CSCI 5525 - Homework 1

Saurabh Mylavaram (mylav008@umn.edu id#:5593072)

September 27, 2020

1 Problem 1

1.1 Part a: Squared loss

We first substitute $l(f(x), y) = (f(x) - y)^2$ into the equation.

$$E_{(x,y)}\left[l(f(x), y)\right] = \int_x \left\{ \int_y l(f(x), y) p(y|x) dy \right\} p(x) dx$$
$$E_{(x,y)}\left[l(f(x), y)\right] = \int_x \left\{ \int_y (f(x) - y)^2 p(y|x) dy \right\} p(x) dx$$

We see that the inner integral is always ≥ 0. To minimize the expected loss it suffices to minimize the integral (point wise).

$$f^* = \underset{f}{\operatorname{argmin}} \int_y (f(x) - y)^2 p(y|x) dy$$

To minimize this, we take derivative with f and set it to 0.

$$\frac{d}{df}\left(\int_y (f(x) - y)^2 \, p(y|x) dy \right) = 0$$
$$\int_y \frac{d}{df} (f(x) - y)^2 \, p(y|x) dy = 0$$
$$\int_y 2(f(x) - y) \, p(y|x) dy = 0$$
$$\int_y f(x) p(y|x) dy = \int_y y \, p(y|x) dy$$
$$f^*(x) = E(y|x)$$

1.2 Part b: Absolute deviation loss

Similar to the previous part,

$$f^* = \underset{f}{\operatorname{argmin}} \int_y |f(x) - y| p(y|x) dy$$

Derivative with respect to $f(x)$ and set to 0:

$$\frac{d}{df}\left(\int_y |f(x) - y| \, p(y|x) dy \right) = 0$$
$$\int_y sign(f(x) - y) \, p(y|x) dy = 0$$
$$\int_{-\infty}^{f(x)} -p(y|x) dy + \int_{f(x)}^{\infty} p(y|x) dy = 0$$
$$\int_{-\infty}^{f(x)} p(y|x) dy = \int_{f(x)}^{\infty} p(y|x) dy$$

Optimal $f(x)$ i.e. $f^*(x)$ must satisfy this condition. Which means $f^*(x)$ must be the *conditional median* of y. (Since the cumulative probability on both sides of f(x) is equal)

2 Problem 2

If we work backwards and expand the RHS of the equation,

$$\begin{aligned} ||Aw-b||_2^2 + c^T w + d &= ||(Aw-b)+f||_2^2 + g \\ &= ((Aw-b)+f)^T((Aw-b)+f) + g \\ &= ||Aw-b||_2^2 + 2f^T(Aw-b) + f^T f + g \\ &= ||Aw-b||_2^2 + (2f^T A)w + (||f||^2 - 2f^T b + g) \end{aligned}$$

Therefore the LHS can be expressed in the form of RHS where $c = 2fA^T$ and $d = (||f||^2 - 2f^T b + g)$

To minimize the RHS of the equation, we compute its gradient and set it to 0.

$$\begin{aligned} \frac{d}{dw}\left(||Aw-(b-f)||_2^2 + g\right) &= 0 \\ \frac{d}{dw}\left(w^T A^T Aw - 2w^T A^T(b-f) + ||b-f||_2^2 + g\right) &= 0 \\ 2A^T Aw - 2A^T(b-f) &= 0 \\ \boxed{w^* = (A^T A)^{-1} A^T (b-f)} \end{aligned}$$

3 Problem 3

3.1 LDA method

In Fischer's LDA, we project the high-dimensional data into 1D or 2D space in which the separation between class is maximized. To compute the vectors onto which data should be projected, we need two matrices (i) Within class co-variance S_w and (ii) between class co-variance S_b. These are given by the following equations:

$$S_w = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} (x_i - \mu_k)(x_i - \mu_k)^T \qquad S_b = \sum_{k=1}^{K} N_k (\mu_k - \mu)(\mu_k - \mu)^T$$

Here, K is the number of classes, μ_k is the mean of all data points belonging to the class C_k. μ is the mean of all data points in the dataset. N_k is the number of points in kth class.

