Ira A. Fulton Schools of Engineering
Arizona State University

School of Computing, Informatics, and Decision Systems Engineering

P.O. Box 878809
Tempe, AZ 85287-8809

(480) 965-3190
Fax: (480) 965-2751
cidse.engineering.asu.edu

Portfolio Summary

Gowtham Sekkilar

Arizona State University

gsekkila@asu.edu

AMAZON REVIEW VISUALISATION:

Each of the products in Amazon have more than thousands of reviews. It is very difficult to analyze and go through all the reviews and rating. One solution is to create a webpage that can provide all insights about the product right from ratings to helpfulness of reviews. A dashboard is created for Amazon product review analysis. The project simplifies the work for users by providing them with an in-depth analysis of product reviews through visualizations. The interactive visualizations enhance the user experience. The dashboard consists of three parts: Rating Analysis, Sentiment Analysis and Product Recommendation. The reviews were visualized as line and bar chart, gauge chart and word cloud. JavaScript and D3.js was used for interactive visualizations. Python programming language was used in sentiment analysis and helpfulness determination. The dashboard was hosted using Amazon Web Service. The web page has been hosted as a dashboard where the product is searched at the top and then the data about the product such as rating range and ratings across years are visualized as line chart. The ratings for a given year is shown as a bar chart. The positive and negative score are visualizations of the positive and negative reviews. Word cloud visualization shows the most frequently used words in the reviews of the product. Product recommendation is based on the product's categories and is visualized as a bubble chart. There is a helpfulness gauge that represents the review helpfulness percentage of the reviews. The dashboard can help people give an insight to all the reviews and the analysis can help the users make better decisions.

RECOMMENDER SYSTEM- MOVIELENS DATA SET:

The project is about Recommender Systems. They predict users' interest and then recommend items that will be interesting for the user. Recommendation speeds up searches, saves a lot of time and makes it easier for users to watch content that they would never have searched for. This project aims to predict the liking of a new user and suggest movies or TV shows that the user would prefer watching. A detailed discussion about the two types of recommender systems, content-based and collaborative filtering is also done. Collaborative filtering makes recommendations based on the user's preference for items. While content-based systems make recommendations using the users profile features. The dataset that is used is obtained from MovieLens. Preprocessing like handling sparsity of the matrix, filing missing values, removing duplicates and splitting the dataset were all done on the dataset. A Neural Network model is developed for predicting recommendations. It has two layers user embedding layer and item embedding layer. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) is used for evaluating the results. Lower the RMSE value, better the model will be. The performance of the neural network model has been shown in graph plots. The results obtained infers that neural networks give the best movie recommendations to the user.

EVALUATION OF GRADIENT DESCENT OPTIMIZATION TECHNIQUES IN DEEP NEURAL NETWORKS:

The project goal is to examine and evaluate the performance of different gradient descent optimization techniques to train a Deep Neural Network. A 3-layered fully connected neural network was used to classify the fashion MNIST dataset. There are ten categories in the dataset which correspond to different categories of clothing and accessories. The model was trained with 6000 training samples and then it was tested with 1000 testing samples. The neural network has 3 layers and uses two different activation functions. The optimization techniques that were studied and analyzed are Polyak's Classical momentum, RMSprop and Adam Optimization. The techniques were compared in terms of their stability, speeds and accuracies. Evaluation was done by finding the average performance of each technique for different epochs, batch sizes and learning rate. Learning rate is an important parameter. If it is very low, then the convergence occurs very slowly and if it is high it can cause the loss function to fluctuate. Increasing the size of the batches up to a threshold helps in minimizing the fluctuations in the graphs. The accuracies were tabulated for every combination. The results were shown as graph plots and tabulated for better comparison of the techniques. After comparing the different optimization techniques, RMSprop and ADAM were found to have the best performance and accuracies at lower learning rates.

Amazon Review Visualisation

Gowtham Sekkilar
School of Computing, Informatics, and Decision Systems Engineering
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe, Arizona, USA - 85281
gsekkila@asu.edu

***Abstract*— Electronic commerce draws on technologies such as internet marketing, mobile commerce, electronic payments, supply chain management, inventory management systems and data collection systems. in short, electronic commerce has helped people review a lot of products over the internet and get them delivered to their homes with just one click. however with such varied options and choices, the kids often find it difficult in choosing a product. The project is to simplify the work for kids such that the kids are presented with the data from the product reviews, thereby providing the user with an in-depth analysis of the products through various visualisations which enhances the experience through interactivity in the visuals.**

I. INTRODUCTION

Amazon has pioneered in the electronic commerce space and has been the most sought after online marketplace thereby providing a platform for millions of products right from the household needs till office supplies. Each of the products have thousands of reviews which has been bought by so many people. However, there exists a constraint on them as there might be huge amounts of them and in unorganised fashion as well. This might worry the customer a bit as they might feel them to be profusely long and could end up skimming the content. The whole purpose of writing a review gets lost in such occasions. Apart from the huge set of existing review data, there is much more being added at an exponential rate. One such category has been the children's section which has toys and games for teen kids. They have so many products which makes it difficult for analysing and going through all the reviews and ratings. Solution is to create a web-page which can provide all the insights about the product right from ratings to the helpfulness of reviews. The data has been processed to present an insight of all the reviews for the particular product. The web-page has been hosted in a dashboard format where the product to be search is on the top and then the data about the product such as the ratings over the time, and the ratings range for a given year, The visualisation of word based sentiment which is the positive and negative score percentages of the textual analysis, Word cloud visualisation which the most used words in the review context for the particular product, the product recommendation which is of similar categories as the given product, and then the analysis of how the reviews have been help-full and true to the product.

II. DATA SET

The Amazon product was taken from SNAP for the project, the data ranging from 1996 till 2014. This data was processed for the visualisation for the respective visualisations. The data has the following information:

- reviewerID - ID of the reviewer
- asin - ID of the product assigned by Amazon
- reviewerName - name of the reviewer
- helpfulness - helpfulness rating of the review, e.g. 2/3
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- reviewTime - time of the review (raw)

Some more data has been added to the main data set which was taken from the meta data.

- title - name of the product
- price - price in US dollars (at time of crawl)
- related - related products (also bought, also viewed, bought together, bought after viewing)
- categories - list of categories the product belongs to
- Sales Rank - The selling rank of the particular product in the respective category

"Toys" data-set has been selected for counting more than 2M with almost 500+ products and found a lot of data are skewed. Hence we try to filter out the top 100 products based on the reviews. Using this as the base data, analytical dashboard is built that give users an idea about the review overview, sentiment, quality, distribution, most frequent patterns and also category (content) based recommendation with Product ID as input.

