Building SVM and Gradient Boosting Models in scikit-learn

Overview

Support Vector Machines are a very popular ML technique for classification

SVMs can work on text as well as images

Often ML models can come together as an ensemble to build a stronger model

Gradient boosting uses decision trees to build a better regression model

SVM Classification

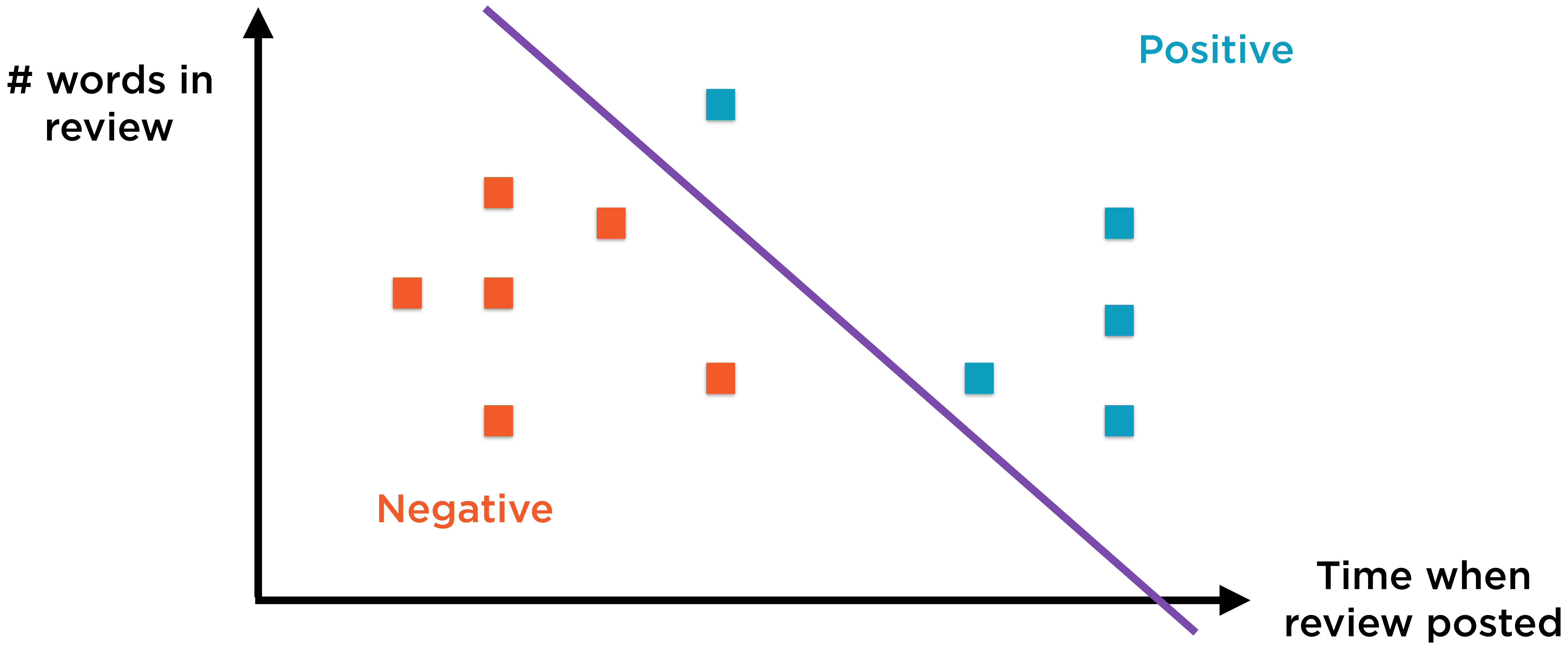
Data in One Dimension

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

Data in One Dimension

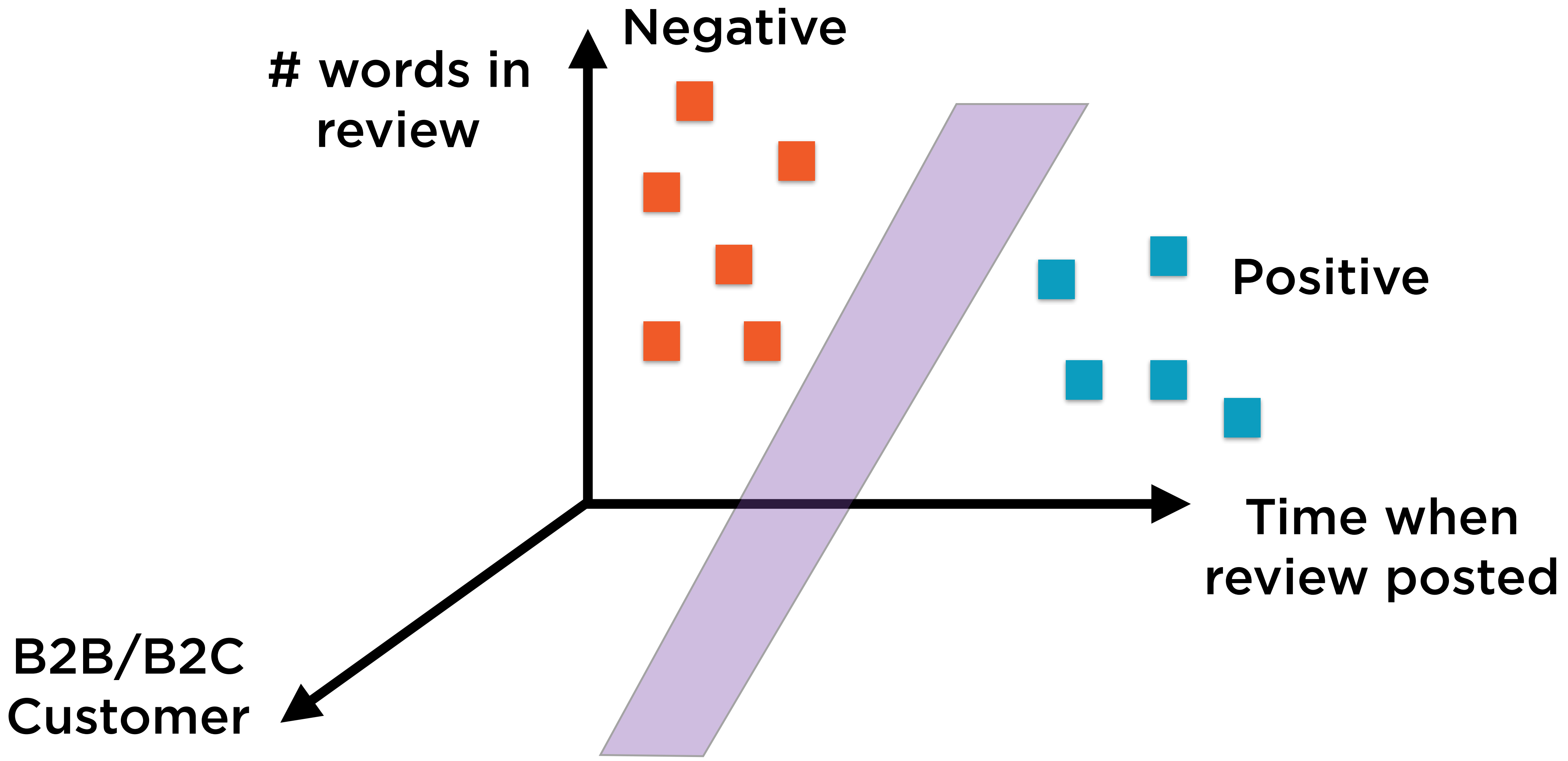
Unidimensional can also be separated, or classified, using a point

Data in Two Dimensions



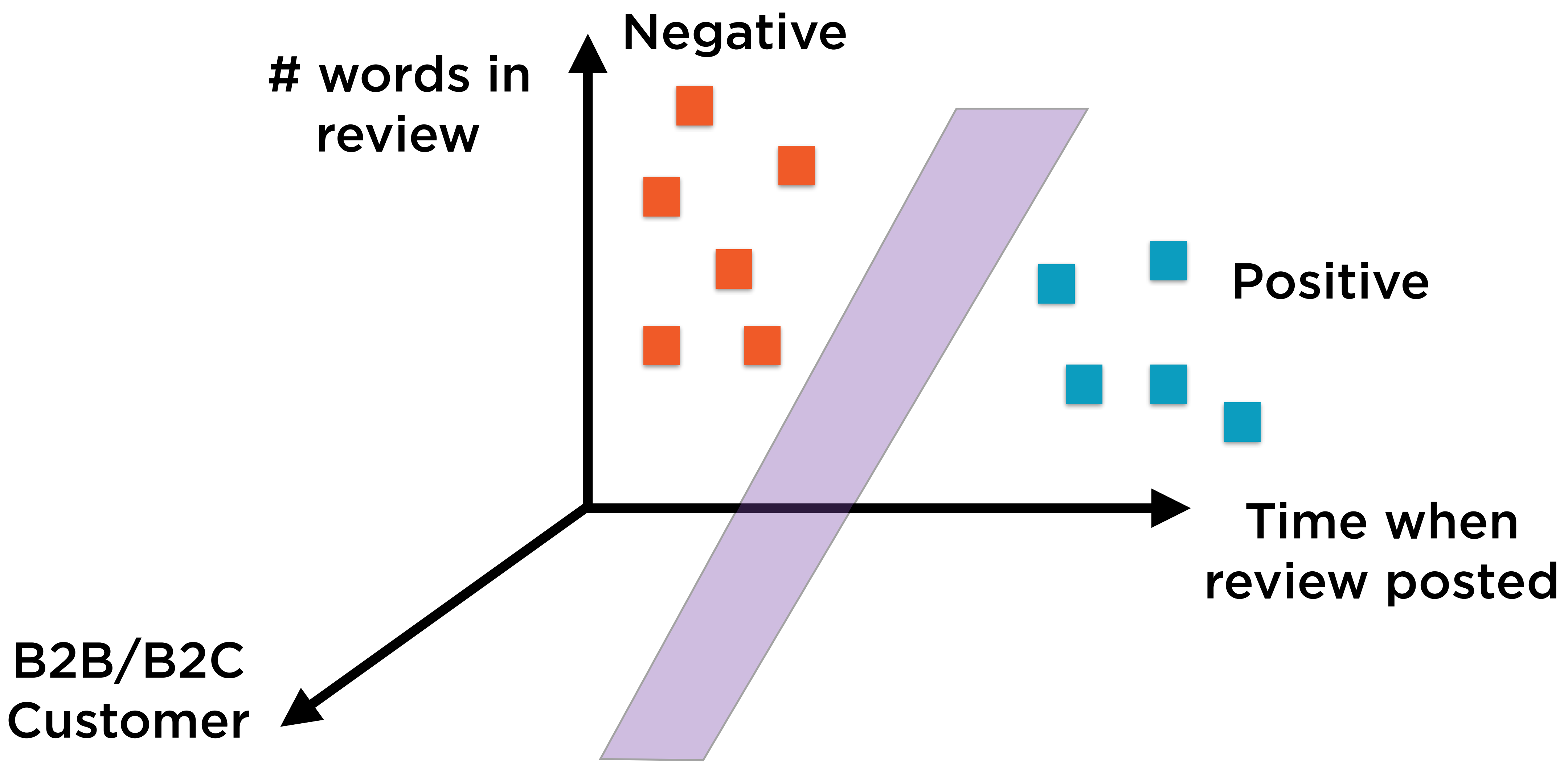
Bidimensional data points can be represented using a plane, and classified using a line

Data in N Dimensions



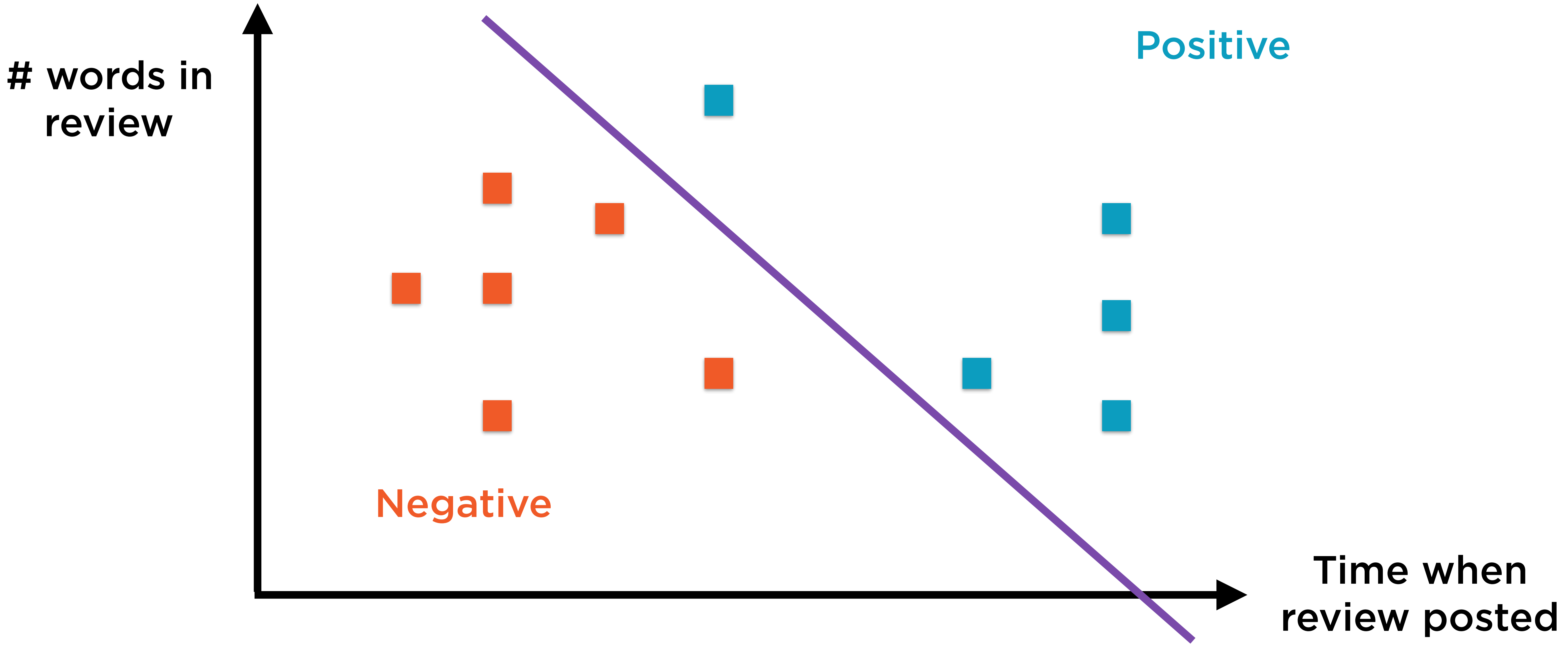
N-dimensional data can be represented in a **hypercube**, and classified using a **hyperplane**

Support Vector Machines



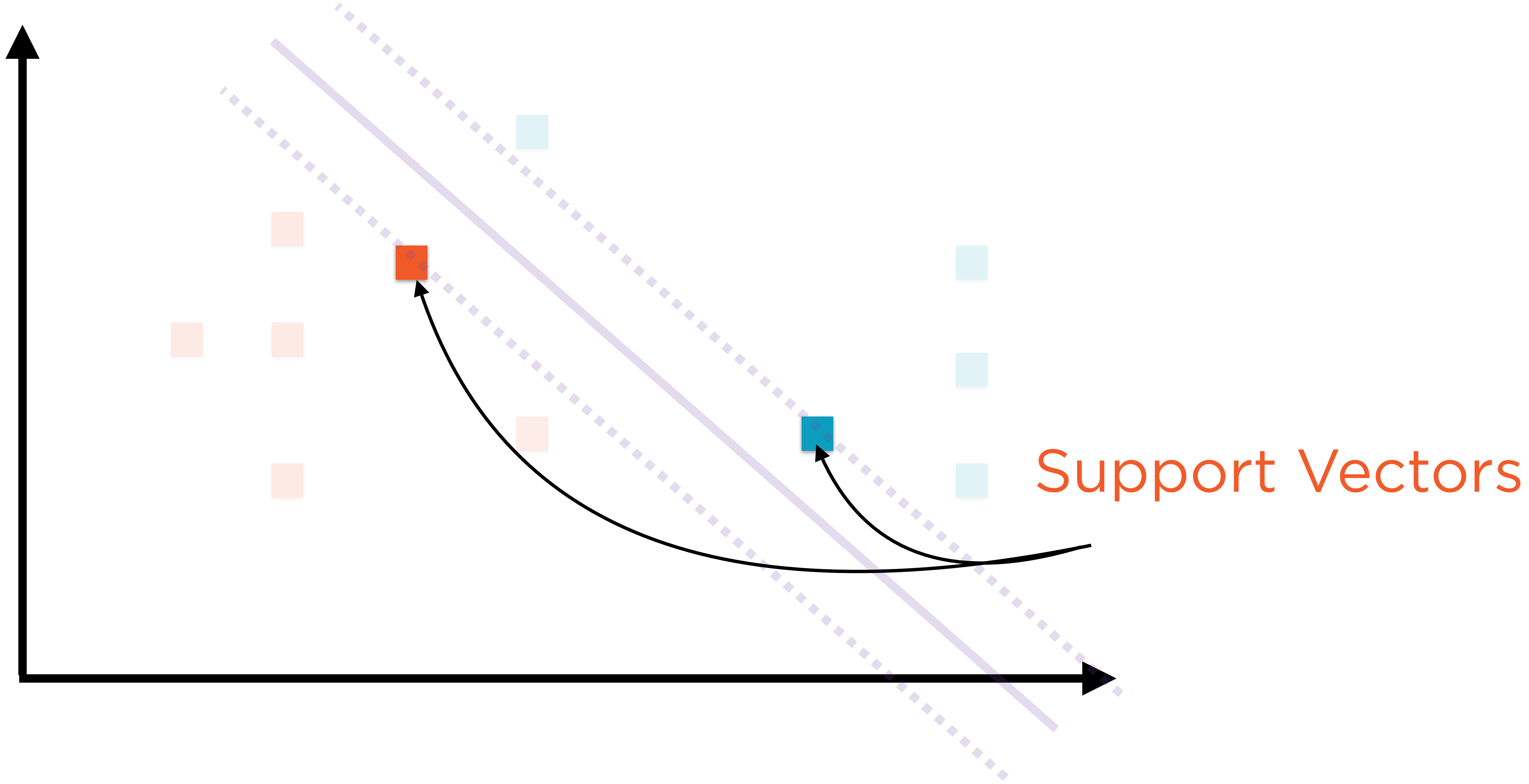
SVM classifiers find the hyperplane that best separates points in a hypercube

Hard Margin Classification



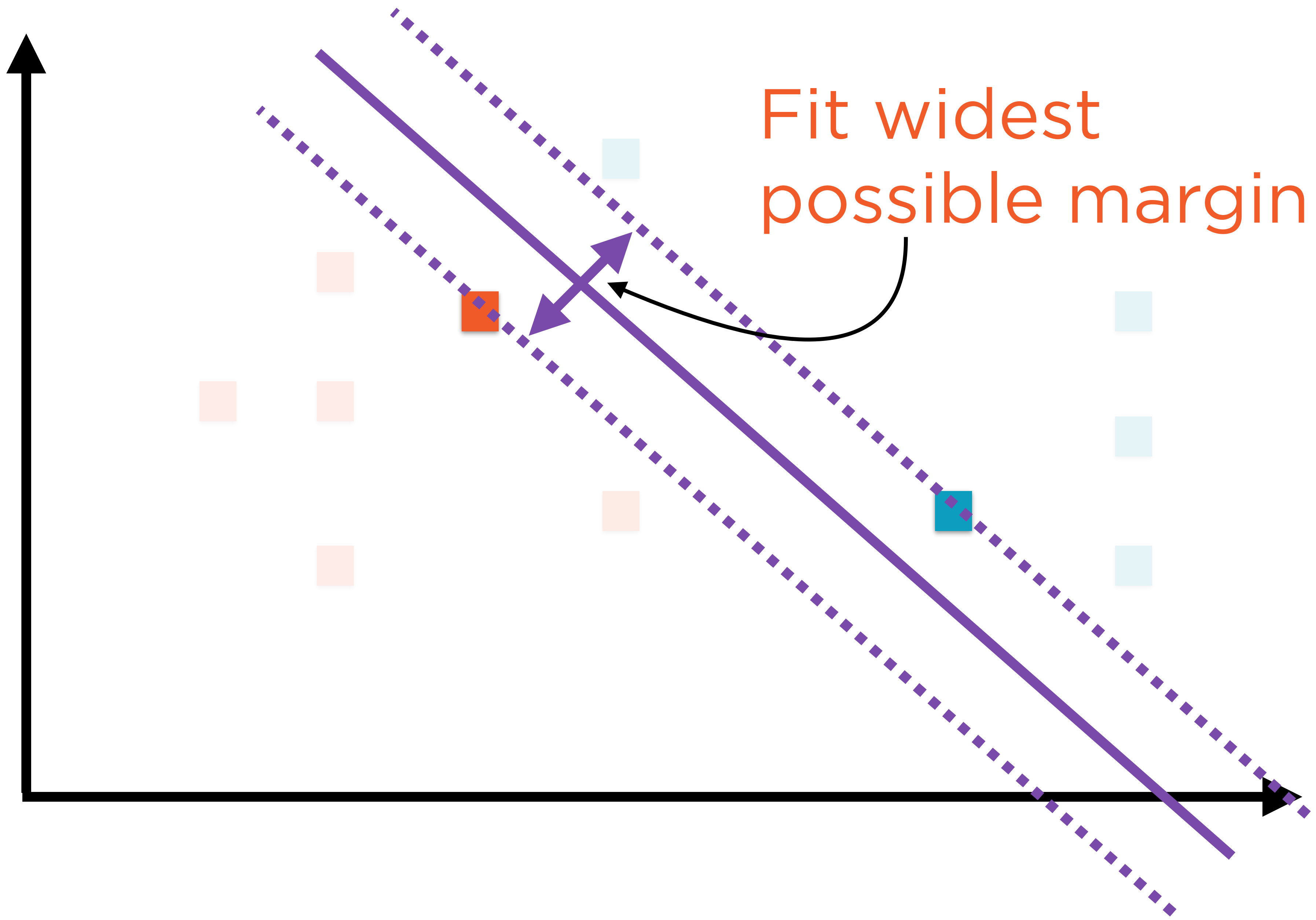
Ideally, data is linearly separable - hard decision boundary

Hard Margin Classification



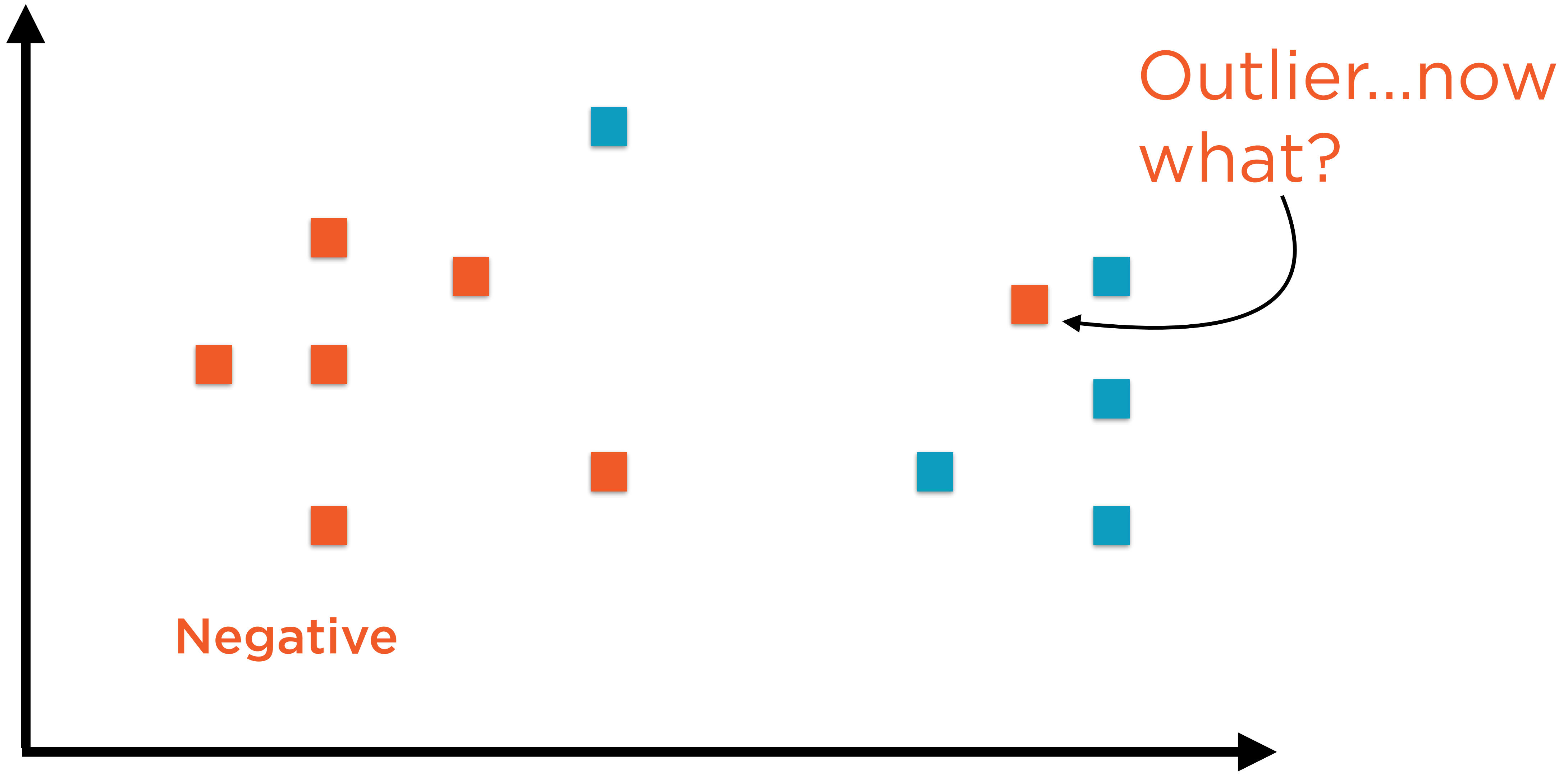
The nearest instances on either side of the boundary are called the support vectors

Hard Margin Classification



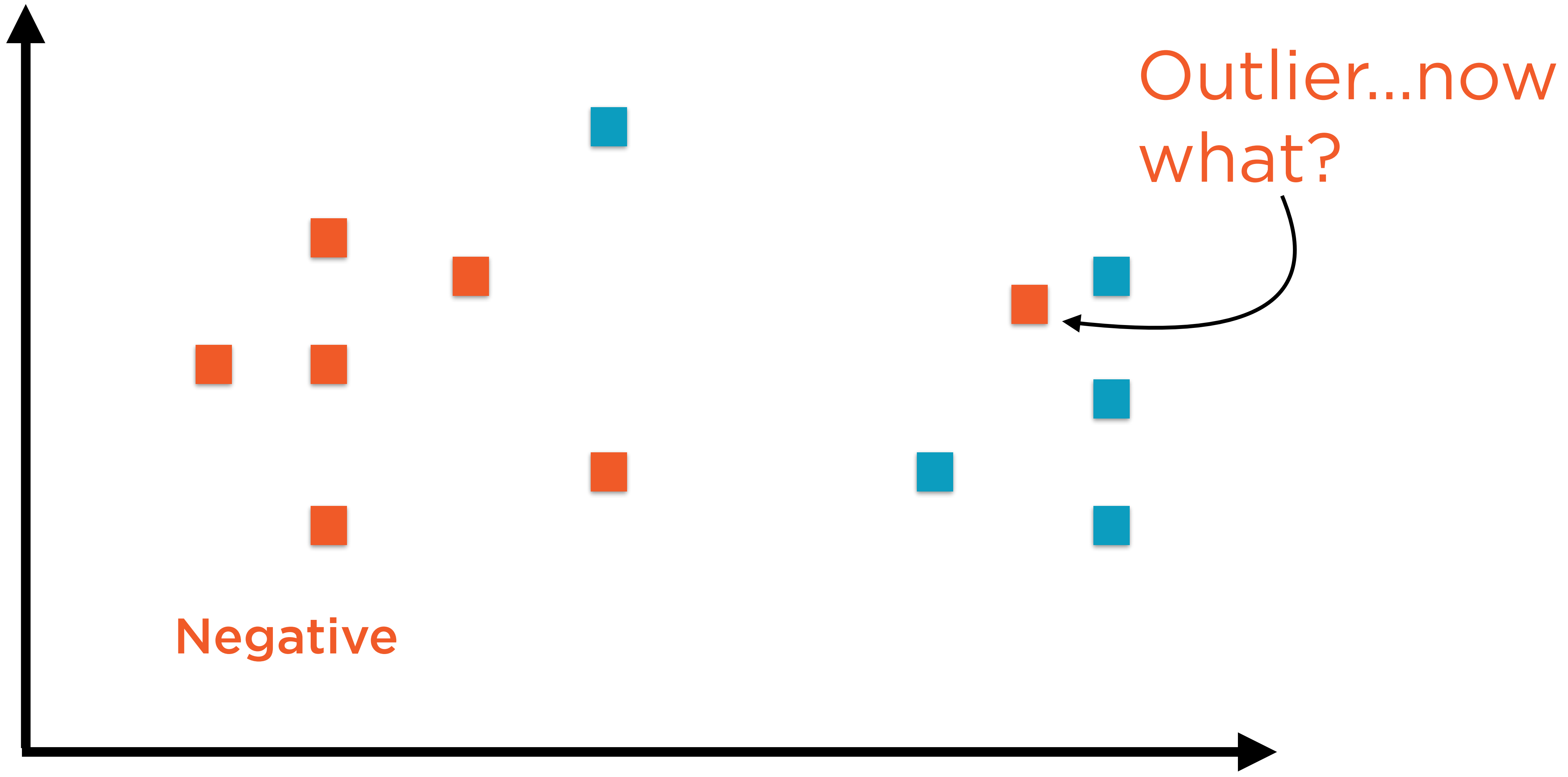
SVM finds the widest street between the nearest points on either side

Soft Margin Classification



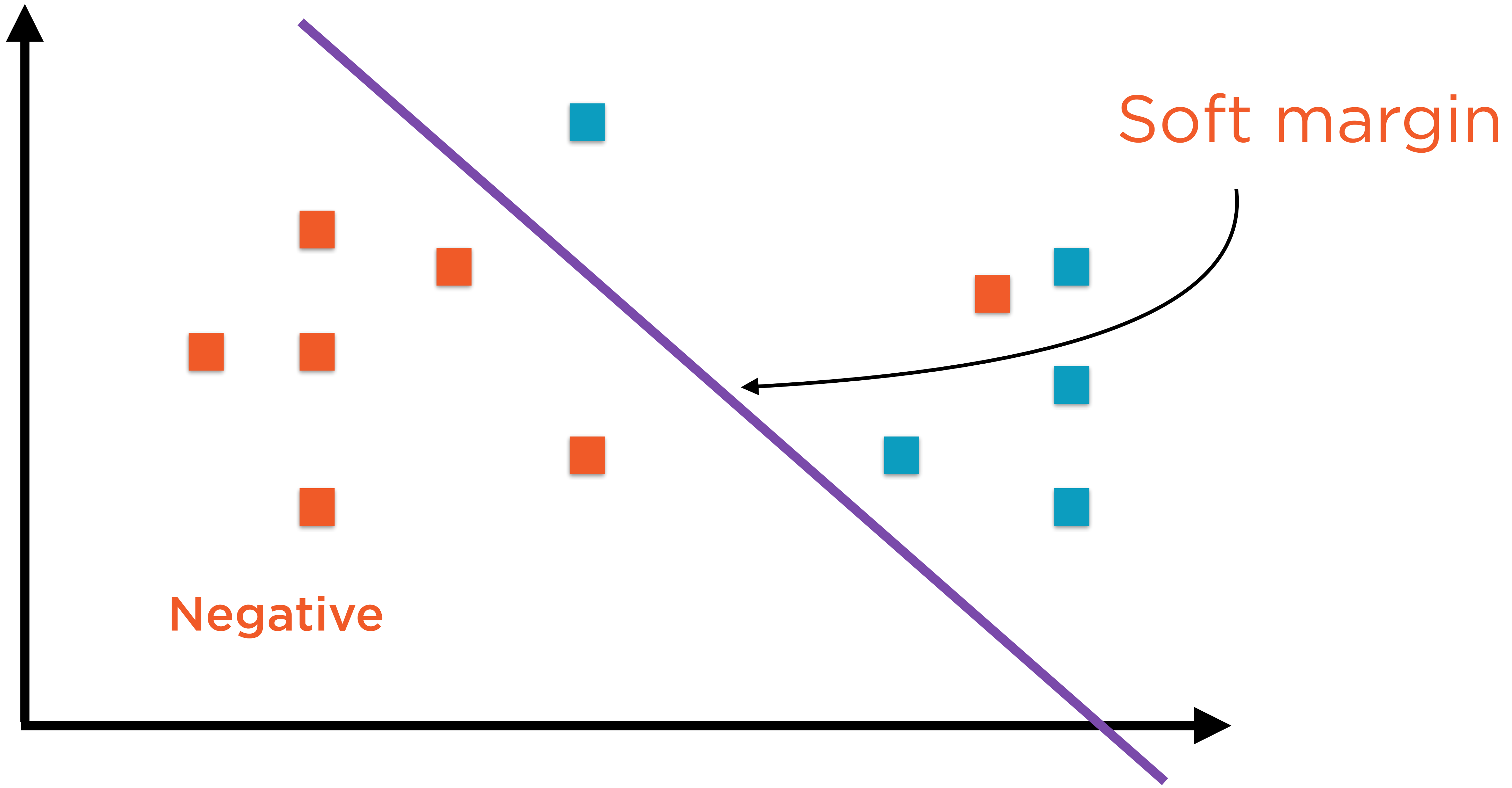
Hard margin classifiers are sensitive to outliers...

