Programming 2

Samuel Cuthbertson SAMUEL.CUTHBERTSON@COLORADO.EDU

1. Analysis

1.1. Implemented Formulas

In `sg_update` I implemented the formula

$$\beta' = \beta + x_i(y_i - \pi_i)\lambda$$

Where x_i is the feature vector, y_i is the class for x_i (1 or -1), π_i is the prediction for class of x_i, β' is the old beta values vector, and λ is the learning rate. This formula is the derivative from the probability formula below (see slides 7 and 8 here), and optimizes beta values to find the most predictive features in x. My implementation can be found in lines 113 and 114 of my `logreg.py`.

Also in `sg_update`, I implemented the formula

$$\pi_i = \sigma(\beta \cdot x_i)$$

Where π_i is the prediction for class of x_i, and $\sigma(w)$ is the sigmoid function $\frac{e^w}{1+e^w}$. β and x_i are the same as above, and $\sigma(w)$ takes the dot product of them. This returns a number between -1 and 1 that is the prediction for the class of x_i. I used the sigmoid function that was already defined in the program, and called it on line 112.

Lastly, for regularization, I implemented

$$\beta'_j = \beta_j(1 - 2\lambda\mu)^{m_j}$$

Where μ is the regularization parameter, j is the current beta value being updated, and m_j is the number of `sg_update` calls since β_j has been updated. The portion $(1 - 2\lambda\mu)$ is the "shrinkage", and is used to update β_0 on line 117. The entire formula is used on beta parameters on line 121. The definition of this formula can be found on slide 8 here.

1.2. Role of the Learning Rate/Number of Passes

Learning rate simply effects the size of updates to β parameters. A high learning rate will change βs a large amount with every update, while a small learning rate will change βs a small amount each update. A small learning rate will take longer to converge, while a large learning rate will converge quickly but to a less accurate value. The most accurate setup is a small learning rate with a high number of passes, but the quickest setup is a large learning rate with a small number of passes. However, to many passes can also lead to over-fitting through updating the β parameters to fit the training data too well, and eliminating the ability to classify unseen data.

1.3. Best/Worst Predictors

The best and worst words for predicting class were found by finding the features with highest and lowest β values, respectively. They are found in the tables below.

Best Predictors	Beta Values
hockey	-1.851942326
runs	1.1462635
playoffs	-1.270940682
hit	1.207308297
pick	-0.9959418613
bat	0.9299477078
playoff	-0.9045412949
points	-0.9012288343
saves	0.9003541079
period	-0.896220583

Worst Predictors	Beta Values
everywhere	0
bloody	0
blasted	0
memoriam	0
rode	0
tone	0
deceased	0
intermissions	0
pitiful	0
vintage	0

It's worth noting that there were an additional 20 to 30 more features with beta values of 0, indicating that there were quite a few features tied for least predictive.

1.4. Regularization when μ is 0

When μ is 0, there is no regularization. The "shrinkage" equation changes from $(1 - 2\lambda\mu)$ to simply 1, and therefor doesn't effect β_j's value.