UNIVERSITY OF SOUTHERN CALIFORNIA, COMPUTER SCIENCE

Machine Learning: HomeWork 5

Rohit Kondekar
740-581-9473

November 30, 2014

1 Question: Principal Component Analysis

1.1 Deriving PCA in terms of minimum reconstruction error

1.1.1

This can be solved by working on single principal direction and then generalizing it. Let $U_j \epsilon R^D$ to denote the basis vector in jth principal direction. Let $x_i \epsilon R^D$ high dimension feature vector and $z_i \epsilon R^d$ low dimension feature vector. Let $\hat{z}_j \epsilon R^N$ to denote jth component of all low dimensional feature vector.

$$\begin{aligned}
J(U_1, z_1) &= \sum_{i=1}^{N} ||x_i - z_{i1} U_i||^2 \\
&= \sum_{i=1}^{N} (x_i - z_{i1} U_i)^T (x_i - z_{i1} U_i) \\
&= \sum_{i=1}^{N} (x_i^T x_i - 2 z_{i1} U_i^T x_i + z_{i1}^2) \qquad z_{i1} \text{ is a scalar} \\
\frac{\partial J(U_1, z_1)}{\partial z_{i1}} &= -2 U_1^T x_i + 2 z_{i1} = 0 \\
z_{i1} &= U_1^T x_i
\end{aligned}$$

1.1.2

Substituting value obtained in earlier eqution in the cost function.

$$\begin{aligned}
J &= \sum_{i=1}^{N} (x_i^T x_i - 2 z_{i1} U_i^T x_i + z_{i1}^2) \\
&= \sum_{i=1}^{N} (x_i^T x_i - 2 z_{i1}^2 + z_{i1}^2) \\
&= \sum_{i=1}^{N} (x_i^T x_i - z_{i1}^2)
\end{aligned}$$

Minimizing J is same as maximizing z_{i1}^2

$$\sum_{i=1}^{N} z_{i1}^2 = \sum_{i=1}^{N} U_1 x_i x_i^T U_i$$
$$= U_1^T \Sigma U_i$$

So the optimization equation is:

$$\text{MAX} \quad U_1 \Sigma U_i$$
$$\text{Constraint} \quad U_1^T U_1 = 1$$

Lagrange Multiplier:

$$L = U_1 \Sigma U_i + \lambda (U_1^T U_1 - 1)$$
$$\frac{\partial L}{\partial U_1} = 2\Sigma U_1 - 2\lambda_1 U_1 = 0$$
$$\Sigma U_1 = \lambda_1 U_1$$

This proves that U_1 is an eigen vector and U is matrix of eigen vectors of Σ

1.2 Projecting a Gaussian distribution

1.2.1

$x \sim N(0, \Sigma)$: x is a Gaussian Random Vector (Multivariate Gaussian) of D jointly Gaussian RVs.

$z = p^T x$: is a linear combination of D random variables. Which comes out to be Univariate Gaussian.
As z, is normally distributed with mean 0, the value of variance would be equal to the second order moment.

$$\sigma_z^2 = E[z^2] = E[p^T X X^T p]$$
$$= p^T \Sigma p$$

Therefore, z is Normally distributed with $N(0, p^T \Sigma p)$

Entropy of z is given by :

$$H(z) = -\int_{-\infty}^{-\infty} f(z) \log\left(\frac{e^{\frac{-z^2}{2\sigma^2}}}{\sqrt{2\Pi\sigma^2}}\right) \quad \textit{where } \sigma^2 \textit{ is given by } p^T \Sigma p$$
$$= \int_{-\infty}^{-\infty} f(z) \log(\sqrt{2\Pi\sigma^2}) dz + \int_{-\infty}^{-\infty} f(z) \frac{z^2}{2\sigma^2} \log_2(e) dz$$
$$H(z) = \frac{1}{2} \log(2\Pi\sigma^2) + \frac{1}{2} \log_2(e)$$

So we need to maximize entropy given the constrain that $p^T p = 1$. Therefore optimization equation with langrange multipler:

$$L = \frac{1}{2} \log(2\Pi p^T \Sigma p) + \frac{1}{2} \log_2(e) + \lambda (p^T p - 1)$$
$$\frac{\partial L}{\partial p} = \frac{\Sigma p}{p^T \Sigma p} + 2\lambda p = 0$$
$$\Sigma p = (\lambda^{'} p^T \Sigma p) p \quad \textit{As } p^T \Sigma p \textit{ is scalar}$$
$$\Sigma p = \lambda^{''} p$$

So here we can see that to maximize Entropy of z, p^* should be eigen vectors of Σ.