Soft Margin Classification



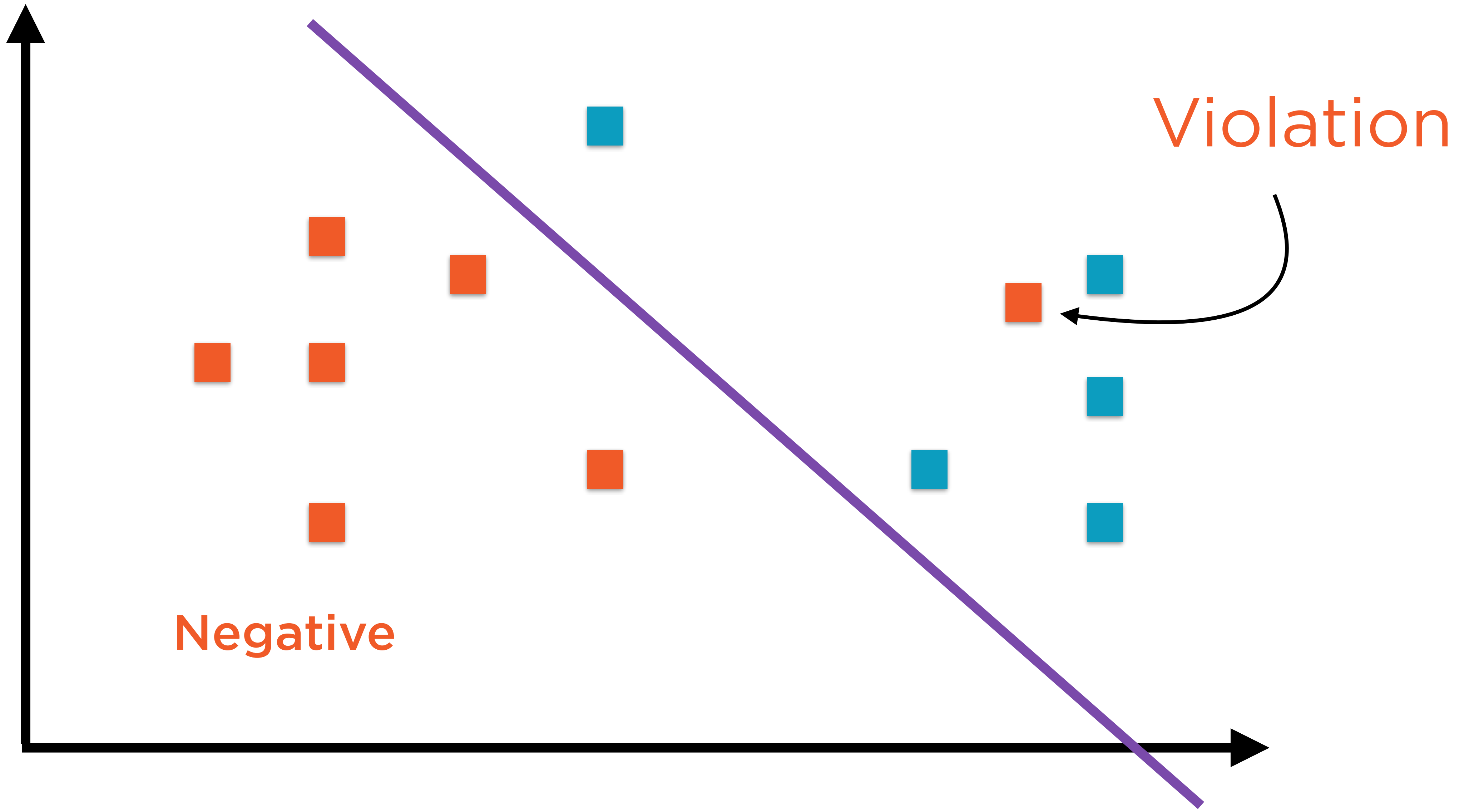
...and require perfectly linear separability in data

Soft Margin Classification



Soft margin classifiers allow some violations of the decision boundary

Soft Margin Classification



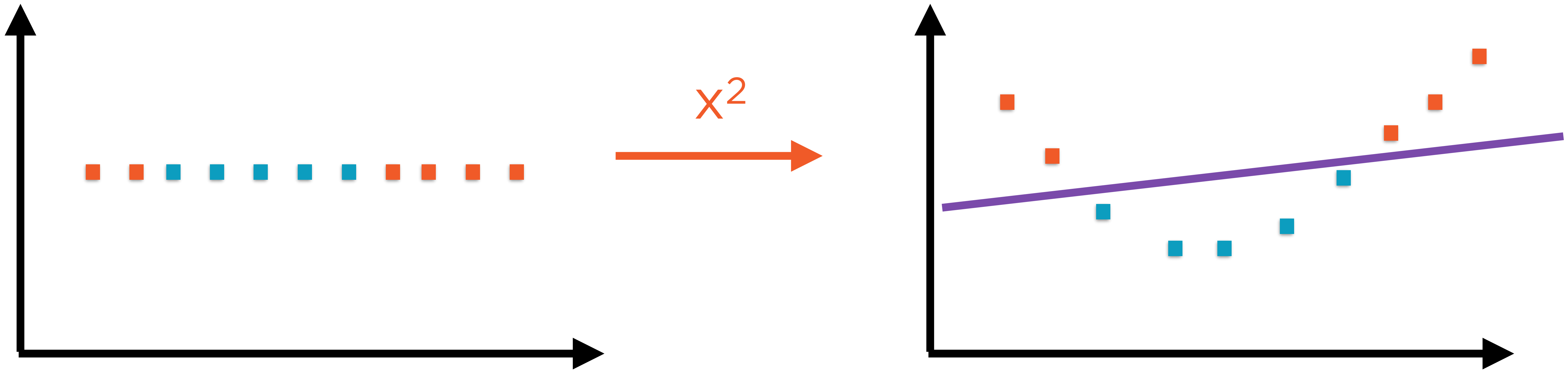
Soft margin classifiers allow some violations of the decision boundary

Non-separable Data

Smart transformations resolve surprisingly many such cases

SVM classification can be extended to almost any data using something called the kernel trick

Nonlinear SVM



Original Data

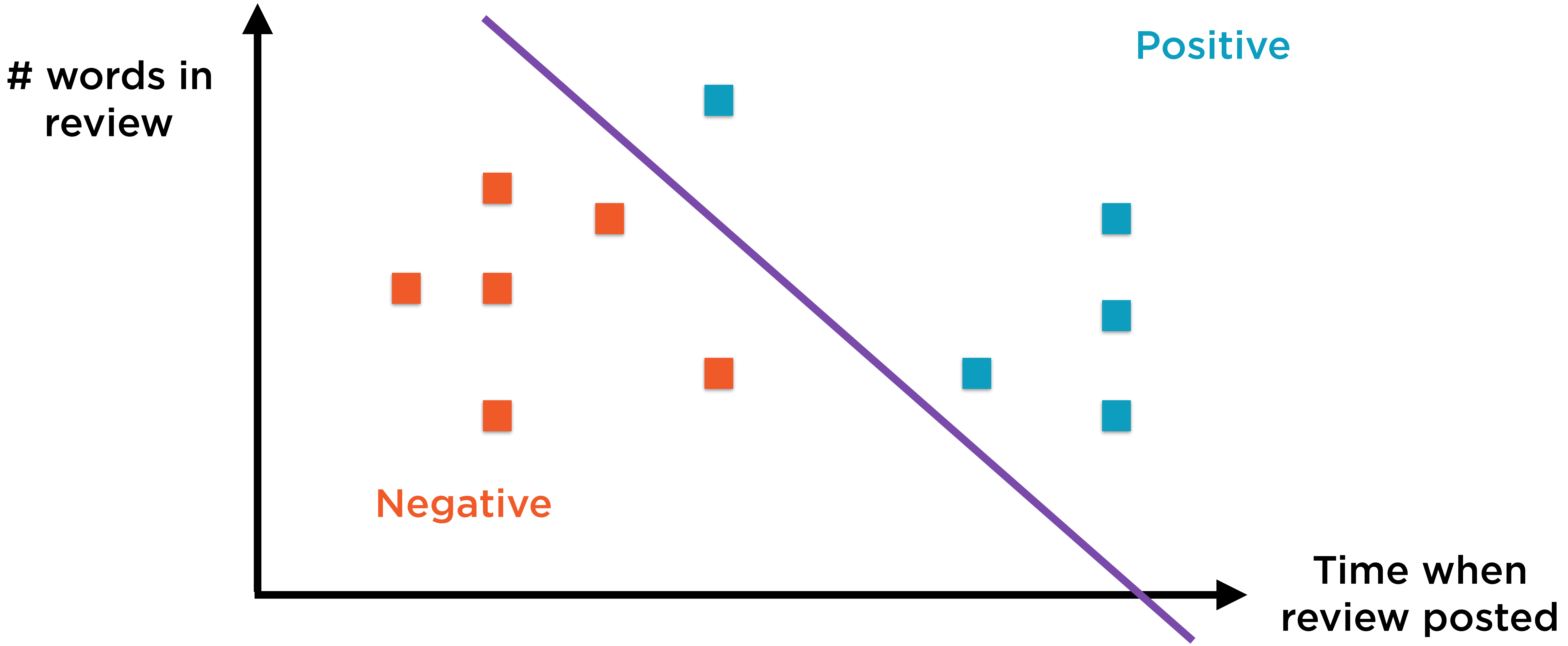
Not linearly separable

Square of original data

Now linearly separable!

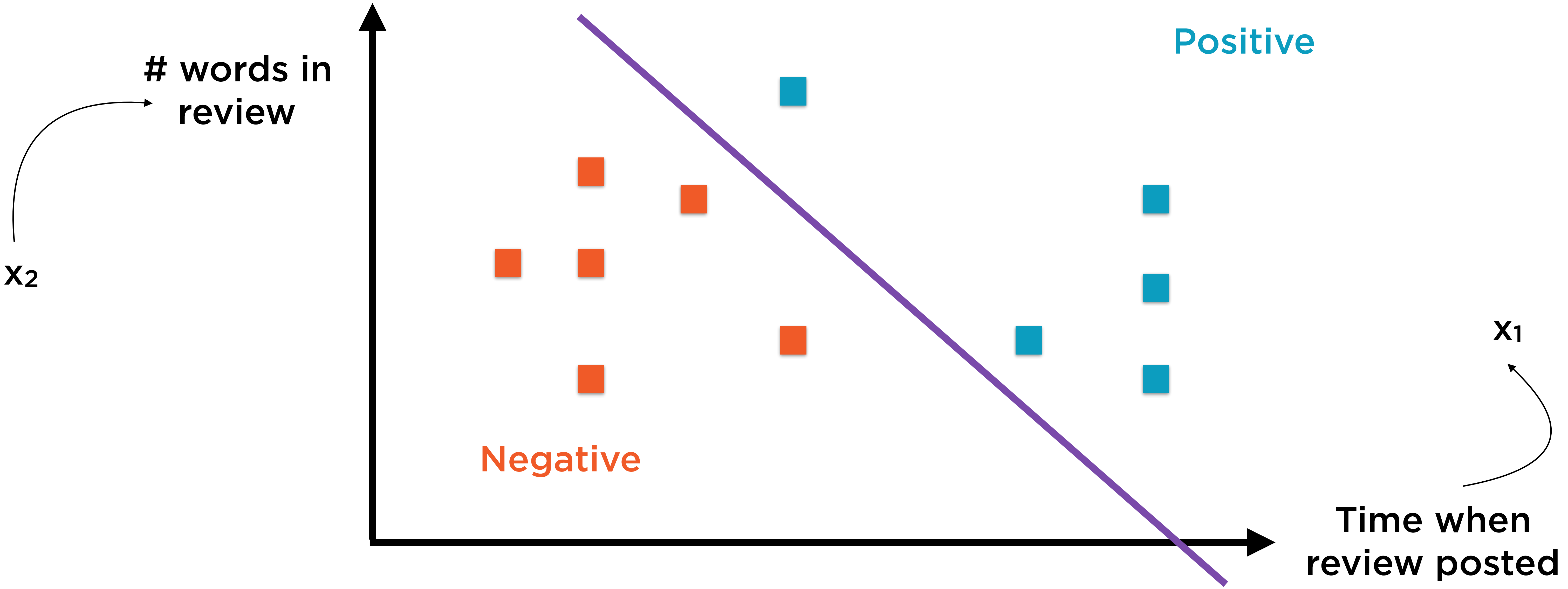
Setting Up the SVM Classification Problem

Margin Classification



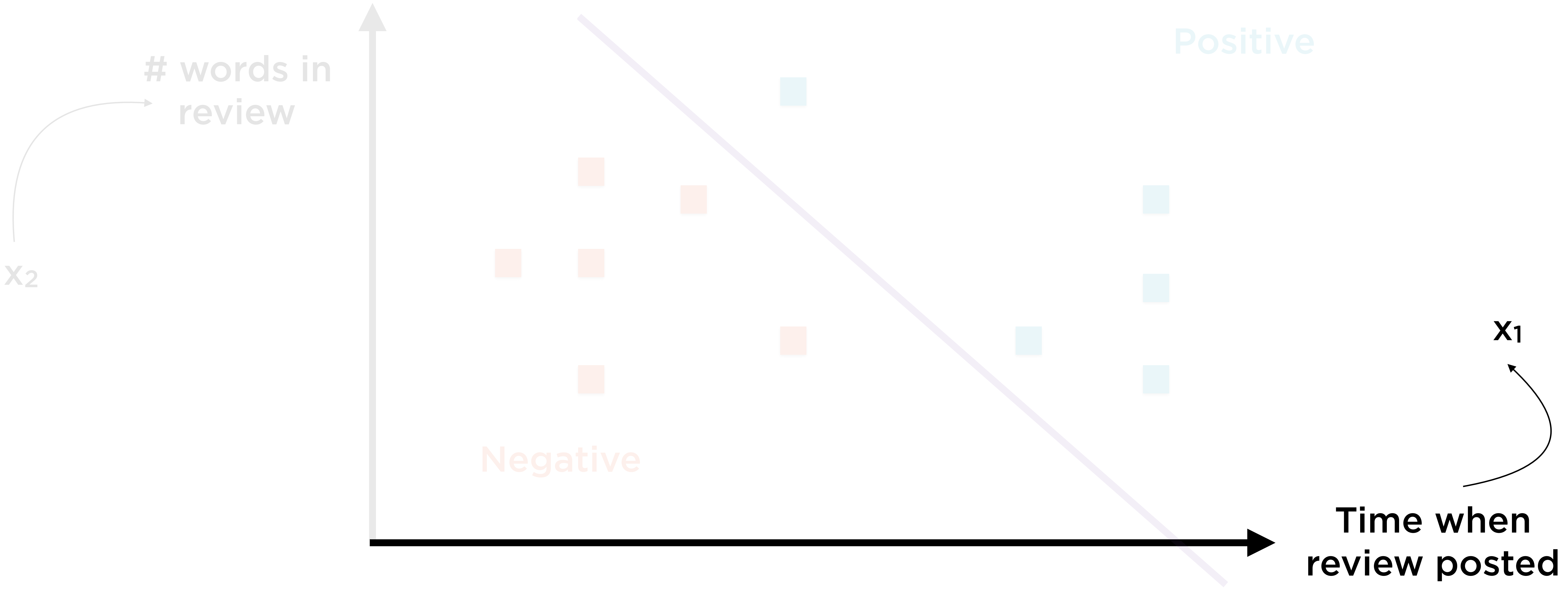
Classify review as positive or negative based on length of review, and time when posted

Margin Classification



Margin Classification

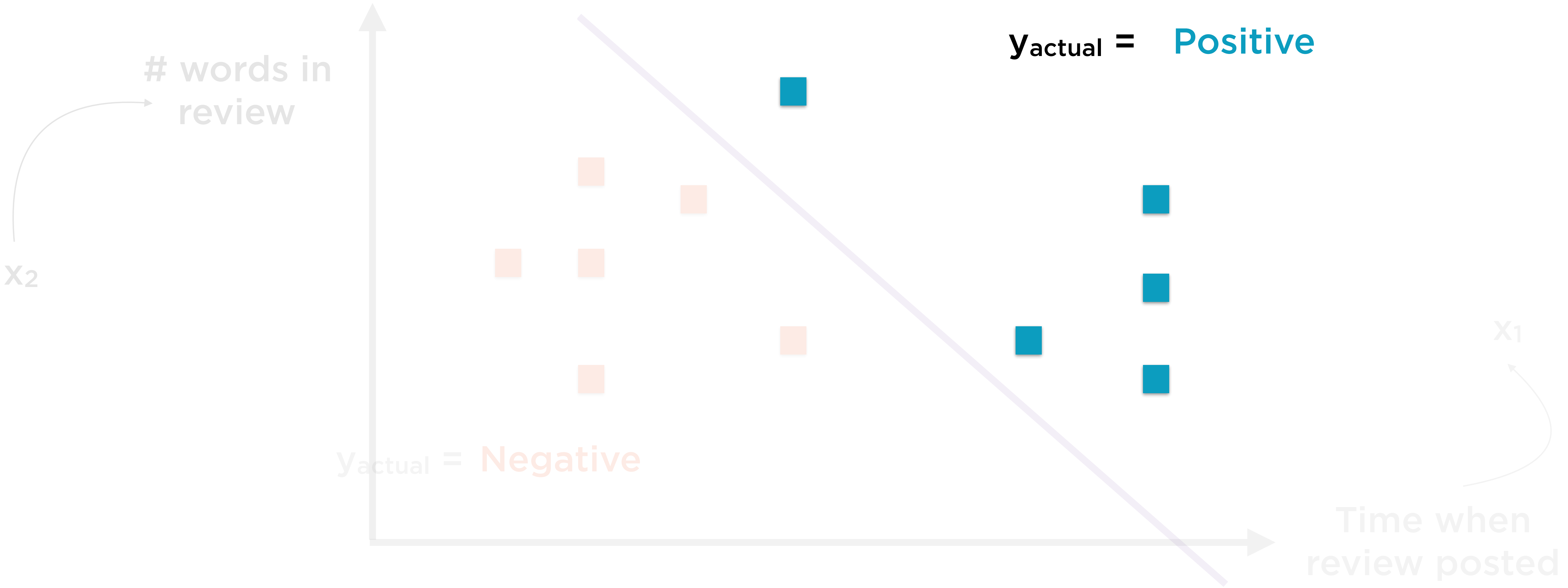
Margin Classification



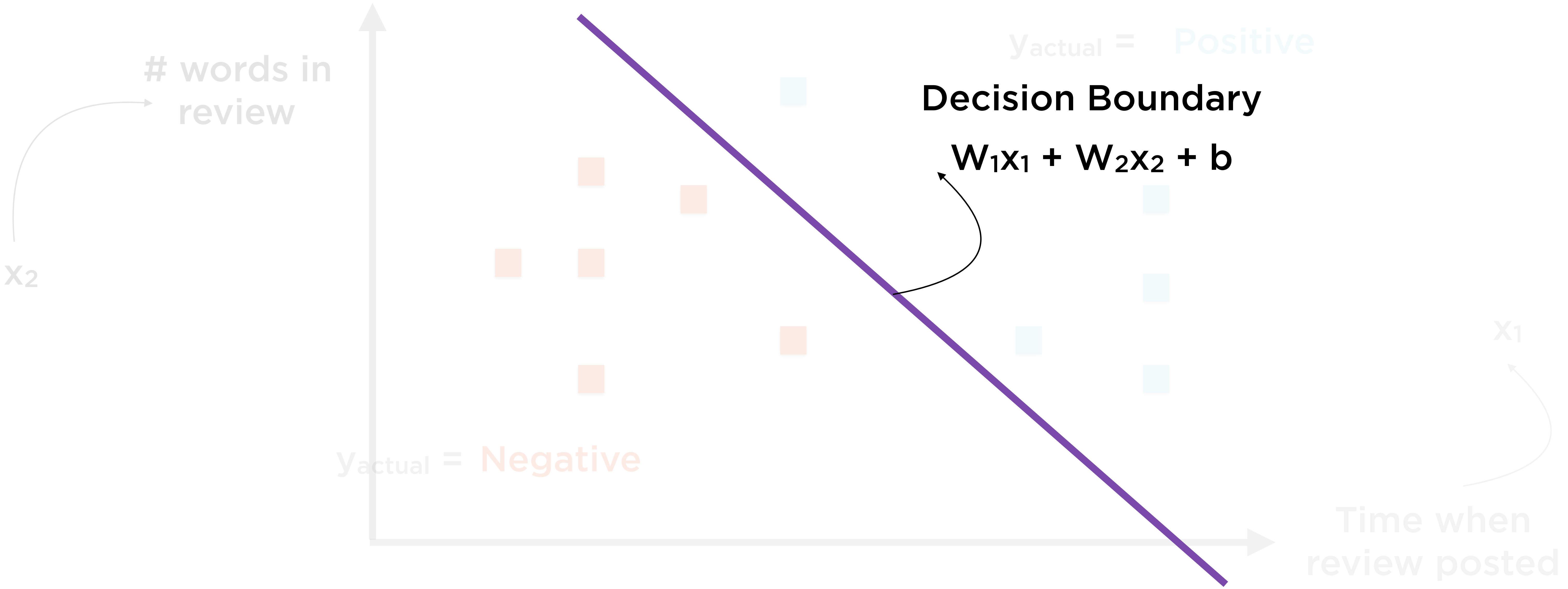
Margin Classification

y_{actual} = Negative

Margin Classification

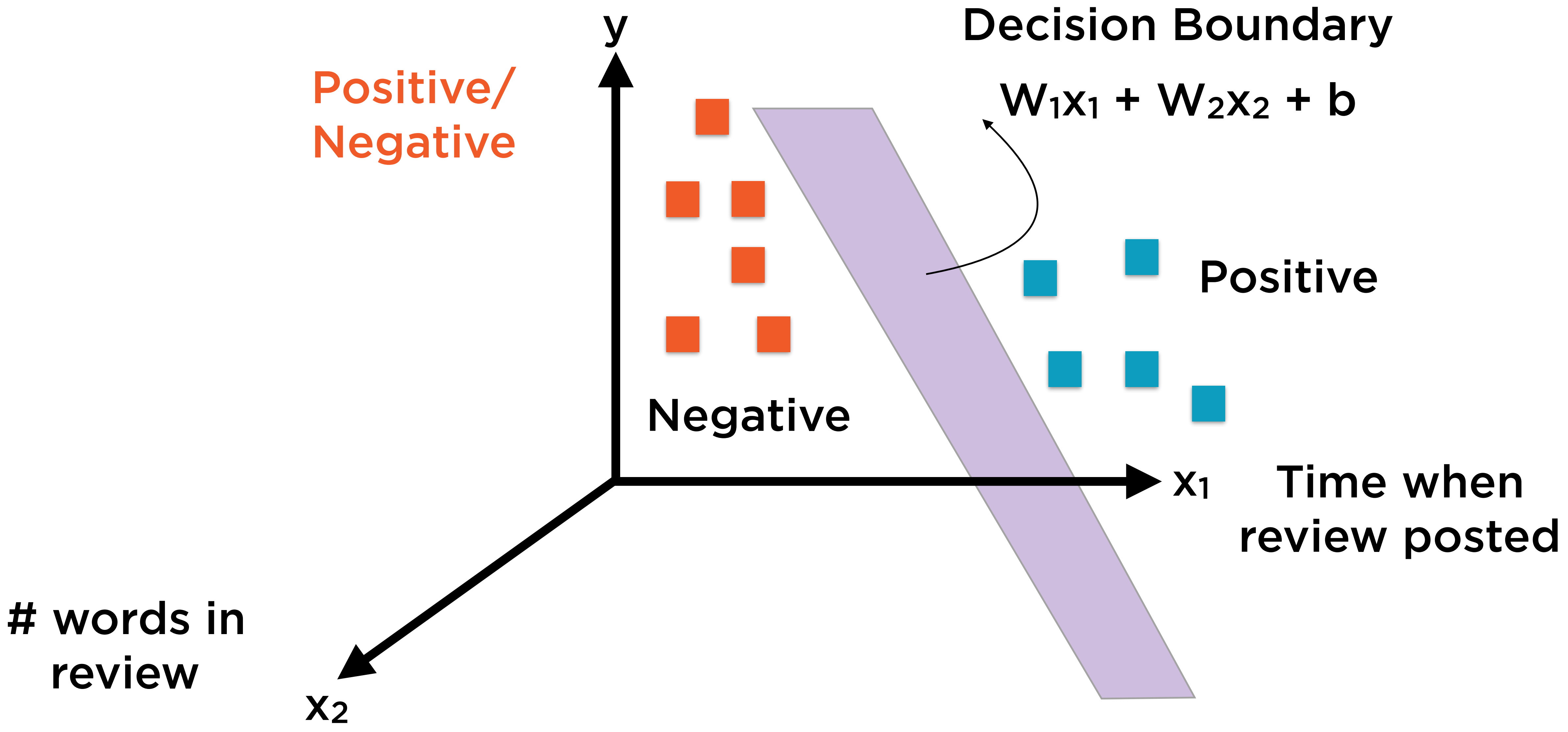


Margin Classification



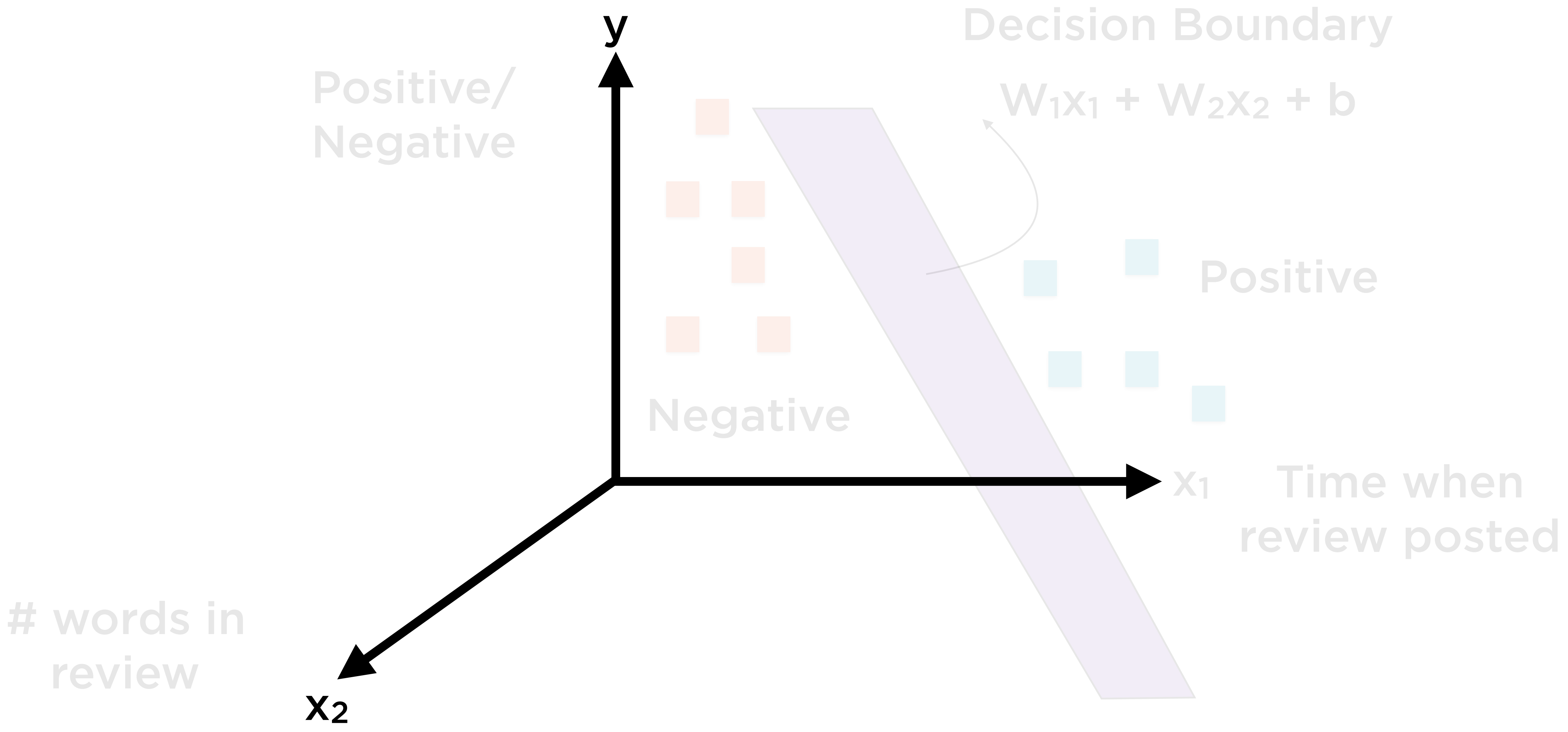
Actually, we need three dimensions to visualize decision boundary correctly

