Implementação de um Sistema de Recomendação de filmes utilizando AutoEncoders – 2021/2*

*Nota: Projeto da discplina Tóícos Avançados de Aprendizagem de Máquina do 2o. Semestre de 2021

Aluno: Carlos Fonseca
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
henrique_fonseca@hotmail.com

Aluno: Willian Guerreiro
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
wguerreiro31@gmail.com

***Resumo*—O crescente volume de informações e de conteúdos oferecidos na internet (jogos, músicas, filmes, etc) torna cada vez mais difícil aos usuários a tarefa de encontrar o que precisam com facilidade e rapidez. Segundo Sam Cook [6], em 2021, a Netflix mantém uma base ativa de, aproximadamente, 15.000 (quinze mil) filmes à disposição de seus mais de 200 milhões de assinantes e utiliza-se de sistemas de recomendação para sugerir aos seus clientes conteúdos relevantes e relacionados aos seus históricos de consumo. A filtragem colaborativa é amplamente usada neste tipo de sistema, porém torna-se inviável quando a base de dados é dispersa e de dimensões elevadas. Este artigo apresenta a implementação de um sistema de recomendação de filmes da base de dados *MovieLens 100k*, por meio de uma rede do tipo *autoencoder* de uma única camada intermediária, com até 100 neurônios e função de ativação linear. Para o treinamento dos pesos da rede foi utilizado o algoritmo SGD (*Stocastic Gradient Descent*) com retropropagação. A técnica apresentou resultados promissores, tendo alcançado rápida convergência e valores do erro médio quadrático (MSE) da ordem de 0.096.**

***Index Terms*—sistemas de recomendação, decomposição em valores singulares, autoencoders, gradiente descendente estocástico**

I. Introdução

Diante da imensa variedade de conteúdos e produtos (músicas, filmes, livros, viagens, etc.) disponíveis nos principais portais de comércio eletrônico na internet (ex: Amazon, Netflix, AliExpress, TripAdvisor, outros), tornou-se uma tendência o uso de ferramentas online, conhecidas como sistemas de recomendação, com o objetivo de impulsionar as vendas, recomendando produtos aos usuários com base em seu perfil e histórico de compras.

Conforme ilustrado na figura 1 [2], as fontes de informações necessárias para produzir recomendações, em geral, são: o feedback do próprio usuário e dos outros usuários pares do sistema, avaliações dos itens que estão sendo recomendados e quais as necessidades eles satisfazem.

É uma prática comum o consumidor buscar informações sobre produtos e serviços com conhecidos que já compraram aquele item. Esse é o conceito da abordagem conhecida como Filtragem Colaborativa, na qual as recomendações são feitas com base em itens consumidos por usuários cujos gostos e preferências são semelhantes aos do usuário alvo.

Como exemplo, Billsus e Pazzani [4] identificaram as principais deficiências das técnicas de filtragem colaborativa baseadas em correlação e mostraram como esses problemas podem ser solucionados por meio de algoritmos de classificação. Realizando experimentos sobre um conjunto de dados de classificações de filmes, aplicaram a técnica de decomposição em valores singulares (SVD) para reduzir a dimensionalidade da matriz com as avaliações dos filmes pelos usuários. Isso permite que os usuários se tornem preditores das preferências uns dos outros, sem sobreposição de itens avaliados.

Por sua vez, Ferreira et al [2] fizeram um estudo comparativo entre a técnica SVD e o modelo de rede neural autoencoder, para os conjuntos de dados Movielens 1M e 10M, e obtiveram melhores resultados, quanto aos valores de RMSE, com o autoencoder.

