Understanding Linear Regression with a Single Neuron

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

Overview

Using linear regression for prediction

Linear regression using a single neuron

Hand-crafting an MSE regression model

Hand-crafting a Ridge regression model

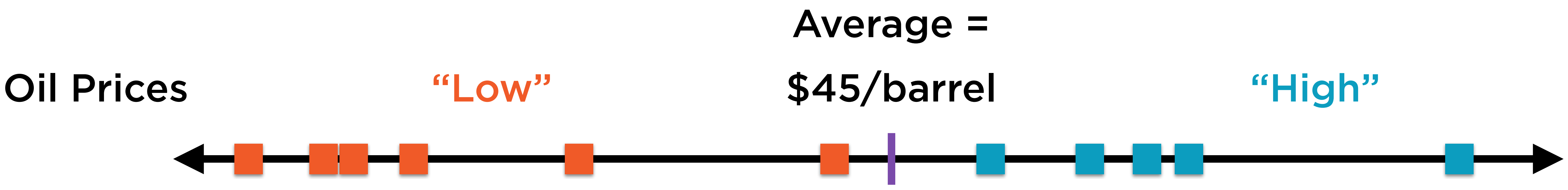
Comparing to scikit-learn’s linear regression estimator

Linear Regression

Data in One Dimension

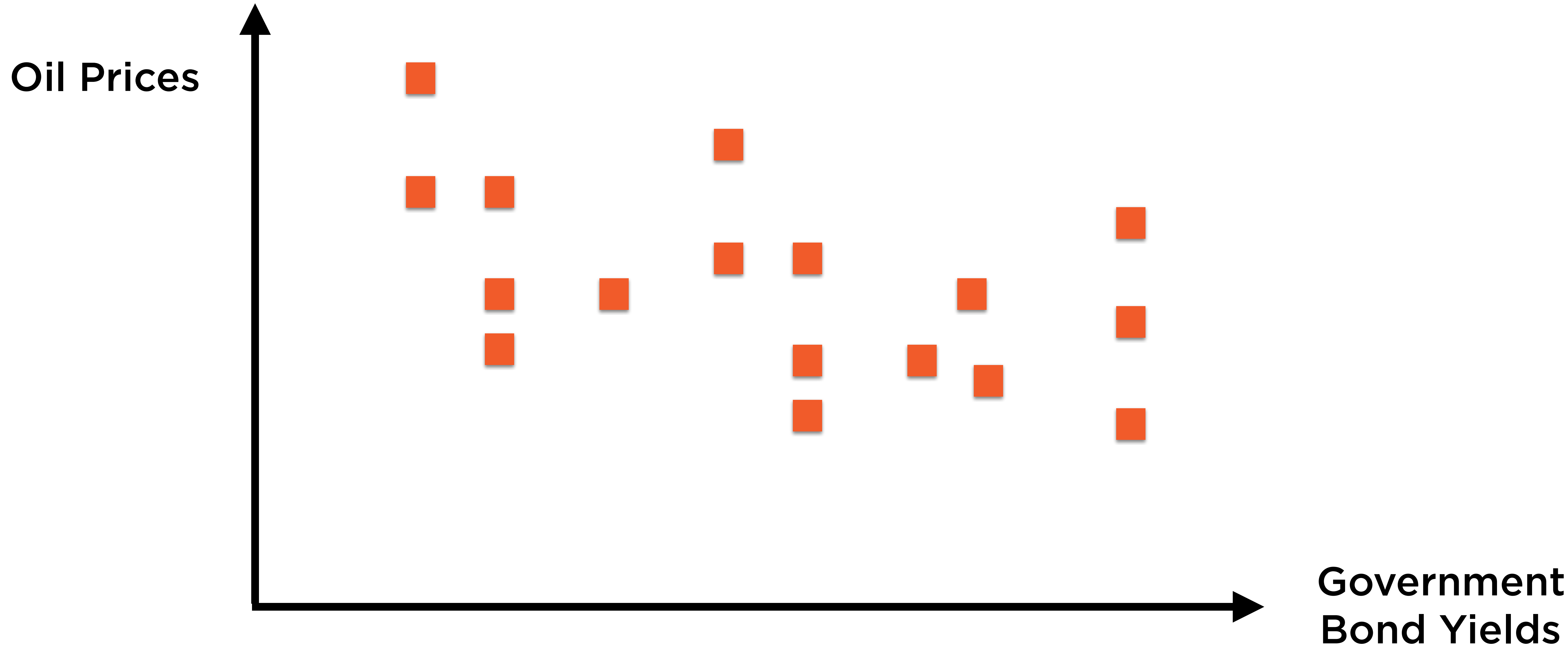
Unidimensional data points can be represented using a line, such as a number line

Data in One Dimension



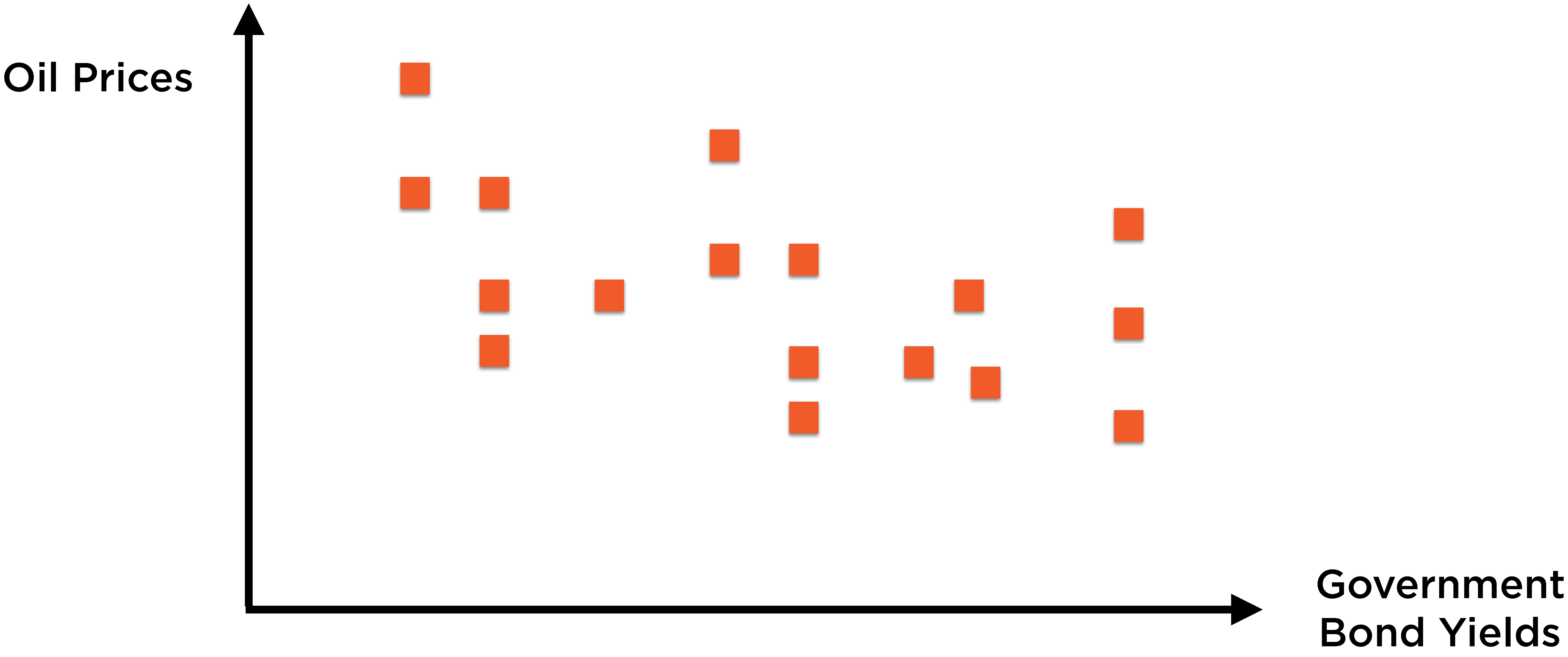
Unidimensional data is analysed using statistics such as mean, median, standard deviation

Data in Two Dimensions



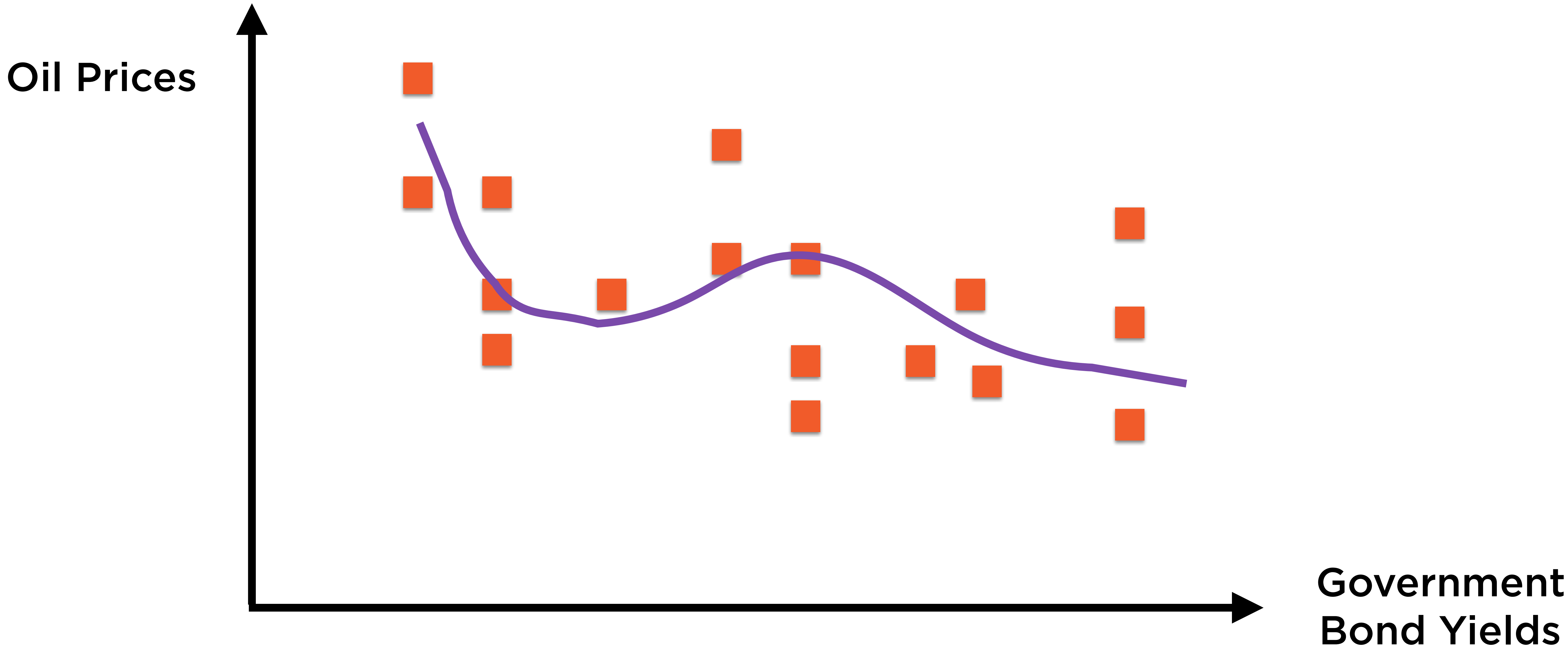
Its often more insightful to view data in relation to some other, related data

Data in Two Dimensions



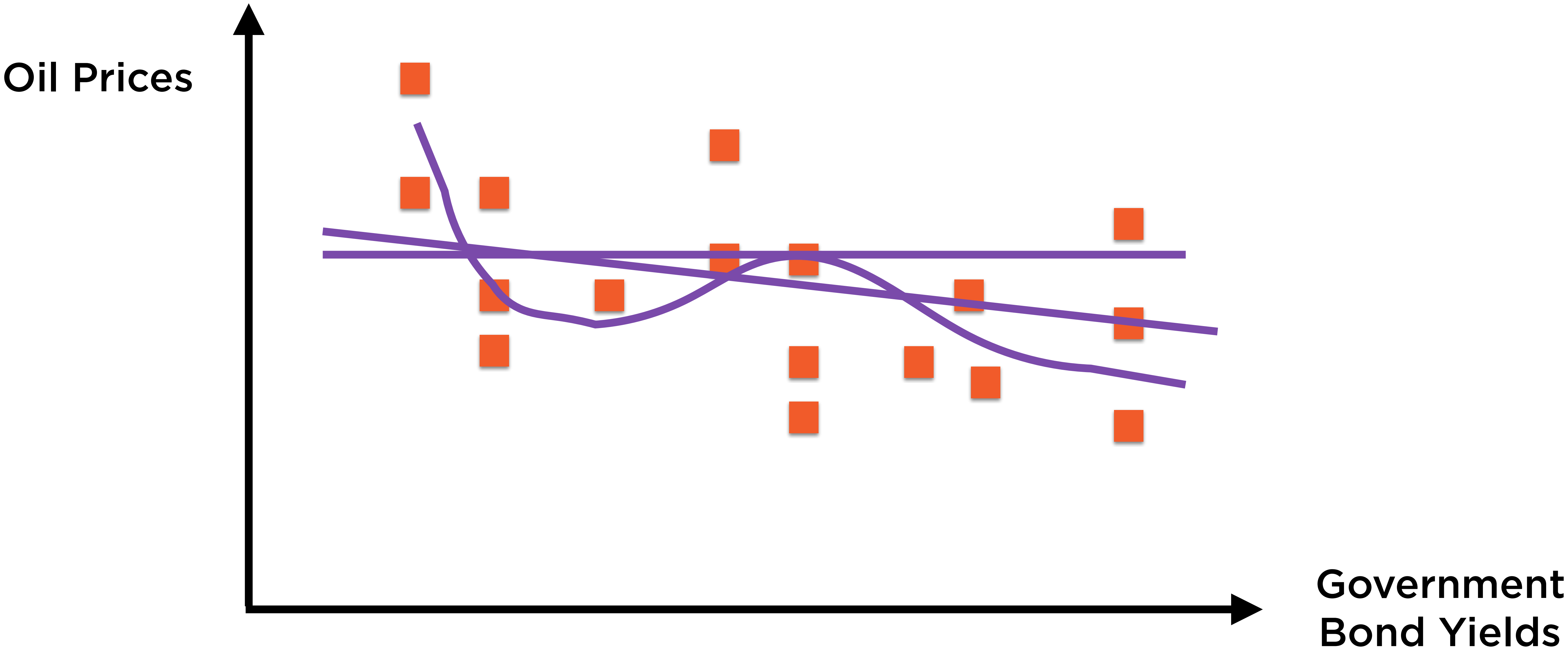
Bidimensional data can be represented in a plane

Data in Two Dimensions



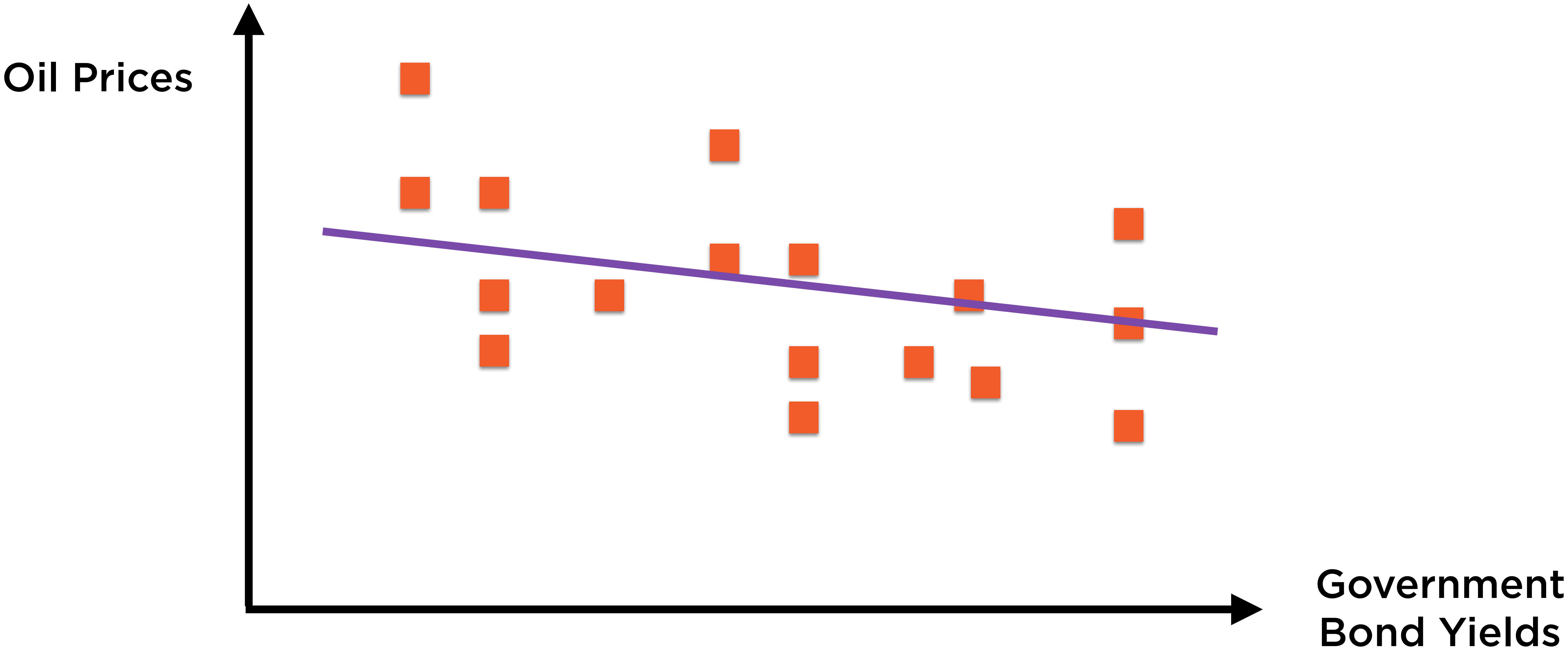
We can draw any number of curves to fit such data

Data in Two Dimensions



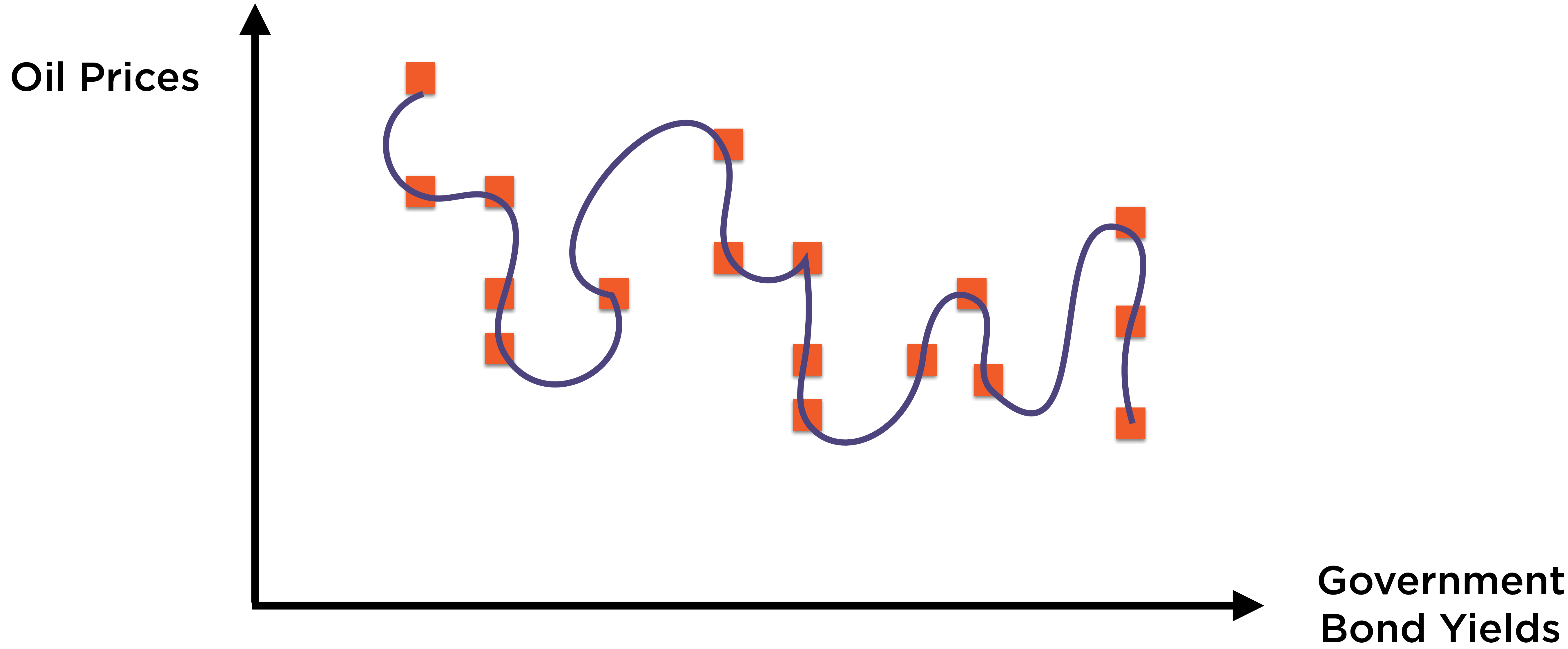
We can draw any number of curves to fit such data

