Table of Contents

[Abstract 1](#_Toc75373444)

[Recommender Systems 1](#_Toc75373445)

[Collaborative Filtering recommender systems 2](#_Toc75373446)

[Content Based recommender systems 3](#_Toc75373447)

[Hybrid recommender systems 3](#_Toc75373448)

[Probabilistic Models for Recommender Systems 4](#_Toc75373449)

[fLDA 4](#_Toc75373450)

[CTR 5](#_Toc75373451)

[CTPF 6](#_Toc75373452)

[LDA 7](#_Toc75373453)

[Formalization 7](#_Toc75373454)

[Learning 8](#_Toc75373455)

[Inference and Parameter Estimation 9](#_Toc75373456)

[Variational Inference 10](#_Toc75373457)

[KL–divergence Derivation 11](#_Toc75373458)

[Jensen’s inequality method 11](#_Toc75373459)

[Mean-field VI 12](#_Toc75373460)

[Mean-field VI in conjugate models 12](#_Toc75373461)

[Mean-field VI in non-conjugate models 14](#_Toc75373462)

[Conjugate Priors and Corresponding Posteriors 14](#_Toc75373463)

[Multinomial distribution and Dirichlet priors 14](#_Toc75373464)

[Poisson distribution and gamma priors 15](#_Toc75373465)

[OPE and BOPE 16](#_Toc75373466)

[OPE for solving 17](#_Toc75373467)

[BOPE for solving 18](#_Toc75373468)

[Collaborative Topic Model for Poisson distributed ratings 19](#_Toc75373469)

[Formalization 19](#_Toc75373470)

[Learning 21](#_Toc75373471)

[Learning Parameters 22](#_Toc75373472)

[Key properties 27](#_Toc75373473)

[Interpretable user profiles 27](#_Toc75373474)

[Evaluation 27](#_Toc75373475)

[Empirical Studies 28](#_Toc75373476)

[Data fetching 28](#_Toc75373477)

[MovieLens 20M 28](#_Toc75373478)

[NETFLIX 29](#_Toc75373479)

[Data pre-processing 29](#_Toc75373480)

[Dataframe creation 29](#_Toc75373481)

[Data cleaning and wrangling 30](#_Toc75373482)

[Vocabulary Extraction 30](#_Toc75373483)

[Document Representation 31](#_Toc75373484)

[Memory reduction 31](#_Toc75373485)

[Model Fit 32](#_Toc75373486)

[Setting Parameters 32](#_Toc75373487)

[Challenges during Python implementation 33](#_Toc75373488)

[Running on Google Cloud 34](#_Toc75373489)

[Model Evaluation 35](#_Toc75373490)

[Transfer Learning between MovieLens 20M and NETFLIX 35](#_Toc75373491)

[References 39](#_Toc75373492)

# Abstract

**.**

**.**

**.**

# Recommender Systems

Nowadays, Recommender Systems are widely recognized as one of the most beneficial applications of Machine Learning, and there is no doubt that they drive nearly every aspect of our lives. The fundamental objective of these Machine Learning-driven Recommenders is to filter, prioritize and efficiently deliver the necessary information to the consumers in the midst of overwhelmingly numerous choices on the internet. It is also described as:

*“Any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options.”* (Burke, 2002)

Therefore, many companies utilize the recommender systems for the purpose of helping the consumers discover new and relevant items such as movies, musics, jobs, etc. They use the consumer data which may be in explicit or implicit form (e.g. likes, clicks), in order to comprehensively assess consumers’ preferences and then recommend the relevant items to them. Due to the various criterias, there are multiple techniques of recommender systems, each of which differs in how a single recommendation is generated. The most common types of recommender systems are described in the following section.

Although there exists a number of different recommender systems in the literature, we will focus on the three most common ones below:

Diagram

Description automatically generated

## Collaborative Filtering recommender systems

Collaborative Filtering recommender systems are one of the most widely used systems next to the content-based recommender systems. Essentially, these systems create a user profile based on the ratings of various items and then aims to compare these against a wider user group[12]. As the word “collaborative” from the name implies, multiple users come together as group – taste of one user will be similar to the other users of group. Therefore, by utilizing the users data which contains their historical preferences on a set of items, system deploys an assumption that the users who have previously agreed are more likely to agree again in the future. So, the system creates the new recommendations by taking the similarities between users based on the ratings into consideration.

Diagram

Description automatically generated

Although, CF recommender systems have been used in the industry for many years [13], they still have a limitation such that they can not address the cold start problem –they are not able to recommend items which are not rated by any users (e.g, new items). As a result, it is possible that only famous items may get recommended. Furthermore, traditional CF systems are also memory-wise and computationally expensive and suffers from scalability problems.

## Content Based recommender systems

While CF recommender systems, as discussed above, recommend the products or items according to the similarities of user preferences which means that recommendation relies on the user-item interactions, Content Based recommender systems, on the other hand, aims to recommend products or items similar to those a given user has rated positive or liked in the past. So, CB systems generate recommendations based on the comparison between the content of the items and the user profile which was created according to the historical user data. Note that the content of items is described by terms, tags, features or even plots in case if the items are movies.

An algorithm used to recommend the movies on the Netflix platform is the prominent example which resembles this recommender systems. If a certain user watches and comedy movie and rates it positive via votes or comments, then the new movie recommendations with the same label of that liked movie will be suggested to the user. In other words, based on the content of the consumed item, these recommender system finds other similar items and recommend them. Note that the such website platforms often keeps the techniques of how the content is actually labeled and matched against each other as secret [12]. Contrary to the CF systems, CB system doesn’t suffer cold-start problem and they can suggest not only famous or older items, but also the unpopular or new items. In addition to this, they are memory-wise and computationally cheap because there is no need for the data of other users in order to be able to compute the recommendation for a specific user.

Diagram

Description automatically generated

## Hybrid recommender systems

Hybrid recommender systems combine two or more types of traditional recommender systems in order to have better performance by benefiting from complementary advantages of subsystems. Hybrid systems which combine collaborative filtering and content based approaches, achieves state-of-the-art results in many cases and are used in many large scale recommender systems nowadays. Detailed comparison on advantages and disadvantages of Hybrid Recommenders along with Collaborative Filtering and Content Based Recommenders are shown in table below**[table#]:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Collaborative Filtering** | **Content Based** | **Hybrid** |
| **Number of users** | * Recommendation based on many users having similar interest | * Recommendation based on single user | * Combination of collaborative and content based filtering |
| **Disadvantages** | * Cold start problem * Data sparsity * Scalability * Memory-wise and computationally expensive | * Limited content analysis * Over-specialization | * Increased complexity * Increased expense of implementation |
| **Advantages** | * Serendipitous recommendation * User and item features are not required * Quality may improve over time as more users interact with items * - Minimal domain knowledge required | * User independent * No cold start problem * Interpretable results * Memory-wise and computationally cheap | * Avoids most of the shortcomings of other approaches. |

# Probabilistic Models for Recommender Systems

**[\*Refer CTR, CTPF papers for extensiveness\*].** The application of probabilistic modeling to the recommendation problem has a rich history which dates back to decades (Breese et al. 1998, Hofmann 2004, Marlin 2004). Many authors incorporated the probabilistic approaches into models which explained the dataset. Initial approaches were probabilistic graphical models such as Bayesian networks and Dependency networks which eventually left their place with subsequent novel topic models such as Latent Dirichles Allocation (LDA) [3] and Probabilistic Latent Semantic Analysis (pLSA) [22]. The term “latent” is used in their name, because both of them are considered probabilistic topic models and the topics they aim to find from the corpus are treated as latent or hidden variables. Detailed explanation of LDA has been discussed **in its own section on later pages**. Note that, asboth models can suggest items which have similar content to other items that a user likes, they have been extensively used for Content Based recommender systems.Furhtermore**,** when it comes to the field of Collaborative Filtering recommender systems, the matrix factorization technique had gained a decent popularity, especially after combined with probabilistic approach [18], [19], [20], [21].

## fLDA

Lately, there has been a lot of interest in combining probabilistic topic modelling with matrix factorization in the field of hybrid recommender systems. One of the major for this is that when a content of item is represented by topic models, the models benefit from interpretable semantics of the latent space characterized by the topic mixtures and this leads to more an interpretable semantics of the item latent factor[\*\*CTMP\*\*]. Initially, Agarwal and Chen proposed probabilistic topic modelling in matrix factorization with fLDA [23] where the item latent factor took the role of topic proportion in the LDA representation. Despite of being an accurate and interpretable model which handles both cold-start and warm-start scenarios, fLDA still had a limitation in dealing with distinguish items where there is an identical topic mixture, but content details that topic mixture can not cover are of concern to different groups of people. To elaborate on this limitation more, consider that we have two articles; A and B, and both of the articles are about an application of machine learning to social networks. Because both articles are identical in terms of their contents, they will also possesses same topic proportions. Now let’s consider that these two articles are of interest to different kind of users: Article A provides a prominent machine learning algorithm which is applied to social network applications, wheares article B implements a standard machine learning algorithm, but provides a crucial data analysis on social network data. As a result, users who work in machine learning will prefer article A and will hardly be interested in article B, wheares users who work in social networks will be more interested in article B instead of A. However, as the topic proportions of both articles are same, both of them will be recommended to both groups of users [24].

## CTR

To tackle the limitation mentioned above, a novel approach called Collaborative Topic Regression (CTR) [24] has been proposed by David M. Blei and Chong Wang. The way CTR addresses that limitation is by allowing the item latent factor be an offset from topic proportion. So, by this way, an offset may help explain, for instance, an article A is more important to researchers interested in machine learning than it is to those interested in social network analysis. Therefore, CTR allows the item latent factor to also account for user ratings.

