K Nearest Neighbors

KNN is a classification ML algorithm that works with the following process:

1. Calculate the distance to every point
2. Choose the k closest points
3. Each point gets an equal vote for classifying the target point

KNN can also be used for regression

KNN is a simple example of a non-parametric model - a model which "memorizes" all of the training data and uses it at test time.

As such there is no training phase and all of training data is used, meaning that the method is very computational expensive especially as the amount of data increases.

Let's look at an implementation of KNN from scratch:

In [0]:
```
from collections import Counter
import math
import numpy as np
```

In [0]:
```
def mean(labels):
  return np.sum(labels) / np.size(labels)

def mode(labels):
  return Counter(labels).most_common(1)[0][0]

def eucl_dist(p1, p2):
  axis_val = 0
  if len(p1.shape) != 1:
    axis_val = 1
  return np.sqrt(np.sum(np.power(np.subtract(p1, p2), 2), axis = axis_val))
```

The above are helper methods to generalize the KNN algorithm that follows.

As we see above, we've defined a distance function (Euclidean used here) and 2 choice functions:

- Mean will average the labels of the k nearest neighbors providing a regression answer
- Mode will yield the most common label of the k nearest neighbors, classifying the query data point

In [0]:
```
def knn(data, query, k, distance_fn, choice_fn):

  data = np.asarray(data)

  #strip labels, assumed to be last column of data
  features = data[:,:-1]
  labels = data[:,-1:].flatten()

  distance = distance_fn(features, query)

  # if distance doesn't have the same length, distance_fn isn't properly vectorized
  assert(len(distance) == len(data))

  #sort features and labels by distance
  inds = distance.argsort()
  sort_feats = features[inds]
  sort_labels = labels[inds]

  #return k closest indices, points and label
  #choose label based on choice_fn, mode for classification, mean for regression
  return dict(zip(inds[:k], sort_feats[:k])), choice_fn(sort_labels[:k])
```

A simple example follows (credit: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761))

In [0]:
```
#data with 1 feature, and last column as labels, 0 or 1
clf_data = [
       [22, 1],
       [23, 1],
       [21, 1],
       [18, 1],
       [19, 1],
       [25, 0],
       [27, 0],
       [29, 0],
       [31, 0],
       [45, 0]]
```

We run KNN here with the eucledian distance function and as a classifier
(using mode instead of mean).

In [0]:
```
neigh, pred = knn(clf_data, [33], 3, eucl_dist, mode)
```

In [0]:
```
print("The k closest points from the dataset to our query point:")
print(np.asarray(list(neigh.values())).flatten())
```

```
The k closest points from the dataset to our query point:
[31 29 27]
```

In [0]:
```
print("Label classification for the query point:")
print(pred)
```

```
Label classification for the query point:
0
```