1.2.2

As from the equation in the first part:

$$H(z) = \frac{1}{2}\log(2\Pi\sigma^2) + \frac{1}{2}\log_2(e)$$

Maximizing Entropy is same is maximizing Variance σ_z^2 which we have obtained in the first part. So by maximizing variance we are maximing the entropy. Therefore p^* maximizes variance of z.

2 Hidden Markov Models

Transition Probabilities:

	S1	**S2**
S1	0.7	0.3
S2	0.3	0.7

Emission Probabilities:

	A	**C**	**G**	**T**
S1	0.4	0.1	0.4	0.1
S2	0.1	0.4	0.1	0.4

Initial State Distribution π:

$\pi_1 = 0.5$
$\pi_2 = 0.5$

Given Sequence: e = CGTCAG

2.1

Forward Probabilities:

C:
$\alpha_1(S_1) = 0.5 * 0.1 = 0.05$
$\alpha_1(S_2) = 0.5 * 0.4 = 0.20$

G:
$\alpha_2(S_1) = (0.05 * 0.7 * 0.4) + (0.20 * 0.3 * 0.4) = 0.038$
$\alpha_2(S_2) = (0.05 * 0.3 * 0.1) + (0.20 * 0.7 * 0.1) = 0.0155$

T:
$\alpha_3(S_1) = (0.038 * 0.7 * 0.1) + (0.0155 * 0.3 * 0.1) = 0.003125$
$\alpha_3(S_2) = (0.038 * 0.3 * 0.4) + (0.0155 * 0.7 * 0.4) = 0.0089$

C:
$\alpha_4(S_1) = (0.003125 * 0.7 * 0.1) + (0.0089 * 0.3 * 0.1) = 0.00048575$
$\alpha_4(S_2) = (0.003125 * 0.3 * 0.4) + (0.0089 * 0.7 * 0.4) = 0.002867$

A:
$\alpha_5(S_1) = (0.00048575 * 0.7 * 0.4) + (0.002867 * 0.3 * 0.4) = 0.00048005$
$\alpha_5(S_2) = (0.00048575 * 0.3 * 0.1) + (0.002867 * 0.7 * 0.1) = 0.0002152625$

G:
$\alpha_6(S_1) = (0.00048005 * 0.7 * 0.4) + (0.0002152625 * 0.3 * 0.4) = 0.0001602455$
$\alpha_6(S_2) = (0.00048005 * 0.3 * 0.1) + (0.0002152625 * 0.7 * 0.1) = 0.000029469875$

Alphas:

T	**1**	**2**	**3**	**4**	**5**	**6**
$\alpha_t(S1)$	0.05	0.038	0.003125	0.00048575	0.00048005	0.0001602455
$\alpha_t(S2)$	0.20	0.0155	0.0089	0.002867	0.0002152625	0.000029469875

Therefore, $P(e|\theta) = 0.0001602455 + 0.000029469875 = 0.000189715375$

2.2

Backward Probabilities:

$\beta_6(S_1) = 1$
$\beta_6(S_2) = 1$

G:
$\beta_5(S_1) = (1 * 0.4 * 0.7) + (1 * 0.1 * 0.3) = 0.31$
$\beta_5(S_2) = (1 * 0.4 * 0.3) + (1 * 0.1 * 0.7) = 0.19$

A:
$\beta_4(S_1) = (0.31 * 0.4 * 0.7) + (0.19 * 0.1 * 0.3) = 0.0925$
$\beta_4(S_2) = (0.31 * 0.4 * 0.3) + (0.19 * 0.1 * 0.7) = 0.0505$