Margin Classification



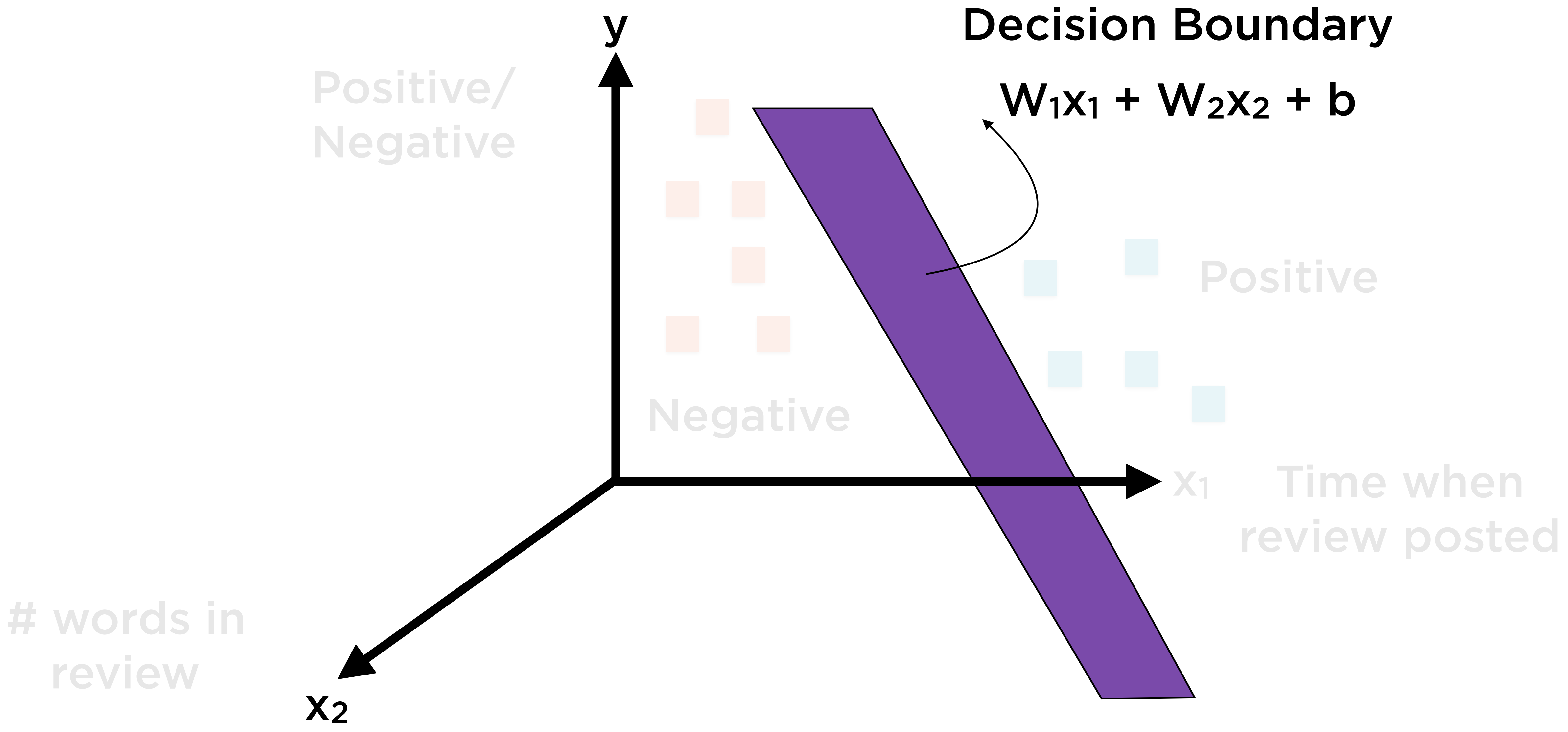
Actually, we need three dimensions to visualize decision boundary correctly

Margin Classification



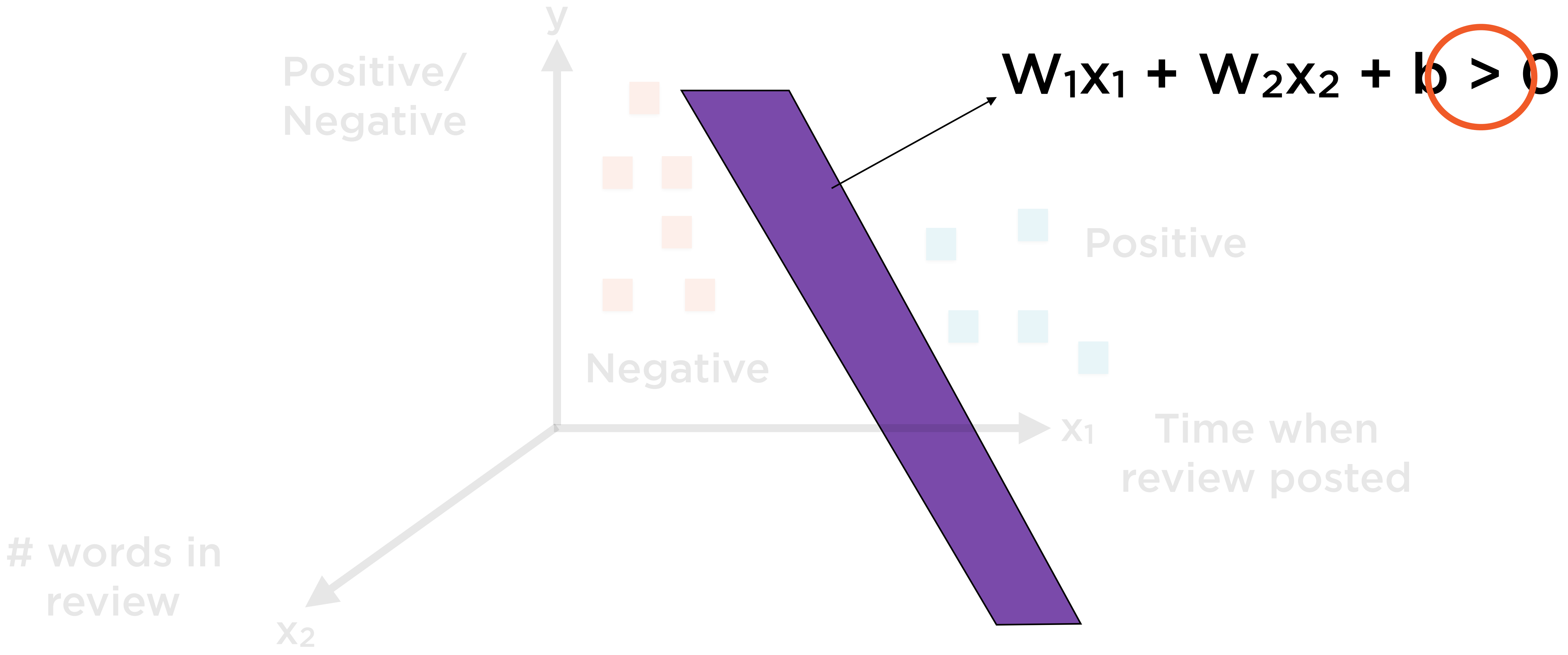
Actually, we need three dimensions to visualize decision boundary correctly

Margin Classification



Actually, we need three dimensions to visualize decision boundary correctly

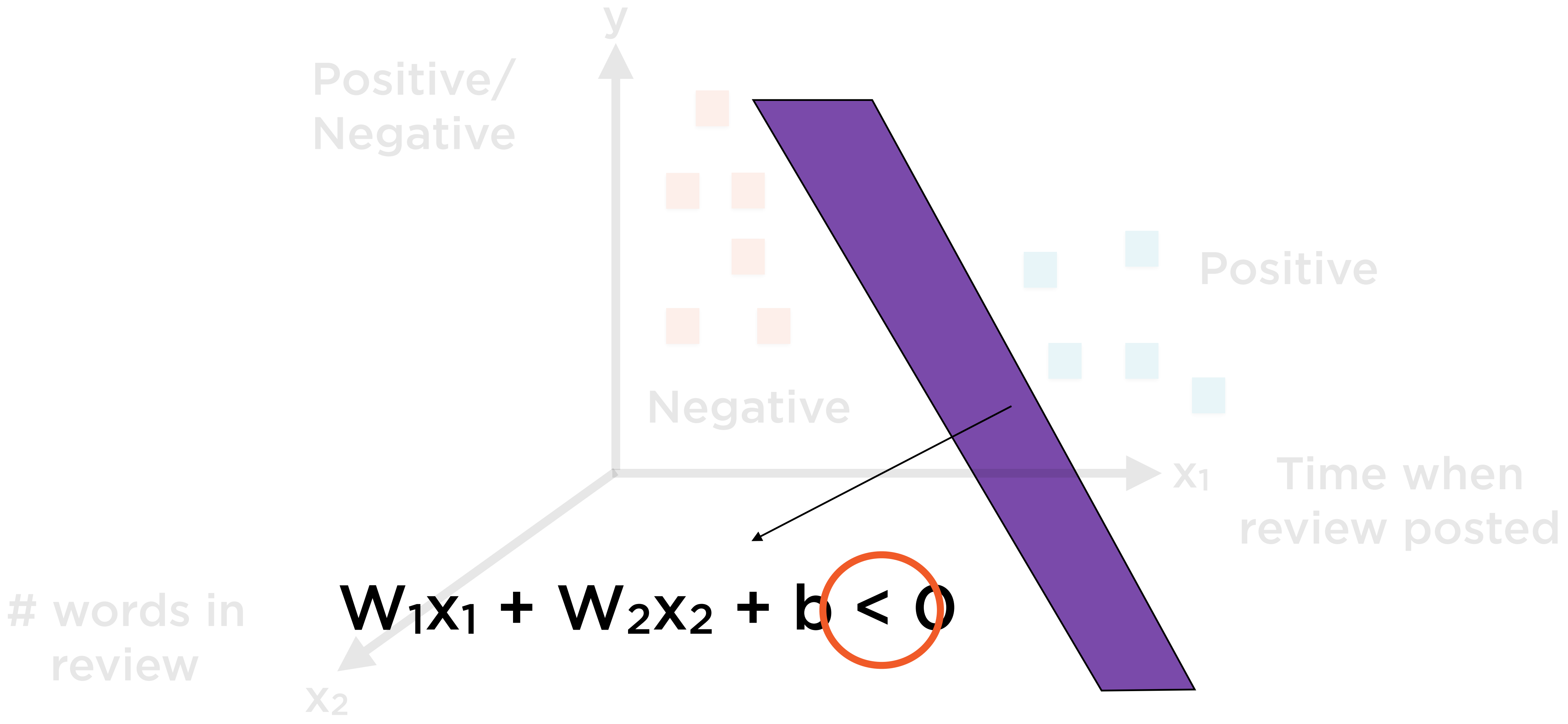
Margin Classification



Decision plane separates points based on whether

$W_1x_1 + W_2x_2 + b$ =,<,> 0

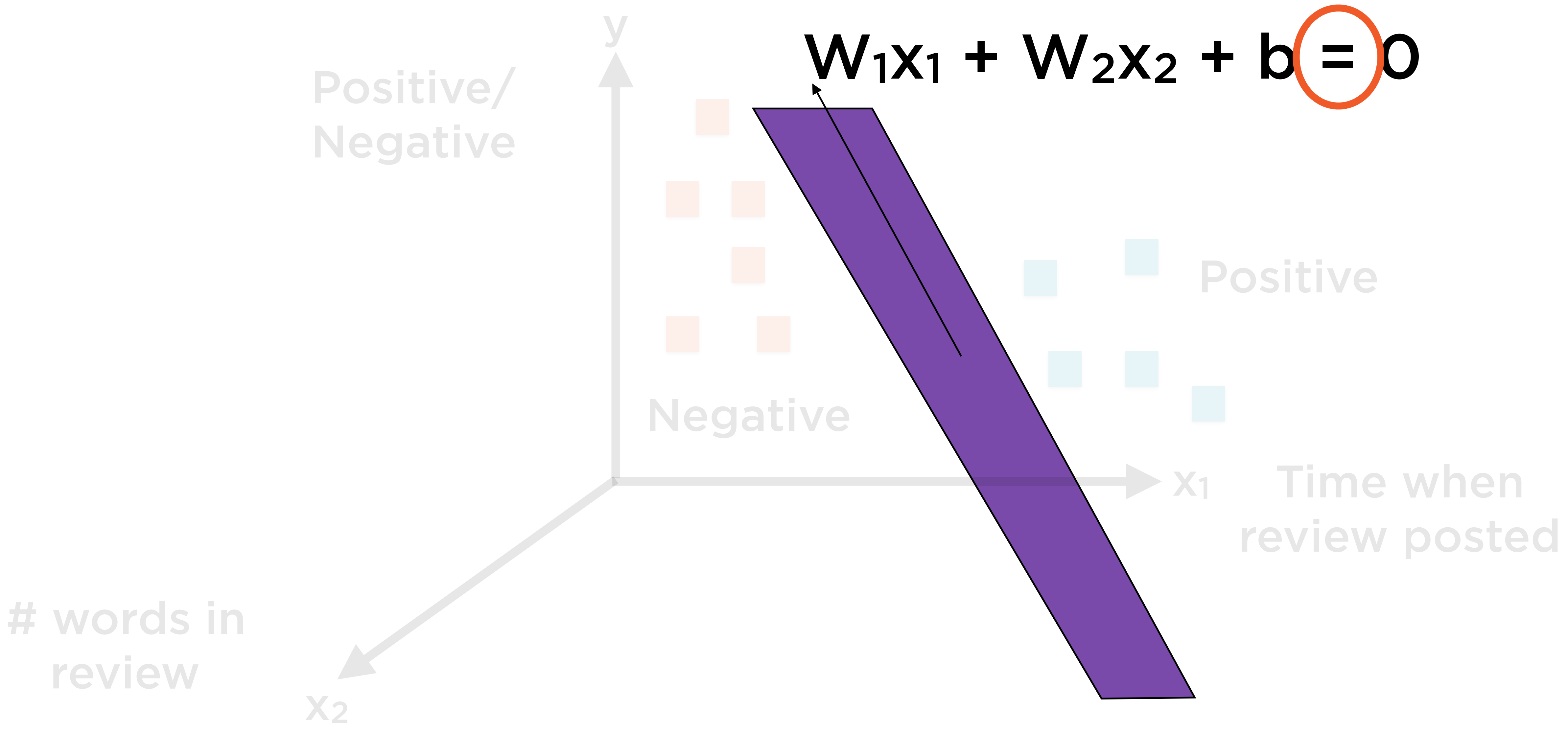
Margin Classification



Decision plane separates points based on whether

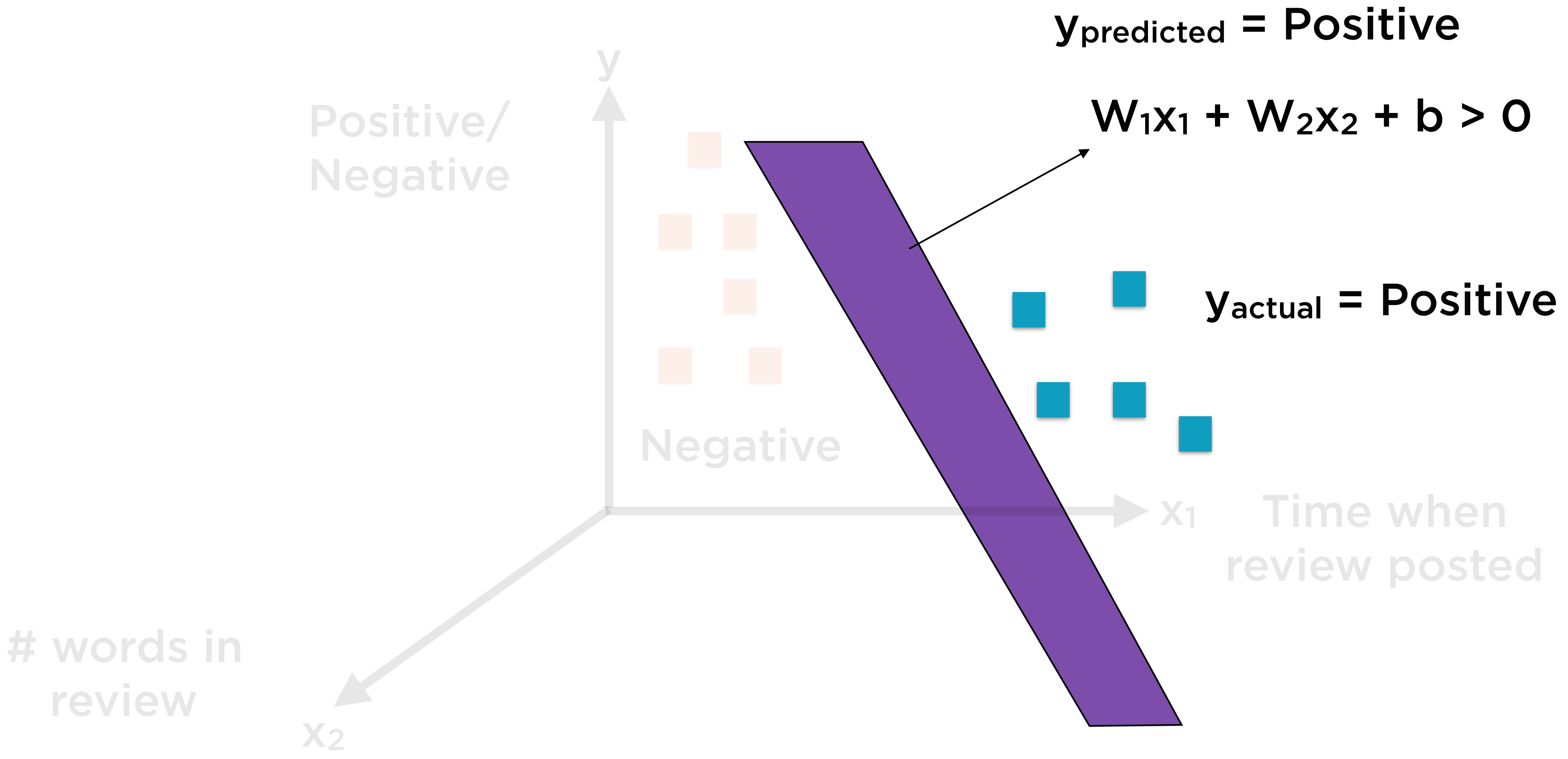
$W_1x_1 + W_2x_2 + b$ =,<,> 0

Margin Classification



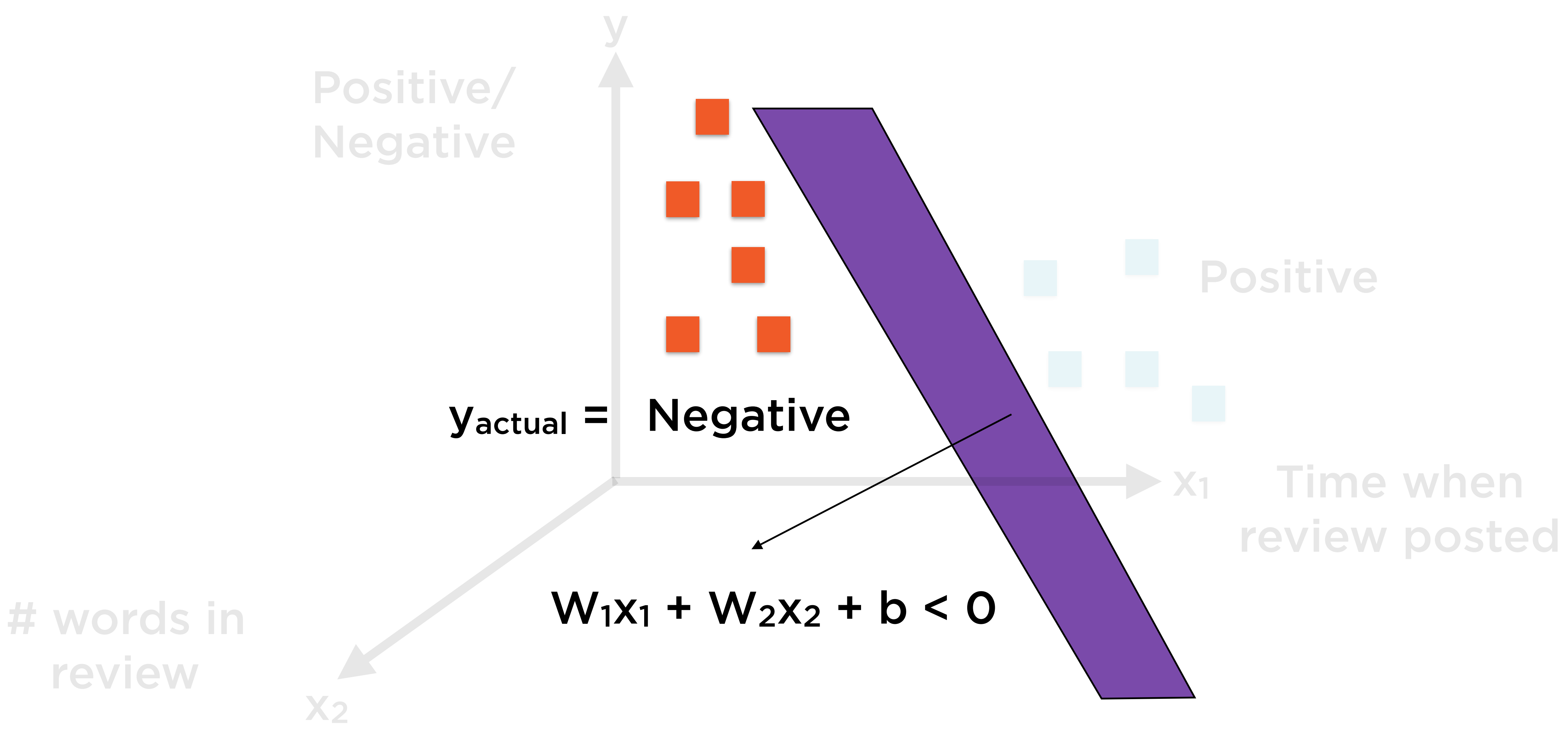
Decision plane separates points based on whether
$W_1x_1 + W_2x_2 + b$ =,<,> 0

Margin Classification



If $W_1x_1 + W_2x_2 + b > 0$ $y_{predicted}$ = Positive

Margin Classification



If $W_1x_1 + W_2x_2 + b <= 0$ $y_{predicted}$ = Negative

Classification Using the Decision Boundary

Find the "best" values of

$$W_1 , W_2 , b$$

Such that

If $W_1x_1 + W_2x_2 + b <= 0$

$y_{predicted} = 0$

If $W_1x_1 + W_2x_2 + b > 0$

$y_{predicted} = 1$

Represent
Negative = 0
Positive = 1

Classification Using the Decision Boundary

Find the "best" values of

W_1 , W_2 , b

Optimization problem

Such that

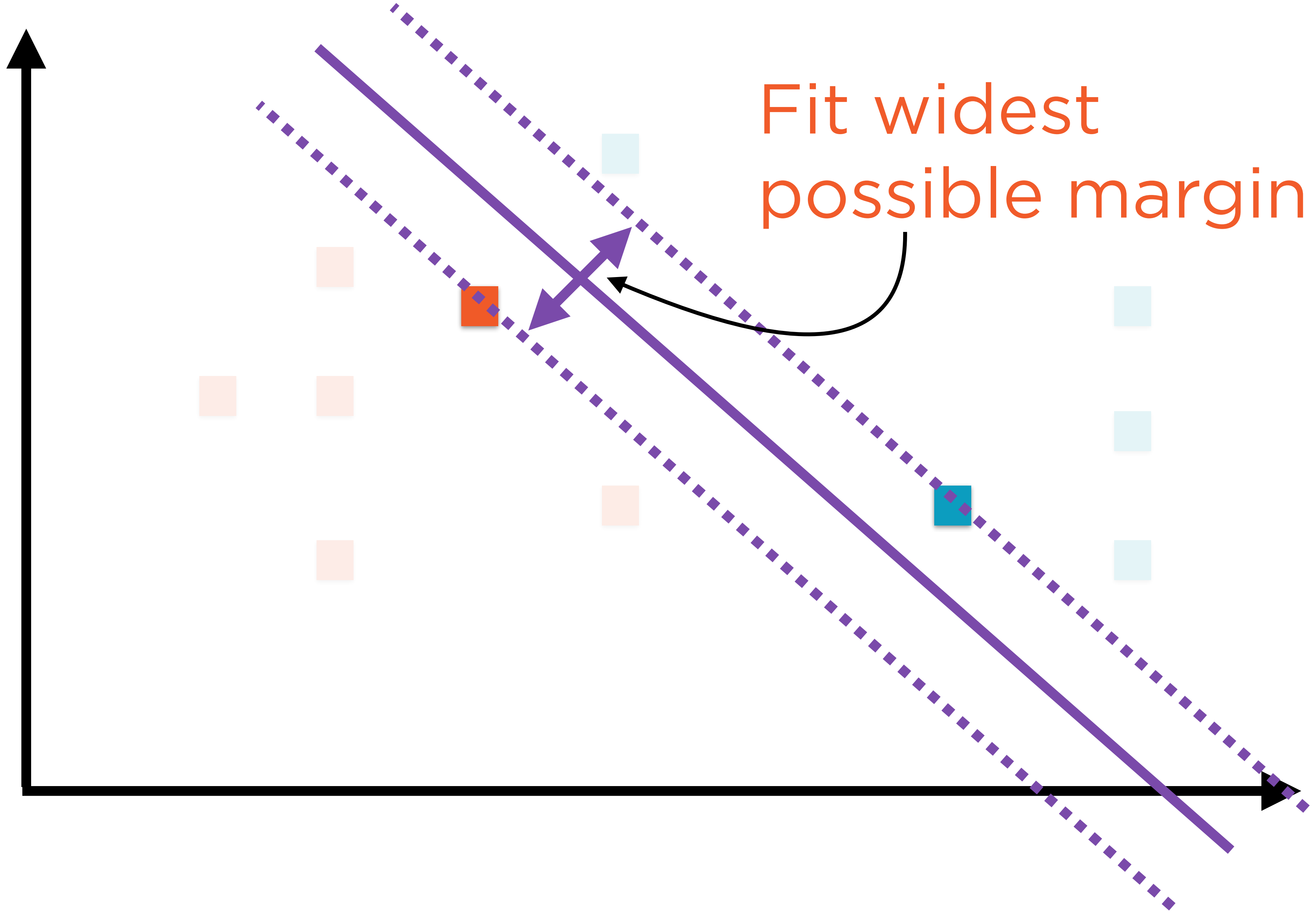
If $W_1x_1 + W_2x_2 + b <= 0$

$y_{predicted} = 0$

If $W_1x_1 + W_2x_2 + b > 0$

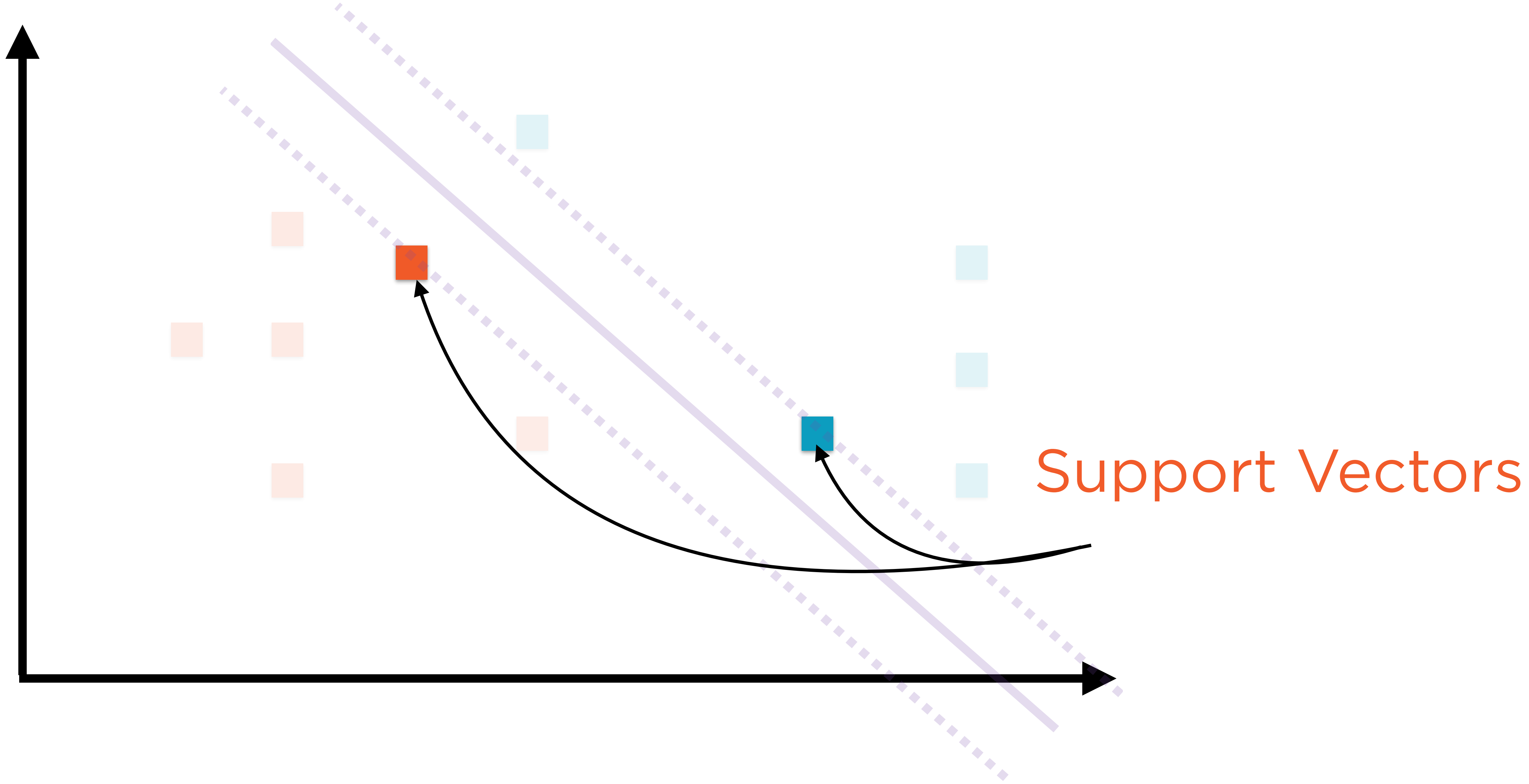
$y_{predicted} = 1$

Classification Using the Decision Boundary



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

Classification Using the Decision Boundary



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

Classification Using the Decision Boundary

Find the "best" values of

W_1 , W_2 , b

Optimization problem

"best" ~ widest margin between support vectors

but...

