UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÉNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Previsão de notas esparsas na filtragem colaborativa a partir de relações de vizinhança com redes neurais artificiais

Adriano Cabral Linhares Mourthé

Orientador

Carlos Eduardo Ribeiro de Mello

RIO DE JANEIRO, RJ - BRASIL
Fevereiro de 2019

Previsão de notas esparsas na filtragem colaborativa a partir de relações de vizinhança com redes neurais artificiais

Adriano Cabral Linhares Mourthé

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Prof. Dr. Carlos Eduardo Ribeiro de Mello - UNIRIO

Prof. Dr. Márcio de Oliveira Barros - UNIRIO

Prof. Dr. Eduardo Fonseca Mendes - FGV/RJ

Prof. Dr. Filipe Braida do Carmo - UFRRJ

RIO DE JANEIRO, RJ - BRASIL
Fevereiro de 2019

Catalogação informatizada pelo(a) autor(a)

C Cabral Linhares Mourth[e, Adriano
Previsão de notas esparsas na filtragem colaborativa a partir de relações de vizinhança com redes neurais artificiais / Adriano Cabral Linhares Mourth[e. -- Rio de Janeiro, 2019.
48

Orientador: Carlos Eduardo Ribeiro de Mello.
Dissertação (Mestrado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2019.

1. Aprendizado de máquina. 2. Filtragem colaborativa. 3. Redes neurais artificias. I. Ribeiro de Mello, Carlos Eduardo, orient. II. Título.

Agradecimentos

Agradeço ao meu orientador Carlos eduardo, por toda paciência e atenção concedidas. Agradeço também a todos os professores e funcionários do CCET pela formação acadêmica.

Agradeço aos meus pais por todo auxílio e suporte, sem vocês este trabalho não seria possível.

Agradeço aos meus amigos, que me incentivaram ao longo do mestrado.

Mourthé, Adriano Cabral Linhares. Previsão de notas esparsas na filtragem colaborativa a partir de relações de vizinhança com redes neurais artificiais. UNIRIO, 2018. 48 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Este trabalho apresenta uma modelagem para o problema de filtragem colaborativa. Nossa proposta modela as notas de um usuário segundo a relação do usuário com sua vizinhança. Para aprender tal relação, desenvolvemos uma rede neural artificial com uma função de custo especifica capaz de lidar com dados esparsos. Os resultados experimentais obtidos a partir de três conjuntos de dados do domínio de streamming de conteúdo indicam que a proposta é extremamente promissora. A modelagem proposta obteve desempenho competitivo ou superior às principais alternativas disponíveis no Estado-da-Arte.

Palavras-chave: Aprendizado de máquina, Filtragem colaborativa, redes neurais artificiais.

ABSTRACT

This work presents a new modeling for the collaborative filtering problem. Our proposal models a item's ratings according to the item's relationship with their neighborhood. To learn such a relationship, we developed an artificial neural network with a specific cost function capable of dealing with sparse data. The experimental results indicate that the proposal is extremely promising.

Keywords: Machine learning, Colaborative filtering, artificial neural network.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Nomenclaturas

RNA	Rede neural artificial.
GDE	Gradiente descendente estocástico.
SR	Sistema de recomendação.
MAE	Mean absolute error.
RMSE	Root mean square error.
COS	Similaridade cosseno.
CCP	Coeficiente de correlação de pearson.
S_{COS}	Matriz de similaridade cosseno.
S_{L_2}	Matriz de similaridade construída com a distância.
RNA Tanh	Rede neural artificial com a função de ativação Tangente hiperbólica.
RNA ReLU	Rede neural artificial com a função ativação ReLU
RNA ELU	Rede neural artificial com a função ativação ELU
RNA SELU	Rede neural artificial com a função ativação SELU
RNA SWISH	Rede neural artificial com a função ativação SWISH
SVR	Regressão por Vetores de Suporte

1. Introdução

Neste capítulo, apresenta-se as motivações, objetivos e a organização do trabalho.

1.1 Motivação

Os sistemas de recomendação se tornaram símbolo das plataformas de comércio eletrônico e de distribuição de mídias digitais. No caso de uma das maiores plataformas de Streamming de filmes, o Netflix, 80% do conteúdo assistido é proveniente de recomendações [1]. Estes sistemas são como ferramentas de suporte a vendas [2], que ajudam os consumidores a identificar seu produto ideal entre a grande variedade de opções disponíveis online. Portanto, é de grande interesse para as plataformas on-line e seus usuários o desenvolvimento de ferramentas que ofereçam boas recomendações e uma melhor experiência de consumo.

Tendo em vista tamanha importância, este trabalho propõe uma nova modelagem capaz de melhorar a qualidade dos sistemas de recomendação.

1.2 Questões de investigações

Neste trabalho é investigado o problema da filtragem colaborativa, que consiste em prever as notas dos pares de usuários e itens que estão faltando na matriz de utilidade [3].

Nossa hipótese é que as notas de um usuário podem ser explicadas pela relação do usuário com sua vizinhança. Tal relação poderia ser mapeada por um modelo supervisionado de aprendizado de máquina

No caso do trabalho, o modelo supervisionado adotado é uma rede neural artifi-

cial capaz de lidar com a esparsidade intrínseca da matriz de utilidade. Para isso, desenvolvemos uma nova função de custo que minimiza a norma de frobenius e não permite que o backpropagation corrija os pesos associados às notas ausentes.

1.2.1 Objetivos

Este trabalho tem como objetivo desenvolver um modelo de filtragem colaborativa capaz prever notas de itens com base na informação espacial de sua vizinhança de acordo com suas preferências.

Portanto, dois grandes objetivos foram almejados:

- Propor uma nova modelagem de filtragem colaborativa baseada em modelo: Uma rede neural feedfoward, com sua função de custo adaptada para o problema de filtragem colaborativa.
- Comparar a modelagem com os modelos do estado- da-arte da literatura: avaliar se a modelagem proposta é eficaz a comparando com outras alternativas do estado-da-arte, considerando os aspectos de previsão de notas.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Validar a função de custo proposta com dados esparsos;
- Avaliar o impacto de diversas funções de similaridade no modelo proposto;
- Testar e avaliar diferentes arquiteturas de redes neurais no modelo proposto; e, por fim,
- Analisar se há algum padrão interpretável na relação entre os pesos da rede neural e a entrada.

1.2.3 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- Capítulo 1: apresenta os objetivos;

- Capítulo 2: apresenta os principais conceitos da literatura de sistemas de recomendação;
- Capítulo 3: apresenta a modelagem proposta neste trabalho;
- Capítulo 4: descrição dos experimentos e resultados;
- Capítulo 5: conclusão e trabalhos futuros

2. Fundamentação teórica

Este capítulo apresenta os conceitos fundamentais para a compreensão de sistemas de recomendação baseados em filtragem colaborativa. Na primeira seção deste capítulo, são definidos os principais conceitos técnicos associados à área de sistemas de recomendação. Na segunda seção, é realizada uma descrição mais detalhada do problema da filtragem colaborativa. Por fim, a última seção, apresenta uma revisão dos principais métodos de filtragem colaborativa e dos trabalhos relacionados com os objetivos deste texto.

2.1 Sistemas de recomendação

Os sistemas de recomendação (SR) são métodos que fornecem sugestões de itens relevantes para usuários de um sistema [3]. No contexto de SR, o termo "item"é usado para representar, de forma genérica, o que um SR recomenda a um usuário. Por exemplo, um item de uma plataforma de streamming de filmes significa um filme.

SRs têm como problema central gerar recomendações de itens que sejam relevantes para os usuários com base nos seus históricos de preferências. Adomavicius et al. [4] definiram formalmente este problema como: dado um conjunto de usuários $\mathcal{U}$, um conjunto com todos os itens do sistema $\mathcal{I}$ e uma função de utilidade $\mathcal{G}$ que mensura a utilidade do item i para o usuário u, i.e, $\mathcal{G} : \mathcal{U} \times \mathcal{I} \mapsto \mathcal{R}$, onde $\mathcal{R}$ é conjunto ordenado que contém as preferências do usuário pelos itens. Então, para cada $u \in \mathcal{U}$ é preciso sugerir o item $i^* \in \mathcal{I}$ que maximize a função $\mathcal{G}$, como exposto a seguir:

$$\forall u \in \mathcal{U}, i^* = \underset{i^* \in \mathcal{I}}{\operatorname{argmax}}\, \mathcal{G}(u, i). \tag{2.1}$$

Portanto, para a construção de um SR são necessários os dados das preferências de usuários pelos os itens que, geralmente, são obtidos pelas suas avaliações recebidas. Dessa forma, é possível construir uma matriz de utilidade [5], onde cada entrada representa a opinião do usuário $\mathfrak{u}$ sobre cada um dos item da base avaliados. A tabela 2.1 a seguir é um exemplo de matriz de utilidade, com quatro filmes e três usuários (que avaliaram os filmes com notas entre 1 e 5).

	Stalker	Solaris	Metropolis	Alphaville
Usuário 1	5	Ø	Ø	Ø
Usuário 2	Ø	2	Ø	1
Usuário 3	Ø	Ø	3	Ø

Tabela 2.1: As entradas com Ø representam a ausência de avaliação

No entanto, como pode ser observado na tabela 2.1 mostra, a função utilitária $\mathcal{G}$ é, geralmente, definida em um subconjunto de $\mathcal{U} \times \mathcal{I}$. Portanto, o SR precisa ser capaz de extrapolar $\mathcal{G}$ para todo espaço $\mathcal{U}\times\mathcal{I}$. De acordo com Adomavicius et al. [4], o preenchimento da matriz de utilidade pode ser feito por meio de heurísticas ou pela utilização de uma função que minimiza uma função de custo, e.g., Root mean squared Error. A questão da extrapolação pode ser tratada de diversas formas e pode ser dividida em seis classes, segundo taxonomia criada por Burke [6]:

- Filtragem colaborativa: os sistemas baseados em filtragem colaborativa utilizam exclusivamente as avaliações dos usuários para fazer recomendações baseadas no gosto de usuários semelhantes;
- Baseada em conteúdo: os sistemas baseados em conteúdo usam características dos produtos para realizar recomendações, como, por exemplo, a descrição textual;
- Baseado em conhecimento: os sistemas baseados em conhecimento realizam recomendações com base no conhecimento histórico do domínio em que o SR está inserido;
- Demográfico: essa classe de sistema recomenda itens com base no perfil demográfico do usuário;
- Baseado em comunidade: essa classe de SR recomenda itens baseado nos relacionamento dos usuários, como, por exemplo, os itens comprados pelos amigos do usuário;e,
- Híbridos: é a classe composta por SR que fazem recomendações por meio da combinação das abordagens supracitadas.

