Expectation Maximisation

Petar Veličković

29 November 2016

Framework

The Expectation Maximisation (EM) algorithm is a procedure that iteratively optimises parameters of a given model, to maximise the likelihood of observing a given (training) dataset. Assuming that our framework has *unobserved data*, X, *observed data*, Y, *parameters* Θ, and a *likelihood function* $L(X, Y, \Theta) = \mathbb{P}(X, Y|\Theta)$, we can derive the steps of the algorithm as follows:

1. Choose initial parameters, Θ_0, at random.
2. (E step) Compute the expression for the expected value of the likelihood, where the unobserved data's probability distribution is derived conditional on the observed data and current parameters:

$$\mathbb{L}(\Theta|\Theta_t) = \mathbb{E}_{X|Y,\Theta_t}(L(X, Y, \Theta)) = \sum_X L(X, Y, \Theta)\mathbb{P}(X|Y, \Theta_t)$$

3. (M step) Find the parameters maximising this expectation, and update the parameters to them:

$$\Theta_{t+1} \leftarrow \underset{\Theta'}{\operatorname{argmax}}\, \mathbb{L}(\Theta'|\Theta_t)$$

4. Repeat steps 2–3 until convergence.

This algorithm is guaranteed to find a local maximum of the likelihood function, in a highly general fashion—the exact computations performed by the E/M step may highly vary depending on the problem at hand. Therefore, it can be applied successfully to a wide variety of situations. Here I will outline three applications of particular interest to the Bioinformatics course: *flipping two biased coins*, *soft k-means clustering* and *HMM training*.

Coin-flipping

In this basic setup, our (observed) dataset consists of a set of sequences of coin-flips, such that each sequence was made with one of two coins. As we assume

the coin flips within a single sequence to be IID, I will aggregate each of the sequences into two numbers: ${}^iY_H$ and ${}^iY_T$, giving the number of heads and tails in the ith sequence, respectively. The unobserved variables here correspond to the actual coin choice for each sequence—we can denote X_i as the probability that coin 1 was used for the ith sequence (the probability that coin 2 was used is therefore $1 - X_i$). Finally, the parameters of the model to optimise are $\Theta = (\theta_1, \theta_2)$, giving the probability of heads for each of the coins.

Under these conditions, our likelihood function is easily definable as

$$L(X, Y, \Theta) = \sum_{i=1}^{n} X_i(\theta_1^{{}^iY_H}(1-\theta_1)^{{}^iY_T}) + (1 - X_i)(\theta_2^{{}^iY_H}(1-\theta_2)^{{}^iY_T})$$

where n is the number of sequences in our training set.

For the E step, we need to first find the conditional probability of X given Y and Θ_t (which will scale the likelihood function for different choices of X). To do this, apply Bayes' theorem:

$$\mathbb{P}(X|Y, \Theta_t) = \frac{\mathbb{P}(Y|X, \Theta_t)\mathbb{P}(X|\Theta_t)}{\mathbb{P}(Y|\Theta_t)} \propto \mathbb{P}(Y|X, \Theta_t)\mathbb{P}(X|\Theta_t)$$

Here we omit the denominator as it will be the same for all choices of X (standard trick when applying Bayes' rule). We can just compute the numerator and subsequently normalise across all choices of X (for a particular sequence, the choices of X_i are 0 (coin 2) and 1 (coin 1)). Finally, as we assume a *uniform prior* (i.e. we are not biased towards choosing a particular coin), we conclude

$$\mathbb{P}(X|Y, \Theta_t) \propto \mathbb{P}(Y|X, \Theta_t)$$

However, this quantity is easily computable, as Y is given:

$$\begin{aligned}
\mathbb{P}(Y|X, \Theta_t) &= \sum_{i=1}^{n} \mathbb{P}({}^iY|X_i, \Theta_t) \\
&= \sum_{i=1}^{n} \mathbb{I}(X_i = 1)(\theta_{1t}^{{}^iY_H}(1-\theta_{1t})^{{}^iY_T}) + \mathbb{I}(X_i = 0)(\theta_{2t}^{{}^iY_H}(1-\theta_{2t})^{{}^iY_T})
\end{aligned}$$

(in fact, due to the given assumptions on the prior, this is the same form as plugging in X as a deterministic RV in $L(X, Y, \Theta)$!)

