CS 613 - Machine Learning

Assignment 2 - Classification

John Obuch

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)?

   We know that the equation for computing entropy is as follows:

$$H(P(v_1), \dots, P(v_n)) = \sum_{i=1}^{n} (-P(v_i) log_2 P(v_i))$$

   Given the equation above, we can compute the probabilities of each class of Y. Let p represent the number of positive classifications and n represent the number of negative classifications, and N be the total number of records in the data. Computing these values we obtain the following results:

$$p = 3 + 4 + 4 + 1 = 12$$
$$n = 0 + 1 + 3 + 5 = 9$$
$$N = 3 + 4 + 4 + 1 + 0 + 1 + 3 + 5 = 21$$

   Leveraging the equation defined above we have:

$$P(v_1) = P(p) = \frac{p}{N} = \frac{12}{21} = \frac{4}{7}$$
$$P(v_2) = P(n) = \frac{n}{N} = \frac{9}{21} = \frac{3}{7}$$

$$H(Y) = H(\frac{4}{7}, \frac{3}{7}) = (-\frac{4}{7} log_2(\frac{4}{7})) + (-\frac{3}{7} log_2(\frac{3}{7})) = 0.461 + 0.524 = 0.985$$

■

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

   We will begin by defining the average entropy with respect to a data set A as follows:

$$E(H(A)) = \sum_{i=1}^{k} \frac{p_i + n_i}{p + n} H(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i})$$

Next we will compute each of the p_i and n_i for $i = T, F$ in each of the classes for each of the features. Let p_T and n_T represent the number of True outcomes in the positive and negative classes respectively and let p_F and n_F signify the number of False outcomes in the positive and negative classes. Computing these values we obtain the following:

Note: Recall that the $p = 12$ and $n = 9$ of the entire system as computed above in part (a).

Feature x_1:

$$p_T = 3 + 4 = 7$$
$$n_T = 0 + 1 = 1$$
$$p_F = 4 + 1 = 5$$
$$n_F = 3 + 5 = 8$$

Feature x_2:

$$p_T = 3 + 4 = 7$$
$$n_T = 0 + 3 = 3$$
$$p_F = 4 + 1 = 5$$
$$n_F = 1 + 5 = 6$$

Implementing the equation defined above, we yield the following:

$E(H(x_1)) = (\frac{7+1}{12+9})(-\frac{7}{7+1}log_2(\frac{7}{7+1}) - \frac{1}{7+1}log_2(\frac{1}{7+1})) + (\frac{5+8}{12+9})(-\frac{5}{5+8}log_2(\frac{5}{5+8}) - \frac{8}{5+8}log_2(\frac{8}{5+8}))$

$= (\frac{8}{21})(-\frac{7}{8}log_2(\frac{7}{8}) - \frac{1}{8}log_2(\frac{1}{8})) + (\frac{13}{21})(-\frac{5}{13}log_2(\frac{5}{13}) - \frac{8}{13}log_2(\frac{8}{13}))$

$= 0.2071 + 0.5951$

$= 0.8021$

$E(H(x_2)) = (\frac{7+3}{12+9})(-\frac{7}{7+3}log_2(\frac{7}{7+3}) - \frac{3}{7+3}log_2(\frac{3}{7+3})) + (\frac{5+6}{12+9})(-\frac{5}{5+6}log_2(\frac{5}{5+6}) - \frac{6}{5+6}log_2(\frac{6}{5+6}))$

$= (\frac{10}{21})(-\frac{7}{10}log_2(\frac{7}{10}) - \frac{3}{10}log_2(\frac{3}{10})) + (\frac{11}{21})(-\frac{5}{11}log_2(\frac{5}{11}) - \frac{6}{11}log_2(\frac{6}{11}))$

$= 0.4197 + 0.5207$

$= 0.9403$

Finally, we need to compute the information gain for each of the features. The equation for computing the information gain is as follows:

$$\text{IG}(A) = H(\frac{p}{p+n}, \frac{n}{p+n}) - E(H(A))$$

Utilizing the equation above, we obtain the following results:

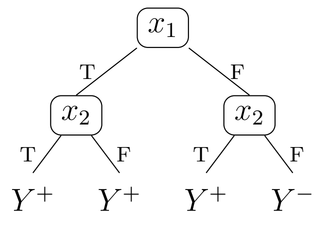
$\text{IG}(x_1) = H(Y) - E(H(x_1)) = 0.985 - 0.802 = 0.183$

$\text{IG}(x_2) = H(Y) - E(H(x_2)) = 0.985 - 0.940 = 0.045$

Analyzing the results above, since feature x_1 has a higher information gain than that of feature x_2, we will choose to select feature x_1 as the splitting attribute.

■

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



Note: Leaf nodes were yielded by taking the *mode* of the counts from the outcome variable.

■

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)

$$P(A = \text{Yes}) = \frac{3}{5} = 0.60$$

$$P(A = \text{No}) = \frac{2}{5} = 0.40$$

■

(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).