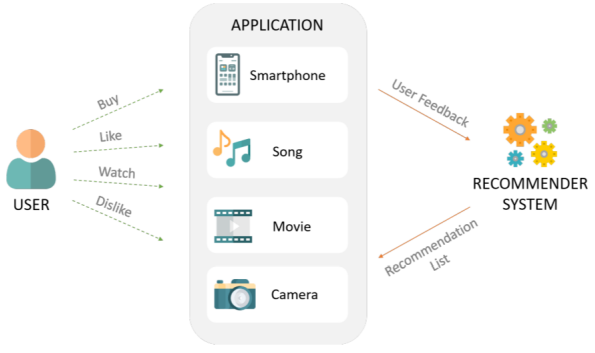

Figura 1. Prefêrências de Usuários e Sistemas de Recomendação[2]

Neste trabalho, implementamos uma rede neural do tipo autoencoder, utilizada para tarefas de filtragem colaborativa, com o objetivo de fornecer recomendações de filmes. Utilizamos para treinamento da rede a base de dados MovieLens [7] com, aproximadamente, 100 mil filmes e 610 usuários. Os filmes são avaliados na escala de 1 a 5 estrelas. A rede neural autoencoder "aprende"a codificar os valores de entrada em uma camada

intermediária de dimensão reduzida e a decodificá-los para a camada de saída.

Na seção 2 será apresentada uma breve Fundamentação Teórica sobre os sistemas de recomendação baseados em redes neurais do tipo autoencoders. Na seção 3, a Metologia adotada para e implementação em Matlab, na seção 4 a comparação dos Resultados encontrados e, por fim, na seção 5 as Conclusões sobre a aplicação do método no problema proposto.

II. Fundamentação Teórica

Os Sistemas de Recomendação fazem parte da área de pesquisa em mineração de dados e aprendizado de máquina. Segundo YANG [3], há diferentes formulações de mecanismos de recomendação: filtragem de conteúdo, filtragem colaborativa e técnicas híbridas.

A filtragem colaborativa (CF) é uma das abordagens mais eficientes. Baseada na coleta e análise uma grande quantidade de informações sobre os comportamentos, atividades ou preferências a fim de prever o que os usuários gostariam com base em sua semelhança com outros usuários.

Conforme Figura 2, a técnica de filtragem colaborativa subdivide-se em abordagens baseadas no conceito de vizinhança (ex: KNN - "k nearest neighborhood"), em modelos de redes neurais (ex: redes autoencoders) e híbridas.

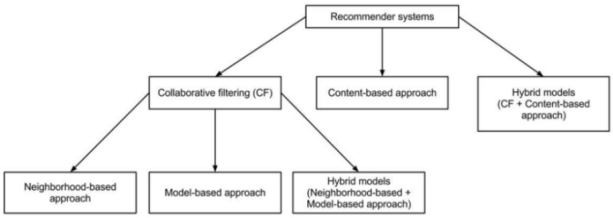

Figura 2. Técnicas de Recomendação e Filtragem Colaborativa [3]

A. Redes neurais artificiais do tipo Autoencoders

Na filtragem colaborativa baseada em classificação, temos m usuários, n itens e uma matriz **R** de classificação de itens do usuário parcialmente observada $R \in R^{mxn}$.

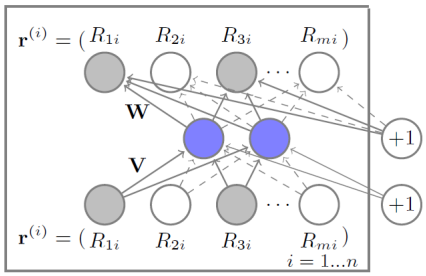

Figura 3. Estrutura de uma rede Autoencoder de uma única camada

A camada intermediária, de dimensão reduzida, da Figura 3 possui k elementos. Seja **V** é uma matriz $m \times k$ e **W** é uma matriz $n \times k$. A capacidade de se fatorar qualquer matriz rank-k nesta forma é um conceito fundamental da álgebra linear (fatoração de matrizes), e há um número infinito de tais fatorações correspondendo a vários conjuntos de vetores de base. A técnica de decomposição em valores singulares (SVD) é um exemplo de tal fatoração em que os vetores de base representados pelas colunas de **V** e **W** são ortogonais entre si.

