67577 – Intro. to Machine Learning **Fall semester, 2004/5**

Lecture 13: EM-I: Algorithm, Mixture of Bernoulli Distributions

Lecturer: Amnon Shashua *Scribe: Amnon Shashua*[1]

We saw previously that a conditional independence statement $P(X_1,..,X_n \mid H)$ introduces a significant complexity reduction as the n-way array $P(X_1,...,X_n \mid H=h_i)$ is described as an outer-product of n vectors $P(X_j \mid H=h_i)$, $j=1,...,n$. We saw also that a set of such statements can form a graph and if the graph is simple enough (such as a tree) then a significant complexity reduction is valid as well (via the message-passing algorithm).

There are situation, however, where it is known that a set of variables are conditionally independent but we have no observations to that effect, i.e., we observe $P(X_1,...,X_n)$ only. The decoupliing variable H is not observed — which is often referred to *hidden* or *latent*. What we have instead is a marginal probability (a projection of the $n+1$-way array onto an n-way array by contraction with a vector of "1"s):

$$P(X_1,...,X_n) = \sum_{H=h_i} P(X_1,...,X_n \mid H=h_i)P(H=h_i).$$

The marginalization couples the variables. The Expectation-Maximization (EM) algorithm, introduced by Dempster, Laird and Rubin in 1977, is an iterative scheme for decoupling the variables or de-obscuring the structure of the n-way array by "completing" the data, i.e., by re-introducing the hidden variable. To make this clear we will start with an example of a "mixture of Bernoulli distributions" (next lecture we will look at mixture of Gaussians).

Assume we have two coins. The first coin has a probability of heads ("0") equal to p and the second coin has a probability of heads equal to q. At each trial we choose to toss coin 1 with probability λ and coin 2 with probability $1-\lambda$. Once a coin has been chosen it is tossed 3 times, producing an observation $\mathbf{x} \in \{0,1\}^3$ a boolean vector. We are given a set of such observations $D = \{\mathbf{x}_1,...,\mathbf{x}_k\}$ where each observation $\mathbf{x}_i$ is a triplet of coin tosses (the same coin). We wish to determine the parameters $\theta = (p,q,\lambda)$ by using the MAP principle:

$$\theta^* = \text{argmax}_\theta P(\theta \mid D).$$

Assuming that the priors $P(\theta)$ are uniformly distributed (i.e., we have no prior information), then the MAP policy is equal to the Maximum Likelihood:

$$\theta^* = \text{argmax}_\theta P(D \mid \theta).$$

Let $y_i \in \{1,2\}$ be a random variable associated with the observation $\mathbf{x}_i$ such that $y_i = 1$ if $\mathbf{x}_i$ was generated by coin 1 and $y_i = 2$ if $\mathbf{x}_i$ was generated by coin 2. If we knew the values of y_i then our task would be simply to estimate two separate Bernoulli distributions by separating the triplets generated from coin 1 from those generated by coin 2). Since y_i is not known, we have the marginal:

$$\begin{aligned} P(\mathbf{x}_i \mid \theta) &= P(\mathbf{x}_i, y_i=1 \mid \theta) + P(\mathbf{x}_i, y_i=2 \mid \theta) \\ &= P(\mathbf{x}_i \mid y_i=1,\theta)P(y_i=1 \mid \theta) + P(\mathbf{x}_i \mid y_i=2,\theta)P(y_i=2 \mid \theta) \\ &= \lambda p^{n_i}(1-p)^{(3-n_i)} + (1-\lambda)q^{n_i}(1-q)^{(3-n_i)}, \end{aligned} \tag{13.1}$$

[1]

where $0 \leq n_i \leq 3$ is the number of heads ("0") in the triplet of tosses. In other words, the likelihood $P(\mathbf{x}_i \mid \theta)$ of the triplet of tosses $\mathbf{x}_i$ is a linear combination ("mixture") of two Bernoulli distributions. From a perspective of a joint distribution array, we have a 6-way array where three dimensions $\mathbf{x} = (x_1, x_2, x_3) \in \{0,1\}^3$ correspond to the three tosses and three dimensions correspond to the parameters p, q, λ. The likelihood $P(\mathbf{x} \mid \theta)$ over all (eight) possible values of $\mathbf{x}$ is a $2 \times 2 \times 2$ array. From eqn. (13.1) we have that:

$$P(\mathbf{x} \mid \theta) = \lambda \begin{pmatrix} p \\ 1-p \end{pmatrix}^{\otimes 3} + (1-\lambda) \begin{pmatrix} q \\ 1-q \end{pmatrix}^{\otimes 3}.$$

