Clustering (COSC 425)

Upcoming Classes/Announcements

- Today: Lab 2 and Clustering
- Thursday, September 15: ***In Person***
  - Topic: Perceptron
- Study guide for Exam 1 provided on Friday, September 16
- Tuesday, September 20: ***Pre-recorded lecture and Zoom for questions during class*** (only between 12:30 and 1:30)
  - Practical Issues
- Thursday, September 22: ***Zoom link for class***
  - Exam Review! ***Don't miss this!***
  - This is your opportunity to ask questions about the exam
  - We will also do practice questions with the clickers
- Lab 2 due on September 23!
- ***Tuesday, September 27: In Person: Exam 1***

Pop Quiz

cs425

Question 1

- Given how you did with lab 1: How comfortable do you feel with matplotlib?
  - A) Don’t feel like I know where to start: I need more examples in class
  - B) I know what to Google
  - C) I feel somewhat comfortable
  - D) I’m an expert

Question 2

- Given how you did with lab 1: How comfortable do you feel with pandas?
  - A) Don’t feel like I know where to start: I need more examples in class
  - B) I know what to Google
  - C) I feel somewhat comfortable
  - D) I’m an expert

Another other feedback on Lab 1?

Lab 2: K-Nearest Neighbors

Lab 2: K-Nearest Neighbors

- In this lab, you will be practicing using K-nearest neighbors and applying it to a dataset. Here, the data will be statistics about the Olympic athletes from the 2016 Rio Olympic games (available here: https://www.kaggle.com/datasets/rio2016/olympic-games).

- You will create a Jupyter notebook that you will submit where you will put the code that you wrote for this lab. You will also submit a separate written report in PDF form in which you include the answers to the questions and/or plots required by the question.

Lab Questions

- **Question 1** (15 points): Read in the data from the given CSV file. Down-select to create a dataframe that only includes three columns (sport, height, and weight) for all of the males who competed in either basketball or cycling. This should result in a dataframe with 3 columns and 450 rows. Convert the non-numerical feature (sport) to numerical values. Further, convert the height values from meters to centimeters. Include in your report a brief description (1-2 sentences) of why it might be a good idea to use the data in centimeters rather than meters for k-nearest neighbors.

- **Question 2** (15 points): For each class (basketball, cycling) and feature (height, weight), compute the following statistics: minimum, maximum, mean, standard deviation, and signal-to-noise ratio (SNR = mean/standard deviation). Create a table that shows the statistics for each class and feature. Do any of the statistics give rise to concern?

Lab Questions

- **Question 3** (10 points): Create a scatter plot of the data, where you color the basketball data and cycling data differently. Make sure to label your axes and include a legend for the labels. Does the scatter plot indicate potential success or failure with respect to separating the two classes using KNN? Why?

- **Question 4** (5 points): Divide the data into training and test data using the standard 80-20 ratio. Apply 10-fold cross-validation to the training data.

Lab Questions

- **Question 5** (25 points): Create and iterate over different numbers of neighbors from 1 to 21 (inclusive, but odd numbers only). For each hyperparameter, calculate the mean and standard deviation of classification accuracy across the different folds of the data. Create a table for your report showing the mean and standard deviation for each hyperparameter. Select which hyperparameter to use based on the mean classification accuracy on the validation data. Note which hyperparameter value is selected in the report.

- **Question 6** (10 points): Build a new K-nearest neighbor classifier using all your training data with the hyperparameter defined in Question 5. Use the resulting model to classify the test data (which haven't used until now). Calculate and report overall training and test data performance.

