Adversarial Collaborative Filtering for Free: A Review

RecSys 2023

1 Abstract

既存の CF 手法はデータのノイズに弱く、推薦精度を低下させるという問題がある。この問題に対処するため、多くの先行研究では敵対的学習を用いて**ユーザー・アイテムの表現を正則化**し、汎化性能とロバスト性を向上させている。これらの手法はしばしばミニマックス最適化のフレームワークで摂動とモデルパラメータを同時に学習する。しかし、次の二つの大きな課題が残る。

1. なぜ摂動を加えることでモデルの汎化性能やロバスト性が向上するのかについての理論的保証が不足している点
2. ミニマックス最適化を解くのには計算コストがかかる点
3. モデルパラメータの更新に加えて、各イテレーションで摂動の更新にも追加計算が必要となるため、産業規模のデータセットにはスケールしにくい

この論文では、追加の計算コストをほとんど伴わずに敵対的学習を実現する、シンプルかつ効果的な手法「Sharpness-aware Collaborative Filtering(SharpCF)」を提案する。

まず既存の敵対的協調フィルタリング手法を見直し、最近提案された Sharpness-aware Minimization との関係を解析する。この解析により、**敵対的学習とは実質的に最適解周辺の損失が一様に低い領域（シャープネスの小さい領域）を探索することであり、それが汎化性能を高める**ことが分かる。

計算コスト削減のために、SharpCF では「**トラジェクトリ損失**」と呼ばれる新しい損失項を導入し、現在のモデル重みと過去の重みとの整合性を測定する。実世界データセット上での実験結果から、SharpCF は敵対的学習と比較してほぼ追加コストゼロで優れた性能を達成することが示されている。

2 Introduction

ほとんどの CF 手法では、ユーザーとアイテムを共通の埋め込み空間にマッピングするエンコーダを学習し、さらにその埋め込みをよく表現できるよう目的関数を最適化して有益な表現を獲得します。CF ベースの手法は、そのシンプルさと有効性から、**Factorization Machines [13, 22]** **やグラフニューラルネットワーク [14, 40]** といった形で産業界にも広く導入されています。

しかし、**多くの最先端 CF 手法はノイズの影響を受けやすく、わずかな摂動が加わるだけで性能が大きく劣化してしまうという課題があります** [4, 5, 29, 34]。実世界のデータにはしばしばノイズが含まれ、ユーザーの真の嗜好と実際のインタラクションが必ずしも一致するとは限りません。

この問題に対処するため、多くの先行研究では敵対的学習の原理を利用してユーザーおよびアイテムの表現を正則化する手法が提案されています [10, 15, 28, 38, 41]。これらの手法はミニマックス最適化の枠組みを採用し、敵対的な摂動とモデルパラメータを交互に学習します。たとえば APR モデル [15] では、行列因子分解モデルに対し、ユーザーとアイテムの埋め込みベクトルに直接敵対的摂動を加える形で敵対的学習を適用しています。

既存の敵対的協調フィルタリング手法は有望な成果を示していますが、なお二つの大きな制約が残っています。

1. **敵対的摂動を加えることでモデルの汎化性能やロバスト性が向上する理由について、理論的な保証が不足している**こと。ノイズデータは本質的に元データに対して意図的に設計された敵対的摂動とは異なるため、その解釈性は未だ明確ではありません [12]。
2. **ミニマックス最適化を解くプロセスに膨大な計算コストがかかる**こと [37]。各イテレーションで、モデルパラメータを更新する「外側の最小化問題」に加えて、学習可能な摂動を更新する「内側の最大化問題」の計算が必要となるため、大規模データセットの学習には適しません。

このように、**より効率的で解釈可能な敵対的協調フィルタリング手法の開発**は、推薦システムの分野において依然として大きな課題となっています。

この研究では、上述した制約を克服するために、基本的なオプティマイザに対して追加の計算コストを伴わずに敵対的学習を可能にする、シンプルかつ効果的な手法**「Sharpness-aware Collaborative Filtering（SharpCF）」**を提案します。まず既存の敵対的協調フィルタリング手法を振り返り、近年提案された Sharpness-aware Minimization [2, 9, 11, 20] との関係を論じます。我々の解析により、**敵対的学習は最適モデルパラメータ周辺において損失が一様に低い近傍（フラットな極小点）を探索することを実質的に目指しており、これが汎化性能の向上につながる**ことが明らかになりました。

計算オーバーヘッドを削減するために、本手法 SharpCF では**「トラジェクトリ損失」**と呼ばれる新たな損失項を導入し、現在のモデル状態と過去のモデル状態との整合性を測定します。このトラジェクトリ損失は、**モデルがシャープな極小点へ収束するのを防ぎ、敵対的学習と同様にパラメータをフラットな領域へ導く性質を持つ**ことを示します。興味深いことに、min-max 最適化を用いる既存の敵対的手法とは異なり、SharpCF は標準的な確率的勾配降下法（SGD）で訓練可能であり、計算時間を大幅に削減します。実世界データセット上での実験結果から、SharpCF は最先端の敵対的協調フィルタリング手法をほぼ追加コストなしに上回る性能を達成することが確認されました。

