What is the expectation maximization algorithm?

Chuong B Do & Serafim Batzoglou

The expectation maximization algorithm arises in many computational biology applications that involve probabilistic models. What is it good for, and how does it work?

Probabilistic models, such as hidden Markov models or Bayesian networks, are commonly used to model biological data. Much of their popularity can be attributed to the existence of efficient and robust procedures for learning parameters from observations. Often, however, the only data available for training a probabilistic model are incomplete. Missing values can occur, for example, in medical diagnosis, where patient histories generally include results from a limited battery of tests. Alternatively, in gene expression clustering, incomplete data arise from the intentional omission of gene-to-cluster assignments in the probabilistic model. The expectation maximization algorithm enables parameter estimation in probabilistic models with incomplete data.

A coin-flipping experiment

As an example, consider a simple coin-flipping experiment in which we are given a pair of coins A and B of unknown biases, θ_A and θ_B, respectively (that is, on any given flip, coin A will land on heads with probability θ_A and tails with probability $1-\theta_A$ and similarly for coin B). Our goal is to estimate $\theta = (\theta_A, \theta_B)$ by repeating the following procedure five times: randomly choose one of the two coins (with equal probability), and perform ten independent coin tosses with the selected coin. Thus, the entire procedure involves a total of 50 coin tosses (**Fig. 1a**).

During our experiment, suppose that we keep track of two vectors $x = (x_1, x_2, \ldots, x_5)$ and $z = (z_1, z_2, \ldots, z_5)$, where $x_i \in \{0,1,\ldots,10\}$ is the number of heads observed during the ith set of tosses, and $z_i \in \{A,B\}$ is the identity of the coin used during the ith set of tosses. Parameter estimation in this setting is known as the complete data case in that the values of all relevant random variables in our model (that is, the result of each coin flip and the type of coin used for each flip) are known.

Here, a simple way to estimate θ_A and θ_B is to return the observed proportions of heads for each coin:

$$\hat{\theta}_A = \frac{\#\text{ of heads using coin A}}{\text{total }\#\text{ of flips using coin A}} \quad (1)$$

and

$$\hat{\theta}_B = \frac{\#\text{ of heads using coin B}}{\text{total }\#\text{ of flips using coin B}}$$

This intuitive guess is, in fact, known in the statistical literature as maximum likelihood estimation (roughly speaking, the maximum likelihood method assesses the quality of a statistical model based on the probability it assigns to the observed data). If $\log P(x,z;\theta)$ is the logarithm of the joint probability (or log-likelihood) of obtaining any particular vector of observed head counts x and coin types z, then the formulas in (1) solve for the parameters $\hat{\theta} = (\hat{\theta}_A, \hat{\theta}_B)$ that maximize $\log P(x,z;\theta)$.

Now consider a more challenging variant of the parameter estimation problem in which we are given the recorded head counts x but not the identities z of the coins used for each set of tosses. We refer to z as hidden variables or latent factors. Parameter estimation in this new setting is known as the incomplete data case. This time, computing proportions of heads for each coin is no longer possible, because we don't know the coin used for each set of tosses. However, if we had some way of completing the data (in our case, guessing correctly which coin was used in each of the five sets), then we could reduce parameter estimation for this problem with incomplete data to maximum likelihood estimation with complete data.

One iterative scheme for obtaining completions could work as follows: starting from some initial parameters, $\hat{\theta}^{(t)} = (\hat{\theta}_A^{(t)}, \hat{\theta}_B^{(t)})$, determine for each of the five sets whether coin A or coin B was more likely to have generated the observed flips (using the current parameter estimates). Then, assume these completions (that is, guessed coin assignments) to be correct, and apply the regular maximum likelihood estimation procedure to get $\hat{\theta}^{(t+1)}$. Finally, repeat these two steps until convergence. As the estimated model improves, so too will the quality of the resulting completions.

The expectation maximization algorithm is a refinement on this basic idea. Rather than picking the single most likely completion of the missing coin assignments on each iteration, the expectation maximization algorithm computes probabilities for each possible completion of the missing data, using the current parameters $\hat{\theta}^{(t)}$. These probabilities are used to create a weighted training set consisting of all possible completions of the data. Finally, a modified version of maximum likelihood estimation that deals with weighted training examples provides new parameter estimates, $\hat{\theta}^{(t+1)}$. By using weighted training examples rather than choosing the single best completion, the expectation maximization algorithm accounts for the confidence of the model in each completion of the data (**Fig. 1b**).

