CSE353 Fall 2020 - Machine Learning - Homework 2

Steven Fong

112140432

steven.fong@stonybrook.edu

N/A

Question 1

Simplifying log likelihood:

$$log(P(Y^i|\bar{X}^i;\boldsymbol{\theta})) = log(\prod_{a=0}^{k-1} P(a|\bar{X}^i;\boldsymbol{\theta})^{\delta(a=Y^i)})$$
$$= \sum_{a=0}^{k-1} log(P(a|\bar{X}^i;\boldsymbol{\theta})^{\delta(a=Y^i)})$$
$$= \sum_{a=0}^{k-1} \delta(a=Y^i) log(P(a|\bar{X}^i;\boldsymbol{\theta}))$$
$$= \sum_{a=0}^{k-1} \delta(a=Y^i) log(\frac{exp(\boldsymbol{\theta}_a^T \bar{X}^i)}{1+\sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)})$$
$$= \sum_{a=0}^{k-1} \delta(a=Y^i)(\boldsymbol{\theta}_a^T \bar{X}^i - log(1+\sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)))$$

Notice that Y^i must only be one class in the range of a (a is iterating through all classes). Since it can only take on one class the summation is really just one term added with a lot of zeroes. If $Y^i = c$, the summation simplifies to, $\boldsymbol{\theta}_c^T \bar{X}^i - log(1+\sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i))$, if not, and $Y^i = a_m$ the summation simplifies to $\boldsymbol{\theta}_{a_m}^T \bar{X}^i - log(1+\sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i))$.

Putting this together we get that the log likelihood is:

$$
\begin{aligned}
&= \delta(c = Y^i)(\boldsymbol{\theta}_c^T \bar{X}^i - log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i))) \\
&\quad + (1 - \delta(c = Y^i))(\boldsymbol{\theta}_{a_m}^T \bar{X}^i - log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i))) \\
&= \delta(c = Y^i)\boldsymbol{\theta}_c^T \bar{X}^i - \delta(c = Y^i) log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&\quad - \delta(c = Y^i)\boldsymbol{\theta}_{a_m}^T \bar{X}^i + \delta(c = Y^i) log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&\quad + \boldsymbol{\theta}_{a_m}^T \bar{X}^i - log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&= \delta(c = Y^i)\boldsymbol{\theta}_c^T \bar{X}^i - \delta(c = Y^i) log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&\quad - \delta(c = Y^i)\boldsymbol{\theta}_{a_m}^T \bar{X}^i + \delta(c = Y^i) log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&\quad + \boldsymbol{\theta}_{a_m}^T \bar{X}^i - log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&= \delta(c = Y^i)\boldsymbol{\theta}_c^T \bar{X}^i - log(1 + \sum_{j=1}^{k-1} exp(\boldsymbol{\theta}_j^T \bar{X}^i)) \\
&\quad - \delta(c = Y^i)\boldsymbol{\theta}_{a_m}^T \bar{X}^i + \boldsymbol{\theta}_{a_m}^T \bar{X}^i
\end{aligned}
$$

Notice that the last two terms are constants, when we take partial derivatives with respect to $\boldsymbol{\theta}_c$ thus we can ignore them and focus on deriving the first two terms.

Now we take derivative:

$$\frac{\partial(log(P(Y^i|\bar{X}^i;\boldsymbol{\theta})))}{\partial(\boldsymbol{\theta}_c)} = \frac{\partial(\delta(c=Y^i)\boldsymbol{\theta}_c^T\bar{X}^i - log(1+\sum_{j=1}^{k-1}exp(\boldsymbol{\theta}_j^T\bar{X}^i)))}{\partial(\boldsymbol{\theta}_c)}$$

