Introduction to Machine learning (CS5011)
Programming assignment #1

Aravind S
EE14B013

20th Sept. 2016

Contents

1 Synthetic Data-set Creation

1.1 Goal

Generate 2-classes data with 20 features each. Each class is given by a multivariate Gaussian distribution. Covariance matrix should not be spherical in nature. Generate 2000 samples for each class and make sure there exists some overlap within the classes. Partition them into train set and test set by randomly picking 70 % data of each class (1400 points each) and the remaining 30 %. This dataset is referred to as DS1.

1.2 Approach

We need to generate 2 different Gaussian distributions with a random centroid and they should sufficiently overlap. Also, the covariance matrix needs to be positive definite.

Let A be a randomly generated matrix of order pxp. Now the matrix B generated from A as follows will be positive definite.
$B = pI + \frac{1}{2}(A + A^T)$

Thus, generating the covariance matrix B like this, we can make sure that it is not spherical and positive definite.

1.3 Results

Thus the 2 distributions are generated and are labelled as 1 and 0 respectively. Performance reported on DS1 are on the test-set which accounts for 30 % of the generated data.

2 Linear Classification

2.1 Goal

To learn a linear classifier by using regression on an indicator variable. The accuracy, precision, recall and f-score on the train set are reported.

2.2 Approach

A linear classifier performs a regression on the data points (X) and the labels (Y). For a given $x \epsilon R^p$, say the label is y.
Now, a linear regression on X finds $\beta, \beta_0 \, \epsilon \, R^p$ such that,

$$y_{predicted} = \beta_0 + x^T \beta$$

subject to minimising,

$$\sum || \; y_{predicted} - y \; ||$$

The solution can be analytically obtained and is found to be,

$$[\beta_0, \beta] = (X^T X)^{-1} X^T Y$$

Classification is done by thresholding the $y_{predicted}$ for a given x and then the appropriate class is predicted. For instance, here if $y_{predicted} > 0.5$, Class-1 is predicted else Class-0 is predicted.

2.3 Results

Measure	Class - 0	Class - 1
Accuracy	75.5 %	
Precision	74.21 %	76.94 %
Recall	78.17 %	72.83 %
F-measure	0.7614	0.7483

Coefficients obtained:

β_0	β_1	β_2	β_3	β_4	β_5	β_6
0.500000	0.002000	-0.010288	-0.001176	0.000754	-0.002957	-0.001910
β_7	β_8	β_9	β_{10}	β_{11}	β_{12}	β_{13}
-0.016318	-0.017257	-0.015294	-0.021997	0.001851	-0.011457	-0.001448
β_{14}	β_{15}	β_{16}	β_{17}	β_{18}	β_{19}	β_{20}
0.004792	-0.027095	0.001261	-0.005440	-0.014305	0.005413	0.001738

Coefficients are attached as a .csv file for better readability.

3 k-NN classifier

3.1 Goal

To use k-NN (k Nearest Neighbours) to build a classifier to classify the points in DS1.

3.2 Approach

k-NN is a classifier which models a function which is region-wise constant. Given a point, k-NN predicts the majority class in the neighbourhood formed by the k nearest points from the trainset. As a result, it predicts the same class for a given region as the majority class remains the same in that neighbourhood.

Here, we have 2 classes, Class-0 and Class-1. Thus, it reduces to a binary classification problem. One important issue with k-NN is fixing the value of k. For a large k, decision boundaries are smoother, whereas for small k it is rigid and hard. Usually, it is determined after experimenting with different values of k. Here the different values of k are considered ranging from 3 to 10 and the performance is reported.

