Homework 1

Homework Policy

- Use comments whenever necessary to explain your code.
- You will use Python as the programming language. However, you are NOT allowed to use built-in functions from Python libraries for Bayes classifier model building. Following are libraries that are NOT allowed:
  - scipy, Bio, Theano, Tensorflow, Keras, PyTorch and any other similar machine learning modules.

  Following are libraries that are allowed:
  - pandas, numpy, math, random, matplotlib, csv, etc... Basically anything that is not related to machine learning. When you are in doubt, simply ask.

  Note: You can use scikit-learn ONLY for cross validation using K-Fold. For other uses, it is not allowed.
- IMPORTANT: This is an individual assignment! You are expected to act according to Student Code of Conduct, which forbids all ways of cheating and plagiarism. It is okay to discuss the homework with others, but it is strictly forbidden to use all or parts of code from other students' codes or online sources and let others do all or part of your homework.
- Only electronic submissions through Ninova will be accepted. You only need to submit your Jupyter Notebook file. Any comments and discussions should be included in the same file.

Homework 1

Jupyter Notebook Installation

You need to have Python and Pip installed in your computer to install Jupyter Notebook.

Windows

On command prompt (cmd.exe with admin mode):

```
C:\**path**> python –m pip install jupyter
```

After changing your current folder to the folder which you want to work on (see 'cd' command):

```
C:\**your_working_folder**> jupyter notebook
```

Then it will be launched on your default browser

Ubuntu/Linux/Unix/Mac

On Terminal:

```
$ pip install notebook
```

Then launch with:

```
$ jupyter notebook
```

For more information: https://jupyter.org/install

Homework 1

Problem Description

You will carry out all the tasks below using **Ipython Notebook**. Simply add all your work to the provided template file **HW1_template.ipynb** using jupyter notebook.

You are given 2 different datasets and each has train and test set separately. You are also given an extra modified dataset for the job that you will find in **PART D**. Note that the provided "test2.csv" file is the test set for both Dataset 2 and Dataset 2(modified).

Number of samples	**Dataset 1**	**Dataset 2**	**Dataset 2(modified)**
Training set	200	1600	1640
Test set	100	400	400
Total	**300**	**2000**	**2040**

Part A (10 Points)

Examine the two training datasets 1 and 2 (In this part, Dataset2_modified is not used):

1. Import your training set (but only the training set for now). You can use pandas' DataFrame for this task (up to you), which is very convenient as it is easy to read .csv files using pandas and can extract data from it as numpy array. You can find a cheatsheet for it here.

2. For each class, calculate and visualize the covariance matrices of datasets 1 and 2 and interpret them (via comment outs) with your own words. Example figure can be seen in Figure 3 **(4 points)**

3. For each dataset, plot two overlaid transparent histograms for each feature (red for class 0 and blue for class 1) and interpret (via comment outs in Jupyter Notebook) them as well. This will allow you to examine the feature distribution within each class and compare them across classes. Example plot can be seen in Figure 1

    - In total you should generate 4 plots (each plot with 2 overlaid histograms): 2 datasets × 2 features each. **(4 points)**

4. Plot each training dataset (feature 1 on x-axis and feature 2 on y-axis). Example plots can be seen in Figure 2. **(2 points)**

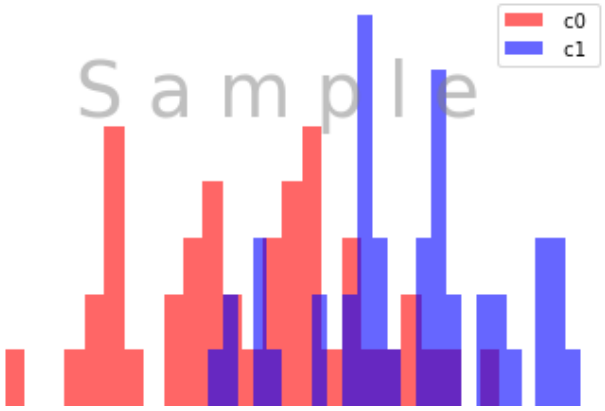

Figure 1: Example plot of distribution of feature 1.

