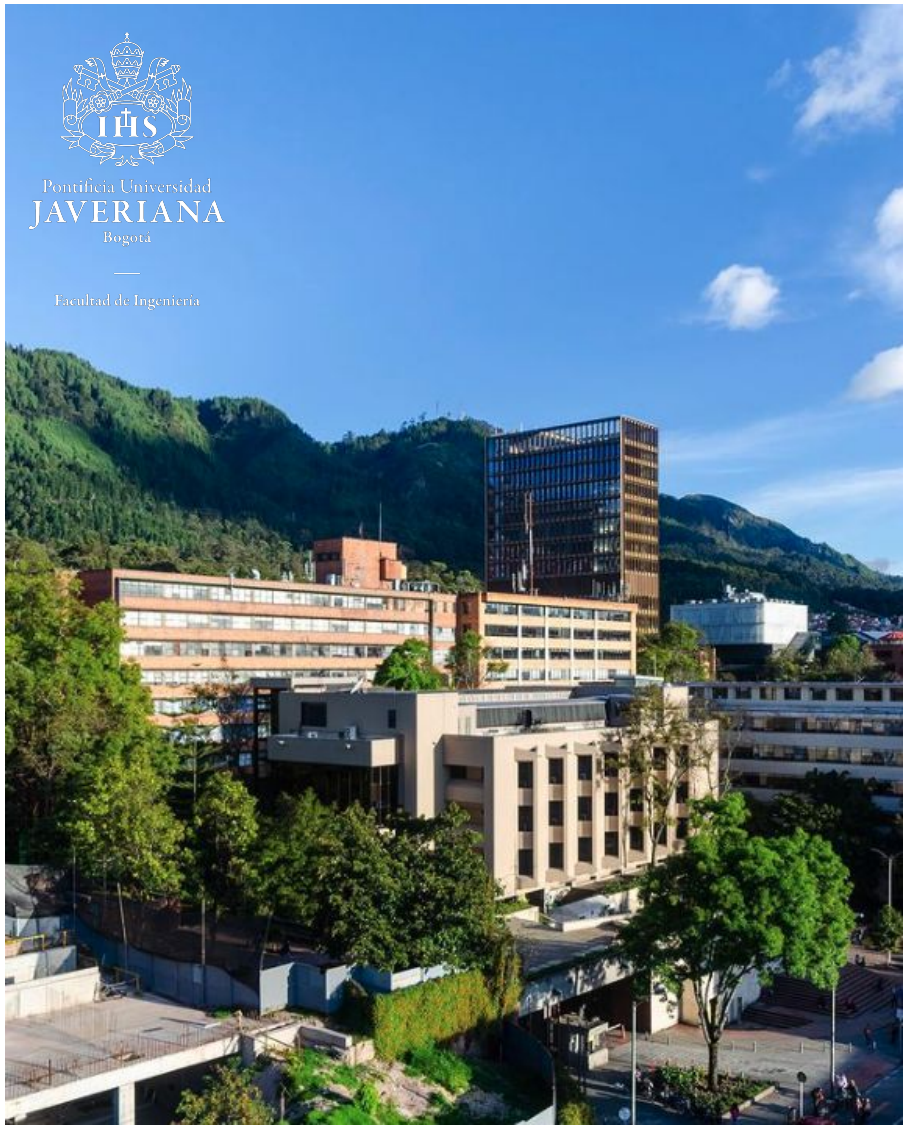
WALC 2023
Guayaquil, Ecuador
13 al 17 de Noviembre
26° Taller sobre Tecnologías de Redes Internet para América Latina y el Caribe
Fundación EsLaRed
Organización General

Pontificia Universidad JAVERIANA Bogotá
Facultad de Ingeniería

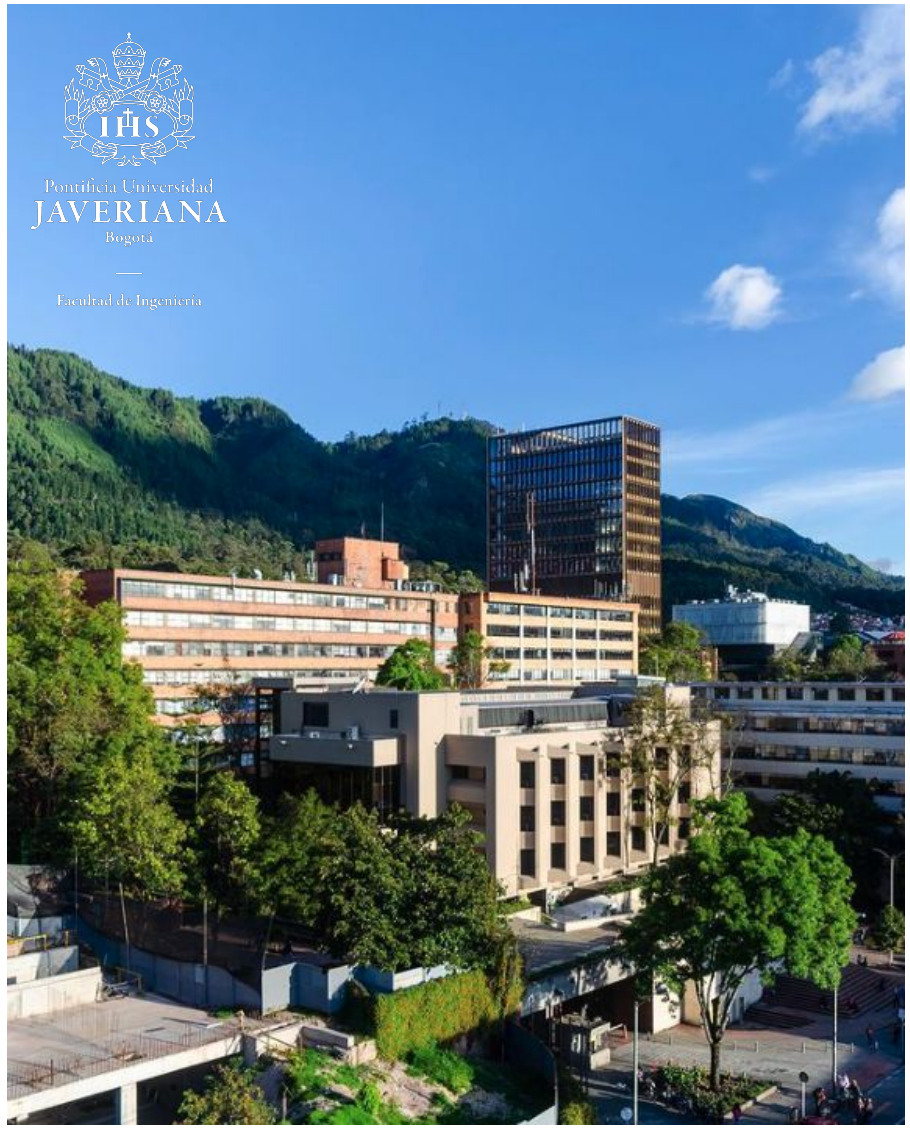
Machine Learning Fundamentals

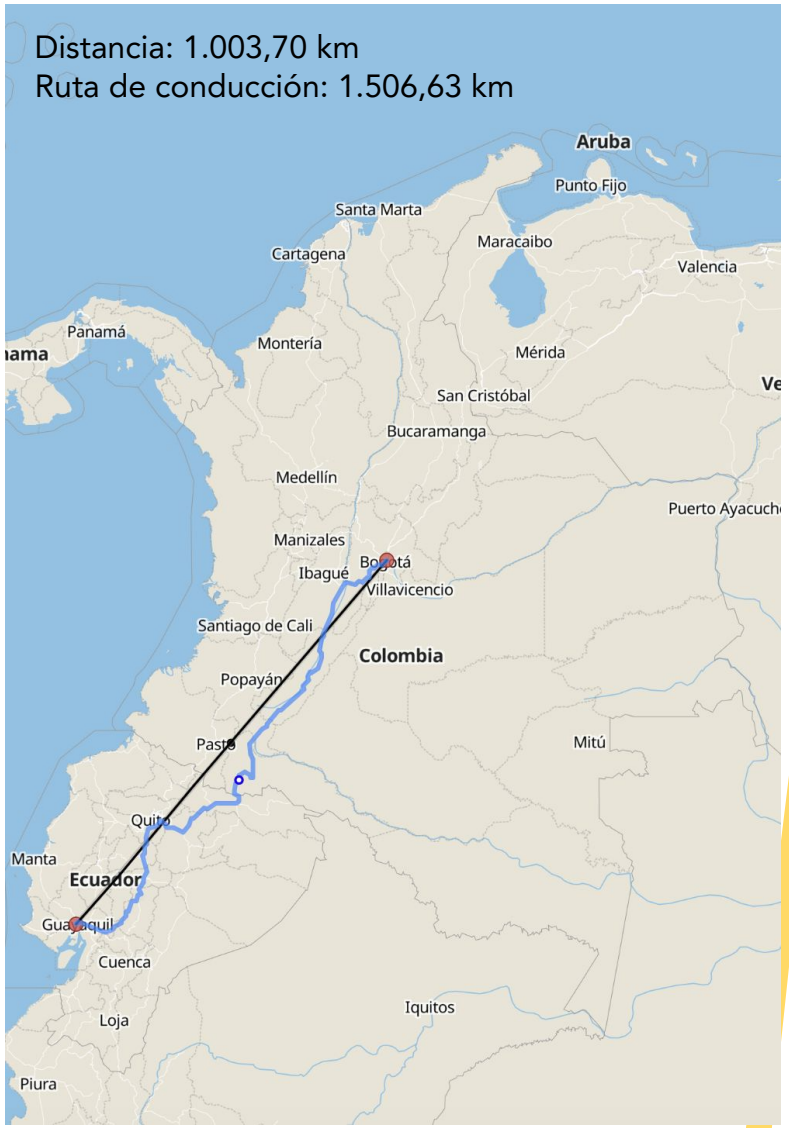
Workshop para América Latina y el Caribe (WALC)
Track 2 – Inteligencia Artificial Aplicada
November 14, 2023

Pontificia Universidad Javeriana – Bogotá, Colombia

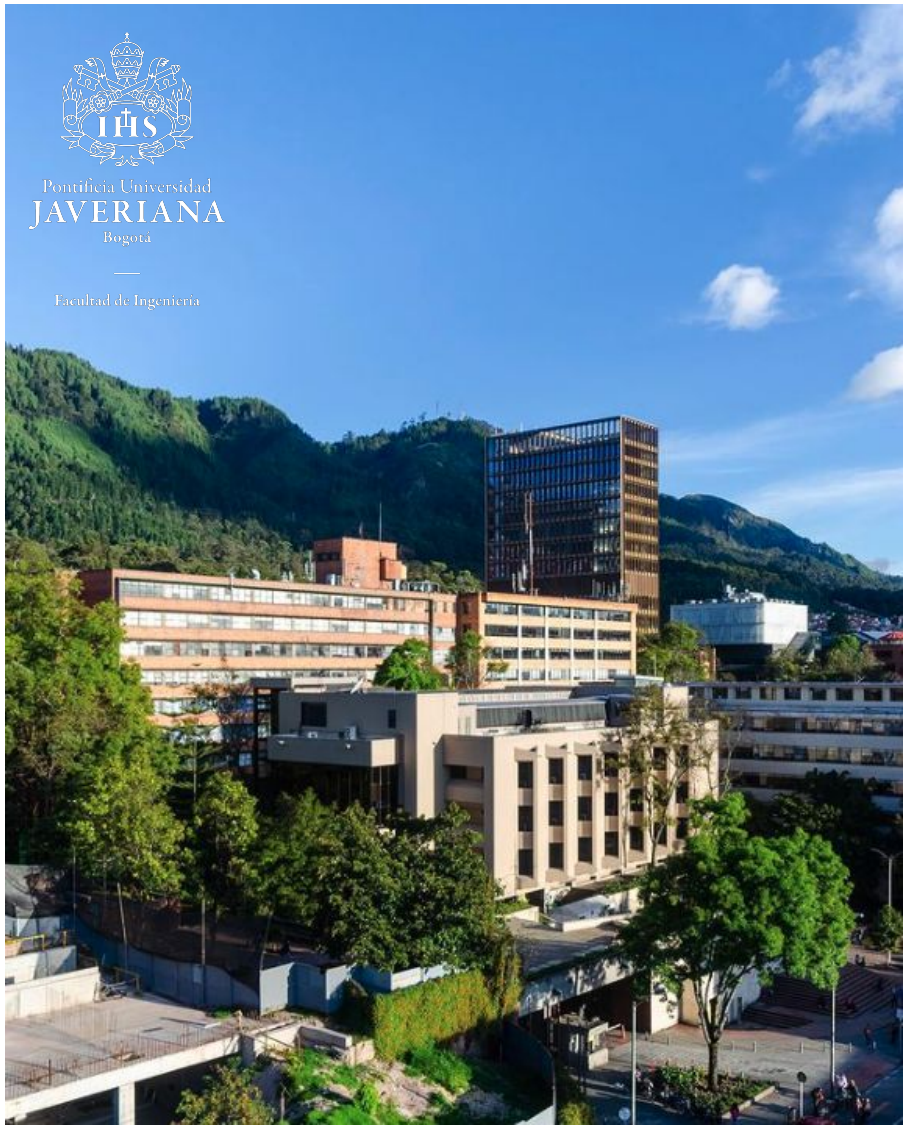


Pontificia Universidad Javeriana – Bogotá, Colombia





Pontificia Universidad Javeriana



18 Schools

62 Departments

15 Institutes

32.000 Students

27.000 Undergradute Students

1.700 Employees

160.000 Alumni

42 > Undergraduate

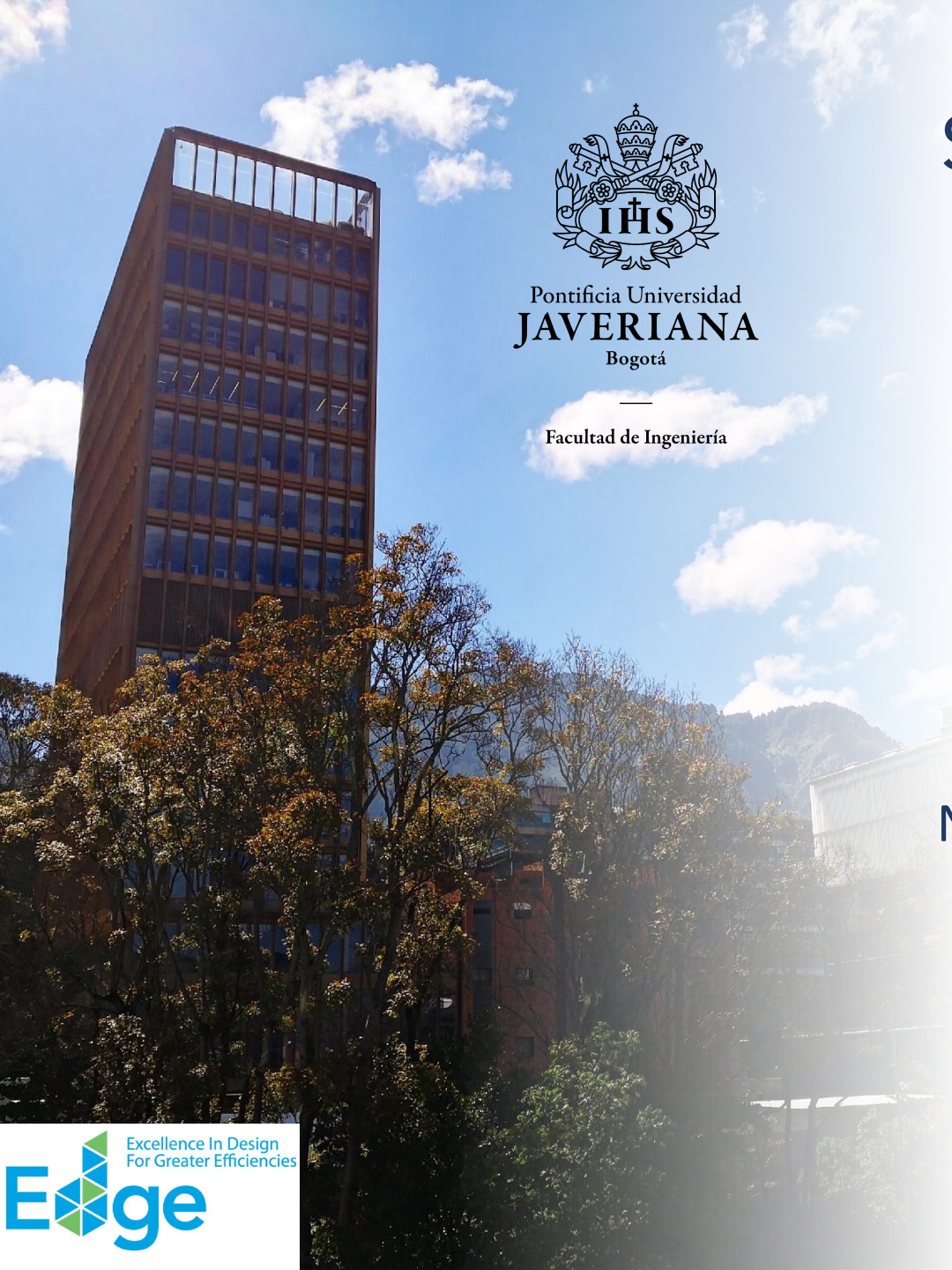
92 > Specializations

67 > Masters

15 > PhD. programs

5 Centers of Excellence

96 Research Groups



Pontificia Universidad JAVERIANA Bogotá

Facultad de Ingeniería

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

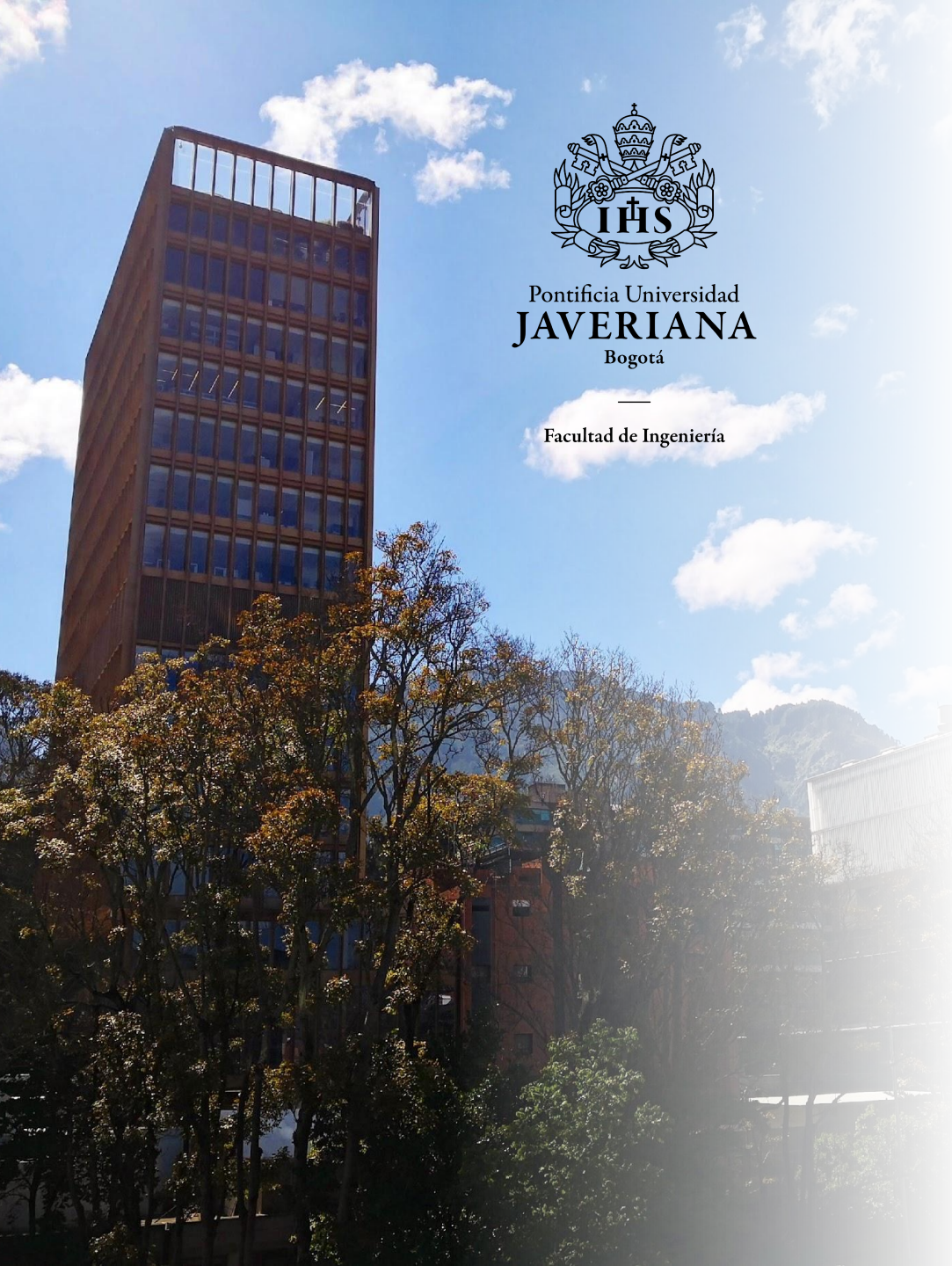
2
Centers of Excellence

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



Pontificia Universidad
JAVERIANA
Bogotá

Facultad de Ingeniería

Our Programs

Undergraduate

- Electronics Eng.
- Civil Eng.
- Industrial Eng.
- Systems and Computers Eng.
- Mechatronics Eng.
- Telecommunications and Networks Eng.
- Bioengineering
- Mechanical Eng.
- Data Science

C D I O

CONCEIVE DESIGN IMPLEMENT OPERATE

ABET

Graduate

- Doctorate in Engineering
- Doctorate in Science and Technology of Materials
- Master in Electronics Eng.
- Master in Civil Eng.
- Master in Hydro-systems Eng.
- Master in Systems and Computing Eng.
- Master in Bioengineering
- Master in Analytics
- Master in Industrial Eng.
- Master in Logistics and Transport
- Master in Energy and Sustainability
- Master in Digital Security NEW
- Master in Internet of Things NEW
- Master in Artificial Intelligence NEW

Diego Méndez Chaves, Ph.D

School of Engineering – Pontificia Universidad Javeriana

2009-2012	Ph.D and M.Sc in Computer Science Full Scholarship	University of South Florida (USA) Participatory sensing and location-based systems
2005-2008	M.Eng in Electronics Engineering and Computers Full Scholarship	Universidad de Los Andes (Colombia) Wireless sensor networks
2000-2004	B.Eng in Electronics Engineering	Universidad Nacional de Colombia Embedded systems design

Diego Méndez Chaves, Ph.D

School of Engineering – Pontificia Universidad Javeriana

Universidad Nacional de Colombia	2006: Instructor Professor Teaching activities
Universidad de Los Andes	2005-2009: Graduate Assistant and Instructor Professor Teaching and research activities
University of South Florida	2009-2012: Graduate Assistant and Instructor Professor Teaching and research activities
Pontificia Universidad Javeriana	2012 - to date: Associate Professor Teaching, research and service activities
External Collaborations	2016: Invited Researcher at the IHP Microelectronics (Frankfurt-Oder, Germany) 2017-2018: Advising Engineering Director and Project Manager in IoT (IKUSI, Colombia)

- Teaching: 21 undergraduate (7), 26 master (7), 5 doctoral (3).
- Research: ~3 international cooperation (2), 1 center of excellence, >10 industry related, >20 research, ~59 publications.

Pontificia Universidad
JAVERIANA
Bogotá

Facultad de Ingeniería

Diego Méndez Chaves, Ph.D

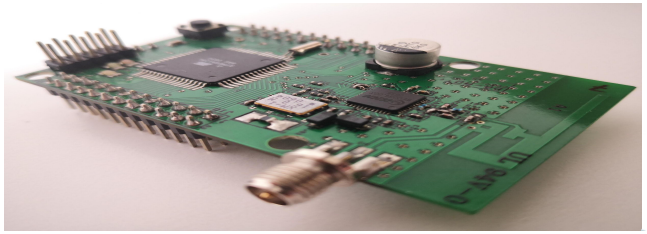
School of Engineering – Pontificia Universidad Javeriana

- 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 – Founding member and local coordinator.
- Research interests: IoT, embedded systems, wireless sensor networks, participatory sensing, digital systems design and embedded operating systems.

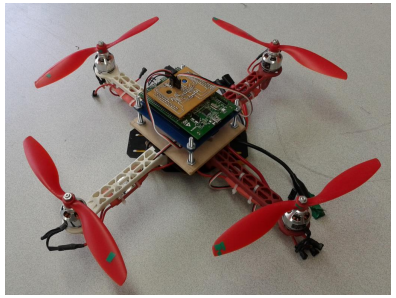
Embedded Research

Digital oscilloscope and a logic analyzer on a GBA

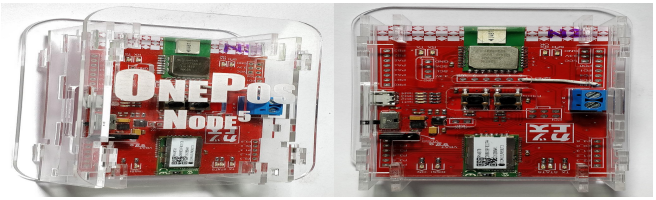
DA-MOTE: A Flexible Platform for WSN Apps



A completely customized UAV for research projects



Frictionless Indoor Location using BLE



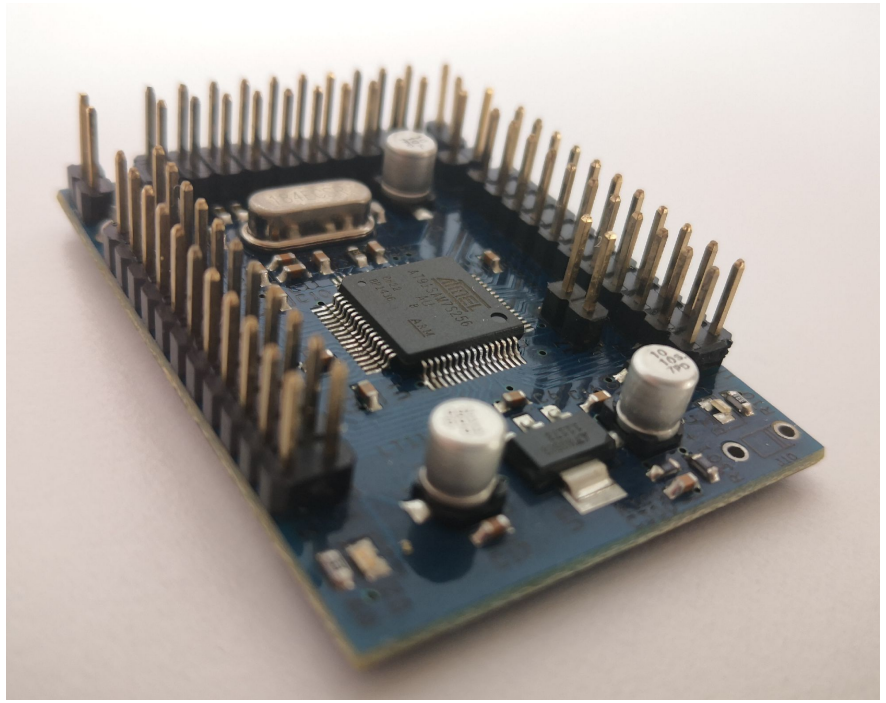
OpenWuR: Open Platform for WuR-based Apps

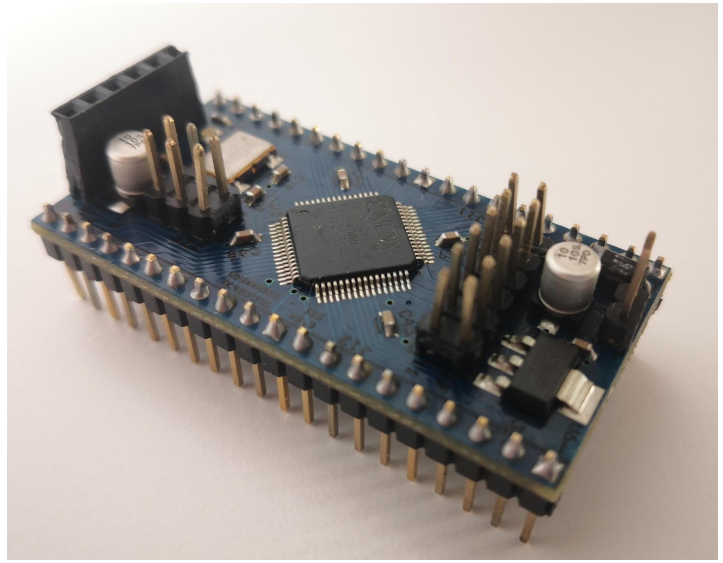
4D Phenotyping with LiDAR and Multispectral Images

The one that started all!

