CPSC 340 Assignment 4 (due Friday, Nov 2 at 11:55pm)

1 Convex Functions

1. $f'(w) = 2\alpha w - \beta$ and $f''(w) = 2\alpha$. Since $2\alpha > 0$, the function is convex.
2. $f'(w) = -\frac{1}{w}$ and $f''(w) = \frac{1}{w^2}$. Since $\frac{1}{w^2} > 0$, the function is convex.
3. $||Xw - y||_1$ and $\frac{\lambda}{2}||w||_1$ are L1 norm so they are convex. Their sum is also convex.
4. $z = -y_i w^T x_i$ is linear so it is convex. Then $g(z) = \sum_{i=1}^{n} \log(1 + \exp(z))$. Then $g(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$ and $g''(z) = -\frac{1}{(1+e^{-z})^2}\frac{d}{dz}e^{-z} = \frac{e^{-z}}{(1+e^{-z})^2}$ Since g"(z) is positive, the function is convex.
5. $|w^T x_i - y_i|$ is a linear function so it is convex. 0 is convex. $\frac{\lambda}{2}||w||_2^2$ is convex because $lambda > 0$ and L2 norm is convex. The sum of all of them are also convex.

2 Logistic Regression with Sparse Regularization

2.1 L2-Regularization

L2 Training Error = 0.002, L2 Validation Error = 0.074, Features used: 101, Gradient Iterations: 36

2.2 L1-Regularization

L1 Training Error = 0, L2 Validation Error = 0.052, Features used: 71, Gradient Iterations: 78

2.3 L0-Regularization

Training error 0.000, Validation error 0.018, Features used: 24

2.4 Discussion

L0 has the lowest validation error out of all 3 regularizations. This may be because it has the most sparsity out of the three and it stops the model from overfitting. However, L0 is considerably slower than the other two because of the while and for loops.

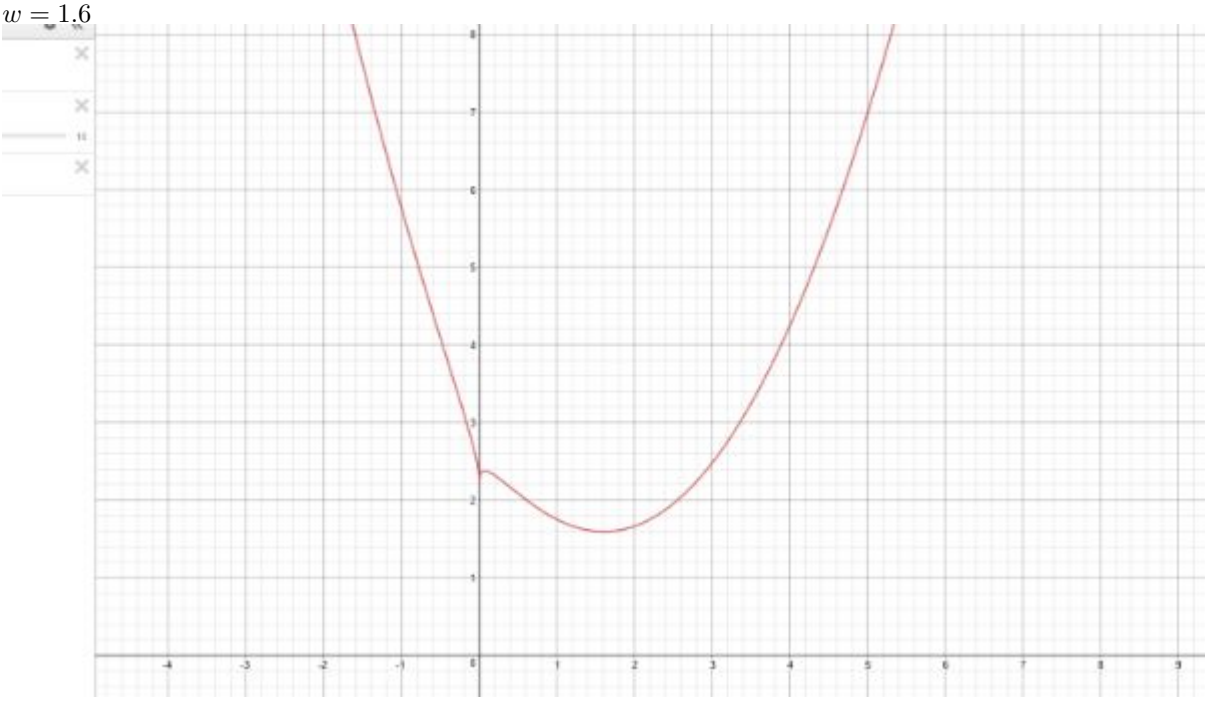
2.5 Comparison with scikit-learn

Scikit-learn's Logistic Regression outputs the same training and validation errors as our own L1 and L2 logistic regression.

