VECTOR-SPACE MAPPING OF MULTIMEDIA AND WEB DATABASE FEATURES

Akshay Muraleedharan Nair Santhy, Suprada Ganesh Nadiger, Hari Kripa Omalur Chandran, Tinu Tomson, Jasmin George, Bosco Paul

ABSTRACT

This project is an experimental implementation of various types of User recommendations, Feedback Systems, Multidimensional Index Structures, Nearest Neighbour Search, and Classification Algorithms using vector and graph models.

The User recommendations are done using SVD, PCA, LDA, Personalized Pagerank, Tensor Decomposition, and an ensemble of all these algorithms. A feedback is provided to these recommendations using 'Probabilistic Relevant Feedback' mechanism.

A Multidimensional Index Structure is implemented using LSH, Nearest Neighbour Search method is used to implement the relevant feedback algorithm.

Classification algorithms are implemented using R-NN, Decision Tree, and SVM

KEYWORDS: Pagerank, SVD, LDA, PCA, LSH, Tensor Decision Tree, Gini Impurity, Information Gain, Index Structure, Hashes, R-NN, SVM, CART, Entropy.

INTRODUCTION

Terminology

$TF_{(t,d)}$ - Term Frequency, which measures how frequently a term t occurs in a document d

$IDF_{(t,D)}$ - Inverse Document Frequency, which measures the importance of a term *t* within the set of documents D. The IDF will decrease the weight of terms which are common to all many documents within the set D.

$TF-IDF_{(t,d,D)}$ - Combined TF and IDF measure for a term t in a document d within the set of documents D.

PageRank - An algorithm used by google search engines to rank the importance of web pages.

Personalized PageRank - A modified version of PageRank to factor in a seeded list of initial pages.

Cosine similarity - Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

Tensor - Tensor is an N dimensional array.

CP - CP is a tensor decomposition technique used for mapping tensors from a high to lower dimensional space.

PCA - Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

SVD - Singular-value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigen-decomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any *mxn* matrix via an extension of the polar decomposition.

***LDA*-** a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics.

KNN- In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a nonparametric method used for classification and regression.[4]

LSH - Locality-sensitive hashing (LSH) reduces the dimensionality of high-dimensional data.[5]

Decision Tree - Decision tree learning is a method commonly used in data mining. They are of mainly 2 types, Classification Tree and Regression Tree. [3]

CART - The term Classification And Regression Tree (CART) analysis is an umbrella term used to refer to both of Classification Tree and Regression Tree. This algorithm is used in the implementation of Decision tree.

Gini Impurity - Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

Information Gain - Information gain is used to decide which feature to split on at each step in building the tree.

Entropy - Information entropy is defined as the average amount of information produced by a stochastic source of data.

Goal Description

Task1:

Goal of this task is to recommend more movies using all the information available for the given movies a user has watched. The recency and the order in which user has watched the movies are taken into consideration to recommend more movies.

1.a In this subtask the movies vectors are reduced to lower dimensions using SVD and the analysis is done on the transformed matrix.

1.b In this subtask the movies vectors are reduced to lower dimensions using LDA and the analysis is done on the transformed matrix.

1.c In this subtask the movies vectors are reduced to lower dimensions using tensor decomposition and the analysis is done on the transformed matrix.

1.d Using all the information available for the given movies the user has watched, recommend 5 other movies using personalized pagerank.

1.e In this subtask the movies vectors are reduced to lower dimensions using SVD, LDA, tensor decomposition, personalized pagerank algorithm and a combined metric is to be created compare movies.

Task 2:

Take the feedback from user about Task 1 and refactor our query so as to output better personalized results. Probabilistic feedback method is applied to refine the query to get improved results based on user feedback.

Task 3:

There are 3 objectives

1. Reduce the dimensions of entire database of movies to 500
2. Implement a Locality Sensitive Hashing tool where user provides no of layers, hashes per layer and dataset of movies
3. Interface for user to input a query movie and the program outputs n similar movies

Task 4:

Take the feedback from user about Task 3 and refactor our query so as to output better personalized results.