C:
$\beta_3(S_1) = (0.0925 * 0.1 * 0.7) + (0.0505 * 0.4 * 0.3) = 0.012535$
$\beta_3(S_2) = (0.0925 * 0.1 * 0.3) + (0.0505 * 0.4 * 0.7) = 0.016915$

T:
$\beta_2(S_1) = (0.012535 * 0.1 * 0.7) + (0.016915 * 0.4 * 0.3) = 0.00290725$
$\beta_2(S_2) = (0.012535 * 0.1 * 0.3) + (0.016915 * 0.4 * 0.7) = 0.00511225$

G:
$\beta_1(S_1) = (0.00290725 * 0.4 * 0.7) + (0.00511225 * 0.1 * 0.3) = 0.0009673975$
$\beta_1(S_2) = (0.00290725 * 0.4 * 0.3) + (0.00511225 * 0.1 * 0.7) = 0.0007067275$

Beta:

T	**1**	**2**	**3**	**4**	**5**	**6**
$\beta_t(S1)$	0.0009673975	0.00290725	0.012535	0.0925	0.31	1
$\beta_t(S2)$	0.0007067275	0.00511225	0.016915	0.0505	0.19	1

Alphas:

T	**1**	**2**	**3**	**4**	**5**	**6**
$\alpha_t(S1)$	0.05	0.038	0.003125	0.00048575	0.00048005	0.0001602455
$\alpha_t(S2)$	0.20	0.0155	0.0089	0.002867	0.0002152625	0.000029469875

Gamma: $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_j \alpha_t(j)\beta_t(j)}$

Let Gamma' = $\gamma_t'(j) = \alpha_t(j)\beta_t(j)$

T	1	2	3	4	5	6
$\gamma_t'(S1)$	0.000048369875	0.000110475500	0.000039171875	0.000044931875	0.000148815500	0.000160245500
$\gamma_t'(S2)$	0.000141345500	0.000079239875	0.000150543500	0.000144783500	0.000040899875	0.000029469875

T	1	2	3	4	5	6
$P_t(S1)$	0.254960227	0.582322334	0.206477071	0.236838343	0.784414547	0.844662695
$P_t(S2)$	0.745039773	0.417677666	0.793522929	0.763161657	0.215585453	0.155337305

2.3

Veterbi Algorithm:

δ Matrix:

T	1	2	3	4	5	6
$\delta_t(S1)$	0.05	0.024	0.00168	1.176E-4	1.3171E-4	3.6879359E-5
$\delta_t(S2)$	0.2	0.01399	0.003919	0.001097599	7.68319E-5	5.378239E-6

Ψ Matrix:

T	1	2	3	4	5	6
$\Psi_t(S1)$	0.0	2.0	1.0	1.0	2.0	1.0
$\Psi_t(S2)$	0.0	2.0	2.0	2.0	2.0	2.0

Sequence Obtained by Veterbi Algorithm:
2,2,2,2,2,1

Sequence of most likely states estimated independently:
2,1,2,2,1,1

So they are not same, so it doesn't happen in this example.

```
//Viterbi.java
double[][] a = {{0,0,0},{0,0.7,0.3},{0,0.3,0.7}};
      double[][] b = {{0,0,0,0,0},{0,0.4,0.1,0.4,0.1},{0,0.1,0.4,0.1,0.4}};

      double[] pi = {0,0.5,0.5};

      double[][] delta = new double[3][7];
      double[][] si = new double[3][7];

      delta[1][1] = pi[1]*b[1][2];
      delta[2][1] = pi[2]*b[2][2];
      si[1][1] = 0;
      si[2][1] = 0;

      int[] val = {0,2,3,4,2,1,3};

      for(int t=2;t<=6;t++){

         for(int j=1;j<=2;j++){

            double val1 = delta[1][t-1]*a[1][j]*b[j][val[t]];
```

```
            double val2 = delta[2][t-1]*a[2][j]*b[j][val[t]];

            if(val1>val2){
               delta[j][t] = val1;
               si[j][t] = 1;
            }
            else{
               delta[j][t] = val2;
               si[j][t] = 2;
            }
         }

      }

      for (int i = 1; i < delta.length; i++) {
         for (int j = 1; j < delta[1].length; j++) {
            System.out.print(delta[i][j]+" , ");
         }
         System.out.println();
      }


      for (int i = 1; i < si.length; i++) {
         for (int j = 1; j < si[1].length; j++) {
            System.out.print(si[i][j]+" , ");
         }
         System.out.println();
      }
   }
```