Embedded Teaching

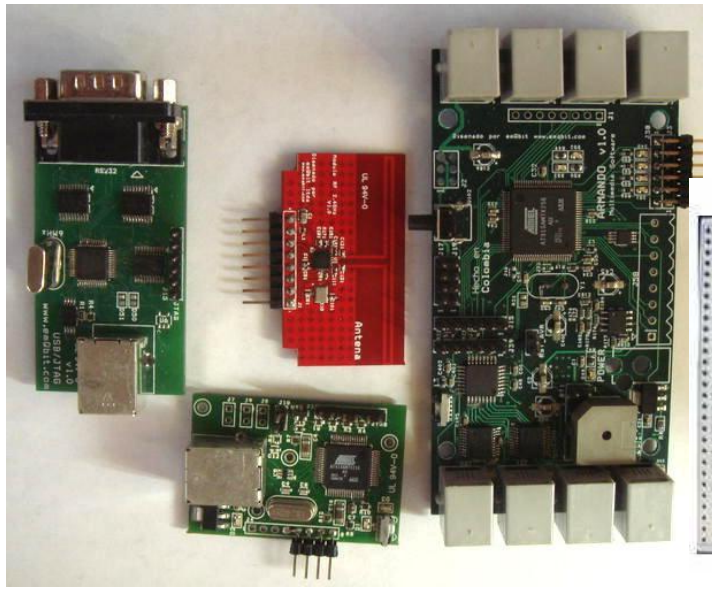
Design with Microcontrollers (SAM7S)

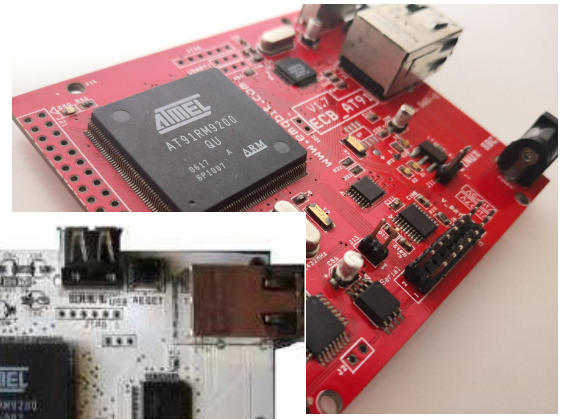




Digital Hardware Design (XC9572XL)

STEM-oriented robotics platform

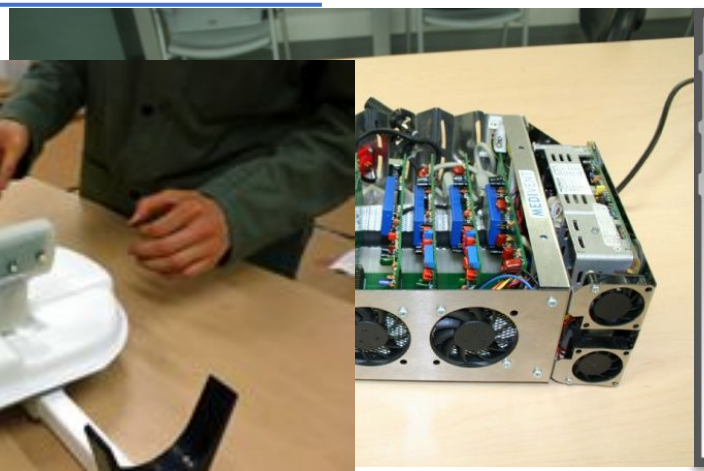


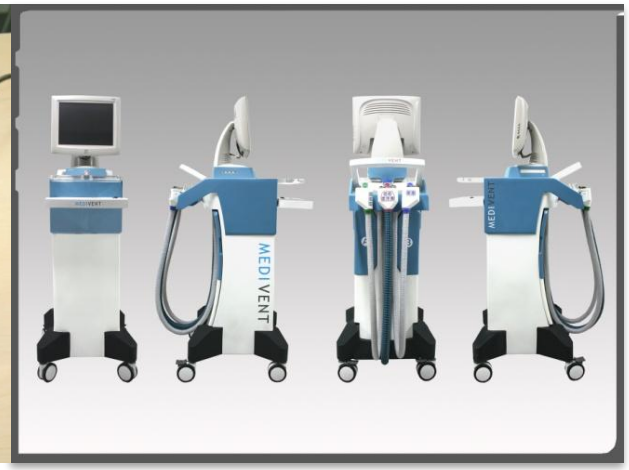


Embedded Linux projects (AT91RM9200)

Co-design projects (SoC + FPGA)

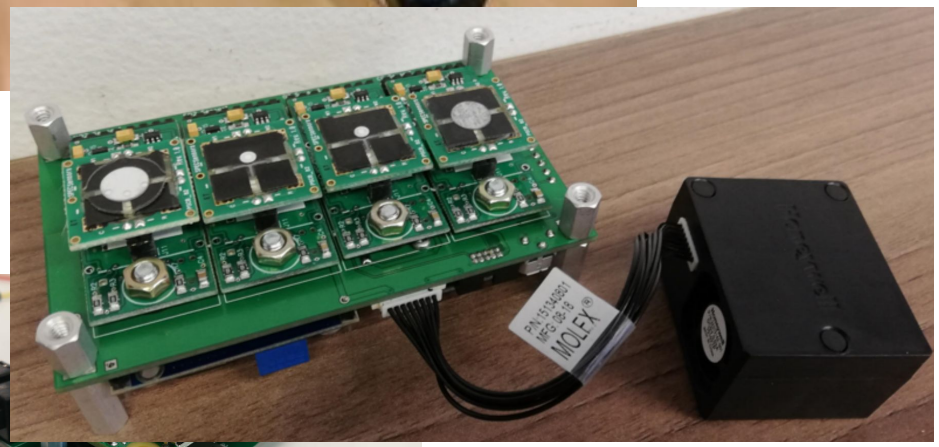
Embedded Service



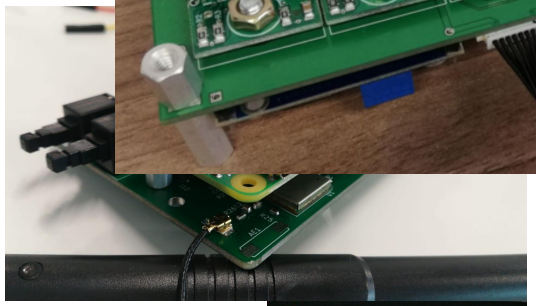


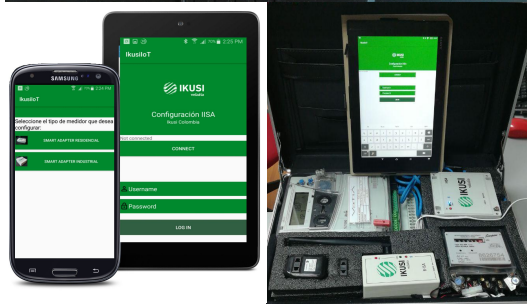
Medivent

Indumil



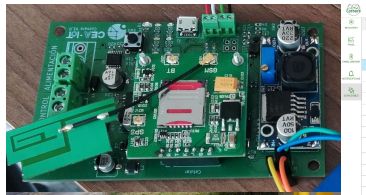
IKUSI

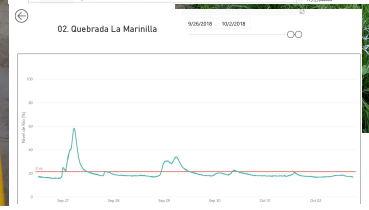




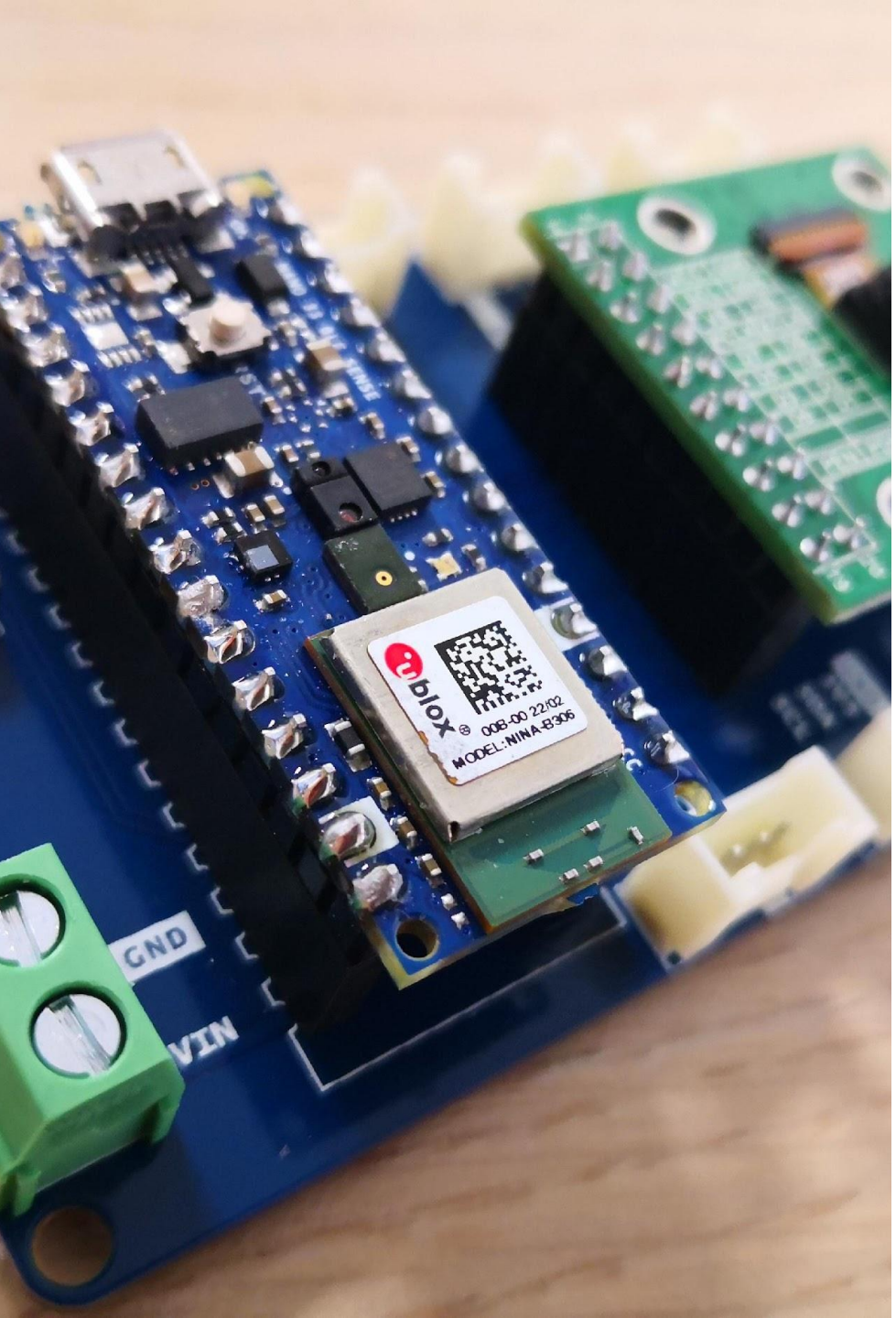
PIMA

Agrosavia





Cornare



Outline

- AI vs ML vs DL
- TinyML's Main Components
- The Machine Learning Paradigm
- Finding the Best Solution and Fitting a Model
- Regression and Classification with NN
- ML Issues
- Challenges of TinyML

Alexa, play some rock music!

Playing the Rock Hits playlist

ElephantEdge

Building The World's Most Advanced **Wildlife Tracker**.

smart parks · hackster.io AN AVNET COMMUNITY · EDGE IMPULSE · Microsoft · Western Digital. · NORDIC SEMICONDUCTOR · TAOGLAS · ublox · WWF · EARTHRANGER

Source: https://wildlabs.net/resources/competition/challenge-elephantedge

ElephantEdge

Risk Monitoring

"Know when an elephant is moving into a high-risk area and send real-time notifications to park rangers."

Conflict Monitoring

"Sense and alert when an elephant is heading into an area where farmers live."

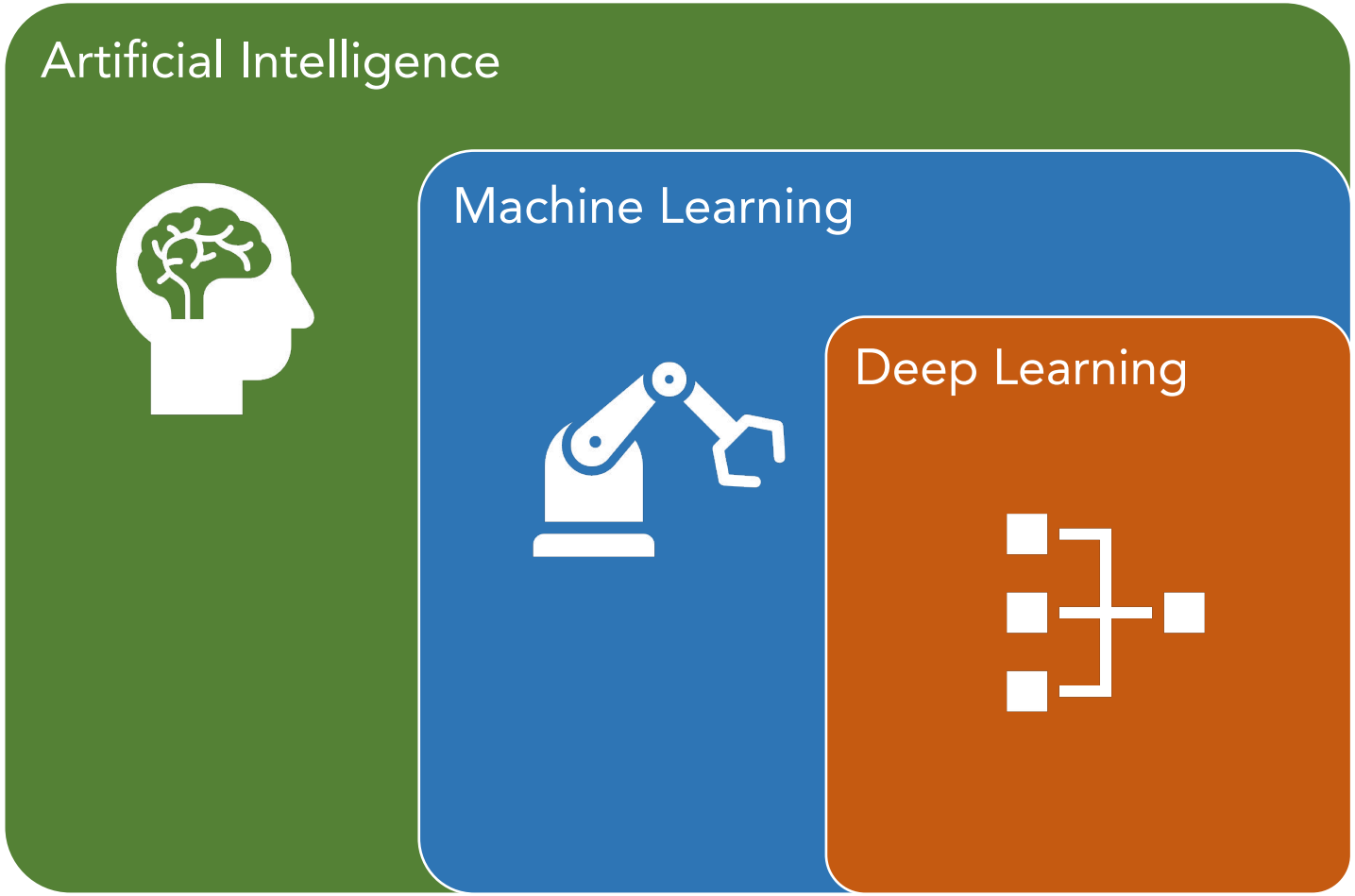
Activity Monitoring

"Classify the general behavior of the elephant, such as when it is drinking, eating, sleeping, etc."

Communication Monitoring

"Listen for vocal communications between elephants via the onboard microphone."

AI vs. ML vs. DL



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

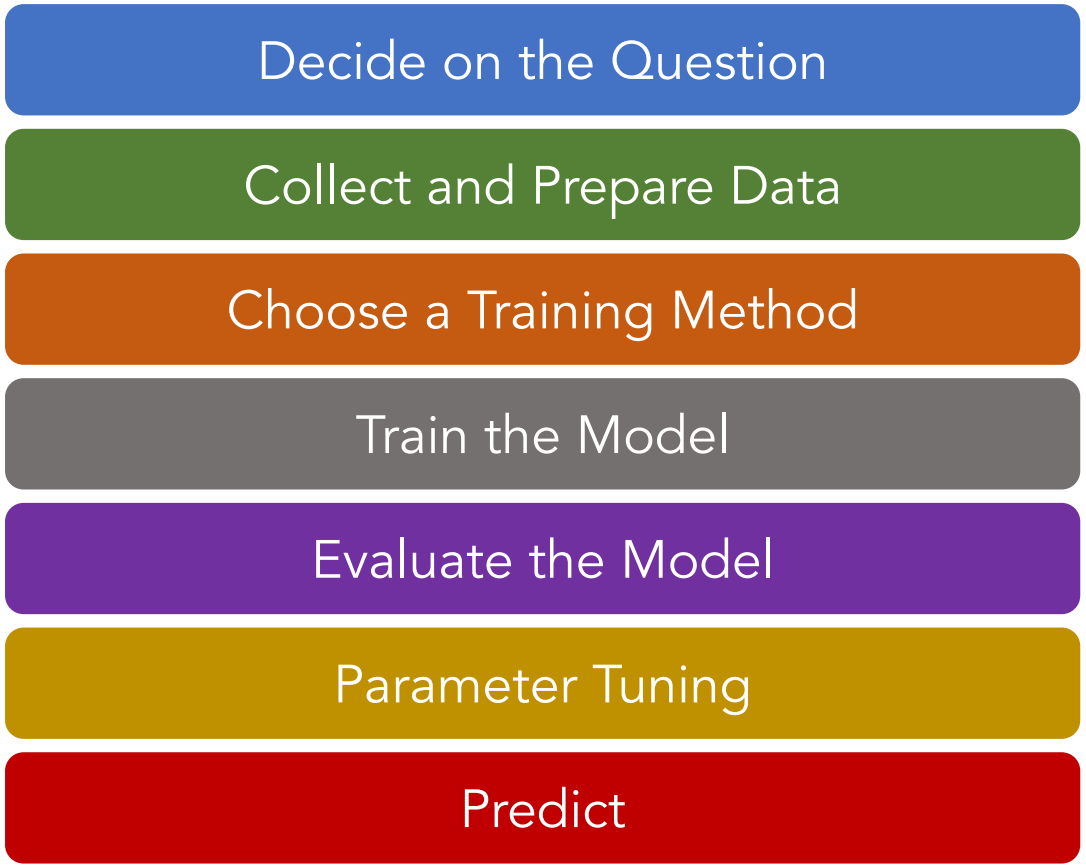
Machine learning (ML): a subset of AI that uses techniques that enable machines to use experience to improve at tasks.

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

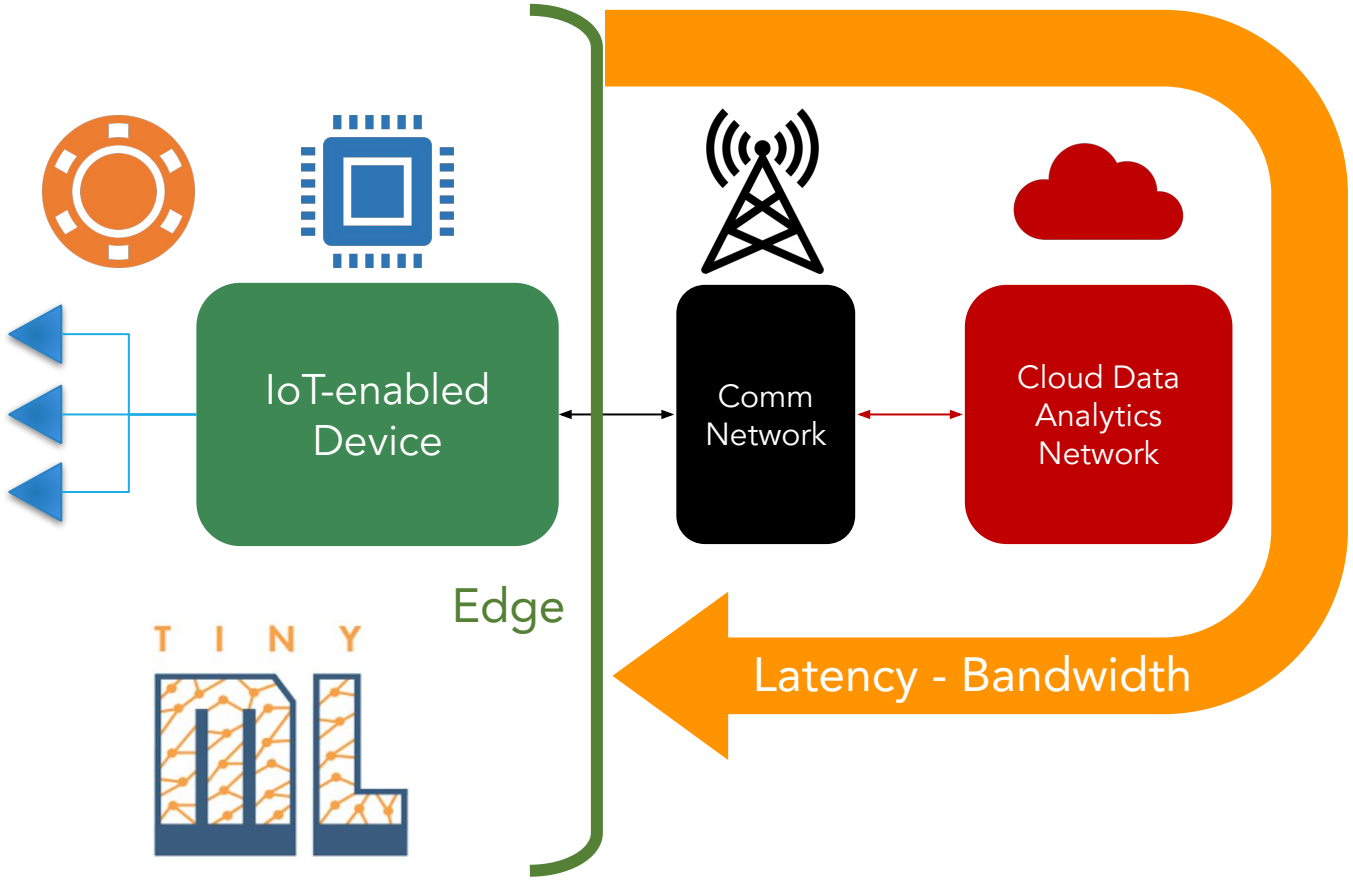
Based on https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

General Steps for Machine Learning

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



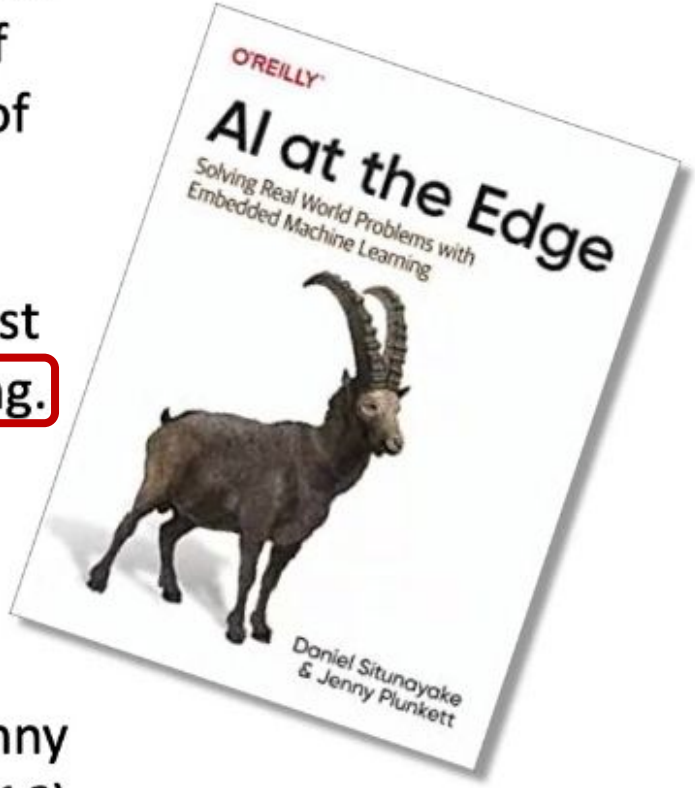
Based on https://microsoft.github.io/ML-For-Beginners/#/1-Introduction/4-techniques-of-ML/README?id=techniques-of-machine-learning



"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."

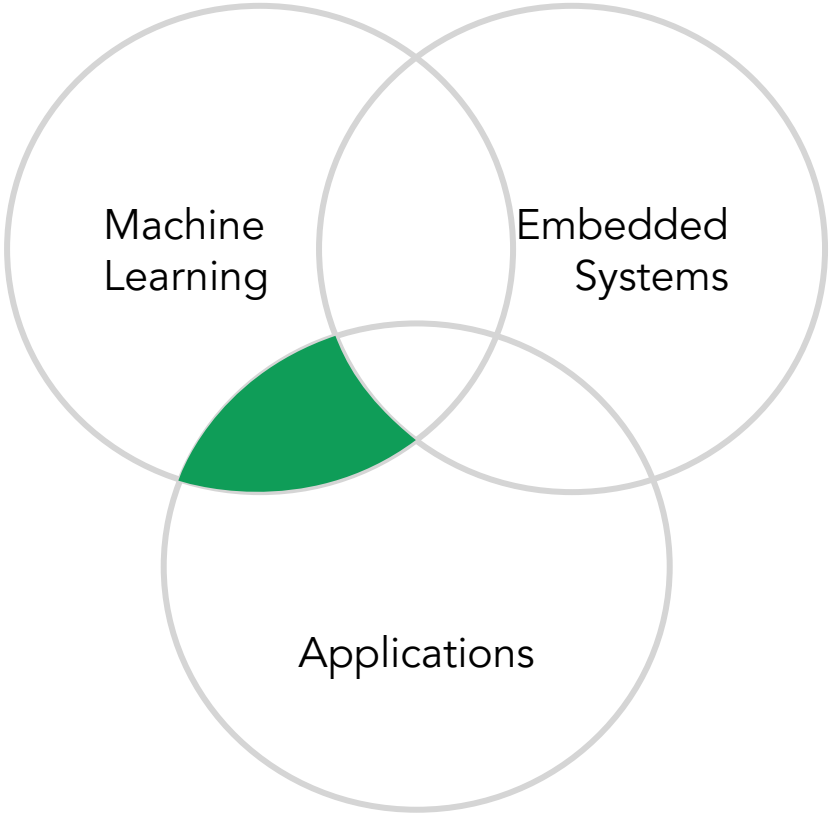


Situnayake, Daniel; Plunkett, Jenny
AI at the Edge (pp. 215-216)
O'Reilly Media

3 main components

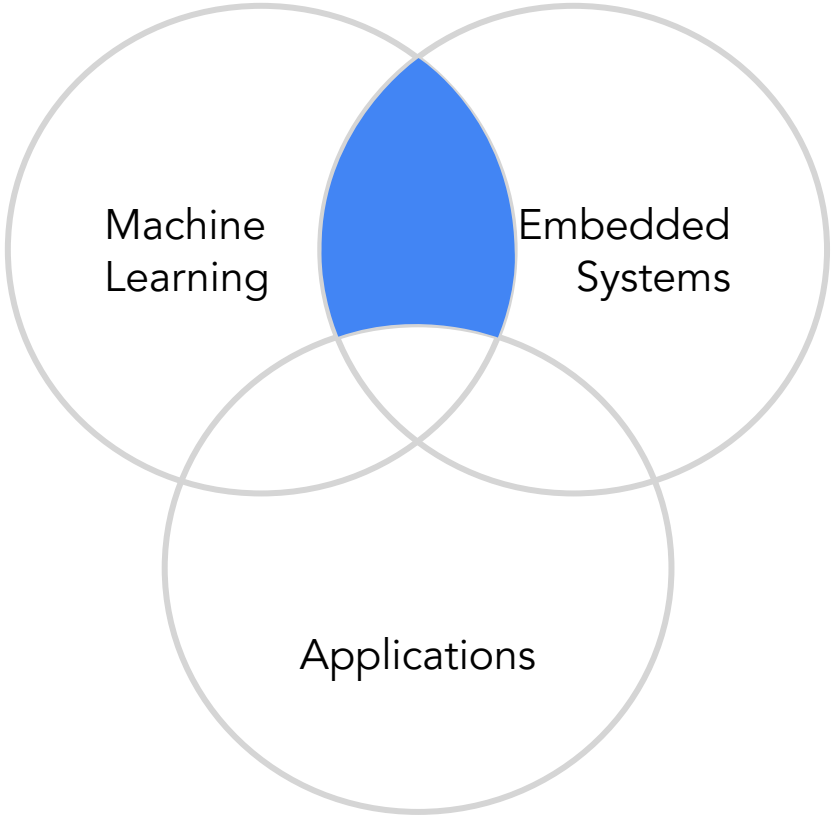
Interactions

How machine learning can enable new and interesting TinyML applications?