III. VISUALIZATIONS

The principles and ideologies taught during the CSE 578 course are implemented diligently which renders a visualization which is pleasing to the users eyes, still gives an informative, interactive, intuitive one which users the proper color code which blends with the data presented. The technologies used for the visualizations are : D3.js, Charts.js, Highcharts, NLTK parser, HTML5, CSS, JavaScript and Bootstrap has been used for the front-end visualization. The data for the visualizations has been extracted using JSON and text files.

1) Rating chart: The above visualisation is the display of rating of the given product over the TimeFrame the product has been on sale. The visualisation gives an idea of how the product has performed over the years, like a dip in the ratings for a particular year means that the there could have been a competition in the market or the quality would have come down. A rise in the ratings could mean that the quality has improved, more people started liking the product. On hovering over a year, we get to see the ratings for that particular year, and the full data is visualised in the form of the bar chart on clicking the point.

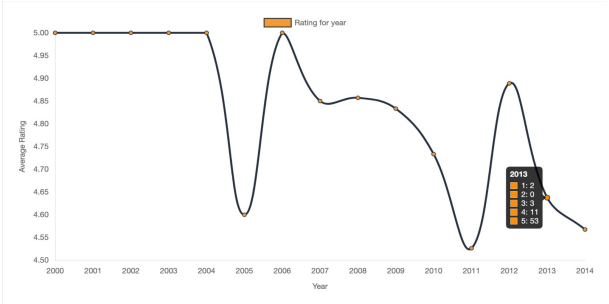

Fig. 1. Ratings over the year

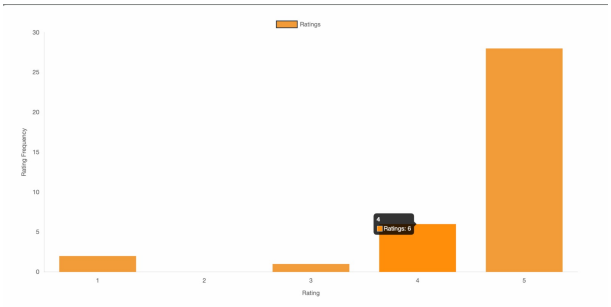

Fig. 2. Ratings for the given year

2) Word Based Sentiment: The visualisation is how the ratings have been for the product, the percentage indicated by the green gauge represents the percentage of positive reviews and the red gauge represents the percentage of the negative reviews

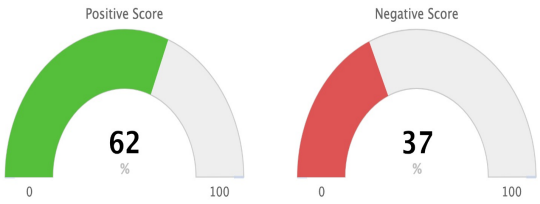

Fig. 3. Sentimental Gauge

3) Frequency Based Word Cloud: The word cloud represents the most frequently used words in the reviews. This is processed through the use of NLTK parser. The distance between the words also represent a meaning of how the frequency of the words vary. The color variation has been used to represent the positive and negative words.

Fig. 4. WordCloud

4) Related Product Categories: The chart is used to represent the related products for each of the categories the given product falls into. This helps the user to suggest products which are similar to the ones which they are looking for. The bigger circles are the representation of categories and the smaller circles which are represented by the same color code are the products belonging to those particular categories.

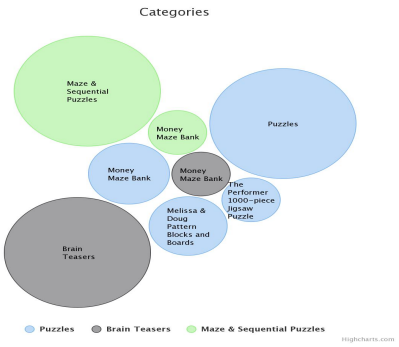

Fig. 5. Category-Product chart

5) Helpfulness Gauge: The chart represents the review helpfulness percentage of the reviews belonging to the product. It is calculated using the formula (no: of helpful reviews) / (no: of non-helpful reviews). This helps user analyze the genuine nature of the review. This is particularly useful in the point of perspective that the total people have found the product to be worthy or not.

IV. COLOR SCHEME

The Color code used for the project is Amazon color codes:

- Blue - RGB (37, 47,62)
- Yellow - RGB (242, 156, 56)
- Grey - RGB (234, 237, 237)

V. TECHNOLOGIES USED

- HTML5
- JavaScript
- CSS
- D3.js
- Charts.js
- HighCharts
- NLTK
- JSON

VI. MY CONTRIBUTION TO THE PROJECT

A. Ranking Analysis

1) Average rating vs year: line chart: A line chart is used since they are best used to display quantitative values over a continuous interval or time period. I and my teammate worked on creating the Line charts. We've made use of chart.js - a simple, clean and engaging HTML5 based JavaScript charts. The data is imported as a JSON file. The user enters the product ID(asin) and this is used as the primary filter. The ratings of that particular product are filtered from the dataset. Multiple ratings are available for a particular product for a particular year. So, the weighted average of the ratings is calculated and used to plot the points in the line chart. We have also added an additional usage which is when you hover over a particular year, the rating distribution for that year is shown. The color schemes have been chosen so that the readings are easily seen to the customer and are also relevant to the Amazon products.

2) Rating Distribution for a year: A rating distribution is displayed when you hover over a particular year. I included another chart to provide better visualization of data. A bar chart is used to show the rating of that particular year vs the rating frequency. The bar chart pops up when a click is made to a year. Chart.js is also used here for the chart implementation. The same color scheme is used in the bar chart.

3) Categories Bubble chart: Each product belongs to different categories. The product id is used to fetch the categories to which that product belongs. I've worked on implementing a bubble chart. The bubble charts provide two pieces of information: the categories and the recommended products for those categories. The size of the category bubble depends on the number of products that belong to that category. On hovering on a category bubble, the recommended products get highlighted.

4) Helpfulness gauge.: Our dataset also contains information about how helpful the reviews are. I have designed a solid gauge that shows the helpfulness for each product. This is calculated by taking all the helpfulness value for all the reviews for the product. I have made use of highchart - an Interactive JavaScript charts for web pages.

B. JSON conversion

We downloaded the entire data set as a CSV file. But we have used the data as a JSON file. We have done some data cleaning in the CSV file. I have written a python script file to convert the CSV file to JSON file. The converted JSON file is then used in our web pages to get the data.

VII. LEARNING FROM THE PROJECT

The project has been a great source of knowledge for technologies for me. I learned a lot about the new technologies that has been established over the years. Understood the uses and importance of data analysis, uses of different charts, using alternatives for d3 such as chart.js highcharts, formatting data in Microsoft Excel, and writing a python file for converting CSV to JSON. This has helped me in a thorough understanding of the creation of interactive charts and the data processing for the charts - creation, processing and visualization. I also learnt about the creation of dynamic websites, website hosting, changing DNS. This has given me an in-depth knowledge of the following:

- Data Analysis
- Data Processing
- Data Creation
- Data Visualization

This will help me in further projects where-in the Data Visualization concepts will be incorporated into the design and visual scenarios. The elements for a good visualization being : Clear headings and Titles, Relevant Comparisons, Data and Evidence, Summaries, Design Elements, Consolidated information. I also learnt a lot of soft-skills such as team work and communication. Working in a team with highly qualified, industry experienced people had its rewards for me. I had help at every step from my team mates and I learnt what team work was all about.

