CS446: Machine Learning **Spring 2017**

Problem Set 3

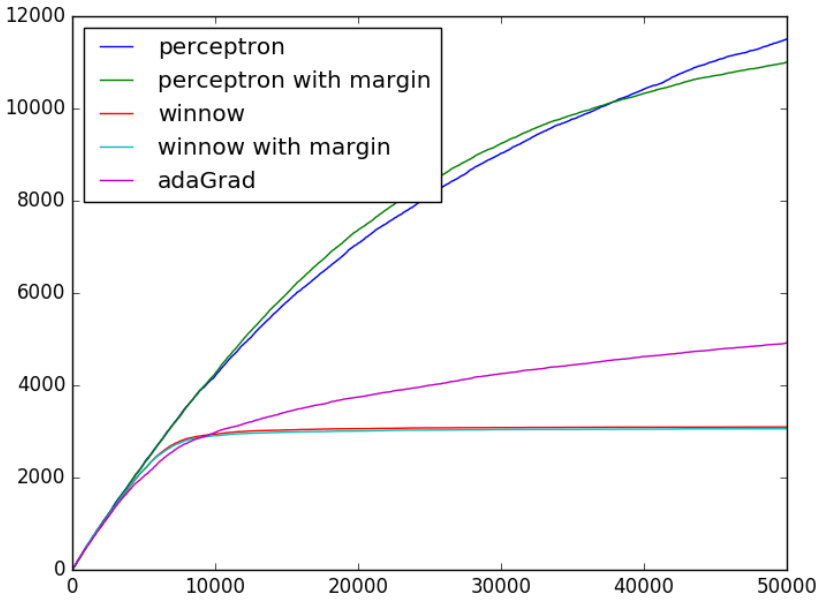
Kaizhao Liang(kl2) *Handed In: February 19, 2017*

1. **Number of examples versus number of mistakes**

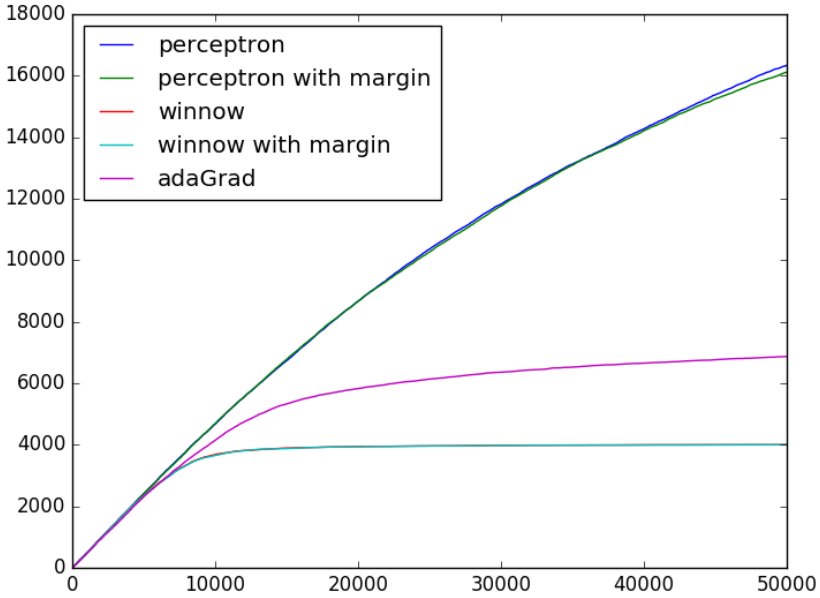
Algorithm	*Parameters*	*Dataset* $n = 500$	*Dataset* $n = 1000$
Perceptron	η	1	1
Perceptron w/margin	η	0.005	0.03
Winnow	α	1.1	1.1
Winnow w/margin	α and γ	α=1.1,γ=0.04	α=1.1,γ=0.3
AdaGrad	η	1.5	0.25

Note that:
In the below graphs the red curve for the winnow and the light blue curve for the winnow with margin blend together, if you zoom in you could see the red one overlapses the light blue one.
In the second graph, the curves for the perceptron and perceptron with margin also blend together.

Dataset n=500



Dataset n=1000



Observation:
By simple observation, on both data sets, the order of number of mistakes required to converge, from large to small, is perceptron, perceptron with margin, AdaGrad, winnow and winnow with margin.
At the beginning, before 10000 training samples, all of the algorithms behave similarly. The winnow and the winnow with margin behave very similarly and both basically stop making mistakes after 10000 training samples, while the other algorithms continue to make more mistakes.
On the dataset n=500, the perceptron with margin makes less mistakes after 40000 samples than does the original perceptron. However, on the dataset n=1000, they behave almost identically.
For the winnows, the numbers of mistakes made and the timing of convergence for winnows are approximately the same in two plots. Yet the number of mistakes made by AdaGrad and perceptrons increases dramatically in second plot. The timing of convergence of these algorithms is also delayed in the second plot.

