Recall: the code demo used *structured* data to hopefully make the material coverage clearer

Thoughts on Interpreting Clustering Results for *Unstructured* Data

Key idea: first run analyses on each cluster separately

- Clustering on text documents:
  - Per cluster: for text documents in the cluster, conduct frequency & co-occurrence analysis (e.g., find most frequent words, pairs of words with highest PMI)
  - If using a k-means/GMM/DP-means/DP-GMM, can look at what text documents are closest to the cluster center

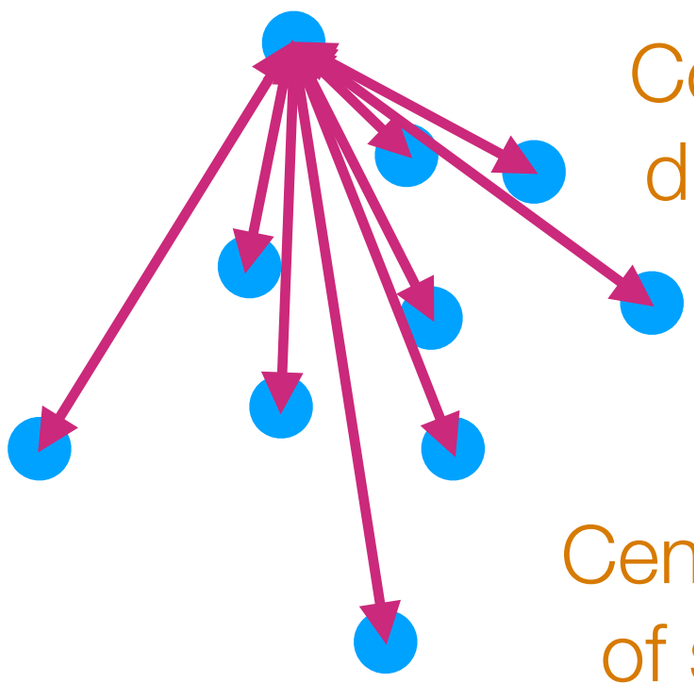
Compare findings from individual clusters to try to understand in what ways the clusters are different

Also works for images instead of text

What About When Using Distances/Similarities Where "Cluster Mean" is Not Obvious?

For example, consider if we use DBSCAN and use a distance that is *not* Euclidean

- Per cluster:
- Find "centroid"



Good choice for being centroid?

Compute sum of squared distances to other points

Distance need not be Euclidean!

Centroid: data point with smallest sum of squared distances to other points

What About When Using Distances/Similarities Where "Cluster Mean" is Not Obvious?

For example, consider if we use DBSCAN and use a distance that is *not* Euclidean

- Per cluster:
  - Find "centroid" (the version I just gave is technically called the Fréchet mean)
  - Can then see what points within the cluster are closest to the centroid (and also ones that are far away)

This strategy works for structured and unstructured data represented as feature vectors

Clustering Last Remarks

Ultimately, *you* have to decide on which clustering method and number of clusters make sense for your data

- After you run a clustering algorithm, make visualizations to interpret the clusters *in the context of your application*!
- Do not just blindly rely on numerical metrics (e.g., CH index)
- Some times it makes more sense to define your own score function for how good a clustering assignment is

If you can set up a prediction task, then you can use the prediction task to guide the clustering

Carnegie Mellon University
Heinz College

Unstructured Data Analysis

Lecture 10: Introduction to predictive data analytics

George Chen

95-865

Part I: Exploratory data analysis

Identify structure present in "unstructured" data

- Frequency and co-occurrence analysis
- Visualizing high-dimensional data/dimensionality reduction
- Clustering
- Topic modeling

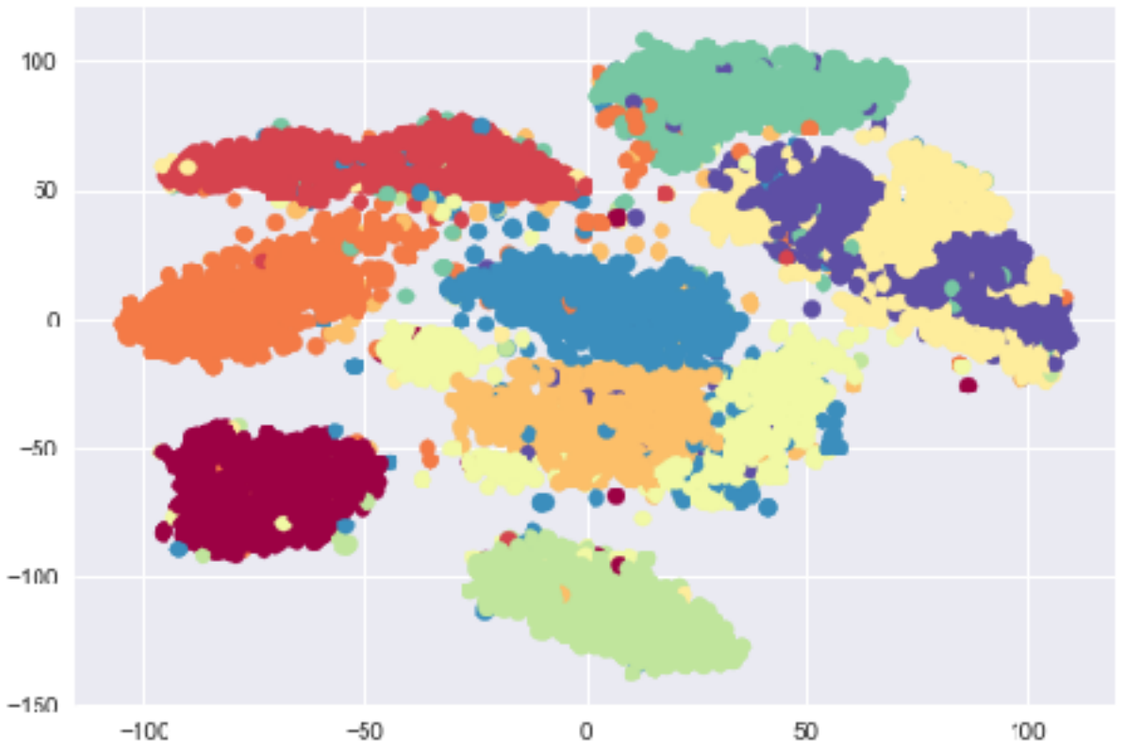
Part II: Predictive data analysis

Make predictions using known structure in data

- Classical classification methods
- Neural nets and deep learning for analyzing images and text

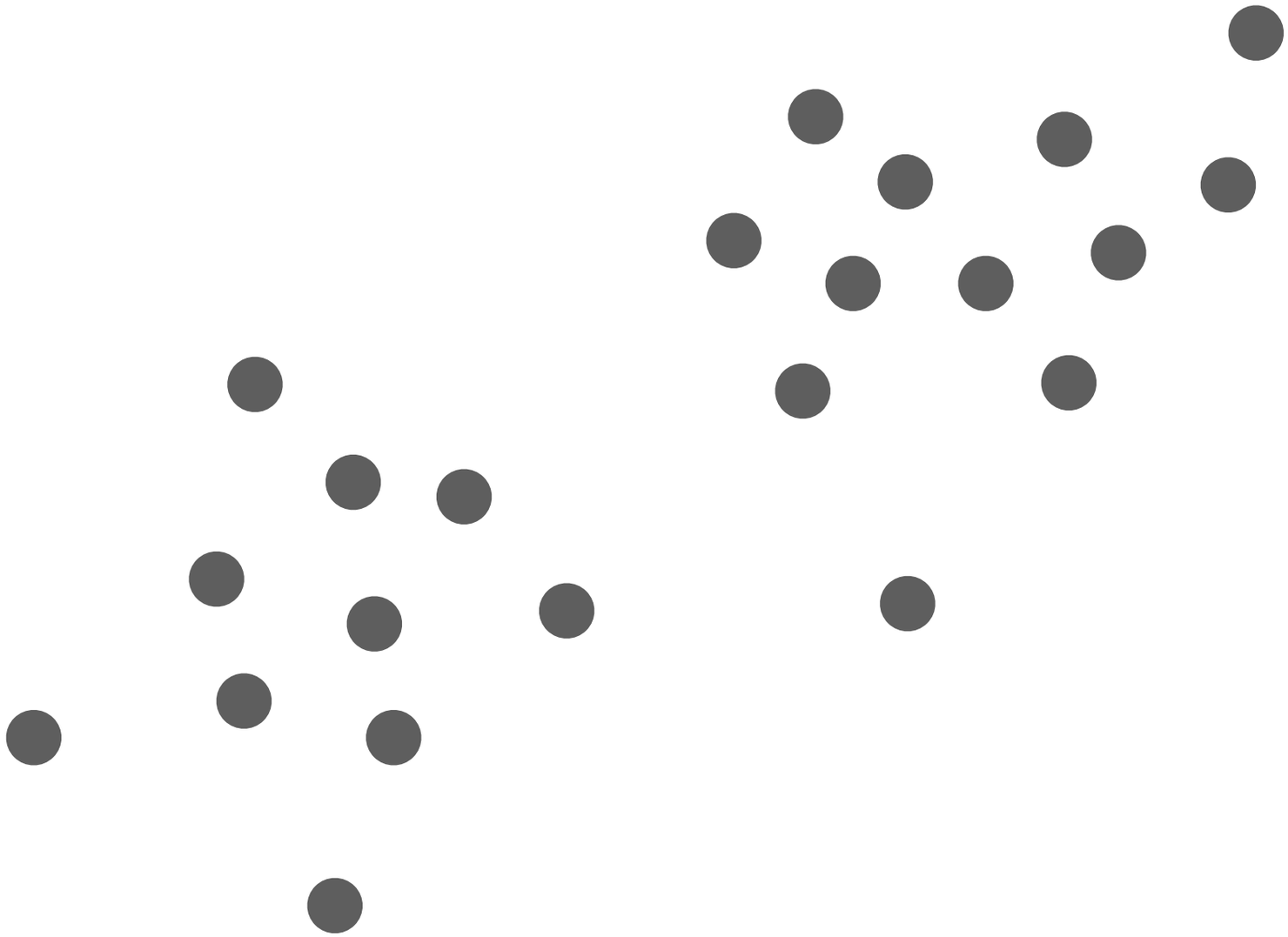
What if we have labels?

