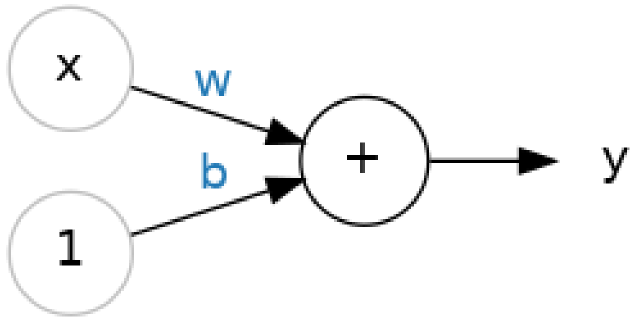
Machine Learning

2022. 8. 8

정 준 수 Ph.D

A Single Neuron

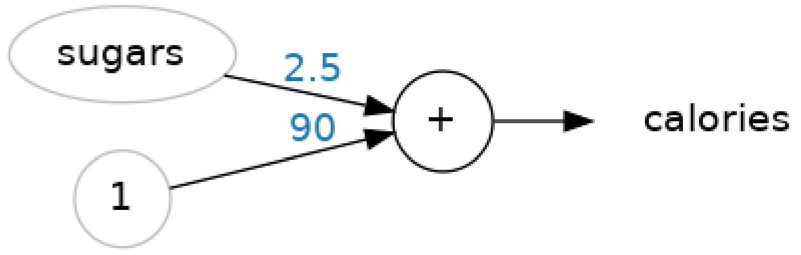
The individual neuron. As a diagram, a neuron (or unit) with one input looks like:



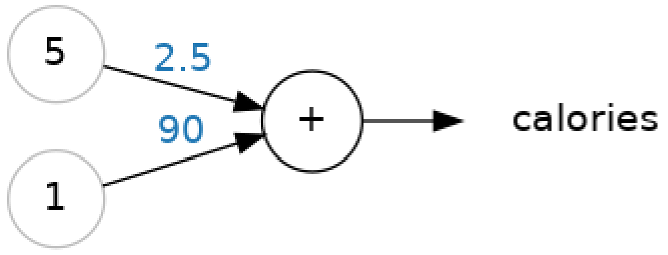
The Linear Unit: **y=wx+b**

선형 모델의 예제

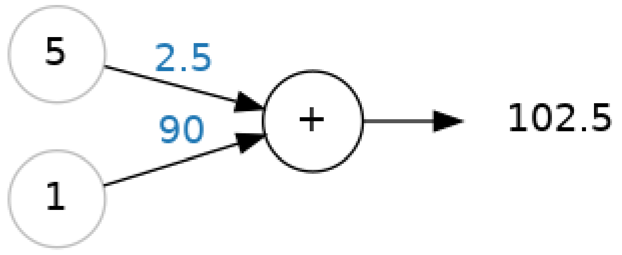
Training a model with 'sugars' (grams of sugars per serving) as input and 'calories' (calories per serving) as output, we might find the bias is b=90 and the weight is w=2.5. We could estimate the calorie content of a cereal with 5 grams of sugar per serving like this:



w=2.5, b=90



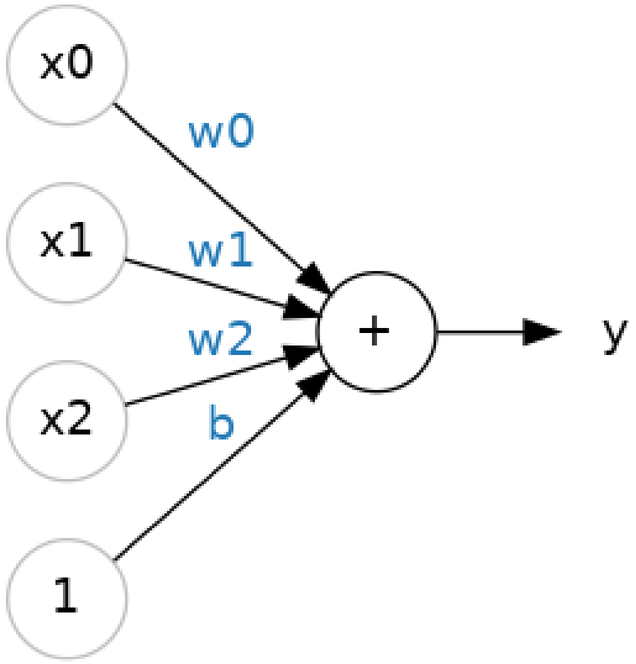
Input sugars=5



Multiply inputs by weights and sum.

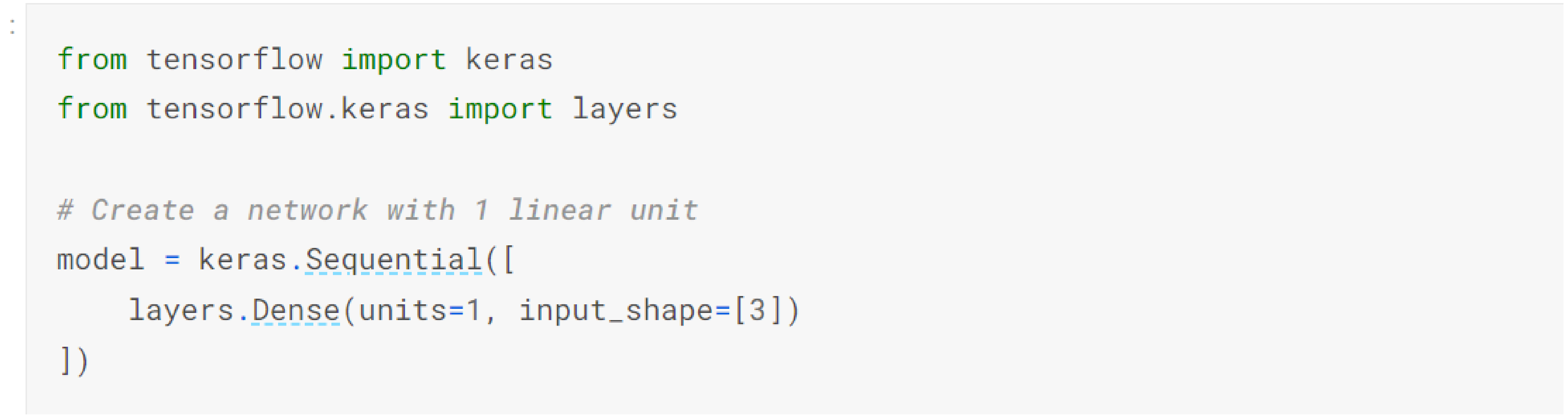
calories=2.5×5+90=102.5

Multiple Inputs



this neuron   y=w0x0+w1x1+w2x2+b 예를 들어
three features as input ('sugars', 'fiber', and 'protein')

Linear Units을 Keras의 Dense 표현하면

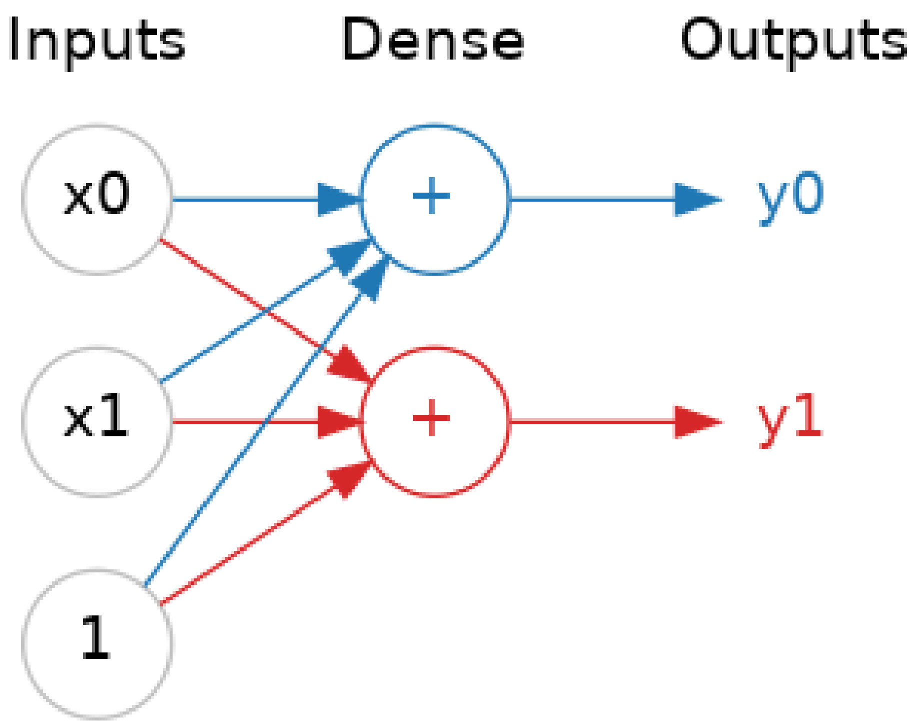

```
from tensorflow import keras
from tensorflow.keras import layers

# Create a network with 1 linear unit
model = keras.Sequential([
    layers.Dense(units=1, input_shape=[3])
])
```

Input [3] 개, Output [1] 개

Layers

Neural networks typically organize their neurons into layers. When we collect together linear units having a common set of inputs we get a dense layer.



