Sistem de recomandare folosind factorizare de matrice și similaritate pe taguri

Gatej Stefan-Alexandru, Potop Horia-Ioan

May 27, 2025

Contents

1 Descrierea aplicației

Scopul acestei componente a sistemului de recomandare este de a genera recomandări de melodii similare pe baza caracteristicilor extrase din tagurile asociate fiecărei piese muzicale. Problema centrală abordată este factorizarea unei matrice binare (melodie × tag) într-un spațiu latent de dimensiune redusă, sub constrângeri de non-negativitate. Această factorizare permite descoperirea unor structuri și similarități ascunse între melodii.

2 Formularea matematică a problemei de factorizare

Problema de bază pentru metodele de factorizare de matrice (NMF, ALS-NNLS) este următoarea: Fie $V \in \mathbb{R}^{m \times n}$ matricea de intrare (în cazul nostru, matricea binară melodie × tag), unde m este numărul de melodii și n este numărul de taguri unice. Dorim să găsim două matrice, $W \in \mathbb{R}^{m \times r}$ (matricea factorilor latenti pentru melodii) și $H \in \mathbb{R}^{r \times n}$ (matricea factorilor latenti pentru taguri), unde $r \ll \min(m, n)$ este dimensiunea spațiului latent (numărul de factori).

Problema de optimizare este de a minimiza eroarea de reconstrucție, adesea măsurată prin norma Frobenius:

$$\min_{W,H} \|V - WH\|_F^2 \tag{1}$$

Constrângeri de Non-Negativitate: Pentru a asigura interpretabilitatea factorilor și pentru a se alinia cu natura aditivă a multor date (cum ar fi prezența tagurilor), se impun constrângeri de non-negativitate:

- $W_{ij} \geq 0$ pentru toți i, j
- $H_{ij} \geq 0$ pentru toți i, j

2.1 Alegerea numărului de factori latenti (r)

Numărul de factori latenti, r, este un hiperparametru crucial în algoritmii de factorizare de matrice, deoarece determină dimensionalitatea spațiului latent în care datele sunt proiectate. O valoare prea mică pentru r poate duce la un model sub-antrenat (underfitting) care nu capturează suficientă informație din datele originale, rezultând într-o eroare de reconstrucție mare și recomandări de calitate inferioară. Pe de altă parte, o valoare prea mare pentru r poate duce la supra-antrenare (overfitting), unde modelul începe să memoreze zgomotul specific setului de date de antrenament și nu generalizează bine pe date noi. De asemenea, un r mare crește complexitatea computațională a antrenării și a utilizării modelului.

În cadrul acestui proiect, pentru a permite o comparație directă și consistentă între diferitele metode de factorizare implementate (NMF Multiplicativ, NMF `sklearn`, ALS-NNLS), am optat pentru o valoare fixă $r = 10$. Această valoare a fost aleasă ca un punct de pornire rezonabil pentru a demonstra funcționalitatea algoritmilor pe setul de date disponibil, permițând în același timp timpi de execuție gestionabili, în special pentru implementările manuale care sunt mai intensive computațional.

Într-o aplicație practică sau într-un studiu mai aprofundat, selectarea optimă a lui r ar necesita o abordare mai sistematică, implicând experimentarea cu o gamă de valori

pentru r și evaluarea performanței modelului folosind metrici adecvate. Metodele comune includ:

- **Monitorizarea erorii de reconstrucție:** Se analizează cum evoluează eroarea de reconstrucție (de ex., norma Frobenius a diferenței $V - WH$) pe măsură ce r crește. Se caută adesea un punct de inflexiune ("cot" sau "elbow point") în graficul erorii vs. r, dincolo de care adăugarea de noi factori nu mai aduce o reducere semnificativă a erorii.
- **Metrici de evaluare specifice sarcinii:** Dacă scopul final este recomandarea și sunt disponibile date de feedback ale utilizatorilor (ex: ratinguri, istoric de ascultare), se pot utiliza metrici de ranking precum Precision@k, Recall@k, NDCG (Normalized Discounted Cumulative Gain) sau MAP (Mean Average Precision) pe un set de date de validare sau test. Valoarea lui r care maximizează aceste metrici ar fi preferată.
- **Stabilitatea factorizării:** Se poate evalua cât de stabile sunt factorizările obținute pentru aceeași valoare r la rulări multiple cu inițializări aleatorii diferite. O stabilitate mai mare este de dorit.
- **Interpretabilitatea factorilor:** În unele aplicații, se preferă un r care produce factori latenti (coloanele lui W sau liniile lui H) care au o interpretare semantică clară și relevantă pentru experții în domeniu.

