ECE368: Probabilistic Reasoning

Lab 1: Classification with Binary and Gaussian Models

1 Naïve Bayes Classifier for Spam Filtering

In the first part of the lab, we use a Naïve Bayes Classifier to build a spam email filter based on binary feature vectors. Suppose that we have a training set containing N emails, and each email n is represented by $\{\mathbf{x}_n, y_n\}, n = 1, 2, \ldots, N$, where y_n is the class label which takes the value

$$y_n = \begin{cases} 1 & \text{if email } n \text{ is spam,} \\ 0 & \text{if email } n \text{ is non-spam (also called ham),} \end{cases} \tag{1}$$

and $\mathbf{x}_n$ is a feature vector of the email n. The feature vector is binary and is defined as follows. Let $\{w_1, \ldots, w_D\}$ be the set of the words (called the vocabulary) that appear at least once in the training set. The feature vector $\mathbf{x}_n$ is defined as a D-dimensional vector $\mathbf{x}_n = [x_{n1}, x_{n2}, \ldots, x_{nD}]$, where each entry $x_{nd}, d = 1, 2, \ldots, D$ takes the binary value

$$x_{nd} = \begin{cases} 1 & \text{if } w_d \text{ occurs in email } n, \\ 0 & \text{if } w_d \text{ does not occur in email } n. \end{cases} \tag{2}$$

We use a generative approach to build the spam/ham classifier. The prior class distribution $p(y_n)$ is modeled as

$$p(y_n = 1) = \pi, \tag{3}$$
$$p(y_n = 0) = 1 - \pi, \tag{4}$$

where π is a fixed parameter (e.g., 0.5). The class conditional distribution $p(x_{nd}|y_n)$ is modeled as

$$p\left(x_{nd}|y_n = 1\right) = \text{Bern}(p_d), \tag{5}$$
$$p\left(x_{nd}|y_n = 0\right) = \text{Bern}(q_d), \tag{6}$$

where $\text{Bern}(p_d)$ and $\text{Bern}(q_d)$ are Bernoulli distributions with the probability of being 1 as p_d and q_d, respectively. In words, (5) means given that an email is spam, the word w_d occurs with probability p_d; (6) means given that an email is ham, the word w_d occurs in the email with probability q_d. Note that p_d and q_d are different, which gives us a way to classify spam vs. ham. For example, words like "dollar","winner" would be more likely to occur in spam than in ham. Both p_d and q_d are parameters that should be estimated from the training data.

The key assumption of a Naïve Bayes Classifier is that we assume given the class label, the probability distributions of the different elements of the feature vector are conditionally independent. In other words, the joint probability distribution of the class labels and the features can be written as

$$p\left(\mathbf{x}_1, \ldots, \mathbf{x}_N, y_1, \ldots, y_N\right) = \prod_{i=1}^{N} \left[p\left(y_n\right) \prod_{d=1}^{D} p\left(x_{nd}|y_n\right) \right]. \tag{7}$$

In the following, we first estimate the parameters of (7), i.e., $p_d, q_d, d = 1, 2, \ldots, D$, using the training set; we then build a classifier based on (7) and make predictions on the testing set.

Download classifier.zip under Files/Labs/Lab1/ on Quercus and unzip the file. The spam emails for training are in the subfolder /data/spam/. The ham emails for training are in the subfolder /data/ham/. The unlabeled emails for testing are in the subfolder /data/testing/.

Please answer the questions below and complete the routine classifier.py. File util.py contains a few functions/classes that will be helpful in writing the code for the classifier.

Questions

1. Training. We estimate the parameters p_d and $q_d, d = 1, 2, \ldots, D$ from the training data as follows.

   (a) The maximum likelihood estimates of p_d and q_d are not the most appropriate to use when the probabilities are very close to 0 or to 1. For example, some words that occur in one class may not occur at all in the other class. In this problem, we use the technique of "Laplace smoothing" to deal with this problem. Please write down such an estimator for p_d and q_d as functions of the training data $\{\mathbf{x}_n, y_n\}, n = 1, 2, \ldots, N$ using Laplace smoothing.

   (b) Complete the function learn_distributions in file classifier.py. In learn_distributions, you first build the vocabulary $\{w_1, \ldots, w_D\}$ by accounting for all the words that appear in the training set at least once; you then estimate p_d and $q_d, d = 1, 2, \ldots, D$ using your expressions in part (a).

