Factorization-Based Data Modeling

Practical Work 1

Students : ZHAO Mengzi, ZHOU Juncheng

PART I : Matrix Factorization with Alternating Least Squares and Gradient Descent

The problem that we aim to solve :

$$(W^*, H^*) = arg \min_{W,H} \frac{1}{2} \|X - WH\|^2$$

X$\in R^{I*J}$, X is the data matrix, W$\in R^{I*K}$ and H$\in R^{K*J}$ are 2 unknown factor matrix.

1 – Implement the ALS rules :

In this question, we need to apply the formula of Alternating Least Squares algorithm :

$$W = XH^T(HH^T)^{-1}$$

$$H = (W^TW)^{-1}W^TX$$

According to formula, we have code :

Wals = X * Hals' * (Hals * Hals')^(-1);

Hals = (Wals' * Wals)^(-1) * Wals' * X;

Explication :

Wals and Hals present 2 unknown factor matrix, we need to calculate by using the formula and their initial values. Xhat is equal to the product of this 2 matrix, Xhat means the value of prediction.

For example, for W, It is from :

The gradient of W is equal to : $(WH - X) * H^T$

For the matrix of $\nabla wf(W, H_0) = \begin{matrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ 0 & \dots & 0 \end{matrix}$

We have :

$$(WH_0 - X){H_0}^T = 0$$
$$W{H_0}{H_0}^T - X{H_0} = 0$$
$$W{H_0}{H_0}^T = X{H_0}$$

$$W = XH_0(H_0{H_0}^T)^{-1}$$

So we have : $W = XH^T(HH^T)^{-1}$

For H, we use the same principle.

2 – Compute the objective function values for each iteration :

We need to compute the objective function according to the formula :

$$(W^*, H^*) = arg \min_{W,H} \frac{1}{2} \|X - WH\|^2$$

we have :

Xhat = Wals * Hals;

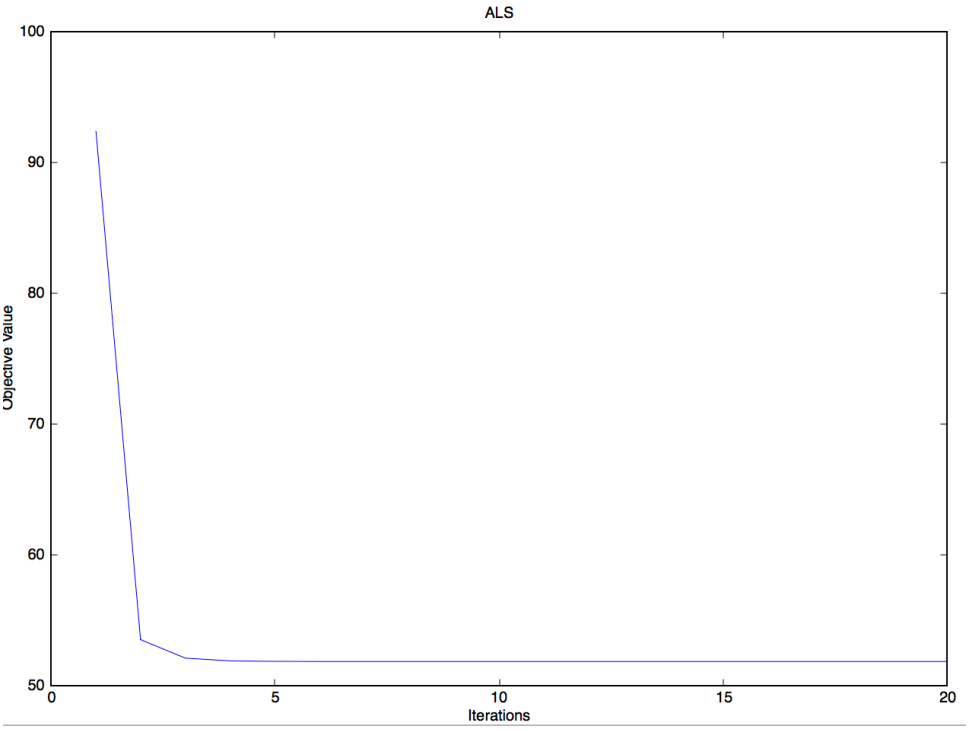
obj_als(i) = (1/2) * (norm((X - Xhat),'fro')).**(2);

Explication :

Xhat is equal to the product of Wals and Hals.

According to the formula, we can write the objective function.

We obtain the graph :



We can notice that the objective function converges.

3 – Implement the GD update rules :

$$W \leftarrow W + \eta(X - WH)H^T$$
$$H \leftarrow H + \eta W^T(X - WH)$$

η is a fixed step-size.

According to the formula, we have code :

Wgd = Wgd + eta * (X - Wgd * Hgd) * Hgd';

Hgd = Hgd + eta * Wgd' * (X - Wgd * Hgd);

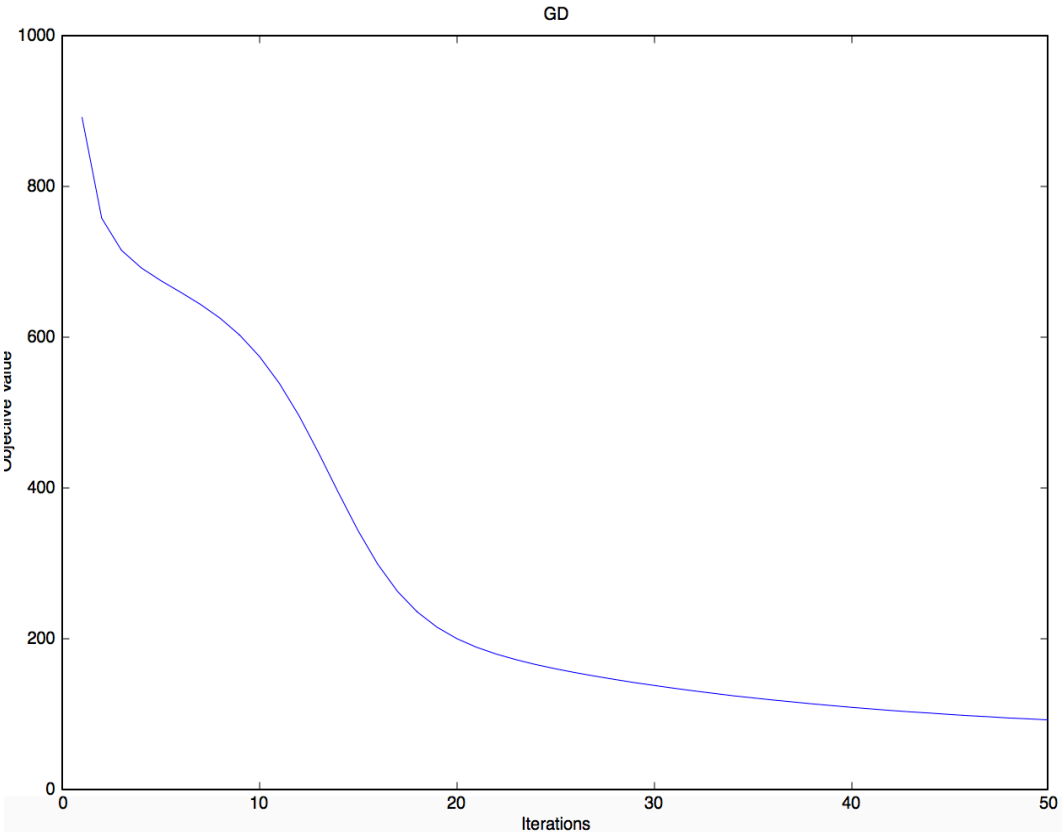
Explication :

Wgd and Hgd are 2 unknown factor matrix, so we need to use the formula above to calculate their value.

