- $$\frac{\partial g(z)}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2}$$
$$\frac{\partial g(z)}{\partial z} = \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1+e^{-z}}{(1+e^{-z})^2} - \frac{1}{(1+e^{-z})^2}$$
$$\frac{\partial g(z)}{\partial z} = \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} = g(z) - g(z)^2 = g(z)(1-g(z))$$

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

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

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

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

- To prove that H, which is a d x d matrix, is positive definite, we need to show that $a^T H a > 0, \forall a \in \mathbb{R}^d$. Let a be a vector in $\mathbb{R}^d$. Also note that for the X matrix in our equation $H = X^T S X$, each row i of X is a training example vector in R^d denoted as $x^{(i)}$ (we'll denote the row vector as $(x^{(i)})^T$). We can let $B = Xa$ be a vector in R^m where $B_i = (x^{(i)})^T a$. With this, we can see that
$$a^T H a = a^T X^T S X a = B^T S B$$
$$a^T H a = \sum_{i=1}^{m} a^T (x^{(i)}) S_{i,i} (x^{(i)})^T a = \sum_{i=1}^{m} S_{i,i}((x^{(i)})^T a)^2$$
Since $S_{i,i} > 0$ for all $1 \le i \le m$, $a^T H a > 0$, which means H is positive definite as desired

Assume for the sake of a contradiction, that $||\theta_{MAP}||_2 > ||\theta_{MLE}||_2$. We note that the pdf for $N(0, \alpha^2 I)$ is:

$$P(\theta) = \frac{1}{\sqrt{(2\pi)^{d+1}|\Sigma|}} \exp\Big(\frac{-1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\Big) = \frac{1}{\sqrt{(2\pi\alpha^2)^{d+1}}} \exp\Big(\frac{-1}{2\alpha^2}||\theta||_2\Big)$$

Because $||\theta_{MAP}||_2 > ||\theta_{MLE}||_2$, that must mean that $P(\theta_{MLE}) > P(\theta_{MAP})$. By the definition of θ_{MLE}, we know that

$$\prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MLE}) \geq \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MAP})$$

Because $P(\theta_{MLE}) > P(\theta_{MAP})$, we can also see that

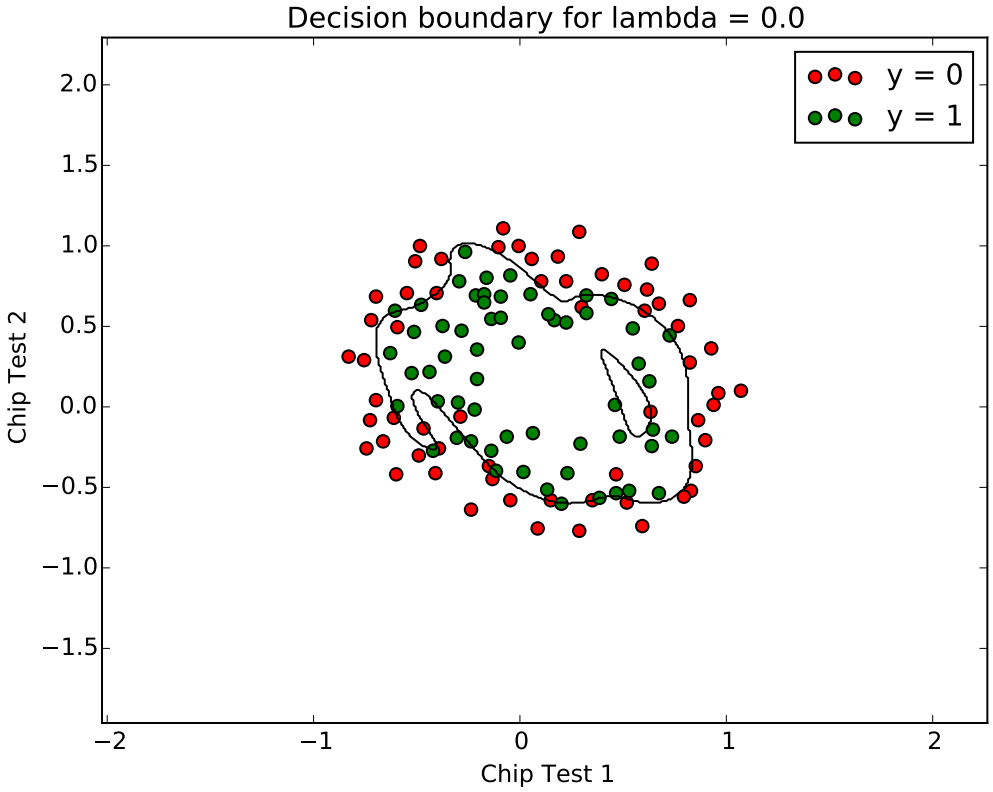
$$P(\theta_{MLE})\prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MLE}) > P(\theta_{MAP})\prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MAP})$$