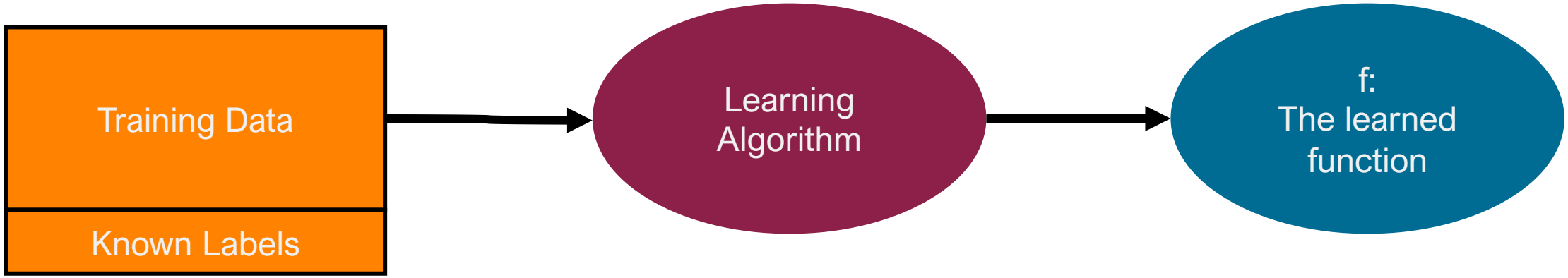
Lab Questions

- **Question 7** (20 points): Create a 2D plot that visualizes the classifier's performance (a decision boundary). Run a mesh of data through the classifier to determine basketball and cycling decision regions. Color them two different light colors (e.g., alpha=0.2) that are easy to visually separate. Then, overlay the test data using two different colors for basketball and cycling. Make sure all plots are labeled and that a legend is included.

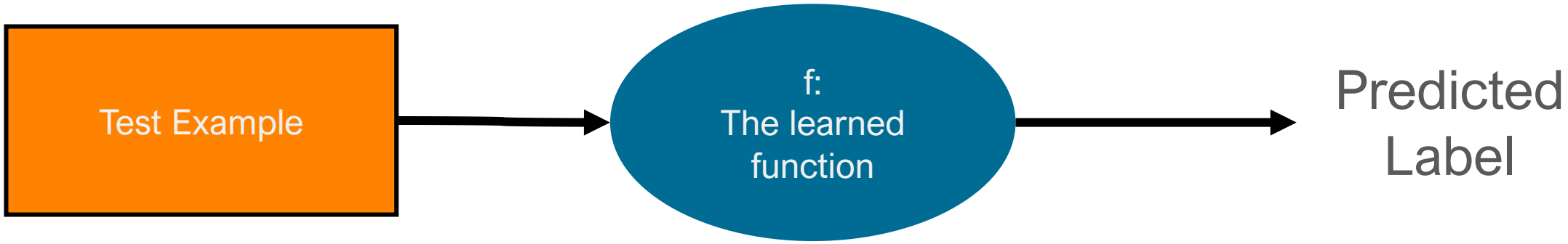
Lab 2

- Submission check list:
  - Jupyter notebook
  - PDF of your report
- Lab 2 is due on September 23