Data in Two Dimensions



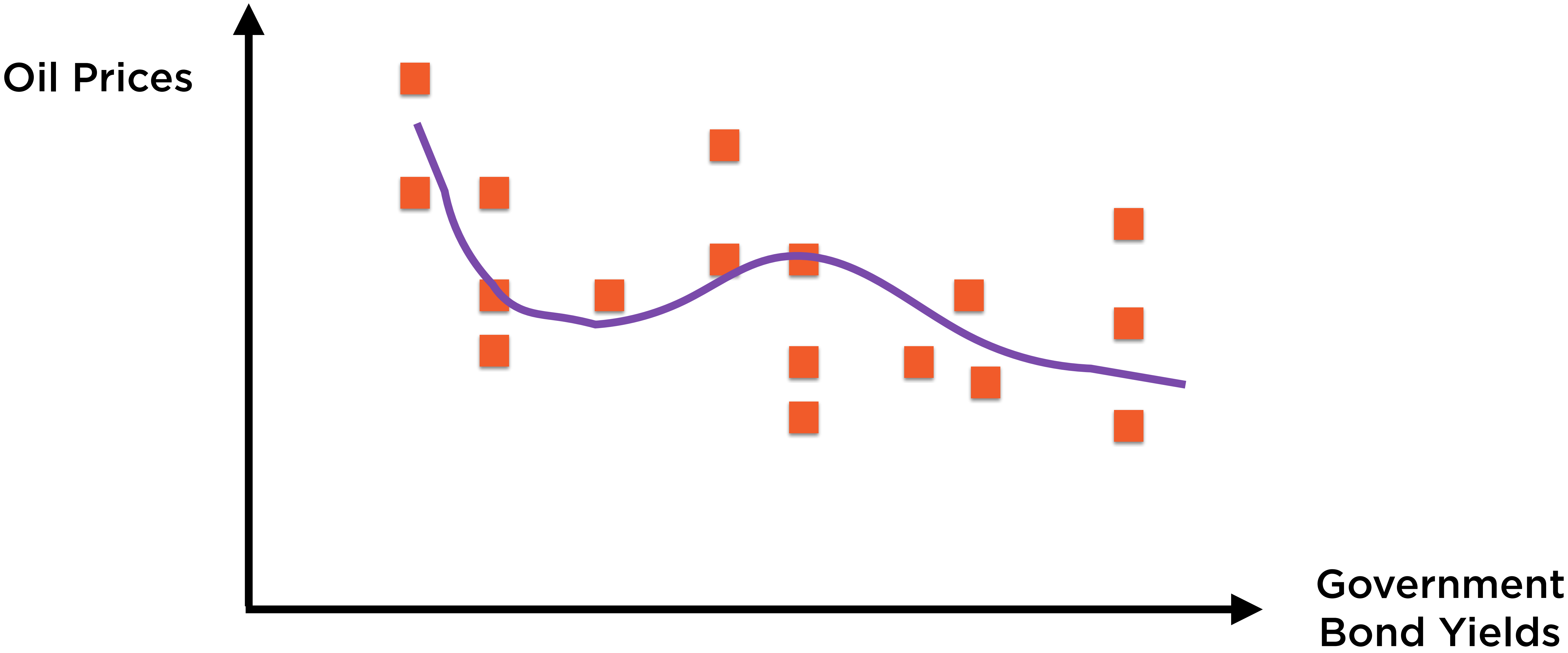
A straight line represents a linear relationship

Data in Two Dimensions



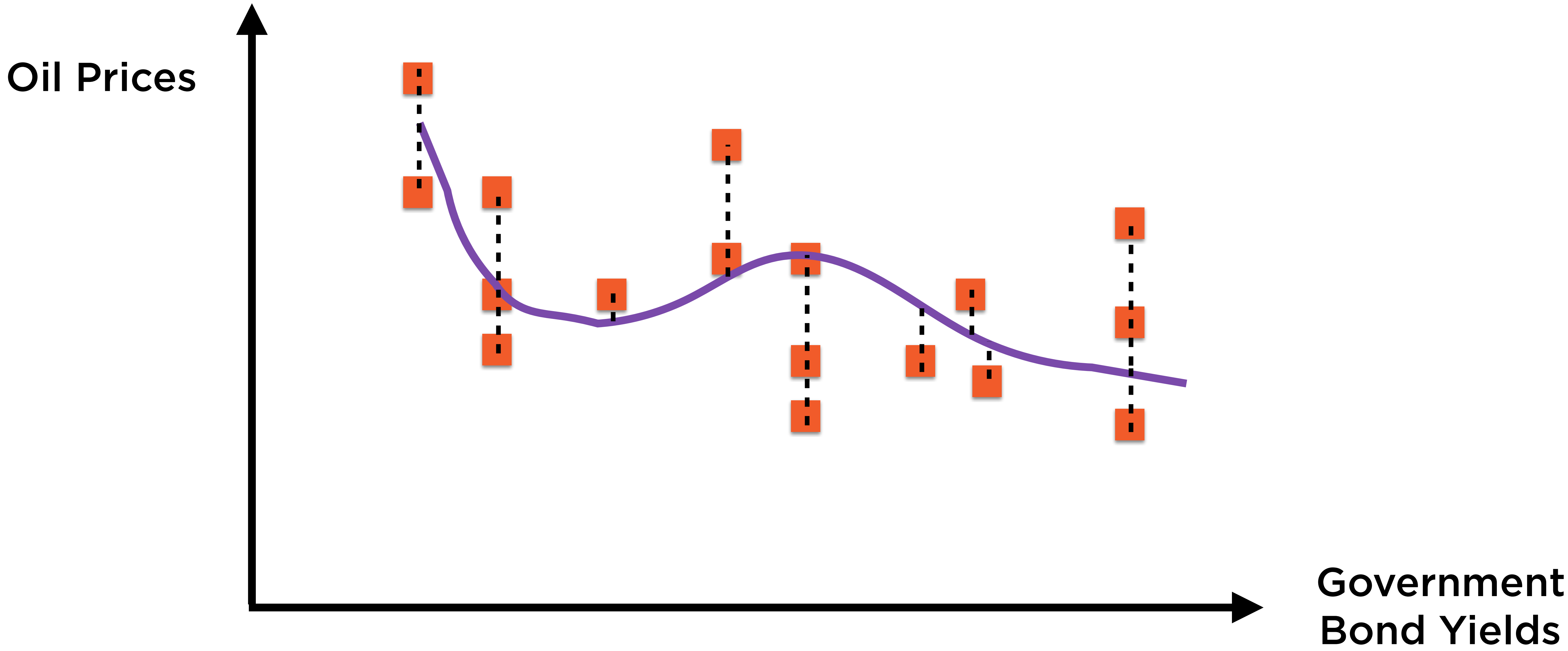
We could either make this curve pass through each point...

Data in Two Dimensions



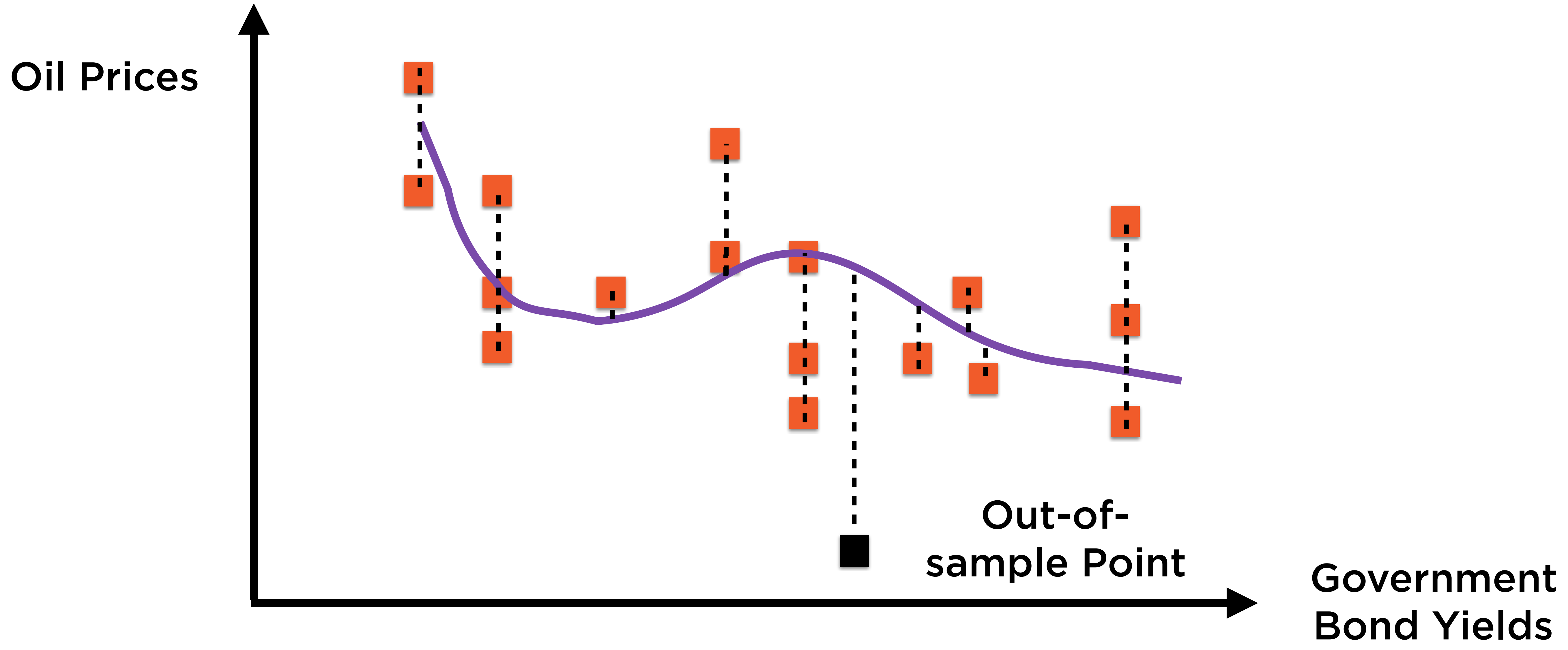
...Or in some sense "fit" the data in aggregate

Data in Two Dimensions



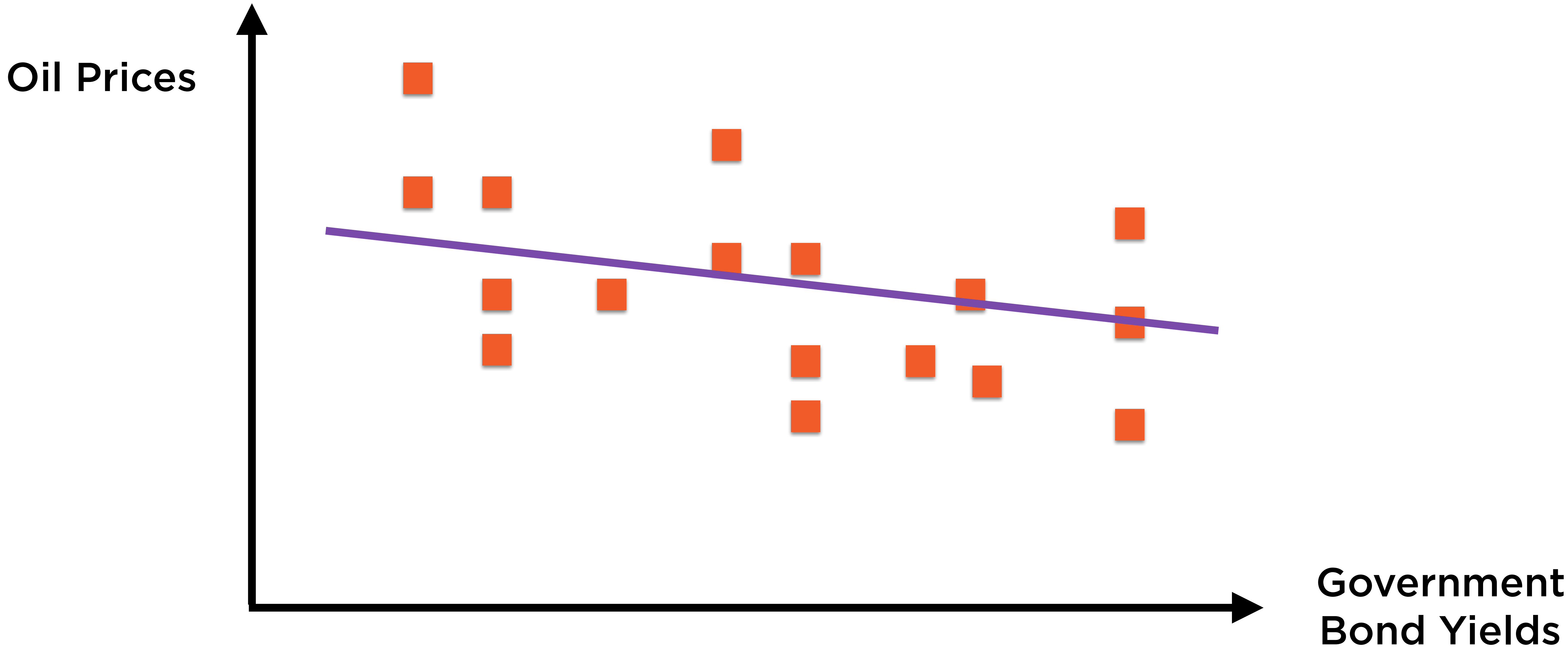
A curve has a “good fit” if the distances of points from the curve are small

Data in Two Dimensions



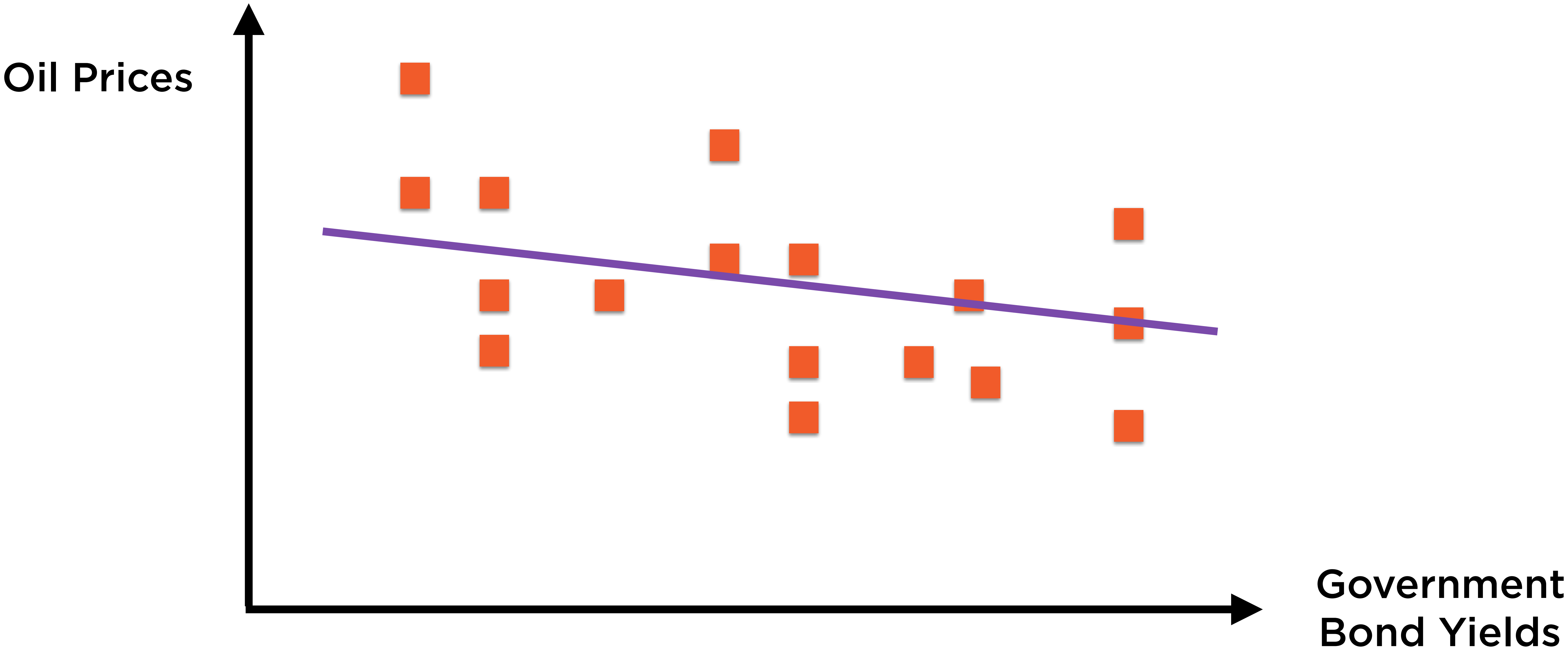
Overfitting by finding a very complicated curve often only hurts predictive accuracy

Data in Two Dimensions



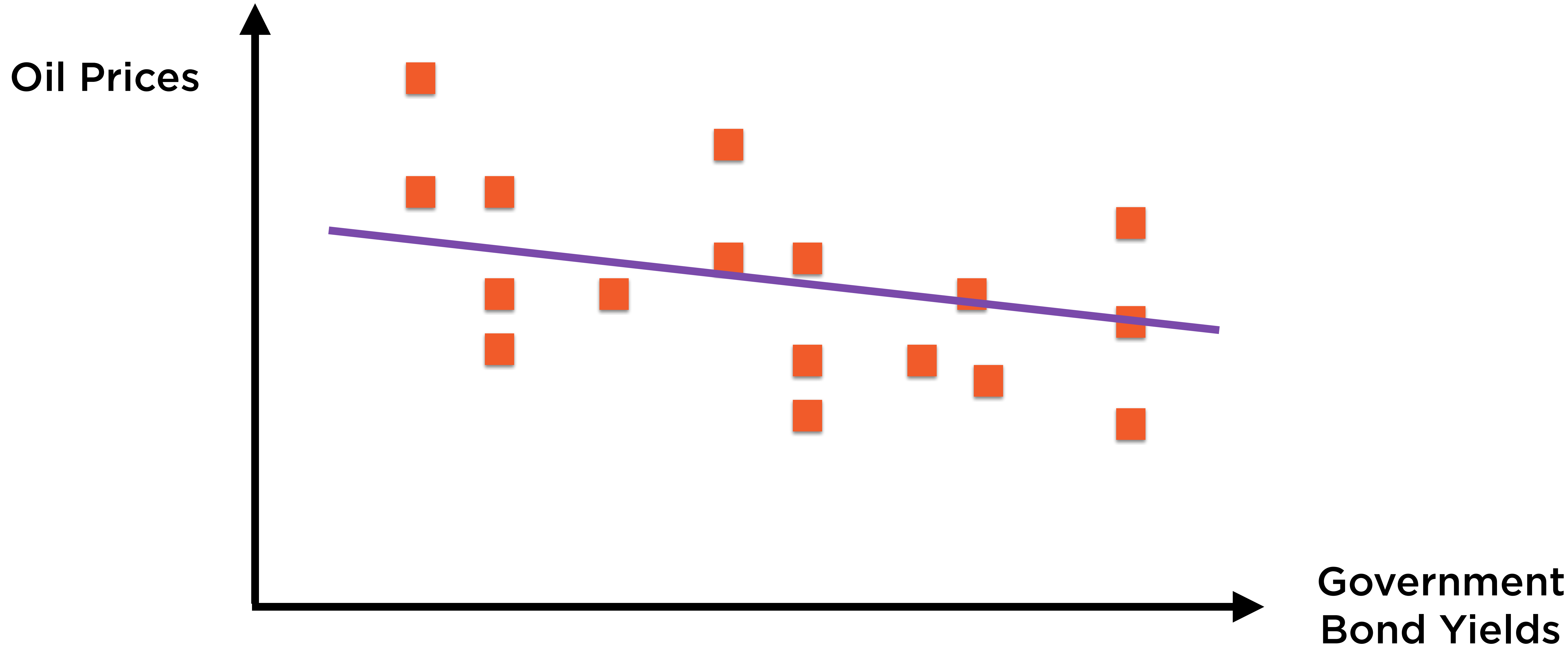
Often, a straight line works just fine

Data in Two Dimensions



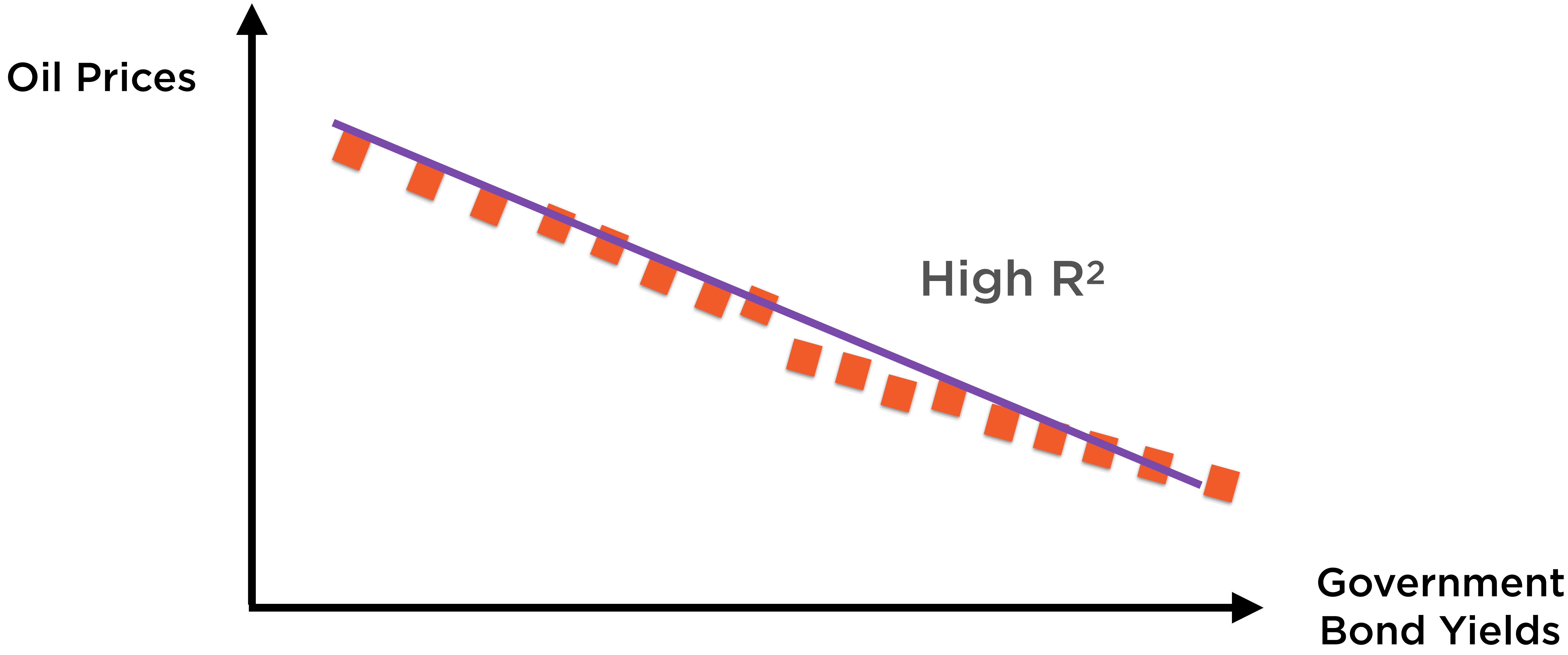
Finding the “best” such straight line is called **Linear Regression**

