Data Synthesis with Expectation-Maximization

Aaron Hertzmann
Dynamic Graphics Project
University of Toronto
www.dgp.toronto.edu/~hertzman

Technical report TR-DGP-2004-001

June 30, 2004

Abstract

A problem of increasing importance in computer graphics is to generate data with the style of some previous training data, but satisfying new constraints. If we use a probabilistic latent variable model, then learning the model will normally be performed using Expectation-Maximization (EM), or one of its generalizations. We show that data synthesis for such problems can also be performed using EM, and that this synthesis process closely parallels learning, including identical E-step algorithms. This observation simplifies the process of developing synthesis algorithms for latent variable models.

1 Introduction

A problem of increasing importance in computer graphics is to generate data with the statistics or style of some previous training data, but satisfying new constraints. For example, in character animation, we may wish to create new animations satisfying user-specified positional constraints, in the style of some original training data.

A powerful way to describe the statistics and styles is to write a PDF $p(\mathbf{x}|\theta)$ over data, where $\mathbf{x}$ denotes a data vector (e.g., a character pose, an animation sequence, a 3D model, etc.), and θ describe model parameters. The likelihood of a collection of data vectors (e.g., training data is) $\prod_i p(\mathbf{x}_i|\theta)$.

In a *latent variable model*, the likelihood of a data vector is described in terms of an additional *latent variable* $\mathbf{h}_i$ (or "hidden variable")

$$\begin{aligned} p(\mathbf{x}|\theta) &= \sum_j p(\mathbf{x}, \mathbf{h} = h_j|\theta) & (1) \\ &= \sum_j p(\mathbf{x}|\mathbf{h} = h_j, \theta) p(\mathbf{h} = h_j|\theta) & (2) \end{aligned}$$

where $\{h_j\}$ are the possible values of $\mathbf{h}$. If $\mathbf{h}$ is a continuous variable, then the summation is replaced by integration of the possible values of $\mathbf{h}$. Many popular models — including Mixtures-of-Gaussians, Linear Dynamical Systems, and Hidden Markov Models —- can be described as latent variable models [11].

In a typical application, there will be two phases:

1. **Learning**. Given training data $\{\mathbf{x}_i\}$, estimate the model θ.

2. **Synthesis**. Given a model θ, and new constraints on $\mathbf{x}$, estimate $\mathbf{x}$

The synthesis step may pose numerical and computational difficulties if the latent variables are unconstrained. Previous work avoided this difficulty by using prior estimates of the hidden variables [2, 3, 6] or user-selected values [8]; such approximations are difficult to come by in the presence of external constraints on the output (such as keyframe constraints). Wang et al. [12] use EM in motion synthesis; in our analysis, their method becomes a special case.

One of the most effective algortihms for learning in latent variable models is the Expectation-Maximization algorithm [4] and its generalizations. In this note, we show that data synthesis for such problems can also be performed using EM and that this synthesis process closely parallels learning. Specifically, the E-step of synthesis is identical to the E-step in learning, and the M-step of synthesis optimizes the same objective function as the M-step of learning. These observations simplify the process of developing synthesis algorithms for latent variable models. Our derivation is based on the free energy formulation of EM [7, 9], and follows from the observation that the free energy is the same for both learning and synthesis.

2 Learning with EM

Learning θ by maximum likelihood is achieved by optimizing:

$$\begin{aligned} \theta^* &= \arg\max_\theta \prod_i p(\mathbf{x}_i|\theta) && (3)\\ &= \arg\min_\theta \mathcal{L}(\theta) && (4) \end{aligned}$$

where

$$\mathcal{L}(\theta) \equiv -\sum_i \ln p(\mathbf{x}_i|\theta) \tag{5}$$

In latent variable models, this objective function is:

$$\mathcal{L}(\theta) \equiv -\sum_i \ln \sum_j p(\mathbf{x}_i, \mathbf{h}_i = h_j|\theta) \tag{6}$$

It often happens that this objective function is difficult or expensive to work with, to optimize or even to evaluate. In these cases, the EM algorithm can be derived by introducing *variational parameters* γ_{ij}. These parameters are free parameters, subject only to the constraint that $\sum_j \gamma_{ij} = 1$.

