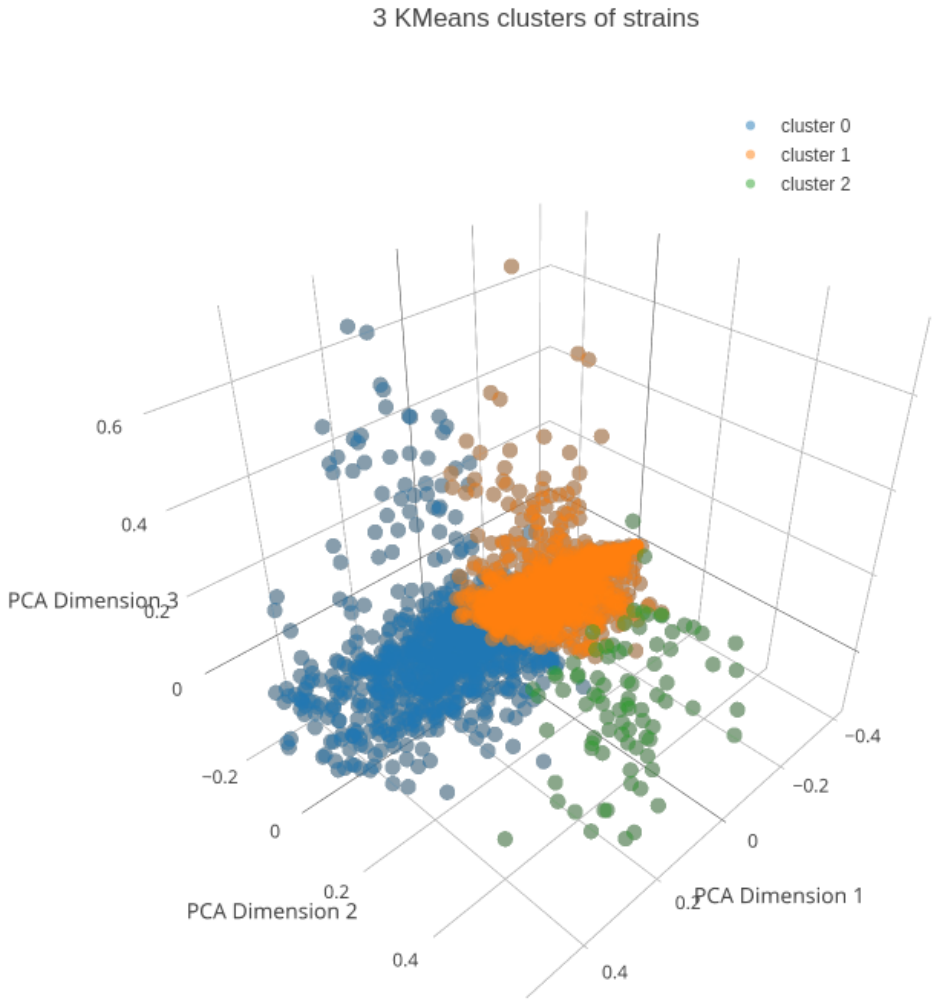
Week 6: K-means and clustering



Week 5 review

- What size of data is SVM good for?
  -
- What is the trick we can use for overlapping data with SVMs?
  -
- What are some of the different neural net layers available?
  -
- How do we train a neural net?
  -

Week 5 review

- What size of data is SVM good for?
  - Not large (< 20k or < 10k points usually, runtime is O(n*d) or O(n*d^2)
- What is the trick we can use for overlapping data with SVMs?
  - Kernel trick – takes the dot product in a higher-D space without knowing the transformation function
  - RBF kernel transforms features into infinite-D space
- What are some of the different neural net layers available?
  - Dense, Convolutional, Maxpool, dropout, batchnormalization,
- How do we train a neural net?
  - Set up the architecture – input dimensions, number of layers, size of layers, type of layers, optimizer, loss function, compile it, then actually train it
  - Data is passed forward (inference), then the errors are backpropagated through the net to update the weights in order to make better predictions

