# Desempenho de uma Rede de Propagação Direta no Reconhecimento de Faces reduzidas pelos métodos 2D2PCA e 2D2LDA – 2021/1*

*Nota: Projeto da discplina Aprendizado de Máquina do 1o. 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 reconhecimento facial é uma aplicação clássica de Visão Computacional e, atualmente, bastante utilizada na identificação biométrica e controle de acesso. Neste trabalho, implementamos as técnicas 2D2PCA (análise de componentes principais bidirecional) e 2D2LDA (Analise de Discriminantes Lineares Bidirecional) para reduzir a dimensionalidade de 400 imagens de face da base de dados ATT *"The Database of Faces"*. Comparamos os resultados obtidos no reconhecimento, primeiramente, utilizando o método da distância euclidiana. Em seguida, desenvolvemos uma rede de propagação direta do tipo perceptron multicamadas, utilizando 70% dos dados para o treinamento da rede e os demais 30% como conjunto de testes. Avaliamos o desempenho da rede com diferentes topologias, variando o número de camadas, quantidade de neurônios por camada, funções de ativação e métodos de otimização. O melhor resultado alcançado após a otimização dos parâmetros foi de 94% de acerto. O algoritmo foi implementado em MATLAB.**



***Index Terms*—perpectron multicamadas, análise de componentes principais, análise de discriminantes lineares**


## I. Introdução

O uso do Reconhecimento Facial como forma de autenticação biométrica disseminou-se, nos últimos anos, com a incorporação dessa funcionalidade nos *smartphones*, não só para o desbloqueio de tela como, também, para a autenticação de transações, eliminando-se a necessidade de digitação de senhas. Adicionalmente, em projetos de cidades inteligentes, câmeras de segurança com biometria estão sendo utilizadas para o rastreamento, identificação e controle de acesso de pessoas.

A capacidade humana de reconhecer indivíduos apenas observando o rosto humano é bastante notável. Esta capacidade persiste mesmo com o passar do tempo, mudanças em aparência e oclusão parcial do rosto, como é possível observar durante os tempos de pandemia do corona vírus em que as pessoas, mesmo utilizando máscaras de proteção, podem ser facilmente reconhecidas. Esse fato sugere que existem características determinantes que descrevem as imagens do rosto das pessoas e que, matematicamente, podem ser representadas pelo seu conjunto de autovalores e autovetores.

Como exemplo, Shereena V.B e Julie M. D.[1] fizeram um estudo comparativo dos métodos de redução de dimensionalidade, PCA e LDA (Análise de Componentes Principais e Análise Discriminante Linear) que visam revelar estruturas significativas e relacionamentos em dados multivariados. As representações dessas imagens em formato reduzido a partir desse métodos foram aplicadas ao problema de reconhecimento facial utilizando redes neurais com retropropagação. Comparando os resultados obtidos, as autoras concluiram que o método PCA tende a superar o LDA em quase todos casos e, portanto, PCA pode ser adotado como uma ferramenta eficaz para a redução de dimensão.

Figura 1. Base de dados de faces AT&T

De forma análoga, neste trabalho, avaliamos as duas técnicas de redução de dimensionalidade, 2D2PCA e 2D2LDA (bidirecionais), e aplicamos as imagens reduzidas em uma rede de propagação direta do tipo perceptron multicamadas a fim de avaliar seu desempenho no reconhecimento de faces. Para isso, utilizamos a base de dados da AT&T, também conhecida por ORL (*"Our Database of Faces"*), que contém 400 imagens de 40 indivíduos distintos. As 10 imagens de cada pessoa foram tiradas em momentos diferentes, variando a iluminação, expressões faciais (olhos abertos / fechados, sorrindo / não sorrindo) e detalhes faciais (com óculos / sem

óculos), ilustrada na fig 1.

Na seção 2 será apresentada uma breve Fundamentação Teórica sobre os métodos de redução dimensional e rede neural utilizados. 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

Quando estamos trabalhando com imagens, precisamos transformá-las do espaço bidimensional $X(m,n)$ (2D) para vetores de uma dimensão $x(m*n)$, a fim de compatibilizá-las com o formato da camada de entrada em uma topologia de redes neurais artificiais. Entretanto, segundo Yang et al (2004)[2], para imagens de alta resolução, tais vetores com $m*n$ posições aumentam a complexidade do modelo exigindo o treinamento de uma enorme quantidade de pesos $w_{i,j}$ associados a essas características de entrada, o que pode se tornar inviável do ponto de vista computacional, dependendo dos recursos disponíveis.