In other words, the "obscurity" introduced by the hidden variable y_i translates to the fact that the 3-way observation "slice" of the 6-way array is of rank=2 (instead of rank=1), i.e., it is a sum of two 3-fold outer-products. So we now have two possible strategies to perform the ML inference: (i) *search* for the "optimal" parameters θ, or (ii) *solve* for the "optimal" parameters.

What is meant by search? when we are given the set of observations D what we are actually given is the $2 \times 2 \times 2$ slice $P(\mathbf{x} \mid \theta^*)$. To see why this is so, consider the 8 possible values of $\mathbf{x}$. Simply count the frequency of each of those 8 possible entries in the data set D. For example, if $k = 100$ and we received the toss $(0,1,0)$ 12 times (i.e., the vector $(0,1,0)$ appeared 12 times in the set of vectors D) then the entry corresponding to $(0,1,0)$ in the $2 \times 2 \times 2$ array represented by $P(\mathbf{x} \mid \theta^*)$ would be 0.12, and so forth. So a search means that we wish to *locate* the 3-way slice we are given as input in the 6-way array (which we can pre-compute). Once we locate it then we have found the values of the 3 missing dimensions, i.e., the values of p^*, q^*, λ^*. Of course a search is not such a great idea because the complexity is exponential with the number of parameters (and their cardinality). For example, if the cardinality of p, q, λ is n, then a search would require $8n^3$ operations.

What is meant by solve? what have just seen is that the rank of the $2 \times 2 \times 2$ array represented by the input slice $P(\mathbf{x} \mid \theta^*)$ is 2 (and moreover it is super-symmetric and the entries are non-negative). This is a constraint because generally a $2 \times 2 \times 2$ array is bounded by above by rank=4. Since the low rank serves as a constraint there is a possibility that we would be able to *factor* the input array into a sum of two super-symmetric non-negative rank=1 3-way arrays. Clearly if we are able to do so then we have found the values of p, q, λ (simply marginalize each of the rank=1 arrays and get the vectors $(p, 1-p)$ and $(q, 1-q)$ up to scale, then λ is recovered so that their combination would be equal to the input array). If we cannot factor uniquely, then the probabilistic problem as stated does not have a unique solution; or if the factorization problem is "hard" in the theoretical computer science sense (which it is indeed), then this means that the solution to the probabilistic problem is also hard and all we can hope for is an approximation.[2]

Now that we have a better picture of what exactly is meant by the hidden variable obscuring the structure of the likelihood function, it is time to introduce the EM algorithm. The EM scheme is an iterative process which ultimately tries to perform the factorization by interleaving two sets of variables: (i) the parameters θ, and (ii) and the probabilistic association between the observations $\mathbf{x}_i$ and and from which factor they came from.

13.1 The EM Algorithm

The EM algorithm is somewhat similar to a "soft" clustering approach where two sets of variables are being optimized. The first set of variables are the parameters θ and the other set of variables

[2]The role of tensor factorizations in probabilistic inference is beyond the scope of this class — for more details you can ask Tamir Hazan who is working on the interplay between algebra and statistics as part of his doctoral studies.

are associations between the observations $\mathbf{x}_i$ and the probability they are arising from each of the factors (in our running example, the probability that a particular observation $\mathbf{x}_i$ came from coin 1 or from coin 2, i.e., $P(y_i \mid \mathbf{x}_i, \theta)$).

