Intro to Clustering; K-Means

Week	THURS. Week 8
Assignment Due	HW4 Due — (Linear Regression)
Assignment Done	☑
Due Date	@March 3, 2023 11:59 PM
Notes Done	☑

Presentation

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1998a77a-8dda-4fd2-bca0-be81384d3edf/L2-FlatClustering.pdf

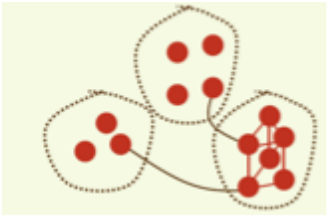
Class Notes

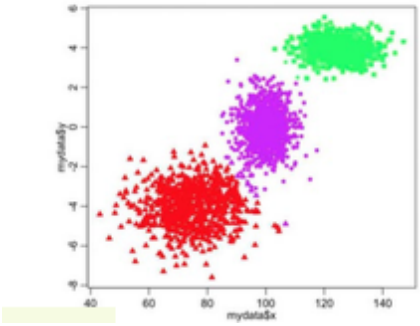
Clustering

Over the next few weeks, we'll review ***unsupervised learning***:

- So far, we've been primarly working with *supervised data*, where for each observation $\mathbf{x}$, we have an associated target value y → together, we have a **supervised dataset**: $\{X_i, Y_i\}_{i=1}^N$
- However, there are a lot of problems where we don't have a target value y → called **unsupervised learning** problems
    - These problems tend to be "tougher" since we don't know the target.
        - Therefore it's difficult to evaluate the *quality* of the result unlike what we've done in supervised learning with minimizing the log likelihood for example.

The most common type of unsupervised learning is **clustering**, where we attempt to learn underlying patterns from the data:





- We aren't necessarily enumerating the classes that data is from, but instead saying that possibly a group of data is similar to one another while dissimilar to others.
    - Our target is that things in a group, called a **cluster**, are similar → similarity is referred to as the ***intra-cluster similarity*** vs. similarity of observations between cluster needing to be dissimilar/low (***inter-cluster similarity***)
- This goes to the idea of similarity/distance, which we can think of in several ways:
    - Similarity: Gaussian, Cosine
    - Distance: L2 (Euclidean), L1 (Manhattan)
- We could also say, instead of similarity, that we want to minimize the distance between clusters.
    - The two most common distance similarities are the **Euclidean** and **Manhattan distance**.
- Difficulties in clusters that arise is how many groups there should be:
    - Given data → without prior knowledge, it may be difficult to see how many groups there *should be.*

        - Note: In certain circumstances/application, we may have a better idea of how many clusters should exist.

- Another difficulty that exists is how must be evaluate cluster quality given that the data we're given is *unsupervised*.
    - While we know we want ***intra-cluster similarity*** to be high and ***inter-cluster simliarity*** to be low, but this doesn't mean that all things in a cluster should be together as there is no supervision at all.

Types of Clustering

Clustering can be separated into two types:

- **Hard clustering** - more common & easier to do
    - Each observation belongs to *exactly* one cluster.
- **Soft clustering** - an observation can belong to more than one cluster
    - And/or belong to clusters with *some probability*

Clustering Algorithms

There is a further division of clutering in terms of algorithms:

- **Flat algorithms** → *flat* in that you always have the same number of clusters
    - Know number of clusters ahead of time
    - Typically start with a some sort of initial guess in clustering groups generated → random clustering/model
        - This model of clustering is refined iteratively.
- **Hierarchical algorithm** → ask what the clustering be with 1 cluster….2 clusters… and so on…building a hierarchy of clustering options; effectively exploring different numbers of clusters
    - Has two types that denote how you begin:
        - ***Bottom-up***, also referred to as ***agglomerative***
            - Start off with everything being its own cluster (start with N observation, have N clusters) and then merge a cluster together to then have $N - 1$ clusters.

        - Continues this clustering process until we end up with a single cluster.
    - ***Top-down***, also referred to as ***divisive/segmenting-hierarchy***
        - Starts off with everything being in *one* cluster, then divide it into 2 clusters.
            - Then one of the clusters divided in half and divide it in half to get 3 total clusters.

K-Means

The algorithm we'll focus on today, core of programming for homework #5.

- **K-means** is a ***hard*** and ***flat*** algorithm.
    - Must know number of clusters ahead of time due to being a flat algorithm.
    - Assumes features are continuous in nature.
- In **k-means**, we have **reference vectors**, where for each cluster, we'll have a representative observation.
    - We can think of this as the average of observations a part of a cluster.
    - This reference vector isn't necessarily a real observation, but one we compute from the data.
        - As we iterate through the **k-means algorithm**, each observation is associated with one and *only one* cluster and therefore will be associated with one and *only one* reference vector.
    - We want to associate each observation to a reference vector that they are ***most similar to*** or conversely, the distance by associating observation to the reference vector they're closest too.
        - For visualization purposes, using distance is easier to understand intuitively

Pseudocode

Note: **K-means** ends up being another iterative algorithm where we'll update model as we go until we see that the clustering converges or nothing changes.

