Composing graphical models with neural networks for structured representations and fast inference

Matt Johnson, David Duvenaud, Alex Wiltschko, Bob Datta, Ryan Adams
image manifold
image manifold

depth video
深度
视频
图像
流形
depth video

image manifold
Recurrent neural networks? [1,2,3]

![LSTM unit](image1)

Probabilistic graphical models? [4,5,6]

![LSTM Autoencoder Model](image2)

---

<table>
<thead>
<tr>
<th>Probabilistic graphical models</th>
<th>Deep learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ structured representations</td>
<td>– neural net “goo”</td>
</tr>
<tr>
<td>+ priors and uncertainty</td>
<td>– difficult parameterization</td>
</tr>
<tr>
<td>+ data and computational efficiency</td>
<td>– can require lots of data</td>
</tr>
<tr>
<td>– rigid assumptions may not fit</td>
<td>+ flexible</td>
</tr>
<tr>
<td>– feature engineering</td>
<td>+ feature learning</td>
</tr>
<tr>
<td>– top-down inference</td>
<td>+ recognition networks</td>
</tr>
</tbody>
</table>
MAKE PGMS GREAT AGAIN
Modeling idea: graphical models on latent variables, neural network models for observations

Inference: recognition networks output conjugate potentials, then apply fast graphical model inference

Application: learn syllable representation of behavior from video
Modeling idea: graphical models on latent variables, neural network models for observations
\[ \pi = \begin{bmatrix} \pi^{(1)} & \pi^{(2)} & \pi^{(3)} \end{bmatrix} \]

\[ z_{t+1} \sim \pi^{(z_t)} \]

\[ x_{t+1} = A^{(z_t)} x_t + B^{(z_t)} u_t \quad u_t \overset{iid}{\sim} \mathcal{N}(0, I) \]
\[ \pi = \begin{bmatrix} \pi^{(1)} & \pi^{(2)} & \pi^{(3)} \end{bmatrix} \]

\[ A^{(1)} \quad A^{(2)} \quad A^{(3)} \]

\[ B^{(1)} \quad B^{(2)} \quad B^{(3)} \]

\[ z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow z_4 \rightarrow z_5 \rightarrow z_6 \rightarrow z_7 \]

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6 \rightarrow x_7 \]
\[ y_t \mid x_t, \gamma \sim \mathcal{N}(\mu(x_t; \gamma), \Sigma(x_t; \gamma)) \]
\[
p(\theta) \quad \text{conjugate prior on global variables}
\]
\[
p(x | \theta) \quad \text{exponential family on local variables}
\]
\[
p(\gamma) \quad \text{any prior on observation parameters}
\]
\[
p(y | x, \gamma) \quad \text{neural network observation model}
\]
Inference?
$p(x \mid \theta)$ is linear dynamical system
$p(y \mid x, \theta)$ is linear-Gaussian
$p(\theta)$ is conjugate prior

$$
\mathcal{L}[q(\theta)q(x)] \triangleq \mathbb{E}_{q(\theta)q(x)} \left[ \log \frac{p(\theta, x, y)}{q(\theta)q(x)} \right]
$$

$q(\theta) \leftrightarrow \eta_\theta \quad q(x) \leftrightarrow \eta_x$
Proposition (natural gradient SVI of Hoffman et al. 2013)

\[ \tilde{\nabla} \mathcal{L}_{\text{SVI}}(\eta_\theta) = \eta_\theta^0 + \mathbb{E}_{q^*(x)}(t_{xy}(x,y), 1) \]
\( p(x \mid \theta) \) is linear dynamical system
\( p(y \mid x, \theta) \) is linear-Gaussian
\( p(\theta) \) is conjugate prior

\[
\mathcal{L}(\eta, \eta_x) \triangleq \mathbb{E}_{q(\theta)q(x)} \left[ \log \frac{p(\theta, x, y)}{q(\theta)q(x)} \right]
\]

\[
\eta_x^*(\eta) \triangleq \arg \max_{\eta_x} \mathcal{L}(\eta, \eta_x) \quad \mathcal{L}_{SVI}(\eta) \triangleq \mathcal{L}(\eta, \eta_x^*(\eta))
\]

Proposition (natural gradient SVI of Hoffman et al. 2013)

\[
\nabla \mathcal{L}_{SVI}(\eta) = \eta_0^0 + \sum_{n=1}^{N} \mathbb{E}_{q^*(x_n)}(t_{xy}(x_n, y_n), 1) - \eta_0
\]
Step 1: compute evidence potentials

Step 1: compute evidence potentials

Step 1: compute evidence potentials

Step 2: run fast message passing

Step 3: compute natural gradient

Natural gradient SVI

Variational autoencoders [1,2]

Structured VAEs

\[ q^*(x) \triangleq \arg \max_{q(x)} \mathcal{L}[q(\theta)q(x)] \]

\[ q^*(x) \triangleq \mathcal{N}(x \mid \mu(y; \phi), \Sigma(y; \phi)) \]

\[ q^*(x) \triangleq ? \]

+ optimal local factor

- suboptimal local factor

+ fast for general obs.

