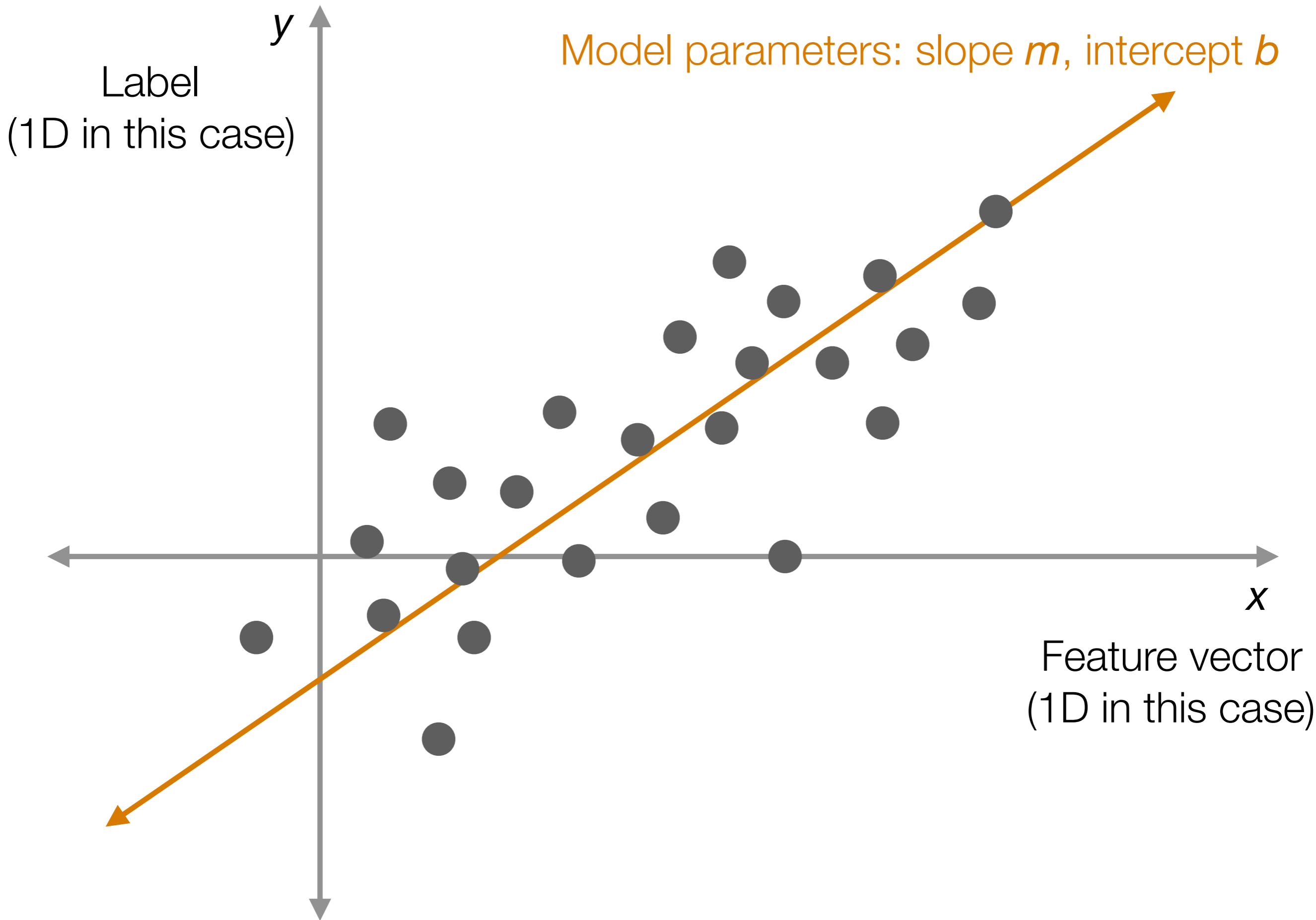


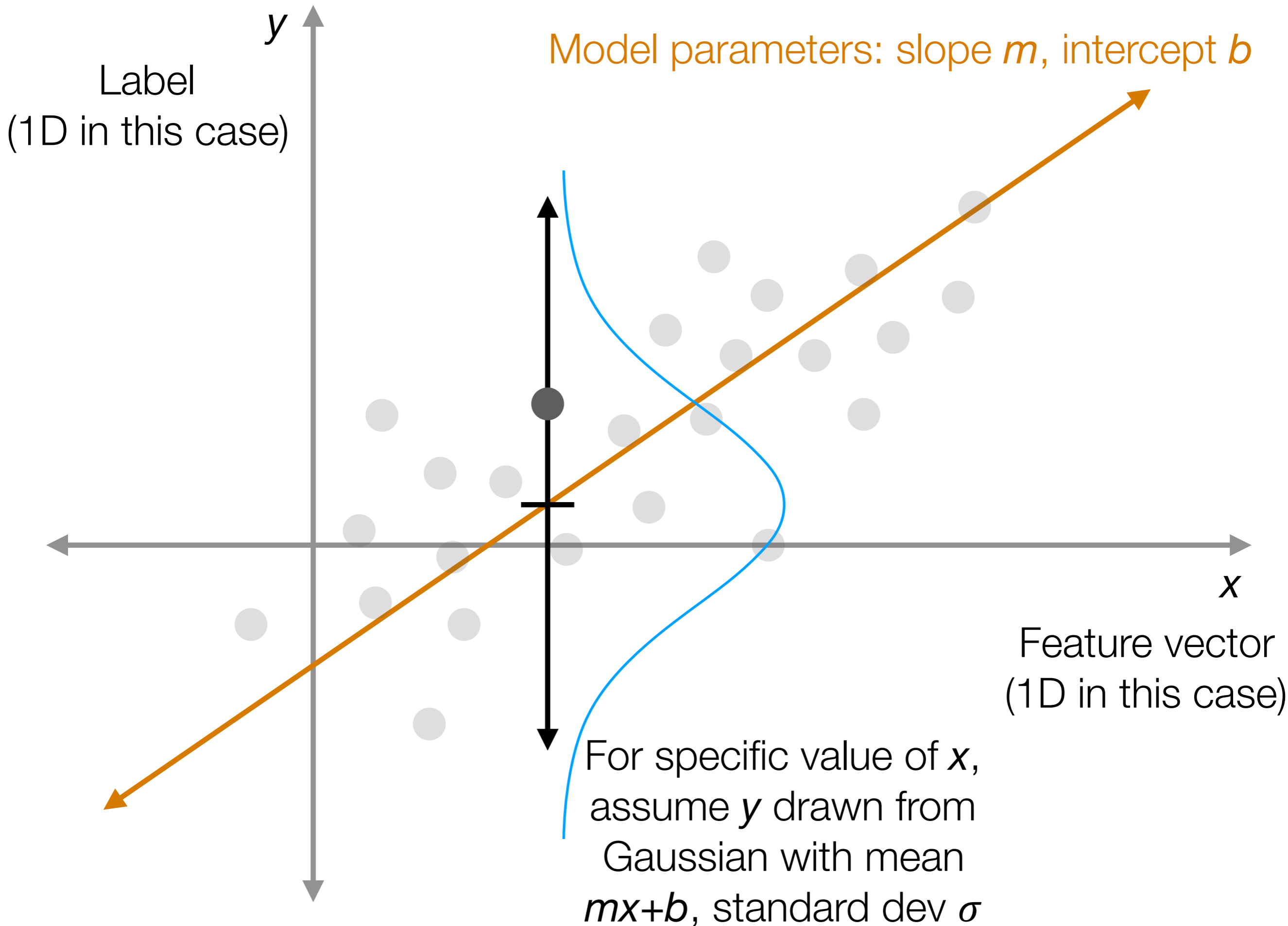
**94-775 Lecture 9:
Prediction and Validation,
Illustrated Using Support Vector
Classification**

George Chen

**You've seen a generative
model before for prediction**

Linear regression!