In summary, the expectation maximization algorithm alternates between the steps

Chuong B. Do and Serafim Batzoglou are in the Computer Science Department, Stanford University, 318 Campus Drive, Stanford, California 94305-5428, USA.
e-mail: chuong@cs.stanford.edu

of guessing a probability distribution over completions of missing data given the current model (known as the E-step) and then re-estimating the model parameters using these completions (known as the M-step). The name 'E-step' comes from the fact that one does not usually need to form the probability distribution over completions explicitly, but rather need only compute 'expected' sufficient statistics over these completions. Similarly, the name 'M-step' comes from the fact that model reestimation can be thought of as 'maximization' of the expected log-likelihood of the data.

Introduced as early as 1950 by Ceppellini *et al.*[1] in the context of gene frequency estimation, the expectation maximization algorithm was analyzed more generally by Hartley[2] and by Baum *et al.*[3] in the context of hidden Markov models, where it is commonly known as the Baum-Welch algorithm. The standard reference on the expectation maximization algorithm and its convergence is Dempster *et al*[4].

Mathematical foundations

How does the expectation maximization algorithm work? More importantly, why is it even necessary?

The expectation maximization algorithm is a natural generalization of maximum likelihood estimation to the incomplete data case. In particular, expectation maximization attempts to find the parameters $\hat{\theta}$ that maximize the log probability $\log P(x;\theta)$ of the observed data. Generally speaking, the optimization problem addressed by the expectation maximization algorithm is more difficult than the optimization used in maximum likelihood estimation. In the complete data case, the objective function $\log P(x,z;\theta)$ has a single global optimum, which can often be found in closed form (e.g., equation 1). In contrast, in the incomplete data case the function $\log P(x;\theta)$ has multiple local maxima and no closed form solution.

To deal with this, the expectation maximization algorithm reduces the difficult task of optimizing $\log P(x;\theta)$ into a sequence of simpler optimization subproblems, whose objective functions have unique global maxima that can often be computed in closed form. These subproblems are chosen in a way that guarantees their corresponding solutions $\hat{\theta}^{(1)},\hat{\theta}^{(2)},\ldots$ and will converge to a local optimum of $\log P(x;\theta)$.

More specifically, the expectation maximization algorithm alternates between two phases. During the E-step, expectation maximization chooses a function g_t that lower bounds $\log P(x;\theta)$ everywhere, and for which $g_t(\hat{\theta}^{(t)}) = \log P(x;\hat{\theta}^{(t)})$. During the M-step, the expectation maximization algorithm moves to a new parameter set $\hat{\theta}^{(t+1)}$ that maximizes g_t. As the value of the lower-bound g_t matches the objective function at $\hat{\theta}^{(t)}$, it follows that $\log P(x;\hat{\theta}^{(t)}) = g_t(\hat{\theta}^{(t)}) \le g_t(\hat{\theta}^{(t+1)}) = \log P(x;\hat{\theta}^{(t+1)})$—so the objective function monotonically increases during each iteration of expectation maximization! A graphical illustration of this argument is provided in **Supplementary Figure 1** online, and a concise mathematical derivation of the expectation maximization algorithm is given in **Supplementary Note 1** online.

As with most optimization methods for nonconcave functions, the expectation maximization algorithm comes with guarantees only of convergence to a local maximum of the objective function (except in degenerate cases). Running the procedure using multiple initial starting parameters is often helpful; similarly, initializing parameters in a way that breaks symmetry in models is also important. With this limited set of tricks, the expectation maximization algorithm provides a simple and robust tool for parameter estimation in models with incomplete data. In theory, other numerical optimization techniques, such as gradient descent or Newton-Raphson, could be used instead of expectation maximization; in practice, however, expectation maximization has the advantage of being simple, robust and easy to implement.

