CS260, Winter 2017
Problem Set 2: Bayes, Logistics, and Decision Trees
Due 2/1/2017

Ray Zhang

1 Q1

Jan 21, 2017

The most frequent words found in the training data are of follows:

{(enron: 600), (will: 351), (please: 291)}

2 Q2

The update rule for logistic regression for batch gradient descent is in the general form:

$w^{t+1} = w^t - \eta \nabla_w f$

$b^{t+1} = b^t - \eta \frac{\partial f}{\partial b}$

For some step size η.

The cost function is:

$L(w) = -\sum_{i=1}^{N} y_i log\sigma(w^T x_i + b) + (1 - y_i) log(1 - \sigma(w^T x_i + b))$

Using every data point to calculate $\nabla_w f$, we get:

$\nabla_w f = -\sum_{i=1}^{N} y_i(1 - \sigma(w^T x_i + b))x_i - (1 - y_i)\sigma(w^T x_i + b)x_i$

$\nabla_w f = -\sum_{i=1}^{N} x_i(y_i(1 - \sigma(w^T x_i + b)) - (1 - y_i)\sigma(w^T x_i + b))$

Solving for the inside:

$y_i(1 - \sigma(w^T x_i + b)) - (1 - y_i)\sigma(w^T x_i + b) =$

$y_i - \sigma(w^T x_i + b)y_i - \sigma(w^T x_i + b) + \sigma(w^T x_i + b)y_i =$

$y_i - \sigma(w^T x_i + b)$

Plugging back in:

$\nabla_w f = -\sum_{i=1}^{N} x_i(y_i - \sigma(w^T x_i + b))$

$\nabla_w f = \sum_{i=1}^{N} (\sigma(w^T x_i + b) - y_i)x_i$

With similar logic, using every data point to calculate $\frac{\partial f}{\partial b}$, we get:

$\frac{\partial f}{\partial b} = \sum_{i=1}^{N} \sigma(w^T x_i + b) - y_i$

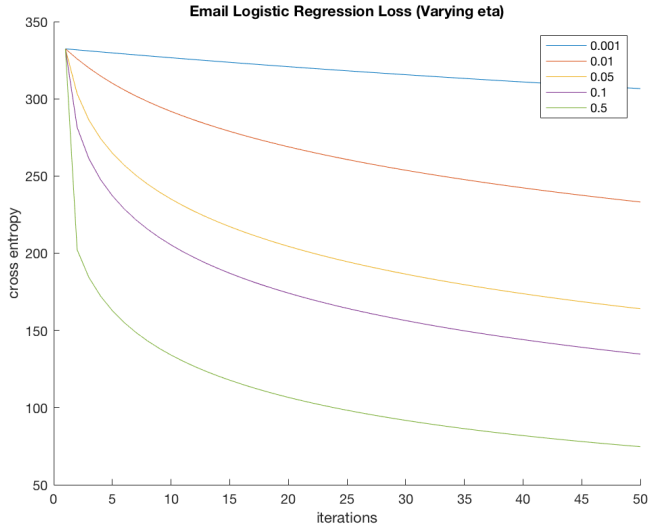
The solution is:

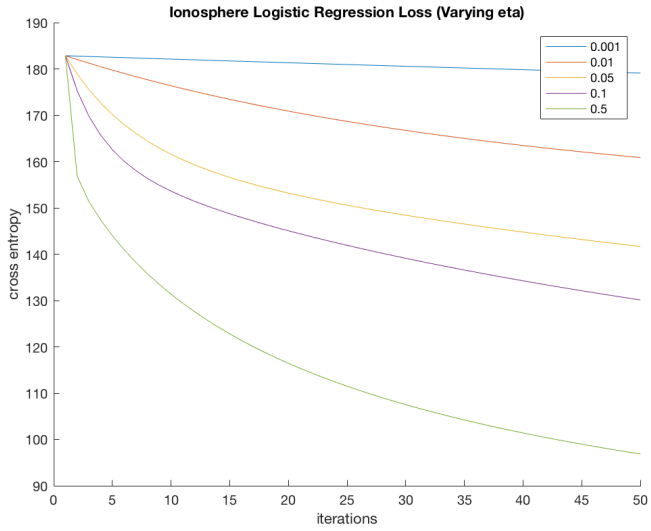
$w^{t+1} = w^t - \eta \sum_{i=1}^{N} (\sigma(w^T x_i + b) - y_i)x_i = w^t - \eta X^T(\sigma(Xw + b) - y)$