3.3 Results

k	Accuracy (in %)	Precision (in %)		Recall (in %)		F-score	
3	68.75	67.78	69.84	71.5	66.0	0.6959	0.6787
4	67.25	63.08	75.30	83.17	51.33	0.7175	0.6105
5	71.17	69.66	72.92	75.0	67.33	0.7223	0.7001
6	69.0	65.0	75.91	82.33	55.67	0.7265	0.6423
7	70.92	69.28	72.86	75.17	66.67	0.7210	0.6962
8	70.0	66.57	75.21	80.33	59.67	0.7281	0.6654
9	71.17	70.83	72.57	73.67	69.67	0.7222	0.7109
10	72.17	69.16	76.28	80.0	64.33	0.7419	0.6980

As we observe linear regression using indicator variables performs better than k-NN on the whole. For some values of k we get similar accuracies and F-scores, whereas for others the regressor performs much better than k-NN classifier.

4 Data imputation

4.1 Goal

We are given the Communities and Crime dataset from UCI repository. Some of the data points have some of the attributes missing. Goal is to fix the data points and fill in the missing values.

4.2 Approach

In order to fill in the missing values, we need to estimate the distribution of the attribute and extrapolate. Statistical measures like mean and median can be used to fill in the missing values. If the attribute is categorical or text, mode (most frequent observation) can be used to impute the missing values. Another approach is to learn a k-NN classifier on that attribute and predict the missing values.

Since the missing attribute is not categorical here, using mode is not suitable. We can't use k-NN as the attributes missing can not be assumed piecewise constant. Among mean and median, median seems to be a better choice as the values taken by the attribute are not uniformly distributed and are sort of discrete. Therefore it makes more sense, to impute the missing values using median.

4.3 Results

The missing values were imputed using median of the given values and complete dataset is attached in the source.

5 Linear Regression

5.1 Goal

The goal is to fit the above data using linear regression. Performance is analysed on the test data averaged over 5 different 80-20 splits.

5.2 Approach

A linear classifier performs a regression on the data points (X) and the labels (Y). For a given $x \epsilon R^p$, say the label is y.
Now, a linear regression on X finds $\beta, \beta_0 \, \epsilon \, R^p$ such that,

$$y_{predicted} = \beta_0 + x^T \beta$$

subject to minimising,

$$\sum || \; y_{predicted} - y \; ||$$

The solution can be analytically obtained and is found to be,

$$[\beta_0, \beta] = (X^T X)^{-1} X^T Y$$

5.3 Results

Performance of the best fit is reported below.

Residual error (averaged over 5 different 80-20 splits)	7.78916045921

Coefficients are attached as a .csv file for better readability.

6 Regularized Linear Regression

6.1 Goal

The goal is to fit the above data using regularized linear regression. Performance is analysed on the test data averaged over 5 different 80-20 splits, for different values of λ.

6.2 Approach

Sometimes the input data tends to have correlated features and as a result some weights tend to go beyond limits (to large values) and nearly cancel out the contribution of these two features to the final prediction. To prevent such solution, we constrain the weights to have small values. Instead of having a hard threshold on the norm of weights, we can formulate it as an optimization problem as shown below.

$$y_{predicted} = \beta_0 + x^T\beta$$

subject to minimising,

$$\sum || \ y_{predicted} - y \ || + \lambda||\beta||$$

Here λ is called the regularization parameter and other symbols have their usual meaning. λ captures the effect of high value of weights in the final solution. The solution can be analytically obtained and is found to be,

$$[\beta_0, \beta] = (X^TX + \lambda I)^{-1}X^TY$$

6.3 Results

Values for λ are varied from 10^{-4} to 10^4 in 18 steps in logspace. Larger the λ more is the emphasis on obtaining smaller coefficients.
Best fit for complete data without feature reduction is:

Lambda	Residual error
5.080218	7.363684

Coefficients are attached as a .csv file for better readability.

When we analyse the coefficients we see that, some of the coefficients are very small in magnitude (of the order of 10^{-4} and lesser). Features corresponding to these coefficients do not contribute much to prediction and can be regarded useless.
Thus, thresholding the coefficients and obtaining corresponding features, we can obtain a reduced set of features. Learning a linear regressor for different values of λ we get,

Lambda	Residual error
5.080218	7.2717319

As we can see, the performance on both, the original set of features and the reduced set of features is comparable but are obtained for different regularization parameters.