Task 5: Classify all movies in database based on the training data given by the user. User gives labels and sample movies as input. Rest of the movies should be classified based on the input

5a: using R- Nearest Neighbour Classification

5b: using Decision Tree Classification

5c: using SVM Classification

Assumptions

(1) PageRank is initialized to value 1. Personalized PageRank iteration is done for 5 times. The probability of random walker jumping back to seed actors is 0.15. Therefore, probability of transition is 0.85

(2) The movies from year 2004 are considered to tensor decomposition so as to reduce the time and space complexity.
(3) Input list of movies to task1 is less than 100.
(4) Assuming input dataset for movies are dense enough for LSH to work
(5) Assuming the user will provide the input for Task 5 in a csv file format with movie id and labels as columns.
(6) Before applying Classification Algorithms, the Movie -Features matrix is reduced to 2 latent features. We assumed only 2 latent feature to reduce computation time.
(7) For feedback on the recommendation, the user will give 3 inputs, If relevant then 'Yes' , 'No' for irrelevant and 'Neutral' for results doesn't make sense.
(8) For SVM classifier the data is assumed to be linearly separable and the data belongs to only two different classes.

FORMULAS

- The Term Frequency (TF) is defined as

$$TF_{(t,d)} = \frac{Number\ of\ times\ a\ term\ t\ appears\ in\ a\ document\ d}{Total\ number\ of\ terms\ in\ the\ document}$$

- The Inverse Document Frequency (IDF) is defined as

$$IDF_{(t,D)} = log(\frac{Total\ number\ of\ documents\ in\ the\ corpus}{Number\ of\ documents\ with\ term\ t\ in\ it})$$

- The IDF can further be used in to eliminate the features which are very common among all the objects.

$$TF - IDF_{(t,d,D)} = TF_{t,d} * IDF_{t,d}$$

- The cosine similarity is defined as

$$Similarity = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- The PageRank of a seeded actor j is defined as

$$= \frac{0.15}{number\ of\ seeded\ actor} + 0.85 * \sum_i PageRank(actor\ i) * TransitionMatrix(i,j)$$

$$+ \ (0.01/(seed\_movies.index(i) + 1))$$

- The PageRank of the other actor j is defined as

$$= \sum_i PageRank(actor\ i) * TransitionMatrix(i,j)$$

- Gini Impurity

$$I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

where,[3]

J = set of labels

Pi = the fraction of items labeled with class **i** in the set

- Entropy

$$H(T) = I_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$ [3]

- Information Gain

$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{H(T)}^{\text{Entropy(parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy(Children)}}$$ [3]

- The probabilistic feedback query is calculated as

$$Q = log(\frac{ri + \frac{ni}{N}}{R - ri + 1} \div \frac{ni - ri + \frac{ni}{N}}{N - ni - (R - ri) + 1})$$

where,

ri sum of weights of relevant items retrieved for term i

ni sum of weights of total number of retrieved items for term i

R total number of relevant retrieved items

N is the total number of items in the collection

For Task 4 query is shifted based on feedback using the following formula

Q' = Q + crel × Σor + cir × Σoir

or - relevant objects

oir - irrelevant objects

Hash Function for LSH

$h(x) = floor((p.x + b)/w)$

x - movie
p - random projection
b - random value between [o,w)
w - length of a segment

PROPOSED SOLUTION AND IMPLEMENTATION

Description of the Proposed Solution

(1) Task 1

So as to include all the information available in the task, we create a matrix by user ratings, movie genre, actors in the movie and their rank and tf-idf of all the tags the movie received. This matrix is dumped into an HD5 file type and is used by all the subtasks for the task.

- Task 1a – The subtask is to perform SVD or PCA on the matrix to extract latent features from the matrix. Top 4 latent semantics were chosen. Cosine similarity between wows corresponding to input movies and every other row in the matrix is calculated and movies are ranked in the order of highest cumulative cosine similarity. The order of the movie is taken into consideration by multiplying similarity matrix with an order factor such that first watched movies is given higher priority. Top five movies are filtered and displayed.
- Task 1b – The subtask is to perform Latent Dirichlet Allocation (LDA) on the matrix to extract latent features from the matrix. Top 4 latent semantics were chosen. Cosine similarity between wows corresponding to input movies and every other row in the matrix is calculated and movies are ranked in the order of highest cumulative cosine similarity. The order of the movie is taken into consideration by multiplying similarity matrix with an order factor such that first watched movies is given higher priority. Top five movies are filtered and displayed. Top five movies are filtered and displayed.
- Task 1c – For the tensor decomposition task, the actor-movie-genre tensor is considered. CP decomposition is performed on the tensor to get the factor matrices in terms of the top 4 latent semantics.
  i. Create actor movie genre tensor using the following constraint
     • Set the value to one if, for any actor-movie-genre triple the given actor played in the stated movie and contains the genre
     • Set the value to zero otherwise.
  ii. Use “parafac” method of tensorly package to perform CP decomposition on the tensor. The decomposition will produce three factor matrices for actors, movies,and genre in terms of four latent semantics.

iii. Use the movie factor matrix to compute cosine similarity with the input movies and recommend movies with the top five values to the user.