$b^{t+1} = b^t - \eta \sum_{i=1}^{N} \sigma(w^T x_i + b) - y_i$

3 Q3

Part A:





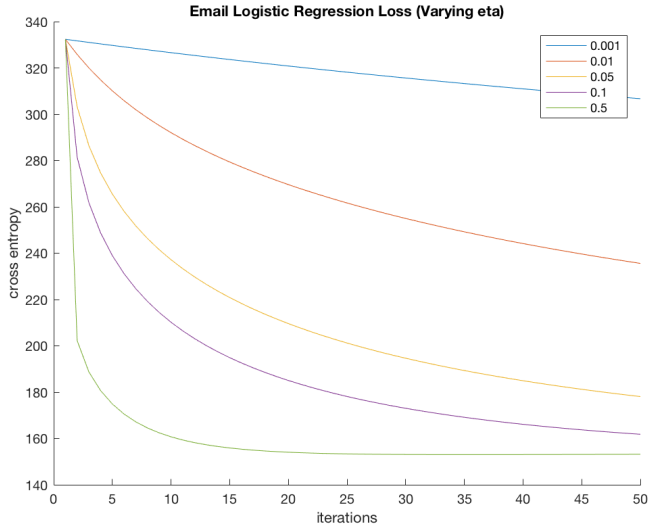
Part B:

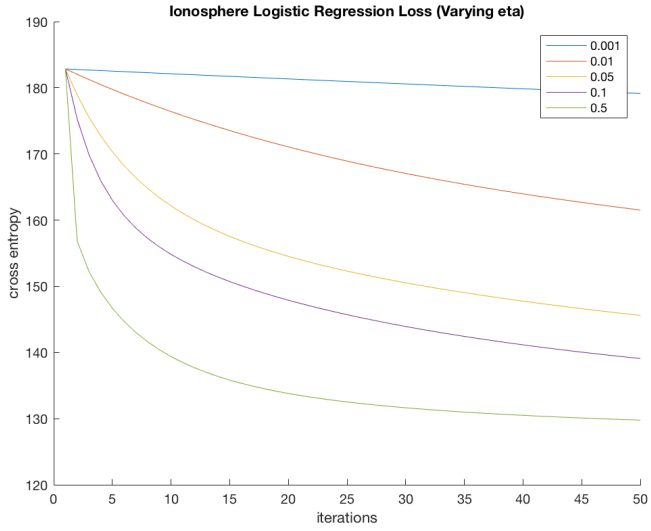
Table 1: ω magnitudes

η	0.001	0.01	0.05	0.1	0.5
Email	5.163215e-02	3.013344e-01	8.047203e-01	1.182283e+00	2.690593e+00
Ionosphere	2.592360e-02	1.966731e-01	5.352835e-01	8.094188e-01	2.076794e+00

4 Q4

Part A:





Part B:

