非負値スパーステンソル因子分解における Multiplicative 更新手法の高速化

松林 達史* 澤田宏†

日本電信電話株式会社 NTT サービスエボリューション研究所／ NTT 機械学習センター

1 はじめに

近年データ分析技術において，再び有用性が再確認されている技術に非負値テンソル因子分解 (Nonnegative Tensor Factorization, NTF) がある．この NTF とは非負値行列因子分解 (Nonnegative Matrix Factorization, NMF) のテンソル分解拡張技法である．NTF のアルゴリズム部分には，行列テンソル積が用いられ，ベクトルの内積和を高頻度に繰り返し演算を行う．NTF は非常に有要な手法である一方，その計算量は非常に高く，例えば 3 次のテンソル分解では，テンソル $\boldsymbol{X} = [x_{ijk}] \in \mathbb{R}_+^{I \times J \times K}$ をランク R に CP 分解する場合は，$O(IJKR)$ の計算量が必要である．また $O(IJK)$ のメモリ空間が必要な事により，現実的には $1000 \times 1000 \times 1000$ 程度のデータ規模の分解に留まり，高速化技術の要求は高まっている．

NMF や NTF は計算量が多い反面，テンソル積など処理が非常に単純で，並列処理による高速化が有効である．実際 NMF や NTF の高速化の先行研究では [1, 2]，GPU を用いて 100 倍以上の高速化を実現している．しかしながら，画像処理や音声認識と異なり，購買ログや NW システムログと言ったデータマイニングにおけるログデータの大半は疎（スパース）として扱うことが可能であり，密テンソルの計算処理を必要としない．例えばデータログ数が L である場合，テンソルのデータ構造は $I \times J \times K$ であっても，通常はログの要素数 L のみ利用すればよく，$O(LR)$ の計算量のみで処理が可能である．また，スパーステンソルデータの分析では，理論的にはメモリ空間も $O(L)$ 程度に抑えることが可能であり，現実的にも大規模なマイニング処理が可能となる．

そこで本研究では，スパーステンソルデータに対して，各モードに対して疎行列展開した二次元データを利用し，演算処理の最適化を行い，スパーステンソル因子分解の高速化を実現した．本研究では，CP 分解および Tucker 分解の両分析手法に適用し，Multiplicative 更新法による高速化実装の例を示す．

2 Sparse Nonnegative Tensor Factorization(SNTF)

本研究で用いるテンソルデータとは，例えば購買ログなどでは「誰が？ (UseID) いつ？ (Time) 何を買ったか？ (Item)」といったように，一つのログに対し複数の特徴量（モード）があるデータを示し，一般に，N 個のモードを持つテンソルデータを，N 次のテンソルデータという．例えば，非負値の実数値をもつ 3 次のテンソルデータは $\boldsymbol{X} = [x_{ijk}] \in \mathbb{R}_+^{I \times J \times K}$ と表す事ができる．ここで I, J, K は各モード要素数で，$\mathbb{R}_+$ は非負の実数値を表す．代表的なテンソル因子分解手法の一つである CP 分解では，$\boldsymbol{X}$ を基底数 R 個の因子に分解する．この時，各因子と要素の関係は因子行列として表現することができ，各モードの要素因子行列 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ はそれぞれ $\boldsymbol{A} = [a_{ir}] \in \mathbb{R}_+^{I \times R}$, $\boldsymbol{B} = [b_{jr}] \in \mathbb{R}_+^{J \times R}$, $\boldsymbol{C} = [c_{kr}] \in \mathbb{R}_+^{K \times R}$ として表せる．因子行列 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ のテンソル積を $\hat{\boldsymbol{X}} = \sum_r^R a_{ir} b_{jr} c_{kr}$ とした時，$\hat{\boldsymbol{X}}$ が $\boldsymbol{X}$ と極力等しくなるように因子行列を求める．これら更新式は，$O(IJKR)$ の計算量が必要となる．

しかしながら，購買ログなどの一般的なログでは，「ユーザ i が，いつ j，アイテム k を購入したか」というデータ形式を用いるが，時間情報の粒度やアイテムの総和数によって，非常にスパースなデータになる．スパースログが $(i, j, k) \in \boldsymbol{L}$ として，ログ数 L $(= |\boldsymbol{L}|)$ で表される時，通常のテンソル因子分解ではなく，スパーステンソル因子分解を用いることによって高速に処理が可能となり，スパーステンソル因子分解における因子行列の更新には，$O(LR)$ の計算量に抑えることが出来る．

3 SNTF の高速化と因子別データ構造

テンソルを扱う際，一般的には高次の配列として扱われるため，単純なデータ構造ではランダムメモリアクセスが生じ，キャッシュヒット率が下がるなど，計算遅延が発生してしまう．そこで我々は，テンソルデータを因子軸毎に異るデータ構造を所有させ，因子行列のみを共通データとして所有し，因子行列の更新計算を各データ構造上で行った．

