Machine Learning and Applications (MLAP) Open Examination 2016-2017

Exam Number: **Y1403115**

The EM algorithm

1 Task 1

For each state (square), the algorithm counts the number of times it was a starting state, the number of times it was transitioned to from each other state, and the number of times each reward was awarded in it. Counts are normalised so probabilities sum to 1.

2 Task 2

The implementation of the EM algorithm closely follows the steps outlined in Lecture 19.

1. First the parameters are initialised with uniform distributions. The alpha recursion and the average log-likelihood for each episode are computed iteratively using the forward algorithm.

$$\alpha^n(h_n) = p(v_t^n|h_t)\sum_{h_{t-1}} p(h_t|h_{t-1})\alpha^n(h_{t-1}) \qquad \alpha^n(h_1) = p(v_1^n|h_1)p(h_1) \tag{1}$$

$$Loglikelihood = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log\sum_{h_t}\alpha(h_t) \tag{2}$$

2. The beta recursion for each episode is computed iteratively using the backward algorithm.

$$\beta^n(h_{t-1}) = \sum_{h_t} p(v_t^n|h_t)p(h_t|h_{t-1})\beta^n(h_t) \qquad \beta^n h_T = 1 \tag{3}$$

3. When computing alpha and beta they are scaled in order to prevent numerical underflow [1].

$$\alpha^n(h_t) = \frac{\alpha^n(h_t)}{\sum\limits_{h_t}\alpha^n(h_t)} \qquad \beta^n(h_t) = \frac{\beta^n(h_t)}{\sum\limits_{h_t}\beta^n(h_t)} \tag{4}$$

4. Hidden state marginals γ and pairwise marginals ξ are computed for each episode.

$$\gamma^n = p(h_t|v_{1:T}^n) = \frac{\alpha^n(h_t)\beta^n(h_t)}{\sum\limits_{h'_t}\alpha^n(h'_t)\beta^n(h'_t)} \tag{5}$$

$$\xi^n = p(h_t, h_{t+1}|v^n_{1:T}) = \frac{\alpha^n(h_t)p(v^n_{t+1}|h_{t+1})p(h_{t+1}|h_t)\beta^n(h_{t+1})}{\sum\limits_{h'_t}\sum\limits_{h'_{t+1}} \alpha^n(h'_t)p(v^n_{t+1}|h'_{t+1})p(h'_{t+1}|h'_t)\beta^n(h'_{t+1})} \tag{6}$$

5. Initial, transition and emission probabilities are updated using hidden state marginals and pairwise marginals in three separate function. The parameter matrices are normalised in order for probabilities to sum to 1.

$$p^{new}(h_1) = \frac{1}{N}\sum_{n=1}^{N} \gamma^n(h_1) \tag{7}$$

$$p^{new}(h_{t+1}|h_t) = \frac{\sum\limits_{n=1}^{N}\sum\limits_{t=1}^{T_n-1} \xi^n(h_t, h_{t+1})}{\sum\limits_{n=1}^{N}\sum\limits_{t=1}^{T_n-1}\sum\limits_{h_{t+1}} \xi^n(h_t, h_{t+1})} \tag{8}$$

$$p^{new}(v_t = i|h_t) = \frac{\sum\limits_{n=1}^{N}\sum\limits_{t=1}^{T_n-1} I[v^n_t = i]\gamma^n(h_t)}{\sum\limits_{n=1}^{N}\sum\limits_{t=1}^{T_n-1} \gamma^n(h_t)} \tag{9}$$

6. If the change in log-likelihood is smaller than the threshold 0.01, the algorithm terminates, otherwise it proceeds with the next iteration.

3 Task 3

3.1 Difference in log-likelihood for different EM runs

The EM algorithm is guaranteed to converge to a local maximum of the likelihood. This means that the closest local maximum to the initial position, in the search space, is reached at each run. When parameters are initialised randomly, at each run the algorithm is likely to converge to a different local maximum, compared to previous runs, since the starting conditions are different.

3.1.1 Uniform vs random initialisation

When using uniform distributions for initial parameters of the EM algorithm, the transition and initial probabilities do not change. The EM algorithm ignores the fact that we have hidden variables and computes the maximum likelihood estimate of the reward, regardless of the state. This is because the priors for hidden variables are constant with respect to the posterior. On the other hand random initialisation of the algorithm results in evolution of initial and transition probabilities, because we are defining non-constant priors with respect to the posterior.

