Implementing Regression and Classification Using Neural Networks in scikit-learn

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Regression models using neural networks in scikit-learn

Mean square error (MSE) loss function

Using the MLPRegressor estimator to build regression models

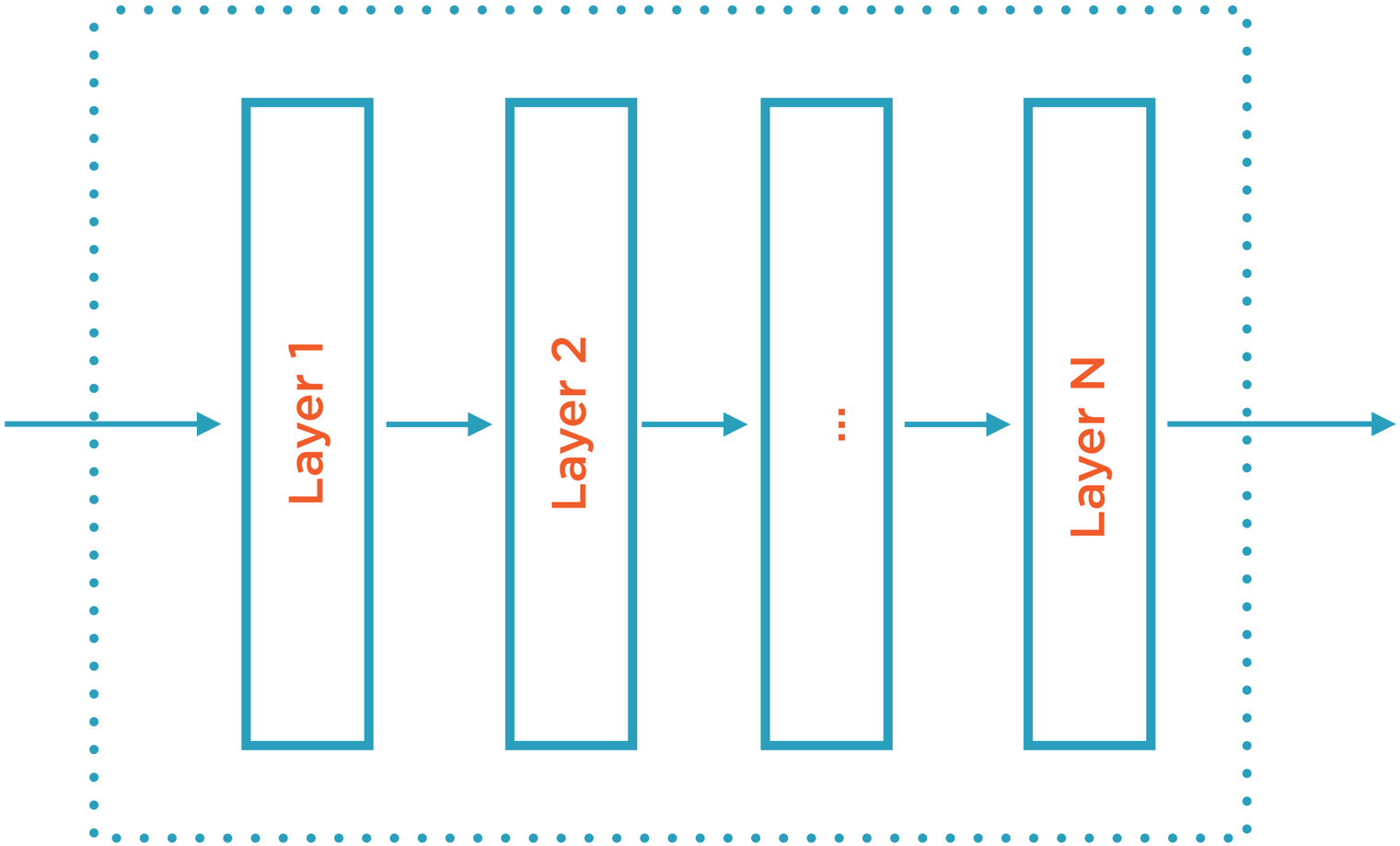
Classification models using neural networks in scikit-learn

Cross-entropy loss function

Using the MLPClassifier estimator to build classification models

Regression Using Neural Networks

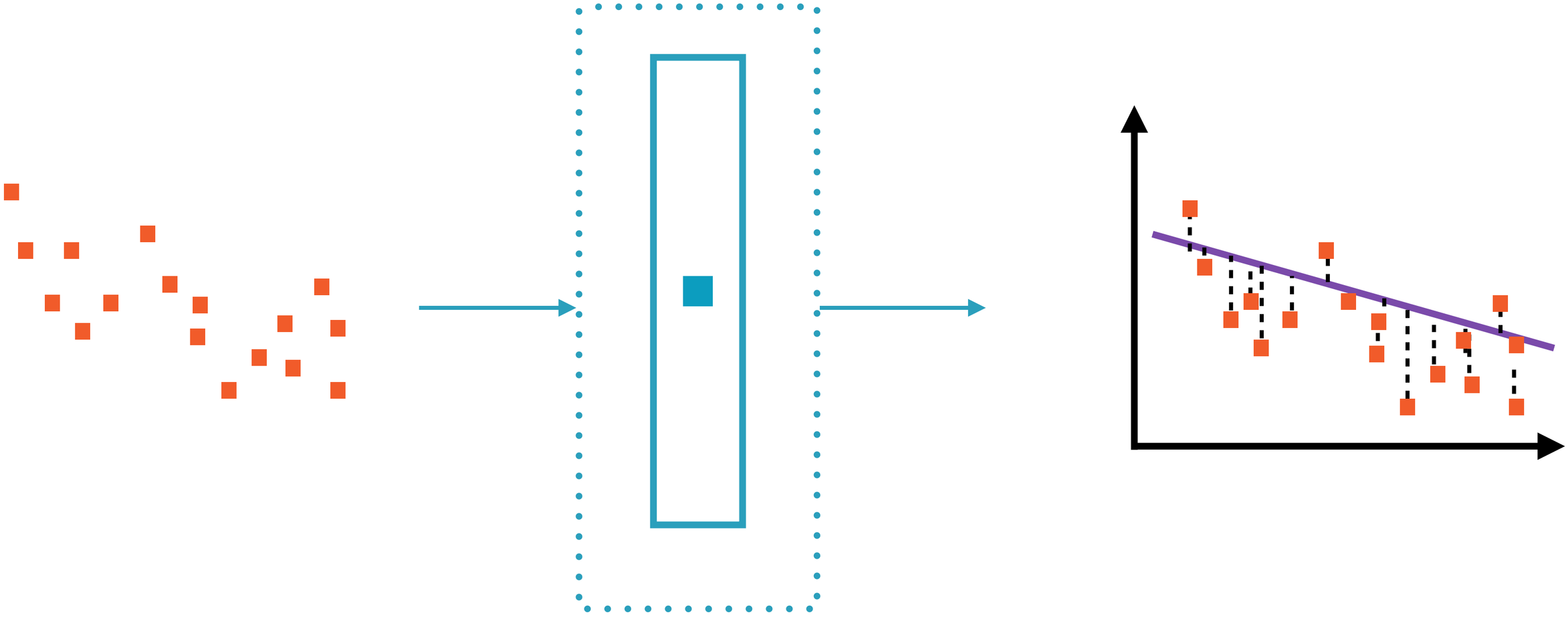
Neural Network Model



Network of interconnected layers

The **weights** and **biases** of individual neurons are determined during the **training** process