1. Start with an initial set of reference vectors $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_k}\}$ (somehow)

    a. If we'll have k different clusters, we need to have k different reference vectors.

2. Until we reach some type of convergence, or other stopping condition (such as when nothing much changes in our model):
    a. For each observation x:
        i. Compare x to each reference vector, and associate x with the reference vector it is most similar to/closest to.
    b. Upon completing this comparison for *all observations*, we *update* the reference vectors to be the average of the observations that are associated with that cluster.
        i. If C_i is the set of observations associated with reference vector $\mathbf{a_i}$, then:
            1. $\mathbf{a_i} \rightarrow \frac{1}{|C_i|} \Sigma_{x \in C_i} x$

K-means is an example of an ***expectation-maximization*** algorithm:

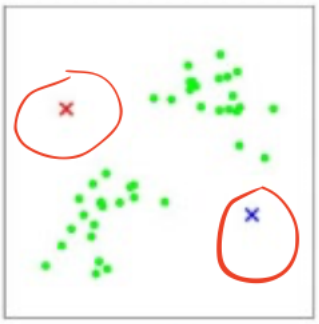
- We have a model and use it to make an expectation, in this case to associate different observations with reference vectors (the **expectation step**).
- The **maximization step** comes in when trying to update the model to better fit the data through moving reference vectors to be at the mean of the observations associated with it.

Visual Example

Imagine having several points in 2D space as our dataset:

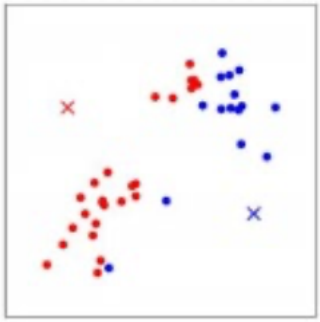
(a)

Decide where reference vectors are based on the expectation that we'd like 2 clusters:



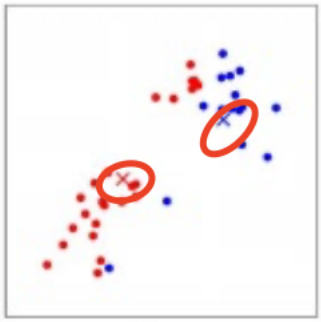
(b)

We then take all observations, compare them to the reference vectors (using Euclidean distance), and associate each observations with one reference vector.



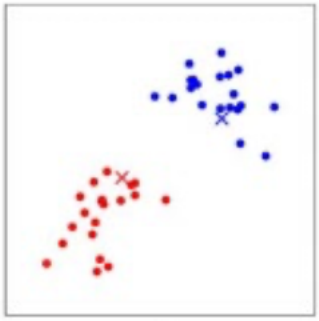
(c)

Now, we update our model to better align with the data:



(d)

Now, we'll do another **expectation** step and associate our observations with the new reference vector locations. We can note here that some points have moved clusters.



(e)

Now, we move our reference vectors *again*.

(f)

We continue this process until eventually none of the osbervations change their cluster association.

K-means Design Considerations

A few design considerations exist with K-means:

- What should k be?
  - Application likely dictate this or we may need to do additional exploration for a good value of k
- What is the logical similarity/distance function to use?
  - Nature of the features likely dictates this:
    - Typically if doing distance, we use Euclidean or Manhattan distance.

        - If using similarity, then a cosine similarity may be useful.
- Where should the reference vectors start?
    - To be covered in future slides.
- What are some termination criteria?
    - To be covered in future slides.

Initial Reference Vector Choice

You are NOT guaranteed to get the same result with different initializations of reference vectors.

- Bootstrapping the algorithm with different init reference vectors can result in different final clusters.
- It may be that some choices of init reference vectors can result in poor convergence, or convergence to sub-optimal clustering.
- Ideas for initial reference vector choice:
    - Use some prior knowledge
        - Look at distribution of features → somehow try to place reference vectors in locations that are representative of the spread of the data
    - Initialize with the results of another method, such as hierarchical clustering, which can give us clustering at level k
    - Try out multiple starting points generated from random locations t amount of times

Termination Criteria

It is nice to have a worst-case scenario, where we've maxxed out the time we're willing to work on convergence OR the best-case scenario, where we've found the clustering that we'd like:

- Several possiblities exist:
    - A fixed number of iterations → worst-case
    - *Between iterations* → ideally occurs from convergence being reached

- Cluster assignments don't change
- Reference vectors don't change (by much)

Proof of K-Means

Question: Why is the mean of the obseravtions in a cluster a good reference vector for it?

