

Expectation Maximization (EM)

The EM algorithm is used to train models involving latent variables using training data in which the latent variables are not observed (unlabeled data). This is to be contrasted with training using labeled data where all variables in the model are observed in the training data. For example, in labeled training of HMMs one assumes that one has pairs of observable sequences and the associated hidden state sequences. In unlabeled training of HMMs the training data consists of just the observation sequences. EM is a training algorithm for the unlabeled case.

Learning latent variable models from unlabeled data is very challenging and has had rather poor results in practice for HMMs and PCFGs. Training has been much more successful for partially labeled data where we have a structured label but lack an alignment between the structured input and the structured label. Automatic alignment is important for training speech recognition systems and machine translation systems. The use of EM for alignment is discussed in section 6.

EM is perhaps easiest to understand through particular examples. The following sections give three special cases of the algorithm before presenting the general case.

1 Mixtures of Gaussians

For $\mu \in R^d$ and Σ a $d \times d$ matrix, we let $\mathcal{N}(\mu, \Sigma)$ be the multivariate Gaussian distribution with mean μ and covariance matrix Σ .

$$\mathcal{N}(\mu, \Sigma)(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)\Sigma^{-1}(x - \mu)\right)$$

A mixture of K Gaussians is defined by a system of parameters $\beta = \langle \lambda_1, \mu_1, \Sigma_1, \dots, \lambda_K, \mu_K, \Sigma_K \rangle$ where we require that $\lambda_1, \dots, \lambda_K$ is a convex combination, i.e., $\lambda_i \geq 0$ and $\sum_{i=1}^K \lambda_i = 1$. The probability model defined by parameter system β is as follows.

$$P_\beta(x) = \sum_{i=1}^K \lambda_i \mathcal{N}(\mu_i, \Sigma_i)(x) \tag{1}$$

Now suppose that we have a sample x_1, \dots, x_T with $x_t \in R^d$ drawn *IID* from some distribution D . Here we are interested in the following optimization problem.

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \sum_{t=1}^T \ln \frac{1}{P_\beta(x_t)} \tag{2}$$

$$= \underset{\beta}{\operatorname{argmax}} \prod_{t=1}^T P_\beta(x_t) \tag{3}$$

Equation (2) defines maximum likelihood density estimation. It is maximum likelihood (ML) rather than MAP (maximum a-posteriori probability) because there is no regularization term such as $\lambda\|\beta\|^2$. Maximum Likelihood is appropriate if T is large compared to the number of parameters being trained.

We can interpret (1) as derived from a latent variable $y \in \{1, \dots, K\}$ where we have the following.

$$P_\beta(y = i) = \lambda_i \quad (4)$$

$$P_\beta(x|y) = \mathcal{N}(\mu_y, \Sigma_y)(x) \quad (5)$$

Equation (1) can then be interpreted as follows.

$$P_\beta(x) = \sum_{y=1}^K P_\beta(y)P_\beta(x|y) \quad (6)$$

$$= \sum_{y=1}^K P_\beta(x, y) \quad (7)$$

We can see that $P_\beta(x, y)$ is a generative model (a Bayesian network). It first generates y and then generates x from y using conditional probabilities. Now suppose that we had labeled training data $S = \langle x_1, y_1 \rangle, \dots, \langle x_T, y_T \rangle$. In generative training for labeled data we set the parameter vector β as follows.

$$\beta^* = \operatorname{argmin}_\beta \sum_{t=1}^T \ln \frac{1}{P_\beta(x_t, y_t)} \quad (8)$$

The solution β^* is then given by count ratios and empirical averages as follows.

$$\lambda_i = \frac{1}{T} \sum_{t=1}^T I[y_t = i] \quad (9)$$

$$\mu_i = \frac{\sum_{t=1}^T x_t I[y_t = i]}{\sum_{t=1}^T I[y_t = i]} \quad (10)$$

$$(\Sigma_i)_{j,k} = \frac{\sum_{t=1}^T (x_t - \mu_i)_j (x_t - \mu_i)_k I[y_t = i]}{\sum_{t=1}^T I[y_t = i]} \quad (11)$$