Os SR também podem ser vistos como um filtro de informação, que evita o sobrecarga do usuário com o excesso de itens, fenômeno conhecido como Choice overload [7].

2.1.1 Filtragem colaborativa

Os usuários recorrem a opiniões de outras pessoas para tomarem decisões. Os SR baseados em filtragem colaborativa (FC) se apoiam neste comportamento social para recomendarem itens aos usuários. De acordo com Adomavicius et al. [4], a modelagem do problema de recomendação com uma abordagem baseada em FC pode ser formalizada como: a utilidade do item i para o usuário u, $\mathcal{G}(i, u)$ é estimada com base na utilidade $\mathcal{G}(U_{u'}, i)$, onde $u' \in \mathcal{U}$ e são os usuários mais similares aos usuários u. Ou seja, a recomendação do SR para um usuário deve ser baseada em usuários com perfis de preferências semelhantes. A principal vantagem introduzida por essa modelagem é que, enquanto outras abordagens, e.g., baseada em conteúdo, necessitam de informações extras sobre o produto, a FC precisa somente das avaliações dos usuários.

Entretanto, a dependência dessas avaliações gera uma severa limitação. Em casos reais, geralmente, os usuários avaliam menos de 1% do total de itens do sistema [8]. Isso faz com que a matriz de utilidade seja intrinsecamente esparsa. Em função disto, um modelo para a extrapolação do espaço de notas se torna um desafio. Outra limitação, que é comumente associada à esparsidade da matriz de utilidade é conhecida como cold start. Este fenômeno ocorre quando não é possível fazer recomendações confiáveis pela falta de avaliações de um item [9]. Por exemplo, um item nunca poderá ser recomendado por um método de FC se ele nunca tiver recebido antes uma avaliação de um usuário.

Entre os métodos de FC existem duas classes [3], os baseados em vizinhança (também conhecidos como baseados em memória ou baseados em heurísticas) e os baseados em modelo. Nas seções 2.1.2 e 2.1.3, cada uma destas abordagens é descrita.

2.1.2 Filtragem colaborativa baseada em memória

Métodos baseados em memória são heurísticas que prevêem as notas de um usuário u com base nas avaliações de seus vizinhos mais próximos [3]. Uma formulação mais formal é dada por Adomavicius et al. [4]: O valor da nota $r_{u,i}$ é calculada como

a agregação das notas do(s) usuário(s) que mais se assemelham ao usuário u. Dessa forma, a nota é calculada como:

$$r_{u,c} = \frac{1}{||\mathcal{N}_i(u)||} \sum_{v \in \mathcal{N}_i(u)} r_{vi}, \tag{2.2}$$

onde $\mathcal{N}_i(u)$ é o conjunto dos vizinhos mais próximos.

Uma limitação da equação 2.2 é que ela não considera que os vizinhos possam ter graus de similaridade distintos. Dois usuários podem ter avaliado os mesmos filmes e atribuírem notas distintas para eles. Uma solução para esse problema é ponderar as notas de cada usuário pela sua similaridade com u [3].

Além da maneira anteriormente mencionada, os algoritmos baseados em memória também podem ser baseados em itens. Isto é, o sistema utiliza os itens mais semelhantes ao item i para prever a preferência do usuário u para esse item. Essa variante é conhecida como baseada em itens, enquanto a apresentada anteriormente é baseada em usuários.

Outra questão importante é a medida empregada para mensurar a similaridade entre usuários ou itens. A escolha da medida de similaridade pode gerar impacto na seleção dos vizinhos mais próximos e, consequentemente, afetar o resultado das recomendações [3]. Na literatura especializada, as duas medidas mais empregadas são o coeficiente de correlação de pearson (CPP) e a similaridade cosseno (COS). Esta última considera somente o ângulo entre dois os vetores de notas de usuários como o grau de similaridade, dado por:

$$COS(u, v) = \frac{\sum_{i \in \mathcal{J}_{u,v}}^{n} r_{ui} r_{vi}}{\sqrt{\sum_{i \in \mathcal{J}_u}^{n} r_{ui}^2} \sqrt{\sum_{i \in \mathcal{J}_v}^{n} r_{vj}^2}}, \tag{2.3}$$

onde $\mathcal{J}_{u,v}$ são os itens co-avaliados por u e v. Já o coeficiente de correlação de Pearson é semelhante ao cosseno, mas remove os efeitos da média e variância de cada usuário, conforme a equação a seguir:

$$CCP(u, v) = \frac{\sum_{i \in \mathcal{J}_{u,v}}^{n} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in \mathcal{J}_u}^{n} (r_{ui} - \overline{r}_u)^2} \sqrt{\sum_{i \in \mathcal{J}_v}^{n} (r_{vi} - \overline{r}_v)^2}}. \tag{2.4}$$

É importante destacar que todas as similaridades são calculadas usando somente itens co-avaliados pelos usuários $\mathcal{I}_{u,v}$ [3], i.e. somente os itens que foram avaliados por ambos usuários. Inclusive, no trabalho de Houle et al. [10] é investigado como essa escolha é efetiva em reduzir o efeito da maldição da dimensionalidade [11]. Além das similaridades COS e CCP, uma revisão da literatura revela outras funções que podem substituir as apresentadas nesta seção. Por exemplos os trabalhos de Symeonidis et al. [12] e Bhattacharya et al. [13].

2.1.3 Filtragem colaborativa baseada em modelo

Diferente dos métodos baseados em memória, os métodos baseados em modelo modelam a interação entre usuários e itens. Para isso, normalmente, são aplicados modelos de aprendizado de máquina ou modelos estatísticos. Nas próximas seções, são apresentados e descritos três modelos clássicos literatura de FC: SVD, SVD++ e Restricted Boltzmann Machine(RBM).

2.1.3.1 SVD

O modelos de Decomposição em Valores Singulares(SVD) para FC descrito por Funk [14], mapeia usuários e itens em um espaço de variáveis latente com dimensão f, de forma que o produto interno seja capaz de explicar as interações entre usuários e itens, i.e. as notas dos usuários [3]. Nesta modelagem, um usuário u é representado por um vetor $p_u \in R^f$, cada item corresponde a um vetor $q_i \in R^f$ e a nota estimada que o usuário u atribuiu para o item i é fornecida pela equação 2.5:

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i^t p_u, \tag{2.5}$$

onde μ é a média global e b_i e b_u são, respectivamente, o intersepto do item e do usuário. O esquema da equação 2.5 pode visualizada na figura 2.1

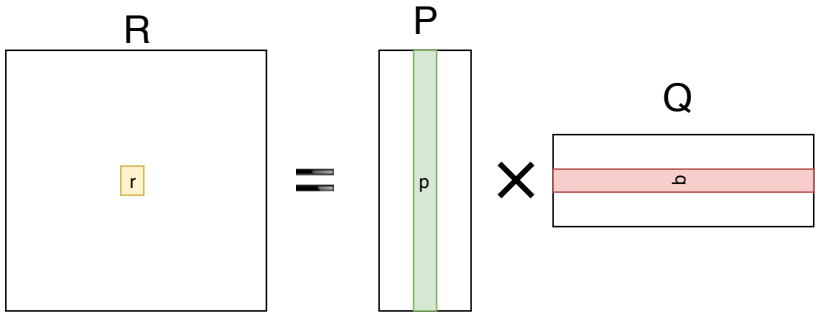

Figura 2.1: Visualização de como r é estimado pelo SVD

Os parâmetros deste modelo podem ser estimados com o uso de gradiente des-

cendente estocástico(GDE) para minimizar o erro quadrático médio regularizado [3], dado por:

$$\underset{b,q,p}{\operatorname{argmin}} \sum_{u,i\in(\text{Notasobservadas})}^{i} (r_{u,i}-\mu-b_i-b_u-q_i^t p_u)^2+\lambda(b_i^2+b_u^2+\|q_i\|^2+\|p_u\|^2), \quad (2.6)$$

onde o termo penalizador é ajustado pela constante λ.

2.1.3.2 SVD ++

Enquanto SVD constrói a previsão de uma nota somente com as iterações explícitas entre itens e usuários, o SVD++ [15] considera o feedback implícito, que são dados sobre o comportamento do usuário [16] de acordo com a equação:

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i^t\left(p_u + \|R(u)\|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j\right) \quad (2.7)$$

Aqui, é adicionado feedback implícito referentes aos itens que um usuário tenha avaliado ou não. A adição dessa informação no modelo melhora a qualidade das recomendações geradas [15].

2.1.3.3 Restricted Boltzmann Machine

Restricted Boltzmann Machine(RBM)[1] é um modelo gráfico probabilístico não direcionado que contém uma camada visível e uma camada escondida (entre as variáveis de cada camada não há conexão [18]). RBMs também podem ser usadas como blocos de construção de modelos profundos, conhecidos como Deep Restricted Boltzmann Machine [19]. A RBM mais simples considera que cada nó do modelo é uma variável aleatória de Bernoulli e sua função de energia, na notação matricial, é dada por:

$$E(v, h) = -b^T v - c^T h - v^T W h, \quad (2.8)$$

onde W é a matriz de pesos, v são as variáveis aleatórias da camada visível, h são as variáveis aleatórias da camada escondida, b e c são, respectivamente, o intercepto da camada de visível e da camada escondida. A figura 2.2 ilustra o grafo bipartido de uma RBM com uma camada visível e uma camada escondida com 6 variáveis latentes.

É importante assinalar que as RBMs foram aplicadas com sucesso em diversos domínios [20] e ganharam notoriedade no desafio do Netflix [21]. Enquanto a maioria

[1]Originalmente, RBM era conhecido como Harmonium [17] e foi desenvolvida por Paul Smolensky em 1986 e aplicada no domínio de sistemas cognitivos.

