- $$L(\theta|D) = P(D|\theta) = \prod_{i=1}^{m} p(x^{(i)}|\theta) = \prod_{i=1}^{m} \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}$$

$$P(D|\theta) = \theta^{\sum_{i=1}^{m} x^{(i)}}(1-\theta)^{m-\sum_{i=1}^{m} x^{(i)}}$$

$$\log(L(\theta|D)) = \log(\theta)\sum_{i=1}^{m} x^{(i)} + \log(1-\theta)(m - \sum_{i=1}^{m} x^{(i)})$$

$$NLL(\theta) = -\log(\theta)\sum_{i=1}^{m} x^{(i)} - \log(1-\theta)(m - \sum_{i=1}^{m} x^{(i)})$$

Also, remember that $\theta_{MLE} = argmin_\theta NLL(\theta)$. That means we need to take the derivative of $NLL(\theta)$ and set it equal to 0

$$\frac{\partial}{\partial\theta} NLL(\theta) = \frac{\partial}{\partial\theta}\Big(-\log(\theta)\sum_{i=1}^{m} x^{(i)} - \log(1-\theta)(m - \sum_{i=1}^{m} x^{(i)})\Big)$$

$$\frac{\partial}{\partial\theta} NLL(\theta) = -\frac{\partial}{\partial\theta}\Big(\log(\theta)\Big)\sum_{i=1}^{m} x^{(i)} - \frac{\partial}{\partial\theta}\Big(\log(1-\theta)\Big)(m - \sum_{i=1}^{m} x^{(i)})$$

$$\frac{\partial}{\partial\theta} NLL(\theta) = -\frac{1}{\theta}\sum_{i=1}^{m} x^{(i)} + \frac{1}{1-\theta}(m - \sum_{i=1}^{m} x^{(i)}) = 0$$

$$(1-\theta_{MLE})\sum_{i=1}^{m} x^{(i)} = \theta_{MLE}(m - \sum_{i=1}^{m} x^{(i)})$$

$$\sum_{i=1}^{m} x^{(i)} = m\theta_{MLE}$$

$$\theta_{MLE} = \frac{\sum_{i=1}^{m} x^{(i)}}{m}$$

- $$L(\theta|D)Beta(\theta|a,b) = P(D|\theta)P(\theta) = \theta^{a-1}(1-\theta)^{b-1}\prod_{i=1}^{m} \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}$$

$$P(D|\theta)P(\theta) = \theta^{a-1+\sum_{i=1}^{m} x^{(i)}}(1-\theta)^{b-1+m-\sum_{i=1}^{m} x^{(i)}}$$

$$\log(P(D|\theta)P(\theta)) = \log(\theta)(a-1+\sum_{i=1}^{m} x^{(i)}) + \log(1-\theta)(b-1+m-\sum_{i=1}^{m} x^{(i)})$$

$$-\log(P(D|\theta)P(\theta)) = NLL_{MAP}(\theta) = -\log(\theta)(a-1+\sum_{i=1}^{m} x^{(i)}) - \log(1-\theta)(b-1+m-\sum_{i=1}^{m} x^{(i)})$$

$$\theta_{MAP} = argmin_\theta NLL_{MAP}(\theta)$$

$$\frac{\partial}{\partial\theta}NLL(\theta) = -\frac{1}{\theta}(a-1+\sum_{i=1}^{m}x^{(i)}) + \frac{1}{1-\theta}(b-1+m-\sum_{i=1}^{m}x^{(i)}) = 0$$

$$(1-\theta_{MAP})(a-1+\sum_{i=1}^{m}x^{(i)}) = \theta_{MAP}(b-1+m-\sum_{i=1}^{m}x^{(i)})$$

$$a-1+\sum_{i=1}^{m}x^{(i)} - a\theta_{MAP} + \theta_{MAP} = \theta_{MAP}(b-1+m-\sum_{i=1}^{m}x^{(i)})$$

$$a-1+\sum_{i=1}^{m}x^{(i)} = \theta_{MAP}(a+b-2+m-\sum_{i=1}^{m}x^{(i)})$$

$$\theta_{MAP} = \frac{a-1+\sum_{i=1}^{m}x^{(i)}}{(a+b-2+m-\sum_{i=1}^{m}x^{(i)})}$$

$$\theta_{MAP}(a=1,b=1) = \frac{\sum_{i=1}^{m}x^{(i)}}{(m-\sum_{i=1}^{m}x^{(i)})} = \theta_{MLE}$$