VIII. TEAM MEMBERS

- Aditya Kanchivakam Ananth
- Deepan Karthik
- Srivathsan Bhaskaran
- Motthilal Baskaran
- Gowtham Sekkilar

IX. ACKNOWLEDGMENT

I would like to thank Professor Doctor Sharon Hsiao for having taught me this course, and also the Teaching Assistant Yancy Vance Paredes for having guided me throughout the project. This project would not have been successful without their constant guidance. I would also like to thank my team members for having put in their contributions and also helped me with my parts which resulted in this project being successful.

X. CONCLUSIONS

The dashboard created has helped resolve the problems faced by children and parents while ordering toys for children. This has helped people give an insight to all the reviews and the analysis from the reviews has helped people make better decisions. It does indeed, add an additional perspective to the user by enabling them to easily grasp what the visualization tries to convey about the product review. The idea of static and interactive use of visualization from the class assignments was also incorporated in the project that entitles pattern predictions and the attributes that influence it. This can be further expanded to all the categories as the dashboard is based on the product reviews. Improvise better and more efficient Machine Learning algorithms to expand the product recommendation, Natural Language Processing to improve the genuity of the reviews thereby improving the user experience. Thus the Dashboard provides a cluster-free experience for the users

Recommender System: MovieLens Data Set

Gowtham Sekkilar
School of Computing, Informatics, and Decision Systems Engineering
Ira A. Fulton Schools of Engineering
Arizona State University

***Abstract*— Recommender systems aim to predict users interest and recommend items that are interesting for them. They personalize your experience on the web. They are one of the most powerful machine learning systems that are used in order to improve the user experience. People like things that are similar to other things they like, and they tend to have similar taste as other people they are close with. Recommender systems try to capture these patterns to help predict what other things you might like. This paper aim's to predict the liking of a new user and suggest movies or TV shows that the user would prefer watching. This is done by making use of prior knowledge, which is a list of users along with the ratings given by them, for a list of movies saved in the form of a table as our data set.**

I. INTRODUCTION

Recommendations typically speeds up searches and makes it easier for users to access the content they are interested in and surprise them with interests that they would never have searched for. There are two types of recommender systems.They are Content-Based and Collaborative Filtering (CF). Collaborative filtering makes recommendations based on the users' preference for items. They are based on user-item interactions. In contrast, content-based recommender systems focuses on the attributes of the items and gives you recommendations based on the similarity between them. Content-based systems make recommendations using a user's item and profile features. They hypothesize that if a user was interested in an item in the past, they will once again be interested in a similar item in the future[1]. Similar items are grouped based on their common features. User profiles are constructed using the users past interactions or by asking users about their interests. Collaborative filtering makes use of user interactions to filter items of interest. We can visualize the set of interactions with a matrix, where each entry of the matrix represents the interaction between user i and item j. One way of looking at collaborative filtering is to think of it as a generalization of classification and regression.

II. RELATED WORKS

Recommendation systems are one of the most successful and widespread application of machine learning technologies in business. The most used recommendation methods are Content-based filtering and Collaborative filtering. PANDORA - a music streaming service is a service that uses 400 attributes each for a song and when a user selects a music station, all the songs with the respective attributes will be played. Users or items have profiles describing their characteristics and the system would recommend an item to a user if the two profiles match. Content-based methods are computationally fast and interpretable. They can be easily adapted to new items or new users. However, some characteristics of items or users may not be easy to capture or describe explicitly.

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations using information such as rating profiles of different users or items. The user-based and item-based nearest neighbor algorithms can be combined to deal with the cold start problem and can also improve recommendation results. Collaborative filtering methods are classified as memory-based and model-based. An example of memory-based approaches is the user-based algorithm, while that of model-based approaches is the Kernel-Mapping Recommender. The major drawback of the collaborative filtering method is the cold start problem - These systems often require a large amount of existing data on a user in order to make accurate recommendations.

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices[2]. The idea behind matrix factorization is to represent users and items in a lower dimensional latent space . Some of the most used and simpler matrix factorization approaches are listed in the following sections.

III. SYSTEM ARCHITECTURE & ALGORITHMS

A. Collaborative Filtering

The process of filtering for information using collaboration among various data sources is called Collaborative Filtering. It is mainly based on the idea that if two users have similar liking in the past, then it is most likely that they will have similar liking in the future[3]. There are two types of Collaborative Filtering:

1) Memory-based: In this method, the ratings of the users are used to compute the similarity. There are further divided into two types:

USER-BASED: It is based on the assumption that similar users have similar liking. The similarity between a given target user and the rest of the users is measured and the target user is recommended movies that is liked by the users

in his/her group. The algorithm consists of the following steps:

- Finding K-nearest neighbors to the target user by making use of a similarity function to measure the distance between each pair of users.
- Filtering the movies that have not been rated by the target user but have been rated by similar users.
- Finding the best rated movie among these movies.

A user-item matrix is being created and the ratings of movies that the target user has not yet seen is being predicted from the matrix with the help of the ratings that the similar users have given for these movies[4]. Advantages of User-based Collaborative Filtering are:

- Easy to implement.
- Independent of the context.
- Highly accurate compared to content-based.

Disadvantages of User-based Collaborative Filtering are:

- Since the number of people who rate movies are low, it is difficult to group similar people with less volume of data.
- Since the prediction's accuracy mainly depends on how many people are grouped together, for higher accuracy, cost of grouping also increases. So we face a trade-off between accuracy and cost.
- Cold start: There will be absolutely no information about a new user that has just started using the service. So to predict the ratings and to recommend a movie to a target user that is a new user is almost impossible.
- Similar to the previous problem of a new user, consider the case where a new movie is introduced to the system. None of the users would have rated this movie because none of the users would have seen this movie. So it is again impossible to determine if this movie is to be recommended to the target user or not.

