CS 6780: Advanced Machine Learning **(Submitted on: 3/6/2019)**

Submission Assignment #2

Instructor: Thorsten Joachims *Name:* Molly Ingram, Julien Neves*, Netid:* msi34, jmn252

Problem 1

(a)
(b)

Problem 2

(a)
(b)
(c)

Problem 3

(a) The batch algorithm we are minimizing is

$$F(\mathbf{w_i}) = \sum_{j=1}^{n} \max(0, -y_j(\mathbf{w_i} \cdot \mathbf{x_j}))$$

The global minimum, 0, is achieved when all n data points are classified correctly, i.e. $y_j(\mathbf{w_i} \cdot x_j) \geq 0$.

So the gradient is $\nabla_{\mathbf{w}} F(\mathbf{w_i}) = \sum_{j \in E} y_j \mathbf{x_j}$ where E is the set of all misclassified data points ($y_j(\mathbf{w_i} \cdot x_j) \leq 0$). Note the that loop in line 5 uses line 7 to sum over the misclassified points, so this algorithm is computing the gradient at $\mathbf{w_i}$.

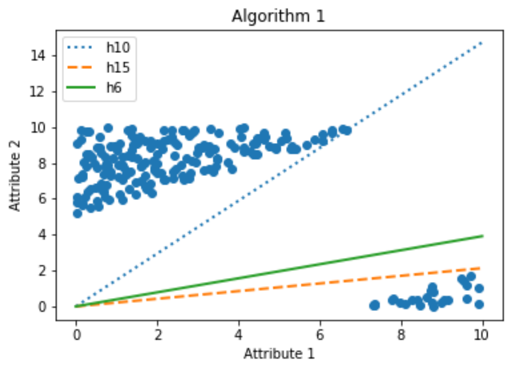
(b) Note the max element is only non-zero if $1 - y_j(\mathbf{w} \cdot \mathbf{x_j}) \geq 0 \leftrightarrow 1 \geq y_j(\mathbf{w} \cdot \mathbf{x_j})$. Let E be that set of all data points such that this non-zero condition is satisfied. Note this set includes misclassified points *and* correctly classified points that are close to the hyperplane. Taking the partial derivate of F wrt the i^{th} element of $\mathbf{w}$ we have

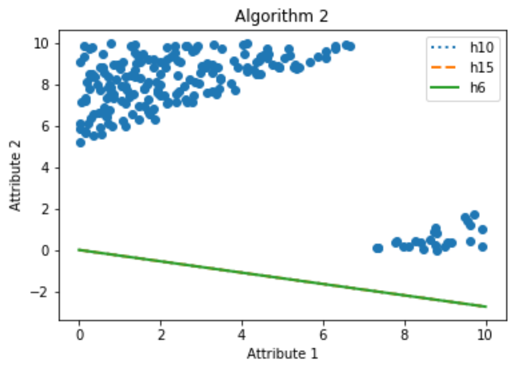
$$F_i(\mathbf{w}) = w_i - C \sum_{j \in E} y_j x_{ji}$$

Stacking the partials, we can rewrite in the vector form $\nabla_{\mathbf{w}} F(\mathbf{w}) = \mathbf{w} - C \sum_{j \in E} y_j \mathbf{x_j}$.

The algorithm is updated as i) $\mathbf{w_i}$ where i indicates the iteration; ii) $1 \geq y_j(\mathbf{w_i} \cdot \mathbf{x_j}$; iii) $\Delta \mathbf{w_{temp}} - C y_j \mathbf{x_j}$

(c) The difference in the update condition is that algorithm 1 only uses points incorrectly classified and algorithm 2 uses misclassified points *and* correctly classified points that are close to the hyperplane. So algorithm 2 includes more in examples in an update. The order of the dataset will not affect the algorithms because they loop over all points when they update, not update one point at a time like the perceptron algorithm we discussed in lecture.





(d)

I have plotted only some of the hyperplanes with different α values for clarity. We can see from the scatter plots that algorithm 1 is more sensitive to α and that it allows the hyperplane to come closer to the examples. Algorithm 2 is less sensitive to α, the hyperplane essentially overlap, and perhaps because of the margins and the few number of points in the bottom right the hyperplane lumps all points in the same classification. ...It also could be I made an error in the algorithm because this doesn't seem right...

The table of weight vectors, shown in the jupyter notebook along with the code, also indicates the algorithm 2 is much less sensitive to α.

(e) On the unsorted examples, the number of updates was 93 and the number of passes was 78. On the sorted examples, The number of updates was 441 and the number of passes was 397. Because the algorithm runs over the examples in order until a separating hyperplane is found, the sorted data groups together those which should not be separated so we have to cover more examples before finding an example that is classifed differently. While in the unsorted data, the algorithm gets to examples classified differently much earlier in looping over the examples.