Labwork Bayes decision

G. Gasso

Objectives: Apply Linear and Quadratic Discrimant Analysis on a synthetic dataset and on two real applications

Provided codes: you will require some useful functions provided in the file `utility_bayes.py` available on Moodle.

1 Synthetic data

Let consider a binary classification problem with class $y \in \{1, 2\}$. The conditional distribution of each class k is $p(x/y = k) = \mathcal{N}(\mu_k, \mathbf{S}_k)$ i.e. a gaussian distribution with mean μ_k and covariance matrix $\mathbf{S}_k$.

1. Generate $n_1 = n_2 = 100$ training samples per class using $\mu_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$, $\mathbf{S}_1 = \frac{3}{2}\begin{pmatrix} 1 & 0.1 \\ 0.1 & 1 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$ and $\mathbf{S}_2 = \mathbf{S}_1$

```python
from utility_bayes import gen_data_twogaussians_2d

# class 1
n1 = 100
mu1 = np.array([0, 2]); S1 = 1.5*np.array([[1, 0.1], [0.1, 1]])
# class 2
n2 = n1
mu2 = np.array([-2, -2]); S2 = S1

X_train, Y_train = gen_data_twogaussians_2d(mu1, S1, mu2, S2, n1, n2)
```

2. Learn LDA and QDA classifiers on the training data and plot their respective decision regions. Comment the obtained results and justify the form of the decision frontier.

```python
from utility_bayes import plot_decision_regions_2d
#LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
clf_lda = LinearDiscriminantAnalysis(solver="svd", store_covariance=True)
clf_lda.fit(X_train, Y_train)
plot_decision_regions_2d(X_train, Y_train, clf_lda, resolution=0.02, title="LDA")

#QDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
clf_qda = QuadraticDiscriminantAnalysis(store_covariance = True)
clf_qda.fit(X_train, Y_train)
plot_decision_regions_2d(X_train, Y_train, clf_qda, resolution=0.02, title="QDA")
```

Hint: check the covariance matrices of both classifiers

```python
print(clf_lda.covariance_)
```

3. Set different covariance matrices $\mathbf{S}_2 \neq \mathbf{S}_1$ and run experiment of question 2. What do you observe?

4. Evaluate the accuracy of LDA and QDA classifiers on the training set

```
from sklearn.metrics import accuracy_score

#LDA
pred_lda = clf_lda.predict(X_train)
acc_train_lda = accuracy_score(pred_lda, Y_train)
print("LDA : accuracy = {}".format(100*acc_train_lda))

#QDA
...
```

2 White dwarfs and the stars

Sloan Digital Sky Survey is an international program with the objective to create "the most detailed three-dimensional maps of the Universe ever made, with deep multi-color images of one third of the sky, and spectra for more than three million astronomical objects". The used dataset consists of the brightness (in logarithmic scale) of different objects in 5 wavelengths (u: ultraviolet, g: green, r:red, i, z: infrared). Hereafter, we will address a binary classification problem aiming to distinguish stars (class $y = 1$) from white dwarfs ($y = 2$). The input variables are formed from the raw data as: $u - g$,$g - r$, $r - i$, $i - z$. The resulting dataset `astrodata.mat` is downloadable on Moodle.

1. Load the dataset.

```
import scipy.io as sio
filename = "astrodata.mat"
X = sio.loadmat(filename)["X"]
Y = np.squeeze(sio.loadmat(filename)["Y"])
```

2. Perform a statistical analysis of the inputs and analyze the figures.

```
# Statistical analysis
# boxplot of the input variables and the output
plt.figure(figsize=(12, 3))
variables = ["u-g", "g-r", "r-i", "i-z"]
plt.boxplot(X, vert=True, labels=variables)
plt.xticks(fontsize=16); plt.yticks(fontsize=16);

# Correlation matrix
plt.figure(figsize=(8, 7))
correlation_matrix = np.corrcoef(np.transpose(X)) #remark: we need to transpose X
plt.matshow(correlation_matrix, cmap=plt.cm.RdBu)
```

3. Split the data into a training set and test set. Justify why do we need to do this? Explain why do we need to set the shuffle option to `True`. Same question for the stratify option.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, shuffle=True, test_size
    =0.5, stratify=Y)
```

4. Normalize (centering and standard scaling) the training and test sets. Justify how the normalization is performed.