Coefficients for the above fit are attached as a .csv file for better readability.

7 Feature Extraction - PCA & LDA

7.1 Goal

We have been given a 3-dimensional dataset (referred to as DS3) consisting of 2 classes. We need to perform feature extraction using PCA and LDA on this data to obtain a reduced set of 1 feature and train linear regression with indicator variables on the train set. Performance on the test set is reported along with a plot of the data and classifier boundary.

7.2 Approach

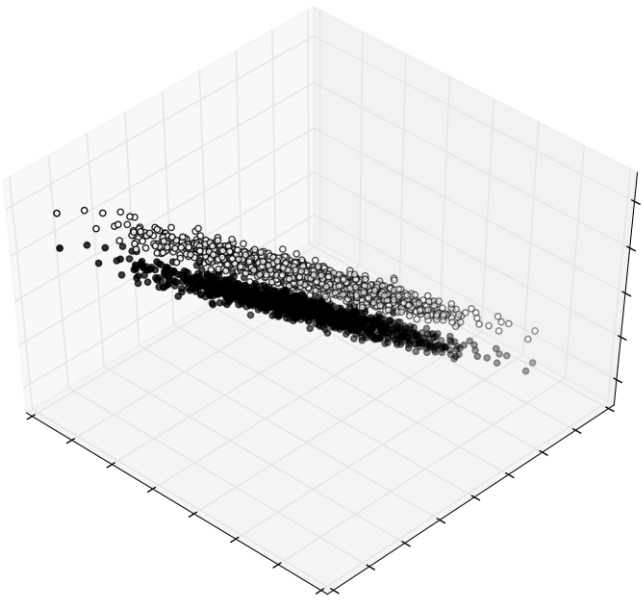
Generally, the given data has a lot of unnecessary information. For instance, the phone number of a customer will not influence whether he will buy a computer or not. As a result, choosing the important features and reducing the dimensionality of the data seems to be important. This process is referred to as feature extraction.

Principle component analysis is one such technique to reduce dimensionality. Applying PCA on the input data, gives us a basis sorted in the decreasing order of the variance of the input data along that direction. For instance, data has the maximum variance along the first principle component. So choosing a subset of these principle components will ensure reduction of dimensionality and thus help in picking up most relevant features.

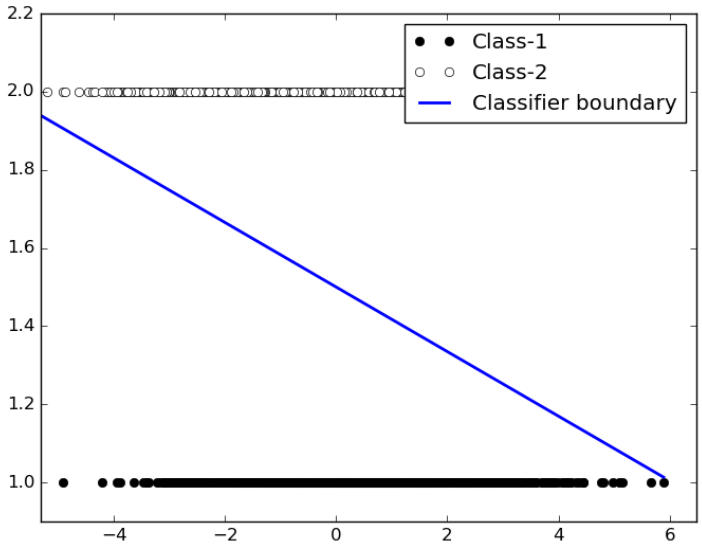
Linear discriminant analysis is another technique of feature extraction and takes into account the class distribution and models the difference between the classes. PCA on the other hand doesn't account any difference in class. So, when the class distributions overlap with each other, LDA will give a better solution than PCA as it considers class distribution and suggests features which best differentiate the classes.

