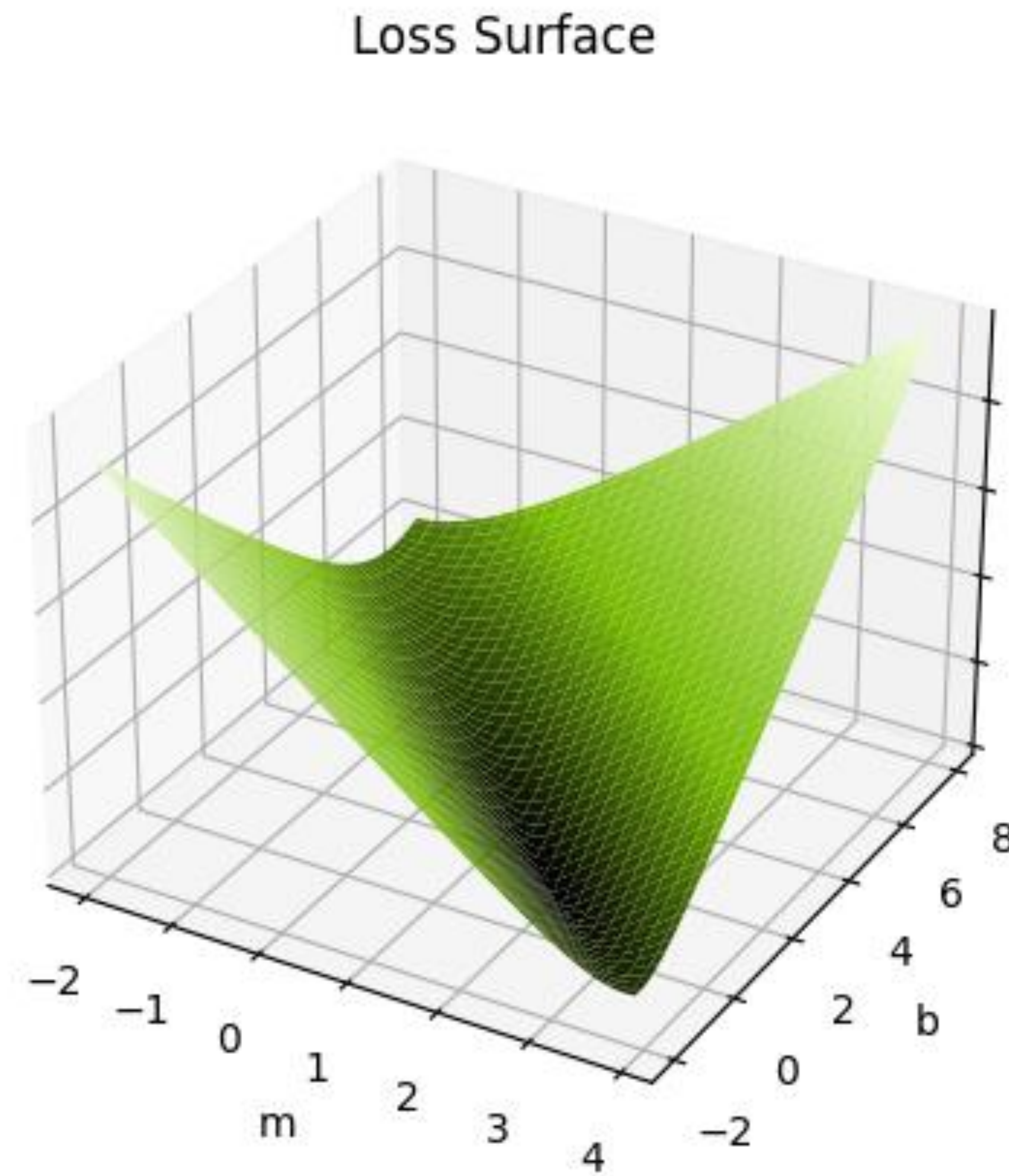
$$\frac{\partial MSE}{\partial b} = 3m + 2b - 8$$