- The logistic regression posterior probabilities are given as follows:

$$P(y=1|x)_{LR} = \frac{1}{1+\exp(-\theta^T x)} = g(\theta^T x)$$

$$P(y=0|x)_{LR} = \frac{\exp(-\theta^T x)}{1+\exp(-\theta^T x)} = \frac{1}{1+\exp(\theta^T x)} = 1 - g(\theta^T x)$$

- The following calculations are the posterior probabilities for the Guassian Naive Bayes model

$$P(y=1|x) = \frac{P(y=1)P(x|y=1)}{P(y=1)P(x|y=1)+P(y=0)P(x|y=0)} = \frac{1}{1+\frac{P(y=0)P(x|y=0)}{P(y=1)P(x|y=1)}}$$

$$P(y=1|x) = \frac{1}{1+\exp\left(\ln\left(\frac{P(y=0)P(x|y=0)}{P(y=1)P(x|y=1)}\right)\right)} = \frac{1}{1+\exp\left(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}\ln\left(\frac{P(x_j|y=0)}{P(x_j|y=1)}\right)\right)}$$

$$P(y=1|x) = \frac{1}{1+\exp\left(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}\ln\left(\frac{\frac{1}{\sigma_j\sqrt{2\pi}}\exp\left(-\frac{(x_j-\mu_j^0)^2}{2\sigma_j^2}\right)}{\frac{1}{\sigma_j\sqrt{2\pi}}\exp\left(-\frac{(x_j-\mu_j^1)^2}{2\sigma_j^2}\right)}\right)\right)}$$

$$P(y=1|x) = \frac{1}{1+\exp(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}\ln(\exp(\frac{(x_j-\mu_j^1)^2-(x_j-\mu_j^0)^2}{2\sigma_j^2})))}$$

$$P(y=1|x) = \frac{1}{1+\exp(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}\frac{(x_j-\mu_j^1)^2-(x_j-\mu_j^0)^2}{2\sigma_j^2})}$$

$$P(y=1|x) = \frac{1}{1+\exp(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}\frac{x_j^2-2x_j\mu_j^1+(\mu_j^1)^2-(x_j^2-2x_j\mu_j^0+(\mu_j^0)^2)}{2\sigma_j^2})}$$

$$P(y=1|x) = \frac{1}{1+\exp(\ln(\frac{1-\gamma}{\gamma}) + \sum_{j=1}^{d}x_j\frac{u_j^0-u_j^1}{\sigma_j^2} + \frac{(\mu_j^1)^2-(\mu_j^0)^2}{2\sigma_j^2})}$$

$$P(y=0|x) = \frac{1}{1+\exp(\ln(\frac{\gamma}{1-\gamma}) + \sum_{j=1}^{d}x_j\frac{u_j^1-u_j^0}{\sigma_j^2} + \frac{(\mu_j^0)^2-(\mu_j^1)^2}{2\sigma_j^2})} = 1 - P(y=1|x)$$

- If class 1 and class 2 are equally likely, then that means that $P(y=1|x) = P(y=0|x) = \gamma = 1-\gamma$. This means that:

$$P(y=1|x) = \frac{1}{1+\exp(\ln(1) + \sum_{j=1}^{d}x_j\frac{u_j^0-u_j^1}{\sigma_j^2} + \frac{(\mu_j^1)^2-(\mu_j^0)^2}{2\sigma_j^2})}$$

$$P(y=1|x) = \frac{1}{1+\exp(\sum_{j=1}^{d}\frac{u_j^0-u_j^1}{\sigma_j^2}x_j + \sum_{j=1}^{d}\frac{(\mu_j^1)^2-(\mu_j^0)^2}{2\sigma_j^2})}$$