Fundamentally, CTR incorporates techniques of both collaborative filtering based on latent factor models and content analysis based on probabilistic topic modelling. According to CTR model, items are generated by a topic model while users are represented with topic interests [24]. Therefore, CTR is considered as one of the excellent hybrid models which shows that the combination of the content modelling with the matrix factorization methods produces more promising results compared to traditional recommender systems. The graphical model of CTR along with its algorithm is shown below.

**CTR Graphical Model**

Diagram

Description automatically generated

**CTR Algorithm**

1. For each user *u*,draw user latent vector
2. For each item *j*,
   1. Draw topic proportions
   2. Draw item latent offset and set the item latent vector as .
   3. For the *n*-th word of item *j*,
      1. Draw topic index
      2. Draw word
3. For each user-item pair (*u, j*), draw the rating

where is the confidence parameter for . For instance, we trust more if is large.

Despite its advantages, CTR model has significant computational limitations as well. The reason is that the model considers user ratings to have a Gaussian distribution which leads to iterating over all of the entries in rating matrix during training. Because of this, CTR is highly inefficient considering that real-world datasets are very big and sparse.

## CTPF

In order to address CTR’s inefficiency mentioned above, a newer hybrid model called Collaborative Topic Poisson Factorization (CTPF) has been proposed [25]. Fundamentally, CTPF model makes an assumption such that both ratings and items of dataset have a Poisson distribution. By this way, CTPF is only concerned with non-zero ratings during training, and therefore it is much more efficient and scalable. Furthermore, for the purpose of making sure that the model is conditionally conjugate and has closed-form updates, CTPF tries to model the content generation by standard mixtures of Gamma distribution. The graphical model of CTPF along with its algorithm is demonstrated below.

**CTPF Graphical Model**

Diagram

Description automatically generated

In machine learning, graphical models are used to represent a repetitive process of the probabilistic model. Essentially, they represent a factorization of joint distribution of hidden and observed random variables. Nodes are random variables, plate boxes denote the “loop” with variable shown in the bottom right corner of the plate representing its number of iterations, and edges mean that there is some kind of dependence between random variables in generative process. Note that grey nodes represent observed variables while blank nodes are hidden variables. In the graphical model[\*figx\*] of CTPF above … explain…

**CTPF Algorithm**

1. **Item model:**
   1. Draw topics
   2. Draw item topic intensities
   3. Draw word count
2. **Recommendation model:**
   1. Draw user preferences
   2. Draw item topic offsets
   3. Draw

All the hybrid models mentioned so far (i.e. fLDA, CTR and CTPF) benefit from the interpretable semantics of the item latent factor. However, they still have some limits in terms of computational cost or predictive performance. Therefore, in this thesis, we will explore and implement recent hybrid model called ***Collaborative Topic Model for Poisson distributed ratings (CTMP)*** model which covers the limitation of CTR by considering ratings in Poisson distribution as CTPF does, while modelling contents with LDA [\*\*CTMP\*\*]. Details of CTMP formalization, graphical model and algorithm is shown **in its own section on later pages**. However, it is worthwhile mentioning that CTMP makes the following contributions to the previously mentioned approaches [\*\*CTMP\*\*]:

1. CMTP has been tested in variety of fields where real-world recommendation is the most challenging one. It has been seen that CTMP outperforms the other existing models significantly and its main competency is in recommending scientific articles and commercial product recommendation. Recommending movies are also amongst these, and indeed, this is what we will implement and test on this thesis.
2. CTMP implements fast and scalable coordinate ascent algorithm because it is non-conjugate model. An implemented algorithm is fast and scalable.
3. According to the experimental studies we conducted on different real-world datasets in this work, we observe the sparse estimates of topic mixtures can be achieved via learning in spite of the fact that the model specification does not encourage so. Note that the sparsity is very critical property as it leads to an efficient storage of a data by offering compact content representation. (Note that we mostly get sparse estimates rather than dense compared to original paper)

# LDA

## Formalization

In machine learning, topic modeling is a statistical model for discovering a set of topics that occur in a collection of documents [1]. It is also considered as a probabilistic model which offers an interpretable low-dimensional representation of the documents. For many years, implementation of topic models for the purpose of document classification, corpus exploration and information retrieval has been of interest.

There are many topic modeling algorithms, among which Latent Dirichlet allocation (LDA) is the most popular one. LDA is a three-level hierarchical Bayesian model, and its basic idea is that documents are represented as random mixtures over underlying set of topics, where each topic is characterized by a distribution over words which are biased around those associated under a single theme [\*\*original-lda-paper\*\*]. Therefore, topic probabilities express an explicit representation of each document. This can also be be explained as:

* **Each document is a mixture of topics**.

Each document contains terms/words from some topics in specific proportions. For instance, if we consider that there are 2 topics in the whole corpus, then we might state that some document could be 75% topic A, and 25% topic B, while another document might be consisted of 30% topic A, and 70% topic B.

* **Each topic is a mixture of words**.

Each topic is expressed by the words that explain it most. For example, if we consider that there are 2 topics, namely, “sports” and “education”, in the whole corpus, then the most used words for the sports topic could be “teammate”, “win”, and “play”, while the education topic could contain the words such as “lecture“, “book” and “class”. It is necessary to note that the same words can appear on the multiple topics. For example, the word “time” could participate in both sports and education topics.

By this way, documents can overlap with each other with relationship to their contents, rather than being seperated into different individual groups. Generative process and graphical model of LDA for each document in the whole corpus are described below.

## Learning

**Terminology**

* A *word* is a term of the vocabulary and it is indexed by .
* A *document* is a series of words given by where is the *n*th word inside the document.
* A *corpus* is a collection of a total ***J***documents and it is given by

**Graphical Model**

Diagram

Description automatically generated

In machine learning, graphical models are used to represent a repetitive process of the probabilistic model. Essentially, they represent a factorization of joint distribution of hidden and observed random variables. Nodes are random variables, plate boxes denote the “loop” with variable shown in the bottom right corner of the plate representing its number of iterations, and edges mean that there is some kind of dependence between random variables in generative process. Note that grey nodes represent observed variables while blank nodes are hidden variables. In the graphical model[\*figx\*] of LDA above, the outer box illustrates the document , and the inner box illustrates repeated choice of topics and words inside the respective document. Each is sampled once in each loop. The parameters that are considered as corpus-level areand , and they are supposed to be sampled during the process of generating the corpus [26]. Lastly, the variable given as and are considered as word-level, and in each loop, they are sampled once for each word inside every document. Note that, during the process, topics are sampled repeatedly within each document. Respective algorithm is described below. It is also important to emphasize that overall LDA process is hidden generative process and according to this process, the model is assumed to generate the observed data (e.g. documents). Obviously, this was just a generative assumption in order to facilitate the algorithm and it does not illustrate the true process of the real data [17].

NOTE: <https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation>

We need Plate notation for LDA with Dirichlet-distributed topic-word distributions.

So, change current Graphical model, add second simplex for describing topics. Include it in algorithm too.

In short, examine Dirichlet distributed LDA from wiki page above, because we use this as example Variational Inference too.

**Algorithm**

1. Draw topic proportions .
2. For the *n*-th word of document *j*,
   1. Draw topic index
   2. Draw word

In the algorithm above, is the topic proportion of particular document, while is the topic representation of particular topic and is the Dirichlet prior parameter. As seen above, the topics that algorithm tries to find from the whole corpus are treated as latent/hidden variables. Additionally, each document of the corpus is represented in terms of topic proportions/latent themes which are also hidden variables.Topic proportion is a *k*-dimensional Dirichlet random variable and its domain is in the (*k* − 1)-simplex. In other words, *k*-vector is in the (*k* − 1)-simplex, therefore, . The Dirichlet is a exponential family distribution on the simplex. One of its important properties is that it is conjugate prior to the multinomial distribution [4]. Note that, this conjugacy helps the development of the inference and parameter estimation for LDA, and it will be discussed later in Variational Inference section.

Illustrating the geometry of the latent space is a another good way for grasping the concept of LDA:

Diagram

Description automatically generated

In summary, Each document in the corpus falls inside the simplex of topics in an example figure above.

## Inference and Parameter Estimation

Computing the posterior distribution over the latent variables given some documents is the the main inferential problem here, because the posterior inference is intractable to compute:

Do you learn B? Change WHOLE LDA

As normalization constant which depends on model parameters i.e., marginal probabiliy contains intractable integrals above, the resulting posterior inference also becomes intractable to compute. Therefore, as an exact posterior distribution is not possible, several approximate inference algorithms can be used for LDA. For example, Variational Inference and relevant variational EM algorithm can be used in order to learn the topics and decompose each document of the corpus according to these learnt topics [26]. Details of Variational Inference are discussed in the following section.

# Variational Inference

Variational Bayesian Methods (i.e. Variational Inference) are a group of widely used techniques in a field of statistical Machine Learning. Suppose the following probabilistic model with the joint distribution of the observed variables ***X*** and the hidden variables ***Z***:

Following the *Bayes’ Theorem*, in order to infer the hidden variables *Z*, the posterior inference is used as follow:

* Prior **–** is the probability of hidden variables before having seen any data. In other words, prior is the probability distribution which expresses one's beliefs about an event before some data is considered.
* Likelihood **–** is the probability of observed variables given hidden variables.
* Posterior – is the probability of hidden variables given the observed variables.
* Normalization constant  **–** is a marginal probability of observed variables, which does not depend on the hidden variables since it contains integral over all possible set of hidden variables. It is called normalization constant, because it makes sure that posterior density integrates to one.

