CS 529: Assignment #2

Z. Berkay Celik
zcelik@purdue.edu
(Due Sunday 9/22/2019 11:59 PM)

Computer Science, Purdue University — September 9, 2019

Instructions

Info: This HW includes both theory and coding problems. Please read course policy before starting your HW.

- Your code must work with Python 3.5+ (you may install the Anaconda distribution of Python).
- You need to submit a report including solutions of theory problems (in pdf format), and a Jupyter notebook for programming problems that includes your source code.
- To answer questions 1 and 2, Introduction to Machine Learning, Ethem Alpaydin, Chapters 3.4 (Discriminant Functions) and 10.3 (Geometry of the Linear Discriminant) are suggested to read (online access available at Purdue Libraries).

1 Problem 1: Decision Theory [25pt]

Info: The goal of Problem 1 and Problem 2 is to make you understand the different ways of constructing the discriminant functions in classification problems. These problems attempt to learn a discriminant function through probabilistic reasoning.

Assume a discriminant function of the form:

$$g_i(x) = \ln p\left(x|w_i\right) + \ln P\left(w_i\right).$$

, which achieves the minimum error classification. Assume that $x \in R$ for class 1 $x \sim N\left(\mu_1, \sigma\right)$, and for class 2, $x \sim N\left(\mu_2, \sigma\right)$. We also assume that $p\left(w_1\right) = p\left(w_2\right)$.

1. Derive the discriminant functions $g_1(x)$ and $g_2(x)$.
2. What is the equation for the separating surface (Hint: $g_1(x)$ - $g_2(x) = 0$)?
3. For μ_1=5 and μ_2=15, σ=5 plot the probability density functions (pdf) of the two classes inputs and also the separating surface.
4. If $p(w_1)$ increases to 0.8, where would the separating surface be?

2 Problem 2: Parametric Methods [25pt]

Assume that you observed the following number of requests to a web server from two domains, Domain A and Domain B (Domain A: Class$_1$, Domain B: Class$_2$).

Number of Requests from Domain A = $\{2, 3, 4, 3, 4, 3, 3, 4, 5, 3, 2, 3, 4, 3, 2, 2, 3, 4, 5, 6\}$

Number of Requests from Domain B: $\{22, 23, 24, 23, 24, 23, 23, 24, 25, 23\}$

Assuming that number of requests is distributed according to normal distribution,

1. Using the data, estimate the mean, std. dev. and prior for each class.
2. What are the decision regions for Class_1 and Class_2 (Hint: $g_A(x)$ - $g_B(x) = 0$)?

3 Problem 3: k-NN [25pt]

Info: You will use the Pima Indians Diabetes data set from the UCI repository to experiment with the k-NN algorithm. Your goal is to find the optimal value for the number of neighbors k.

Running and Deliverable. You are provided with a **Problem3.ipynb** file to put the implementation code for all questions below. Make sure your notebook is runnable on Anaconda with Python 3.5+. The results and discussions should be included in the PDF file.
You do not need to implement the k-NN algorithm and encouraged to use the implementation in scikit-learn. The problem 3 in Homework-1 lead to some questions about cross validation and normalization. Before you start this question, please read the cross validation details here and check the related discussion for normalization at Piazza. Below is a simple code showing the steps to use the NN implementation when the number of neighbors is 3:

```
from sklearn.neighbors import KNeighborsClassifier
# Create NN classifier
knn = KNeighborsClassifier(n_neighbors = 3)
# Fit the classifier to the data
knn.fit(X_train,y_train)
# Predict the labels of test data
yhat = knn.predict(X_test)
# Or, directly check accuracy of our model on the test data
knn.score(X_test, y_test)
```

To accomplish this task, please do:

1. Download the provided **Pima.csv** data file and load it using pandas. As a sanity check, make sure there are 768 rows of data (potential diabetes patients) and 9 columns (8 input features including Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, and 1 target output). Note that the data file has no header and you might want to explicitly create the header. The last value in each row contains the target label for that row, and the remaining values are the features. Report the statistics of each feature (min, max, average, standard deviation) and the histogram of the labels (target outputs).
2. Split the data into training and test sets with 80% training and 20% test data sizes. You can easily do this in scikit-learn, e.g.,

