Data Scientist Challenge

https://www.datascientistchallenge.com/

by Citi Group

Gautam Kumar (17)
Jaideep Karkhanis(18)
Jayanta Mandi(19)
Sajal Jain(40)

Credit Card Industry

Information rich data

Millions of customers

Transaction data is one good data source

On average approximately 120 annual records are created for each active customer.

Incredible insights into customer preferences and lifestyle choices.

Sheer size of the data and the processing power required,

Objective

The objective of this challenge is to identify the Top 10 merchants that customers have not transacted within the past 12 months and are likely to transact with, in the next 3 months.

Personalize these recommendations at a customer level

Can be used to give customized offers from those merchants to customer, from whom he is likely to shop.

Prime business reason behind all this analysis

Data Format

Customer #	Merchant #	Month	Category	# of transactions
1	1	1	Grocery	3
2	2	1	Dining	2
3	3	1	Grocery	5
4	4	2	Apparel	1

Number of customers: 374328
Number of merchants: 9822

Number of transaction:63659708
Number of merchant categories: 15

Data Summary

No missing data

Data size = 690.5 mb

dim(citi_data)

[1] 18532355   5

range(citi_data$TXN_MTH)

[1] 201309 201408

The15 categories are disjoint sets with merchants

R Session Aborted

R encountered a fatal error.

The session was terminated.

Start New Session

Scoring

If Merchant 1 has been identified correctly for a customer, 10 points are added to the score. Correctly identifying Merchant 2 adds 9 points, and so on.

Customer #	Merchant 1	Merchant 2	Merchant 3	Merchant 4	Merchant 5	Merchant 6	Merchant 7	Merchant 8	Merchant 9	Merchant 10
1	76	61	91	98	6	95	3	29	4	88
2	47	69	80	58	21	82	1	35	37	61
3	67	39	91	32	86	12	28	14	55	99
4	44	19	4	37	35	89	28	99	96	80
5	84	5	33	21	60	85	96	68	10	100
6	21	89	9	72	96	1	12	14	41	84
7	64	48	53	44	31	9	10	83	25	17

Code to be executed on a High Performance Machine with 96 GB RAM.

Algorithm should run within 12 hours in terms of execution time.

Category of Problem

It was a collaborative filtering problem, where people can be recommended about items with similar taste have bought before.

Approaches Tried

Naive approach:

- Find out the weightage of categories for individual customer, based on number of transactions.
- Find best merchants in each of the categories.
- For a customer, we know which are his/her most important category. Find out the merchants they transacted with in the last 1 year in each category.
- Subtract the merchant customer has transacted in the past 1 year from the best merchants in the category. Recommend the top 10 merchants for customer for categories based on the weightage of categories.

Approaches Tried (Continued...)

- Data seems to be time (quarter) independent.
- Additional weightage to the 1st quarter as we need to predict for next 1st quarter.
- Detailed time series analysis doesn’t seem to be appropriate. (¼th of the total time duration we need to predict).
- Cyclicity in terms of annual transactions.

Approaches Tried (Continued...)

Matrix factorization based collaborative filtering:

- Create a matrix which is a Customer x Merchant Matrix. Each element represents the number of transactions a customer has done with the specific matrix. Since normal matrix of size 3,75,000 x 10,000 was not possible, so we create a sparse matrix.
- We factorized the sparse matrix doing SVD using “Irlba” package.
- We did SVD using 10,20,30,50,75,100,125,150,200 rank approximations
- We reconstruct one row of the original at a time by doing Ui X Eii X Vi

SVD based method explained

The reduced orthogonal dimensions resulting from SVD are less noisy than the original data and capture the latent associations

Reducing the dimensionality of the customer-merchant space, we can increase density of the matrix and thereby generate predictions

SVD provides the best lower rank approximations of the original matrix R, in terms of Frobenius norm

The reconstructed matrix Rk = Uk.Sk.Vk’ minimizes the Frobenius norm i.e. || R- Rk || over all rank-k matrices

Even though the Rk matrix is dense, its special structure allows us to use sparse SVD algorithms (e.g. Lanczos) whose complexity is almost linear to the number of nonzeros in the original matrix R.