SVM optimization and kernel trick

- After many pages of math (this is on page 57 of the ebook) we get to the Wolfe dual Langrangian problem, which is our algorithm for SVMs:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

SVM optimization and kernel trick

- (pg 57 of the ebook) Wolfe dual Langrangian
- Kernel function is the dot product in a higher-D space (pg 75 in ebook): $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- The RBF kernel is in infinite-D space

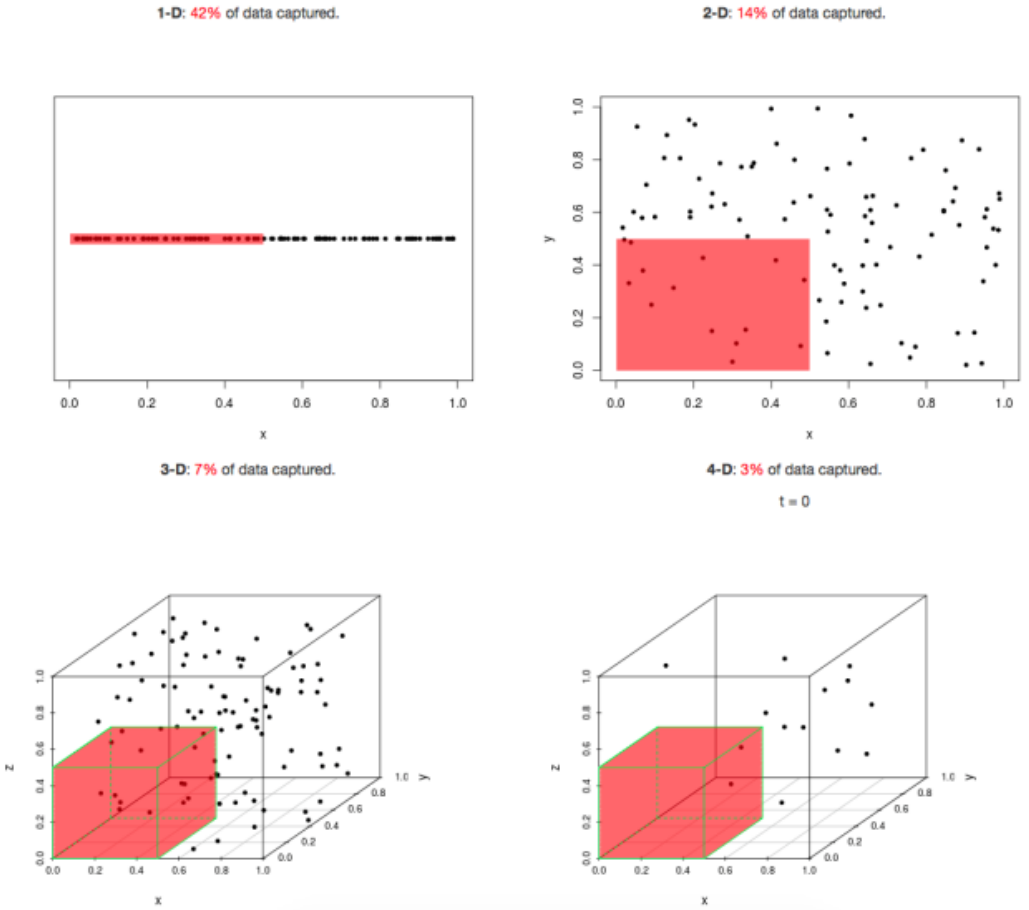
$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \text{ for any } i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

Week 5 neural net review quiz (in Python)

- Load heart.disease clean data csv with Pandas
  - Data under week 2
  - Use starter code (week5_neural_net_review_quiz.ipynb)
- Split full dataset into train/test
- Set up a neural net with some dense layers
- Use ‘binary_crossentropy’ as a loss function
- Report metrics (accuracy, etc) on train/test
- Drop under assignments in “Week 5 review quiz: ANN/SVM”