Data in Two Dimensions



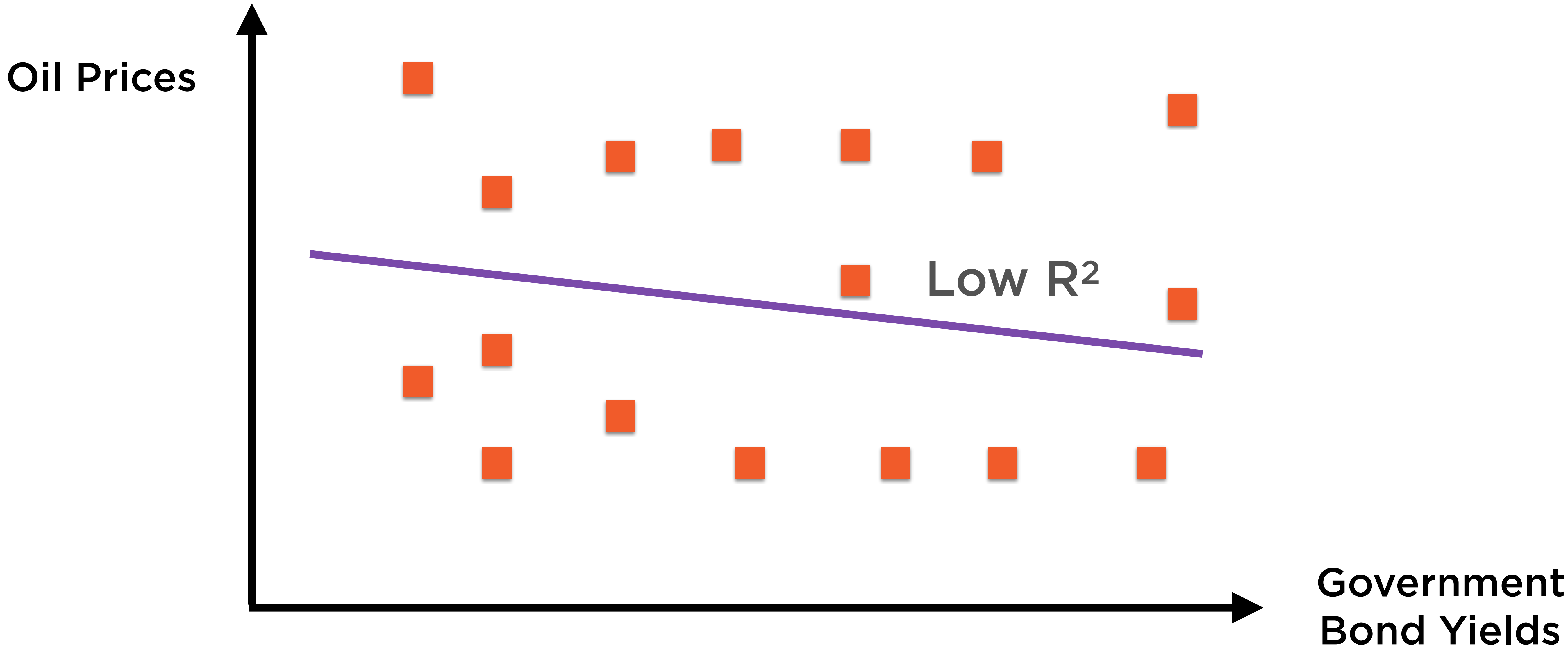
Regression not only gives us the equation of this line, it also signals how reliable the line is

Data in Two Dimensions



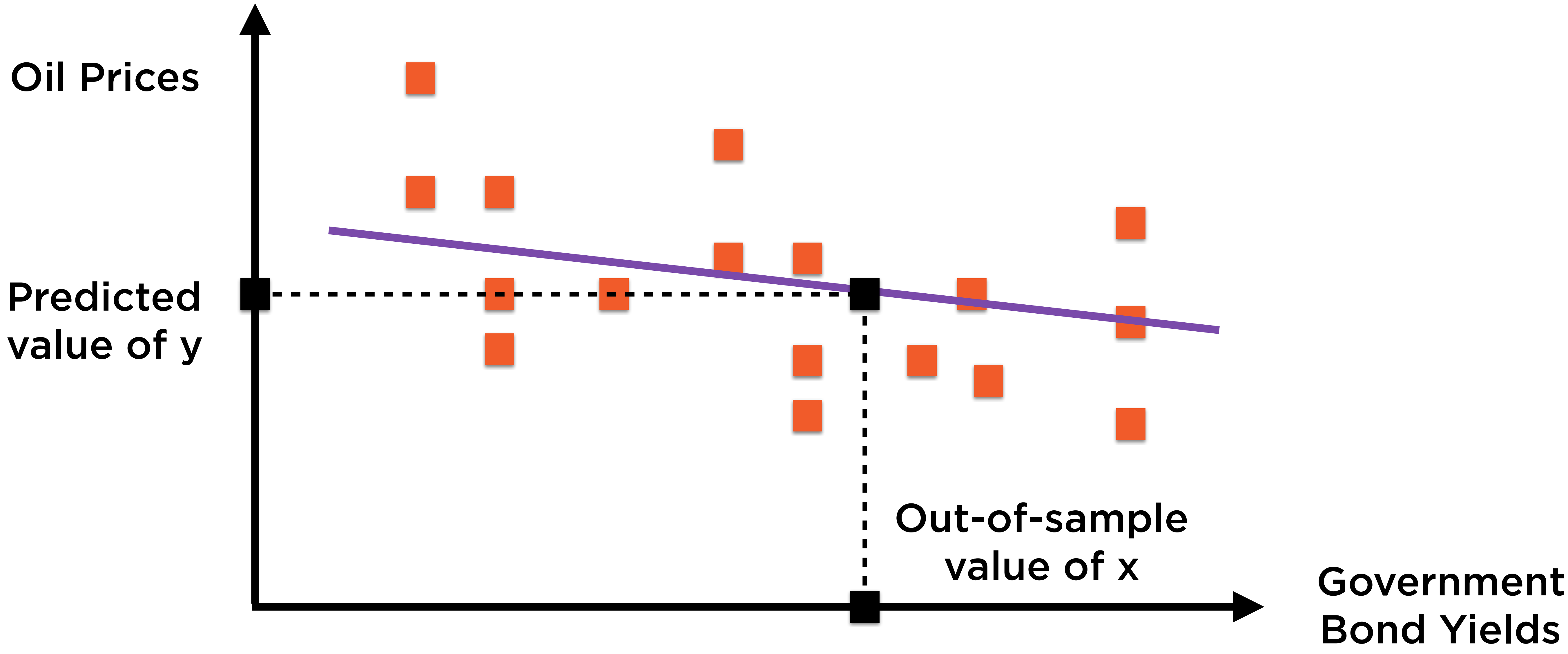
High quality of fit

Data in Two Dimensions



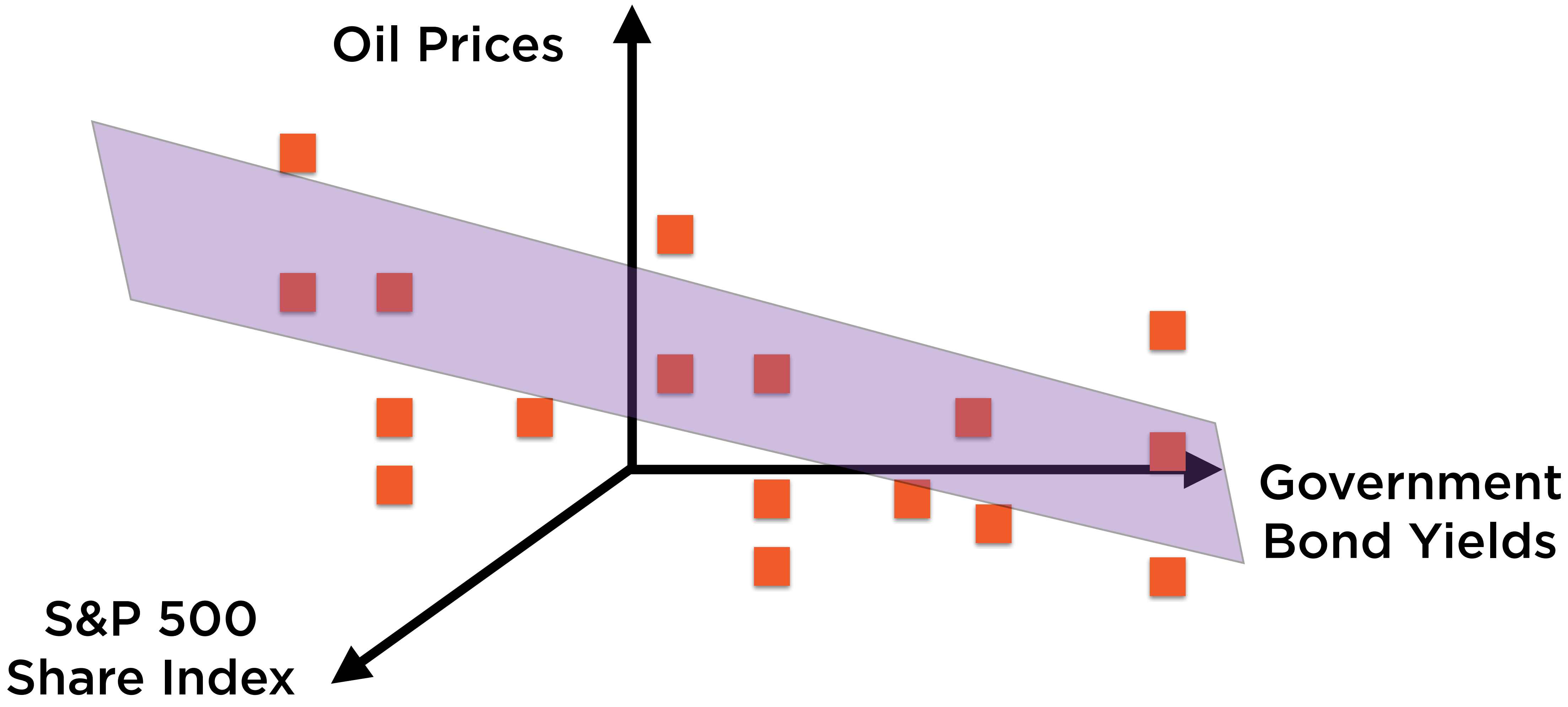
Low quality of fit

Prediction Using Regression



Given a new value of x, use the line to predict the corresponding value of y

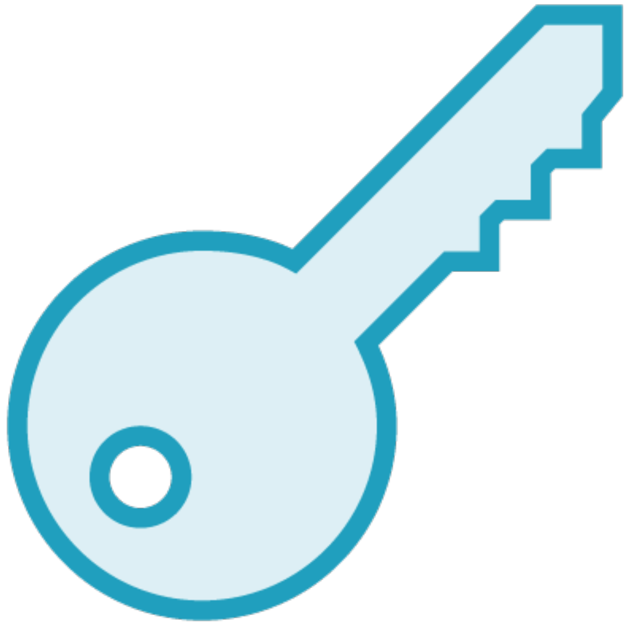
Data in N Dimensions



Linear Regression can easily be extended to n-dimensional data

Setting Up the Regression Problem

X Causes Y



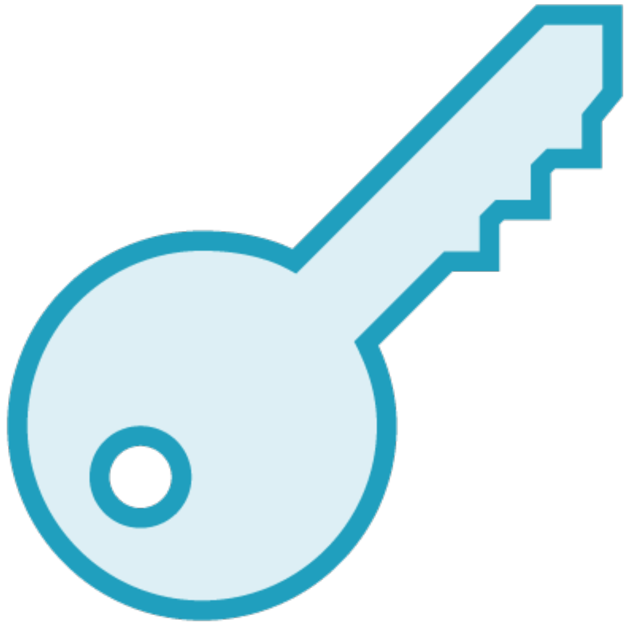
Cause

Independent variable

Effect

Dependent variable

X Causes Y



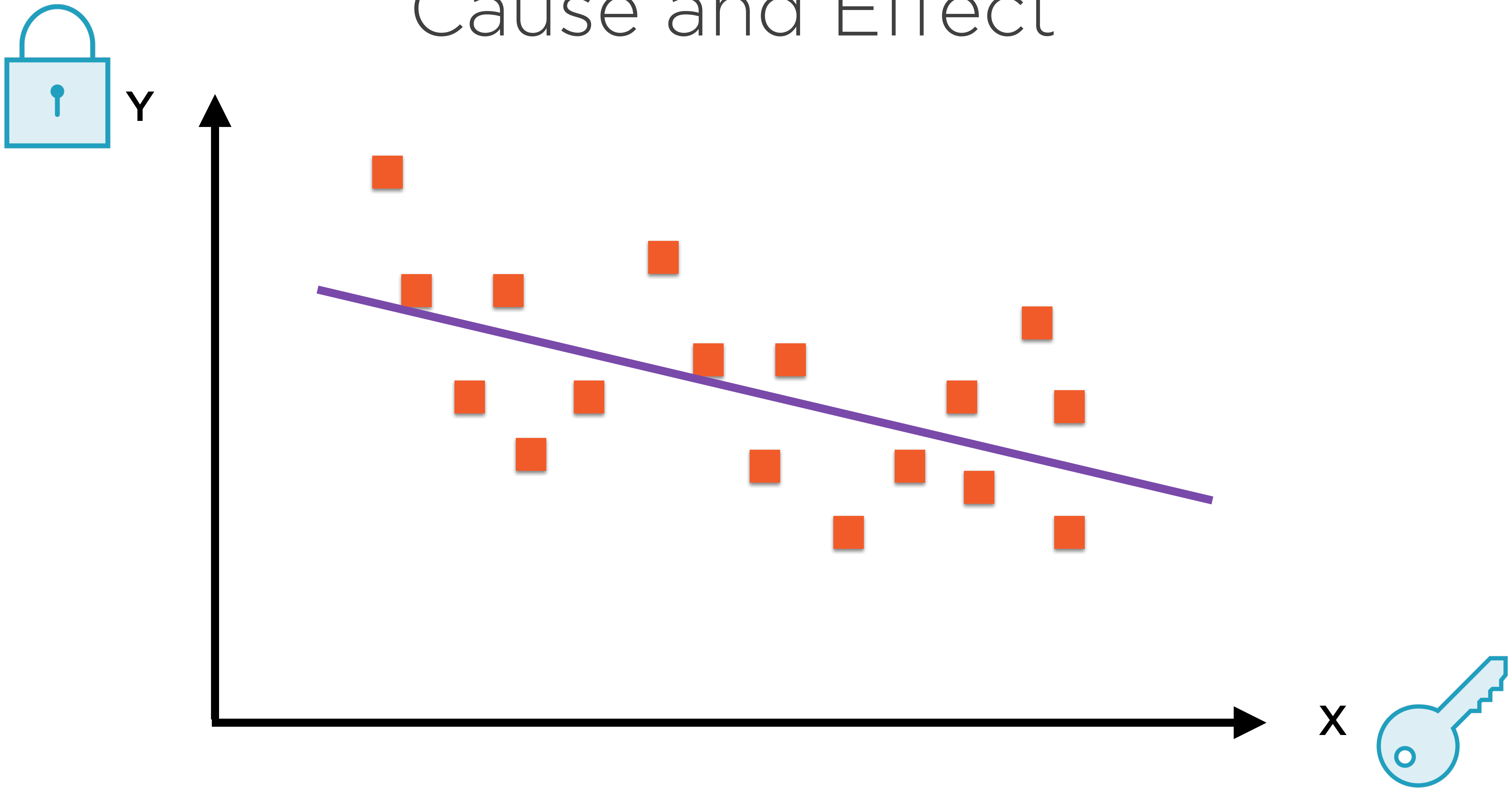
Cause

Explanatory variable

Effect

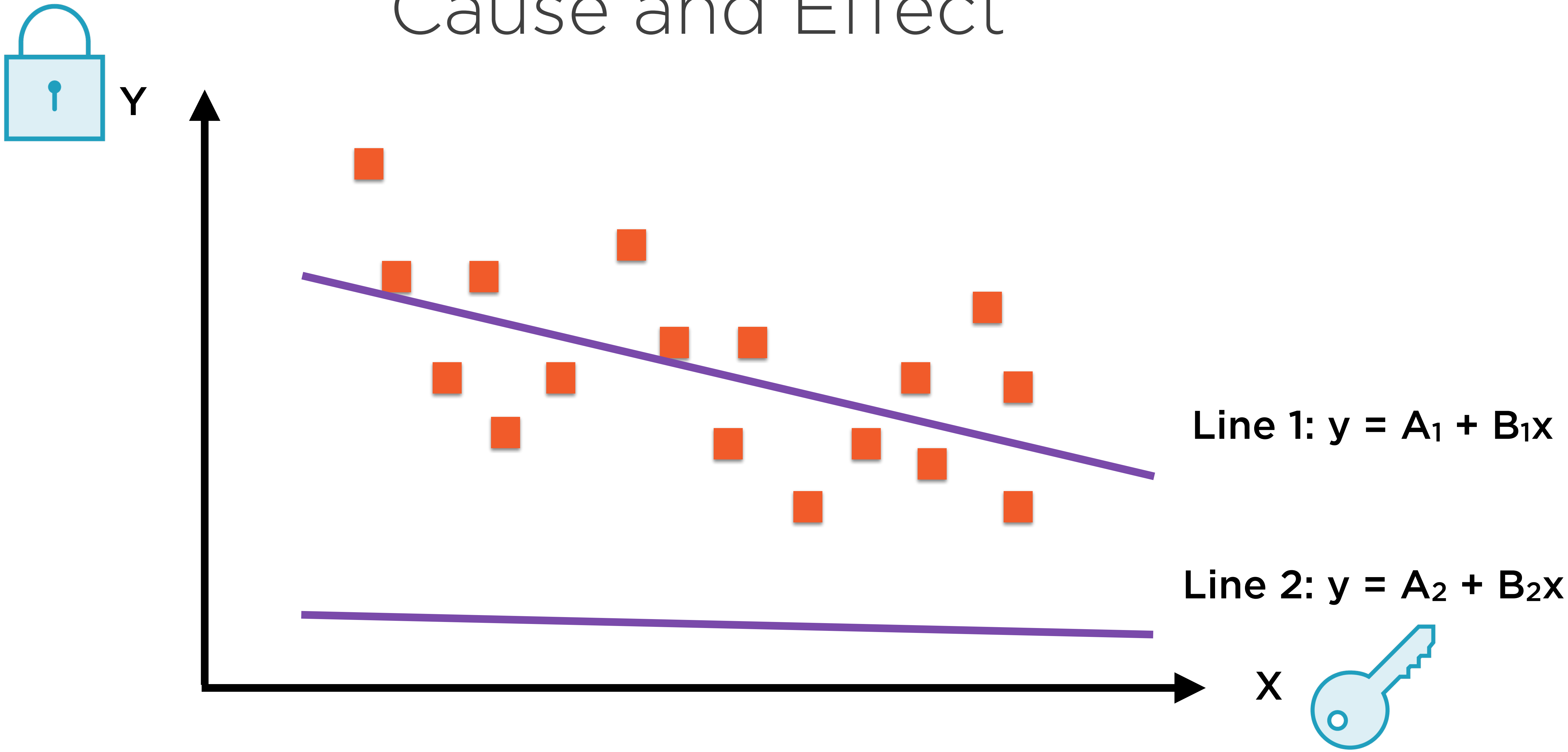
Dependent variable

Cause and Effect



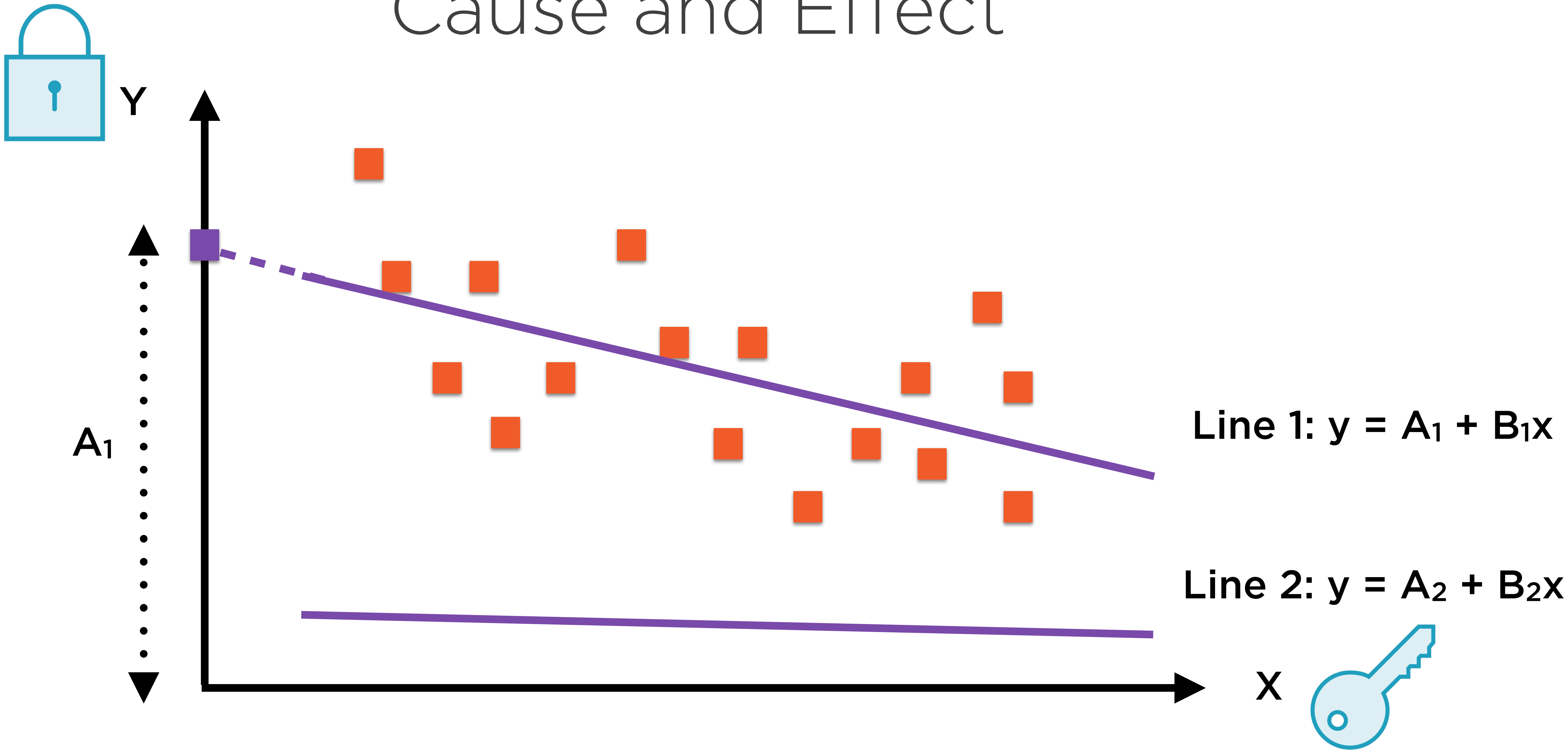
Linear Regression involves finding the “best fit” line

