Select and Sample – A model of efficient neural inference and learning

Jacquelyn Shelton

Frankfurt Institute for Advanced Studies

with Jörg Bornschein, Saboor Sheikh, Pietro Berkes, Jörg Lücke

Feb 15th, 2012
Introduction

- **Experimental neuroscience evidence**: perception encodes and maintains posterior probability distributions over possible causes of sensory stimuli
- Most likely stimulus interpretation(s) + associated uncertainty
Introduction – Motivation

- Full posterior representation costly/complex – very high-dimensional, multi-modal, possibly highly correlated
- But, the brain can nevertheless perform rapid learning and inference
- Evidence for fast feed-forward processing and recurrent processing
Introduction – Motivation

Questions:

▶ Can we find rich representation of the posterior for very high-dimensional spaces?
▶ This goal believed to be shared by the brain, can find a biologically plausible solution reaching it?

Goals:

▶ Want: method to combine feed-forward processing and recurrent stages of processing
▶ Idea: formulate these 2 ideas as approximations to exact inference in a probabilistic framework
The Setting

- **Probabilistic generative model** with latent causes/objects \( \vec{s} = (s_1, \ldots, s_H) \) for sensory data \( \vec{y} = (y_1, \ldots, y_D) \), and parameters \( \Theta \):

\[
p(\vec{y} | \Theta) = \sum_{\vec{s}} p(\vec{y} | \vec{s}, \Theta) p(\vec{s} | \Theta)
\]

- **Optimization problem**: given data set \( Y = \{\vec{y}_1, \ldots, \vec{y}_N\} \) find maximum likelihood parameters \( \Theta^* \):

\[
\Theta^* = \arg\max_{\Theta} p(Y | \Theta)
\]

using expectation maximization (EM).
Maximize objective function $\mathcal{L}(\Theta) = \log p(Y \mid \Theta)$ w.r.t. $\Theta$ by optimizing a lower bound, the free-energy,

$$
\mathcal{L}(\Theta) \geq \mathcal{F}(\Theta, q) = \sum_s q(\vec{s} \mid \Theta) \log \frac{p(\vec{y}, \vec{s} \mid \Theta)}{p(\vec{s} \mid \Theta)} \\
= \langle \log p(\vec{y}, \vec{s}) \rangle_{q(\vec{s} \mid \Theta)} + H[q(\vec{s})]
$$

...using EM: iteratively optimize $\mathcal{F}(\Theta, q),$

- **E-step:** estimate posterior distribution $q$, parameters fixed
  
  $$
  \arg\max_{q(\vec{s} \mid \Theta)} \mathcal{F}(\Theta, q) \rightarrow q_n(\vec{s} \mid \Theta) := p(\vec{s}^{(n)} \mid \vec{y}^{(n)}, \Theta)
  $$

- **M-step:** estimate model parameters, $q$ fixed
  
  $$
  \arg\max_{\Theta} \mathcal{F}(\Theta, q) \rightarrow \Theta := \arg\max_{\Theta} \langle \log p(\vec{y}, \vec{s}) \rangle_{q(\vec{s} \mid \Theta)}
  $$
The Setting - EM example

**Mixture of Gaussians:** using EM iteratively optimize $\mathcal{F}(\Theta, q)$:

Task: cluster data into 2 classes/Gaussians → Initialize parameters randomly before iterating E- and M-steps
**Mixture of Gaussians**: using **EM** iteratively optimize $\mathcal{F}(\Theta, q)$:

Iteration 1:

**E-step**: estimate posterior distribution $q$, parameters fixed

$$\text{argmax } \mathcal{F}(\Theta, q) \rightarrow q_n(\vec{s} | \Theta) := p(\vec{s}(n) | \vec{y}(n), \Theta)$$
The Setting - EM example

Mixture of Gaussians: using EM iteratively optimize $\mathcal{F}(\Theta, q)$:

Iteration 1:

M-step: estimate model parameters, $q$ fixed

$$\argmax_{\Theta} \mathcal{F}(\Theta, q) \rightarrow \Theta := \argmax_{\Theta} \langle \log p(\mathbf{y}, \mathbf{s}) \rangle_{q(\mathbf{s} | \Theta)}$$
The Setting - EM example

Mixture of Gaussians: using EM iteratively optimize $\mathcal{F}(\Theta, q)$:

Iteration 5:

**E-step**: estimate posterior distribution $q$, parameters fixed

$$\arg\max_{q(n)} \mathcal{F}(\Theta, q) \rightarrow q_n(\bar{s}|\Theta) := p(\bar{s}^{(n)}|\bar{y}^{(n)}, \Theta)$$

