Jack Stevenson
Dr. Wong
CS 434H

1. Deriving the posterior distribution for a Beta-Bernoulli conjugate prior:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int P(D|\theta)P(\theta)d\theta}$$

   Let B represent the following:

$$B(\beta_H, \beta_T) = \frac{\Gamma(\beta_H)\Gamma(\beta_T)}{\Gamma(\beta_H + \beta_T)}$$

$$P(\theta|D) = \frac{[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\frac{1}{B(\beta_H,\beta_T)}\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]}{\int[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\frac{1}{B(\beta_H,\beta_T)}\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]d\theta}$$

$$P(\theta|D) = \frac{\frac{1}{B(\beta_H,\beta_T)}[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]}{\frac{1}{B(\beta_H,\beta_T)}\int[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]d\theta}$$

$$P(\theta|D) = \frac{[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]}{\int[\Pi_{i=1}^{N}\theta^{x_i}(1-\theta)^{1-x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]d\theta}$$

$$P(\theta|D) = \frac{[\theta^{\sum x_i}(1-\theta)^{N-\sum x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]}{\int[\theta^{\sum x_i}(1-\theta)^{N-\sum x_i}][\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}]d\theta}$$

$$P(\theta|D) = \frac{[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]}{\int[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]d\theta}$$

   We know the following expression is equal to 1, as it is a PDF:

$$\frac{1}{B(\beta_H+\sum x_i, \beta_T+N-\sum x_i)}\int[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]d\theta = 1$$

   These expressions follow:

$$\frac{1}{B(\beta_H+\sum x_i, \beta_T+N-\sum x_i)} \times B(\beta_H+\sum x_i, \beta_T+N-\sum x_i) = 1$$

$$\int[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]d\theta = B(\beta_H+\sum x_i, \beta_T+N-\sum x_i)$$

   Then we can establish the following:

$$P(\theta|D) = \frac{[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]}{B(\beta_H+\sum x_i, \beta_T+N-\sum x_i)}$$

$$P(\theta|D) = \frac{1}{B(\beta_H+\sum x_i, \beta_T+N-\sum x_i)}[\theta^{\beta_H+\sum x_i-1}(1-\theta)^{\beta_T+N-\sum x_i-1}]$$

$$P(\theta|D) = Beta(\beta_H+\sum_{i=1}^{N} x_i, \beta_T+N-\sum_{i=1}^{N} x_i)$$

This establishes that $P(\theta|D)$ follows a Beta distribution. This means that the posterior distribution for a Beta-Bernoulli conjugate prior is also a Beta distribution.

2. Recall and precision:
$Recall = \frac{\#TruePositives}{\#TruePositives+\#FalseNegatives}$
(Of people who had COVID, how many had accurate test results?)
(Did our COVID tests catch a lot of people who had COVID?)
$Precision = \frac{\#TruePositives}{\#TruePositives+\#FalsePositives}$
(Of positive COVID tests, how many represented real infections?)
(If you test positive, is there a high chance you have COVID?)

Threshold of $t = 0$: Since every point has $P(y|x) > 0$, all points will be predicted to be within the positive class. This means that we have 8 true positives, 8 false positives, 0 true negatives, and 0 false negatives.
$Recall = \frac{8}{8+0} = 1$
$Precision = \frac{8}{8+8} = 0.5$

Threshold of $t = 0.2$: 8 true positives, 6 false positives, 2 true negatives, 0 false negatives.
$Recall = \frac{8}{8+0} = 1$
$Precision = \frac{8}{8+6} = 0.57$

Threshold of $t = 0.4$: 6 true positives, 3 false positives, 5 true negatives, 2 false negatives.
$Recall = \frac{6}{6+2} = 0.75$
$Precision = \frac{6}{6+3} = 0.67$

Threshold of $t = 0.6$: 6 true positives, 1 false positive, 7 true negatives, 1 false negative.
$Recall = \frac{6}{6+1} = 0.86$
$Precision = \frac{6}{6+1} = 0.86$

Threshold of $t = 0.8$: 4 true positives, 1 false positive, 7 true negatives, 4 false negatives.
$Recall = \frac{4}{4+4} = 0.5$
$Precision = \frac{4}{4+5} = 0.44$

