

# Building Specialized Regression Models in scikit-learn

---

# Overview

**Regression models and measuring fit of a model**

**The bias-variance trade-off and overfitted models**

**Lasso and Ridge regression to mitigate overfitting**

**Support vector regression models**

# Setting Up The Regression Problem

---

# Types of Machine Learning Problems



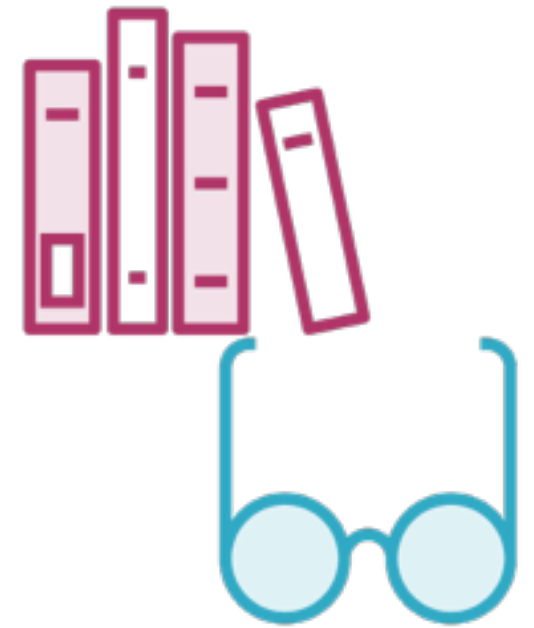
**Classification**



**Regression**



**Clustering**



**Rule-extraction**

# Types of Machine Learning Problems



Classification



**Regression**



Clustering



Rule-extraction

X Causes Y



**Cause**

**Independent variable**



**Effect**

**Dependent variable**

X Causes Y



**Cause**

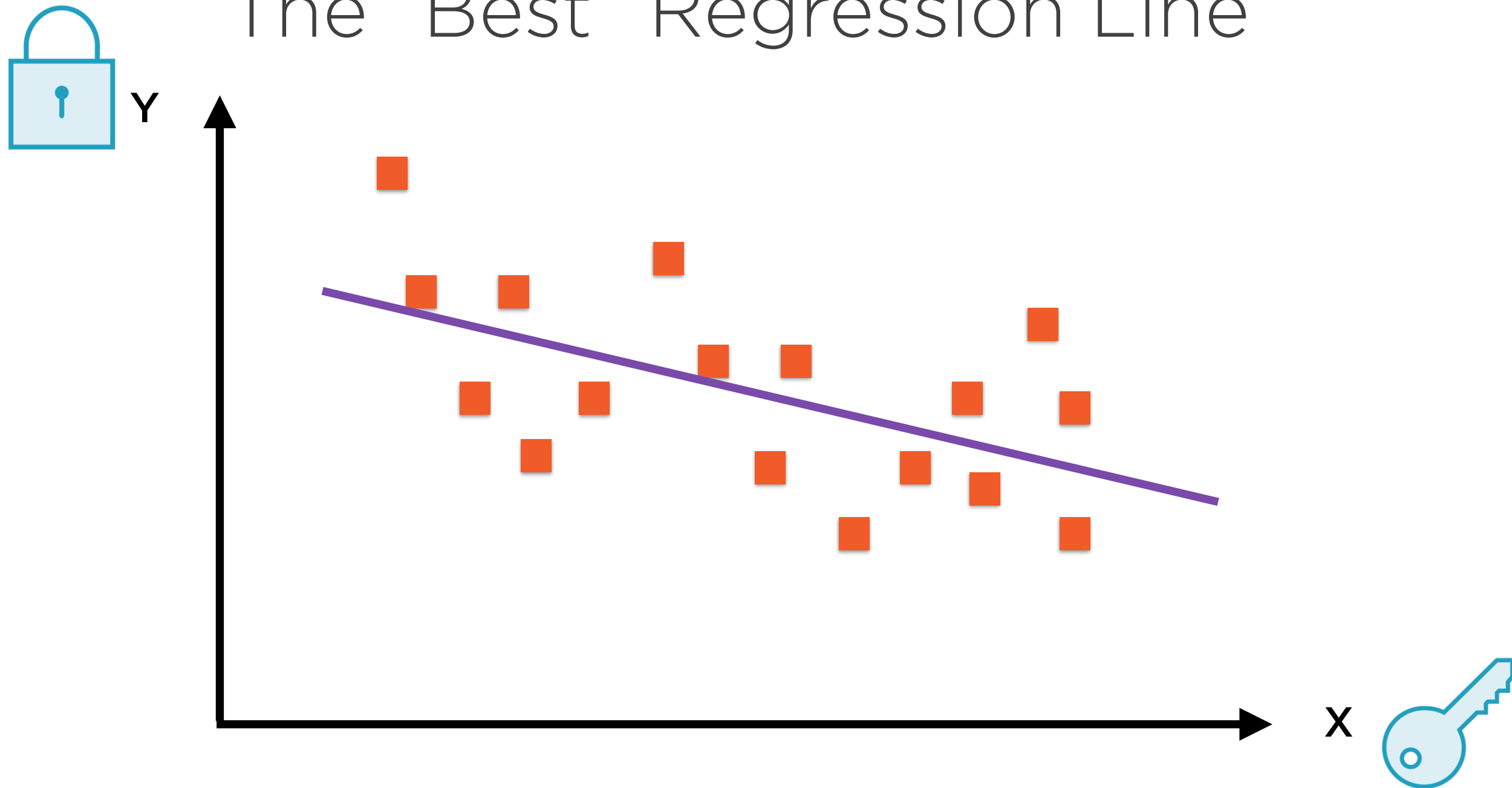
**Explanatory variable**



**Effect**

**Dependent variable**

# The “Best” Regression Line



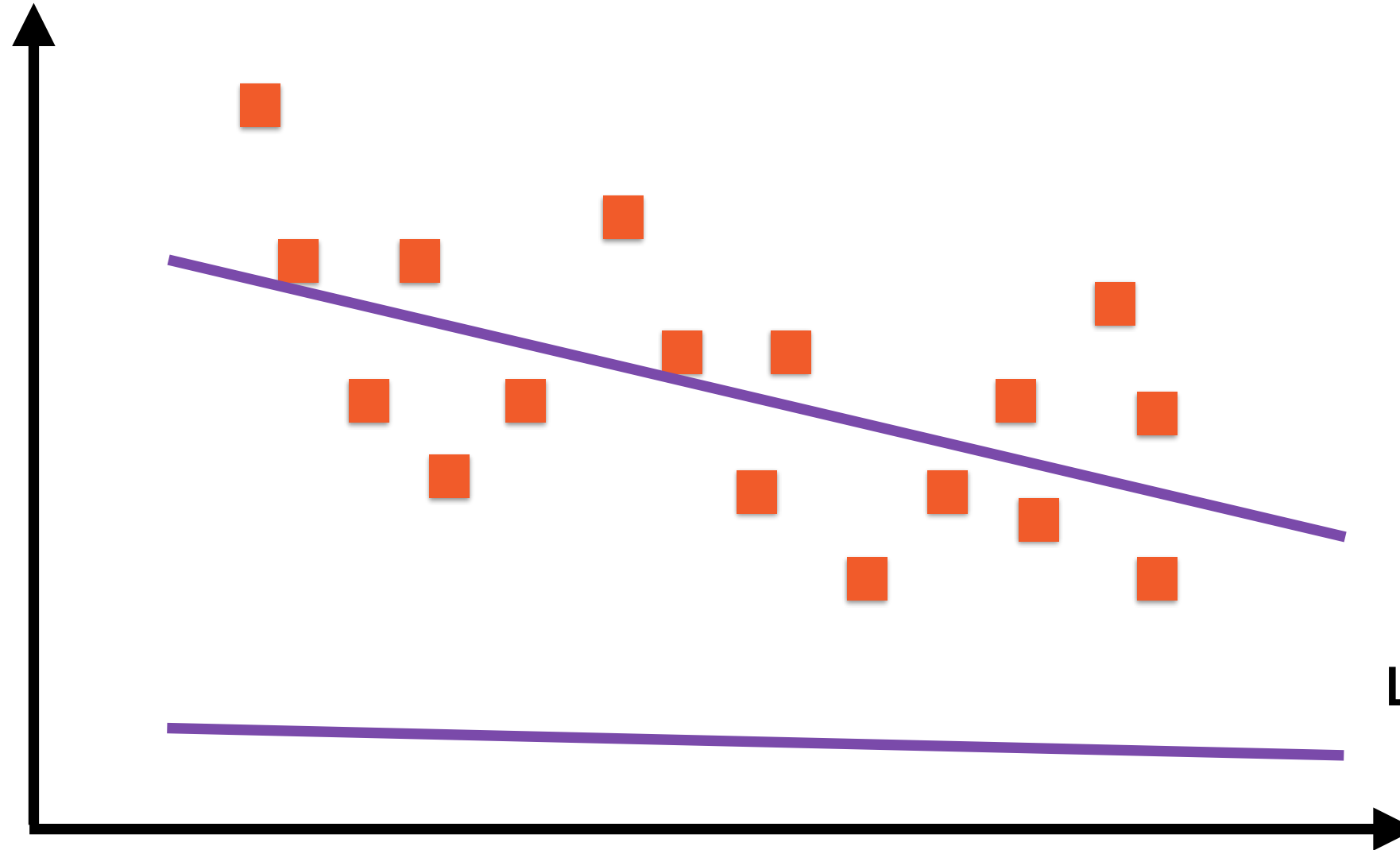
Linear Regression involves finding the “best fit” line



# The “Best” Regression Line



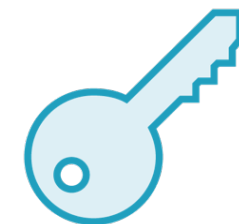
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

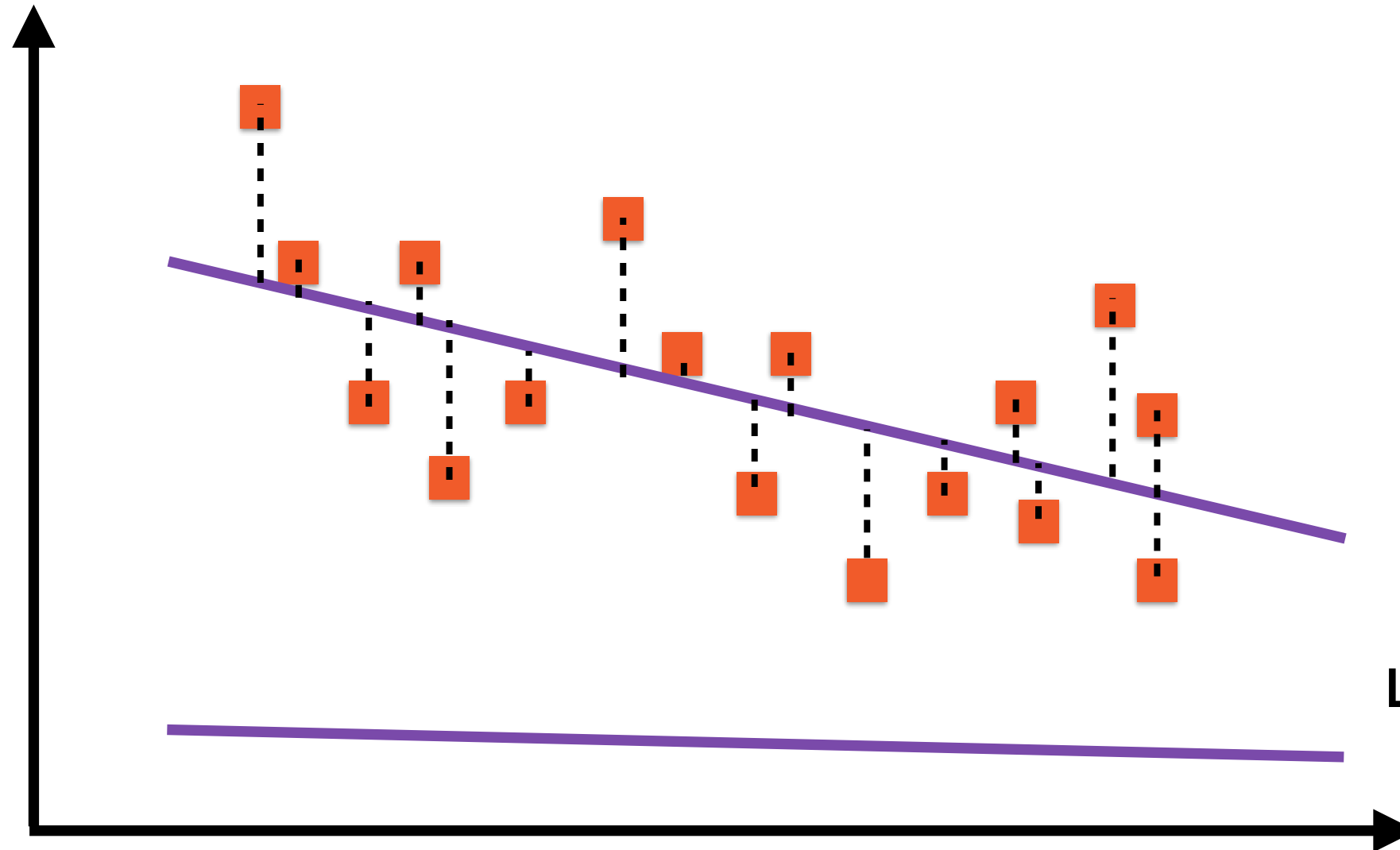


Let's compare two lines, Line 1 and Line 2

# Minimising Least Square Error



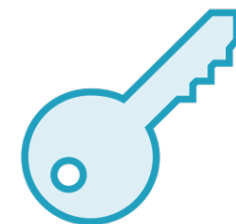
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

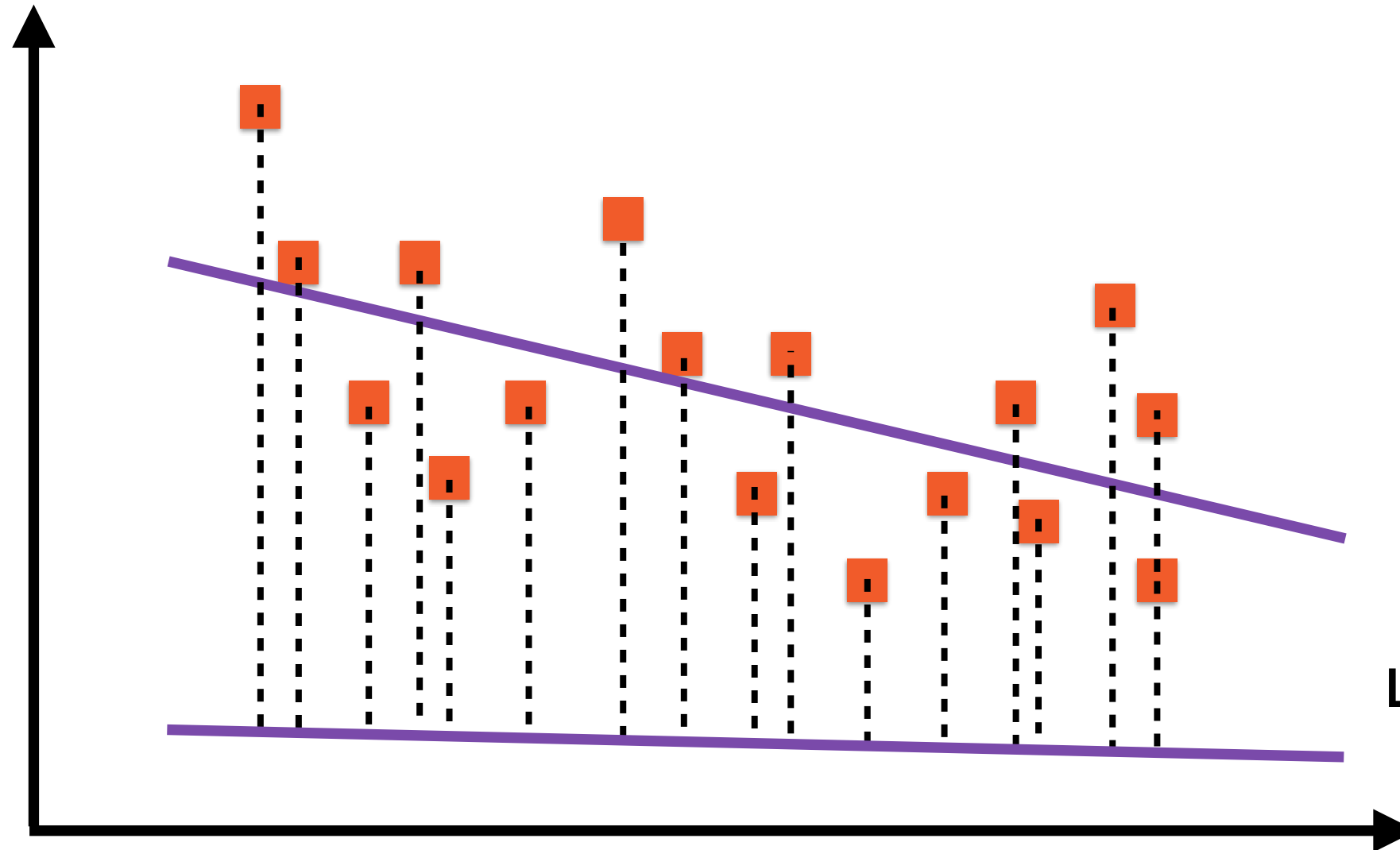


Drop vertical lines from each point to  
the lines 1 and 2

# Minimising Least Square Error



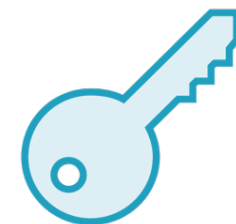
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

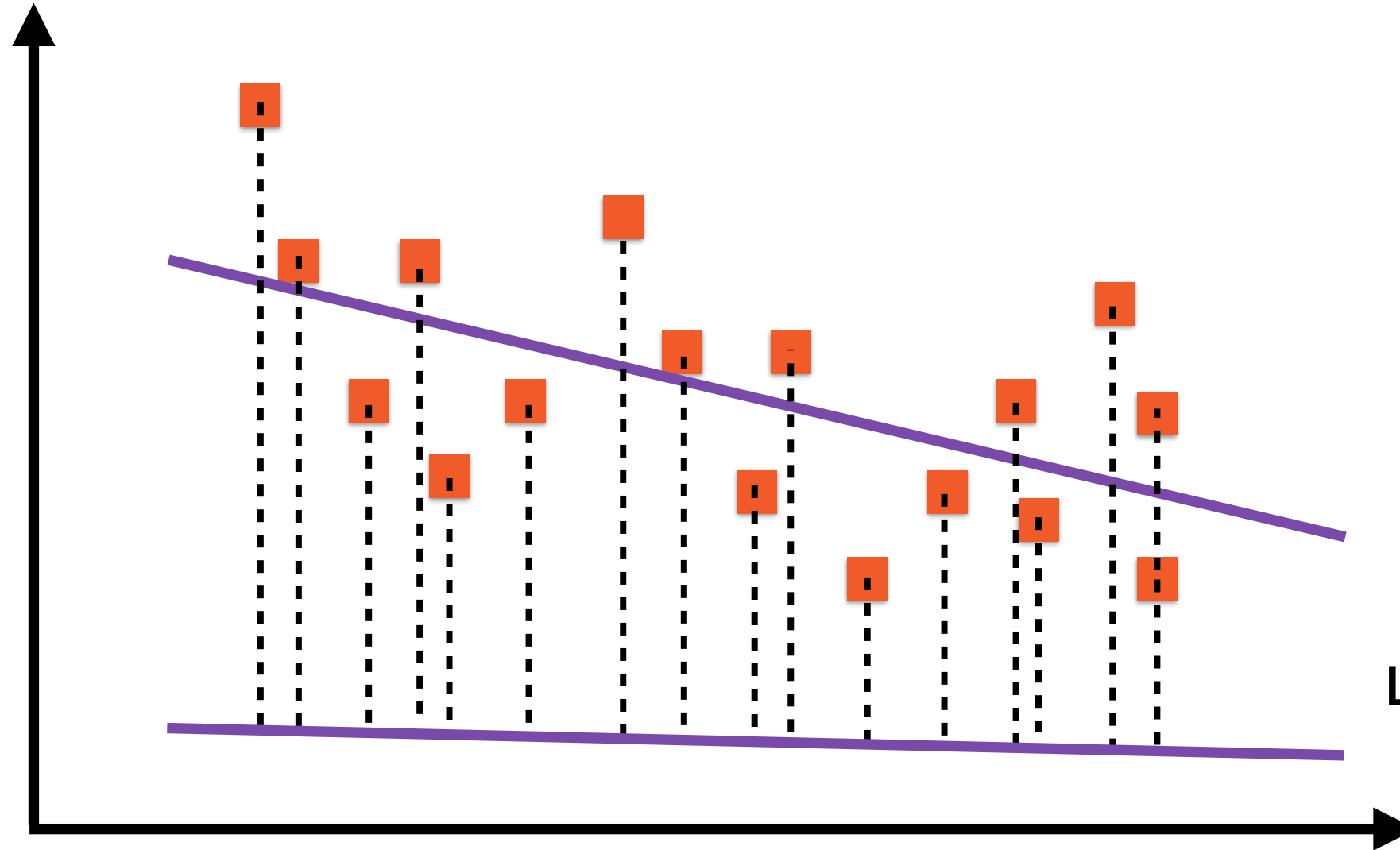


Drop vertical lines from each point to  
the lines 1 and 2

# Minimising Least Square Error



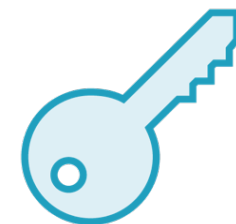
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

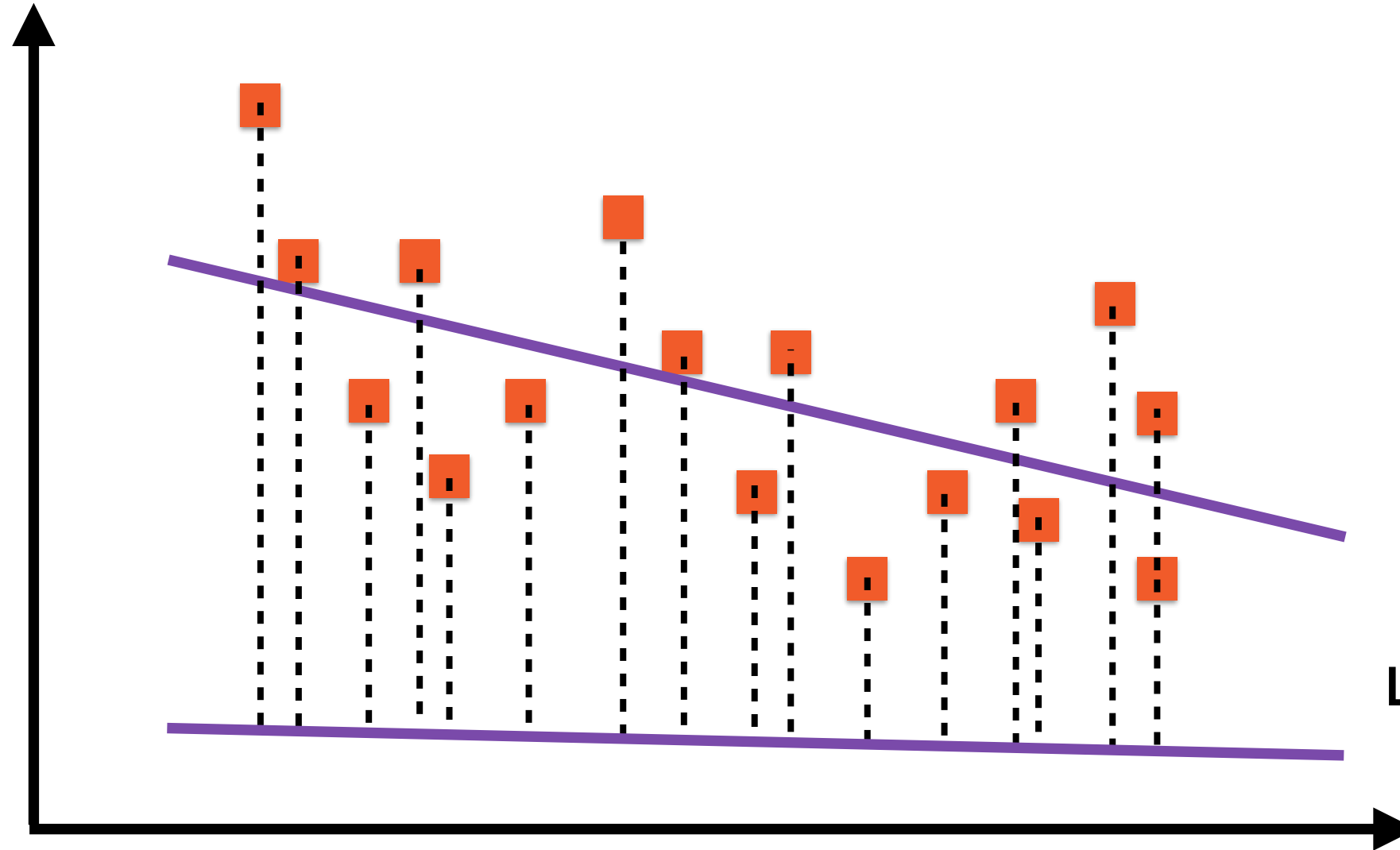


The “best fit” line is the one where the sum of the squares of the lengths of these dotted lines is minimum

# Minimising Least Square Error



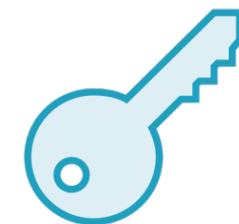
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

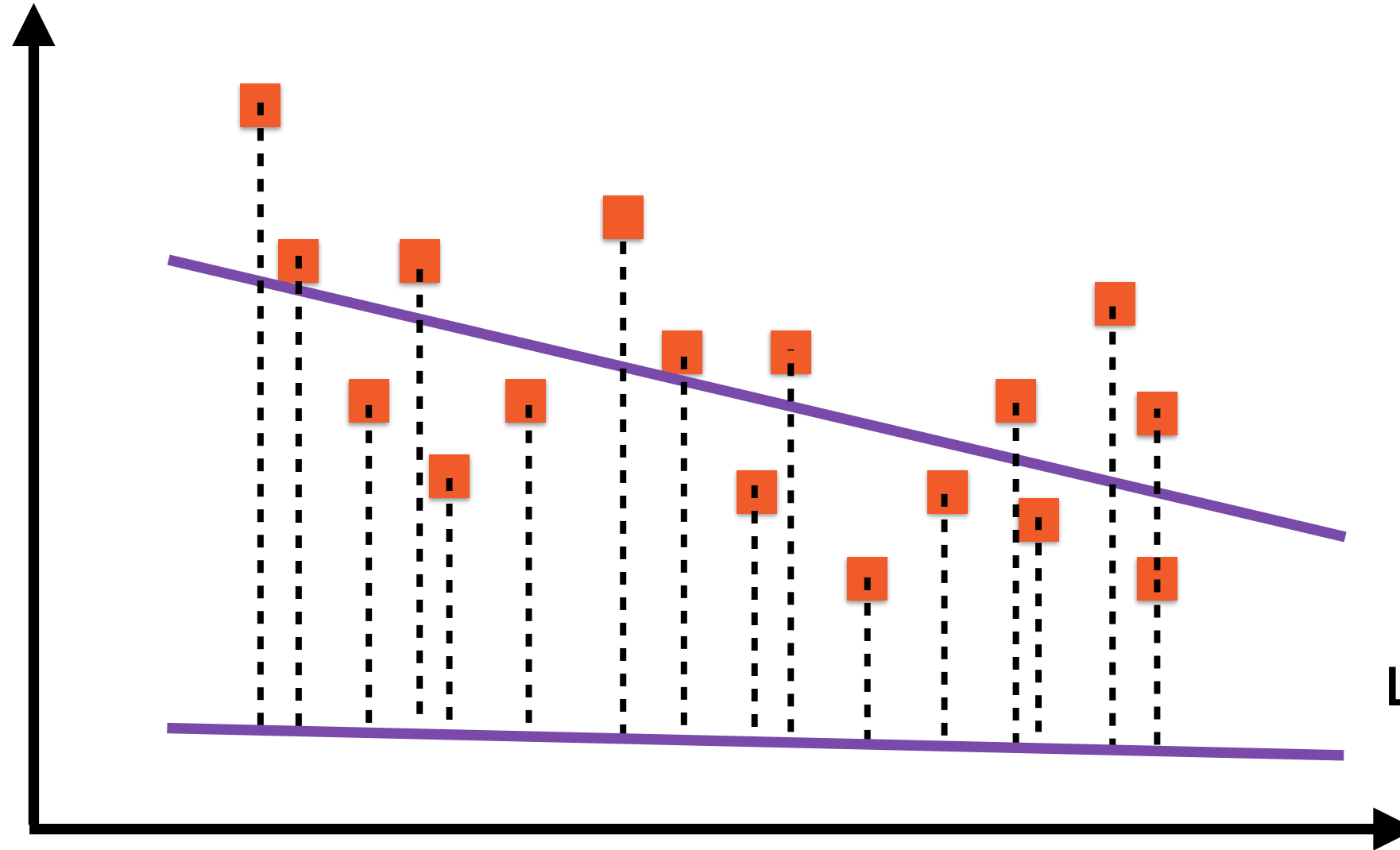


The “best fit” line is the one where the sum of the squares of the **lengths of these dotted lines** is minimum

# Minimising Least Square Error



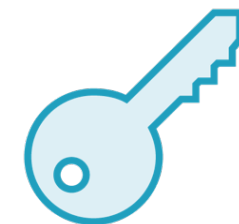
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

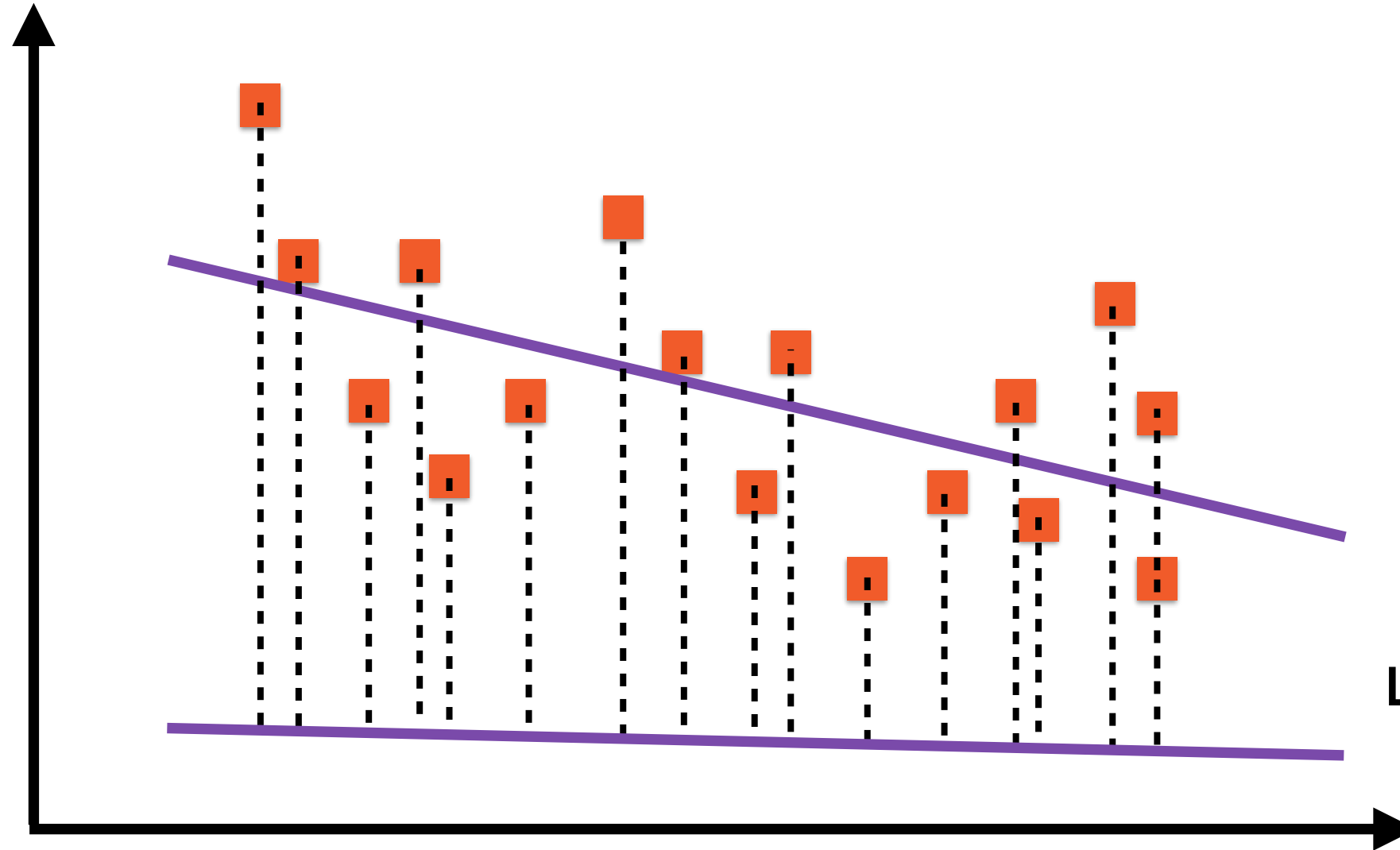


The “best fit” line is the one where the  
**sum of the squares** of the lengths of  
these dotted lines is minimum

# Minimising Least Square Error



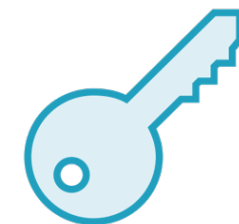
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

