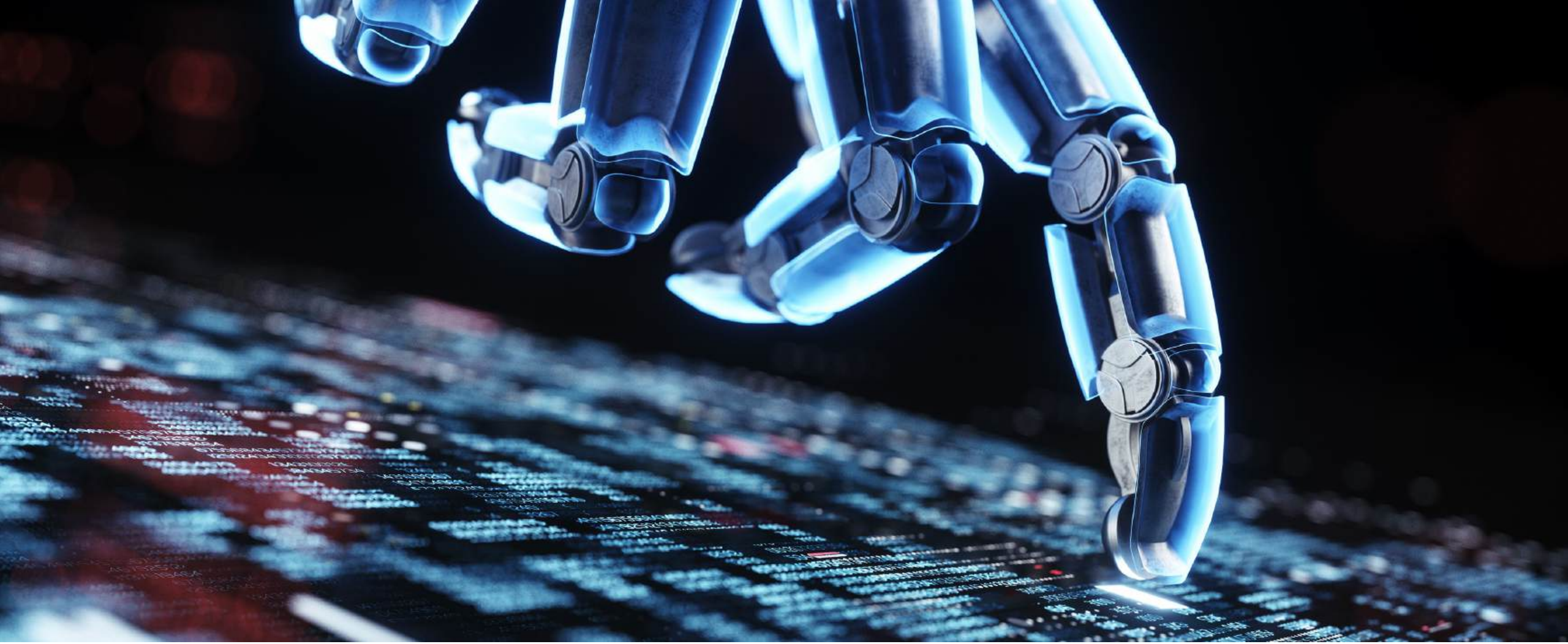


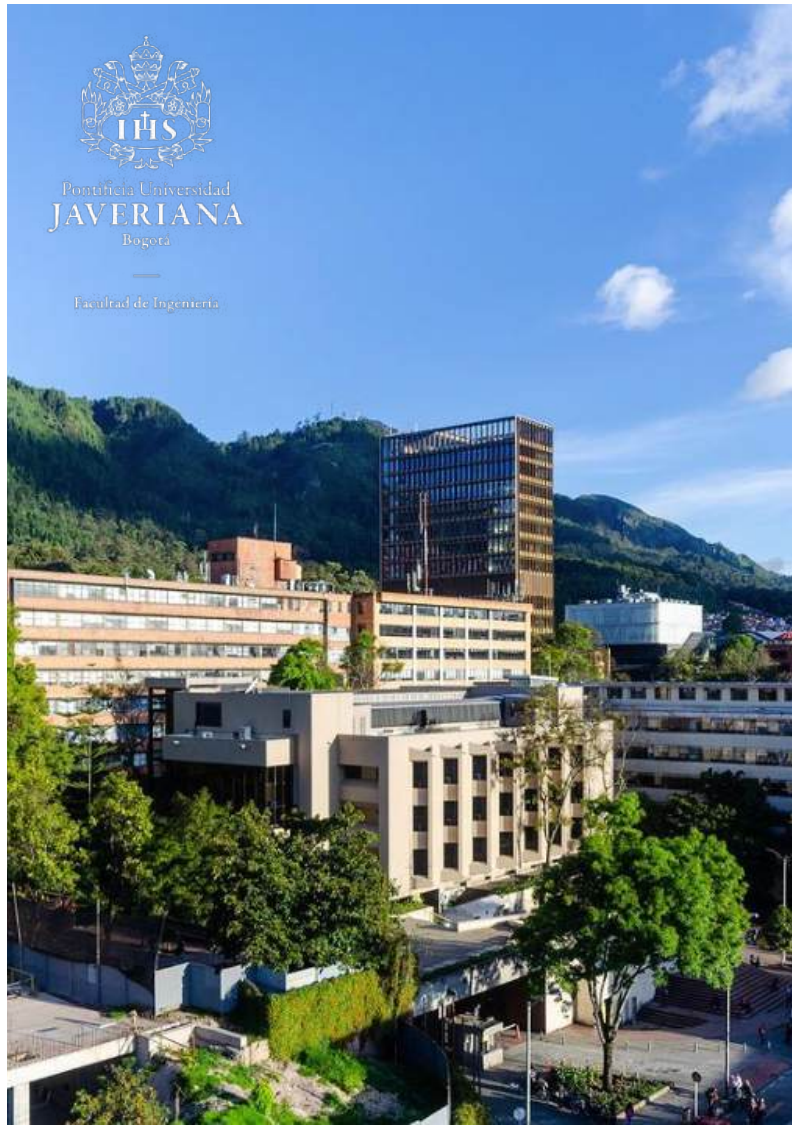
WALC 2024 Online
11 al 15 de Noviembre
27º Taller sobre Tecnologías de Redes Internet para América Latina y el Caribe
Fundación EsLaRed
Organización General

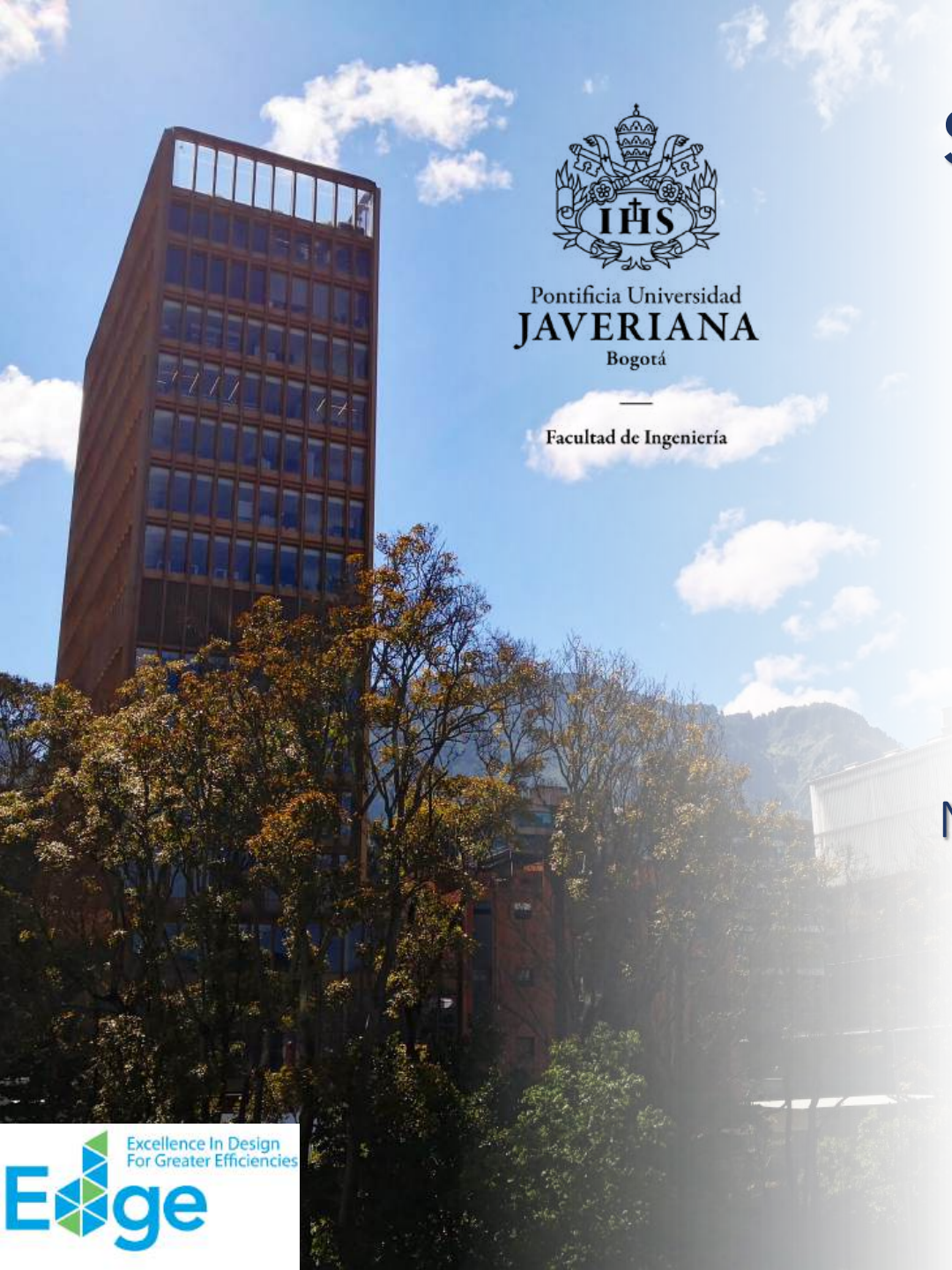
Pontificia Universidad JAVERIANA Bogotá
Facultad de Ingeniería

Machine Learning Fundamentals

Workshop para América Latina y el Caribe (WALC)
Track 3 – Inteligencia Artificial Aplicada
November 12, 2024

Pontificia Universidad Javeriana – Bogotá, Colombia





School of Engineering

- 3.750 students (3000 undergrad)
- 104 professors
- 20.000 alumni

4
Departments

- 8 undergraduate programs
- 7 specializations
- 12 master programs
- 2 doctoral programs

2
Centers of Excellence

New Research and Laboratories Building

- 14.082 m^2 (700 for student workspaces)
- 93 meters high
- 15 floors and 3 basements
- 10.000 pieces of state-of-the-art equipment
- https://ingenieria.javeriana.edu.co/nuestro-edificio

1
Institutes

Excellence In Design For Greater Efficiencies
Edge

Diego Méndez Chaves, Ph.D

School of Engineering – Pontificia Universidad Javeriana

Website: https://perfilesycapacidades.javeriana.edu.co/en/persons/diego-mendez-chaves

email: diego-mendez@javeriana.edu.co

- Associate Professor at the Department of Electronics Engineering.
- Director of the Master Program in Internet of Things.
- Director of the Master Program in Electronics Engineering.
- Technical Director of the Center of Excellence and Adoption in IoT (CEA-IoT)
- Research associate at the Marconi Lab in the International Centre for Theoretical Physics (ICTP), Trieste - Italy.
- TinyML Academic Network – Coordinator.
- Research interests: IoT, embedded systems, wireless sensor networks, participatory sensing, digital systems design and embedded operating systems.

Track 3

Inteligencia Artificial Aplicada

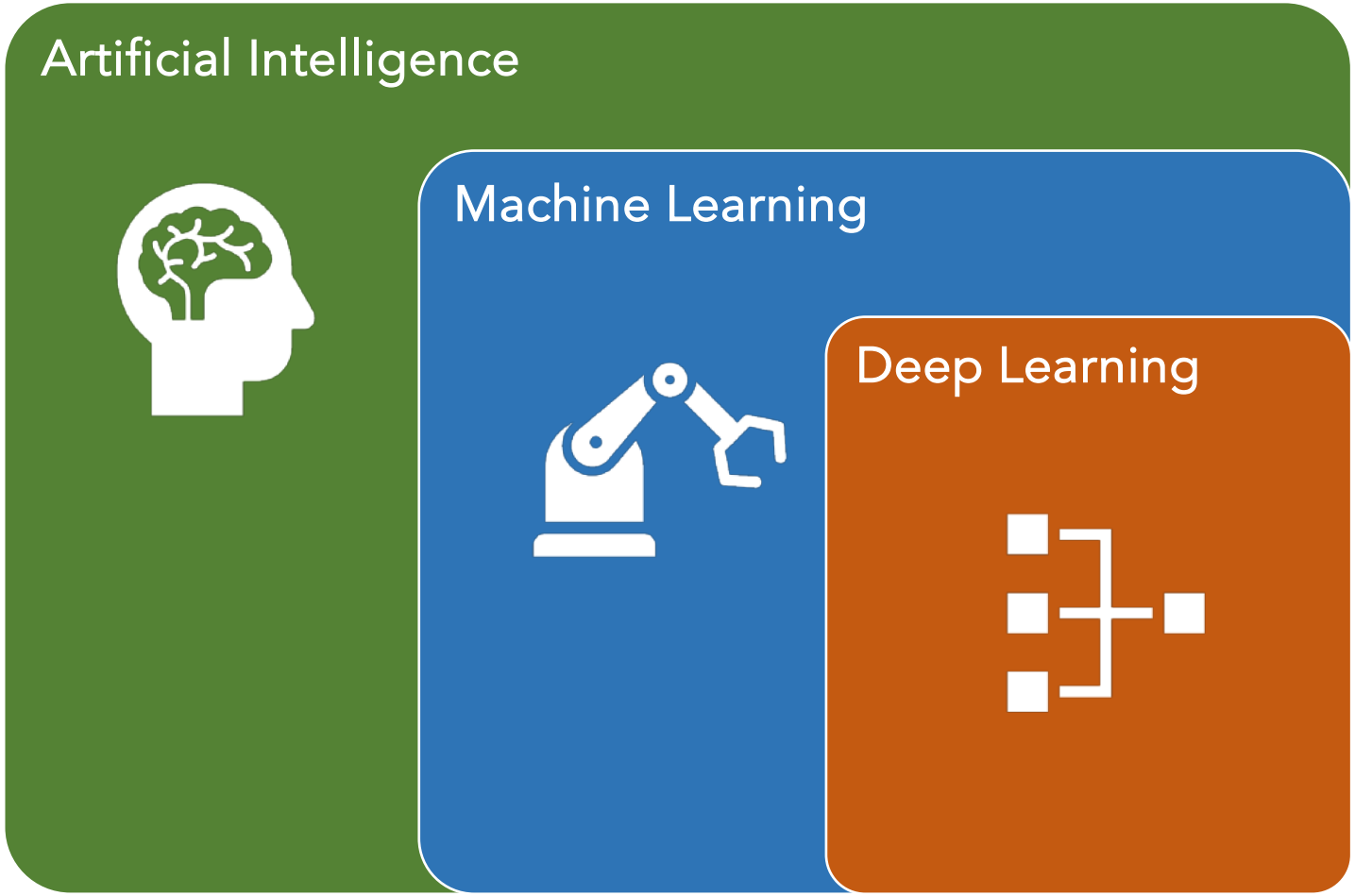
Agenda

- **Parte 1:**
  - Introducción
  - El paradigma de ML
  - Exploración de la función de pérdida y costo
  - Redes neuronales artificiales
- **Parte 2:**
  - Dense Neural Networks – Regresión
  - Dense Neural Networks – Clasificación
  - Métricas de ML

Alexa, play some rock music!

Playing the Rock Hits playlist

AI vs. ML vs. DL



Artificial intelligence (AI): any technique that enables computers to mimic human intelligence.

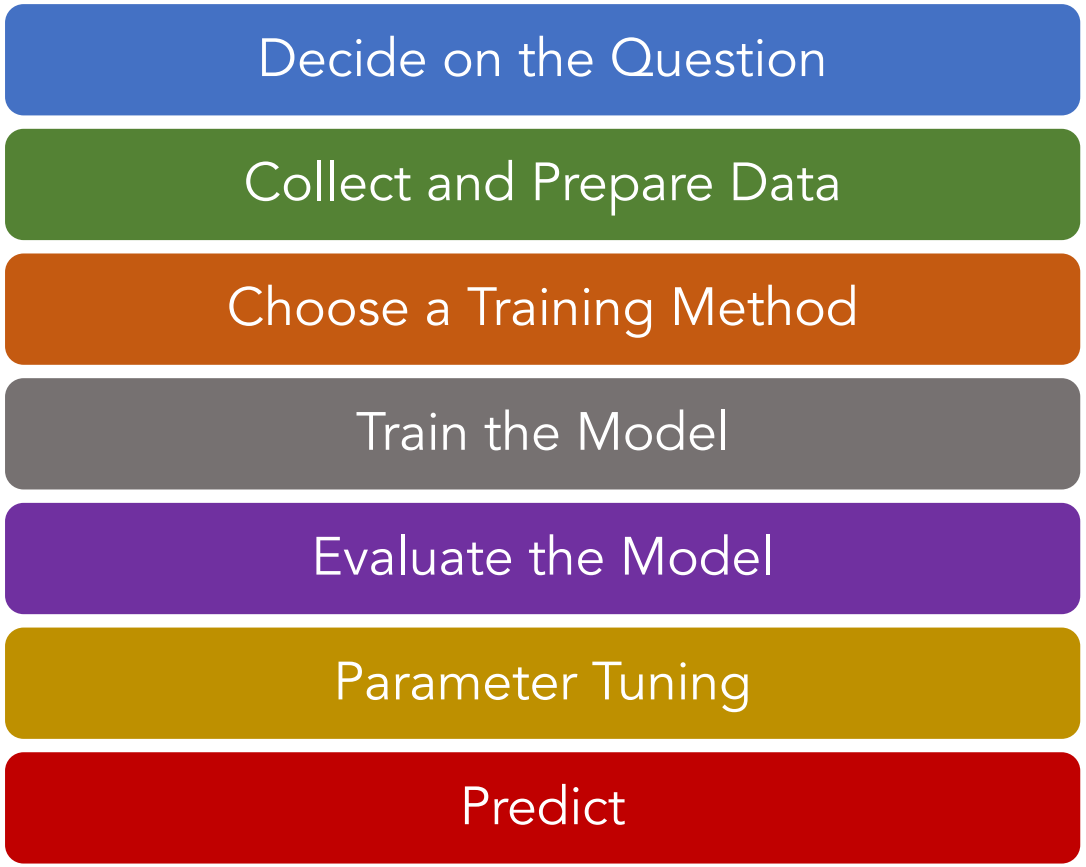
Machine learning (ML): a subset of AI that uses techniques that enable machines to use experience to improve at tasks.

Deep learning (DL): a subset of ML based on artificial neural networks (ANN). The learning process is deep because of its structure.

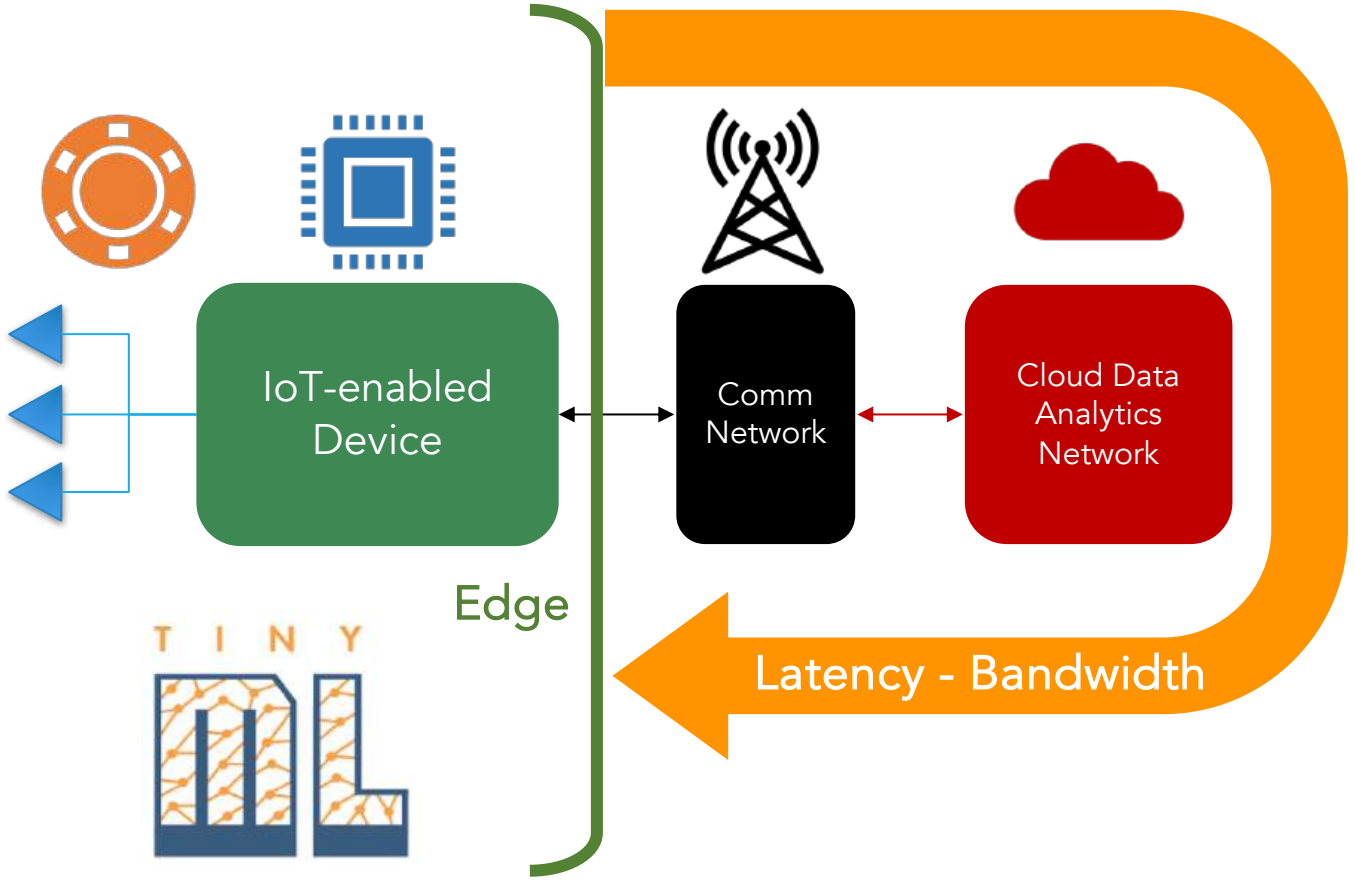
Based on https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

General Steps for Machine Learning

On a high level, the craft of creating machine learning (ML) processes is comprised of several steps:



Based on https://microsoft.github.io/ML-For-Beginners/#/1-Introduction/4-techniques-of-ML/README?id=techniques-of-machine-learning



"The future of ML is ***tiny*** and bright."

