# **Lab 13**

*Signal Processing on Graphs*

*Classification of Cancer Types*

Braden Fineberg

Jonathan Potter

**2.1 Understanding the data**

Script:

close all;

clc;

load('geneNetwork\_rawPCNCI.mat');

load('signal\_mutation.mat');

load('histology\_subtype.mat');

gN = +gN;

%%%%%I renamed the gene network to gN%%%%%

%%%%%I renamed the signal mutation to sigMut%%%%%

%%%%%I renamed the histology\_subtype to hist%%%%%

loops = sum(diag(gN) ~= 0);

directed = ~issymmetric(gN);

wieghted = (max(max(gN)) > 1);

spy(gN);

title('Visualized Gene Network');

%%% With self loops

D = diag(sum(gN~=0,2));

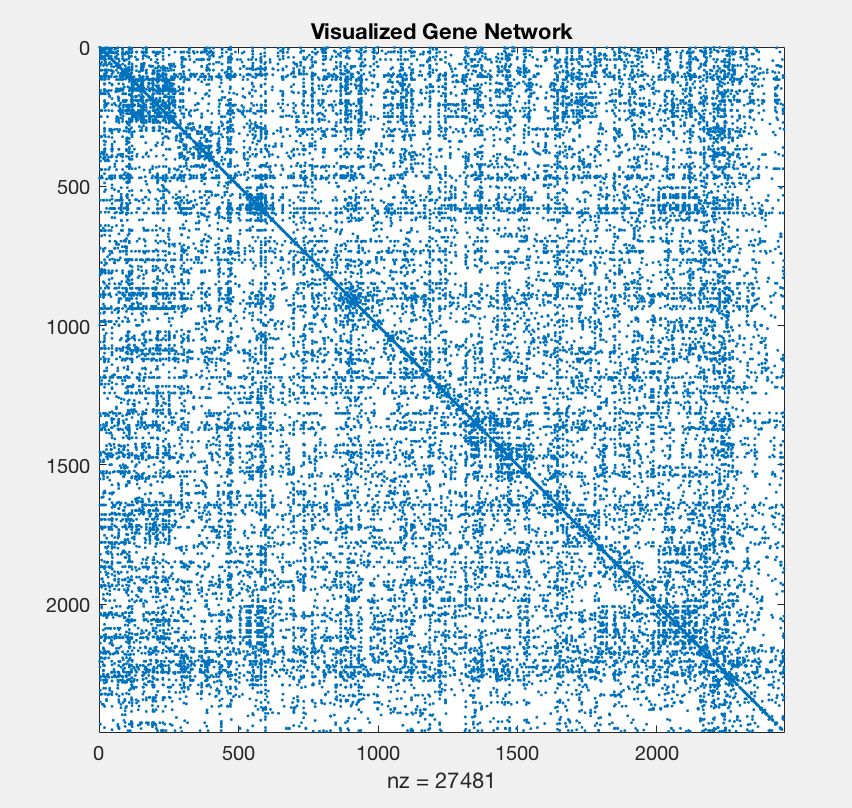
lp = D - gN;

%%% Remove diag

gNd = gN - diag(diag(gN));

g = graph(gNd);

l = laplacian(g);



Considering there are values greater than 0 along the diagonal of the adjacency matrix, there are self-loops. By summing along a the diagonal, we discovered there are 1525 self-loops in this graph. Additionally, we know the graph is undirected and unweighted since the adjacency matrix is symmetric and on exhibits values of 0 and 1. With regards to L and Ltilda, we know these matrices are the same. Removing the self-loops in A actually ensures no change between the matrices based on the nature of the Laplacian’s formula, D-A.