Threshold of $t = 1$: 0 true positives, 0 false positives, 8 true negatives, 8 false negatives.
$Recall = \frac{0}{0+8} = 0$
$Precision = \frac{0}{0+0} = N/A$

3. Sum squared error:
If we have a case where the partial derivative is equal to 0 and the 2nd partial derivative is positive, we have found a (local) minimum. Thus, when $w*$ and $b*$ correspond to values that produce a minimum SSE:

$$\frac{\partial SSE}{\partial b} = 0 = -\sum_{i=1}^{N} 2(y_i - (w*)x_i - b*)$$

$$0 = \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} (w*)x_i - \sum_{i=1}^{N} b*$$

$$\sum_{i=1}^{N} b* = \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} (w*)x_i$$

$$b * \sum_{i=1}^{N} 1 = \sum_{i=1}^{N} y_i - w * \sum_{i=1}^{N} x_i$$

$$Nb* = \sum_{i=1}^{N} y_i - w * \sum_{i=1}^{N} x_i$$

$$b* = \frac{1}{N}\sum_{i=1}^{N} y_i - w * \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$b* = \bar{y} - (w*)\bar{x}$$

Since $\frac{\partial^2 SSE}{\partial b^2} = 2N$, we know that it will always be positive and this expression for $b*$ corresponds to a minimum SSE. Next:

$$\frac{\partial SSE}{\partial w} = 0 = -\sum_{i=1}^{N} 2(y_i - (w*)x_i - (b*))x_i$$

$$0 = -\sum_{i=1}^{N} 2(y_i x_i - (w*)x_i^2 - (b*)x_i)$$

$$0 = -2\sum_{i=1}^{N} (y_i x_i - (w*)x_i^2 - (b*)x_i)$$

$$0 = \sum_{i=1}^{N} (y_i x_i - (w*)x_i^2 - (b*)x_i)$$

$$0 = \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} (w*)x_i^2 - \sum_{i=1}^{N} (b*)x_i$$

$$0 = \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} (w*) x_i^2 - \sum_{i=1}^{N} (\bar{y} - (w*)\bar{x}) x_i$$

$$0 = \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} (w*) x_i^2 - \sum_{i=1}^{N} \bar{y} x_i + \sum_{i=0}^{N} (w*)\bar{x} x_i$$

$$0 = \sum_{i=1}^{N} y_i x_i - w * \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} \bar{y} x_i + w * \sum_{i=0}^{N} \bar{x} x_i$$

$$w * \sum_{i=1}^{N} x_i^2 - w * \sum_{i=0}^{N} \bar{x} x_i = \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} \bar{y} x_i$$

$$w * (\sum_{i=1}^{N} x_i^2 - \sum_{i=0}^{N} \bar{x} x_i) = \sum_{i=1}^{N} y_i x_i - \sum_{i=1}^{N} \bar{y} x_i$$

$$w * (\frac{1}{N} \sum_{i=1}^{N} x_i^2 - \frac{1}{N} \sum_{i=0}^{N} \bar{x} x_i) = \frac{1}{N} \sum_{i=1}^{N} y_i x_i - \frac{1}{N} \sum_{i=1}^{N} \bar{y} x_i$$

$$w * (\bar{x}^2 - \frac{1}{N} \sum_{i=0}^{N} \bar{x} x_i) = \bar{xy} - \frac{1}{N} \sum_{i=1}^{N} \bar{y} x_i$$

We can use the Cauchy-Schwartz Inequality to evaluate nested summations:

$$w * (\bar{x}^2 - \frac{1}{N} \sum_{i=0}^{N} (\sum_{i=0}^{N} x_i) x_i) = \bar{xy} - \frac{1}{N} \sum_{i=1}^{N} (\sum_{i=1}^{N} y_i) x_i$$