A fim de otimizar esse processo, utilizam-se técnicas para reduzir o tamanho desses vetores de entrada, preservando as características principais da imagem, dentre elas:

### A. Análise de Componentes Principais Bidirecional

Esta técnica também conhecida como 2D2PCA realiza a redução de dimensão nas duas direções, horizontal e vertical. Consiste em encontrar a matriz de covariâncias $G_H$ e $G_V$, a partir do conjunto de $N$ imagens $A_i$, expressas por:

$$G_H = \frac{1}{N}\sum_{i=1}^{N}(A_i - \bar{A})^T(A_i - \bar{A}) \tag{1}$$

$$G_V = \frac{1}{N}\sum_{i=1}^{N}(A_i - A)(A_i - A)^T \tag{2}$$

Calcular as matrizes de dispersão $U$ e $V$, formadas por:

$U$: autovetores que correspondem aos $q$ maiores autovalores da matriz de covariância $G_H$, realiza a projeção da imagem original reduzindo a dimensão na direção vertical.

$V$: autovetores que correspondem aos $d$ maiores autovalores da matriz de covariância $G_V$, que servirão para realizar a projeção da imagem original, reduzindo a dimensão na direção vertical.

E, por fim, obter as imagens projetadas (reduzidas) $A_p$ com dimensional d x q

$$A_p = V^T AU \tag{3}$$

### B. Análise de Discriminantes Lineares Bidirecional

A Análise Discriminante Linear é, também, bastante utilizada para a redução de dimensionalidade na etapa de pré-processamento de aplicações de reconhecimento de padrões e aprendizado de máquina. A seleção de características no LDA é obtida maximizando a diferença entre as classes dada pela matriz de covariância $S_b$ e minimizando a distância intraclasses dada pela matriz de covariâcia $S_w$. Segundo SOUZA[4], a técnica visa encontrar os vetores que melhor discriminem as classes.

Assumindo que existam $C$ classes de padrões conhecidos no conjunto de treinamento $A$ de tamanho $N$, as matrizes de dispersão $S_b$ e $S_w$, nas duas direções (horizontal e vertical) são dadas por:

$$S_{bH} = \sum_{i=1}^{C} N_i(A_i - \bar{A})^T(A_i - \bar{A}) \tag{4}$$

$$S_{wH} = \sum_{i=1}^{C}\sum_{j=1}^{Ni}(\bar{A}_j - \bar{A}_i)^T(\bar{A}_j - \bar{A}_i) \tag{5}$$

$U$: projeção horizontal obtida pelos autovetores que correspondem aos maiores autovalores de $S_{wH}^{-1}S_{bH}$

$$S_{bV} = \sum_{i=1}^{C} N_i(A_i - \bar{A})(A_i - \bar{A})^T \tag{6}$$

$$S_{wV} = \sum_{i=1}^{C}\sum_{j=1}^{Ni}(\bar{A}_j - \bar{A}_i)^T(\bar{A}_j - \bar{A}_i)^T \tag{7}$$

$V$: projeção vertical obtida pelos autovetores que correspondem aos maiores autovalores de $S_{wV}^{-1}S_{bV}$

De forma análoga à técnica 2D2PCA, a matriz $A_p$ correspondente às imagens $A$ com dimensão reduzida é obtida pela equação (3).

### C. Similaridade pela Distância Euclidiana

Sendo $B_1$ é o conjunto de imagens de testes com dimensionalidade $d$ reduzida por meio das técnicas 2D2PCA ou 2D2LDA, e $B_2$ o conjunto de imagens reduzidas e utilizadas no conjunto de treinamento. Segundo WANG X. [2], podemos definir uma medida de similaridade entre um elemento $B_1(i) = [Y_1^{(i)}, Y_2^{(i)}, ..Y_d^{(i)}]$ e qualquer elemento de $B_2(j) = [Y_1^{(j)}, Y_2^{(j)}..Y_d^{(j)}]$ por meio da distância euclidiana entre os vetores dessas 02 matrizes, dada pela equação (8). Nesse caso, o padrão reconhecido é aquele mais próximo, ou seja, o de menor distância euclidiana. Para o caso de imagens reduzidas pelas técnicas bidirecionais, propusemos, neste trabalho, uma variação do método de cálculo da distância euclidiana (eq.9), calculando-a a partir da soma da menor distância euclidiana entre os vetores projetados nas 02 dimensões, horizontal(distância entre os vetores-coluna $Y$ de $B_1$ e $B_2$) e vertical(distância entre os vetores-linha $X$ de $B_1$ e $B_2$).

