K-Nearest Neighbors

Introduction:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification (mostly) and regression (less often) problems. The K-Nearest Neighbors algorithm assumes that similar things exist in close proximity and makes predictions based on the nearest neighbors[1]. The base KNN algorithm does not actually do any "learning," rather it makes predictions based solely on the distances of nearby data points. Choosing the k value takes special consideration. A low k value will be influenced by the noise in the data, but a high k value is computationally expensive. Common rules of thumb for choosing the k value include: $k = \sqrt{n}$ or $k = (\sqrt{n})/2$, where n is the number of data points. Though these rules of thumb are good places to start you may have to run through a bunch of tries to find the k that fits your data the best. Final tip, make the k value odd to reduce the chances of a tie[3].

Necessary Math

Euclidean Distance:

- Needed to determine how "close" points are to each other
- Formula: $D = \sqrt{\Sigma (x_i - y_i)^2}$
    - Expanded: $D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \ldots + (x_i - y_i)^2}$

How does it work? (High level algorithm)

1. Load in the training (labeled) data
2. Initialize the value of k
3. Predict label for X (run for each data point in the set)
    a. Calculate the distance between X and every other data point
    b. Sort these distances in ascending order
    c. Get k shortest distances
    d. Return the most frequent label

Pros and Cons[4]

Pros	Cons
Simple and easy to implement	Does not work well with high dimensional data
Does not require prior learning to make predictions so it performs faster than other algorithms that do require learning	High prediction cost for large datasets since the distance for each prediction must be calculated to each defined point
Since pre training is not required new data can be added easily	Does not work well with categorical data since it is hard to find distances to these values

Implementation - Scratch

```
class KNN():

 class Point():
   def __init__(self, loc, label):
     self.location = loc
     self.label = label

 def __init__(self, k, data):
   self.k = k
   self.points = []
   for point in data:
     self.points.append(self.Point(point[0], point[1]))

 def distance(self, p1, loc):
   sum = 0
   for x,y in zip(p1.location, loc):
     sum += (x - y)**2
   return (sum **0.5)

 def predict(self,location):
    distances = []
   for point in self.points:
     distances.append((self.distance(point, location), point.label))
   distances = sorted(distances)[:self.k]
   d, labels = list(zip(*distances))
   return max(set(labels), key = labels.count)
```

Implementation - With Libraries

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Sources

1. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
2. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

3. https://stackoverflow.com/questions/11568897/value-of-k-in-k-nearest-neighbor-algorithm
4. https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/