3 main components

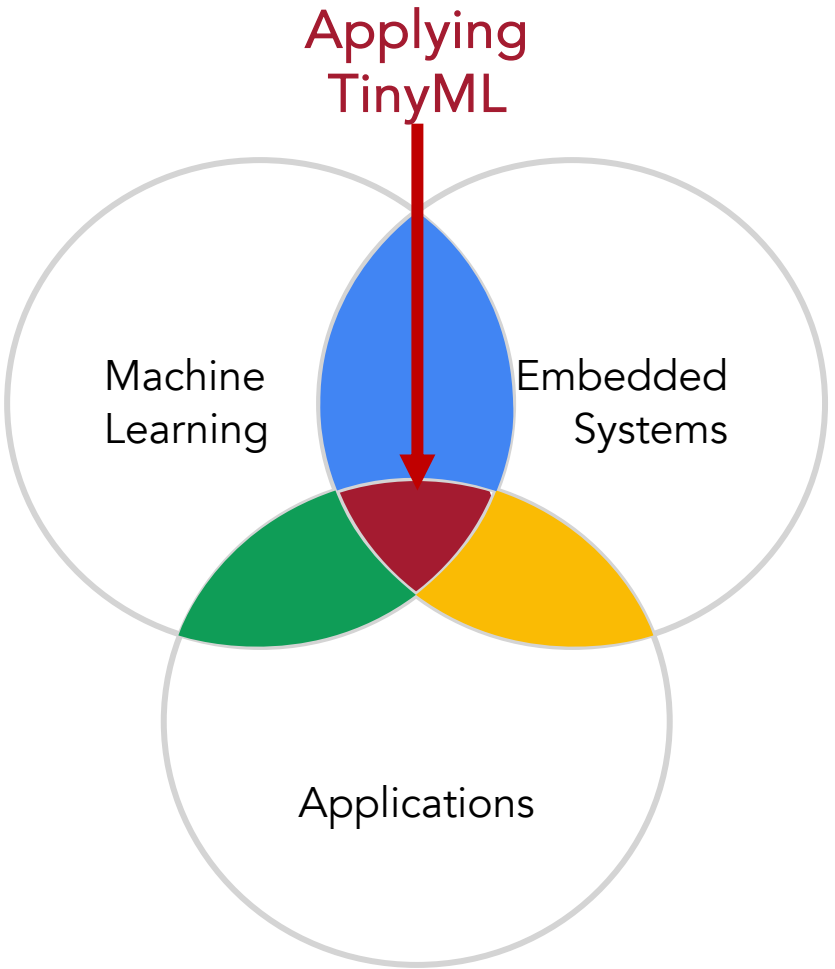
"Edge AI is a truly complete technology. As a topic, it makes use of knowledge from everything from the physical properties of semiconductor electronics all the way up to the engineering of high-level architectures that span devices and the cloud. It demands expertise in the most cutting-edge approaches to artificial intelligence and machine learning along with the most venerable skills of bare-metal embedded software engineering. It makes use of the entire history of computer science and electrical engineering, laid out end to end."

Situnayake, Daniel; Plunkett, Jenny
AI at the Edge (pp. 215-216)
O'Reilly Media

3 main components

Interactions

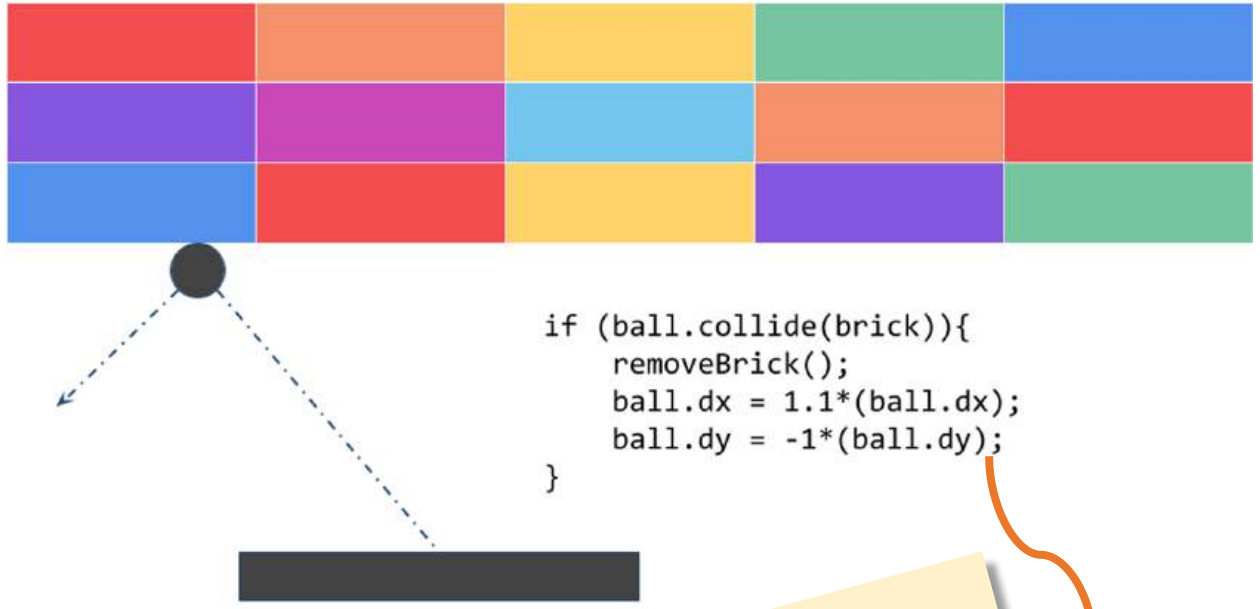
Given your understanding of things at these various intersections, you will have a deep understanding for how to apply TinyML



The Machine Learning Paradigm

Explicit Coding

- Defining rules that determine behavior of a program
- Everything is pre-calculated and pre-determined by the programmer
- Scenarios are limited by program complexity



```
if (ball.collide(brick)){
    removeBrick();
    ball.dx = 1.1*(ball.dx);
    ball.dy = -1*(ball.dy);
}
```

Rules
- If the ball collides with the brick:
  - Remove brick
  - Change ball direction
  - Increase ball speed
- …

https://en.wikipedia.org/wiki/Breakout_(video_game)

The Traditional Programming Paradigm

Consider Activity Detection

```
if(speed<4){
    status=WALKING;
}
```

```
if(speed<4){
    status=WALKING;
} else {
    status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// ???
```

Way too complex to code!

The Traditional Programming Paradigm

The Machine Learning Paradigm

Activity Detection with Machine Learning

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING

The Machine Learning Paradigm

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010
```

Label = WALKING

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011
```

Label = RUNNING

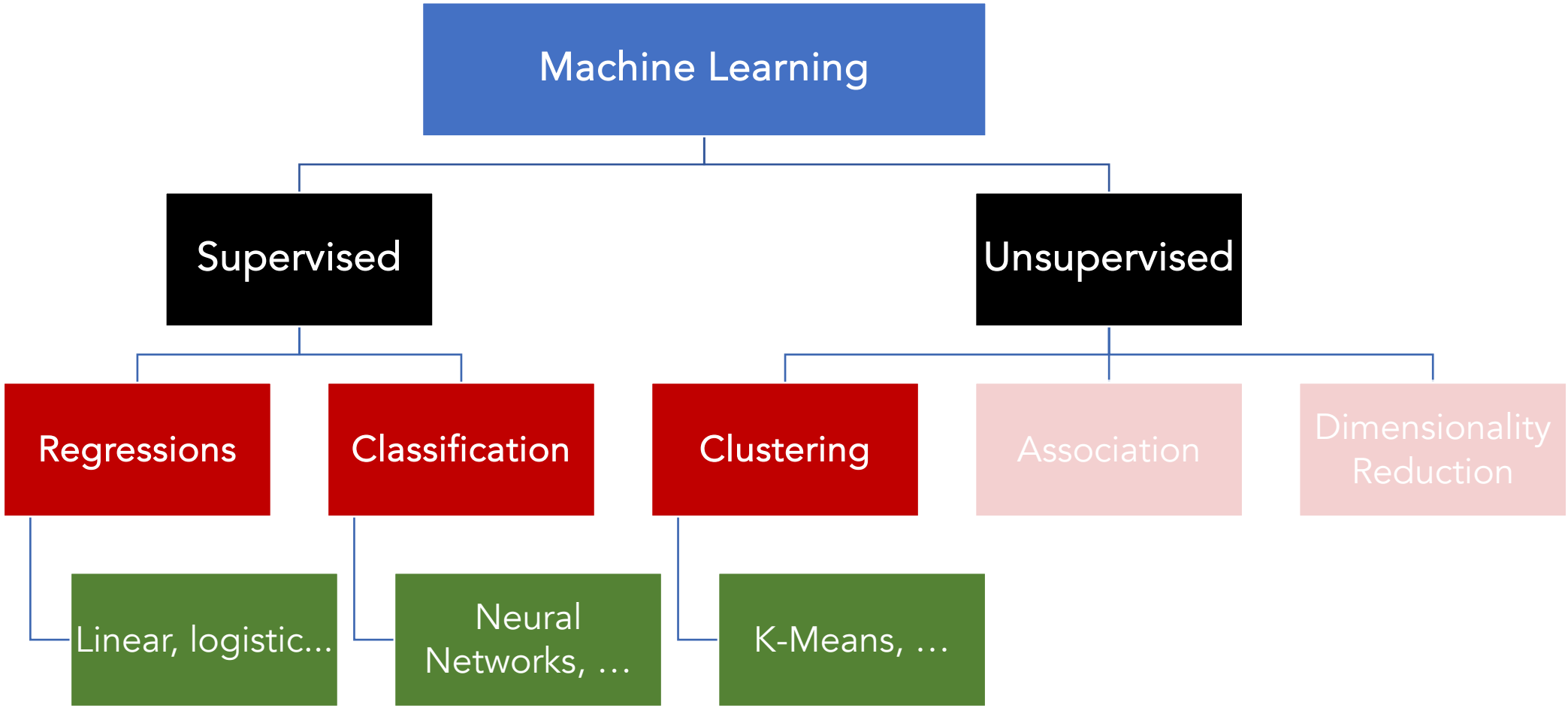
```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

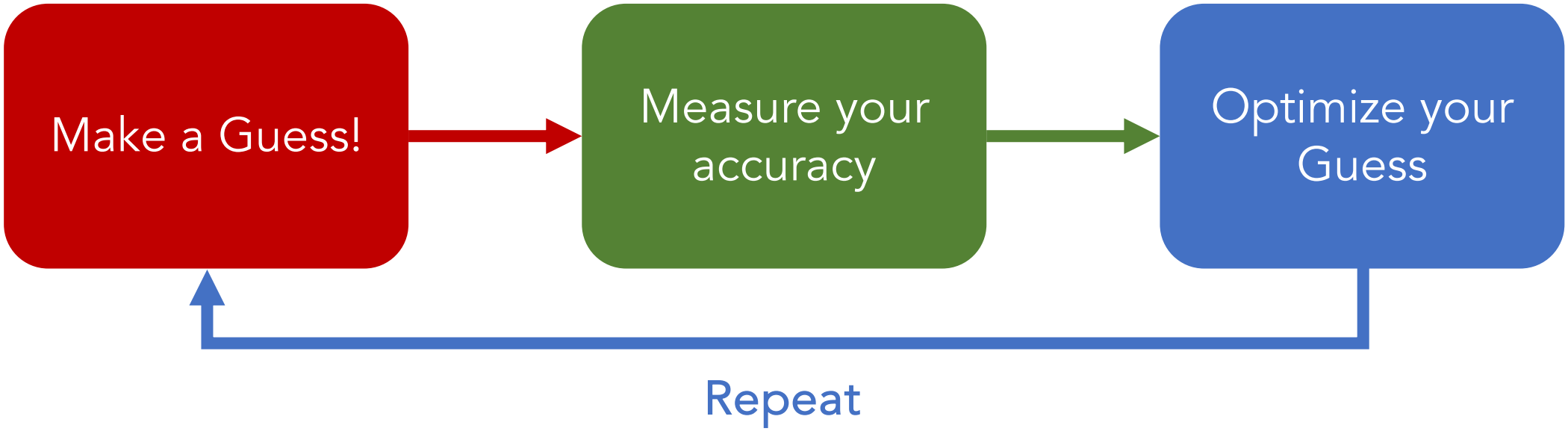
```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING

Two Approaches

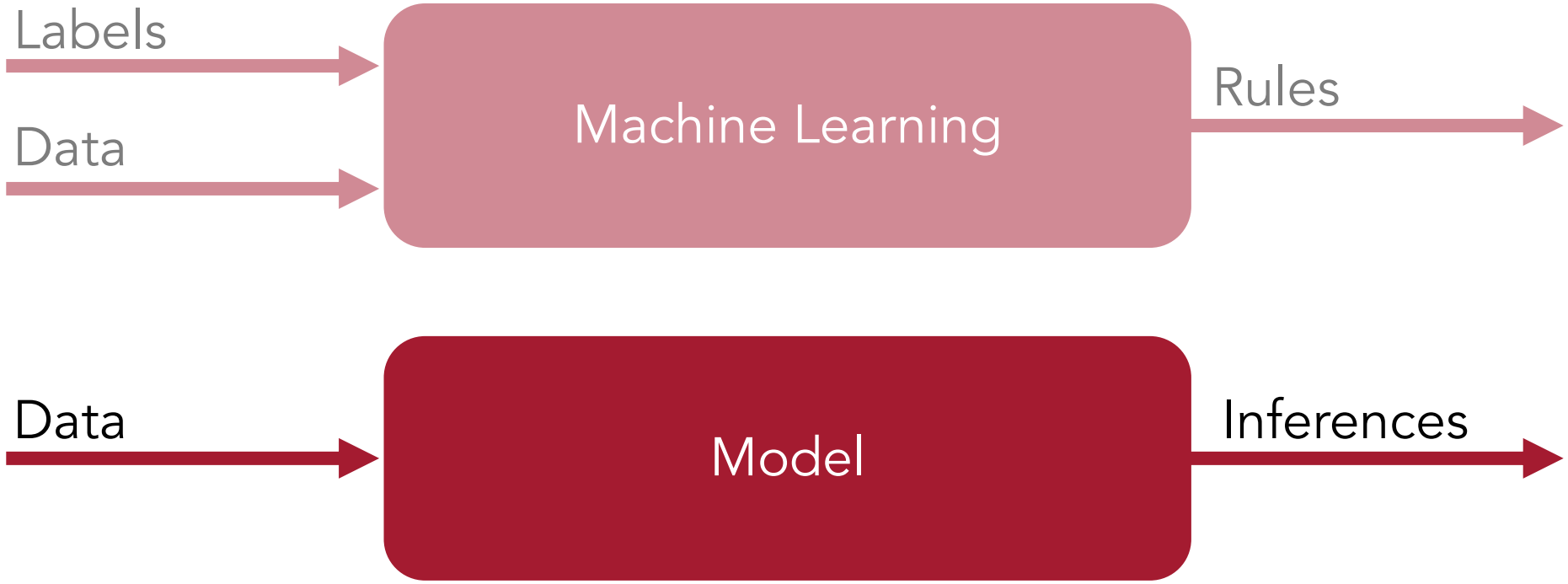


The Machine Learning Paradigm



The Machine Learning Paradigm

The Machine Learning Paradigm



How good is your model?

a way to measure your accuracy

Matching X to Y

$X = \{ -1, 0, 1, 2, 3, 4 \}$

$Y = \{ -3, -1, 1, 3, 5, 7 \}$

https://pixabay.com/photos/thinking-person-person-thinking-2681494/

Make a guess!

$Y = 3X - 1$

$X = \{ -1, 0, 1, 2, 3, 4 \}$

My $Y = \{ -4, -1, 2, 5, 8, 11 \}$

How good is the guess?

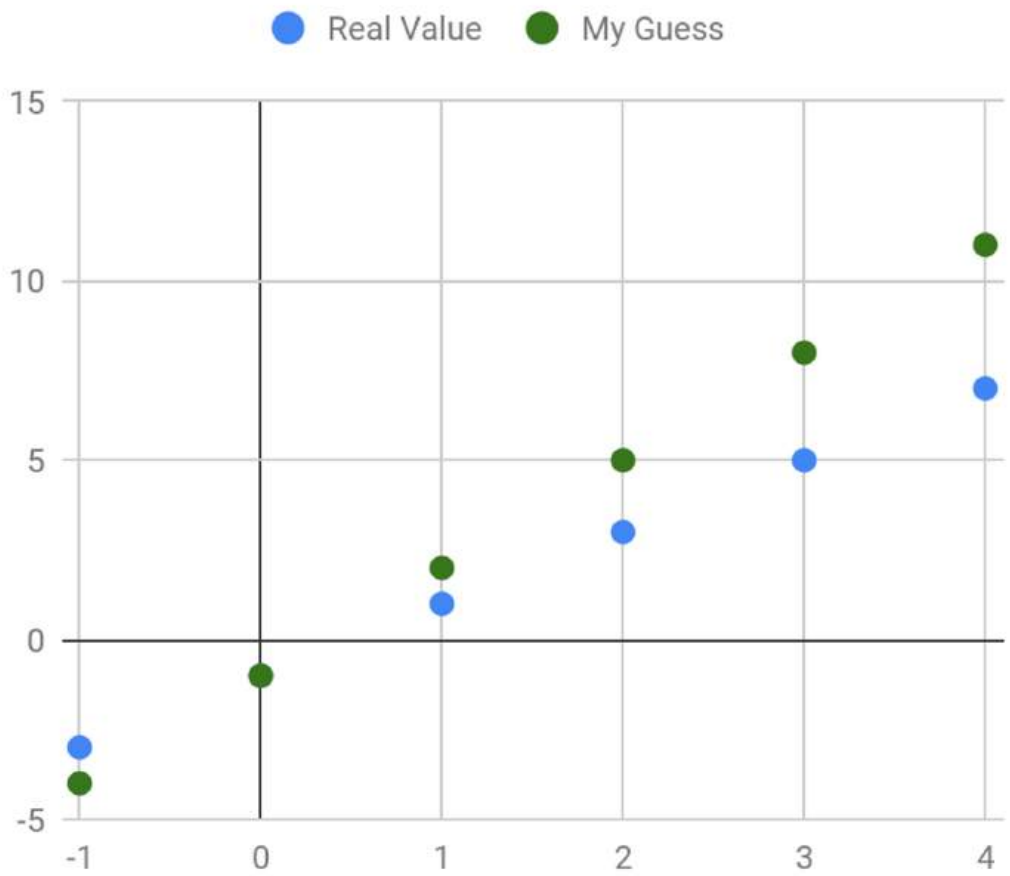
$Y = 3X - 1$

$X = \{ -1, 0, 1, 2, 3, 4 \}$

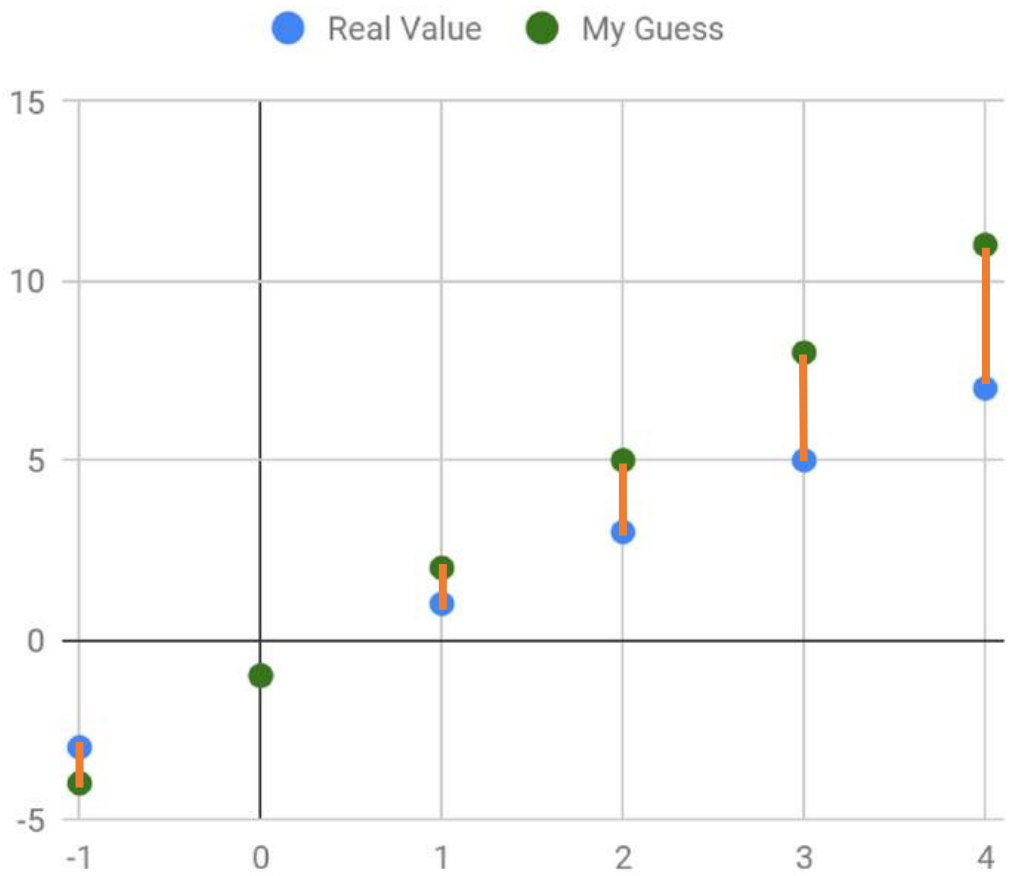
My $Y = \{ -4, -1, 2, 5, 8, 11 \}$

Real $Y = \{ -3, -1, 1, 3, 5, 7 \}$

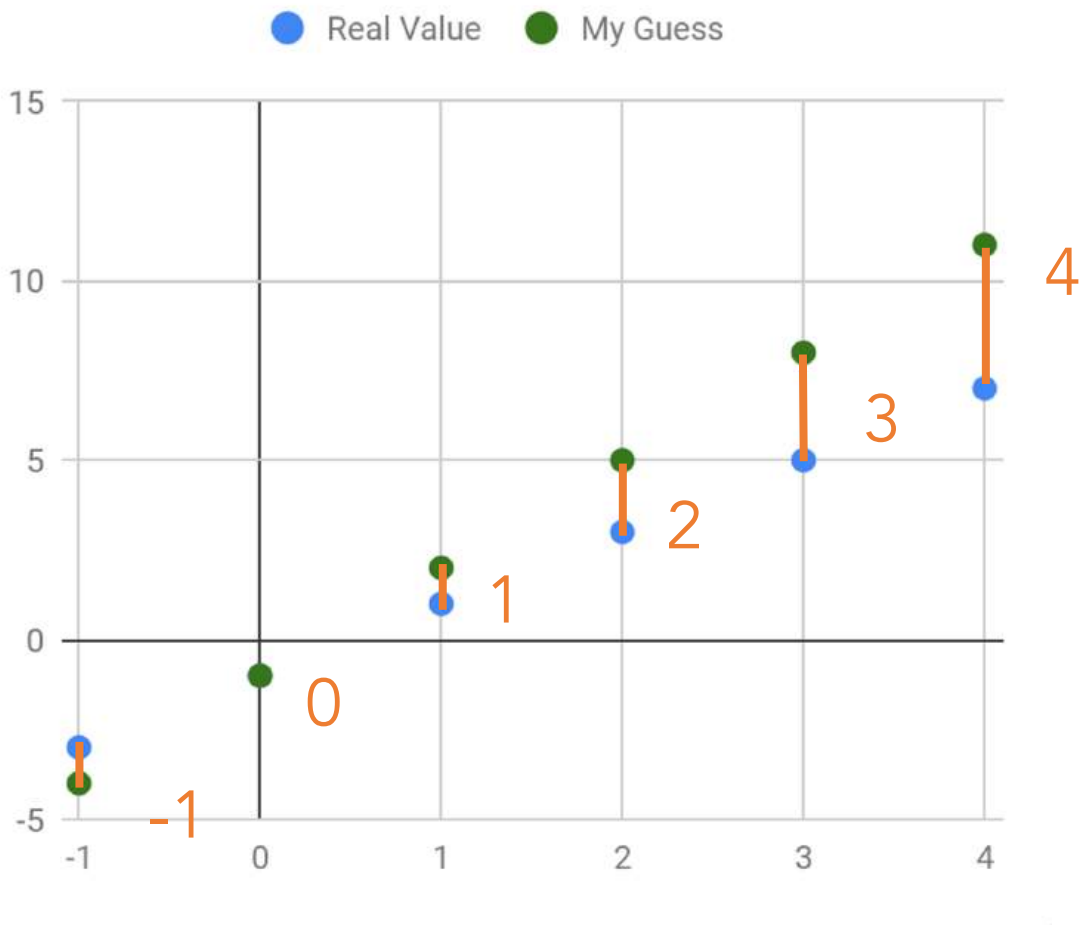
Let’s measure it!



