

**ISEL**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
ÁREA DEPARTAMENTAL DE MATEMÁTICA

Fatorização de Matrizes em Sistemas de Recomendação

Otimização de Modelos Latentes: Tuning de Hiperparâmetros e Análise de Desempenho Computacional em Ambiente Restrito

Pedro Baltasar Nogueira

49821

Métodos Matemáticos em Inteligência Artificial

16 de dezembro de 2025

Resumo

Este relatório apresenta a implementação e comparação de dois métodos de fatorização de matrizes para dados de recomendação implícita: Non-negative Matrix Factorization (NMF) e Weighted Matrix Factorization (WMF). Ambos os modelos foram treinados utilizando Stochastic Gradient Descent (SGD). O estudo foca-se não apenas na precisão dos modelos ou na análise de perfis, mas também numa análise crítica da eficiência computacional, contrastando uma implementação manual baseada em NumPy com bibliotecas otimizadas, num contexto de limitações de hardware (computador portátil) para um universo de estudo de dimensão superior a 25 milhões de interações.

1 Introdução

Os sistemas de recomendação modernos lidam frequentemente com *feedback implícito* (cliques, visualizações, compras), onde a ausência de interação não implica necessariamente desagrado. Este projeto explora métodos latentes para modelar estas preferências.

Os objetivos principais deste trabalho são:

1. Implementar NMF e WMF utilizando otimização por SGD com *mini-batches*;
2. Implementar a estratégia de *Negative Sampling* para o método WMF;
3. Comparar a performance e eficiência de uma implementação manual contra bibliotecas padrão otimizadas “*scikit-learn*”, “*implicit*”, “*PyTorch*”, ...;
4. Analisar o impacto dos hiperparâmetros k e α , entre outros, na qualidade das recomendações.

2 Formulação Matemática

Na presente secção apresentam-se as formulações matemáticas dos métodos NMF e WMF, detalhando os problemas de otimização, funções objetivo e algoritmos de treino baseados em SGD com mini-batches para dados de feedback implícito.

2.1 Nonnegative Matrix Factorization Method (NMF)

Dada uma matriz de interações implícitas $\mathbf{R} \in \mathbb{R}^{m \times n}$, onde m representa o número de utilizadores e n o número de itens, pretende-se encontrar uma fatorização não-negativa:

$$\mathbf{R} \approx \mathbf{U}\mathbf{V}^T, \quad (1)$$

onde $\mathbf{U} \in \mathbb{R}^{m \times r}$ representa os fatores latentes dos utilizadores e $\mathbf{V} \in \mathbb{R}^{n \times r}$ representa os fatores latentes dos itens, com dimensão latente fixa $r = 20$.

2.1.1 Conjunto de Interações Observadas

Para dados de feedback implícito, considera-se apenas as interações efetivamente observadas. Define-se o conjunto:

$$\Omega = \{(u, i) : R_{ui} > 0\}, \quad (2)$$

que contém todos os pares (utilizador, item) onde existe interação registada.

2.1.2 Esparsidade da Matriz

A matriz \mathbf{R} é tipicamente esparsa, isto é, a maioria dos seus elementos são zeros. A esparsidade de uma matriz é definida como a proporção de entradas nulas:

$$\text{Esparsidade} = \frac{\text{número de zeros}}{m \times n} = \frac{mn - |\Omega|}{mn} = 1 - \frac{|\Omega|}{mn}. \quad (3)$$

Em sistemas de recomendação, cada utilizador interage apenas com uma pequena fração dos itens disponíveis, resultando em $|\Omega| \ll mn$. Por exemplo, se um utilizador consumiu 100 filmes de um catálogo de 10 000, apenas 1% das entradas correspondentes a esse utilizador são não-nulas. Esta esparsidade é uma característica fundamental dos dados de feedback implícito e justifica a abordagem de treinar apenas nas interações observadas Ω , ignorando a vasta maioria de pares não observados.

2.1.3 Função Objetivo

O problema de otimização consiste em minimizar a seguinte função de custo:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \quad & \mathcal{J}(\mathbf{U}, \mathbf{V}) = \frac{1}{2} \sum_{(u,i) \in \Omega} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i)^2 + \frac{\lambda}{2} (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) \\ \text{sujeito a} \quad & U_{uf} \geq 0, \quad \forall u \in \{1, \dots, m\}, f \in \{1, \dots, r\} \\ & V_{if} \geq 0, \quad \forall i \in \{1, \dots, n\}, f \in \{1, \dots, r\} \end{aligned} \quad (4)$$

onde:

- O primeiro termo mede o erro de reconstrução apenas nas interações observadas;
- $\|\cdot\|_F$ denota a norma de Frobenius, definida como $\|\mathbf{A}\|_F^2 = \sum_{ij} A_{ij}^2$;
- $\lambda \geq 0$ é o coeficiente de regularização L2 que controla o overfitting.

2.2 Otimização do Método NMF por SGD Mini-batch

2.2.1 Função Objetivo do Mini-batch

O conjunto de interações Ω é particionado em mini-batches $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_N$. Para um mini-batch $\mathcal{B}_t \subseteq \Omega$ de tamanho $b = |\mathcal{B}_t|$, a função de custo aproximada é:

$$\mathcal{J}_{\mathcal{B}_t}(\mathbf{U}, \mathbf{V}) = \frac{1}{2b} \sum_{(u,i) \in \mathcal{B}_t} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i)^2 + \frac{\lambda}{2} (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2). \quad (5)$$

2.2.2 Cálculo dos Gradientes

Define-se o erro de predição para cada par (u, i) :

$$e_{ui} = R_{ui} - \mathbf{U}_u^T \mathbf{V}_i = R_{ui} - \sum_{f=1}^r U_{uf} V_{if} \quad . \quad (6)$$

2.2.2.1 Derivadas Parciais em relação a \mathbf{U}

Para o fator U_{uf} de um utilizador u presente no *mini-batch*:

$$\frac{\partial \mathcal{J}_{\mathcal{B}_t}}{\partial U_{uf}} = -\frac{1}{b} \sum_{i:(u,i) \in \mathcal{B}_t} e_{ui} \cdot V_{if} + \lambda U_{uf} \quad . \quad (7)$$

