インタラクション履歴を利用したRNNに基づく推薦手法

周 彦† 牛尼 剛聡‡

†九州大学芸術工学府 〒815-8540 福岡県福岡市南区塩原 4-9-1
‡九州大学芸術工学研究院 〒815-8540 福岡県福岡市南区塩原 4-9-1
E-mail: †zhou.yan.302@s.kyushu-u.ac.jp, ‡ushiama@design.kyushu-u.ac.jp

あらまし 近年，オンラインゲームプラットフォームや音楽配信サービスを始めとして，ユーザが継続的にコンテンツを利用するインタラクティブプラットフォームが増大している．インタラクティブプラットフォームにおいて,ユーザのインタラクション履歴は重要なデータの一つであり,ユーザの嗜好に関する重要な情報を含んでいる．しかし，従来の推薦手法では，インタラクション履歴の特徴や潜在的な情報を利用してユーザの興味とその変化を予測することに対応できない．本研究では，再帰的ニューラルネットワーク（RNN）に基づいた深層学習手法を用いて，ユーザのコンテンツ利用に関するインタラクション履歴を学習し，ユーザの興味とその変化を考慮した推薦手法を提案する.この手法では,インタラクション履歴を深層学習で学習してユーザの興味のトレンドを予測する．予測結果により，Top-N に基づいて推薦するアイテムを決定する．さらに，独自のデータ前処理手法と行列因子分解に基づく次元削減手法を提案し，推薦の性能を向上させる．代表的なゲームプラットフォームである Steam のデータセットを利用して評価実験を行い，精度，効率，多様性という 3 つの基準から提案手法を評価する．

キーワード 推薦システム，インタラクション履歴，RNN

1. はじめに

近年，インタラクティブプラットフォームはインターネットサービスにおいて重要な役割を果たすようになった．インタラクティブプラットフォームとは，ユーザが継続的にアイテムを利用するプラットフォームである.例えば,人気ゲームプラットフォームの Steam や音楽配信サービスの Spotify は，典型的なインタラクティブプラットフォームである．Steam では，ユーザは同じゲームを継続して断続的にプレイし，Spotify では，ユーザは同じ曲を繰り返して再生する．対照的に，従来の多くの推薦手法が対象としてきた非インタラクティブプラットフォームでは，ユーザとアイテムの関係は一回限りである．例えば，Amazon や Rakuten など通販サイトにおいて，一度販売が成立したら，ユーザと通販サイトに販売されている商品（アイテム）の利用関係が更新されることはない．

他のサービスプラットフォームと同様に，インタラクティブプラットフォームにも莫大なアイテムが存在している．例えば，2019 年 1 月の時点で，Steam には 10 万件以上のアプリケーションと 30 万件以上のパッケージが存在する[1]．このように莫大な候補の中からユーザに適したアイテムを発見するため，推薦システムが重要な役割を果たす．協調フィルタリングは，現在最も広く利用されている推薦アルゴリズムの一つであり，高いパフォーマンスを持つことが示されている[2]．一般的な協調フィルタリングは静的なユーザ評価に基づいて推薦を行う[2]．しかし，ユーザの興味は静的ではなく，時間に伴い動的に変化することが多い[3]．一般的な協調フィルタリングでは，ユーザの興味とその変化を考慮しないため，推薦する時点におけるユーザの興味に合うアイテムを選択することが困難である．

インタラクティブプラットフォームは，他のプラットフォームと違い，ユーザと同一アイテムとの間の複数のインタラクションを記録し，これをインタラクション履歴と呼ぶ．例えば，Spotify のインタラクション履歴は，ユーザが聞いた音楽とその時刻から構成される．インタラクション履歴にはユーザの興味とその変化に関する情報が含まれているが，一般的な協調フィルタリングはこの変化しているデータを適切に処理できない．本研究では，インタラクション履歴に含まれた情報を適切に利用し，RNN に基づいた深層学習手法を用いて，ユーザの興味とその変化を予測することにより，アイテム推薦手法を提案する．

本論文の構成を以下に示す．第 2 章では，関連研究について述べる．第 3 章では，前処理であるインタラクション履歴のベクトル化について述べる．第 4 章では，RNN に基づく推薦手法を提案する．第 5 章では，評価実験で提案手法を検証する．第 6 章では，本論文のまとめを述べる．

2. 関連研究

一般的な推薦システムにおけるアイテムに対するユーザ評価は静的であることを前提としている．それに対して，本研究で対象とするインタラクション履歴は動的であり，時間に伴い変化する．インタラクション履歴は時系列として捉えることができ，時系列にお

