Exercise - K-Nearest Neighbours

Jaya Nilakantan
Dawson College
jnilakantan@dawsoncollege.qc.ca

1. Introduction

In this exercise, we are introduced to both data sets and a first algorithm in supervised learning. We will be using the famous Iris dataset for a first algorithm for classification.

1.1. Recall: Supervised learning

Supervised learning start with a set of labeled data (called *training* data) which is used to make a mathematical or statistical model. The model is used for either prediction or classification purposes.
In order to validate the model, *test* data is used. The test data is like the training data, that is chosen to properly reflect the different cases/demographics/conditions (called *features*) which are being looked at. Training and test data are important: they drive the correctness of the algorithm (the confidence with which you can make a conclusion).

1.2. Where to find data?

There are various places where you can find data to create and test models:

- repositories of open data
  - websites like Kaggle (kaggle.com)
  - Microsoft Research open data (msropendata.com)
  - government open data (e.g., donnees.ville.montreal.qc.ca)
  - Classic data sets (list at https://en.wikipedia.org/wiki/Data_set#Classic_data_sets)
- APIs
  - for example, Reddit, Stack Overflow, NY Times, Wikipedia, …
- web scraping

In this exercise, we are using a famous data set, Fisher's *Iris* flower dataset used for classifying irises. Fisher was a statistician and biologist who measured

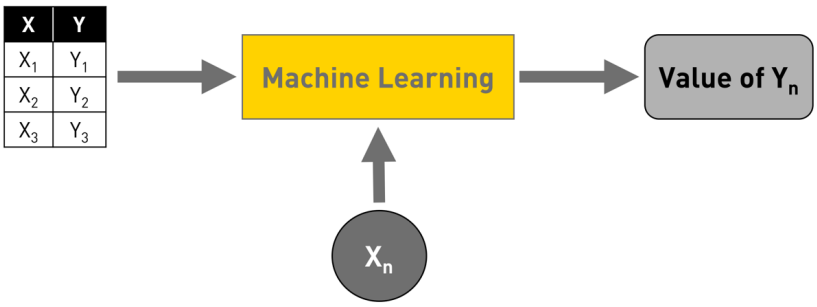

Figure 1. Predicting a value based on existing data

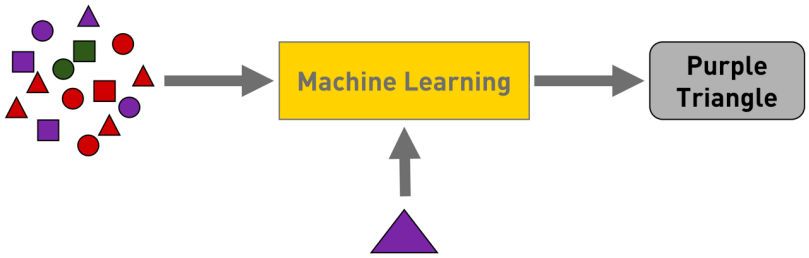

Figure 2. Classifying an object based on existing data

features on three related species of irises. Fun fact - there is a Quebec connection, 2 of the 3 species were measured from a pasture in the Gaspesie!

2. Understanding a classification problem

Consider the example where we ask students prior to taking a test how many hours they slept during the past 12 hours, and how many hours they studied during that same time. We then get pass/fail results from the test. The results might look something like this *sniff*:

It is often easier to visualize the data through a chart. In this case, the colour legend indicates if the student passed (blue) or failed (red). A new point x is added - would you predict that this student passed or failed the exam?

2.1. Features and Labels

Every data point in this dataset, just like the salary dataset and the housing prices dataset, has *features* (the variables that influence the result, i.e., hours studied and hours slept), as well as a *label* (the result). Your instincts in this case tell you that the poor student probably failed - not because you were able to draw a line and extrapolate, but because you noticed *clusters*. With this

Hours studied	Hours slept	Result
2	9	Fail
2	5	Fail
4	7	Fail
7	5	Pass
6	2	Pass
8	2	Pass

Figure 3. Hours slept and studying vs result

dataset, each point is discrete. We use the data to try to classify the test data into a particular discrete class: a student can either pass or fail, nothing else.

2.2. K-Nearest Neighbours algorithm

You want to find out if the data point being tested (drawn x) belongs to the Pass cluster or the Fail cluster. We basically want to see which other training data points it is close to, and guess its classification based on the neighbours. k indicates the number of nearest neighbours you want to calculate to get the vote. Let's say we say that $k = 3$ for the dataset above. Which are the three closest neighbours to x?

The easiest distance measure is *Euclidean*. In our example, we have two features, therefore 2 dimensions. Recall, in two dimensions, if point $p = (p_1, p_2)$ and point $q = (q_1, q_2)$, then distance between point p and q is:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{1}$$

More generally, when you have n dimensions (each representing a feature), then the Euclidean distance between 2 points is calculated as:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{2}$$

We can use Euclidean distance to find the 3 closest example students to the student x at $(4, 5)$ (4 hours studied, 5 hours slept). We see the three closest points are $(4, 7)$ (distance of 2), $(2, 5)$ (distance of 2), and $(7, 5)$ (distance of 3). Two out of 3 neighbours fails, thus we would predict that this students fails also.

