D-035

次元圧縮を用いたクロスメディアレコメンデーション方式の提案

A Proposal for Applying Dimension Reduction in Cross-Media Recommendation

柳原 正†　　帆足 啓一郎†　　松本 一則†　　菅谷 史昭†

Tadashi Yanagihara　Keiichiro Hoashi　Kazunori Matsumoto　Fumiaki Sugaya

概要

ユーザの過去の履歴からユーザの嗜好に合わせたコンテンツを推奨するレコメンデーションシステムの多くは協調フィルタリングと呼ばれる方式を採用しており、ユーザの履歴情報を基にアイテム間相関行列を生成し、ユーザと相関が高いアイテムを推奨するという特徴を持つ。しかし、この方式では履歴情報がないユーザに対してレコメンド結果がうまく生成できないという欠点を持つ。

これまで、履歴情報を持たないユーザに対してレコメンデーションを行うための手法として、クロスメディアレコメンデーション方式を提案した[4]が、データ量の増大に伴い処理時間が増加するという課題があった。そこでシステム内で利用されるユーザ・アイテム間相関行列において次元圧縮を適用する改良モデルを提案した。評価実験を行った結果、アイテム数を 10%まで圧縮した際、精度を犠牲にせず、総処理時間を 30%縮めることができることができた。

1. クロスメディアレコメンデーション

クロスメディアレコメンデーション方式とは、推奨したいアイテムに関する履歴情報がないユーザにレコメンド結果を生成するために、別メディアのアイテムに関する履歴情報を分析し、似た行動を行ってきたユーザ同士を含んだ潜在クラスを形成し、相関が高い潜在クラス同士のアイテムをレコメンドする方式である。図 1 にシステム動作図を提示する。

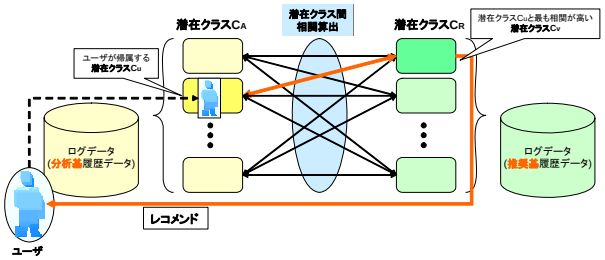

図 1. クロスメディアレコメンデーション方式の動作手順

† 株式会社 KDDI 研究所

クロスメディアレコメンデーション方式の動作手順は次の通りである。ユーザにレコメンドしたいアイテムに関する履歴情報である推奨基データと、それに対し、レコメンドを行うために学習するための履歴情報である分析基データの二種類を用いて、それぞれのデータに対し、各ユーザが応答したアイテムを特徴量とし、行動が起因した潜在的要因が共通だと思われるユーザ同士をグループ化した潜在クラスを抽出する。複数の潜在クラスが形成される際に、それぞれのユーザがどのクラスへ帰属するかについては EM アルゴリズムを用いた最尤推定で推測できる。[3]

ユーザの各潜在クラスへの帰属確率が求まると、次に分析基側の潜在クラスと推奨基側の潜在クラスにおいて、共通して現れる共通ユーザに着目し、それらのユーザに対する履歴情報をもとにアイテム間行列による協調フィルタを形成し、分析基データから抽出されたクラスと推奨基データから抽出されたクラスのそれぞれの相関関係を求める。相関関係が求まれば、推奨基データに履歴情報を持たないユーザに対し、レコメンデーション結果が生成可能となる。これは、履歴情報がないユーザが帰属する分析基側の潜在クラスと相関関係が高い推奨基側の潜在クラスを選択し、そのクラス内で最も人気が高いアイテムをレコメンドする。

数式で表すと次のように表現できる。L_a と L_r をそれぞれ分析基データ及び推奨基データから抽出された潜在クラス群と定義する。このとき、L_a と L_r を次の公式(1)及び公式(2)として表すことができる。

$$L_a = \{t_{a_1}, t_{a_2} \ldots t_{a_c}\} \ldots (1)$$

$$L_r = \{t_{r_1}, t_{r_2} \ldots t_{r_d}\} \ldots (2)$$

t_a と t_r はそれぞれ分析基データ及び推奨基データから抽出された潜在クラスそのものを表す。このとき、L_a と L_r はそれぞれ c 個と d 個の潜在クラスを含有する。