We wish to maximize $P(D \mid \theta)$ over θ, which is equivalent to maximizing the log-likelihood:

$$\theta^* = \text{argmax}_\theta \log P(D \mid \theta) = \log\left(\sum_{\mathbf{y}} P(D, \mathbf{y} \mid \theta)\right),$$

where $\mathbf{y}$ represents the hidden variables. We will denote $l(\theta) = \log P(D \mid \theta)$. Let $q(\mathbf{y} \mid D, \theta)$ be some (arbitrary) distribution of the hidden variables $\mathbf{y}$ conditioned on the parameters θ and the input sample D, i.e., $\sum_{\mathbf{y}} q(\mathbf{y} \mid D, \theta) = 1$. We define a *lower bound* on $l(\theta)$ as follows:

$$\begin{aligned}
l(\theta) &= \log\left(\sum_{\mathbf{y}} P(D, \mathbf{y} \mid \theta)\right) \\
&= \log\left(\sum_{\mathbf{y}} q(\mathbf{y} \mid D, \theta)\frac{P(D, \mathbf{y} \mid \theta)}{q(\mathbf{y} \mid D, \theta)}\right) \\
&\geq \sum_{\mathbf{y}} q(\mathbf{y} \mid D, \theta) \log \frac{P(D, \mathbf{y} \mid \theta)}{q(\mathbf{y} \mid D, \theta)} \\
&= Q(q, \theta).
\end{aligned}$$

The inequality comes from Jensen's inequality $\log \sum_j \alpha_j a_j \geq \sum_j \alpha_j \log a_j$ when $\sum_j \alpha_j = 1$. What we have obtained is an "auxiliary" function $Q(q, \theta)$ satisfying

$$l(\theta) \geq Q(q, \theta),$$

for all distributions $q(\mathbf{y} \mid D, \theta)$. The strategy of the EM algorithm is to maximize the lower bound $Q(q, \theta)$ with the hope that if we ascend on the lower bound function we will also ascend with respect to $l(\theta)$. We will see that this is indeed the case and the strategy is therefore valid.

The maximization of $Q(q, \theta)$ proceeds by interleaving the variables q and θ as we separately ascend on each set of variables. At the $(t+1)$ iteration we fix the current value of θ to be $\theta^{(t)}$ of the t'th iteration and maximize $Q(q, \theta^{(t)})$ over q, and then maximize $Q(q^{(t+1)}, \theta)$ over θ:

$$\begin{aligned}
q^{(t+1)} &= \text{argmax}_q Q(q, \theta^{(t)}) \\
\theta^{(t+1)} &= \text{argmax}_\theta Q(q^{(t+1)}, \theta).
\end{aligned}$$

The optimal q of $\max_q Q(q, \theta^{(t)})$ can be described in closed form:

Claim 1 (Jordan-Bishop) *The optimal $q(\mathbf{y} \mid D, \theta^{(t)})$ at each step is $P(\mathbf{y} \mid D, \theta^{(t)})$.*

Proof: We will show that $Q(P(\mathbf{y} \mid D, \theta^{(t)}), \theta^{(t)}) = l(\theta^{(t)})$ which proves the claim since $l(\theta) \geq Q(q, \theta)$ for all q, θ, thus the best q-distribution we can hope to find is one that makes the lower-bound meet $l(\theta)$ at $\theta = \theta^{(t)}$.

$$\begin{aligned}
Q(P(\mathbf{y} \mid D, \theta^{(t)}), \theta^{(t)}) &= \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta^{(t)}) \log \frac{P(D, \mathbf{y} \mid \theta^{(t)})}{P(\mathbf{y} \mid D, \theta^{(t)})} \\
&= \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta^{(t)}) \log \frac{P(\mathbf{y} \mid D, \theta^{(t)}) P(D \mid \theta^{(t)})}{P(\mathbf{y} \mid D, \theta^{(t)})}
\end{aligned}$$

$$\begin{aligned} &= \log P(D \mid \theta^{(t)}) \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta^{(t)}) \\ &= l(\theta^{(t)}) \end{aligned}$$

□

The proof provides also the validity for the approach of ascending along the lower bound $Q(q, \theta)$ because at the point $\theta^{(t)}$ the two functions coincide, i.e., the lower bound function at $\theta = \theta^{(t)}$ is equal to $l(\theta^{(t)})$ therefore if we continue and *ascend* along $Q(\cdot)$ we are *guaranteed* to ascend along $l(\theta)$ as well[3]. The second step of maximizing over θ then becomes:

$$\theta^{(t+1)} = \text{argmax}_\theta Q(q^{(t+1)}, \theta) = \text{argmax}_\theta \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta^{(t)}) \log P(D, \mathbf{y} \mid \theta).$$

