

# CSCI 5521: Spring'17

## Introduction To Machine Learning

### Homework 2

(Due Fri, March 3, 11:55 pm)

- (40 points) Let  $\mathcal{X} = \{x_1, \dots, x_n\}$  be a set of  $n$  samples drawn i.i.d. from an univariate distribution with density function  $p(x|\theta)$ , where  $\theta$  is an unknown parameters. In general  $\theta$  will belong to a specified subset of  $\mathbb{R}$ , the set of reals. For the following choices of  $p(x|\theta)$ , derive the maximum likelihood estimate of  $\theta$  based on the samples  $\mathcal{X}$ :<sup>1</sup>
  - (10 points)  $p(x|\theta) = \frac{1}{\sqrt{2\pi\theta}} \exp\left(-\frac{x^2}{2\theta^2}\right), \theta > 0$ .
  - (10 points)  $p(x|\theta) = \frac{1}{\theta} \exp\left(-\frac{x}{\theta}\right), 0 \leq x < \infty, \theta > 0$ .
  - (10 points)  $p(x|\theta) = \theta x^{\theta-1}, 0 \leq x \leq 1, 0 < \theta < \infty$ .
  - (10 points)  $p(x|\theta) = \frac{1}{\theta}, 0 \leq x \leq \theta, \theta > 0$ .
- (20 points) Let  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$  be a set of  $n$  samples drawn i.i.d. from a multivariate Gaussian distribution in  $\mathbb{R}^d$  with mean  $\mu$  and covariance matrix  $\Sigma$ . Recall that the density function of a multivariate Gaussian distribution is given by:

$$p(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right].$$

- (10 points) Derive the maximum likelihood estimates for the mean  $\mu$  and covariance  $\Sigma$  based on the sample set  $\mathcal{X}$ .<sup>12</sup>
- (5 points) Let  $\hat{\mu}_n$  be the estimate of the mean. Is  $\hat{\mu}_n$  a biased estimate of the true mean  $\mu$ ? Clearly justify your answer by computing  $E[\hat{\mu}_n]$ .
- (5 points) Let  $\hat{\Sigma}_n$  be the estimate of the mean. Is  $\hat{\Sigma}_n$  a biased estimate of the true covariance  $\Sigma$ ? Clearly justify your answer by computing  $E[\hat{\Sigma}_n]$ .

#### Programming assignment:

The next problem involve programming. For Question 3, we will be using the 2-class classification datasets from `Boston50`, `Boston75`, and the 10-class classification dataset from `Digits` which were used in Homework 1.

- (40 points) We will develop a parametric classifier by modeling each class conditional distribution  $p(\mathbf{x}|C_i)$  as a multivariate Gaussian. In particular, using the training data, we will estimate the class prior probabilities  $p(C_i)$  and the class conditional probabilities  $p(\mathbf{x}|C_i)$  based on the estimated mean  $\hat{\mu}_i$  and the estimated covariance  $\hat{\Sigma}_i$  for each class  $C_i$ . The classification will be done based on the following discriminant function:

$$g_i(\mathbf{x}) = \log p(C_i) + \log p(\mathbf{x}|C_i).$$

---

<sup>1</sup> You have to show the details of your derivation. A correct answer without the details will not get any credit.

<sup>2</sup> You can use material from the `Matrix Cookbook` for your derivation.

We will develop code for `MultiGaussClassify` with corresponding `MultiGaussClassify.fit(X,y)` and `MultiGaussClassify.predict(X)` functions. Parameters for each class should be initialized to be zero mean and identity covariance, i.e.,  $\mu_i = \mathbf{0}$  and  $\Sigma_i = \mathbb{I}$ .

We will compare the performance of `MultiGaussClassify` with `LogisticRegression`<sup>3</sup> on three datasets: `Boston50`, `Boston75`, and `Digits`. Using `my_cross_val` with 5-fold cross-validation, report the error rates in each fold as well as the mean and standard deviation of error rates across folds for the two methods: `MultiGaussClassify` and `LogisticRegression`, applied to the three classification datasets: `Boston50`, `Boston75`, and `Digits`.

You will have to submit (a) **code** and (b) **summary of results**:

- (a) **Code:** You will have to submit code for `MultiGaussClassify()` as well as a wrapper code `q3()`.