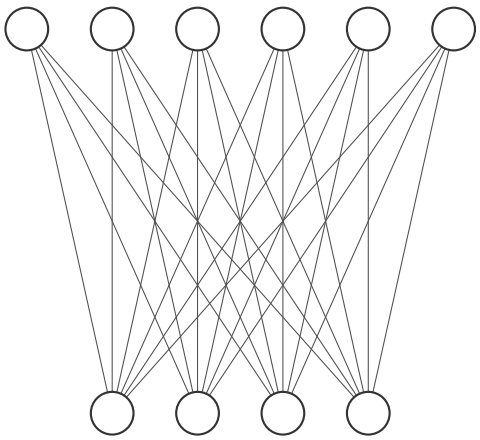

Figura 2.2: RBM com uma camada visível com 4 variáveis e uma camada escondida com 6 variáveis latentes

dos modelos empregados neste desafio eram baseados em vizinhos mais próximos ou em fatoração de matrizes, a RBM surgiu como a primeira solução competitiva baseada em redes neurais e esteve, inclusive, entre os 800 modelos usados na solução vencedora [22]. Na proposta de Hinton et al. [20] para o desafio do Netflix, cada usuário era modelado por uma RBM binária com parâmetros compartilhados. O treinamento das RBMs era feito por meio da minimização da contrastive divergence [23]. Seguindo a modelagem, suponha que um usuário tenha avaliado m filmes (sua RBM conterá m unidades visíveis) e X é uma matriz com dimensões k e m, onde $x_i^y = 1$ indica que o usuário u avaliou o filme y ou, caso contrário, $x_i^y = 0$, então:

$$p(v_d^y = 1|h) = \frac{\exp(b_i^y + \sum_{j=1}^{F} h_j W_{ij}^y)}{\sum_{l=1}^{k} \exp(b_i^l + \sum_{j=1}^{F} h_j W_{ij}^l)} \tag{2.9}$$

e

$$p(h_j = 1|X) = \theta(b_j + \sum_{i=1}^{m} \sum_{y=1}^{k} x_i^y W_{ij}^y), \tag{2.10}$$

onde W_{ij}^y é a matriz de pesos entre as notas y do filme i e a unidade escondida j, b_i^y é o intersepto da nota y para o filme i e b_j é o intersepto da unidade escondida j.

2.1.4 Trabalhos relacionados

Além dos métodos clássicos apresentados, durante a revisão da literatura, encontramos duas modelagens que abordam alguns dos objetivos deste trabalho. No primeiro trabalho descrito em [24], propõem o uso de modelos Autoencoders cuja função de custo é semelhante a proposta.

a função de custo usada pelos autores é semelhante a função de custo desta dissertação. O segundo, embora faça uso de um modelo para cada usuário, também modela as notas dos usuários como um resultado das interações entre ele e seus vizinhos. Em outras palavras, utiliza a vizinhança entre usuários como dados de entrada para o modelo. Os dois trabalhos são detalhados nas seções e

2.1.4.1 Autorec

O Autorec é um modelo baseado em Autoenconder [24] para encontrar uma representação latente da matriz de notas [25]. Para isso, a camada escondida do Autoencoder possui uma quantidade de variáveis inferior[2] ao da camada entrada. O exemplo de um Autoencoder é apresentado na figura 2.3

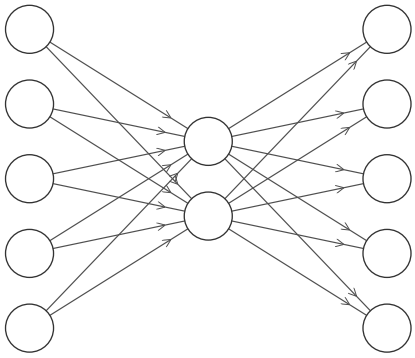

Figura 2.3: Um Autoencoder undercomplete

Entretanto, no caso do FC, a maior parte da matriz de notas não é observada. Sendo assim, os autores do Autorec modificaram a função de custo de forma que somente os pesos associados às notas observadas sejam corrigidos durante o treinamento do modelo. A função de custo é dada por:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i} \|r_i - h(r_i;\theta)\|_{\mathcal{O}}^2 + \lambda \cdot reg, \tag{2.11}$$

onde $h(r_i;\theta) = f(W \cdot g(V \cdot r_i + \mu) + b)$,$f(\cdot)$ e $g(\cdot)$ são funções de ativação, $\theta = \{W, V, \mu, b\}$, W, V as matrizes de peso e μ, b correspondem aos interceptos do autoencoder. Além disso, para a otimização da função de custo, os autores usaram Resilient propagation [26]. Os resultados obtidos com este modelo baseado em autoencoder superaram os resultados com RBM e SVD [24].

Existem outras experiências na literatura que estendem esse trabalho com auto-

[2]Um Autoenconder que possua a camada escondida com um número de neurônios inferior a dimensão da entrada é chamado de Autoencoder undercomplete

encoders, que contam com arquiteturas mais avançadas. Exemplos podem ser encontrados no trabalho de Strub [27], onde é aplicado um denoising autoenconder [28], e no trabalho de Pu et al. [29] que faz uso de um Autoenconder variacional [30].

2.1.4.2 Landmarks SVR

Os autores deste trabalho propuseram uma nova modelagem capaz diminuir o volume de dados necessários para treinar algoritmos supervisionados no problema de CF. A modelagem consiste em associar as notas dos usuários com as variáveis dos itens avaliados. As variáveis dos itens são definidas como as similaridade entre os itens e um conjunto de itens pré selecionados $\mathcal{L}$, composto pelos itens mais avaliados na base de dados. Portanto, o vetor com as variáveis do item v é definido como:

$$w_v = (s(i, \mathcal{L}_1), s(i, \mathcal{L}_2), .., s(i, \mathcal{L}_{|\mathcal{L}|})), \tag{2.12}$$

onde $s(\cdot)$ é uma função que mensura a similaridade entre dois itens. No caso deste trabalho, a função empregada foi a similaridade cosseno (equação 2.4). Em seguida, os autores propõe a construção de um conjunto de treinamento em que os vetores de features estão associados as notas de um usuário. Com o conjunto de treinamento construído, ele é usado como entrada para treinar um Support vector regression (SVR) [31], que deve aprender a mapear as interações dos itens com as notas atribuídas pelo usuário. Com essa modelagem os autores alcançaram uma redução significativa no tempo para o treinamento do SVR e não houve degradação na qualidade das notas previstas.

3. Proposta

Neste capítulo a proposta central do trabalho é descrita e formalmente defendida.

3.1 Modelagem proposta

Para atingir os objetivos estabelecidos no primeiro capítulo, propomos uma nova modelagem para o problema de filtragem colaborativa. Esta modelagem consiste em utilizar um modelo de aprendizado de máquina para aprender o mapeamento entre as relações de vizinhança dos itens e as notas da matriz de utilidade. Formalmente, a modelagem pode ser definida como: dado um conjunto de itens I, um conjunto de usuários U, uma matriz utilidade R_i, o modelo supervisionado deve aprender o mapeamento $\mathcal{F} : \overrightarrow{\mathsf{S}_\mathsf{i}} \mapsto \mathsf{R}_\mathsf{i}$, onde $\overrightarrow{\mathsf{S}_\mathsf{i}}$ é o vetor linha, de dimensão $\#\mathsf{I}$, contendo o valor de similaridade de $\mathsf{i} \in \mathcal{I}$ para todos os itens.

Para aprender $\mathcal{F}$ propomos utilizar uma rede neural artificial (RNA) feedfoward backpropagation. Esta escolha é baseada na capacidade deste modelo de aprender complexas relações não lineares [18]. A figura 3.1 ilustra como seriam as camadas de entrada e saída da RNA para o caso de $\#\mathsf{I} = 4$.

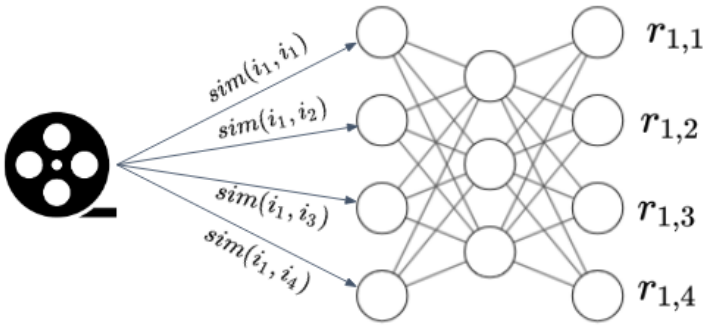

Figura 3.1: Entradas e saídas da RNA proposta

Entretanto, como visto no capítulo 2, a matriz R_i é intrinsecamente esparsa, ou

seja, R_i não supervisiona todas as entradas da RNA. Portanto, para aprender $\mathcal{F}$, é preciso modificar a RNA de forma com que somente as notas observadas sejam consideradas durante o backpropagation. Para isso, propomos uma função de custo que ignora o erro da RNA no caso de notas não supervisionados, a seguir

$$J(w) = \sum_{i \in (I)} \|(h_w(S_i) - R_i) \odot B_i\|_F^2, \tag{3.1}$$

$$J(w) = \sum_{i \in (I)} \|(h_w(S_i) \odot B_i - R_i \odot B_i)\|_F^2, \tag{3.2}$$

onde R_i é matriz de utilidade, S_i é a matriz de similaridade construída com a similaridade $s(\cdot)$, B é uma matriz "booleana"(com dimensões iguais a R) que indica se a nota R_i está disponível ou não, $\odot$ é o produto de Hadamard e F indica que queremos minimizar a norma de Frobenius. Reescrevendo a equação 3.2 chegamos em:

$$J(w) = \|(h_w(S) \odot B - R \odot B)\|_F^2 = Tr((h_w(S) \odot B)(h_w(S) \odot B)^t) \tag{3.3}$$

Nesta função de custo, o produto de Hadamard pondera a matriz notas R_i em função da disponibilidade das notas. Portanto, o algoritmo backprogation [32] não corrige os pesos associados as notas não disponíveis durante o treinamento da RNA. Enquanto isso, no caso dos pesos de notas disponíveis, estes serão corrigidos normalmente. Um esquema geral de como é realizado o treinamento da RNA é apresentado na figura 3.1.

Note que no esquema proposto da figura 3.1, o treinamento da RNA tem início com com a matriz de utilidade R e a matriz de similaridade de itens S_i. Cada linha de S_i é dada como entrada para a RNA com saída correspondente a i-ésima coluna de R.