**M-step**: estimate model parameters, $q$ fixed

$$\arg\max_{\Theta} \mathcal{F}(\Theta, q) \rightarrow \Theta := \arg\max_{\Theta} \langle \log p(\bar{y}, \bar{s}) \rangle_{q(\bar{s}|\Theta)}$$
M-step usually involves a small number of expected values w.r.t. the posterior distribution:

\[
\langle g(\vec{s}) \rangle_{p(\vec{s} | \vec{y}(n), \Theta)} = \sum_{\vec{s}} p(\vec{s} | \vec{y}(n), \Theta) g(\vec{s})
\]

where \( g(\vec{s}) \) e.g. elementary function of hidden variables
- \( g(\vec{s}) = \vec{s} \) or \( g(\vec{s}) = \vec{s}\vec{s}^T \) for standard sparse coding

Computation of expectations is usually the computationally demanding part
**Approach: Select and Sample**

Method of attack: approximate expectation values in 2 ways

1. **Selection** \(\approx\) feed-forward processing: Restrict approximate posterior to pre-selected states:

2. **Sampling** \(\approx\) recurrent processing: approximate expectations using samples from the posterior distribution in a Monte Carlo estimate of expectations
Approach: Select and Sample

1. Selection \(\approx\) feed-fwd: Restrict approximate posterior to pre-selected states:

\[
q_n(\bar{s}; \Theta) = \sum_{\bar{s}' \in K_n} p(\bar{s}' | \bar{y}(n), \Theta) \delta(\bar{s} \in K_n)
\]

Choose set \(K_n\) w/ selection function \(S_h(\bar{y}, \Theta)\); efficiently selects candidates \(s_h\) with most posterior mass:

Efficiently compute expectations in \(O(|K_n|)\):

\[
\langle g(\bar{s}) \rangle_{p(\bar{s} | \bar{y}(n), \Theta)} \approx \langle g(\bar{s}) \rangle_{q_n(\bar{s}; \Theta)} = \frac{\sum_{\bar{s} \in K_n} p(\bar{s}, \bar{y}(n) | \Theta) g(\bar{s})}{\sum_{\bar{s}' \in K_n} p(\bar{s}', \bar{y}(n) | \Theta)}
\]
Method of attack: approximate expectation values in 2 ways

- **2. Sampling** \(\approx\) recurrent processing: approximate expectations using samples from the posterior distribution in a Monte Carlo estimate:

\[
\langle g(\vec{s}) \rangle_{p(\vec{s} | \vec{y}^{(n)}, \Theta)} \approx \frac{1}{M} \sum_{m=1}^{M} g(\vec{s}^{(m)})
\]

with \(\vec{s}^{(m)} \sim p(\vec{s} | \vec{y}, \Theta)\)

- Obtaining samples from true posterior often difficult
Approach: Select and Sample

Method of attack: approximate expectation values in 2 ways

- Combine **Selection + Sampling**: approx. using samples from the **truncated** distribution:

\[
\langle g(\vec{s}) \rangle_{\mathcal{K}_n} \approx \frac{1}{M} \sum_{m=1}^{M} g(\vec{s}(m))
\]

with \( \vec{s}(m) \sim q_n(\vec{s}; \Theta) \)

- Subspace \( \mathcal{K}_n \) is small, allowing MCMC algorithms to operate more efficiently, i.e. shorter burn-in times, reduced number of required samples
**Example application** - **Binary sparse coding**

Apply *select and sample* - sparse coding model with binary latents:

\[
p(\vec{s}|\pi) = \prod_{h=1}^{H} \pi^{s_h}(1 - \pi)^{1-s_h}
\]

\[
p(\vec{y}|\vec{s}, W, \sigma) = \mathcal{N}(\vec{y}; W\vec{s}, \sigma^2 I)
\]

- \( \vec{y} \in \mathbb{R}^D \) observed variables
- \( \vec{s} \in \{0, 1\}^H \) hidden variables
- \( W \in \mathbb{R}^{D \times H} \) dictionary

\[
p(\vec{y}|\Theta) = \sum_{s} \mathcal{N}(\vec{y}; W\vec{s}, \sigma^2 I) \prod_{h=1}^{H} \pi^{s_h}(1 - \pi)^{1-s_h}
\]

**Selection function**: cosine similarity - take \( H' \) highest scored \( s_h \) with:

\[
S_h(\vec{y}^{(n)}) = \frac{\vec{W}_h^T \vec{y}^{(n)}}{||\vec{W}_h||}
\]
Example application - Binary sparse coding

- **Inference:** selection + Gibbs sampling; selection posterior equivalent to full post. with only selected dims

\[
p(s_h = 1 \mid \vec{s}_{\setminus h}, \vec{y}) = \frac{p(s_h = 1, \vec{s}_{\setminus h}, \vec{y})^\beta}{p(s_h = 0, \vec{s}_{\setminus h}, \vec{y})^\beta + p(s_h = 1, \vec{s}_{\setminus h}, \vec{y})^\beta}
\]

