# Machine Learning Assignment3

Xiaoxu Gao
highsmallxu@gmail.com

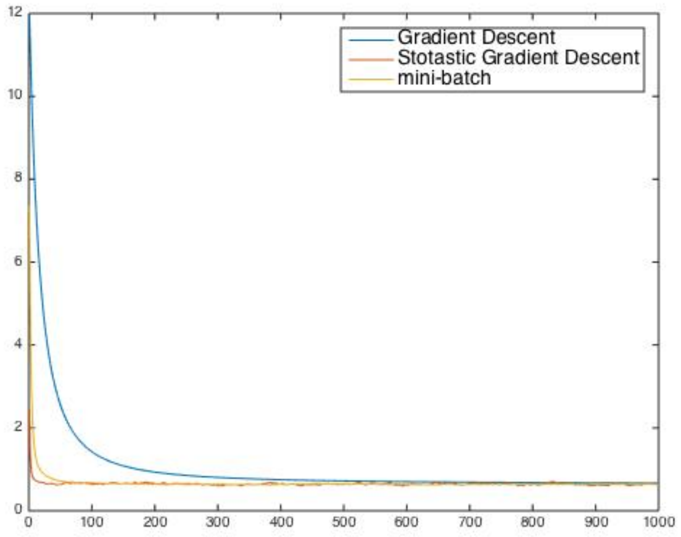
## Question a

The loss funtion is $f(x) = x^4 - 2x^2 + 2x$, and the goal of this question is to find a $w = argmin_w f(w)$ by three GD version.

```
MATLAB Code:

alpha = 0.001; %learning rate
N = 50; %number of samples
w = 2;   %initial weight
x = normrnd(0,2,[N,1]); %samples
cost = [];
n = randi([1,N],[1,1]);
M = 5;
m = randi([1,N],[M,1]);

for epoch = 1:1000
    y = w.^4-2*w.^2+2*w;
    cost1 = [cost1;y];
    grad = 4*w.^3-4*w+2 + x;
    w_new = w - (alpha/N)*sum(grad);  %Gradient Descent
    w_new = w - (alpha)*sum(grad(h(t))); %SGD
    w_new = w - (alpha/n)*sum(grad(mini_batch(t,:))); %mini-batch=5
    w = w_new;
    epoch = epoch+1;
end
```



**Result:**
In this question, there are 50 samples in the dataset and 1000 epoches totally. From the above figure, we can see that SGD converges much faster compared to GD and mini-batches GD, but the cost function is not as well minimized as in the case of GD. It converges with some vibration. However, the minimum costs that we got from 3 variances are close enough that we can ingore the tiny difference.

## Question b

In this question, I choose the first four iterations of each method.The results can been seen from the following figures. It's obvious that in fig2, SGD converges much faster because it's based on only one sample. Besides, due to its stochastic nature, the path towards the minimum cost is not as direct as in GD. GD with mini-batch is something between GD and SGD.