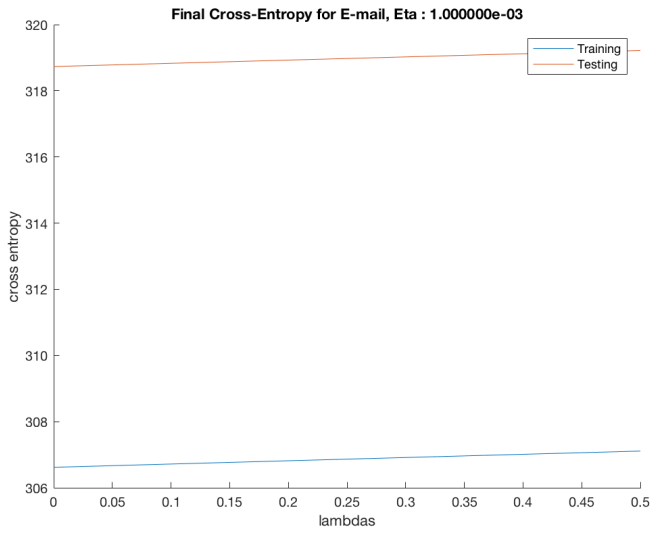
Table 2: ω magnitudes

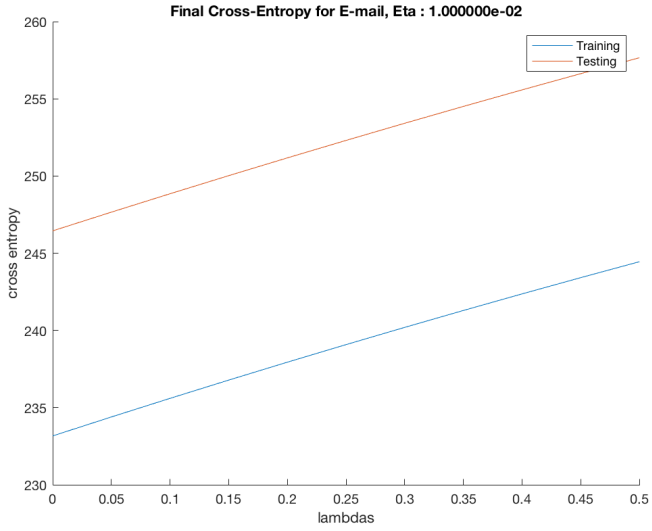
λ	0	0.05	0.1	0.15	0.2
Email	3.013344e-01	2.943992e-01	2.876993e-01	2.812259e-01	2.749702e-01
Ionosphere	1.966731e-01	1.922872e-01	1.880358e-01	1.839143e-01	1.799182e-01

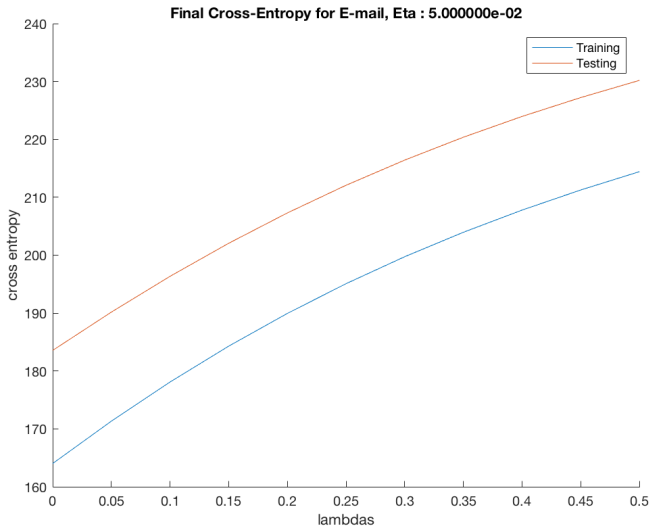
Table 3: ω magnitudes(cont'd)