**2.2 Total variations**

Script:

S = lp;

[V, D] = eig(S);

d=diag(D);

[~,order]=sort(abs(d),'ascend');

V=V(:,order); d=d(order); D=diag(d);

if max(max(abs(S-V\*D\*V')))<1e-9

S=V\*D\*V';

else

error('The eigendecomposition is not good.\n')

end

variation=diag(V'\*S\*V);

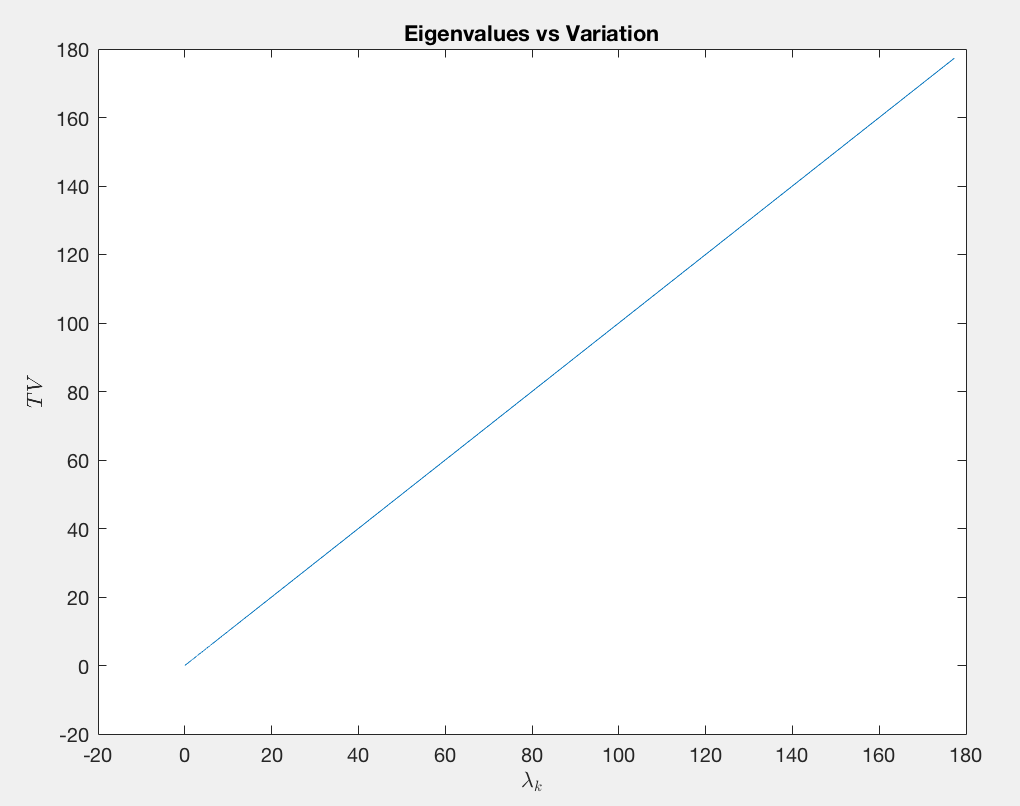
figure;

plot(diag(D), variation);

title('Eigenvalues vs Variation');

xlabel('$\lambda\_{k}$','Interpreter','LaTeX');

ylabel('$TV$','Interpreter','LaTeX');



Using the above graph, we can determine that the total variation of the signal increases as the values of eigenvalues increase. In other words, eigenvectors associated with larger eigenvalues oscillate more rapidly than the eigenvectors associated with smaller eigenvalues. There is a correlation of 1:1.

**3.1 Distinguishing power**

Script:

D = diag(sum(gN));

lp = D - gN;

[V, ~] = eigs(lp, length(lp));

H = V';

gft = zeros(240,2458);

for i=1:240

gft(i,:) = sigMut(i,:)\*H;

end

type1 = gft(hist==1, :);

mu1 = mean(type1, 1);

type2 = gft(hist==2, :);

mu2 = mean(type2, 1);

n = sum(abs(gft), 1);

res = abs(mu1-mu2);

res = res./n;