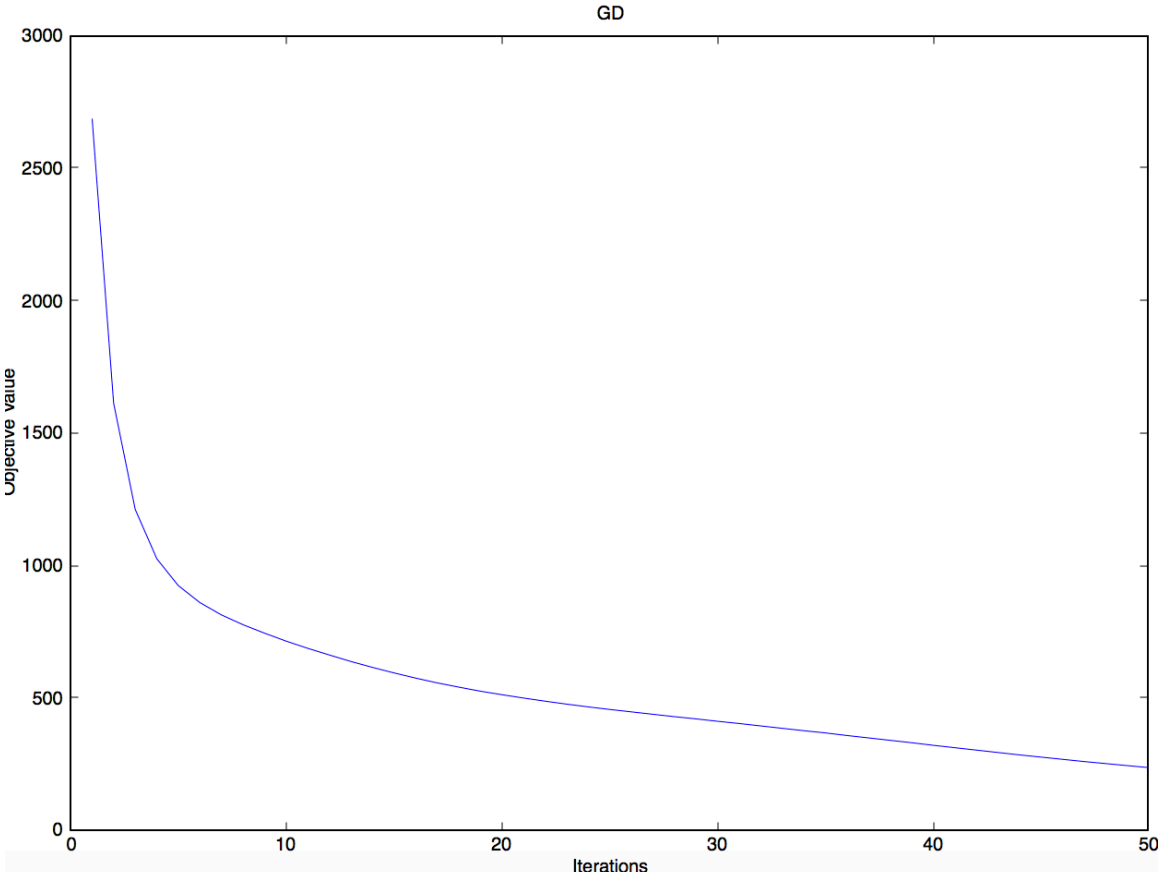
4 – What is the effect of η ?

η is a fixed step-size, it is the learning rate which controls the speed of the gradient descent.

When $\eta = 0.01$, we have :



When $\eta = 0.005$, we have :



We can notice that when $\eta = 0.01$ the objective function converges more quickly. For example, for $\eta = 0.01$, when the number of iterations is equal to 20, the objective value is about 250, but for $\eta = 0.005$, the objective value is more than 500. So for the same algorithm, the effect of converging of the first graph is better than the second one. But the value of η is not as small as possible, because when η is too small, so we will obtain 2 values token from the functions, these 2 values are almost equal, so if these 2 values are almost equal, they can easily satisfy the condition of converging, so in this case, the function will not continue converging, so the η cannot be too small. η can neither be too large, the speed of converging will be too quick, it is also possible to take 2 values for example x1, x2, their value of function y1, y2 are equal, so we can have a value η which make the process of converging shock to and forth between x1 and x2, so when η is too large, it is also not good.

5 – Compute the objective function values for each iteration (GD).

We need to compute the objective function according to the formula :

$$(W^*, H^*) = arg \min_{W,H} \frac{1}{2} \|X - WH\|^2$$

We have code :

```
Xhat = Wgd * Hgd;
obj_gd(i) = (1/2) * (norm((X - Xhat),'fro')).**(2);
```

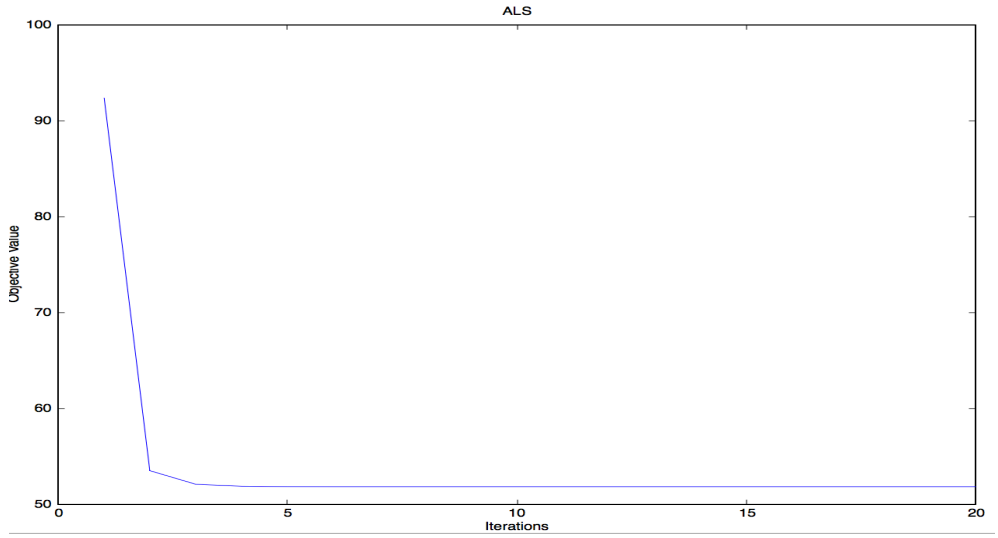
Explication :

Xhat is equal to the product of Wgd and Hgd.

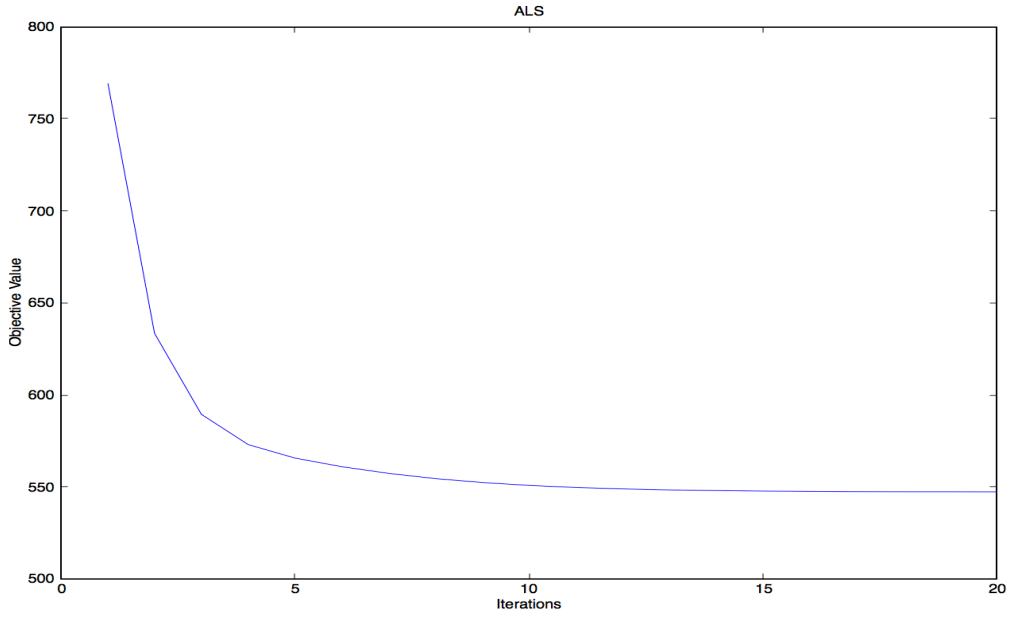
According to the formula, we can write the objective function for GD.

6 – Play with the variable « dataNoise ». What is the effect of this parameter on the algorithms.