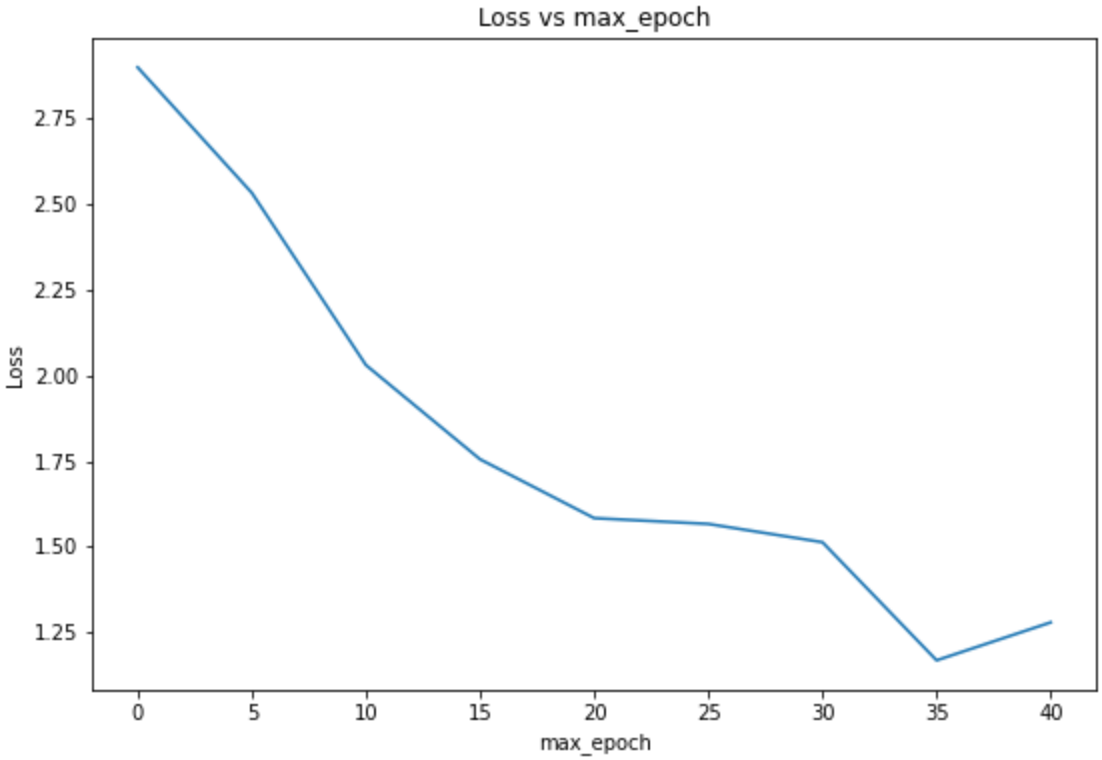
$$= \delta(c=Y^i)\bar{X}^i - \frac{exp(\boldsymbol{\theta}_c^T\bar{X}^i)\bar{X}^i}{1+\sum_{j=1}^{k-1}exp(\boldsymbol{\theta}_j^T\bar{X}^i)}$$

$$= (\delta(c=Y^i) - \frac{exp(\boldsymbol{\theta}_c^T\bar{X}^i)}{1+\sum_{j=1}^{k-1}exp(\boldsymbol{\theta}_j^T\bar{X}^i)})\bar{X}^i$$

$$= (\delta(c=Y^i) - P(c|\bar{X}^i;\boldsymbol{\theta}))\bar{X}^i$$

Question 3.1

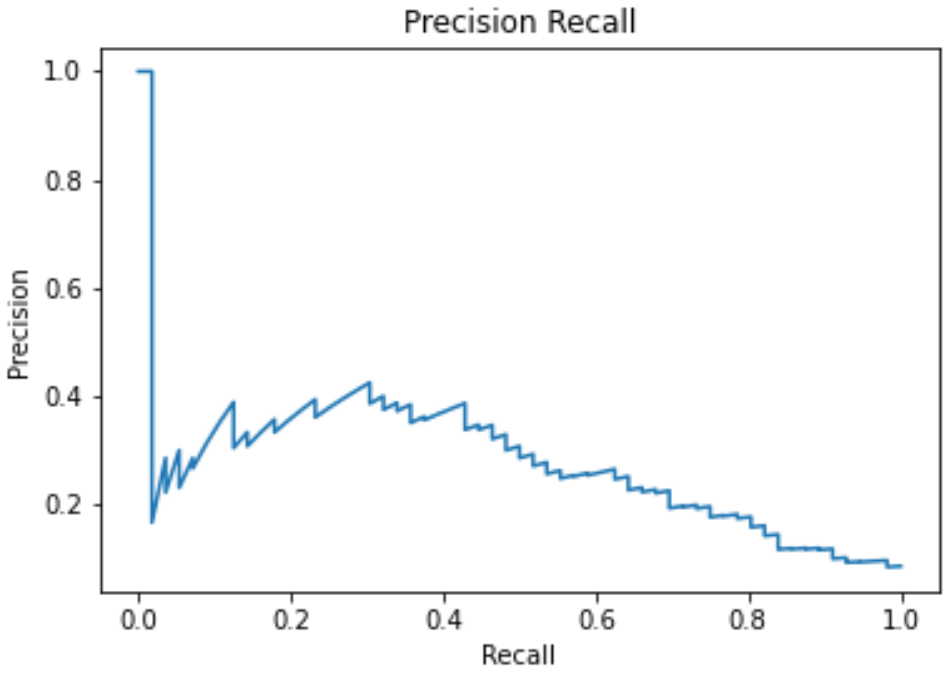
A) As the max_epoch increases the loss decreases and approaches zero.



