# Collaborative Filtering on Keywords Recommendation for Clinical Trial Records

Xiao Zhou

June 2020

**Abstract**

Being able to discover similar clinical studies is critical for therapeutic product development. However, due to the complexity of medical research and terminology, a query may not contain all the necessary keywords, affecting the final search results. Here, a keyword-recommending algorithm using collaborative filtering is developed and examined. The recommended keywords can be used in the downstream document matching algorithm, providing a potentially more accurate way of retrieving similar documents.

## 1 Introduction

Clinical trials, the key steps in developing medical products, are often of high cost and extreme time-consuming. To avoid unnecessary investment and to gain insights on other similar projects, medical researchers need to search for clinical trials similar to their own studies and compare. Bag of words methods such as locality sensitivity hashing (LSH)[1] and Okapi BM25 [2] are examples of the well-established algorithm to retrieve similar documents against a query. Both methods rely on the query to provide keywords. However, in reality, the end-user may not be able to put all the relevant keywords in the query, due to the complexity of medical research and terminology.

In this study, I developed and examined a keyword-recommending algorithm inspired by the collaborative filtering algorithm which has been successfully used in the Netflix contest [3]. In detail, when a user provides a query, either a short sentence or a block body of text, new keywords will be recommended based on (1) the keywords extracted from the query and (2) the whole corpus the model has been trained on. The new keywords will be added to the query keywords and will be used by the downstream document retrieving algorithm. This study focuses on the recommending algorithm, not the document retrieving step.

## 2 Data Source

### 2.1 Clinical Trial Records

The clinical trial database, ClinicalTrials.gov (https://clinicaltrials.gov/), hosted by the National Library of Medicine provides the most complete historical clinical trial records conducted in the United States and five other countries. This database will be used as the data source. A clinical trial records consist of several sections, and the keywords will be extracted from ”brief_title”, ”official_title”, ”brief_summary”, ”detailed_description”, ”condition”, and ”eligibility”. These sections are more relevant to the content of a clinical trial.

### 2.2 Medical Subject Headings

A great challenge is to extract the keywords from medical documents. The challenge not only lies in how to filter out irrelevant words, but also how to deal with the synonyms of medical terms. For example, the word ”a” is a typical stop word, but ”A” is often used in gene names to indicate this gene is the first member of a gene family. In a special case, ”Protein A” is the name of a cell wall protein from the bacteria Staphylococcus aureus which is widely used in biological research. Also very often, a gene (protein, chemical, or disease) name has many synonyms or abbreviations. For example, the famous breast cancer gene BRCA1 is also called BRCC1, FANCS, PPP1R53, and RNF53. Hence, generic extraction methods used on general text will have a detrimental effect on the quality of the extracted keywords.

To tackle this issue, the Medical Subject Headings (MeSH) (meshb.nlm.nih.gov) produced and maintained by the National Library of Medicine were used. The MeSH thesaurus is a controlled and hierarchically-organized vocabulary for medical subjects. In this study, the MeSH terms were transformed into a Trie data structure which was then used to scan text for keywords. Since this is not the main focus of this paper, the details are omitted here. In brief, a sequence of words forming a MeSH term will be extracted, and the MeSH terms pointing to the same concept bears the same index.

## 3 Modules

The implementation of this study has three modules. First is the data engineering module which extracts keywords from text, transforms keywords to indices, and loads the indices into a utility matrix which will be used by the second machine learning module. The machine learning module implements the algorithm developed in this study, performs train, validation, test, and prediction tasks. The final module is the data visualization module which provides a web UI for user's query and result display. The last module is not the main focus of this project and is not included in this report.