In the EM algorithm one constructs a series of parameter values $\beta^0, \beta^1, \beta^2, \dots$. We will ignore how to construct the initial value β^0 . In the EM algorithm one uses the distribution $P_\beta(y_t|x_t)$ as a substitute for labels where $P_\beta(y_t|x_t)$ can be computed by Bayes rules from (4) and (5). In EM for Gaussian mixtures we compute β^{n+1} from β^n by using a variant of (9), (10), and (11) in which the indicator function $I[y_t = i]$ is replaced by the probability $P_{\beta^n}(y_t = i|x_t)$.

$$\lambda_i^{n+1} = \frac{1}{T} \sum_{t=1}^T P_{\beta^n}(y_t = i | x_t) \quad (12)$$

$$\mu_i^{n+1} = \frac{\sum_{t=1}^T x_t P_{\beta^n}(y_t = i | x_t)}{\sum_{t=1}^T P_{\beta^n}(y_t = i | x_t)} \quad (13)$$

$$(\Sigma_i^{n+1})_{j,k} = \frac{\sum_{t=1}^T (x_t - \mu_i)_j (x_t - \mu_i)_k P_{\beta^n}(y_t = i | x_t)}{\sum_{t=1}^T P_{\beta^n}(y_t = i | x_t)} \quad (14)$$

2 EM for HMMs

In HMMs we have that x_t and y_t are both sequences $x_t^1, \dots, x_t^{N_t}$ and $y_t^1, \dots, y_t^{N_t}$. Here we have $x_t^i \in \mathcal{O}$ where \mathcal{O} is a set of possible observations and $y_t^i \in \mathcal{S}$ where \mathcal{S} is a set of possible hidden states. For $s \in \mathcal{S}$ we have a parameter Π_s specifying $P_{\beta}(y_t^1 = s)$. For $s, w \in \mathcal{S}$ we have a parameter $T_{s,w}$ specifying $P_{\beta}(y_t^{i+1} = w | y_t^i = s)$. Finally for $s \in \mathcal{S}$ and $u \in \mathcal{O}$ we have a parameter $O_{s,u}$ specifying $P_{\beta}(x_t^i = u | y_t^i = s)$. This gives the following joint probability.

$$P_{\beta}(x, y) = \Pi_{y^1} \prod_{i=1}^{N-1} T_{y^i, y^{i+1}} \prod_{i=1}^N O_{y^i, x^i} \quad (15)$$

Now suppose we had labeled training data $\langle x_1, y_1 \rangle, \dots, \langle x_T, y_T \rangle$. For HMMs the generative training equation (8) has a closed form solution as follows.

$$\Pi_s^* = \frac{\sum_{t=1}^T I[y_t^1 = s]}{T} \quad (16)$$

$$T_{s,w}^* = \frac{\sum_{t=1}^T \sum_{i=1}^{N_t-1} I[y_t^i = s \wedge y_t^{i+1} = w]}{\sum_{t=1}^T \sum_{i=1}^{N_t-1} I[y_t^i = s]} \quad (17)$$

$$O_{s,u}^* = \frac{\sum_{t=1}^T \sum_{i=1}^{N_t} I[y_t^i = s \wedge x_t^{i+1} = u]}{\sum_{t=1}^T \sum_{i=1}^{N_t} I[y_t^i = s]} \quad (18)$$

Again each component of β^* is set to a count ratio. To convert labeled generative training to the EM algorithm we again replace the indicator functions by probabilities as follows.

$$\Pi_s^{n+1} = \frac{\sum_{t=1}^T P_{\beta^n}(y_t^1 = s | x_t)}{T} \quad (19)$$

$$T_{s,w}^{n+1} = \frac{\sum_{t=1}^T \sum_{i=1}^{N_t-1} P_{\beta^n}(y_t^i = s \wedge y_t^{i+1} = w | x_t)}{\sum_{t=1}^T \sum_{i=1}^{N_t-1} P_{\beta^n}(y_t^i = s | x_t)} \quad (20)$$

$$O_{s,u}^{n+1} = \frac{\sum_{t=1}^T \sum_{i=1}^{N_t} P_{\beta^n}(y_t^i = s \wedge x_t^{i+1} = u | x_t)}{\sum_{t=1}^T \sum_{i=1}^{N_t} P_{\beta^n}(y_t^i = s | x_t)} \quad (21)$$

The probabilities appearing in equations (19), (20), and (21) can be computed with the forward-backward procedure.