Em forma vetorial, o gradiente completo do vetor de fatores do utilizador u é:

$$\nabla_{\mathbf{U}_u} \mathcal{J}_{\mathcal{B}_t} = -\frac{1}{b} \sum_{i:(u,i) \in \mathcal{B}_t} e_{ui} \cdot \mathbf{V}_i + \lambda \mathbf{U}_u \quad . \quad (8)$$

2.2.2.2 Derivadas Parciais em relação a \mathbf{V}

Para o fator V_{if} de um item i presente no *mini-batch*:

$$\frac{\partial \mathcal{J}_{\mathcal{B}_t}}{\partial V_{if}} = -\frac{1}{b} \sum_{u:(u,i) \in \mathcal{B}_t} e_{ui} \cdot U_{uf} + \lambda V_{if} \quad . \quad (9)$$

Em forma vetorial, o gradiente completo do vetor de fatores do item i é:

$$\nabla_{\mathbf{V}_i} \mathcal{J}_{\mathcal{B}_t} = -\frac{1}{b} \sum_{u:(u,i) \in \mathcal{B}_t} e_{ui} \cdot \mathbf{U}_u + \lambda \mathbf{V}_i \quad . \quad (10)$$

2.2.3 Regras de Atualização

Os fatores são atualizados seguindo a direção oposta ao gradiente (para minimizar o erro), com projeção no cone não-negativo. Para cada utilizador u e item i presentes no *mini-batch* \mathcal{B}_t :

$$\mathbf{U}_u \leftarrow \max(0, \mathbf{U}_u - \eta \nabla_{\mathbf{U}_u} \mathcal{J}_{\mathcal{B}_t}) \quad , \quad (11)$$

$$\mathbf{V}_i \leftarrow \max(0, \mathbf{V}_i - \eta \nabla_{\mathbf{V}_i} \mathcal{J}_{\mathcal{B}_t}) \quad , \quad (12)$$

onde $\eta > 0$ é a taxa de aprendizagem.

Nota sobre o operador \max : O operador $\max(0, \cdot)$, aplicado elemento a elemento, não maximiza o erro mas sim garante a não-negatividade dos fatores. O gradiente descendente (subtração $-\eta \nabla$) minimiza o erro, mas pode produzir valores negativos. O $\max(0, \cdot)$ projeta esses valores negativos de volta para zero, mantendo a restrição $U_{uf} \geq 0$ e $V_{if} \geq 0$ necessária para o NMF. Formalmente, esta é uma projeção no cone não-negativo:

$$\max(0, x) = \begin{cases} x & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases} \quad (13)$$

Algorithm 1 NMF com SGD Mini-*batch*

```
1: Entrada:  $\mathbf{R}, \Omega, r, \lambda, \eta, b, n_{\text{epochs}}$ 
2: Saída:  $\mathbf{U}, \mathbf{V}$ 
3:
4: Inicializar  $\mathbf{U}, \mathbf{V} \geq 0$  aleatoriamente
5:
6: for  $e = 1$  to  $n_{\text{epochs}}$  do
7:   Embaralhar  $\Omega$  aleatoriamente
8:   Particionar  $\Omega$  em mini-batches  $\mathcal{B}_1, \dots, \mathcal{B}_N$  de tamanho  $b$ 
9:
10:  for  $t = 1$  to  $N$  do
11:    // Calcular erros
12:    for cada  $(u, i) \in \mathcal{B}_t$  do
13:       $e_{ui} \leftarrow R_{ui} - \mathbf{U}_u^T \mathbf{V}_i$ 
14:    end for
15:
16:    // Atualizar fatores dos utilizadores
17:    for cada  $u$  que aparece em  $\mathcal{B}_t$  do
18:       $\nabla \mathbf{U}_u \leftarrow -\frac{1}{b} \sum_{i:(u,i) \in \mathcal{B}_t} e_{ui} \cdot \mathbf{V}_i + \lambda \mathbf{U}_u$ 
19:       $\mathbf{U}_u \leftarrow \max(0, \mathbf{U}_u - \eta \nabla \mathbf{U}_u)$ 
20:    end for
21:
22:    // Atualizar fatores dos itens
23:    for cada  $i$  que aparece em  $\mathcal{B}_t$  do
24:       $\nabla \mathbf{V}_i \leftarrow -\frac{1}{b} \sum_{u:(u,i) \in \mathcal{B}_t} e_{ui} \cdot \mathbf{U}_u + \lambda \mathbf{V}_i$ 
25:       $\mathbf{V}_i \leftarrow \max(0, \mathbf{V}_i - \eta \nabla \mathbf{V}_i)$ 
26:    end for
27:  end for
28: end for
29:
30: return  $\mathbf{U}, \mathbf{V}$ 
```

2.3 Weighted Matrix Factorization (WMF)

O método *Weighted Matrix Factorization*, proposto por Hu, Koren e Volinsky (2008), estende a fatorização de matrizes tradicional ao incorporar pesos diferenciados para cada interação, permitindo modelar a confiança associada a cada observação em dados de feedback implícito.

2.3.1 Problema de Minimização

Dada a mesma matriz de interações $\mathbf{R} \in \mathbb{R}^{m \times n}$, pretende-se encontrar a fatorização:

$$\mathbf{R} \approx \mathbf{U}\mathbf{V}^T, \quad (14)$$

onde $\mathbf{U} \in \mathbb{R}^{m \times r}$ e $\mathbf{V} \in \mathbb{R}^{n \times r}$, com $r = 20$.

2.3.2 Matriz de Pesos

Ao contrário do NMF, o WMF considera todas as entradas da matriz (observadas e não observadas), atribuindo-lhes pesos diferenciados. Define-se a matriz de pesos $\mathbf{C} \in \mathbb{R}^{m \times n}$ como:

$$C_{ui} = 1 + \alpha R_{ui}, \quad (15)$$

onde $\alpha > 0$ é um hiperparâmetro que controla a importância relativa das interações observadas. Note-se que:

- Para interações não observadas ($R_{ui} = 0$): $C_{ui} = 1$ (peso base)
- Para interações observadas ($R_{ui} > 0$): $C_{ui} = 1 + \alpha R_{ui}$ (peso aumentado)

2.3.3 Função Objetivo

O problema de otimização do WMF é:

$$\min_{\mathbf{U}, \mathbf{V}} \mathcal{J}(\mathbf{U}, \mathbf{V}) = \sum_{u=1}^m \sum_{i=1}^n C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i)^2 + \lambda (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2), \quad (16)$$

onde:

- O primeiro termo mede o erro de reconstrução ponderado em todas as entradas;
- C_{ui} pondera cada erro de acordo com a confiança na observação;
- $\lambda \geq 0$ é o coeficiente de regularização L2.