次に、潜在クラス t_{a_i} と潜在クラス t_{r_j} の間にある相関関係を計算するために、t_{a_i} 及び t_{r_j} 内に含まれる共通ユーザ数 n を n 次元のベクトルである $\vec{t}$ を次の公式(3)として表現できる。

$$\vec{t} = (q_t^{U_1}, q_t^{U_2}, \ldots q_t^{U_n}) \ldots (3)$$

公式(3)において、q_t^{Ui} はユーザ U_i が潜在クラス t に帰属する確率を表現する。潜在クラス間の相関係数はピアソン相関係数やコサイン類似度などの相関係数を用いて求めることができる。公式 (4) では、$\vec{t}_{a_i}$ と $\vec{t}_{r_j}$ 間のコサイン類似度の計算方式を表記している。

$$Sim(\vec{t}_a, \vec{t}_r) = \frac{\vec{t}_a \cdot \vec{t}_r}{\left|\vec{t}_a\right|\left|\vec{t}_r\right|} \ldots (4)$$

ユーザU_xに対しレコメンデーション結果を作成するためには、まず初めにそのユーザがL_a内のどの潜在クラスに対して帰属するかについては EM アルゴリズムを使って最尤推定を行う。L_aに含まれる潜在クラスのうち、帰属確率が最も高いクラスがt_{a_x}の場合、次にt_{a_x}に対し、最も相関関係が高い潜在クラスt_{r_x}が洗濯される。最後に、t_{r_x}内で最も人気が高いアイテム上位 n 件がユーザU_xに対し、レコメンデーション結果として提示される。

本方式の精度に関しては、クロスメディアレコメンデーション方式と協調フィルタリングを実装したプロトタイプで比較した結果、推奨したい特定ジャンルのアイテムに関する履歴情報がないユーザに対してレコメンデーション結果を生成した場合に、クロスメディアレコメンデーション方式の精度がより高いことがわかった。[4]

2. 次元圧縮適用の提案

クロスメディアレコメンデーション方式及び従来の協調フィルタリングの欠点として、ユーザ数とアイテム数の増加に伴い、潜在クラスを抽出するために必要とする計算処理時間が線形的に増加してしまう点が挙げられる。これは、ユーザ u 人とアイテム i 件の間で相関行列を生成する際に、$u*i$ 分だけの行列が生成され、これらの組み合わせを $(u*i)^2$ 分だけ計算しなければならない。アイテム間の相関行列を作成する手法[6]も提案されているが、これによって相関行列が i^2 に縮まるが、やはり新しいアイテムが追加にされる度に計算処理時間が線形的に増えてしまう。

このような大規模な行列では、特異値分解などによる行列の次元圧縮を行ったり、ユーザやアイテムをクラスタリングしたりすることで、計算処理時間を省略できることが一般的に知られている。そこで、本方式において、履歴情報から生成されたユーザ・アイテム間の相関行列に対し、特異値分解による次元圧縮を適用することを提案する。具体的には、以下の図 2 のとおりである。改良点として、離散化を行う前の段階に新しく次元圧縮モジュールを設置し、計算処理を行う点が挙げられる。

なお、次元圧縮を行う際の計算式は次の公式として表すことができる。ユーザ数 m に対し、アイテム n 件からレコメンドする場合を想定したとき、m 人のユーザが利用したアイテム n 件を表す行列 D は特異値分解により、以下の式 5 のように表現できる。

$$D = U \sum V \quad \ldots(5)$$

行列 D を K 次元の行列に圧縮する際に、U は $m * K$ の行列となり、V は $n * K$ の行列となり、Σ は $K * K$ のベクトル固有値を格納した行列となる。これらのうち、$m * K$ の行列である、全ユーザとそれに対する新しいアイテム K 次元の行列である U を元に、潜在クラスの抽出用のデータとして取り扱う。

