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].

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

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

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

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

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

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

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

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

6. If the change in log-likelihood is smaller than the threashold 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 Difference in behaviour of EM when using random versus uniform parameter 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. On the other hand random initialisation of the algorithm results in evolution of initial and transition probabilities, because we are defining non-constant priors for them.

4 Task 4

Manifold learning

Construction/Implementation

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Data proximity graphs

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Algorithm

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Face recognition

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.