Module 2 - Recommendation systems:

Group: 12
Anton Claesson, 971104-3217, MPDSC, canton@student.chalmers.se, 16~h
Lukas Martinsson : 980203-9678, MPDSC, malukas@student.chalmers.se, ~16h

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions

Reflection about the Netflix Price papers: Anton

Summary of take-aways:

The Netflix price was a competition in which Netflix challenged and offered rewards to researchers and individuals to create a recommendation system that could beat the performance of their existing system called Cinematch. The challenge revolved around a dataset consisting of 100M user ratings from over 480k users on 18k movies- a very large dataset.
The difficulty of the task mainly lies within the fact that data is sparse, i.e. most users have only watched and provided a rating for a small subset of the movies. Most successful approaches therefore utilized some kind of collaborative filtering method which utilizes similarities between users and items simultaneously to provide recommendations.

There were three main areas which were identified as important when it came to improving the accuracy of the user ratings:

- Performance could be improved by improving the quality of the existing models, mostly by finding shortcomings of the current models with regards to the problem at hand.
- Combining the predictions of existing models that complement each other. For example, it was discussed that neighborhood models are good at detecting localized relationships (such as a trilogy of movies) while latent factor models capture features on a more global scale that relate simultaneously to most or all models (such as genres). This is called following a multi-scale approach.
- Finally, incorporation of implicit factors in addition to explicit user rating behavior in order to better model the users. This is helpful due to the sparse ratings of most users. Including information such as which movies a user rate rather than how they rate was something that complemented the user rating information well. In real-life systems one could utilize even more implicit user data such as browsing patterns, purchasing history and much more.

Discussion of design features:

Let’s talk about the two latter points above. Combining different models that are more specialized in local or global features seems to be something that would generally be good for most applications of recommendation systems, although of course the complexity of the model increases which is something that needs to be considered if top of the line performance is not necessary. A parallel could be drawn to the area of deep neural networks

where transformers, which are models that act on both a global and local scale, in most areas now outperform traditional convolutional networks that act on more localized features.

Reflection about the Netflix Price papers: Lukas

The Netflix Prize competition was a machine learning, data, AI competition where the goal was to make a recommender system as accurate as possible. To win the prize of one million dollars one would have to improve Netflix current system Cinematch by 10%. Netflix released a massive dataset containing 100 million ratings from 480 thousand different users on 18 thousand different movie titles. Since no one achieved the improvement state above, they gave out a progress prize instead. The reason why it is so hard to make a recommender system like this is since each user in the dataset only rates a few movies. This makes traditional machine learning models less useful since most data points for each movie is 0 (i.e no rating). This makes it harder to find correlations between data points, which is further amplified since even if there were more data points it would be hard to determine someone's “taste” in movies since it's determined by a magnitude of reasons. Thus to make the system as accurate as possible the model has to both utilize algorithms that can see the bigger picture, i.e see overall trends in a given users review and be able to for example categorize them into a groups, and algorithms that are able to give recommendation of movies that are in close proximity to the movies they have rated, for example sequels of movies.

It was concluded that there were three main areas which the largest improvement stemmed from. Firstly by “simply” improving the quality of the models, secondly by incorporating as stated above different models that are efficient at different tasks. For example using k-nn with is a small k in relationship to the dataset (<50) to find movies that are closely related and then using latent methods for the larger perspective. Lastly, to see which movies were reviewed by each user, instead of what the review themselfs was. This is both a way to see trends in their choices and also to try to gather a larger dataset, by getting users to watch movies that they are more likely to review.

To understand what types of models to use in tandem one would have to both understand what the goal of the prediction is as well as the dataset itself. For example a mixture of both k-nn and latent models might not be as efficient for a dataset where theoretically all users would have given review on most movies where the goal was to determine what a new user would want to watch. Then maybe a different value for k would be better or maybe even replacing k-nn with something else, also in that case it is irrelevant to see whether users have reviewed a movie or not. Finally there is both the issue of whether the model will be overfitted and whether the extra complexity of combining several different models is worth it since it both requires more time to develop and to run.