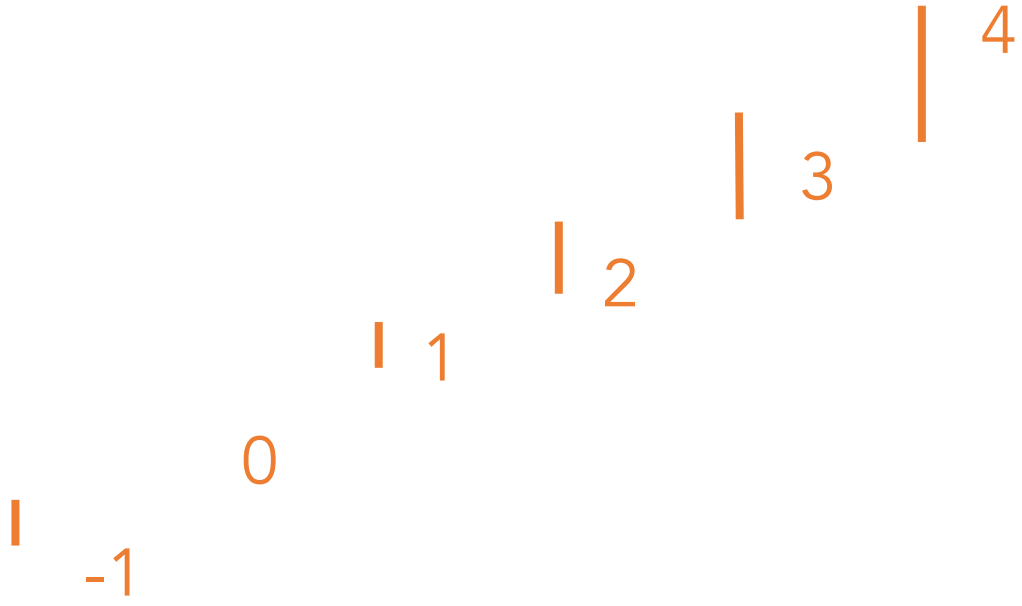
Let’s measure it!



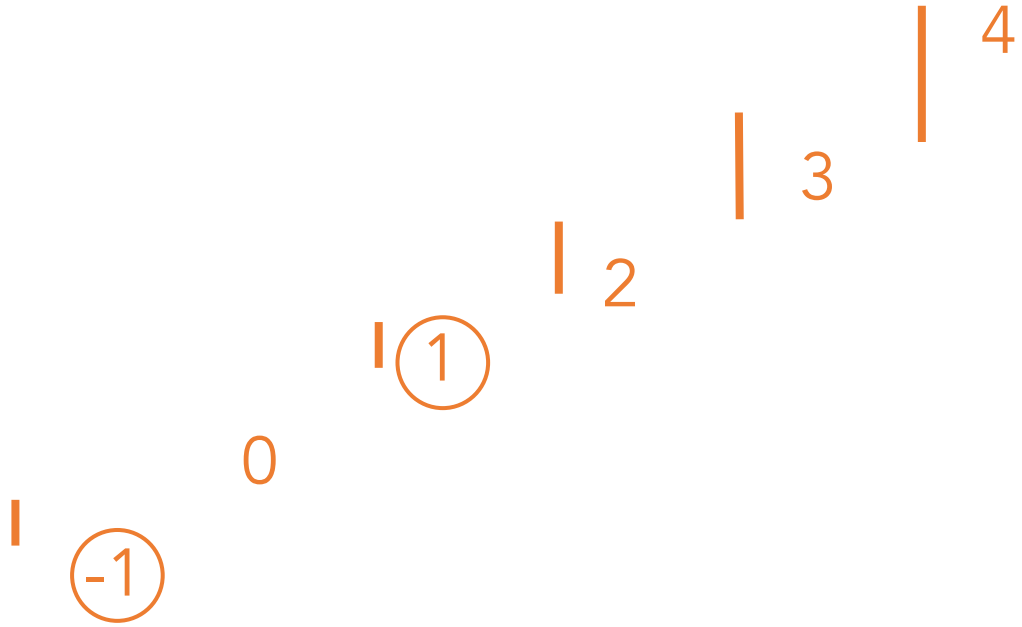
Let’s measure it!



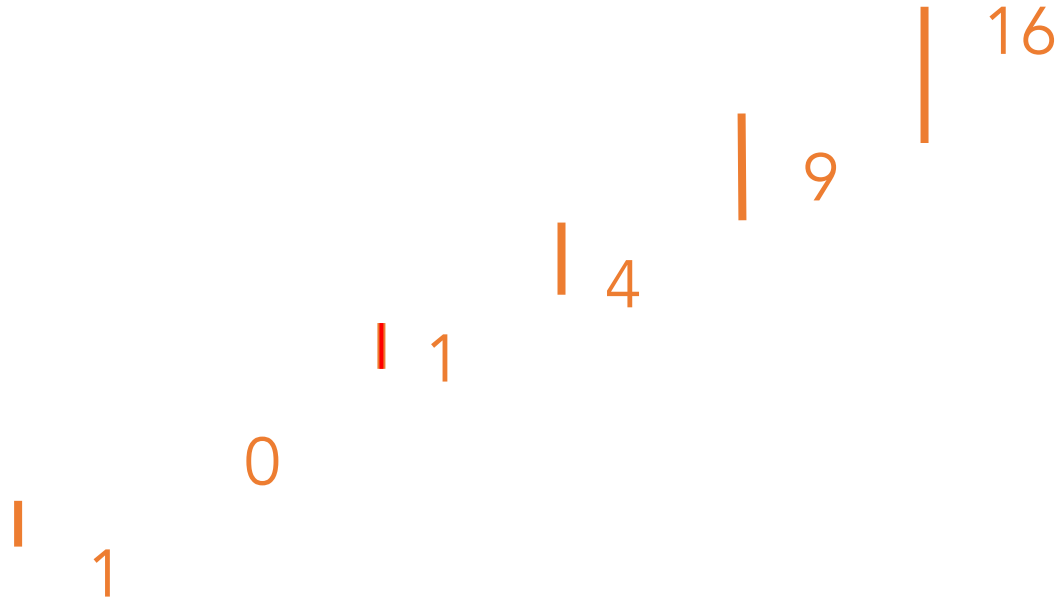
Let’s measure it!

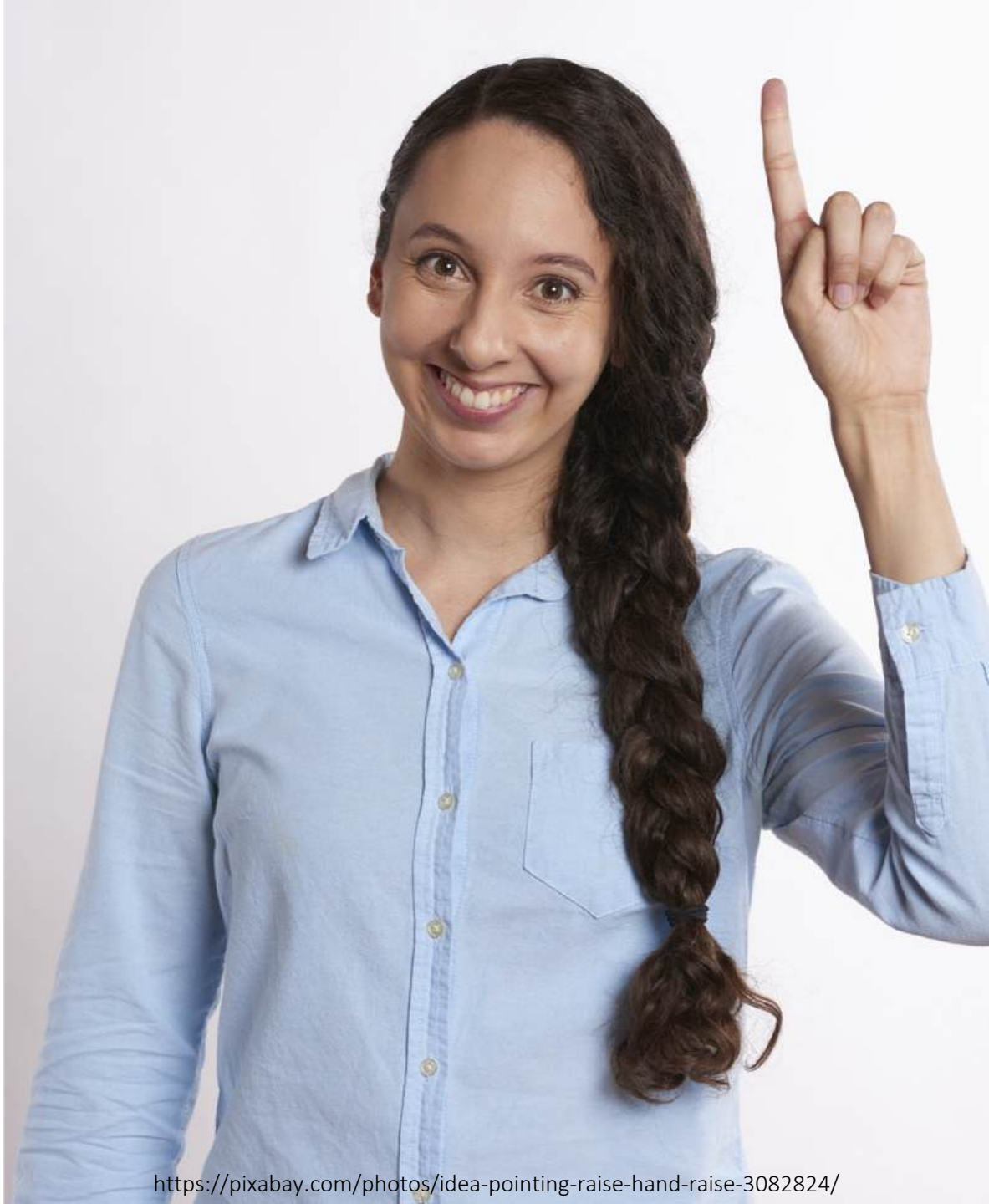


Houston, we have a problem!



What if we **square**^{2} them?





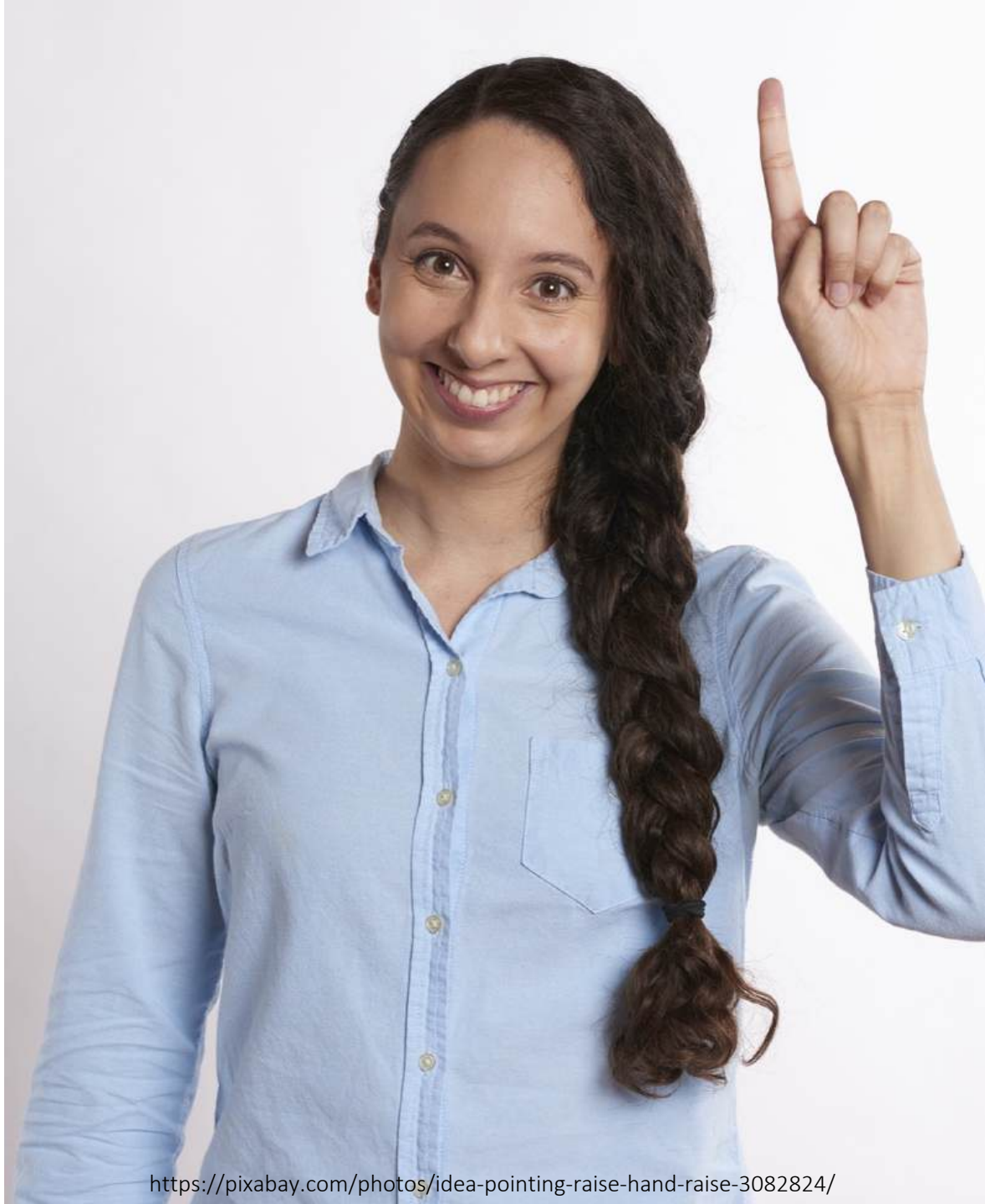
https://pixabay.com/photos/idea-pointing-raise-hand-raise-3082824/

Total that (∑) and take the square root √

sqrt(1 + 1 + 4 + 9 + 16)

= sqrt(31)

= 5.57



https://pixabay.com/photos/idea-pointing-raise-hand-raise-3082824/

Make another guess!

$Y = 2X - 2$

$X = \{ -1, 0, 1, 2, 3, 4 \}$

My $Y = \{ -4, -2, 0, 2, 4, 6 \}$

Real $Y = \{-3, -1, 1, 3, 5, 7\}$

$Diff^2 = \{1, 1, 1, 1, 1\}$

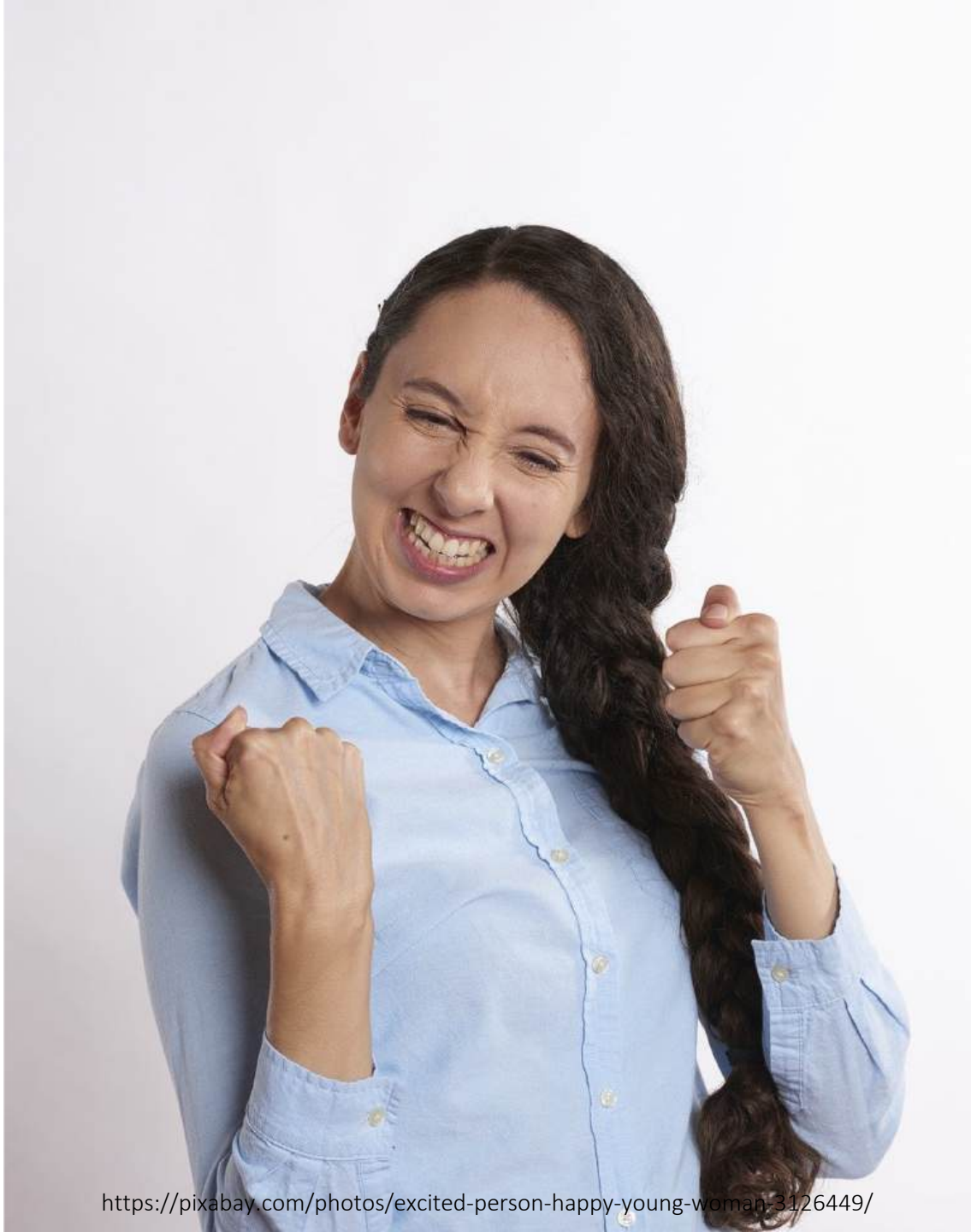
https://pixabay.com/photos/thinking-person-person-thinking-2681494/

Get the same difference, repeat the same process.

sqrt(1 + 1 + 1 + 1 + 1)

= sqrt(5)

= 2.23



https://pixabay.com/photos/excited-person-happy-young-woman-3126449/

Make another guess!

$Y = 2X - 1$

$X = \{ -1, 0, 1, 2, 3, 4 \}$

My $Y = \{ -3, -1, 1, 3, 5, 7 \}$

Real $Y = \{-3, -1, 1, 3, 5, 7\}$

$Diff^2 = \{0, 0, 0, 0, 0\}$

https://pixabay.com/photos/sunglasses-cool-fashion-style-2649326/

Make another guess!

