The University of British Columbia

CPSC 340

Machine Learning and Data Mining

Homework #5

Student name:

ALI SIAHKOOHI

Student ID:

84264167

Instructor:

Professor Schmidt

Date Submitted:

November 26th, 2016

Question #1

Part 1.1.1

After centering the data we get the following data points:

x_1	x_2
-2	$-1-1=-2$
-1	$0-1=-1$
0	$1-1=0$
1	$2-1=1$
2	$3-2=2$

So two features are the same therefore the the PC is $\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}$. Because all the data points lie on this line.

Part 1.1.2

$$W=\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix},$$

$$Z=\hat{X}W^T=\begin{pmatrix}3 & 3\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=3\sqrt{2},$$

$$\Rightarrow X_{estimaed}=ZW=\begin{pmatrix}3 & 3\end{pmatrix},$$

$$\text{Reconstruction error }=\left\|\hat{X}-X_{estimated}\right\|_2=0. \qquad (1-1)$$

Part 1.1.3

$$W=\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix},$$

$$Z=\hat{X}W^T=\begin{pmatrix}3 & 4\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=\frac{7}{\sqrt{2}},$$

$$\Rightarrow X_{\text{estimaed}} = ZW = \begin{pmatrix} \frac{7}{2} & \frac{7}{2} \end{pmatrix},$$

$$\text{Reconstruction error } = \left\| \hat{X} - X_{\text{estimated}} \right\|_2 = \left\| \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix} \right\|_2 = \frac{1}{\sqrt{2}}.$$

Part 1.2

```
load animals.mat

[n,d] = size(X);
X = standardizeCols(X);

figure(1);
imagesc(X);
figure(2);
i = ceil(rand*d);
j = ceil(rand*d);

[model] = dimRedPCA(X,2);
W = model.W;
Z = X*W'*(W*W')^-1;

plot(Z(:,1),Z(:,2),'.');
gname(animals);
```

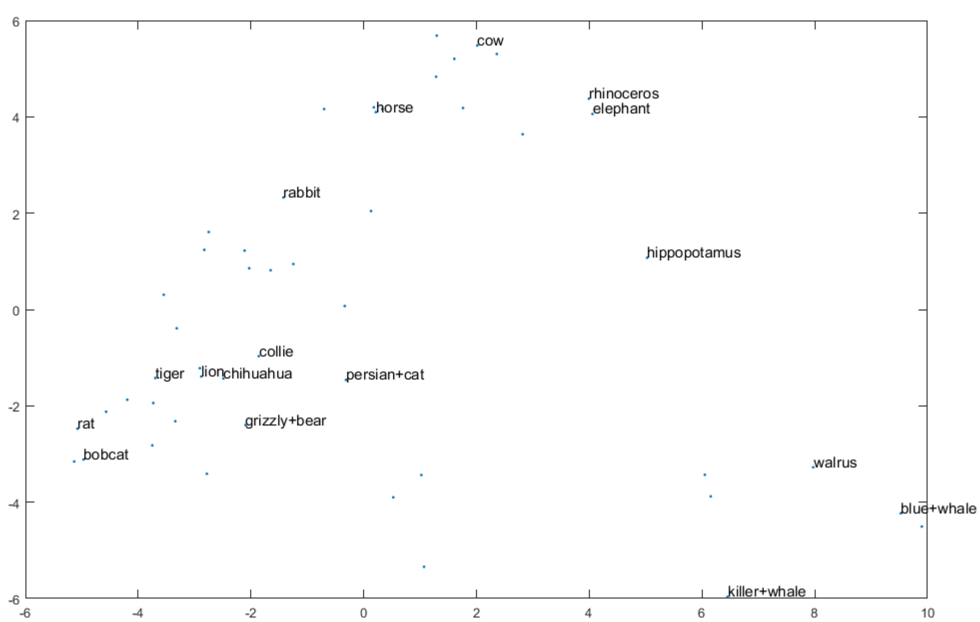

Figure (1 – 1) Exercise 1.2

Part 1.3.1

$$1-\frac{\|ZW-X\|_F}{\|X\|_F}=0.1645$$

Part 1.3.2

We need 14 PCs to explain 50% of the variance in the data because:

$$k=13\Rightarrow 1-\frac{\|ZW-X\|_F}{\|X\|_F}=0.4995,$$
$$k=14\Rightarrow 1-\frac{\|ZW-X\|_F}{\|X\|_F}=0.5190,$$