Implementation of recommendation system

Dataset exploration

We note the following properties of the dataset:

- Number of movies: 2000
- Number of users: 600
- Mean number of reviews per user: 27.5
- Max number of reviews per user: 46
- Min number of reviews per user: 12
- Number of genres: 26
- Mean number of genres per movie: 2.9
- Max number of genres per movie: 8
- Min number of genres per movie: 1

Fig 1: Distribution of the number of reviews per user

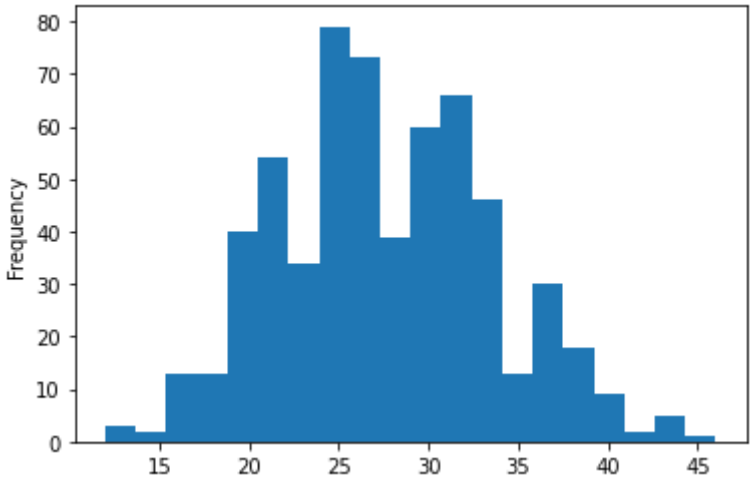
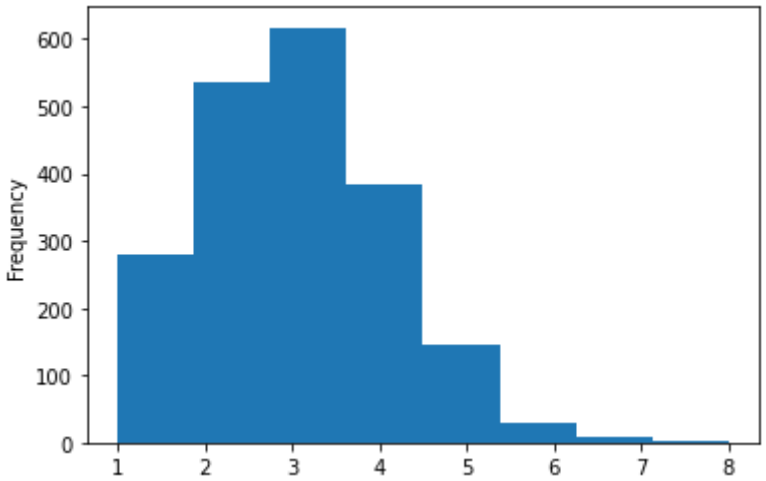


Fig 2: Distribution of the number of genres per movie



After a short exploration of the datasets we note that each user has at minimum made 12 reviews. We also see that the first five users (those that we are interested in) have made 39, 30, 36, 32 and 28 reviews respectively which seems large enough that we should eventually be able to get somewhat accurate recommendations.

Since we also have access to features of each movie in the form of genres, we believe a content-based filtering approach would work well in this case.

Problem formulation and implementation

With the observations of the dataset exploration in mind, we define an objective function using regularized linear regression, which we describe below:

- Create $r_{i,j} = 1$ if user i has rated movie j else 0
- Create $y_{i,j} =$ rating given by user i for movie j
- Create $x_{j,k} =$ feature value k for movie j
- Create $\theta_{i,k} =$ feature value k for user i

For user i, movie j and genre k the function we wish to minimize is:

$$\min_{\theta_i,\dots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n_u}\sum_{j:r_{i,j}=1}^{n_u}\left(\theta_i x_j^T - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_u}\sum_{k=1}^{n_g}(\theta_{i,k})^2$$

where n_u is the number of users and n_g is the number of genres.

The gradient becomes:

$$\frac{\partial J(\theta_i)}{\partial \theta_{i,k}} = \sum_{j:r_{i,j}=1}^{n_u}\left(\theta_i x_j^T - y_{i,j}\right)x_{j,k} + \lambda\theta_{i,k}$$

Here, we assume a user's preference can be described as a linear function of the genres of a movie. A drawback of this is that the model does not take into account how a combination of genres affects the review. For example, if person A loves thrillers and documentaries on their own but dislikes them in combination the linear regression model would not be able to take this into account. Furthermore since the model is content-based it will not be able to see any trends indicating this, unlike a collaborative filtering. A pro with this model is that it assumes a movie that receives a rating of 4 is as close to a rating of 5 as it is to a rating of 3. This could be an issue since some argue that a rating of 4-5 is almost the same while some would argue that a jump from 2-4 is less then from 4-5. Since each user has their own linear model, this will be taken into account.

We implemented regular gradient descent to optimize this objective function using the calculated gradient. Of course there are more advanced and better optimization algorithms but we thought this method would be sufficient for the task at hand (minimizing linear regression error).

Next, we decided to use the Mean Squared Error (MSE) between the predictions and the given reviews as an evaluation metric for our system.

