Carnegie Mellon University
Heinz College

Some Classics of Classification

Nearest neighbors, evaluating prediction methods, naive Bayes

George Chen

CMU 95-865 Spring 2018

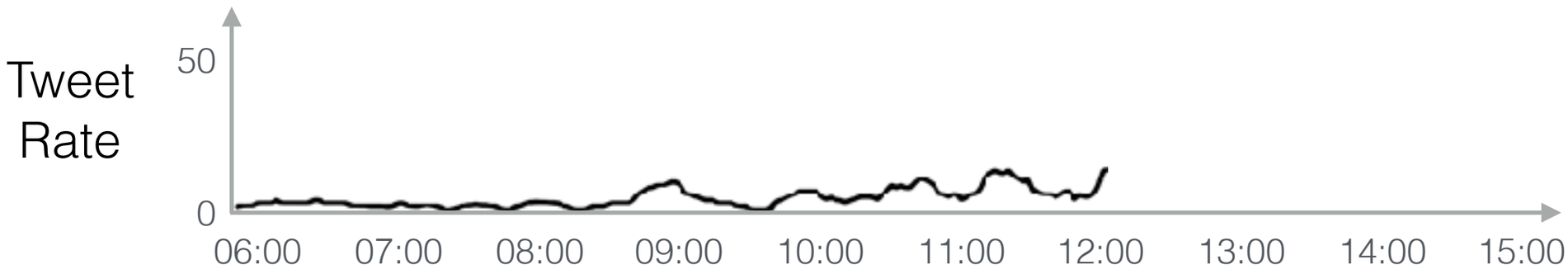
Announcements

- HW 2 due next Monday Feb 19, 10:30am instead
    - Please concentrate your mental firepower on prepping for the quiz
    - Lots of questions on material in HW2 not yet covered in lecture (that will be hopefully covered today)
    - No point in releasing HW3 on Wednesday anyways (lecture coverage for HW3 only starts next Monday)
- I will *not* be holding my regular office hours this week (I'm giving a talk out of town…)

Disclaimer: unfortunately "***k***"
means many things

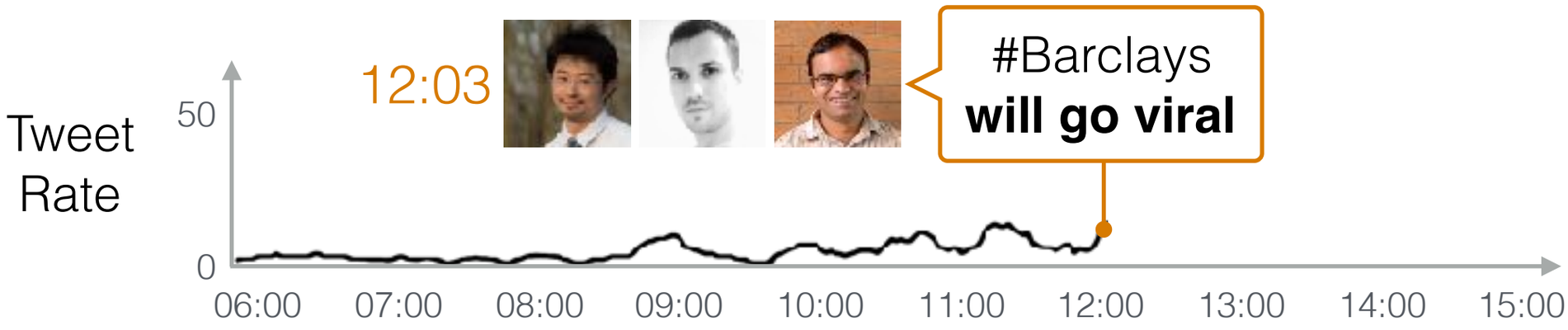

BARCLAYS

News Activity for #Barclays



GMT Time (June 27, 2012)

News Activity for #Barclays



News Activity for #Barclays

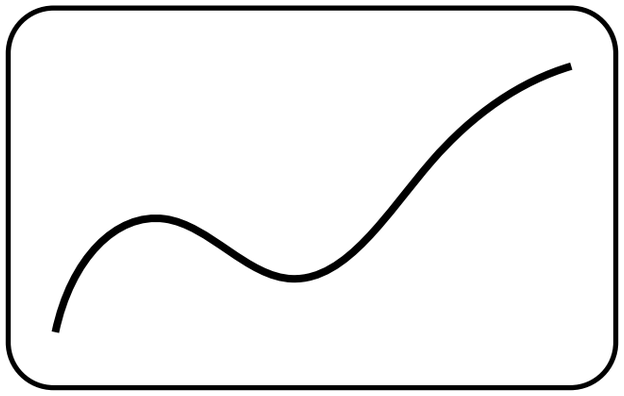
GMT Time (June 27, 2012)

How we did this: **weighted majority voting**

Chen, Nikolov, and Shah. A Latent Source Model for Nonparametric Time Series Classification. NIPS 2013.