If we let $\theta_0 = -\sum_{j=1}^{d}\frac{(\mu_j^1)^2-(\mu_j^0)^2}{2\sigma_j^2}$ and $\theta_j = -\frac{u_j^0-u_j^1}{\sigma_j^2}$, we get:

$$P(y=1|x)_{NB} = \frac{1}{1+\exp(-\theta_0 - \sum_{j=1}^{d}\theta_j x_j)} = \frac{1}{1+\exp(-\theta^T x)} = P(y=1|x)_{LR}$$

- 3.1
  Note that initially, $\theta_j^{(k)} \sim N(0, .0001)$, for all $0 \leq j \leq d, 1 \leq k \leq K.$, and $\lambda = 0$. This means that $\theta^{(1)} \approx \theta^{(2)} \approx ... \approx \theta^{(K)}$. This in turn gives us

$$J(\theta_{init}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp(\theta^{(k)^T} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)^T} x^{(i)})} + \frac{\lambda}{2m}\sum_{i=1}^{d}\sum_{k=1}^{K} \theta_{jk}^2$$

$$J(\theta_{init}) \approx -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp(\theta^{(1)^T} x^{(i)})}{K \exp(\theta^{(1)^T} x^{(i)})} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{1}{K}$$

$$J(\theta_{init}) \approx -\frac{1}{m}\sum_{i=1}^{m} \log \frac{1}{K} = -\frac{m}{m} \log \frac{1}{K} = -\log \frac{1}{K}$$

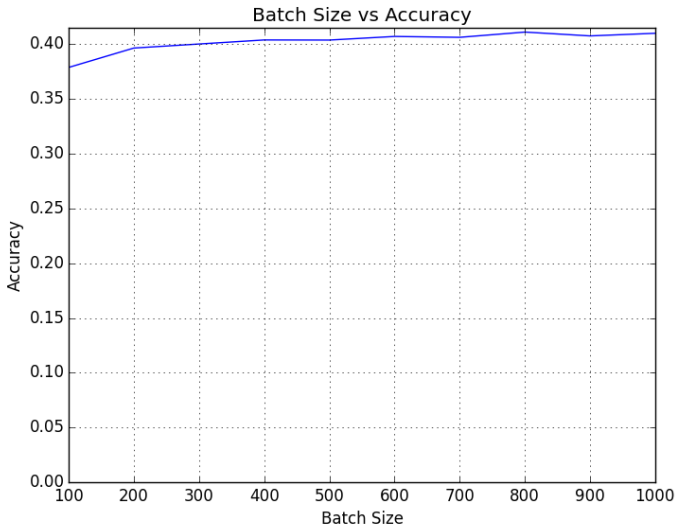
  $K = 10$, so $J(\theta_{init}) \approx -\log \frac{1}{10}$

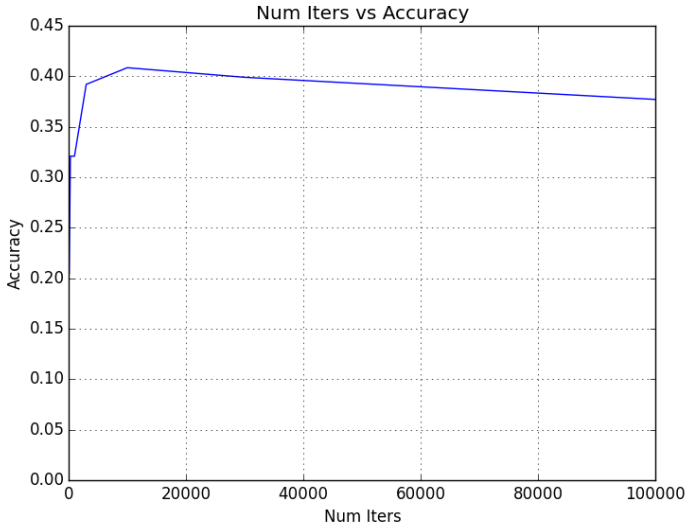
- 3.7
  Below is our confusion matrix:

$$\begin{pmatrix}
502 & 51 & 37 & 23 & 12 & 20 & 26 & 45 & 206 & 78 \\
60 & 492 & 18 & 38 & 22 & 34 & 44 & 39 & 94 & 159 \\
120 & 56 & 235 & 70 & 111 & 92 & 168 & 62 & 59 & 27 \\
50 & 70 & 82 & 218 & 56 & 186 & 146 & 61 & 57 & 74 \\
65 & 34 & 118 & 54 & 286 & 69 & 191 & 112 & 37 & 34 \\
44 & 43 & 100 & 137 & 63 & 336 & 101 & 68 & 77 & 31 \\
22 & 51 & 63 & 80 & 73 & 79 & 531 & 32 & 23 & 46 \\
57 & 47 & 57 & 46 & 82 & 72 & 65 & 408 & 54 & 112 \\
152 & 77 & 7 & 19 & 7 & 48 & 15 & 14 & 554 & 107 \\
69 & 171 & 13 & 25 & 10 & 16 & 46 & 43 & 117 & 490
\end{pmatrix}$$

  We can see decent numbers along the diagonal, corresponding to our roughly 40% accuracy. The biggest pairwise confusion from our confusion matrix was 206 images of planes being misclassified as ships (151 images vice-versa), 186 images of cats being misclassified as dogs (137 images vice-versa), and 171 images of cars being misclassified as trucks (159 images vice-versa). Looking at CIFAR theta pdf, this makes sense because the images produced for plane and ship are similar, and the images for car and truck are relatively similar. Also, images were frequently misclassified as deer, with over 500 images cats, deer, dogs and frogs being misclassified. Looking at the CIFAR theta pdf again, this is intriguing because it looks like the cat and dog images are similar and the bird and deer images are similar, but the frog has a lot of green in the middle. These misclassifications may stem from the noise in the outside of the frog model image. Obviously, we can create a better model, but it seems as if the model was able to at least learn the difference between the 6 animate objects and the 4 inanimate objects

- Extra credit 3.8
  Here are the plots of test accuracy against the batch size and number of iterations





Using the results from the **validation data**, we came up with the following for the best combination of hyperparameters:
Batch size = 600
Number of Iterations = 10000
Learning rate = 5e-7
Regularization Strength = 500000
We found this combination by looping over a range of possible values for each parameter,

testing on the validation set and picked the best set of parameters. The accuracy we get on the test set with the parameters chosen just off the training and validation sets is 40.69%.

We implemented the functionality for the fmin bfgs function and for testing various lambdas for fmin bfgs softmax models but ran into strange "divide by 0" errors during log operations while fmin bfgs was calling our fmin bfgs helper functions. The code for our attempt is included in softmax.py and softmax-hw.py.

- 3.8
  Below are the per-genre classification accuracies using the CEPS dataset (on which we were able to get much higher accuracies) for One-vs-all classification with L2 regularization and Softmax:

Genre	OVA L2	Softmax
Blues	0.42857143	0.57142857
Classical	0.91304348	1
Country	0.41666667	0.25
Disco	0.42857143	0.0952381
Hiphop	0.31578947	0.15789474
Jazz	0.33333333	0.04166667
Metal	0.84	1.
Pop	1.	1.
Reggae	0.55	0.05
Rock	0.11764706	0.

For the most part, Softmax performs worse than OVA for every genre except for metal, classical, and blues, in which case it performs notaably better (.15 or so). This makes sense, because many of the genres have overlaps, which means the data has some overlap, whereas Softmax performs better on more separable data. That also might be why Softmax performs better on Metal, Classical, and Blues, because those genres of music are very different (more separable) than the other types of music. Thus, for this type of problem, because we're trying to classify many types of music, some of which are similar, One-vs-all is a better method for this music classification than Softmax, unless you are specifically interested in classifying Metal, Blues, and/or Classical music, in which case the Softmax method is the way to go. A quick note - we used the FFT data to train OVA and Softmax models as well, but these had .265 and .18 accuracies, respectively, so interpretting these models is much less meaningful since they don't do much better than the baseline .1 accuracy of a randomly guessing model. However, looking at these FFT-trained models, we noticed that OVA outperformed Softmax on every genre except Blues; however, after looking at the confusion matrix, we noticed that this "improvement" was due to the fact that most guesses by the softmax FFT model were the Blues genre. Again, this supports our conclusion that OVA performs better on the music classification problem than Softmax, likely due to the large overlap of many genres.