

Unstructured Data Analysis for Policy

Lecture 9: Topic modeling (cont'd);
intro to predictive data analytics

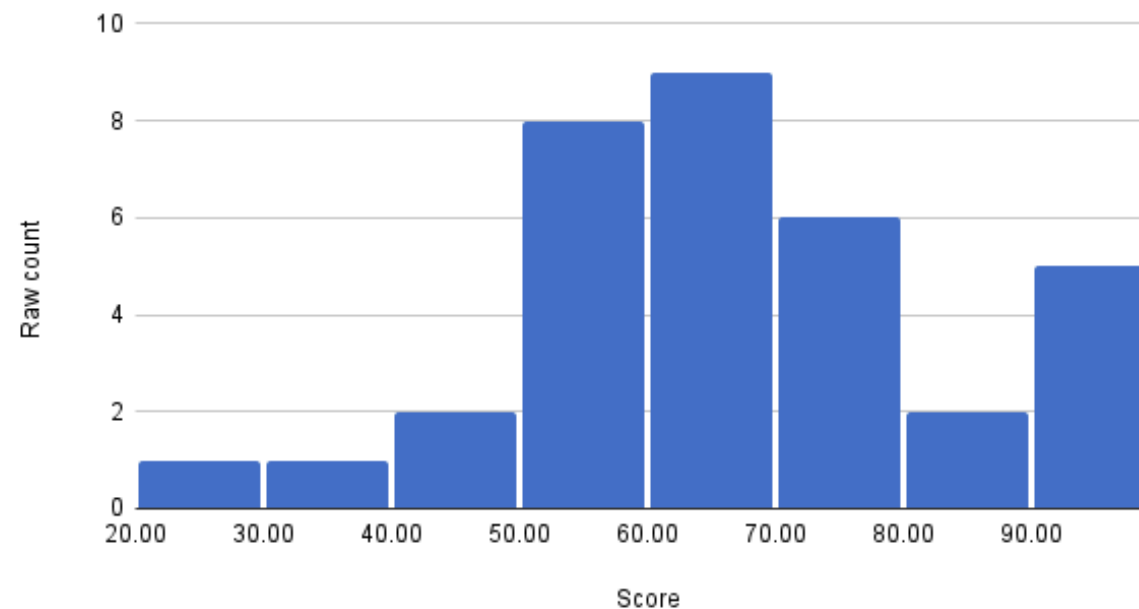
George Chen

Administrivia

In case you missed the Canvas announcement:

- Lectures this week are remote-only as I'm at CMU Australia, teaching one of my other classes in-person
- The quiz has been graded
(mean: 66.4, std dev: 17, max achieved 96)

94-775 Spring 2022 Quiz Score Histogram



- See Canvas announcement for regrade request procedure
(note that these are due at the end of today Pittsburgh time)

HW1/HW2 Questionnaire Preface

- Unusual semester: much higher enrollment than usual (my 95-865 course is way overbooked and on “in-person rotation” since everyone can’t fit) — this mini I’m teaching 200+ students
 - Unfortunately this has meant that my email volume is also much higher than normal so my response times are slower =(
 - I’m occasionally still getting emails about debugging

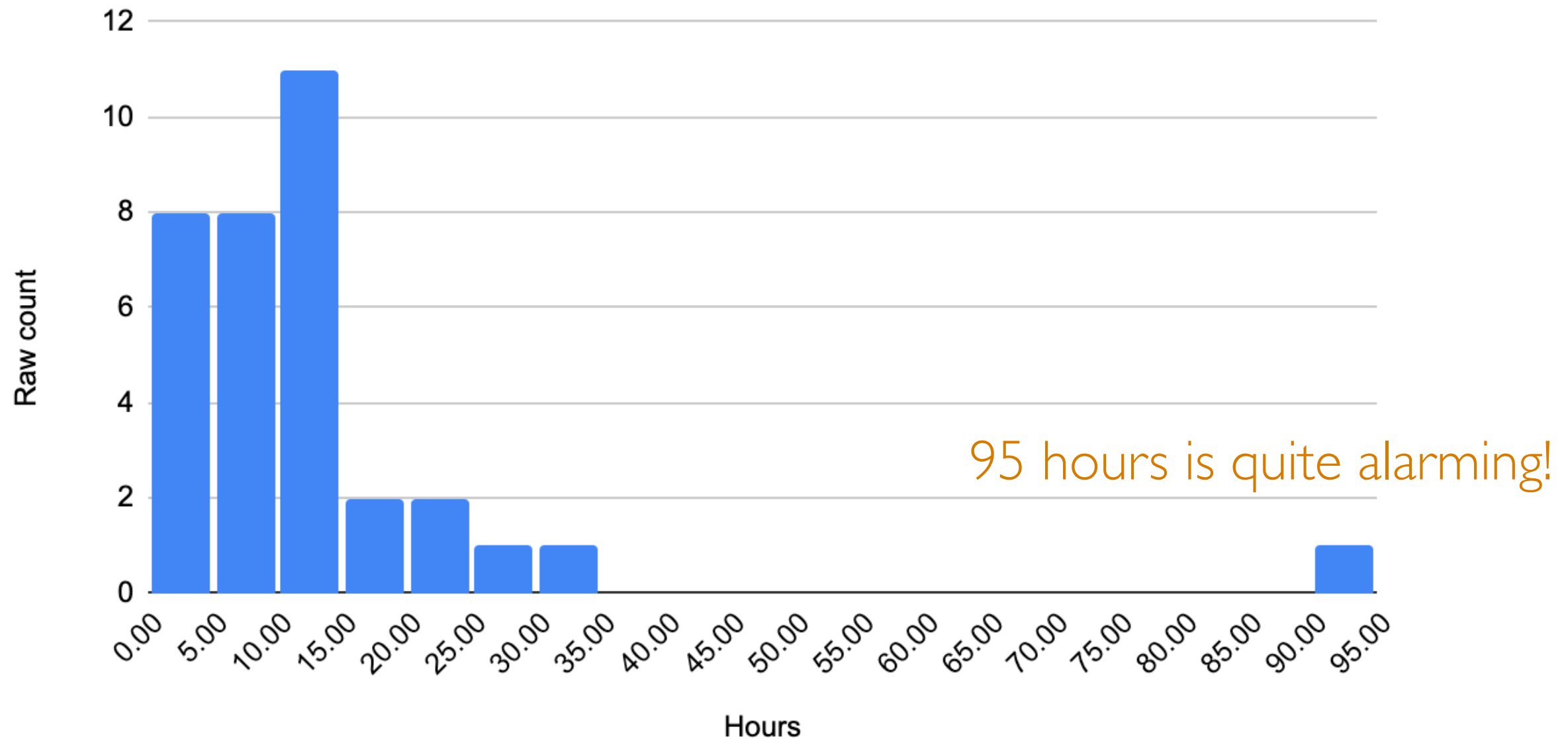
Please ask these questions in office hours instead
(learning to debug is better when it’s interactive rather than over email)

- Student feedback is very important to me and has helped me improve 94-775 a lot over the years

Mid way through the mini, I was told that extremely likely what will happen is that a new faculty member will take over teaching 94-775 (instead I’m being asked to teach even more sections of 95-865)

HWI Questionnaire

How many hours did you take (roughly) to complete homework 1?

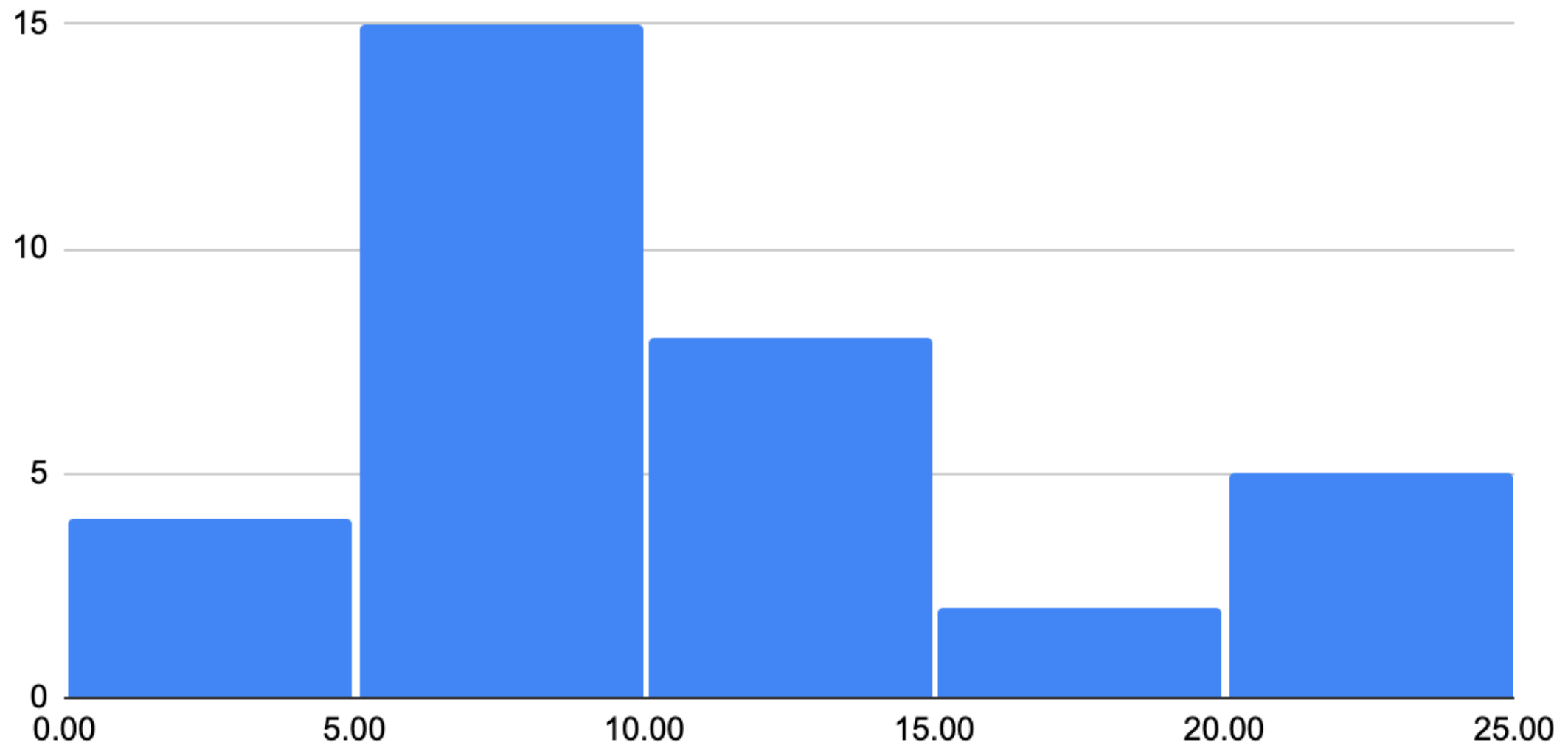


Homework is designed to take upwards of ~15 hours

Most students are finishing in the amount of time that we expect

HW2 Questionnaire

How many hours did you take (roughly) to complete homework 2?