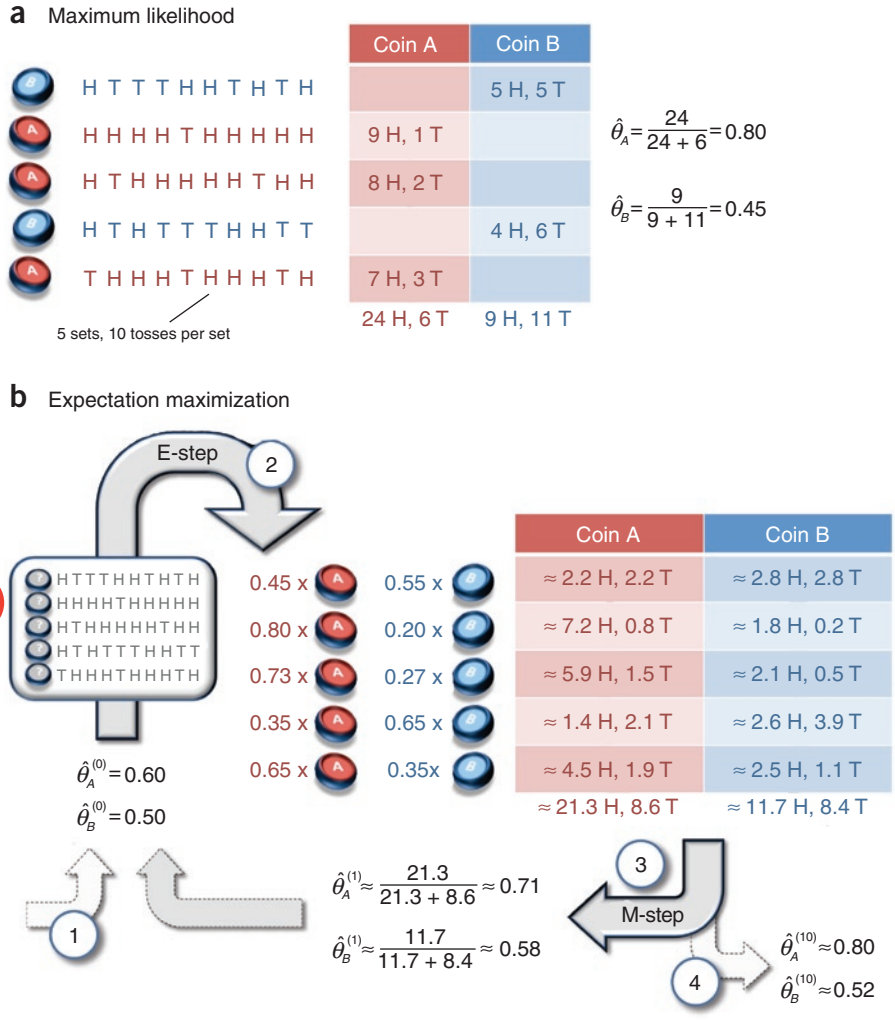

Figure 1 Parameter estimation for complete and incomplete data. (**a**) Maximum likelihood estimation. For each set of ten tosses, the maximum likelihood procedure accumulates the counts of heads and tails for coins A and B separately. These counts are then used to estimate the coin biases. (**b**) Expectation maximization. 1. EM starts with an initial guess of the parameters. 2. In the E-step, a probability distribution over possible completions is computed using the current parameters. The counts shown in the table are the expected numbers of heads and tails according to this distribution. 3. In the M-step, new parameters are determined using the current completions. 4. After several repetitions of the E-step and M-step, the algorithm converges.

Applications

Many probabilistic models in computational biology include latent variables. In some

cases, these latent variables are present due to missing or corrupted data; in most applications of expectation maximization to computational biology, however, the latent factors are intentionally included, and parameter learning itself provides a mechanism for knowledge discovery.

For instance, in gene expression clustering[5], we are given microarray gene expression measurements for thousands of genes under varying conditions, and our goal is to group the observed expression vectors into distinct clusters of related genes. One approach is to model the vector of expression measurements for each gene as being sampled from a multivariate Gaussian distribution (a generalization of a standard Gaussian distribution to multiple correlated variables) associated with that gene's cluster. In this case, the observed data x correspond to microarray measurements, the unobserved latent factors z are the assignments of genes to clusters, and the parameters θ include the means and covariance matrices of the multivariate Gaussian distributions representing the expression patterns for each cluster. For parameter learning, the expectation maximization algorithm alternates between computing probabilities for assignments of each gene to each cluster (E-step) and updating the cluster means and covariances based on the set of genes predominantly belonging to that cluster (M-step). This can be thought of as a 'soft' variant of the popular k-means clustering algorithm, in which one alternates between 'hard' assignments of genes to clusters and reestimation of cluster means based on their assigned genes.