7.3 Results

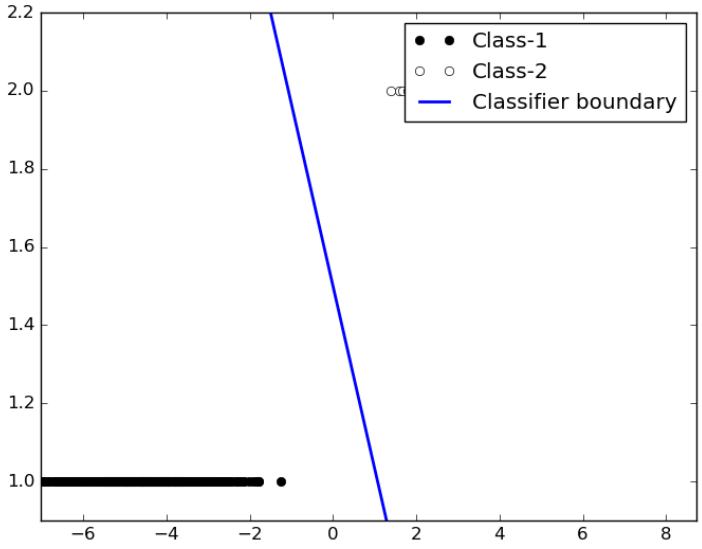
3D plot of the given dataset



Plot of projected data, using features extracted from PCA



Plot of projected data, using features extracted from LDA



From the two experiments, we infer that LDA performs much better than PCA in feature extraction. This is because PCA tries to identify the direction along which the given data has maximum variance, thus representing the data to the maximum extent possible with lesser dimensions. On the other hand, LDA tries to find the direction along which the points can be well differentiated and classified. Hence, in this case LDA performs better and it can be observed from the plot as the margin is higher in the case of LDA than PCA.

8 Logistic Regression

8.1 Goal

From the given dataset (called DS2) which consists of a number of images of different categories, we need to perform a 2-class logistic regression on it, by extracting features from it. Performance using both L1 and L2 penalty is analysed and reported.

8.2 Approach

Logistic regression is a regression model where the dependent variable is categorical. Using linear regression in such cases faces 2 problems: it might predict value outside the range of the values which the dependent variable can take, and also solutions get affected due to class imbalance. Logistic regression solves these problems by a different cost function which maximises likelihood.

A logistic regressor performs a regression on the data points (X) and the labels (Y). For a given $x \epsilon R^p$, say the label is y.
Now, logistic regression on X finds $\theta, \theta_0 \, \epsilon \, R^p$ such that,

$$y_{predicted} = h_\theta(x)$$

where,

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + x^T \theta)}}$$

and the cost function is minimised.

$$cost,\ L(h_\theta(x),\ y) = \begin{cases} -log(h_\theta(x)) & if\ y = 1 \\ -log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

This cost function maximises likelihood and our prediction is by modelling a sigmoid of a linear fit.

8.3 Results

For L2 penalty, we get the following results.

Measure	Class - 1	Class - 2
Accuracy	52.5 %	
Precision	52.0 %	53.33 %
Recall	65.0 %	40.0 %
F-measure	0.5778	0.0457

For L1 penalty, the best fit results are furnished below. They were obtained for regularization parameter, $\lambda = 10^{-6}$.

Measure	Class - 1	Class - 2
Accuracy	62.5 %	
Precision	57.57 %	85.71 %
Recall	95.0 %	30.0 %
F-measure	0.716981	0.444444

We see that, L1 regularised logistic regression performs better than L2 regularised logistic regression and we have obtained higher accruacy, precision, recall and F-measure in the former case. The scripts for running Boyd's l1_logreg code and analysis of it are present in the code directory.