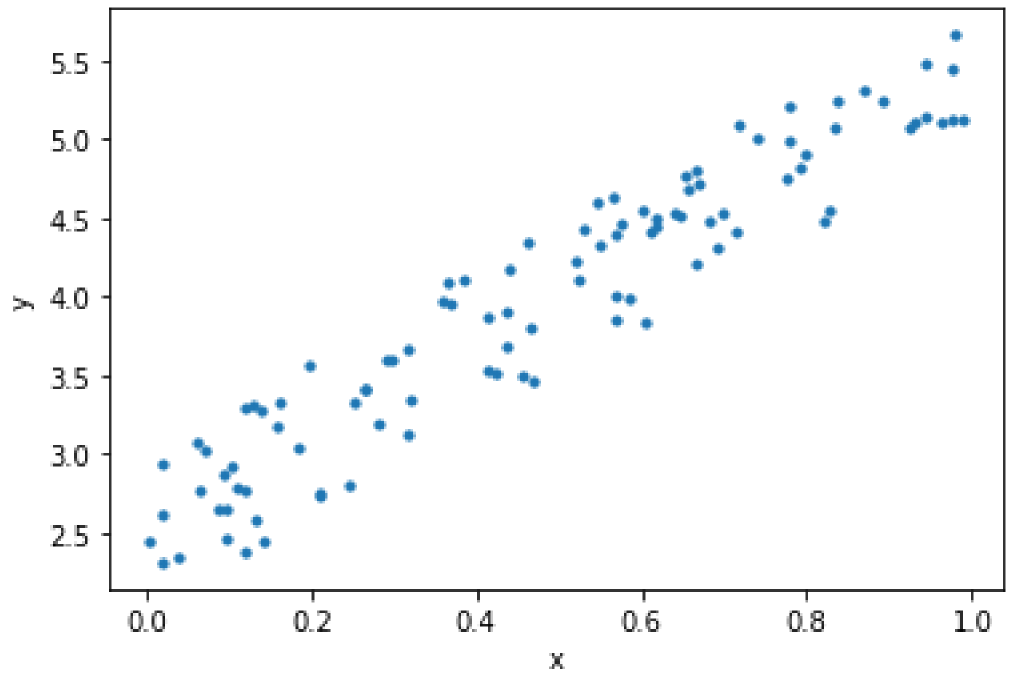
왼쪽 그림은 Layers 가 몇 개일까요?

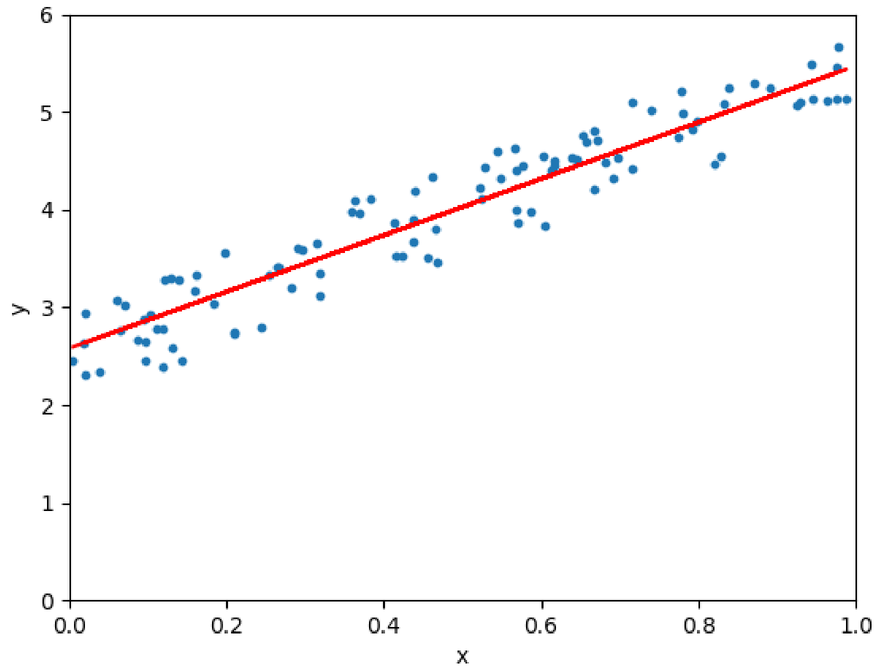
Keras – Part I

실습

https://github.com/JSJeong-me/Machine_Learning/blob/main/ML/Class-deeplearning.ipynb

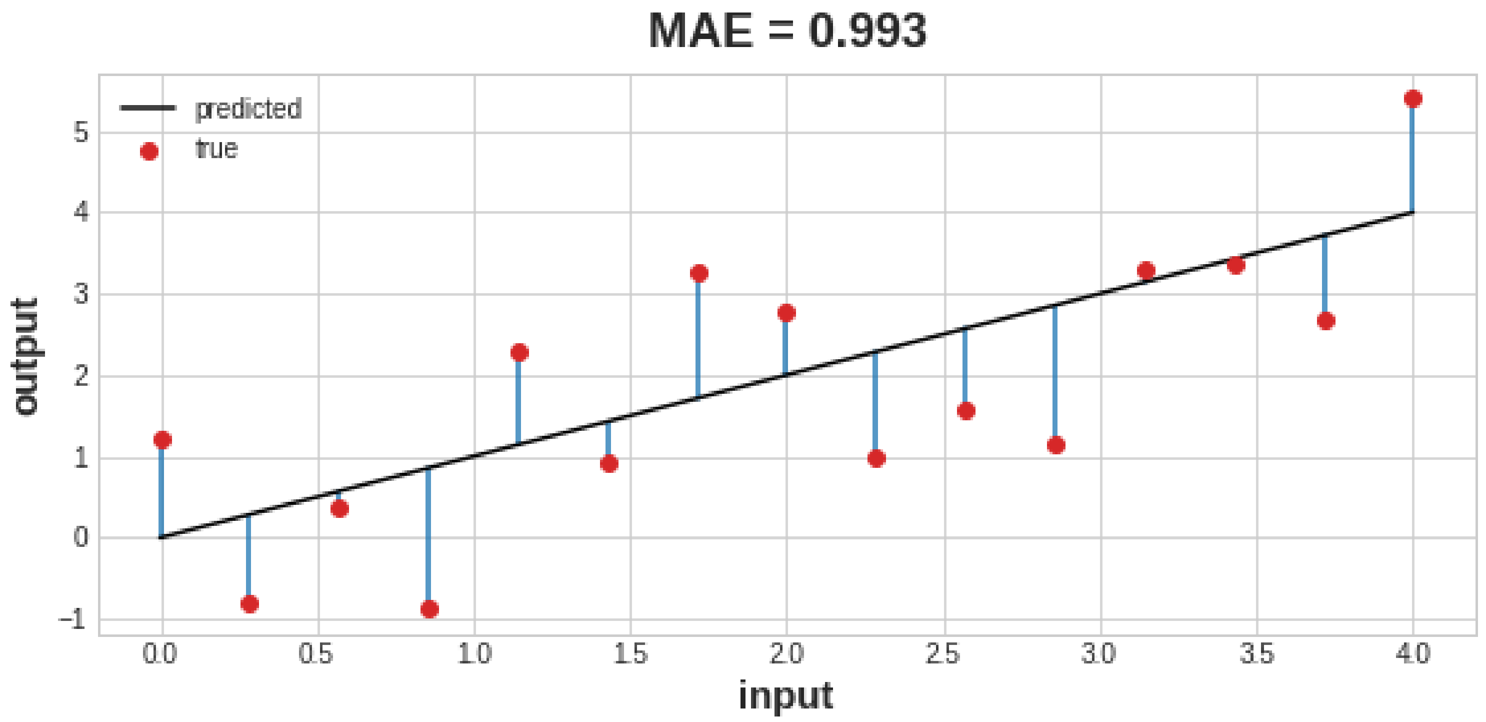
회기(Regression) 예측분석



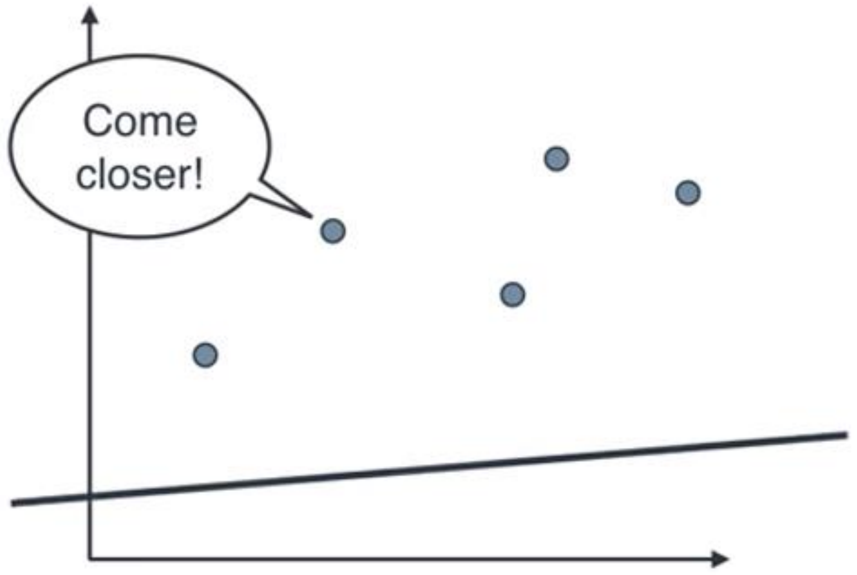


The Loss Function

mean absolute error or MAE



Linear regression algorithm



Step 1: Start with a random line

Step 2: Pick a large number. 1000 (number of repetitions, or epochs)

