Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

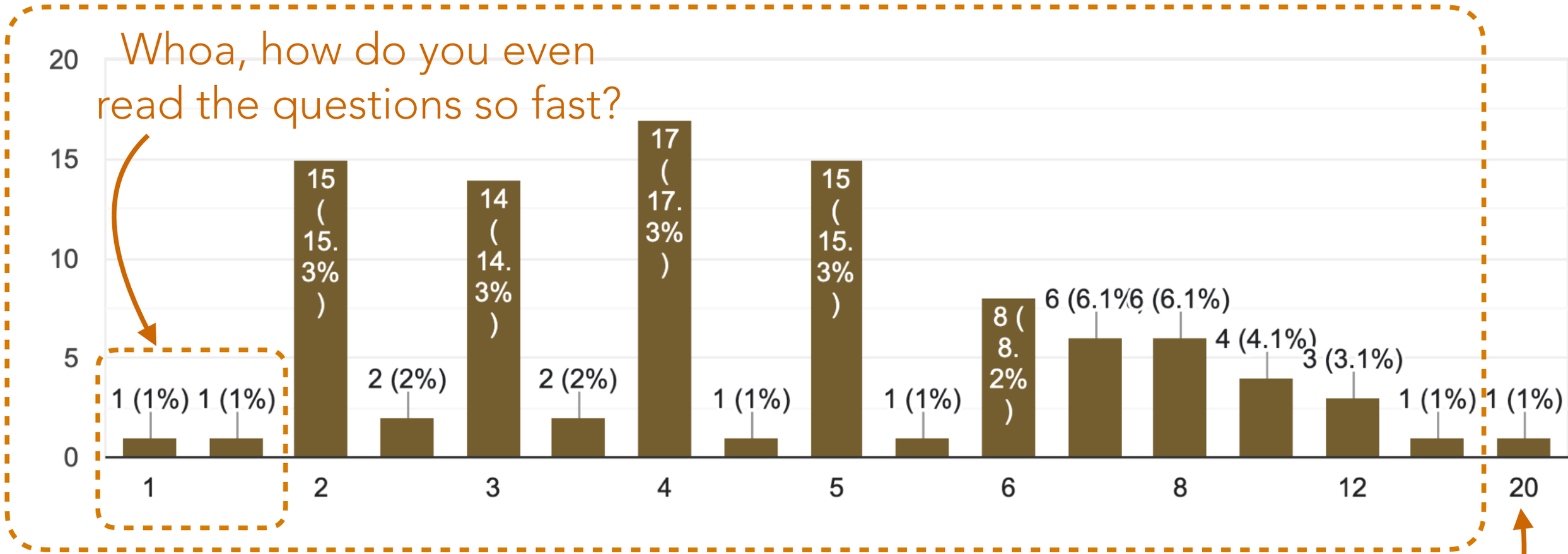
Lecture 10: Wrap up topic modeling, intro to predictive data analytics

Slides by George H. Chen

HW1 Questionnaire (1/5)

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

98 responses



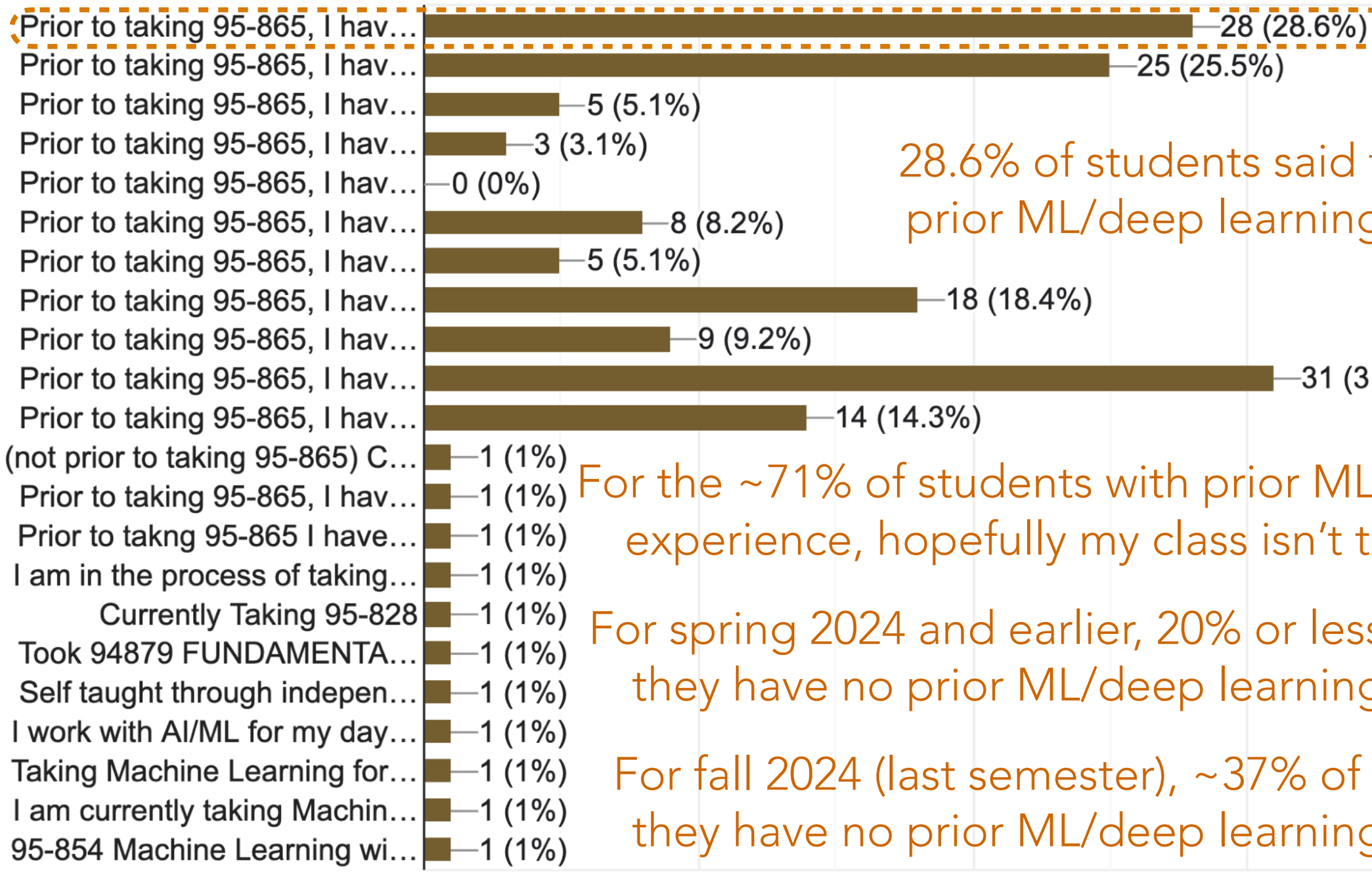
The vast majority of students are finishing within 15 hours
(which is our target of the maximum time that should be spent)

Spending over 15 hours on each homework is not normal!
Please do swing by OH to chat if you're still spending a lot of time on HW2

HW1 Questionnaire Results (2/5)

We have been trying to figure out to what extent students come into the class already having data analysis/machine learning knowledge. Please select... classes you are taking concurrently with 95-865.