**iteration=4**

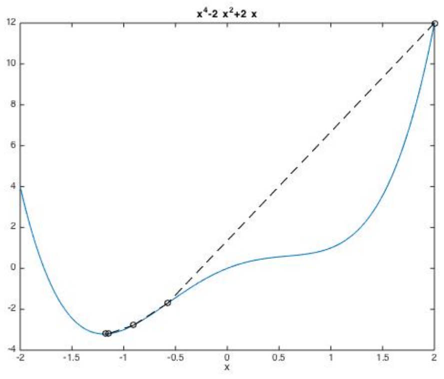


*Fig1. Gradient Descent*

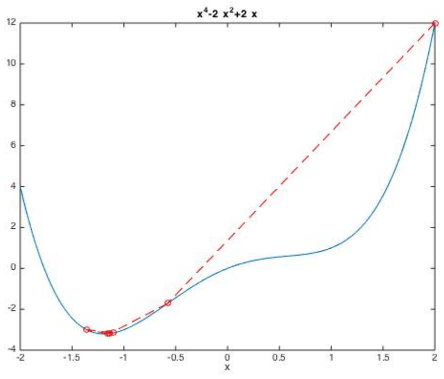


*Fig2. Stochastic GD*

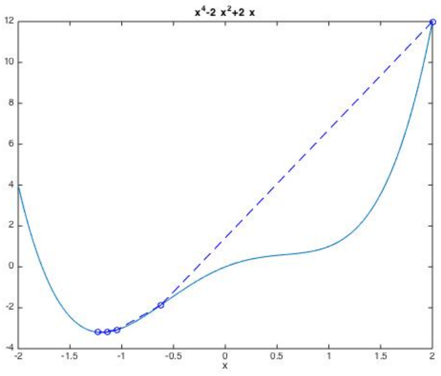
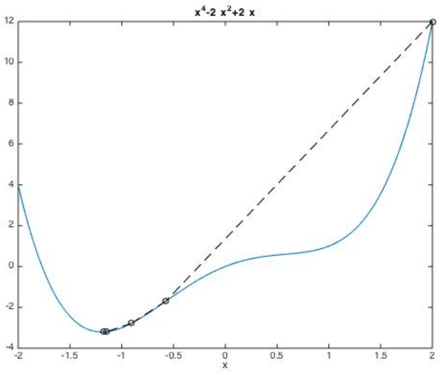


*Fig3. GD mini-batch*

**iteration=50**



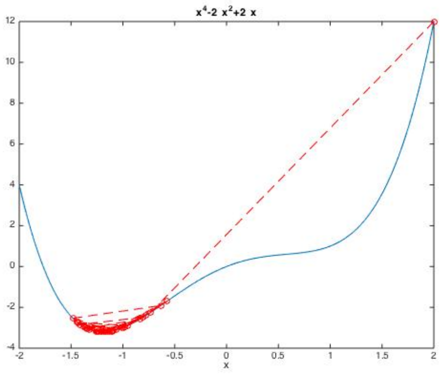


*Fig1. Gradient Descent* *Fig2. Stochastic GD*

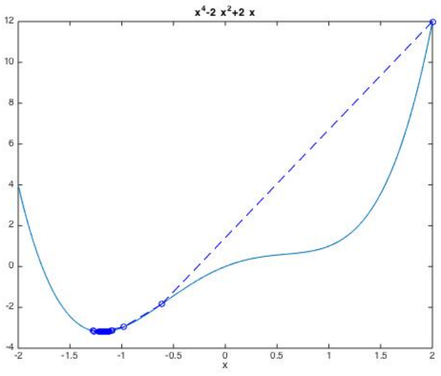
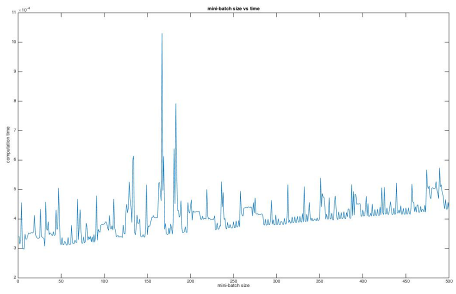
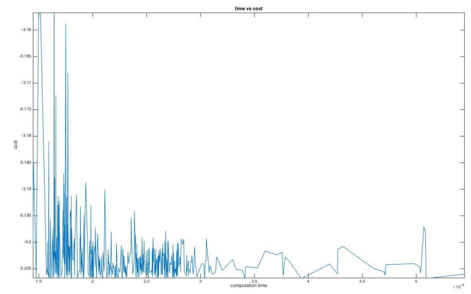


*Fig3. GD mini-batch*

## Question c

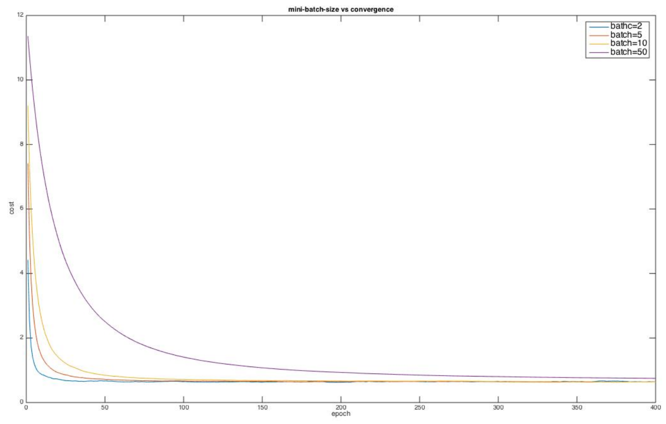
The left figure shows the correlation between the size of mini-batch and computation time. The right figure shows the correlation between computation time and cost. We can get the conclusion that with the increase of computatio time, the cost decreases gradually. The reason is that the computation time has a positive relation with the numbr of samples, while the number of samples has a negative relation with the minimum cost.