We can derive the following variational bound [7, 9]:

$$\begin{aligned} \mathcal{L}(\theta) &= -\sum_i \ln \sum_j p(\mathbf{x}_i, \mathbf{h}_i = h_j|\theta) && (7)\\ &= -\sum_i \ln \sum_j p(\mathbf{x}_i, \mathbf{h}_i = h_j|\theta)\frac{\gamma_{ij}}{\gamma_{ij}} && (8)\\ &\leq -\sum_{ij} \gamma_{ij} \ln \frac{p(\mathbf{x}_i, \mathbf{h}_i = h_j|\theta)}{\gamma_{ij}} && (9) \end{aligned}$$

$$= -\sum_{ij} \gamma_{ij} \ln p(\mathbf{x}_i, \mathbf{h}_i = h_j|\theta) - \sum_{ij} \gamma_{ij} \ln \gamma_{ij} \tag{10}$$

$$\equiv \mathcal{F}(\theta, \gamma) \tag{11}$$

The bound follows from Jensen's inequality ($\ln \sum_i \lambda_i x_i \geq \sum_i \lambda_i \ln x_i$ if $\sum_i \lambda_i = 1$). The quantity $\mathcal{F}(\theta, \gamma)$ is called the variational free energy.

Suppose, for a given value of θ we want to minimize the free energy w.r.t. γ:

$$\gamma^* = \arg\min_\gamma \mathcal{F}(\theta, \gamma) \tag{12}$$

By solving $\frac{\partial}{\partial \gamma} \mathcal{F}(\theta, \gamma) = 0$ subject to the constraints $\sum_j \gamma_{ij} = 1$ (enforced using Lagrange multipliers), we obtain:

$$\gamma_{ij}^* = p(\mathbf{h}_i = h_j|\mathbf{x}_j, \theta) \tag{13}$$

In other words, the optimal variational parameters is the probability distribution over the latent variables, given the input data.

Substituting γ^* into the free energy and rearranging terms gives:

$$\mathcal{L}(\theta) = \mathcal{F}(\theta, \gamma^*) \tag{14}$$

$$= \min_\gamma \mathcal{F}(\theta, \gamma) \tag{15}$$

From this follows the key result, that minimizing $\mathcal{F}(\theta, \gamma)$ with respect to θ and γ is equivalent to minimizing $\mathcal{L}(\theta)$:

$$\min_\theta \mathcal{L}(\theta) = \min_{\theta, \gamma} \mathcal{F}(\theta, \gamma) \tag{16}$$

The EM algorithm alternates between minimizing the free energy with respect to γ and θ:

```
**loop** until convergence
    **E-step:** $\gamma \leftarrow \arg\min_\gamma \mathcal{F}(\theta, \gamma)$
    **M-step:** $\theta \leftarrow \arg\min_\theta \mathcal{F}(\theta, \gamma)$
```

At convergence, the estimate θ is guaranteed to be a local minimum of $\mathcal{L}(\theta)$.

3 Synthesis with EM

Given a model θ and constraints C, we can define $p(\mathbf{x}|\theta, C) = p(\mathbf{x}|\theta) p(\mathbf{x}|C)/Z$ where Z is a normalization constant. This might correspond to a model in which the constraints C are observed as a function of the unknown $\mathbf{x}$, i.e., according to $p(\mathbf{C}|\mathbf{x})$. These constraints could be provided by user interaction, by observations, or other computations. Then, we wish to generate data by maximizing the probability of the data and the constraints:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}|\theta, C) \tag{17}$$

$$
\begin{aligned}
&= \arg\max_{\mathbf{x}} p(\mathbf{x}|\theta)p(\mathbf{x}|C) && (18)\\
&= \arg\min_{\mathbf{x}} -\ln p(\mathbf{x}|\theta) - \ln p(\mathbf{x}|C) && (19)\\
&= \arg\min_{\mathbf{x}} -\ln \sum_j p(\mathbf{x}, \mathbf{h} = h_j|\theta) - \ln p(\mathbf{x}|C) && (20)\\
&\equiv \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}) && (21)
\end{aligned}
$$

If we have hard constraints instead of soft constraints, then the synthesis problem is:

$$
\begin{aligned}
\mathbf{x}^* &= \arg\min_{\mathbf{x}} -\ln p(\mathbf{x}|\theta) && (22)\\
& \quad \text{s.t. } C(\mathbf{x}) = 0 && (23)
\end{aligned}
$$

This corresponds to a constraint distribution that is a delta-function around $\mathbf{x}$ values that satisfy the constraints. For the rest of the discussion, we will assume soft constraints.