Cause and Effect



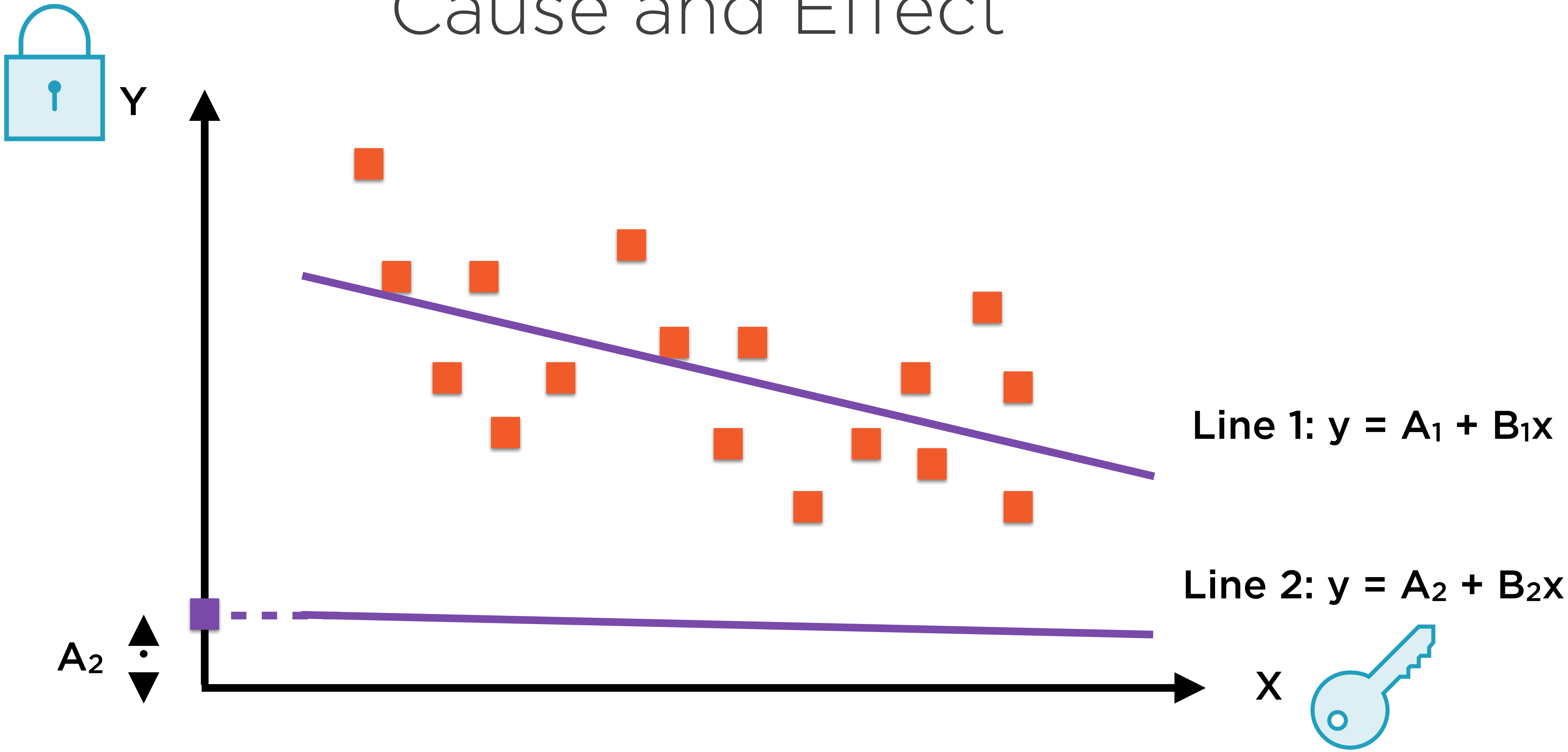
Let's compare two lines, Line 1 and Line 2

Cause and Effect



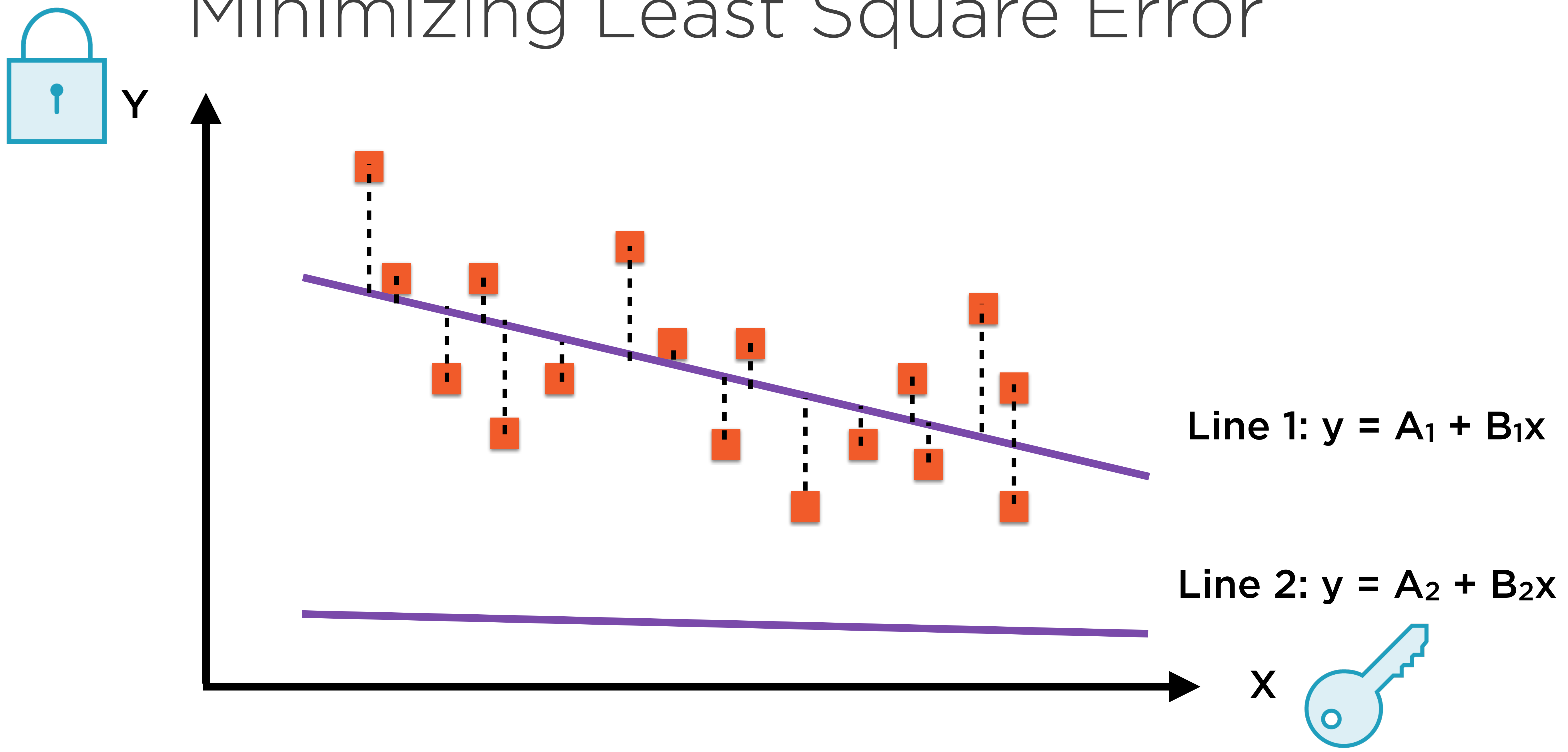
The first line has y-intercept A_1

Cause and Effect

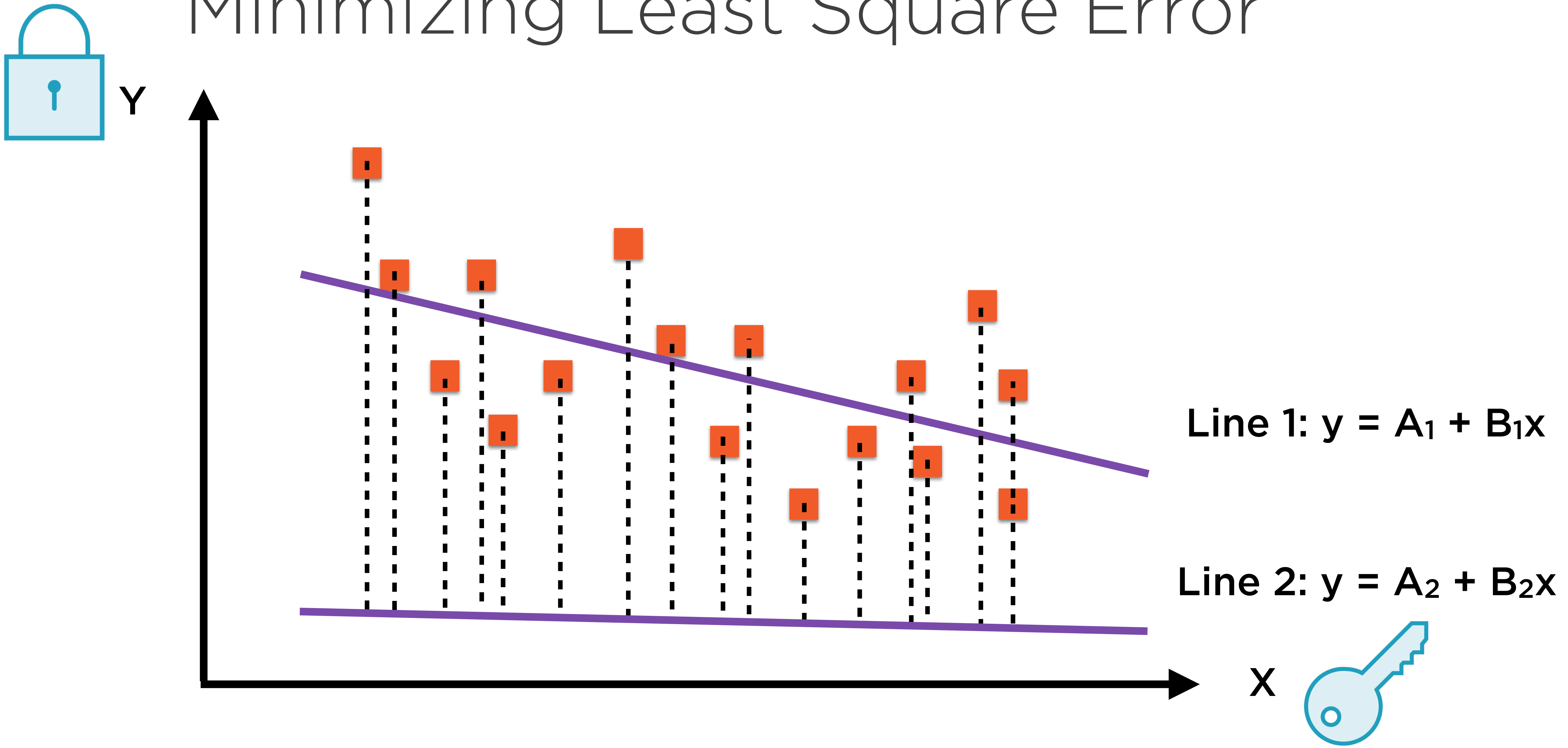


The second line has y-intercept A_2

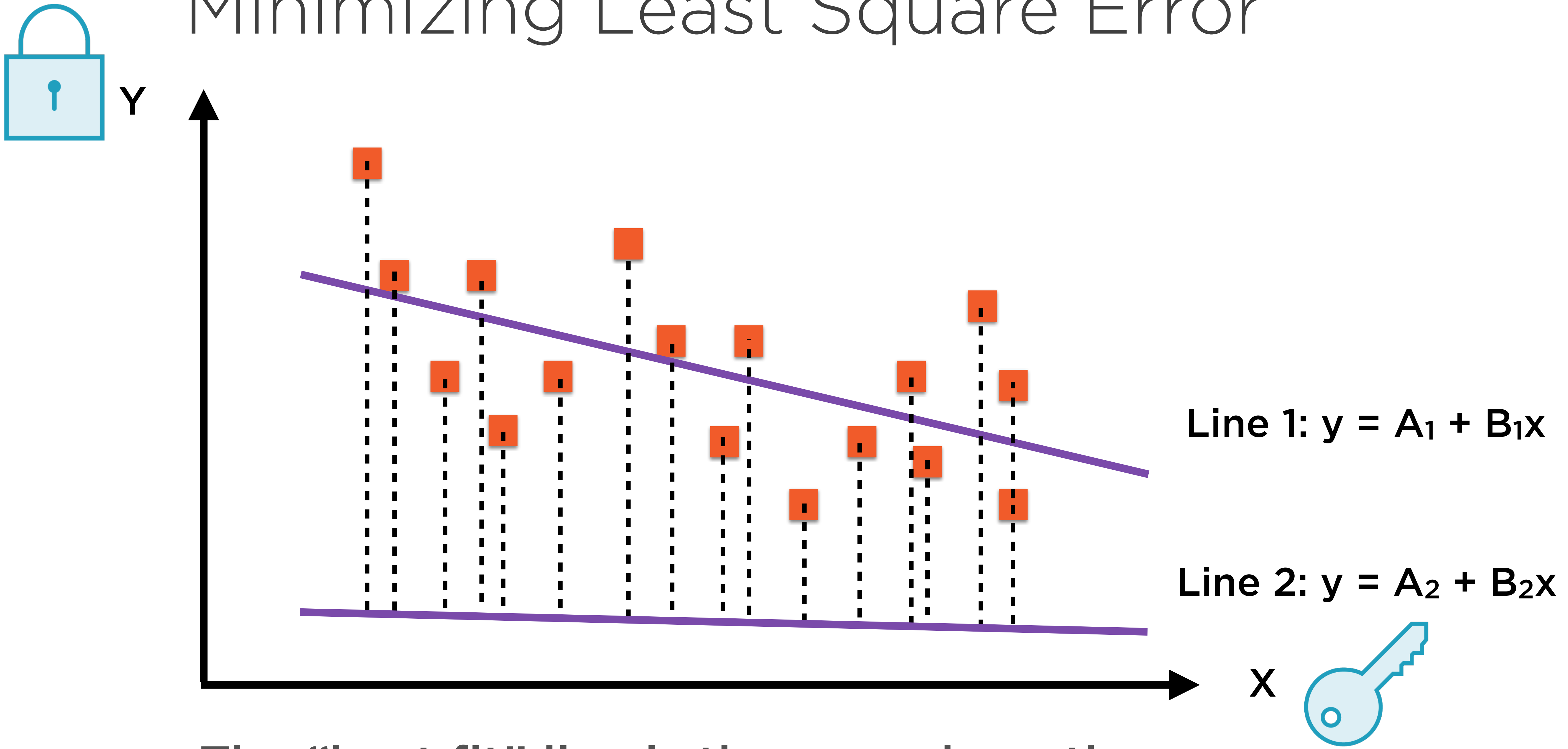
Minimizing Least Square Error



Minimizing Least Square Error

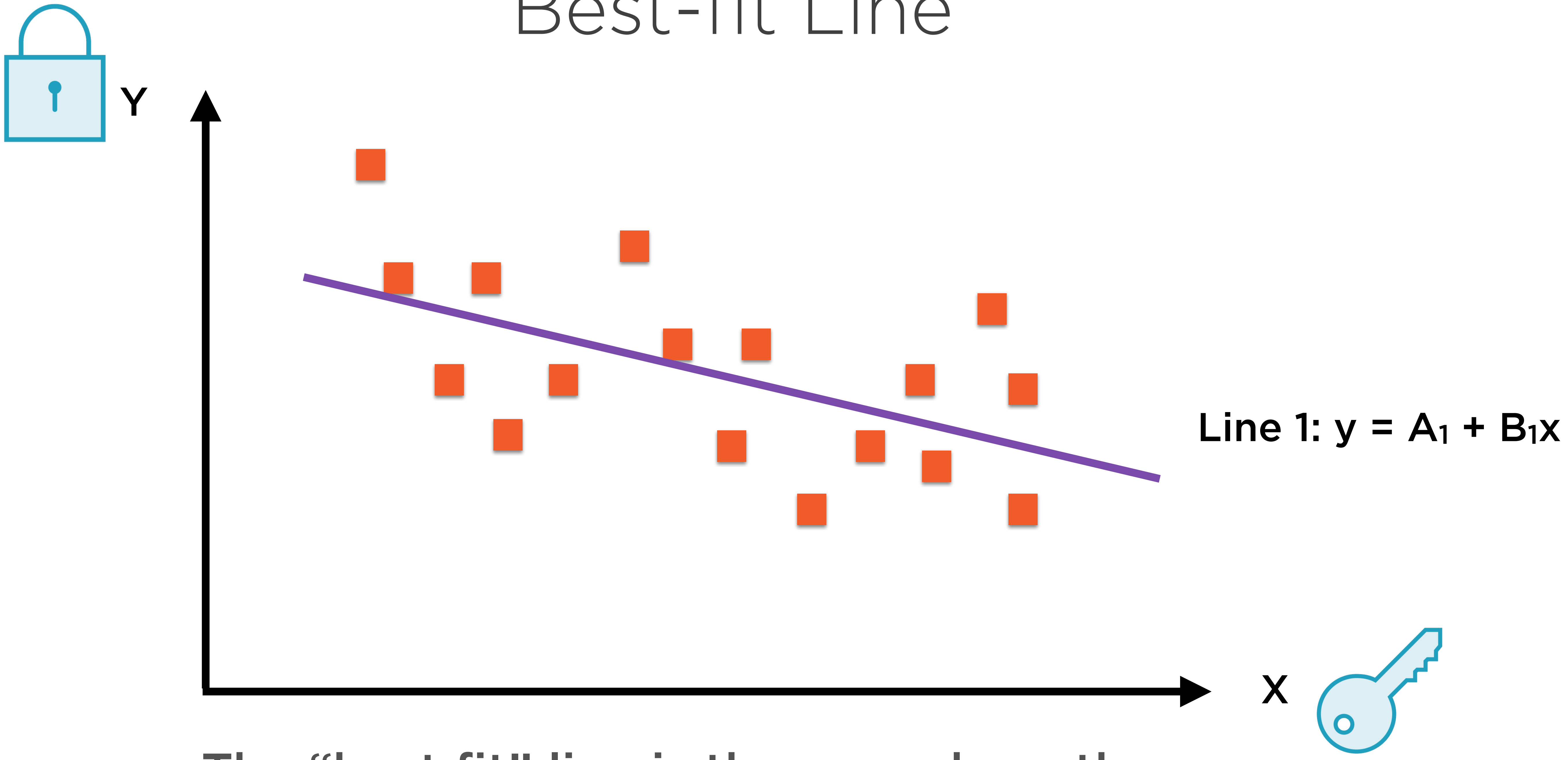


Minimizing Least Square Error



The “best fit” line is the one where the sum of the squares of the lengths of the errors is minimum

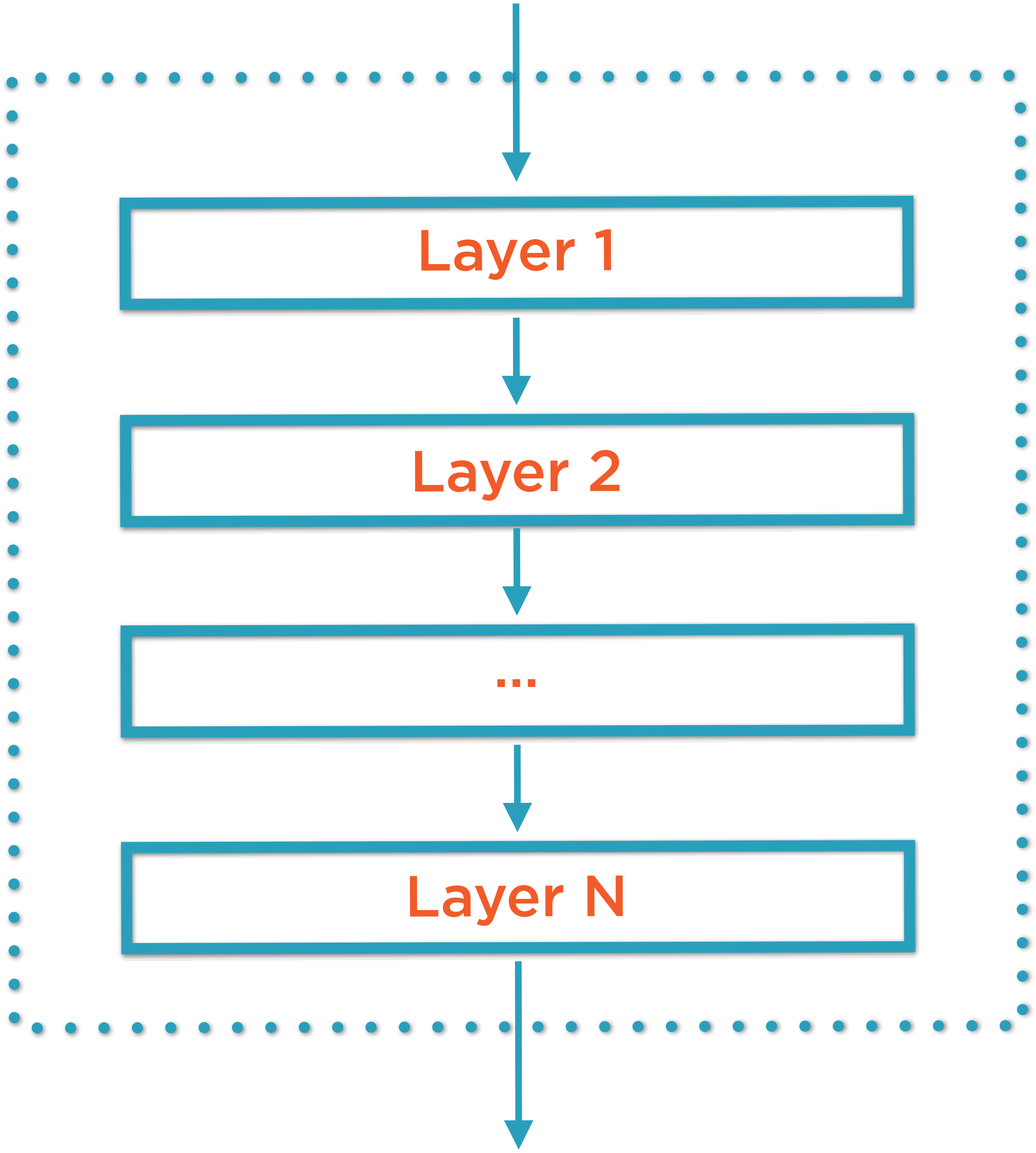
Best-fit Line



The “best fit” line is the one where the sum of the squares of the lengths of these dotted lines is minimum

