Unit 2: Problem Setup and Nearest Neighbors

Problem Setup

1. **All model is wrong but useful to make predictions**.
2. **Intercept** tells you featureless version of the model that just predicts the mean. It is the **baseline** for linear model.
3. **R^2 t**ells you what % is explained by the model.
    a. With more features (aka params), you can get higher R^2. For that reason, you need to use test set to double check that the model is a good model if you found that the R^2 is high on the training set.
4. What are the different data sets used for:
    a. Train with **training set**
    b. Look at results with **dev set** to analyze errors
        i. Also, you can use the dev set to choose hyperparameters: parameters you tweak that can affect the outcome
    c. Run once on the **test set** to ensure model is a good model
5. To separate the different data sets, you apply **cross validation** (split single set of data into training and test sets in many different way if you have a small data set).
    a. Types of cross validation:
        i. Jack-knife: splits data into training set and test set
        ii. Leave one out: use all of the data for training except for one
        iii. Randomized splits (recommended): jack knife split with random partitions, can include dev data

Nearest Neighbors [[Classification]]

Instance-based, nonparametric modeling

1. **Output differs based on the distance metrics you choose.**
2. What it does? Nearest neighbors model **predicts the same label as the nearest neighbor in the training data**.
3. How do you **select the k value** for number of groups?
    a. **You don't want k to be an even number** so you don't get stuck with an even vote; you don't want to feel divided between the groups
    b. Example:
        i. If num_group = 2, then k = 3
        ii. If num_group = 3, then k = 5
4. Different types of distance metrics:

a. Ln-norm: you care about the size of the difference

$$L^n(x_1, x_2) = \sqrt[n]{\sum_{i=1}^{\#\dim} |x_{l,i} - x_{2,i}|^n}$$

i. n = 0: Hamming distance (categorical)
Measure things that do not overlap between two vectors

ii. n = 1: Manhattan distance (numerical)
Basically it adds up the absolute value of the distance difference

iii. n = 2: Euclidean distance (numerical)
Compare any given point to all other points

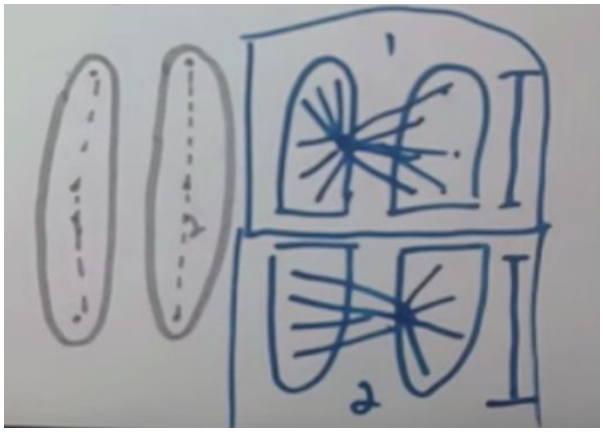
Pros	Cons
Symmetrical, spherical, treats all dimensions equally	Very sensitive to deviation even if it's just a single attribute

b. Cosine similarity: you don't care about the size of the difference but you care about the angle in between them

$$\frac{A \cdot B}{\|A\| \ \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} \left(A_i\right)^2} \times \sqrt{\sum_{i=1}^{n} \left(B_i\right)^2}}$$

Nearest Neighbors Pros and Cons list:

Pros	Cons
No training step. Therefore, training and testing are the same thing.	Load all training into memory. Therefore, the larger the dataset, the slower it runs.
It can be calculated in parallel.	You have to specify ahead of time the number of clusters. Therefore, you can draw wrong conclusions.
Simple and straightforward up to gigabyte of data.	If the data contain two clusters that are close to one another, k-means might incorrectly identify the cluster because it tries to find cluster that minimize the distance. For example, actual is specified by gray and

	k-means is specified by blue. 

To fix these problem, use affinity-based clustering, also known as spectral clustering.

Datasets

archive.ics.uci.edu/ml/datasets.html

References

- https://www.youtube.com/channel/UCUcpVoi5KkJmnE3bvEhHR0Q
- https://www.youtube.com/user/joshstarmer
- https://www.youtube.com/user/sentdex