Lecture Notes on Recommendation Systems – Draft Advanced Machine Learning

Prof: Yannet Interian

January 25, 2017

Required Reading[1]:

1. Chapter 9 from "Mining of Massive Datasets". Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman. [2]
2. https://engineering.quora.com/Machine-Learning-at-Quora

Optional: `https://www.youtube.com/watch?v=bLhq63ygoU8`

1 Introduction

Based on "Mining of Massive Datasets". Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman (chapter 9)

Recommendation systems: web applications that involve predicting user responses to options.

Examples:

1. Offering news articles to on-line newspaper readers, based on a prediction of reader interests. (Google News)

[1] Questions from required reading will go into the quizzes

2. Offering customers of an on-line retailer suggestions about what they might like to buy, based on their past history of purchases and/or product searches. (Amazon)

3. Recommending YouTube videos.

4. Netflix offers users recommendations of movies they might like.

5. Pandora recommends songs.

6. Quora recommends stories to users.

We can classify these systems into two broad groups:

- **Content-based systems** consists in matching up the attributes of a user profile with the attributes of a content object (item), in order to recommend to the user new interesting items. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the "cowboy" genre.
- **Collaborative filtering** systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users.

1.1 The Utility Matrix

In a recommendation system there are two classes of entities, which we shall refer to as *users* and *items*. Users have preferences for certain items, and these preferences must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item.

Example: In Figure 1 we see an example utility matrix, representing users' ratings of movies on a 1-5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for Harry Potter I, II, and III, TW for Twilight, and SW1, SW2, and SW3 for Star Wars episodes 1, 2, and 3. The users are represented by capital letters A through D.

Table 1: A utility matrix representing ratings of movies on a 1-5 scale

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Notice that most user-movie pairs have blanks, meaning the user has not rated the movie. In practice, the matrix would be even sparser, with the typical user rating only a tiny fraction of all available movies. The goal of a recommendation system is to predict the blanks in the utility matrix. For example, would user A like SW2? In most real system the goal is to discover some entries in each row that are likely to be high.

1.2 Populating the Utility Matrix

Without a utility matrix, it is almost impossible to recommend items. Acquiring data from which to build a utility matrix is often difficult. There are two general approaches to discovering the value users place on items.

1. *Ask users to rate items* (explicit rating). Movie ratings are generally obtained this way, and some on-line stores try to obtain ratings from their purchasers. This approach is limited in its effectiveness, since generally users are unwilling to provide responses, and the information from those who do may be biased by the very fact that it comes from people willing to provide ratings.

2. *Make inferences from users' behavior* (implicit rating). If a user buys a product at Amazon, watches a movie on YouTube, or reads a news article, then the user can be said to "like" this item.

1.3 Utility Matrix for Implicit Ratings

In the case of implicit ratings, the Utility Matrix has just one value, 1 means that the user likes the item (watch a movie / video, viewed a product). Often, we find a utility matrix with this kind of data shown with 0's rather

Table 2: A utility matrix representing movies watched by users

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	1			1	1		
B	1	1	1				
C				1	1	1	
D		1					1

Table 3: A utility matrix representing movies watched by users

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	1	0	0	1	1	0	0
B	1	1	1	0	0	0	0
C	0	0	0	1	1	1	0
D	0	1	0	0	0	0	1

than blanks where the user has not purchased or viewed the item. However, in this case 0 is not a lower rating than 1. It is no rating at all. More generally, one can infer interest from behavior other than purchasing. For example, if an Amazon customer views information about an item, we can infer that they are interested in the item, even if they don't buy it.

If in the example for Table 1 instead of having ratings we had the information that the user watch a particular movie our initial utility matrix would look like in Table 2. For this case we would consider instead the matrix in Table 3, where blanks are replaced by 0's.

2 Content-Based Recommendations

In a content-based system, we must construct for each item and each user a *profile*. A profile is a record or collection of records representing important characteristics of that item or user. In simple cases, the profile consists of some characteristics of the item (user) that are easily discovered. For example, consider the features of a movie that might be relevant to a recommendation system.

1. The set of actors of the movie. Some viewers prefer movies with their favorite actors.

2. The director. Some viewers have a preference for the work of certain directors.

3. The year in which the movie was made. Some viewers prefer old movies, others watch only the latest releases.

4. The genre or general type of movie. Some viewers like only comedies, others dramas or romances.

Can we get features from the movie descriptions or reviews? How can we compute the profile of a news article?