X

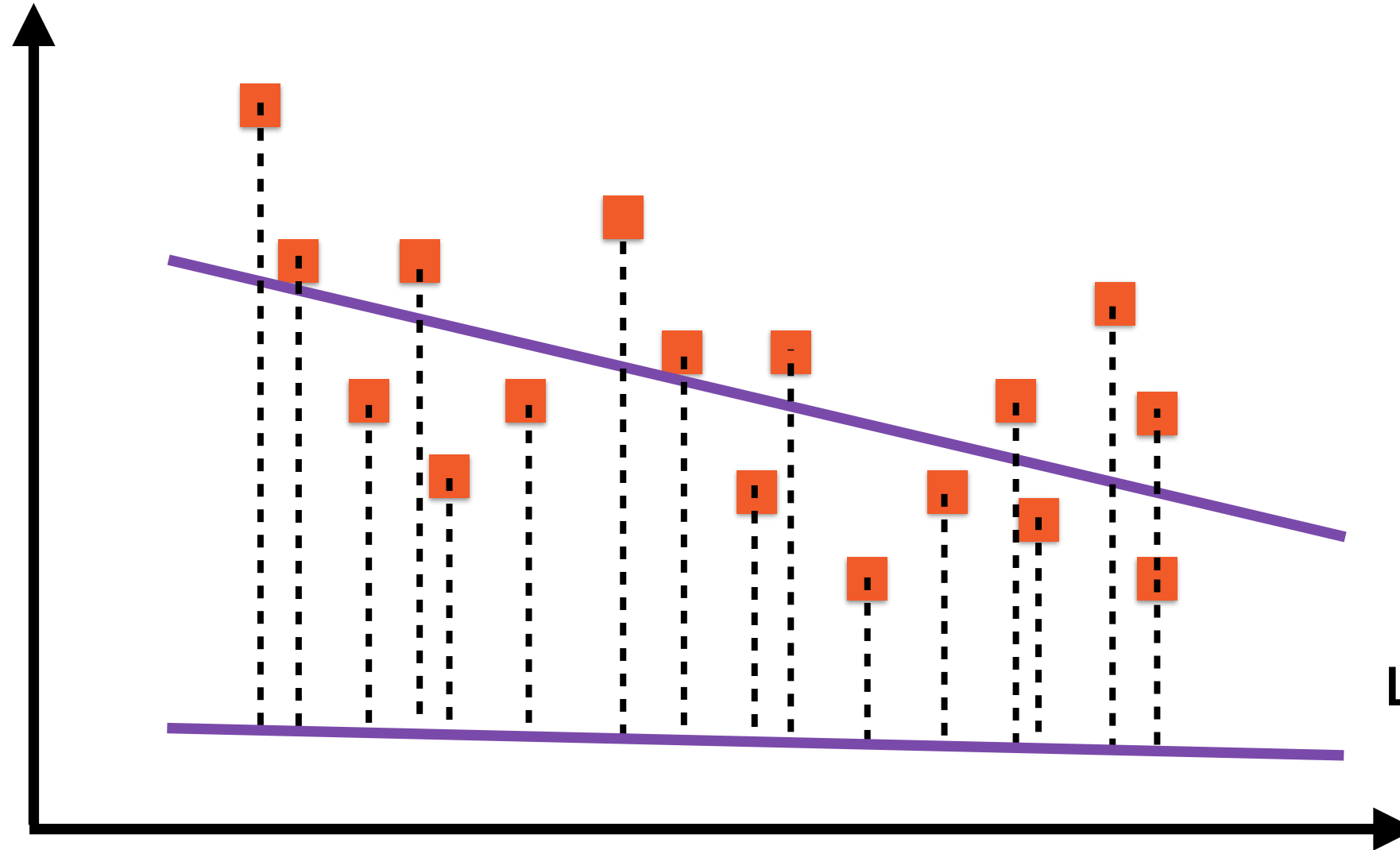


The “best fit” line is the one where the sum of the squares of the lengths of **the errors** is minimum

# Minimising Least Square Error



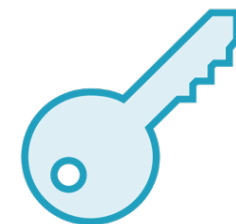
Y



Line 1:  $y = A_1 + B_1x$

Line 2:  $y = A_2 + B_2x$

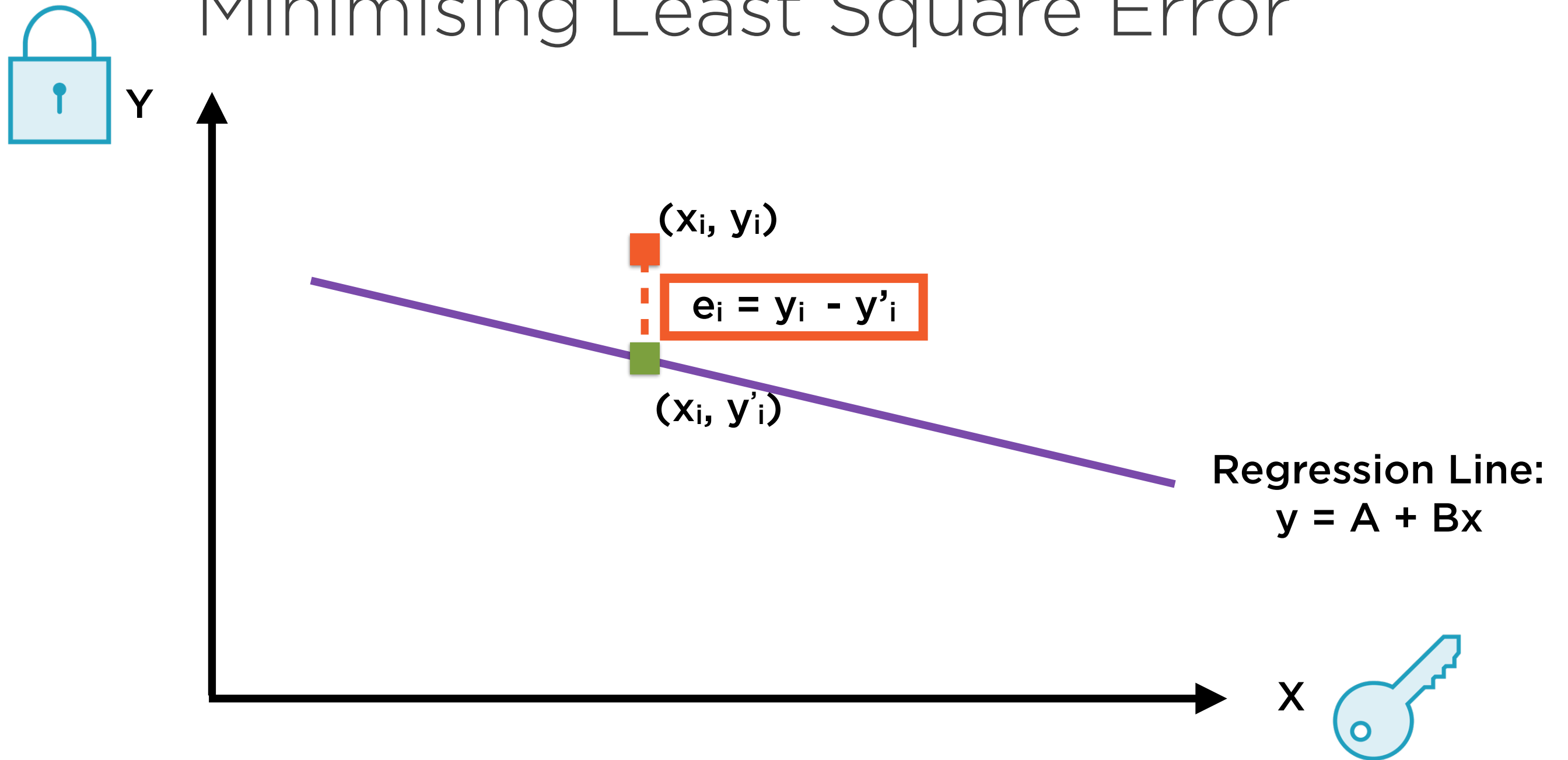
X



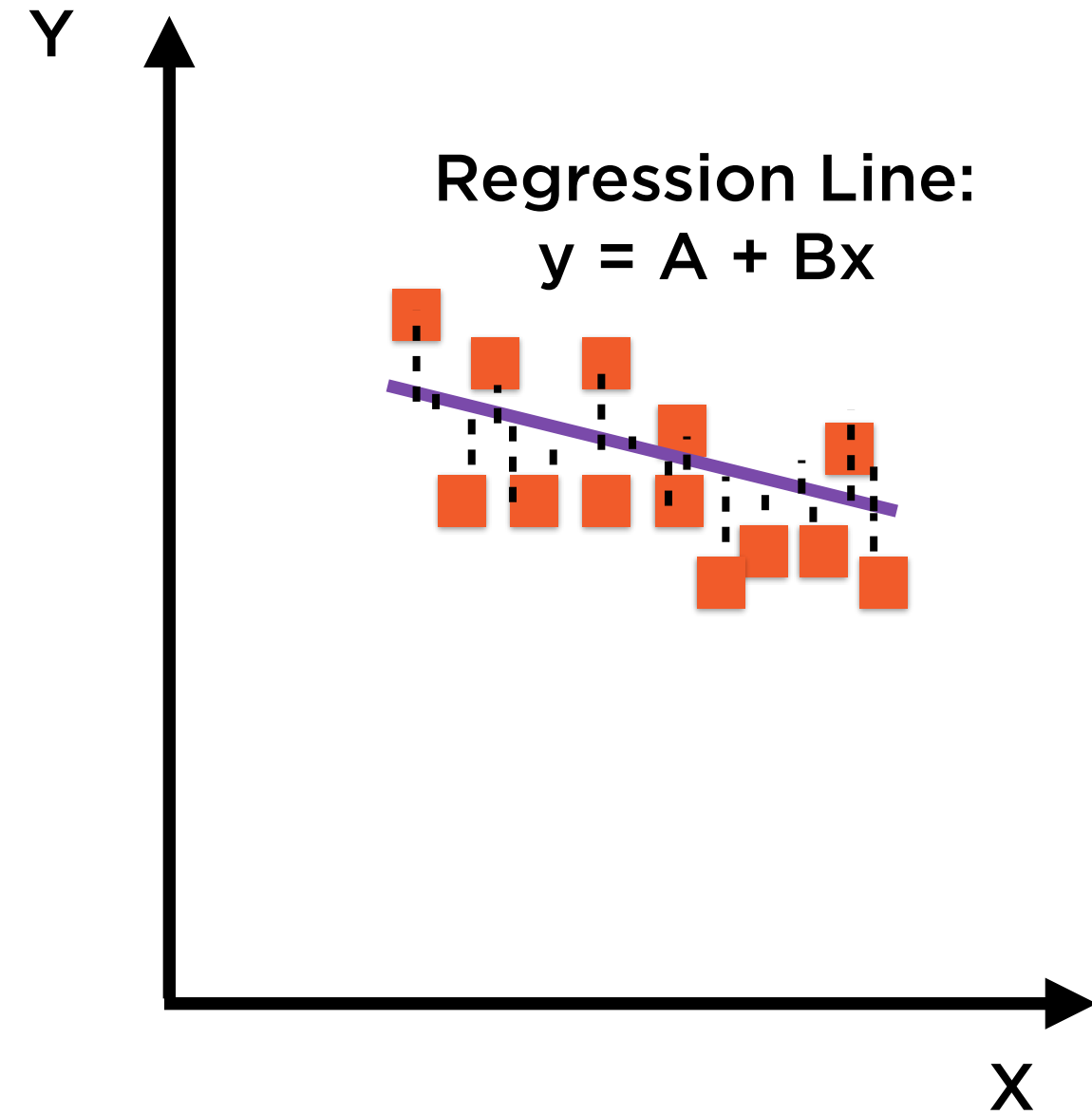
The “best fit” line is the one where the sum of the squares of the lengths of the errors is minimum



# Minimising Least Square Error



**Residuals** of a regression are the difference between actual and fitted values of the dependent variable



**Ideally, residuals should**

- have zero mean
- common variance
- be independent of each other
- be independent of  $x$
- be normally distributed

# Linear Regression as an Optimization Problem

---

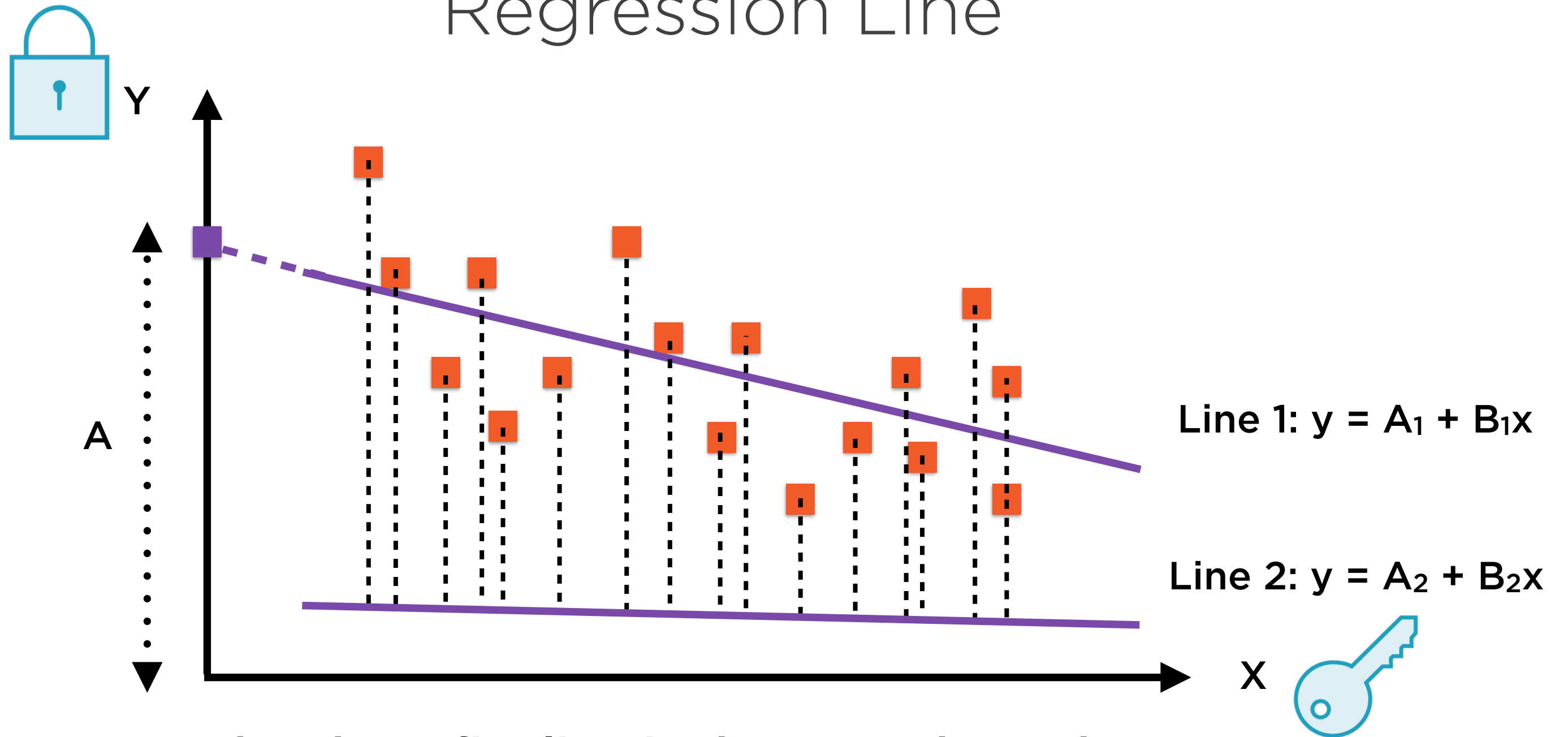
$$y = A + Bx$$

---

## Regression Line

The “best fit” line which minimizes the sum of the squares of the errors

# Regression Line



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

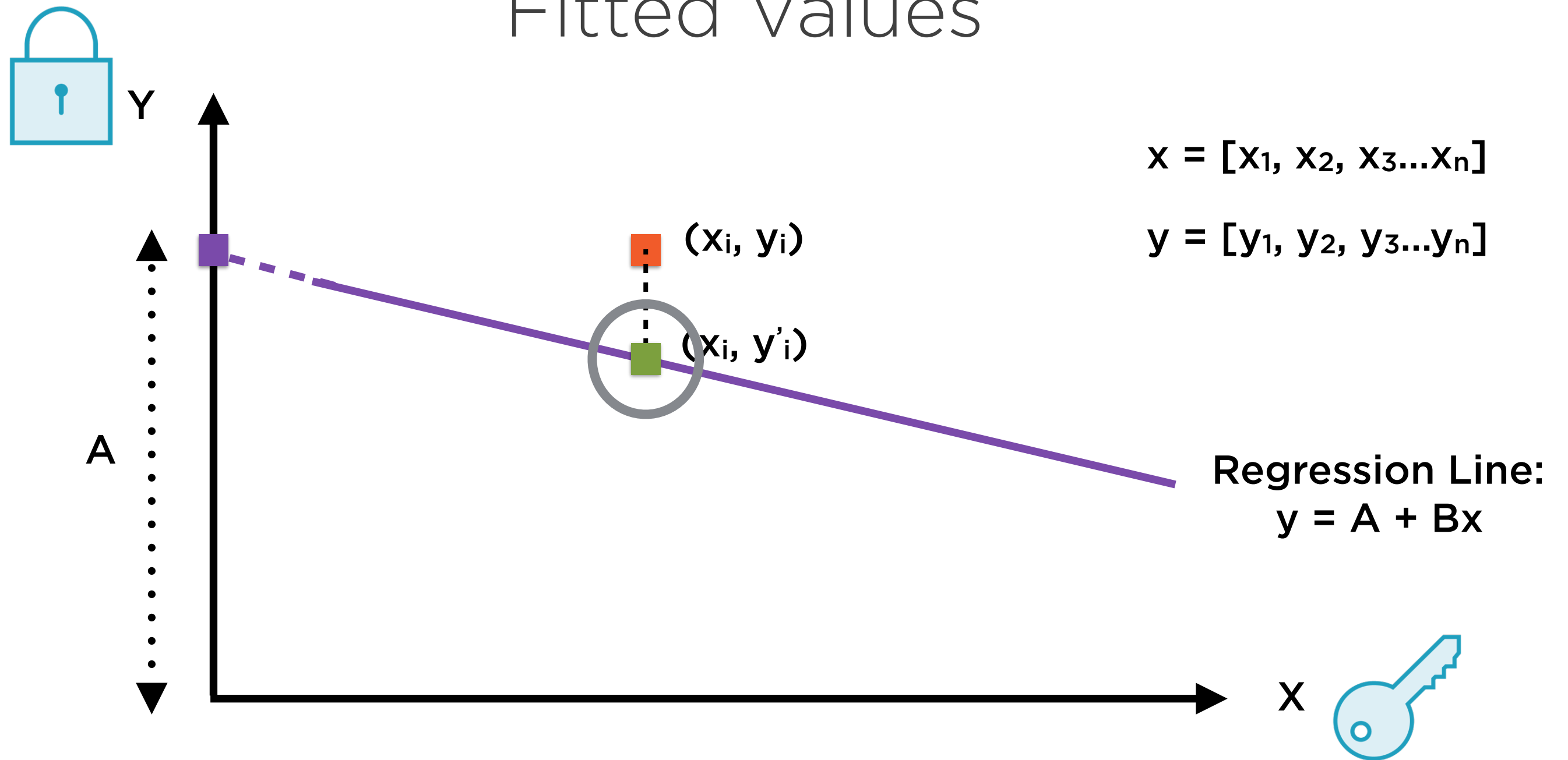
$$y' = A + Bx$$

---

## Fitted Values of Dependent Variable

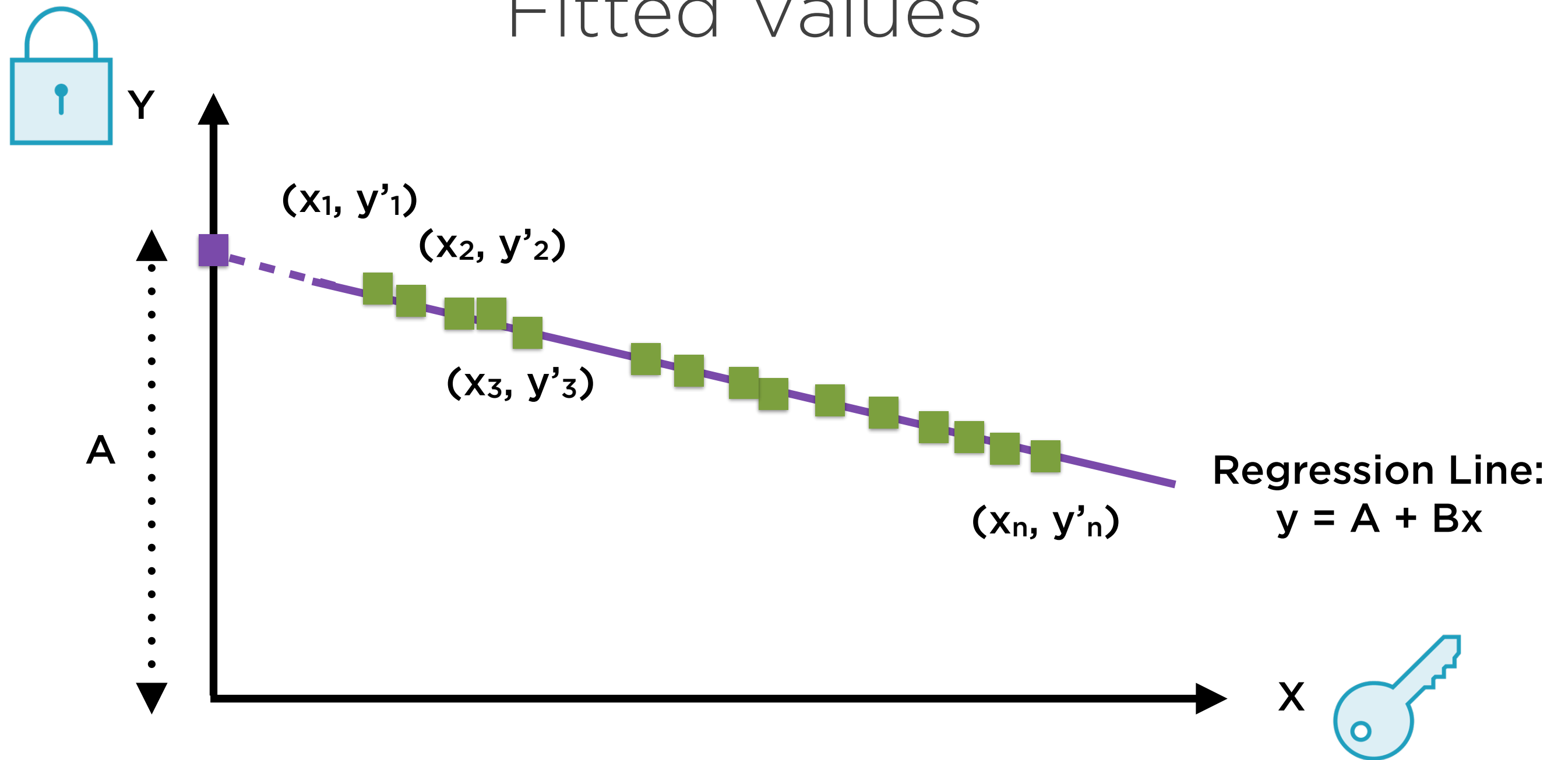
**The fitted line  $y = A + Bx$  will yield a different set of values, called the fitted values**

# Fitted Values



Each point  $(x_i, y_i)$  has a corresponding point  $(x_i, y'_i)$  on the regression line

# Fitted Values



The corresponding values of  $y'_i$  are called the fitted values



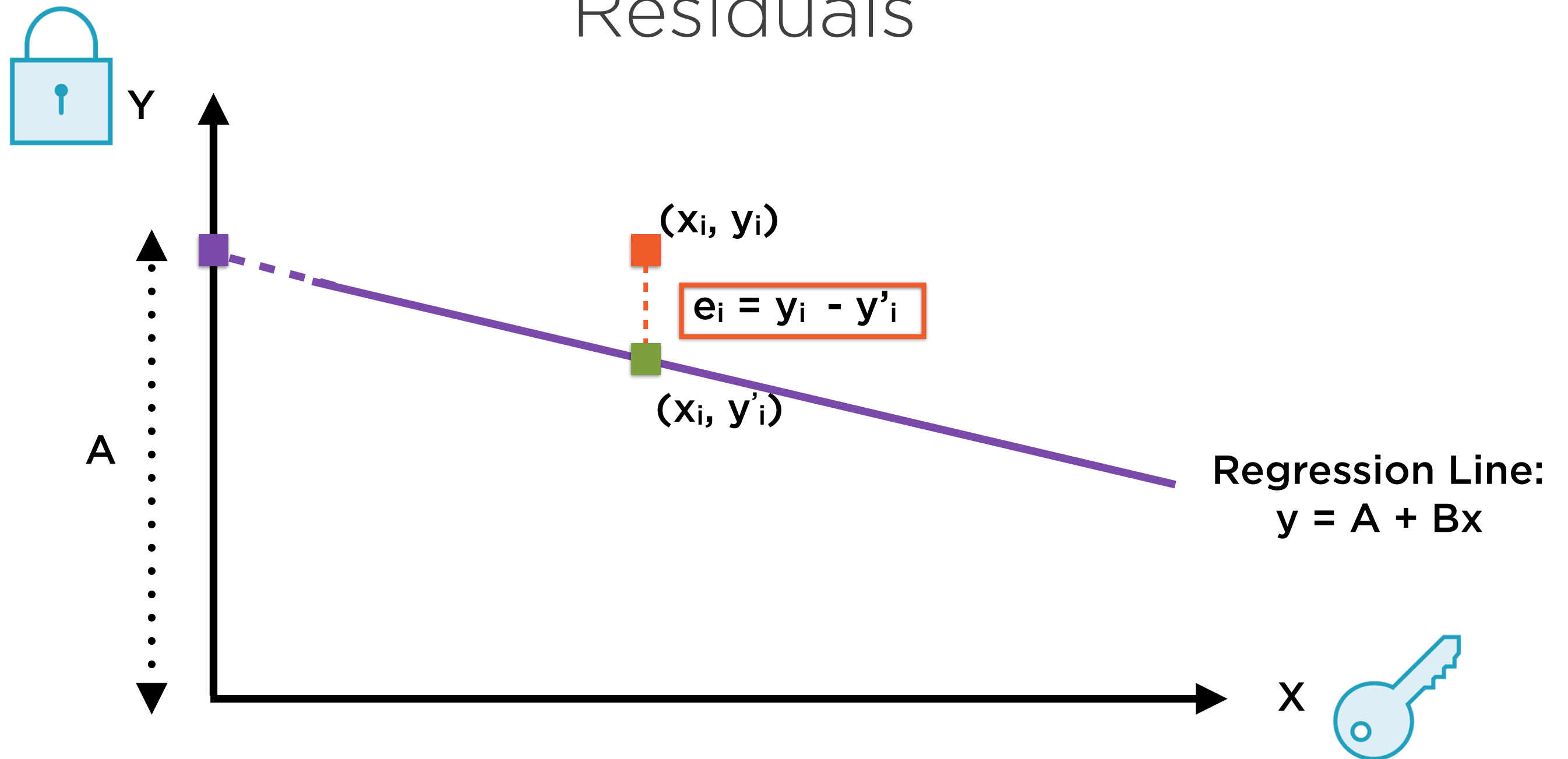
$$e = y - y'$$

---

## Residuals

**The residuals, or errors, are the differences between the actual and fitted values of the dependent variable**

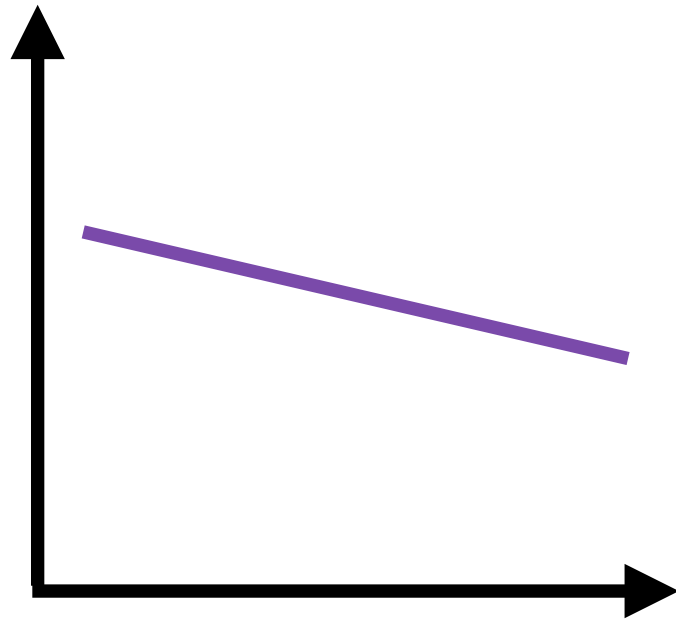
# Residuals



Residuals of a regression are the difference between actual and fitted values of the dependent variable

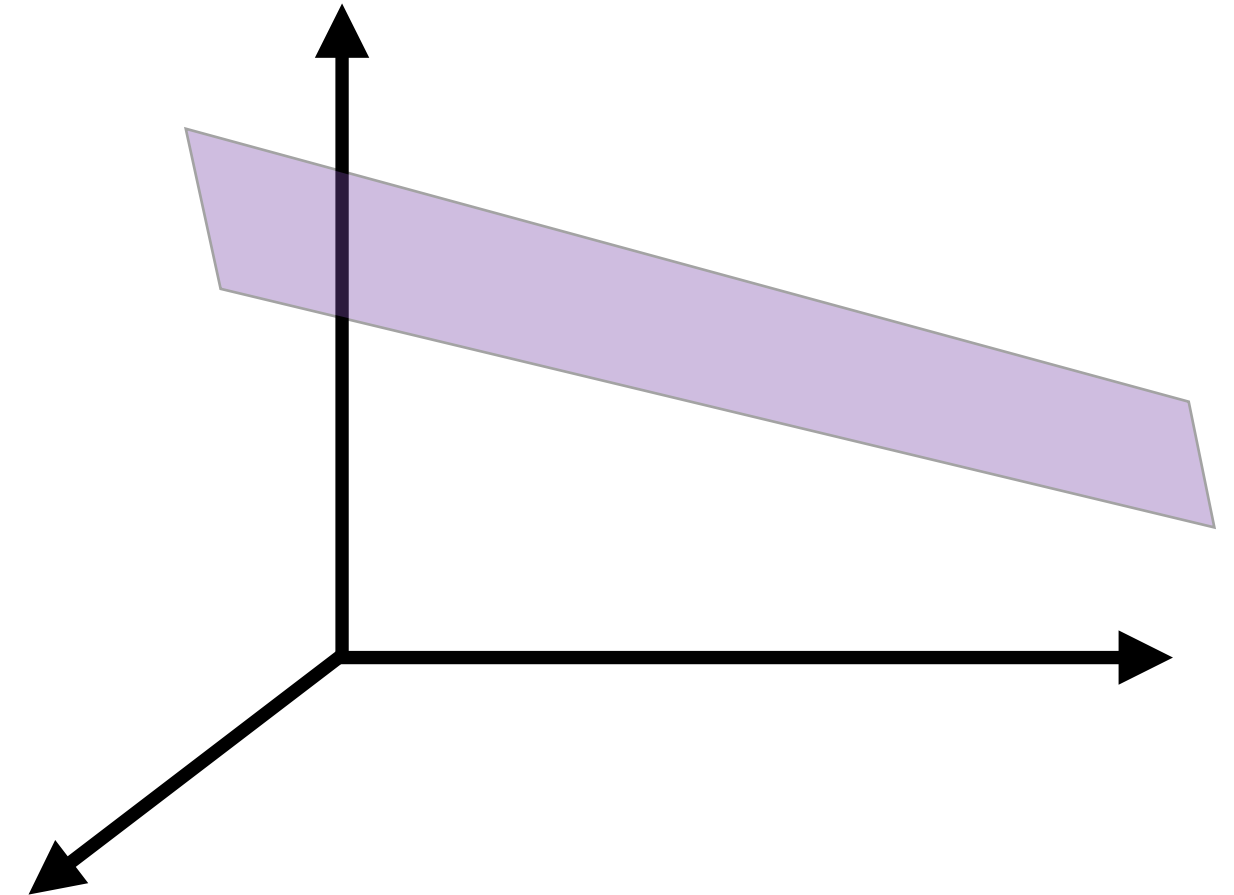
The regression line is that line which  
minimizes the variance of the residuals  
(MSE)

# MSE Minimization Extends To Multiple Regression



**Simple Regression**

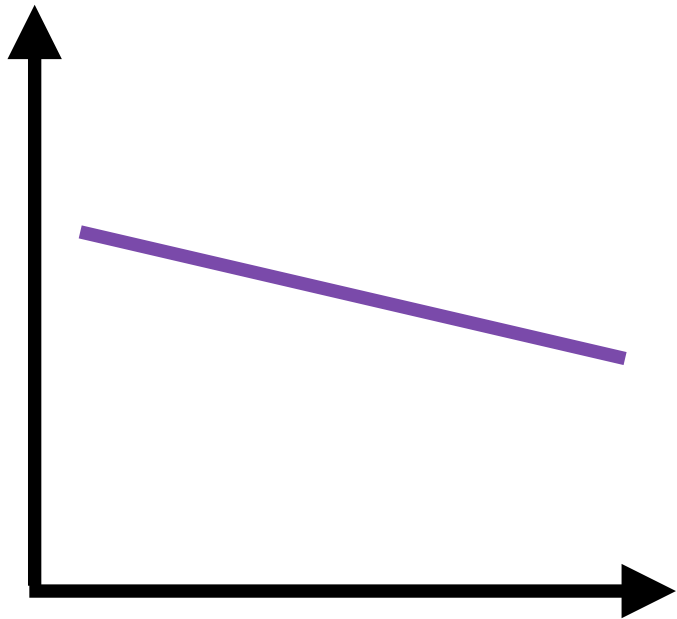
Data in 2 dimensions



**Multiple Regression**

Data in  $> 2$  dimensions

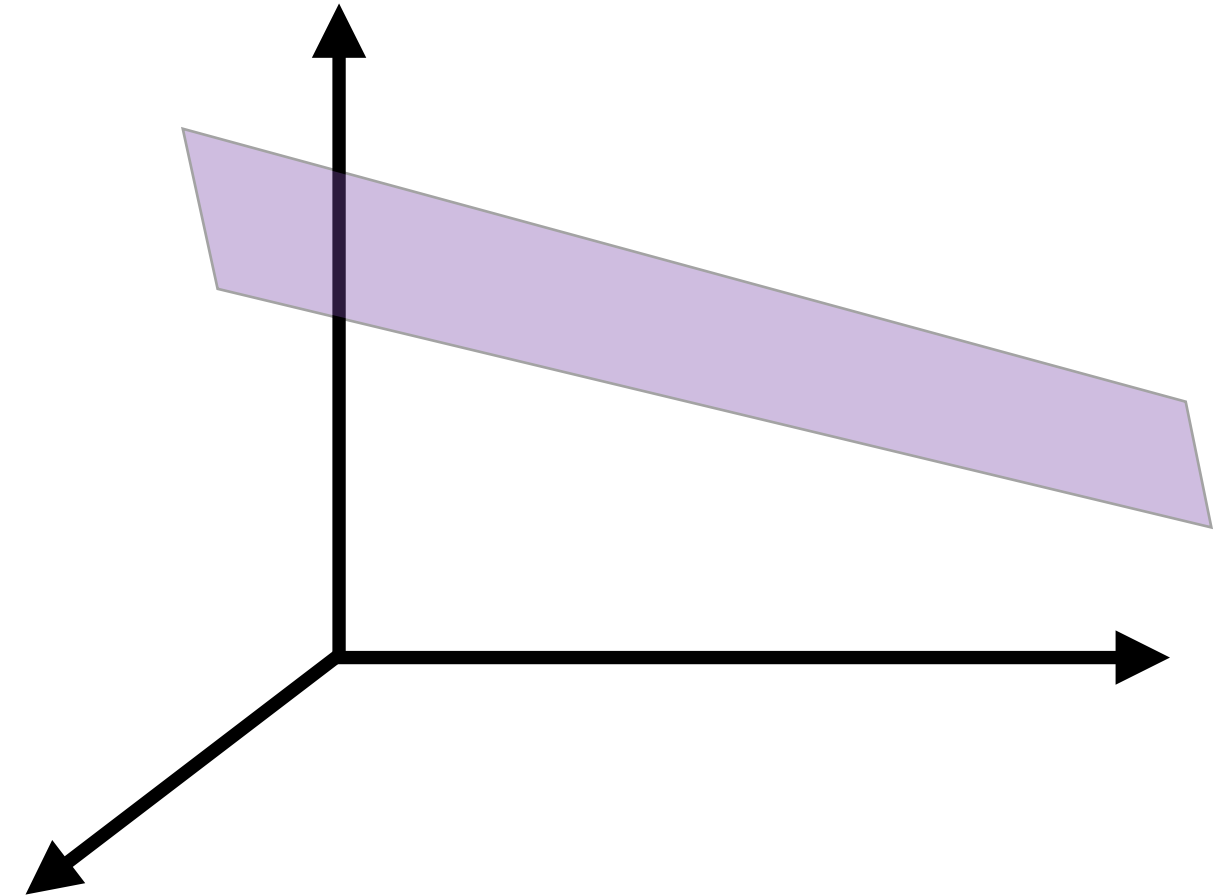
# Simple and Multiple Regression



**Simple Regression**

One independent variable

$$y = A + Bx$$



**Multiple Regression**

Multiple independent variables

$$y = A + B_1x_1 + B_2x_2 + B_3x_3$$

# “Best Linear Unbiased Estimator” (BLUE)

## “Best”

Coefficients have minimum variance, i.e. are estimated with relatively high certainty

