The Chinese University of Hong Kong, Shenzhen
School of Data Science
DDA 4260: Networked Life
(Term 2, 2022/23)

Project Report

Group 1:

Joseph Ariel Christopher Teja (120040002)
Darren Boesono (120040022)

I. Background

Recommendation systems, which improve the user experience in different websites, have become an important part of our everyday lives. These systems provide the user with personalized recommendations of products, services and content that are based on their preferences, browsing behavior and historical data. Rapid growth of the Internet, and growing amounts of data collected by users have led to an increase in demand for efficient recommendation systems over these past few years.

Recommendation systems have had a major impact on the film industry. It can be hard for users to discover and choose movies that correspond with their preferences, given an overwhelming amount of films available on a variety of streaming platforms. Movie rating recommendation systems aim to address this issue by predicting and suggesting movies that users are likely to enjoy, based on their previous interactions and preferences. It is not only user satisfaction that is enhanced by these recommendations, but also helps service providers retain customers and generate revenue.

In this project, we explore the application of two machine learning models, Linear Regression and Restricted Boltzmann Machine (RBM) and implement it using the Python programming language.

A linear regression is an easy and widely used taught learning algorithm that maps the relationships between a dependency variable and one or more independently defined variables. Linear regression can be used to predict user ratings for a movie based on its characteristics, such as the user's demographics, film types and historical rating data, in the context of movie recommendation. By training the model on a dataset of user ratings and movie features, it learns to predict a user's rating for a movie, which can then be used to generate personalized recommendations.

Restricted Boltzmann Machines (RBMs) are a type of unsupervised learning algorithm, specifically a generative stochastic artificial neural network. RBMs consist of two components: a visible layer representing the input data,eg. movie ratings or abstract features and an invisible layer which can be accessed when you have to import some feature or pattern. To learn latent representations of user preference and film features based on observed pairs of user movie ratings, an RBM can be used in a recommendation system. Once trained, the RBM can predict a user's preference for a movie, which can then be translated into personalized movie recommendations.

II. Methodology

A. Linear Regression

When implementing the linear regression model, our task is to find the optimal value b in order to solve the optimization problem below:

$$minimize_b \, ||Ab - c||_2^2$$

Where $A \in \mathbb{R}^{n \times n}$ is matrix that contains information from the training data and $c \in \mathbb{R}^n$ is given results. To solve for b, We can expand the objective function:

$$(Ab - c)^T (Ab - c) = b^T A^T Ab - 2b^T A^T c + c^T c$$

Taking the derivative w.r.t b and set the equation equal to 0:

$$2A^TAb - 2A^Tc = 0$$

Solving for b, we have:

$$b = (A^TA)^{-1}A^Tc$$

Meanwhile, in case of overfitting the we penalized the objective function using a regularization parameter, λ. The optimization problem becomes:

$$minimize_b \, ||Ab - c||_2^2 + \lambda||b||_2^2$$

Solving for b by expanding the objective function and taking its derivative w.r.t b and setting it equal to 0 gives us:

$$b = (A^TA + \lambda I)^{-1}A^Tc$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix.

B. Restricted Boltzmann Machine (RBM)

RBM has a bipartite graph structure and has the typical ANN structure. It comprises visible and hidden layers stacked on top of each other. The visible layer passes the input values to the hidden layer nodes through weighted connections. The weights of these connections are learned by the RBM during training. The hidden layer will then do a non-linear transformation on the output of the visible layer.

Let us first define our notations. Let W denote the weights of the connections in an RBM. Specifically, the weight of the connection between the ith node in the visible layer and the jth node in the hidden layer is denoted as W_{ij}. We represent the input (i.e., the value of the visible layer) with a vector v, where v_i represents the ith value of the visible layer. Let h_j the jth value of the hidden layer.

Let us now go through RBM's inference process. First, when the visible layer receives an input, the input is transformed and sent to the hidden layer. This transformation is done following this equation.

$$P(h_j = 1|\, v) = \sigma(\sum_{i\in V} v_i W_{ij}),\ where\ \sigma(x) = \frac{1}{1+e^{-x}}$$

To pass values from the hidden layer to the visible layer, we use the following equation.

$$P(v_i^k = 1|\, h) = softmax(\sum h_j W_{ij}^k),\ where\ softmax(x_k) = \frac{e^{x_k}}{\sum_{l\in K} e^{x_l}}$$

$P(v_i^k = 1|\, h)$ represents the likelihood of movie i being rated k points by the current user. To get the final rating for this movie, we can adopt one of two approaches. The first option is to select the maximum value in the vector v_i, while the second is to calculate the weighted average of the elements of v.