2.4 Otimização do Método WMF por SGD Mini-batch com *Negative Sampling*

Dado que considerar todas as entradas de \mathbf{R} (incluindo os zeros) é computacionalmente proibitivo em matrizes esparsas, utiliza-se a técnica de negative sampling para aproximar o gradiente de forma eficiente.

2.4.1 Estratégia de *Negative Sampling*

Para cada interação positiva $(u, i) \in \Omega$ incluída num mini-batch, amostram-se k itens não observados pelo utilizador u . Seja \mathcal{N}_u^k o conjunto de k itens negativos amostrados para o utilizador u :

$$\mathcal{N}_u^k \subset \{j : R_{uj} = 0\}, \quad |\mathcal{N}_u^k| = k. \quad (17)$$

O mini-batch expandido \mathcal{B}_t^+ contém:

- Pares positivos: $(u, i) \in \mathcal{B}_t$ com $R_{ui} > 0$ e peso $C_{ui} = 1 + \alpha R_{ui}$
- Pares negativos: (u, j) com $j \in \mathcal{N}_u^k$, $R_{uj} = 0$ e peso $C_{uj} = 1$

2.4.2 Função Objetivo do Mini-batch

Para um mini-batch \mathcal{B}_t de interações positivas de tamanho $b = |\mathcal{B}_t|$, a função de custo aproximada é:

$$\begin{aligned} \mathcal{J}_{\mathcal{B}_t}(\mathbf{U}, \mathbf{V}) = & \frac{1}{b(1+k)} \left[\sum_{(u,i) \in \mathcal{B}_t} C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i)^2 \right. \\ & + \sum_{(u,i) \in \mathcal{B}_t} \sum_{j \in \mathcal{N}_u^k} C_{uj} (0 - \mathbf{U}_u^T \mathbf{V}_j)^2 \left. \right] \\ & + \frac{\lambda}{2} (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) \quad . \end{aligned} \quad (18)$$

2.4.3 Cálculo dos Gradientes

Define-se o erro de predição ponderado para cada par:

$$e_{ui} = C_{ui}^{1/2} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i) \quad . \quad (19)$$

2.4.3.1 Derivadas Parciais em relação a \mathbf{U}

Para o fator U_{uf} de um utilizador u presente no mini-batch:

$$\frac{\partial \mathcal{J}_{\mathcal{B}_t}}{\partial U_{uf}} = -\frac{2}{b(1+k)} \left[\sum_{i:(u,i) \in \mathcal{B}_t} C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i) V_{if} + \sum_{j \in \mathcal{N}_u^k} C_{uj} (0 - \mathbf{U}_u^T \mathbf{V}_j) V_{jf} \right] + \lambda U_{uf} \quad . \quad (20)$$

Em forma vetorial:

$$\nabla_{\mathbf{U}_u} \mathcal{J}_{\mathcal{B}_t} = -\frac{2}{b(1+k)} \left[\sum_{i:(u,i) \in \mathcal{B}_t} C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i) \mathbf{V}_i - \sum_{j \in \mathcal{N}_u^k} C_{uj} (\mathbf{U}_u^T \mathbf{V}_j) \mathbf{V}_j \right] + \lambda \mathbf{U}_u \quad . \quad (21)$$

2.4.3.2 Derivadas Parciais em relação a \mathbf{V}

Para o fator V_{if} de um item i presente no mini-batch (positivo ou negativo):

Se i é um item positivo ($i \in \mathcal{B}_t$):

$$\frac{\partial \mathcal{J}_{\mathcal{B}_t}}{\partial V_{if}} = -\frac{2}{b(1+k)} \sum_{u:(u,i) \in \mathcal{B}_t} C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i) U_{uf} + \lambda V_{if} \quad . \quad (22)$$

Se j é um item negativo amostrado ($j \in \mathcal{N}_u^k$ para algum u):

$$\frac{\partial \mathcal{J}_{\mathcal{B}_t}}{\partial V_{jf}} = -\frac{2}{b(1+k)} \sum_{u:j \in \mathcal{N}_u^k} C_{uj} (0 - \mathbf{U}_u^T \mathbf{V}_j) U_{uf} + \lambda V_{jf} \quad . \quad (23)$$

Em forma vetorial unificada:

$$\nabla_{\mathbf{V}_i} \mathcal{J}_{\mathcal{B}_t} = -\frac{2}{b(1+k)} \sum_{u:(u,i) \in \mathcal{B}_t^+} C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i) \mathbf{U}_u + \lambda \mathbf{V}_i \quad . \quad (24)$$

onde \mathcal{B}_t^+ denota o mini-batch expandido com positivos e negativos.

2.4.4 Regras de Atualização

Os fatores são atualizados seguindo a direção oposta ao gradiente. Para cada utilizador u e item i presentes no mini-batch expandido \mathcal{B}_t^+ :

$$\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta \nabla_{\mathbf{U}_u} \mathcal{J}_{\mathcal{B}_t} \quad , \quad (25)$$

$$\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \nabla_{\mathbf{V}_i} \mathcal{J}_{\mathcal{B}_t} \quad , \quad (26)$$

onde $\eta > 0$ é a taxa de aprendizagem. Notemos que ao contrário do NMF, o WMF não impõe restrições de não-negatividade, pelo que não é necessária a projeção $\max(0, \cdot)$.