## “Unbiased”

Residuals have zero mean, are uncorrelated to each other and have equal variance

Solving the regression problem with the method of **least squares** gives a **BLUE** solution

$$R^2 = ESS / TSS$$

---

$R^2$

$$R^2 = \text{Explained Sum of Squares} / \text{Total Sum of Squares}$$

---

$R^2$

**ESS - Variance of fitted values**

**TSS - Variance of actual values**



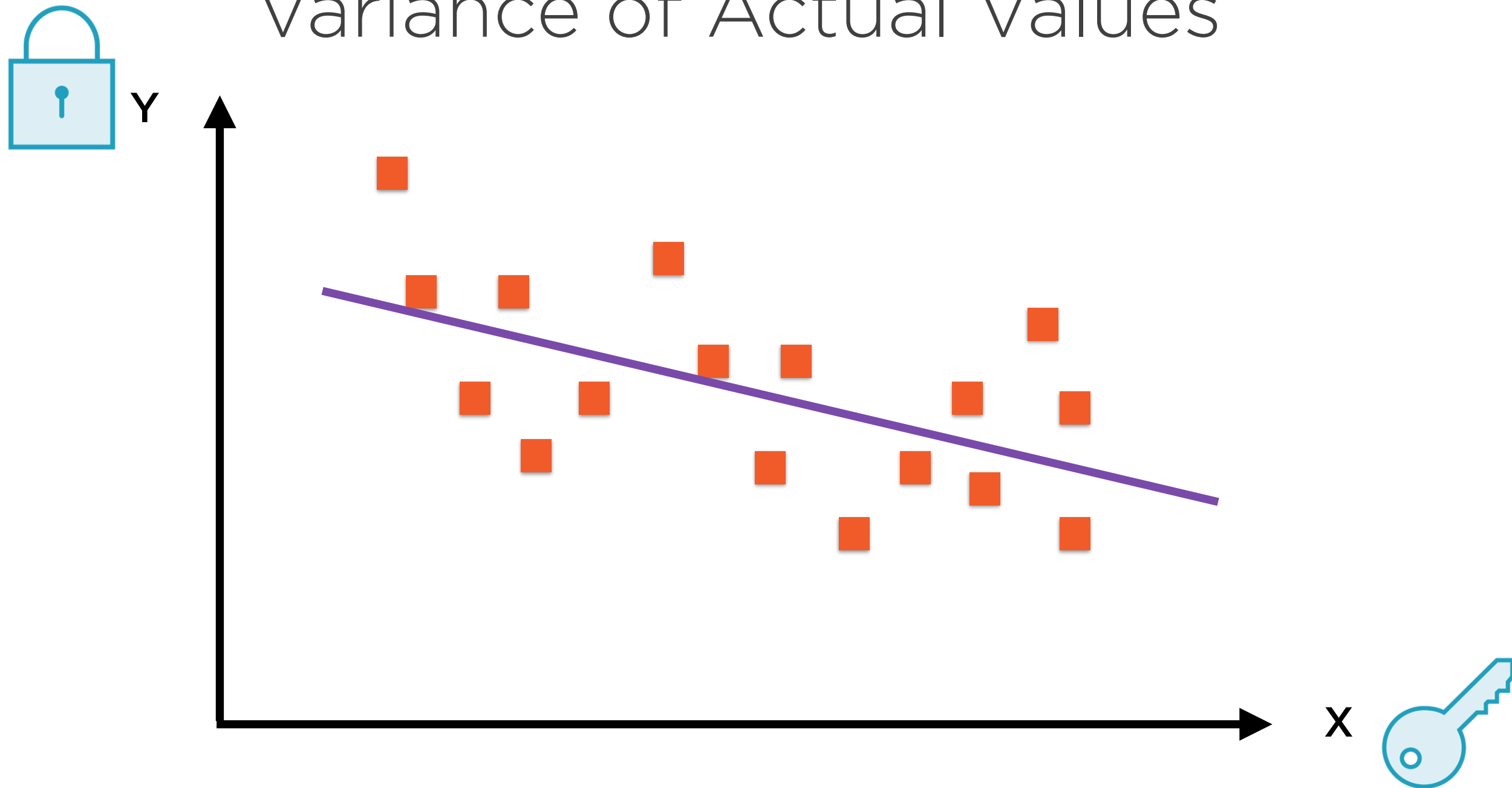
$$R^2 = \text{Explained Sum of Squares} / \text{Total Sum of Squares}$$

---

$R^2$

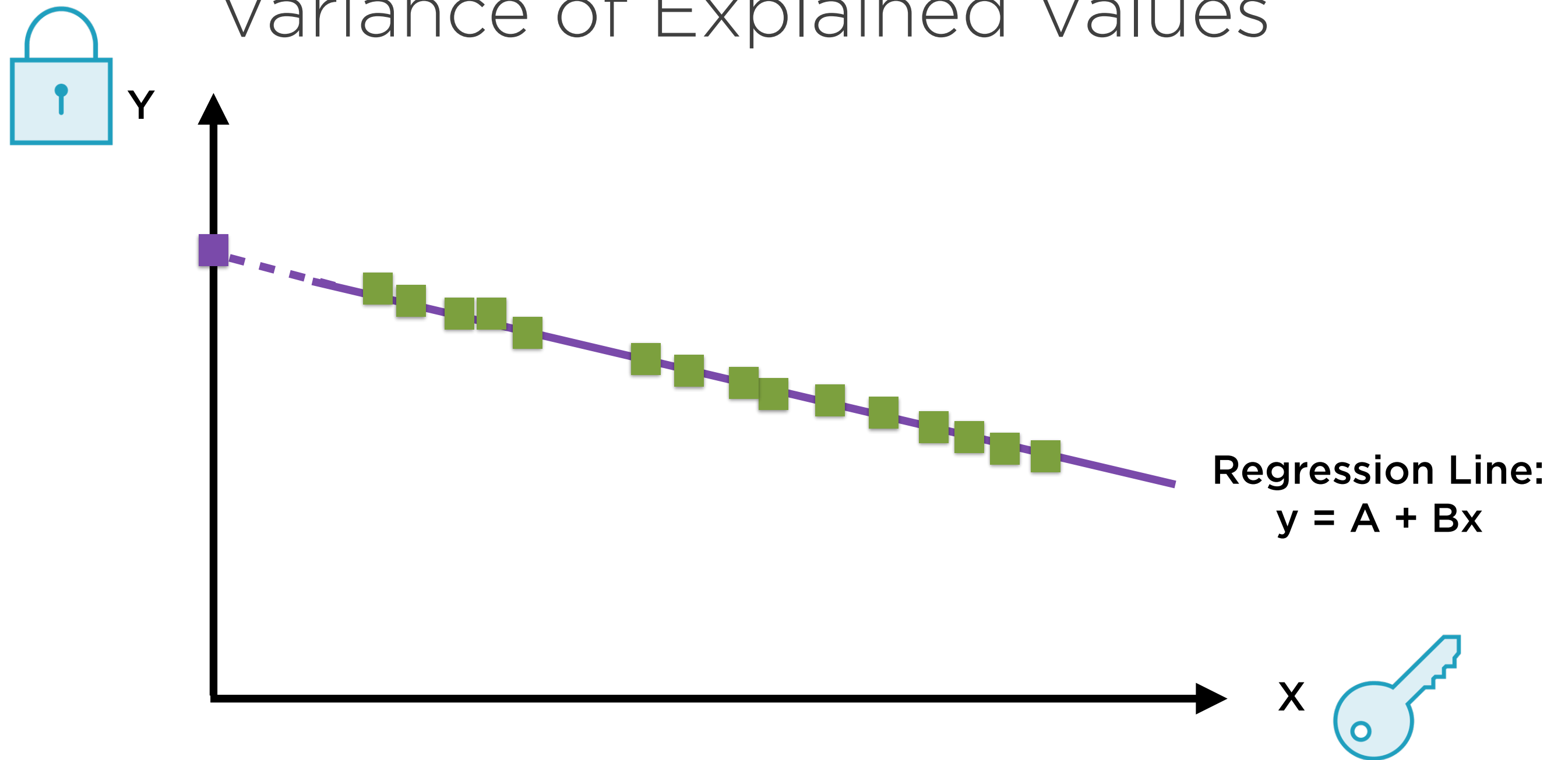
The percentage of total variance explained by the regression. Usually, the higher the  $R^2$ , the better the quality of the regression (upper bound is 100%)

# Variance of Actual Values



The original data points have some variance (TSS)

# Variance of Explained Values



The fitted data points have their own variance (ESS)

$$R^2 = ESS / TSS$$

---

$R^2$

**How much of the original variance is captured in the fitted values?**

**Generally, higher this number the better the regression**

The regression line found by  
minimizing variance of residuals (MSE)  
is the line with the **best  $R^2$**

# Other Types of Regression

Other forms of regression alter the objective function of the optimization

Ridge

Lasso

Elastic Net

SVR

Demo

**Implementing linear regression in  
scikit-learn**

# Distance Measures

---



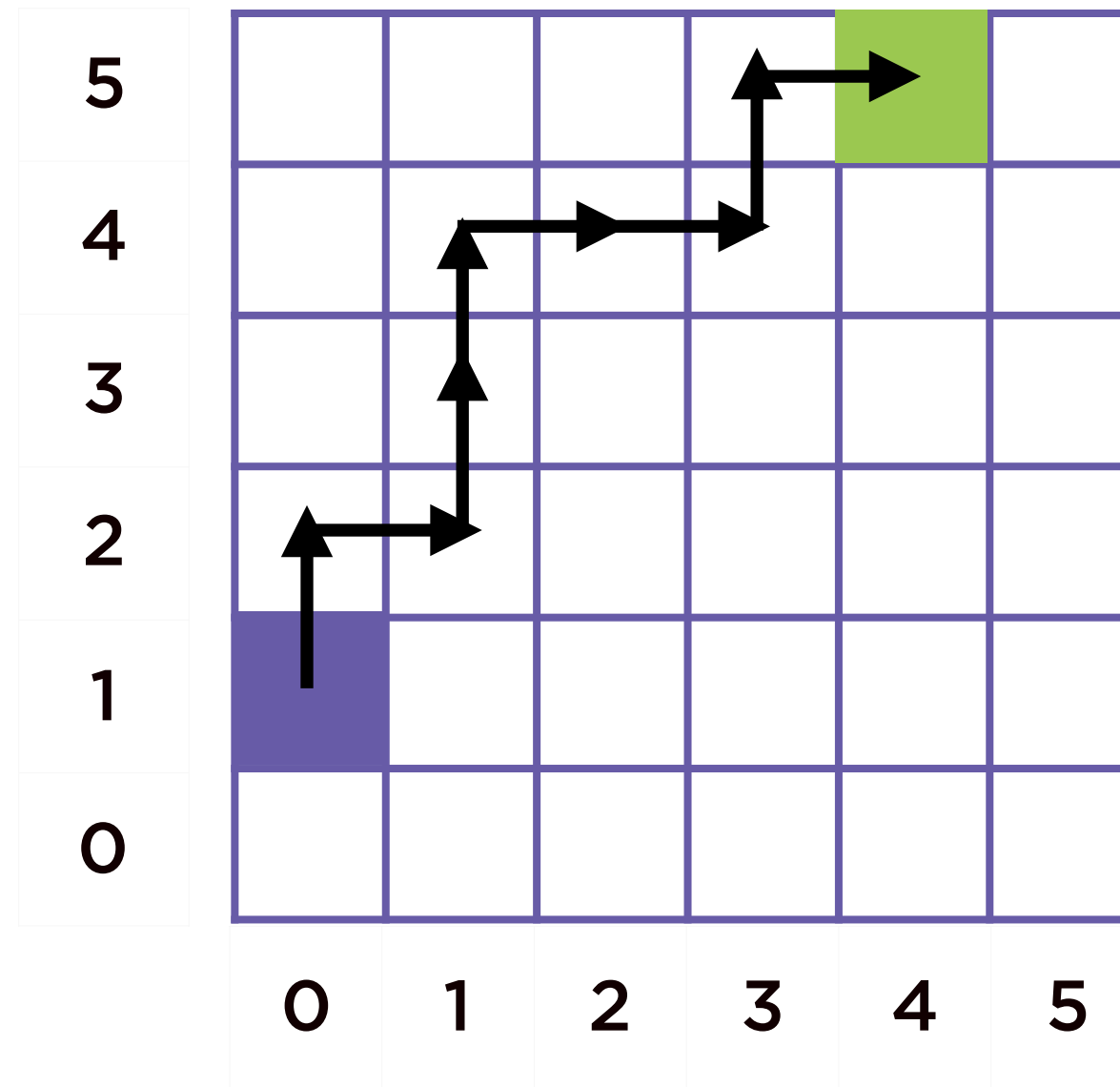
# Distance Measure

**L1 distance**

**Snake distance**

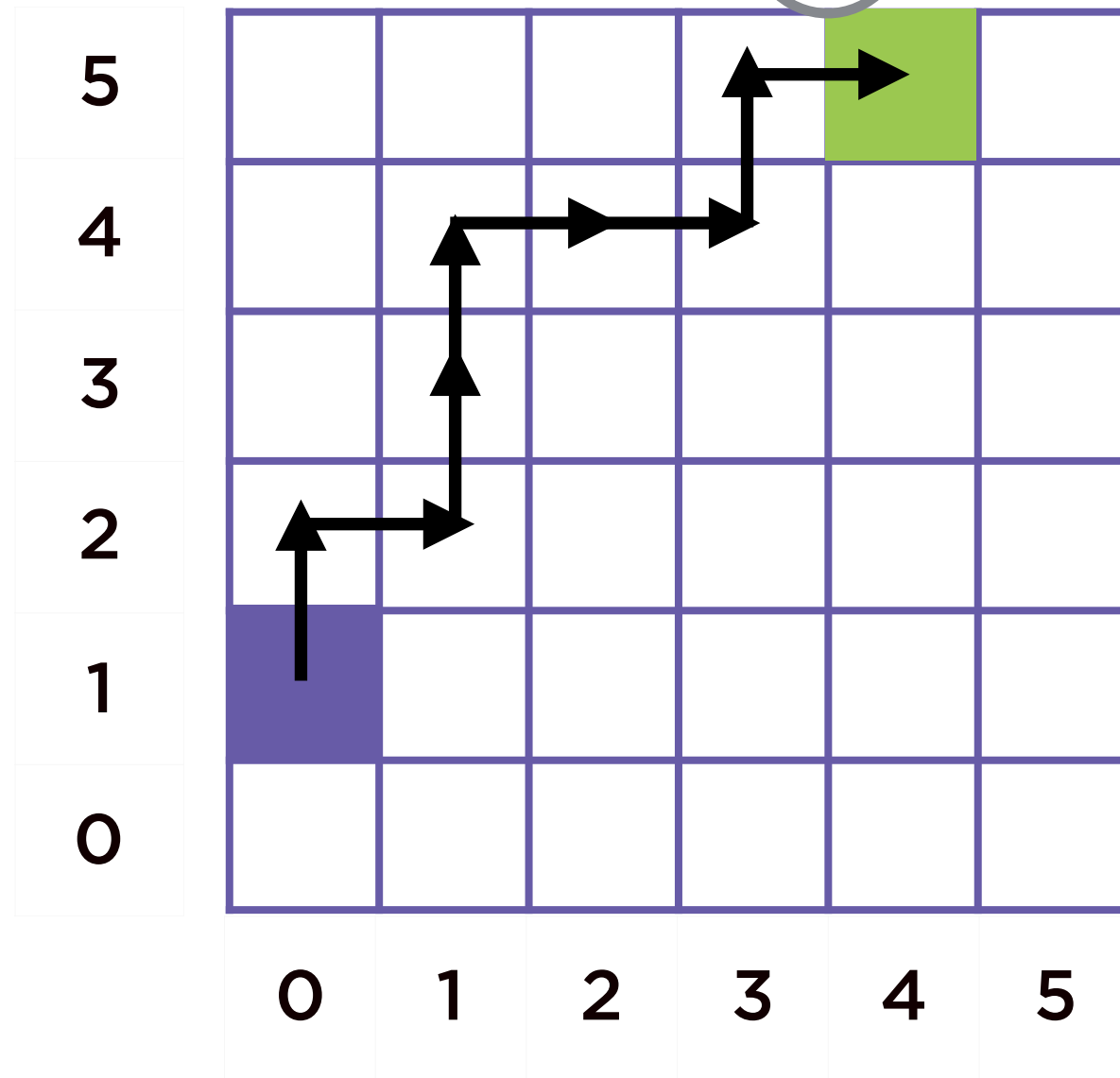
**City block distance**

**Manhattan distance**



# L1 Distance

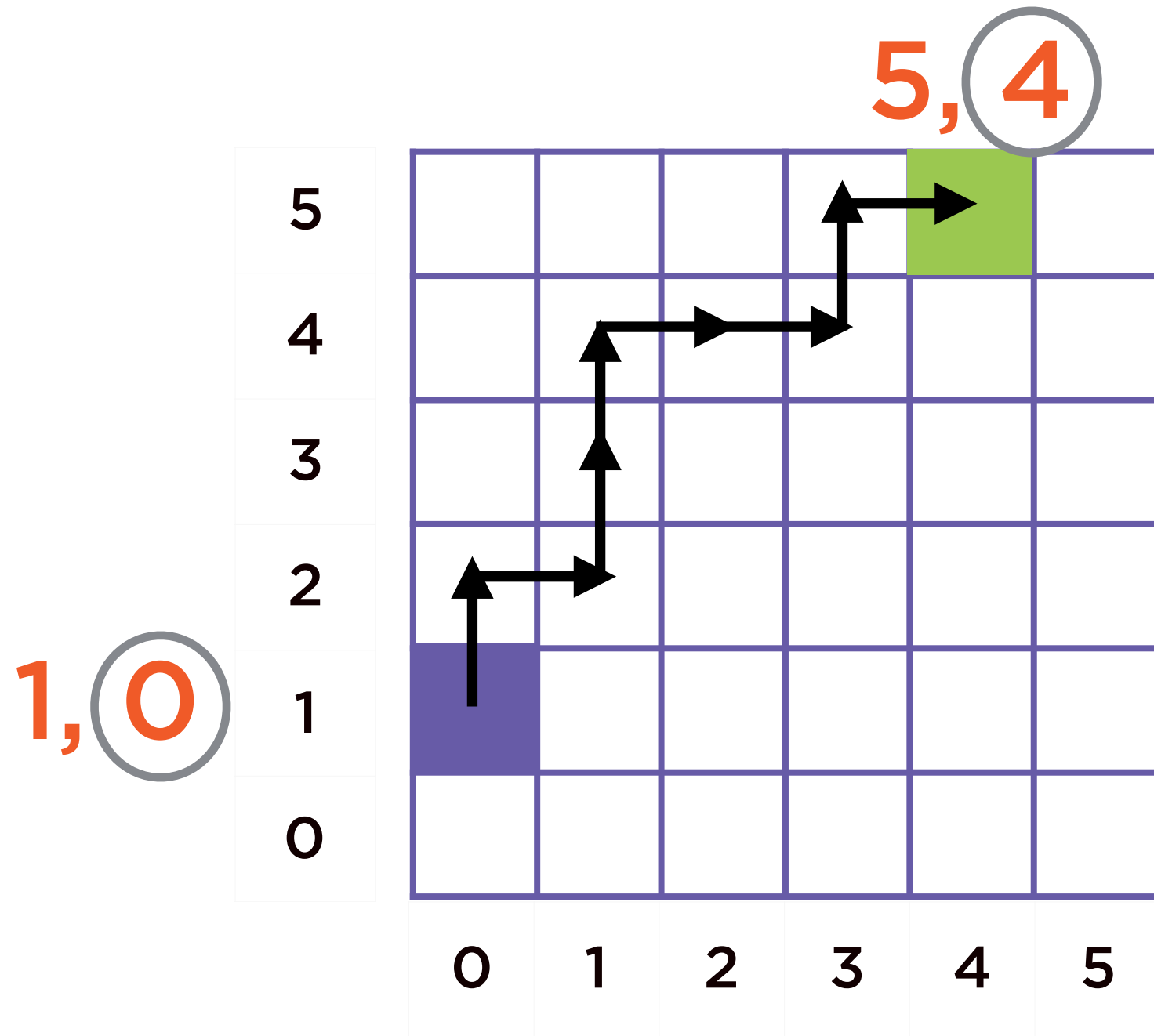
**1, 0**



**5, 4**

$$5 - 1 = 4$$

# L1 Distance

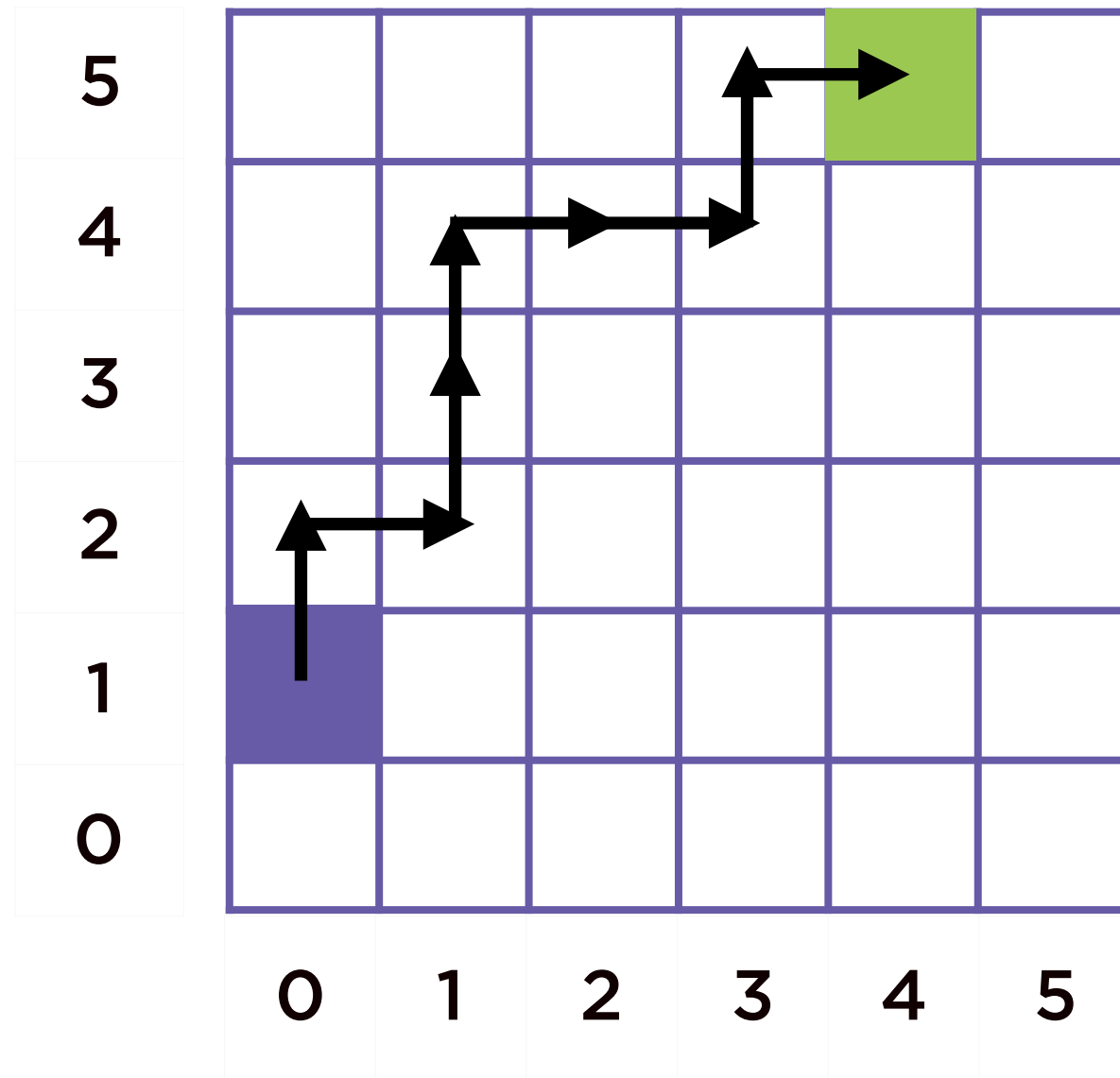


$$5 - 1 = 4$$

$$4 - 0 = 4$$

# L1 Distance

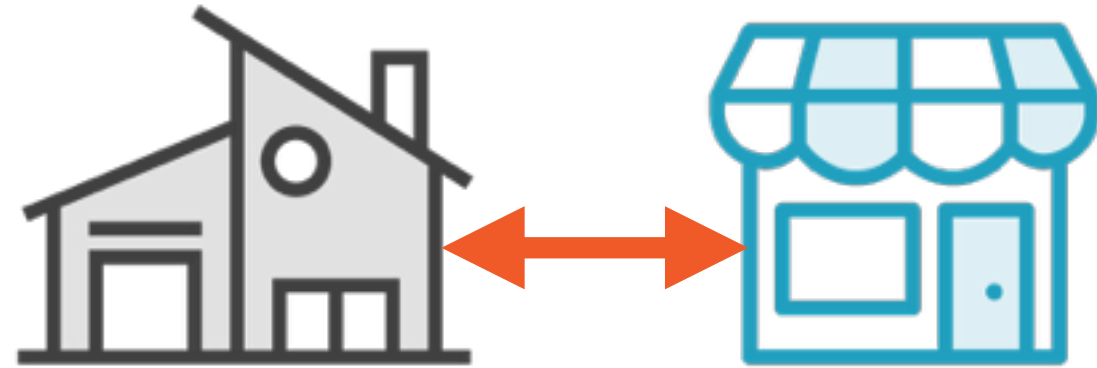
1, 0



5, 4

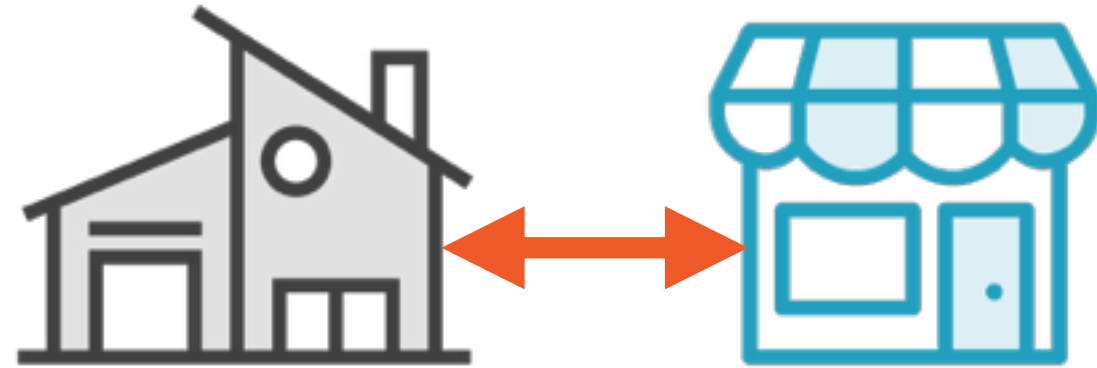
$$\begin{aligned} 5-1 &= 4 \\ 4-0 &= 4 \\ &= 8 \end{aligned}$$

# Distance Measures



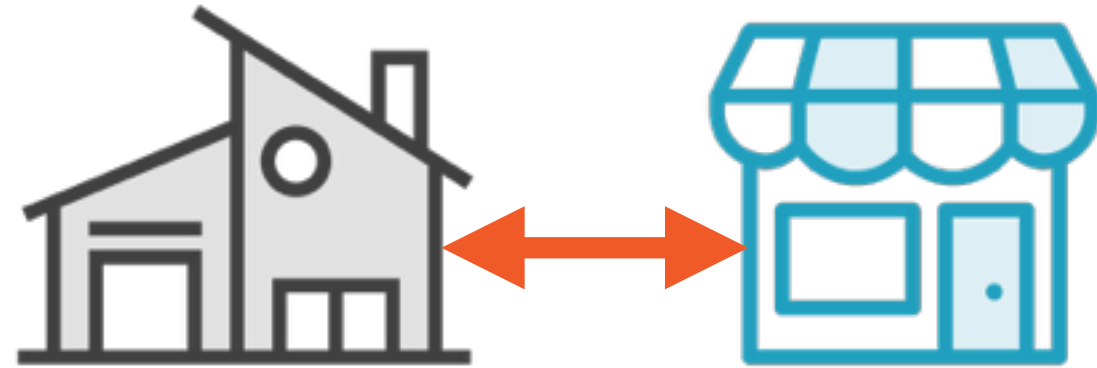
$$\text{L1 Distance}(t1, t2) = \text{sum}(\text{abs}(t1_i - t2_i))$$

# Distance Measures



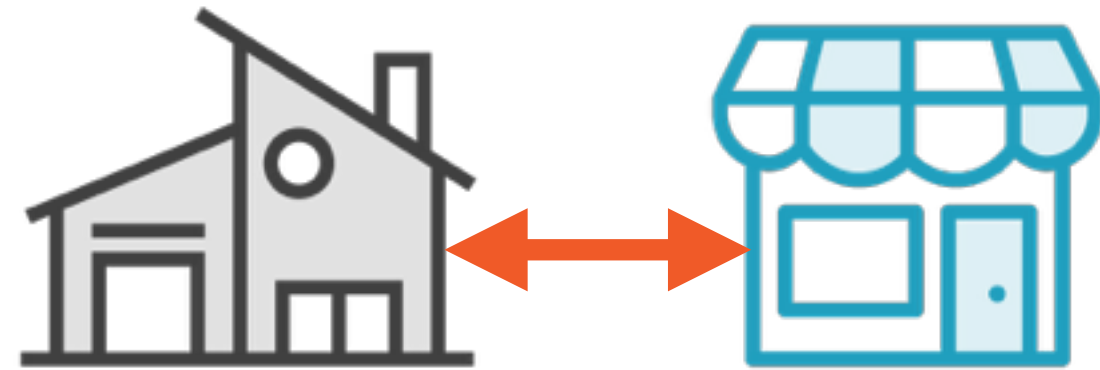
$$\text{L1 Distance}(t1, t2) = \text{sum}(\text{abs}(t1_i - t2_i))$$