Predictive Data Analysis

Training data

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

Goal: Given new feature vector x , predict label y

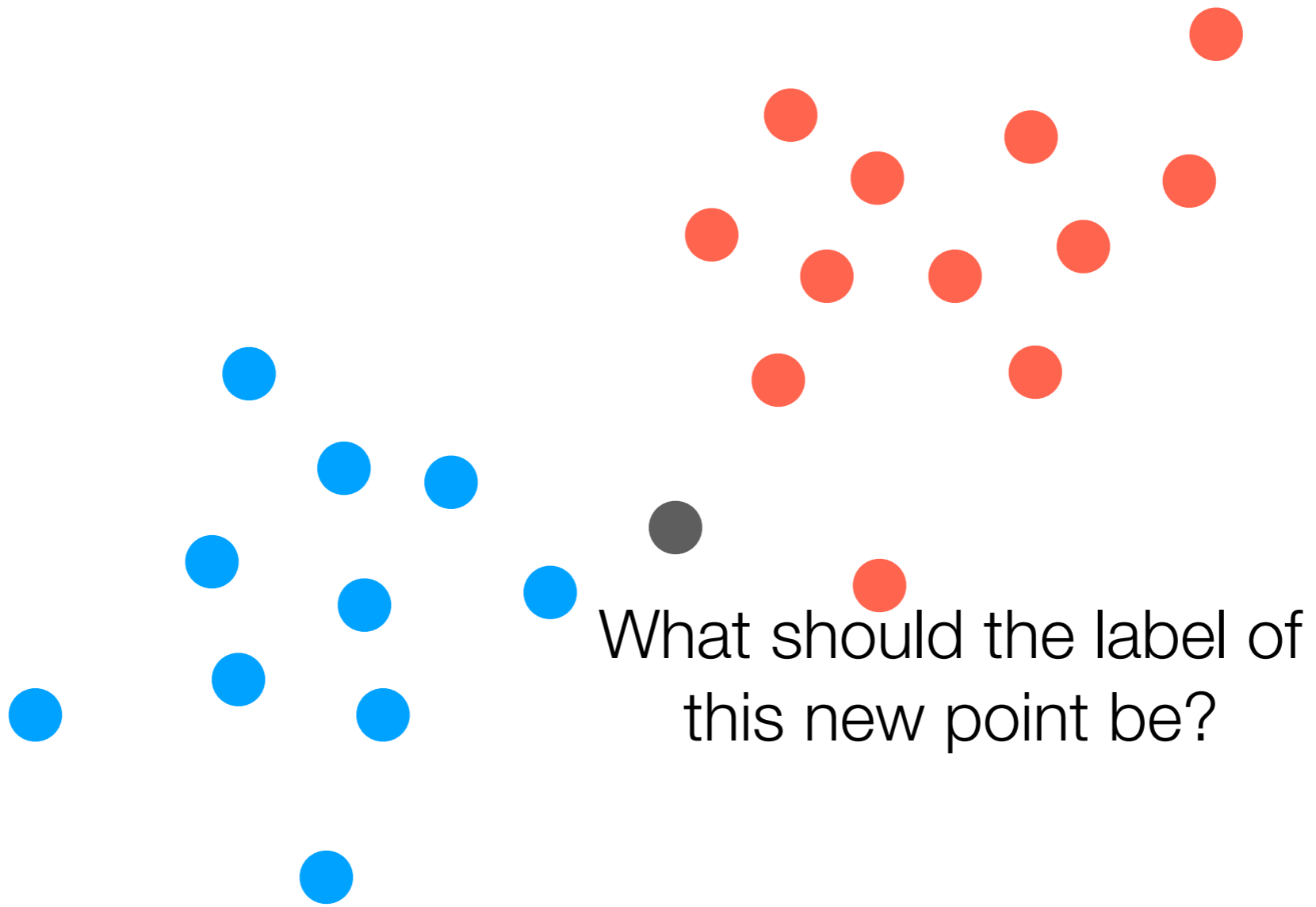
- y is discrete (such as colors **red** and **blue**)
→ prediction problem is called **classification**
- y is continuous (such as a real number)
→ prediction problem is called **regression**

A giant zoo of methods

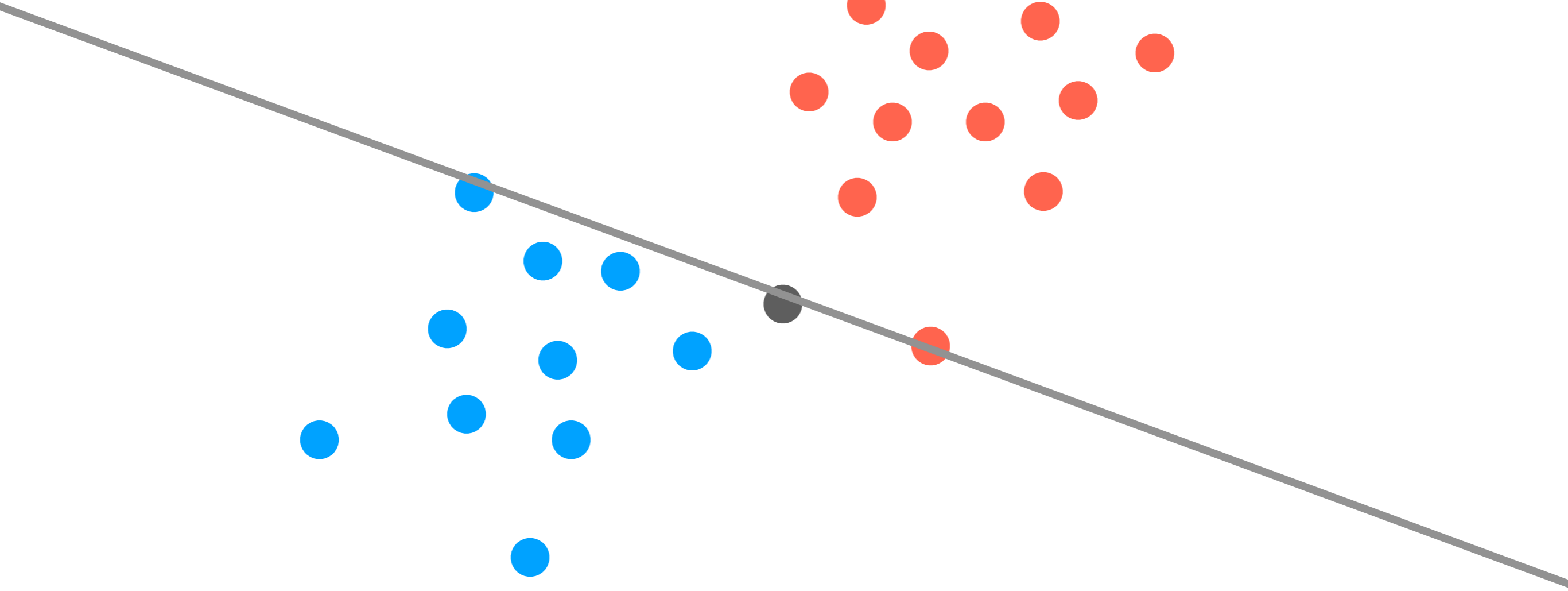
Generative Models

- Hypothesize a specific way in which data are generated
- After learning a generative model:
 - We can generate new synthetic data from the model
 - Usually generative models are probabilistic and we can evaluate probabilities for a new data point
- In contrast to generative models, there are *discriminative* methods that just care about learning a prediction rule

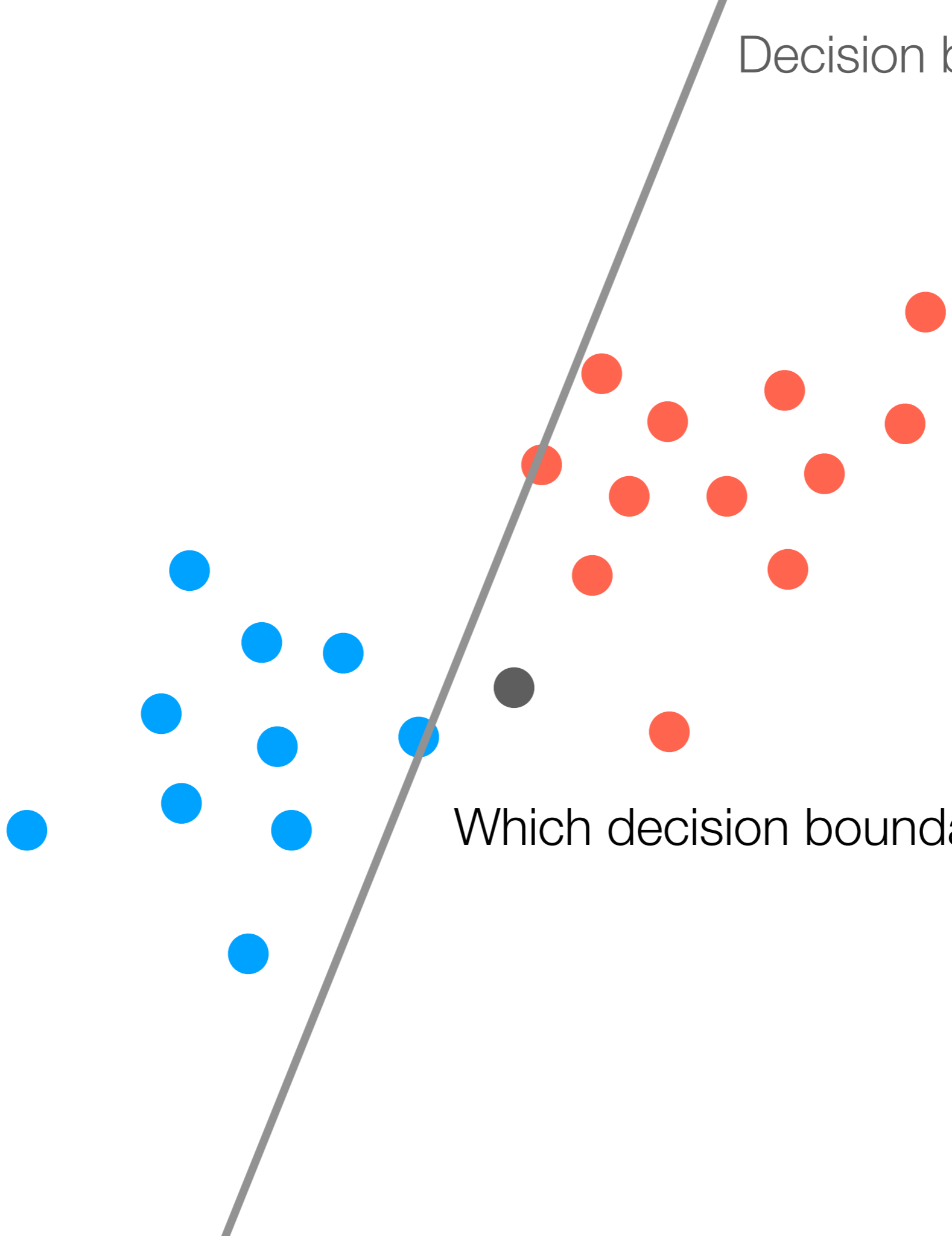
**Example of a Discriminative
Method: Support Vector
Machines**