3 EM for PCFGs

For probabilistic context free grammars (PCFGs) we have that x_t is a string and y_t is a parse tree whose yield (sequence of leaf nodes) is x_t . We assume that y_t must use only productions from a fixed (finite) set of productions \mathcal{G} in Chomsky normal form, i.e., every production is either of the form $X \rightarrow YZ$ where $X, Y,$ and Z are nonterminal symbols or of the form $X \rightarrow a$ where a is a terminal symbol. For each production $X \rightarrow \alpha \in \mathcal{G}$ we assume a probability $P(X \rightarrow \alpha)$ satisfying the following normalization constraint at each nonterminal X .

$$\sum_{\alpha: X \rightarrow \alpha \in \mathcal{G}} P(X \rightarrow \alpha) = 1$$

We let β be the set of all probabilities of the form $P(X \rightarrow \alpha)$ and we define $P_\beta(x_t, y_t)$ as follows.

$$P_\beta(x_t, y_t) = \begin{cases} 0 & \text{if } x_t \text{ is not the yield of } y_t \\ \prod_{X \rightarrow \alpha \in y_t} P_\beta(X \rightarrow \alpha) & \text{otherwise} \end{cases}$$

In the above product over productions we allow a given production to occur more than once. We write $\#(X \rightarrow \alpha, y_t)$ to be the number of times that the production $X \rightarrow \alpha$ occurs in the tree y_t . We let $\#(X, y_t)$ be the number of nodes in the tree y_t labeled with the nonterminal X . For PCFGs the generative training equation (8) has a closed form solution given as follows.

$$P^*(X \rightarrow \alpha) = \frac{\sum_{t=1}^T \#(X \rightarrow \alpha, y_t)}{\sum_{t=1}^T \#(X, y_t)} \quad (22)$$

Again, in generative training each component of β^* is set to a count ratio. We convert generative training to an EM update by replacing the counts with expected counts (this corresponds to replacing indicator functions by probabilities).

$$P^{n+1}(X \rightarrow \alpha) = \frac{\sum_{t=1}^T \mathbb{E}_{y \sim P_{\beta^n}(\cdot | x_t)} [\#(X \rightarrow \alpha, y)]}{\sum_{t=1}^T \mathbb{E}_{y \sim P_{\beta^n}(\cdot | x_t)} [\#(X, y)]} \quad (23)$$

The expectations appearing in (23) can be computed using the inside-outside algorithm.

4 General EM

In general EM we assume arbitrary sets \mathcal{X} and \mathcal{Y} (for now we will assume these sets are discrete). We assume a distribution on $\mathcal{X} \times \mathcal{Y}$ parameterized by

parameter vector β , i.e., for a given value of β , and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, we have a probability $P_\beta(x, y)$. We are given a sample x_1, \dots, x_t with $x_t \in \mathcal{X}$ and we want to train the model using the following.

$$\beta^* = \operatorname{argmin}_\beta \sum_{t=1}^T \ln \frac{1}{P_\beta(x_t)} \quad (24)$$

$$P_\beta(x) = \sum_{y \in \mathcal{Y}} P_\beta(x, y) \quad (25)$$

EM is an iterative algorithm that computes a sequence of parameter vectors $\beta^0, \beta^1, \beta^2, \dots$. We will ignore the question of how to set β^0 and concentrate instead on how to compute β^{n+1} from β^n . The basic idea is that β^n defines a probability distribution $P_{\beta^n}(y|x)$. This distribution is treated as implicit labels. If we had a labeled data set $\langle x_1, y_1 \rangle, \dots, \langle x_T, y_T \rangle$ then generative training would set β using the following.

$$\beta^* = \operatorname{argmin}_\beta \sum_{t=1}^T \ln \frac{1}{P_\beta(x_t, y_t)} \quad (26)$$

In EM the following variant of this equation is used to compute β^{n+1} from β^n .

$$\beta^{n+1} = \operatorname{argmin}_\beta \sum_{t=1}^T \mathbb{E}_{y \sim P_{\beta^n}(\cdot|x_t)} \left[\ln \frac{1}{P_\beta(x_t, y)} \right] \quad (27)$$

In each of the tree examples given in the previous sections, (26) has a closed form solution in which each parameter is given by a count ratio. In such cases equation (27) also has a closed form solution where each parameter is given by a ratio of expectations or probabilities. In each of the above examples, only the closed form solution for β^{n+1} given. But general EM algorithm is defined by (27).