Disclaimer: unfortunately "*k*" means many things



Example: MNIST handwritten digits have known labels

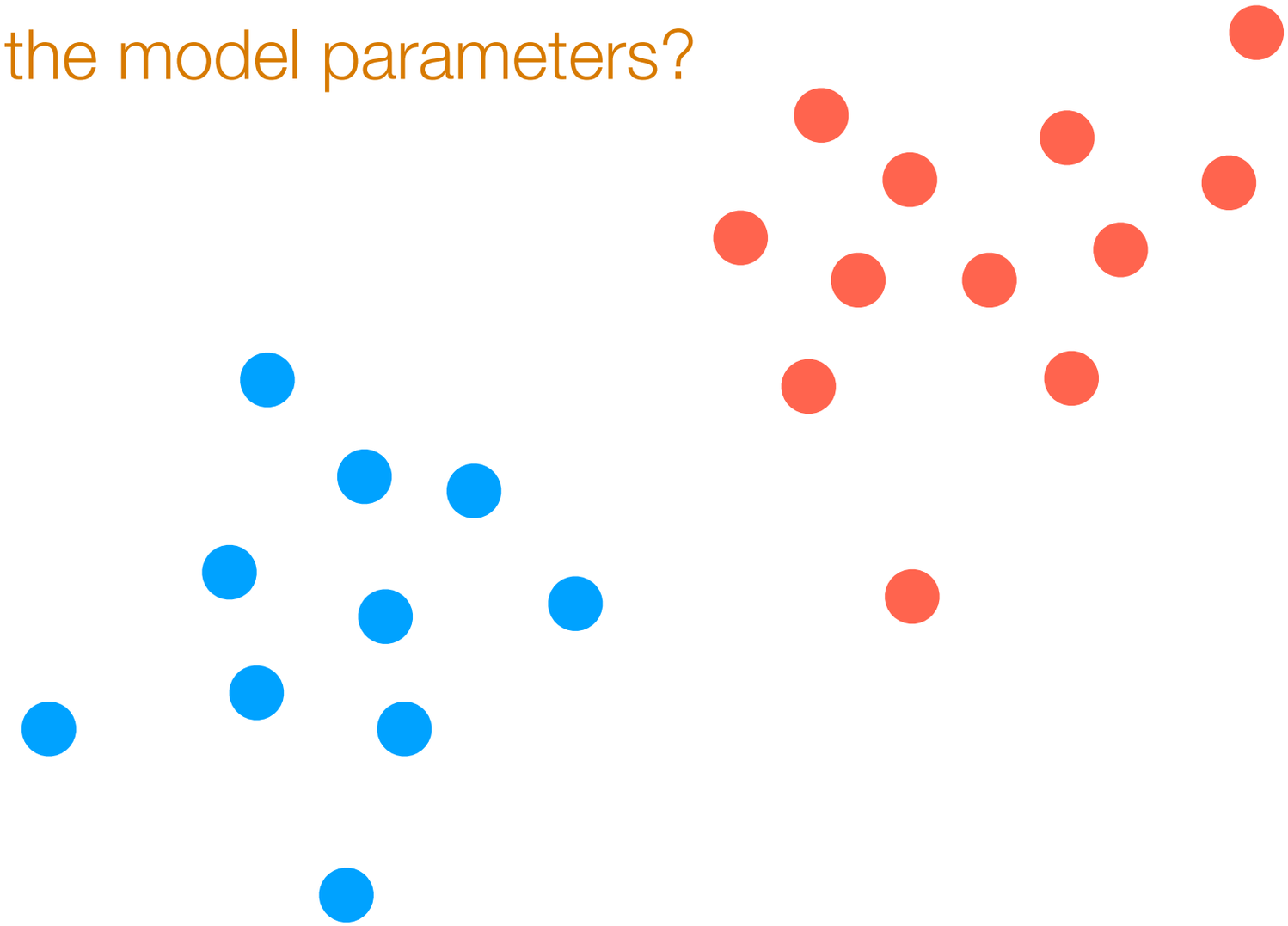
If the labels are known…



If the labels are known…

And we assume data generated by GMM…

What are the model parameters?



Flashback: Learning a GMM

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

Step 0: Pick k

Step 1: Pick guesses for **cluster probabilities, means, and covariances** (often done using k-means)

Repeat until convergence:

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

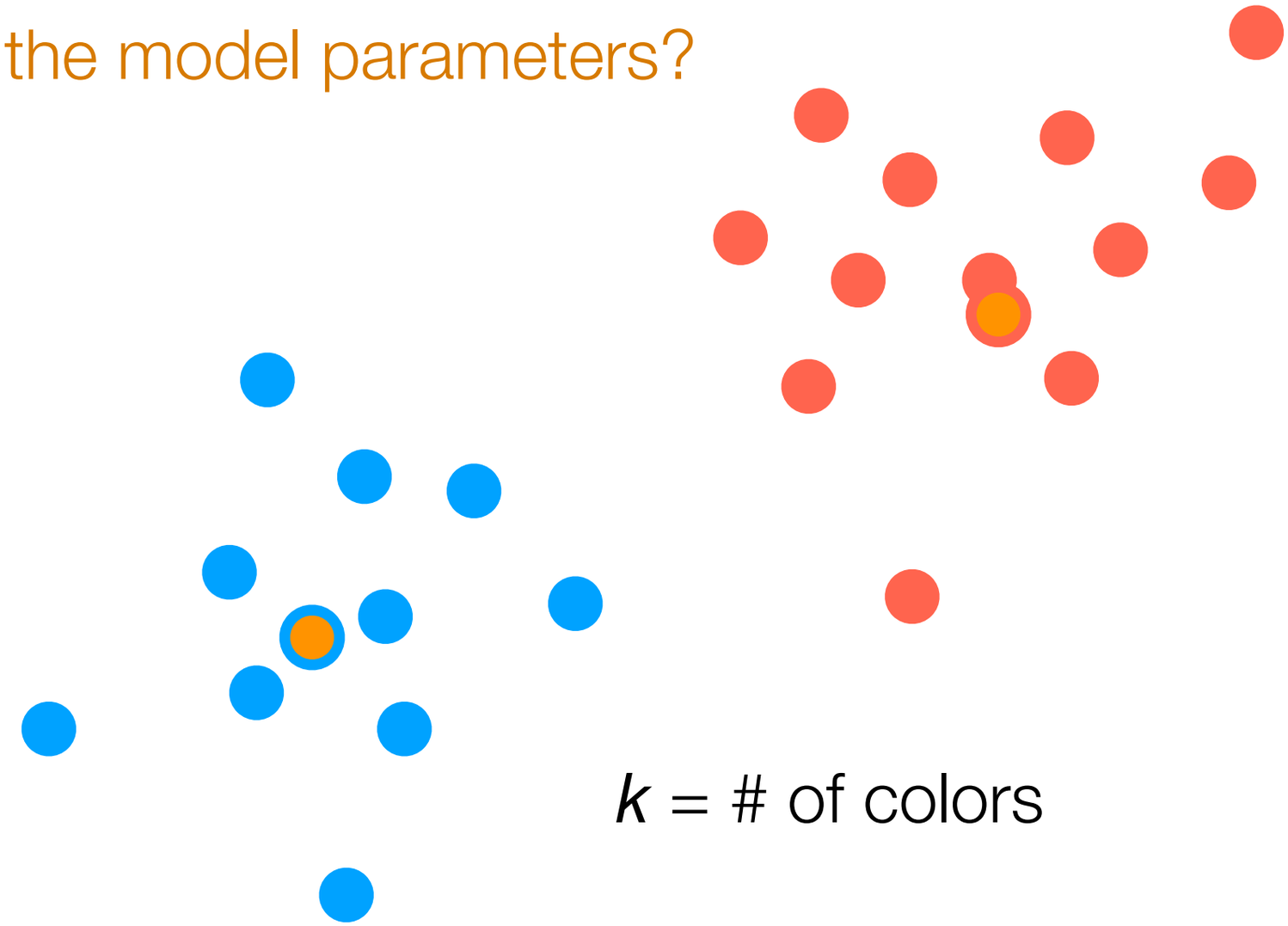
Step 3: Update **cluster probabilities, means, and covariances** carefully accounting for probabilities of each point belonging to each of the clusters