Nas redes neurais artificiais do tipo autoencoder, a matriz **V** terá o papel de fazer a codificação ou redução de dimensionalidade de $(m, n) \rightarrow (m, k)$, enquanto que a matriz **W**, a decodificação de $(m, k) \rightarrow (m, n)$

O treinamento da rede autoencoder consiste em encontrar os valores de v_{is} e w_{js}, de tal forma que $R \approx VW^T$ aproxime-se ao máximo dos valores das classificações dos usuários atribuídas aos elementos de R. Para isso, será necessária a aplicação de uma técnica de otimização que minimize a função de custo J representada pelo erro quadrático médio entre a saída esperada y, expressa pelos elementos r_{ij} e a saída calculada $\hat{y}$ expressa pela equação 01.

$$\hat{r_{ij}} = \sum_{s=1}^{k} u_{is}.v_{js} \tag{1}$$

$$Min(J) = \frac{1}{2} \sum_{(i,j) \in S} e_{ij}^2 = \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \sum_{s=1}^{k} u_{is}.v_{js})^2 \tag{2}$$

B. SGD e Retropropagação

Neste trabalho utilizamos a técnica de otimização SGD (*Stochastic Gradient Descent*) para encontrar os valores dos pesos da rede, V_{is} e W_{js}, a partir do algoritmo de retropropagação do erro dado pela equação 4:

$$v_{iq} = v_{iq} - \alpha \frac{\delta J}{\delta v_{iq}} \qquad \forall q \in 1...k \tag{3}$$

$$w_{jq} = w_{jq} - \alpha \frac{\delta J}{\delta w_{jq}} \qquad \forall q \in 1...k \tag{4}$$

Conforme proposto por Sedhain [1], durante o processo de busca dos pontos de ótimos-locais, os valores dos pesos v_{iq} e w_{jq} mantenham-se os mais baixos possíveis, utiliza-se a técnica de regularização da rede que consiste na penalização desses termos na função de custo J por uma fator lambda λ, conforme equação 05 abaixo:

$$Min(J) = \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \sum_{s=1}^{k} u_{is}.v_{js})^2 + \lambda v_{iq}^2 + \lambda w_{jq}^2 \tag{5}$$

A partir das equações (4), (5) e (6), pode-se então demonstrar que:

$$v_{iq} = v_{iq}(1 - 2\alpha\lambda) + 2\alpha e_{ij} w_{jq} \qquad \forall q \in 1...k \tag{6}$$

$$w_{jq} = w_{jq}(1 - 2\alpha\lambda) + 2\alpha e_{ij} v_{iq} \qquad \forall q \in 1...k \tag{7}$$

III. Metodologia

Nesta seção serão descritos os procedimentos metodológicos necessários à solução do problema proposto. Serão apresentadas as ferramentas de software, o método utilizado para recomendação de filmes com base em autoencoders, a topologia da rede adotada, bem como os parâmetros de simulação.

A. Ferramentas computacionais

A principal ferramenta utilizada neste trabalho foi o *software* MATLAB, que apesar de possuir um conjunto de recursos para redes neurais, teve sua utilização limitada à implementação de um algoritmo iterativo, com operações lógicas e aritméticas, o qual será exposto na seção E.

B. Descrição do problema proposto

A base de dados "MovieLens", na sua versão "small", é constituída de 9742 filmes dos mais variados gêneros. Em um universo de 610 usuários, cada indivíduo atribuiu notas de 1 a 5 para os filmes que já assistiu, sendo 1 considerada uma nota ruim e 5 uma nota excelente. O volume de avaliações aproxima-se de 100.000, por este motivo, esta base de dados também pode ser identificada como "MovieLens 100k".

Desta maneira, para este sistema considera-se como entrada do algoritmo as diversas avaliações individuais dos usuários, que ao serem submetidas à rede produzem o efeito de filtragem colaborativa, na qual as avaliações particulares influenciam e são influenciadas de acordo com a similaridade entre as preferências de cada indivíduo.

Portanto, o modelo deverá ter como saída uma lista de filmes para cada usuário, que em um pós-processamento será ordenada, representando os filmes de maior relevância, de acordo com o modelo. Para este processo de aprendizado de máquina será avaliado o erro médio quadrático entre os filmes que o usuário já atribuiu nota em relação às previsões do algoritmo. Além disso, será observada a norma de Frobenius da matriz de erros e a taxa de convergência do modelo.