2.4

Alphas:

T	1	2	3	4	5	6
$\alpha_t(S1)$	0.05	0.038	0.003125	0.00048575	0.00048005	0.0001602455
$\alpha_t(S2)$	0.20	0.0155	0.0089	0.002867	0.0002152625	0.000029469875

Forward Probabilities for all possible emissions:

For A:
$\alpha_{T+1}(S1) = (0.0001602455 * 0.7 * 0.4) + (0.000029469875 * 0.3 * 0.4) = 0.00004840512$
$\alpha_{T+1}(S2) = (0.0001602455 * 0.3 * 0.1) + (0.000029469875 * 0.7 * 0.1) = 0.00000687025$

For C:
$\alpha_{T+1}(S1) = (0.0001602455 * 0.7 * 0.1) + (0.000029469875 * 0.3 * 0.1) = 0.00001210128$
$\alpha_{T+1}(S2) = (0.0001602455 * 0.3 * 0.4) + (0.000029469875 * 0.7 * 0.4) = 0.00002748102$

For G:
$\alpha_{T+1}(S1) = (0.0001602455 * 0.7 * 0.4) + (0.000029469875 * 0.3 * 0.4) = 0.00004840512$
$\alpha_{T+1}(S2) = (0.0001602455 * 0.3 * 0.1) + (0.000029469875 * 0.7 * 0.1) = 0.00000687025$

For T:
$\alpha_{T+1}(S1) = (0.0001602455 * 0.7 * 0.1) + (0.000029469875 * 0.3 * 0.1) = 0.00001210128$
$\alpha_{T+1}(S2) = (0.0001602455 * 0.3 * 0.4) + (0.000029469875 * 0.7 * 0.4) = 0.00002748102$

P(A) = P(G) = 0.00005527537
P(C) = P(T) = 0.0000395823

Hence A or G are mostly likely to be the next symbol. (Equally Likely).

3 Sample questions

3.1 Expection Maximization

Deriving Generic EM Model to work on:

$$\sum_{i=1}^{N} \log p(x_i|\theta) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} p(x_i, z_k|\theta)$$

Let Q_i be distribution over z's s.t. $\sum_{k=1}^{K} Q_i(z_k) = 1$ and $Q_i >= 0$. Therefore:

$$\begin{aligned} &= \sum_{i=1}^{N} \log \sum_{k=1}^{K} Q_i(z_k) \frac{p(x_i, z_k|\theta)}{Q_i(z_k)} \\ &\geq \sum_{i=1}^{N} \sum_{k=1}^{K} Q_i(z_k) \log \frac{p(x_i, z_k|\theta)}{Q_i(z_k)} \quad \textit{Using Jenson's Inequality} \end{aligned}$$

For this to be a tight bound, $E[x] = x$ should be satisfied, i.e. x is constant. Therefore:
E Step:

$$\begin{aligned} \frac{p(x_i, z_k|\theta)}{Q_i(z_k)} &= c \\ Q_i(z_k) &= \frac{p(x_i, z_k|\theta)}{c} \\ Q_i(z_k) &= \frac{p(x_i, z_k|\theta)}{\sum_{k=1}^{K} p(x_i, z_k|\theta)} \qquad As \sum_{k=1}^{K} Q_i(z_k) = 1 \\ &= \frac{p(x_i, z_k|\theta)}{p(x_i|\theta)} \\ Q_i(z_k) &= p(z_k|x_i, \theta) \\ &= \frac{p(x_i|z_k, \theta) p(z_k|\theta)}{\sum_{k=1}^{K} p(x_i|z_k, \theta)} \qquad \textit{Bayes Rule} \end{aligned}$$