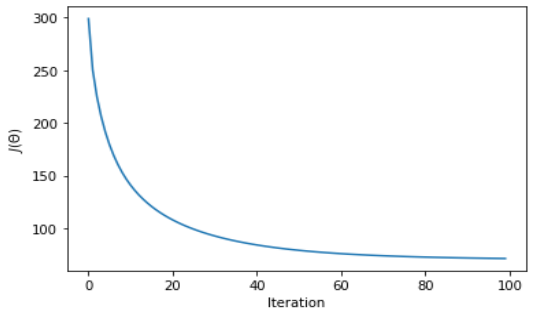
Model fitting and evaluation

In order to get a reasonable estimate for how well our model performs we need to predict the scores of (to a user) unseen movies and compare these predictions to the actual ratings given by the user. In order to do this, we carry out 5-fold cross validation for each of the five users of interest (obtaining between 4 and 7 test movies and >= 23 train movies in each split depending on user), and calculate the MSE of the test sets averaged over these five users. Note that splitting the reviews a user has made into a training and test set as done in this method is required since the preference of each user is fitted individually and without regards to other users.

We do this process a few times in order to obtain somewhat suitable hyperparameters, and pick the hyperparameters resulting in the lowest MSE. We note that using λ=1.8 for 100 iterations of gradient descent with a learning rate of 0.01 results in a MSE of 0.683 when running our evaluation method.

We then perform a final fit on all five users with the full review data, using the obtained hyperparameters. Fig 3. shows how the cost is decreasing during the iterative gradient descent optimization.

Fig 3: Cost function per step of gradient descent in final fit



With the optimized θ's from the final fit, we predict the score of all unseen movies for all users and sort them in decreasing order. Picking out the top-5 scoring movies, we present the following list of recommendations:

Top 5 recommended movies for Vincent:
1) Score: 6.00, Name: What the #$*! Do We (K)now!?
2) Score: 5.90, Name: Dark City
3) Score: 5.68, Name: 9
4) Score: 5.54, Name: The Unborn
5) Score: 5.54, Name: What Lies Beneath

Top 5 recommended movies for Edgar:
1) Score: 6.60, Name: Alpha and Omega 4: The Legend of the Saw Toothed Cave

2) Score: 6.30, Name: The Magic Sword: Quest for Camelot
3) Score: 6.17, Name: 9
4) Score: 6.12, Name: Centurion
5) Score: 6.10, Name: Lilo & Stitch

Top 5 recommended movies for Addilyn:
1) Score: 5.25, Name: Alpha and Omega 4: The Legend of the Saw Toothed Cave
2) Score: 5.05, Name: Alvin and the Chipmunks: The Road Chip
3) Score: 5.05, Name: Alvin and the Chipmunks: Chipwrecked
4) Score: 5.01, Name: Dylan Dog: Dead of Night
5) Score: 5.00, Name: Hannah Montana: The Movie

Top 5 recommended movies for Marlee:
1) Score: 5.55, Name: Zodiac
2) Score: 5.54, Name: Alpha and Omega 4: The Legend of the Saw Toothed Cave
3) Score: 5.53, Name: Zero Effect
4) Score: 5.48, Name: The Net
5) Score: 5.48, Name: Breakdown

Top 5 recommended movies for Javier:
1) Score: 6.17, Name: Sinbad: Legend of the Seven Seas
2) Score: 6.13, Name: Lilo & Stitch
3) Score: 5.93, Name: Alpha and Omega 4: The Legend of the Saw Toothed Cave
4) Score: 5.76, Name: The Magic Sword: Quest for Camelot
5) Score: 5.65, Name: Chicken Run

Looking at Vincent's top three recommended movies we can check if the genres overlap, indicating that the recommendation system is at least somewhat reliable.

What the #$*! Do We (K)now!?:
Comedy, Documentary, Drama, Fantasy, Mystery, Sci-fi
Dark City:
Action, Drama, Fantasy, Mystery, Sci-fi, Thriller
9:
Action, Adventure, Animation, Drama, Mystery, Sci-fi, Thriller

Comedy: 1/3
Documentary: 1/3
Drama: 3/3
Fantasy: 2/3
Mystery: 3/3
Sci-fi: 3/3
Action: 2/3
Adventure: 1/3
Animation: 1/3
Thriller: 1/3

Since we can see a clear overlap between movies in regards to their genres, the recommender system can give at least somewhat reliable recommendations. However more exploration of how well the predictions correspond to user preferences might be carried out to further evaluate the performance of the system.