- **Complexity of E-step (all 4 BSC cases):**

\[
\mathcal{O} \left( N S \left( \frac{D}{p(\vec{s}, \vec{y})} + \frac{1}{\langle \vec{s} \rangle} + \frac{H}{\langle \vec{s} \vec{s}^T \rangle} \right) \right)
\]

where \( S \) is \# of evaluated hidden states
Experiments - 1. Artificial data

- **Goal**: observe convergence behavior; sanity check for our method with ground-truth

- **Data**: \( N = 2000 \) bars data consisting of \( D = 6 \times 6 = 36 \) pixels with \( H = 12 \) bars:

- **Experiments**: binary sparse coding with:
  - (1) exact inference, (2) selection alone,
  - (3) sampling alone, (4) selection + sampling
Experiments - 1. Artificial data

Convergence behavior of 4 methods

- $\mathcal{L}(\theta)$
- $\mathcal{L}(\theta_{ground-truth})$
- Dictionary elements $W_h$

- Shown: dotted line $\mathcal{L}(\theta_{ground-truth})$, dictionary elements $W_h$, and log-likelihood for multiple runs over 50 EM steps for all 4 methods

$\rightarrow$ select and sample extracts GT parameters; likelihood converges
Experiments – 2. Natural image patches


- Data: $N = 40,000$ image patches with $D = 26 \times 26 = 676$ pixels, with $H = 800$ hidden dimensions:

- Experiments: binary sparse coding with $12 \leq H' \leq 36$ for all inference methods:
  (1) selection alone, (2) sampling alone,
  (3) selection + sampling
Experiments - 2. Natural image patches

Evaluation of select and sample approach

- Shown: end approx. log-likelihood after 100 EM-steps vs. # samples per data point and # states must evaluate for $H' = 20$

$S = 200 \times H'$

$\rightarrow$ 200 samples/hid dimension sufficient: $\leq 1\%$ likelihood increase

$\rightarrow$ Select and sample $\times 40$ faster than sampling
Experiments - 3. Large scale on image patches

- **Goals:** large scale using \# of samples determined in exp 2

- **Data:** $N = 500,000$ image patches $D = 40 \times 40 = 1600$ pixels, with $H = 1600$ hidden dimensions and $H' = 34$

- **Experiments:** binary sparse coding for:
  1. selection alone,
  2. sampling alone,
  3. selection + sampling
Experiments - 3. Large scale on image patches

1600 latent dimensions with sampling-based posterior

- Shown: handful of the inferred basis functions $W_h$ and comparison of computational complexity for selection and sample

→ Select and sample scales linearly with $H'$; selection exponentially
Summary

To summer-ize...

- Method scales well to high dimensional data (i.e. $H = 1600$)
- ...while maintaining sampling-based representation of posterior
- All model parameters learnable
- Combined approach represents reduced complexity and increased efficiency

Future/current:

- Generalized sparse coding
  - continuous hidden variables
  - compare diff inference methods (other variational, samplers)
- Generalized select-and-sample approach
  - try with other models
Thanks for your attention! Questions?


Appendix – Free-energy for latent variable models

Observed data $\mathcal{X} = \{x_i\}$; Latent variables $\mathcal{Y} = \{y_i\}$; Parameters $\theta$.

**Goal:** Maximize the log likelihood (i.e. ML learning) wrt $\theta$:

$$\ell(\theta) = \log P(\mathcal{X}|\theta) = \log \int P(\mathcal{Y}, \mathcal{X}|\theta) d\mathcal{Y},$$

Any distribution, $q(\mathcal{Y})$, over the hidden variables can be used to obtain a lower bound on the log likelihood using Jensen’s inequality:

$$\ell(\theta) = \log \int q(\mathcal{Y}) \frac{P(\mathcal{Y}, \mathcal{X}|\theta)}{q(\mathcal{Y})} d\mathcal{Y} \geq \int q(\mathcal{Y}) \log \frac{P(\mathcal{Y}, \mathcal{X}|\theta)}{q(\mathcal{Y})} d\mathcal{Y} \overset{\text{def}}{=} F(q, \theta).$$

Now,

$$\int q(\mathcal{Y}) \log \frac{P(\mathcal{Y}, \mathcal{X}|\theta)}{q(\mathcal{Y})} d\mathcal{Y} = \int q(\mathcal{Y}) \log P(\mathcal{Y}, \mathcal{X}|\theta) d\mathcal{Y} - \int q(\mathcal{Y}) \log q(\mathcal{Y}) d\mathcal{Y}$$

$$= \int q(\mathcal{Y}) \log P(\mathcal{Y}, \mathcal{X}|\theta) d\mathcal{Y} + H[q],$$

where $H[q]$ is the entropy of $q(\mathcal{Y})$.