For many interesting models, the denominator is computationally intractable, mainly because of integrals. This means that the exact inference of posterior is not possible, so, one possible solution is to perform an approximate posterior inference, which is what Variational Inference (VI) offers. The reason behind the popularity of VI methods is that they let us solve statistical inference problem as an optimization problem and solve it for hidden parameters by maximizing its objective function. The most often used VI method is the **Mean-field Variational Inference** which will be discussed later below. But before this, let’s explore the main idea behind Variational Inference and the forms of statistical models it is applied to.

Diagram

Description automatically generated

[\*\*Blei-PPT\*\*]

The figure[x] above simply illustrates a technique of Variational Inference. Let’s remember that the aim of VI is to approximate the true posterior distribution . To start with, one needs to posit a variational family of distribution over the hidden parameters/variables. This variational family is represented as an ellipse area in the figure [x]. As seen, it is also parametrized by variational parameters Next, the goal is to find within this family of distributions, such that the corresponding approximate posterior distrubution is closest to the true posterior distribution . Note that this closeness is measured by ***Kullback-Leibler divergence (KL-divergence)***. Idea is to start at some initial set of variational parameters , and then optimize them, i.e. minimize KL divergence [15] to find the point where is closest to .

## KL–divergence Derivation

where ***L***above is called **the variational lower bound or evidence lower bound (ELBO)**. We reformulate the equation above as follow:

Because the *KL* divergence is always non-negative (i.e., we get . This proves that *L* is the lower bound on the log marginal probability of the observations. So, the final goal is to, using a coordinate ascent optimization algorithm (e.g., variational EM [43]), **maximize this lower bound *L* i.e. minimize *KL* divergence with respect to variational parameters** Note that in the formula above is fixed against all variational parameters .

**BOTH derivations return same ELBO, refer and compare to https://mpatacchiola.github.io/blog/2021/01/25/intro-variational-inference.html**

## Jensen’s inequality method

Apart from the derivation mentioned above, there is also an alternative way to arrive at the similar conclusions using the Jensen’s inequality which states for the *concave log function* as follow:

So, the last term in equation above is the **variational lower bound or ELBO**. Note that in the equation belongs to the Shannon entropy:

where . Essentially, it is again shown that *L* is the lower bound of the log marginal probability of the observations and our goal is to maximize this lower bound.

## Mean-field VI

Furthermore, if we consider Mean Field Variational Inference, then the variational distribution over the hidden variables factorizes as follow:

Basically, the mean-field approximation makes a simplifying assumption by partitioning the hidden parameters into independent parts [42]. In other words, this assumption enforces a full independence among all hidden parameters. The reason why this independence is very useful is that, when we use a coordinate ascent optimization algorithm such as variational EM*,* this assumption enables us to compute the update rules for each unknown parameter in isolation by keeping all others fixed [43].

## Mean-field VI in conjugate models

Most importantly, it must be emphasized that there is actually specific form for statistical models in which the coordinate ascent in mean field variational inference yields closed-form updates. It is called exponential family conditionals or conditionally conjugate models. *Fundamentally,* *for a model to be coditionally conjugate, a complete conditional of each parameter must be in the exponential family and be in the same family as its prior [25].* (A complete conditional is the conditional probability of the hidden variable given all the observed variables and other hidden variables).

A generic example for the purpose of understanding conditionally conjugate models is defined as follow (note that it was taken from [46]):

Diagram

Description automatically generated

where are observed variables, are local hidden variables and are global hidden variables. Note that, the main difference between local and global hidden variable is that the *i-*thdata only depends on global on and . In other words, it is not dependent on any other *j*-th local data point. Now, the factorized joint distribution of the model is as follow:

and as usual, our goal is to compute a posterior .

Firstly, for the selected generic model above, following complete conditionals must be in the exponential family:

.

In mathematical terms, an exponential family is expressed as follow:

where is natural parameter, is the sufficient statistics, is the log normalizer and is the base density. In short, If some parameter is in exponential family, then it can be written in the form above.

Secondly, complete conditional must be in the same family as its conjugate prior. This is actually why the exponential family was a crucial requirement in the first place. The reason is that for the exponential family of distributions, the likelihood distribution is a standarized function of the parameter, and therefore we can make conjugate priors by simulating the likelihood's form. Afterwards, when a likelihood and a prior with same exponential form are multipled, the posterior maintains the same form, which was required as a second condition above. Fundamentally, an exponential family of distributions provides a beautiful theory around conjugate priors and corresponding posteriors, and connects closely to variational inference [45]. Note that, examples for conjugate priors and their corresponding posteriors are shown in the next section below.

In fact, there are many popular conditionally conjugate models, some of which are as follow:

* Bayesian mixtures of exponential family models with conjugate priors.
* Any model containing only conjugate pairs and multinomials.
* Mixed-membership models of exponential families (e.g., LDA).

Maybe delete????

As seen above, LDA is also considered conditionally conjugate model, and therefore, it provides coordinate updates of mean-field variation inference in closed-form. The reason behind this is that Dirichlet is a conjugate prior for its multinomial distributed likelihood, which implies that the corresponding posterior is also Dirichlet distributed.

## Mean-field VI in non-conjugate models

Write down later:

[[[Above, CTPF is conjugate model 🡪 closed-form solution

Here, CTMP is non-conjugate 🡪 no closed-form solution 🡪 develop a co-ordinate ascent algorithm that can fit our non-conjugate model]]]

## Conjugate Priors and Corresponding Posteriors

The main idea is that given a likelihood distribution, one needs to select a family of prior distributions such that computed posterior distribution is also included in this family. *By this way, chosen conjugate prior enables us to estimate the posterior distribution just by updating the parameters of the prior distribution.*

Exponential family of distributions are the best example for this. The Gaussian, beta, binomial, Dirichlet, multinomial, gamma, Poisson, exponential, geometric, categorical, chi-sequared, log-normal are all in exponential family. Some pairs of conjugate distributions from exponential family are shown below in detail.

### Multinomial distribution and Dirichlet priors

Remember that the multinomial distribution is the probability distribution where outcomes from experiments are discrete and they involve two or more variables. It is also considered as a multivariate generalization of binomial distribution which involves only two outcomes. Mathematically, it is defined as follow:

where, indicates the number of times outcome *i* occurs out of *n* trials, while signifies the probability that outcome *i* occurs.

Now, let’s remember the Dirichlet distribution which is a continuous multivariate probability distribution. It is also considered as a multivariate generalization of beta distribution and defined as follow:

where is a *k*-vector with components is a *k*-dimensional random variable which is in -simplex, therefore Additionally, denotes the gamma function [37], where

According to conjugate Bayesian analysis, the Dirichlet distribution is considered as a conjugate prior for the multinomial distribution. Therefore, when we multiply the likelihood expressed in multinomial form with the prior expressed in dirichlet form, we get the posterior distribution as follow:

which we can confirm that it has the form of Dirichlet distribution. So, as shown above, we estimate the posterior distribution just by updating the parameters of the prior distribution.

**add that lda is this kind of, therefore its conjugacy helps closed form solutions in vi.**

*links:*

[*https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf*](https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf)

[*http://www.inf.ed.ac.uk/teaching/courses/mlpr/assignments/multinomial.pdf*](http://www.inf.ed.ac.uk/teaching/courses/mlpr/assignments/multinomial.pdf)

[*http://www.cs.columbia.edu/~jebara/4771/tutorials/lecture12.pdf*](http://www.cs.columbia.edu/~jebara/4771/tutorials/lecture12.pdf)

*http://gregorygundersen.com/blog/2019/03/16/conjugacy/*

### Poisson distribution and gamma priors

Let’s now take into account the Poisson distribution from discrete exponential family distributions:

where conjugate prior to this Poisson likelihood must also have the form of Poisson distribution:

This conjugate prior can be easily expressed as *gamma distribution*:

where

denotes the gamma function above. Now, prior-to-posterior update is as follow:

where we can confirm that it has the form of *gamma distribution* which was our the intension from the beginning. Essentially, as seen above, choosing Gamma conjugate prior and multiplying it to Poisson likelihood yielded an the posterior inference which also has Gamma distribution, and therefore we can estimate the posterior distribution just by updating the parameters of the prior distribution, while successfully ignoring the intractable marginal probability in denominator. In other words, if are all identically independently distributed, then conjugate prior for is Gamma(,) and the respective posterior, which is proportional to likelihood multiplied by prior becomes Gamma(+, ).

**add that ctpf is this kind of, therefore its conjugacy helps closed form solutions in vi.**

*links*

[***https://vioshyvo.github.io/bayesian\_inference/conjugate-distributions.html***](https://vioshyvo.github.io/Bayesian_inference/conjugate-distributions.html)

[***https://web.stanford.edu/class/archive/stats/stats200/stats200.1172/lecture21.pdf***](https://web.stanford.edu/class/archive/stats/stats200/stats200.1172/Lecture21.pdf)

***https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf***

**again note that this conjugacy facilitates mean-field variational inference to have an closed-form updates which mentioned in a previous section.**

# OPE and BOPE

Maximum a posteriori probability (MAP) estimation has a significant impact on doing posterior inference (i.e. estimating hidden parameters) in many probabilistic models. Especially, many interesting MAP problems are continuous, non-convex and intractable. In the field of non-convex optimization, there have been a variety of different techniques such as Frank–Wolfe [29], Natasha2 [30], Stochastic Majorization-Minimization [31], Concave-Convex procedure [32] which aim to solve the MAP problem [27]. However, non-convex optimization is NP-hard, and techniques mentioned above may not provide viable solution for MAP problem, because they disregard its special underlying structure. Therefore, for solving non-convex MAP problems with state-of-the-art convergence rate, we will explore two efficient algorithms **Online Maximum a Posteriori Estimation (OPE)** [27] and its regularized, general and more flexible version **Bernoulli randomness in Online maximum a Posteriori Estimation (BOPE)** [28]. First, we introduce MAP estimation as following task:

(1)

where we denote as hidden variable, D as the observed data and denotes domain. Note that there also have been proposed many algorithms which directly tries to estimate a full posterior distribution mentioned above, i.e., Variational Bayesian Methods (VBM) [33], Collapsed Gibbs Sampling (CGS) [34], Hessian Approximated Markov Chain Monte Carlo (HAMCMC) [35]. However, these methods provided suboptimal solutions along with slow convergence rate. Therefore, we continue by using *Bayes’ Theorem*:

(2)

where we denote as likelihood of *D,* as x’s prior, and as s marginal probability. Using (2), we rewrite (1) as following:

(3)

We will focus on the conditions where MAP problem is continuous and non-convex, hence intractable, i.e., is non-convex over the continuous compact domain [28]. As previously mentioned, MAP problem (3) will be treated as an optimization problem. Therefore, objective function defines the complexity of this optimization problem where and . So, our problem (3) becomes as a non-convex constrained optimization problem as follow:

(4)

So, in the following sections we will discuss OPE and BOPE algorithms for solving the optimization problem shown above.

