Learning Reflection Week 5

Created	@August 2, 2024 7:14 PM

Summary

This week we learned about convolutional neural networks, which are broadly layers in a NN that apply convolutions to an image in order to extract features from it. Additionally, we discussed how to calculate the precision and recall of a model, along with what a ROC curve is and what it can tell us. We also started our discussion on the nearest neighbor algorithm, with 1NN and kNN, which are used for similarity. Finally, we discussed how to define our distance metric and how to represent the words in a embedding vector.

Uncertainties

I don’t understand anything from slides 1-14: review

how does a computer create voronoi diagrams?

is it better to use kNN or logistic regression for the task of classification?

how are the weights of weighted distance calculated? are they hyperparameters?

Concepts

[paper notes]

In KNN, we return the k nearest documents in the corpus that have the smallest distance (closest) to the document x_i

We need to figure out how to find the distance between 2 documents as well as how to represent the documents

Document representation

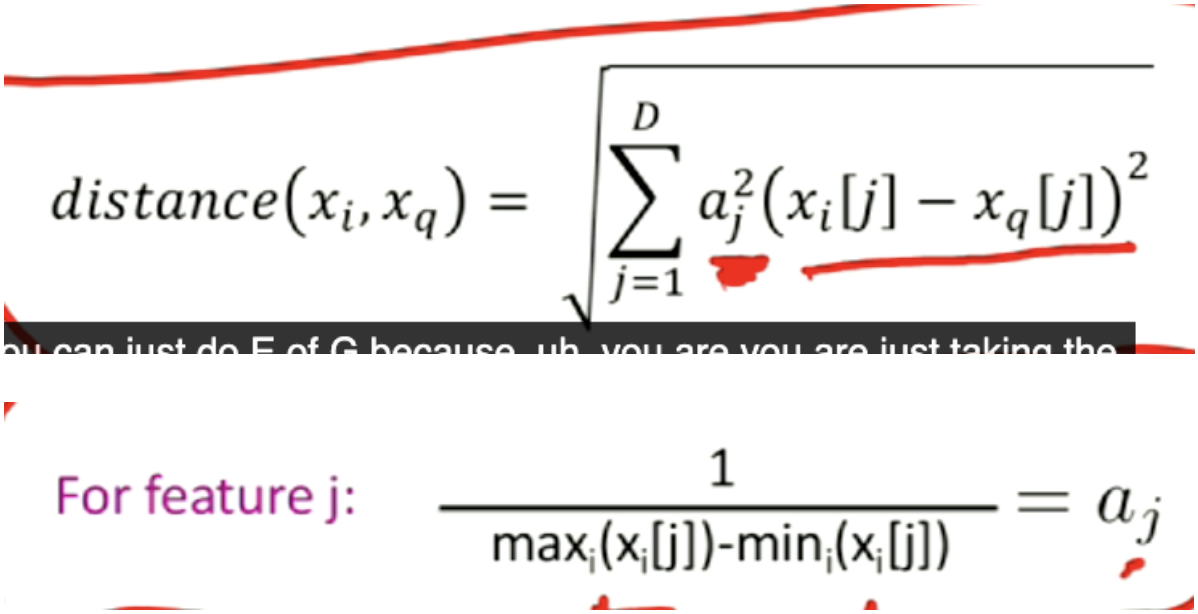
Represent the document as a vector

- Bag-of-words representation
    - you have a dictionary, where the keys are the words, and the values are the counts of the occurrences of that word
    - pros: easy to explain and easy to computer
    - cons: it can count unnecessary/meaningless words such as the, and, and, etc
    - this means that looking at the uncommon words is more useful/interesting since they are usually what uniquely define a text
- TF-IDF
    - our goal is to emphasize the important words
    - the important words are those that are more unique to one document compared to another
    - Term Frequency (TF)
        - word counts dictionary
    - Inverse Document Frequency
        - log(#docs/(1+#docs that have that word))
        - this is the inverse document frequency
        - For every word, do TF * IDF
        - more unique words will have a larger TF-IDF value

Ways of measuring distance

- Euclidean distance (L2 Norm): distance formula (sum of the squares of the distances)
    - smooth voronoi diagram boundaries
- Manhattan distance (L1 norm): sum of the absolute values of the distances
    - jagged voronoi diagram boundaries
- Weighted distances

- put more weight on the dimension that doesn't change as much as the other ones9for example weight kilometers more over seconds)
- less weight to the data that is more spread out
- weighted euclidean distance formula
-



- Similarity
  - NOT a measure of distance
  - higher similarity number is better
  - Similarity metrics:
    - Natural similarity
      - dot product of 2 vectors (multiplying each side by side and then summing it up)
      - helps us see how similar the magnitudes are
    - Cosine similarity
      - take the cosine of the angle between the 2 vectors
      - dot product of the 2 vectors / euclidean distance magnitude

        - this helps us see how similar the directions of the 2 vectors are
        - very efficient for sparse vectors since the dot product is more efficient (mostly zeros)
        - distance = 1 - similarity
    - Jaccard similarity
        - compares the overlap of words between the 2 texts
        -

$$J(Doc_i, Doc_j) = \frac{|Doc_i \cap Doc_j|}{|Doc_i \cup Doc_j|}$$

- Normalization of embeddings
    - normalization isn't good for when you have 2 documents of differing sizes
    - maximum cap normalization: cap the maximum word count
- Weighted KNN
    - put more emphasis on closer points than farther points
- Kernel regression
    - use a kernel to weight all training points
- Efficient Nearest Neighbors algorithm
    - no training required
    - however it has an O(n) runtime
    - you can approximate the nearest neighbor instead of actually finding the real one which will improve efficiency