Such that

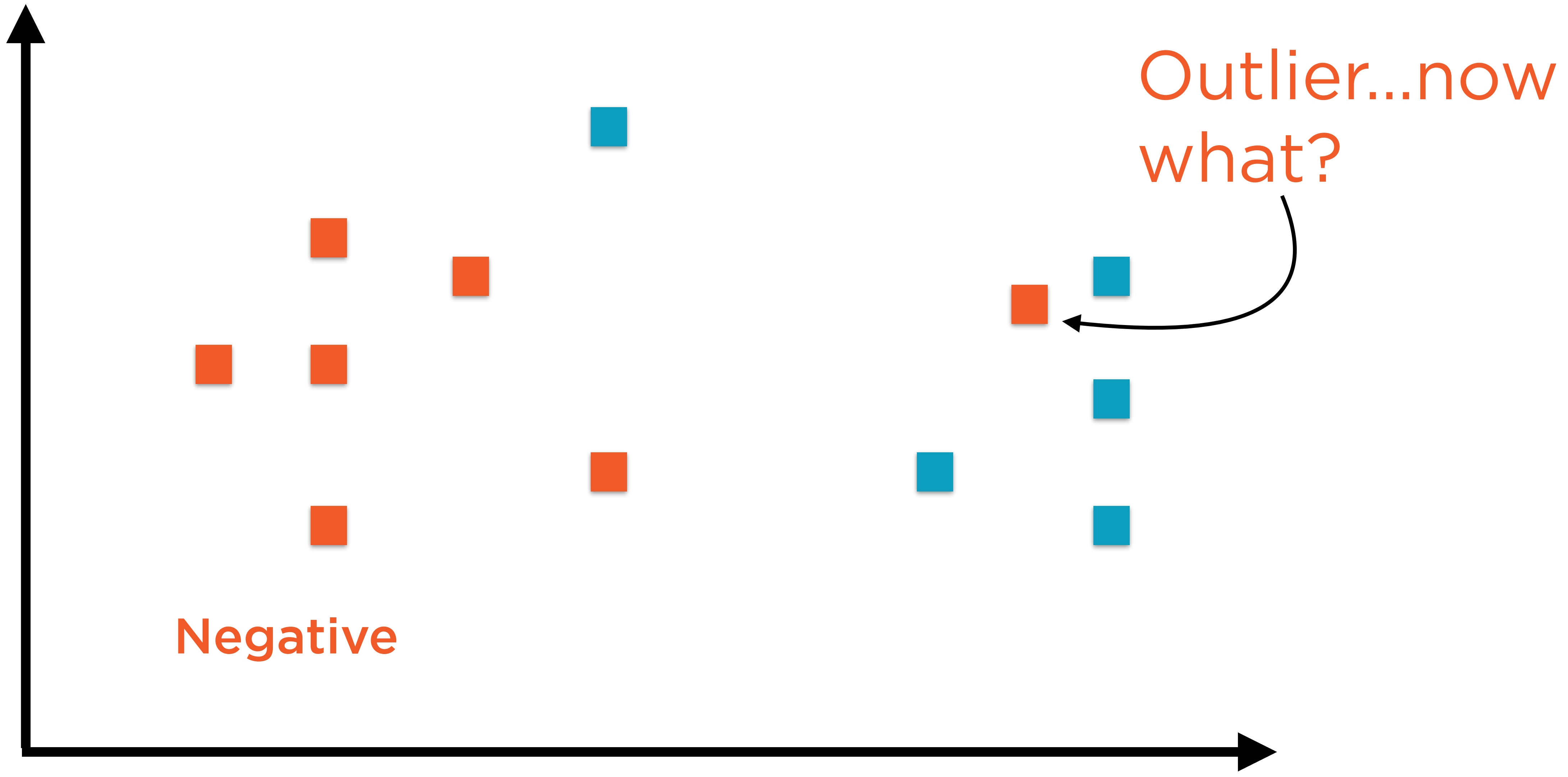
If $W_1x_1 + W_2x_2 + b <= 0$

$y_{predicted} = 0$

If $W_1x_1 + W_2x_2 + b > 0$

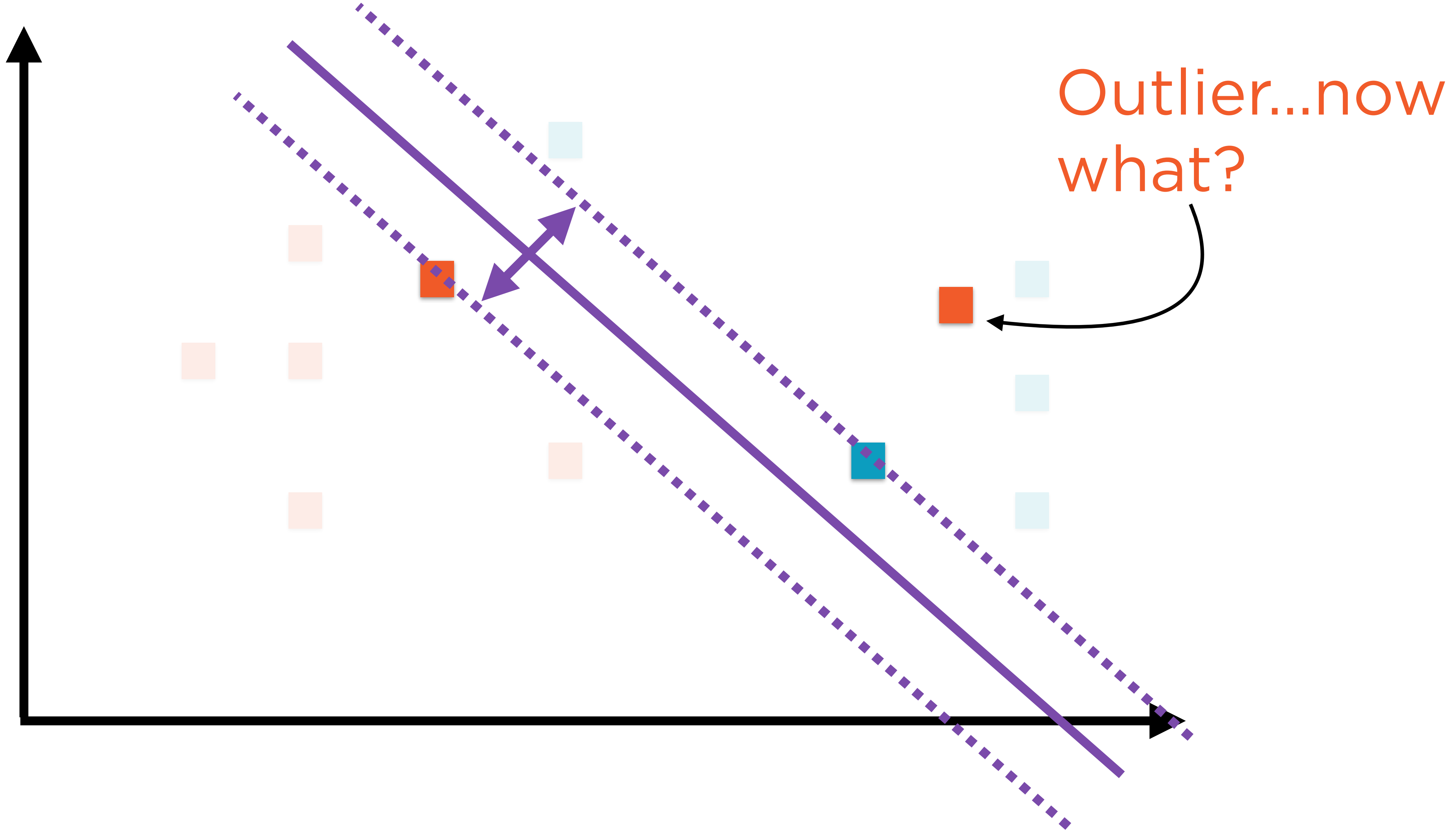
$y_{predicted} = 1$

Classification Using the Decision Boundary



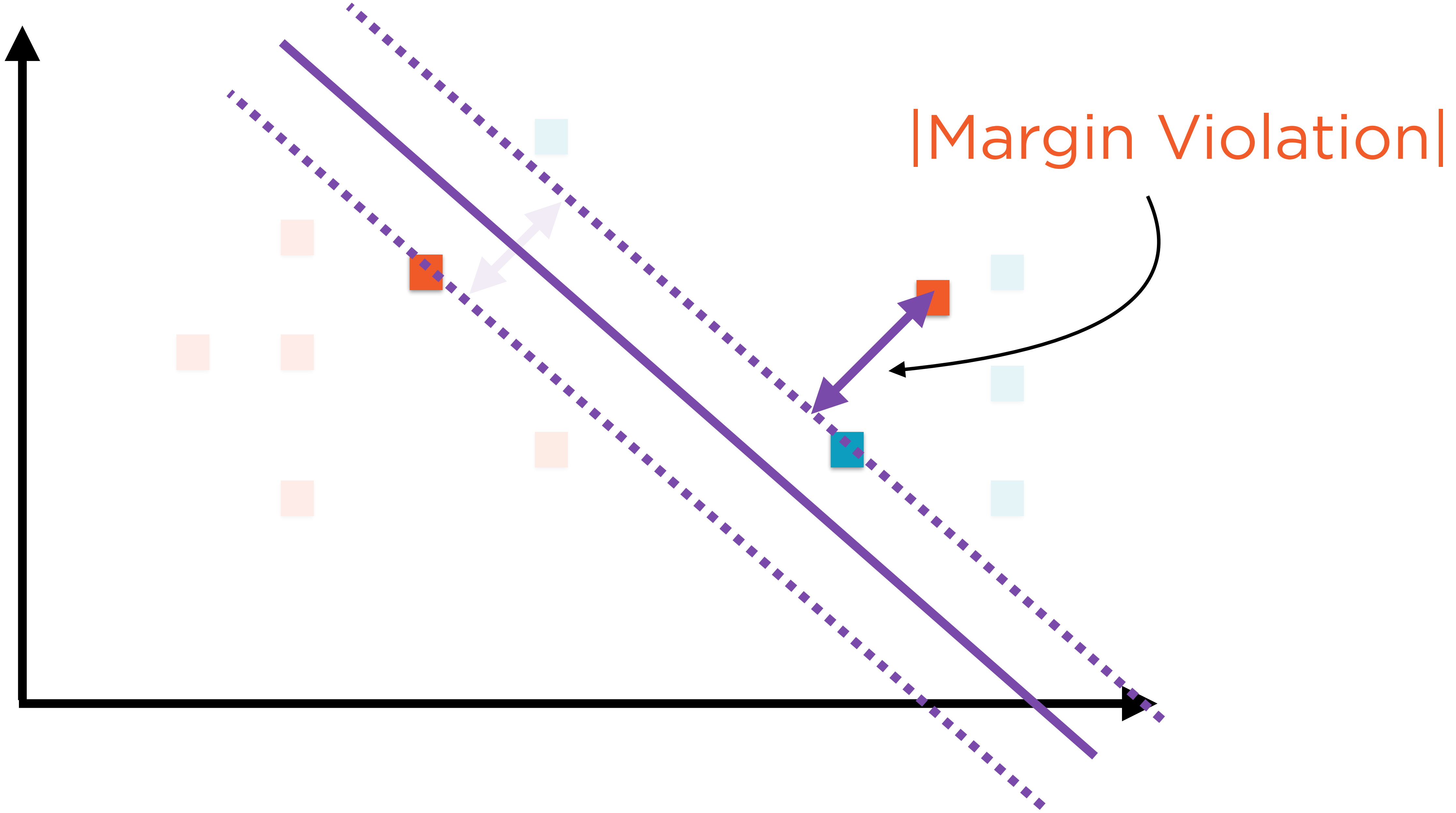
But "best" must also avoid or minimize outliers (by penalizing them during the optimization)

Classification Using the Decision Boundary



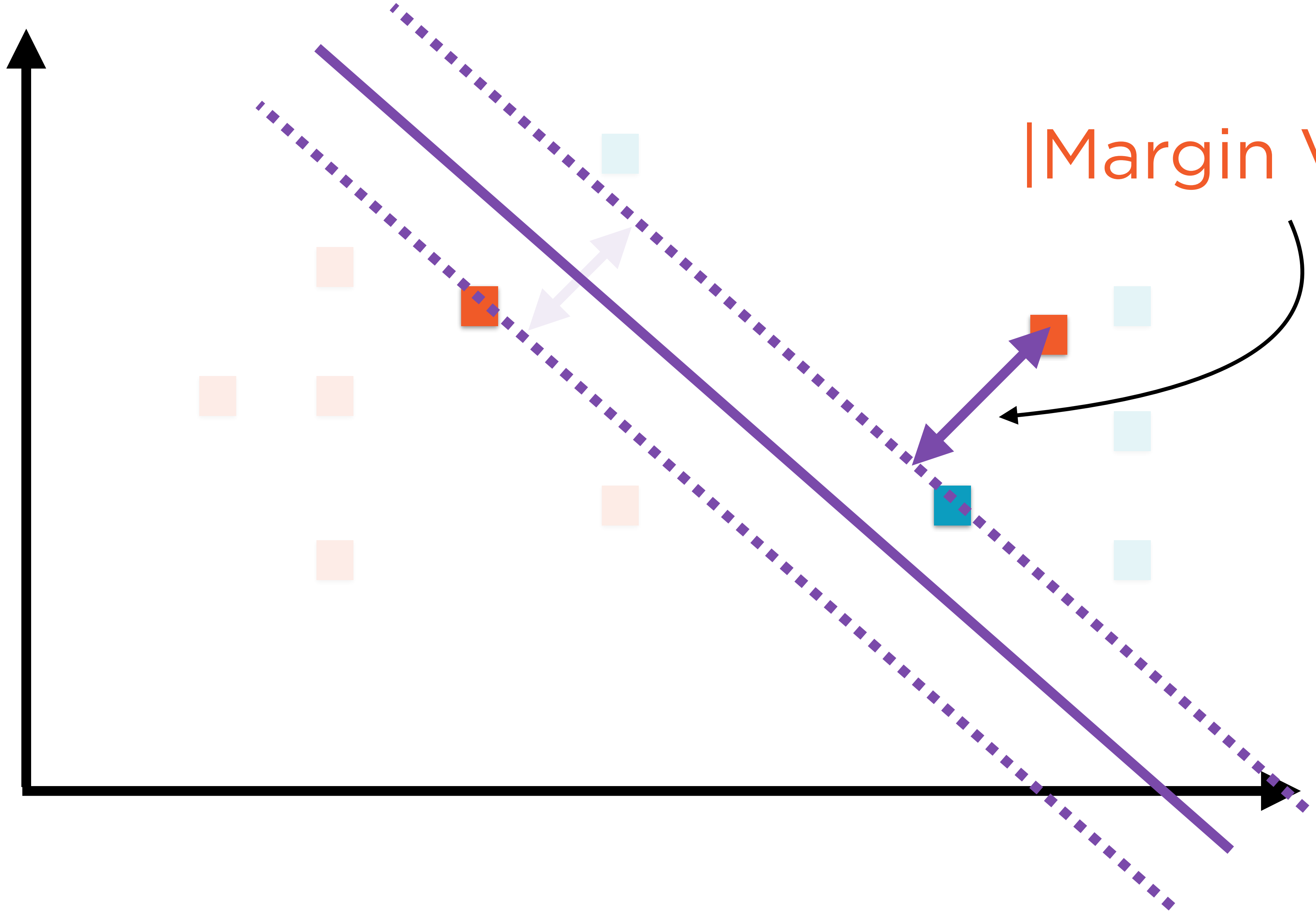
But "best" must also avoid or minimize outliers (by penalizing them during the optimization)

Classification Using the Decision Boundary



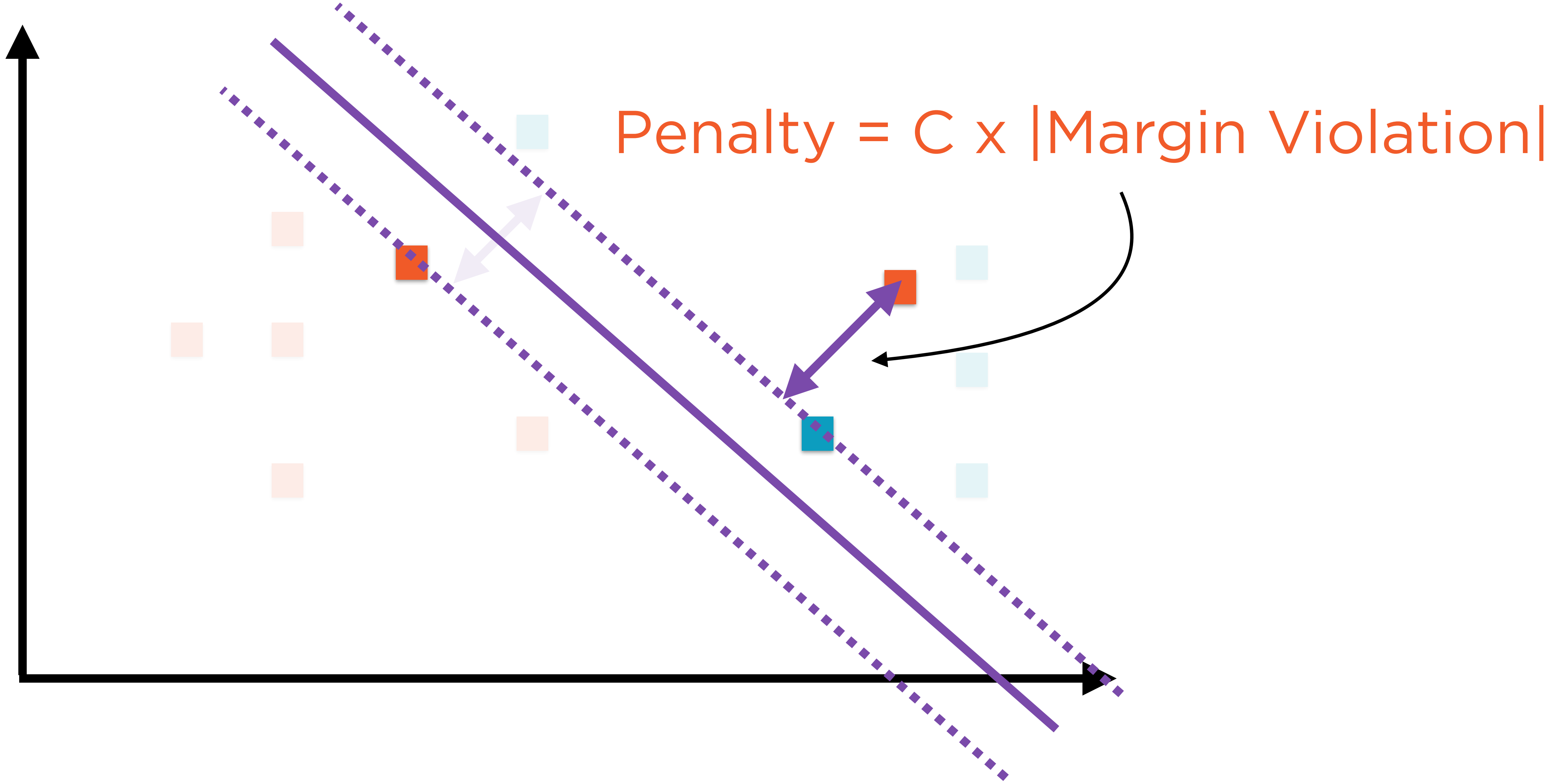
Calculate the magnitude of the margin violation for each point on the wrong side of the boundary

Classification Using the Decision Boundary



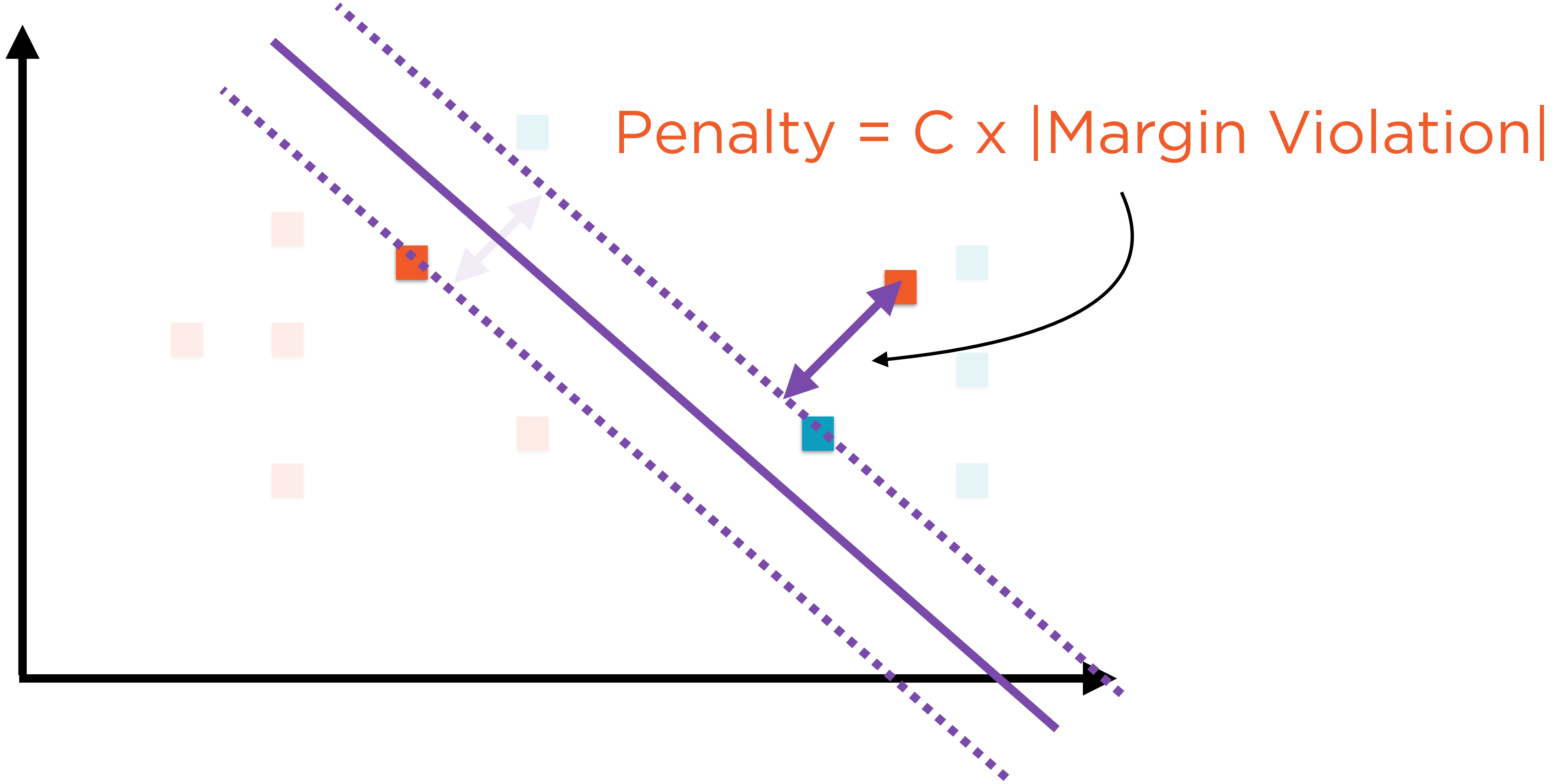
Multiply this magnitude of margin violation by a penalty factor C