M Step:

$$\theta = \text{argmax}_\theta \sum_{i=1}^{N} \sum_{k=1}^{K} Q_i(z_k) \log \frac{p(x_i, z_k|\theta)}{Q_i(z_k)}$$

Gaussian Mixture Models:

E Step: $Q_i(z_k)$ is given by $p(z_k|x_i, \theta)$, the responsibility that cluster k takes data point i.

$$\begin{aligned} Q_i(z_k) &= \frac{p(x_i|z_k, \theta) p(z_k|\theta)}{\sum_{k=1}^{K} p(x_i|z_k, \theta)} \\ Q_i(z_k) = r_{ik} &= \frac{w_k N(x_i|\mu_k, \Sigma_k^2)}{\sum_{k'=1}^{K} w_{k'} N(x_i|\mu_{k'}, \Sigma_{k'}^2)} \end{aligned}$$

M Step:

$$\sum_{i=1}^{N}\sum_{k=1}^{K} Q_i(z_k)\log\frac{p(x_i,z_k|\theta)}{Q_i(z_k)} = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\log\frac{\frac{1}{(2\pi)^{n/2}|\Sigma_k^{-1}|^{1/2)}}exp(\frac{-1}{2}((x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)))w_k}{r_{ik}}$$

Solving for μ_k

$$\nabla_{\mu_k}(\sum_{i=1}^{N} -r_{ik}\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k))$$

$$\mu_k = \frac{\sum_{i=1}^{N} r_{ik}x_i}{r_k}$$

Solving for Σ_k

$$\nabla_{\Sigma_k}(\sum_{i=1}^{N} -r_{ik}\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k))$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} r_{ik}(x_i-\mu_k)(x_i-\mu_k)^T}{r_k}$$

Solving for w_k

$$w_k = \frac{1}{N}\sum_{i=1}^{N} r_{ik}$$

4 Dimensionality Reduction

4.1

In Cordinate Component Analysis, all the principal components are just the dimensions of the data. Whereas in PCA, the principal components are the eigen vectors of the covariance matrix. This can have a serious impact because, CCA may not find much covariance in direction of just the dimensions and may lead to no reduction in dimensionality, whereas in PCA it finds the component with the highest variance with restriction of they being orthogonal.

4.2

CCA and PCA may lead to same solution when the data is distributed just around certain dimension of data itself, thereby giving the principal component the dimension of data itself, in case of PCA.
E.g. 2 dimensional data:

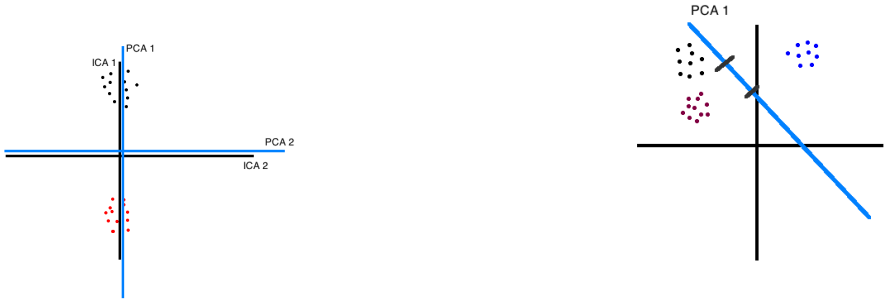

Figure 4.1: ICA and PCA comparison, second figure shows the disadvantage

5 Programming (PCA)

5.1

5.2

5.3

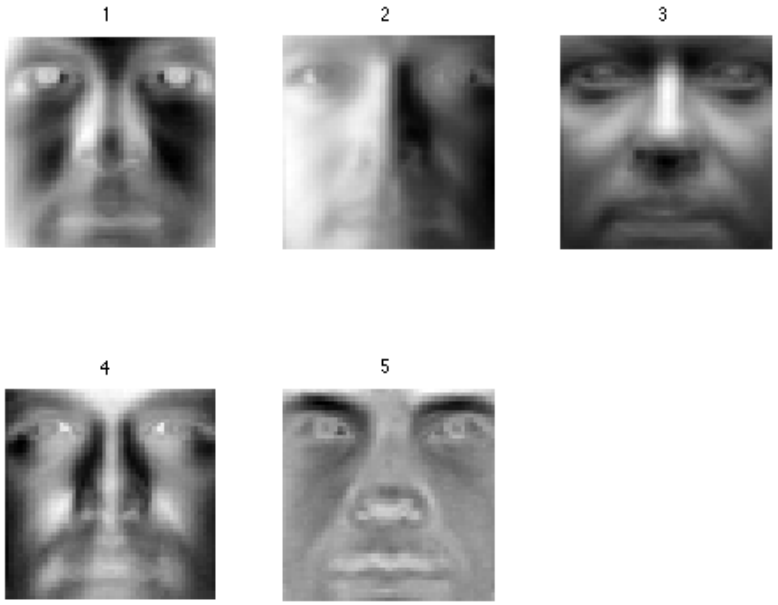

Figure 5.1: Eigen Faces

5.4

5.4.1 Linear SVM

Validation Linear SVM : Test Subset =1 d =20 C =4 Avg Validation Accuracy =83.1532
Validation Linear SVM : Test Subset =2 d =20 C =4 Avg Validation Accuracy =82.2948
Validation Linear SVM : Test Subset =3 d =20 C =4 Avg Validation Accuracy =82.2948
Validation Linear SVM : Test Subset =4 d =20 C =4 Avg Validation Accuracy =87.1178
Validation Linear SVM : Test Subset =5 d =20 C =4 Avg Validation Accuracy =85.8333
Test Linear SVM : d =20 Avg Test Accuracy =86.2356

Validation Linear SVM : Test Subset =1 d =50 C =0.25 Avg Validation Accuracy =88.1657
Validation Linear SVM : Test Subset =2 d =50 C =1 Avg Validation Accuracy =86.8609
Validation Linear SVM : Test Subset =3 d =50 C =1 Avg Validation Accuracy =86.8609
Validation Linear SVM : Test Subset =4 d =50 C =1 Avg Validation Accuracy =90.9085
Validation Linear SVM : Test Subset =5 d =50 C =16 Avg Validation Accuracy =92.5
Test Linear SVM : d =50 Avg Test Accuracy =90.7043

Validation Linear SVM : Test Subset =1 d =100 C =4 Avg Validation Accuracy =90.058
Validation Linear SVM : Test Subset =2 d =100 C =16 Avg Validation Accuracy =91.0056
Validation Linear SVM : Test Subset =3 d =100 C =16 Avg Validation Accuracy =91.0056
Validation Linear SVM : Test Subset =4 d =100 C =1 Avg Validation Accuracy =93.1015

Validation Linear SVM : Test Subset =5 d =100 C =16 Avg Validation Accuracy =93.6607
Test Linear SVM : d =100 Avg Test Accuracy =90.8333

Validation Linear SVM : Test Subset =1 d =200 C =16 Avg Validation Accuracy =90.0329
Validation Linear SVM : Test Subset =2 d =200 C =4 Avg Validation Accuracy =90.8741
Validation Linear SVM : Test Subset =3 d =200 C =4 Avg Validation Accuracy =90.8741
Validation Linear SVM : Test Subset =4 d =200 C =0.25 Avg Validation Accuracy =92.9527
Validation Linear SVM : Test Subset =5 d =200 C =16 Avg Validation Accuracy =94.2262
Test Linear SVM : d =200 Avg Test Accuracy =90.8271