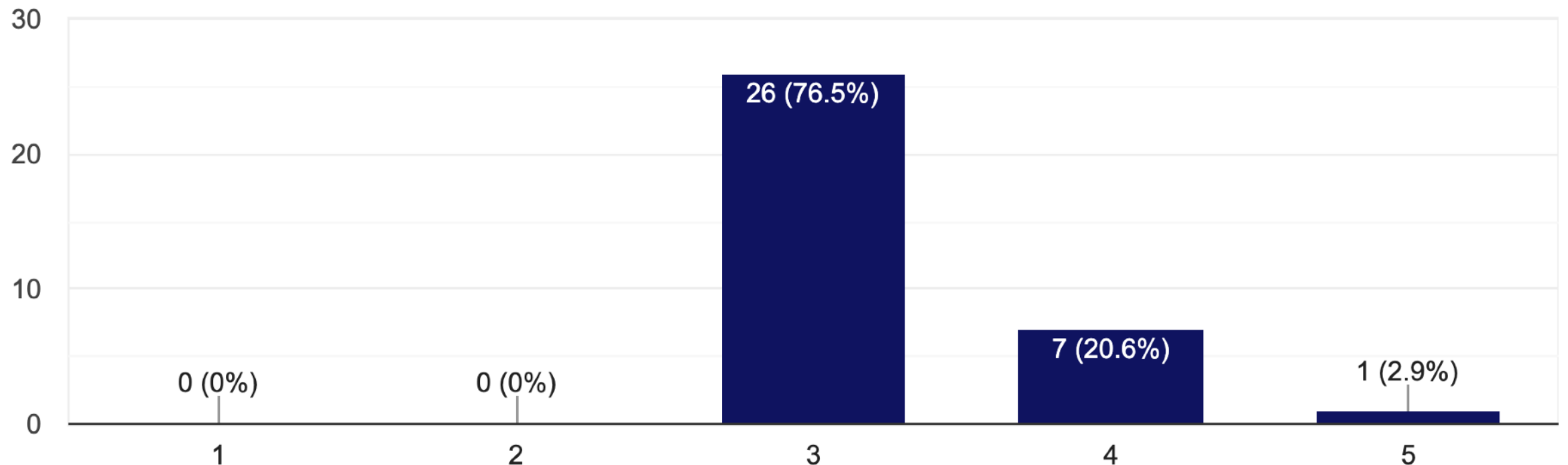
How many hours did you take (roughly) to complete homework 2?

The amount of time taken for HW2 looks more reasonable!

HWI Questionnaire

How do you find the lecture pace?

34 responses



- Oh no! This is actually *unusual* for 94-775 — historically this plot was more symmetric...
- In the free response text, several students also complained about the lectures being too fast, especially the demo code

Questionnaire Overall Thoughts

- Coding:
 - If you find that you're spending a lot more time coding, it could be helpful swinging by office hours to discuss *debugging strategies*
 - Often this makes a huge difference in how much time students spend on coding assignments is knowing how to identify where errors are
 - Unfortunately this course has a wider range of backgrounds in terms of coding and it's difficult for me to have a pace that works for absolutely everyone =(
- State-of-the-art approaches to unstructured data analysis heavily use neural nets and deep learning, and we'll only be catching a glimpse of this
 - I did get feedback where students wanted to see more of this; I'll present as much as I can — the coding is *more challenging and not required for 94-775...*

How to choose the number of topics k ?

Look at *within topic variability* and *between topic variability*

Within Topic Variability

Let's look at top-20 word lists (the ones from the demo)

```
[Topic 0]
good : 0.01592254908129389
like : 0.01584763067117222
just : 0.015714974597809874
think : 0.014658035148150044
don : 0.01336602502776772
time : 0.012159230893303024
year : 0.011442050656933937
new : 0.008768217977593912
years : 0.00843922077825026
game : 0.008416482579473757
make : 0.008318270139852606
ve : 0.00805613381872604
know : 0.00786552901690738
going : 0.007357414502894818
better : 0.007305177940555176
really : 0.007282768897233162
got : 0.007100242166187475
way : 0.007020258221618519
team : 0.006901091494924322
car : 0.006860678090522195
```

```
[Topic 1]
drive : 0.025114459755967225
card : 0.01904504522714293
scsi : 0.01574807346309645
disk : 0.015086151949241311
use : 0.01311205775591249
output : 0.012487568705565076
file : 0.011474974819227298
bit : 0.011450491727323115
hard : 0.010426435918865882
entry : 0.009962381704950415
memory : 0.009892938703385204
mac : 0.009531449582947765
video : 0.009451338641933656
drives : 0.00907129942775757
pc : 0.0090703286112168
windows : 0.008135023842197355
16 : 0.00796825814975238
bus : 0.007927283819698584
control : 0.007892257876122581
program : 0.00784268458596016
```

```
[Topic 2]
10 : 0.0320291203
00 : 0.0269643305
25 : 0.0218296912
15 : 0.0206063577
11 : 0.0206043503
20 : 0.0204957609
12 : 0.0203766844
14 : 0.0180554708
16 : 0.0164602656
13 : 0.0160230124
17 : 0.0160189031
18 : 0.0159314160
30 : 0.0134871298
29 : 0.01291330831
24 : 0.0131269045
19 : 0.0125205615
55 : 0.0125002331
21 : 0.012264247
40 : 0.0119211525
22 : 0.0112072317
```

If we see the word “good”, how likely are we to see the word “years”?
 $P(\text{see word “years”} \mid \text{see word “good”})$

Focus on a single topic at a time

If this probability is high for every pair of words in the top-20 list, then in some sense the topic is more “coherent”

Between Topic Variability

Let's look at top-20 word lists (the ones from the demo)

[Topic 0]

good : 0.01592254908129389
like : 0.01584763067117222
just : 0.015714974597809874
think : 0.014658035148150044
don : 0.01336602502776772
time : 0.012159230893303024
year : 0.011442050656933937
new : 0.008768217977593912
years : 0.00843922077825026
game : 0.008416482579473757
make : 0.008318270139852606
ve : 0.00805613381872604
know : 0.00786552901690738
going : 0.007357414502894818
better : 0.007305177940555176
really : 0.007282768897233162
got : 0.007100242166187475
way : 0.007020258221618519
team : 0.006901091494924322
car : 0.006860678090522195

[Topic 1]

drive : 0.025114459755967225
card : 0.01904504522714293
scsi : 0.01574807346309645
disk : 0.015086151949241311
use : 0.01311205775591249
output : 0.012487568705565076
file : 0.011474974819227298
bit : 0.011450491727323115
hard : 0.010426435918865882
entry : 0.009962381704950415
memory : 0.009892936703385204
mac : 0.009531449582937765
video : 0.009451338641933656
drives : 0.009074000962777757
pc : 0.0090703286112168
windows : 0.008135023862197355
16 : 0.00798823814975238
bus : 0.007927283819698584
controller : 0.007902057876189581
program : 0.00784268458596016

[Topic 2]

10 : 0.0320292203
00 : 0.0269643305
25 : 0.0218296912
15 : 0.0206063577
11 : 0.0206043503
20 : 0.0204957609
12 : 0.0203766844
14 : 0.0180554708
16 : 0.0164602656
13 : 0.0160230124
17 : 0.0160189031
18 : 0.0159314160
30 : 0.0134871298
50 : 0.0133230831
24 : 0.0131269045
19 : 0.0125205615
55 : 0.0125002331
21 : 0.0122642479
40 : 0.0119281525
22 : 0.0112072317

→ If “good” only shows up in the top-20 word list for topic 0, then it is considered a **unique top word** for topic 0

Each topic has a **number of unique top words**

How to Choose Number of Topics k ?

For a specific topic, look at the m most probable words (“top words”)

Coherence (within topic variability):

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

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

avoid numerical issues

Number of unique words (between topic variability):

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

Can average each of these across the topics

Can plot average coherence vs k , and average # unique words vs k (for values of k you are willing to try)

Unlike for CH index, no clear way to trade off between avg. coherence and avg. # unique words (they aren't even in the same units!!!)

Topic Modeling: Last Remarks

- There are actually *many* topic models, not just LDA
 - Hierarchical Dirichlet Process, correlated topic models, SAGE, anchor word topic models, ProdLDA, embedded topic model, ...
- Dynamic topic models: track how topics change *over time*
- Trivial to add supervision to topic models! Can have topics learned help with prediction tasks!
- Reminder: learning topic models can be very sensitive to random initialization

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

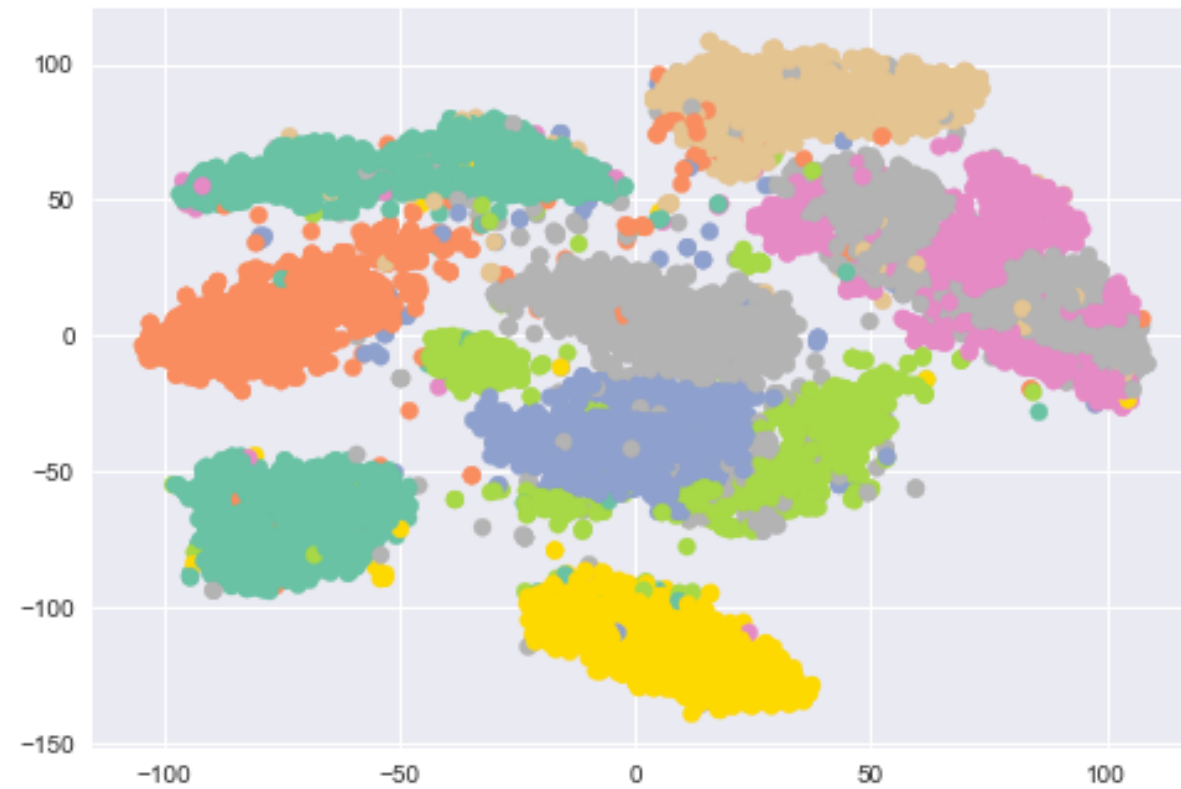
Part II: Predictive data analysis

Make predictions using known structure in data

- Basic concepts and how to assess quality of prediction models
- Neural nets and deep learning for analyzing images and text

