CS446: Machine Learning **Fall 2015**

Problem Set HW 3

Shivam Upadhyay *Handed In: 10/9/2015*

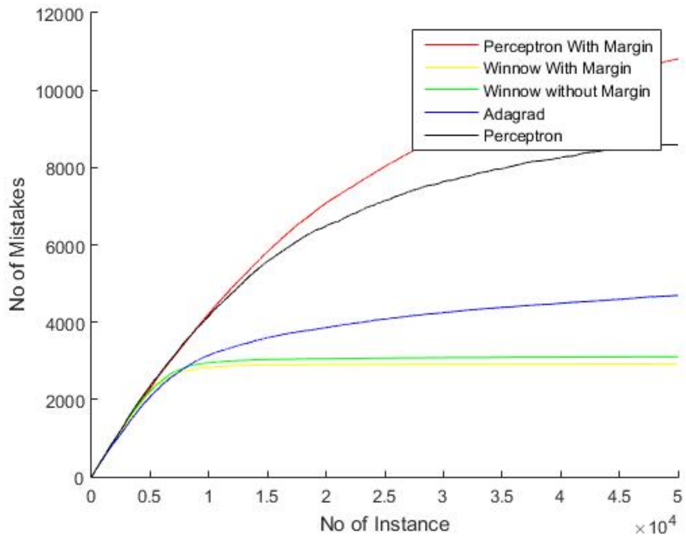
1. The accuracy and parameter values for the given on-line setting is as following

Algorithm	Parameters	Dataset n=500	Dataset n=1000
Perceptron	LearningRate =1	99.99	98.97
Perceptron w/margin	0.03,0.005	91.42	87.62
Winnow	1.1 ,1.1	99.98	99.98
Winnow w/margin	Margin =2.0,0.04 Learning-Rate = 1.1, 1.1	100	99.86
AdaGrad	0.25,0.25	98.08	99.25

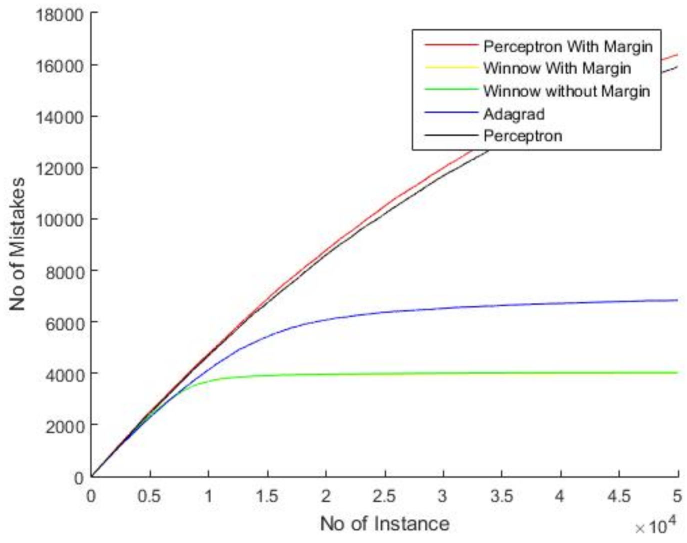
Observations

Winnow algorithm was best performing algorithm amongst the set of given algorithms when data was noise free as can be seen from the table. Winnow algorithm always bounds the number of mistakes of order k(log(n) where k is the number of disjunctions to be learnt in the hypothesis space of n dimension. Thus, given no restriction of time taken to converge winnow will bounded number of mistakes which can be seen in the figures as well. Both winnow without margin and winnow with margin show identical behaviour and have bonded number of mistake which is affirmed by the fact that both curves flattens out after seeing around 10000 examples which proves that winnow is a mistake bound algorithm. As the function given was l of m of n, i.e. if at least l out of m values are active than label is positive or negative otherwise. Perceptron and Perceptron with margin are additive algorithms and for the sparse feature space such as the one multiplicative algorithms such as winnow and winnow with margin will have a definite advantage over additive algorithms in terms of bounding the number of mistakes as number of mistakes made by multiplicative algorithms depends upon the number of relevant features which is smaller than n which govenrs the mistake bound of additive algorithms (O(kn)). Same holds for AdaGrad as well. It can also be seen that how doubling the total number N of variables has very little effect on Winnow, when the number k of relevant variables is kept as a small constant. The Perceptron algorithm suffers much more from such a doubling.Thus, the bias of the Perceptron algorithm does not allow it to take advantage of the number of relevant variables being small. The lower bounds actually apply to a very general class of additive algorithms.