SVD based method explained

We use SVD in recommender systems to perform 2 different tasks:-

First, we use SVD to capture the latent relationships between customers and merchants that allow us to compute the predicted likeliness of a certain merchant by a customer.

Second, we use SVD to produce a low-dimensional representation of the original customer-merchant space and then compute neighborhood in the reduced space. We then use that to generate a list of top-N product recommendations for customers.

SVD based method explained

The optimal choice of the value ‘k’ is critical to high quality prediction generation.

We are interested in a value of k that is large enough to capture all the important structures in the matrix yet small enough to avoid overfitting errors.

For an m x n matrix the SVD decomposition requires a time in the order of O((m+n)^3)

In terms of storage SVD is efficient. We need to store just two reduced customer and merchant matrices of size m x k and k x n respectively, a total of O(m+n), since k is constant.

Singular Value decomposition

If A is an $m \times n$ matrix with rank r

Then there exists a factorization of A as:

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m}\ \underbrace{\Sigma}_{m\times n}\ \underbrace{V^T}_{n\times n}$$

where U $(m \times m)$ and V $(n \times n)$ are orthogonal, and

Σ $(m \times n)$ is a diagonal-like matrix

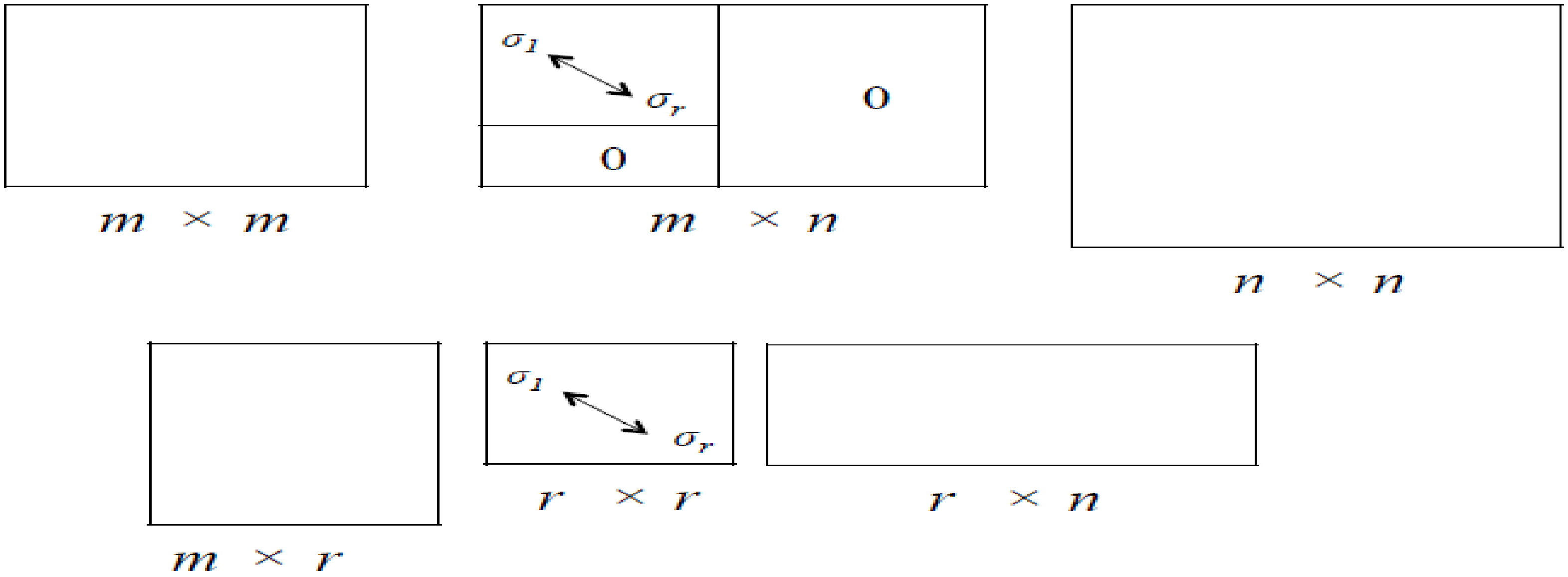
$\Sigma = (\sigma_{ij})$, where $\sigma_{ii} = \sigma_i$, for $i = 1, \ldots, r$ are the singular values of A, all non-diagonal entries of Σ are zero

