MACHINE LEARNING

Assignment-2

Generative Learning

SPRING – 2016

submitted to

Gady Agam

by

Darpan Patel
A20345898

PROBLEM STATEMENT

Here, In this assignment I will implement algorithms for generative learning. I will use two different types of datasets of UCI Machine Learning Repository one with continuous and one with discrete features. I

I implemented mainly two algorithms of classification with class labels of 2 – class and k – class. This both algorithms are also implementing for single dimension (1D) and multiple dimension (nD). Another algorithm is Naive Bayes with two different approaches are Bernoulli features and Binomial features.

1.) Gaussian Discriminant Analysis (1D , nD (2-class) and nD k-class)

- I selected one datasets named databanknote-authentication.txt as 2-class dataset with continuous nD features but I just extract one features to make it as 1D feature.

- Estimate model parameter like mu(**μ**) and sigma(**σ**) for 1D 2-class Features and apply discriminant function for 2-class [0,1].

- For nD 2-class, I used same datasets which has multiple features and then estimate model parameter mean(mu(**μ**)) and sigma matrix (co-varience (Σ)) for each 2-class and apply discriminant function for 2-class[0,1].

- For nD k-class, I used "iris" datasets which has multiple features and then estimate model parameter mean(mu(**μ**)) and sigma matrix (co-varience (Σ)) for each k-class and apply discriminant function for k-class[0,1,2,3…,k]

- Classify the testing examples created by K-Fold cross Validation and measure the error along with confusion matrix, precision, recall, F-measure and accuracy.

2.) Naive Bayes with Bernoulli and Binomial Features

- I used "Spam Base" dataset which 2-class dataset for Naive Bayes Bernoulli. I converted it into binary nD features from given frequency of each word and character in each document.

- I used same dataset for Naive Bayes Binomial. I assume the document length

as 100 for every document and count each word frequency based on given frequency of each word.

- Estimate the model parameter as prior probability and log likelihood for each class [0,1] and apply discriminant function assuming Naive Bayes assumption.

- Classify the testing examples created by K-Fold cross Validation and measure the error along with confusion matrix, precision, recall, F-measure and accuracy.

Analyzed different experiment's results using different training and testing datasets and compare the results.

PROPOSED SOLUTION

- I have implemented three separate program for Gaussian Discriminant Analysis (1D, nD 2-class and nD k-class) named "nD_GDA-2c.py" , Naive Bayes Bernoulli(naive_bayes_bernoulli.py) and Naive Bayes Binomial(naive_bayes_binomial.py).
- I have implemented all the function from scratch using some basic python libraries like numpy - for array and matrix functionalities, matplotlib – for plotting 2D graph and sklearn – for just check for correctness of outputs.
- I used mainly 3 different datasets to implement all the classification algorithms mentioned above. GDA has their own model parameter and NB has their own model parameter so just calculate it and classification based on membership function to each class.

Algorithm of Gaussian Discriminant Analysis

- First of all load the data into numpy array using "loadtxt" in-built function and split it into two different array of Training and Testing. Find the model parameter for 1D 2-class GDA (**μ** and **σ**) based on training dataset so that model is trained. Apply t membership function to each and every testing value

and find membership for each 2-class. Use it determine the membership of each value using discriminant function and Predict the value of testing data using it. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.).

- Apply the same algorithm for nD 2-class and find model parameter (**μ** and **Σ (co-varience matrix)**). Sigma is matrix of (n x n) where n = no. of features. Apply the membership function to each testing value. For k-class the discriminant function is only different like we need to find maximum from all membership function value and then Predicted the class label ass per maximum membership function value. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.).
- At the end find the mean-square-error, Confusion Matrix and find Accuracy, Precision, Recall and F-Measure for all the fold and then take maximum accuracy fold and retrieve it for better results and efficiency.

Algorithm of Naive Bayes Bernoulli

- First of all load the data into numpy array using "loadtxt" in-built function and split it into two different array of Training and Testing. Make trainning data as binary features only zeros and once as features value. So, consider every non-zero values as one. And then,Find the model parameter (prior (**α**) and likelihood(posterior) (**α(y=l)**)) based on training dataset so that model is trained. Apply t membership function to each and every testing value and find membership for each 2-class. Use it determine the membership of each value using discriminant function and Predict the value of testing data using it. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.).
- Find the mean-square-error, Confusion Matrix and find Accuracy, Precision, Recall and F-Measure for all the fold and then take maximum accuracy fold and retrieve it for better results and efficiency.

- Check the correctness of parameter with in-built function's parameter using sklearn.matrics libraries.

