Carnegie Mellon University
Heinz College

95-865 Pittsburgh Lecture 8: Topic Modeling, Introduction to Predictive Data Analytics

George Chen

Disclaimer: unfortunately "k" means many things

LDA

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: "document-word" matrix, and pre-specified # topics *k*

i-th row, *j*-th column: # times word *j* appears in doc *i*

- Output: the *k* topics' distribution of words

LDA

Demo

How to Choose Number of Topics *k*?

Something like CH index is also possible:

avoid numerical issues

For a specific topic, look at the *m* most probable words ("top words")

Coherence (within cluster/topic variability):

$$\sum_{\substack{\text{top words } v,w \\ \text{that are not the same}}} \log \frac{\text{\# documents that contain both } v \text{ and } w + 0.1}{\text{\# documents that contain } w}$$

log of P(see word v | see word w)

Inter-topic similarity (between cluster/topic variability):

Can average each of these across the topics

Count # top words that do not appear in any of the other topics' *m* top words

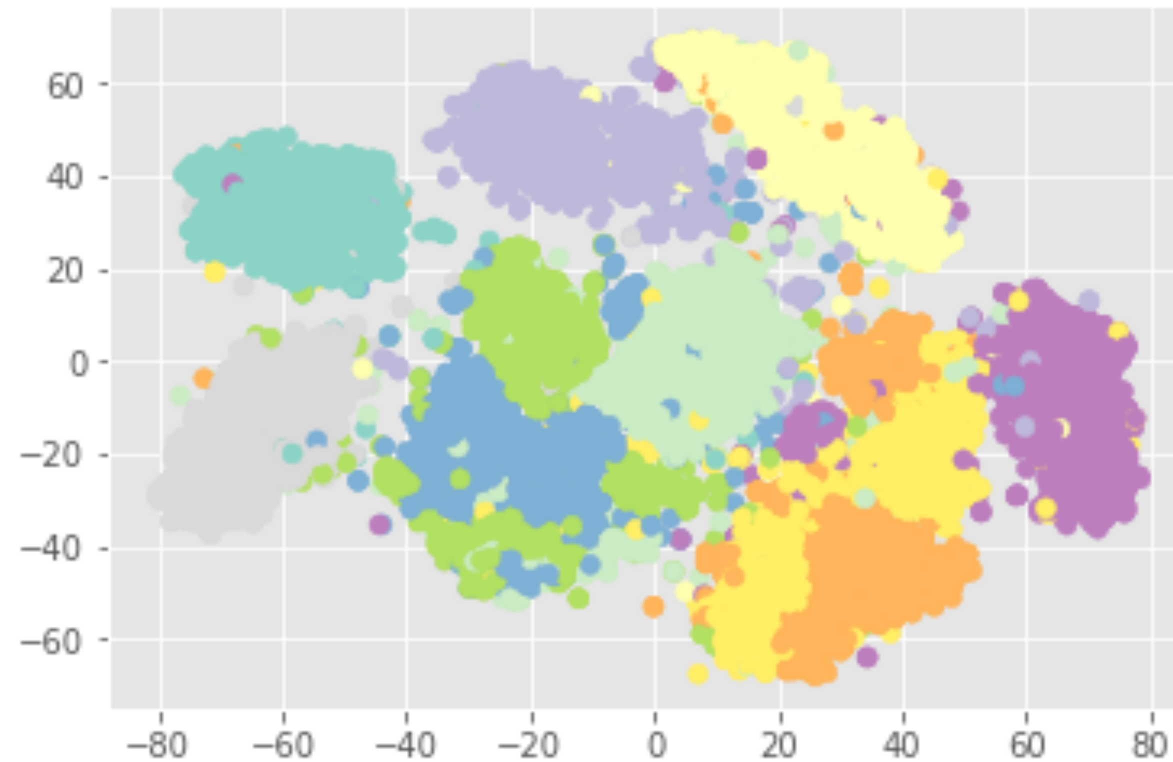
(number of "unique words")

Topic Modeling: Last Remarks

- There are actually *many* topic models, not just LDA
  - Correlated topic models, Pachinko allocation, biterm topic models, anchor word topic models, …

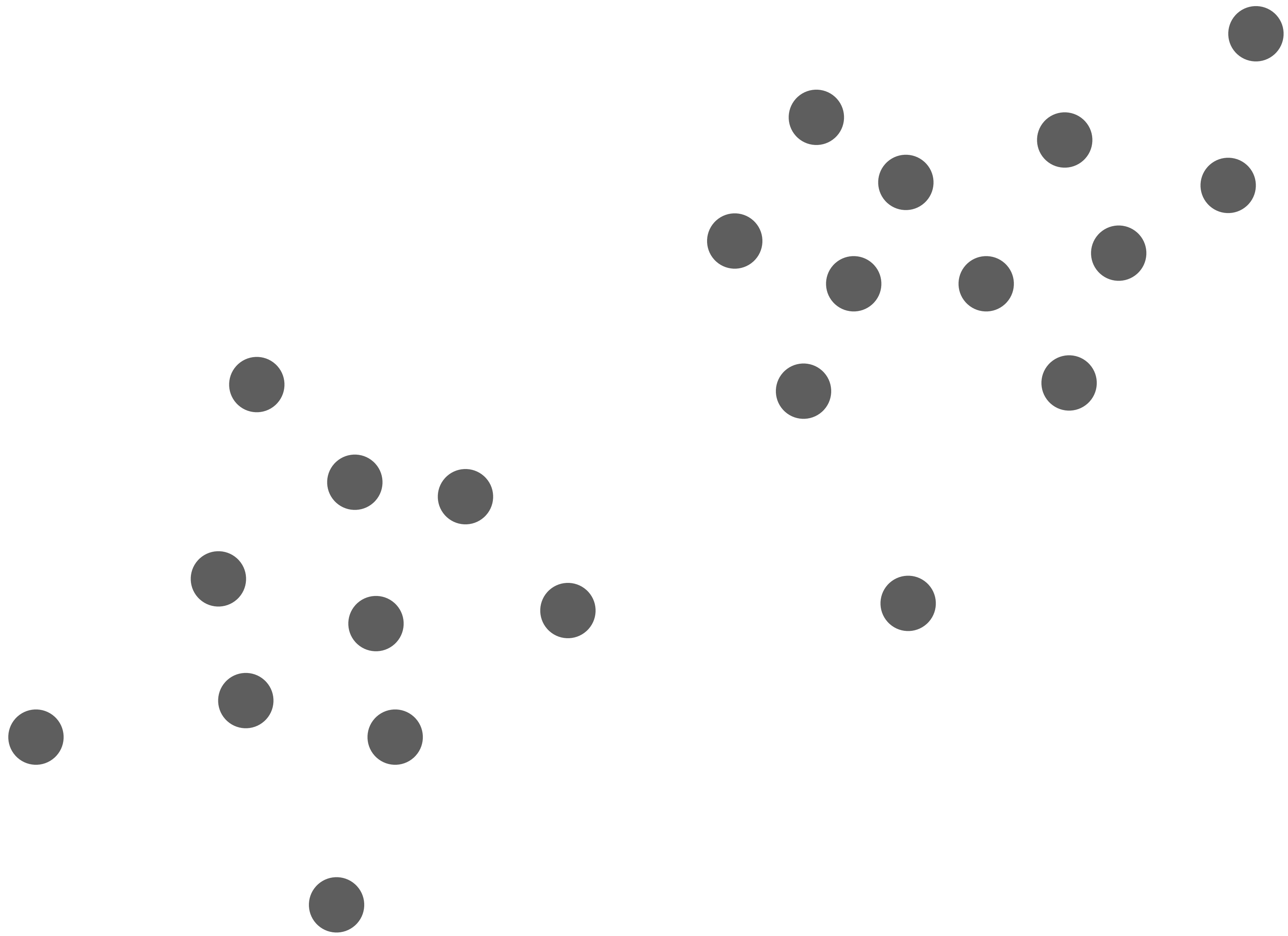
- Dynamic topic models: tracks how topics change *over time*
  - Example: for text over time, figure out how topics change
  - Example: for recommendation system, figure out how user tastes change over time

What if we have labels?