Question #2

```
function [model] = dimRedRPCA(X,k)

[n,d] = size(X);

% Subtract mean
mu = mean(X);
X = X - repmat(mu,[n 1]);

% Initialize W and Z
W = randn(k,d);
Z = randn(n,k);

R = Z*W-X;
f = sum(sum(R.^2));
for iter = 1:50
    fOld = f;

    % Update Z
    Z(:) = findMin(@funObjZ,Z(:),10,0,X,W);

    % Update W
    W(:) = findMin(@funObjW,W(:),10,0,X,Z);

    R = Z*W-X;
    f = sum(sum(R.^2));
    fprintf('Iteration %d, loss = %.5e\n',iter,f);

    if fOld - f < 1e-4
        break;
    end
end
```

```
model.mu = mu;
model.W = W;
model.compress = @compress;
model.expand = @expand;
end

function [Z] = compress(model,X)
[t,d] = size(X);
mu = model.mu;
W = model.W;
k = size(W,1);

X = X - repmat(mu,[t 1]);
% We didn't enforce that W was orthogonal so we need to optimize to find Z
Z = zeros(t,k);
Z(:) = findMin(@funObjZ,Z(:),100,0,X,W);
end

function [X] = expand(model,Z)
[t,d] = size(Z);
mu = model.mu;
W = model.W;

X = Z*W + repmat(mu,[t 1]);
end

function [f,g] = funObjW(W,X,Z)
% Resize vector of parameters into matrix
d = size(X,2);
k = size(Z,2);
W = reshape(W,[k d]);



epsil = 1e-4;

R = Z*W-X;
f = sum(sum(abs(R)));

g = zeros(size(W));
for p = 1:k
    for q = 1:d
        for i = 1:size(X, 1)
            g(p, q) = g(p, q) + (Z(i, p)*W(p, q) - X(i, q))*Z(i, p)/ ...
                sqrt((Z(i, p)*W(p, q) - X(i, q))^2 + epsil);
        end
    end
end

% Return a vector
g = g(:);
end

function [f,g] = funObjZ(Z,X,W)
% Resize vector of parameters into matrix
n = size(X,1);
k = size(W,1);
```

```
Z = reshape(Z,[n k]);

epsil = 1e-4;
% Compute function value
R = Z*W-X;
f = sum(sum(abs(R)));
g = zeros(size(Z));
for p = 1:n
    for q = 1:k
        for j = 1:size(X, 2)
            g(p, q) = g(p, q) + (Z(p, q)*W(q, j) - X(p, j))*W(q, j)/ ...
                sqrt((Z(p, q)*W(q, j) - X(p, j))^2 + epsil);
        end
    end
end

% Return a vector
g = g(:);
end
```

Part 2.2.1

Increase in λ_W causes matrix W to be more sparse and doesn't have an effect on sparsity of Z.

Increasing λ_Z doesn't have an effect on sparsity of Z, although by increasing λ_Z we are imposing L2 norm of Z to be more reduced comparing to the L1 norm of W. in other words by keeping λ_W constant and increasing λ_Z we can indirectly reduce the sparsity of W.

Part 2.2.2

Effect of λ_Z:

By decreasing λ_Z we can improve the training error because it doesn't impose a weight on having a small L2 norm for Z, therefore we would get a lower data fitting error.

By increasing λ_Z, we are preventing the algorithm from predicting the data precisely, therefore we are reducing the chance of over fitting. The training error becomes a better approximation of test error.

Effect of k:

By increasing k, we can capture all the details in the data therefore we improve the training error.

By increasing k, since we over fit the data, the training error becomes a worse approximation of testing error.

Part 2.2.3

No it won't change. The parameter λ_W itself has a same effect as λ_Z.

Part 2.2.4

I would use the logistic loss:

$$F(W,Z) = \sum_{(i,j)\in\mathbb{R}} \log\left(1+\exp\left(-x_{ij}w_j^T z_i\right)\right) + \lambda_W \sum_{i=1}^{d} \|w_i\|_1 + \frac{\lambda_Z}{2}\sum_{i=1}^{n}\|z_i\|^2$$