This defines the EM algorithm. Often the "Expectation" step is described as taking the expectation of:

$$E_{\mathbf{y} \sim P(\mathbf{y} \mid D, \theta^{(t)})} \left[\log P(D, \mathbf{y} \mid \theta) \right],$$

followed by a Maximization step of finding θ that maximizes the expectation — hence the term EM for this algorithm. The algorithm always ascends thus is guaranteed to stop at a local maxima of the likelihood function — a global maxima cannot be guaranteed (and a clue to this effect we have seen from the rank analogy: finding the rank of a tensor is NP-hard).

13.2 Back to the Coins Example

We will apply the EM scheme to our running example of mixture of Bernoulli distributions. Assuming that the set of observations D is i.i.d. we have:

$$P(D \mid \theta) = \prod_{i=1}^{k} P(\mathbf{x}_i \mid \theta), \qquad P(D, \mathbf{y} \mid \theta) = \prod_{i=1}^{k} P(\mathbf{x}_i, y_i \mid \theta).$$

We wish to compute

$$Q(\theta, \theta^{(t)}) = \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta^{(t)}) \log P(D, \mathbf{y} \mid \theta),$$

and then maximize $Q()$ with respect to p, q, λ.

$$\begin{aligned} Q(\theta, \theta') &= \sum_{\mathbf{y}} P(\mathbf{y} \mid D, \theta') \sum_{i=1}^{k} \log P(\mathbf{x}_i \mid y_i, \theta) P(y_i \mid \theta) \\ &= \sum_{i=1}^{k} \left[P(y_i = 1 \mid \mathbf{x}_i, \theta') \log P(\mathbf{x}_i \mid y_i = 1, \theta) P(y_i = 1 \mid \theta) \right] \\ &+ \sum_{i=1}^{k} \left[P(y_i = 2 \mid \mathbf{x}_i, \theta') \log P(\mathbf{x}_i \mid y_i = 2, \theta) P(y_i = 2 \mid \theta) \right] \\ &= \sum_{i} \left[\mu_i \log(\lambda p^{n_i} (1-p)^{(3-n_i)}) + (1-\mu_i) \log((1-\lambda) q^{n_i} (1-q)^{(3-n_i)}) \right] \end{aligned}$$

[3]this manner of deriving EM was adapted from Jordan and Bishop's book notes, 2001.

where θ' stands for $\theta^{(t)}$ and $\mu_i = P(y_i = 1 \mid \mathbf{x}_i, \theta')$. The values of μ_i are known since $\theta' = (\lambda_o, p_o, q_o)$ are given from the previous iteration. The Bayes formula is used to compute μ_i:

$$\begin{aligned}\mu_i &= P(y_i = 1 \mid \mathbf{x}_i, \theta') = \frac{P(\mathbf{x}_i \mid y_i = 1, \theta')P(y_i = 1 \mid \theta')}{P(\mathbf{x}_i \mid \theta')} \\ &= \frac{\lambda_o p_o^{n_i}(1-p_o)^{(3-n_i)}}{\lambda_o p_o^{n_i}(1-p_o)^{(3-n_i)} + (1-\lambda_o) q_o^{n_i}(1-q_o)^{(3-n_i)}}\end{aligned}$$

We wish to compute: $\max_{p,q,\lambda} Q(\theta, \theta')$. The partial derivative with respect to λ is:

$$\frac{\partial Q}{\partial \lambda} = \sum_i \mu_i \frac{1}{\lambda} - \sum_i (1-\mu_i)\frac{1}{1-\lambda} = 0,$$

from which we obtain the update formula of λ given μ_i:

$$\lambda = \frac{1}{k}\sum_{i=1}^{k} \mu_i.$$

The partial derivative with respect to p is:

$$\frac{\partial Q}{\partial p} = \sum_i \frac{\mu_i n_i}{p} - \sum_i \frac{\mu_i(3-n_i)}{1-p} = 0,$$

from which we obtain the update formula:

$$p = \frac{1}{\sum_i \mu_i}\sum_i \frac{n_i}{3}\mu_i.$$

Likewise the update rule for q is:

$$q = \frac{1}{\sum_i (1-\mu_i)}\sum_i \frac{n_i}{3}(1-\mu_i).$$

To conclude, we start with some initial "guess" of the values of p, q, λ, compute the values of μ_i and update iteratively the values of p, q, λ where at the end of each iteration the new values of μ_i are computed.