Regression: The Simplest Neural Network

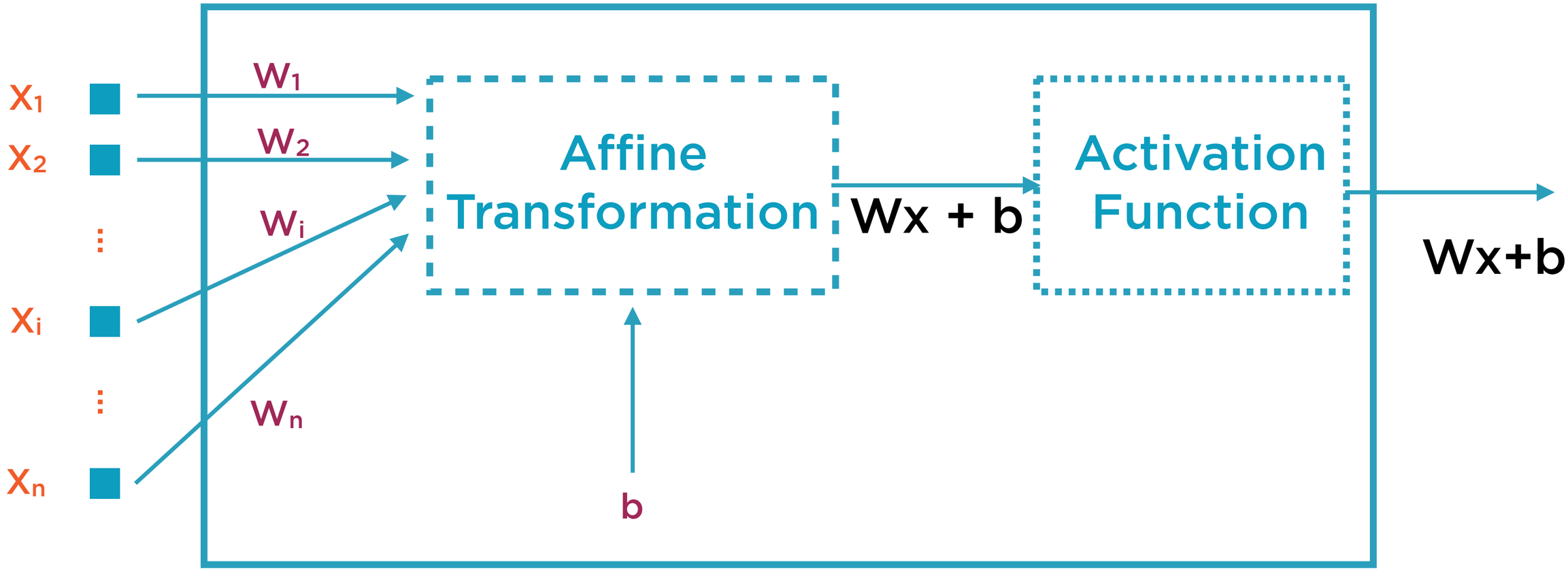


Set of Points

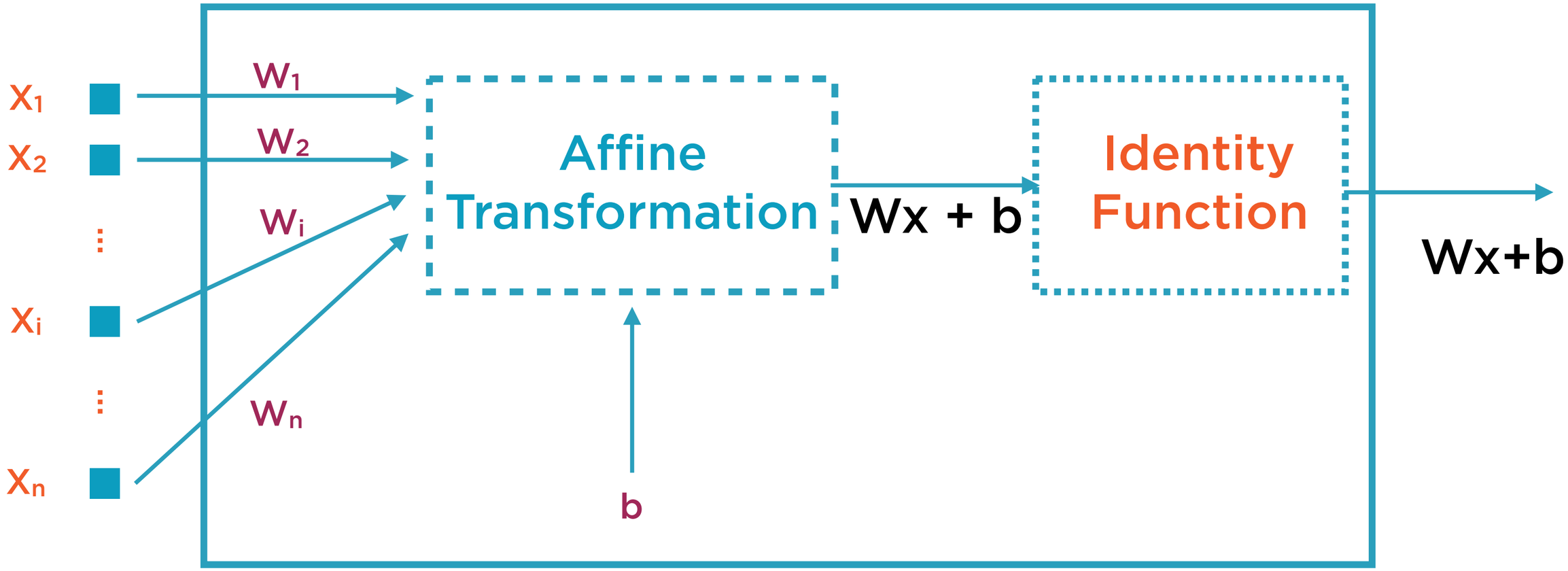
Single Neuron

Regression Line

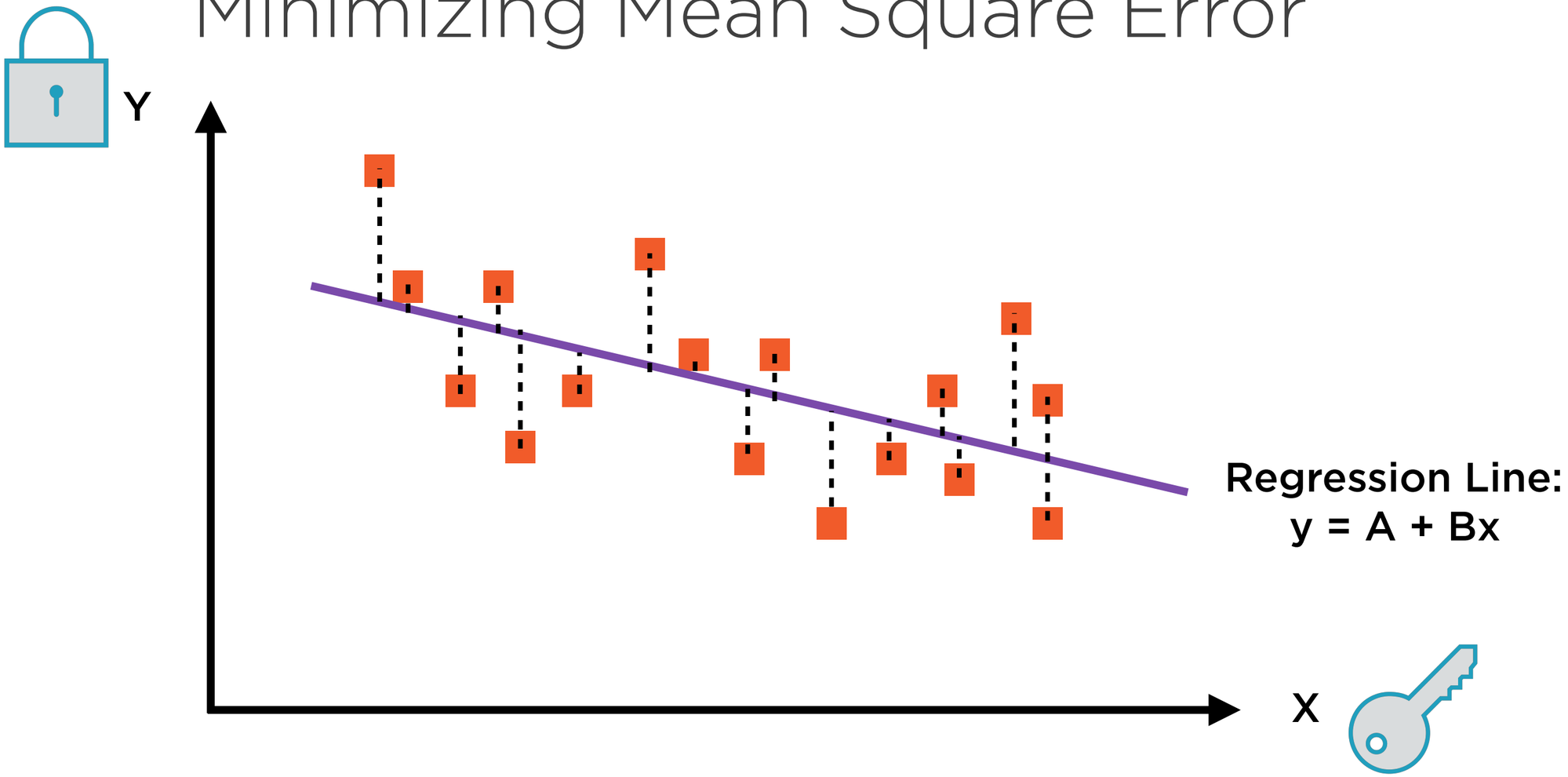
Regression: The Simplest Neural Network



Regression: The Simplest Neural Network



Minimizing Mean Square Error



The “best fit” line is called the regression line

The actual training of a neural network happens via Gradient Descent Optimization

Linear Regression as an Optimization Problem

Objective Function

Minimize variance of the residuals (MSE)

Linear Regression as an Optimization Problem

Objective Function

Minimize variance of the residuals (MSE)

Constraints

Express relationship as a straight line

$$y = Wx + b$$

Linear Regression as an Optimization Problem

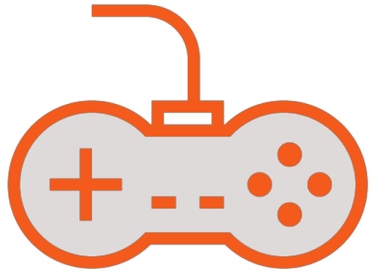
Objective Function

Minimize variance of the residuals (MSE)