Question #3

Part 3.1

MATLAB code:

```
function [Z] = visualizeISOMAP(X,k,names)

[n,d] = size(X);

% Compute all distances
D = X.^2*ones(d,n) + ones(n,d)*(X').^2 - 2*X*X';
D = sqrt(abs(D));

G = zeros(size(D));
for i = 1:size(D, 1)
    [~, I] = sort(D(i, :));
    G(i, I(1:(k+1))) = D(i, I(1:(k+1)));
end

for i = 1:size(D, 1)
    for j = 1:size(D, 2)
        if G(i, j) ~= 0
            G(j, i) = G(i, j);
        end
    end
end


DD = zeros(size(D));
for i = 1:size(D, 1)
    for j = 1:size(D, 2)
        [L, ~] = dijkstra(G,i,j);
        DD(i, j) = L;
    end
end
```

```
% Initialize low-dimensional representation with PCA
model = dimRedPCA(X,2);
Z = model.compress(model,X);

Z(:) = findMin(@stress,Z(:),500,0,DD,names);

end

function [f,g] = stress(Z,D,names)

n = length(D);
k = numel(Z)/n;

Z = reshape(Z,[n k]);

f = 0;
g = zeros(n,k);
for i = 1:n
    for j = i+1:n
        % Objective Function
        Dz = norm(Z(i,:)-Z(j,:));
        s = D(i,j) - Dz;
        f = f + (1/2)*s^2;

        % Gradient
        df = s;
        dgi = (Z(i,:)-Z(j,:))/Dz;
        dgj = (Z(j,:)-Z(i,:))/Dz;
        g(i,:) = g(i,:) - df*dgi;
        g(j,:) = g(j,:) - df*dgj;
    end
end
g = g(:);

% Make plot if using 2D representation
if k == 2
    figure(3);
    clf;
    plot(Z(:,1),Z(:,2),'.');
    if ~isempty(names)
        hold on;
        for i = 1:n
            text(Z(i,1),Z(i,2),names(i,:));
        end
    end
    pause(.01)
end
end
```

Result with k = 3:

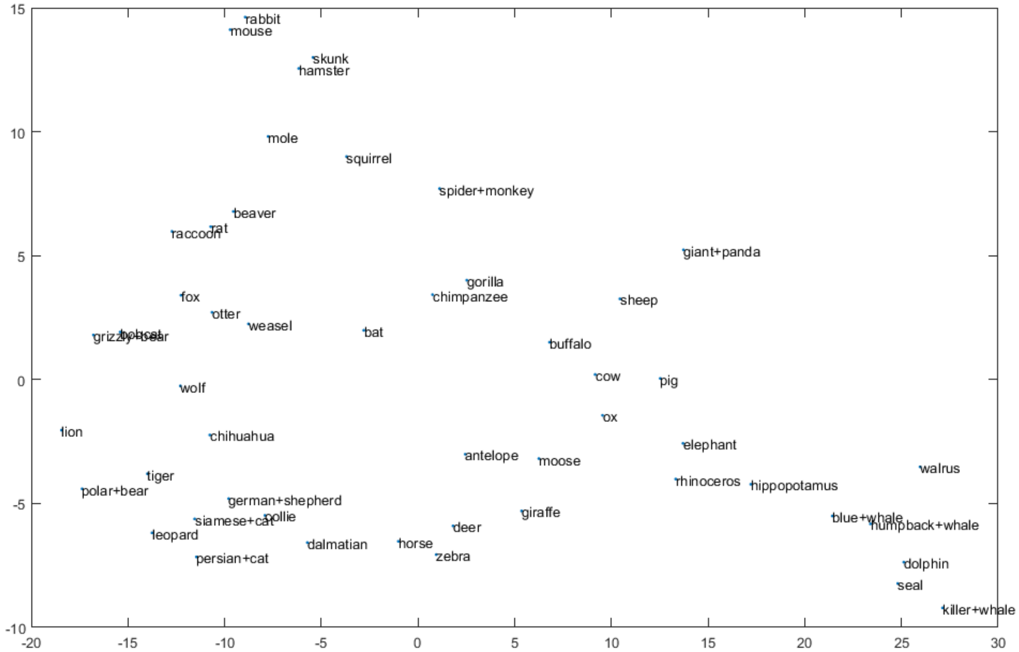

Figure (3 – 1)

