Machine Learning: Homework 3

Instructions

- **Collaboration policy:** Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in an undergraduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online on NYU Classes. You need to submit (1) a PDF which contains the solutions to all questions (2) `x.py` or `x.ipynb` files for the programming questions. We recommend that you use LaTeX, but we will accept scanned / pictured solutions as well.

Problem 1: Logistic Regression

In problem1 and problem2, we cover two probabilistic linear classifiers we have covered in class so far. First, a discriminative linear classifier: logistic regression. Second, a generative linear classifier: Linear discriminant analysis (LDA). Both the algorithms find a linear decision boundary that separates the data into two classes, but make different assumptions. Our goal in this problem is to get a deeper understanding of the similarities and differences of these two algorithms.

1. **[15 Points]** We have see in the lecture that given training data set (x_i, y_i), the cross entropy loss for logistic regression is

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(p_w(x_i)) + (1-y_i)log(1-p_w(x_i))]$$

Recall that in lecture we were able to prove the gradient of $J(w)$ w.r.t. w is

$$\frac{\partial J(w)}{\partial w} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - p_w(x_i))x_i$$

where w and x_i are all $p \times 1$ vectors.

We have provided two data files `logistic_x.txt, logistic_y.txt`. These files contain the inputs ($x_i \in R^2$) and outputs ($y_i \in \{-1, 1\}$), respectively for a binary classification problem, with one training example per row. Implement gradient descent method to optimize $J(w)$, and apply it to fit a logistic regression model to the given data (remember to transform the target variable y to ($y_i \in \{0, 1\}$)). What are the coefficients w resulting from your fit? (Remember to include the intercept term.)

Solution: See code file.

2. **[15 Points]** Plot the training data (your axes should be x_1 and x_2, corresponding to the two coordinates of the inputs, and you should use a different symbol for each point plotted to indicate whether that example had label 1 or -1). Also plot on the same figure the decision boundary fit by logistic regression. (This should be a straight line showing the boundary separating the region where $p_w(x) > 0.5$ from where $p_w(x) \leq 0.5$)

Problem 2: Linear Discriminative Analysis

Recall that in linear discriminant analysis (LDA), we make prediction based on the following posterior probability

$$P(y=1|x) = \frac{P(x|y=1) * P(y=1)}{P(x|y=1) * P(y=1) + P(x|y=0)P(y=0)}$$

$$P(x|y=0) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0))$$

$$P(x|y=1) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1))$$

Here μ_0, μ_1, Σ and the prior probability $\phi = P(y=1)$ are our model parameters (Note that for LDA we assume the negtive class and the positive class share the same co-variance matrix Σ)

1. **[15 Points]** Suppose we have already learned $\phi, \mu_0, \mu_1, \Sigma$ and now want to predict y given a new point x. To show that LDA results in a classifier that has a linear decision boundary, show the posterior distribution can be written as

   $$P(y=1|x;\mu_0,\mu_1,\Sigma) = \frac{1}{1+exp(-\theta_0+\theta^T x)}$$

   where $\theta \in R^d$ and $\theta_0 \in R$ are appropriate functions of $\phi, \mu_0, \mu_1, \Sigma$

   Solution:
   The decision boundary occures when:

   $$P(y=1|x;\mu_0,\mu_1,\Sigma) = P(y=0|x;\mu_0,\mu_1,\Sigma) \tag{1}$$

   $$P(y=1)\cdot P(X|Y=1) = P(y=0)\cdot P(X|Y=0) \tag{2}$$

   $$2\ln\frac{\phi}{1-\phi} = (x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - (x-\mu_1)^T\Sigma^{-1}(x-\mu_1) \tag{3}$$

   $$2(\mu_1-\mu_0)^T\Sigma^{-1}x = 2\ln\frac{\phi}{1-\phi} + (\mu_0\Sigma^{-1}\mu_0) - (\mu_1\Sigma^{-1}\mu_1) \tag{4}$$

   Here we find

   $$\theta^T = (\mu_1-\mu_0)^T\Sigma^{-1},\ \theta_0 = \ln\frac{\phi}{1-\phi} + \frac{1}{2}[(\mu_0\Sigma^{-1}\mu_0) - (\mu_1\Sigma^{-1}\mu_1)]$$

2. **[20 Points]** Coding problem. In `gda.py`, fill in the code to calculate $\phi, \mu_0, \mu_1, \Sigma$. Note that the MLE for μ_0, μ_1, Σ can be obtrained using

   $$\hat{\mu}_0 = \frac{\sum_{X_i \in C0} x_i}{n_0}$$

   $$\hat{\mu}_1 = \frac{\sum_{X_i \in C1} x_i}{n_1}$$

   $$\hat{\Sigma} = \frac{\sum_{X_i \in C0}(x_i-\hat{\mu}_0)(x_i-\hat{\mu}_0)^T + \sum_{X_i \in C1}(x_i-\hat{\mu}_1)(x_i-\hat{\mu}_1)^T}{n}$$

   where n_0 and n_1 are the number of training data points in the negative and positive class respectively. After we get he parameter estimation for $\phi\mu_0, \mu_1, \Sigma$, we can make predictions based on the posterior probability. Fit the model using the breast cancer data that is available in `sklearn`. Plot the testing data and use a different symbol to indicate the ground truth label. Also plot on the same figure the decision boundary given by LDA.