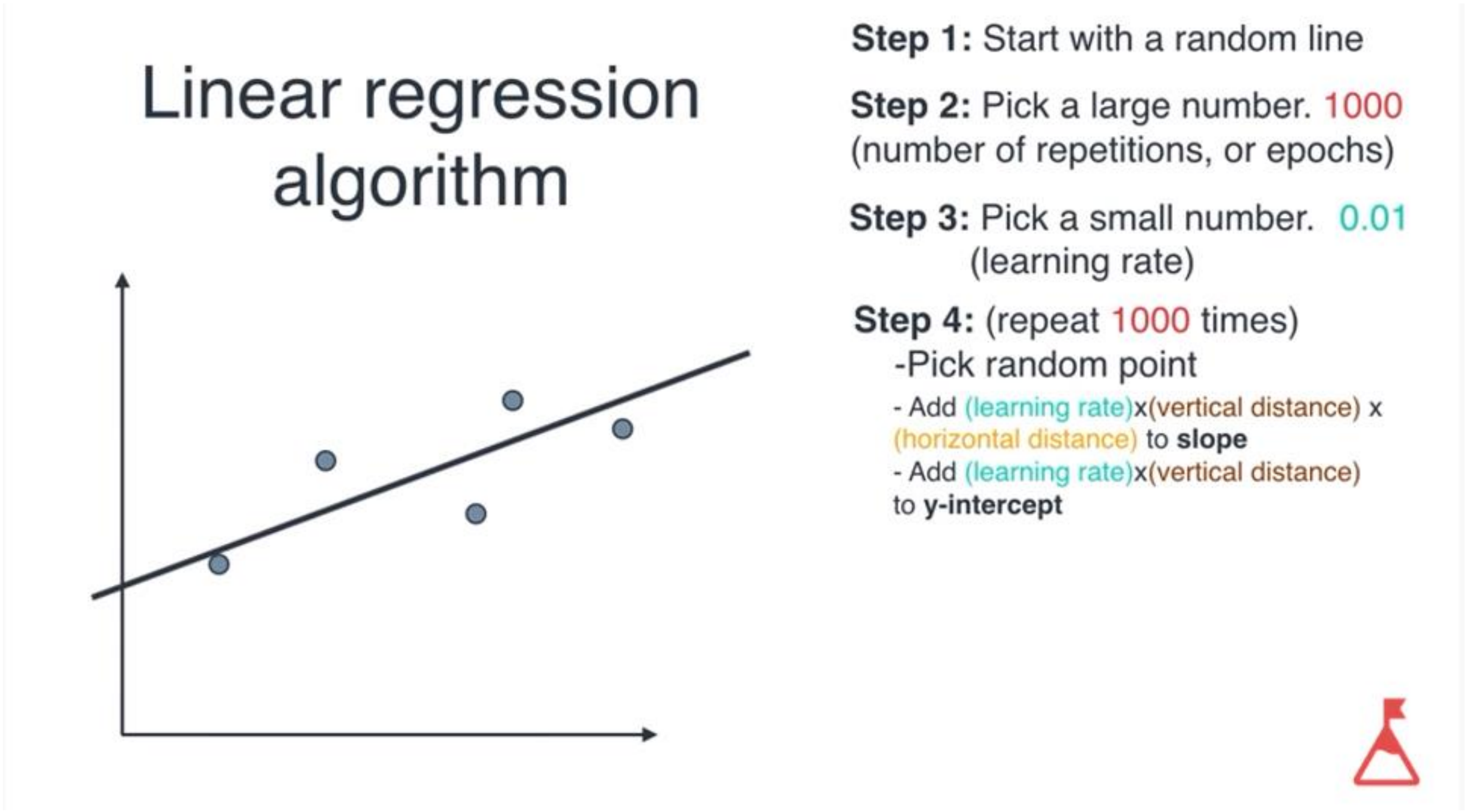
Step 3: Pick a small number. 0.01 (learning rate)

Step 4: (repeat 1000 times)

-Pick random point

- Add (learning rate)x(vertical distance) x (horizontal distance) to **slope**
- Add (learning rate)x(vertical distance) to **y-intercept**

https://www.youtube.com/watch?v=wYPUhge9w5c



Linear regression algorithm

Step 1: Start with a random line

Step 2: Pick a large number. 1000
(number of repetitions, or epochs)

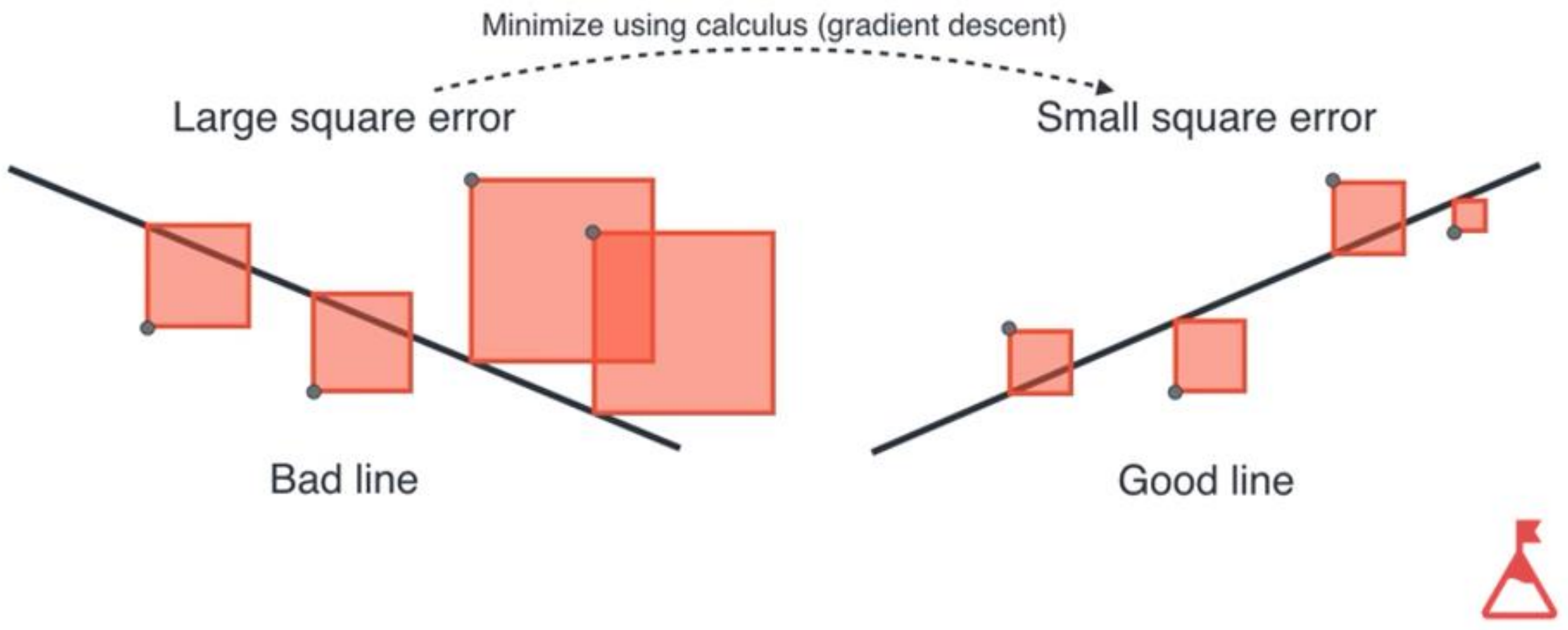
Step 3: Pick a small number. 0.01
(learning rate)

Step 4: (repeat 1000 times)
-Pick random point
- Add (learning rate)x(vertical distance) x (horizontal distance) to **slope**
- Add (learning rate)x(vertical distance) to **y-intercept**

https://www.youtube.com/watch?v=wYPUhge9w5c

Square error

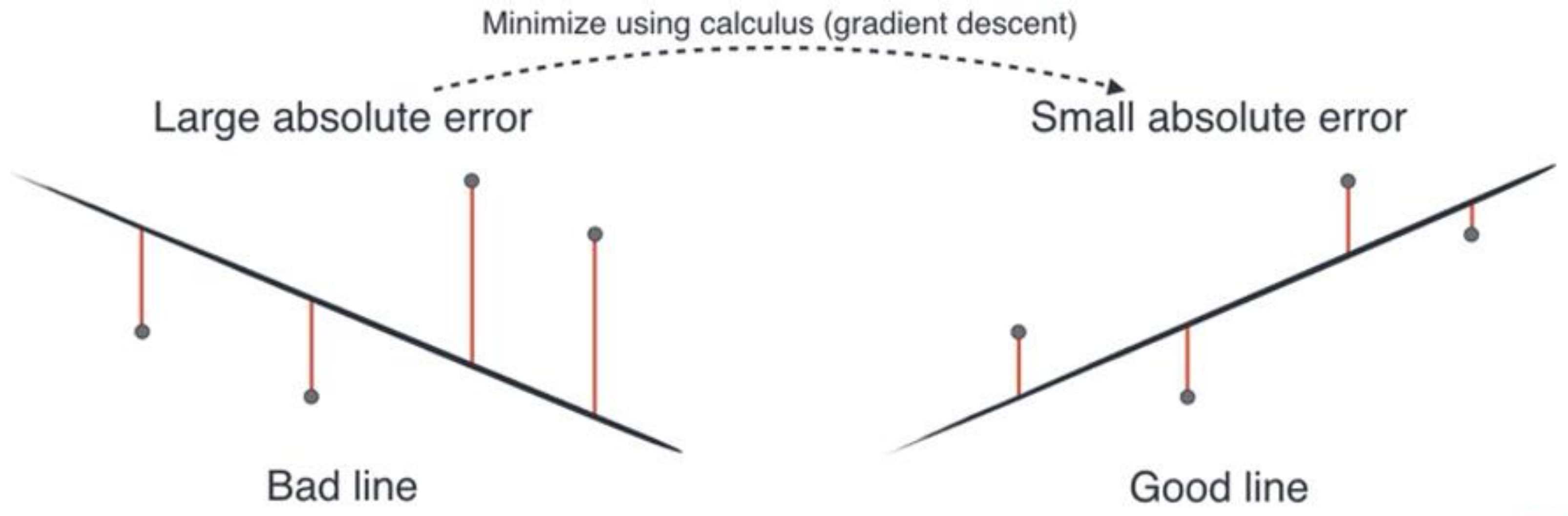
Same as the square trick!



