Machine Learning@SUTD
Assignment 1

Linear Mappings / k-nearest neighbors

Due on Wednesday, September 23, 2015

This sheet is long but the tasks are simple. Easy points to collect on your way! It might not be that easy in later assignments.

1. upload all files for all tasks in one zip-(or .tar.gz)-file. for the images provide a description which image is for what, e.g. give the descriptions in one file which contains all other task results (equations, textual explanations) and descriptions for all images. best is one long pdf (or word) file with embedded images, equations, textual explanations.

   The code for task 2 should be in one single .py file, and not embedded in any pdf or word. Please make it readable, with comments etc.

2. install python,numpy,scipy, matplotlib, ipython (interactive shell for python) (0P)

   - `http://www.scipy.org/install.html`

3. k-nearest neighbors (8P)

   (a) deliver the whole code for this k-NN task in one .py file. it should be executable and runnable.

   (b) draw 5000 training data points from a uniform distribution in $[-1,1] \times [-1,1] \subset \mathbf{R}^2$. Let $\mathbf{X_2}$ be this data. generate binary labels $\mathbf{y}_2$ for them in the following way: If the second component of the data (i.e. $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, then the second component is x_2) is positive, then with probability of 75% the label is +1.If the second component of the data is negative, then with probability of 75% the label is -1. Draw 1000 testing data points $\mathbf{X_3}$ with labels $\mathbf{y}_3$ with the same distribution. Plot the training points and deliver the plot as jpg (2P)

   (c) implement a k-nearest neighbor classifier with euclidean metric and number of neighbors parameter k as variable (2P)
   the interface should be:

```
def knn(xtrain,ytrain,xtest,k):
    #xtrain, xtest have shape (2,number of data)
    ...
    #ytest are the predicted labels for xtest
    return ytest
```

   (d) design hint: do you need the square-root of the euclidean metric for the k-nearest neighbors algorithm? give a justification for your answer (1P)

   (e) run the k-nearest neighbor with $k = 100$ on the test data, compute the zero-one error on the test data, plot the predictions, deliver this plot as a jpg file, deliver the (average!) 0-1-error as a score (1P)

   (f) now scale the first dimension of training features $\mathbf{X}_2$ and test features $\mathbf{X}_3$ (i.e. $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, then the first component is x_1) by multiplying it with a factor of 1000, run the k-nearest neighbor with $k = 100$ on the test data, compute the zero-one

error on the test data, plot the predictions, deliver this plot as a jpg file, deliver the 0-1-error as a score (1P)

(g) compare the test error for this and the last dataset – some algorithms are sensitive to scaling of the features

(h) compute the standard deviation for both dimensions of the training data, print the standard deviation for both dimensions and deliver it

(i) divide each dimension of training features $\mathbf{X}_2$ and test features $\mathbf{X}_3$ by their respective standard deviations. print the standard deviation again for both dimensions and deliver it. now run the k-nearest neighbor with $k = 100$ on the test data, compute the zero-one error on the test data, plot the predictions, deliver this plot as a jpg file, deliver the 0-1-error as a score (1P)

(j) note the effect of normalizing the input dimensions by letting them have standard deviation equal to 1. A more general transformation is: at first subtracting the mean, and then dividing by standrard deviation. The data then has zero mean and unit variance.

(k) can you write down a distance function inspired by the euclidean measure which would work well with the badly scaled data ($*1000$) from above? (1 extra P)

4. Linear transforms in 2 dimensions (in python) (5P)

(a) Draw $n = 100$ data points from a two-dimensional gaussian Distribution with mean $\begin{pmatrix}1\\3\end{pmatrix}$ and standard deviation of 1. The data matrix should be formatted having shape $(2, 100)$. Let $\mathbf{X}$ be this data.

(b) Plot the data as a set of points by the python/matplotlib command
`plot(x[0,:],[1,:],'.')`.

(c) deliver this plot as a jpg file (some mild compression is ok, it is not a foto) (1P)

(d) create a linear mapping $\mathbf{A_1}$ (represented by a 2×2 matrix) which mirrors the data along the y-axis, apply this mapping to the data: $\mathbf{A_1 X}$, and plot the data

(e) deliver this plot as a jpg file (1P)

(f) create a linear mapping $\mathbf{A_2}$ which scales the x-axis by 0.5, apply this mapping to the data, and plot the data

(g) deliver this plot as a jpg file (1P)

(h) create a linear mapping $\mathbf{A_3}$ which rotates the data by 45 degrees clock-wise, apply this mapping to the data, and plot the data

(i) deliver this plot as a jpg file (1P)

(j) create a linear mapping $\mathbf{A_4}$ which mirrors the data along the x-axis

(k) compute the matrix $\mathbf{A_5} = \mathbf{A_2 A_1 A_4}$, apply this mapping to the data, and plot the data

(l) deliver this plot as a jpg file (1P)