9 Naive Bayes Classifier

9.1 Goal

To design and implement a Naive Bayes classifier to classify email messages as either spam (unwanted) or ham (useful).

9.2 Approach

Naive Bayes methods are a set of supervised learning algorithms based on the Bayes' theorem with the "naive" assumption of independence between every pair of features. Bayesian probability approach can be summarised as:

$$posterior = \frac{prior \times likelihood}{evidence}$$

Here *prior* captures the assumption of the solution we want and *evidence* describes the absolute probability distribution of the data given to us.
In this problem, four types of Naive Bayes methods are analysed.

- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- Dirichlet Naive Bayes
- Beta distribution based Naive Bayes

9.3 Results

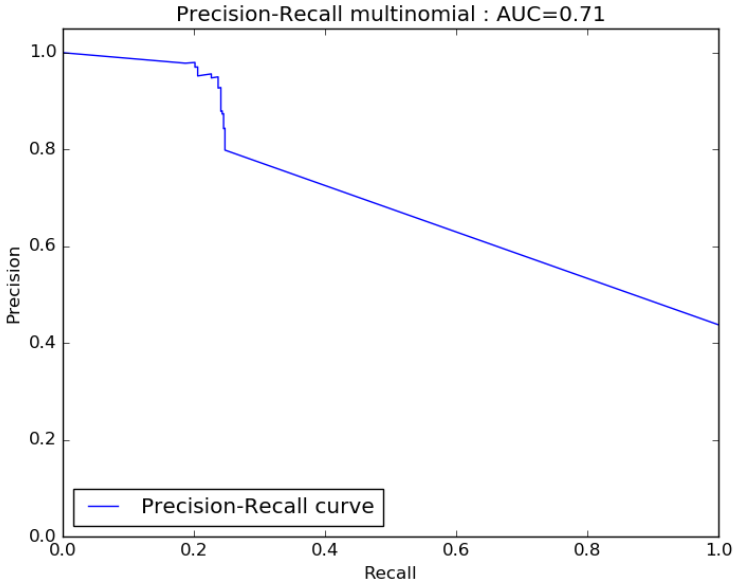
Performance measures are summarized below.

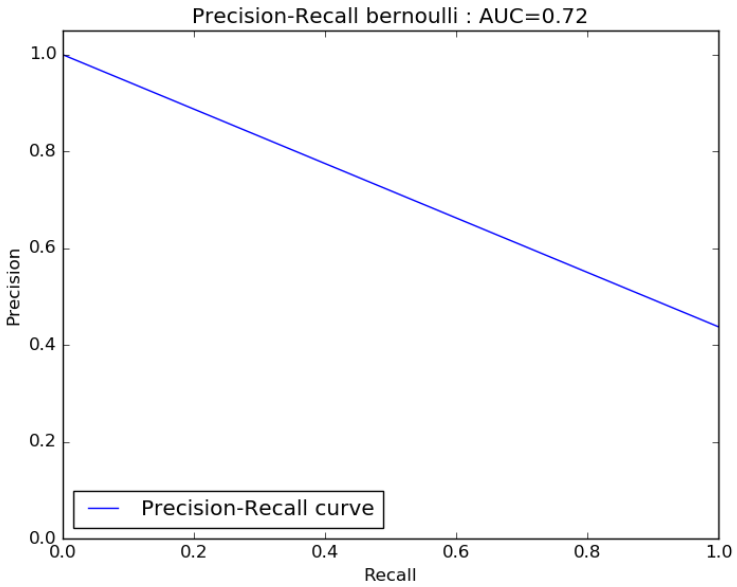
Multinomial Naive Bayes		Bernoulli Naive Bayes	
Measure	Value (spam class)	Measure	Value (spam class)
Accuracy	65.878509 %	Accuracy	56.232927 %
Precision	92.141619 %	Precision	0.0 %
Recall	24.112972 %	Recall	0.0 %
F-measure	0.38173784	F-measure	0.0
Dirichlet Naive Bayes		Beta Naive Bayes	
Measure	Value (spam class)	Measure	Value (spam class)
Accuracy	65.241308 %	Accuracy	56.232927 %
Precision	91.680952 %	Precision	0.0 %
Recall	22.658935 %	Recall	0.0 %
F-measure	0.36315229	F-measure	0.0