B) The accuracy as calculated by the accuracy_score function defined by sklearn library is abou 90-91 percent. The accuracy as calculated using the confusion matrix is equal to the ones calculated by the sklearn library.
Train Accuracy(sklearn): 0.941027496382055
Test Accuracy(sklearn): 0.9044862518089725

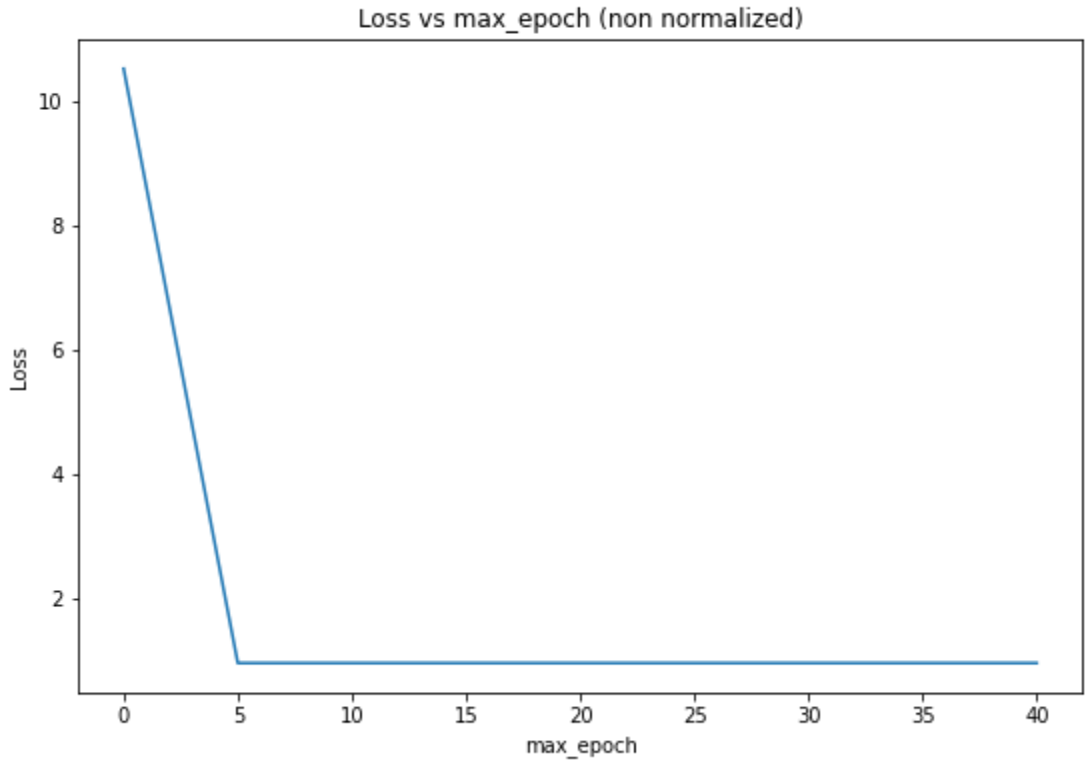
Train Confusion matrix: [[2487 42]
[121 114]]
Test Confusion matrix: [[608 27]
[39 17]]
Computed Train Accuracy(matrix): 0.941027496382055
Computed Test Accuracy(matrix): 0.9044862518089725

