#15 atmaCup
3位解法
～行列分解の正則化～

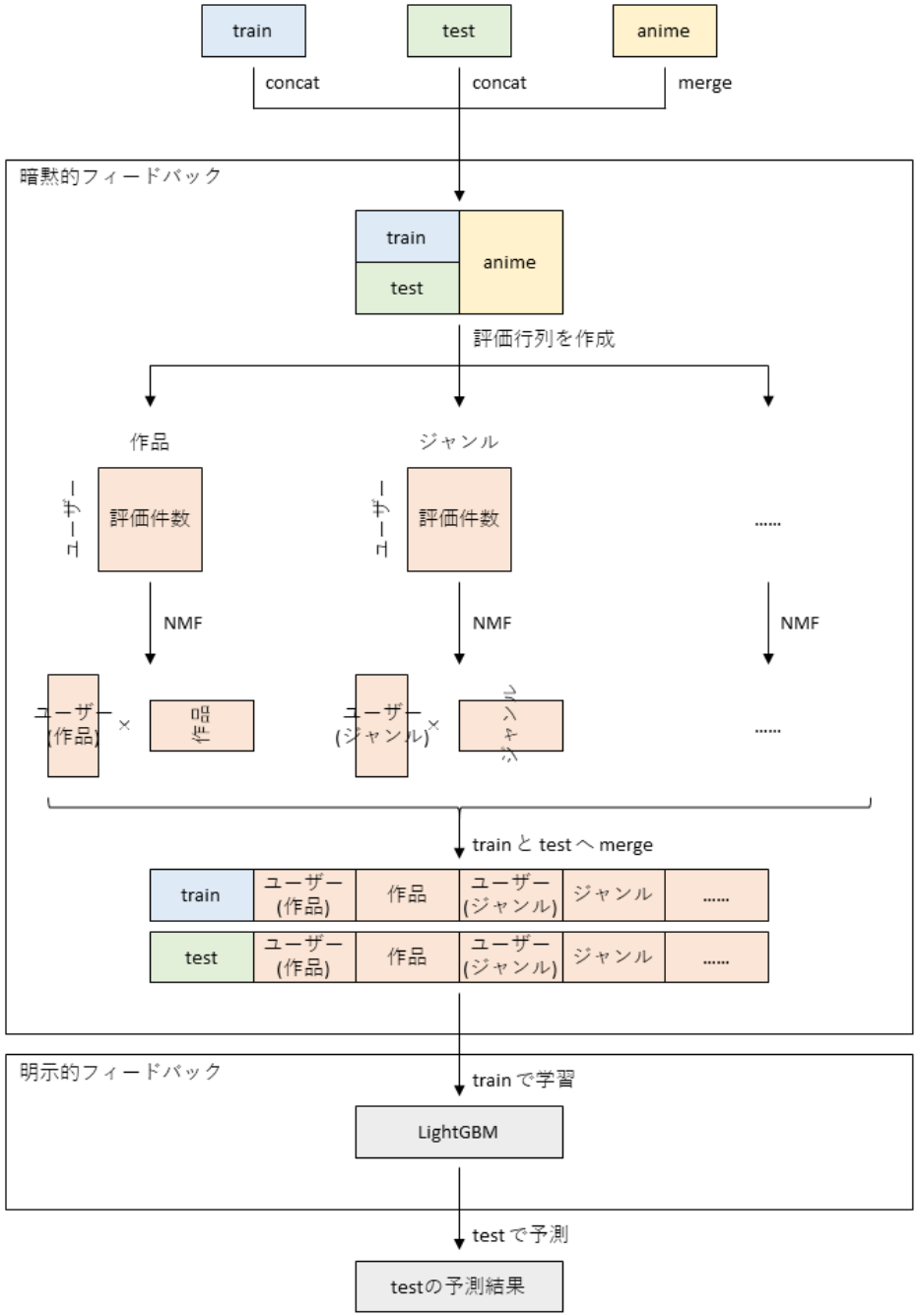
2023年8月7日

チーム marupuro（ざこぷろ、まるやま）

3位解法の概要

- LightGBMを使って
  評価値を予測する回帰モデルを作成。
- 説明変数には、
  評価行列をNMFで行列分解して作った
  ユーザーベクトルや作品ベクトルなどを使用。
  - ユーザー x 作品
  - ユーザー x ジャンル
  - ユーザー x 制作会社
  - ユーザー x ライセンス所有者
  - ユーザー x スタジオ
  - ユーザー x 作品の元ネタ
  - ユーザー x IP（作品名の先頭4文字）

