# Movie Recommendation System using Stacked Autoencoders

CS 677 Data Science with Python

Paritosh Shirodkar
paritosh@bu.edu

# The Dataset

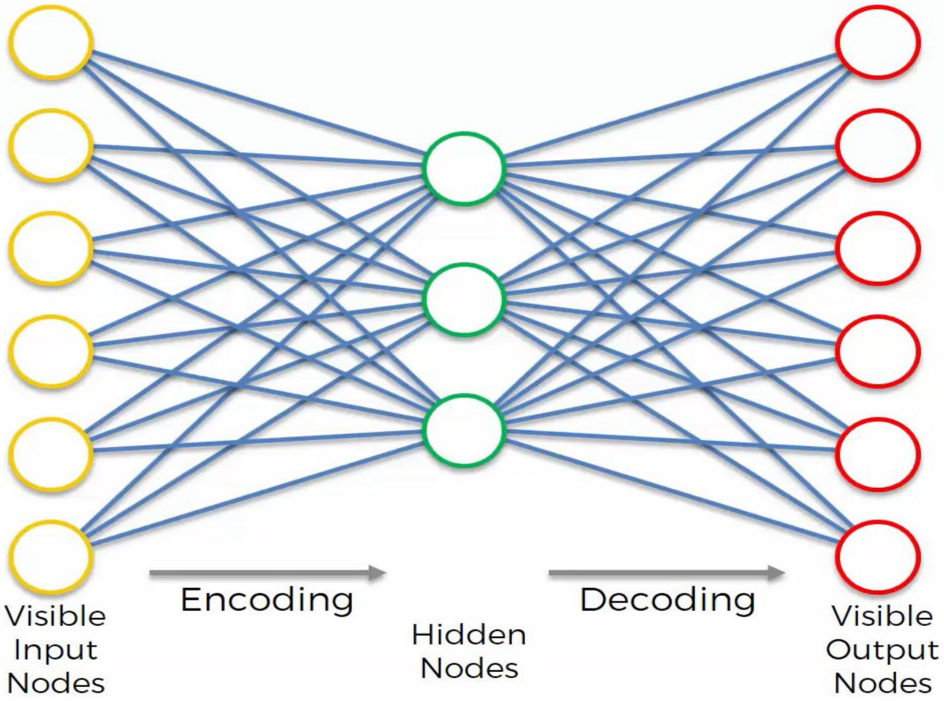
- I have used the MovieLens dataset for this project
- The dataset consists of 100000 instances
- The training set consists of 80000 instances
- The test set consists of 20000 instances
- The training set consists of the UserID, MovieID, Rating and Timestamp
- Additionally the data has information about popular genres, IMDb link, occupation, age and gender of the users who gave the ratings.
- Overall, there are 943 users, 1682 movies and 100000 ratings

Source: https://grouplens.org/datasets/movielens/

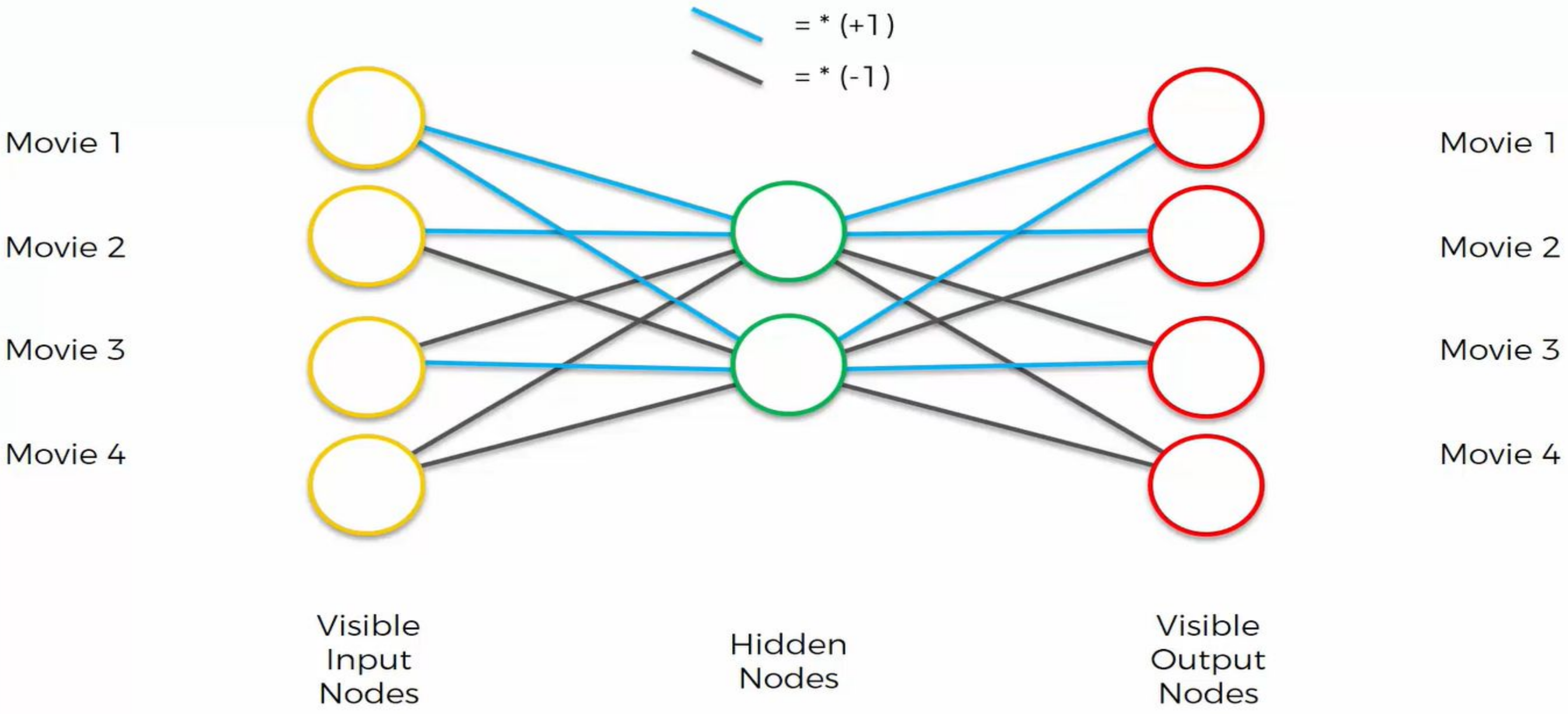
# What are Autoencoders ?