Inductive Machine Learning



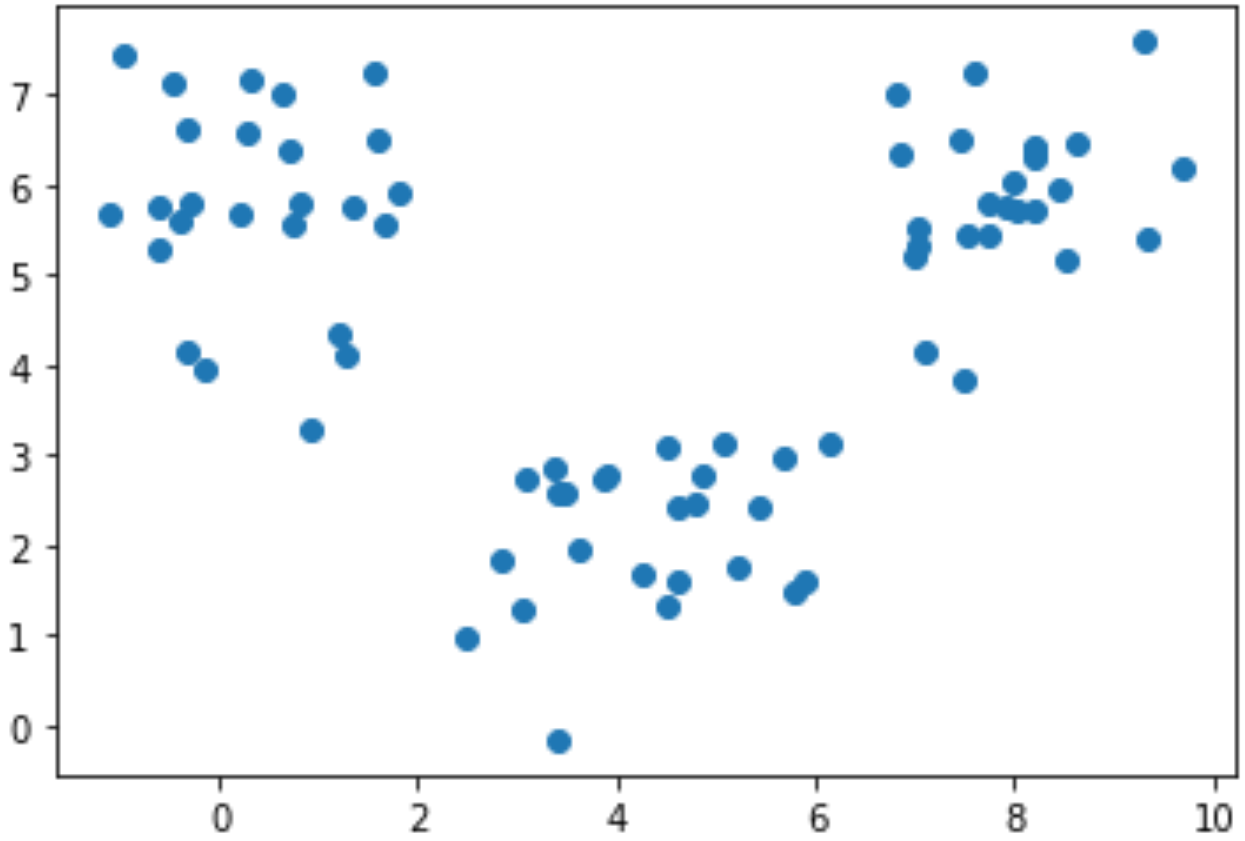
Inductive Machine Learning



Clustering

Unsupervised Learning

- In this case, we ONLY have the data/features, we do not have the labels at all
- Our job as machine learning users is to still interpret and make sense of the data even without labels
- Your job is to split it up into clusters/groups

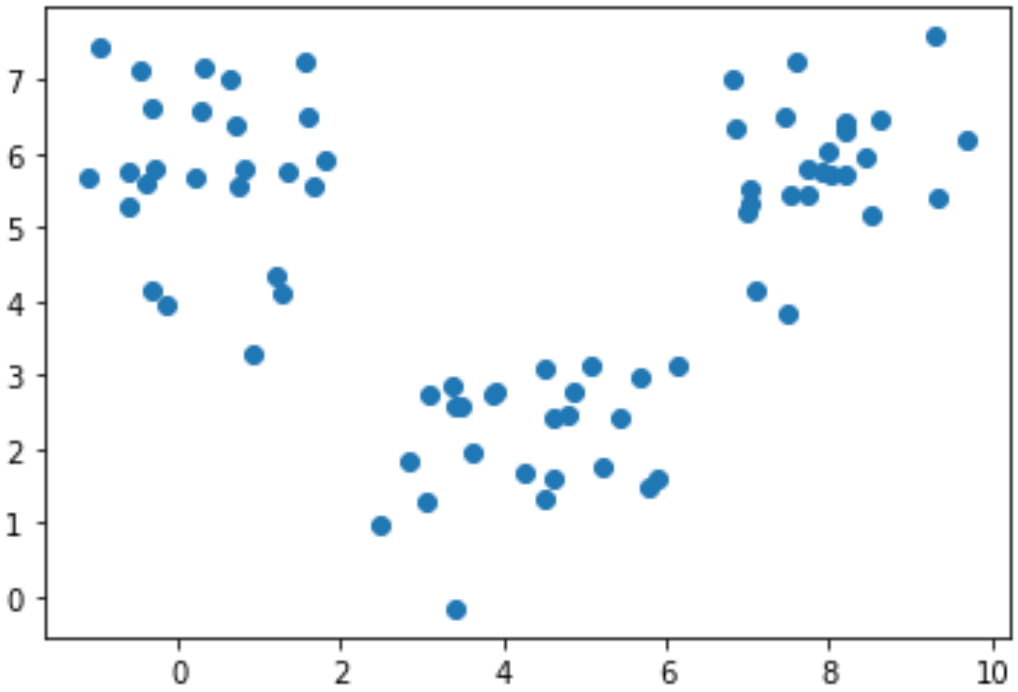


Pop Quiz

cs425

Question 3

- How many clusters do you see?