Constraints

Express relationship as a straight line

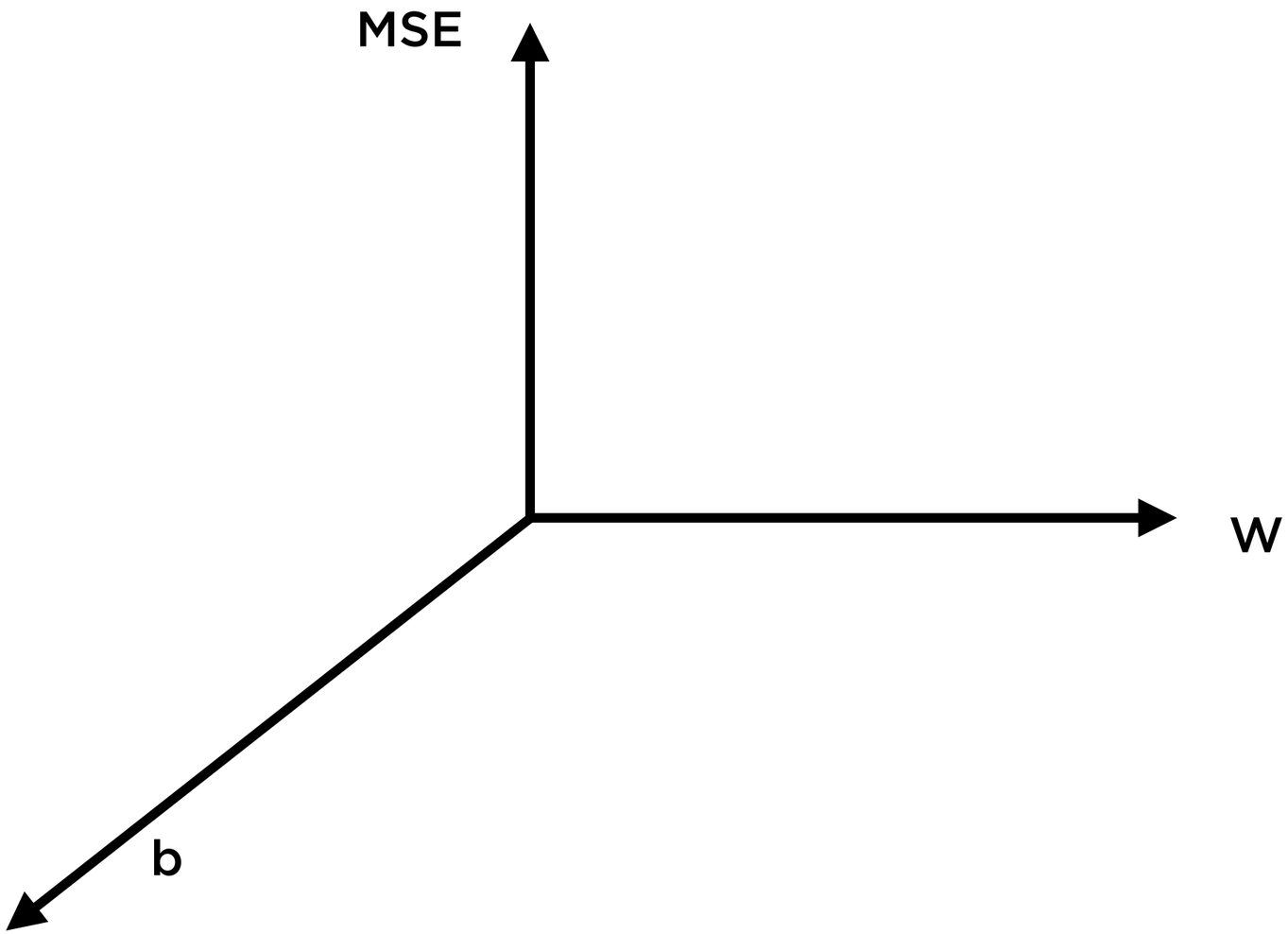
$$y = Wx + b$$



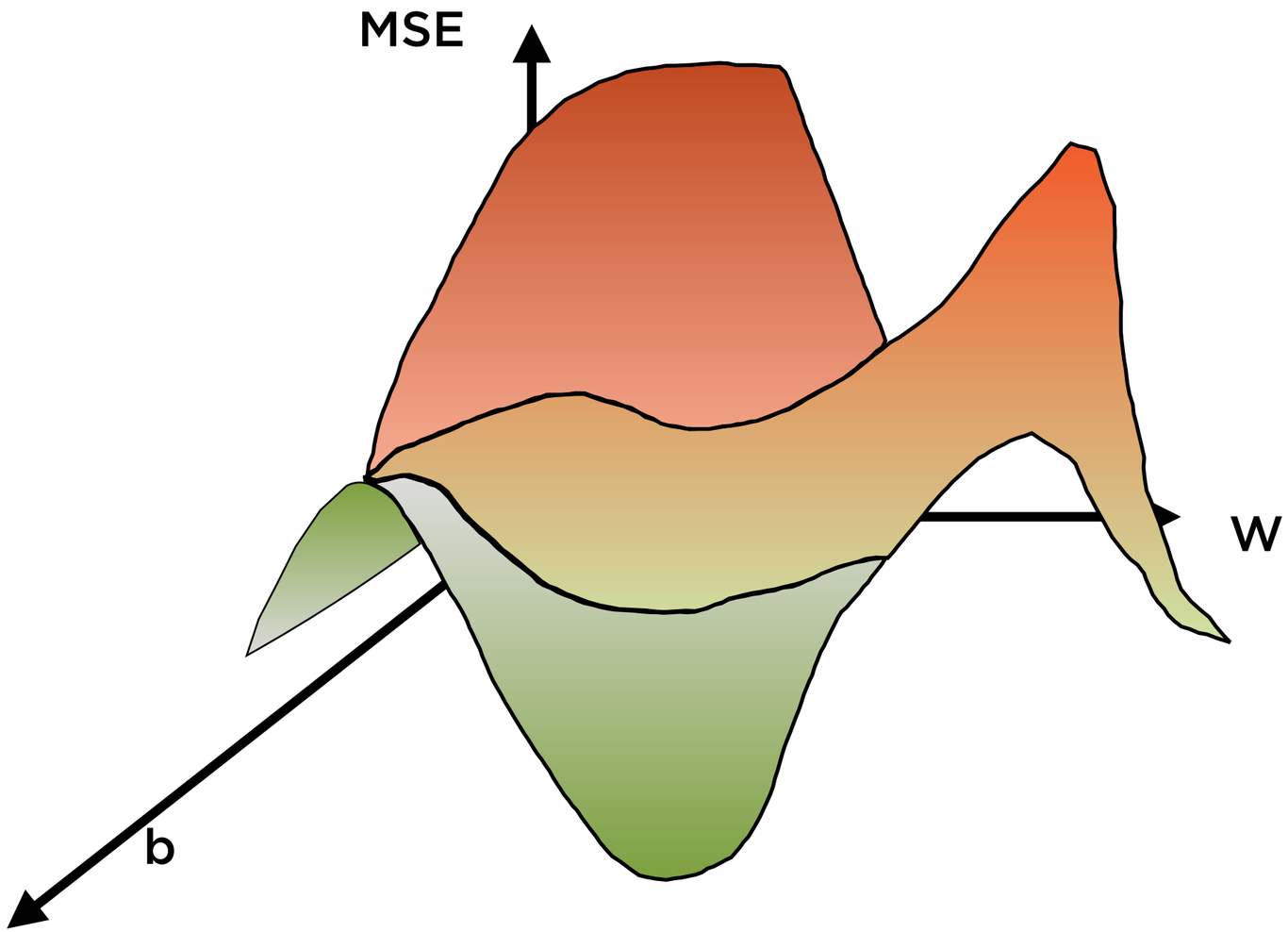
Decision Variables

Values of W and b

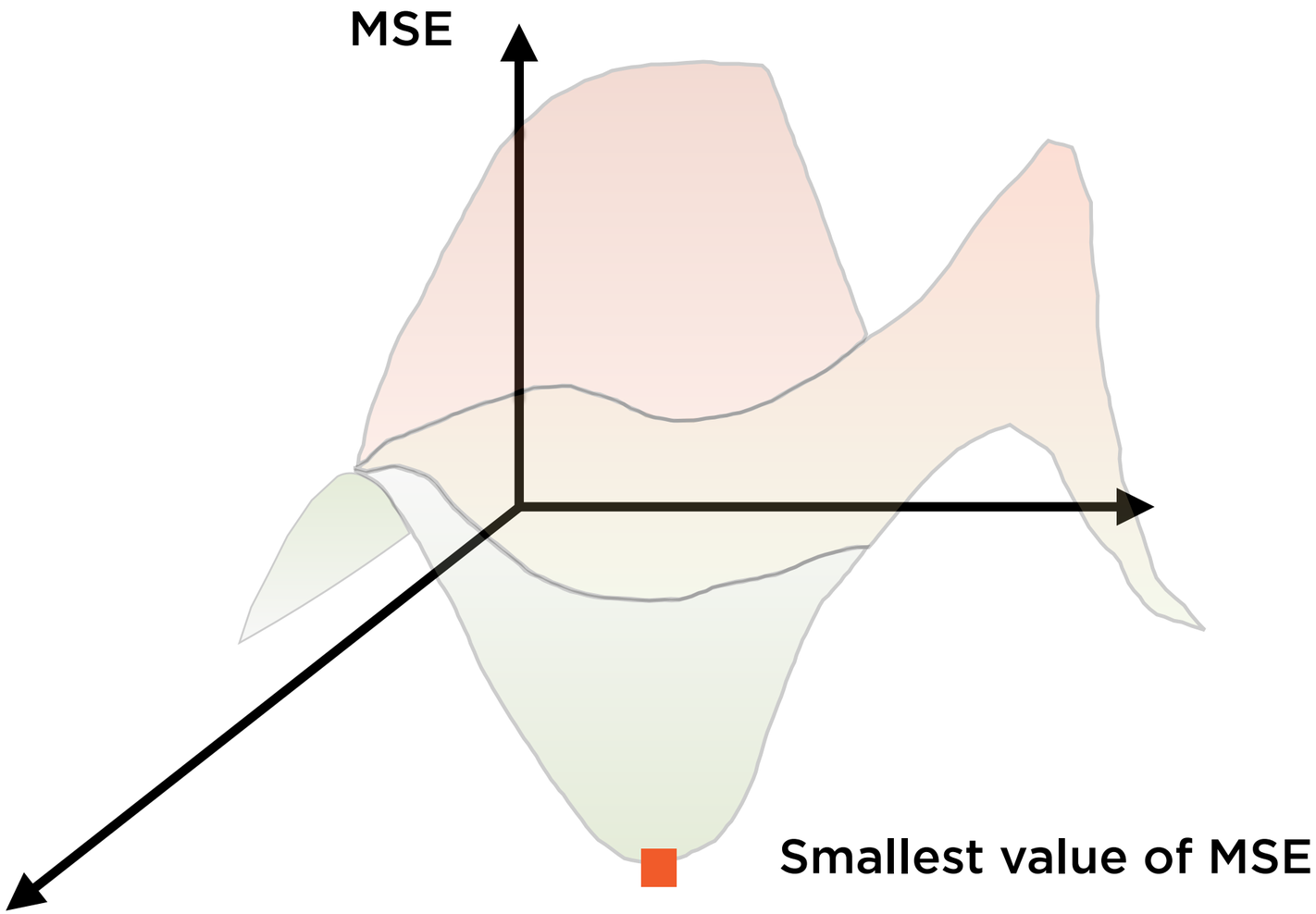
Minimizing MSE



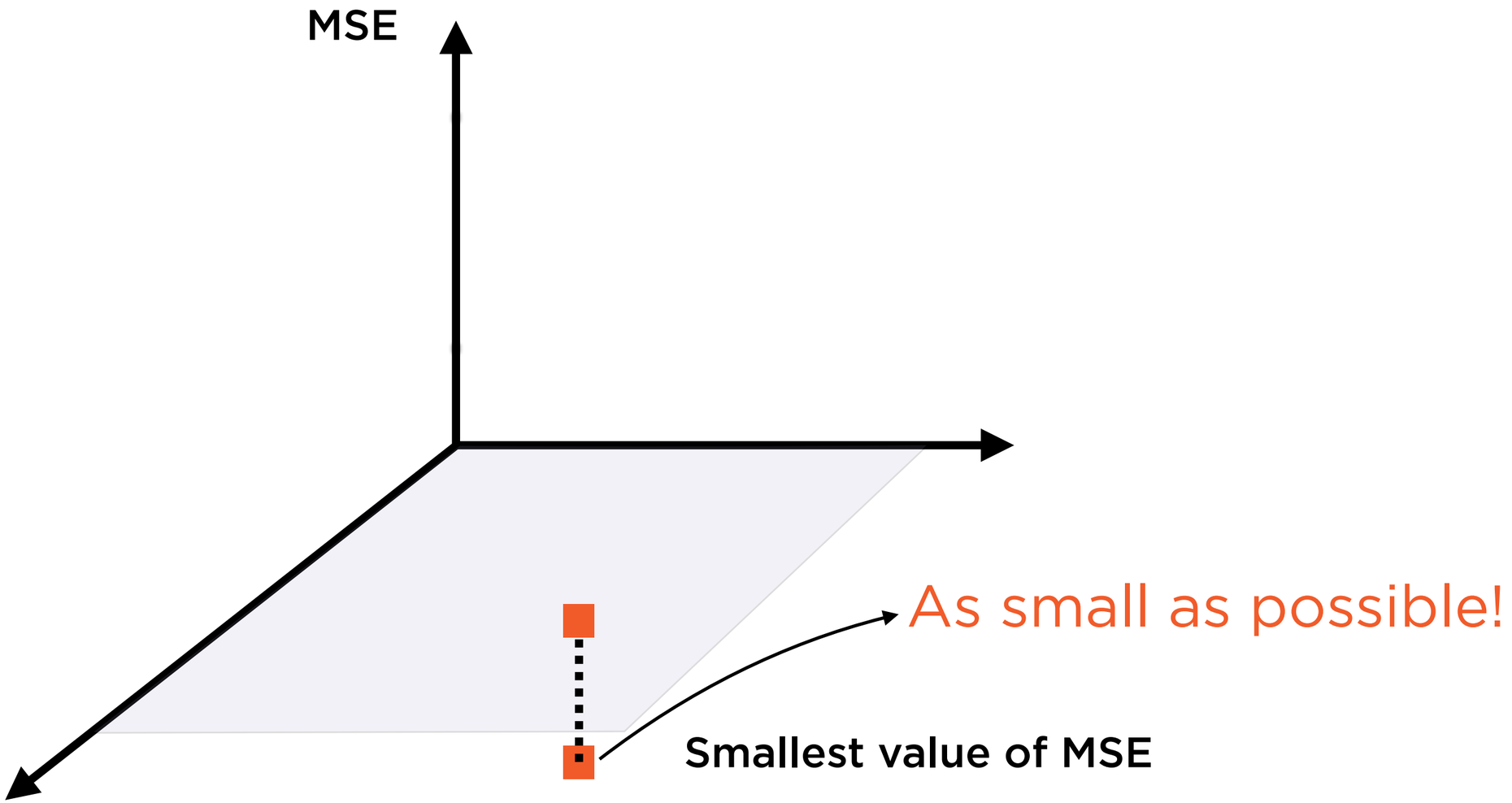
Minimizing MSE



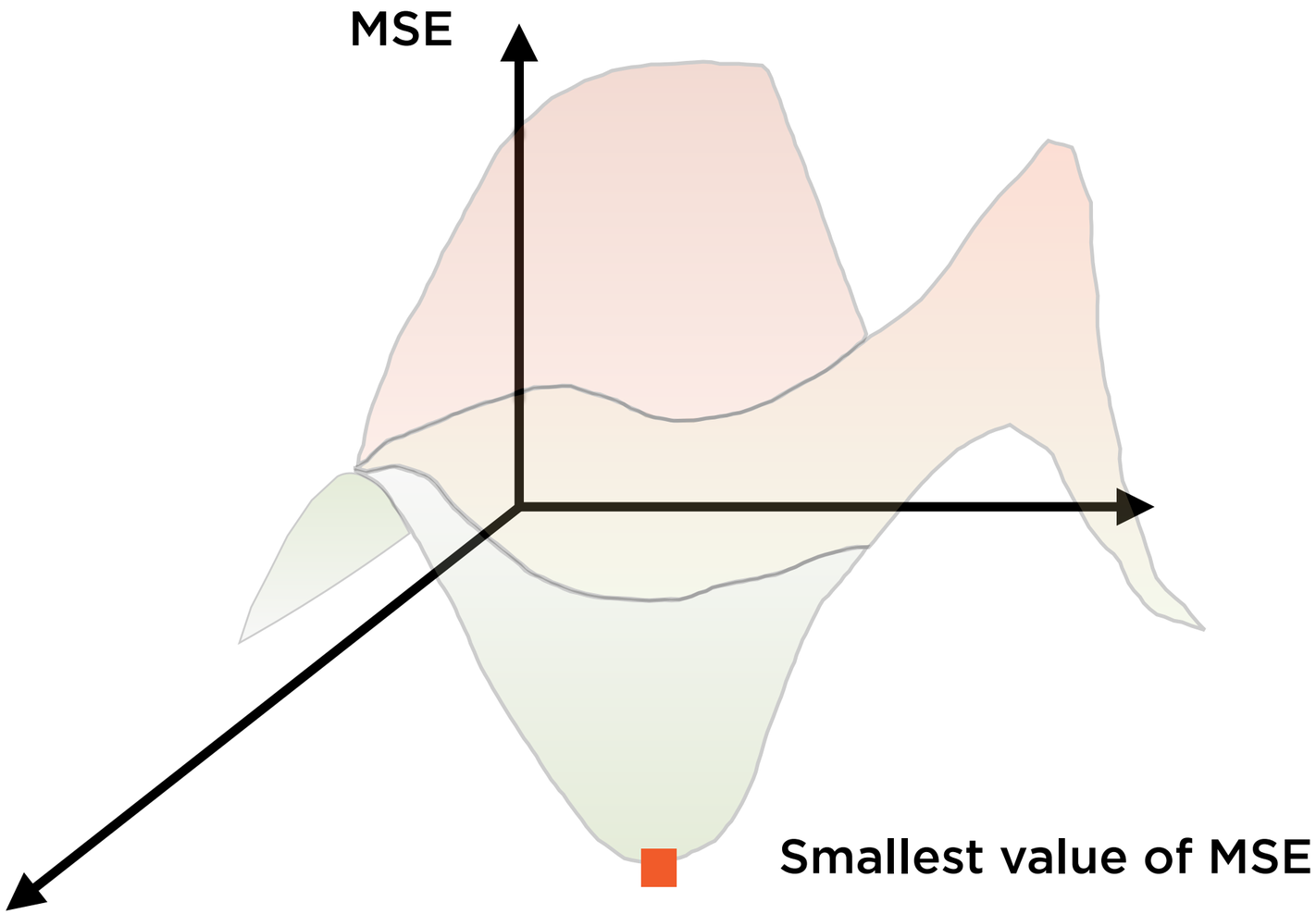
Minimizing MSE



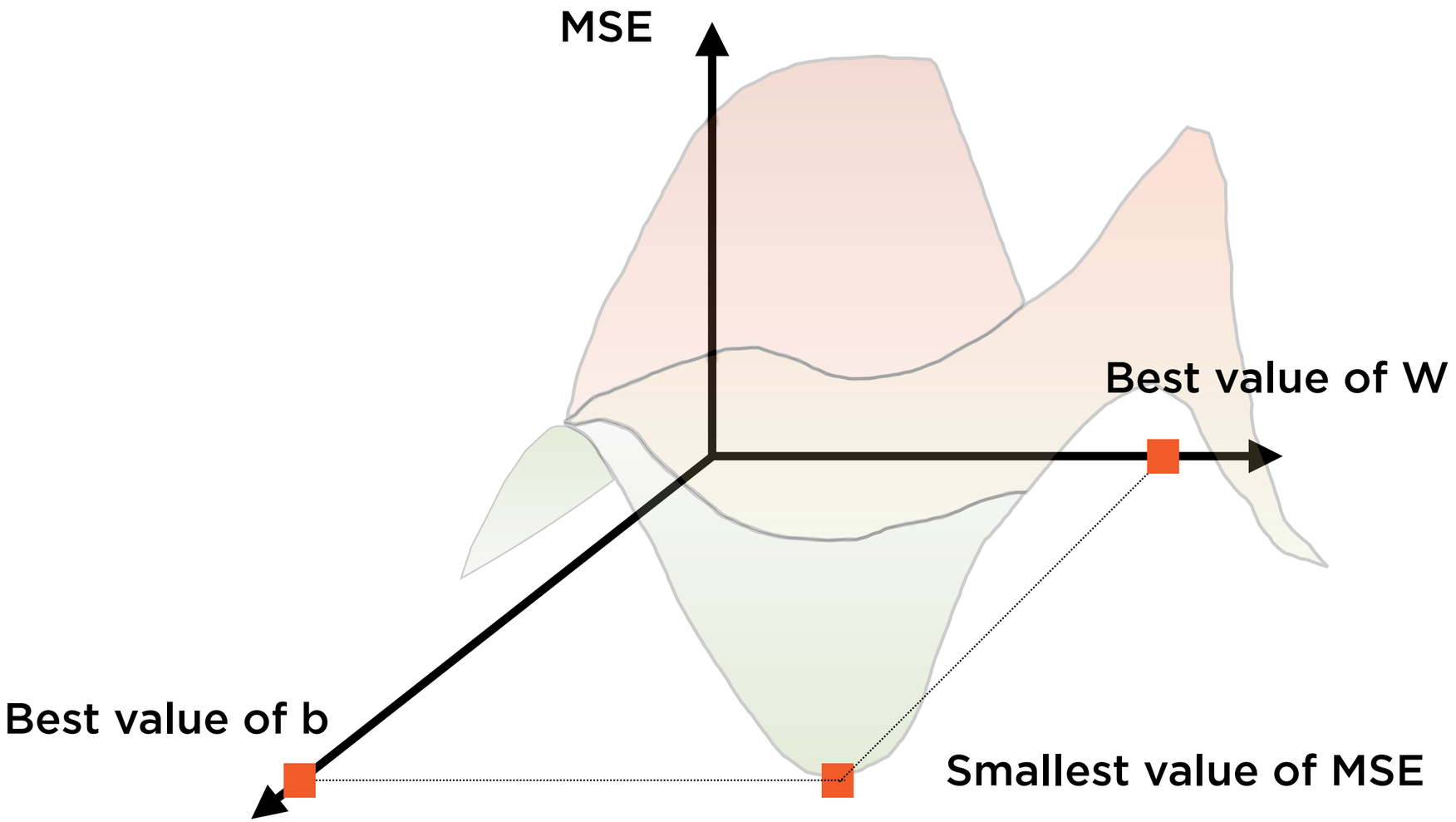
Minimizing MSE



Minimizing MSE

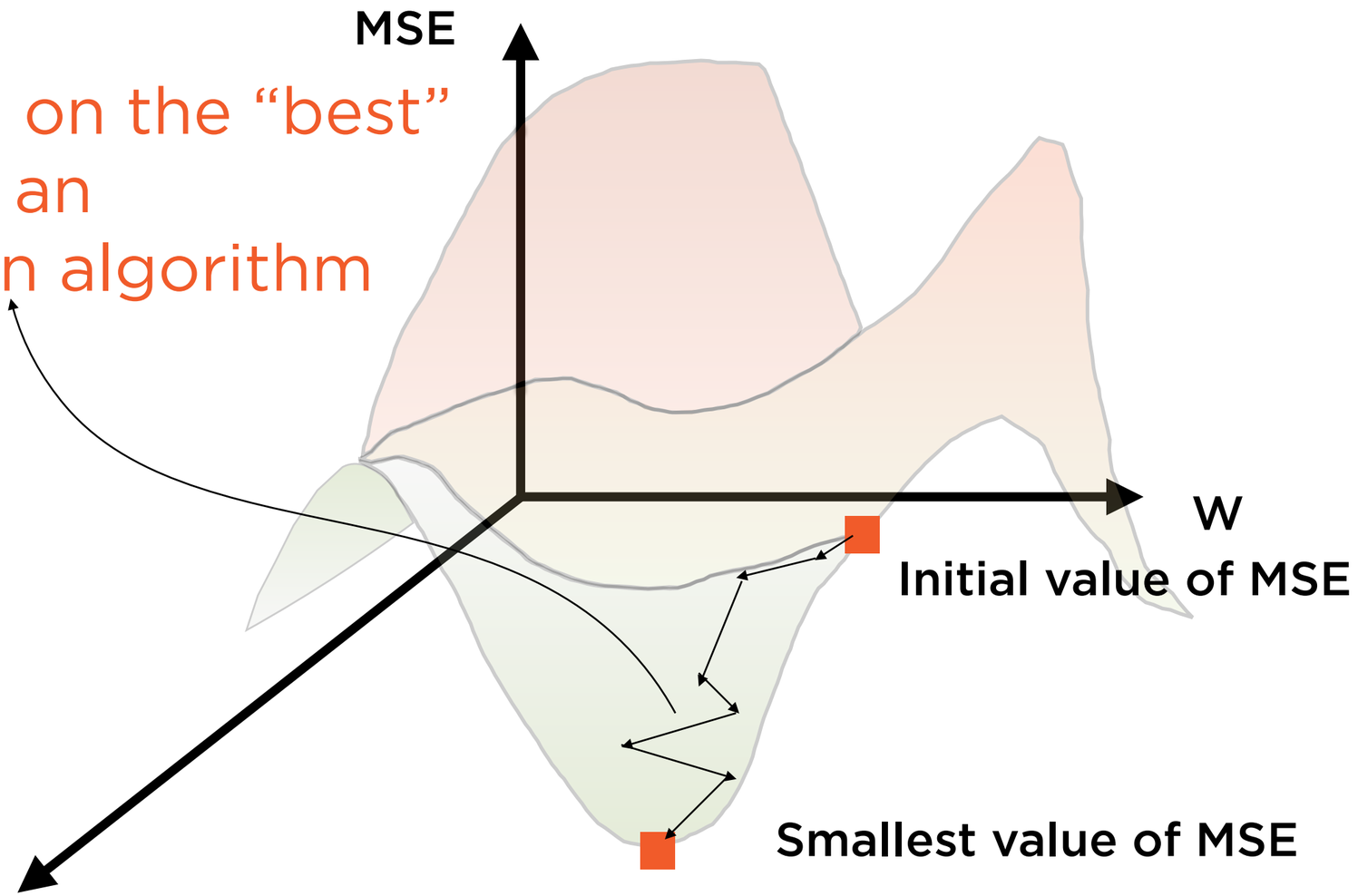


Minimizing MSE

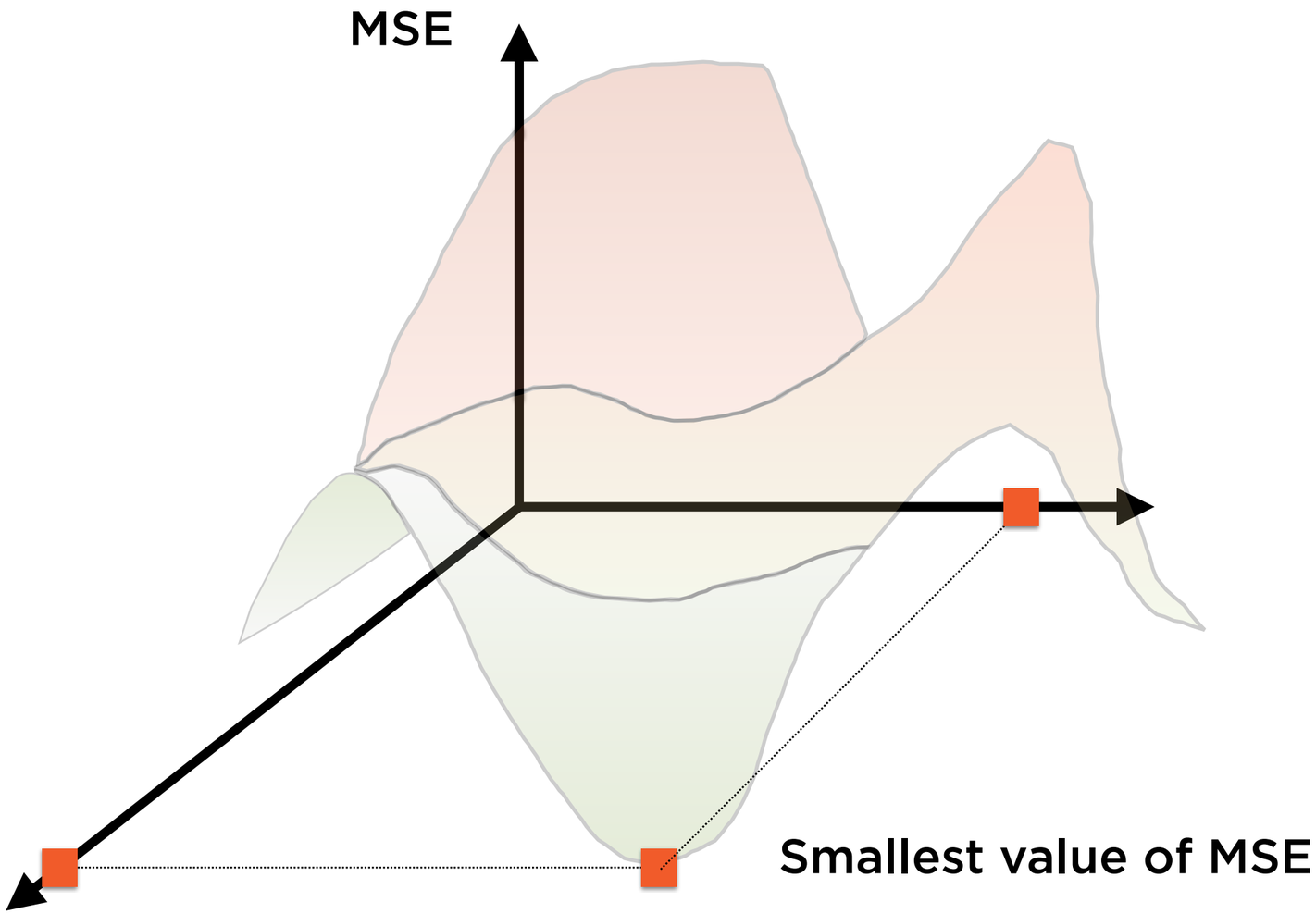


“Gradient Descent”

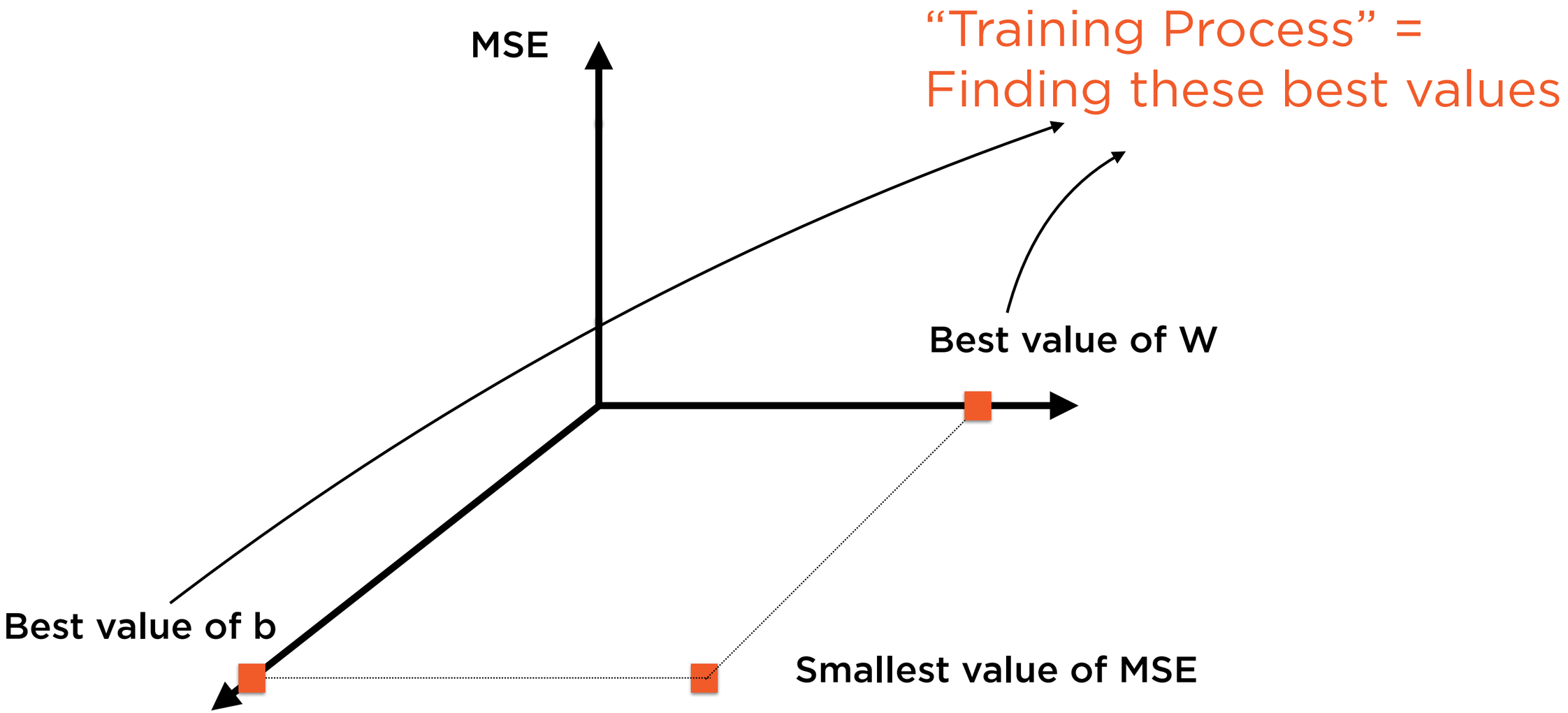
Converging on the “best” value using an optimization algorithm



Minimizing MSE



“Training” the Algorithm