So:

$$F(q, \theta) = \langle \log P(\mathcal{Y}, \mathcal{X}|\theta) \rangle_{q(\mathcal{Y})} + H[q]$$
The free energy can be re-written

\[ F(q, \theta) = \int q(\mathcal{Y}) \log \frac{P(\mathcal{Y}, \mathcal{X}|\theta)}{q(\mathcal{Y})} \, d\mathcal{Y} \]

\[ = \int q(\mathcal{Y}) \log \frac{P(\mathcal{Y} | \mathcal{X}, \theta) P(\mathcal{X} | \theta)}{q(\mathcal{Y})} \, d\mathcal{Y} \]

\[ = \int q(\mathcal{Y}) \log P(\mathcal{X} | \theta) \, d\mathcal{Y} + \int q(\mathcal{Y}) \log \frac{P(\mathcal{Y} | \mathcal{X}, \theta)}{q(\mathcal{Y})} \, d\mathcal{Y} \]

\[ = \ell(\theta) - \text{KL}[q(\mathcal{Y}) || P(\mathcal{Y} | \mathcal{X}, \theta)] \]

The second term is the Kullback-Leibler divergence.

This means that, for fixed \( \theta \), \( F \) is bounded above by \( \ell \), and achieves that bound when \( \text{KL}[q(\mathcal{Y}) || P(\mathcal{Y} | \mathcal{X}, \theta)] = 0 \).

But \( \text{KL}[q || p] \) is zero if and only if \( q = p \). So, the E step simply sets

\[ q^{(k)}(\mathcal{Y}) = P(\mathcal{Y} | \mathcal{X}, \theta^{(k-1)}) \]

and, after an E step, the free energy equals the likelihood.
M-step equations for binary sparse coding:

\[ W^{\text{new}} = \left( \sum_{n=1}^{N} \vec{y}(n) \langle \vec{s} \rangle_{q_n}^T \right) \left( \sum_{n=1}^{N} \langle \vec{s} \vec{s}^T \rangle_{q_n} \right)^{-1}, \]

\[ (\sigma^2)^{\text{new}} = \frac{1}{ND} \sum_n \langle \left\| \vec{y}(n) - W \vec{s} \right\|^2 \rangle_{q_n}, \]

\[ \pi^{\text{new}} = \frac{1}{N} \sum_n | < \vec{s} >_{q_n} |, \text{ where } | \vec{x} | = \frac{1}{H} \sum_h x_h. \]

The EM iterations can be associated with neural processing by the assumption that neural activity represents the posterior over hidden variables (E-step), and that synaptic plasticity implements changes to model parameters (M-step).
Appendix - Select and Sample

▶ **Selection**: Restrict approximate posterior to pre-selected states:

\[
p(\vec{s} | \vec{y}^{(n)}, \Theta) \approx q_n(\vec{s}; \Theta) = \frac{p(\vec{s} | \vec{y}^{(n)}, \Theta)}{\sum_{\vec{s}' \in \mathcal{K}_n} p(\vec{s}' | \vec{y}^{(n)}, \Theta)} \delta(\vec{s} \in \mathcal{K}_n)
\]

▶ Choose set \( \mathcal{K}_n \) w/ selection function \( S_h(\vec{y}, \Theta) \); efficiently selects candidates \( s_h \) with most posterior mass:

\[
\mathcal{K}_n = \{ \vec{s} | \text{for all } h \notin \mathcal{I}_n : s_h = 0 \}
\]

where \( \mathcal{I}_n \) contains the \( H' \) indices \( h \) with the highest values of \( S_h(\vec{y}^{(n)}, \Theta) \), most likely contributors

▶ Can be seen as variational approximation to posterior

▶ Efficiently computable expectations in \( \mathcal{O}(|\mathcal{K}_n|) \):

\[
\langle g(\vec{s}) \rangle_{p(\vec{s} | \vec{y}^{(n)}, \Theta)} \approx \langle g(\vec{s}) \rangle_{q_n(\vec{s}; \Theta)} = \frac{\sum_{\vec{s} \in \mathcal{K}_n} p(\vec{s}, \vec{y}^{(n)} | \Theta) g(\vec{s})}{\sum_{\vec{s}' \in \mathcal{K}_n} p(\vec{s}', \vec{y}^{(n)} | \Theta)}
\]
Select and sample on $40 \times 40$ image patches

(a) Learned $W$ bases.

(b) Log-likelihood

(c) Learned $\sigma^2$

(d) Learned $\pi H'$.
Just a kitty

MATH
I don't even want to know what she's trying to solve.

icanhascheezburger.com by 🍔 & 🍔