We can derive a free energy following the same derivation as before, obtaining in the end:

$$
\mathcal{F}(\mathbf{x}, \gamma) = -\sum_j \gamma_j \ln p(\mathbf{x}, \mathbf{h} = h_j|\theta) - \ln p(\mathbf{x}|C) - \sum_j \gamma_j \ln \gamma_j \tag{24}
$$

The variational parameter γ is now a vector (with $\sum_j \gamma_j = 1$), since we are only synthesizing a single $\mathbf{x}$ data point. We again have the useful property:

$$
\mathcal{L}(\mathbf{x}) = \min_{\gamma} \mathcal{F}_{\mathbf{x}}(\mathbf{x}, \gamma) \tag{25}
$$

The EM for synthesis algorithm is:

```
loop until convergence
    E-step: $\gamma \leftarrow \arg\min_\gamma \mathcal{F}(\mathbf{x}, \gamma)$
    M-step: $\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \mathcal{F}(\mathbf{x}, \gamma)$
```

Note that the free energy for synthesis is nearly identical to the free energy for learning, except for the additional constraints, and of course the fact that there is only one data vector. The two main results are:

1. **The E-step for synthesis is identical to the E-step for learning.**

2. **The M-step for synthesis optimizes the same objective function as M-step for synthesis, plus constraints.**

This parallel between the two algorithms simplifies development of synthesis algorithms — one may even use the same source code for the E-step in synthesis as for learning. The M-step will be different For example, to synthesize animations from a Hidden Markov Model, we would alternate between numerical optimization of the target animation, and an E-step that is identical to the standard Forward-Backward algorithm used for HMMs. In retrospect, it is clear that the analysis and synthesis E-steps used in [12] should be identical.

Similar intuitions apply to generalizations of EM. For example, if an exact E-step is intractable, then variational learning may be useful [7]. In this case, one may use the same approximate E-step for both learning and synthesis.

Another way of understanding the correspondence between learning and synthesis is to observe that both learning and synthesis are equivalent to maximizing the joint probability of the data and the model:

$$p(\mathbf{x}, \theta) = p(\mathbf{x}|\theta)p(\theta) = p(\theta|\mathbf{x})p(\mathbf{x}) \tag{26}$$

In other words: in learning, we compute $\arg\max_\theta p(\theta|\mathbf{x}) = \arg\max_\theta p(\mathbf{x}, \theta)$, and, in synthesis, we compute $\arg\max_\mathbf{x} p(\mathbf{x}|\theta) = \arg\max_\mathbf{x} p(\mathbf{x}, \theta)$. We can derive a single free energy that bounds this joint probability:

$$\mathcal{F}(\mathbf{x}, \theta, \gamma) \quad \geq \quad -\ln p(\mathbf{x}, \theta) \tag{27}$$

In learning with EM, we minimize this free energy w.r.t. $\mathbf{x}$ and γ, and, in synthesis, we minimize this same free energy w.r.t. θ and γ.

4 Example: Mixtures-of-Gaussians

As an example, we consider the Mixtures-of-Gaussian (MoG) model [1, 10]. An MoG consists of K Gaussian PDFs, with Gaussian j having mean μ_j and covariance ϕ_j. A K-dimensional vector π is provided, satisfying $\sum_{j=1}^K \pi_j = 1$. A vector is sampled from the MoG model by first selecting one of the Gaussians — Gaussian i is selected with probability π_j — and then sampling from that Gaussian. We can also label a data point with a latent variable L that indicates which Gaussian the data point was sampled from. More formally:

$$\begin{aligned} P(L = i|\theta) &= \pi_j & (28) \\ p(\mathbf{x}|L = i, \theta) &= \mathcal{N}(\mathbf{x}|\mu_j; \phi_j) & (29) \end{aligned}$$

where the parameter vector $\theta = [\pi, \mu, \phi]$ encapsulates the parameters of the MoG, and $\mathcal{N}(\mathbf{x}|\mu_j; \phi_j)$ denotes a multidimensional Gaussian PDF with mean μ_j and covariance ϕ_j:

$$\mathcal{N}(\mathbf{x}|\mu; \phi) = \frac{1}{\sqrt{(2\pi)^d |\phi|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \phi^{-1} (\mathbf{x} - \mu)\right) \tag{30}$$

where d is the dimensionality of the data vector. We can write the entire MoG PDF by marginalizing over L:

$$\begin{aligned} p(\mathbf{x}|\theta) &= \sum_{j=1}^K p(\mathbf{x}, L = i|\theta) = \sum_{j=1}^K p(\mathbf{x}|L = i, \theta) P(L = i|\theta) & (31) \\ &= \sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\mu_j; \phi_j) & (32) \end{aligned}$$

Hence, the complete PDF $p(\mathbf{x}|\theta)$ is a linear combination of Gaussians.