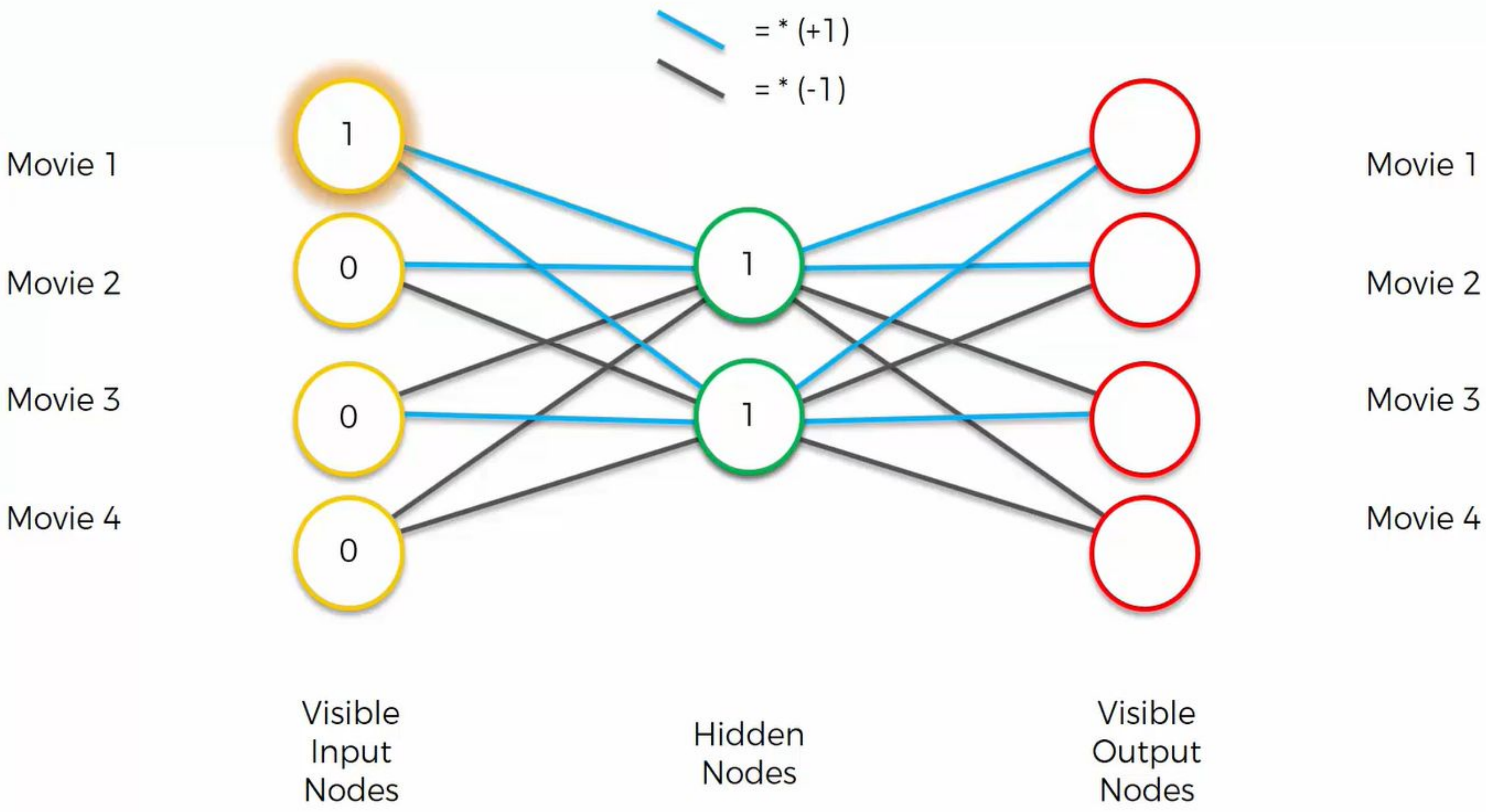
- It is a directed type of Neural Network
- It comes under the umbrella of unsupervised learning
- The philosophy behind Autoencoders is that it takes some inputs encodes them using the hidden neurons and then decodes them in an attempt to recreate the input
- Then the output is compared to the input and the error is computed
- Based on this error the weights of the network are adjusted to minimize the error