詳細：ディスカッション、再現コード



3位解法の特徴的なところ

評価行列を分解する際に正則化していること。

- sklearnのNMFはデフォルトで正則化が無効になっているため、
  多くの人は正則化せずにNMFを使っているはず。
- SVDで分解した場合は、そもそも正則化という概念がない。

→ 行列分解がうまく機能しなかった人もいる中で、3位解法がうまく行った主要因では？

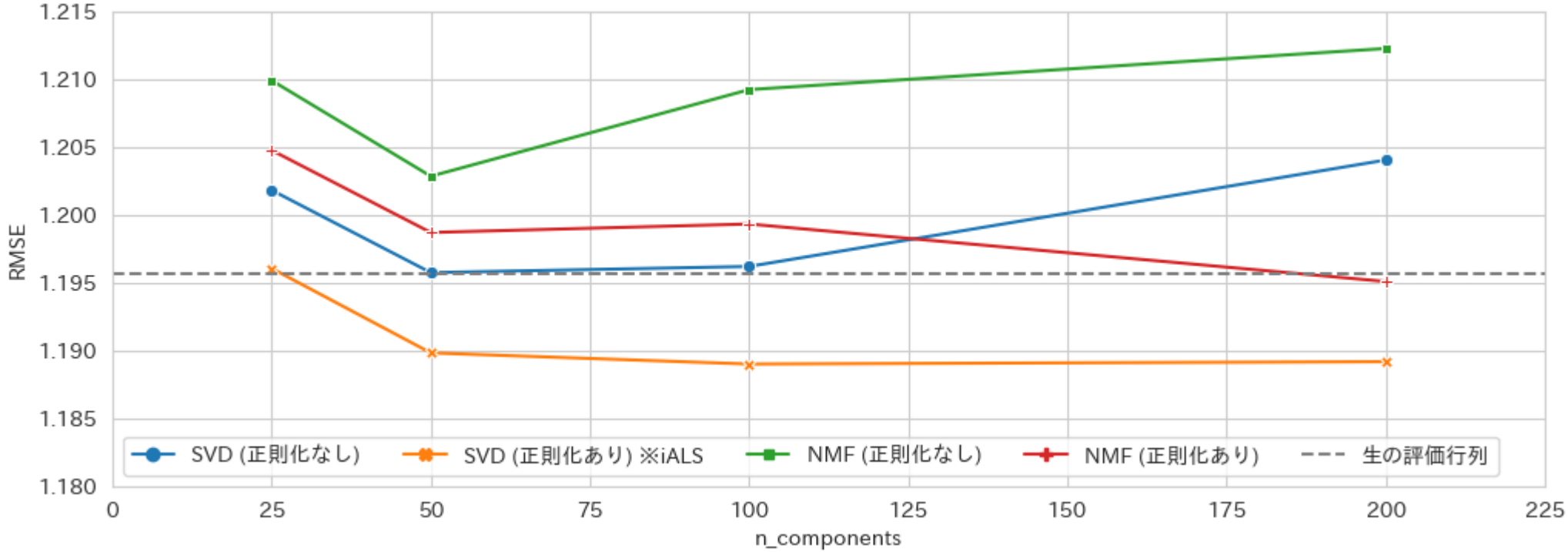
```
from sklearn.decomposition import NMF
nmf = NMF(
    n_components=50,
    alpha_W=0.01,  # 行列分解の際にL2正則化をかけている。
    max_iter=1000,
    random_state=0
)
user_embedding = nmf.fit_transform(user_anime_matrix)
anime_embedding = nmf.components_.T
```

正則化の有無の精度比較 ～正則化の重要性～

埋め込み次元数と正則化の有無を変えて精度を評価した。

- 埋め込み次元数が小さすぎるとそもそも精度が出ないが、大きすぎると過学習する。
- 正則化すれば、埋め込み次元数を大きく設定しても精度は下がらない。

→ 大きめの埋め込み次元数で表現力を担保し、正則化で過学習を抑えるのがGood。



user_id x anime_id の行列分解を説明変数に、LightGBMを学習率0.05で学習。精度はテストデータの約23%がunseenになるよう分割した5分割CVで検証。