## OPE for solving

Online Maximum a Posteriori Estimation (OPE) is considered as a type of iterative optimization algorithm, which is the stochastic version of Frank–Wolfe algorithm. The biggest advantage of OPE is that it has provably faster convergence rate of to local maximal point compared to the existing stochastic algorithms for nonconvex problems, where signifies the number of iterations during training of its following algorithm [27]:

**OPE Algorithm**

**Output**: which maximizes the objective function over the compact domain .

Initialize arbitrary in .

1. **for** **do**
2. Pick uniformly from


6. **end for**

As illustrated above, the OPE algorithm solves a linear program at each iteration, i.e. directing the optimization solution to the good vertex in the convex hull of compact input domain. In more detail, what OPE does is to develop a sequence of stochastic functions that approximates to by alternatively selecting an from uniformly randomly at each iteration *t*. As proved in its original paper [27], converges to as .

Despite of fast convergence rate, OPE still has a limitation. As stated in algorithm, either likelihood or prior is being used while we are building an approximation function of However, when dealing with new samples, we can rely on likelihood if we have seen enough data, or rely on prior if there is a lack of data.

## BOPE for solving

In order to overcome the OPE’s limitation mentioned above, new approximation technique to OPE has been proposed as BOPE which retains all theoretical guarantees of OPE’s convergence while being more general and flexible by using Bernoulli distribution and two stochastic bounds [28]. BOPE solves problem (4) by employing Bernoulli distribution with parameter which is supposed to replace the uniform distribution of OPE on likelihood and prior. Furthermore, two stochastic sequences are constructed and they converge to objective function : the lower sequence , the upper sequence . It is worth noting that the Bernoulli parameter determines an impact of likelihood and prior on and . So, during each iteration, using both and stochastic sequences provides further information about , so that we increase our chances of reaching more quickly [28]. Both lower and upper sequences are guaranteed to converge to as BOPE algorithm is described in detail below:

**BOPE Algorithm**

**Input:** Bernoulli parameter

**Output:** which maximizes over the compact domain .

**1. init** Initialize

**return**

It’s important to note that one of the reasons why BOPE outperforms OPE is that we can create variants of BOPE by altering the Bernoulli parameter . In addition to this, another property of BOPE is that in order to prevent overfitting of learning process which is the widespread issue that affects all machine learning techniques, BOPE employs implicit regularization. Specifically, according to original paper [28], Bernoulli randomness operates as a regularizer and BOPE uses an implicit prior that is stochastically vanishing with respect to iterations *T*, and this implicit prior is not same as the prior used in MAP estimation. This implicit regularization is very critical, especially in recommender systems where most of the datasets are considered as *sparse*. Therefore, using BOPE in Collaborative Topic Model for Poisson distributed ratings (CTMP) will help us a lot to estimate the unknown parameters without overfitting, because we will be using sparse datasets such as MovieLens and Netflix.

# Collaborative Topic Model for Poisson distributed ratings

In this section, we describe learning, prediction phases and key properties of CTMP – hybrid, scalable and interpretable probabilistic content-based collaborative filtering model.

## Formalization

Before diving into technical parts, let’s provide some notations:

* *U:* represents the number of users inside the dataset
* *J:* representsthe number of items inside the dasaset
* describes the bag-of-word representation for each item *j* where expresses the frequency of term/word in item *j.*
* represents the vocabulary size of the corpus.
* describes the dataset where is a rating provided by user *u* to item *j,* while is the bag-of-word representation of item *j* as already explained above. represents the ratings given to movies by users. Every rating is expressed as binary 0 or 1. If user *u* liked an item *j,* then . On the contrary, if the user *u* do not know about the item *j* or do not like it, then .
* *K:* represents the number of topics inside corpus.
* describes the topic representation. More precisely, every topic *k* is a distribution over the vocabulary. It is described as and . Note that, lies in the (*k* – 1)-simplex.
* describes the topic proportion of the items. is the vector of the distribution on topics for item *j,* and . Note that, lies in the (*k* – 1)-simplex.

In order to learn the topics , we use the Latent Dirichlet allocation (LDA) and its Expectation-Maximization (EM) approach which was described in the respective section of LDA. Furthermore, by learning the topic proportion of each item we later describe each item and user in the *K*-dimensional space. Note that these learning procedures will be explained in the further sections below.

Now, we present **latent factors** for each user and item in terms of *K*-dimensional vectors and , respectively. As discussed in [[[Related Work Section]]], the reason why we consider rather than as the latent factor for item is that in order to have better recommendation system, we allowed an offset between and which accounts for the user-specific preference on the item content that alone can not capture. Therefore we denote that where is an offset term which has Gaussian distribution. Note that in the formula above represents an *K*-dimensional identity matrix , and is a regularization parameter. So, we have .

Furthermore, as shown below, the ratings and users’ latent factors are modeled by Poisson and Gamma distributions, respectively. To put everything together, the generative process and graphical model of CTMP is as follow:

**CTMP Algorithm**

1. For each user *u,* draw where
2. For each item *j*:
3. Draw topic proportion
4. For the *n-*th word of item *j*:
   1. Draw topic index
   2. Draw word
5. Draw latent factor
6. For each user-item pair (*u, j*), draw

Note that steps 2(a-b) corresponds to LDA.

**CTMP Graphical Model**

Diagram

Description automatically generated

## Learning

Full posterior of latent variables is given as follow:

(1)

The problem with this posterior is that it is intractable, and therefore exact inference is impossible. In order to tackle this problem, we have two methods:

1. Maximum A Posteriori (MAP) for point estimation
2. Bayesian Learning such as MCMC Sampling or Variational Methods for approximate inference

As the prior and posterior distributions of hidden variables and are not conjugate in CTMP model, using Variational Inference Methods in order to infer these hidden variables does not get us closed-form solution. Therefore, we will carry out the point estimates of and using MAP – coordinate ascent algorithm developed by authors of original paper of CTMP [\*\*CTMP\*\*].

Furthermore, in order to facilitate the learning, authors added **a** **new auxiliary variable *y****,* where Poisson( and . Note that we approximate the posterior of and via mean-field variational inference [\*\*5\*\*]. The mean-field variational inference is a type of Variational Bayesian Method which allows to re-write a statistical inference problems as an optimization problem[[\*\*\*insertsomething\*\*]]. Therefore, we can convert the inference problem of CTMP into a full optimization problem where the single objective function which needs to be maximized is as follow:

(2)

As shown in (2), the term of integration and summation over the whole space causes optimization to be intractable. However, Variational method [\*\*5\*\*] also tackles this problem which will be discussed in detail below.

Note that has Poission distribution, and the *K*-dimensional vector follows multinomial distribution: [\*\*6\*\*] So, we get the variational distribution as follow:

(3)

where such that is variational parameter of , and are variational parameters of . Note that . Now we get the **evidence lower bound (*l*)** by applying Jensen’s inequality[[\*\*insertpaperofJensen\*\*]]:

(4)

Note that before learning the hidden parameters, are considered as fixed parameters in the model.

Next, the lower bound *l*( shp, rte, ) is maximized with respect to shp, rte, . According to Appendix A of the original CTMP paper [\*\*CTMP\*\*], the terms are expressed in detail as follow:

(5)

## Learning Parameters

Equation (5) is the optimization problem and as mentioned before we solve it by coordinate ascent algorithm. CTMP algorithm for learning and is demonstrated below in Algorithm 1

Algorithm 1.

**Input:** Observed data *w, r,*Bernoulli parameter

**Output:** Estimates

**1. init** Initialize

**Learning** . In order to find the point estimate of local topic proportion where

(6)

we use Bernoulli randomness for Online Maximum a Posteriori Estimation (BOPE) algorithm [\*\*7\*\*]. Note that in original paper of CTMP [\*\*CTMP\*\*], authors used a simple Online Maximum a Posteriori Estimation (OPE) algorithm and this difference is the most important one between this and original CTMP paper. By using Bernoulli randomness, BOPE has a faster convergence rate, is more general and flexible compared to OPE. Furthermore, “BOPE implicitely employs a prior which plays a regularization role”[\*\*7\*\*]. Comparison of BOPE with respect to OPE has been carried out in BOPE section in detail. Include this in comparison::: such properties are not found in the common approximate posterior inference methods for topic models, such as Gibbs sampling and variational Bayes. [[[or discuss here???]]] Note that both algorithms tries to lead the solution of the optimization to the closed neighbours of the vertices in the convex hull of compact input domain and they have a fast convergence rate of along with proven quality bound [\*\*8\*\*]. Furthermore, as mentioned earlier too, every topic proportion holds and lies in the (*k* – 1)-simplex. BOPE algorithm for learning is described in Algorithm 2.