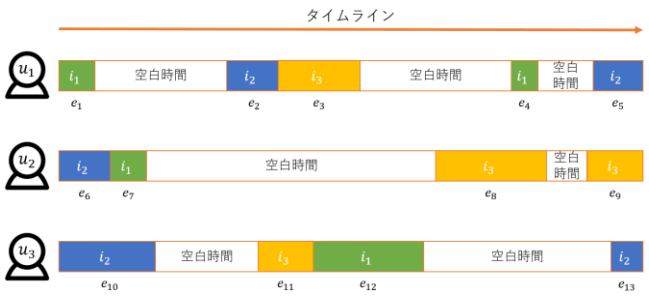

図 1：インタラクション履歴

いて，時間は最も重要な要因である[4]．従来にも，時間的要因を考慮した推薦手法が提案されている．Ding ら[3]は，ユーザの決定は主に最近の興味の影響を受けており，推薦するときには，最新の評価の重要性を増やすべきであると主張した．Koren[5]はデータセットにおける時間的要因を考慮し，SVD++モデルを改善した．Gordea ら[6]は，時間要因により評価データにペナルティ値を設定する手法を提案した．この手法では，時間の距離がペナルティ値より大きい場合，評価データを無効とする．Baltrunas ら[7]は，時間に基づいたフィルタリング手法を提案した．この手法では，同じ期間に現れた評価データのみでモデルをトレーニングする．時系列から要素データを抽出し，同じ時刻のデータに基づいて，ユーザ同士の類似度を計算する．Zimdars ら[8]は正規化したデータ集合を利用した推薦手法を提案した．正規化したデータ集合とは，ユーザとアイテムのインタラクションの順番から作成されたデータ集合である．Lu ら[9]は，ユーザとアイテム両方の特徴が時間に伴い変化すると考えており，時空間に基づいた協調フィルタリングモデルを提案した．このモデルでは，時空間フィルタリングアプローチを用いて，行列分解に基づき，ユーザ要因とアイテム要因を推定する．これらの研究は，動的評価を静的評価に変換し，推薦を行う．そのため，以上の研究は一部の因子しか考慮していないため，興味の変化など，時系列における情報を全面的に考えることができない．

一般的に，推薦システムはオンライン推薦とオフライン推薦に大別できる[10]．オンライン推薦とは，リアルタイムに評価情報を利用し，推薦アイテムを計算する手法であり，オフライン推薦とは，評価履歴を利用し，事前に推薦アイテムを計算する手法である．これまでに提案されている時系列データを対象とした推薦システムの研究では，オンライン推薦を対象としたものが多い．Lommatzsch ら[11]は，リアルタイムデータのコンテキスト情報に注目し，異なるコンテキストでの推薦精度を分析し，複数の推薦アルゴリズムを組み合わせた手法を提案した．この手法により，システムはコンテキストを条件として，適切な推薦アルゴリズムを選択することができる．Chen ら[12]は情報ストリームを分析することにより，ユーザとソーシャルトピックスに対する興味予測モデルを提案した．Subbian ら[13]はリアルタイム推薦の効率を改善するため，ハッシュテーブルを用いた推薦モデルを提案した．リアルタイム推薦では，ユーザの短期的な興味と流行を重視し，主にシステムの効率化を目的とするが，過去の評価データを十分に利用しないため，推薦アイテムの多様性が足りないという問題があり，候補アイテムが少ない場合，適切なアイテムの推薦が困難である[14]．この問題に対し，Kille ら[15]はオンラインとオフライン両方とも考慮した手法を提案し，オフライン推薦の優位性を示している．この結果に基づいて，本研究では，オフライン推薦を対象として，推薦アルゴリズムを提案する．

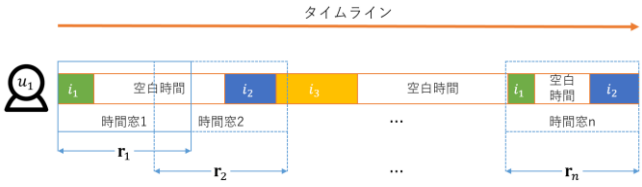

図 2：スライディングウィンドウ

ユーザインタラクションを考慮したとき，従来の研究では，セッションベース推薦手法を提案することが多い．セッションベース推薦とは，ユーザのセッションの期間におけるインタラクションに基づいて，推薦を行うことである[16]．セッションベース推薦では，短いセッションの期間を対象とするため，短期的な興味を重視する[17,18]．しかし，インタラクション履歴には，長期的な興味と短期的な興味の両方が含まれるため，長期的な興味を正確に予測し，バランスを取って評価する推薦システムが必要である．

本研究で使用するインタラクション履歴は，典型的な暗黙的フィードバックである．ユーザの直接評価と比べ，暗黙的フィードバックからユーザの嗜好を抽出するのは困難である．これまでに，確率モデルや予測モデルを用いて，暗黙的フィードバックを考慮した協調フィルタリング手法が提案されている[19,20]．一部の研究では，深層学習を用いて，暗黙のフィードバックを考慮する推薦システムを構築する[17,18,21]．そのため，暗黙的なフィードバックであるインタラクション履歴を使用する前に，適切な前処理が必要である．

3. インタラクション履歴のベクトル化

3.1. インタラクション履歴

本研究で利用するインタラクション履歴を形式的に定義する．いま，ユーザ集合が**U**で，アイテム集合が

Iで表されるとき，個々のインタラクションeを，ユーザ$u \in \mathbf{U}$，アイテム$i \in \mathbf{I}$，インタラクションの開始時点t_{start}と終了時点t_{end}の組として$e = (u, i, t_{start}, t_{end})$と定義する．また，インタラクション履歴を集合$\mathbf{E} = \{e_1, e_2, e_3 \cdots\}$で表わす．

図 1 はインタラクション履歴の例を表している．図 1 では，ユーザ集合$\mathbf{U} = \{u_1, u_2, u_3\}$，アイテム集合$\mathbf{I} = \{i_1, i_2, i_3\}$と想定している．色がついている区間は，左側のユーザによる異なるアイテムとのインタラクションeを表す．図 1 により，このインタラクション履歴集合には 13 個の要素が含まれている．上の矢印は時間の流れを表し，区間の左端点はt_{start}，右端点はt_{end}とする．色がついていない区間は，左側のユーザが，その期間にプラットフォームを利用していなかったことを表し，空白時間と呼ぶ．図 1 では，3 人のユーザ$\{u_1, u_2, u_3\}$に関して，このタイムラインの期間内の 3 つのアイテム$\{i_1, i_2, i_3\}$に関する利用状況を示している．