98 responses



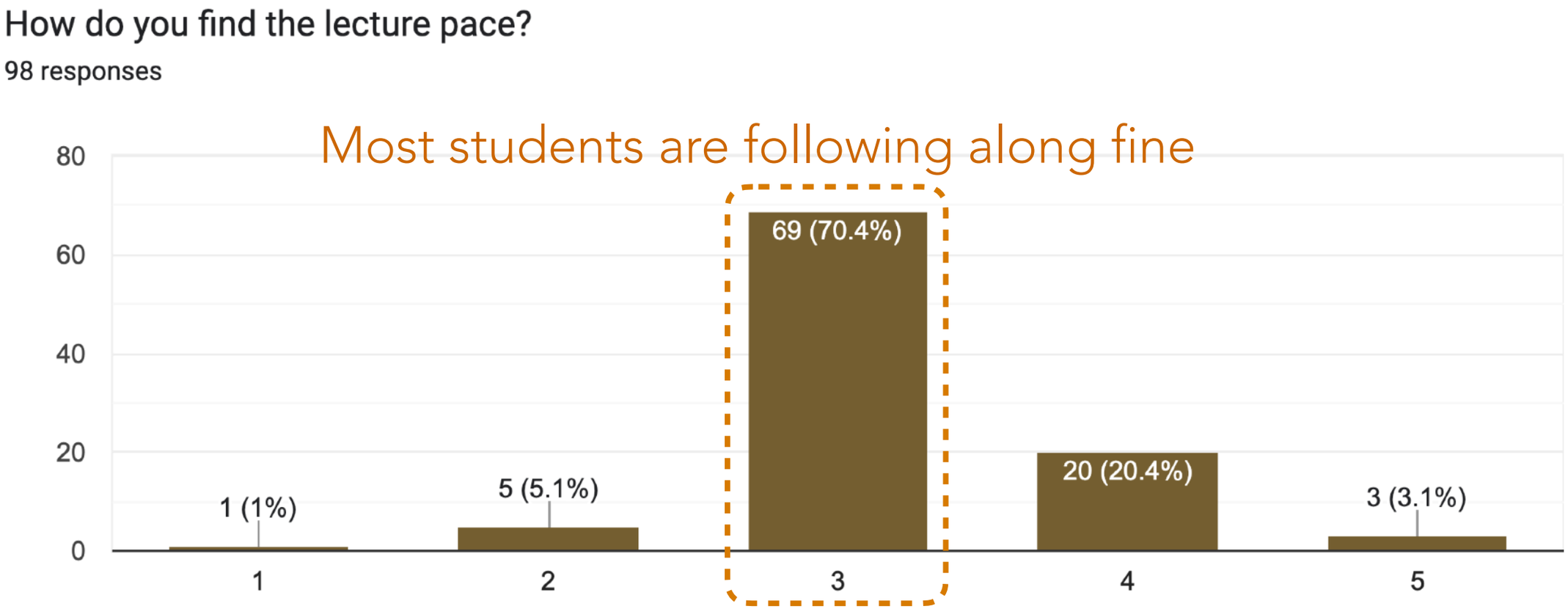
28.6% of students said they have no prior ML/deep learning experience

For the ~71% of students with prior ML/deep learning experience, hopefully my class isn't too boring 😅

For spring 2024 and earlier, 20% or less students said they have no prior ML/deep learning experience

For fall 2024 (last semester), ~37% of students said they have no prior ML/deep learning experience

HW1 Questionnaire (3/5)



Very importantly, I think it's okay to feel that the material takes time to digest & that it doesn't somehow just immediately all makes sense

It's like how when you watch some movies, the first time you see it, it doesn't fully make sense yet but watching a second time helps to clarify

(This is why we have lecture recordings!)

Overall, I'll try to keep the pacing about the same or slightly slower

HW1 Questionnaire (4/5)

Free response parts: It's really hard to satisfy everyone's requests!

- Some students wrote in the free response part that lectures are too fast
- Some students wrote in the free response part that lectures could be faster and to cover less coding in lecture
- Many students asked for more coding (e.g., more code examples)

Some commentary:

- We already usually have 1 code demo per lecture
  - Hard to squeeze more code demos in lecture!
  - Some demos are already only covered partially or briefly in lecture
- Please actually play with demos yourself after class
  - Understand every line of code & whether it can be coded differently
  - A major theme in the course is that of *infinite design choices*: identify where design choices appear & change them to see what happens
  - Try deleting all code cells except part that loads in data & see if you can redo the exploratory data analysis on your own

HW1 Questionnaire (5/5)

- Some students asked why we aren't covering more up-to-date content
  - I want my course to teach you fundamental concepts underlying unstructured data analysis, starting from simple ideas people tried years ago that are easier to understand before progressing to more modern methods that can be more difficult to understand & interpret
  - As I mentioned in week 1 already, we are going to be covering a basic version of generative pretrained transformers ("GPTs") in the last week of the course

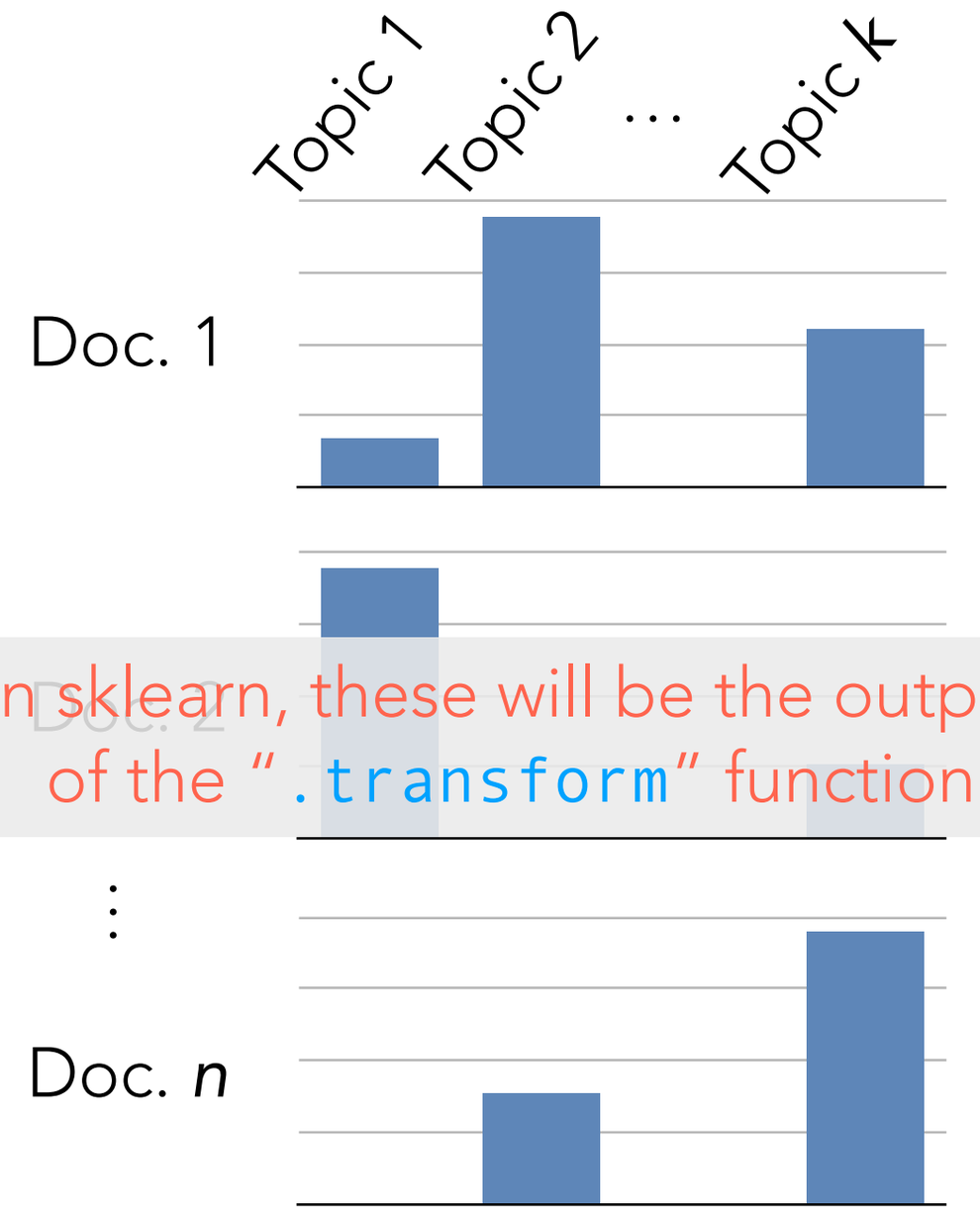
I think it's helpful to see how ideas have evolved over time
& what limitations are of older models that newer models try to address