13.3 EM as a "soft" Clustering Scheme

What we have seen so far is that the EM scheme introduces another set of variables, the μ_i in our running example, and performs an interleaving optimization over the two sets of variables the θ and the μ_i. We will observe more closely now the role of μ_i in achieving the factorization in our running example of coin tosses.

Recall from our discussion above that what we are given is $P(\mathbf{x} \mid \theta^*)$ which is a $2 \times 2 \times 2$ array which we will denote by G. It will be convenient to represent each triplet of tosses $\mathbf{x}_i$ as a $2 \times 2 \times 2$ array A_i with the value "1" at the entry corresponding to the value of $\mathbf{x}_i$ (there are 8 possible values to $\mathbf{x}_i$) and "0" in the remaining 7 entries. Therefore,

$$G = \frac{1}{k}\sum_{i=1}^{k} A_i.$$

Let $I_1 \subseteq \{1,...,k\}$ be the subset of indices of tosses arising from coin 1 and let $I_2 \subseteq \{1,...,k\}$ be the subset of indices arising from coin 2. We have that $I_1 \bigcap I_2 = \emptyset$ and $I_1 \bigcup I_2 = \{1,...,k\}$. Let $k_1 = |I_1|$ and $k_2 = |I_2|$ be the cardinality of each subset. Then, our desire to factorize G is expressed as follows:

$$\begin{aligned}
G = \frac{1}{k}\sum_{i=1}^{k} A_i &= \frac{1}{k}\sum_{i\in I_1} A_i + \frac{1}{k}\sum_{j\in I_2} A_j \\
&= \frac{k_1}{k}\frac{1}{k_1}\sum_{i\in I_1} A_i + \frac{k_2}{k}\frac{1}{k_2}\sum_{j\in I_2} A_j \\
&= \lambda\frac{1}{k_1}\sum_{i\in I_1} A_i + (1-\lambda)\frac{1}{k_2}\sum_{j\in I_2} A_j \\
&= \lambda G_1 + (1-\lambda) G_2
\end{aligned}$$

The factorization is possible if we *know* from which coin each observation A_i came from (knowing the value of the hidden variables y_i). The 3-way array G_1 represents $P(\mathbf{x} \mid y_i = 1, \theta^*)$ and the 3-way array G_2 represents $P(\mathbf{x} \mid y_i = 2, \theta^*)$. Our factorization is done if we know the sets I_1 and I_2. Thus, a possible way to look at this is that we wish to divide the set of indices $\{1,...,k\}$ into two subsets I_1, I_2.

Consider next some arbitrary values μ_i, $i = 1,...,k$. Then,

$$G = \frac{1}{k}\sum_{i=1}^{k} \mu_i A_i + \frac{1}{k}\sum_{i=1}^{k}(1-\mu_i)A_i,$$

and this holds for all μ_i. This is the analogue of $l(\theta) \geq Q(q,\theta)$ where the inequality was because of the logarithm — we do not actually need to introduce the logarithm to get a picture of what the EM is about to do. The point is that there *exists* a setting of the values of μ_i such that:

$$\frac{1}{k}\sum_{i=1}^{k} \mu_i A_i = \lambda G_1, \qquad \frac{1}{k}\sum_{i=1}^{k}(1-\mu_i)A_i = (1-\lambda)G_2.$$

In other words, there exists μ_i such that $\sum_i \mu_i A_i$ is rank=1 and $\sum_i (1-\mu_i)A_i$ is also rank=1 and the scaled sum of the two arrays equals to G. The value of μ_i therefore represents a "soft" membership, a probability, that A_i came from I_1 or from I_2. Thus μ_i represents $P(y_i = 1 \mid \mathbf{x}_i, \theta^*)$. The EM algorithm iteratively solves for the values of μ_i thus effectively clustering the observations to the two sources (the coins) that generated them.