$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$

Columns of U are the left singular vectors of A

Singular value decomposition

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m}\ \underbrace{\Sigma}_{m\times n}\ \underbrace{V^T}_{n\times n}$$



K-Rank Approximation

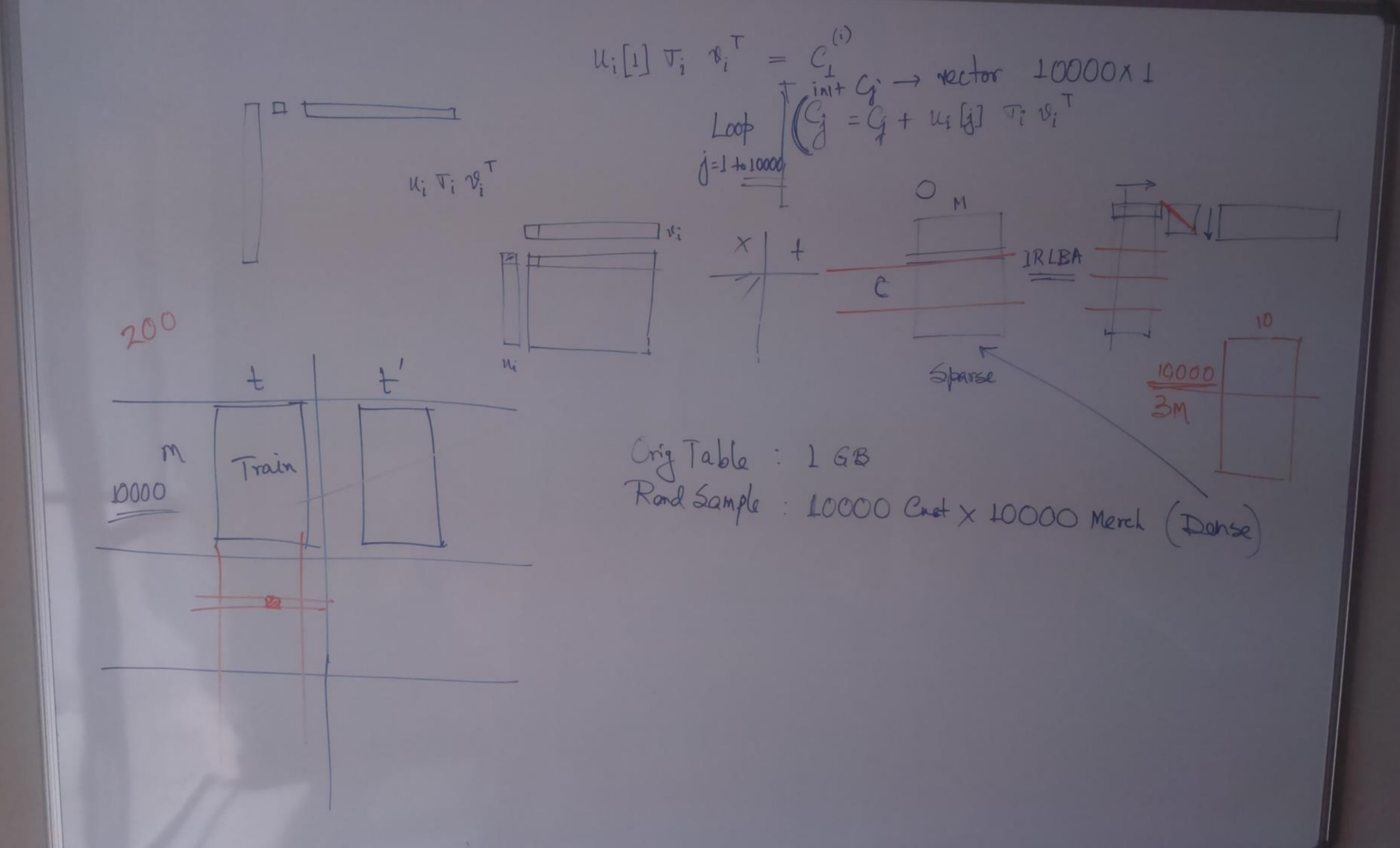
$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times m}\ \underbrace{\Sigma}_{m\times n}\ \underbrace{V^T}_{n\times n}$$