インタラクション履歴には，アイテムの利用時間と利用順序という 2 種類の情報が含まれている．利用時間はユーザの興味，利用順序はその変化を反映するため，インタラクティブプラットフォームにおける推薦システムを構築するとき，両方とも考慮することが必要である．図 1 に示すように，インタラクション履歴は連続的であり，一般的なモデルを利用して学習するのは困難である．そのため，推薦モデルを考慮する前に，インタラクション履歴を離散化して，時系列ベクトルに変換することが必要である．その上，離散化するとき，元データが持つ情報が失われる可能性がある．そこで，インタラクション履歴の離散化では，可能な限りインタラクション履歴の情報，アイテムの利用時間と利用順序を保持することが重要である．本論文では，この時系列ベクトルを評価ベクトルと呼び，離散化をベクトル化と呼ぶ．

3.2. スライディングウィンドウ

時系列データのベクトル化手法の代表的な手法の 1 つに，スライディングウィンドウ手法がある[22]．スライディングウィンドウ手法では，最初に与えたサイズにより，時間窓を作る．そして，時間窓をタイムラインに沿って与えられたスライディング幅で移動させる．移動する度に，窓の中のインタラクション時間をアイテムごとに累計して評価ベクトル$\mathbf{r}$を構成する．図 2 は図 1 のユーザu_1を例としたスライディングウィンドウによるベクトル化の例を示す．スライディングウィンドウ手法は一定期間ごとに評価ベクトルを生成するため，評価ベクトルの要素，即ち，単位期間のインタラクション時間で直接にユーザが各アイテムへの興味を評価できる．同時に，評価ベクトル系列$\{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_n\}$の

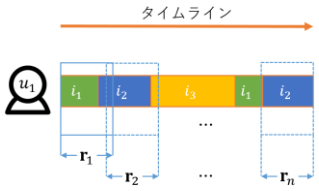

図 3：空白時間が排除されたインタラクション履歴に基づくスライディングウィンドウ

変化はユーザ興味の変化を表す．本論文では，スライディングウィンドウに基づいて，ベクトル化を行う手法を提案する．

3.3. 空白時間の除外

多数のユーザにとって，タイムラインの大半は空白時間である．そのため，図 2 の様に，多数の時間窓には，短いインタラクション時間しか存在していない．このような場合，評価ベクトルがまばらになる．一般的な機械学習手法では，このような，まばらなベクトルから特徴を学習することは難しい．また，それ以外にも，インタラクション時間の長さは様々な外的な要因の影響を受けやすい．例えば，ユーザが忙しくてプラットフォームを利用できない期間では，インタラクション時間が短くなり，暇である期間では，インタラクション時間が長くなる場合が考えられる．そのため，空白時間のあるインタラクション履歴から生成された評価ベクトル系列の信頼度が低い．本手法では，ベクトル化する前に，インタラクション履歴から空白時間を排除する．例として，図 3 に空白時間が排除された図 2 のインタラクション履歴に基づくスライディングウィンドウを示す．

3.4. 不完全なインタラクション履歴

インタラクション履歴から空白時間を排除して，スライディングウィンドウを適用すれば，インタラクション履歴をベクトル化できる．しかし，この様なインタラクション履歴は，全てのインタラクションの発生時点t_{start}と終了時点t_{end}は正確に記録されている理想的な状態を想定している．しかし，ユーザの継続的な行動データを API で収集できるプラットフォームは少数である．たとえ収集できたとしても，プラットフォームから継続的にデータを収集することは困難である．システムに対するコストを考慮した場合，一定時間毎に収集しかできないこともある．

このような状況で収集されたインタラクション履歴を，不完全なインタラクション履歴と呼ぶ．いま，インタラクション履歴を集合$\mathbf{E} = \{e_1, e_2, e_3 \cdots\}$で表わすとき，$e$は個々のインタラクションではなく，一定期間

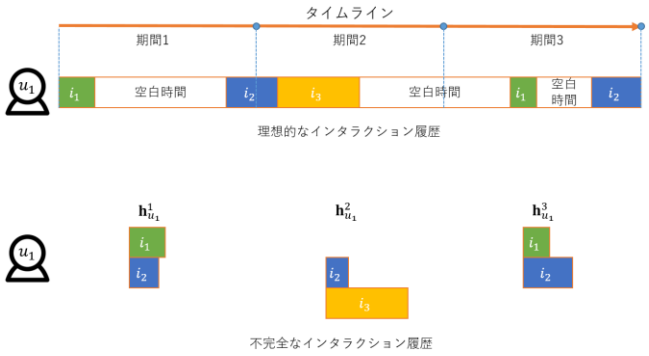

図 4：理想的なインタラクション履歴と不完全なインタラクション履歴

ごとに収集したデータである．データはこの期間にユーザと各アイテムの合計インタラクション時間である．即ち，$e_n = \{\mathbf{h}_u^n | u \in \mathbf{U}\}$であり，$n$は時間順を表す．$\mathbf{h}_u^n$は履歴ベクトルであり，，このベクトルの要素は期間$n$の間にユーザ$u$と各アイテムの合計インタラクション時間である．図 4 は，図 1 のユーザu_1のインタラクション履歴と同じ対象に対する不完全なインタラクション履歴の例を表す．