$$w * (\bar{x}^2 - (\bar{x})^2) = \bar{xy} - \bar{y}\bar{x}$$

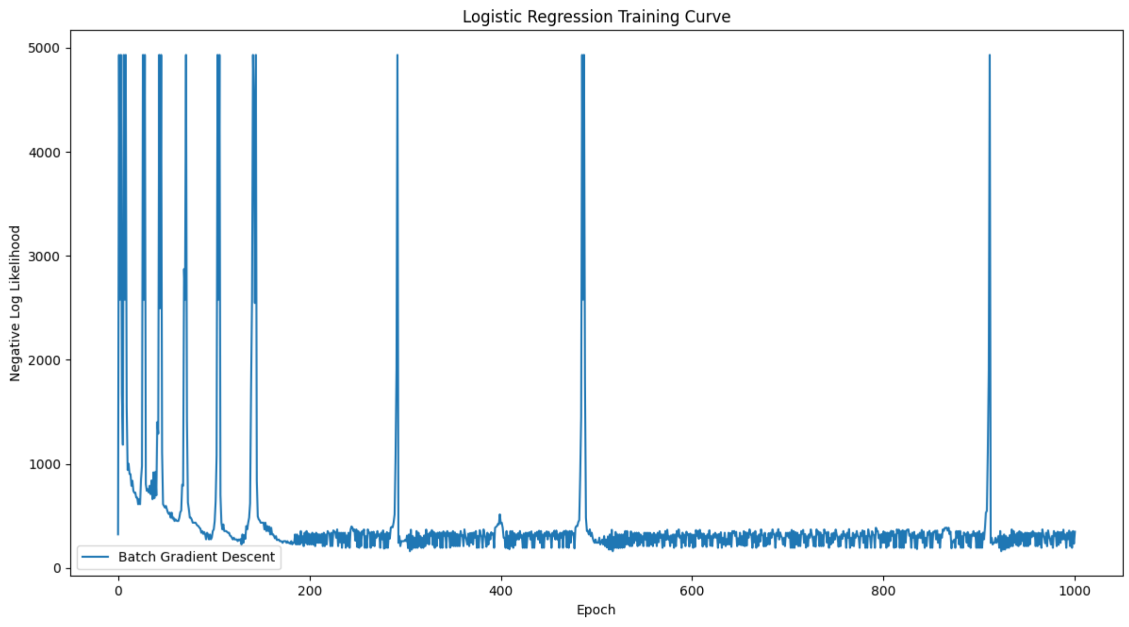
$$w* = \frac{\bar{xy} - \bar{y}\bar{x}}{\bar{x}^2 - (\bar{x})^2}$$

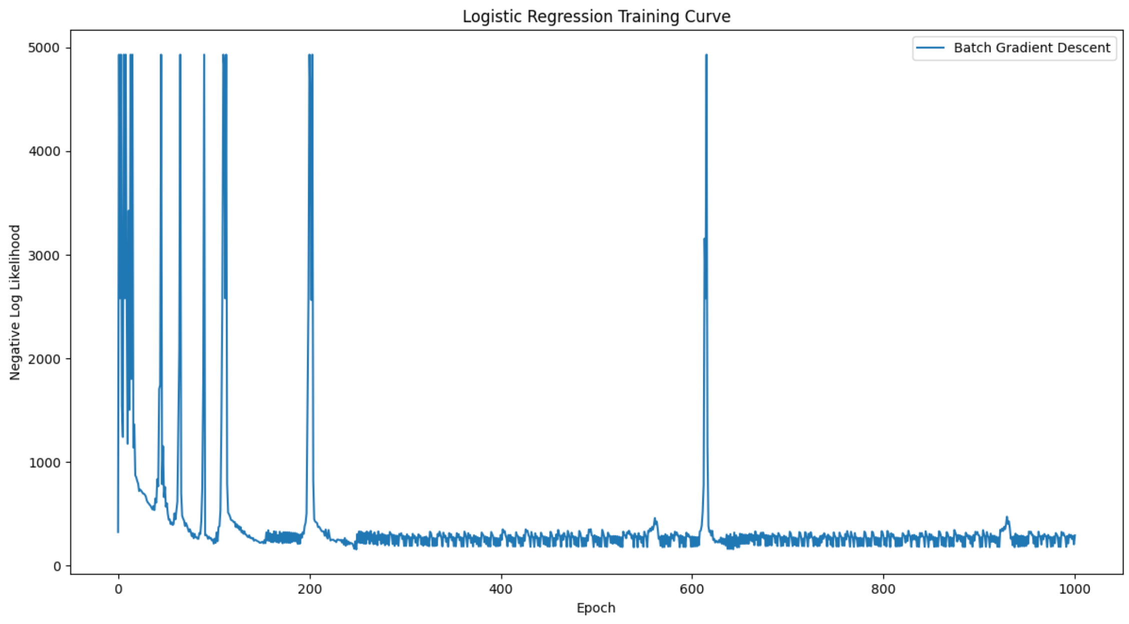
Since $\frac{\partial^2 SSE}{\partial w^2} = 2 \sum_{i=1}^{N} x_i^2$, we know that it will always be positive and this expression for $w*$ corresponds to a minimum SSE. We now know that $b*$ and $w*$ are values corresponding to the minimum SSE.