ITEM-BASED : In this method, the similarity between items is measured and using this, movies with similar ratings are grouped together and recommendations are made based on the grouping. The algorithm consists of two steps:

- Calculate the similarity between items
  - Cosine Similarity : Considers the user ratings as vectors and calculates the angle between the two vectors.
  - Correlation-Based Similarity : Considers the user ratings as vectors and calculates the strength of the linear relationship between the two vectors.
  - Jaccard distance : It compares items from two sets to see to closeness of the items. It is measured as a percentage. Higher the percentage, more similar the items are.
- Prediction
  - Weighted sum
  - Regression

2) Model based techniques: The main drawback of Memory-based Collaborative Filtering is that it makes use of a large amount of in-line memory. This problem escalates quickly because of the increasing count of users that use the system. The performance of the system reduces drastically because it takes much longer to compute the recommendation. This in turn causes slower responses to user requests. All of these above problems are intended to be solved by Model-based Collaborative Filtering. The two most common techniques are:

- PCA - This is a dimensionality reduction based technique where we find the low rank latent factors of the original ratings matrix. This can be done by finding the covariance matrix of the original ratings matrix. The original ratings matrix is not a square matrix, so it is not possible to find the covariance matrix of the ratings matrix. To solve this, we multiply the ratings matrix with its own transpose thereby obtaining a square matrix. This process is computationally expensive as finding the transpose and multiplying the original matrix with the transpose will take a lot of time since the matrix is very large. Furthermore, PCA creates latent factors by projecting the data along the directions of maximum covariance but that does not necessarily mean the features having highest variance are important. Because of these drawbacks of PCA, we implemented a more effective technique known as SVD.

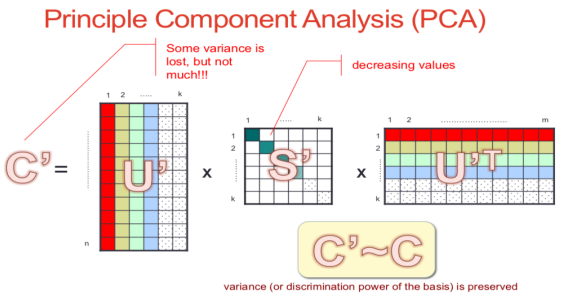

Fig. 1. PCA

- SVD - The most common approach of finding the low rank latent factors is SVD. Each user can be represented by a vector 'u' and each movie can be represented by a vector 'v'. In this method, the ratings matrix is split into three matrices. It can be expressed as: X = U x S x Vt. In this expression, S is the diagonal matrix corresponding to the singular values and U and Vt are orthogonal matrices which corresponds to users and items respectively. The factors corresponding to top k singular values are then used to predict the user ratings.

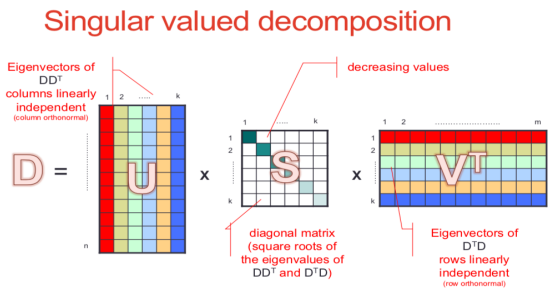

Fig. 2. SVD

3) Neural Network:

- Collaborative Filtering Model: This is the basic model in Neural Network which is similar to the working of Collaborative Filtering through Matrix Factorization. It has two layers: User-embedding layer and Item-embedding layer which are merged using dot product. This merged layer is connected to one single output node.

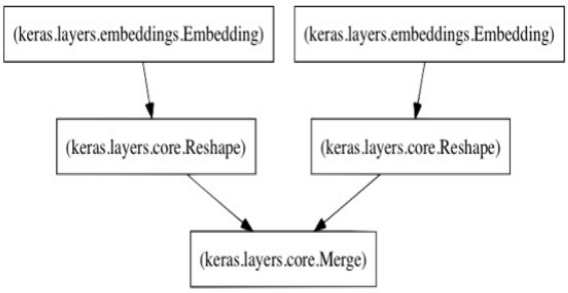

Fig. 3. Simple Neural Network Model

- Deep Network Model: This model takes the idea of Basic Collaborative Filtering Model and improves it using various hidden layers with different activation functions. In the basic model, the dot product of user and item embeddings are taken. In Deep Network Model, we concatenate them and use them as a feature for the neural network. This enhances our model to learn non-linear relationships. It finds the co-relation between the features because of dropping the outputs for each hidden layer. This leads to the model learning from the data after each layer instead of memorizing it.

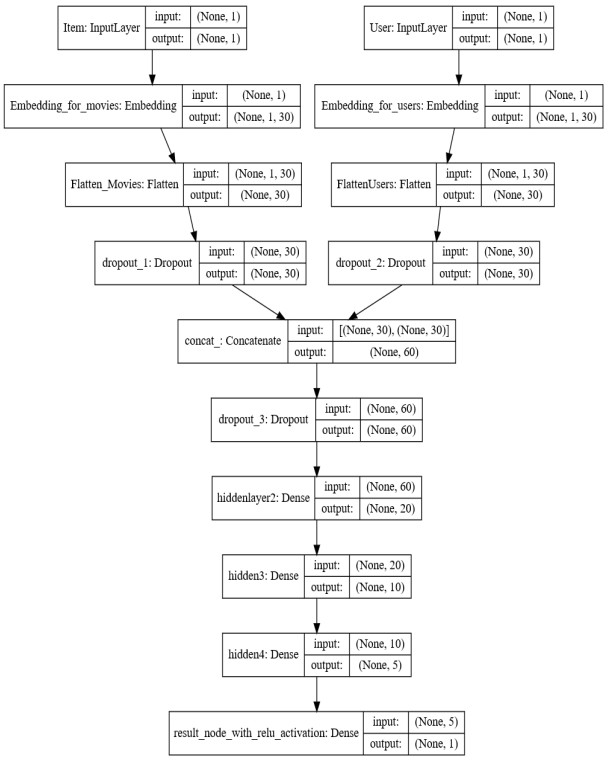

Fig. 4. Neural Network Model

IV. DATA SET

The data set has been obtained from MovieLens [5]. MovieLens is a web based recommender system and an online community that recommends movies for its users to watch. We are using the MovieLens 1M data set having 1 million ratings from 6040 users on 3706 movies. The three main files of our data set are:

- USER-DATA-FILE = "users.dat." This file contains four columns: UserID, Gender, Age, Occupation, Zip-code.
- MOVIE-DATA-FILE = "movies.dat." This file contains three columns: MovieID, Title, Genre.
- RATING-DATA-FILE = "ratings.dat." This file contains four columns: UserID, MovieID, Rating, Timestamp.