Weighted Majority Voting

Test data



Election results

Viral: 1.3 votes

Not viral: 0.1 votes

Red = viral

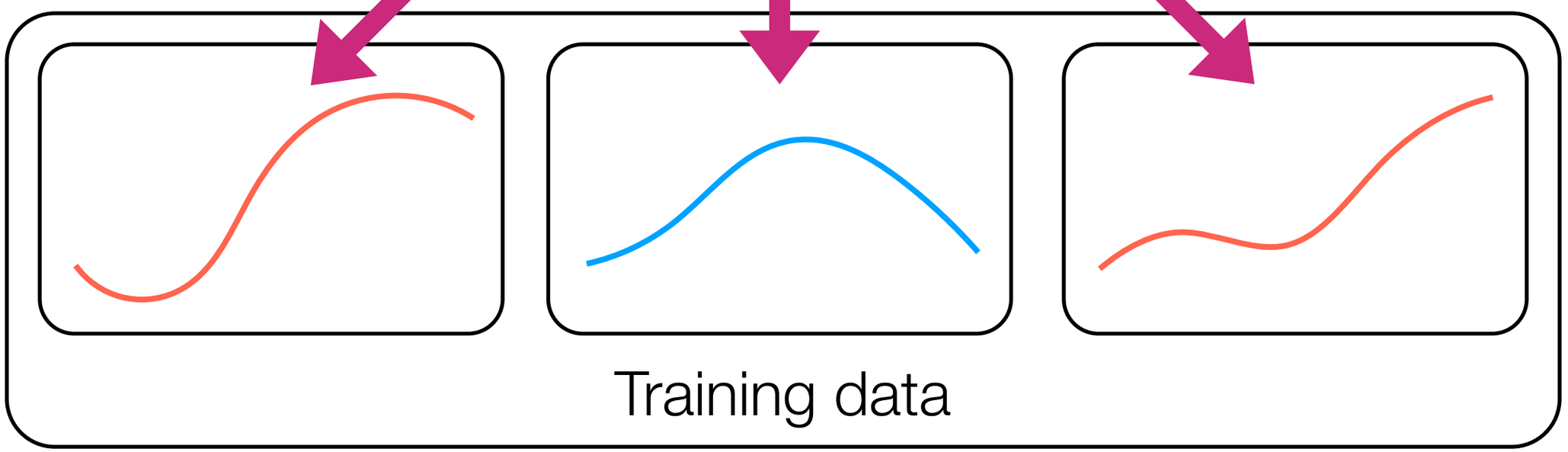
Blue = not viral

Compute similarities

0.5

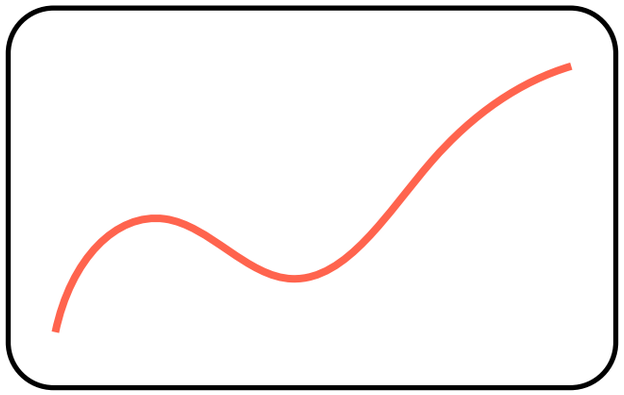
0.1

0.8



Nearest Neighbor Classification

Test data



Election results
Viral: **0.8** votes
Not viral: **0.0** votes

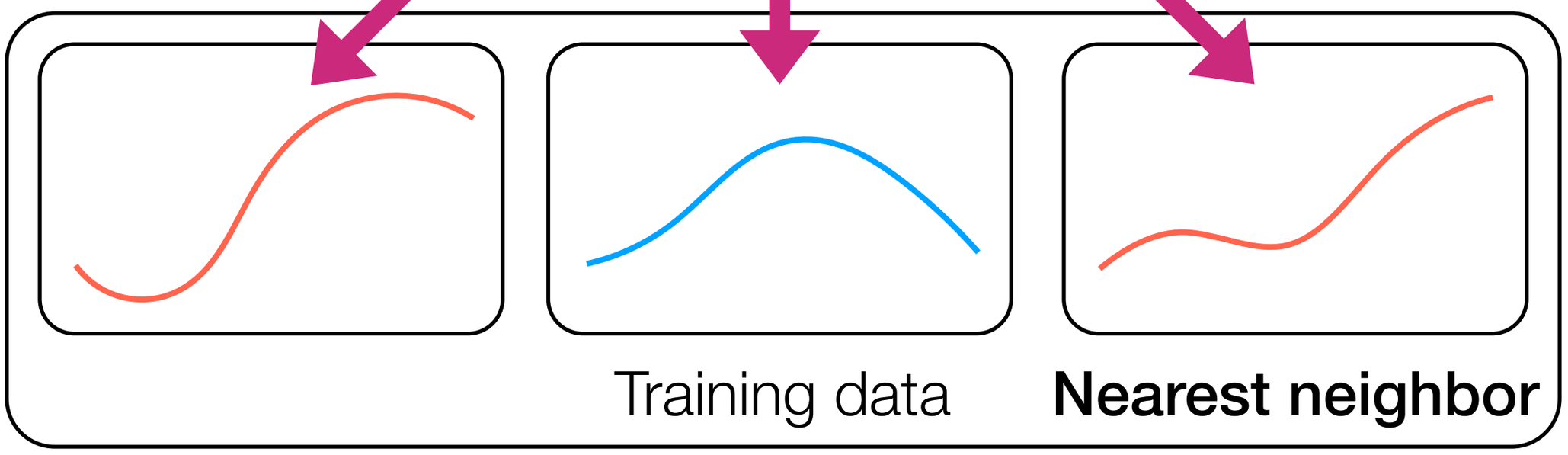
Red = viral
