ADMM SLIM における L0 正則化項の有効性

桜木 佑記

1 はじめに

最近サービスでは, 数多く存在する各ユーザーに対してそのユーザーが好むであろうアイテムを推薦することでアプリケーションの利便性を向上させることが求められる. そのような top-N 推薦を行う手法の一つとして Sparse Linear Method(SLIM)[1] という手法がある. これは数多くの論文の中でその有用性は示されている手法であるが, 一方で SLIM のモデルは学習に要する計算コストが高いことも知られている.

その問題を解消するための手法として, 訓練データにおけるユーザー数はとても大きく, 一方でアイテム数は比較的少ないといった現実世界によくあるシナリオを前提として,Alternating Directions Method of Multipliers(ADMM)[2] という最適化手法を SLIM の目的関数の最適化に用いることで, 計算コストを下げてかつ推薦精度も上がるという ADMM SLIM という研究 [3] がある.

問題提起として目的関数にかかる正則化項や制約条件を一部変更することで、精度や計算コストがどのように変化するのか調べることを考えた. そこで提案手法として従来法にて目的関数にかかっていた L1 正則化項を L0 正則化項に置き換えることを考える.L1 正則化を使う場合はプログラム内部でソフト閾値処理を行い, 代わりに L0 正則化を使うのであればソフト閾値処理の代わりにハード閾値処理という処理を行うことで実現ができる. このようにすることで L1 正則化と L0 正則化の比較検証を行う.

2 従来手法

本章では従来法にて使われていた L1 正則化項を使った最適化手法の計算と, 提案手法である L0 ノルムを使った最適化手法の計算の計算手順の違いについて述べる.

2.1 アルゴリズム

SLIM の ADMM を用いた最適化のアルゴリズムの概要を Algorithm1 に示す.

Algorithm 1 :ADMM Training in python2.7

Input: User-item interaction-matrix X
Regularization parameters lambda1,lambda2>0
ADMM penalty parameter rho>0
Number of ADMM iterations K
Output: Item-item weight-matrix C

#pre-compute
$XtX = X.T.dot(x)$
$diag\_indices = numpy.diag\_indices(XtX.shape[0])$
$XtX[diag\_indices] += lambda2 + rho$
$P = numpy.linalg.inv(XtX)$
$XtX[diag\_indices] -= lambda2 + rho$
$B\_aux = P.dot(XtX)$
#iterate until convergence
for _ in range(K): **do**
  $B\_tilde = B\_aux + P.dot(rho * C - Gamma)$
  $gamma = numpy.diag(B\_tilde)/numpy.diag(P)$
  $B = B\_tilde - P * gamma$
  $C = softthreshold(B+Gamma/rho, lambda1/rho)$
  $C = numpy.maximum(C, 0.)$
  $Gamma += rho * (B - C)$
end for
return C

オリジナルの SLIM 最適化問題 ユーザーとアイテム, そしそのユーザーがアイテムにつけた点数からなるデータセットがあるとし, 何点以上ならそのユーザーとアイテムに関係があるか基準を設ける. そして関係があるなら 1, そうでないなら 0 の値を持つユーザー-アイテム行列を生成する. この暗黙のフィードバックデータであるユーザー-アイテム行列 $\boldsymbol{X} \in \mathbb{R}^{|\boldsymbol{U}|\times|\boldsymbol{I}|}$ とし, また求めたい行列をアイテム-アイテム 重み行列 $\boldsymbol{B} \in \mathbb{R}^{|\boldsymbol{I}|\times|\boldsymbol{I}|}$ としてオリジナルの SLIM 最適化問題は次の式で表せられる.

$$\begin{aligned} \min_{\boldsymbol{B}} \quad & \frac{1}{2} \cdot ||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{B}||_F^2 + \frac{\lambda_2}{2} \cdot ||\boldsymbol{B}||_F^2 + \lambda_1 \cdot ||\boldsymbol{B}||_1 \\ \text{s.t.} \quad & \text{diag}(\boldsymbol{B}) = 0 \quad \boldsymbol{B}_{i,j} \geq 0 \quad \forall_{i,j} \in \boldsymbol{I} \end{aligned} \tag{1}$$

2.2 L1 正則化項の除去

オリジナルの SLIM 最適化問題式 (1) の L1 正則化項を取り除いた場合, 変更される Algorithm1 は 13 行目の箇所だけであり, $C = B$ と変更すれば実現できる.

2.3 非負制約条件の除去

オリジナルの SLIM 最適化問題式 (1) の非負制約条件を取り除いた場合, 変更される Algorithm1 は 14 行目の箇所だけであり, この行を取り除いてしまえば実現できる.