The Ratings Histogram based on the data set is as follows:

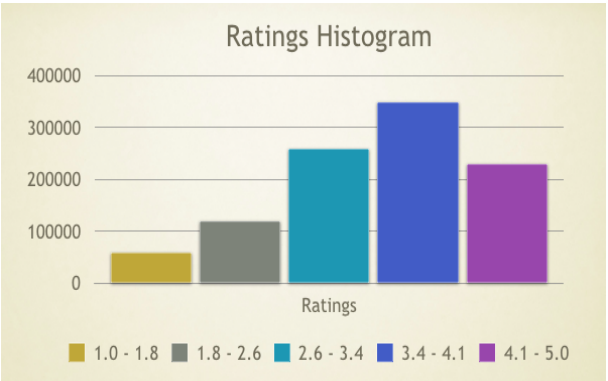

Fig. 5. Histogram of Distribution of Ratings

PRE-PROCESSING:

- SVD
  - We used ratings.dat to create the ratings matrix.
  - In our data set, all the users have only seen and rated a small subset of movies. So the ratings matrix has a lot of missing values. The ratings matrix created was 95.71 sparse.
  - The sparsity was handled by trying out two different approaches:
    * Filling the missing values with 0.
    * Filling the missing values in the matrix by the column average and then creating the new matrix by subtracting this matrix from the original matrix.
  - The ratings matrix was split into 9:1 ratio into train and test set.
- Neural Networks
  - We separated the information of each user which is separated by a "::" in RATING-DATA-FILE. We saved it into a new csv file: ratings.csv which consisted of the columns: userid, movieid, rating, timestamp, user-emb-id and movie-emb-id.
  - We converted data for each user in the format: userid, gender, age, occupation, zipcode from the given USER-DATA-FILE.
  - We converted data for each movie in the format: movieid, title, genre.
  - We removed the duplicates and incomplete records.
  - We shuffled the records in the ratings.csv file. We did this for training purpose and we found the highest user-id and movie-id which is required for embeddings.
  - The data set was then split into train set and test set in a ratio of 9:1.

V. EVALUATION

The metrics which was used for evaluating the results of the different recommender models are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). MAE is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. RMSE is the square root of the average of squared differences between prediction and actual observation. The lesser the RMSE value, the better the model will be with the best RMSE being 0.

Among Item-Based collaborative filtering method and User-Based Collaborative filtering method, User-Based collaborative performed better with a RMSE score of 3.11. The RMSE value for SVD was 0.8735 which is better than both Item-Based and User-Based collaborative filtering methods. The Neural Network Model gave the Best RMSE value of 0.7993.

SINGULAR VALUE DECOMPOSITION

For SVD, we used different values for the number of latent features(k) and the best result was for k = 60. This is because

Model	RMSE
User-Based Collaborative Filtering	3.11
Item-Based Collaborative Filtering	3.20
SVD	0.87
Neural Network	0.7993

with more latent features we include irrelevant features which results in poor performance and thus high RMSE.

K	RMSE
50	0.8615
60	0.860
70	0.8632
80	0.8647
100	0.8658
100	0.8664

We compared the top 10 recommendations for Neural network and SVD based on the ratings matrix for the user 1 :

MOVIE ID	Genre
364	Animation, Children's Musical
318	Drama
2858	Comedy, Drama
110	Action, Drama, War
1198	Action, Adventure
1036	Action, Thriller
2324	Comedy, Drama
480	Action, Adventure, Sci-Fi
1225	Drama
2396	Comedy, Romance

Fig. 6. Recommendation - SVD

NEURAL NETWORK

The Training and Validation loss for the Neural Network model is as follows:

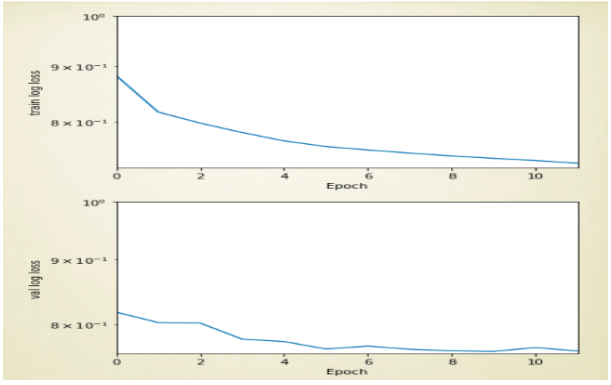

Fig. 7. Neural Network Log Loss

The predicted ratings for the movies which user hasn't watched are then sorted and then the top 10 ratings are integrated with movie data to generate recommendations:

	movieid	prediction	title	genre
0	1197	4.454967	Princess Bride, The (1987)	Action|Adventure|Comedy|Romance
1	1035	4.301311	Sound of Music, The (1965)	Musical
2	2018	4.298745	Bambi (1942)	Animation|Children's
3	588	4.217417	Aladdin (1992)	Animation|Children's|Comedy|Musical
4	919	4.076934	Wizard of Oz, The (1939)	Adventure|Children's|Drama|Musical
5	1207	4.062168	To Kill a Mockingbird (1962)	Drama
6	1246	3.976522	Dead Poets Society (1989)	Drama
7	1022	3.915877	Cinderella (1950)	Animation|Children's|Musical
8	594	3.894730	Snow White and the Seven Dwarfs (1937)	Animation|Children's|Musical
9	1270	3.844824	Back to the Future (1985)	Comedy|Sci-Fi

Fig. 8. Recommendations - Deep Neural Network

For the deep neural network, different combinations of activation functions and size of embeddings were tried which are mentioned in Figure 9 with their RMSE values:

Exp_number	Size of Embeddings (User and movie embeddings)	Layer 1 No of nodes,activation function	Layer 2	Layer 3	Layer 4	RMSE
0	60,60	60,relu	30,relu	15,relu	15,relu	0.8738
1	60,60	60,sigmoid	30,sigmoid	15,sigmoid	15,sigmoid	0.8766
2	60,60	60,tanh	30,tanh	15,tanh	15,tanh	0.8710
3	60,60	60,tanh	30,sigmoid	15,tanh	15,sigmoid	0.8748
4	30,30	30,relu	20,relu	10,relu	5,relu	**0.7993**
5	30,30	30,tanh	20,tanh	10,tanh	5,tanh	0.8704
6	30,30	30,tanh	20,sigmoid	10,tanh	5,sigmoid	3.7530
7	20,20	20,relu	10,relu	5,relu	2,relu	0.8720

Fig. 9. Neural Network Results for Each Combinations

- Changing size for user and movies embeddings affects the overall performance of the model(mentioned in figure 10). Performance of deep neural network first improves with reduction in embedding size, but if embedding size is too small, performance decreases.

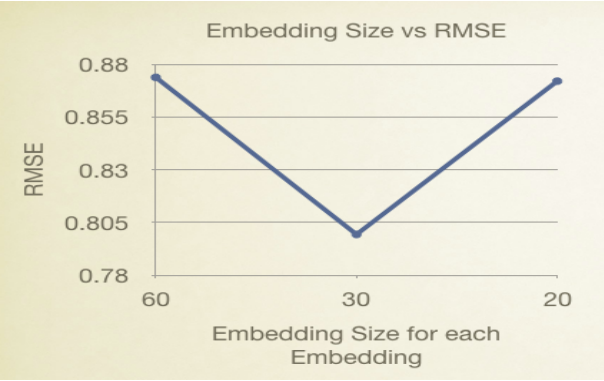

Fig. 10. Embedding size vs RMSE for Neural Network (activation:RELU for all hidden layers)

- Performance of different activation functions is compared in figure 11 and 12 while embedding size and hidden layer sizes are fixed.It shows that model with RELU activation for all hidden layers is best while embedding size is 30.For embedding size =60, best model has tanh activation for all hidden layers.