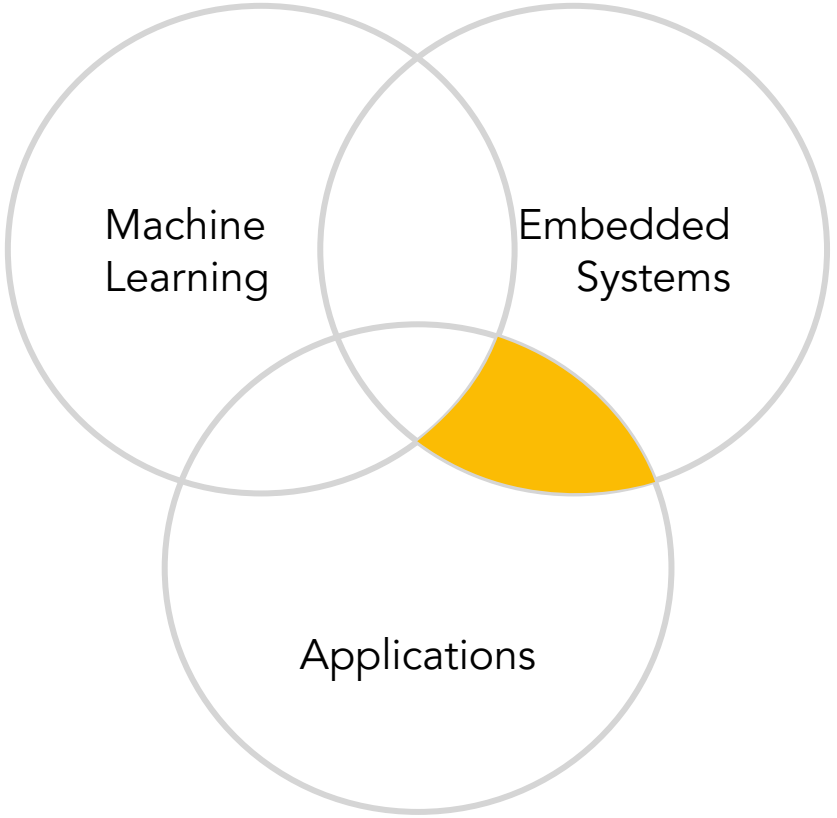
Interactions

What are the challenges with enabling machine learning on tiny, resource-constrained embedded devices?



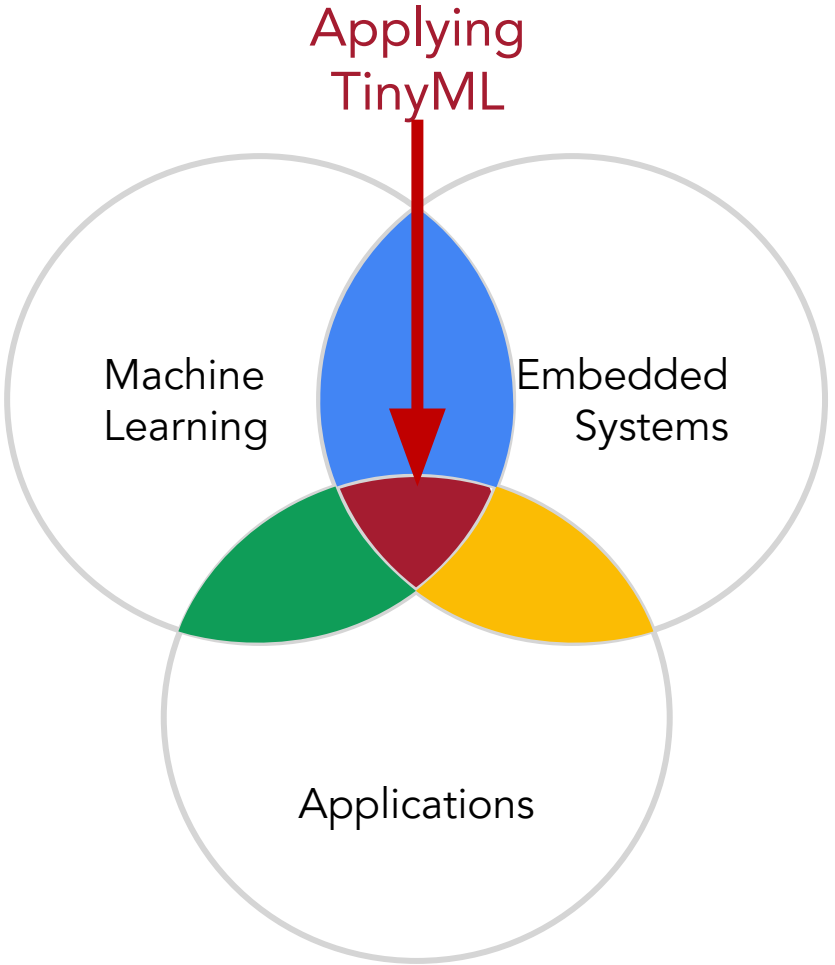
Interactions

What type of new use cases can we possibly enable on embedded systems that we could not otherwise do before?



Interactions

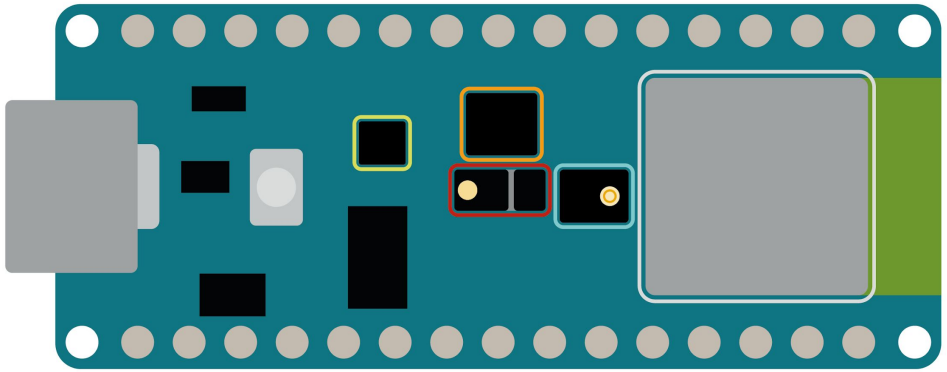
Given your understanding of things at these various intersections, you will have a deep understanding for how to apply TinyML



Out platform for experimental deployments

Arduino Nano 33 BLE Sense

NANO 33 BLE SENSE



- Color, brightness, proximity and gesture sensor
- Digital microphone
- Motion, vibration and orientation sensor
- Temperature, humidity and pressure sensor
- Arm Cortex-M4 microcontroller and BLE module

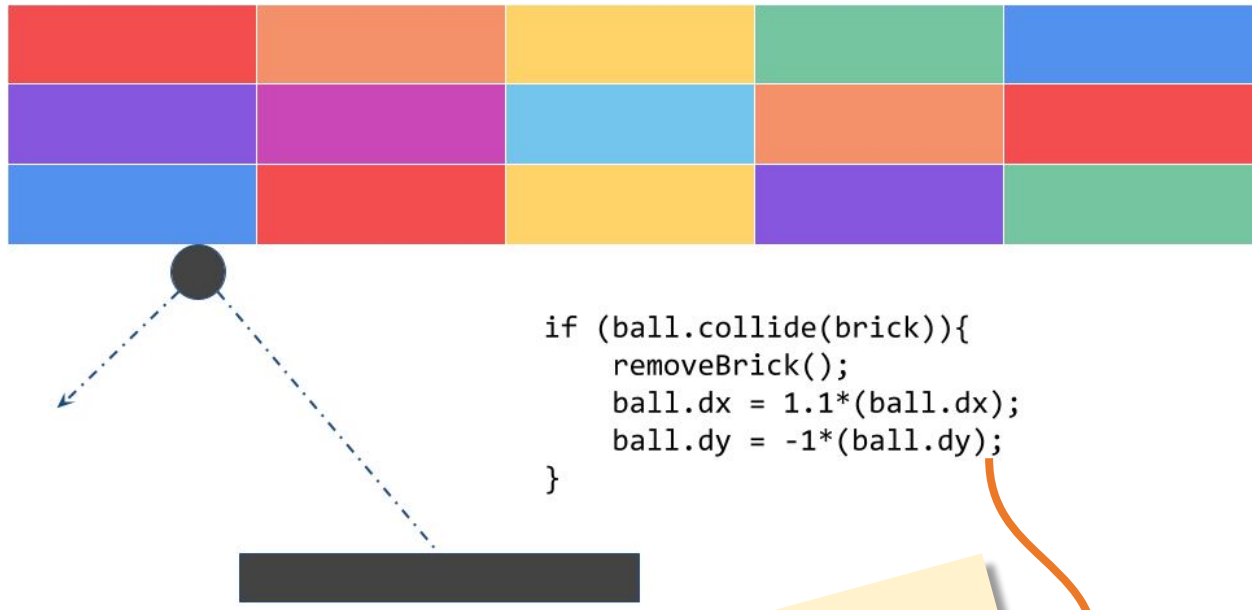
We will run through this long process

This is a first encounter with ML, but many things will be left to be experimented or developed.

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

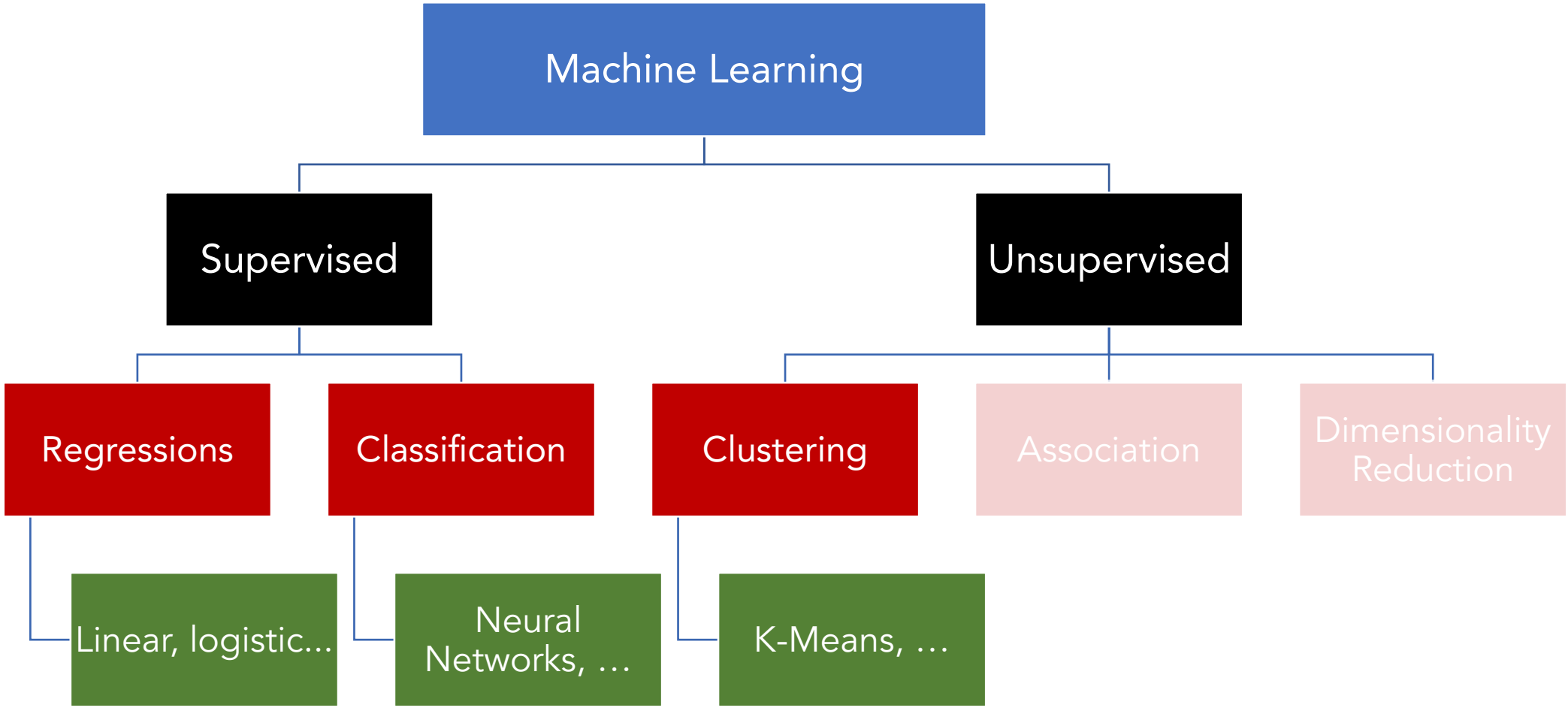
```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101
```

Label = BIKING

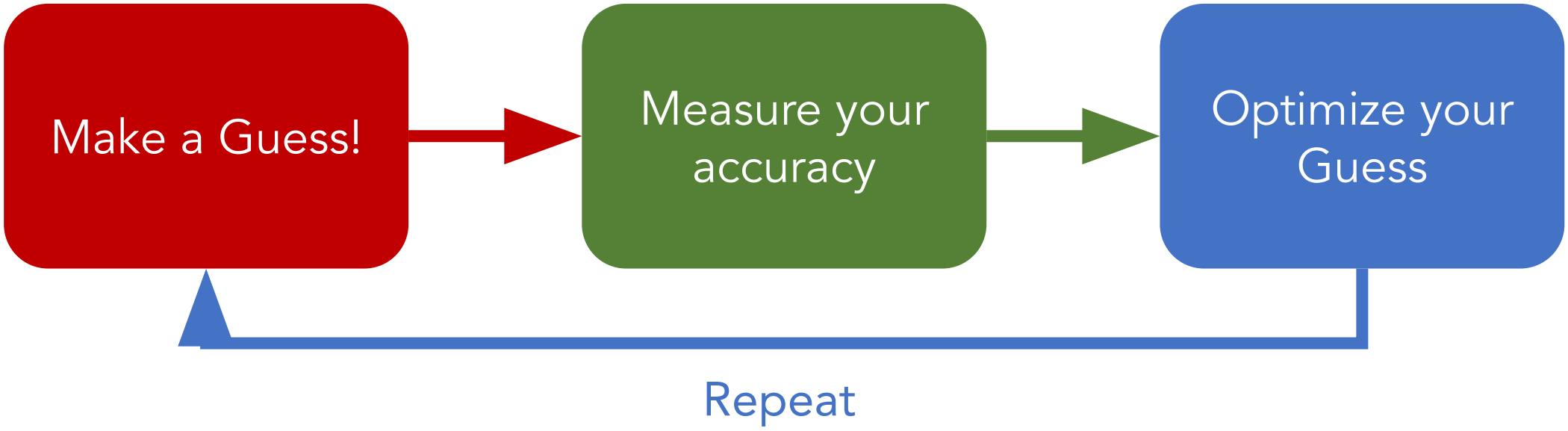
```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110
```

Label = GOLFING

Two Approaches

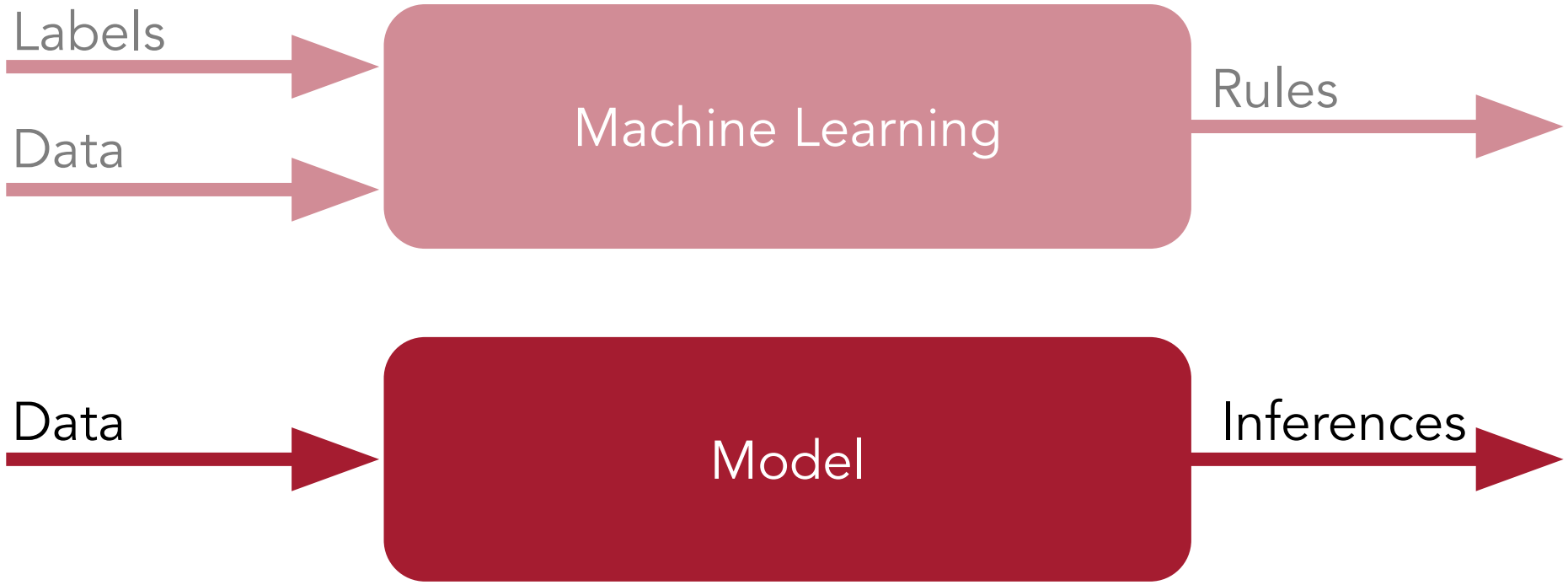


The Machine Learning Paradigm



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?

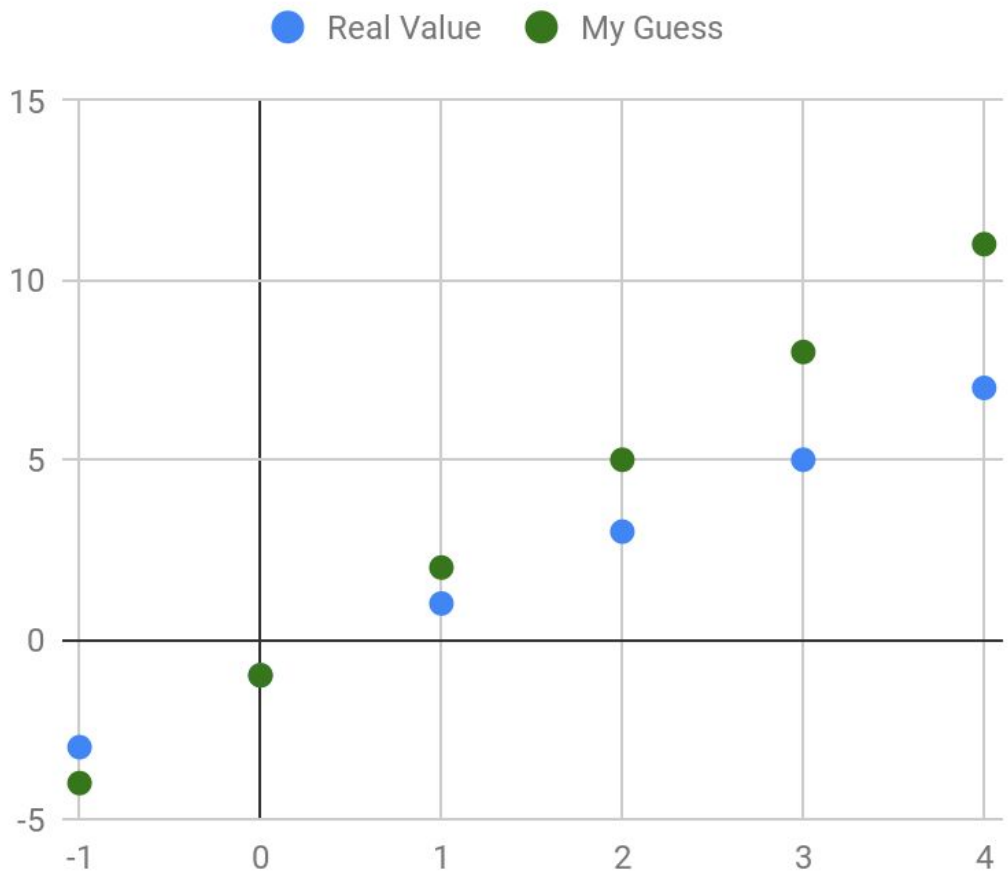
$Y = 3X - 1$

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

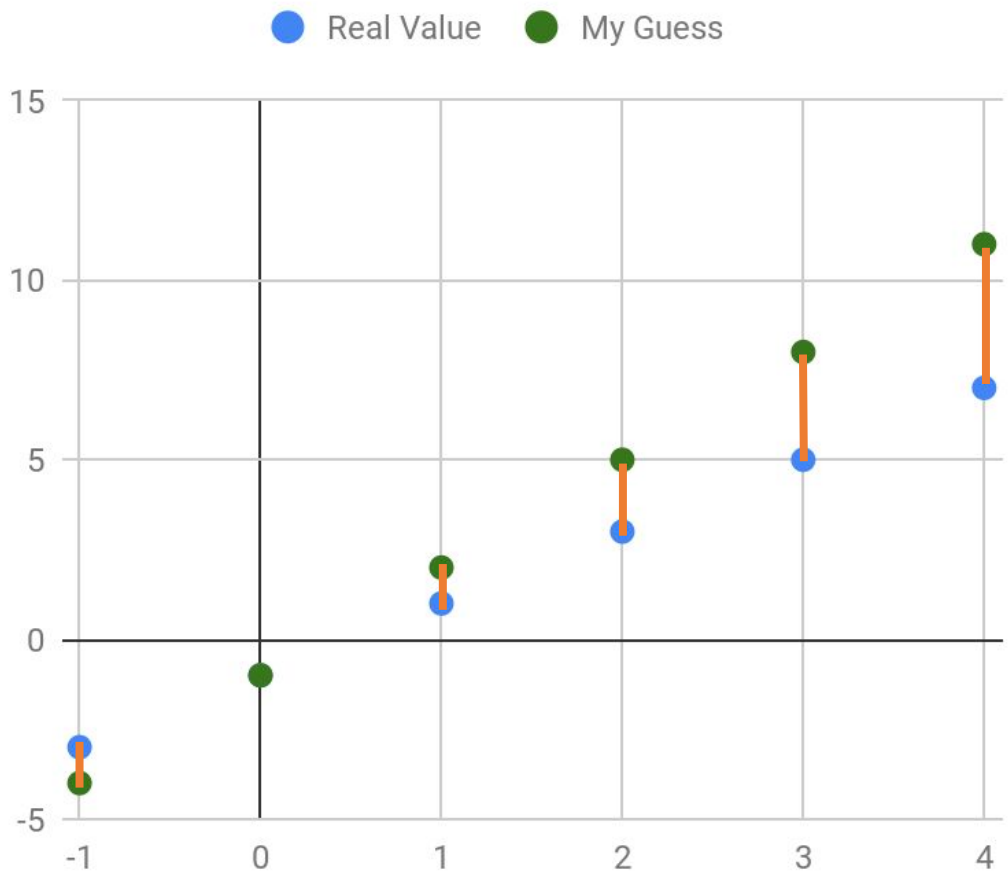
My $Y = \{ -4, -1, 2, 5, 8, 11 \}$

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

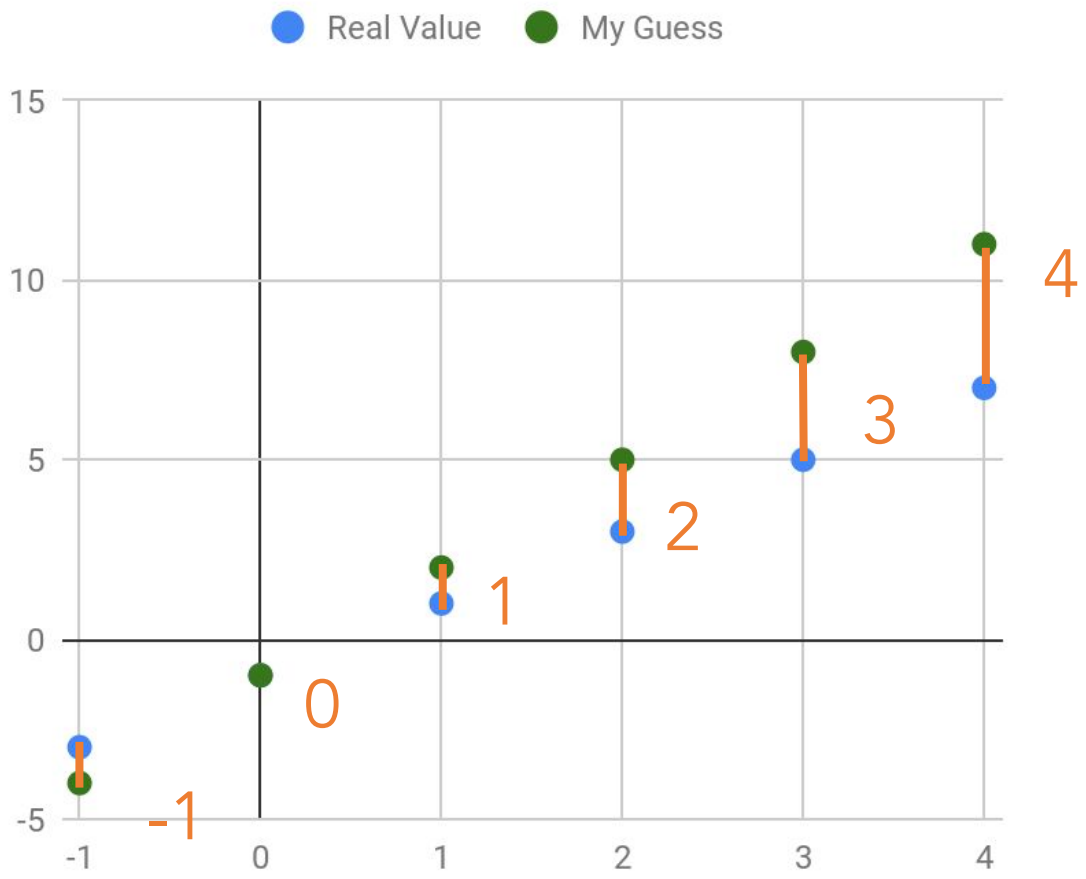
Let’s measure it!