## Question d

In this question, I used 4 different sizes of mini-batch, batch=2, batch=5, batch=10, batch=50. We can see from the figure that gradient descent with larger mini batch converges more slowly than gradient descent with smaller mini batch. The reason is that GD with small mini batch computes less samples in each iteration. Therefore, GD with small mini batch



## Question e

alpha = 0.06;

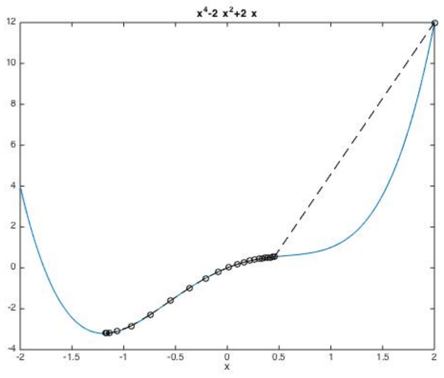


*Fig1. Gradient Descent*

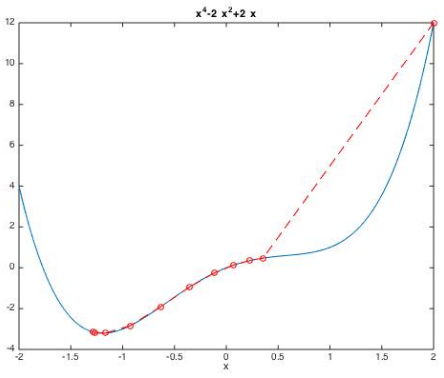


*Fig2. Stochastic GD*

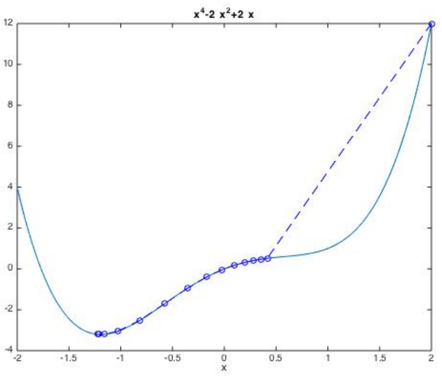


*Fig3. GD mini-batch*

alpha = 0.12;

