1. Complete the stochastic gradient algorithm.

In order to fit the massive data situation, we need to the stochastic gradient algorithm. The stochastic gradient descendent, we couldn't take all the data to do the gradient, we need the choose a batch of sub dataset. Using the function ranperm(N, batchSize) to create the index vector, where N is number of non-zeros and batchSize is the batch size that we have chosen we take 1000.

For each iteration, we need to choose the current coordinate by using

```
cur_i = Xlist(data_index(i),1);
cur_j = Xlist(data_index(i),2);
cur_x = Xlist(data_index(i),3);
```

Where Xlist is the matrix that using the function find(X) to find non-zero value in this matrix X. It will output (xi, xj, value) where value!=0.

Applying the SGD update rule:

$$W \leftarrow W - \eta \frac{0.5 * d\left|\left|X - W * H\right|\right|_F^2}{dw}$$
$$W \leftarrow W + \eta(X - W * H) * H^T$$

$$H \leftarrow H - \eta \frac{0.5 * d\left|\left|X - W * H\right|\right|_F^2}{dh}$$
$$H \leftarrow H + \eta(X - W * H) * H^T$$

Here since just a take the grad of each coordinate.

```
grad_w = -(cur_x-cur_xhat)*transpose(H(:,cur_j));
grad_h = -transpose(W(cur_i,:))*(cur_x-cur_xhat);

%take a gradient step
W(cur_i,:) = W(cur_i,:) - eta*grad_w;
H(:,cur_j) = H(:,cur_j) - eta*grad_h;
```

The following is my code:

```
data_index = randperm(N,batchSize);
for i = 1:batchSize
    cur_i = Xlist(data_index(i),1);
    cur_j = Xlist(data_index(i),2);
    cur_x = Xlist(data_index(i),3);

    cur_xhat = W(cur_i,:)*H(:,cur_j);%todo

    grad_w = -(cur_x-cur_xhat)*transpose(H(:,cur_j));%todo
    grad_h = -transpose(W(cur_i,:))*(cur_x-cur_xhat);%todo

    W(cur_i,:) = W(cur_i,:) - eta*grad_w;%todo
```

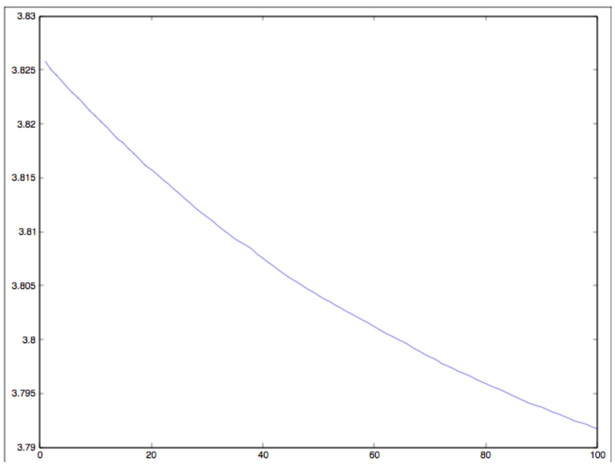
```
        H(:,cur_j) = H(:,cur_j) - eta*grad_h;%todo
    End
```

2. At the end of each iteration, compute the roor-mean-squared-error by using this rule.

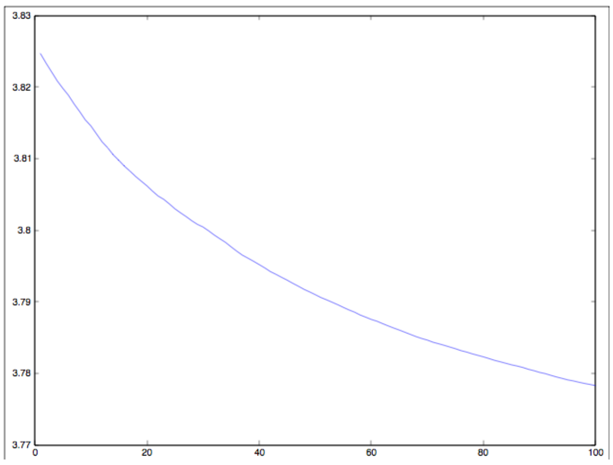
$$\mathrm{RMSE} = \sqrt{\frac{\|M \odot (X - WH)\|_F^2}{N}}$$

```
rmse_sgd(t)=sqrt(sum(sum(((M.*(W*H))-Xsp).^2))/N);
```

3. Play with the algorithm parameters, i.e. the step-size, the batch-size, initialization, and the rank of the factorization. What do you observe? How do the step-size and the batch-size interact?