空白時間が存在するため，履歴ベクトルはスライディングウィンドウを利用した場合の結果と同じく，外的な要因の影響を受けやすい等の問題があり，直接学習するのは困難である．加えて，不完全なインタラクション履歴では，二つの履歴ベクトル間のインタラクションの順序は分かるが（例えば，$\mathbf{h}_{u_1}^1$のi_2と$\mathbf{h}_{u_1}^2$のi_2），同じ履歴ベクトル内のインタラクションの順序が分からない（例えば，$\mathbf{h}_{u_1}^1$におけるi_1とi_2）．そのため，図 3 の様に，直接，空白時間を排除することは不可能である．このような不完全なインタラクション履歴から評価ベクトル系列を作るため，履歴ベクトルの再構築を行う．

3.5. 履歴ベクトルの再構築

履歴ベクトルの再構築のために，ユーザuに対し，履歴ベクトル系列$\{\mathbf{h}_u^n | n \in \{0,1,2,\cdots\}\}$から均等の評価ベクトル系列$\{\mathbf{r}_u^t | t \in \mathbb{N}\}$を構築する．再構築の際には，履歴ベクトルにおける利用時間と利用順序に関する情報を出来る限り保留する．再構築手法では，まず，評価ベクトル$\mathbf{r}_u^t$の数Tを決める．即ち，$t \in [0, \mathrm{T}-1]$である．初期化のために，すべての$\mathbf{r}_u^t = \mathbf{0}$とし，初期値として，$n, t = 0$とする．ユーザ$u$の全部の評価ベクトル$\mathbf{r}_u^t$の$L^1$-ノルム$norm_u$を式(1)により計算する．

$$norm_u = \frac{\sum_{n\in\{0,1,\cdots\}} \|\mathbf{h}_u^n\|_1}{\mathrm{T}} \tag{1}$$

そして，すべてのユーザ$u \in \mathbf{U}$に対して，以下のプロセスを，すべての履歴ベクトル$\mathbf{h}_u^n$が処理されるまで繰り返す．

1. 式(2)により，$\mathbf{h}_u^n$から引いて，$\mathbf{r}_u^t$に加える比率$ratio$を計算する．

$$ratio = \frac{norm_u - \|\mathbf{r}_u^t\|_1}{\|\mathbf{h}_u^n\|_1} \tag{2}$$

2. $ratio$を判断する．
   a) $ratio > 1$の場合，$\mathbf{r}_u^t = \mathbf{r}_u^t + \mathbf{h}_u^n$とし，$n = n + 1$とする．
   b) $ratio = 1$の場合，$\mathbf{r}_u^t = \mathbf{r}_u^t + \mathbf{h}_u^n$とし，$n = n + 1$および$t = t + 1$とする．
   c) $ratio < 1$の場合，$\mathbf{r}_u^t = \mathbf{r}_u^t + ratio \cdot \mathbf{h}_u^n$，$\mathbf{h}_u^n = (1 - ratio)\mathbf{h}_u^n$とし，$t = t + 1$とする．

全てのベクトルに対して，上記の手順を繰り返し適用すると，全ての履歴ベクトル$\{\mathbf{h}_u^n | n \in \{0,1,2,\cdots\}, u \in \mathbf{U}\}$は変換され，評価ベクトル系列$\{\mathbf{r}_u^t | t \in [0, \mathrm{T}-1], u \in \mathbf{U}\}$に格納されている．このとき，評価ベクトル$\mathbf{r}_u^t$は$t$番目の長さ$norm_u$のインタラクション時間にユーザ$u$と各アイテムのインタラクション時間を意味する．上記の手順の疑似コードは以下のように示される．

Algorithm

Input: $\{\mathbf{h}_u^n | n \in \{0,1,2,\cdots\}\}, \mathrm{T}$
Output: $\{\mathbf{r}_u^t | t \in [0, \mathrm{T}-1]\}$
$\mathbf{c} \leftarrow \mathbf{0}$
$norm_u = \sum_{n\in\{0,1,2,\cdots\}} \|\mathbf{h}_u^n\|_1 / \mathrm{T}$
$t \leftarrow 0$
foreach $n \in \{0,1,2,\cdots\}$ ***do***
    while $\|\mathbf{c}\|_1 + \|\mathbf{h}_u^n\|_1 > norm_u$ ***do***
        $ratio \leftarrow (norm_u - \|\mathbf{c}\|_1) / \|\mathbf{h}_u^n\|_1$
        $\mathbf{r}_u^t \leftarrow \mathbf{c} + \mathbf{h}_u^n \cdot ratio$
        $\mathbf{c} \leftarrow \mathbf{0}$
        $t \leftarrow t + 1$
        $\mathbf{h}_u^n \leftarrow \mathbf{h}_u^n \cdot (1 - ratio)$
    end while
    $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{h}_u^n$
end for

3.6. バイアス

3.5 で生成した評価ベクトルの要素は，インタラクション時間であり，異なるユーザと異なるアイテムに対し，インタラクション時間の分布が異なる．インタラクション時間から実際の評価を推測するため，本論文はユーザとアイテムバイアスを考慮して，評価ベクトル$\mathbf{r}_u^t$の各要素$r_u^t(i)$に対し，式(3)により新しい要素$r'^t_u(i)$を計算する．

$$r'^t_u(i) = (e-1)\frac{|\mathbf{U}|}{\sum_{v\in\mathbf{U}} r_v^t(i)} r_u^t(i) + 1 \tag{3}$$