$$Parameters^{t+1} = Parameters^{t} - learning\_rate \times Gradient(\theta^{t})$$

Basic SGD Optimizer

Move each parameter value in the direction of reducing gradient

$$Parameters^{t+1} = Parameters^{t} - \mathbf{learning\_rate} \times Gradient(\theta^{t})$$

Basic SGD Optimizer

The size of the step in the direction of reducing gradients

$$momentum\_vec^{t} = momentum\_coeff + learning\_rate \times Gradient(\theta)^{t}$$

$$Parameters^{t+1} = Parameters^{t} - momentum\_vec$$

Momentum-based Optimizer

Momentum vector helps accelerate in the direction where gradient is decreasing

$$momentum\_vec^{t} = momentum\_coeff + learning\_rate \times Gradient(\theta)^{t}$$

$$Parameters^{t+1} = Parameters^{t} - momentum\_vec$$

Momentum-based Optimizer

Gradients at each step weighted by those in previous step

Benefit: Faster convergence

$$momentum\_vec^{t} = momentum\_coeff + learning\_rate \times Gradient(\theta)^{t}$$

$$Parameters^{t+1} = Parameters^{t} - momentum\_vec$$

Momentum-based Optimizer

Need a momentum coefficient, between 0 and 1 to prevent overshooting

scikit-learn Optimizers

Relatively small number of optimizers supported

Plain-vanilla SGD optimizer

Adam optimizer

L-BFGS optimizer

Neural Networks in scikit-learn

Multi-layer Perceptrons (MLP)

Restricted Boltzmann Machines (RBM)

Neural Networks in scikit-learn

Multi-layer Perceptrons (MLP)

Restricted Boltzmann Machines (RBM)