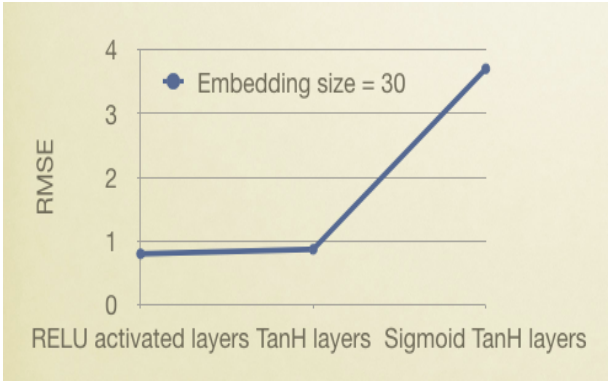

Fig. 11. Different Activation Functions vs RMSE for Neural Network (Embedding size:30)

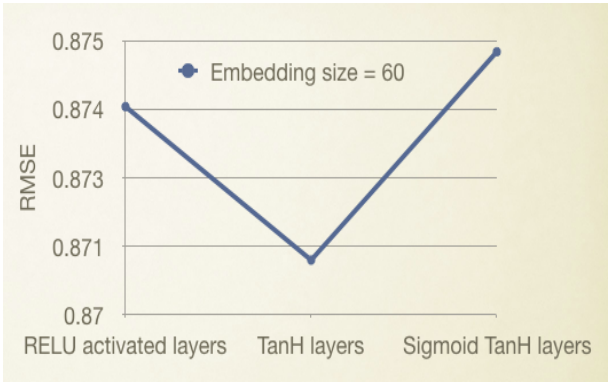

Fig. 12. Different Activation Functions vs RMSE for Neural Network(Embedding size:60)

VI. TEAM MEMBERS

- Akash Narayana
- Ankit Deka
- Kaustubh Milindrao Sardar
- Srivathsan Bhaskaran
- Gowtham Sekkilar

VII. CONTRIBUTION TO THE PROJECT

I worked on preprocessing the dataset, implementing the experiment, evaluation and creating the presentations. I have used the data set obtained from Movielens. Preprocessing the dataset included handling sparsity of the matrix, filling the missing values, making changes in the rating data file and user data file, removing duplicates, and splitting the data set into training set and testing set.I have used Mean Absolute Error (MSE) and Root Mean Squared Error (RMSE) as the metrics for evaluating the results of the different recommender models. The results were tabulated and the top 10 recommendations for the user is shown.

VIII. SKILLS LEARNED

The project has been a great source of knowledge for understanding machine learning concepts. I learned a lot about neural networks and recommender systems. Understood how MSE and RMSE can be used for evaluating the recommender model. Studied about the different collaborative filtering and content based filtering techniques. I also learnt a lot of soft skills such as teamwork and communication. Working in a team with highly qualified, industry experienced people has taught me a lot. I had help at every step from my team members

IX. ACKNOWLEDGEMENT

I would like to thank Professor Doctor Hasan Davulcu for having taught me the course "Semantic Web Mining", and also the Teaching Assistant Amin Salehi for guiding our team throughout the project. This project would not have been successful without their constant guidance.

X. CONCLUSIONS

Neural Network has proven to be the best model with RMSE value: 0.7993. The Singular Value Decomposition has been the second Best Recommendation model with RMSE value of 0.8735. The Item-Based and User-Based have given values of 3.22 and 3.11 respectively. Hence we can conclude that the Neural Network gives the best movie recommendations to the user. It finds the relations between latent features of each embeddings. Also, the dropout after each layer forces neural network to learn from different nodes in previous layers instead of memorizing the data pattern thus avoiding overfitting. This is also one of the reasons as to why neural network performed well on the test data.

REFERENCES

[1] The Age of Recommender Systems, 2018.
[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.
[3] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu, "Dual-regularized one-class collaborative filtering," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 759–768.
[4] Recommender Systems- User-Based and Item-Based Collaborative Filtering, 2017.
[5] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016.

Evaluation of Gradient Descent Optimization Techniques in Deep Neural Networks

Gowtham Sekkilar

School of Computing, Informatics, and Decision Systems Engineering

Ira A. Fulton Schools of Engineering

Arizona State University

Abstract— **The aim of this paper is to evaluate and compare the performance of different gradient descent optimization techniques to train a Deep Neural Network. Different parameters such as stability, speed and accuracy of the prediction were used to compare the performance. The Fashion MNIST dataset [1] was used to benchmark the techniques . A model was trained using 6000 training samples and the trained model was tested using 1000 samples from the test dataset. The different optimization techniques that were studied and analyzed are Polyak's classical momentum [2], RMSprop and Adam Optimization[3].**

Keywords—Gradient Optimization, Deep Neural Network, Polyak's classical momentum, RMSprop , Adam Optimization.

I. INTRODUCTION

The field of Deep Learning has seen a tremendous growth in the research community in the recent years. A neural network's power depends on its ability to learn from data efficiently in a short time and provide an accurate predication. Gradient Descent algorithm is one of the most successful methods that has been developed in order to minimize the cost function.

Gradient descent is an optimization algorithm that is used to minimize a function by iteratively moving in the direction of the steepest descent as defined by the negative of the gradient. Momentum is a method which accelerates the gradient vector in the optimal direction so that the objective function can converge faster.

In this paper, a neural network with 3-layers are implemented. The input layer contains 784 input neurons in order to receive the greyscale level from 784 pixels(28*28). The first layer contains 500 neurons and this is connected to the second layer containing 100 neurons. The third or the output layer has 10 neurons, each corresponding to the ten categories in the fashion MNIST dataset. The activation function that were used for the hidden layers is Rectifier Linear Unit (ReLU) and for the output layer the it is SoftMax function (Normalized activation function)

II. RELATED WORK

The momentum method in gradient descent was proposed by Rumelhart, Hinton and Williams' seminal paper on backpropagation learning [4]. Momentum method accelerates the gradient descend by remembering the update Δw at each iteration and the next update is determined as a linear combination of the gradient and the previous update[5]

$$\Delta w := \alpha \Delta w - \eta \nabla Q_i(w)$$

$$w := \Delta w + w$$

that leads to,

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w$$

where the parameter w, which minimizes $Q_i(w)$ is to be estimated and η is called the learning rate. The main idea behind momentum is to accelerate progress along the dimension which the gradient points to and slowly progress along the dimensions where the sign of the gradient continues to change [6]. A brief description of the different gradient descent optimization techniques that have been used is as follows:

A. Polyak's classical momentum

Polyak has showed that momentum can accelerate convergence to a local minimum, requiring $\sqrt{R}$ times fewer iterations than steepest descent to give the same level of accuracy. Here R is the condition number of the curvature at the minimum and η is set to $(\sqrt{R} - 1)\ (\sqrt{R} + 1)$.