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

   Use 5-fold cross-validation on training data to decide the best number of neighbours k. To this end, you can use the built in functionality in scikit-learn such as `cross_val_score` (note that this function returns the accuracy for each fold in an array and you have to average them to get the accuracy for all splits). For $k = 1, 2, 3, \ldots, 15$ compute the 5-fold cross validation error and plot the results (with values of k on the x-axis and accuracy on the y-axis). Include the plot in your report and justify your decision for picking a particular number of neighbors k.
3. Evaluate the k-NN algorithm on test data with the optimal number of neighbours you obtained in previous step and report the test error (you will use the same value of k you have found in 2.).

4. Process the input data by subtracting the mean (a.k.a. centralization) and dividing by the standard deviation (a.k.a. standardization) over each dimension (feature), repeat the previous part and report the accuracy. Do centralization and standardization impact the accuracy? Why?

4 Problem 4: Decision Trees [25pt]

Info: You will investigate building a decision tree for a binary classification problem. The training data is given in Figure 1 with 16 instances that will be used to learn a decision tree for predicting whether a mushroom is edible or not based on its attributes `(Color, Size and Shape)`. Please note the label set is a binary set `{Yes, No}`. For this problem, you need to submit a report in PDF for your solutions.

Instance	Color	Size	Shape	Edible?
D1	Yellow	Small	Round	Yes
D2	Yellow	Small	Round	No
D3	Green	Small	Irregular	Yes
D4	Green	Large	Irregular	No
D5	Yellow	Large	Round	Yes
D6	Yellow	Small	Round	Yes
D7	Yellow	Small	Round	Yes
D8	Yellow	Small	Round	Yes
D9	Green	Small	Round	No
D10	Yellow	Large	Round	No
D11	Yellow	Large	Round	Yes
D12	Yellow	Large	Round	No
D13	Yellow	Large	Round	No
D14	Yellow	Large	Round	No
D15	Yellow	Small	Irregular	Yes
D16	Yellow	Large	Irregular	Yes

Table 1: Mushroom data with 16 instances, three categorical features, and binary labels.

1. Which attribute would the algorithm choose to use for the root of the tree. Show the details of your calculations. Recall from lectures that if we let S denote the data set at current node, A denote the feature with values $v \in V$, H denote the entropy function, and S_v denote the subset of S for which the feature A has the value v, the gain of a split along the feature A, denoted $\texttt{InfoGain}(\texttt{S}; \texttt{A})$ is computed as:

$$\text{InfoGain}(\mathcal{S}, A) = H(\mathcal{S}) - \sum_{v \in V} \left(\frac{|\mathcal{S}_v|}{|\mathcal{S}|} \right) H\left(\mathcal{S}_v\right)$$

   That is, we are taking the difference of the entropy before the split, and subtracting of the entropy of each new node after splitting, with an appropriate weight depending on the size of each node.

2. Draw the full decision tree that would be learned for this data (assume no pruning and you stop splitting a leaf node when all samples in the node belong to the same class, i.e., there is no information gain in splitting the node).

3. Table 1 includes only categorical features. However, some datasets such as the ones in security often include real valued (numerical) features. Handling real valued (numerical) features is totally different from categorical features in splitting nodes. In this problem, we look for a simple solution to decide good thresholds for splitting based on numerical features. Specifically, when there is a numerical feature in data, an option would be treating all numeric values of feature as discrete, i.e., proceeding exactly as we do with categorical data. What problems may arise when we use a tree derived this way to classify an unseen example? Please support your claims.

5 Problem 5 [Extra Credit, 50pt]

Info: You are provided with a sample data sets (data.npy). You can load these files into numpy array using this syntax: `np.load("data.npy")`. The sample data for this homework has 100 data points $(x_i; y_i)$ in a 100x2 numpy array. Please note that you need to add a column of all ones to data to handle the intercept term, making your data a 100x3 numpy array. For the plotting part, make sure that your plots have appropriate title, x and y-axis labels. You need to submit two python files `Problem5.py` and `Problem5Plot.py` and a pdf file including all the plots. In Problem5.py everything except the imports should be inside the functions definition we mention below. The file Problem5Plot.py is where you are generating the plots.