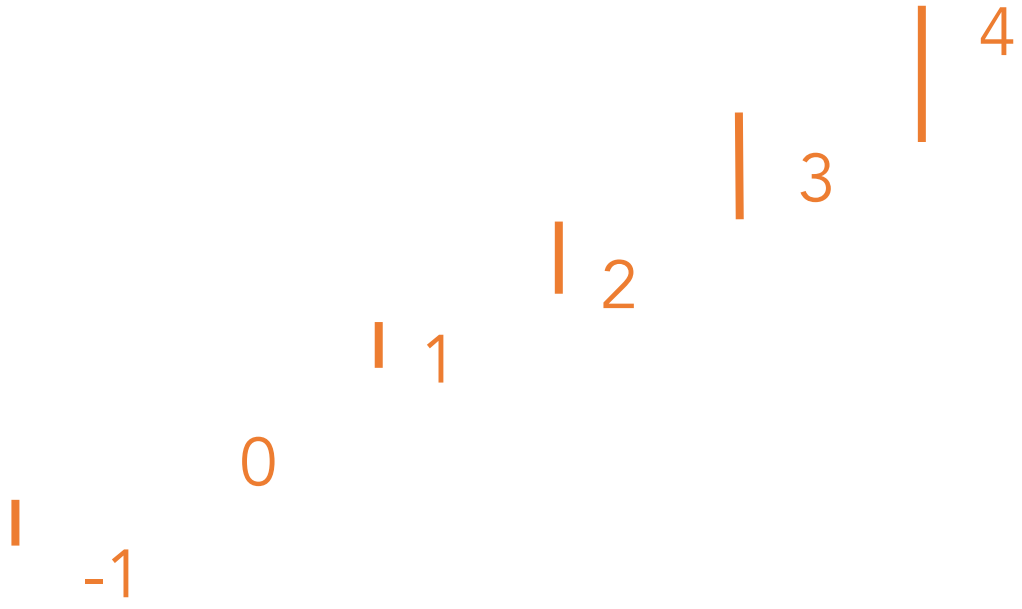
Let’s measure it!



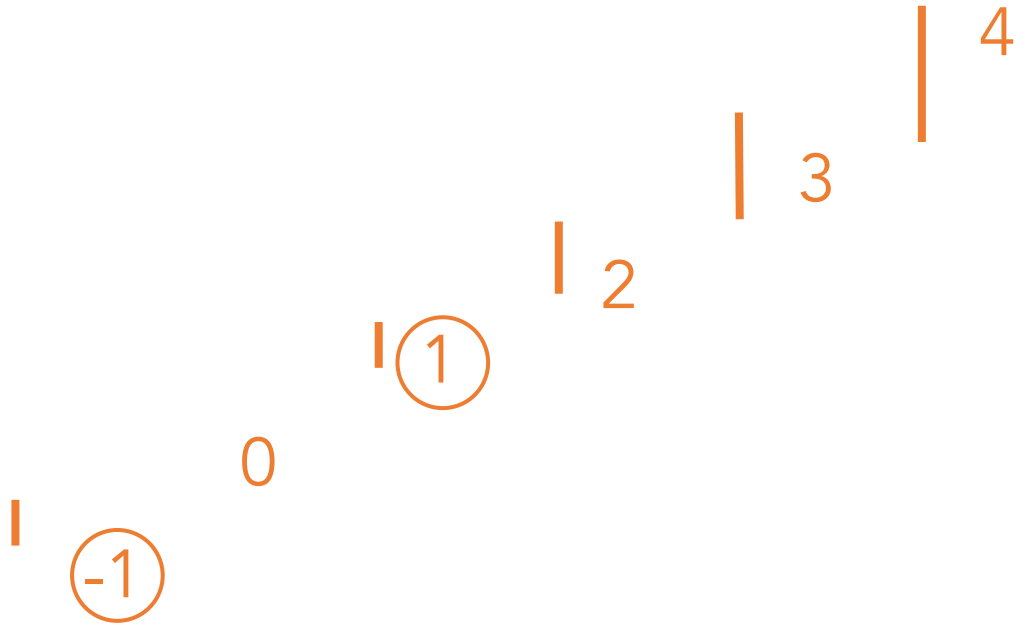
Let’s measure it!



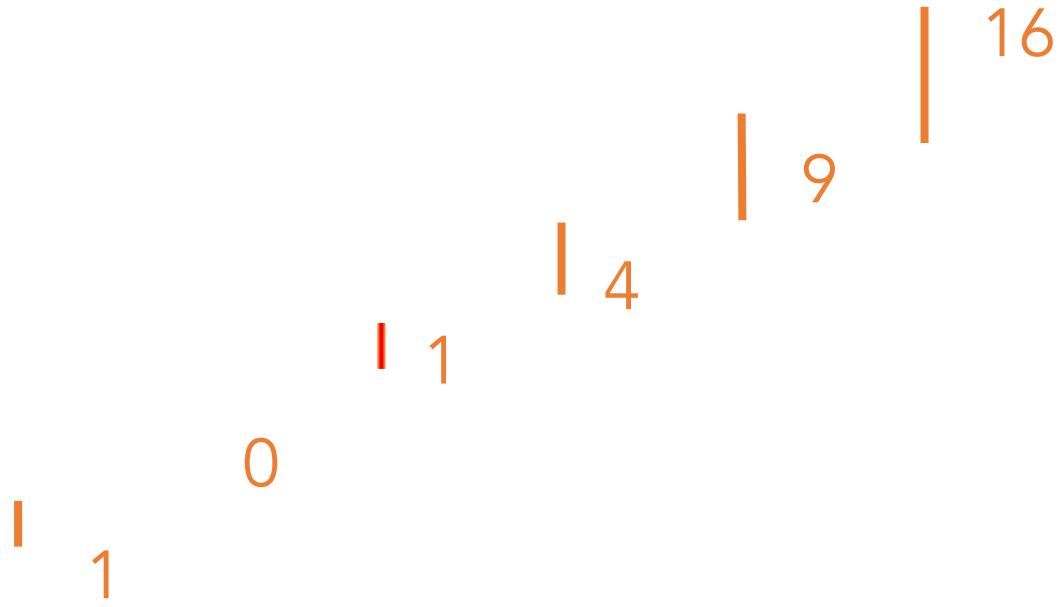
Let’s measure it!



Houston, we have a problem!



What if we square 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 $Y = \{-3, -1, 1, 3, 5, 7\}$

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

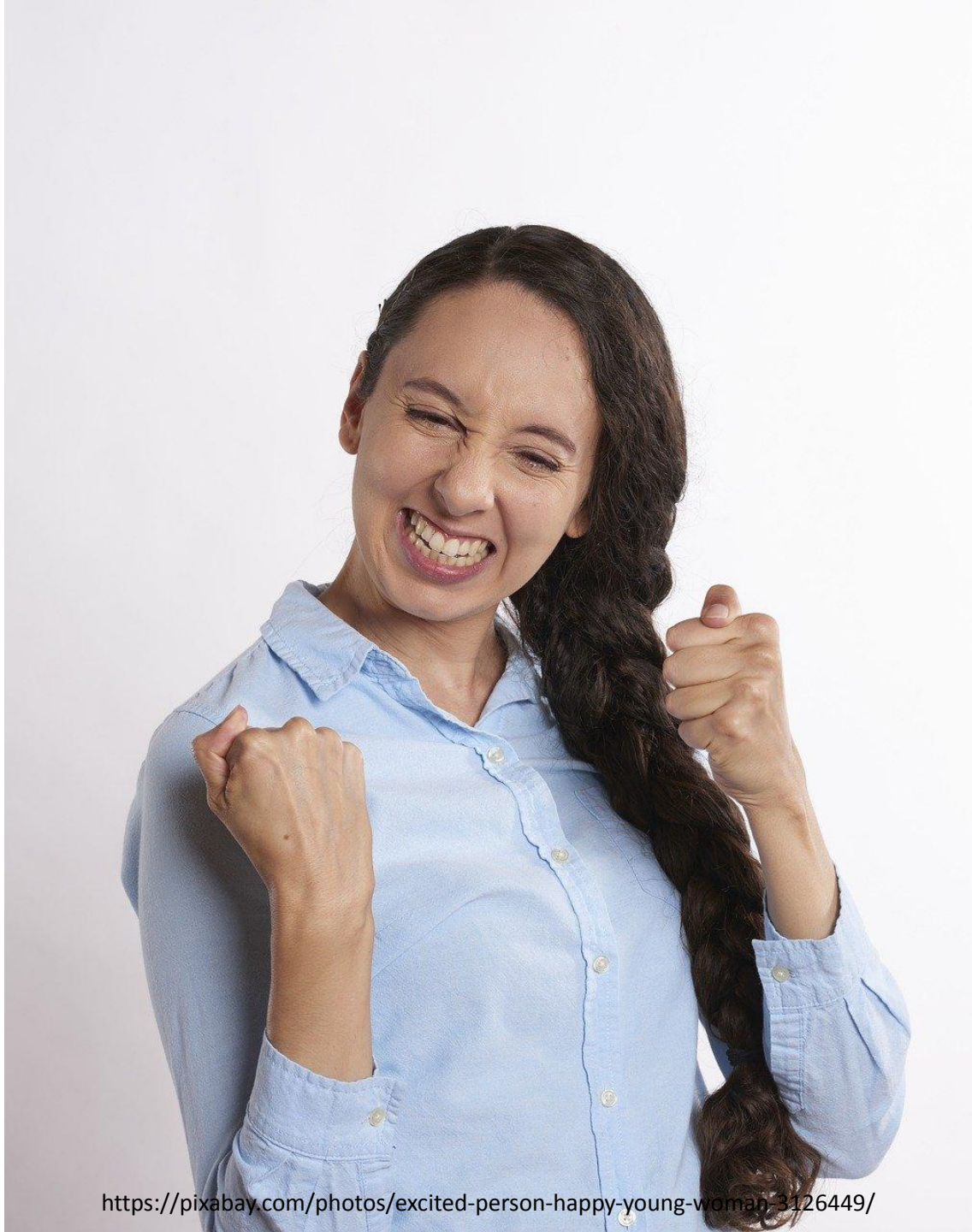
https://pixabay.com/photos/thinking-person-person-thinking-2681494/

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!

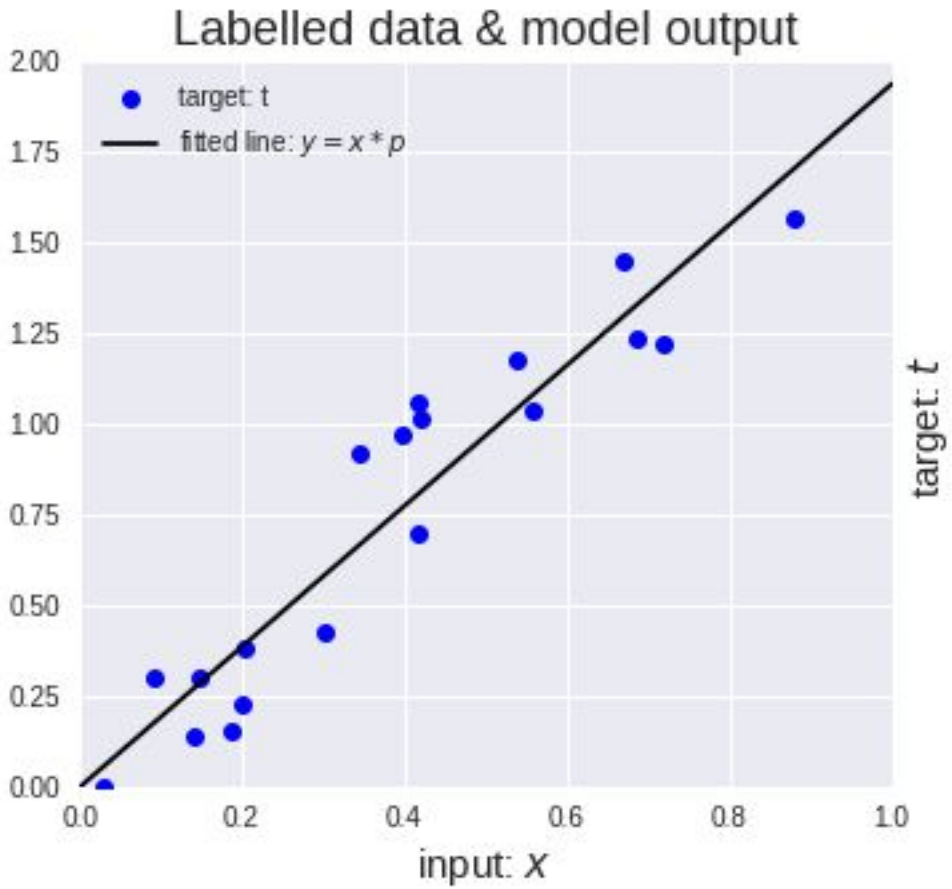
$Y = 2X - 1$

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

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

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

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

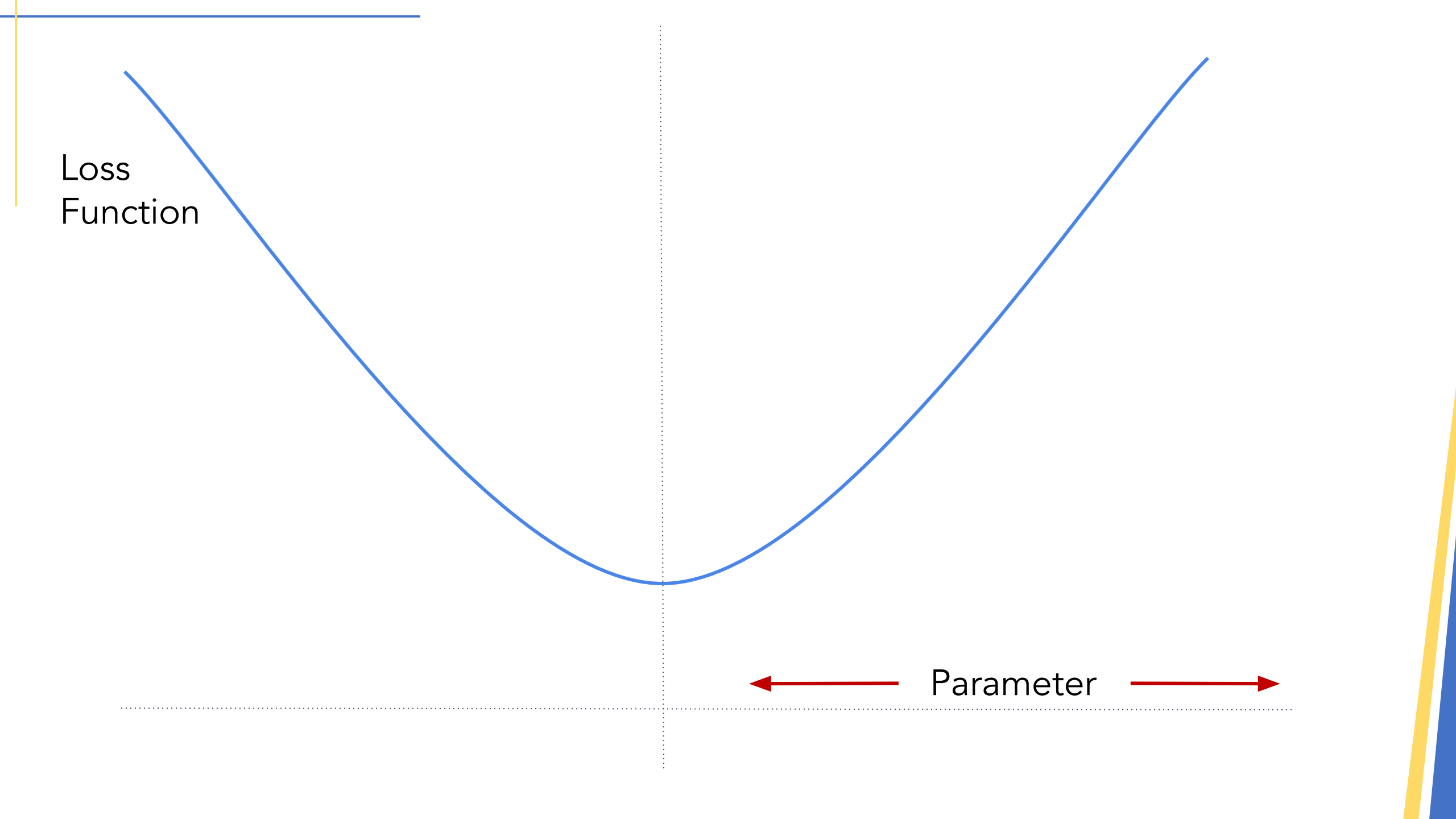


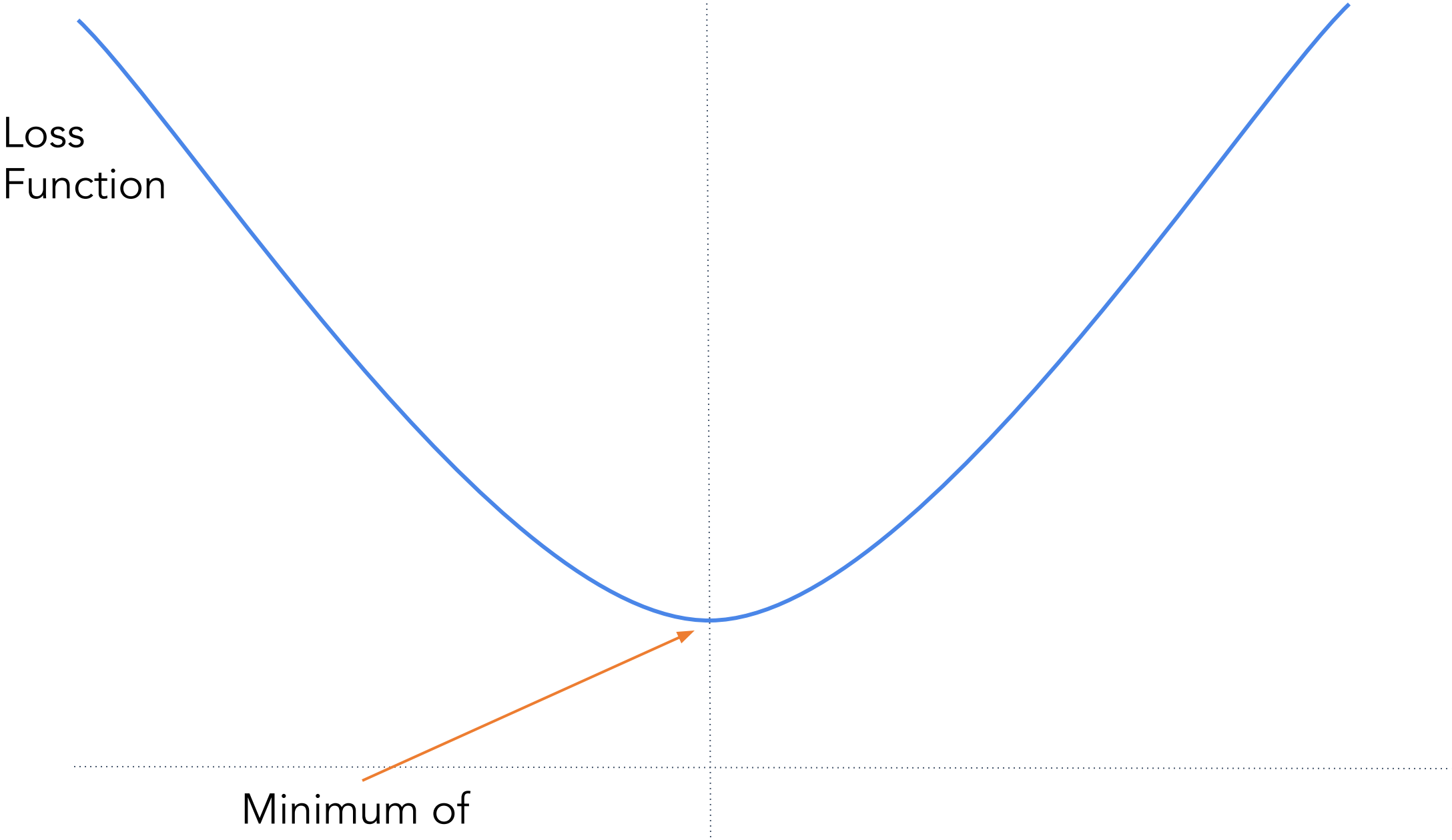
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

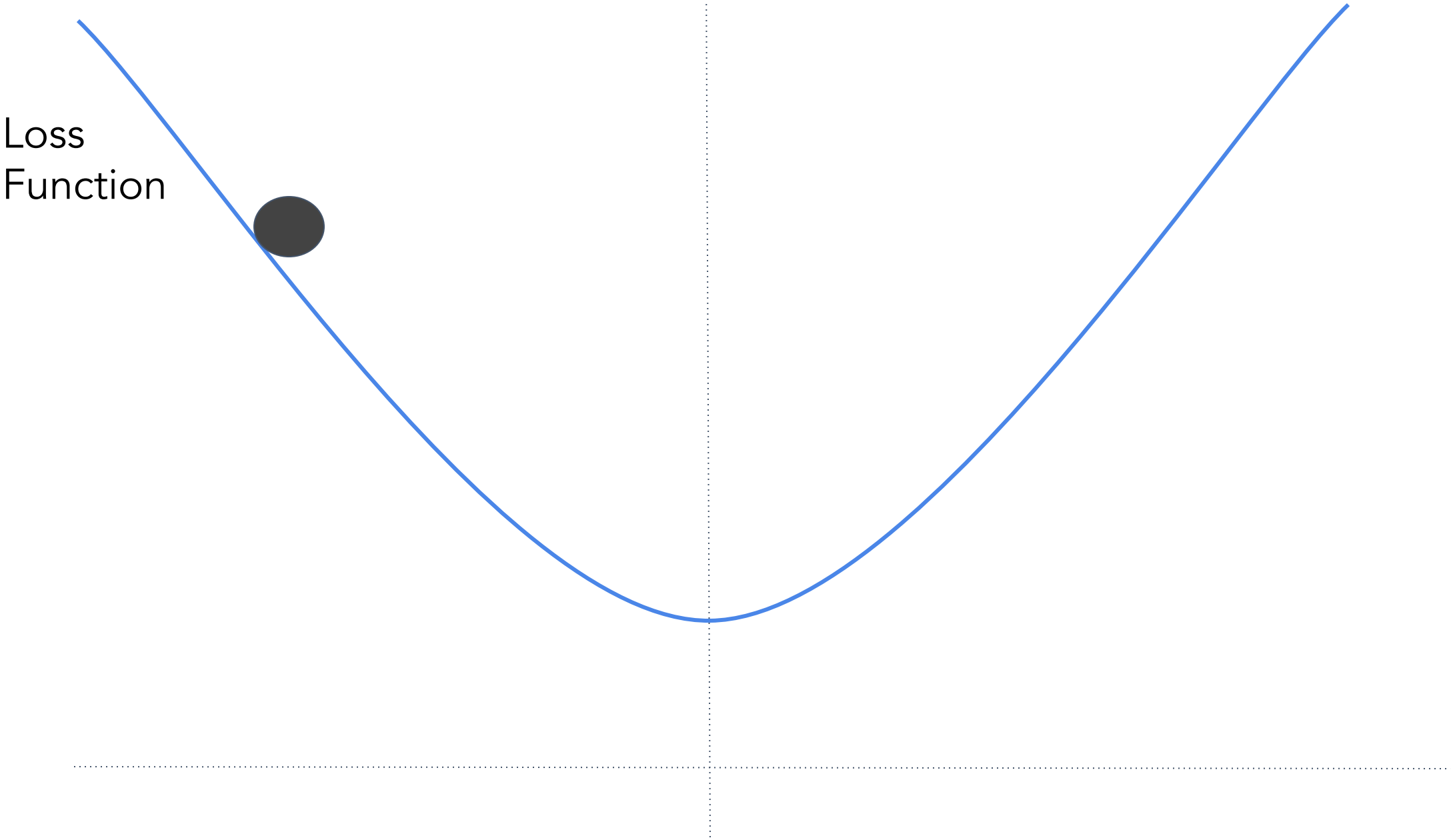
Goal is to Minimize MSE (Mean Squared Error)