- \( \phi \) does all local inference

+ exploits conj. graph structure

- \( \phi \) does all local inference

+ exploits conj. graph structure

- no natural gradients

+ natural gradients

- no natural gradients

+ natural gradients on \( \eta_\theta \)


**Inference:** recognition networks output conjugate potentials, then apply fast graphical model inference.
\[
\mathcal{L}[q(\theta)q(\gamma)q(x)] \triangleq \mathbb{E}_{q(\theta)q(\gamma)q(x)} \left[ \log \frac{p(\theta,\gamma,x)p(y|x,\gamma)}{q(\theta)q(\gamma)q(x)} \right]
\]

\[
q(\theta) \leftrightarrow \eta_\theta \quad q(\gamma) \leftrightarrow \eta_\gamma \quad q(x) \leftrightarrow \eta_x
\]
\[ \mathcal{L}(\eta_\theta, \eta_\gamma, \eta_x) \triangleq \mathbb{E}_{q(\theta)q(\gamma)q(x)} \left[ \log \frac{p(\theta, \gamma, x)p(y \mid x, \gamma)}{q(\theta)q(\gamma)q(x)} \right] \]

\[ \hat{\mathcal{L}}(\eta_\theta, \eta_x, \phi) \triangleq \mathbb{E}_{q(\theta)q(\gamma)q(x)} \left[ \log \frac{p(\theta, \gamma, x)\exp\{\psi(x; y, \phi)\}}{q(\theta)q(\gamma)q(x)} \right] \]

where \( \psi(x; y, \phi) \) is a conjugate potential for \( p(x \mid \theta) \)

\[ \eta^*_x(\eta_\theta, \phi) \triangleq \arg \max_{\eta_x} \hat{\mathcal{L}}(\eta_\theta, \eta_x, \phi) \quad \mathcal{L}_{\text{SVAE}}(\eta_\theta, \eta_\gamma, \phi) \triangleq \mathcal{L}(\eta_\theta, \eta_\gamma, \eta^*_x(\eta_\theta, \phi)) \]
Fact (conjugate graphical models are easy)

The local variational parameter $\eta^*_x(\eta_\theta, \phi)$ is easy to compute.

Proposition (log evidence lower bound)

$\max_{\eta_x} \mathcal{L}(\eta_\theta, \eta_\gamma, \eta_x) \leq \log p(y)$

$\max_{\eta_x} \mathcal{L}_{\text{SVAE}}(\eta_\theta, \eta_\gamma, \phi) \leq \log p(y)$

if $\exists \phi \in \mathbb{R}^m$ with $\psi(x; y, \phi) = \mathbb{E}_{q(\gamma)} \log p(y \mid x, \gamma)$

Proposition (reparameterization trick)

Estimate $\nabla_{\eta_\gamma, \phi} \mathcal{L}_{\text{SVAE}}(\eta_\theta, \eta_\gamma, \phi)$ with samples $\hat{\gamma} \sim q(\gamma)$ and $\hat{x} \sim q^*(x \mid \phi)$ via

$\mathcal{L}_{\text{SVAE}}(\eta_\theta, \eta_\gamma, \phi) \approx \log p(y \mid \hat{x}, \hat{\gamma}) - \text{KL}(q(\theta)q(\gamma)q^*(x \mid \phi) \parallel p(\theta, \gamma, x))$

Proposition (easy natural gradient)

$\tilde{\nabla}_{\eta_\theta} \mathcal{L}_{\text{SVAE}}(\eta_\theta, \eta_\gamma, \phi) = (\eta^0_\theta + \mathbb{E}_{q^*(x \mid \phi)} (t_x(x), 1) - \eta_\theta) + (\nabla_{\eta_x} \mathcal{L}(\eta_\theta, \eta_\gamma, \eta^*_x(\eta_\theta, \phi)), 0)$
Step 1: apply recognition network
Step 1: apply recognition network
Step 1: apply recognition network

Step 2: run fast PGM algorithms

Step 3: sample, compute flat grads

Step 4: compute natural gradient
from autograd import value_and_grad as vgrad
from util import add, sub, scale, unbox

def make_gradfun(run_inference, recognize, loglike, eta_prior, return_flat=False):
    saved = lambda: None

    def mc_vlb(eta, gamma, phi, y_n, N, L):
        T = y_n.shape[0]
        nn_potentials = recognize(y_n, phi)
        samples, stats, global_kl, local_kl = run_inference(
            eta_prior, eta, nn_potentials, num_samples=L)
        saved.stats = scale(N, unbox(stats))
        return -global_kl + N * (-local_kl + loglike(y_n, samples, gamma))

    def gradfun(y_n, N, L, eta, gamma, phi):
        objective = lambda gamma, phi: mc_vlb(eta, gamma, phi, y_n, N, L)
        vlb, (gamma_grad, phi_grad) = vgrad(objective)((gamma, phi))
        eta_natgrad = sub(add(eta_prior, saved.stats), eta)
        return vlb, (eta_natgrad, gamma_grad, phi_grad)

    return gradfun
Application: learn syllable representation of behavior from video
fall from rear
grooming
Discovery of Heterozygous Phenotypes in Ror1b Mice

... and high and low doses of each drug

from Alex Wiltschko preprint
**Modeling idea:** graphical models on latent variables, neural network models for observations

**Inference:** recognition networks output conjugate potentials, then apply fast graphical model inference

**Application:** learn syllable representation of behavior from video
Limitations and future work

- **capacity**
  - How expressive is latent linear structure?
    - word embeddings [1], analogical reasoning in image models
    - SVAE can use nonlinear latent structure

- **complexity**
  - PGMs get complicated
    - SVAE keeps complexity modular

- **future work**
  - model-based reinforcement learning
  - automatic structure search [2,3]
  - semi-supervised applications

---

github.com/hips/autograd
Thanks!