Decision boundary

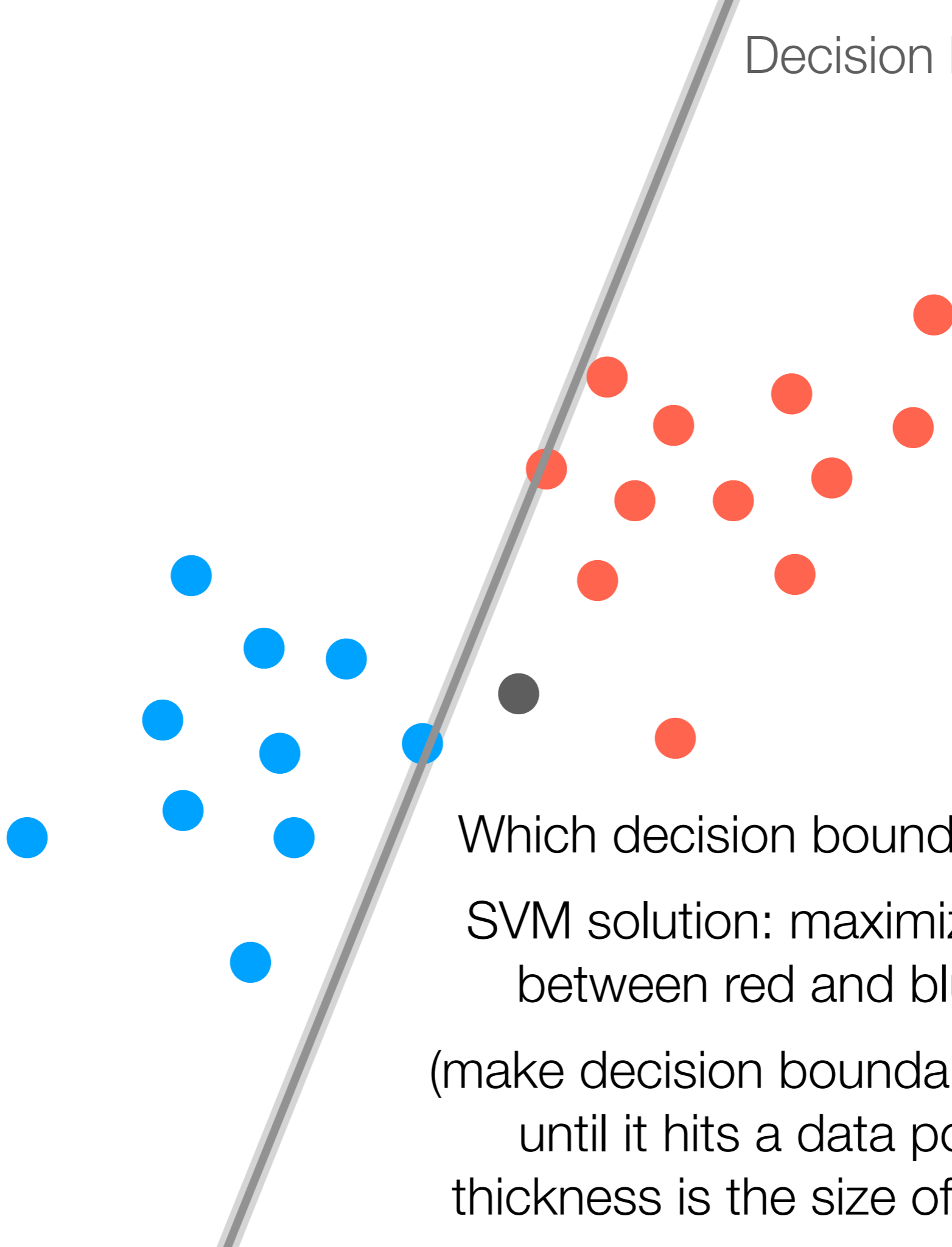


Decision boundary



Which decision boundary is best?

Decision boundary



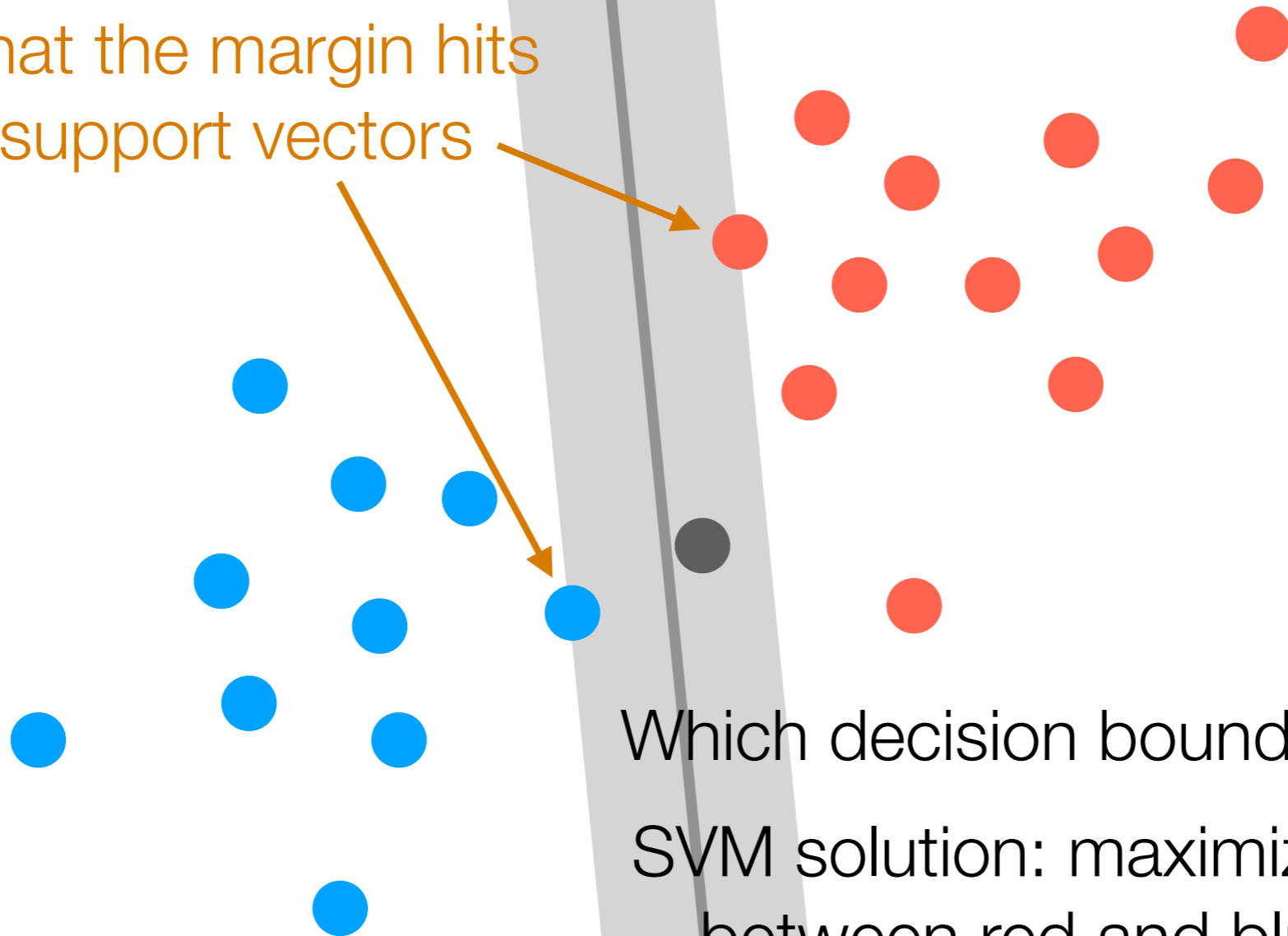
Which decision boundary is best?

SVM solution: maximize “margin”
between red and blue points

(make decision boundary line thicker
until it hits a data point—this
thickness is the size of the margin)

Decision boundary

The points that the margin hits
are called support vectors



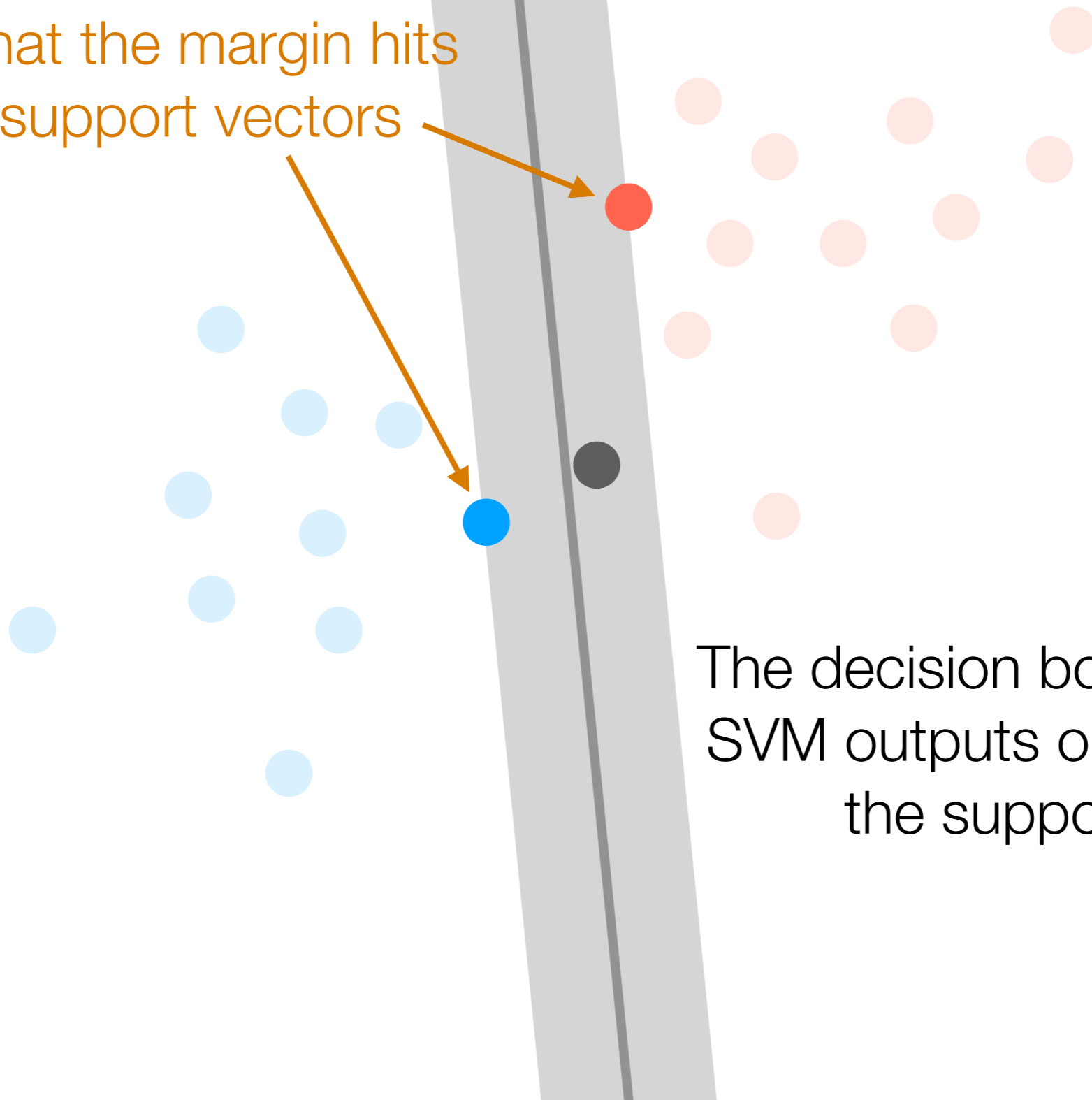
Which decision boundary is best?

