Hidden Markov Models Implementation

Goal

We are provided with a training set as well as a testing set, 'EMGaussian.data' and 'EMGaussian.test'. Here we are using an HMM model to account for a possible temporal structure within the data. We consider this data of the form $u_t = (x_t, y_t)$ where $u_t \in \mathbb{R}^2$ for $t = 1, ..., T$. The goal of the following implementation is to implement the probabilistic inference algorithm and the EM algorithm to learn parameters, as well as the Viterbi algorithm.

In this case, the following HMM model is considered : the chain (q_t) has $K = 4$ possible states, with an initial probability distribution $\pi \in \mathbb{R}^4$ and a probability transition matrix $A \in \mathbb{R}^{4\times4}$. Conditionally on the current states, observations are obtained from Gaussian emission probabilities $u_t|q_t = i \sim \mathcal{N}(\mu_i, \Sigma_i)$.

Functions

— `Kmeans.m` and `Kmeans_runs.m` implement the K-means algorithm, and allow to initialize centers and covariance matrices for the EM algorithm

— `EM.m` implements the EM algorithm with general covariance matrices. The `EM(X,K,runs)` function takes as inputs a sample dataset, a number of clusteres, and a number of runs (for the Kmeans initialization), and returns `[Pi_clust,q,mu,sigma,Lc]` : probabilities for each cluster, latent variables, clusters centers, covariance matrices and complete likelihood.

— `log_alpha.m, log_beta.m, log_gamma.m, log_eps.m` implement the α and β recursions, and compute the posterior probabilities $\gamma(q_t)$ et $\xi(q_t, q_{t+1})$. We have the following equations :

$$\alpha_t(q_t) = p(u_t|q_t) \sum_{q_{t-1}} p(q_t|q_{t-1})\alpha_{t-1}(q_{t-1}) \text{ } \textit{forward procedure}$$

$$\beta_t(q_t) = \sum_{q_{t+1}} p(q_{t+1}|q_t)p(u_{t+1}|q_{t+1})\beta_{t+1}(q_{t+1}) \text{ } \textit{backward procedure}$$

with the following initialization

$$\alpha_1(q_1) = p(u_1|q_1)\pi_0(q_1)$$
$$\beta_T(q_T) = 1$$

Since

$$p(q_t|u_1,...,u_T) = \gamma_t(q_t) = \frac{\alpha_t(q_t)\beta_t(q_t)}{\sum_{q_t}\alpha_t(q_t)\beta_t(q_t)}$$

the posterior probabilities can be obtained with the `log_gamma` function. In the implementation, logarithmic values have been computed, since values are too small to be computed as such. We have therefore used the following :

$$\log\left(\sum_i e^{\log(a_i)}\right) = \max_i \log(a_i) + \log\left(\sum_i e^{\log(a_i)-\max_i \log(a_i)}\right)$$

Coocurrences probabilities $\xi_t(q_t, q_{t+1})$ (logarithmic values) can be computed in the `log_eps` function with the following equation :

$$p(q_t, q_{t+1}|u_1,...,u_T) = \xi_t(q_t, q_{t+1}) = \frac{\alpha_t(q_t)p(u_{t+1}|q_{t+1})\gamma_{t+1}(q_{t+1})A_{q_t,q_{t+1}}}{\alpha_{t+1}(q_{t+1})}$$

— `HMM_EM.m` introduces a

`HMM_EM(nb_iterations,X,K,mu_init,sigma_init,A_init,pi_0_init)`

function that implements the HMM algorithm. This function takes as inputs the number of iterations of the algorithm, a sample dataset, a number of clusters, initial values for clusters centers, covariance matrices, a transition matrix and initial probability distribution, and returns

`[pi_0,A,mu,sigma,loglikelihood]`

. i.e the different model parameters optimized with Expectation-Maximization :

$$\pi_0, A, (\mu_k)_k, (\Sigma_k)_k, \left(p(y|\theta^{(t)})\right)_{1\leq t\leq T}$$

as well as log-likelihood values for the observations.

— `Viterbi.m` introduces a `Viterbi(X,K,mu,sigma,A,pi_0)` function which takes as inputs a sample dataset, a number of clusters and the model parameters (optimized with HMM), and returns `Seq`, the most probable sequence of states given the observations.

— `main.m` is the main file : it imports training and testing set and calls the functions described before.

Viterbi algorithm

The Viterbi algorithm can be implemented as follows : setting

$$v_i(t) = \max_{q_1,...,q_{t-1}} p(q_1, ..., q_{t-1}, q_t = i, u_1, ..., u_T) \quad \forall t, i = 1, ..., 4$$

and using the graphical model structure, we obtain the following recursive equation :

$$v_j(t) = \max_{1 \leq i \leq 4} [v_i(t-1)A_{ij}]\, p(u_t|q_t = j) \quad \forall t, j = 1, ..., 4$$

If we set :

$$s_{t+1}(j) = \text{argmax}_{1 \leq i \leq 4} [v_i(t)A_{ij}]$$

where $s_{t+1}(j)$ is the state at step t for which a transition towards state j maximizes the $v_j(t+1)$ probability, then the sequence of states leading to the maximum probability $p^\star = \max_{1 \leq i \leq 4} v_i(T)$ can be found with backward procedure using :

$$q_T = \text{argmax}_{1 \leq i \leq 4} v_i(T)$$
$$q_t^\star = s_{t+1}(q_{t+1}^\star) \quad \forall t \leq T-1$$

The most probable sequence of states given the observations can eventually be computed with :

$$q_1^\star, ..., q_T^\star = \underset{q_1,...,q_T}{\text{argmax}}\, p(q_1, ..., q_T, u_1, ..., u_T) = \underset{q_1,...,q_T}{\text{argmax}}\, p(q_1, ..., q_T|u_1, ..., u_T)$$