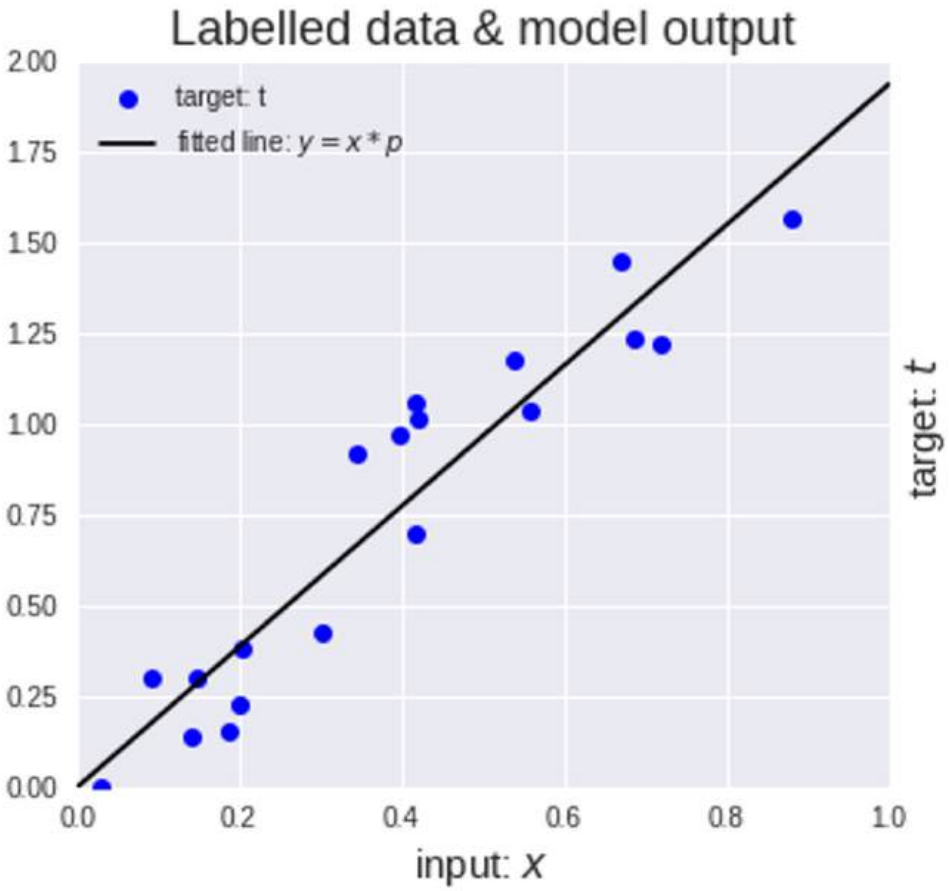
$Y = 2X - 1$

$X = \{ -1, 0, 1, 2, 3, 4 \}$

My $Y = \{ -3, -1, 1, 3, 5, 7 \}$

Real $Y = \{-3, -1, 1, 3, 5, 7\}$

Diff^2 $= \{0, 0, 0, 0, 0, 0\}$

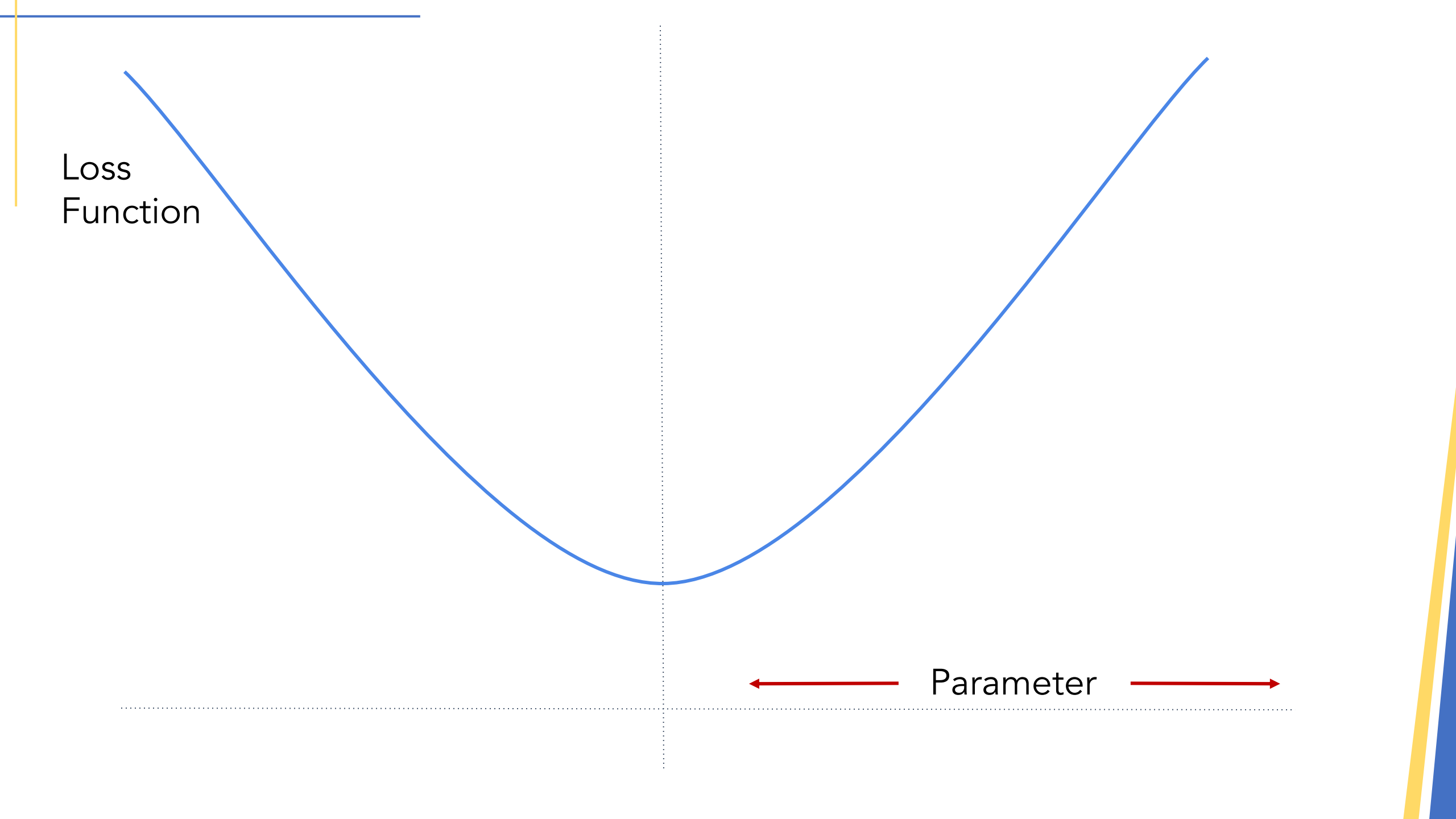


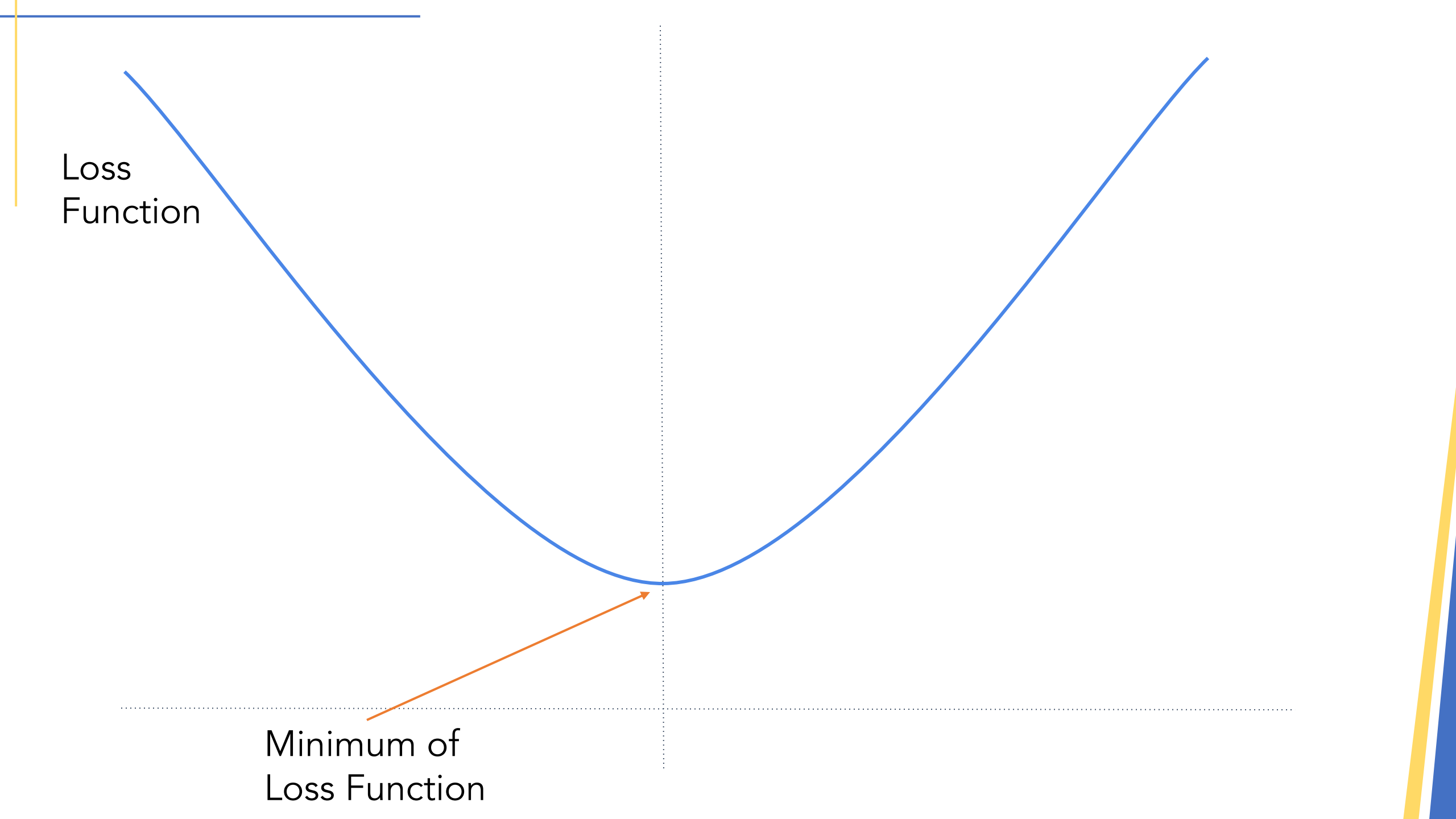
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

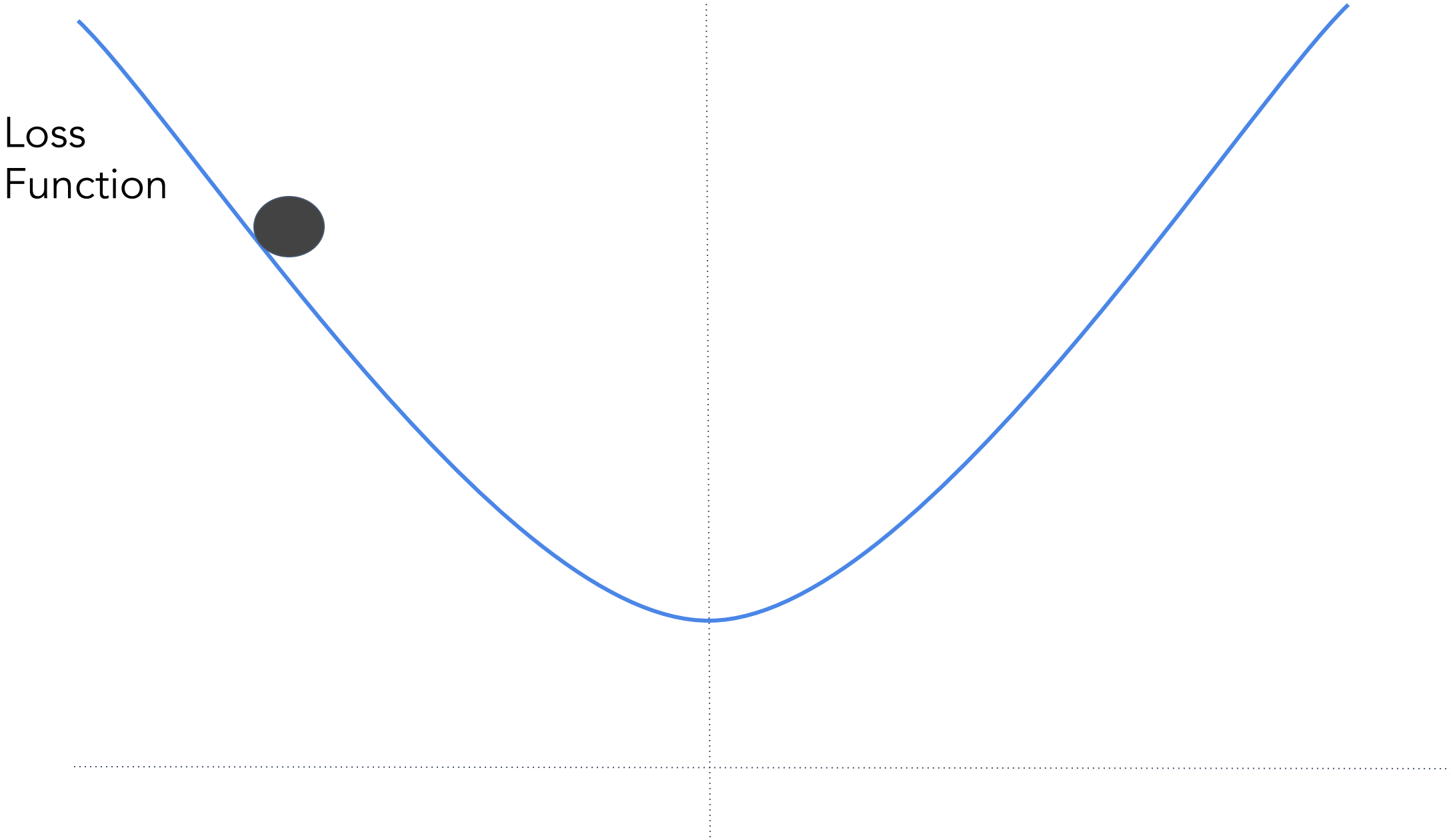
Goal is to Minimize MSE (Mean Squared Error)