- Task 1d – Recommend 5 movies to the user which are similar to the other movies the user has watched using personalized pagerank, the order and recency of the watched movies should also be considered.
  (1) Create a Movie features Table which contains all the features of a movie (Year, Actors, Genre, TF-IDF of Tags, Rating). This table is created at the beginning of the project as it's being used by all the Tasks. This file is saved in a HD5 Table.
  (2) Pick the HD5 table of movies. This movie- feature table is too huge to apply Personalized pagerank, so apply a dimensionality reduction on this file.
  (3) Task 3 has to reduce the Movie features table into 500 dimensions. Use this reduced matrix as the input for Personalized Pagerank.
  (4) Compute a Cosine similarity on moives to movies and create a normalized movie - movie matrix. The similarity between movies will act as the edges of the movie graph.
  (5) The values of row i represents the probabilities with which a random walker jumps to another movie from movie-i. (normalized sum adds to 1)
  (6) This normalized matrix can be used as the transition matrix. Transition Matrix(i,j) represents the probability with which random walker jumps from movie-i to movie-j. Pagerank of all actors is initialized to 1.
  (7) Pagerank is then iteratively calculated using the formulas mentioned in formula section. Here the order in which the user given the input is considered, the movie watched first gets a higher weight and the movie watched last gets a lower grade.
  (8) The movie matrix is created with the the year in which it was released, and the year value is normalized in such a way that latest movies gets a higher weight. Hence, when the user inputs movies, their recency will also considered in the calculation.
  (9) Once the calculations are done, the 5 Movies (except seed movies) with highest pagerank are then displayed.

- Task 1e – So as to combine all the above metrics SVD, LDA, tensor decomposition and PageRank algorithm are performed as like above and a similarity metric is calculated as per each task. We then average these similarity weight and top five movies are filtered and displayed.

(2) Task 2 - Probabilistic relevance feedback is used in this task for improving the movie recommendations. User feedback is taken on the movie recommendations as input for processing the new query.

I. Probabilistic relevance feedback for task 1a, 1b, 1c and 1e

i. Based on the relevance feedback, the new query is formed using the probabilistic feedback formula (see formula section for detailed explanation of individual terms)

$$Q = log(\frac{ri + \frac{ni}{N}}{R - ri + 1} \div \frac{ni - ri + \frac{ni}{N}}{N - ni - (R - ri) + 1})$$

ii. The newly obtained query is used for computing cosine similarity with the movie matrix and the top five movies are recommended. The movies that are already watched and the irrelevant movies are excluded from the recommendation.

II. Probabilistic relevance feedback for task 1d (Pagerank)

i. The page ranks of the movies are recomputed by increasing the rank of relevant movies by 0.02 each time the movie node is encountered in the random walk and by decreasing the rank of the irrelevant movies by 0.02.

III. Probabilistic relevance feedback for task 1e

i. Based on the relevance feedback, the new query is formed using the probabilistic feedback formula (see formula section for detailed explanation of individual terms)

$$Q = log(\frac{ri + \frac{ni}{N}}{R - ri + 1} \div \frac{ni - ri + \frac{ni}{N}}{N - ni - (R - ri) + 1})$$

ii. The cosine similarity of new query formed is calculated with the svd, lda and tensor latent movie vectors. The pagerank for the movie vector is recalculated with the relevance feedback as mentioned in relevance feedback for task 1d. All the movie vectors are combined by aggregating the values and the movies with top 5 values are returned.

(3) Task 3 -

The matrix used for Phase 2 Task 4 is used as our base for movies. This matrix is compressed to 500 dimensions using SVD. No of hashes h, no of layers l and set of movies are taken as input from user. Set of movies is optional(entire movies taken if not provided). Using this in memory index structure is created using LSH. Each movie is hashed h times on a layer. The bucket of the movie is concatenation of h hashes in a layer. The same is repeated for L layers. Hash function is calculated using formula

h(x) = floor((p.x + b)/w)

x - movie

p - random projection

b - random value between [o,w)

w - length of a segment(ideally should be >>k, but fixed as 20)

There will be l*h p and b values. A movie will fall only in one bucket per layer. It will be there in one bucket at every layer.
Next the user provides an input query movie and asks to output k nearest movies. All the movies in buckets where query_movie fell in is considered. The top k similar movies among them is displayed to the user.