# Distance Measures



$$\text{L1 Distance}(t1, t2) = \text{sum}(\text{abs}(t1_i - t2_i))$$

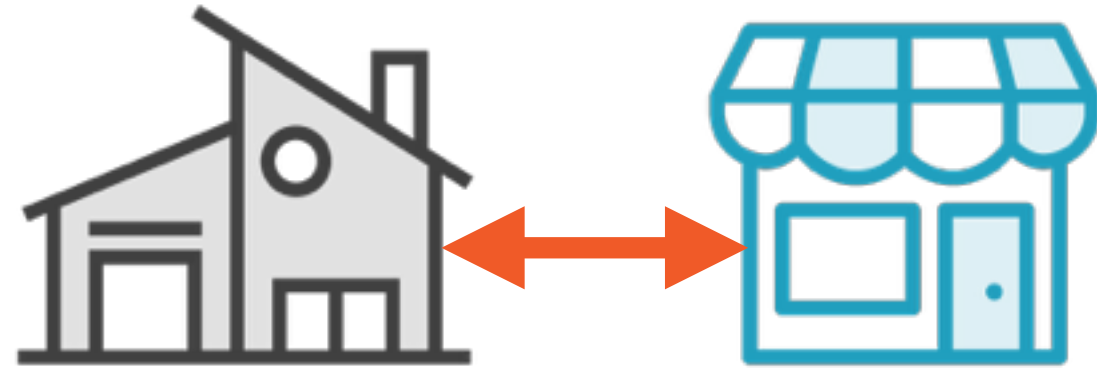
# Distance Measures



$$\text{L1 Distance}(t1, t2) = \text{sum}(\text{abs}(t1_i - t2_i))$$

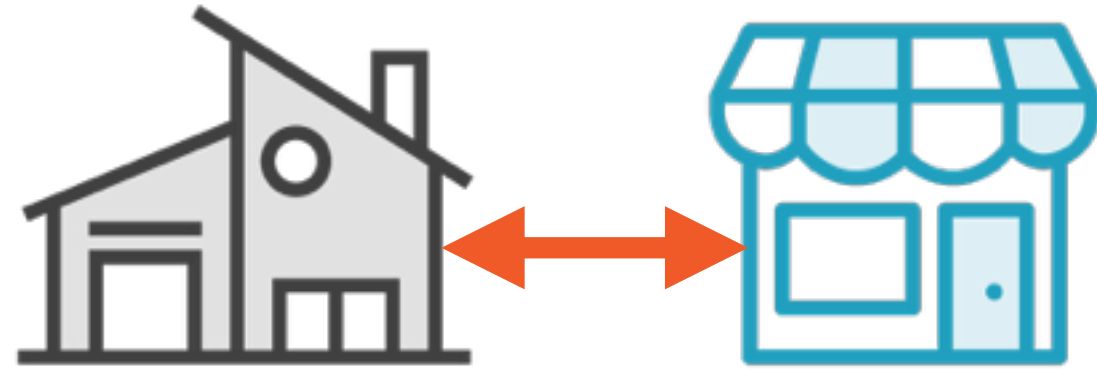


# Distance Measures



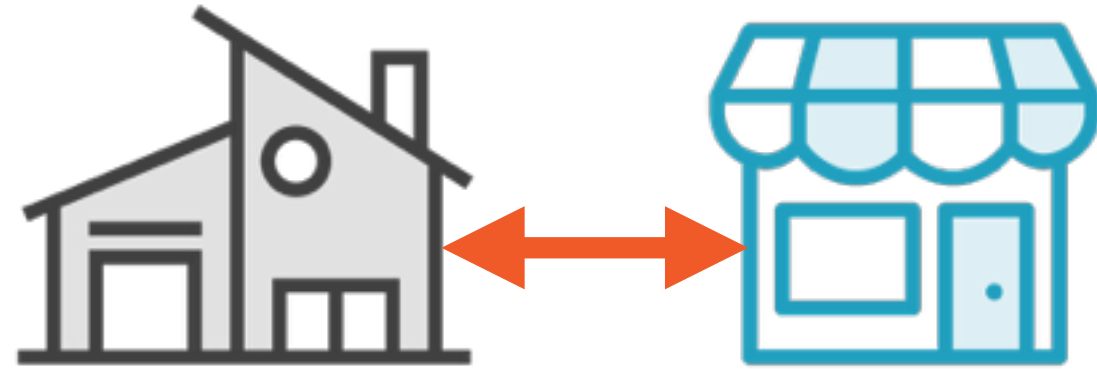
$$\text{L1 Distance}(t1, t2) = \text{sum}(\text{abs}(t1_i - t2_i))$$

# L-1 Norm



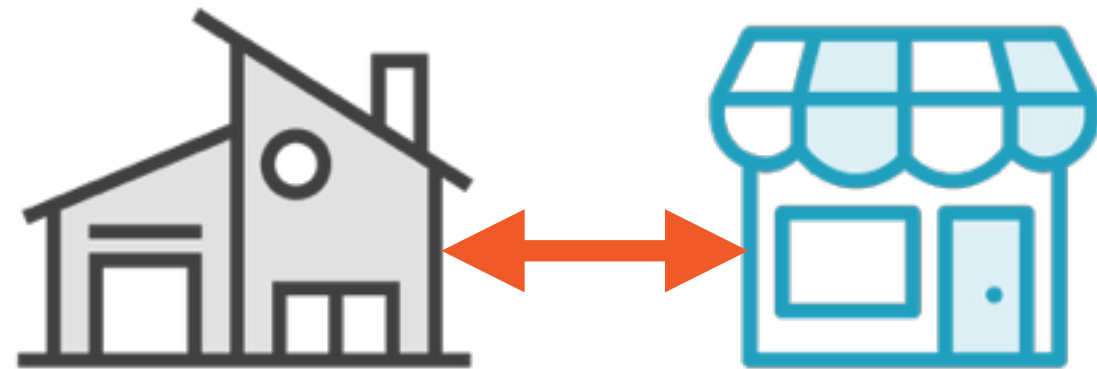
$$\text{L1-Norm}(A, B_1, B_2 \dots B_n) = |A| + |B_1| + |B_2| \dots + |B_n|$$

# L-1 Norm



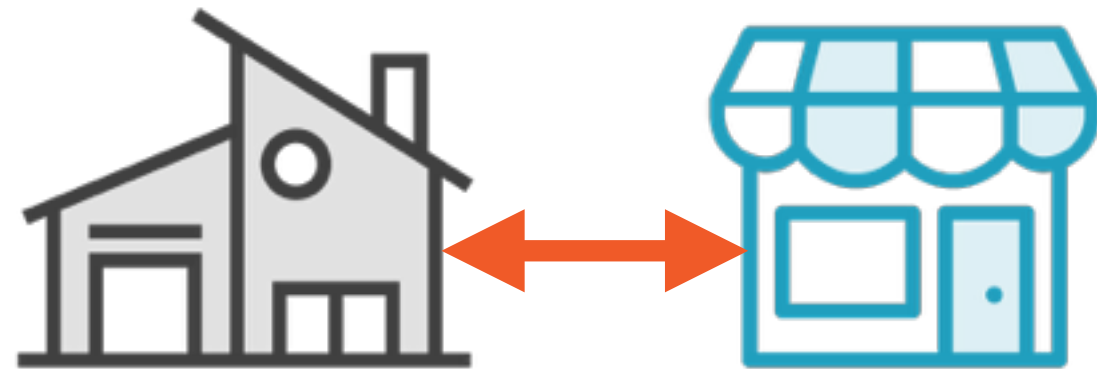
$$L^1\text{-Norm}(A, B_1, B_2 \dots B_n) = |A|^1 + |B_1|^1 + |B_2|^1 \dots + |B_n|^1$$

# L-2 Norm



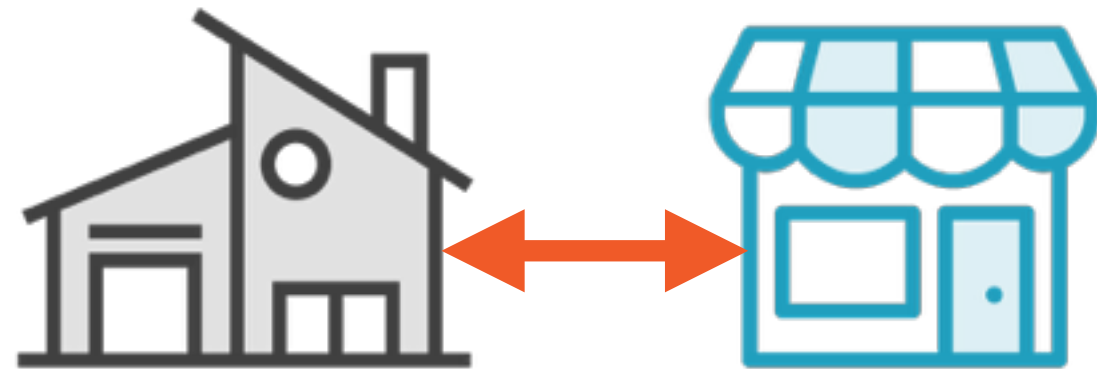
$$L^2\text{-Norm}(A, B_1, B_2 \dots B_n) = |A|^2 + |B_1|^2 + |B_2|^2 \dots + |B_n|^2$$

# L-2 Norm



$$\text{L2-Norm}(A, B_1, B_2 \dots B_n) = |A|^2 + |B_1|^2 + |B_2|^2 \dots + |B_n|^2$$

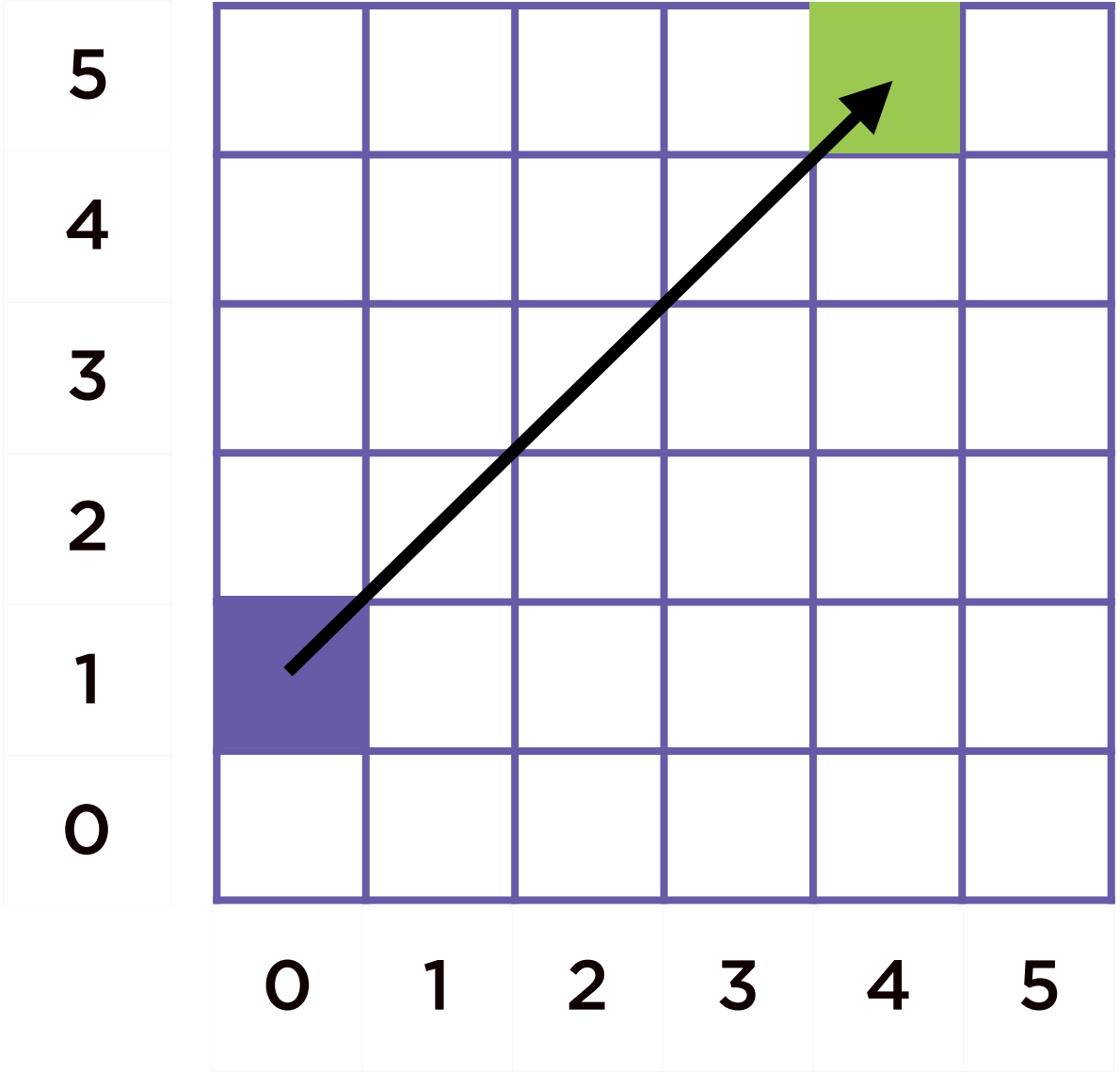
# L-2 Norm



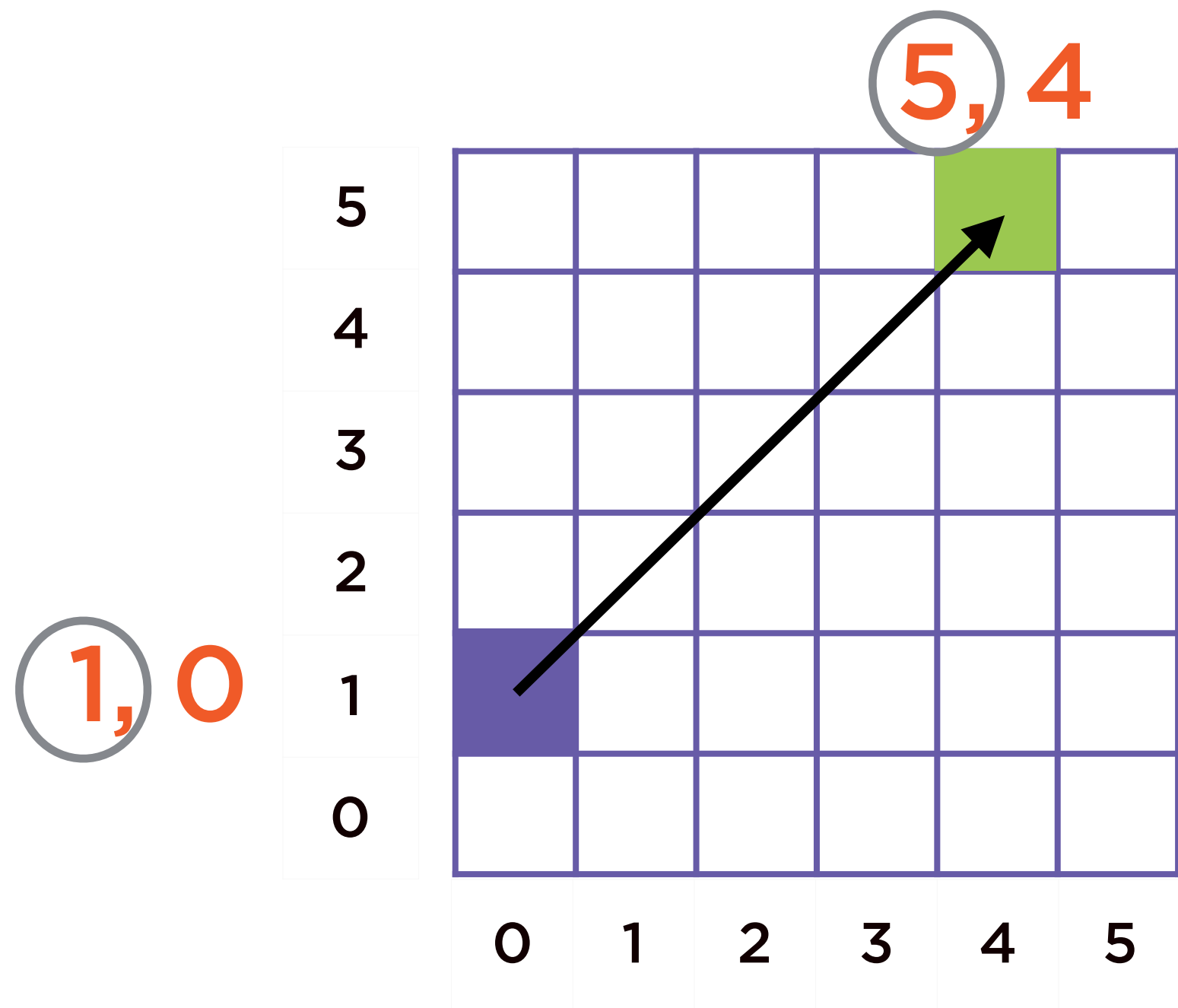
$$L2\text{-Norm}(A, B_1, B_2 \dots B_n) = A^2 + B_1^2 + B_2^2 \dots + B_n^2$$

**Euclidean Distance**  
**As the crow flies**

# Distance Measure



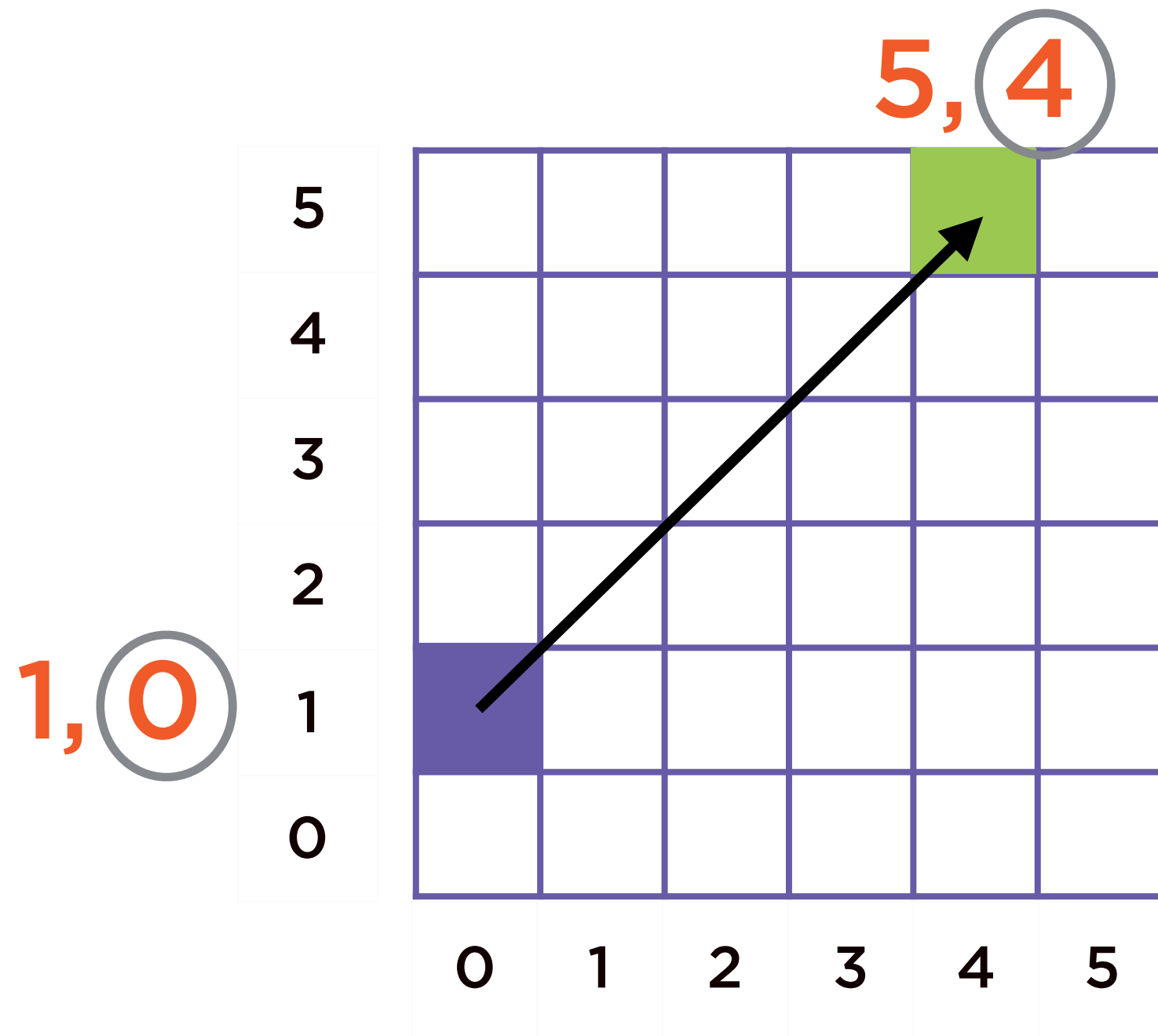
# L2 Distance



$$(5-1)^2 = 16$$



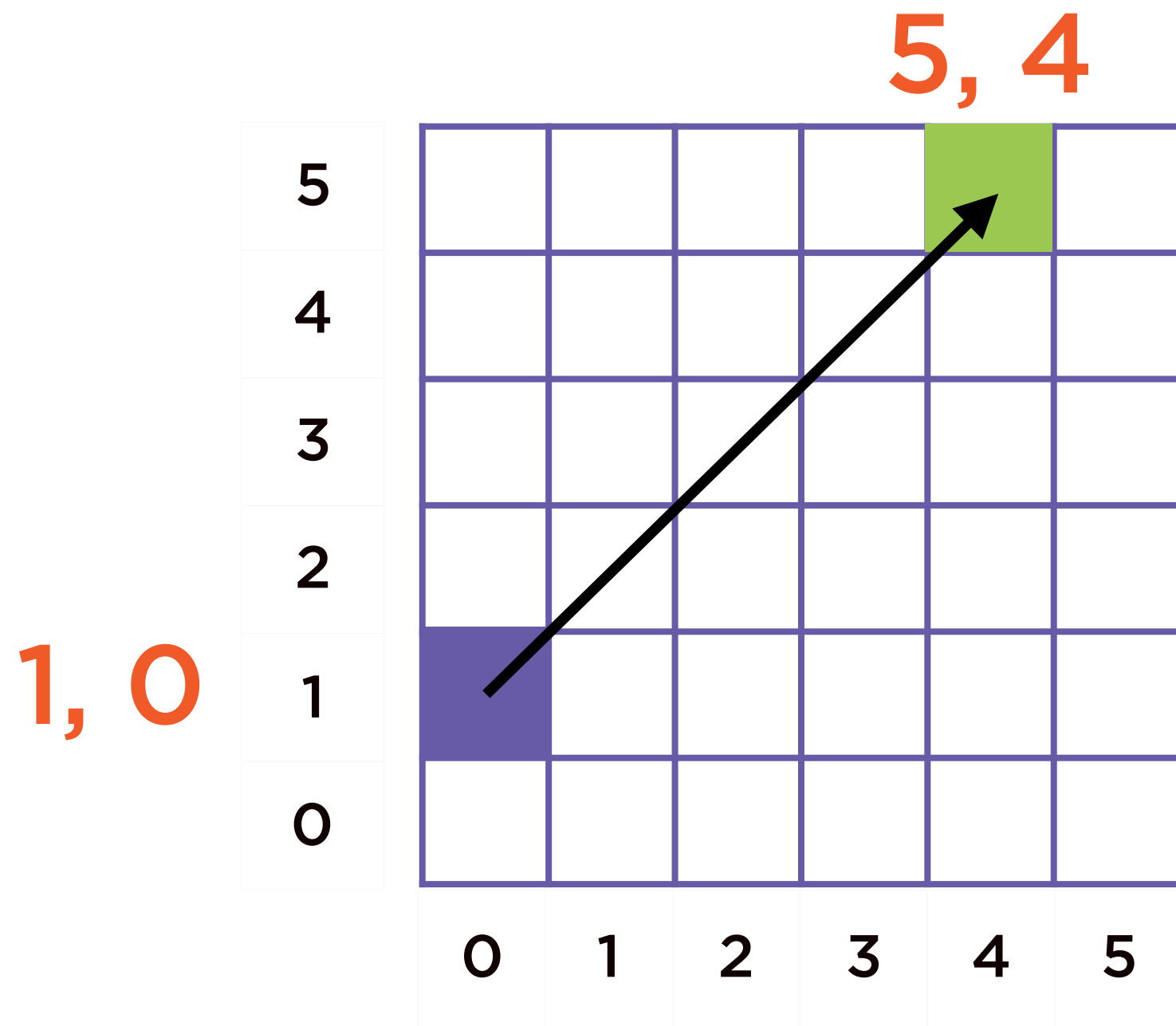
## L2 Distance



$$(5-1)^2 = 16$$

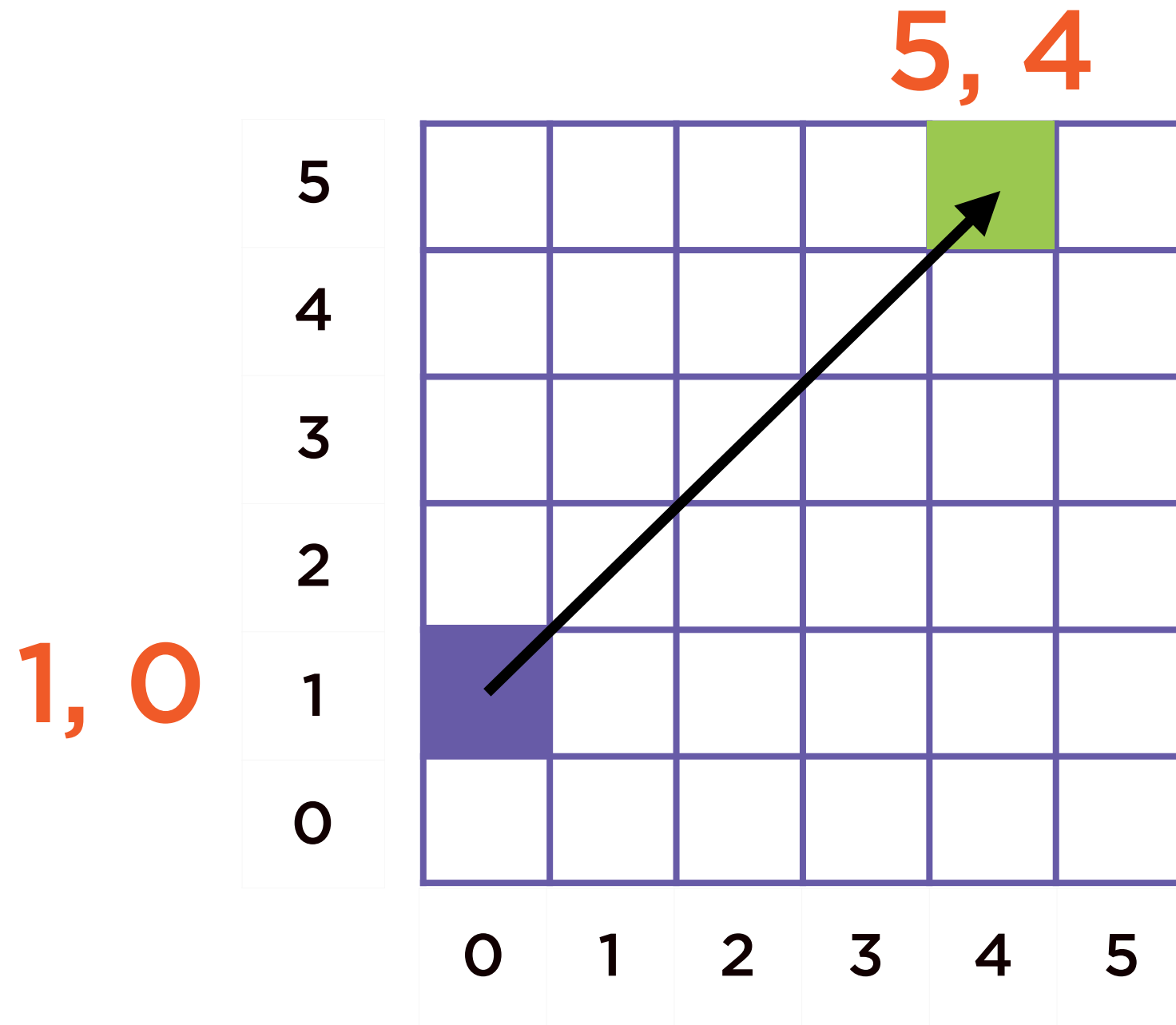
$$(4-0)^2 = 16$$

# L2 Distance



$$\begin{aligned}(5-1)^2 &= 16 \\ (4-0)^2 &= 16 \\ &= 32\end{aligned}$$

# L2 Distance



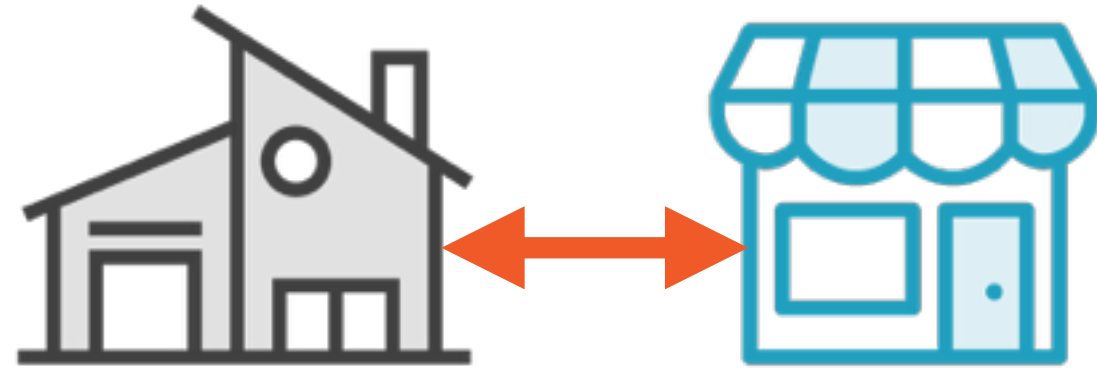
$$(5-1)^2 = 16$$

$$(4-0)^2 = 16$$

$$= 32$$

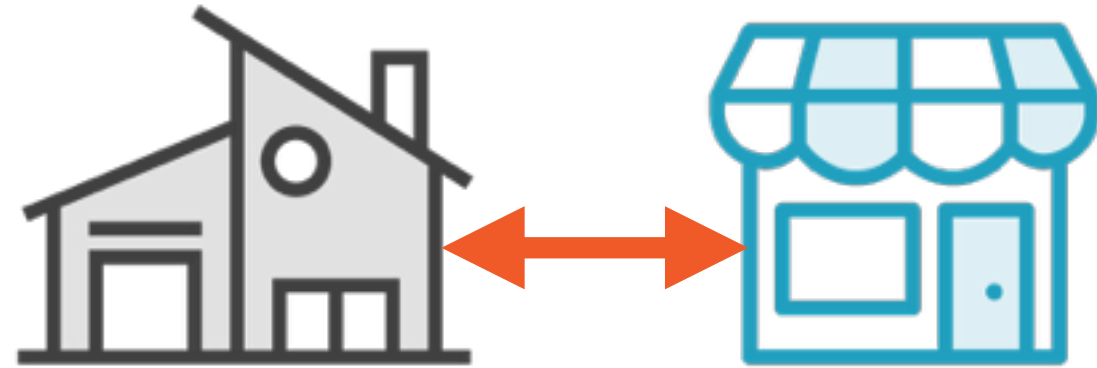
$$\text{sqrt}(32) = 5.65$$

# Distance Measures



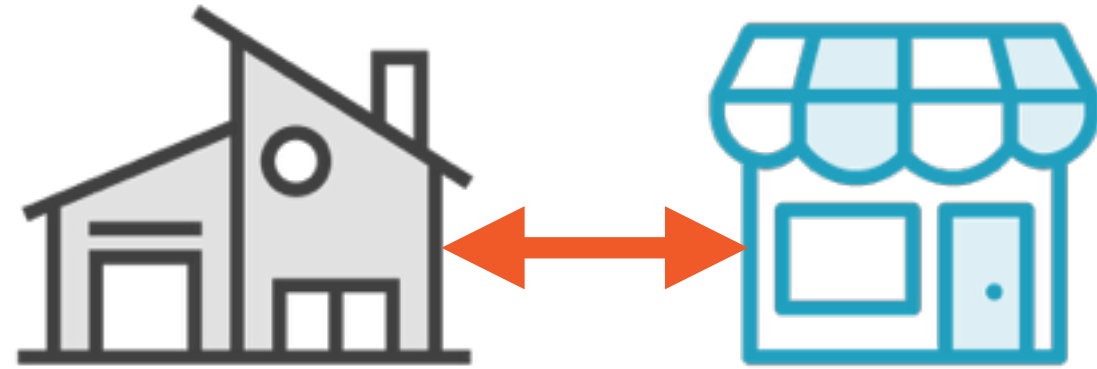
$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$

# Distance Measures



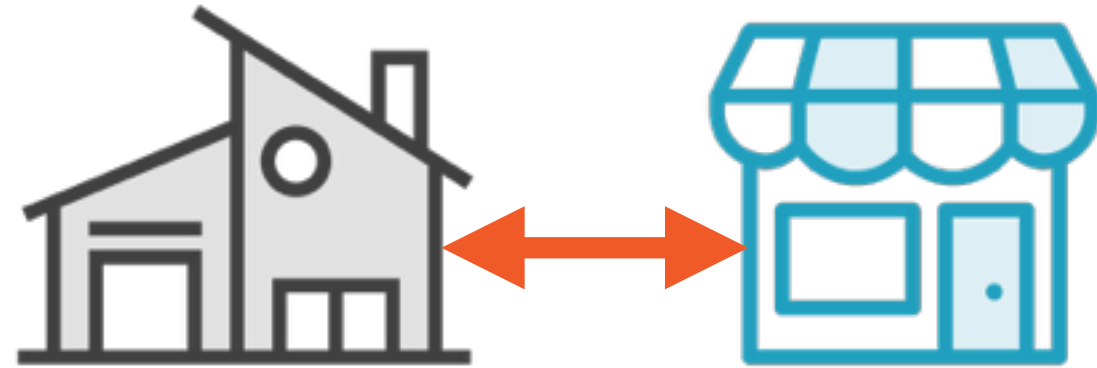
$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$

# Distance Measures



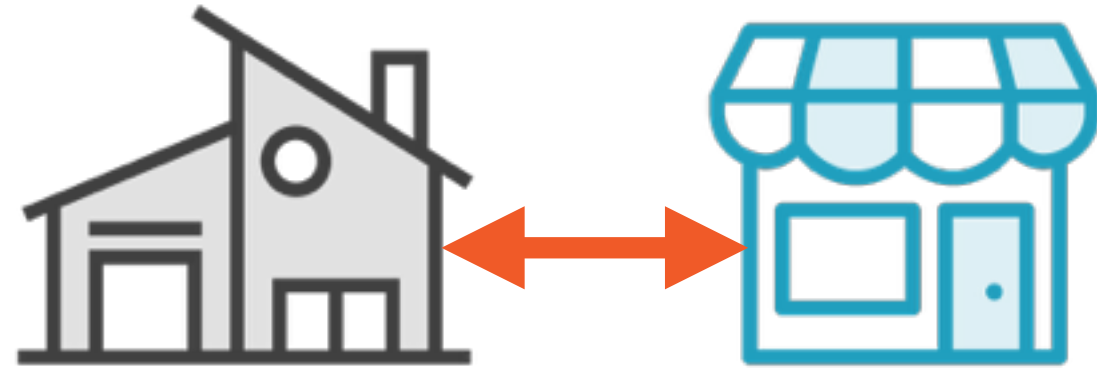
$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$