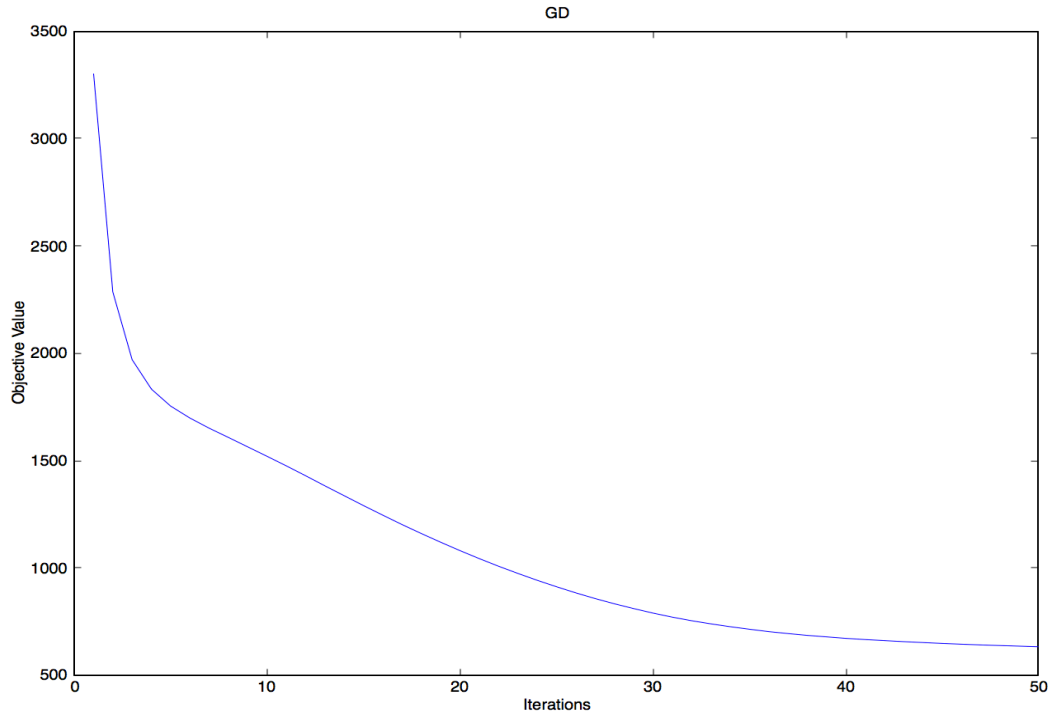
For ALS, when dataNoise = 1



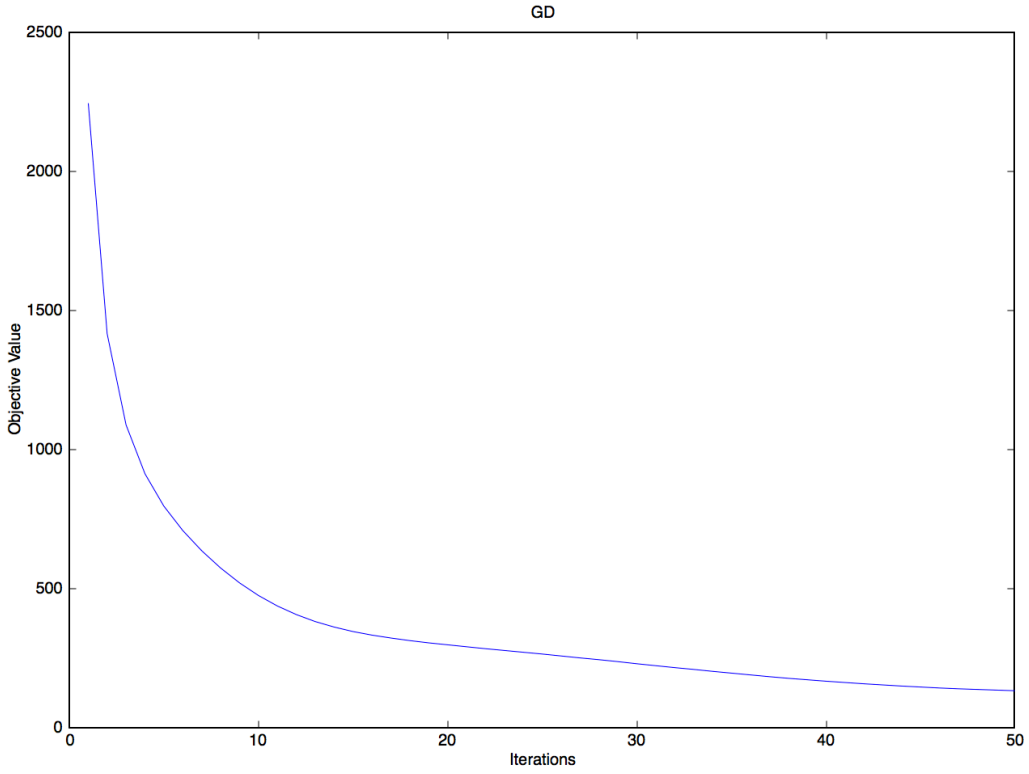
For ALS, when dataNoise = 3



For GD, when dataNoise = 1



For GD, when dataNoise = 3



The objective function is

$$(W^*, H^*) = arg \min_{W,H} \frac{1}{2} \|X - WH\|$$

It means we should find the argument which can make the value of the right part of this formula minimal.

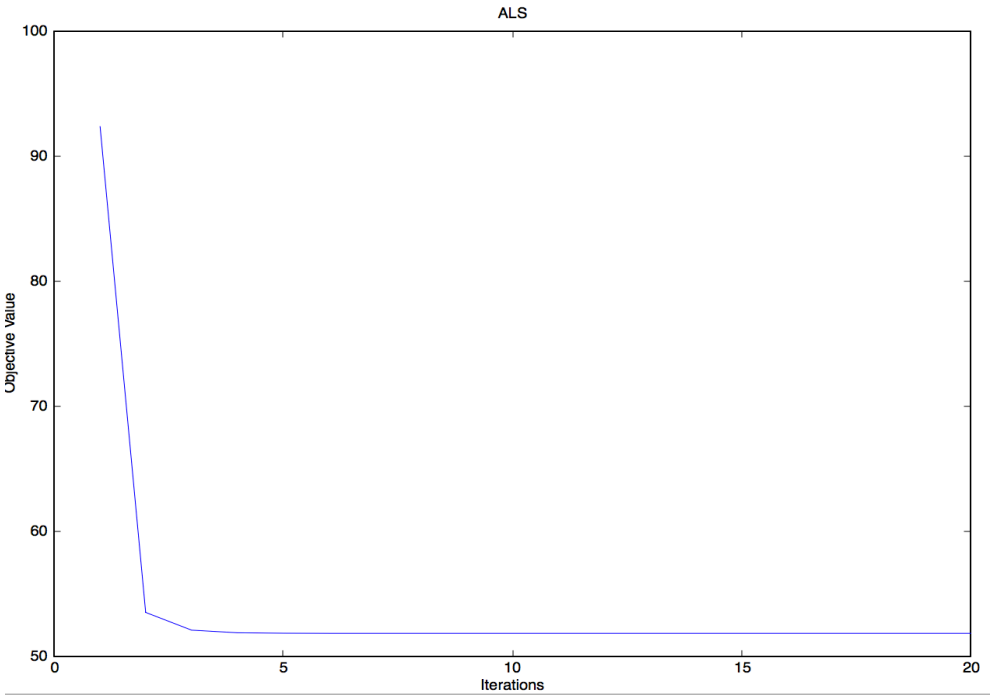
For ALS, comparing these 2 graphs, we can notice that when the value of dataNoise is larger, the result that we obtain is larger, when dataNoise = 1, the curve of the objective function is stable when the objective value is about 51, when dataNoise = 3, the curve of the objective function is stable when the objective value is about 550. According to the aim of the objective function, we can notice that the result when dataNoise = 1 is much better than the result when dataNoise = 3.

For GD, it is the same. When dataNoise = 1, the curve is stable when the value of the objective function is about 150, when dataNoise = 3, the curve is stable when the value of the objective function is about 750.