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

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

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

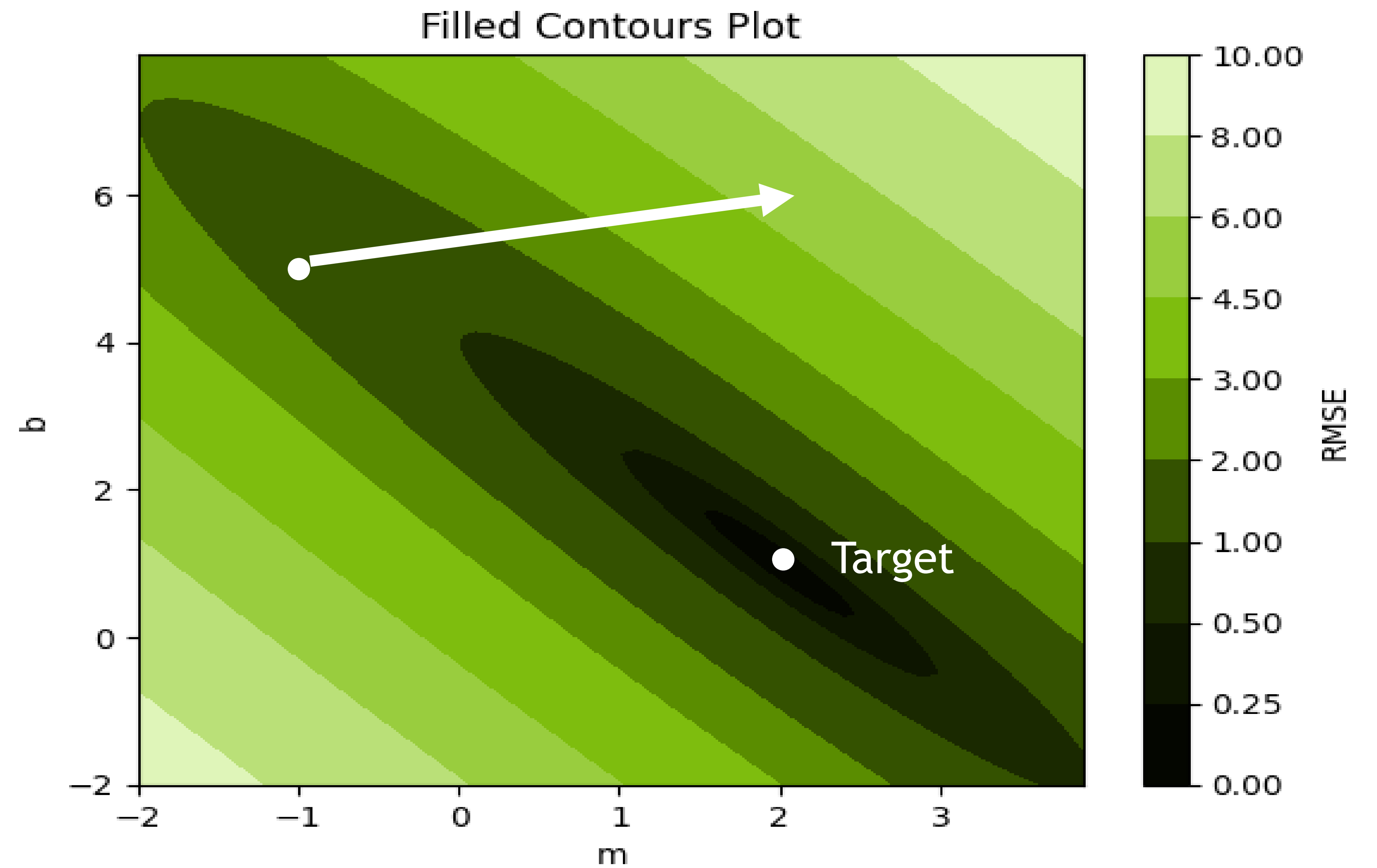
The Loss Curve



The Loss Curve