4. RNN に基づく推薦手法

4.1. 予測モデル

インタラクション履歴を用いて，推薦では，ユーザの未来のインタラクションの予測が必要となる．本章

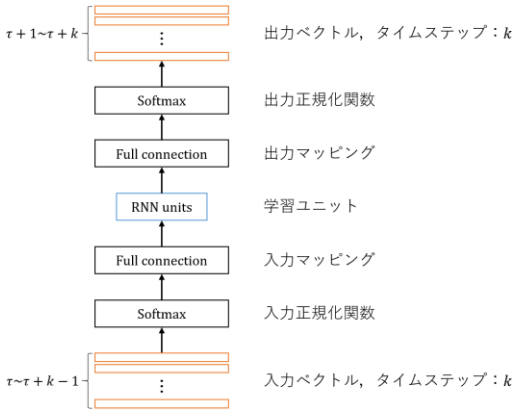

図 5：予測モデル

では，深層学習を用いた予測モデルを提案し，予測結果に基づいて推薦を行う．深層学習は，ニューラルネットワークにより，データセットの特徴を学習する．第 3 章に示したように，本研究のデータセットは時系列の評価ベクトルであるため，図 5 に示すリカレントニューラルネットワーク（Recurrent Neural Network，RNN）予測モデルを利用する．この RNN の入力と出力は，要素数がkであるベクトル系列である．Softmax 関数を用いてベクトルを正規化し，全結合層を用いてベクトルの次元を学習ユニットの数と一致させる．モデルを訓練するとき，ユーザuに対し，τ時点からの評価ベクトル系列$\{\mathbf{r}_u^t | t \in [\tau, \tau+k-1]\}$を入力とする場合，正解出力は$\tau+1$時点からの評価ベクトル系列$\{\mathbf{r}_u^t | t \in [\tau+1, \tau+k]\}$である．推薦を行うとき，対象ユーザ$u$の最近の評価ベクトル系列$\{\mathbf{r}_u^t | t \in [\mathrm{T}-k, \mathrm{T}-1]\}$を訓練されたモデルに入力し，出力ベクトル$\mathbf{r'}_u^{\mathrm{T}}$を予測結果として得る．

4.2. RNN ユニット

図 5 の予測モデルにおいて，実際に学習を行うのは RNN ユニットである．時系列学習において，RNN 以外にも，LSTM[23]や GRU[24]など，多くのモデルが提案された．一般的な RNN モデルと比べ，LSTM と GRU モデルは時系列における長期的な情報を学習できることが示されている[25,26]．本研究では，時系列の情報はインタラクション履歴に隠したユーザの興味とその変化である．本研究は，ユーザの短期興味と長期興味両方及び興味の変化を考慮するため，RNN，LSTM，GRU の 3 種類のモデルを利用して，予測モデルを構築し，実験し，それらの性能を考察する．

4.3. 次元削減

これまで，一般的に，深層学習を用いて推薦システムを構築するのは現実ではないとされてきた．その理由は，行列分解をはじめとする一般的な推薦手法と比べ，ニューラルネットワークの訓練コストは非常に大きいからである．推薦システムでは，膨大なユーザを持つ以上，許容できる時間に推薦を出さなければならない．特に，時系列に基づく推薦システムでは，静的な評価に基づく推薦システムより，数倍のデータを使う必要があるため，処理の効率化は避けられない課題である．

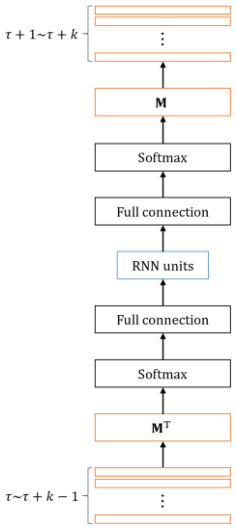

図 6：次元削減を考慮した予測モデル

一般的に，ニューラルネットワークの訓練効率を向上させるには，2 つの重要なポイントがある．訓練データが巨大である場合には，複雑な LSTM モデルと比べ，単純な GRU 等のモデルを利用することが考えられる．本研究は RNN，LSTM，GRU の 3 つのモデルを検討する．一方，訓練データとするベクトルの次元が大きい場合には，ベクトルの次元の大きさは最もシステムの効率に影響を与える原因だと考えられ，次元削減は必要である．そこで，本節では，次元削減手法を提案する．

第 3 章に示したように，評価ベクトルは$\mathbf{r}_u^t \in \mathbb{R}^{|\mathbf{I}|}$であり，$\mathbf{I}$はアイテム集合を表す．評価ベクトルの次元はアイテムの数により決定され，数万や数十万になる可能性がある．例えば，Steam データセットでは評価ベクトルの次元はアイテム数の 14,649 である．評価ベクトルを低次元空間$\mathbb{R}^k$に圧縮するには，$|\mathbf{I}| \times k$次元の射影行列$\mathbf{M}$が利用されるのが一般的である．本手法では，式(4)に示すように，低次元評価ベクトル$\boldsymbol{\gamma}_u^t$を射影行列$\mathbf{M}$と元評価ベクトル$\mathbf{r}_u^t$の積として求める．

$$\boldsymbol{\gamma}_u^t = \mathbf{M}^{\mathrm{T}} \mathbf{r}_u^t \tag{4}$$

次元削減を考慮した予測モデルを図 6 に示す．

LFM（Latent Factor Model）は行列分解手法の 1 つである．LFM では，$|\mathbf{U}| \times |\mathbf{I}|$のユーザ-アイテム評価行列$\mathbf{R}$に対し，式(5)に示す様に，$|\mathbf{U}| \times k$次元のユーザ行列$\mathbf{N}$と，