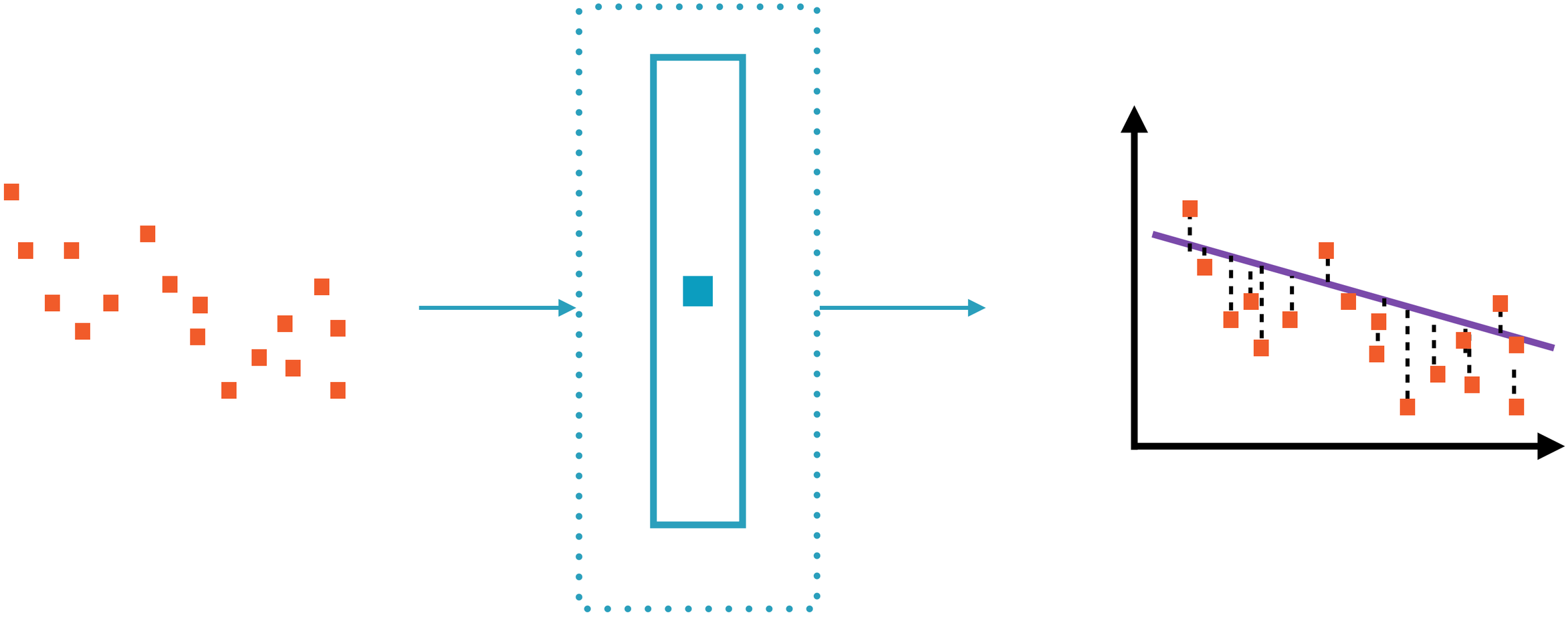
MLPRegressor estimator object to build regression models

Demo

Build and train a regression model using neural networks in scikit-learn using the MLPRegressor estimator