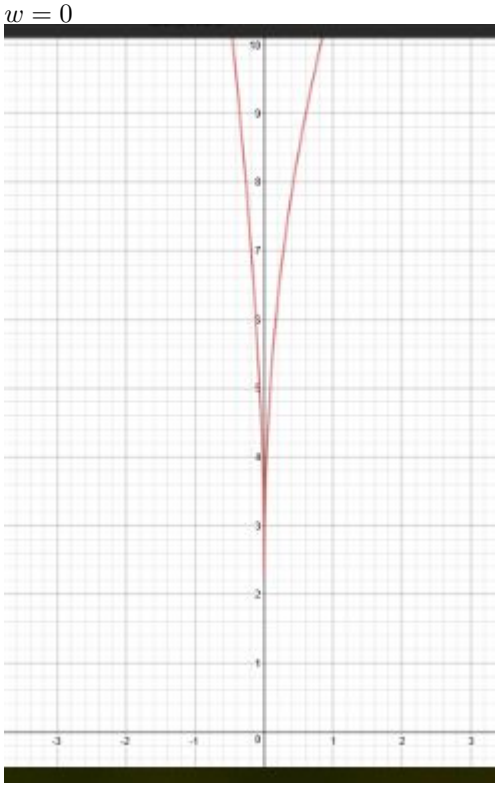
2.6 $L\frac{1}{2}$ regularization

1. $0.5((w-2)^2 + 0.5) + \lambda\sqrt{|w|}$
2. $w = 2$
3. $w = 0$

4. $w = 1.6$



5. $w = 0$



6. It behaves more like L1 regularization because there is sparsity in $L\frac{1}{2}$ regularization
7. It is not a convex optimization as you can see from the graph when $\lambda = 10$, the graph flares outwards.

3 Multi-Class Logistic

3.1 Softmax Classification, toy example

Class 1: $w_1\hat{\dot{x}} = 1$
Class 2: $w_2\hat{\dot{x}} = 4$
Class 3: $w_3\hat{\dot{x}} = 2$

We would assign 2 to the test example.

3.2 One-vs-all Logistic Regression

logLinearClassifier Validation error: 0.070

3.3 Softmax Classifier Gradient

$$f(w) = \sum_{i=1}^{n}[-w_{yi}^T x_i + log(\sum_{c'=1}^{k} exp(w_{c'}^T x_i))]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[\frac{\partial}{\partial w_{cj}}(-w_{yi}^T x_i) + \frac{\partial}{\partial w_{cj}}(log(\sum_{c'=1}^{k} exp(w_{c'}^T x_i)))]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[\frac{\partial}{\partial w_{cj}} - (w_{c1}^T x_{i1} + ... + w_{cd}^T x_{id}) + \frac{\partial}{\partial w_{cj}}(log(\sum_{c'=1}^{k} exp(w_{c'}^T x_i)))]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[(-x_{ij}\dot{I}(y_i = c)) + \frac{\partial}{\partial w_{cj}}(log(\sum_{c'=1}^{k} exp(w_{c'}^T x_i)))]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[(-x_{ij}\dot{I}(y_i = c)) + \frac{exp(w_{c'}^T x_i)}{\sum_{c'=1}^{k} exp(w_{c'}^T x_i)}\frac{\partial}{\partial w_{cj}}(w_{c'}^T x_i)]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[(-x_{ij}\dot{I}(y_i = c)) + \frac{exp(w_{c'}^T x_i)}{\sum_{c'=1}^{k} exp(w_{c'}^T x_i)}x_{ij}]$$

$$\frac{\partial}{\partial w_{cj}} = \sum_{i=1}^{n}[(x_{ij}(p(y_i = c|W, x_i) - I(y_i = c))]$$

3.4 Softmax Classifier Implementation

Validation error: 0.008

3.5 Comparison with scikit-learn, again

One vs All: Training error 0.084, Validation error 0.070 Softmax Function: Training error 0.000, Validation error 0.016

Scikit Learn's Implementation of One vs All yields the exact same validation error as our own logLinearClassifier. However, their validation error for sotmax classifier is actually double our validation error.

3.6 Cost of Multinomial Logistic Regression

1. Every iteration of gradient descent runs over O(n) examples. Each example costs O(dk) because we need to compute $w^T x_i$, which is O(d) for k values. The iterations of gradient descent is given by O(T)

so we have O(Tndk).

2. We need O(dk) to classify one example. Since there are t test examples its, O(tdk)

4 Very-Short Answer Questions

1. This is difficult to do because "relevance" is hard to define. Features may only be "relevant" in the context of other features. It also becomes more difficult if there is collinearity between features because we wouldn't know which feature to pick. Other issues that make it difficult may be conditional independence; where features can be irrelevant given other features.

2. This is similar to the first answer where a feature may be colinear with another feature so both would be selected. Another problem with this is that feature 1 may depend on feature 2 and only feature 1 is picked. Then it wouldm't make sense if feature 1 was picked without feature 2.

3. We would use L1-loss in datasets with a lot of outlier points and we would use L1-regularization in datasets that contain irrelevant features

4. L1 and L2 regularization are convex. Only L2 regularization yield unique solutions. L1 and L0 regularizations yield sparse solutions.

5. Increasing λ directly increases sparsity. Small lambda selects more features which causes Training Error to go down while approxiamation error goes down. Large lambda selects fewer features which can be a better approximation of the test data so approxiamation error decreases and training error increases.

6. We can try using the bootstrap method. We would run the feature selection on each sample. Each run would give us a set of features and we would use the union of all the sets of features. This allows us to take all the features in the bootstrap, so it allows us to grab all relevant features

7. This means that all the classes can be separated by a hyperplane

8. If we want to minimize the number of classification error, we would use the 0-1 Loss. However, the 0-1 Loss is non-smooth and not convex. We use logistic loss for its convexity and smoothness so we can minimize this with gradient descent.

9. Support vectors are vectors from the closest points to the boundary. These are useful because we can use them to find the boundary with the maximum distance from each point.

10. Perceptron will only work for linear models that have linearly separable datasets

11. Multiclass classifiers are better when there may be more than one "correct" class label and we can fit "k" binary classifers.

12. The σ controls the width of the bumps. The smaller the σ, the more complicated the model is. As the model gets more complicated, it will start to overfit and training error goes up. On the other side, for larger σ, we will have higher training error but lower approxiamation error.