2.1 本研究の貢献

- 既存の敵対的協調フィルタリング手法を再検討し、最近提案された Sharpness-aware Minimization との関連性を明らかにしました。この解析により、**敵対的学習はシャープな極小点よりもフラットな極小点を好む傾向があり、それがモデルの汎化性能向上につながる**ことを示しました。
- 新たに「Sharpness-aware Collaborative Filtering（SharpCF）」を提案しました。本手法では、**現在のモデル状態と過去のモデル状態との整合性を測定する「トラジェクトリ損失」を導入することで、min-max 最適化を解く必要を排除し、追加の計算コストなしに敵対的学習を実現**します。
- 実験結果により、**SharpCF が既存の協調フィルタリング手法を上回る性能を発揮する**ことを示しました。具体的には、ベースラインである BPR に対して平均 18.1% の改善を、APR に対して平均 6.82% の改善を達成しました。また、計算時間の観点では、SharpCF は BPR と同程度の学習速度を維持し、APR の約 2 倍の高速性を実現しています。

3 Related Work

3.1 Collaborative Filtering

深層ニューラルネットワークの成功を受けて、より強力なユーザー／アイテム表現を学習するニューラル CF モデルも提案されています。たとえば、Wide & Deep 推薦モデル［7］は線形モデルと深層ニューラルネットワークを組み合わせ、記憶（memorization）と汎化（generalization）の両方を捉えます。He ら［16］は、ユーザー–アイテム相互作用のモデル化において内積をニューラルネットワークアーキテクチャに置き換えたニューラル協調フィルタリングモデルを提案しました。また、DeepFM［13］はファクタライゼーションマシンとニューラルネットワークを融合し、低次および高次の特徴相互作用を学習します。さらに、XDeepFM［22］は新たにクロスネットワークを導入してより複雑な特徴相互作用を捉え、xLightFM［17］は量子化技術によりファクタライゼーションマシンのメモリフットプリントを大幅に削減しています。

行列因子分解に加え、グラフニューラルネットワークも近年注目を集めており、LightGCN［14］など多くのグラフベース CF モデルが提案されています。これらのモデルは推薦タスクで大きな成功を収め、さまざまなドメインで活用されています［8, 19, 25, 32, 35, 39］。

しかしながら、ノイズの多いデータ（たとえば偽陽性フィードバック）は推薦性能に大きな影響を与えます。一般に、ノイズデータは人気アイテムへのバイアスを生み、正確かつ多様な推薦を難しくします［4, 29, 33, 34, 36］。この問題に対処するため、先行研究ではトレーニング時にマルチタイプのクリックやユーザーのテキストコメントなど、より多様なフィードバックを取り込む方法が検討されています。Wen ら［36］はトレーニングシナリオとして「クリック完了」「クリックスキップ」「非クリック」の 3 種類を提案し、後者 2 つを負例として扱います。Julian ら［24］はユーザーの潜在因子とテキストレビューを組み合わせて評価を補強しました。**しかし、追加フィードバックは多くの場面でコストがかかるか入手困難です。**そこで、**研究者らは追加情報を必要とせずにノイズフィードバックの悪影響を軽減する手段として、敵対的トレーニングの活用を試みています［3, 10, 15, 38, 41］。**

3.2 Adversarial Training

敵対的トレーニングは、推薦システムにおけるノイズデータの悪影響を緩和する有望な手法として注目されています［3, 6, 10, 15, 38, 41］。敵対的トレーニングでは、クリーンなデータと敵対的データを組み合わせてモデルを学習します。**敵対的データとは、入力データに対して最悪のケースを想定した摂動を加えて生成されるデータです［6, 12, 15］。**この手法により、モデルがこれらの摂動に対して頑健になるよう学習され、その結果、未知のデータに対する汎化能力が向上します。例えば、**APR［15］**では行列因子分解モデルにおいてユーザーおよびアイテムの埋め込みベクトルに摂動を加えることで、より頑健かつ高精度な推薦システムを実現しています。NMRN［31］はストリーミング推薦モデルに敵対的トレーニングを適用し、長期的な安定した興味と短期的な動的行動の両方を発見することを目指しています。ACAE［41］は協調オートエンコーダに敵対的トレーニングを組み合わせ、高い競争力を持つ最先端の推薦手法を上回る性能を示しました。ATF［3］では、コンテキスト対応型推薦の精度向上に敵対的トレーニングの利点を活かしています。

これらの手法は一般に、**ミンマックス最適化の枠組みを採用し、敵対的摂動とモデルパラメータを交互に学習します。**しかし、ミンマックス最適化を解くには基礎オプティマイザに対して計算コストが二重にかかるため、**実際の大規模データセットには適用しづらいという課題があります。**そこで近年では、高速化を図った敵対的トレーニング手法がいくつか提案されています［1, 27, 37］。たとえば FreeAT［27］は、モデル学習時に得られる勾配情報を再利用することで敵対的例の生成コストを削減しています。GradAlign［1］は摂動領域内での勾配の整合性を最大化する手法です。これらの最先端手法は高い性能を示す一方で、良好な初期化、大きなステップサイズ、サイクル学習率スケジュールなど一連のヒューリスティックな戦略に依存しており、**安定性に欠けて破滅**