Example: MNIST handwritten digits have known labels

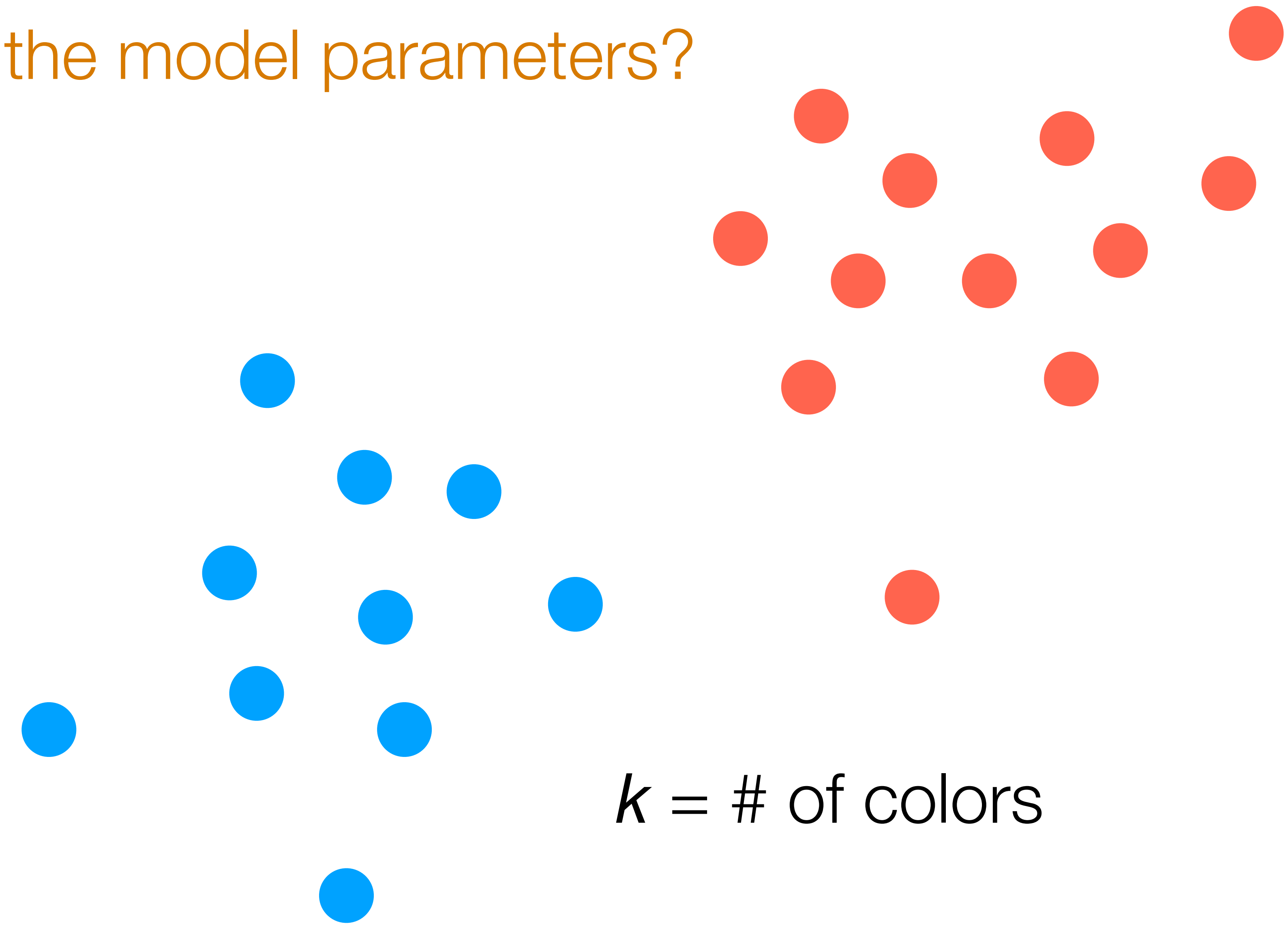
If the labels are known…



If the labels are known…

And we assume data generated by GMM…

What are the model parameters?



k = # of colors

We can directly estimate
cluster means, covariances

Flashback: Learning a GMM

Don’t need this top part if we know the labels!

Step 0: Pick *k*

Step 1: Pick <u>guesses</u> for **cluster means and covariances**

Repeat until convergence:

Step 2: Compute probability of each point belonging to each of the *k* clusters

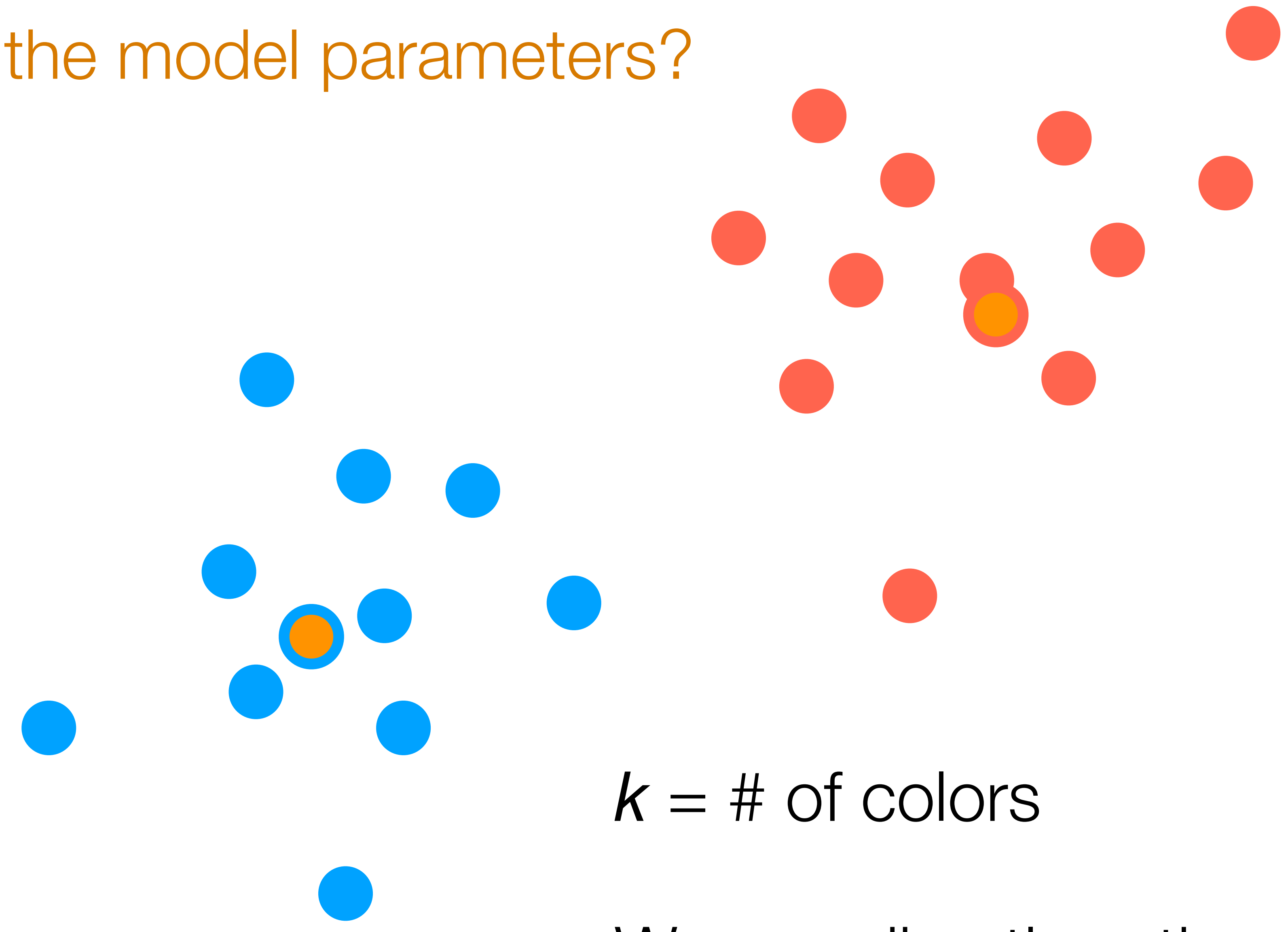
Step 3: Update **cluster means and covariances** carefully accounting for probabilities of each point belonging to each of the clusters

We don’t need to repeat until convergence

If the labels are known…

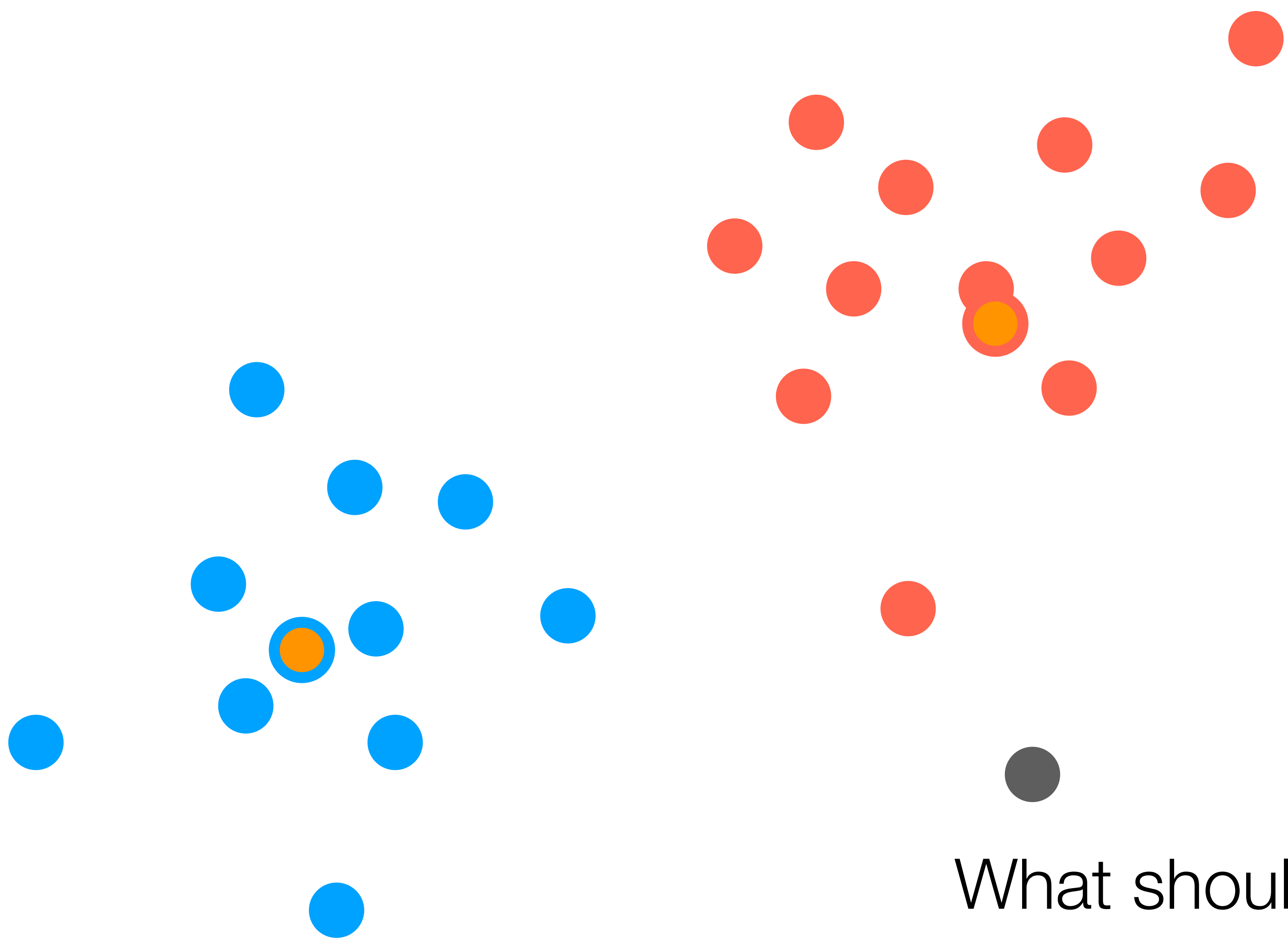
And we assume data generated by GMM…

What are the model parameters?