We begin by computing the mean and standard deviation of the both of the X_i features for $i = 1, 2$. Let μ_C and σ_C be the mean and standard deviation of the *# of Chars* feature respectively. Similarly, let μ_L and σ_L represent the mean and standard deviation of the *Average Word Length* feature. Computing these values we have that:

$$\mu_C = 208$$

$$\sigma_C = 145.2153$$

$$\mu_L = 4.026$$

$$\sigma_L = 1.3256$$

After applying the standardization transformation $X_{\text{stdz}} = \frac{X_i - \mu_i}{\sigma_i}$, the results of the standardized data are as follows:

Standardized Data using N-1 degrees of freedom:

# of Chars	Average Word Length	Give an A
0.055	1.248	Yes
-0.957	0.569	Yes
0.647	-1.295	No
-1.019	-0.653	Yes
1.275	0.131	No

Next, we compute the mean and standard deviation of the standardized data for each class (i.e. the parameters of the Gaussians).

Student Gets an A - Yes:

# of Characters

$$\mu_{C,\text{Yes}} = \frac{(0.055 - 0.957 - 1.019)}{3} = -0.6403$$

$$\sigma^2_{C,\text{Yes}} = \frac{(0.055 - (-0.6403))^2 + (-0.957 - (-0.6403))^2 + (-1.019 - (-0.6403))^2}{2} = 0.3636$$

$$\sigma_{C,\text{Yes}} = \sqrt{\sigma^2_{C,\text{Yes}}} = \sqrt{0.3636} = 0.603$$

Average Word Length

$$\mu_{L,\text{Yes}} = \frac{(1.248 + 0.569 - 0.653)}{3} = 0.388$$

$$\sigma^2_{L,\text{Yes}} = \frac{(1.248 - 0.388)^2 + (0.569 - 0.388)^2 + (-0.653 - 0.388)^2}{2} = 0.928$$

$$\sigma_{L,\text{Yes}} = \sqrt{\sigma^2_{L,\text{Yes}}} = \sqrt{0.928} = 0.963$$

Student Gets an A - No:

# of Characters

$$\mu_{C,\text{No}} = \frac{(0.647 + 1.275)}{2} = 0.961$$

$$\sigma^2_{C,\text{No}} = \frac{(0.647 - 0.961)^2 + (1.275 - 0.961)^2}{1} = 0.197$$

$$\sigma_{C,\text{No}} = \sqrt{\sigma^2_{C,\text{No}}} = \sqrt{0.197} = 0.444$$

Average Word Length

$$\mu_{L,\text{No}} = \frac{(-1.295 + 0.131)}{2} = -0.582$$

$$\sigma^2_{L,\text{No}} = \frac{(-1.295 - (-0.582))^2 + (0.131 - (-0.582))^2}{1} = 1.017$$

$$\sigma_{L,\text{No}} = \sqrt{\sigma^2_{L,\text{No}}} = \sqrt{1.017} = 1.008$$

■

(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).

Given $X_C = 242$ and $X_L = 4.56$, we will begin by standardizing these data elements with respect to all of the data which we will notate as $\widetilde{X_C}$ and $\widetilde{X_L}$ representing the standardized values of the essay with 242 characters and an average word length of 4.56 respectively. Doing this yields us the following:

$$\widetilde{X_C} = \frac{(242 - 208)}{145.2153} = 0.234$$

$$\widetilde{X_L} = \frac{(4.56 - 4.026)}{1.3256} = 0.403$$

Next we will leverage the Gaussian Probability Density Function to determine the probability of a student receiving an A given the features. The equation is defined as follows:

$$P(f_k = x_k \mid y = i) \propto \frac{1}{\sigma_i\sqrt{2\pi}} e^{\frac{-(x_i - \mu_i)^2}{2\sigma_i^2}}$$

Where μ_i is the mean value of feature f_k for the data from class $y = i$ and σ_i is the standard deviation of feature f_k for the data from class $y = i$.

Given the equation outlined above, we can compute the following:

$$
\begin{aligned}
P(\widetilde{X_C} \mid \text{Yes}) &= \frac{1}{0.603\sqrt{2\pi}} e^{\frac{-(0.234-(-0.6403))^2}{2(0.3636)}} = 0.2312 \\
P(\widetilde{X_L} \mid \text{Yes}) &= \frac{1}{0.963\sqrt{2\pi}} e^{\frac{-(0.403-0.388)^2}{2(0.928)}} = 0.4142 \\
P(\widetilde{X_C} \mid \text{No}) &= \frac{1}{0.444\sqrt{2\pi}} e^{\frac{-(0.234-0.961)^2}{2(0.197)}} = 0.2349 \\
P(\widetilde{X_L} \mid \text{No}) &= \frac{1}{1.008\sqrt{2\pi}} e^{\frac{-(0.403-(-0.582))^2}{2(1.017)}} = 0.2456
\end{aligned}
$$

Given these values, we can proceed by computing the following:

$$
\begin{aligned}
P(C = 242, L = 4.56 \mid \text{Yes}) &= P(\widetilde{X_C} \mid \text{Yes})P(\widetilde{X_L} \mid \text{Yes}) = (0.2312)(0.4142) = 0.0958 \\
P(C = 242, L = 4.56 \mid \text{No}) &= P(\widetilde{X_C} \mid \text{No})P(\widetilde{X_L} \mid \text{No}) = (0.2349)(0.2456) = 0.0577
\end{aligned}
$$