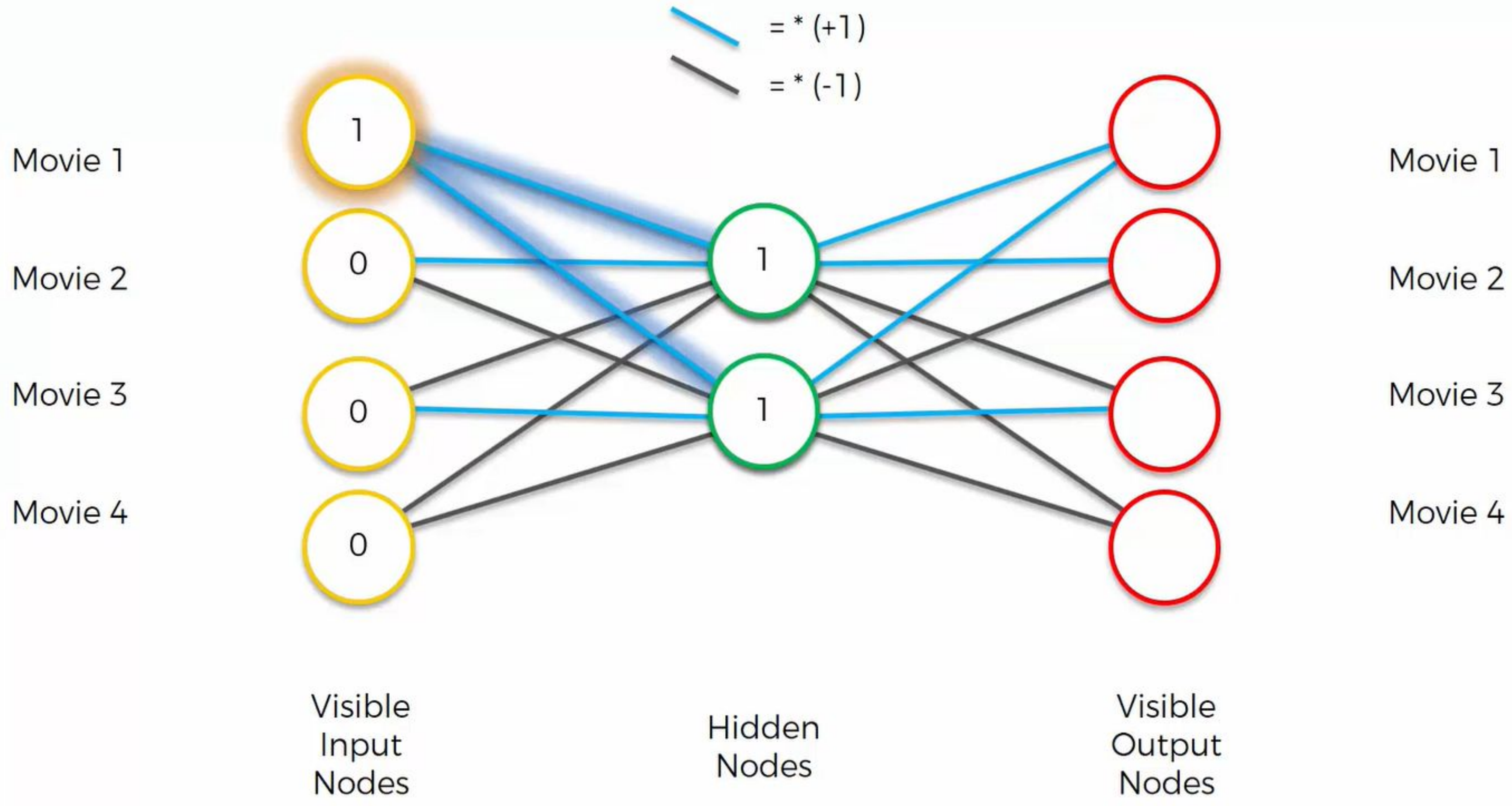
# Architecture of Autoencoders

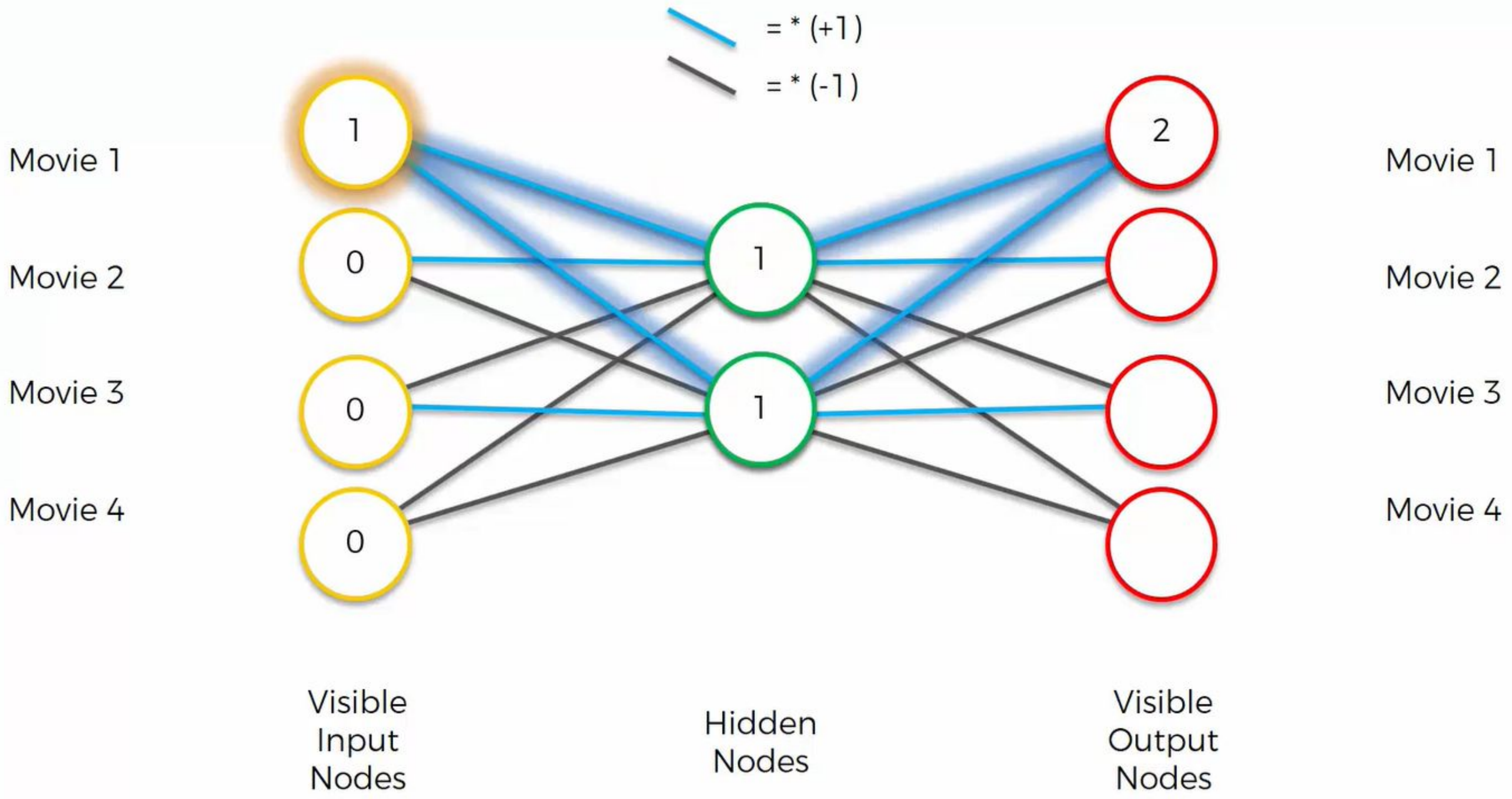


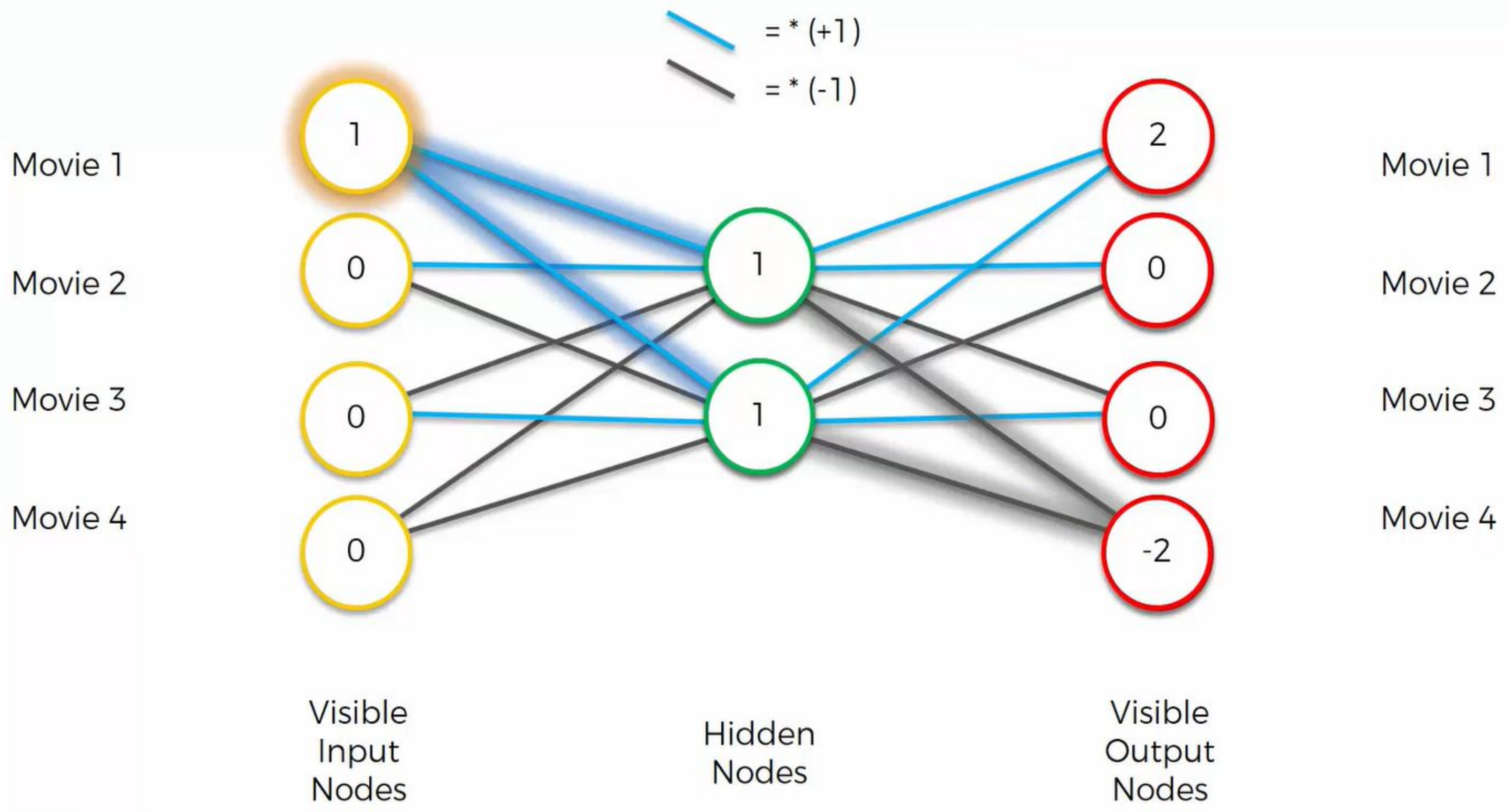
# The working on an intuitive level