## 4 The Collaborative Filtering Model

### 4.1 TF-IDF Utility Matrix

The TF-IDF of MeSH terms in each clinical trial record (hereby referred to as "document") is calculated based on the following formula and organized into a utility matrix $U$:

$$U_{ij} = \underbrace{\log_{10}(1 + \text{count of MeSH } i \text{ in document } j)}_{TF} \cdot \underbrace{\log_{10}\left(\frac{\text{total documents}}{\text{count of documents having MeSH } i}\right)}_{IDF}$$

Denote the dimension of $U$ as $m \times n$. That is the matrix has $m$ MeSH terms and $n$ documents. Hence, each row can be viewed as a vectorized MeSH term and each column vector can be viewed as a vectorized document, as shown by the following structure of U:

$$\begin{array}{c} 1 \\ \vdots \\ i \\ \vdots \\ m \end{array} \quad \underbrace{\begin{array}{cccc} \square & \square & \cdots & \square \\ \vdots & \ddots & \cdots & \square \\ \square & \square & \cdots & \square \\ \vdots & \ddots & \cdots & \square \\ \square & \square & \cdots & \square \end{array}}_{\text{total n}}$$

Notice that $U_{ij} \geq 0$, and $U_{ij} = 0$ only if and only if MeSH $i$ is not present in document $j$. After U is constructed using all the clinical trial records, it is divided randomly (randomly select column vectors) to three matrices: the training matrix $R$, the validation matrix $X_{dev}$, and the test matrix $X_{test}$, with the ratio of the number of column vectors as $8:1:1$.

### 4.2 Naive Model

In the training matrix $R$, $R_{ij}$ is the TF-IDF score of the MeSH term $i$ in document $j$. The hypothesis is that for MeSH terms $i$ in document $j$, its TF-IDF can be estimated by a linear combination of the TF-IDF of $i$'s neighboring MeSH terms in document $j$. That is

$$R_{ij} = \underbrace{\sum_{f=1}^{m} W_{if}\Theta_{if}R_{fj} + \vec{b}_i}_{\hat{R}_{ij}} + \vec{\epsilon}_i$$

where $\Theta$ is the parameter matrix needs to be learned from the data, and $W$ is a neighbor weight matrix measuring the distance between MeSH term $i$ and $f$. This is similar to the neighboring weight used in the Locally Linear Regression. To obtain $W$, the cosine distance between MeSH term $i$ and $f$ is transformed to a value $\in [0, 1]$:

$$W_{if} = e^{\tau[1-cos(R_{i\cdot}, R_{f\cdot})]^k}$$

where $\tau < 0$ and $k > 0$ dictate the width and the top plateau of the curve. The closer the two MeSH items, the higher the value. Because $R_{ij} \geq 0$, $cos(R_{i\cdot}, R_{f\cdot}) \in [0, 1]$, and this is true for all the documents. To remove "self interactions" from the calculation, set $W_{ii} = 0$. $\vec{b}_i$ is the baseline of the $i$-th mesh term, and $\vec{\epsilon}_i$ is the random noise for the $i$-th mesh term.

The loss function can be written as

$$J(\Theta, \vec{b}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{fj} + \vec{b}_i - R_{ij}\right)^2$$

Therefore,

$$\begin{aligned}
\frac{\partial J(\Theta, \vec{b})}{\partial \Theta_{kl}} &= \frac{\partial}{\partial \Theta_{kl}}\frac{1}{2}\sum_{i=1}^{m}\left[\left(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{f1} + \vec{b}_i - R_{i1}\right)^2 + \cdots + \left(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{fn} + \vec{b}_i - R_{in}\right)^2\right] \\
&= \sum_{j=1}^{n}\left(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k - R_{kj}\right) W_{kl}R_{lj} \\
&= W_{kl}\sum_{j=1}^{n}\left(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k - R_{kj}\right) R_{lj}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J(\Theta, \vec{b})}{\partial \vec{b}_k} &= \frac{\partial}{\partial \vec{b}_k}\frac{1}{2}\sum_{i=1}^{m}\left[\left(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{f1} + \vec{b}_i - R_{i1}\right)^2 + \cdots + \left(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{fn} + \vec{b}_i - R_{in}\right)^2\right] \\
&= \sum_{j=1}^{n}\left(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k - R_{kj}\right)
\end{aligned}$$

In matrix form,

$$\begin{aligned}
J(\Theta, \vec{b}) &= \frac{1}{2}\mathbf{1}^T\{[(W \odot \Theta)R + \vec{b} - R]^2\}\mathbf{1} \\
&= \frac{1}{2}\mathbf{1}^T(\hat{R} - R)^2\mathbf{1}
\end{aligned}$$

where $\hat{R} = (W \odot \Theta)R + \vec{b}$, and the square is element-wise square (same below).