2. Testing. We classify the unlabeled emails in /data/testing/ using the trained classifier.

   (a) Let $\{\mathbf{x}, y\}$ be a data point from the testing set whose class label y is unknown. Write down the posterior distribution $p(y|\mathbf{x})$ based on (7) using the parameters p_d and q_d. Please assume that $\pi = 0.5$. To avoid numerical underflow, you should work with the log probability $\log p(y|\mathbf{x})$. Specify the maximum a posterior (MAP) rule to decide whether $y = 1$ or $y = 0$ based on the feature vector $\mathbf{x}$.

   (b) Complete the function classify_new_email in file classifier.py to implement the MAP rule, and run it on the testing set. There are two types of errors in classifying unlabeled emails: Type 1 error is defined as the event that a spam email is misclassified as ham; Type 2 error is defined as the event that a ham email is misclassified as spam. Write down the numerical values of these two numbers of errors made by your classifier on the testing data.

   (c) In practice, Type 1 error and Type 2 error lead to difference consequences (or costs). Therefore, we may wish to trade off one type of error against the other in designing the classifier. For example, we usually want to achieve a very low Type 2 error since the cost of missing a useful email can be severe, while we can tolerate a relative high Type 1 error as it merely causes inconvenience. Please provide a way to modify the decision rule in the classifier such that these two types of error can be traded off. In other words, change the decision rule in a way such that Type 2 error would decrease at a cost of Type 1 error, and vice versa. Test your method on the testing set and provide the following plot: Let the x-axis be the number of Type 1 errors and the y-axis be the number of Type 2 errors in the testing data set. Plot at least 10 points corresponding to different pairs of Type 1 and Type 2 errors, as a result of adjusting the classification rule. The two end points of the plot should be: 1) the one with zero Type 1 error; and 2) the one with zero Type 2 error. The code should be included in file classifier.py.

We acknowledge Prof. Greg Wornell of MIT in the process of creating this lab, and the following reference from which the training and test data are taken: V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam Filtering with Naive Bayes – Which Naive Bayes?" *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006.*

2 Linear/Quadratic Discriminant Analysis for Height/Weight Data

When the feature vector is real-valued (instead of binary), a Gaussian vector model is appropriate. In this part of the lab, we use linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) for the height/weight data of people, and visualize the classification of male and female persons based on height and weight.

Suppose that the data set contains N samples. Let $\mathbf{x}_n = [h_n, w_n]$ be the feature vector, where h_n denotes the height and w_n denotes the weight of a person indexed by n. Let y_n denote the class label. Here $y_n = 1$ is male, and $y_n = 2$ is female. 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 $\boldsymbol{\mu}_m$ be the mean of $\mathbf{x}_n$ if class label y_n is male, and let $\boldsymbol{\mu}_f$ be the mean of $\mathbf{x}_n$ if class label y_n is female. For LDA, a common covariance matrix is shared by both classes, which is denoted by $\boldsymbol{\Sigma}$; for QDA, different covariance matrices are used for male and female, which are denoted by $\boldsymbol{\Sigma}_m$ and $\boldsymbol{\Sigma}_f$, respectively.

Download ldaqda.zip under Files/Labs/Lab1/ on Quercus and unzip the file. The data set for training is in file trainHeightWeight.txt, whereas the data set for testing is in file testHeightWeight.txt. Each file uses the same format to represent the data: the first column corresponds to the class labels, the second column corresponds to the heights, and the third column corresponds to the weights.

Please answer the questions below and complete function 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 trainHeightWeight.txt and visualize the LDA/QDA model.

   (a) Please write down the maximum likelihood estimates of the parameters $\boldsymbol{\mu}_m$, $\boldsymbol{\mu}_f$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}_m$, and $\boldsymbol{\Sigma}_f$ 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 the height with range $[50, 80]$ and the vertical axis is the weight with range $[80, 280]$. 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 male and red represents female); 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 $[50, 80] \times [80, 280]$ by using function plt.meshgrid. You then compute the conditional Gaussian density at each point in the grid for each class. Finally use 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 testHeightWeight.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 male and female) over all samples.

We acknowledge the following text from which the data are taken: K. Murphy, *Machine Learning: A Probabilistic Approach*, MIT Press, 2012.