**Tumor Classification**

Conor Schleicher & Han Song

University of Washington

Department of Applied Mathematics

Seattle, WA

**Abstract**: In this paper, we are going to use a

Keywords: SVD, FFT, FFTSHIFT, SVM, classification. GitHub Repo: [Id.git](https://github.com/quinnhan/Id.git)

1. **Introduction and Overview:**

Tumor is an abnormal mass tissue results from either rapid growth of cells or cells do not die when they should. Tumor can be benign (not cancer) or malignant (cancer) [1]. The oldest recorded description about cancer dated back to 3000 BC in Egypt. From this documentation, cancer was called the Edwin Smith Papyrus and the writing also said, “there is no treatment.” Fast forward to the twenty first century, human had made remarkable progress in early detection, prevention, and treatment. We would like to emphasize on early detection process as most people who experienced cancer, don’t develop symptom until it’s late in the game. Being cautious, people start to get annual check-up and X-ray done. In this paper, we are going to develop an AI to determine whether an X-ray photo is showing sign of a tumor. We are going to build a photo library and implement function such as Fast Fourier Transform (FFT), Singular Value Decomposition (SVD), add name (SVM) and many other MATLAB manipulation techniques.

1. **Theoretical Background:**

We are going to approach with a theorem which states that every matrix **A** has a singular value decomposition and it takes form . Each of these matrix components has specific meaning to matrix (m x n matrix)**. U** matrix is a unitary matrix consists of n orthonormalized eigenvectors of which will rotate matrix **A** so that its axis the principle coordinates,consists of non-negative square roots of the eigenvalues of in a descending order which will “modify” the rotated axes to scale with the principle coordinates and**,** and lastly matrix is a unitary matrix consists of n orthonormalized eigenvectors of which will rotate the data point from the rotated principle axis.

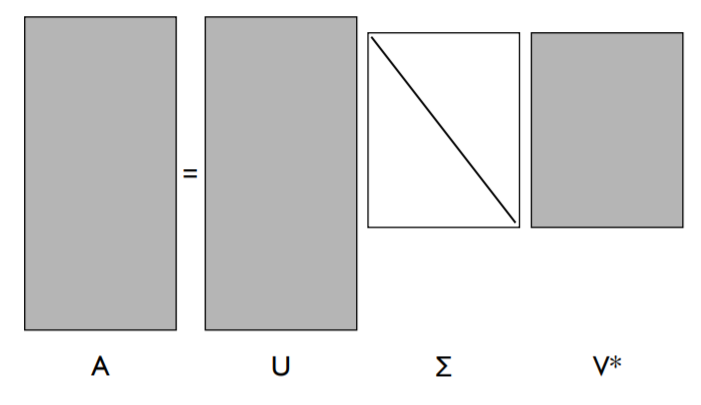


Figure 1: Graphical description of SVD (Credit: Professor Kutz)

Need to drill down more on SVD

Wavelet transform

SVM :

1. **Algorithm Implementation and Development:**

X-ray images were collected into a tumor present (yes folder) or a healthy person (no folder). Because all images’ size are not uniform, we grabbed the smallest image from both folder to be the comparison image:

if val < val2

smallestimage = fullfile(yesfolder,yesfiles(idx).name); % find name of smallest image

else

smallestimage = fullfile(nofolder,nofiles(idx2).name);

end

We then go through all images in both folders to scale all to the small image above. Since tumor appear on X-ray, it will likely being lit up or have a brighter color than black. Transformation to frequency domain is needed to detect those tumors. In frequency domain, all data is put into a big matrix X which will go through wavelet transform or SVD transform. A subset of matrix X will become test data and the rest of the columns are consider training data. Finally, we use SVM to predict the test images, and produce accuracy percentage.

for k = 1:length(truth)

if strcmp(pre{k},truth{k})

num\_correct = num\_correct + 1;

end

end

percentage(p) = (num\_correct/length(truth))\*100;

1. **Computational Results and Analysis:**

Need to fix code to show x and y label.

1. **Summary:**

This paper demonstrates that SVD can reduce significant size of data and still represent data with correct unique characteristic. KNN and SVM are somewhat efficient to classify data that are highly distinguishable such as classical music vs pop music.

Reference

[1] NCI Dictionary of Cancer Terms. National Cancer Institute. [www.cancer.gov](http://www.cancer.gov). Web. Mar 17, 2020

Appendix A: MATLAB functions

MODEL=fitcsvm(X,Y) is an alternative syntax that accepts X as an

N-by-P matrix of predictors with one row per observation and one column per predictor. Y is the response and is an array of N class labels.

[U,S,V] = svd(X,'econ') also produces the "economy size"

decomposition. If X is m-by-n with m >= n, then it is

equivalent to svd(X,0). For m < n, only the first m columns

of V are computed and S is m-by-m.