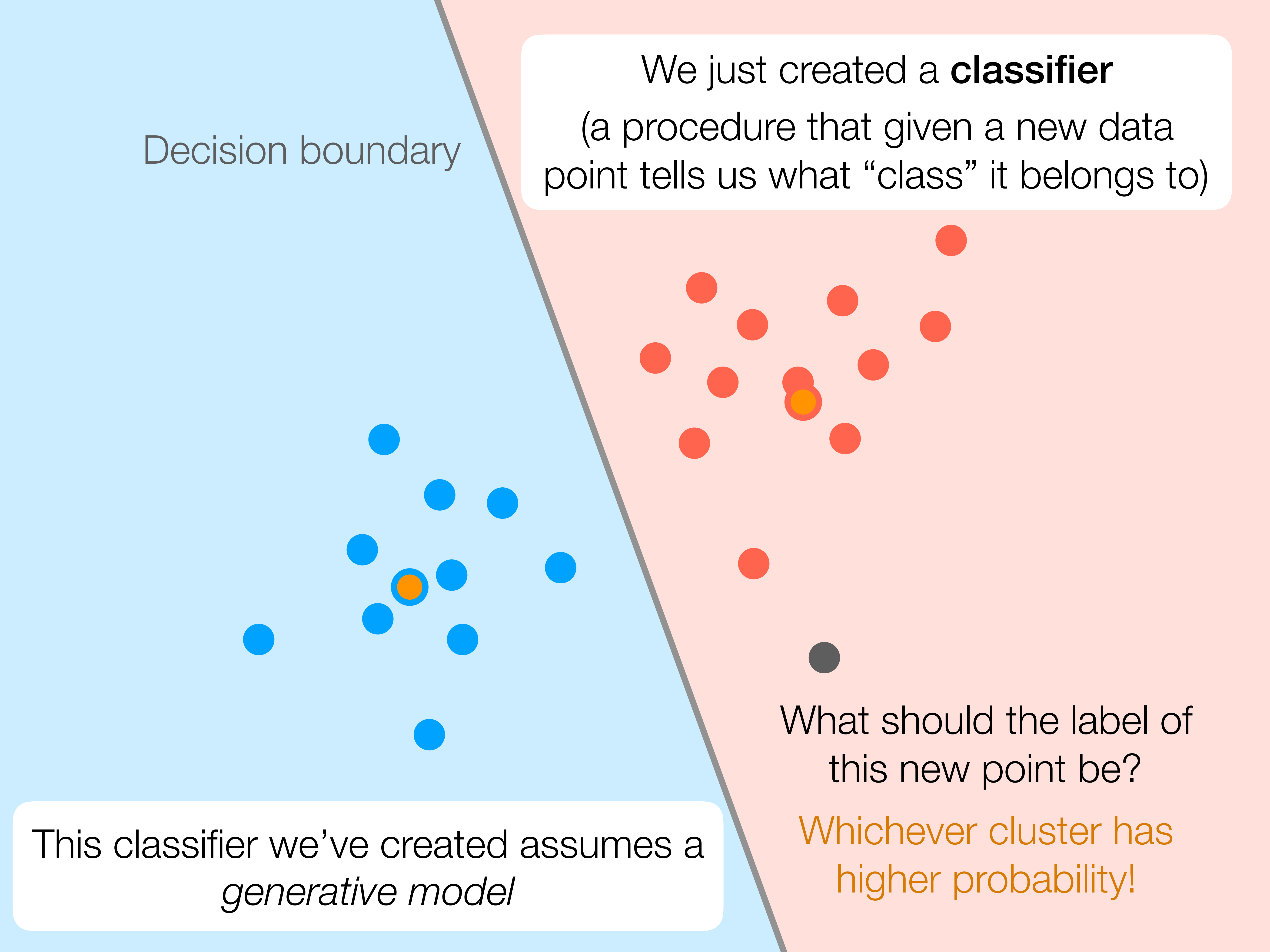
k = # of colors

We can directly estimate cluster means, covariances



What should the label of
this new point be?

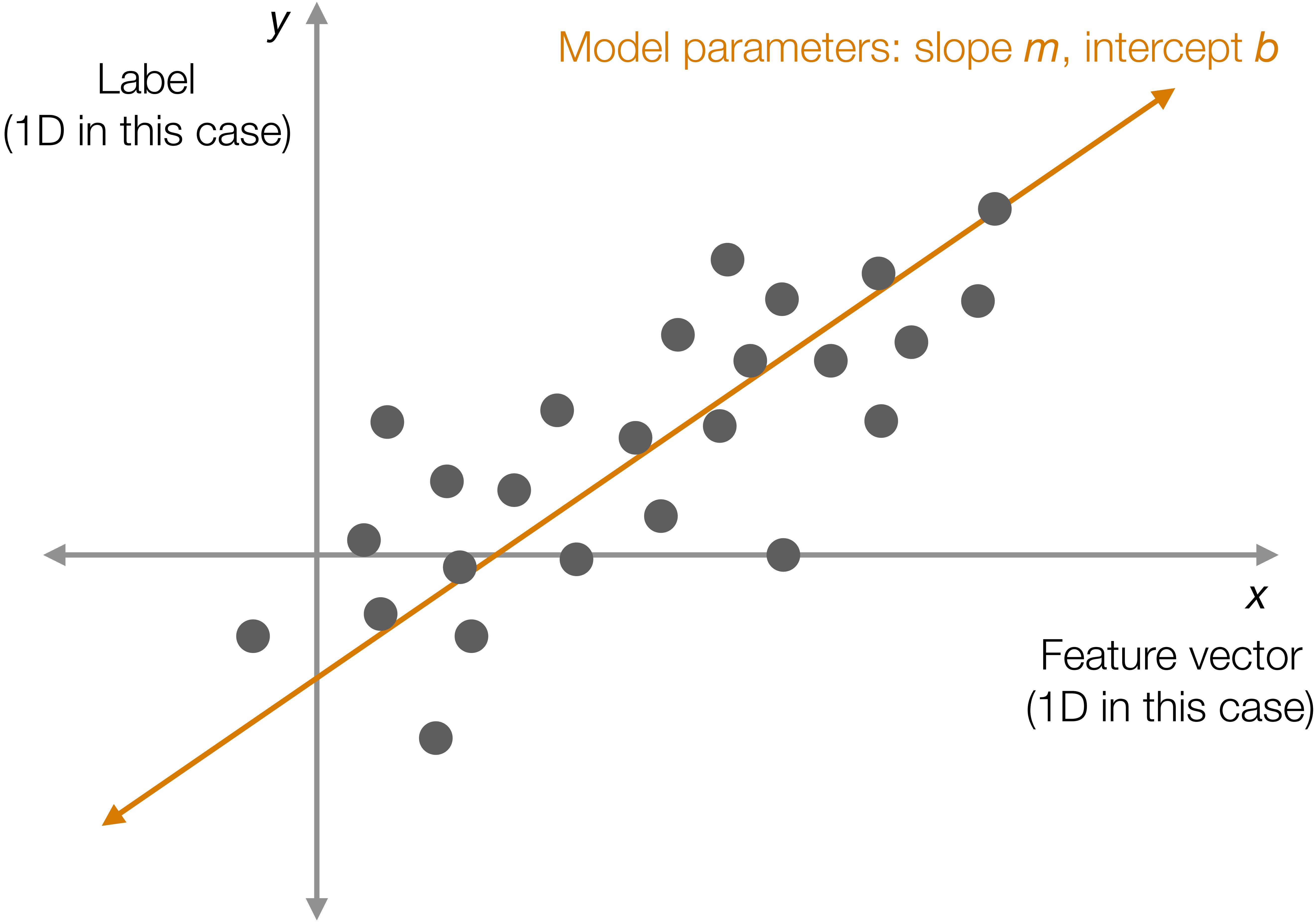
Whichever cluster has
higher probability!