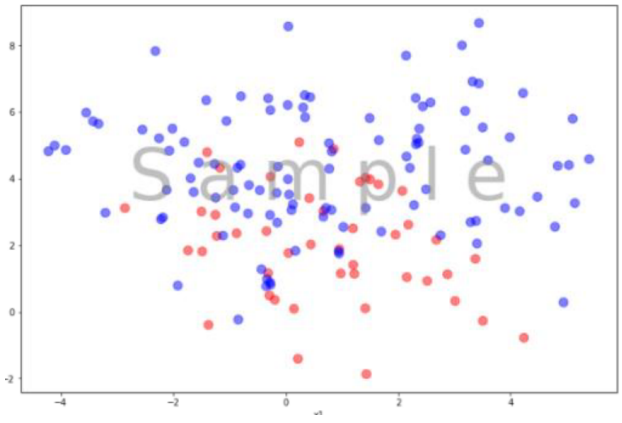

Figure 2: Example scatter plot of classes.

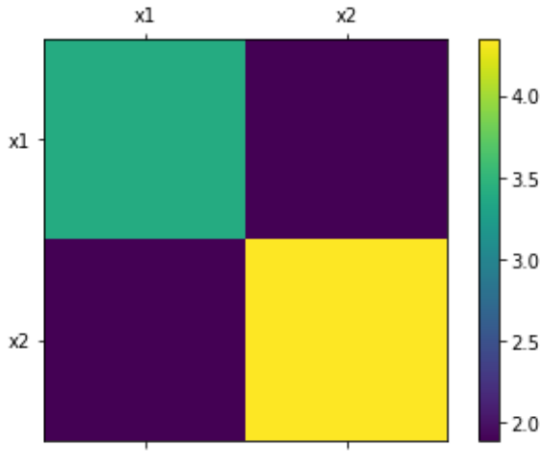

Figure 3: Example covariance matrix.

Homework 1

Part B: Evaluation of Bayes classifier using random single dataset split into train and test sets (45 Points)

You will use **Datasets 1 and 2** in this part. Assume that each class is generated from a multivariate Gaussian distribution.

1. Estimate the mean vectors for each class using the training data and consider the covariance matrices that you have calculated in Part A.: (just **print()** them)

$\mathbf{c}_1$ **(class 1)**	$\mathbf{c}_2$ **(class 2)**
Mean Vector $(\mu_1 = [??])$	Mean Vector $(\mu_2 = [??])$
Covariance Matrix $\Sigma_1 = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$	Covariance Matrix $\Sigma_2 = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$

2. Compute the linear discriminant function $g_i(x)$ for each class regarding the following cases: **(35 points)**

   - Case 1 **(15 points):** Equal covariance matrices ($\Sigma_1 = \Sigma_2$).
   - Case 2 **(20 points):** General case where the discriminant function $g_i(x)$ is quadratic.

   You may use the following code block for this part (or feel free to use your own code, it is up to you):

   Algorithm 1: Example code block

```
def trainBayes(trainingSamples, trainingLabels):
    if "covariance matrices are equal":
        g_iX = ...
    else: #general case
        g_iX = ...
    return g_iX
```

3. **For both datasets 1 and 2,** test your Bayes classifier **(10 points):**

   Apply the model to the test set after you import it and save the results in a vector.

   Note that, you should pretend like you don't know the labels (classes) of the test set while predicting.

Homework 1

Compare the predicted labels by Bayes classifier you have obtained with the actual (ground truth) labels of the test set and calculate the classification error rate as follows:

$$\mathbf{e} = \frac{\#\text{ mis Classification}}{\#\text{ testSamples}} \times 100$$

Homework 1

Part C: Evaluation of Bayes classifier using 3-fold cross-validation (25 Points)

In this part, you will use Dataset 1 and 2. Following steps will be applied to Datasets 1 and 2 **seperately**.

1. Combine the training set and the test set row-wise (concat) **(1 point)**
2. Apply 3-fold cross validation on this new dataset: **(20 point)**
   (a) Train your classifier and follow the same steps that given in **Part B.1, B.2 and B.3.**. To implement K-Fold, you can use sklearn's 'KFold' module (Click here for info).
      i. Compute means and covariance matrices
      ii. Compute $g_i(x)$
      iii. Find the classification error rate by applying your model to each left-out fold for testing
   (b) Save your result
3. Compute the average of your testing results **(1 Point)**
4. Compare and interpret the avg. result to the result that you have obtained in Part B (i.e., using the random single split of the dataset). Which evaluation strategy is most reliable? Discuss and justify. **(3 Points)**

Warning 1: There might be an issue with CV since it perturbs the covariance of the data as we shuffle the samples. For this reason, use the same random seed in Python: random.seed(1773).