The original RBM (without extensions) is trained using vanilla gradient descent where the weight is updated as follows, let γ denote the learning rate, G^+ denote the positive gradient. G^- denote the negative gradient:

$$W = W + \gamma \cdot (G^+ - G^-)$$

$$G_{ijk}^+ = P(h_j = 1|v)v_i^k$$

$$G_{ijk}^- = P(h_j = 1|v')v'^k_i$$

To try and further improve performance, we also employ several of the extensions mentioned in the project description. The first of which is momentum which is employed as follows where m_n is the momentum at a time n and α is the momentum coefficient:

$$m_n = \alpha m_{n-1} + (1 - \alpha) G_n$$

We then update the gradient as the value of m_n.

The second extension we employ is the adaptive learning rate which simply means that we reduce the learning rate if the RMSE decreases slowly or even stops decreasing. Next, we implemented early stopping which stops the training process entirely if the RMSE too slowly stops decreasing or even increases which implies performance degradation.

The next extension we implemented is regularization which is implemented following this equation:

$$W = W - \gamma \frac{\lambda}{2}, \; where \; \lambda \; is \; the \; regularization \; parameter$$

The next extension is mini-batch which is implemented in order to reduce training time. Next, we also added biases in order to make the ANN complete. Let b_v denote the biases for the visible layer and let b_h denote the biases for the hidden layer. We derive those two values using the following equations:

$$b_v = v - v'$$

$$b_h = P(h = 1|v) - P(h = 1|v')$$

The addition of biases meant that the original equations for the learning and unlearning would be modified slightly by adding b_v and b_h to the corresponding weights.

Lastly, we employ the neighborhood method for the baseline prediction. The theory itself completely follows the one in Q4 of the slides while the implementation closely follows the code already made for Exercise 2 of Homework 2.

III. Implementation

A. Linear Regression

Matrix A is sparse matrix corresponding to the raters and movies. For each rating in the dataset, assign a binary value of 1 for the corresponding user and movie in the feature vector, and 0 for the rest of the elements. For example, if there are 5 users and 5 movies in total, and user 2 rates movie 3, the feature vector will be [0, 1, 0, 0, 0, 0, 0, 1, 0, 0]. This can be done using the $getA()$ function. Meanwhile, the vector c is created by subtracting the actual ratings by the mean value of all rates. We can get the mean value of all rate using the $rBar()$ function. We then get the vector b at optimal value using the information explained in the methodology section. The b vector contains the movie and user bias. To compute the prediction rating we use the following formula:

$$\widehat{r_{ui}} = \bar{r} + b_u + b_i$$

Where b_u and b_i are the user's and movie's bias respectively. We get the prediction by using the $predict()$ function. Our final prediction will be restricted in [1, 5] and if one prediction exceeds the value 5, it will be considered as 5 and the same goes if the prediction output is lower than 1.

B. Restricted Boltzmann Machine (RBM)

The implementation of basic RBM closely follows the project description and template. As such, it would be quite redundant to repeat it in sections. As such, this section will mainly focus on the extensions.

For extensions such as momentum, regularization, and biases. We simply implement it according to the formula described in the methodology. For the neighborhood model, as mentioned in the methodology section, we simply reuse the code used for Exercise 2 of Homework 2.

For adaptive learning rates, we implement it by checking if the difference between the average value of the last x RMSEs and the average value of the last y RMSEs is smaller than a certain threshold. If the difference is smaller than the threshold, the learning rate is decreased by z. x, y, z are all modifiable parameters.

For early stopping, the implementation is similar to the one for adaptive learning rates with the only difference being the termination of the training process entirely as a result and not just the decrease in learning rates.