Algorithm of Naive Bayes Binomial

- First of all load the data into numpy array using “loadtxt” in-built function and split it into two different array of Training and Testing. Make training data discrete for Naive Bayes Binomial. So, We need document length for each of the document and according to frequency convert it into discrete features. So Here, I consider document length as **100.** And then,Find the model parameter (prior (**α**) and likelihood(posterior) (**α(y=l)**)) based on training dataset so that model is trained. Apply t membership function to each and every testing value and find membership for each 2-class. Use it determine the membership of each value using discriminant function and Predict the value of testing data using it. Find the training error and testing error from predicted values. Use K-fold cross validation for all training datasets. default value of K_fold is 10(User can change it.).
- Find the mean-square-error, Confusion Matrix and find Accuracy, Precision, Recall and F-Measure for all the fold and then take maximum accuracy fold and retrieve it for better results and efficiency.
- Check the correctness of parameter with in-built function's parameter using sklearn.matrics libraries.

Implementations Details

Design Issue :

- Math.log function is not working correctly.
- Data Fetching Time from INTERNET is little bit high for bigger file
- Difficult to combinatorial of higher values
- Design issued with discriminant function of binomial Naive Bayes.

How Problem Solved :

- faced a problem this kind of function from scratch in editor so that I used PyCharm IDE for python coding.
- Use in-built libraries of combinatorial gmpy.comb.
- use Iterative method for membership function of all algorithms.

How to run program :

- Install gmpy library on your systerm
  - sudo apt-get install pyhton-gmpy
- Run nD_GDA_2c.py (1-3 Questions)
- Run naive_bayes_bernoulli.py (4^{th} Question)
- Run naive_bayes_binomial.py (5^{th} Question)

Results and Discussion

1D 2-class Gaussian Discriminant Analysis

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
1 - class	0.5839	0.5161	0.2759	0.3596	0.4161
0 - class		0.6038	0.8101	0.6919	

nD 2-class Gaussian Discriminant Analysis

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
1 - class	0.6204	0.6279	0.4286	0.5095	0.3796
0 - class		0.617	0.7838	0.6905	

nD k-class Gaussian Discriminant Analysis (iris dataset)

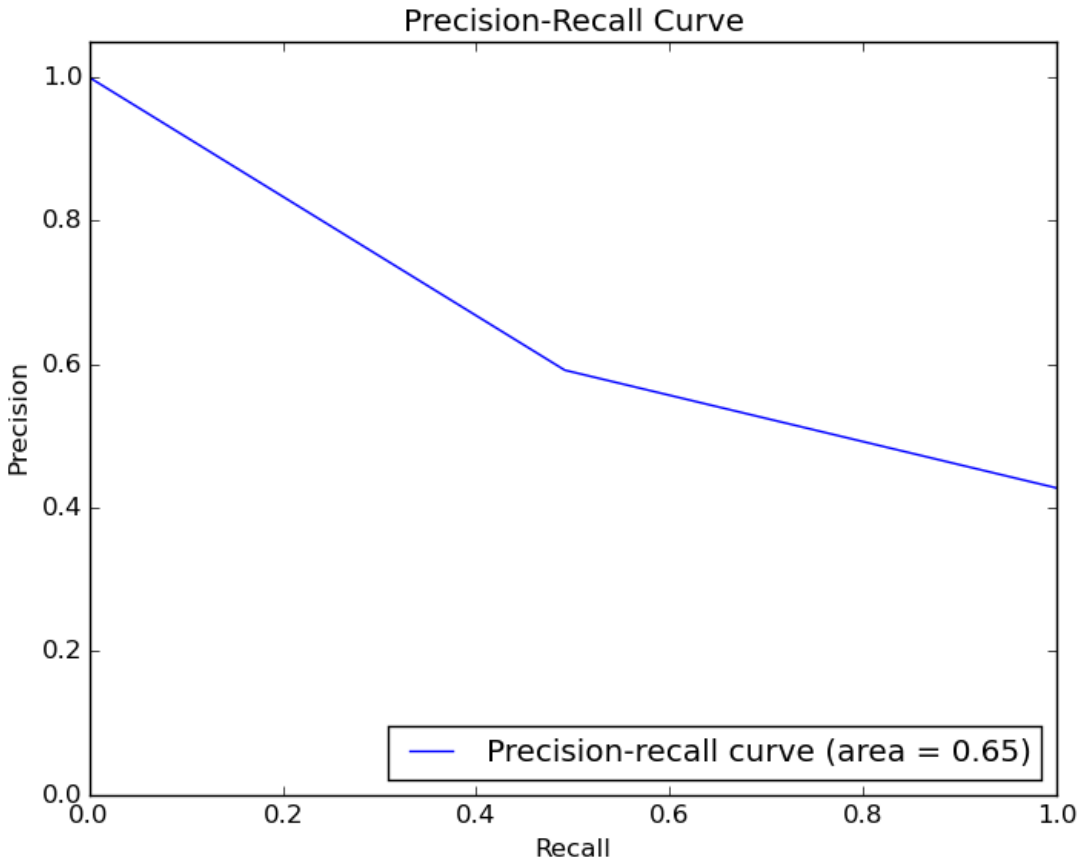
No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
0 - class	0.9333	1.0	1.0	1.0	0.0667
1 - class		1.0	0.8	0.889	
2 – class		0.8	1.0	0.889	