In contrast, it is known that the number of mistakes Winnow makes is linear in the number of relevant variables, but only logarithmic in the number of irrelevant variables.



Data set n =500



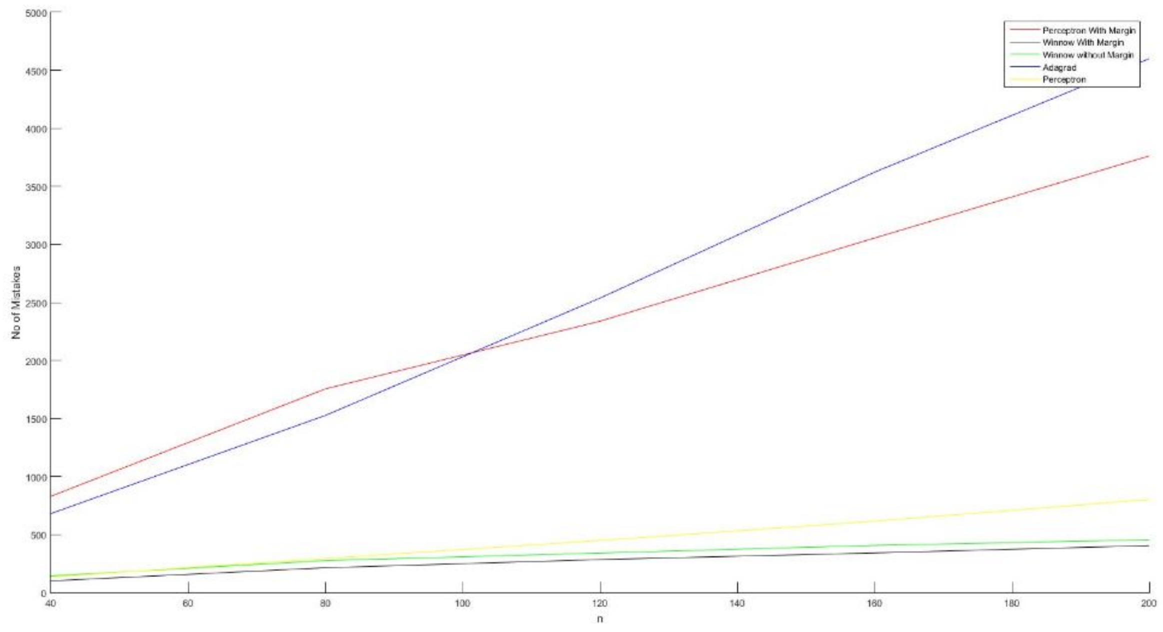
Data set n =1000

The number of mistakes for various values of n is captured in the below table

Algorithm	Parameters	n=40	n=80	n=120	n=160	n=200
Perceptron	1	136	294	449	616	802
Perceptron w/margin	0.03,0.25	829	1755	2338	3055	3762
Winnow	1.1	144	276	341	407	455
Winnow w/margin	margin=2 and learningrate = 1.1	101	214	285	342	405
AdaGrad	1.5	680	1526	2537	3622	4600