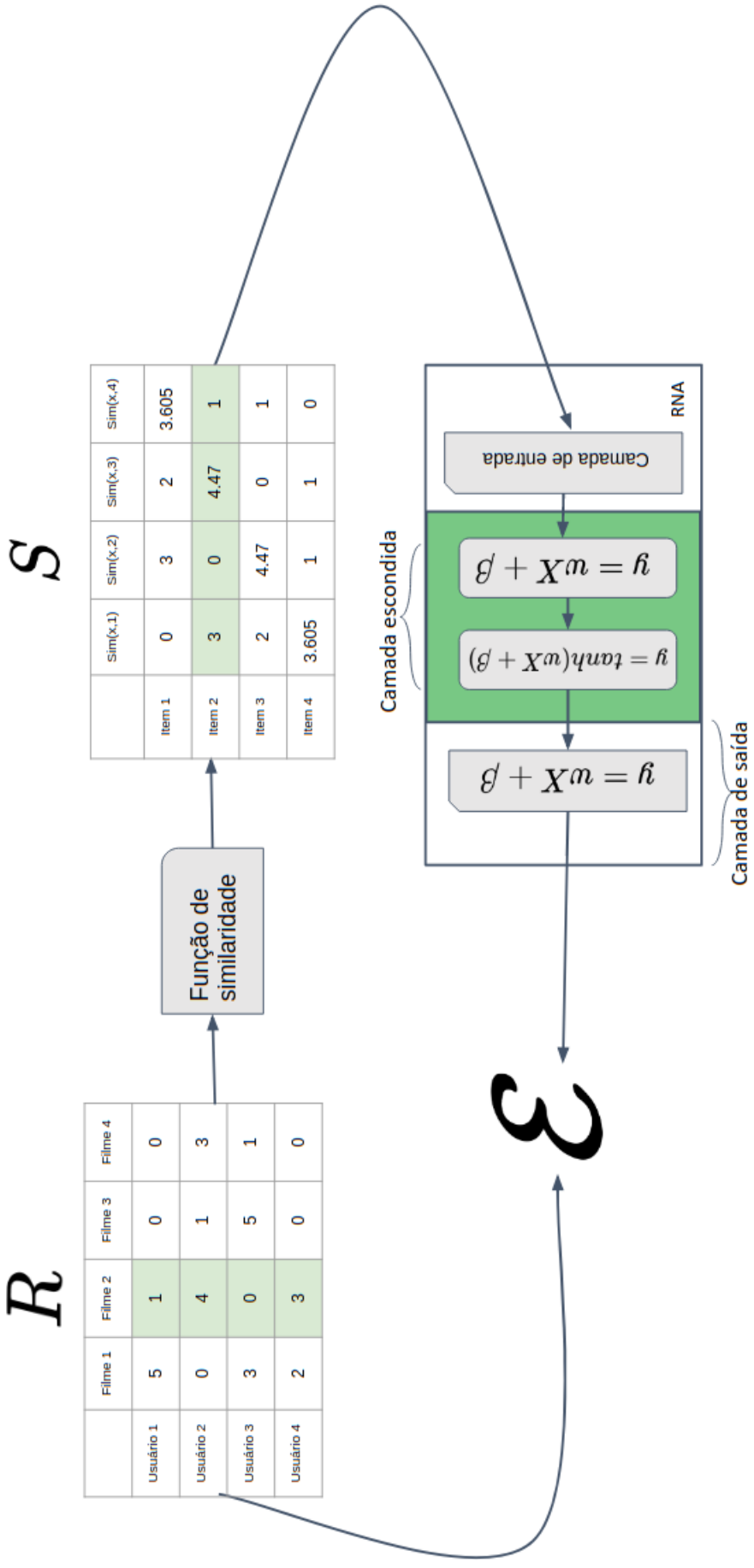

Figura 3.2: Visão geral do treinamento da RNA proposta

3.2 Exemplo visual do funcionamento da função de custo

A fim de ilustrar o funcionamento da RNA com a nova função de custo, apresentamos o exemplo a seguir: removemos aleatoriamente os valores de alguns pixels de uma imagem(figura 3.3) e verificamos se a RNA proposta é capaz de reconstruir a imagem original.

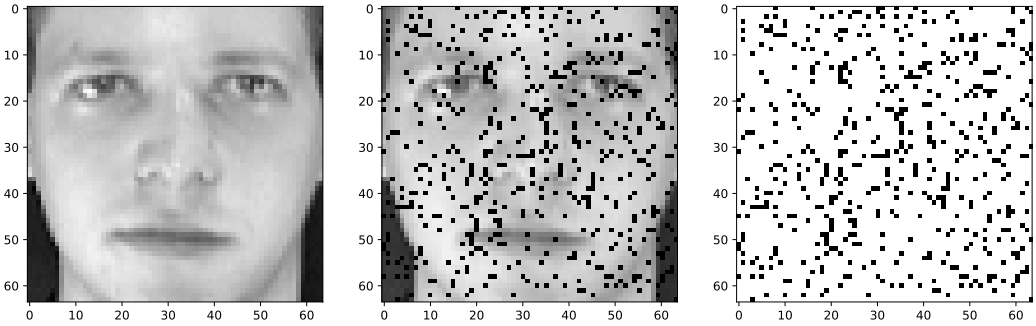

Figura 3.3: Imagem original, imagem corrompida e a matriz B

A reconstrução da imagem original foi feita com uma RNA com função de ativação tangente hiperbólica com uma camada escondida, seguiu o esquema apresentado na seção 3.1. A reconstrução final é apresentada na imagem 3.4

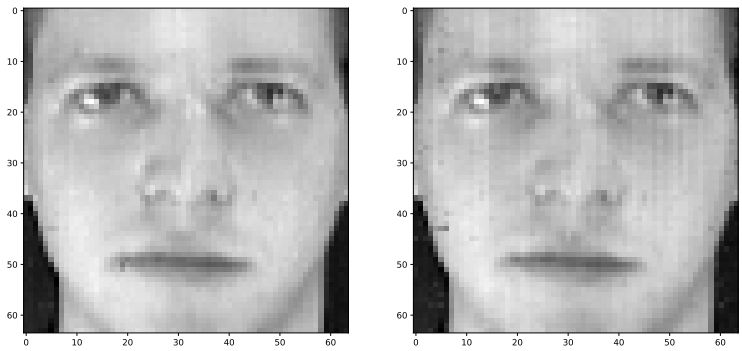

Figura 3.4: Imagem reconstruída usando S_{l2} e S_{COS} respectivamente

Conforme indicado pelos resultados na figura 3.4, a RNA é capaz de reconstruir a imagem original.Observe também, que a qualidade da reconstrução de S_{cos} é menor do a reconstrução de S_{l2}. Tal diferença será melhor explorada na seção 4.7.1.

3.3 Comparação da proposta com outros trabalhos da literatura

No trabalho de Lima et al. [33] as notas também são modeladas em função das relações de vizinhança. No entanto, os autores utilizaram uma SVR para aprender

cada uma das relações de vizinhança da base, ou seja, o número de SVR utilizadas é igual à #I. Enquanto isso, nossa proposta utiliza um único modelo para aprender todas as relações da base. Em função disso, o tempo de treinamento e inferência do modelo proposto é substancialmente menor.

Já no trabalho de Sedhain et al. [24], os autores também fizeram uso de uma rede neural com uma função de custo que mascara as notas não observadas. Entretanto, neste trabalho o modelo supervisionado não emprega nenhuma informação sobre a vizinhança dos itens.

4. Experimentos

Este capítulo tem como objetivo descrever os experimentos conduzidos, a metodologia empregada e a avaliação dos resultados experimentais.

4.1 Objetivos

O objetivo geral dos experimentos é verificar a hipótese acerca do uso da representação de usuários/itens como informação suficiente para um modelo supervisionado, com sua função de custo adaptada à saída esparsa do conjunto de treino para realizar boas previsões na filtragem colaborativa. Assim, com os experimentos buscamos responder

- qual é influência dos hiperparâmetros no desempenho do modelo?
- como o desempenho do método proposto se compara ao desempenho dos modelos do estado-da-arte?
- qual é a relação entre os pesos da camada escondida da RNA e os dados?

Para tal, faz-se necessário avaliar as condições de convergência, ajuste de hiperparâmetros e limitações do modelo em relação ao estado-da-arte.

4.2 Dados experimentais

Os experimentos conduzidos foram feitos em três conjuntos de dados do domínio de recomendação de filmes. Estes conjuntos foram adotados em diversos trabalhos de FC. O conjunto de dados MovieLens100k [34] é formado por cem mil notas. Com

valores entre 1 e 5, de 943 usuários da plataforma MovieLens que avaliaram 1682 filmes. O segundo conjunto de dados tem a mesma origem, mas possui um milhão de avaliações de 6040 usuários sobre 3952 filmes, conhecido como MovieLens1M.

Dados	N. de usuários	N. de filmes	N. de notas	Esparsidade
MovieLens100k	943	1682	100.000	93,7%
MovieLens1M	6040	3706	1.000.209	95,53%
FilmTrust35k	1508	2071	35.497	98,86%

Tabela 4.1: Características das bases de dados usadas nos experimentos

O último conjunto de dados é disponibilizado pela plataforma FilmTrust [35], contém 35 mil notas, com valores entre 0,5 e 5, de 1508 usuários que avaliaram 2071 filmes.

Embora os três conjuntos sejam do domínio de filmes, estes possuem características distintas. A base com a menor densidade é a FilmTrust35k, com somente 1,14% das avaliações da matriz de utilidade preenchida. Além disso, a discretização entre as bases é diferente, o que também pode influenciar os resultados. A tabela 4.1 apresenta um resumo das características mencionadas. Repare que, a princípio, quanto maior a esparsidade, mais complexo é o modelo. Além disso, FilmTrust35k contém dados sobre as relações de confiança entre os usuários. Tais informações foram desconsideradas nos experimentos.

4.3 Pré-processamento

Na etapa de pré-processamento, a matriz de similaridade S e a matriz de notas R_{treino} receberam um processamento específico. A matriz S foi redimensionada de forma que cada ponto estivesse no intervalo [-1,1]. Para isso, foi aplicada a normalização Min-Max, dado por:

$$\hat{S_i} = 2\frac{S_i - \min(S_i)}{\max(S_i) - \min(S_i)} - 1, \tag{4.1}$$

onde S_i é i-ésima coluna de S.A escolha desse método de normalização deu-se a fim de evitar possíveis distorções nos dados, gerando problemas de convergência e viés.

