CS446: Machine Learning **Fall 2014**

Problem Set 3

Official Solution *Handed In: October 10, 2014*

The solution presented below, including all the figures and plots, is based on the TA's solution. As you can see in your own implementation, there is some variation in the results due to randomness in the data generation. However, we expect that the key trends and observations will remain the same.

1. [Number of examples versus number of mistakes - 20 points]

Algorithm	Parameters	Data Set	
		$n = 500$	$n = 1000$
Perceptron w/margin	η	0.005	0.005
Winnow	α	1.1	1.1
Winnow w/margin	α	1.1	1.1
	γ	2	2
AdaGrad	η	0.25	0.25

The plots of the cumulative mistakes are given Figure 1 and Figure 2, with $n = 500, 1000$. You can see that for both Perceptrons the number of mistakes goes up linearly with n, while for the Winnows the growth seems logarithmic as a function of n. AdaGrad seems closer to Winnow, but does not scale as well.

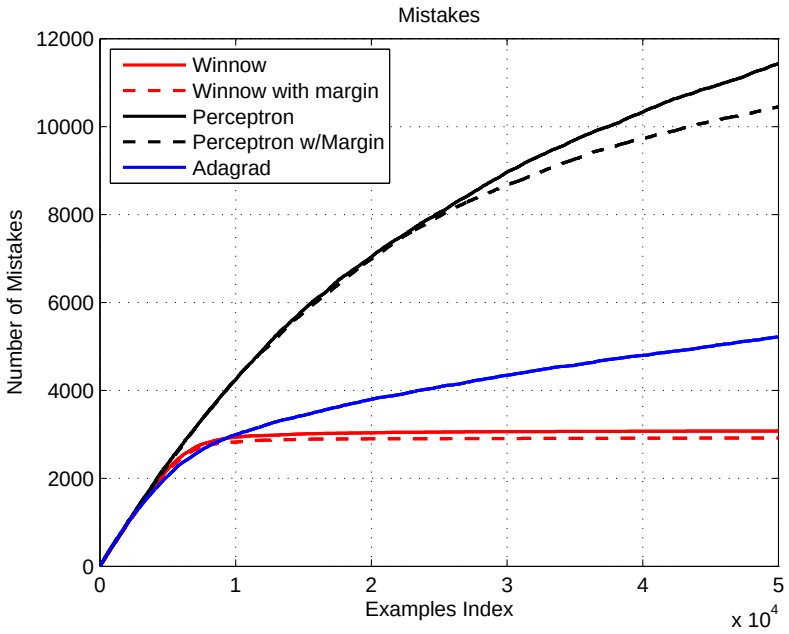

Figure 1: Mistake plot (n=500)

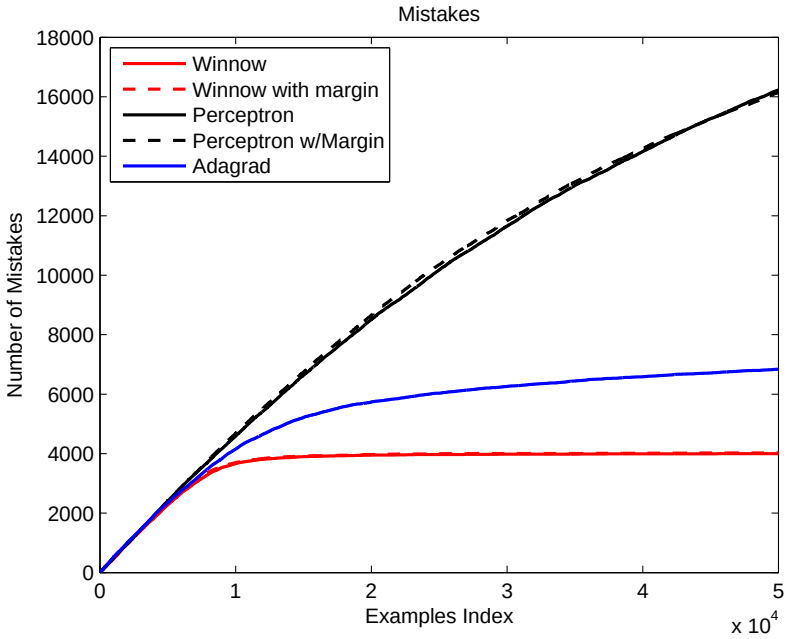

Figure 2: Mistake plot (n=1000)

2. [Learning curves of online learning algorithms - 35 points]

Algorithm	Parameters	Data Set				
		$n = 40$	$n = 80$	$n = 120$	$n = 160$	$n = 200$
Perceptron w/margin	η	1.5	0.03	0.25	0.25	0.25
Winnow	α	1.1	1.1	1.1	1.1	1.1
Winnow w/margin	α	1.1	1.1	1.1	1.1	1.1
	γ	2	2	2	2	2
AdaGrad	η	1.5	0.25	1.5	1.5	1.5