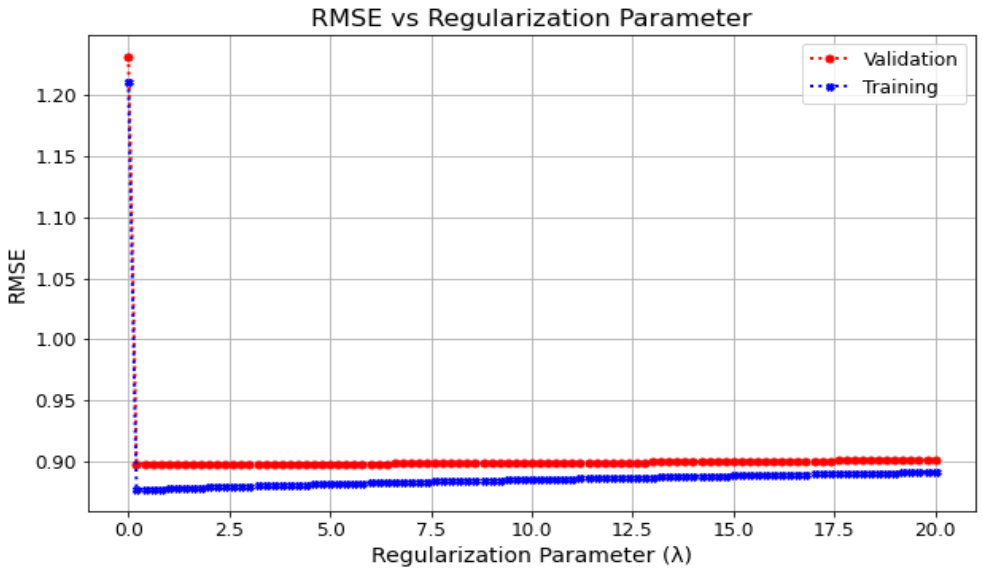
For mini-batch, we simply add a counter for the loop in the number of users. If the counter reaches the mini-batch size, then we update the weights and reset the counter and if the counter has not reached the mini-batch size, then we simply continue the loop whilst adding the counter.

III. Results

A. Linear Regression

101 points are taken from the interval [0, 20]. The RMSE graph along the different λ values in the x-axis is as follows:

The minimum RMSE is 0.8978, when $\lambda = 3.2$ from the validation dataset. While for λ=0, the RMSE for validation and training are 1.2320 and 1.2111 respectively. It could be seen that, with regularization, even a small λ could bring about critical changes of RMSE.

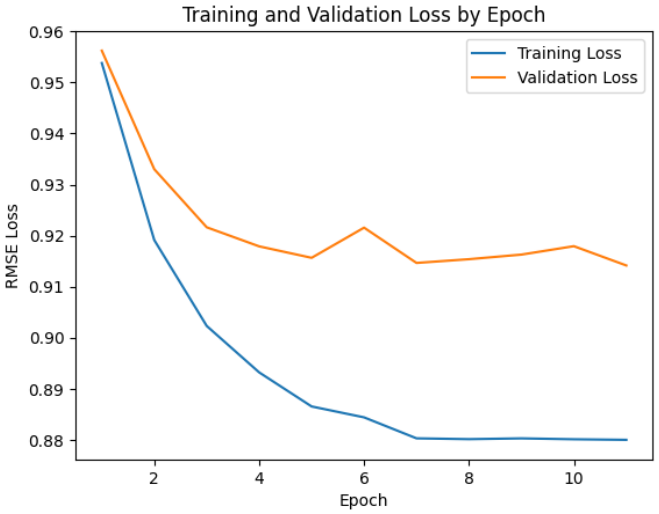


After the dramatic decrease, the RMSE will slightly increase along with λ.

B. Restricted Boltzmann Machine (RBM)

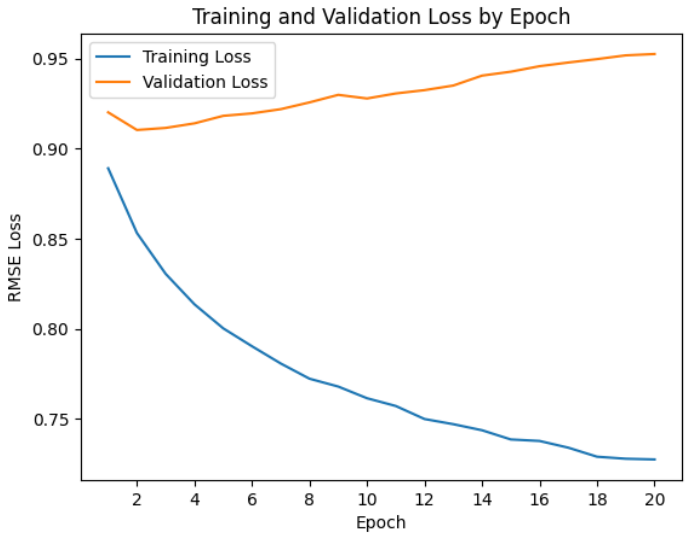
For the RBM, in our submission it can be seen that week 1 has a better performance compared to our submission in week 4. Week 1 is the plain RBM without extensions while week 4 is the RBM with full extensions. After debugging this abnormality, this turned out to be caused by a faulty mini-batch implementation where the counter did not update which in turn meant the weights were not being updated the entire time. This in turn made it so that the weights were not being updated every time

which explains why the performance was much worse compared to the plain RBM implementation. We have since corrected this issue and the performance was better compared to the plain RBM. From this point onwards, we will use that model as the one we analyze. Our revised model achieved a training loss of 0.879989 and a validation loss of 0.914150. Here is the training plot.