Pandora hires musicologists to characterize songs, built the Music Genome. (`https://www.pandora.com/about/mgp`)

2.1 Features from documents

We may then take as the features of a document the n words with the highest TF-IDF scores. Think of the sets of high TF-IDF words as a vector, with one component for each possible word.

1. Remove stopwords

2. Compute TF-IDF scores

3. Keep works with high score (top $n = 1000, 5000, 1M$)

Computing TF-IDF
TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

TF: Term Frequency, which measures how frequently a term occurs in a document.

$$\mathrm{TF}(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}.$$

IDF: Inverse Document Frequency, which measures how important a term is with respect to a corpus. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$\text{IDF}(t) = \log(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}).$$

$$\text{TF-IDF}(t) = \text{TF}(t) \cdot \text{IDF}(t)$$

Example: Consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., TF) for cat is then $(3/100) = 0.03$. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., IDF) is calculated as $log(10,000,000/1,000) = 4$. Thus, the TF-IDF weight is the product of these quantities: $0.03 * 4 = 0.12$.

How would you construct features for images?

2.2 Representing Item Profiles and Users Profiles

Our ultimate goal for content-based recommendation is to create both an item profile consisting of feature-value pairs and a user profile using the same features as the item profiles. The user profile summarizes the preferences of the user.

Example 1: **News article recommendations**
First we select N words to represent the space of all important works in the vocabulary.
Item profile (document):
Vector of 0's and 1's, where a 1 represented the occurrence of a high TF-IDF word in the document. (length N)
User profile:
Summary of the profile of all of the articles that this user have read.
$x = (p_1, \ldots p_N)$ where p_i is the probability that a use likes an article with

the word i.

Example 2: **Netflix recommendations**
Item profile:
x_1 = has Julia Roberts,
x_2 = directed by Lars von Trier,
x_3 = is horror,
x_4 = is a comedy
User profile:
x_1 = probability that the user likes movies with Julia Roberts,
x_2 = probability that the user likes movies directed by Lars von Trier,
x_3 = probability that the user that likes horror movies,
x_4 = probability that the user likes comedies

Item profiles and movie profiles can also have numerical features such as the movie rating. We can combine numerical and boolean features, although we should give some thought to the appropriate scaling of the non-boolean components, so that they neither dominate the calculation of distance nor are they irrelevant.

2.3 Similarities

To recommend items we need a notion of similarities between items and users. The appropriate similarity to use in a particular domain is subject of research. Here are some common examples.

Jaccard Similarity
Given A and B two sets the Jaccard similarity is defined as.

$$\frac{|A \cap B|}{|A \cup B|}$$

Now suppose I have two vector x, y in R^n how would you define a Jaccard similarity between x and y? In which examples the Jacard Similarity make sense?

We could ignore values in the matrix and focus only on the sets of items rated. If the utility matrix only reflected purchases, this measure would be

a good one to choose. However, when utilities are more detailed ratings, the Jaccard distance loses important information.

Cosine Similarity
Give vector x and y in R^n the cosine similarity if defined as:

$$cosine(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Cosine similarity is a measure of similarity between two vectors. It measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

Cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval and text mining, each term is assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter.

Pearson Correlation Coefficient (Similarity)
The Pearson Correlation Coefficient for vector x and y in R^n is defined as:

$$r(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

In statistics, the Pearson correlation coefficient is a measure of the linear correlation between two variables, giving a value between $+1$ and -1 inclusive, where 1 is total positive correlation, 0 is no correlation, and -1 is total negative correlation. It is widely used as a measure of the degree of linear dependence between two variables.

For the homework we are using a related metric that accounts for the fact that using are rating just a subset of the items. Given users i and j, let I_i

be the set of items that user i has rated. We can define similarity between i and j as:

$$\frac{\sum_{k \in I_i \cap I_j} (y_{ik} - \bar{y}_i)(y_{jk} - \bar{y}_j)}{\sqrt{\sum_{k \in I_i \cap I_j} (y_{ik} - \bar{y}_i)^2 \sum_{k \in I_i \cap I_j} (y_{jk} - \bar{y}_j)^2}}$$

Where $\bar{y}_i = \frac{1}{|I_i|} \sum_{k \in I_i} y_{ik}$ is the average rating for user i. If $I_i \cap I_j = \emptyset$ the similarity should be 0.

2.4 Recommending Items to Users based on Content – Unsupervised

With profile vectors for both users and items, we can estimate the degree to which a user would prefer an item by computing a *similarity* between a user and an item and recommend to a user items with *high similarity*.