Classification Using the Decision Boundary



Penalize each outlier using hyperparameter C

Classification Using the Decision Boundary



Very large values of C ~ hard margin classification

Very small values of C ~ soft margin classification

Classification Using the Decision Boundary

Find the "best" values of

W_1 , W_2 , b

Optimization problem

"best" ~ widest margin between support vectors

but…

…penalize each margin violator using hyperparameter C

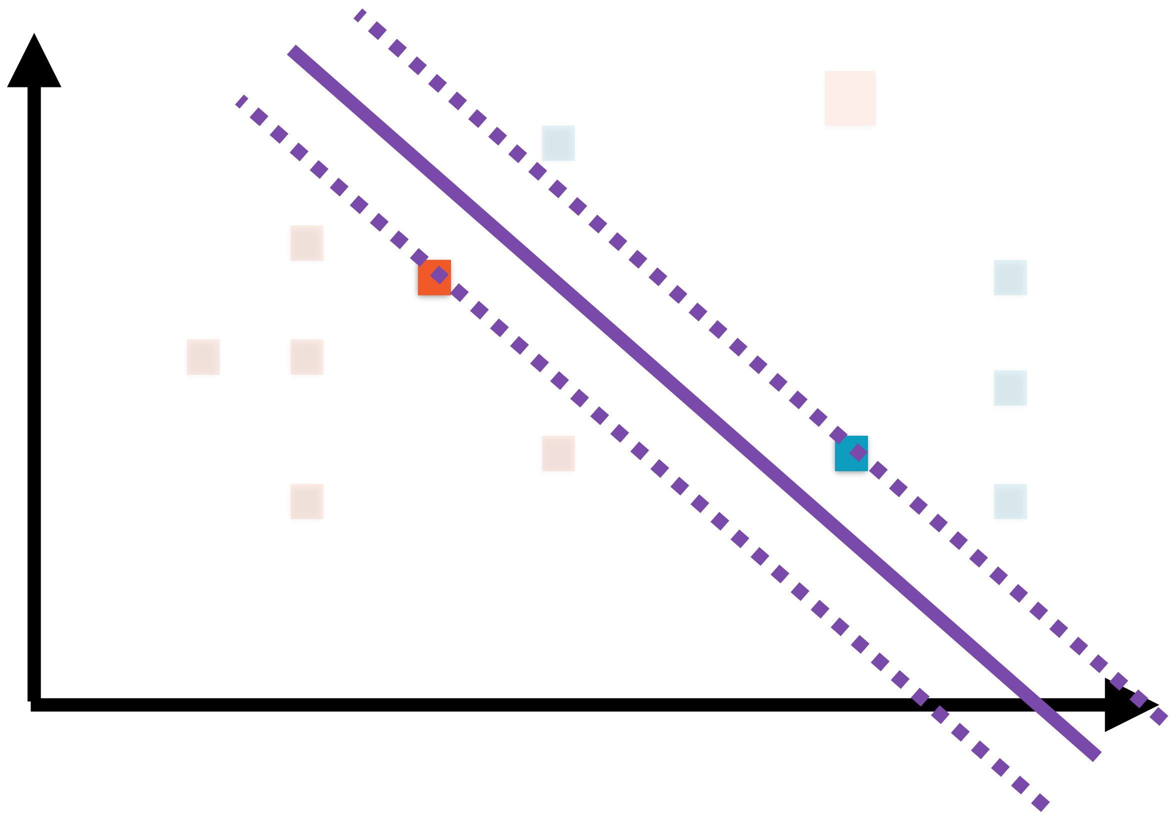
Such that

If $W_1x_1 + W_2x_2 + b <= 0$

$y_{predicted} = 0$

If $W_1x_1 + W_2x_2 + b > 0$

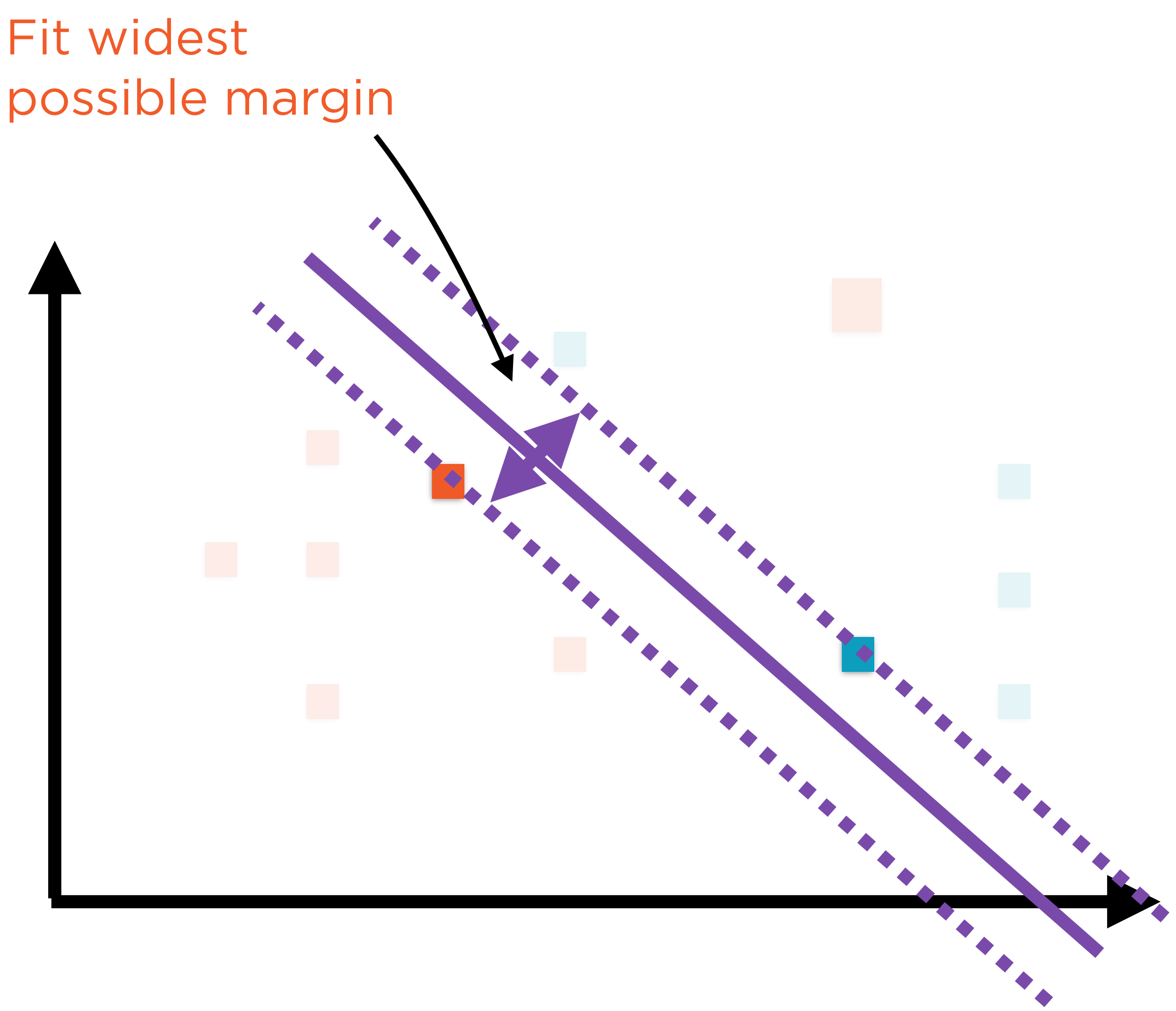
$y_{predicted} = 1$



Solving SVM Optimization

Don't need to know precise math

Understand that "best" decision boundary

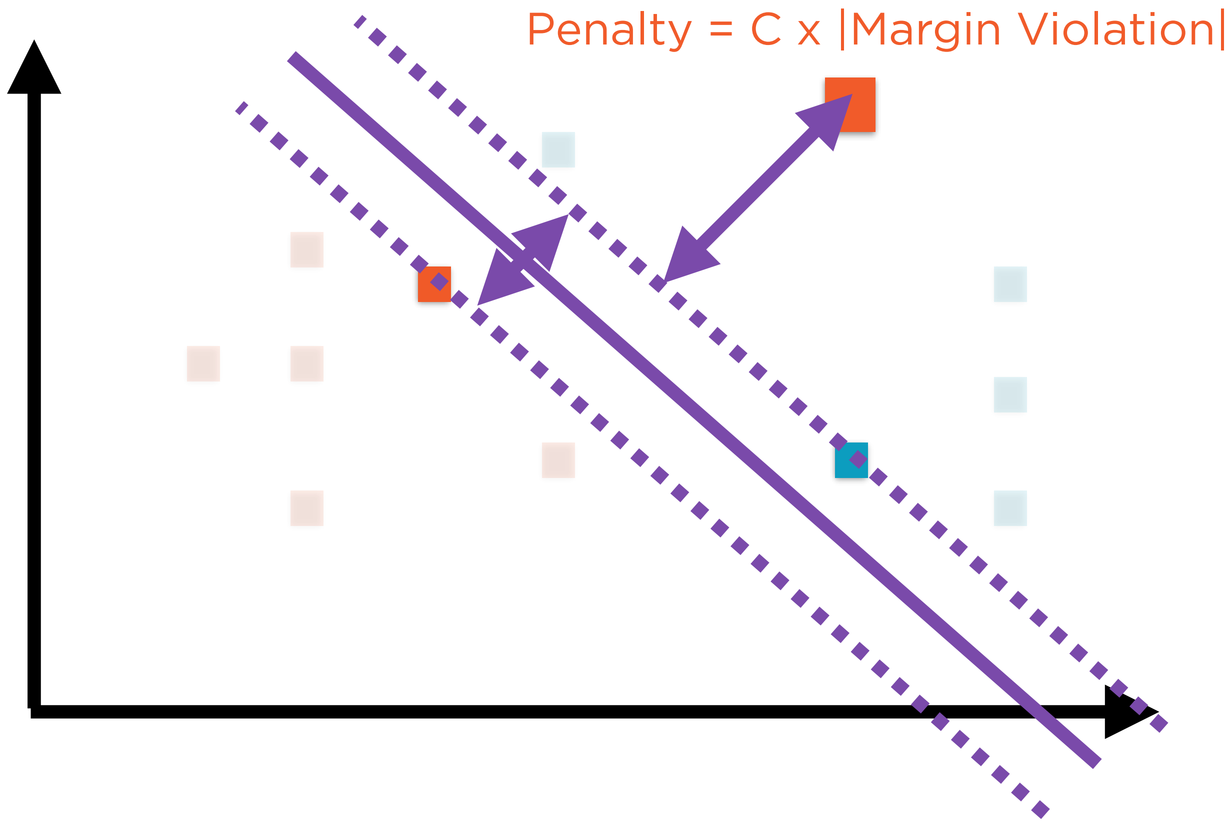


Solving SVM Optimization

Don't need to know precise math

Understand that "best" decision boundary

- **seeks to maximize width of street**

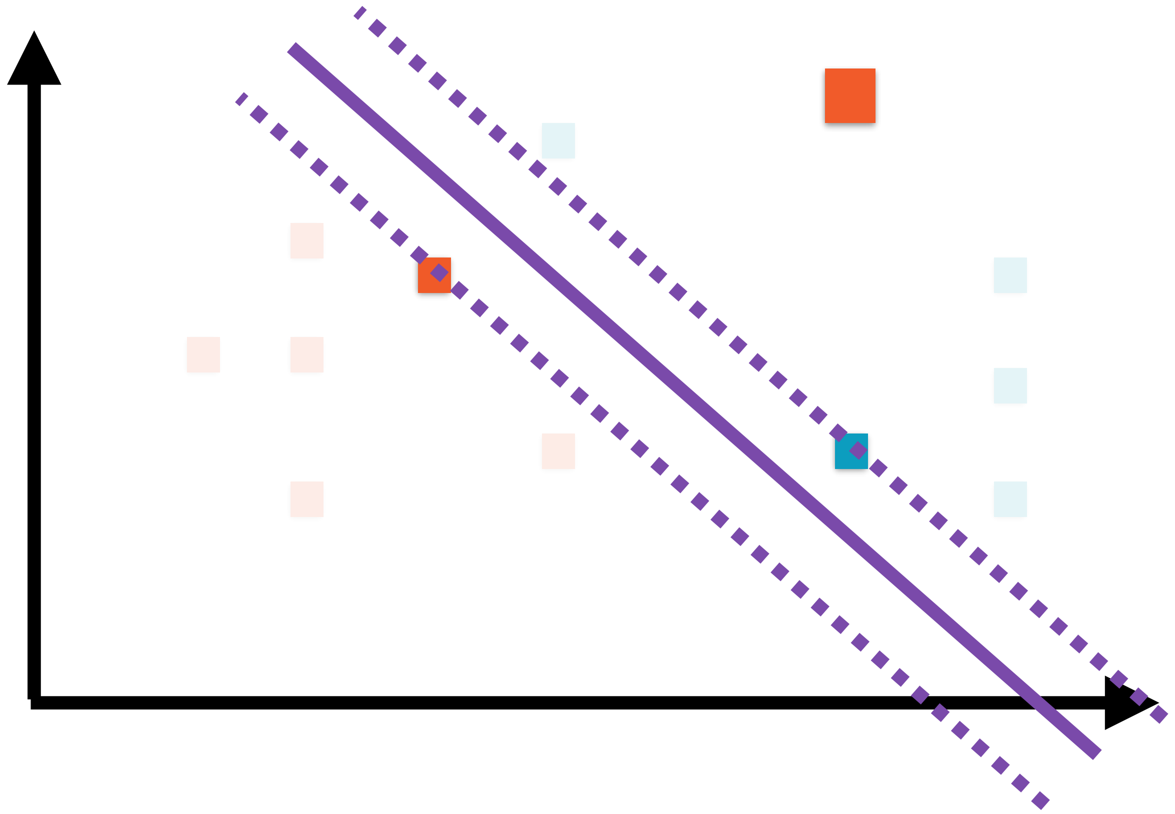


Solving SVM Optimization

Don't need to know precise math

Understand that "best" decision boundary

- **seeks to maximize width of street**
- **seeks to minimize margin violations**



Solving SVM Optimization

Don’t need to know precise math

Understand that “best” decision boundary

- seeks to maximize width of street
- seeks to minimize margin violations

These two objectives are in conflict with each other

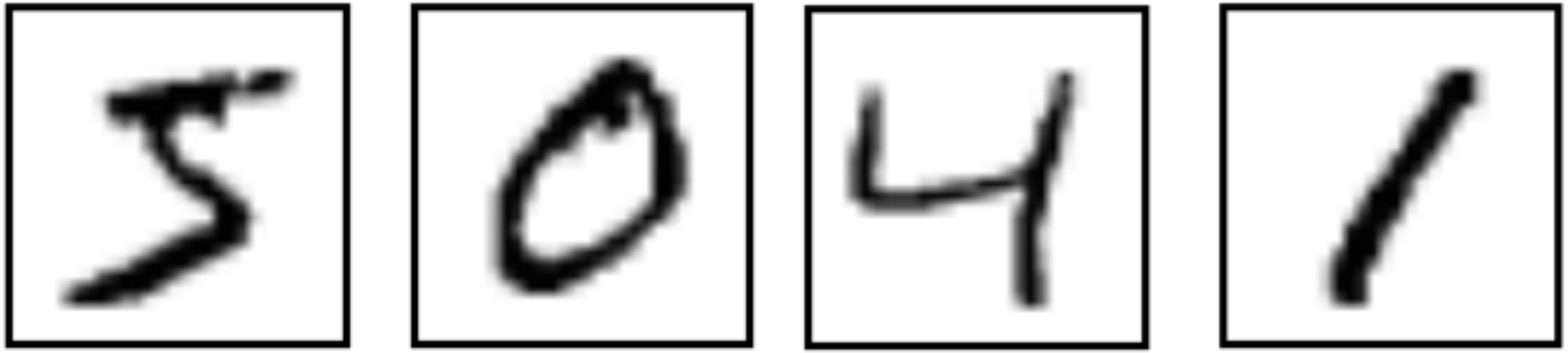
Demo

Document classification with SVMs in scikit-learn

Demo

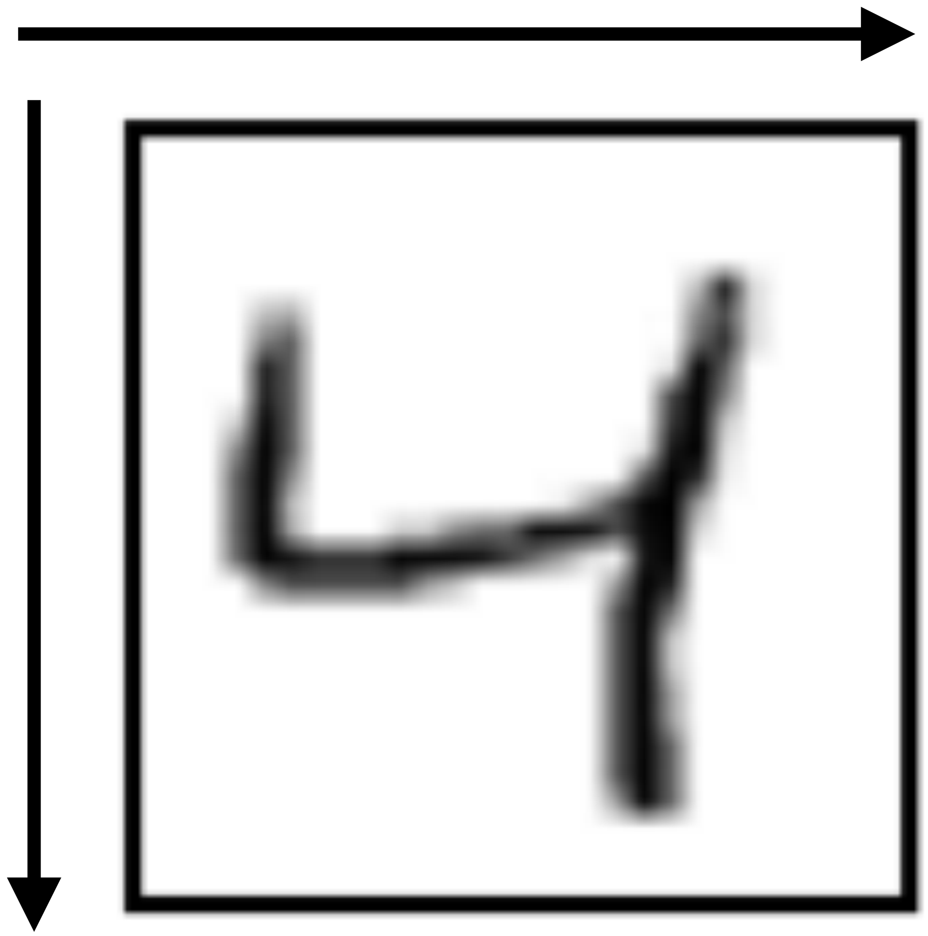
MNIST image classification with SVMs in scikit-learn

MNIST Dataset



Each digit is in grayscale

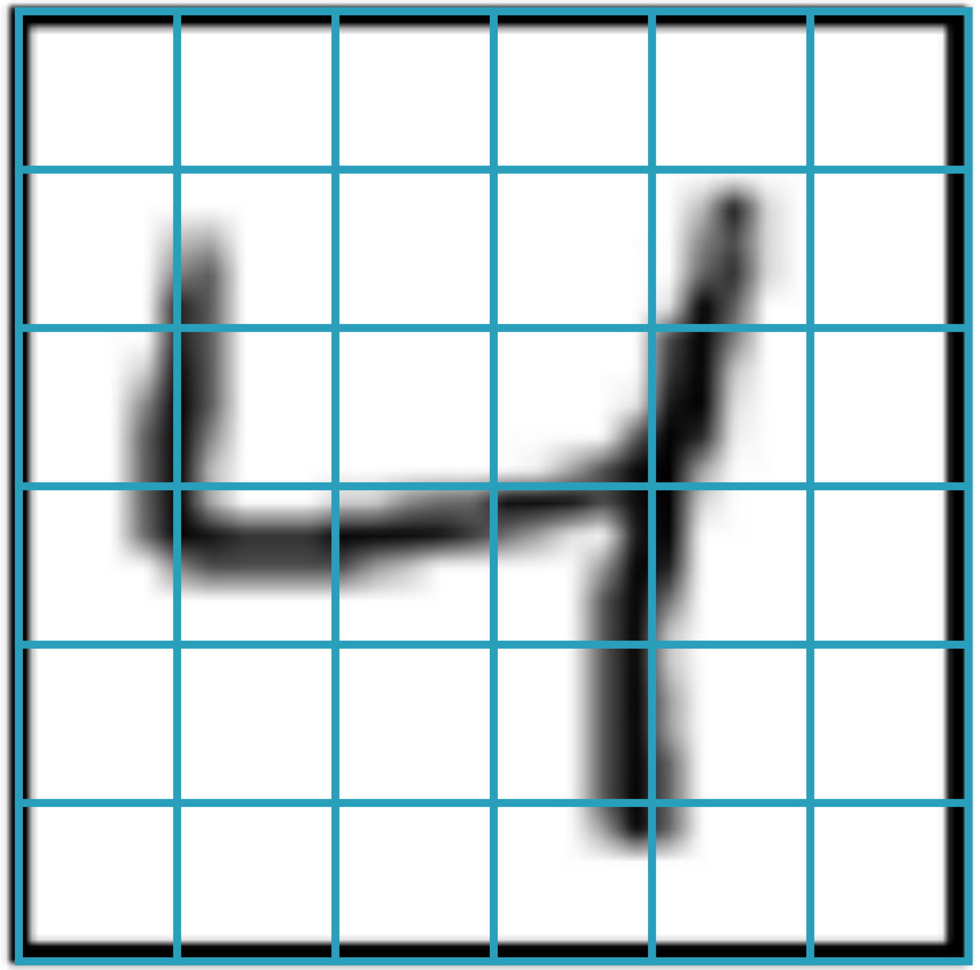
MNIST Dataset



Every image is standardized to be of size 28x28

= 784 pixels

MNIST Dataset



Every pixel holds a single value for intensity

MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

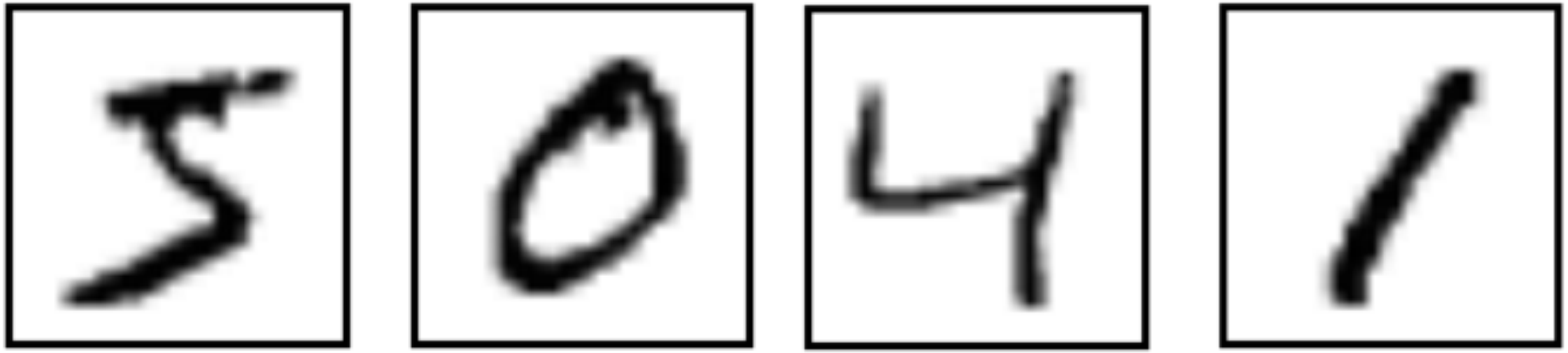
MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

MNIST Dataset



Every image has an associated label

Decision Trees

Jockey or Basketball Player?

Jockeys

Tend to be light to meet horse carrying limits

Basketball Players

Tend to be tall, strong and heavy

Jockey or Basketball Player?

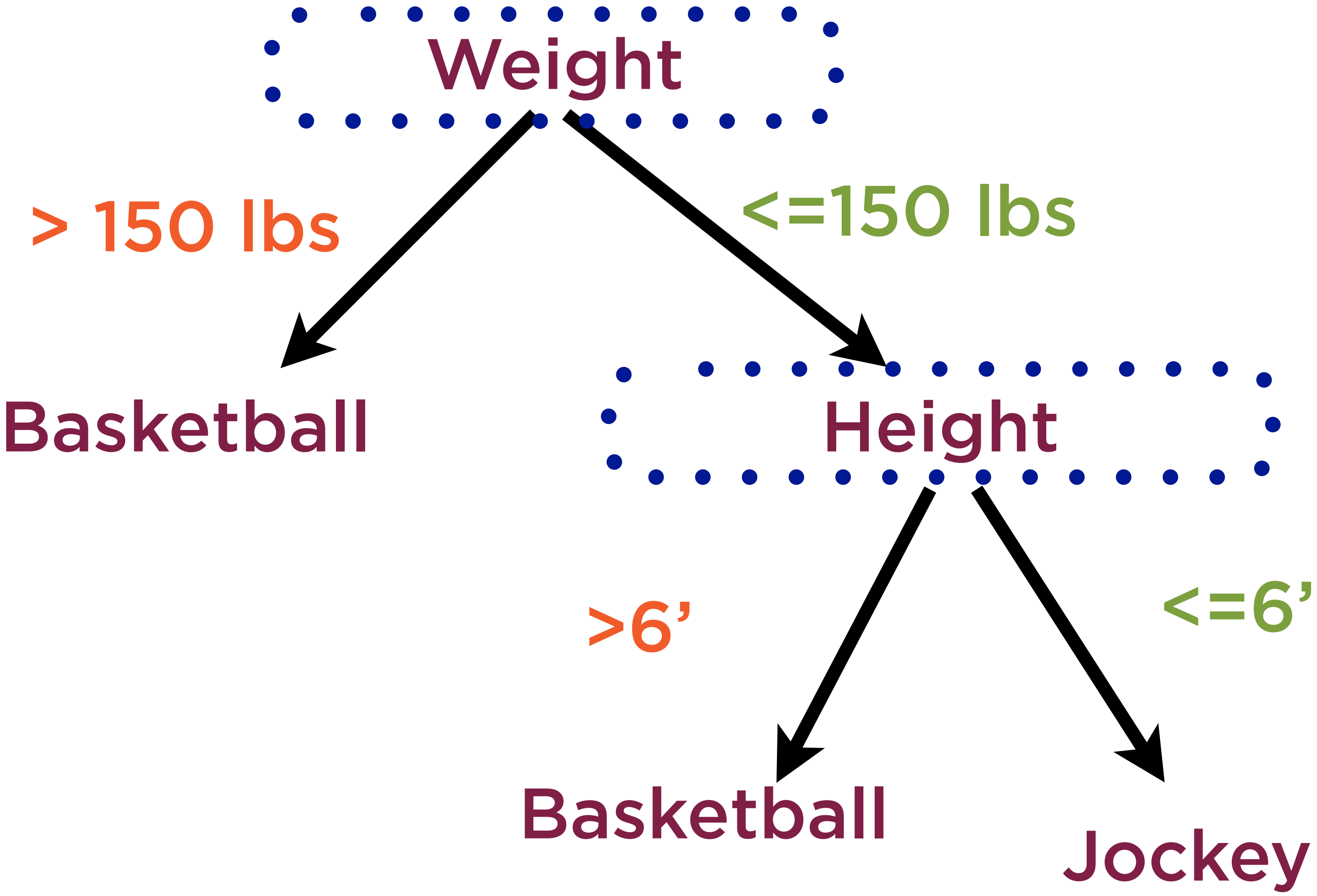
Intuitively know

- **jockeys tend to be light…**
- **…and not very tall**
- **basketball players tend to be tall**
- **…and also quite heavy**