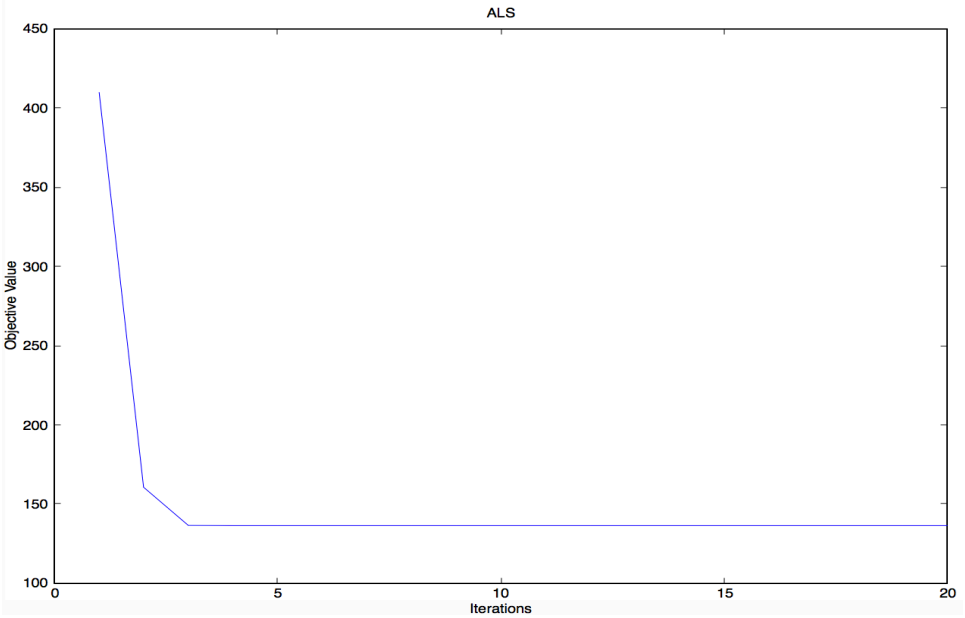
So we can know that the dataNoise can influence the result, when the value of dataNoise is larger, the result will be worse, because dataNoise make the curve farther from the axe X.

7 – Play with the size of the data (I, J) and the rank of the factorization K, what behavior do you observe ?

For the first graph, the size of the data (I,J) = (10, 20), for the second graph, the size of the data (I,J) = (15,25)

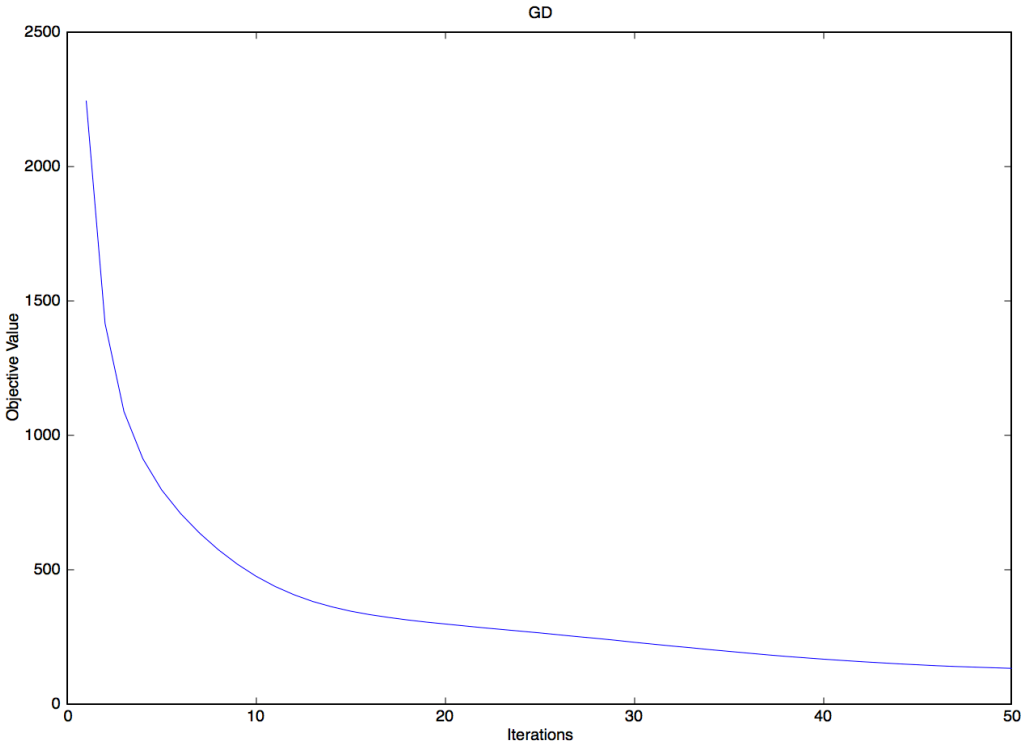


when I = 15 J = 25

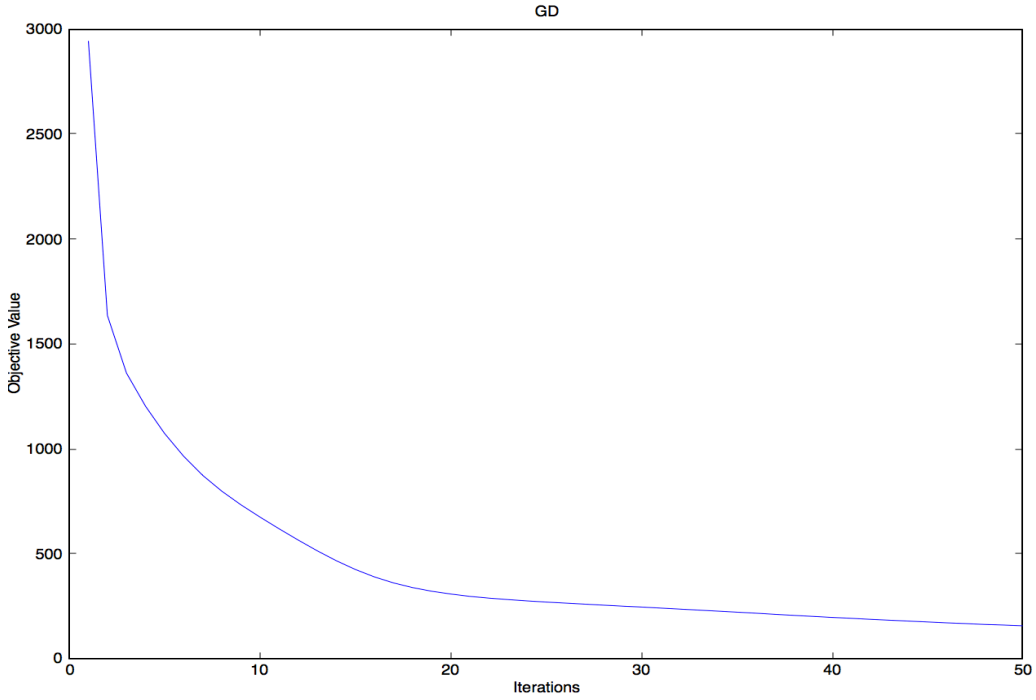


We can notice that the curve in the first graph is stable when the objective value = 52, and the curve of the second graph is stable when the objective value is about 130. So we can know the result of the first one is better than the second one.

For GD, all parameters of these 2 graphs are the same excepting the size of the data. For the first one, the size of the data is (10,20), for the second one, the size of the data is (15,25).



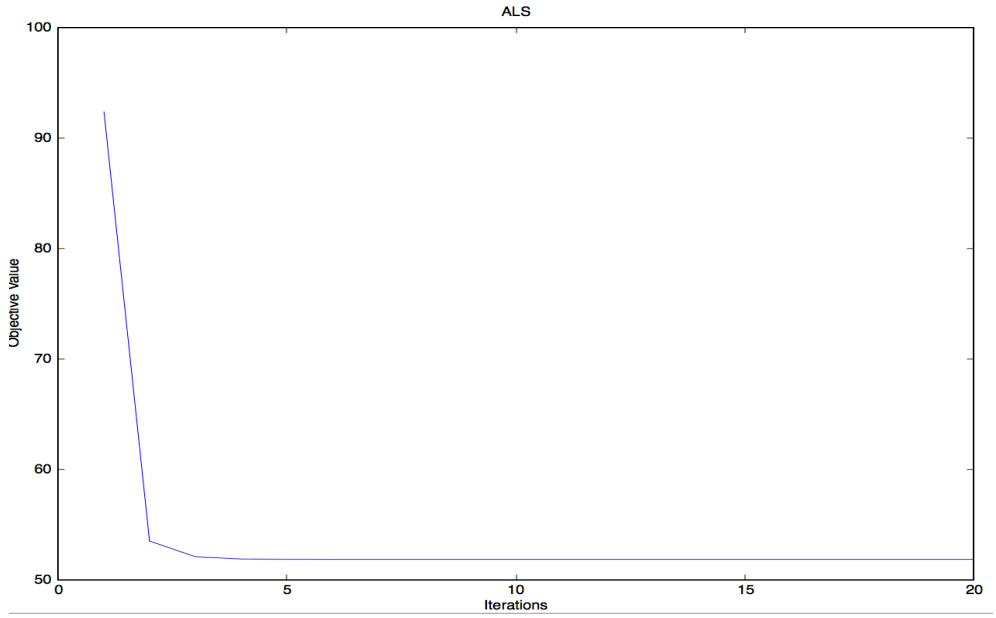
When the size of the data (I,J) = (15,25).



