ML 4/M KNN Lab

Aims

- To implement a K-nearest neighbours classifier for binary data
- To investigate classification performance as a function of K

Tasks

- Download `trainx.csv` and `testx.csv` from Moodle. In `trainx.csv`, each row corresponds to an instance. The first two columns are the values for two *features* and the third is the class label. The same format is used in `testx.csv`.
- Load these datasets into python (`numpy.loadtxt`) and create an $\mathbf{X}$ matrix consisting of the first two columns and a $\mathbf{t}$ vector as the last one. Do the same for the test data so you have four objects: $\mathbf{X}$, $\mathbf{X}_{test}$, $\mathbf{t}$ and $\mathbf{t}_{test}$.
- Implement a KNN function that takes a single test example and a value of K and returns a classification. Your function should find the K closest (see below) training points to the test point and return the majority class amongst these training points.
- Compute the classification accuracy over the entire test set (the proportion of test points it gets right).
- Wrap your code in a loop over K and plot the test classification accuracy as a function of K. Is there an optimal value of K?

Python help

If your training data is in a numpy array with 100 rows and 2 columns, then the distance between a test point and the ith row is given by:

```
sq_diff = (test_row - trainx[i,:])**2
dist = np.sqrt(sq_diff.sum())
```

where `test_row` is a row of $\mathbf{X}_{test}$.

The first line creates a new vector which holds the squared difference of the two pairs of values. The second line takes the sum of these differences and then takes the square root. This is computing the Euclidean distance. Other distance metrics could also be used.