埋め込み次元を増やす際の注意点

- 前頁の実験で正則化がうまく機能することがわかったが、
  ここで正則化しているのは「行列分解」であって「LightGBM」ではない。
- すなわち、「特徴量が増えすぎたことでLightGBMが過学習している」のではなく
  「分解した行列自体が過学習している」と考えるべき。

→ 埋め込み次元数を増やす場合、行列分解の段階で過学習対策が必要。

SVDの正則化

Truncated SVD

本資料で「SVD（正則化なし）」と呼んでいる行列分解モデル

- 特異値上位 k 個の特異ベクトルから P と Q を構成すると実は以下の L が最小になる。
- $L = \|R - PQ^T\|_F^2 = \sum_{u \in U} \sum_{i \in I} \left(r_{ui} - p_u q_i^T\right)^2$ … この L に正則化項を足せばOK！

Matrix Factorization（iALS）※1

本資料で「SVD（正則化あり）」と呼んでいる行列分解モデル

- $\min_{P,Q} L = \min_{P,Q} (L_1 + L_0 + L_R)$
- $L_1 = \sum_{(u,i) \in S} \left(1 - p_u q_i^T\right)^2$ … フィードバックが得られている部分の損失
- $L_0 = \alpha_0 \sum_{u \in U} \sum_{i \in I} \left(0 - p_u q_i^T\right)^2$ … フィードバックが得られていない部分の損失
- $L_R = \lambda \left(\sum_{u \in U} (I(u) + \alpha_0 |I|) \|p_u\|^2 + \sum_{i \in I} (U(i) + \alpha_0 |U|) \|q_i\|^2\right)$
  … 頻度で重み付けしたL2正則化項

R: 評価行列、P: ユーザー行列、p_u: ユーザー行列を成すユーザー u のベクトル、Q: アイテム行列、q_i: アイテム行列を成すアイテム i のベクトル、
U: ユーザーの集合、I: アイテムの集合、k: 埋め込み次元数（ユーザーベクトル・アイテムベクトルの次元数）、S: フィードバックの得られたユーザーとアイテムのペアの集合、
$I(u)$: ユーザー u から得られたフィードバックのアイテム数、$U(i)$: アイテム i へフィードバックを返したユーザー数。

※1 Rendle, Steffen, et al. ”Revisiting the performance of ials on item recommendation benchmarks.” RecSys 2022.

正則化の強さの精度比較 ～強さ調整の重要性～

正則化の強さを決める λ（reg）と α_0（unobserved_weight）を変えて精度を評価した。

- λ が大きすぎると、精度が大幅に悪くなる。
- λ が小さいと、α_0 を小さくしすぎたときの精度が大幅に悪くなる。

→ 正則化の強さに依存して精度が大きく変わるため、慎重に調整する必要あり。

embedding_dim	unobserved_weight	reg 0.000300	0.001000	0.003000	0.010000	0.030000	0.100000
25	0.003000	1.225640	1.225073	1.218592	1.210282	1.216207	1.263551
	0.010000	1.215758	1.212580	1.214389	1.208359	1.204781	1.244577
	0.030000	1.204021	1.204235	1.207886	1.201208	1.199634	1.231482
	0.100000	1.200756	1.200167	1.201907	1.198672	1.202038	1.244686
	0.300000	1.197640	1.196042	1.198833	1.196886	1.210001	1.277040
	1.000000	1.204123	1.201374	1.201022	1.202452	1.253001	1.413997
50	0.003000	1.219732	1.217781	1.208363	1.204562	1.213038	1.258530
	0.010000	1.210798	1.211925	1.207722	1.200411	1.203575	1.241295
	0.030000	1.205805	1.201070	1.196979	1.195655	1.195049	1.228615
	0.100000	1.199107	1.195596	1.196400	1.191127	1.195532	1.243615
	0.300000	1.192917	1.196014	1.191708	1.189812	1.202829	1.278610
	1.000000	1.193108	1.193147	1.193002	1.198186	1.255797	1.422388