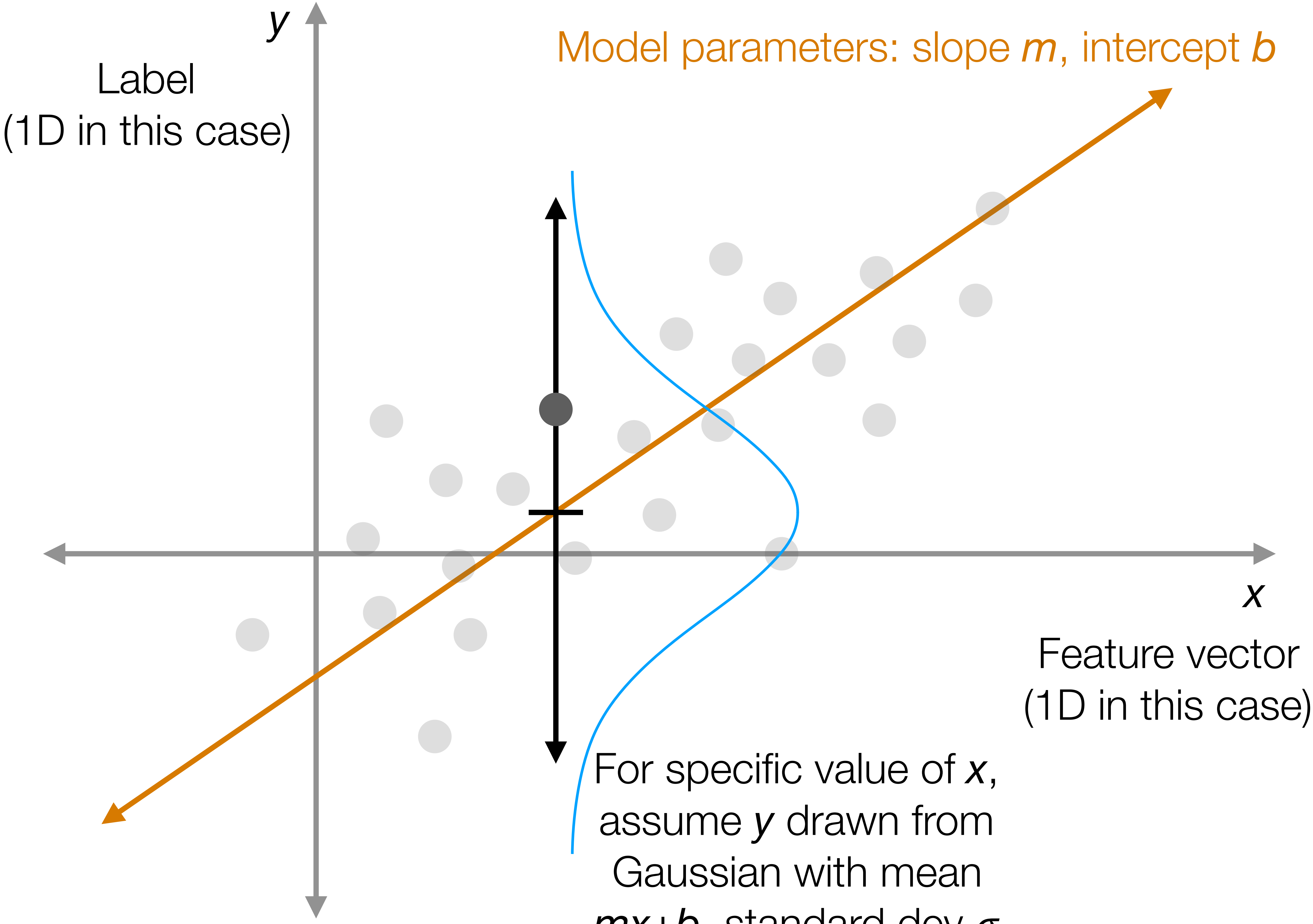
We just created a **classifier**
(a procedure that given a new data point tells us what "class" it belongs to)
Decision boundary
What should the label of this new point be?
Whichever cluster has higher probability!
This classifier we've created assumes a *generative model*

You’ve seen generative models before for prediction

Linear regression!



y
Label
(1D in this case)
Model parameters: slope m, intercept b
x
Feature vector
(1D in this case)



y
Label
(1D in this case)
Model parameters: slope m, intercept b
x
Feature vector
(1D in this case)
For specific value of x,
assume y drawn from
Gaussian with mean
mx+b, standard dev σ

Predictive Data Analysis

Training data

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

Goal: Given new feature vector x, predict label y

- y is discrete (such as colors red and blue)
  ➔ prediction method is called a **classifier**

- y is continuous (such as a real number)
  ➔ prediction method is called a **regressor**

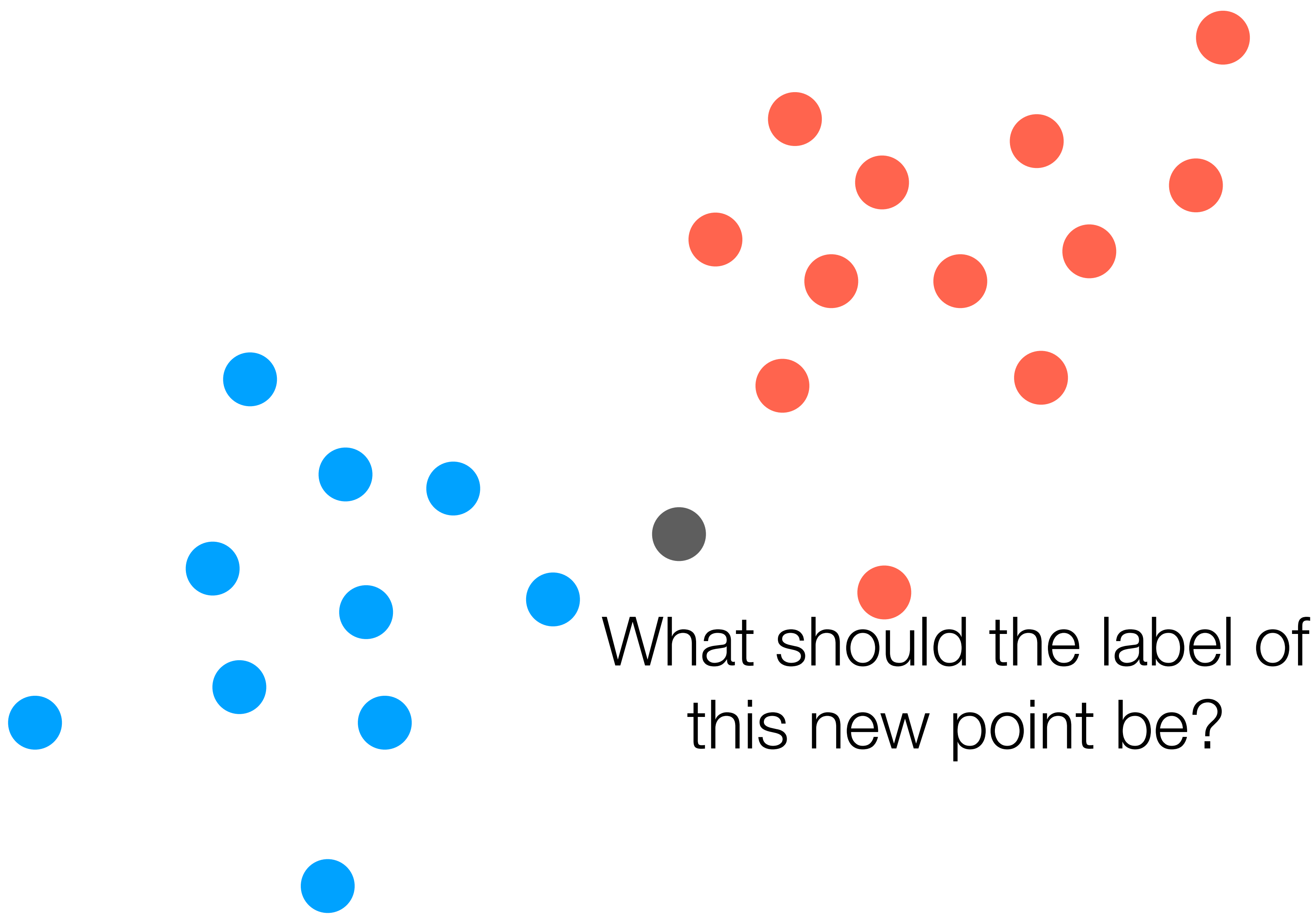
A giant zoo of methods

Generative Models

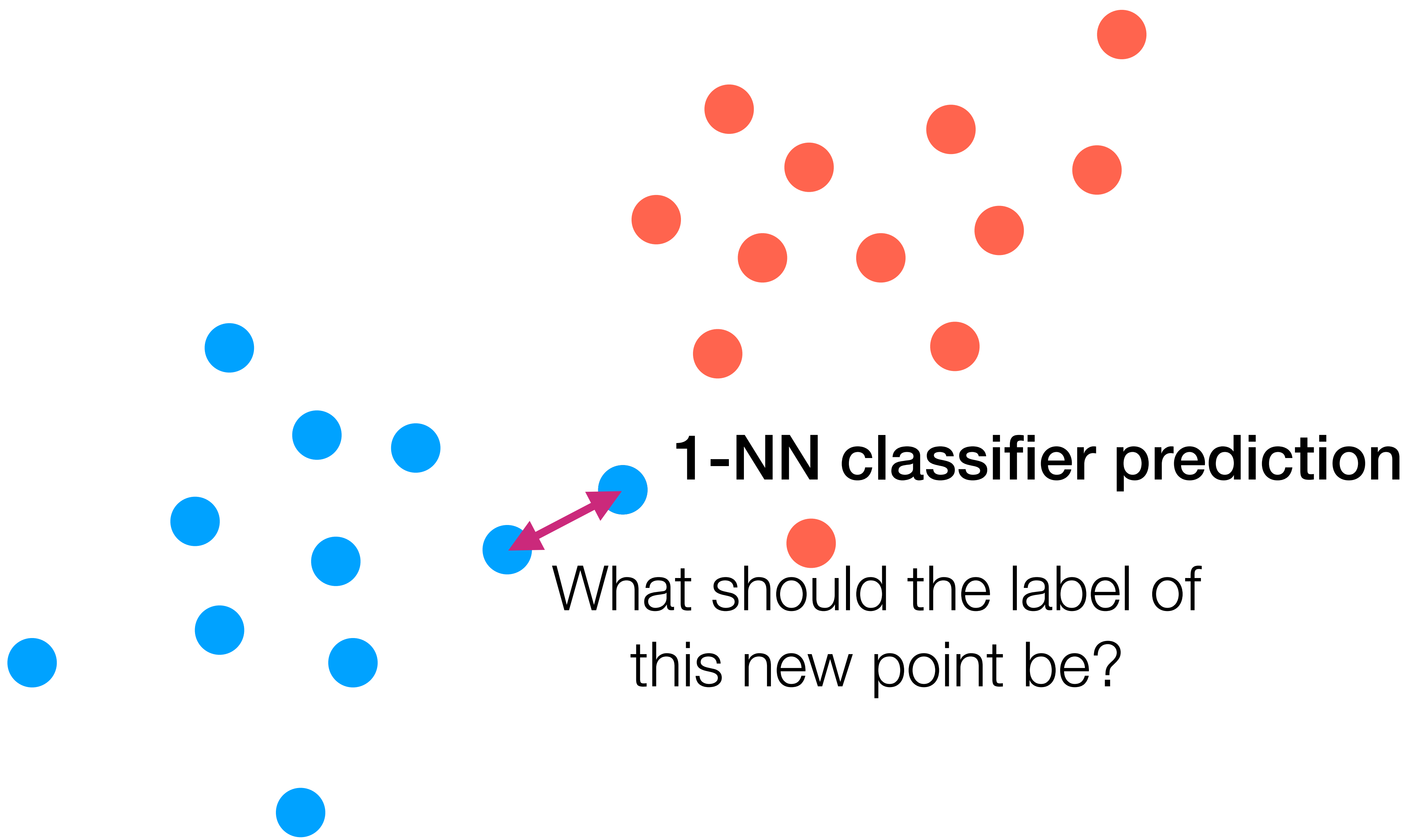
- Hypothesize a specific way in which data are generated
- After learning a generative model:
    - We can generate new synthetic data from the model
    - Usually generative models are probabilistic and we can evaluate probabilities for a new data point
- In contrast to generative models, there are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