# Distance Measures



$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$

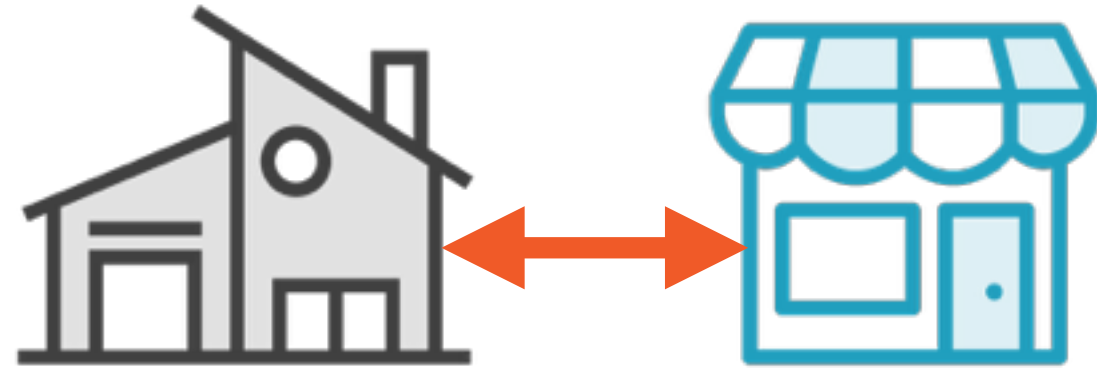
# Distance Measures



$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$



# Distance Measures

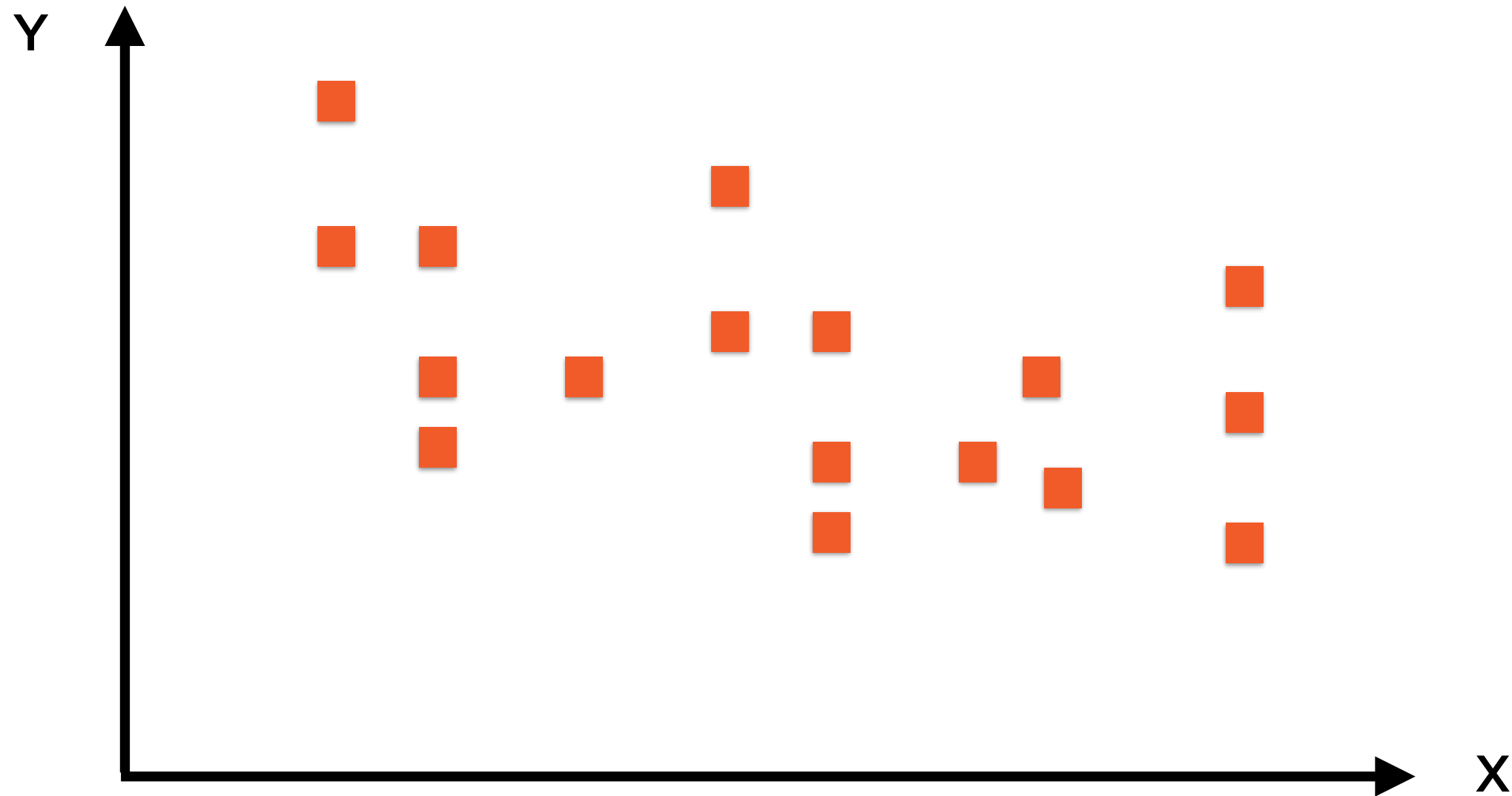


$$\text{L2 Distance}(t1, t2) = \text{sqrt}(\text{sum}((t1_i - t2_i)^2))$$

# Bias-variance Trade-off

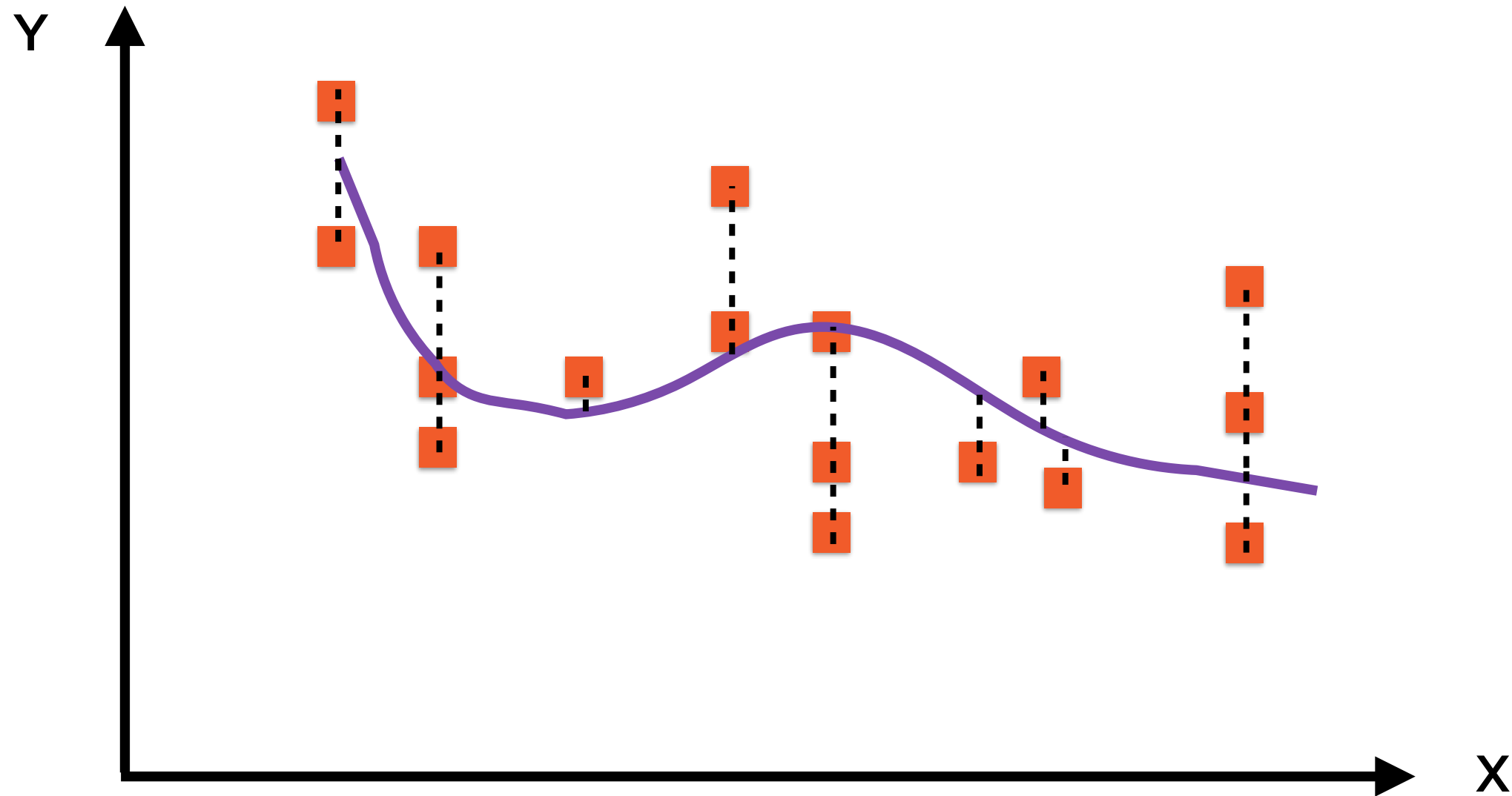
---

# Connecting the Dots



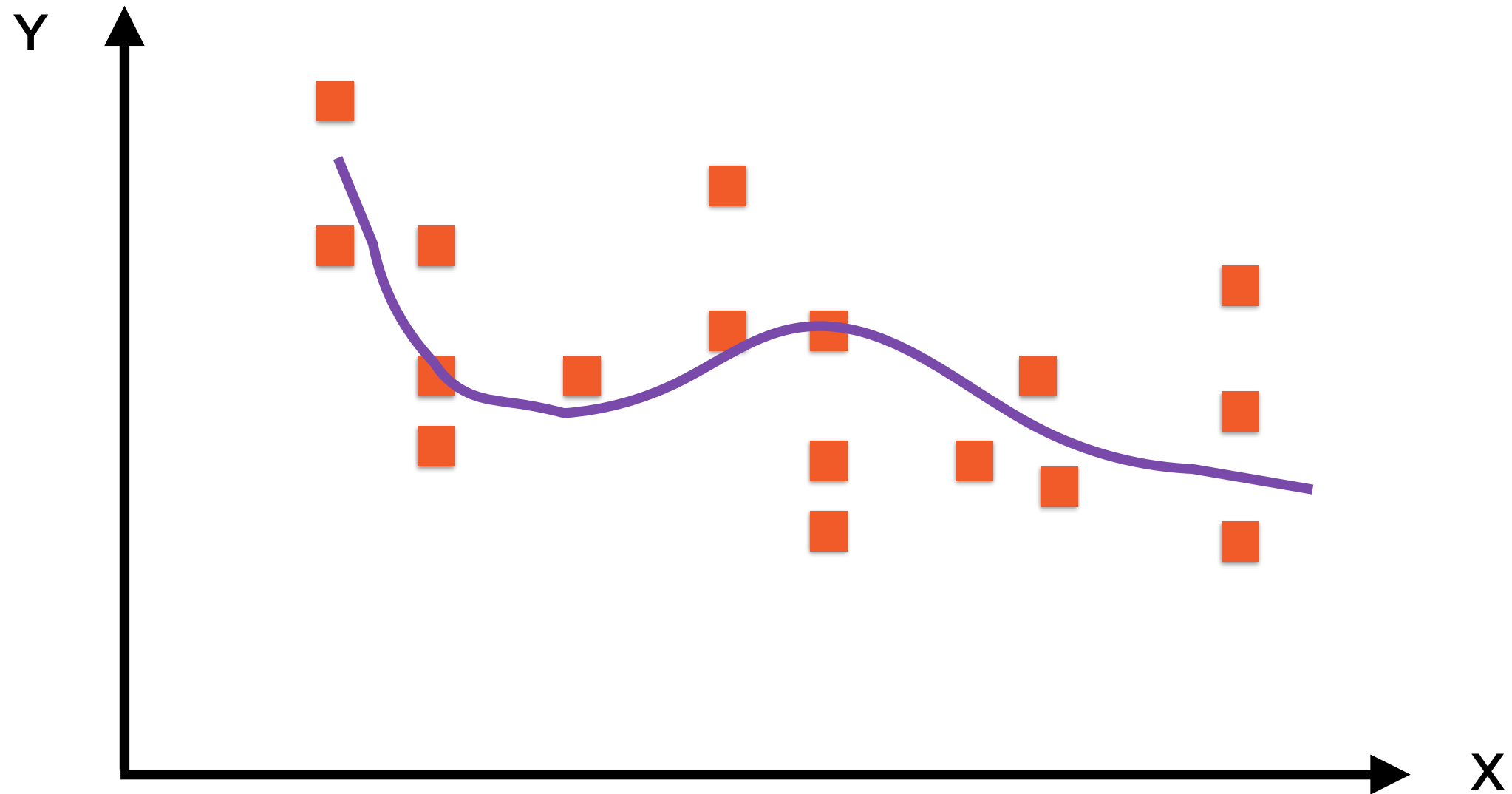
Challenge: Fit the “best” curve through these points

# Good Fit?



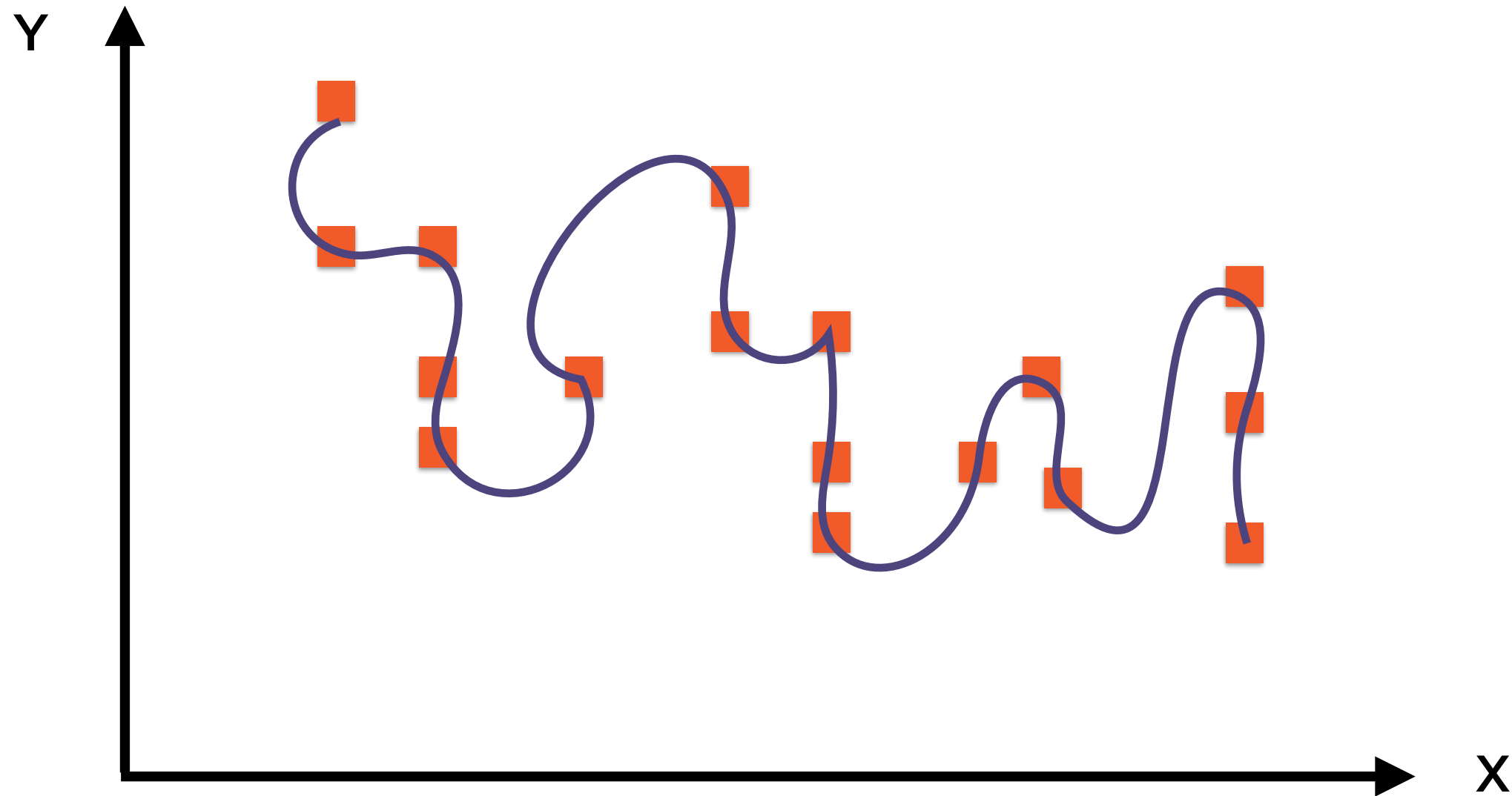
A curve has a “good fit” if the distances of points from the curve are small

# Connecting the Dots



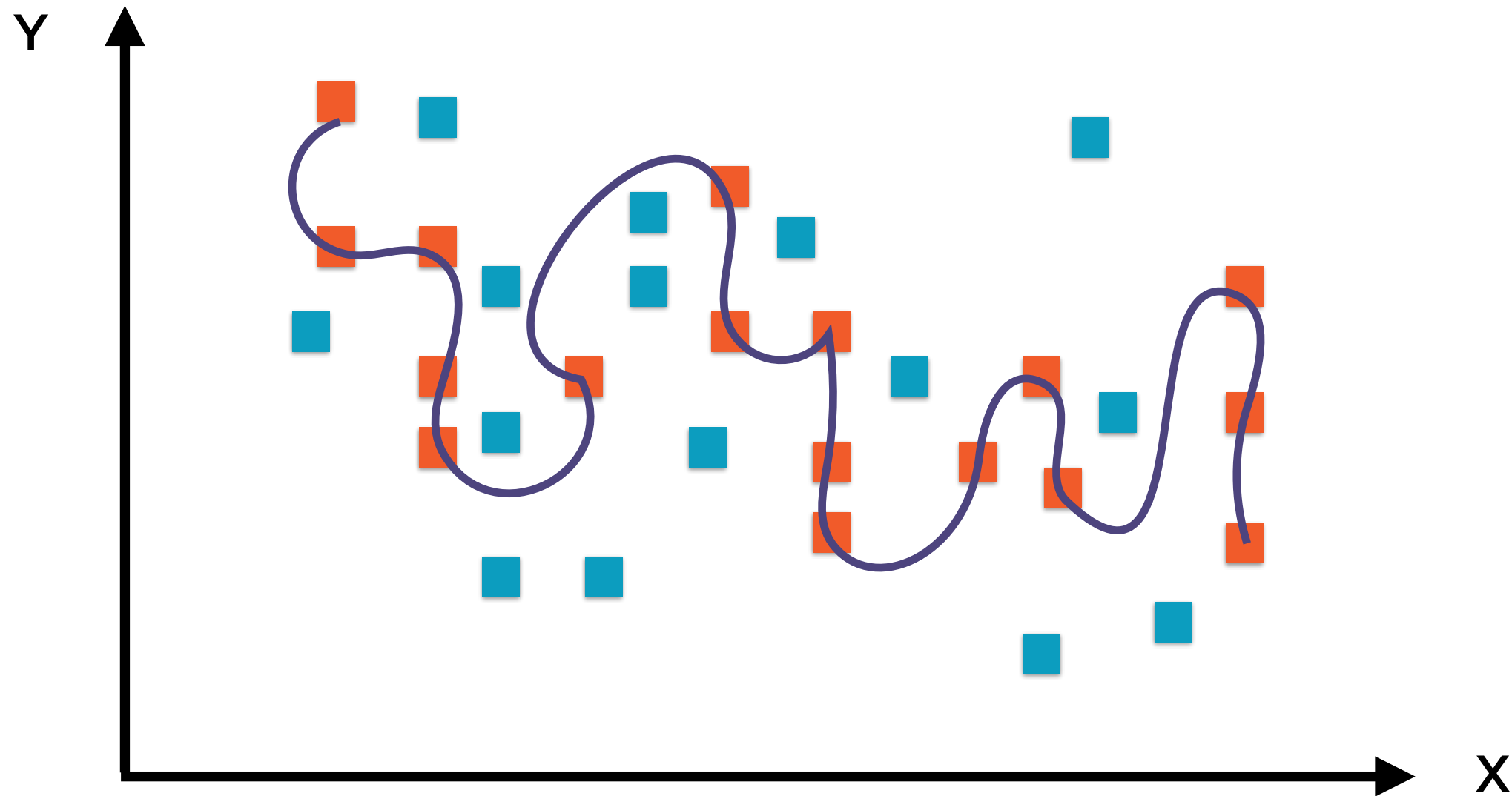
We could draw a pretty complex curve

# Connecting the Dots



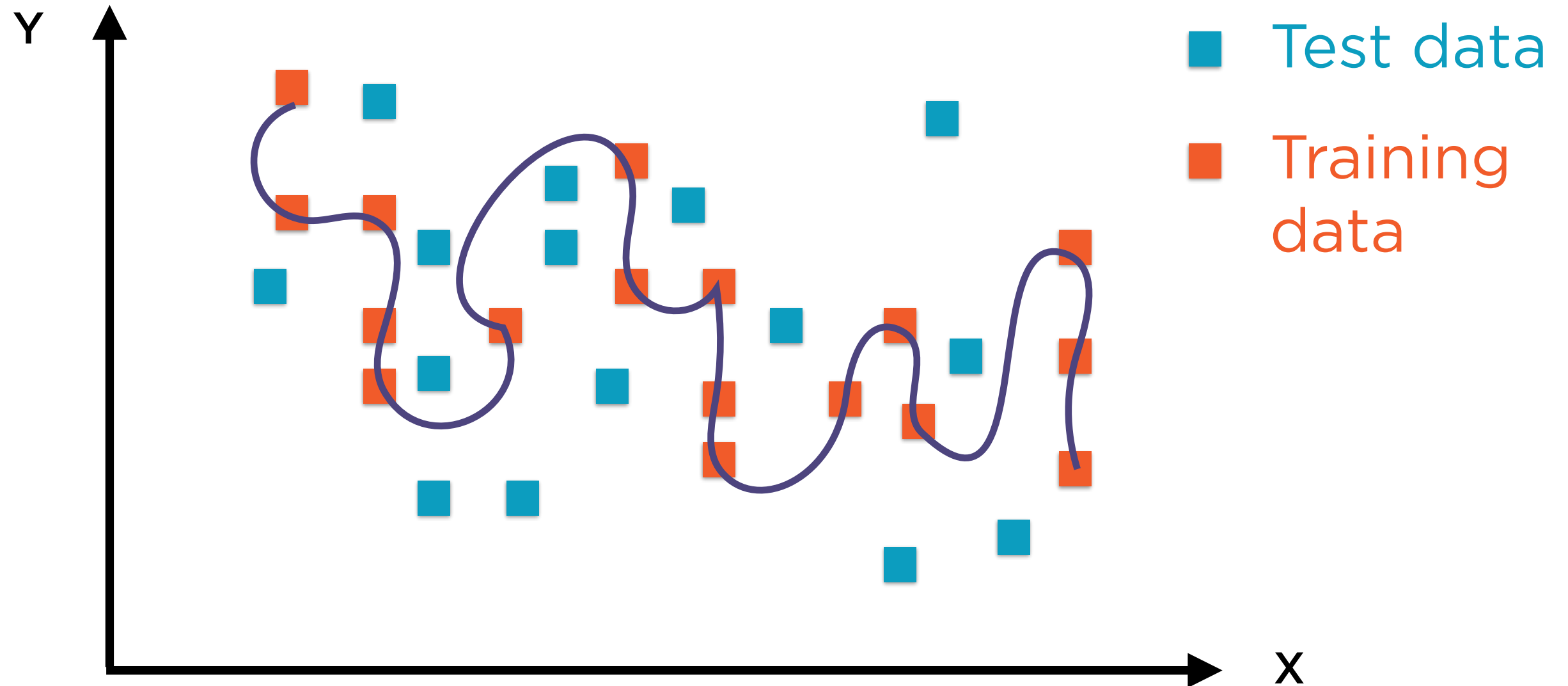
We can even make it pass through every single point

# Connecting the Dots



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

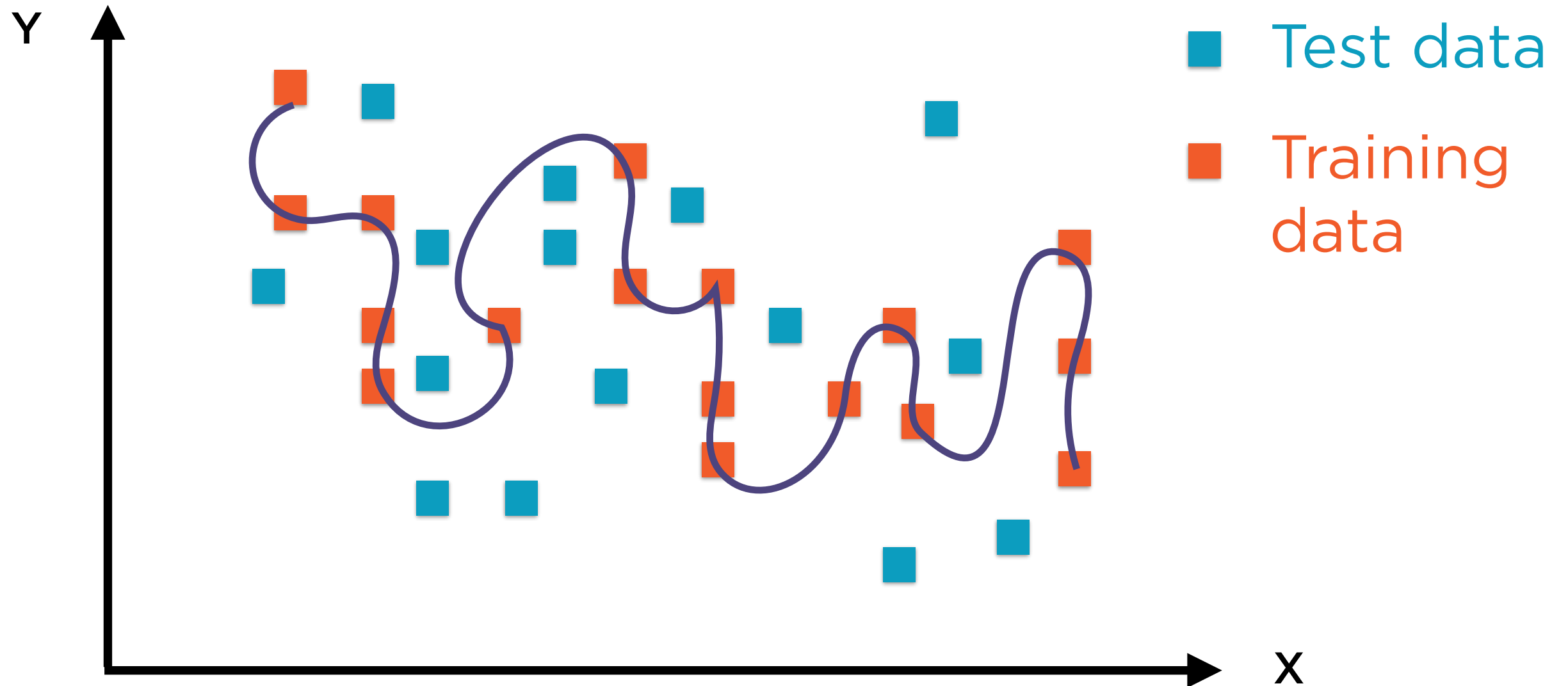
# Connecting the Dots



The original points were “training data”, the new points are “test data”

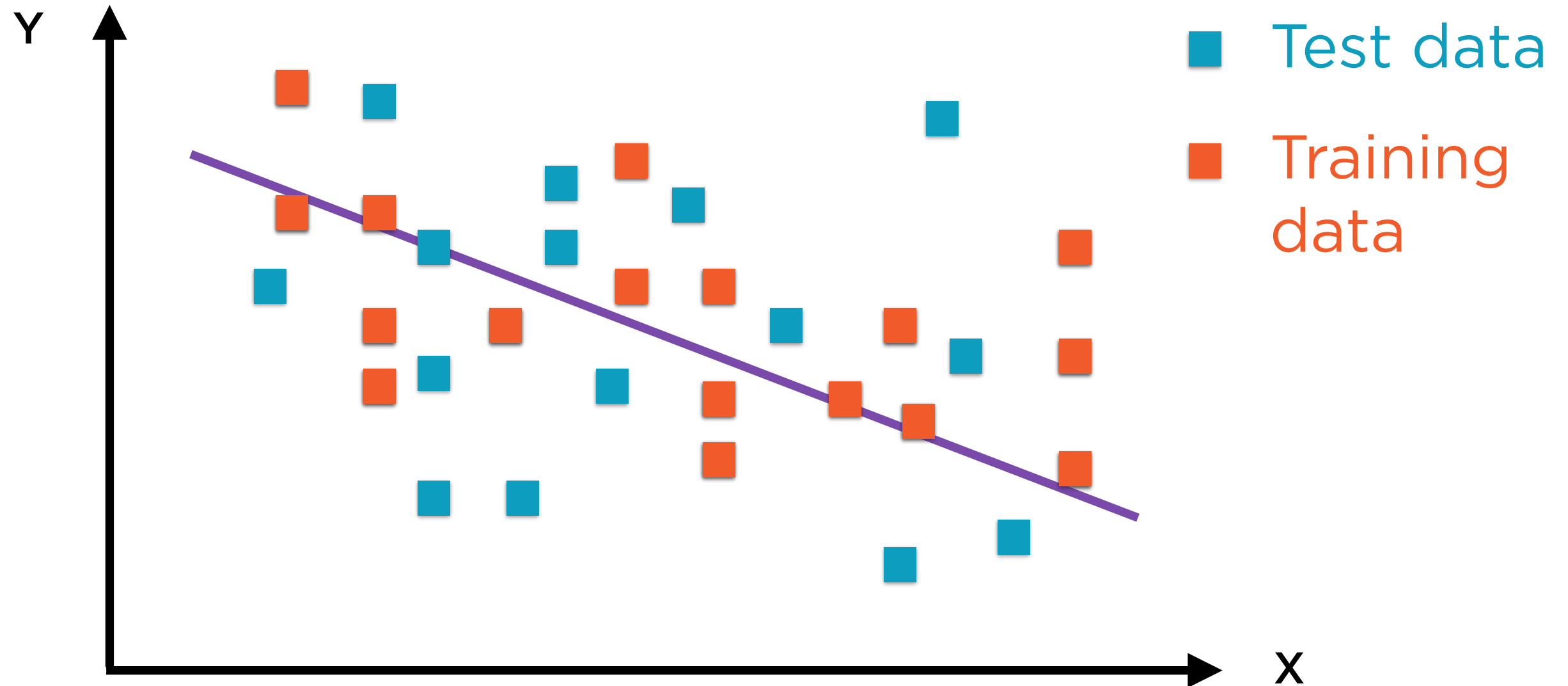


# Overfitting



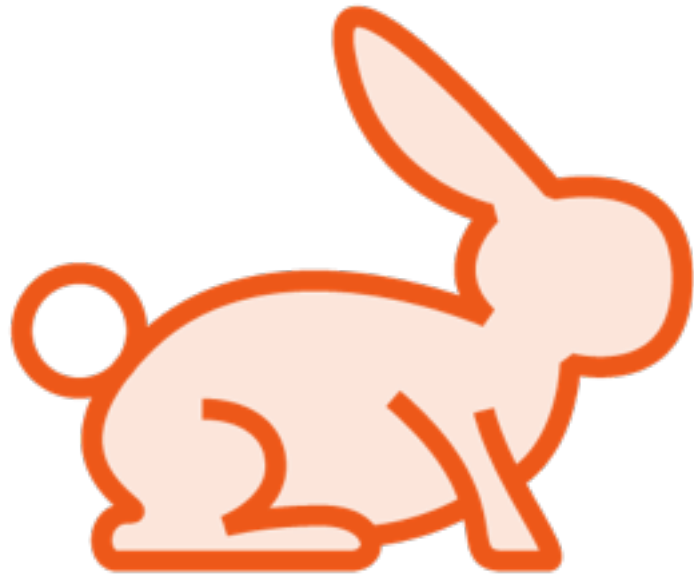
Great performance in training, poor performance in real usage

# Connecting the Dots



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

# Overfitting



**Low Training Error**

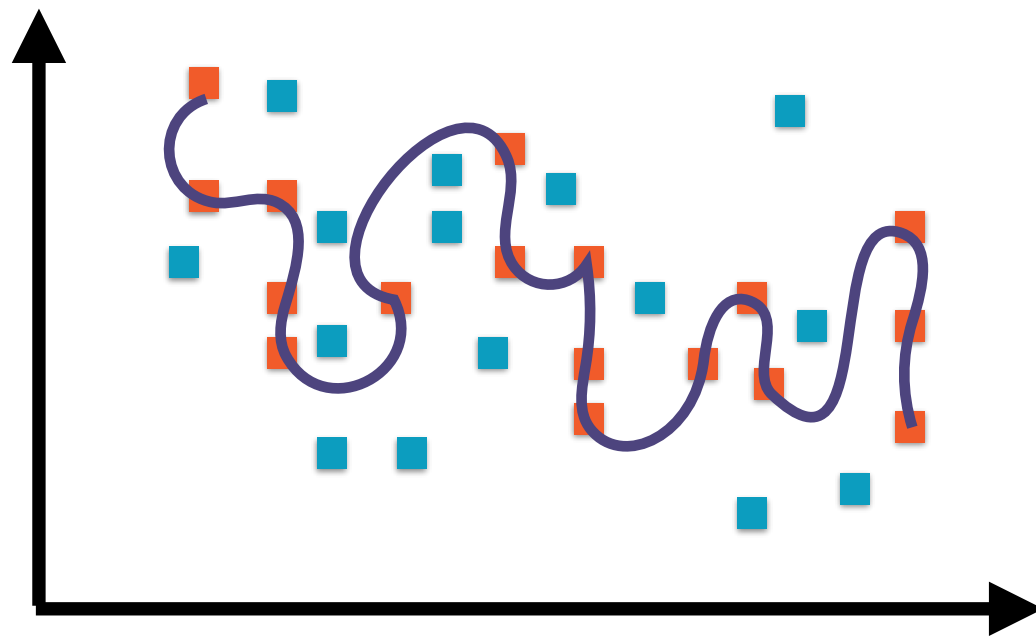
**Model does very well in training...**



**High Test Error**

**...but poorly with real data**

# Cause of Overfitting



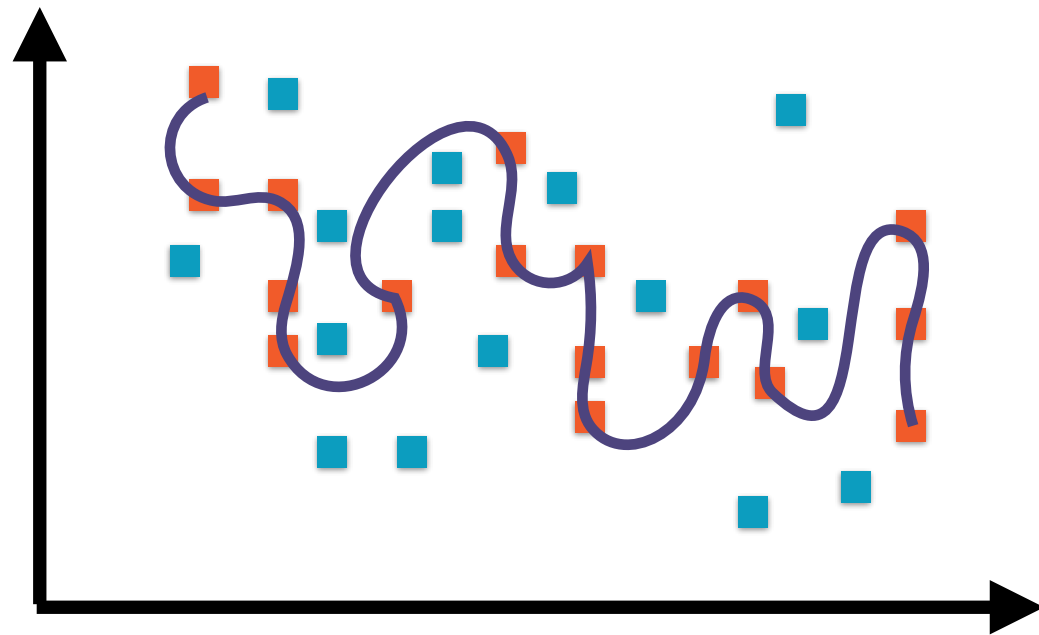
Sub-optimal choice in the **bias-variance** trade-off