4 Task 4

Transition probabilities are initialised to 0, for transitions from non neighbouring squares and squares with walls between them.

Manifold learning

Algorithm Descriptions

4.1 Isomap

Isomap [2] aims to project a set of data points from the high dimensional space, in which they exist, to a lower dimensional representation, while preserving the global geodesic distances (distances between every two points) between them. The algorithm assumes data points are embedded in an intrinsically low dimensional manifold in the higher dimensional space. It starts by constructing a nearest neighbour graph of the data points, in which only data points which are close to each other are connected. Then it computes the euclidean distances between neighbouring data points. After that it computes the distances for all other data points by taking the shortest path between them in the graph. The computed distances are approximate geodesic distances in the manifold. In the last step the algorithm uses the MDS algorithm to find the lower dimensional representation of the data.

4.2 Laplacian Eigenmap

Laplacian Eigenmap [3] also projects data to a lower dimensional representation. In contrast to Isomap, Laplacian Eigenmap aims to preserve only local distance information. In the first step of the algorithm, the nearest neighbour graph is computed as in Isomap, with edges between points which are close to each other. The distances between neighbouring points are either set to 1 or computed from the euclidean distances, using a gaussian. The new coordinates for the data points are the eigenvector solutions (in order of their eigenvalues) to the generalised eigenvector problem for the graph Laplacian and diagonal weight matrices. The eigenvector with the smallest eigenvalue is excluded.

4.3 Locality Preserving Projection

Locality Preserving Projection [4] follows the same steps as Laplacian Eigenmap - constructing a nearest neighbour graph and computing the edge weights(distances). The generalised eigenvector problem is solved for the Laplacian and diagonal weight matrices of the graph, each pre and post multiplied by the matrix of data vectors and matrix of data vectors transposed, respectively. The first m rows of The resulting eigenvector matrix, multiplied by the a data vector, gives the new m-dimensional representation of the data vector. In contrast to Laplacian Eigenmap, this algorithm is linear and can be kernelised.

Data proximity graph

All three methods assume data lives on an intrinsically low dimensional manifold. In Isomap, a data proximity graph is used to approximate geodesic distances between each two data points on the manifold (global distances). In the other two approaches the graph is used to approximate distances only between points, that are close to each other (local distances). In Isomap the local distances in the graph are euclidean $d_{ij} = ||\vec{x}_i - \vec{x}_j||$. In the other two approaches a heat kernel is used for local distances $d_{ij} = \exp\left(-\frac{||\vec{x}_i - \vec{x}_j||^2}{t}\right)$ or they are simply set to 1.

5 Laplacian matrix and Fiedler vector

5.1 Laplacian matrix construction

A weighted graph $G(V, E)$ has vertices $v \in V$, edges between nodes i and j $e_{ij} \in V$ and weights of edges $w(e) \in \mathbb{R}$ (for an unweighted graph assume $w(e) = 1$). The adjacency matrix of a graph $G(V, E)$ is a square matrix defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E \quad \text{for } i, j \in V \\ 0 & otherwise \end{cases} \tag{10}$$

The degree matrix D of a graph is a diagonal matrix, which contains the degree(number of edges connected to) each node along the diagonal:

$$D_{ii} = \sum_{j}^{N} A_{ij} \quad \text{for } \forall i, \forall j \in V \qquad D_{ij} = 0 \quad \text{for } \forall i, \forall j \in V, \text{ where } i \neq j \tag{11}$$

The weight matrix W and diagonal weight matrix D' of $G(V, E)$ are defined in the following two equations:

$$W_{ij} = \begin{cases} w(e_{ij}) & \text{if } e_{ij} \in E \quad \text{for } i \in V, j \in V \\ 0 & otherwise \end{cases} \tag{12}$$

$$D'_{ii} = \sum_{j}^{N} W_{ij} \quad \text{for } \forall i, \forall j \in V \qquad D'_{ij} = 0 \quad \text{for } \forall i, \forall j \in V, \text{ where } i \neq j \tag{13}$$

The Laplacian matrix L of a weighted graph is:

$$L = D' - W \tag{14}$$

The Laplacian matrix L of an unweighted graph

$$L = D' - W = D - A \tag{15}$$

A derviation of the Laplacian of the unweighted undirected graph in figure 1 is shown in equation 16. The example was taken from [5].