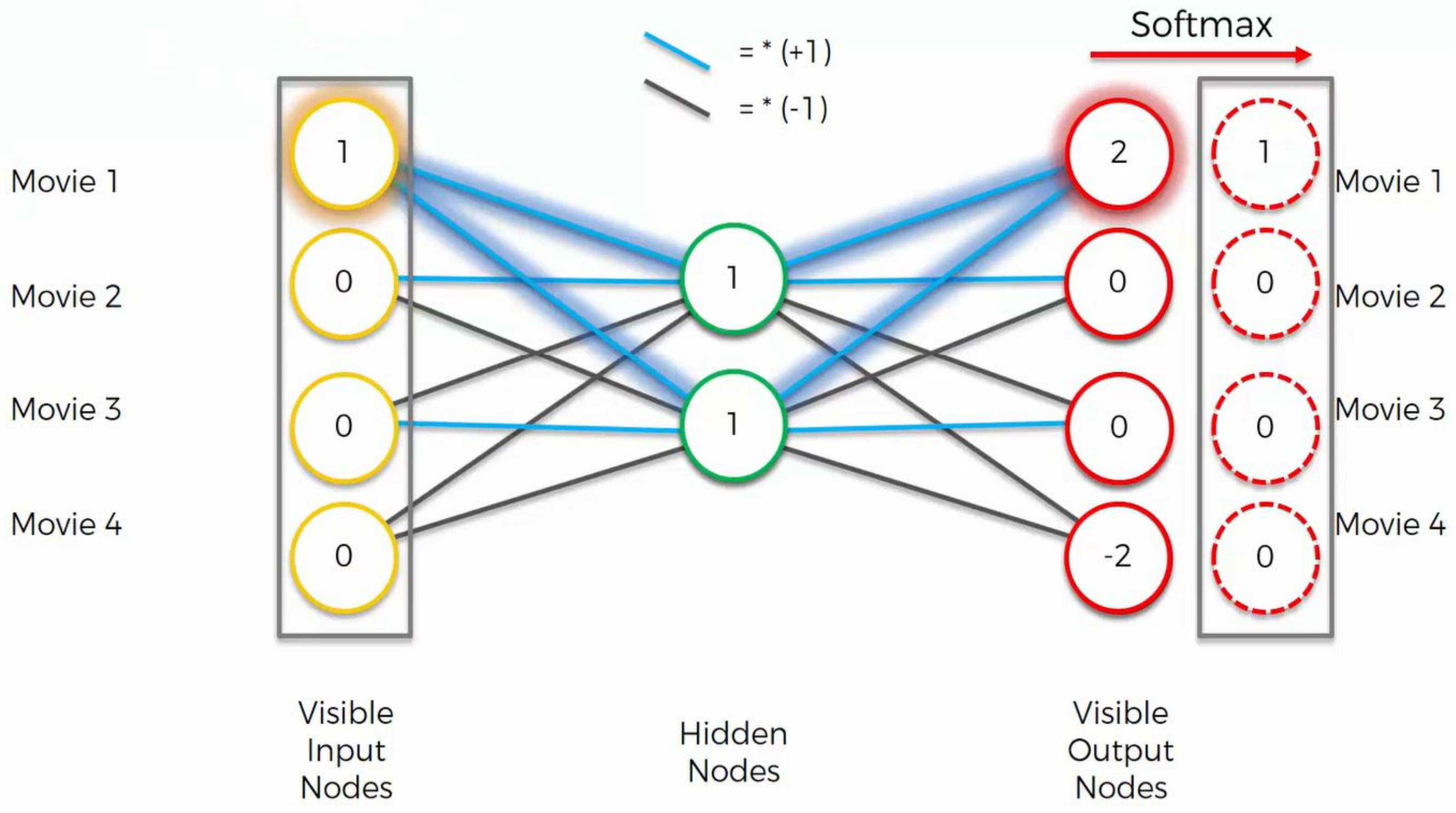

= * (+1)
= * (-1)
Movie 1
Movie 2
Movie 3
Movie 4
1
0
0
0
1
1
Movie 1
Movie 2
Movie 3
Movie 4
Visible Input Nodes
Hidden Nodes
Visible Output Nodes