C. Análise dos dados

Tendo em vista que a matriz de avaliações é esparsa, foi realizada uma etapa de análise dos dados disponíveis. Para esta análise foram estudadas as seguintes métricas:

- Total de avaliações feitas por cada usuário
- Total de pontos acumulados por cada filme

A partir da análise dos gráficos das figuras 4 e 5, foi possível notar que:

- Alguns usuários avaliaram uma quantidade expressiva de filmes frente aos demais.
- De toda a base de filmes, ordenada em número crescente de índices, a distribuição de pontos foi mais densa em índices menores, ou, decresceu à medida que os índices aumentaram.

Na figura 4 os usuários estão dispostos radialmente. Do gráfico, nota-se que o usuário 414 tem grande contribuição sobre o universo de avaliações.

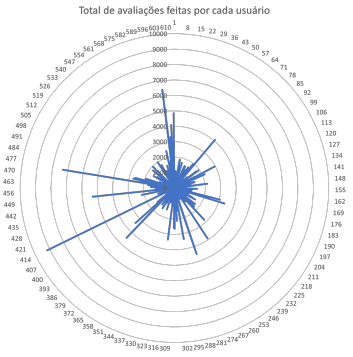

Figura 4. Total de avaliações feitas por cada usuário

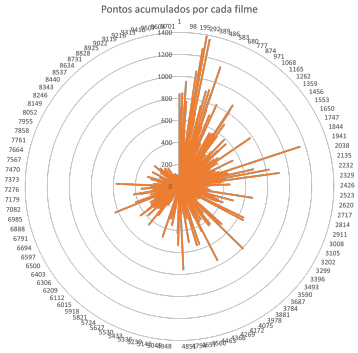

Figura 5. Pontos acumulados por cada filme

D. Particionamento dos dados em conjuntos de treino e teste

Como estratégia de particionamento dos dados, os primeiros 7742 filmes foram reservados ao treinamento da rede, o que corresponde a cerca de 80% dos filmes disponíveis. Os 2000 filmes restantes foram reservados à etapa de teste, correspondendo a cerca de 20% da base MovieLens.

Usuários	Filme 1	Filme 2	Filme 3	...	Filme 9741	Filme 9742
1	4	3	1	...	4	1
2	...	...	...	...	3	2
...	...	...	...	...	4	3
610	5	2	2	2	4	3

80% 20%

Conjunto de treino
Conjunto de teste

Figura 6. Particionamento dos conjuntos de treino e teste

Estando os conjuntos definidos, o treinamento da rede pôde ser realizado. Desta maneira, uma vez atingido o critério de parada, realizou-se a aferição do erro médio quadrático dos conjuntos de teste e treinamento com respeito à matriz de

avaliações originais, com isto, o progresso do algoritmo pôde ser acompanhado a cada época.

E. Implementação do modelo

Para a construção do modelo de aprendizado de máquina, foi desenvolvido um script na linguagem MATLAB, tendo o suporte das funções *xlsread* e *readtable* para a leitura do dados, que encontravam-se em tabelas.

Nesta etapa, utilizou-se como modelo o pseudocódigo proposto por AGGARWAL [5], que implementa o algoritmo do Gradiente Descendente Estocástico com retropropagação e regularização dos pesos **V** e **W**, tal como ilustrado nas equações 6 e 7 da seção II.

Este algoritmo pode ser visto na figura 7. Conforme destacado pelo autor, é de extrema importância inicializar os pesos de **V** e **W** com valores randômicos baixos, entre [-1, 1], pois impactam diretamente na qualidade dos resultados obtidos pelo modelo.

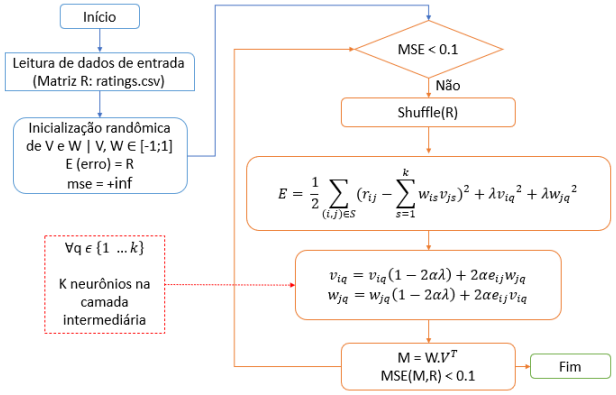

Figura 7. Algoritmo do SGD com retropropagação e regularização

Como parte da definição da arquitetura da rede, foram realizados experimentos com quantidades de neurônios na camada intermediária variando de 50 a 100. Além do mais, a taxa de aprendizado α e regularização λ também foram parâmetros obtidos experimentalmente.