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

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

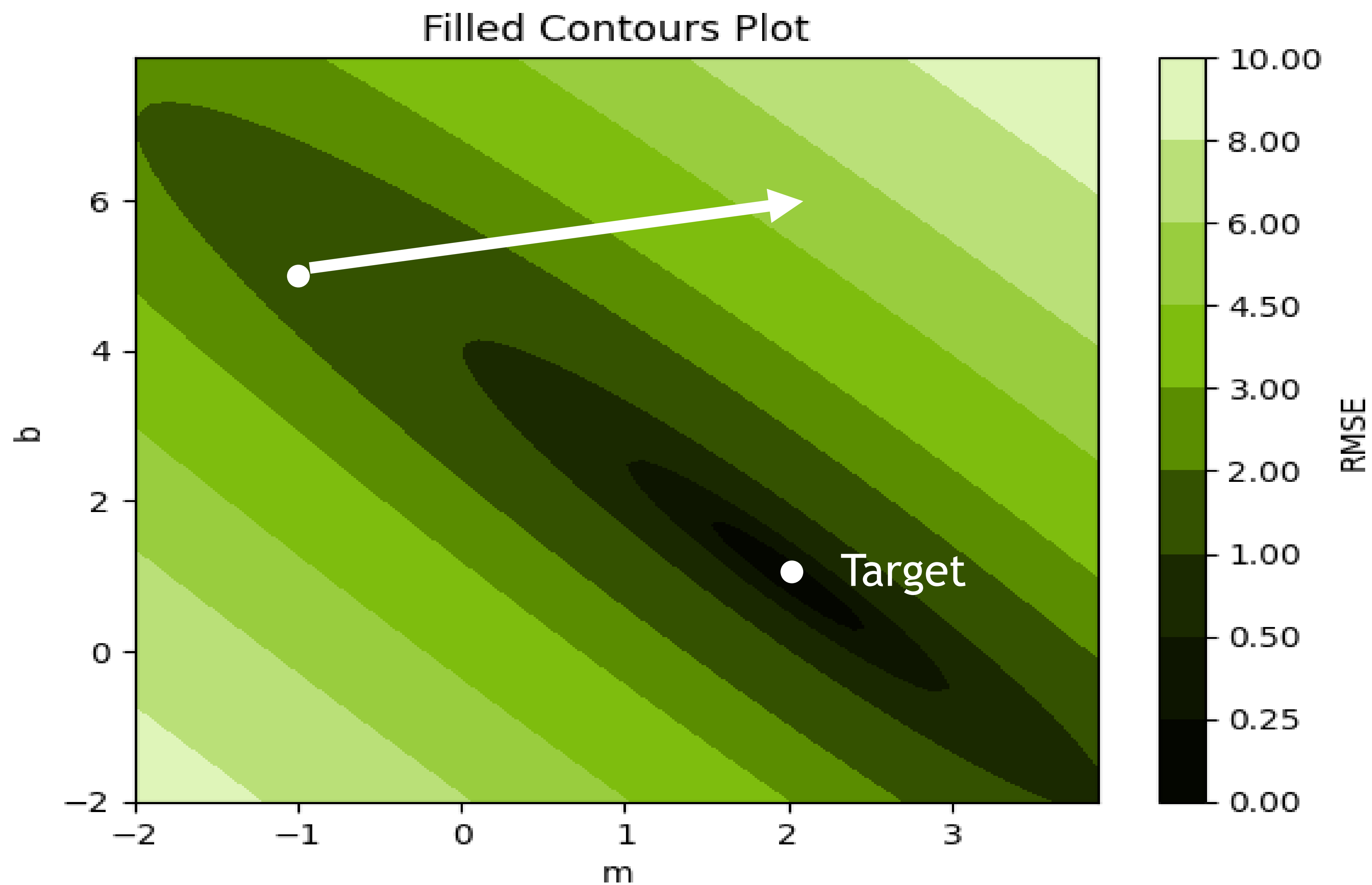


The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \quad \frac{\partial MSE}{\partial b} = -1$$