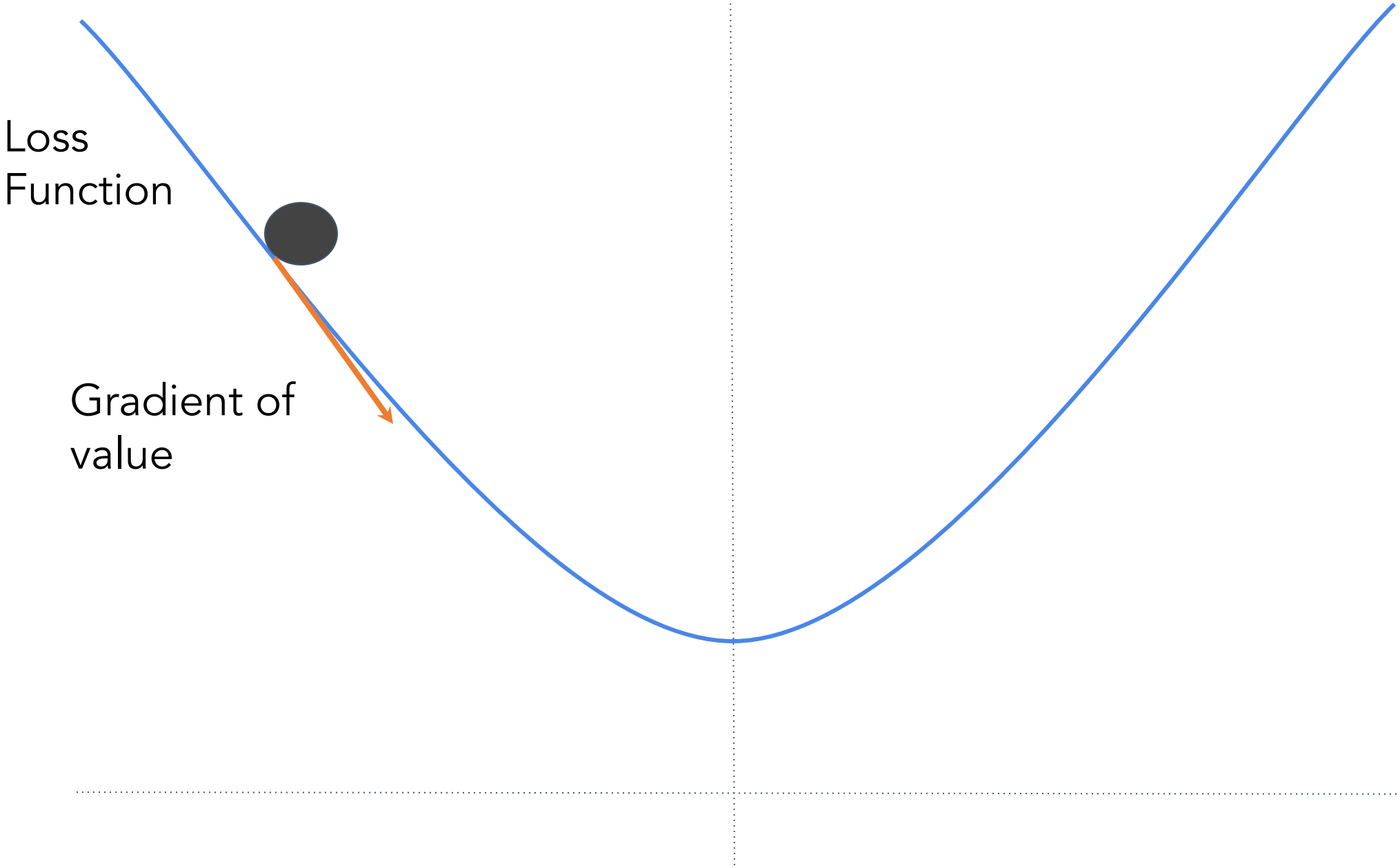
Finding out the best solution

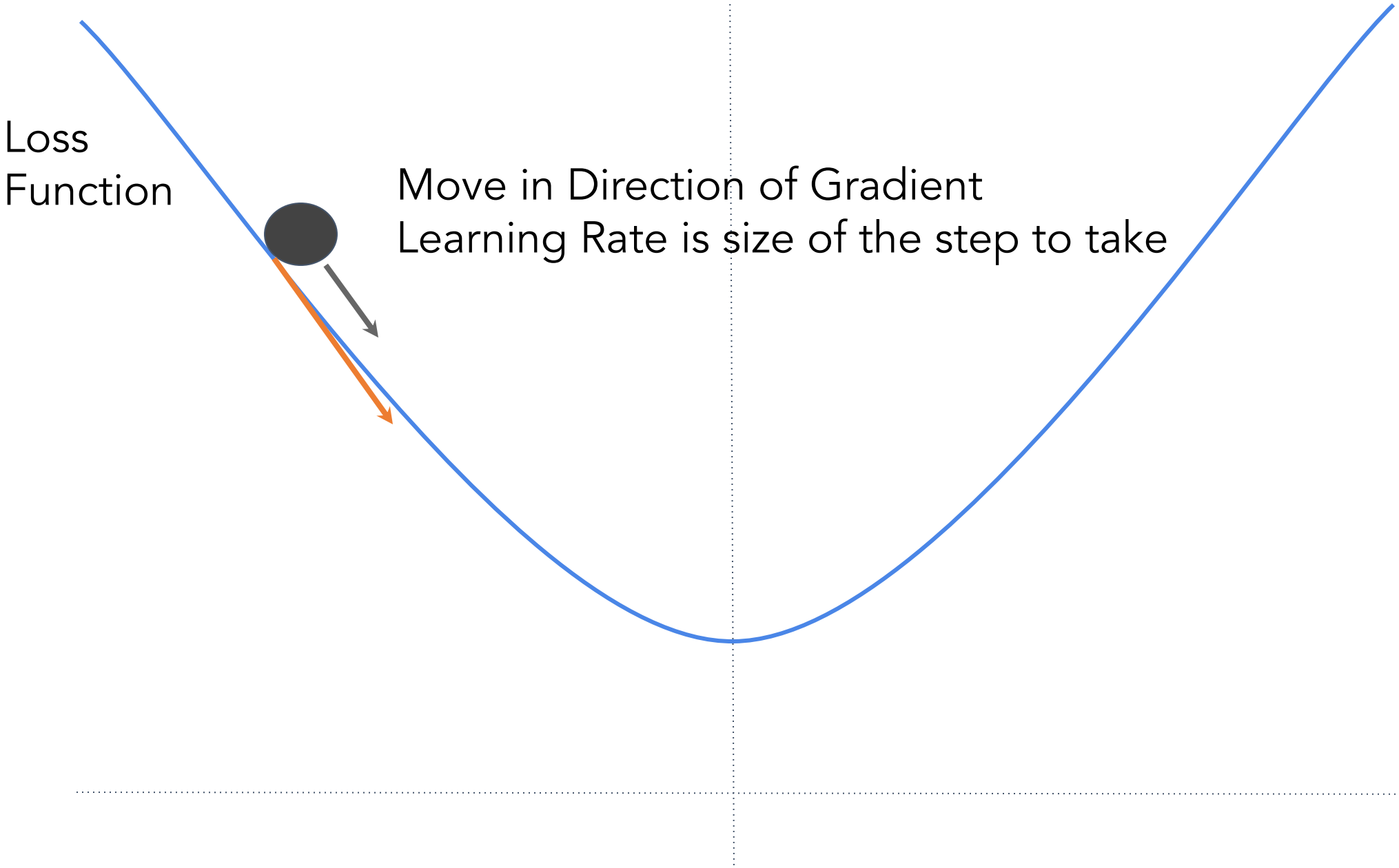
Trial and error approach

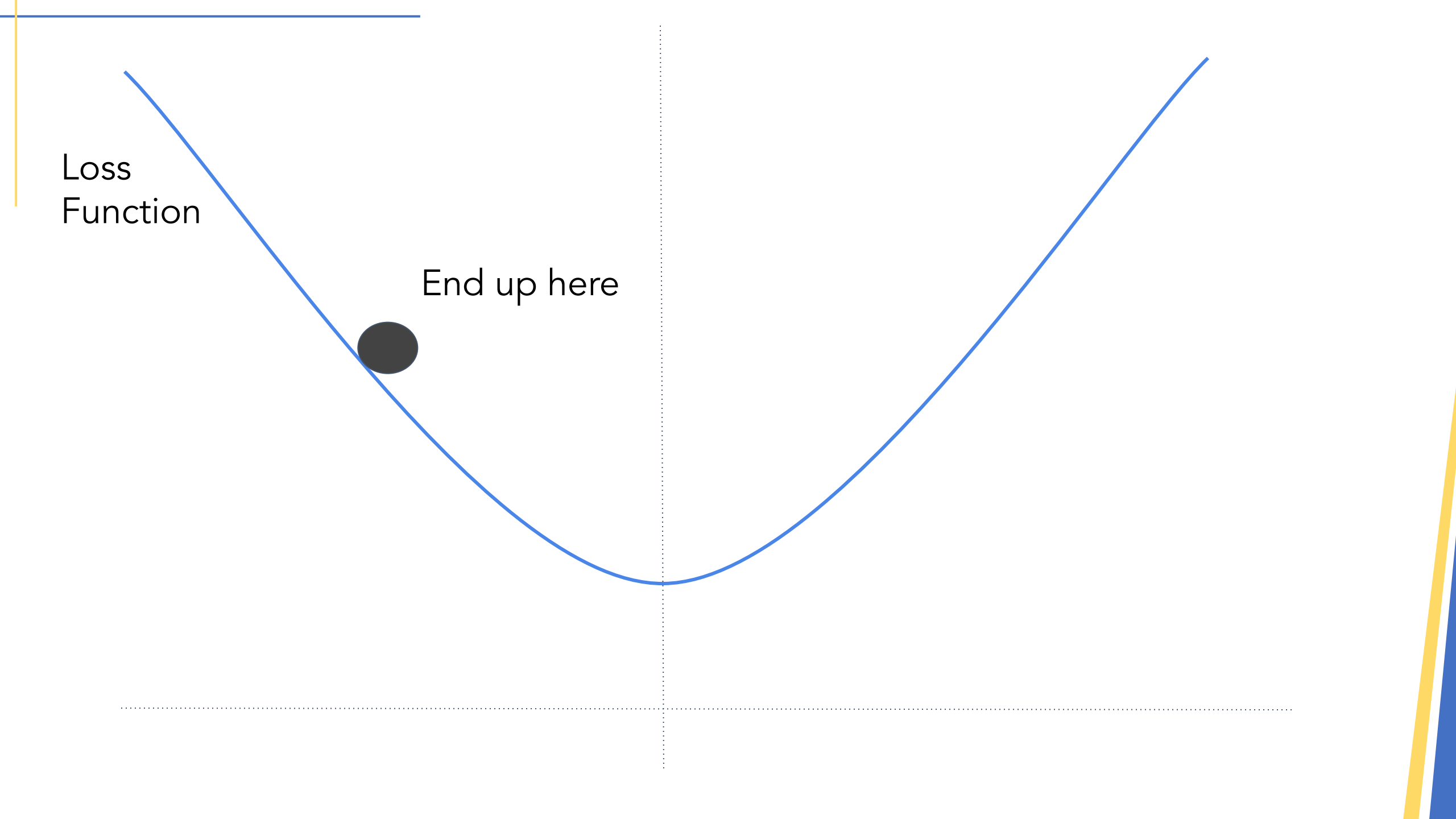

Loss
Function
Parameter

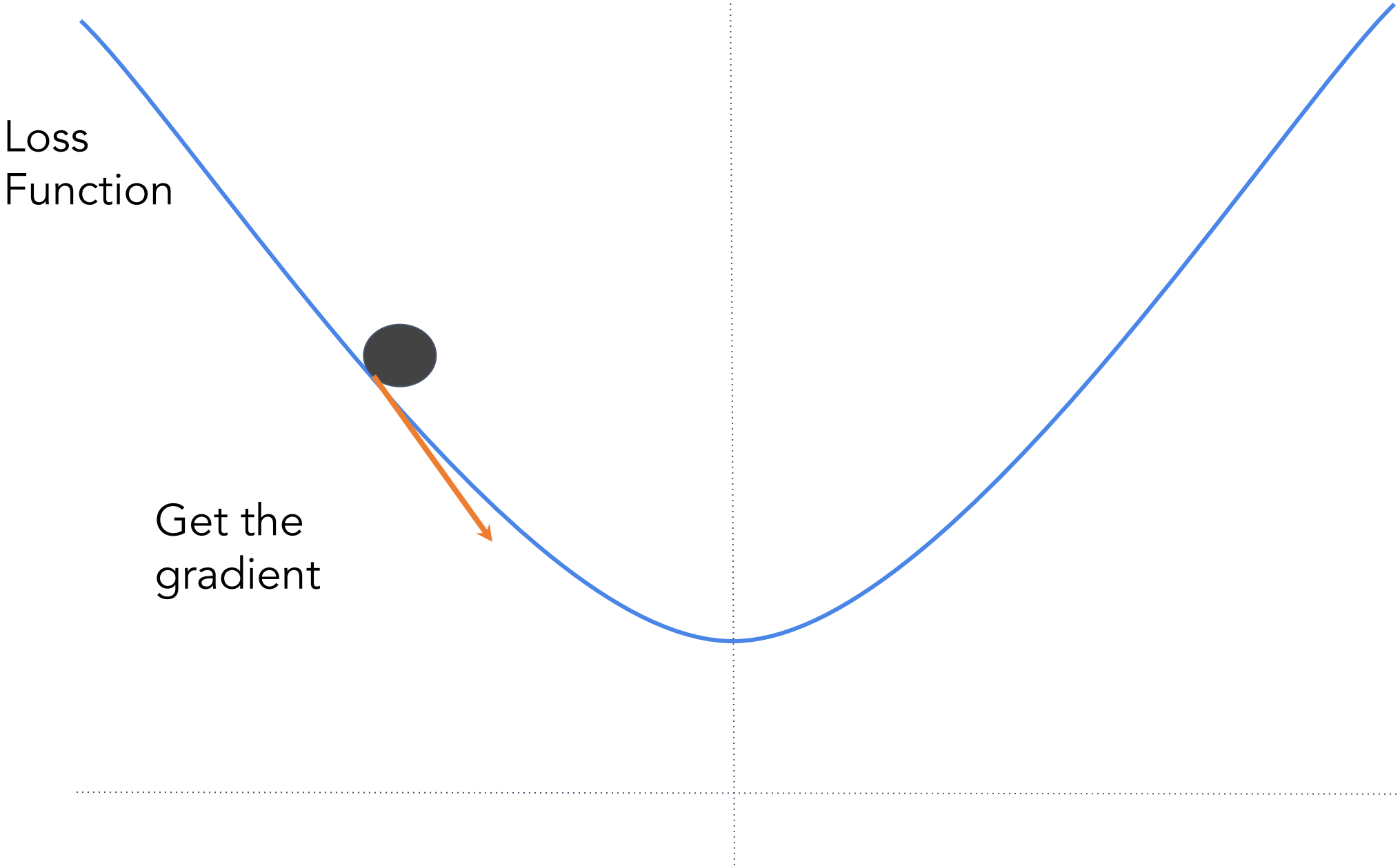

Loss
Function
Minimum of
Loss Function

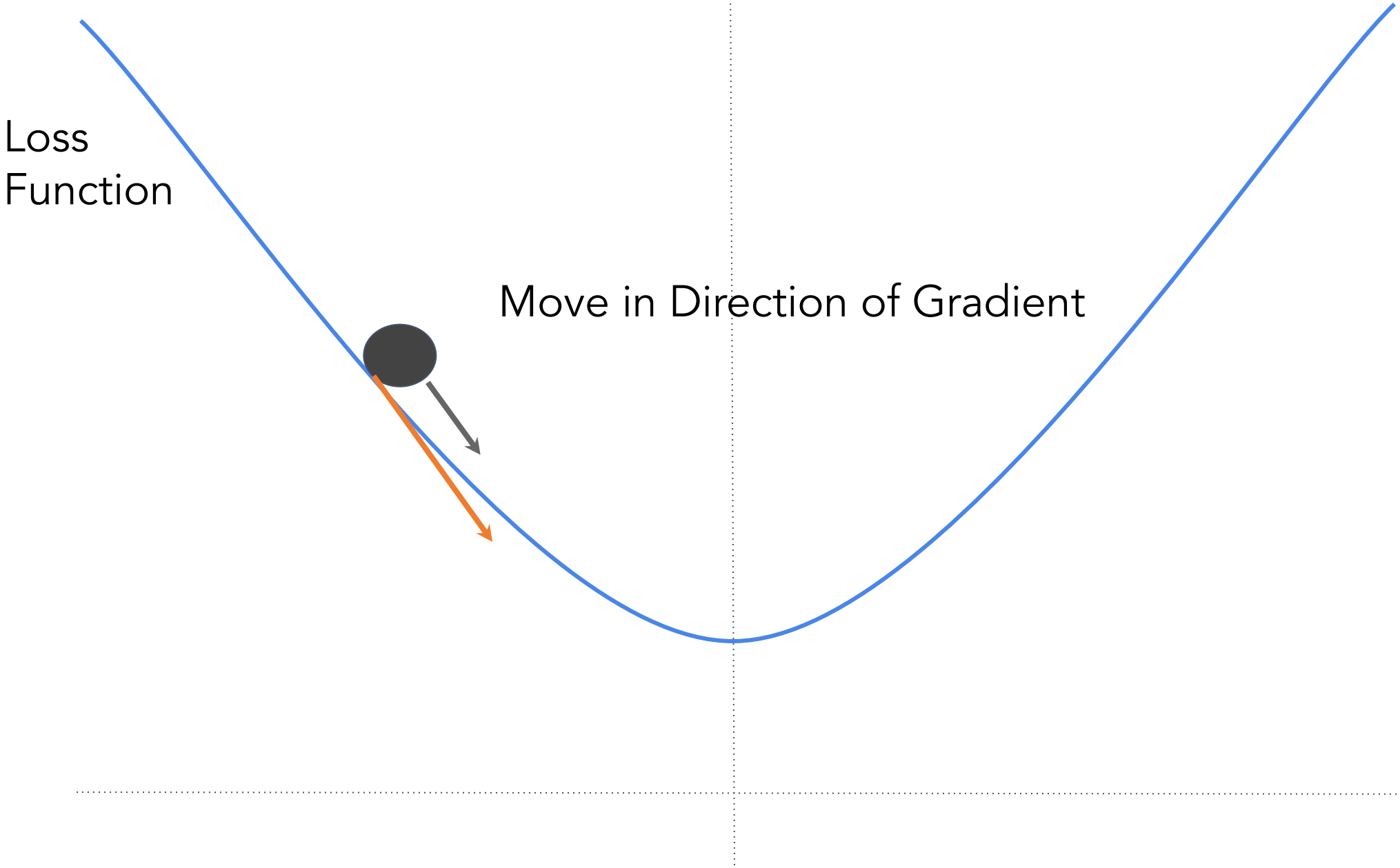

Loss
Function

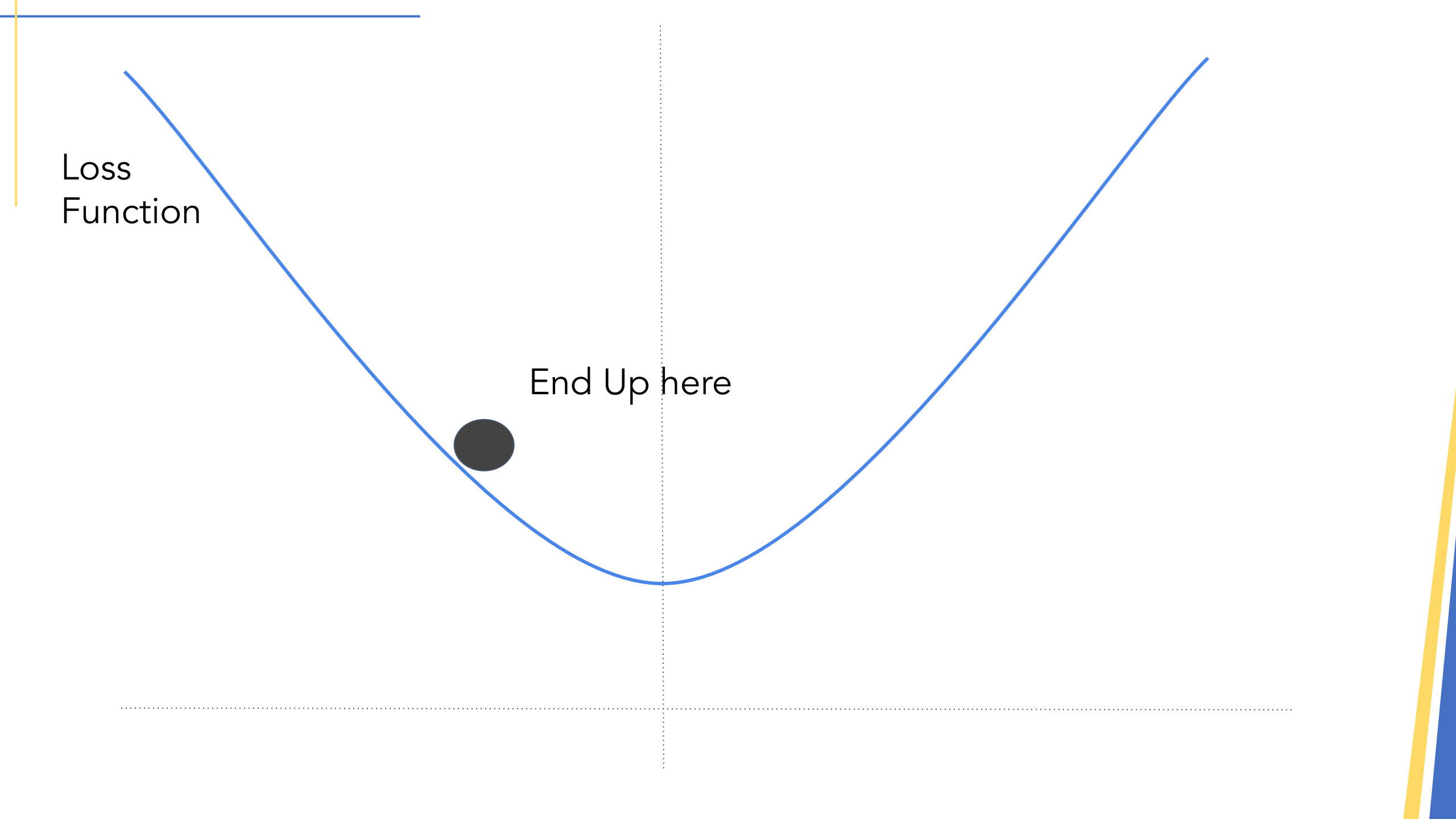

Loss
Function
Gradient of
value


Loss
Function
Move in Direction of Gradient
Learning Rate is size of the step to take