2.5 Recommending Items to Users – Supervised

A completely different approach to a recommendation system is to treat the problem as one of machine learning. Regard the given data as a training set, build a classifier that predicts rating based on item features and user features. You can use your predictions to give recommendations. To predict ratings we can use any regression algorithm such as linear regression or random forest. Note that this approach can also be used with features that are not based on content.

2.6 Advantages of Content Based Recommendations

- You don't need data on other users.
- You don't have cold-start problem for a new item. Able to recommend new and unpopular items.
- You can provide explanations of recommended items by listing content features that caused an item to be recommended.

2.7 Challenges with Content Based Recommendations

What is the problem with content based recommendations?

- Constructing the feature vector could be a difficult task (need domain knowledge).
- New genres "dogme 95".
- Some kid of items are not amenable to easy feature extraction methods (movies, music)
- Hard to exploit quality of judgements of other users.

2.8 Cold Start Problem

Personalized recommender systems take advantage of users past history to make predictions. The cold start problem concerns the personalized recommendations for users with no or few past history (new users).

The cold-start problem can addressed in many ways including:

- Recommending popular items
- Asking the user at the beginning to rate some items
- Asking the users explicitly to state their taste in aggregate
- Recommending items to the user based on the collected demographic information

There is a similar cold start problem for items. This problem can be addressed by using a hybrid approach between content-based and collaborative filtering systems. New items would be assigned a rating based on the ratings from similar items. Item similarity would be determined based on the items content-based characteristics.

3 Recommendation as Classification: YouTube

One example of using classification for recommendation is the YouTube recommendation system [1]. The prediction problem becomes accurately classifying a specific video watch w_t at time t among millions of videos i (classes) from a corpus V based on a user U and context C. This system models ranking of a video for a particular user in a context as a multi-class classification problem.

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

where u represents a high-dimensional "embedding? of the user, context pair and the v_j represent embeddings of each candidate video.

They use neural networks to learn user embeddings u as a function of the user's history and context that are useful for discriminating among videos with a softmax classifier.

4 Collaborative Filtering Recommendations

Collaborative filtering recommends items by matching users with other users having similar interests. The items recommended to a user are those preferred by similar users.

Collaborative filtering systems are usually categorized into two subgroups: memory-based and model-based methods. Memory-based methods memorize the utility (rating) matrix and make recommendations based on the relationship between the queried user and the rest of the utility matrix. Model-based methods fit a parameterized model to the given utility matrix and then make recommendations based on the model.

Here we first describe a memory based approach then we introduce a model based approach called Matrix Factorization.

4.1 Memory Based Algorithms

Let $y_{ij}, i \in 1, \ldots n_u$ and $i \in 1, \ldots n_m$ be the rating on item j given by user i where n_u is the number of users while n_m is the number of items. Let $\bar{y}_i$ be the mean rating for user i, that is

$$\bar{y}_i = \frac{1}{|I_i|} \sum_{j \in I_i} y_{ij}$$

where I_i is the set of items that user i has rated.

For user i and movie / item k we can predict rating $\hat{y}_{ik}$ to be

$$\hat{y}_{ik} = \bar{y}_i + \frac{1}{\sum_{a \in U_k} |w_{ia}|} \sum_{a \in U_k} w_{ia}(y_{ak} - \bar{y}_a)$$

where U_k are the users that have rated item k. The quantity w_{ia} is a similarity between users i and a. You can use any definition of similarity that is appropriate for your problem. Note that for this to work higher values of similarity imply that uses are more similar.

4.2 Low Rank Matrix Factorization (MF)

An entirely different approach to estimating the blank entries in the utility matrix is to conjecture that the utility matrix is actually the product of two long, thin matrices. This view makes sense if there are a relatively small set of features of items and users that determine the reaction of most users to most items. In this section, we sketch one approach to discovering two such matrices; the approach has different names: "UV-decomposition", "low rank matrix factorization."