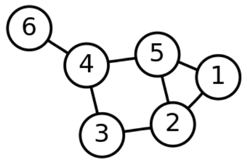

Figure 1: Unweighted Graph

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad L = D - A = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

5.2 Significance of Fiedler vector

The second smallest eigenvalue of the Laplacian matrix of a graph is known as the algebraic connectivity of the graph [6]. It is greater than zero iff the graph is connected. The eigenvector corresponding to the algebraic connectivity is known as the Fiedler vector [7]. Groups of values of the components of the Fiedler vector which are close to each other naturally represent clusters in the graph. This is why it has been used in various clustering and graph partition algorithms.

6 Algorithm

Given a set of data $X = \{\vec{x}_1, \dots \vec{x}_N\}$ where $\vec{x}_i$ is the i-th vector which is of length d and there are N such vectors, the three algorithms go through the following steps:

6.1 Isomap

1. Initialise nearest neighbour graph $G(V, E)$, with weight matrix W, with $V = \{1, \dots, N\}$ and $E = \emptyset$.

2. Construct an edge between vertices i and j if $\vec{x}_i$ and $\vec{x}_j$ are "close" to each other, using one of the following methods:

   (a) For each pair of nodes $i, j \in V$, construct an edge between i and j iff i is among the k nearest neighbours of j or j is among the k nearest neighbours of i, where k is a user defined constant and $i \neq j$.

   (b) For each pair of nodes $i, j \in V$, construct an edge between i and j iff $||\vec{x}_i - \vec{x}_j|| < \epsilon$, where ϵ is a user defined constant and $i \neq j$.

3. Set weights of edges to euclidean distances.

$$W_{ij} = \begin{cases} ||\vec{x}_i - \vec{x}_j|| & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

4. Initialise distance matrix D.

$$D_{ij} = \begin{cases} W_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases} \tag{18}$$

5. Replace all distances in D with their shortest paths in G.

$$D_{ij} = \min(D_{ij}, e_{ik} + e_{kj}) \quad \text{for } k = 1, 2, \dots N, \text{ where } e_{ik}, e_{kj} \in E \tag{19}$$

6. Compute the squared distance matrix S.

$$S_{ij} = D_{ij}^2 \quad \text{for } \forall i \in [1; N], \forall j \in [1; N] \tag{20}$$

7. Compute kernel matrix K using equation 21, where I is the identity matrix and J is the matrix of all ones.

$$K = \frac{(I - \frac{J}{N})S(I - \frac{J}{N})}{2} \tag{21}$$

8. Compute the eigenvalues and eigenvectors of K.

$$K = U \Lambda U^T \tag{22}$$

9. Assuming the resulting eigenvectors are $\vec{u}_1, \dots \vec{u}_N$, where $\vec{u}_l(i)$ is the i-th component of the l-th vector, and the corresponding eigenvalues, ordered in decreasing order, are $\lambda_1 > \lambda_2 > \dots > \lambda_N$, the new m-dimensional representation of each vector $\vec{x}_i \in X$ is $\vec{y}_i$ and is given by:

$$\vec{y}_i(l) = \sqrt{\lambda_l}\vec{u}_l(i) \quad \text{for } l = 1, \dots m, \text{ for } i = 1, \dots N \quad \text{where } m < d \tag{23}$$

6.2 Laplacian Eigenmap

1. Initialise nearest neighbour graph $G(V, E)$, with weight matrix W, with $V = \{1, \dots, N\}$ and $E = \emptyset$.

2. Construct an edge between vertices i and j if $\vec{x}_i$ and $\vec{x}_j$ are "close" to each other, using either k nearest neighbours or ϵ-neighbourhoods algorithms described in step 2a and step 2b of the Isomap algorithm (section 6.1).

   (a) **K nearest neighbours**
   For each pair of nodes $i, j \in V$, construct an edge between i and j iff i is among the k nearest neighbours of j or j is among the k nearest neighbours of i, where k is a user defined constant and $i \neq j$.