SVM solution: maximize “margin”
between red and blue points

(make decision boundary line thicker
until it hits a data point—this
thickness is the size of the margin)

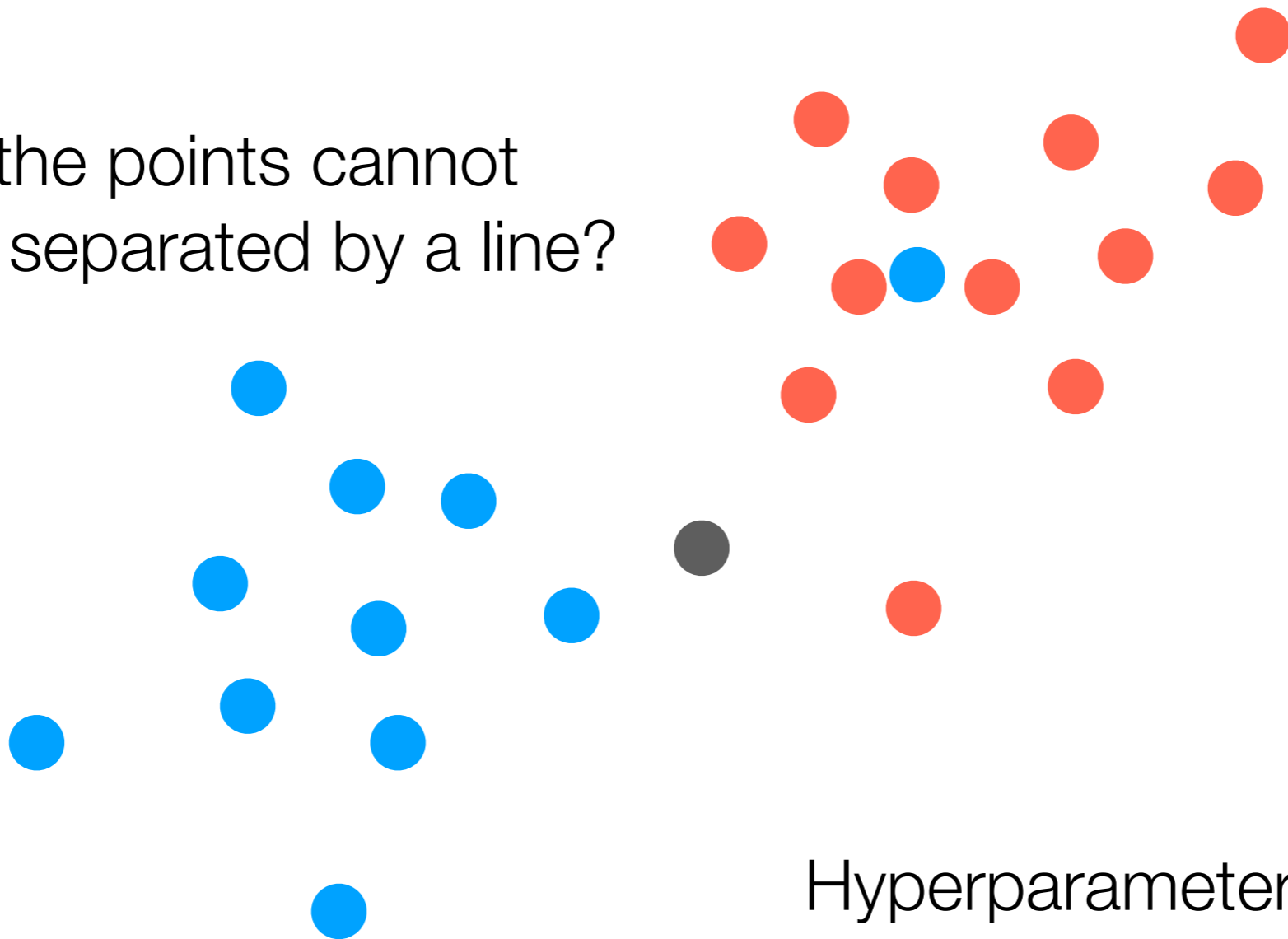
Decision boundary

The points that the margin hits
are called support vectors



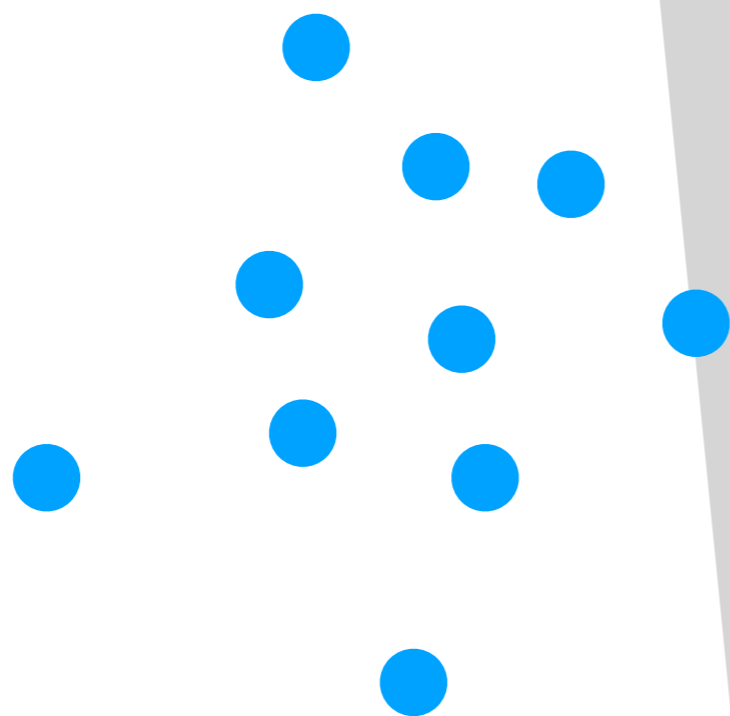
The decision boundary that the
SVM outputs only depends on
the support vectors

What if the points cannot actually be separated by a line?

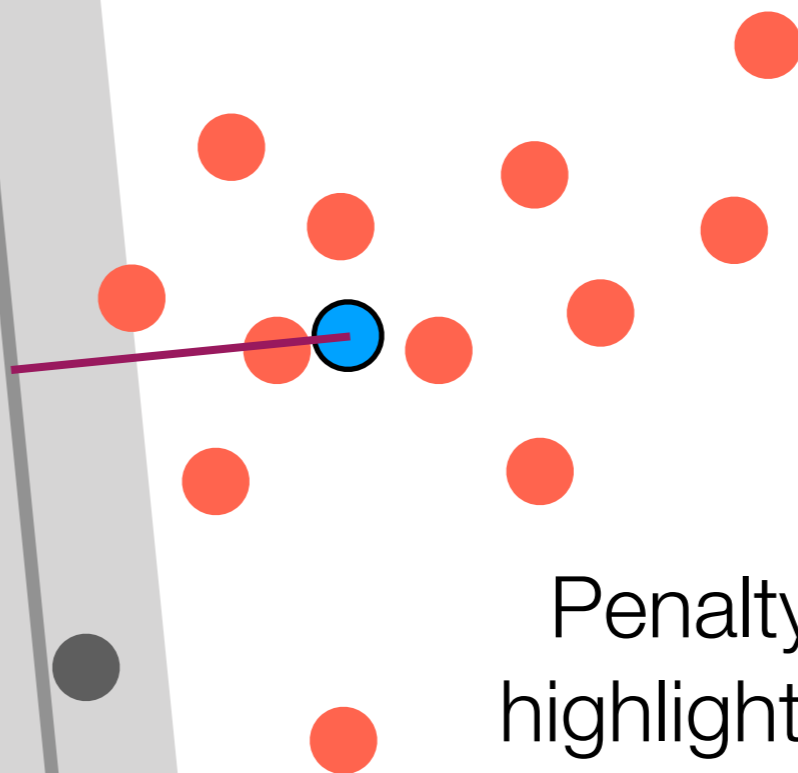


Hyperparameter C is a penalty for a point being on the wrong side of the decision boundary

What if the points cannot actually be separated by a line?