K-Means Clustering

- If we know the center of each cluster, then we can assign each point to its nearest center
- If we know the assignment of points to clusters, we can calculate the center
- So, what do we do in this chicken and egg problem?

K-Means Clustering

Iteration!

To the Notebook!

K-Means Clustering

K-Means Clustering

- We can add a stopping condition to say stop updating when the centers stop updating
- Given that: Will it converge?
- Yes, it will converge!
- How long will it take?
  - Theoretically, a long time
  - In practice, usually not that long

K-Means Clustering Objective

- K-Means is optimizing (i.e., minimizing) the sum of square distances from any data point to its assigned center

$$L(z, \mu, D) = \sum_n \left\| x_n - \mu_{z_n} \right\|_2^2$$

Theorem: K-Means Convergence

- For any dataset D and any number of clusters K, the K-means algorithm converges in a finite number of iterations, where convergence is measured by L ceasing to change

Algorithm 4 K-MEANS(**D**, K)

```
for k = 1 to K do
    μ_k ← some random location          // randomly initialize center for kth cluster
end for
repeat
    for n = 1 to N do
        z_n ← argmin_k ||μ_k − x_n||    // assign example n to closest center
    end for
    for k = 1 to K do
        X_k ← { x_n : z_n = k }         // points assigned to cluster k
        μ_k ← MEAN(X_k)                 // re-estimate center of cluster k
    end for
until μs stop changing
return z                                // return cluster assignments
```

Proof of K-Means Convergence

- In line 6:
  - Suppose the previous value of z_n is a and the new value is b
  - This has to be true:
    $$\|x_n - \mu_b\|_2 \leq \|x_n - \mu_a\|$$
  - So, L will decrease
- In line 9:
  - μ_k is the mean of the data points for which $z_n = k$
  - This is precisely the point that minimizes the squared distances, so it can only decrease L (or keep it the same)

Algorithm 4 K-MEANS(D, K)

```
1:  for k = 1 to K do
2:      μ_k ← some random location          // randomly initialize center for kth cluster
3:  end for
4:  repeat
5:      for n = 1 to N do
6:          z_n ← argmin_k ||μ_k − x_n||    // assign example n to closest center
7:      end for
8:      for k = 1 to K do
9:          X_k ← { x_n : z_n = k }         // points assigned to cluster k
10:         μ_k ← MEAN(X_k)                 // re-estimate center of cluster k
11:     end for
12: until μs stop changing
13: return z                                // return cluster assignments
```

Proof of K-Means Convergence

- After the first pass through the data, there are finitely many possible assignments of z (the cluster labels) and μ, because z is discrete and μ can only take on a finite number of values, which is the means of some subset of the data
- L is lower-bounded by zero,
- Together: L can only decrease a finite number of times

Algorithm 4 K-MEANS($\mathbf{D}$, K)

```
for k = 1 to K do
    μ_k ← some random location          // randomly initialize center for kth cluster
end for
repeat
    for n = 1 to N do
        z_n ← argmin_k ||μ_k − x_n||    // assign example n to closest center
    end for
    for k = 1 to K do
        X_k ← { x_n : z_n = k }         // points assigned to cluster k
        μ_k ← MEAN(X_k)                 // re-estimate center of cluster k
    end for
until μs stop changing
return z                                // return cluster assignments
```

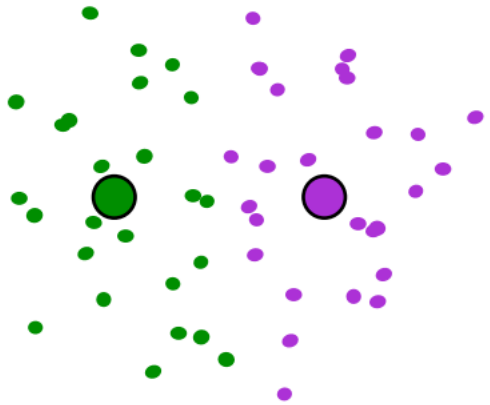
Important Notes on Convergence

- The convergence will only be to a local optima of L!
- In practice:
  - You should run it >= 10 times with different initializations and pick the one that minimizes L
- Again, theoretically exponential convergence time
- In practice, on a limited floating point precision machine, it will converge in polynomial time