Locality Sensitive Hashing (LSH)

Make bins and choose only the points nearest to that point in that bin

sometimes if its efficient you can look at nearby bins

Review slides 1-14

Pick a random hyperplane that separates the points

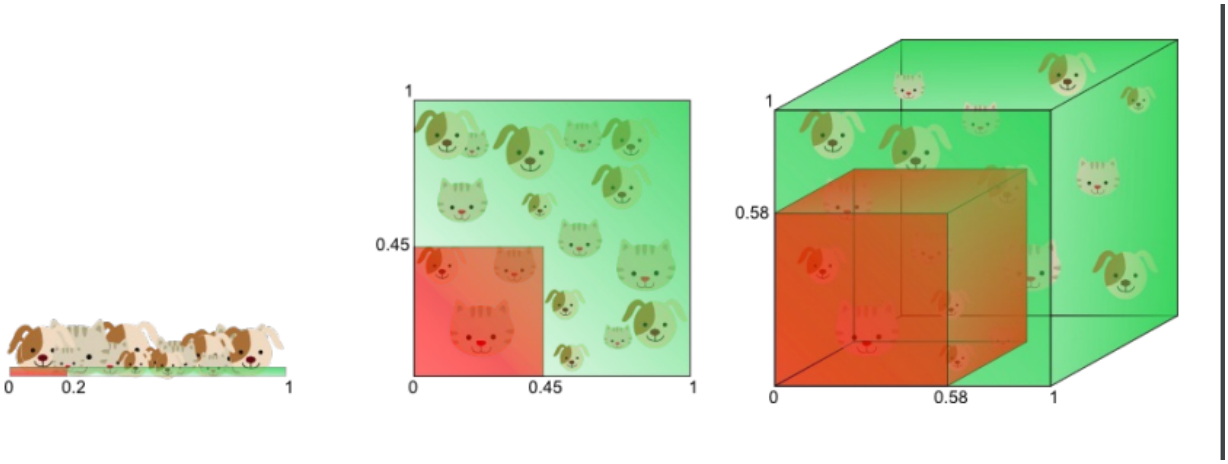
h is the number of lines

h=ln(# of dimensions)

Curse of Dimensionality

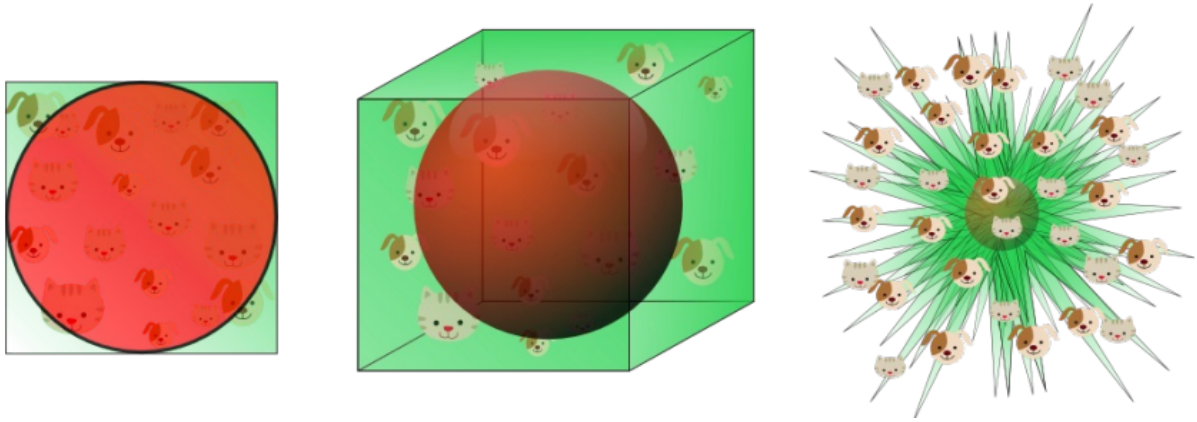
Higher Dimensions

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

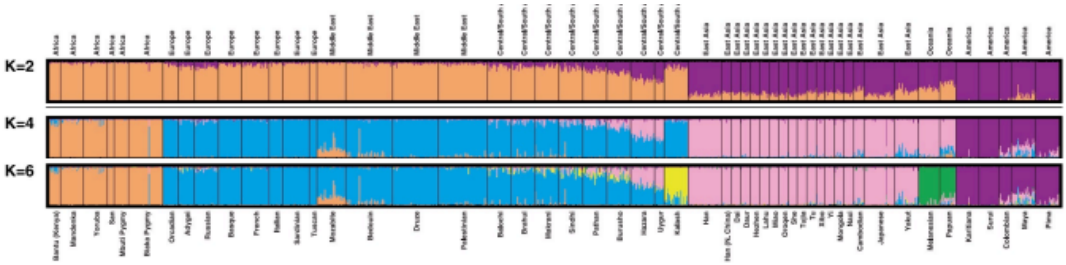


Nearest neighbors start becoming farther and farther because they are on different dimensions

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