

BITS F464 - Machine Learning

Assignment 1

L Srihari

2017A7PS1670H

Rikil Gajarla

2017A7PS0101H

Raj Kashyap Mallala

2017A7PS0025H

Fisher's Linear Discriminant Analysis

Fisher linear discriminant approach aims to classify the points by reducing the dimension of input vector space. Consider first the case of two classes, and suppose we take the D-dimensional input vector x and project it down to one dimension using $y = w^T x$. Then we can place a threshold on $y = w_0$ and classify the point if $y \geq w_0$ as $C1$ and $y < w_0$ as $C2$.

Here we consider two class problem in which there are $N1$ points of $C1$ and $N2$ points of $C2$. We define mean vectors as

$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n, \quad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n.$$

The within class covariance is given by

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

The Fisher criterion is defined as

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}.$$

Writing the same in terms of w , S_B and S_W , where

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Then, maximising the above expression we get w as

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

To choose a threshold value y_0 , reduce the points of two classes into single dimension, fit a gaussian curve individually to both classes and find the point of intersection y_0 .

Results

For the given two datasets the following are scatter plots and gaussian curves.

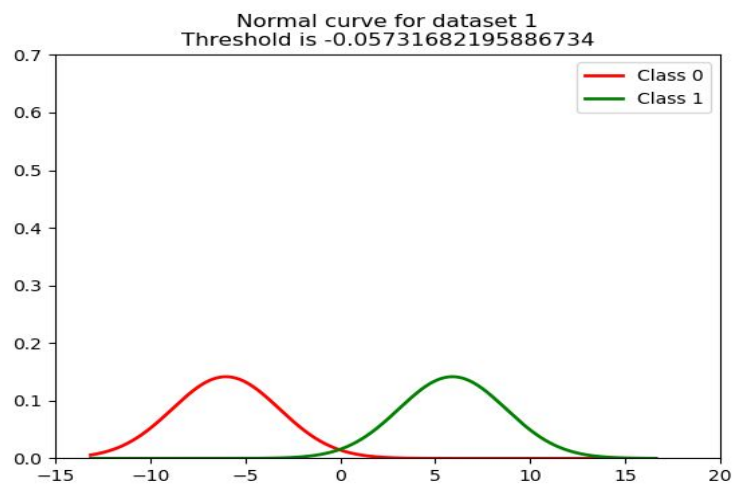
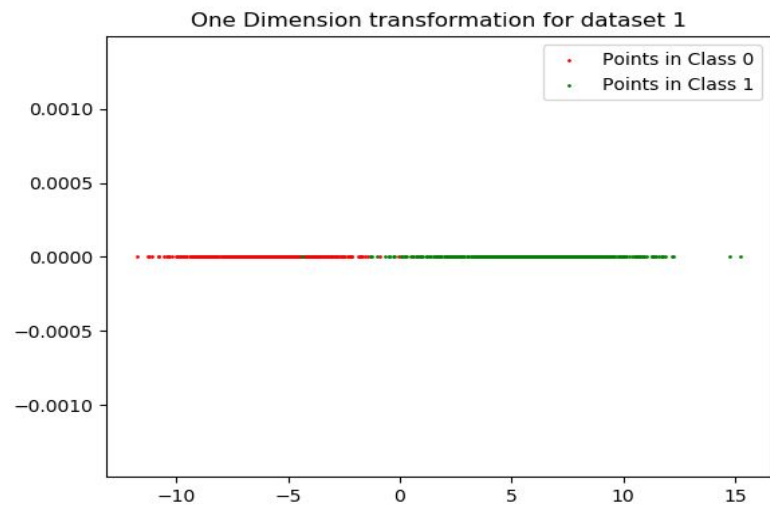
The threshold values are follows