Algorithm 2

**Input:** Bernoulli parameter

**Output:** which maximizes over the compact domain }

**1. init** Initialize

**Learning** If we know the estimates of other hidden variables, then solving analytically is possible because the objective function regarding the is a *concave* function.

(7)

The partial derivative of function with respect to , i.e. for all *k,* is the estimate of . This is also so called the stationary point of . Because is the quadratic function in terms of we can Vieta’s formula for the analytical derivation of the function’s root as follow:

(8)

where

(9)

**Learning** . We use the mean-field variational inference for approximating the conditional posterior of and as in [\*\*9\*\*]. So, in order to solve for the variational parameters of and which are , we solve for the stationary point of with respect to each variational parameter, while holding the others same. The update expression of variational parameters is given in Table/Figure?? [\*\*x\*\*] below. The detailed derivation of these expressions is described in Appedix B and Appendix C. One of the biggest advantages of CTMP algorithm is that whenever , we get and and therefore, we only have to update over non-zero ratings (. This property of our model diminishes the training time significantly, so the total training time is much lower than of other models such as CTR, especially whenever the rating dataset is highly sparse. Because, during each epoch of training, we only consider the positive ratings for updating the expression of and skip all *zero* ratings.

Variational parameter updates

Note that the function in the update expression of denotes the digamma function [\*\*10\*\*]:

where denotes the gamma function.

(10)

**Learning** . So far, we have provided the update expression of the variables regarding both documents and users such as . Now, we have to do the remaining task which is to solve for First, we express the log likelihood of the whole items corpus *C* as in [\*\*10\*\*]:

(11)

By using Jensen’s inequality, the last term is derived, due to the fact that Next, the lower bound of is maximized with respect to as in [\*\*10\*\*]:

(12)

where Note that the each is separable from each other inside the objective function of . So, we can solve solve each individually. This is carried out by considering the Lagrange function and setting its derivatives to 0 which results in formula of as follow:

(13)

## Prediction

We rank the items in order to generate recommendations for each user *u* based on their predictive score after we have learned all the parameters. Because the ratings in the dataset are discrete Poisson variables, can be the expectation of the rate parameter given the observed data i.e. as in CTPF[\*\*CTPF\*\*]. However, the derivation in CTMP is a bit different because CTMP neither aims to approximate solely by point estimate nor require **conjugacy** between the complete **conditional distributions** for the inference as CTPF does [\*\*CTMP\*\*] :

Note that only is the MAP estimation of complete conditional distribution. Furthermore, is nearly the expectation over the respective variational distributions of ’s:

Note that both and are the estimation of variational parameters that we learned in above section.

## Key properties

1) by modelling ratings as discrete Poisson variables, all zero entries can be discarded in the estimation of user latent factor η (see Algorithm 1). Therefore, the learning of CTMP is particularly efficient with highly sparse datasets.

2) Furthermore, as we will show by the empirical results in Section 4.2, OPE updates also recover sparse solutions for the estimates . Sparsity is an important property of document-specific topic mixture. On one hand, it reflects closely the number of topics in a real-life document, which would express only a few topics in details, rather than small bits of 50 or 100 topics. On another hand, it largely reduces memory for storage of the content representation, allowing efficient computation of other tasks in industrial settings (e.g. near real-time recommendation for products of the same topics/categories in which the

users are interested).

3) Add 3rd contribution of CTMP in Related work

**.**

**.**

## Interpretable user profiles

**.**

**.**

**.**

## Evaluation

**.**

**.**

# Empirical Studies

We have carried out empirical studies on performance of CTMP algorithm on different datasets. All experiments have been coded in Python language.

## 

## Data fetching

In order to carry out an empirical study of CTMP algorithm under real-world recommendation context, we use two different datasets, namely **MovieLens 20M** and **NETFLIX**. Both datasets are fetched from Oracle Database using SQL queries.

### MovieLens 20M

**[take citation from https://files.grouplens.org/datasets/movielens/ml-20m-README.html]**

This dataset describes 5-star rating and free-text tagging activity from https://www.movielens.org which is a movie recommendation service. It is very popular and is considered as stable benchmark dataset. It was created on October 17, 2016 and its raw data consists of 20,000,263 ratings, 465,564 tag applications, 27,278 movies and 138,493 users, according to [47]. There was random selection for users to be included in the dataset and all selected users users rated at least 20 movies. Note that each user is only represented by an id, therefore no other information (e.g. demographic) is taken into account.

Note that, for our empirical study, we don’t use the very original MovieLens 20M dataset mentioned above, instead, we use slightly updated version of it which was carried out by *Michal Kopecký, Ph.D* *(Department of Software Engineering, Faculty of Mathematics and Physics, Charles University).* We fetched this updated version of MovieLens 20M dataset from Oracle database and its most important tables are as following:

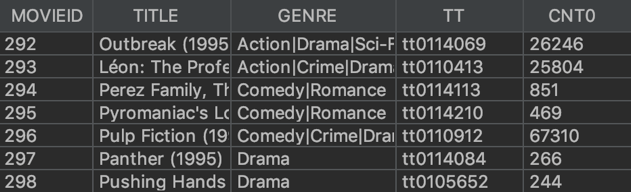
|  |  |  |  |
| --- | --- | --- | --- |
| **Important Tables** | | | |
| **MMOVIES** | **MUSERS** | **MRATINGS** | **IMDB** |

**MUSERS –** is the user table with ids and other features. The necessary column for our studies is *USERID* and snippet from table is as below.

A picture containing text, scoreboard

Description automatically generated

**MMOVIES –** is the table where movies are listed along with their id, title, genre, identifier and other features. Respectively, the necessary columns of the table are *MOVIEID, TITLE, GENRE* and *TT* (it is an identifier which is used to refer more details about movies from IMDB table). Snippet from table is as below.



**MRATINGS –** is the table that provides information about ratings that users gave to movies. The necessary columns are *USERID, MOVIEID, RATING.* Not that ratings are made on 0.5 – 5.0 star scale.

**A picture containing text, scoreboard

Description automatically generated**

**IMBD –** is the table that contains other information about movies such as its plot, awards, imdb rating, and so on. Note that for our studies, this table is only used to fetch *PLOT* of movies given their TT identifier. Plots are given by XML data type in *XML* column.

**A picture containing table

Description automatically generated**

Note that, one of the most important difference between updated version given by Michal Kopecký, Ph.D and original version of MovieLens 20M dataset is that in updated version, movies are also equipted with additional TT identifiers which help us to fetch their plot from IMDB table. In other words, TT identifier is a foreign key or link between MMOVIES and IMDB tables. These plots will later be used for document representation for the models to work with.

### NETFLIX

.

.

.

## Data pre-processing

After fetching the datasets from Oracle database, we continue with data pre-processing steps for transforming the raw data into a useful and efficient format. Sequential steps of data pre-processing are given in the following sections.

### Dataframe creation

So far, we have fetched all the necessary columns of tables for both MovieLens 20M and NETFLIX datasets. Now, by utiliying data wrangling techniques, we make individual dataframes out of these necessary columns (dataframe is two-dimensional, size-mutable and potentially heterogeneous tabular data provided by Python library called **pandas**). Created dataframes are as follow:

|  |  |  |
| --- | --- | --- |
| MovieLens 20M | | |
| df\_user | df\_movie | df\_rating  Text  Description automatically generated with low confidence |

|  |  |  |
| --- | --- | --- |
| NETFLIX | | |
| df\_user  Table  Description automatically generated with low confidence | df\_movie  Text  Description automatically generated | df\_rating  Text  Description automatically generated with low confidence |

Note that constructed dataframes are still raw, meaning that ids in df\_user dataframes are sequentially inconsistent, some movies in df\_movie dataframes do not have plots and ratings in df\_rating dataframes are in 0.5-5.0 scale. So, next section uses these raw datasets for turning them into useful formats.

### Data cleaning and wrangling

First, we start cleaning process by representing ratings in binary form in ratings table. This is done by converting ratings bigger or equal to 4 into 1 and remanings into 0. This way, it is presumed that users like or dislike the given movie. As previously stated, we do this because the CTMP model implies that ratings are Poisson distributed.

Next, we drop movies that lack plot. This means that, when matched with TT identifier from IMDB table, not all movies had plots. We drop duplicates and inconsistent movies using couple of inner join technique between ratings and movies table, and get final tables. Note that dropping movies results in the removal of the ratings connected with these movies. As a result of many data wrangling techniques, we get the results as follow:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataframe** | **Dataset** | **Raw size** | **Final size** |
| df\_user | MovieLens 20M | 138,493 | 138,493 |
| NETFLIX | 479,870 | 479,870 |
| df\_movie | MovieLens 20M | 27,278 | 25,900 |
| NETFLIX | 9,324 | 7,882 |
| df\_rating | MovieLens 20M | 20,000,263 | 19,994,181 |
| NETFLIX | 90,217,939 | 82,725,788 |

### 

### Vocabulary Extraction

From previous steps, we already obtained a clean df\_movies dataframes for both datasets. We now aim to extract a vocabulary for each dataset from the respective corpus of movie plots. The reason why we want to have a corpus vocabulary is that it will be used for numerical representation of each movie in the next section. Before we begin, we sequentially merge each movie plot into a single text – corpus. Extracting vocabulary from this corpus is a little difficult, because it needs a careful investigation of deciding whether a word should be included in vocabulary or filtered out. So, extraction steps are as follow:

* Remove stop words – they are commonly used words (such as “a”, “an”, “are”, “the”, “about”) of a language which do not add much meaning to a sentence. They can be safely ignored by keeping them out of vocabulary. There are many Python libraries that provide a list of stopwords for many languages. We used NLTK Library’s stop words on English language.
* Remove words of length less than 3 – these words can also be ignored. In English language, there rarely is a word of length less than 3 which adds a meaning to a sentence.
* Remove words with numbers or underscores – we don’t want to include these words into vocabulary either.