We don’t need to repeat until convergence

If the labels are known…

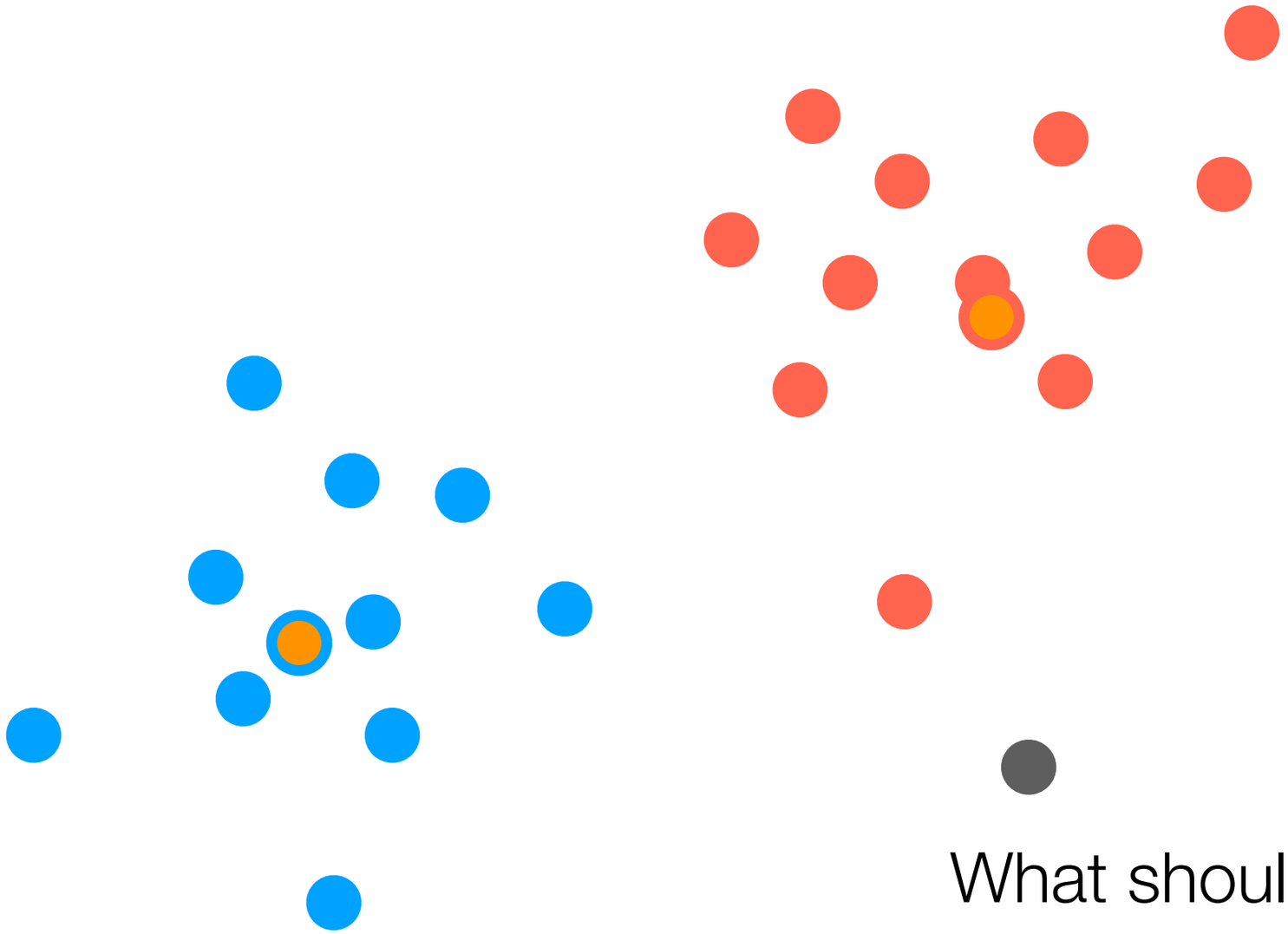
And we assume data generated by GMM…

What are the model parameters?



k = # of colors

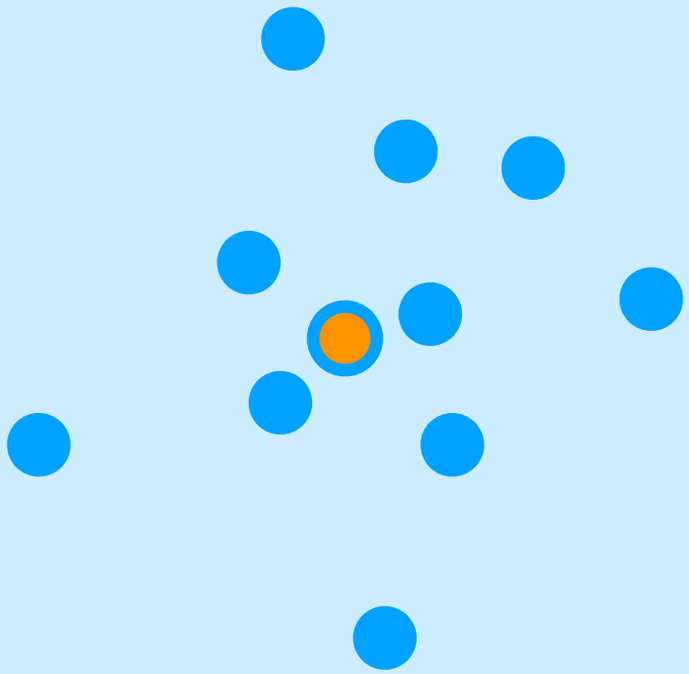
We can directly estimate cluster means, covariances



What should the label of this new point be?

Whichever cluster has higher probability!

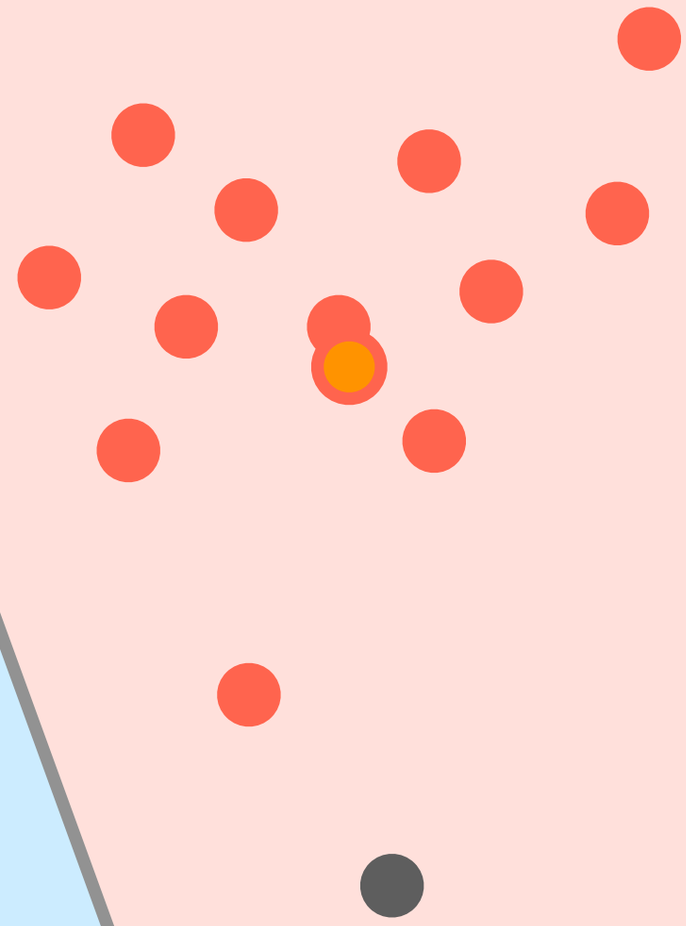
Decision boundary



This classifier we've created assumes a *generative model*

We just created a **classifier**

(a procedure that given a new data point tells us what "class" it belongs to)



What should the label of this new point be?

Whichever cluster has higher probability!

You’ve seen a prediction model that is partly a generative model

Linear regression!

Model parameters: slope *m*, intercept *b*

y

Label
(1D in this case)

x

Feature vector
(1D in this case)

Label
(1D in this case)

y

Model parameters: slope *m*, intercept *b*

x

Feature vector
(1D in this case)

Note: Standard linear regression has no generative procedure for generating values of *x* though!