Output of Program nD_GDA_2c.py (K-Fold =10)

```
/usr/bin/python2.7 /home/dbp/PycharmProjects/Assignment2/nD_GDA_2c.py
-----------------------------------------------------------------------------------------------
                1-D 2-class Gaussian Discriminant Analysis
-----------------------------------------------------------------------------------------------

-------------------------------------------------------------------------
                Parameter Evaluation
-------------------------------------------------------------------------

MSE ::  0.4161      OR      41.61  %

Accuracy ::  0.5839     OR      58.39  %

Precision ::  {'1': 0.5161, '0': 0.6038}

Recall ::  {'1': 0.2759, '0': 0.8101}

F-measure ::  {'1': 0.3596, '0': 0.6919}

-----------------------------------------------------------------------------------------------
                n-D 2-class Gaussian Discriminant Analysis
-----------------------------------------------------------------------------------------------

-------------------------------------------------------------------------
                Parameter Evaluation
-------------------------------------------------------------------------

MSE ::  0.3796      OR      37.96  %

Accuracy ::  0.6204     OR      62.04  %

Precision ::  {'1': 0.6279, '0': 0.617}

Recall ::  {'1': 0.4286, '0': 0.7838}

F-measure ::  {'1': 0.5095, '0': 0.6905}

-----------------------------------------------------------------------------------------------
                n-D k-class Gaussian Discriminant Analysis
-----------------------------------------------------------------------------------------------

-------------------------------------------------------------------------
                Parameter Evaluation
-------------------------------------------------------------------------

MSE ::  0.0667      OR      6.67  %

Accuracy ::  0.9333     OR      93.33  %

Precision ::  {'Iris-virginica': 1.0, 'Iris-setosa': 1.0, 'Iris-versicolor': 0.8}

Recall ::  {'Iris-virginica': 1.0, 'Iris-setosa': 0.8, 'Iris-versicolor': 1.0}

F-measure ::  {'Iris-virginica': 1.0, 'Iris-setosa': 0.8889, 'Iris-versicolor': 0.8889}

-----------------------------------------------------------------------------------------------

Process finished with exit code 0
```

PRECISION – RECALL CURVE

- I used sklearn inbuilt library for precision recall library and find it for maximum accuracy fold of output.
- Area is mention at the bottom right corner.



Precision-Recall Curve

Naive Bayes-Bernoulli Features with nD and 2-Class

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
1 - class	0.9065	0.0.9167	0.8415	0.8775	0.0935
0 - class		0.9007	0.9495	0.9245	

```
/usr/bin/python2.7 /home/dbp/PycharmProjects/Assignment2/naive_bayes_bournoli.py

--------------------------------------------------------------------------------
                Parameter Evaluation
--------------------------------------------------------------------------------

MSE ::  0.0935      OR      9.35  %

Accuracy ::  0.9065     OR      90.65  %

Precision ::  {'1': 0.9167, '0': 0.9007}

Recall ::  {'1': 0.8415, '0': 0.9495}

F-measure ::  {'1': 0.8775, '0': 0.9245}


--------------------------------------------------------------------------------
                In Built Function Parameter
--------------------------------------------------------------------------------


MSE ::  0.0935      OR      9.35  %

Accuracy ::  0.9065     OR      90.65  %

Precision ::  0.9167

Recall ::  0.8415

F-measure ::  0.8775

Process finished with exit code 0
```

Naive Bayes-Binomial Features with nD and 2-Class

No. of Class	Accuracy	Precision	Recall	F-Measure	MSE
1 - class	0.8783	0.778	0.9669	0.8621	0.1217
0 - class		0.9745	0.8208	0.8911	

```
--------------------------------------------------------------------------------
                Parameter Evaluation
--------------------------------------------------------------------------------

MSE ::  0.1217          OR              12.17  %

Accuracy ::  0.8783     OR              87.83  %

Precision ::  {'1': 0.7778, '0': 0.9745}

Recall ::  {'1': 0.9669, '0': 0.8208}

F-measure ::  {'1': 0.8621, '0': 0.8911}


--------------------------------------------------------------------------------
                In Built Function Parameter
--------------------------------------------------------------------------------

MSE ::  0.1109          OR              11.09  %

Accuracy ::  0.8891     OR              88.91  %

Precision ::  0.8095

Recall ::  0.9639

F-measure ::  0.88

Process finished with exit code 0
```