$$\begin{aligned}\frac{\partial J(\Theta, \vec{b})}{\partial \Theta} &= W \odot \left[ [(W \odot \Theta)R + \vec{b} - R]R^T \right] \\ &= W \odot \left[ (\hat{R} - R)R^T \right]\end{aligned}$$

$$\begin{aligned}\frac{\partial J(\Theta, \vec{b})}{\partial \vec{b}} &= [(W \odot \Theta)R + \vec{b} - R]\mathbf{1} \\ &= (\hat{R} - R)\mathbf{1}\end{aligned}$$

For gradient descent,

$$\begin{aligned}\Theta &:= \Theta - \alpha W \odot \left[ (\hat{R} - R)R^T \right] \\ \vec{b} &:= \vec{b} - \alpha(\hat{R} - R)\mathbf{1}\end{aligned}$$

For a given document with a column vector $\hat{\vec{r}}$ which represents the TF-IDFs of the MeSH terms , the prediction is

$$\hat{\vec{r}} = W \odot \Theta \vec{r} + \vec{b}$$

## 4.3 ReLu Model with Regularization

One drawback of the naive model is that the predicted TF-IDF values can be negative. Not only does this not fit the TF-IDF definition (which should be $\geq 0$), but also makes the model too rigid (a negative $\hat{R}_{ij}$ will increase the $J(\Theta, \vec{b})$). Therefore, a ReLu function is applied to make the model more flexible. To constrain the magnitude of $\Theta$, its squared Frobenius norm is added, and the loss function can be written as

$$J(\Theta, \vec{b}) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \left( \mathrm{ReLu}(\sum_{f=1}^{m} W_{if} \Theta_{if} R_{fj} + \vec{b}_i) - R_{ij} \right)^2 + \frac{1}{2} \lambda \sum_{i=1}^{m} \sum_{j=1}^{m} \Theta_{ij}^2$$

Therefore,

$$\begin{aligned}\frac{\partial J(\Theta, \vec{b})}{\partial \Theta_{kl}} &= \frac{\partial}{\partial \Theta_{kl}} \frac{1}{2} \sum_{i=1}^{m} \Bigg[ \left( \mathrm{ReLu}(\sum_{f=1}^{m} W_{if} \Theta_{if} R_{f1} + \vec{b}_i) - R_{i1} \right)^2 + \cdots \\ &\quad + \left( \mathrm{ReLu}(\sum_{f=1}^{m} W_{if} \Theta_{if} R_{fn} + \vec{b}_i) - R_{in} \right)^2 \Bigg] + \frac{\partial J(\Theta, \vec{b})}{\partial \Theta_{kl}} \left[ \frac{1}{2} \lambda \sum_{i=1}^{m} \sum_{j=1}^{m} \Theta_{ij}^2 \right] \\ &= \sum_{j=1}^{n} \left( \mathrm{ReLu}(\sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k) - R_{kj} \right) \mathrm{ReLu}'(\sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k) W_{kl} R_{lj} + \lambda \Theta_{kl} \\ &= W_{kl} \sum_{j=1}^{n} \left( \mathrm{ReLu}(\sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k) - R_{kj} \right) \mathrm{ReLu}'(\sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k) R_{lj} + \lambda \Theta_{kl} \\ &= W_{kl} \sum_{j=1}^{n} \left( \sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k - R_{kj} \right) \mathrm{ReLu}'(\sum_{f=1}^{m} W_{kf} \Theta_{kf} R_{fj} + \vec{b}_k) R_{lj} + \lambda \Theta_{kl}\end{aligned}$$

The last step is due to that $\text{ReLu}(x)\text{ReLu}'(x) = x\text{ReLu}'(x) = 0$ when $x < 0$ and $\text{ReLu}(x)\text{ReLu}'(x) = x\text{ReLu}'(x) = x$ when $x \geq 0$. Notice that in contrast to the convention of $\text{ReLu}'(0) = 0$, here $\text{ReLu}'(0) = 1$. Otherwise, $\frac{\partial J(\Theta,\vec{b})}{\partial \Theta} = 0$ when $\Theta = 0$. Similarly,