Gradient Descent

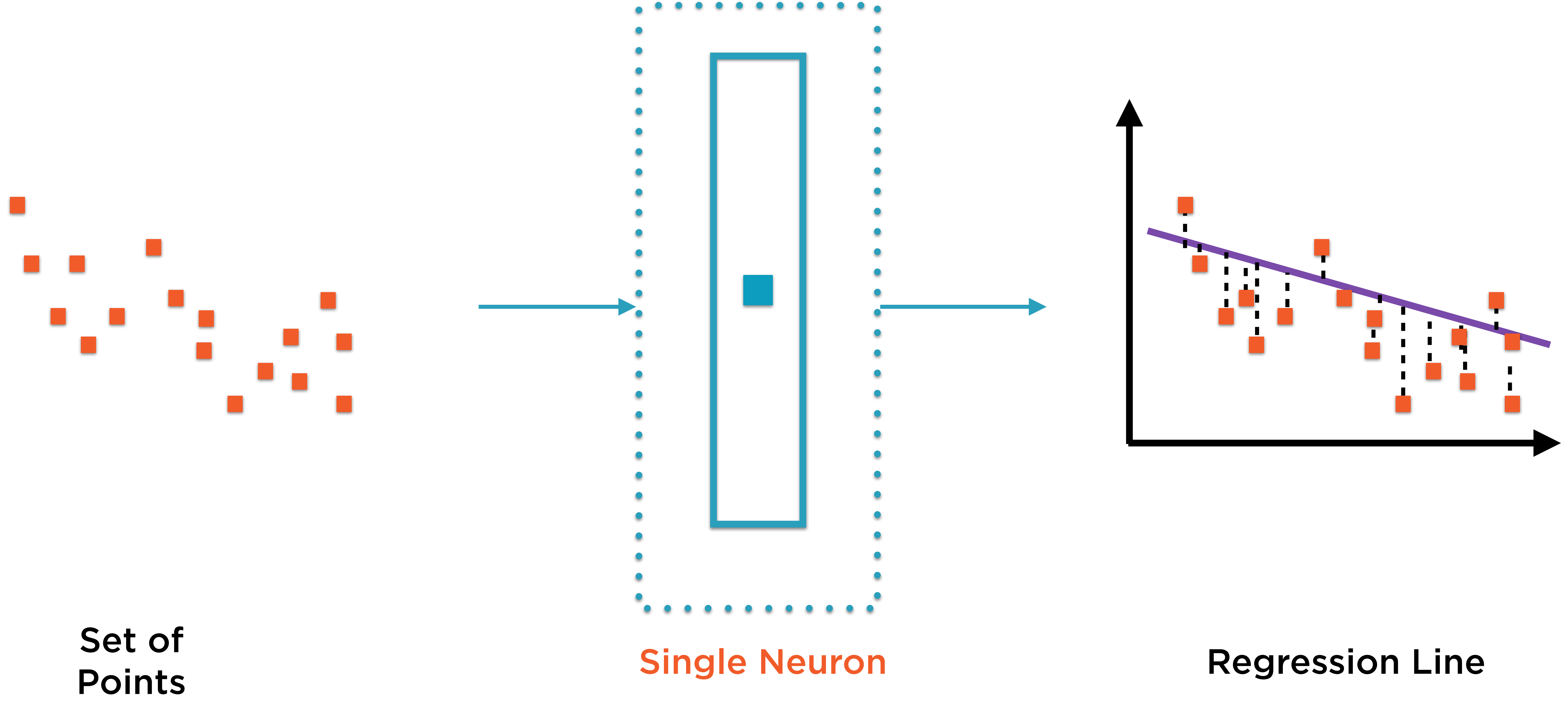
Neural Network Model



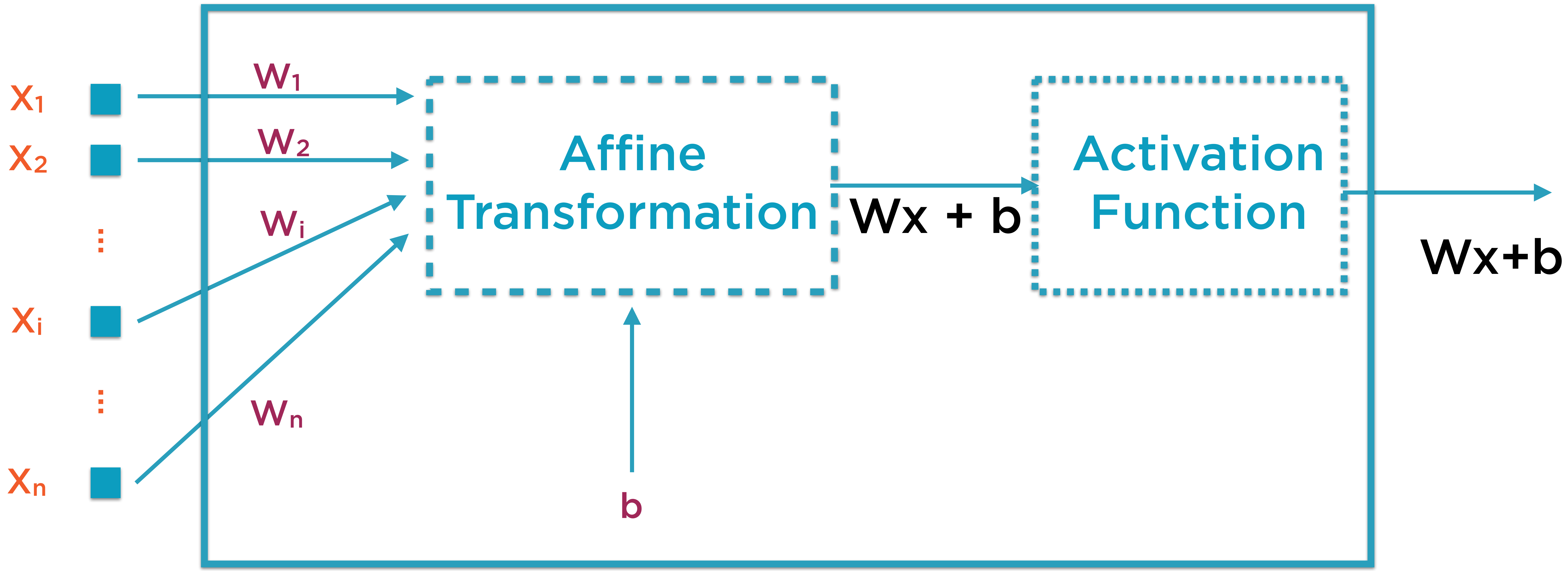
Network of interconnected layers

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

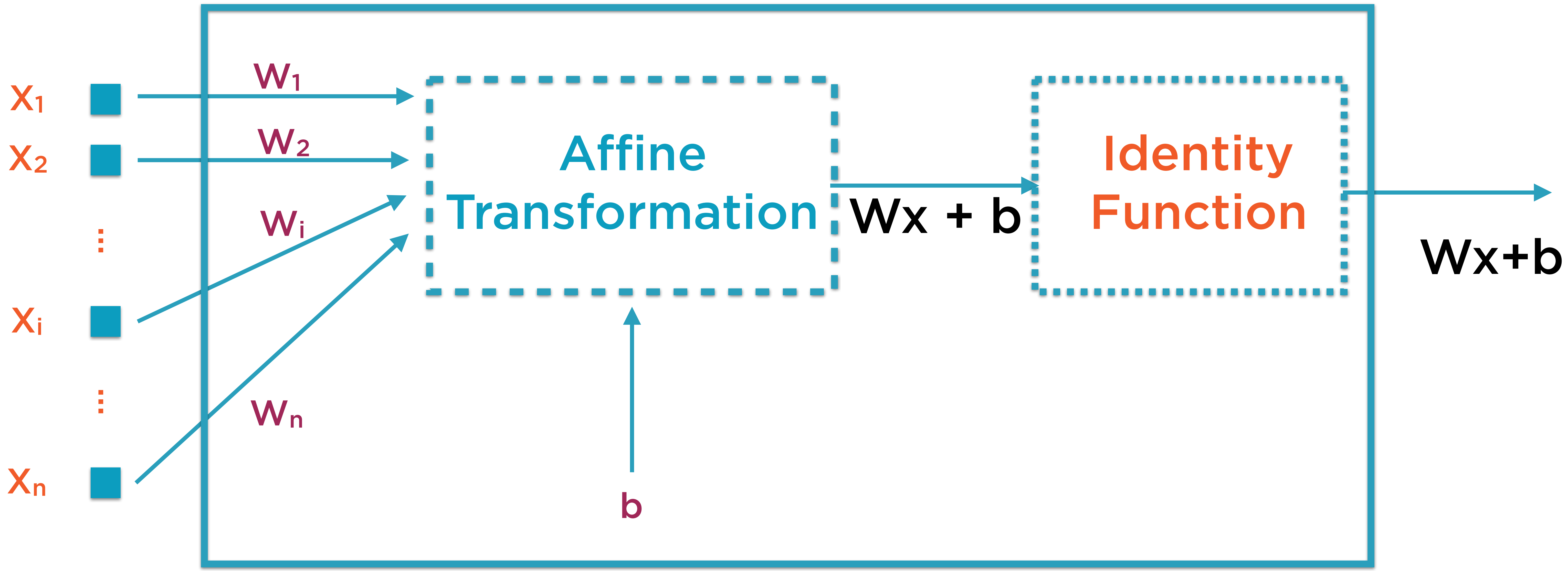
Regression: The Simplest Neural Network



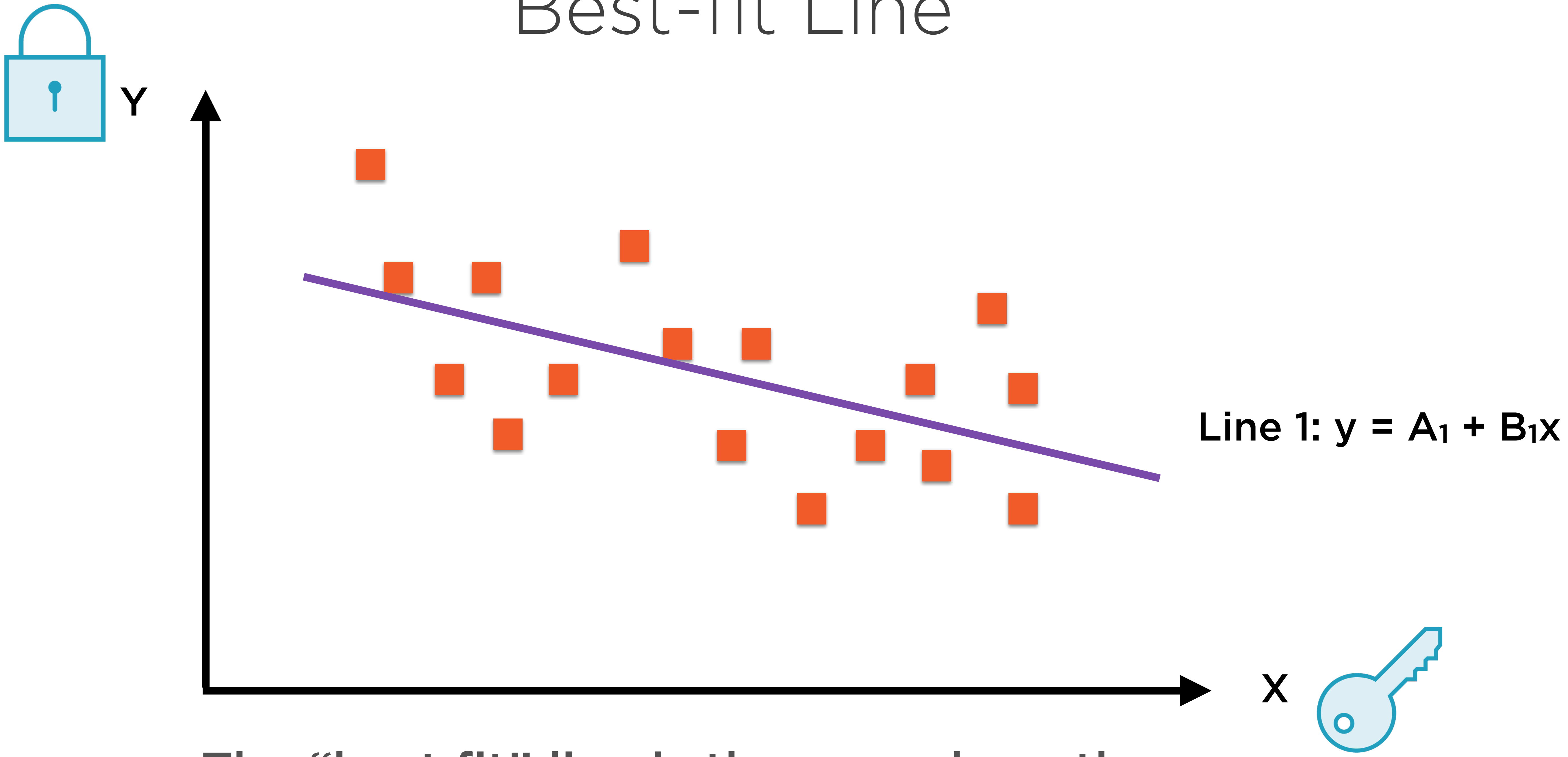
Regression: The Simplest Neural Network



Regression: The Simplest Neural Network



Best-fit Line



The “best fit” line is the one where the sum of the squares of the lengths of these dotted lines is minimum

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

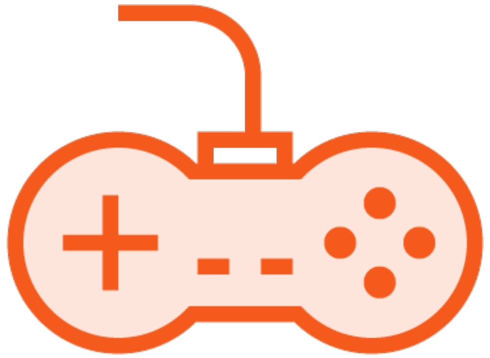
Objective Function

Minimize variance of the residuals (MSE)