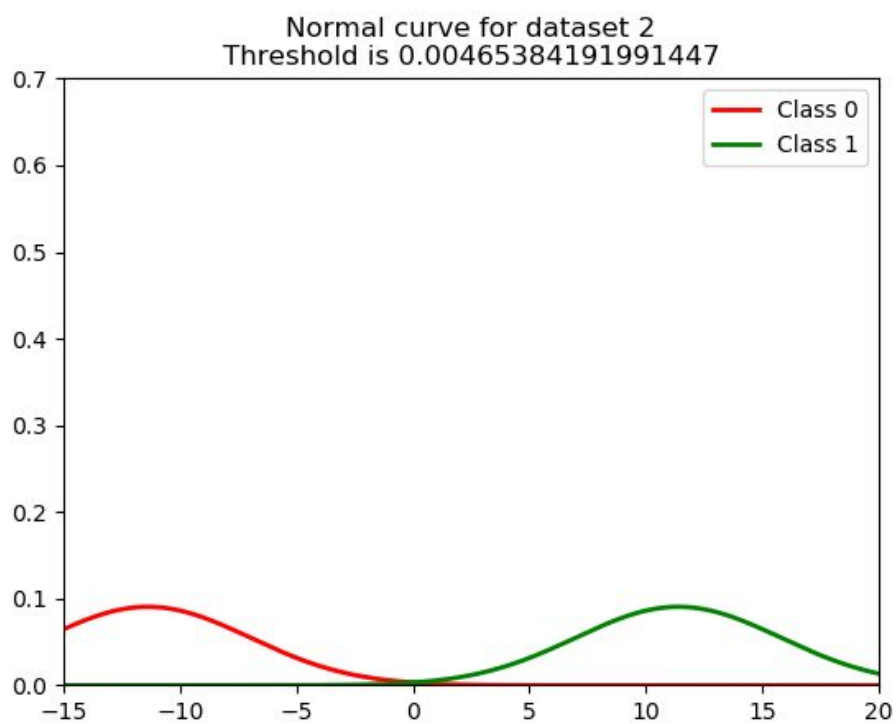
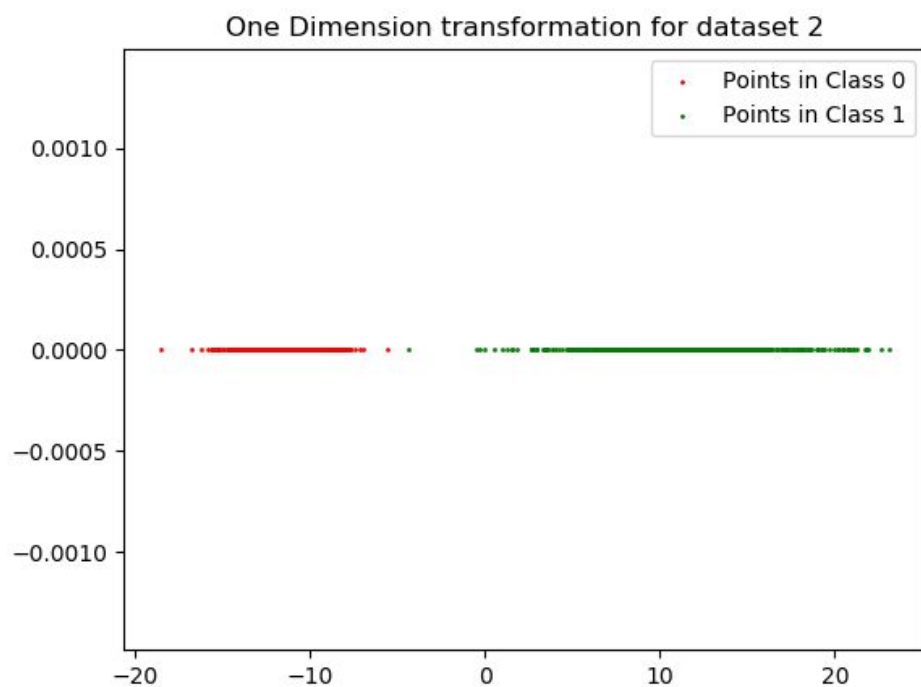
Dataset-1 : -0.05731682195886734

Accuracy : 99.3%

Dataset-2 : 0.00465384191991447

Accuracy: 100%





Sentiment Analysis Using Naive Bayes

The aim of this approach is to determine the sentiment of the user based on the review given by him/her and classify it as either satisfied(1) or unsatisfied(0). This approach involves preprocessing the review(as text), breaking down the user review into words and converting each review into its corresponding feature vector. We pre-calculate the probabilities of words which help in determining the target sentiment. When we get a new review, we apply the naive bayes assumption to decide the sentiment of the review.

Classification Using Bayes Theorem

We know from bayes theorem that

$$P(C_k / x) = \frac{P(C_k) \cdot P(x / C_k)}{P(x)}$$

Where C_k is the kth class and x is the d dimensional feature vector $x = (x_1, x_2, \dots, x_d)$.

The naive bayes assumption states that each feature x_i of vector x is conditionally independent of all other features $x_j \forall j, j \neq i$, for a given class C_k . using this assumption, the above bayes equation can be rewritten as follows

$$P(C_k / x) = \frac{P(C_k) \cdot \prod_{i=1}^d P(x_i / C_k)}{P(x)}$$

From the above equation, we can observe that the probability of x belonging to class C_k is proportional to numerator as denominator is a fixed constant for all features for a given k . Hence,

$$P(C_k / x) \propto P(C_k) \prod_{i=1}^d P(x_i / C_k)$$

So, whichever class maximizes the RHS probabilities, that is the most possible class the review must belong to. To avoid underflow in code computations, the above problem can be modelled to a similar problem of maximizing the log of probabilities. So, the final equation is

$$y = \operatorname{argmax}_k (\ln(P(C_k)) + \sum_{i=1}^d \ln(P(x_i / C_k)))$$

Where y is the target class number

Data Preprocessing

One of the most important steps to get good accuracy is to preprocess data to make it clean. Standard preprocessing techniques include:

- Converting text to lowercase
- Stripping text of any invalid characters and punctuation symbols
- Removing stop words (most common words in english like a, the, etc)
- Performing techniques like stemming and lemmatization

One or more of the above steps can be followed and the results of the model may vary based on preprocessing. After preprocessing, each review is converted to a vector whose length is the total no of words present in the database. The value of each index in the vector corresponds to the word being present(1) or absent(0) in the review.