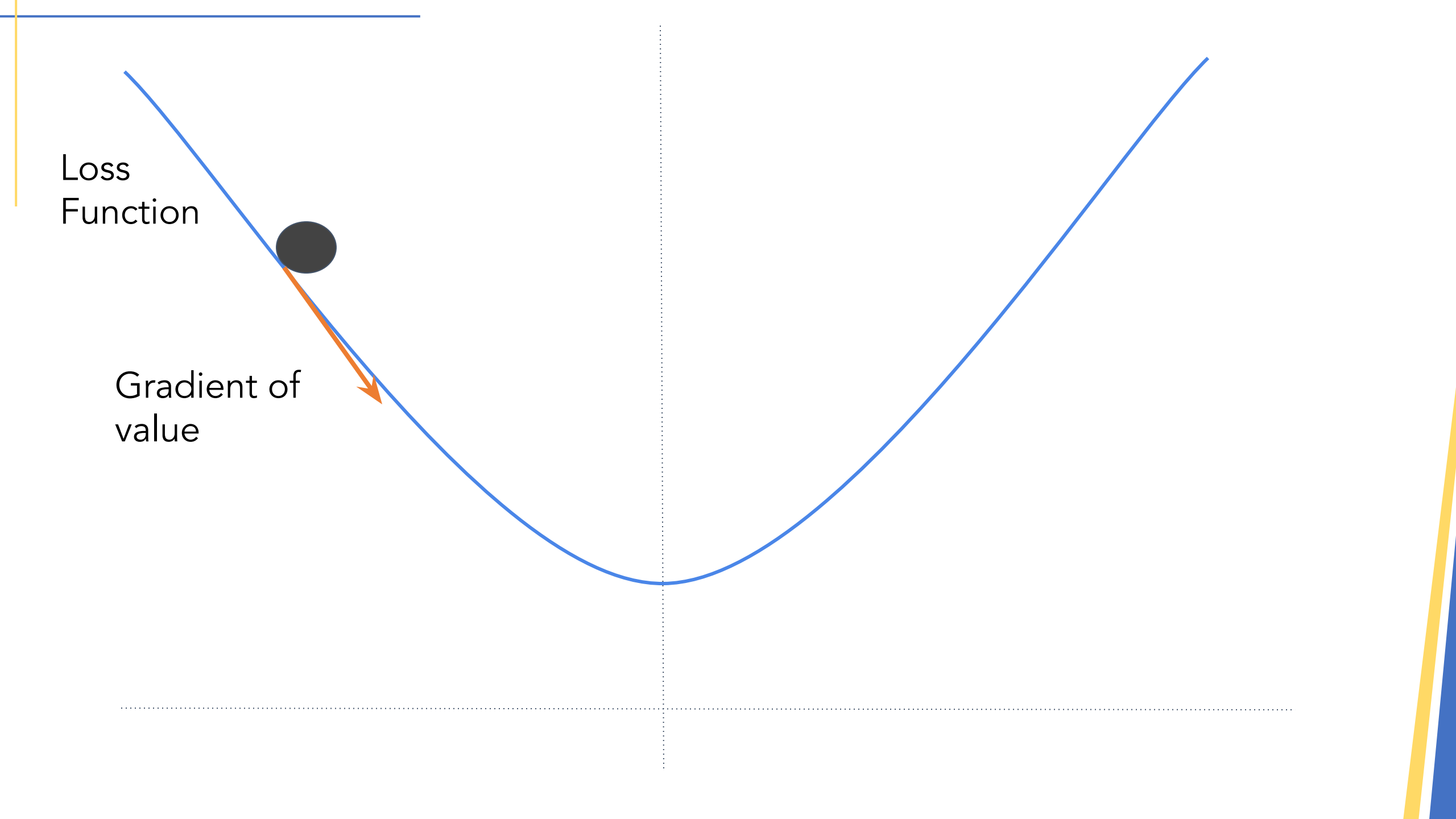
Finding out the best solution

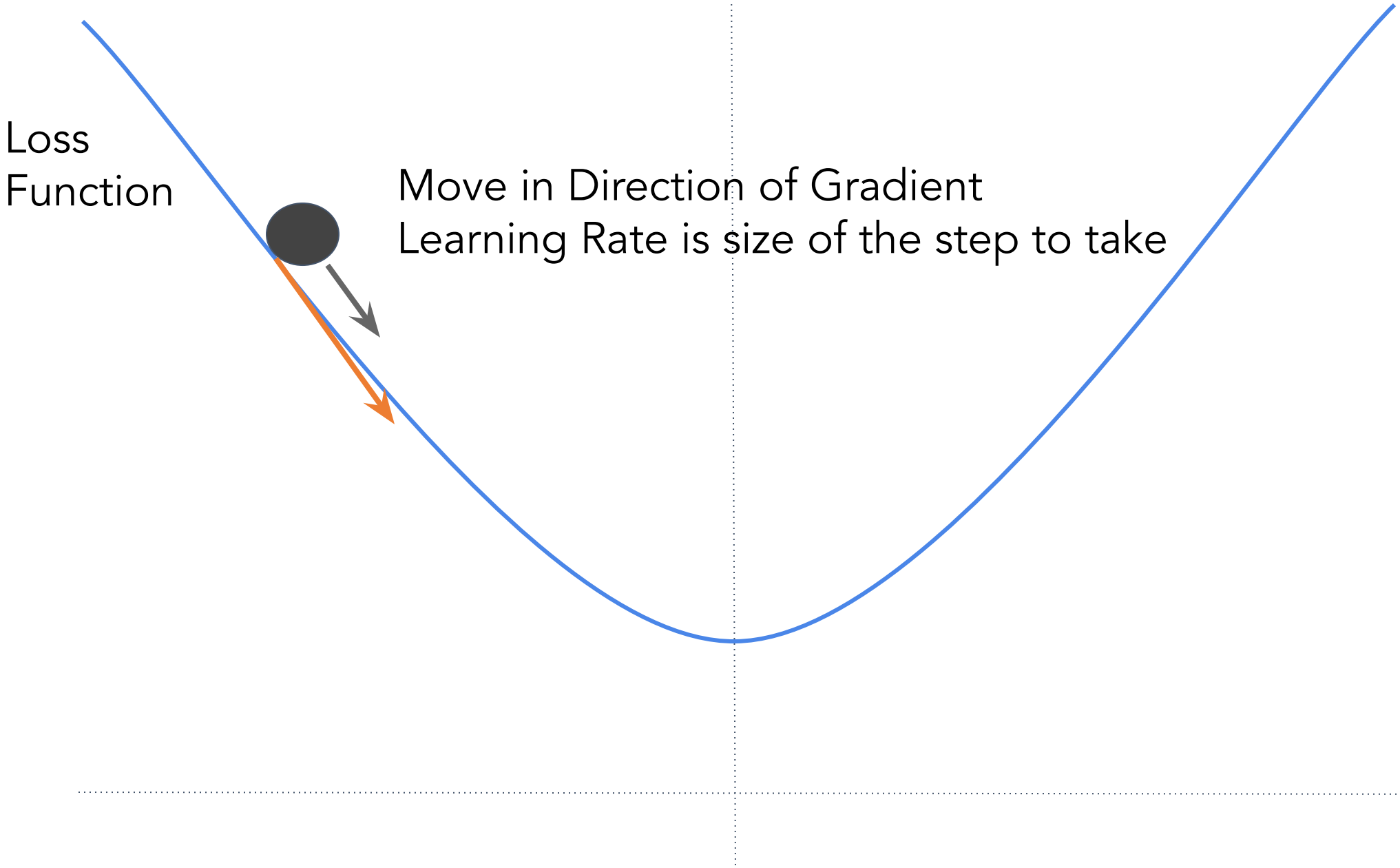
Trial and error approach

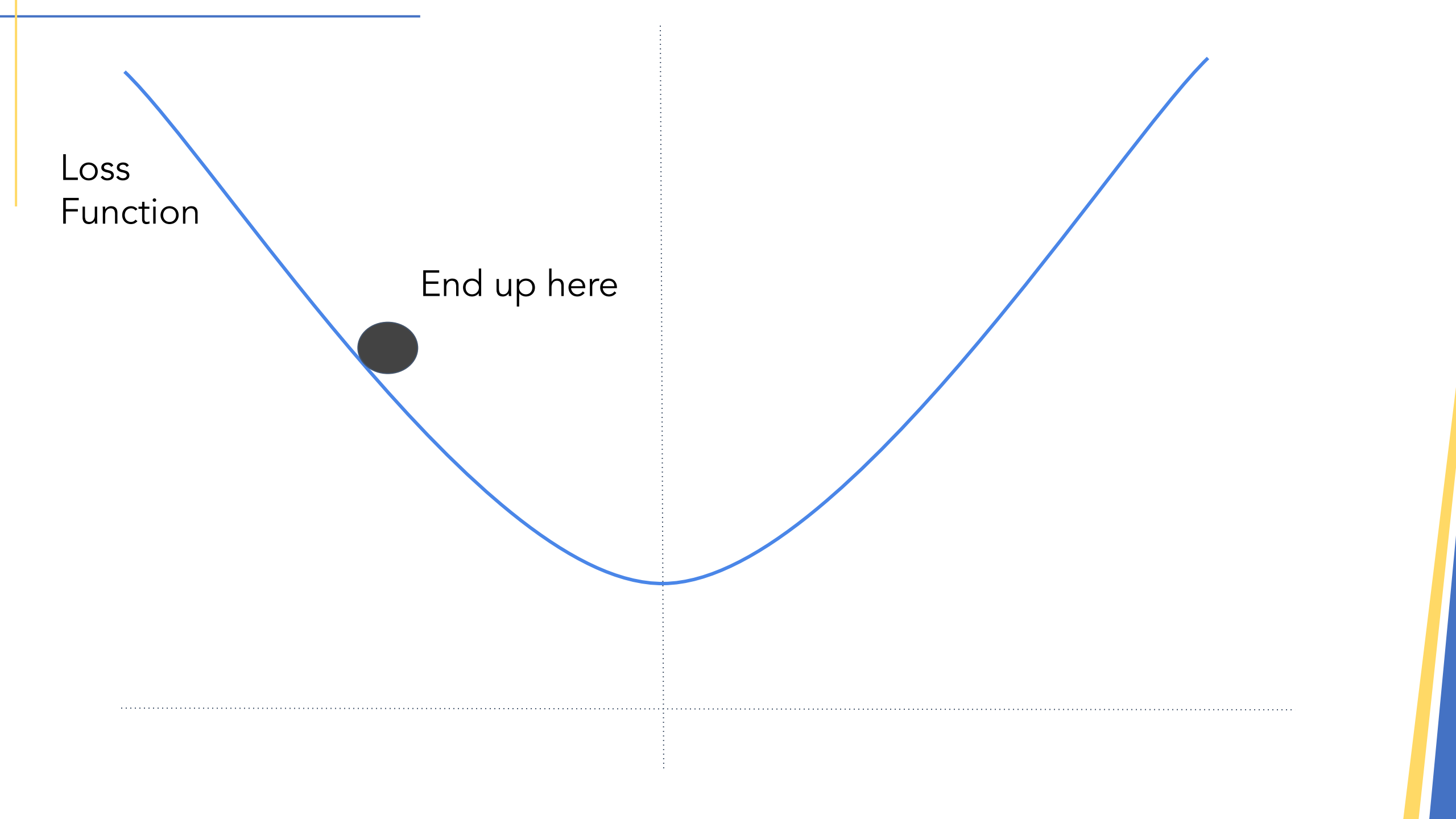

Loss
Function
Parameter

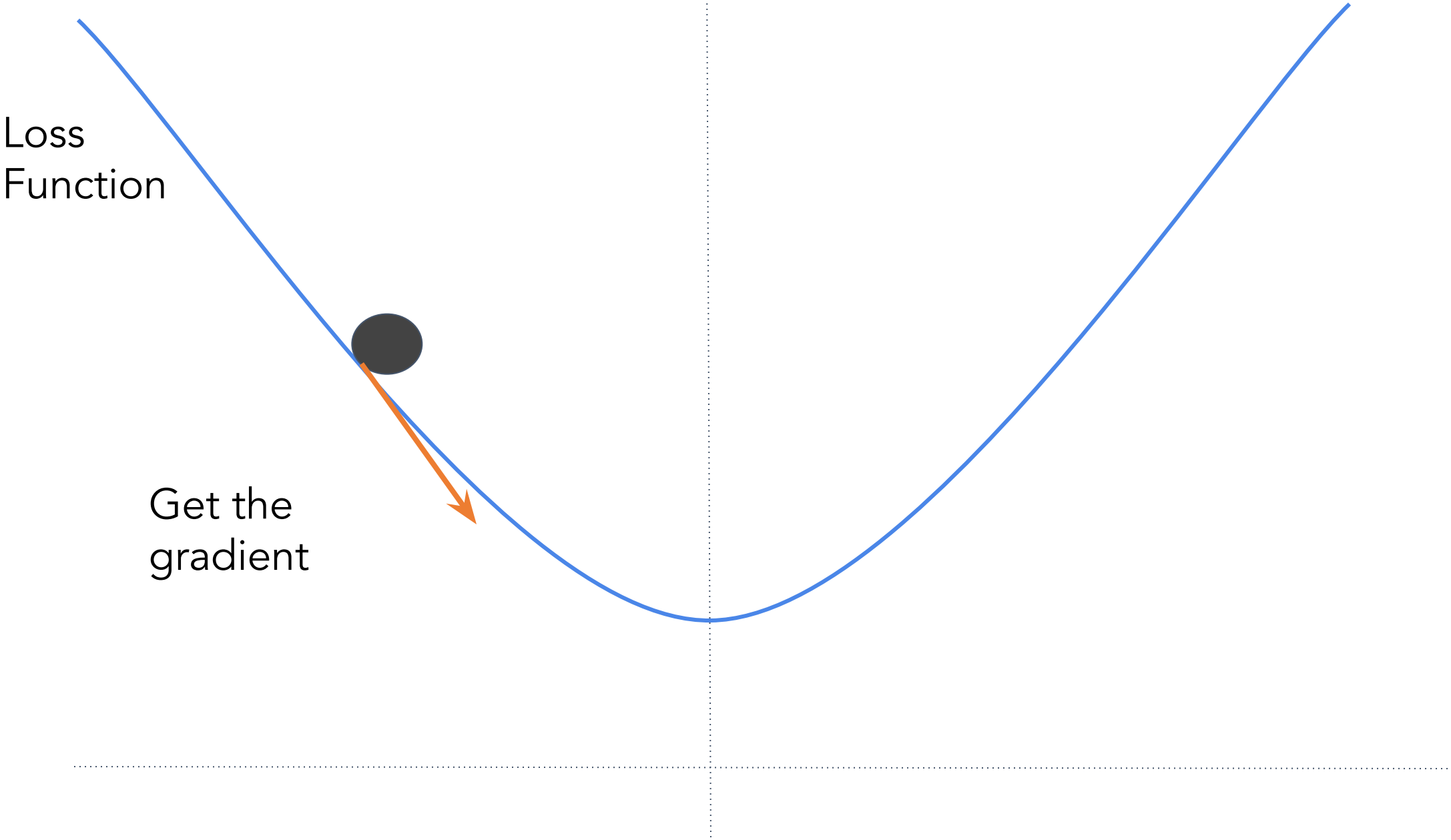

Loss
Function
Minimum of
Loss Function

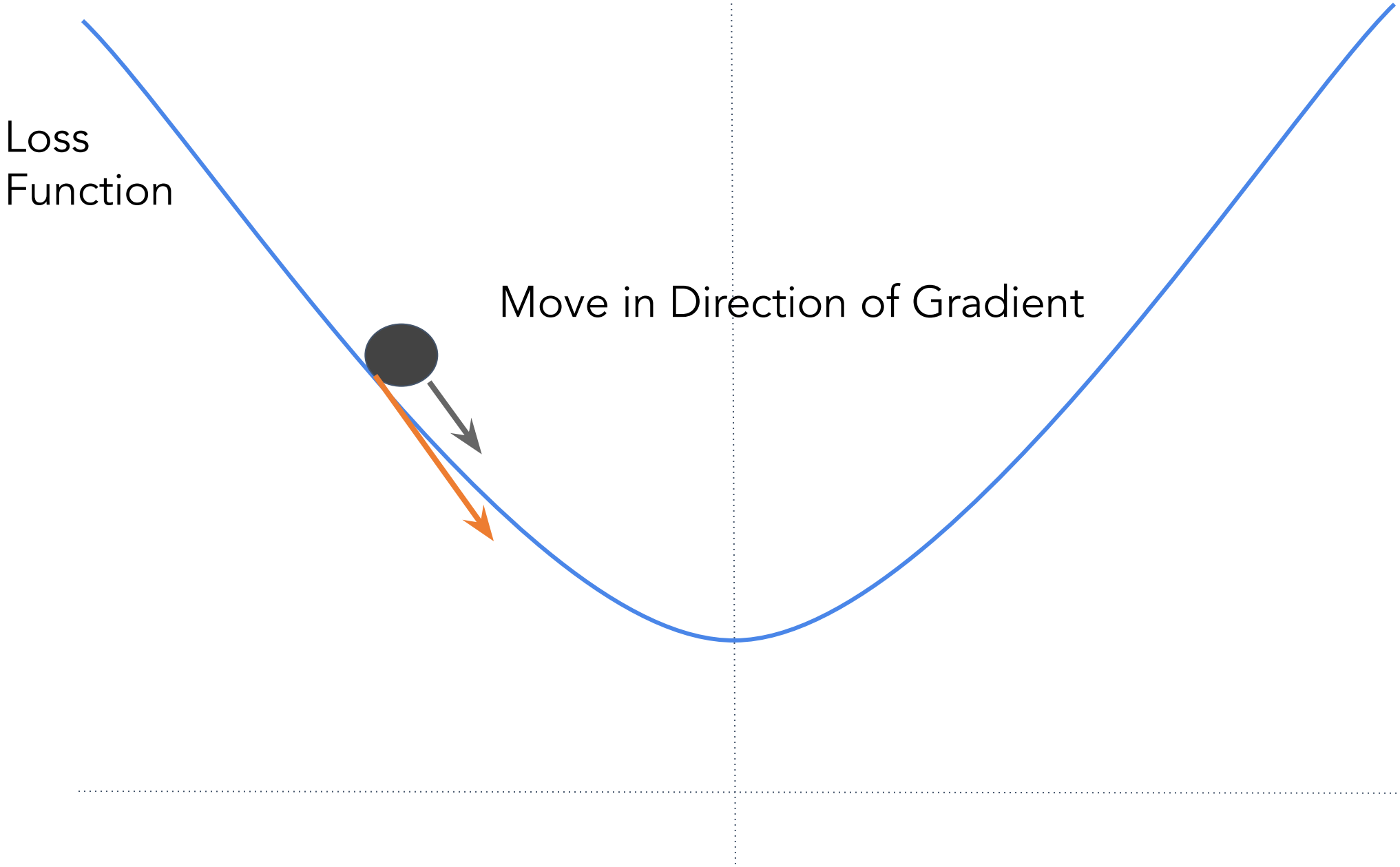

Loss
Function

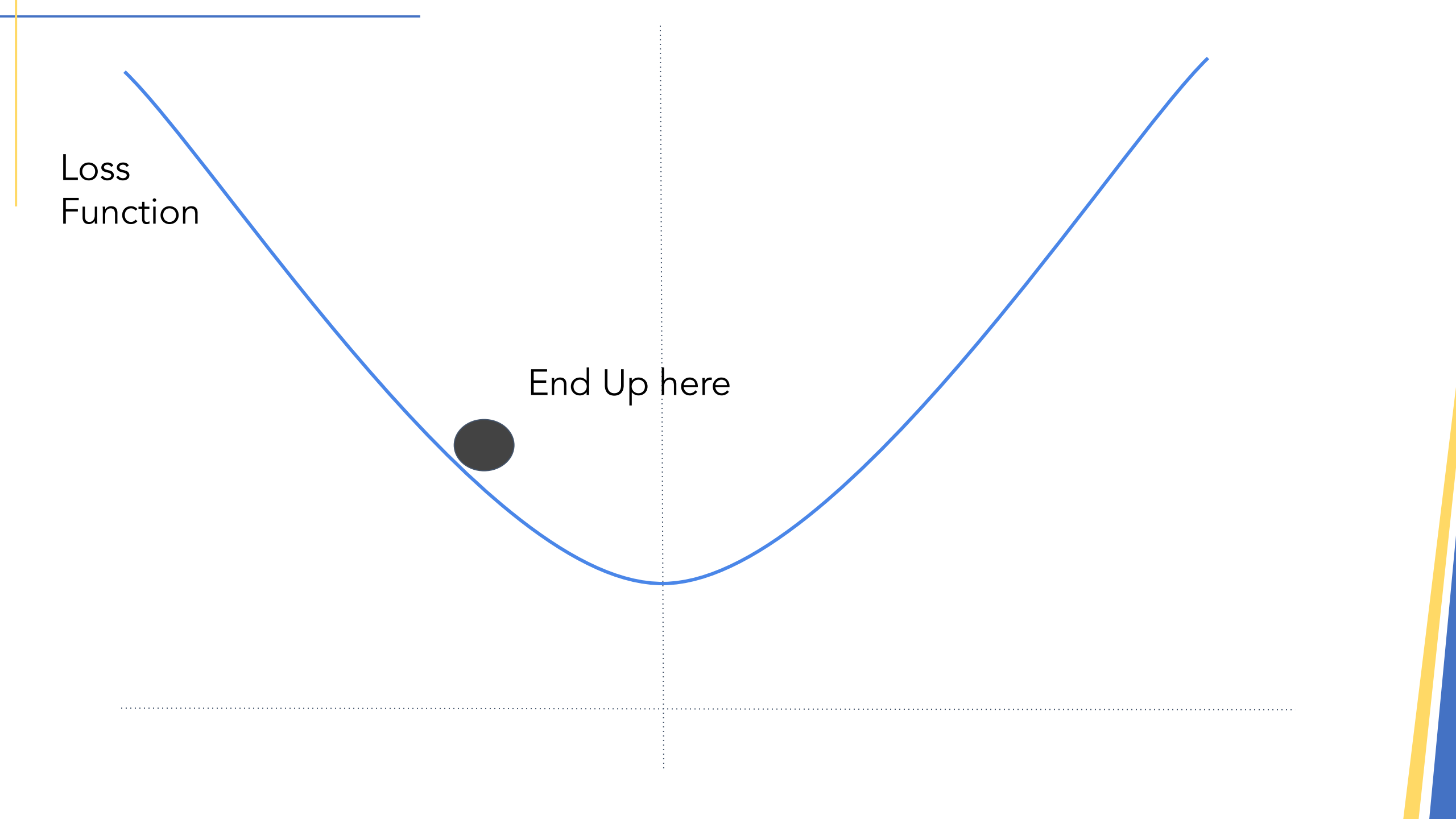

Loss
Function
Gradient of
value


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

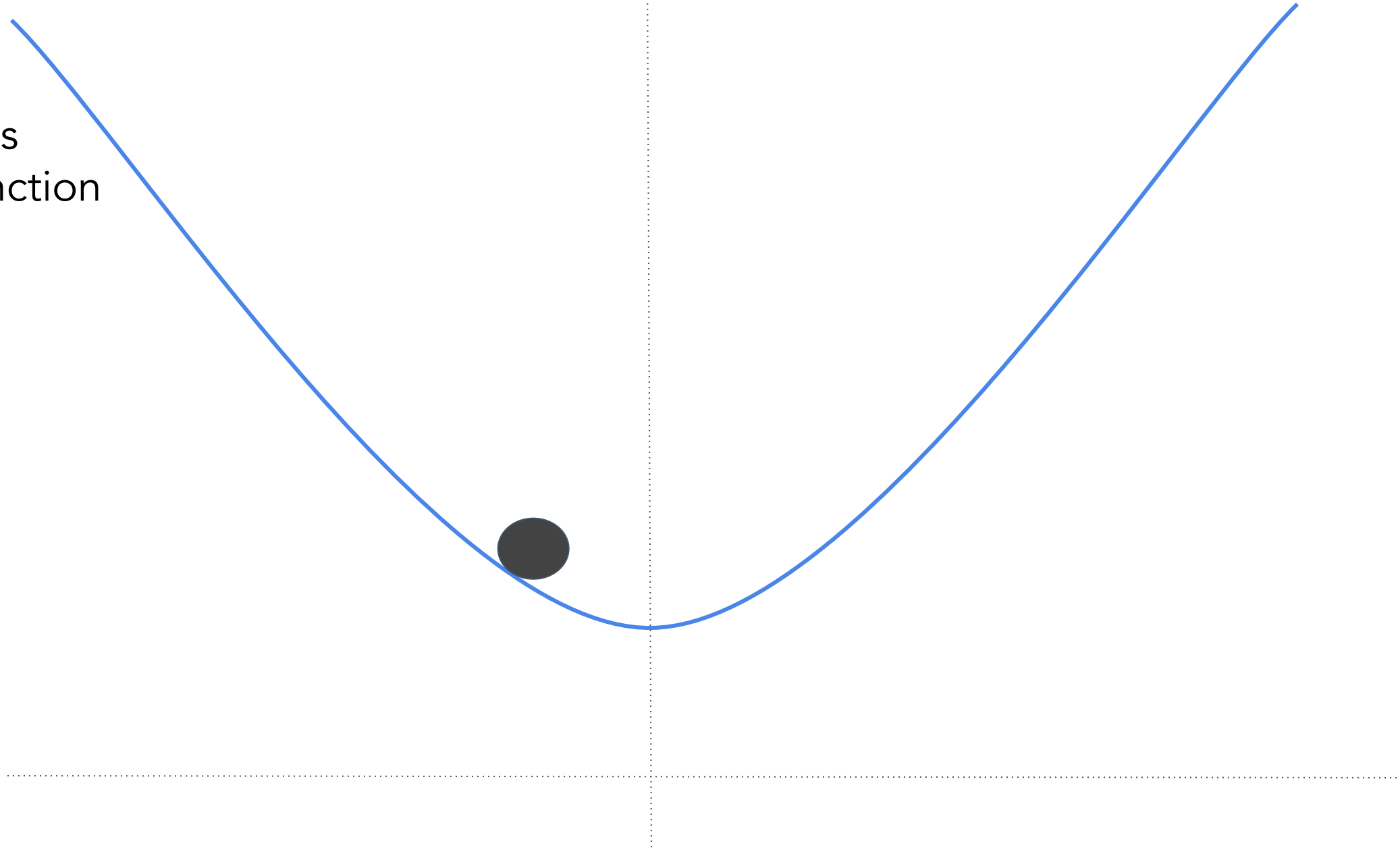

Loss
Function
End up here


Loss
Function
Get the
gradient

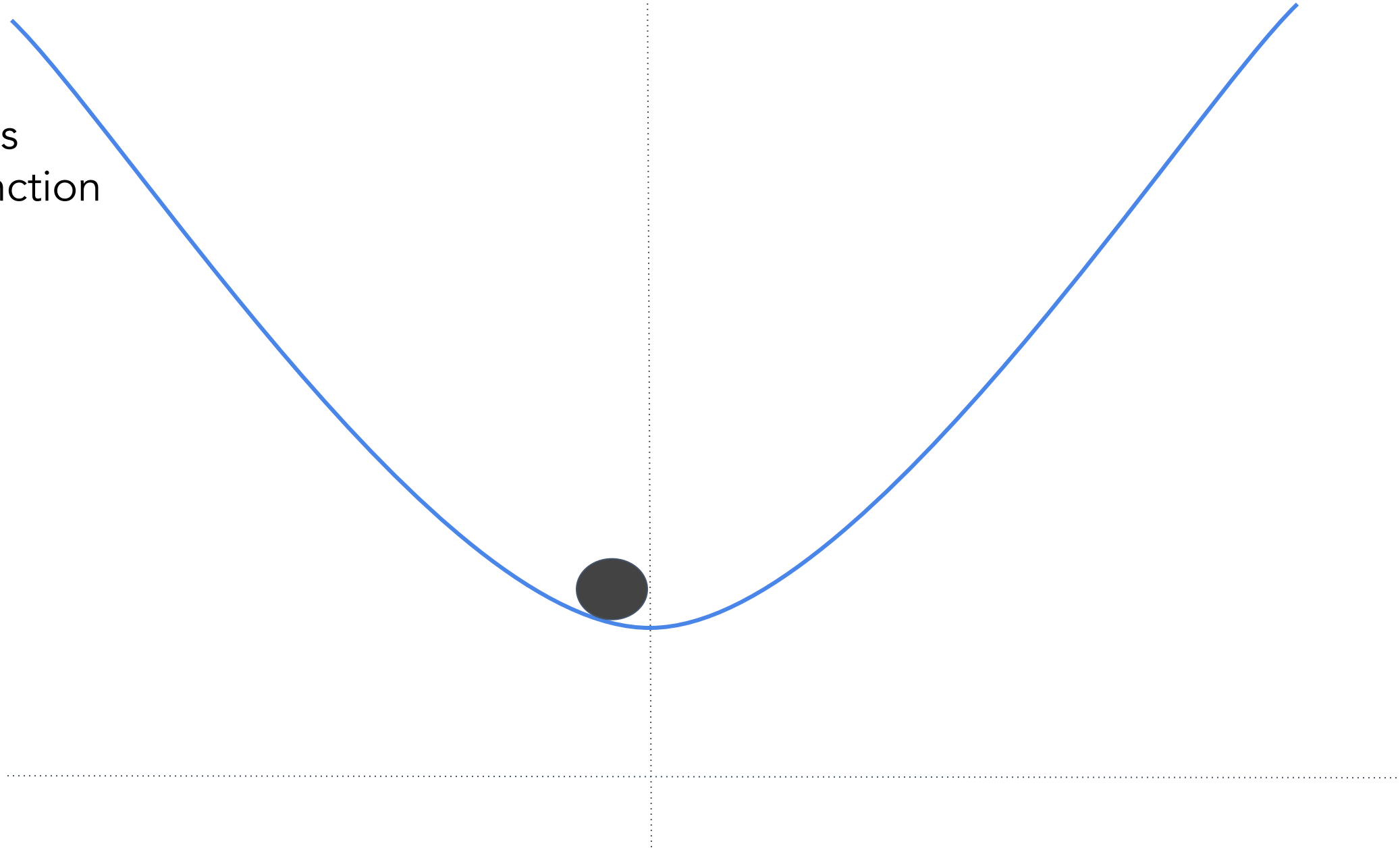