2. **Observations**

   Similar to the observation in first part of the assignment winnow algorithms doesn't made much mistakes before converging while perceptron and AdaGrad made large number of mistakes. Adaptive algorithms have better performance when gradient vectors are spares. Most important observation is that perceptron converged which indicates that **DATA IS LINEARLY SEPARABLE** i.e. there exists a weight vector which separates the given set of data another thing to notice is the choice of learning rate for perceptron doesn't affect its converges as learning rate just scales the value of W i.e it can expedite the process of convergence of but wont cause the convergence. And for all the algorithms the number of mistakes were linearly increasing with that of increase in number of dimension of the Boolean vector keeping the hypothesis same, but after few iterations winnow algorithm converged while perceptron kept on making mistake, which further substantiates the mistake bound order of additive and multiplicative algorithms.Further, When the data are sparse and features have different frequencies, a single learning rate for every weight update can have exponential regret. AdaGrad chooses learning rate dynamically by adapting to the data. AdaGrad calculates a different learning rate for every feature. Unlike winnow and perceptron where we fixed the learning rate. One plausible reason for the performance of Adagrad maybe the initial learning rate, as it is very sensitive to the initila learning rate.



3. The problem required us to switch to Batch mode from the on-line mode. The major difference lies in updating the weight vector or threshold value. On-line mistake driven algorithm starts from one hypothesis and on every mistake updates the value of weight vector prior to encountering the next example. This method is particularly useful for processing the stream of data. While batch processing keep track of the mistakes made by saving the updates to be applied later i.e. in batch mode parameters are update per epoch as compare to per instance updation of values in on-line algorithms. Though, both on-line and batch processes algorithms converge to same local minima but the number of updates made in both of them is entirely different. In euclidean plane the path covered by both of these algorithms will be entirely different. So they differ in performance. In the given problem parameters l,n are kept constant while varying the value of m. Thus, in effect increasing the number of relevant attributes. Training data was noisy and the experimental results were as following:

Algorithm	m=100		m=500		m=1000	
	acc.	params.	acc.	params.	acc.	params.
Perceptron	1	53.54	1	52.62	1	67.30
Perceptron w/margin	1.5	50	1.5	50	0.03	50.10
Winnow	1.1	50	1.1	50	1.1	50
Winnow w/margin	2,1.1	50	0.006, 1.1	50.54	2,1.1	50
AdaGrad	1.5	50	1.5	50	1.5	57.26

Batch results

I tried for both batch and on-line algorithm for the given data set. For batch algorithm the accuracy was around 50 percent for all the algorithms if number of iterations were kept 20. I tweaked the number of iterations to 50 for all the algorithms for three parameters only the perceptron without margin got to the accuracy of 51 %. Beyond that I could only gather the data of perceptron without margin for no of iteration = 200. The values were 87 % , 67.23 % , 78.99% for three parameters.But beyond that, I was not able to calculate rest of the values as computation was too slow or halted because of the limited capability. I therefore took observations for the noisy set with that of Batch as well as on-line as well. Beacuse I couldn't garner much insight from the results obtained from batch updating the results.

Algorithm	m=100		m=500		m=1000	
	acc.	params.	acc.	params.	acc.	params.
Perceptron	1	65.34	1	54.90	1	50
Perceptron w/margin	0.03	59.47	0.25	50.63	0.03	73.84
Winnow	1.1	88.39	1.1	50.54	1.1	78.59
Winnow w/margin	2,1.1	93.47	0.006,1.1	92.33	1,1.1	78.55
AdaGrad	0.25	78.78	0.25	58.07	1.5	82.62

On-line results

It is clear from the above table that Winnow has more utility when it comes to finding the k-disjunctions in a lof m of n kind of data as per the discussion above. AdaGrad has more applicability in convex problems, similarly additive algorithms perceptron and perceptron works well where irrelevant attributes are fewer in number and in case of linear separable data will always converge.Thus, the bias of the Perceptron algorithm does not allow it to take advantage of the number of relevant variables being small. The lower bounds actually apply to a very general class of additive algorithms. In contrast, it is known that the number of mistakes Winnow makes is linear in the number of relevant variables, but only logarithmic in the number of irrelevant variables.Simple eperiments show that this effect also occurs outside of our worst-case analysis:On seemingly natural random data, the Perceptron algorithm suffers from additional irrelevant variables much more than Winnow. Another feature of the worst-case bounds is that the Perceptron algorithm can take advantage of sparse instances: If only few variables are active in the input at any given time but most of the variables are relevant, the Perceptron algorithm will outperform Winnow.