embedding_dim	unobserved_weight	reg 0.000300	0.001000	0.003000	0.010000	0.030000	0.100000
100	0.003000	1.216690	1.212585	1.205447	1.200427	1.213149	1.253265
	0.010000	1.210696	1.207037	1.201990	1.197049	1.198823	1.238812
	0.030000	1.207218	1.207198	1.200316	1.195710	1.193606	1.224489
	0.100000	1.202312	1.199062	1.196946	1.191648	1.194117	1.243183
	0.300000	1.201540	1.194772	1.194287	1.188992	1.204618	1.268927
	1.000000	1.195505	1.195300	1.191550	1.194321	1.249464	1.419779
200	0.003000	1.215525	1.209433	1.196957	1.197533	1.208573	1.256686
	0.010000	1.218929	1.209286	1.199010	1.193626	1.196205	1.235596
	0.030000	1.212187	1.205897	1.200586	1.190771	1.191356	1.222503
	0.100000	1.208069	1.206640	1.200418	1.189166	1.192418	1.238127
	0.300000	1.207741	1.201418	1.197017	1.189476	1.197620	1.267516
	1.000000	1.200112	1.200070	1.195125	1.193666	1.247191	1.379751

user_id x anime_id の行列分解を説明変数に、LightGBMを学習率0.05で学習。精度はテストデータの約23%がunseenになるよう分割した5分割CVで検証。

（参考）NMFにおける正則化の強さと精度

NMFの正則化

$$
\begin{aligned}
L(W,H) = {} & 0.5 * ||X - WH||^2_{loss} \\
& + alpha\_W * l1\_ratio * n\_features * ||vec(W)||_1 \\
& + alpha\_H * l1\_ratio * n\_samples * ||vec(H)||_1 \\
& + 0.5 * alpha\_W * (1 - l1\_ratio) * n\_features * ||W||^2_{Fro} \\
& + 0.5 * alpha\_H * (1 - l1\_ratio) * n\_samples * ||H||^2_{Fro}
\end{aligned}
$$

※sklearnのドキュメント（https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html）から抜粋。

正則化の強さの精度比較

- L2正則化のみ（l1_ratio=0）かつWとHで同じ強さ（alpha_H=alpha_W）で正則化。
- 埋め込み次元数が大きい場合に正則化のご利益が大きい。

alpha_W	0.000000	0.000300	0.001000	0.003000	0.010000	0.030000	0.100000
n_components							
25	1.209895	1.209583	1.210081	1.206764	1.204748	1.232027	1.451748
50	1.202833	1.206084	1.207553	1.208667	1.198688	1.232849	1.426319
100	1.209216	1.202897	1.203727	1.199313	1.200120	1.225843	1.374419
200	1.212248	1.206905	1.202992	1.204649	1.195074	1.223843	1.357148

user_id x anime_id の行列分解を説明変数に、LightGBMを学習率0.05で学習。精度はテストデータの約23%がunseenになるよう分割した5分割CVで検証。

（参考）LBのスコアと順位

Public LB 27位相当、Private LB 35位相当。
評価行列をそのまま使うより、行列分解したほうが精度が高い（ちょっとだけ……）

#	検証方法	RMSE	備考
1	CV	1.1898	手法: iALS（$k = 50$、$\lambda = 0.01$、$\alpha_0 = 0.3$）、学習率: 0.05。
2	CV	1.1811	#1の学習率を0.01に変更。
3	Public LB	1.1920	#2をsubmit。27位相当。
4	Private LB	1.1623	#2をsubmit。35位相当。
5	CV	1.1957	評価行列をそのまま説明変数に使用（12位解法）、学習率: 0.05。
6	CV	1.1834	#5の学習率を0.01に変更。
7	Public LB	※1 1.1951	#6をsubmit。37位相当。
8	Private LB	※1 1.1634	#6をsubmit。40位相当。

※1 本実験ではseenとunseenを分けていないため、12位解法で報告されているスコア（Public LB 1.1882 / Private LB 1.1562）よりも悪くなっている点に注意。

まとめ

行列分解は正則化が大事！

- 埋め込み次元が小さいと精度が頭打ちになるため、
  とりあえず埋め込み次元は大きく設定する。
- 埋め込み次元を大きく設定すると過学習のリスクが高まるため、
  正則化して過学習を抑える。
- 正則化の強さが精度を大きく左右するため、
  正則化関連のハイパーパラメーター調整は慎重に行う。