Unsupervised Learning and the “Right Answer”

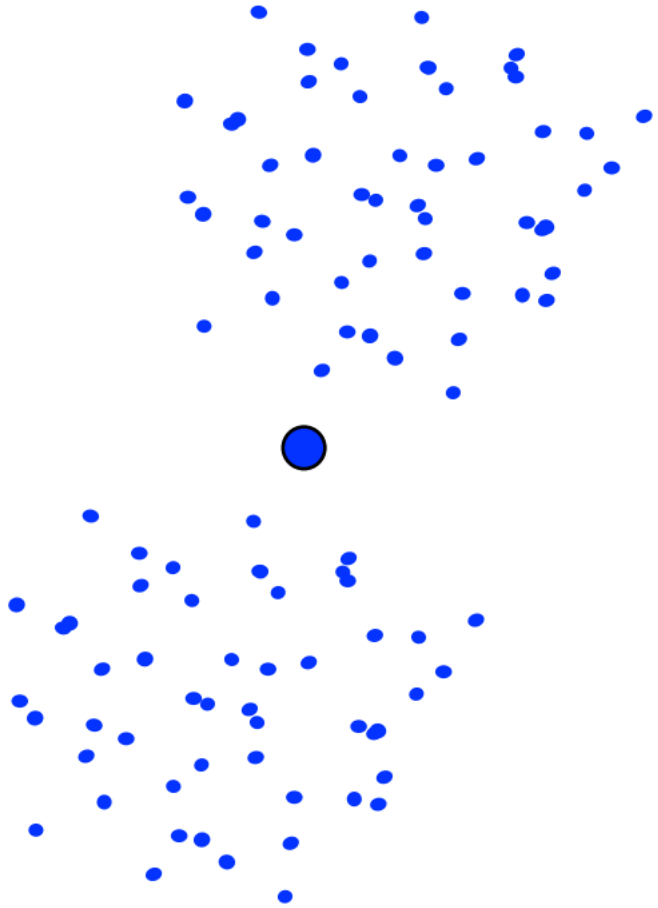
- Because k-means clustering is only guaranteed to converge to a local optima, there’s no way of knowing if it’s the best solution
- Additionally, with unsupervised learning, there’s no way to know if the solution that was produced is the “right answer”
- K-Means Algorithm is a heuristic
  - Requires initial means: it matters what you pick!

K-Means Getting Stuck



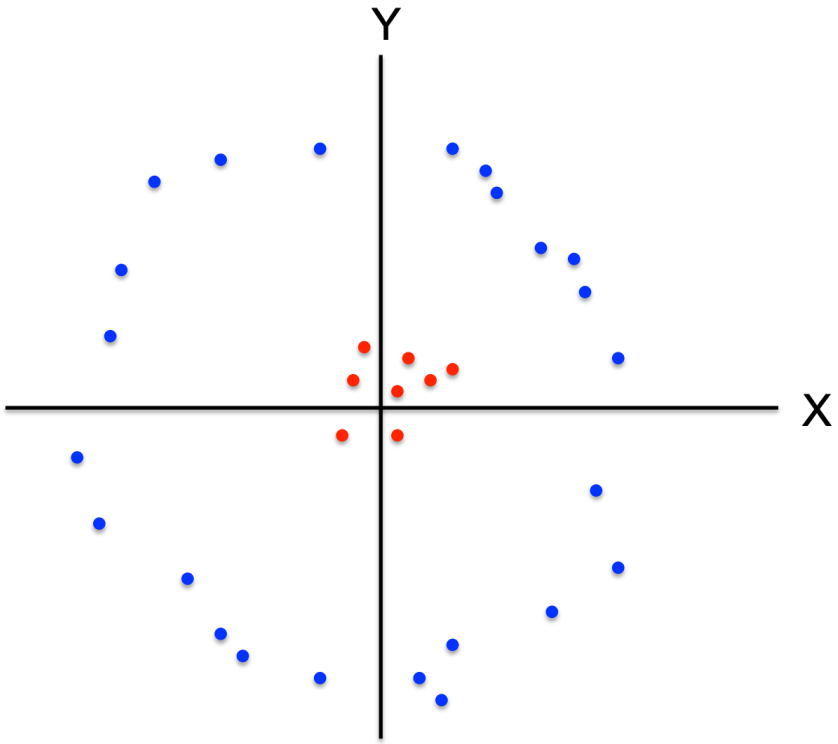
Would be better to have one cluster here