Example of a Discriminative Method: *k*-NN Classification

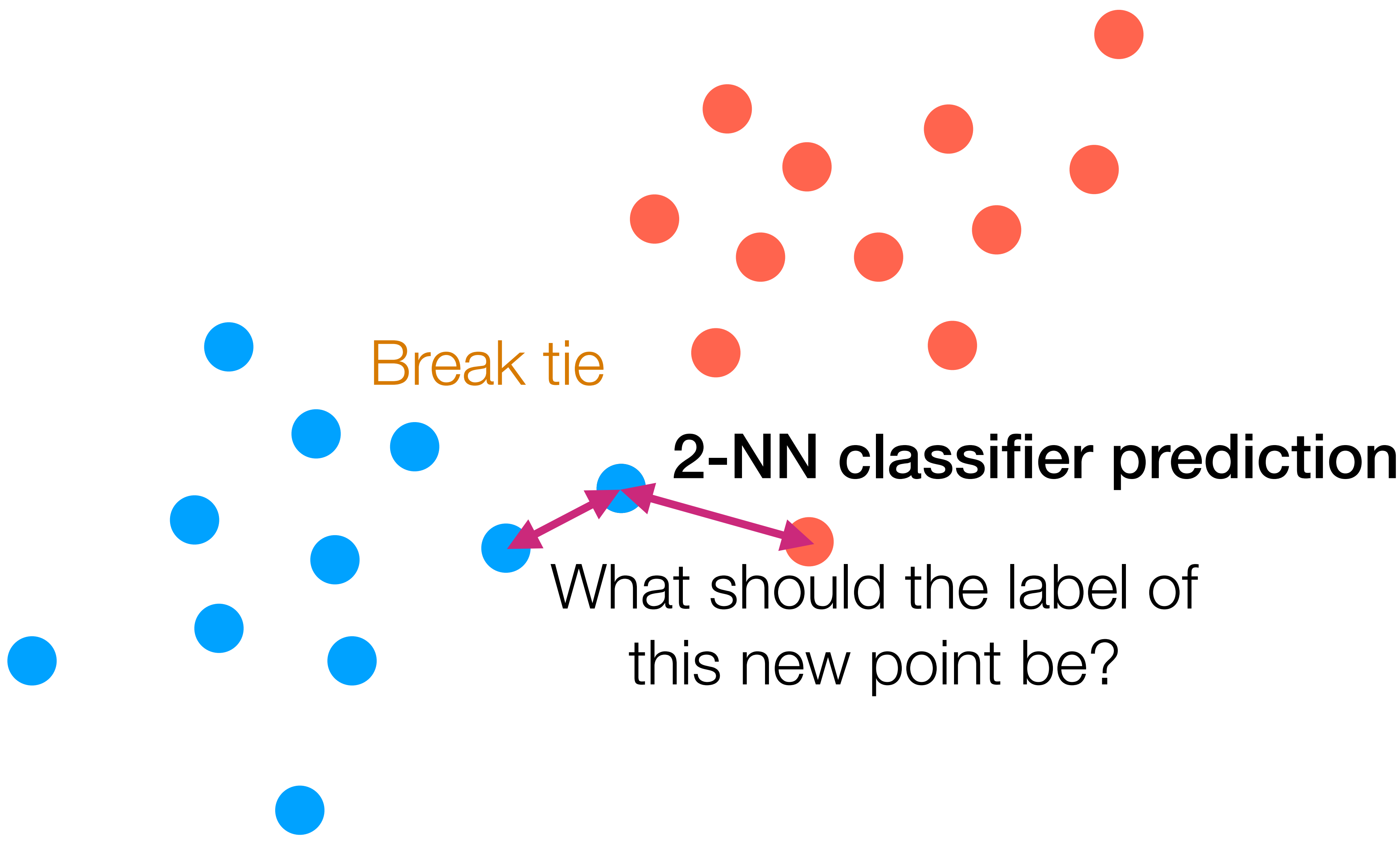
Example: *k*-NN Classification



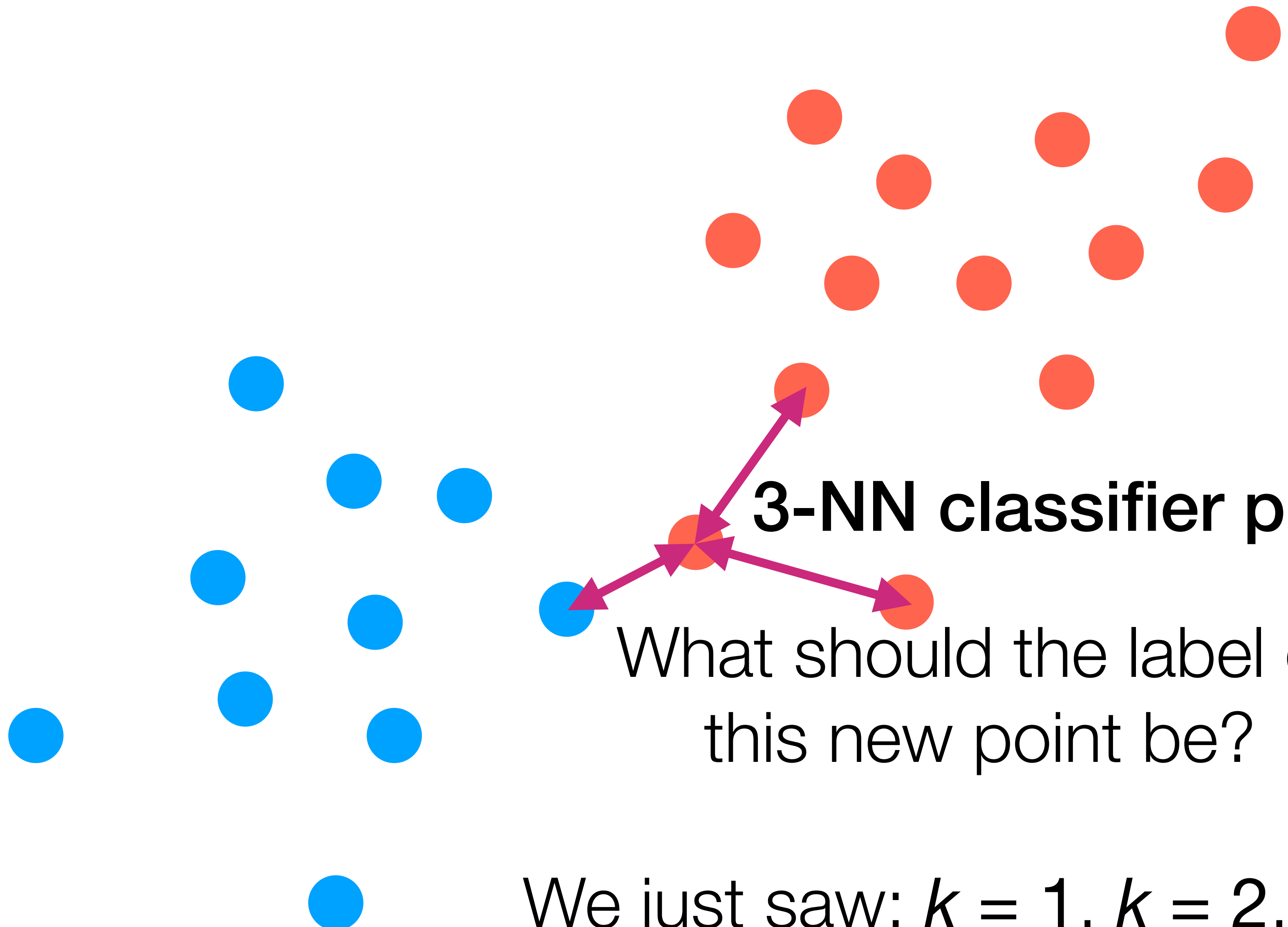
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification



We just saw: k = 1, k = 2, k = 3

What happens if $k = n$?