Naïve Bayes Binomial Model Parameter Solution

$$l(\theta) = \left[\sum_{i=1}^{m} \sum_{j=1}^{n} \log \binom{p^{(i)}}{x_j^{(i)}} \alpha_{j|y=y^{(i)}}^{x_j^{(i)}} \cdot \left(p^{(i)} - \alpha_{j|y=y^{(i)}} \right)^{p^{(i)} - x_j^{(i)}} \right]$$

Now derivate with respect to $\alpha_{j|y=y^{(i)}}$ and equate it to zero

$$\frac{\partial l(\theta)}{\partial \alpha_{j|y=y^{(i)}}} = 0$$

$$\frac{\partial}{\partial \alpha_{j|y=y^{(i)}}} \left[\sum_{i=1}^{m} \sum_{j=1}^{n} \log \binom{p^{(i)}}{x_j^{(i)}} \alpha_{j|y=y^{(i)}}^{x_j^{(i)}} \left(p^{(i)} - \alpha_{j|y=y^{(i)}} \right)^{p^{(i)} - x_j^{(i)}} \right] = 0$$

$$\frac{\partial}{\partial \alpha_{j|y=y^{(i)}}} \left[\sum_{i=1}^{m} \sum_{j=1}^{n} \log \binom{p^{(i)}}{x_j^{(i)}} + x_j^{(i)} \log \alpha_{j|y=y^{(i)}} + \left(p^{(i)} - x_j^{(i)} \right) \log \left(1 - \alpha_{j|y=y^{(i)}} \right) \right] = 0$$

$$\therefore \left[0 + \sum_{i=1}^{m} \sum_{j=1}^{n} x_j^{(i)} \Big/ \alpha_{j|y=j}^{(i)} + \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\frac{(p^{(i)} - x_j^{(i)})}{(1 - \alpha_{j|y=y^{(i)}})} \right] (-1) \right] = 0$$

$$\therefore \sum_{i=1}^{m} \frac{x_p^{(i)}}{\alpha_{j|y=j}} = \sum_{i=1}^{m} \frac{(p^{(i)} - x_p^{(i)})}{(1 - \alpha_{j|y=j})}$$

$$\left[y^{(i)} = j \Big/ \text{class } j^{th} \right]$$

$$\therefore \frac{1}{\alpha_{j|y=j}} \sum_{i=1}^{m} x_p^{(i)} = \frac{1}{1 - \alpha_{j|y=j}} \left[\sum_{i=1}^{m} p^{(i)} - x_p^{(i)} \right]$$

$$\therefore \boxed{\alpha_{j|y=j} = \frac{\sum_{i=1}^{m} 1(y=j)\, x_p^{(i)}}{\sum_{i=1}^{m} 1(y=j)\, p^{(i)}}}$$

$$\frac{\partial l(\theta)}{\partial \alpha_k} = \frac{\partial}{\partial \alpha_k} \sum_{i=1}^{m} \log\left(P^{(i)}(y=k)\right)$$

α_k ↙ prior probability

$$0 = \frac{\sum_{i=1}^{m} 1(y=k)\, X}{\sum_{i=1}^{m} P(y=k)}$$

$$\boxed{P^{(i)}(y=k) = \frac{\sum_{i=1}^{m} 1(y=k)}{m}}$$

$k = 1, 2, 3 \ldots k$

COMMENTS ON RESULTS

- Error of 1D Gaussian Discriminant Analysis is bit high because I extracted single feature from 4 features datasets so my model is not trained fully. So that predicted results is not accurate.
- N-D 2-class GDA and nD k-class GDA has accurate results according to trained model. I checked with the inbuilt function named LDA.
- Naive Bayes Bernoulli and Naive Bayes Binomial is checked with inbuilt function named Bernoulli() and Multinomial() of sklearn.naive_bayes library.

REFERENCE

- NUMPY - http://www.numpy.org/
- http://scikitlearn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html#sklearn.discriminant_analysis.LinearDiscriminantAnalysis
- http://scikit-learn.org/stable/modules/naive_bayes.html
- http://scikitlearn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html#sklearn.metrics.precision_recall_curve