$$\begin{aligned}
\frac{\partial J(\Theta,\vec{b})}{\partial \vec{b}_k} &= \frac{\partial}{\partial \vec{b}_k}\frac{1}{2}\sum_{i=1}^{m}\Bigg[\left(\text{ReLu}(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{f1} + \vec{b}_i) - R_{i1}\right)^2 + \cdots \\
&\quad + \left(\text{ReLu}(\sum_{f=1}^{m} W_{if}\Theta_{if}R_{fn} + \vec{b}_i) - R_{in}\right)^2\Bigg] + \frac{\partial J(\Theta,\vec{b})}{\partial \vec{b}_k}\left[\frac{1}{2}\lambda\sum_{i=1}^{m}\sum_{j=1}^{m}\Theta_{ij}^2\right] \\
&= \sum_{j=1}^{n}\left(\text{ReLu}(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k) - R_{kj}\right)\text{ReLu}'(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k) \\
&= \sum_{j=1}^{n}\left(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k - R_{kj}\right)\text{ReLu}'(\sum_{f=1}^{m} W_{kf}\Theta_{kf}R_{fj} + \vec{b}_k)
\end{aligned}$$

In matrix form,

$$\begin{aligned}
J(\Theta,\vec{b}) &= \frac{1}{2}\mathbf{1}^T\{[\text{ReLu}((W\odot\Theta)R) - R]^2\}\mathbf{1} + \frac{1}{2}\lambda\mathbf{1}^T\Theta^2\mathbf{1} \\
&= \frac{1}{2}\mathbf{1}^T\{[\text{ReLu}(\hat{R}) - R]^2\}\mathbf{1} + \frac{1}{2}\lambda\mathbf{1}^T\Theta^2\mathbf{1}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J(\Theta,\vec{b})}{\partial \Theta} &= W\odot\Big([(W\odot\Theta)R + \vec{b} - R]\odot\text{ReLu}'((W\odot\Theta)R + \vec{b})R^T\Big) + \lambda\Theta \\
&= W\odot\Big([(\hat{R} - R)\odot\text{ReLu}'(\hat{R})]R^T\Big) + \lambda\Theta \\
\frac{\partial J(\Theta,\vec{b})}{\partial \vec{b}} &= \{[(W\odot\Theta)R + \vec{b} - R]\odot\text{ReLu}'((W\odot\Theta)R + \vec{b})\}\mathbf{1} \\
&= [(\hat{R} - R)\odot\text{ReLu}'(\hat{R})]\mathbf{1}
\end{aligned}$$

Since the computation of batch gradient descent is expensive, mini-batch gradient descent is used for training. For the matrix $R$, a mini-batch of $R$'s column vectors, $B$, are used as training data set at each iteration, and the mini-batch gradient descent algorithm is:

**Algorithm 1:** Mini-Batch Gradient Descent of the ReLu Model with Regularization

```
$\Theta \leftarrow 0$
$epochCount \leftarrow 0$
for $epochCount < maxNumberOfEpoch$ do
    //$maxNumberOfEpoch$ needs to be determined by experiments
    for $B$, a batch of column vectors from $R$ do
        $\hat{B} \leftarrow (W\odot\Theta)B + \vec{b}$
        $\Theta \leftarrow \Theta - \alpha\frac{1}{|B|}W\odot\Big([(\hat{B} - B)\odot\text{ReLu}'(\hat{B})]B^T\Big) + \lambda\Theta$
        $\vec{b} \leftarrow \vec{b} - \alpha\frac{1}{|B|}[(\hat{B} - B)\odot\text{ReLu}'(\hat{B})]\mathbf{1}$
    end
    Report the loss $J(\Theta,\vec{b}) = \frac{1}{2n}\mathbf{1}^T\{[\text{ReLu}(\hat{R}) - R]^2\}\mathbf{1}$
    $epochCount \leftarrow epochCount + 1$
end
```

### 4.4 Model Assessment

To validate the model, two assessment measurements are used: (1) mean squared error (MSE) and (2) average precision of the top $K$ predicted keywords (APTK). MSE is used to define the loss function, and here, it is to measure the overall performance of the model. APTK is to measure how well the model can predict the missing keywords, which is the practical goal of the model. Both measures require a masking step. In detail, for a known document vector, a percentage of its $> 0$ values will be randomly masked to 0. Then the model will predict on the masked document vector, and the performance of the prediction is measured. The masking percentage is a key parameter affecting the result. If the masking percentage is too high, then the model will not have enough information to use. If the masking percentage is too low, then it becomes an extremely unbalanced classification problem. Here, it is set to 50%.