The free energy for the MoG model and N data points $\mathbf{x}_i$ is given by:

$$\begin{aligned} \mathcal{F}(\{\mathbf{x}_i\}, \theta, \gamma) &= \frac{1}{2}\sum_{ij} \gamma_{ij} (\mathbf{x}_i - \mu_j)^T \phi_j^{-1} (\mathbf{x}_i - \mu_j) + \frac{1}{2}\sum_{ij} \gamma_{ij} \ln (2\pi)^d |\phi_j| - \sum_{ij} \gamma_{ij} \ln \pi_j & (33) \\ &\quad - \sum_{ij} \gamma_{ij} \ln \gamma_{ij} & (34) \end{aligned}$$

In learning, we alternate between minimization w.r.t. γ, and with respect to θ. The E-step update is:

$$\gamma_{ij} \quad \leftarrow \quad \frac{\pi_j \mathcal{N}(\mathbf{x}_i|\mu_j;\phi_j)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_i|\mu_j;\phi_j)} \tag{35}$$

In the M-step, the update to θ is:

$$\pi_j \quad \leftarrow \quad \sum_{j=1}^N \gamma_{ij}/N \tag{36}$$

$$\mu_j \quad \leftarrow \quad \sum_{j=1}^N \gamma_{ij}\mathbf{x}_i / \sum_j \gamma_{ij} \tag{37}$$

$$\phi_j \quad \leftarrow \quad \sum_{j=1}^N \gamma_{ij}(\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T / \sum_j \gamma_{ij} \tag{38}$$

Note that π_j is computed as the proportion of data points labeled with Gaussian i; μ_j is the weighted mean of the data points; and ϕ_j is the weighted covariance of the data points.

In synthesis, we wish to generate a single data vector $\mathbf{x}$ subject, given a model $\mathbf{x}$ and some additional constraints C. In other words, our goal is to minimize the free energy w.r.t. $\mathbf{x}$ and γ subject to $C(\mathbf{x}) = 0$. The E-step is the same as before, except that γ is now a vector (since there is only one data point):

$$\gamma_j \quad \leftarrow \quad \frac{\pi_j \mathcal{N}(\mathbf{x}|\mu_j;\phi_j)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}|\mu_j;\phi_j)} \tag{39}$$

For the M-step, the free energy can be written as:

$$\mathbf{A} \quad = \quad \sum_j \gamma_j \phi_j^{-1} \tag{40}$$

$$\mathbf{b} \quad = \quad \sum_j \gamma_j \phi_j^{-1} \mu_j \tag{41}$$

$$\mathcal{F}(\mathbf{x},\gamma) \quad = \quad \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{42}$$

plus constant terms that do not depend on $\mathbf{x}$. The M-step entails minimization of this quadratic objective function w.r.t. $\mathbf{x}$, s.t. $C(\mathbf{x}) = 0$. If $\mathbf{x}$ is unconstrained, then the M-step update is:

$$\mathbf{x} \quad \leftarrow \quad \mathbf{A}^{-1}\mathbf{b} \tag{43}$$

5 Generalizations

The observations here apply as well to the various generalizations of the EM algorithm, such as variational learning, Monte Carlo EM, and EM-ECG. In each case, the generalized E-steps used in synthesis mirror those used in learning.

Acknowledgments

These observations arose from an early version of [5] (in which human poses were generated from Mixtures-of-Gaussian models) developed at University of Washington in collaboration with Keith Grochow, David Hsu, Eugene Hsu, Steven L. Martin, and Zoran Popović. Thanks to Nebojsa Jojic and Li Zhang for helping me understand variational learning.

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] Matthew Brand. Voice Puppetry. *Proceedings of SIGGRAPH 99*, pages 21–28, August 1999.

[3] Matthew Brand and Aaron Hertzmann. Style machines. *Proceedings of SIGGRAPH 2000*, pages 183–192, July 2000.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[5] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-Based Inverse Kinematics. *ACM Transactions on Graphics*, August 2004. (Proc. SIGGRAPH).

[6] Tony Jebara and Alex Pentland. Statistical Imitative Learning from Perceptual Data. *Proc. ICDL 02*, June 2002.

[7] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[8] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002. (Proc. SIGGRAPH 2002).

[9] Radford M. Neal and Geoff E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[10] Richard A. Redner and Homer F. Walker. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, 26(2), April 1984.

[11] Sam Roweis and Zoubin Ghahramani. A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2), February 1998.

[12] Tienshu Wang, Nan-Ning Zheng, Yan Li, Ying-Qing Xu, and Heung-Yeung Shum. Learning Kernel-based HMMs for Dynamic Sequence Synthesis. *Proc. Pacific Graphics*, pages 87–95, 2002.