No que concerne à matriz R_{treino}, seu procedimento de normalização seguiu o descrito por Braida et al. [36]. Nesta normalização, cada entrada da matriz R_{treino} é subtraída pela nota média do usuário que a gerou. De acordo com os autores, esta

foi a normalização da matriz de notas que produziu os melhores resultados quando usada em conjunto com uma rede neural.

4.4 Medidas de performance

Como medidas para a avaliação das previsões geradas nos experimentos, foram usadas duas métricas sugeridas por Adomavicius et al. [4]: Mean Absolute Error(MAE) e Root Mean Squared Error(RMSE).

O MAE mensura o erro médio absoluto entre as notas previstas pelo modelo e nota verdadeira, conforme a equação a seguir.

$$\text{MAE} = \frac{\sum_{i,u} |\hat{r}_{i,u} - r_{i,u}|}{n}, \tag{4.2}$$

onde $r_{i,u}$ é a nota dada pelo usuário u para o item i e a previsão desta nota é dada por $\hat{r}_{i,u}$. A principal vantagem dessa métrica é a interpretabilidade, por exemplo, um erro de 0,7 indica que o modelo em média está com um desvio de 0,7 das notas dadas pelos usuários do SR.

A medida RMSE em comparação com o MAE não possui o mesmo grau de interpretabilidade, pois penaliza os erros de forma diferente. Enquanto o MAE calcula o erro absoluto, o termo quadrático do RMSE penaliza mais os maiores erros, conforme descrito em:

$$\text{RMSE} = \sqrt{\frac{\sum_{i,u} (\hat{r}_{i,u} - r_{i,u})^2}{n}}. \tag{4.3}$$

Embora existam outras medidas para avaliar SRs, nosso foco foi explorar a acurácia das previsões de notas.

4.5 Metodologia experimental

Para a avaliação da técnica proposta cinco experimentos foram realizados:

- Experimento I: investiga o impacto de 4 funções de similaridade $s(\cdot)$ na qualidade das recomendações: Cosseno(COS), Coeficiente de correlação de Pearson(CCP) e a distancia de Minkowski com graus 1 e 2. Para isso, foram

geradas 4 matrizes de similaridade distintas a partir de cada uma das funções, em seguida, treinou-se os modelos e avaliou-se seus respectivos MAE.

- Experimento II: realizou-se testes para determinar valores apropriados da taxa de aprendizado e do tamanho de batch do GDE. Para tal, os melhores valores foram obitods por busca de em grade,

    1. Para a taxa de aprendizado, a busca teve início na vizinhança de 1e-3, valor considera como padrão para o início de buscas [37].
    2. Em relação ao tamanho do batch, não há na literatura um trabalho sobre o tamanho do batch no domínio de FC, portanto a busca abrangeu um grande espaço[8,...,512]. Além disso, o tamanho ideal para treinamento ainda é um assunto de intenso debate na comunidade. De acordo com Masters et, al [38] batchs grandes, i.e. $\mathfrak{m} > 32$, prejudicam o poder de generalização do modelo, entretanto o trabalho de Hoffer et. al [39] indica que o tamanho do batch não influência na capacidade de generalização do modelo.

- Experimento III: é responsável por investigar o comportamento do modelo em função dos hiperparâmetros da RNA. Considera-se os seguintes hiperparâmetros:

    1. Número de neurônios: variou-se entre 64 e 1024.
    2. Weight decay, para isso foi realizada uma busca de valores seguindo uma escala logarítmica no intervalo[1e-3,1e0].
    3. Função de ativação: em uma RNA rasa com 64 neurônios. Tratou-se com diferentes tipos de funções.
    4. O número de camadas escondidas: variou-se entre 2 e 8.

- Experimento IV: analisou-se a relação entre os pesos da camada escondida e os itens de entrada. Dessa forma, é esperado que seja possível estabelecer uma relação entre a o número de avaliações de um item e a representação que a camada escondida da RNA aprendeu. Está análise é baseada nas evidências encontradas no trabalho de Lima et. al. [33].

- Experimento V: a modelagem proposta foi comparada com seis modelos de FC[1] em três bases de dados descritas na subseção anterior. Os modelos utilizados

[1]Os códigos referentes aos modelos são da biblioteca Librec [40]

nesta comparação foram: AutoRec [24], Bayesian Probabilistic Matrix Factorization [41], Non-Negative Matrix Factorization [42], Probabilistic Matrix Factorization [43], SVD++ [15] e Restricted Boltzmann Machine [20].

4.6 Esquema de treinamento e avaliação de erro fora da amostra

A avaliação dos experimentos utilizou validação cruzada Hold-out, i.e. os dados foram separados em três subamostras: 80 % para treino, 10 % para validação e 10% para teste. Tal divisão é ilustrada na figura 4.1

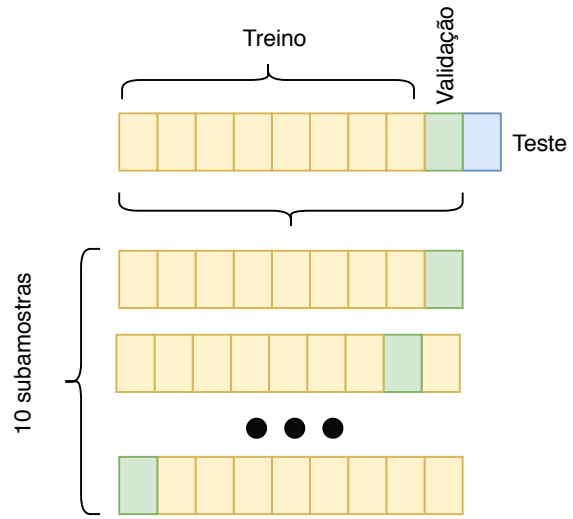

Figura 4.1: Esquema de validação cruzada utilizado nos experimentos

Os experimentos para análise e seleção de hiperparâmetros usaram a base de dados MovieLens 100k e validação cruzada 10 folds com os conjunto de treino e de validação. Já a subamostra de teste foi reservada para o experimento de comparação de modelos.

4.6.1 Busca por hiperparâmetros

No que se refere a busca por hiperparâmetros da RNA, a metodologia seguida foi semelhante a sugerida por Bengio Yoshua [37]. Segundo o autor, a busca por hiperparâmetros, quando há um único computador[2] disponível, pode ser realizada de forma análoga ao coordinate descent, i.e. a otimização trata de somente um hiperparâmetro por vez.

[2]Os experimentos foram realizados em um único computador equipado com: Geforce GTX 1060 6GB, AMD Ryzen 1700 e 32GB de memória Ram.

4.6.2 Treinamento da RNA

Quanto ao treinamento da RNA, todos os experimentos usaram gradiente descendente estocástico(GDE) com momento definido em 9e-1 e com uma taxa de decaimento exponencial de 95e-2 a cada 10 épocas. Os otimizadores adaptativos, e.g. ADAM [44], não foram considerados para essa tarefa devido as questões teóricas, que ainda não foram completamente esclarecidas, sobre a convergência e a generalização dessa classe de optimizadores. Trabalhos recentes como [45] e [46] abordam essa questão e mostram evidências que RNAs treinadas com GDE possuem maior poder de generalização do que redes treinadas com otimizadores adaptativos.

Outro aspecto do treinamento considerado neste trabalho foi a inicialização dos pesos da RNA. Para iniciar os pesos das redes ReLU [47], ELU [48] e Swish [49] foram usadas as heurísticas Xavier uniforme [50] ou Kaiming uniforme [51]. No caso das redes SELU [52], seguiu-se a recomendação de Klambauer et al. [52].

4.7 Resultados e discussão

Nesta seção, os resultados de cada experimento são apresentados e analisados.

4.7.1 Experimento I

Neste experimento foi encontrada uma pequena diferença entre os valores de MAE(tabela 4.2) a partir de diferentes funções de similaridade.

Similarity	MAE
COS	0.7285
CCP	0.7295
L_2	0.7217
L_1	0.7259

Tabela 4.2: Resultados do experimento I

Embora estas diferenças sejam pequenas, as RNAs treinadas com similaridade L_2 e L_1 apresentaram uma taxa de convergência maior do que aquelas treinadas com COS e CCP. Tal diferença de taxa pode ser observada na figura 4.2.

Observe que o número de épocas necessárias para a RNA COS atingir o plateau é maior do que a RNA L_2. Este comportamento sugere que L_2 possui vantagem no sentido de informar melhor a RNA. Tal argumento pode ser sustentado uma vez que

COS contém somente a informação sobre o ângulo entre os vetores de notas [35], enquanto que L_2 incorpora o ângulo e a norma. Dados estes resultados, os próximos experimentos foram desenvolvidos com a distância L_2.

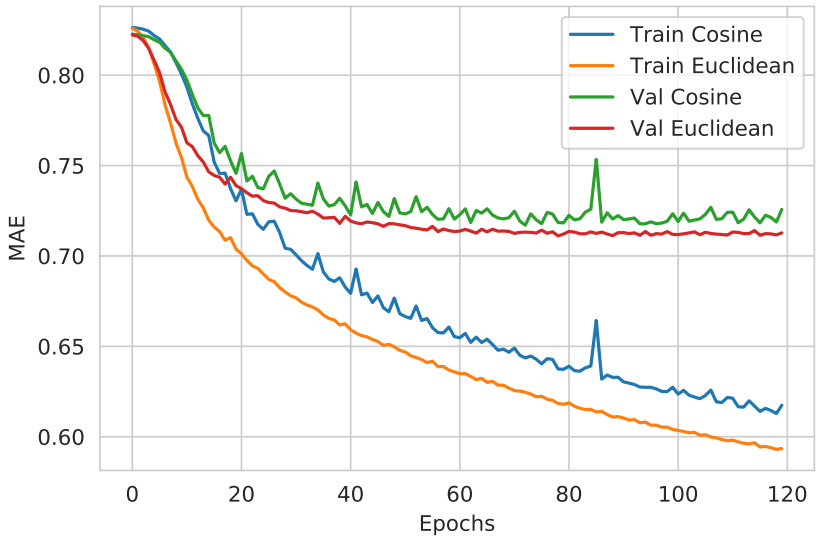

Figura 4.2: Treino de uma RNA COS e de uma RNA L_2

4.7.2 Experimento 2

Neste experimento são avaliados dois hiperparâmetro do GDE: Taxa de aprendizado e tamanho do batch. A taxa de aprendizado foi avaliada no intervalo [1e-4,5e-2] seguindo uma escala logarítmica, e o resultado desta avaliação está na figura 4.3.