An overfitted model has:

- high variance error
- low bias error

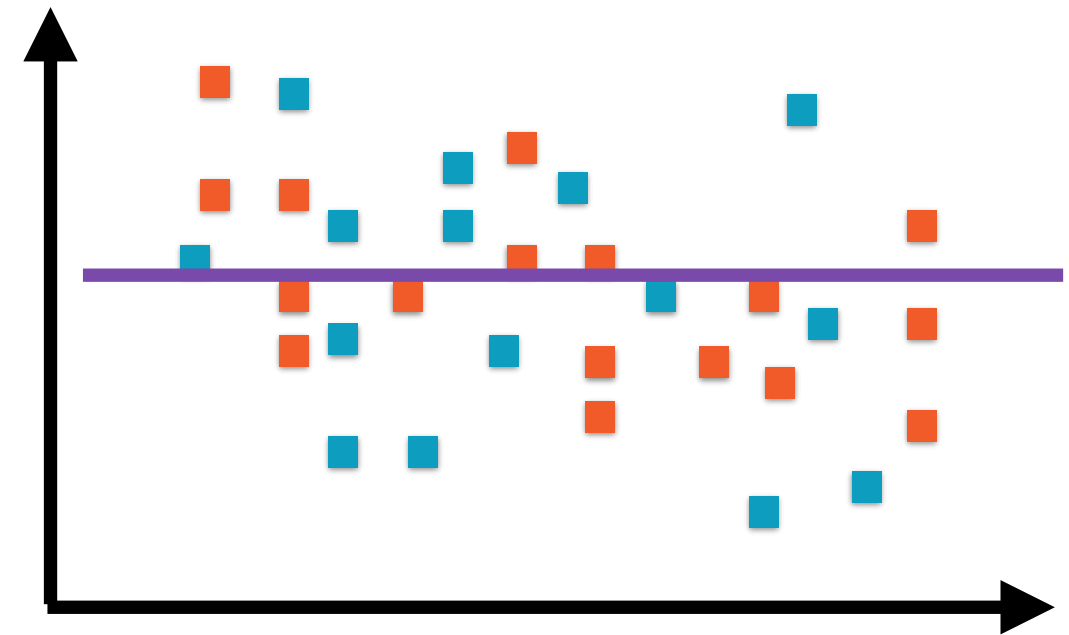


Bias



**Low bias**

Few assumptions about the underlying data

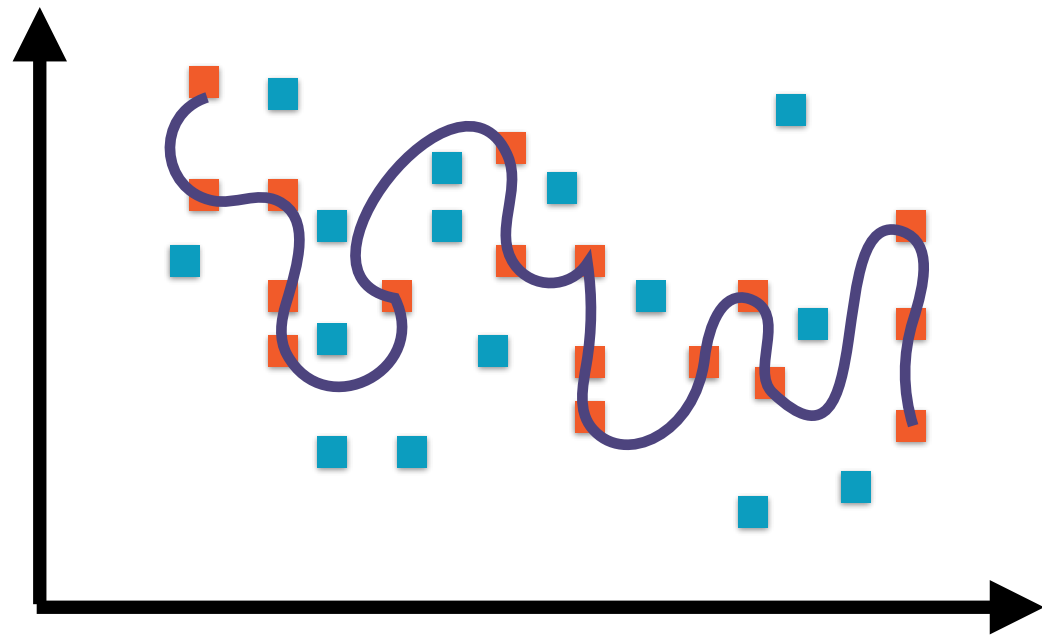


**High bias**

More assumptions about the underlying data

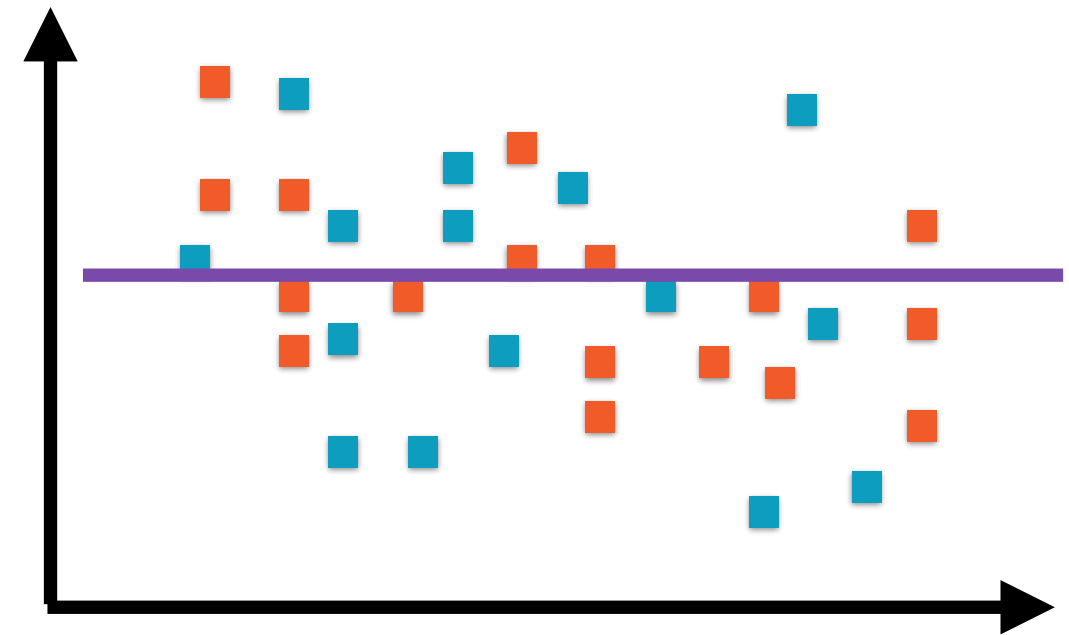


Bias



**Model too complex**

Training data all-important, model  
parameter counts for little

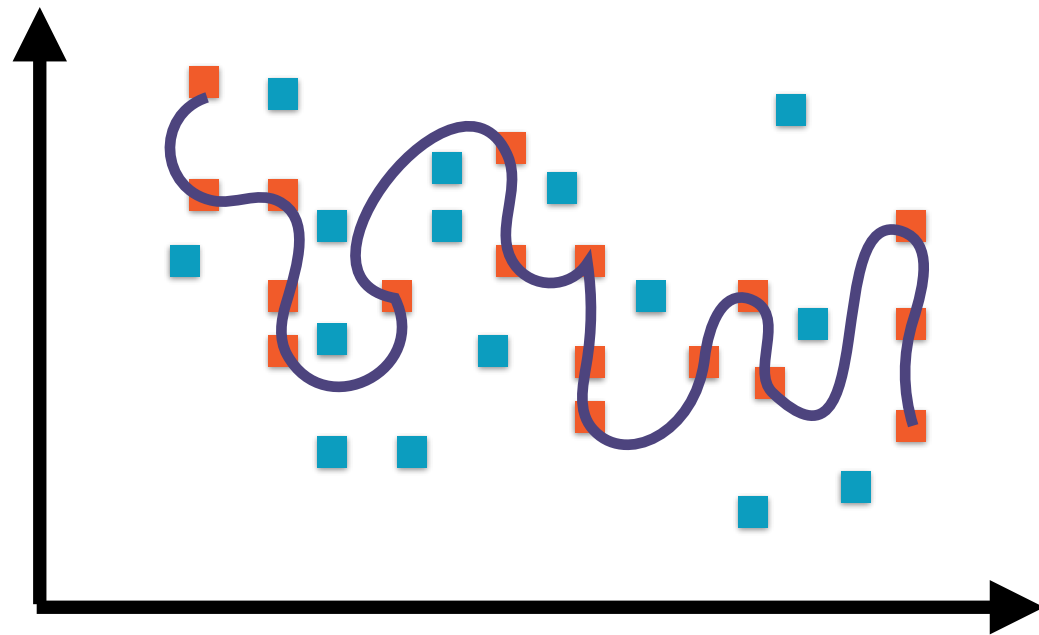


**Model too simple**

Model parameter all-important,  
training data counts for little

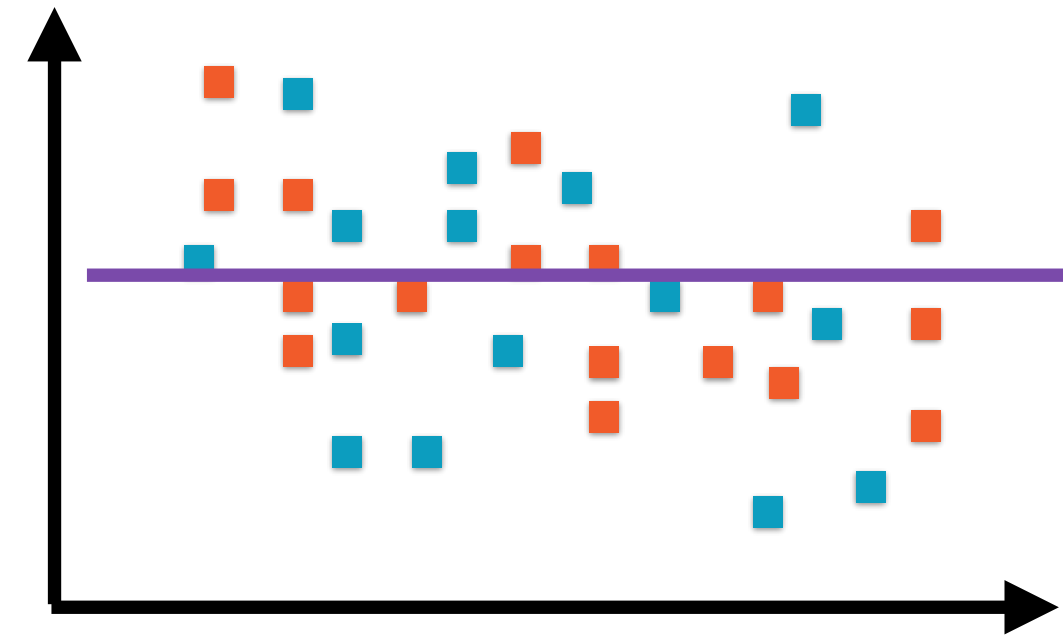


Variance



**High variance**

The model changes significantly  
when training data changes

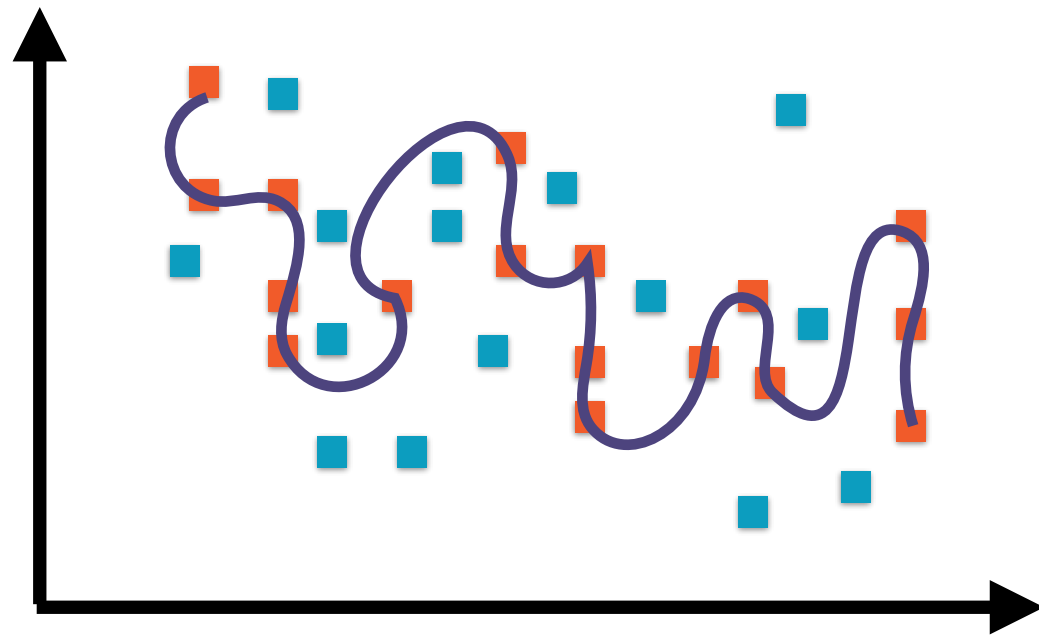


**Low variance**

The model doesn't change much  
when the training data changes

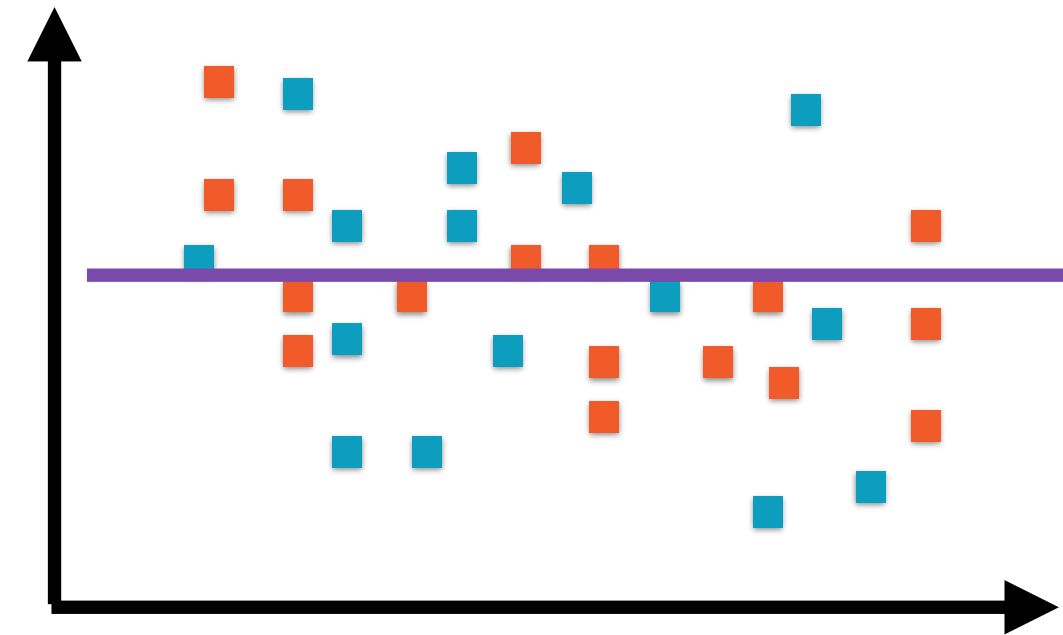


Variance



**Model too complex**

Model varies too much with changing  
training data

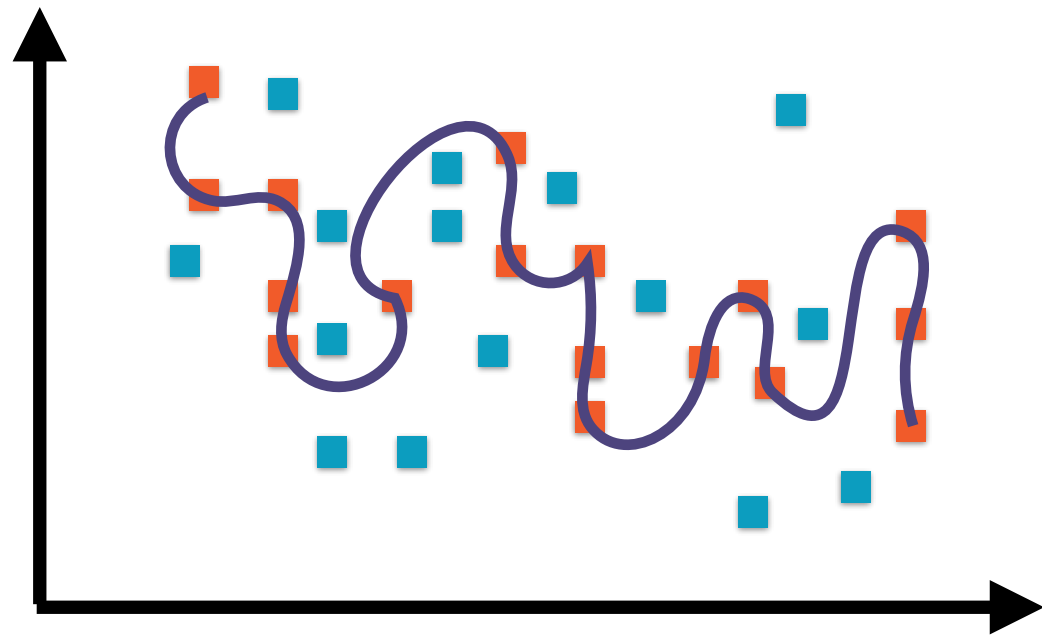


**Model too simple**

Model not very sensitive to training  
data

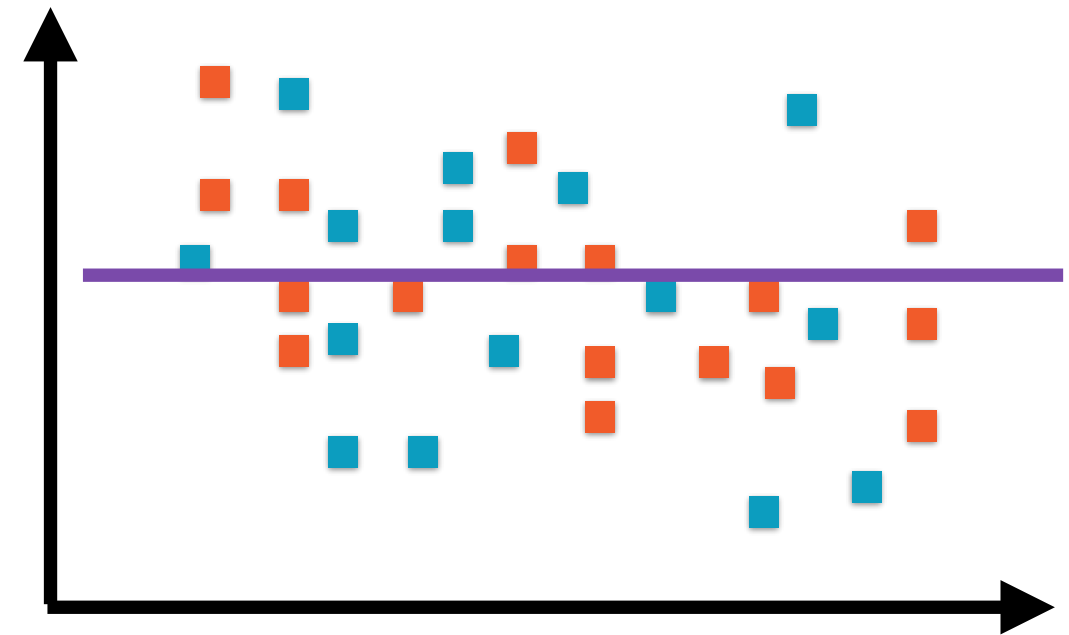


# Bias-variance Trade-off



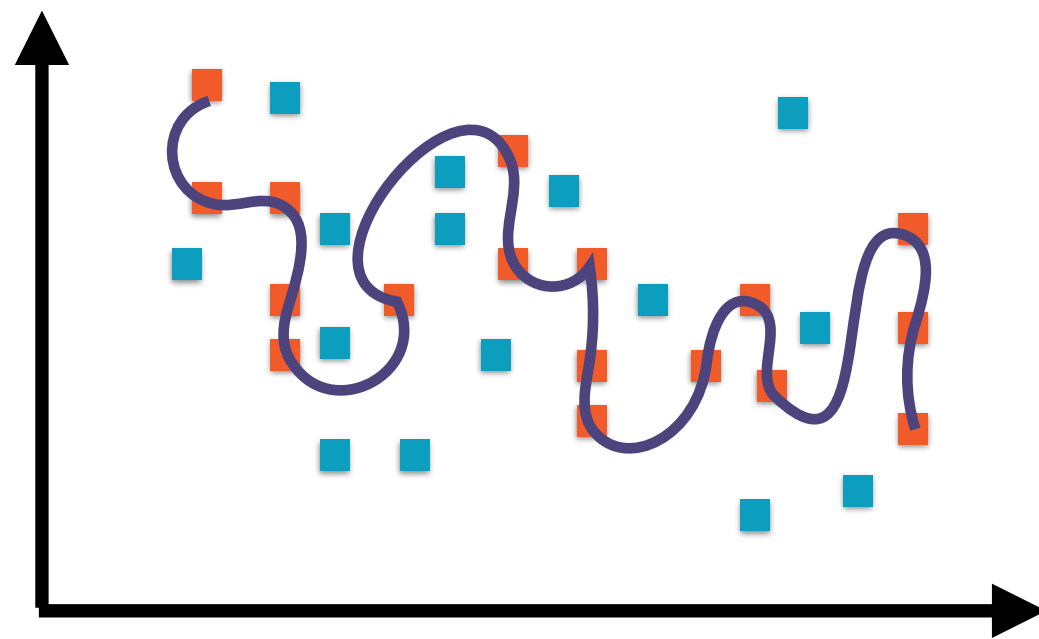
**Model too complex**

High variance error



**Model too simple**

High bias error



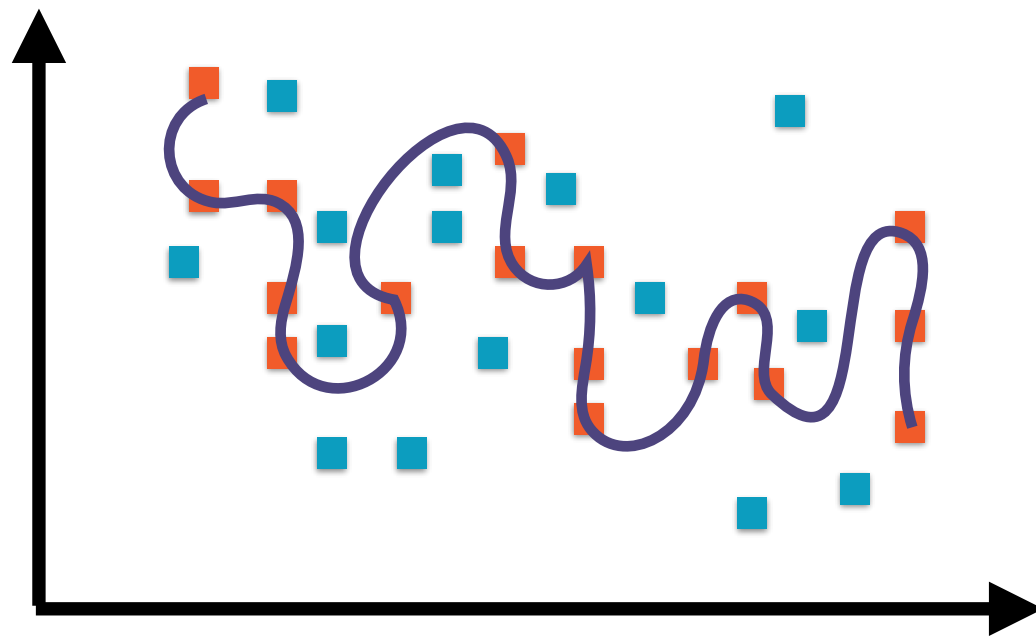
# Bias-variance Trade-off

- **High-bias algorithms: simple parameters**
  - Regression
- **High-variance algorithms: complex parameters**
  - Decision trees
  - Dense neural networks

# Overfitting, Dropout and Regularization

---

# Preventing Overfitting



- **Regularization**
- **Cross-validation**
- **Ensemble learning**
  - **Dropout**

# 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**



# Cross-Validation

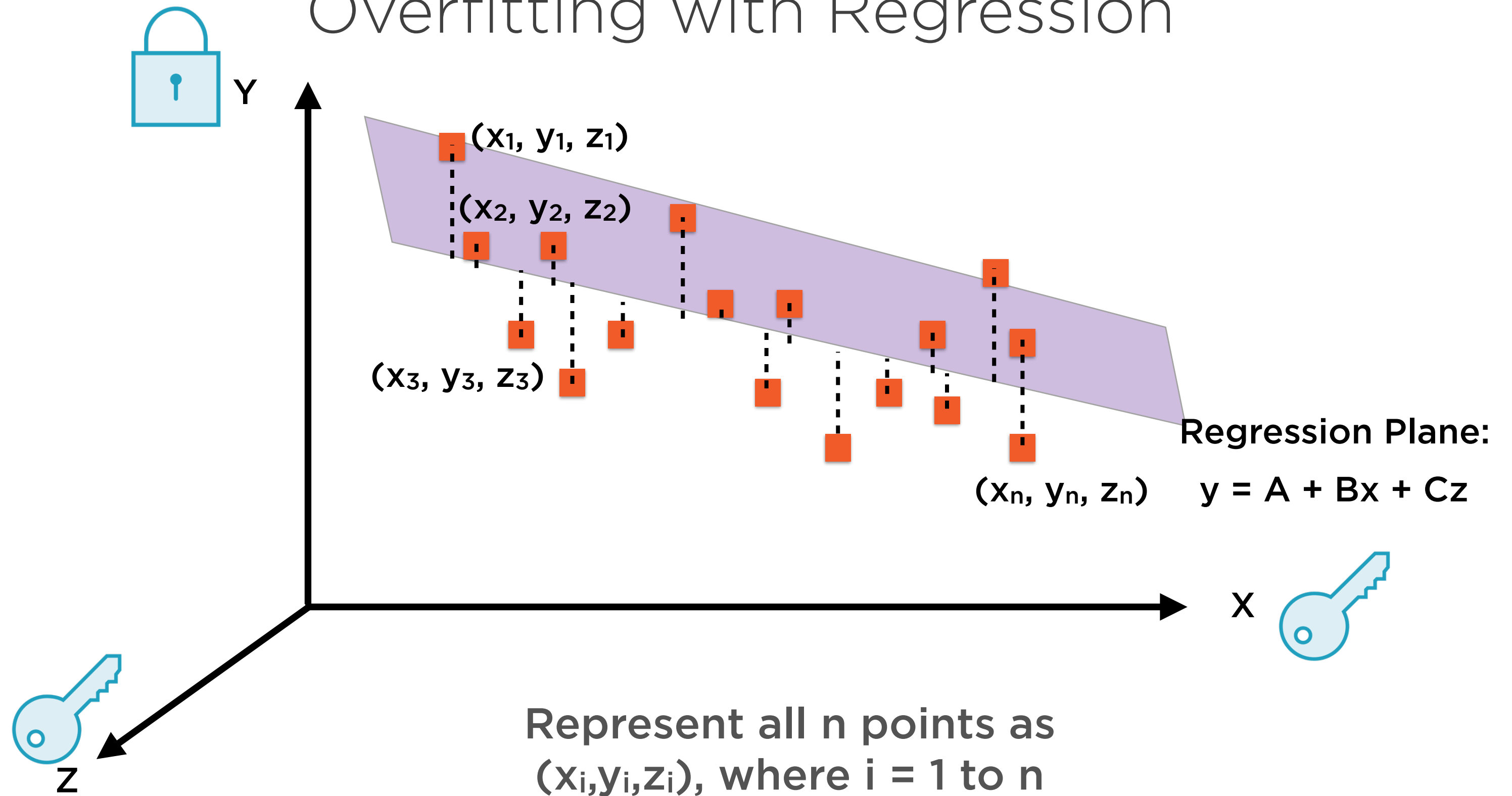
**Distinct training and validation phases**

**Train different models (with training data only)**

**Select model that does best on validation data**

**“Hyperparameter tuning”**

# Overfitting with Regression





# Multiple Regression

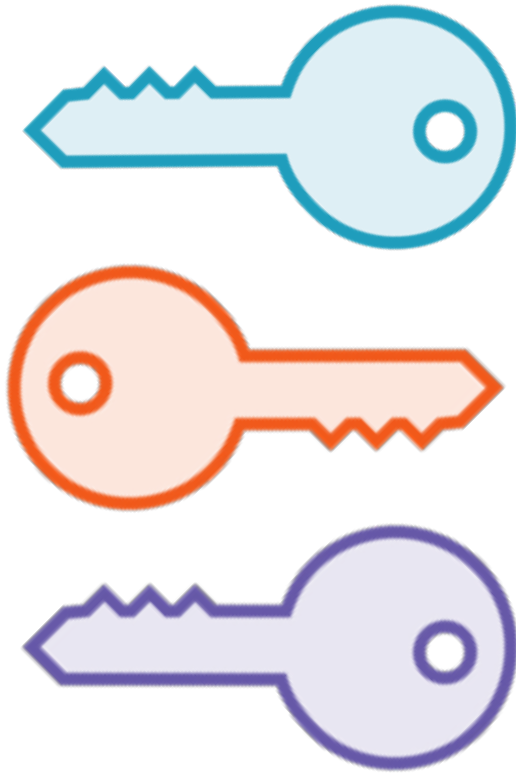
## **Regression Equation:**

$$y = C_1 + C_2x_1 + \dots + C_{k+1}x_k$$

Linear regression involves finding  $k+1$  coefficients,  $k$  for the explanatory variables, and 1 for the intercept

A big risk with regression is  
**multicollinearity**: X variables  
containing the same information