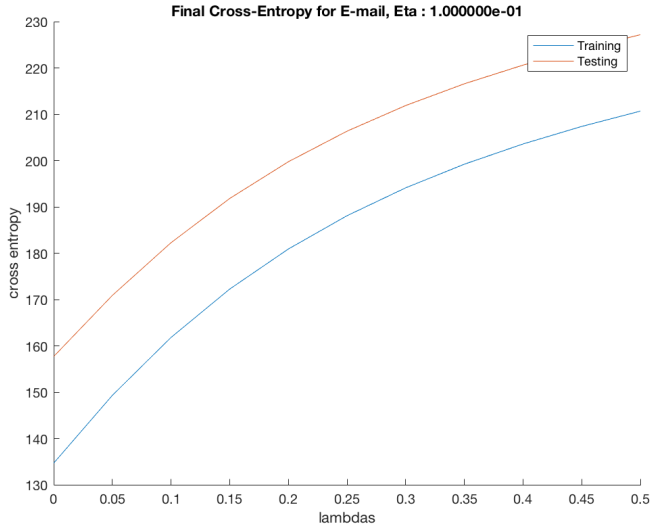
...	0.25	0.3	0.35	0.4	0.45	0.5
...	2.689240e-01	2.630793e-01	2.574285e-01	2.519641e-01	2.466792e-01	2.415669e-01
...	1.760430e-01	1.722847e-01	1.686392e-01	1.651026e-01	1.616712e-01	1.583415e-01

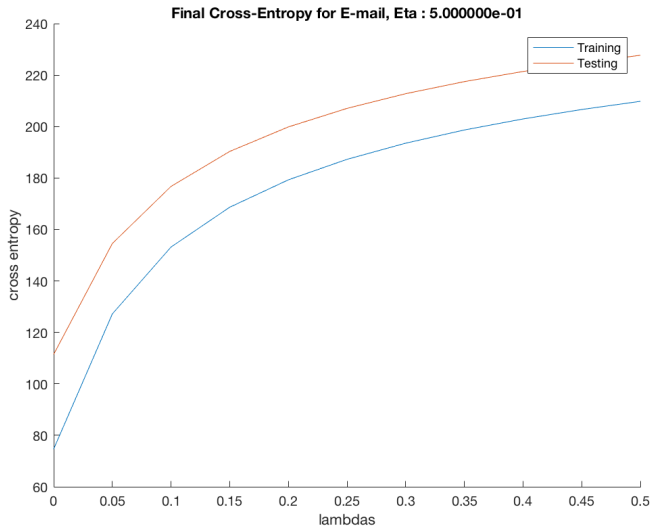
Part C:

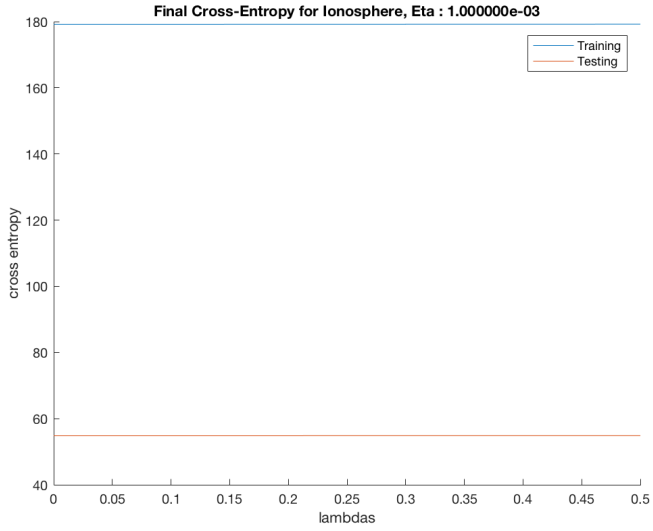


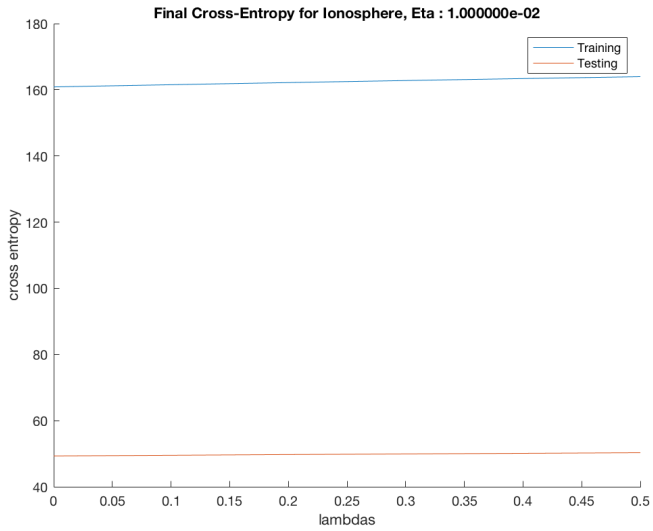

Final Cross-Entropy for E-mail, Eta : 1.000000e-02
Training
Testing
cross entropy
lambdas
260
255
250
245
240
235
230
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

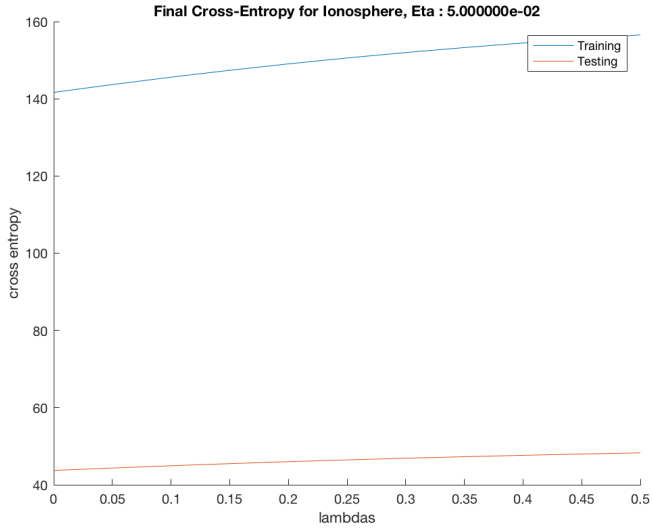

Final Cross-Entropy for E-mail, Eta : 5.000000e-02
Training
Testing
cross entropy
lambdas
240
230
220
210
200
190
180
170
160
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

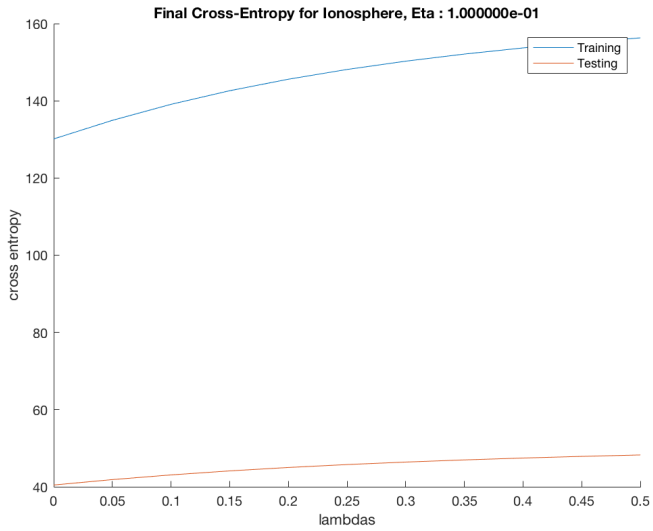

Final Cross-Entropy for E-mail, Eta : 1.000000e-01
Training
Testing
cross entropy
lambdas
230
220
210
200
190
180
170
160
150
140
130
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