Better to have two clusters here



Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

K-Means not able to properly cluster



Source: https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf

Scikit-Learn Kmeans: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Get ready for some upsetting math

The Dangers of High Dimensions

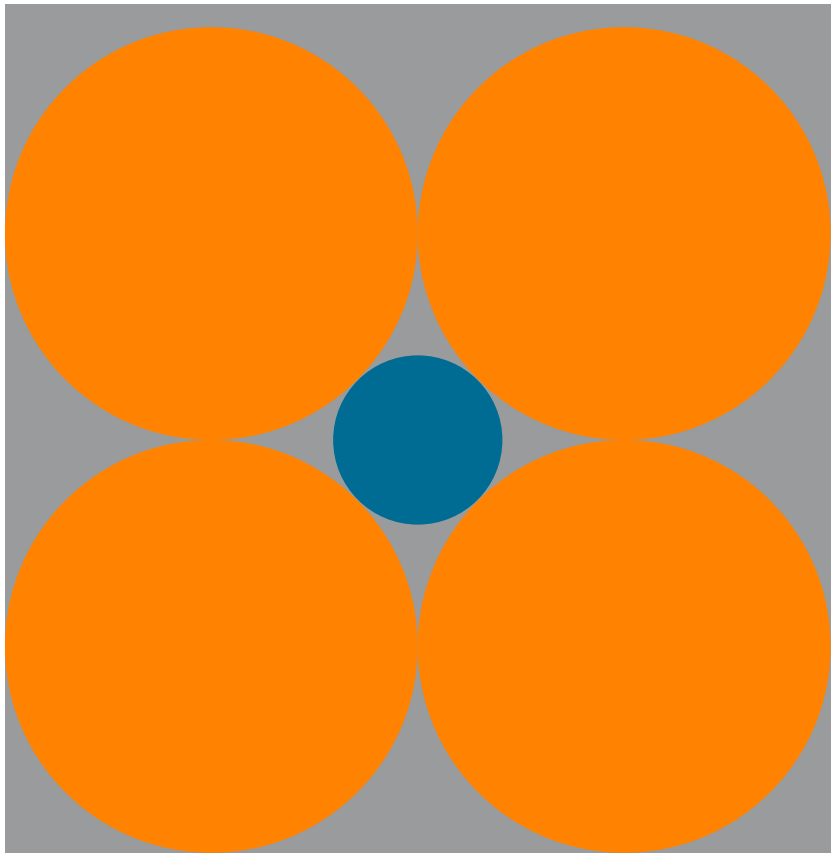
- We've built up intuition about how these algorithms work by visualizing the data
- This is easy for two, slightly harder for three, and really hard for anything larger than three
- This can make building up an understanding of the data and which machine learning approach might be useful *really difficult*

Computational Curse of Dimensionality

- For K-Nearest Neighbors, the speed of prediction is slow for a large dataset
- You have to look at all of the training data every time you do a prediction
- You might want to create an indexing structure to make these easier
  - Divide up your ”grid” into regions and only look at the points in the region that the prediction point lies inside
  - This can give HUGE computational savings
- This is easy for 2 dimensions, but by the time we get to 20 dimensions, the gridding technique is only useful if you have enough data