$$Dist(B_1, B_2) = \sum_{k=1}^{d}\sqrt{(Y_i^k)^T(Y_j^k)} \tag{8}$$

$$Dist(B_1, B_2) = \sum_{k=1}^{d} Min\left(\sqrt{(Y_i^k)^T(Y_j^k)}, \sqrt{(X_i^k)(X_j^k)^T}\right) \tag{9}$$

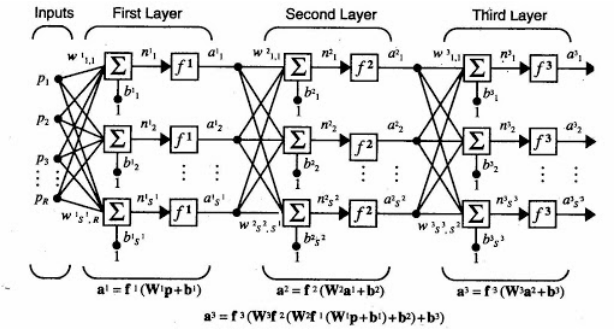


Figura 2. Perceptron Multicamadas

### D. Rede de Propagação Direta Perceptron Multicamadas

As redes de propagação direta do tipo perceptron multicamadas, quando utilizadas para o reconhecimento de padrões, são organizados na forma de camadas conectadas, conforme ilustrado na figura 2., sendo:

Camada de Entrada: recebe a imagem ou matriz de características com dimensão reduzida convertida para o espaço de uma dimensão (vetor).

Camadas Escondidas sensoriais: uma ou mais camadas intermediárias formadas por um conjunto de neurônios interligados à camada anterior.

Camada de saída: última camada com a dimensão correspondente ao número de classes do problema proposto. No caso da aplicação de reconhecimento facial, as classes correspondem aos diferentes indivíduos do conjunto de treinamento.

## 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 classificação com base em distância, a topologia da rede para a abordagem de redes neurais, bem como os parâmetros de simulação.

### A. Ferramentas computacionais

A principal ferramenta utilizada neste trabalho foi o *Deep Learning Toolbox* do *software* MATLAB, que apesar de possuir um conjunto de ferramentas para redes neurais profundas, também dispõe de recursos para o projeto de redes de propagação direta mais simples, a exemplo da *Patternet*.

1) Patternet: A função patternet especifica uma rede neural com foco em reconhecimento de padrões, sendo parametrizada com o número de camadas escondidas, função de custo e função de performance, sendo definida como *crossentropy* como padrão.

2) Trainbr: Esta função realiza a atualização dos pesos e polarizações de acordo com o método de Levenberg-Marquadt com o intuito de minimizar o erro quadrático. O objetivo final é produzir uma rede que generalize bem. Todo o processo é dito regularização Bayesiana.

3) Perform: A função *perform* calcula o desempenho da rede de acordo com a função de perfomance previamente definida, onde geralmente é analisada a acurácia de classificação. Esta função é parametrizada com os dados de entrada e a resposta esperada.

4) Confusionchart: Esta função realiza o *plot* da matriz de confusão. Nesta matriz as linhas correspondem aos dados corretos e as colunas correspondem aos dados classificados pelo modelo.

### B. Descrição do problema proposto

A base de dados "ORL" é constituída por 40 pastas, sendo cada pasta um conjunto de dados isolado contendo 10 imagens da face de uma pessoa, sob diferentes condições de iluminação e posicionamento. Cada imagem possui originalmente as dimensões 112x92 pixels. Em virtude desta quantidade expressiva de dados por observação, foram implementadas as técnicas de redução de dimensionalidade abordadas nas seções A e B, cujo efeito principal foi proporcionar a redução para 100 características por observação.