Constraints

Express relationship as a straight line

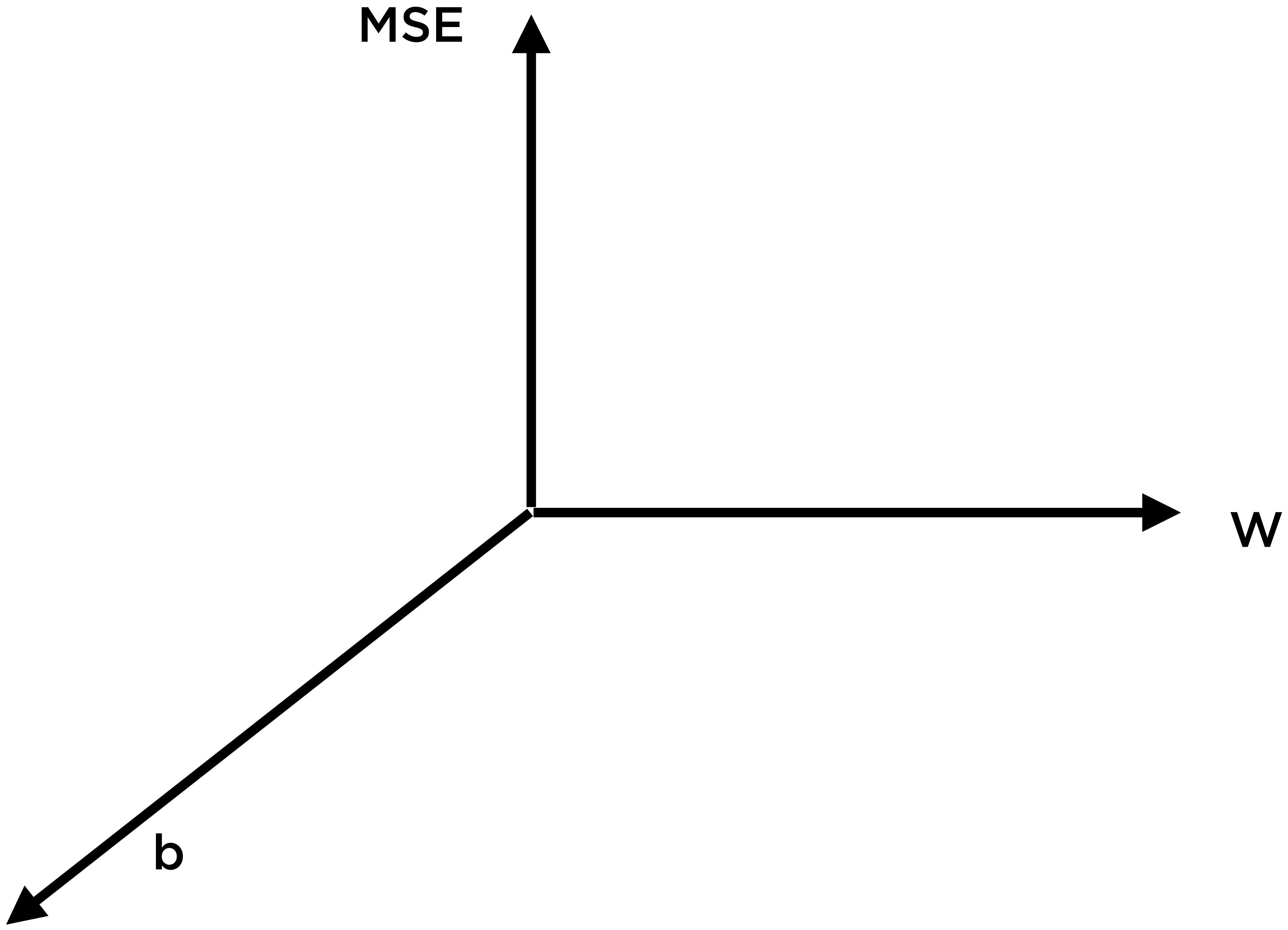
$y = Wx + b$



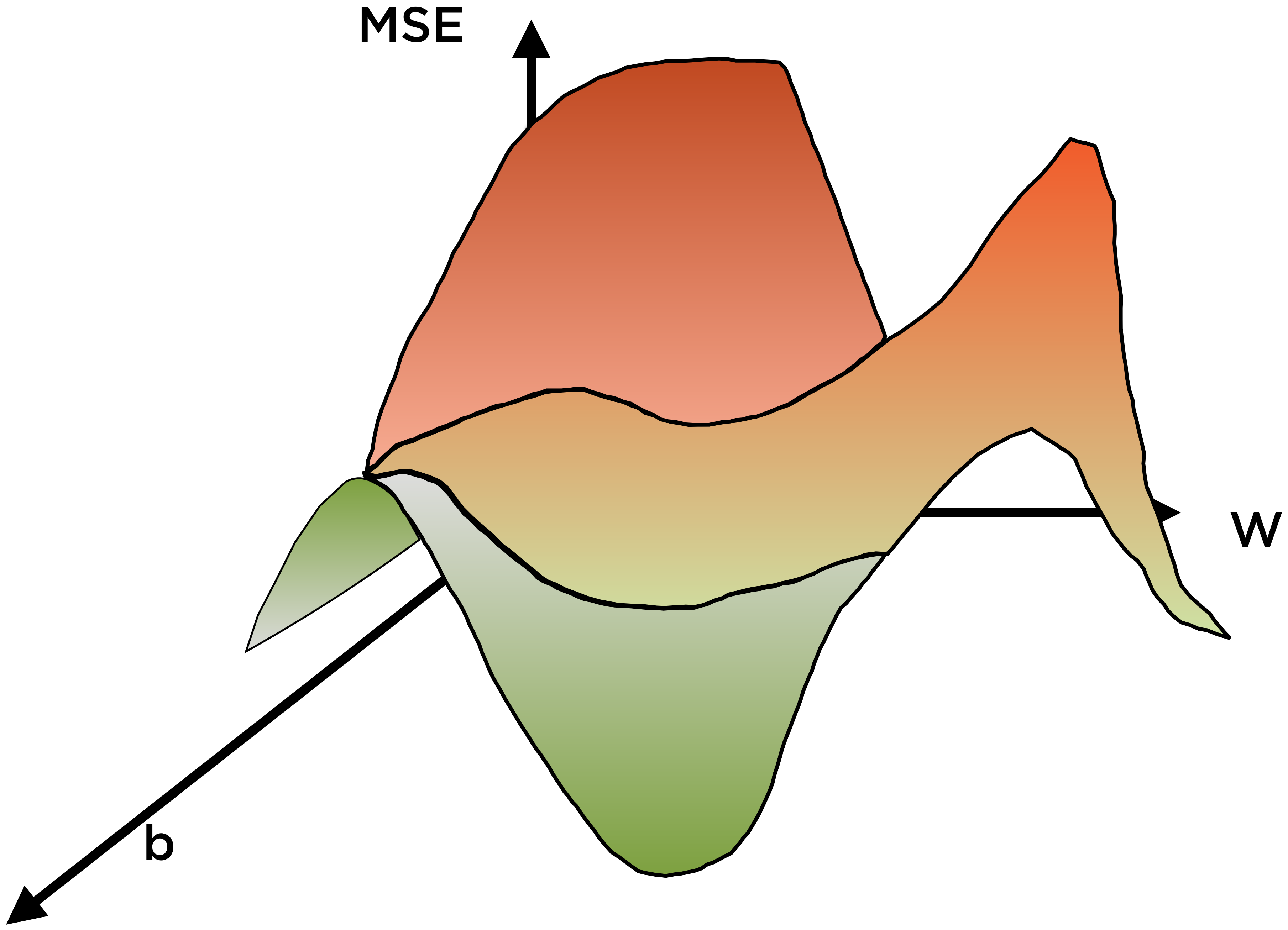
Decision Variables

Values of W and b

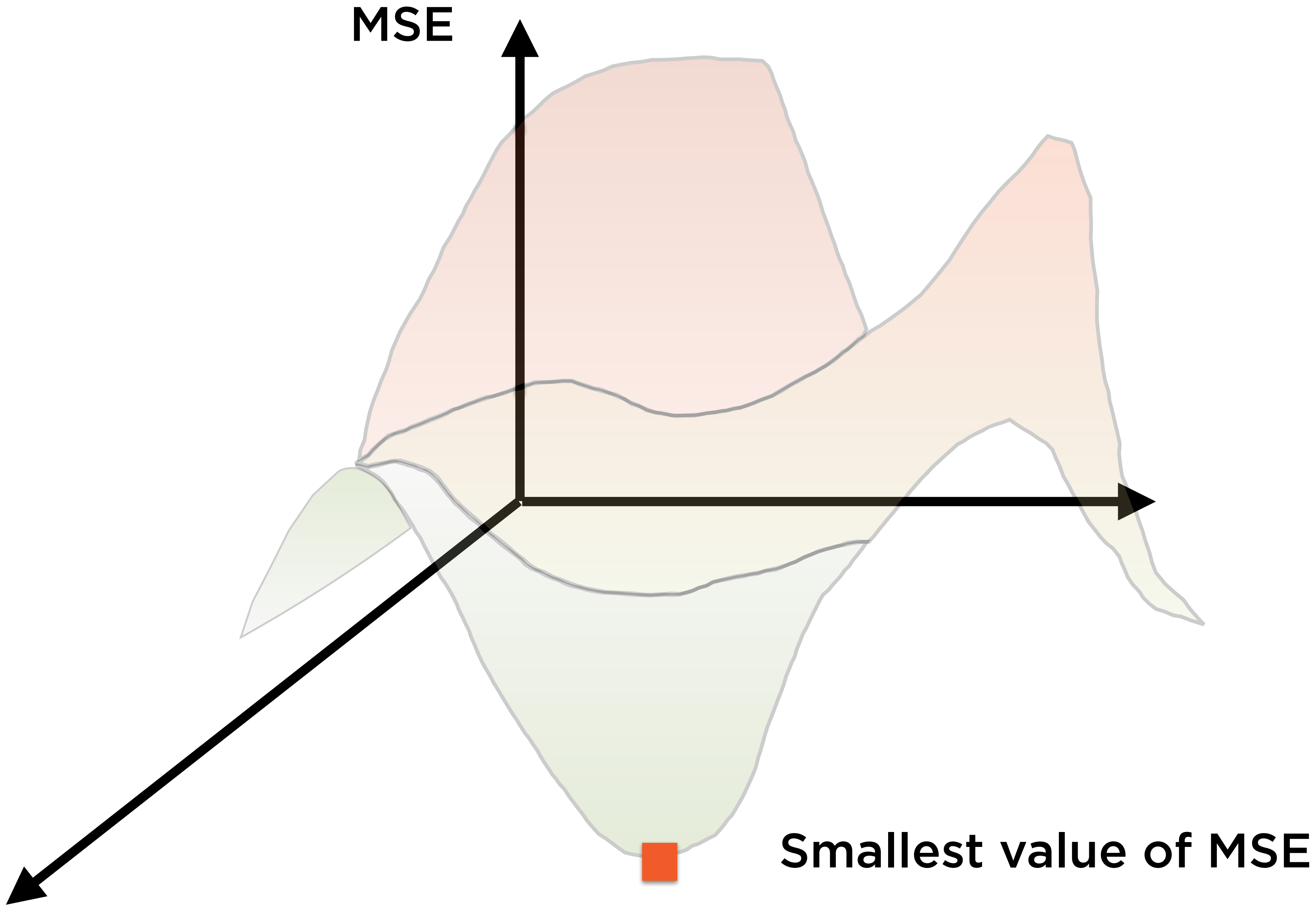
Minimizing MSE



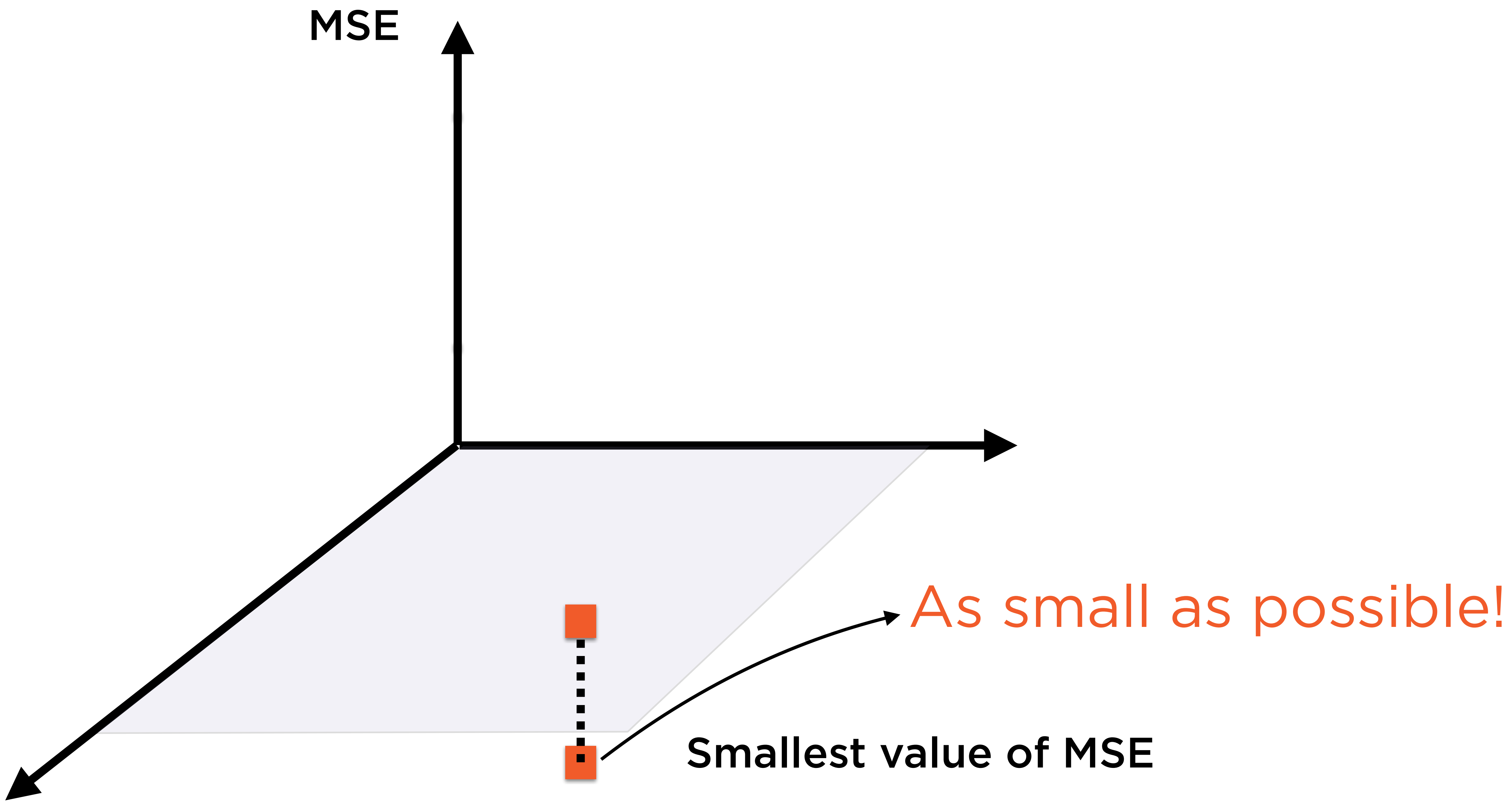
Minimizing MSE



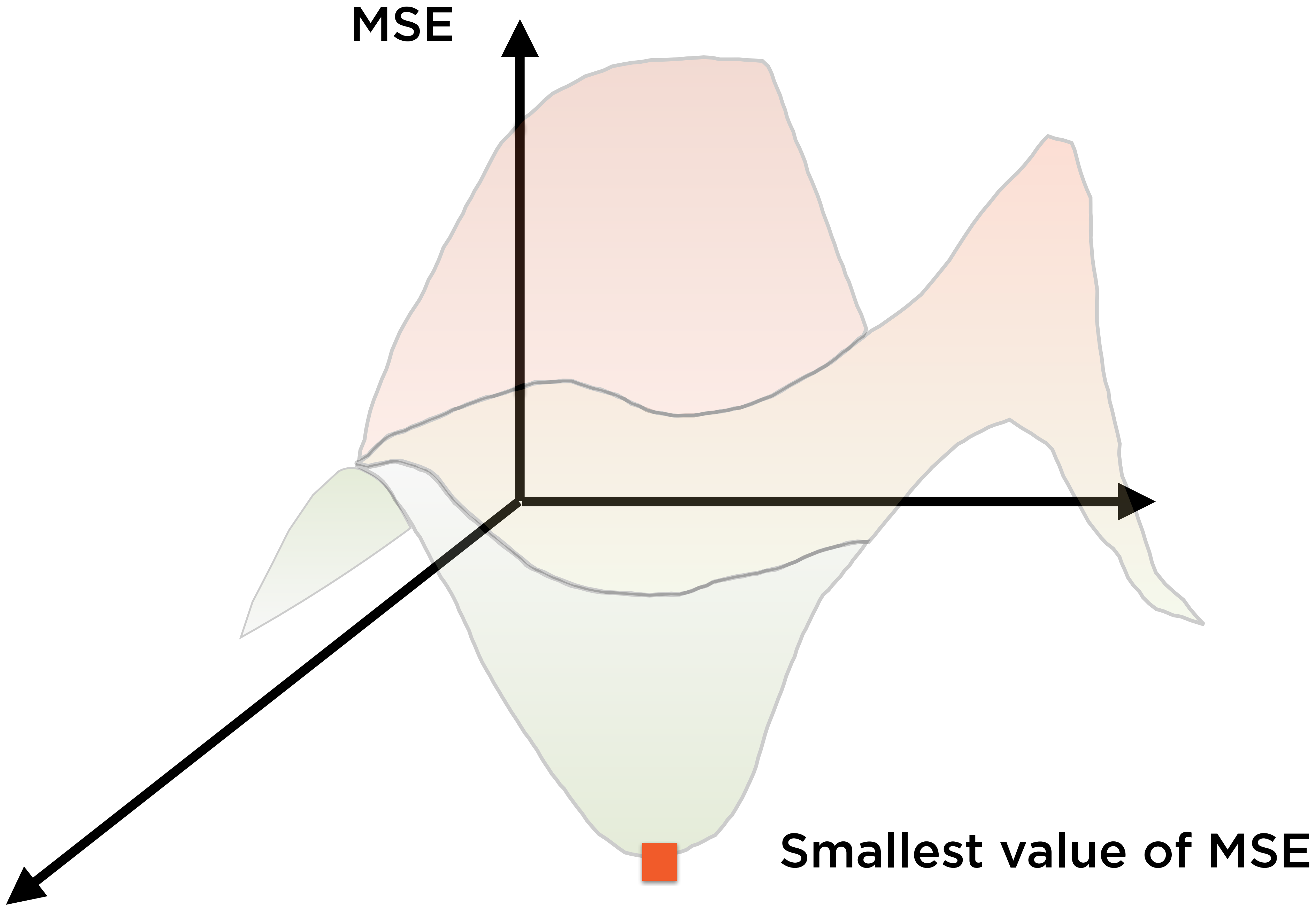
Minimizing MSE



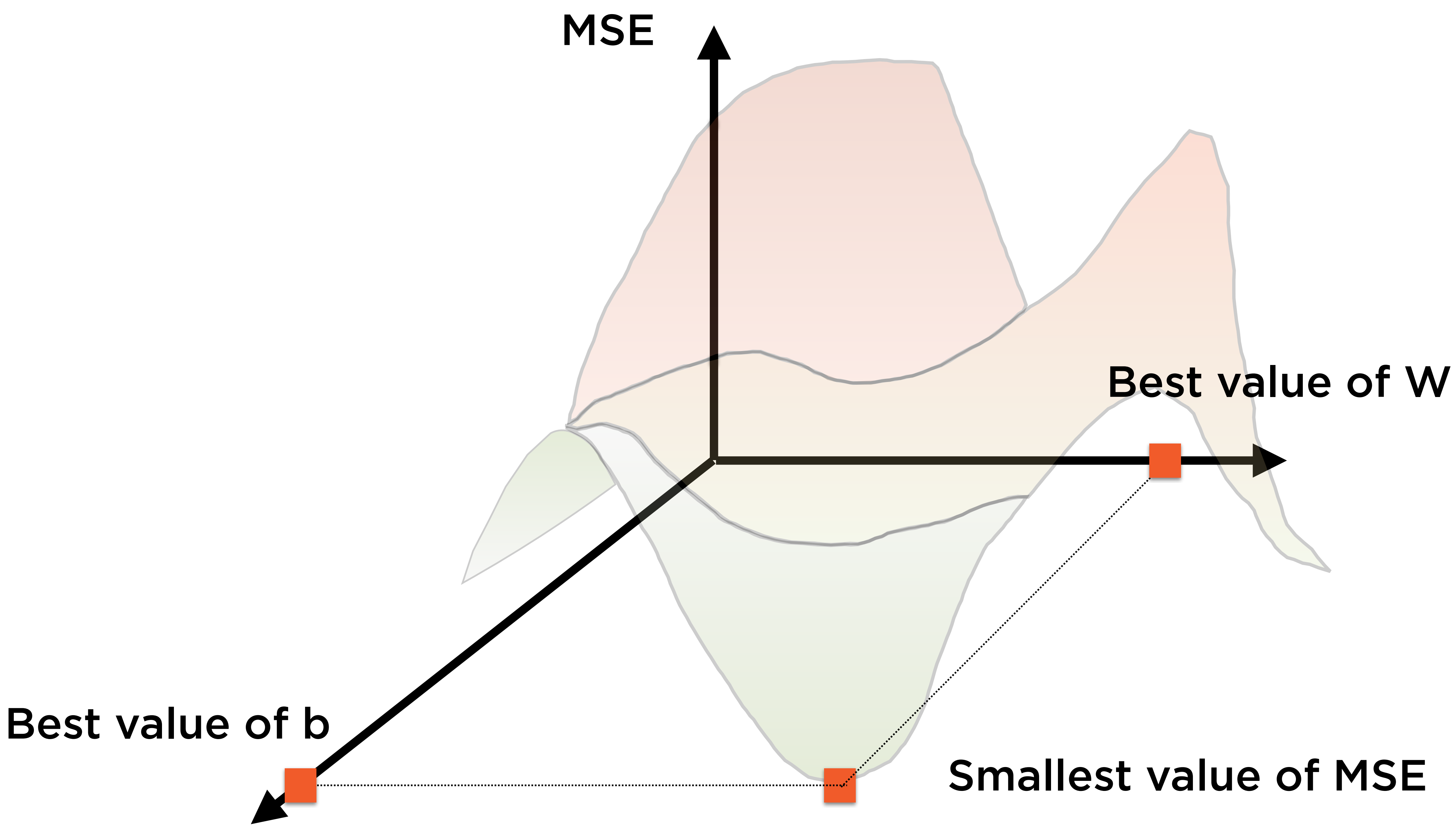
Minimizing MSE



Minimizing MSE

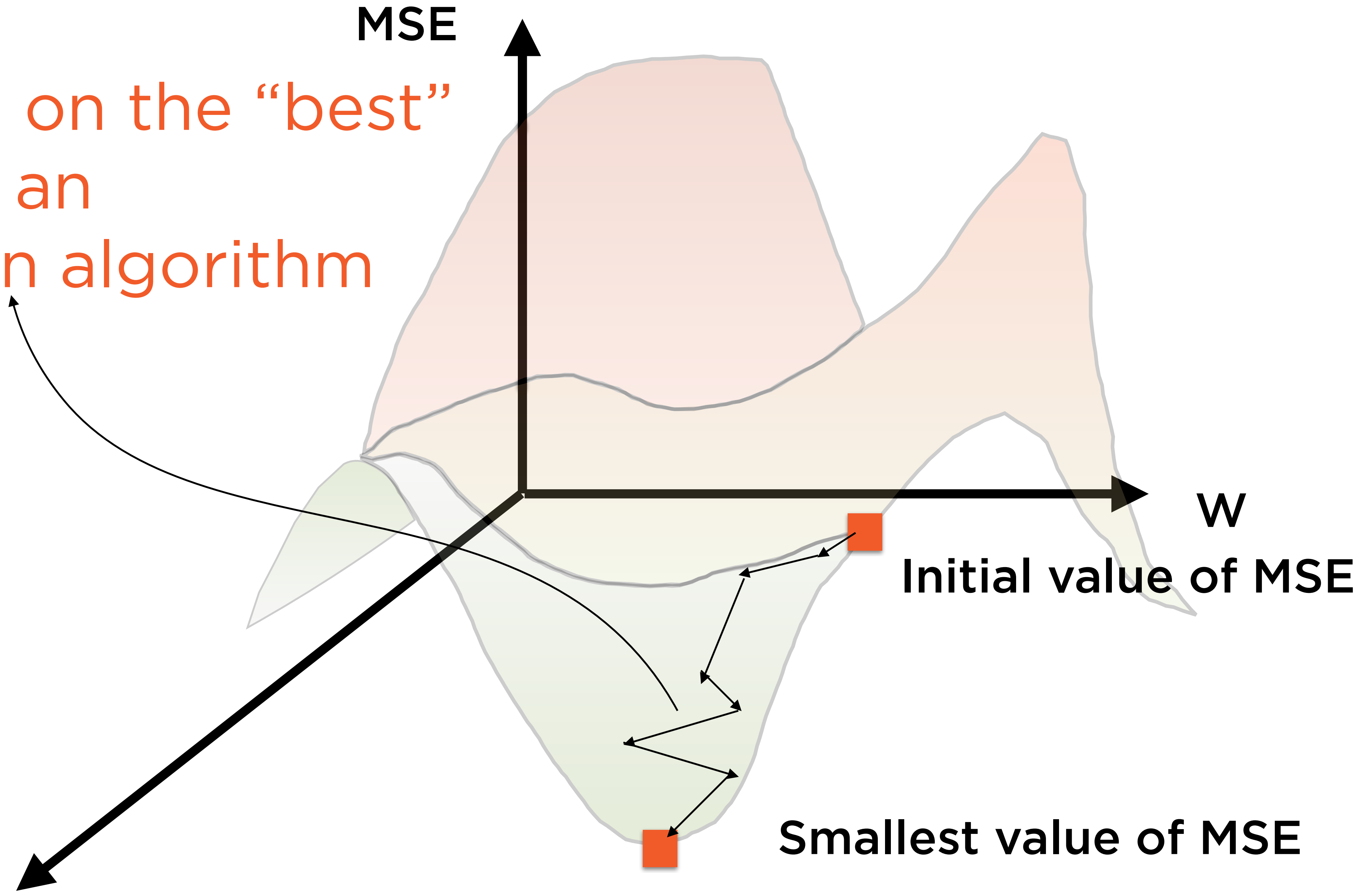


Minimizing MSE

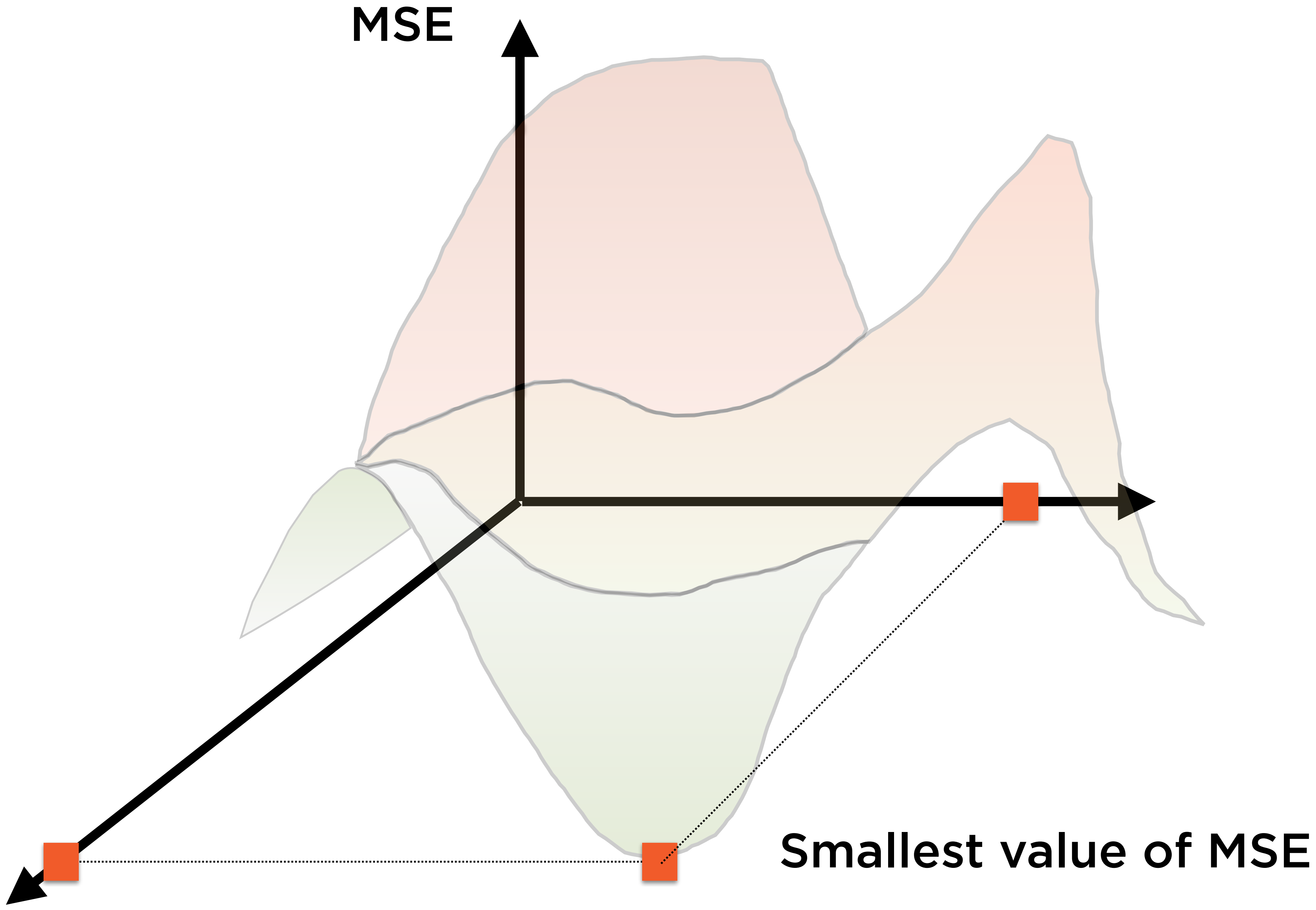


“Gradient Descent”