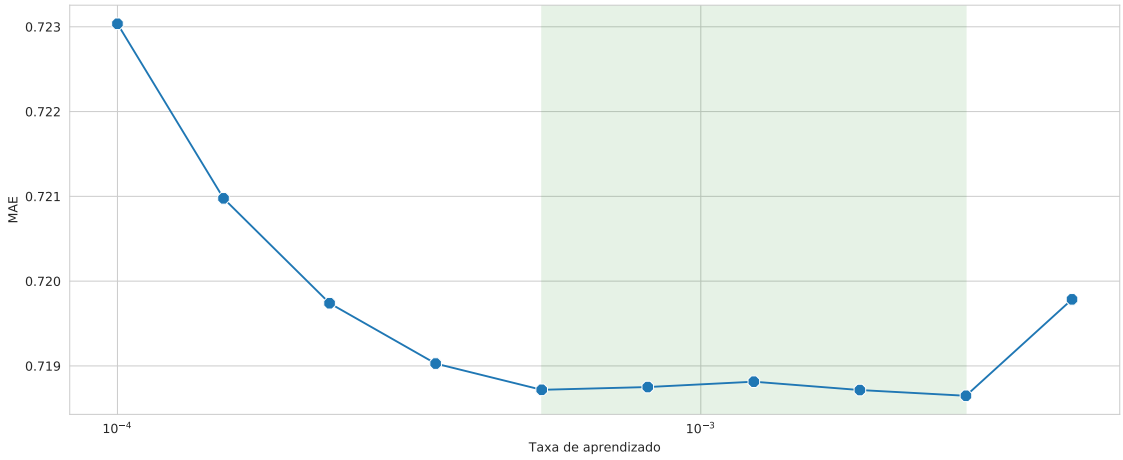

Figura 4.3: Impacto da taxa de aprendizado

Os resultados indicam que há uma região que resulta num pequeno ganho de performance. Entre as taxas de aprendizado da região sombreada, selecionamos o

valor 8e-3, pois este não está na borda da região e é pequeno o suficiente para não prejudicar à convergência nos próximos experimentos.

Na avaliação do impacto do batch batch(entre 8 e 128),a figura 4.4 mostra que mais do que 32 amostras ou menos do que 16 amostras por iteração do GDE degradam a performance da rede. O restante dos experimentos foi feito com um tamanho do batch igual à 32.

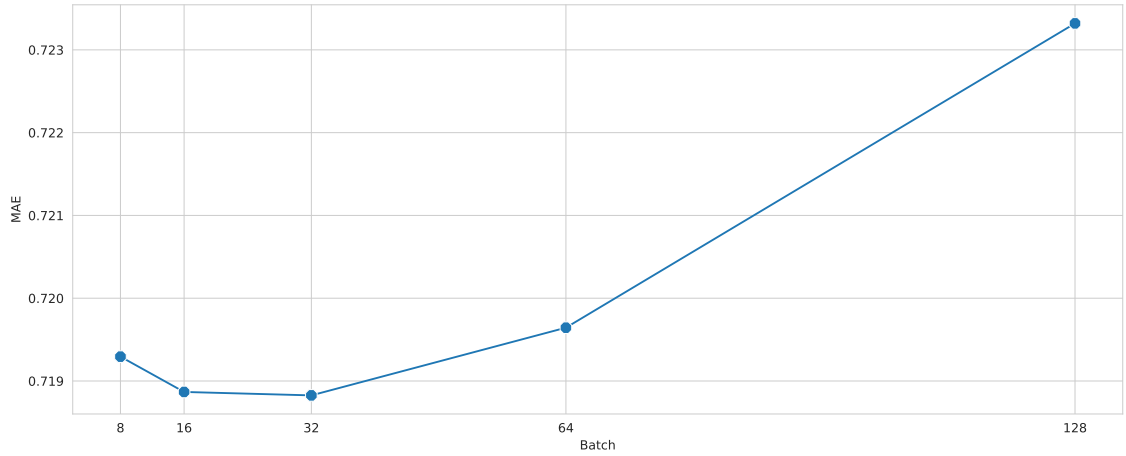

Figura 4.4: Impacto do tamanho do batch

4.7.3 Experimento 3

A figura 4.5 apresenta os valores de MAE variando-se o número de neurônios.

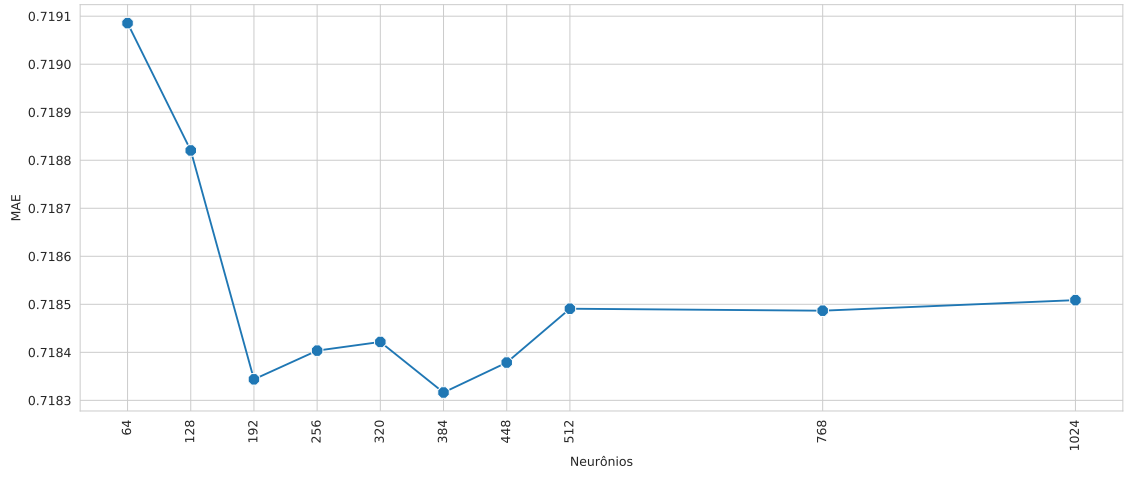

Figura 4.5: Efeito do número de neurônios em uma rede de uma única camada escondida

Note que, mesmo uma rede incompleta com 64 neurônios possui complexidade suficiente para generalizar para dados de fora da amostra. Entretanto, selecionamos a arquitetura com 384 neurônios tendo em vista que a regularização Weight

decay ainda será ajustada e, de acordo com Bengio [37], valores maiores que o ideal geralmente não prejudicam muito o desempenho da generalização.

Em seguida, analisamos o papel do Weight decay no controle da complexidade do modelo.

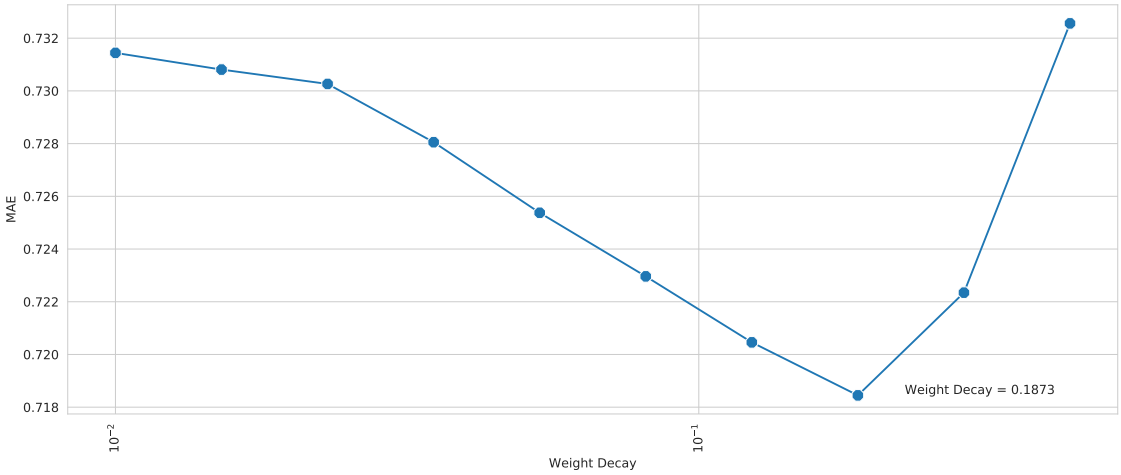

Figura 4.6: Impacto do Weight decay

A figura 4.6 apresenta o Weight decay variando o controle da complexidade do modelo. Observe que, o Weight decay é extremamente importante na modelagem.

Uma restrição excessivamente forte ou fraca no tamanho dos pesos da rede pode afetar diretamente no resultado final. O fator de regularização que apresentou o melhor resultado foi 0.1873, portanto será emprega nos próximos experimentos.

A próxima componente analisada é a função de ativação, para isso foi avaliado o MAE de quatro funções de ativação. A figura 4.7 mostra os valores de MAE para cada função. Repare que, o resultado desta avaliação indica que a escolha da função de ativação não desempenha um papel fundamental no desempenho de modelos rasos. As diferenças observadas podem ser explicadas devido às funções de ativação necessitarem de diferentes taxas de aprendizado [48], enquanto que neste experimento foi utilizada somente uma (8e-4).