的過学習を起こしやすいという指摘もあります［42］。さらに、既存の高速敵対的トレーニングアルゴリズムの多くは連続値（例：画像のピクセル）に対する摂動生成を想定しており、**離散空間である情報検索タスクには適用が難しいという問題があります［15, 30］。**

これに対し本研究では、**現在のモデル状態と過去のモデル状態との整合性を測る新たなトラジェクトリ損失関数を提案し、ミンマックス最適化を解くことなく計算コストを削減する手法を示します。**また、このトラジェクトリ損失関数が最近提案された Sharpness-aware Minimization［2, 11, 20］と関連し、フラットな極小点を探索することでモデルの汎化性能を高めることを示しています。

4 Preliminaries

4.1 Collaborative Filtering

本論文では、**暗黙的フィードバック**（クリック、閲覧、コメント、購入など）からユーザーの嗜好を学習するタスクに着目します。一般に、**未観測のインタラクションは必ずしもネガティブではなく**、単にユーザーがそのアイテムを知らないだけかもしれません。協調フィルタリングの目的は、アイテムに対するユーザーの嗜好を推定することです。

4.2 Bayesian Personalized Ranking

上記の課題に対処するため、Adversarial Personalized Ranking（APR）［15］は、推薦モデルをパーソナライズされたランキング性能と敵対的摂動への耐性の両面で最適化する目的関数を設計しています。具体的には、潜在因子に対して敵対的摂動 Δ を加えたときのモデル損失を定量化するために、次のように予測スコアを定義します：

$$\hat{y}_{ui}(\Theta + \Delta) = (e_u + \Delta_u)^{\mathsf{T}}(e_i + \Delta_i), \tag{1}$$

ここで摂動ベクトル Δ はそれぞれの潜在因子に対応し、たとえば $\Delta_u \in \mathbb{R}^d$ はユーザー潜在ベクトル e_u に対する摂動を表します。敵対的摂動（worst-case perturbations）はモデルに最大の影響を与えるよう設計されるため、APR ではまず次式で最適な摂動 Δ_{adv} を求めます：

$$\Delta_{\text{adv}} = \underset{\|\Delta\|_2 \leq \rho}{\arg\max}\, L_{\text{BPR}}(\hat{\Theta} + \Delta), \tag{2}$$

ここで ρ は摂動の大きさを制御するハイパーパラメータ、$\|\cdot\|_2$ は L_2 ノルム、$\hat{\Theta}$ は途中のモデルパラメータを示します。

こうして得られた敵対的摂動を用いて、APR はパーソナライズドランキングの合理性と摂動耐性の両立を目指す新たな目的関数を定式化します。すなわち、BPR 損失と敵対的 BPR 損失を次のように統合して最小化します：

$$L_{\text{APR}}(\Theta) = L_{\text{BPR}}(\Theta) + \alpha \cdot L_{\text{BPR}}(\Theta + \Delta_{\text{adv}}), \tag{3}$$

ここで α は敵対的摂動の影響度を制御する重みです。特に $\alpha = 0$ の場合は元の BPR（式 (3)）に帰着するため、APR はモデルの頑健性を考慮に入れた BPR の一般化と見なせます。

式 (6) に示されたこのミンマックス最適化は、モデルパラメータ Θ を最小化プレイヤー、摂動 Δ を最大化プレイヤーとみなしてゲームのように交互に更新を行い、収束を目指します。

しかしながら、APR には次の二点の課題が残されています。

1. **なぜ摂動を加えることでモデルの汎化性能が向上するのかについての理論的保証がないこと。**
2. **ミンマックス最適化を解くプロセスに多大な計算コストがかかること。**

これらの問題意識から、本論文では個別化ランキング学習のための新たな手法 SharpCF を提案します。**SharpCF は理論的保証を提供しつつ、APR と比較してほぼ追加計算コストゼロで学習できる**点が特徴です。

5 The Proposed SharpCF

5.1 Simplify APR

中間変数 Δ が Θ を最小化する目的関数を最大化することから，式 (6) の最適化問題は以下のように表せます：

$$\Theta^*, \Delta^* = \arg\min_{\Theta} \max_{\|\Delta\|_2 \leq \rho} [L_{\mathrm{BPR}}(\Theta) + \alpha\, L_{\mathrm{BPR}}(\Theta + \Delta)]. \tag{4}$$

ここで，古典的な勾配降下–上昇（gradient descent-ascent）アルゴリズムを用いると，変数を交互に固定しながら更新できます：

- 敵対的摂動 Δ の更新（固定 Θ の下で最大化）
- モデルパラメータ Θ の更新（固定 Δ の下で最小化）

$$\Delta^{(t+1)} \leftarrow \arg\max_{\|\Delta\|_2 \leq \rho} L_{\mathrm{BPR}}(\Theta^{(t)} + \Delta) \tag{5}$$

$$\Theta^{(t+1)} \leftarrow \arg\min_{\Theta} L_{\mathrm{BPR}}(\Theta) + L_{\mathrm{BPR}}(\Theta + \Delta^{(t+1)}), \tag{6}$$