In this problem, we consider a simple linear regression model with a modied loss function and try to solve it with Gradient Descant (GD) and Stochastic Gradient Descant (SGD).

In general setting, the data has the form $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i$ is the d-dimensional feature vector and y_i is a real-valued target. For this regression problem, we will be using linear prediction $\mathbf{w}^T\mathbf{x}_i$ with the objective function:

$$f(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} g_\delta(\boldsymbol{w}; \boldsymbol{x}_i, y_i) + \lambda \sum_{j=1}^{d} w_j^2$$

where $g_\delta(\mathbf{w}; \mathbf{x}, y)$ is the error of linear model with parameter vector $\mathbf{w} \in R^d$ on a single training pair (x, y) defined by:

$$g_\delta(\boldsymbol{w}; \boldsymbol{x}, y) = \begin{cases} (y - \boldsymbol{w}\cdot\boldsymbol{x} - \delta)^2 & \text{if } y \geq \boldsymbol{w}\cdot\boldsymbol{x} + \delta \\ 0 & \text{if } |y - \boldsymbol{w}\cdot\boldsymbol{x}| < \delta \\ (y - \boldsymbol{w}\cdot\boldsymbol{x} + \delta)^2 & \text{if } y \leq \boldsymbol{w}\cdot\boldsymbol{x} - \delta \end{cases}$$

Please note that we simply dropped the intercept term by the simple trick we discussed in the lecture, i,e., adding a constant feature, which is always equal to one to simplify estimation of the "intercept" term.

Gradient Descent: In this part, you are asked to optimize the above objective function using gradient descent and plot the function values over different iterations, which can be done using the python library matplotlib.

To this end, in Problem5.py, fill in the function **bgd_l2(data, y, w, eta, delta, lam, num_iter)** where `data` is a two dimensional numpy array with each row being a feature vector, `y` is a one-dimensional numpy array of target values, `w` is a one-dimensional numpy array corresponding to the weight vector, `eta` is the learning rate, `delta` and `lam` are parameters of the objective function. This function should return new weight vector, history of the value of objective function after each iteration (python list). Run this function for the following settings and plot the history of objective function (you should expect a monotonically decreasing function over iteration number):

- $\eta = 0.05, \delta = 0.1, \lambda = 0.001, \texttt{num\_iter} = 50$
- $\eta = 0.1, \delta = 0.01, \lambda = 0.001, \texttt{num\_iter} = 50$
- $\eta = 0.1, \delta = 0, \lambda = 0.001, \texttt{num\_iter} = 100$
- $\eta = 0.1, \delta = 0, \lambda = 0, \texttt{num\_iter} = 100$

Stochastic Gradient Descent. Fill in the function **sgd_l2(data, y, w, eta, delta, lam, num_iter, i)** where data, `y`, `w`, `lam`, `delta` and `num_iter` are same as previous part. In this part, you should use $\frac{\eta}{\sqrt{t}}$ as a learning rate, where `t` is the iteration number, starting from 1. The variable `i` is for testing the correctness of your function. If `i` set to -1 then you just need to apply the normal SGD (randomly select the data point), which runs for `num_iter`, but if `i` set to something else (other than 1), your code only needs to compute SGD for that specific data point (in this case, the `num_iter` will be 1!).

Run this function for the settings below and plot the history of objective function:

- $\eta = 1, \delta = 0.1, \lambda = 0.5, \texttt{num\_iter} = 800$
- $\eta = 1, \delta = 0.01, \lambda = 0.1, \texttt{num\_iter} = 800$
- $\eta = 1, \delta = 0, \lambda = 0, \texttt{num\_iter} = 40$
- $\eta = 1, \delta = 0, \lambda = 0, \texttt{num\_iter} = 800$