$$\begin{bmatrix} 5 & 1 & 4 & 5 & 1 \\ & 5 & 2 & 1 & 4 \\ 1 & 4 & 1 & 1 & 2 \\ 4 & 1 & 5 & 5 & 4 \\ 5 & 3 & 3 & & 4 \\ 1 & 5 & 1 & 1 & 1 \\ 5 & 1 & 5 & 5 & 4 \end{bmatrix} \approx \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1K} \\ u_{21} & u_{22} & \ldots & u_{2K} \\ u_{31} & u_{32} & \ldots & u_{3K} \\ u_{41} & u_{42} & \ldots & u_{4K} \\ u_{51} & u_{52} & \ldots & u_{5K} \\ u_{61} & u_{62} & \ldots & u_{6K} \\ u_{71} & u_{72} & \ldots & u_{7K} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{21} & v_{31} & v_{41} & v_{61} \\ v_{12} & v_{22} & v_{32} & v_{42} & v_{62} \\ \vdots & & & & \\ v_{1K} & v_{2K} & v_{3K} & v_{4K} & v_{6K} \end{bmatrix}$$

As before, we call the elements of the utility matrix $y_{ij}, i \in 1, \dots n_u$ and $i \in 1, \dots n_m$ where n_u is the number of users while n_m is the number of items. The decomposition has the equation $\hat{Y} = UV^T$ where U is $n_u \times K$ and V is $n_m \times K$. Every rating y_{ij} gets the prediction

$$\hat{y}_{ij} = u_i \cdot v_j$$

Here is a toy example in witch we have 7 users and 5 movies $K = 2$.

$$\begin{bmatrix} 5 & 1 & 4 & 5 & 1 \\ & 5 & 2 & 1 & 4 \\ 1 & 4 & 1 & 1 & 2 \\ 4 & 1 & 5 & 5 & 4 \\ 5 & 3 & 3 & & 4 \\ 1 & 5 & 1 & 1 & 1 \\ 5 & 1 & 5 & 5 & 4 \end{bmatrix} \approx \begin{bmatrix} 0.2 & 3.4 \\ 3.6 & 1.0 \\ 2.6 & 0.6 \\ 0.9 & 3.7 \\ 2.0 & 3.4 \\ 2.9 & 0.5 \\ 0.8 & 3.9 \end{bmatrix} \times \begin{bmatrix} 0.0 & 1.5 & 0.1 & 0.0 & 0.7 \\ 1.3 & 0.0 & 1.2 & 1.4 & 0.7 \end{bmatrix}$$

It is important to understand what these matrices represent. U is the representation of users in some low dimension space (say romance, action). And V is the representation of products in the very same low dimension space. However, we don't know exactly what these factors mean. They could be romance/action or they could each be a combination of multiple genres. This is also why this method is sometimes called latent factor Matrix Factorization.

4.3 Computing similar movies (items)

The row vector $v_i = (v_{i1}, \dots v_{iK})$ can be interpret as a learnt feature vector for item i. If we now wanted to find items related to i we can look for items that v_j that have small Euclidean distance to i. That is we want

$$||v_i - v_j||$$

to be small.

Can we compute similar users?

4.4 Root-Mean-Square Error

While we can pick among several measures of how close the product UV^T is to M (Utility Matrix), the typical choice is the root-mean-square error (RMSE), where we

- Sum, over all **nonblank** entries in M the square of the difference between that entry and the corresponding entry in the product UV^T .
- Take the mean (average) of these squares by dividing by the number of terms in the sum (i.e., the number of nonblank entries in M).
- Take the square root of the mean.

Let r_{ij} be 1 if the element y_{ij} of the Utility Matrix is **nonblank** and 0 otherwise. Let u_i (v_i) be the vector corresponding to the i row of the matrix U (V). The Root-Mean-Square Error is defined as

$$\sqrt{\frac{1}{N}\sum_{(i,j):r_{ij}=1}(y_{ij}-u_iv_j)^2}$$

where N is the number of non-blank elements of the utility matrix M.

Minimizing the sum of the squares is the same as minimizing the square root of the average square, so we generally omit the last two steps in our running example.

4.5 Gradient descent.

We can minimize the Mean Square Error using gradient descent. Let's first review gradient descent.

If we are trying to learn some weights w and we have some error function $E(w)$, at each iteration we update w as a small step into the direction of the negative gradient

$$w^{(t+1)} = w^{(t)} - \eta\nabla E(w^{(t)})$$

where the parameter η is known as the learning rate.

