# 1 Bayesian Logistic Regression For Image Classification

**Equation of Interest**

$Y \sim Ber(\frac{1}{1+e^{-X^T\beta}})$

$\beta \sim N(0, \sigma^2 I)$

a Here I solve for the unnormalized posterior of $\beta|Y$.

$$P(\beta|y;x,\sigma) \propto \prod P(y_i|\beta;x_i)P(\beta;\sigma) \text{ formula for posterior}$$

$$P(\beta|y;x,\sigma) \propto \prod (\frac{1}{1+e^{-X^T\beta}})^{y_i}(1-\frac{1}{1+e^{-X^T\beta}})^{1-y_i}(\frac{e^{\frac{-\beta^2}{2\sigma}}}{\sigma\sqrt{2\pi}}) \text{ Bernoulli and Gaussian distributions}$$

b Here I prove that the unnormalized posterior is proportional to $exp(-U(\beta))$ where $U(\beta) =$

$\sum(1-y_i)x_i^T\beta + ln(1+e^{-x_i^T\beta}) + \frac{1}{2\sigma^2}||\beta||^2$

$$ln(P(\beta|y;x,\sigma)) \propto ln(\prod(\frac{1}{1+e^{-X^T\beta}})^{y_i}(1-\frac{1}{1+e^{-X^T\beta}})^{1-y_i}(\frac{e^{\frac{-\beta^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}))$$ take the $ln$ of both sides

$$ln(P(\beta|y;x,\sigma)) \propto \sum ln((\frac{1}{1+e^{-X^T\beta}})^{y_i}(1-\frac{1}{1+e^{-X^T\beta}})^{1-y_i}(\frac{e^{\frac{-\beta^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}))$$ product rule

$$\propto \sum ln((\frac{1}{1+e^{-X^T\beta}})^{y_i}(\frac{e^{-X^T\beta}}{1+e^{-X^T\beta}})^{1-y_i}(\frac{e^{\frac{-\beta^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}))$$ simplification

$$\propto \sum ln(\frac{1}{1+e^{-X^T\beta}})^{y_i} + ln(\frac{e^{-X^T\beta}}{1+e^{-X^T\beta}})^{1-y_i} + ln(\frac{e^{\frac{-\beta^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}})$$ Product Rule

$$\propto \sum y_i ln(1) - y_i ln(1+e^{-X^T\beta}) + (1-y_i)(ln(e^{-X^T\beta}) - ln(1+e^{-X^T\beta})) + ln(e^{\frac{-\beta^2}{2\sigma^2}}) - ln(\sigma\sqrt{2\pi})$$

Power Rule and Quotient Rule

$$\propto \sum -y_i ln(1+e^{-X^T\beta}) + (1-y_i)(-(X^T\beta) - ln(1+e^{-X^T\beta})) - \frac{\beta^2}{2\sigma^2}$$

Simplification and removing constants

$$\propto \sum -y_i ln(1+e^{-X^T\beta}) - (1-y_i)(X^T\beta) + (1-y_i)ln(1+e^{-X^T\beta}) - \frac{\beta^2}{2\sigma^2}$$ Distribution

$$\propto \sum -y_i ln(1+e^{-X^T\beta}) - (1-y_i)(X^T\beta) - ln(1+e^{-X^T\beta}) + y_i ln(1+e^{-X^T\beta}) - \frac{\beta^2}{2\sigma^2}$$ Distribution

$$\propto \sum -(1-y_i)(X^T\beta) - ln(1+e^{-X^T\beta}) - \frac{\beta^2}{2\sigma^2}$$ Simplification

$$-ln(P(\beta|y;x,\sigma)) \propto (1-y_i)(X^T\beta) + ln(1+e^{-X^T\beta}) + \frac{\beta^2}{2\sigma^2}$$ Multiply both sides with $-1$

The task of this part of the project is to simply classify digits 0/1 and 6/8 from MNIST using Bayesian Logistic Regression. Since gradient descent is used as the optimization approach, finding the gradient of the exponent, negative of the equation from part b proved important. The gradient was calculated to be $\frac{\partial exp(-U(\beta))}{\partial\beta_k} = \sum_{i=0}^{n}(((1-y_i) + \frac{e^{-X_i\beta}}{1+e^{-x_i\beta}})x_{ik}) + \frac{\beta_k}{\sigma^2}$. A vectorized version of the gradient descent was calculated for computation time interests. The stopping condition for the gradient descent algorithm would be whenever the norm of all the gradients reaches below a certain threshold. The average error rate was calculated using the zero-one loss, $E[|y-\tilde{y}|]$.

