ASSIGNMENT 4

KARAN MANGLA
201301205

Q1.

Assignment 4

Karan Mangla
201301205.

Q1. To extend LDA to non-linear mappings, the data can be mapped to a new feature space, F, via some function ϕ. In this new feature space, the function that needs to be maximised is:

$$J(w) = \frac{w^T S_B^\phi w}{w^T S_w^\phi w}$$

where

$$S_B^\phi = (m_2^\phi - m_1^\phi)(m_2^\phi - m_1^\phi)^T$$

$$S_w^\phi = \sum_{i=1,2} \sum_{n=1}^{l_i} (\phi(x_n^i) - m_i^\phi)(\phi(x_n^i) - m_i^\phi)^T$$

$$m_i^\phi = \frac{1}{l_i} \sum_{j=1}^{l_i} \phi(x_j^i)$$

Note $w \in F$. We use kernel trick in which the dot product in the new feature space is replaced by a kernel function.

$$k(x,y) = \phi(x) \cdot \phi(y).$$

LDA now can be reformulated in terms of dot products by noting that $w = \sum_{i=1}^{l} \alpha_i \phi(x_i)$.

$$w^T \cdot m_i^\phi = \frac{1}{l_i} \sum_{j=1}^{l} \sum_{k=1}^{l_i} \alpha_j \, k(x_j, x_k^i) = \alpha^T M_i.$$

where $(M_i)_j = \frac{1}{l_i} \sum_{k=1}^{l_i} k(x_j, x_k^i)$.

numerator of $J(w)$ can be written as:

$$w^T S_B^\phi w = w^T (m_2^\phi - m_1^\phi)(m_2^\phi - m_1^\phi)^T w$$

$$= \alpha^T M \alpha$$

where $M = (M_2 - M_1) \cdot (M_2 - M_1)^T$.

denominator $\rightarrow$ ~~M =~~

$$w^T S_W^\phi w = \alpha^T N \alpha.$$

$$\text{where } N = \sum_{j=1,2} K_j (I - 1_{l_j}) K_j^T$$

with n^{th}, m^{th} component of K_j defined as $k(x_n, x_m^j)$.

I is the identity matrix, and 1_{l_j} the matrix with all entries equal to $1/l_j$.

$$w^T S_W^\phi w = \left(\sum_{i=1}^{l} \alpha_i \phi^T(x_i)\right)\left(\sum_{j=1,2} \sum^{l_j} (\phi(x_n^j) - m_j^\phi)(\phi(x_n^j) - m_j^\phi)^T\right) \cdot \left(\sum_{k=1}^{l} \alpha_k \phi(x_k)\right)$$

$$= \sum_{j=1,2} \sum_{i=1}^{l} \sum_{n=1}^{l_j} \sum_{k=1}^{l} \alpha_i \phi^T(x_i)(\phi(x_n^j) - m_j^\phi)(\phi(x_n^j) - m_j^\phi)^T \alpha_k \phi(x_k)$$

$$= \sum_{j=1,2} \sum_{i=1}^{l} \sum_{n=1}^{l_j} \sum_{k=1}^{l} \left(\alpha_i k(x_i, x_n^j) - \frac{1}{l_j} \sum_{p=1}^{l_j} \alpha_i k(x_i, x_p^j)\right) \cdot \left(\alpha_k k(x_k, x_n^j) - \frac{1}{l_j}\left(\sum_{q=1}^{l_j} \alpha_k k(x_k, x_q^j)\right)\right)$$

$$= \sum_{j=1,2} \left(\sum_{i=1}^{l} \sum_{n=1}^{l_j} \sum_{k=1}^{l} \left(\alpha_i \alpha_k k(x_i, x_n^j) - \frac{\alpha_i \alpha_k}{l_j} \sum k(x_i, x_n^j) \cdot k(x_k, x_p^j)\right)\right)$$

$$= \sum_{j=1,2} \alpha^T K_j K_j^T \alpha - \alpha^T K_j 1_{l_j} K_j^T \alpha$$

$$= \alpha^T N \alpha.$$

with these equations for the numerator & denominator of $J(w)$, the equation for J can be rewritten as

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha}.$$

differentiating w.r.t α and setting equal to 0 gives.

$$(\alpha^T M \alpha) N \alpha = (\alpha^T N \alpha) \cdot M \alpha.$$

since only the direction of w, and hence direction of α, matters, the above can be solved for α as

$$\alpha = N^{-1}(M_2 - M_1)$$

N is usually singular, so multiple of I is added to it

$$N_\epsilon = N + \epsilon I.$$

Given the solution for α, the projection of a new data point is given by.