Fit knowledge into rules

Each rule involves a threshold

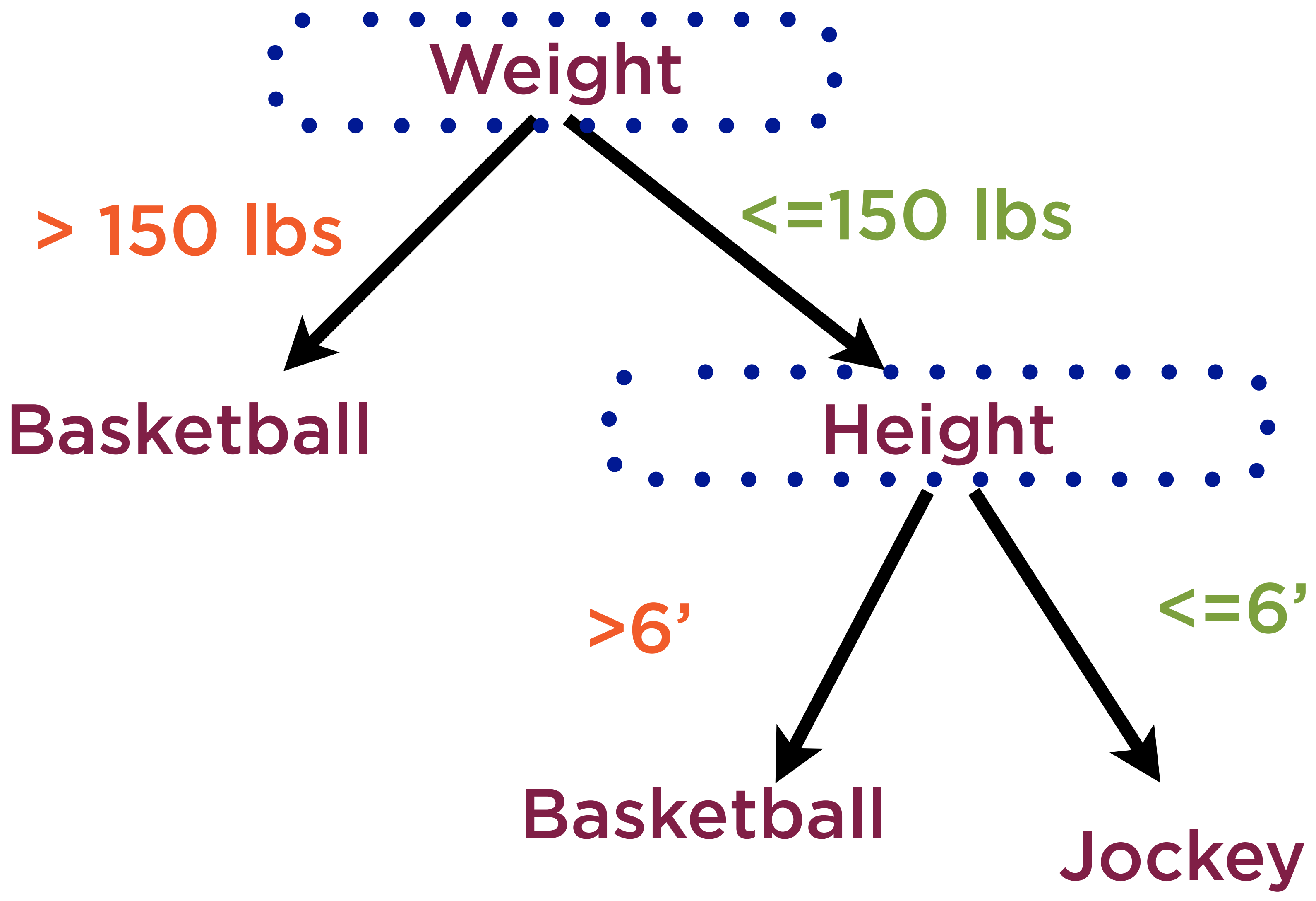
Decision Tree



Order of decision variables matters

Rules and order found using ML

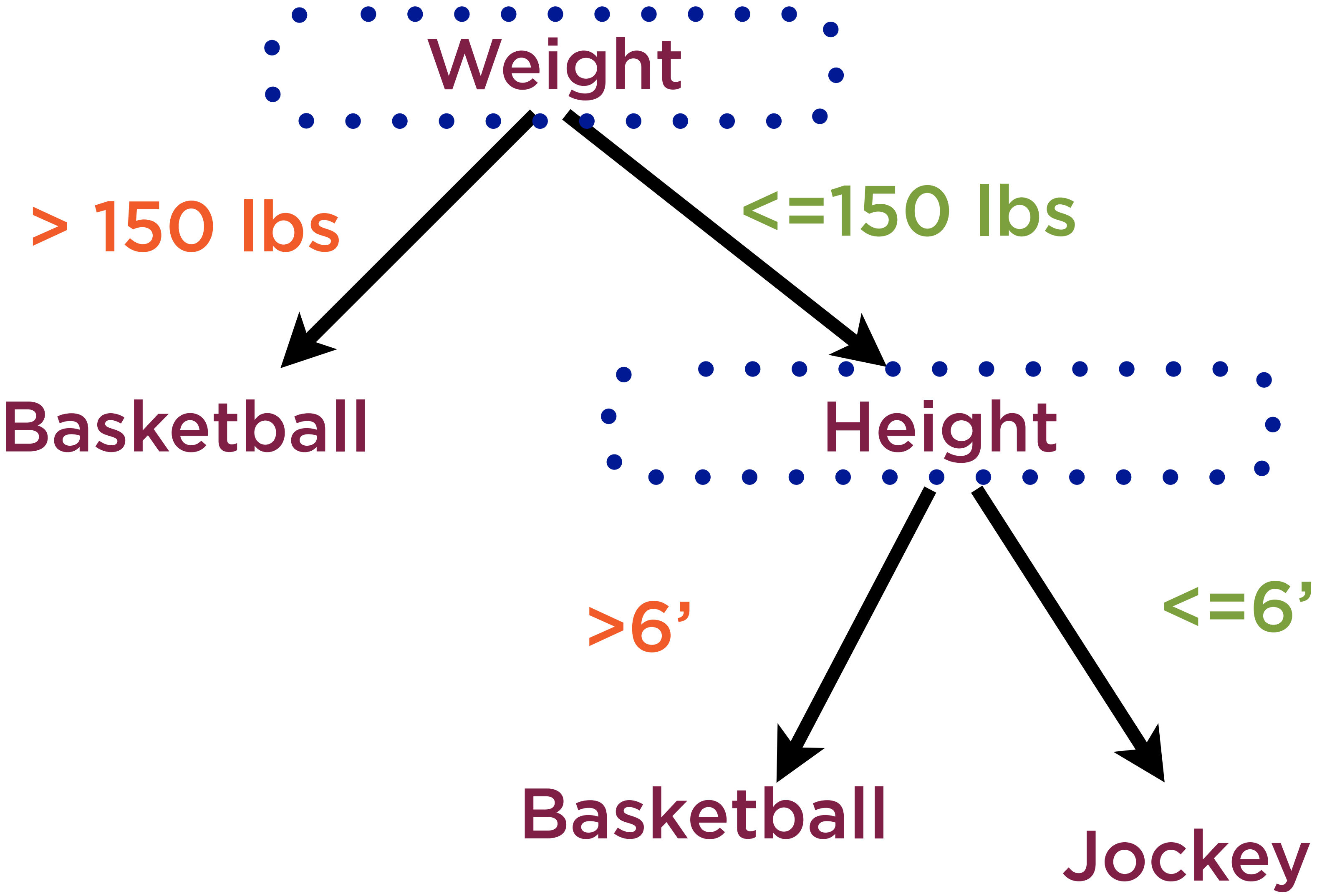
Decision Tree



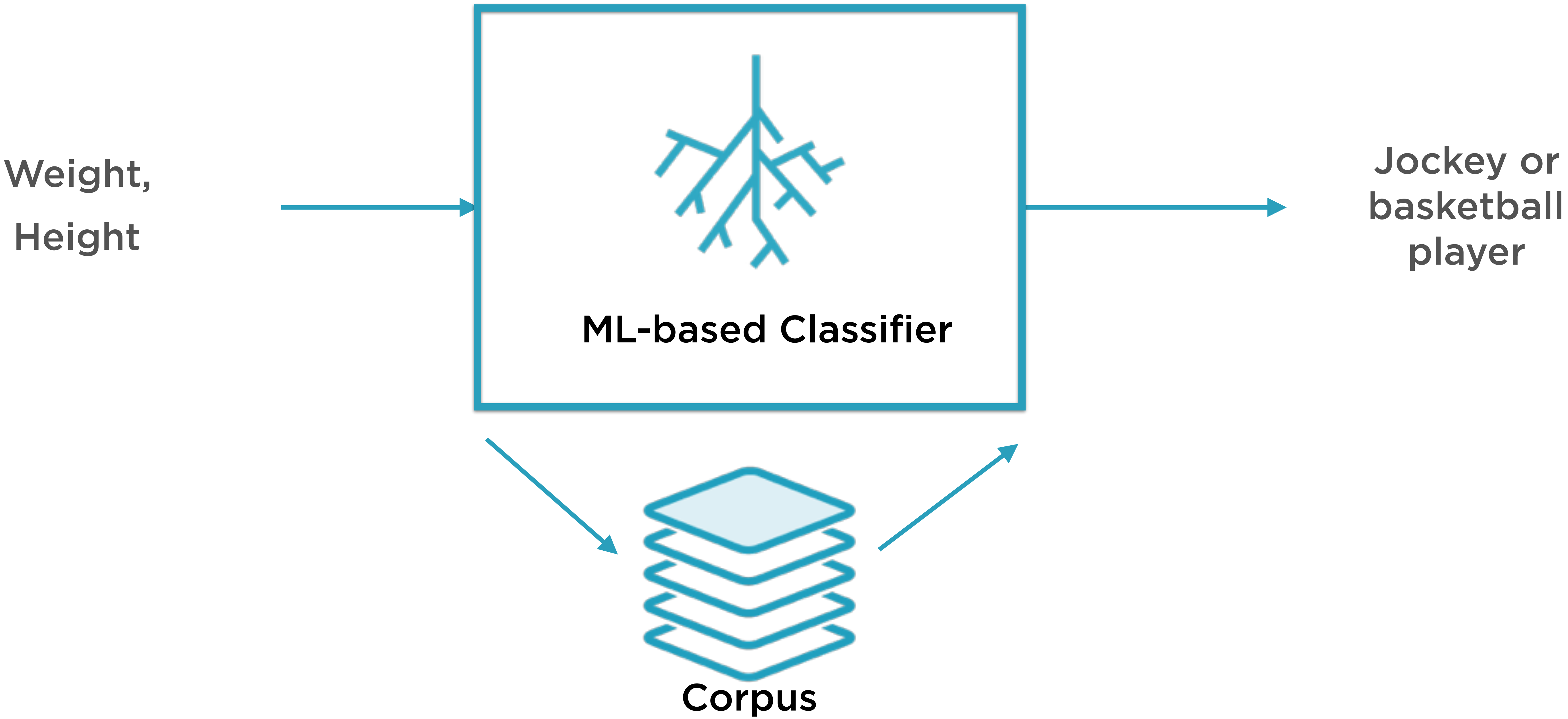
“CART”

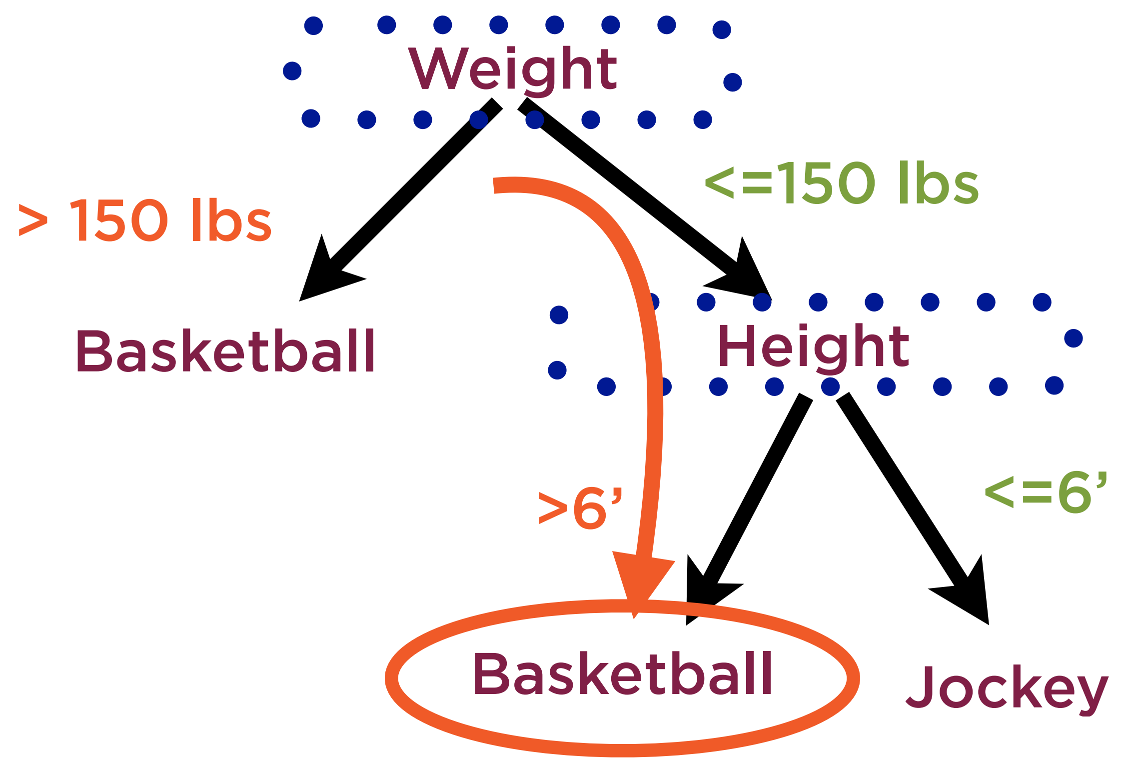
<u>C</u>lassification <u>A</u>nd <u>R</u>egression <u>T</u>ree

Decision Tree



Decision Trees for Classification



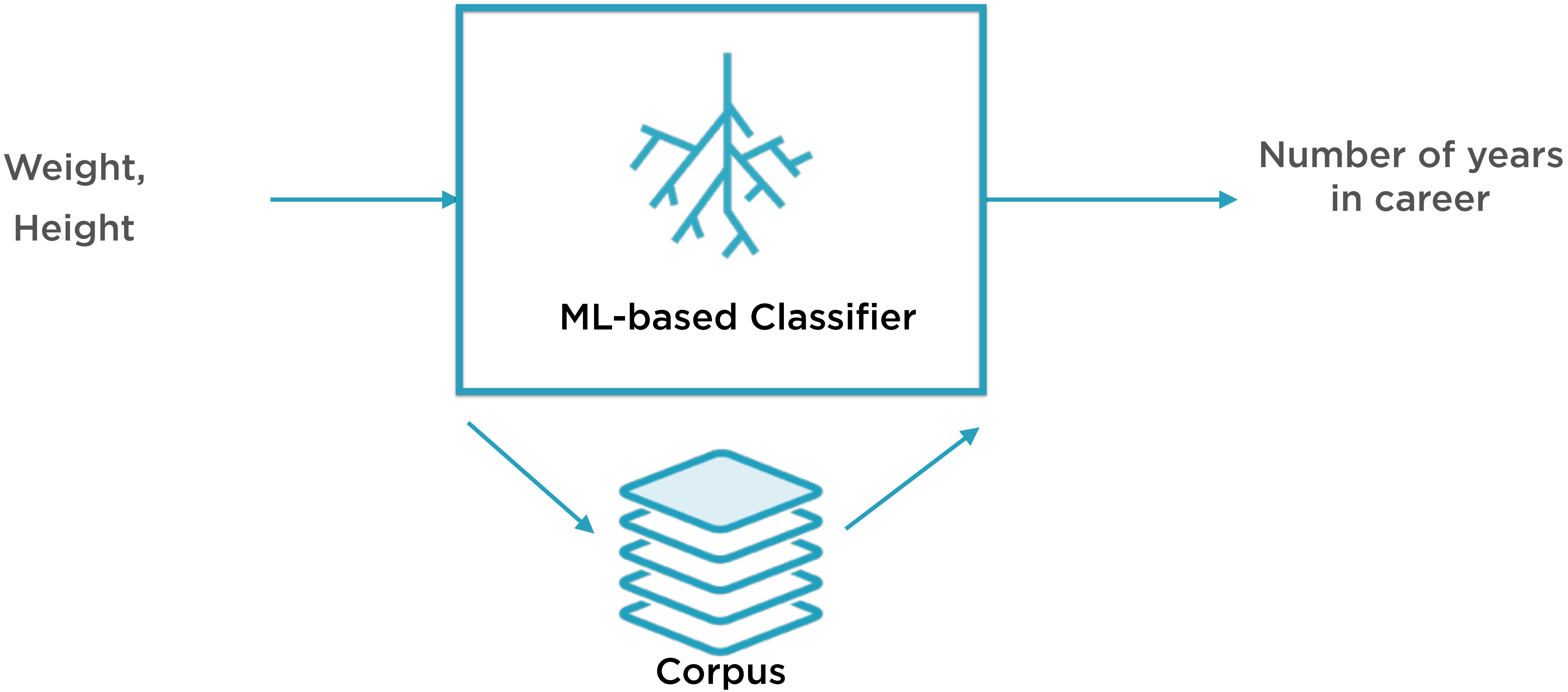


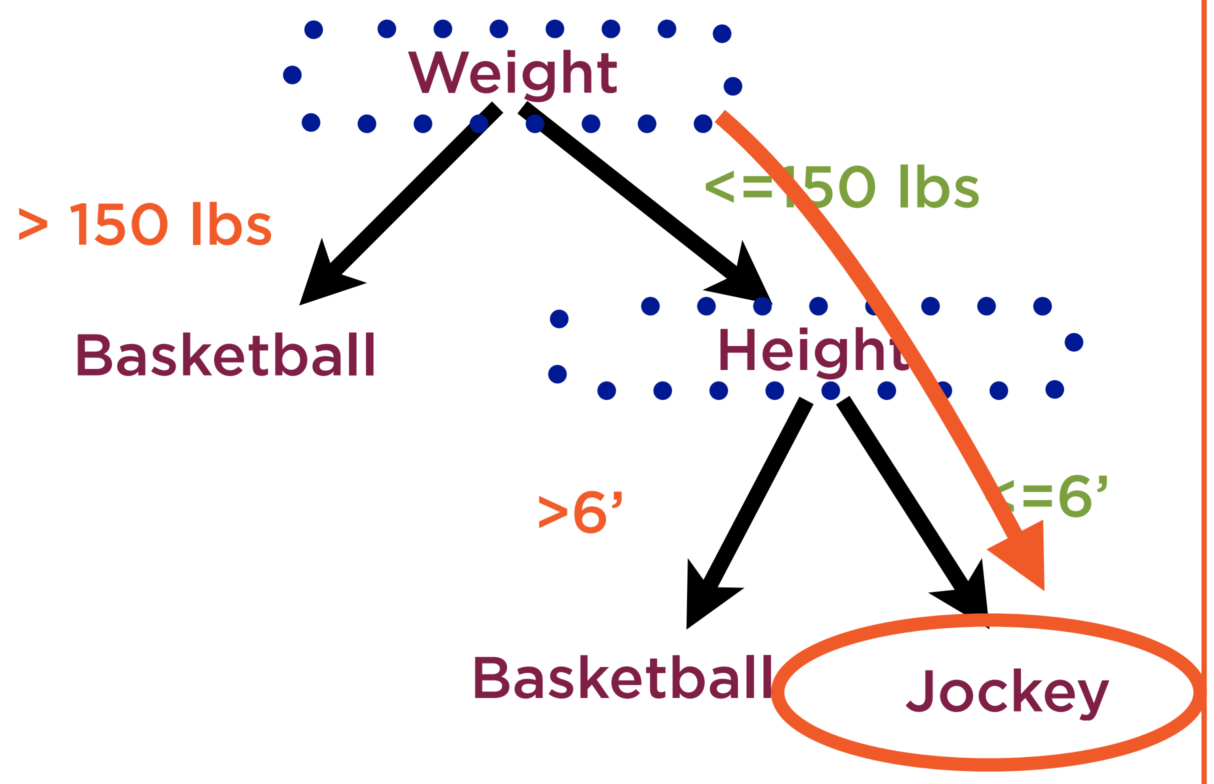
Decision Trees for Classification

To solve

- Traverse tree to find right node
- Return most frequent label of all training data points in that node

Decision Trees for Regression





Decision Trees for Regression

To solve

- **Traverse tree to find right node**
- **Return average number of years of all training data points in that node**

Muggsy Bogues

Shortest player ever in the NBA

5’3” and 135 lbs

Our tree would classify him as Jockey

No threshold is perfect!

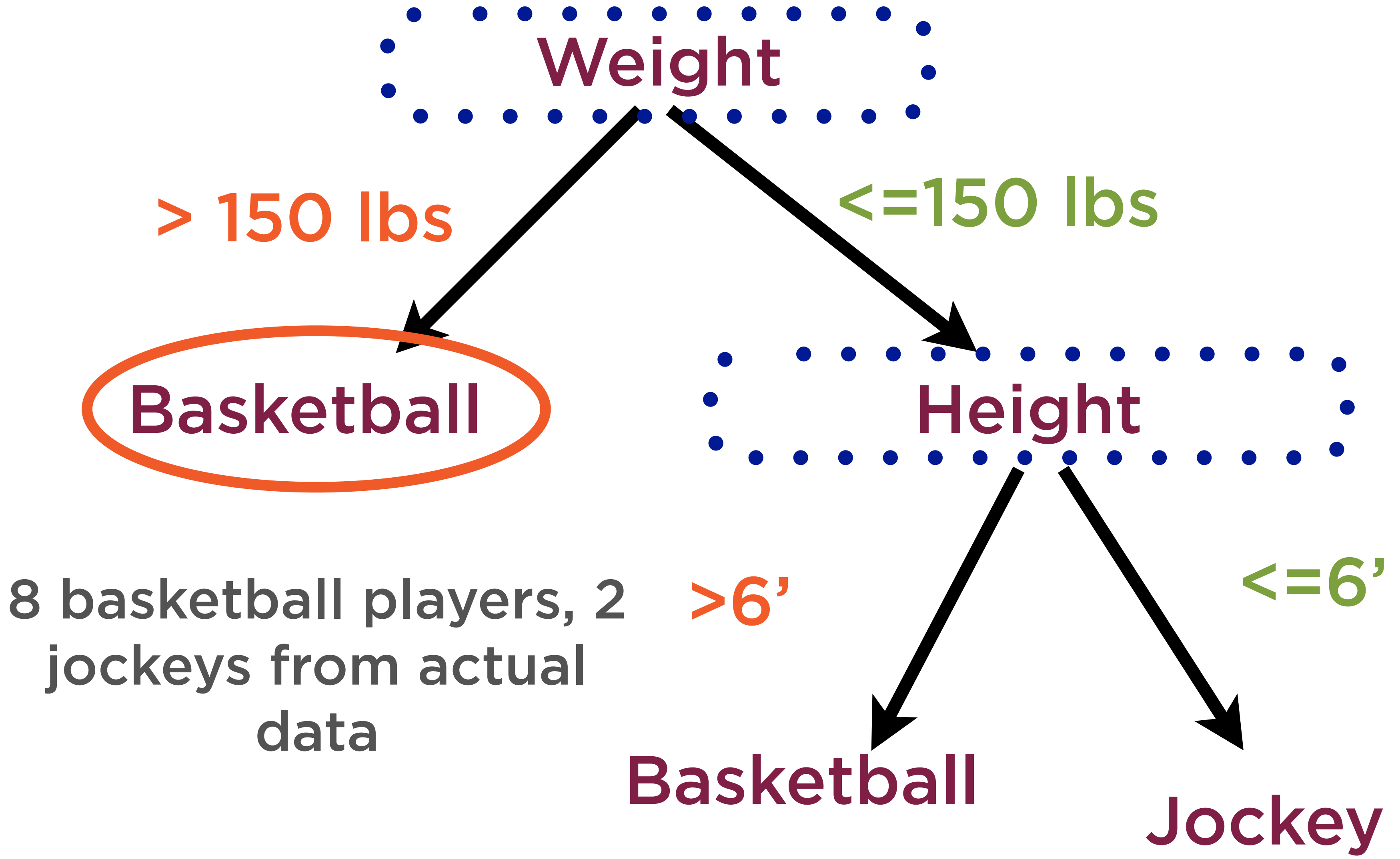
Tree Construction

CART optimizes tree construction

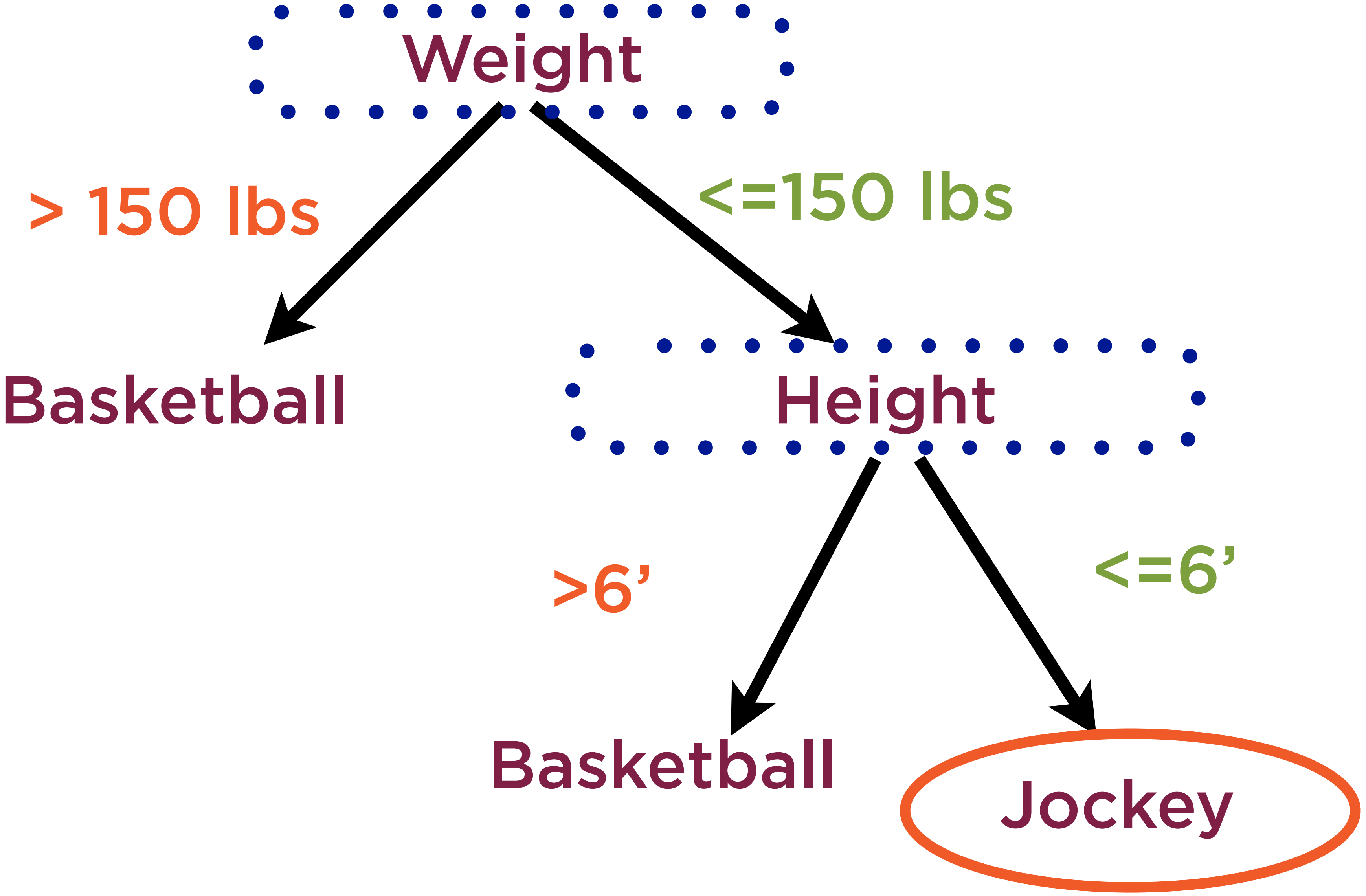
Minimizes “impurity” of each node

Impurity ~ misclassified data points

Impurity


Weight
> 150 lbs
<=150 lbs
Basketball
Height
8 basketball players, 2 jockeys from actual data
>6’
<=6’
Basketball
Jockey

Impurity



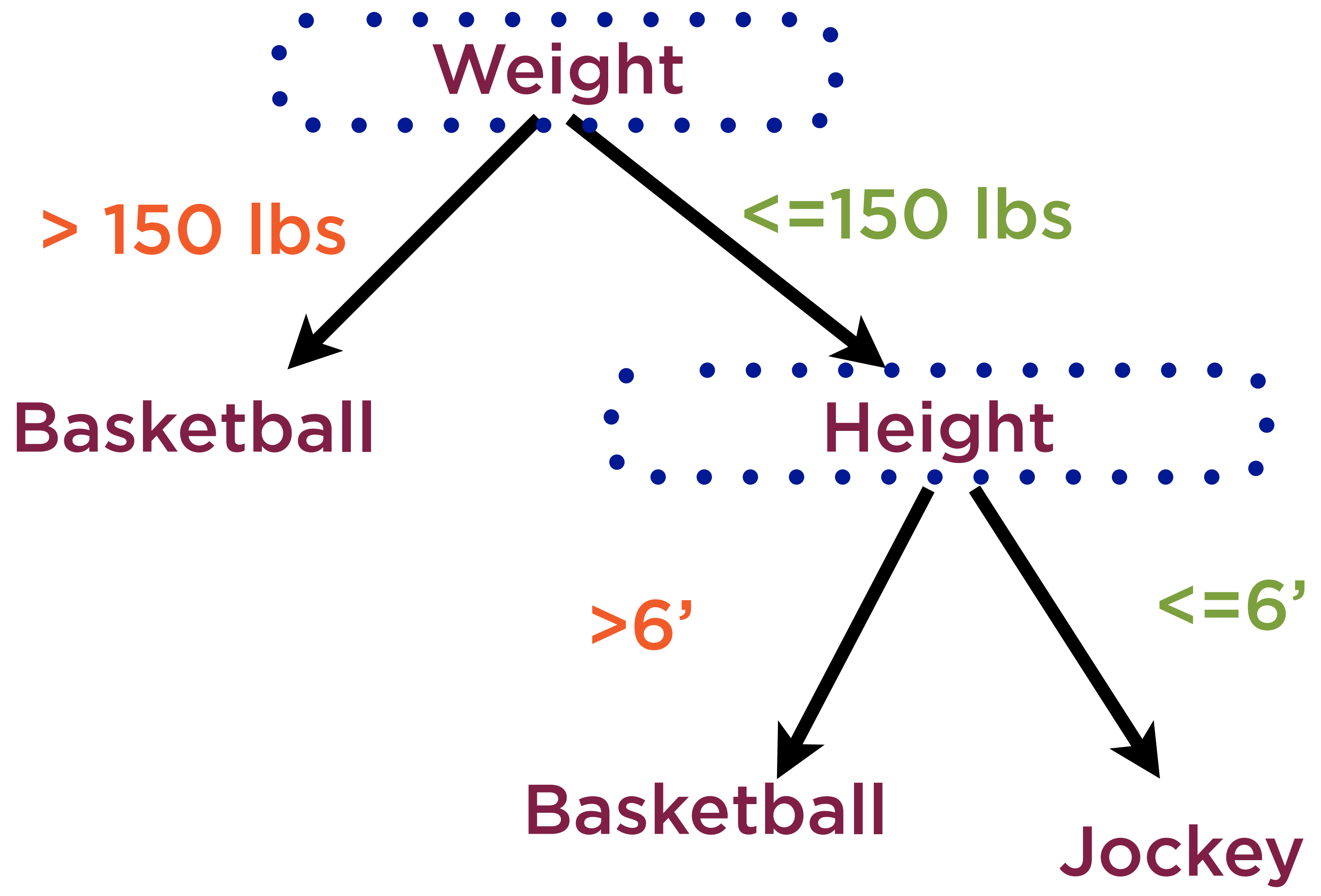
7 jockeys, 3 basketball players from actual data