$$\Downarrow$$

$$\underbrace{A_k}_{m\times n} = \underbrace{U_k}_{m\times k}\ \underbrace{\Sigma_k}_{k\times k}\ \underbrace{V_k^T}_{k\times n}$$

Rank k

Still m dimensional vectors


$u_i[1] \sigma_i v_i^T = C_1^{(i)}$
init C_j → vector 10000 x 1
Loop
j=1 to 10000
$C_j = C_j + u_i[j] \sigma_i v_i^T$
$u_i \sigma_i v_i^T$
v_i
u_i
x
t
O
M
C
IRLBA
Sparse
10
10000
3M
200
t
t'
M
10000
Train
Orig Table : 1 GB
Rand Sample : 10000 Cust x 10000 Merch (Dense)

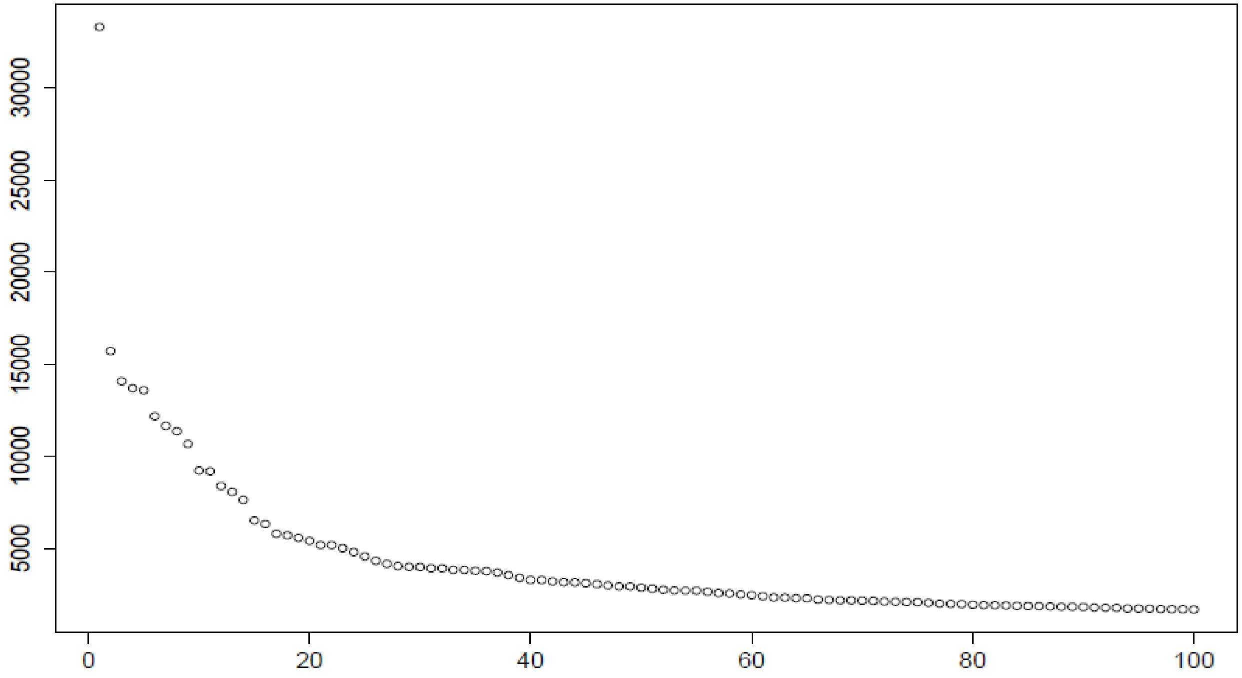
Approaches Tried (Continued...)