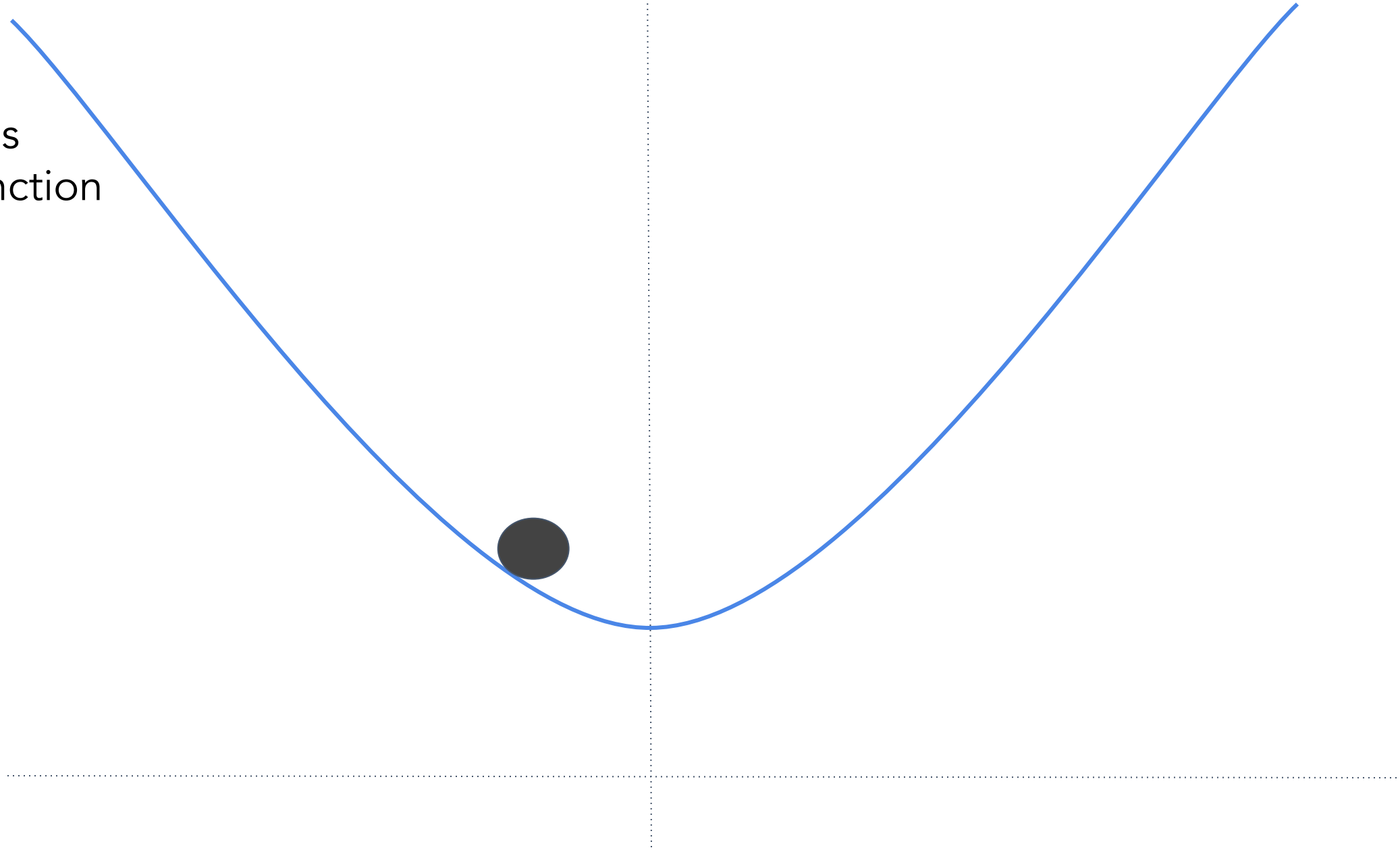

Loss
Function
End up here


Loss
Function
Get the
gradient

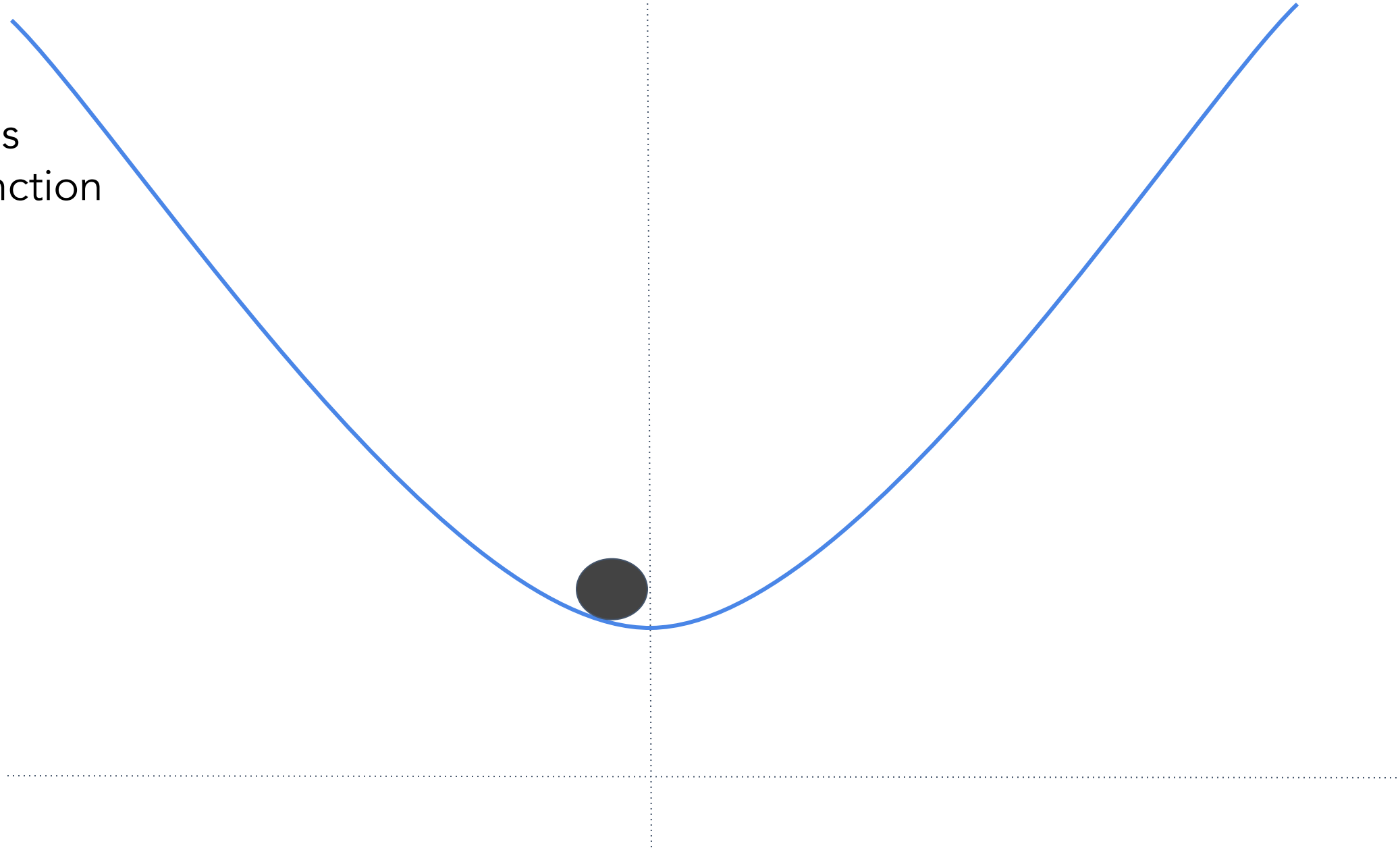

Loss
Function
Move in Direction of Gradient


Loss
Function
End Up here

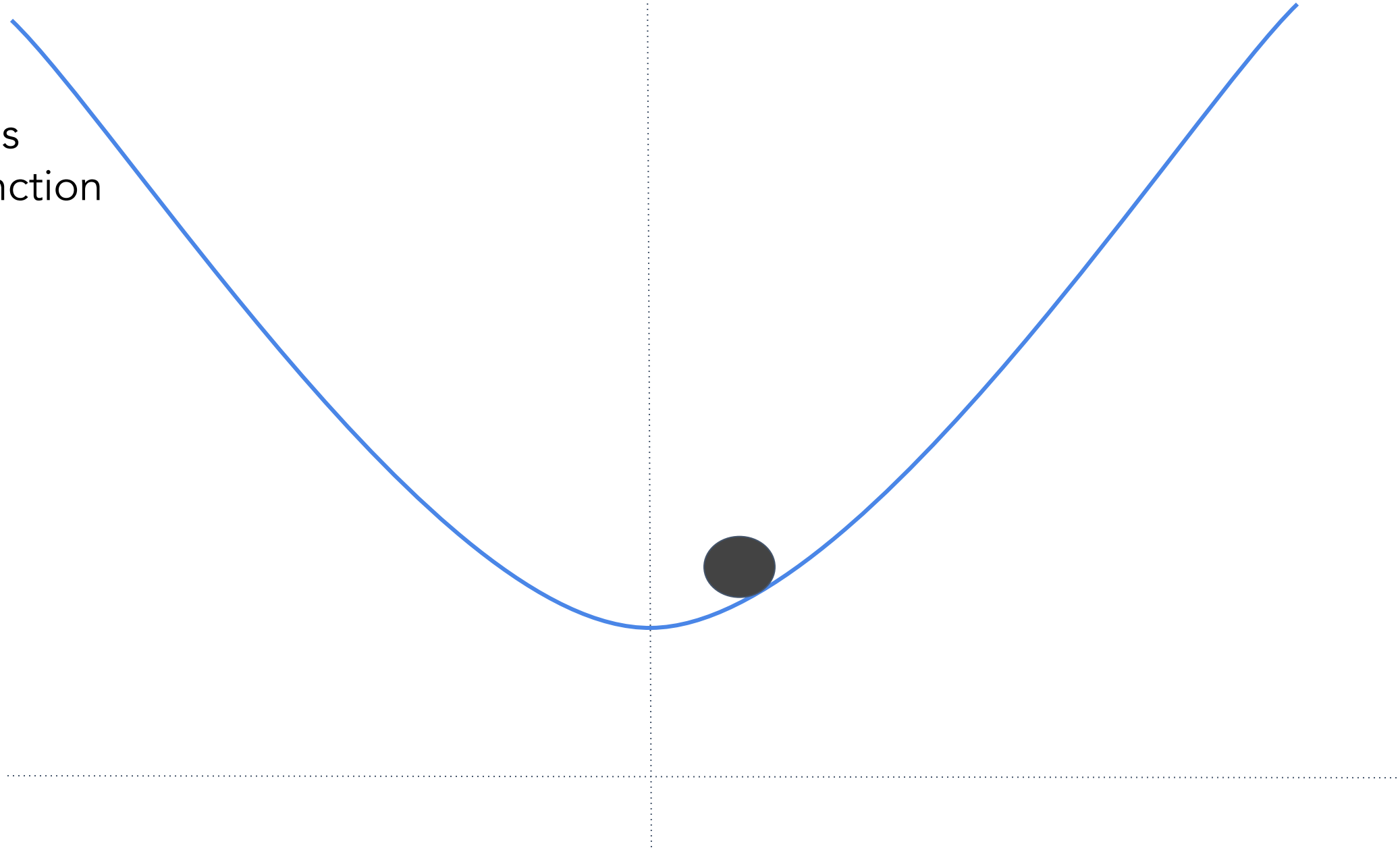
Loss
Function



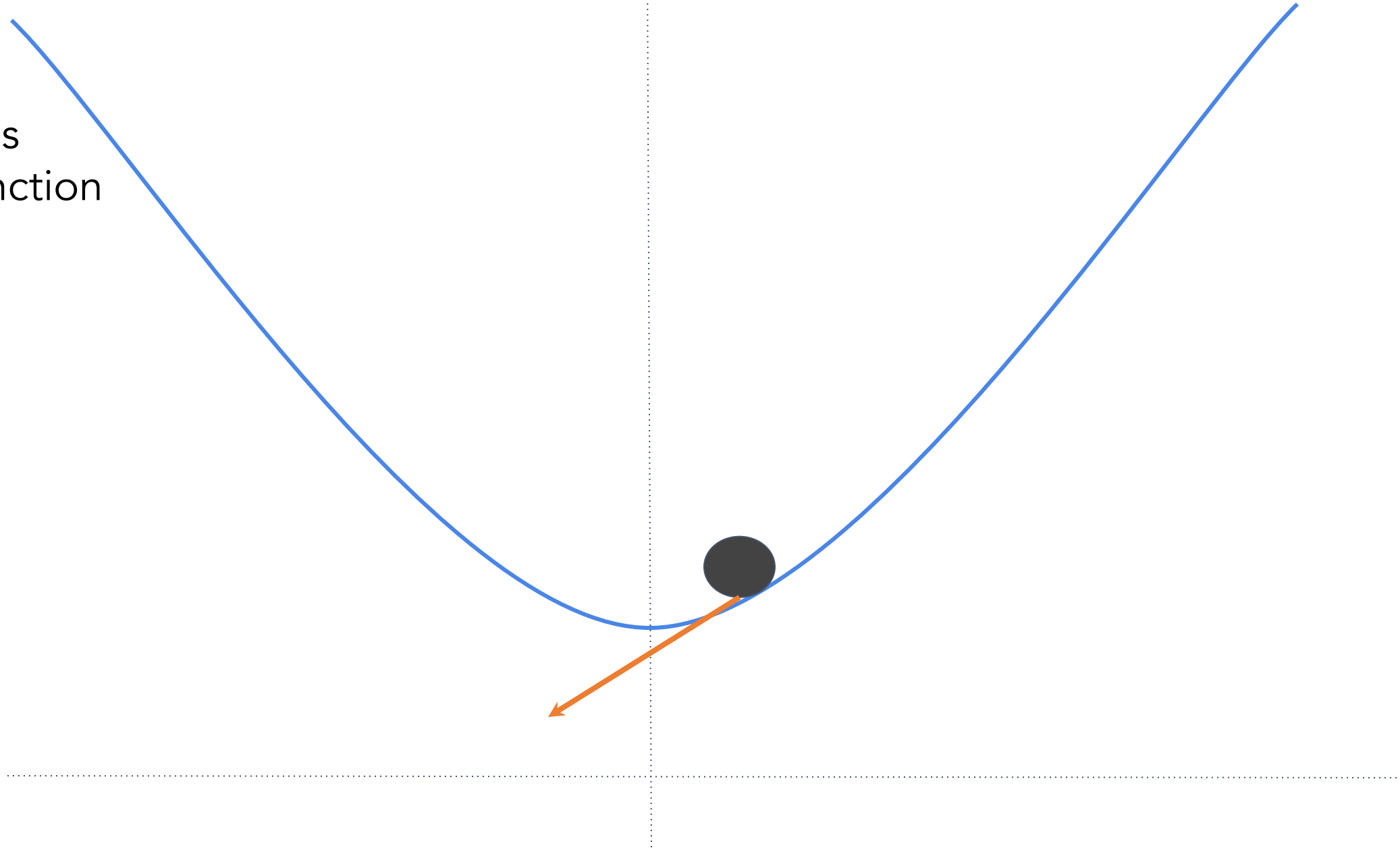
Loss
Function



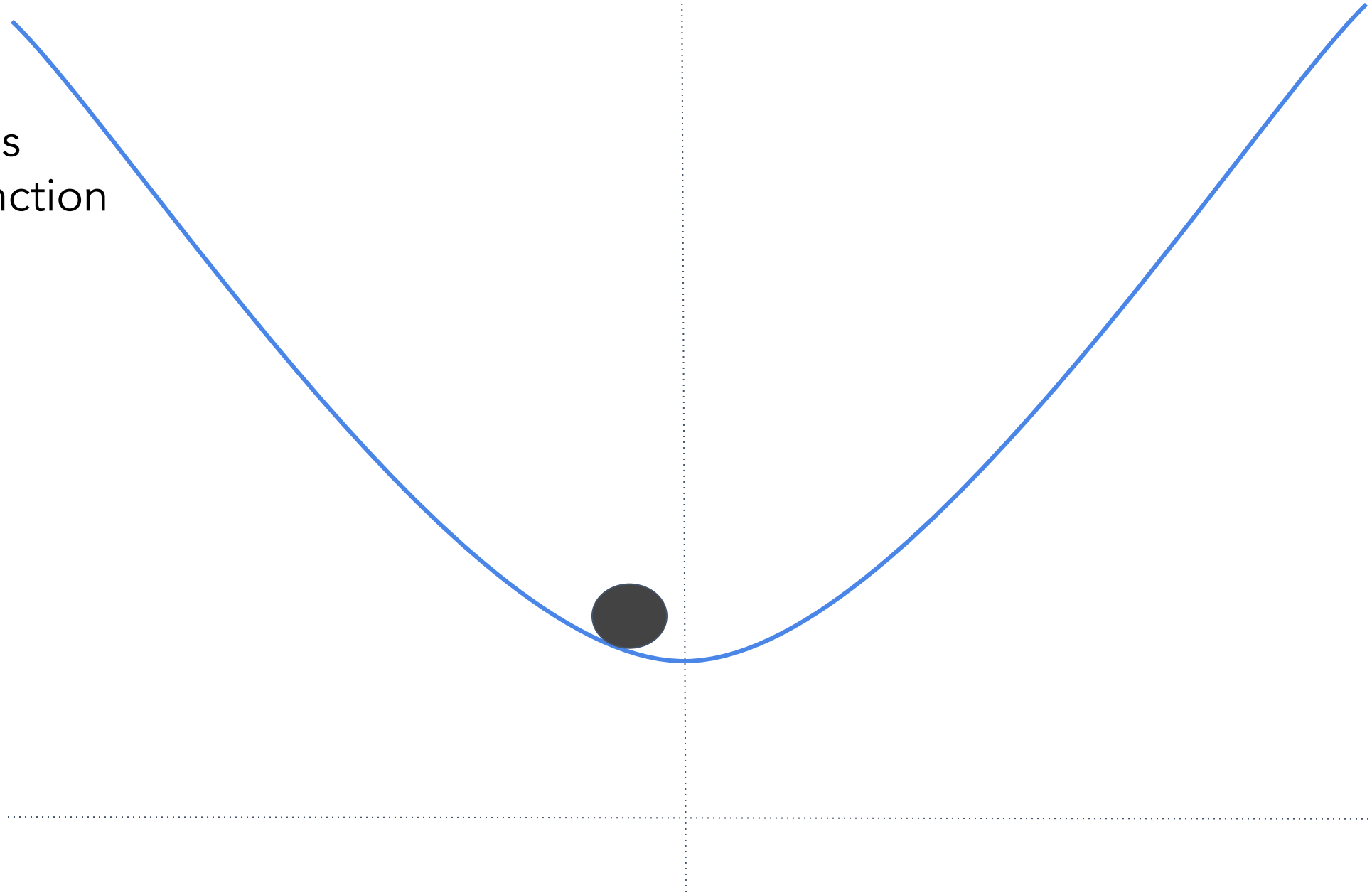
Loss
Function



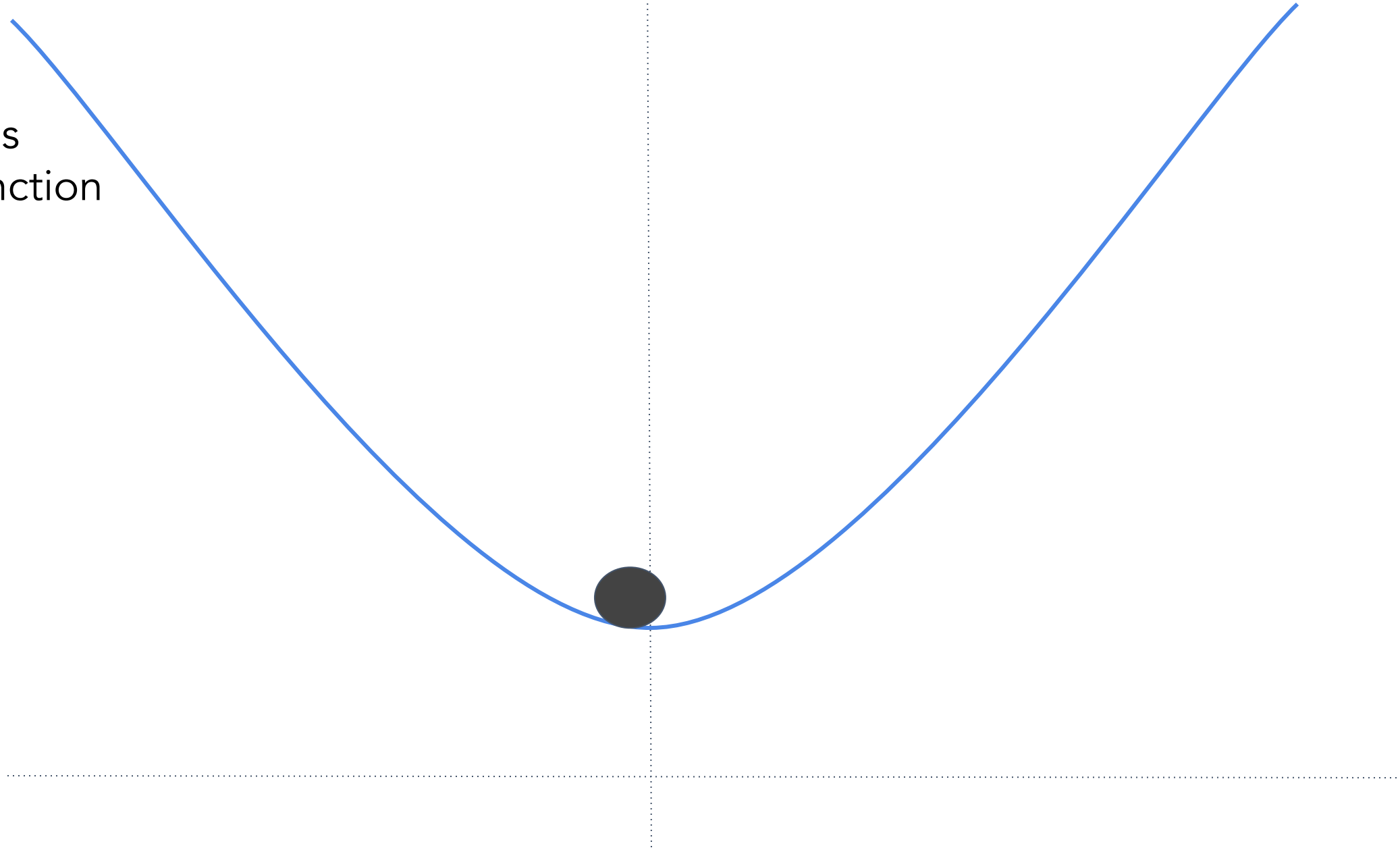
Loss
Function



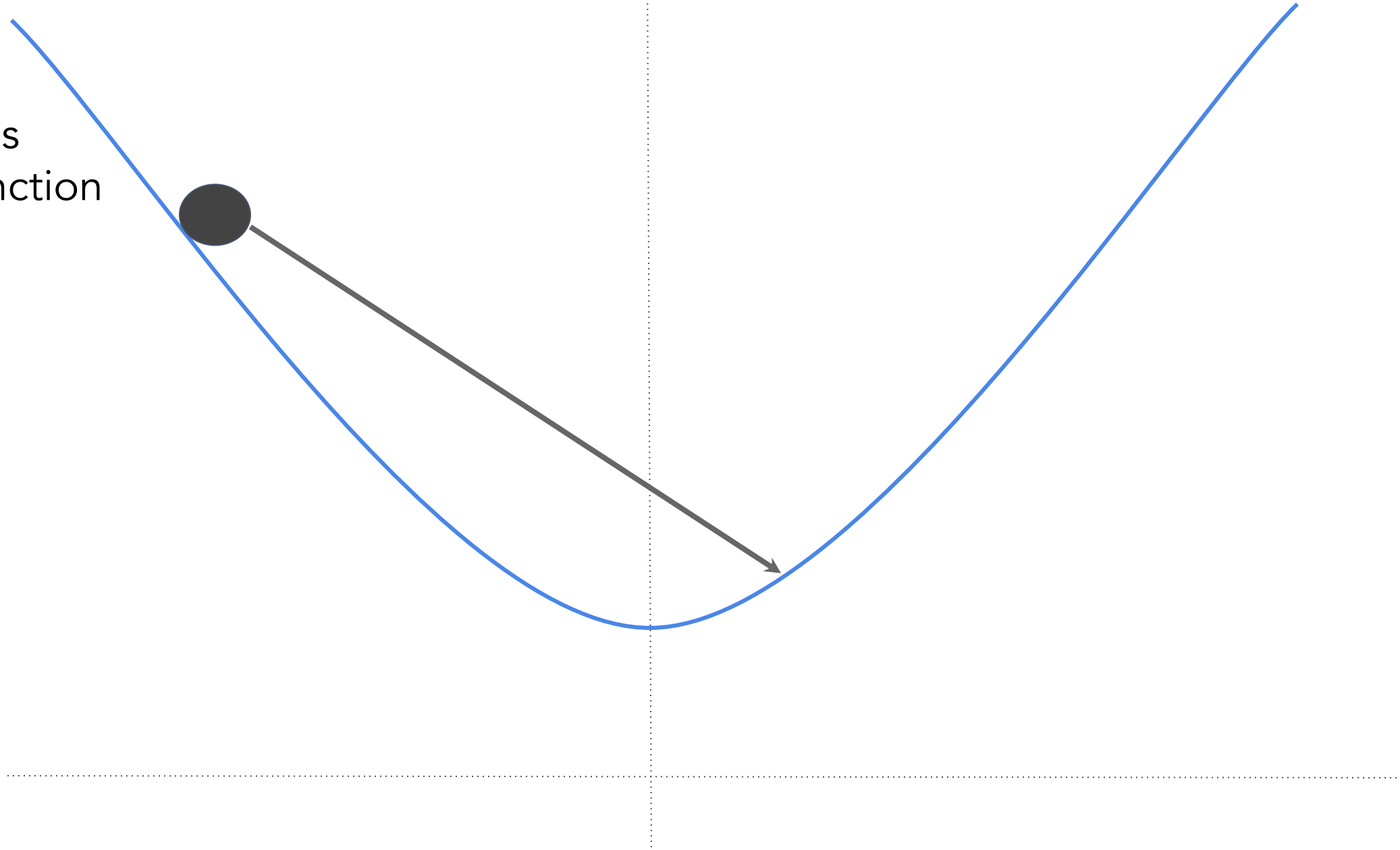
Loss
Function



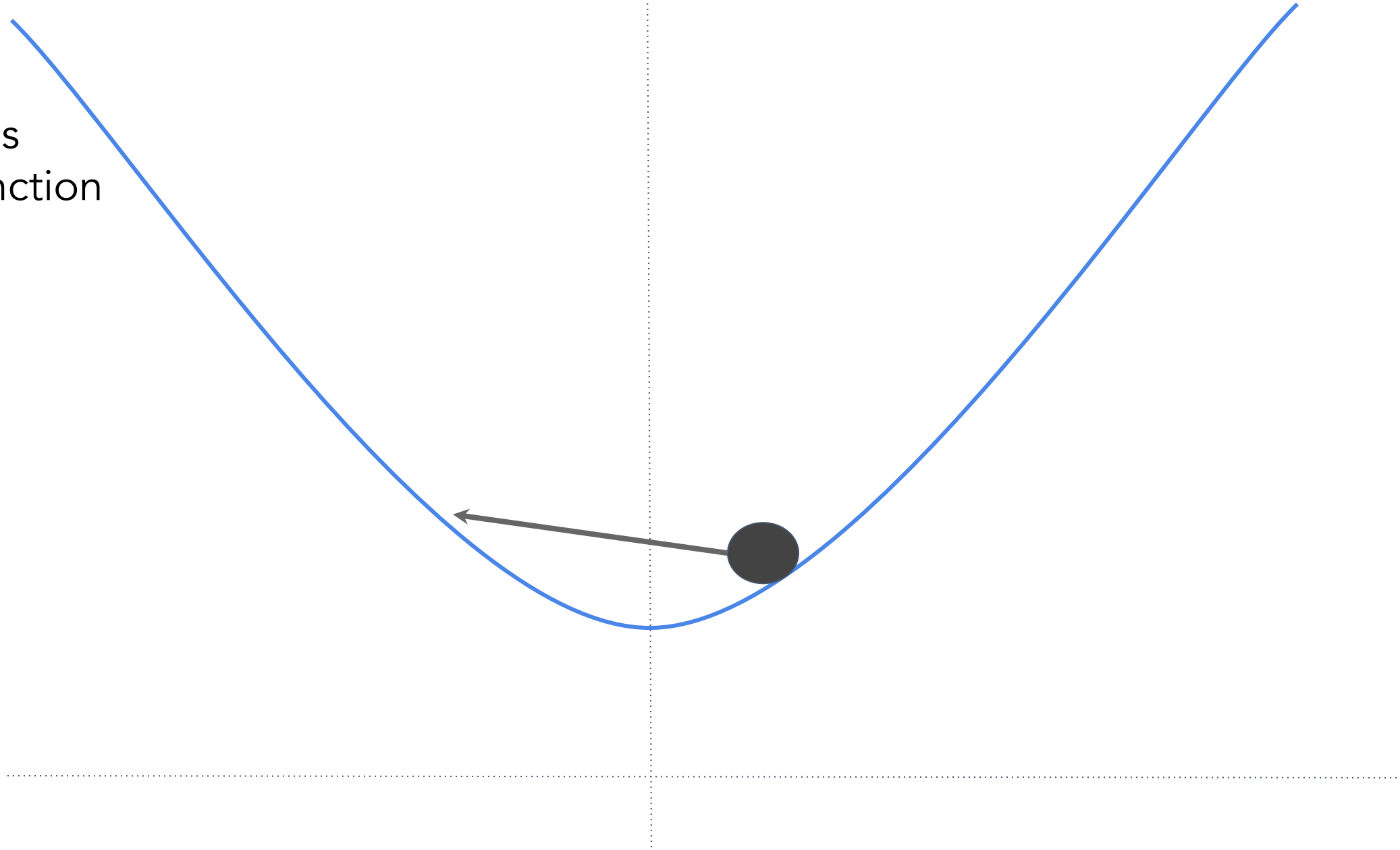
Loss
Function



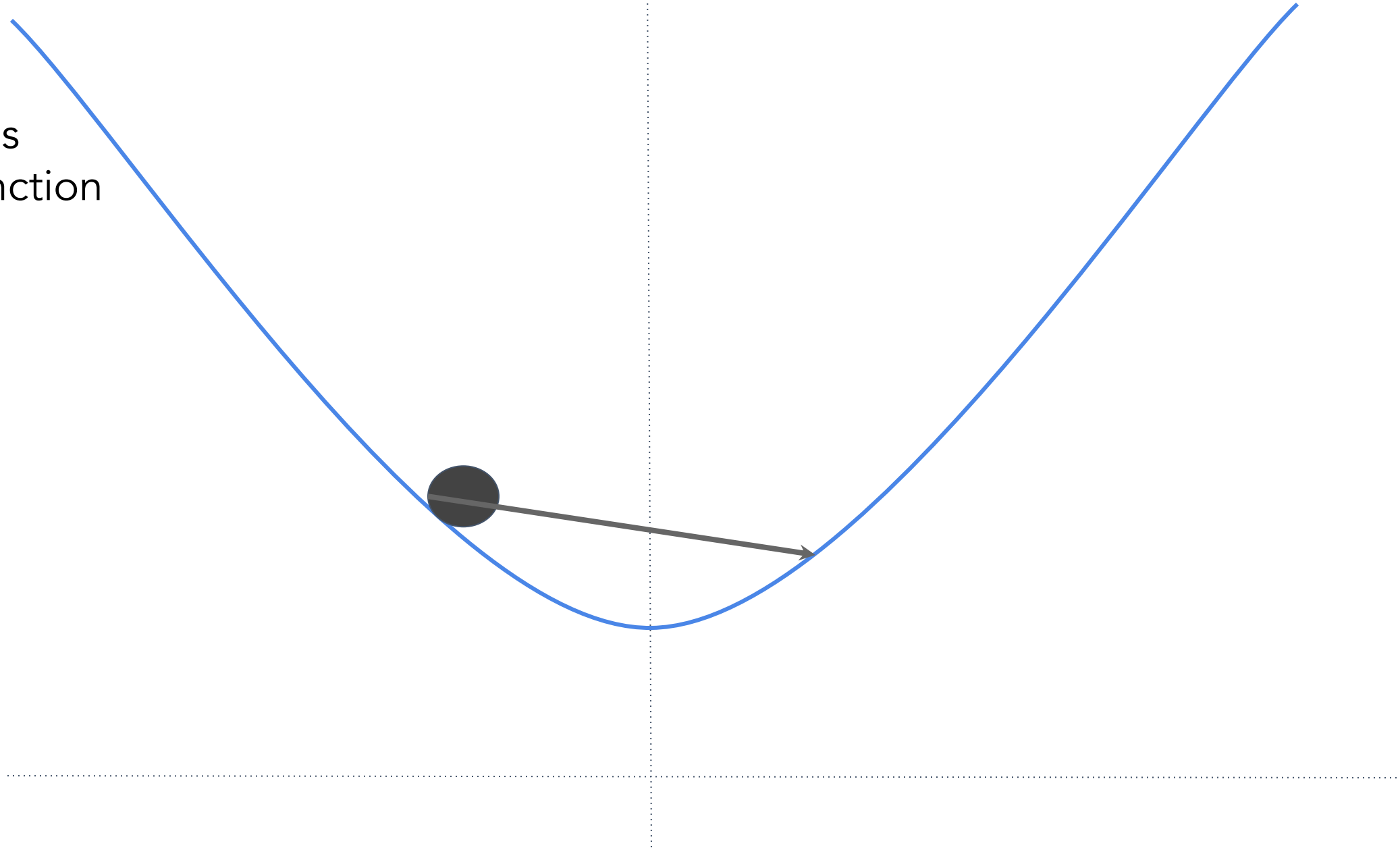
Loss Function

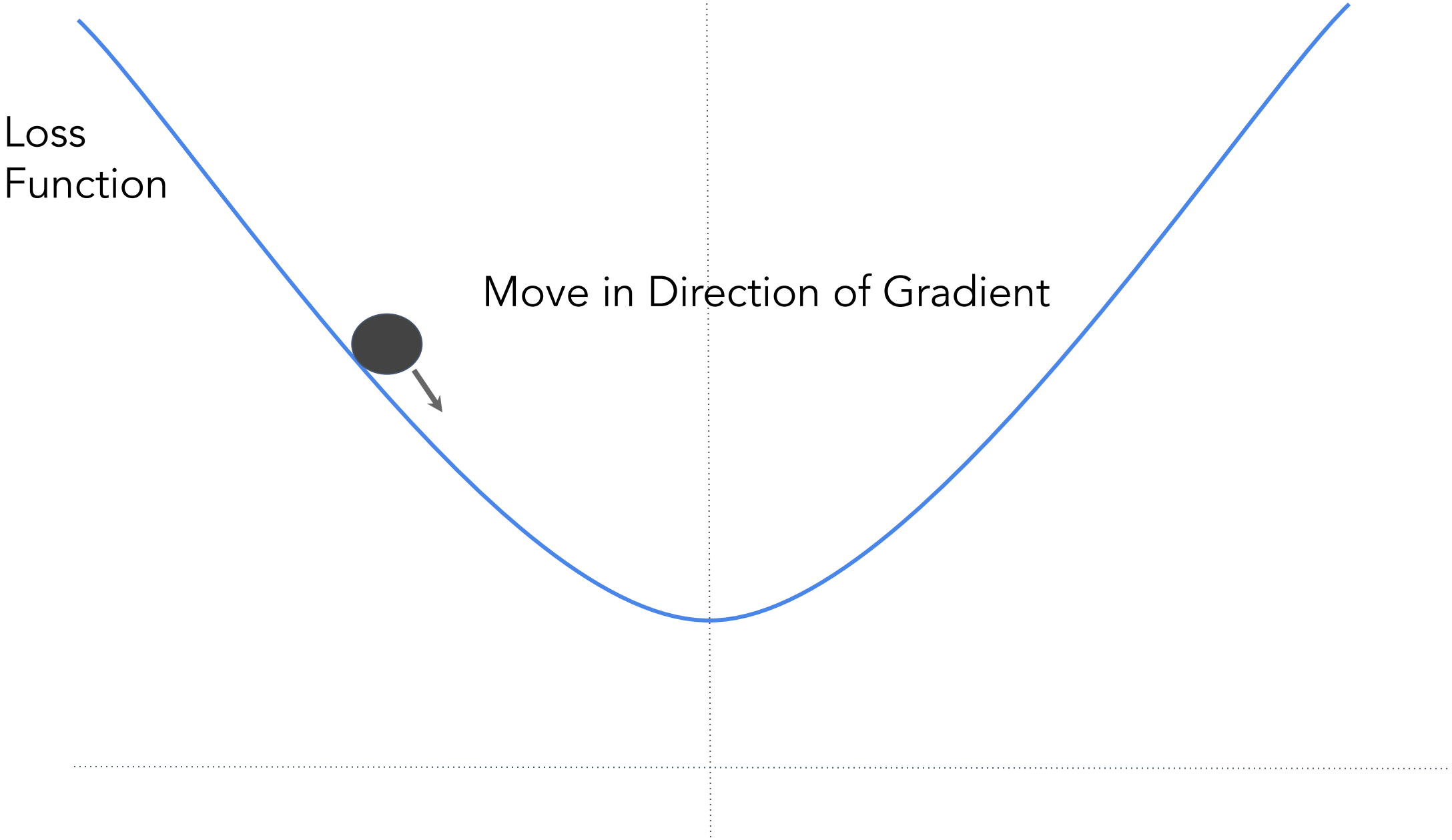