Larger $C \rightarrow$ work harder to fit all points

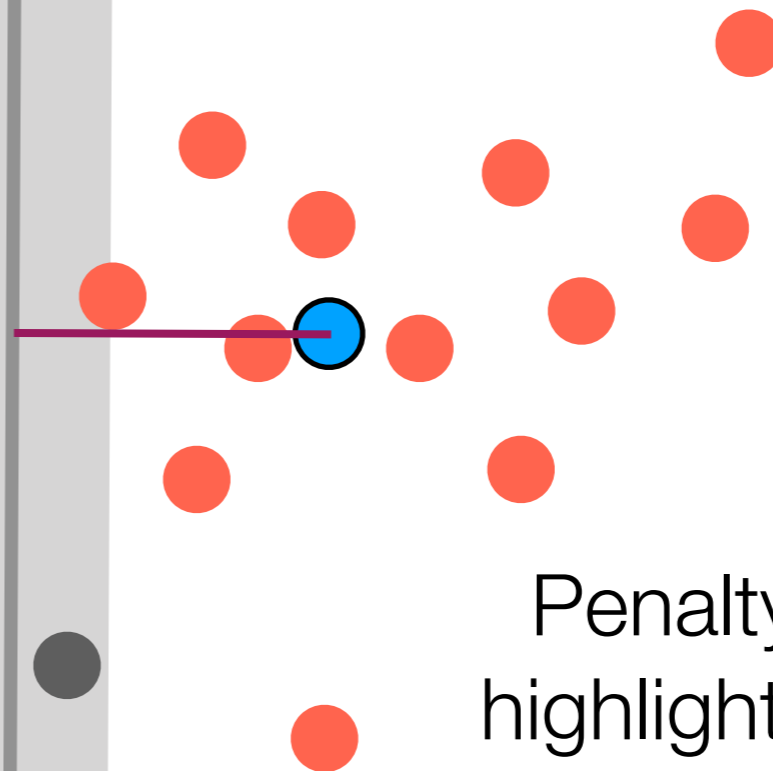
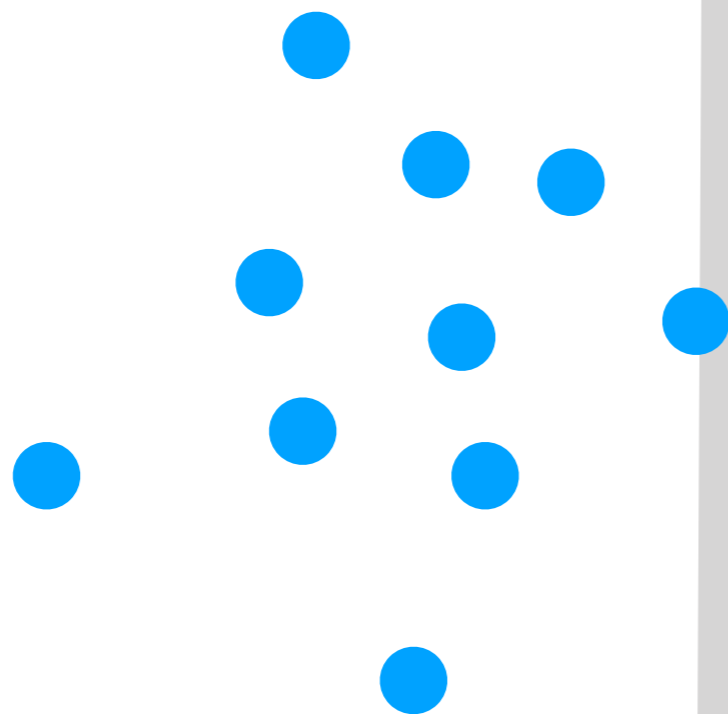


Penalty incurred for highlighted blue point:
 $C \times$ length of purple line

Hyperparameter C is a penalty for a point being on the wrong side of the decision boundary

C-Support Vector Classification

What if the points cannot actually be separated by a line?



Penalty incurred for highlighted blue point:
 $C \times \text{length of purple line}$

Larger $C \rightarrow$ work harder to fit all points

Hyperparameter C is a penalty for a point being on the wrong side of the decision boundary

C-Support Vector Classification

- Basic version measures distance using Euclidean distance
 - Turns out to correspond to measuring similarity between two points by taking their dot product
 - This is called **linear svm**
- Can instead use a different similarity function (“kernel” function) instead (popular choice: Gaussian kernel, also called “radial basis function” kernel)
 - This is called **kernel svm**
- Also: support vector *regression* (these are all in sklearn)

How do we choose C ?

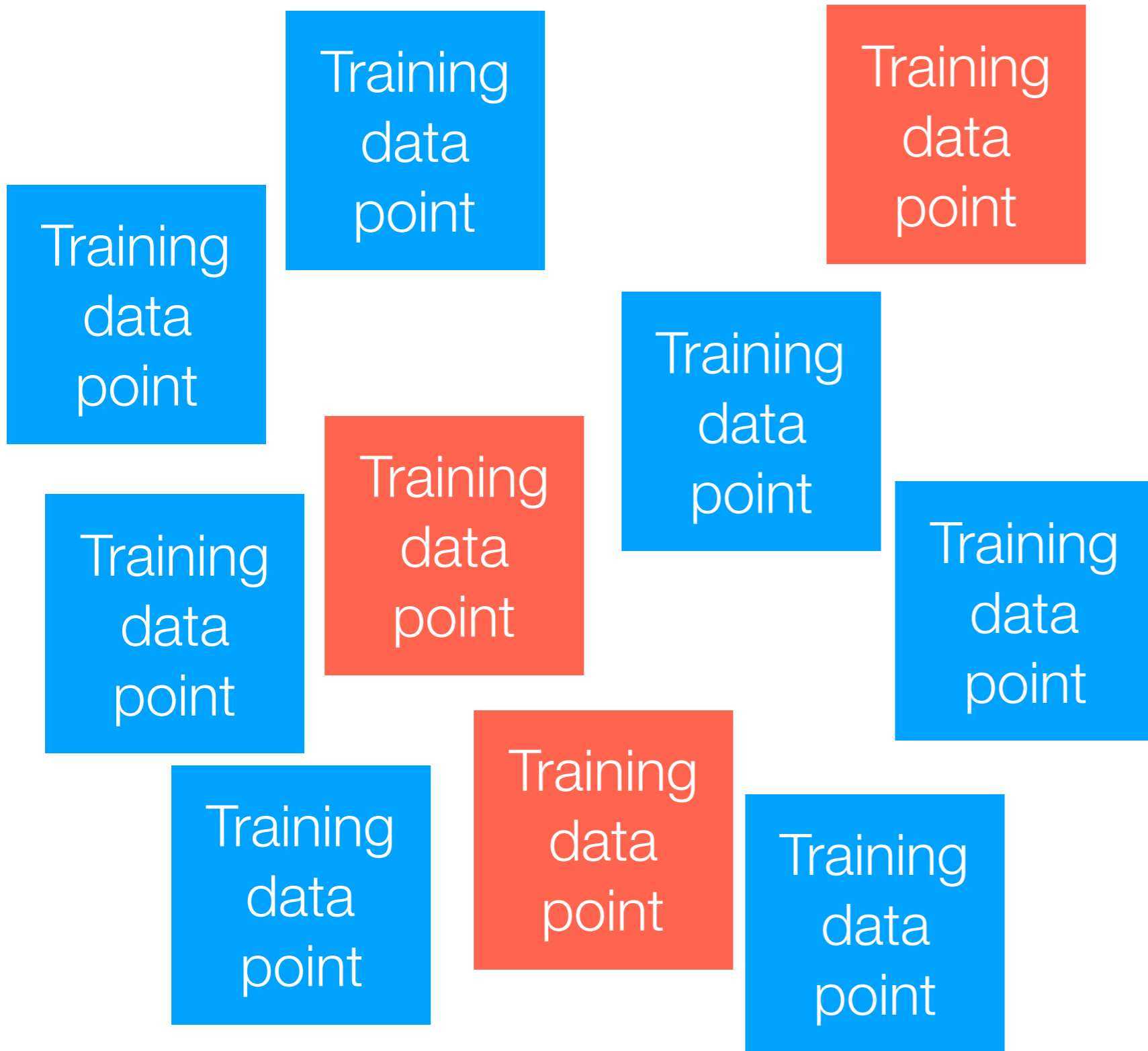
What I'll describe next can be used to select hyperparameter(s) for *any* prediction method

First: How do we assess how good a prediction method is?