Curse of Dimensionality

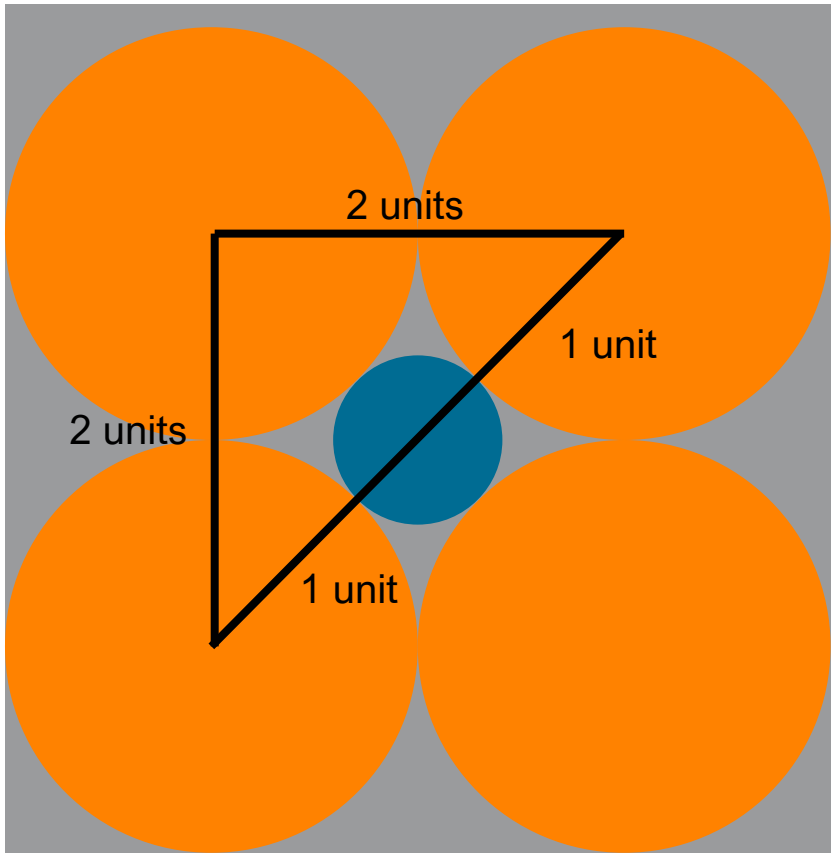
- Mathematical Curse of Dimensionality:
  - The intuitions built up for 2 and 3 dimensions don't carry over
  - From the textbook: High dimensional spheres "look more like porcupines than balls"
- Arbitrarily, suppose we have a D dimensional space, where we will have 2^D orange hyperspheres with radius 1, each of which touch exactly n-many other orange hyperspheres
- By the Pythagorean theorem, the radius of the blue sphere in the middle that touches ALL of them is $r = \sqrt{D} - 1$



Curse of Dimensionality

- Arbitrarily, suppose we have a D dimensional space, where we will have 2^D orange hyperspheres each with radius 1, each of which touch exactly n-many other orange hyperspheres
- By the Pythagorean theorem, the radius of the blue sphere in the middle that touches ALL of them is $r = \sqrt{D} - 1$

$$2^2 + 2^2 = (2 + 2r)^2$$
$$2^2(1 + 1) = 2^2(1 + r)^2$$
$$2 = (1 + r)^2$$
$$\sqrt{2} = 1 + r$$
$$\sqrt{2} - 1 = r$$