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

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



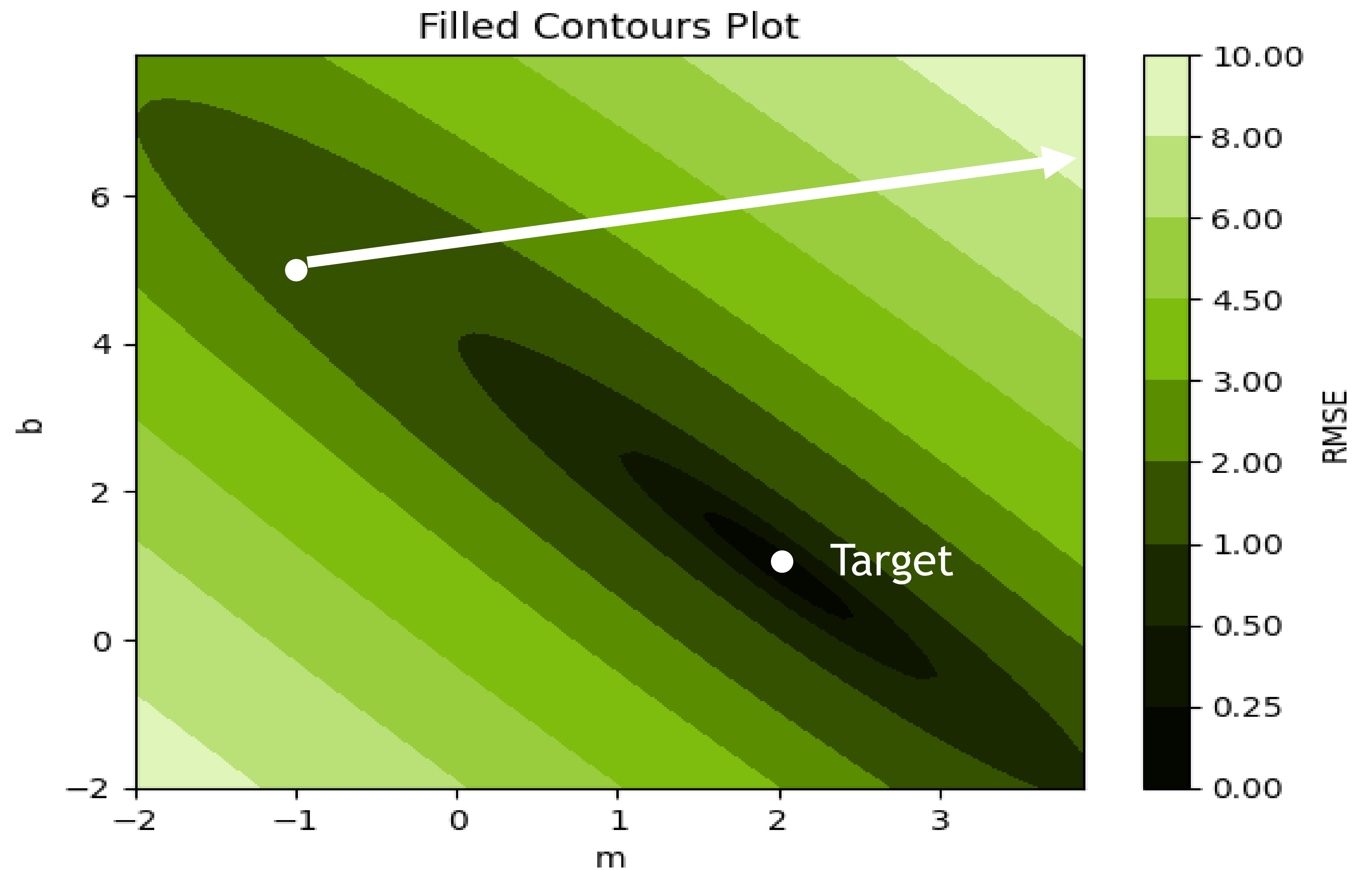
The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \quad \frac{\partial MSE}{\partial b} = -1$$