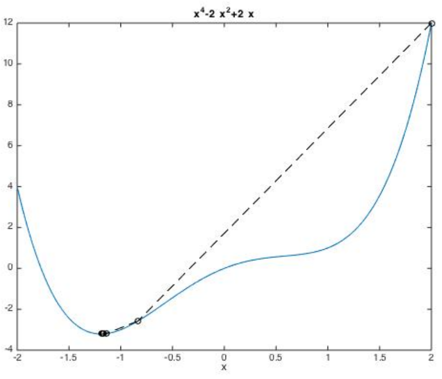


*Fig1. Gradient Descent*

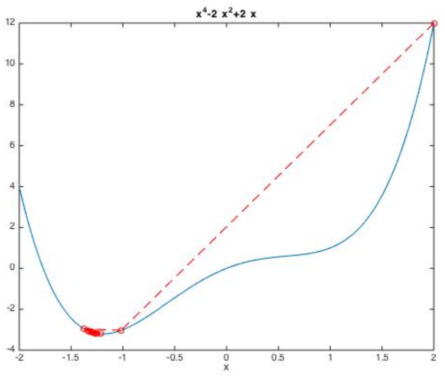


*Fig2. Stochastic GD*

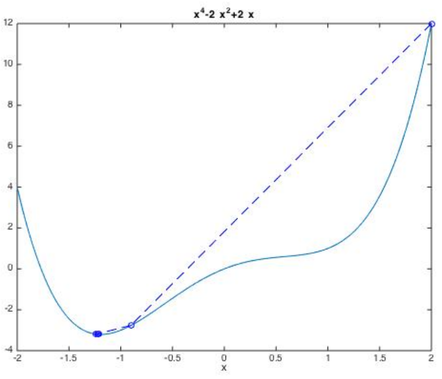


*Fig3. GD mini-batch*

From the above pictures, we can get two conclusions. First, learning rate decides the speed of reaching to the minmum cost. If learning rate is higher, it can reach to the minmum cost much faster. Then, in terms of the convergence, we can see that SGD converges faster than the others, but it vibrates a bit before it gets to its minmum.

## Question f

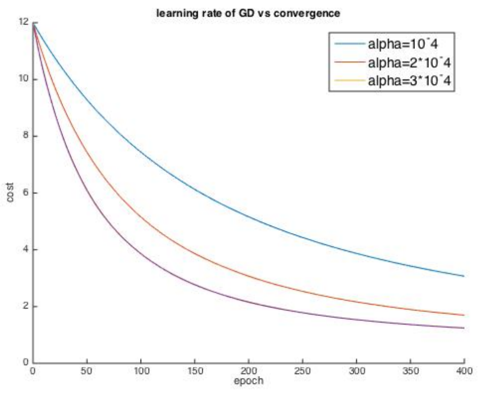


*Fig1. Gradient Descent*

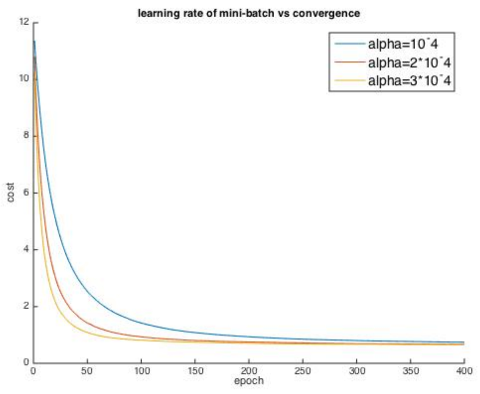


*Fig2. GD-mini batch*

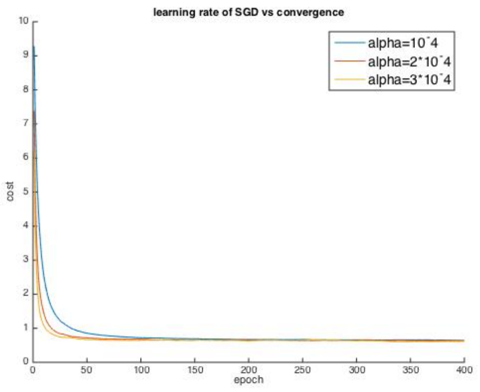
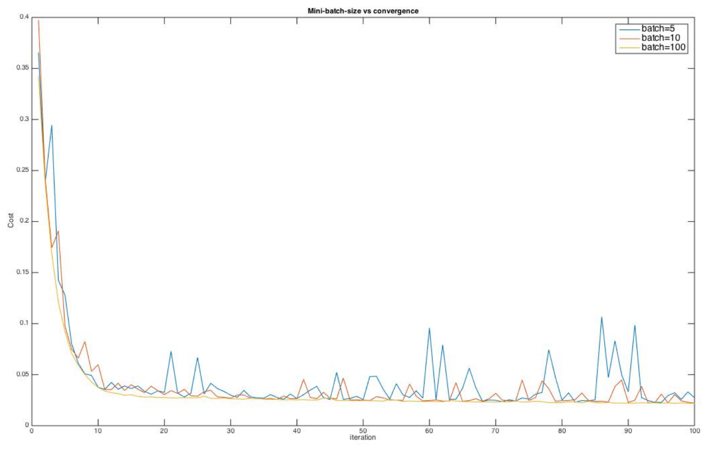


*Fig3. SGD*

In general, when learning rates are the same, the speed of convergence increases with the descrease of mini-batch size (SGD converges most fast and BGD converges most slowly). On the other side, when using the same method, the speed of convergence increases with the increase of learning rate.