具体的には，例えば因子行列 $\boldsymbol{A} = [a_{ir}]$ では，目的関数の距離計算に KL ダイバージェンスを用いた場合は，SNTF の更新式は下記のようになり，右辺の分子部分の処理を高速に行う必要がある，

$$a_{ir} := a_{ir} \times \frac{\sum_{(j,k) \in \boldsymbol{L}_i} \frac{x_{ijk}}{\hat{x}_{ijk}} b_{jr} c_{kr}}{\sum_j^J \sum_k^K b_{jr} c_{kr}} \tag{1}$$

ここで $\boldsymbol{L}_i$ とは，i に関する j, k のログの集合で $\sum_i^I |\boldsymbol{L}_i| = L$ である．上式に処理においては，(j, k) に依存した数値を各 (i, r) に対して処理を行う必要が有り，処理の遅延を防ぐためには，シーケンシャルアクセスを行うために X[I][J][K] としてデータ構造を保持しておくと効果的である．しかしながら，因子行列 $\boldsymbol{B}$ 対しては X[I][J][K] に対するアクセスがランダムになるため，高速化のためには X[J][K][I] として別途データ構造を保持する必要がある．

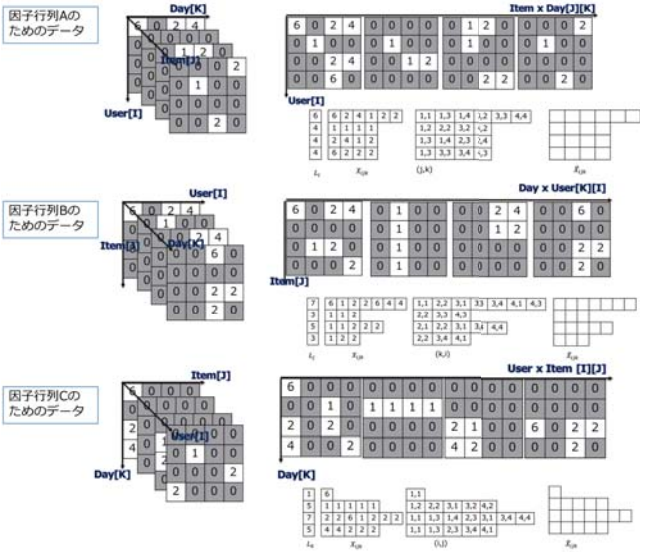

図 1 各因子行列の保持データ構造．

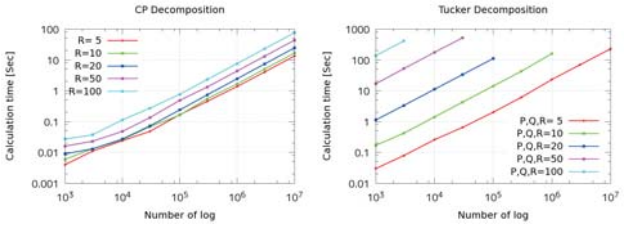

図 2 人工データによる 1 反復当たりの計算処理時間．CP 分解では 1000 万ログのデータを数十秒で処理が可能であり，全体の処理も数時間程度で処理が可能になる．

図 1 は，3 次元のスパーステンソルに対しての各因子行列 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ に対する保持データ構造である．例えば，行列 $\boldsymbol{A}$ に対しては X[I][$(j, k) \in \boldsymbol{L}_i$]，行列 $\boldsymbol{B}$ に対しては X[J][$(k, i) \in \boldsymbol{L}_j$] というようにデータを保持する．特に SNTF ではスパースデータを扱うため，高次の配列構造ではなく，疎行列構造（もしくは 1 次元配列）としてデータを保持する事が効果的である．また本手法は，N 次の SNTF に対しても装用に考えることができ，k 番目の因子行列 $\boldsymbol{Z}^{\boldsymbol{k}}$ に対して X[M^k][$(k+1, k+2, \cdots N, 1, 2, \cdots k-1) \in \boldsymbol{L}_k$] というようにデータを保持する事によって，同様にメモリアクセスを高速に行うことができる．

図 2 は，人工データによる CP 分解と Tucker 分解によるベンチマークである．本研究会では実データによる分析応用例と，GPU への実装適用に関しての議論も合わせて実施予定である．

参考文献

[1] Battenberg, Eric, and David Wessel. “Accelerating Non-Negative Matrix Factorization for Audio Source Separation on Multi-Core and Many-Core Architectures.” ISMIR. 2009.

[2] Jukka Antikainen, et al., “Nonnegative tensor factorization accelerated using gpgpu.” Parallel and Distributed Systems, IEEE Transactions on, 22(7):1135–1141, 2011.

* matsubayashi.tatsushi@lab.ntt.co.jp
† sawada.hiroshi@lab.ntt.co.jp