   (b) **ϵ-neighbourhoods**
   For each pair of nodes $i, j \in V$, construct an edge between i and j iff $||\vec{x}_i - \vec{x}_j||^2 < \epsilon$, where ϵ is a user defined constant and $i \neq j$.

3. Set weights of the created edges using heat kernel or simple methods described below.

   (a) **Heat kernel**

   $$W_{ij} = \exp\Big(-\frac{||\vec{x}_i - \vec{x}_j||^2}{t}\Big) \tag{24}$$

   t is a user defined constant.

   (b) **Simple**

   $$W_{ij} = 1 \tag{25}$$

4. If the graph G is not connected proceed with the next steps of the algorithm for each connected part of G. The number of connected disjoint parts of a graph correspond to the number of zero eigenvalues of the Laplacian matrix of the graph (see section 5.2).

5. Compute the Laplacian matrix $L = D - W$ of the graph as described in section 5.1, using the weight matrix W and the diagonal weight matrix D, where $D_{ii} = \sum_{j=1}^{N} W_{ji}$.

6. Solve the generalised eigenvector problem for L and D.

$$L\Phi = \Lambda D \Phi \tag{26}$$

7. Assuming the resulting eigenvectors are $\vec{\phi}_0, \dots \vec{\phi}_{N-1}$, where $\vec{\phi}_k(i)$ is the i-th component of the k-th vector, and they are ordered in ascending order of their eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1}$, the new m-dimensional representation of each vector $\vec{x}_i \in X$ is $\vec{y}_i$ and is given by:

$$y_i(k) = \phi_k(i) \quad \text{for } k = 1, \dots m \quad \text{for } i = 1, \dots N \tag{27}$$

   Eigenvector $\vec{\phi}_0$ is discarded and the next m eigenvectors are used.

6.3 Locality Preserving Projection

1. Initialise nearest neighbour graph $G(V, E)$, with weight matrix W, with $V = \{1, \dots, N\}$ and $E = \emptyset$.

2. Construct an edge between vertices i and j if $\vec{x}_i$ and $\vec{x}_j$ are "close" to each other, using either k nearest neighbours or ϵ-neighbourhoods algorithms described in step 2a and step 2b of the Laplacian eigenmap algorithm (section 6.2).

3. Set weights of the created edges using heat kernel or simple methods described in step 3a and step 3b of the Laplacian eigenmap algorithm (section 6.2).

4. Compute the Laplacian matrix $L = D - W$ of the graph as described in section 5.1, using the weight matrix W and diagonal weight matrix D, where $D_{ii} = \sum_{j=1}^{N} W_{ji}$.

5. Solve the generalised eigenvector problem for $\boldsymbol{X}L\boldsymbol{X}^T$ and $\boldsymbol{X}D\boldsymbol{X}^T$, where $x_i \in X$ is the i-th column of matrix $\boldsymbol{X}$.
$$\boldsymbol{X}L\boldsymbol{X}^T\Phi = \Lambda\boldsymbol{X}D\boldsymbol{X}^T\Phi \tag{28}$$

6. Assuming $\phi_0, \dots \phi_{N-1}$ are the column eigenvector solutions to equation 28, in ascending order of their eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_N$, the new m-dimensional coordinates vector $\vec{y}$, for a given coordinate vector $\vec{x}$ in the original d-dimensional space, is:
$$\vec{y} = A^T\vec{x} \quad \text{where } A = (\phi_0, \dots, \phi_{m-1}) \tag{29}$$

7 Face recognition

All three methods perform dimensionality reduction by preserving intrinsic properties of the manifold, on which the original data lies. Research has shown that face data in images possibly resides on an intrinsically low dimensional manifold in the high dimensional image space [8] [9] [10], which means that the nonlinear manifold structure preserving propertiy of the three algorithms is an advantage for the problem of face recognotion[11]. However as [11] and [4] state, Isomap and Laplacian eigenmap methods define a mapping only for the training data points, which means they can not be used to recognise new images without modification or extensions. LLP on the other hand defines a mapping for unobserved data points, which makes it more suitable to use as a face recognition algorithm.