IV. Resultados

Estando estabelecido o algoritmo a ser utilizado, foram realizados experimentos com o intuito de definir os hiperparâmetros do modelo, tais como: o número de neurônios na camada intermediária, a taxa de aprendizado α e a regularização λ. A tabela I contêm os resultados para os 3 experimentos realizados ao serem utilizados 50, 75 e 100 neurônios na camada intermediária.

Os resultados da tabela I foram coletados para uma taxa de aprendizado de 0,002 e um fator de regularização de 0,001. A partir dos resultados obtidos, foi possível observar decréscimo no número de épocas e consequentemente tempo de simulação à medida que o número de neurônios aumentou. Para todas as simulações adotou-se um erro médio quadrático de 0,1 como critério de parada. Para todos os experimentos a norma de Frobenius finalizou em torno de 100 no momento em que o

Tabela I
Primeira definição de parâmetros

	Exper. 1	**Exper. 2**	**Exper. 3**
Neurônios na camada intermediária	50	75	100
Tempo de simulação (s)	93	82	77
Número de épocas	63	43	36
Norma de Frobenius	101,2	101,48	100,52
MSE (treino)	0,098	0,098	0,094
MSE (teste)	0,040	0,050	0,050

Valores obtidos em computador Intel(R) i5 10th CPU @ 1.60GHz

critério de parada foi atingido. Desta maneira, para as demais análises, padronizou-se como 100 o número de neurônios na camada intermediária, de forma que nesta configuração espera-se que em torno de 36 épocas o critério de parada seja atingido.

Para cada experimento foram traçados os gráficos que relacionam o número de épocas com o erro médio quadrático dos conjuntos de treino e teste, bem como a norma de Frobenius. Através da análise do gráfico, é possível verificar a dinâmica do algoritmo do gradiente descendente, bem como a relação direta entre a norma de Frobenius e o erro médio quadrático dos conjuntos avaliados.

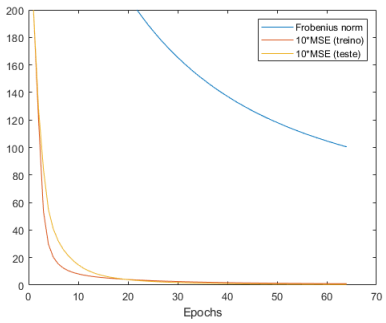

Figura 8. 50 Neurônios na camada intermediária

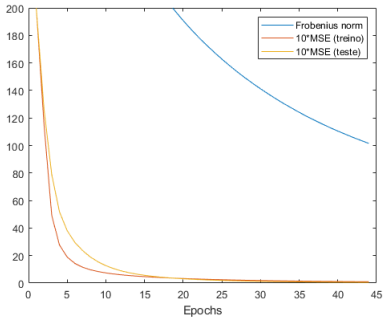

Figura 9. 75 Neurônios na camada intermediária

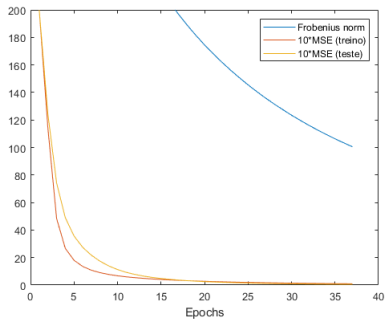

Figura 10. 100 Neurônios na camada intermediária

Ao final do processo de treinamento das matrizes de pesos **V** e **W**, sendo este treinamento condicionado ao algoritmo do gradiente descendente estocástico, obtêm-se uma matriz **M**, que diferente da matriz **R** de entrada, possui observações para todos os filmes do conjunto de dados de entrada. Desta forma, é possível, através de um sistema de ranking, recomendar filmes para determinado usuário, com exceção daqueles já assistidos. A representação gráfica do processo descrito acima encontra-se na figura 11.