- **K-means** is using the squared-error type of formulation, as in linear regression, in trying to figure out the ideal reference vector.
  - it is based on using the sum of the square of the distances to measure the "goodness" of our solution.
- Let $\mathbf{a_i}$ be the reference vector for cluster i
  - Then we can create an objective function:

$$J = \sum_{x \in C_i} (x - a_i)^2$$

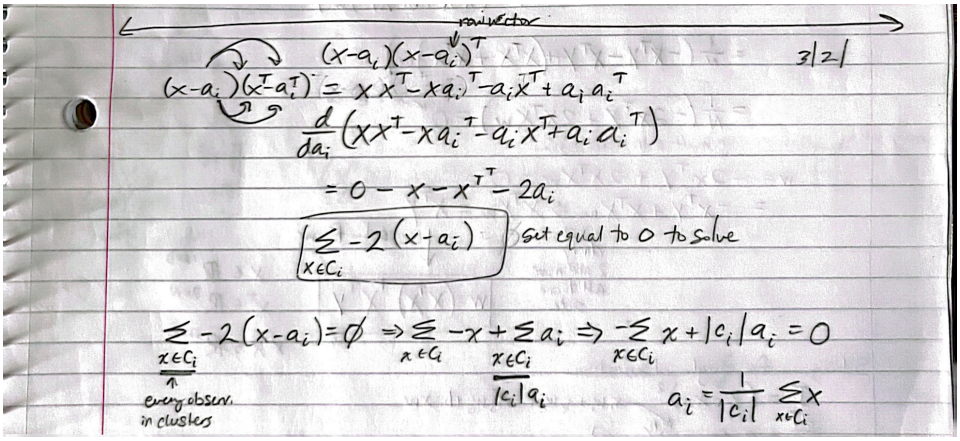
    - Tells us "how good is $\mathbf{a_i}$ at generalizing this cluster"?
      - We compute the square distance of each observation to the reference vector and sum these, giving us an objective function measurement of how good this reference vector is at representing the cluster.
- Since $\mathbf{x}$ and $\mathbf{a_i}$ are vector, we can write this as:

$$J = \sum_{x \in C_i} (x - a_i)(x - a_i)^T$$

  - $\mathbf{x} - \mathbf{a_i}$ gives us a row vector, where it's transpose will be a column vector.
    - In multiplying these together, we get the sum of the squares of its elements.
      - This is effectively the squared Euclidean distance.

In our proof, we aren't trying to find what $\mathbf{a_i}$ is yet, but rather what value $\mathbf{a_i}$ should have that will minimize the overall accumulated square distance in our cluster.

- We'll take the derivative here with respect to $\mathbf{a_i}$ after foiling out the initial objective function:



- We set this value that we're solving for to 0 to attempt to find the optimal value for $\mathbf{a_i}$.
    - Now, we're left with:

$$\sum_{\boldsymbol{x \in C_i}} -2(\boldsymbol{x - a_i}) = 0$$

        - We can divide out the -2 since it's a scalar value.
        - We can note in this summation we're iterating over all x observations in cluster i.
            - In noting this, we can see that $\mathbf{a_i}$ is independent of that summation over x observations.
        - Then, we can separate the summations as we see above.
            - If we say that $|C_i|$ is the number of observations in cluster i, then the summation for that independent term that doesn't have an x is then

transformed into $|C_i|$, giving us $|C_i|a_i$.

- We can move around the negative next to the summation with x and continue to solve for $\mathbf{a_i}$, moving that summation over with x and dividing by the elements in cluster i.

- This result will now give us the value of $\mathbf{a_i}$ that will minimize the squared distance between it and all the elements in cluster i.
  - The result is literally the mean of all observations in cluster i.
- This shows that by assigning the reference vector of cluster i to be the mean of the observations in cluster i, we are minimizing the squared error between the reference vector and all other elements in that cluster.

Weaknesses of k-Means

K-means is a common and intutitive algorithm to use, but does have weaknesses:

- Its use only makes sense if the mean is a defined value (computing the mean between values makes sense).
  - For categorical nominal data, this wouldn't make sense.
    - For this data type, an alternative is used instead called **k-mode**.
      - In this method, all members of a group are observed and the reference vector will take the mode of each feature of a group.
- An inherent drawback of the method is that the user needs to specify k.
- K-means is also sensitive to outliers, which may be a result of errors in data recording or some special data points with very different values.
  - An outlier will pull the reference vector towards it.
  - If you note that there are outliers in your data, an alternative to k-means is **k-medoids**.
    - In this method, the median of each feature is used instead of the mean of each feature since medians are not susceptible to variance from outliers as means are.

K-means questions:

- If you are in high dimensional space, coverage of that space is sparse.
    - Groups within a cluster would be more spread out.
    - To visualize things, we'd need to get our data back into 3 dimensions over several.

The scale of different features does affect the nature of k-means, so it would motivate for the standardization of the data.

Next Tuesday will look at soft clustering via mixture models and next Thursday will look at hierarchical models that can be used to bootstrap k-means and help find a k value.

Homework #5 Notes