= * (+1)
= * (-1)
Movie 1
Movie 2
Movie 3
Movie 4
1
0
0
0
1
1
Movie 1
Movie 2
Movie 3
Movie 4
Visible Input Nodes
Hidden Nodes
Visible Output Nodes

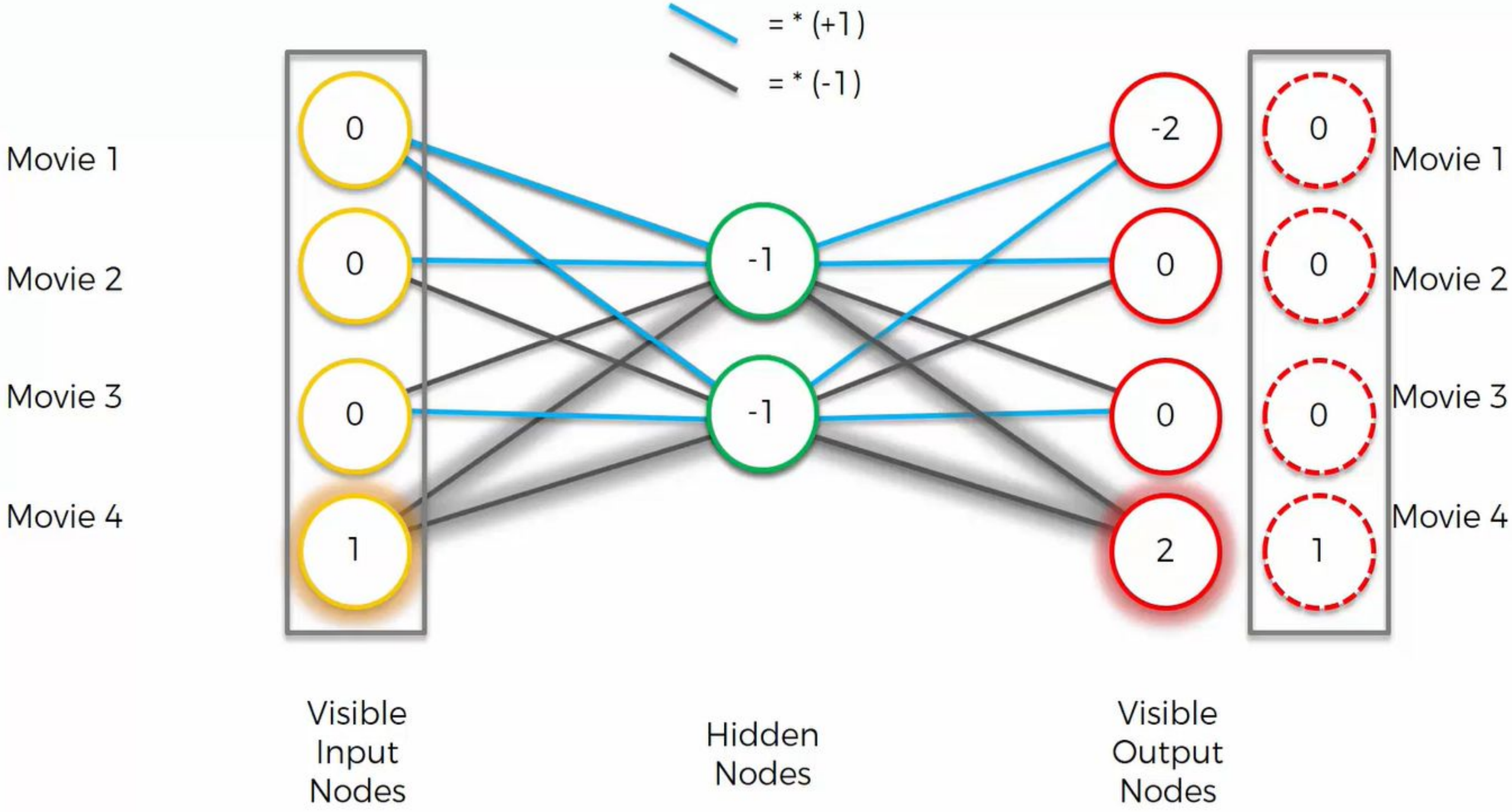

= * (+1 )
= * (-1 )
Movie 1
Movie 2
Movie 3
Movie 4
1
0
0
0
1
1
2
Movie 1
Movie 2
Movie 3
Movie 4
Visible
Input
Nodes
Hidden
Nodes
Visible
Output
Nodes



= * (+1)
= * (-1)
Movie 1
Movie 2
Movie 3
Movie 4
1
0
0
0
1
1
2
0
0
-2
Movie 1
Movie 2
Movie 3
Movie 4
Visible Input Nodes
Hidden Nodes
Visible Output Nodes



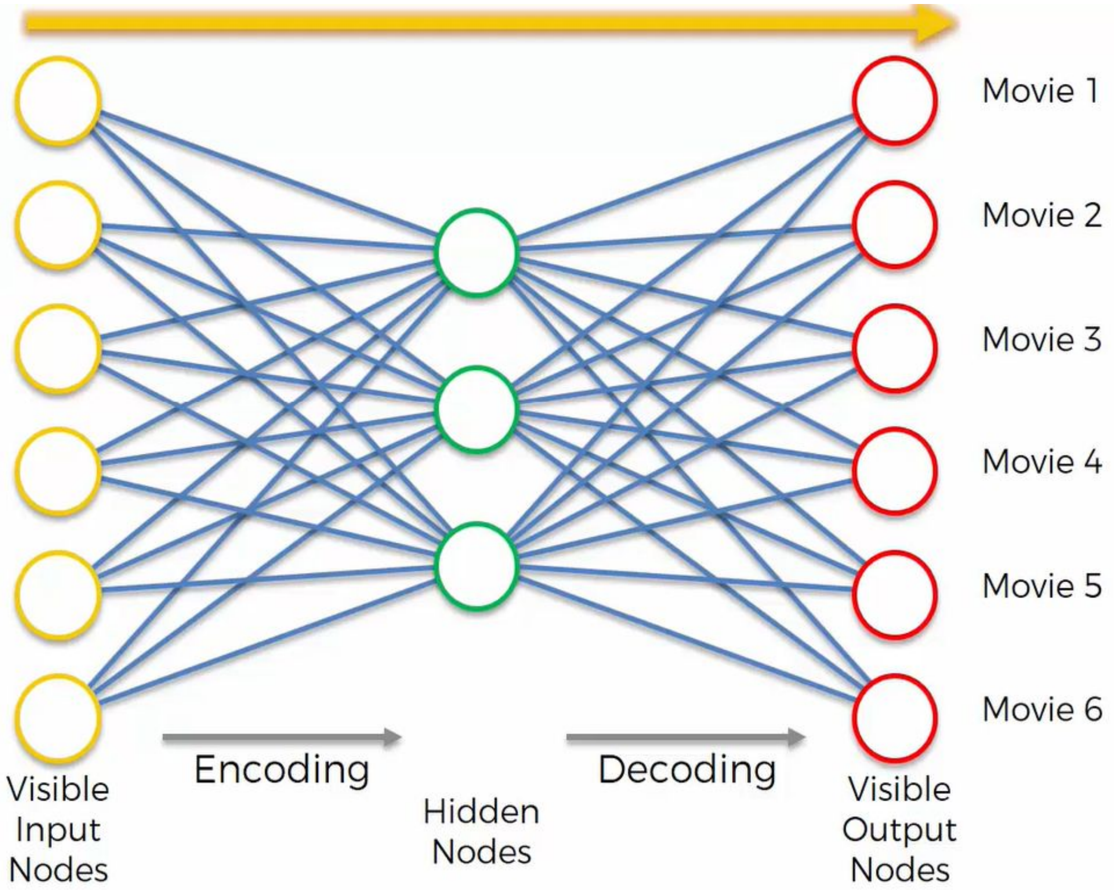
Softmax
= * (+1)
= * (-1)
Movie 1
Movie 2
Movie 3
Movie 4
1
0
0
0
1
1
2
0
0
-2
1
0
0
0
Movie 1
Movie 2
Movie 3
Movie 4
Visible Input Nodes
Hidden Nodes
Visible Output Nodes


