1 Stochastic Gradient Descent

1.1

The **objective** of the Matrix Factorization regression model with global bias, user bias, item bias and L2 regularization is:

$$\min_{\boldsymbol{\theta}} \sum_{(u,i,r_{ui})\in D} (r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)^2 + \lambda_x \sum_u ||\boldsymbol{x}_u||^2 + \lambda_y \sum_i ||\boldsymbol{y}_i||^2 + \lambda_{b_u} \sum_u b_u^2 + \lambda_{b_i} \sum_i b_i^2$$

From here and below, we will refer the objective as $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the model's parameters group:

$$\boldsymbol{\theta} = \Big[\{\boldsymbol{x}_u\}_{u=1}^U, \{b_u\}_{u=1}^U, \{\boldsymbol{y}_i\}_{i=1}^I, \{b_i\}_{i=1}^I\Big]$$

1.2

1.2.i Update for b_u:

The derivative of the objective by b_u is:

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial b_u} &= -2(r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i) + 2\lambda_{b_u} b_u \\ &= -2e_{ui} + 2\lambda_{b_u} b_u\end{aligned}$$

Therefore, the update step is:

$$\boxed{b_u^{new} \leftarrow b_u^{old} + \eta\big[e_{ui} - \lambda_{b_u} b_u^{old}\big]}$$

Where η is the learning rate hyper parameter.

1.2.ii Update for b_i:

The derivative of the objective by b_i is:

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial b_i} &= -2(r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i) + 2\lambda_{b_i} b_i \\ &= -2e_{ui} + 2\lambda_{b_i} b_i\end{aligned}$$

Therefore, the update step is:

$$\boxed{b_i^{new} \leftarrow b_i^{old} + \eta\big[e_{ui} - \lambda_{b_i} b_i^{old}\big]}$$

1.2.iii Update for $\boldsymbol{x}_u$:

The derivative of the objective by $\boldsymbol{x}_u$ is:

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{x}_u} &= -2(r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)\boldsymbol{y}_i + 2\lambda_x \boldsymbol{x}_u \\ &= -2e_{ui}\boldsymbol{y}_i + 2\lambda_x \boldsymbol{x}_u\end{aligned}$$

Therefore, the update step is:

$$\boxed{\boldsymbol{x}_u^{new} \leftarrow \boldsymbol{x}_u^{old} + \eta\big[e_{ui}\boldsymbol{y}_i^{old} - \lambda_x \boldsymbol{x}_u^{old}\big]}$$

1.2.iv Update for $\boldsymbol{y}_i$:

The derivative of the objective by $\boldsymbol{y}_i$ is:

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{y}_i} &= -2(r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)\boldsymbol{x}_u + 2\lambda_y \boldsymbol{y}_i \\ &= -2e_{ui}\boldsymbol{x}_u + 2\lambda_y \boldsymbol{y}_i\end{aligned}$$

Therefore, the update step is:

$$\boxed{\boldsymbol{y}_i^{new} \leftarrow \boldsymbol{y}_i^{old} + \eta\left[e_{ui}\boldsymbol{x}_u^{old} - \lambda_y \boldsymbol{y}_i^{old}\right]}$$

1.3

Below is a pseudo code for the SGD algorithm for the full model of the Matrix Factorization. Notice that the latent vectors dimension d, the variance of the normal distribution σ^2, and the learning rate η are part of the model hyper parameters set. As such, their values are set as part of hyper-parameters tuning stage.

Algorithm 1 Matrix Factorization Full Model using Stochastic Gradient Descent Algorithm

Input: A training set $(u, i, r_{ui}) \in D^t$, a validation set $(u, i, r_{ui}) \in D^v$, an hyper-parameters set H

Output: Model parameters: $\forall i, u : \boldsymbol{y}_i, \boldsymbol{x}_u \in \mathbb{R}^d,\ b_u, b_i \in \mathbb{R}$

Initialize:
$\mu = \frac{\sum_{D^t} r_{ui}}{|D^t|}$
$\forall u : b_u = \textit{user } u \textit{ average rating} - \mu;\ \boldsymbol{x}_u \sim \mathcal{N}(0,\ \sigma^2)$
$\forall i : b_i = \textit{item } i \textit{ average rating} - \mu;\ \boldsymbol{y}_i \sim \mathcal{N}(0,\ \sigma^2)$
while model not converge **do**
  for each $(i, u, r_{ui}) \in D_t$ **do**
    $e_{ui} \leftarrow r_{ui} - (\mu + b_u + b_i + x_u^T y_i)$
    $b_u \leftarrow b_u + \eta\left[e_{ui} - \lambda_{b_u} b_u\right]$
    $b_i \leftarrow b_i + \eta\left[e_{ui} - \lambda_{b_i} b_i\right]$
    $\boldsymbol{x}_u \leftarrow \boldsymbol{x}_u + \eta\left[e_{ui}\boldsymbol{y}_i - \lambda_x \boldsymbol{x}_u\right]$
    $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i + \eta\left[e_{ui}\boldsymbol{x}_u - \lambda_y \boldsymbol{y}_i\right]$
  $\eta \leftarrow 0.9\eta$

For details on how we have judged whether the model has converged, please follow Section 1.5.