https://www.youtube.com/watch?v=wYPUhge9w5c

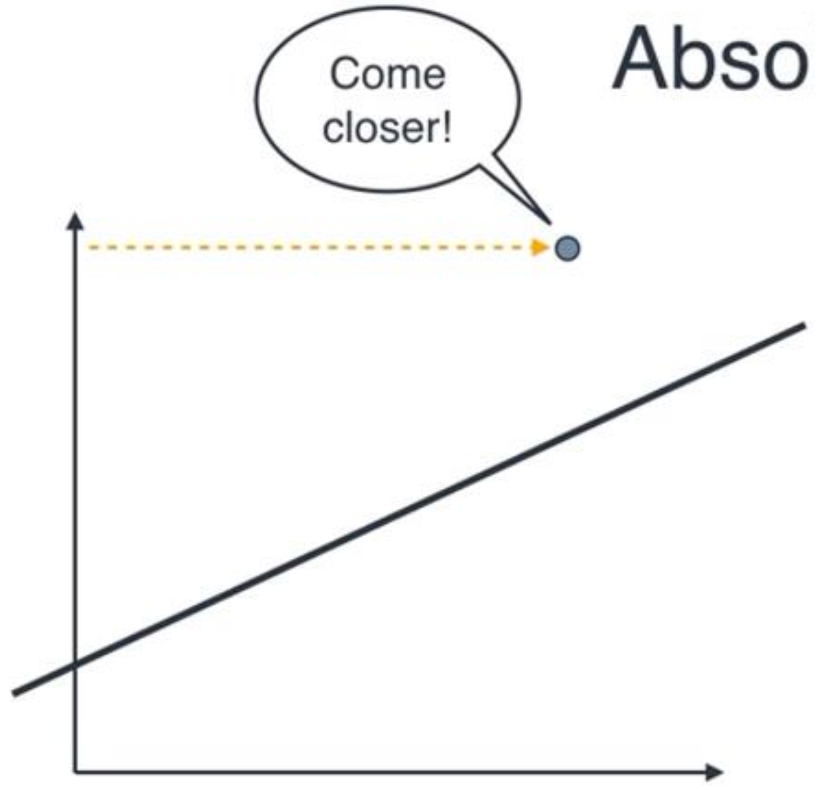
Absolute error

Develop an absolute trick!



https://www.youtube.com/watch?v=wYPUhge9w5c

Absolute trick



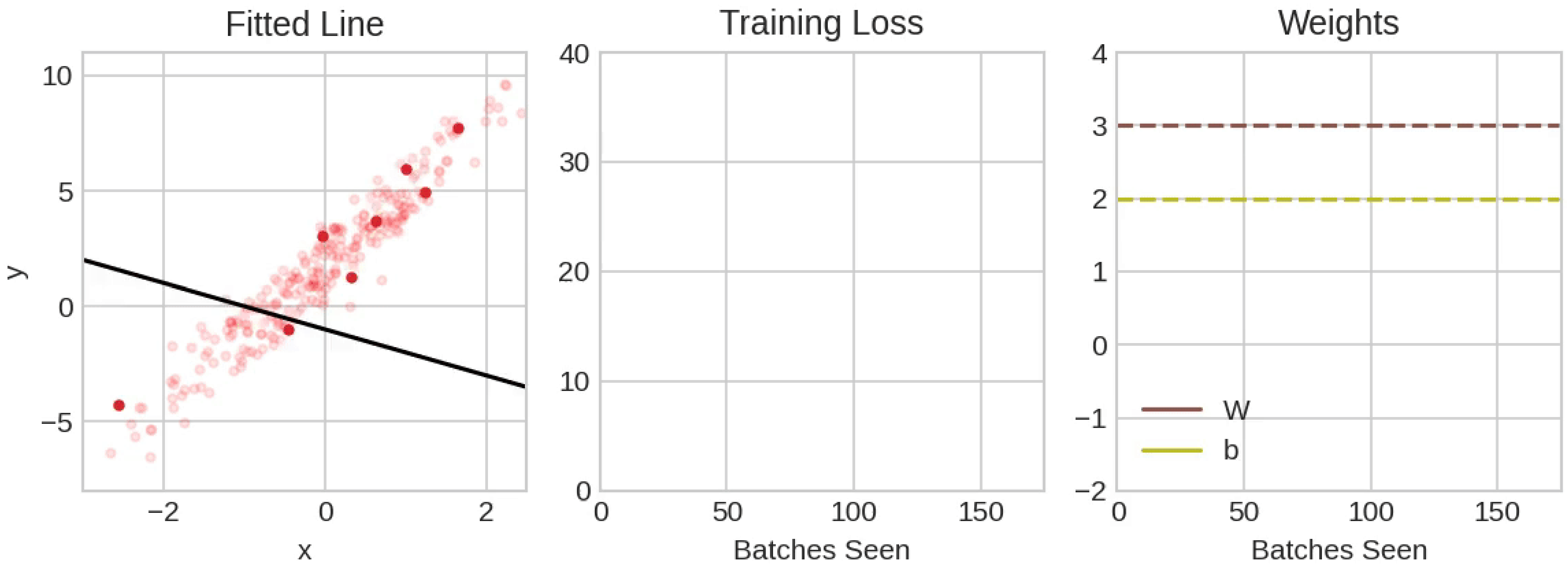
Step 1:
Pick a small number (learning rate)

Step 2:

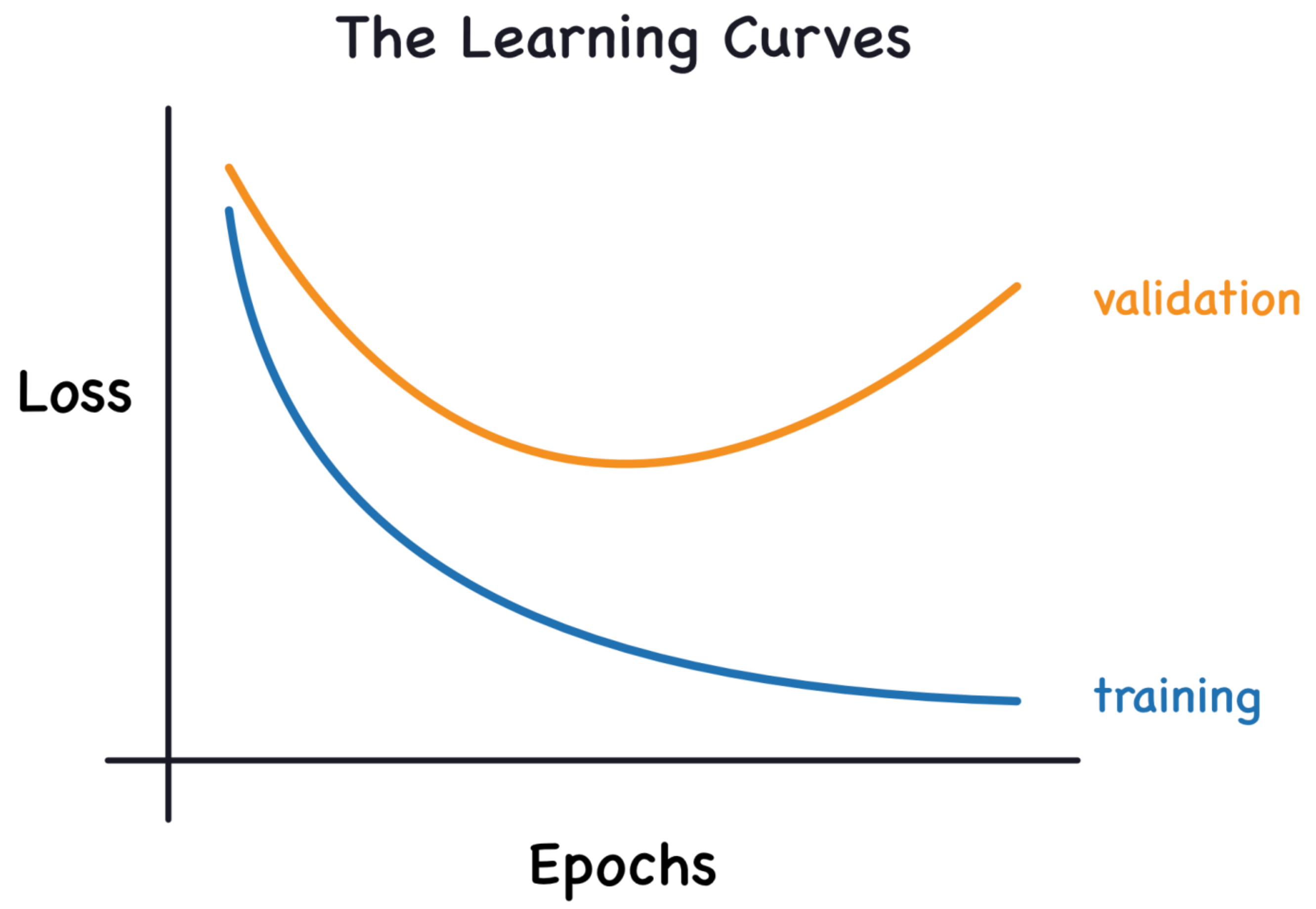
- If point is above the line:
  - Add (learning rate) x (horizontal distance) to **slope**
  - Add (learning rate) to **y-intercept**
- If point is below the line:
  - Subtract (learning rate) x (horizontal distance) to **slope**
  - Subtract (learning rate) to **y-intercept**

