レコメンドシステムのアルゴリズムについて

提出者：大阪大学基礎工学部情報科学科 津久井 佑樹
メールアドレス：menoshitanoaka@gmail.com

提出年月日：平成 28 年 7 月 11 日

1　実装するアルゴリズムについて

レコメンドシステムを実現するために有効と思われるアルゴリズムとして NMF(Non-Negative Matrix Factorization) をあげる．これはある行列 R が存在したときに $R = U^T V$ となるような U, V を求めるためのアルゴリズムである．レコメンドシステムに適用する場合，ユーザーの数を m，ユーザーが評価したアイテムの総数を n，R を $m \times n$ 行列 (ユーザー i のアイテム j に対する評価点を a としたとき, $r_{ij} = a$)，U,V をそれぞれ $K \times m$ 行列,$K \times n$ 行列 ($\exists K << m, n$) とする．このとき U の列ベクトルはそれぞれのユーザーの特徴を抽出したベクトルとなり，V の列ベクトルはそれぞれのアイテムの特徴を抽出したベクトルとなる．また $U^T V$ から生成される行列は R の要素が 0 だった点が別の値に置き換えられており，この値を未評価のアイテムに対する評価の予測値として扱うことができる．これを用いてユーザーに対してレコメンドを行う．

行列 R を U と V に分解する際には確率的勾配降下法を用いる．このアルゴリズムは初めにランダムに生成した U と V に対して，R と $U^T V$ の二乗誤差が最小になるように修正を繰り返すものである．今回実装する際には純粋な二乗誤差ではなく

$$f(U, V) = \frac{1}{2} \sum_{(i,j) \in D} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \frac{\lambda}{2} (||U||_F^2 + ||V||_F^2)$$

を最小にする U と V を求める．$(i, j) \in D$ は評価済みのユーザーとアイテムの集合を表し λ は正則化パラメータ，$(||U||_F^2 + ||V||_F^2)$ は正則化項である．U,V は以下の式

$$\mathbf{u}_i = \mathbf{u}_i - \eta\{-(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)\mathbf{v}_j + \lambda \mathbf{u}_i\}$$

$$\mathbf{v}_j = \mathbf{u}_i - \eta\{-(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)\mathbf{u}_i + \lambda \mathbf{v}_j\}$$

で更新する．η は学習率を表し，

$$\eta = \frac{1}{\alpha(t + \beta)}$$

を用いて更新する．t は反復回数を表し，α,β はパラメータである．これらの更新は規定回数を上限として更新前と更新後の $f(U, V)$ の差が一定値一定値以上の際に繰り返し行う．U と V がそれぞれ求まったら以下の式

$$\max_{(i,j) \in E} \mathbf{u}_i^T \mathbf{v}_j$$

からユーザー i の未評価アイテム j の予測値の最高値が得る．この j を i に対してレコメンドする．なお，$(i, j) \in E$ は未評価のユーザーとアイテムの集合を表す.

今回実装する際に各パラメータは $K = 30, \lambda = 0.1, \alpha = 0.05, \beta = 500$ とし，更新の上限回数を 5000，更新前と更新後の $f(U, V)$ の差が 0.0001 以上の場合に更新を繰り返すものとした．また，以上で用いた全式は [1] を参考にした．

2　実装した手法の改善点

第一の改善点として実装した手法ではユーザーの評価に関する時間的な特性を考慮に入れていない点が挙げられる．与えられたデータセットにはユーザーが評価を行った際のタイムスタンプが含まれているが，それが結果の導出過程で一切触れられていないため時間の経過等がユーザーの評価に与える影響を反映できていない．また，結果の導出に関わらないため結果に対して実際にそれが影響を与えているかの判断も不可能である．[2] によるとユーザーが初めて評価をしてから時間が

経過するごとに厳しく評価するようになる傾向があると述べられているため，それを手法に取り入れて実際にその傾向を示すか確かめる必要があると思われる．

第二に処理時間が長いという点が挙げられる．今回実装したプログラムは Python3.51 で記述しており，私の実行環境 (Windows10Pro，Intel(R)Core(TM)i7-4500U CPU @1.80GHZ 2.39GHz，RAM4.00GB) において与えられたデータセットから学習を行い，確率的勾配降下法を用いての更新を 5000 回した場合には約 16 時間ほど必要とした．これは後述の改善点にも影響を与える問題のため，C 言語等の処理が速い言語に書き換える，アルゴリズムを改善する等の必要がある．また，η 等のパラメータは結果の収束速度に影響を及ぼすためこれらを修正することによる改善も可能であると考えられる．

第三にデータとして既存のユーザーやアイテムに対するレコメンドは行えるが新規の物に対してそれを行えないという点が挙げられる．実装したアルゴリズムでは既存のユーザー，アイテム，評価点からなる行列をユーザーとアイテムそれぞれの特徴ベクトルに分解することでレコメンドを行っている．対して，新規のユーザーやアイテムに関しては特徴ベクトルが不明なため元のデータに加えてから再計算を行わない限りレコメンドできない．しかし，前途のように一度の学習に非常に時間がかかるため計算し直す際のコストが高い．

第四に導出結果の妥当性が示されていない．実装したプログラムでは $\frac{1}{2}\sum_{(i,j)\in D}\frac{1}{2}(r_{ij}-\mathbf{u}_i^T\mathbf{v}_j)^2$ の値を求めているが，これだけでは妥当性を示すものとして不十分である．改善策のひとつとして交差検証が挙げられるが前途のように一度の処理に時間がかかるため交差検証の結果の出力にはさらなる時間がかかることが予想される．

第五に 1 人のユーザーに対して 1 つのアイテムしかレコメンドしていない点が挙げられる．レコメンドされたアイテムに対するユーザーの反応としてユーザーがアイテムに興味を持ち実際に楽しめた場合，ユーザーが興味を持つも楽しめなかった場合，ユーザーが興味を持たなかった場合が挙げられる．一般にレコメンド数を増やすと，ユーザーがアイテムに興味を持ち実際に楽しめる場合が増えると予想されるが同時にその他の場合も増えると予想される．それぞれに場合についてユーザーがシステムについてどれほどの好印象を抱いたか，あるいは悪印象を抱いたかのデータを収集することによってレコメンド数として妥当な値の算出ができるものと思われる．例えば，ユーザーがレコメンドされたアイテムに興味を示さなかった場合のシステムに対する悪印象が少ないと予想される場合は，興味を示さないようなアイテムが多く含まれると予想されてもレコメンド数を増やすべきである．

参考文献

[1] 推薦システム / 上嶌 敏弘
''http://www.kamishima.net/archive/recsys.pdf''
2016 年 7 月 10 日にアクセス

[2] Winning the Netflix Prize: A Summary / Edwin Chen
''http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/''
2016 年 7 月 10 日にアクセス

[3] Introduction to Algorithms for Behavior Based Recommendation / Kimikazu Kato
''http://www.slideshare.net/hamukazu/introduction-to-behavior-based-recommendation-system''
2016 年 7 月 10 日にアクセス

[4] Collaborative Filtering for Implicit Feedback Datasets / Yifan Hu, Yehuda Koren, Chris Volinsky
''http://yifanhu.net/PUB/cf.pdf''
2016 年 7 月 10 日にアクセス