C) Precision: 0.38636363636363635



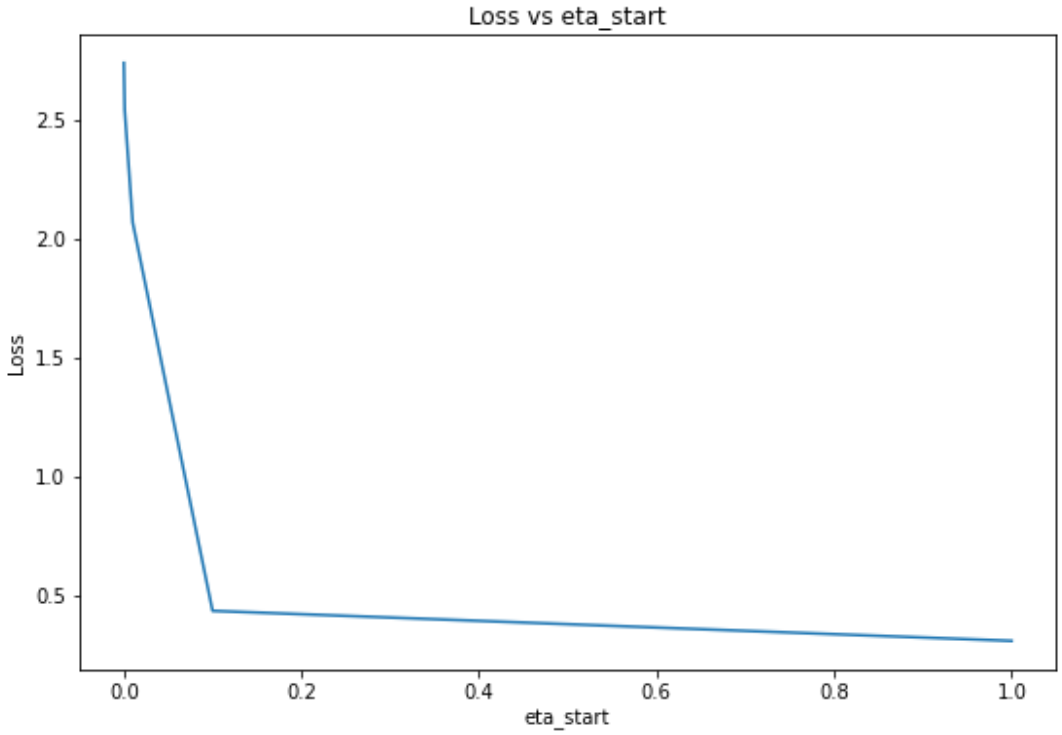
Question 3.2

3.2a

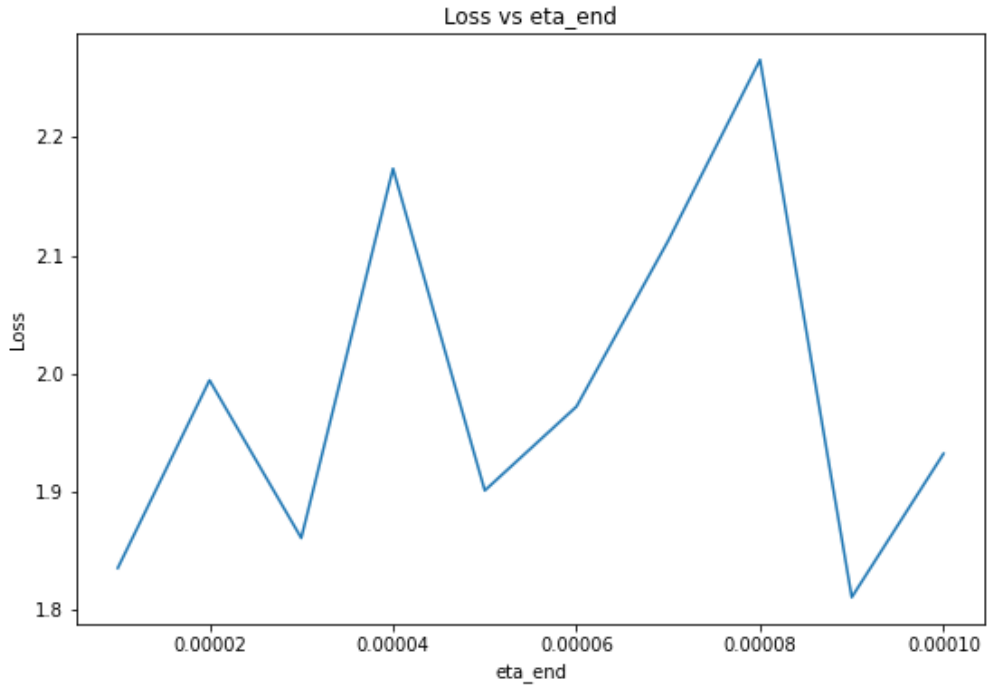


The graph shows that even with only 5 full passes of the training data, the loss converges to zero very quickly. If we check the confusion matrix, we see that the model predicted all people to live. The performance is within 90 to 91 percent accurate. This is because the given training data does not have a 50-50 split of people who died (1) and people who lived(0), thus simply by saying that the people lived, the model will have a high accuracy score. The precision or recall would be more important in this case because we care more about predicting who will die. In this case, the recall is zero.