After calculating these matrices, we need to find the eigenvectors of $S_w^{-1} S_b$ and sort them in the descending order of their eigenvalues. We can select the eigenvector with highest eigenvalue for 1D, and the top two for 2D case. Each eigenvector will be of the size Dx1. where D is the dimensionality of the data.

The dataset can be transformed into lower dimensions by taking dot-product of each datapoint(1xD) with the eigenvectors.

3.2 Classification

After the dataset is projected onto lower dimensions (1D or 2D) we can perform classification in this space. We'll assume that the distribution of datapoints of each class is a Gaussian. The mean and co-variance for the Gaussian distribution of each class can be estimated from the training data which belong to the class. Mean is given by sample mean and the covariance is the sample covariance. These are the maximum likelihood estimates.

$$\mu_k = \sum_{i \in \mathcal{C}_k} x_i \qquad \Sigma_k(i,j) = \sigma(x_i, x_j) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)(x_j - \mu)$$

After obtaining the mean and covariance estimates for each class in the lower dimension, any new datapoint can be classified into the class whose Gaussian give the highest probability for the point.

$$Prob(x_i \in \mathcal{C}_k) = \mathcal{N}(x_i, \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)\right)$$

$$y_i(predicted) = \underset{k}{\operatorname{argmax}} \left[Prob(x_i \in \mathcal{C}_k)\right]$$

3.3 Results

Raw output from terminal for each file is shown below:

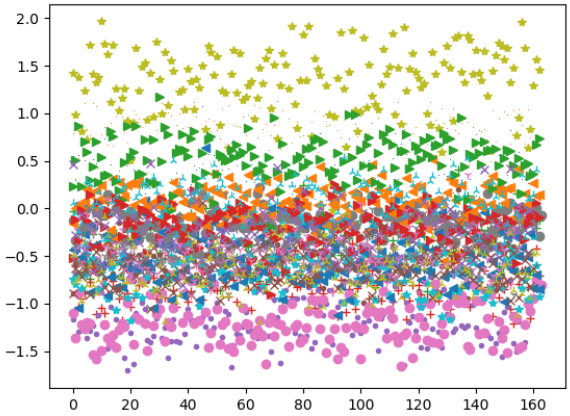
Results for 1D LDA for Boston50 dataset:

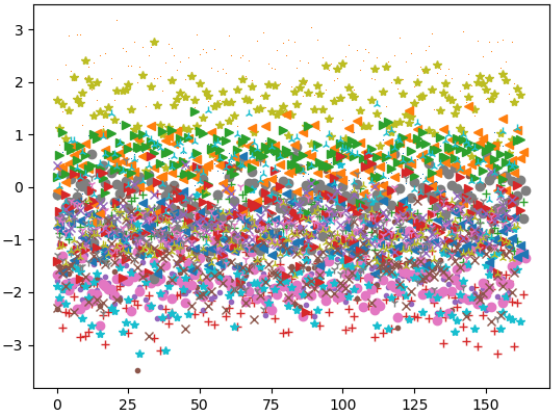
```
Running 1-D LDA on Boston-50 dataset...
Training Error mean:24.75% std:0.01%
Testing Error mean:25.33% std:0.06%
```

Results for 2D LDA for Digits dataset:

```
Running 2-D LDA on Digits dataset...
Training Error mean:29.26% std:0.01%
Testing Error mean:30.72% std:0.04%
```

We can also visualize the separation between points belonging to each class in the projected 2D space. Here each color is a different class.