https://www.youtube.com/watch?v=wYPUhge9w5c

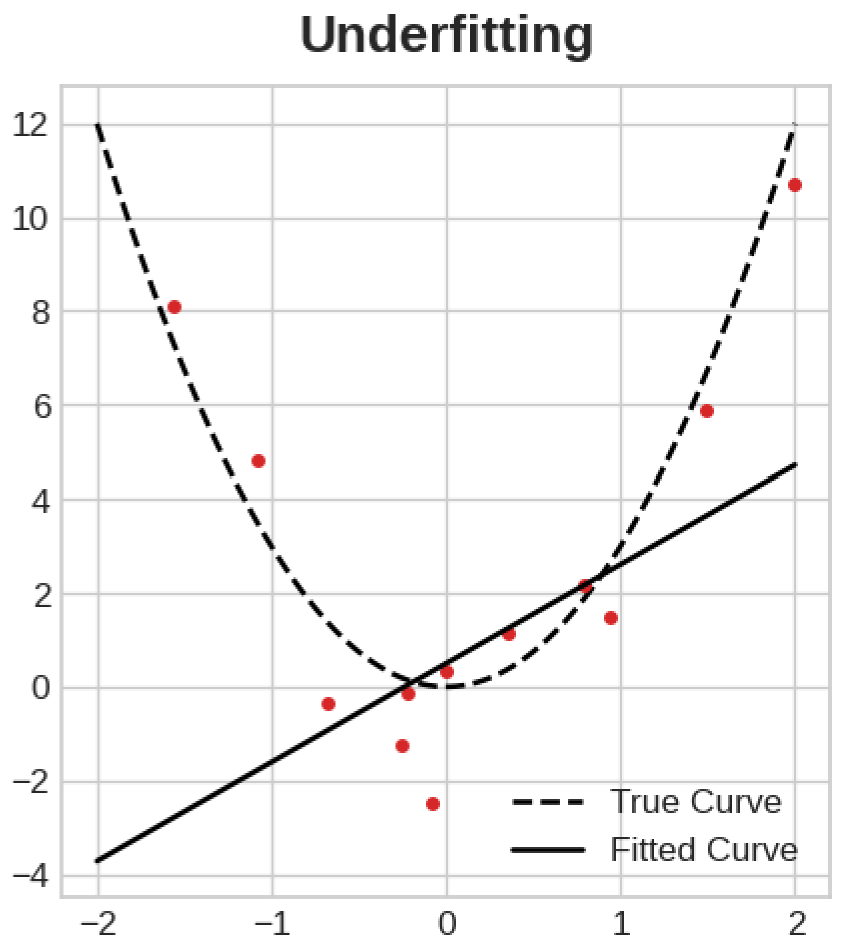
Stochastic Gradient Descent (The Optimizer)