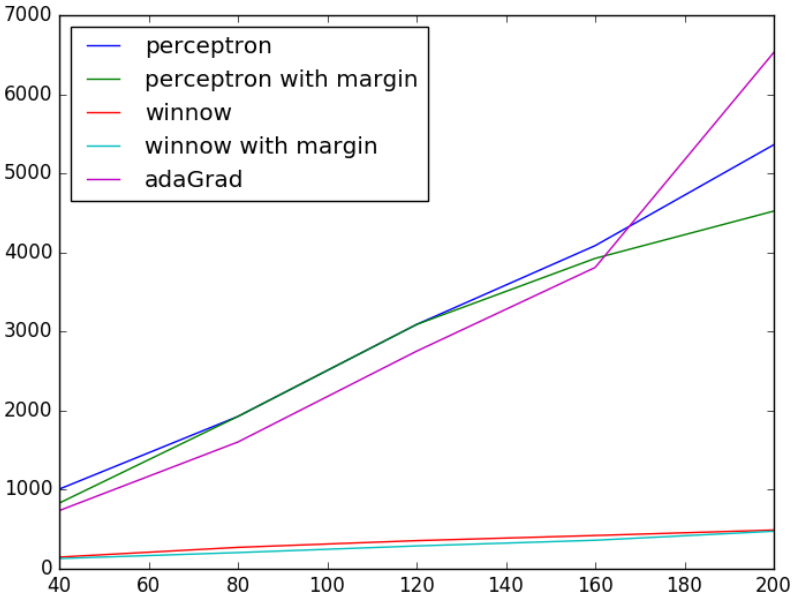
Analysis and Explanation:
Given that the target function $\|u\|$ is sparse in both cases(100 relevant features out of 500 or 1000), the multiplicative algorithms, such as winnow, have advantages over additive algorithms, such as perceptron and AdaGrad, in both converging speed and number of mistakes made. So it's understandable that winnow with or without margin converges faster and stops making mistakes earlier than do the other algorithms.
Increasing the sparseness of the target function does not make a great difference to the winnows' converging speed and the number of mistakes made upon convergence. However, it increases the number of mistakes made by the multiplicative algorithms and slows down the convergence of those algorithms, thus confirming the fact that the

multiplicative algorithm has advantages in learning sparse functions.
The AdaGrad keeps track on very feature's gradient and thereby focuses on the features that are inactive. It updates the weight faster than does the perceptron. So it is reasonable that it converges faster than does the perceptron.
The margin clearly helps the perceptron to learn by forcing it to update more frequently and to learn a hyperplane with thickness. So, the perceptron with margin starts to converge in the first plot after seeing 40000 samples, while the perceptron keeps making mistakes in almost the same rate. However, on the dataset n=1000, the target function could be too sparse for the margin's influence to be perceivable and 50000 samples are simply not enough for the perceptron to converge.

2. **Learning curves of online learning algorithm**

Algorithm	*Parameters*	n=40	n=80	n=120	n=160	n=200
Perceptron	η	1	1	1	1	1
Perceptron w/margin	η	1.5	0.25	0.03	0.03	0.03
Winnow	α	1.1	1.1	1.1	1.1	1.1
Winnow w/margin	(α,γ)	(1.1,2.0)	(1.1,0.3)	(1.1,0.3)	(1.1,0.3)	(1.1,2.0)
AdaGrad	η	1.5	1.5	1.5	1.5	1.5

Learning curves for five algorithms
Note that
in the below graph the red curve for the winnow and the blue curve for the winnow with margin blend together, if you zoom in you could see the red one overlapses the blue one.