1.4

The hyper-parameters for the full model of SGD algorithm are:

1) **Learning rate:** η - determines the step size to be taken in the opposite direction of the parameters gradient for each update step, in the way towards a minimum of a loss function.

2) **The vector dimension:** d - the dimension of the latent vectors $\boldsymbol{x}_u$ and $\boldsymbol{y}_i$, which represent each of the users' and items' in the data.

3) **Regularization coefficients:** λ_{b_u}, λ_{b_i}, λ_x, λ_y - four regularization coefficients for each group of the model parameters. These coefficients help to prevent from the model to over-fit to the train set.

4) **Standard deviation of parameters initialization:** σ - we sample random values from the normal distribution $\mathcal{N}(0, \sigma^2)$ as a starting point of the users and items latent vectors.

As the search method for finding the best hyper-parameters set, we have used the **Randomized Grid Search.** In this method we **control the number of hyper-parameters sets we examine,** and with only 100 samples we can gain one of the **top 3% sets of hyper-parameters with 95% confidence**[1]**,** and thus reducing dramatically the search space.

The best hyper-parameter set we have found is:

η	d	λ_{b_u}	λ_{b_i}	λ_x	λ_y	σ
0.01	40	0.01	0.01	0.001	0.01	0.0001

1.5

At the end of each epoch, after updating the parameters of the model with all the training set examples, we will **calculate the objective function value** of the current model (with the current parameters) **over the validation** set examples. To **check for convergence,** we will **allow the objective function on validation to increase three times,** and stop the parameters' updating steps after the third time of increment. It is worth mention that the returned model is the one that got the lowest value of the objective function on the validation set examples before the early stop.

1.6

After **finding the optimal hyper-parameters set in the tuning step** (those who have minimized the RMSE measure on the validation set), we will use it for the prediction on the test set. The model that we will use for this prediction will be **trained over the training set and estimate the model RMSE with the validation set.** Finally, we will use this trained model to predict the ratings in the test set.

1.7

Our SGD implementation is in `SGD_optimization.py` file, attached to the submission.

Our **main work items** in the search for the final model were:

1) **Calculating the μ and biases from the training data** as "smart" starting point. They are calculated as the average rating from the whole training data for μ, and the subtraction of μ from the average rating for each item and user for the biases.

2) **Running the Bias model** for 10 epochs and return the best biases found on validation data. We don't continue with the epochs until the bias model converges since these parameters keep to optimize with the full model updates anyway.

[1] `https://towardsdatascience.com/hyper-parameter-tuning-with-randomised-grid-search-54f865d27926`

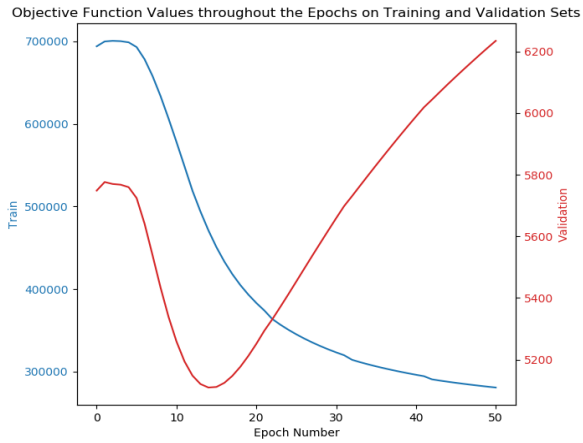

Figure 1: The model's objective function value on the train set (blue) and the validation set (red) as a function of the epoch number, using latent vectors in $\mathbb{R}^{50}$. The difference of the scale between the graphs are expected since the objective function has no consideration such as average over the amount of examples while the train set is much larger than the validation set.

3) **Running the Full model** until convergence, with the biases found from the Bias model. We used clipping function over the model predictions on both the bias and full models while training.

$$clipping(\hat{r}_{ui}) = \min\Big(\max(\hat{r}_{ui}, 1), 5\Big) = \begin{cases} 1 & \hat{r}_{ui} \leq 1 \\ \hat{r}_{ui} & 1 < \hat{r}_{ui} < 5 \\ 5 & \hat{r}_{ui} \geq 5 \end{cases}$$

4) **Examining the number of epochs until the model converge** for different dimension of the latent vectors. For example, for the dimension $d = 50$, we noticed the model converges after 14 epochs before over fitting to the train data (Figure 1).