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

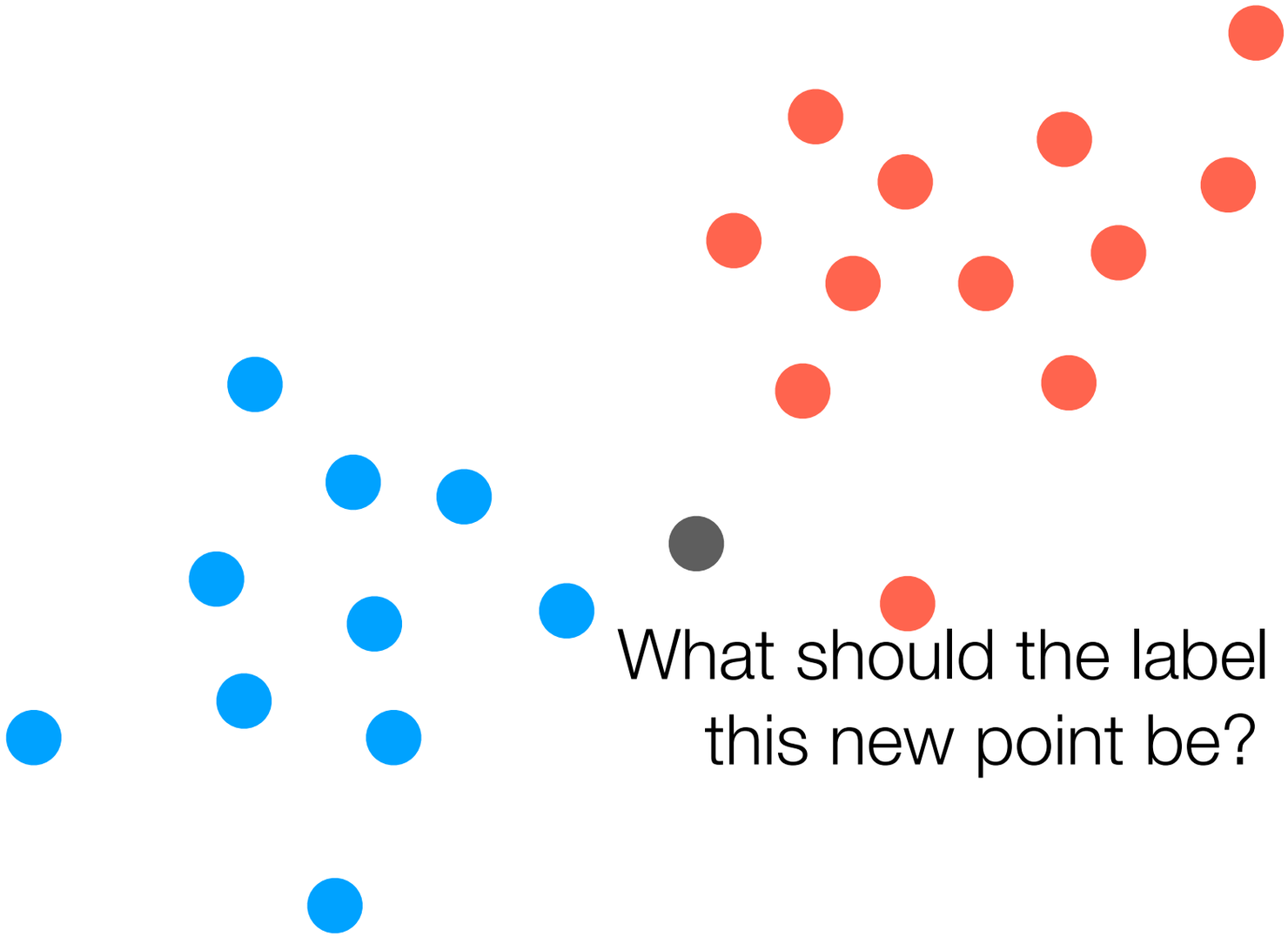
- y is discrete (such as colors red and blue)
  ➔ prediction method is called a **classifier**
- y is continuous (such as a real number)
  ➔ prediction method is called a **regressor**

A giant zoo of methods

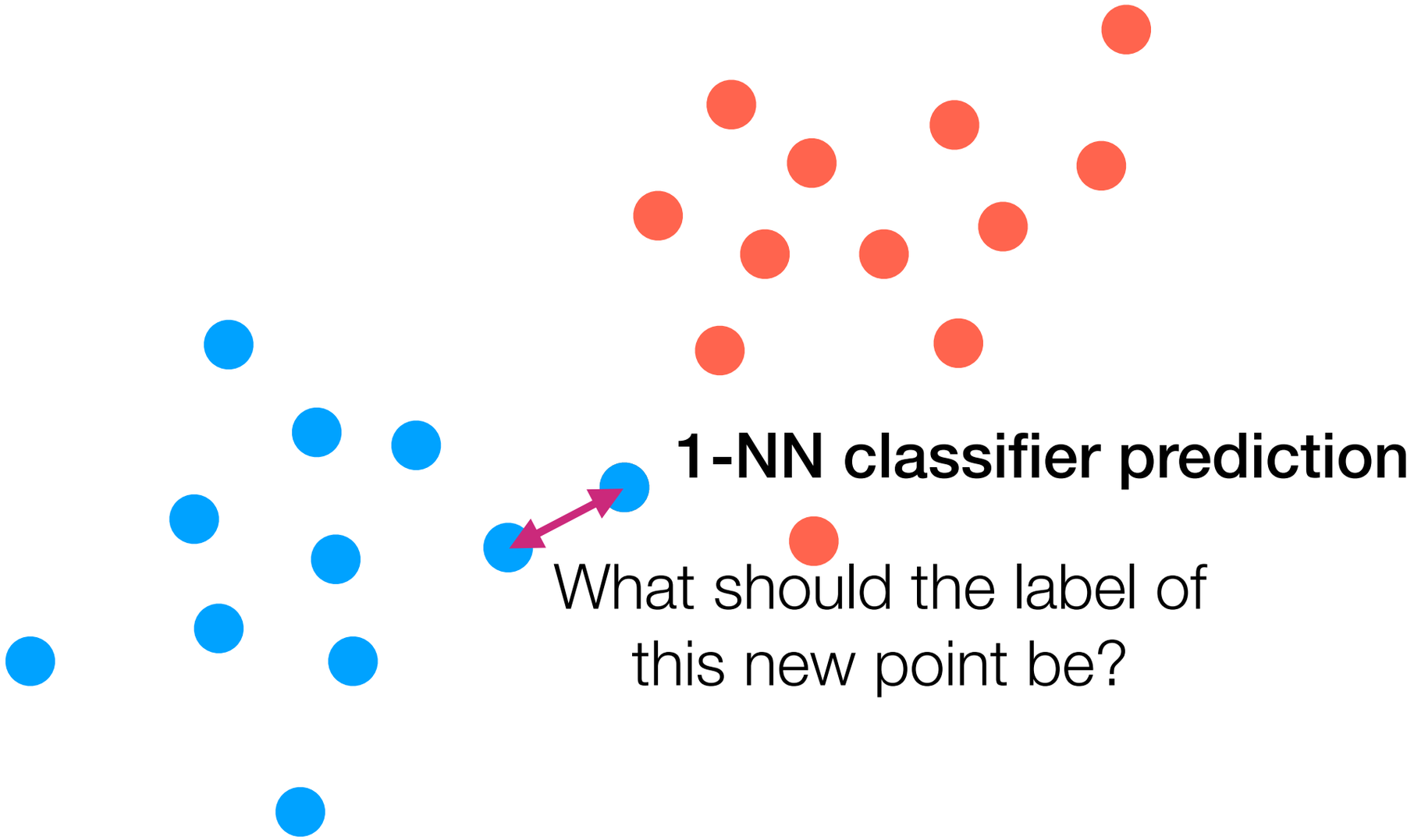
- Generative models (like what we just described)
- Discriminative methods (just care about learning prediction rule *without* assuming generative model)