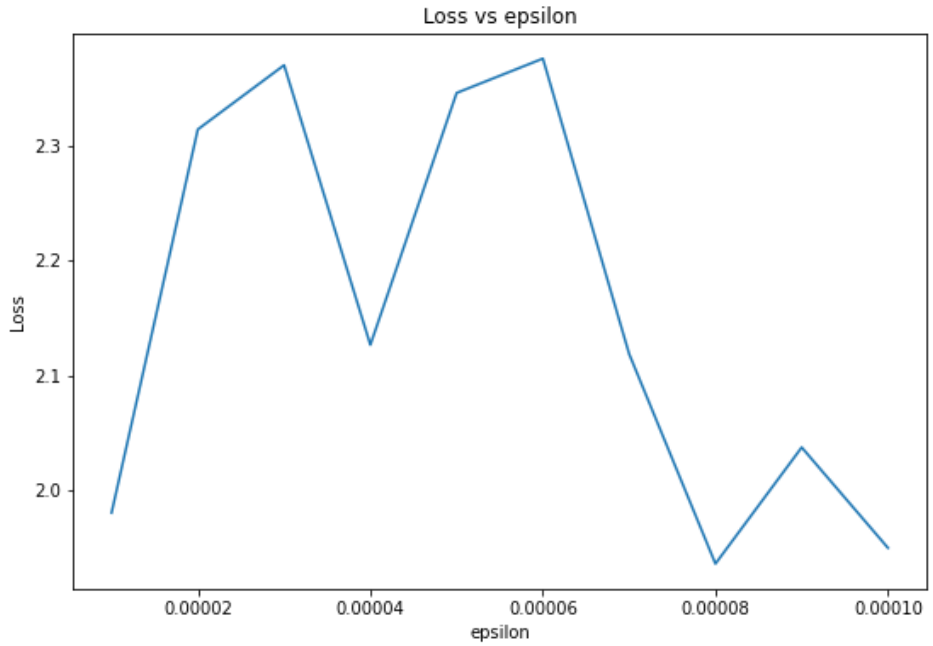
3.2b



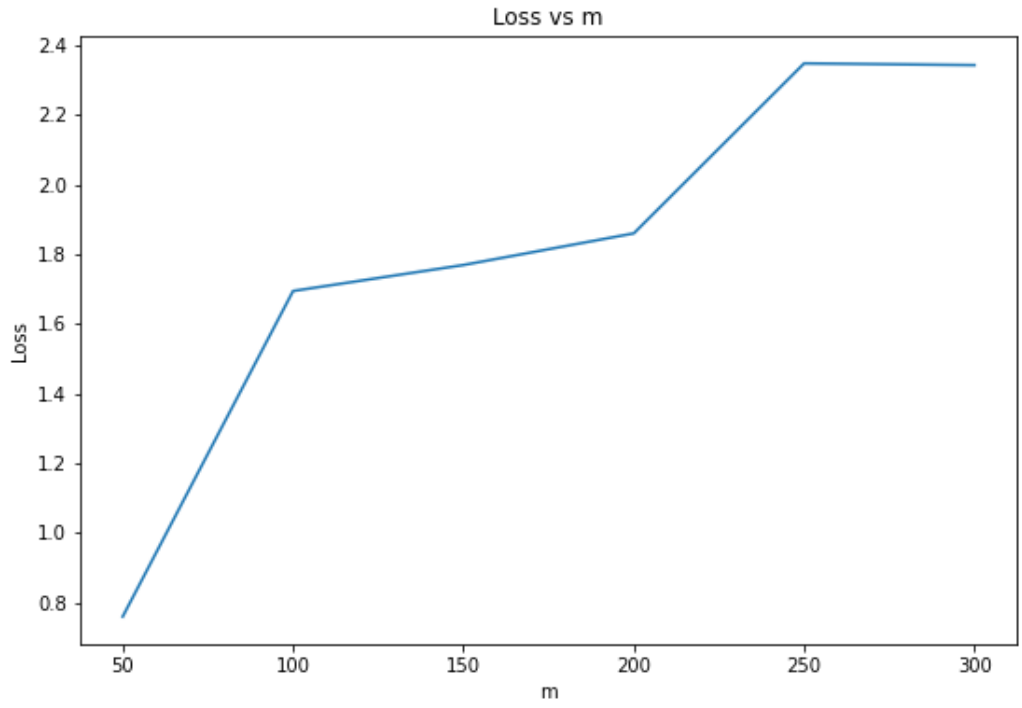
As eta_start increases the loss decreases going approaching zero. We will choose a low value of est_start, like 1.