4 Problem 4

4.1 Logistic Regression

For logistic regression, we assume the posterior probability of a class as the sigmoid of a linear function on the feature vector.

$$p(\mathcal{C}_k|x) = \sigma(\mathrm{w}^T x) = \sigma(z) \qquad \text{where} \quad \sigma(a) = \frac{1}{1+e^{-a}}$$

By training the algorithm, we want to learn the weights w which minimize the cross-entropy loss function shown below

$$E(\mathrm{w}) = -\sum_{i=1}^{N} \left[y_i \log(p(\mathcal{C}_k|x)) + (1-y_i)\log(1-p(\mathcal{C}_k|x)) \right]$$

$$E(\mathrm{w}) = -\sum_{i=1}^{N} \left[y_i \sigma(z) + (1-y_i)\log(1-\sigma(z)) \right]$$

Gradient for this error function is given by:

$$\nabla E(\mathrm{w}) = \sum_{i=1}^{N} (\sigma(z) - y_i) x_i$$

Since there is no closed form solution for the equation $\nabla E(\mathrm{w}) = 0$, we solve this using gradient descent.

4.1.1 Gradient descent

1. We start with a random initialization for our w vector.
2. Calculate predictions $\sigma(\mathrm{w}^T x)$ for the training data.
3. Now, we update our w vector based on the gradient of loss function according to the equation shown below:

$$\mathrm{w}_{new} = \mathrm{w}_{old} - \eta \times \nabla E(\mathrm{w})$$

4. We go back to step 2 with this updated w and repeat the loop until the specified number of iterations are reached.

4.1.2 Making predictions

After gradient descent we have our revised estimate for w, let us call it $\hat{\mathrm{w}}$. For a new input vector x, the probability that it belongs to class 1 is given by the sigmoid function.

$$y(\text{predicted}) = \begin{cases} 1, & \text{if } \sigma(\hat{\mathrm{w}}^T x) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

4.1.3 Multi-class classification

For the Digits data set where there are multiple classes, we trained 1-vs-all classifiers for each class. An input vector x will be assigned to the class whose sigmoid function gives the highest probability.

$$y(\text{predicted}) = \underset{k}{\operatorname{argmax}}\, \sigma(\hat{\mathrm{w}}_k^T x)$$

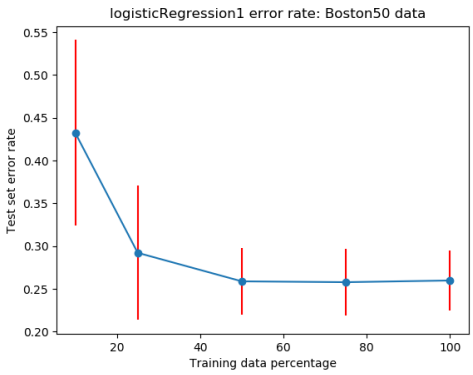
Where $\hat{\mathrm{w}}_k^T$ is the weight estimate for the kth class.

4.1.4 Results

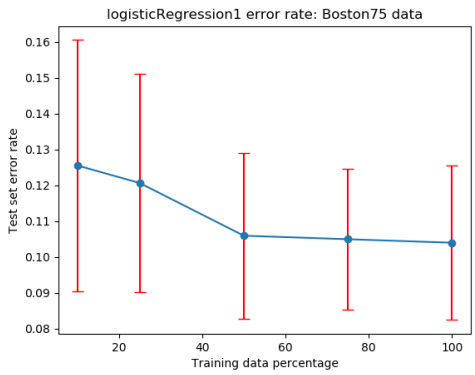
When 100% of the datasets are used the average test set error rates given by logistic regression are shown below.