Desta maneira, os 100 atributos de cada observação serão utilizados como a primeira camada de uma Rede Neural Artifical tipo Perceptron Multicamadas. Com este modelo objetiva-se classificar corretamente um indivíduo dentre 40 de um banco de dados pré-estabelecido, sendo essa classificação realizada por meio da fotografia de sua face. O modelo será treinado com 7 faces de cada indivíduo e testado com as 3 faces remanescentes. Para este processo de aprendizado de máquina serão avaliadas a taxa de convergência, a matriz de confusão e a acurácia do modelo.

### C. Implementação do modelo

A implementação deste trabalho foi realizada através do desenvolvimento de um script na linguagem MATLAB, tendo o suporte das ferramentas de aprendizado de máquina disponibilizadas pelo *software*.

Inicialmente foi realizada a leitura do conjunto de dados, seguida do seu particionamento na razão 7:3, ou seja, para cada pasta 7 elementos constituem o conjunto de treino e 3 elementos constituem o conjunto de teste, totalizando 280 elementos no conjunto de treino e 120 elementos no conjunto de teste. Em seguida, iniciou-se o processo de redução de dimensionalidade, implementado de maneira a possibilitar o uso de 2D2LDA ou 2D2PCA, para fins de análises futuras.

Para a realização do fluxo de aprendizado de máquina, foram utilizados alguns parâmetros de simulação, a citar:

- Definição de cada Neurônio: Na ferramenta MATLAB esta funcionalidade foi implementada pela função **net**. Para esta aplicação definiu-se a topologia 100x20x40, sendo utilizada a função de ativação *ReLU* na camada intermediária e *Softmax* na última camada.
- Definição da função de otimização: Neste trabalho foram realizados testes com o gradiente descente simples ('trainscg') e gradiente conjugado('traincgf'), passados como parâmetro na função *patternet*.
- Definição da função de perda: O valor padrão 'crossentropy' foi mantido na função *patternet*.

- Treinamento do modelo: Realizado pela função **trainbr**, parametrizada com a rede e os conjuntos de treinamento.
- Condição de parada: Módulo do gradiente descendente menor que $10^{-7}$.
- Teste do modelo e obtenção de métricas.

Ao final da parametrização, espera-se uma topologia de rede tal como ilustra a figura 3.

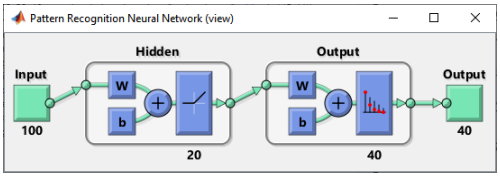


Figura 3. Topologia da rede

### D. Mínima Distância Euclidiana Horizontal e Vertical

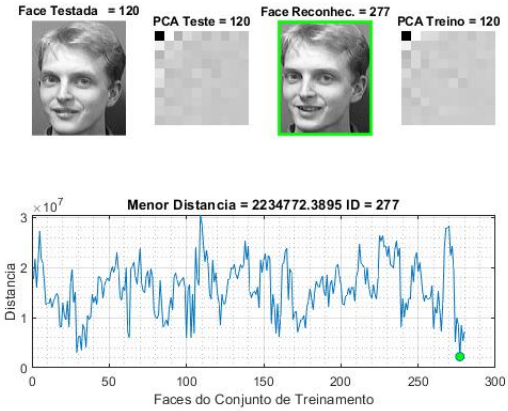


Figura 4. Base de dados de faces AT&T

Para a análise de similaridade entre as imagens do conjunto de teste e do conjunto de treinamento, foi implementado em MATLAB o código correspondente às equações 8 e 9 a fim de comparar, entre si, o índice de acertos no reconhecimento das faces. Na figura 4 é ilustrada a interface desenvolvida, mostrando na parte superior a imagem original e a imagem reduzida $B_1(i)$ pela técnica 2D2PCA do conjunto de teste que está sendo procurada dentre as 280 imagens reduzidas $B_2(j)$ do conjunto de treinamento. Foi traçado, na interface, um gráfico mostrando o valor de $Dist(B_1(i), B_2(j))$ para $i = [1:120]$ e $j = [1:280]$ em que o valor mínimo é assinalado e atribuído como a face encontrada.