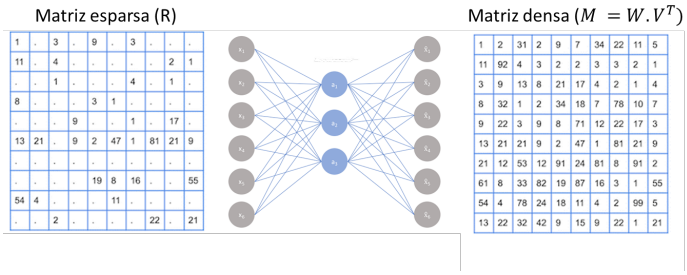

Figura 11. Representação do processo de autoencoder

De posse da matriz de recomendações, foram selecionados alguns usuários. Para estes usuários, foi efetuada busca na matriz de recomendações de forma a extrair os 5 primeiros filmes recomendados pelo algoritmo. A figura 12 ilustra o resultado desta busca e exemplifica o mecanismo de recomendação elaborado neste trabalho.

V. Conclusões

A implementação da rede neural com topologia de autoencoder mostrou-se eficaz na resolução do problema de sistema de recomendação. Dentre os diferenciais deste tipo de rede, está o fato de permitir conjuntos de dados esparsos, isto é, nem todas as características estão presentes em cada observação, que para o problema em questão, materializa-se no fato de que dificilmente algum usuário atribuiria um conceito para todos os 9742 filmes disponíveis na base de dados. Além da facilidade de implementação do algoritmo, notou-se expressiva velocidade no processamento dos dados, alcançando cerca de 1 minuto na etapa de treinamento, tendo em vista que 100 neurônios foram utilizados na camada intermediária da rede. Deve-se ressaltar que há grande sensibilidade na escolha das

usuário	1°	2°	3°	4°	5°	MSE
40	Seven Samurai (Shichinin no samurai) (1954)	Out of Sight (1998)	Fight Club (1999)	Animal House (1978)	Inglourious Basterds (2009)	0,084
92	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	American Pie (1999)	Stranger than Fiction (2006)	Shine (1996)	Three Kings (1999)	0,074
123	Man Bites Dog (Cest arrivÃ© prÃ¨s de chez vous) (1992)	Hours, The (2002)	Streetcar Named Desire, A (1951)	WALLÂ·E (2008)	Green Mile, The (1999)	0,061
245	Clerks (1994)	Contact (1997)	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Airplane! (1980)	Truman Show, The (1998)	0,159
312	Godfather, The (1972)	Jungle Book, The (1967)	Unforgiven (1992)	City of God (Cidade de Deus) (2002)	Pulp Fiction (1994)	0,110
460	Terminator, The (1984)	Princess Bride, The (1987)	Dark Knight, The (2008)	Casablanca (1942)	Schindlers List (1993)	0,079
514	Wallace & Gromit: The Best of Aardman Animation (1996)	2001: A Space Odyssey (1968)	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	Heat (1995)	Lady Eve, The (1941)	0,088
590	Man Bites Dog (Cest arrivÃ© prÃ¨s de chez vous) (1992)	Army of Darkness (1993)	Dersu Uzala (1975)	In the Name of the Father (1993)	Manon of the Spring (Manon des sources) (1986)	0,115
MSE AVG						0,096

Figura 12. Recomendação de filmes para usuários selecionados

condições iniciais do sistema, visto que taxas de aprendizado superiores a 0,1 implicavam na falta de convergência do modelo. Neste trabalho não utilizou-se taxa de aprendizado adaptativa dada à sensibilidade no ajuste dos parâmetros. Além da taxa de aprendizado, constatou-se que as matrizes de pesos, denominadas **V** e **W**, impactam diretamente na qualidade do treinamento caso não sejam inicializadas com valores aleatórios na faixa de -1 a 1, mais precisamente obteve-se desempenho satisfatório ao utilizar a faixa de -0,25 a 0,25.Por fim, um erro quadrático médio de 0,096 foi obtido para os usuários selecionados(figura 12), o que confere robustez à rede treinada.

Referências

[1] SEDHAIN, S., MENON, A. K., SANNER, S., XIE, L. ”AutoRec: Autoencoders Meet Collaborative Filtering”. WWW’15 Companion: Proceedings of the 24th International Conference on World Wide Web, pages 111-112, May 2015.

