420-LCW-MS Programming Techniques and Applications - Lab Exercise 9

April 27, 2022

Goals for this lab:

- Practice with clustering.

Lab assignments will be marked pass/fail and will count for 1% of your final mark.

I do NOT necessarily expect you to finish all of the exercises, but do think about them and see if you can imagine how a solution might work.

Introduction

Clustering is a very basic technique for *unsupervised* machine learning. The general concept is simple. Start with a data set consisting of a n data items with m features each. Now choose a number of classes k. The goal is to choose an assignment of the n items to the k classes that minimizes some error measurement.

There are many clustering algorithms possible, but one of the most basic methods is known as *k-means* clustering. The algorithm computes a mean vector that acts as a "centre" for each of the k clusters, and assigns each item in the dataset to the cluster with the closest centre. The distance from an item to the mean is typically computed using the the Euclidean distance.

The k-means algorithm consists of three basic steps:

1. **Initialize:** select k values to define the "centres" of the prospective clusters. These can be chosen just by picking k elements of your dataset and using their feature vectors as the initial centres.
2. **Assignment step:** assign each item to be a member of the cluster with the closest centre.
3. **Update step:** given the new assignment, recompute the centre of the clusters by averaging the feature vectors currently assigned to that cluster.

The assignment step and update step are performed repeatedly, until the assignment converges. The assignment converges when the assignment step does not change any of the selected classes.

There are many possible variations on the algorithm, especially with the choice of initialization. It is also possible to perform the update step first, base on an initially random choice of assignments.

There are (at least) two fundamental problems with the k-means algorithm. First, because the initialization is chosen at random, the algorithm is not deterministic. This means that every run of the algorithm may give different results, and these results may not be equally good.

Another problem with the algorithm is the choice of k itself. In some cases it may be known ahead of time. However, it may be unknown, and so one of the user's goals may be to determine the best choice for k. It is possible to use the algorithm in this way, but not trivial.

Details

I've provided with a simple implementation of k-means and some supporting files:

- `k_means.py`: implements the k-means algorithm. You need to complete this file for exercise 1.

- `lab9.py`: a very short main program that loads some data files and tests the k-means algorithm. You can use this file to complete exercises 2 and 3. Make any observations in comments in this file.
- `classifier.py`: defines an abstract classifier and some supporting functions.
- `dataset.py`: defines functions to read several datasets.

There are two functions of particular interest defined in the file `k_means.py`:

- `iterations, dissimiliarity, classes = k_means(dataset, k)` - Implements the k-means algorithm. The return values are:
  - The number of iterations required to converge.
  - The dissimilarity score is the total distance of all items to their cluster centres. In general, the smaller the dissimilarity, the better the clustering.
  - The cluster assignments for each item in the dataset. This is a list of integers between 0 and $k-1$ that gives the current assignment of each item in the dataset.
- `x = purity(dataset, classes, k)` - Computes the quality of the clustering results. The return value is a number between 0 and 1, where 1 is perfect clustering.

Exercise 1 - implementation

The `k_means.py` file I have provided is almost complete, except for the code for the *assignment step*. You need to finish the algorithm by finishing this code.

Your code needs to assign values to the `assignments` list such that `assignments[j]` is an `int` from 0 to `k-1` that represents the current best class assignment for `dataset[j]`. Remember the best class assignment is based on the closest centre, so you are trying to find the value that minimizes:

```
distance(dataset[j].data, centre[assignments[j]])
```

In addition, your code needs to calculate the `dissimilarity`, which is the sum of the minimum distances for each of the items:

$$d = \sum_{j=0}^{N} D(x_j, c_i) \tag{1}$$

where N is the number of items, $D(x, y)$ is the Euclidean distance function, and c_i is the class centre closest to x_j.

Exercise 2 - multiple runs

Modify the `lab9.py` file by adding some code that runs the k-means algorithm ten times and selects the *best* cluster assignment found over n runs. Use the dissimilarity value returned by the `k_means` function to select the best classification. Use the `purity()` function to evaluate the final classification. The purity score measures the fraction of items from the dataset which are grouped properly by the clustering algorithm. The best possible purity score is 1.0, representing perfect clustering.

Does the reduced dissimilarity necessarily result in a better purity score?

Exercise 3 - selecting k

Each of the datasets used are known to have three classes. Suppose you didn't know this already. How could you discover it? Implement a method to try various values of k systematically and observe how the dissimilarity and purity results change with k. Could you use the dissimilariy measure to determine which k is the best choice?

Make a table or graph to show how the dissimilarity and purity change with different values of k.

Submitting your work

Submit your revised `k_means.py` and `lab9.py` files in a single ZIP file. Place data and comments in your `lab9.py`.