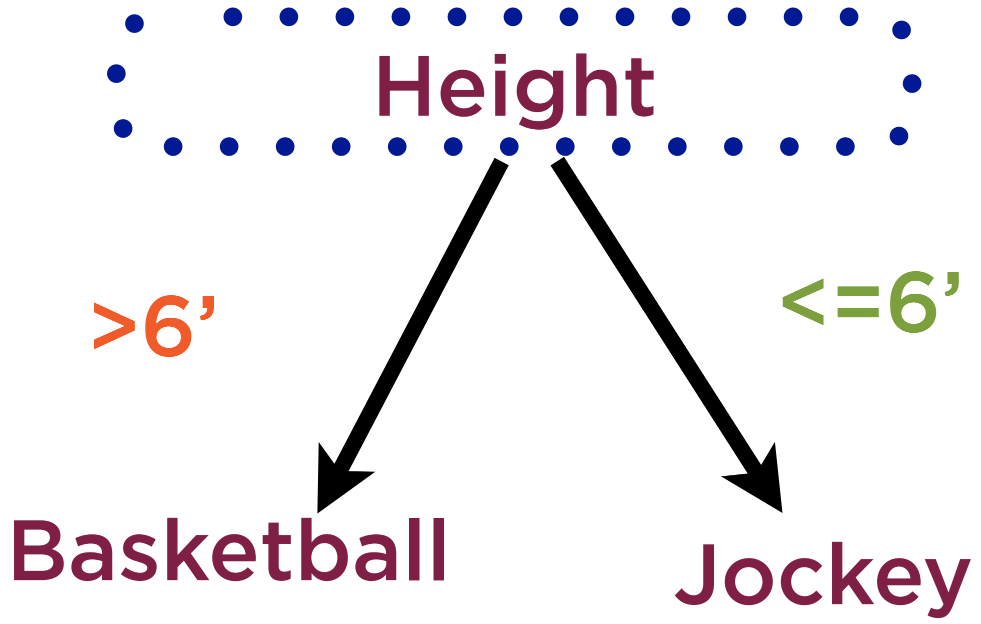
Two ways to measure impurity

- Gini impurity
- Entropy

Yield similar trees

Tree Construction

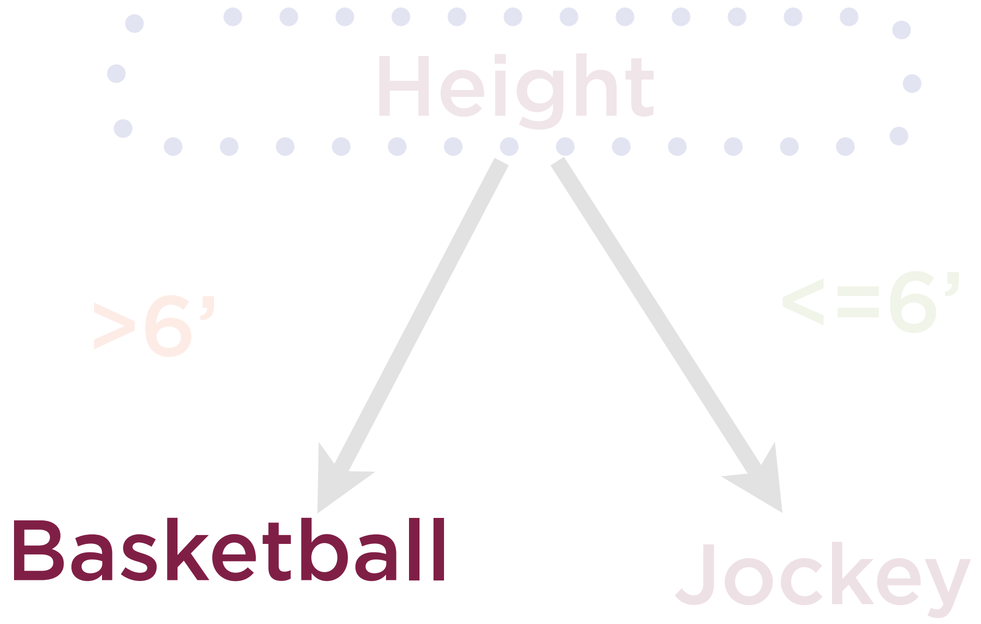




Gini Impurity

CART seeks to minimize Gini impurity at each node

Gini impurity is found from rule violations in training data



$G_i = 0.095$

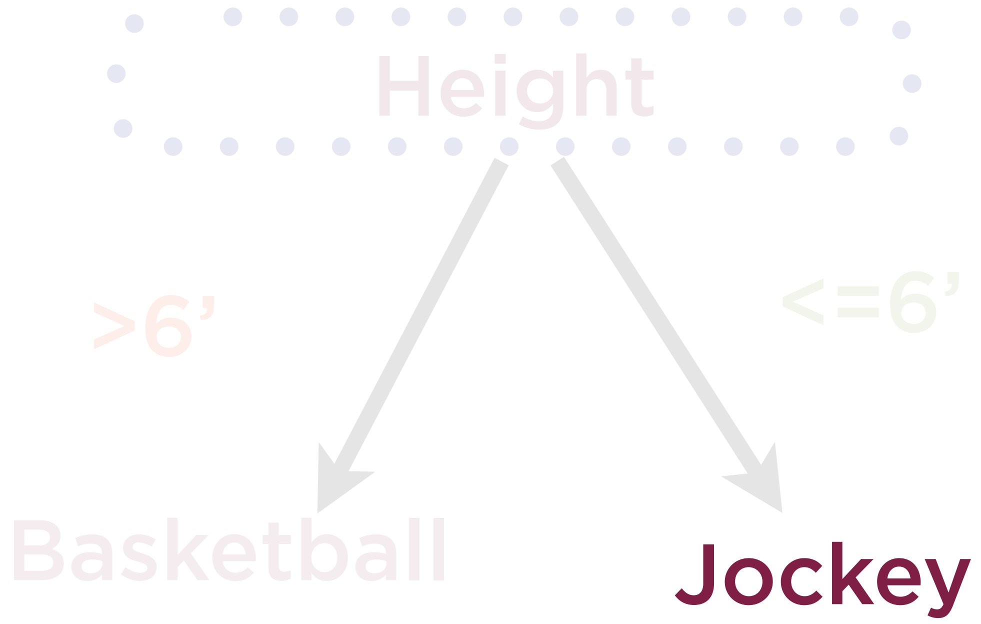
Gini Impurity

In training data:

100 samples with height > 6'

- 95 basketball players
- 5 jockeys

$G_i = 1 - (95\%)^2 - (5\%)^2 = 0.095$



$G_i = 0.0$

Completely pure

Gini Impurity

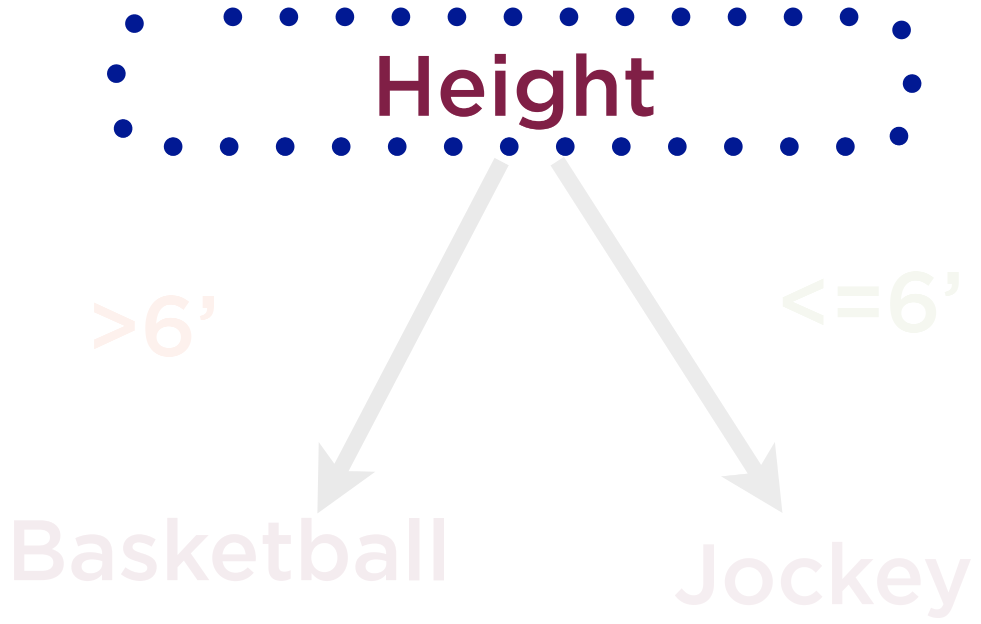
In training data:

100 samples with height <= 6’

- 0 basketball players
- 100 jockeys

$$G_i = 1 - (0\%)^2 - (100\%)^2 = 0$$

G_i = 0.49875



Gini Impurity

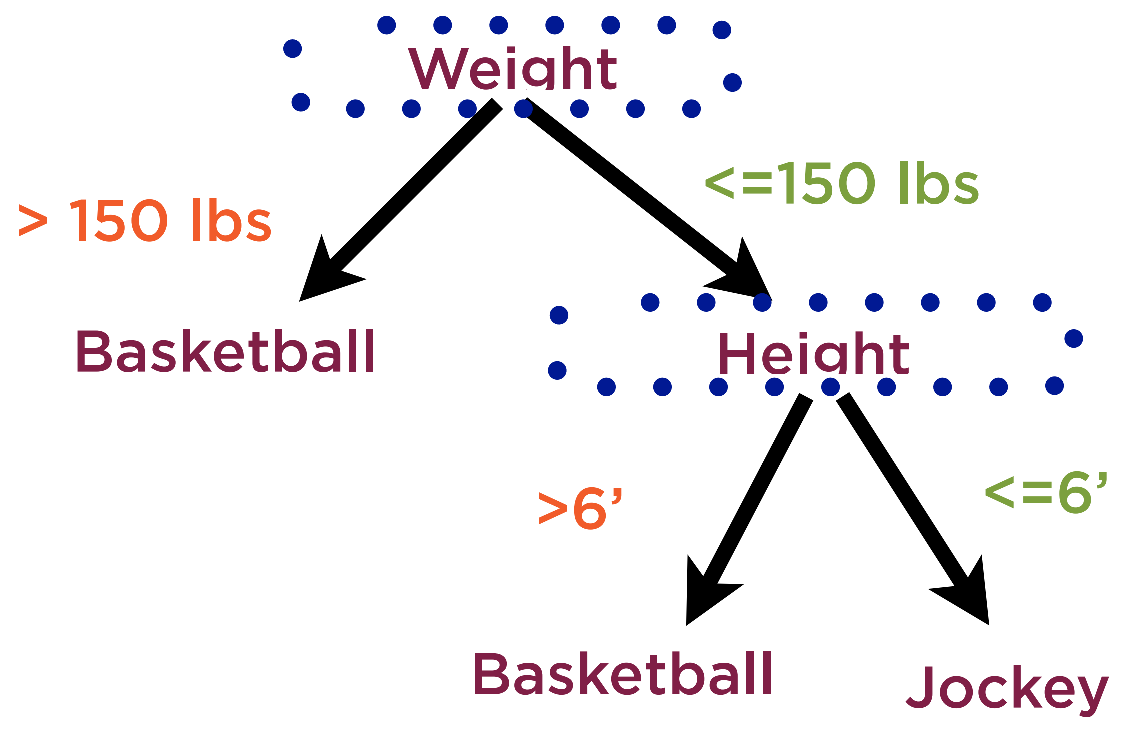
Gini impurity at node

200 samples (sum of the leaf nodes)

- 95 basketball players
- 105 jockeys

$$G_i = 1 - (95/200)^2 - (105/200)^2$$

$$= 0.49875$$

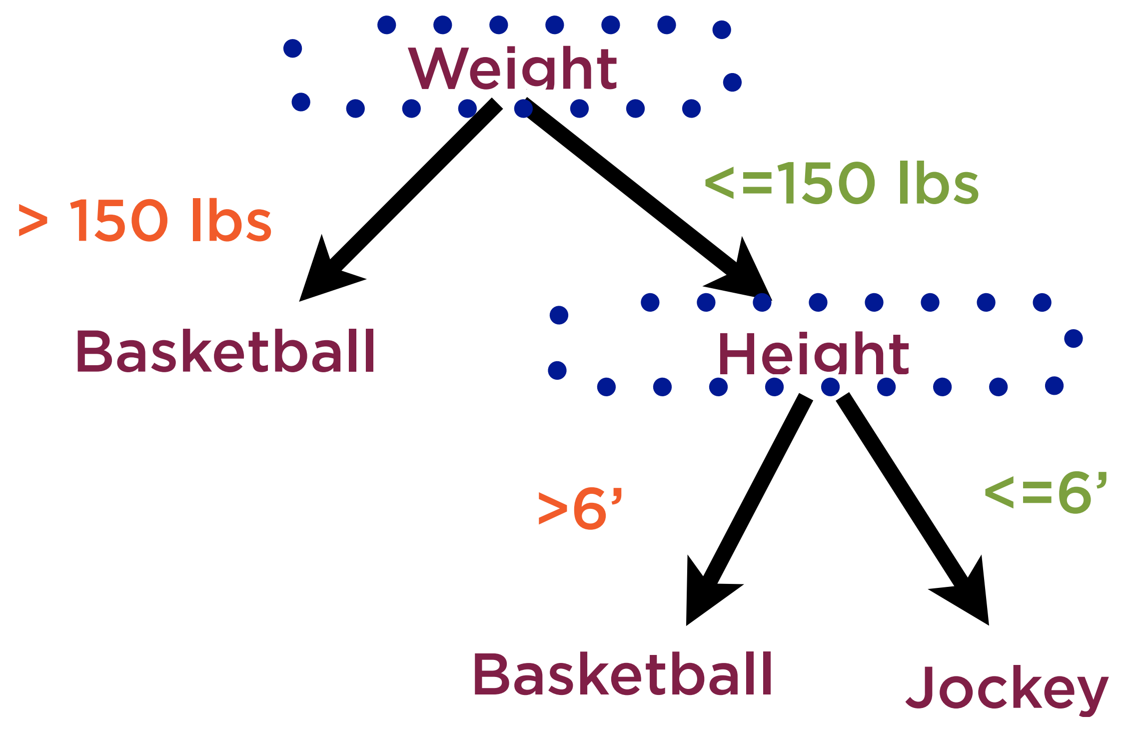


Advantages of Decision Trees

"White Box" ML ~ leverage experts

Non-parametric

- Little hyperparameter tuning
- Little data prep



Drawbacks of Decision Trees

Prone to overfitting

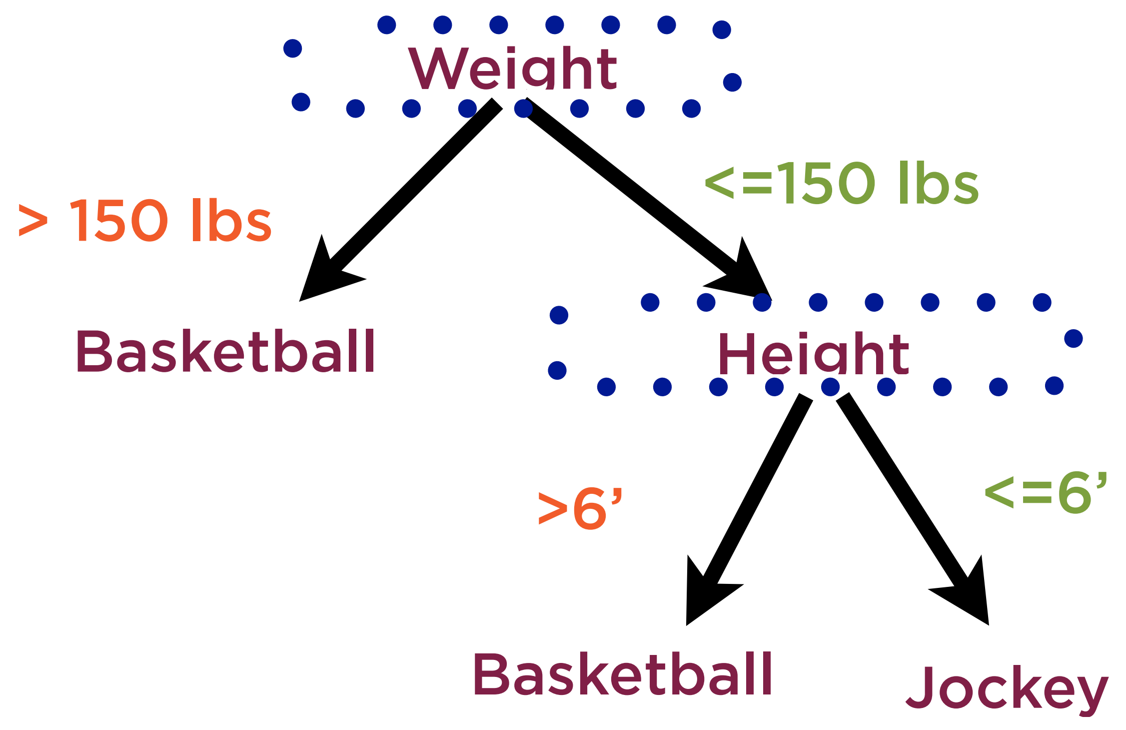
- Common risk with non-parametric

Unstable

- Small changes in data cause big changes in model

“If everyone in the room is thinking the same thing, then somebody isn’t thinking.”

General Patton



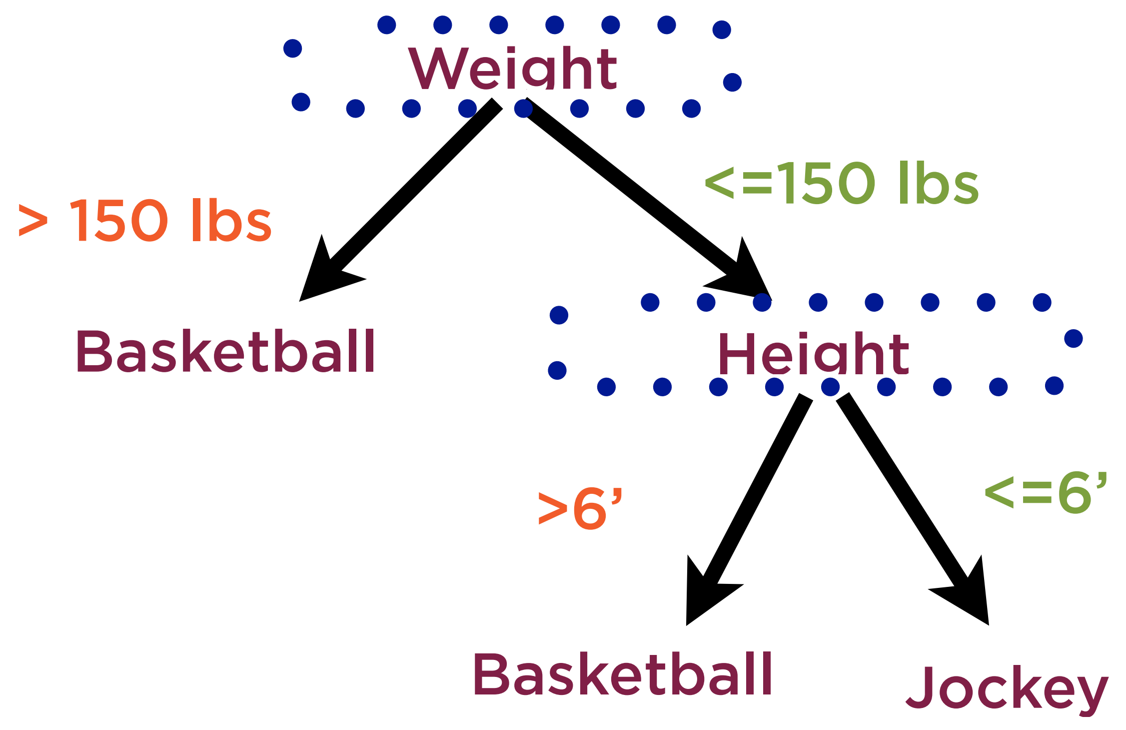
Random Forests

Train many decision trees

- each on random sample of data

Combine their output

- averaging for regression
- mode for classification



Random Forests

Extremely powerful technique

Example of ensemble learning

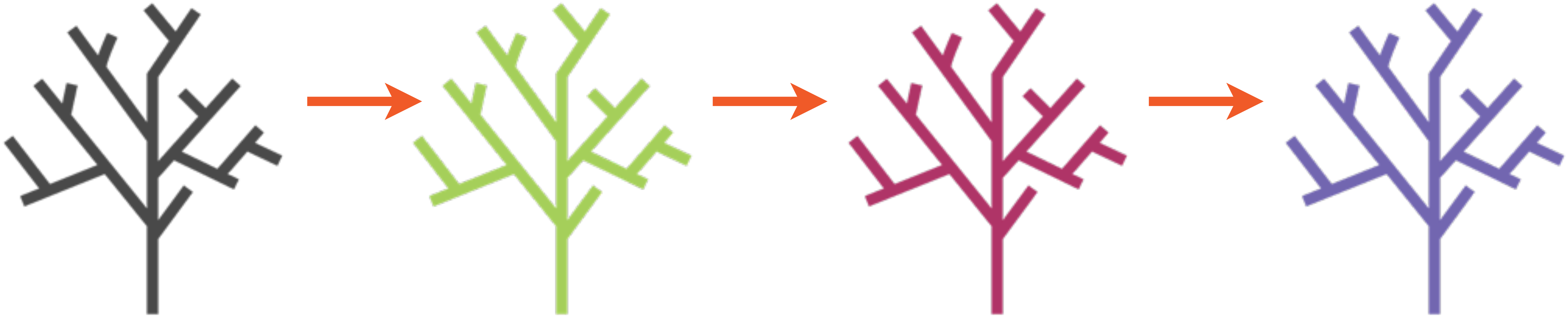
Individual trees should be as different as possible

Gradient Boosting

“Build up your weaknesses until they become your strong points.”

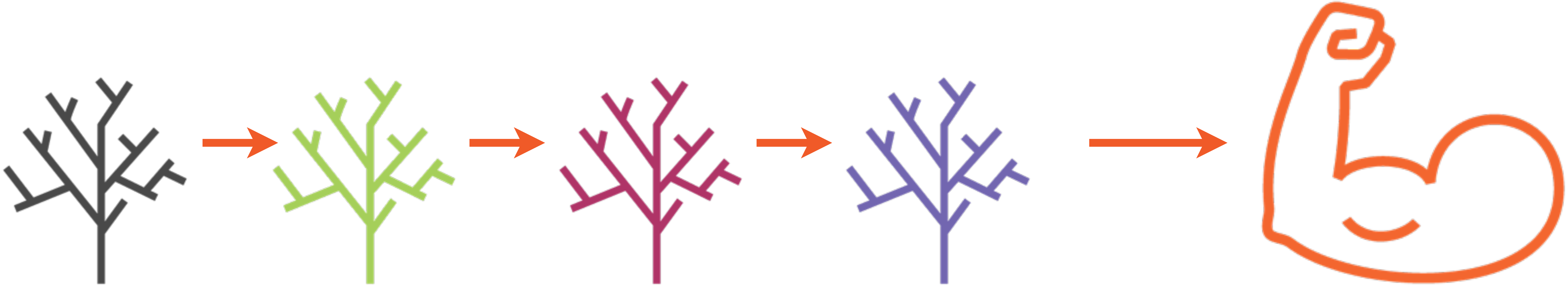
Knute Rockne

Gradient Boosting



Many machine learning models come together to work on the training data

Gradient Boosting



Many weak learners

Gradient Boosting

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

“Weak learners” individually

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

"Strong learner" when combined

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Residuals from Model 1

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1 x + e_1$$

Residuals from Model 1

What Model 1 fails to learn'

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Residuals from Model 1

What Model 1 fails to learn'

Learn in Model 2

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Focuses on what previous model failed to learn

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Residuals from Model 2

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Residuals from Model 2

What Model 2 fails to learn'

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Focuses on what previous model failed to learn

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

These residuals are now unlearnt

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + \mathbf{e_3}$$

Only these residuals are now unlearnt

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

"Weak learners" individually

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

"Ensemble" of learners

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\, x + e_3$$

Gradient Boosting

In practice: 100-200 weak learners, each learning from previous mistakes

Model 1:

$$y = A_1 + B_1 x + e_1$$

Model 2:

$$e_1 = A_2 + B_2 x + e_2$$

Model 3:

$$e_2 = A_3 + B_3 x + e_3$$

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

"Strong learner" when combined

Combined Model:

$$y = A_1 + A_2 + A_3 + (B_1 + B_2 + B_3)\,x + e_3$$

Gradient Boosting

Model 1:

$$y = A_1 + B_1x + e_1$$

Model 2:

$$e_1 = A_2 + B_2x + e_2$$

Model 3:

$$e_2 = A_3 + B_3x + e_3$$

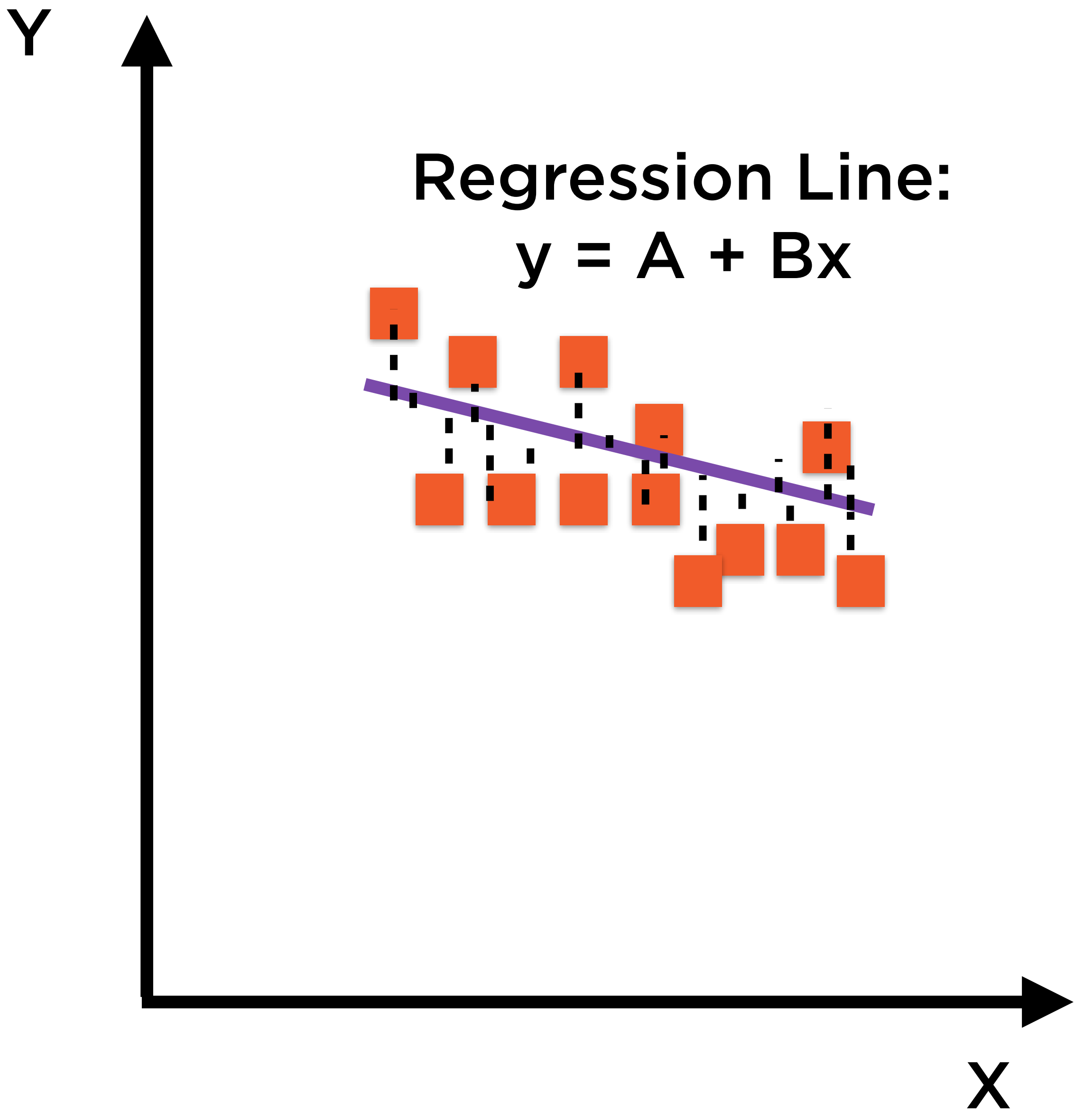
"Strong learner" when combined

Combined Model:

y = Sum of outputs of weak learners

Gradient Boosting will not work if the weak learners use MSE regression

Regression using Decision Trees work well



MSE Regression

MSE regression will not improve with Gradient Boosting

Residuals are uncorrelated with

- X variables
- predicted Y

Math just does not work out

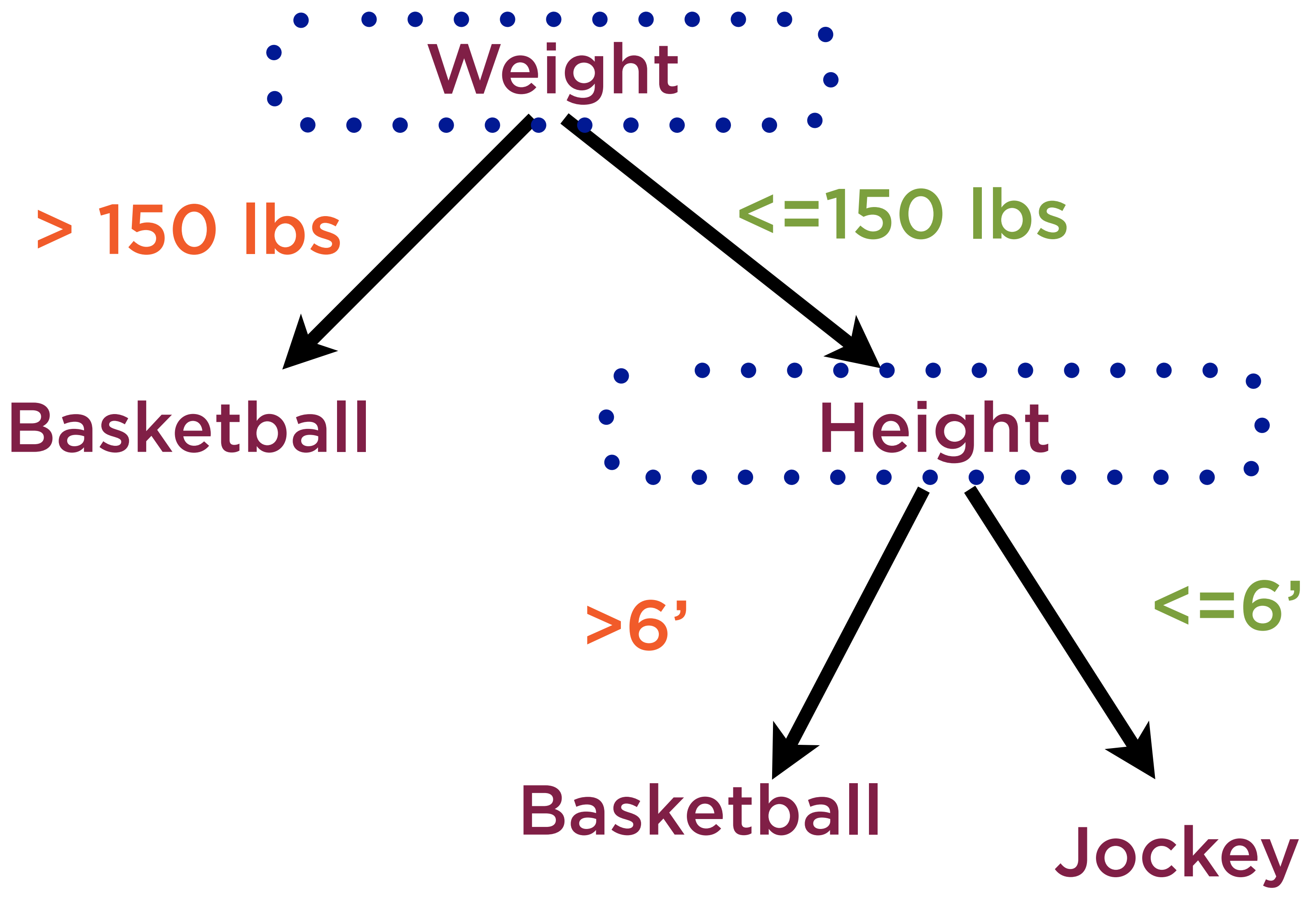
Decision Trees work great though

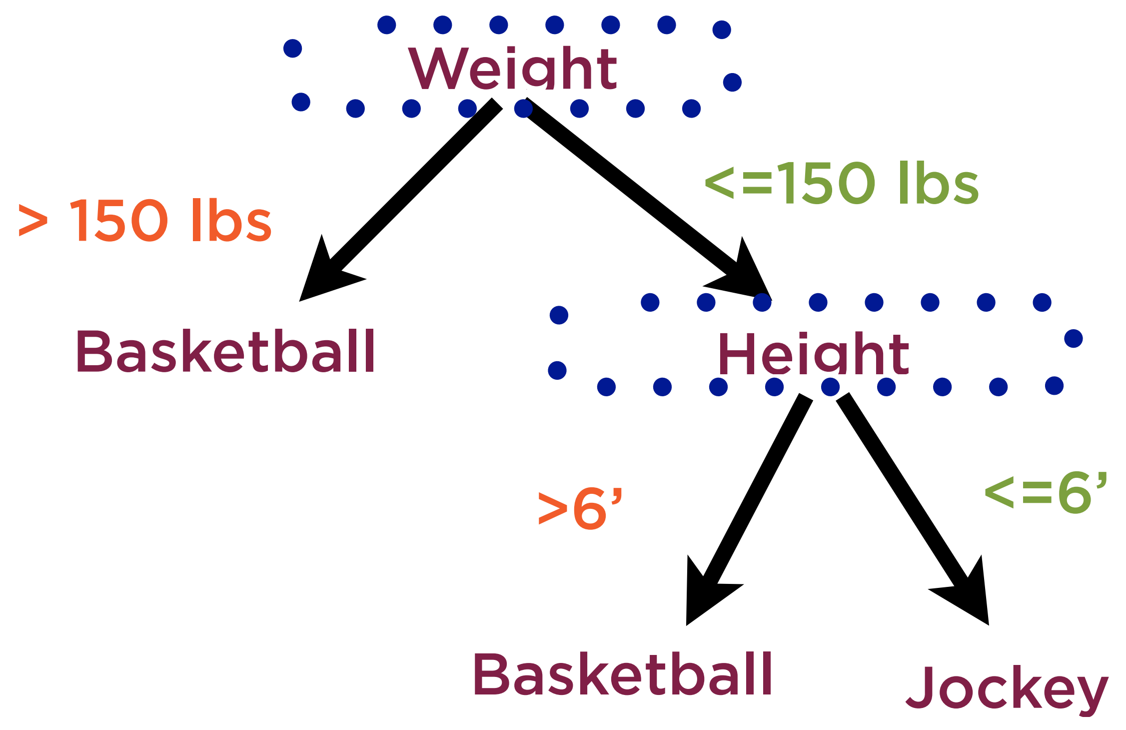
Non-parametric ML technique

Hyperparameter: how many levels?