$|\mathbf{I}| \times k$次元のアイテム行列$\mathbf{M}$の積で近似する．

$$\mathbf{R} \approx \mathbf{N}\mathbf{M}^{\mathrm{T}} \tag{5}$$

ここで，kは隠れ因子（Latent Factor）である．そのため，ユーザ行列$\mathbf{N}$はユーザの隠れ因子への嗜好を表し，アイテム行列$\mathbf{M}$はアイテムの各特徴（隠れ因子）の重みを表す[27]. ここのアイテム行列$\mathbf{M}$は射影行列である．LFM を使うため，静的なユーザ-アイテム評価行列$\mathbf{R}$が必要である．しかし，第 3 章に説明した様に，インタラクション履歴から生成した評価ベクトル系列は時系列であるため，評価ベクトルで構成された評価行列は動的である．LFM では，動的評価行列から射影行列を計算するため，本研究は「アイテム特徴の変化速度はユーザ興味の変化速度より非常に遅い」と仮説を立てた．この仮説のもとでは，動的な評価行列$\mathbf{R}$は動的なユーザ行列$\mathbf{N}$と静的なアイテム行列$\mathbf{M}$の積の近似と見ることができる．射影に必要なのはアイテム行列$\mathbf{M}$だけであるため，本研究は全体的な評価行列で動的な評価行列の代わりに用いて，LFM でアイテム行列を計算する．全体的な評価行列は式(6)で計算する．

$$\mathbf{R} = \left[\sum_{t\in[0,\mathrm{T}-1]} \mathbf{r}_{u_1}^{t}, \cdots, \sum_{t\in[0,\mathrm{T}-1]} \mathbf{r}_{u_v}^{t}\right]^{\mathrm{T}} \tag{6}$$

4.4. Top-N 推薦

訓練された予測モデルを用いて，推薦を行う．ユーザの最近の評価ベクトル系列$\{\mathbf{r}_u^t | t \in [\mathrm{T}-k, \mathrm{T}-1]\}$をモデルに入力した際の出力は，ベクトル系列$\{\mathbf{r}'^t_u | t \in [\mathrm{T}-k+1, \mathrm{T}]\}$である．系列の最後のベクトル$\mathbf{r}'^{\mathrm{T}}_u$は予測ベクトルである．式(7)を用いて，ユーザが既知のアイテムを除外して，優先度が最も高い N 個のアイテムを推薦する．

$$\left\{ i \left| \sum_{t=0}^{\mathrm{T}-1} r_a^t(i) = 0 \right. \right\} \tag{7}$$

5. 評価実験

5.1. データセット

本論文では，ゲームプラットフォーム Steam のデータを利用して，提案手法の有効性を評価する．推薦アイテムがゲームである場合，インタラクション時間はゲームのプレイ時間である．既存の履歴が存在しないため，本研究は Steam Web API[28]を使用してプラットフォームからインタラクション履歴を直接収集した．指定したユーザ ID に対して，この API は過去 2 週間のゲーム時間の一覧を取得することができる．13 回の収集で得られた 26 週間のデータセットには，10,243 人のユーザと 14,649 件のゲームの 604,109 件のデータが含まれる．

5.2. 実験方法

実際の推薦プロセスをシミュレートするため，最近 2 週間のデータをテスト用のデータセットとし，それ以前の 24 週間のデータをトレーニング用のデータセットとする．Steam のデータセットは 3.4 に説明した不完全なインタラクション履歴であるため，3.5 の再構築手法を用いてトレーニングセットを評価ベクトル系列に変換する．なお，ここでの系列の数は$\mathrm{T} = 12$とした．

各 RNN モデルを評価するため，本研究は 3 つの実験（RNN，LSTM，GRU）を行って，時間を考慮しないユーザベース協調フィルタリング手法（UserCF）をと比較する．再構築手法，バイアスと次元削減の有効性を評価するため，LSTM に基づいて，再構築を適用しない（LSTMnoRC），バイアスを考慮しない（LSTMnoBias）と次元削減を考慮しない（LSTMnoDR）対照実験を行う．全ての実験では，推薦アイテムの数は$\mathrm{N} = 40$とする．予測モデルのパラメータについて，RNN ユニットの数は 256，タイムスタンプkは 8，学習回数は 100，動的学習率は 0.001-0.0001 とした．

評価のために，式(8)で表されるリコール率 Recall を計算する．ここでは，ユーザuに対し，推薦集合は$\mathbf{Rec}_u$，テストセットは$\mathbf{New}_u$，トレーニングセットのアイテム集合は$\mathbf{Known}_u$で表される．

$$\mathrm{Recall}_{40} = \frac{\sum_{u\in\mathbf{U}} |\mathbf{Rec}_u \cap (\mathbf{New}_u - \mathbf{Known}_u)|}{\sum_{u\in\mathbf{U}} |\mathbf{New}_u - \mathbf{Known}_u|} \tag{8}$$