Hyperparameters vs. Parameters

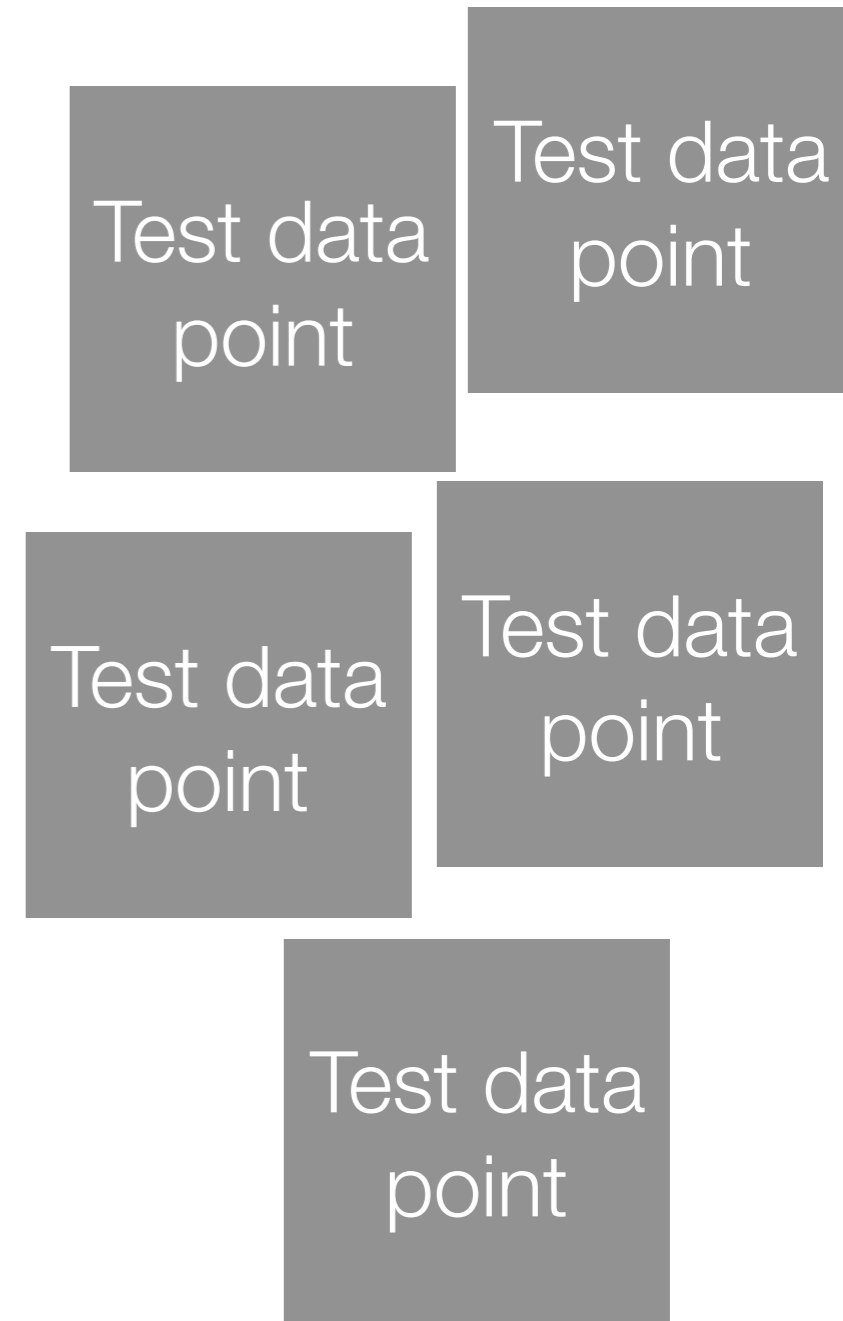
- We fit a model's parameter to training data (terminology: we “learn” the parameters)
- We choose values of hyperparameters and they do *not* get fit to training data
- Example: Gaussian mixture model
 - Hyperparameter: number of clusters k
 - Parameters: cluster probabilities, means, covariances
- Example: C -SVM classification
 - Hyperparameter: C
 - Parameters: coefficients for the hyperplane equation

Training data



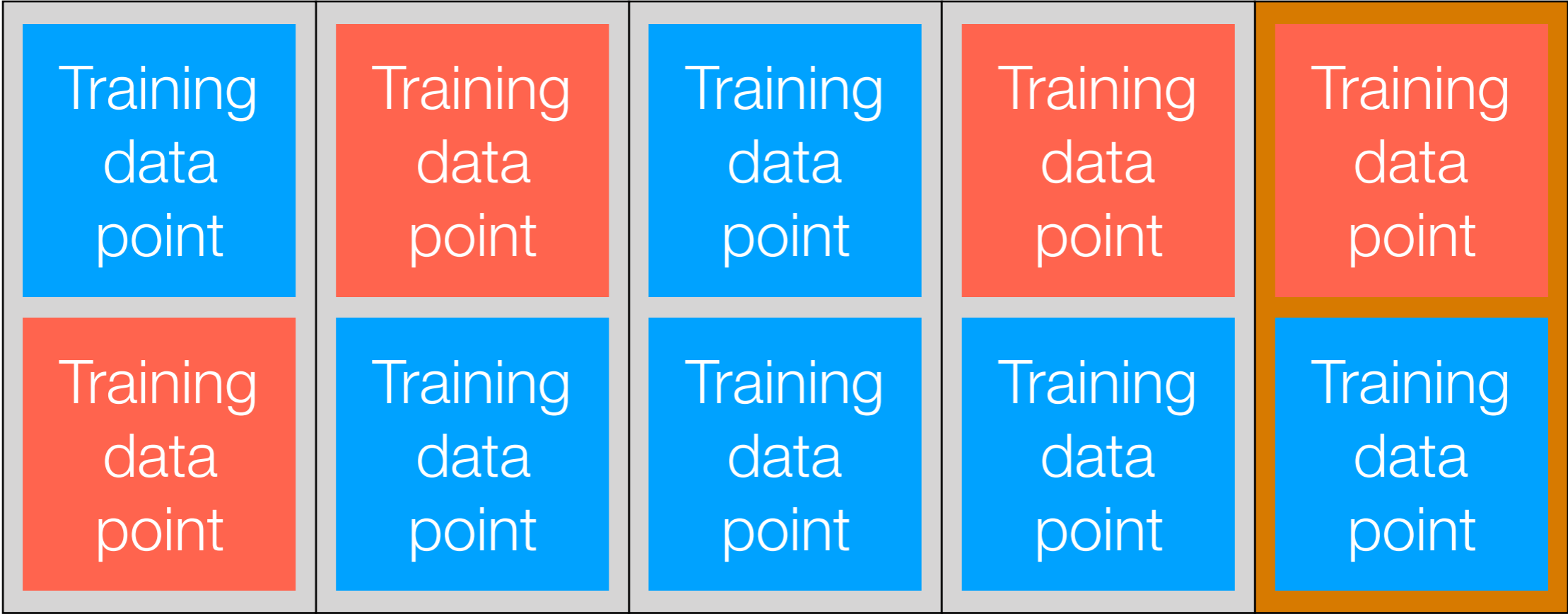
Example: Each data point is an email and we know whether it is spam/ham