It is important to keep in mind that while (26) and (27) are often easily solved, equation (24) is much more challenging. For models in the exponential family (MRFs, CRFs) we have that (26) and (27) are convex in β but (24) is not. The EM algorithm often converges in practice to a local optimum with a value significantly larger than the global optimum of (24).

5 The EM Theorem

The EM theorem states, essentially, that the EM update makes progress in optimizing (24) on every step. Before proving the theorem we note that without loss of generality we can consider just a single pair of variables X and Y with a model $P_\beta(X, Y)$. This does not lose generality because we take X to be the sequence x_1, \dots, x_T and Y to be the sequence y_1, \dots, y_T . We can now state the EM theorem as follows.

EM Theorem: If

$$\mathbb{E}_{Y \sim S} \left[\ln \frac{1}{P_{\beta'}(X, Y)} \right] < \mathbb{E}_{Y \sim S} \left[\ln \frac{1}{P_{\beta}(X, Y)} \right] \quad (28)$$

$$\text{where } S = P_{\beta}(\cdot|X) \quad (29)$$

then

$$P_{\beta'}(X) > P_{\beta}(X) \quad (30)$$

Proof:

$$\mathbb{E}_{Y \sim S} \left[\ln \frac{1}{P_{\beta}(X, Y)} \right] - \mathbb{E}_{Y \sim S} \left[\ln \frac{1}{P_{\beta'}(X, Y)} \right] > 0$$

$$\mathbb{E}_{Y \sim S} \left[\ln \frac{P_{\beta'}(X, Y)}{P_{\beta}(X, Y)} \right] > 0$$

$$\mathbb{E}_{Y \sim S} \left[\ln \frac{P_{\beta'}(X)P_{\beta'}(Y|X)}{P_{\beta}(X)P_{\beta}(Y|X)} \right] > 0$$

$$\mathbb{E}_{Y \sim S} \left[\ln \frac{P_{\beta'}(X)}{P_{\beta}(X)} \right] + \mathbb{E}_{Y \sim S} \left[\ln \frac{P_{\beta'}(Y|X)}{P_{\beta}(Y|X)} \right] > 0$$

$$\ln \frac{P_{\beta'}(X)}{P_{\beta}(X)} - \mathbb{E}_{Y \sim S} \left[\ln \frac{P_{\beta}(Y|X)}{P_{\beta'}(Y|X)} \right] > 0$$

$$\ln \frac{P_{\beta'}(X)}{P_{\beta}(X)} - KL(S, P_{\beta'}(\cdot|X)) > 0$$

$$\ln \frac{P_{\beta'}(X)}{P_{\beta}(X)} > 0$$

$$P_{\beta'}(X) > P_{\beta}(X)$$

6 EM for Alignment

EM does not work well in practice for unlabeled training of HMMS or PCFGs. However, EM is useful in practice for solving alignment problems that arise in partially labeled data. Speech recognition systems are trained with pairs of speech signals and corresponding sentences (word sequences). Although the whole speech signal is labeled with the whole sentence, the sentence and the signal are not aligned. The correspondence between a particular time in the speech signal and a particular place in the sentence is determined automatically. This

is done by running EM on a large corpus of such pairs treating the alignment as the latent variable. EM is used similarly for alignment in translation between natural languages. In this case the training data consists of translation pairs each of which is a pair of a sentence in the source language (say English) and its translation into the target language (say French). Although each source language sentence is labeled with its translation, the alignment is not specified — there is no explicit labeling of which words or phrases in the source correspond to which words or phrases in the target. In practice the alignment is done automatically by running EM on a large corpus of training pairs treating the alignment as the latent variable.

7 Problems

1. Suppose we initialize the PCFG to be deterministic in the sense that for any word string there is at most one parse of that string with nonzero probability and suppose that each x_h does have a parse under the initial grammar. How rapidly does EM converge in this case? What does it converge to? Justify your answer.