# Success as a Salesperson



## Causes

Number of cold calls, years of  
experience in sales jobs



## Effect

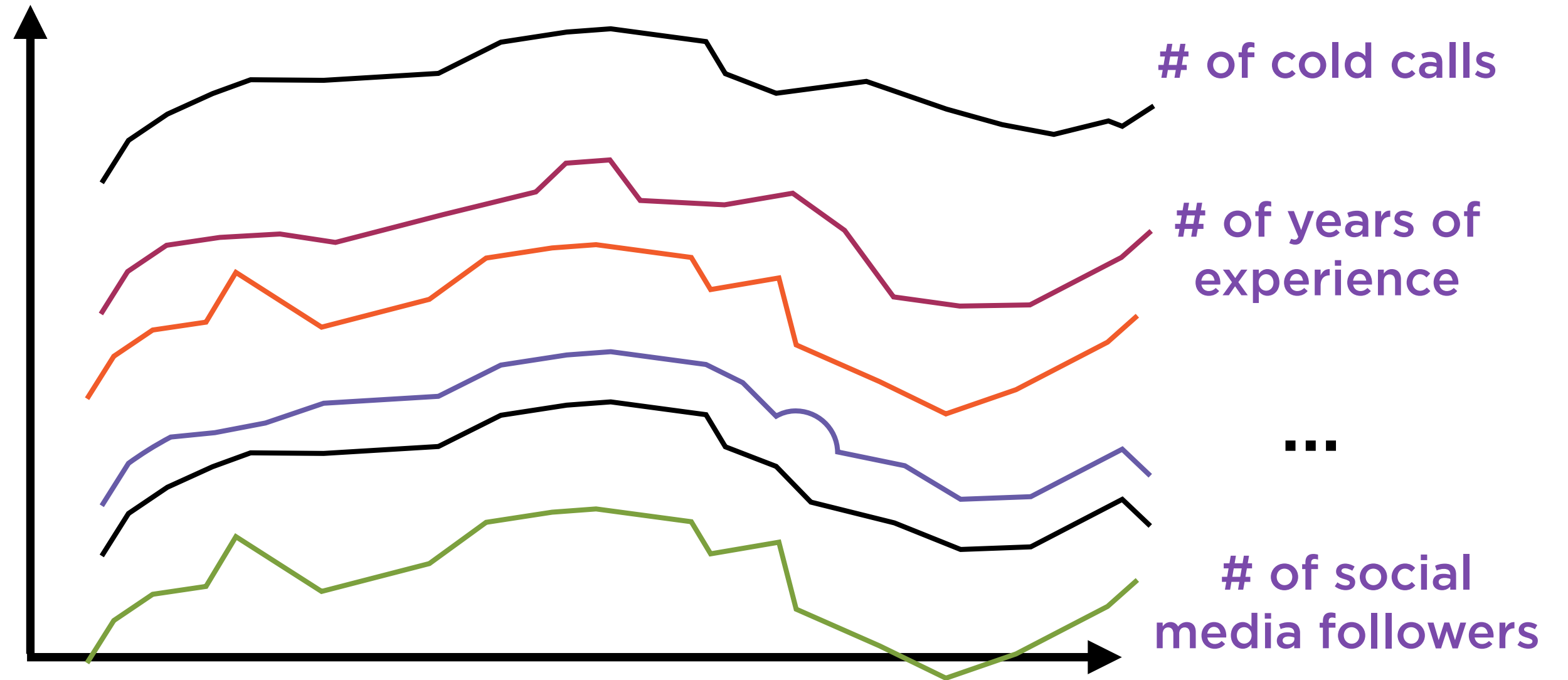
Bonus as member of sales team

# Kitchen Sink Regression

**Proposed Regression Equation:**

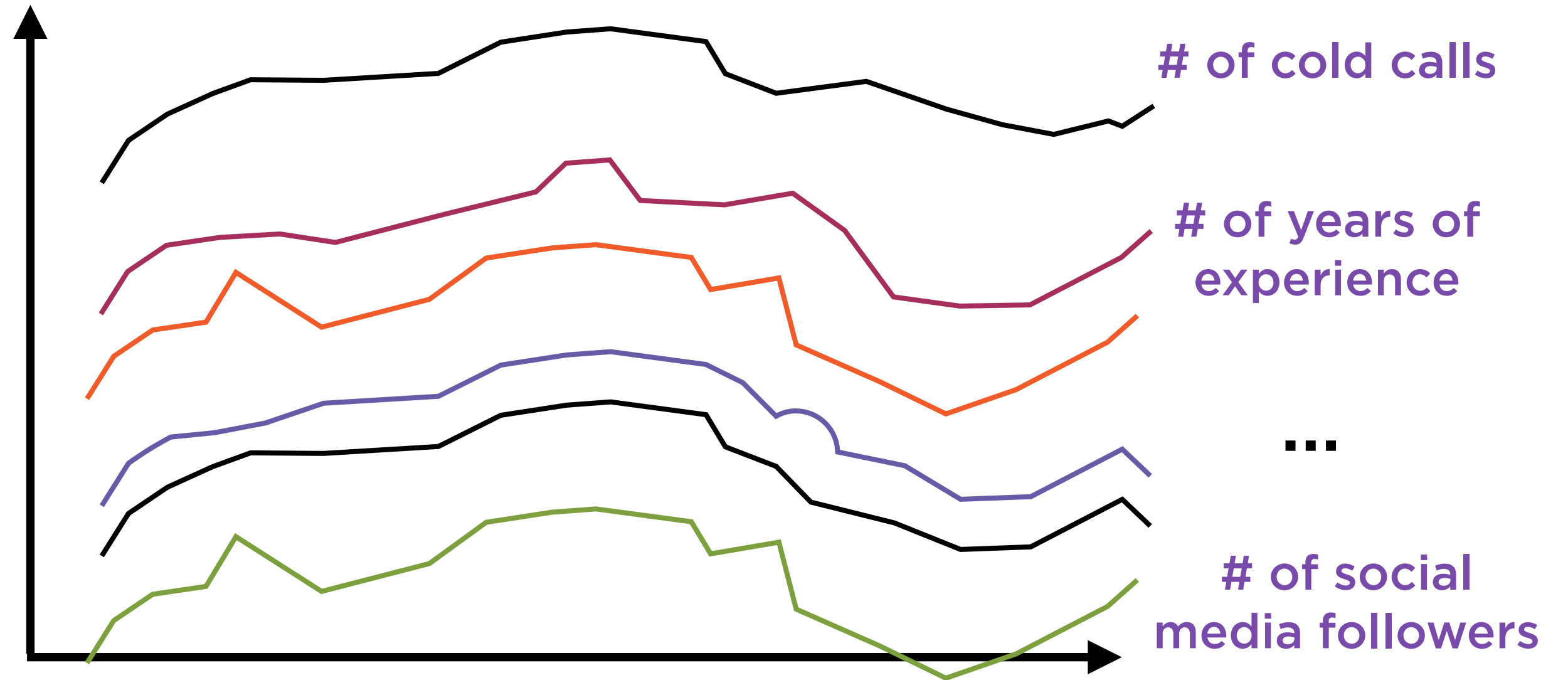
$$\begin{aligned} \text{BONUS} = & A + B \text{ COLDCELLS} + C \text{ EXPERIENCE} + D \\ & \text{NUMFOLLOWERS} + E \text{ HONESTY} + F \text{ PUNCTUALITY} \\ & + \dots \end{aligned}$$

# Bad News: Multicollinearity Detected



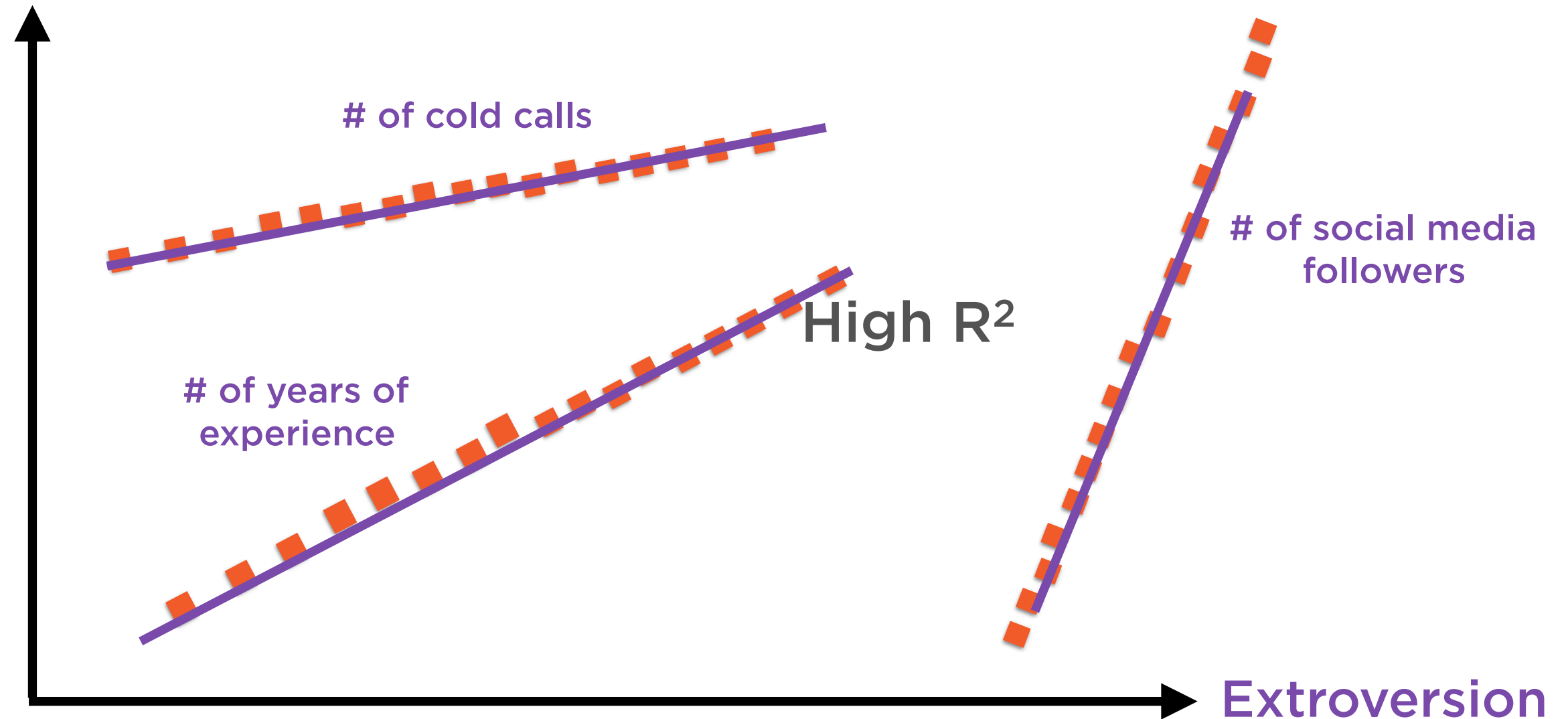
6 of 10 explanatory variables are highly correlated with each other

# Underlying Cause: Extroversion



Each of these explanatory variables is caused by an underlying personality trait

# Underlying Cause: Extroversion



Simply measure extroversion and use it instead of the correlated explanatory variables

# Kitchen Sink Regression



**10 Causes**

Cold calls, experience, social media followers, perceived honesty, billing punctuality...



**1 Effect**

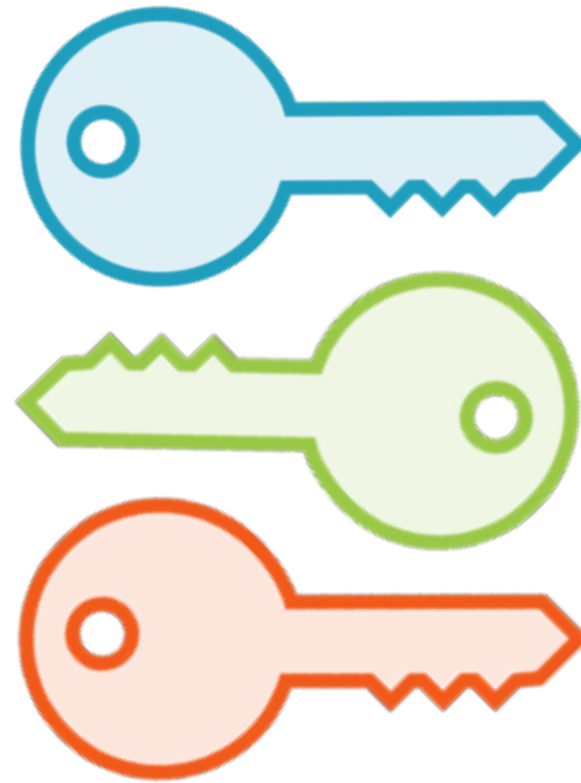
Bonus in sales team



# Factor Analysis



**Many Observed  
Causes**

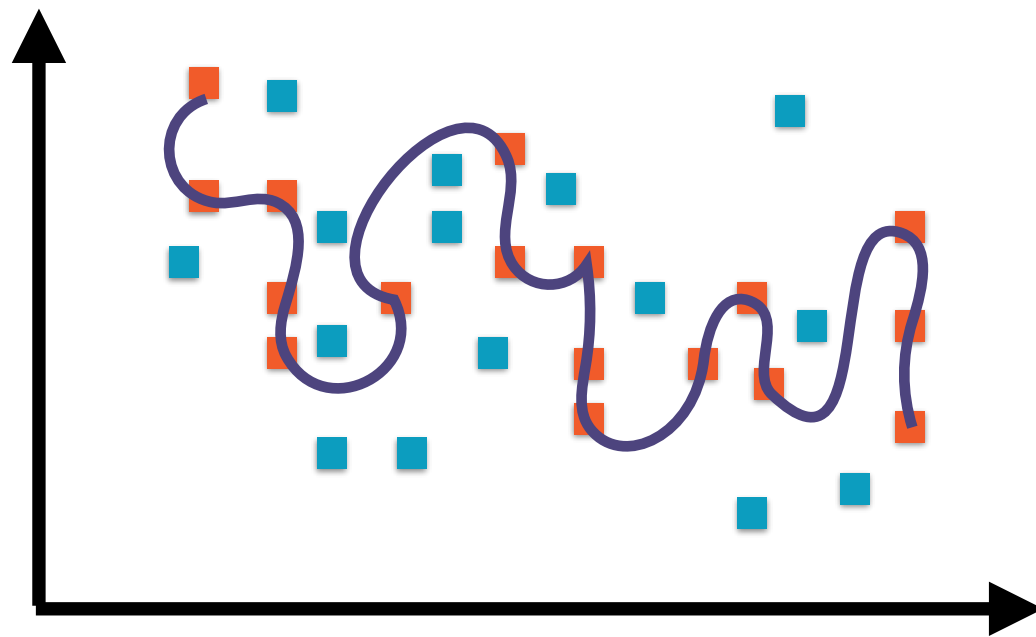


**Few Underlying  
Causes**



**One Effect**

# Overfitting in Regression



**Multi-collinearity in regression leads to overfitting**

**Model performs well in training, poorly in prediction**

**Various techniques to improve regression algorithm**

# Preventing Overfitting

---

# Regularized Regression Models

## Lasso Regression

Penalizes large regression coefficients

## Ridge Regression

Also penalizes large regression coefficients

## Elastic Net Regression

Simply combines lasso and ridge



# Regularization

**Penalize complex models**

**Add penalty to objective function**

**Penalty as function of regression coefficients**

**Forces optimizer to keep it simple**



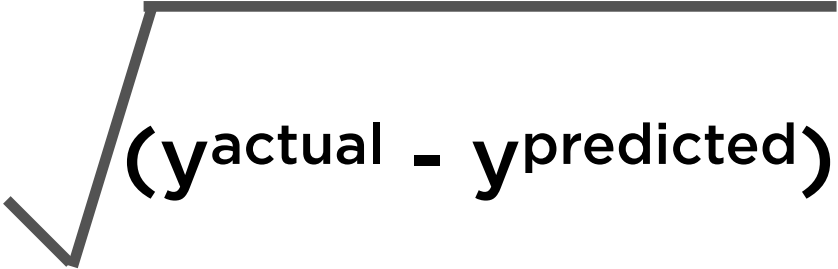
# Regularization

**Regularization reduces variance error**

**But increases bias**

# Ordinary MSE Regression

**Minimize**


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

**To find**

**A, B**

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

$$y = A + Bx$$

# Lasso Regression

Minimize

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

$$+ \alpha (|A| + |B|)$$

To find

A, B

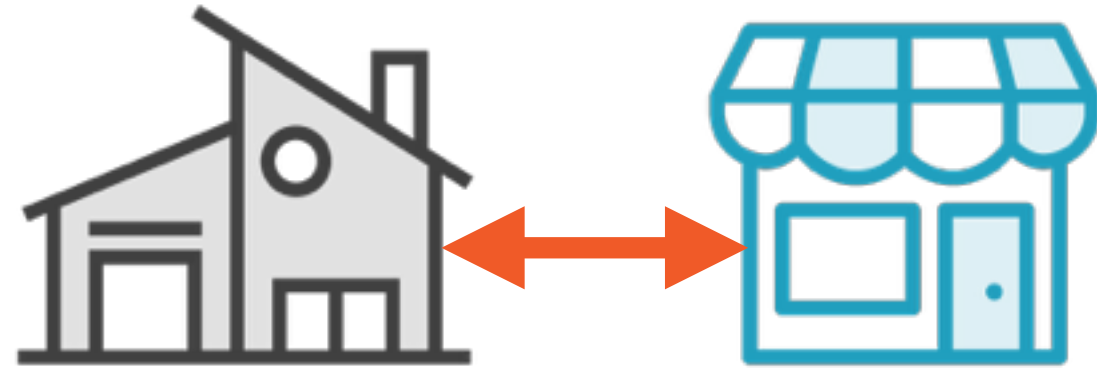
$\alpha$  is a hyperparameter

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

$$y = A + Bx$$



# L-1 Norm



$$L^1\text{-Norm}(A, B_1, B_2 \dots B_n) = |A|^1 + |B_1|^1 + |B_2|^1 \dots + |B_n|^1$$

# Lasso Regression

Minimize

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

$$+ \alpha (|A| + |B|)$$

To find

A, B


$\alpha$  is a hyperparameter

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

$$y = A + Bx$$

# Lasso Regression

Minimize


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

$$+ \alpha (|A| + |B|)$$

To find

A, B



L-1 Norm of regression  
coefficients

$\alpha$  is a hyperparameter

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

$$y = A + Bx$$

# Ridge Regression

Minimize

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

$$+ \alpha (|A|^2 + |B|^2)$$

To find

A, B

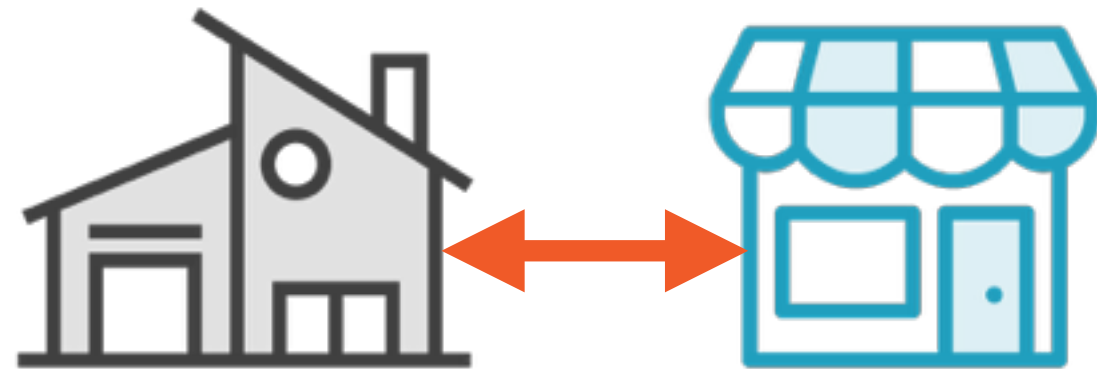
L-2 Norm of regression  
coefficients

$\alpha$  is a hyperparameter

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

$$y = A + Bx$$

# L-2 Norm



$$L^2\text{-Norm}(A, B_1, B_2 \dots B_n) = |A|^2 + |B_1|^2 + |B_2|^2 \dots + |B_n|^2$$



# Lasso Regression

Add penalty for **large coefficients**

Penalty term is L-1 norm of coefficients

Penalty weighted by **hyperparameter  $\alpha$**



# Lasso Regression

$\alpha = 0$  ~ Regular (MSE regression)

$\alpha \rightarrow \infty$  ~ Force small coefficients to zero

Model selection by tuning  $\alpha$

Eliminates unimportant features



# Lasso Regression

“Lasso” ~ Least Absolute Shrinkage and Selection Operator

Math is complex

No closed form, needs numeric solution



# Ridge Regression

Minimize

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

To find

A, B

$$+ \alpha (|A|^2 + |B|^2)$$

L-2 Norm of regression  
coefficients

$\alpha$  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  $\alpha$****

# Ridge Regression



**Unlike lasso, ridge regression has closed-form solution**

**Unlike lasso, ridge regression will not force coefficients to 0**

- Does not perform model selection**

Demo

**Implementing Lasso and Ridge  
regression in scikit-learn**

# Setting Up the SVM Regression Problem

---

SVMs are typically used for  
classification problems

SVRs use the same underlying  
principles with a **different**  
**objective function**

# Data in One Dimension



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

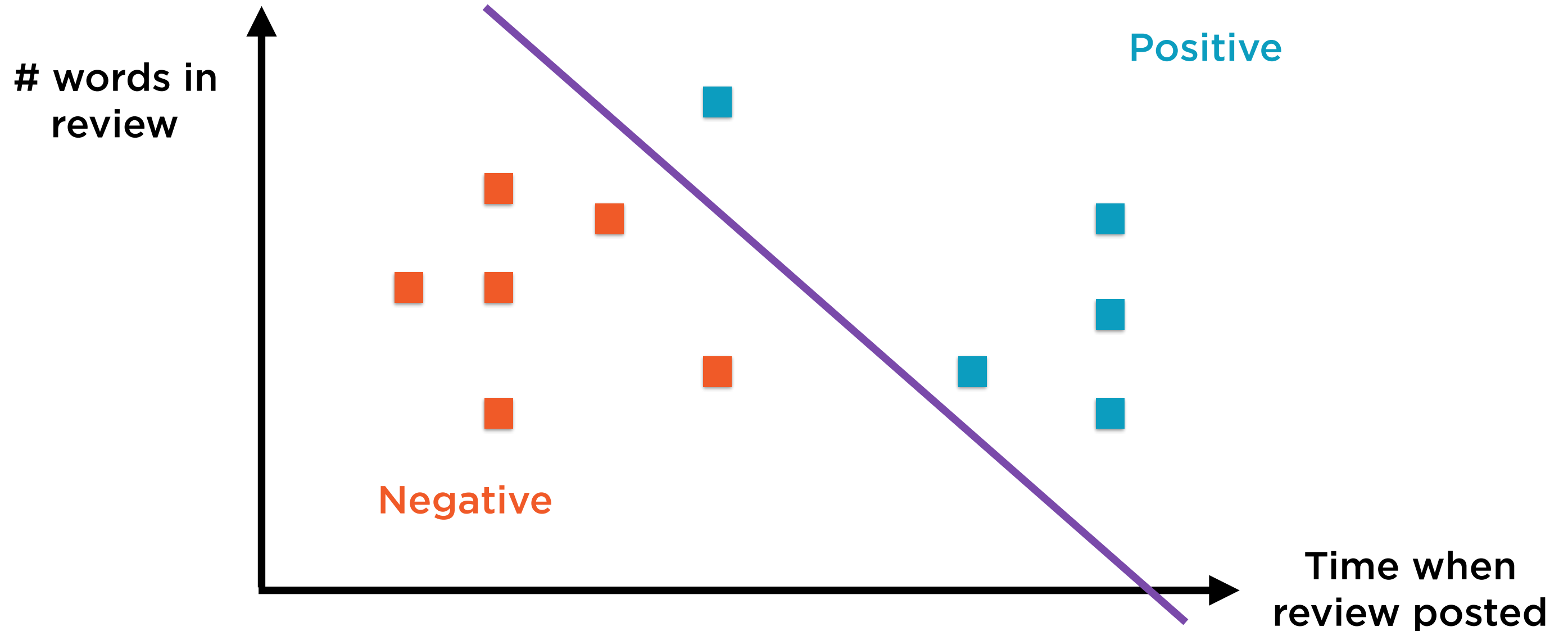
# Data in One Dimension



Unidimensional can also be separated, or classified,  
using a **point**

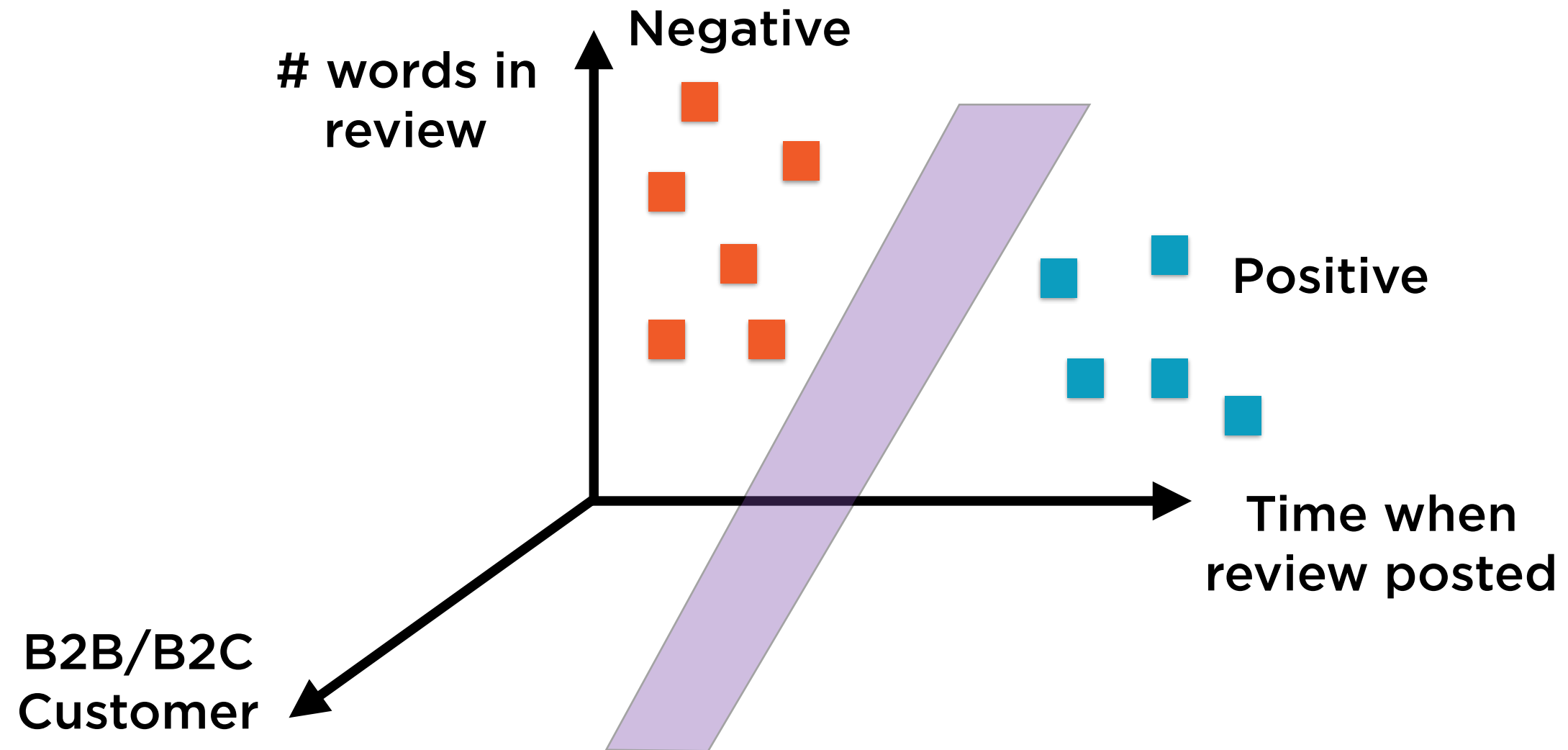


# Data in Two Dimensions



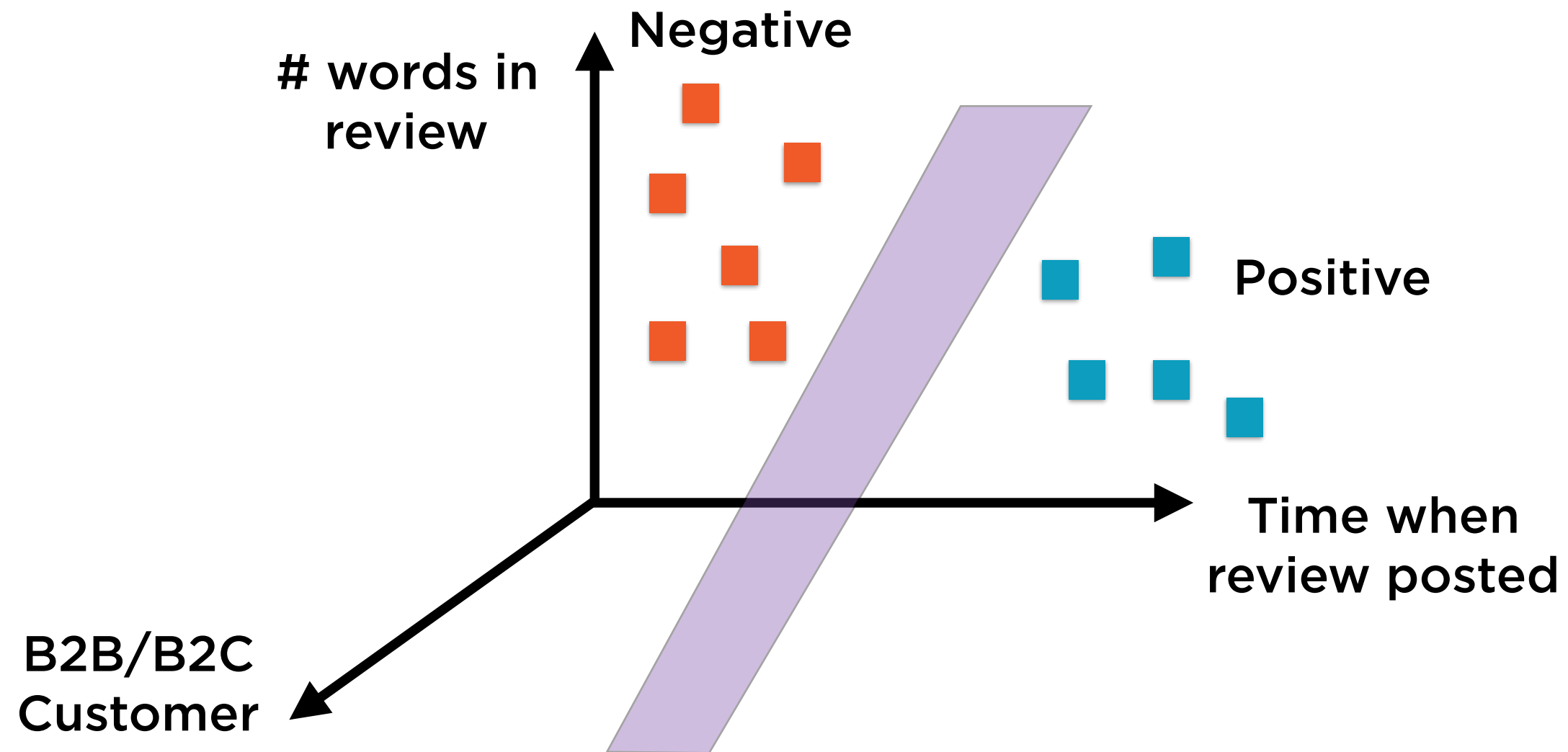
Bidimensional data points can be represented using a plane, and classified using a line

# Data in N Dimensions



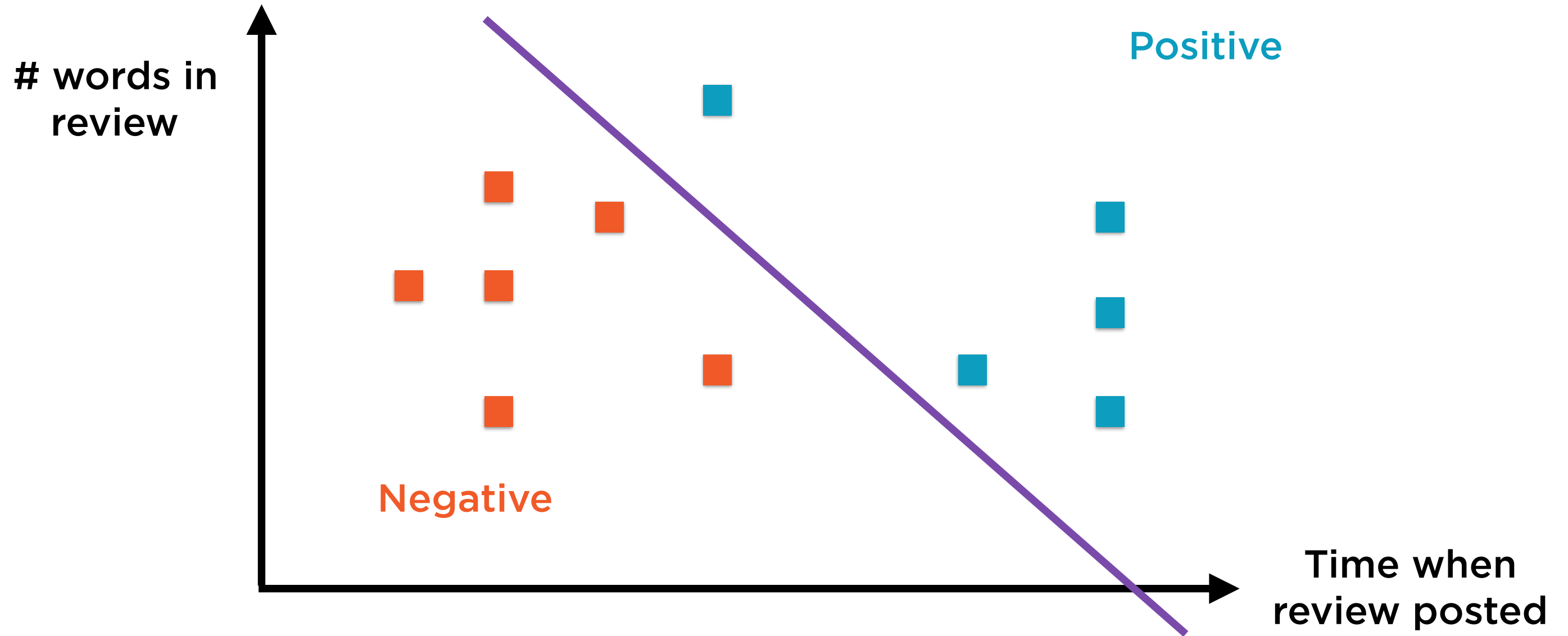
N-dimensional data can be represented in a **hypercube**, and classified using a **hyperplane**

# Support Vector Machines



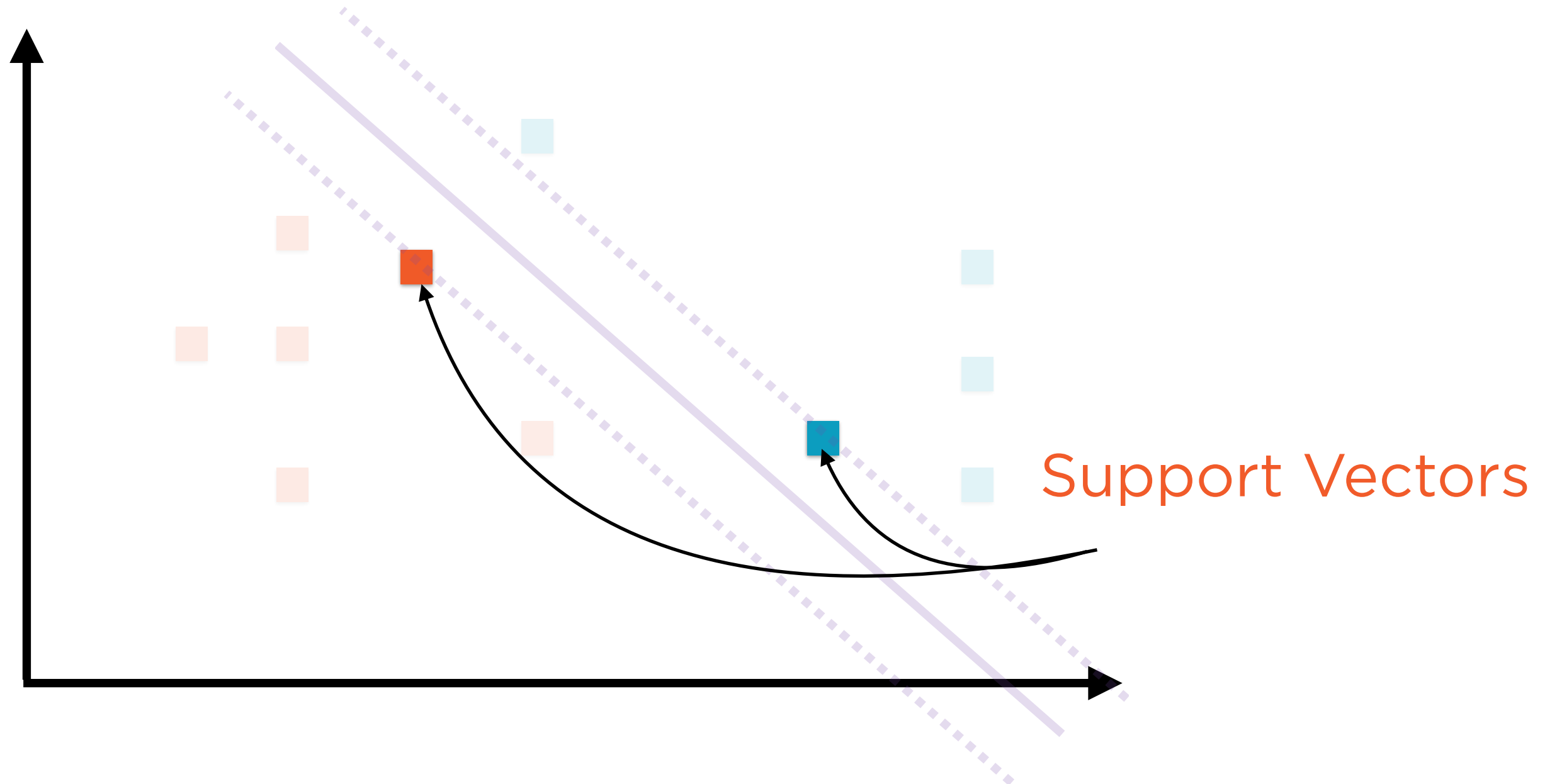
SVM classifiers find the hyperplane that best separates points in a hypercube