= * (+1)
= * (-1)
Movie 1
Movie 2
Movie 3
Movie 4
0
0
0
1
-1
-1
-2
0
0
2
0
0
0
1
Movie 1
Movie 2
Movie 3
Movie 4
Visible Input Nodes
Hidden Nodes
Visible Output Nodes

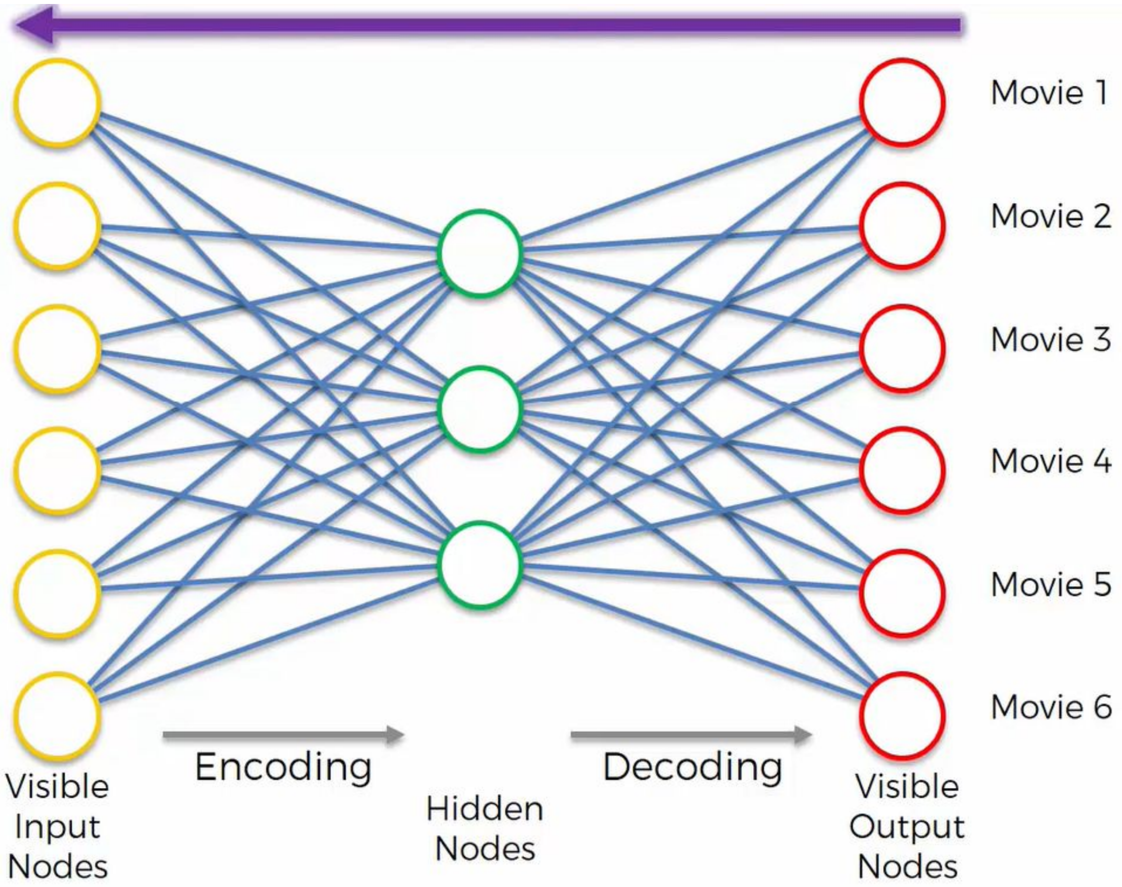
# Why Stacked Autoencoders ?

| | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 | Movie 6 |
|---|---|---|---|---|---|---|
| User 1 | 1 | 0 | | 1 | 1 | 1 |
| User 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| User 3 | | 1 | 1 | 0 | 0 | |
| User 4 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 5 | 0 | | 1 | 1 | | 1 |
| User 6 | 0 | 0 | 0 | 0 | 1 | |
| User 7 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 8 | 0 | 1 | 1 | | 0 | 1 |
| User 9 | | 0 | 1 | 1 | 1 | 1 |
| User 10 | 1 | | 0 | 0 | | 0 |
| User 11 | 0 | 1 | 1 | 1 | 0 | 1 |