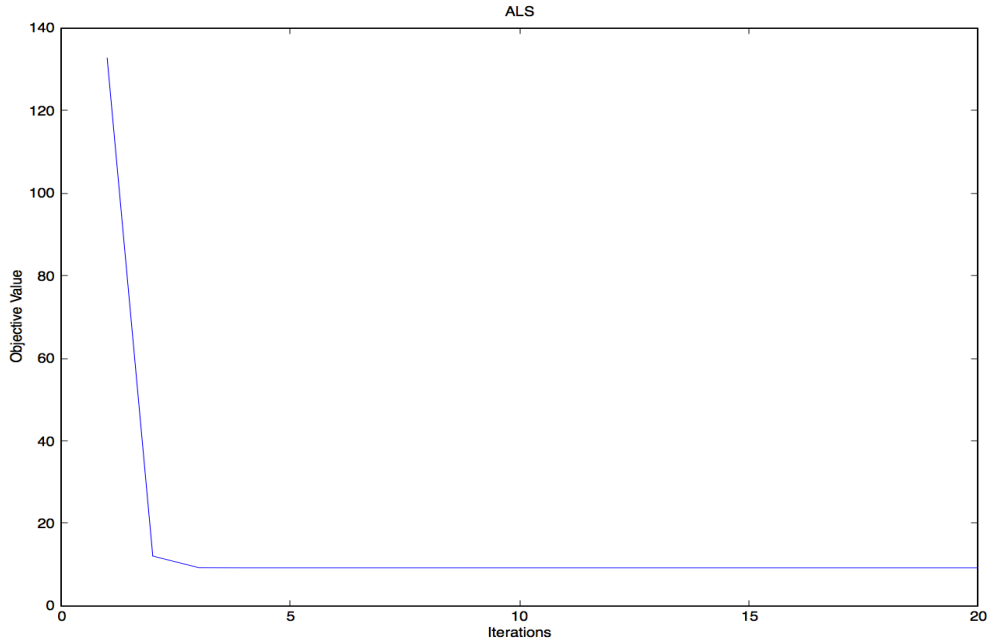
We can notice that the curve in the first graph is stable when the objective value is about 100, and the curve of the second graph is stable when the objective value is about 155. So we can know the result of the first one is better than the second one.
And we can notice also the first graph is almost stable when the number of iteration is about 50, but the second graph is stable when the number of the iterations is about 30.
According to these graphs, we can know that the when the size of data is smaller, we can get a better result. Because this process is to separate a large matrix to 2 small matrix with lower dimension, in other word, this process is to do dimensionality reduction. So when the size of the data is larger, it will be more difficult to do the dimensionality reduction.

We also try to find the influence of the K : rank of the factorization.
For ALS, we set for the first graph K = 3, and the second graph, K = 8.

K = 3

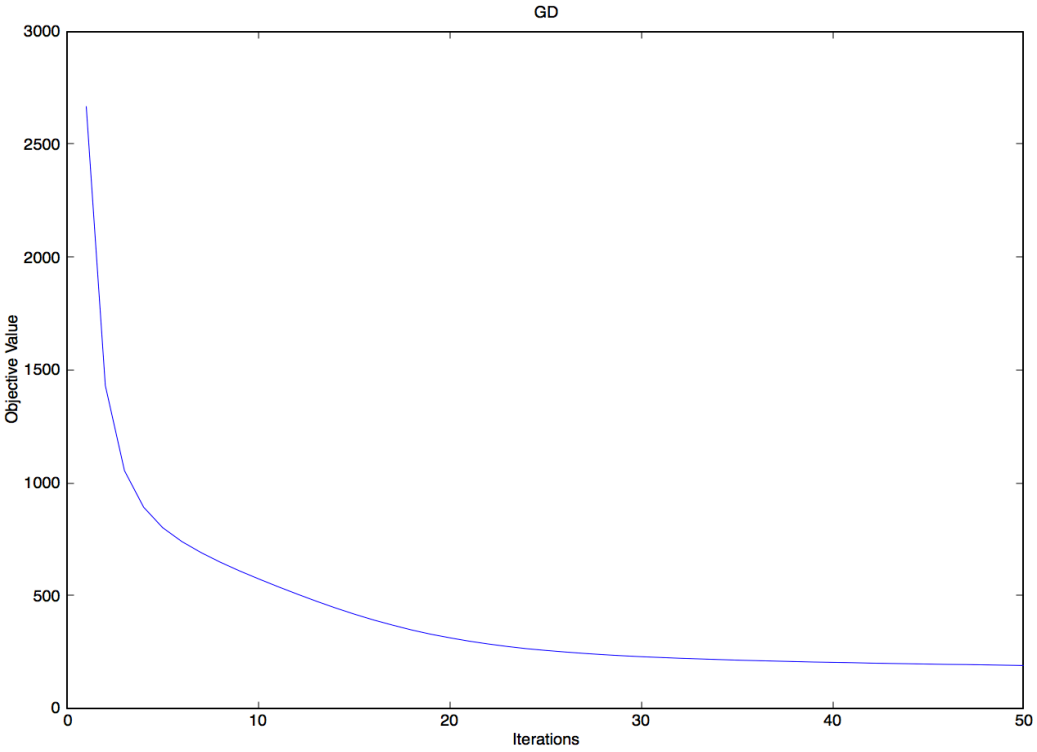


K = 8

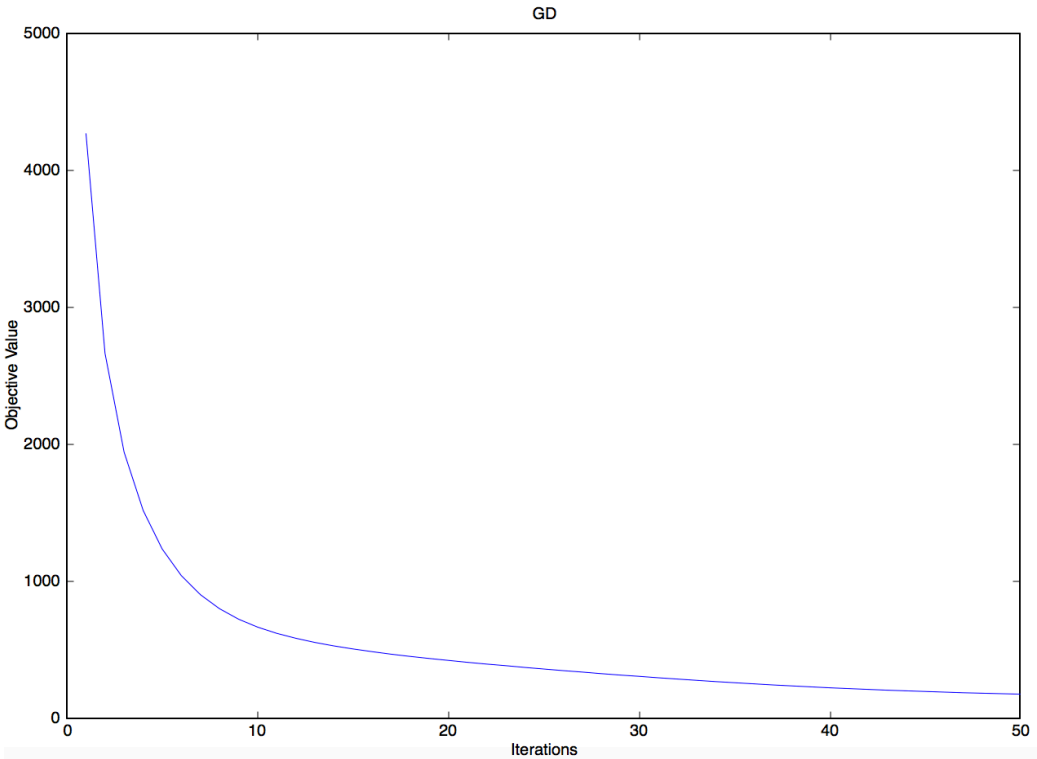


We can notice that when K = 3, the curve is stable when the objective value is about 52, and for the second graph, the curve is stable when the objective value is about 10. When K is bigger, it means the dimension of matrix W and H is bigger, so there are more features, so we can get a better result. But when K is larger, the speed of converging will me slower than the case "when k is smaller". So the speed of converging of the first graph is more quick than the second graph.