In motif finding[6], we are given a set of unaligned DNA sequences and asked to identify a pattern of length W that is present (though possibly with minor variations) in every sequence from the set. To apply the expectation maximization algorithm, we model the instance of the motif in each sequence as having each letter sampled independently from a position-specific distribution over letters, and the remaining letters in each sequence as coming from some fixed background distribution. The observed data x consist of the letters of sequences, the unobserved latent factors z include the starting position of the motif in each sequence and the parameters θ describe the position-specific letter frequencies for the motif. Here, the expectation maximization algorithm involves computing the probability distribution over motif start positions for each sequence (E-step) and updating the motif letter frequencies based on the expected letter counts for each position in the motif (M-step).

In the haplotype inference problem[7], we are given the unphased genotypes of individuals from some population, where each unphased genotype consists of unordered pairs of single-nucleotide polymorphisms (SNPs) taken from homologous chromosomes of the individual. Contiguous blocks of SNPs inherited from a single chromosome are called haplotypes. Assuming for simplicity that each individual's genotype is a combination of two haplotypes (one maternal and one paternal), the goal of haplotype inference is to determine a small set of haplotypes that best explain all of the unphased genotypes observed in the population. Here, the observed data x are the known unphased genotypes for each individual, the unobserved latent factors z are putative assignments of unphased genotypes to pairs of haplotypes and the parameters θ describe the frequencies of each haplotype in the population. The expectation maximization algorithm alternates between using the current haplotype frequencies to estimate probability distributions over phasing assignments for each unphased genotype (E-step) and using the expected phasing assignments to refine estimates of haplotype frequencies (M-step).

Other problems in which the expectation maximization algorithm plays a prominent role include learning profiles of protein domains[8] and RNA families[9], discovery of transcriptional modules[10], tests of linkage disequilibrium[11], protein identification[12] and medical imaging[13].

In each case, expectation maximization provides a simple, easy-to-implement and efficient tool for learning parameters of a model; once these parameters are known, we can use probabilistic inference to ask interesting queries about the model. For example, what cluster does a particular gene most likely belong to? What is the most likely starting location of a motif in a particular sequence? What are the most likely haplotype blocks making up the genotype of a specific individual? By providing a straightforward mechanism for parameter learning in all of these models, expectation maximization provides a mechanism for building and training rich probabilistic models for biological applications.

Note: Supplementary information is available on the Nature Biotechnology website.

ACKNOWLEDGMENTS

C.B.D. is supported in part by an National Science Foundation (NSF) Graduate Fellowship. S.B. wishes to acknowledge support by the NSF CAREER Award. We thank four anonymous referees for helpful suggestions.

1. Ceppellini, R., Siniscalco, M. & Smith, C.A. *Ann. Hum. Genet.* **20**, 97–115 (1955).
2. Hartley, H. *Biometrics* **14**, 174–194 (1958).
3. Baum, L.E., Petrie, T., Soules, G. & Weiss, N. *Ann. Math. Stat.* **41**, 164–171 (1970).
4. Dempster, A.P., Laird, N.M. & Rubin, D.B. *J. R. Stat. Soc. Ser. B* **39**, 1–38 (1977).
5. D'haeseleer, P. *Nat. Biotechnol.* **23**, 1499–1501 (2005).
6. Lawrence, C.E. & Reilly, A.A. *Proteins* **7**, 41–51 (1990).
7. Excoffier, L. & Slatkin, M. *Mol. Biol. Evol.* **12**, 921–927 (1995).
8. Krogh, A., Brown, M., Mian, I.S., Sjölander, K. & Haussler, D. *J. Mol. Biol.* **235**, 1501–1543 (1994).
9. Eddy, S.R. & Durbin, R. *Nucleic Acids Res.* **22**, 2079–2088 (1994).
10. Segal, E., Yelensky, R. & Koller, D. *Bioinformatics* **19**, i273–i282 (2003).
11. Slatkin, M. & Excoffier, L. *Heredity* **76**, 377–383 (1996).
12. Nesvizhskii, A.I., Keller, A., Kolker, E. & Aebersold, R. *Anal. Chem.* **75**, 4646–4658 (2003).
13. De Pierro, A.R. *IEEE Trans. Med. Imaging* **14**, 132–137 (1995).