推薦アイテムの数$\mathrm{N} = 40$の場合，また，式(9)で表す DCG も評価に利用する．ここでは，推薦アイテムは優先度$p_u(i)$の順に並べ替えられ，$q_u(i)$はテストセットにおいて，順位iのアイテムに対するユーザuの評価を表す．

$$\mathrm{DCG}_{40} = \sum_{u\in\mathbf{U}} \left(\log(q_u(1)+1) + \sum_{i=2}^{40} \frac{\log(q_u(i)+1)}{\log_2 i} \right) \tag{9}$$

DCG は推薦アイテムを並べ替えて，スコアを与え，前ほどスコアが高くなる．比べて，リコール率では，推薦アイテムを集合とし，全てのアイテムのスコアが同じである．即ち，DCG はアイテムのランキングを考慮し，リコール率は考慮しない．平均 DCG は式(10)により計算する

$$\mathrm{AvgDCG}_{40} = \frac{\mathrm{DCG}_{40}}{|\mathbf{U}|} \tag{10}$$

5.3. 実験結果

RNN，LSTM，GRU ユニットを利用したとき，RMSE の変化は図 7 のようになる．表 1 と表 2 は比較手法と対照手法の推薦精度を示す．表 3 は提案手法（RNN，LSTM，GRU）と次元削減を考慮しない場合（noDR）

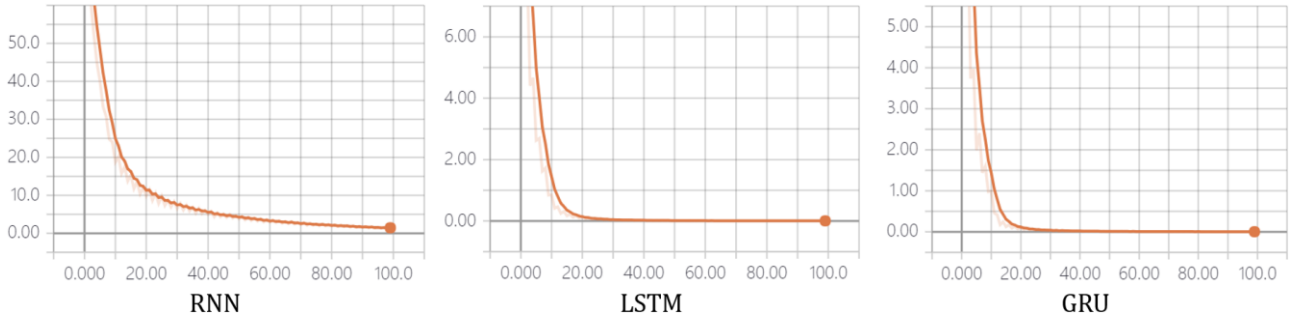

図 7：RMSE の降下率

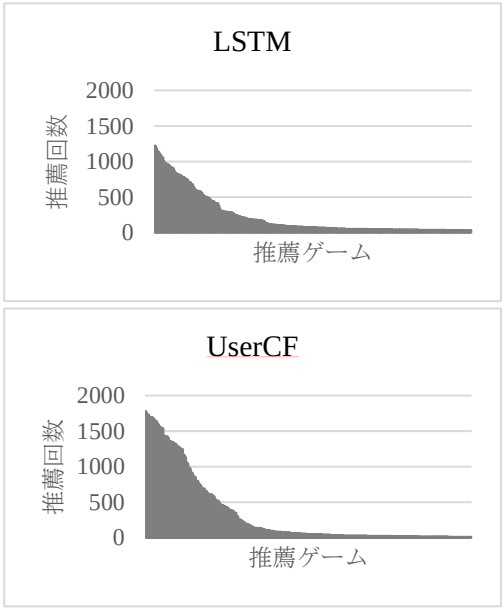

図 8：推薦回数が多いゲームトップ 300

表 1：比較手法の推薦精度

	リコール率	平均 DCG
RNN	0.0958	1.153
LSTM	**0.1194**	**1.471**
GRU	**0.1242**	**1.466**
UserCF	0.0608	0.994

表 2：対照手法の推薦精度

	リコール率	平均 DCG
LSTM	**0.1194**	**1.471**
LSTMnoRC	0.0897	1.270
LSTMnoBias	0.1006	**1.514**
LSTMnoDR	**0.1202**	**1.443**

表 3：訓練時間（単位：秒）

	次元削減	削減しない
RNN	**1951**	3076
LSTM	**2182**	3531
GRU	**2104**	3255

の訓練時間を示す．図 8 は LSTM ユニットを利用したときの提案手法とベースラインである UserCF により，推薦回数が多いゲーム上位 300 の分布を示す．

5.4. 考察

図 7 から，パラメータが同じである場合，LSTM と GRU の誤差降下率は RNN より速いため，本研究のデータセットに対し，LSTM と GRU の学習性能は一般的な RNN モデルより良いと考えられる．表 1 により，3 つの提案手法，特に LSTM と GRU はベースラインである UserCF と比べ，高いリコール率と平均 DCG を持ち，高い推薦精度を持つことが示された．表 2 により，対照実験 LSTMnoRC と LSTMnoBias と比べ，LSTM はより高い推薦精度を示し，再構築手法とバイアスの提案手法に対する有効性も証明された．一方，LSTM は LSTMnoDR と比べ，同じ程度のリコール率と平均 DCG を持つが，表 3 により，提案手法の訓練時間が noDR より大幅に少ないため，少ない処理コストでほぼ同等の性能を持つことを示された．まだ，表 3 によりも，3 つのユニットの効率は RNN > GRU > LSTM の順序となった．図 8 に示した推薦ゲームの分布により，LSTM による推薦ゲームは UserCF より広く分散していると見えるため，提案手法は UserCF よりもダイバーシティが高いことが証明された．

