ECE368: Probabilistic Reasoning

Lab 2: Classification with Gaussian Models and Bayesian Linear Regression

1 Classification with Gaussian Models

In the first part of the lab, we use linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) on the 2D data in ldaqda.zip, and visualize the classification results for class 1 and class 2 based on the features in the data.

Suppose that the dataset contains N samples. Let $\mathbf{x}_n = [h_n, w_n]$ be the feature vector, where h_n denotes feature 1 and w_n denotes feature 2 of the n-th data point. Let y_n denote the class label where $y_n = 1$ or $y_n = 2$. We model the class prior as $p(y_n = 1) = \pi$ and $p(y_n = 2) = 1 - \pi$. For this problem, let $\pi = 0.5$.

For the class conditional distributions, let $\boldsymbol{\mu}_1$ be the mean of $\mathbf{x}_n$ if class label $y_n = 1$, and let $\boldsymbol{\mu}_2$ be the mean of $\mathbf{x}_n$ if class label $y_n = 2$. For LDA, a common covariance matrix is shared by both classes, which is denoted by $\mathbf{\Sigma}$; for QDA, different covariance matrices are used for class 1 and class 2, which are denoted by $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$, respectively.

Download ldaqda.zip from Quercus and unzip the file. The dataset for training is in file trainData.txt, whereas the dataset for testing is in file testData.txt. Each file uses the same format to represent the data: the first column corresponds to the class labels, the second column corresponds to feature 1 values, and the third column corresponds to feature 2 values.

Please answer the questions below and complete the two functions in ldaqda.py. File util.py contains a few functions/classes that will be useful in writing the code.

Questions

1. Training and visualization. We estimate the parameters in LDA and QDA from the training data in trainData.txt and visualize the LDA/QDA model.
   (a) Please write down the maximum likelihood estimates of the parameters $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\mathbf{\Sigma}$, $\mathbf{\Sigma}_1$, and $\mathbf{\Sigma}_2$ as functions of the training data $\{\mathbf{x}_n, y_n\}, n = 1, 2, \ldots, N$. The indicator function $\mathbb{I}(\cdot)$ may be useful in your expressions.
   (b) Once the above parameters are obtained, you can design a classifier to make a decision on the class label y of the new data $\mathbf{x}$. The decision boundary can be written as a linear equation of $\mathbf{x}$ in the case of LDA, and a quadratic equation of $\mathbf{x}$ in the case of QDA. Please write down the expressions of these two boundaries.
   (c) Complete function discrimAnalysis in file ldaqda.py to visualize LDA and QDA. Please plot one figure for LDA and one figure for QDA. In both plots, the horizontal axis is Feature 1 with range $[-4, 6]$ and the vertical axis is Feature 2 with range $[-5, 5]$. Each figure should contain: 1) N colored data points $\{\mathbf{x}_n, n = 1, 2, \ldots, N\}$ with the color indicating the corresponding class labels (e.g., blue represents class 1 and red represents class 2); 2) the contours of the the conditional Gaussian distribution for each class (To create a contour plot, you need first build a two-dimensional grid for the range $[-4, 6] \times [-5, 5]$ by using the function np.meshgrid. You then compute the conditional Gaussian density at each point in the grid for each class. Finally use the function plt.contour, which takes the two-dimensional grid and the conditional Gaussian density on the grid as inputs to automatically produce the contours.); 3) the decision boundary, which can also be created by using plt.contour with appropriate contour level.

2. Testing. We test the obtained LDA/QDA model on the testing data in testData.txt. Complete function misRate in file ldaqda.py to compute the misclassification rates for LDA and QDA, defined as the total percentage of the misclassified samples (both classes) over all samples.