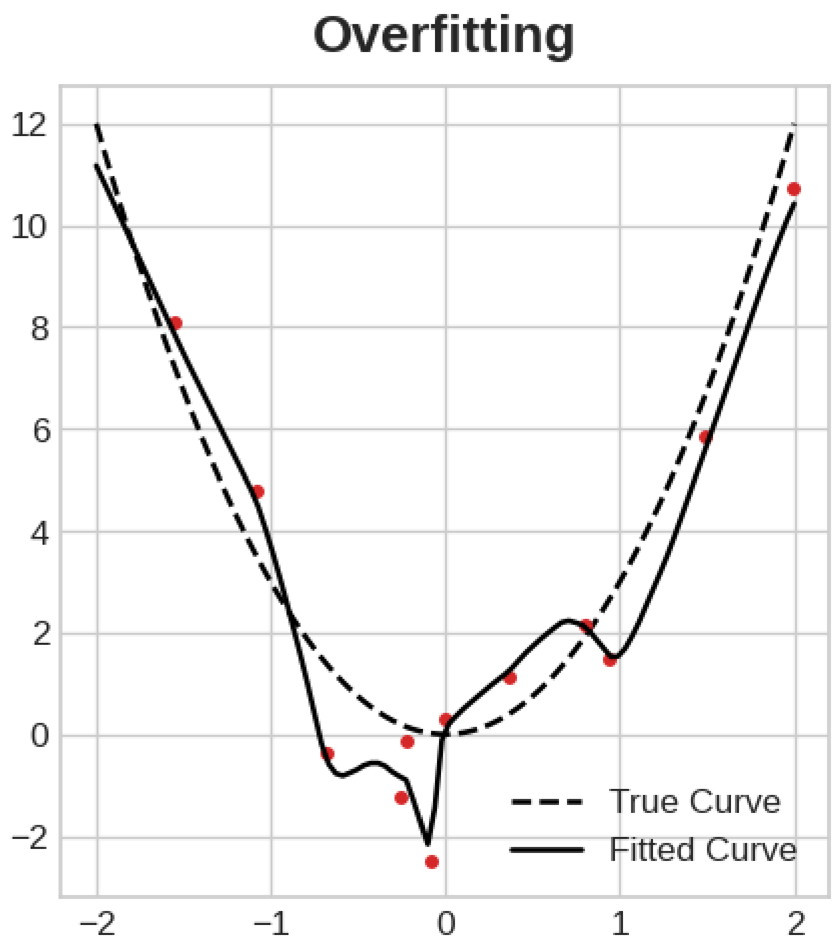


Learning Curves

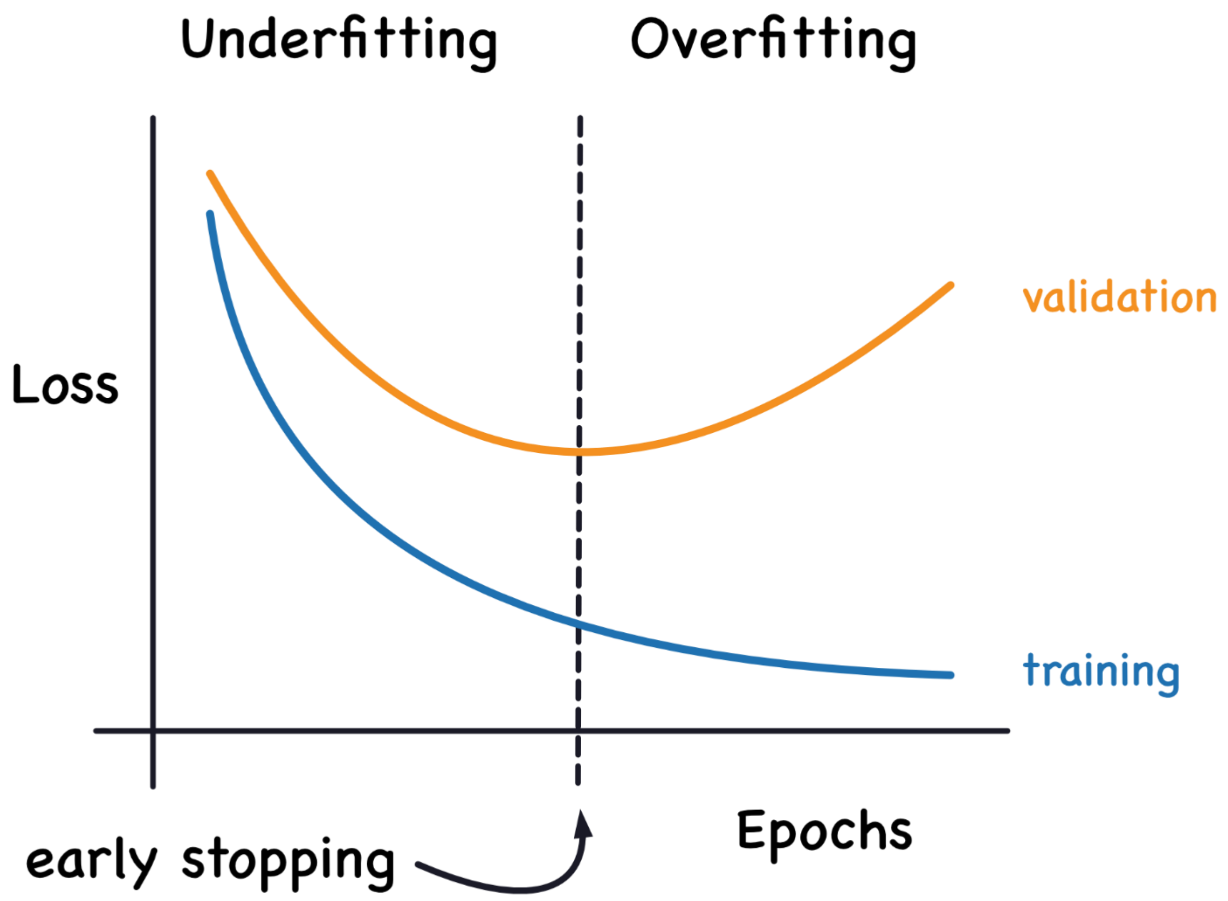


Overfitting and Underfitting

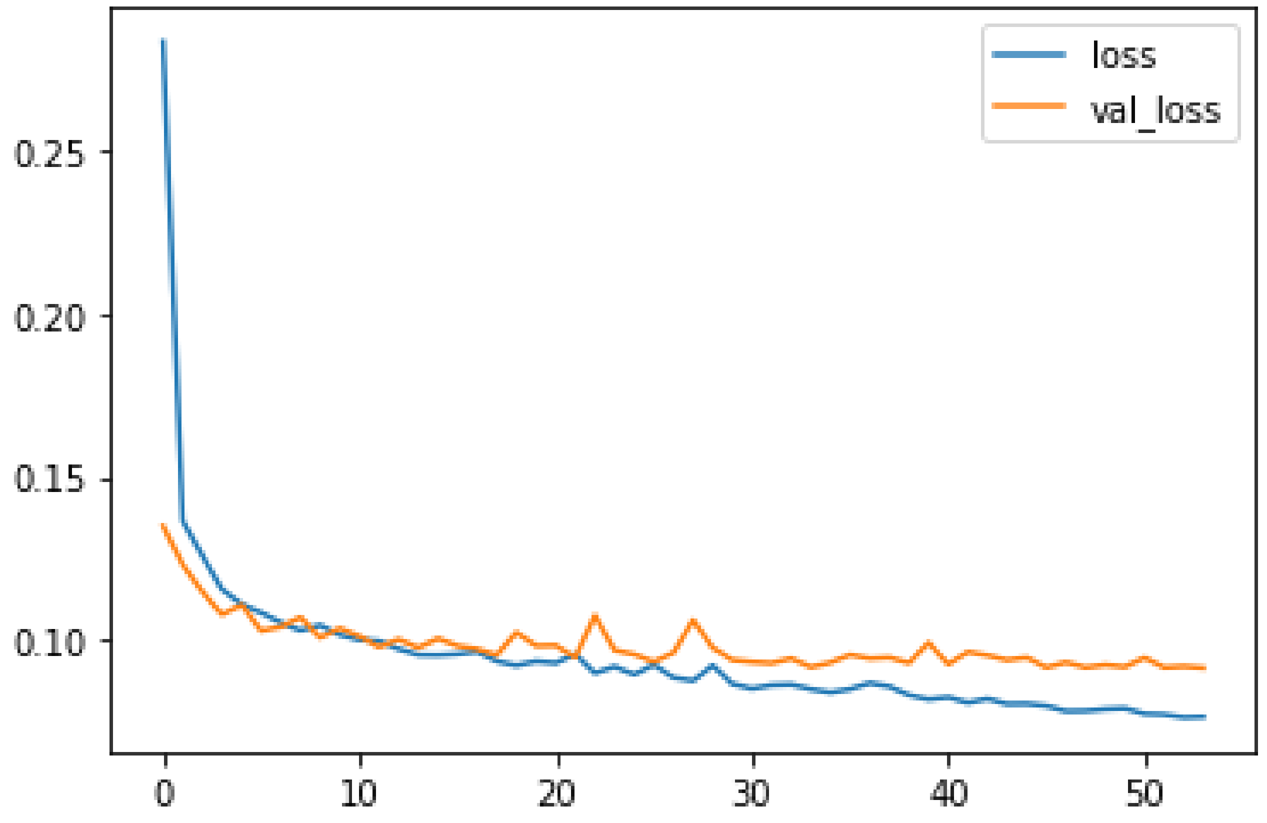




모델 학습은 언제 멈추어야 할까요? (Early Stopping)



학습 데이터와 검증 데이터 셋의 Loss 변환

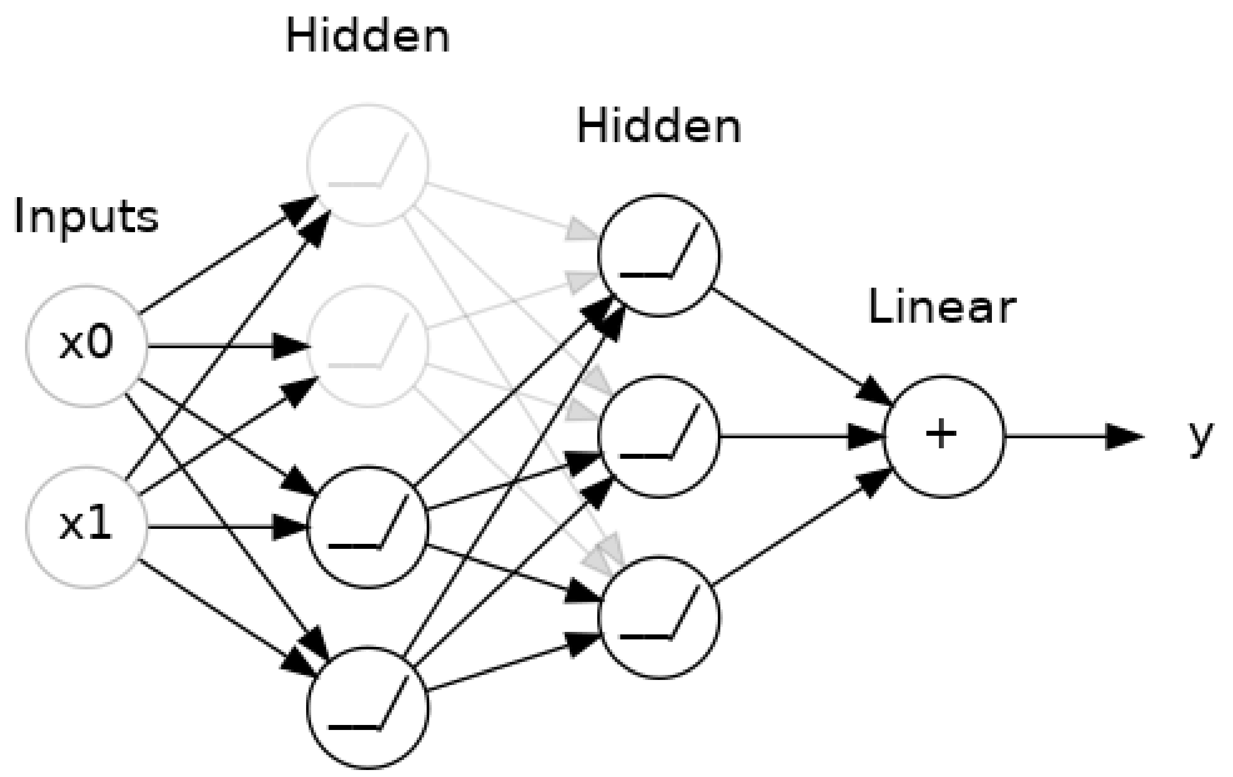


Adam : Minimum Validation Loss: **0.0155** epoch: 30

SGD : Minimum Validation Loss: **0.0166** epoch: 350

RMSProp : Minimum Validation Loss: **0.0159** **epoch: 38**

과적합을 방지하는 방법 : Dropout



Adding Dropout

In Keras, the dropout rate argument `rate` defines what percentage of the input units to shut off. Put the `Dropout` layer just before the layer you want the dropout applied to:

```
keras.Sequential([
    # ...
    layers.Dropout(rate=0.3), # apply 30% dropout to the next layer
    layers.Dense(16),
    # ...
])
```

과적합을 방지하는 방법 : Batch Normalization

A batch normalization layer looks at each batch as it comes in, first normalizing the batch with its own mean and standard deviation, and then also putting the data on a new scale with two trainable rescaling parameters. Batchnorm, in effect, performs a kind of coordinated rescaling of its inputs.

Adding Batch Normalization

It seems that batch normalization can be used at almost any point in a network. You can put it after a layer...

```
layers.Dense(16, activation='relu'),
layers.BatchNormalization(),
```

... or between a layer and its activation function:

```
layers.Dense(16),
layers.BatchNormalization(),
layers.Activation('relu'),
```

And if you add it as the first layer of your network it can act as a kind of adaptive preprocessor, standing in for something like Sci-Kit Learn's `StandardScaler`.

Activation Function

A neural network is comprised of layers of nodes and learns to map examples of inputs to outputs.

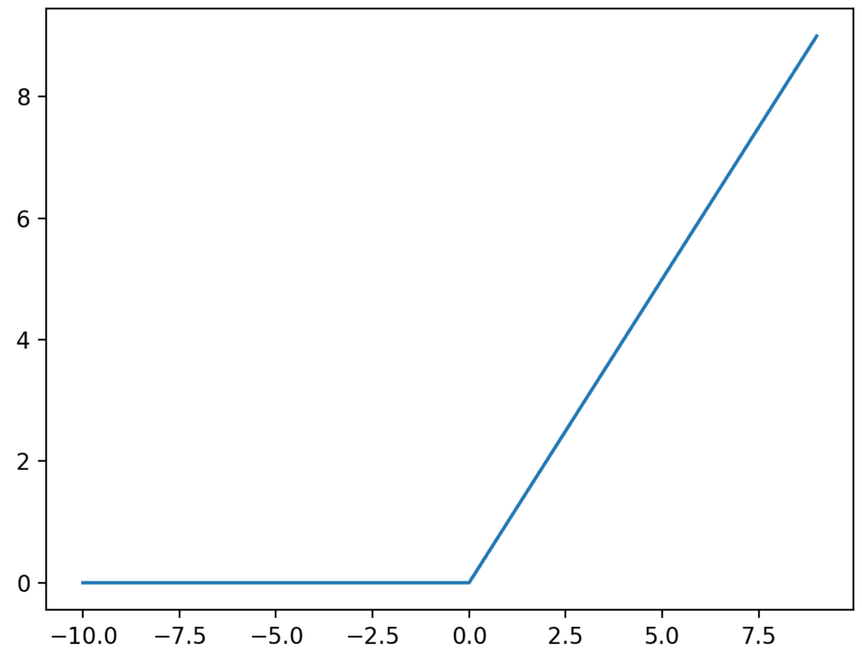
For a given node, the inputs are multiplied by the weights in a node and summed together. This value is referred to as the summed activation of the node. The summed activation is then transformed via an activation function and defines the specific output or “activation” of the node.

The simplest activation function is referred to as the linear activation, where no transform is applied at all. A network comprised of only linear activation functions is very easy to train, but cannot learn complex mapping functions. Linear activation functions are still used in the output layer for networks that predict a quantity (e.g. regression problems).

Nonlinear activation functions are preferred as they allow the nodes to learn more complex structures in the data. Traditionally, two widely used nonlinear activation functions are the **sigmoid** and **hyperbolic tangent** activation functions.