[2] FERREIRA, D., SILVA, S., ABELHA, A., MACHADO, J. ”Recommendation Systems Usiing Autoencoders”, Applied Science, 2020, 10, 5510; doi:10.3390/app10165510 www.mdpi.com/journal/applsci.

[3] YANG, S. C., LIU, Z. ”Online News Recommender based on Stacked Auto-Encoder”. IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017.

[4] BILLSUS, D., PAZZANI, M. ”Learning Collaborative Information Filters”Technical Report WS98-09, 1998.

[5] AGGARWAL, C. ”Recommender Systems”. Springer International Publishing Switzerland, 2016.

[6] https://www.comparitech.com/blog/vpn-privacy/netflix-statistics-facts-figures/

[7] https://grouplens.org/datasets/movielens/

ANEXO I

```
% Projeto 1: Sistema de Recomendacao utilizando Redes Neurais do tipo AutoEncoder
% Disciplina: Topicos Avancados de Aprendizado de Maquina
% Prof.: Cicero Costa
% Alunos: Carlos Fonseca e Willian Guerreiro
% Data: 07.09.2021

    close all;  % Fecha todas as figuras abertas
    clear all;  % Limpa as variaveis do WorkSpace
    clc         % Limpa a janela de comandos

    disp('Loading dataset..')
    Mov = readtable("Movies.xlsx");      % Arquivo com o indice de Filmes e Titulos
    T = xlsread("Ratings.xlsx");         % Arquivo com as avaliacoes dos filmes pelos usuarios (1 a 5 estrelas)

    NoUsers = 610;                       % Qtde de usuarios na base Movie.Lens 100k
    NoMovies = 9742;                     % Qtde de filmes na base Movie.Lens 100k
    NoRatings = 100836;                  % Qtde de avaliacoes na base Movie.Lens 100k

    NoHiddenLayers = 100;                % Qtde de neuronios na camada intermediaria da rede


    R = zeros(NoUsers,NoMovies);         % R -> Matriz de {usuarios, filmes}
    U = ones(NoUsers,  NoHiddenLayers); % U:  m k input-to-hidden layer (encoder)
    V = ones(NoMovies, NoHiddenLayers); % V:  nxk hidden layer-to-output matrix (decoder)

    % R    a matriz com a tripla: usu rios, filmes, ratings
    for j = 1:NoRatings
        R(T(j,1),T(j,2)) = T(j,4);
    end

    for i=1:NoUsers
        for j=1:NoMovies
            if j > 7742
                R_teste(i,j) = R(i,j);
            else
                R_train(i,j) = R(i,j);
            end
        end
    end


    U = -0.25 + 0.5*rand(size(U));
    V = -0.25 + 0.5*rand(size(V));

    %U = rand(size(U));
    %V = rand(size(V));


    U_plus = U;
    V_plus = V;


    alpha = 0.002;
    lambda = 0.001;


    e = R_train;
    frobenius_norm = 200;
    frobenius_array = [frobenius_norm];

    mse_value = 200;
    mse_array =[mse_value];

    mse_teste_value = 200;
    mse_teste_array = [mse_teste_value];

    h1 = plot(frobenius_array,'YDataSource','frobenius_array','DisplayName','Frobenius norm');
    ylim([0 200]);
    xlabel('Epochs');
    hold on;
    h2 = plot(mse_array,'YDataSource','mse_array','DisplayName','10*MSE (treino)');
    ylim([0 200]);
    legend;
    h3 = plot(mse_teste_array,'YDataSource','mse_teste_array','DisplayName','10*MSE (teste)');
    ylim([0 200]);
    legend;

    disp('Starting ...');
    tic;
    epochs = 0;
    while (mse_value > 0.1)
        epochs = epochs + 1;
        i_shuffled = randperm(NoUsers);
        for i = 1:NoUsers
            j_shuffled = randperm(NoMovies);
            for j = 1: NoMovies
                soma = 0.0;
                somal = 0.0;
                if R(i_shuffled(i), j_shuffled(j)) ~= 0