Loss
Function
Move in Direction of Gradient


Loss
Function
End Up here

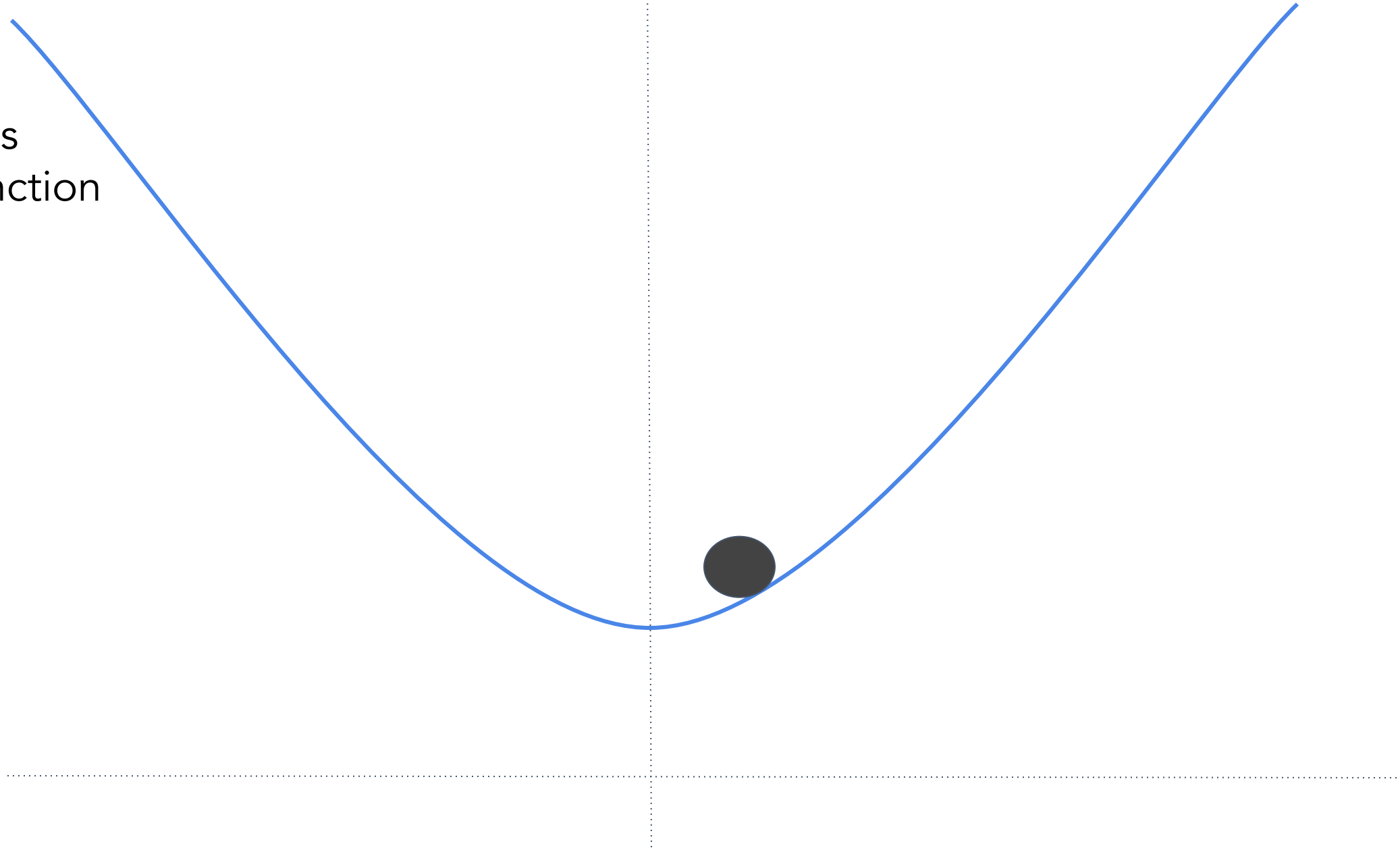
Loss
Function



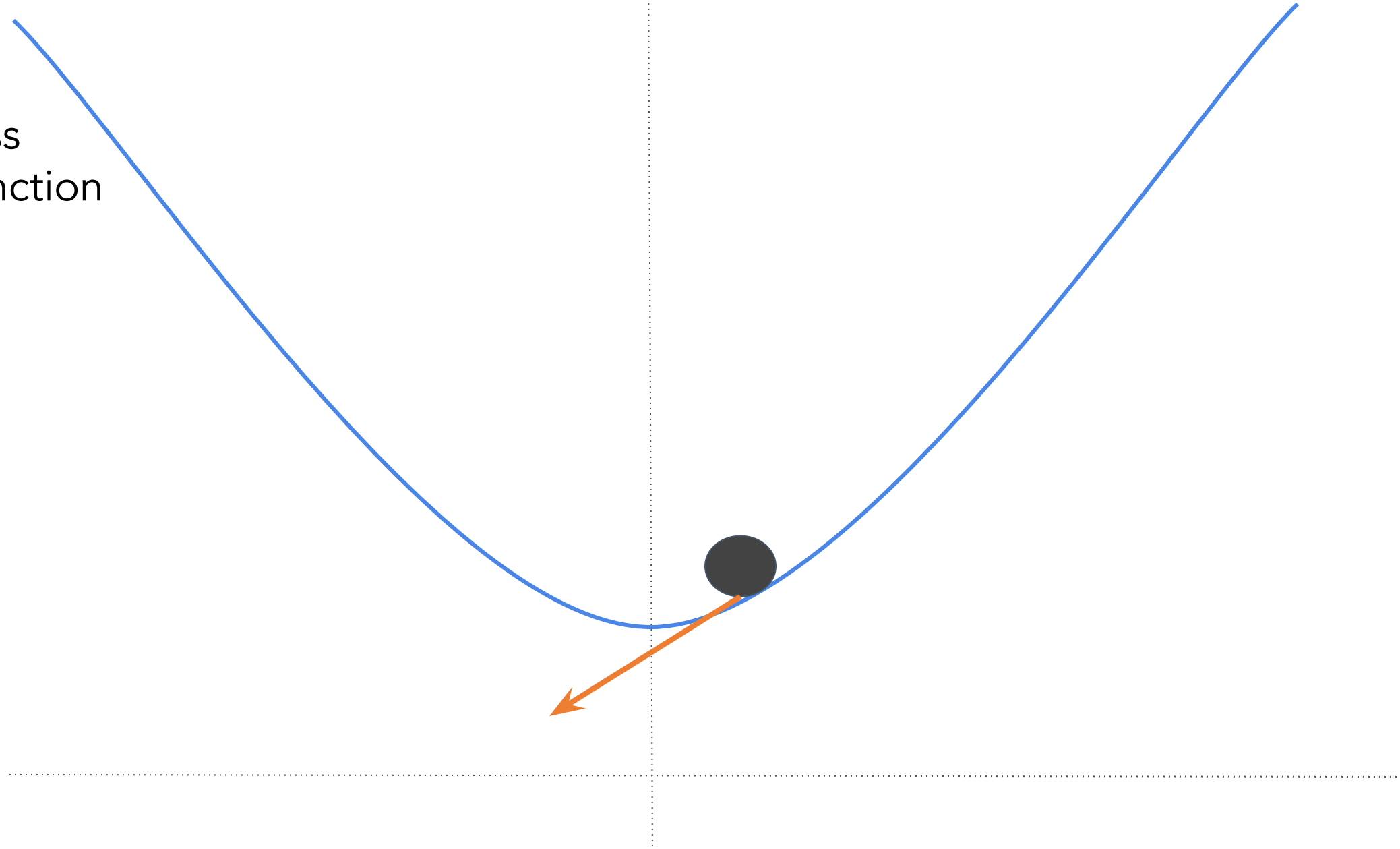
Loss
Function



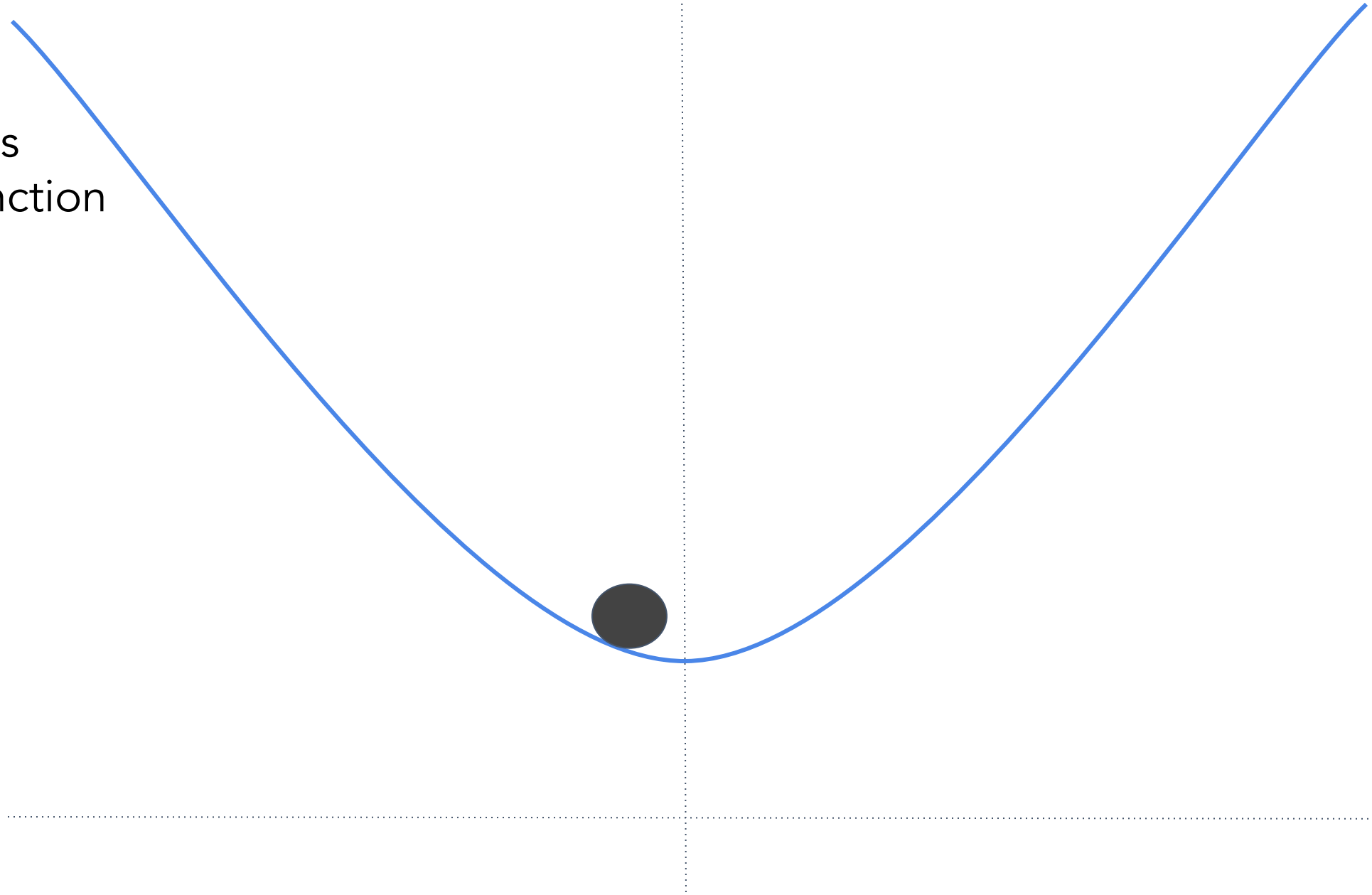
Loss Function



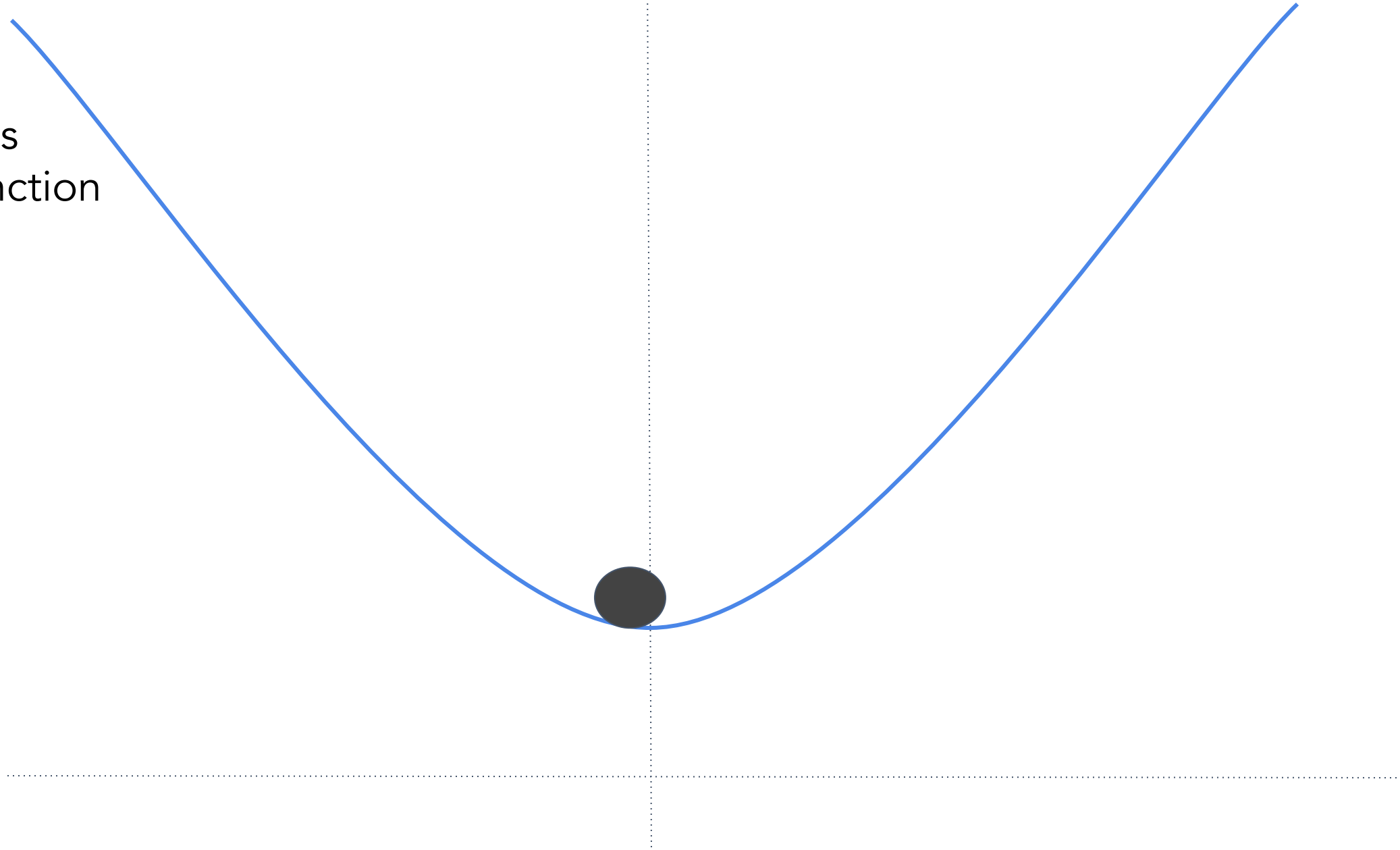
Loss
Function

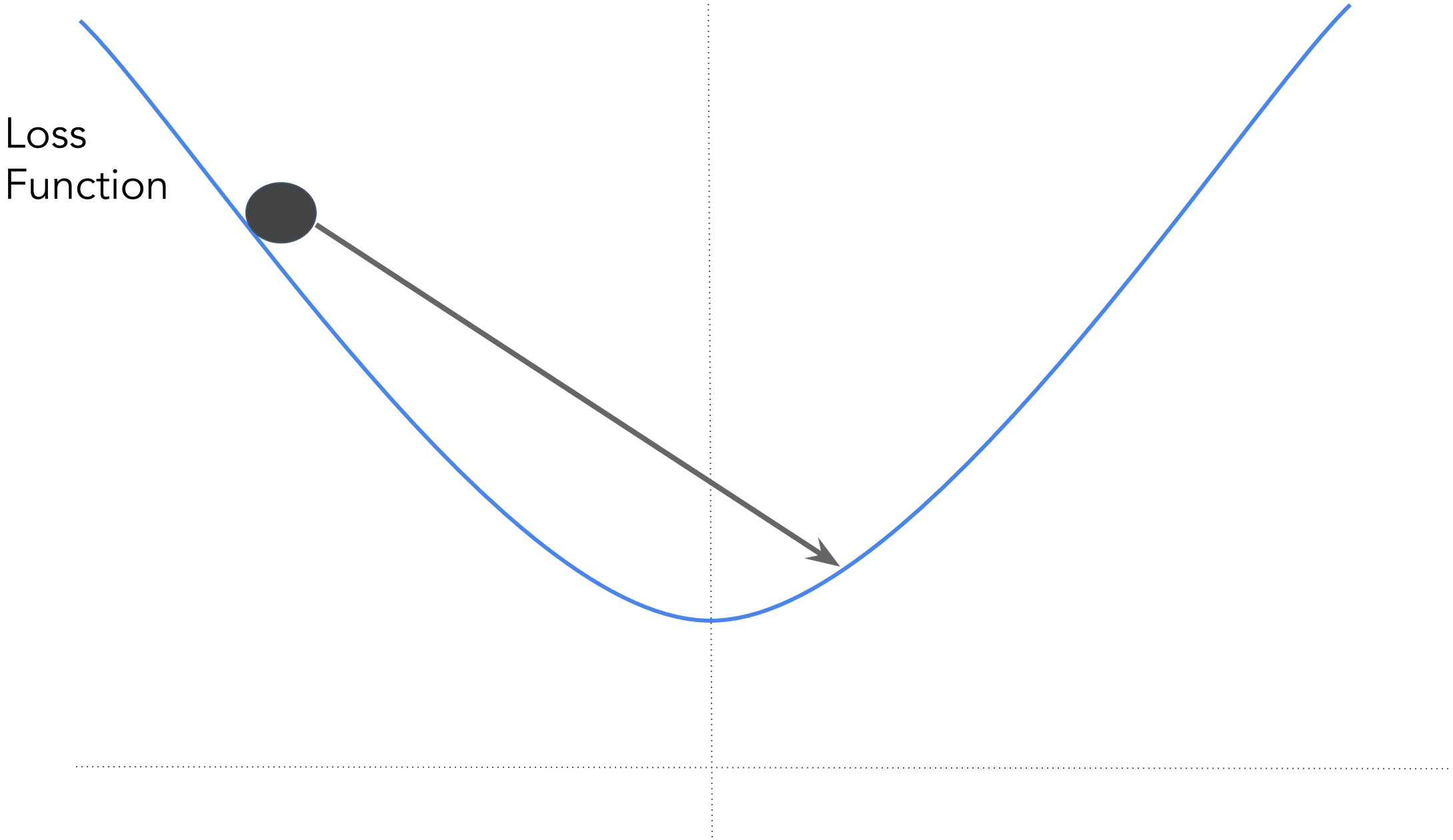


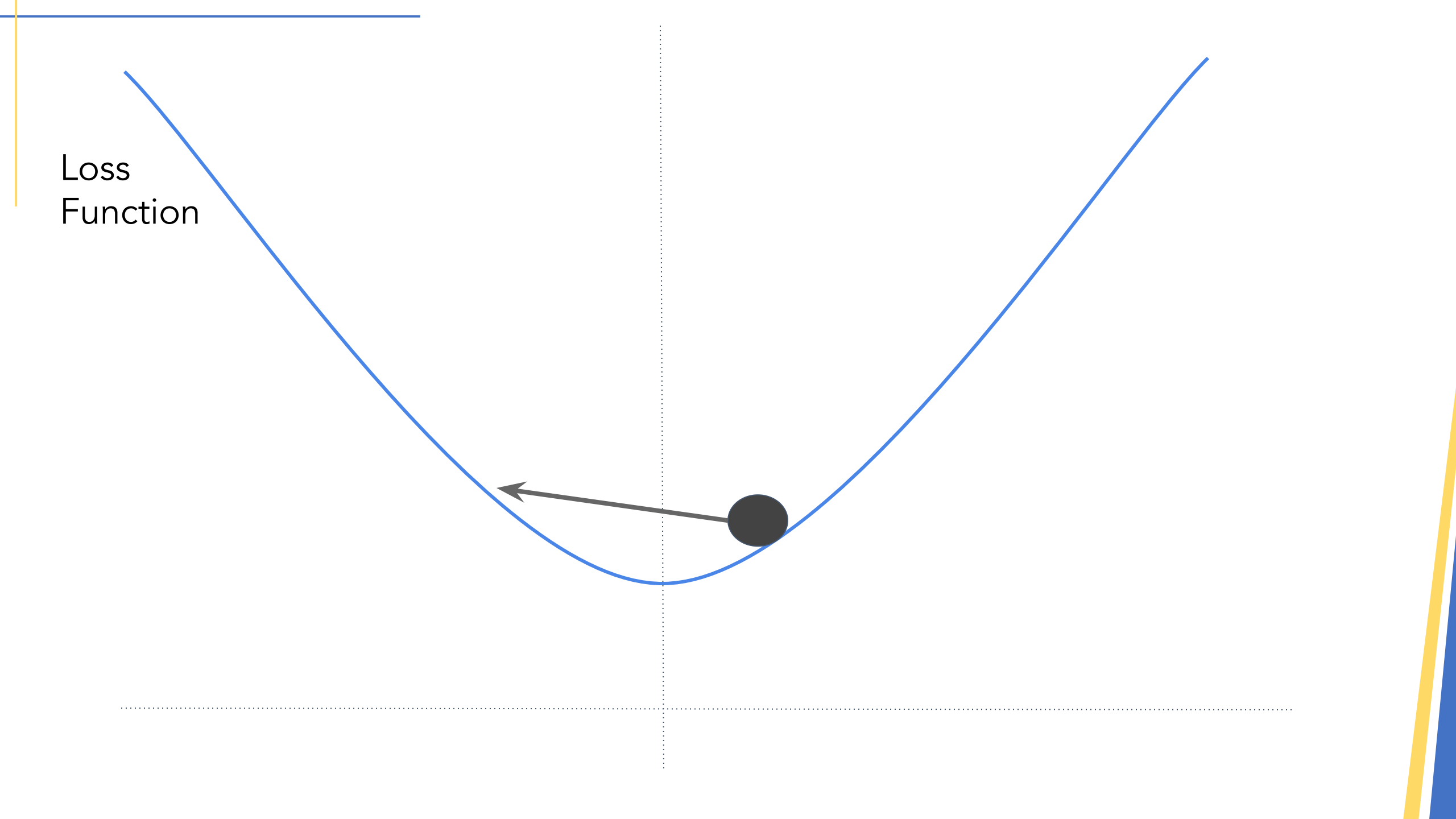
Loss
Function

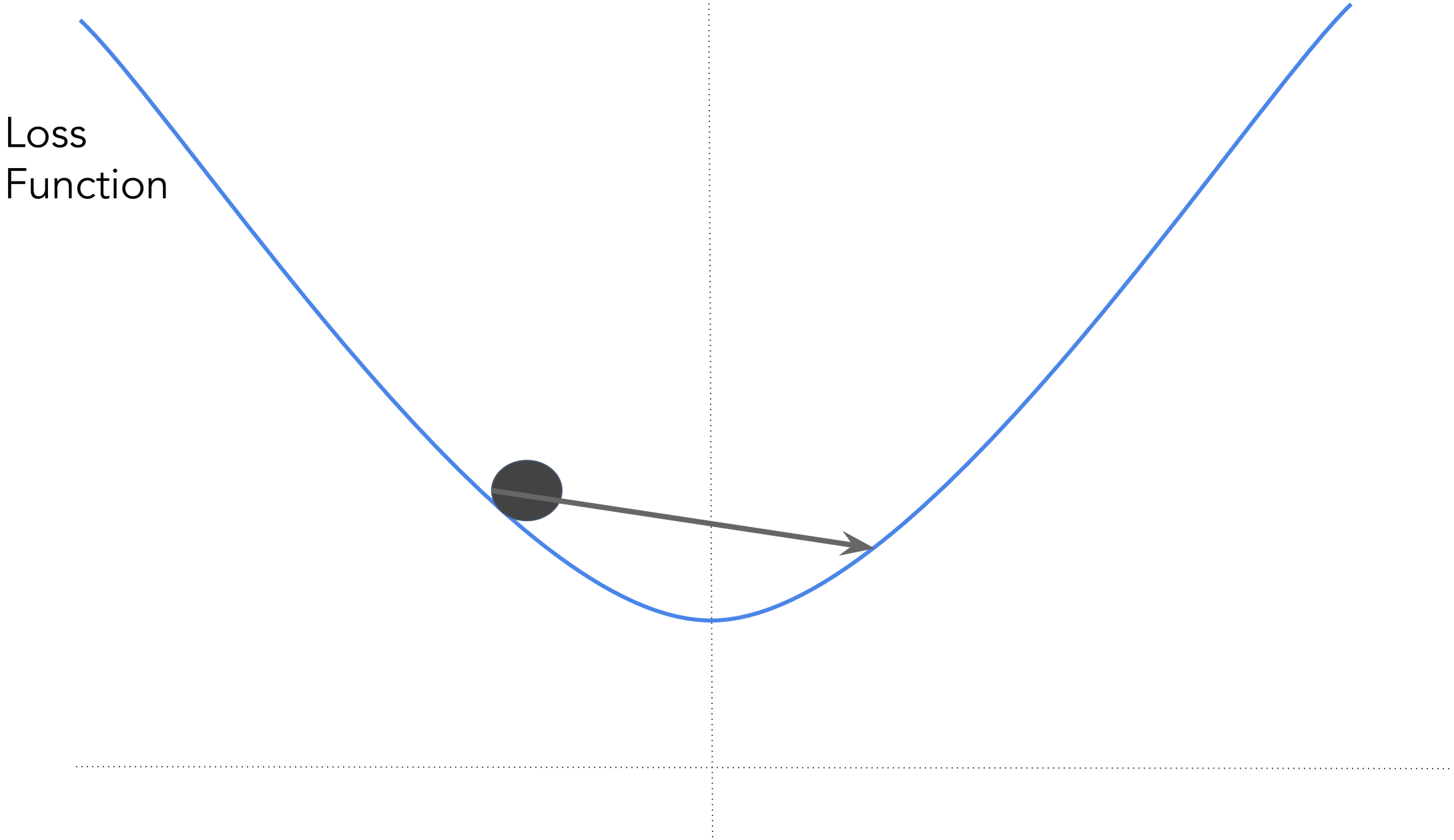


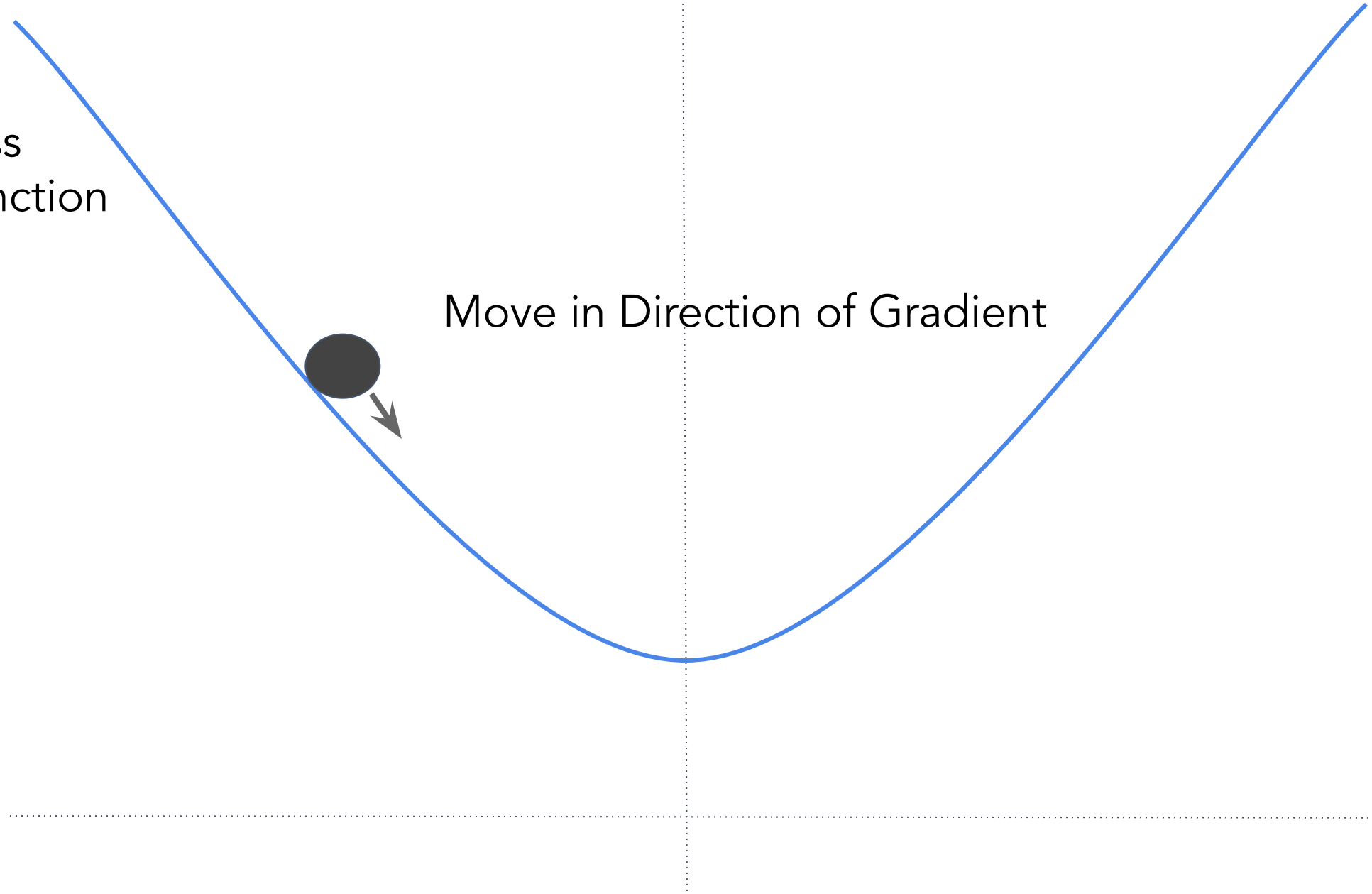
Loss
Function

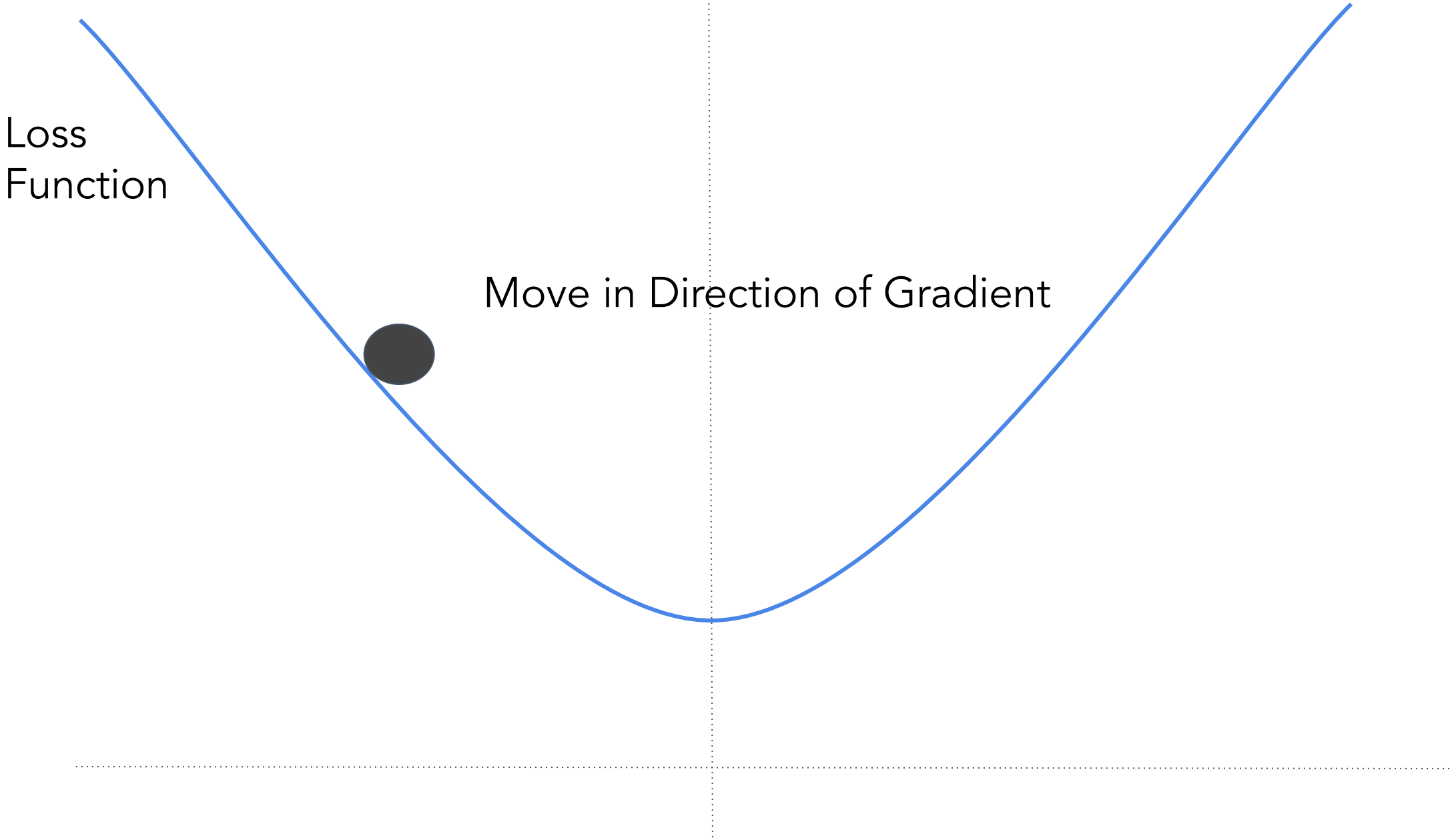


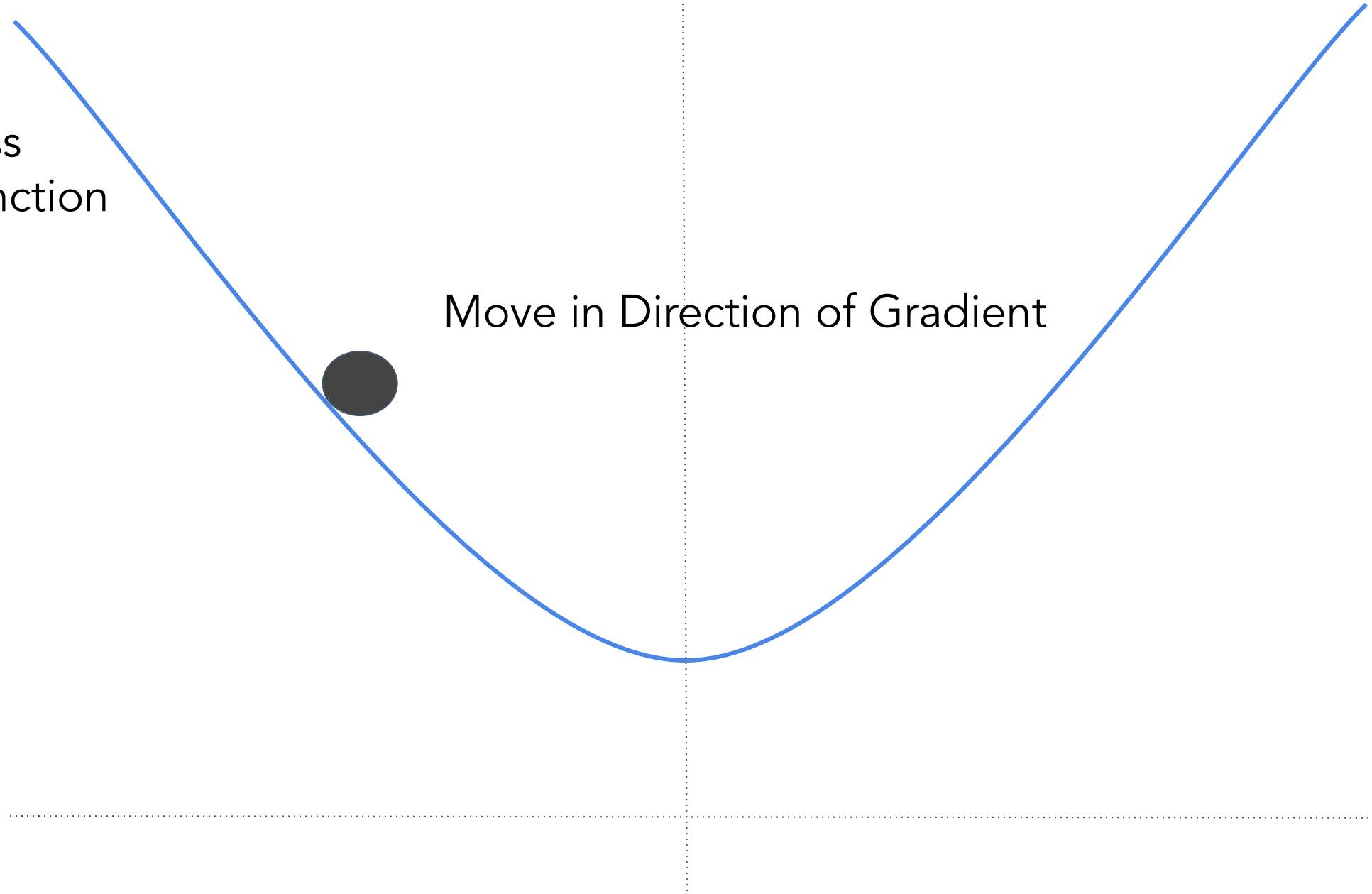

Loss