For GD, we set for the first graph K = 3, and the second graph, K = 8.



For GD, K = 8



We can notice that when K = 3, the curve is stable when the objective value is about 160, and for the second graph, the curve continues converging, it is not stable in this

graph. But we can know when the number of iterations is 50, the first graph has the objective value 160, and the second has the objective value about 190, so we can know that the first curve converges more quick than the second. And in the second graph, the curve continues converging, in general, we can get a better result in the second graph than the first graph.
But when the value of K is too large, it may be cause overfitting.

8 – Which algorithm do you think is better, why ?

The goal of ALS is to find the least square sum of the error function. There are two types of objective functions, linear and nonlinear. In the linear case, the solution of ALS is closed form, and the result is globally optimal. In the non-linear case, we need to iteratively update the unknown variables at each step. The amount of calculation of ALS is relatively large, we all know that finding the inverse matrix of a matrix is very time-consuming, and at this time the result cannot guarantee the global optimum, which is generally the local optimum. So when using linear ALS it is appropriate.
The gradient descent uses an iterative method that can compute both linear and nonlinear problems. The number of iterations is relatively high, the convergence speed is slow, but the computation is not large, so the gradient descent method is more suitable than the ALS when the amount of large data is large.

PART II : Non-negative matrix factorization with multiplicative update rules

1 – Implement the MUR update rules.

We write the code according to the formula of the update rules :

$$W \leftarrow W \circ \frac{(X/Xhat)H^T}{OH^T}$$

$$H \leftarrow H \circ \frac{W^T(X/Xhat)}{W^TO}$$

Each element of the matrix O is equal to 1.
We have code :

```
Wmur = Wmur .* (((X./Xhat)*Hmur')./(O*Hmur'));
Hmur = Hmur .* ((Wmur'*(X./Xhat)))./(Wmur'*O);
```

2 – Compute the objective function values for each iteration.

We write the code according to the formula of the update rules :

$$(W^*, H^*) = \ argmin \sum_{i=1}^{I} \sum_{j=1}^{J} (x_{ij} * \log \frac{x_{ij}}{xhat_{ij}} - x_{ij} + xhat_{ij})$$

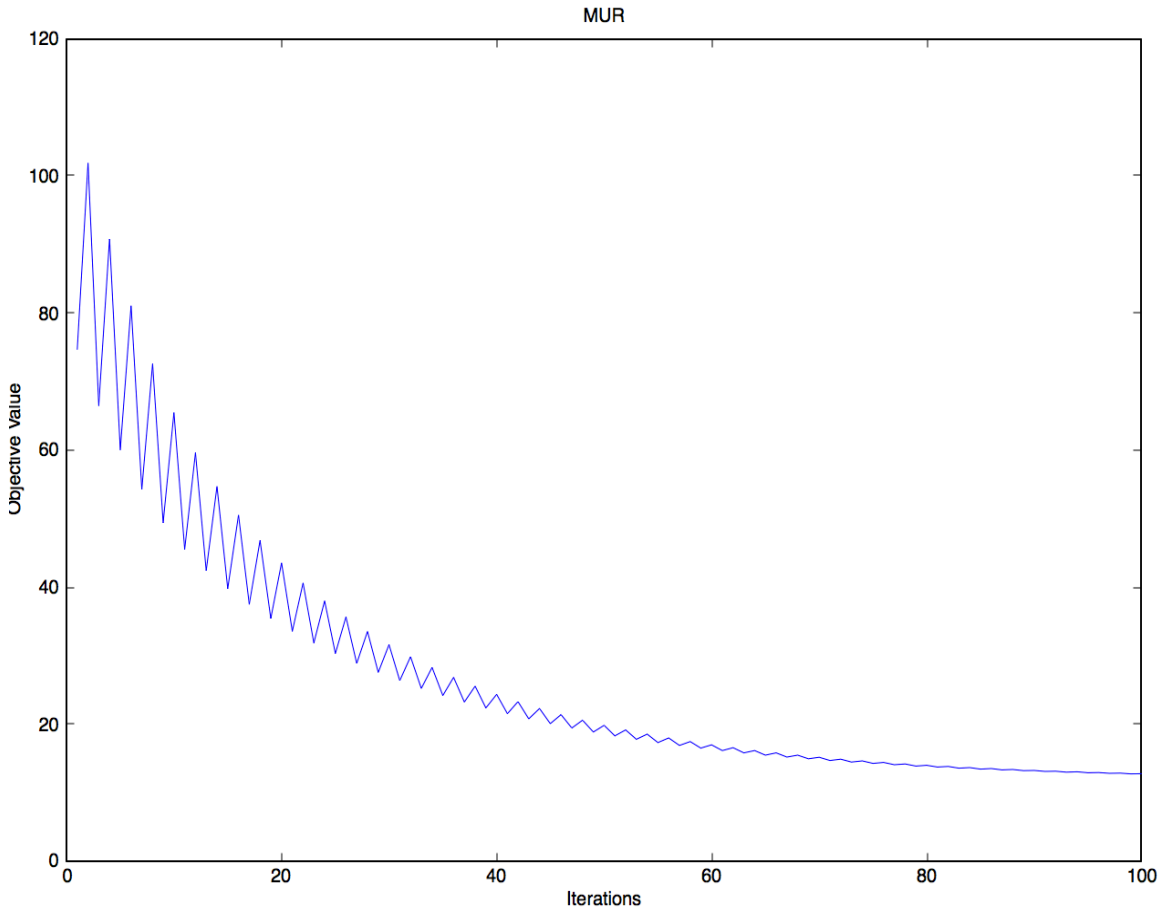
So we have code :

```
func = X.*log(X./Xhat)-X+Xhat;
obj_mur(i) = sum(func(:));
```

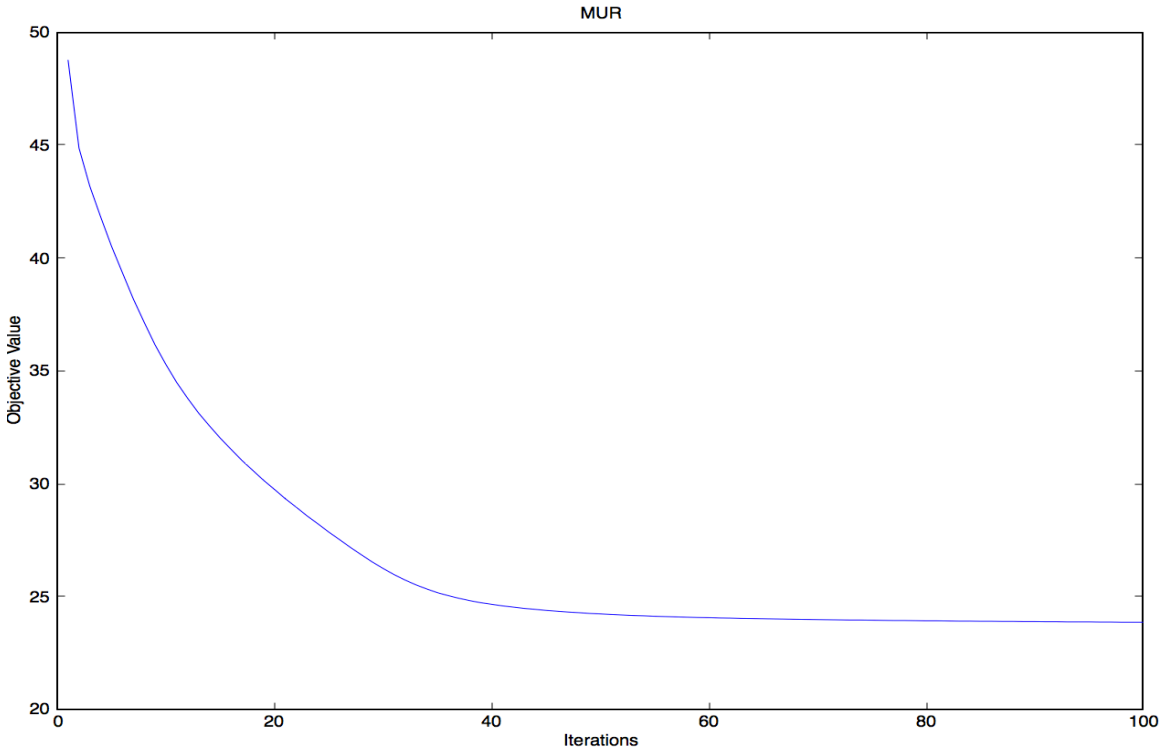
3 – Play with the variable dataNoise, what is the effect of this parameter on the algorithms.

