Welcome!!

We will start with summarizing the plans/schedules and offer some comments on what you will see/experience for the rest of the course. After this brief tour of the course, we will directly get into the basics.

Session 1: Introduction

In this session, we will start with the basics of machine learning. We will discuss what we mean when we say a machine is able to learn and how that happens. We will demistify the technical space, while setting the fundamental understanding.

Now let us see a glimpse of how a wide variety of problems that we see in real world will get connected to the algorithms and approaches that we use. Let us now look at a very simple (possibly the most intuitive) method based on nearest neighbours. Let us see how the intuition of "similar people show similar characters" can lead to Nearest Neighbour based schemes. Most popular in this space is K Nearest Neighbour (KNN) classification.

Now we have a working solution, let us go deeper and see what NN schemes enable us. We will see some of the extensions, applications and analysis of the nearest neighbour schemes.

We will follow up with some experiments in the labs for you to

1. be comfortable with using the environment,
2. working with the data sets, and
3. probing into the algorithms and improving insights.

Experiment 0

This is to be carried out in the Lab.

This is to get familiar with Jupyter notebook, and simple uses of pandas and matplotlib

Experiment 1

This is to be carried out in the Lab.

In this experiment, we will see a simple data set and how a simple K nearest neighbour classification works. To motivate, let us consider a simple story [1]. We want to find and label a fruit automatically. Fruits are characterized by:

- weight (in grams)
- colour as an integer
    1. red
    2. orange
    3. yellow
    4. green
    5. blue
    6. purple
- Let us also represent the labels as (apple (A) and banana(B)).

We are given some samples such as $< 303, 3, apple >$ meaning the fruit with 303 gram weight, and yellow colour is an apple. A set of such "training examples" is given in "01-train.csv". This has a small set of "labeled" 17 examples.

We are given a set of *test* data where only weight and colur is given (eg. $< 373, 1 >$). We should design a simple Nearest Neighbour classifier that will find the fruit label. i.e., apple or banana. We have 102 such test cases.

As an objective of finding, how good is our prediction, we are also given labelled csv files, which have labels for all the test cases. If your predicted label (fruit name) is correct, you have done well !!. But there is also a chance that your predictions can go wrong. Let us be realistic. We can use 17 examples the same or similar as in [1]. We can create test samples random but around the labeled examples.

Exercises

1. Plot the training data on a 2D plane. Do you see that similar fruits come close in the feature space?

2. Now plot the test samples as "black" on the same plot.
3. Find the accuracy of the prediction (percentage of the samples that are correctly predicted) with a simple KNN classifier with K=3 and distance we Euclidean distance. You may observe that with a small number of labeled examples (17), we are predicting the labels "accurately" on a larger novel set of examples.
4. Find the accuracy of another test set.
5. Find the accuracy of samples in the test set with $K = 1$, $K = 5$ and $K = 17$, . What do you observe? Why is $K = 17$ a bad choice? What happens to all the labels now?
6. If we have used the weight in Kg (i..e, dividing the weight in grams by 1000), what will be the accuracy for $K = 3$?
7. Assume we have used only the weight in grams (by discarding the colour feature) for calculating distance. Calculate the accuracy on the test set. (modify only the distance function).
8. If we have used sum of absolute differences:

$$d(x, y) = \sum_{i=1}^{d} |x_i - y_i|,$$

can the accuracies change? (This is the *Manhattan* distance).

Summary

In the above experiment, we find that a simple nearest neighbor method can successfully predict labels with a small number of labelled examples. But we had also seen that the results can go really wrong if we make some wrong choices (like weight in Kg, or a very large K). This should remind you about the practical expertise and experimental skills that will become equally important as we move forward.

Experiment 2

This is to be carried out in the Lab.

Now we have a "generic" implementation of KNN classifier. It can use a user defined distance function. Let us use a more realistic setting and a more realistic way of getting test vs training data.

UCI has a dataset for breast cancer [2].

In practice, as a designer of the algorithm you are given only one set of data. The real test data is with your teacher/examiner/customer. The standard way is to create a small validation data from your training data and use it for evaluating the performance and also

for parameter tuning. Let us *randomly* split the data into 80:20. We will use 80% for training and the rest 20% for evaluating the performance.

Using all the 32 attributes/features, predict the patients in the test data as M = Malignant, B = Benign. Use $K = 5$ and Euclidean distance. Find the accuracy (as percentage) on the test data.

Repeat the above (creating random partitions and evaluating the performance) 5 times. You will see that they vary in different trials. An average of these attempts is in fact a better estimate!!

Vary K from 3 to 11 and find the best K. In practice, we will use similar ideas for finding best parameters ("validation"). More at a later stage.

Summary

We had now seen how KNN works in practice on a real data. We also have some hints on how to find the "best" parameters by "internally testing" the performance on a small portion of the data that we have. Also we have seen that there could be "statistical variation" in performance across multiple experiment and a more reliable way to estimate the performance is to find the average across multiple experiments.

Case Study 1: Fraud Detection

This is to be carried out in the Lab.

We use the credit card data set for this problem. The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Universit Libre de Bruxelles) on big data mining and fraud detection. More details on current and past projects on related topics are available on http://mlg.ulb.ac.be/BruFence and http://mlg.ulb.ac.be/ARTML

Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

In this case study, the goal is to explore the ramifications of using kNN for a highly unbalanced dataset.

The datasets contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions.

Please note, that due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction

Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Experiment 3

This is to be carried out in the Lab, if you are able to complete the other Experiments ahead of time. Our expectation is that this will be done by almost all, after the Lab Session.

The KNN classifier can be extended to a multiclass classification problems. i.e., when the number of classes/labels are more than two). Let us use iris data for this.

Case Study 2: Recommendation Systems

This is expected to be carried out after the Lab Session at home.

Consider the problem of recommending movies to users. We have M Users and N Movies.

A user either had seen the movie (1) or not seen the movie (0). We can represent this as a matrix of size $M \times N$ (M rows and N columns). We have actually used a dictionary with the keys userId and movieId to represent this matrix.

Each element of the matrix is either zero or one. If (u, m) entry in this matrix is 1, then the u^{th} user has seen the movie m .

Now, we want to predict whether a given test user x will watch movie y.

User x has seen and not seen few movies in the past. We will use x's movie watching history as feature for our recommendation system.

We will use KNN to find the K nearest neighbour users (users with similar taste) to x, and make predictions based on their entries for movie y.

We have given the code for Cosine distance, when computing nearest neighbours.

Please refer to [4] for more detailed (slightly different) exercises.

References

1 http://www.jiaaro.com/KNN-for-humans

2 https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

3 http://www.michaeljgrogan.com/knn-breast-cancer-data-python/

4 http://kevinmolloy.info/teaching/cs504_2017Fall/CS504_FinalProjectDescription_2.pdf