Finally, we apply Bayes Theorem:

$P(\text{Yes} \mid C = 242, L = 4.56) = \frac{P(C=242, L=4.56|\text{Yes})P(A=\text{Yes})}{P(C=242, L=4.56|\text{Yes})P(A=Yes)+P(C=242, L=4.56|\text{No})P(A=No)}$
$= \frac{(0.0958)(0.60)}{(0.0958)(0.60)+(0.0577)(0.40)}$
$= \frac{0.0575}{0.0805}$
$= 0.7134$
$P(\text{No} \mid C = 242, L = 4.56) = 1 - P(\text{Yes} \mid C = 242, L = 4.56) = 0.2866$

Analyzing the results of the Bayesian classifier, we observe the following:

$$P(\text{Yes} \mid C = 242, L = 4.56) = 0.7134 > 0.50 > P(\text{No} \mid C = 242, L = 4.56) = 0.2866$$

Thus, given an essay with a character count of 242 and an average word length of 4.56, the Bayesian classification of the essay receiving a letter grade of A is "Yes." Therefore, based on the features of the essay, the student will receive an A.

■

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?

   Response:
   We can use the validation set and also leverage S-folds cross-validation and implement the KNN algorithm on each fold and observe the type-errors associated to each hold-out fold (i.e. the misclassification rate). By analyzing the average error-rate for each S-fold hold-out group for various values of k and choosing the k that demonstrates the lowest mean error-rate would provide a good indication of 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.

My Results:

$$\hat{Y} = \theta_0 + \theta_1 \mathbf{X_1} + \theta_2 \mathbf{X_2} = 2.1543 + 3.7971\mathbf{X_1} - 2.5777\mathbf{X_2}$$

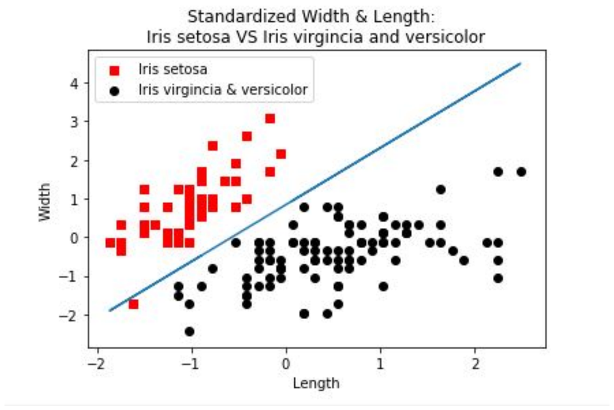


Figure 1: Plot of Length VS Width.

Sklearn Results:

$$\hat{Y} = \theta_0 + \theta_1 \mathbf{X_1} + \theta_2 \mathbf{X_2} = 49.2632 + 69.7372\mathbf{X_1} - 30.9021\mathbf{X_2}$$

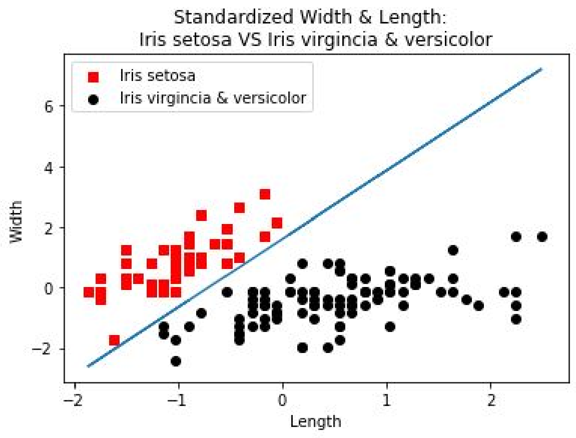


Figure 2: Plot of Length VS Width.

See source code for further detail

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.

Results:
Accuracy: 0.869651086240948
Precision: 0.8703374777975134
Recall: 0.7967479674796748
F1-Score: 0.8319185059422751

See source code for further detail

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 $0 log 0$ 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.

Results:
Accuracy: 0.771689497716895
Precision: 0.6252771618625277
Recall: 0.9791666666666666
F1 Score: 0.7631935047361298

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 true 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.

Results:
Accuracy: 0.863013698630137
Precision: 0.7675438596491229
Recall: 0.9114583333333334
F1 Score: 0.8333333333333334

Note: My results from the ID3 algorithm can likely be optimized to obtain better model evaluation KPIs. I believe this can be obtained by altering the case statements within the algorithm.

If more time allotted, I would explore a more efficient way of obtaining the mode of the data vs explicitly assigning a value of 1 as a work around (*see source code for further detail*).

Sklearn Result:
Accuracy: 0.9119373776908023

Data Science From Scratch Results:
Accuracy: 0.9439834024896265
Precision: 0.945872801082544
Recall: 0.9621472814865795
F1-Score: 0.9539406345957011

See source code for further detail

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