Niko Krasowski
niko.krasowski@iwr.uni-heidelberg.de

Exercise 2

Deadline: 03.10.2014

This time, we are focusing on the implementation and evaluation of linear/quadratic discriminant analysis (LDA/QDA) and the Nearest Mean Classifier. We will work on the digits dataset again.

1 Data Preparation (1 point)

As in exercise 1 we will try to distinguish digits from the digits dataset provided by sklearn. Reminder:

```
from sklearn.datasets import load_digits

digits = load_digits()

print digits.keys()

data         = digits["data"]
images       = digits["images"]
target       = digits["target"]
target_names = digits["target_names"]
```

We will once again try to distinguish "1"from "7". Please filter the dataset such that only these two digits are left. There filtered dataset is supposed to have 359 examples. Split this filtered dataset again in a training- and a testset (#train/#test = 3/2).

1.1 Dimension Reduction (2 points)

To make your life harder (and to simplify the visualization of the featurespace) you are supposed to restrict yourself to 2 feature-dimensions. You can for example do this by choosing two pixels, that seem to have a big influence. The decision on how to choose the features is completely up to you. Maybe you want to have a look at the accumulated images of each class or their difference image. You can also come up with a clever combination of the 64 features that results in 2 features ($\hat{f}_i$). E.g.: $\hat{f}_1 = 0.3 f_{23} + 42 \frac{f_{13}}{f_{64}}$ and $\hat{f}_2 = f_{33} - f_{62}$. For the following exercises on this sheet you are supposed to work only with these two features. Note that the quality of your features determines the intrinsic error and therefore is a limiting factor for the quality of the predictions.

2 Nearest Mean (2 points)

Implement a nearest mean classifier. Therefore find the mean of the feature vectors of each class in the training set and assign each testpoint the label of its nearest mean. Argue what kinds of classification problems can be tackled by this approach and for which cases the nearest main classifier will probably fail.

3 QDA

3.1 Implementing QDA - Training (4 points)

As a first step, implement a matlab function
`[mu0, mu1, covmat0, covmat1, p0, p1] = compute_qda(trainingy,trainingx)`
that accepts a 1 x n1 vector trainingy of input labels (that must be either 0 or 1) and a d x n1 matrix trainingx of features. The output should contain the d x 1 mean vectors, the d x d covariance matrices and the priors p0 and p1 as scalars.

Niko Krasowski
niko.krasowski@iwr.uni-heidelberg.de

3.2 Implementing QDA - Test (3 points)

Now, using the output from the previous section implement a function
`[qda_prediction] = perform_qda(mu0, mu1, covmat0, covmat1, p0,p1, testx)`
which predicts the labels of a d x n2 dataset testx using QDA.

3.3 Scatterplot (2 points)

Apply the QDA prediction to the **training** data (the one that you constructed in 1 and 1.1 that has 2 feature dimensions and only labels "1änd "7"), compute the correct classification rate and visualize the results: A good way is to make a 2D plot of the data with regard to the two scores. Mark true the class (1,7) with different colors and also add the QDA estimate (1,7) using a different symbol (x instead o). You can produce a scatterplot via:

```
import matplotlib.pyplot as plt
import numpy as np

x1 = np.random.randn(200)
y1 = np.random.randn(200)
x2 = np.random.randn(200)
y2 = np.random.randn(200)

plt.title('Scatter Plot')
plt.xlabel('X axis')
plt.ylabel('Y axis')

size = 20
# b: blue,    g: green,   r: red,    c: cyan,
# m: magenta, y: yellow, k: black, w: white
plt.scatter(x1,y1, marker="x", c="r", s=size)
plt.scatter(x2,y2, marker="o", c="b", s=size)
plt.show()
```

3.4 Visualize the decision boundary (2 points)

It can be quite insightful to obtain an image of the decision boundary. Therefore, create a sensible grid of input values for the two scores (since you created the features on your own you will have to adapt the grid such that it covers your whole feature-domain) and perform the QDA for each combination of inputs. A 100×100 grid provides a sufficient resolution. Make a plot to visualize the decision boundary. Using this plot and your results from the previous section to inspect the quality of the QDA results on the **training** data. You should observe some misclassifications. Where do misclassifications on training data using QDA stem from (compared for example with NN, that doesn't have any misclassifications on training data)?

3.5 Prediction (2 points)

To get an idea of the generalization of the results. Therefore apply the QDA to the **test** data and compute the correct classification rate (we are still working on the 1 against 7 task). Make a similar 2D plot as in 3.3.

4 LDA (2 points)

Modify the training of your classifier in 3.1 in a way that it results in LDA. How does the prediction quality suffer?