Data Analysis/Training

Our final equation requires probabilities of each class C_k and probabilities of each word x_i conditioned on class C_k . Probabilities of each class C_k can be found out by counting the no of reviews which are classified in C_k divided by total no of reviews. And, each feature x_i being a word, $P(x_i / C_k)$ boils down to no of times word x_i appears in reviews of class C_k divided by total no of words in C_k . So,

$$P(x_i / C_k) = \frac{N_{ik} + \alpha}{N_k + \alpha n}$$

Here N_{ik} is the number of times ith feature(x_i) appears in class C_k , N_k is the number of words in class C_k . α Here is the smoothing factor which lies between (0, 1) and helps cases when we encounter a new word not in the training set. When α is set to 1, it's called laplacian smoothing.

Code Analysis

Python with numpy and pandas libraries were used to write code for this classifier.

Nltk package was used for getting a list of stopwords and using the word stemmer (Porter Stemmer).

A 5-fold cross validation was used to determine the accuracy of the model

Results

Accuracy obtained from using naive bayes model by applying lowercase, punctuation removal and smoothing factor 1 is an accuracy of $80.1\% \pm 2.56\%$ and f-score of 0.8030 ± 0.028 . With fold accuracies:

- Fold 1: 84%, F-Score: 0.841
- Fold 2: 81.5%, F-Score: 0.832
- Fold 3: 76.5%, F-Score: 0.775
- Fold 4: 80.5%, F-Score: 0.789
- Fold 5: 78.5%, F-Score: 0.7749

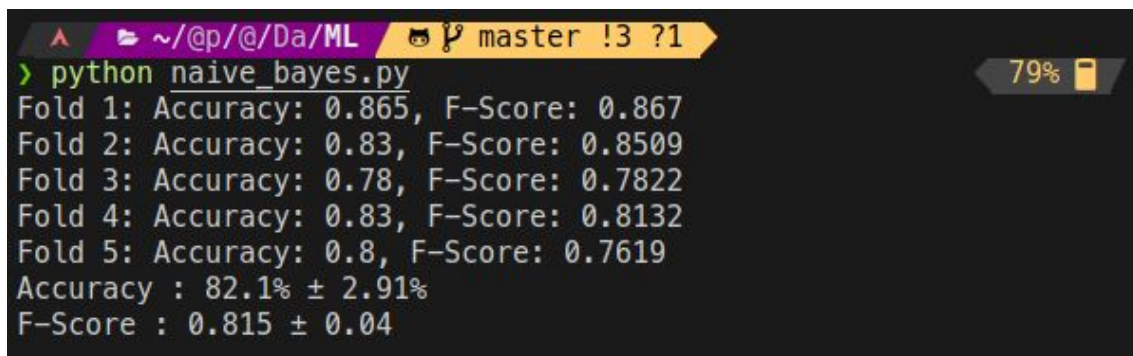
Including advanced techniques like Stemming, we found an increase $\sim 1\%$ in mean accuracy with approximately same variance with accuracy of $81\% \pm 2.69\%$ and f-score of 0.80 ± 0.041 with fold accuracies:

- Fold 1: 85%, F-Score: 0.851
- Fold 2: 83%, F-Score: 0.854
- Fold 3: 77.5%, F-Score: 0.778
- Fold 4: 81%, F-Score: 0.784
- Fold 5: 79%, F-Score: 0.752

By using Lemmatization, we were able to gain at most an additional $\sim 1\%$ in mean accuracy giving an F-Score of $0.815 \pm 0.04\%$ with fold accuracies:

- Fold 1: 86.5%, F-Score: 0.867
- Fold 2: 83%, F-Score: 0.850
- Fold 3: 78%, F-Score: 0.782
- Fold 4: 83%, F-Score: 0.813
- Fold 5: 80%, F-Score: 0.761

Setting smoothing factor to 0 is giving a reduced accuracy of $75.6\% \pm 2.87\%$ i.e., $\sim 5\%$ less than the highest. Other text processing like stopwords removal and stemming are also giving a reduced mean accuracy of around 70-72% only



```
~/@p/@/Da/ML master !3 ?1
> python naive_bayes.py
Fold 1: Accuracy: 0.865, F-Score: 0.867
Fold 2: Accuracy: 0.83, F-Score: 0.8509
Fold 3: Accuracy: 0.78, F-Score: 0.7822
Fold 4: Accuracy: 0.83, F-Score: 0.8132
Fold 5: Accuracy: 0.8, F-Score: 0.7619
Accuracy : 82.1% ± 2.91%
F-Score : 0.815 ± 0.04
```