### E. Diagrama em blocos do sistema

A tarefa de identificação dos indivíduos com base em suas fotografias de rosto, pôde ser implementada a partir do viés das redes neurais artificiais e sob o viés da distância euclidiana entre dois vetores, considerando a distância mínima tomada no conjunto de teste, tal como exposto nas equações 8 e 9. Desta maneira, nota-se que o procedimento difere apenas na etapa de classificação, e estes dois fluxos podem ser comparados através dos diagramas das figuras 5 e 6.

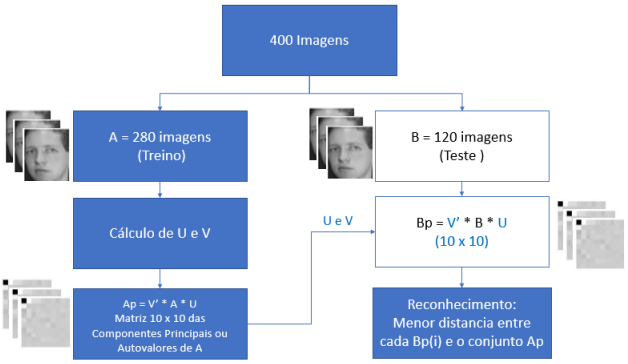


Figura 5. Método baseado na distância euclidiana

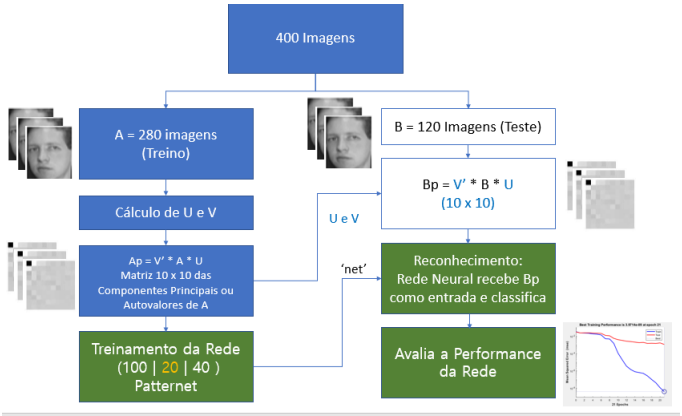


Figura 6. Método baseado em Redes Neurais

## IV. Resultados

O processo de obtenção de resultados compreendeu os experimentos realizados com diferentes quantidades de camadas intermediárias, utilização de 2D2LDA ou 2D2PCA, e função de otimização sendo Gradiente Descendente ('trainsgd') ou Gradiente Conjugado ('trainsgf').

Com o intuito de estabelecer o método de redução de dimensionalidade e a função de otimização, fixou-se o número de camadas intermediárias em 1 camada de 10 neurônios, e variou-se os demais parâmetros.

Tabela I
PRIMEIRA DEFINIÇÃO DE PARÂMETROS

| Parâmetro | Exper. 1 | Exper. 2 | Exper. 3 | Exper. 4 |
|---|---|---|---|---|
| 2D2LDA | 1 | 0 | 1 | 0 |
| 2D2PCA | 0 | 1 | 0 | 1 |
| TRAINSGD | 1 | 1 | 0 | 0 |
| TRAINSGF | 0 | 0 | 1 | 1 |
| Acurácia | 94,16% | 81,67% | 91,66% | 85% |

1 - Opção utilizada  0 - Opção não utilizada

Tendo em vista que a acurácia de classificação foi determinante para a obtenção dos demais parâmetros, foi possível fixar o método de redução de dimensionalidade como o 2D2LDA, bem como a função de otimização como sendo o gradiente descendente, com base na tabela I.

Os experimentos realizados apontaram melhores desempenhos para apenas 1 camada escondida, sendo mais eficaz a utilização de 20 neurônios nesta camada. A tabela II realiza uma comparação entre o modelo desenvolvido com redes neurais e o método da distância Euclidiana.

Tabela II
SEGUNDA DEFINIÇÃO DE PARÂMETROS

| Método | Acurácia |
|---|---|
| Rede neural 100x20x40 | 94% |
| Distância Euclidiana | 98,33% |

Estando a topologia da rede definida, foi possível avaliar os gráficos de convergência da rede considerando a função de gradiente descendente e gradiente conjugado, cujos respectivos gráficos estão expostos nas figuras 7 e 8.