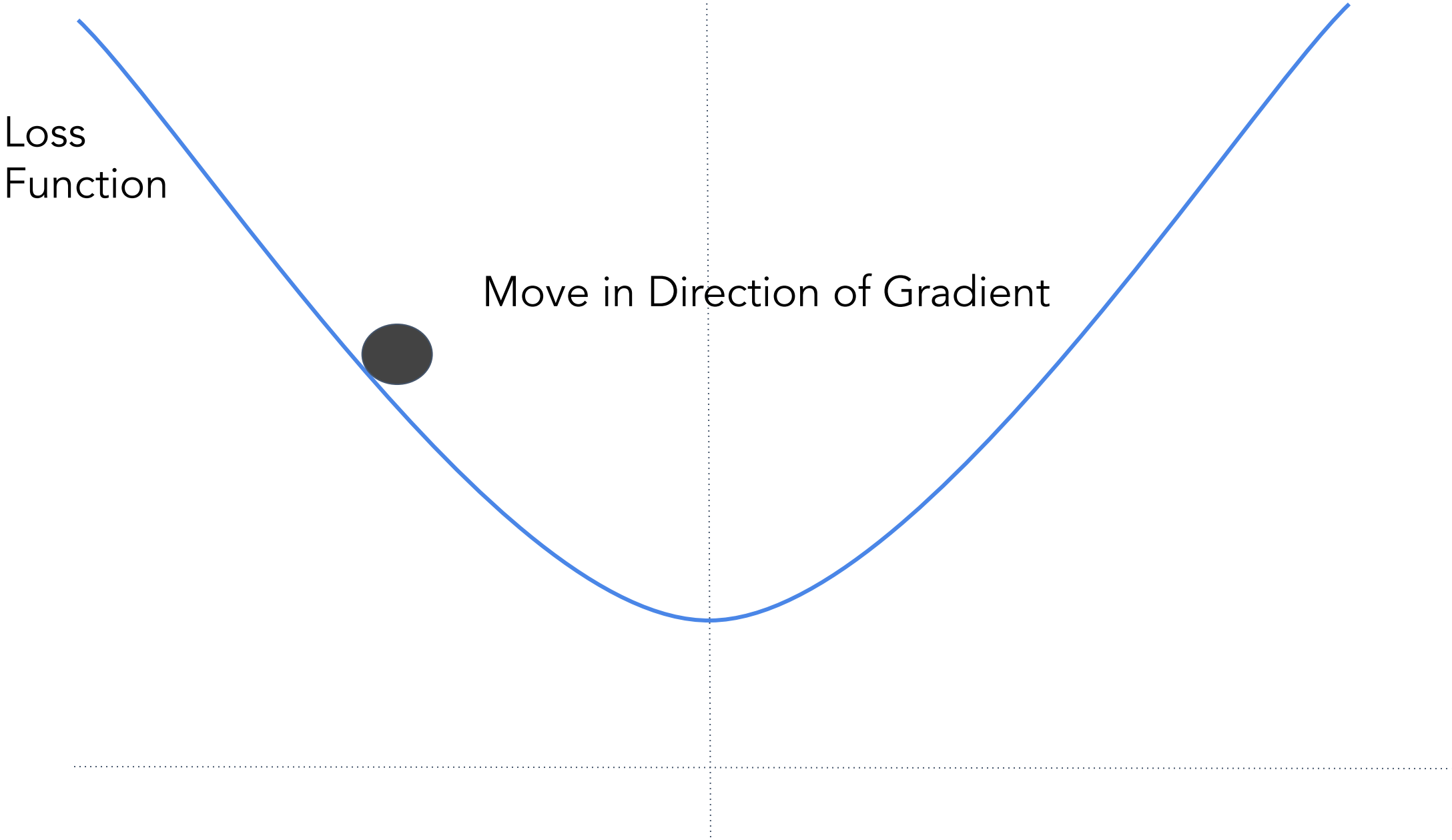
Loss
Function

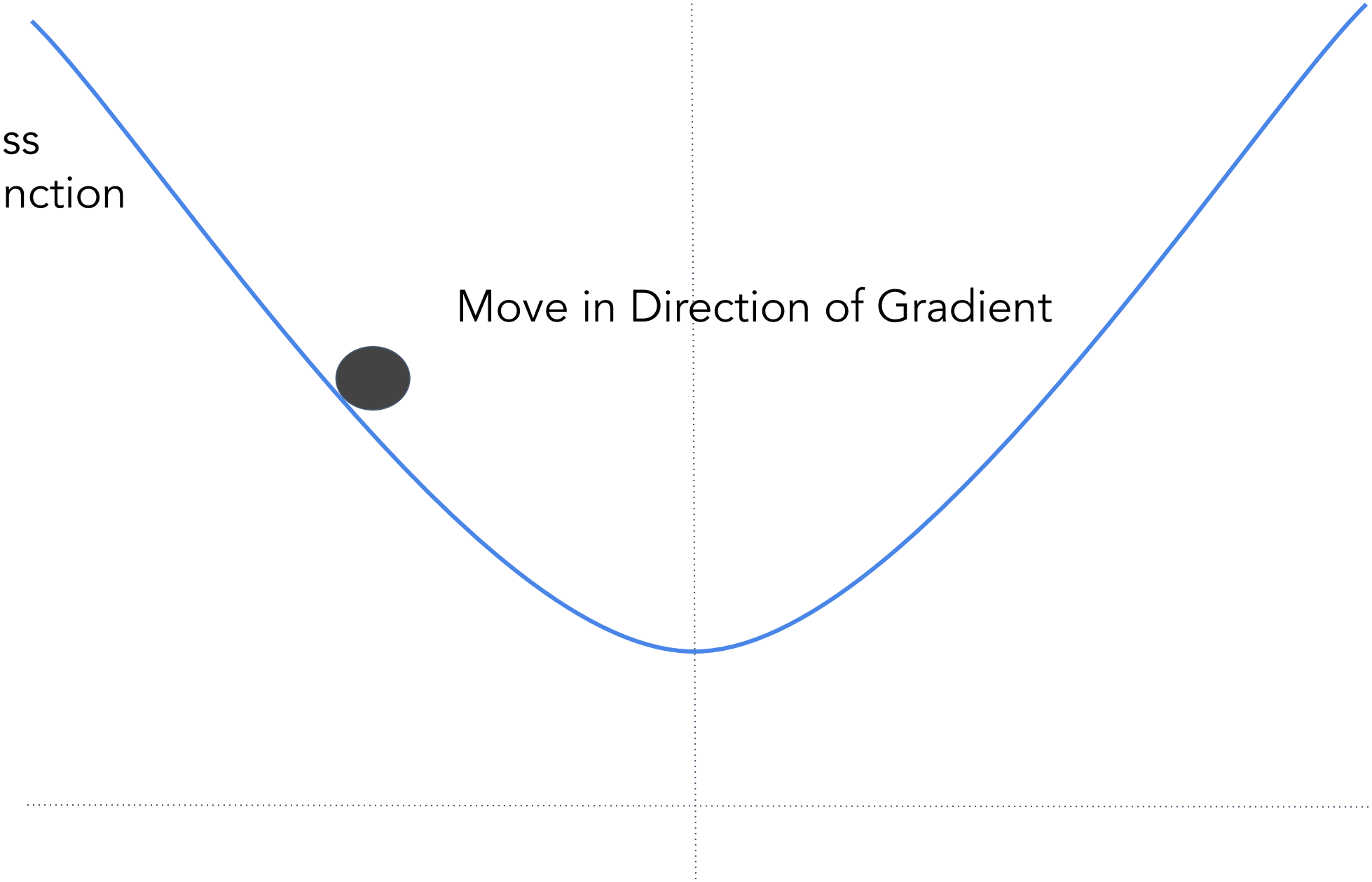


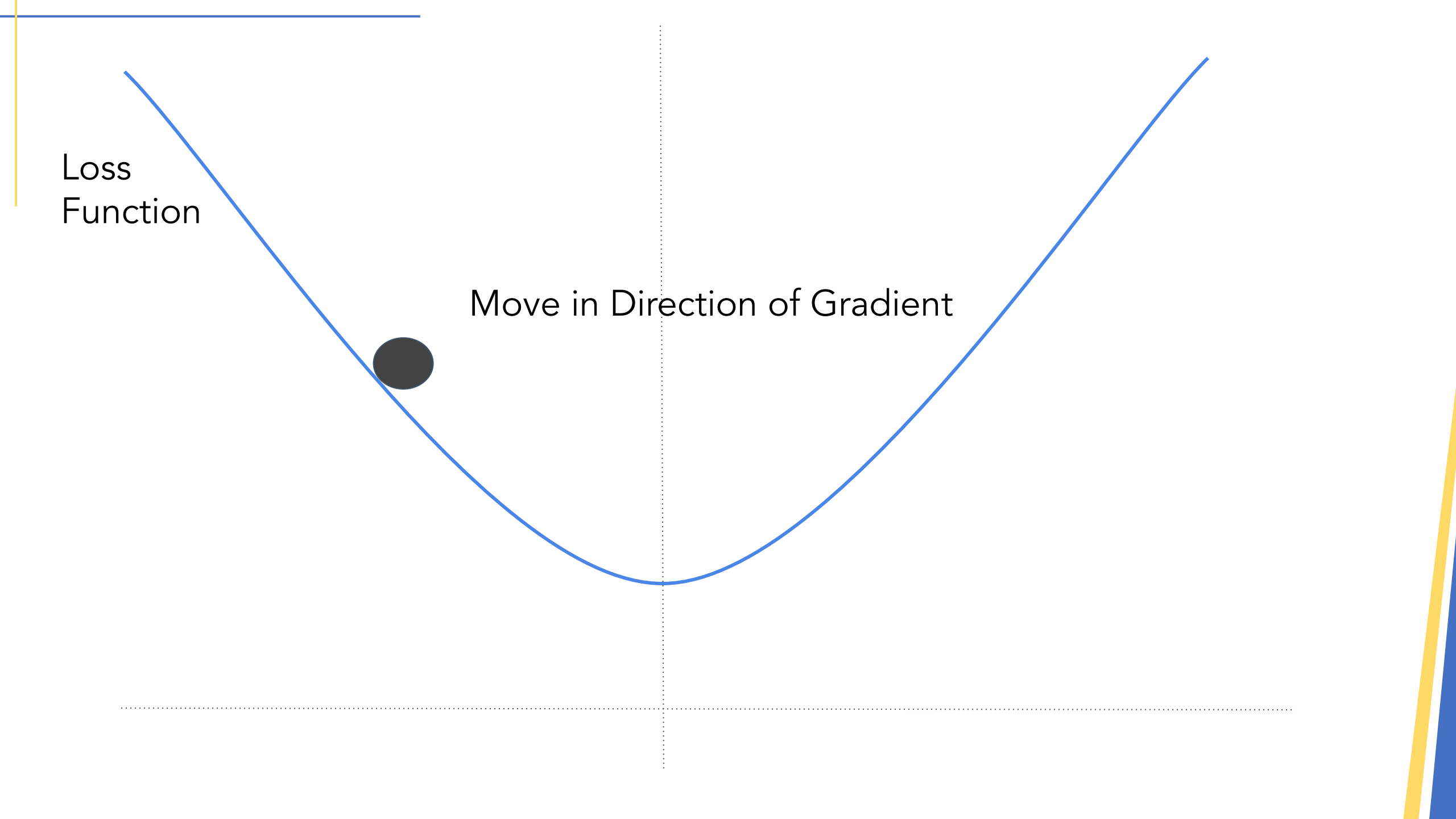
Loss
Function

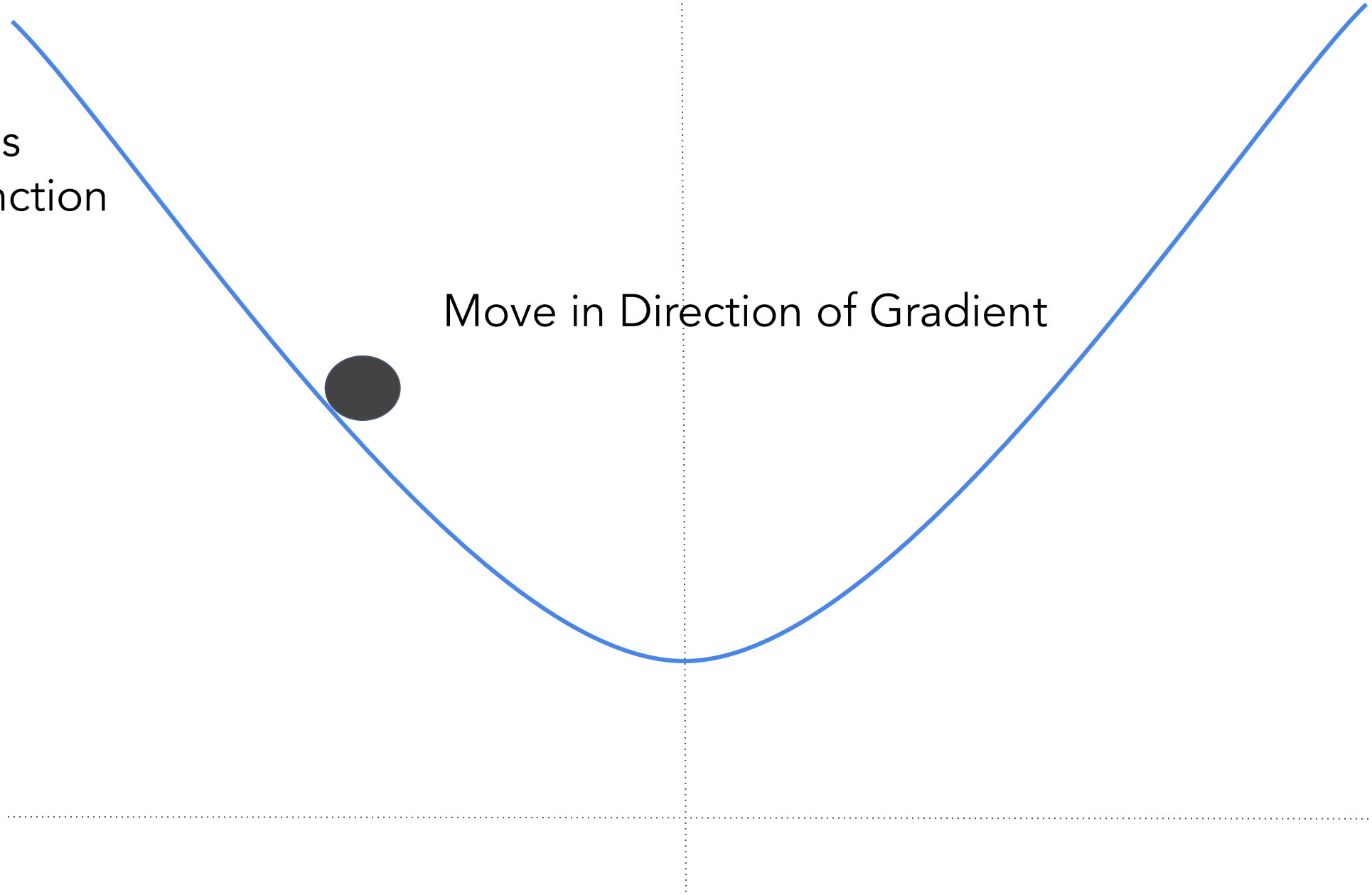



Loss
Function
Move in Direction of Gradient


Loss
Function
Move in Direction of Gradient

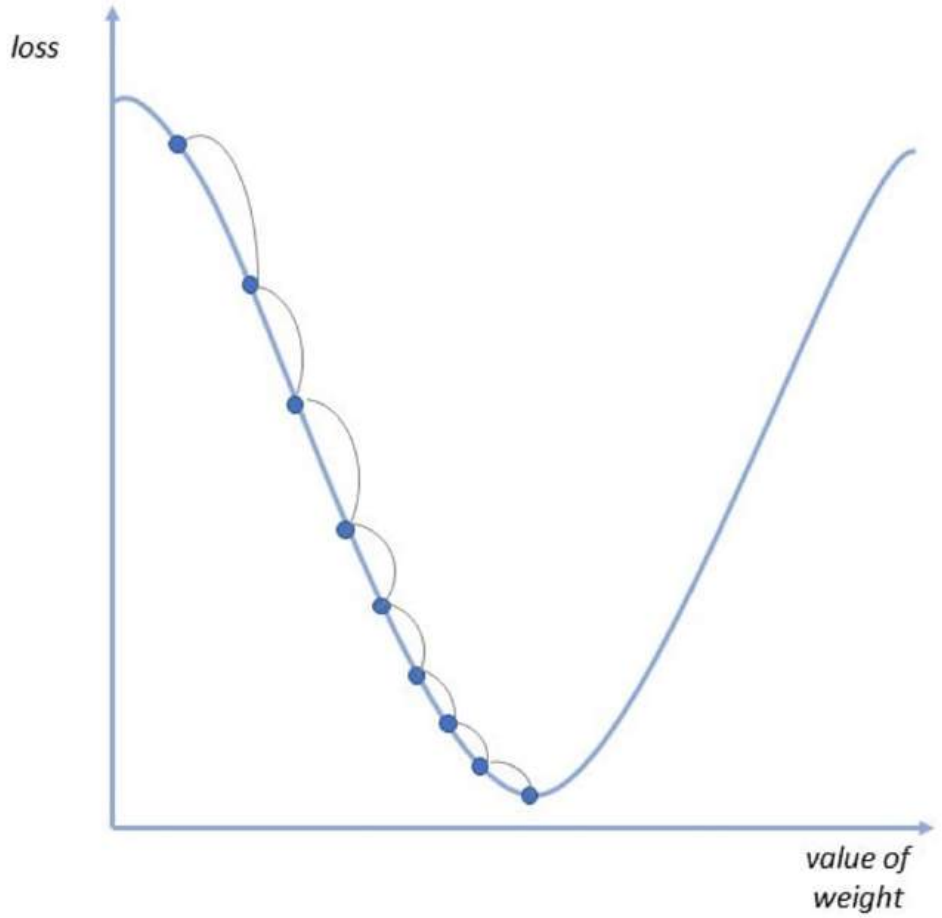

Loss
Function
Move in Direction of Gradient


Loss
Function
Move in Direction of Gradient

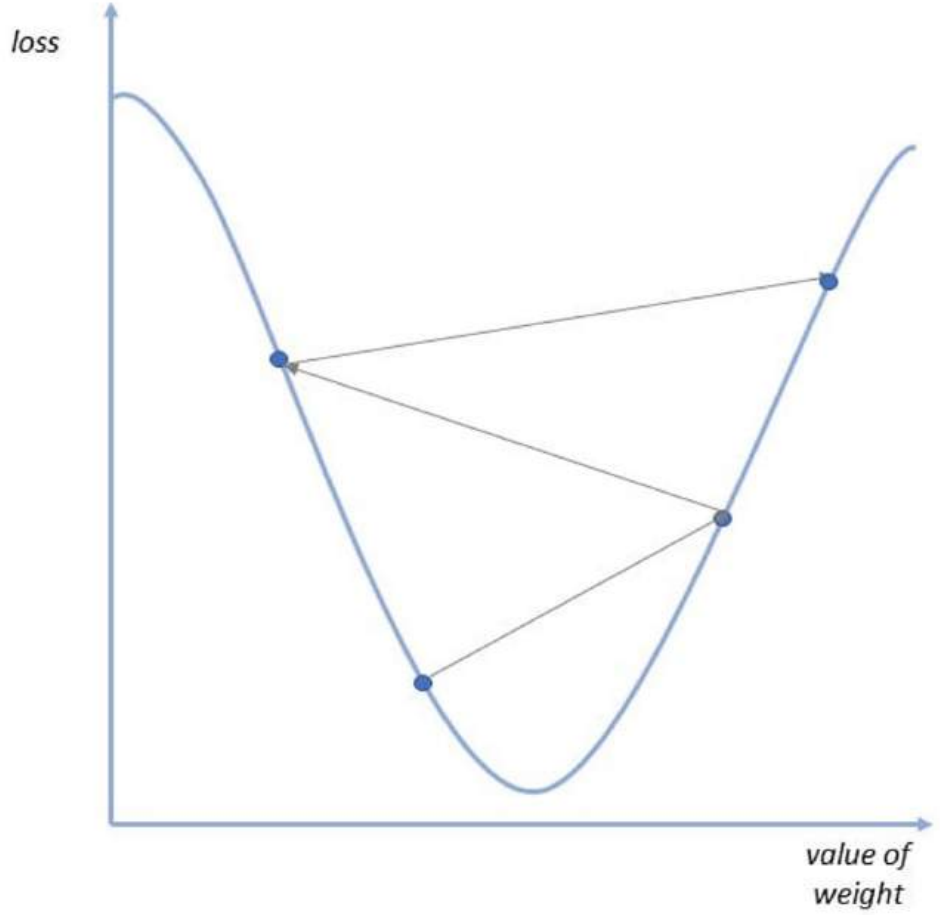

Loss
Function
Move in Direction of Gradient

It is important to choose the correct Learning Rate (size of the step)

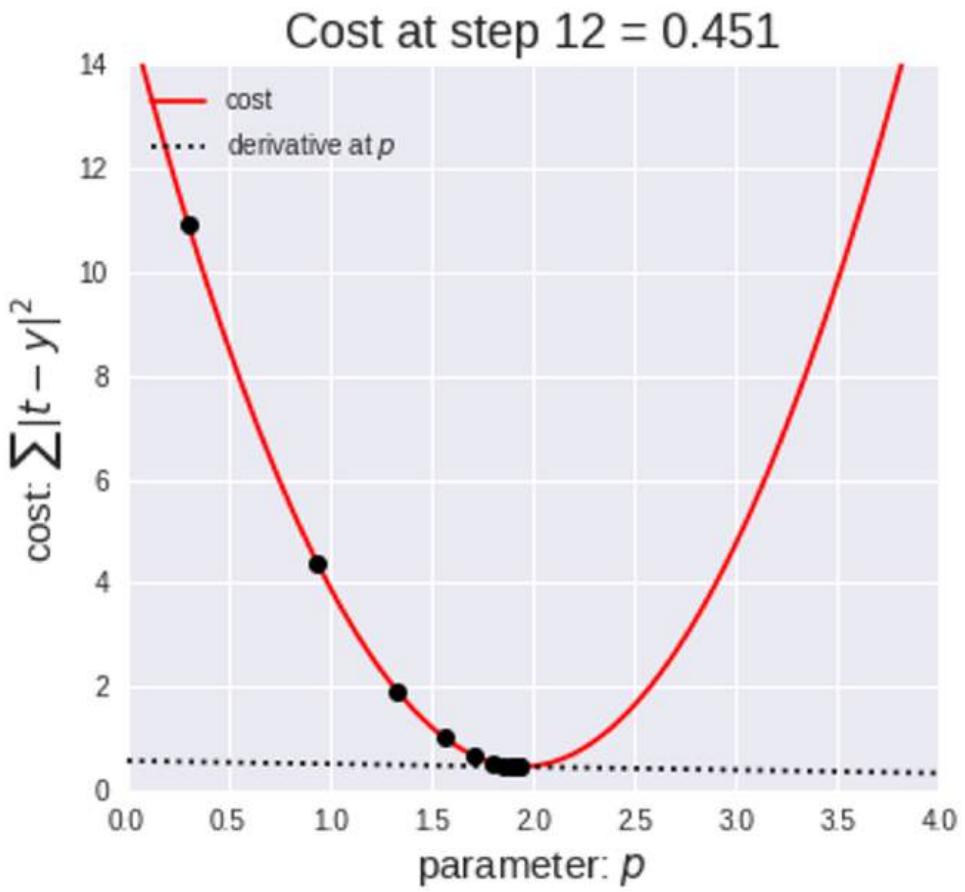
If the Learning Rate is too small it may take a long time to reach the minimum