Curse of Dimensionality

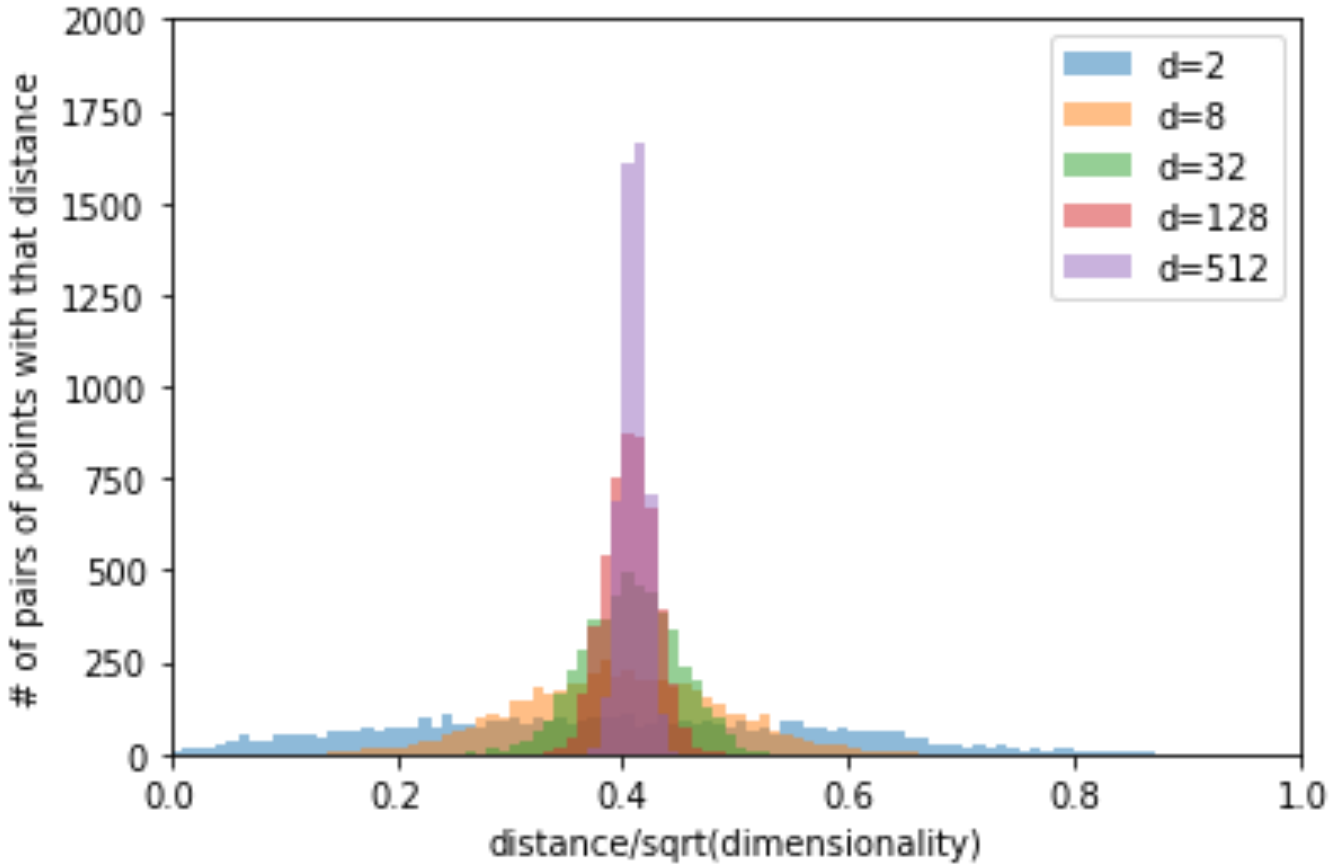
- As D grows, the radius of the blue hypersphere grows without bound
- Suppose $D = 9$: the radius of the blue hypersphere is now 2.
  - $r = \sqrt{D} - 1 = \sqrt{9} - 1 = 3 - 1 = 2$
  - With the radius of 2, it's squeezing between the orange hyperspheres and touching the hypercube surrounds them
- When we get to $D \geq 10$, the radius is around 2.16 and now the blue hypersphere is poking outside of the hypercube!

To the Notebook!

Curse of Dimensionality

What does this mean???

- Even for reasonable sized dimensionality (512 features), distance between points starts to be similar
- Distance doesn't mean much for larger dimensionality problems
- KNN ONLY gets distances!



What does this mean???

- Does this mean we can't/shouldn't use KNN for larger dimensionality problems?
- NOT NECESSARILY:
  - Data is not uniformly distributed in the hypercube

To the Notebook!

Distance and Digits

Before You Go

- Interested in using Deep Learning for composites research? The Polymer Composite Additive Manufacturing lab (https://pcam.utk.edu/) has a paid research assistant position available for undergraduate students interested in using state-of-the-art Deep Learning architectures for image processing. The project will involve implementing Machine Learning and Deep Learning models to segment and measure fibers within images. Undergrads interested in applying should have moderate experience with Python programming (functions, classes, etc.) and be a self-starter. Ideal candidates will have experience with Python libraries like NumPy, Matplotlib, and TensorFlow. The chosen candidate will be expected to present their work at the end of the year and attend 2 required workshops. There is also potential for co-authorship on any work which may be incorporated into PCAM research group papers. If you are interested in applying, please send your resume to Jay Pike (graduate student) - jpike1@vols.utk.edu.

For next class

- We'll go over the Perceptron!
  - The first fundamental building block of neural networks
- Pay attention to the class announcements on Canvas for when you should/shouldn't be coming in person!
- Don't forget that Exam 1 is scheduled for September 27
  - Review will be on September 22
  - Study guide will be available this Friday (September 16)