- Reconstruction of a row, representation of behaviour which is expected behaviour for a particular customer. This means the row we get after re construction gives an idea of, which merchants the customer would have bought normally.

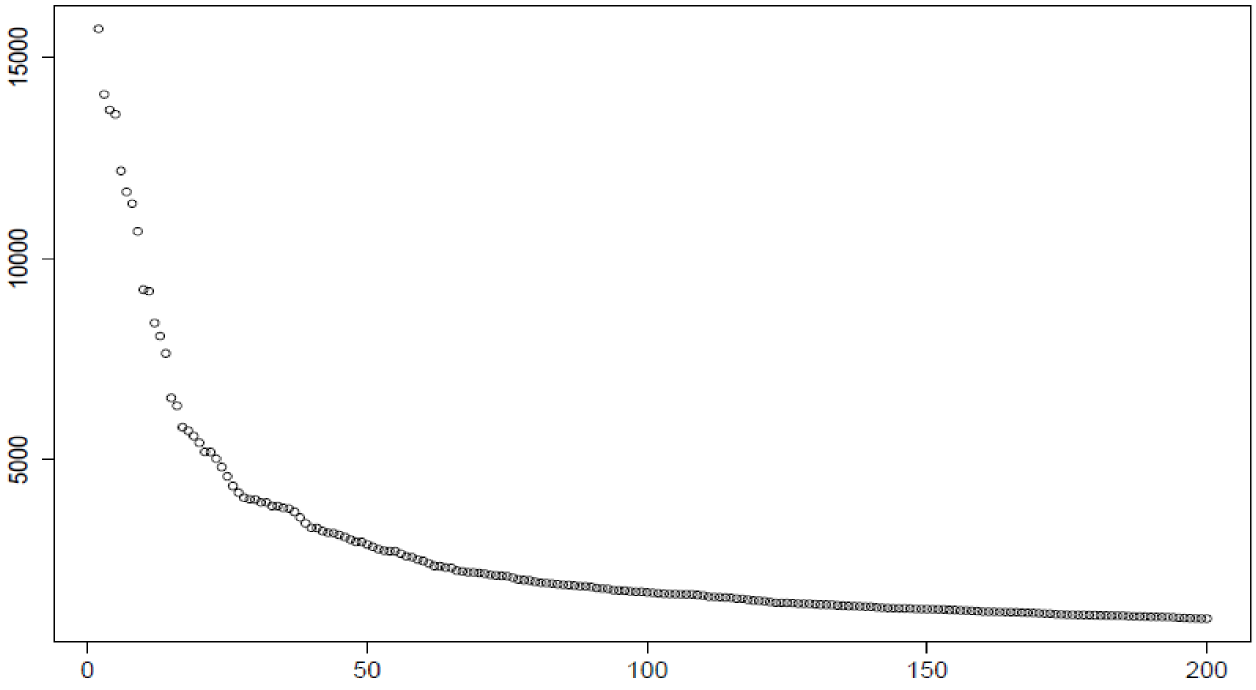
- So we take the difference of this reconstructed row from the original given row and see that, which are the entries in the row which changed from zero to nonzero.

- These elements will give the merchants from whom the customer are expected to buy. So we select the 10 merchants and recommend for that customer.

