CSCI 5525 Machine Learning Assigment 1

Ritiz Tambi

tambi004

Utilizing 1 Grace Day

Problem 1

1) Given

data (x, y) has joint distribution $p(x,y) = p(y|x)p(x)$

expected loss of a function $f(x)$ to model y using loss function $E[l(f(x), y)]$

$$E[l(f(x), y)] = \int_x \left\{ \int_y l(f(x), y)\, p(y|x)\, dy \right\} p(x)\, d(x)$$

$$= \int_x \left\{ \int_y l(f(x), y)\, p(y|x) p(x)\, dy \right\} dx$$

$$= \int_x \int_y l(f(x), y)\, p(x, y)\, dy\, dx$$

Hence, we minimize the expected loss to get optimal f.

$$\frac{\delta}{\delta f} \int_y l(f(x), y)\, p(x, y)\, dy = 0$$

$$\Rightarrow \int_y \frac{\delta}{\delta f} l(f(x), y)\, p(x, y)\, dy = 0 \quad - (1)$$

(Ignoring integral over x as it can be chosen independent of $f(x)$)

a) $l(f(x), y) = (f(x) - y)^2$, substituting in 1

$$\int_y \frac{\delta}{\delta f} (f(x) - y)^2 p(x, y)\, dy = 0$$

$$\Rightarrow \int_y (f(x) - y)\, p(x, y)\, dy = 0$$

$$\Rightarrow \int_y f(x)\, p(x,y)\, dy = \int_y p(x,y)\, y\, dy$$

$$\Rightarrow f(x) \underbrace{\int_y p(x,y)\, dy}_{\text{Marginalizing}} = \int_y p(x,y)\, y\, dy$$

$$\Rightarrow f(x) \cdot p(x) = \int_y y \cdot p(x,y)\, dy$$

$$f(x) = \int_y y \cdot \frac{p(x,y)}{p(x)}\, dy$$

$$= \int y \cdot p(y/x)\, dy$$

$$\Rightarrow \boxed{f(x) = E_y[y/x]}$$. Optimal $f(x)$ is conditional mean of y given x

b) $l(f(x), y) = |f(x) - y|$. Substituting in 1

$$\frac{\partial}{\partial f} \int_y |f(x) - y|\, p(x,y)\, dy = 0$$

$$= \int_{-\infty}^{f(x)} p(x,y)\, dy + \int_{-f(x)}^{\infty} (-1)\, p(x,y)\, dy = 0$$

$$\Rightarrow \int_{-\infty}^{f(x)} p(x,y)\, dy = \int_{-f(x)}^{\infty} p(x,y)\, dy$$

$f(x)$ satisfying this is the conditional mean of y given x

Problem 2

(a) Given

data with joint distribution $p(x, y)$

y - M-valued discrete target variable

we have $$P(C_k|x) = \frac{P(x|C_k)\,P(C_k)}{\sum_{k=1}^{M} P(x|C_k)\,P(C_k)}$$ - Bayes Rule

For a region R_k

$\forall x \in R_k$, "correctly labeled as C_k"

$$P(x \in R_k, C_k) = \int_{x \in R_k} p(x, C_k)\,dx$$ [In this region $p(y=C_k|x) > p(y=C_j|x)$ $\forall j \neq k$]

Misclassification for Region k, for all the pairs that have been misclassified

$$P(x \in R_k, C_j)_{j \neq k} = \sum_{\substack{j=1 \\ j \neq k}}^{M} \int_{x \in R_k} p(x, C_j)\,dx \quad -(1)$$

Hence for a new data pt., the probability of it being misclassified

is $\sum_{j=1}^{M} P(\text{Misclassification for } x \text{ and } C_j)$

$= \cancel{\sum_{j=1}^{M} P(x)} = \sum_{j=1}^{M} \sum_{\substack{i=1 \\ i \neq j}}^{M} P(x \in R_i, C_j)$

For a single class j

$$\boxed{\text{error}[j] = \sum_{\substack{i=1 \\ i \neq j}}^{M} P(x \in R_i, C_j)}$$ — Misclassification prob for that class

$$= \sum_{i=1}^{M} P(x \in R_i, C_j) - P(x \in R_j, C_j)$$

$$= \sum_{i=1}^{M} \int_{x \in R_i} P(x, C_j)\,dx - \int_{x \in R_j} P(x, C_j)\,dx$$

$$\Downarrow$$

$$= \sum_{i=1}^{M} \int_{x \in R_i} P(C_j \mid x)\,p(x)\,dx - \int_{x \in R_j} p(C_j \mid x)\,p(x)\,dx$$