5) **Tuning of the hyper-parameters** using **Randomized Grid Search** as the search method, while trying to optimize the RMSE on validation set. For more details about this method, follow Section 1.4. We randomly drew one set of hyper-parameters from the full grid to compute the RMSE at every iteration. We tried the Nelder-Mead method as well, but it took too long to find the best hyper-parameters for each dimension.

6) Lastly, we **trained the final model** with the **best hyper parameters values set** found on the previews part. We also used the clipping function on the model prediction of the test examples.

1.8

Evaluation	RMSE	MAE	R^2
Mean Model	1.1271	0.9556	-0.0216 [2]
Bias Model	0.9595	0.7509	0.2595
Full Model	**0.8905**	**0.6851**	**0.3622**

[2]It is not surprising that the R^2 coefficient is negative. It happens when a model fits worse than a horizontal line of the mean data (the null model). Indeed, the suggested Mean Model uses the mean rating of the training set as the prediction, whereas the null model of the R^2 is defined as the mean rating of the validation set. Thus the former performs worse. If we use the mean rating of the validation set as the prediction, we get that R^2 is very close to zero as expected.

Where **Mean Model** is the model that contains only the mean rating of the training set as the prediction: $\hat{r}_{ui} = \mu$.

2 Alternating Least Squares

2.1

The **objective** of a regression model with global bias, user bias, item bias and L2 regularization is:

$$\min_{\boldsymbol{\theta}} \sum_{(u,i,r_{ui})\in D} (r_{ui} - \mu - b_u - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)^2 + \lambda_x \sum_u ||\boldsymbol{x}_u||^2 + \lambda_y \sum_i ||\boldsymbol{y}_i||^2 + \lambda_{b_u} \sum_u b_u^2 + \lambda_{b_i} \sum_i b_i^2$$

Where $\boldsymbol{\theta}$ is the same as in Section 1.1.
This function is the same as the SGD objective function, only the optimization method has changed while the model remains the same.

2.2

The update steps for each parameter of the model is:

2.2.i Update for b_u:

$$b_u^{new} \leftarrow (|D_u| + \lambda_{b_u})^{-1} \sum_{D_u} (r_{ui} - \mu - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)$$

2.2.ii Update for b_i:

$$b_i^{new} \leftarrow (|D_i| + \lambda_{b_i})^{-1} \sum_{D_i} (r_{ui} - \mu - b_u - \boldsymbol{x}_u^T \boldsymbol{y}_i)$$

2.2.iii Update for $\boldsymbol{x}_u$:

$$\boldsymbol{x}_u^{new} \leftarrow \left(\sum_{D_u} \boldsymbol{y}_i \boldsymbol{y}_i^T + \lambda_x \boldsymbol{I}_d \right)^{-1} \sum_{D_u} (r_{ui} - \mu - b_i - b_u) \boldsymbol{y}_i$$

2.2.iv Update for $\boldsymbol{y}_i$:

$$\boldsymbol{y}_i^{new} \leftarrow \left(\sum_{D_i} \boldsymbol{x}_u \boldsymbol{x}_u^T + \lambda_y \boldsymbol{I}_d \right)^{-1} \sum_{D_i} (r_{ui} - \mu - b_i - b_u) \boldsymbol{x}_u$$

Where $\boldsymbol{I}_d$ is the identity matrix of size d, as the dimension of the latent vectors, and $D_{u'}$ and $D_{i'}$ are subsets of the data set that is relevant to user u' and item i', respectively.

2.3

Below is a pseudo code for the ALS algorithm for the full model of the Matrix Factorization. Similarly to SGD in Section 1.3, the dimension d, and the variance of the normal distribution σ^2 are part of the model hyper parameters set. As such, their values are set as part of hyper-parameters tuning stage.

Algorithm 2 Matrix Factorization Full Model using Alternating Least Squares Algorithm

Input: A training set $(u, i, r_{ui}) \in D^t$, a validation set $(u, i, r_{ui}) \in D^v$, an hyper-parameters set H

Output: Model parameters: $\forall i, u : \boldsymbol{y}_i, \boldsymbol{x}_u \in \mathbb{R}^d$, $b_u, b_i \in \mathbb{R}$

Initialize:

$\mu = \frac{\sum_{D^t} r_{ui}}{|D^t|}$

$\forall u : b_u = user\ u\ average\ rating - \mu$; $\boldsymbol{x}_u \sim \mathcal{N}(\boldsymbol{x}_u; 0, \sigma^2)$