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

$$\lambda = 2$$

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



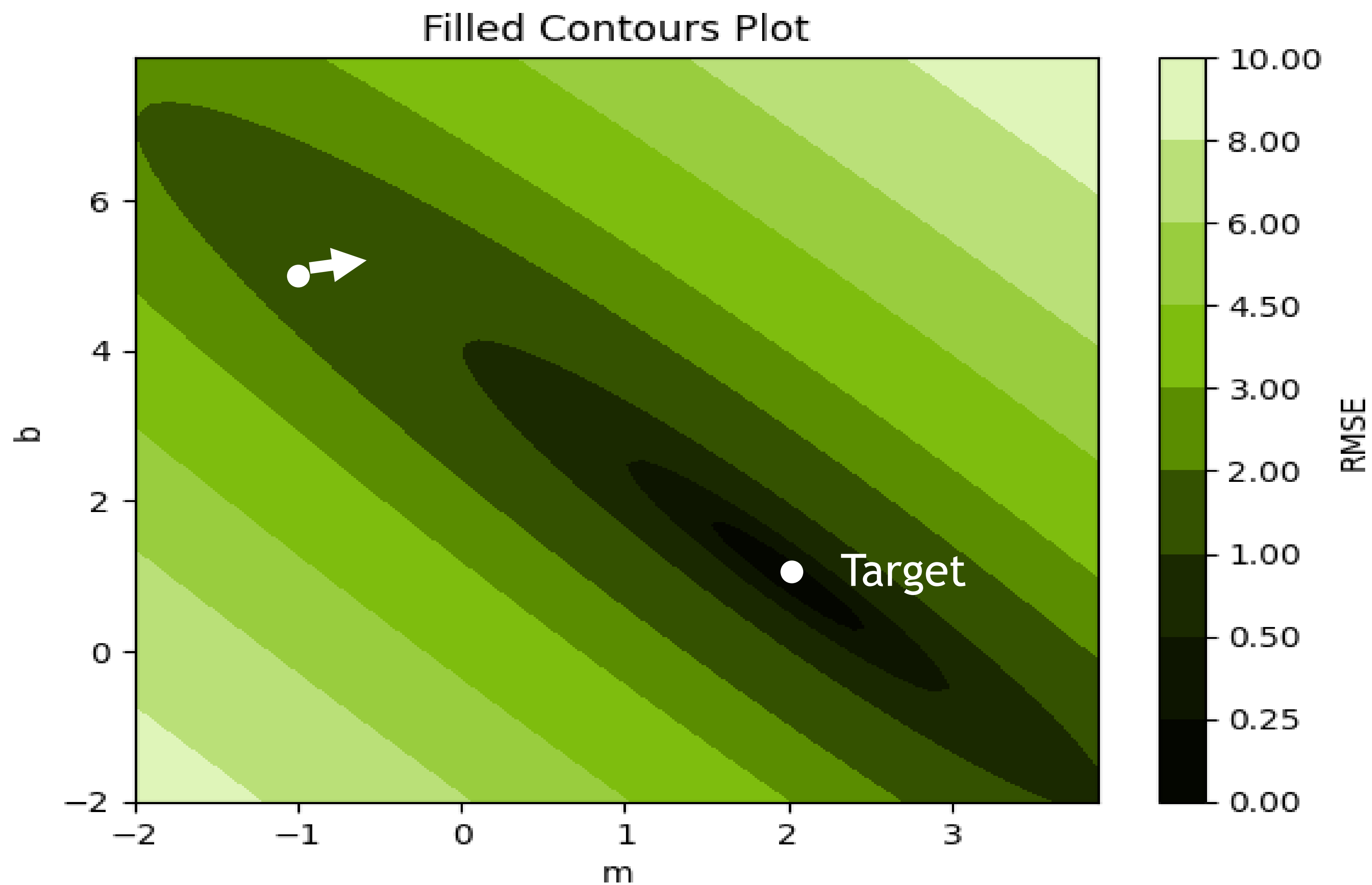
The Loss Curve

$$\frac{\partial MSE}{\partial m} = -3 \quad \frac{\partial MSE}{\partial b} = -1$$