We test with 2 values for dataNoise, at the beginning, we make dataNoise = 1, for the second, we make dataNoise = 3.

dataNoise = 1 :



dataNoise = 3 :

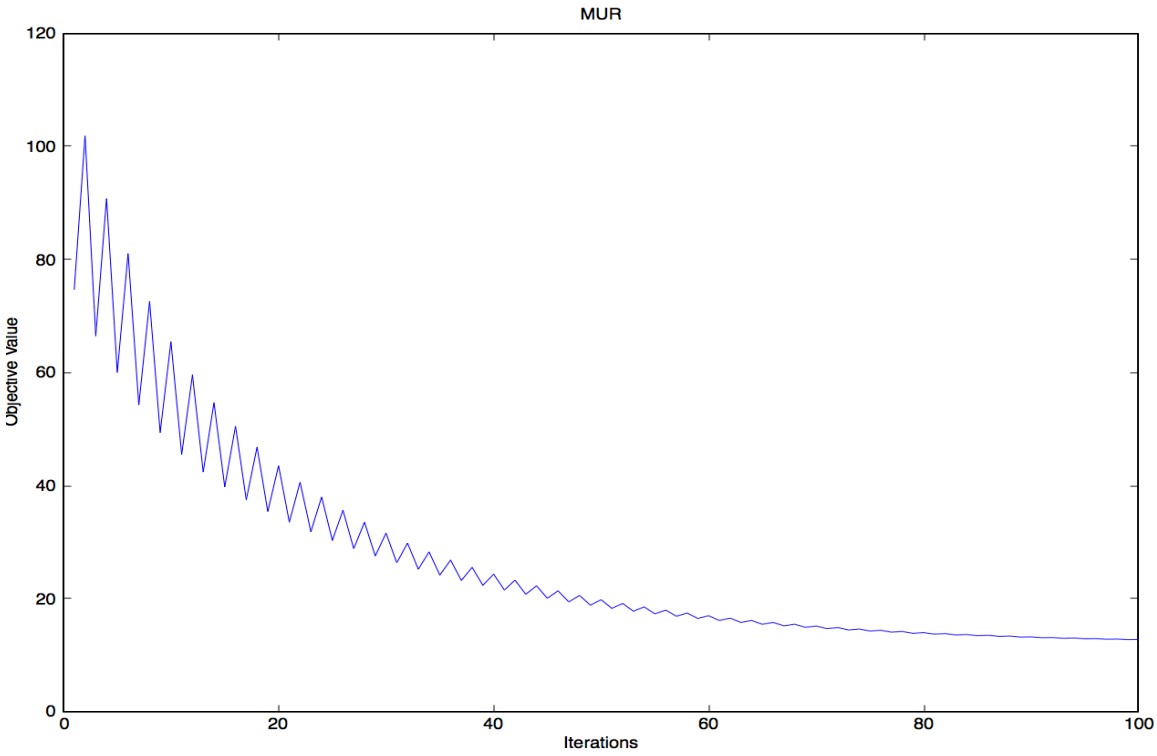


We can notice that when dataNoise = 1, the curve will be stable when the objective value is equal to 10, and when dataNoise = 3, the curve will be stable when the objective value is equal to 24. So we can say when the dataNoise is larger, dataNoise will influence more the result, when the dataNoise is smaller, the result is better.

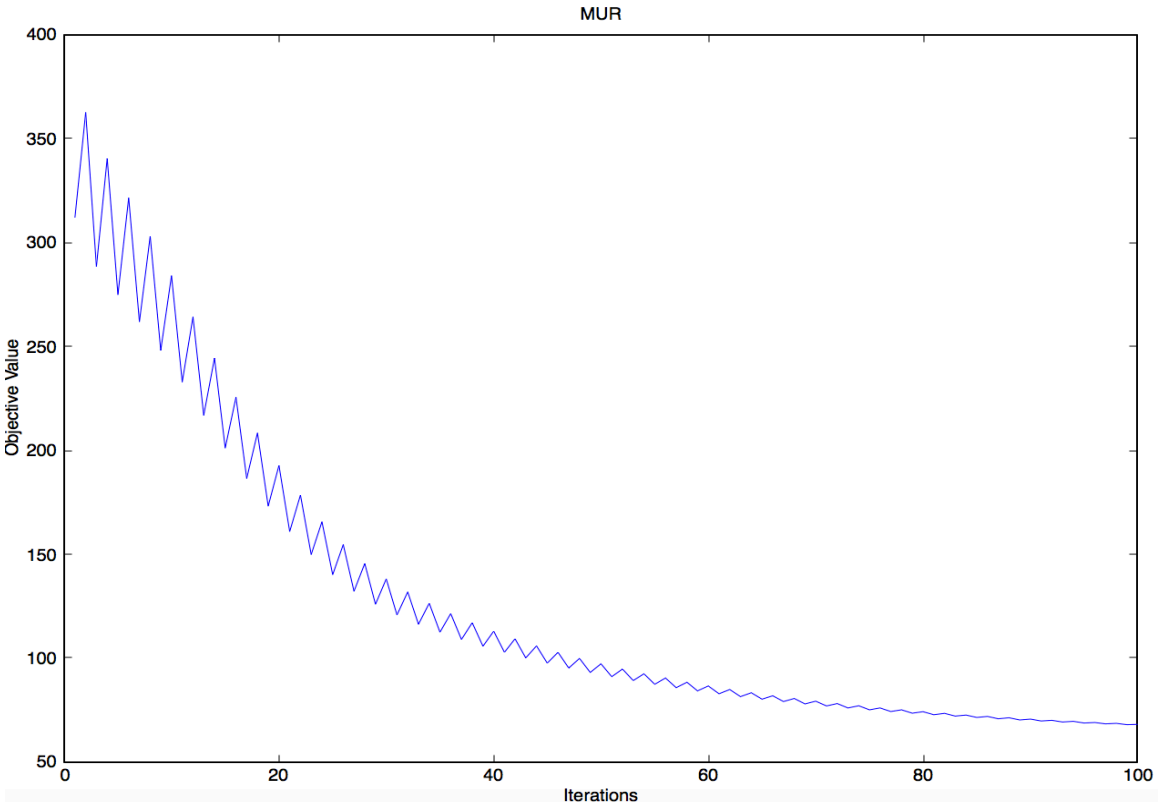
4 – Play with the size of the data (I, J) and the rank of the factorization K. What behavior do you observe ?

At the beginning, we set (I,J) = (10,20), and for the second part, we set the (I,J) = (25,35).