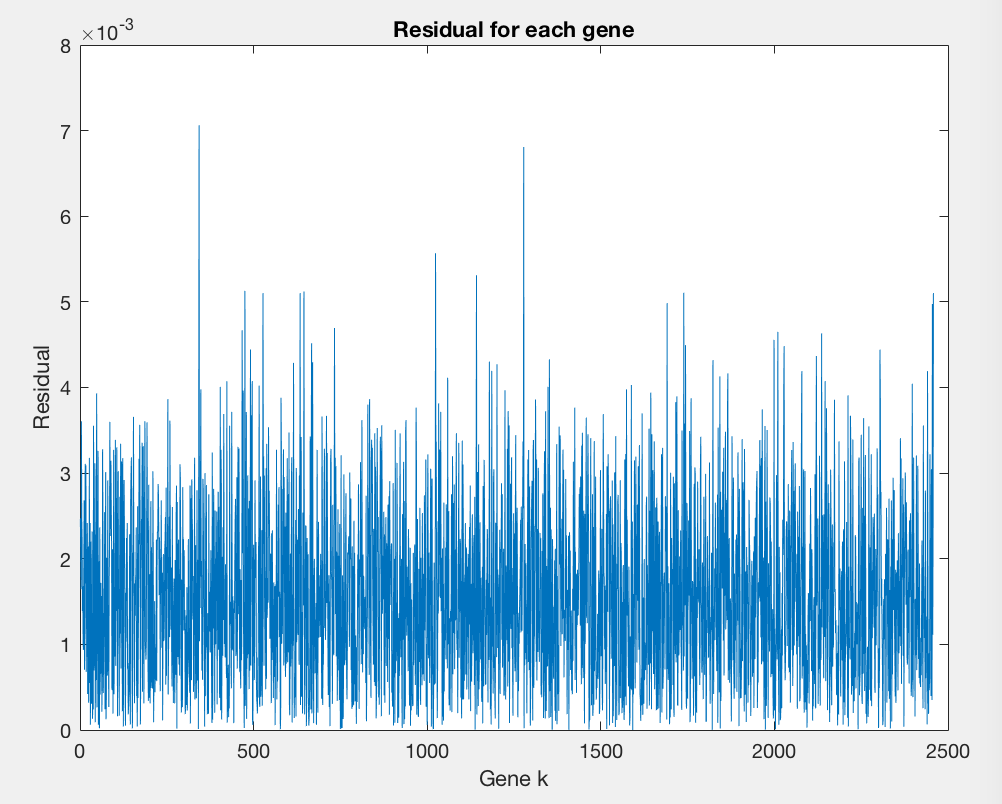
figure;

plot(1:length(sigMut), abs(res));

title('Residual for each gene');

xlabel('Gene k');

ylabel('Residual');