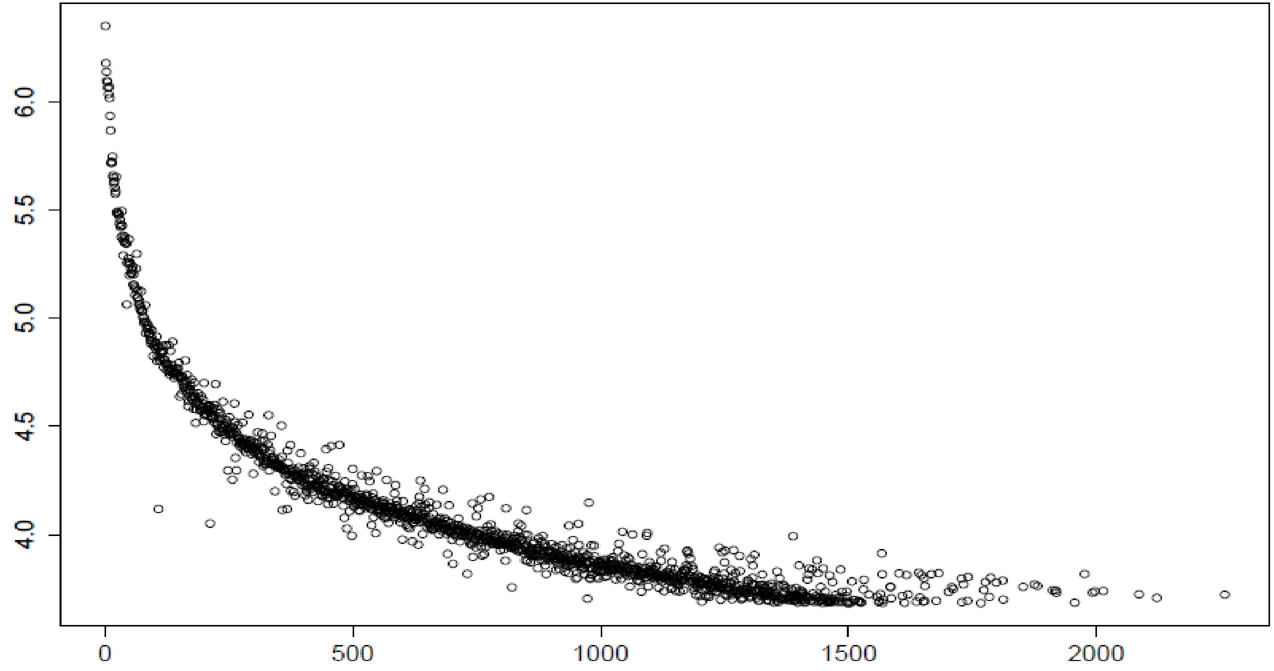
Plot of 1st 100 eigen values



Plot of 2nd to 200th eigen values



Log-plot of 1st 1500 Merchants by Transactions




IRLBA

What is IRLBA?

- The Implicitly Restarted Lanczos bidiagonalization algorithm (IRLBA) of Jim Baglama and Lothar Reichel is a state of the art method for computing a few singular vectors and corresponding singular values of huge matrices.

- With it, computation of partial SVDs and principal component analyses of very large scale data is very time and space -efficient. The package works well with sparse matrices.

SVD and IRLBA

- The singular value decomposition (SVD) of an m×n c matrix A can be written as $A = U\Sigma V^T$

- For Lanczos Bidiagonalization we consider a trimmed version

$$A = P_n B_n Q_n^T$$

Here B_n will of the special form:

$$\begin{pmatrix} \alpha_1 & \beta_1 & & & & & \\ & \alpha_2 & \beta_2 & & & & \\ & & \alpha_3 & \beta_3 & & & \\ & & & \alpha_4 & \beta_4 & & \\ & & & & & & \\ & & & & & \alpha_{n-1} & \beta_{n-1} \\ & & & & & & \alpha_n \end{pmatrix}$$