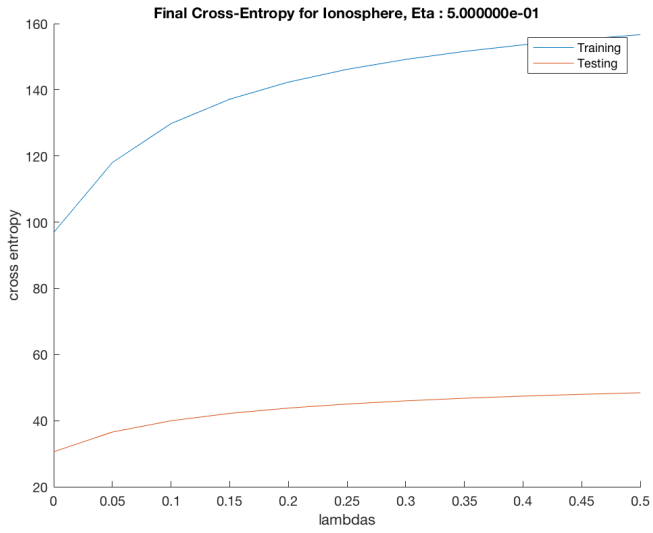

Final Cross-Entropy for E-mail, Eta : 5.000000e-01
Training
Testing
cross entropy
lambdas
240
220
200
180
160
140
120
100
80
60
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Final Cross-Entropy for Ionosphere, Eta : 1.000000e-03
Training
Testing
cross entropy
lambdas
180
160
140
120
100
80
60
40
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Final Cross-Entropy for Ionosphere, Eta : 1.000000e-02
Training
Testing
cross entropy
lambdas
180
160
140
120
100
80
60
40
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Final Cross-Entropy for Ionosphere, Eta : 5.000000e-02
Training
Testing
cross entropy
lambdas
160
140
120
100
80
60
40
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Final Cross-Entropy for Ionosphere, Eta : 1.000000e-01
Training
Testing
cross entropy
lambdas
160
140
120
100
80
60
40
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5



5 Q5

Finding a zero(solution) in a univariate case can be simplified to the equation, when under the assumption that the function is a line:

$f(x) = 0 = f(a) + (x - a)f'(a)$

Which comes from the point slope equation(where the slope is the slope of the tangent line at $f(a)$):

$f'(a)(x - a) = f(x) - f(a)$, where $f(x) = 0$ because it's a solution.

$f'(a)x - f'(a)a = -f(a)$ leads to:

$x = a - f(a)/f'(a)$ after separating x. This is the newton's method.

Generalizing to a higher dimension, we can see that:

$f(x) = 0 = f(a) + \nabla_x f(a)(x - a)$ is the analog.

We can solve and get $x = a - \nabla_x f(a)^{-1} f(a)$

However, we want to find the zero of the gradient of a function, not just the function itself.

$\nabla f(x) = 0 = \nabla f(a) + H(a)(x - a)$

$x = a - H(a)^{-1}\nabla f(a)$

Using the above, we can conclude that we need to find the gradient and the hessian of $f(x)$ to compute every iteration of Newton's method.

We already know the gradient:

$\nabla f(w) = \sum_{i=1}^{N}(\sigma(w^T x_i + b) - y_i)x_i$

$\frac{\partial f}{\partial w_i} = \sum_{n=1}^{N}(\sigma(w^T x_n + b) - y_i)x_{ni}$

To take the second derivative to find the hessian, take the partial of any arbitrary w_i and then partial of w_j. We have the following cases:

Case 1: $i = j$

$\frac{\partial^2 f}{\partial w_i \partial w_i} = \frac{\partial}{\partial w_i}\sum_{n=1}^{N}(\sigma(w^T x_n + b) - y_i)x_{ni} =$

$\sum_{n=1}^{N}\sigma(w^T x_n + b)(1 - \sigma(w^T x_n + b))x_{ni}^2$

Case 2: $i \neq j$

$\frac{\partial^2 f}{\partial w_i \partial w_i} = \frac{\partial}{\partial w_i}\sum_{n=1}^{N}(\sigma(w^T x_n + b) - y_i)x_{ni} =$

$\sum_{n=1}^{N}\sigma(w^T x_n + b)(1 - \sigma(w^T x_n + b))x_{ni}x_{nj}$

In fact, we can merge both of these cases into one, and express in matrix products:

$H(w) = \sum_{n=1}^{N}\sigma(w^T x_n + b)(1 - \sigma(w^T x_n + b))x_n x_n^T =$

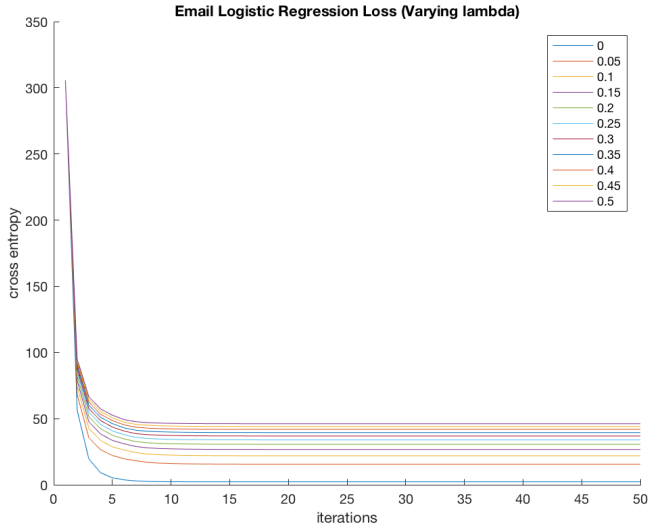
$X^T diag(\sigma(Xw + b)(1 - \sigma(Xw + b)))^T X$

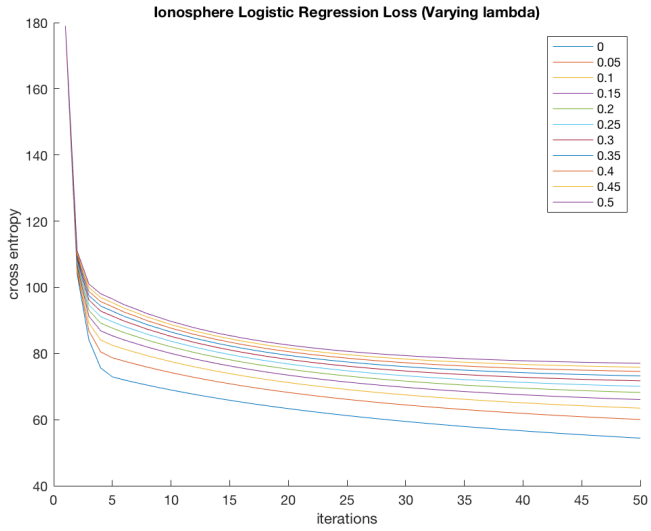
6 Q6

Refer to Q7.

7 Q7

Part A:


Email Logistic Regression Loss (Varying lambda)
cross entropy
iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Ionosphere Logistic Regression Loss (Varying lambda)
cross entropy
iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Part B:

Table 4: ω magnitudes

λ	0	0.05	0.1	0.15	0.2
Email		1.246767e+01	1.036353e+01	9.221122e+00	8.454301e+00
Ionosphere	1.168685e+01	8.574101e+00	7.295088e+00	6.546401e+00	6.036899e+00

Table 5: ω magnitudes(cont'd)

...	0.25	0.3	0.35	0.4	0.45	0.5
...	7.885981e+00	7.439461e+00	7.074766e+00	6.768513e+00	6.505895e+00	6.276976e+00
...	5.658916e+00	5.362405e+00	5.120572e+00	4.917604e+00	4.743483e+00	4.591504e+00

Part C:

Table 6: Cross Entropy (test set)

λ	0	0.05	0.1	0.15	0.2
Email	3.622788e+03	1.223549e+02	1.161244e+02	1.139292e+02	1.130654e+02
Ionosphere	4.018797e+01	3.294442e+01	3.121129e+01	3.106500e+01	3.144736e+01

Table 7: Cross Entropy (test set, cont'd)

...	0.25	0.3	0.35	0.4	0.45	0.5
...	1.128120e+02	1.128872e+02	1.131552e+02	1.135418e+02	1.140032e+02	1.145121e+02
...	3.199465e+01	3.258293e+01	3.316763e+01	3.373220e+01	3.427090e+01	3.478245e+01

8 Q8

Regarding Q3.a):

The cross entropy function value decreases because the function is convex and we are going down the gradient. Intuitively, the bigger the gradient descent, the greater the magnitude of decrease(not including edge cases i.e. must be convex, didn't skip over minima, etc). Thus, the larger the step size the sooner it converges towards a reasonable minima.