On multiple runs creating this graph, the lowest loss always changes, thus I think that in general a low value for loss would be good enough so long as it is not extremely big. On multiple runs most of the lowest points were around the original value specified, so we will choose 0.00009.

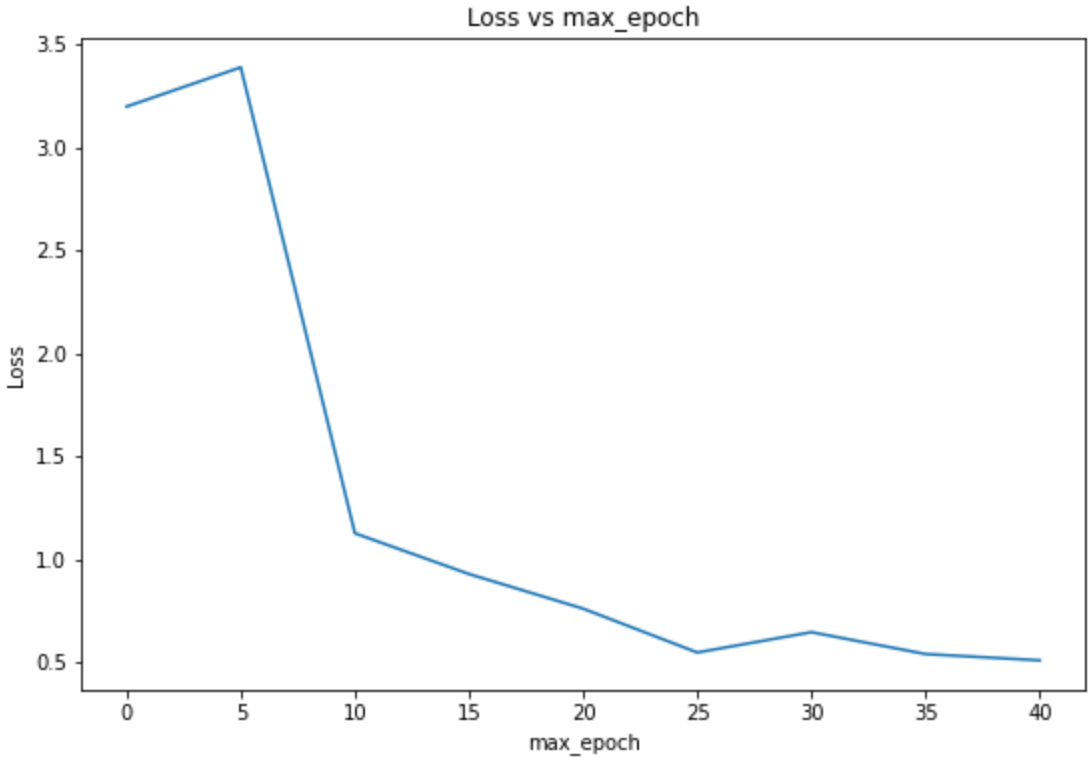


On multiple runs, this graph also changes drastically. On most runs however, it was also around the original value for epsilon so we will choose an epsilon of 0.00008.



The loss tends to increase as m increases so we can choose an m of 50. Any lower might be too small
From the graphs, as well as from testing, the max_epoch and m seems to have a big effect on the final performance. If max_epoch is too small or m is too small, the precision and accuracy drops drastically. This is also true for eta_start, if it is too small or if it is close to eta_end, the precision and accuracy also drops severely. For example with eta as 0.001 and eta_end about the same, the accuracy and precision drops 30 percent from the default inputs.

3.2c



The values were chosen based on the graphs produced in part b of the question. I chose the data values based on the lowest loss values shown in the graph. That said for the eta_end and epsilon graphs, the results were for the most part inconsistent, but in general it returned a lower loss value around the default values given in the beginning of question 3. As we can see in the graph here, as compared to the first one we ran with all the default values, this graph converged faster as the max_epoch increases and, on average, has a lower loss. As a final test, I ran the fit at smaller max_epoch values like 40 and 25 and they tend to perform better in terms of precision than max_epoch = 1000 or 100. Either way, the highest precision I could get was around 40 percent, which is probably because the training data contains more than 90% of class 0.