$$y(x) = (w \cdot \phi(x)) = \sum_{i=1}^{l} \alpha_i \cdot k(x_i, x).$$

Q2.
Arcene- Kernel PCA

```
a = importdata('arcene_train.data');
b = importdata('arcene_train.labels');
c = importdata('arcene_valid.data');
d = importdata('arcene_valid.labels');
sigma = 10000;
[x1 y1] = size(a);
[x2 y2] = size(c);
kernel = zeros(x1,x1);

for i=1:x1
   for j=1:x1
      kernel(i,j) = exp(-norm(a(i,:)-a(j,:))^2/sigma^2);
   end
end
temp = ones(x1,x1)/x1;
kernel_new = kernel - temp*kernel - kernel*temp + temp*kernel*temp;
[eigenvectors1 eigenvalues1] = eig(kernel_new);
eigenvalues1 = diag(eigenvalues1);

%eigenvalues1 = eigenvalues1(end:-1:1);
%eigenvectors1 = fliplr(eigenvectors1);

kernel_t = zeros(x2,x1);
for i=1:x2
   for j=1:x1
      kernel_t(i,j) = exp(-norm(c(i,:)-a(j,:))^2/sigma^2);
   end
end

for i=1:x1
   eigenvectors1(:,i) = eigenvectors1(:,i)/eigenvalues1(i);
```

```
end

for t=1:2
    v1 = eigenvectors1(:,1:10^t);
    train1 = kernel_new*v1;
    test1 = kernel_t*v1;
    trainmodel1 = svmtrain(train1,b);

acc_linear=100*(size(find(svmclassify(trainmodel1,test1)==d),1)/size(tr
ain1,1));
    disp(acc_linear);
    trainmodel1 = svmtrain(train1,b,'kernel_function','rbf','rbf_sigma',5);

acc_rbf=100*(size(find(svmclassify(trainmodel1,test1)==d),1)/size(trai
n1,1));
    disp(acc_rbf);
end
```

Arcene- Kernel LDA

```
a = importdata('arcene_train.data');
b = importdata('arcene_train.labels');
c = importdata('arcene_valid.data');
d = importdata('arcene_valid.labels');
sigma = 10000;
[x1 y1] = size(a);
[x2 y2] = size(c);
kernel = zeros(x1,x1);
for i=1:x1
    for j=1:x1
        kernel(i,j) = exp(-norm(a(i,:)-a(j,:))^2/sigma^2);
    end
end
```

```
temp = ones(x1,x1)/x1;
kernel_n = kernel - temp*kernel - kernel*temp + temp*kernel*temp;

kernel_t = zeros(x2,x1);
for i=1:x2
   for j=1:x1
      kernel_t(i,j) = exp(-norm(c(i,:)-a(j,:))^2/sigma^2);
   end
end

m1ind = find(b==1);
m2ind = find(b==-1);
M1 = mean(kernel_n(m1ind,:));
M2 = mean(kernel_n(m2ind,:));
N =
kernel_n(m1ind,:)'*(eye(size(m1ind,1))-(1/size(m1ind,1)))*kernel_n(m1i
nd,:) +
kernel_n(m2ind,:)'*(eye(size(m2ind,1))-(1/size(m2ind,1)))*kernel_n(m2i
nd,:);
N1 = N + 644*eye(size(kernel_n,1));
N = N + 8000*eye(size(kernel_n,1));
train = kernel_n*inv(N)*(M1-M2)';
test = kernel_t*inv(N)*(M1-M2)';
train1 = kernel_n*inv(N1)*(M1-M2)';
test1 = kernel_t*inv(N1)*(M1-M2)';
trainmodel = svmtrain(train,b);
accuracy = size(find(svmclassify(trainmodel, test)==d),1);
trainmodel = svmtrain(train1,b,'kernel_function','rbf');
accuracy1 = size(find(svmclassify(trainmodel, test1)==d),1);
accuracy
accuracy1
```

<table>
<tr><th></th><th>PCA</th><th>LDA</th></tr>
<tr><td>Arcene</td><td><table><tr><td>K=10</td><td>Linear</td><td>56.0</td></tr><tr><td>K=10</td><td>RBF</td><td>70.0</td></tr><tr><td>K=100</td><td>Linear</td><td>56.0</td></tr><tr><td>K=100</td><td>RBF</td><td>68.0</td></tr></table></td><td><table><tr><td>Linear</td><td>68.0</td></tr><tr><td>RBF</td><td>69.0</td></tr></table></td></tr>
<tr><td>Breast Cancer</td><td><table><tr><td>K=10</td><td>Linear</td><td>65.6891</td></tr><tr><td>K=10</td><td>RBF</td><td>67.1554</td></tr><tr><td>K=100</td><td>Linear</td><td>67.1554</td></tr><tr><td>K=100</td><td>RBF</td><td>52.4927</td></tr></table></td><td><table><tr><td>Linear</td><td>68.9150</td></tr><tr><td>RBF</td><td>76.2463</td></tr></table></td></tr>
</table>

Observation:

- More dimension PCA increases the predictability of data.
- Gaussian Kernel give poor accuracies in case of PCA for K =10 , better results for K =100.
- Gaussian kernel gives poor results in LDA compared to Linear Kernel.