Maximum Entropy Inverse Reinforcement Learning

Michael Staniek & Philipp Wiesenbach & Marvin Koss

University of Heidelberg
Imitation Learning

October 16, 2018

Overview

Maximum Entropy IRL

- Expert optimizes reward value of trajectory ζ, which is the reward applied to the path's feature counts, $f_\zeta = \sum_{s_j \in \zeta} f_{s_j}$

$$reward(f_\zeta) = \theta^T f_\zeta = \sum_{s_j \in \zeta} \theta^T f_{s_j}$$

- demonstrated by single trajectories $\widetilde{\zeta}_i$
- m ζs provide expected empirical f count $\widetilde{f} = \frac{\sum_i f_{\widetilde{\zeta}_i}}{m}$

Maximum Entropy IRL

- We match feature expectations between observed policy and the learners behavior:

$$\sum_{Path \zeta_i} P(\zeta_i) f_{\zeta_i} = \tilde{f}$$

- Two ambiguities:
  - Each policy can be optimal for many reward functions
  - Many policies lead to same feature counts

Maximum Entropy

$\rightarrow$chooses distribution that only matches feature expectations and doesnt have additional preferences

$$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^T f_{s_j}}$$

Maximum Entropy IRL

Learning from Demonstrated Behavior

Maximizing the distribution's entropy while matching feature constraints means maximizing likelihood of observed data

$$\theta^* = \underset{\theta}{argmax} L(\theta) = \underset{\theta}{argmax} \sum_{examples} (\tilde{\zeta}|\theta)$$

Convex loss, use gradient descent:

$$\nabla L(\theta) = \tilde{f} - \sum_{\zeta} P(\tilde{\zeta}|\theta) f_{\zeta}$$

Global Normalization

Daniel Andor et al. (2016). "Globally Normalized Transition-Based Neural Networks". In: *CoRR* abs/1603.06042

Main Idea

Use globally normalized sequence probabilities and beam search during inference to enhance results on structured NLP tasks.

- Global normalization counteracts the Label Bias Problem (higher expressiveness of the model)
- Beam Search can be used as a function approximator (reducing calculation costs)
- Early updates (faster learning?)

Global Normalization

- Scoring function:

$$\rho(s, d; \theta) = \rho(d_{1:j-1}, d; \theta) = \phi(s; \theta^{(l)}) \cdot \theta^{(d)}$$

  - $\theta^{(l)}$ being network parameters without last layer
  - $\theta^{(d)}$ being parameters of last layer
  - $\phi(s; \theta^{(l)})$ being representation of state s
  - the score is *linear* under $\theta^{(d)}$

- Conditional probabilities:

$$p(d_j | d_{1:j-1}; \theta) = \frac{e^{\rho(d_{1:j-1}, d_j; \theta)}}{Z_L(d_{1:j-1}, d'; \theta)}$$

- with partition function:

$$Z_L(d_{1:j-1}, d'; \theta) = \sum_{d' \in \mathcal{A}(d_{i:j-1})} e^{\rho(d_{1:j-1}, d_j; \theta)}$$

Global Normalization

- Sequence probability (locally normalized):

$$p_L(d_{1:n}) = \prod_{j=1}^{n} p(d_j | d_{1:j-1}; \theta) \quad = \frac{exp \sum_{j=1}^{n} \rho(d_{1:j-1}, d_j; \theta)}{\prod_{j=1}^{n} Z_L(d_{1:j-1}, d'; \theta)}$$

- Sequence probability (globally normalized):

$$p_G(d_{1:n}) = \frac{exp \sum_{j=1}^{n} \rho(d_{1:j-1}, d_j; \theta)}{Z_G(\theta)}$$

$$with\ Z_G(\theta) = \sum_{d'_{1:n} \in \mathcal{D}_n} exp \sum_{j=1}^{n} \rho(d_{1:j-1}, d_j; \theta)$$

→ inference: only have to calculate argmax $\sum_{j=1}^{n} \rho(d_{1:j-1}, d_j; \theta)$

Global Normalization

Training objectives:

- Negative log-likelihood (locally normalized):

$$
\begin{aligned}
L_{local}(d^*_{1:n};\theta) &= -\ln p_L(d*_{1:n};\theta) \\
&= -\sum_{j=1}^{n}\rho(d_{1:j-1}, d_j;\theta) + \sum_{j=1}^{n}\ln Z_L(d^*_{1:n};\theta)
\end{aligned}
$$

- Negative log-likelihood (globally normalized):

$$
\begin{aligned}
L_{global}(d^*_{1:n};\theta) &= -\ln p_G(d*_{1:n};\theta) \\
&= -\sum_{j=1}^{n}\rho(d_{1:j-1}, d_j;\theta) + \ln Z_G(\theta)
\end{aligned}
$$

Global Normalization

- Approximating Partition Function $Z_G(\theta)$ using beam search
- If the *goldpath* falls out of the beam at step j, SGD is taken on the following objective:

$$L_{global-beam}(d^*_{1:n};\theta) =$$

$$-\sum_{j=1}^{n} \rho(d_{1:j-1}, d_j; \theta) + \ln \sum_{d'_{1:j} \in \mathcal{B}_j} exp \sum_{i=1}^{j} \rho(d'_{1:i-1}, d'_i; \theta)$$

What is our goal?

- We want to use the Andor et al. (2016) model for a recurrent encoder decoder NMT system to learn the reward function of humans
- Straightforward: Use the same beam search algorithm

Approaches

- $\rho(s, d; \theta) = \phi(s; \theta^{(l)}) \cdot \theta^{(d)}$
- backpropagate through everything a la Andor et al. (2016)
  $\overleftarrow{\text{Encoder} \rightarrow \text{Decoder}}$
- only backpropagate through last layer of the Decoder a la Ziebart et al. (2008) (linear model)
  $\text{Encoder} \rightarrow \overleftarrow{\text{Decoder}}$

Approaches

- Andor et al. (2016) gives two options: start from scratch or use pretrained model
  $\overleftarrow{\text{Encoder} \rightarrow \text{Decoder}}$
- Ziebart et al. (2008) has to be pretrained
  $\text{Encoder} \rightarrow \overleftarrow{\text{Decoder}}$

Alternatives for beam search

- NCE (maybe)
- MC sampling (no)

Framework of Choice

- Any framework thats based on PyTorch
- For example joey-nmt

Afterwards

- As soon as we have a trained reward estimator:
  - Use the trained reward estimator as is
  - Use it to train a deeper model with policy gradient and see what we can reach
- Here we will try both, hopefully

Literatur

- Andor, Daniel et al. (2016). "Globally Normalized Transition-Based Neural Networks". In: *CoRR* abs/1603.06042.
- Ng, Andrew Y. and Stuart J. Russell (2000). "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA, pp. 663–670.
- Ziebart, Brian D. et al. (2008). "Maximum entropy inverse reinforcement learning". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 3, pp. 1433–1438.