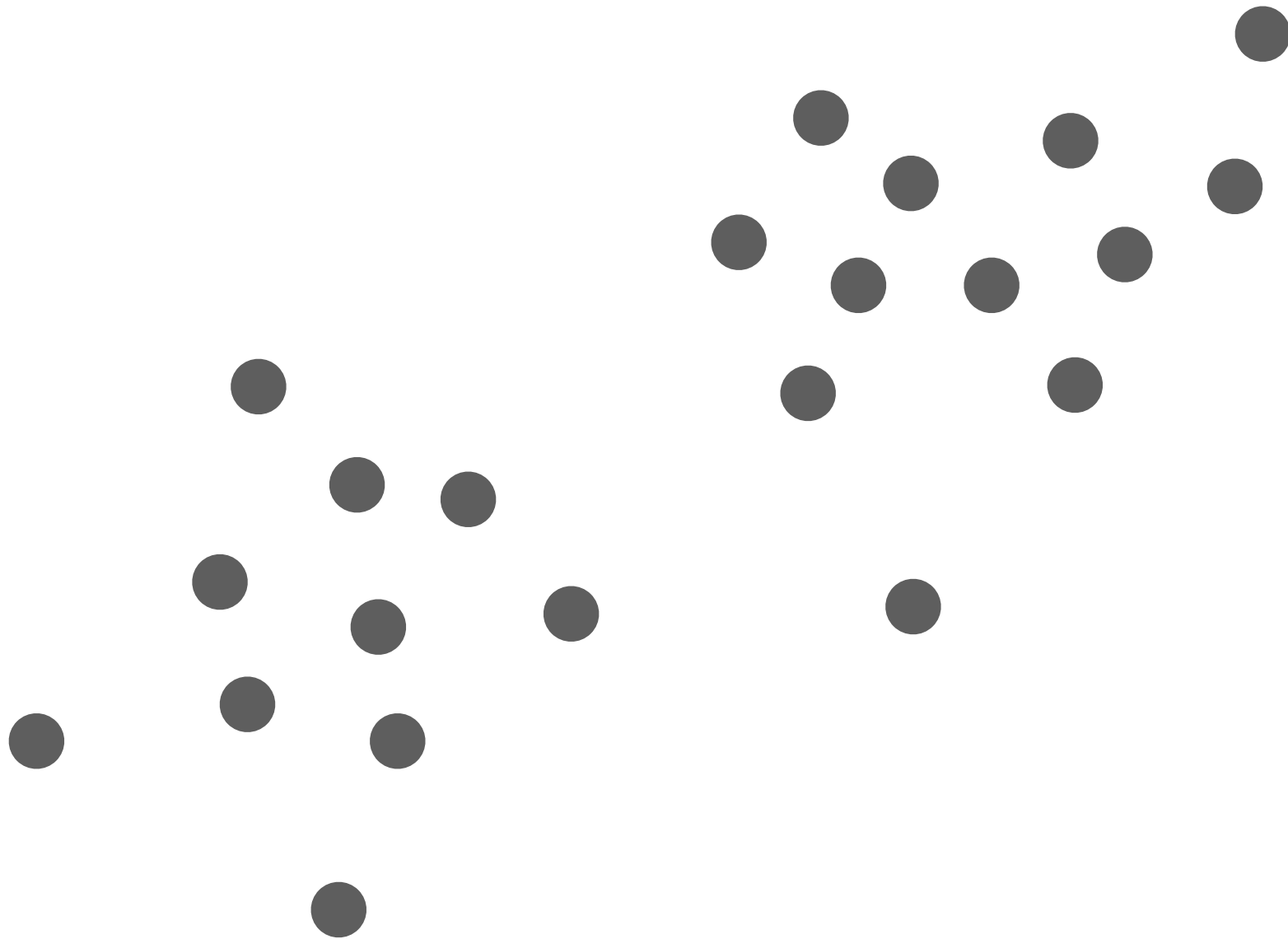
What if we have labels?

Disclaimer: unfortunately “*k*”
means many things



Example: MNIST handwritten digits have known labels

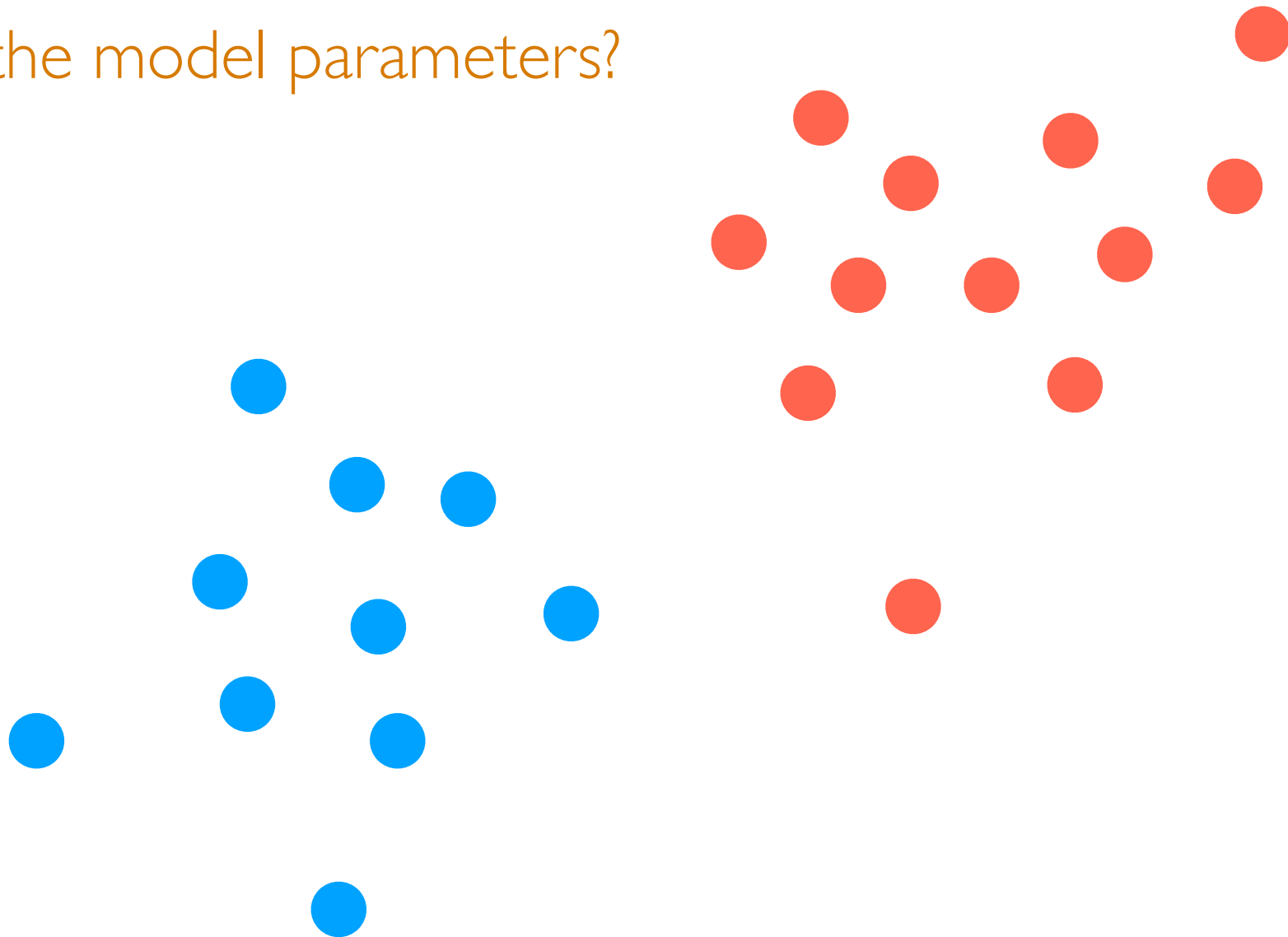
If the labels are known...



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

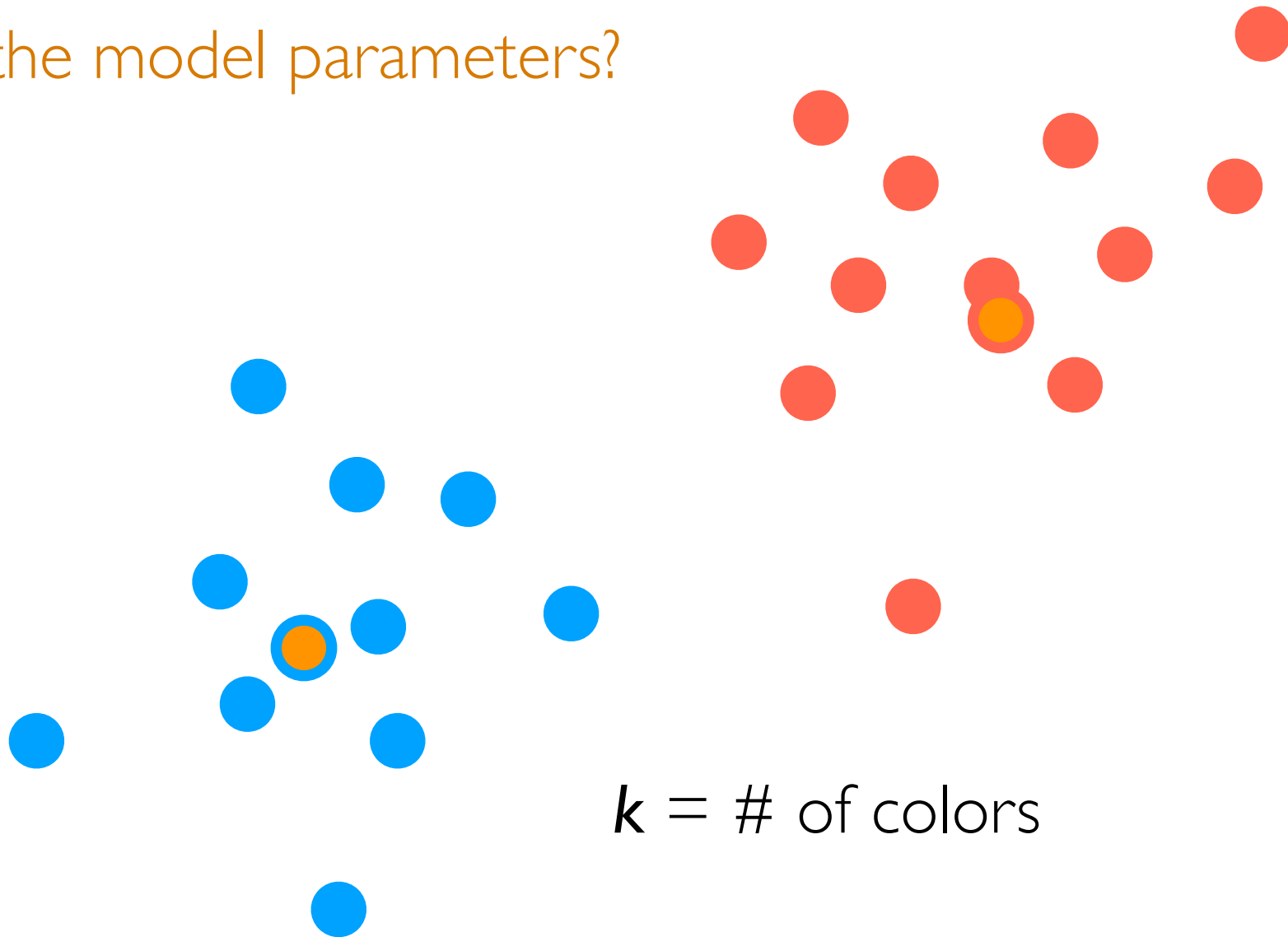
Step 3: Update **cluster probabilities, means, and covariances** carefully accounting for probabilities of each point belonging to each of the clusters

We don't need to repeat until convergence

If the labels are known...