#### 4.4.1 MSE and MSER

For a testing matrix $X$, its masked matrix $X'$, and the predicted $\hat{X}'$.

$$MSE = \frac{\mathbf{1}^T(\hat{X}' - X)^2\mathbf{1}}{|X|}$$

where $|X| = m \times n$, the total number of elements in $X$. However, $X$ is very sparse, and MSE is usually a small number. Therefore, a mean squared error of the relevant (MSER) is defined:

$$MSER = \frac{\mathbf{1}^T(\hat{X}' - X)^2\mathbf{1}}{|\{X > 0\} \cup \{\hat{X}' > 0\}|}$$

The denominator means the total number of elements that is not 0 in either $X$ or $\hat{X}'$. MSER has a better resolution than MSE (MSER$\geq$MSE) and is used instead of MSE in this study.

#### 4.4.2 APTK and APRK

Denote each document vector of $X$ as $\vec{x}$, the masked document vector as $\vec{x}'$, and the predicted document vector as $\hat{\vec{x}}'$. The MeSH terms, which is 0 in $\vec{x}'$ but $> 0$ in $\hat{\vec{x}}'$, can be ranked, and the top $K$ can be selected. The precision@$K$ for $\vec{x}$ is

$$\text{precision@}K = \frac{\text{number of top } K \text{ MeSH is also} > 0 \text{ in } \vec{x}}{K}$$

APTK is the average of the precision@$K$ over all document vectors in $X$. As a comparison, a random selector will uniformly pick $K$ MeSH terms instead of the top $K$ to calculate the average precision at random $K$(APRK). For this study, $K = 3$.

## 5 Results

### 5.1 Data Statistics and Data Selection

The whole data set contains total 337371 documents with 15071 MeSH terms (data downloaded from ClinicalRecords.gov on April 27th, 2020).

Due to the limitation of computational power, only MeSH terms present in 0.5% of all documents are considered first (total 947, ie. $m = 947$). The frequency of the TF-IDF values in the train data set is shown in Figure 1A which roughly following the Zipf's Law. For each document, the number non-zero TF-IDF MeSH terms is counted and the distribution is a shown in Figure 1B. It is obvious that the TF-IDF matrix is very sparse. For most documents, they have less than 100 non-zero MeSH terms. Therefore, this is very unbalanced data and predicting MeSH terms from more than 900 MeSH terms is a very challenging task for the model.

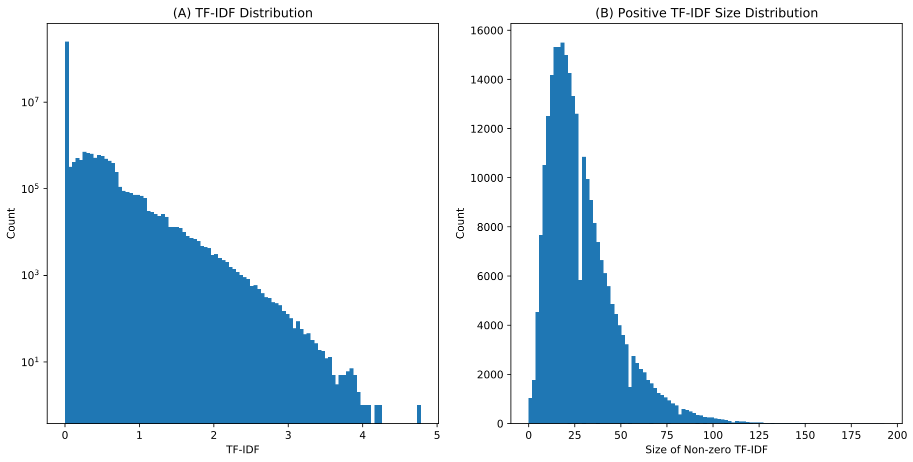


Figure 1: Histogram of frequency of the TF-IDF values in the train data set

## 5.2 Model Training and Test

To train the model efficiently, the learning rate $\alpha$ was first searched (other variables fixed:$\tau = -1$, $k = 4$, and $\lambda = 1e - 6$). The loss descent curve (Figure 2) and the MSER as well as the APTK on $X_{dev}$ was calculated after 30 epochs. In the end, $\alpha = 0.1$ was chosen, because it gave a quick descent and reached the lowest MSER and the highest APTK. In addition, at the end of the 30th epoch, the loss was still descending, suggesting a longer training might give a better result.