- For what it's worth: from working in the machine learning field for over a decade now, I can say that what technology is hot keeps changing
  - Be careful with getting too caught up in trends that might go out of fashion soon!
  - GPTs might be hot now but in 5-10 years, we'll probably be using something different!

Administrivia

- Reminder: HW1 & Quiz 1 regrade requests due tonight April 7, 11:59pm
  - See Canvas announcement I sent out last Wednesday April 2 for details
- Reminder: if you have a hard conflict with Quiz 2 (Friday May 2, 1pm-2:20pm), please let me know by tonight April 7, 11:59pm
  - See Canvas announcement I sent out last Tuesday April 1 for details
  - After tonight, we will **not** be taking requests for taking Quiz 2 at a different time than the regular time
    (except for a medical emergency on the day of Quiz 2)

(Flashback) LDA Generative Model



In sklearn, these will be the output of the "`.transform`" function

Goal: Learn these distributions

In sklearn, these will be in "`.components_`"

LDA models each word in document *i* to be generated as:

1. Randomly choose a topic *Z* (use topic distribution for doc *i*)
2. Randomly choose a word (use word distribution for topic *Z*)

How to choose the number of topics *k*?

How "Coherent" is a Topic?

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 : 0.009768217977593912
years : 0.00843922077825026
game : 0.0084164825794737 57
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
```

If we see the word "good", how likely are we to see the word "years"?

P(see word "years" | see word "good")

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

Focus on a single topic at a time

Coherence of topic:

$$\sum_{\text{top words } v,w \text{ that are not the same}} \log \mathbb{P}(\text{see word } v | \text{see word } w)$$

$$\frac{\#\text{ documents that mention both } v \text{ and } w}{\#\text{ documents that mention } w} \quad +0.1$$

avoid numerical issues

How Different is a Topic from the Others?

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 # of unique top words

How to Choose Number of Topics *k*?

Compute:

Topic `0`'s coherence score	Topic `0`'s # unique top words