Appendix B: MATLAB Codes

%% Brain Tumor Classification

% initialize work space

clear all, close all, clc

% First task is to import the image data

yesfolder = 'TumorDataset/yes';

nofolder = 'TumorDataset/no';

yesfiles = dir(fullfile(yesfolder));

nofiles = dir(fullfile(nofolder));

yesfiles(1:2) = []; % deleting directory elements

% find the smallest image to resize to

% when combining them into one large matrix, they need to be the same size

[val, idx] = min([yesfiles.bytes]); % get min value and it's index

% repeat for the no cases

nofiles(1:2) = [];

[val2,idx2] = min([nofiles.bytes]);

% have to check for which is smaller then will resize images to all that

% size

if val < val2

smallestimage = fullfile(yesfolder,yesfiles(idx).name); % find name of smallest image

else

smallestimage = fullfile(nofolder,nofiles(idx2).name);

end

[rows, columns, colorchannels] = size(imread(smallestimage)); % get size of smallest image

% loop through each folder and add images to matrix

% yes dataset

YesData = [];

for i = 1:length(yesfiles)

basefilename = yesfiles(i).name;

fullfilename = fullfile(yesfolder,basefilename);

im = imread(fullfilename);

if size(im,3) > 1 % checks if image is not gray, rgb2gray can't work on already gray scale images

im = rgb2gray(im); % converting to gray scale for easier analysis

end

imr = imresize(im,[rows,columns]);

YesData(:,i) = double(imr(:));

end

% no dataset

NoData = [];

for j = 1:length(nofiles)

basefilename = nofiles(j).name;

fullfilename = fullfile(nofolder,basefilename);

im = imread(fullfilename);

if size(im,3) > 1 % checks if image is not gray

im = rgb2gray(im); % converting to gray scale for easier analysis

end

imr = imresize(im,[rows,columns]);

NoData(:,j) = double(imr(:));

end

%% Look at FFT of some images

% comparing the yes and the no images along with their respective FFT's

for i = 1:3

ys = YesData(:,i);

ns = NoData(:,i);

im\_y = reshape(ys,rows,columns);

im\_n = reshape(ns,rows,columns);

L = length(ys)/8400; % i didn't know what to put for here but I don't think it matters?

n = length(ys);

k=(2\*pi/L)\*[0:n/2 -n/2:-1];

ks=fftshift(k(1:end-1));

n2 = length(ns);

kn = (2\*pi/L)\*[0:n2/2 -n2/2:-1];

kns = fftshift(kn);

ys\_f = fft(ys);

ns\_f = fft(ns);

figure (i)

subplot(2,2,1)

imshow(uint8(im\_y))

subplot(2,2,2)

imshow(uint8(im\_n))

subplot(2,2,3)

plot(ks,abs(fftshift(ys\_f))/max(ys\_f))

subplot(2,2,4)

plot(kns(1:end-1),abs(fftshift(ns\_f))/max(ns\_f))

end

%% Split into training test groups

testruns = 20;

percentage = zeros(1,testruns);

for p = 1:testruns

q1 = randperm(size(YesData,2));

q2 = randperm(size(NoData,2));

% split the training/test data by a set amount

split\_yes = floor(0.8\*length(q1));

split\_no = floor(0.8\*length(q2));

YesData\_train = YesData(:,q1(1:split\_yes));

YesData\_test = YesData(:,q1((split\_yes + 1):end));

NoData\_train = NoData(:,q2(1:split\_no));

NoData\_test = NoData(:,q2((split\_no + 1):end));

X\_test = [YesData\_test, NoData\_test];

%% FFT the images and take SVD

X = [YesData\_train, NoData\_train];

% X\_fft = zeros(size(X));

% for l = 1:length(YesData\_train(1,:))

% X\_fft(:,l) = abs(fft(YesData\_train(:,l)));

% end

% try wavelet transform instead

X\_fft = tc\_wavelet(X,rows,columns);

X\_test\_wav = tc\_wavelet(X\_test,rows,columns);

[U,S,V] = svd(X\_fft,'econ');

%% Graph some stuff

%figure()

sig = diag(S);

[M,N] = size(X);

%subplot(1,2,1), plot(sig(1:50),'ko','Linewidth',[1.5])

ylabel('Singular Values')

xlabel('Singular Value Along Diagonal')

%subplot(1,2,2), semilogy(sig(1:50),'ko','Linewidth',[1.5])

ylabel('Log of Singular Values')

xlabel('Singular Value Along Diagonal')

%% Run through matlab classify

numFeat = 30;

xtrain = V(:,1:numFeat);

xtest = U'\*X\_test\_wav;

ctrain = [repmat({'Tumor'},[size(YesData\_train,2),1]);repmat({'NoTumor'},[size(NoData\_train,2),1])];

truth = [repmat({'Tumor'},[size(YesData\_test,2),1]);repmat({'NoTumor'},[size(NoData\_test,2),1])];

svm.mod = fitcsvm(xtrain,ctrain);

pre = predict(svm.mod,xtest(:,1:numFeat));

num\_correct = 0;

for k = 1:length(truth)

if strcmp(pre{k},truth{k})

num\_correct = num\_correct + 1;

end

end

percentage(p) = (num\_correct/length(truth))\*100;

end

mean(percentage)