Algorithm 2 WMF com SGD Mini-batch e *Negative Sampling*

```

1: Entrada:  $\mathbf{R}, \Omega, r, \lambda, \alpha, k, \eta, b, n_{\text{epochs}}$ 
2: Saída:  $\mathbf{U}, \mathbf{V}$ 
3:
4: Inicializar  $\mathbf{U}, \mathbf{V}$  aleatoriamente
5:
6: for  $e = 1$  to  $n_{\text{epochs}}$  do
7:   Embaralhar  $\Omega$  aleatoriamente
8:   Particionar  $\Omega$  em mini-batches  $\mathcal{B}_1, \dots, \mathcal{B}_N$  de tamanho  $b$ 
9:
10:  for  $t = 1$  to  $N$  do
11:    // Construir mini-batch expandido com negative sampling
12:     $\mathcal{B}_t^+ \leftarrow \mathcal{B}_t$ 
13:    for cada  $(u, i) \in \mathcal{B}_t$  do
14:      Amostrar  $k$  itens negativos:  $\mathcal{N}_u^k \subset \{j : R_{uj} = 0\}$ 
15:       $\mathcal{B}_t^+ \leftarrow \mathcal{B}_t^+ \cup \{(u, j) : j \in \mathcal{N}_u^k\}$ 
16:    end for
17:
18:    // Calcular erros ponderados
19:    for cada  $(u, i) \in \mathcal{B}_t^+$  do
20:       $C_{ui} \leftarrow 1 + \alpha R_{ui}$ 
21:       $e_{ui} \leftarrow C_{ui} (R_{ui} - \mathbf{U}_u^T \mathbf{V}_i)$ 
22:    end for
23:
24:    // Atualizar fatores dos utilizadores
25:    for cada  $u$  que aparece em  $\mathcal{B}_t^+$  do
26:       $\nabla_{\mathbf{U}_u} \leftarrow -\frac{2}{b(1+k)} \sum_{i:(u,i) \in \mathcal{B}_t^+} e_{ui} \cdot \mathbf{V}_i + \lambda \mathbf{U}_u$ 
27:       $\mathbf{U}_u \leftarrow \mathbf{U}_u - \eta \nabla_{\mathbf{U}_u}$ 
28:    end for
29:
30:    // Atualizar fatores dos itens
31:    for cada  $i$  que aparece em  $\mathcal{B}_t^+$  do
32:       $\nabla_{\mathbf{V}_i} \leftarrow -\frac{2}{b(1+k)} \sum_{u:(u,i) \in \mathcal{B}_t^+} e_{ui} \cdot \mathbf{U}_u + \lambda \mathbf{V}_i$ 
33:       $\mathbf{V}_i \leftarrow \mathbf{V}_i - \eta \nabla_{\mathbf{V}_i}$ 
34:    end for
35:  end for
36: end for
37:
38: return  $\mathbf{U}, \mathbf{V} = 0$ 

```

3 Metodologia e Implementação

Na presente secção apresenta-se a metodologia adotada para a implementação e avaliação comparativa dos métodos NMF, ver secção 2.2 e WMF, ver secção 2.3, em dados de feedback implícito. Detalham-se os procedimentos de preparação e pré-processamento dos dados, as estratégias de implementação dos algoritmos, os critérios de otimização de hiperparâmetros e as métricas de avaliação empregues. A abordagem metodológica reflete os objetivos centrais deste trabalho: analisar o desempenho preditivo de ambos os métodos, estudar a sensibilidade aos hiperparâmetros específicos do WMF (α e k), e comparar a eficiência computacional entre implementações manuais e bibliotecas otimizadas.

3.1 Dados e Pré-processamento

O presente estudo utilizou o *dataset* MovieLens 25M, uma referência padrão para o desenvolvimento e avaliação de sistemas de recomendação. O dataset é composto por 25 milhões de avaliações explícitas (notas de 1 a 5 estrelas), abrangendo aproximadamente 160 mil utilizadores e 60 mil filmes, porém como o projeto foca-se no contexto de recomendação implícita (*Implicit Feedback*), onde o objetivo não é prever a nota exata (1 a 5), mas sim a probabilidade de interação (consumo/interesse), foram tomadas as seguintes decisões:

- Binarização do universo de dados: Para refletir este contexto, as avaliações originais foram binarizadas. Qualquer avaliação existente foi tratada como um sinal de interesse (1), e a ausência de avaliação foi tratada como um sinal de não observação/ausência de interesse (0).
- Uso da Avaliação Explícita: O valor original da avaliação (1-5 estrelas) foi reservado exclusivamente para a comparação qualitativa final, onde é utilizado para validar se as recomendações do modelo se alinham com os itens mais gostados pelo utilizador.

Ainda assim, dado que o trabalho foi efetuado com recursos computacionais bastante limitados (portátil genérico com 8Gb de RAM) decidi filtrar os filmes e os utilizadores, ou seja:

1. Nos filmes foram apenas considerados filmes que receberam mais de 1000 visualizações/avaliações.
2. Nos *users* foram apenas considerados utilizadores que visualizaram pelo menos 50 filmes.

Esta decisão permitiu reduzir o número de users de em cerca de 40% e os filmes em cerca de 94% mantendo o número de interações em cerca de 80% do volume original, de modo a garantir a integridade dos dados mas também deixar a matriz \mathbf{R} mais gerenciável do ponto de vista computacional, especialmente no âmbito de implementações manuais.

A divisão dos dados foi feita pelas interações, isto para garantir que fosse possível usar todos os utilizadores e desse modo construir um perfil baseado no seu histórico, considerando aproximadamente:

- Treino \rightarrow 70%, que corresponde a 14 milhões de interações;
- Validação \rightarrow 15%, que corresponde a 3 milhões de interações;
- Teste \rightarrow 15%.

Esta metodologia garante que as matrizes de fatores latentes (\mathbf{U} e \mathbf{V}) obtidas são robustas, pois são treinadas com a rede completa de utilizadores.

3.2 Estratégia de Implementação: Manual vs. Otimizada

Para cumprir os objetivos pedagógicos e realizar a análise de eficiência, foram desenvolvidas duas abordagens para ambos os métodos:

3.2.1 Abordagem Manual

Implementação “do zero” onde o cálculo dos gradientes e a lógica de SGD são explícitos, e existe um maior controle sobre o código.

- SGD com Mini-batches: Vetorização das operações dentro do *batch* para mitigar a lentidão do Python.
- Negative Sampling Dinâmico (para WMF): Geração de k negativos em tempo de execução para cada batch positivo.