Topic `1`'s coherence score	Topic `1`'s # unique top words
⋮	⋮
Topic (`k-1`)'s coherence score	Topic (`k-1`)'s # unique top words
Compute average coherence	Compute average # unique top words

Can plot average coherence vs *k*, and average # unique top 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 top words (they aren't even in the same units!!!)

How to Choose Number of Topics *k*?

Demo

Topic Modeling: Last Remarks

- There are score functions aside from coherence & # unique top words (e.g., normalized-mutual-information coherence, (log) lift score, …)
- There are *many* topic models, not just LDA
  (e.g., Hierarchical Dirichlet Process, correlated topic models, SAGE, anchor word topic models, Scholar, embedded topic model, …)
- *Dynamic* topic models can track how topics change *over time*
  - Requires time stamp for every text document we fit the model to
- **Warning:** learning topic models is very sensitive to random initialization
  - Can try fitting data multiple times using different random seeds & seeing which topics consistently show up across random seeds
- There are variants of topic models where users provide supervision (e.g., user specifies what some topics should be about in terms of top words, or where a topic should predict some outcome)
- On the course webpage, I posted a link to Maria Antoniak's practical guide for using LDA (very helpful if you want to use LDA in the future!)

95-865

Part I: Exploratory data analysis

Identify structure present in "unstructured" data

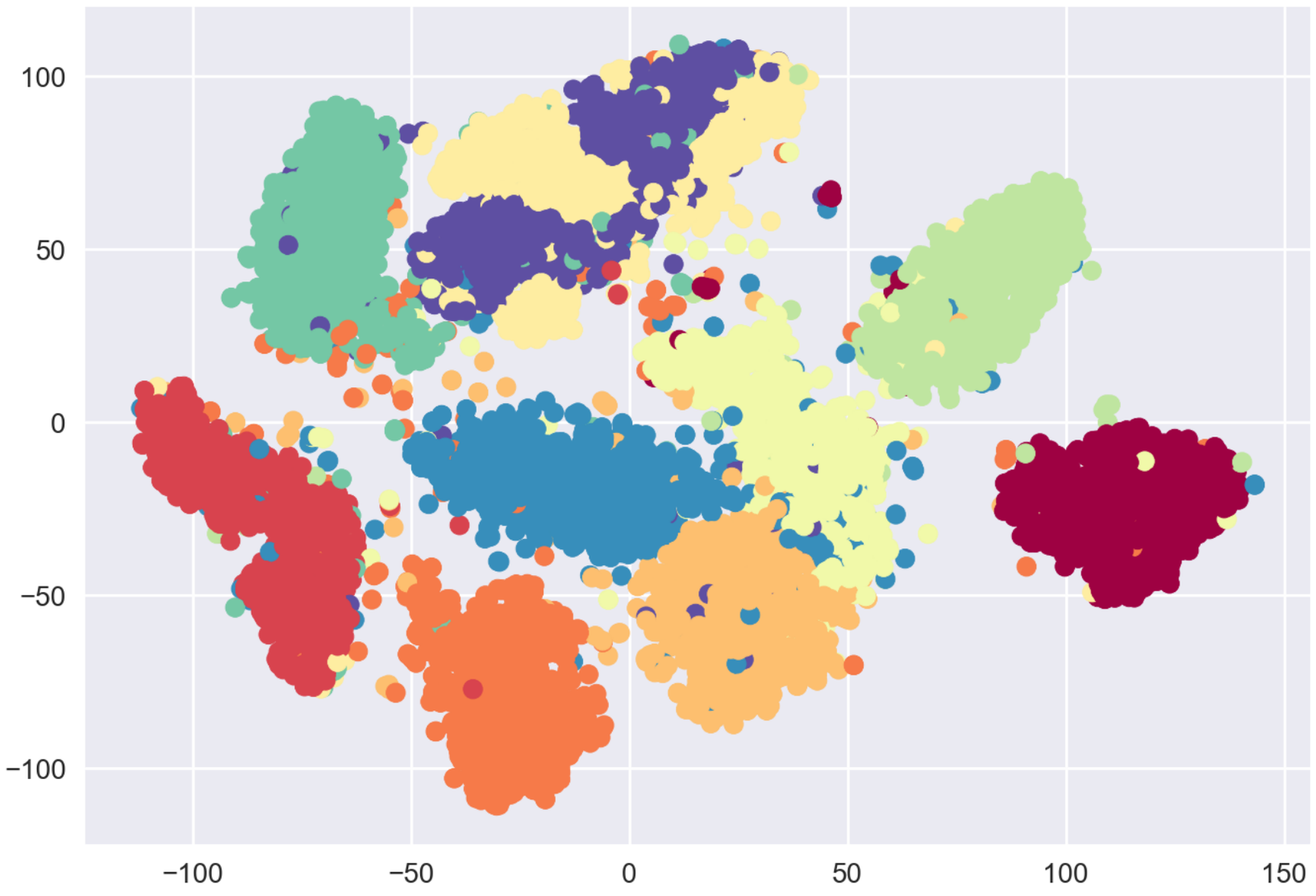
- Frequency and co-occurrence analysis
- Visualizing high-dimensional data/dimensionality reduction
- Clustering
- Topic modeling

Part II: Predictive data analysis

Make predictions using known structure in data

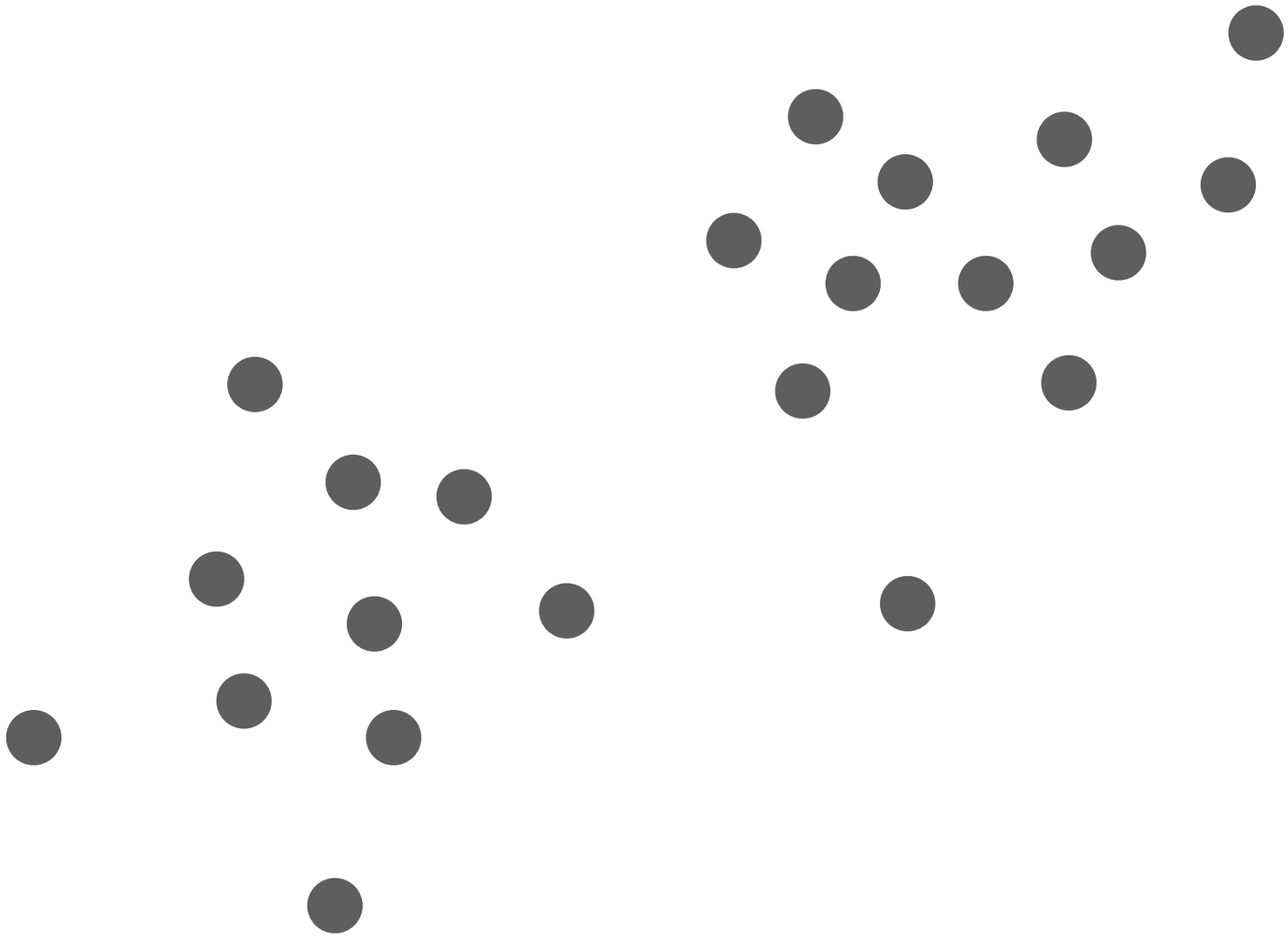
- Basic concepts and how to assess quality of prediction models
- Neural nets and deep learning for analyzing images and text

What if we have labels?



Example: MNIST handwritten digits have known labels

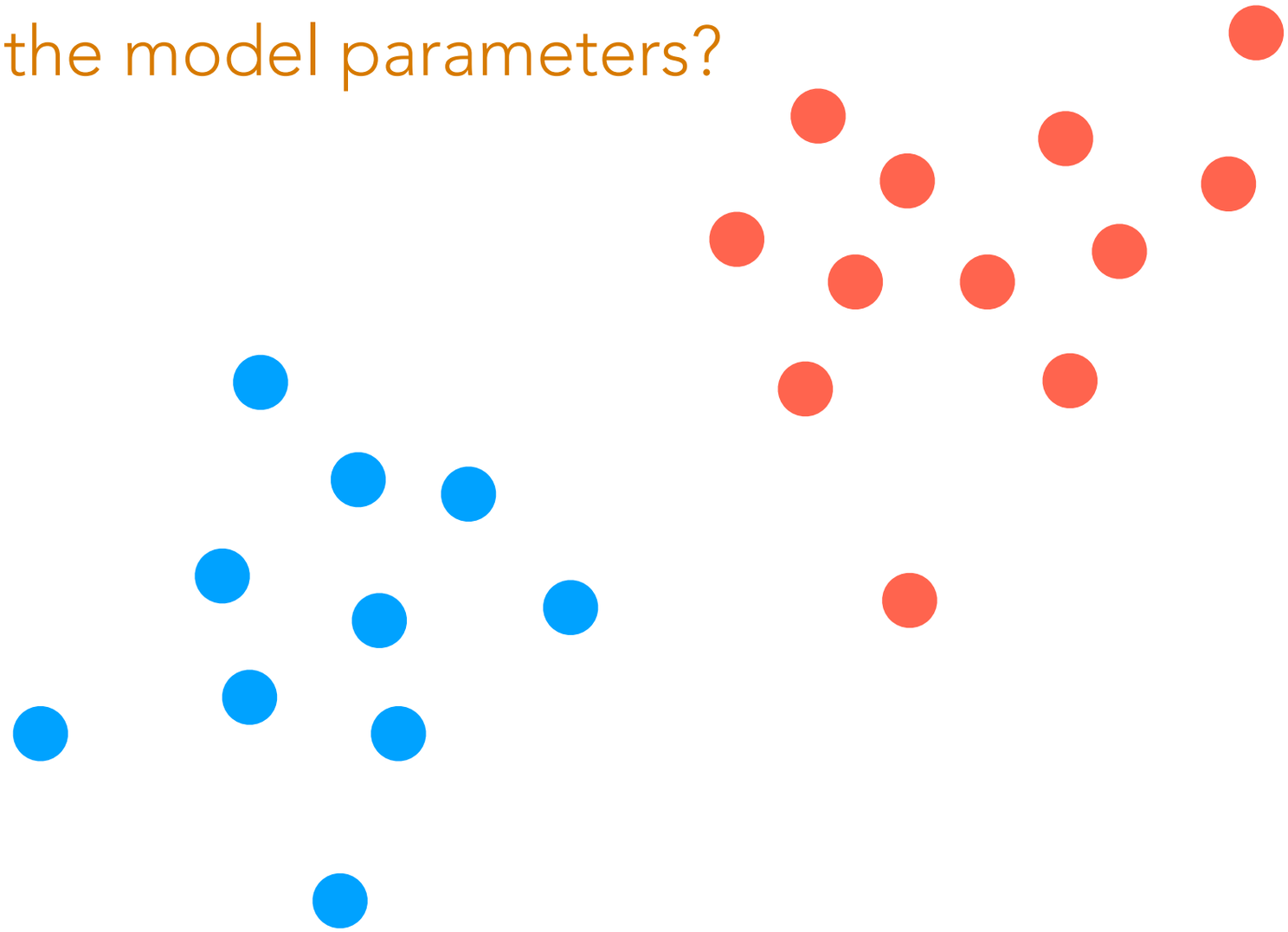
If the labels are known…



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: Guess k