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

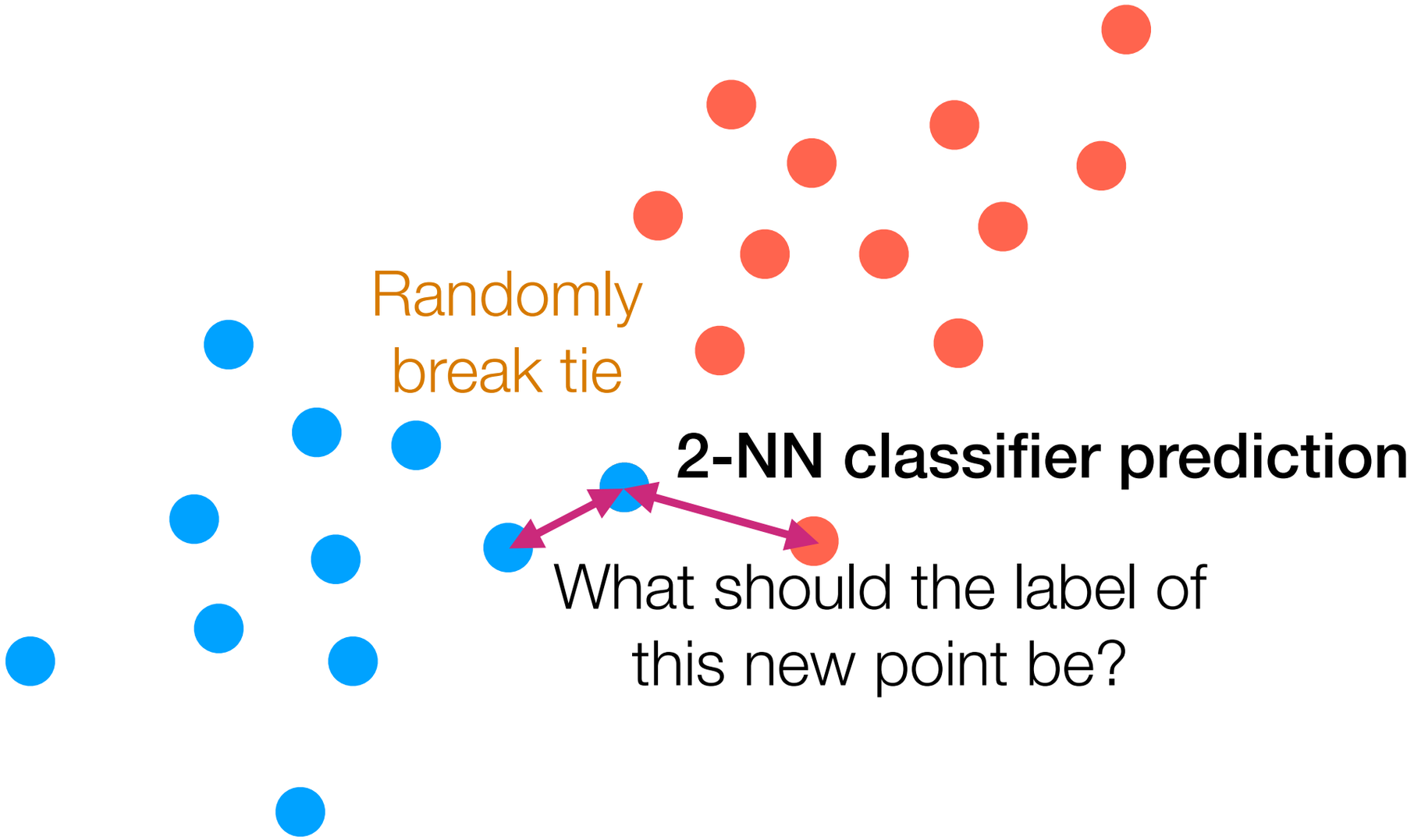
Example: *k*-NN Classification



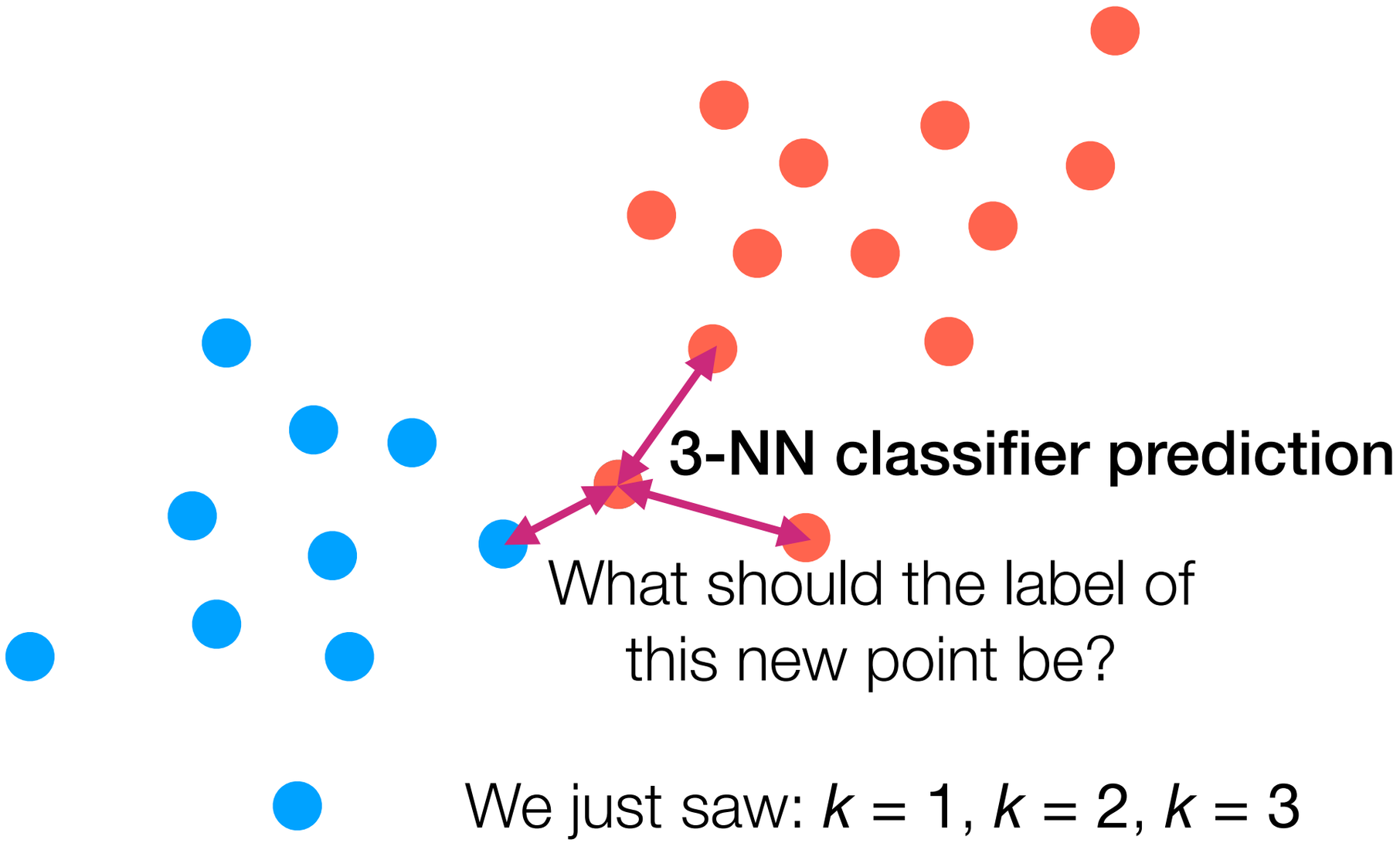
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification



What should the label of this new point be?

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

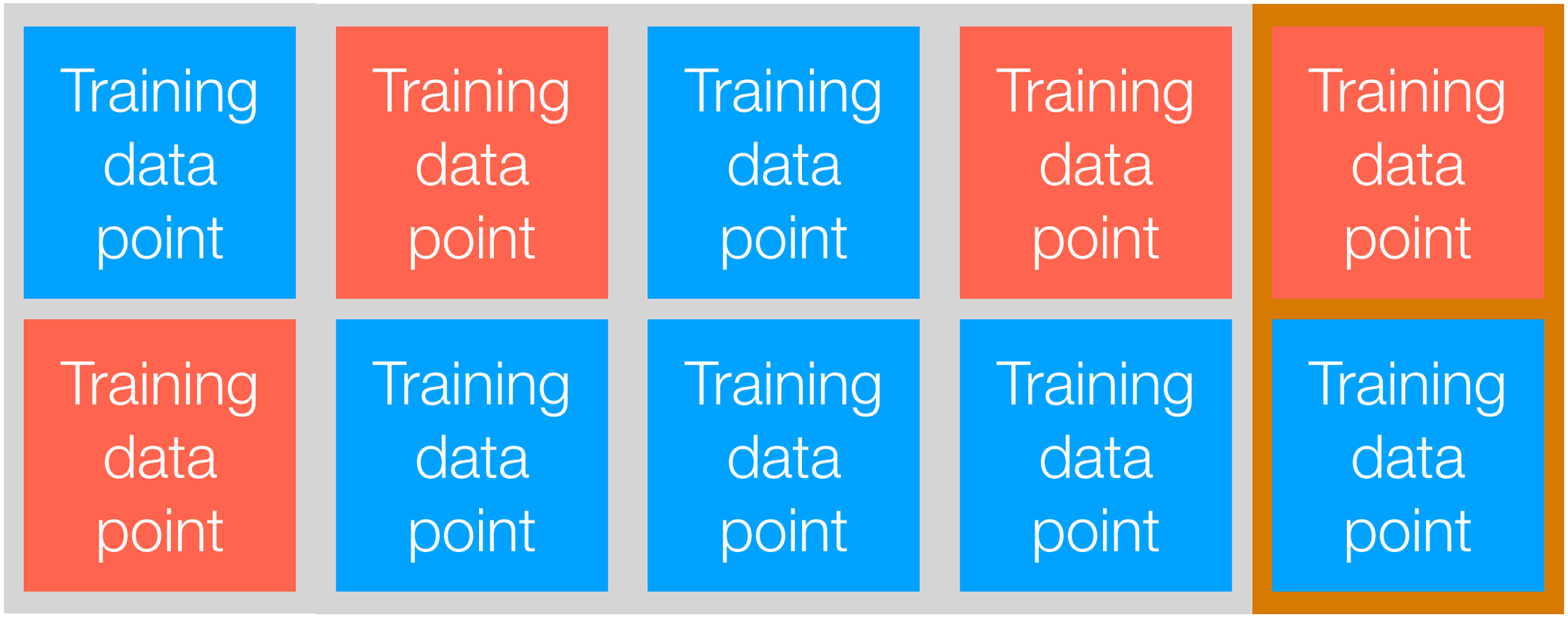
Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