Dataset (100%)	**Avg. error Rate**
Boston-50	24.12 %
Boston-75	10.39 %
Digits	11.73 %

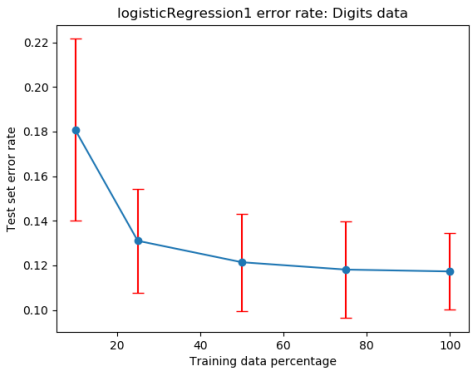
For each dataset we can see the error rate when increasing amount of training data is used in below figures



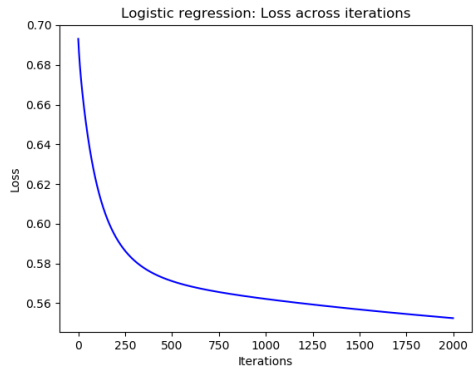
(a) Boston-50 Error rate



(b) Boston-75 Error rate



(c) Digits Error rate



(d) Loss value

Figure 1: Logistic Regression

4.2 Naive Bayes

In Naive Bayes classifier, the posterior probability of a class is given by the Bayes rule. The simplifying assumption made here is that each feature in the input values are independent of each other

$$\begin{aligned} p(\mathrm{C}_k|x) &\propto p(x|\mathrm{C}_k) \times p(\mathrm{C}_k) \\ &\propto [p(x_1|\mathrm{C}_k) \times p(x_2|\mathrm{C}_k) \times ...p(x_d|\mathrm{C}_k)] \times p(\mathrm{C}_k) \\ &\propto \prod_{i=1}^{D} p(x_i|\mathrm{C}_k) \times p(\mathrm{C}_k) \end{aligned}$$

Here the terms x_1 to x_d are the features of input vector. In case of continuous feature values, we assume each of these features to be drawn from an independent underlying Gaussian distribution for each class.

$$p(x_i|\mathrm{C}_k) \sim \mathcal{N}(\mu_{ki}, \sigma_{ki})$$

We split the training dataset based on class labels and calculate the feature means (μ_k) and feature variance(σ_k) for each class. They are nothing but the sample mean and sample variances in that class-specific dataset.

Class priors are calculated based on the percentage of that class in training sample.

4.2.1 Making predictions

We now have the mean and variance for all features for each class. For a new input variable, we calculate the probabilities of each feature for each class label.

$$p(\mathrm{C}_k|x) \propto \prod_{i=1}^{D} \mathcal{N}(x_i, \mu_{ki}, \sigma_{ki}) \times p(\mathrm{C}_k)$$

After calculating all such probabilities, the vector is assigned to the class for which the probability is the highest

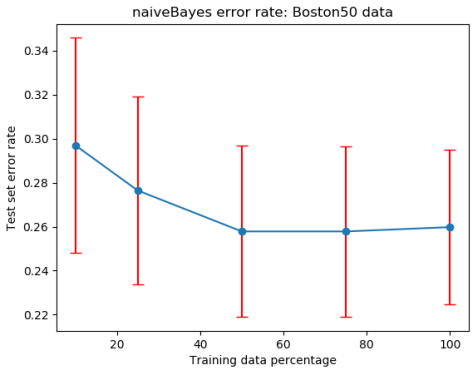
$$y(\text{predicted}) = \underset{k}{\operatorname{argmax}} \left[\prod_{i=1}^{D} \mathcal{N}(x_i, \mu_{ki}, \sigma_{ki}) \times p(\mathrm{C}_k) \right]$$

4.2.2 Results

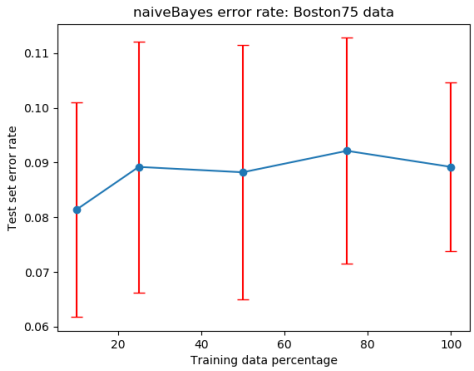
When 100% of the datasets are used the average test set error rates given by Naive Bayes classifier are shown below.