Step 1: Guess cluster probabilities, means, and covariances

(often done using k-means)

Repeat until convergence:

Step 2: Compute probability of each point being in each of the k clusters

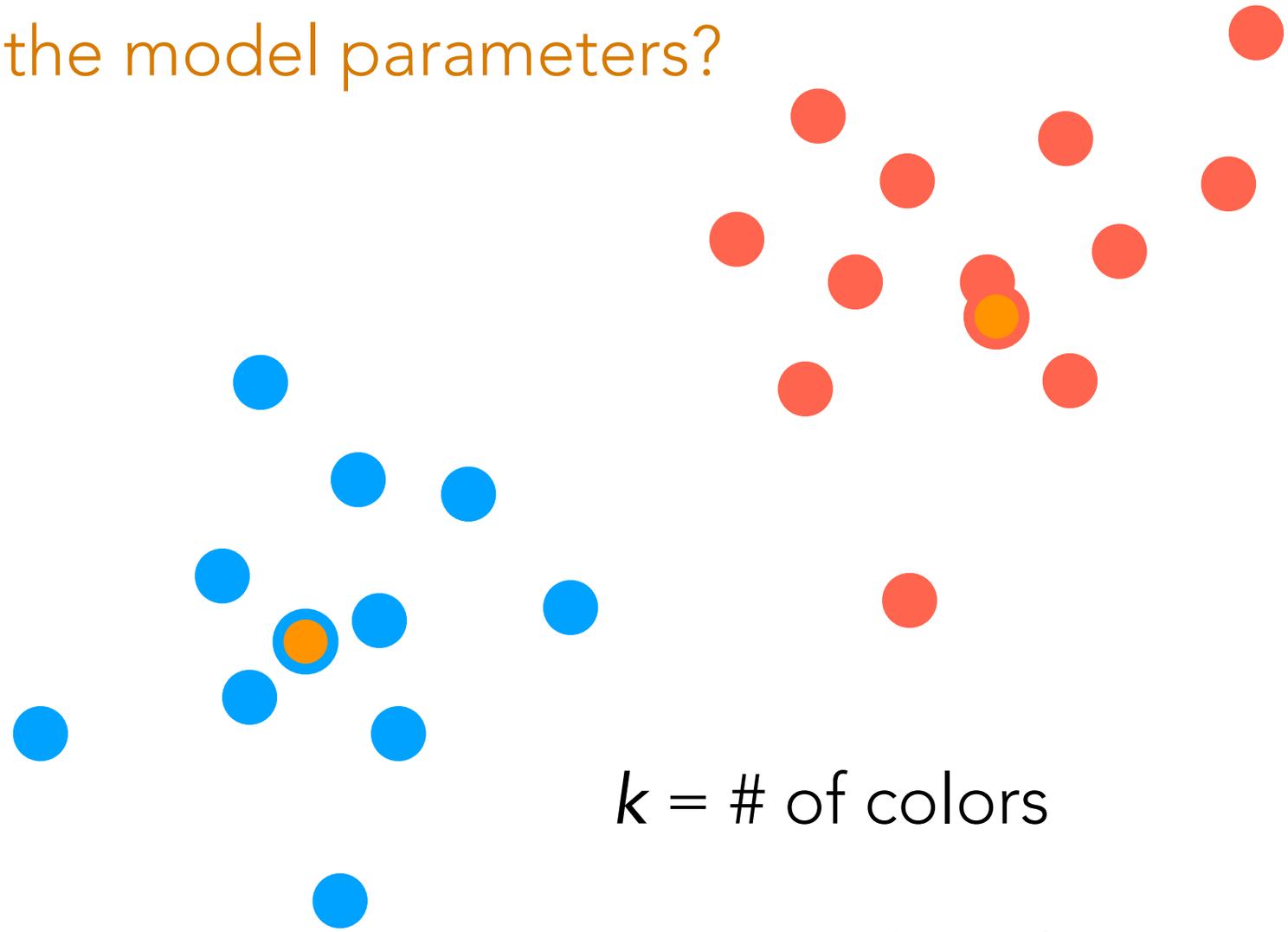
Step 3: Update cluster probabilities, means, and covariances accounting for probabilities of each point belonging to each of the clusters

We don't even need to repeat until convergence

If the labels are known…

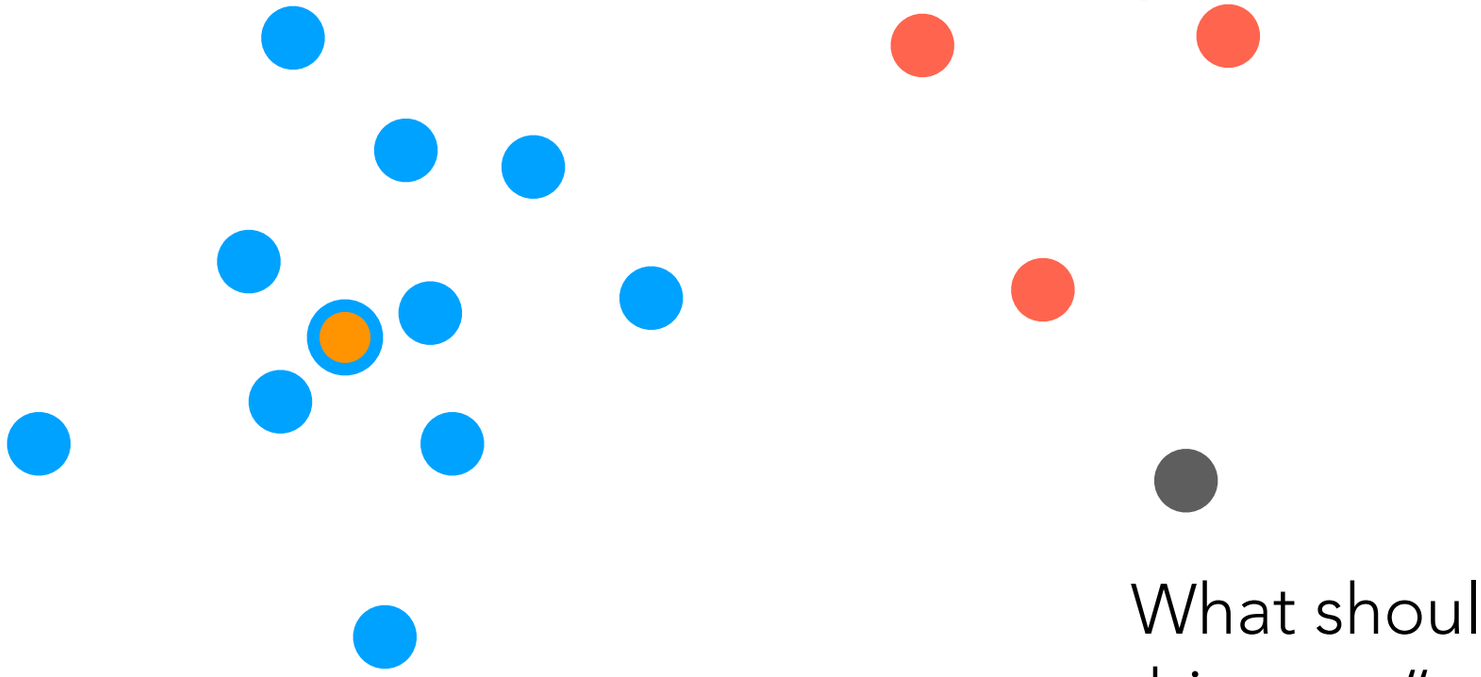
And we assume data generated by GMM…

What are the model parameters?



k = # of colors

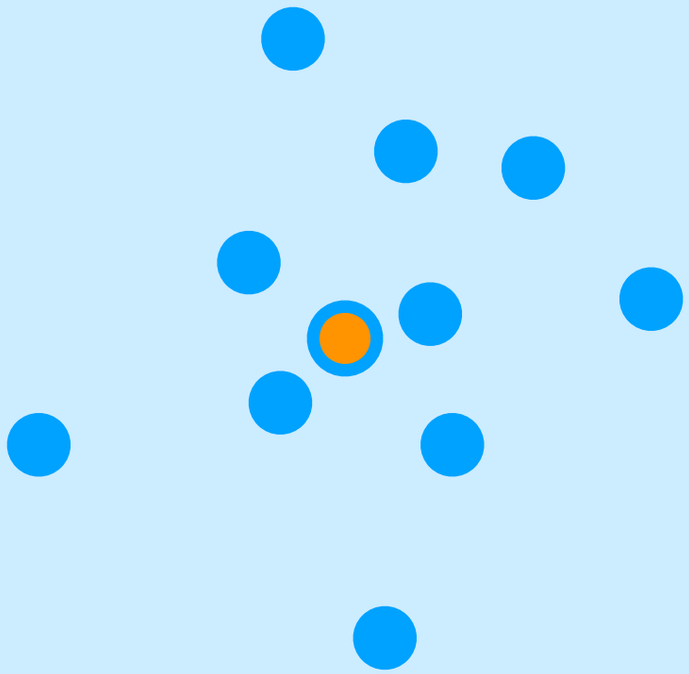
We can directly estimate cluster means, covariance matrices



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

Whichever cluster has higher probability!

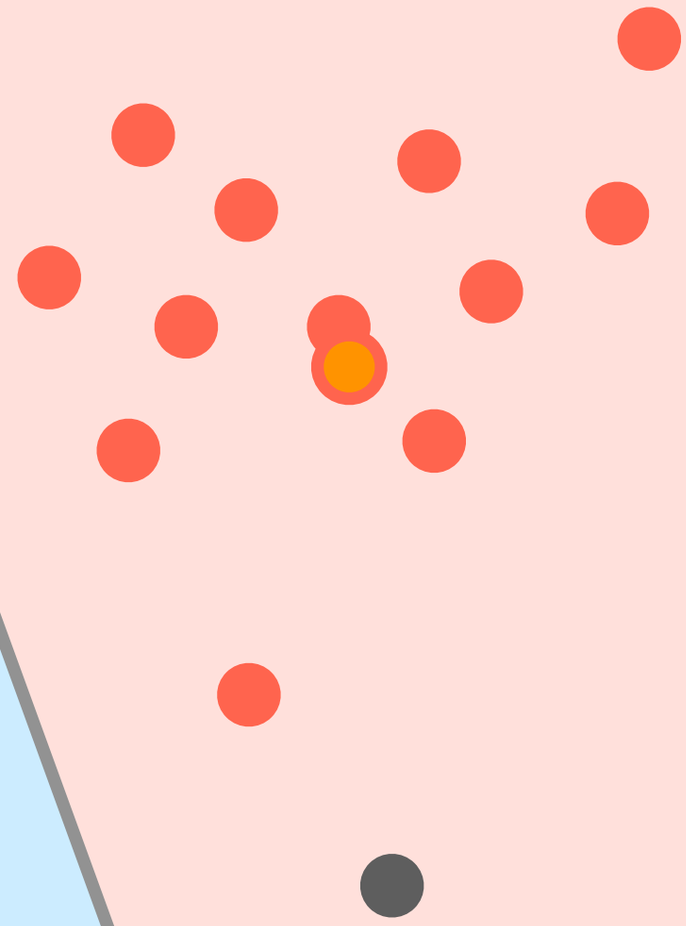
Decision boundary



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

We just created a **classifier**

(a procedure that given a 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!

Predictive Data Analysis

Training data

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