Blue = not viral

Compute similarities

0.5

0.1

0.8



Training data

Nearest neighbor

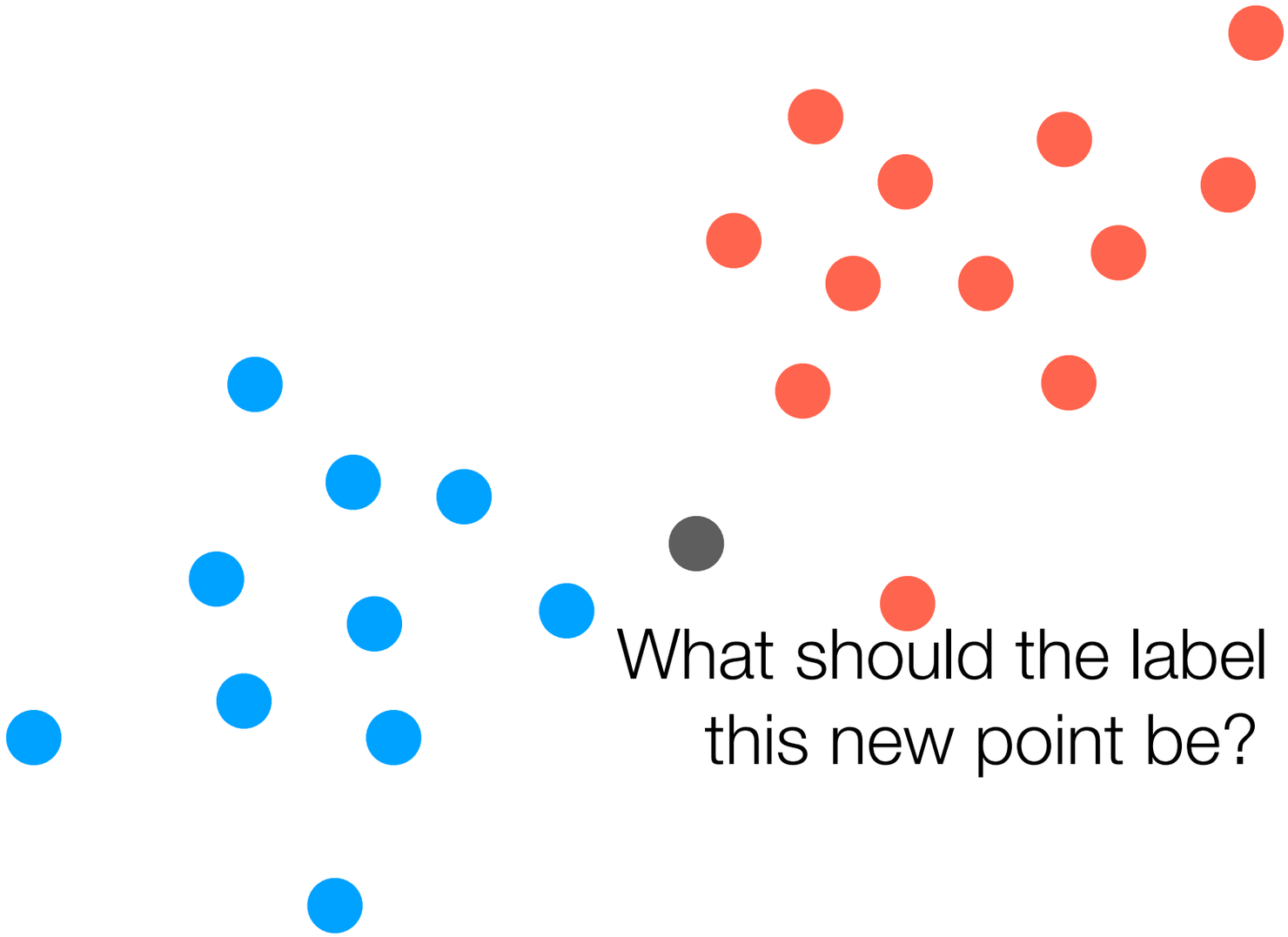
NN Classification Variants

not the same *k* as in *k*-means

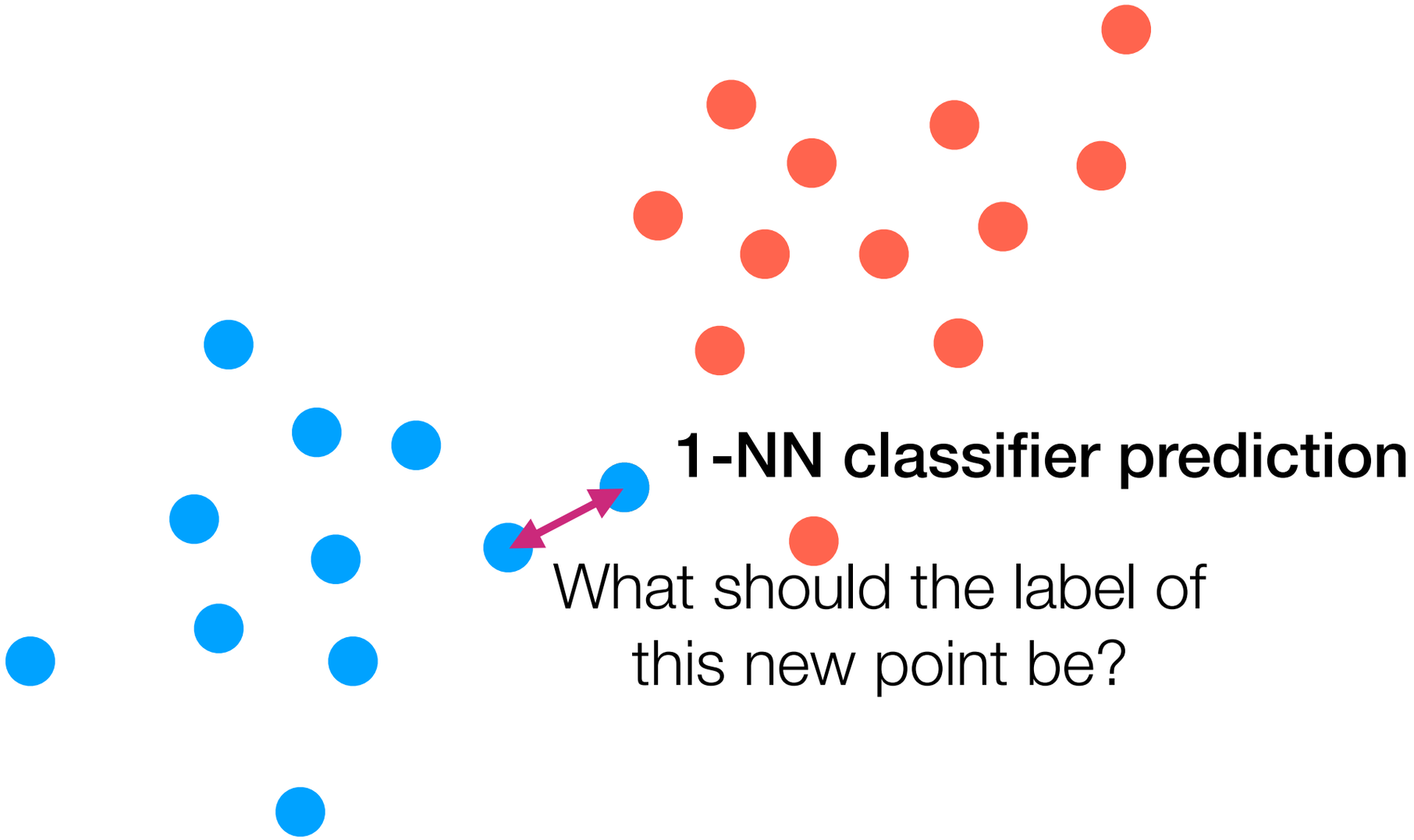
- ***k*-NN classification:** consider *k* most similar training data to test data point
  - **Weighted:** when tallying up votes, use the similarities that we computed
  - **Unweighted:** when tallying up votes, have each of the *k* nearest neighbors have an equal vote of 1
    (terminology: "*k*-NN classification" by default is unweighted)

- **Fixed-radius near neighbor classification:** consider all training data at least some similarity threshold close to test data point (i.e., use all training data distance ≤ *h* away)
  - Once again, can use weighted or unweighted votes