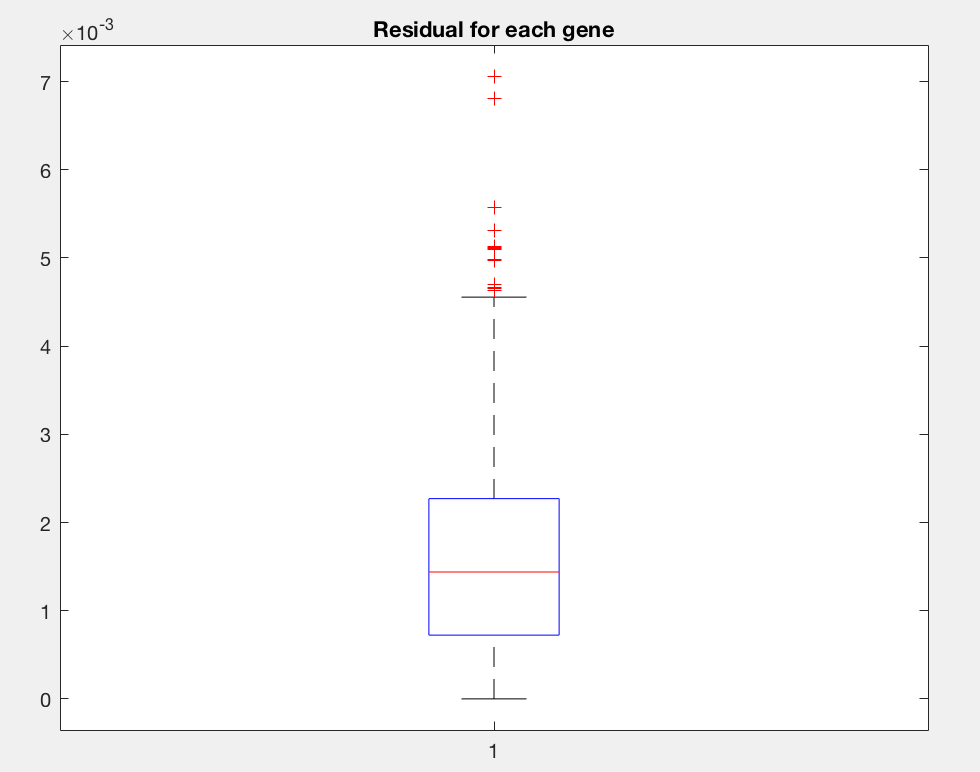
**3.2 Interpretation**

Script:

figure;

boxplot(abs(res));

title('Residual for each gene');



The boxplot indicates that the majority of genes are not indicative of the phenotype. However, there are several outliers that could help determine the type of cancer that the patient has. The gene with the greatest magnitude, at 7e-3, will likely be indicative of the type of cancer.

**4.1 k-NN for leave-one-out cross validation**

Function: kNN

function [k, acc ] = kNN(sig, type, max\_k)

%KNN Summary of this function goes here

% Detailed explanation goes here

pd = squareform(pdist(sig));

k= 1:max\_k;

acc=zeros(1,max\_k);

for i=1:max\_k

z = zeros(length(type),1);

for j = 1:length(type)

[~, nn] = sort(pd(:,j),1,'ascend');

z(j) = mode(type(nn(2:(i+1),:)),1);

end

acc(i) = sum(z == type) / length(type);

end

end

Script:

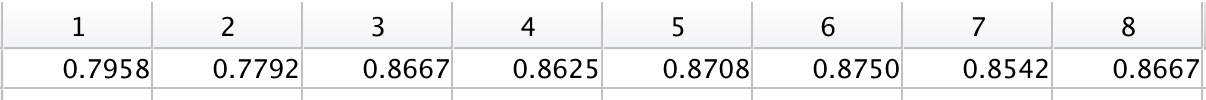
[k, acc] = kNN(sigMut, hist, 100);

plot(k, acc);