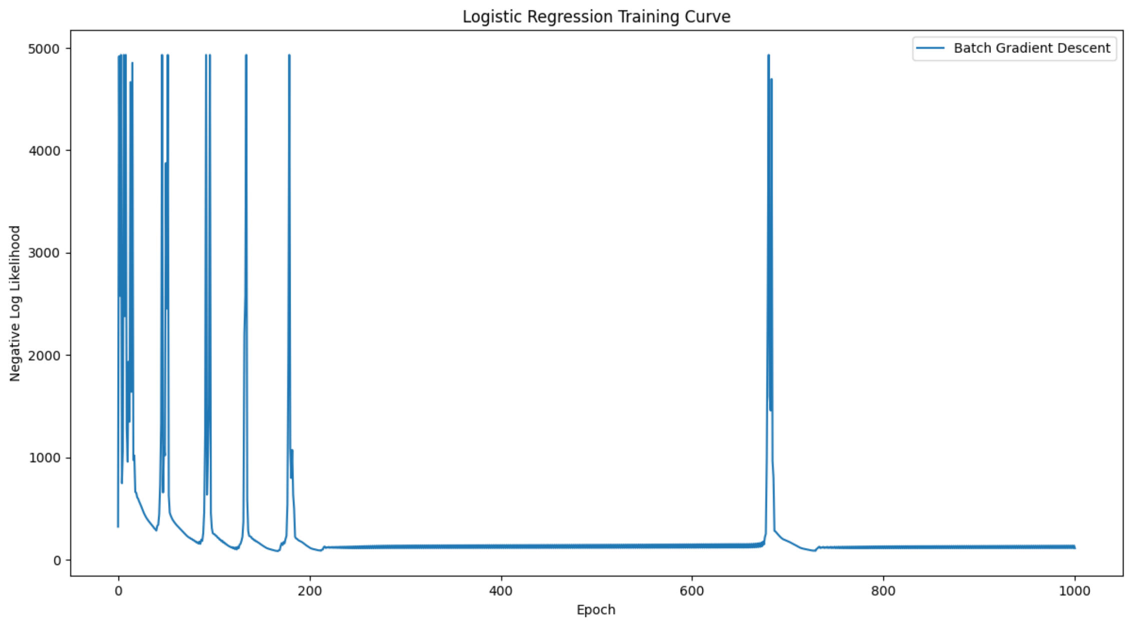
Step sizes:

As step size decreases, there is little effect on accuracy but the amount of noise is greatly reduced. Training accuracy was always around 96-97% for step sizes of 1, 0.1, 0.01, and 0.0001. When it was decreased to 0.00001, the accuracy dropped off to around 90%, as there were not enough iterations to converge. For batch logistic regression, the provided value of 0.0001 seemed the best in terms of low noise and accuracy.