Function

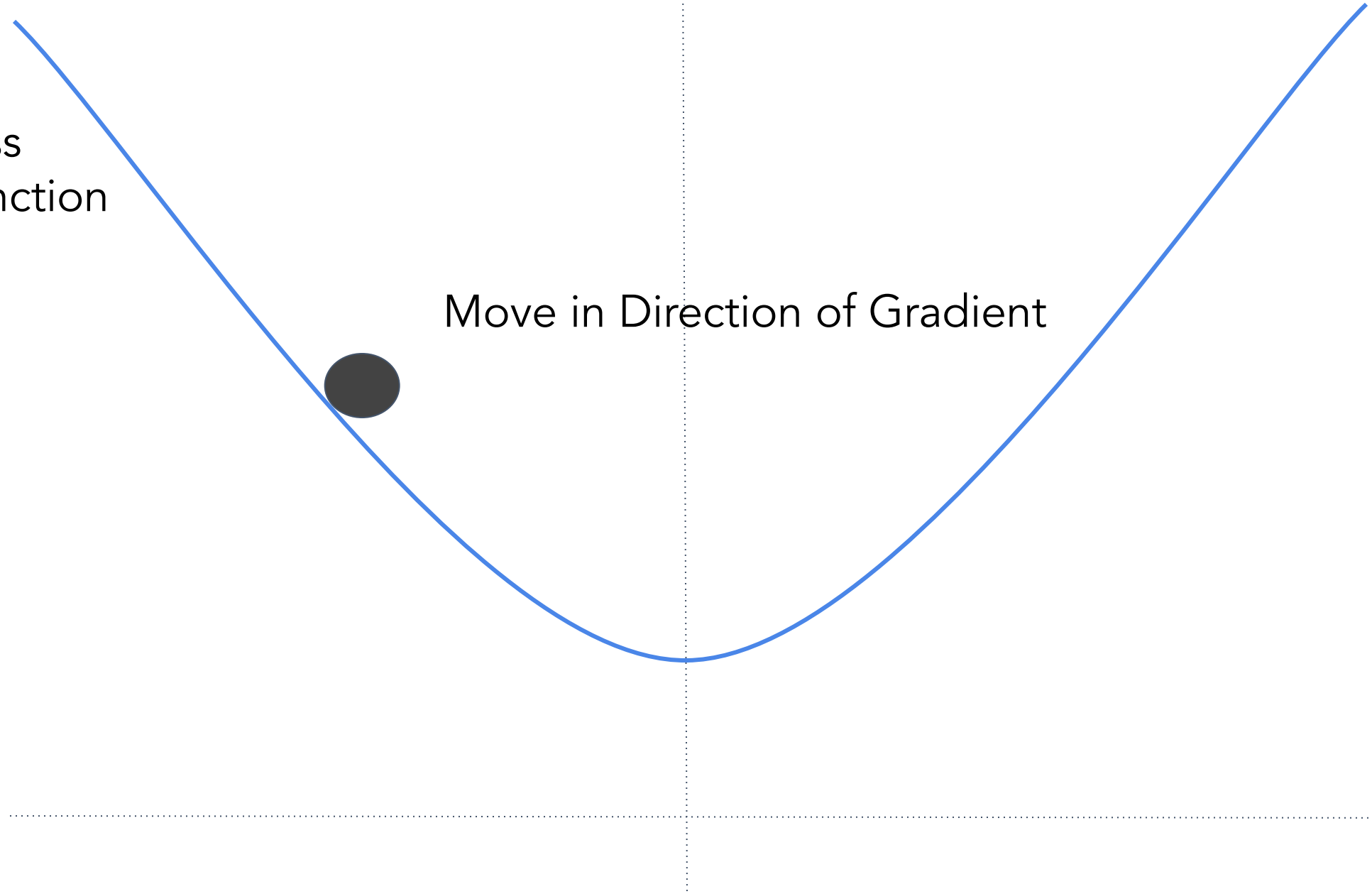

Loss
Function

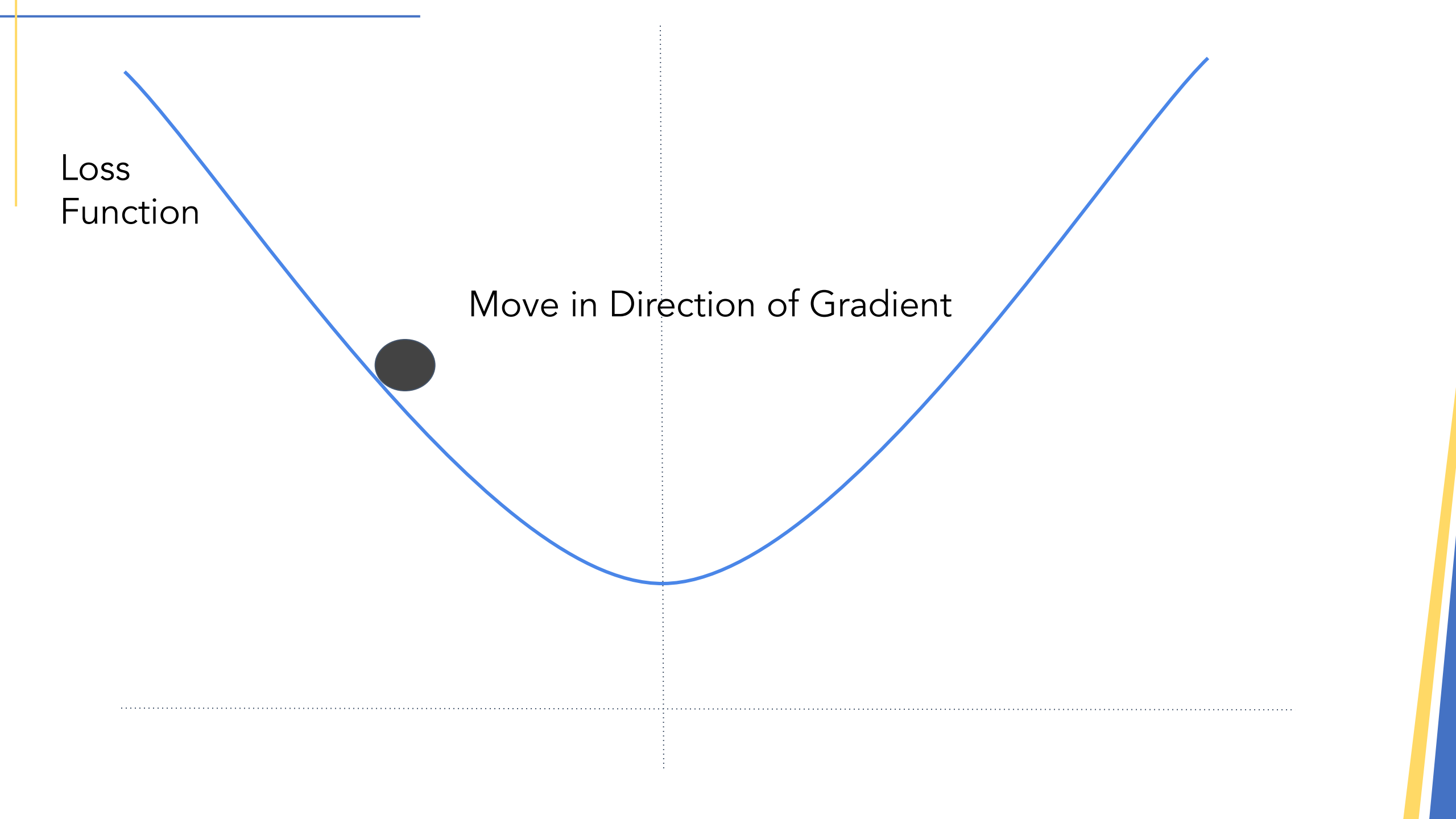

Loss
Function


Loss
Function
Move in Direction of Gradient


Loss
Function
Move in Direction of Gradient

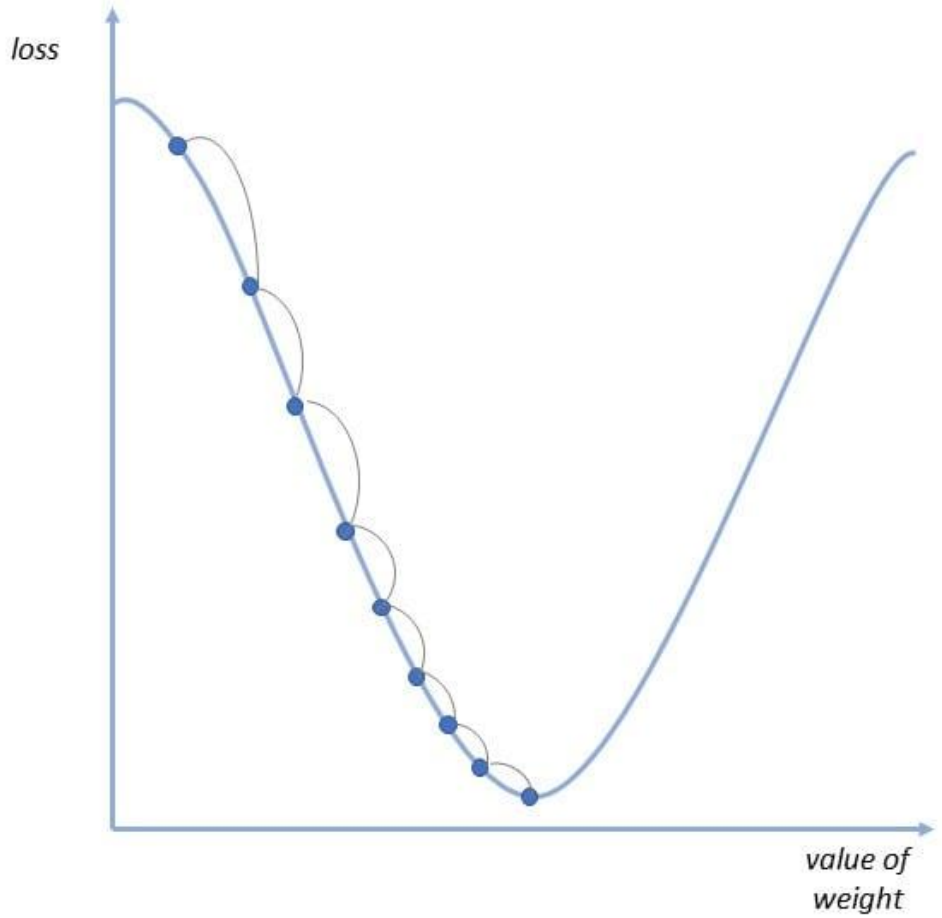

Loss
Function
Move in Direction of Gradient


Loss
Function
Move in Direction of Gradient

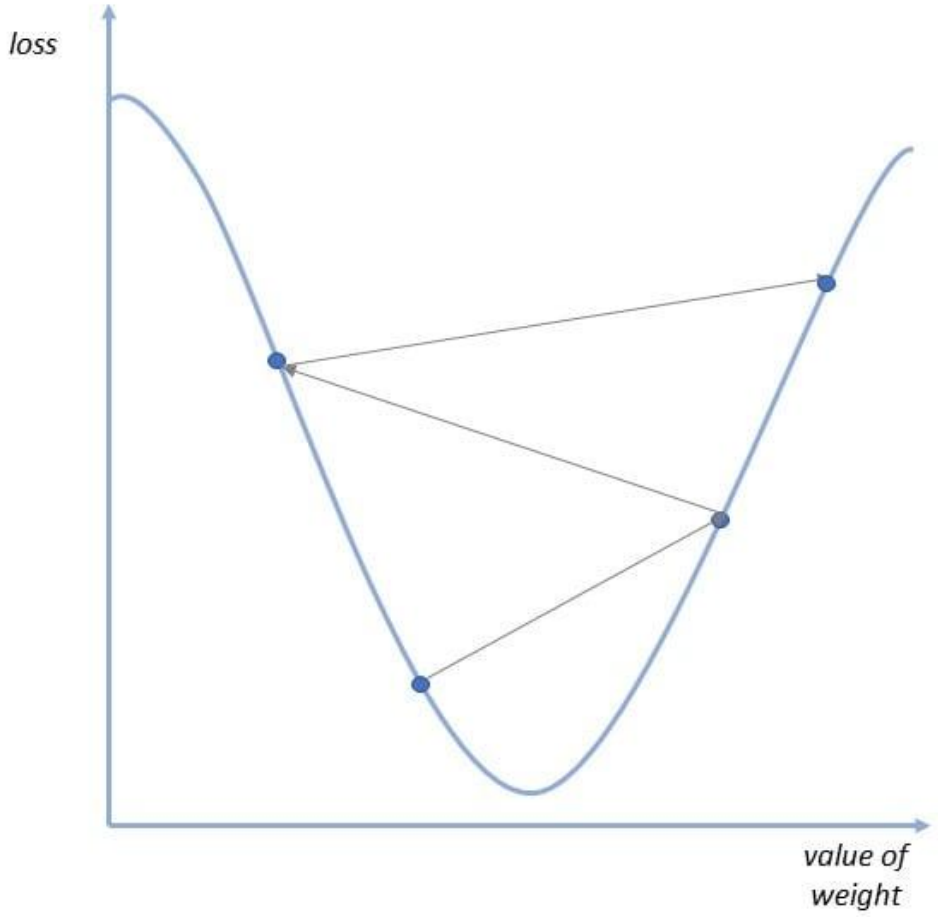

Loss
Function
Move in Direction of Gradient

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

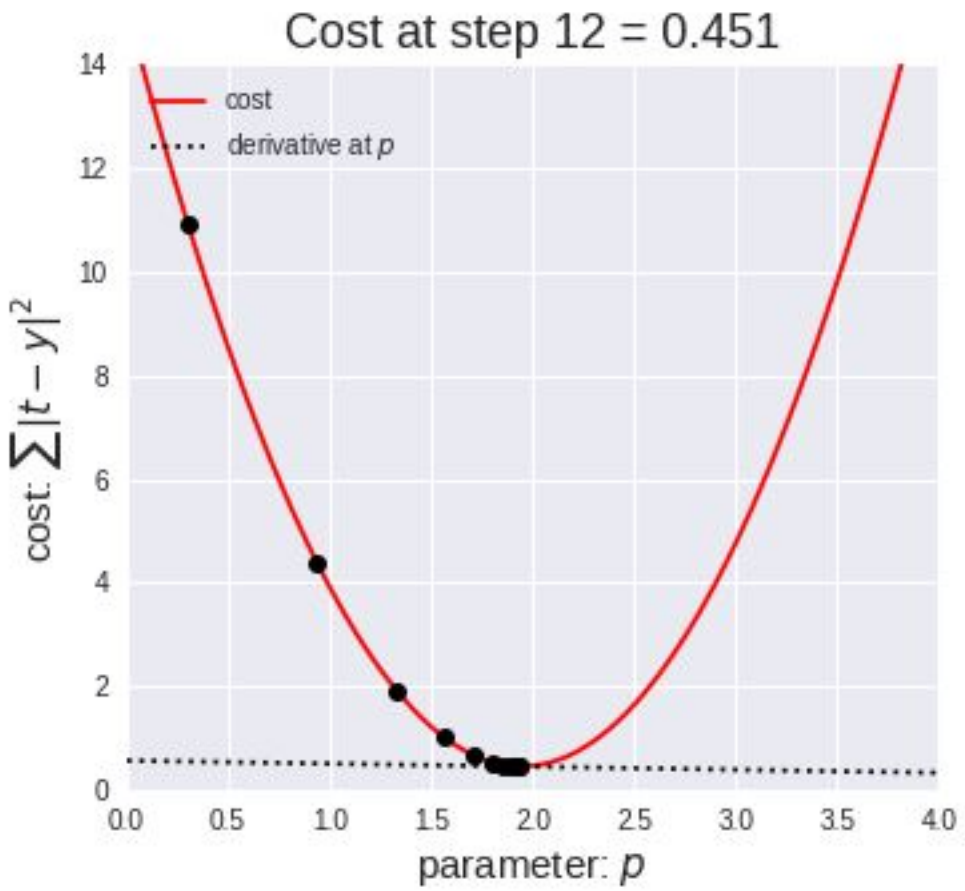
If the Learning Rate is too small it may take a long time to reach the minimum