Bidiagonalization Algorithm

- Expressions can be written in vector form by equating the jth column
- $Aq_j = \beta_{j-1}\, p_{j-1} + \alpha_j p_j,$
- $A^T p_j = \alpha_j q_j + \beta_j q_{j+1}.$
- From This we can obtain the recursive relation:
- $\alpha_j p_j = Aq_j - \beta_{j-1} p_{j-1}$
- $\beta_j q_{j+1} = A^T p_j - \alpha_j q_j$

Partial Lanczos Bidiagonalization:Algorithm

- <u>Input:</u>
- A∈ $\mathbb{R}^{m\times n}$,the matrix whose SVD is to be carried out
- q1 ∈ $\mathbb{R}^{n}$:an initial vector of unit length ;
- n:number of bidiagonalization step

I. Q1← q_1 $p_1 \leftarrow Aq_1$

II. $\alpha_1 = \|p_1\|; p_1 \leftarrow {p_1}/{\alpha_1}; P_1 \leftarrow p_1$

III. for j=1:n

  I. $r_j \leftarrow A^T p_j - \propto_j q_j$

  II. $r_j \leftarrow r_j - Q_j Q_j^T r_j$

  III. if $j < n$

    I. $\beta_j \leftarrow \|r_j\|; q_{j+1} \leftarrow {r_j}/{\beta_j}$

    II. $Q_{j+1} \leftarrow [Q_j : q_{j+1}]$

    III. $p_{j+1} \leftarrow Aq_{j+1} - \beta_j p_j$

    IV. $p_{j+1} \leftarrow p_{j+1} - P_j P_j^{\ T} p_{j+1}$

    V. $\alpha_{j+1} \leftarrow \|p_{j+1}\|; p_{j+1} \leftarrow {p_{j+1}}/{\alpha_{j+1}}$

    VI. $P_{j+1} \leftarrow [P_j : p_{j+1}]$

Verifying for the algorithm

Sampling small chunk and then ensembling the data

For verifying the algorithm, took first 9 month data

Predicted for next 3 months

Compared with the data which we have.

Random samples of 10,000 customers were taken for each trial.

Keep taking samples and average if value already exists.

Cosine Similarity

- We are interested to find out similarity between two merchants. As the number of customers uncommon between two merchants Euclidean distance and Peasron correlation coefficient would not be effective

- We use cosine similarity metrics to determine similarity between two merchants.

- The cosine distance metric between two vectors $\underline{x}$ and $\underline{y}$ is calculated as follows:

- $Cosine\_similarity(\underline{x}, \underline{y}) = \dfrac{\underline{x} . \underline{y}}{\|\underline{x}\| \|\underline{y}\|}$

- Now we find out top ten neighboring merchants of each merchant

Generating Score function for each customer

- For a customer i we calculate the score vector by the following algorithm-
- For the (customer i, merchant j) pair
- If

  The customer i already made transaction with that merchant j then score for this pair would be 0

- Else
  - $\underline{s}$ = the top 10 similarity score for the merchant j
  - $\underline{h}$ = transaction history of customer i with the top-10 neighboring merchants of merchants j
  - The score function for the pair(customer i, merchant j)= $\frac{\|\underline{h}^T \underline{s}\|}{\|\underline{s}\|}$
  - Now for each customer we recommend the merchants whose have obtained highest scores for that customer

References

"Application of Dimensionality Reduction in Recommender System - A Case Study" Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John T. Riedl Department of Computer Science and Engineering, University of Minnesota

"Topics in Data Analytics: Collaborative Filtering, Finding Similar Items in Very High Dimension, Sentiment Analysis"
Debapriyo Majumdar, Indian Statistical Institute Kolkata

http://www.math.kent.edu/~reichel/publications/blklbd.pdf

http://qubit-ulm.com/wp-content/uploads/2012/04/Lanczos_Algebra.pdf


Questions?