For `MultiGaussClassify()`, you are encouraged to consult the code for `LinearSVC` (or `LogisticRegression`) in `scikit-learn` to build your code for `MultiGaussClassify`.<sup>4</sup> You need to make sure you have `__init__`, `fit`, and `predict` implemented in `MultiGaussClassify`. If you consult the code from `scikit-learn`, keep in mind that your code does not have to exactly follow the structure used in `scikit-learn` (e.g., class inheritance) since your code is expected to be much simpler than that. For example, for `Logistic Regression` in `sklearn`, you can look for the class definition of `LogisticRegression` in `logistic.py`, and the function definition of `__init__()`, `fit()`, etc. within that class. This is just to give you some hints on the code structure and some parameters, inputs and outputs you may need to consider in your own code for a new class `MultiGaussClassify()`. Your class will **NOT** inherit any base class in `sklearn`. Again, the three functions you must implement in this class are `__init__`, `fit`, and `predict`.

**The wrapper code** (main file) has no input and is used to prepare the datasets, and make calls to `my_cross_val(method,X,y,k)` to generate the error rate results for each dataset and each method. The code for `my_cross_val(method,X,y,k)` must be developed by yourself (e.g., code you made in HW1 with modifications as needed) and you cannot use `cross_val_score()` in `sklearn`. The results should be printed to terminal (not generating an additional file in the folder). Make sure the calls to `my_cross_val(method,X,y,k)` are made in the following order and add a print to the terminal before each call to show which method and dataset is being used:

1. `MultiGaussClassify` with `Boston50`;
2. `MultiGaussClassify` with `Boston75`;
3. `MultiGaussClassify` with `Digits`;
4. `LogisticRegression` with `Boston50`;
5. `LogisticRegression` with `Boston75`;
6. `LogisticRegression` with `Digits`.

\*For the wrapper code, you need to make a `q3.py` file for it, and one should be able to run your code by calling "python q3.py" in command line window.

- (b) **Summary of results:** For each dataset and each method, report the test set error rates for each of the  $k = 5$  folds, the mean error rate over the  $k$  folds, and the standard deviation

---

<sup>3</sup>You should use `LogisticRegression` from `scikit-learn`, similar to HW1.

<sup>4</sup>The exact location of the python code will depend on your implementation. For example, if you are using Anaconda, look for `LinearSVC` code in `Anaconda3\Lib\site-packages\sklearn\svm\classes.py` and `LogisticRegression` in `Anaconda3\Lib\site-packages\sklearn\linear_model\logistic.py`. Your code should be considerably simpler than these examples.

of the error rates over the  $k$  folds. Make a table to present the results for each method and each dataset (6 tables in total). Each column of the table represents a fold and add two columns at the end to show the overall mean error rate and standard deviation over the  $k$  folds.

**Additional instructions:** Code can only be written in Python (**not** IPython notebook); no other programming languages will be accepted. One should be able to execute all programs directly from command prompt (e.g., "`python q3.py`") without the need to run Python interactive shell first. Test your code yourself before submission and suppress any warning messages that may be printed. Your code must be run on a CSE lab machine (e.g., `cse-kh1260-01.cselabs.umn.edu`). Please make sure you specify the version of Python you are using as well as instructions on how to run your program in the README file (must be readable through a text editor such as Notepad). Information on the size of the datasets, including number of data points and dimensionality of features, as well as number of classes can be readily extracted from the datasets in `scikit-learn`. Each function must take the inputs in the order specified in the problem and display the output via the terminal or as specified.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. Please put comments in your code so that one can follow the key parts and steps in your code.

**Follow the rules strictly. If we cannot run your code, you will not get any credit.**

- **Things to submit**

1. hw2.pdf: A document which contains the solution to Problems 1, 2, and 3 including the summary of results for 3. This document must be in PDF format (no word, photo, etc. is accepted). If you submit a scanned copy of a hand-written document, make sure the copy is clearly readable, otherwise no credit may be given.
2. Python code for Problem 3 (must include the required `q3.py`).
3. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, the full Python version you are using, any assumptions you are making, and any other necessary details. The file must be readable by a text editor such as Notepad.
4. Any other files, except the data, which are necessary for your code.