```
from sklearn.preprocessing import StandardScaler

# define the scaler
sc = StandardScaler(with_mean=True, with_std=True)
# set the parameters of the scaler using training data
sc = sc.fit(X_train)
# apply the so set scaler to normalize the data
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

5. Learn a LDA model and evaluate its accuracy either on the training and test sets.

6. Run the same experiment as previously using a QDA classifier. Compare the performances of the two classifiers.

3 Cat and grass

We are going to classify an image I which pixels consist of a cat and grass[1] (see figure 1). The objective is to learn a classifier that predicts the cat ($y = 1$) from the grass (class

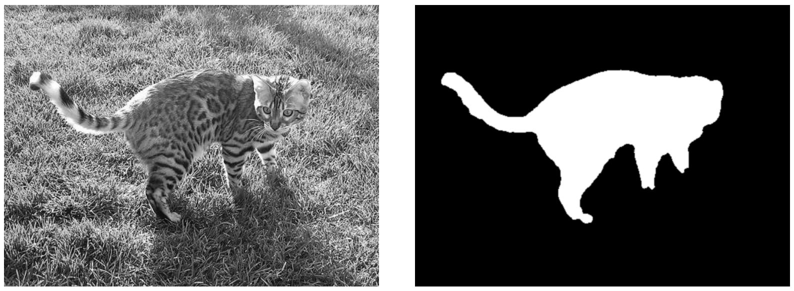

Figure 1: Left: the image, Right: labeled ground truth

$y = 0$) using 8×8 patches extracted from the image. For the pixel (i, j), the corresponding patch is defined as $I[i : i + 8, j : j + 8]$. The training samples are respectively provided in

[1]This exercise is mainly inspired from https://engineering.purdue.edu/ChanGroup/ECE595/files/homework3.pdf

the files `train_cat.txt` and `train_grass.txt`. Each row represents a vector of dimension $64 = 8 \times 8$, the size of a patch.

1. Load the data and form a binary classification problem with `X_train` $\in \mathbb{R}^{N \times d}$ and `Y_train` $\in \mathbb{R}^N$ the sample arrays and corresponding vector of labels.

```
train_cat = np.array(np.loadtxt("train_cat.txt", delimiter = ","))
train_grass = np.array(np.loadtxt("train_grass.txt", delimiter = ","))
# define sample arrays and label vector for cat and grass
train_cat = np.transpose(train_cat)
y_cat = np.ones(train_cat.shape[0])
train_grass = np.transpose(train_grass)
y_grass = np.zeros(train_grass.shape[0])
# form the classification problem
X_train = np.concatenate((train_cat, train_grass), axis=0)
Y_train = np.concatenate((y_cat, y_grass), axis=0)
```

2. Compute the number of samples of each class (cat vs grass). Is the classification problem balanced?

3. Learn a QDA model using the training data. Let call `clf_qda_cat_and_grass` this model.

4. We want to evaluate this classifier on the image `cat_grass.jpg`. Load the test image and normalize it such that the pixels are in the range $[0, 1]$.

```
Xtest = plt.imread("cat_grass.jpg")
Xtest = Xtest / 255
```

The class of pixel (i, j) is predicted as the assigned label to the corresponding path $Xtest[i : i + 8, j : j + 8]$. Notice that the resulting patches overlap. Write the code that outputs the classification result of `clf_qda_cat_and_grass` on `cat_grass.jpg`

```
M, N = Xtest.shape
Output_qda = np.zeros((M,N))
for i in range(M-8):
  for j in range(N-8):
    z = Xtest[i:i+8, j:j+8]
    # flatten z
    z = z.flatten()
    #classify the patch
    ypred = clf_qda_cat_and_grass.predict(z.reshape(1, -1)) #the reshape is required
         to comply with the parameters of "predict" method of clf_qda_cat
    Output_qda[i,j] = ypred
```

5. Plot the output (the output is an image) and compare to the ground truth image `truth.png`.

```
ground_truth = plt.imread("truth.png")
plt.subplot(1, 2, 1)
plt.imshow(ground_truth, cmap = "gray"); plt.title("True classes")
plt.subplot(1, 2, 2)
plt.imshow(Output_qda*255, cmap = "gray"); plt.title("Predictions")
```

Comment the results.
Compute the accuracy of your classifier. Hint: binarize the ground truth image to obtain a $\{0, 1\}$ image and compute the accuracy using the binarized image

```
from utility_bayes import binarize_ground_truth

ground_truth = binarize_ground_truth(ground_truth)
accuracy = ...
```

6. What will be the performance of a LDA model?