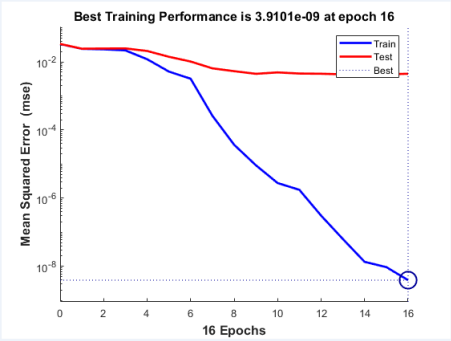


Figura 7. Taxa de convergência do gradiente descendente

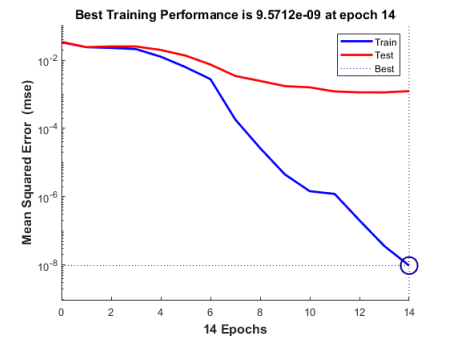


Figura 8. Taxa de convergência do gradiente conjugado

Como medida final de avaliação do modelo, elaborou-se a matriz de confusão exibida na figura 9, considerando 40 classes e 120 observações do conjunto de teste.

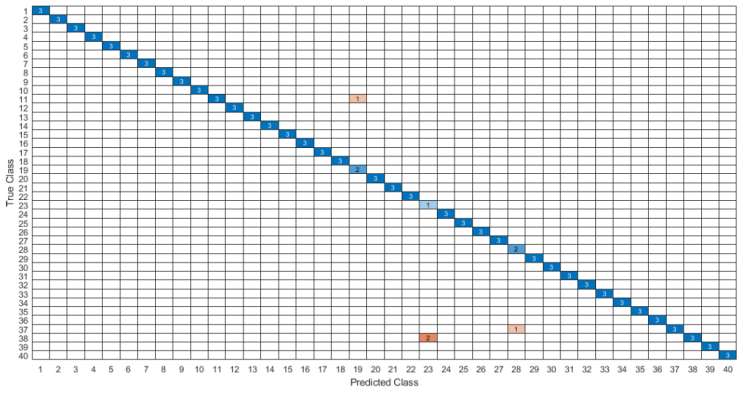


Figura 9. Matriz de confusão

## V. CONCLUSÕES

A implementação da rede neural de propagação direta tipo perceptron multicamadas mostrou-se eficaz na resolução do problema apresentado. Os conceitos de redução de dimensionalidade vistos em sala de aula puderam ser solidificados na implementação em MATLAB, com visualização imediata do efeito de compressão dos dados. Foi possível observar que o experimento no qual o método de redução foi baseado em 2D2LDA e teve como função de otimização o gradiente descente, obteve melhor desempenho, fato este comprovado pelas acurácias exibidas na tabela I e pela figura 7, a qual ilustra maior minimização do erro médio quadrático. A matriz de confusão exibida na figura 9 demonstra 4 erros frente a um conjunto extenso de dados, o que confere certa confiabilidade no modelo adquirido. No entanto, nota-se através da tabela II que o método baseado na distância euclidiana superou a rede desenvolvida, o que sugere a necessidade de aprimoramento do modelo gerado.

## REFERÊNCIAS

[1] SHEREENA, V. B.; DAVID, J.M. ”Significance of Dimensionality Reduction in Image Processing”. Signal Image Processing : An International Journal (SIPIJ) Vol.6, No.3, June 2015.

[2] YANG, J., ZHANG, D., YONG, X., YANG, J. ”Two-dimensional discriminant transform for face recognition”, Pattern Recognition Society, Vol. 38, P. 1125-1129, 2005.

[3] WANG, X.; HUANG, C., FANG, X., LIU, J. ”2DPCA vs. 2DLDA: Face Recognition using Two-Dimensional Method”, International Conference on Artificial Intelligence and Computational Intelligence, 2009.