Dataset (100%)	**Avg. error Rate**
Boston-50	25.98 %
Boston-75	8.92 %
Digits	13.52 %

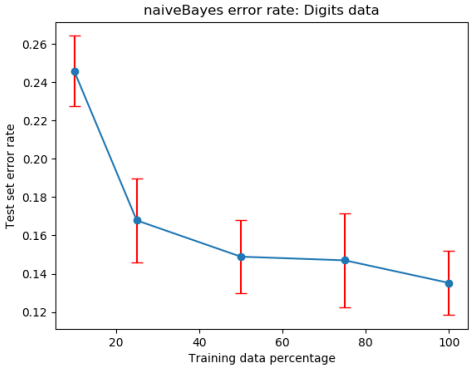
For each dataset we can see the error rate when increasing amount of training data is used in below figures



(a) Boston-50 Error rate



(b) Boston-75 Error rate



(c) Digits Error rate

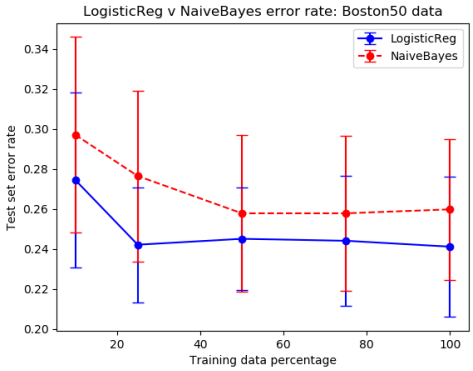
Figure 2: Naive Bayes

4.3 Comparison plots

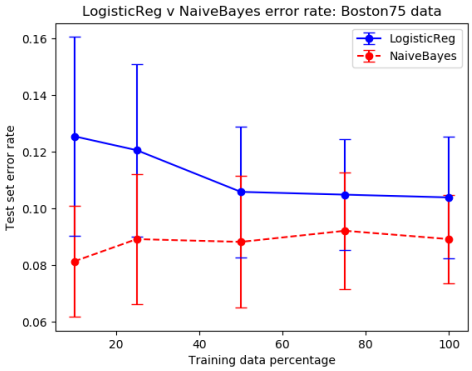
Figure 3 (on the next page) shows comparison plots between Naive Bayes and Logistic Regression.

In case of Boston-75 dataset, the error of asymptotic error of Naive Bayes seems lower and reached faster than Logistic Regression. In other cases we see Logistic Regresion has lower error and they seem to reach their respective asymptotic errors at the same speed.

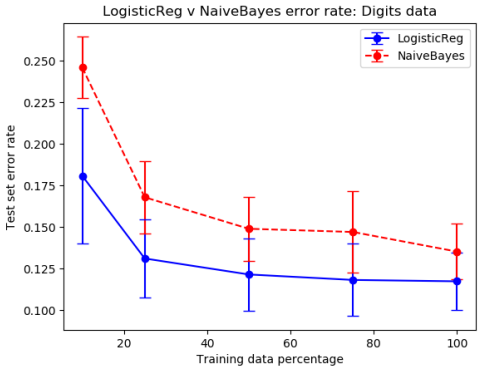
Also the logistic regression error rates were influenced by what parameters are chosen for learning-rate and minibatch size. Which could affect these comparison results.



(a) Boston-50 Error rate



(b) Boston-75 Error rate



(c) Digits Error rate

Figure 3: Naive Bayes vs Logistic Regression