Lesson 15 – Unsupervised Learning

**Questions for Mentor:**

**Overview of Unsupervised Learning:**

* No labels in our training data
* Image recognition is unsupervised learning
* Euclidean & Manhattan distances
  + Euclidian distance
    - based on Pythagorean scale (triangles AC^2 = AB^2 + BC^2)
    - Same formula applies for all dimensions
  + Manhattan distance
    - |(x2 – x1)| + |(y2 – y1)|
    - Name Manhattan because Manhattan is on grid system – can remember as distance in blocks
* K-Means clustering
  + K cluster centers
  + Establishes decision boundary based on which center is closer to point
  + Reset centers, based on learning will be much closer to center
  + Cluster centers guaranteed to converge
    - But depends on initialization (where centers are initially placed)
  + # of clusters is important
  + Sensitive to outliers
    - Use median instead of mean for updates
  + How to find K?
    - Choose K such that increasing it does not model data much better
    - Knee or elbow method can visualize this
* Unsupervised Learning in Python
  + Dimension = number of features
  + Can’t see 4-D, unsupervised learning helps us gain knowledge about it though
  + KMeans(n\_clusters=K)
  + Evaluate clustering
    - Cross tabulations
    - Pandas library
    - Pd.crosstab(df[0], df[1])
      * This stategy must know there are data broken out (i.e. species)
    - Inertia measures clustering quality
      * Distance from data point to centroid
      * Create model and fit to data
      * Use inertia\_ attribute to get inertia value
        + Can plot these values in a for loop to get elbow plot
  + StandardScaler
    - in kmeans: feature variance = feature influence
    - transforms each feature to have mean 0 and variance 1
    - scaler = StandardScaler
    - scaler.fit()
    - var = scaler.transform()
    - use pipeline to combine multiple steps
      * make\_pipeline(scaler, kmeans)
      * pipeline.fit(samples)
    - preprocessing step
  + Normalizer() rescales each sample (row) instead of each feature (column) like StandardScaler
  + How do you specify that you’ve converged?
    - Find expsilon and once epsilon is small, you can stop
  + Getting rid of K?
    - Mean shift
      * Put window around each point
      * Compute means of each point in frame
      * Shift window to mean
      * Repeat until convergence
    - Mean shift method can handle arbitrary sized clusters
    - Doesn’t need to know # of clusters
    - Robust initialization
    - Needs bandwith parameter (window size)
    - Computationally expensive
  + Hierarchical clustering
    - Produces complete structure
      * Whole tree
    - No single number of clusters
    - Cluster linkage
      * Single linkage
        + Minimum distance between closest points in diff clusters
      * Complete linkage
        + Maximum distance between furthest points in diff clusters – take smaller of distances and that is complete linkage
    - Evaluation criteria is hard
      * Based on expert knowledge
      * Debatable for real data
      * Hidden unknown structures could be present
      * Do we even want to just reproduce known structure?
    - Clustering can be used to assign labels and transform the problem into a supervised learning problem
  + t-SNE – 2 dimensional view of dataset
  + organizes data points into hierarchies (animals and plants example)
  + Hierarchical clusetering
    - Starts with all separate clusters
    - Each step – two closest clusters merged
    - Continue until all countries in single cluster
    - This is agglomerative hierarchical clustering
    - Linkage() function will perform hierarchical clustering
    - Height on dendrogram = distance between merging clusters
    - Different linkage methods give different hierarchical clusters
  + T-SNE
    - Maps samples to 2 or 3 dimensional space so they can be visualized
    - T-SNE only has fit\_transform() method, not separate
      * Cant remap to new samples, must start over every time
    - Learning rate can be changed based on dataset
      * Try values b/w 50 and 200
    - Axes of T-SNE plot don’t have meaning
  + Dimension reduction
    - More efficient storage and computation
    - Remove less informative ‘noise’ features
    - Principal component analysis
      * Aligns data with axes
      * Shifts data so they have mean 0
      * Learns principal components of the data
      * Aligns these principal components to the axes
      * Intrinsic dimension = # of features needed to approximate the dataset, # of PCA features with significant variance
      * PCA identifies intrinsic dimension when samples have any # of features
      * Intrinsic dimension can be ambiguous
      * N\_components= will reduce dimensions to a specified number of components
    - Non-negative matrix factorization
      * Dimension reduction technique
      * NMF are interpretable
      * Samples must be > 0
      * Must always specify number of components (n\_components)
      * Same fit/transform as with PCA
      * NMF learns interpretable parts
* When to use different clustering methods
  + Density based clustering methods
    - Instead of assuming every point is part of cluster, we only look at points that are tightly packed
    - We don’t need to specify number of clusters for DBC
* Cosine Similarity
  + Metric used to measure how similar documents are irrespective of their size
  + Measures cosine angle b/w two vectors in multi-dim space
* One stop shop for Principal Component Analysis
  + Want to reduce dimension of feature space
  + Feature extraction combines existing features in a specific way and only keeps the ‘new’ features that help predict