Learning Reflection Week 6

Created	@August 6, 2024 8:15 PM

Summary

This week, we covered kernel methods, locality sensitive hashing, and clustering. LSH involves using random lines to group objects with high cosine angle similarity. Nearby adjacent bins can be checked if time/computational power allows for it. The curse of dimensionality is the phenomenon of how higher dimensions cause increased data sparsity, which can then lead to certain issues. Clustering is an unsupervised learning method with unlabeled datasets that detects patterns in unlabeled data. K-means++ offers improved initialization so that it is more likely to converge to a local optima point and hierarchical clustering uses dendrograms.

Uncertainties

- How would KNN and K-means be applied to the imputation of epigenetic data? I've been trying to do some research on this topic
- How exactly is the K-means++ algorithm different from the k-means++ algorithm
- How is K-means different from KNN and LSH?
- In general, I just don't get the differences between the various models.

Kernel Methods; Locality Sensitive Hashing + Clustering

Locality Sensitive Hashing (LSH)

How to choose lines that divide bins?

- Randomly

    - choose a slope between 0 and 90 degrees because 2 objects that have a high cosine similarity will have a high chance of being grouped together

    - You can look at adjacent bins to check if something did get micategorized

Bin indexing

- 0 means that the bin is above that particular line
- 1 means that the bin is below that particular line
- the line is based on the 1-indexed index value
- algorithm
    - draw h lines randomly
    - find the bin index of all points
    - figure out which point you want to find a match for and find its bin index
    - do exact nearest neighbor search only in that bin for a similarity match
    - look at nearby bins if you have time

Optimization/fine-tuning

- We can look at more bins to overcome the tradeoff that comes with better efficiency

Pick a random hyperplane that separates the points

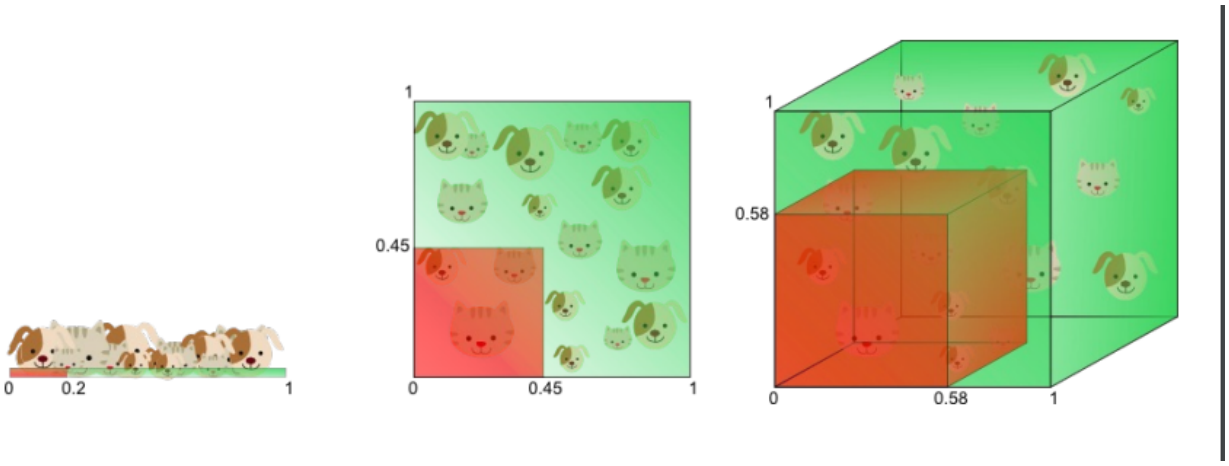
h is the number of lines

h=ln(# of dimensions)