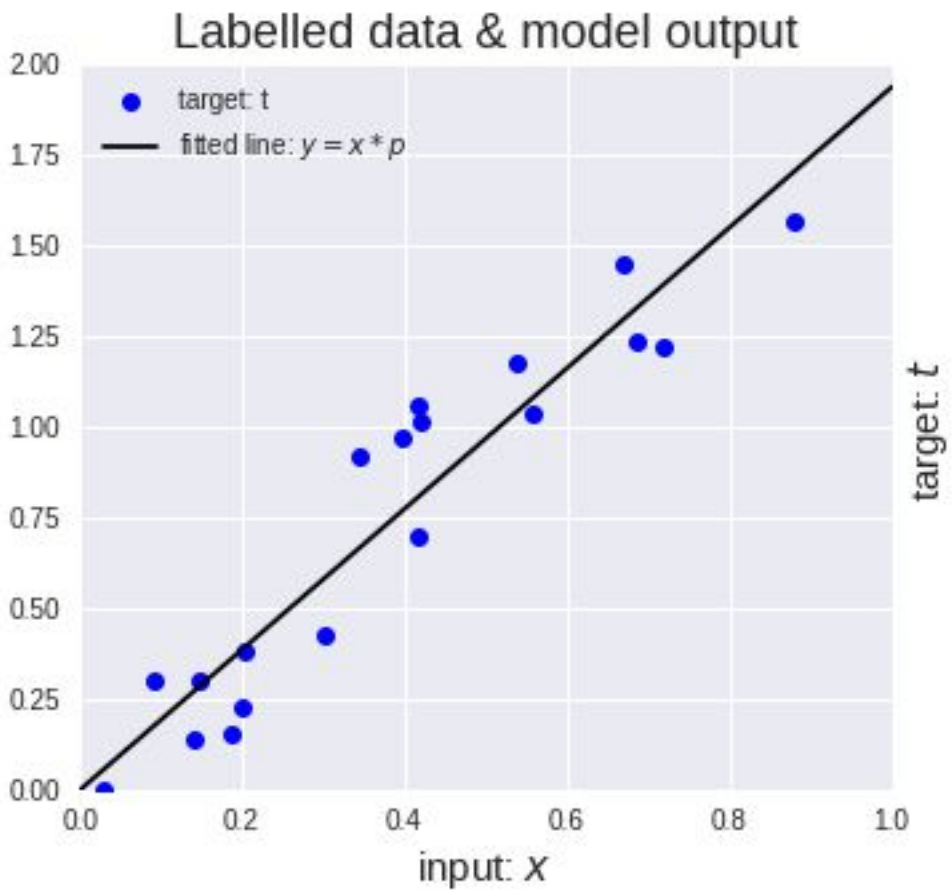


If the Learning Rate is too large we may never reach the minimum

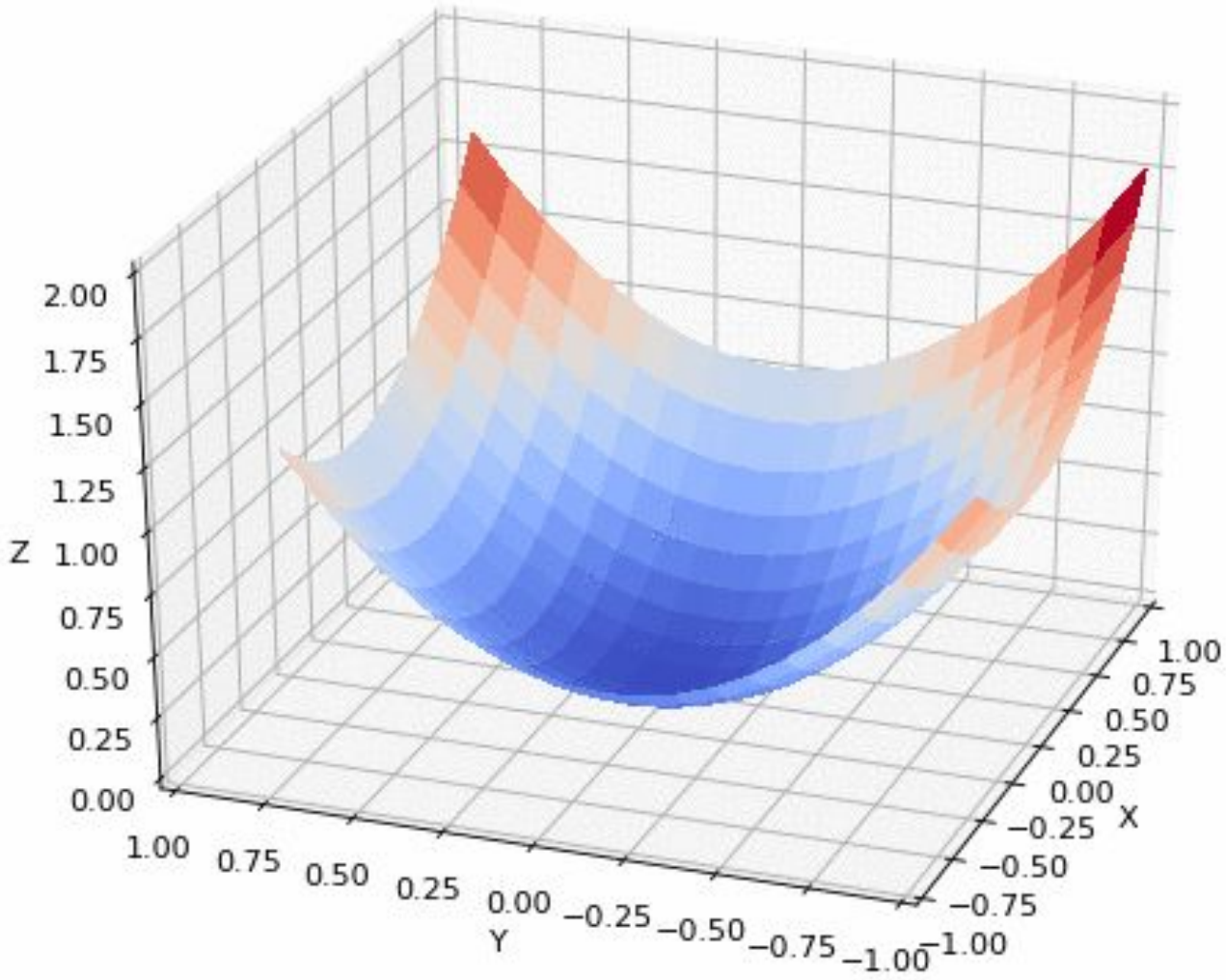


Gradient Descent algorithm





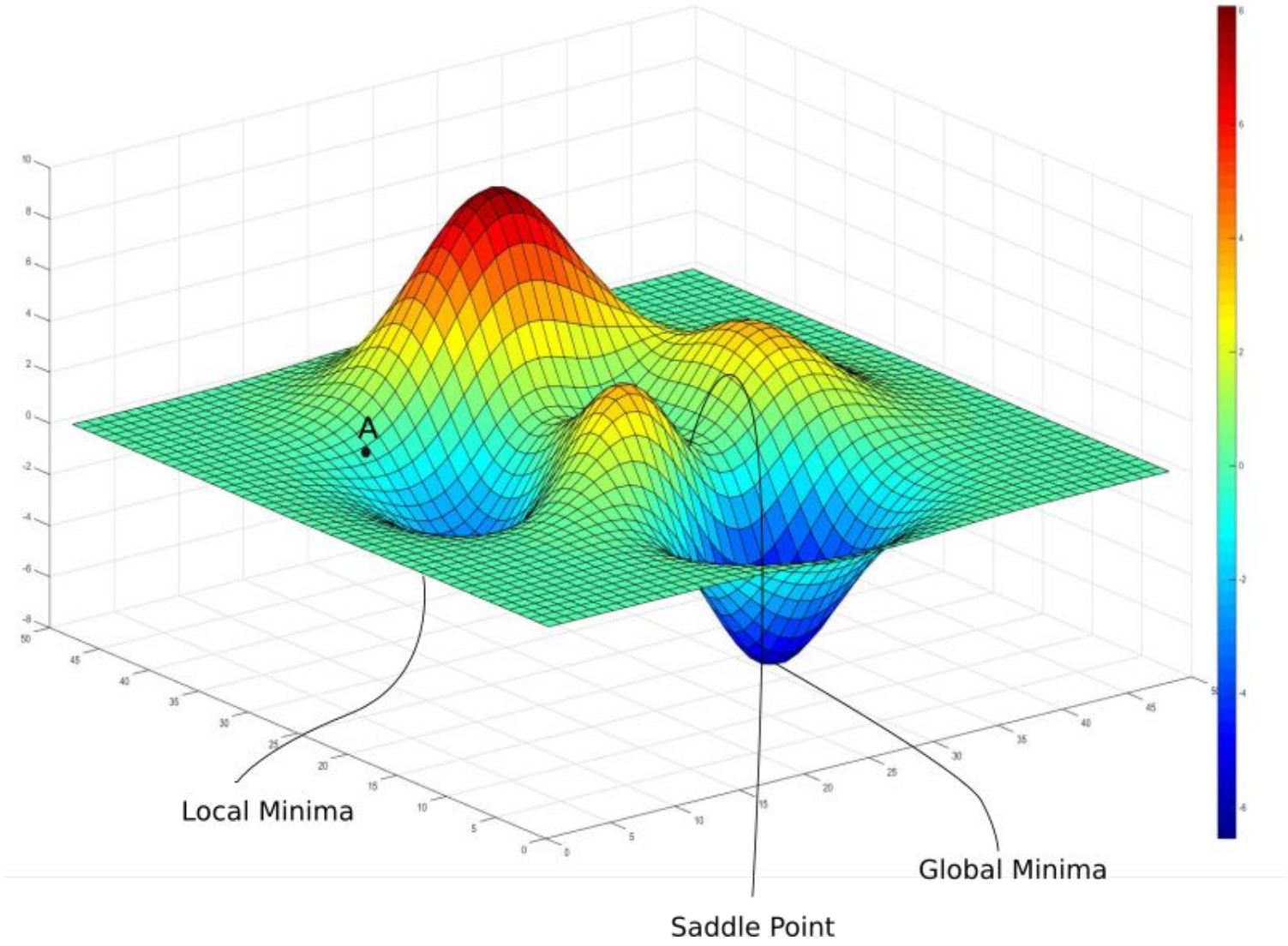
Gradient Descent for Two Parameters



A single minima
Global minima

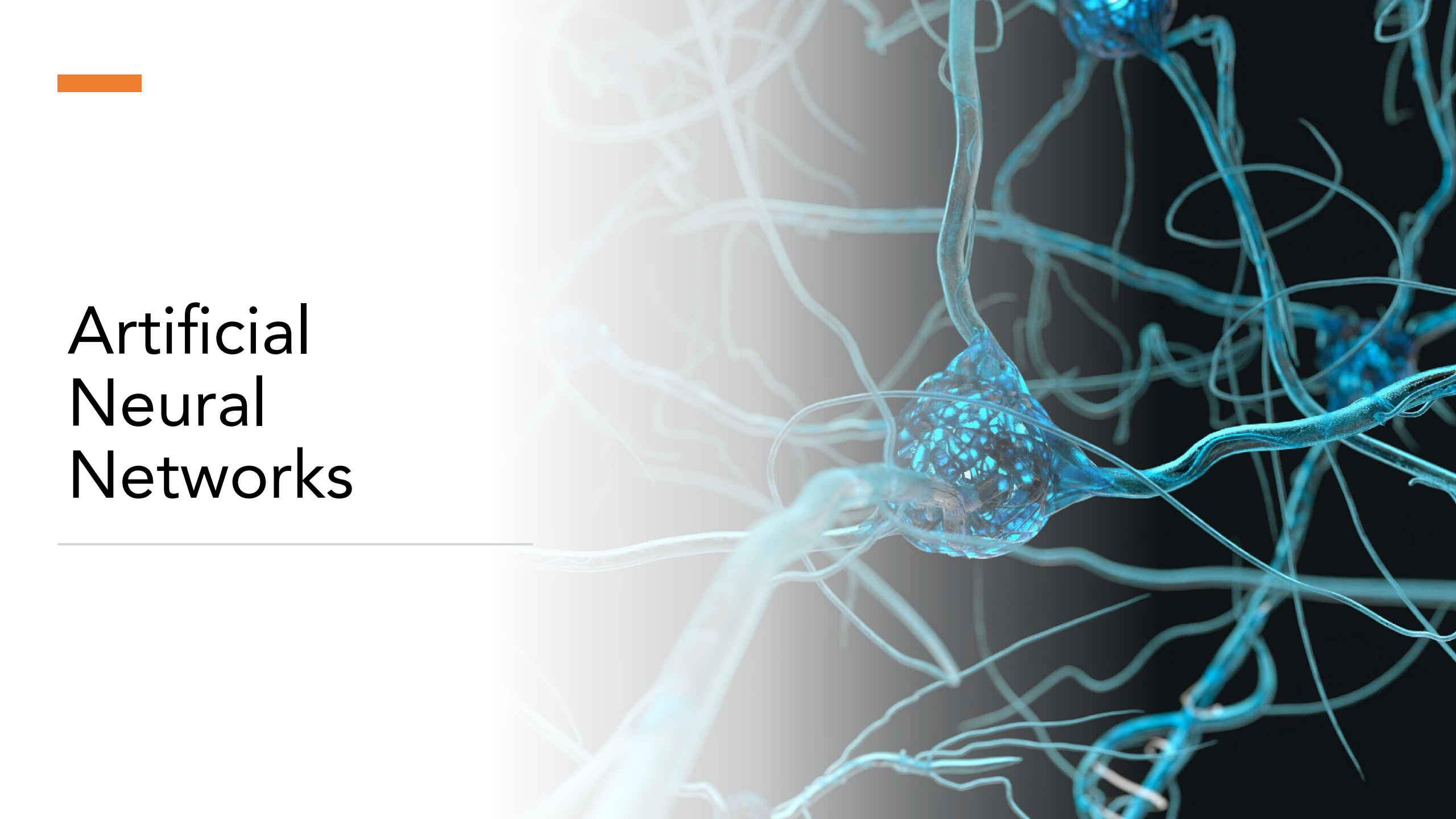
https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

Gradient Descent for Two Parameters

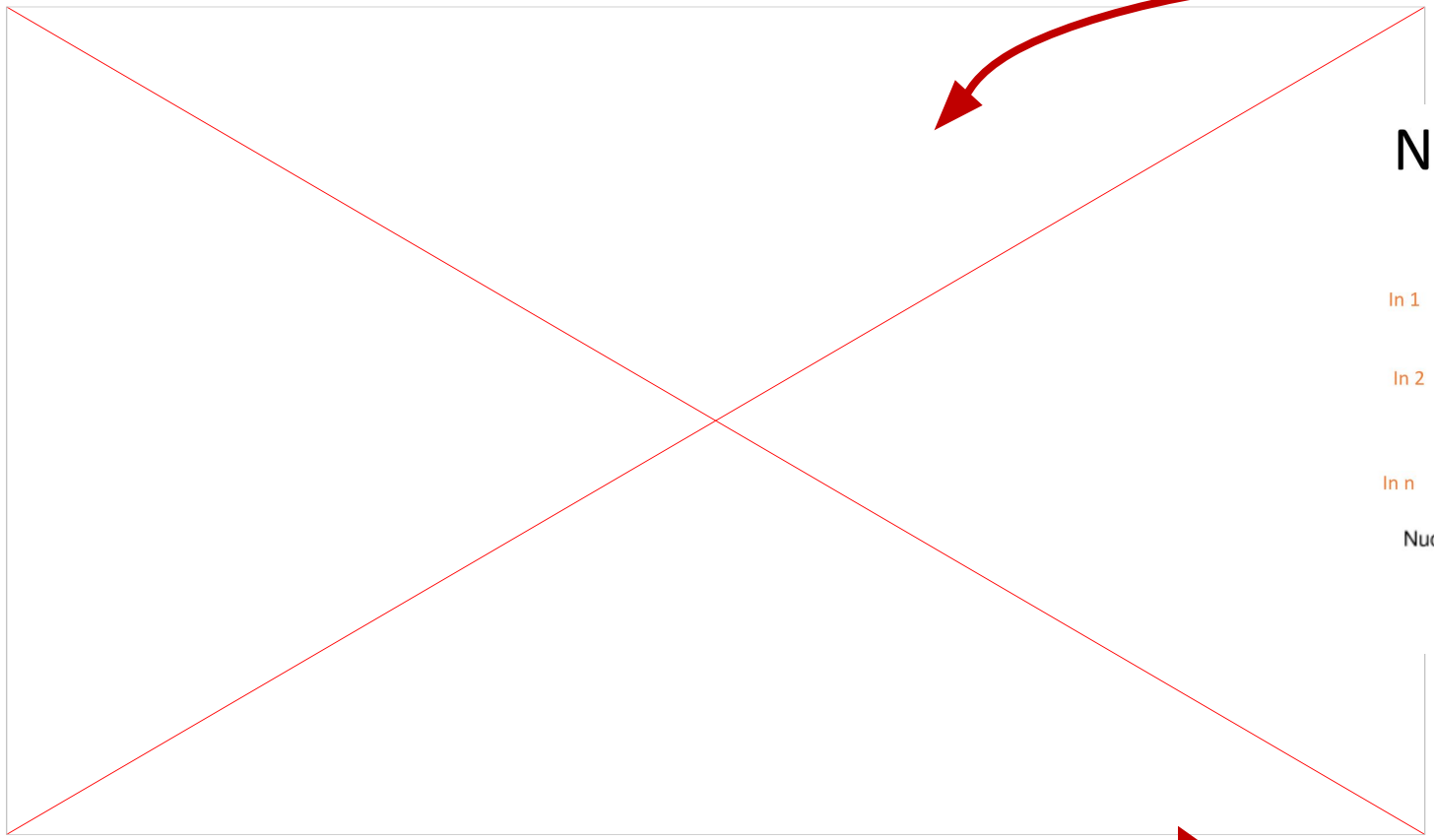


https://medium.com/analytics-vidhya/journey-of-gradient-descent-from-local-to-global-c851eba3d367

Artificial Neural Networks

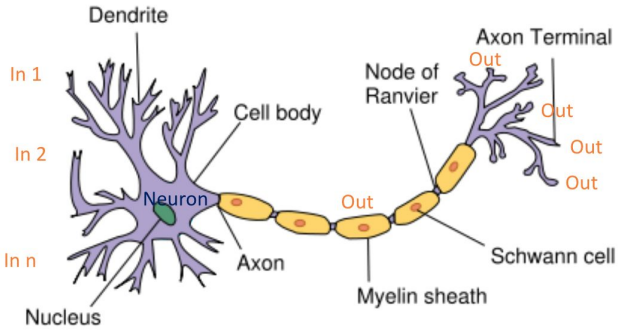


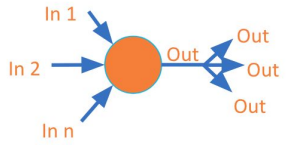
What is an Artificial Neural Network (ANN)?

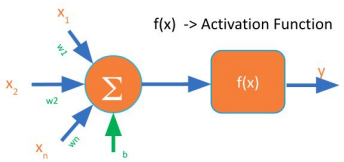


Sequential

Neuron (Perceptron)







Parameters

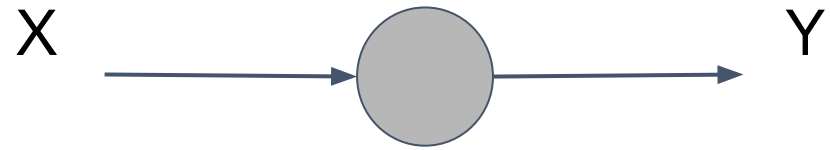
$$y = f(\sum_{i=1}^{n} x_i w_i + b)$$

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]))
```

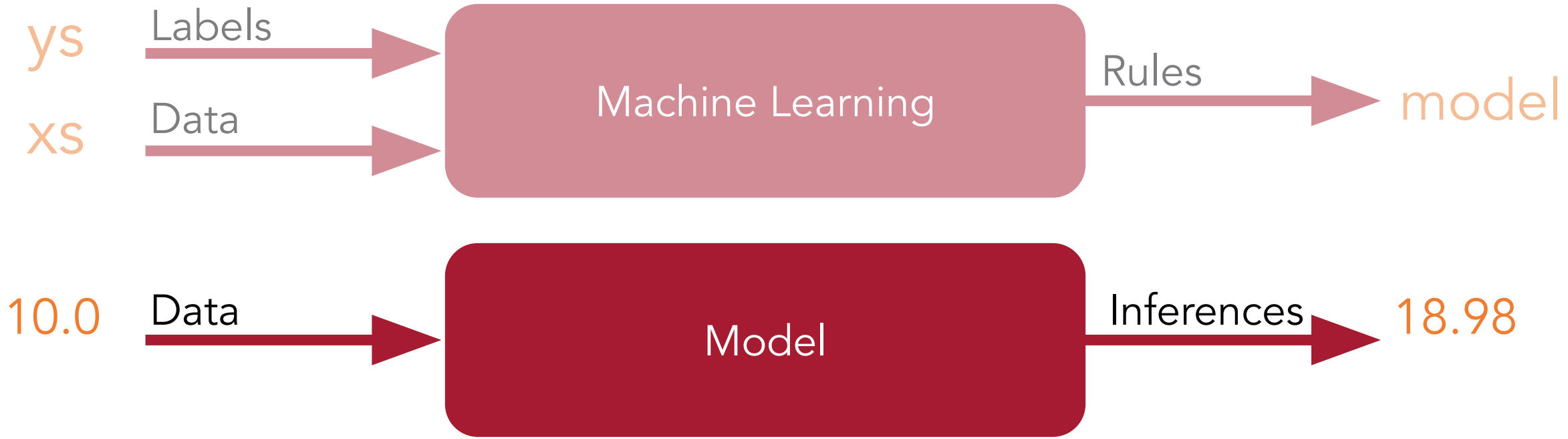
```
[[18.982813]]
```

```
[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)
```

$Y = 2X - 1$

```
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)]
```

$Y = 1.9975X - 0.9922$

Not perfect,
but good enough for most cases!

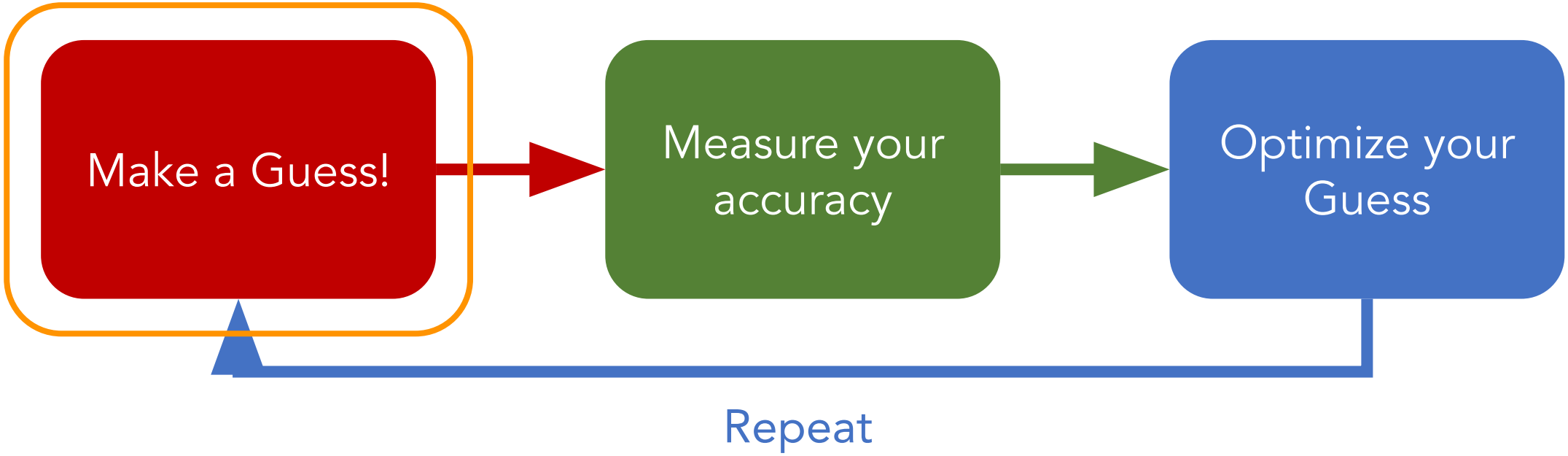
A Neuron in Tensor Flow

```
class Model(object):
  def __init__(self):
    self.w = tf.Variable(10.0)
    self.b = tf.Variable(10.0)

  def __call__(self, x):
    return self.w * x + self.b
```

A Neuron in Tensor Flow

```
class Model(object):
  def __init__(self):
    self.w = tf.Variable(10.0)
    self.b = tf.Variable(10.0)

  def __call__(self, x):
    return self.w * x + self.b
```

The Machine Learning Paradigm



A Neuron in Tensor Flow

```
model = Model()
xs = [-1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
ys = [-3.0, -1.0, 1.0, 3.0, 5.0, 7.0]
print(model(xs))
```

```
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]]
```

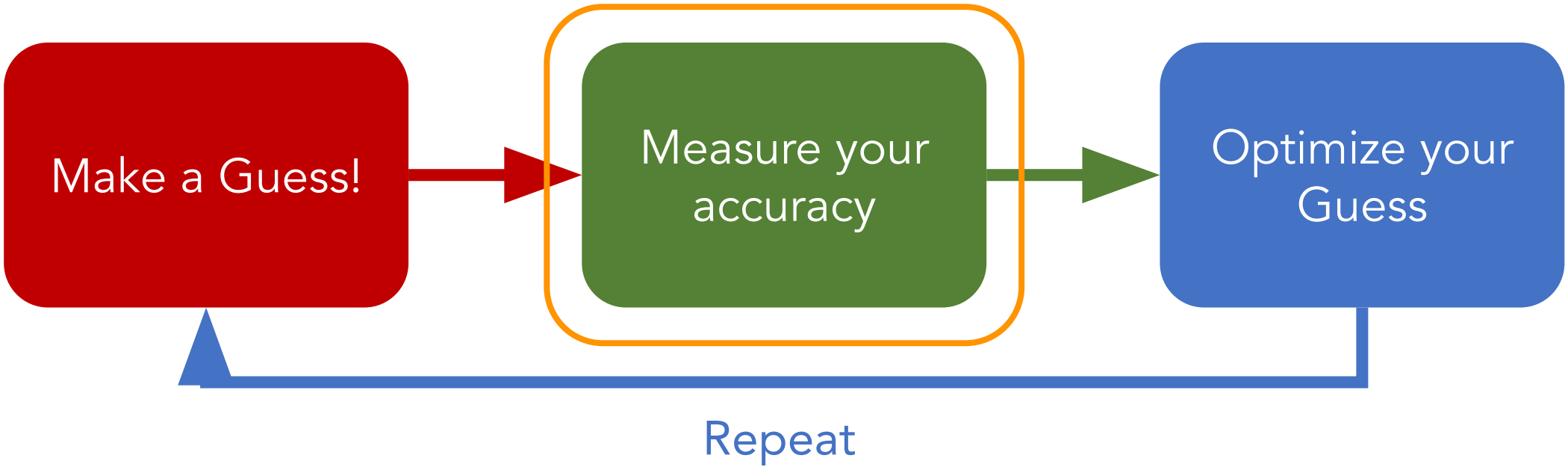
```
[5] print(model.predict(xs))
```

```
[[-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)]
```

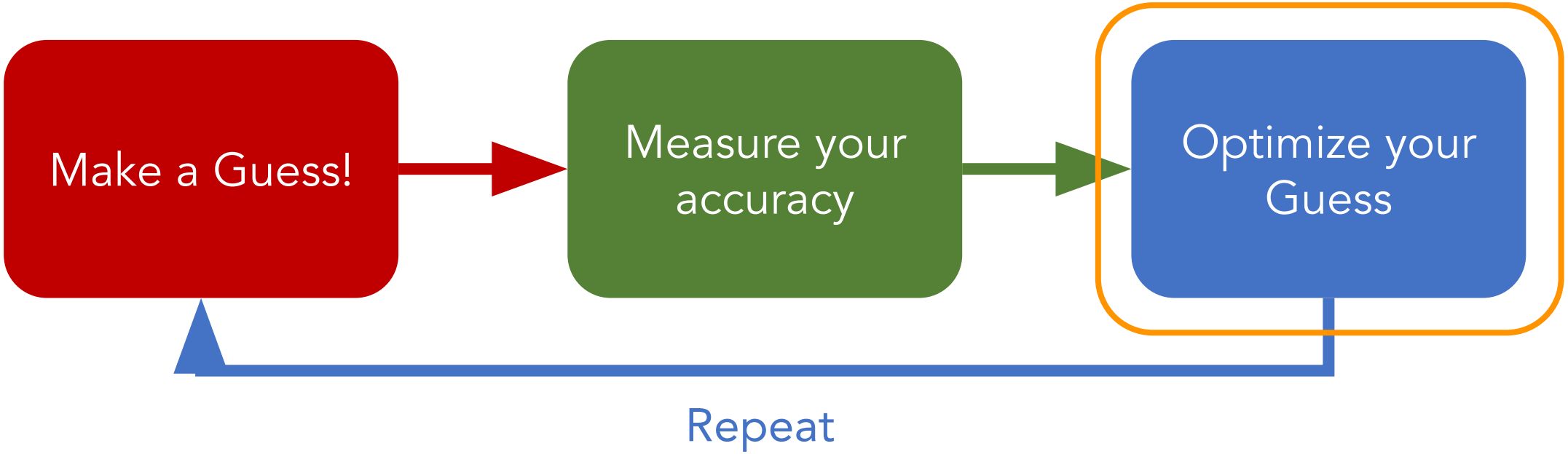
The Machine Learning Paradigm



```
def loss(predicted_y, target_y):
  return tf.reduce_mean(tf.square(predicted_y - target_y))
```

```
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

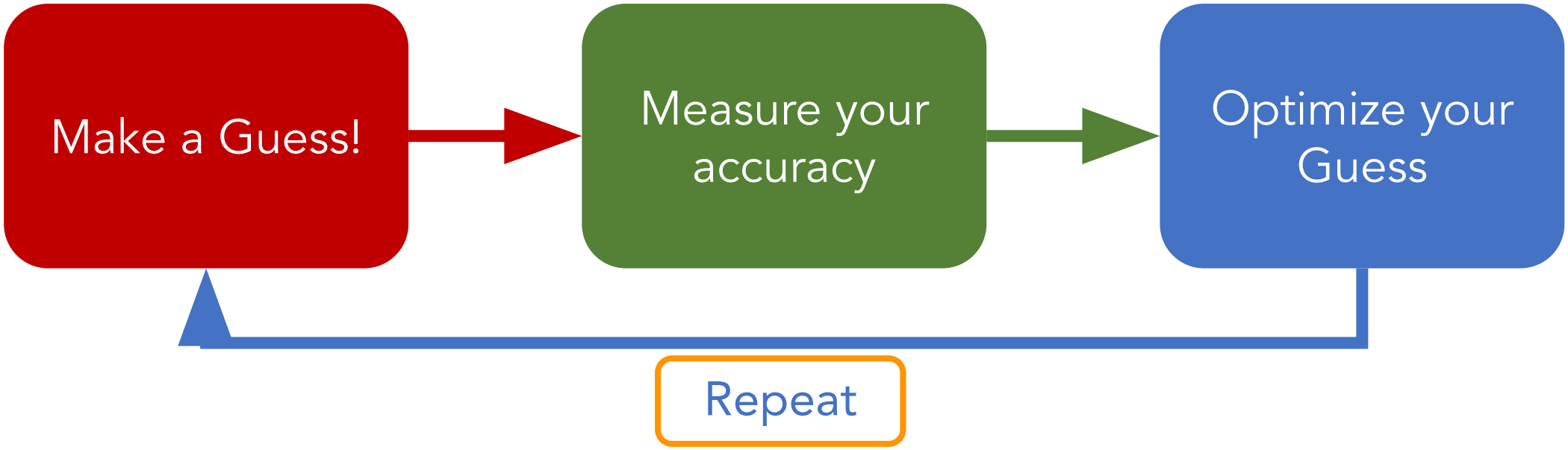
The Machine Learning Paradigm



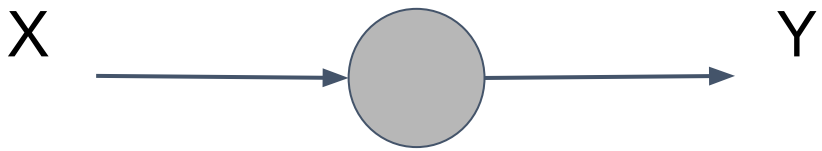
```
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

```
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

The Machine Learning Paradigm



```
for epoch in range(50):
  current_loss = train(model, xs, ys, learning_rate=0.1)
```



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

$$y = 1.9975x - 0.9922$$

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
email: diego-mendez@javeriana.edu.co