3.2.2 Abordagem Otimizada

Utilização de bibliotecas compiladas (e.g., *implicit* ou *scikit-learn*) para estabelecer um termo de comparação em termos de tempo computacional e métricas finais, procurando destacar o seu impacto e superioridade no que toca a tratamento de dados de “grande” escala.

3.3 Critérios de Seleção e Desempenho

Em vez de optar por uma pesquisa exaustiva de hiperparâmetros, definiu-se uma grelha de pesquisa estratégica que combina valores extremos (baixo e alto) para cada hiperparâmetro, permitindo explorar eficientemente o espaço de soluções.

Os critérios de avaliação e seleção de modelos consideram dois aspetos fundamentais:

1. Qualidade preditiva: Medida através da métrica Precision@K¹, que avalia a proporção de itens relevantes entre os K itens mais recomendados. Para um utilizador u e conjunto de validação, define-se:

$$\text{Precision@K}_u = \frac{|\{\text{itens relevantes}\} \cap \{\text{top-K recomendados}_u\}|}{K} \quad (27)$$

onde os itens relevantes correspondem às interações observadas no conjunto de validação, e os top-K recomendados são os K itens com maior valor predito $\hat{r}_{ui} = \mathbf{U}_u^T \mathbf{V}_i$. A métrica global é obtida pela média sobre todos os utilizadores:

$$\text{Precision@K} = \frac{1}{|U|} \sum_{u \in U} \text{Precision@K}_u \quad (28)$$

Esta métrica reflete diretamente a utilidade prática do sistema, pois apenas as primeiras K posições do ranking são tipicamente apresentadas ao utilizador. Valores mais elevados indicam maior capacidade do modelo em identificar itens de interesse nas posições superiores do ranking.

2. Eficiência computacional: Quantificada pelo tempo de treino do modelo. Tempos reduzidos são valorizados, particularmente na comparação entre implementações manuais e bibliotecas otimizadas.

O processo de seleção de hiperparâmetros é realizado em duas etapas sequenciais:

¹Para este trabalho fixou-se K=10 em todas as avaliações, correspondendo ao número típico de recomendações apresentadas em interfaces de utilizador de sistemas comerciais (e.g., Netflix, Spotify).

1. Pesquisa na grelha inicial: Treina-se cada configuração de hiperparâmetros definida pela grelha estratégica (valores extremos) utilizando o conjunto de treino, e avalia-se o desempenho no conjunto de validação através das métricas Precision@K e tempo de treino. Identificam-se as configurações candidatas que apresentam melhor compromisso qualidade-eficiência.
2. Refinamento local: Para a configuração candidata identificada na Fase 1, explora-se a vizinhança dos hiperparâmetros, testando valores intermédios adjacentes. Esta pesquisa localizada permite descobrir configurações ótimas sem incorrer no custo de uma pesquisa exaustiva global. A configuração que apresentar maior Precision@K nesta fase é selecionada como modelo final.

3.3.1 Avaliação Final

Após a seleção dos hiperparâmetros ótimos através do procedimento acima descrito, o modelo é retreinado utilizando a união dos conjuntos de treino e validação. Este modelo final é então avaliado no conjunto de teste (nunca visto durante a otimização), reportando-se a Precision@K de teste como estimativa imparcial do desempenho esperado em produção.

4 Resultados e Estudo Experimental

Na presente secção apresentam-se os resultados experimentais da aplicação dos métodos NMF e WMF em dados de feedback implícito. A análise aborda três componentes principais: a comparação de eficiência computacional entre implementações manuais e bibliotecas otimizadas, o estudo de sensibilidade aos hiperparâmetros do WMF (α e k), e a avaliação final no conjunto de teste. Os resultados evidenciam os trade-offs entre eficiência e qualidade preditiva, demonstrando os ganhos das implementações otimizadas e as relações críticas entre hiperparâmetros e desempenho dos modelos.

4.1 Análise de Eficiência Computacional

A Tabela 1 apresenta a comparação de desempenho entre implementações manuais (NumPy) e bibliotecas otimizadas (Scikit-Learn e Implicit) para os métodos NMF e WMF.

Modelo	Implementação	Tempo/Época (s)	Tempo Total (min)	Precision@10
NMF	Manual (SGD)	27.9	23.2 (20 épocas)	2.8%
NMF	Scikit-Learn (CD)	0.5	0.2 (20 épocas)	24.3%
WMF ($k = 5$)	Manual (SGD)	91.7	12.0 (5 épocas)	3.8%
WMF	Implicit (ALS)	1.9	0.1 (4 épocas)	25.4%

Tabela 1: Comparação de desempenho entre implementações manuais e otimizadas para as melhores configurações de hiperparâmetros identificadas.

As bibliotecas otimizadas demonstram ganhos significativos de eficiência em ambos os métodos. Para o NMF, o Scikit-Learn processa cada época em 0.5s versus 27.9s da implementação manual, um fator de aceleração de aproximadamente $56\times$. No WMF, a biblioteca Implicit executa cada época em 1.9s comparado com 91.7s da implementação manual (fator $48\times$). Estas diferenças resultam de operações matriciais implementadas em C/Cython, alocações otimizadas de memória, estruturas de dados especializadas para matrizes esparsas, entre outros.

A discrepância na qualidade preditiva é ainda mais pronunciada: as implementações otimizadas alcançam 24.3% (NMF) e 25.4% (WMF) de Precision@10, enquanto as manuais obtêm

apenas 2.8% e 3.8%, respectivamente. Esta diferença não reflete incorreções algorítmicas, mas sim severas limitações na exploração de hiperparâmetros. A pesquisa inicial com 16 configurações consumiu aproximadamente 4.5 horas para o NMF manual, inviabilizando a fase de refinamento (estimada em $>22h$). As bibliotecas otimizadas permitiram uma exploração abrangente do espaço de hiperparâmetros, resultando em configurações substancialmente superiores.

Comparando os dois métodos, o WMF apresenta maior custo computacional que o NMF nas implementações manuais (91.7s vs. 27.9s por época, fator $3.3\times$), devido ao *negative sampling* e ao cálculo dos pesos C_{ui} . Nas bibliotecas otimizadas, esta diferença reduz-se para $3.8\times$ (1.9s vs. 0.5s), demonstrando que otimizações de baixo nível mitigam parcialmente a complexidade adicional do WMF. Ambos os métodos apresentam desempenho similar quando adequadamente otimizados (24.3% vs. 25.4%), com ligeira vantagem para o WMF na validação.