Problem 3

To obtain HomeVal50 dataset we threshold at the median. The instructions and assumption for the code are given in ReadMe.

a)

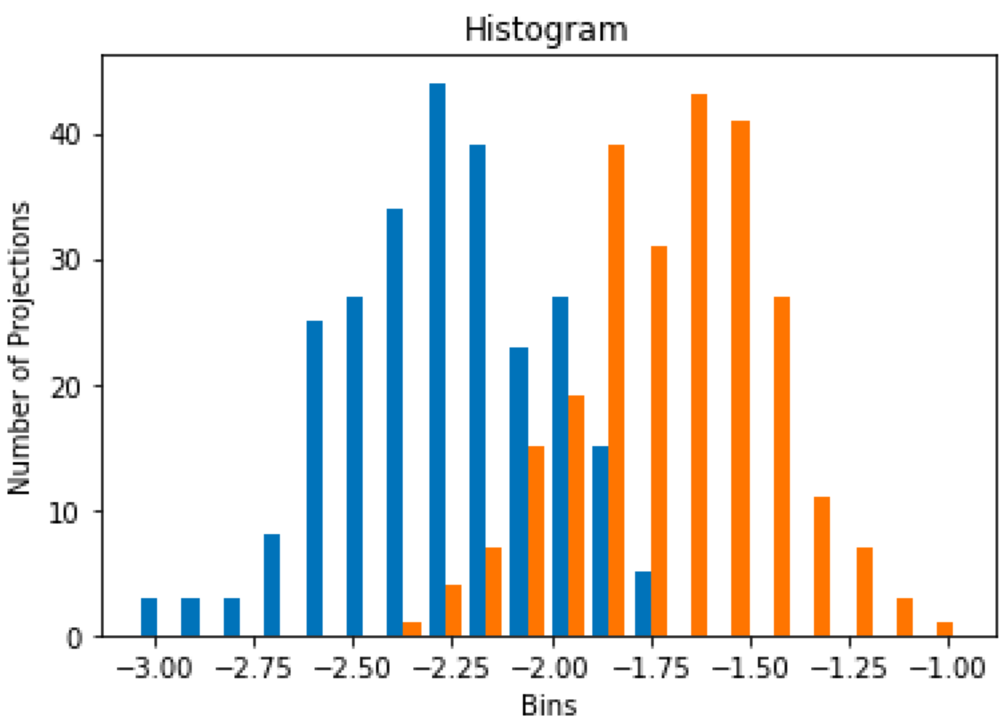
To apply LDA, we compute S_B and S_W

$S_B = (m_2 - m_1)(m_2 - m_1)^T$, where m_1 and m_2 are means of classes 0 and 1 respectively.

$S_W = \sum_{x_n \in C_1}(x_n - m_1)(x_n - m_1)^T + \sum_{x_n \in C_2}(x_n - m_2)(x_n - m_2)^T$, where C_1 refers to the class with y=0 and C_2 refers to the class with y=1.

We get the eigenvectors from $(S\_W)^{-1}$*S_B

We sort eigenvectors in decreasing order w.r.t their eigenvalues and take the largest k-1 vectors, where k refers to number of classes.



The data is decently modelled by LDA. Even though the Within Class Scatter is quite high, the between class scatter is good enough to compensate for that.

b)

No, we cannot use LDA to project Boston50 (HomeVal50) dataset to R2. For a K class problem, we can project the data to maximum K-1 dimensions. This comes from the observation that at most K-1 independent eigenvectors are obtained as S_B has max rank of K-1. Hence the projection matrix will not contain more than K-1 eigenvectors.

For our problem K = 2, (0 and 1)
Hence, we can project the data to maximum of 1 dimension.

c)

We can apply LDA to digits dataset. We use the same S_B and S_W equations as above. In this case S_W will consist of sum of 10 classes.
We then take two eigenvectors corresponding to largest eigenvalues to form our projection matrix.
We project the original data to obtain the 2D projected data X.
For Gaussian Generative Modelling, we learn Gaussian parameters on the training data in cross-validation and apply it to subsequent test fold.