                    for q = 1:NoHiddenLayers
                        soma = soma + U(i_shuffled(i), q)*V(j_shuffled(j), q);
                        somal = somal + lambda*U(i_shuffled(i), q)^2 + lambda*V(j_shuffled(j), q)^2;
                    end

                    e(i_shuffled(i), j_shuffled(j)) = (R(i_shuffled(i), j_shuffled(j)) - soma) + somal;
                    %e(i_shuffled(i), j_shuffled(j)) = R(i_shuffled(i), j_shuffled(j)) - soma;

                    U_plus(i_shuffled(i),:) = U(i_shuffled(i),:)*(1-2*alpha*lambda) + 2*alpha*e(i_shuffled(i),j_shuffled(j))*V(j_shuffled(j),:);
                    V_plus(j_shuffled(j),:) = V(j_shuffled(j),:)*(1-2*alpha*lambda) + 2*alpha*e(i_shuffled(i),j_shuffled(j))*U(i_shuffled(i),:);
                    U(i_shuffled(i),:) = U_plus(i_shuffled(i),:);
                    V(j_shuffled(j),:) = V_plus(j_shuffled(j),:);

                end

            end
```

```
    end

    %%%%%%%%%%%%%%%%%%%% C lculo de M e plot de gr ficos %%%%%%%%%%%%%%%%%%%%
    M = U * V';
    M_teste = U * V';

    for i = 1:NoUsers
        for j = 1: NoMovies
            if R_train(i,j) == 0
                M(i,j) = 0;
            end
        end
    end

    for i = 1:NoUsers
        for j = 1: NoMovies
            if R_teste(i,j) == 0
                M_teste(i,j) = 0;
            end
        end
    end

    mse_value = sum(sum((M - R_train).^2))/nnz(R_train);
    mse_teste = sum(sum((M_teste - R_teste).^2))/nnz(R_teste);

    mse_array = [mse_array ; 10*mse_value];
    disp(['mse ',num2str(mse_value)])

    mse_teste_array = [mse_teste_array ; 10*mse_teste];
    disp(['mse teste ',num2str(mse_teste)])

    frobenius_norm = norm(e,'fro');
    frobenius_array = [frobenius_array ; frobenius_norm];
    disp(['Frobenius norm ',num2str(frobenius_norm), newline]);

    refreshdata;
    drawnow;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end


%%% 2a. Parte: Recomenda o de Filmes para usu rios selecionados:

R_base = U * V';
Tabela_Rec = zeros(8,6);
Tabela_Rank = zeros(8,6);
Users = [40, 92, 123, 245, 312, 460, 514, 590];

for u = 1:8

 contador = 0;
 for j = 1:NoMovies
    if R_train(Users(u),j) > 0
        R_base(Users(u),j) = 0; % despreza as avalia  es originais dos filmes feitas pelo usu rio "u"
        contador = contador + 1;
    end
 end

 [Rec, I] = sort(R_base(Users(u),:), 'descend');

 for col = 1:5
    Tabela_Rec(u, col)  = I(col);
    Tabela_Rank(u, col) = R_base(Users(u),I(col));
 end


    Tabela_Rec(u, 6)  =  sum((M(Users(u), :) - R_train(Users(u), :)).^2)/contador;

end

%%%

elapsed_time = toc;
disp(['Elapsed Time: ',num2str(elapsed_time)]);
disp(['Epochs : ',num2str(epochs)]);

tabela_movies = cell(8,6,1);
for i=1:8
    for j=1:6
        if j~= 6
            tabela_movies{i,j} = table2array(Mov(Tabela_Rec(i,j),3));
        else
            tabela_movies{i,j} = Tabela_Rec(i,j);
        end
    end
end


movies_scores = zeros(8,5);

for i=1:8
    for j=1:5
        movie_index = Tabela_Rec(i,j);
        movies_scores(i,j) = sum(R(:,movie_index));
    end
end

path = strcat(datestr(datetime(),'mmmm-dd-yyyy HH-MM-SS'),'.xls');
path = strcat('results2/','tabela_rec_',path);
xlswrite(path,Tabela_Rec);

path = strcat(datestr(datetime(),'mmmm-dd-yyyy HH-MM-SS'),'.xls');
path = strcat('results3/','movie_scores_',path);
xlswrite(path,movies_scores);
```