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

$$\lambda = .1$$

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

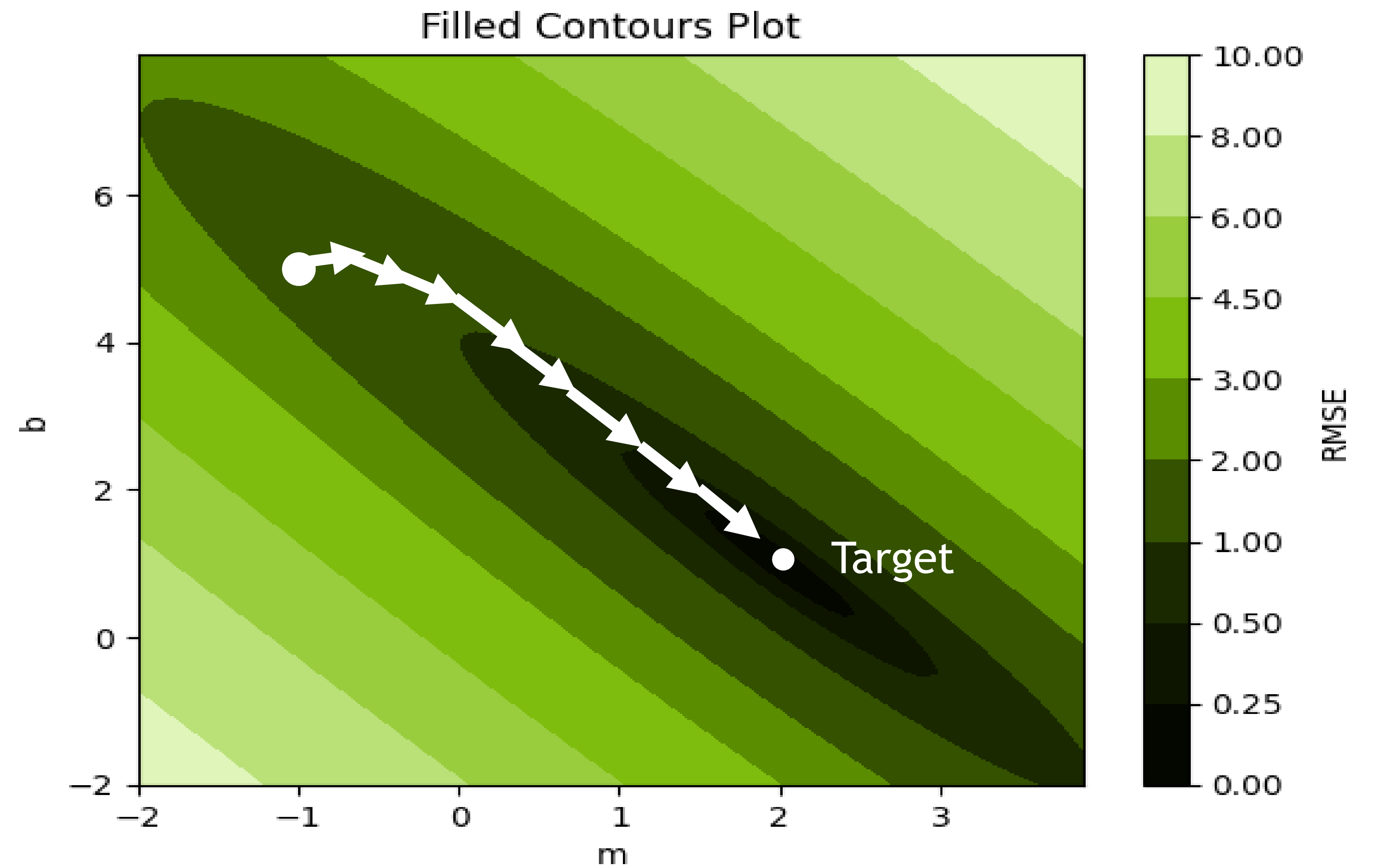


The Loss Curve

$$\lambda = .1$$

$$m := -1 + 3 \lambda = -0.7$$

$$b := 5 + \lambda = 5.1$$





Other

Recap of the Exercise

What just happened?