Want to classify these points correctly



Example: future emails to classify as spam/ham

Predicted labels

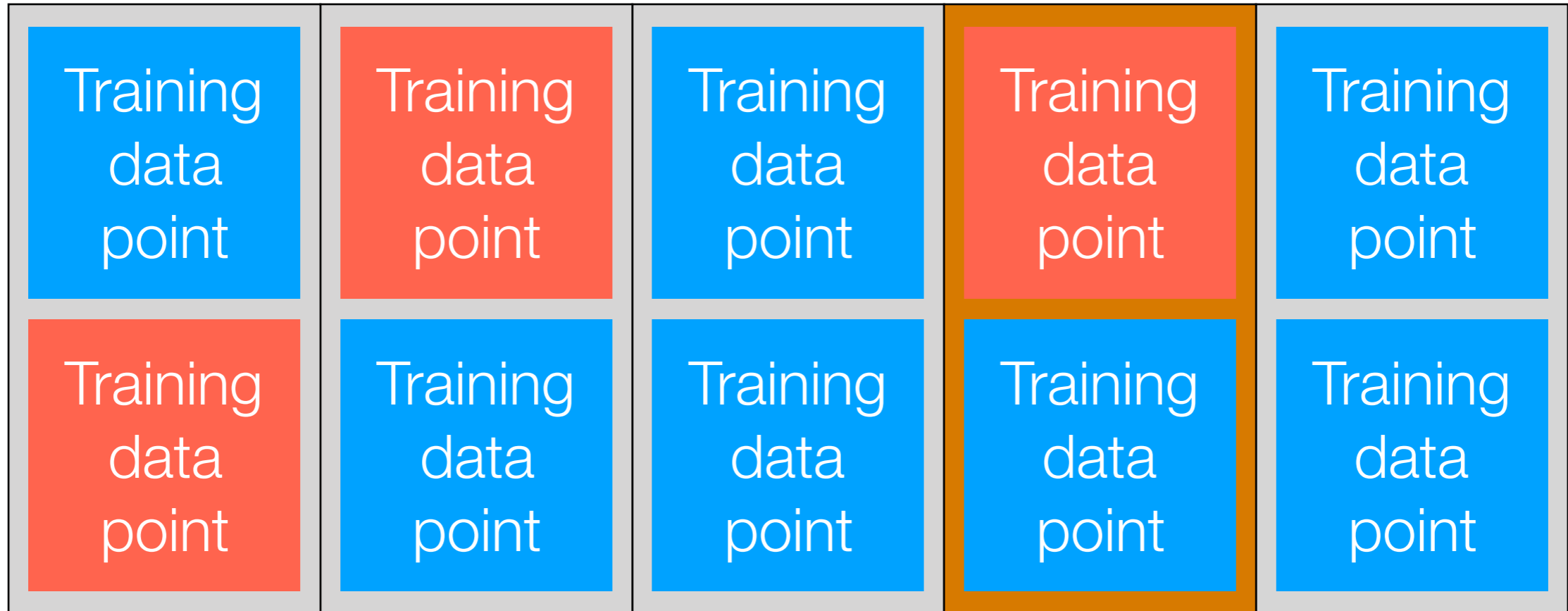


Train method on data in gray

Predict on data in orange

Compute prediction error

50%



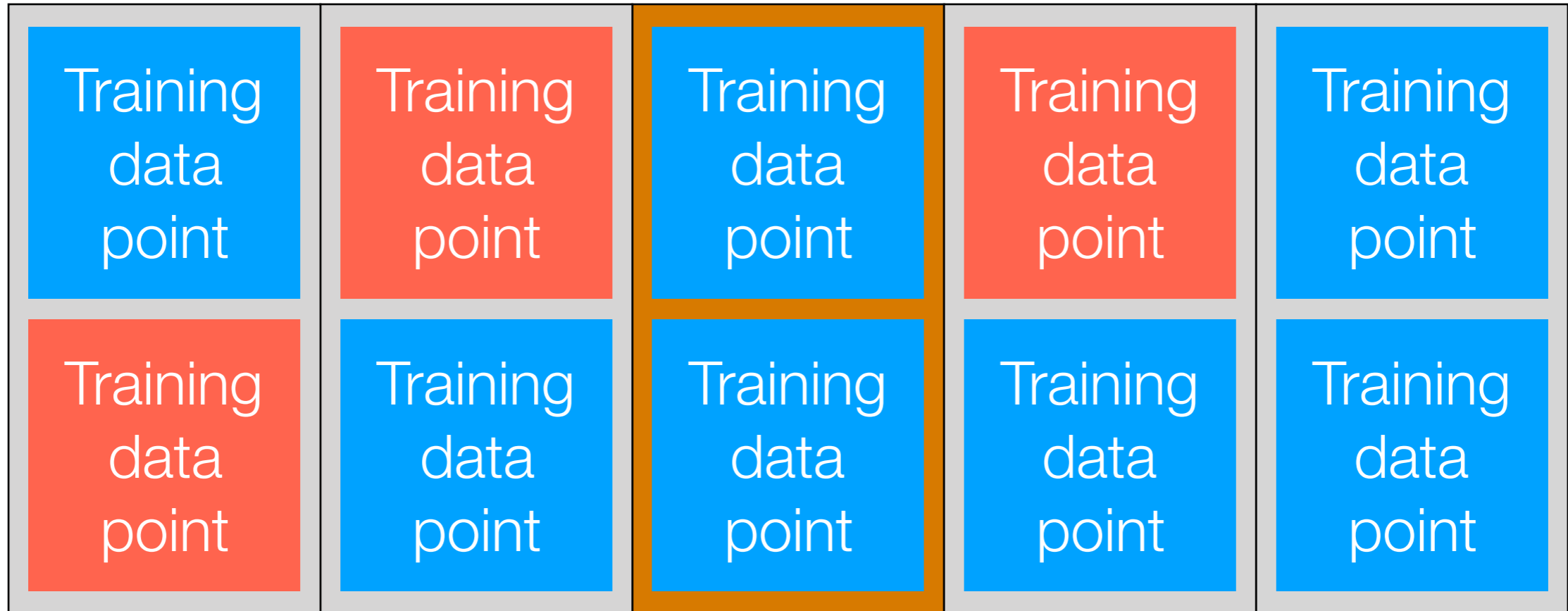
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%