Pentru scopurile acestei lucrări, ne-am concentrat pe compararea fundamentală a algoritmilor de optimizare cu constrângeri pentru un r fix, recunoscând că optimizarea detaliată a acestui hiperparametru este un pas important, dar distinct, într-un scenariu de dezvoltare a unui sistem de recomandare de producție.

2.2 Rolul matricelor W și H în sistemul de recomandare

În urma procesului de factorizare, obținem două matrice, W și H, care au interpretări și utilizări distincte în contextul recomandărilor:

- **Matricea $W \in \mathbb{R}^{m \times r}$ (Melodii $\times$ Factori Latenți):** Fiecare linie W_i din această matrice reprezintă **profilul latent** al melodiei i. Acest vector r-dimensional încorporează caracteristicile "ascunse" ale melodiei, așa cum au fost ele învățate de algoritm. Pentru recomandările de tip melodie-melodie, așa cum sunt implementate în acest proiect, similaritatea dintre aceste profiluri latente (de exemplu, similaritatea cosinus între W_i și W_j) este utilizată pentru a identifica melodii similare. Astfel, W este matricea principală utilizată direct în funcția de generare a recomandărilor.
- **Matricea $H \in \mathbb{R}^{r \times n}$ (Factori Latenți $\times$ Taguri):** Această matrice leagă factorii latenti de tagurile originale. Fiecare coloană H_j poate fi interpretată ca profilul latent al tagului j, iar fiecare linie H_k arată importanța sau "încărcătura" factorului latent k pentru fiecare tag original. Matricea H este crucială pentru:
  - **Interpretarea factorilor latenți:** Analizând liniile lui H, putem încerca să dăm un sens semantic fiecărui factor latent (ex: un factor ar putea fi puternic asociat cu taguri de "rock clasic", altul cu "muzică electronică ambientală").

– **Recomandarea de taguri pentru melodii:** Produsul $W_i H$ poate fi folosit pentru a prezice relevanța tagurilor pentru melodia i.
– **Găsirea de taguri similare:** Similaritatea dintre coloanele lui H poate identifica taguri care sunt conceptual apropiate în spațiul latent.
– **Calitatea factorizării:** Ambele matrice, W și H, sunt esențiale pentru reconstrucția $V \approx WH$ și, implicit, pentru calitatea generală a modelului.

Deși funcția noastră principală de recomandare utilizează direct matricea W, matricea H este o componentă indispensabilă a procesului de învățare și oferă posibilități valoroase de analiză și extindere a sistemului.

Concepte cheie: Profil latent și Cluster latent

Profil latent Un profil latent este o reprezentare numerică (un vector) a unei entități (de exemplu, o melodie) în spațiul de factori ascunși de dimensiune r, descoperit de algoritmul de factorizare. Liniile matricei W (adică W_i) reprezintă aceste profiluri latente pentru melodii. Acești factori nu sunt direct observați în datele inițiale, ci sunt învățați pentru a surprinde structuri subiacente. De exemplu, un factor ar putea corespunde unui "grad de energie" al melodiei, altul unui "stil vocal specific", etc., chiar dacă aceste concepte nu sunt taguri explicite.

Cluster latent Un cluster latent se referă la un grup de entități (melodii) care au profiluri latente similare, adică sunt apropiate în spațiul latent r-dimensional. Deși nu aplicăm explicit un algoritm de clustering separat după NMF/ALS în această componentă pentru a defini clustere discrete, ideea este că melodiile cu vectori W_i similari sunt considerate conceptual parte dintr-un "cluster" sau o regiune densă în spațiul latent. Recomandările se bazează pe această proximitate.

Pe scurt: Profilul latent este vectorul de factori ascunși care descrie o melodie, iar similaritatea dintre aceste profiluri stă la baza recomandărilor bazate pe factorizare.

3 Soluții implementate pentru factorizarea cu constrângeri

Am implementat și comparat trei algoritmi pentru rezolvarea problemei de factorizare de matrice cu constrângeri de non-negativitate:

3.1 Algoritmul multiplicativ pentru NMF (NMF-MULT)

Acesta este un algoritm clasic pentru NMF, propus de Lee și Seung. El actualizează iterativ matricele W și H folosind următoarele reguli multiplicative, care garantează noncreșterea funcției obiectiv (1) și menținerea non-negativității:

$$H_{kj} \leftarrow H_{kj} \frac{(W^T V)_{kj}}{(W^T W H + \epsilon)_{kj}} \tag{2}$$

$$W_{ik} \leftarrow W_{ik} \frac{(V H^T)_{ik}}{(W H H^T + \epsilon)_{ik}} \tag{3}$$

unde ϵ este o constantă mică pentru a evita diviziunea cu zero.