2.4 L1 正則化項と非負制約条件の除去

オリジナルの SLIM 最適化問題式 (1) の L1 正則化項と非負制約条件を取り除いた場合, ADMM での繰り返しは必要なく, 解は閉形式で求まる.$P = (X^T X + \lambda_2 \cdot I)^{-1}$ として, ゼロ対角条件を持つ閉形式解 B^{dense} は次の式で求まる.

$$B^{\text{dense}} = I - P \cdot \text{diagMat}(\mathbf{1} \oslash \text{diag}(P)) \tag{2}$$

ここで diagMat($\mathbf{V}$) はベクトル $\mathbf{V}$ を対角成分に持つ行列のことで,$\mathbf{1}$ はすべての要素が 1 であるベクトルを表している.

3 提案手法

オリジナルの SLIM 最適化問題式 (1) の L1 正則化項を L0 正則化項に変換する. このとき変更されるアルゴリズム algorithm 1 は 13 行目の箇所であり, この行を $C = hardthreshold(B + Gamma/rho, lambda1/rho)$ と変更すれば実現できる.

hardthreshold の定義 hardthreshold(**A**,threshold) は行列 A の各成分 $A_{i,j}$ に対して, $A_{i,j}^2 \geq threshold$ なら $A_{i,j} = A_{i,j}$, そうでないなら $A_{i,j} = 0$ という処理を行うものである.

4 実験

4.1 データセット

本実験では 2 つのデータセット Anime Recommendations Database[1] と [4] でも用いられた MovieLens 20M Dataset[2] の二つのデータセットを利用して実験を行った.Anime Recommendations Database では user_id と anime_id, それと user がその anime につけた点数 (rating) からなる csv ファイルから, 暗黙のフィーバックデーター行列であるユーザー-アイテム行列 X を作る.X はバイナリの行列 (0 or 1) であり,rating が 1 から 10 の値ならばユーザーがそのアニメを視聴しているので関係があると評価して 1 の値を,rating の値が-1 であればユーザーがアニメを視聴していないので関係がないと評価して 0 の値を入れることで行列 X を生成する. このときにマ

[1] https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database
[2] https://grouplens.org/datasets/movielens/20m/

イナーなユーザーとアニメを取り除くために,10 個以上のアニメを見ていないユーザーと,30 以上のユーザーに視聴されていないアニメはあらかじめ取り除いてから行列 X を生成した.

MovieLens 20M Dataset では, 同様に userId と movieId,rating から成る csv ファイルからバイナリであるユーザー-アイテム行列 X を作成し,rating が 1 以上 4 未満であればユーザーがムービーを低く評価しているので関係がないと判断して 0 の値を,rating が 4 以上 5 以下であればユーザーがムービーを高く評価しているので関係があると判断して 1 の値を入れる. このとき前者と同様にマイナーなユーザーとムービーを取り除いておくために,5 個以上のムービーを見ていないユーザーと 15 人以上のユーザーに見られていないムービーはあらかじめ取り除いてから行列 X を生成した.

4.2 実験方法

先ほどの基準でデータセットから生成した暗黙のフィードバックデータである行列 X を用いてプログラム上で algorithm 1 を実行して式 (1) におけるアイテム-アイテム重み行列 B を求める. そして行列 X と重み行列 B とで内積をとり, 予測スコアを表す行列 X·B を得る. これを各ユーザーに対しスコアの高い順に並び替えることで top-N 推薦を行う. その際に使用するハイパーパラメーターは $\lambda_1 = 5, \lambda_2 = 1000, \rho = 10000$ として, ADMM の繰り返しは収束条件を満たすまで行う. また top-N 推薦における N は N=7 として, 上位 7 つのアイテムを推薦するようにした.

4.3 実験結果

従来手法と提案手法による実験の結果を表 1,2 に示す.

表 1: Anime Recommendations Database を使った実験結果

手法	hit	recall	MAP	学習時間 [s]
$ADMM \geq 0\&L1$	**0.991**	0.11869	0.88470	76.358
$ADMM \geq 0$	**0.991**	0.11845	0.88408	70.019
$ADMML1$	0.990	**0.11906**	**0.88869**	89.671
Dense	0.986	0.10225	0.79936	**7.8436**
$ADMM \geq 0\&L0$	0.990	0.11758	0.88361	179.14
$ADMML0$	0.990	0.11760	0.88414	295.38

表 2: MovieLens 20M Dataset を使った実験結果

手法	hit	recall	MAP	学習時間 [s]
$ADMM \geq 0\&L1$	0.919	0.10848	0.63485	118.8
$ADMM \geq 0$	0.914	0.10756	0.62677	137.33
$ADMML1$	**0.922**	**0.11135**	**0.65240**	140.76
Dense	0.907	0.09909	0.59548	**27.805**
$ADMM \geq 0\&L0$	0.911	0.10582	0.62665	162.17
$ADMML0$	0.917	0.10674	0.62882	350.85

表中の $ADMM \geq 0\&L1$ は目的関数に非負制約条件と L1 正則化項がかかっているもの, $ADMM \geq 0$ は非負制約条件のみがかかっているもの,$ADMML1$ は L1 正則化項のみがかかっているもの, Dense は両方かかっていないものを表している. 同様に $ADMM \geq 0\&L0$ は目的関数に非負制約条件と L0 正則化項がかかっているもの, $ADMML0$ は L0 正則化項のみかかっているものを表す.