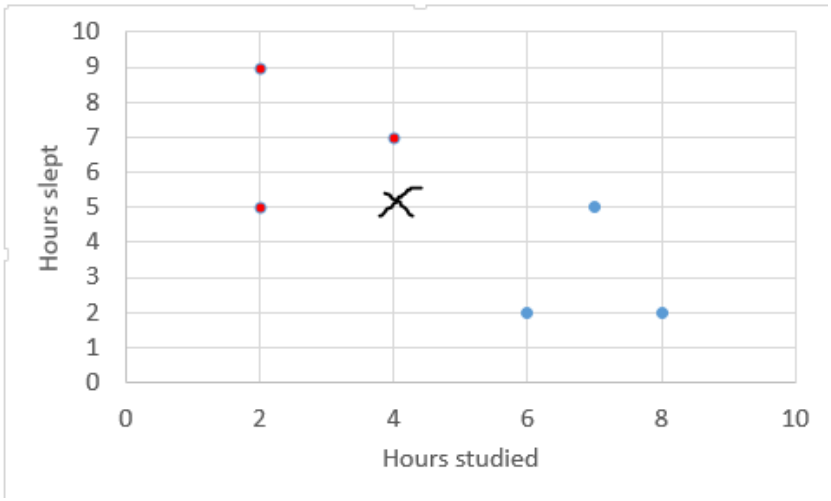

Figure 4. Hours slept and studying vs result

2.3. How to choose k?

Question: if the dataset contains m datapoints, does it make sense for $k = m$? i.e., should every datapoint have a vote as a nearest neighbour?

Answer: No! If you do that, you are giving a vote to all datapoints. So if you have an odd number of points, you will predict all to be one type or another

Question: should I choose a small value, like $k = 1$?

Answer: No! Your algorithm may become to sensitive to noise/outliers in the data set

There is in fact an optimal value of k, where the error rates (false classifications) are minimized. There is a technique called k-fold cross validation that is used. It is beyond the scope of this exercise, but we will examine it in a future exercise.

2.4. KNN pseudocode

```
Load the data
Initialise the value of $k$
Iterate from 1 to $m$ (total number of training data points)
   Calculate the Euclidean distance between test data and the training datapoint.
Sort the calculated distances in ascending order
Get top k rows from the sorted list
Count the most frequent classification of these rows
```

3. Exercise 1

Code the KNN algorithm in a php script by completing the following classes:

```
/** Represents a point p with n attributes (i.e, the size of the array in n)
**/
class Point {
  private $array; //array containing the values in each dimension p1, p2, p3, ...

  public function __construct($array = []) {
     $this->array = array;
   }
  /** Returns Euclidean distance between $this and the parameter.
    Throws an InvalidArgumentException if the dimension of p are not
    the same as $this.
  **/
  public function getEuclideanDistance (Point $p){
  }
}

/** Represents a dataset element, with n attributes and a classification
**/
class DataPoint {
  private $point; //attributes
  private $class; //the classification

  public function __construct($array = [], $class = ''){
     $this->point = new Point($array);
     $this->class = $class;
  }

  /** Returns Euclidean distance between $this and the parameter, based on their
    Points.
  **/
  public function getEuclideanDistance (Point $p){
  }
}

/** Encapsulates the k-nearest-neighbour logic
**/
class KNN {
  private $testpoints; //array of DataPoints


  public function __construct($arrayDataPoints = []){
     $this->$testpoints = $arrayDataPoints;
```

```
  }

  /** Returns an array with the Euclidean distance with the given
  Point
  **/
  private function distances (Point $test):array {
  }

  /** Returns an array with the k closest neighbours to Point
  */
  private function getKNearest(int $k, Point $test):array{
  }

  /** Get top choice: counts the top choices and selects the majority
  */
  public function getPrediction : string (int $k, Point $test):string{
  }
}
```

Next use your classes to check if the student at datapoint (4, 5) is indeed predicted to fail when $k is 3.

3.1. Quiz

1- When are the majority of computations done? When we train the model? Or when we test a datapoint?

2- True or false: The algorithm performs with the same efficiency, regardless how many features you have.

3- True or False: the computational complexity for classifying a new datapoint grows linearly with the size of the training set (i.e., it has O(N) complexity, where N is the sixe of the training set)

4- True or False: Euclidean distance is the only distance algorithm that you should use

Referring to Figure 5, you want to predict the class of data point *(1, 1)* (is it *o* or *+*). Answer the following two questions:

5- Using Euclidean distance and k = 3, in which class would you predict that the data point belongs?