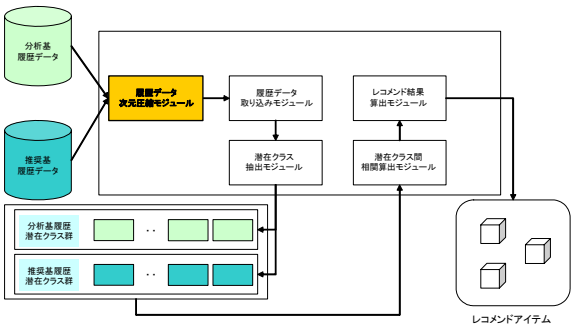

図 2. モジュール設計図

3. 評価実験

本システムへの改良を行うにあたり、その有効性を検証するために以下の二点について評価する。一点目は、処理しなければならない行列を圧縮することで、処理速度が向上する点である。二点目は、次元圧縮を行うことで、レコメンド結果に対する精度への影響である。

評価実験の概要として、評価用データとして GroupLens[2]プロジェクトで公開されている MovieLens[1]データセットを利用し、全ユーザの履歴情報を分析基用データと推奨基用データとして分ける。この場合、前者は MovieLens のデータセットにおいて最も人気があるジャンルである Drama のジャンルに該当する映画(1176 件)を分析基として用いて、Drama 以外のジャンルに該当する映画(2707 件)を推奨基として用いる。次に、サンプリング用として 10%のユーザを選び、それらのユーザが持つ推奨基データに含まれる履歴情報を取り除く。推奨基データの履歴情報と分析基データの履歴情報を基にレコメンド結果を算出したあと、サンプリング用として選び出したユーザに対するレコメンド結果と、取り除いたサンプリング用ユーザの履歴情報を比較し、一致した値をそれらのユーザに対しレコメンドした結果を割った結果を Precision として、レコメンデーションの精度を計測する。

なお、GroupLens プロジェクトにて公開されているデータでは、100,000 data set と 1 million data set と呼ばれるデータセットが二種類公開されているが、これらのうち、よりデータ容量が大きい 1 million data set のものを使用する。このデータの特徴として、総ユーザ数は 6040 人、総アイテム数は 3883 件、履歴数は 1000209 件である。

それぞれのレコメンデーションエンジンにおいて，ユーザ一人に対して生成するレコメンデーション結果の件数を 10 件、分析基データと推奨基データの潜在クラス数をそれぞれ 10 個とし、相関係数はコサイン類似度を用いた。また、次元圧縮の機能を実装するにあたり、SVDPACK[5]を利用した。また、次元圧縮を行うためのアルゴリズムとして、収容効率と処理速を考慮して、las2 アルゴリズムを用いた。

4. 結果と考察

評価項目として次元圧縮の比率と計算処理時間の関係、及び次元圧縮の比率と精度の関係に着目する。図 3 ではこれらのうち、次元圧縮の比率と計算処理時間の関係を表す。横軸は SVD の次元圧縮の比率を表す。0 のときは圧縮は行っていない状態を表し、次にデータを 90%に圧縮した状態、それ以降は 10%ずつ圧縮の比率を高め、最終的にもとのデータの 10%になるまで圧縮を行った。一方、縦軸ではユーザ・アイテム間行列を生成してからレコメンド結果を生成するまでの計算処理時間(秒数)を表す。