Predicted labels



Train method on data in gray

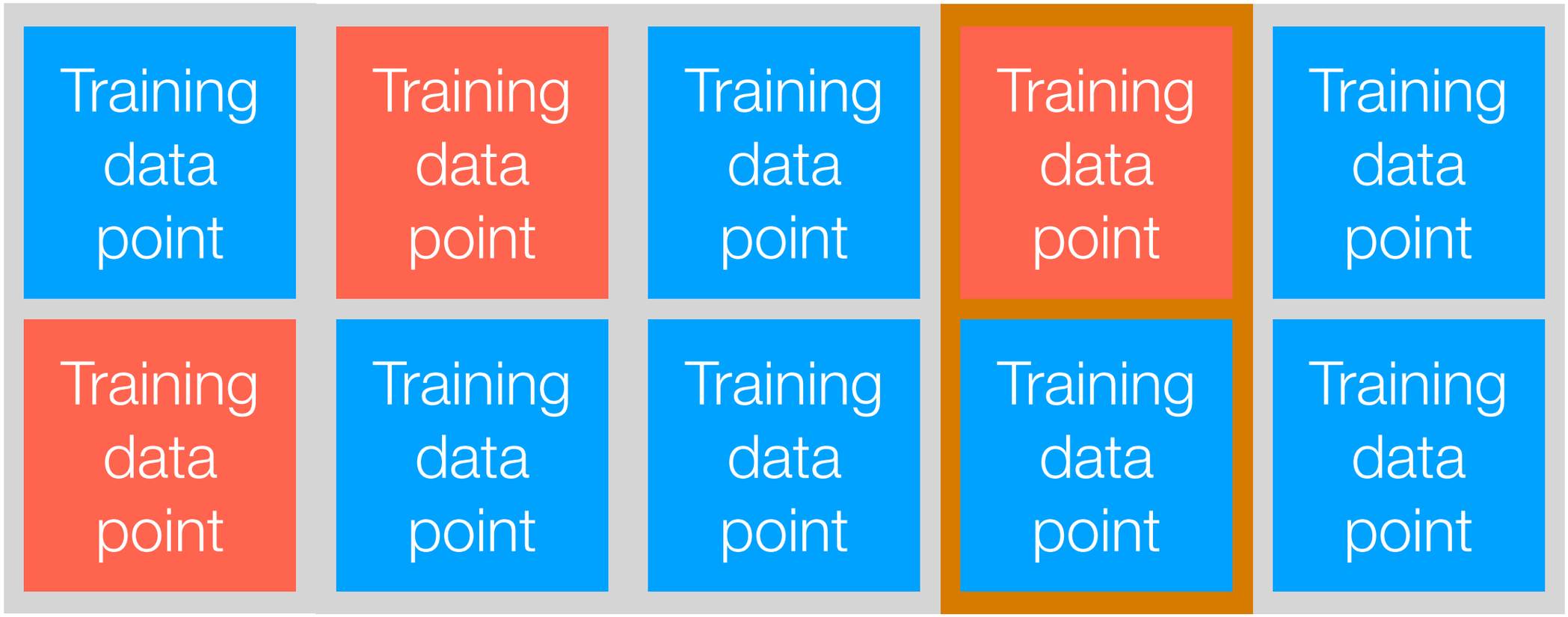
Predict on data in orange

Compute prediction error

50%

Data splitting/“train test split”

In this example: we did a 80%-20% split



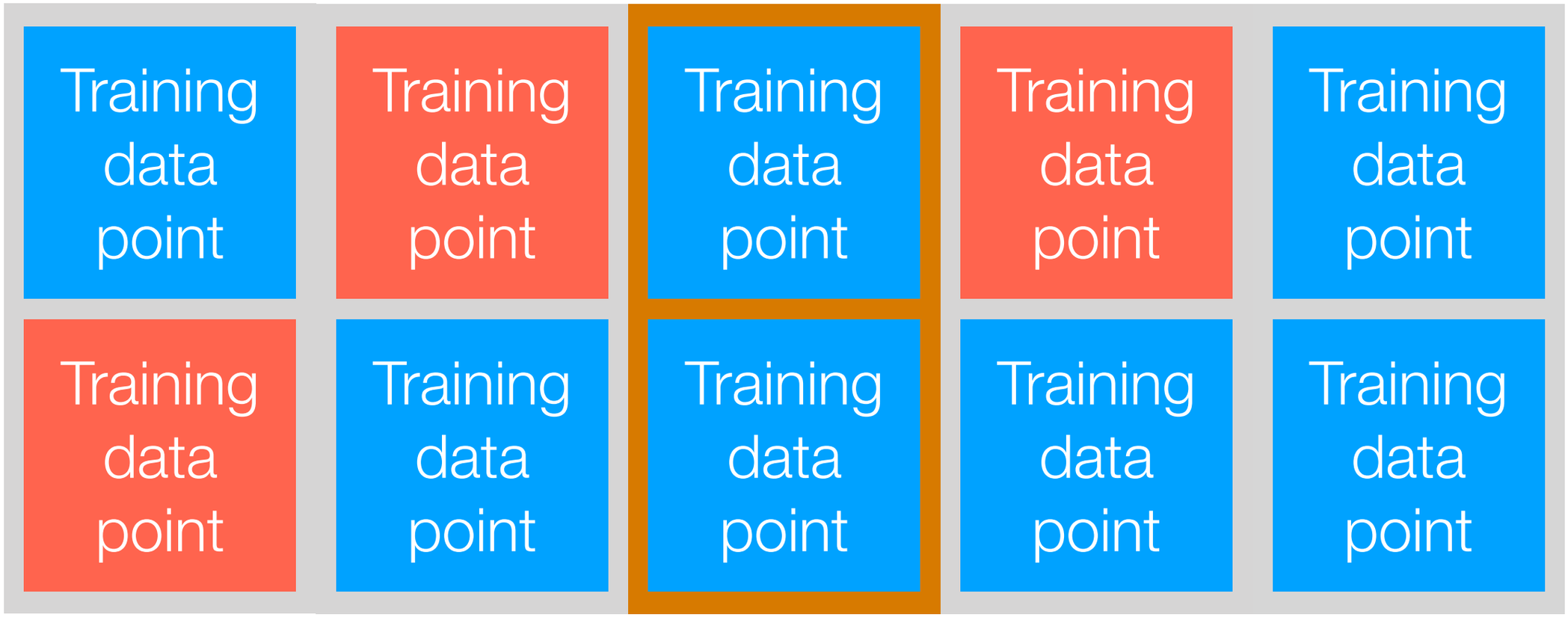
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%



Train method on data in gray

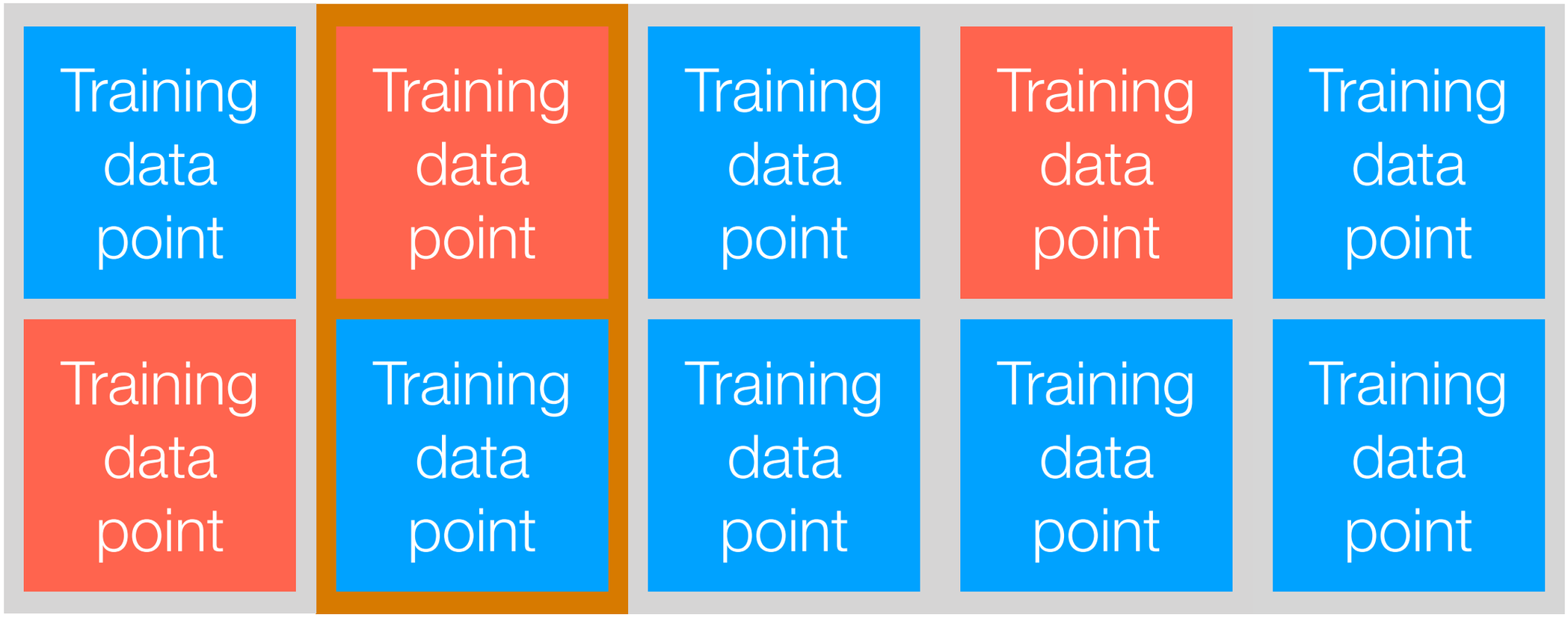
Predict on data in orange

Compute prediction error

50%

0%

50%



Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%

0%

50%

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

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%

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

1. Shuffle data and split them into 5 roughly equal size portions
2. For each “fold” (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