When there are not enough(k) movies in the buckets gathered, we try to get the movies from similar buckets also. Specifically checking the buckets with only one hash value different from original query in each layer. This could ideally be done till all buckets are searched. We stop either when we find k movies or when all buckets with hamming distance 1 is queried.(we do not go further than 1 hamming distance).

(4) Task 4
After the output in task 3, user is asked for feedback of the movies. User has an option of giving irrelevant, relevant or neutral feedback. We move the query based on the user feedback using the formula.

$$Q' = Q + \text{crel} \times \sum \text{or} - \text{cir} \times \sum \text{oir}$$

or - relevant objects
oir - irrelevant objects
crel and cir was fixed as 1/(no of relevant objects+1) and 1/(no of irrelevant objects+1)respectively.

This improves the query based on the user feedback.

(5) Task 5

- Task 5a –

Input from user which include labels and movie id is taken in form of a csv file. From the rest of movies, each movie's cosine similarity to input movies is calculated. The top k similar training data movies vote on the new movies label. This is done for rest of the movies. This is an implementation of KNN algorithm for classification with majority voting using cosine similarity.[4]

- Task 5b –
  (1) Read the movie -features table(HD5) created at Task1, which contains all the features of a movie
  (2) There are too many feature in this table to apply decision tree, so apply a dimensionality reduction first and reduce the feature size to 2.(feature size of 2 is a random choice to reduce computation time)

(3) Follow CART[3] implementation algorithm to implement Decision tree. Which is, build tree, assign all the training data to the root node.
(4) Find the most discriminating features of the training data set. Create a condition with the most discriminating feature and compute the Gini Impurity of the condition. Which is, the quantitative data saying 'labels in the new child node/ total labels). lesser this factor, better the result is.
(5) split the data based on the condition given. Here it is to compare the numerical latent feature values. Compute the Information gain on each branches, chose the highest split and create new nodes in the tree.
(6) Call this build tree function recursively
(7) assign labels at every leaf node
(8) Once the tree is build, feed the test data into the tree, read all the labeled leaf nodes where the test data is dropped according to the conditions.
(9) Print all the leaf nodes and their labels

- Task 5c – Is to create an n-ary SVM based classification algorithm.
  SVM classifier is implemented based on the assumption that the data in linearly separable and the data will belong to only two labels.
  Step1: A matrix is created by representing all the movies in the database in terms of actors, tags, genres and users (This was implemented in phase 2 for Task 4).
  Step2: SVD is performed on this matrix to reduce the dimensionality to 2. These two dimensions are used as the features for classification.
  Step3: For a given training set of movie id's and labels, for each movieId the corresponding latent semantics are extracted and the labels are plotted in the space of these two latent semantics.
  Step4: The optimized values of 'w' and 'b' are found such that it maximizes the boundary between two groups of labels (The optimization part has not been coded by the team. The optimizer from pythonprogramming.net was used to solve the optimization problem).
  Step5: After the training is done, for the test set of movieId's the corresponding two latent semantics are extracted.
  Step6: The prediction of label is done based on which part of the separator the new movie belongs to.

Interface Specifications

The algorithms described above were implemented using Python 2.7. The scripts and functions were tested on Linux and Mac OS.

System Requirements

(1) Linux and Mac OS

(2) Python 2.7 with pandas, numpy, scipy, sklearn, tensorly, tables packages installed.

Installation and Execution Instructions

pip install pandas, numpy, scipy, sklearn, tensorly, tables,

Task 1 & 2:

cd Code/task1

```
python pca.py 3189 3216 3233
python svd.py 3189 3216 3233
python lda.py 3189 3216 3233
python tensor_creation.py
python tensordecomposition.py 8779 7056 8843
python pagerank_recommend.py 3189,3216,3233
python combined.py 8779 7056 8843
```

Task 2:

Task 2 is continuation of task 1 so it is executed from the same interface as task1.

