K Nearest Neighbors Handout (*Machine Learning Seminar, December 1, 2022*)

Sommer Harris
harris.som@northeastern.edu

Niranjan Velraj
velraj.n@northeastern.edu

K Nearest Neighbors (KNN) is a supervised learning algorithm that can be used for classification or regression, though most commonly classification. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other ("Birds of a feather flock together"). Broadly speaking, the algorithm works by measuring the distance between one point and the others, in terms of the similarity of features (most similar will be the closest neighbors), and classifying the point the same as the majority of it's k neighbors. K is a positive integer, usually small.

KNN does not require the user to run training, but rather keeps memory of the data points and directly compares this information to classify the data point in question. KNN is often an effective classification algorithm in cases where the data points are highly labeled. Some common applications for KNN are recommendation systems, image classification, pattern recognition (e.g. in customer behavior), stock prediction, credit rating, and estimating missing values in data preparation.

Some advantages of the KNN algorithm are that it has a quick calculation time, is easy to implement, is useful for both regression and classification, and makes no assumptions about data. Some disadvantages of the KNN algorithm are that with large datasets, prediction will be slow, it requires lots of memory because it needs to store all the training data, and feature scaling is required for all dimensions.

1. Definitions

k - the number of neighbors we are looking at
distance - the distance between neighbors, which could be measured in many different ways. Some common distances are: Euclidean distance (Straight line distance), Manhattan distance (Difference of Cartesian coordinates), Hamming distance (Finds number of dissimilar points).
collaborative filtering - looks at similarity in user preferences to determine recommendations
content based filtering - looks at similarity in the items themselved to determine recommendations

2. Example Problem: NEUflix Movie Predictions

Let us now explore how the algorithm works by focusing on specific examples. In this problem, we build a small collaborative filtering movie recommendation system based on regression (part 1) and classification (part 2). We asked everyone to fill out a short survey, rating a list of movies, before our presentation. We will use the results from this data in the following questions. For each part, we will walk through an explanation of the algorithm, then pose a particular question to solve with our algorithm program.

3. Part 1: Regression

The following is an example of KNN regression. We have movie ratings for a bunch of different movies. In this example we will only use two movies (Spiderman and Batman) to predict a rating of the Superman movie because two features is easy to display on a graph, figure 1 (regression). In reality, we could have many more features included. Our sample data includes ratings on Spiderman, Batman, and Superman movies. For our user, we know what he thought of the Spiderman and Batman movies, but not what he thought of the Superman movie. We will use regression to solve this.

In this case, K=3. We will find the 3 closest neighbors on the Batman vs Spiderman graph, figure 1, and average their ratings of the Superman movie.

The three closest neighbors on the graph are marked in bold on the table, figure 2. We can also see their superman ratings in this table. To predict our user's rating for the superman movie, we can simply take the mean of the superman ratings for the three closest neighbors: (3.5 + 2.5 + 2)/3 = 2.67

Now we can solve the following regression problem using data collected from the class. Predict how Richard will rate the Batman vs. Superman movie, given his ratings on the other movies in the dataset. The answer should be a number rating from 1-5.

4. Part 2: Classification

The following is an example of KNN classification. We have movie ratings for a bunch of different movies. In this example we will only use two movies (Spiderman and Batman) to get our user classification because two features is easy to display on a graph, figure 1 (classification). In reality, we could have many more features included. Our sample data includes ratings on Spiderman movie, Batman movie, and whether or not they liked the Superman movie ('yes' or 'no'). We have info on our user about how they rated the various movies, but not on whether they liked the Superman movie. We can use KNN classification to solve this. In this case, K=3. The closest 3 neighbors did not like the Superman movie, so we will classify this user as a 'no' in terms of if we would recommend him the movie.

Now we can solve the following classification problem using data collected from the class. Predict if Richard will like the Batman vs. Superman movie, given his ratings on the other movies in the dataset. The answer should be a 'yes' or 'no' classification.

5. Figures

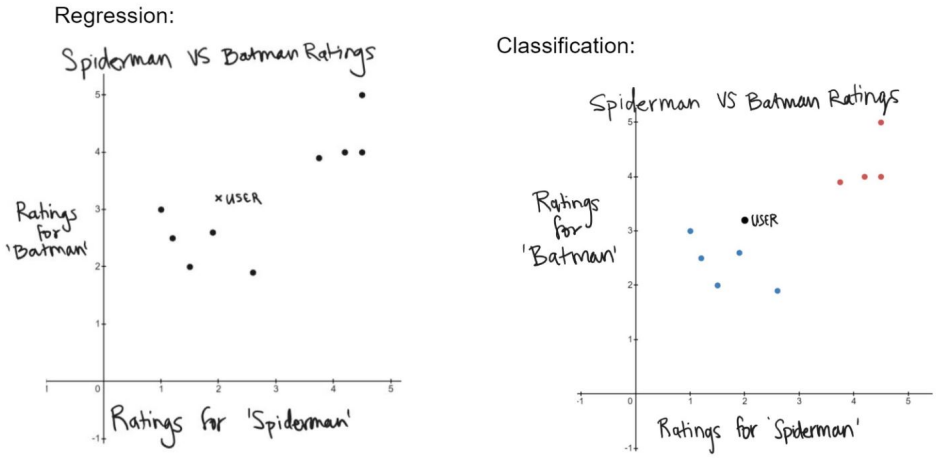

Figure 1. Classification and Regression: Spiderman vs. Batman ratings to answer the question 'how will our user rate the Batman movie?' (for classification, red data points are users who liked it, blue data points are users who didn't)

Spiderman	4.5	4.2	1.5	**1**	**1.2**	**2.6**	1.9	4.5	3.75	2
Batman	5	4	2	**3**	**2.5**	**1.9**	2.6	4	3.9	3.2
Superman	4.7	4	2	**3.5**	**2.5**	**2**	2	4.2	3.7	3

Figure 2. ratings to answer the question 'how will our user rate the newest superhero movie?'