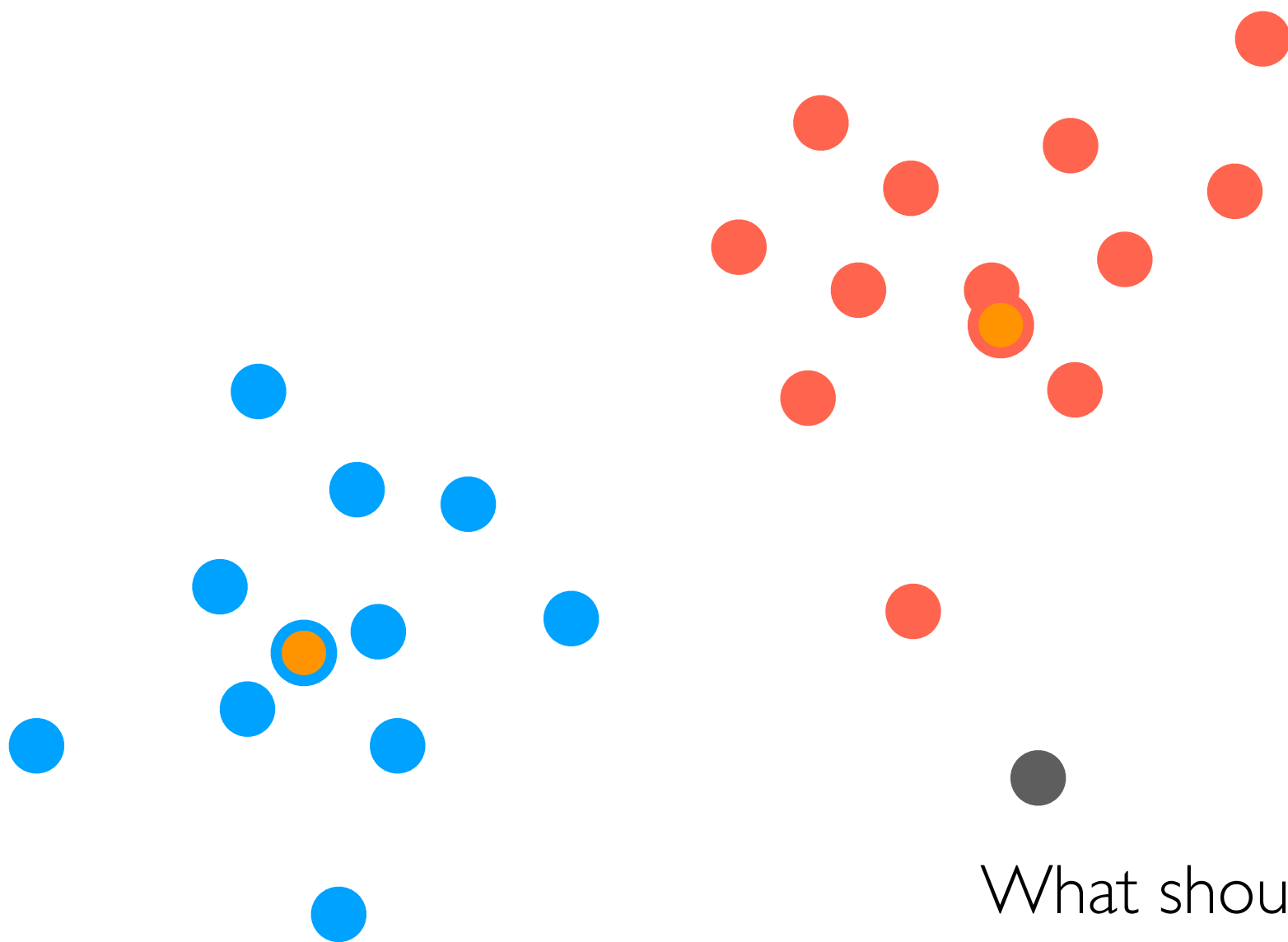
And we assume data generated by GMM...

What are the model parameters?



$k = \#$ of colors

We can directly estimate
cluster means, covariances

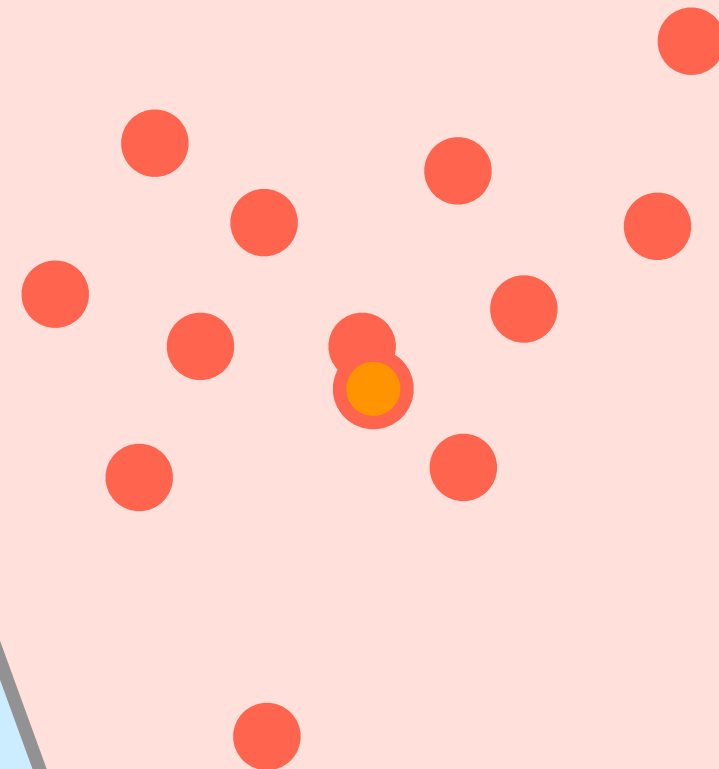
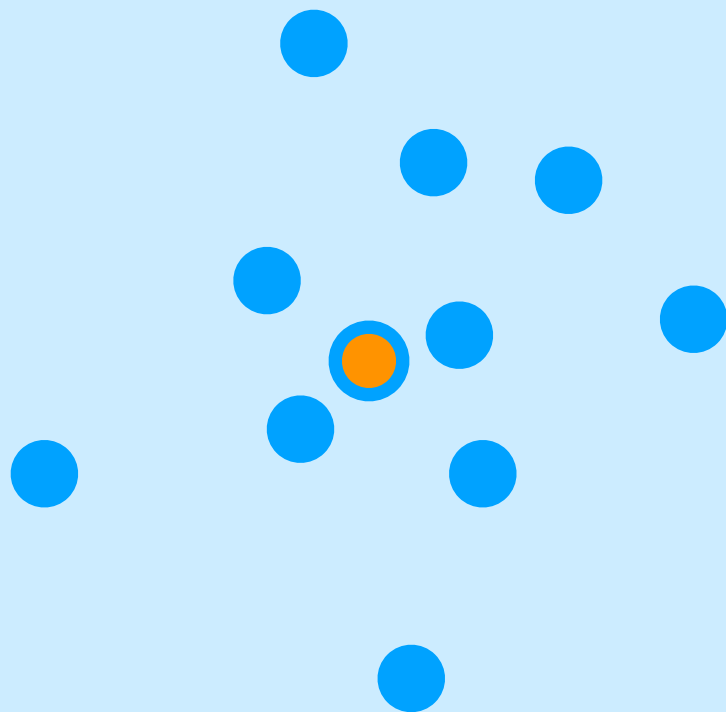


What should the label of this new “test” point be?

Whichever cluster has higher probability!

Decision boundary

We just created a **classifier**
(a procedure that given a test data point
tells us what “class” it belongs to)



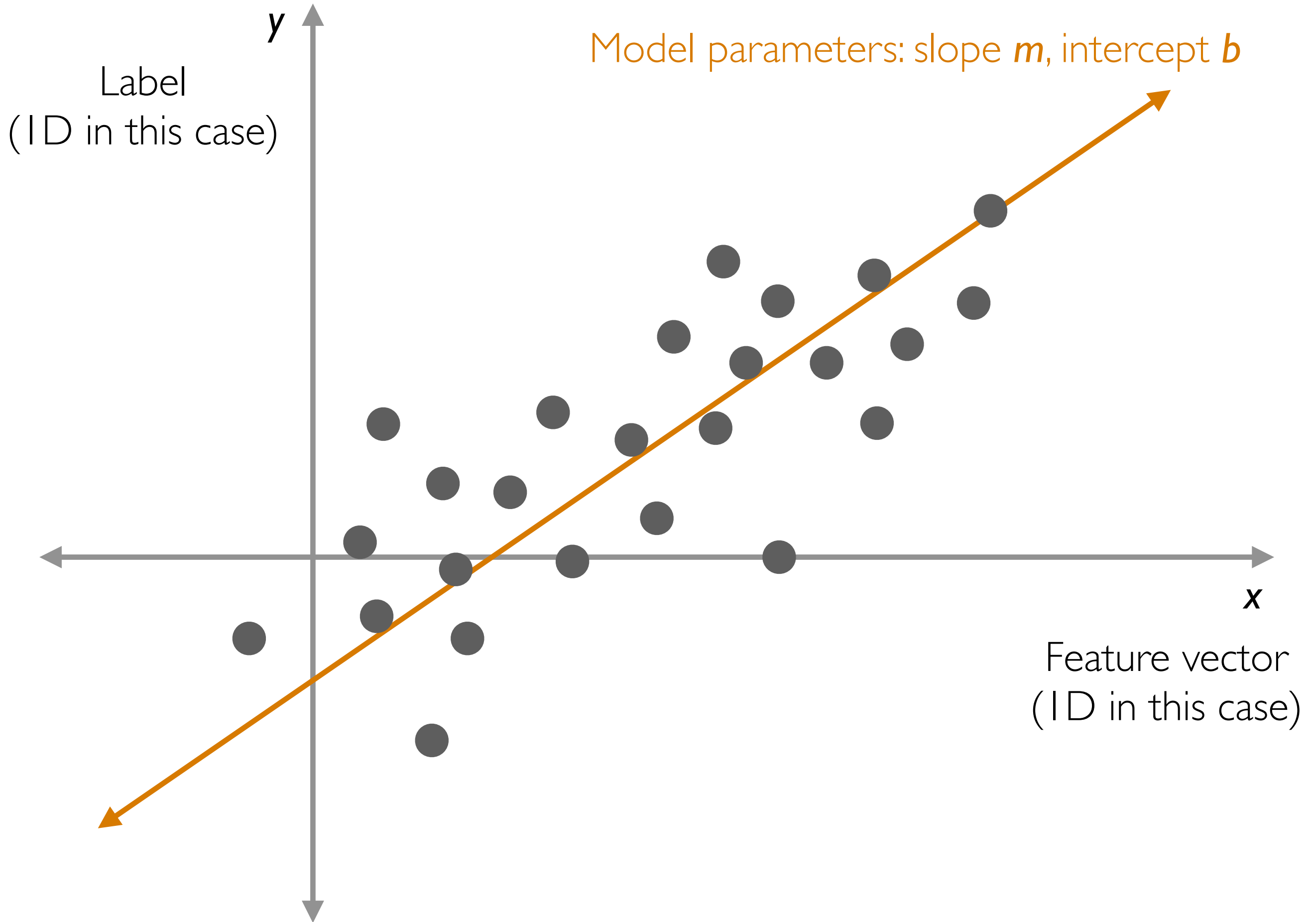
What should the label of
this new “test” point be?