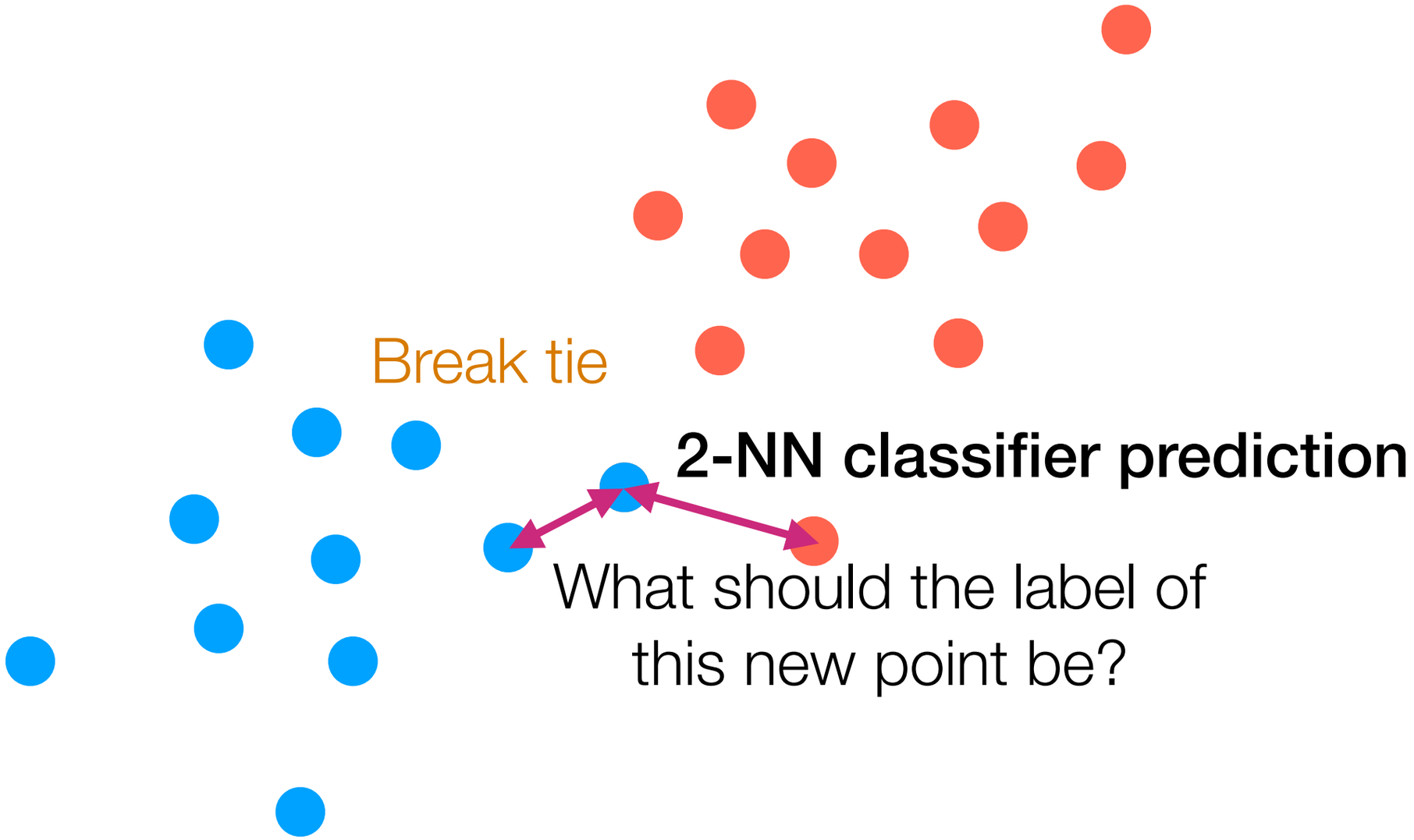
Example: *k*-NN Classification



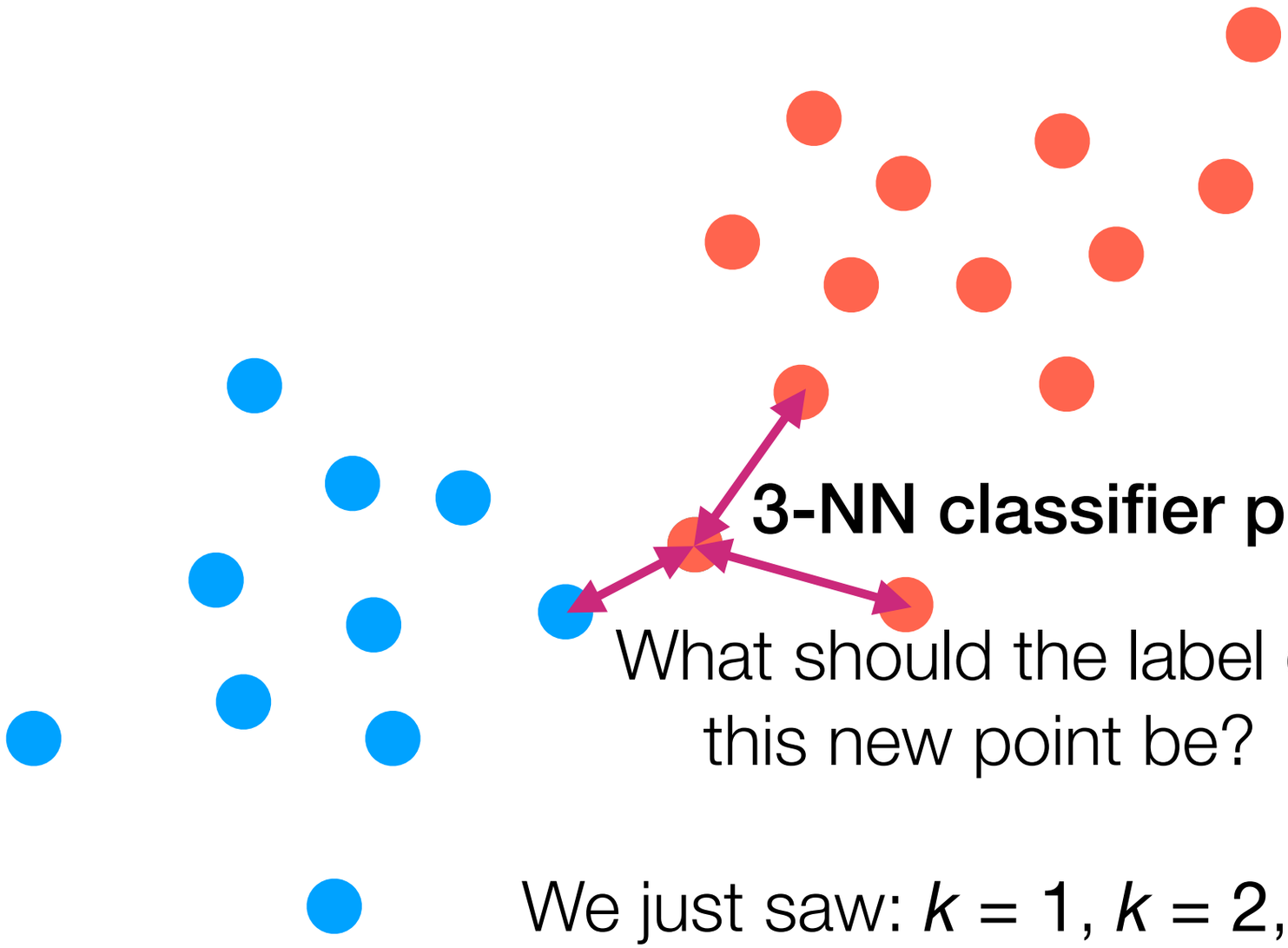
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification



3-NN classifier prediction

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 parameter 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

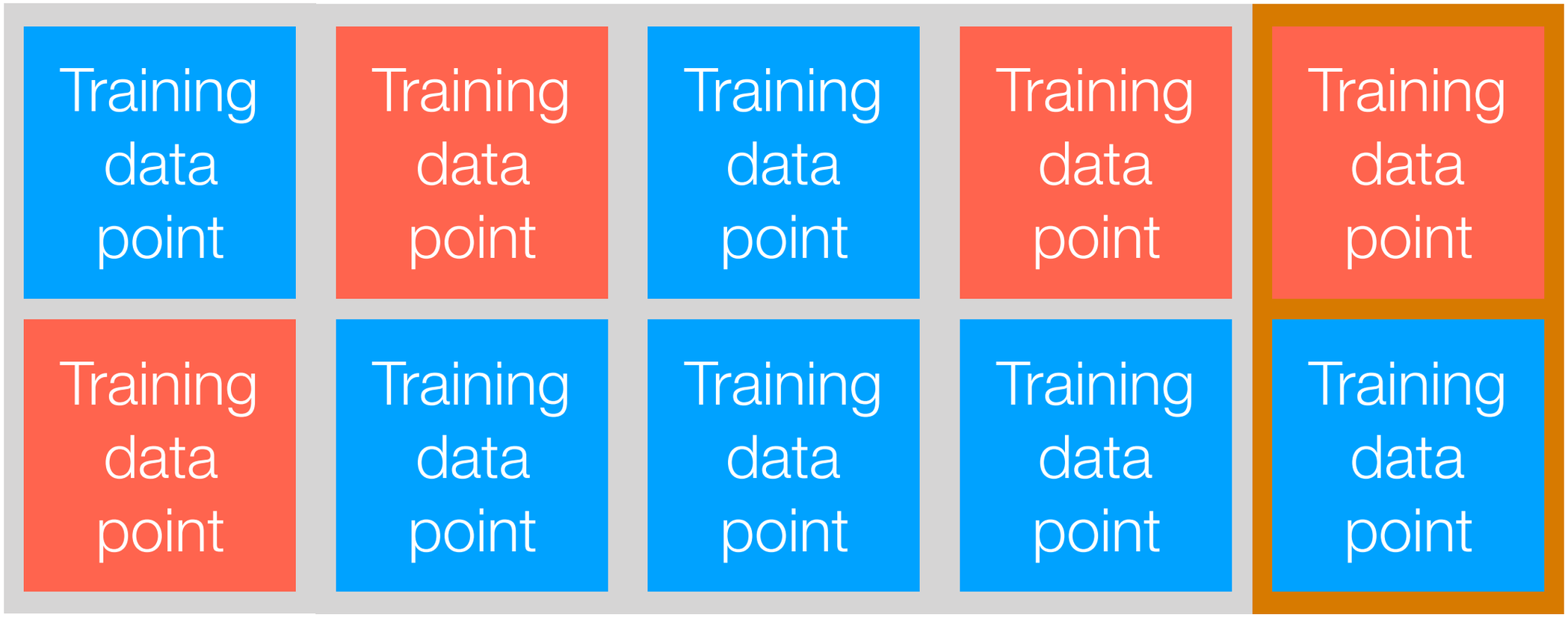
Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

