CSCI 567-Machine Learning Assignment-5

1 Principle Component Analysis

$$J(\vec{v}) = \sum_{i=1}^{n} ||\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v}||^2 \tag{1}$$

We want to minimize the loss.

$$\min J(\vec{v}) = \min \sum_{i=1}^{n} ||\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v}||^2$$

$$= \sum_{i=1}^{n} (\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v})^T(\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v})$$

$$= \sum_{i=1}^{n} (\vec{x}_i^T - ((\vec{v}^T\vec{x}_i)\vec{v})^T)(\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v})$$

$$= \sum_{i=1}^{n} (\vec{x}_i^T - \vec{v}^T\vec{x}_i^T\vec{v})(\vec{x}_i - (\vec{v}^T\vec{x}_i)\vec{v})$$

$$= \sum_{i=1}^{n} (\vec{x}_i^T\vec{x}_i - \vec{x}_i^T(\vec{v}^T\vec{x}_i)\vec{v} - \vec{v}^T\vec{x}_i^T\vec{v}\vec{x}_i + \vec{v}^T\vec{x}_i^T\vec{v}\vec{v}^T\vec{x}_i\vec{v})$$

$\vec{v}\vec{v}^T = 1$ is known. $\vec{x}_i^T\vec{x}_i$ is constant and doesn't depend on $\vec{v}$. Also, $\vec{v}^T\vec{x}_i = \vec{x}_i^T\vec{v}$ since it is scalar.

So, minimizing

$$\sum_{i=1}^{n} -\vec{x}_i^T(\vec{v}^T\vec{x}_i)\vec{v} - \vec{v}^T\vec{x}_i^T\vec{v}\vec{x}_i + \vec{v}^T\vec{x}_i^T\vec{x}_i\vec{v} = \sum_{i=1}^{n} -\vec{v}^T\vec{x}_i\vec{x}_i^T\vec{v}^T$$

is equals to maximizing

$$R(\vec{v}) = \vec{v}^T X X^T \vec{v}$$

2 HMM Algorithm

2.1 Forward Algorithm

$$\alpha_1(j) = \pi_j P(Y_1 = 0 | X_1 = S_j)$$

For $t > 1$

$$\alpha_t(j) = P(Y_t|X_t = S_j) \sum_i a_{ij}\alpha_{t-1}$$

Let's calculate α_1:

$$\alpha_1(1) = \pi_1 P(Y_1 = 0|X_1 = S_1) = 0.6 * 0.7 = 0.42$$

$$\alpha_1(2) = \pi_2 P(Y_1 = 0|X_1 = S_2) = 0.4 * 0.8 = 0.32$$

$$\alpha_2(1) = e_{1,1}[a_{1,1}\alpha_1(1) + a_{2,1}\alpha_1(2)] = 0.3 * [0.9 * 0.42 + 0.2 * 0.32] = 0.1326$$

$$\alpha_2(2) = e_{2,1}[a_{1,2}\alpha_1(1) + a_{2,2}\alpha_1(2)] = 0.2 * [0.1 * 0.42 + 0.8 * 0.32] = 0.0596$$

$$\alpha_3(1) = e_{1,0}[a_{1,1}\alpha_2(1)+a_{2,1}\alpha_2(2)] = 0.7*[0.9*0.1326+0.2*0.0596] = 0.091882$$

$$\alpha_3(2) = e_{2,0}[a_{1,2}\alpha_2(1)+a_{2,2}\alpha_2(2)] = 0.8*[0.1*0.1326+0.8*0.0596] = 0.048752$$

