Probabilistic Perspectives on Meta and Reinforcement Learning

Yee Whye Teh
Statistics @ Oxford
DeepMind

http://csml.stats.ox.ac.uk/people/teh/
Reinforcement Learning
Efficient Learning from Multiple Tasks
Transferrable Structures

● Transferrable structure in policies/solutions
  ○ Learning prior policies using KL-regularised RL:
  ○ Neural probabilistic motor primitives
    ■ [Merel, Hasenclever et al ICLR 2019]

● Transferrable structure in environments
  ○ Meta-learning with neural processes:
Markov Decision Processes

\[ \pi(a_t | s_t) \]

\[ r_t, s_t \]

\[ p(s_{t+1} | s_t, a_t) \]

\[ r_t = r(s_t, a_t) \]

\[ \max \pi \mathbb{E}_\pi \left[ \sum_{t=1}^{\infty} \gamma^t r_t \right] \]
KL-regularised Reinforcement Learning

$$\max_{\pi} \mathbb{E}_\pi \left[ \sum_{t=1}^{\infty} \gamma^t \left( r(s_t, a_t) + \alpha \log \frac{\pi_0(a_t | s_t)}{\pi(a_t | s_t)} \right) \right]$$

- **expected rewards**
- **default behaviour**
- **KL divergence**
- **diverse solutions**
KL-regularised Reinforcement Learning

$$\max_\pi \mathbb{E}_\pi \left[ \sum_{t=1}^{\infty} \gamma^t \left( r(s_t, a_t) + \alpha \log \frac{\pi_0(a_t | s_t)}{\pi(a_t | s_t)} \right) \right]$$

- log likelihood
- prior
- KL divergence
- posterior
KL-regularised Reinforcement Learning

\[
\max_{\pi} \mathbb{E}_\pi \left[ \sum_{t=1}^{\infty} \gamma^t \left( r(s_t, a_t) + \alpha \log \frac{\pi_0(a_t \mid s_t)}{\pi(a_t \mid s_t)} \right) \right]
\]

\[
\pi^*(a_t \mid s_t) \propto \pi_0(a_t \mid s_t) \exp(Q^*(a_t \mid s_t))
\]
KL-regularised Reinforcement Learning

- Distribution \( p(w) \) over tasks \( w \):

\[
\max_{\pi} \sum_w p(w) \left[ \mathbb{E}_{\pi_w} \left[ \sum_{t=1}^{\infty} \gamma^t \left( r_w(s_t, a_t) + \alpha \log \frac{\pi_0(a_t | s_t)}{\pi_w(a_t | s_t)} \right) \right] \right]
\]

\[
\pi_w^*(a_t | s_t) \propto \pi_0(a_t | s_t) \exp(Q_w^*(a_t | s_t))
\]

\[
\pi_0^*(a_t | s_t) = \arg \max_{\pi_0} \sum_{w} p(w) \mathbb{E}_{\pi_w} \left[ \sum_t \gamma^t \log \pi_0(a_t | s_t) \right]
\]
Information asymmetry in KL-regularized RL

- Default policy trained alongside policy
- Default policy sees partial information
- “Information hiding” forces generalization

Expert policy

Baseline
- Proprio
- Proprio+Box
- Transfer Proprio
- Transfer Proprio+Box
Introducing Structure via Latent Variables

\[ \pi(\tau) = \int \pi(\tau | z) dz \]

 richer action distribution

 goal conditional policy

 temporal correlations — e.g. options

Yee Whye Teh
Hierarchical Structure in Policy and Prior

\[ \mathbb{E}_{\pi}(\tau)[\sum_t r_t] - \text{KL}[\pi(\tau) || \pi_0(\tau)] \geq \mathbb{E}_{\pi}(\tau)[\sum_t r_t] - \text{KL}[\pi(z) || \pi_0(z)] - \text{KL}[\pi(\tau | z) || \pi_0(\tau | z)] \]

Parameter sharing \( \pi = \pi_0 \)