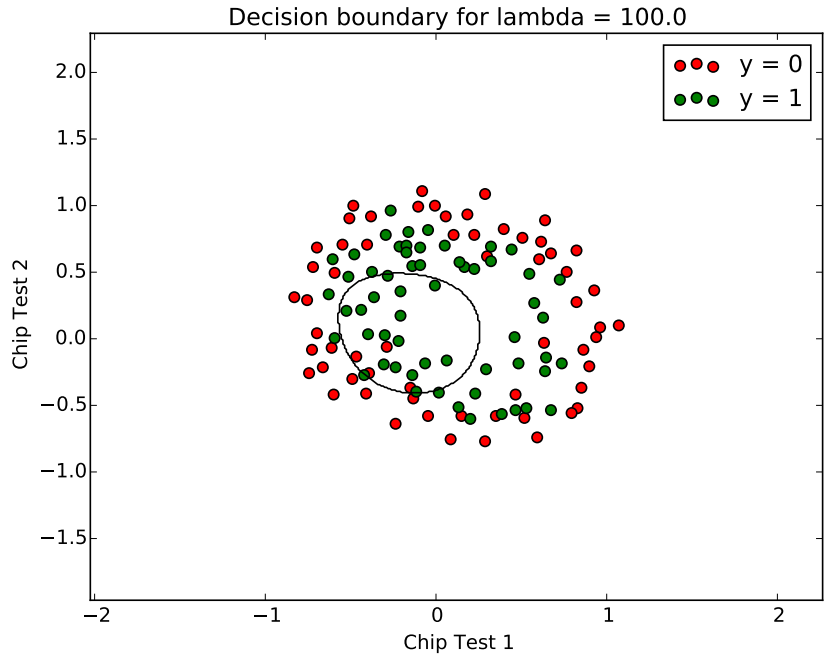
Likewise, we can use the definition of θ_{MAP} to get the following inequality

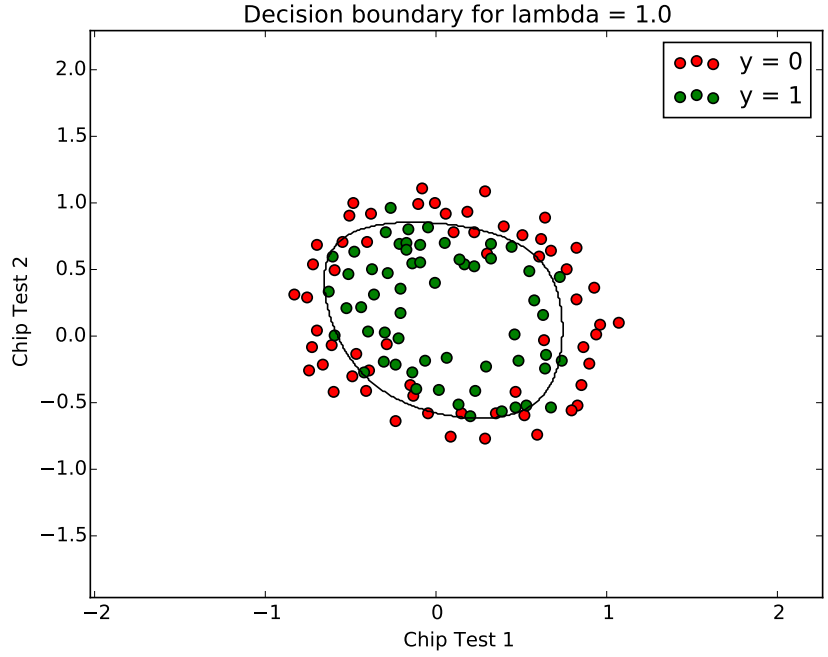
$$P(\theta_{MAP})\prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MAP}) \geq P(\theta_{MLE})\prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta_{MLE})$$