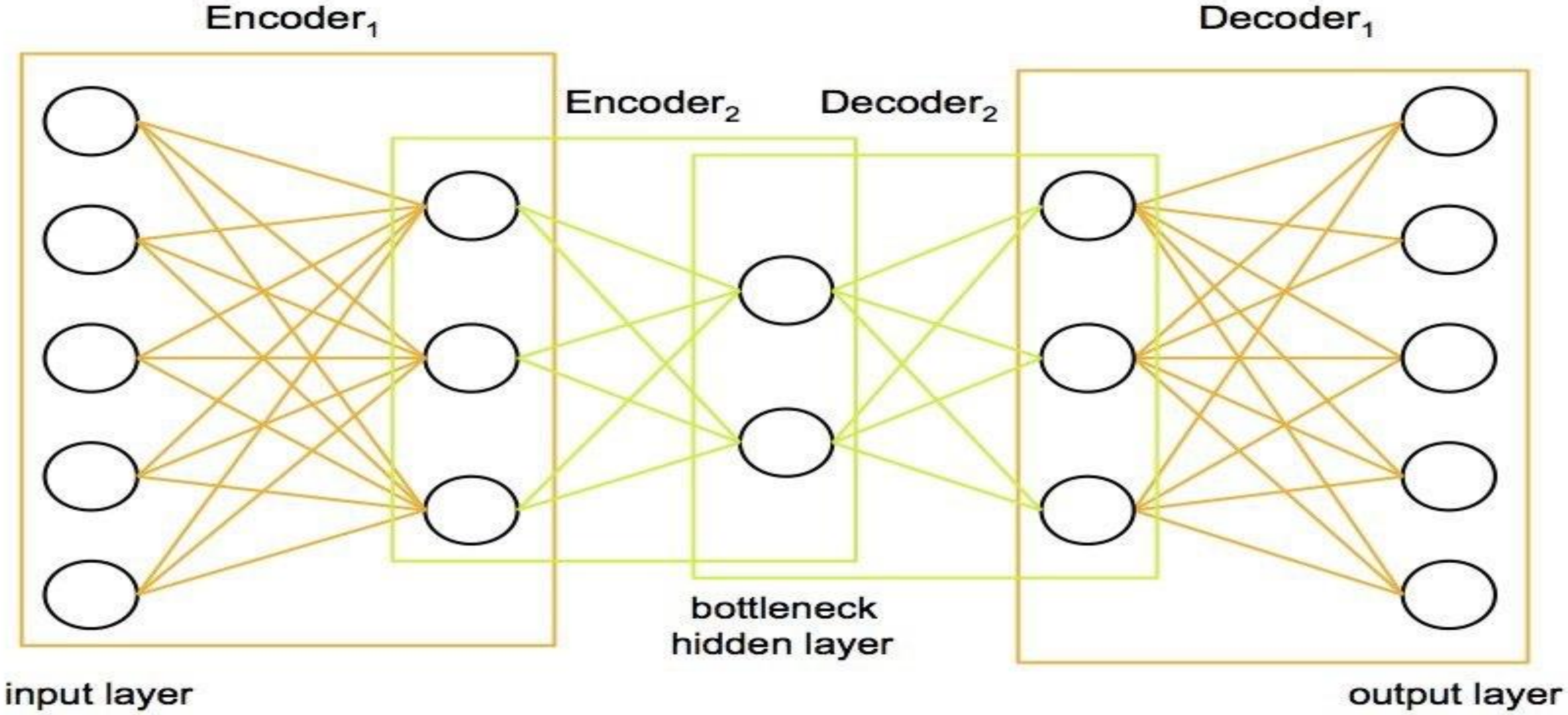
| | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 | Movie 6 |
|---|---|---|---|---|---|---|
| User 1 | 1 | 0 | | 1 | 1 | 1 |
| User 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| User 3 | | 1 | 1 | 0 | 0 | |
| User 4 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 5 | 0 | | 1 | 1 | | 1 |
| User 6 | 0 | 0 | 0 | 0 | 1 | |
| User 7 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 8 | 0 | 1 | 1 | | 0 | 1 |
| User 9 | | 0 | 1 | 1 | 1 | 1 |
| User 10 | 1 | | 0 | 0 | | 0 |
| User 11 | 0 | 1 | 1 | 1 | 0 | 1 |



| | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 | Movie 6 |
|---|---|---|---|---|---|---|
| User 1 | 1 | 0 | | 1 | 1 | 1 |
| User 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| User 3 | | 1 | 1 | 0 | 0 | |
| User 4 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 5 | 0 | | 1 | 1 | | 1 |
| User 6 | 0 | 0 | 0 | 0 | 1 | |
| User 7 | 1 | 0 | 1 | 1 | 0 | 1 |
| User 8 | 0 | 1 | 1 | | 0 | 1 |
| User 9 | | 0 | 1 | 1 | 1 | 1 |
| User 10 | 1 | | 0 | 0 | | 0 |
| User 11 | 0 | 1 | 1 | 1 | 0 | 1 |



# Architecture of Stacked Autoencoders



# Challenges

- Rapid training of the neural network
- Choosing the optimal parameters for the neural networks
- Making predictions with the least possible test loss

# Solutions

- Using the RMSprop Optimizer
- Developing the neural network using the PyTorch Framework
- Selecting the learning rate to be 0.01 and number of training epochs to be 200