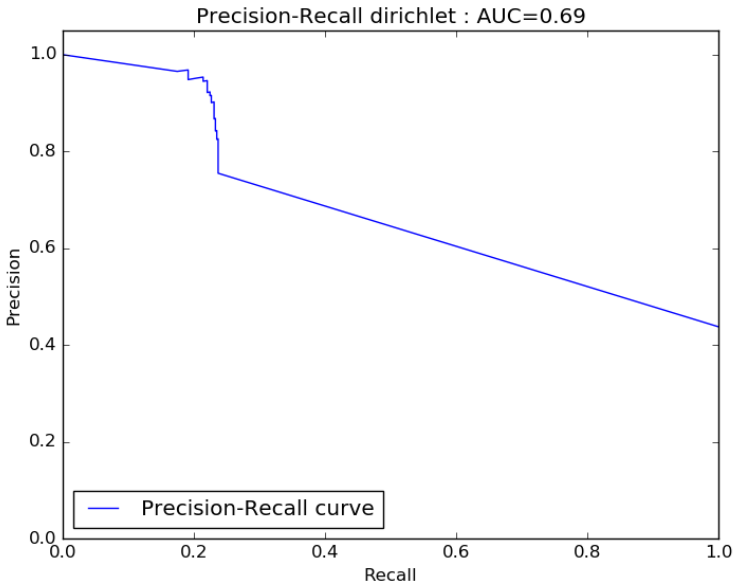
The best classifier among these is the Multinomial Naive Bayes estimator as it has the highest average F-measure.

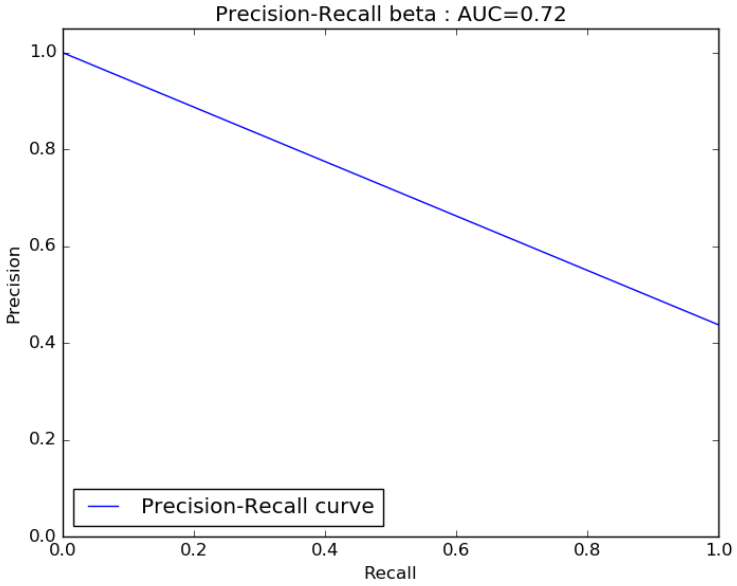
9.4 Plots

Precision-Recall plots of all the four methods are attached below.


Precision-Recall multinomial : AUC=0.71
Precision
Recall
Precision-Recall curve
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall bernoulli : AUC=0.72
Precision
Recall
Precision-Recall curve
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall dirichlet : AUC=0.69
Precision
Recall
Precision-Recall curve
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall beta : AUC=0.72
Precision
Recall
Precision-Recall curve
0.0
0.2
0.4
0.6
0.8
1.0