Loaded and visualized our data

Edited our data (reshaped, normalized, to categorical)

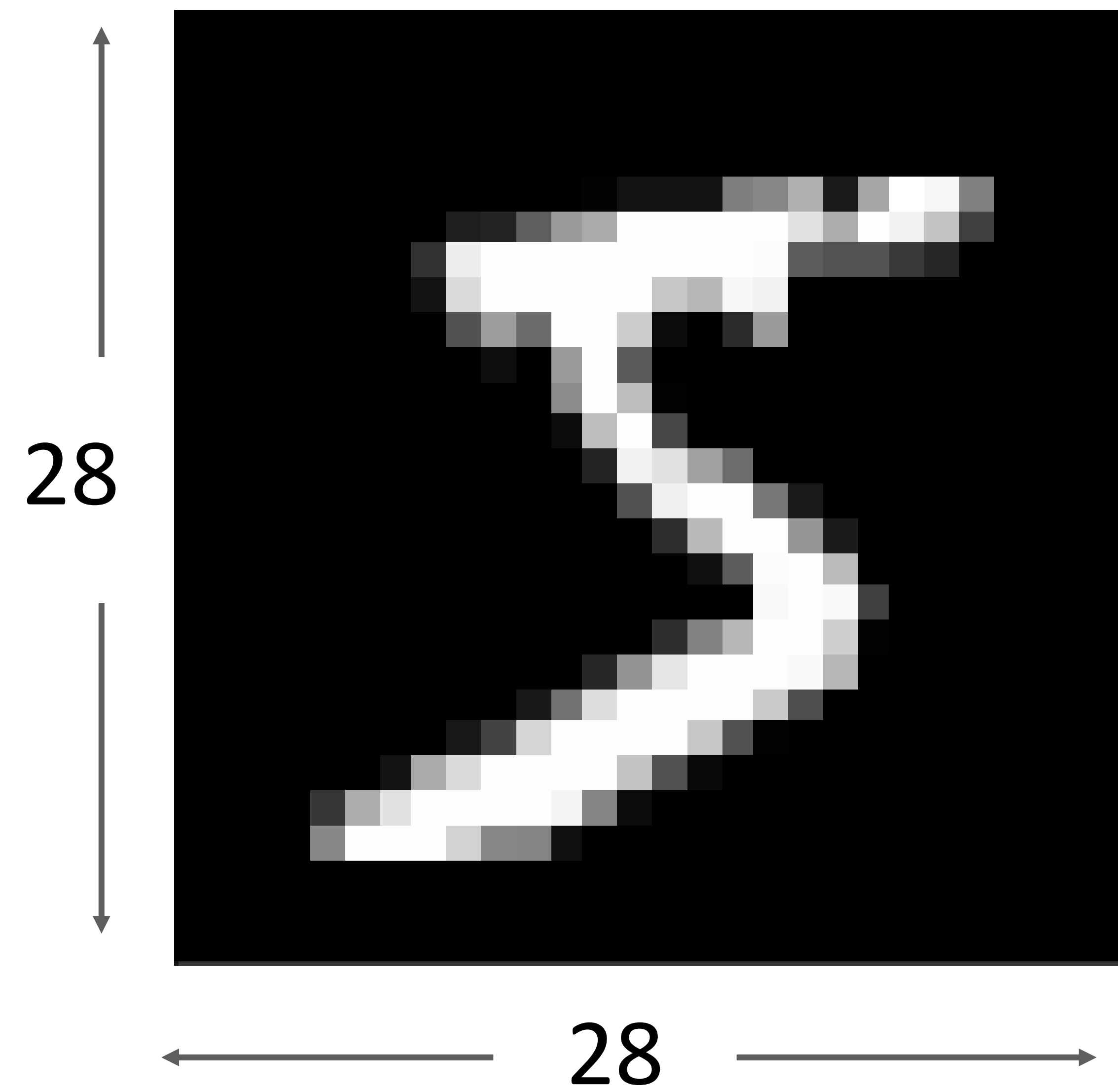
Created our model

Compiled our model

Trained the model on our data

Data Preparation

Input as an Array



[0,0,0,24,75,184,185,78,32,55,0,0,0...]

An Untrained Model