$\forall i : b_i = item\ i\ average\ rating - \mu$; $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{y}_i; 0, \sigma^2)$

while model not converge **do**

  for $u \leftarrow 1$ to U **do**

    $b_u \leftarrow (|D_u| + \lambda_{b_u})^{-1} \sum_{D_u} (r_{ui} - \mu - b_i - \boldsymbol{x}_u^T \boldsymbol{y}_i)$

    $\boldsymbol{x}_u \leftarrow (\sum_{D_u} \boldsymbol{y}_i \boldsymbol{y}_i^T + \lambda_x \boldsymbol{I}_d)^{-1} \sum_{D_u} (r_{ui} - \mu - b_u - b_i) \boldsymbol{y}_i$

  for $i \leftarrow 1$ to I **do**

    $b_i \leftarrow (|D_i| + \lambda_{b_i})^{-1} \sum_{D_i} (r_{ui} - \mu - b_u - \boldsymbol{x}_u^T \boldsymbol{y}_i)$

    $\boldsymbol{y}_i \leftarrow (\sum_{D_i} \boldsymbol{x}_u \boldsymbol{x}_u^T + \lambda_y \boldsymbol{I}_d)^{-1} \sum_{D_i} (r_{ui} - \mu - b_u - b_i) \boldsymbol{x}_u$

For details on how we have judged whether the model has converged, please follow Section 2.5.

2.4

The hyper-parameters for the full model of ALS algorithm are:

1) **The vector dimension:** d - the dimension of the latent vectors $\boldsymbol{x}_u$ and $\boldsymbol{y}_i$, which represent each of the users' and items' in the data.

2) **Regularization coefficients**: λ_{b_u}, λ_{b_i}, λ_x, λ_y - four regularization coefficients for each group of the model parameters. This coefficients help to prevent from the model to over-fit to the train set.

3) **Standard deviation of parameters initialization:** σ - we sample random values from the normal distribution $\mathcal{N}(0, \sigma^2)$ as starting point of the users and items latent vectors.

For the search method, we have used the **Randomized Grid Search** for finding the best hyper-parameters set (the same as for the SGD optimization). For more details about the method, please follow Section 1.4.

The best hyper-parameter set we found is:

d	λ_{b_u}	λ_{b_i}	λ_x	λ_y	σ
5	0.01	0.001	0.001	0.001	0.001

2.5

Similarly to the SGD in Section 1.5, **at the end of each epoch,** after updating all the parameters of the model, we will **calculate the objective function value** of the current model (with the current parameters) **over the validation** set examples. To **check for convergence,** we will **allow the objective function on validation to increase three times,** and stop the parameters' updating steps after the third time of increment. It is worth mention that the returned model is the one that got the lowest value of the objective function on the validation set examples before the early stop.

2.6

Our ALS implementation is in `ALS_optimization.py` file, attached to the submission.

Our **main work items** in the search for the final model were similar to the steps performed on Section 1.7 above, so they will be presented in titles only.

1) **Calculating from the μ and biases the training data** as "smart" starting point.
2) **Running the Bias model** for 10 epochs and return the best biases found on validation data.
3) **Running the Full model** until convergence, with the biases found from the Bias model.
4) **Examining the number of epochs until the model converge** for different dimension of the latent vectors.
5) **Hyper parameters tuning** using **Randomized Grid Search** as the search method.
6) **Training the final model** with the **best hyper parameters values set** found on the previews part. We used clipping function on the model prediction of the test examples (as in Item 3) Section 1.7).

2.7

Evaluation	RMSE	MAE	R^2
Mean Model	1.1271	0.9556	-0.0216 [3]
Bias Model	0.9571	0.7491	0.2632
Full Model	**0.9226**	**0.7094**	**0.3154**

Where **Mean Model** is the model that contain only the mean rating in the training data as the prediction of the rating: $\hat{r}_{ui} = \mu$.

[3] See explanation about the negative R^2 coefficient in Section 1.8.

2.8

Comparison between the ALS and SGD algorithms:
Implementation: The SGD algorithm requires more epochs to converge as oppose to the ALS algorithm (compatible with what was in the lecture). In addition, the SGD algorithm is easier to implement compared to the ALS algorithm which has more complex update steps and requires matrix inversion.

Training: The hyper-parameters values are different between the two models - the dimension d and most of regularization coefficients are lower in the ALS than in the SGD, while σ is higher in the SGD. Also, both train and validation errors of the ALS are higher than the SGD errors.

Quality: The full SGD model is better than the ALS model in all the measures. However, the bias model trained with ALS gets better measures than the bias model trained with SGD, probably because the second was not able to converge after only 10 epochs of training while the ALS was able to converge after the predefined number of epochs.