The Isomap algorithm is developed based on a reconstruction principle and preserves global distance information, which is a disadvantage in classification problems, where maximum class separation is more important than true distance information [12]. Furthermore, Isomap requires a big training sample, with sample points smoothely distributed on the embedding manifold, in order to create an accurate aproximation of geodesic distances the manifold and this is rarely the case in face image data sets [12].

LPP is a linearised version of Laplacian eigenmap [13], which, in contrast to the latter, defines a mapping for unseen data points [11][4]. These properties of LPP make it the algorithm of choice for face recognition. LPP, however, requires the number of training samples (images) to be greater than the dimensions of of the samples (number of pixels) [14] [11] [15]. A further drawback of LPP is that the solution to the eigenvalue problem (equation 28) might be sub-optimal for representing the local structure of the manifold if there are zero eigenvalues [14]. Also according to [16], LPP deemphasises discriminant information between clusters, which is a disadvantage in classification problems such as face recognition.

There are several modifications and extensions of LPP for face recognition [14] [15] [16] [17] [18] [19] [20] which aim to overcome some of its limitations.

References

[1] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[2] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, pp. 1373–1396, June 2003.

[4] H. Xiaofei, *Locality Preserving Projections.* PhD thesis, Chicago, IL, USA, 2005. AAI3195015.

[5] "Laplacian matrix - wikipedia." `https://en.wikipedia.org/wiki/Laplacian_matrix`. (Accessed on 05/07/2017).

[6] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[7] M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematical Journal*, vol. 25, no. 4, pp. 619–633, 1975.

[8] Y. Chang, C. Hu, and M. Turk, "Manifold of facial expression," in *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, AMFG '03, (Washington, DC, USA), pp. 28–, IEEE Computer Society, 2003.

[9] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman, "Video-based face recognition using probabilistic appearance manifolds," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, pp. I–313–I–320 vol.1, June 2003.

[10] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[11] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face recognition using laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 328–340, Mar. 2005.

[12] M.-H. Yang, "Face recognition using extended isomap," in *Proceedings. International Conference on Image Processing*, vol. 2, pp. II–117–II–120 vol.2, 2002.

[13] L. Qiao, S. Chen, and X. Tan, "Sparsity preserving projections with applications to face recognition," *Pattern Recognition*, vol. 43, no. 1, pp. 331 – 341, 2010.

[14] Y. Xu, A. Zhong, J. Yang, and D. Zhang, "{LPP} solution schemes for use with face recognition," *Pattern Recognition*, vol. 43, no. 12, pp. 4165 – 4176, 2010.

[15] D. Cai, X. He, J. Han, and H. J. Zhang, "Orthogonal laplacianfaces for face recognition," *IEEE Transactions on Image Processing*, vol. 15, pp. 3608–3614, Nov 2006.

[16] W. Yu, X. Teng, and C. Liu, "Face recognition using discriminant locality preserving projections," *Image and Vision Computing*, vol. 24, no. 3, pp. 239 – 248, 2006.

[17] X. He, S. Yan, Y. Hu, and H.-J. Zhang, “Learning a locality preserving subspace for visual recognition,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV ’03, (Washington, DC, USA), pp. 385–, IEEE Computer Society, 2003.

[18] Z. Zheng, F. Yang, W. Tan, J. Jia, and J. Yang, “Gabor feature-based face recognition using supervised locality preserving projection,” *Signal Processing*, vol. 87, no. 10, pp. 2473 – 2483, 2007. Special Section: Total Least Squares and Errors-in-Variables Modeling.

[19] R. Zhi and Q. Ruan, “Facial expression recognition based on two-dimensional discriminant locality preserving projections,” *Neurocomputing*, vol. 71, no. 79, pp. 1730 – 1734, 2008. Progress in Modeling, Theory, and Application of Computational Intelligenc15th European Symposium on Artificial Neural Networks 200715th European Symposium on Artificial Neural Networks 2007.

[20] J. Cheng, Q. Liu, H. Lu, and Y.-W. Chen, “Supervised kernel locality preserving projections for face recognition,” *Neurocomputing*, vol. 67, pp. 443 – 449, 2005. Geometrical Methods in Neural Networks and LearningGeometrical Methods in Neural Networks and Learning.