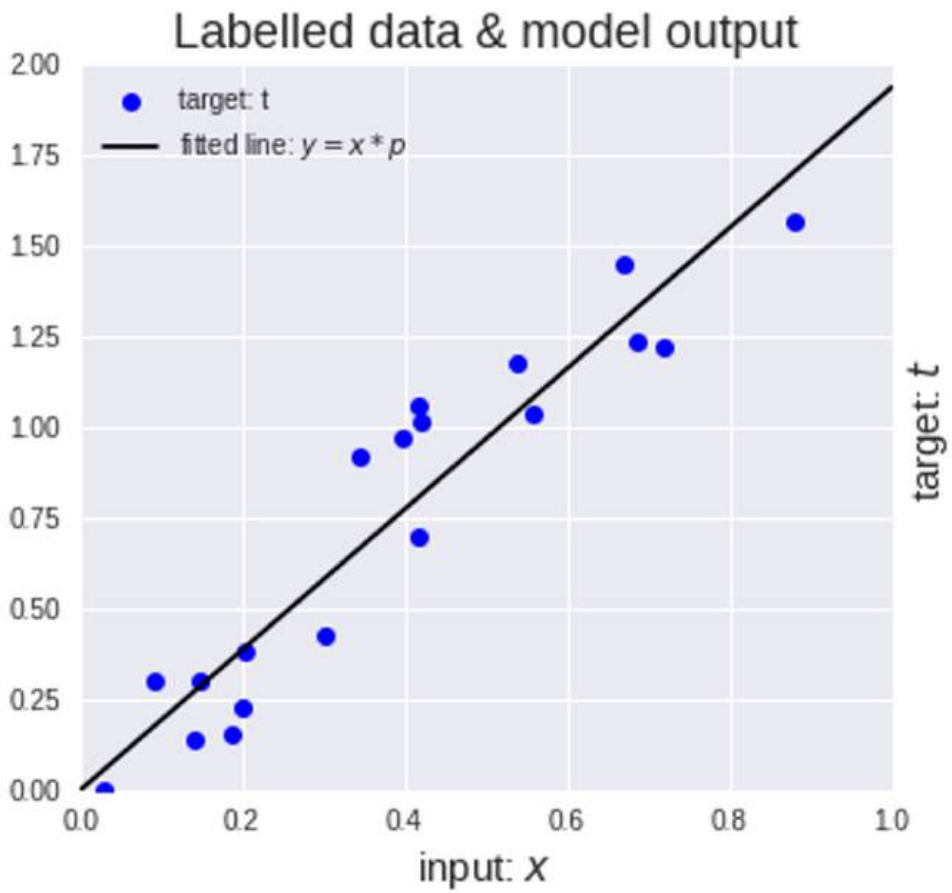


If the Learning Rate is too large we may never reach the minimum



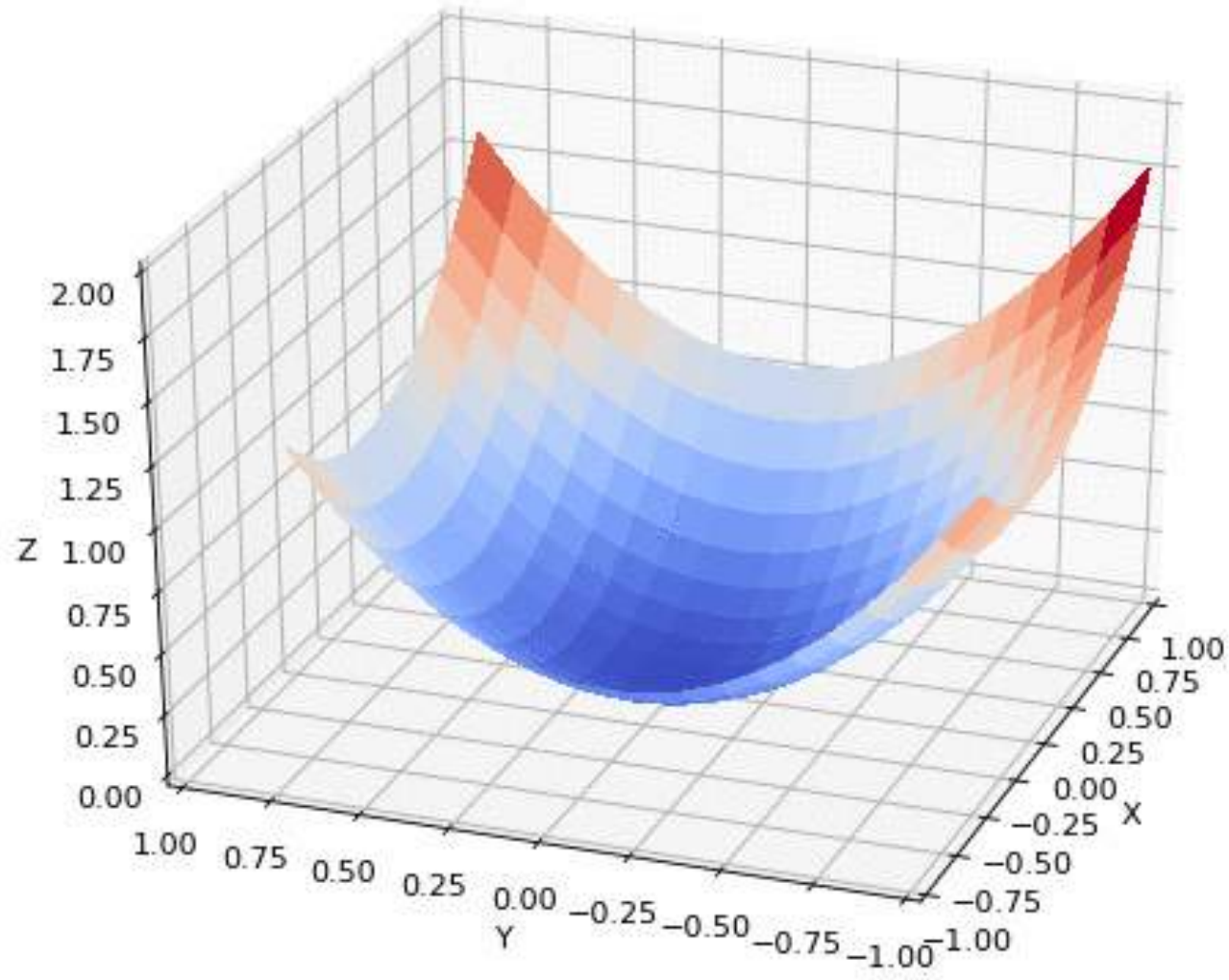
Gradient Descent algorithm





Open in Colab

Gradient Descent for Two Parameters

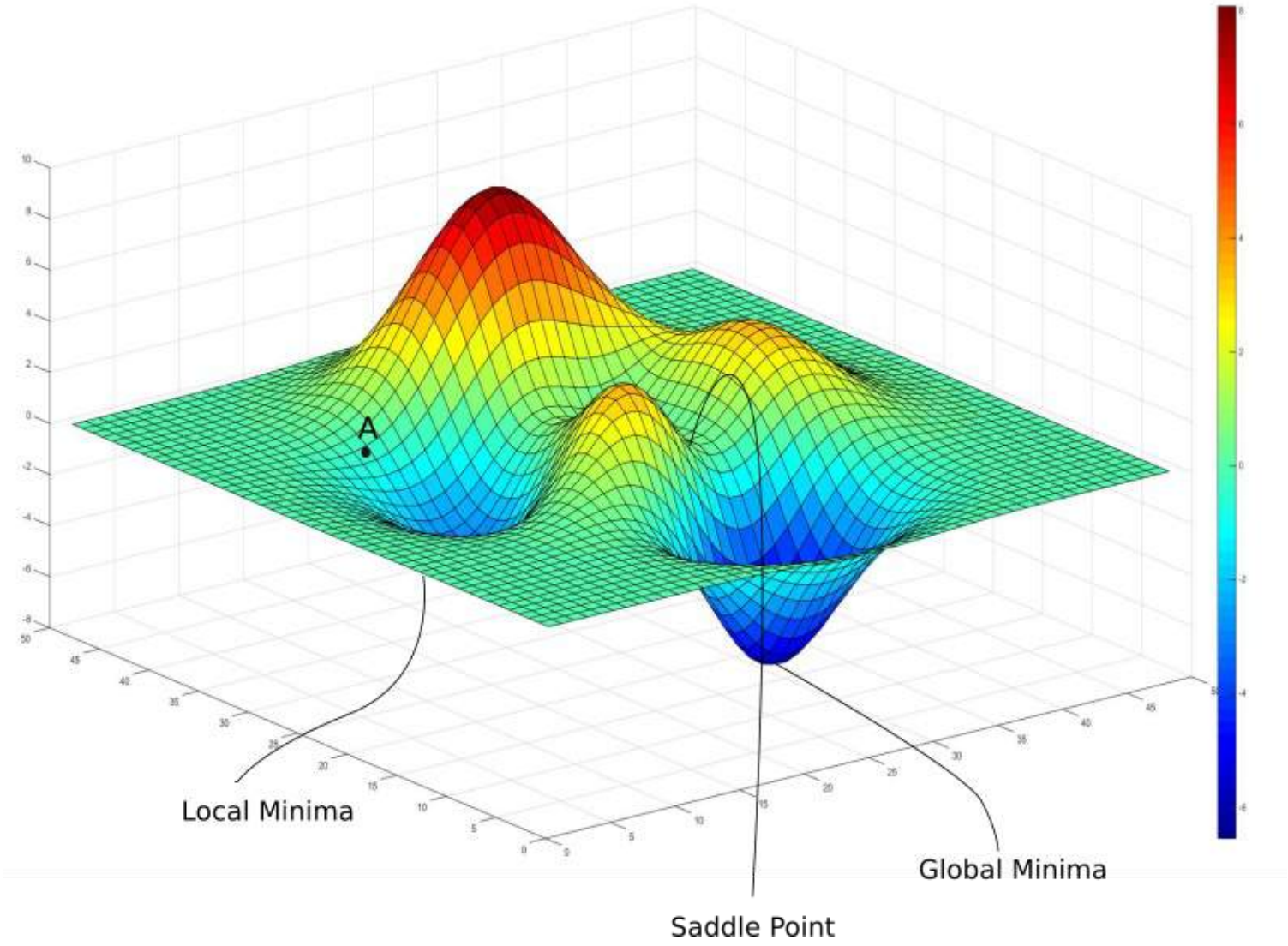


A single minima

Global minima

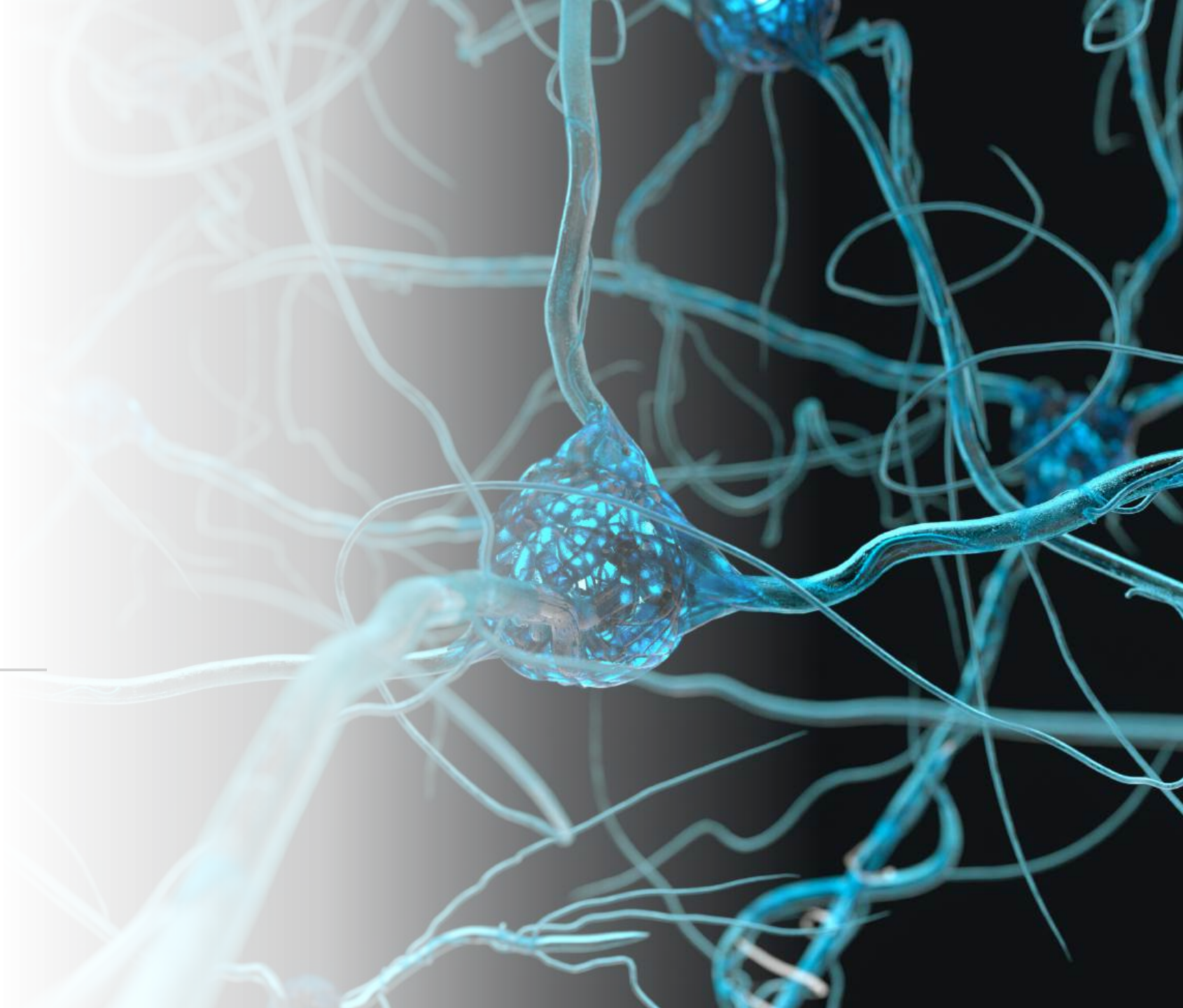
https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

Gradient Descent for Two Parameters

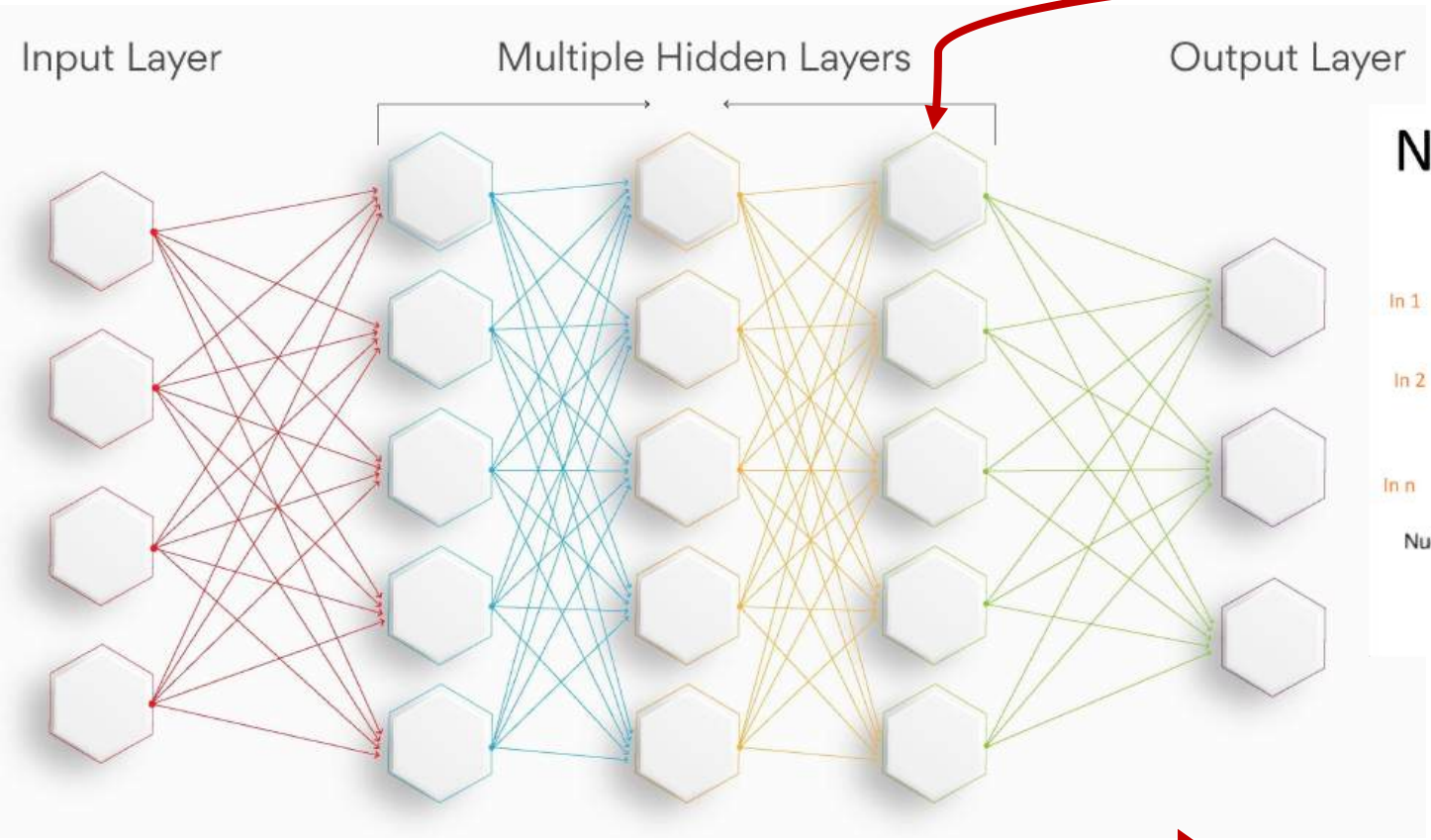


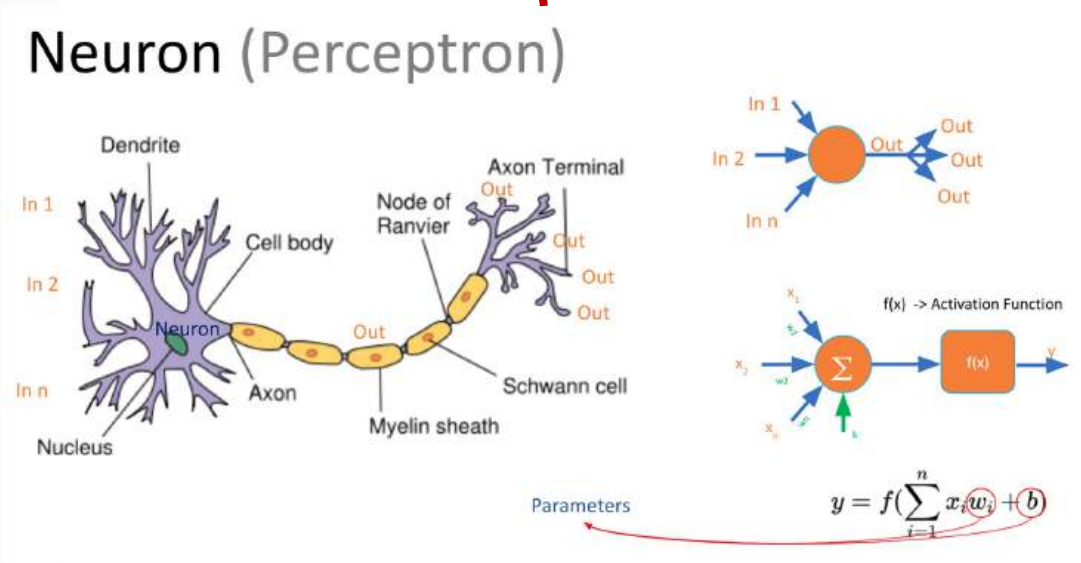
https://medium.com/analytics-vidhya/journey-of-gradient-descent-from-local-to-global-c851eba3d367

Artificial Neural Networks



What is an Artificial Neural Network (ANN)?





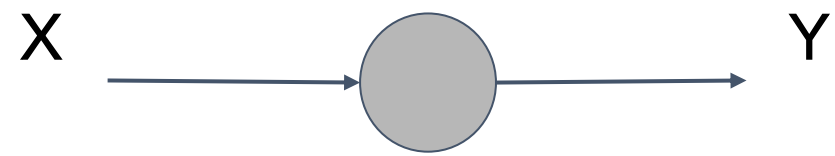
Sequential

Dense Neural Network (DNN)

https://www.turing.com/kb/importance-of-artificial-neural-networks-in-artificial-intelligence

A neuron

a neuron's output is a function of its inputs (in this case only one)



$$y = f(x) = wx+b$$

There are only two parameters to adjust:
The weight for each input and a bias

First scenario: a regression

Linear Regression with a Single Neuron

colab.research.google.com

Regression.ipynb

```
[2] import tensorflow as tf
    import numpy as np
    from tensorflow import keras
```

```
# define a neural network with one neuron
# for more information on TF functions see: https://www.tensorflow.org/api_docs
my_layer = keras.layers.Dense(units=1, input_shape=[1])
model = tf.keras.Sequential([my_layer])
```

```
# use stochastic gradient descent for optimization and
# the mean squared error loss function
model.compile(optimizer='sgd', loss='mean_squared_error')
```

```
# define some training data (xs as inputs and ys as outputs)
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

```
# fit the model to the data (aka train the model)
model.fit(xs, ys, epochs=500)
```

1 layer, 1 neuron, 1 input

Stochastic gradient descent

Inputs and outputs (labels)

Train the model


ys
Labels
xs
Data
Machine Learning
Rules
model

Linear Regression with a Single Neuron

colab.research.google.com

Regression.ipynb

```
[2] import tensorflow as tf
    import numpy as np
    from tensorflow import keras
```

```
# define a neural network with one neuron
# for more information on TF functions see: https://www.tensorflow.org/api_docs
my_layer = keras.layers.Dense(units=1, input_shape=[1])
model = tf.keras.Sequential([my_layer])

# use stochastic gradient descent for optimization and
# the mean squared error loss function
model.compile(optimizer='sgd', loss='mean_squared_error')

# define some training data (xs as inputs and ys as outputs)
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

# fit the model to the data (aka train the model)
model.fit(xs, ys, epochs=500)
```

```
Epoch 500/500
1/1 [==============================] - 0s 6ms/step - loss: 3.4704e-05
<keras.callbacks.History at 0x7f1d6ccd7f10>
```

```
[4] print(model.predict([10.0]))
```

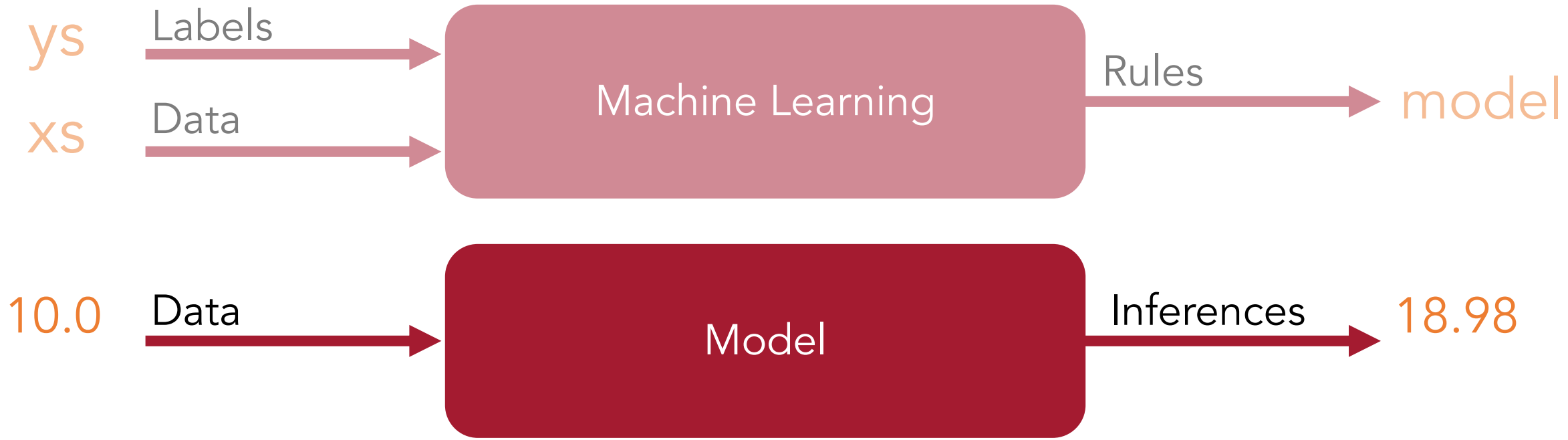
```
[[18.982813]]
```

```
[5] print(model.predict(xs))
```

```
[[-2.9897861]
 [-0.992277 ]
 [ 1.005232 ]
 [ 3.0027409]
 [ 5.00025  ]
 [ 6.997759 ]]
```

```
[6] print(my_layer.get_weights())
```

```
[array([[1.997509]], dtype=float32), array([-0.992277], dtype=float32)]
```


ys
Labels
xs
Data
Machine Learning
Rules
model
10.0
Data
Model
Inferences
18.98

Linear Regression with a Single Neuron

colab.research.google.com

Regression.ipynb

```
[2] import tensorflow as tf
    import numpy as np
    from tensorflow import keras
```

```
# define a neural network with one neuron
# for more information on TF functions see: https://www.tensorflow.org/api_docs
my_layer = keras.layers.Dense(units=1, input_shape=[1])
model = tf.keras.Sequential([my_layer])

# use stochastic gradient descent for optimization and
# the mean squared error loss function
model.compile(optimizer='sgd', loss='mean_squared_error')

# define some training data (xs as inputs and ys as outputs)
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

# fit the model to the data (aka train the model)
model.fit(xs, ys, epochs=500)
```

```
Epoch 500/500
1/1 [==============================] - 0s 6ms/step - loss: 3.4704e-05
<keras.callbacks.History at 0x7f1d6ccd7f10>
```

```
[4] print(model.predict([10.0]))

    [[18.982813]]
```

```
[5] print(model.predict(xs))

    [[-2.9897861]
     [-0.992277 ]
     [ 1.005232 ]
     [ 3.0027409]
     [ 5.00025  ]
     [ 6.997759 ]]
```

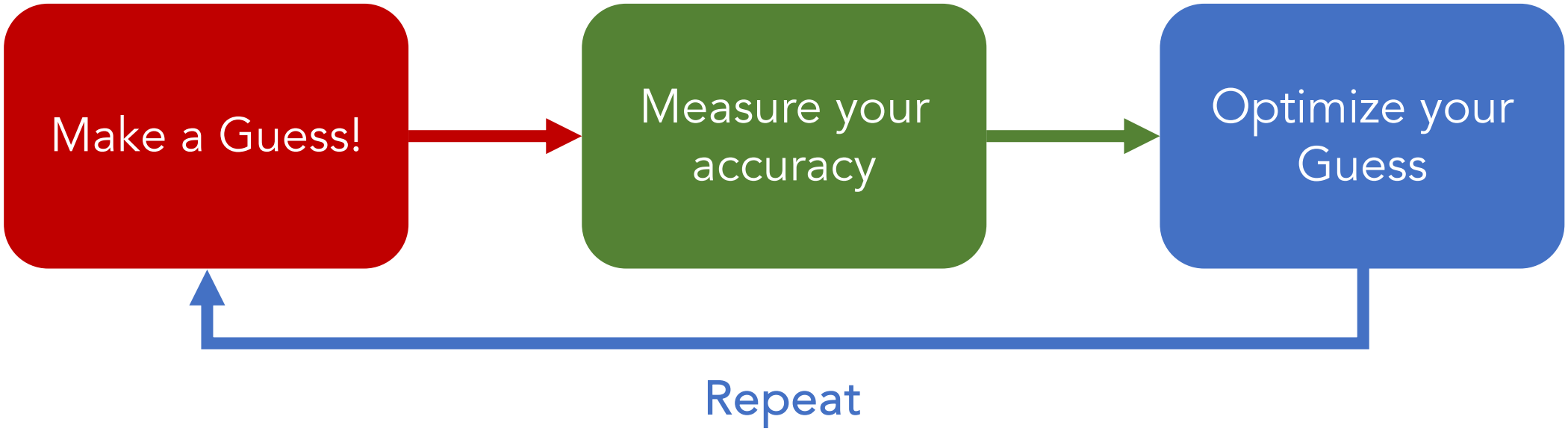
```
[6] print(my_layer.get_weights())

    [array([[1.997509]], dtype=float32), array([-0.992277], dtype=float32)]
```

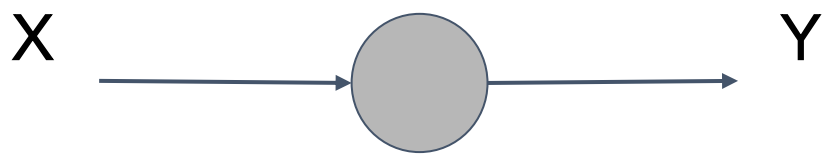
$Y = 2X - 1$

$Y = 1.9975X - 0.9922$

Not perfect,
but good enough for most cases!

The Machine Learning Paradigm





$$y = f(x) = wx+b$$

$$y = 1.9975x - 0.9922$$

https://pixabay.com/photos/sunglasses-cool-fashion-style-2649326/

Gracias!

Prof. Diego Méndez Chaves, Ph.D

Associate Professor - Electronics Engineering Department
Director of the Master Program in Internet of Things
Director of the Master Program in Electronics Engineering

https://perfilesycapacidades.javeriana.edu.co/en/persons/diego-mendez-chaves

diego-mendez@javeriana.edu.co