$\log(P(C_k|x)) = \log(P(x|C_k)) + \log(P(C_k))$

$P(x|C_k) = \frac{1}{2\pi^{(d/2)}} * \frac{1}{|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\}$,
where μ_k is mean for Class k, Σ is covariance

$P(C_k)$ is prior probability for Class k
$P(C_k)$= (Number of elements of Class K in Training data)/(Training sample size)

The class k for which $\log(P(C_k|x))$ is assigned to the test tuple.

```
Train error for cross_validation_iteration 1 is  28.88064316635745
Test error for fold 1 is  26.81564245810056

Train error for cross_validation_iteration 2 is  28.509585652442794
Test error for fold 2 is  31.843575418994412

Train error for cross_validation_iteration 3 is  28.385899814471244
Test error for fold 3 is  31.28491620111732

Train error for cross_validation_iteration 4 is  28.509585652442794
Test error for fold 4 is  30.16759776536313

Train error for cross_validation_iteration 5 is  29.066171923314783
Test error for fold 5 is  24.581005586592177

Train error for cross_validation_iteration 6 is  28.385899814471244
Test error for fold 6 is  31.843575418994412

Train error for cross_validation_iteration 7 is  28.324056895485466
Test error for fold 7 is  28.49162011173184

Train error for cross_validation_iteration 8 is  28.57142857142857
Test error for fold 8 is  27.932960893854748

Train error for cross_validation_iteration 9 is  28.324056895485466
Test error for fold 9 is  30.726256983240223

Train error for cross_validation_iteration 10 is  28.801986343885783
Test error for fold 10 is  28.64864864864865

Standard Deviation for Test errors is : 2.258533506422999
```

Problem 4

a) **Logistic Regression (LR) with Iteratively Reweighted Least Squares**

Some Sources followed:
http://www2.maths.lth.se/matematiklth/personal/fredrik/Session3.pdf
https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares

Algorithm:

Training

Initialize W as a Random_Normal_Matrix(-0.001,0.001) of Size[No. of Classes, No. of Features]

For each iteration t till Convergence

$B^{(t)} = (X^T W^{(t)} X)^{-1} (X^T W^{(t)} y)$

For i in data X:

$W_i^{(t)} = 1/\max\{epsilon, y_i - X_i B^{(t)}\}$

$Test = sum(|W^{(t)} - W^{(t-1)}|)$

If Test < Threshold : Convergence is True

Return W

Testing

*X_test_scores = X_test*W*^{T}

Y_test_pred = []

For each x in X_test:

*x_test_scores = x *W*^{T}

softmax_scores = softmax(x_test_scores)

y_ = c , where c = max(softmax_score) in all classes

Add y_ to Y_test_pred

Error

*(1- count(Y_test == Y_test_pred) / size(Y_test))*100*

Note – Taking epsilon = 0.05 (Best Results obtained for this value as found out through simulations)

b) **Naïve-bayes with marginal Gaussian Distributions (GNB)**

Algorithm:
Training

Testing
Y_test_pred = []
For each x in X_test:

$$P(x|C_k) = \frac{1}{2\pi^{(d/2)}} * \frac{1}{\prod_{i=1}^{D} \sigma_{ik}} \exp\{-1\backslash 2 * \frac{(x\_i - \mu_{ik})^2}{2\sigma_{ik}^2}\}$$

p(x|C_k*)* = $\prod_{i=1}^{D} p(x\_i|C_k)$
$\log(P(C_k|x)) = \log(P(x|C_k)) + \log(P(C_k))$
y_ = c , where c = max($\log(P(C_k|x))$*) in all classes C*$_k$
Add y_ to Y_test_pred

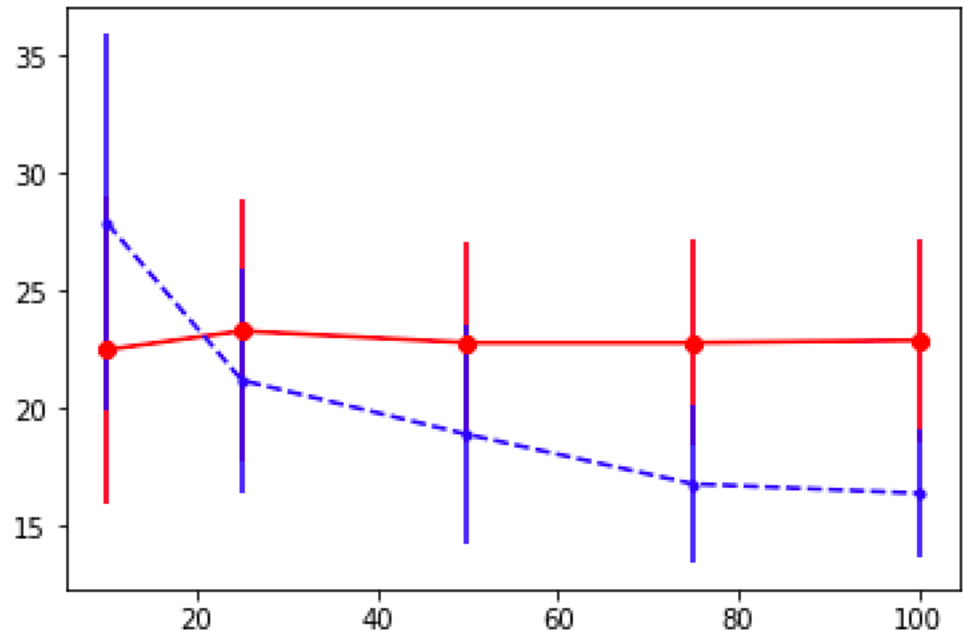
Error
*(1- count(Y_test == Y_test_pred) / size(Y_test))*100*