After $\alpha$ was fixed at 0.1, a grid search was performed to find the best $\tau$, $k$, and $\lambda$. The details are shown in Figure S1. It is obvious that $\lambda = 0.01$ is the best. To determine $\tau$ and $k$, the the area under the curve of $e^{tau(1-x)^k}$ ($x \in [0, 1]$) was used, and it is clear that area is closer to 1 the better. This is probably due to the fact that only MeSH terms of high frequencies were considered, and they are all related. Therefore, in the final model $tau = -0.001$, $k = 1000$, $\lambda = 0.01$, and $\alpha = 0.1$. The model was trained for 60 epochs. On $X_{dev}$ it reached an MSE of 0.008049, an MSER of 0.008332, and an APTK of 0.4750 (APRK=0.01713). On the $X_{test}$, the model reached similar performance: an MSE of 0.008089, an MSER of 0.008371, and an APTK of 0.4792 compared to the APRK=0.01690, suggesting the model works reasonably well.

Because of the success on the data set having 947 MeSH terms, the model was trained and tested on the full data set which has 15071 MeSH terms, and use the same hyperparameters. On the $X_{dev}$ the model has an MSE of 0.001054, an MSER of 0.001618, and an APTK of 0.5035 (APRK=0.001029). On the $X_{test}$, the model reached an MSE of 0.001047, an MSER of 0.001608, and an APTK of 0.5031 (APRK=0.001532).

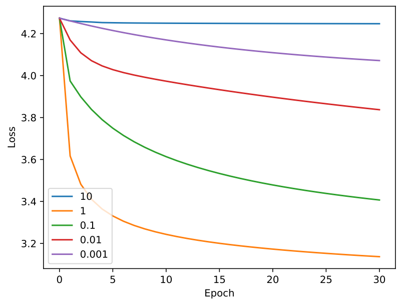


Figure 2: Searching $\alpha$

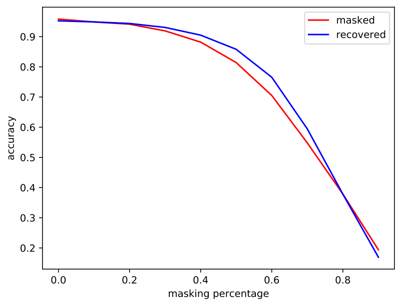
## 6 Discussion

### 6.1 Other Usage: Data Recovery

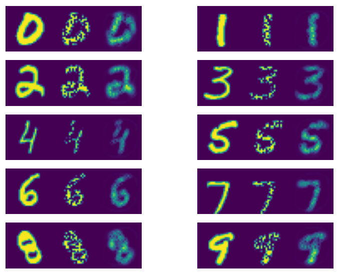
The collaborative filtering model developed here learns latent patterns from a data set $R$ and then it predicts missing information from another data set $X$ which follows the same pattern as $R$. Therefore, it can be used as an information recovery method. To test this, the MNIST Handwritten Digit Classification Data Set is used. In the course problem set, we have built a neuron network that classifies the handwritten digits in high accuracy. The idea is to use the collaborative filtering model to recover damaged images and see if the recovery helps to increase the accuracy of the neuron network.

In detail, (1) train both the collaborative filtering model and the neuron network on the train data set and development data set; (2) randomly masks (set to zero) a percentage of the pixels in the images of the test data set (call this the masked data set); (2) use the trained collaborative filtering model to "recover" the masked data set (call this the recovered data set); (3) use the trained neuron networks to predict the digits from the the masked and the recovered data set, and then compare the accuracy.