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

1. Shuffle data and split them into 5 roughly equal size portions

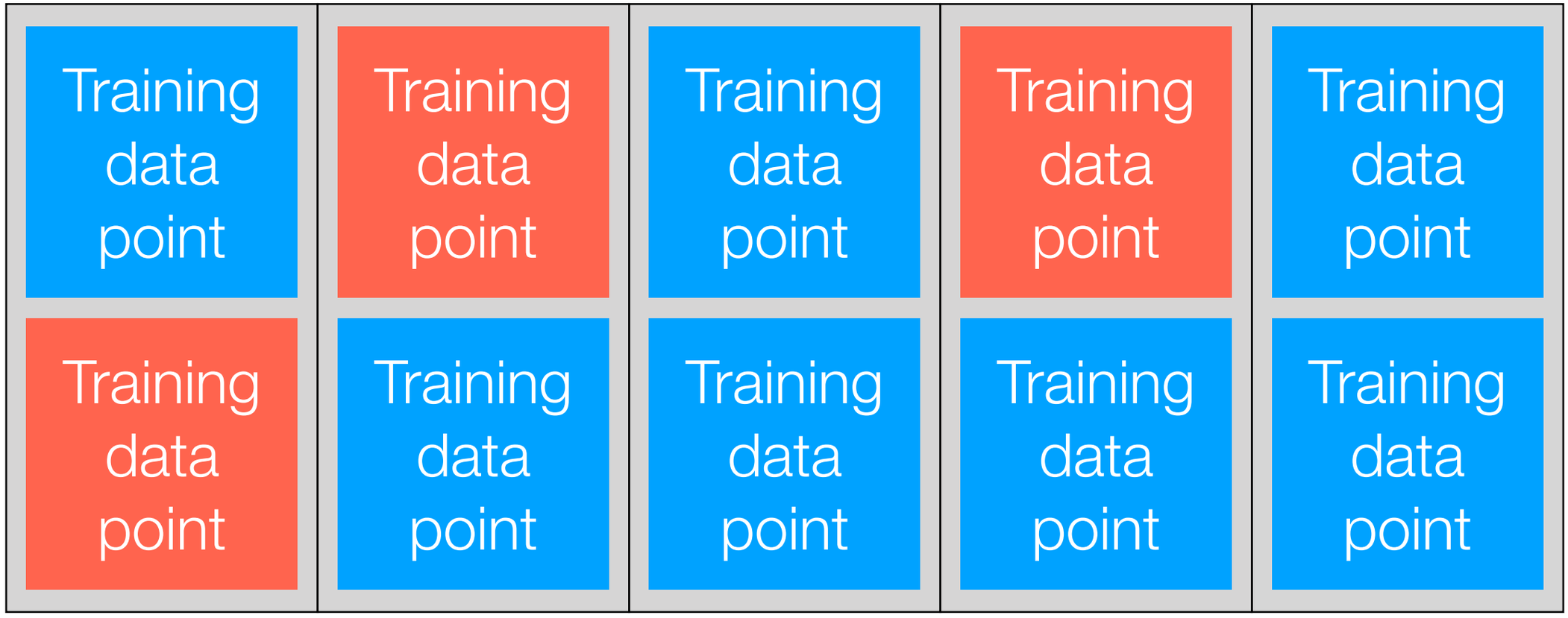
$k = 5$

2. For each “fold” (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 split them into 5 roughly equal size portions

$k = 5$

2. For each "fold" (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

For each $k = 1, 2, 3, \ldots$, 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

Important: the errors from simple data splitting and cross-validation are *estimates* of the true error on test data!

Example: earlier, we got a cross validation score of 20% error

This is a guess for the error we will get on test data

This guess is <u>not</u> always accurate!

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

Cross-Validation Remarks

- *k*-fold cross-validation is a randomized procedure
  - Re-running CV results in different cross-validation scores!
- Suppose there are *n* training data points and *k* folds
  - If we are trying 10 different hyperparameter settings, how many times do we do model fitting? $10k$
    - If this number is similar in size to *n*, CV can overfit!
  - How many training data are used in each model fit during cross-validation? $[(k-1)/k]n$
    - Smaller # folds typically means faster training
- If $k = n$, would re-running cross-validation result in different cross-validation scores? What about $k = 2$?
    - For deterministic training procedure: same CV result for $k = n$ (since shuffling doesn't matter), different for $k = 2$

Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

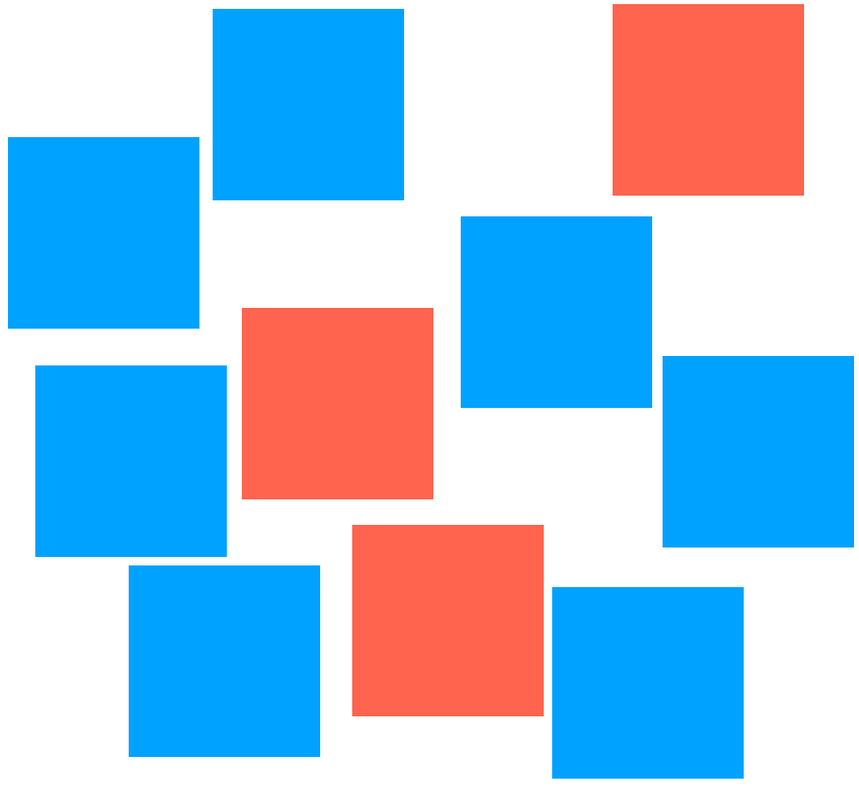
In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