## Question g



In the end, lager size of mini-batch can get lower minimum cost. When smaller mini-batch comes to convergence, the result is not stable as shown in the figure. Mini-batch Gradient Descent can get best minimum cost when taking more samples into account.

## Question h

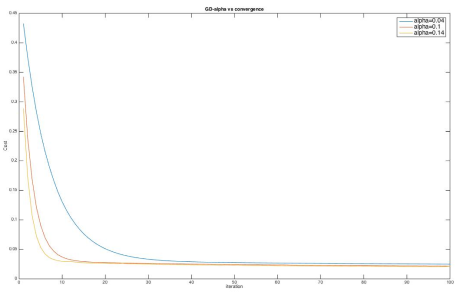


*Fig1. Gradient Descent*

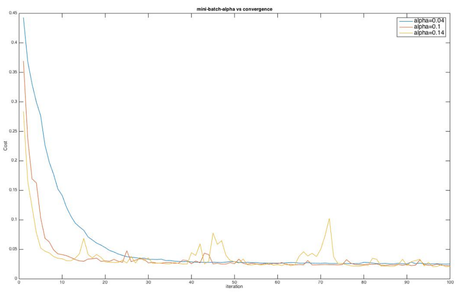


*Fig2. Mini-Batch GD*

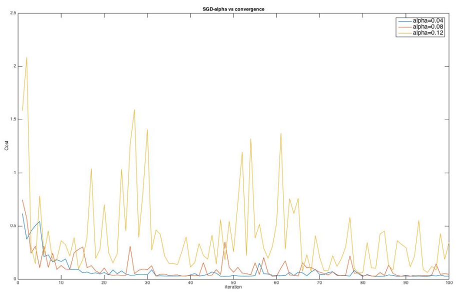


*Fig3. Stochastic GD*

Bascially, the result is similar to question f: the higher learning rate can result in much faster convergence in each variance.

## Question i

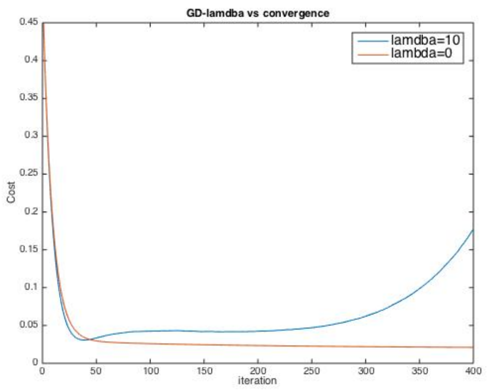


*Fig1. Gradient Descent*

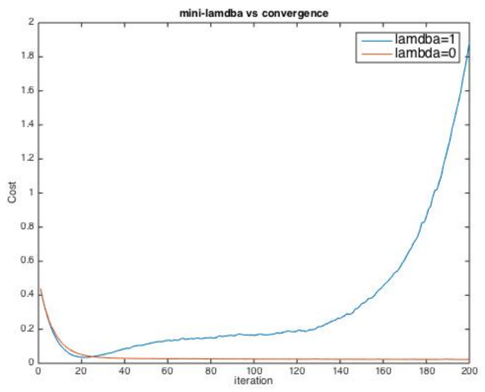
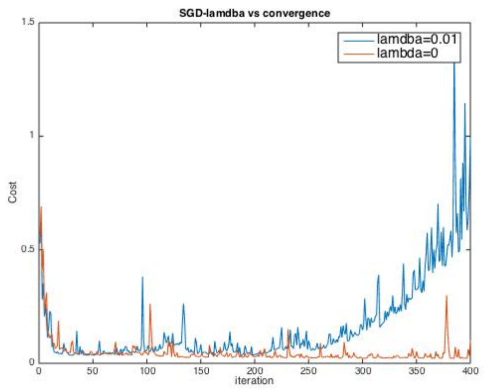


*Fig2. Mini-Batch GD*



Regularization amis to avoid overfitting. When minimum cost comes to a low value, it will take it as overfitting and then adjust the cost to a lager value. So, it's better to leave out regularization if we want to analyze the convergence.