"Reusable LL controller" / "Skills"
Hierarchical Structure in Policy and Prior

Prior
Hierarchical
Unstructured
None

Policy (Posterior)
Hierarchical Structure in Policy and Prior

- Prior
- Hierarchical
- Unstructured
- None

Low level controller trained on light box: Agent struggles to control heavy box

Separate LL

Shared LL

Yee Whye Teh
Neural Probabilistic Motor Primitives
Yee Whye Teh

\[
\begin{align*}
\log \pi_0(a_{1:T}|s_{1:T}) &= \int \pi_0(a_{1:T}|s_{1:T}, z_{1:T}) p_z(z_{1:T}) \, dz_{1:T} \\
&\geq \mathbb{E}_q \left[ \sum_{t=1}^{T} \log \pi_0(a_t|s_t, z_t) + B \left( \log p_z(z_t|z_{t-1}) - \log q(z_t|z_{t-1}, x_t) \right) \right]
\end{align*}
\]
Proprio

Vision

Motor intention

Action

Yee Whye Teh
Transferrable Structures

● Transferrable structure in policies/solutions
  ○ Learning prior policies using KL-regularised RL:
  ○ Neural probabilistic motor primitives
    ■ [Merel, Hasenclever et al ICLR 2019]

● Transferrable structure in environments
  ○ Meta-learning with neural processes:
Model-based Reinforcement Learning

\[ h_1 \rightarrow a_1, r_1 \]
\[ h_2 \rightarrow a_2, r_2 \]
\[ h_3 \rightarrow a_3, r_3 \]
\[ \vdots \]
\[ h_t \rightarrow a_t, r_t \]

\[ \hat{p}(s' | s, a) \]
\[ \hat{r}(s, a) \]

Plan

state, reward

action
Model-based Reinforcement Learning

\[ h_1 \rightarrow a_1, r_1 \]
\[ h_2 \rightarrow a_2, r_2 \]
\[ h_3 \rightarrow a_3, r_3 \]
\[ \vdots \]
\[ h_t \rightarrow a_t, r_t \]

\[ \hat{p}(s' \mid s, a) \]
\[ \hat{r}(s, a) \]

inputs x

outputs y
Model-based Reinforcement Learning

\[ h_1 \rightarrow a_1, r_1 \]
\[ h_2 \rightarrow a_2, r_2 \]
\[ h_3 \rightarrow a_3, r_3 \]
\[ \vdots \]
\[ h_t \rightarrow a_t, r_t \]

\[ \hat{p}(s' | s, a) \]
\[ \hat{r}(s, a) \]

\[ x_i \rightarrow y_i \]

Yee Whye Teh
Specifying Stochastic Processes

- Gaussian processes are typically described via marginal distributions:

\[
\begin{pmatrix}
    f(x_1) \\
    f(x_2) \\
    \vdots \\
    f(x_t)
\end{pmatrix}
\sim \mathcal{N}
\begin{pmatrix}
    \mu(x_1) \\
    \mu(x_2) \\
    \vdots \\
    \mu(x_t)
\end{pmatrix},
\begin{pmatrix}
    K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_t) \\
    K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_t) \\
    \vdots & \vdots & \ddots & \vdots \\
    K(x_t, x_1) & K(x_t, x_2) & \cdots & K(x_t, x_t)
\end{pmatrix}
\]
Specifying Stochastic Processes

- Gaussian processes can equivalently be described via its conditional distributions:

\[ f(x_{t+1}) \mid f(x_1) = y_1, \ldots, f(x_t) = y_t \]

\[ \sim \mathcal{N}(\mu(x_{t+1}) + K_{t+1,1:t} K_{1:t,1:t}^{-1} y_{1:t}, K_{t+1,t+1} - K_{t+1,1:t} K_{1:t,1:t} K_{1:t,t+1}) \]

- In general, stochastic processes can also be described using a consistent family of conditional distributions:

\[ \mathbb{P}(f(x_{t+1}) = y_{t+1} \mid f(x_1) = y_1, \ldots, f(x_t) = y_t) \]

for training sets \( \{x_{1:t}, y_{1:t}\} \) and test sets \( \{x_{t+1}, y_{t+1}\} \).
Learning Neural Stochastic Processes