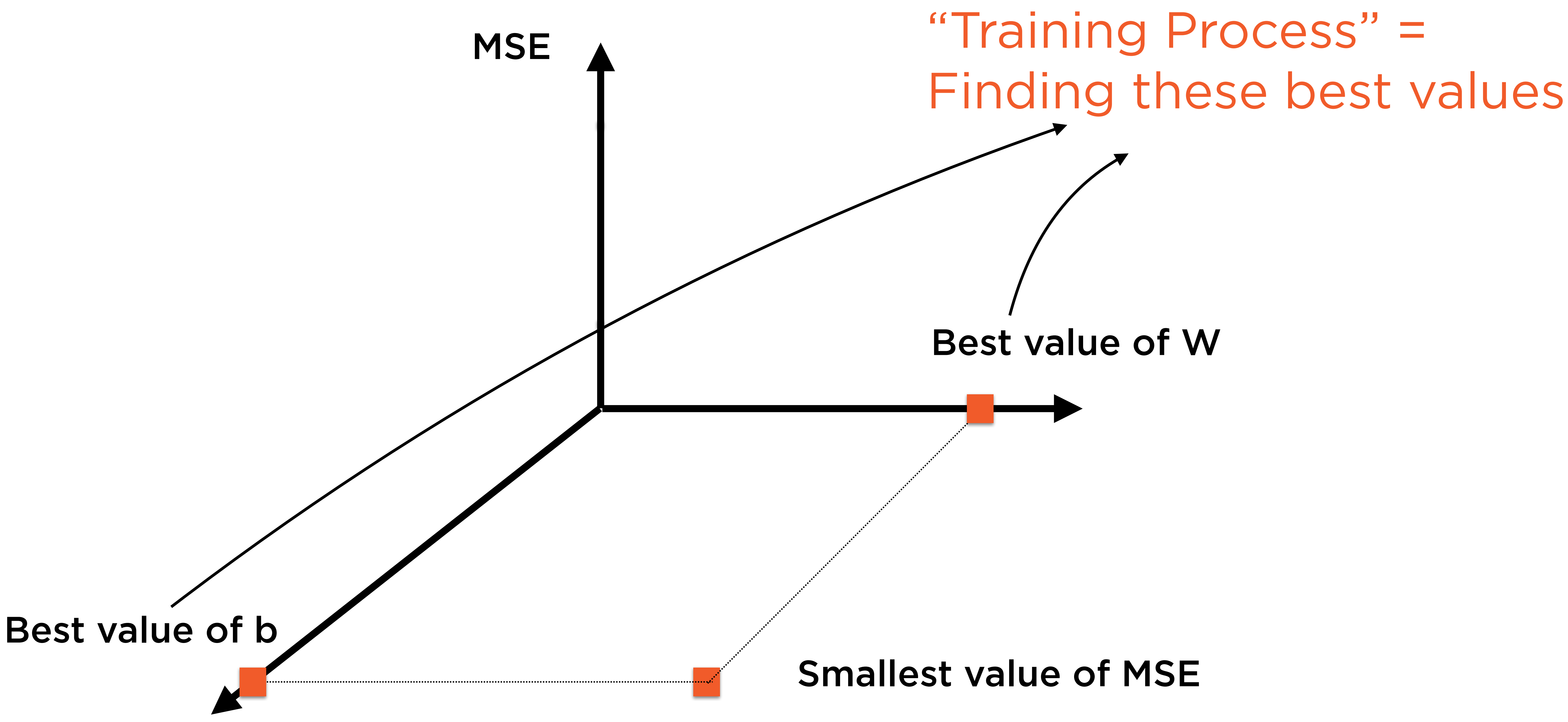
Converging on the “best” value using an optimization algorithm



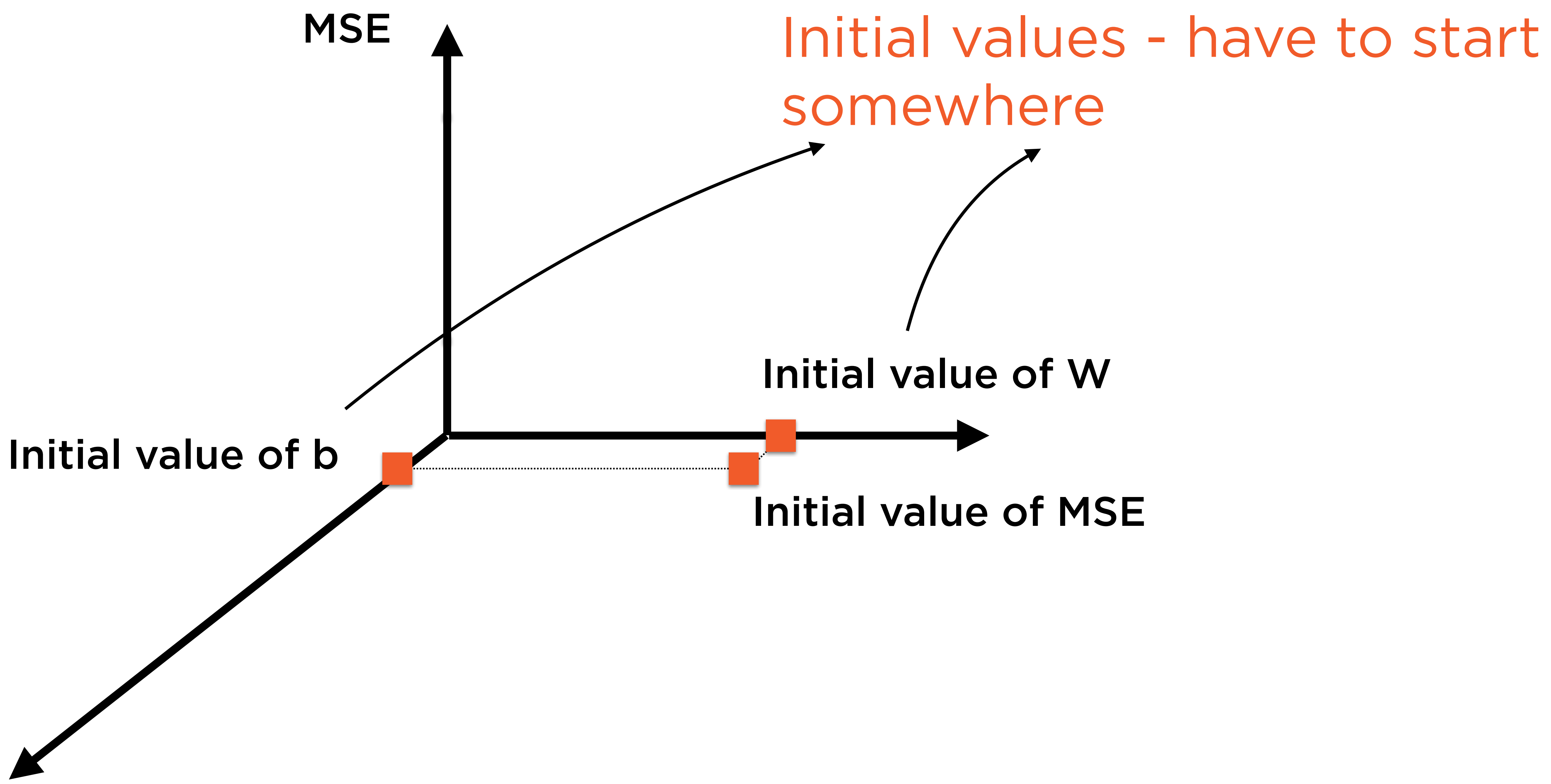
Minimizing MSE



“Training” the Algorithm

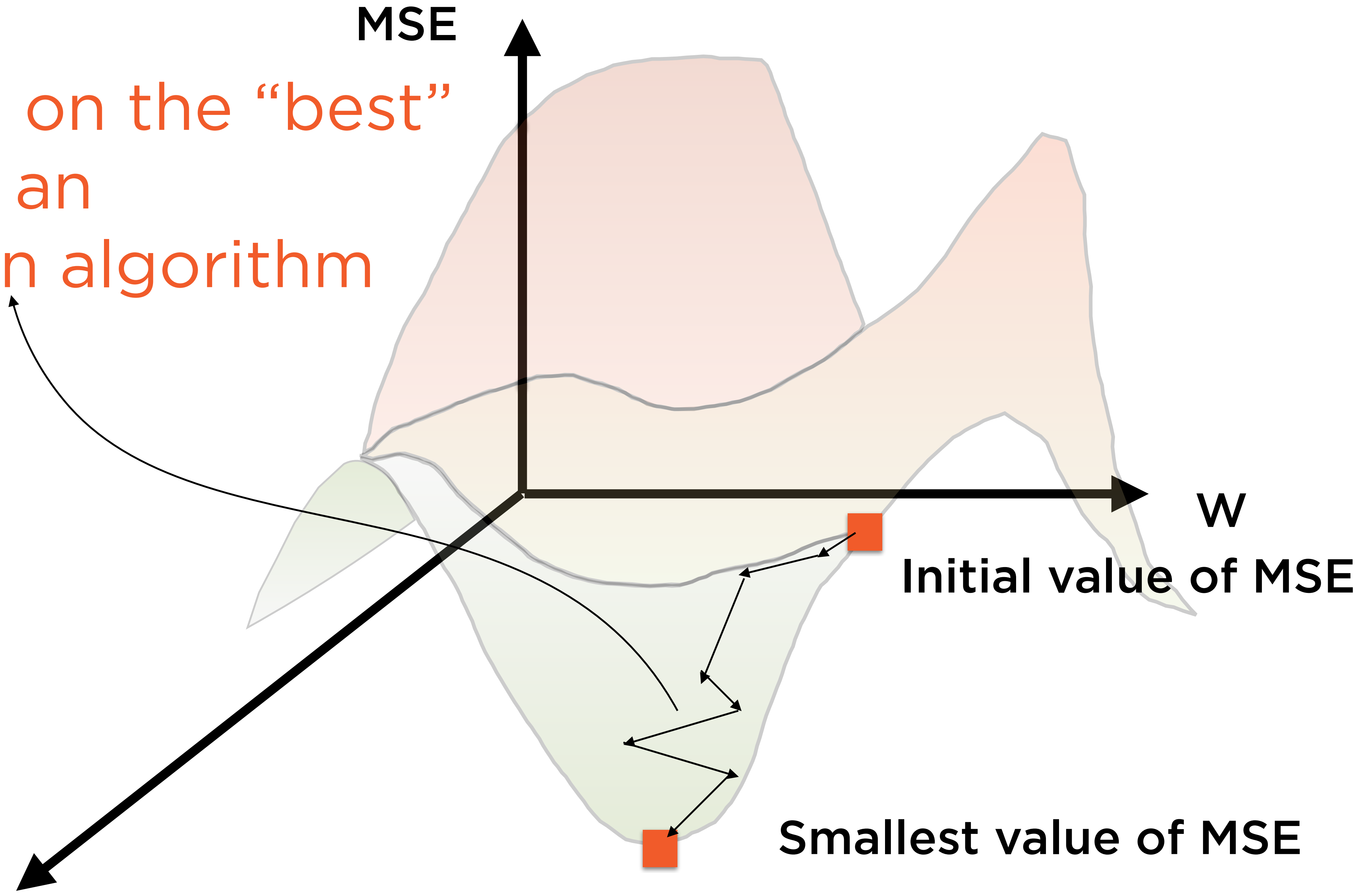


Start Somewhere



“Gradient Descent”

Converging on the “best” value using an optimization algorithm

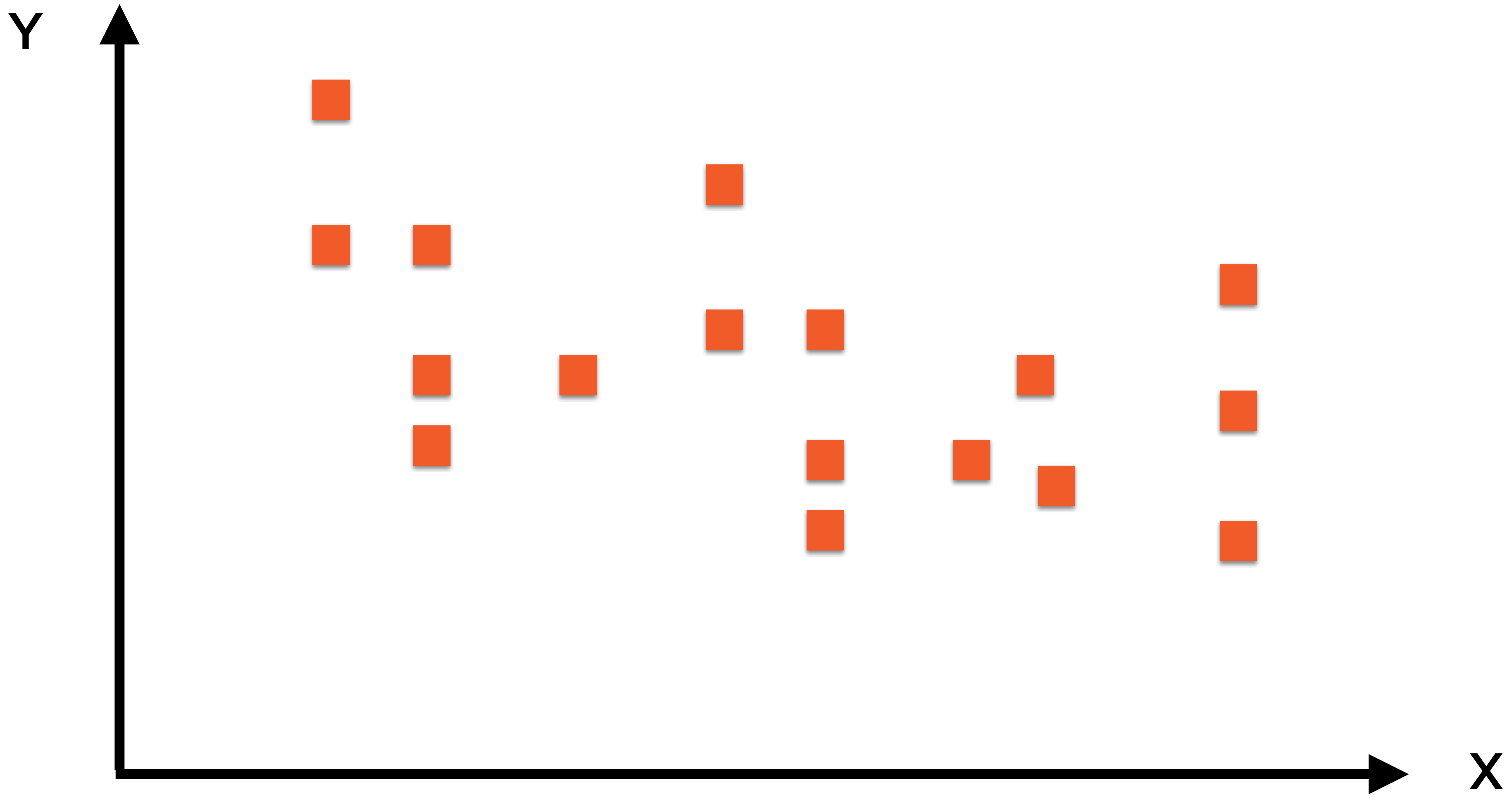


Demo

Simple Regression Using Weights Biases and Autograd

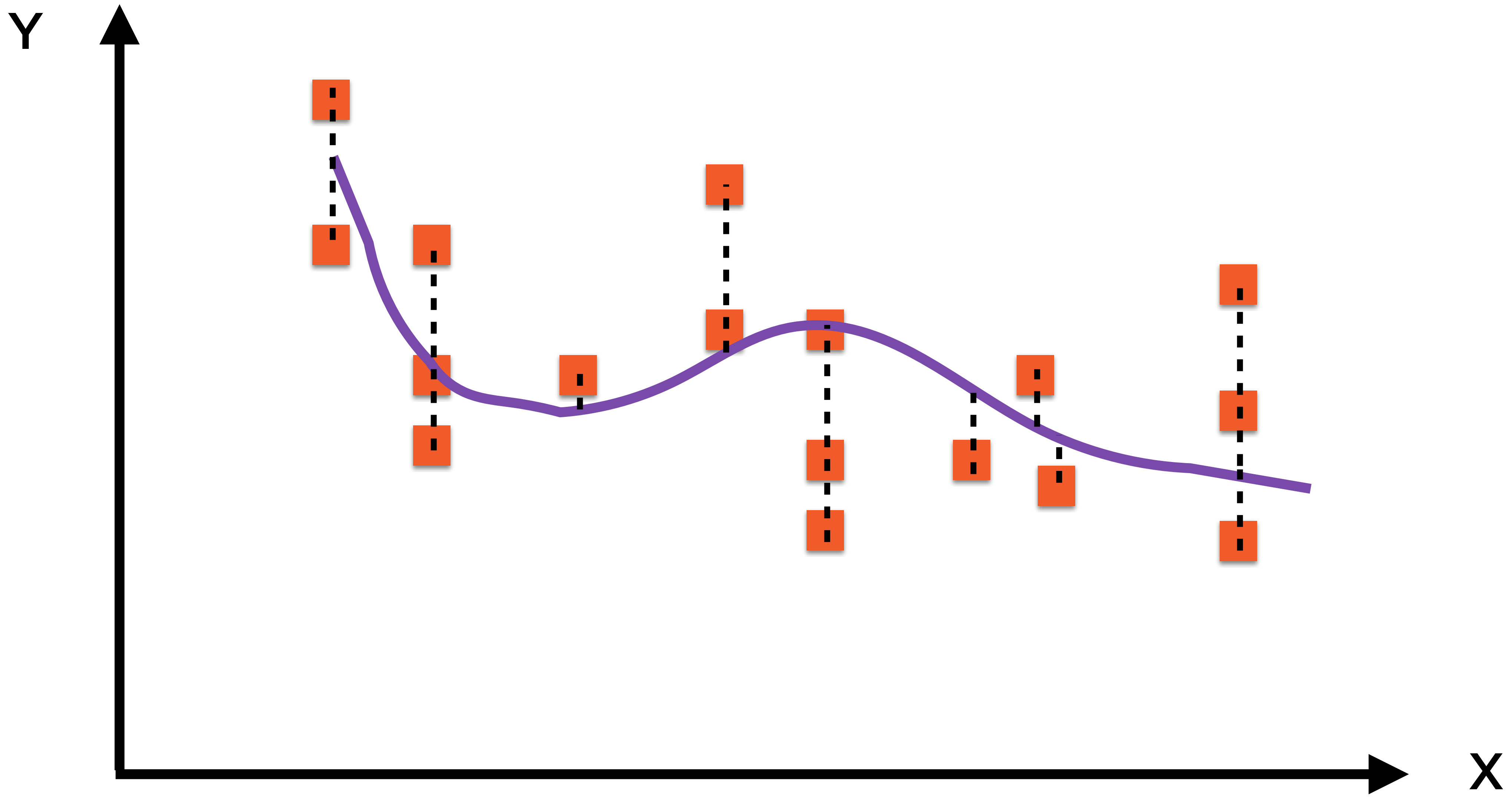
Overfitted Models

Connecting the Dots



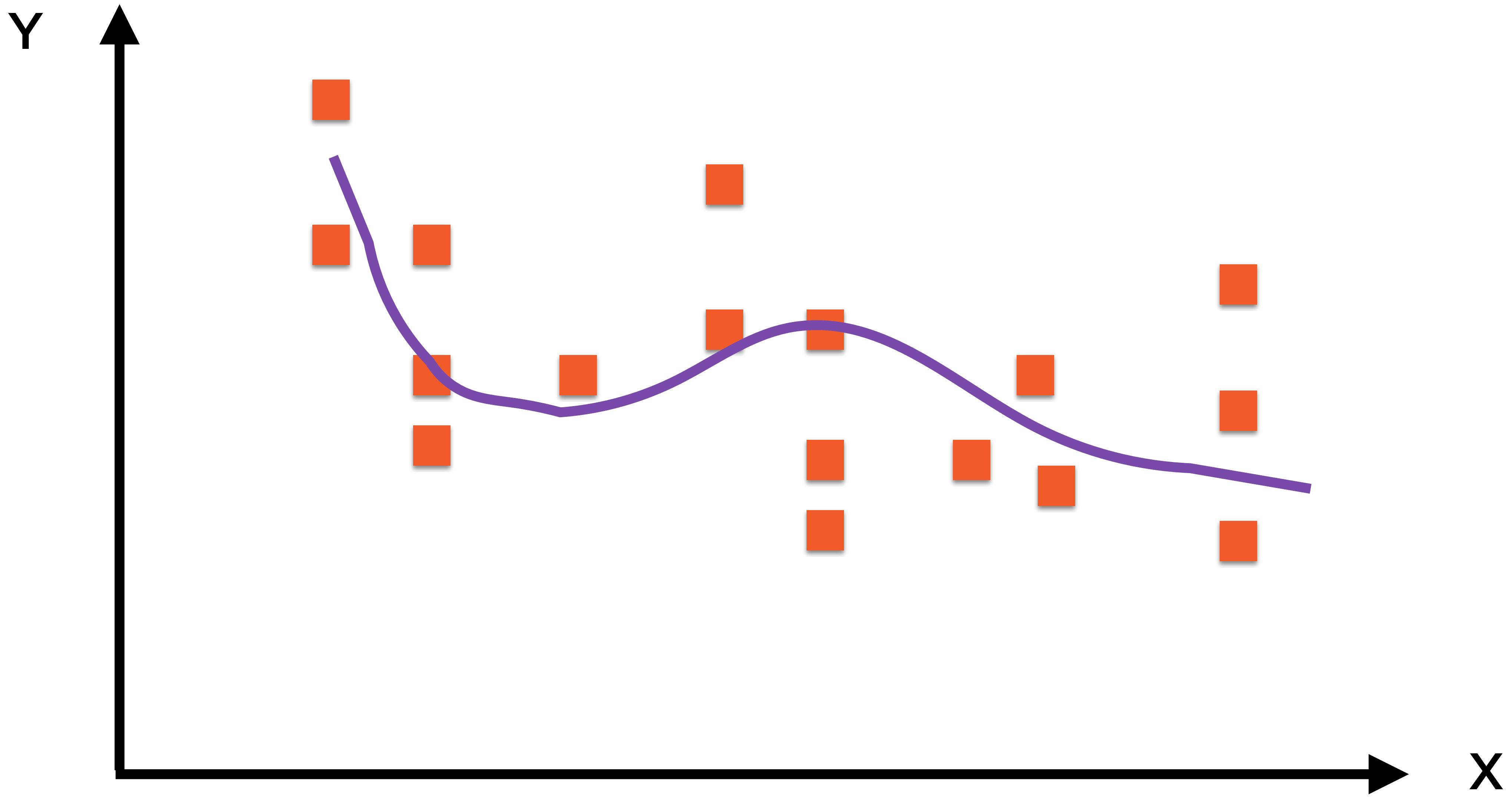
Challenge: Fit the "best" curve through these points

Good Fit?