具体的には，APR [15] では内側の最大化問題に Fast Gradient Sign Method を使い，外側の最小化問題には標準的な確率的勾配降下法（SGD）を適用しています。この更新式を見ると，

- Δ を更新する際には「敵対的損失」$L_{\mathrm{BPR}}(\Theta + \Delta)$ の勾配のみが寄与し，
- Θ を更新する際には，固定された Δ によって，敵対的損失の勾配が元の BPR 損失の勾配を包含している

ことがわかります。これを踏まえると，「最適化において元の BPR 損失 $L_{\mathrm{BPR}}(\Theta)$ を明示的に含める必要があるのか？」という疑問が生じます。

この疑問に答えるために，MovieLens1M, Gowalla, Yelp2018, Amazon-Book の 4 つの公開データセットを用いて実験を行いました。埋め込み次元 $d = 128$，摂動の大きさ $\rho = 0.5$ と固定し，正則化パラメータ α を $\{0, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, +\infty\}$ の範囲で変化させ，以下の 2 つの損失で学習性能を比較しました：

- ハイブリッド損失：BPR 損失と敵対的損失の組み合わせ（$\alpha \neq 0, +\infty$）
- 敵対的損失のみ：$\alpha = +\infty$ として BPR 損失を完全に排除

図 1 に示すように，$\alpha = 0$（BPR のみ）を上回る性能が常に得られ，特に $\alpha < 100$ の範囲で性能が漸増します。さらに，α が十分大きい（$\alpha \geq 100$）場合，ハイブリッド損失と敵対的損失のみの間に統計的有意な差は見られません。これは，モデルが大きな α によって主に敵対的損失に最適化されるようになり，BPR 損失を外しても性能が低下しないことを示唆しています。

以上より，APR の式 (6) は以下のように簡略化できることがわかりました：

$$\min_{\Theta} \max_{\|\Delta\|_2 \leq \rho} L_{\text{BPR}}(\Theta + \Delta). \tag{7}$$

次節では，この簡略化された敵対的最適化と，最近注目されている Sharpness-aware Minimization との関係を明らかにします。

5.2 Connect to Sharpness-aware Minimization

敵対的トレーニングの汎化性能を理解することは重要です。式 (9) の学習目標は訓練データに完全に適合する複数の局所最適解を持ち得ますが、それぞれの最適解では汎化性能が大きく異なる可能性があるためです。式 (9) の汎化をより深く理解するために、以下のように分解します：

$$\min_{\Theta} L(\Theta) + R(\Theta), \tag{8}$$

where

$$R(\Theta) = \max_{\|\Delta\|_2 \leq \rho} [L(\Theta + \Delta) - L(\Theta)], \tag{9}$$

（ここで L は L_{BPR} の略です。）興味深いことに、式 (10) は Sharpness-aware Minimization（SAM）［11］と同一の形をしており、**項 $R(\Theta)$ はパラメータ Θ を近傍領域 $\{\Theta + \Delta \mid \|\Delta\|_2 \leq \rho\}$ に動かしたときに訓練損失がどれだけ増加し得るかを測ることで、関数 L の" 鋭さ" を捉えています。**したがって、式 (3) のように訓練損失の低い点 Θ を探すのではなく、式 (10) の敵対的トレーニングでは、近傍全体で一様に低い損失を持つフラットな極小点を探すことを実質的に目指していると言えます。言い換えれば、敵対的トレーニングは鋭い極小点よりもフラットな極小点を好み、それが汎化性能の向上につながるのです。

さらに、以下の一般化境界を導出できます：

Proposition 1. 任意の $\rho > 0$ に対し、期待損失を L_D、訓練損失を L_S とする。訓練セット S はデータ分布 D から *i.i.d.* によって引かれるとき、次が成り立つ：

$$L_D(\Theta) \leq \max_{\|\Delta\|_2 \leq \rho} L_S(\Theta + \Delta) + h(\|\Theta\|_2^2 / \rho^2), \tag{10}$$

ここで $h : \mathbb{R}_+ \to \mathbb{R}_+$ は厳密に増加する関数（例：重み減衰としての L_2 ノルム）です。この不等式の証明は *PAC-Bayes* 理論［*23*］を用いることで導けます（詳細は［*11*］中の定理 *2* を参照）。**この命題は、なぜ敵対的トレーニングが汎化性能の向上に寄与するのかを説明する一般化上界を提供します。**

しかしながら、敵対的トレーニング［15］や Sharpness-aware Minimization［11］はいずれも勾配降下–上昇フレームワークを採用しており、バッチ中の各サンプルに対して**「摂動 Δ を更新するための勾配上昇ステップ」と「モデル Θ を更新するための勾配降下ステップ」という二つの順伝搬・逆伝搬が必要**です。これは基礎オプティマイザと比べて計算時間が約二倍になるため、産業規模の大規模データセットに対してはスケーラビリティに課題があります。