Task 3:-

Run python map.py

This reduces dimensions of movies to 500 and stores in file movie_matrix_svd.pkl

Run python lsh.py #layers #hashes #movie_list

#movie_list is optional(all movies considered when given empty)

Eg:- python lsh.py 5 10

python lsh.py 5,10 65,3111

The above code asks questions on console

Sample questions on console

Enter movie id to query:- 12

Enter range of query:- 1

Task 4:-

Interface for task4 is provided with task3 code itself

Sample questions on console

Give Feedback as String of 1(Good),0(Neutral),-1(Bad) :- 1

66 0.835438

Give Feedback as String of 1(Good),0(Neutral),-1(Bad) :-

Task 5a:-
Input file is movie_labels.csv
>>python rnn.py
It outputs rest of movies and classified labels
Task 5b:
Input file is movie_labels.csv
>>python movie_matrix_svd.py
>>python decision_tree.py
Task 5c:
Input file for training should be named as training-data.csv
Input file for testing should be named as test-data.csc
Both of these files should be in task5 folder
>>python svm.py

Related Work

- Efficient Indexing for High-Dimensional Similarity Search[1] uses an interesting approach that guarantees a fixed no of results when searching using LSH. They used a novel idea to probe the hash function so as to access data from neighbouring hash functions also. This could theoretically be repeated which will result in entire database being queried.
- Google Page-ranking- for ranking and indexing search results of web pages
- Relevance feedback is used in the ecommerce domain extensively for giving personalised suggestions to users.

SUMMARY/CONCLUSIONS

Task 1:
SVD, LDA and tensor decomposition are useful ways to reduce dimensions of a high dimensional data. The vectors from the latent space can be used for similarity computation with an input query and can be used to find relevant documents which can be useful in recommendation systems.

Task 1d:
Applied Personalized pagerank on movie to movie similarity matrix of latent feature to include all features of a give movie in a concise and easily computable format to recommend 5 similar movies to the input movie set. It was helpful in understanding that pagerank can also be applied on weighted edge graphs, taking into consideration of the weights of the edges.

Task 2:

Applied probabilistic relevance feedback to modify the query space and refine the feedback based on user feedback. Probabilistic feedback shifts the query in direction of relevant objects and away from irrelevant objects.

Task3:
LSH was applied on the movies and nearest neighbour search was performed on the LSH dataset. This gave fast results sacrificing little accuracy. Sometimes the query did not return enough result when the data was sparse even after probing the search query.

Task4:
The query was modified using the formula (in formula section) based on the feedback given by the user. This shifts the query in the direction of relevant objects away from irrelevant objects. When the new query point is used as our basis, we get better results personalized for the user.

Task5:
This task implemented three different classification algorithms such as k-nearest neighbor, Decision tree and Support Vector Machines. All three classifiers were trained using the training data. Based on this training the classifiers will predict the class for a new set of data for which class labels are not known.

REFERENCES/BIBLIOGRAPHY

[1]Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search Qin Lv William Josephson Zhe Wang Moses Charikar Kai Li
http://www.cs.princeton.edu/cass/papers/mplsh_vldb07.pdf
[2] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science. 41, pp. 288-297, 1990.
[3] https://en.wikipedia.org/wiki/Decision_tree_learning
[4] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[5] https://en.wikipedia.org/wiki/Locality-sensitive_hashing
[6] https://en.wikipedia.org/wiki/PageRank
[7] https://cs.stanford.edu/people/plofgren/bidirectional_ppr_thesis.pdf
[8]https://pythonprogramming.net/support-vector-machine-fundamentals-machine-learning-tutorial/
[9] https://cs.stanford.edu/people/plofgren/bidirectional_ppr_thesis.pdf
[10] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery.In KDD, pages 653658, 2004.
[11]https://tensorly.github.io/stable/modules/generated/tensorly.decomposition.parafac.html

APPENDIX

SPECIFIC ROLES OF GROUP MEMBERS

The tasks were distributed among the group. The implementation details were discussed in the group but the actual implementation was carried out by individual group members as follows.

Task 1.a - Tinu Tomson
Task 1.b - Tinu Tomson
Task 1.c - Jasmin George
Task 1.d - Akshay Muraleedharan Nair Santhy
Task 1.e - Tinu Tomson
Task 2 - Jasmin George
Task 3 - Hari Kripa Omalur Chandran, Bosco Paul
Task 4 - Hari Kripa Omalur Chandran, Bosco Paul
Task 5a - Hari Kripa Omalur Chandran, Bosco Paul
Task 5b - Akshay Muraleedharan Nair Santhy
Task 5c - Suprada Ganesh Nadiger

Report - Suprada Ganesh Nadiger, Tinu Tomson, Jasmin George, Akshay Muraleedharan Nair Santhy, Hari Kripa Omalur Chandran, Bosco Paul.

Appendix B