[출처] https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

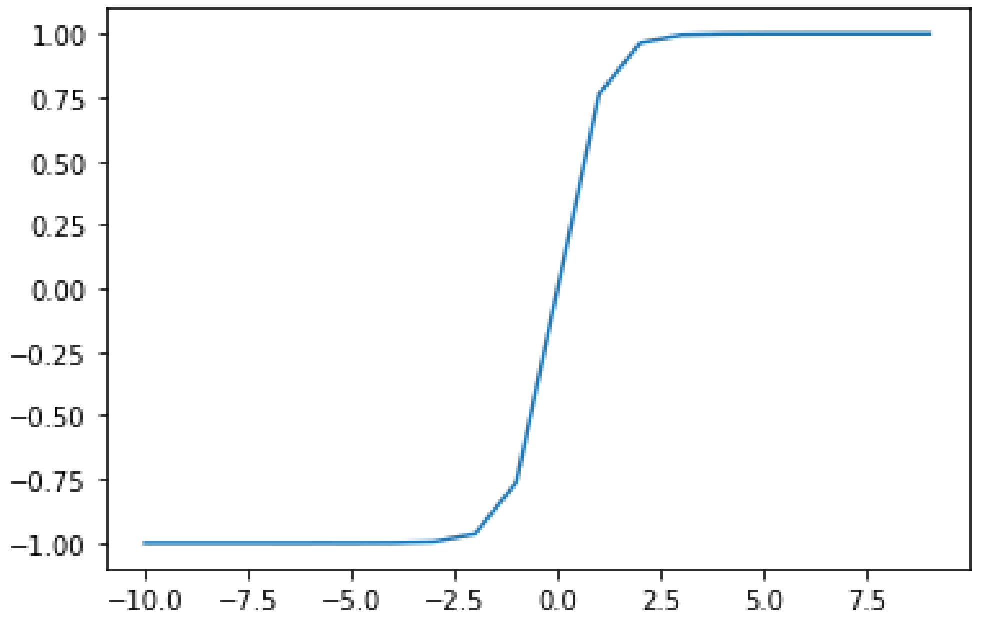
ReLU Hidden Layer Activation Function



코드 -> https://github.com/JSJeong-me/Machine_Learning/blob/main/ML/6-ReLU.ipynb

[출처]]https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Tanh Hidden Layer Activation Function



코드 -> https://github.com/JSJeong-me/Machine_Learning/blob/main/ML/6-tanh.ipynb

[출처]]https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

- The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem.

- The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

- The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.

Stochastic Optimization Algorithms

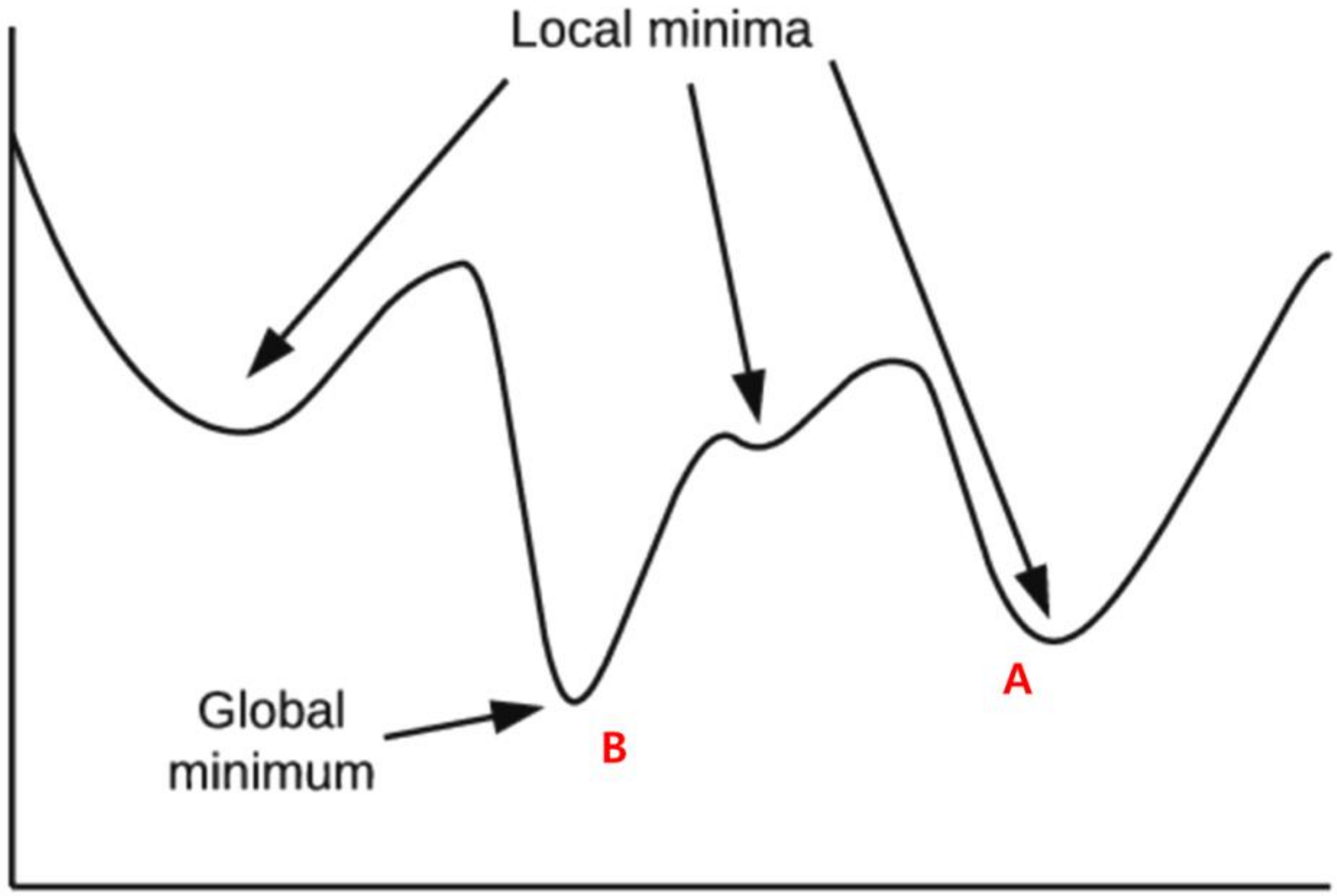
•Stochastic optimization algorithms make use of randomness as part of the search procedure.

•Examples of stochastic optimization algorithms like simulated annealing and genetic algorithms.

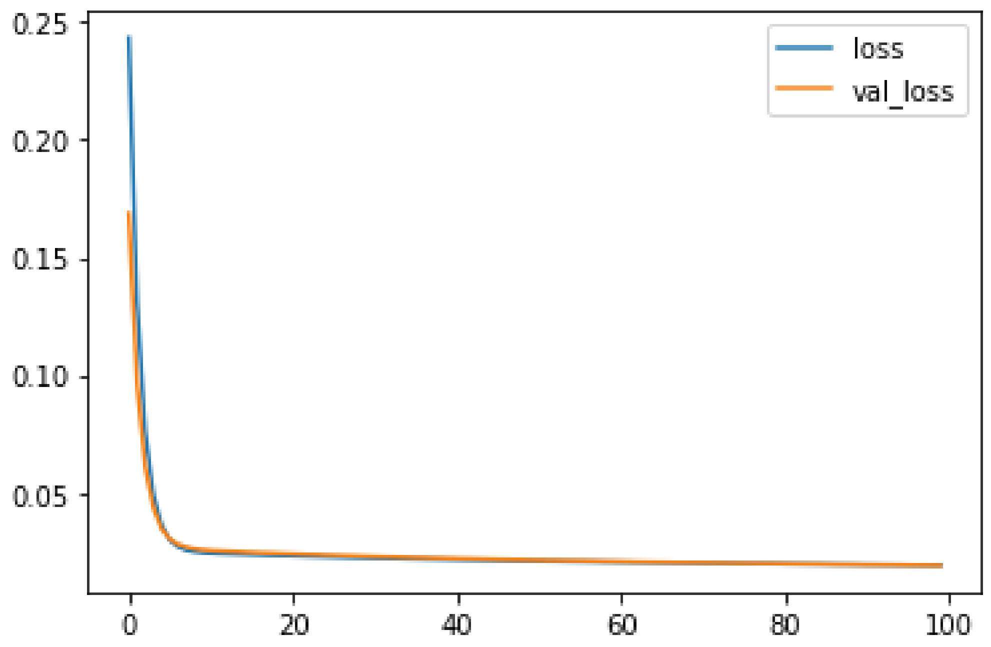
•Practical considerations when using stochastic optimization algorithms such as repeated evaluations.

Stochastic Optimization Algorithms - 참고자료

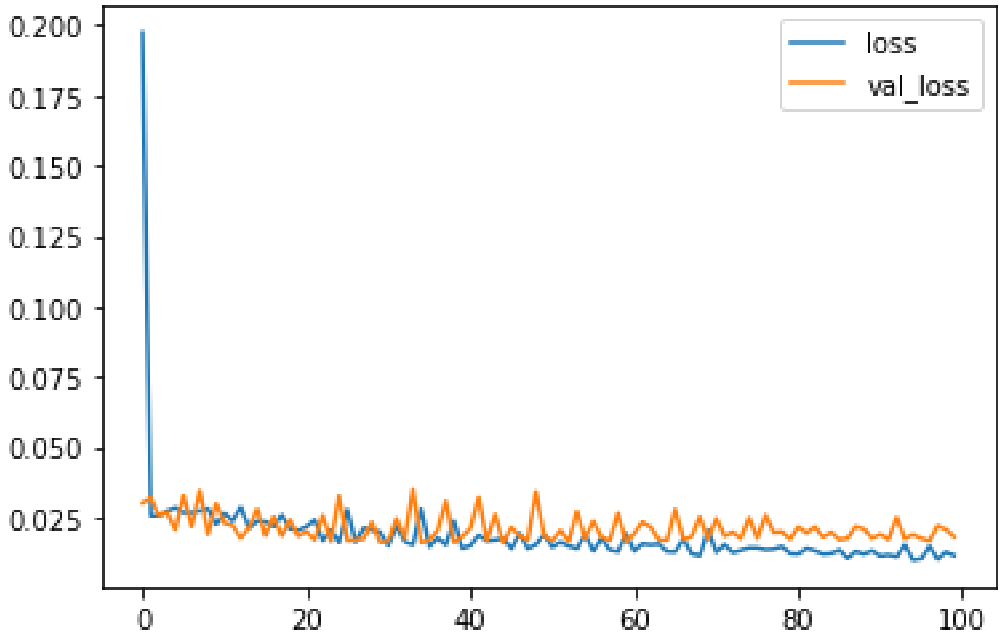
https://machinelearningmastery.com/adam-optimization-from-scratch/