Curse of dimensionality



- https://eranraviv.com/curse-of-dimensionality/
- https://en.wikipedia.org/wiki/Curse_of_dimensionality
- https://stats.stackexchange.com/questions/169156/explain-curse-of-dimensionality-to-a-child
- https://www.quora.com/What-is-the-curse-of-dimensionality

Curse of dimensionality

- As we add more dimensions (features), everything becomes more difficult
  - Samples are more sparse/spaced out, and we need more to get a good distribution throughout the spaces
- Example: looking for a penny in a 100-yard line
  - Looking for the penny in 100 yards ^2
  - In 100 yards ^3
  - It takes exponentially longer to search the space for the penny
- If we have 100 samples in 1d, we have a decent representation of their distribution
  - In 2d, gets a lot more sparse
  - In 3d, quite sparse, and even worse the higher we go
  - Makes it hard to fit a meaningful ML model to the data

Unsupervised learning

- Why would we do this?

Unsupervised learning

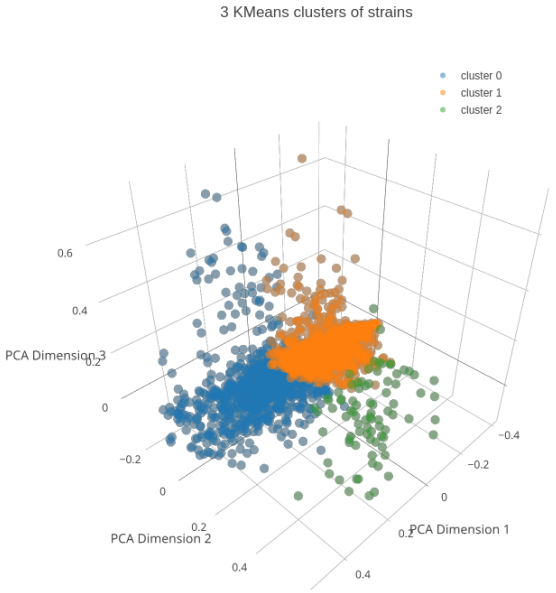
- Why would we do this?
  - If we don’t have target labels, but we want to understand the structure of our data
- Examples?

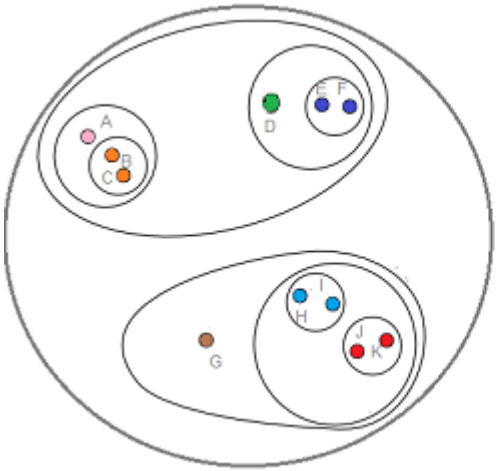
Unsupervised learning

- Why would we do this?
  - If we don’t have target labels, but we want to understand the structure of our data
- Examples?
  - Youtube videos, news articles, websites – group into topics
  - Medical data – find patients with similar conditions to understand/fix their problems
  - Any text data – group into topics

Clustering

- Two primary clustering techniques (but there are others):
  - Kmeans
  - Hierarchical (dendrograms)





Kmeans algorithm

- Similar to KNN
- Most common algo: Lloyd's algorithm (alternatives)
- **1. We set k** (# of clusters, a hyperparameter)
- **2. Pick centers of clusters** as points (kmeans++ picks centers that are distributed)
- **3. Calculate distance** from each point to each cluster center
- **4. Assign each point** to the closest cluster center
- **Repeat 3 and 4** until assignments no longer change

Kmeans runtime (Lloyd's algo)

- O(n * K * I * d)

- n : number of points
- K : number of clusters
- I : number of iterations
- d : number of attributes

- How could we reduce the runtime, not removing any training points?

Kmeans runtime (Lloyd's algo)

- O(n * K * I * d)

- n : number of points
- K : number of clusters
- I : number of iterations
- d : number of attributes

- How could we reduce the runtime, not removing any training points?
  - Use PCA to reduce the dimensions of the features (d)

Should we do cross-validation with clustering?