The hyperparamters of the collaborative filtering model were determined in the same way as before. The final parameters are $\tau = -10$, $k = 4$, $\lambda = 0.001$, and $\alpha = 0.01$. In contrast to the MeSH recommendation problem, the under curve area governed by $\tau$ and $k$ is 0.51 not 1. The pixel values were normalized to $\in [0, 1]$ (directly divided by 255). The masking percentage affects the results, therefore, a range of masking percentage was tested. As shown in Figure 3a, the recovery by the collaborative filtering model is most helpful (about 5% accuracy increase) when the masking percentage is around 50%. This is reasonable, since the model needs existing information to recover the lost. Examples of recovered image that were damaged at 50% masking percentage are shown in Figure 3b. Although the overall intensity decreased and the intensity distribution changed, the disconnect traces of the masked digits were filled in.



(a)



(b)

Figure 3: Effects of the recovery of the masked digital images. (a) Effects on prediction accuracy of the neuron network under different masking percentage. (b) Examples of recovered digital images damaged at 50% masking percentage: original (left), masked (middle), and recovered (right).

### 6.2 Feature Mapping and the Kernel Method

The model developed in this study can be seen as a one layer neuron network. The neighboring weight W and the ReLu function add non-linearity to the model. The hyper parameter $\tau$ and $k$ used to calculate W are not trained but tuned. The question is instead of using the $W$, $\vec{b}$, and $ReLu$, is it possible to map the features to other non-linear features? Can the kennel trick be used on this problem?

Let the $\Phi$ be the feature-mapping function, $\Phi : m \times n \to m' \times n$, that is $\Phi$ is a function that maps a matrix having $m$-dimension column vectors to a matrix having $m'$-dimension column vectors. The loss function is

$$J(\Theta) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{f=1}^{m'}\Theta_{if}\Phi(R)_{fj} - R_{ij}\right)^2$$

Therefore,

$$\begin{aligned}\frac{\partial J(\Theta)}{\partial \Theta_{kl}} &= \frac{\partial}{\partial \Theta_{kl}}\frac{1}{2}\sum_{i=1}^{m}\left[\left(\sum_{f=1}^{m'}\Theta_{if}\Phi(R)_{f1} - R_{i1}\right)^2 + \cdots + \left(\sum_{f=1}^{m'}\Theta_{if}\Phi(R)_{fn} - R_{in}\right)^2\right] \\ &= \sum_{j=1}^{n}\left(\sum_{f=1}^{m'}\Theta_{kf}\Phi(R)_{fj} - R_{kj}\right)\Phi(R)_{lj}\end{aligned}$$

Hence,

$$J(\Theta) = \mathbf{1}^T[(\Theta\Phi(R) - R)\odot(\Theta\Phi(R) - R)]\mathbf{1}$$

$$\frac{\partial J(\Theta)}{\partial \Theta} = (\Theta\Phi(R) - R)\Phi(R)^T$$

The goal is to proof that at the $i$'s training iteration, $\Theta^{(i)} = \Omega^{(i)}\Phi(R)^T$, where $\Omega^{(i)}$ is a $m \times n$ matrix.
For $\Theta^{(0)} = 0$, this is obviously true with $\Omega^{(0)} = 0$.
For $\Theta^{(i+1)}$,

$$\begin{aligned}\Theta^{(i+1)} &= \Omega^{(i)}\Phi(R)^T - \alpha\left(\Omega^{(i)}\Phi(R)^T\Phi(R) - R\right)\Phi(R)^T \\ &= \Omega^{(i)}\Phi(R)^T + \alpha\left(R - \Omega^{(i)}\Phi(R)^T\Phi(R)\right)\Phi(R)^T \\ &= \Omega^{(i)}\Phi(R)^T + \alpha R\Phi(R)^T - \alpha\Omega^{(i)}\Phi(R)^T\Phi(R)\Phi(R)^T \\ &= \left(\Omega^{(i)} + \alpha R - \alpha\Omega^{(i)}\Phi(R)^T\Phi(R)\right)\Phi(R)^T \\ &= \left[\Omega^{(i)} + \alpha\left(R - \Omega^{(i)}\Phi(R)^T\Phi(R)\right)\right]\Phi(R)^T\end{aligned}$$

Therefore, $\Theta^{(i+1)} = \Omega^{(i+1)}\Phi(R)^T$ and $\Omega^{(i+1)} = \Omega^{(i)} + \alpha\left(R - \Omega^{(i)}\Phi(R)^T\Phi(R)\right)$. The prediction on $\vec{x}$ after training is $\Omega\Phi(R)^T\Phi(\vec{x})$.

