EECS545 - Homework 3

February 12th, 2018

Instructions

- This homework is Due Monday, February the 26th at 5pm. No late submissions will be accepted.
- You will submit a write-up and your code for this homework.
- You are expected to use Python for the programming questions in this homework. Use of Python 3 is recommended.
- Submit your plots and answers as a pdf on Canvas under filename `uniqname_hw3.pdf`.
- Submit all your python code in a compressed zip file named `uniqname_hw3_code.zip`. Your zip file should contain `prob1.py`, `prob3.py`, and `prob4.py`.
- We expect you to implement models and algorithms using numpy functions. You will not need any specific machine learning libraries for this homework.

1. (**25 points**) In this problem, you will implement a **Naive Bayes** spam filter. In the second part of the problem you will show that a binary Naive Bayes classifier is a linear classifier.

   We will use bag-of-words features.

   - The vocabulary size is $D = 1448$.
   - Feature x_d for $d \in \{1, 2, \cdots, D\}$ indicates the existence of the d-th word. That is, $x_d = 1$ iff the d-th word appears in the email, and $x_d = 0$ otherwise.

   We have processed the raw data into four files:

   - `spam_train_features.npy`, `spam_train_labels.npy` contain the training set.
   - `spam_test_features.npy`, `spam_test_labels.npy` contain the test set.

   There are 1400 instances in `spam_train_features.npy` and 800 instances in `spam_test_features.npy`; each instance is represented as a 1448-dimensional vector. The labels are stored as binary vectors consisting of values of 0 or 1 in `spam_train_labels.npy` and `spam_test_features.npy`, where 1 indicates spam emails. You will learn the class priors π and class-conditional probabilities θ and use these parameters to predict labels for the test set. Use the parameter estimates in section "*Naive Bayes: Mean Estimate*" in the lecture notes with the α's and β's set to 1.

   Make sure your submitted python script has the same name as the starter code file `prob1.py`.

(a) Implement a spam filter and report the error rate on test data. The error rate is defined as:

$$\text{Error rate} = \frac{\text{Number of mis-classified instances in test set}}{\text{Number of instances in test set}} \times 100\%$$

(b) Show that a Naive Bayes binary classifier is a linear classifier (classifier with a linear decision boundary). That is, show that the two-class Naive Bayes classifier can be represented as

$$y = \begin{cases} 1 \text{ if } w^T \phi(x) \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

for an appropriate choice of w, where $\phi(x) = [x_1, x_2, \cdots, x_D, 1]^T$.
Hint: Consider defining $w = w_c - w_{\bar{c}}$ where w_c is given by

$$w_c = \left(\log \frac{\theta_{c11}}{1-\theta_{c11}}, \log \frac{\theta_{c21}}{1-\theta_{c21}}, \log \frac{\theta_{cD1}}{1-\theta_{cD1}}, \log \pi_c + \sum_{d=1}^{D} \log(1-\theta_{cd1}) \right)^T$$

and $w_{\bar{c}}$ has a similar form and $\theta_{cdm} = P(x_d = m|c)$.

2. (**15 points**) In this problem, we will consider ways of constructing new kernels from existing kernels, and use them to show that the Gaussian kernel $\kappa(x, x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2})$ is a valid kernel (corresponding to an infinite dimensional feature function).

(a) Let

- κ_1, κ_2: positive-definite kernel functions defined on $\mathbb{R}^s \times \mathbb{R}^s$
- a: a positive real number
- $f : \mathbb{R}^s \to \mathbb{R}$ be a function,
- $p : \mathbb{R} \to \mathbb{R}$ a polynomial function with positive coefficients.