Classification Using Neural Networks

Linear Regression with One Neuron

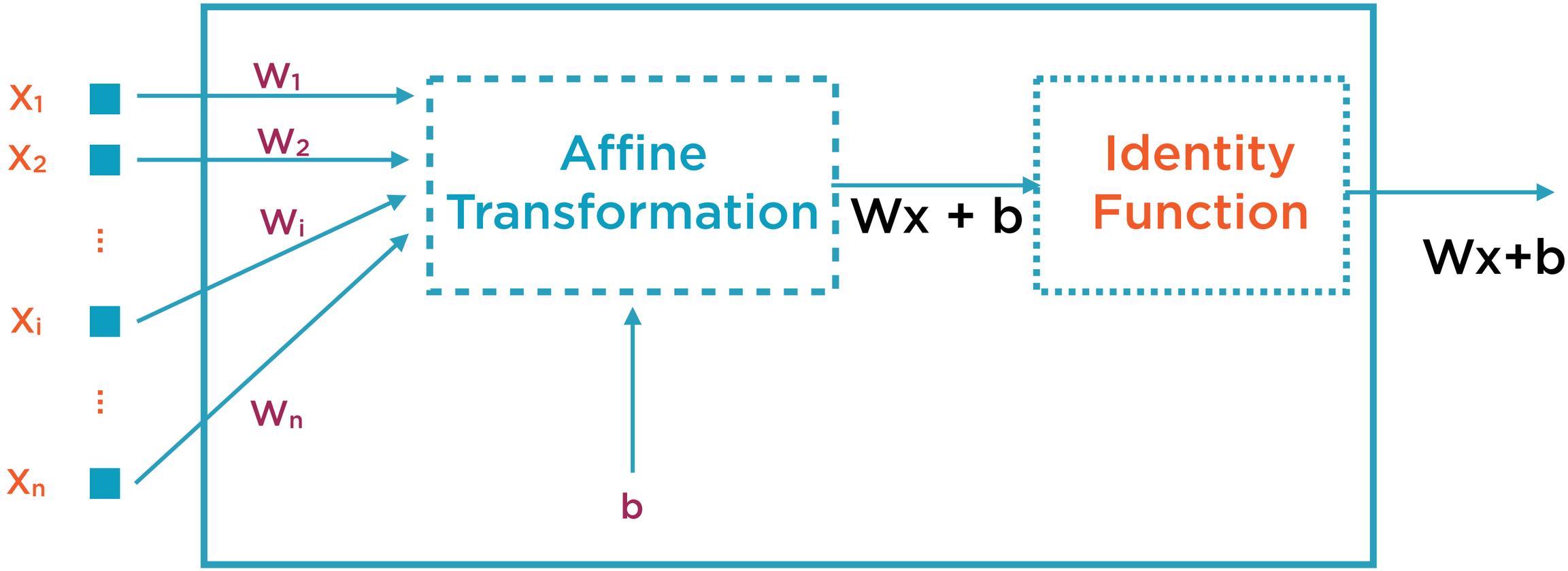


Set of Points

Single Neuron

Regression Line

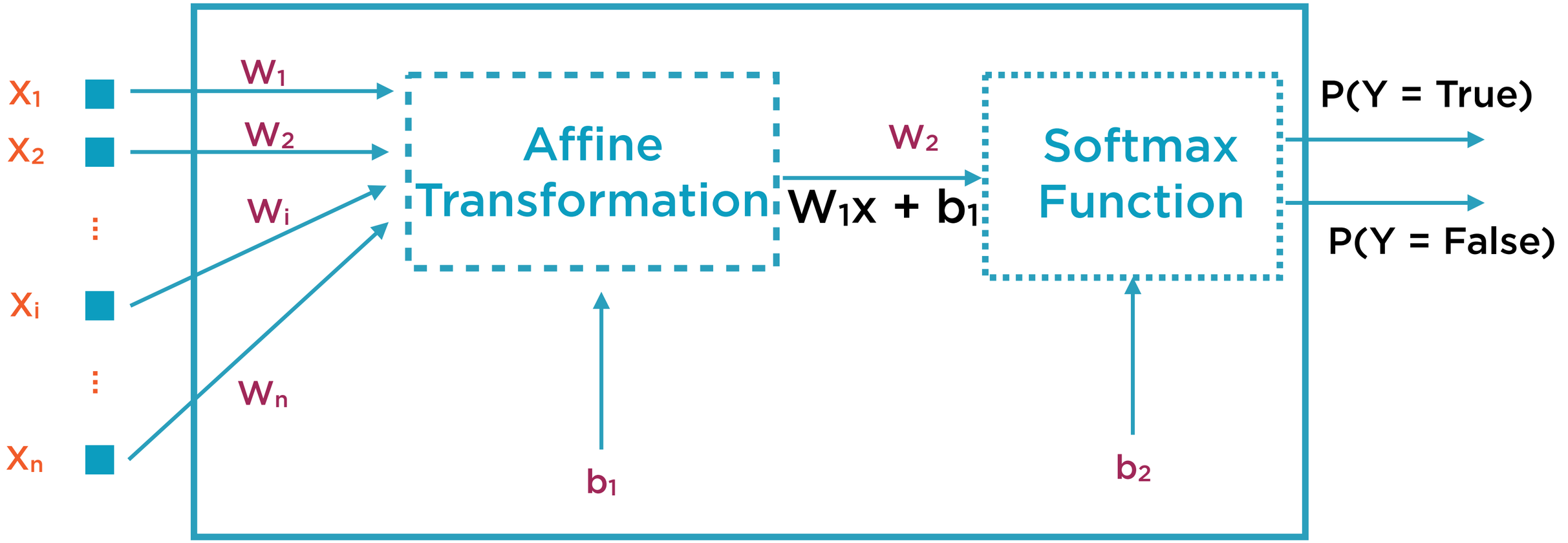
Linear Regression with One Neuron



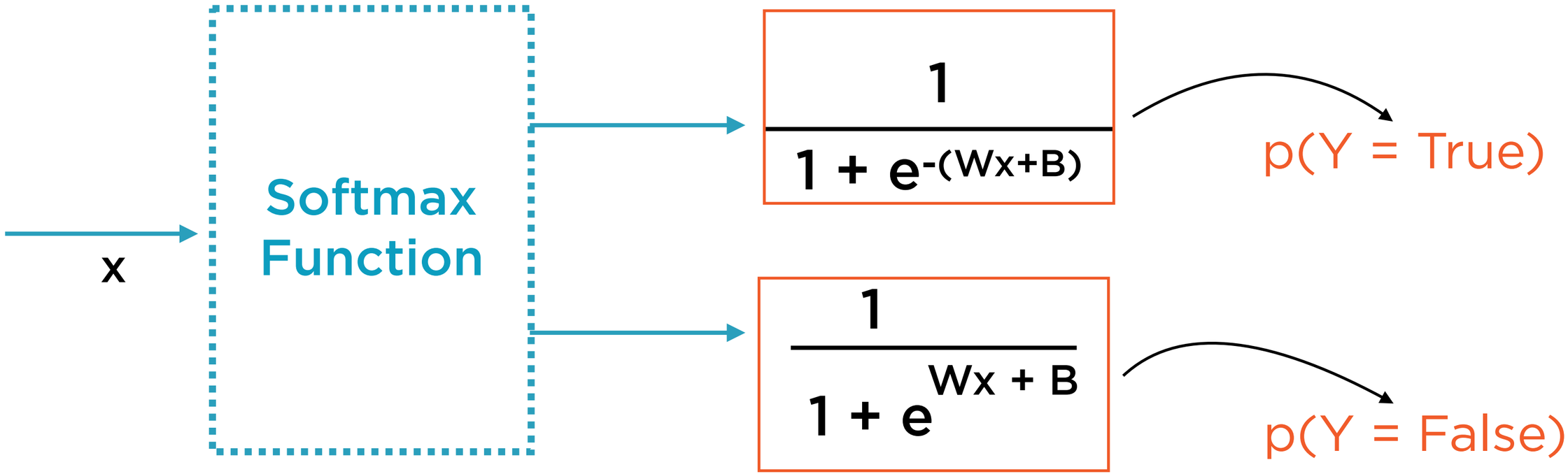
Linear Regression with One Neuron

Linear Classification with One Neuron

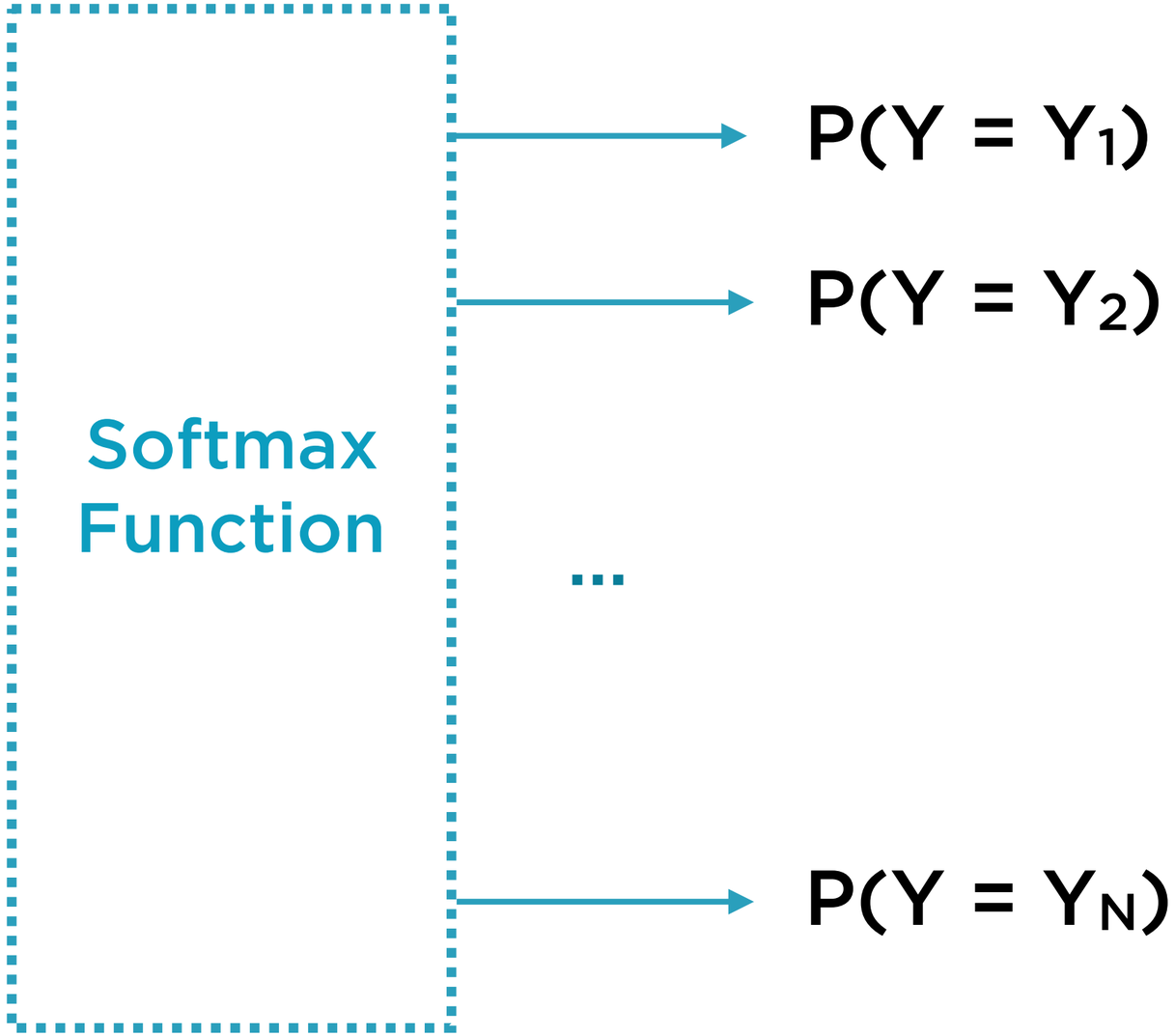
Linear Classification with One Neuron



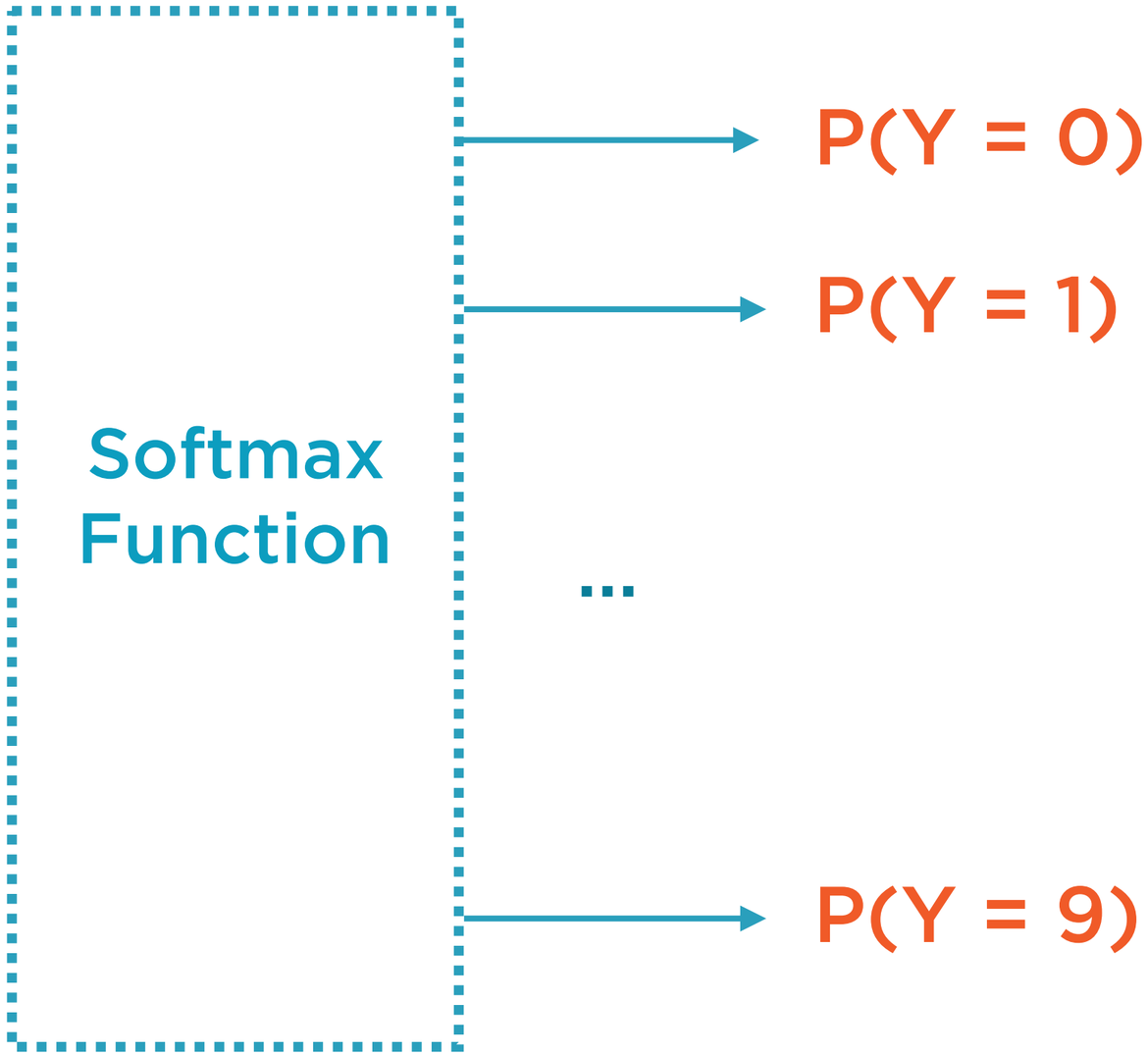
SoftMax for True/False Classification



SoftMax N-category Classification

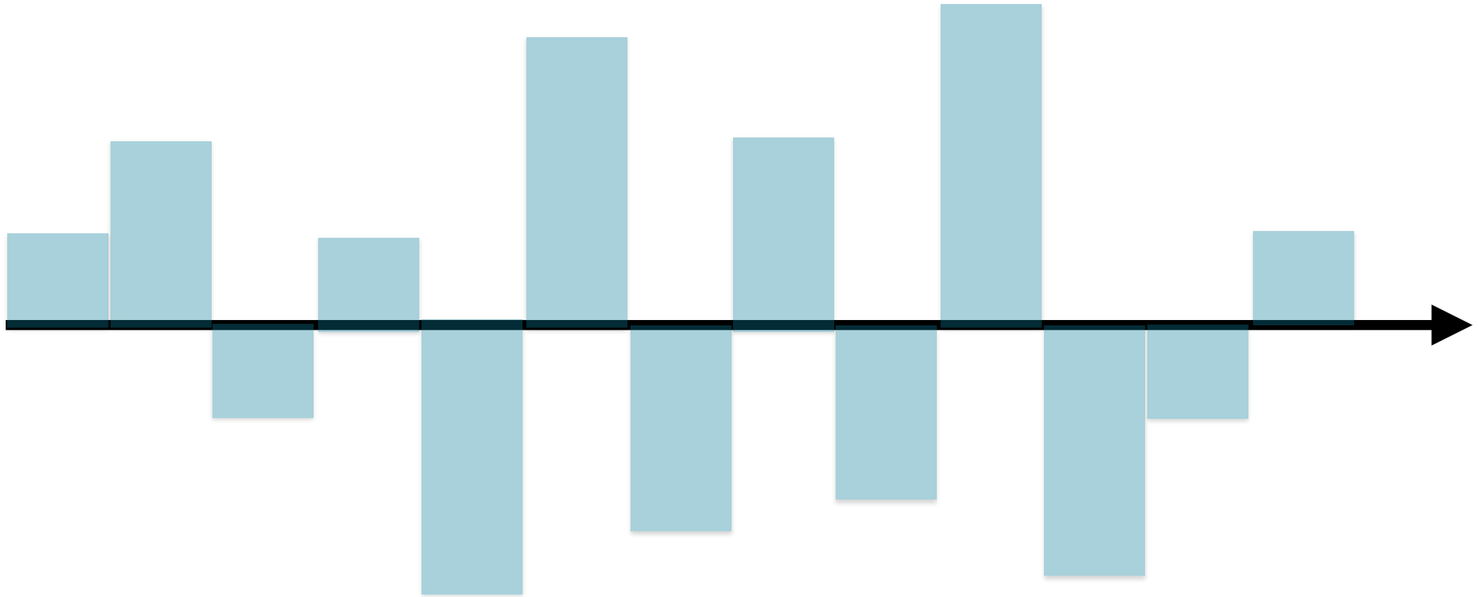


SoftMax for Digit Classification

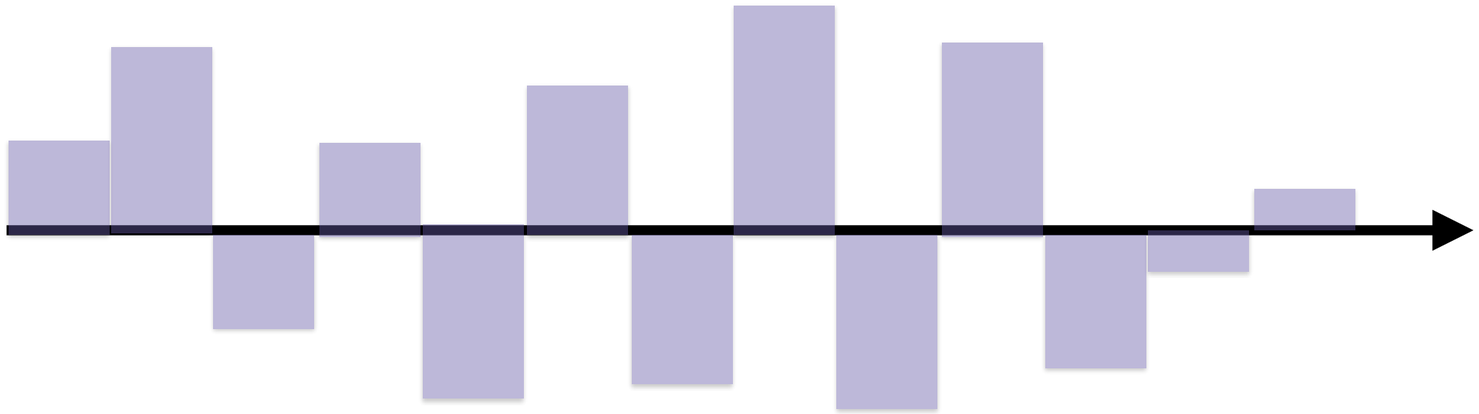


Intuition: Low Cross Entropy

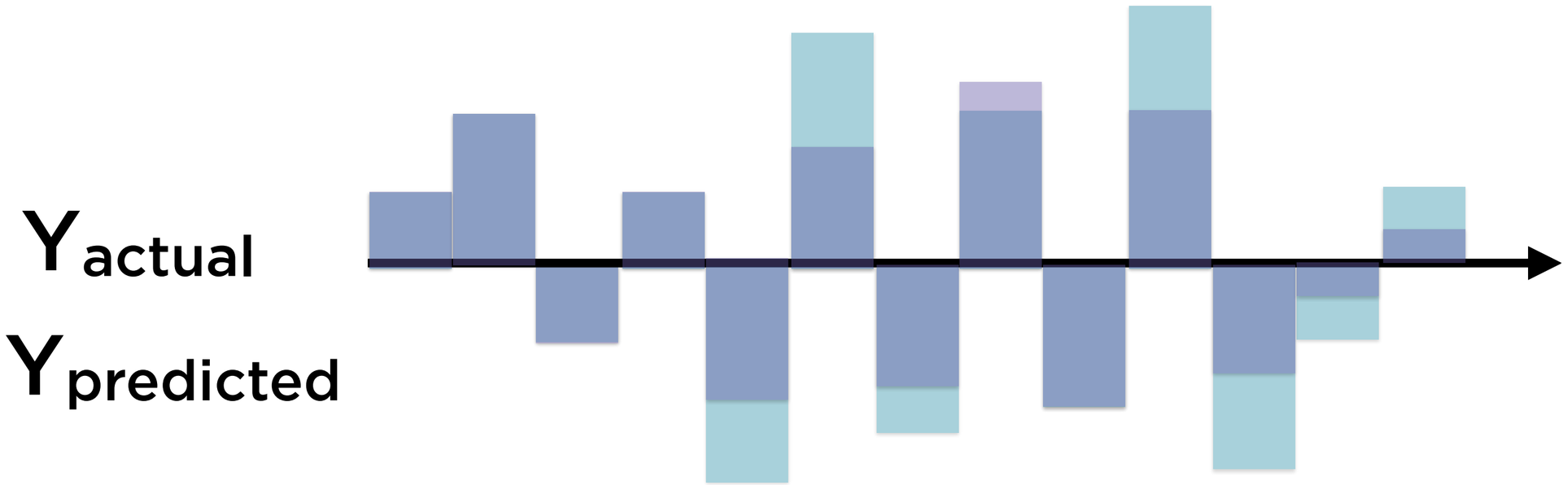
Y_{actual}



$Y_{predicted}$



