CSE574 Introduction to Machine Learning
Programming Assignment 3

Classification and Regression

Due Date: **November 19th 2024**
Maximum Score: 100

Note A zipped file containing skeleton Python script files and data is provided. Note that for each problem, you need to write code in the specified function withing the Python script file. **Do not use any Python libraries/toolboxes, built-in functions, or external tools/libraries that directly perform classification, regression, function fitting, or ridge regression**. Using any external or built in code will result in 0 points for the corresponding problem.

Evaluation We will evaluate your code by executing script.py file, which will internally call the problem specific functions. Also submit an assignment report (pdf file) summarizing your findings. In the problem statements below, the portions under REPORT heading need to be discussed in the assignment report.

Data Sets: Two data sets are provided:

1. A 2D sample data set in the file "sample.pickle" along with the class assignment.
2. A medical data set is provided in the file "diabetes.pickle" along with the target assignment. The input variables correspond to measurements (physical, physiological, and blood related) for a given patient and the target variable corresponds to the level of diabetic condition in the patient. It contains:
   - $\mathbf{x}_{\mathbf{train}}$ (242×64) and $\mathbf{y}_{\mathbf{train}}$ (242×1) for training.
   - $\mathbf{x}_{\mathbf{test}}$ (200×64) and $\mathbf{y}_{\mathbf{test}}$ (200×1) for testing.

Submission You are required to submit a single file called ***proj3.zip*** using UBLearns. File ***proj3.zip*** must contain 2 files: ***report.pdf*** and ***script.py***.

- Submit your report in a pdf format. Please indicate the **team members** on the top of the report.
- The code file should contain all implemented functions. Please do not change the name of the file.

Using UBLearns Submission: You should submit one solution per group through the project 3 submissions system in UBLearns.

Problem 1: Experiment with Gaussian Discriminators (10 code + 10 report = 20 points)

Implement **Linear Discriminant Analysis** (LDA) and **Quadratic Discriminant Analysis** (QDA). Implement two functions in Python: ldaLearn and qdaLearn which take a training data set (a feature matrix

and labels) and return the means and covariance matrix (or matrices). Implement two functions `ldaTest` and `qdaTest` which return the true labels for a given test data set and the accuracy using the true labels for the test data. The format of arguments and the outputs is provided in the base code.

REPORT 1.

Train both methods using the sample training data (**sample_train**). Report the accuracy of LDA and QDA on the provided test data set (**sample_test**). Also, plot the discriminating boundary for linear and quadratic discriminators. The code to plot the boundaries is already provided in the base code. Explain why there is a difference in the two boundaries.

Problem 2: Experiment with Linear Regression (5 code + 5 report = 10 Points)

Implement *ordinary least squares* method to estimate regression parameters by minimizing the squared loss.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \tag{1}$$

In matrix-vector notation, the loss function can be written as:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \tag{2}$$

where $\mathbf{X}$ is the input data matrix, $\mathbf{y}$ is the target vector, and $\mathbf{w}$ is the weight vector for regression.

Note that this is same as maximizing the log-likelihood in the Bayesian setting. You need to implement the function `learnOLERegression`. Also implement the function `testOLERegression` to apply the learnt weights for prediction on both training and testing data and to calculate the mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \tag{3}$$

REPORT 2.

Calculate and report the MSE for training and test data for two cases: first, without using an intercept (or bias) term, and second with using an intercept. Which one is better?

Problem 3: Experiment with Ridge Regression (10 code + 10 report = 20 Points)

Implement parameter estimation for ridge regression by minimizing the regularized squared loss as follows:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w} \tag{4}$$

In matrix-vector notation, the squared loss can be written as:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{1}{2}\lambda\mathbf{w}^\top\mathbf{w} \tag{5}$$

You need to implement it in the function `learnRidgeRegression`.

REPORT 3.

Calculate and report the MSE for training and test data using ridge regression parameters using the the `testOLERegression` function that you implemented in Problem 2. Use data with intercept. Plot the errors on train and test data for different values of λ. Vary λ from 0 (no regularization) to 1 in steps of 0.01. Compare the relative magnitudes of weights learnt using OLE (Problem 2) and weights learnt using ridge regression. Compare the two approaches in terms of errors on train and test data. What is the optimal value for λ and why?

Problem 4: Using Gradient Descent for Ridge Regression Learning (20 code + 5 report = 25 Points)

As discussed in class, regression parameters can be calculated directly using analytical expressions (as in Problem 2 and 3). To avoid computation of $(\mathbf{X}^\top\mathbf{X})^{-1}$, another option is to use gradient descent to minimize the loss function (or to maximize the log-likelihood) function. In this problem, you have to implement the gradient descent procedure for estimating the weights $\mathbf{w}$.

You need to use the `minimize` function (from the **scipy** library). You need to implement a function `regressionObjVal` to compute the regularized squared error (See (4)) and its gradient with respect to $\mathbf{w}$. In the main script, this objective function will be used within the minimizer.

REPORT 4.

Plot the errors on train and test data obtained by using the gradient descent based learning by varying the regularization parameter λ. Compare with the results obtained in Problem 3.

Problem 5: Non-linear Regression (10 code + 5 report = 15 Points)

In this problem we will investigate the impact of using higher order polynomials for the input features. For this problem use the third variable as the only input variable:

```
x_train = x_train[:,2]
x_test = x_test[:,2]
```

Implement the function `mapNonLinear` which converts a single attribute $\mathbf{x}$ into a vector of p attributes, $1, x, x^2, \ldots, x^p$.

REPORT 5.

Using the $\lambda = 0$ and the optimal value of λ found in Problem 3, train ridge regression weights using the non-linear mapping of the data. Vary p from 0 to 6. Note that $p = 0$ means using a horizontal line as the regression line, $p = 1$ is the same as linear ridge regression. Compute the errors on train and test data. Compare the results for both values of λ. What is the optimal value of p in terms of test error in each setting? Plot the curve for the optimal value of p for both values of λ and compare.

Problem 6: Interpreting Results (0 code + 10 report = 10 points)

Using the results obtained for previous 4 problems, make final recommendations for anyone using regression for predicting diabetes level using the input features.

REPORT 6.

Compare the various approaches in terms of training and testing error. What metric should be used to choose the best setting?