Sum of $\alpha_3(1)$ and $\alpha_3(2) = P(Y_1 = 0, Y_2 = 1, Y_3 = 0) = 0.140634$

2.2 Backward Algorithm

$$\beta_3(1) = 1$$

$$\beta_3(2) = 1$$

For $t < 3$:

$$\beta_{t-1}(i) = \sum_j \beta_t(j) a_{i,j} P(Y_{t+1}|X_{t+1} = S_j)$$

$$\beta_2(1) = \beta_3(1)a_{1,1}e_{1,0} + \beta_3(2)a_{1,2}e_{2,0} = 1 * 0.9 * 0.7 + 1 * 0.1 * 0.8 = 0.71$$

$$\beta_2(2) = \beta_3(1)a_{2,1}e_{1,0} + \beta_3(2)a_{2,2}e_{2,0} = 1 * 0.2 * 0.7 + 1 * 0.8 * 0.8 = 0.78$$

$$\beta_1(1) = \beta_2(1)a_{1,1}e_{1,1}+\beta_2(2)a_{1,2}e_{2,1} = 0.71*0.9*0.3+0.78*0.1*0.2 = 0.2073$$

$$\beta_1(2) = \beta_2(1)a_{2,1}e_{1,1}+\beta_2(2)a_{2,2}e_{2,1} = 0.71*0.2*0.3+0.78*0.8*0.2 = 0.1674$$

$$P(Y_1 = 0, Y_2 = 1, Y_3 = 0) = \alpha_1(1)\beta_1(1)+\alpha_1(2)\beta_1(2) = 0.42*0.2073+0.32*0.1674 = 0.140634$$

2.3 Results of Backward and Forward Algorithms

They both gives the same result, which shows they agree.

2.4 Most Likely Setting for Each State

For state 1, most likely setting is S_1:

$$\alpha_1(1) * \beta_1(1) = 0.42 * 0.2073 = 0.087066$$

$$\alpha_1(2) * \beta_1(2) = 0.32 * 0.1674 = 0.053568$$

For state 2, most likely setting is S_1:

$$\alpha_2(1) * \beta_2(1) = 0.1326 * 0.71 = 0.094146$$

$$\alpha_2(2) * \beta_2(2) = 0.0596 * 0.78 = 0.046488$$

For state 3, most likely setting is S_1:

$$\alpha_3(1) * \beta_3(1) = 0.091882 * 1 = 0.091882$$

$$\alpha_3(2) * \beta_3(2) = 0.048752 * 1 = 0.048752$$

2.5 Most Likely Setting for All States

For only state 1:

$$\delta_1(1) = \pi_1 e_{1,0} = 0.6 * 0.7 = 0.42$$

$$\delta_1(2) = \pi_2 e_{2,0} = 0.4 * 0.8 = 0.32$$

For state 2:

$$\delta_2(1) = \max_i \delta_1 i * a_{i,1} * e_{1,1} = max(\delta_1(1) * a_{1,1} * e_{1,1}, \delta_1(2) * a_{2,1} e_{1,1})$$

$$= max(0.42 * 0.9 * 0.3, 0.32 * 0.2 * 0.3) = 0.1134$$

$$\phi_2(1) = \arg\max_i \delta_1(i) a_{i,1} = 1$$

$$\delta_2(2) = max(\delta_1(1) * a_{1,2} * e_{2,1}, \delta_1(2) * a_{2,2} * e_{2,1})$$

$$= max(0.42 * 0.1 * 0.2, 0.32 * 0.8, 0.2) = 0.0512$$

$$\phi_2(2) = \arg\max_i \delta_1(i) a_{i,2} = 2$$

For state 3:

$$\delta_3(1) = max(\delta_2(1) * a_{1,1} * e_{1,0}, \delta_2(2) * a_{2,1} * e_{1,0})$$