Note that this result is before the neighborhood model is implemented as we have no way of checking performance after the neighborhood model is used. The parameters used are K = 5, F = 20, epochs = 100, γ= 0.02, α= 0.2, λ= 0.1, mini-batch size = 1, with early stopping and adaptive learning rate being implemented which is why the training is stopped at epoch 11.

We will now compare the results from RBM with extensions to the plain RBM model with parameters K=5, F = 20, epochs = 20, γ= 0.1. It achieved 0.910299 best validation loss while it ended with 0.952472 validation loss and 0.727364 training loss. We can observe that the performance degrades quite quickly and although this beat the one with extensions in terms of best validation loss, it eventually performed worse compared to the one with extensions.

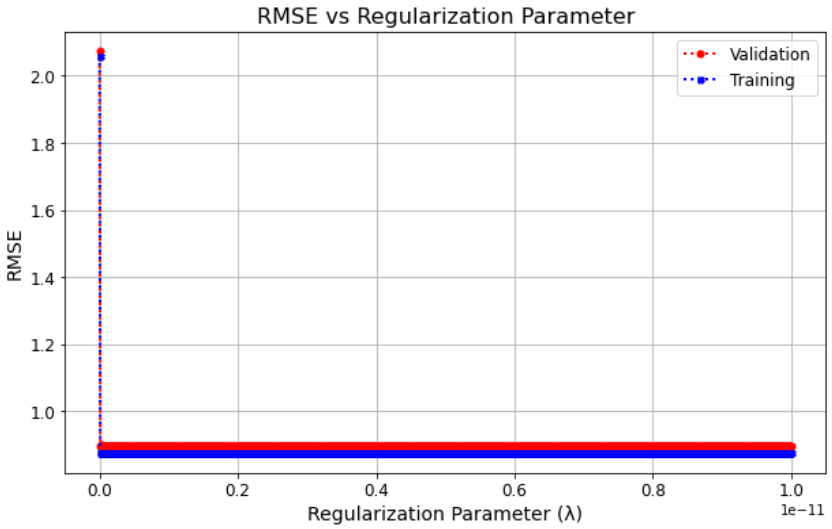


IV. Analysis

A. Linear Regression

While the RMSE meet its minimum value at $\lambda = 3.2$, there was a dramatic decrease occurred when λ is close to 0 for both validation and training dataset. As you can see from the RMSE graph, there is a dramatic decrease as the parameter λ moves along the x-axis. Therefore, 1001 points are taken from the interval $[10^{-15}, 10^{-11}]$. The results are as follows:

It is indicated above that critical drop of RMSE at about $\lambda = 0.0000001$ and it will continue to stay around about 0.8 RMSE.



B. Restricted Boltzmann Machine (RBM)

In both cases, the training loss lowers much quicker than the validation loss, implying that perhaps overfitting occurred. From the results section we can observe that although the RBM with extensions performs better than the plain one, the plain one still beats it when it comes to best validation loss. However, as the plain RBM degrades quickly, the one with extensions ultimately performs better. As such, we infer that from all the extensions we implemented, at least using the parameters we did, the extensions that matter most are early stopping and adaptive learning rates as they help prevent performance degradation which in both the case of the one with extensions and the plain one is mainly overfitting. This is also supported by the fact that the validation losses are not that different between both the plain RBM with the ones with extensions and the main reason why plain RBM performs worse is degradation in performance.

Ideally, it would be best to do a thorough study of optimal parameters. However, the nature of this program makes this computationally expensive and time-consuming as one run through the program takes several minutes and from personal experience, doing grid search takes hours and even days to run which is not feasible with the amount of memory the local computer has.

V. Conclusion

A. Linear Regression

Linear regression is a suitable and efficient method for data exhibiting linear patterns. However, the Netflix datasets demonstrate a significant deviation from linearity. To achieve better precision, it is essential to employ more sophisticated algorithms such as the Restricted Boltzmann Machine (RBM).

B. Restricted Boltzmann Machine

From our results, it can be seen that RBM performs quite well. At the very least, it performs better than linear regression. However, more time is needed to pick out better parameters for our model as it can severely impact the performance. Additionally, RBM with extension performs better than the plain one. However, the wrong parameters may cause it to perform worse than the plain RBM. As such, picking suitable parameters is very important. However, as previously stated, to pick them out is very time consuming as one run through one combination takes quite a while which makes doing grid search quite a heavy workload. We can also perhaps further improve the model by employing more sophisticated gradient descent techniques and employ cross-validation.