Hence, a kernel trick can be used and the training can be finished in the order of $n$, instead of the dimension of the mapped features.

## References

[1] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020.

[2] Stephen E Robertson and Steve Walker. "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval". In: *SIGIR'94*. Springer. 1994, pp. 232–241.

[3] Andreas Töscher, Michael Jahrer, and Robert M Bell. "The bigchaos solution to the netflix grand prize". In: *Netflix prize documentation* (2009), pp. 1–52.

## 7 Supplemental Information

### 7.1 Figure

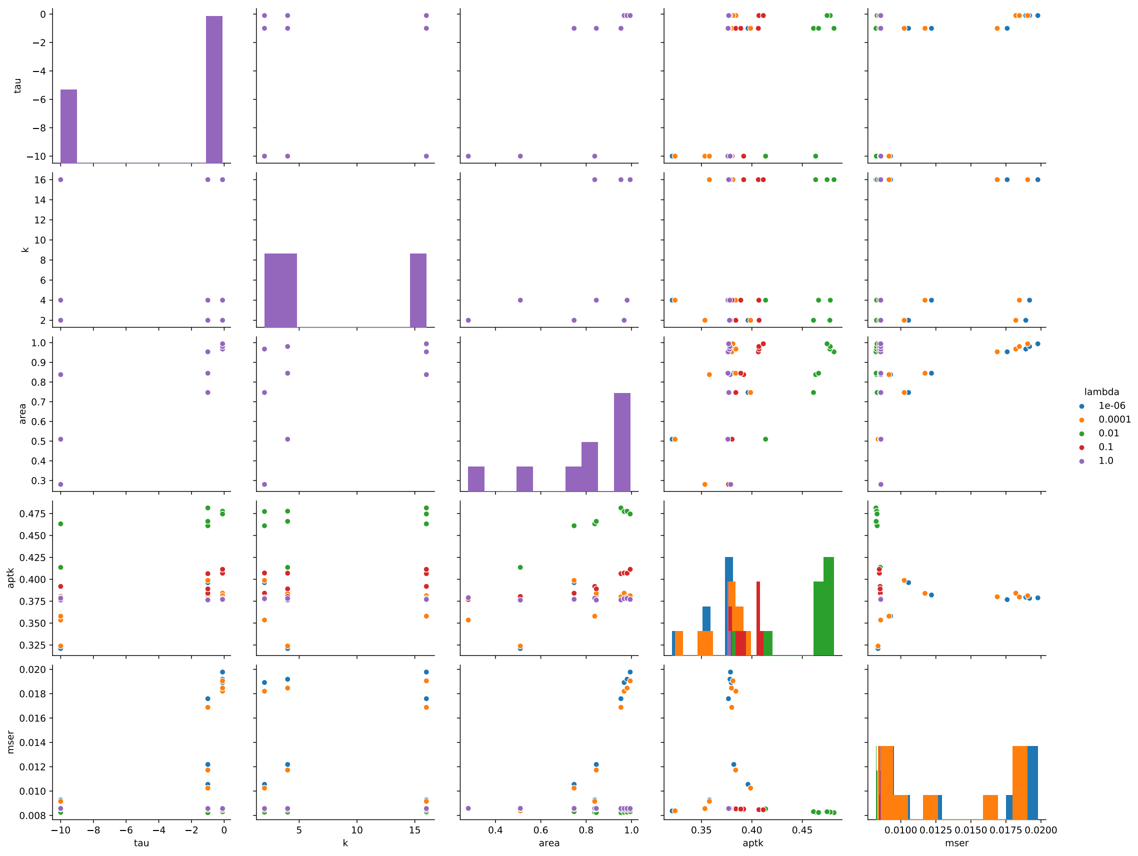


Figure S1: Searching $\tau$, $k$, and $\lambda$, under $\alpha = 0.1$

### 7.2 Source Code

`https://github.com/xiaomutt/xliniq`
NOTICE:

- The source code for the Handwritten Digits in the discussion is not included, because it is also a problem set question for Stanford CS229.
- Original data are not included. They can be downloaded from the NLM directly.
- The outputs and results files are not included. They can be obtained by running the source code.
- The GitHub Repo is subject to updates.