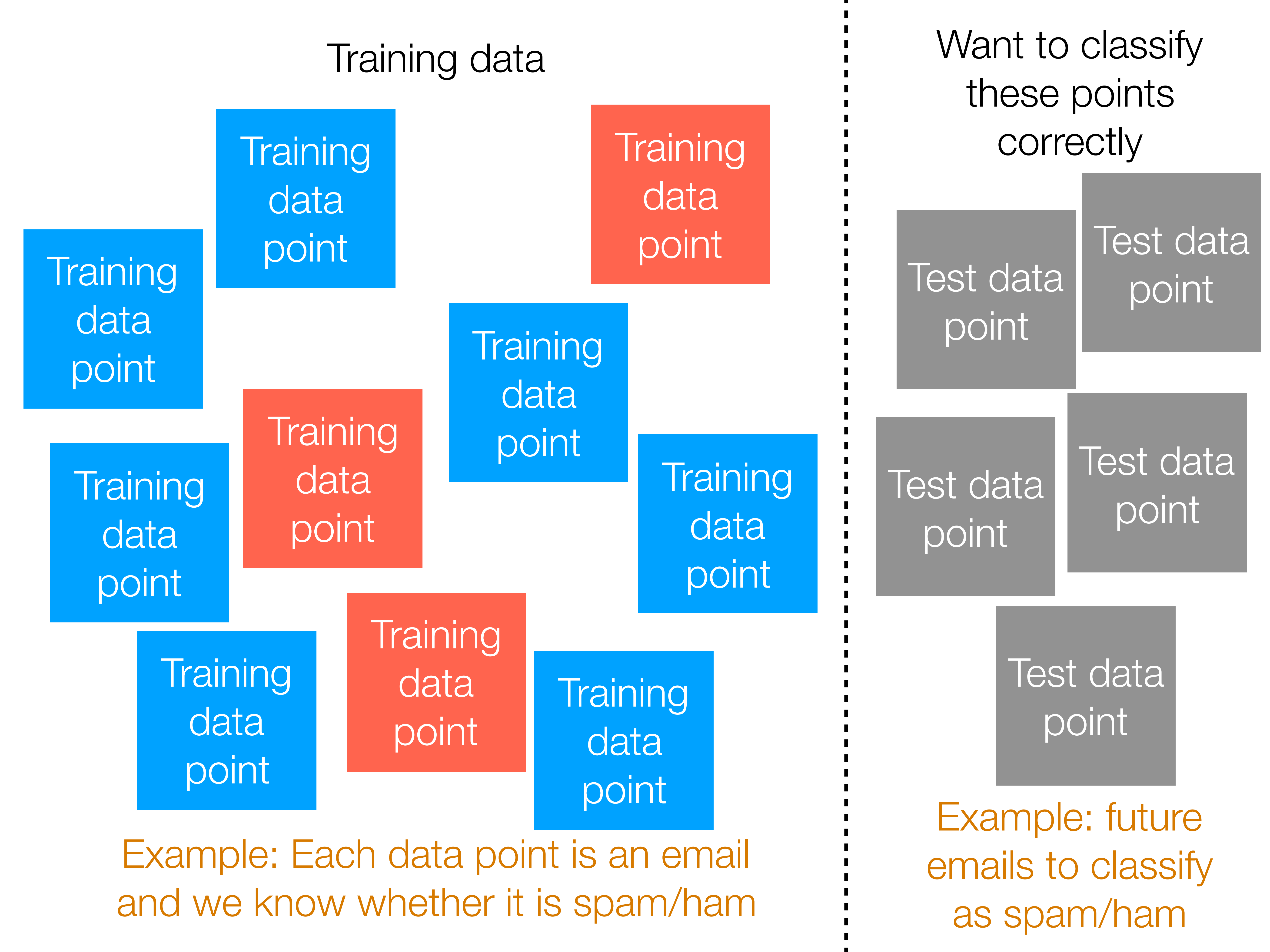
How do we choose k?

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

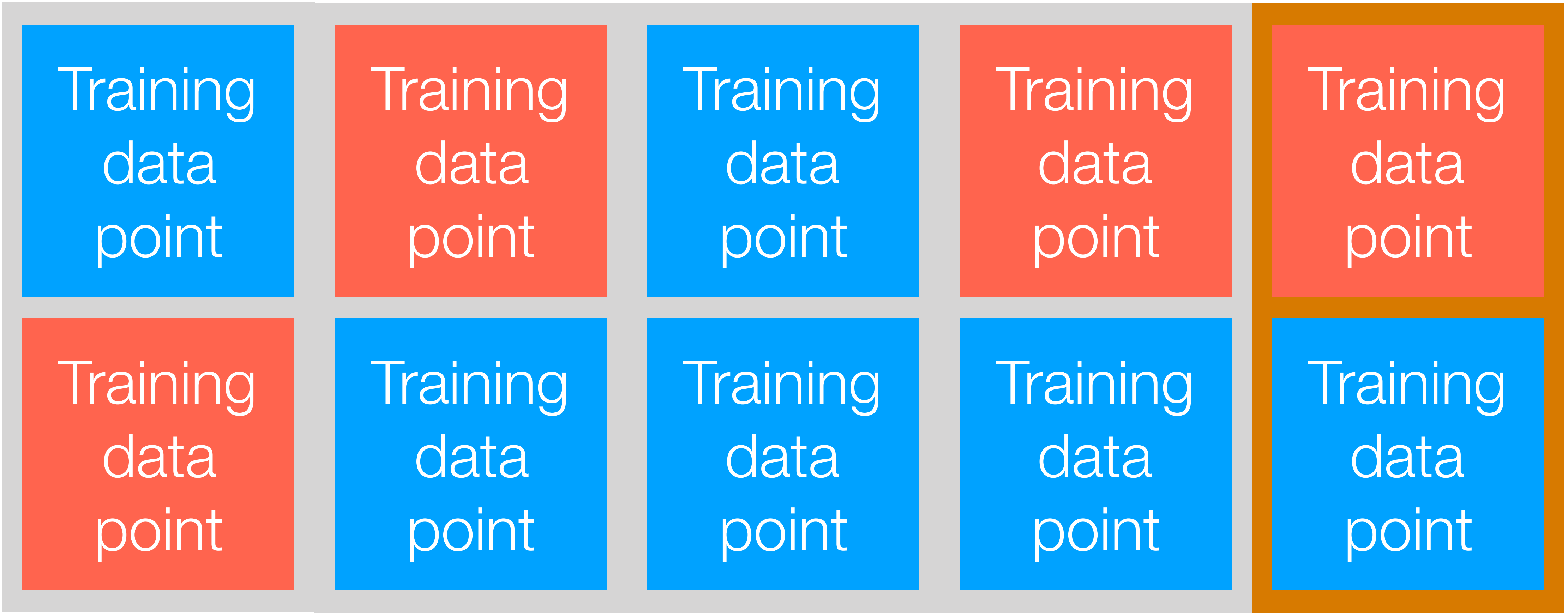
First: How do we assess how good a prediction method is?

Hyperparameters vs. Parameters

- We fit a model's parameters to training data (terminology: we "learn" the parameters)
- We pick 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: k-NN classification
  - Hyperparameter: number of nearest neighbors k
  - Parameters: N/A


Training data
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Example: Each data point is an email and we know whether it is spam/ham
Want to classify these points correctly
Test data point
Test data point
Test data point
Test data point
Test data point
Example: future emails to classify as spam/ham