It is worth mentioning that in the domain of Natural Language Processing, well-known techniques such as Stemming and Lemmatization are utilized for a purpose of text normalization. These techniques help a lot when it comes to filtering unwanted words during vocabulary construction. Stemming technique removes the suffix from a word and therefore reduce it to its root/stem. For example, stem of the word “count” is just “count”. So, stemming the words such as “counts”, “counting” and “counted” will result in “count”. In contrast to stemming, lemmatization goes beyond the word reduction by evaluating a language’s whole lexicon for applying a morphological analysis to words. In other words, lemmatization technique does not simply chop off inflections as stemming does, but instead it relies on a lexical knowledge base for obtaining the correct base forms of words. For example, given the word “mice”, “ran” lemmatization convert them into “mouse” and “run”, respectively. Essentially, by using stemming and lemmatization techniques, one can construct more concise and accurate vocabulary. However, we don’t use these techniques during our vocabulary construction. The reason behind this is that, according to recent works, stemming is claimed to reduce the model fit and negligibly affect topic coherence [48]. Utility of lemmatization on topic models is also vague and rather needs further investigation. Therefore, we avoid stemming and lemmatization of the corpus as a data pre-processing step.

**Snippet**

A picture containing table

Description automatically generated

The figure above is a snippet from resulting vocabulary of MovieLens 20M dataset. We observe that the word “accident” has many forms with different suffixes. This is because we did not use stemming and lemmatization. We believe that letting vocabulary be rich like this will later facilitate model fit into corpus.

After all, the MovieLens 20M vocabulary contains 59,110 words, while the NETFLIX dataset vocabulary has 34,177 words.

### Document Representation

After extracting vocabulary from corpus, we use it for numerical representation of each movie/plot. Reason for this is that documents should be in a such form that models (e.g., LDA, CTMP) can work with. Therefore, one way is to succinctly represent each document as a sparse vector of word counts. It is as follow:

[M] [word\_1]:[count] [word\_2]:[count] ... [word\_N]:[count]

where [M] denotes the number of unique words in the document/plot. Furthermore, [word\_i] is an integer and it denotes an index of the word inside vocabulary and [count] associated with this word denotes how many times the word appears in the movie plot. Note that if any word of document does not appear on corpus vocabulary, then we disregard it in document representation.

**Snippet**

Text

Description automatically generated

***It is critical to remember that, authors of both original CTMP paper [\*\*CTMP\*\*] and BOPE paper [28] implemented their models using features/tags rather than movie plots. This is one of the most significant distinctions between our work and theirs***. Precision unmatchiness between our work and original CTMP paper is probably because of this. During the evaluation of results, we will compare our model’s output against theirs and determine whether or not using movie plots improved the previous results.

### Memory usage reduction

This technique helps us to reduce the memory usage of dataframes. It iterates through all integer or float columns of given dataframe/table, and modifies the data type to reduce the memory usage.

For instance, let’s assume an integer column given as int64 datatype. The way function decided whether to reduce the memory or not is by first getting minimum and maximum values this column. Next, it checks whether these minimum and maximum values are in the range of int32, int16 or int8 datatypes and if they are, then it will convert the column’s int64 datatype into the respective lower datatype. The same procedure applies to float columns as well. As proof, when we used this technique to reduce the memory storage of df\_rating dataframe of MovieLens 20M dataset, it achieved 58.3% reduction in memory.

## Model Fit

After preparing the corpus vocabulary, converting ratings into binary form, numerically representing movies/plots and using memory reduction technique on all dataframes, we now continue with fitting the CTMP model on these datas for both MovieLens 20M and NETFLIX. Let’s also note that neither df\_user nor df\_movie dataframes are already needed in model fitting phase, because df\_users just contains sequentially consistent user ids which can be replaced by integers during training, and df\_movie is already used for vocabulary extraction and numerical document representation.

We use **stratified 5-fold cross-validation** of ratings for training the model. Generally, k-fold cross-validation (CV) is widely used technique that assess the efficacy of machine learning models since it produces a **less biased** estimate of their effectiveness. The parameter kindicates the number of folds that a given dataset is to split into. Pseudo-algorithm for model fitting with k-fold CV is as follow:

For each unique fold:

* Take this fold as test data
* Take the remaining folds as training data
* Fit a model on the training data and evaluate its performance on the test data
* Keep the evaluation score and continue with other loops

At the end of entire train-test circle, we average the evaluation scores kept for each fold to obtain a final evaluation score.

For our case, we utilize stratified K-fold cross-validation, which has the same purpose as standard k-fold cross-validation but produces stratified folds which are made by preserving the percentage of samples for each class. This way, both train and test folds will contain an information about each user, ensuring that the model will be trained and tested with all users. An example of stratified 5-fold cross validation with small dataset is shown below:

Table

Description automatically generated

### Setting Parameters

Thus far, we have prepared all of the necessary input data, including train folds, test folds, docs.txt, and vocabulary.txt, in order to begin training the model. But before we begin, one final task is to create a settings.txt file in which we may change the model’s parameters on a per-run basis. Example snippet settings files for both datasets are shown below;

Text

Description automatically generated Text

Description automatically generated

where,

- ***num\_docs* (1 : *J*)** is the number of documents in the corpus,

- ***num\_terms* (1: *V*)** is the number of words inside respective vocabulary,

- ***num\_topics* (*K*)** is the number of topics we want to discover from corpus,

- ***user\_size* (1: *U*)** is the number of users in dataset,

- ***tops*** is the number of top words to extact from each topic for eyeballing/analyzing the quality of topics learnt by the model,

- ***lamb*** **()** is the offset precision for documents,

- ***e, f*** are both the Gamma prior,

- ***alpha* (** is the Dirichlet prior.

- ***iter\_infer*** is the number of inference steps during an estimation of document proportions within BOPE,

- ***iter\_train*** is the number ofepochs to train the model,

- ***p*** is the Bernoulli parameter for BOPE.

It must be noted that the only parameters which are entirely dependent on the dataset are ***num\_docs, num\_terms***, and ***user\_size***. Thus, they are fixed for MovieLens 20M and NETFLIX.

Other parameters are modifiable, and generally, their main domain of interest is as follow:

|  |  |
| --- | --- |
| **Parameter** | **Domain** |
| num\_topics | {50, 100} |
| tops | {10} |
| lamb |  |
| e, f | {0.3} |
| alpha | {1, 0.1, 0.01} |
| iter\_infer | {100} |
| iter\_train | {50} |
| p | {0.7, 0.9} |

It’s worth noting that we picked Bernoulli parameter rather than 0.5 since this parameter replaces the uniform distribution of OPE on likelihood and prior, and adjusting it to 0.5 would result in a BOPE algorithm that approximates OPE which is not our goal.

Finally, as shown on table above [\*\*\*], in order to fit all hidden parameters, it was enough to set training iterations/epochs of CTMP algorithm to 50 and and inference iterations of BOPE inside CTMP to 100 (i.e., iter\_train=50, iter\_infer=100).

### Challenges during Python implementation

1. **Initialization;**

In Python, first challenge during the implementation of CTMP algorithm was how to handle an initialization of parameters before starting algorithm. Initializations for some parameters were mentioned in the original paper[\*\*CTMP\*\*], but not for others:

|  |  |
| --- | --- |
| **Mentioned** | **Not mentioned** |
| *e, f,* , , *K, ,* | , shp, rte |

* Initializations of *e, f,* *(lamb)*, *(alpha)*, *K(num\_topics)* havealready been discussed in previous chapters [\*\*chapter xx\*\*], and they are similar to those mentioned on [\*\*BOPE\*\*], [\*\*CTMP\*\*].
* According to original paper [\*\*CTMP\*\*], initializations of and had to be made by their respective estimates from LDA. Apparently, **this** **didn’t work** in our case, because when and were initialized as LDA-learnt parameters for CTMP algorithm, It was unable to continue updating and fitting these parameters along with others beyond the initial iterations. Therefore, we decided to initialize both and randomly in (k1)-simplex. This resulted in appropriate update and fit of parameters across the iterations.
* We decided to initialize each equal to , and let CTMP algorithm learn the offset for each document across the iterations.
* It has been decided to initialize the user's variational parameters shp and rte to *e* and *f*, respectively, as follow:
* Finally, we need to initializate rating’s variational parameter which is the most difficult of all because it is a three-dimensional matrix with dimensions equal to user\_size, num\_docs and num\_topics, respectively. Unfortunately, **due to memory limits**, we could not directly generate a 3D matrix with this shape in Python. As a result, we discovered a workaround by dividing the entire 3D matrix into smaller 3D matrices that can be constructed within the memory limits. Then, we put all of these smaller 3D matrices sequentially into a list. For example, if we consider running CTMP on MovieLens 20M dataset with *K*=100 topics, then shape of will be as follow:

where, user\_size = 138493 and num\_docs = 25900 for MovieLens 20M. Thus, in order to be able to store , we do the following steps;

* Create 138 empty 3D matrices with shape = [1000, 25900, 100]
* Create 1 remainder empty 3D matrix with shape = [493, 25900, 100]
* Put all created 139 matrices sequentially into a list.

As seen above, with the help of the steps mentioned above, we now are able to represent the whole initial matrix with small chunks of matrices.

Furthermore, note that, in order to reduce the memory storage, all matrices mentioned above are created with the data type being **float32**. Because we have successfully tested that when two separate models are run using float64 and float32, the difference in results is insignificant.