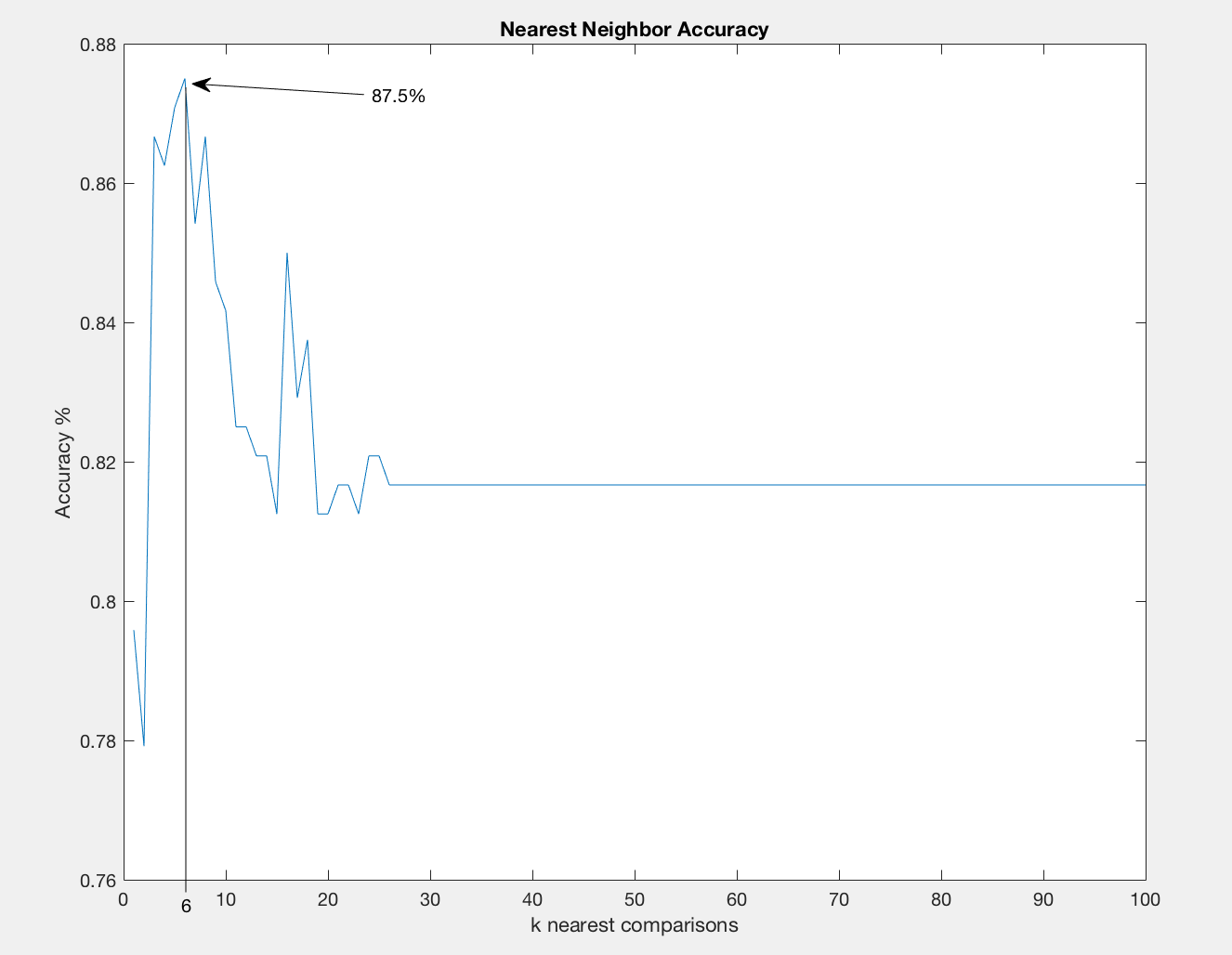
title('Nearest Neighbor Accuracy');

xlabel('k nearest comparisons');

ylabel('Accuracy %');



As you can see above, the accuracies corresponding with K values 3, 5, and 7 are 0.8667, 0.8708, and 0.8542, respectively.



The graph above shows the accuracy for every k comparisons between 0 and 100. We find that k=6 allows for the greatest accuracy before overfitting causes the error to increase.

**4.2 Graph filters**

Script:

[~, k] = max(gft, [], 2);

H1 = zeros(240, 2458);

for i=1:240

H1(i,k(i)) = 1;

end

fgft = H1.\*gft;

fsig = fgft \* V;