Train method on data in gray

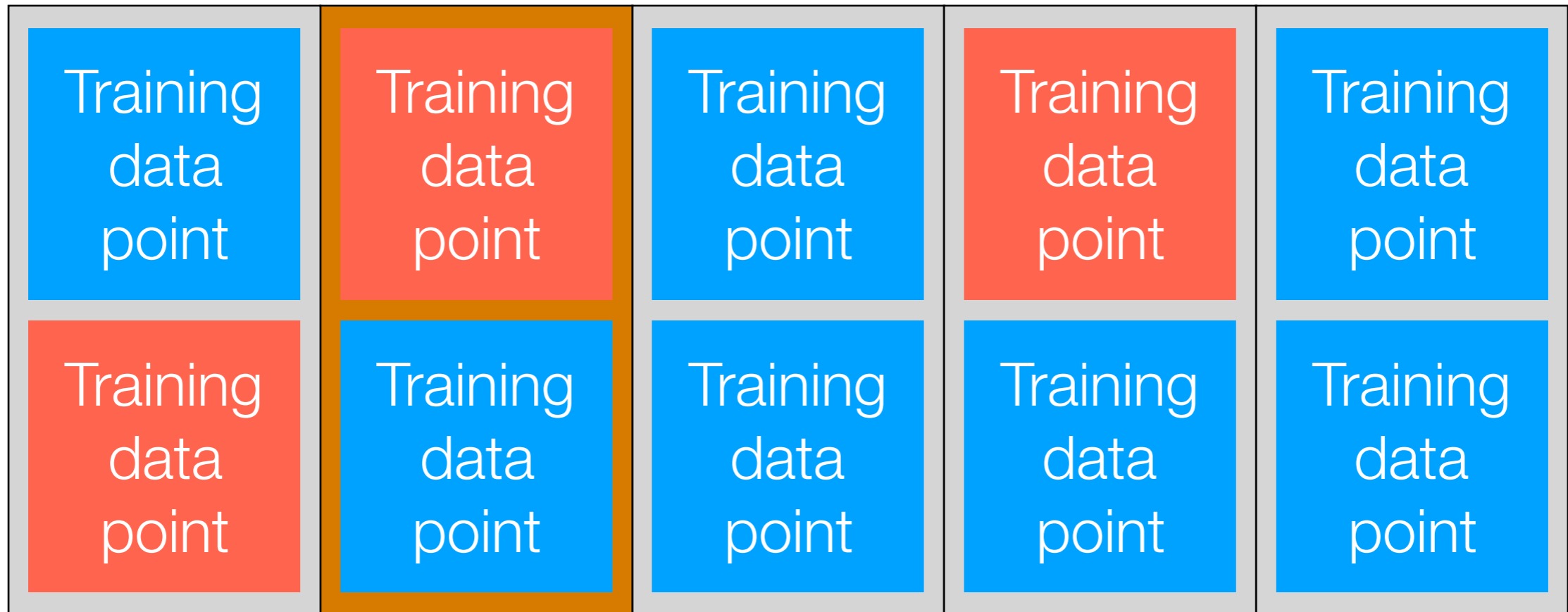
Predict on data in orange

Compute prediction error

50%

0%

50%



Train method on data in gray

Predict on data in orange

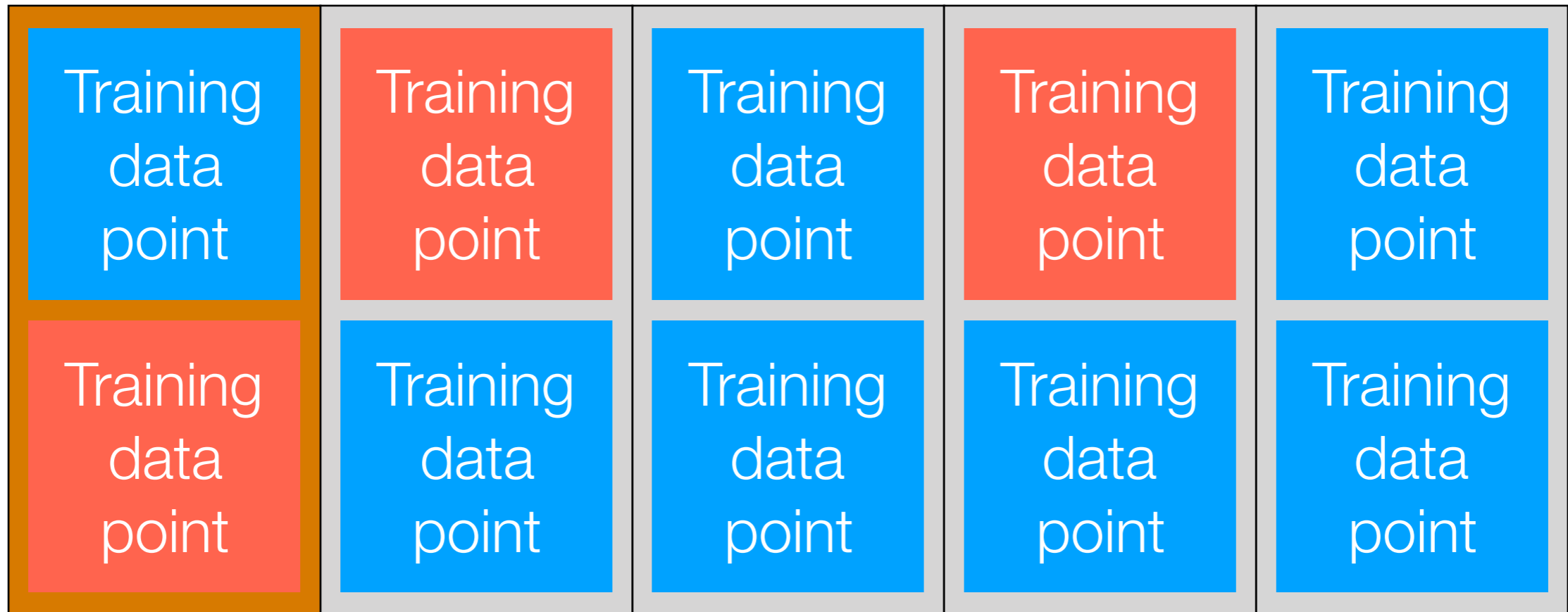
Compute prediction error

0%

50%

0%

50%



Train method on data in gray

Predict on data in orange

Compute prediction error

0%

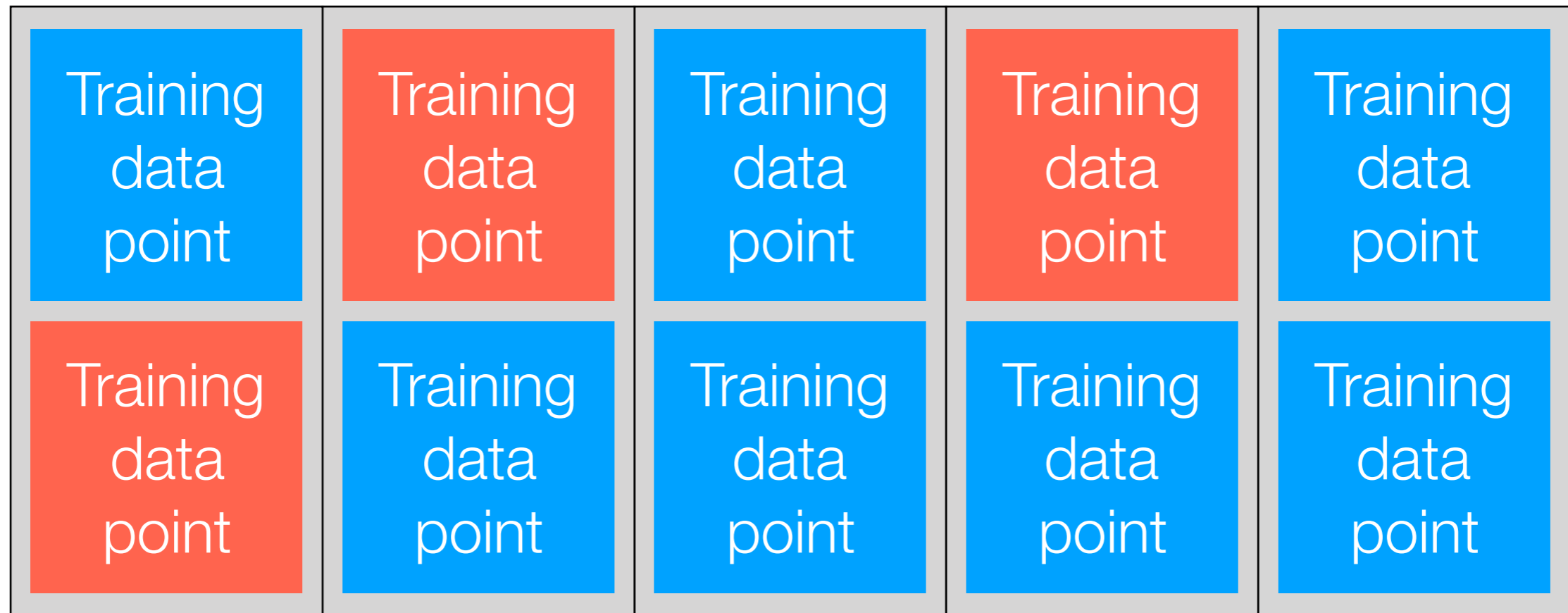
0%

50%

0%

50%

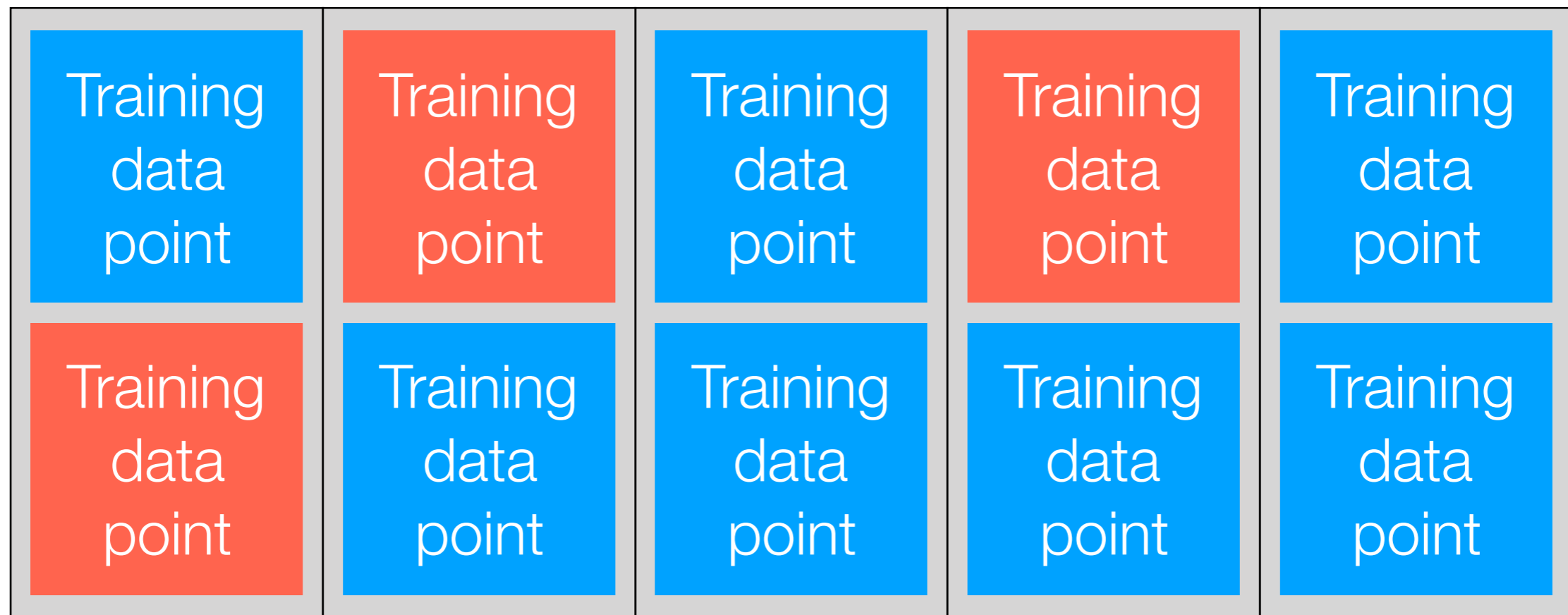
Average error: $(0+0+50+0+50)/5 = 20\%$



1. Shuffle data and put them into “folds” (5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
 - (a) Predict on fold’s training data, test on fold’s validation data
 - (b) Compute prediction error
3. Compute average prediction error across the folds

not the same k as in k -means

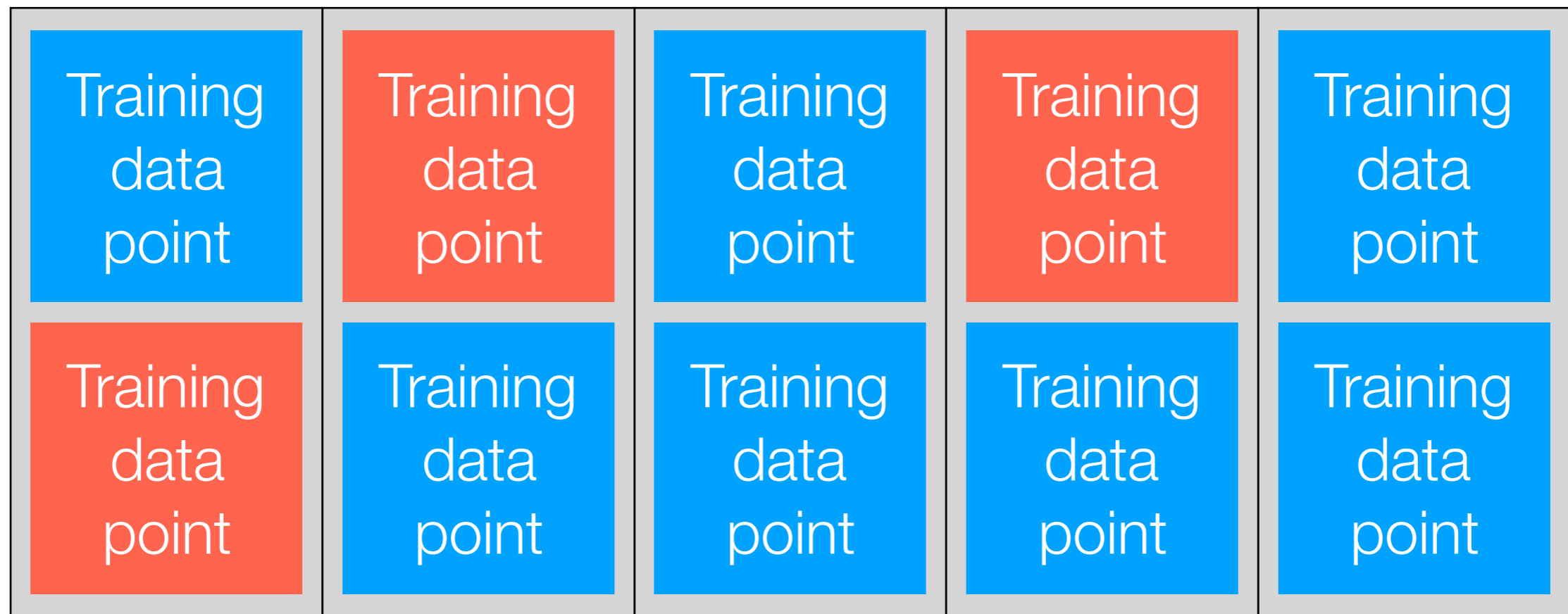
k -fold Cross Validation



1. Shuffle data and put them into “folds” ($k=5$ folds in this example)
2. For each fold (which consists of its own train/validation sets):
 - (a) Predict on fold’s training data, test on fold’s validation data
 - (b) Compute prediction error
3. Compute average prediction error across the folds

not the same k as in k -means

k -fold Cross Validation



1. Shuffle data and put them into “folds” ($k=5$ folds in this example)
2. For each fold (which consists of its own train/validation sets):
 - (a) Predict on fold’s training data, test on fold’s validation data
 - (b) Compute **some sort of prediction score**
3. Compute **average prediction score** across the folds
“cross validation score”

Automatic Hyperparameter Selection

Suppose the prediction algorithm you're using has hyperparameters θ (such as C for C -SVM)

For each hyperparameter setting θ you are willing to try:

Compute **5**-fold cross validation score using your algorithm with hyperparameters θ

Use whichever θ has the best cross validation score

Why 5?



People have found using 10 folds or 5 folds to work well in practice but it's just empirical — there's no deep reason

Prediction and Validation

Demo