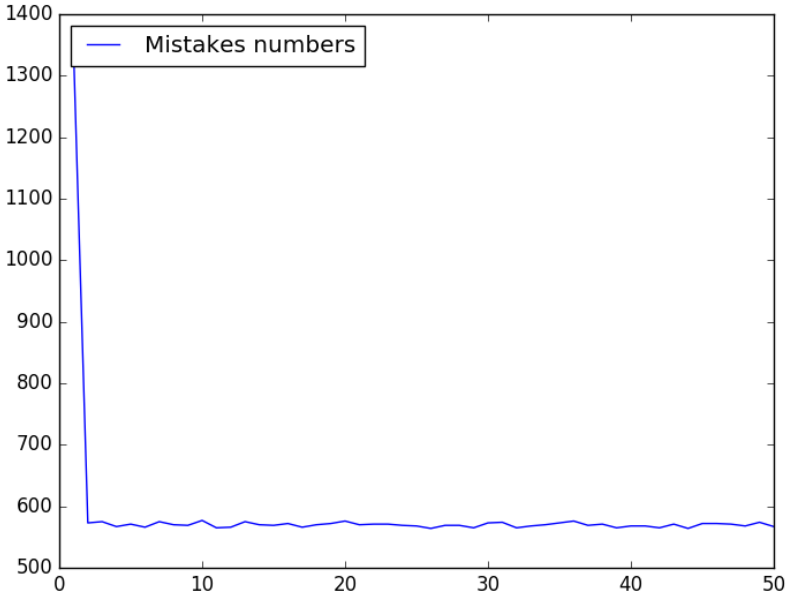
Observation
Generally, as the n grows, the number of mistakes made upon convergence grows.However, the winnows' performance on different ns is more stable and the numbers of mistakes made by those algorithms upon convergence only change slightly.
The order of mistakes required to converge for different algorithms, from large to small is perceptron, perceptron with margin, adaGrad, winnow and winnow with margin.
Analysis and Explanation
From the result, it can concluded that the sparseness of the target function does not influence the performance of the winnows' as much as it does to the perceptrons and the AdaGrad.
Also the target function is still sparse (only half of the features are relevant at most), which explains why the winnows make least mistakes, since the multiplicative algorithms have advantages in learning sparse functinos.
However, it's mysterious that the adaGrad needs more mistakes to converge than does the perceptrons when n =200 . Because normally speaking, the adaGrad updates the weight faster than does the perceptron, thus converges faster.

3. **Use online learning algorithms as batch learning algorithms**

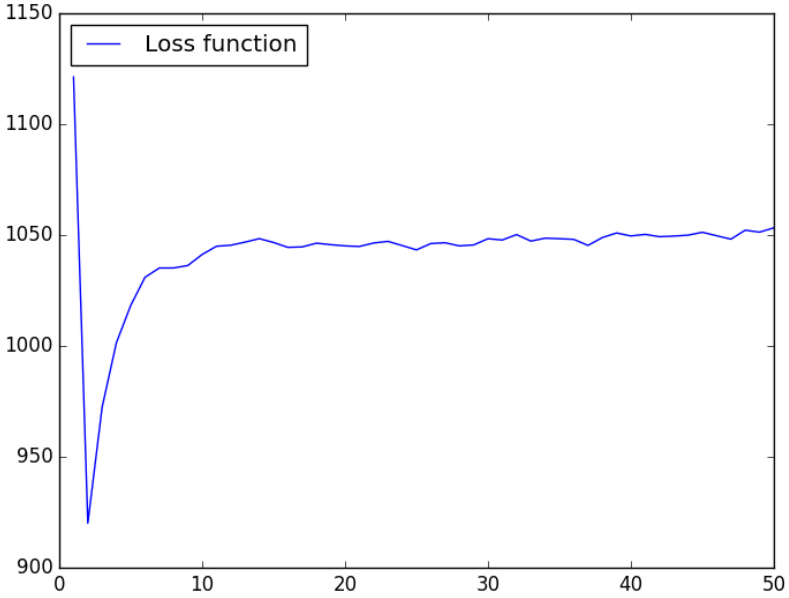
Algorithm	m=100		m=500		m=1000	
	acc.	params.	acc.	params.	acc.	params.
Perceptron(η)	0.97	1	0.93	1	0.78	1
Perceptron w/margin(η)	0.99	0.001	0.96	0.25	0.90	0.03
Winnow(α)	0.98	1.1	0.90	1.1	0.80	1.1
Winnow w/margin(α,γ)	0.98	(1.1,0.001)	0.88	(1.1,0.3)	0.79	(1.1,0.3)
AdaGrad(η)	0.97	1.5	0.99	1.5	0.88	1.5

Obeservation and Explanation
When the target function is sparse, five algoriths perform similarly. As the density of the target function grows, the accuracy on test sets declines in general.
AdaGrad and perceptron with margin appear to be more resilient to the growth of the density of the target function, while winnows and original perceptron's performance decline dramatically. So AdaGrad and perceptron appear to be the better choice of batch online learning algorithm.
Different from the previous experiments, the training data for this experiment contains noise. As the number of relevant features increases, the influence of the noise grows. While the previous experiments indicate that the winnow is a better algorithm since it needs less mistakes, i.e. samples to converge, this experiment implies that winnow is not as robust as the perceptron and adaGrad when the training data is not ideal but noisy.

4. **Bouns**
Mistakes



Hinge Loss



Observation and Explanation

The number of mistakes made on each round drops to and stays at 580 or so and only fluctuates only slightly. For the hinge loss, at first it drops and then grows back to 1050 or so, fluctuating only slightly afterward. So it firstly learns the linear separator that fits the data set tightly, or in other word, overfits the data set, because the hinge loss is very low and the learning rate is very high initially. Then it continues to find the best linear separator with a proper margin, when the hinge loss grows back and the

learning rate diminishes, but the number of mistakes made stays low. Once it learns the best separator, the hinge loss stops to grow and stays stable.