B. RMSprop

RProp combines the idea of using only the sign of the gradient with the idea of adapting the step size separately for each weight. It would increment the weight nine times and decrement it once by about the same amount.

RMSProp is an unpublished adaptive learning rate method. In RMS prop, a moving average of the squared gradient for each weight is used [7].

$$MeanSquare(w,t) = 0.9*MeanSquare(w, t-1)+0.1(\partial E / \partial w^{(t)})^2$$

Dividing the gradient by $\sqrt{MeanSquare(w,t)}$ makes the learning work much better.

C. Adams's optimization

Adaptive Momentum Estimation computes adaptive learning rates for each parameter. Like momentum, this technique keeps an exponentially decaying average of the past gradients m. On an error surface, the adam optimization technique leads to a flat minima.

Let m_t and v_t be the estimates of the mean and uncentered variance of the gradients respectively.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

These estimates are biased towards zero as they are initialized as vectors of 0's. In order to resolve this, the corrected bias estimates are computed.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Updating the above parameters yields the Adam update rule as

$$\Theta_{t+1} = \theta_t - \eta\frac{\eta}{\sqrt{\hat{v_t}+\varepsilon}}\hat{m}_t$$

β_1 is initialized to a default value of 0.9, β_2 to a default value of 0.999 and ε to a default value of 10^{-8} in this optimization technique.

III. METHOD

A. *Fashion MNIST*

The fashion MNIST dataset was used for analyzing the performance of the gradient descent techniques. The dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image and is associated with a label from 10 classes. The ten categories are nothing but different categories of clothing and accessories.

B. *Neural Network Configuration*

A 3-layer neural network was implemented. The first layer is the input layer which contains 784 neurons, and there are two hidden layers, each containing 500 and 100 neurons respectively. The last layer is the output layer which contains 10 neurons matching with the number of categories that are to be classified.

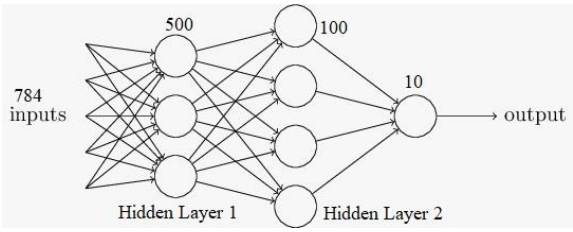

Fig.1 Neural Network Architecture

C. *Activation Function*

The first two hidden layers use rectified linear Units (ReLU) as the activating function. The SoftMax function is used as the activation function for the output layer. This calculates the cross-entropy loss between the predicted and actual labels.

IV. INFERENCES

A. *Choosing Learning Rate*

The learning rate is perhaps the most important hyperparameter that needs to be tuned. If learning rate is lowered, the convergence might happen after a long time. On the other hand, if the learning rate is high, it can hinder convergence and cause the loss function to fluctuate. Different models with different learning rates were evaluated and the results are given in the tables below:

Table 1. Gradient descent with no momentum

Learning rate	Training Accuracy	Testing Accuracy
0.01	84.05	83.60
0.07	88.80	85.20
0.1	**89.18**	**85.30**

Table 2.Gradient descent with Polyak's momentum

Learning rate	Training Accuracy	Testing Accuracy
0.01	95.50	86.00
0.07	87.85	85.00
0.1	**96.83**	**86.0**

Table 3.Gradient descent with RMSprop

Learning rate	Training Accuracy	Testing Accuracy
0.00001	77.50	74.50
0.00007	76.03	75.00
0.0007	**97.65**	**85.50**
0.0001	88.50	85.40

Table 4.Gradient descent with Adam

Learning rate	Training Accuracy	Testing Accuracy
0.00007	86.83	84.90
0.0007	**99.43**	**86.90**
0.0001	88.78	85.70

B. *Analysis of Loss with training cycles*

The system quickly learns to predict the labels with higher accuracy when the number of training cycles is increased. The loss rate decreases rapidly which is represented by a steep curve in the plots given below. Here, plots with different learning rates are compared to identify the parameters giving maximum efficiency.

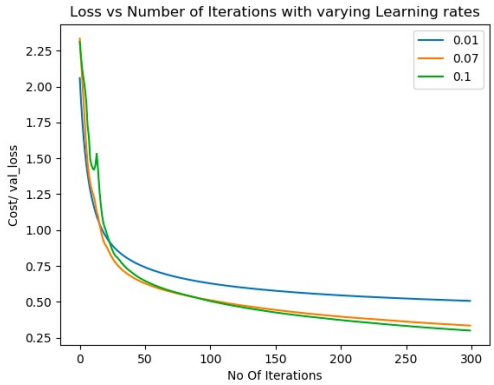

Fig. 2 Plot of Loss with training cycles (No Momentum)

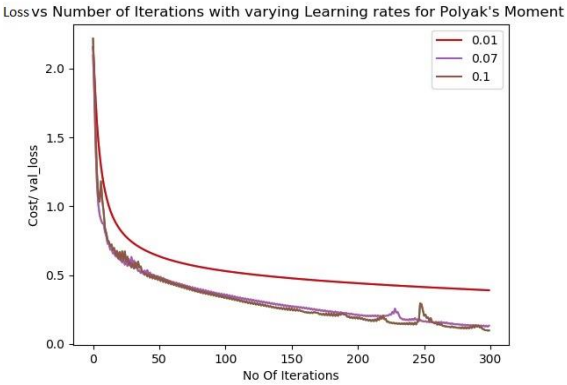

Fig. 3 Plot of Loss with training cycle (Polyak's Momentum)

C. *RMSProp and Adam*

In RMSprop and Adam optimization technique, the problem of radically diminishing gradients is avoided by storing an exponentially decaying average of past squared gradients. Thus they avoid fluctuations and recover quickly from a saddle point. The plots shown in Figure 4 and 5 are smoother with little noise associated with them. Additionally, these two techniques help in finding the optimal direction for descent and converges quickly compared to the classic approach.