Dataset HomeVal50
Logistic Regression

```
Test_Error for Train Percentage  10  across all splits:  27.9
Test_Error for Train Percentage  25  across all splits:  21.2
Test_Error for Train Percentage  50  across all splits:  18.9
Test_Error for Train Percentage  75  across all splits:  16.8
Test_Error for Train Percentage  100  across all splits:  16.4
```

Naïve_Bayes

```
Test_Error for Train Percentage  10  across all splits:  22.5
Test_Error for Train Percentage  25  across all splits:  23.3
Test_Error for Train Percentage  50  across all splits:  22.8
Test_Error for Train Percentage  75  across all splits:  22.8
Test_Error for Train Percentage  100  across all splits:  22.9
```



Blue Line – Logistic Regression,   Red Line – NaïveBayes

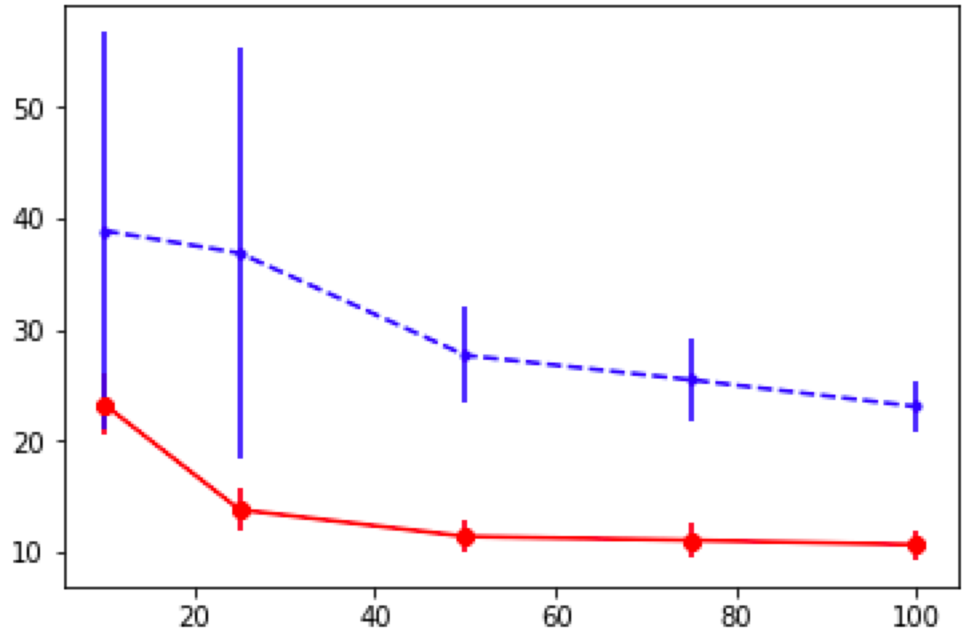
Dataset Digits

Logistic Regression

```
Test_Error for Train Percentage  10  across all splits:  38.87005649717514
Test_Error for Train Percentage  25  across all splits:  36.86440677966102
Test_Error for Train Percentage  50  across all splits:  27.68361581920904
Test_Error for Train Percentage  75  across all splits:  25.480225988700568
Test_Error for Train Percentage  100  across all splits:  23.07909604519774
```

Naïve_Bayes

```
Test_Error for Train Percentage  10  across all splits:  23.27683615819209
Test_Error for Train Percentage  25  across all splits:  13.8135593220339
Test_Error for Train Percentage  50  across all splits:  11.412429378531073
Test_Error for Train Percentage  75  across all splits:  11.073446327683616
Test_Error for Train Percentage  100  across all splits:  10.677966101694915
```



Blue Line – Logistic Regression,   Red Line – NaïveBayes