Curse of Dimensionality

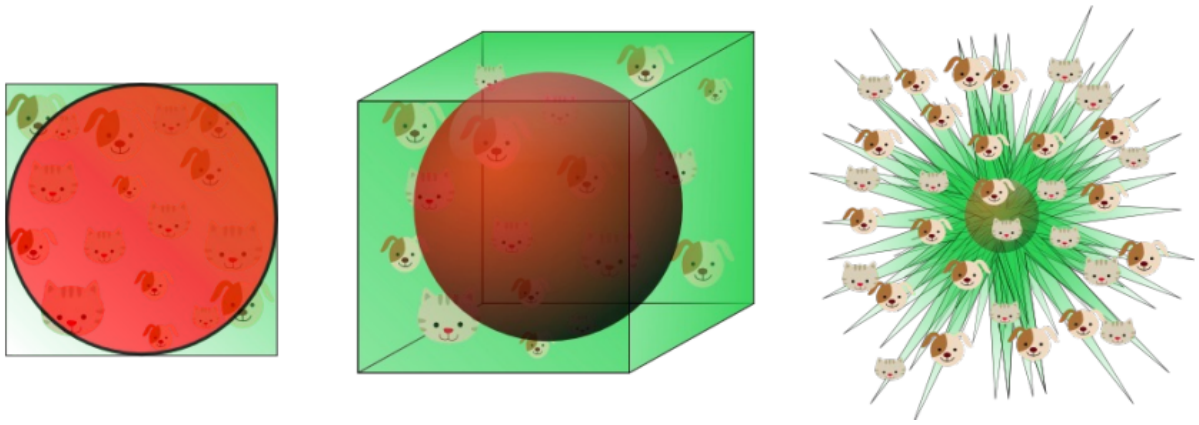
Higher Dimensions

The sparsity of the data increases as the number of dimensions increases



So you need more data to reduce the sparsity (more space, so you need more data points to fill that space)

Most of the points become clustered at the corners of the shape as the number of dimensions increases



so your data becomes more sparse

and also your nearest neighbors start becoming your farthest neighbors

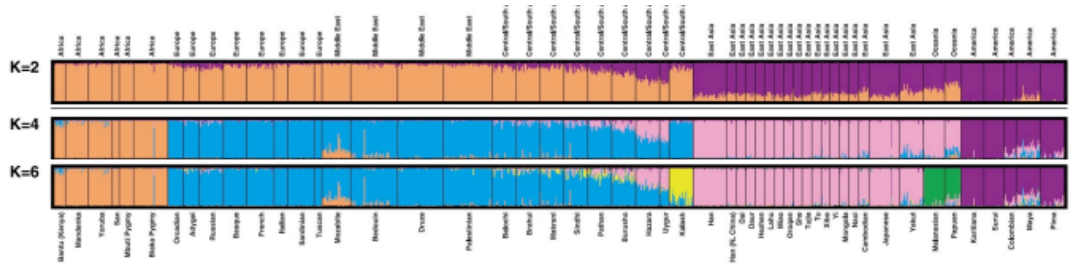
Nearest neighbors start becoming farther and farther because they are on different dimensions

Lasso Regularization can be used to reduce the number of dimensions

Clustering Overview

- unsupervised learning

    - CV, overfitting, bias-variance tradeoff, accuracy, error, etc don’t apply because you don’t have any labeled inputs or outputs
    - clustering gene sequences



    - detects patterns in data that is not labeled
    - quality metric in the ML pipeline is harder to evaluate for unsupervised learning because the outputs aren’t labeled/set in stone
- Document retrieval case study
    - given that someone read a sports article, what similar articles would you recommend
    - if the data is labeded, you could use multi class classification
    - but usually there isnt a define boundary/difference between different categories or they might not be labeled
    - clustering finds groups in a dataset
    - a cluster is defined by
        - centroid: location of the center
        - spread: shape and size of the data
    - 2 parts
        - find the clusters
        - assign each example to a cluster based on how close it is
        - closer distance means stronger similarity

K-Means Clustering Algorithm

- algorithm to cluster different points together
- algorithm
    - first randomly initialize h points
    - map each point to the closest point
    - place the point in the average of the points in that cluster
    - repeat until convergence
- results heavily depend on the initialization
- k-means++ is more smart and is more likely to reach a local optima

Hierarchical Clustering

- We use clustering to group together unlabeled datasets
    - clusters are defined with their centroid and spread
    - the k-means algorithm initializes to random points and then continuously updates the centroids until it hopefully converges to some local optima
        - cons:
            - it doesn't work well for clusters with different spreads or sizes
- We can use the Mixture of Gaussians approach to mitigate this using the Expectation Maximization algorithm
    - this assigns probabalistic values to how likely a point is in a cluster

Hierarchical Clustering

- Uses the natural relationship between real-world entities (for example species diagram) to create clusters
- allows us to not pick how many clusters we want
- use dendrograms to visualize different granularities of clusters
- pros

    - any distance metric can be used
    - can model more complex cluster shapes by establishing relationships
- the goal of a hierarchical clustering model is to create dendrograms
- there are 2 types of models
    - Divisive (top down)
        - starts with one large cluster and then recursively divides those clusters until we have what we want
        - e.g. recursive k-means
    - Agglomerative (bottom up)
        - starts with a large number of clusters and then combines them until they are together in one big cluster
        - e.g. single linkage
        -
- Assessing performance for clustering algorithms
    - Don’t know
    - more distance between the outer boundaries of each cluster is better
- heterogeneity objective
    - the model is trying to minimize the distance between each of the points and the centroid
    - this is like the error metric
    - we are trying to minimize this value

Detecting outliers for k-means and hierarchical clustering

- if there are clusters with <2-3 datapoints that are far away from all of the other clusters