CSci 5521: Fall'19
Introduction To Machine Learning

Homework 2

(Due Friday, October 18, 11:59 pm)

1. (30 points) Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of n samples drawn i.i.d. from an univariate distribution with density function $p(x|\theta)$, where θ is an unknown parameter. In general, θ will belong to a specified subset of $\mathbb{R}$, the set of reals. For the following choices of $p(x|\theta)$, derive the maxmimum likelihood estimate of θ based on the samples $\mathcal{X}$:[1]

   (a) (7.5 points) $p(x|\theta) = \frac{1}{\sqrt{2\pi}\theta} \exp\left(-\frac{x^2}{2\theta^2}\right), \theta > 0.$

   (b) (7.5 points) $p(x|\theta) = \frac{1}{\theta} \exp\left(-\frac{x}{\theta}\right), 0 \leq x < \infty, \theta > 0.$

   (c) (7.5 points) $p(x|\theta) = \theta x^{\theta-1}$, $0 \leq x \leq 1, 0 < \theta < \infty.$

   (d) (7.5 points) $p(x|\theta) = \frac{1}{\theta}$, $0 \leq x \leq \theta, \theta > 0.$

2. (20 points) Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$ be a set of n samples drawn i.i.d. from a multivariate Gaussian distribution in $\mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. Recall that the density function of a multivariate Gaussian distribution is given by:

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

   (a) (10 points) Derive the maximum likelihood estimates for the mean μ and covariance Σ based on the sample set $\mathcal{X}$.[1,2]

   (b) (5 points) Let $\hat{\mu}_n$ be the maximum likelihood estimate of the mean. Is $\hat{\mu}_n$ a biased estimate of the true mean μ? Clearly justify your answer by computing $E[\hat{\mu}_n]$.

   (c) (5 points) Let $\hat{\Sigma}_n$ be the maximum likelihood estimate of the covariance matrix. Is $\hat{\Sigma}_n$ a biased estimate of the true covariance Σ? Clearly justify your answer by computing $E[\hat{\Sigma}_n]$.

Programming assignment:

The next problem involves 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.

3. (50 points) We will develop two parametric classifiers by modeling each class's conditional distribution $p(\mathbf{x}|C_i)$ as multivariate Gaussians with (a) full covariance matrix Σ_i and (b) diagonal covariance matrix Σ_i. In particular, using the training data, we will compute the maximum likelihood estimate of the class prior probabilities $p(C_i)$ and the class conditional probabilities $p(x|C_i)$ based on the maximum likelihood estimates of the mean $\hat{\mu}_i$ and the

[1] You have to show the details of your derivation. A correct answer without the details will not get any credit.

[2] You can use material from the `Matrix Cookbook` for your derivation.

(full/diagonal) covariance $\hat{\Sigma}_i$ for each class C_i. The classification will be done based on the following discriminant function:

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

We will develop code for a class `MultiGaussClassify` with two key functions: `MultiGaussClassify.fit(self,X,y,diag)` and `MultiGaussClassify.predict(self,X)`. For `fit(self,X,y,diag)`, the inputs (X, y) are respectively the feature matrix and class labels, and `diag` is boolean (TRUE or FALSE) which indicates whether the estimated class covariance matrices should be a full matrix (`diag`=FALSE) or a diagonal matrix (`diag`=TRUE). For `predict(X)`, the input X is the feature matrix corresponding to the test set. For the class, the `__init__(self,k)` function can initialize the parameters for each class to be uniform prior, zero mean, and identity covariance, i.e., $p(C_i) = 1/k$, $\mu_i = \mathbf{0}$ and $\Sigma_i = \mathbb{I}$, $i = 1, \ldots, k$. Here, the number of classes k is passed as an argument to the constructor of `MultiGaussClassify`.

We will compare the performance of three models: (a, b) `MultiGaussClassify` with full class covariance matrices and diagonal covariance matrices, and (c) `LogisticRegression`[3] , each 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 three models applied to the three classification datasets

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`.[4] 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 the `MultiGaussClassify` class are `__init__`, `fit`, and `predict`.

The wrapper code `q3()` (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 yours (e.g., code you developed in HW1 with modifications as needed) and you cannot use `cross_val_score()` in `sklearn`. For the `method` argument in `my_cross_val`, you can

[3] You should use `LogisticRegression` from `scikit-learn`, similar to HW1.

[4] These can be found online at
github.com/scikit-learn/scikit-learn/blob/master/sklearn/svm/classes.py
and
github.com/scikit-learn/scikit-learn/blob/master/sklearn/linear_model/logistic.py
Your code should be considerably simpler than these examples.

call the method corresponding to `MultiGaussClassify` with full covariance matrix as just 'multigaussclassify' and the method corresponding to `MultiGaussClassify` with diagonal covariance matrix as 'multigaussdiagclassify.'

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,$\mathbf{y}$,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 full covariance matrix on `Boston50`; 2. `MultiGaussClassify` with full covariance matrix on `Boston75`; 3. `MultiGaussClassify` with full covariance matrix on `Digits`; 4. `MultiGaussClassify` with diagonal covariance matrix on `Boston50`; 5. `MultiGaussClassify` with diagonal covariance matrix on `Boston75`; 6. `MultiGaussClassify` with diagonal covariance matrix on `Digits`; 7. `LogisticRegression` with `Boston50`; 8. `LogisticRegression` with `Boston75`; 9. `LogisticRegression` with `Digits`.

For example, the first call to `my_cross_val`(`method`,X,$\mathbf{y}$,k) should result in the following output:

```
Error rates for MultiGaussClassify with full covariance matrix on Boston50:
Fold 1:  ###
Fold 2:  ###
...
Fold 5:  ###
Mean:  ###
Standard Deviation:  ###
```

(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 of the error rates over the k folds. Make a table to present the results for each method and each dataset (9 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. For example:

Error rates for MGC with full cov matrix on Boston50						
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	SD
#	#	#	#	#	#	#

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., "`python3 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., csel-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 your 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.