5.3 Our SharpCF

計算コストのボトルネックを開所数るために、本研究では Sharpness-aware Collaborative Filtering(SharpCF) を提案します。本手法では、現在のモデル状態と過去のモデル状態との整合性を測る新しい**トラジェクトリ損失**を導入する。まず、式 (8) における内側の最大化問題を Δ に関する一次のテイラー展開で近似する。

$$\begin{aligned}\tilde{\Delta} &= \arg\max_{\|\Delta\|_2\leq\rho} L(\Theta+\Delta)\\ &\approx \arg\max_{\|\Delta\|_2\leq\rho} L(\Theta)+\Delta^T\nabla_\Theta L(\Theta)\\ &= \arg\max_{\|\Delta\|_2\leq\rho} \Delta^T\nabla_\Theta L(\Theta)\\ &= \rho\frac{\nabla_\Theta L(\Theta)}{\|\nabla_\Theta L(\Theta)\|_2}\end{aligned}$$

この近似 $\tilde{\Delta}$ を用いて、各バッチサイズ B に対するシャープネス項 $R_B(\Theta)$ を次のように書き換える。

$$\begin{aligned}R_B(\Theta) &= \max_{\|\Delta\|_2\leq\rho} L_B(\Theta+\Delta)-L_B(\Theta)\\ &\approx \max_{\|\Delta\|_2\leq\rho} L_B(\Theta)+\tilde{\Delta}^T\nabla_\Theta L_B(\Theta)-L_B(\Theta)\\ &= \rho\frac{\nabla_\Theta L_B(\Theta)^T}{\|\nabla_\Theta L_B(\Theta)\|_2}\nabla_\Theta L_B(\Theta)\\ &= \rho\|\nabla_\Theta L_B(\Theta)\|_2\end{aligned}$$

すなわち、**シャープネス項 R_B を最小化することは、vanilla 損失 $L_B(\Theta)$ の勾配の L2 ノルムを最小化することと同値**である。しかし、勾配ノルムを直接最適化するにはヘッセ行列などの二次微分情報が必要となり、計算コストが膨大になる。そこで我々は，過去のモデル状態との整合性を測る「トラジェクトリ損失」を導入し，**二次微分を用いずに勾配ノルムの最適化を「間接的に」実現します。**t 回目のイテレーションにおける過去の重み履歴を $\Theta=\{\Theta_1,\ldots,\Theta_{t-1}\}$ とし，現在の重みを Θ_t と表します。

標準的な SGD 更新は $\Theta_{t+1}=\Theta_t-\eta_t\nabla_{\Theta_t}L_{B_t}(\Theta_t)$ です。

このとき，各反復でシャープネス項 $R_{B_t}(\Theta_t)\geq 0$ を最小化する操作は，以下の期待値形式に帰着します。

まず、シャープネス項 $R_{B_t}(\Theta_t)$ を最小化する問題は次のように書き直せます：

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) \Longleftrightarrow \arg\min_{\Theta_t} \eta_t\cos(\Phi_t)R_{B_t}(\Theta_t)R_{B_t}(\Theta_t). \tag{11}$$

ここで η_t は学習率、Φ_t はバッチ B_t と他バッチとの勾配間の角度を表し、$\cos(\Phi_t)=1$ とします。

さらに、過去のステップにわたる寄与を合わせると、

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) \Longleftrightarrow \arg\min_{\Theta_t} \left[\eta_t\cos(\Phi_t)R_{B_t}(\Theta_t)^2+\sum_{i<t}\eta_i\cos(\Phi_i)R_{B_t}(\Theta_i)R_{B_i}(\Theta_i)\right]. \tag{12}$$

これは各ステップを独立に最小化する形に分解でき、最終的にパラメータ集合 $\Theta=\{\Theta_1,\ldots,\Theta_t\}$ 上の一様分布に関する期待値として表現されます：

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) = \mathbb{E}_{\Theta_i \sim \mathrm{Unif}(\Theta)} \left[\eta_i \cos(\Phi_i) R_{B_t}(\Theta_i) R_{B_i}(\Theta_i) \right]. \tag{13}$$

ここで、$\Theta_i \sim \mathrm{Unif}(\Theta)$ は Θ_i が Θ の要素から一様に選ばれることを、$\mathbb{E}[\cdot]$ は期待値を示します。次に、式 (13) の結果 $R_B(\Theta) \approx \rho \|\nabla_\Theta L_B(\Theta)\|_2$ を代入すると、

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) \Longleftrightarrow \mathbb{E}_{\Theta_i \sim \mathrm{Unif}(\Theta)} \left[\rho^2 \eta_i \cos(\Phi_i) \|\nabla_{\Theta_i} L_{B_t}(\Theta_i)\|_2 \|\nabla_{\Theta_i} L_{B_i}(\Theta_i)\|_2 \right]. \tag{14}$$