Predicted labels

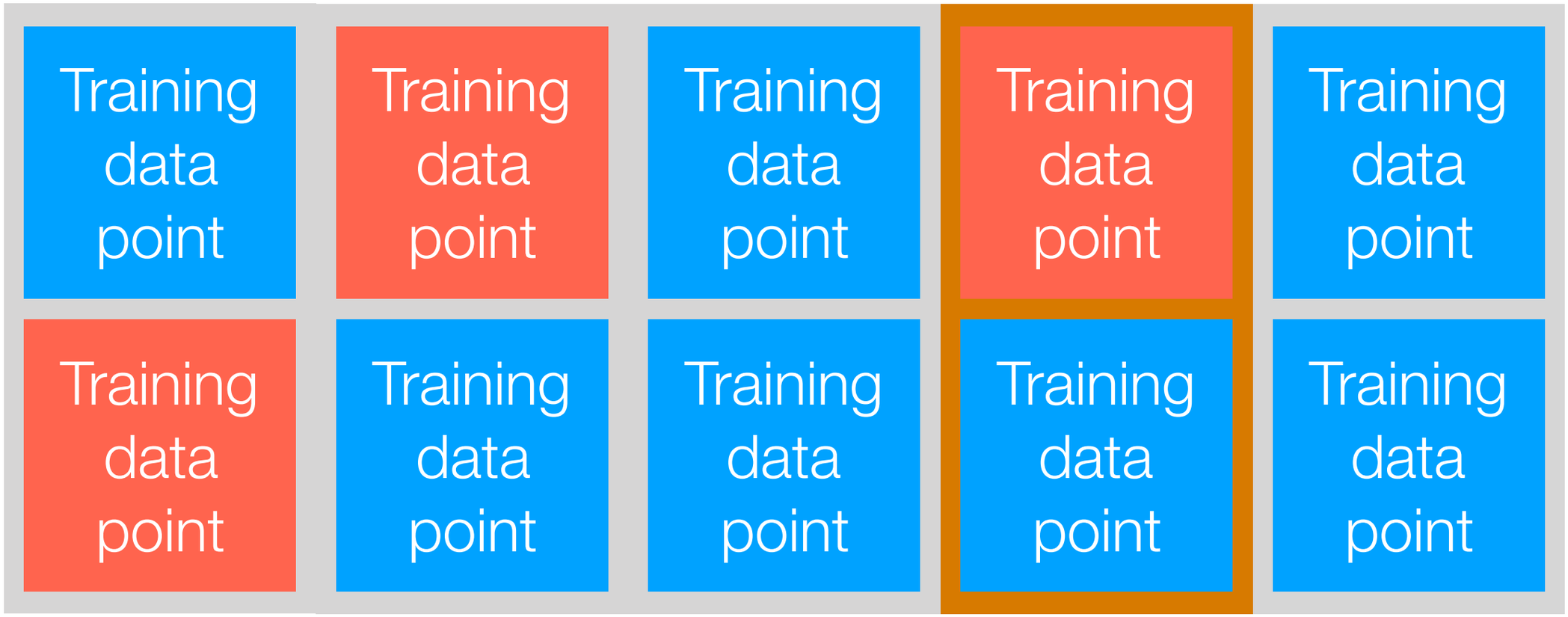


Train method on data in gray

Predict on data in orange

Compute prediction error

50%



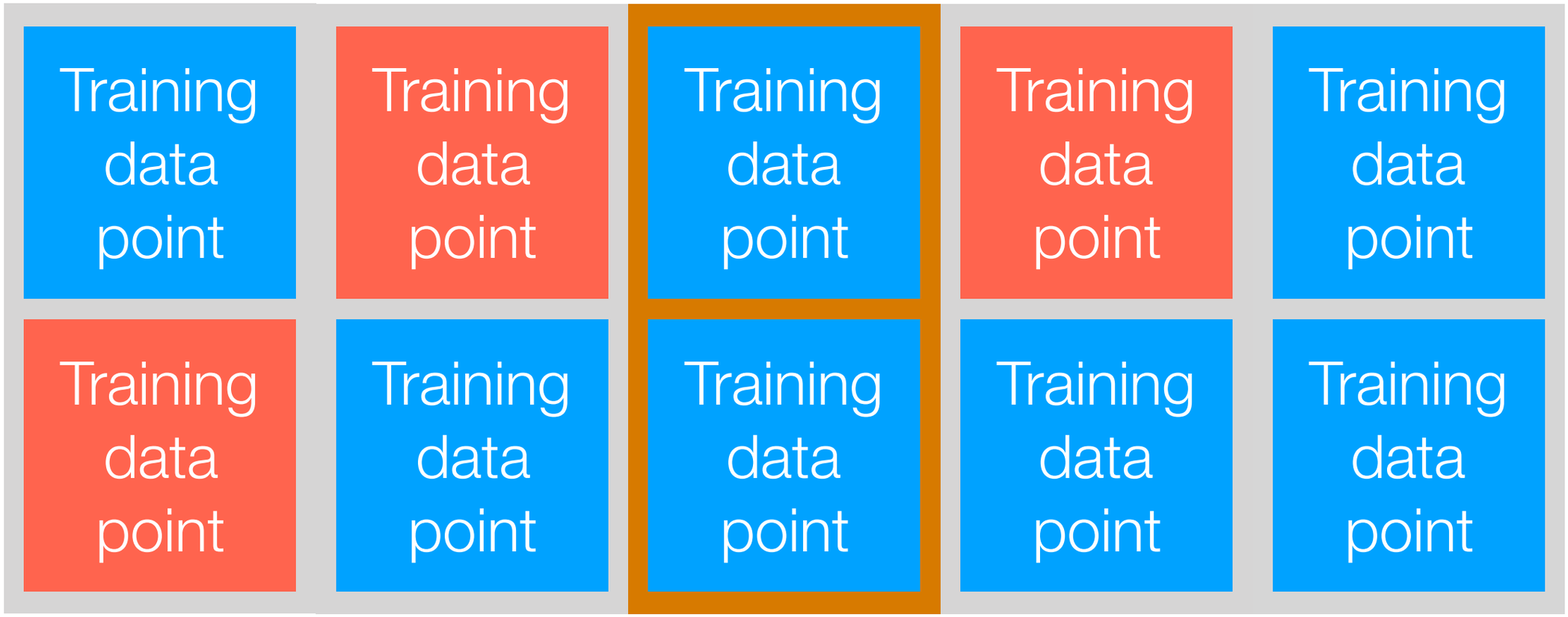
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%