Works beautifully with Gradient Boosting

Decision Tree





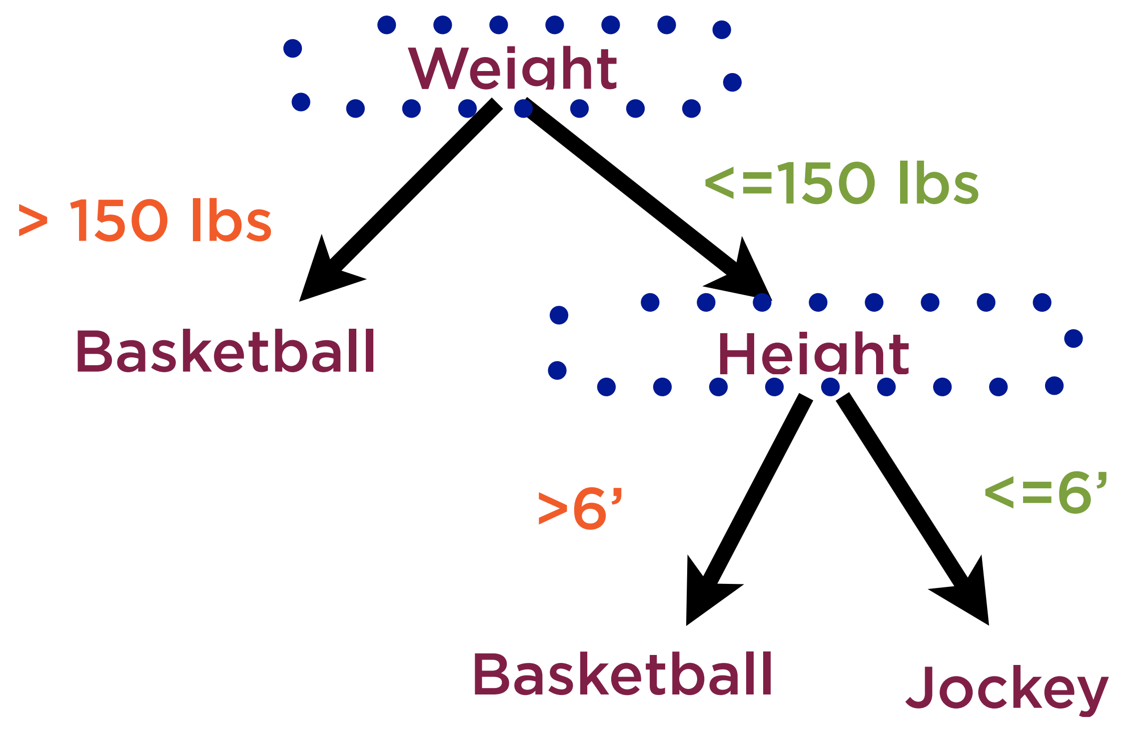
Number of Weak Learners

Gradient Boosting works best with a large number of weak learners

Large ~ several hundred

Early learners learn the most

Later learners learn from mistakes of early learners



Ensemble Learning

Gradient boosting is a form of Ensemble Learning

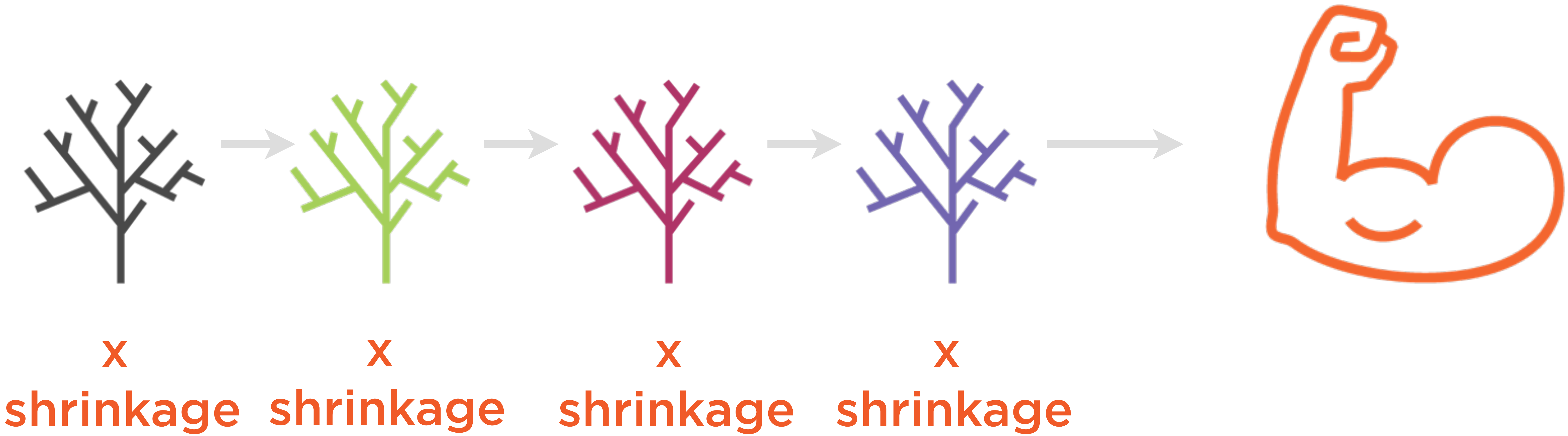
Ensemble ~ "together" in French

Aggregate many models together

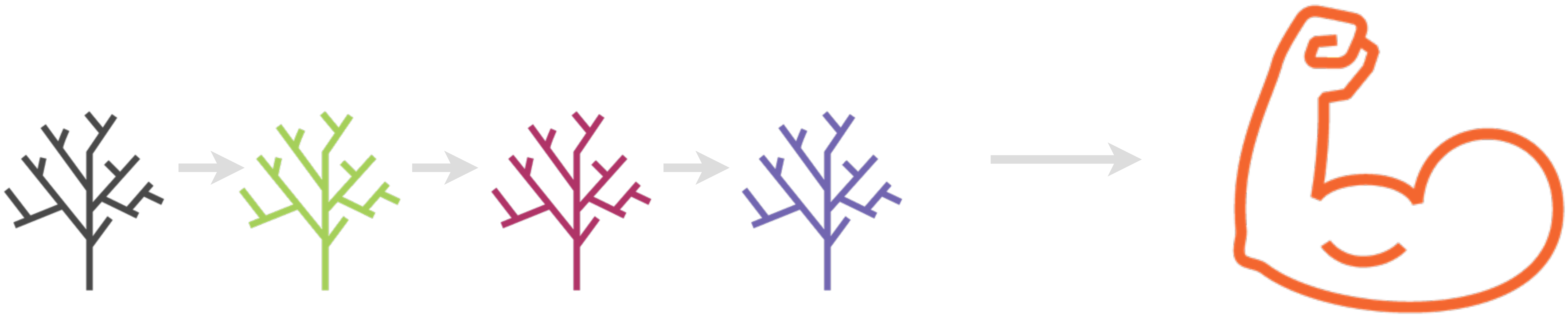
Standard regularization technique

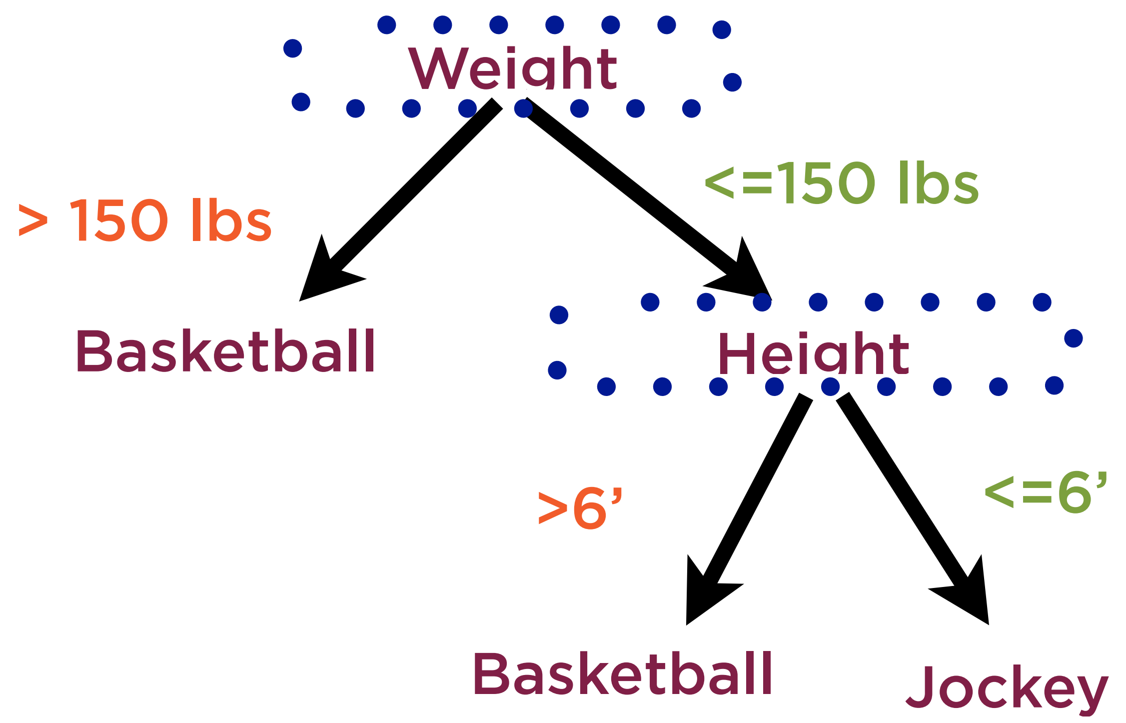
Reduces overfitting and variance error

Shrinkage Factor



Shrinkage Factor





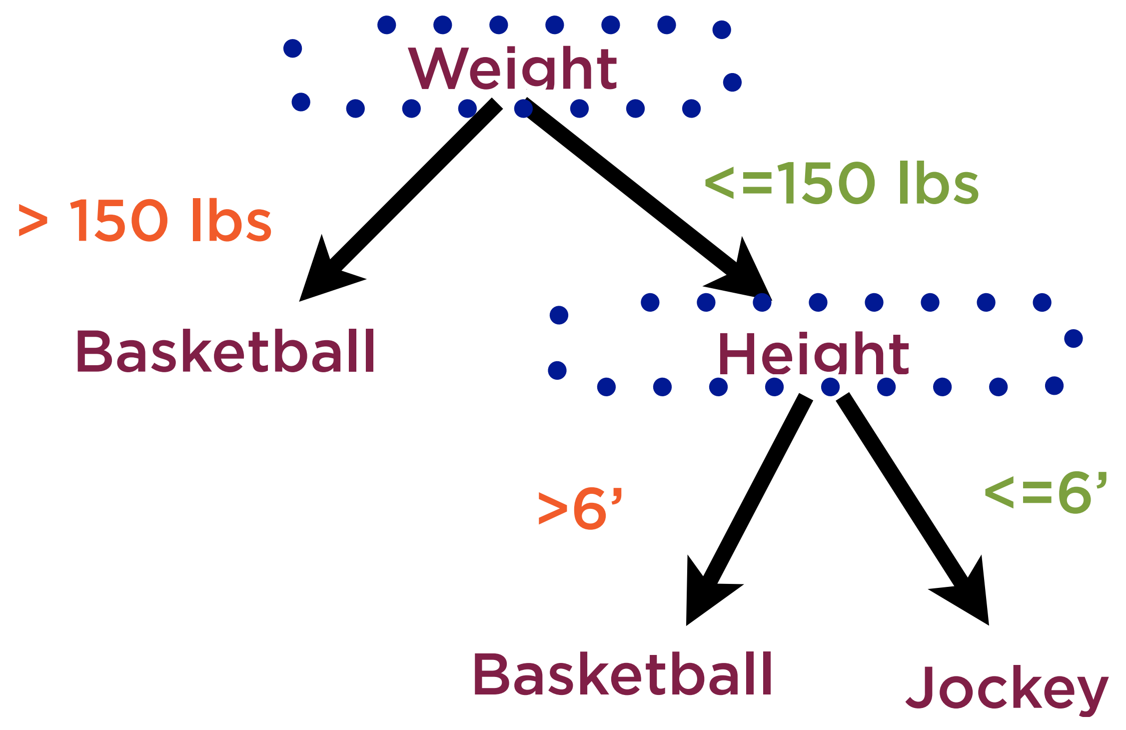
Shrinkage Factor

Scale output of each model by a constant factor

High shrinkage factor scales down importance of each learner

Slows down learning

Reduces overfitting

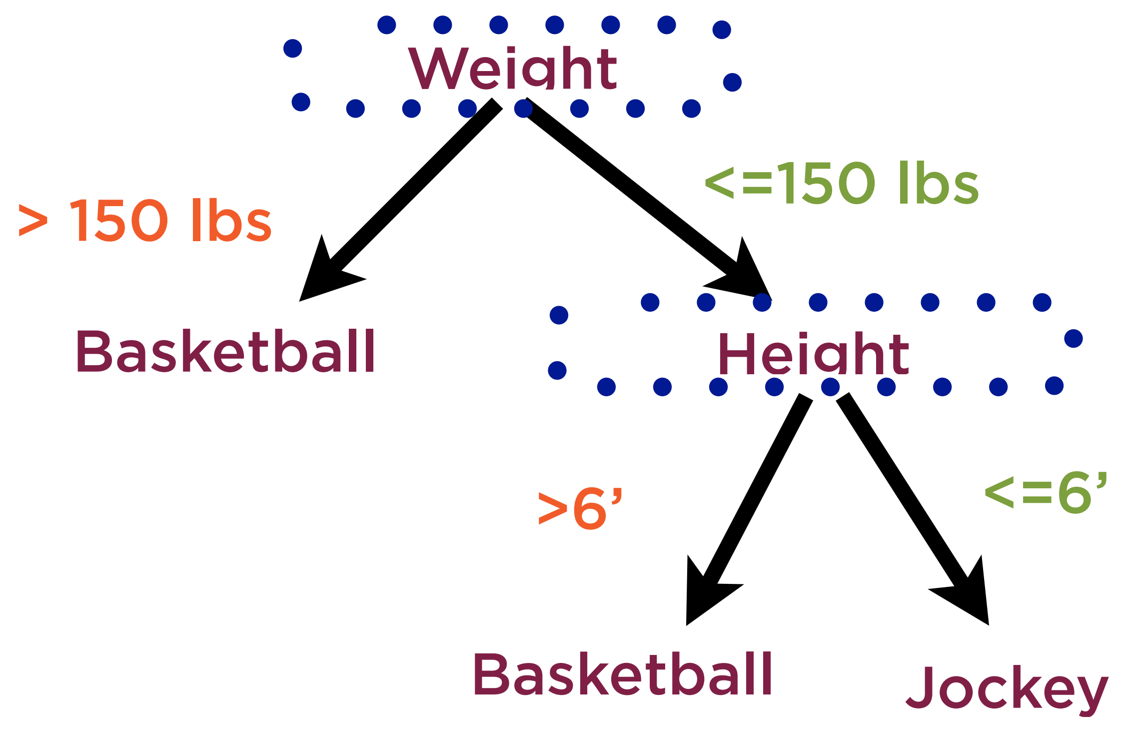


Shrinkage Factor

More learners ~ shrink a lot

Few learners ~ shrink a little

Typical values ~ 0.1, 1



Shrinkage Factor

Naive Gradient Boosting

- ShrinkageFactor = 1

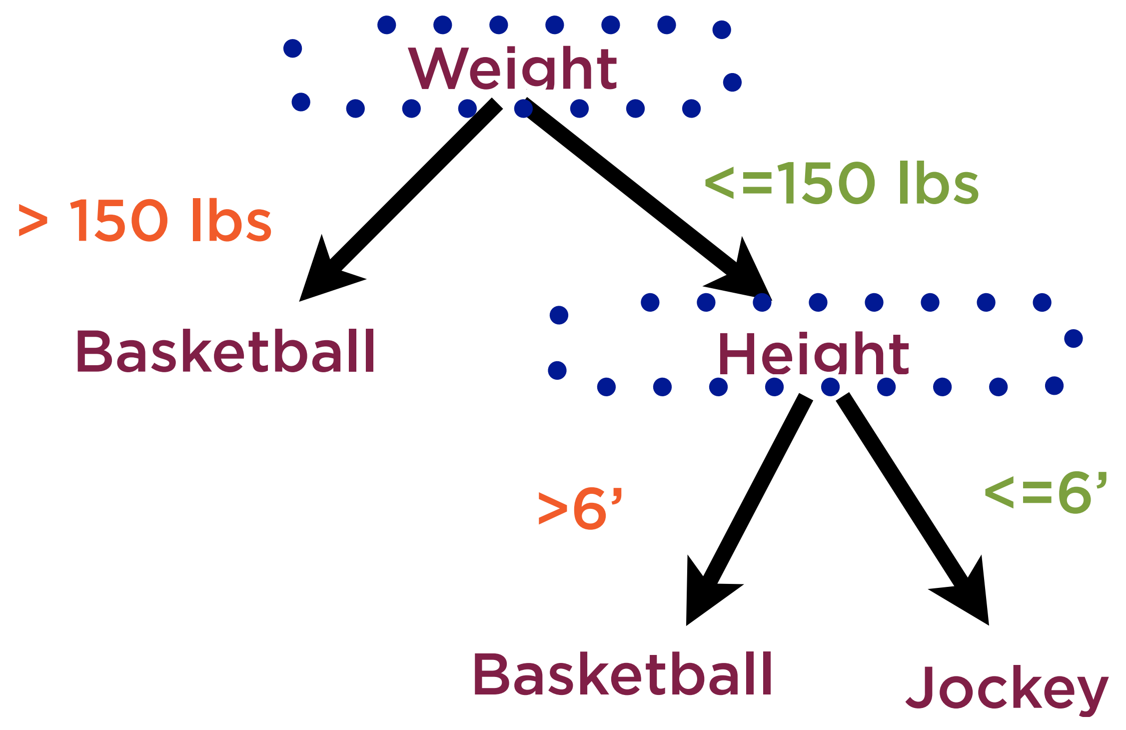
Gradient Boosting with Shrinkage

- ShrinkageFactor < 1

```
GradientBoostingRegressor(max_depth=4, n_estimators=200,
learning rate=0.1)
```

Gradient Boosted Regression Tree

scikit-learn has a high-level estimator object for this algorithm



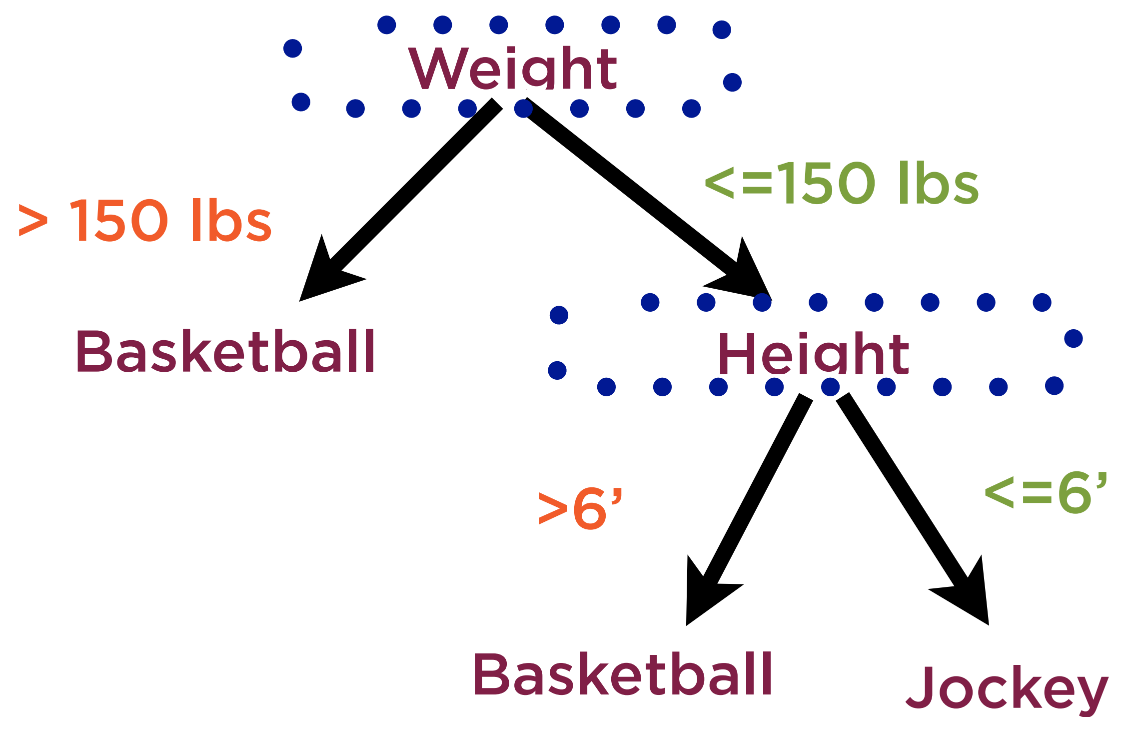
Grid Search in Scikit

Hyperparameter tuning is important

Best accomplished using Grid Search

Fancy name for nested for loops

Very convenient



Other Hyperparameters

subsample

- Train each tree on subset of training data selected at random
- By default, each tree trained on all training data

warm_start

- Preserve old trees to make learning faster during training

loss

- Loss function for decision trees

```
GradientBoostingRegressor(max_depth=4, n_estimators=200,
learning rate=0.1)
```

Gradient Boosted Regression Tree

Scikit-Learn has a high-level estimator object for this algorithm

```
GradientBoostingRegressor(max_depth=4, n_estimators=200,
learning rate=0.1)
```

How Many Weak Learners?

Each weak learner is a decision tree

```
GradientBoostingRegressor(max_depth=4, n_estimators=200,
learning rate=0.1)
```

How Many Weak Levels?

The maximum depth of each individual decision tree

```
GradientBoostingRegressor(max_depth=4, n_estimators=200,
learning rate=0.1)
```

How much weight for each weak learner?

Here each of 200 weak learner models is assigned the weight of 0.1

Jockey or Basketball Player?

Jockeys

Tend to be light to meet horse carrying limits

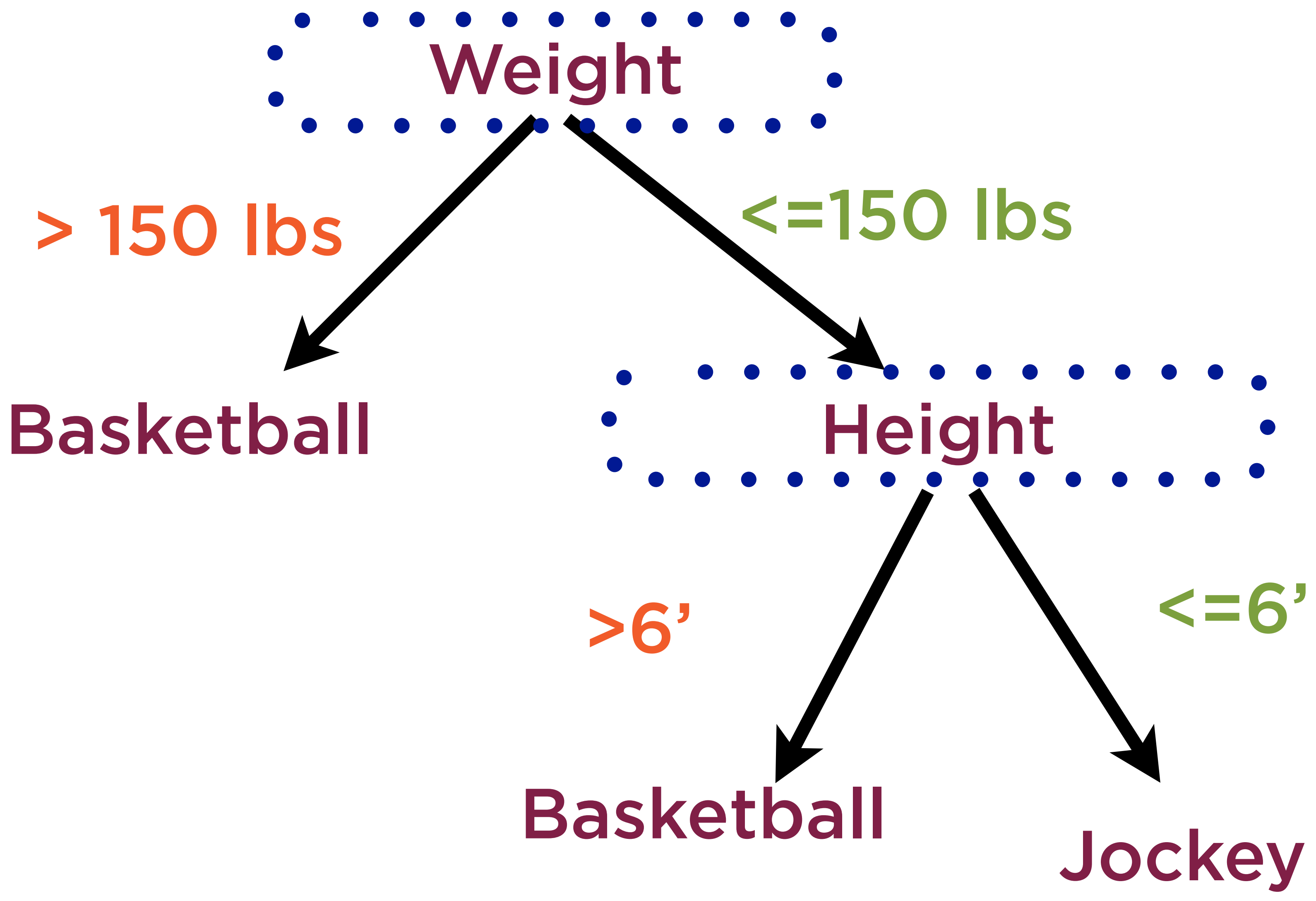
Basketball Players

Tend to be tall, strong and heavy

Fit knowledge into rules

Each rule involves a threshold

Decision Tree



Demo

Regression using Gradient Boosting in scikit-learn

Summary

Support Vector Machines are a very popular ML technique for classification

SVMs can work on text as well as images

Often ML models can come together as an ensemble to build a stronger model

Gradient boosting uses decision trees to build a better regression model