4.2 Sensibilidade aos Hiperparâmetros (WMF)

Estudou-se o impacto de k e α na métrica de precisão (Precision@10).

1. Variação de $k \in \{1, 5, 10, 20\}$: Aumentar k melhora a robustez do modelo, mas aumenta linearmente o tempo de treino.

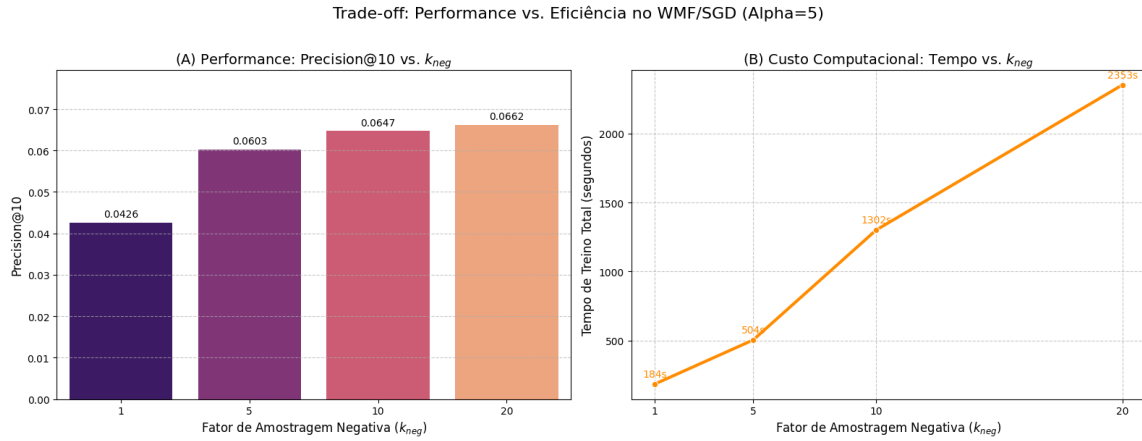


Figura 1: Trade-off: Performance vs. Eficiência no WMF/SGD ($\alpha=5$, k variável)

2. Variação de $\alpha \in \{5, 10, 20, 40\}$: Valores elevados de α focam o modelo nas interações observadas.

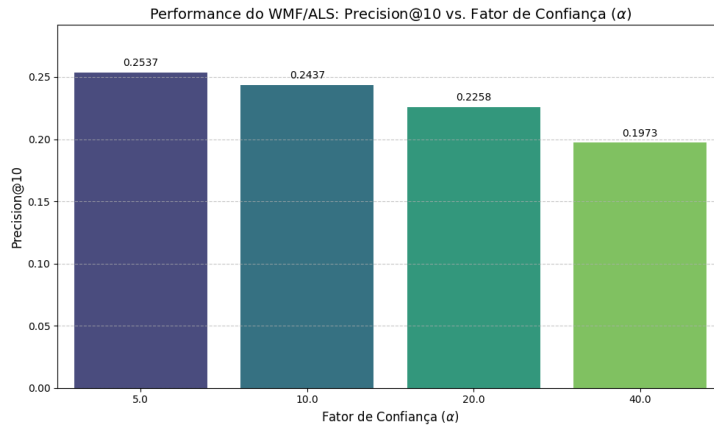


Figura 2: Performance do WMF/ALS: Precision@10 vs. Fator de Confiança (α)

Analisando as figuras 1 e 2, observamos que a variação dos hiperparâmetros no WMF demonstra trade-offs assimétricos, sendo crucial não quantificar os valores de Precision@K devido à disparidade algorítmica (SGD vs. ALS). A análise relativa mostra que o α (Fator de Confiança), otimizado no solver ALS, revelou uma relação inversa e crítica com a qualidade preditiva, onde o Precision@K melhorou consistentemente à medida que α diminuía, sugerindo que o ótimo para este modelo se encontra com a ponderação de confiança mais baixa (ou mais uniforme). Em contraste, a variação de k_{neg} (Amostragem Negativa) no SGD manual resultou em ganhos marginais de performance para o Precision@K, enquanto o custo computacional associado ao aumento de k se tornou excessivo e insustentável (cujo tempo de treino disparou de forma linear).

4.3 Resultados no Conjunto de Teste

Após a seleção dos hiperparâmetros ótimos, os modelos foram retreinados utilizando a união dos conjuntos de treino e validação (aproximadamente 17 milhões de interações), maximizando a informação disponível antes da avaliação final no conjunto de teste. A Tabela 2 apresenta os resultados finais.

Modelo	Implementação	Tempo/Época (s)	Tempo Total (min)	Precision@10
NMF	Manual (SGD)	38.2	31.2 (20 épocas)	3.5%
NMF	Scikit-Learn (CD)	0.5	0.2 (10 épocas)	32.9%
WMF ($k = 10$)	Manual (SGD)	216.9	18.4 (5 épocas)	29.5%
WMF	Implicit (ALS)	1.6	0.1 (4 épocas)	34.6%

Tabela 2: Desempenho final no conjunto de teste após retreino com treino mais validação.

A qualidade preditiva no conjunto de teste revela ganhos expressivos para as implementações otimizadas: o NMF Scikit-Learn alcança 32.9% de Precision@10 (um aumento de 8.6 pontos percentuais relativamente aos 24.3% da validação), enquanto o WMF Implicit atinge 34.6% (ganho de 9.2 p.p. face aos 25.4% da validação). Estes ganhos refletem o impacto positivo do retreino com dados adicionais e da otimização bem-sucedida dos hiperparâmetros. As bibliotecas otimizadas mantêm eficiência superior, processando épocas em 0.5s (NMF) e 1.6s (WMF) versus 38.2s e 216.9s das implementações manuais, respetivamente.