# Classification



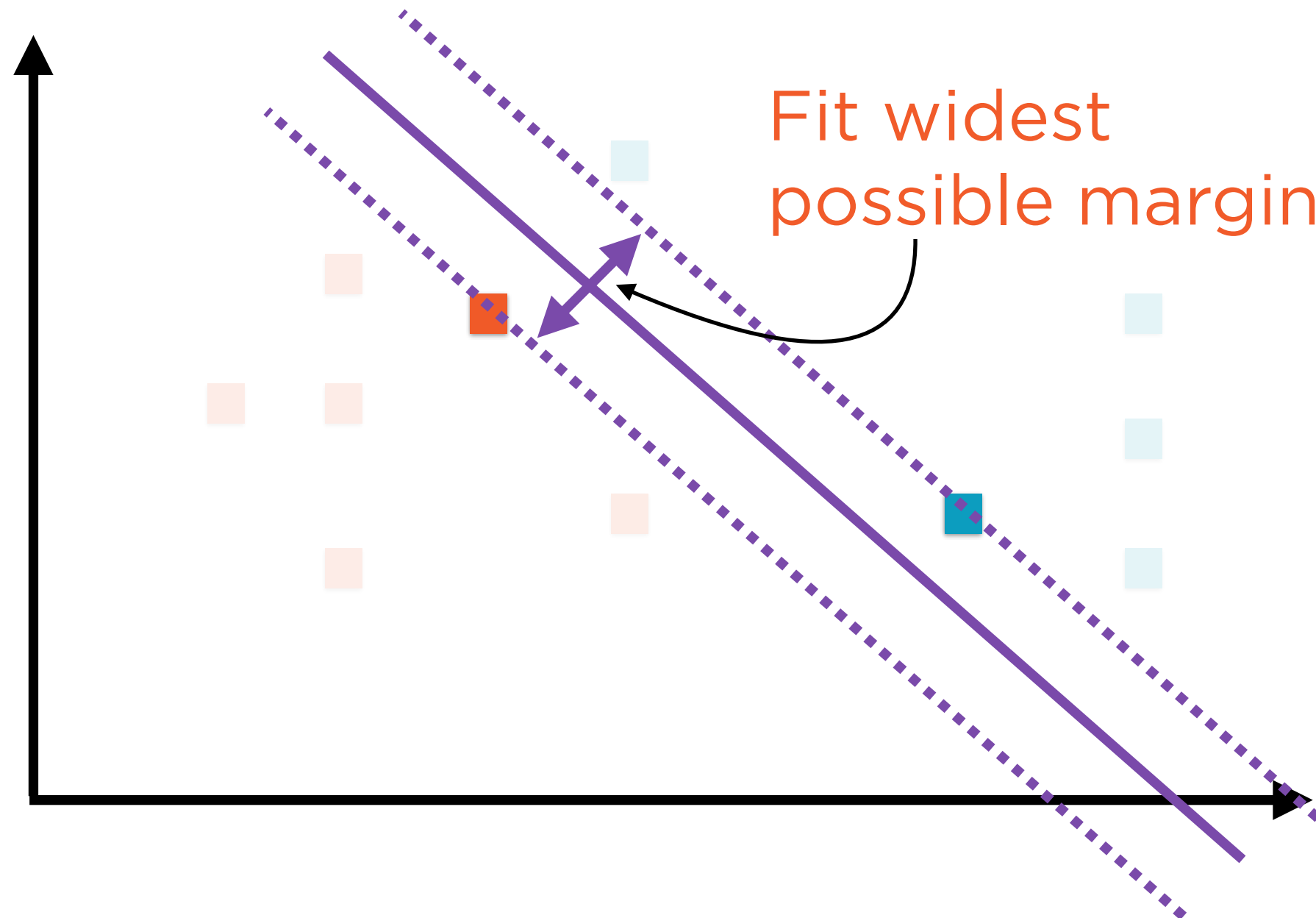
Ideally, data is linearly separable - hard decision boundary

# Classification



The nearest instances on either side of the boundary are called the support vectors

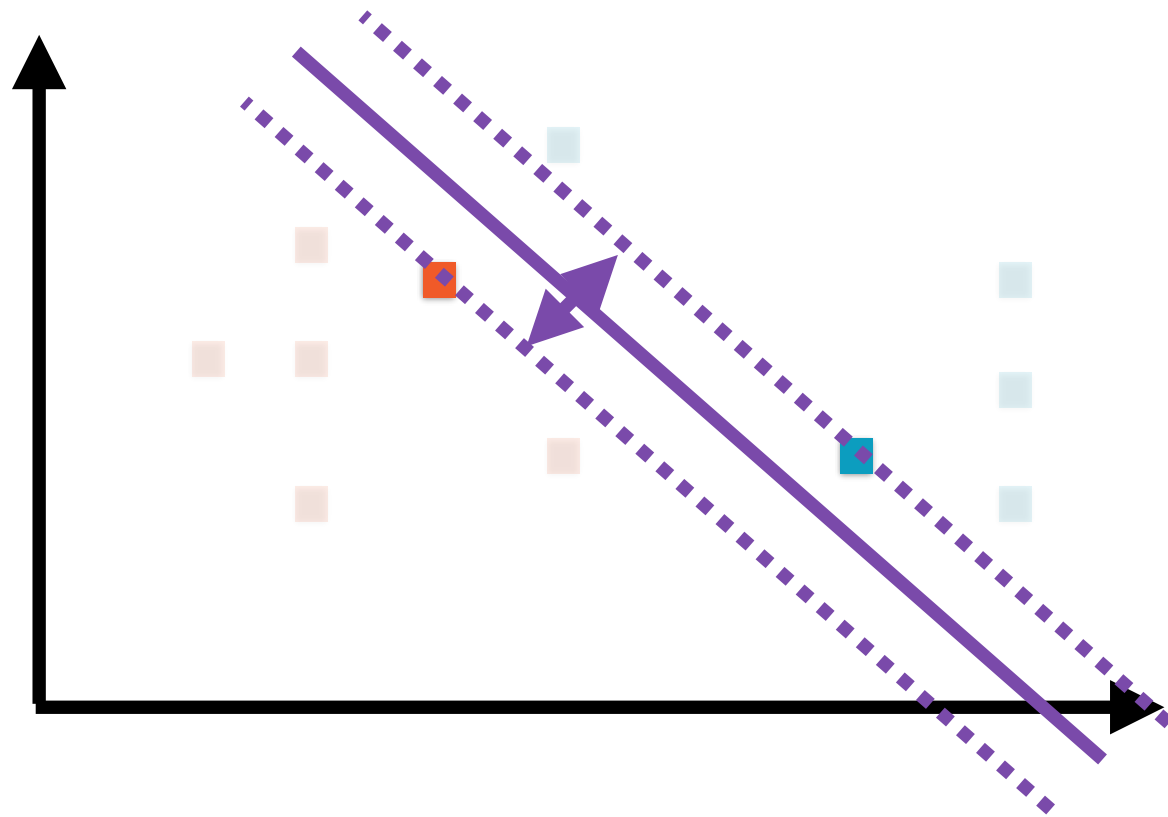
# Classification



SVM finds the widest street between the nearest points on either side

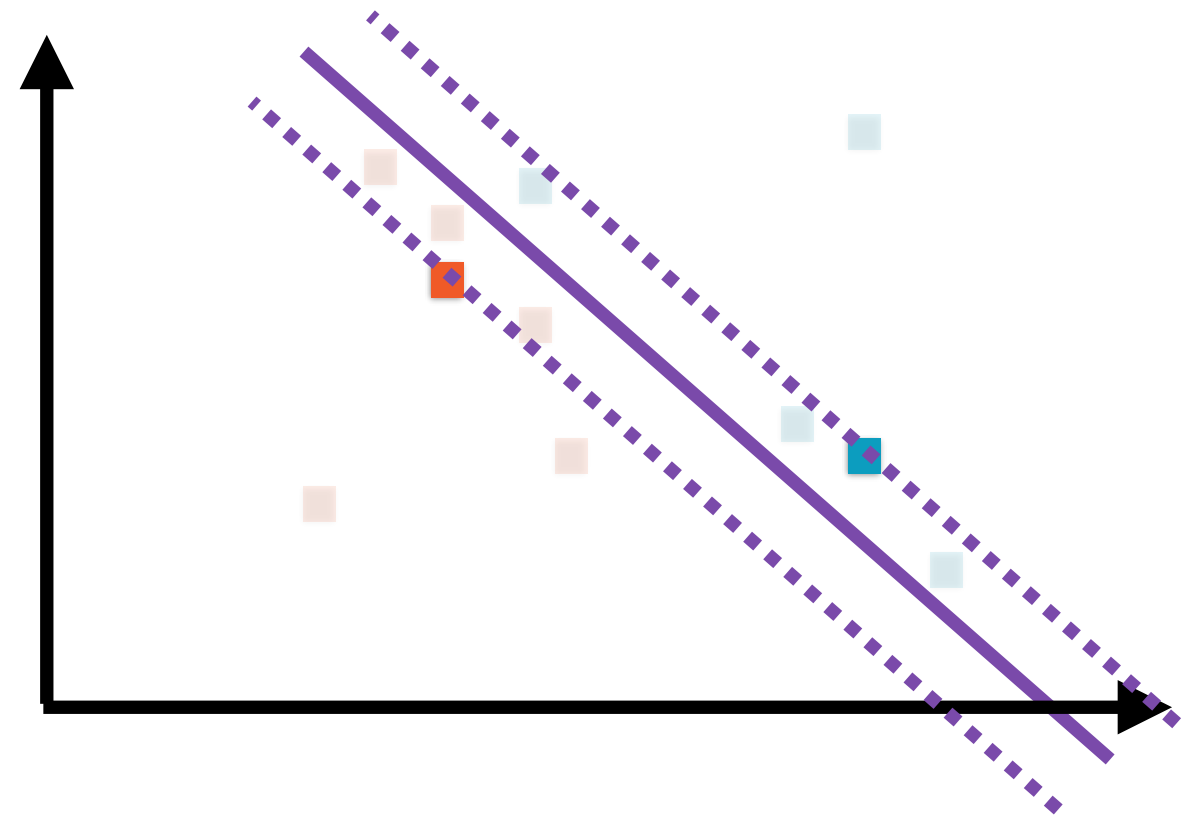
# Similar, yet Different

## SVM Classification



Find widest margin with most distance from nearest points (support vectors)

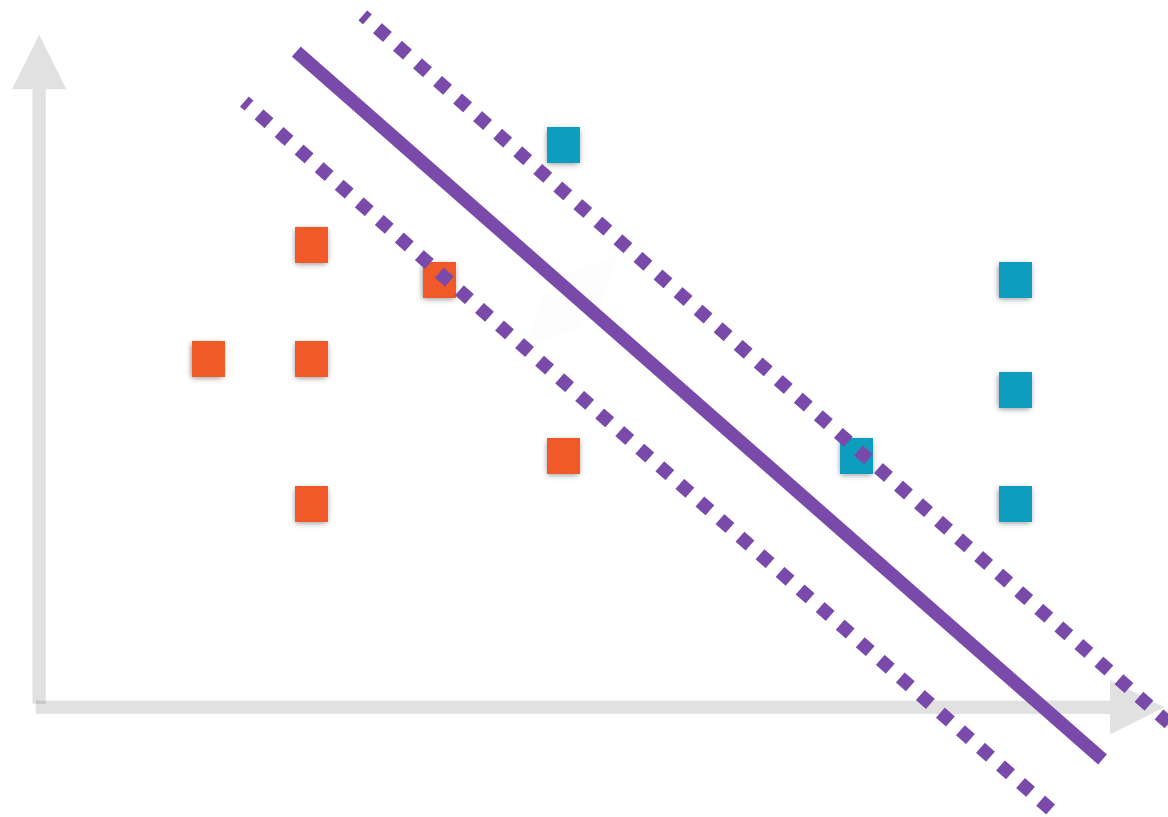
## SVM Regression



Find line that “best fits” the points

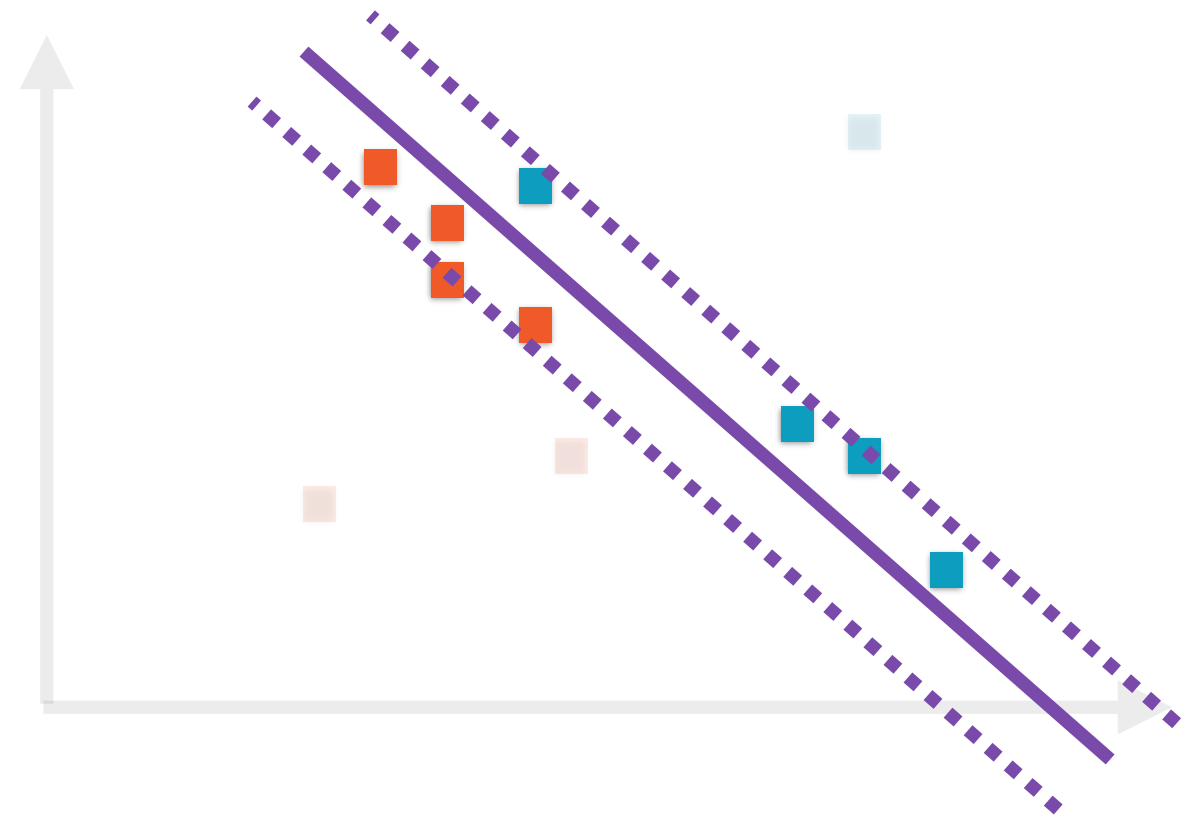
# Similar, yet Different

## SVM Classification



No points are inside the margin

## SVM Regression

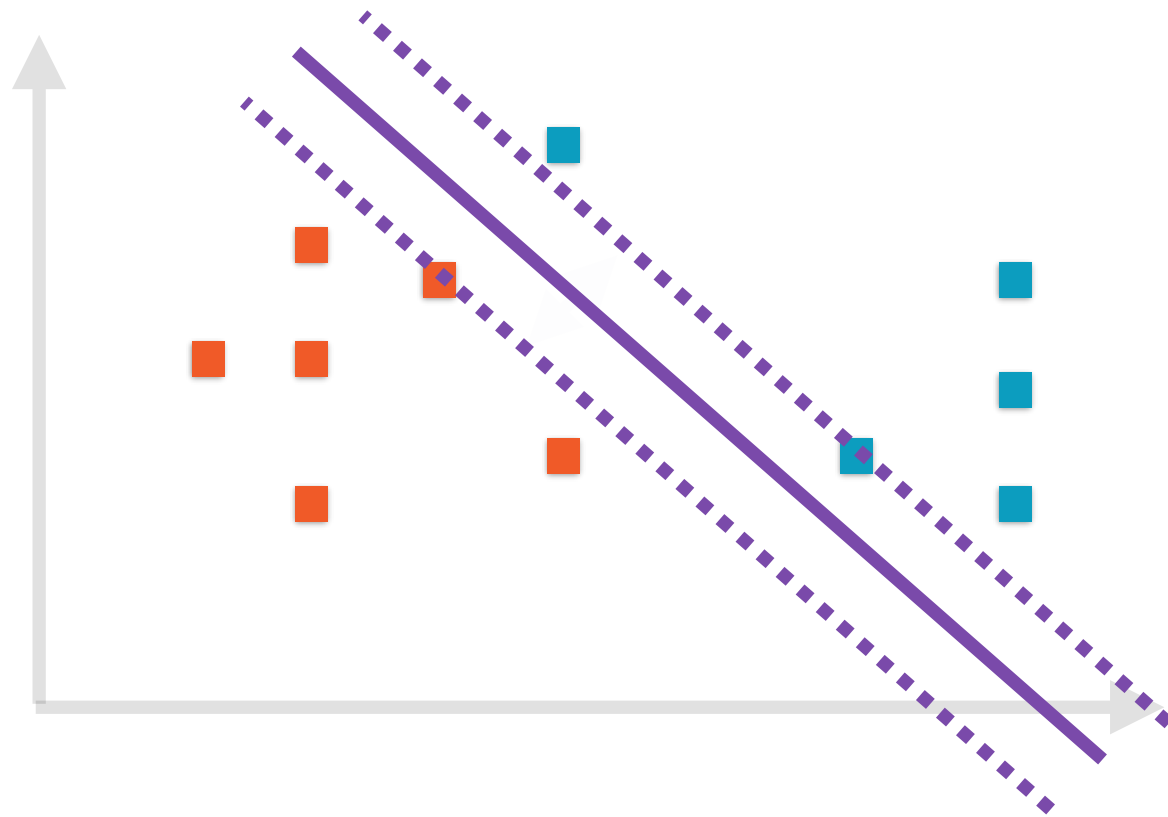


Seek to maximize the number of points inside the margin



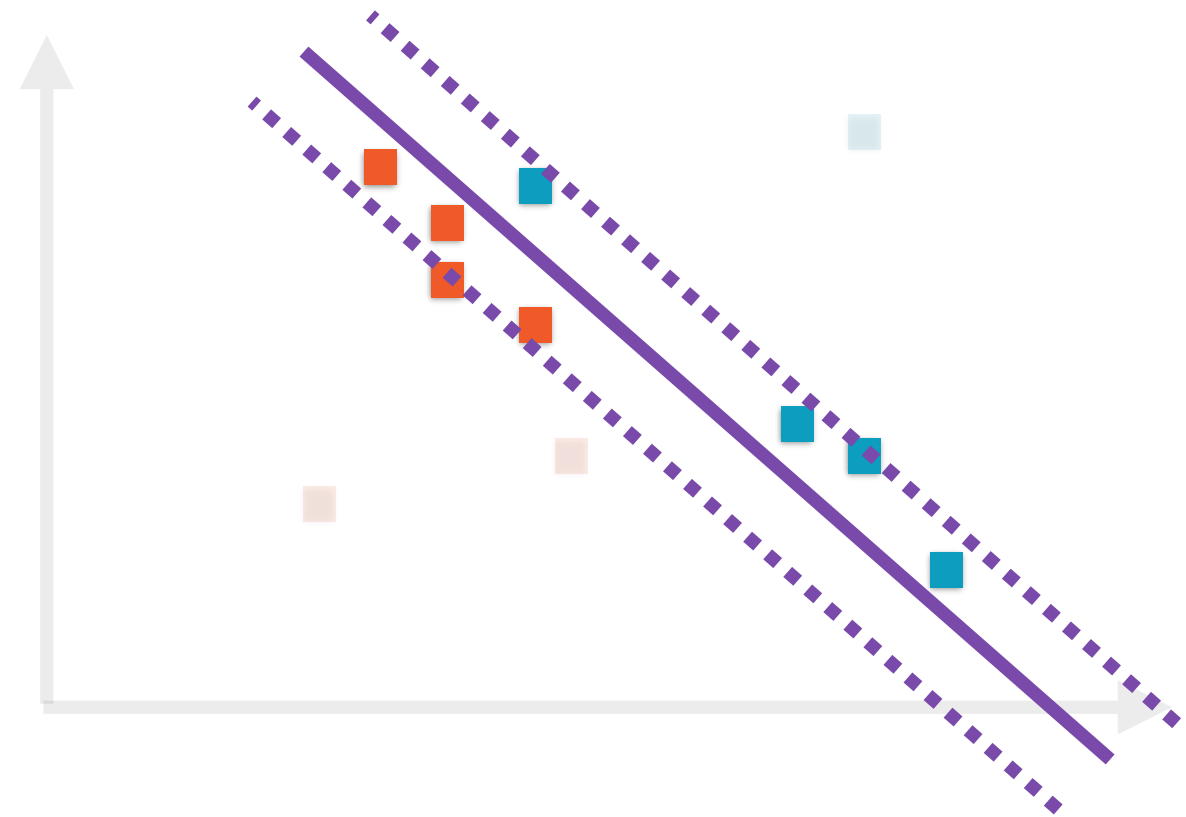
# Similar, yet Different

## SVM Classification



Points far from the margin are  
“good” (improve objective function value)

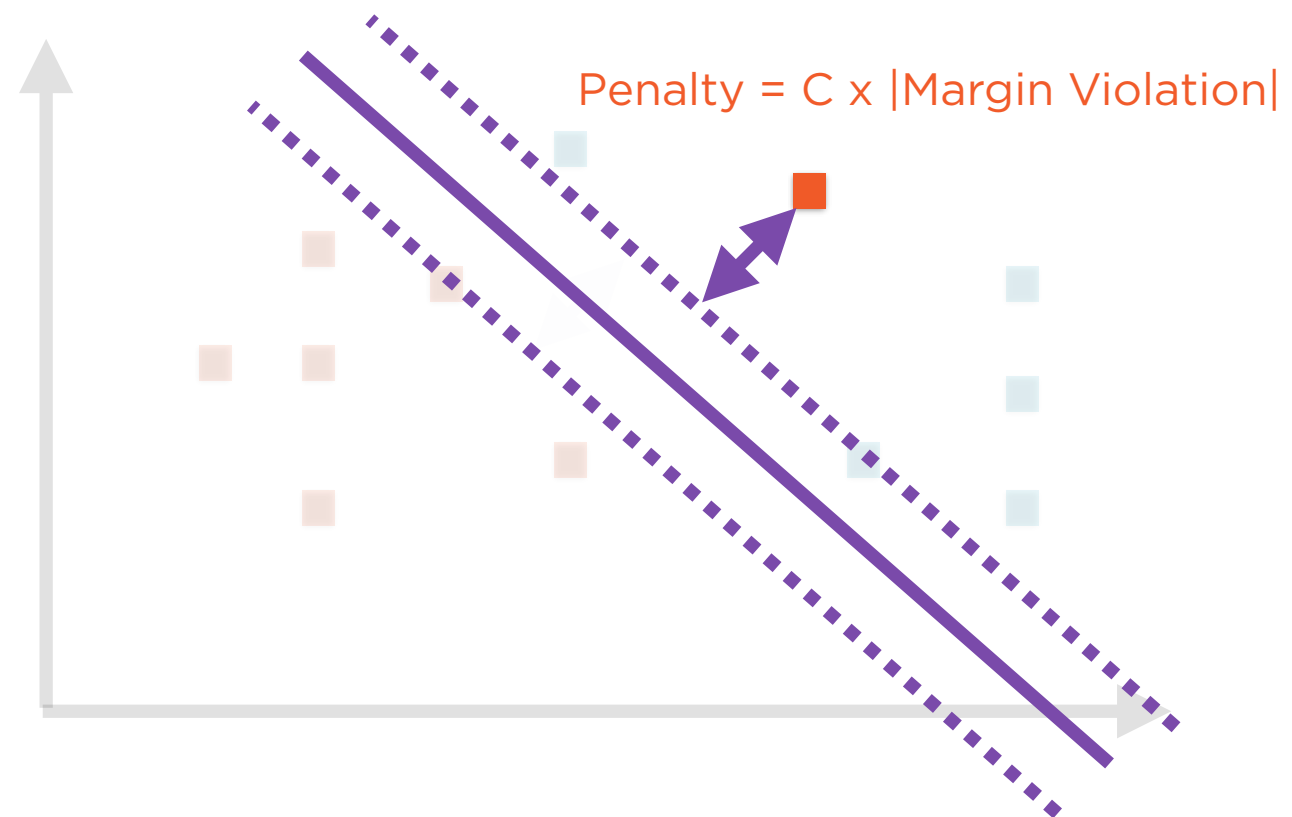
## SVM Regression



Points far from the margin are  
“bad” (worsen objective function value)

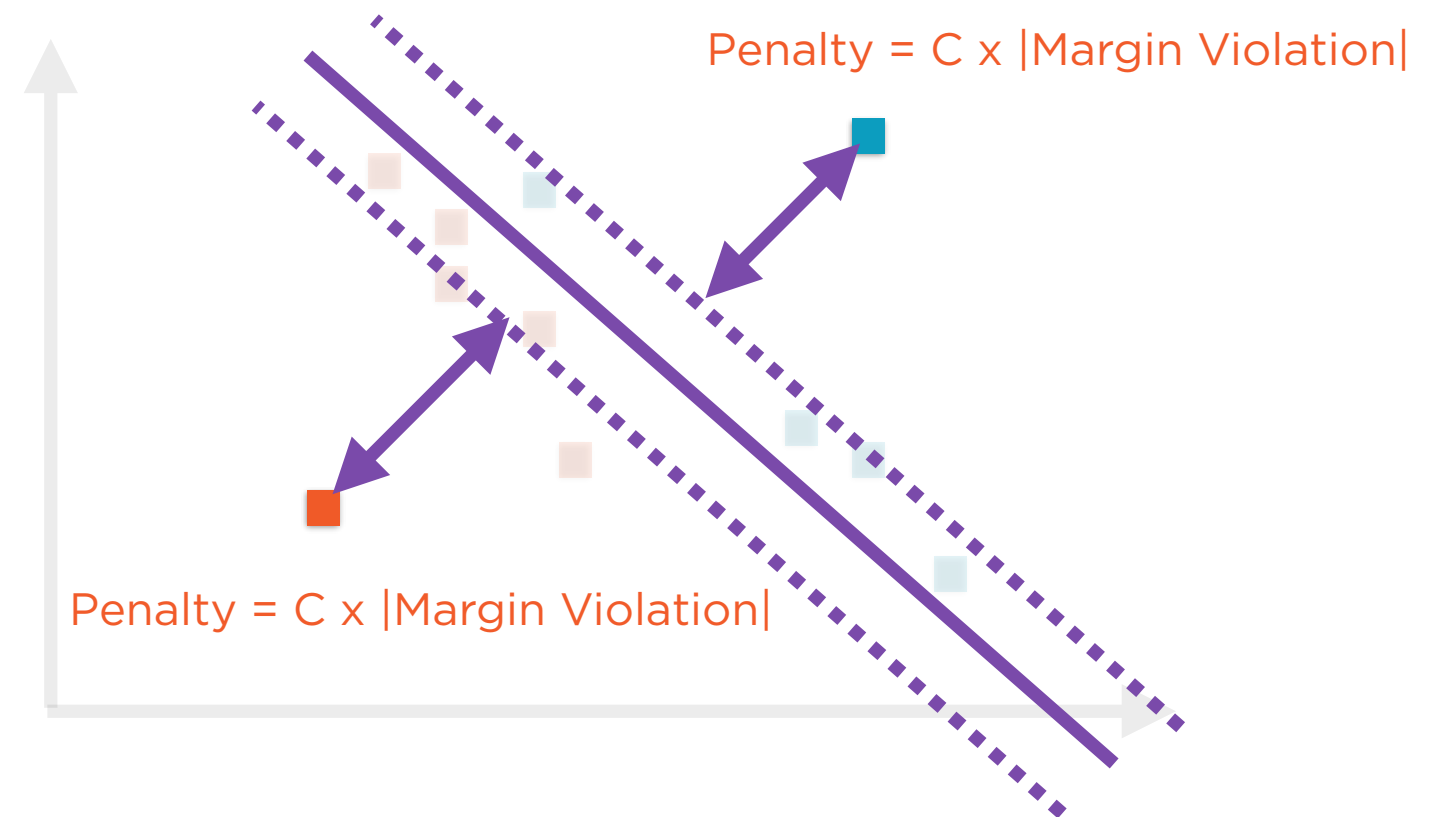
# Similar, yet Different

## SVM Classification



Outliers on “wrong” side of line are penalised

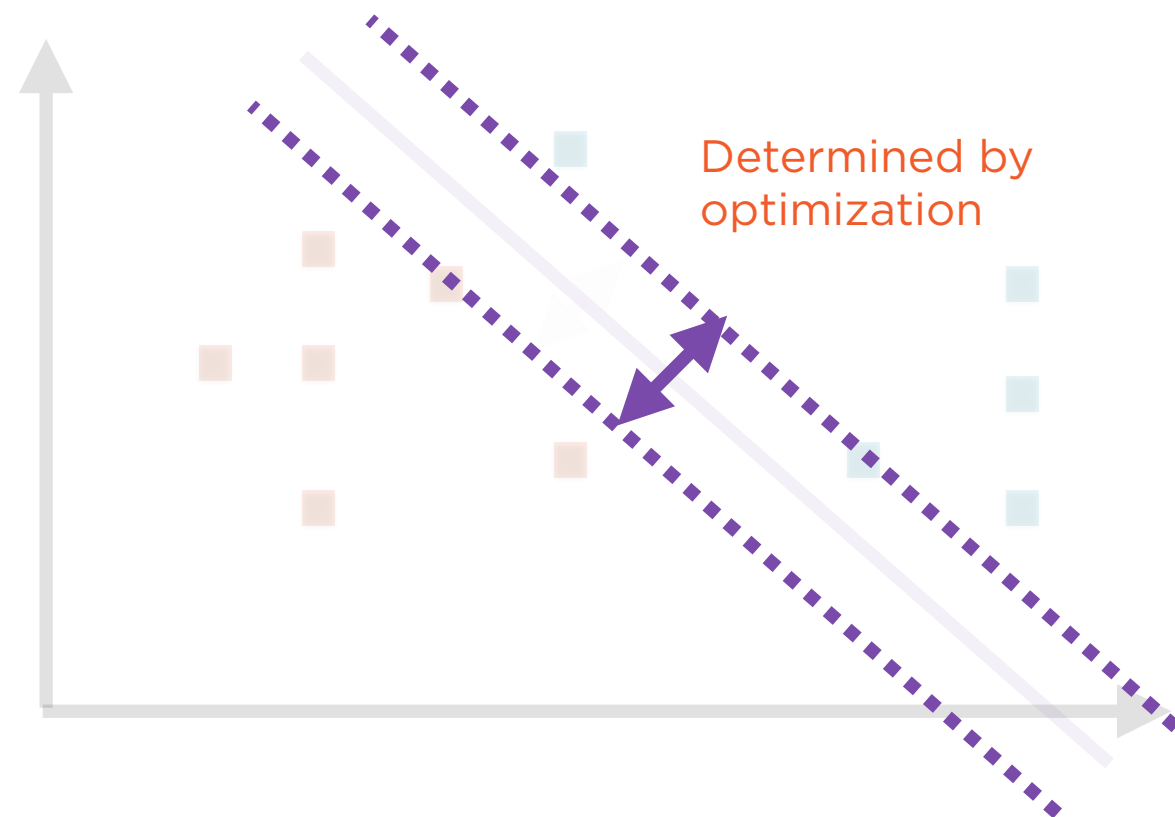
## SVM Regression



Points far from the margin are penalized

# Similar, yet Different

## SVM Classification



Width of margin found by optimizer (make as wide as possible)

## SVM Regression



Width of margin specified in model (requires another hyperparameter  $\varepsilon$ )

Demo

**Implementing Support Vector  
Regression in scikit-learn**

# Summary

**Regression models and  $R^2$  for measuring model fit**

**The bias-variance trade-off and overfitted models**

**Lasso and Ridge regression to mitigate overfitting**

**Support vector regression models**