Deep Generative Models with Learnable Knowledge Constraints

Overview

- Rich deep generative models (DGMs): GANs, VAEs, auto-regressive nets
- Difficult to exploit problem structures and domain knowledge (e.g., human body structure in image generation, Fig.1) in these DGMs.

Existing approaches:

- A popular way of adding structured knowledge with deep neural networks is to design specialized neural architectures
  - E.g., Conv-pooling architecture of ConvNet to hard-code translation-invariance of image classification
- Usually only applicable to specific knowledge, models, or tasks
- Posterior Regularization (PR) is a principled framework to impose knowledge constraints on posterior distributions of probabilistic models [1] or neural networks [2]. But with difficulties:
  - Many of the DGMs are not formulated with the probabilistic Bayesian framework and do not possess a posterior distribution or even meaningful latent variables
  - Require a priori fixed constraints. Users have to fully specify the constraints beforehand — impractical due to heavy engineering; suboptimal without adaptivity to the data and models.

This paper:

- A general means of incorporating arbitrary structured knowledge with any types of deep (generative) models in a principled way.
- Formal connections between PR and reinforcement learning (RL)
- Extends PR to learn constraints as the extrinsic reward in RL

Connecting Posterior Regularization (PR) to RL

1) (Adapted) PR for Deep Generative Models (DGMs)

- Consider a generative model \( x \sim p(x) \) with parameters \( \theta \)
- Consider constraint function \( f(x) \in \mathbb{R} \). A higher \( f(x) \) value indicates a better \( x \) in terms of the particular knowledge.
- PR assumes a variational distribution \( q \), and the objective:

\[
\min_{q_{\theta}} \mathcal{L}(\theta, q) = KL(q(x)\|p(x)) - \alpha E_q[f(x)],
\]

which is solved with an EM-style procedure:

E-step: \( q(x) = p(x) \exp\{\alpha f(x)/Z\} / Z \),

M-step: \( \min_{q_{\theta}} KL(q(x)\|p(x)) = \min_{q_{\theta}} -E_q[\log p(x)] + \text{const.} \) (2)

- In PR, constraint \( f \) is fixed. It’s sometimes desirable or necessary to enable learnable constraints so that practitioners are allowed to specify only the known components of \( f \) while leaving any unknown or uncertain components automatically learned (e.g., the human part parser in Fig.1).
- Denote the constraint function with learnable components as \( f_{\phi}(x) \)

2) Entropy-Regularized Policy Optimization (ERPO)

- ERPO augments policy gradient with information theoretic regularizers e.g., KL divergence between new and old policies for stabilized learning.
- Assume state \( s \), action \( a \), policy \( p_a(s,a) \), reward \( R(s,a) \in \mathbb{R} \)
- Let \( x = (s,a) \) denote the state-action pair, and \( p_\pi(x) = \mu^\pi(s)p_a(s,a) \) where \( \mu^\pi(s) \) is the stationary state distribution.
- Let \( q_\pi(x) \) be the new policy, \( p_\pi(x) \) the old. In some ERPO such as relative entropy policy search, \( q_\pi \) is non-parametric. Objective:

\[
\min_{q_\pi} \mathcal{L}(q_\pi) = KL(q_\pi(x)\|p_\pi(x)) - \alpha E_{q_\pi}[R(x)].
\]

Close resemblance between Eq. (1) and Eq. (3):

- Generative model \( p_\pi(x) \) in PR ⇔ reference (old) policy \( p_\pi(x) \)
- Constraint \( f \) in PR ⇔ reward \( R \)
- Solution for \( q_\pi \) is in the same form of Eq.(2)

3) Maximum-Entropy Inverse Reinforcement Learning (MaxEnt IRL)

- Learns reward \( R(\cdot) \) with unknown parameters \( \phi \)
- Assumes \( p_\phi \) a uniform \( \Rightarrow q_\phi(x) := \exp\{\alpha R(\cdot)/Z_\phi \} \). Learns \( \phi \) with:

\[
\phi^* = \arg \max_{\phi} E_{q_\phi}[\log q_\phi(x)].
\]

Algorithm

With the connection between PR and RL, we can transfer the MaxEnt IRL technique of reward learning for constraint learning. The resulting algorithm alternates the optimization of constraint \( f_\phi \) and model \( p_\pi \).

Learning the Constraint \( f_\phi \)

Use the same objective of MaxEnt IRL (Eq.1), replacing \( q_\pi \) with \( q(x) \) from Eq.2:

\[
\nabla_x E_{\pi(x)} \log q(x) = \nabla_x [E_{\pi(x)}[\alpha f_\phi(x)] - \log Z_\phi] = E_{\pi(x)}[\alpha \nabla_x f_\phi(x)] - E_{\phi} [\alpha \nabla_x f_\phi].
\]

Learning the Generative Model \( p_\pi \)

Given the current parameter state \( (\theta, \phi^*) \), and \( q(x) \) evaluated at the parameters, we continue to update the generative model.

- For explicit model, we use the M-step as in Eq.(2):

\[
\min_{q_{\theta}} KL(q(x)\|p(x)) = \min_{q_{\theta}} -E_q[\log p(x)] + \text{const.}
\]

- For implicit model that permits only simulating samples but not evaluating density, we propose to minimize the reverse KL divergence:

\[
\min_{q_{\theta}} KL(p_\pi(x)\|q(x)) = \min_{q_{\theta}} -E_{q_\pi}[\log p_\pi] + KL(p_\pi\|p_\phi) + \text{const.}
\]

See paper for efficient approximations and connections to GANs.

Experiments

<table>
<thead>
<tr>
<th>Method</th>
<th>SSIM</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy-based GAN</td>
<td>0.716</td>
<td>–</td>
</tr>
<tr>
<td>Base model</td>
<td>0.676</td>
<td>0.03</td>
</tr>
<tr>
<td>W/ fixed constraint</td>
<td>0.676</td>
<td>0.12</td>
</tr>
<tr>
<td>W/ learned constraint</td>
<td>0.727</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Table 2: Results of Human Pose Image Generation (Right, Fig.2) and Template Guided Sentence Generation (Right, Fig.2(b)). Pls see the paper for more details.

References