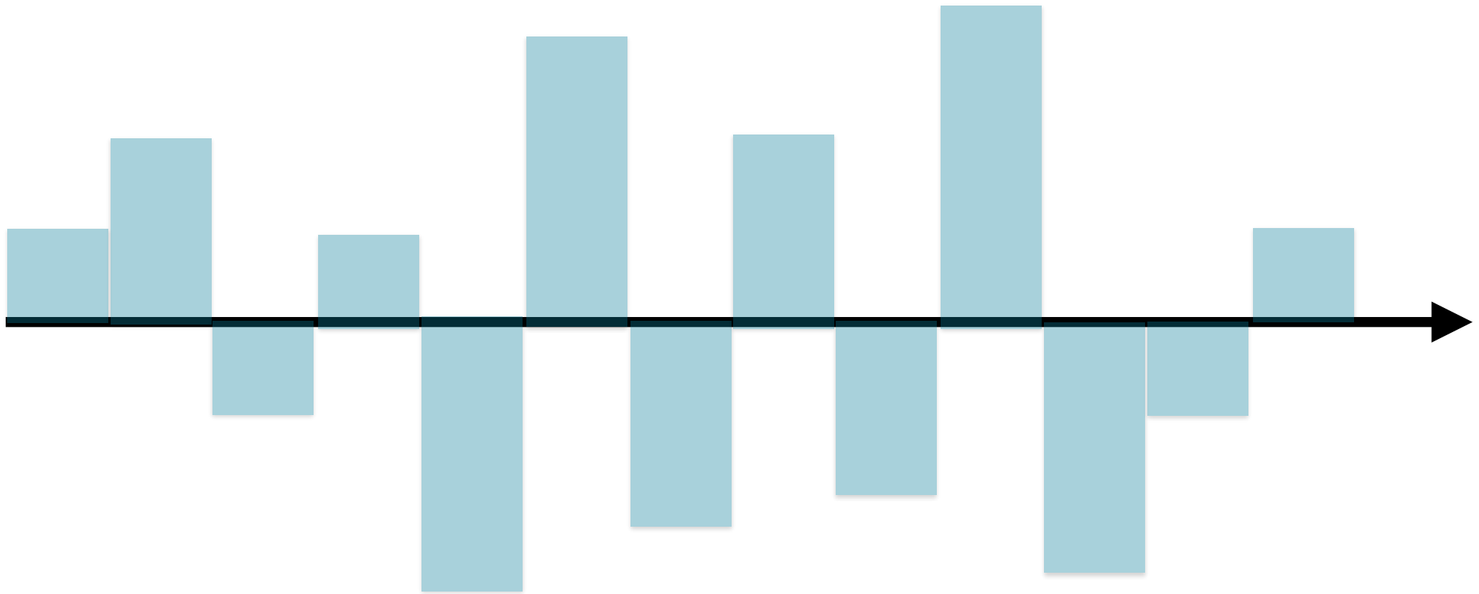
Intuition: Low Cross Entropy



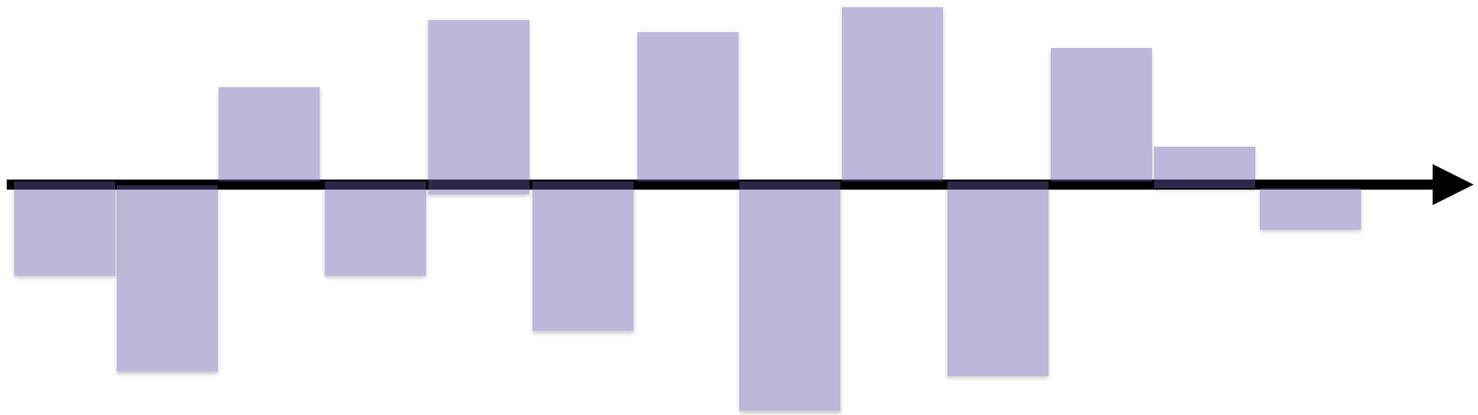
The labels of the two series are in-synch

Intuition: High Cross Entropy

Y_{actual}



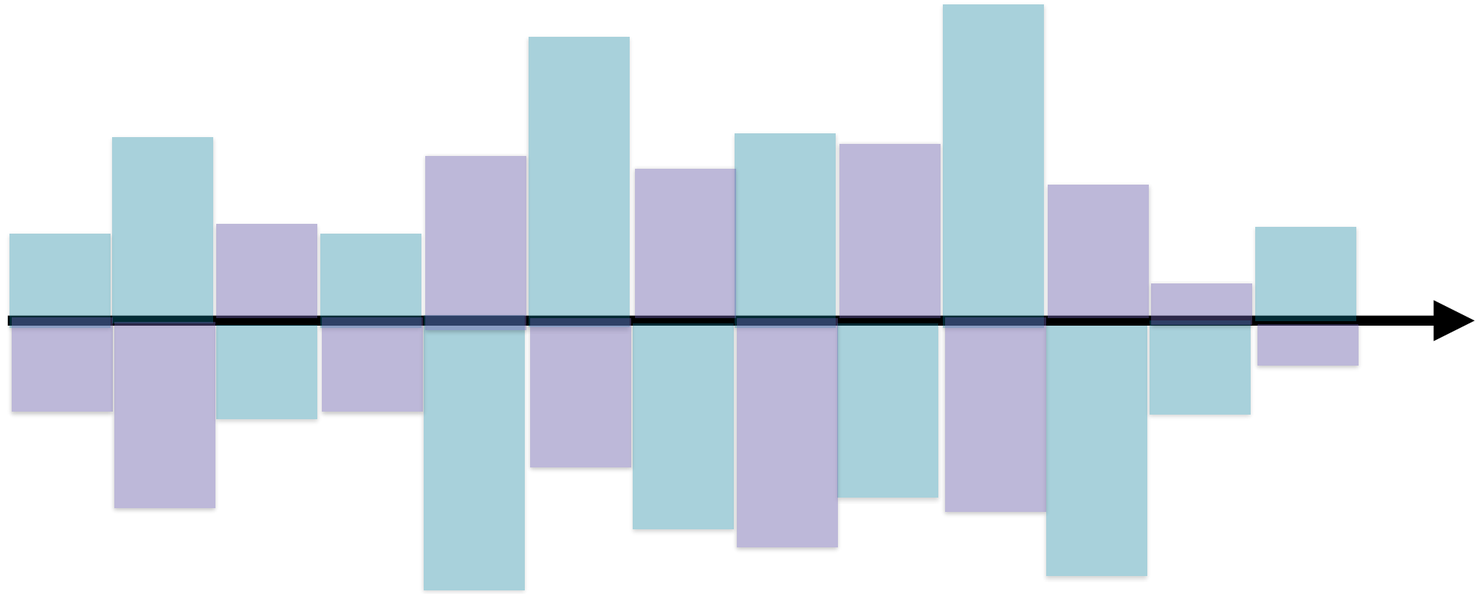
$Y_{predicted}$



Intuition: High Cross Entropy

Y_{actual}

$Y_{predicted}$



The labels of the two series are out-of-synch

Demo

Build and train a classification model using neural networks in scikit-learn using the MLPClassifier estimator

Summary

Regression models using neural networks in scikit-learn

Mean square error (MSE) loss function

Using the MLPRegressor estimator to build regression models

Classification models using neural networks in scikit-learn

Cross-entropy loss function

Using the MLPClassifier estimator to build classification models