- **Avantaje:** Simplu de implementat, relativ rapid pentru multe seturi de date.
- **Dezavantaje:** Poate converge lent și la minime locale.

3.2 NMF din scikit-learn (`sklearn.decomposition.NMF`)

Am utilizat implementarea NMF din biblioteca `scikit-learn`, care oferă mai mulți solveri. Am ales solver-ul 'cd' (Coordinate Descent), care este eficient și adesea converge mai rapid decât regulile multiplicative simple pentru anumite tipuri de date, minimizând aceeași funcție obiectiv (1) sub constrângeri de non-negativitate.

- **Avantaje:** Implementare optimizată, robustă, cu opțiuni pentru regularizare.
- **Solver:** Coordinate Descent actualizează fiecare element dintr-o matrice (ținând cealaltă fixă) rezolvând o subproblemă de optimizare univariată.

3.3 Alternating Least Squares cu Non-Negative Least Squares (ALS-NNLS)

ALS este o altă abordare populară pentru factorizarea de matrice. Algoritmul optimizează alternativ pentru W (ținând H fix) și apoi pentru H (ținând W fix), minimizând funcția obiectiv (1) la fiecare pas. Fiecare subproblemă devine o problemă de least squares. Pentru a impune constrângerile de non-negativitate, fiecare subproblemă de least squares este rezolvată ca o problemă de Non-Negative Least Squares (NNLS).

- **Subproblema pentru W (pentru fiecare linie W_i):**

$$\min_{W_i \geq 0} \|V_i - W_i H\|_F^2$$

- **Subproblema pentru H (pentru fiecare coloană H_j):**

$$\min_{H_{\cdot j} \geq 0} \|V_{\cdot j} - W H_{\cdot j}\|_F^2$$

Am implementat manual atât logica ALS, cât și un solver NNLS bazat pe metoda proiectării gradientului.

NNLS Manual (Proiectarea Gradientului): Pentru a rezolva $\min_{x \geq 0} \|Ax - b\|_2^2$:

1. Se inițializează $x \geq 0$.
2. Se calculează gradientul funcției obiectiv $\frac{1}{2}\|Ax - b\|_2^2$: $\nabla_x = A^T(Ax - b)$.
3. Se face un pas în direcția opusă gradientului: $x \leftarrow x - \eta \nabla_x$.
4. Se proiectează x înapoi pe domeniul fezabil: $x_j \leftarrow \max(0, x_j)$.
5. Se repetă pașii 2-4 până la convergență.

- **Avantaje:** Demonstrează înțelegerea detaliată a optimizării cu constrângeri și a descompunerii problemei.
- **Dezavantaje:** Implementarea manuală (mai ales NNLS) poate fi lentă și sensibilă la parametrii precum rata de învățare.

4 Rezultate numerice și concluzii

4.1 Erori de reconstrucție

Pentru un număr de factori latenti $r = 10$, erorile de reconstrucție (norma Frobenius $\|V - WH\|_F$) obținute au fost:

- **NMF Multiplicativ (Manual):** 337.03
- **NMF (sklearn, solver 'cd'):** 336.42
- **ALS (NNLS Manual):** 430.48

Se observă că implementarea din `sklearn` obține o eroare de reconstrucție puțin mai mică, probabil datorită optimizărilor interne și a robusteții solver-ului Coordinate Descent. Algoritmul multiplicativ obține rezultate comparabile. Implementarea ALS-NNLS manuală prezintă o eroare semnificativ mai mare, indicând posibile probleme de convergență în subproblemele NNLS manuale, în special legate de alegerea ratei de învățare și a numărului de iterații pentru solver-ul de proiectare a gradientului.

4.2 Recomandări generate (Exemplu)

Pentru a ilustra tipurile de recomandări generate de fiecare metodă, am selectat piesa de referință **"Murder One" de Metallica**. Această piesă este un tribut adus lui Lemmy Kilmister de la Motörhead și se încadrează în genul Thrash Metal/Heavy Metal. Mai jos sunt prezentate top 5 recomandări pentru fiecare abordare implementată:

- **Referință: Murder One - Metallica**
  - *Recomandări (Tag Similarity - Similaritate Cosinus pe taguri):*
    - Necrophiliac - Slayer
    - Blacklist - Exodus
    - Electric Crown - Testament
    - Murder Fantasies - Kreator
    - Practice What You Preach - Testament
  - *Recomandări (NMF Multiplicativ Manual):*
    - Crionics - Slayer
    - Praise of Death - Slayer
    - Blackmail The Universe - Megadeth
    - Washington Is Next! - Megadeth
    - Back In the Day - Megadeth
  - *Recomandări (NMF `sklearn`, solver 'cd'):*
    - Blacklist - Exodus
    - At Dawn They Sleep - Slayer
    - Bite The Hand - Megadeth
    - Uplift - Pantera

    - Praise of Death - Slayer

- *Recomandări (ALS cu NNLS Manual):*
    - Practice What You Preach - Testament
    - Bite The Hand - Megadeth
    - Hy Pro Glo - Anthrax
    - Deviance - Slayer
    - Tormentor - Slayer

Observații asupra exemplului: Se poate observa că toate metodele tind să recomande piese din subgenuri similare ale metalului (Thrash, Speed, Heavy Metal), ceea ce este de așteptat.

- **Similaritatea pe taguri** pare să identifice corect piese cu tematici sau stiluri lirice/muzicale apropiate, menționate explicit în taguri (ex: Kreator - Murder Fantasies).
- **Metodele NMF** (atât manuală, cât și `sklearn`) și **ALS** generează recomandări care, deși tot din sfera metal, pot proveni de la artiști diferiți, indicând descoperirea unor profiluri latente comune. De exemplu, prezența consistentă a pieselor de la Slayer și Megadeth în recomandările NMF și ALS sugerează că aceste trupe partajează caracteristici latente similare cu Metallica în contextul datelor analizate.
- Diferențele subtile între recomandările NMF și ALS pot fi atribuite modului diferit în care algoritmii converg și ponderează factorii latenti, precum și performanței de reconstrucție a fiecărei metode.

Acest exemplu ilustrează capacitatea metodelor de factorizare de a merge dincolo de potrivirile directe de taguri și de a sugera melodii relevante pe baza unor similarități structurale mai profunde.

Recomandările bazate pe factorizare (NMF, ALS) tind să identifice similarități latente. Calitatea recomandărilor ALS-NNLS este direct influențată de performanța sa în minimizarea erorii de reconstrucție.

4.3 Analiza convergenței și a erorilor de reconstrucție

Performanța algoritmilor de factorizare este adesea evaluată prin eroarea de reconstrucție și prin modul în care aceasta evoluează pe parcursul iterațiilor.

NMF Multiplicativ (Manual): După cum se observă din log-ul de execuție, algoritmul NMF multiplicativ prezintă o scădere monotonă a erorii de reconstrucție pe măsură ce numărul de iterații crește:

```
Multiplicative NMF Iteration 0: error = 403.9834
Multiplicative NMF Iteration 10: error = 350.3428
...
Multiplicative NMF Iteration 90: error = 337.0462
```

Eroarea finală raportată a fost de **337.03**. Această convergență graduală este caracteristică regulilor de update multiplicative.

NMF (`sklearn`, solver 'cd'): Implementarea din `sklearn` cu solver-ul Coordinate Descent a raportat direct o eroare finală de reconstrucție de **336.42**. Acest solver este adesea mai rapid în a atinge un minim local bun datorită optimizărilor sale.

ALS cu NNLS Manual (ALS-NNLS): Pentru implementarea ALS cu subprobleme NNLS rezolvate manual, s-a observat următorul comportament:

```
ALS-NNLS Iteration 0: Frobenius error = 426.8031
ALS-NNLS Iteration 5: Frobenius error = 430.4727
ALS-NNLS converged at iteration 8.
```

Eroarea finală de reconstrucție raportată a fost de aproximativ **430.48**. Se observă că eroarea inițială este mai mare comparativ cu NMF și, în mod interesant, eroarea a crescut ușor între iterația 0 și iterația 5, înainte de a se stabiliza și a algoritmului să conveargă la iterația 8. Această creștere temporară a erorii poate apărea în ALS dacă subproblemele NNLS nu sunt rezolvate cu suficientă precizie la fiecare pas, sau dacă rata de învățare aleasă pentru NNLS manual (`lr_nnls = 1e-4`) și numărul de iterații interne NNLS (`nnls_iters = 50`) nu sunt optimi pentru toate subproblemele întâlnite.