2 Bayesian Linear Regression

In this part of the lab, we use Bayesian regression to fit a linear model. Consider a linear model of the form

$$z = a_1 x + a_0 + w, \tag{1}$$

where x is the scaler input variable, and $\mathbf{a} = (a_0, a_1)^T$ is the vector-valued parameter with unknown entries a_0, a_1, and w is the additive Gaussian noise:

$$w \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

where σ^2 is a known parameter.

Suppose that we have access to a training dataset containing N samples $\{x_1, z_1\}, \{x_2, z_2\}, \dots, \{x_N, z_N\}$. We aim to estimate the parameter $\mathbf{a}$ by finding its posterior distribution. When the training finishes, we make predictions based on new inputs. We consider a Bayesian approach, which models the parameter $\mathbf{a}$ as a zero mean isotropic Gaussian random vector whose probability distribution is expressed as

$$p(\mathbf{a}) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \beta & 0 \\ 0 & \beta \end{bmatrix} \right), \tag{3}$$

where β is a known hyperparameter.

Download reg.zip from Quercus and unzip the file. First, open the file generate_data.py and replace the student numbers in the code with your actual student numbers. Then, run generate_data.py to create your personalized training data (training.txt) and make sure to record the ground truth values for a_0 and a_1 that are printed after running generate_data.py.

File training.txt contains the training data: the first column is the inputs; the second column is the targets. The training data is generated from $z = a_1 x + a_0 + w$ where the actual values of a_1 and a_0 are available by running generate_data.py. Please answer the questions below and complete regression.py. File util.py contains a few useful functions.

Questions

1. Express the posterior distribution $p(\mathbf{a}|z_1, \dots, z_N; x_1, \dots, x_N)$ using $\sigma^2, \beta,\ x_1, z_1, x_2, z_2, \dots, x_N, z_N$. This notation should be read as "the conditional distribution of $\mathbf{a}$ given $z_1, \dots, z_N$ under the parameters $x_1, \dots, x_N$". In this problem $z_1, \dots, z_N$ and $\mathbf{a}$ are random variables while $x_1, \dots x_N$ are unknown constant parameters.

2. Let $\sigma^2 = 0.1$ and $\beta = 1$. Based on the posterior distribution obtained in the last question, draw four contour plots corresponding to $p(\mathbf{a})$, $p(\mathbf{a}|z_1; x_1)$, $p(\mathbf{a}|z_1, \dots, z_5; x_1 \dots x_5)$, and $p(\mathbf{a}|z_1, \dots, z_{100}; x_1 \dots x_{100})$. In all contour plots, the x-axis represents a_0, and the y-axis represents a_1. The range is set as $[-1, 1] \times [-1, 1]$. In each figure, also draw the true value of $\mathbf{a}$.

3. Suppose that there is a new input x, for which we want to predict the target value z. Write down the distribution of the prediction z, i.e., $p(z|z_1, \dots, z_N; x, x_1, \dots x_N)$.

4. Let $\sigma^2 = 0.1$ and $\beta = 1$. Suppose that the set of the new inputs is $\{-4, -3.8, -3.6, \dots, 0, \dots, 3.6, 3.8, 4\}$. Plot three figures corresponding to the following three cases:

(a) The predictions are based on one training sample, i.e., based on $p(z|z_1; x, x_1,)$.

(b) The predictions are based on 5 training samples, i.e., based on $p(z|z_1, \ldots, z_5; x, x_1, \ldots, x_5)$.

(c) The predictions are based on 100 training samples, i.e., based on $p(z|z_1, \ldots, z_{100}; x, x_1, \ldots, x_{100})$.

In all figures, the x-axis is the input, the y-axis is the target, and the range is set as $[-4, 4] \times [-4, 4]$. Each figure should contain three components: 1) the new inputs and the predicted targets; 2) a vertical interval at each predicted target, indicating the range within one standard deviation; 3) the training sample(s) that are used for the prediction. Use `plt.errorbar` for 1) and 2); use `plt.scatter` for 3).

This lab is adapted from the originals by Greg Wornell and Wei Yu.