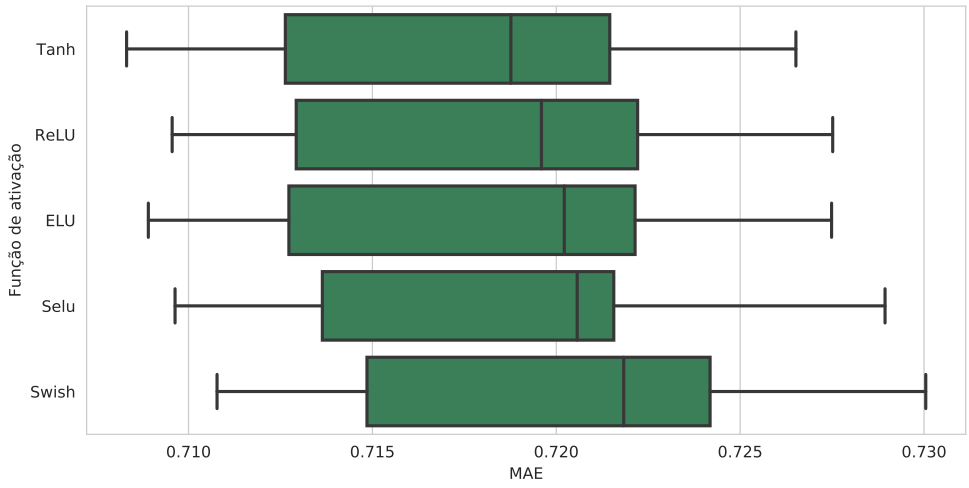

Figura 4.7: Funções de ativação em uma rede com uma camada escondida

O último hiperparâmetro da RNA a ser avaliado é o impacto do número de camadas escondidas com diferentes funções de ativação no MAE. A figura 4.8 apresenta valores de MAE variando com a profundidade da arquitetura com o número de neurônios constante. De acordo com estes resultados, o número de camadas escondidas desempenha um papel importante na performance do modelo. Por exemplo, a RNA ELU com 5 camadas escondidas reduziu o MAE em 1,97 % em relação ao modelo raso.

Outro aspecto do resultado diz respeito às funções de ativação. Note, a estabilidade do MAE das RNA SELU. Este comportamento pode ser explicado pela característica da função SELU, pois ela possui a mesma propriedade do Batch normalization [53], que reduz o covariate shift interno da RNA.

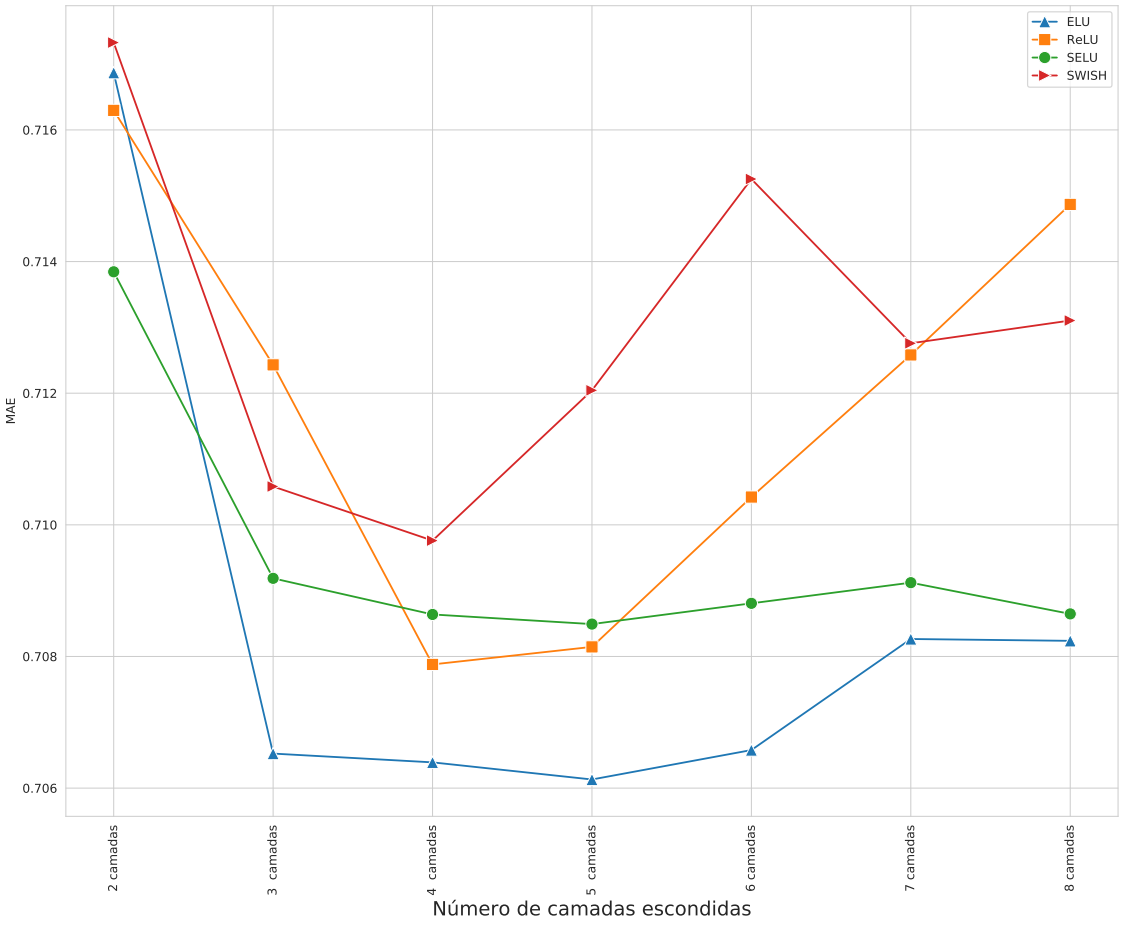

Figura 4.8: Impacto da profundidade da RNA no MAE

4.7.4 Experimento 4

A figura 4.9 representa a matriz de pesos da camada escondida de uma RNA ReLU com 64 neurônios.

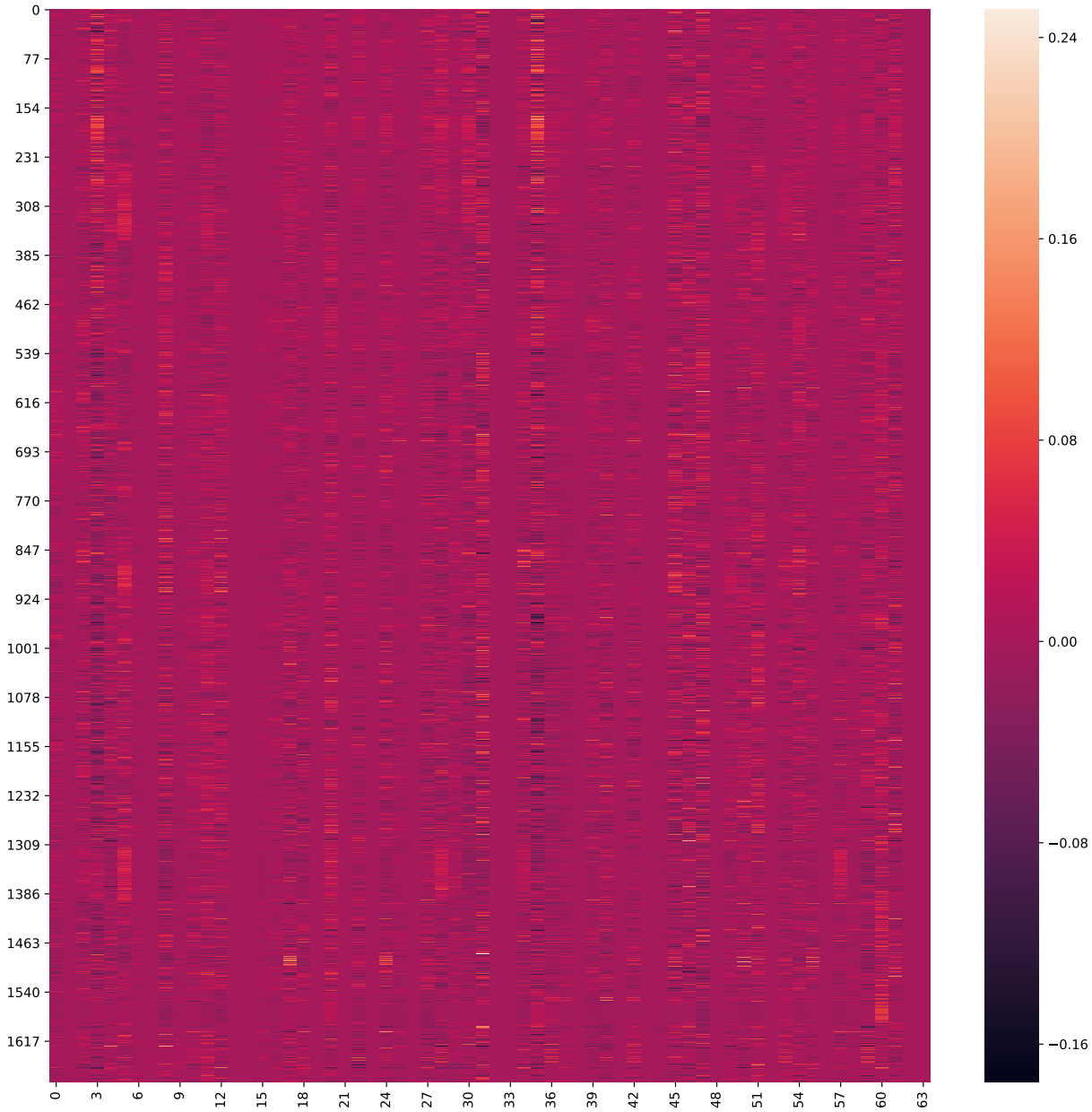

Figura 4.9: Visualização da matriz de pesos

A primeira vista não é possível encontrar nenhum padrão na matriz de pesos. Já o histograma com a distribuição das normas L_2 dos pesos, (figura 4.10), revela que alguns itens são muito mais informativos para a rede do que outros.

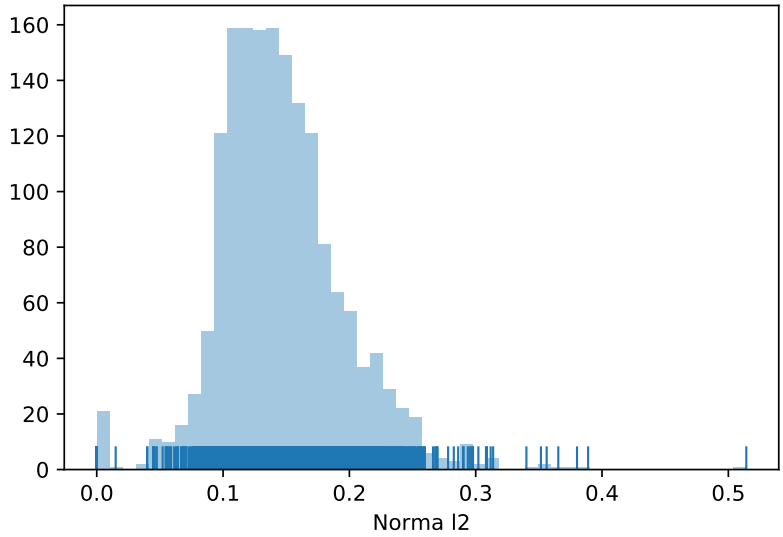

Figura 4.10: Distribuição das normas dos pesos

A análise dos itens menos informativos, indica que tais itens estão associados a itens que foram avaliados por usuários com poucas co-avaliações.