Este important de menționat că parametrii pentru ALS-NNLS manual (precum `als_iters = 30`, `nnls_iters = 50`, `lr_nnls = 1e-4`) au fost aleși pentru a obține un echilibru între calitatea soluției și timpul de execuție, în special având în vedere natura computațional intensivă a rezolvării manuale a multiplelor subprobleme NNLS. O explorare mai exhaustivă a spațiului de hiperparametri pentru `lr_nnls` și `nnls_iters` ar putea duce la o eroare de reconstrucție mai mică pentru ALS-NNLS, dar ar crește semnificativ timpul de rulare. Valoarea de "Explained variance" raportată pentru ALS-NNLS este foarte mică, ceea ce este consistent cu o eroare de reconstrucție mare și sugerează că modelul, în configurația actuală, nu a reușit să captureze la fel de bine structura datelor ca metodele NMF.

4.4 Erori de reconstrucție (Sumar)

Pentru un număr de factori latenti $r = 10$, erorile de reconstrucție (norma Frobenius $\|V - WH\|_F$) obținute au fost:

- **NMF Multiplicativ (Manual):** 337.03
- **NMF (`sklearn`, solver 'cd'):** 336.42
- **ALS (NNLS Manual):** 430.48

4.5 Timp de execuție și Complexitate Computațională

- **NMF (`sklearn`):** Cel mai rapid, datorită implementării optimizate în C și a eficienței algoritmului Coordinate Descent.
- **NMF Multiplicativ (Manual):** Relativ rapid. Fiecare iterație implică operații matriciale, dar nu rezolvarea unor subprobleme de optimizare distincte.
- **ALS (NNLS Manual):** Cel mai lent. Fiecare iterație a buclei principale ALS necesită rezolvarea a $m + n$ subprobleme NNLS. Deoarece fiecare subproblemă NNLS este rezolvată printr-un algoritm iterativ (proiectarea gradientului), costul computațional total este semnificativ mai mare. Timpul de execuție este direct

influențat de `max_iter_als`, `max_iter_nnls` și de eficiența implementării NNLS. Parametrii aleși în experimentele noastre au vizat un compromis pentru a permite finalizarea într-un timp rezonabil.

4.6 Concluzii specifice metodelor

- Metodele NMF (atât implementarea manuală multiplicativă, cât și cea din `sklearn`) au demonstrat capacitatea de a reduce eroarea de reconstrucție la valori comparabile, `sklearn` fiind marginal superior și mai rapid.
- Implementarea ALS cu NNLS manual a prezentat o eroare de reconstrucție considerabil mai mare în configurația testată. Acest rezultat subliniază provocările asociate cu implementarea manuală a algoritmilor de optimizare iterativi imbricați, în special sensibilitatea la hiperparametri precum rata de învățare și numărul de iterații pentru subproblemele NNLS. Deși ALS este o metodă puternică, performanța sa aici a fost limitată de eficiența și convergența solver-ului NNLS manual. Alegerea parametrilor a fost un compromis pentru a asigura finalizarea execuției într-un interval de timp acceptabil pentru testare.
- Toate metodele de factorizare generează profiluri latente care pot fi utilizate pentru recomandări, dar calitatea acestora (și, implicit, a recomandărilor) este corelată cu capacitatea metodei de a minimiza eroarea de reconstrucție.

5 Comparare cu alte funcții Python

Pentru problema de factorizare non-negativă, funcțiile generice de optimizare cu constrângeri din `scipy.optimize.minimize` (ex: cu solver-ul 'SLSQP' sau 'L-BFGS-B' cu bounds) ar putea fi teoretic aplicate. Totuși, acestea ar necesita "aplatizarea" matricelor W și H într-un singur vector de variabile, ceea ce ar duce la un număr foarte mare de variabile ($m \times r + r \times n$). Pentru astfel de probleme la scară mare cu structură specifică (factorizare de matrice), algoritmii dedicați precum NMF multiplicativ, Coordinate Descent (din `sklearn`), sau ALS sunt semnificativ mai eficienți. Am ales să implementăm ALS cu NNLS manual ca o altă metodă specifică problemei, dar care ilustrează optimizarea iterativă a subproblemelor cu constrângeri.

6 Bibliografie

- Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems, 13*.
- Chih-Jen Lin. (2007). Projected Gradient Methods for Non-negative Matrix Factorization. *Neural Computation, 19(10)*, 2756-2779. (Referință pentru metode de gradient pentru NMF/NNLS)
- Zdunek, R., & Cichocki, A. (2007). Non-negative matrix factorization with quadratic programming. *Computers & Mathematics with Applications, 54(1)*, 70-83. (Discută despre NNLS în context NMF)

- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html
- https://en.wikipedia.org/wiki/Non-negative_matrix_factorization
- https://en.wikipedia.org/wiki/Alternating_least_squares