6- Using Euclidean distance and k = 7, in which class would you predict that the data point belongs?

Answers: 1- There is no training phase per se. All we do is set up the arrays and points. Each time we want to test a new point, we need to calculate the distances, so the majority of computations are only done when we have a test point to classify.

2- False: the more features (i.e., attributes), the more computations are necessary to calculate the distance.

3- True

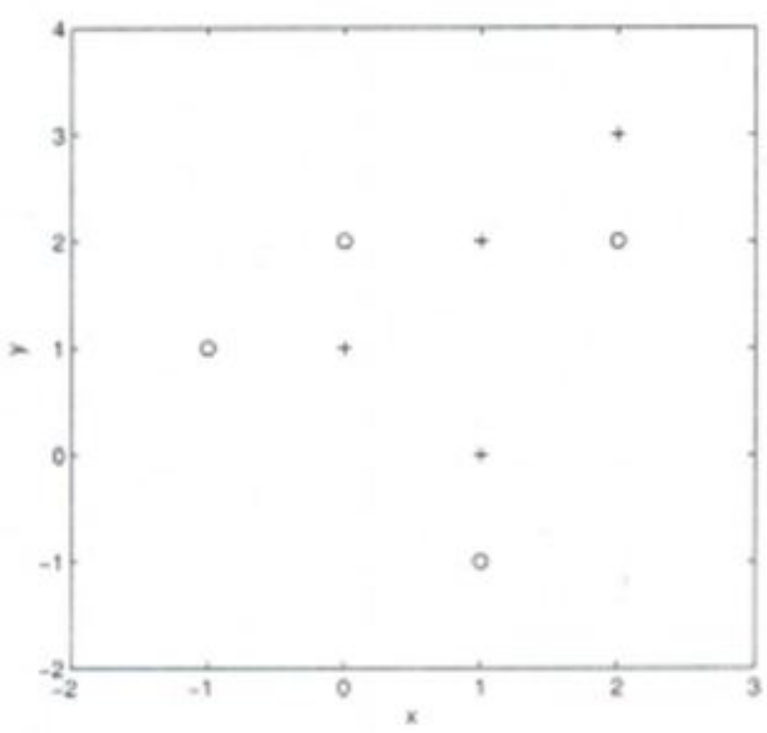

Figure 5. Scatter plot for questions 5 and 6

4- False. Depending on your dataset, you may consider other distance algorithms (which we may examine in the future). For example, Manhattan (a.k.a city block, taxi cab), Minkowski, or Jaccard (amongst many other measures).

5- All 3 nearest neighbours are +

6- There are 4 o and 3 +, so the predicted class is o.

4. Iris flower dataset

The flower dataset contains measurements of 3 different but related species of irises. For each species, there are 50 flowers which were measured; and each flower had 4 attributes measured

- sepal length
- sepal width
- petal length
- petal width

You can see the dataset at iris.csv. A visualization:

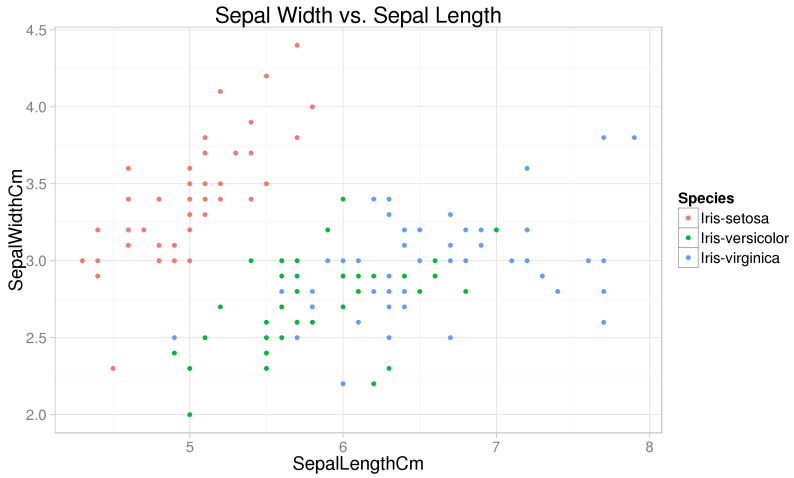

Figure 6. Scatter plot of Irises

4.1. Exercise 2

Use the class that you coded to classify the iris with these attributes: [7.2, 3.6, 5.1, 2.5]. Use $k = 1$, $k = 3$ and $k = 5$.

4.2. Exercise 3

PHP-ML provides a KNearestNeighbours implementation. Read the documentation, and change write an application that trains and predicts the same unknown iris. Are your results the same?

References

http://archive.ics.uci.edu/ml/datasets/Iris

http://caisplusplus.usc.edu/blog/curriculum-supplement/knn

https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

https://www.analyticsvidhya.com/blog/2017/09/30-questions-test-k-nearest-neighbors-algorithm/