1. Computational Speed

@njit

mostly used Numpy + its vectorization

### Running on Google Cloud

After we have prepared all necessary input data and set input model parameters, we will run the CTMP models on **Google Cloud’s Compute Engine** which is compute service that lets us create and run virtual machines on Google’s infrastructure. Google Cloud offers a wide range of compute engine variants with a variety of configuration options. For our model, we require a large amount of RAM memory because the dataset are large and we perform 5-fold cross-validation in paralell. Therefore, the most relevant machine type for our work is as follow:

|  |  |
| --- | --- |
| **Google Cloud Virtual Machine** | |
| Machine family | GENERAL-PURPOSE |
| Series | N1 |
| Machine type | n1-highmem-16 |
| CPU | 16x Intel® optimized with Intel® MKL and CUDA 11.0 |
| RAM memory | 104 GB |
| Boot Disk | Standard Persistent Disk 100GB |

Remember that, as mentioned in the Model Fit section, we have set number of iterations to 50 and number of BOPE inference to 100.

As 5-fold cross validation fits separate models on different folds, we decided to run these separate models on parallel in Virtual Machine. This way, we significantly minimize time consumption by fivefold and get the most out of CPUs.

Generally, time per iteration of the model depends on which dataset and parameters have been selected and effectiveness of the code written in Python. We have optimized our code by utilizing many techniques **(refer to section …)** in order to reduce the time complexity and consumption of the CTMP model. Finally, we arrived at the model that performed as follows:

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Average time per iteration** | **Total average time** |
| MovieLens 20M | ~ 200 seconds | ~ xx hours |
| NETFLIX | ~ (300 max) seconds | ~ xx hours |

Text

Description automatically generated

netflix

25/175 -- movielens

Reason why NETFLIX takes more time is its # of users.

## Model Evaluation

After running and fitting the model on both datasets with various parameters each time, we now continue with evaluating the performance of these CTMP models. Performance evaluation will be divided into the following parts: interpretability of learnt topics, predictive performance , sparsity in topic proportion estimates and sensitivity to hyperparameters.

### Interpretability of topics

MovieLens 20M

Lets eyeball the top10 words of first 10 topics learnt by CTMP on MovieLens 20M

with parameters {alpha=1, lamb=1, K=100)

|  |  |
| --- | --- |
| Topic 1 | man, father, young, wife, mother, daughter, son, brother, new, old |
| Topic 2 | new, life, one, love, finds, girl, school, two, job, friend |
| Topic 3 | get, new, johnny, one, house, wife, town, police, killer, back |
| Topic 4 | one, two, begins, friend, girl, mother, man, young, life, friends |
| Topic 5 | new, life, world, story, young, love, find, two, help, husband |
| Topic 6 | life, one, new, father, get, woman, friend, two, mother, family |
| Topic 7 | film, documentary, world, story, life, one, new, interviews, footage, history |
| Topic 8 | life, family, town, one, love, home, old, new, war, man |
| Topic 9 | young, family, one, husband, police, mother, father, wife, finds, woman |
| Topic 10 | town, love, new, family, young, father, man, old, two, find |
| Change to 1.2.3.4.5 … 95,96,97,98,99,100. |  |

NETFLIX

Lets eyeball the top10 words of first 10 topics learnt by CTMP on NETFLIX

with parameters {alpha=1, lamb=1, K=100)

|  |  |
| --- | --- |
| Topic 1 | wife life one team man goes story esmeralda get old |
| Topic 2 | year town takes life friends family michael one must son |
| Topic 3 | life father mother new young family scott son finds friend |
| Topic 4 | find killer new town victims years become one back help |
| Topic 5 | new one time children island get years girl find long |
| Topic 6 | man new one bill father york wife son two love |
| Topic 7 | life love two time get one new find jonathan george |
| Topic 8 | new agent drug back fbi cia help american two friends |
| Topic 9 | find new back one love also two family life get |
| Topic 10 | life man take wants new young back money help school |

### Predictive performance

Model's predictive performance is measured by its ability to recommend in-items and cold-items. Note that in-items are those containing information from user ratings while cold-items do not possess such information. Thus, recommending items which are all in-items is referred as ***in-matrix prediction,*** wheares recommending both in-items and cold-items is called ***out-of-matrix prediction***. Both prediction types are evaluated by recall and precision for all ….

Talk about formulas of them and how in-matrix prediction and out-of-matrix prediction evaluated for our case.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

### Sparsity

Graphical user interface, chart, application, treemap chart

Description automatically generated

### Sensitivity to hyperparameters

.

.

.

## Transfer Learning between MovieLens 20M and NETFLIX

**Movie**: The Naked City (1948)

**Original Plot**: “Amid a semi-documentary portrait of New York and its people, Jean Dexter, an attractive blonde model, is murdered in her apartment. Homicide detectives Dan Muldoon and Jimmy Halloran investigate. Suspicion falls on various shifty characters who all prove to have some connection with a string of apartment burglaries. Then a burglar is found dead who once had an elusive partner named Willie. The climax is a very rapid manhunt sequence. Filmed entirely on location in New York City.”

Note that, on both MovieLens 20M and NETFLIX datasets, this movie is included. We begin by separately training a CTMP model on both datasets. Modifiable model parameters of settings files are as follow:

|  |  |
| --- | --- |
| CTMP on MovieLens 20M | CTMP on NETFLIX |
| **num\_topics: 100** | **num\_topics: 50** |
| tops: 10 | tops: 10 |
| lamb: 1 | lamb: 1 |
| e: 0.3 | e: 0.3 |
| f: 0.3 | f: 0.3 |
| alpha: 1 | alpha: 1 |
| iter\_infer: 100 | iter\_infer: 100 |
| iter\_train: 50 | iter\_train: 50 |

Top-10 for each topic for both MovieLens and NETFLIX

(i.e. topn\_outpu.txt)

Put screenshots from both top10 words, and discuess that they make sense. Then move on

After 50 iterations of training, both models are fit to the respective dataset. Let’s take a look at top10 words of the topics learnt from MovieLens 20M and NETFLIX datasets.

Following that, we compare the topic proportions estimated by both models for “The Naked City (1948)” movie. Estimated topic proportions are as follow:

Chart, diagram, histogram

Description automatically generated

As illustrated in the figure above, CTMP model trained on MovieLens 20M estimates that “The Naked City (1948)” movie has of 92.9% of content related to topic 77 and 7% of content related to topic 7 (the remaining 0.1% comes from other topics, which we disregard because it is insignificant). Now, let's look at what those significant topics are about by looking at the top-10 words of each topic learnt by the model:

* **Topic 77 (92.9%)** – killer, police, detective, serial, one, murder, murders, case, young, two.
* **Topic 7 (7%)** – film, documentary, world, one, life, new, interviews, story, history, footage.

We can see that model estimates that the movie will be mostly centered on the keywords murder, detective, police, killer, and so on. Moreover, it also provides little topic proportion about documentary, cinema, footage, etc. When these results are compared to the movie's original plot, it is clear that the model accurately estimates the topics proportions of the movie. Topic 7 being assigned a 7% share is also a good sign, because related sentences to this topic were only mentioned in the first and last sentences of the original plot, and they were not particularly relevant to the broader narrative.

On the other hand, a CTMP model trained on the NETFLIX dataset suggests that "The Naked City (1948)” movie contains 96.9% conten associated to topic 11 and 3% content belonging to topic 48. (the remaining 0.1 percent comes from other topics, which we disregard because it is insignificant). Let's take a look at what those important topics are about by checking the top-10 words of each topic learnth by the model:

* **Topic 11 (96.9%)** – police, murder, killer, detective, new, man, life, one, crime, father.
* **Topic 48 (3%)** – jimmy, new, one, film, life, father, alison, time, george, make.

It is apparent that this model, too, accurately approximates the movie's content by giving the most weight to Topic 11, which is concentrated on the phrases police, murder, killer, and so on. 3% being assigned to Topic 48 is again a good sign that this model, too, could associate another negligible topic regarding to the first and last sentences of original plot.

Finally, when we compare the outputs of the two models, we can see that they both learned the topic proportions ***accurately and similarly***. They both gave extremely high proportion to the topic, which covers the majority of the movie’s content, and very little weight to the topic, which is not the primary motivation for the movie's plot.