Local minima
Global minimum
A
B

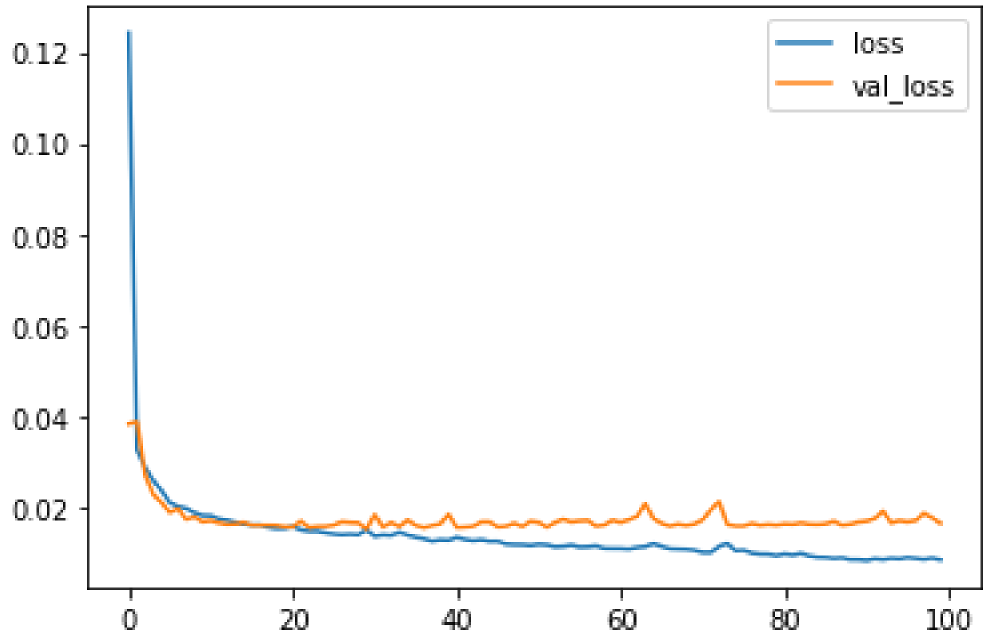
Optimizer: SGD



Optimizer: RMSProp



Optimizer: Adam



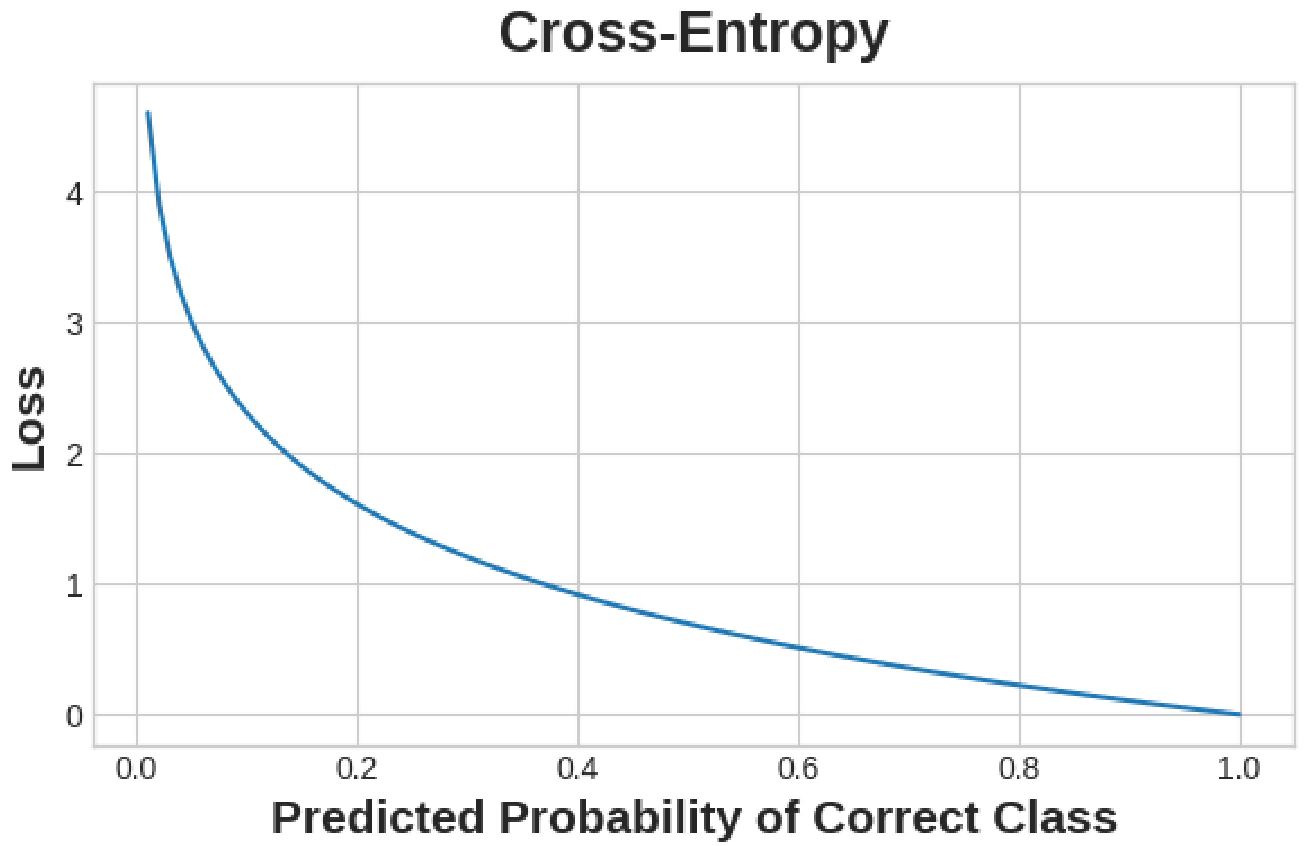
Keras – Part II

실습

https://github.com/JSJeong-me/Machine_Learning/blob/main/ML/Part2-deeplearning.ipynb

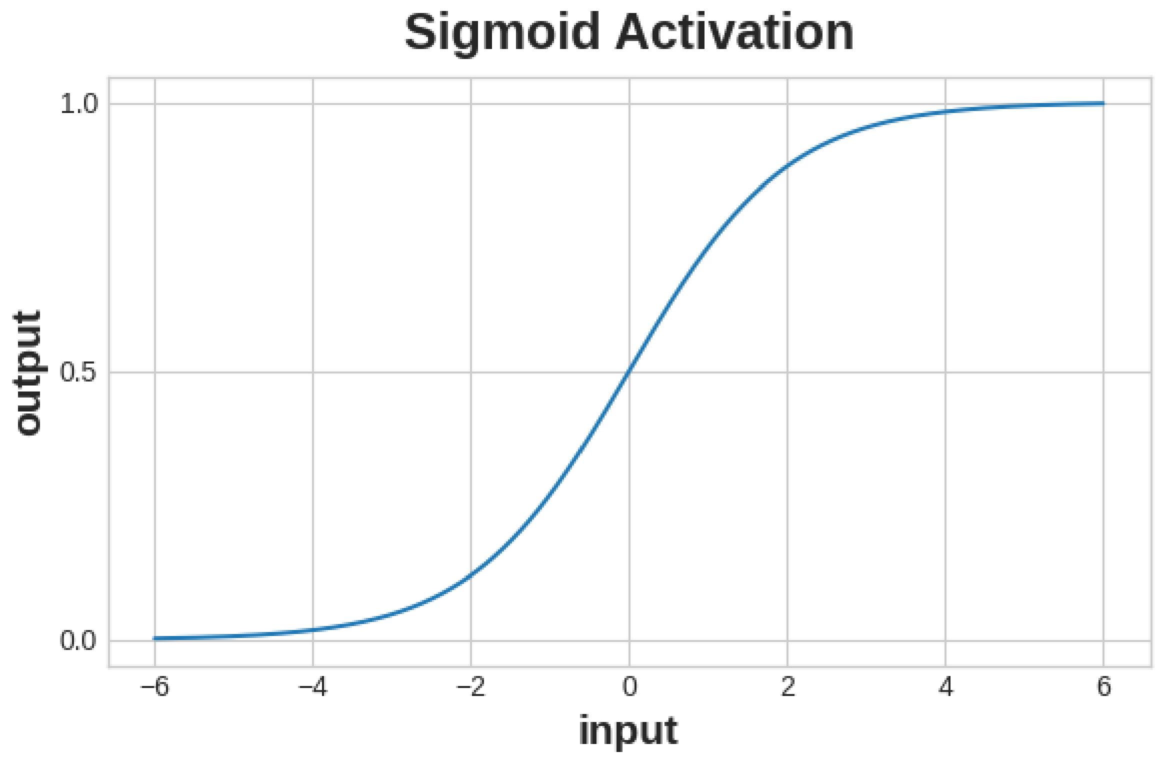
Cross-entropy

For classification, what we want instead is a distance between probabilities, and this is what cross-entropy provides. Cross-entropy is a sort of measure for the distance from one probability distribution to another.



Making Probabilities with the Sigmoid Function

The cross-entropy and accuracy functions both require probabilities as inputs, meaning, numbers from 0 to 1.



The sigmoid function maps real numbers into the interval [0,1] .

정 준 수 / Ph.D (jsjeong@hansung.ac.kr)

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: Computer Vision, 머신러닝(ML), RPA
- https://github.com/JSJeong-me/