CS 613 - Machine Learning

Assignment 2 - Classification

Introduction

In this assignment you will perform classification using Logistic Regression, Naive Bayes and Decision Tree classifiers. You will run your implementations on a binary class dataset and report your results.

You may **not** use any functions from a ML library in your code unless explicitly told otherwise.

Grading

Part 1 (Theory)	15pts
Part 2 (Logistic Regression)	15pts
Part 3 (Spam Logistic Regression)	10pts
Part 4 (Naive Bayes)	25pts
Part 5 (Decision Trees)	25pts
Report	10pts
TOTAL	100

Datasets

Iris Dataset (sklearn.datasets.load_iris) The Iris flower data set or Fishers Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.
The iris data set is widely used as a beginner's dataset for machine learning purposes. The dataset is included in the machine learning package Scikit-learn, so that users can access it without having to find a source for it. The following python code illustrates usage.

```
from sklearn.datasets import load_iris
iris = load_iris()
```

Spambase Dataset (spambase.data) This dataset consists of 4601 instances of data, each with 57 features and a class label designating if the sample is spam or not. The features are *real valued* and are described in much detail here:

https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names

Data obtained from: https://archive.ics.uci.edu/ml/datasets/Spambase

1 Theory

1. Consider the following set of training examples for an unknown target function: $(x_1, x_2) \rightarrow y$:

Y	x_1	x_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

   (a) What is the sample entropy, $H(Y)$ from this training data (using log base 2) (2pts)?

   (b) What are the information gains for branching on variables x_1 and x_2 (2pts)?

   (c) Draw the deicion tree that would be learned by the ID3 algorithm without pruning from this training data (3pts)?

2. We decided that maybe we can use the number of characters and the average word length an essay to determine if the student should get an A in a class or not. Below are five samples of this data:

# of Chars	Average Word Length	Give an A
216	5.68	Yes
69	4.78	Yes
302	2.31	No
60	3.16	Yes
393	4.2	No

   (a) What are the class priors, $P(A = Yes), P(A = No)$? (2pt)

   (b) Find the parameters of the Gaussians necessary to do Gaussian Naive Bayes classification on this decision to give an A or not. Standardize the features first over all the data together so that there is no unfair bias towards the features of different scales (2pts).

   (c) Using your response from the prior question, determine if an essay with 242 characters and an average word length of 4.56 should get an A or not (3pts).

3. Consider the following questions pertaining to a k-Nearest Neighbors algorithm (1pt):

   (a) How could you use a *validation set* to determine the user-defined parameter k?

2 Logistic Regression

Let's train and test a *Logistic Regression Classifier* to classify flowers from the Iris Dataset.

First download import the data from sklearn.datasets. As mentioned in the Datasets area, The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. We will map this into a binary classification problem between Iris setosa versus Iris virgincia and versicolor. We will use just the first 2 features, width and length of the sepals.

For this part, we will be practicing gradient descent with logistic regression.

Use the following code to load the data, and binarize the target values.

```
iris = skdata.load_iris()
X = iris.data[:, :2]
y = (iris.target != 0) * 1
```

Write a script that:

1. Reads in the data with the script above.
2. Standardizes the data using the mean and standard deviation
3. Initialize the parameters of θ using random values in the range [-1, 1]
4. Do **batch** gradient descent
5. Terminate when absolute value change in the loss on the data is less than 2^{-23}, or after $10,000$ iterations have passed (whichever occurs first).
6. Use a learning rate $\eta = 0.01$.
7. While the termination criteria (mentioned above in the implementation details) hasn't been met
   (a) Compute the loss of the data using the logistic regression cost
   (b) Update each parameter using *batch* gradient descent

Plot the data and the decision boundary using matplotlib. Verify your solution with the LogisticRegression sklearn method.

```
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression(penalty='none',solver='lbfgs',max_iter=10000)
lgr.fit(X,y)
```

In your writeup, present the thetas from gradient descent that minimize the loss function as well as plots of your method versus the built in LogisticRegression method.

3 Logistic Regression Spam Classification

Let's train and test a *Logistic Regression Classifier* to classifiy Spam or Not from the Spambase Dataset.