The values of 1 and .1 were used for $\sigma$ and the values of .0001, .001, and .01 were used for the learning rate. Ultimately, $\sigma = 1$ and the learning rate = .0001 were used for the 0/1 classification task. The average error rate for the 0/1 classification task was 0.0. The error was so low most likely due to the high number iterations since the gradient descent algorithm is dependent on a small threshold as opposed to fixed number of iterations. For the 6/8 classification task, $\sigma = 1$ and the learning rate = .001 were used and the label of 6 was treated as 0 and 8 treated as 1. The error rate for the 6/8 classification was 0.018. The low error rate can be attributed to the gradient descent stopping condition with a small threshold. Figures 1 and 2 display the convergence graph for the 0/1 and 6/8 convergence graph, respectively.

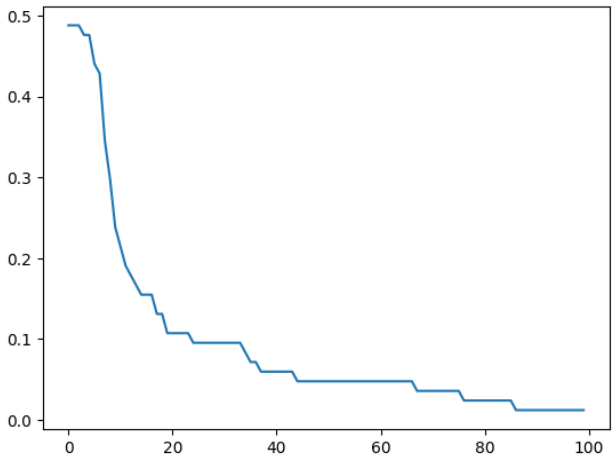


Figure 1: Convergence Graph for 0/1 classification

As shown above, the convergence for the 0/1 classification task yields a choppy but decreasing loss.

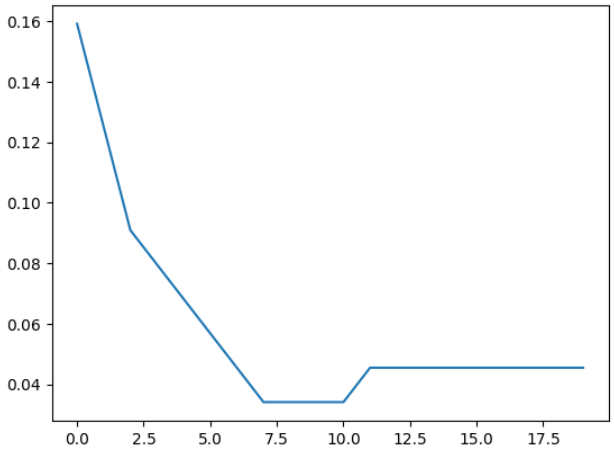


Figure 2: Convergence Graph for 6/8 classification

As shown above, the convergence for the 6/8 classification task yields a choppy but decreasing loss.

## 2 PCA

This particular task aimed to use PCA for 10, 20, and 30 principal components for the 0/1 classification task to test its accuracy and speed. For 10 principal components, the accuracy rate was .99 and took about 1.35 seconds. The 20 principal components yield an accuracy rate of about .99 and a time consumption of about 1.93 seconds. The 30 principal component approach yielded an accuracy rate of about .99 and a time consumption of 3.34 seconds. The results were interesting because all number of principal components yielded the same accuracy but different times. The reason is due to the stopping condition of the gradient descent algorithm. For larger number of

principal components, more time is needed to reach the threshold due to the dimensionality of the data.

## 3 Image Augmentation

The final task was to use image augmentation to synthesize more data for the model. In this particular, I apply a random rotation $\theta \in [-15°, 15°]$ and a random translation in both the x and y direction $T \in [-5, 5]$. The task was run for the 6/8 classification for 3 times and 5 times of the training data for the augmented data. The accuracy for 3 times yielded an accuracy rate of .98 and the 5 times yield an accuracy rate of .96. This is interesting because it seems as though that augmentation has negatively impacted the accuracy. The reasoning for such results could be the introduction of artifacts during the augmentation process.

I learned a lot from this project particularly applying bayesian logistic regression to a binary classification problem. One major flaw with my approach is that it can take a bit of time to reach the threshold for the gradient. This would make the algorithm less scalable due to computational resources.

## 4 Sources


https://numpy.org/doc/

https://matplotlib.org/

scikit-learn.org

https://stackoverflow.com/questions/53171057/numpy-matrix-rotation-for-any-degrees

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.copy.html