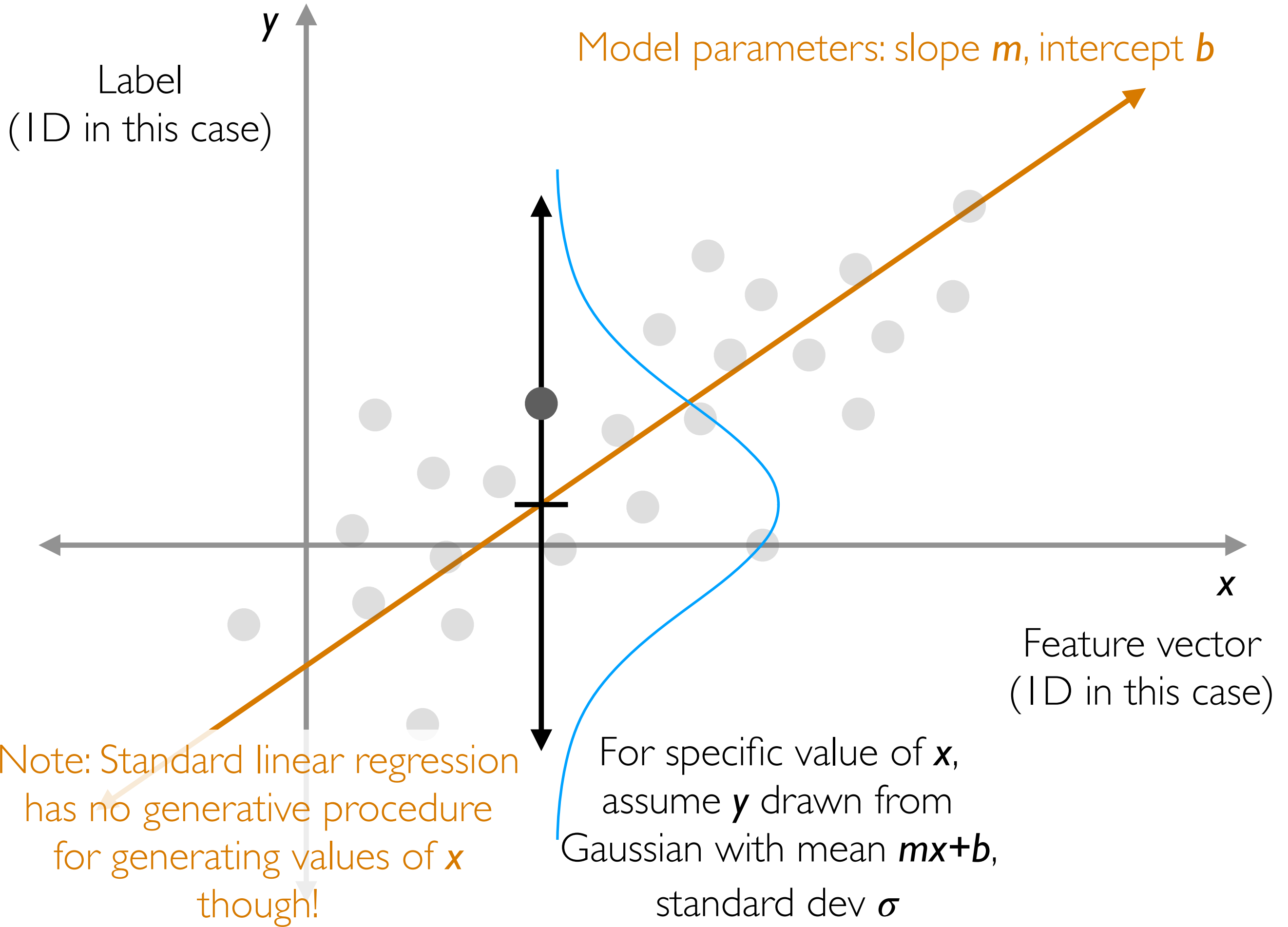
Whichever cluster has
higher probability!

This classifier we’ve created assumes a
generative model

**You've seen a prediction model that
is partly a generative model**

Linear regression!





Predictive Data Analysis

Training data

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

Goal: Given new test feature vector x , predict label y

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

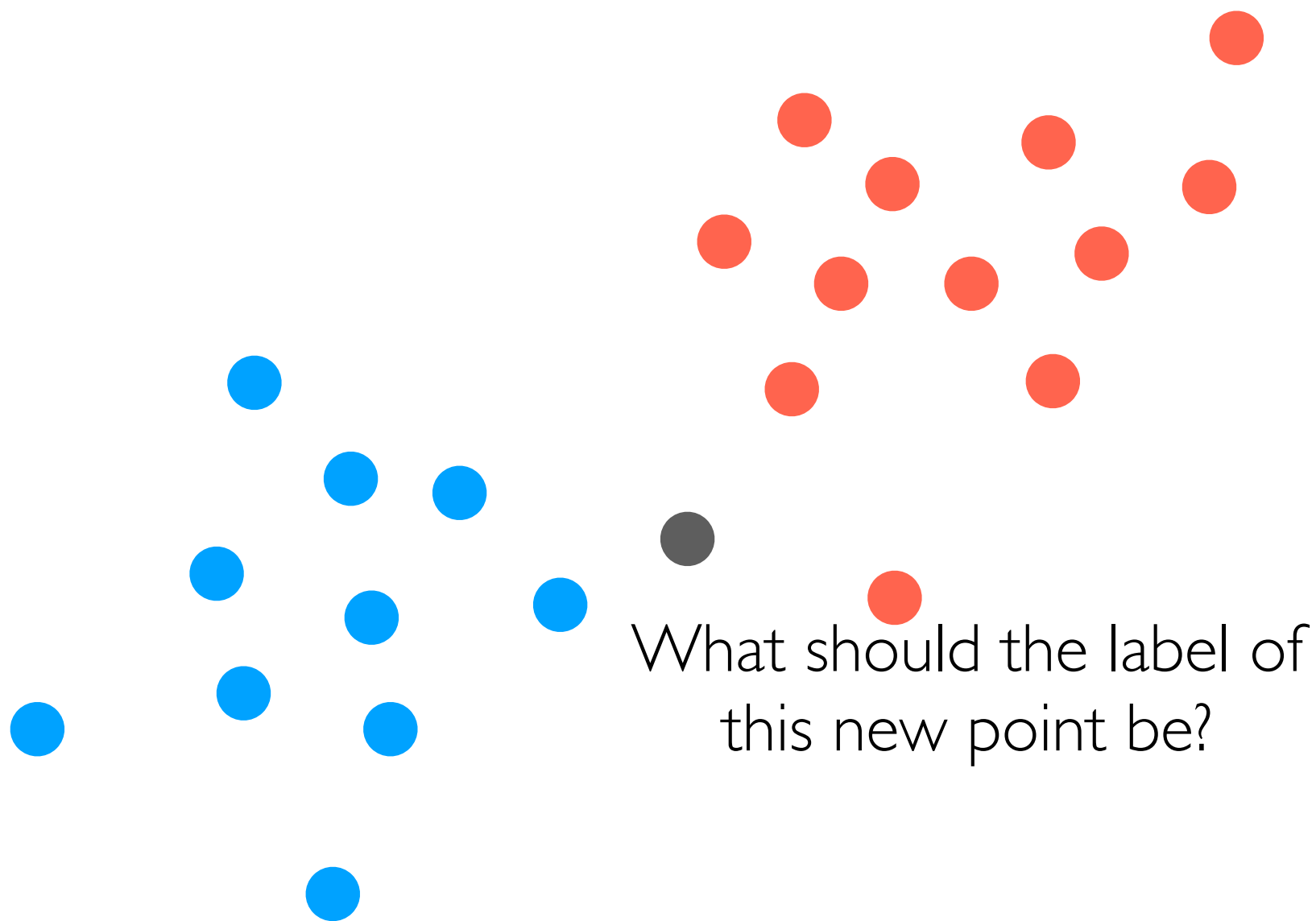
We could have many such test feature vectors, which we collectively refer to as *test data*

A giant zoo of methods

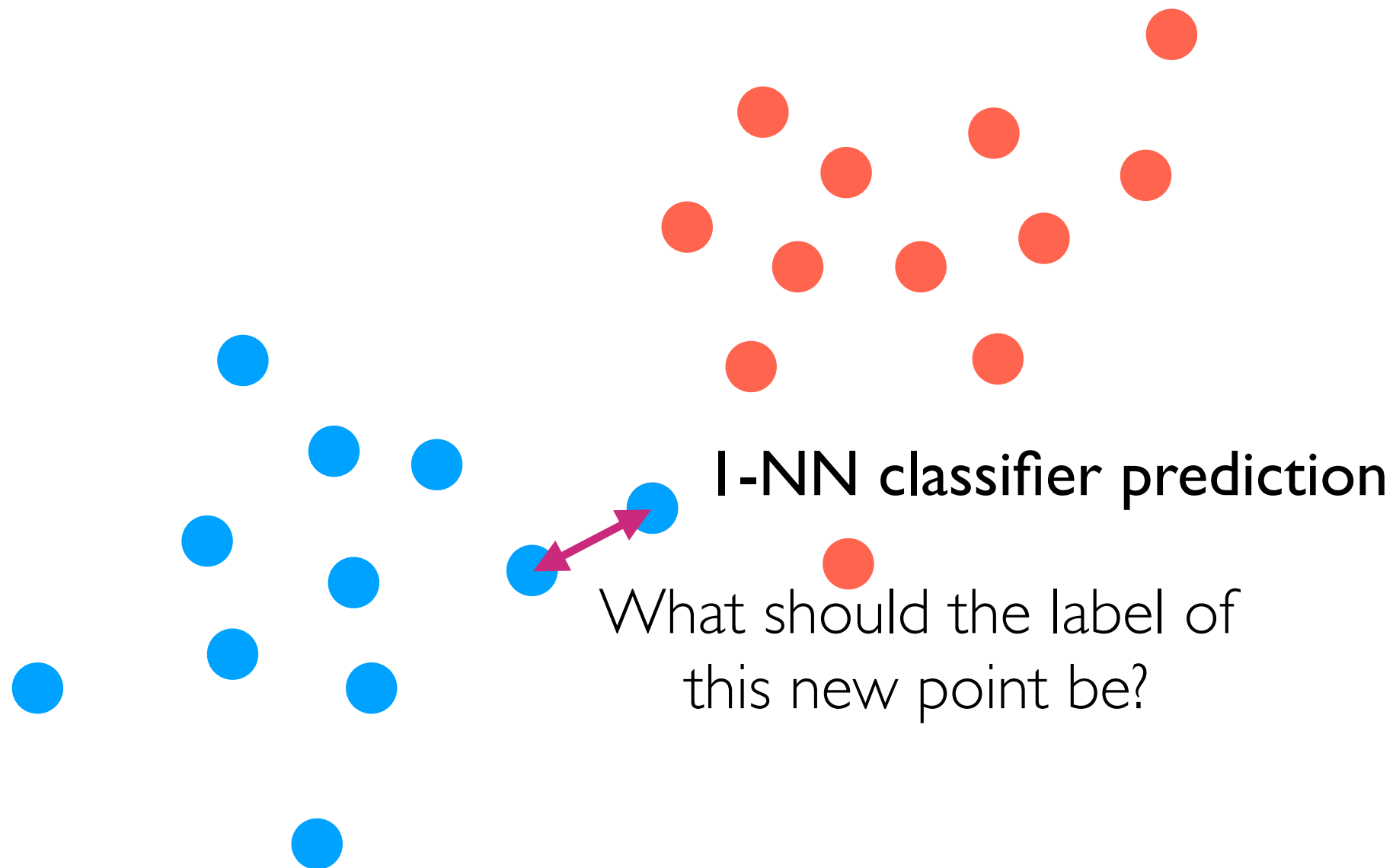
- Generative models (like what we just described)
- Discriminative models (just care about learning prediction rule; after training model, we don't have a way to generate data)