First download the dataset *spambase.data* from Blackboard. As mentioned in the Datasets area, this dataset contains 4601 rows of data, each with 57 continuous valued features followed by a binary class label (0=not-spam, 1=spam). There is no header information in this file and the data is comma separated.

Write a script that:

1. Reads in the data.
2. Randomizes the data.
3. Selects the first 2/3 (round up) of the data for training and the remaining for testing (you may use **sklearn train_test_split** for this part)
4. Standardizes the data (except for the last column of course) using the training data
5. Initialize the parameters of θ using random values in the range [-1, 1]
6. Do **batch** gradient descent
7. Terminate when absolute value change in the loss on the data is less than 2^{-23}, or after $1,500$ iterations have passed (whichever occurs first, this will likely be a slow process).
8. Use a learning rate $\eta = 0.01$.
9. Classify each testing sample using the model and choosing the class label based on which class probability is higher.
10. Computes the following statistics using the testing data results:
    (a) Precision
    (b) Recall
    (c) F-measure
    (d) Accuracy

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. There are a lot of θs and this will likely be a slow process

In your report you will need:

1. The statistics requested for your Logistic classifier run.

4 Naive Bayes Classifier

Let's train and test a *Naive Bayes Classifier* to classifiy Spam or Not from the Spambase Dataset.

First download the dataset *spambase.data* from Blackboard. As mentioned in the Datasets area, this dataset contains 4601 rows of data, each with 57 continuous valued features followed by a binary class label (0=not-spam, 1=spam). There is no header information in this file and the data is comma separated. As always, your code should work on any dataset that lacks header information and has several comma-separated continuous-valued features followed by a class id $\in 0, 1$.

Write a script that:

1. Reads in the data.
2. Randomizes the data.
3. Selects the first 2/3 (round up) of the data for training and the remaining for testing
4. Standardizes the data (except for the last column of course) using the training data
5. Divides the training data into two groups: Spam samples, Non-Spam samples.
6. Creates Normal models for each feature for each class.
7. Classify each testing sample using these models and choosing the class label based on which class probability is higher.
8. Computes the following statistics using the testing data results:
   (a) Precision
   (b) Recall
   (c) F-measure
   (d) Accuracy

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. You may want to consider using the log-exponent trick to avoid underflow issues. Here's a link about it: https://stats.stackexchange.com/questions/105602/example-of-how-the-log-sum-exp-trick-works-in-naive-bayes
3. If you decide to work in log space, realize that python interprets $0log0$ as inf. You should identify this situation and either add an EPS (very small positive number) or consider it to be a value of zero.

In your report you will need:

1. The statistics requested for your Naive Bayes classifier run.

5 Decision Trees

Let's train and test a *Decision Tree* to classify Spam or Not from the Spambase Dataset.

Write a script that:

1. Reads in the data.
2. Randomizes the data.
3. Selects the first 2/3 (round up) of the data for training and the remaining for testing
4. Standardizes the data (except for the last column of course) using the training data
5. Divides the training data into two groups: Spam samples, Non-Spam samples.
6. Trains a decision tree using the ID3 algorithm without any pruning.
7. Classify each testing sample using your trained decision tree.
8. Computes the following statistics using the testing data results:
   (a) Precision
   (b) Recall
   (c) F-measure
   (d) Accuracy

Implementation Details

1. Seed the random number generate with zero prior to randomizing the data
2. Depending on your perspective, the features are either continuous or finite discretize. The latter can be considered tru since the real-values are just the number of times a feature is observed in an email, normalized by some other count. That being said, for a decision tree we normally use categorical or discretized features. **So for the purpose of this dataset, look at the range of each feature and turn them into binary features by choosing a threshold. I suggest using the median or mean.**

In your report you will need:

1. The statistics requested for your Decision Tree classifier run.

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Python notebook Code

The PDF document should contain the following:

1. Part 1:
    (a) Answers to Theory Questions
2. Part 2:
    (a) Requested Logistic Regression thetas and plots
3. Part 3:
    (a) Requested Classification Statistics
4. Part 4:
    (a) Requested Classification Statistics
5. Part 5:
    (a) Requested Classification Statistics