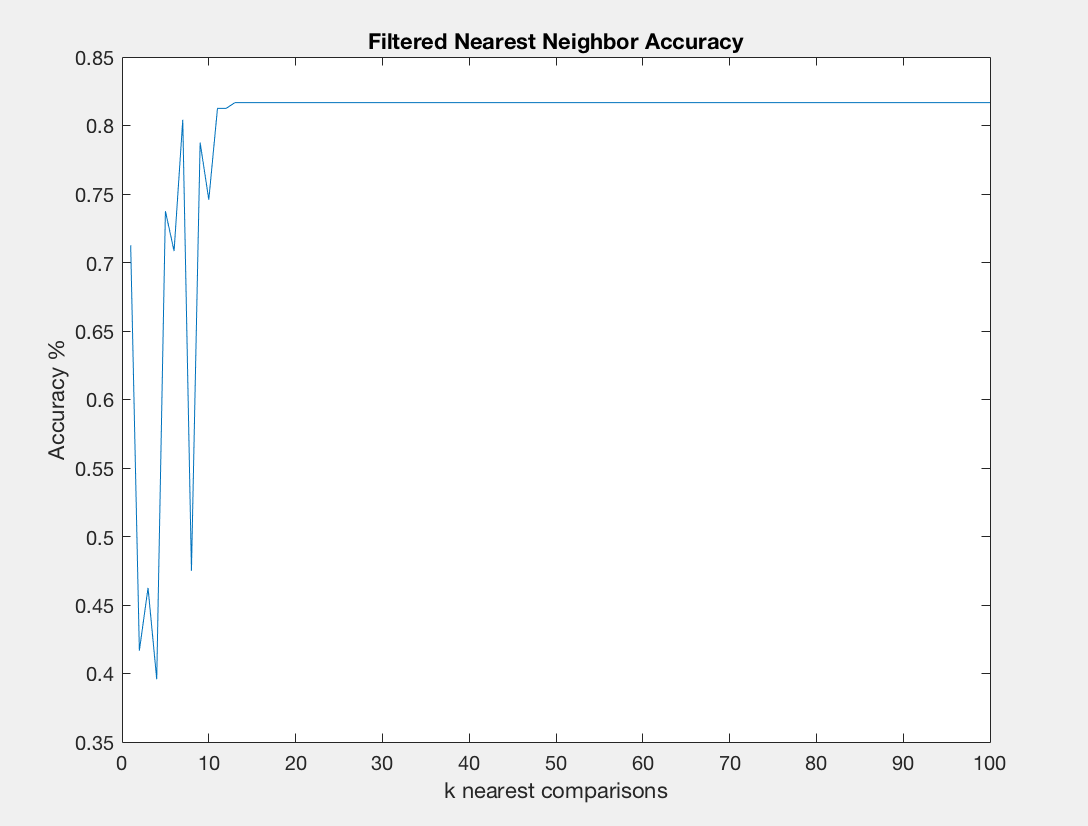
[k, acc] = kNN(fsig, hist, 100);

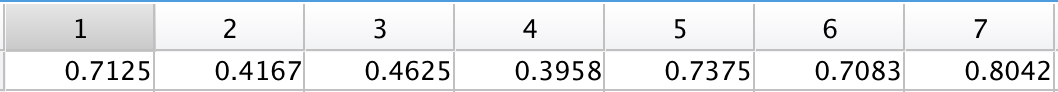
plot(k, acc);

title('Filtered Nearest Neighbor Accuracy');

xlabel('k nearest comparisons');

ylabel('Accuracy %');

****

****

As you can see above, the accuracies corresponding with K values 3, 5, and 7 are 0.4625, 0.7375, and 0.8042, respectively.

This accuracy is lower than the kNN comparison because we are only comparing one value for each patient.

The percentile kNN worked the best when we looked at the 50th percentile and compared with 5 nearest neighbors, generating an accuracy of 93.3%.

Script:

p = [20 40 50 60 80 99];

[~, srt] = sort(res, 'descend');

figure;

for i=1:length(p)

per = round((p(i)/100) \* length(res));

fltr = zeros(1, 2458);

fltr(res\*1000000 >= srt(per)) = res(res\*1000000 >= srt(per));

fgft = fltr.\*gft;

fsig = fgft \* V;

[k, acc] = kNN(fsig, hist, 10);

plot(k, acc);

hold on;

end

legend('20th Percentile', '40th Percentile', '50th Percentile','60th Percentile', '80th Percentile', '99th Percentile');

xlabel('K nearest neighbor Comparison');

ylabel('accuracy %');

title('Percentile Filtered kNN');