4.7.5 Experimento 5

A tabela 4.3 apresenta os resultados da comparação entre modelos na base Filmtrust35k. Nesta primeira comparação, o método proposto obteve a melhor performance em ambos os critérios.

Técnica	MAE	RMSE	Variáveis latentes
Modelo proposto	**0.583**	**0.765**	128
Autorec	0.621	0.820	200
SVD++	0.600	0.798	12
PMF	0.667	0.907	10
NMF	0.639	0.852	150
BPMF	0.598	0.780	20
RBM	0.653	0.866	400

Tabela 4.3: Resultados Filmtrust35k

O modelo que proporcionou o resultado da tabela 4.3 é uma RNA ELU com uma

única camada escondida com 128 neurônios.

Na base de dados Movielens100k (tabela 4.4) o modelo não obtém o melhor resultado no RMSE. No entanto, a diferença entre o modelo e o melhor resultado(BPMF) é inferior a 1%.

Técnica	MAE	RMSE	Variáveis latentes
Modelo proposto	**0.695**	0.896	320,320,320
Autorec	0.700	0.894	200
SVD++	0.711	0.906	10
PMF	0.724	0.919	10
NMF	0.750	0.951	100
BPMF	0.697	**0.891**	20
RBM	0.755	0.962	300

Tabela 4.4: Resultados Movielens100k

A última comparação é feita no conjunto de dados MovieLens1M (tabela 4.5). O método proposto obteve, junto com SVD++, o melhor MAE entre os modelos avaliados. Quanto ao RMSE, o modelo que obteve o melhor resultado foi o Autorec.

Técnica	MAE	RMSE	Variáveis latentes
Modelo proposto	**0.664**	0.852	320,320,320
Autorec	0.667	**0.848**	500
SVD++	**0.664**	0.856	12
PMF	0.697	0.878	12
NMF	0.726	0.919	100
BPMF	0.673	0.853	20
RBM	0.716	0.927	300

Tabela 4.5: Resultados Movielens1M

4.8 Considerações finais

Após realizarmos os experimentos, podemos arriscar respostas para as questões levantadas no início deste capítulo.

A representação usuário/itens e a função de custo, segundo os resultados do experimento 5, geram boas recomendações. Inclusive, nos três conjuntos de dados que utilizamos para os experimentos, a modelagem proposta deste trabalho obteve resultados melhores ou similares ao estado-da-arte.

No que diz respeito ao ajuste de hiperparâmetros do método proposto, os experimentos indicaram que o Weight decay e a profundidade da RNA são os dois hiperparâmetros com maior impacto no resultado final. Embora os experimentos 2 e 3 avaliassem os resultados sobre os hiperparâmetros da rede e do GDE, uma análise mais sensível sobre os resultados é de difícil elaboração. Uma interpretação mais minuciosa sobre os resultados das buscas por hiperparâmetros não é necessária e está fora do escopo de trabalho.

Em relação a análise dos pesos da rede, foi possível estabelecer que há uma relação entre o número de avaliações dos itens e dos usuários com o tamanho da norma do peso da camada escondida. Isto permite pensar em desdobramentos para a melhoria de novos métodos e técnicas de FC.

5. Conclusões e trabalhos futuros

Este capítulo aborda as considerações finais ao trabalho desenvolvido e suas possíveis extensões.

5.1 Conclusões

Este trabalho apresentou uma nova modelagem para o problema de filtragem colaborativa. Nossa proposta modela as notas dos itens de um SR a partir de relações espaciais dada por suas preferências. Para aprender tais associações, desenvolvemos uma rede RNA com uma função de custo capaz de lidar com dados de elevadíssima esparsidade.

Para avaliar nossa proposta, realizamos diversos experimentos. Em tais experimentos exploramos o comportamento do modelo em função de seus hiperparâmetros e o comparamos com outros modelos da literatura. A comparação mostrou que nossa proposta é capaz de igualar ou superar modelos do estado da arte de FC nas bases de dados testadas. Isto sugere que nossa proposta é promissora, no sentido de ser uma alternativa viável ao estado-da-arte.

Outro aspecto explorado neste trabalho é a relação entre os pesos da RNA e a popularidade dos itens. Em um dos experimentos, apresentamos evidências sobre a relação entre o número de co-avaliações de um usuário e os pesos da RNA. Tal resultado nos sugere que a utilidade de itens/usuários possuem valor informativos distintos, segundo possível futuramente explorar esse aspecto em questões como cold start.

A contribuição mais relevante deste trabalho foi validar a hipótese de que as relações entre os vizinhos de um item são suficientes para explicar as notas do mesmo. Para aprender tais relações, foi desenvolvida uma nova arquitetura de RNA

feedfoward backpropagation, com uma função de custo específica capaz de lidar com saídas esparsas.

5.2 Trabalhos futuros

Os resultados do experimento I indicam que a função de similaridade utilizada para a construção da matriz de similaridade tem um papel importante na modelagem. Portanto, futuros trabalhos poderiam explorar se outras funções de similaridade disponíveis na literatura aumentam a acurácia do modelo. Além disso, também seria possível substituir a matriz de similaridade por uma RNA que aprende as similaridade entre os itens. Por exemplo, no trabalho de Chopra et al. [54], é feito algo semelhante a essa hipótese, mas no domínio de visão computacional.

Referências Bibliográficas

[1] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," ACM Transactions on Management Information Systems (TMIS), vol. 6, no. 4, p. 13, 2016.

[2] T. Hennig-Thurau, A. Marchand, and P. Marx, "Can automated group recommender systems help consumers make better choices?" Journal of Marketing, vol. 76, no. 5, pp. 89–109, 2012.

[3] F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook. Springer, 2015. [Online]. Available: https://doi.org/10.1007/978-1-4899-7637-6

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Transactions on Knowledge & Data Engineering, no. 6, pp. 734–749, 2005.

[5] J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of massive datasets. Cambridge university press, 2014.

[6] R. Burke, "Hybrid recommender systems: Survey and experiments," User modeling and user-adapted interaction, vol. 12, no. 4, pp. 331–370, 2002.

[7] D. Bollen, B. P. Knijnenburg, M. C. Willemsen, and M. Graus, "Understanding choice overload in recommender systems," in Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010, pp. 63–70.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM, 2001, pp. 285–295.

[9] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal, "A collaborative filtering approach to mitigate the new user cold start problem," Knowledge-Based Systems, vol. 26, pp. 2f25–238, 2012.

[10] M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Can shared-neighbor distances defeat the curse of dimensionality?" in International Conference on Scientific and Statistical Database Management. Springer, 2010, pp. 482–500.

[11] R. Bellman, Dynamic programming. Courier Corporation, 2013.

[12] P. Symeonidis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos, "Collaborative filtering: Fallacies and insights in measuring similarity." Universitaet Kassel, 2006.

[13] G. Bhattacharya, K. Ghosh, and A. S. Chowdhury, "An affinity-based new local distance function and similarity measure for knn algorithm," Pattern Recognition Letters, vol. 33, no. 3, pp. 356–363, 2012.

[14] S. Funk, "Netflix update: Try this at home," 2006.

[15] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008, pp. 426–434.

[16] D. W. Oard, J. Kim et al., "Implicit feedback for recommender systems," in Proceedings of the AAAI workshop on recommender systems, vol. 83. WoUongong, 1998.

[17] P. SMOLENSKY, "Information processing in dynamical systems: Foundations of harmony theory."

[18] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep learning. MIT Press, 2016, vol. 1.

[19] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in Iberoamerican Congress on Pattern Recognition. Springer, 2012, pp. 14–36.

[20] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in Proceedings of the 24th international conference on Machine learning. ACM, 2007, pp. 791–798.

[21] J. Bennett, S. Lanning et al., "The netflix prize," in Proceedings of KDD cup and workshop, vol. 2007. New York, NY, USA, 2007, p. 35.

[22] A. Feuerverger, Y. He, and S. Khatri, "Statistical significance of the netflix challenge," Statistical Science, pp. 202–231, 2012.

[23] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," Neural computation, vol. 14, no. 8, pp. 1771–1800, 2002.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in Proceedings of the 24th International Conference on World Wide Web. ACM, 2015, pp. 111–112.

[25] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," arXiv preprint arXiv:1707.07435, 2017.

[26] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in Neural Networks, 1993., IEEE International Conference on. IEEE, 1993, pp. 586–591.

[27] F. Strub and J. Mary, "Collaborative filtering with stacked denoising autoencoders and sparse inputs," in NIPS workshop on machine learning for eCommerce, 2015.

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 1096–1103.

[29] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017, pp. 305–314.

[30] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in Advances in neural information processing systems, 2016, pp. 2352–2360.

[31] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in Advances in neural information processing systems, 1997, pp. 155–161.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, p. 533, 1986.

[33] G. R. Lima, C. E. Mello, and G. Zimbrao, "A new modeling for item ratings using landmarks," in 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018, pp. 1–8.

[34] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, pp. 19:1–19:19, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2827872

[35] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel bayesian similarity measure for recommender systems." in IJCAI, 2013, pp. 2619–2625.

[36] F. Braida, C. E. Mello, M. B. Pasinato, and G. Zimbrão, "Transforming collaborative filtering into supervised learning," Expert Systems with Applications, vol. 42, no. 10, pp. 4733–4742, 2015.

[37] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in Neural networks: Tricks of the trade. Springer, 2012, pp. 437–478.

[38] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," arXiv preprint arXiv:1804.07612, 2018.

[39] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in Advances in Neural Information Processing Systems, 2017, pp. 1731–1741.

[40] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith, "Librec: A java library for recommender systems." in UMAP Workshops, vol. 4, 2015.

[41] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 880–887.

[42] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1273–1284, 2014.

[43] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in Advances in neural information processing systems, 2008, pp. 1257–1264.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[45] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," 2018.

[46] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2018.

[47] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[48] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," arXiv preprint arXiv:1511.07289, 2015.

[49] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2018.

[50] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in NIPS-W, 2017.

[52] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in Advances in Neural Information Processing Systems, 2017, pp. 971–980.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[54] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 539–546.