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

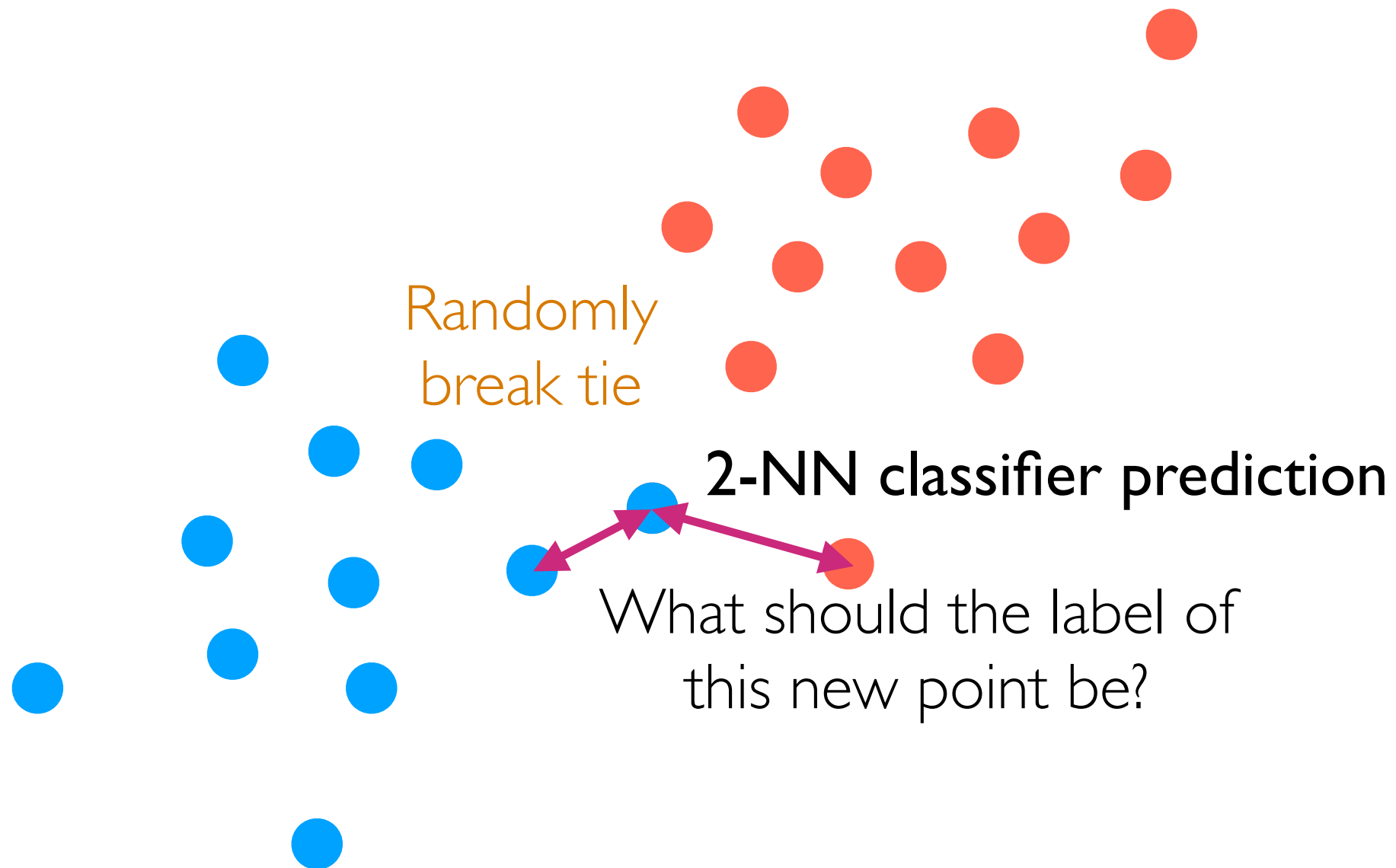
Example: k -NN Classification



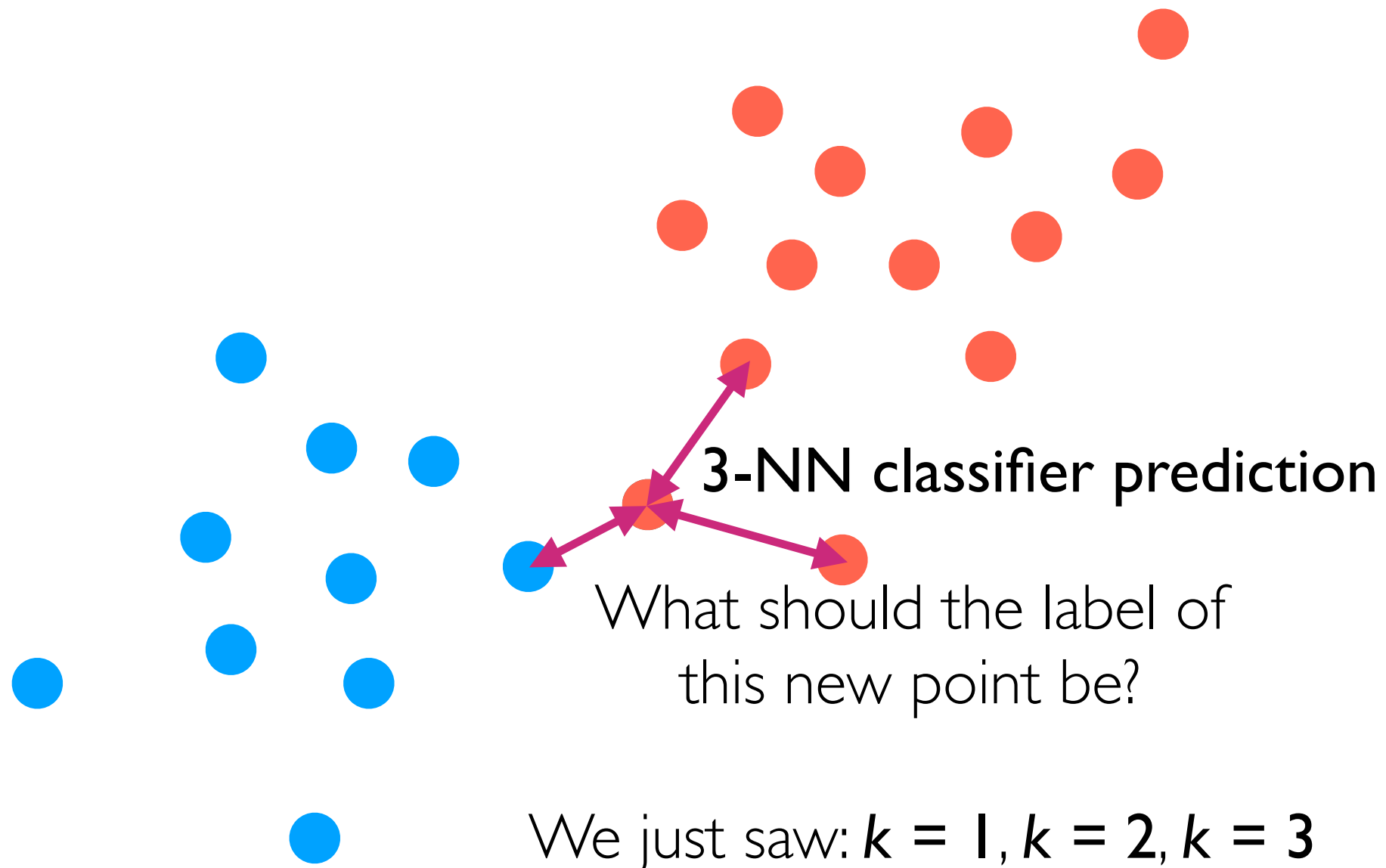
Example: k -NN Classification



Example: k -NN Classification



Example: k -NN Classification



What 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

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

Hyperparameters vs. Parameters

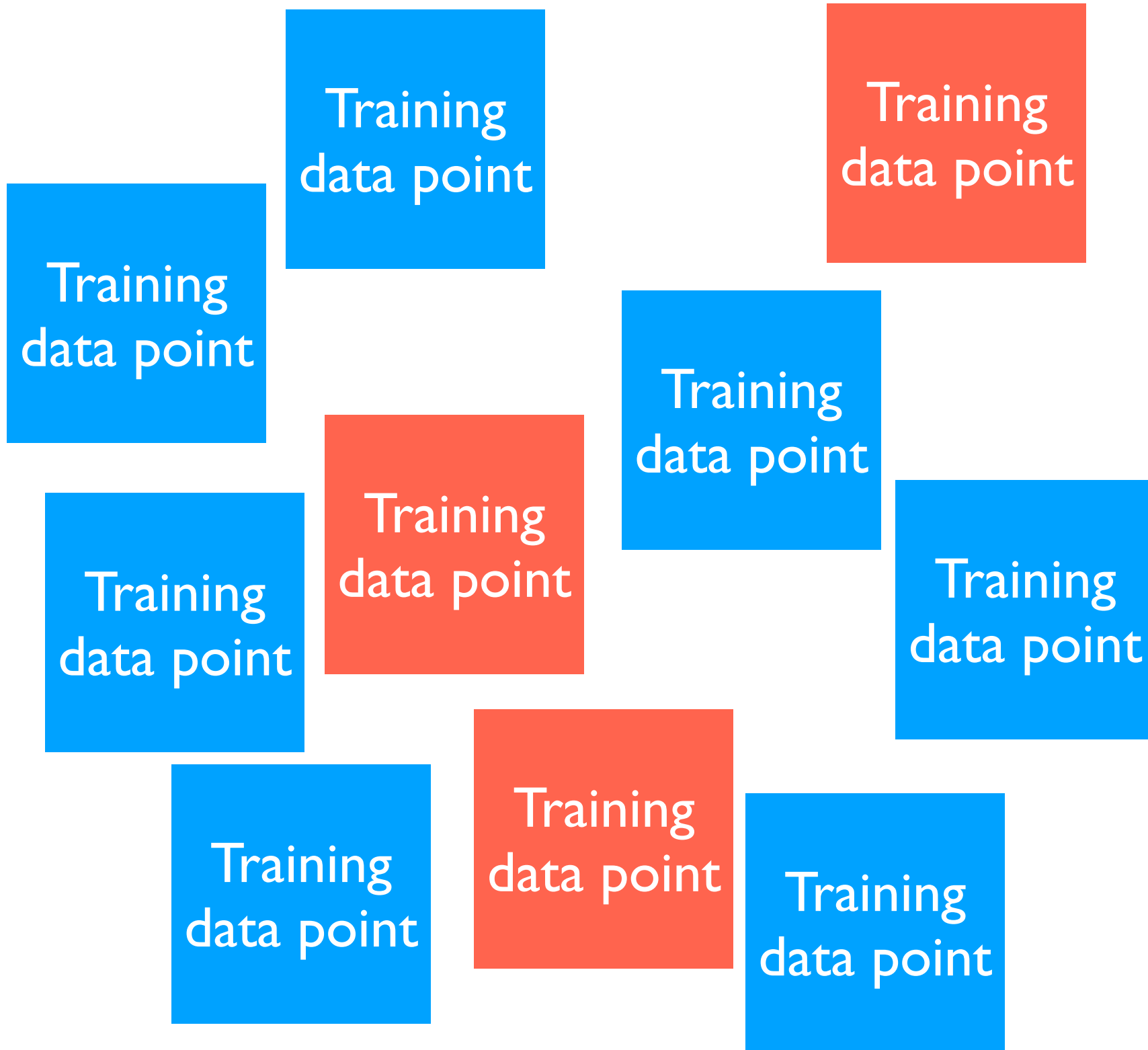
- We fit a model's parameters to training data (terminology: we “learn” the parameters)
- We pick values of hyperparameters and they do *not* get fit to training data
- Example: Gaussian mixture model
 - Hyperparameter: number of clusters k
 - Parameters: cluster probabilities, means, covariances
- Example: k -NN classification
 - Hyperparameter: number of nearest neighbors k
 - Parameters: N/A