計算処理時間は総計算処理時間及び潜在クラスを抽出するために必要とする時間の二点で比較を行う。

まず初めに総計算処理時間について着目する。以下の図 3 で次元圧縮の比率と総計算時間の比例関係を示す。

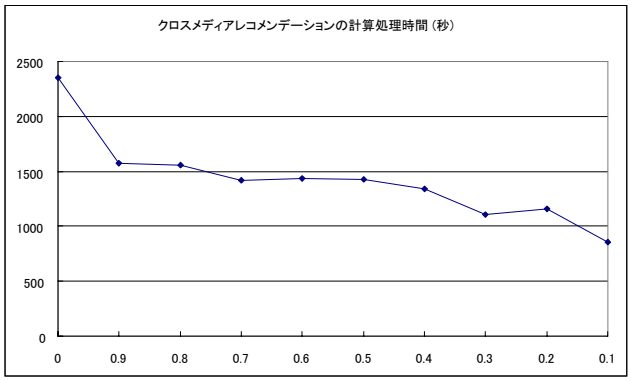

図 3: 次元圧縮の比率と総計算処理時間に関する図

データを 90%に圧縮することで潜在クラスを抽出する時間を 64%に縮めることができ、圧縮率を 40%まで引き下げるにつれて、総計算処理時間は緩やかに減少した。なお、総計算処理時間の内訳として、履歴情報からユーザ・アイテム間の相関行列を生成するのは分析基データ及び推奨基データの分を合わせて約 710 秒かかるため、これらの値を引くと潜在クラスを抽出するための時間を求めることができる。

さらに分析を進め、分析基データと推奨元データの次元圧縮の比率を 10%から 90%のときのレコメンデーションの精度との組み合わせを計算した。これらの結果を表 1 に示す。横は分析基データの圧縮比率、縦は推奨基データの圧縮比率を表し、表内の各値は精度(Precision)を表す。評価実験を行った結果、分析基データに対し次元圧縮を行った場合において、行わない場合に対して 2～3%の精度改善(分析基データの次元圧縮の比率が 0.9 のとき)が見られたものの、全体的に飛躍的に精度が高まる条件を発見することができなかった。これは、測定した比率の範囲内で次元圧縮を行う分では精度に対する影響力はほぼないことと考えられる。これにより、次元数が少なくなることに対する精度への影響力が少ないため、計算処理時間を減らす上で次元圧縮は有効であると言える。

5. まとめ

本論文において、潜在クラスモデルを使ったレコメンデーションシステムにおいて、ユーザ・アイテム間行列において次元圧縮を行った際、精度の低下なく、処理時間を短縮できることがわかった。今後、さらに圧縮の比率が高い状態での測定を行い、精度と計算処理時間に対する影響力について検証を行う。

6. 参考文献

[1] “MovieLens” http://movielens.umn.edu/
[2] “GroupLens Project” http://www.cs.umn.edu/Research/GroupLens/
[3] 岡太彬訓，木島正明，守口 剛, “マーケティングの数理モデル” , 朝倉書店, 2001
[4] 柳原正, 帆足啓一郎, 松本一則, 菅谷史昭, “潜在クラスを利用したクロスメディアレコメンデーション方式の提案”, 第 68 回全国大会, 2006, No.3, pp.1-2
[5] “SVDPACK” http://www.netlib.org/svdpack/
[6] M. Deshpande and G. Karypis, “Item-based Top-N Recommendation Algorithms”, ACM Transactions on Information Systems (TOIS), 2004

	0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
0	0.450514	0.443984	0.447616	0.441416	0.447395	0.447872	0.432099	0.445763	0.461811	0.471156
0.9	0.44325	0.448643	0.451798	0.451798	0.45088	0.451027	0.461212	0.445763	0.461811	0.471156
0.8	0.450697	0.45077	0.453265	0.453265	0.452531	0.453522	0.463424	0.445763	0.461811	0.471156
0.7	0.447469	0.448943	0.453668	0.452825	0.452238	0.451541	0.463424	0.445763	0.461811	0.471156
0.6	0.447395	0.447799	0.452128	0.454916	0.452971	0.453815	0.461212	0.445763	0.461811	0.471156
0.5	0.447872	0.447359	0.448606	0.449413	0.447726	0.448716	0.461212	0.467608	0.461811	0.471156
0.4	0.451577	0.451064	0.452128	0.452825	0.451321	0.453815	0.461212	0.465537	0.461811	0.471156
0.3	0.44945	0.45088	0.450477	0.452531	0.45077	0.450037	0.46066	0.471751	0.461811	0.471156
0.2	0.448496	0.448899	0.449193	0.450734	0.448973	0.449046	0.461212	0.471751	0.461811	0.471156
0.1	0.447542	0.447432	0.448753	0.449596	0.447726	0.445677	0.459923	0.470621	0.461625	0.471579

表 1. 分析基と推奨基の圧縮比率と精度の関係図