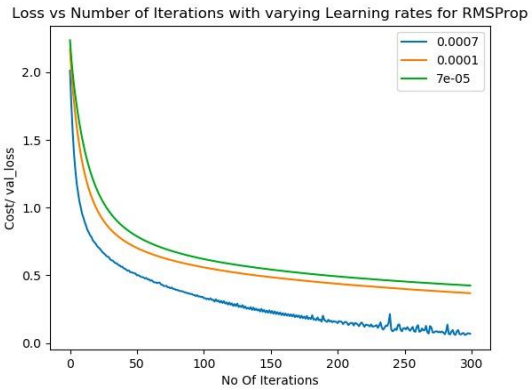

Fig. 4 Plot of Loss with training cycles (RMSprop)

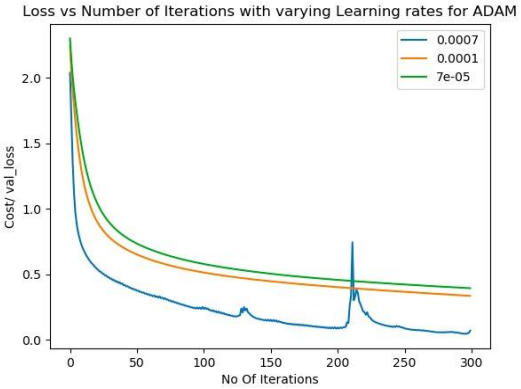

Fig. 5 Plot of Loss with training cycles (Adam)

D. *Batch Size:*

Varying the batch size, changes the amount of fluctuations in the plot. When the batch size is small, the gradients are noisy and this reflects as spikes in the plot. Increasing the sizes of the batches to a certain threshold helps in reducing the fluctuations in the plot. Figures 6 and 7 shows the results of the varying batch sizes between 15 and 20.

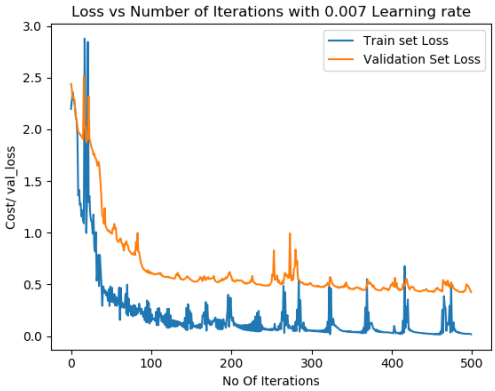

Fig. 6 Adam with batch size 15

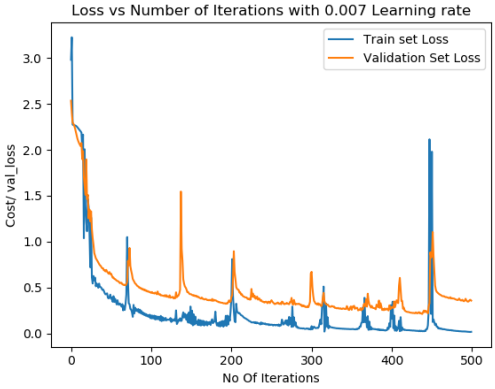

Fig. 7 Adam with batch size 20

E. *Overall Performance*

The performance of gradient descent algorithms was compared by changing different parameters such as number of epochs and batch size. The accuracies for each setting are shown in the tables below. It can be deduced that the accuracy increases as the number of iterations and the batch size increases till a threshold.

Table.5. Gradient Descent with no momentum and LR=0.1

Number of Iterations	Batch size	Training accuracy	Testing accuracy
500	10	92.16	86.10
500	15	95.86	86.20
500	20	97.05	85.50
700	10	93.63	86.00
700	15	97.40	86.60
700	20	98.40	85.80

Table 6. Gradient Descent with Polyak's momentum and LR=0.1

Number of Iterations	Batch size	Training accuracy	Testing accuracy
500	10	97.81	85.20
500	15	98.36	86.40
500	20	86.61	85.40
700	10	99.86	85.60
700	15	99.61	85.70
700	20	99.70	86.60

Table 7. Gradient Descent with RMSprop and LR=0.00007

Number of Iterations	Batch size	Training accuracy	Testing accuracy
500	10	89.45	86.20
500	15	90.45	84.80
500	20	88.43	82.90
700	10	91.21	86.20
700	15	92.03	84.70
700	20	89.30	81.50

Table 8. Gradient Descent with Adam and LR=0.007

Number of Iterations	Batch size	Training accuracy	Testing accuracy
500	10	94.15	84.10
500	15	81.50	75.00
500	20	84.91	77.30
700	10	91.27	82.40
700	15	87.85	79.70
700	20	83.11	74.70

F. Different variants of Gradient techniques

Stochastic Gradient Descent (SGD) updates the model parameters for each training data and provides an immediate insight into the performance & accuracy of the model at the cost of the number of computations. While batch gradient is relatively faster and provides a smooth gradient descent because the updates are much fewer. Mini batch finds a perfect balance between performance and robustness from both SGD and Batch gradient. It does this by splitting the data into several batches and providing better performance with fewer updates.

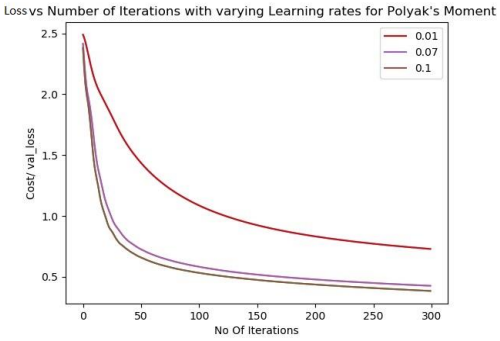

Fig. 8 Batch implementation For Polyak's Momentum

V. Conclusion

Through several executions of training cycles, we evaluated the performances of each technique by varying Batch sizes, learning rates and number of epochs. RMSprop and ADAM showed best performances and accuracies at lower learning rates and the results were robust against fluctuations and noise introduced by the configurations.

VI. Team members

- Raghavendran Ramakrishnan
- Balaji GokulaKrishnan
- Arun Vignesh Malarkkan
- Gowtham Sekkilar

VII. My contribution to the project

I worked on the performance evaluation and analysis of the different optimization techniques. The evaluation of the different techniques was done by comparing the testing accuracies of the techniques by varying mini batch, learning rate and epoch. The different mini batch sizes that were used are 10,15 and 20. The learning rate was varied between 0.00001 to 0.1. The loss rate for 500 and 700 epochs were studied. The accuracies were obtained for each combination of epochs, batch size and learning rate. The results were then tabulated for better comparison.

VIII. Skills learned

The project has been a great source of knowledge for understanding machine learning concepts. I learned a lot about neural networks, gradient descent and optimization algorithms. Understood how changing epochs, learning rate and mini bath size affects the accuracies of the algorithm. I also learnt a lot of soft skills such as teamwork and communication. Working in a team with highly qualified, industry experienced people has taught me a lot. I had help at every step from my team members.

IX. Acknowledgement

I would like to thank Professor Hemanth Venkateswara for having taught me the Fundamentals of Statistical Learning course and for guiding our team throughout the project. This project would not have been successful without his constant guidance.

References

[1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," pp. 1–6, 2017.

[2] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, Jan. 1964.

[3] D. P. Kingma and J. Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION."

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams,

"Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533, Oct. 1986.

[5] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," 2013.

[6] M. D. Zeiler, "ADADELTA: AN ADAPTIVE LEARNING RATE METHOD."

[7] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning, vol. 4, no. 2, pp. 26–31, 2012.