⚠ Major assumption:
the training and test data “look similar”
(technically: training and test data are i.i.d.
sampled from the same underlying distribution)

In other words, we assume that there is an *unknown* generative process that produces every pair (x_i, y_i) from the exact same distribution

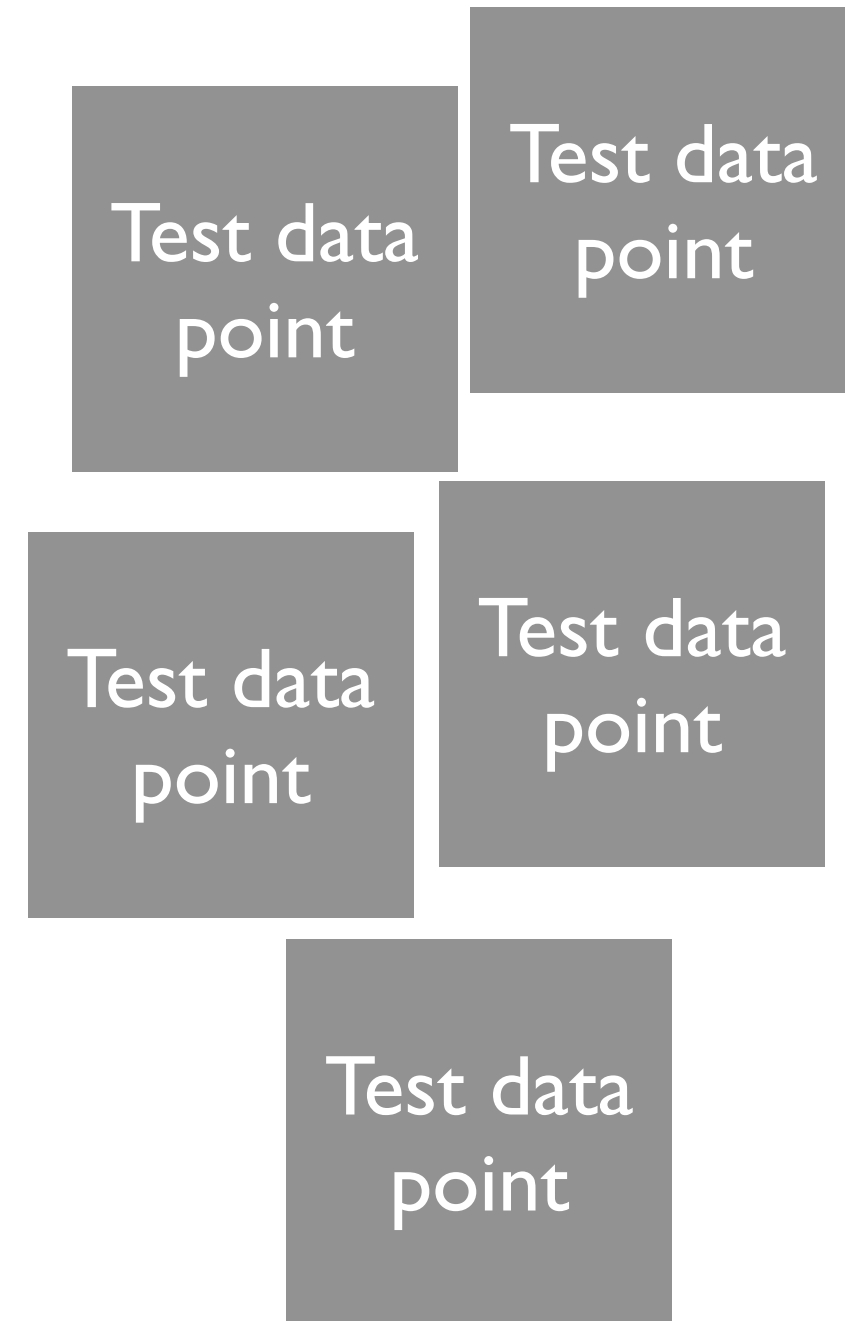
Prediction becomes harder when training and test data appear quite different!

Training data



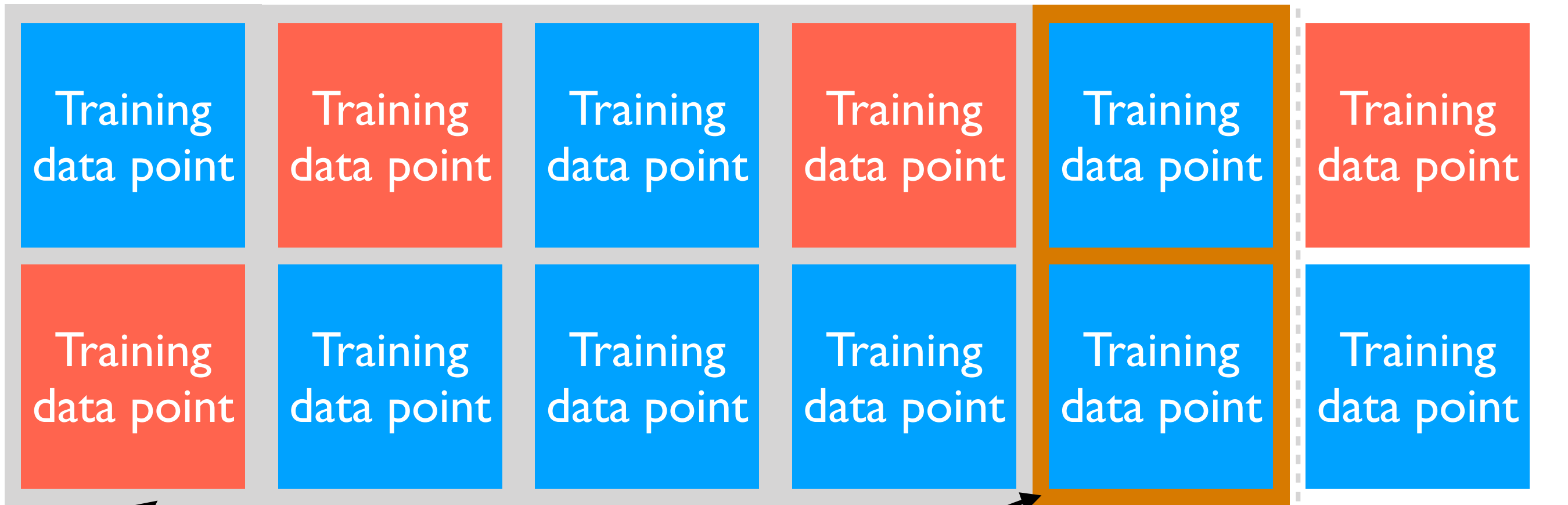
Example: Each data point is an email and we know whether it is spam/ham

Want to classify these points correctly



Example: future emails to classify as spam/ham

(this shuffling makes sense since we assume data are i.i.d.)



Train method on data in gray

Predict on data in orange

Terminology for this class:
“**Proper training data**”
(the gray box)

“**Validation data**”
(the orange box)

Compute prediction error

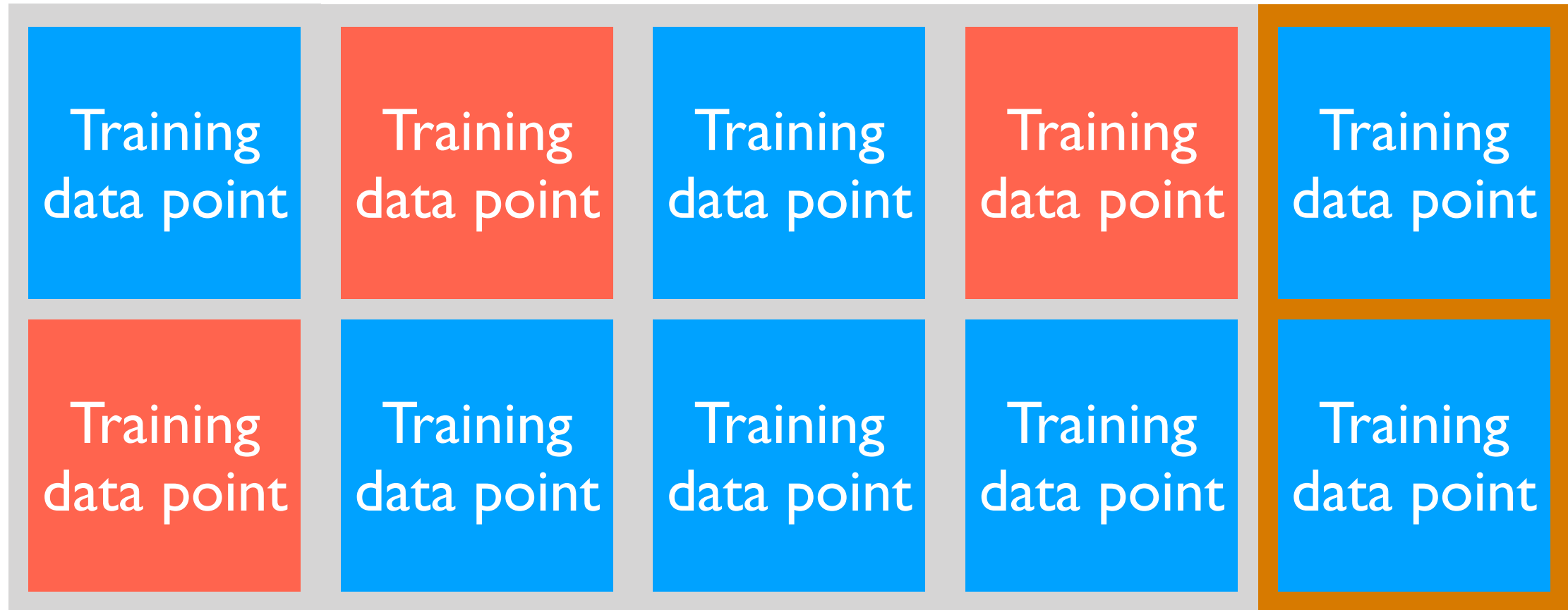
This is called data splitting/“train-validation split”

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

50%

Some people, including sklearn, call this “train-test split” but in this class, we will use “test data” to refer to true test data that the training procedure does not see

But we could have chosen different proper training/validation data!