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db$$

$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{db}$$

Gradient descent with momenttum

Source: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2$$

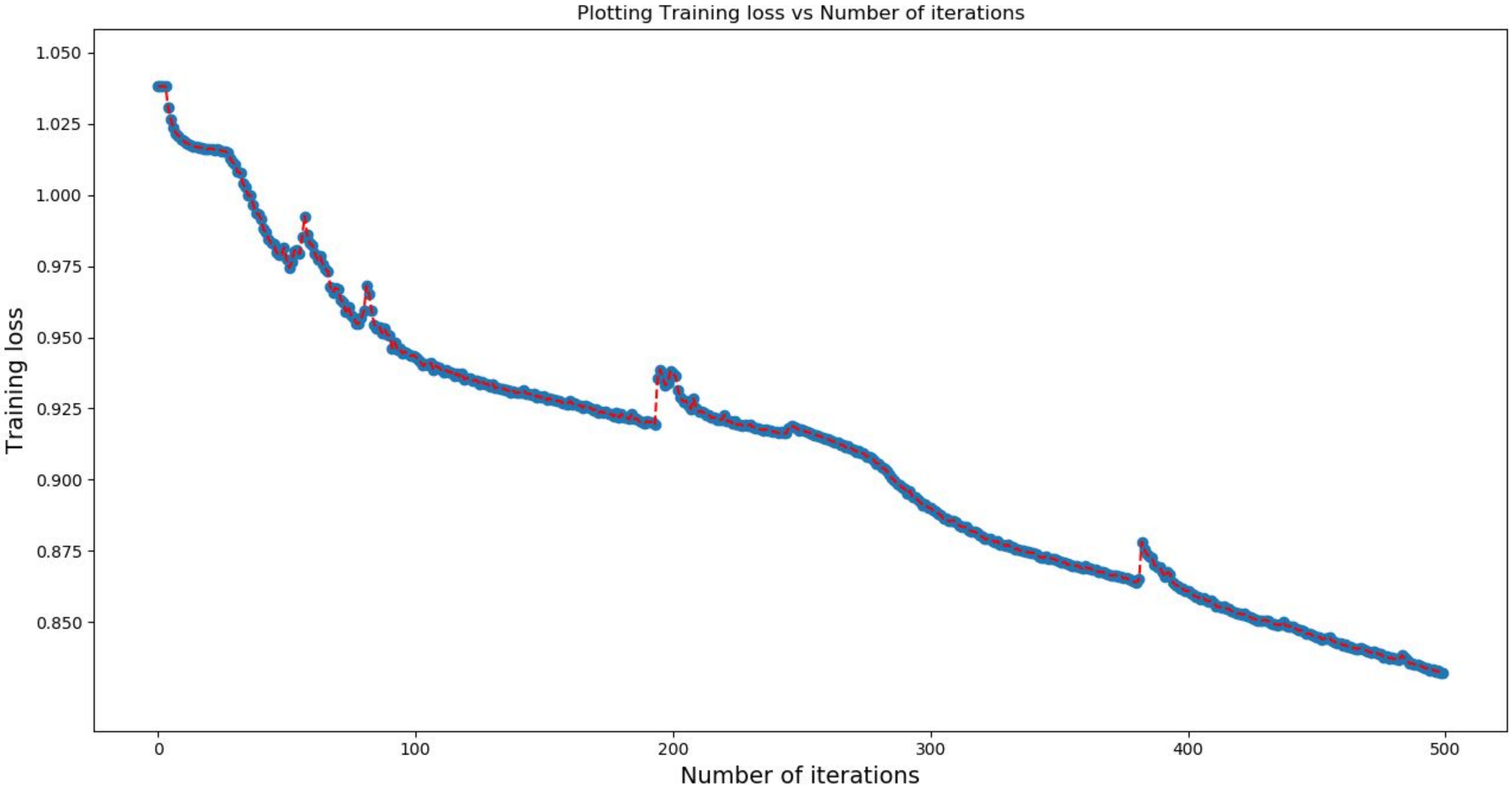
$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

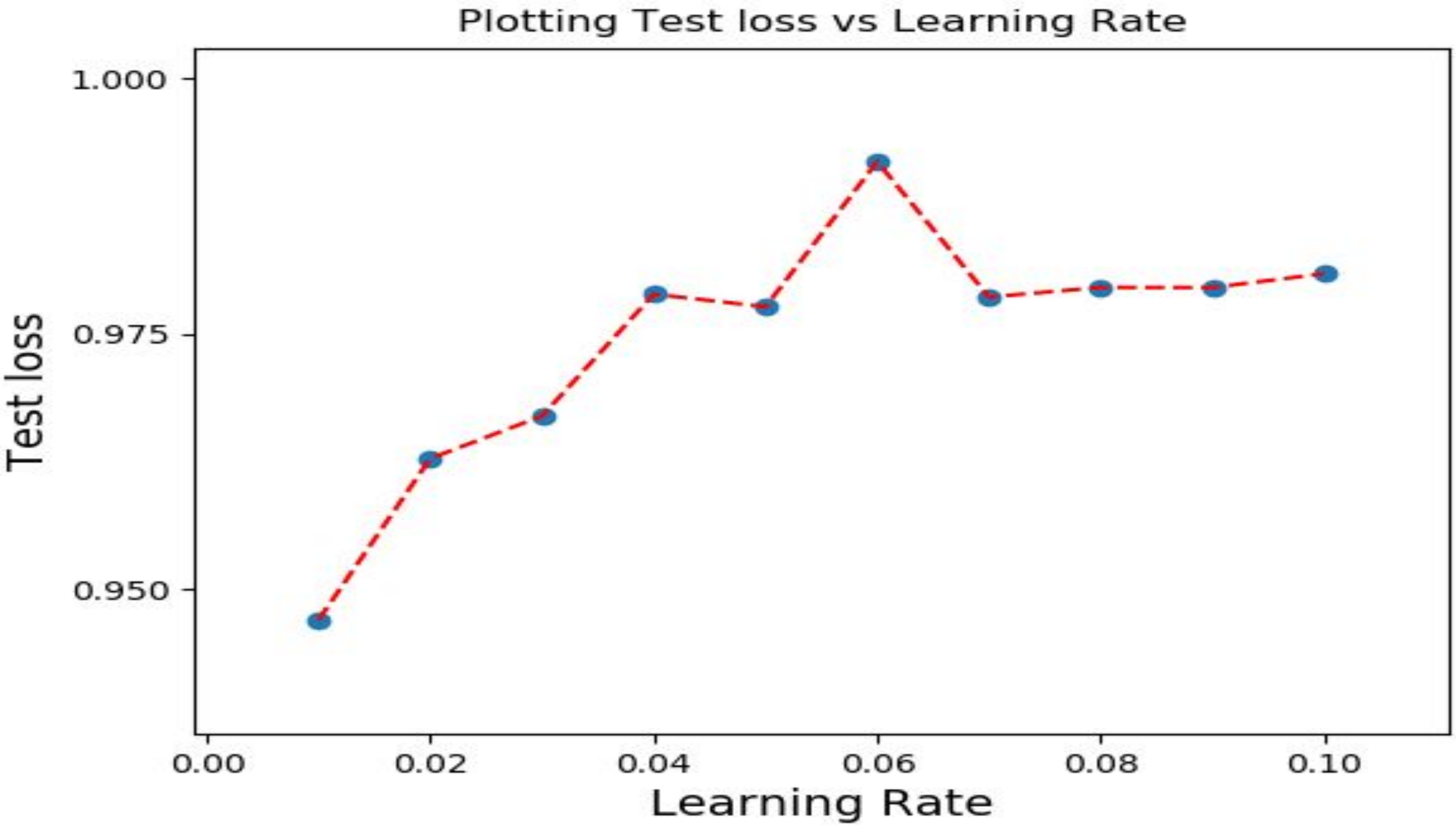
$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

RMSprop optimizer

Source: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b



Plotting Training loss vs Number of iterations
Training loss
Number of iterations
1.050
1.025
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0
100
200
300
400
500


Plotting Test loss vs Learning Rate
1.000
0.975
0.950
Test loss
0.00
0.02
0.04
0.06
0.08
0.10
Learning Rate

# Conclusion

So the model that I have developed is able to predict the movie ratings for users, where the predicted rating would be different from the actual rating only by one star.

# Future Improvements

- Movie lens also has a dataset with 1 million rating
- It would be interesting to fine tune my model to be used with that dataset and see how well it performs

# References


- https://probablydance.com/2016/04/30/neural-networks-are-impressively-good-at-compression/
- https://www.superdatascience.com/deep-learning/
- https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b