4.4 評価指標

top-N 推薦の結果と, 実際のユーザーとアイテムの関係を表す行列 X を比較することで精度を算出する. 評価指標には hit,recall,MAP の三つがある.

hit は各ユーザーに対して推薦された上位 7 つのアイテムのうち, 行列 X と比較して実際にユーザーと関係のあるアイテムが 1 つでもある (hit する) のなら 1, そうでない (hit しない) ならば 0 としたときの全ユーザーでの平均値である.

また recall の値が全体的に低い理由としては, このプログラム上での recall は各ユーザーに対してユーザーと関係のある (1 の値の) アイテムの数のうち,top-N 推薦されたアイテム数の割合なので, 分母がユーザーごとに異なり 11,28,50,190 など大きい値をとることがあるのに対して, 分子は必ず N 個以下になる (例えば N = 7) ので値が低くなると考えられる. よってここでは N の値は 7 として実験を行ったが, この N の値を大きくすればするほど recall の値は大きくなっていく.

MAP は様々なユーザーに対する AP の平均 (Mean) である.AP とは Average Precision の意味でその順位までにおける正解率を各データで平均をとったものである. 例えばあるユーザー u に対する top-7 推薦されたアイテムの結果が, 実際に行列 X にて 1 の値となっているアイテムを true, そうでない場合 False として [T,F,F,T,F,T,T] となっているとする (左から順に 7 位,6 位,...,1 位). その順位までにおける正解率は 1 位から順に数えていくので $[\frac{4}{7}, \frac{3}{6}, \frac{3}{5}, \frac{3}{4}, \frac{2}{3}, \frac{2}{2}, \frac{1}{1}]$ となり,AP はこれの平均であるので $(\frac{4}{7}+\frac{3}{6}+\frac{3}{5}+\frac{3}{5}+\frac{3}{4}+\frac{2}{3}+\frac{2}{2}+\frac{1}{1})/7 = 0.726...$ となる. これを全ユーザーで平均をとったものが MAP となる.

4.5 精度比較

まず $ADMM \geq 0\&L1$ と $ADMM \geq 0$ を比べてみて, 精度はほとんど変わりがないことが分かる. つまり L1 正則化項はあったとしても精度をほとんど向上させないということである.

また $ADMM \geq 0\&L1$ と $ADMML1$ を比べてみると, 後者のほうが全体的に精度が上がっていることが分かる. つまり, 非負制約条件はむしろ精度を悪くしてしまうということである.

L1 正則化項と非負制約の両方を除去して学習を行った Dense は $ADMM \geq 0\&L1$ と比べて精度はそこまで落ちておらず, また ADMM の繰り返しが必要ないので学習にかかる時間が非常に短いことが分かる.

L1 正則化項を使った $ADMM \geq 0\&L1$ と L1 正則化項の代わりに L0 正則化項を用いた $ADMM \geq 0\&L0$ を比べてみると, どちらのデータセットにおいても精度はそこまで変わりはないが, 前者に比べて後者のほうは学習にかかる時間がかなり長くなっている. また $ADMML0$ は $ADMM \geq 0\&L0$ と比べて学習にかかる時間がずっと長いこともわかる.

5 まとめと今後の展望

今回行った実験では，従来手法の ADMM SLIM において L1 正則化項の代わりに L0 正則化項を用いて精度の向上を図る手法を提案したが，精度の向上は見られず, むしろ学習時間が長くなってしまった．今後は, 従来手法が用いた L1 正則化と今回用いた L0 正則化について，それぞれに適したユースケースを解明することが課題となる．

参考文献

[1] Ning, X., Karypis, G.: SLIM: Sparse LInear Methods for top-N recommender systems, *Proceedings - 11th IEEE International Conference on Data Mining, ICDM 2011*, pp. 497–506 (2011).

[2] Boyd, S., Parikh, N., Chu, E., Peleato, B. and Eckstein, J.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends® in Machine Learning*, Vol. 3, No. 1, pp. 1-122 (2011).

[3] Steck, H., Dimakopoulou, M., Riabov, N. and Jebara, T.: ADMM SLIM: Sparse Recommendations for Many Users, *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, p. 555–563, New York, NY, USA, Association for Computing Machinery (2020).

[4] Liang, D., Krishnan, R. G., Hoffman, M. D. and Jebara, T.: Variational autoencoders for collaborative filtering, *Proceedings of the 2018 world wide web conference*, pp. 689–698 (2018).