$$= max(0.1134 * 0.9 * 0.7, 0.0512 * 0.2, 0.7) = 0.071442$$

$$\phi_3(1) = \arg\max_i \delta_2(i) a_{i,1} = 1$$

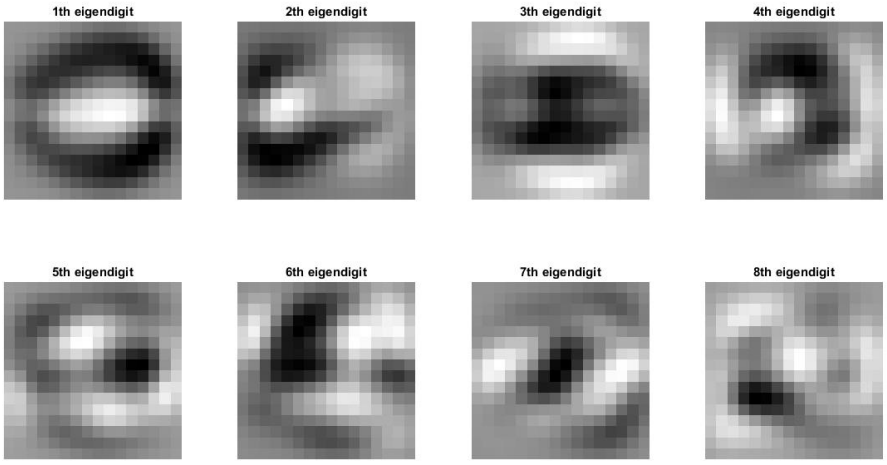
$$\delta_3(2) = max(\delta_2(1) * a_{1,2} * e_{2,0}, \delta_2(2) * a_{2,2} * e_{2,0})$$
$$= max(0.1134 * 0.1 * 0.8, 0.0512 * 0.8, 0.8) = 0.0327$$
$$\phi_3(2) = \arg\max_i \delta_2(i) a_{i,2} = 2$$

Since $\delta_3(1)$ is larger than $\delta_3(2)$, we choose the path that it suggests. The most possible setting for $X_1, X_2, and X_3$ is S_1, S_1, S_1.

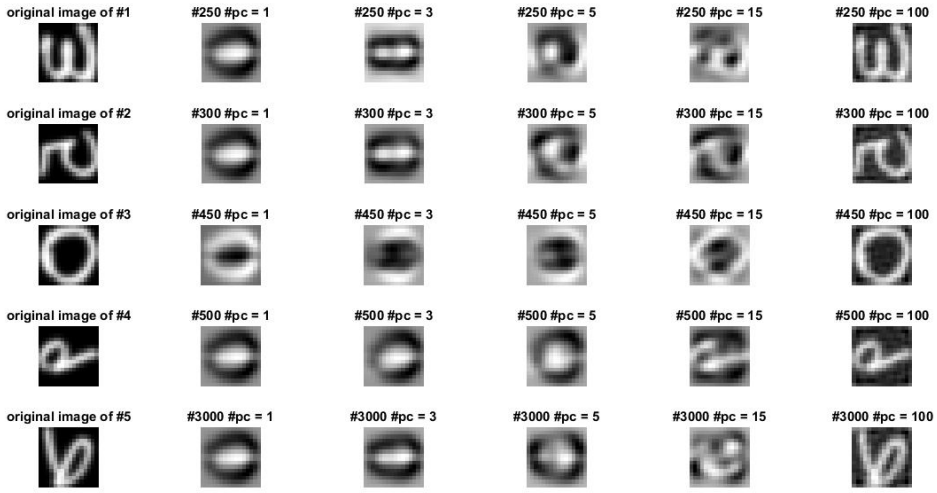
3 PCA Programming

3.1 Implementation of Eigenvectors Function

3.2 Reconstruction of 8 Top Eigendigits



3.3 Reconstruction of Some Digits



3.4 Classification on Compressed Data

K	Training Acc	Testing Acc	Time
1	0.6014	0.5473	0.4337
3	0.8134	0.7432	0.5207
5	0.9204	0.84	0.6358
15	0.9666	0.88	1.2236
100	0.9686	0.8795	8.7162

Table 1: Training and Testing Accuracy for Compressed Data

Accuracy increases while K is increasing. This can be explained by the loss ratio of compressed data. When we compress the data into smaller dimension, we make a greater loss, which results in a worse classification accuracy.

On the other hand, required time for computation increases with K. The reason behind this is that, classification trees need to evaluate all features to decide on the next partitioning. When we have more features, it requires larger time to evaluate the information gain on each of the features. So, larger K causes a longer computation time.