6. おわりに

本論文では，インタラクティブプラットフォームにおけるユーザのインタラクション履歴を考慮したアイテム推薦手法を提案した．動的なデータであるインタラクション履歴からユーザの興味とその変化を抽出し，興味の予測により，推薦アイテムを計算することが本研究の課題である．本研究では，RNN を用い，インタラクション履歴に基づいた推薦システムを構築した．最後に，Steam のデータセットを利用し，提案手法を評価した．実験結果により，提案手法は推薦精度，推薦効率及び推薦アイテムの多様性，この 3 つの基準で，従来の研究より良い結果を示した．

今後，本論文提案手法に利用した技術に対して，更

に検討を行う必要がある．また，他のデータセットに対しても，提案手法の有効性を評価するための実験を行う予定である．

参 考 文 献

[1] Steam Database, https://steamdb.info/

[2] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor. Recommender Systems Handbook. Springer, 2010.

[3] Y. Ding, X. Li. Time Weight Collaborative Filtering. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05, ACM, 2005, pp. 485-492.

[4] Michael Folk, et al. A first Course on Time Series Analysis: Examples with SAS. GNU Free Documentation Licence, 2012.

[5] Y. Koren. Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, KDD '09, pp. 447-456. ACM, 2009.

[6] S. Gordea, M. Zanker. Time Filtering for Better Recommendations with Small and Sparse Rating Matrices. In Proceedings of the 8th International Conference on Web Information Systems Engineering, WISE '07, pp. 171-183. Springer-Verlag, 2007.

[7] L. Baltrunas, X. Amatriain. Towards Time-dependant Recommendation Based on Implicit Feedback. Workshop on Context-Aware Recommender Systems, CARS '09, 2009, pp. 1-5.

[8] A. Zimdars, D. M. Chickering, C. Meek. Using Temporal Data for Making Recommendations. In Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence, UAI '01, ACM, 2001, pp. 580-588.

[9] Z. Lu, D. Agarwal, I. S. Dhillon. A Spatio-temporal Approach to Collaborative Filtering. In Proceedings of the third ACM Conference on Recommender Systems, RecSys '09, ACM, 2009, pp. 13-20.

[10] Beel J, Genzmehr M, Langer S, et al. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation. ACM, 2013, pp. 7-14.

[11] A. Lommatzsch, S. Albayrak. Real-time Recommendations for User-item Streams. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, ACM, 2015, pp. 1039-1046.

[12] J. Chen, R. Nairn, L. Nelson, M. Bernstein, E. Chi. Short and Tweet: Experiments on Recommending Content from Information Streams. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, ACM, 2010, pp. 1185-1194.

[13] K. Subbian, C. Aggarwal, K. Hegde. Recommendations for Streaming Data. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, ACM, 2016, pp. 2185-2190.

[14] S. Chang, Y. Zhang, J. Tang, D. Yin, Y.Chang, M. A. Hasegawa-Johnson, T. S. Huang. Streaming Recommender Systems. In Proceedings of the 26th International Conference on World Wide Web, WWW '17, ACM, 2017, pp. 381-389.

[15] B. Kille, A. Lommatzsch, R. Turrin, A. Serény, M. Larson, T. Brodt, J. Seiler, F. Hopfgartner. Stream-based Recommendations: Online and Offline Evaluation as A Service. In Proceedings of the 6th International Conference on Experimental IR Meets, CLEF'15, Springer-Verlag, 2015, pp. 497-517.

[16] Smith B R, Linden G D, Zada N K. Content personalization based on actions performed during a current browsing session: U.S. Patent 6,853,982. 2005-2-8.

[17] D. Jannach, M. Ludewig. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pp. 306-310. ACM, 2017.

[18] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pp. 130-137. ACM, 2017.

[19] Y. Hu, Y. Koren, C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In Proceedings of the 8th International Conference on Data Mining, ICDM '08, pp. 263-272. IEEE, 2008.

[20] G. Takács, I. Pilászy, D. Tikk. Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering. In Proceedings of the 5th ACM conference on Recommender systems, RecSys '11, pp. 297-300. ACM, 2011.

[21] P. Covington, J. Adams, E. Sargin. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, pp. 191-198. ACM, 2016.

[22] F. Hopfgartner, B. Kille, T. Heintz, R. Turrin. Real-time recommendation of streamed data. In Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, ACM, 2015, pp. 361-362.

[23] A. Gers F., J. Schmidhuber, F. Cummins. Learning to Forget: Continual Prediction with LSTM. Neural Computation, 1999.

[24] K. Cho, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP'14, 2014, pp. 1724–1734.

[25] H. Sak, A. Senior, F. Beaufays. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In Proceedings of the 15th Annual Conference of the International Speech Communication Association, INTERSPEECH '14, pp. 338-342. ISCA, 2014.

[26] K. Greff, R. Srivastava, J. Koutnik, B. Steunebrink, J. Schmidhuber. LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, 2222-2232, 2017.

[27] Y. Koren, R. Bell, C. Volinsky. Matrix Factorization Techniques for Recommender Systems. Computer, IEEE, 2009, pp. 42-49.

[28] Steam Web API, https://developer.valvesoftware.com/wiki/Steam_Web_API