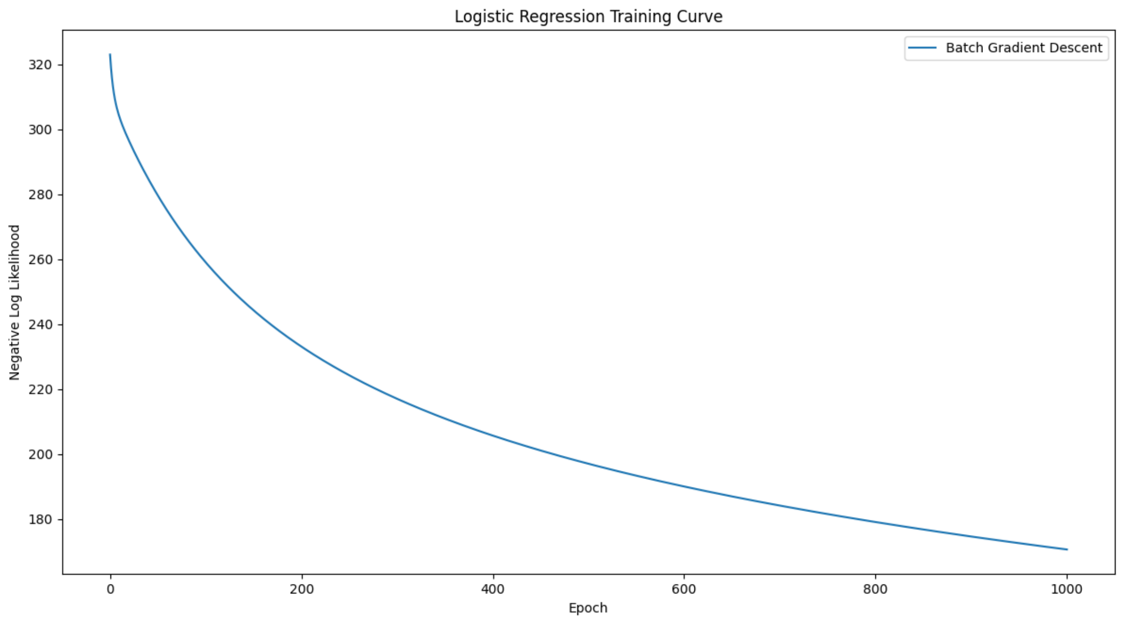
Figures depict step sizes of 1, 0.1, 0.01, 0.0001, and 0.00001, respectively.


Logistic Regression Training Curve
Negative Log Likelihood
Epoch
Batch Gradient Descent
0
1000
2000
3000
4000
5000
0
200
400
600
800
1000

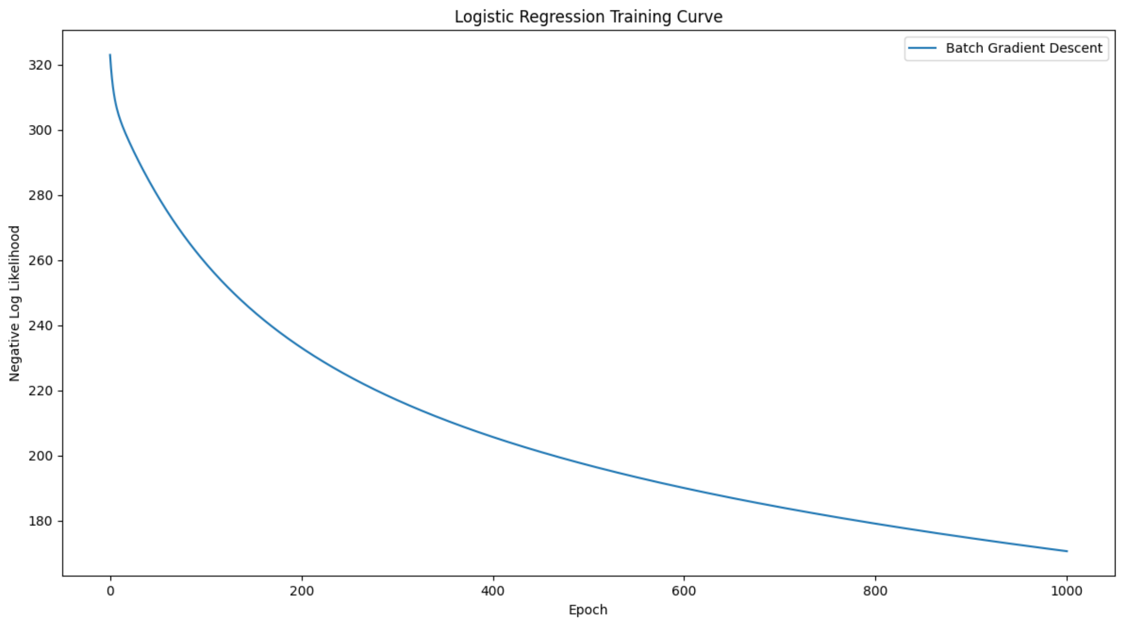

Logistic Regression Training Curve
Negative Log Likelihood
Epoch
Batch Gradient Descent
0
1000
2000
3000
4000
5000
0
200
400
600
800
1000