Impacto da otimização de hiperparâmetros: A comparação entre as Tabelas 1 e 2 evidencia o papel decisivo da análise de sensibilidade dos hiperparâmetros α e k no WMF manual. O ajuste de $k = 5$ para $k = 10$ resultou num ganho extraordinário de 25.7 pontos percentuais (3.8% \rightarrow 29.5%), aproximando-se significativamente da implementação otimizada (34.6%). Este resultado demonstra que, mesmo com limitações computacionais, a otimização cuidadosa de hiperparâmetros permite recuperar grande parte da capacidade preditiva do modelo. Em contraste, o NMF manual permaneceu subótimo (3.5%), pois a inviabilização computacional da fase de refinamento impediu a exploração necessária do espaço de hiperparâmetros, resultando num diferencial de 29.4 pontos para a versão Scikit-Learn.

Em termos práticos, três dos quatro modelos (NMF otimizado, WMF manual otimizado e WMF otimizado) conseguiram captar adequadamente as características dos perfis dos utilizadores, apresentando Precision@10 entre 29.5% e 34.6%. Isto significa que, em média, 3 em cada 10 recomendações são coerentes com o perfil do utilizador, correspondendo a itens efetivamente visualizados no conjunto de teste.

5 Interpretação dos Fatores e Avaliação Qualitativa das Previsões

Além da análise quantitativa baseada em métricas agregadas, procedeu-se a uma avaliação qualitativa das recomendações geradas pelo modelo WMF Implicit (ALS) no conjunto de teste. Esta

análise permite compreender, na prática, como o sistema de recomendação captura as preferências individuais dos utilizadores e a coerência das sugestões face ao seu histórico de visualizações.

5.1 Análise Qualitativa das Previsões

A Tabela 3 apresenta três casos representativos de utilizadores com diferentes níveis de acerto no Top-10, ilustrando a capacidade do modelo em identificar padrões de interesse e propor recomendações alinhadas com os géneros cinematográficos preferidos.

Utilizador	Hit@10	Histórico no Teste (Géneros)	Top-5 Recomendações do Modelo
User 0	4/10	Adventure, Western, Comedy, Fantasy, Action, War, Thriller.	1. <i>Lord of the Rings: Two Towers</i> ✓ 2. <i>Matrix, The</i> ✓ 3. <i>Men in Black</i> ✓ 4. <i>Titanic</i> × 5. <i>Terminator, The</i> ×
User 1	9/10	Animation, Action, Sci-Fi, Comedy, Crime, Drama, War, Adventure.	1. <i>I, Robot</i> ✓ 2. <i>Dark Knight, The</i> ✓ 3. <i>Avatar</i> ✓ 4. <i>Pirates: Black Pearl</i> ✓ 5. <i>LOTR: Two Towers</i> ✓
User 2	8/10	Adventure, Fantasy, Action, Crime, Animation, Sci-Fi, Children.	1. <i>Star Wars: Episode V</i> ✓ 2. <i>Interstellar</i> ✓ 3. <i>Star Wars: Episode IV</i> ✓ 4. <i>LOTR: Return of the King</i> ✓ 5. <i>Iron Man</i> ×
User 3	2/10	Animation, Action, Drama, Comedy, Sci-Fi, Mystery, Western.	1. <i>Jurassic Park</i> × 2. <i>Toy Story</i> ✓ 3. <i>Speed</i> × 4. <i>Terminator 2</i> × 5. <i>Mission: Impossible</i> ×
User 4	5/10	Comedy, Drama, Adventure, Sci-Fi, Action, Mystery, Horror.	1. <i>Schindler's List</i> × 2. <i>Apollo 13</i> ✓ 3. <i>Twister</i> × 4. <i>Silence of the Lambs</i> ✓ 5. <i>Terminator 2</i> ✓

Tabela 3: Análise qualitativa das recomendações (Top-5) vs. Interações reais no Teste.

5.2 Observações Qualitativas:

- User 0: O modelo capturou corretamente a preferência por filmes de aventura e fantasia (*Lord of the Rings*), bem como ação e ficção científica (*Matrix*, *Men in Black*). As sugestões não acertadas, como *Titanic* e *E.T.*, embora não presentes no teste, mantêm coerência temática com drama e elementos fantásticos.
- User 3: Com apenas 2 acertos, este caso ilustra as limitações do modelo. Apesar de acertar *Toy Story*, as recomendações de *Jurassic Park* e *Speed* não correspondem exatamente ao teste, embora partilhem géneros como ação e thriller presentes no histórico do utilizador.
- User 4: O modelo demonstrou boa capacidade ao recomendar *Apollo 13*, *Terminator 2* e *Speed* (3 dos 4 acertos), filmes que o utilizador efetivamente visualizou. As sugestões não acertadas, como *Schindler's List* e *Godfather*, embora não presentes no teste, são filmes de elevada qualidade que podem representar descobertas valiosas.

Esta análise revela que, mesmo quando o modelo não acerta exatamente nos filmes do conjunto de teste, as recomendações tendem a manter coerência com os gêneros e estilos cinematográficos do perfil do utilizador, cumprindo o objetivo de sistemas de recomendação implícita: sugerir conteúdos relevantes que aumentem a probabilidade de interação futura.

A análise qualitativa dos vetores latentes V permitiu identificar agrupamentos semânticos. A Figura 3 apresenta uma projeção t-SNE dos itens.