However, this inequality directly contradicts our the inequality before it, which means that $||\theta_{MAP}||_2 \leq ||\theta_{MLE}||_2$, as desired

- 3B3.
  Below, we have the decision boundaries described by the model when regularized with a λ of 0, 100, and 1. We can see that when $\lambda = 0$, the model does a great job of separating the red dots from the green dots: in fact, it is doing too good of a job and overfitting the given data. On the other hand, a regularization parameter of 100 results in underfitting the model, with the model not doing a great job of separating the green and red dots, and not big enough to accurately categorize new (or old) data points. The regularization parameter of 1 results in a model that does a good job of separating the green and red points but doesn't overfit the given data.




Decision boundary for lambda = 100.0
y = 0
y = 1
Chip Test 2
Chip Test 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2


Decision boundary for lambda = 1.0
y = 0
y = 1
Chip Test 2
Chip Test 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2

- 3C.
  The optimal θ obtained using L1 regularization is significantly more sparse than the θ obtained using L2 regularization. Most of the coefficients in θ_{L1} were 0, whereas, for θ_{L2}, while there were many tiny coefficients, there were no 0's. The performance for each type of regularization and each type of data transform is given below:

  L2:
  STD: 92.96875
  LOG: 93.8802083333
  BIN: 92.7734375
  L1:
  STD: 92.2526041667
  LOG: 94.2057291667
  BIN: 92.578125

  The L2 regularization performs better when the data is transformed by subtracting the mean and dividing by the standard deviation, or by binarizing the data. However, transforming the data by applying the $log(x + .1)$ function actually works the best for either regularization, and the L1 regularization combined with that function gives the highest accuracy of 94.21%. To us, this makes sense when we look at the sparsities of the models. L1 regularization model being so sparse indicates that some of the words picked for characteristics of spam actually have no bearing on whether the email is spam or not, whereas the L2 regularization model thinks those words are factors and slightly overfits the data given.

- 3D.
  Compare the prediction performance with the FFT representation and the Mel Cepstral representation. Is the performance sensitive to choice of regularization parameter? Which genre is easiest to classify? Which is the most difficult? Discuss how you can improve the performance of such a classifier. Include answers to these questions in your writeup.
  The performance with FFT representation over several runs of our classifiers was around 30%, whereas the performance with Mel Cepstral representation was around 55%. Furthermore, running our classifier on Mel Cepstral data was much quicker than running classifiers on FFT representations. After running the classifiers on both FFT and Mel Ceptstral represented data, we also found that different λ parameters performed differently. We tested λ values of $10^n, -6 \leq n \leq 6$, and the classifiers performed the best when $-1 \leq n \leq 3$

  In terms of easiest genres to classify, our L1 classifier did the best job of classifying classic and pop music, with accuracies of 95% and 88% respectively. On the other hand, rock and hip hop were the hardest to classify, with 7% and 29% respectively.

  With average accuracies of 55%, there's definitely room for improvement. One way to improve is to combine the FFT and Mel Cepstral data sets into one data set so that the models could leverage information from both datasets. Another way to improve is to compute more coefficients similar to the Mel Cepstral data and use them as features as well as using more than just the first 10 seconds for FFT. Finally, we could also use more than just the few songs we were given (there are millions of songs out there!).