Alggorithmic changes:

* Change circles of user-doc. Why? Ans: because of initializations
* Phi normalization

**put (num\_iter, num\_infer) somewhere**

* + sum(mu) =! 1 but it worked, comment on this.
  + CLOB (Large Character Strings) while fetching plots
* Small type during mu update, why we decided to change it, and the challenges it produced when implemented wrongly

|  |  |
| --- | --- |
| **For both MovieLens 20M and NFLX** | |
| **Parameters fixed** | **Parameters changed** |
| p=0.7; k=50; alpha=1 | lamb (1, 10, 100) |
| p=0.7; k=50; alpha=0.1 | lamb (1, 10, 100) |
| p=0.7; k=50; alpha=0.01 | lamb (1, 10, 100) |
| p=0.7; k=100; alpha=1 | lamb (1, 10, 100) |
| p=0.7; k=100; alpha=0.1 | lamb (1, 10, 100) |
| p=0.7; k=100; alpha=0.01 | lamb (1, 10, 100) |
| p=0.9; k=50; alpha=1 | lamb (1, 10, 100) |
| p=0.9; k=50; alpha=0.1 | lamb (1, 10, 100) |
| p=0.9; k=50; alpha=0.01 | lamb (1, 10, 100) |
| p=0.9; k=100; alpha=1 | lamb (1, 10, 100) |
| p=0.9; k=100; alpha=0.1 | lamb (1, 10, 100) |
| p=0.9; k=100; alpha=0.01 | lamb (1, 10, 100) |

**NFLX**

Parameters fixed Parameters changed

p=0.7, k=50, alpha=1 lamb (1, 10, 100) 96.3344%, 91.3271%, **61.0266**%

p=0.7, k=50, alpha=0.1 lamb (1, 10, 100) 97.9888%, 97.8130%, 91.3925%

p=0.7, k=50, alpha=0.01 lamb (1, 10, 100) 97.9878%, 97.8059%, 91.3563%

p=0.7, k=100, alpha=1 lamb (1, 10, 100) 98.3988%, 95.8885%, **80.2701**%

p=0.7, k=100, alpha=0.1 lamb (1, 10, 100) 98.9981%, 98.9496%, 95.9888%

p=0.7, k=100, alpha=0.01 lamb (1, 10, 100) 98.9975%, 98.9480%, 96.1034%

p=0.9, k=50, alpha=1 lamb (1, 10, 100) 95.8686%, 89.9838%, **56.2083**%

p=0.9, k=50, alpha=0.1 lamb (1, 10, 100) 97.9947%, 97.8990%, 91.8691%,

p=0.9, k=50, alpha=0.01 lamb (1, 10, 100) 97.9952%, 97.9076%, 92.3017%

p=0.9, k=100, alpha=1 lamb (1, 10, 100) 98.2607%, 95.1714%, **77.6233**%

p=0.9, k=100, alpha=0.1 lamb (1, 10, 100) 98.9992%, 98.9796%, 96.4929%

p=0.9, k=100, alpha=0.01 lamb (1, 10, 100) 98.9995%, 98.9805%, 96.5180%

MovieLens (float32 used . Delete it when NFLX is used)

p=0.7, k=50, alpha=1 lamb (1, 10, 100) 93.9395%, 90.2419%, **66.3825%,**

p=0.7, k=50, alpha=0.1 lamb (1, 10, 100) 97.9460%, 97.4862%, 92.3768%

p=0.7, k=50, alpha=0.01 lamb (1, 10, 100) 97.9464%, 97.4624%, 92.3967%

p=0.7, k=100, alpha=1 lamb (1, 10, 100) 97.1312%, 94.9307%, **82.0474%**

p=0.7, k=100, alpha=0.1 lamb (1, 10, 100) 98.9878%, 98.8174%, 96.2750%

p=0.7, k=100, alpha=0.01 lamb (1, 10, 100) 98.9873%, 98.8286%, 96.3212%

p=0.9, k=50, alpha=1 lamb (1, 10, 100) 93.7492%, 89.2140%, **63.4297%**

p=0.9, k=50, alpha=0.1 lamb (1, 10, 100) 97.9834%, 97.6863%, 92.7979%

p=0.9, k=50, alpha=0.01 lamb (1, 10, 100) 97.9792%, 97.6967%, 92.8476%

p=0.9, k=100, alpha=1 lamb (1, 10, 100) 96.9735%, 94.4133%, **80.2497%**

p=0.9, k=100, alpha=0.1 lamb (1, 10, 100) 98.9960%, 98.9113%, 96.6508%

p=0.9, k=100, alpha=0.01 lamb (1, 10, 100) 98.9960%, 98.9095%, 96.6926%

WE RUN ON GOOGLE CLOUD. Things to mention

* + - We run each fold in separate terminal to get maximum cpu usage
    - Which CPU, Memory specs used for to be able to run 5 terminal on total.
    - How much time it took to run
    - ***Which optimizations have been carried out to speed up the results***

**Transfer-learning results: k=100 is bad, try k=50. (NFLX)**

**~~Explain how data was retreived from Oracle/~~**

**~~Mention movie plots used as features. Cleaning of plots + vocab construction out of corpus plots~~**

**~~Add description of datasets.~~**

**NFLX is good with k=50, Movielense with k=100 (when we print top10 from each topic, and check words)**

**~~Mention 5-fold cross validation~~**

**Estimate confidence interval with 10% subsampling.**

**~~Python challenges faced: how phi’s are stored 3D matrix.~~**

**Higher lamb 3-30%, but high lamb with high alpha less%. Test it (sparsity)**

* Two most common tasks in recommender systems are predicting the score the user might give for a product (the rating prediction task), and recommending a ranked list of most relevant items (the top-N recommendation task)
* [[Take definitions of “in-matrix” and “out-of-matrix’ from CTR paper]]
* Mention that we run topic modelling LDA on **plots** not other features.
* Discussion on my experiments while implementation on Python
* Put top 20 words for each topic from pure or CTMPs LDA
* Train Time + Machine Specs of Google Cloud (maybe make table on time spent for each hyperparameters)
* Where used “document”, where used “item”. Decide which one.
* Sparsity plots
* ~~Why we choose p {0.7, 0.9} and not 0.5? because 0.5 indicates uniform distribution, which was intention behind OPE.~~
* Include somewhere change of doc and user cycle compared to original paper [include in theory, or in experimental part]
* Compare p=0.7 with p=0.9, is higher better? [in BOPE its mentioned so]
* Python libraries used mostly numpy, pandas [+ numba jit]

Mention that 90% of movies from NFLX is also contained in MovieLens 20M. So, compare LDA results from both datasets, explore the common topics they learn.

**During Model Evaluation, put “Interpretable user profiles” somewhere.**

Refer Pavel thesis http://cbir.utia.cas.cz/homepage/projects/phd\_thesis/vacha\_phd.pdf

**Difference on recall&precision between our and original work may be because we use plots instead of tags/features.**

**Add More**

Put gamma distribution explanation with formula, somewhere

ADD connectivity between sections [e.g., between Probab. Models for RS and LDA]

# References

[1] <https://en.wikipedia.org/wiki/Topic_model>

[2] <https://www.tidytextmining.com/topicmodeling.html>

[3] <https://dl.acm.org/doi/pdf/10.5555/944919.944937>

[4] <https://stephentu.github.io/writeups/dirichlet-conjugate-prior.pdf>

[5] <https://people.eecs.berkeley.edu/~jordan/papers/variational-intro.pdf>

[6] Univariate Discrete Distributions, vol. 444 [[[[take from CTMP]]]]

[7] <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9138369>

[8] <https://arxiv.org/pdf/1512.03308.pdf>

[9] <https://www.scopus.com/record/display.uri?eid=2-s2.0-84982318199&origin=inward&txGid=117c9f14425c2abc105f8cd8ac63fa5f>

[10] <https://www.sciencedirect.com/topics/mathematics/digamma-function>

[11] Fully Sparse Topic Model (FSTM)

[12] <http://www.diva-portal.org/smash/get/diva2:1219240/FULLTEXT01.pdf>

[13] [https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf](https://datajobs.com/data-science-repo/Recommender-Systems-%5bNetflix%5d.pdf)

[14] <https://en.wikipedia.org/wiki/Collaborative_filtering>

[15] <https://link.springer.com/referenceworkentry/10.1007%2F978-3-642-04898-2_327>

[16] https://blog.evjang.com/2016/08/variational-bayes.html

[17] <https://dl.acm.org/doi/pdf/10.5555/1378245.1378272>

[18] <https://www.cs.toronto.edu/~amnih/papers/bpmf.pdf>

[19] <https://pubmed.ncbi.nlm.nih.gov/24467759/>

[20] <https://papers.nips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf>

[21] <https://www-users.cs.umn.edu/~baner029/papers/10/gpmf.pdf>

[22] <https://arxiv.org/pdf/1301.6705.pdf>

[23]<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.564.5299&rep=rep1&type=df>

[24] CTR

[25] CTPF

[26] LDA - <https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>

[27] OPE - https://arxiv.org/abs/1512.03308

[28] BOPE - <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9138369>

[29] Frank–Wolfe – take from BOPE

[30] Natasha2 – take from BOPE

[31] Stochastic Majorization-Minimization – take from BOPE

[32] Concave-Convex procedure – taken from BOPE

[33] Variational Bayesian – taken from BOPE

[34] CGS – taken from BOPE

[35] HAMCMC – taken from BOPE

[36] Berkeley Conjugate Priors - <https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf>

[37] Gamma f - <https://en.wikipedia.org/wiki/Gamma_function>

[38] Variational Inference: A Review for Statisticians https://arxiv.org/pdf/1601.00670.pdf

[39] Variational Inference in Nonconjugate Models https://arxiv.org/pdf/1209.4360.pdf

[40] Closed-form solution <https://www.sciencedirect.com/topics/engineering/closed-form-solution>

[41] Closed-form expression <https://en.wikipedia.org/wiki/Closed-form_expression>

[42] mean-field VI https://www.cs.cmu.edu/~epxing/Class/10708-17/notes-17/10708-scribe-lecture13.pdf

[43] variational EM <https://chrischoy.github.io/research/Expectation-Maximization-and-Variational-Inference/>

[44] exponential family conditionals <https://www.cs.cmu.edu/~epxing/Class/10708-17/notes-17/10708-scribe-lecture13.pdf>

[45] exp fam connects to VI <https://people.eecs.berkeley.edu/~wainwrig/Papers/WaiJor08_FTML.pdf>

[46] Blei-PPT-longer <https://www.eurandom.tue.nl/wp-content/uploads/2019/05/Blei_lectures.pdf>

[47] MovieLens 20M <https://grouplens.org/datasets/movielens/20m/>

[48] Lemma, Stemming https://arxiv.org/pdf/1608.03995.pdf

https://people.eecs.berkeley.edu/~jordan/papers/variational-intro.pdf -

In particular, they make a link between this lower bound and parameter estimation via the EM algorithm. ALSO ADD variation EM explanation:::

Text, letter

Description automatically generated