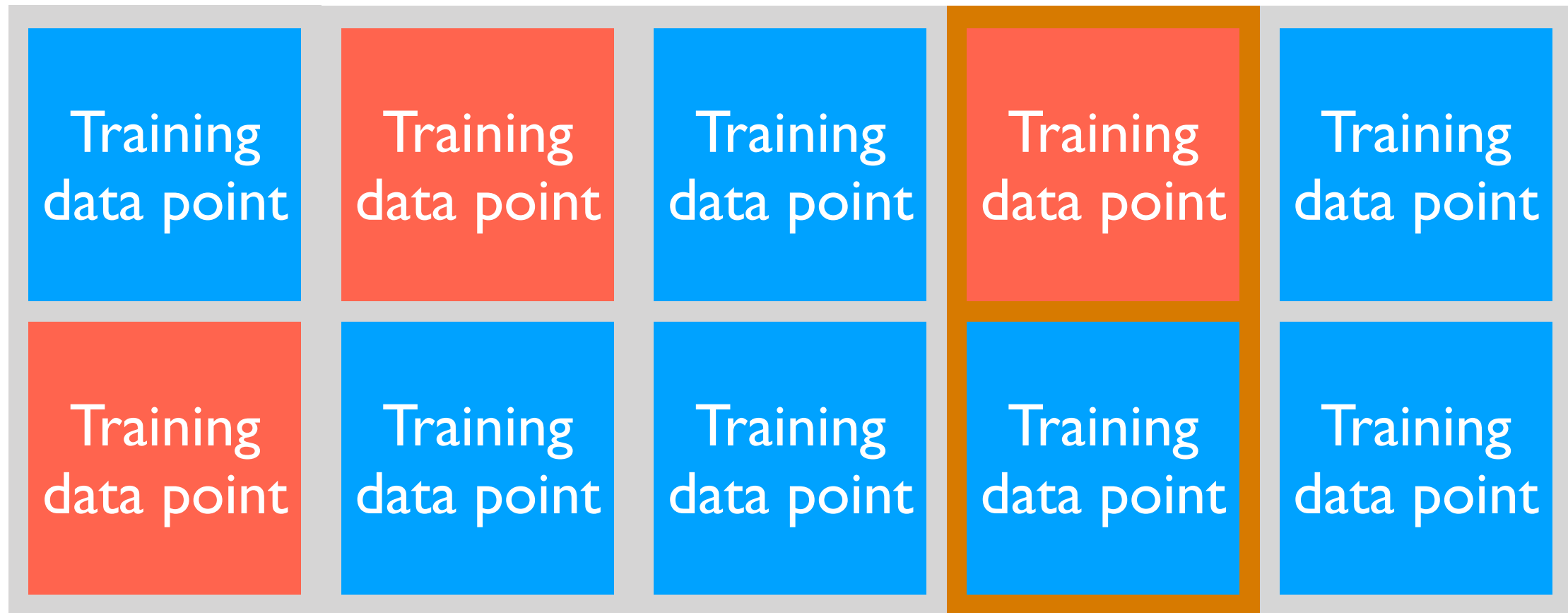
Train method on data in gray

Predict on data in orange

Compute prediction error

50%

But we could have chosen different proper training/validation data!



Train method on data in gray

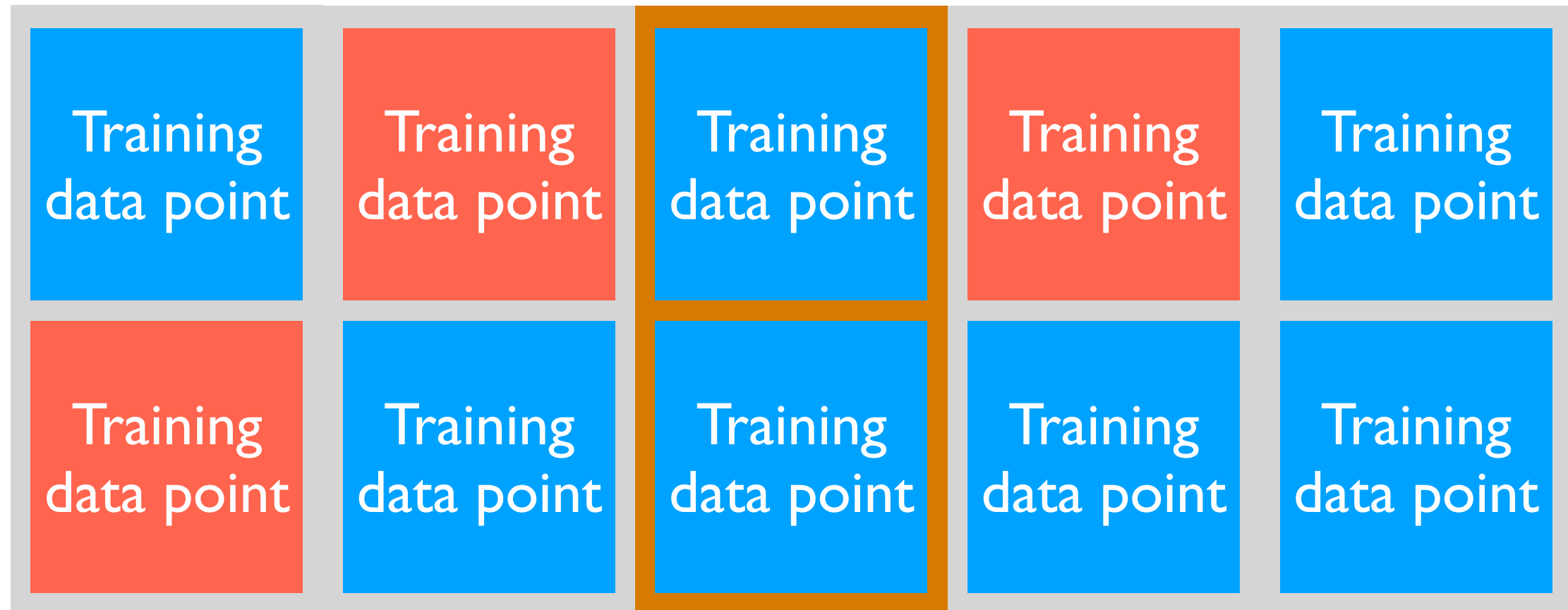
Predict on data in orange

Compute prediction error

0%

50%

But we could have chosen different proper training/validation data!



Train method on data in gray

Predict on data in orange

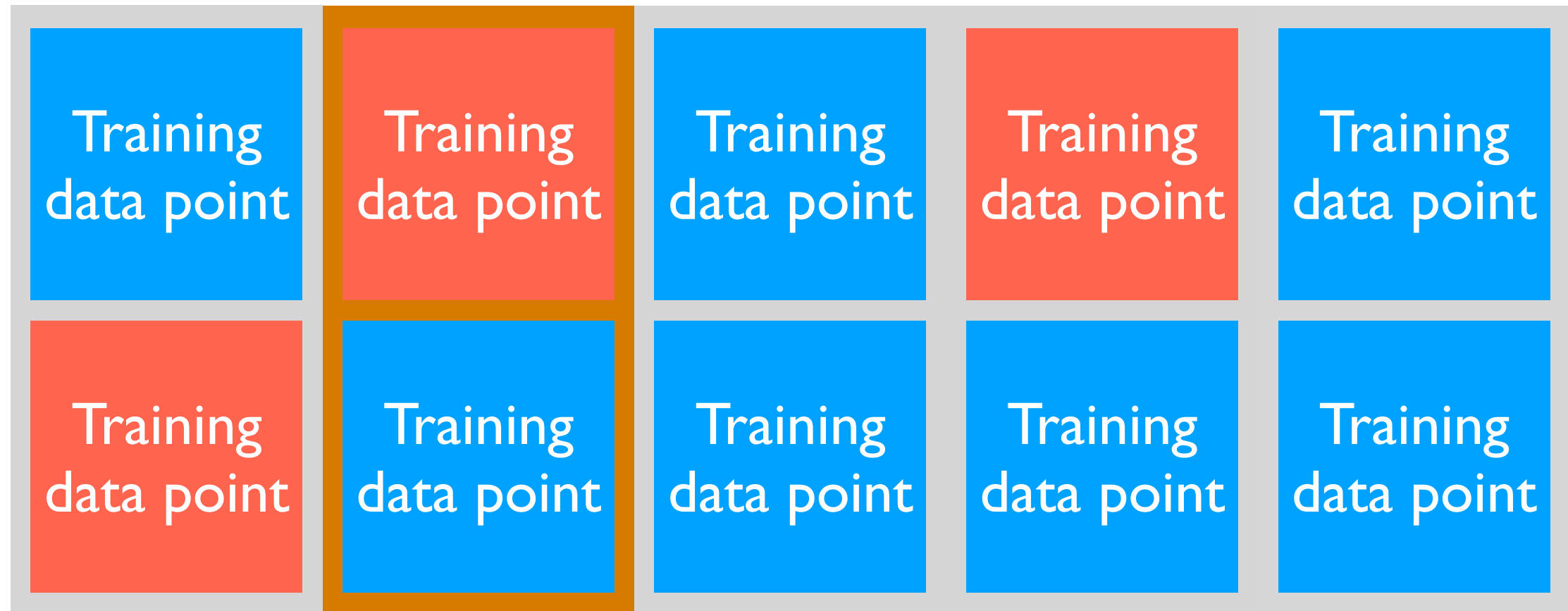
Compute prediction error

50%

0%

50%

But we could have chosen different proper training/validation data!



Train method on data in gray

Predict on data in orange

Compute prediction error

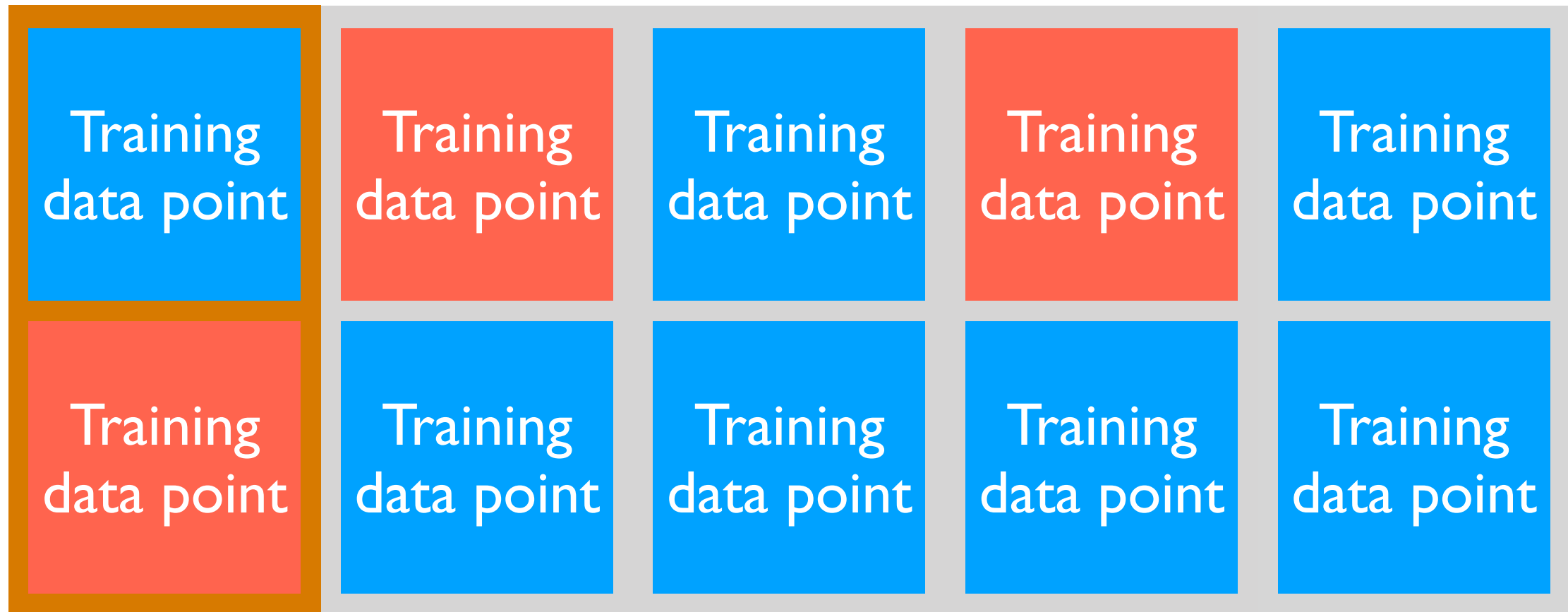
0%

50%

0%

50%

But we could have chosen different proper training/validation data!



Train method on data in gray

Predict on data in orange

We get 5 different prediction errors...
which is more accurate? 🤔

Compute prediction error

0%

0%

50%

0%

50%

Unclear which is best, so let's just average: $(0+0+50+0+50)/5 = 20\%$