However, it should be noted that the descent for the largest step size($eta = 0.5$) is no longer decreasing quickly after a couple iterations. This is because differentiable convex functions have their hessians to be positive semidefinite. This means the gradient always increases from the negative infinity boundary. If differentiable, the gradient will become smaller and smaller towards the minima. This also implies the gradient jump will be smaller near the optima.

Conversely, the smaller step sizes($eta = 0.001$) will train very slowly, but will likely converge to a better minima, because large step sizes will jump over the minima.

Regarding Q3.b):

The norm always increased as the iterations followed. This is due to the gradients increasing the magnitude of w. At every gradient step we are essentially subtracting a non-positive term to w:

$\frac{\partial f}{\partial w_i} = \sum_{n=1}^{N}(\sigma(w^T x_n + b) - y_n)x_{ni}$

This means we are constantly increasing the dot product within $w^T x$. And thus, the norm continuously increases.

Regarding Q4.a):

When we add a larger penalty term, it is obvious that the cost function will be monotonically larger. We are essentially multiplying the regularization term by a larger value.

One interesting point is that for a lambda ¿ 0, gradient descent using a larger step size will be penalized the most. This is due to the fact that because our norm(w) always increases during gradient descent, the norm(w) will be penalized as fast as it's growing due to the regularizer. Smaller step sizes are not as affected because their change in norm(w) is less drastic.

Regarding Q4.b):

Because we have a non-zero regularization term, the norm(w) will obviously be smaller than it was before(without regularization). And it is.

Regarding Q4.c):

We should suspect that our cross entropy function value for both the test and the training datasets after w has been trained, will be very similar in their respective scales. Since there are less data points in the test dataset, the unnormalized cross entropy will obviously be lower.

Interesting Points:

- Rate of convergence(in terms of α-sublevels) vs. η: Positively correlates linearly. According to the graph, given 2 step sizes, x, y, where $x = ky$, then denote n as the number of iterations until a certain α-sublevel is reached for x. y will reach the same α-sublevel after kn iterations.

- The change in magnitude of w vs. λ: Negatively correlates. According to the metrics, for a reasonably large η, i.e. 0.1 or 0.5, the magnitude of w is severely penalized. For example: 7.502437e-01 vs. 2.076794e+00. It appears that the magnitude of w is severely penalized after the function has been optimized to a degree. At first, because w is naturally small(we initialized as 0), the penalty of L2 regularizer is not significant.

- The value of cross entropy function for different values of λ, η are explained in the previous observations in Regarding Q4.a) and Q3.a).

9 Q9

Newton's method and gradient descent are two different ways to minimize a convex objective. Newton's method applies well to smaller datasets with smaller features, such as the ionosphere dataset, while gradient descent is the more reasonable solution to larger featured datasets.

The reason for Newton's method performing well in smaller datasets is because of its fast convergence time. Computing the hessian takes $O(ND^2)$, and inverting it takes $O(D^3)$, and for smaller dimensions this is fine. The fast convergence time can be seen from the graph of iterations vs. cross entropy value.

The reason for gradient descent performing well in larger datasets is because of its fast computing time. Computing the (batch) gradient takes $O(ND)$, and this is very scalable for large featured datasets. For example, on the email dataset, there are 1000 features, and thus computing the hessian took

way too long, meanwhile the gradient descent method was able to converge in very fast time.

10 Discussion

I have discussed the problem set with the TA's of the class, Denali Molitor, and Frank Chen.