Now we can use this to compute $\mathbb{P}({}^iY|X_i = 1, \Theta_t)$ and $\mathbb{P}({}^iY|X_i = 0, \Theta_t)$—normalising them gives us the current estimates for X_i and $1 - X_i$ (corresponding to rows of *HiddenMatrix* in the lecture notes!). This fully specifies our conditional expected likelihood $\mathbb{L}(\Theta|\Theta_t)$, and therefore concludes the E step.

For the M step, we need to find a choice of Θ' to maximise this quantity. However, this is simple to do:

- For each sequence, we can take the (frequentist) probability of flipping heads, i.e.
$$\mathbb{P}({}^iY_j = H) = \frac{{}^iY_H}{{}^iY_H + {}^iY_T}$$
as the most likely value of θ associated with it.
- Once we have this for each sequence, we may then assign a fraction of this amount to each of the two coins, depending on how likely it is that they have produced this sequence under the current parameters (i.e. the entries of HiddenMatrix).

This amounts to the following formulae

$$\theta_1' = \frac{\sum_{i=1}^{n} X_i \mathbb{P}({}^iY_j = H)}{\sum_{i=1}^{n} X_i}$$

$$\theta_2' = \frac{\sum_{i=1}^{n} (1 - X_i) \mathbb{P}({}^iY_j = H)}{\sum_{i=1}^{n} (1 - X_i)}$$

and concludes the M step.

Soft k-means clustering

In the soft clustering setup, we would like to assign given data points in a high-dimensional space to k centroids. Unlike the *hard clustering* scenario, here the points are not committed to a single centroid—rather, each point is assigned a *probability distribution* over the centroids, determining the degree of "responsibility" of each centroid towards this data point. We can map this setup to the EM framework as follows:

- The centroids are the *parameters*, $\Theta = (\theta_1, \ldots, \theta_k)$, of the model[1];
- The observed data, Y, are the data points themselves;
- The assignment of points to centroids correspond to the unobserved data (e.g. $X_i = j$ if point Y_i is assigned to centroid θ_j).

In a similar vein to introducing a distance metric $d(A, B)$ between pairs of data points (usually Euclidean or Manhattan distance) in order to find something for k-means to optimise in the first place, to perform the E step we need to have an assumption on the *force function*

$$F_{ij} \propto \mathbb{P}(X_i = j | Y_i, \Theta_t)$$

[1]This could have also been inferred from the fact that they are the expected output of the clustering algorithm, and optimised parameters are the expected outputs of EM.

that will, once normalised, provide the values of X. The Bioinformatics course introduces two such assumptions; the *gravity-based* force

$$F_{ij} = \frac{1}{d(Y_i, \theta_{jt})^2}$$

and the *statistical mechanics-based* force

$$F_{ij} = \exp\left(-\beta \cdot d(Y_i, \theta_{jt})\right)$$

where $\beta \geq 0$ is the *stiffness parameter*, corresponding to the level of "hardness" of the assignment to the closest centroid:

- For $\beta = 0$, the distances become irrelevant, and the assignment is always the *uniform distribution*, i.e. $\mathbb{P}(X_i = j | Y_i, \Theta_t) = \frac{1}{k}$.
- As $\beta \to +\infty$, the influence of the closest centroid, $j' = \operatorname{argmin}_j d(Y_i, \theta_{jt})$, infinitely outweighs the influences of all other centroids, and therefore the assignment corresponds to *hard clustering*, i.e. $\mathbb{P}(X_i = j' | Y_i, \Theta_t) = 1$, while $\mathbb{P}(X_i = j | Y_i, \Theta_t) = 0$ for all $j \neq j'$.