さらに内積表示に書き直し、ミニバッチ学習の更新による損失減少量の近似を用いることで、

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) \approx \rho^2 \mathbb{E}_{\Theta_i \sim \mathrm{Unif}(\Theta)} \left[L_{B_t}(\Theta_i) - L_{B_t}(\Theta_{i+1}) \right]. \tag{15}$$

これを初期状態 Θ_1 から現在 Θ_t まで合計すると、

$$\arg\min_{\Theta_t} R_{B_t}(\Theta_t) \approx \frac{\rho^2}{t-1} \left[L_{B_t}(\Theta_1) - L_{B_t}(\Theta_t) \right]. \tag{16}$$

以上より、**シャープネス項の最小化は、初期モデルと現在モデルのバッチ損失差 $L_{B_t}(\Theta_1) - L_{B_t}(\Theta_t)$ を最小化することと同値である**ことが示されます。

我々はまず、シャープネス項 $R_{B_t}(\Theta_t)$ の最小化が、訓練損失差 $L_{B_t}(\Theta_1) - L_{B_t}(\Theta_t)$ の最小化と同値であることに着目しました。しかし、このままでは第二項である $-\frac{\rho}{t-1} L_{B_t}(\Theta_t)$ がバニラ損失と部分的に打ち消し合ってしまうため、損失差そのものではなくその L_2 ノルムである $\|L_{B_t}(\Theta_1) - L_{B_t}(\Theta_t)\|_2$ を用いることにします。

さらに、初期状態 Θ_1 の情報量が少ない点を考慮し、各エポック e においては直近 E エポック分のモデル状態のみを追跡し、$\{\Theta_{e-E}, \ldots, \Theta_e\}$ に対してトラジェクトリ損失を適用します。このとき、ミニバッチ B に対する SharpCF のエポック e での損失は次のように定義されます：

$$L_{\mathrm{SharpCF}}(\Theta_e) = L_B(\Theta_e) + \frac{\lambda}{|B|} \|L_B(\Theta_e) - L_B(\Theta_{e-E})\|_2^2, \tag{17}$$

ここで λ はバニラ損失とトラジェクトリ損失のバランスをとる正則化パラメータ、$L_B(\Theta_{e-E})$ は E エポック前の同一ミニバッチ B に対する損失であり、学習中に記録されています。

ただし、$L_B(\Theta_{e-E})$ および $L_B(\Theta_e)$ ともに式 (3) に従う負例サンプリングを含むため、**同じ正例ペアに対しても異なる負例ペアがランダムに生成される問題があります。**これを緩和するため、本手法ではまず BPR 損失のみでモデルをプリトレーニングし、所定のエポック E_{start} 以降でトラジェクトリ損失の算出を開始します。

プリトレーニングを行ったエポック E_{start} 以降、ユーザー／アイテム表現はより信頼性・安定性を増し、BPR 損失により正例と負例のスコア間に大きなマージン（$\hat{y}_{ui} \gg \hat{y}_{uj}$, $i \in I_u^+$, $j \in I_u^-$）が確保されます。したがって経験的に、ミニバッチ中の正例ペアに対する予測スコアのみを追跡すれば十分であることが分かりました。すなわち、$\hat{Y}_B(\Theta_{e-E})$ と $\hat{Y}_B(\Theta_e)$ のみを比較対象とします。この工夫により、シンプルかつ効果的な経験的損失関数（式 (17)）を次のように定義できます：

$$L_{\mathrm{SharpCF}}^{\mathrm{emp}}(\Theta_e) = L_B(\Theta_e) + \frac{\lambda}{|B|} \|\hat{Y}_B(\Theta_e) - \hat{Y}_B(\Theta_{e-E})\|_2^2. \tag{18}$$

本損失は、トレーニング損失の変化率を抑制し、鋭い局所最小値への収束を防ぐ役割を果たします。アルゴリズム 1 に SharpCF の全体的な学習手順をまとめています。

Algorithm 1 SharpCF 学習アルゴリズム

Require: 訓練データ $\boldsymbol{O}$、正則化パラメータ λ、総エポック数 $\bar{E}$、トラジェクトリ損失開始エポック E_{start}、トラジェクトリウィンドウサイズ E
Ensure: 学習済みモデルパラメータ Θ^*

モデルパラメータ Θ を初期化
for $e \leftarrow 1$ **to** $\bar{E}$ **do**
  for each mini-batch $\boldsymbol{B} \subset \boldsymbol{O}$ **do**
    正例ペアのスコア $\hat{Y}_{\boldsymbol{B}}(\Theta_e)$ を計算
    負例ペアのスコア $\hat{N}_{\boldsymbol{B}}(\Theta_e)$ を計算
    BPR 損失を計算
    E エポック前の損失 $\hat{Y}_{\boldsymbol{B}}(\Theta_{e-E})$ をキャッシュ
    if $e > E_{\text{start}}$ **then**
      $L^{\text{emp}}_{\text{SharpCF}}(\Theta_e) = L_{\boldsymbol{B}}(\Theta_e) + \frac{\lambda}{|\boldsymbol{B}|}\|\hat{Y}_{\boldsymbol{B}}(\Theta_e) - \hat{Y}_{\boldsymbol{B}}(\Theta_{e-E})\|_2^2$
    else
      $L^{\text{emp}}_{\text{SharpCF}}(\Theta_e) = L_{\boldsymbol{B}}(\Theta_e)$
    end if
    $L^{\text{emp}}_{\text{SharpCF}}(\Theta_e)$ を用いて SGD で Θ_e を更新
  end for
end for
return Θ^*