Evaluation of recommendation systems discussion:

In this case, we had sufficient user reviews to be somewhat certain we could fit a linear model for each of the users we wanted to recommend new movies to, and at the same time have enough data to test our system.
Unfortunately, in real-world scenarios we often do not have sufficient reviews from all users, in fact we probably have few reviews for most users. This makes it incredibly hard to verify the performance of a model in the general case. You can't really know how a user that you know nothing of would rate a movie.
In the real world, one might evaluate a recommendation system in an online-manner, predicting the score of a user and comparing the prediction to the real score once the user watches the movie and gives a review. But the problem of not having enough reviews is still present.
Another approach would be to utilize AB-testing, where group A receives recommendations from one system and group B receives recommendations from a different system. Then one might evaluate which system is the best one. However this seems like an expensive and time-consuming process, and as discussed in the lectures there is an explore-exploit tradeoff where we have to balance the information gained with suboptimal experiments with the utility provided for the user.

Individual Summary and Reflection: Anton

Lecture summary (Recommendation systems)

This lecture first gave some reasoning and motivation for why recommendation systems are useful. It also had quite a large focus on objectives and proxies.
For example, it might be assumed that personalized recommendations for an online book store would make the users more likely to purchase books, in turn increasing sales and profit. Here, this assumption is related to objectives and proxies. The objective of increased profit might be hard to realize directly. Instead we create the proxy of user engagement, which we further would assume correlates with user rating. Thus, our approach to maximizing the profit objective might be to maximize user ratings of purchased products; possibly by recommending items the user is likely to rate highly. We note that in order to provide a personalized recommendation we must always have at least some information regarding the user.

Next, the lecture focused on giving an outline for implementation of a linear regression-based content filtering approach.

Here we learn a mapping from an item and user preference to the user rating of that item. This is possible when we have knowledge about the item.
If we don't have any good features/knowledge of the item, we could utilize the assumption that "Users with similar histories give similar scores to the same products". Then we would use what is called a collaborative filtering approach, in which we learn both user preferences and item features jointly.

Then, there was a LinkedIn case study in which evaluation of recommenders was discussed. One way of evaluating is by using AB testing and experiments. I.e. compare system A to system B on separate groups and see which performs best. This is not trivial in practice as we have an explore-exploit tradeoff.

Reflection on the previous module (Intro to AI problem solving)

- AI is a broad term that often confuses. It is being used in all kinds of settings and has changed a lot from its original meaning.
- In order to solve an AI problem, one should first decide on how to model the problem. After that, a suitable solution may be selected.
- There are often many possible and varying complex ways of tackling a problem. It is important to search broadly and consider many different approaches; some will certainly be more fitting than others for the problem at hand.
- There is no superior approach that best fits all problems. However, some approaches are more general than others. For example, neural networks might be applied to a wide range of problems, although one most often needs to select a suitable network architecture for everything but the simplest tasks.
- Complex is not always better; often it is better to use the simplest solution, at least initially.

Individual Summary and Reflection: Lukas

Lecture summary (Recommendation systems)

This week's lecture gave an overview of recommender systems. Through the usage of a web store selling books the importance of using some sort of recommender systems was explained. Different strategies were explained, for example a uniform strategy that only displays the most popular books overall. However, since individuals have different preferences, it would probably be better to implement some sort of personalized recommendation system. But even if we say we want to maximize profits and thus use a personalized recommendation system we can not just create a ML program with the objective of maximizing profit. We have to use some sort of proxy for profit. Through several examples the difficulty of both choosing the main objective and then finding appropriate proxies and sometimes even proxies for the proxies was illustrated.

Furthermore the most common AI/ML strategy “Empirical risk minimization” was explained. Its purpose is to compare and try to minimize the predicted data to the observed. Then the purpose of regularization was explained which is useful espåecially when you have more features on a dataset then in this case reviews. Then two models, “Content filtering” and “Collaborative filtering” were explained that can utilize this “Empirical risk minimization algorithm”. Content filtering makes predictions based on one's previous review whilst collaborative bases it on similar movies other users similar to you have watched.

Lastly a pull and push model when it comes to delivery mechanism was explained as well as how to test new models against each other in practice. This can be done by applying the new models to a small subset of the users and comparing the new results. However the explore-exploit tradeoff has to be considered as well, that is the most optimal model for profit short term is not maybe the best to get new users and thus the best long term and vice versa.

Reflection on the previous module (Intro to AI problem solving)

Intro to AI problem solving presented several interesting concepts, firstly and in a more general sense, what defines an AI? Since the word itself is used so broadly today it is important to try and define its actual meaning. Is it something that could pass the Turing test? Is it something that could achieve something a normal human would not be able to or is it something even more general. Secondly when trying to solve a problem within the realms of machine learning and AI it is important to realize a few things. Firstly there is not a single model that fits every problem, and on the flip side there is not only one way of solving a problem. Furthermore when choosing how to solve the problem, the simplest ways should be considered since there is no point in making a problem harder just to fit one's desired choice of model. To conclude, there is no clear cut way of defining an AI, more complex models are not always better and could sometimes be even worse and there is no one model that fits all ML problems.