Should we do cross-validation with clustering?

- We can do this if we want
  - https://arxiv.org/abs/0909.3052
- Train the clustering algo on train data, then calculate stats on train and validation sets

Evaluating k-means performance

- We want to optimize k
- Scree (elbow) plot of within sum of squares (WSS)
- Silhoutte score
- Calinski and Harabaz score
- Many other methods (link for R)
  - link for Python

Scree plots

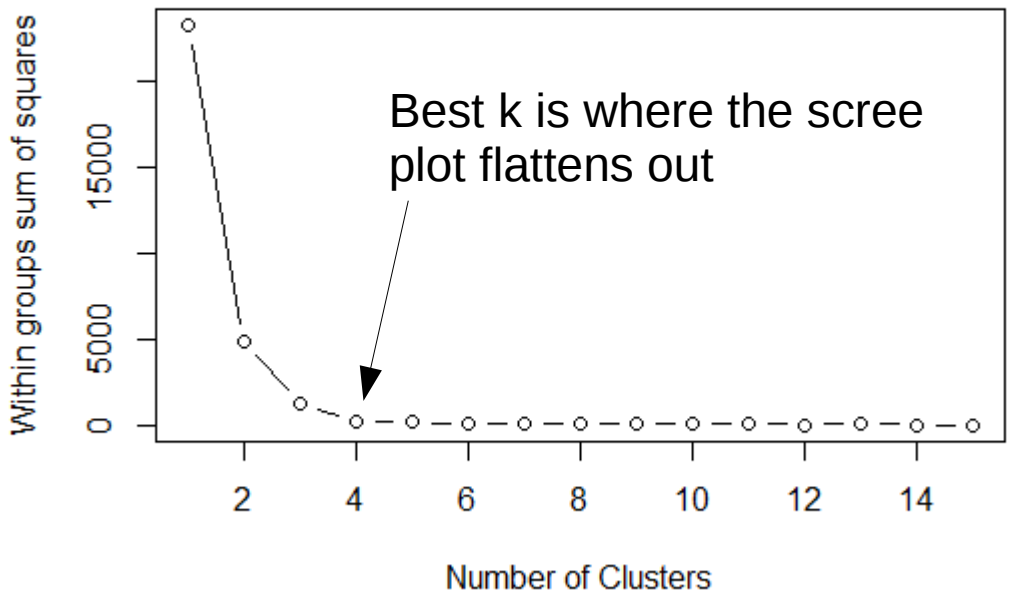
- Scree is broken rock at the bottom of crags or mountain cliffs
- Called a “scree plot” because it looks like this

https://en.wikipedia.org/wiki/Scree#/media/File:Yamnuska_bottom_cliff.jpg

Scree plot

- The WSS is the sum of squared distances between the points and their cluster centers

$$WSS = \sum_{i=1}^{k} \sum_{x \in S_i} distance\left(x, cluster.center\right)^2$$