Predicted labels

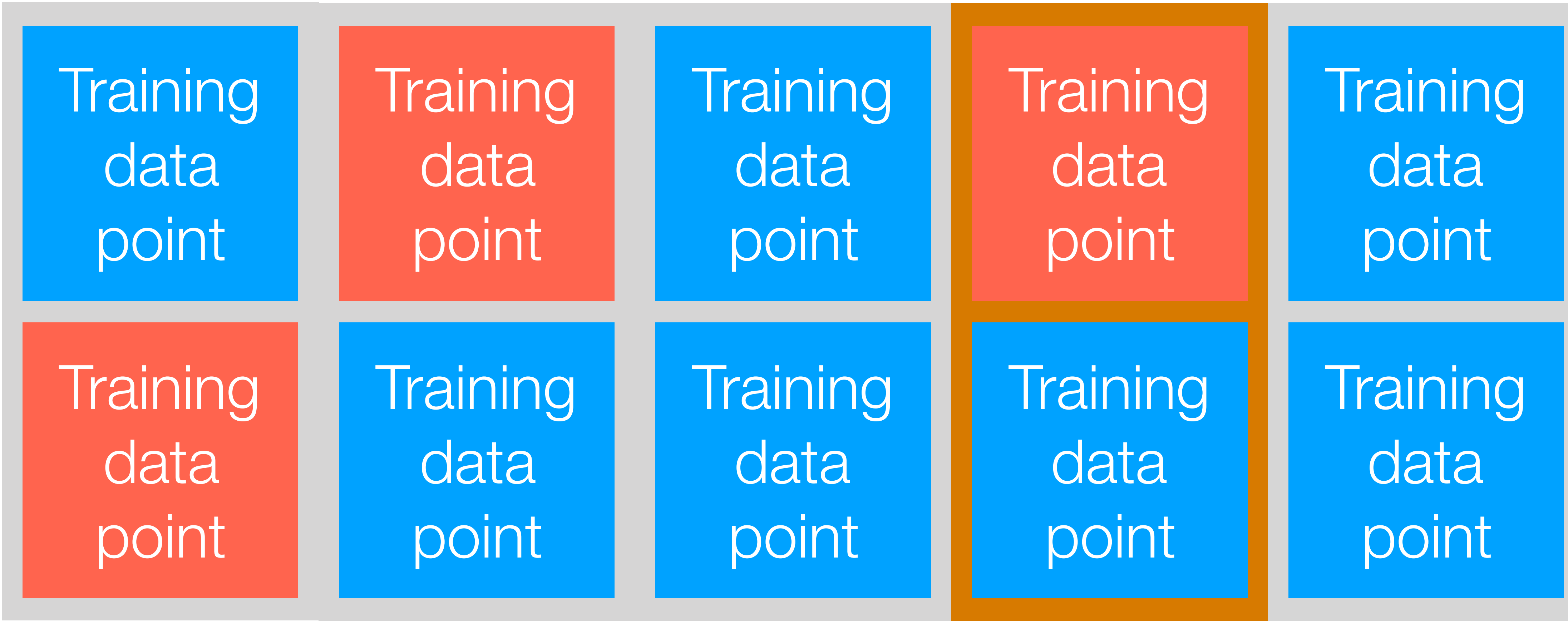


Train method on data in gray

Predict on data in orange

Compute prediction error

50%



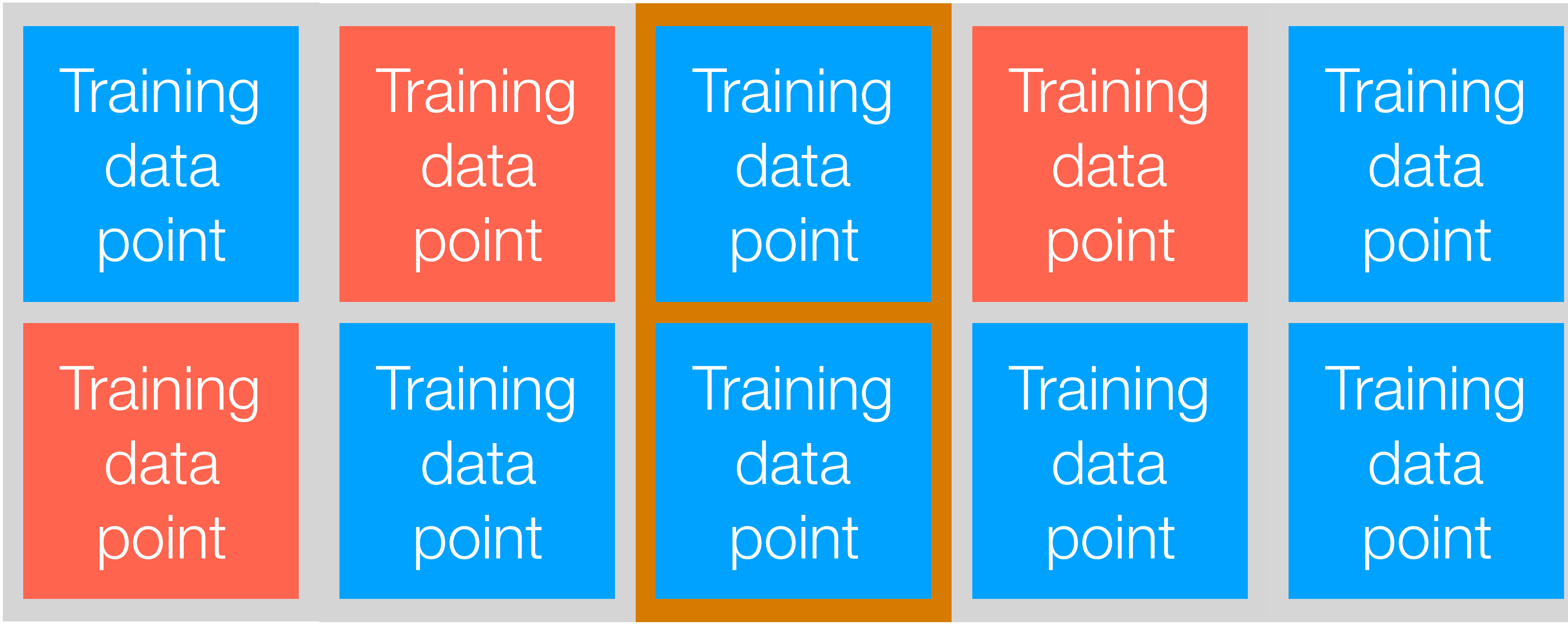
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%