The E step now consists of simply computing the force values for all point-centroid pairs, and normalising; i.e.

$$\mathbb{P}(X_i = j | Y_i, \Theta_t) = \frac{F_{ij}}{\sum_{x=1}^{k} F_{ix}}$$

Once we have the probability distributions on X, the M step proceeds in a very similar vein to the coin-flipping scenario, exploiting an independence assumption on the data points:

- For each data point Y_i individually, the optimal centroid is *exactly* that point, assuming $d(A, B) = 0 \iff A = B$.
- Now we can effectively assign a "fraction" of each data point to the updated centroid positions, corresponding to how likely it is that the data point is assigned to each of the centroids (i.e. the $\mathbb{P}(X_i = j | Y_i, \Theta_t)$ values, which were computed in the E step).

This amounts to the following formula (treating the data points Y_i as vectors)

$$\theta'_j = \frac{\sum_{i=1}^{n} Y_i \mathbb{P}(X_i = j | Y_i, \Theta_t)}{\sum_{i=1}^{n} \mathbb{P}(X_i = j | Y_i, \Theta_t)}$$

and concludes the M step[2].

[2] **N.B.** Compare this with the hard clustering case, where this amounts to making the centroids the *centres of mass* of their associated clusters.

Baum-Welch algorithm

This algorithm consumes an observed output sequence, $\vec{y}$, and seeks to optimise parameters of a hidden Markov model Θ, such that the probability that the HMM has produced the sequence, $\mathbb{P}(\vec{y}|\Theta)$, is optimised[3]. As such, this algorithm can almost be viewed as a *direct application of EM* to HMMs—meaning we need no further assumptions. The HMM setup is easy to translate into the framework:

- The parameters, $\Theta = (\vec{\pi}, \mathbf{T}, \mathbf{E})$, correspond to the start-state probabilities $\pi_x = \mathbb{P}(X_1 = x)$, transition probabilities $\mathbf{T}_{ij} = \mathbb{P}(X_{t+1} = j|X_t = i)$, and emission probabilities $\mathbf{E}_{xy} = \mathbb{P}(Y_t = y|X_t = x)$ of the HMM[4];
- The observed data is exactly the observed sequence of emissions, $\vec{y}$;
- The unobserved data is the underlying sequence of states, X, that the HMM went through while producing this sequence.

The E step of the algorithm then computes two quantities, conditional on the observed sequence:

- The probabilities of being in state x at time t, $\gamma_t(x)$:

$$\gamma_t(x) = \mathbb{P}(X_t = x|\vec{y}, \Theta)$$

- The probabilities of transitioning from state i to state j at time t, $\xi_t(i, j)$:

$$\xi_t(i, j) = \mathbb{P}(X_t = i, X_{t+1} = j|\vec{y}, \Theta)$$

I will now derive a method for computing $\gamma_t(x)$, applying Bayes' theorem:

$$\begin{aligned}
\mathbb{P}(X_t = x|\vec{y}, \Theta) &= \frac{\mathbb{P}(\vec{y}|X_t = x, \Theta)\mathbb{P}(X_t = x|\Theta)}{\mathbb{P}(\vec{y}|\Theta)} && (1)\\
&\propto \mathbb{P}(\vec{y}|X_t = x, \Theta)\mathbb{P}(X_t = x|\Theta) && (2)\\
&= \mathbb{P}(y_{1:t}|X_t = x, \Theta)\mathbb{P}(y_{t+1:T}|X_t = x, \Theta)\mathbb{P}(X_t = x|\Theta) && (3)\\
&= \mathbb{P}(y_{1:t}|X_t = x, \Theta)\mathbb{P}(X_t = x|\Theta)\mathbb{P}(y_{t+1:T}|X_t = x, \Theta) && (4)\\
&= \mathbb{P}(y_{1:t}, X_t = x|\Theta)\mathbb{P}(y_{t+1:T}|X_t = x, \Theta) && (5)\\
&= \alpha_t(x)\beta_t(x) && (6)
\end{aligned}$$

Here, $\alpha_t(x)$ and $\beta_t(x)$ correspond to the intermediate results of the forward and backward algorithm (respectively) for state x at time t. In line 2, we once again apply the fact that the denominator will be the same for all choices of x, while in line 3 we exploit the Markov property (sequence emissions being conditionally independent from one another).

[3] Typically only to a *local* maximum.

[4] **N.B.** We almost always assume the HMM to be *time-homogeneous*, i.e. the above expressions do not depend on the choice of t!