Show that the following are valid kernels.

i. $\kappa(x, x') = a\kappa_1(x, x')$
ii. $\kappa(x, x') = \kappa_1(x, x') + \kappa_2(x, x')$
iii. $\kappa(x, x') = \kappa_1(x, x')\kappa_2(x, x')$
iv. $\kappa(x, x') = f(x)f(x')$
v. $\kappa(x, x') = \kappa_1(x, x')^d, d \in \mathbb{N}$
vi. $\kappa(x, x') = p(\kappa(x, x'))$

(b) Prove that the Gaussian kernel $\kappa(x, x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2})$ can be expressed as $\phi(x)^T\phi(x')$ where $\phi(\cdot)$ is an infinite-dimensional vector.
Hint: Use power series and Taylor's theorem.

3. (**25 points**) In this problem, you will read about **Kernel Perceptron** and implement it. Please submit your code in a file named `prob3.py`.

(a) Recall that we use $y(x) = \text{sgn}(w^T x)$ as the predictor in Perceptron. In Kernel Perceptron, w is usually defined as:

$$w = \sum_i \alpha_i y_i \phi(x_i)$$

where α_i's are coefficients, x_i's and y_i's are training examples and labels respectively, $\phi(\cdot)$ is the implicit feature map corresponding to the chosen kernel.

Given several training examples $x_1, x_2, \cdots, x_n$ from input space $\mathcal{X}$, their labels $y_1, y_2, \cdots, y_n$ where $y_i \in \{-1, 1\}$ and a valid kernel function $\kappa(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, express the predictor $y(x)$ for Kernel Perceptron in terms of x_i, y_i, α_i $(i \in \{1, ..., n\})$. Describe a learning algorithm for this model in pseudocode.

(b) Now you are given several 2-dimensional points (starter code `prob3.py` generates the data) belonging to two different classes. Points are given in array 'data', and labels are given in array 'target' consisting values of -1 or 1. Based on the definition of w in (a), determine α_i's such that for each point x_i and corresponding target value y_i, we have

$$w^T \phi(x_i) > 0 \leftrightarrow y_i > 0$$

where ϕ is the feature space induced by the Gaussian kernel $\kappa(x, x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2})$.

For each of the following values of $\sigma \in \{0.1, 1.0\}$, construct a plot showing:

- A scatter plot of the given points, colored coded based on the corresponding target value.
- The decision boundary learned by your algorithm. (point $x \in \mathbb{R}^2$ on this curve should satisfy $w^T \phi(x) = 0$)

Hint: You may find the `levels` option for `matplotlib.pyplot.contour` useful.

Note: For (a), you are supposed to provide an expression for the predictor and pseudo code of your learning algorithm. You will provide two plots for (b) satisfying the given requirements.

4. (**25 points**) In this problem, you will calculate boundaries separating samples drawn from two distributions via **Linear Discriminant Analysis (LDA)** and **Quadratic Discriminant Analysis (QDA)**. In `prob4.py` you are provided with a dataset (drawn from the real-world iris dataset). Each element of the dataset is represented by two features and maps to one of two classes, C_0 or C_1. Elements in `data_0` map to C_0 and elements in `data_1` map to C_1. Assume a uniform prior between the two classes (that is, $P(C_1) = P(C_0) = .5$).

   (a) Find the mean vector and covariance matrix of the entire dataset (both `data_0` and `data_1`) and use these to plot a linear boundary between elements in class C_0 or class C_1. That is, find the contour along which $P(C_0|x) = P(C_1|x)$ for some vector in feature space x. Create a plot showing your boundary and the points in each dataset.

   (b) Repeat this process, calculating the mean vector and covariance matrix for datapoints in each cluster seperately, and use these to plot a quadratic boundary between clusters. Again, create a plot showing your boundary and the datapoints.

   Your solution should consist of 2 plots. Please submit the code you use to generate these plots under `prob4.py`.

   Hint: Once again, You may find conveinent the `levels` option for `matplotlib.pyplot.contour`.

Note: Assume that the data you have is the whole population and you are trying to simply find the best boundary between these two. Don't use, for example, the sample covariance rather than the population covariance.