WEEK 3

Objective : To develop a Graph Neural Network for Social Recommendation

Topic : Learn Multi-Layer Perceptron and General Matrix Factorization

Multi-Layer Perceptron –

A perceptron is the basic unit powering deep learning. A perceptron can also be realized as an artificial neuron, which has the ability to learn and solve complex problems. It is basically a binary classification algorithm, that takes inputs and produces either a 1(yes) or 0(no) as their output.

The combination of such several perceptron stacked in different layers is known as Multi-Layer Perceptron or MLP. The union of these artificial neurons forms the Artificial Neural Network, a complex system which is mastered by training to solve tough problems. MLP takes real inputs which are then multiplied by weights and are fed into the activation function after adding the bias factor, ultimately producing a classification decision as output. Neural network algorithms learn by discovering better and better weights that result in a more accurate prediction, as weights allow the perceptron to evaluate the relative importance of each of the outputs. Backpropagation is one such common algorithm used to select the correct weight, which performs iterative back passes in attempt to minimize the loss between the correct and actual model prediction. The perceptrons uses different weights for each signal going from one perceptron of the input to the other of the hidden layer, and further to the output layer.

A three layer MLP is called a Non-deep or Shallow neural Network whiles a four or more layered MLP is called a Deep Neural Network. Deep Neural Network is more efficient than Shallow Neural Network because the decision functions uses activation function other than step functions, resulting in output of real values (usually between 0 and 1, or between -1 and 1).

Methods Used :

Matrix factorization is a specific type of collaborative filtering method that in general sense tries to approximate sub-matrices that are equivalent to a larger matrix. It represents user/item as a vector of latent features which are projected into a shared feature space. In this feature space, the user-item interactions could be modeled using the inner product of user-item latent vectors.

Matrix factorization grew in popularity because of the sparse and expanse of data being collected by companies.

<u>Dataset</u> :

The dataset rating.csv includes the rating information of movies. It includes five columns which are :

- ❖ userid
- ❖ productid
- ❖ categoryid
- ❖ rating
- ❖ helpfulness

  The figures provide information about a user rating a specific product of a particular category and the helpfulness of the rating to recommend further items to the user. The model is trained with the given stats of userid, productid and rating so as to build a proper recommender system.

<u>Packages and Libraries</u> :

- Different libraries used in the code are –
  1. Pandas
  2. Tensorflow
  3. Numpy
  4. Sklearn
  5. Math
  6. Keras
     - keras.layers
     - keras.optimisers
     - keras.utils
     - keras.regularizers
     - keras.initializers
     - keras.model

- Optimizers used in the code while trial and error –

1. SGD – SGD is an iterative method for optimizing an objective function with suitable smoothness properties. It picks single data from the dataset randomly per iteration so as to reduce the computations enormously.
2. RMSprop – Root Mean Square Propagation is a gradient based optimization technique designed for training artificial neural networks.
3. Adagrad – Adagrad is effectively SGD with a per-node learning rate scheduler built into the algorithm thus improving SGD by giving weights historically accurate leaning rates.
4. Adadelta – Adadelta optimization is a stochastic gradient descent method that is based on adaptive learning rate per dimension. It is basically a robust extension of Adagrad.
5. Adam – Adaptive Moment Estimation is a combination of Adagrad and RMSprop, which is used to calculate the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients.
6. AdaMax – It a variant of Adam based on the infinity room. AdaMax provides an important advanyage of being much less sensitive to the choice of the hyper-parameters.
7. Nadam – Nesterov – accelerated Adaptive Moment Estimation is a combination of NAG and Adam. It is employed for noisy or high curvature gradients. The learning process is accelerated by summing up the exponential decay of the moving averages for the previous and current gradient.
8. Ftrl – Follow the Regularized Leader optimizer is a per-coordinate learning rate system.

- A Metric function is used to judge the performance of a model. Functions mse() and mae() are Regression Metrics which used for loss determination or to calculate accuracy of the model.

<u>Findings</u> :

Latent Factors=20, Epochs=10, Number of layers=6, Softmax, Lecun_uniform

Optimizer	**Loss at start**	**Loss at end**	**Mae at start**	**Mae at end**	**Time for each epoch(approx.)**
Adagrad	11.1252	11.1252	3.1627	3.1627	134.8s
Adam	11.1250	11.1250	3.1627	3.1627	141.6s
RMSprop	11.1250	11.1250	3.1627	3.1627	185.4s
Adadelta	11.1250	11.1250	3.1627	3.1627	287s
Adamax	11.0493	11.0493	3.1627	3.1627	130.4s
Nadam	11.1250	11.1250	3.1627	3.1627	438.6s
Ftrl	11.1250	11.1250	3.1627	3.1627	391.8s
SGD	11.1252	11.1252	3.1627	3.1627	238.2s

<u>Experimental result</u> :

Based upon the findings with Latent Factor = 20, Epochs = 10 and Number of layers = 6, it can be inferred that Adamax is the most suitable optimizer for the model.

<u>Conclusion</u> :

For the third week we have completed making a neural network or deep learning based recommendation system through Multi-Layer Perceptron (MLP). We have also tried to use several optimizers and see which optimizer would best fit the model and finally chose the Adamax optimizer. This week, we learnt the implementation of neural networks and became more familiar with the deep learning Keras library.