時間計算量の観点では，従来の敵対的トレーニング手法［11, 15］が補助変数 Δ とモデルパラメータ Θ を交互に更新するために順伝搬・逆伝搬を各々二度行うのに対し，SharpCF は標準的な SGD の一度の更新のみで Θ を更新できます。これにより，SharpCF の時間計算量は元の BPR モデル［25］とほぼ同等となり，損失トラジェクトリの記録に必要なごくわずかなオーバーヘッド以外に追加コストはほとんど発生しません。

メモリ計算量の観点では，APR［15］はユーザー数 $|U|$ とアイテム数 $|I|$ の合計に埋め込み次元 d を掛けた

$$O\left((|U| + |I|)d\right) \tag{19}$$

のメモリを必要とします。一方，SharpCF は損失トラジェクトリをキャッシュするために追加で

$$O\left(E|\boldsymbol{O}|\right) \tag{20}$$

のメモリを要します（E はトラジェクトリウィンドウサイズ，$|\boldsymbol{O}|$ は訓練データ数)。しかし E は通常 3 や 5 といった小さな定数であるため，SharpCF のメモリ計算量は BPR［25］と比べても無視できる程度です。

総じて，SharpCF は協調フィルタリングタスクにおいて，計算コスト・メモリコストともに非常に効率的かつ実運用に適した手法であると言えます。

6 Experiments

6.1 Experimental Settings

本研究では，以下の 4 つの公開ベンチマークデータセットを用いて推薦性能を評価しました。

MovieLens-1M：約 6,040 名のユーザーが 2000 年に参加し，約 3,900 本の映画に対して合計 1,000,209 件の匿名評価（レーティング）が記録された，推薦システムで広く使われるデータセットです。

Gowalla：位置情報共有型ソーシャルネットワーキングサービスのチェックイン履歴を集めたデータセットで，2009 年 2 月から 2010 年 10 月までのユーザーのチェックイン記録を含みます。

Yelp2018：Yelp チャレンジで公開されている，ビジネス情報・レビュー・ユーザーデータのサブセットを含むデータセットで，本実験では 2018 年版を使用しました。

Amazon-Book：Amazon.com の書籍カテゴリにおけるユーザーレビュー・評価・タイムスタンプ・商品メタデータを含む大規模コーパスで，本研究では最も大きい「Book」カテゴリを選択しました。

本研究の主目的は、敵対的トレーニングのメカニズムを深く理解しつつ、その計算オーバーヘッドを低減することにあるため、以下の 2 つの代表的手法と比較します。

BPR［25］：Bayesian Personalized Ranking 損失を最適化する古典的モデル。推薦スコアの予測器として行列因子分解を用いています。

APR［15］：BPR と同様に行列因子分解をバックボーンとし、学習時に敵対的摂動を組み込む手法です。

BPR、APR、および本手法 SharpCF では、シンプルかつ有効性の高い行列因子分解を共通のバックボーンとして採用しました。ただし、**SharpCF は BPR と同様に、グラフニューラルネットワーク［5, 14］など任意のエンコーダにも適用可能**です。本研究では新たなエンコーダ開発ではなく、敵対的協調フィルタリングの解釈性向上と高速化に焦点を当てているため、その拡張は今後の課題とします。

開発環境：PyTorch，NVIDIA Tesla V100（32GB）上で実装

埋め込み次元：ユーザー・アイテム埋め込みの次元

$$d = 128 \tag{21}$$

(Eq.(1))

APR のハイパーパラメータ：原論文の設定を初期値とし、性能最適化のため微調整

SharpCF の設定

エポックウィンドウ：

$$E = 3 \tag{22}$$

トラジェクトリ損失開始エポック：

$$E_{\text{start}} = 200 \tag{23}$$

評価指標：トップ-K 推薦性能として Recall と NDCG（Normalized Discounted Cumulative Gain）［14］を使用

デフォルトの K 値は

$$K \in \{10, 20\} \tag{24}$$

本報告では全ユーザーについて Recall@10 と NDCG@10 の平均値を示します。

推論時設定：トレーニングセットで未インタラクションのアイテムを候補とし、全候補アイテムに対してモデルが予測するスコアでランキングを行い、Recall@10 ／ NDCG@10 を算出

実験の再現性：各モデルについて独立に 5 回実験を繰り返し、その平均結果を報告

6.2 Experimental Results

本研究では，提案手法 SharpCF を BPR および APR と，４つの実データセット上で比較しました。Table 2 に各手法の実験結果およびエポックあたりの学習時間をまとめています。実験から，次の 2 点の知見が得られました。