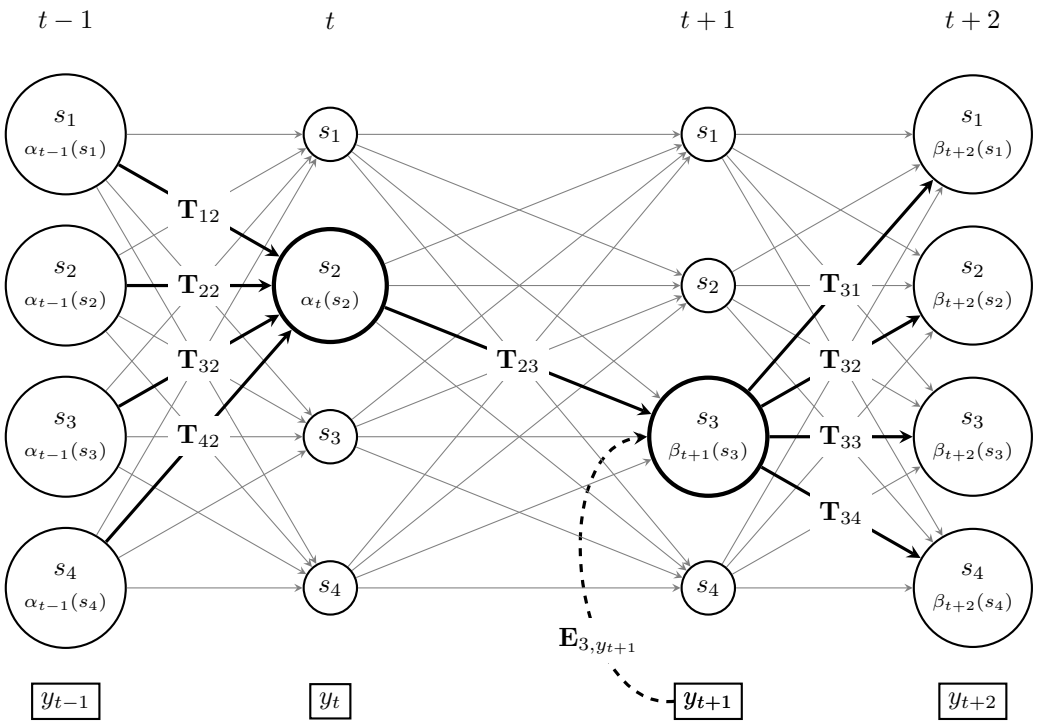

Figure 1: Illustration of the relationship between the quantities used to compute $\xi_t(s_2, s_3) = \alpha_t(s_2)\mathbf{T}_{23}\mathbf{E}_{3,y_{t+1}}\beta_{t+1}(s_3)$.

Using a similar (albeit more tedious) process, we may derive a formula for $\xi_t(i, j)$ in a similar fashion, as

$$\xi_t(i, j) \propto \alpha_t(i)\mathbf{T}_{ij}\mathbf{E}_{j,y_{t+1}}\beta_{t+1}(j)$$

An insight into how these quantities are related by the formula is given by Figure 1. As before, we may compute all the required products (after computing the necessary α and β values by the *forward-backward algorithm*), and then normalise them appropriately.

To perform the M step, note the following three points (one per each HMM parameter):

- The start-state probability π_x simply corresponds to the probability of being in state x at time 1, and therefore

$$\pi'_x = \gamma_1(x)$$

- The transition probability $\mathbf{T}_{ij}$ can be computed by aggregating and normalising the probability of taking this transition across all time steps of

the observed sequence, i.e.

$$\mathbf{T}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

- Lastly, the emission probability $\mathbf{E}_{xy}$ can be considered as the normalised probability of being in state x across any time steps where y was produced (here we once again make advantage of the *indicator function* as was the case with coin-flipping):

$$\mathbf{E}_{xy} = \frac{\sum_{t=1}^{T} \mathbb{I}(y_t = y)\gamma_t(x)}{\sum_{t=1}^{T} \gamma_t(x)}$$

This concludes the M step of the algorithm.