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**

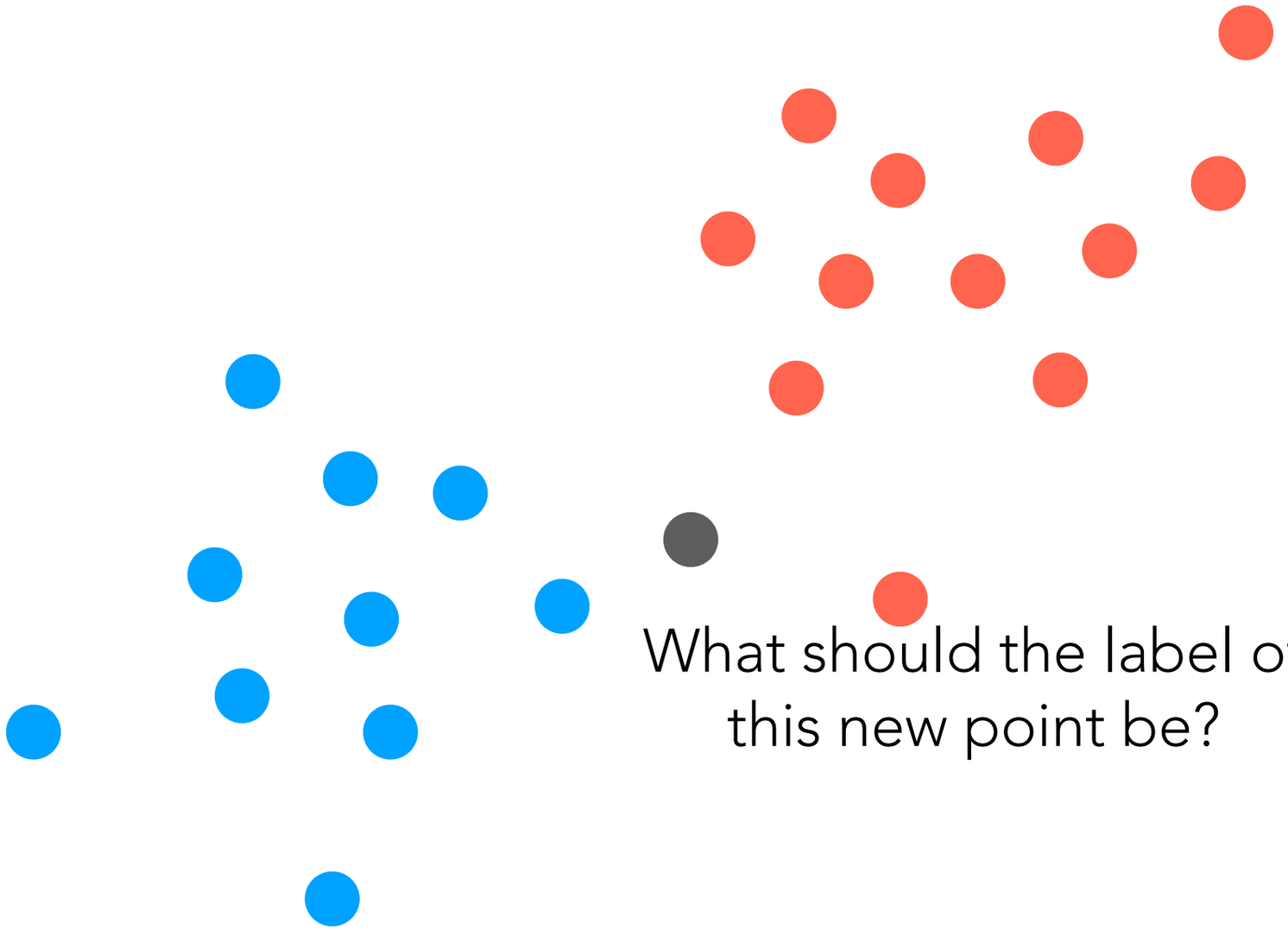
We could have many such test feature vectors, which we collectively refer to as *test data*

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)

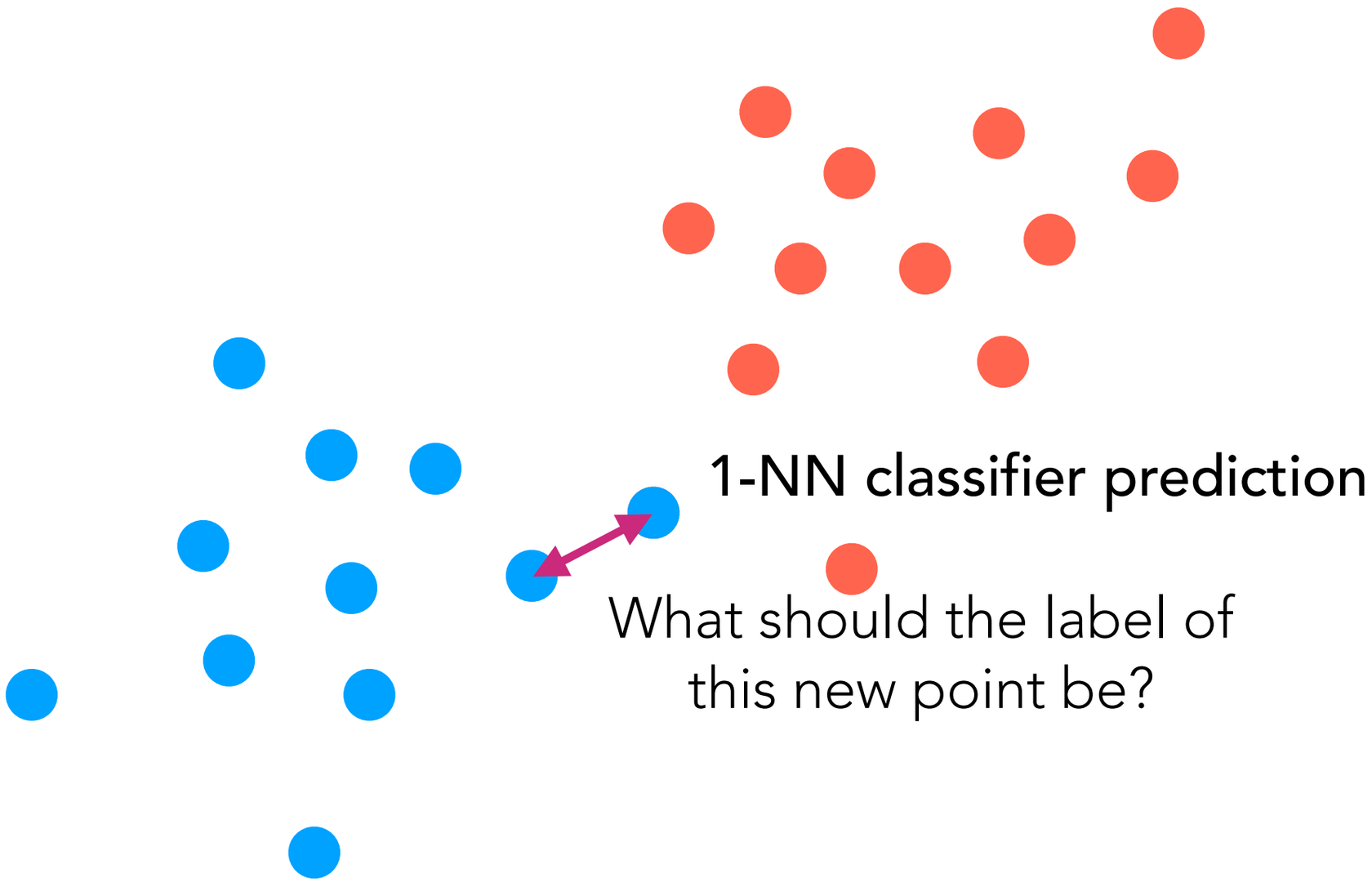
Example of a Discriminative Method: *k*-NN Classification

Example: *k*-NN Classification

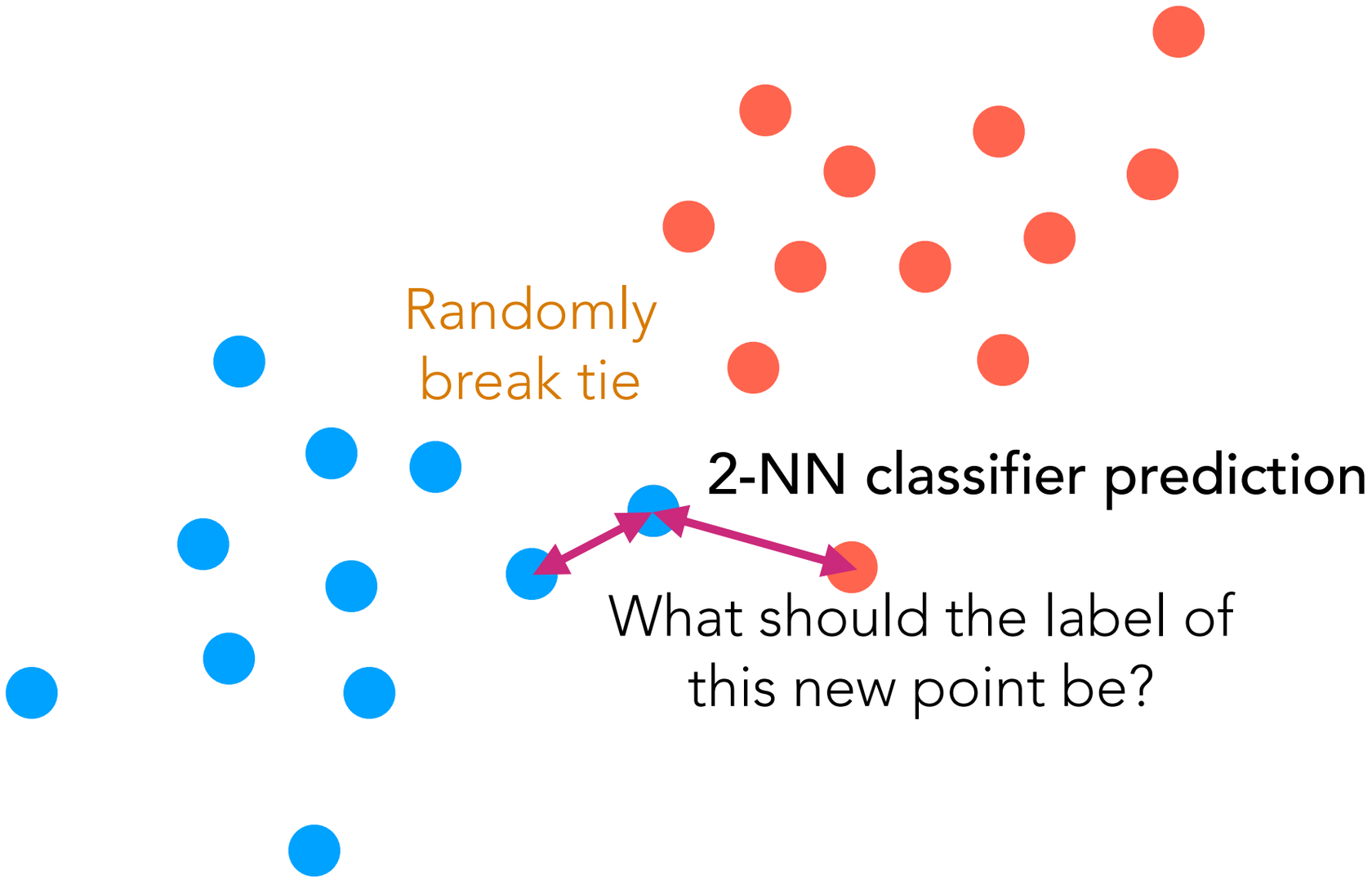


What should the label of this new point be?

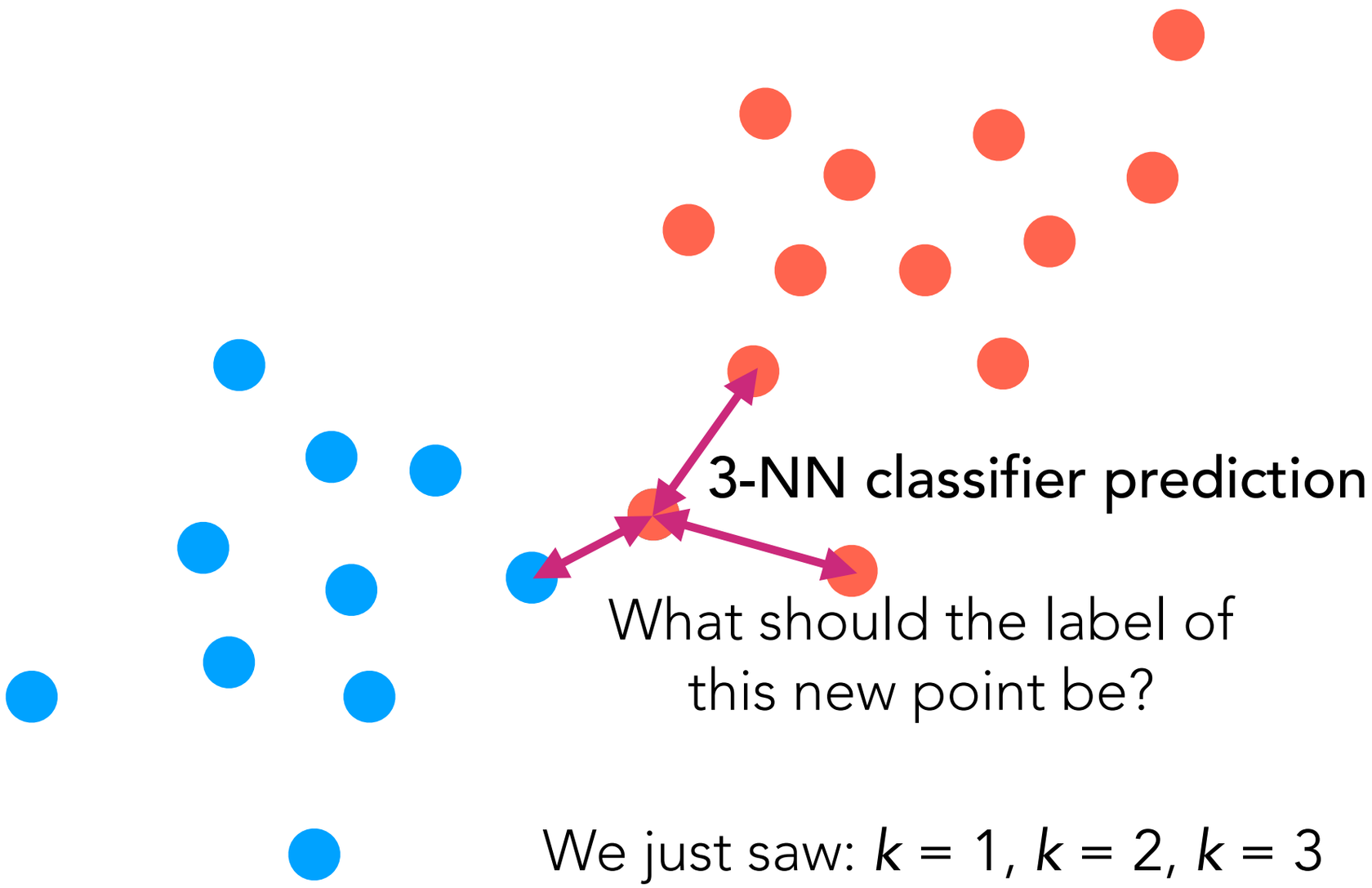
Example: *k*-NN Classification



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

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* automatically get fit to training data

- Example: Gaussian mixture model
  - Hyperparameter: number of clusters *k*
  - Parameters: cluster probabilities, means, covariance matrices

- Example: *k*-NN classification
  - Hyperparameter: number of nearest neighbors *k*
  - Parameters: N/A

Actually, there's another hyperparameter: distance function to use
(for simplicity, we assume Euclidean distance for now)