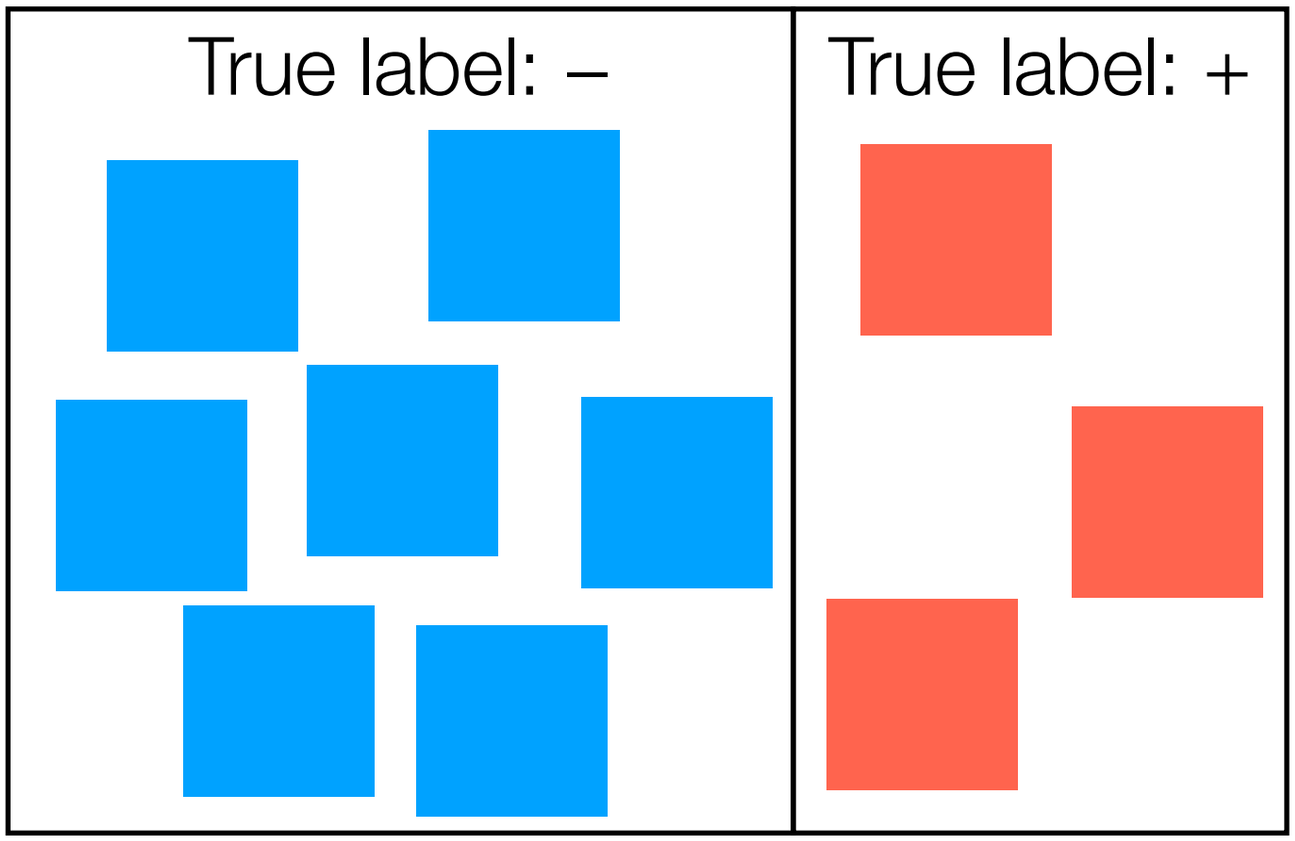


Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In "binary" classification (there are 2 labels such as spam/ham) when 1 label is considered "positive" and the other "negative":



Different Ways to Measure Accuracy

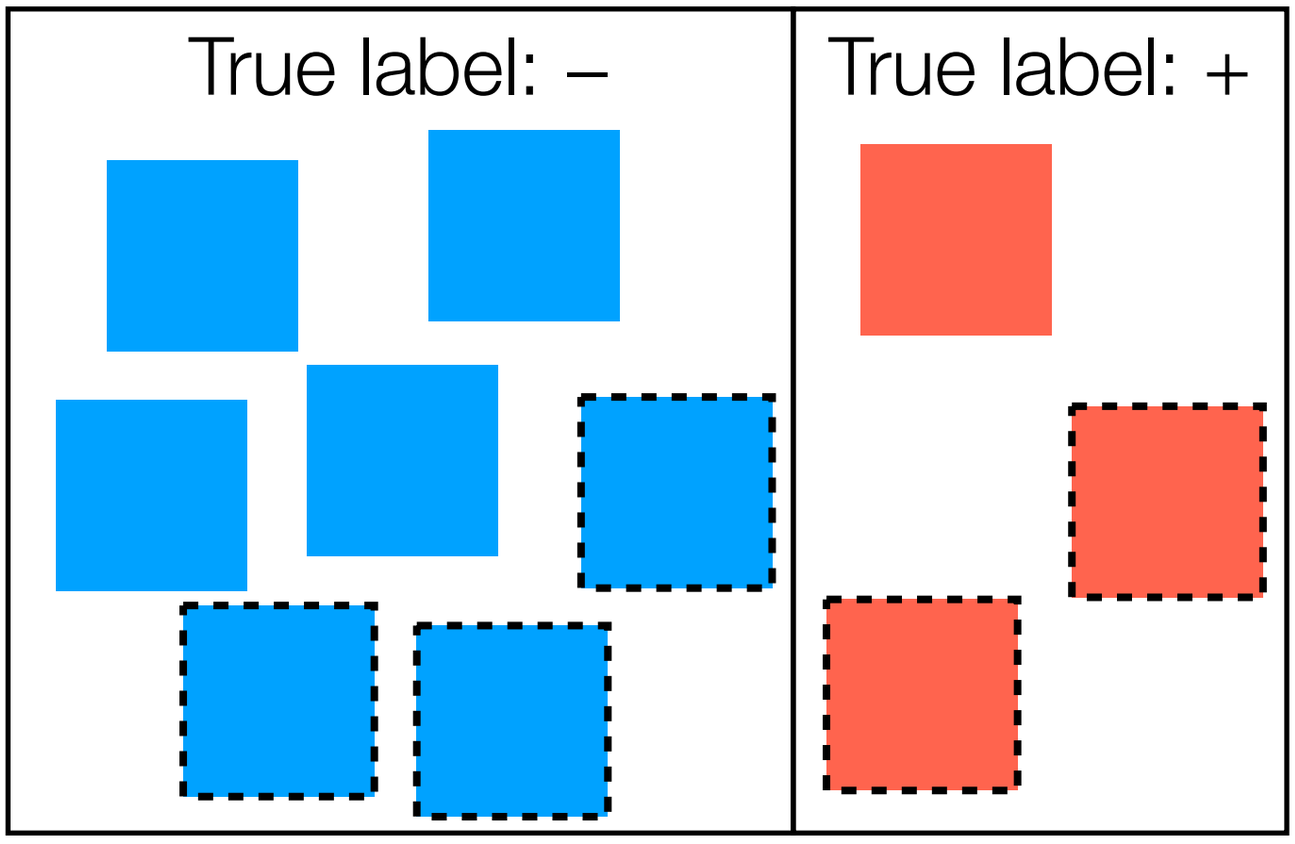
Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In "binary" classification (there are 2 labels such as spam/ham) when 1 label is considered "positive" and the other "negative":

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of dotted line in true label + = 2/3

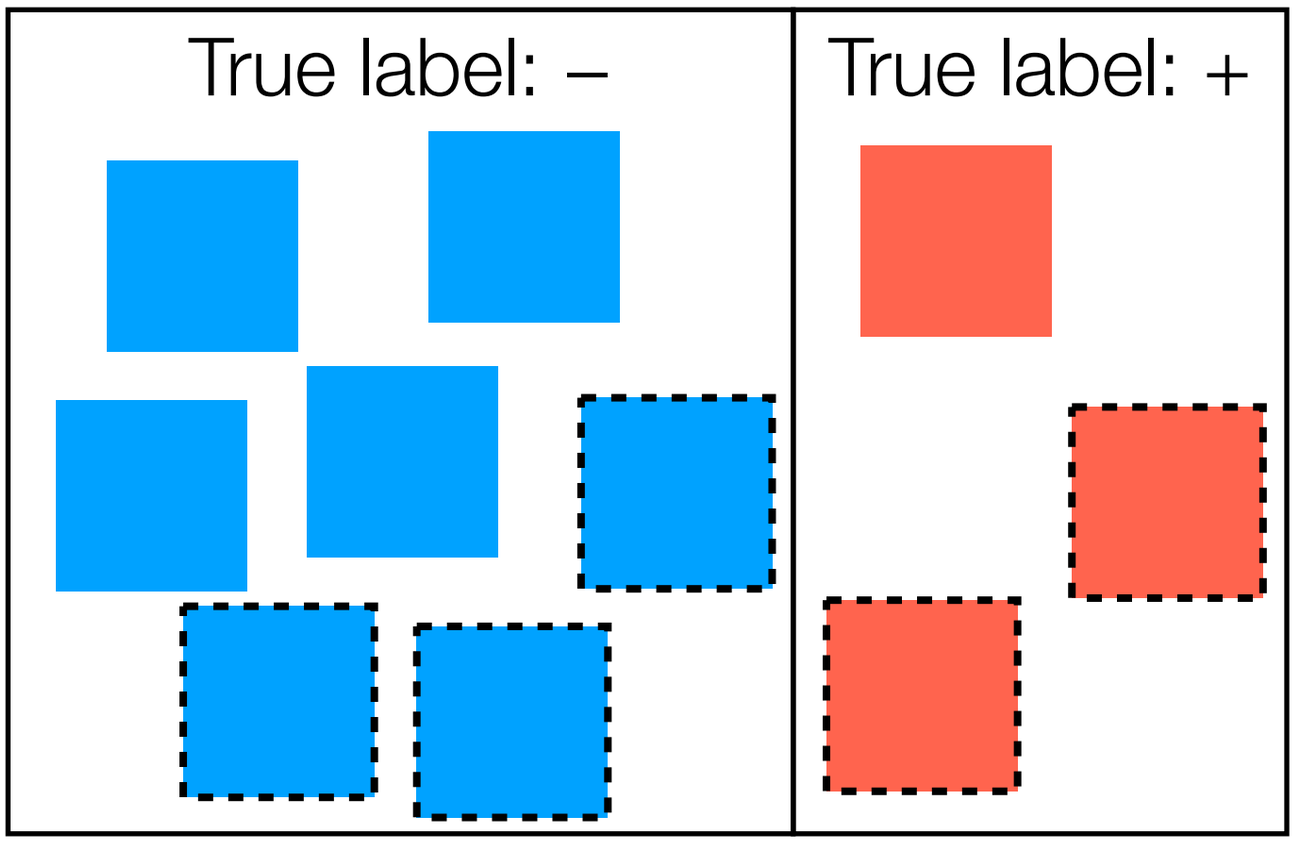
Precision: fraction of + in dotted line

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In "binary" classification (there are 2 labels such as spam/ham) when 1 label is considered "positive" and the other "negative":

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of dotted line in true label + = 2/3

Precision: fraction of + in dotted line = 2/5

False Positive Rate: fraction of dotted line in true label – = 3/7

F1 score: $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 1/2$

Prediction and Model Validation

Demo