The learning curves are as shown in Figure 3

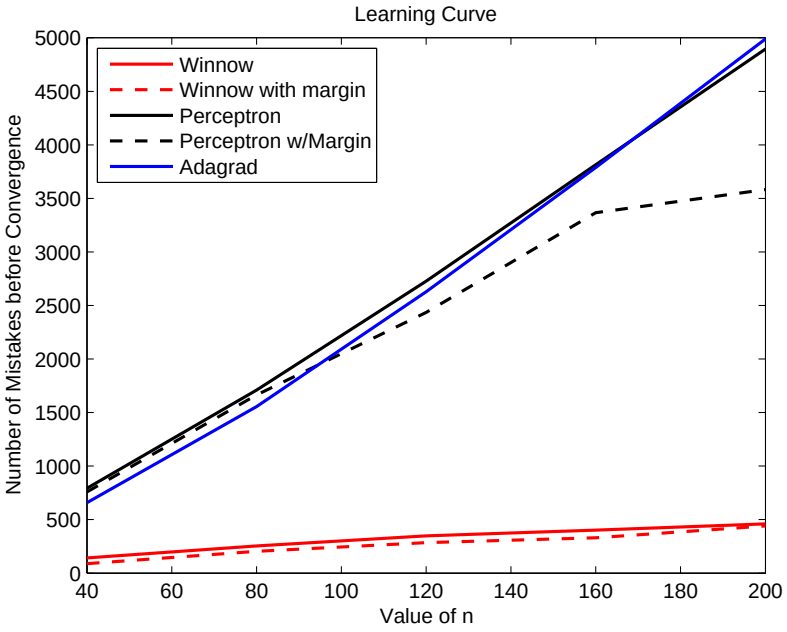

Figure 3: Learning curves of different algorithms

Comments: You are expected to find that Winnow and Winnow with margin always make the fewest errors before convergence. The Winnow with margin should be a little bit better than Winnow, but the two are pretty close. It's OK if in your plot the difference is not discernible. Perceptron with margin should be better than the original perceptron, but they can also be close. AdaGrad is more sensitive to the data generation; while it can be better or worse than the two perceptrons, it is always worse than the Winnows. Also note that the number of mistakes should always go up as n increases since our problem is harder as n is larger. However, you might see some variability due to the randomness of the data generation, especially for AdaGrad. You will get the credit if the difference between Winnow, Perceptron and AdaGrad is reasonable.

3. [Use online learning algorithms as batch learning algorithms - 45 points]

Algorithm	Parm & Accy	Data Set ($l = 10$, $n = 1000$)		
		$m = 100$	$m = 500$	$m = 1000$
Perceptron	Accy %	90.2	82.7	76.3
Perceptron w/margin	η	0.03	0.25	1.5
	Accy%	97.1	83.1	77.2
Winnow	α	1.1	1.1	1.1
	Accy%	97.4	76.4	74.3
Winnow w/margin	α	1.1	1.1	1.1
	γ	0.3	0.006	0.04
	Accy%	93.7	86.8	71.8
AdaGrad	η	0.25	1.5	1.5
	Accy%	98.4	71.2	77.3

Comments: A few observations: (1) The best parameters are not so sensitive to changes in m, but the accuracy is sensitive to changes in m, and generally goes down with the increase in m. (2) The fact that we train on noisy data has significant impact on the performance of all algorithms. (3) Notice that in this case, as m grows, the target function becomes dense; almost all the features become important; this explains the fact that, unlike the previous experiments, here Perceptron is doing better than Winnow.

4. [Bonus - 10 points]

   For the bonus question, we use an update rule that is similar to the regular perceptron, except that the γ we choose to compare $y(w^\intercal + \theta)$ to is dependent on the true label y: we compare to γ_{-1} if $y = -1$, and γ_{+1} if $y = +1$.

   To validate this empirically, we reused the experimental protocol with configuration P3 for question 3 ($l = 10$, $m = 1000$). We chose P3 because the concept that configuration represents is the hardest. Obviously, we should use $\gamma_{+1} > \gamma_{-1}$, so that more updates are done for the positive examples. We choose learning rate $\eta \in \{0.03, 0.005, 0.001\}$ $\gamma_{+1} \in \{2, 0.3, 0.04\}$ and $\gamma_{-1} \in \{0.001, 0.006\}$ and tune the parameters as we have done in the previous sections.

   The results can be seen in the table below. By comparing the overall accuracy values along with the accuracy values on positive and negative examples (Pos Accy% and Neg Accy% in the table) of the modified Perceptron with the original perceptron algorithm, we find that while the original perceptron is quite robust, the modified version is still doing better.

Algorithm	Parameters	Accy%	Pos Accy%	Neg Accy%
Perceptron w/unbalanced margin	$\eta = 0.001$, $\gamma_{+1} = 0.04$, $\gamma_{-1} = 0.001$	99.59	100	99.54
Perceptron	NA	99.16	98.10	99.28