https://stackoverflow.com/a/15376462/4549682

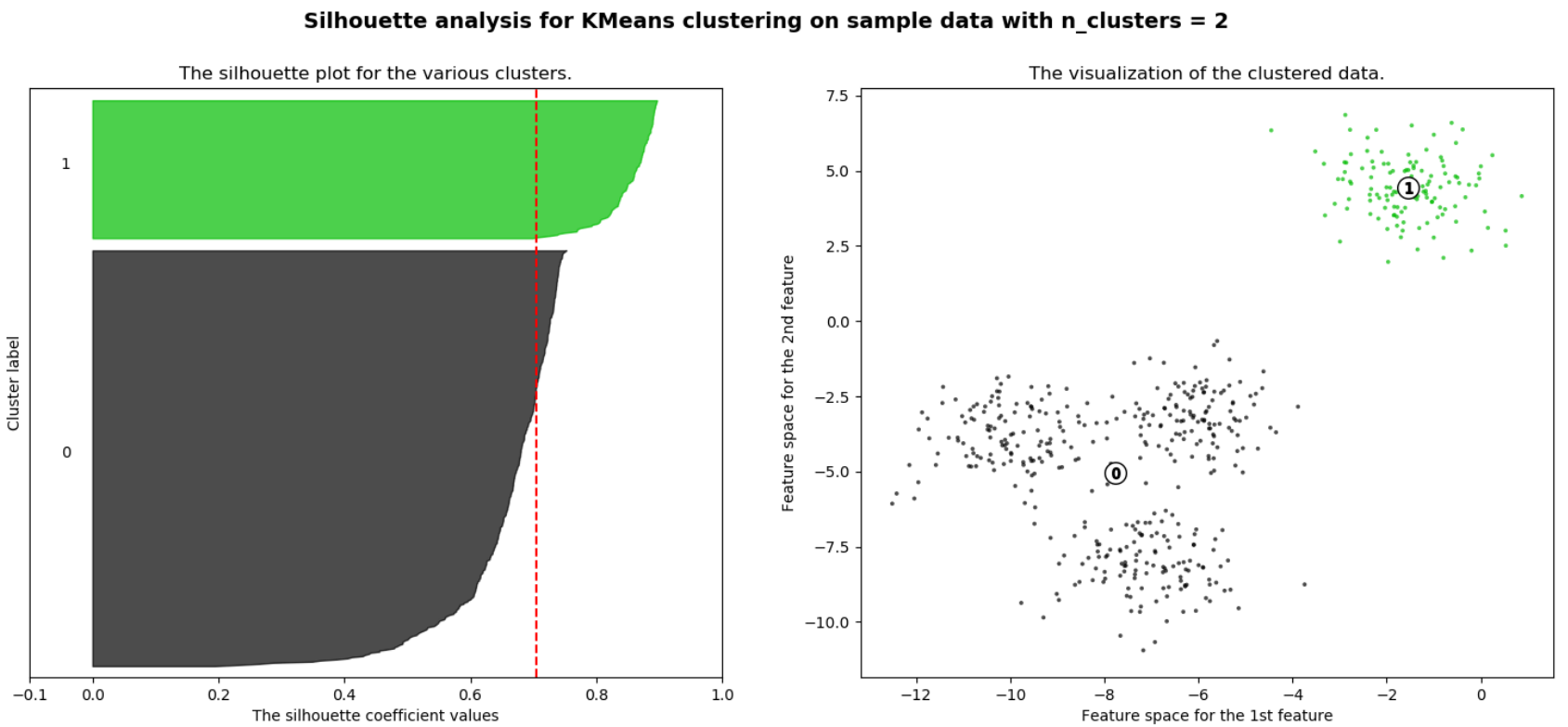
Silhouette score

- Range of [-1, 1] (-1 is worst, 1 is best)
- Silhouette score: mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b)
- b is the distance between a sample and the nearest cluster that the sample is not a part of
- If clusters are tightly packed and spaced out (good cluster structure), a is small, b is large, and the silhouette score will be close to 1 (approximately b/b).
- Worse clustering means silhouette score moves towards -1

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

Silhouette score

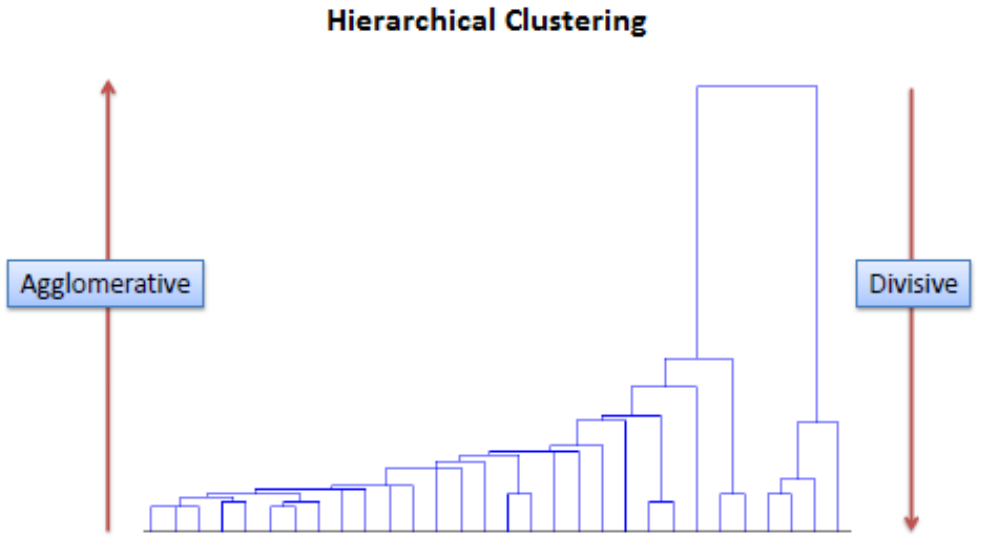
- Can calculate for each point, and get the average of all points