4.6 Minimizing Root-Mean-Square Error

We are trying to minimize $\sqrt{\frac{1}{N}\sum_{(i,j):r_{ij}=1}(y_{ij}-u_iv_j)^2}$, which is equivalent to minimizing

$$E(U,V)=\frac{1}{N}\sum_{(i,j):r_{ij}=1}(y_{ij}-u_iv_j)^2$$

where N is the number of non-blank elements of M and $u_iv_j=\sum_{k=1}^{K}u_{ik}v_{jk}$. Consider the derivative of E with respect to u_{ik}, note that u_{ik}

$$\frac{\partial E}{\partial u_{ik}}=-\frac{2}{N}\sum_{j:r_{ij}=1}(y_{ij}-\sum_{k=1}^{K}u_{ik}v_{jk})v_{jk}$$
$$\frac{\partial E}{\partial v_{jk}}=-\frac{2}{N}\sum_{i:r_{ij}=1}(y_{ij}-\sum_{k=1}^{K}u_{ik}v_{jk})u_{ik}$$

Finally, the gradient descent equations are

$$u_{ik}^{t+1}=u_{ik}^{t}+\frac{2\eta}{N}\sum_{j:r_{ij}=1}(y_{ij}-\sum_{k=1}^{K}u_{ik}^{t}v_{jk}^{t})v_{jk}^{t}$$
$$v_{jk}^{t+1}=v_{jk}^{t}+\frac{2\eta}{N}\sum_{i:r_{ij}=1}(y_{ij}-\sum_{k=1}^{K}u_{ik}^{t}v_{jk}^{t})u_{ik}^{t}$$

We can write the equations in a compact form:

$$u_{i}^{t+1}=u_{i}^{t}+\frac{2\eta}{N}\sum_{j:r_{ij}=1}(y_{ij}-u_{i}^{t}v_{j}^{t})v_{j}^{t}$$
$$v_{j}^{t+1}=v_{j}^{t}+\frac{2\eta}{N}\sum_{i:r_{ij}=1}(y_{ij}-u_{i}^{t}v_{j}^{t})u_{i}^{t}$$

Initialization
This cost (error) function may have many local minima. We may need to run the algorithm from many random starting points.

Normalization
We can subtract from y_{ij} the average rating by the user i. At prediction time, we need to undo the normalization.

Learning Rate
To choose η try $0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1$ and choose and η such that the cost (error) decreases at each iteration.

4.7 Avoiding overfitting

We can add a regularization term to our error function.

$$\frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2 + \lambda (\sum_{i=1}^{n_u} \sum_{j=1}^{K} u_{ij}^2 + \sum_{i=1}^{n_m} \sum_{j=1}^{K} v_{ij}^2)$$

Where $N = \sum_{ij} r_{ij}$

How would the gradient descent equations change?

4.8 Matrix Factorization as Automatic Feature Selection

Note that the Matrix Factorization algorithm gives us a vector in $\mathbf{R}^K$ for every user i and every item j. This approximation gives a prediction $u_i v_j$ for the rating the user i gives to item j. Also, we can use these vector to compute distances between two different users and two different items.

4.9 Training, Testing, Validation sets

We can take a sample of the known ratings as a test / validation sets.

4.10 Non Negative Matrix Factorization (NMF)

Nonnegative matrix factorization (NMF) approximates a nonnegative matrix by the product of two low-rank nonnegative matrices.

Non-negative matrix factorization (NMF) is another type of matrix factorization. Given M, a non-negative matrix and K, the number of factors, find U and V with K columns such that we minimize the same error

$$\frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2$$

with the constrain that $U \geq 0$ and $V \geq 0$. NMF is very popular in various domains because the matrix U and V are often intuitive and easy to interpret because they are sparse.

NMF finds applications in such fields as computer vision, document clustering audio signal processing and recommender systems.

4.11 NMF for Topic Modeling

In NMF decompositions the matrices U and V has non negative entries. This makes results from NMF easy to interpret which makes it useful in clustering applications. In particular NMF has been widely used as a clustering method especially for document data, and as a topic modeling method.

To apply NMF to a set of documents we first find a set of terms (words) that are going to be used to represent the documents. A row of the utility matrix corresponds to a document, a column corresponds to each term. An element in the matrix could be the tf-idf score of a term in a document or 0 if the term doesn't appear in the document.

The decomposition give us a the weights of every document in each of the K topics (U) and the weights of very word in each of the K topics (V).

5 Interview Questions

1. What types of recommender systems exist and can you describe them?
2. How to evaluate a recommender system?
3. How to design a recommender system for scientists on w.mendeley.com?
4. What is the "cold start" phenomenon?
5. How would you design a recommendation system (like amazon)?
6. How do you build a system similar than Netflix?
7. Give an example of an application of non-negative matrix factorization.

References

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA, 2016. ACM.

[2] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2nd edition, 2014.