Part 3.2

MATLAB code:

```
function [Z] = visualizeISOMAP(X,k,names)

[n,d] = size(X);

% Compute all distances
D = X.^2*ones(d,n) + ones(n,d)*(X').^2 - 2*X*X';
D = sqrt(abs(D));

G = zeros(size(D));
for i = 1:size(D, 1)
    [~, I] = sort(D(i, :));
    G(i, I(1:(k+1))) = D(i, I(1:(k+1)));
end

for i = 1:size(D, 1)
    for j = 1:size(D, 2)
        if G(i, j) ~= 0
            G(j, i) = G(i, j);
        end
    end
end
```

```
DD = zeros(size(D));
for i = 1:size(D, 1)
    for j = 1:size(D, 2)
        [L, ~] = dijkstra(G,i,j);
        DD(i, j) = L;
    end
end

vecD = DD(:);
vecD = sort(vecD, 'descend');
ind = find(vecD ~= inf, 1);
MaxDD = vecD(ind);
for i = 1:size(DD, 1)
    for j = 1:size(DD, 2)
        if DD(i, j) == inf
            DD(i, j) = MaxDD;
        end
    end
end

% Initialize low-dimensional representation with PCA
model = dimRedPCA(X,2);
Z = model.compress(model,X);

Z(:) = findMin(@stress,Z(:),500,0,DD,names);

end

function [f,g] = stress(Z,D,names)

n = length(D);
k = numel(Z)/n;

Z = reshape(Z,[n k]);

f = 0;
g = zeros(n,k);
for i = 1:n
    for j = i+1:n
        % Objective Function
        Dz = norm(Z(i,:)-Z(j,:));
        s = D(i,j) - Dz;
        f = f + (1/2)*s^2;

        % Gradient
        df = s;
        dgi = (Z(i,:)-Z(j,:))/Dz;
        dgj = (Z(j,:)-Z(i,:))/Dz;
        g(i,:) = g(i,:) - df*dgi;
        g(j,:) = g(j,:) - df*dgj;
    end
end
g = g(:);

% Make plot if using 2D representation
```

```
if k == 2
    figure(3);
    clf;
    plot(Z(:,1),Z(:,2),'.');
    if ~isempty(names)
        hold on;
        for i = 1:n
            text(Z(i,1),Z(i,2),names(i,:));
        end
    end
    pause(.01)
end
end
```

Result with k = 2 and avoiding inf entries of the matrix:

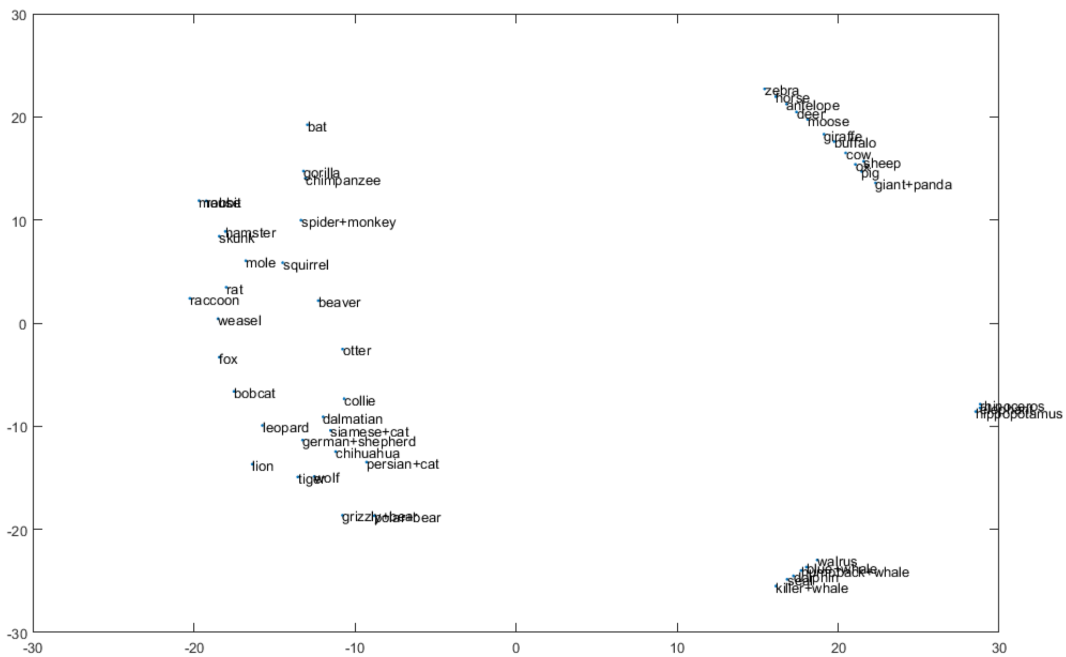

Figure (3 – 2)