http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Hierarchical clustering

- End up with a dendrogram – horizontal lines are clusters, y-axis is distance
- Can start as single points (agglomerative) or as one cluster (divisive)

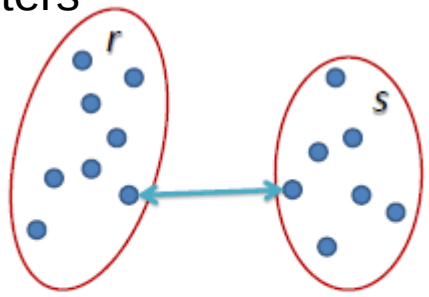


http://www.saedsayad.com/clustering_hierarchical.htm

Agglomerative clustering

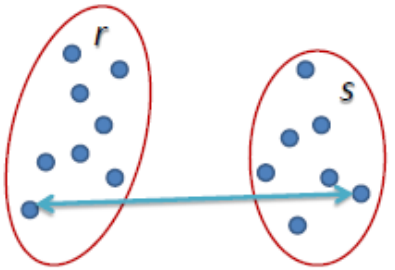
- Calculate distance between all points
- Nearest 2 points become a cluster
- Repeat, treating clusters as points
- Cluster distance can be single, complete, or average linkage

Single linkage – closest points in clusters



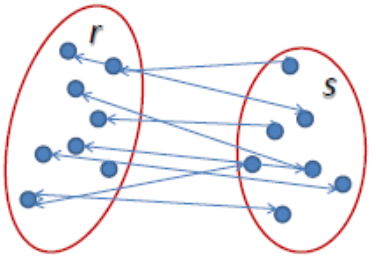
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete linkage – furthest points in clusters



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average linkage – avg distance between clutsers



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

http://www.saedsayad.com/clustering_hierarchical.htm

How do we pick number of clusters with HCA?

- No standard method, ongoing field of research
- paper from 2017 proposed a method
  - Compact-separate proportion (CSP) index
  - (sd – cd) / (sd + cd), where sd is ***inter***cluster separation, and cd is ***intra***cluster compactness
  - Complex math (minimum spanning trees with Dijkstra's or Kruskal's algorithm, etc)
- We can cut the tree to get a specified number of groups
- HCA and kmeans are more exploratory than objective

Example exercise (50% of exercise grade)

- clustering.demo.exercise.R file under week 6
- Student performance dataset from UCI, student-mat.csv under week 6
- Cluster students to see if there are some natural groupings
- Pair exercise:
  - Try other hierarchical linkages
  - Use the kmeans/hierarchical clusters to compare the groups of students and find commonalities
  - If you have time, use PCA to plot the clusters

Assignment (other 50% of exercise grade)

- Use kmeans and/or HCA on another dataset of your choice
- If using kmeans, find optimum k, if HCA, pick a number of clusters you think seems to work best – support with some sort of evidence/argument
- Explore some of the differences in the properties of the clusters
- Make a scatter plot, with points colored by clusters
- Write some interpretation of your results
- Post to the “week 6 cluster assignment” discussion
- Possible datasets: https://archive.ics.uci.edu/ml/datasets.html