[4] SOUZA, Robson S. ”Reconhecimento das Configurações de mão da língua brasileira de sinais (Libras) em Imagens de profundidade através da análise de componentes principais e do classificador de k-vizinhos mais próximos”, Dissertação de Mestrado - Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, Manaus, 2015.

```
Projeto4: 2DPCA Bidirecional
UFAM/PPGEE/Apredizado de M quina 2021-1
Carlos Fonseca, Willian Guerreiro

clc
clear all;
close all;
load ORL_FaceDataSet;  % ATT Face Dataset com 40 indiv duos (classes) e 10 amostras de cada
A=double(ORL_FaceDataSet);


Num_Class = 40;
No_SampleClass = 10;
DIM = 10;     % Qtde de Autovetores (DIM: 1 a n)

%  Defini  o do Conjunto de Treinamento
%  ------------------------------------
No_TrainSamples =  7;
No_TestSamples  =  3;

[TrainData, TestData]     = Train_Test(A,No_SampleClass,No_TrainSamples,No_TestSamples);
[m,n,TotalTrainSamples]   = size(TrainData);
% m = 112
% n =  92
% TotalTrainSamples: 200

%  Defini  o do Conjunto de Testes
[m1,n1,TotalTestSamples] = size(TestData);
[TrainLabel,TestLabel]   = LebelSamples(Num_Class, No_TrainSamples, No_TestSamples);

% m1 = 112
% n1 =  92
% TotalTestSamples: 200


% C lculo da Matriz de Covariancia (Proje  o Horizontal)
% Gh = 1/n * SUM (Ai - Mean(A))'*(Ai - Mean(A)), i = 1:n
%--------------------------------------------------------------------------
TrainMean = mean(TrainData,3); % M dia do conjunto de treinamento na dimens o "z"
Gh=zeros([ n n]);

% TrainData    um conjunto de 200 imagens de tamanho 'mxn':
% m = 112
% n =  92

for i = 1:TotalTrainSamples
    Temp = TrainData(:,:,i) - TrainMean; % (Ai - Mean(A))
    Gh = Gh + Temp'*Temp;                % (Ai - Mean(A))'*(Ai - Mean(A))
end

Gh=Gh/TotalTrainSamples;                 % fator 1/N, N: 200
%Gh     n x n:  92 x 92

% C lculo da Matriz de Dispers o Vertical
% Gv = 1/n * SUM (Ai - Mean(A))*(Ai - Mean(A))', i = 1:n
%--------------------------------------------------------------------------
TrainMean = mean(TrainData,3); % M dia do conjunto de treinamento na dimens o "z"
Gv=zeros([ m m]);

for i = 1:TotalTrainSamples              % TotalTrainSamples : 200
    Temp = TrainData(:,:,i) - TrainMean; % (Ai - Mean(A)): m x n
    Gv = Gv + Temp*Temp';                % (Ai - Mean(A))'*(Ai - Mean(A)): (m x n) * (n x m): m x m
end

Gv=Gv/TotalTrainSamples;                 % fator 1/N
```

```
%Gv     m x m:  112 x 112

%Redu   o de dimensionalidade com 2DLDA_BI

[U, V] = R_2DLDA_BI('ORL');


% Matriz de Proje   o Horizontal U --> EigVect_H
% Maiores "q" Autovalores e Autovetores de Gh ("q" = DIM)
%-----------------------------------------
[EigVect1,EigVal1] = eig_decomp(Gh);
EigVect_H          = EigVect1(:,1:DIM);
% U = EigVect_H = n x q, n: 92 e q = DIM

% Matriz de Proje   o Vertical V --> EigVect_V
% Maiores "q" Autovalores e Autovetores de Gh ("q" = DIM)
%-----------------------------------------
[EigVect1,EigVal1] = eig_decomp(Gv);
EigVect_V          = EigVect1(:,1:DIM);
% V = EigVect_V = m x r, m: 112 e r = DIM

% Calculo das Imagens Projetadas do Cj. Treinamento
% Ap = V' * A * U (Ap: imagem projetada, A: imagem original)
% Ap = (r x m) * (m x n) * (n x q) = (r x m) * (m x q) = r x q
% ---------------------------------------------------------------------------

for i = 1:TotalTrainSamples