For (I,J) = (10,20), we have the graph :

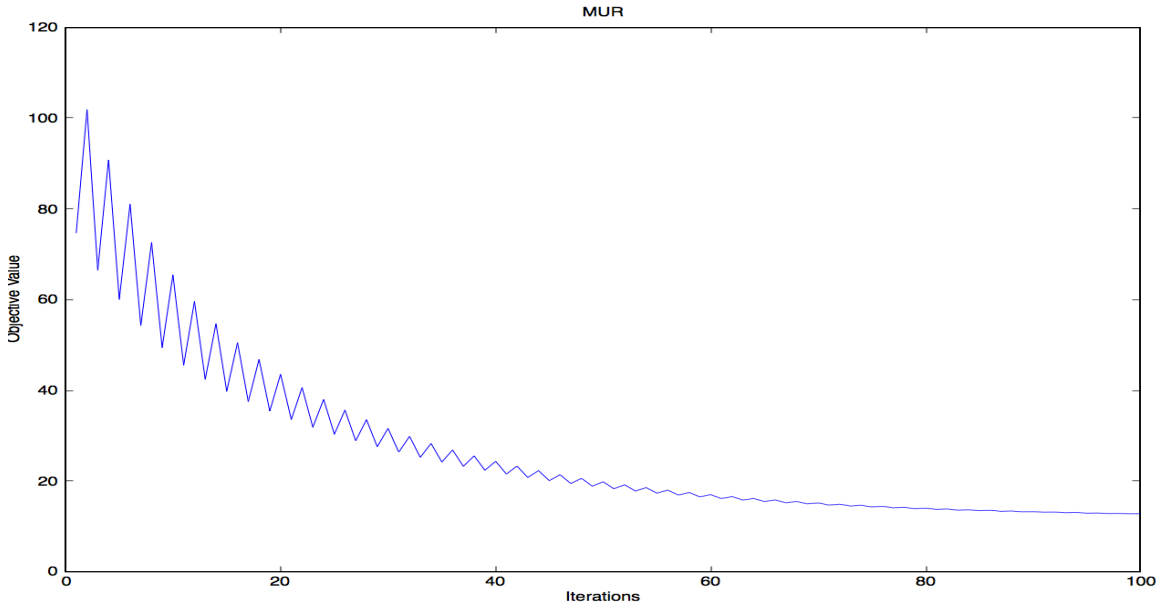


For (I,J) = (25,35), we have the graph :

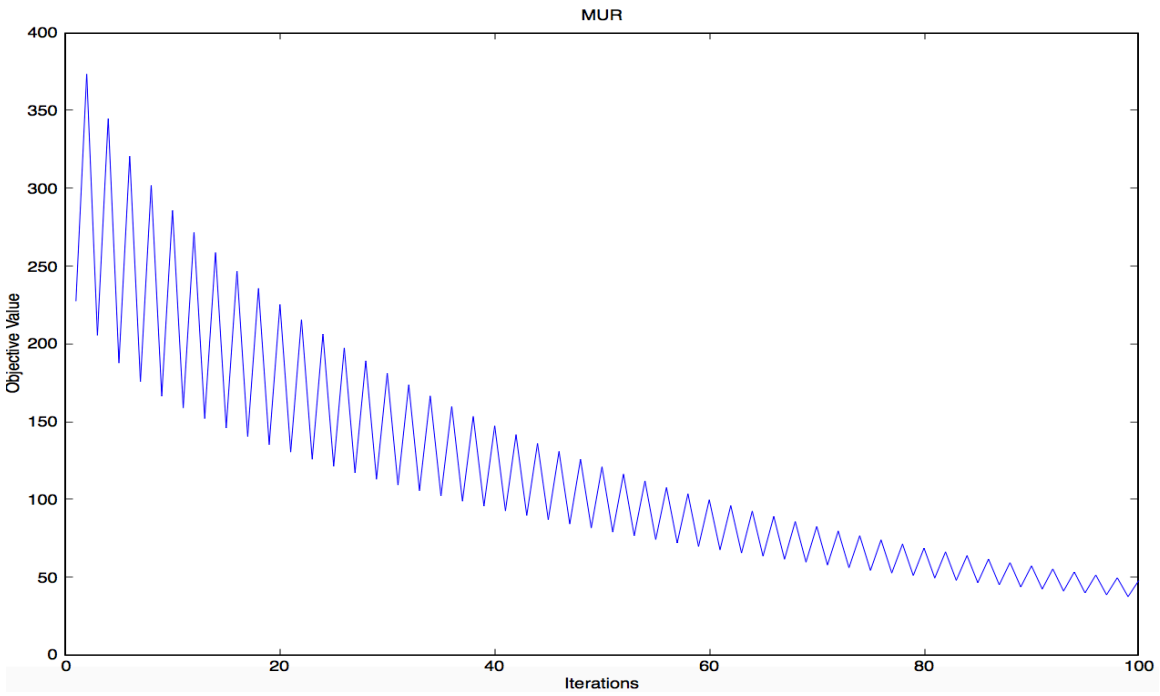


We can notice that, for the first graph, the curve is stable when the objective value is about 15, for the second graph, the curve is stable when the objective value is about 70. So we can say when (I, J) or the size of matrix is larger, the objective value will be bigger, but in this case, the goal of this function is to find the parameters of the function when the objective value is minimal, so the objective value is smaller, the better result can we have. So the result of first graph is better than the second, so the size is smaller, the result is better.

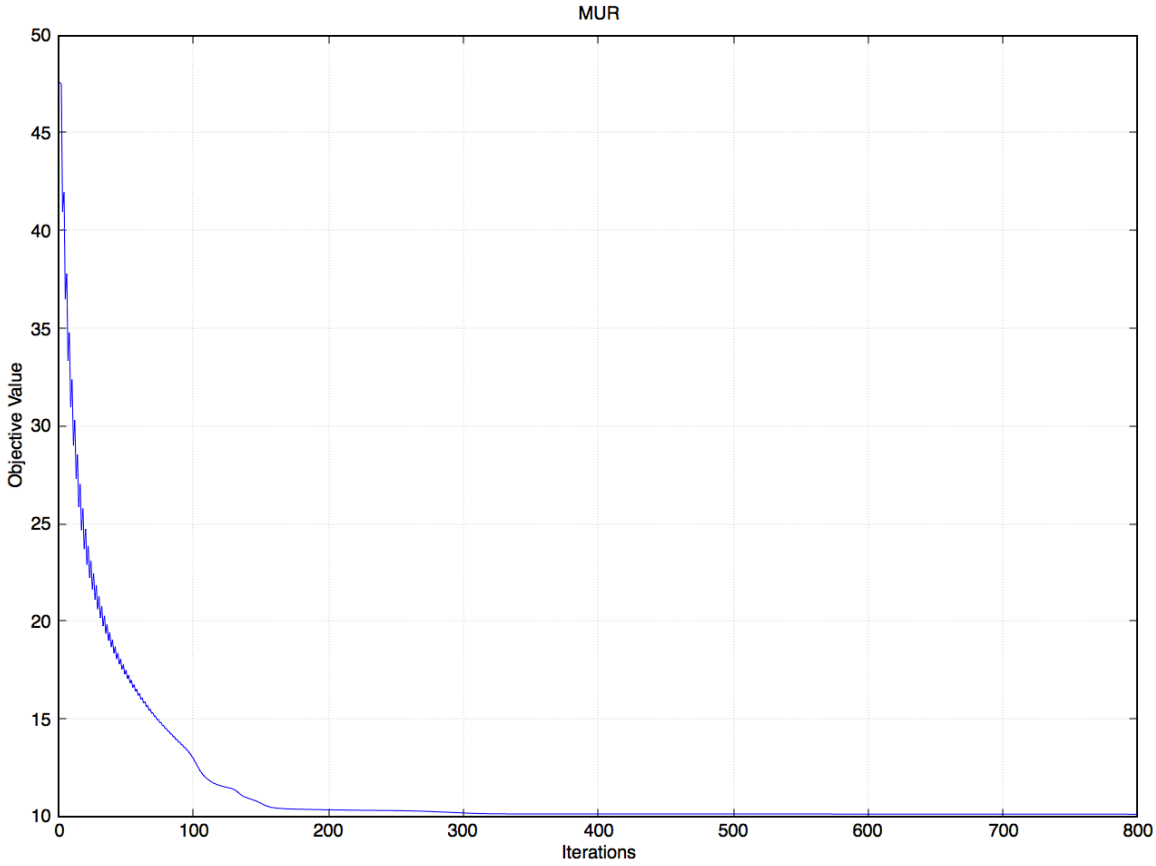
K = 3



K = 8



We can notice that when k = 3, the curve is stable when the number of iteration is about 100, at this moment, the objective value is about 15, but when k = 8, we can notice the curve does not finish converging even if the number of iteration arrives 100, at this moment, the objective value is about 50, so the curve continues converging, for k = 8, we set the number of iteration larger, we set it 800. We can obtain a graph as below :



We can notice that when the curve is stable, the objectif value is about 12, so when k is bigger, the result is better. But when k is too large, it may cause overfitting.