5.4.2 RBF Kernerl SVM

Validation RBF SVM : Test Subset =1 d =20 C =4096 Gamma =6.1035e-05 Avg Validation Accuracy =82.8258
Validation RBF SVM : Test Subset =2 d =20 C =4096 Gamma =0.00024414 Avg Validation Accuracy =82.9198
Validation RBF SVM : Test Subset =3 d =20 C =4096 Gamma =0.00024414 Avg Validation Accuracy =82.9198
Validation RBF SVM : Test Subset =4 d =20 C =1024 Gamma =0.00097656 Avg Validation Accuracy =86.2077
Validation RBF SVM : Test Subset =5 d =20 C =1024 Gamma =0.0039062 Avg Validation Accuracy =86.5774
Test RBF Kernel SVM : d =20 Avg Test Accuracy =83.9825

Validation RBF SVM : Test Subset =1 d =50 C =1024 Gamma =6.1035e-05 Avg Validation Accuracy =88.1908
Validation RBF SVM : Test Subset =2 d =50 C =256 Gamma =0.0039062 Avg Validation Accuracy =88.6607
Validation RBF SVM : Test Subset =3 d =50 C =256 Gamma =0.0039062 Avg Validation Accuracy =88.6607
Validation RBF SVM : Test Subset =4 d =50 C =1024 Gamma =0.00097656 Avg Validation Accuracy =90.9085
Validation RBF SVM : Test Subset =5 d =50 C =256 Gamma =0.0039062 Avg Validation Accuracy =91.9643
Test RBF Kernel SVM : d =50 Avg Test Accuracy =87.6692

Validation RBF SVM : Test Subset =1 d =100 C =1024 Gamma =6.1035e-05 Avg Validation Accuracy =90.2146
Validation RBF SVM : Test Subset =2 d =100 C =16384 Gamma =6.1035e-05 Avg Validation Accuracy =90.8271
Validation RBF SVM : Test Subset =3 d =100 C =16384 Gamma =6.1035e-05 Avg Validation Accuracy =90.8271
Validation RBF SVM : Test Subset =4 d =100 C =1024 Gamma =0.00024414 Avg Validation Accuracy =93.1015
Validation RBF SVM : Test Subset =5 d =100 C =16384 Gamma =6.1035e-05 Avg Validation Accuracy =93.2738
Test RBF Kernel SVM : d =100 Avg Test Accuracy =91.2481

Validation RBF SVM : Test Subset =1 d =200 C =1024 Gamma =6.1035e-05 Avg Validation Accuracy =90.3462
Validation RBF SVM : Test Subset =2 d =200 C =4096 Gamma =6.1035e-05 Avg Validation Accuracy =90.3979
Validation RBF SVM : Test Subset =3 d =200 C =4096 Gamma =6.1035e-05 Avg Validation Accuracy =90.3979
Validation RBF SVM : Test Subset =4 d =200 C =1024 Gamma =0.00024414 Avg Validation Accuracy =93.0247
Validation RBF SVM : Test Subset =5 d =200 C =16384 Gamma =6.1035e-05 Avg Validation Accuracy =93.6607
Test RBF Kernel SVM : d =200 Avg Test Accuracy =90.9323

6 Programming (HMM)

6.1

6.2 Implement the Baum-Welch algorithm

6.3 Obtain parameter estimates

For Initial State Probability $\pi = [1, 0]$

Transition Matrix A:

T	S1	S2
S1	0.9305	0.0695
S2	0.0349	0.9651

Emission Matrix B:

T	1	2	3	4
S1	0.3711	0.1227	0.1290	0.3773
S2	0.0716	0.4465	0.4217	0.0602

For Initial State Probability $\pi = [0.5, 0.5]$

Transition Matrix A:

T	S1	S2
S1	0.7000	0.3000
S2	0.3000	0.7000

Emission Matrix B:

T	1	2	3	4
S1	0.3250	0.1725	0.1740	0.3285
S2	0.3250	0.1725	0.1740	0.3285

Using hmmtrain - Matlab Library

Transition Matrix A:

T	S1	S2
S1	0.9249	0.0751
S2	0.0844	0.9156

Emission Matrix B:

T	1	2	3	4
S1	0.3850	0.1048	0.1165	0.3937
S2	0.0891	0.4386	0.4003	0.0720

My implementation and matlab implementation are approximately similar for $\pi = [1, 0]$.