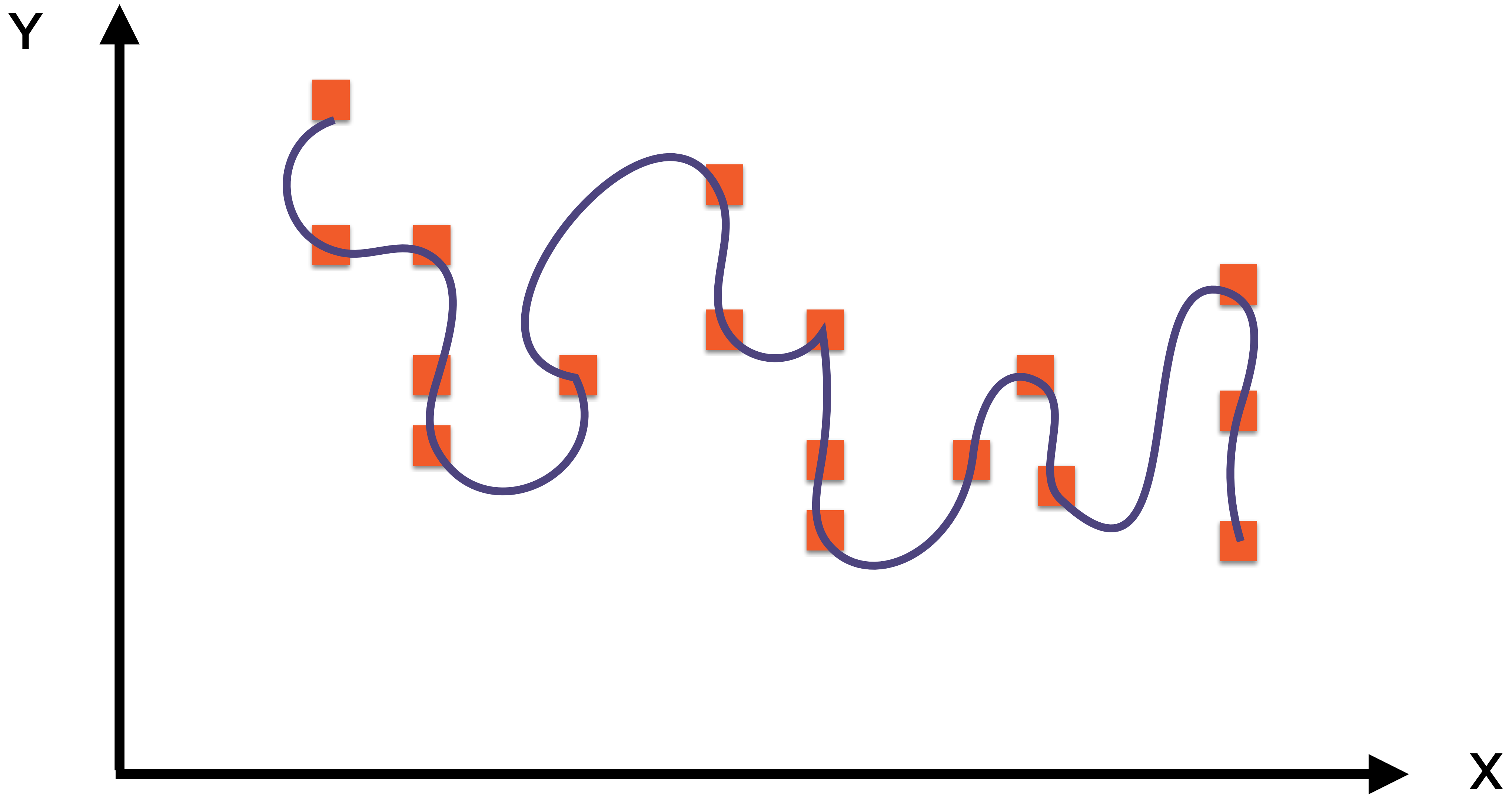
A curve has a "good fit" if the distances of points from the curve are small

Connecting the Dots



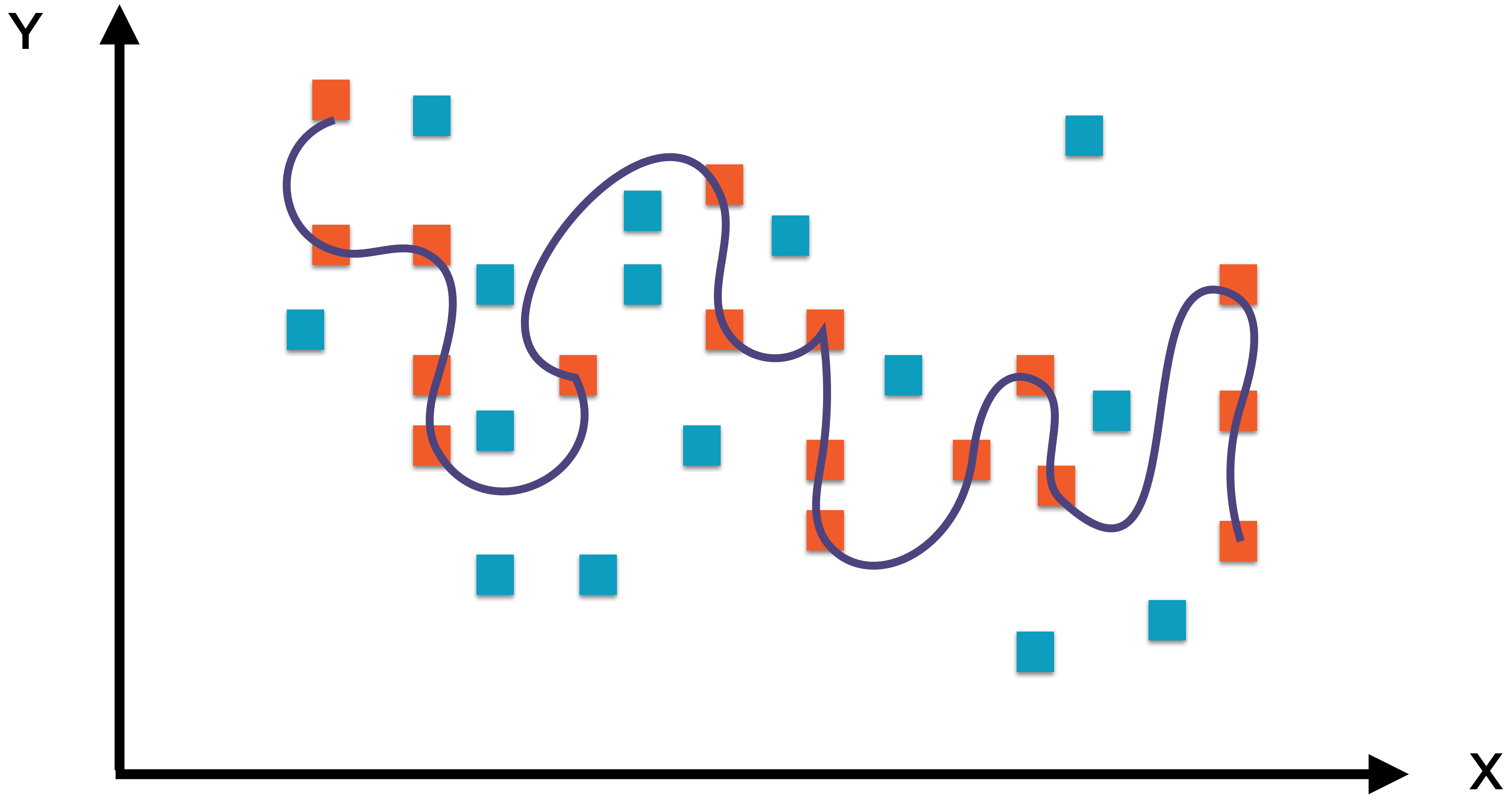
We could draw a pretty complex curve

Connecting the Dots



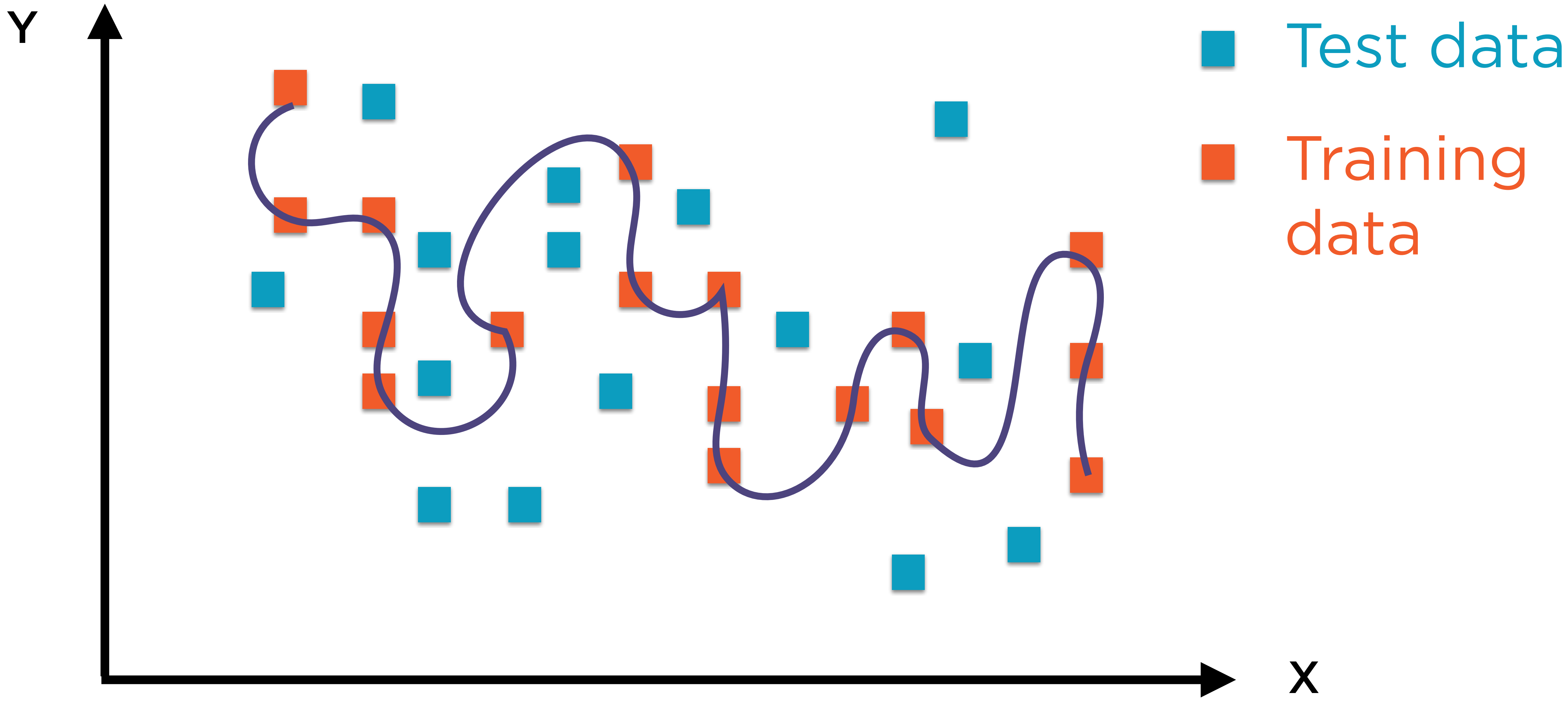
We can even make it pass through every single point

Connecting the Dots



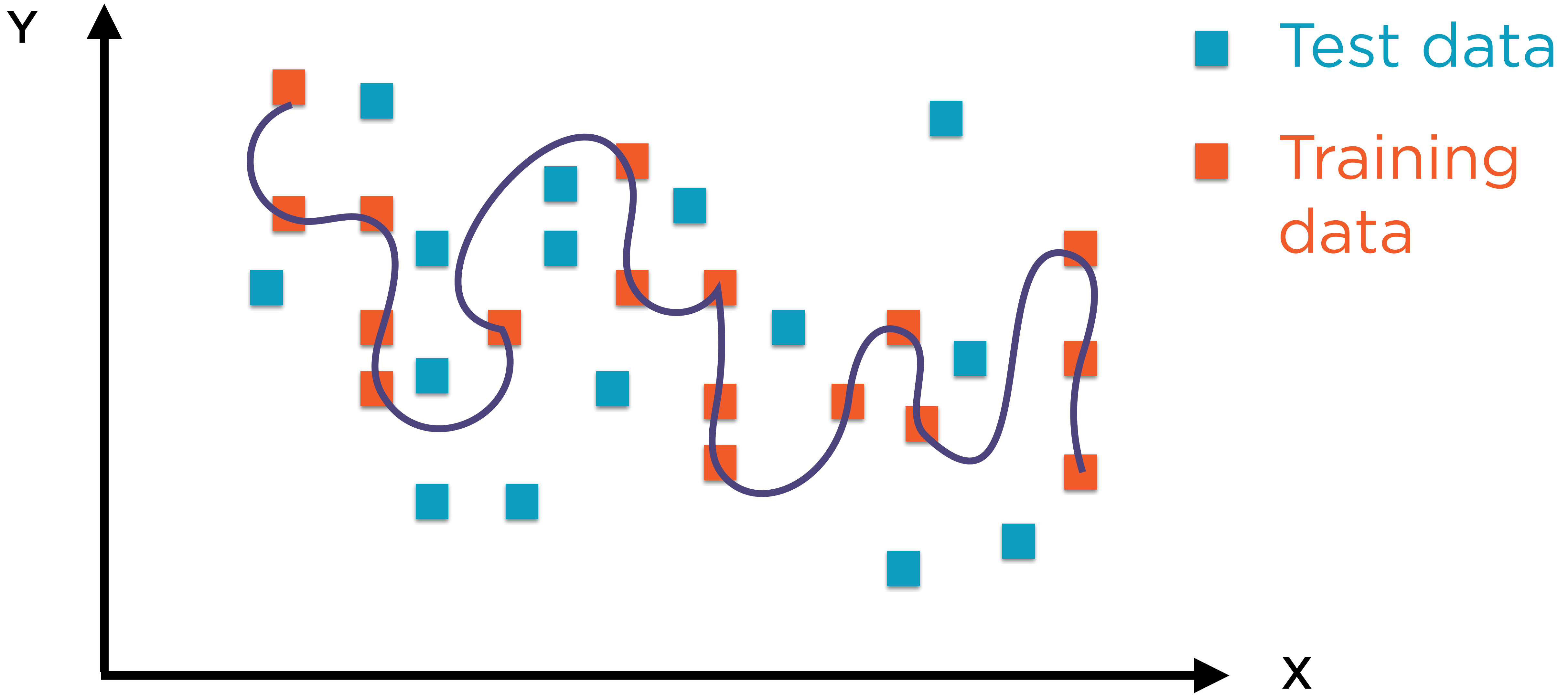
But given a new set of points, this curve might perform quite poorly

Connecting the Dots



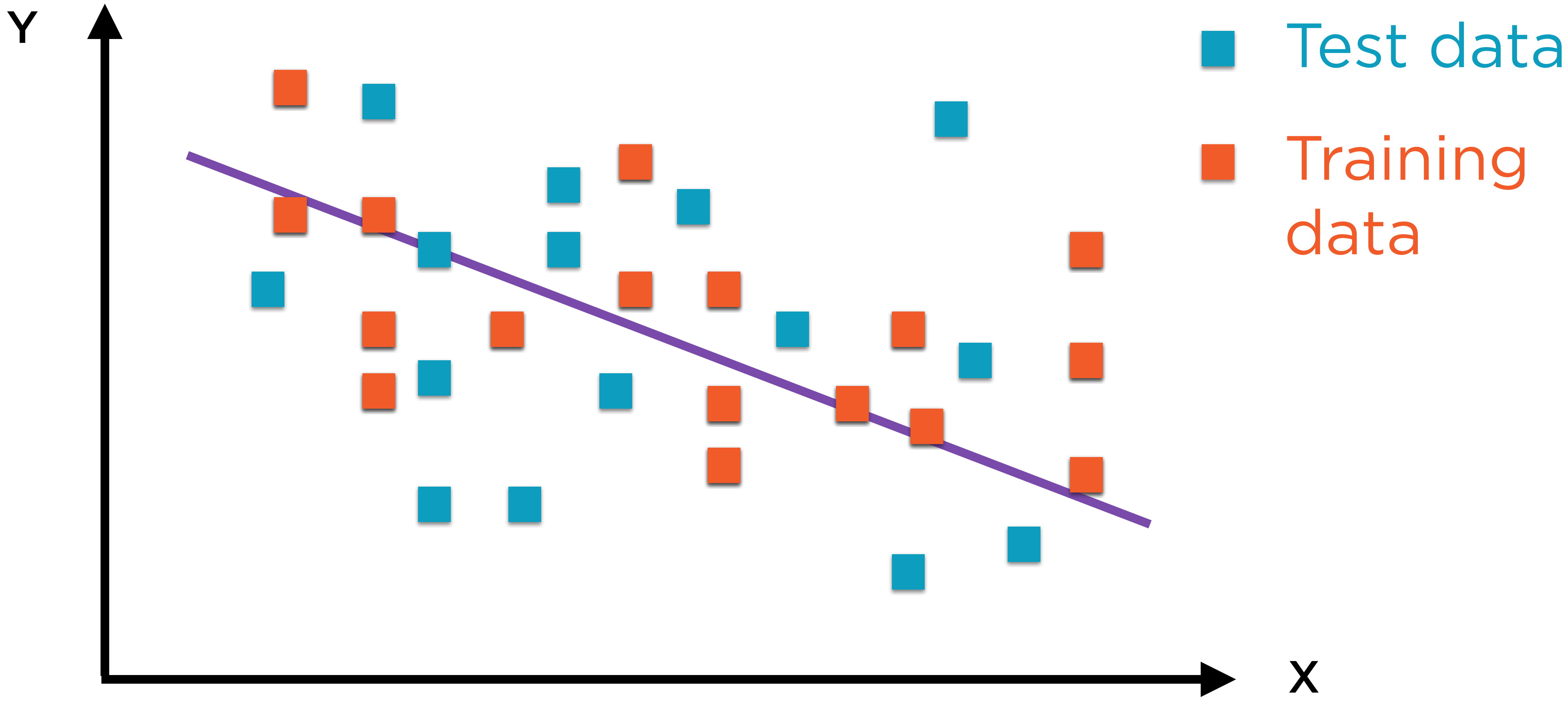
The original points were "training data", the new points are "test data"

Overfitting



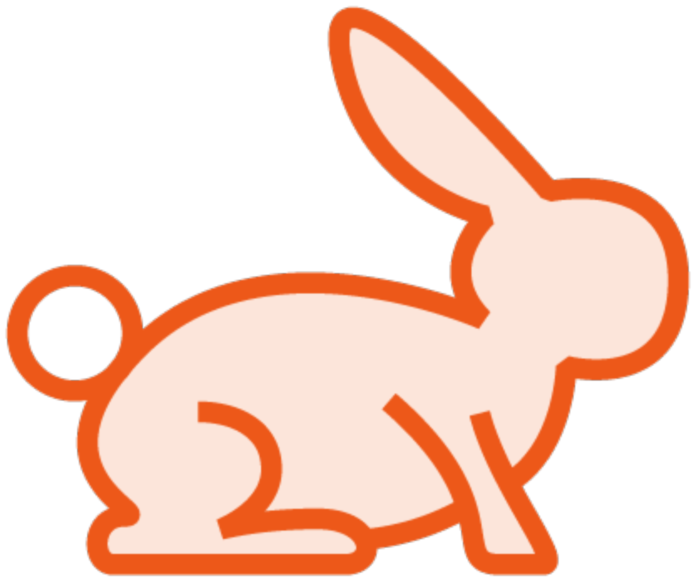
Great performance in training, poor performance in real usage

Connecting the Dots



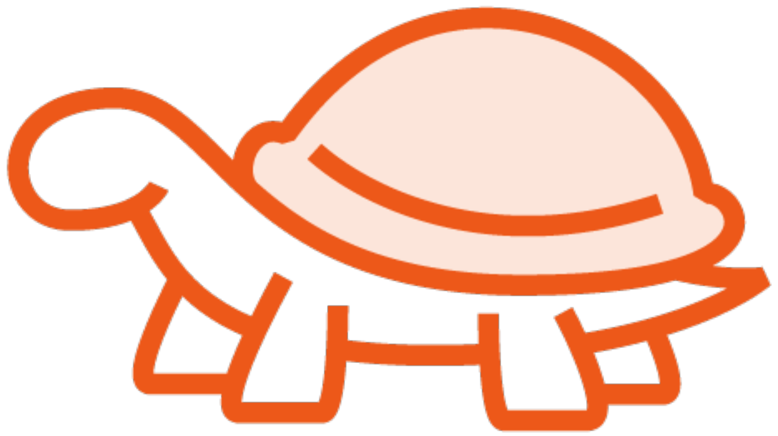
A simple straight line performs worse in training, but better with test data

Overfitting



Low Training Error

Model does very well in training...

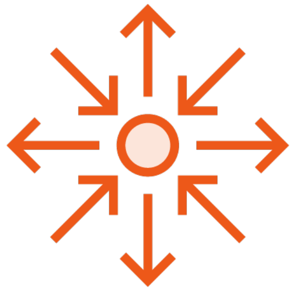


High Test Error

...but poorly with real data

Preventing Overfitting

Regularization - Penalize complex models



Cross-validation - Distinct training and validation phases

Dropout (NNs only) - Intentionally turn off some neurons during training

Regularization

Penalize complex models

Add penalty to objective function

Penalty as function of regression coefficients

Forces optimizer to keep it simple

Regularization

Regularization reduces variance error

But increases bias

Ordinary MSE Regression

Minimize

$$\sqrt{(y^{actual} - y^{predicted})^2}$$

To find

A, B

The value of A and B define the "best fit" line

y = A + Bx

Ridge Regression

ridge regression(岭回归)

Minimize $\sqrt{(y_{actual} - y_{predicted})^2} + \alpha \, (|A|^2 + |B|^2)$

L-2 Norm of regression coefficients

To find

A, B

α is a hyperparameter

The value of A and B still define the "best fit" line

$y = A + Bx$

Ridge Regression

Minimize

$$\sqrt{(y_{actual} - y_{predicted})^2} + \alpha\,(|A|^2 + |B|^2)$$

L-2 Norm of regression coefficients

To find

A, B

α is a hyperparameter

The value of A and B still define the “best fit” line

$y = A + Bx$

Ridge Regression

Add penalty for large coefficients

Penalty term is L-2 norm of coefficients

Penalty weighted by hyperparameter α

Demo

Implementing Ridge Regression

Summary

Using linear regression for prediction

Linear regression using a single neuron

Hand-crafting an MSE regression model

Hand-crafting a Ridge regression model

Comparing to scikit-learn's linear regression estimator