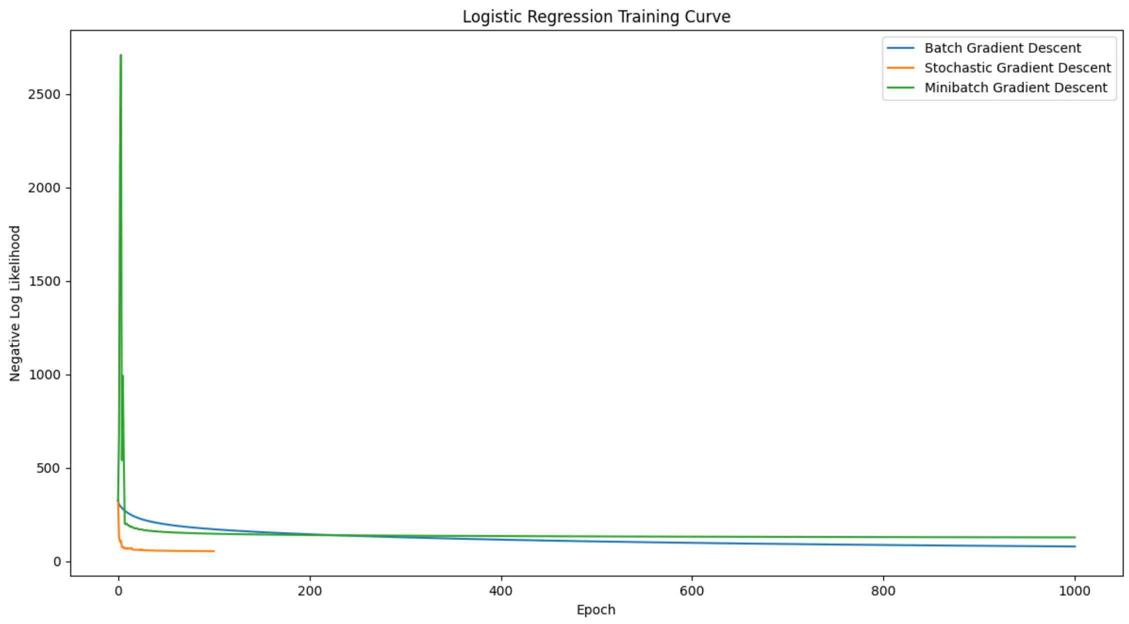
Logistic Regression Training Curve
Batch Gradient Descent
Negative Log Likelihood
Epoch
0
1000
2000
3000
4000
5000
0
200
400
600
800
1000



Logistic Regression Training Curve
Batch Gradient Descent
Negative Log Likelihood
Epoch
180
200
220
240
260
280
300
320
0
200
400
600
800
1000



Batch logistic regression provided a decently fast and noise-free result. I used the matrix implementation of the equation, and I found that it was very simple to implement. Stochastic was by far the fastest and didn't have much of a drop in accuracy. It had more noise than standard logistic regression, but it still clearly converged. Minibatch had several wild spikes at the beginning but ended up converging regardless. It was slightly faster than batch logistic regression. I noticed that large step sizes contributed to quicker accuracy but did not always converge. Other times, they led the function to grow wildly instead of decreasing. I found that the learning rate for stochastic was around 500 times what was ideal for batch, and the rate for minibatch was around 50 times the standard rate. I had trouble getting minibatch to decrease while maintaining high accuracy as it spiked higher with larger learning rates. I found that batch required more iterations than the others, and that stochastic required far fewer than either of the others. Batch had an accuracy of 96.5%,, stochastic had 97%, and minibatch had around 95%.



Debriefing

1. Approximately how many hours did you spend on this assignment?
   I spent around 12 hours on this assignment.
2. Would you rate it as easy, moderate, or dificult?
   I found the math to be somewhat difficult, especially the first question, but the programming was relatively straightforward. I found the assignment significantly easier than the last one.
3. Did you work on it mostly alone or did you discuss the problems with others?
   I worked on it mostly alone. I went to office hours, discussed the first problem with a classmate, and asked Dr. Lee to check the graphs and accuracies produced by the code.
4. How deeply do you feel you understand the material it covers (0%-100%)?
   I understand this material pretty well. I think I need more practice deriving and understanding expressions, but I generally feel comfortable with the concepts.
5. Any other comments?
   There is sometimes a warning for overflow when very large step sizes in standard logistic regression (1, 0.1) but it doesn't cause the program to crash. It doesn't seem to affect any of the results. Also, I was unsure whether or not we were supposed to analyze the test data. I added a few lines process the test set and save as CSV (45:51).