ECE368: Probabilistic Reasoning

Lab 1: Classification with Multinomial 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 whether and how many times each word in a fixed vocabulary occurs in the email. Suppose that we need to classify a set of 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. We use a multinomial model to construct the feature vector $\mathbf{x}_n$. Let $\mathcal{W} = \{w_1, w_2, \ldots, w_D\}$ 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 $x_{nd}, d = 1, 2, \ldots, D$ is the number of occurrences of word w_d in email n. Thus the total number of words in email n can be expressed as $l_n = x_{n1} + x_{n2} + \ldots + x_{nD}$.

We assume that each email n of length l_n is generated by a sequence of l_n independent events that randomly draw words from the vocabulary $\mathcal{W}$. (This is known as the naïve Bayes assumption.) For each event, let $p(w_d \mid y_n = 1)$ be the probability that word w_d is picked, given that the email belongs to spam; let $p(w_d \mid y_n = 0)$ be the probability that word w_d is picked, given that the email belongs to ham. Note that $p(w_d \mid y_n = 1)$ and $p(w_d \mid y_n = 0)$ 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. Also, note that both $p(w_d \mid y_n = 1), d = 1, 2, \ldots, D$ and $p(w_d \mid y_n = 0), d = 1, 2, \ldots, D$ should sum to one, i.e.,

$$\sum_{d=1}^{D} p(w_d \mid y_n = 1) = 1, \tag{2}$$

$$\sum_{d=1}^{D} p(w_d \mid y_n = 0) = 1. \tag{3}$$

The probabilities $p(w_d \mid y_n = 1)$, $p(w_d \mid y_n = 0), d = 1, \ldots, D$ should be learned from the training data.

We make use of the word frequencies to model each email n probabilistically. Since each word in the email is seen as independently drawn from the vocabulary $\mathcal{W}$, the distribution of the feature vector $\mathbf{x}_n$ given label y_n can be seen as a multinomial distribution as follows,

$$p(\mathbf{x}_n \mid y_n) = \frac{(x_{n1} + x_{n2} + \ldots + x_{nD})!}{(x_{n1})!(x_{n2})! \ldots (x_{nD})!} \prod_{d=1}^{D} p(w_d \mid y_n)^{x_{nd}}. \tag{4}$$

We assume that the prior class distribution $p(y_n)$ is modeled as

$$p(y_n = 1) = \pi, \tag{5}$$

$$p(y_n = 0) = 1 - \pi, \tag{6}$$

where π is a fixed parameter (e.g., 0.5).

In the following, we first estimate the probabilities $p(w_d \mid y_n = 1)$, $p(w_d \mid y_n = 0), d = 1, \ldots, D$ using the training set; we then build a classifier based on Bayes' rule and make predictions on the testing set.

Download classifier.zip under Modules/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 conditional probability distribution of the D-ary random variable as specified by $p(w_d \mid y_n = 1)$ and $p(w_d \mid y_n = 0), d = 1, \ldots, D$, from the training data using a *bag-of-words* model as follows. For notational simplicity, we define $p_d = p(w_d \mid y_n = 1)$ and $q_d = p(w_d \mid y_n = 0)$.

   (a) We put all the words from the spam emails in the training set in a bag and simply count the number of occurrences of each word w_d, $d = 1, \cdots, D$. We do the same for ham emails. The maximum likelihood estimates of p_d and q_d based on these counts 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 for the D-ary random variable.

   (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 maximum a posterior (MAP) rule to decide whether $y = 1$ or $y = 0$ based on the feature vector $\mathbf{x}$. The d-th entry of $\mathbf{x}$ is denoted by x_d. Please incorporate p_d and q_d in your expression. Please think carefully how to treat words that do not appear in either ham or spam training sets. Please assume that $\pi = 0.5$.

   (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. To avoid numerical underflow in your code, please work with the log probability $\log p(y|\mathbf{x})$ in your code.

   (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.

(d) Why do we need Laplace smoothing? Briefly explain what would go wrong if we use maximum likelihood estimation in the training process, by considering a scenario in which a testing email contains *both* a word w_1 that appears *only* in the ham training set (but not in the spam training set), *and* a word w_2 that appears *only* in the spam training set (but not in the ham training set). How does Laplace smoothing resolve this issue?

The training and test data for this problem are taken from 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 results 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 Modules/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 np.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.

The data for this problem are taken from: K. Murphy, *Machine Learning: A Probabilistic Approach*, MIT Press, 2012.