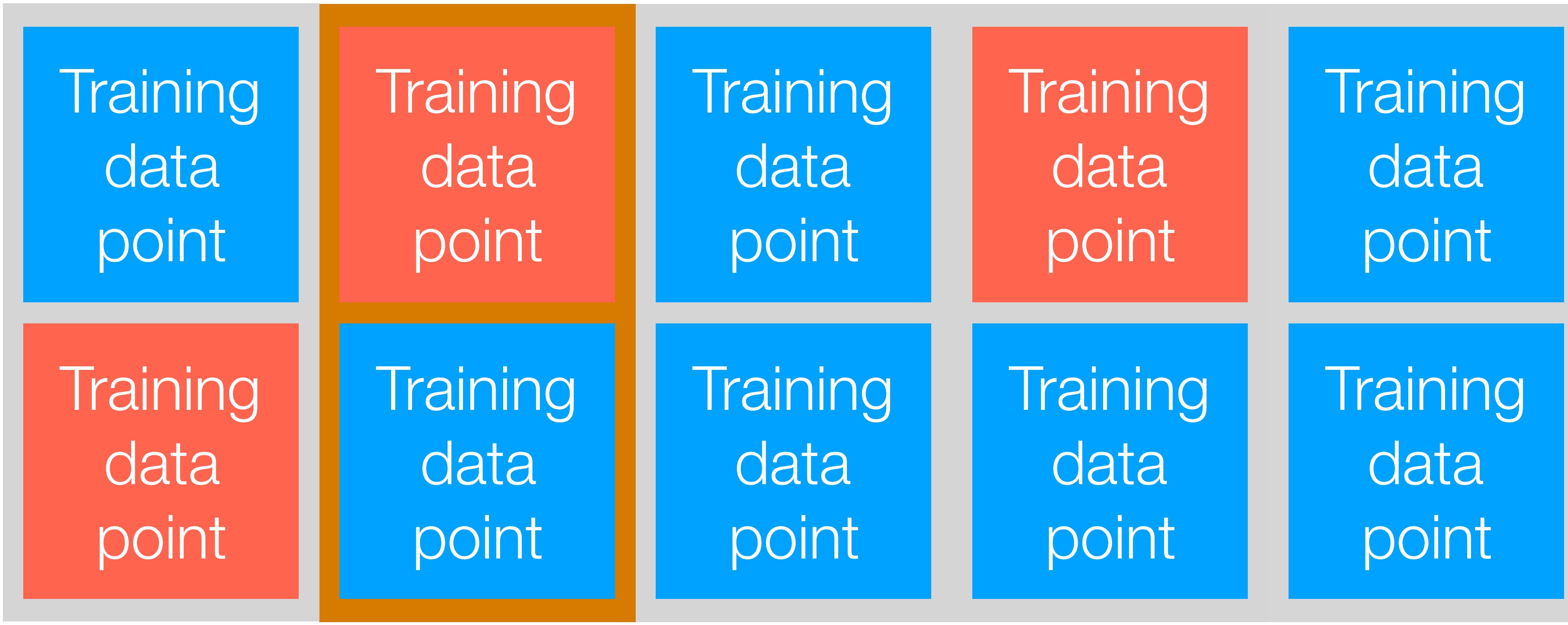


Train method on data in gray

Predict on data in orange

Compute prediction error

50% 0% 50%

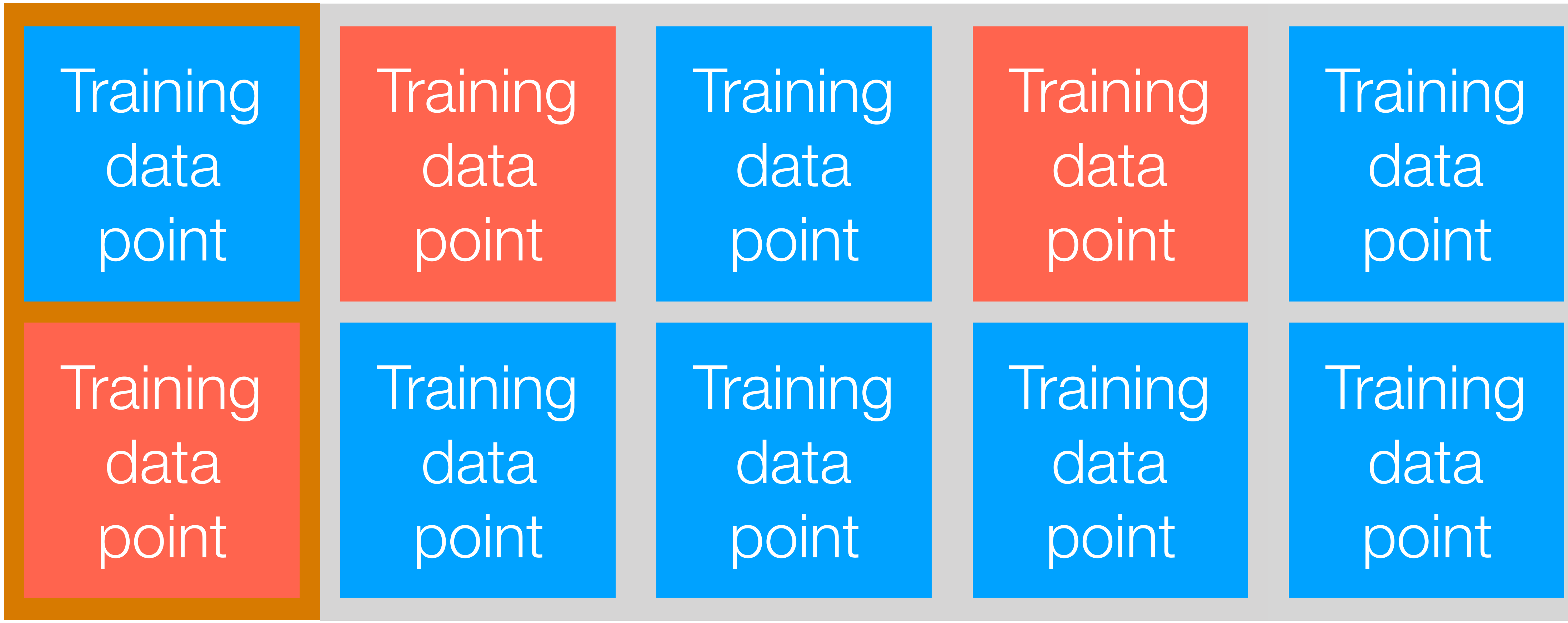


Train method on data in gray

Predict on data in orange

Compute prediction error

0% 50% 0% 50%



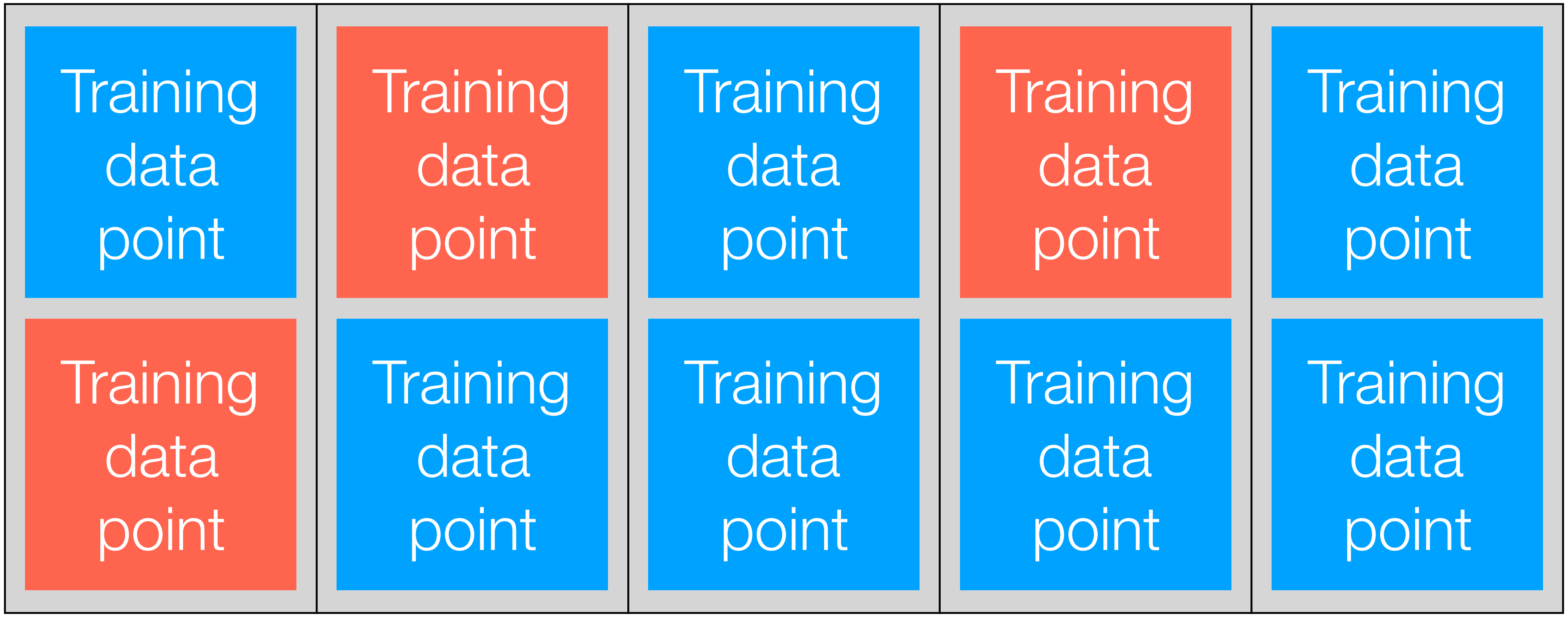
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 0% 50% 0% 50%

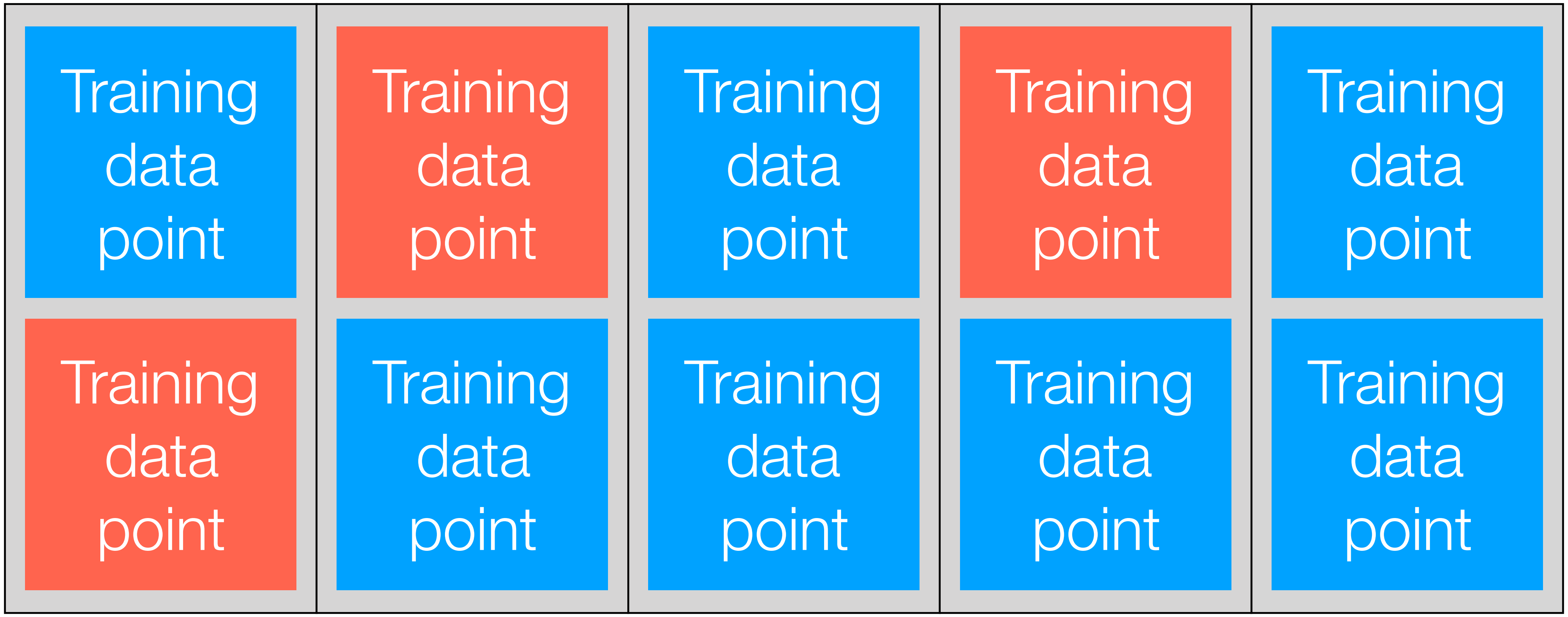
Average error: (0+0+50+0+50)/5 = 20%



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) Train 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 or *k*-NN classification

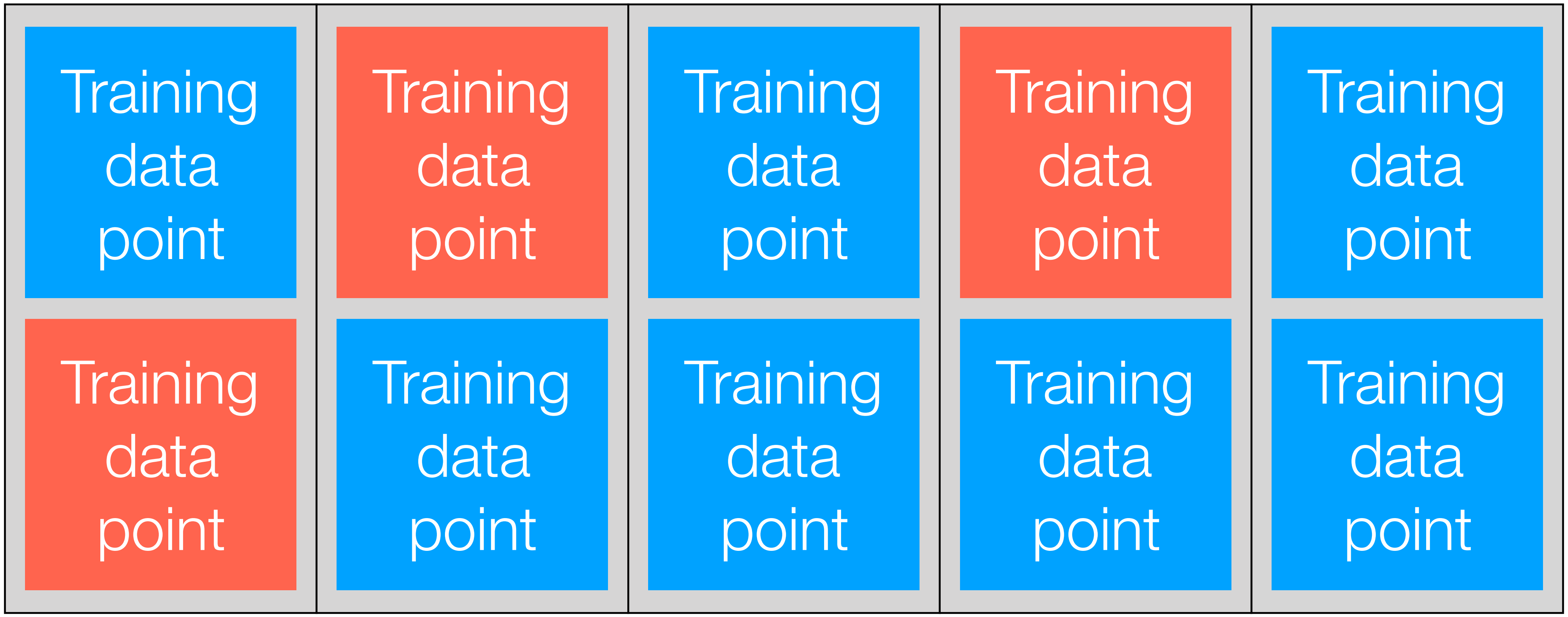
k-fold Cross Validation



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) Train 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 or *k*-NN classification

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) Train 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”

Choosing k in k-NN Classification

Note: k-NN classifier has a single hyperparameter k

For each k = 1, 2, 3, …, the maximum k you are willing to try:

    Compute 5-fold cross validation score using k-NN classifier as prediction method

Use whichever k has the best cross validation score

Automatic Hyperparameter Selection

Suppose the prediction algorithm you're using has hyperparameters θ

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