Warning 2: Note also that the covariance matrix of each class will randomly change. In case, the covariance matrix becomes invertible, you can use a pseudo-inverse. There are Python codes (strategies) which estimate a pseudo-inverse of a non-invertible matrix. In case you use such codes, include the references in your Jupyter notebook and explain in brief how you generated the matrix pseudo-inverse. This can be added as a test in your code (i.e., whether the covariance matrix is invertible or not).

Homework 1

Part D: Evaluation of Bayes classifier on Dataset2_modified (20 Points)

1. Plot training Dataset2_modified (feature 1 on x-axis and feature 2 on y-axis). What do you notice in comparison with Dataset2 plot? **(5 points)**

2. Apply the work on **Part B** on the Dataset2_modified and compute the classification error on the test set of Dataset2_modified. **(5 points)**

3. Compare the results of Dataset2 in Experiment 1 (i.e., random single partition of the data) and those of Dataset2_modified. What can you conclude about Bayes classifier? **(10 points)**

.

Mathematical notation	Definition
$\mathcal{D}$	dataset
n	number of samples in a dataset $\mathcal{D}$
d	number of features
$\mathbf{x} \in \mathbb{R}^{d \times 1}$	feature vector or data point (sample)
$\mathbf{x}^{sample}_{feature}$	–
$\mathbf{x}^i \in \mathbb{R}^{d \times 1}$	i^{th} sample in the population
$\mathbf{x}^i_j \in \mathbb{R}$	j^{th} feature of i^{th} sample in the population
$\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$	covariance matrix of data population $\{\mathbf{x}^i\}_{i=1}^n$
$\lvert\mathbf{A}\rvert \in \mathbb{R}$	determinant of matrix A
$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(\frac{-1}{2}\frac{\lvert\lvert x-\mu\rvert\rvert_2^2}{\sigma^2})$	probability density function of a variable $x \in \mathbb{R}$
$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\lvert\mathbf{\Sigma}\rvert^{1/2}} exp[\frac{-1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)]$	probability density function of a multidimensional variable $\mathbf{x} \in \mathbb{R}^{d \times 1}$
$\mu \in \mathbb{R}^{d \times 1}$	sample mean $\mu = \frac{1}{n}\sum_{i=1}^n \mathbf{x}^i$
$\lvert\lvert\mathbf{x}-\mu\rvert\rvert_{\mathbf{\Sigma}^{-1}} = (\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu) \in \mathbb{R}$	Mahalanobis distance between $\mathbf{x}$ and μ
$\mathbf{I}_{d \times d} \in \mathbb{R}^{d \times d}$	identify matrix of size d ×d
$\lvert\lvert\mathbf{x}-\mu\rvert\rvert_{\mathbf{I}_{d \times d}} = (\mathbf{x}-\mu)^T(\mathbf{x}-\mu) \in \mathbb{R}$	Euclidean distance between $\mathbf{x}$ and μ also noted as L_2 norm $\lvert\lvert \cdot \rvert\rvert_2$
$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	discriminant Bayes function for class i when $\mathbf{\Sigma}_i = \sigma_i^2\mathbf{I}$ general case: when $\sigma_i^2 \neq \sigma_j^2$ for classes i and j (i.e., different means $\mu_i \neq \mu_j$ but constant variance for all data features in each class) special case: when $\sigma_i^2 = \sigma_j^2$ for classes i and j (i.e., different means $\mu_i \neq \mu_j$ but constant variances across all classes) (i.e., lines connecting means of different classes are perpendicular to decision boundaries) if $ln(p(c_i)) = ln(p(c_j))$, $g_i(\mathbf{x}) = -\lvert\lvert\mathbf{x}-\mu_i\rvert\rvert_2^2$
$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	discriminant Bayes function for class i when $\mathbf{\Sigma}_i = \mathbf{\Sigma}_j = \mathbf{\Sigma}$ (i.e., constant data feature covariance $\mathbf{\Sigma}$ across classes) (i.e., lines connecting means of different classes are not perpendicular to decision boundaries) if $ln(p(c_i)) = ln(p(c_j))$, $g_i(\mathbf{x}) = -\frac{1}{2}\lvert\lvert\mathbf{x}-\mu_i\rvert\rvert_{\mathbf{\Sigma}^{-1}}^2$
$g_i(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	quadratic discriminant function (decision boundaries are nonlinear)