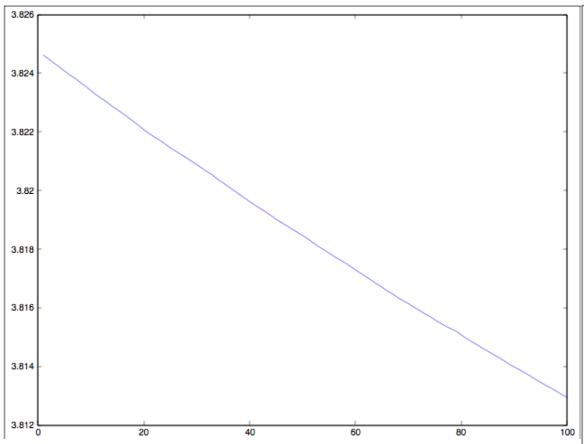


batchSize=500;

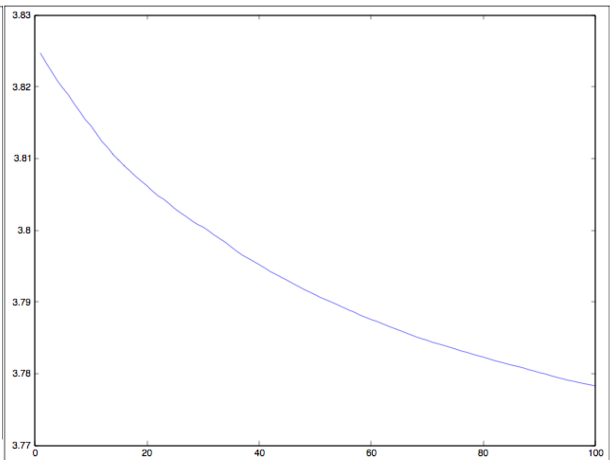


batchSize=1000;

When we decrease the batch Size into 500, I find after 100 iterations, the loss is larger than the loss when batchSize=1000. The more data we use, the less loss we get.

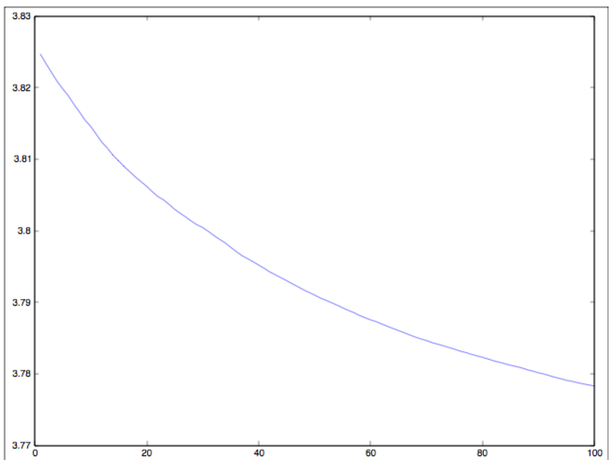


eta=0.001;

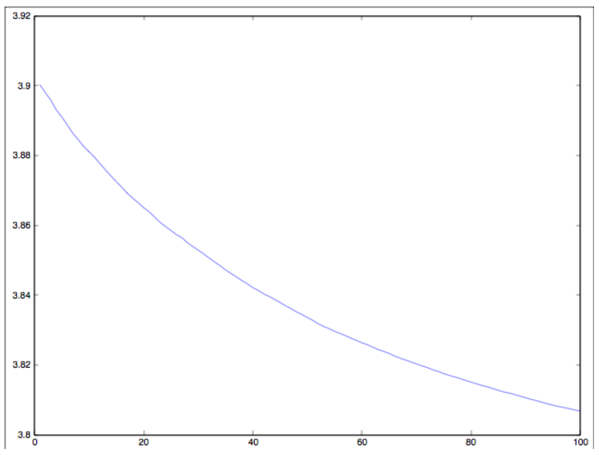


eta=0.01

When we decrease the eta, the loss function will almost be linear, however still the loss value is larger than eta=0.01. When eta is small, the change of gradient will be small.



rank=10;



rank=20;

When rank is large, the loss value will be large.

4. After estimating W and H, use them to recommend a movie for a given user.

Here we choose the user 11, to recommend a movie for him. W*H(:user_index) will give us the predict movie for the user 11, we choose the max one to recommend to the user.

```
user_index = 11;
user_xhat=(W*H(:,user_index));
[xx,movie_index] = max(user_xhat);
```

The max value in the vector will recommend to user.

Recommend movie 981 to user 11