Machine Learning Fall 2016 Programming Assignment 2

Alex Ring

1. Functions Implemented

The beta update function that I implemented was : $\beta[j] = \beta[j]' + \lambda(y_i - \pi_i)x_i$ where $\pi = \frac{exp(\beta^T x_i)}{1+exp(\beta^t x_i)}$. The first step in my update is to index all of the non-zero values that we are passed from train example. I use this index to select values to update from the beta and x vectors respectively. Numpy allows us to index into their array objects with another array, this lets us update and select without a for loop, speeding up computation. I set the P variable (representing π in the equation) and then set the step using the provided step function. Delta is set according to the beta update function above. I then go through and update the beta values for the entire vector. I decided to do this update before regularization because we don't always regularize and this saves computation in that case. The if statement following the update checks if μ is greater than 0. If μ is zero or less regularization does not make any sense, so we can skip the computation. The regularization itself happens in the for loop, I calculate shrinkage based on the λ and μ. I then use the last update variable to check the last time we updated to set my exponent correctly (m in the equation) and finally set the beta parameters with a second update using shrinkage and gap. In total the regularization update is: $1 - (2\lambda\mu)^{m_j}$

2. What is the role of the learning rate?

The learning rate determines how far you are going to "step" in the gradient descent. Graphically it is how far down the curve you move from the previous iteration. Having a learning rate that is too large will result no convergence, a small learning rate takes long to converge. In short, the learning rate determines how fast we are converging towards optimal weights and how much the beta value is updated each pass.

3. How many passes over the data do you need to complete?

Three passes provided a 99% accuracy on training data and a 94% accuracy on test data. I would consider this acceptable. Results in table 1.

Table 1. Passes over data

Passes	TP	HP	TA	HA
1	-112.590763	-24.318733	0.964286	0.932331
2	-68.103173	-19.901148	0.989662	0.939850
3	-52.946700	-19.408414	0.996241	0.947368

Table 2. Good and bad predictors

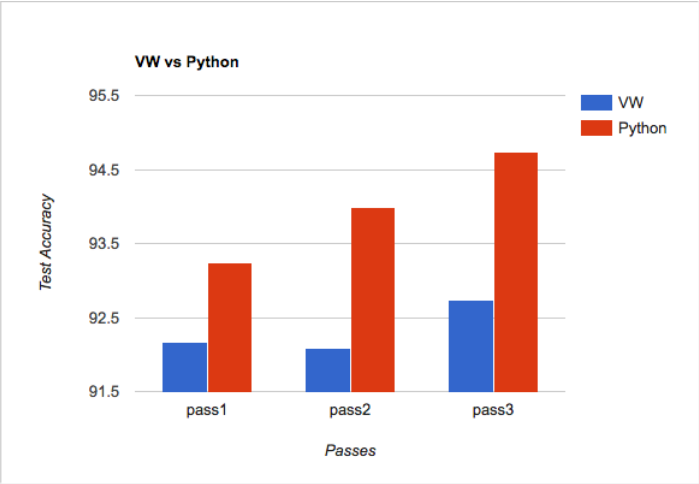
	1	2	3	4	5
Baseball	Pitching	Saves	Bat	Runs	Hit
Hockey	Points	Playoff	Pick	Playoffs	Hockey
Worst	Racist	Bloody	Blasted	Hooked	Intermissions

4. What words are good and bad predictors?

Mathematically logistic regression pays more heed to words associated with larger beta weights. Essentially, $Class = \beta_1 x_1 + \beta_2 x_2 + ...\beta_n x_n$ Larger beta values in the positive direction lead to a positive classification, and larger (negative) values in the negative direction lead to negative classification. Our data classifies baseball as positive and hockey as negative. We can easily find the best words for baseball by sorting the beta vector in ascending weight order and looking at the highest values. You can do the same to find the best hockey words by looking at the lowest values. Mathematically the worst predictors of both classes are going to be the words that have beta weights of zero associated with them. A given $\beta_i x_i$ where β = 0 will result in 0, causing no change in the calculation. Words that do this are the worst words for classification. You can find 0 values in the beta vector and the words associated with them. My results appear in table 2.

5. What happens to regularization if mu is 0?

In Logistic regression the regularization part of the function is: $1 - (2\lambda\mu)^{m_j}$ here we can see that if μ is equal to zero, the entire center part is going to be zero. So the function would simplify to one. (1 - 0) This would have no affect on the beta update because we would just be multiplying the beta parameter by one. Neither shrinking or enlarging it.



6. Extra Credit Vowpal Wabbit vs. Python Implementation

I decided to compare the accuracy on test data with different numbers of passes over the data using both VW and this Python implementation. Overall Python preformed better and had a greater increase to accuracy with more passes.