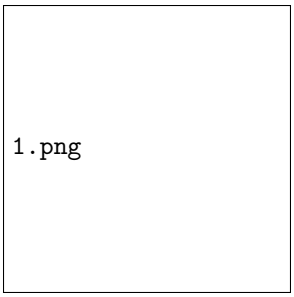

Figure 1: Sample e-mail in SpamAssassin corpus before pre-processing.

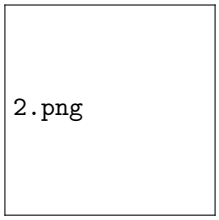

Figure 2: Preprocessed version of the sample email from Figure1.

Problem 3: Naive Bayes

In this problem we will implement the Naive Bayes classifier and apply it to spam email classification problem. Save your code in `NaiveBayes.py`.

Data files: We have provided you with two files: `spam_train.txt`, `spam_test.txt`. Each row of the data files corresponds to a single email. The first column gives the label (1=spam, 0=not spam).

Pre-processing: The dataset included for this exercise is based on a subset of the SpamAssassin Public Corpus. Figure 1 shows a sample email that contains a URL, an email address (at the end), numbers, and dollar amounts. While many emails would contain similar types of entities (e.g., numbers, other URLs, or other email addresses), the specific entities (e.g., the specific URL or specific dollar amount) will be different in almost every email. Therefore, one method often employed in processing emails is to "normalize" these values, so that all URLs are treated the same, all numbers are treated the same, etc. For example, we could replace each URL in the email with the unique string "httpaddr" to indicate that a URL was present. This has the effect of letting the spam classifier make a classification decision based on whether any URL was present, rather than whether a specific URL waspresent. This typically improves the performance of a spam classifier, since spammers often randomize the URLs, and thus the odds of seeing any particular URL again in a new piece of spam is very small.

We have already implemented the following email preprocessing steps: lower-casing; removal of HTML tags; normalization of URLs, e-mail addresses, and numbers. In addition, words are reduced to their stemmed form. For example, "discount", "discounts", "discounted" and "discounting" are all replaced with "discount". Finally, we removed all non-words and punctuation. The result of these preprocessing steps is shown in Figure 2. below.

1. **[20 Points]** In order to train a Naive Bayes classier, we need to transform the data into feature vectors. To do that we first build a vocabulary list using only the 5000 e-mail training set by finding all words that occur across the training set. Note that we assume that the data in the test sets is completely unseen when we train our model, and thus we do not use any information contained in them. Ignore all words that appear in fewer than X= 26 e-mails of the 5000 e-mail in training set – this is both a means of preventing over-fitting (to be discussed in class) and of improving scalability. To do this,

write a function `create_wordlist(data)` which takes the training data as input and returns a Python list containing all the words that occur in at least 26 emails.

Then we will implement a Naive Bayes classifier only considering the words in the vocabulary. Recall that we make classification based on the postior probability

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x|y=1)P(y=1) + P(x|y=0)P(y=0)}$$

And under the independence assumption, we have

$$P(x|y) = P(x_1, x_2, ..., x_d|y) = P(x_1|y)P(x_2|y)...P(x_d|y)$$

where d is the size of the vocabulary.

The maximum likelihood estimates for the parameters are

$$P(y=1) = \frac{\sum_{i=1}^{n} I(y_i = 1)}{n}$$

$$P(x_j = 1|y=1) = \frac{\sum_{i=1}^{n} I(y_i = 1 \ \& \ x_j = 1)}{\sum_{i=1}^{n} I(y_i = 1)}$$

$$P(x_j = 1|y=0) = \frac{\sum_{i=1}^{n} I(y_i = 0 \ \& \ x_j = 1)}{\sum_{i=1}^{n} I(y_i = 0)}$$

In the situation where the testing data contain new words which the training data has never seen before, we may get 0 posterior probability. **To avoid that, we use the following equations as the estimation of the parameters:**

$$P(x_j = 1|y=1) = \frac{\sum_{i=1}^{n} I(y_i = 1 \ \& \ x_j = 1) + 1}{\sum_{i=1}^{n} I(y_i = 1) + 2}$$

$$P(x_j = 1|y=0) = \frac{\sum_{i=1}^{n} I(y_i = 0 \ \& \ x_j = 1) + 1}{\sum_{i=1}^{n} I(y_i = 0) + 2}$$

2. **[15 Points]** To improve the accuracy, feel free to change the dimensionality of features by using a different vocabulary threshold T. Tell me about what you try and what result you get.

   Solution: I have tried the threshold form 1-34. The optimal threshold based on the validation error is `threshold` $= 10$.