Train method on data in gray

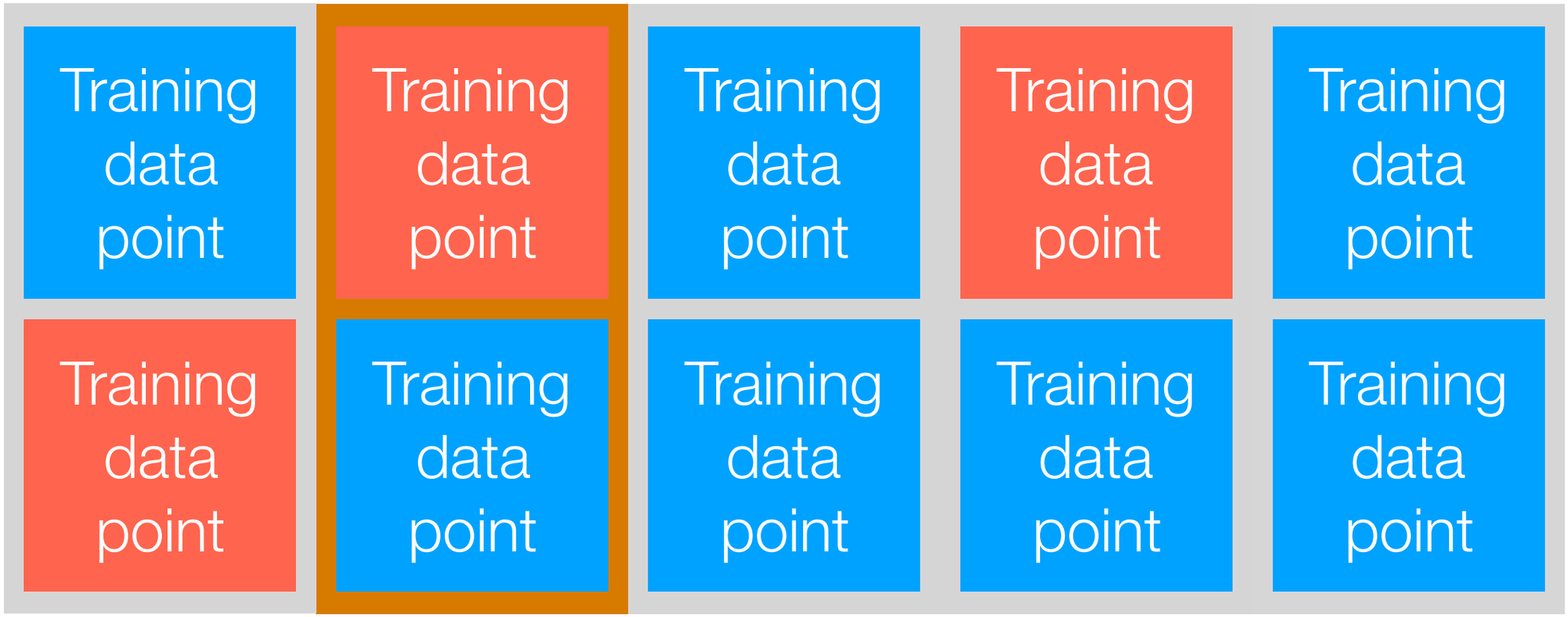
Predict on data in orange

Compute prediction error

50%

0%

50%



Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%

0%

50%

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 put them into “folds” (5 folds in this example)
2. For each fold:
   (a) Predict on the fold using model trained on all other folds
   (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 put them into “folds” (*k*=5 folds in this example)
2. For each fold:
   (a) Predict on the fold using model trained on all other folds
   (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 put them into “folds” (*k*=5 folds in this example)
2. For each fold:
   (a) Predict on the fold using model trained on all other folds
   (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 parameter *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

Important: the cross validation score is trying to predict what the prediction quality will be on the unseen test data

Our earlier example had a cross validation score of 20% error

This is a guess at how well the prediction method should perform 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

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”:

- **Precision:** among data points predicted to be “positive”, what fraction of these predictions is correct?
- **Recall:** among data points that are actually “positive”, what fraction of these points is predicted correctly as “positive”? (also called **true positive rate**)
- **F1 score:**

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Naive Bayes

(a generative model)

Many other ways to specify a naive Bayes model (features need not be binary)

Email spam classification example

Each email represented by feature vector saying whether a word is present or not (for pre-specified dictionary of words)

1. Flip coin with unknown probability s:
   If heads: new email is spam
   If tails: new email is ham
2. If new email is spam:
   For each word w in vocabulary:
   Flip coin with probability p_w for whether word w appears

   If new email is ham:
   For each word w in vocabulary:
   Flip coin with probability q_w for whether word w appears

Whether one word appears has no effect on whether another word appears!
(why model is called “naive”)

How many parameters are there in this example?