同一の推奨モデル（行列因子分解）を用いても，APR と SharpCF はどちらも BPR を上回る性能を示す APR は最悪ケースの敵対的摂動を明示的に導入することで，SharpCF は学習軌跡を暗黙に平滑化することで，いずれも損失ランドスケープをフラットな極小点へ誘導し，高い性能を達成しています。これは，推薦システムにおいては「モデルの構造」以上に「どのように学習させるか」が重要であり，既存のバニラ損失を少し改変するだけで性能を大きく向上できることを示唆します。

SharpCF はすべての比較において BPR を最大で約 30%上回り，平均で 18.1%（標準偏差 7%）の改善を達成した一方，APR との差分改善はそれほど大きくないものの，SharpCF は平均 6.82%（標準偏差 3.81%）の向上を示しました。これは，SharpCF が min-max 最適化を必要とせず，APR でしばしば問題となるサドルポイントへの停滞リスクを回避できるためと考えられます（APR 論文で指摘されている通りです）。

Figure 2 に，各データセットにおける学習曲線を示します。200 エポックのプリトレーニング後，APR と SharpCF の継続学習では大きく性能が向上するのに対し，BPR の学習はほとんど伸び悩むことが分かります。たとえば Amazon-Book では，BPR の最大 Recall@10 が約 0.0207 であるのに対し，SharpCF では訓練後に 0.0267 まで改善し，約 29% の相対向上を達成しま

Table 2 にはさらに，各手法のエポックあたりの学習時間も併記しています。一般に BPR と SharpCF では学習時間がほぼ同等であるのに対し，APR は約 2 倍の計算コストを要します。たとえば最も大規模な Amazon-Book では，BPR が 127.8 秒／エポック，APR が 230.1 秒／エポック，SharpCF が 128.9 秒／エポックとなっています。

(表追加予定)

実験結果全体を通じて，提案手法 SharpCF の優位性が明らかになりました。特に，SharpCF は４つのデータセットすべてにおいて **BPR および APR を上回る性能を示しながら，計算コストは BPR モデルと同等に抑えられています。**これらの特長から，SharpCF は産業規模のアプリケーションにも実用的であると言えます。

6.3 Further Probe

6.3.1 5.3.1 Long-tail Recommendation

モデルの汎化性能をさらに検証するため、アイテムを度数（インタラクション数）別にグループ化し、各グループにおける平均性能を可視化しました。特に Gowalla と Amazon-Book はデータがまばらなため注目し、度数区間を $[0, 100), [100, 200), [200, 300)$ の 3 つに分けてロングテールアイテムに着目しています。Figure 4 に示すように、低度数アイテム（ロングテール）においても SharpCF と APR は他手法を上回る性能を発揮しています。これは、損失ランドスケープを平滑化し汎化性能を高めるアプローチにより、まばらなインタラクションしか持たないアイテムにも高品質な推薦が可能であることを示しています。

(図追加予定)

6.3.2 5.3.2 The Impact of the Coefficient λ

SharpCF の目的関数（式 (17)）では、オリジナルのタスク損失とトラジェクトリ損失のバランスを取るために係数 λ を導入しています。λ の影響を調べるため、λ を 0.001 から 10 まで変化させて実験を行い、Figure 5 に結果を示しました。両データセットとも、λ を約 0.1 〜 1.0 に設定すると性能が向上し、学習トラジェクトリを追跡することの有効性が確認できます。一方、$\lambda \approx 10.0$ と非常に大きくすると正規化が強くかかりすぎて通常の学習に支障を来し、SharpCF の性能が急激に低下するため、実運用では λ の過度な増大は避けるべきです。

(図追加予定)

6.3.3 5.3.3 The Impact of the Epoch Window Size E

SharpCF では、式 (17) のエポックウィンドウサイズ E が重要なハイパーパラメータとして性能に影響します。大きな E はより長期の学習トラジェクトリを平滑化に利用できる一方、キャッシュすべきモデル状態も増えるためメモリ消費が増大します。Figure 6 に示す通り、E を 1 から 7 まで変化させた場合でも SharpCF の推薦性能は比較的安定しており、特に Amazon-Book では E を大きくするとわずかに性能向上が見られます。本研究では、E = 3 とすることで十分な推薦性能と許容可能なメモリ要件が両立できることを確認しました。

(図追加予定)

7 Conclusion

本論文では，まず既存の敵対的協調フィルタリング手法を再検討し，敵対的トレーニングが鋭い極小点よりもフラットな極小点を好むことで汎化性能が向上することを示しました。次に，基本オプティマイザに対して追加の計算コストをほとんど伴わずに敵対的トレーニングを実現するシンプルかつ効果的な手法「Sharpness-aware Collaborative Filtering（SharpCF）」を提案しました。提案手法は，４つの公開実データセット上で既存の敵対的協調フィルタリング手法を上回る優れた性能を示しつつ，合理的な時間計算量を維持することが確認できました。今後の研究では，公平性を考慮した推薦を含む多様な推薦タスクにおける敵対的トレーニング手法の有効性をさらに評価するとともに，本フレームワークを汎用的なモデル汎化向上の訓練技術として拡張していく予定です。