    %Ytrain(:,:,i) = EigVect_V' * TrainData(:,:,i) * EigVect_H; % Ap = V' * A * U
     Ytrain(:,:,i) = V' * TrainData(:,:,i) * U; % Ap = V' * A * U
end
% YTrain = Ap --> (r x q) = (DIM x DIM) = (10 x 10)


% Reconhecimento das Faces com o Conjunto de Teste
%
%-------------------------------------------------
TestResult = zeros(TotalTestSamples,1);

for i=1:TotalTestSamples

    %Ytest(:,:,i) = EigVect_V' * TestData(:,:,i) * EigVect_H;
     Ytest(:,:,i) = V' * TestData(:,:,i) * U;
end

% Implementa   o do Perceptron Multicamadas para Classifica   o
% e Reconhecimento das Faces
% Arquitetura da Rede: 100 x 10 x 10 x 40
% Entrada: vetor com 100 caracter sticas: Imagens Proj. (PCA) de 10x10
% Sa da  : Um vetor com os valores preditos para cada um dos indiv duos
% 1o. approach: y = '1' para o indiv duo reconhecido, y = '0' para os demais
% 2o. approach: y = densidade de probabilidade [0 a 1] para a semelhan a
% entre a entrada e os 40 elementos poss veis na sa da
% ---------------------------------------------------------------------
%
%  Fun   es do Toolbox do Matlab
%
%  net = feedforwardnet(hiddenSizes,trainFcn)
%
%  hiddenSizes: [10 10], 2 camadas de tamanho 10
%
%  trainFcn:
%  1) 'traingd'  :      Gradient Descent
%  2) 'traingdm' :  Gradient Descent with Momentum
%
%  net:    a representa   o da rede construida
%
%  Treinamento da Rede
```

```
%
%  net = train(net, xtrain, ytrain);
%  view(net); visualiza  o da rede treinada
%


    %net = patternnet([10 10]); % condi  o que alcancou 91%
    net = patternnet([20], 'trainscg');
    %trainscg
    %net = feedforwardnet([20],'traincgf');
    net.layers{1}.transferFcn = 'poslin';
    net.layers{2}.transferFcn = 'softmax';

    t = zeros(Num_Class,1);

    for i = 1:TotalTrainSamples
     B = Ytrain(:,:,i);
     x = B(:); % converte uma matriz para um vetor coluna
     YTrainVector(:,i) = x;
     tVector(:,i) = t;
     tVector(TrainLabel(i),i) = 1;

    end


     % Alcancou 87% de acerto
     % ----------------------

     net.trainParam.epochs = 2000;
     net.trainParam.goal = 1e-6;
     net.trainParam.min_grad = 1e-6;
     net.trainParam.max_fail = 40;


     %YTrainVector_Norm = normalize(YTrainVector);
     YTrainVector_Norm = YTrainVector;
     %net  = train(net, YTrainVector_Norm, tVector);
     net  = trainbr(net, YTrainVector_Norm, tVector);

     view(net);

%      net.trainParam.epochs = 1000;
%      net.trainParam.goal = 1e-7;
%      net.trainParam.min_grad = 1e-7;
%      net.trainParam.max_fail = 200;
%      YTrainVector_Norm = normalize(YTrainVector);
%      net  = train(net, YTrainVector_Norm, tVector);


    % Testando a Classifica  o

    t = zeros(Num_Class,1);
    for i = 1:TotalTestSamples
     B = Ytest(:,:,i);
     x = B(:); % converte uma matriz para um vetor coluna
     YTestVector(:,i) = x;
     tTestVector(:,i) = t;
     tTestVector(TestLabel(i),i) = 1;
    end


    %YTestVector_Norm = normalize(YTestVector);
    YTestVector_Norm = YTestVector;
    y_pred = net(YTestVector_Norm(:,:));

    classes = vec2ind(y_pred);
    classes_reais = vec2ind(tTestVector);
```

```
Result = (classes_reais == classes);

%Result = '1' para os acertos (TestResult = TestLabel)

CorrectRate = 100*(sum(Result/TotalTestSamples))
%   ndice   de Acerto

perf = perform(net,y_pred,tTestVector);

cm = confusionchart(classes,classes_reais);
```