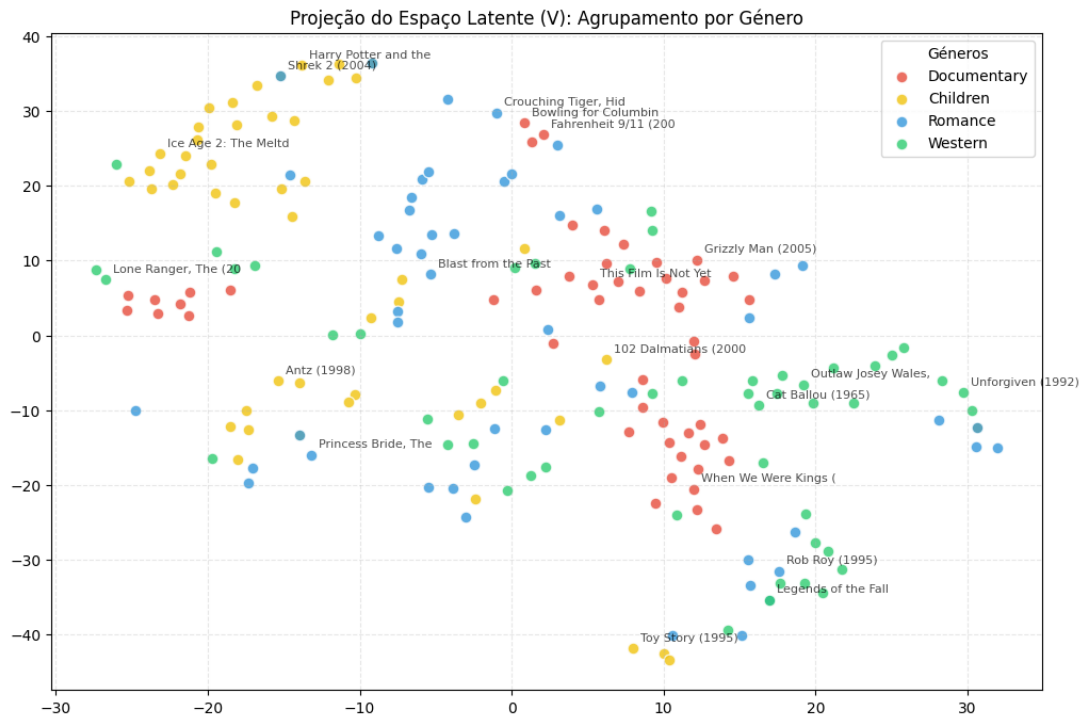


Figura 3: Projeção t-SNE dos fatores latentes dos itens (Matriz V). As cores representam gêneros distintos, evidenciando a formação de clusters semânticos coerentes aprendidos pelo modelo WMF.

Esta projeção t-SNE confirma que o modelo WMF traduziu eficazmente as interações implícitas em relações geométricas coerentes. A organização dos pontos demonstra que o espaço latente de $r = 20$ capturou a semântica dos filmes sem auxílio de metadados durante o treino. A figura 3 permite capturar os seguintes pontos:

- Segregação de Nicho: O cluster de Documentários (Vermelho) está claramente definido e centralizado, indicando um padrão de consumo único e distinto da ficção.
- Proximidade de Franquias: Filmes como *Shrek 2* e *Harry Potter* (Amarelo/Azul no topo) partilham vizinhança, validando a capacidade do modelo em agrupar cinema familiar e de fantasia.
- Dispersão de Gêneros Híbridos: O Romance (Azul) aparece mais espalhado, refletindo a sua natureza transversal enquanto gênero secundário em animações ou dramas.
- Eixo de Género Western (Verde): Filmes como *Unforgiven* e *Outlaw Josey Wales* surgem na periferia direita, evidenciando uma “assinatura” latente para gêneros de nicho clássico.

A estrutura visível comprova que o ajuste de um α baixo foi bem-sucedido: o modelo focou-se na estrutura global de gostos em vez de memorizar ruído, permitindo que a afinidade de perfil entre filmes semelhantes fosse maximizada.

6 Conclusões e Observações Finais

O presente trabalho explorou a aplicação de técnicas de fatorização de matrizes para sistemas de recomendação com feedback implícito, utilizando o conjunto de dados *MovieLens 25M*. A análise centrou-se na comparação entre o *Non-negative Matrix Factorization* (NMF) e o *Weighted Matrix Factorization* (WMF), avaliando o impacto da otimização de hiperparâmetros e da eficiência computacional em ambientes de hardware restrito. A transição de uma implementação manual em NumPy para bibliotecas otimizadas revelou-se fundamental, uma vez que estas últimas permitiram processar milhões de interações em segundos, demonstrando que a escalabilidade nestes sistemas depende criticamente da exploração da esparsidade das matrizes através de algoritmos como o *Alternating Least Squares* (ALS) ou o *Coordinate Descent* (CD).

A comparação crítica entre os métodos evidenciou a superioridade do WMF perante dados implícitos. Enquanto o NMF tradicional tenta reconstruir a matriz original focando-se apenas nas interações observadas, o WMF introduz o conceito de confiança (α), tratando a ausência de interação como uma indicação de baixo interesse, mas não necessariamente como um feedback negativo. Esta abordagem permitiu ao modelo capturar relações mais profundas e menos enviesadas pela popularidade dos itens. A análise qualitativa e a projeção t-SNE confirmaram esta eficácia, mostrando que o modelo organizou o espaço latente em clusters semânticos coerentes, onde géneros de nicho como documentários ou filmes de animação ficaram claramente segregados no mapa de afinidades, mesmo sem acesso prévio aos metadados.

A avaliação exaustiva no conjunto de teste revelou que a precisão do sistema é altamente dependente de um equilíbrio rigoroso entre o peso da regularização e a intensidade da amostragem negativa. Observou-se que modelos com erros de treino mais baixos nem sempre resultaram nas melhores recomendações para o utilizador final, reforçando que, em sistemas de recomendação, a ordenação relativa dos itens (*Ranking*) é frequentemente mais valiosa do que a precisão exata do valor previsto. As “sugestões” geradas pelo modelo, embora contabilizadas como erros nas métricas rígidas de Precision@K, demonstraram uma elevada coerência temática com o histórico real dos utilizadores, validando a capacidade do sistema em generalizar preferências.

Em síntese, o trabalho sublinha que a escolha do algoritmo deve ser pautada pela natureza do feedback disponível. O WMF consolidou-se como a solução mais robusta para lidar com a esparsidade extrema e o silêncio dos dados implícitos, oferecendo um compromisso ideal entre interpretação latente e precisão prática. Os resultados obtidos demonstram que a fatorização de matrizes continua a ser uma ferramenta poderosa para mapear o universo de preferências dos utilizadores, transformando interações esparsas numa arquitetura geométrica de gostos e afinidades.

Referências

- [1] Aggarwal, C. C. (2020). *Linear Algebra and Optimization for Machine Learning*. Springer.
- [2] Hu, Y., Koren, Y., & Volinsky, C. (2008). *Collaborative filtering for implicit feedback datasets*. In Proceedings of ICDM.
- [3] Silva, L. *Slides da Unidade Curricular de Métodos Matemáticos em Inteligência Artificial*. ISEL.