- Use a neural network to parameterise the conditional distributions.
Learning Neural Stochastic Processes

\[
\begin{align*}
    x_1^1 &\rightarrow y_1^1 \\
    x_t^1 &\rightarrow y_t^1 \\
    x_{t+1}^1 &\rightarrow y_{t+1}^1 \\
    x_{t+s}^1 &\rightarrow y_{t+s}^1 \\
    x_1^2 &\rightarrow y_1^2 \\
    x_t^2 &\rightarrow y_t^2 \\
    x_{t+1}^2 &\rightarrow y_{t+1}^2 \\
    x_{t+s}^2 &\rightarrow y_{t+s}^2 \\
    x_1^3 &\rightarrow y_1^3 \\
    x_t^3 &\rightarrow y_t^3 \\
    x_{t+1}^3 &\rightarrow y_{t+1}^3 \\
    x_{t+s}^3 &\rightarrow y_{t+s}^3 \\
    x_1^4 &\rightarrow y_1^4 \\
    x_t^4 &\rightarrow y_t^4 \\
    x_{t+1}^4 &\rightarrow y_{t+1}^4 \\
    x_{t+s}^4 &\rightarrow y_{t+s}^4
\end{align*}
\]

\[
\begin{align*}
    x_1^w &\rightarrow y_1^w \\
    x_t^w &\rightarrow y_t^w \\
    x_{t+1}^w &\rightarrow y_{t+1}^w \\
    x_{t+s}^w &\rightarrow y_{t+s}^w
\end{align*}
\]

\[
\begin{align*}
    y_1^w &\rightarrow y_{t+1}^w \\
    y_2^w &\rightarrow y_{t+2}^w \\
    y_3^w &\rightarrow y_{t+3}^w \\
    \end{align*}
\]

\[
\max_{\eta} \sum_w p(w) \sum_{j=1}^{s} \log p_{\eta}(y_{t+j}^w \mid x_{t+j}^w, \{x_i^w, y_i^w\}_{i=1}^{t})
\]

- A probabilistic perspective on meta-learning.
Few Shot Image Classification

Training
- terrier
- beagle
- labrador
- cat
- poodle

Test
- ?
- ?
- ?
- ?
- ?

Credit: Andrei Rusu, ImageNet
Few Shot Image Classification via Meta-Learning

Training
- sloth
- lipstick
- hot dog
- barrette
- tank

Meta-training
- sloth
- lipstick
- hot dog
- barrette
- tank

Test
- hot dog
- sloth
- tank
- barrette
- lipstick

Credit: Andrei Rusu, miniImageNet
Optimization Perspective on Meta-Learning

Meta-parameter $\eta$

Training Data $\{(x_i, y_i)\}$

Learning Algorithm $A_\eta: \text{train } \Rightarrow \theta$

Predictor $f_{\eta,\theta}(x)$

Test Data

Test Loss

20%

0%

20%
Optimization Perspective on Meta-Learning

Meta-parameter $\eta$

Training Data
$\{(x_i, y_i)\}$

Learning Algorithm
$\text{train} \Rightarrow \theta$

Predictor
$f_{\eta, \theta}(x)$

Test Data

Test Loss

20%

0%

20%
Optimization Perspective on Meta-Learning

- Task $w$

Training Data → Learnt model Parameter $\theta$ → Evaluate

Meta-Parameter $\eta$

Test Data

Generative Process

Learning Process

Yee Whye Teh
Probabilistic Perspective on Meta-Learning

- Task \( w \)

Training Data
- Meta-Parameter \( \eta \)

Test Data
- Generative Process
- Learning Process

\[ P(w|\text{train}) \]

Yee Whye Teh
Variational Auto-Encoders

- Prior $p(Z)$
- Latent representation
- Encoder $q(Z|X)$
- Decoder $p(X|Z)$
- "Amortized inference"
- Generative model

Data $X$ flows through the encoder to the latent representation $Z$, which is then used by the decoder to reconstruct the original data $X$. This process allows for effective dimensionality reduction and generation of new data points.
Probabilistic Model for Meta-Learning

**Encoder q(Z|Training)**  
“Amortized learning”

**Task representation**

**Decoder p(Test|Z)**  
*Generative model*

**Training Data**

**Test Data**

**η**  
*meta-parameter*

*Images of tank, sloth, lipstick, hot dog, barrette*
Neural Processes

\[ \sum \]

encoding

\[ r_1 \]
\[ r_2 \]
\[ r_3 \]

outputs

inputs

Training data

Test data

inputs

outputs

meta-parameter

parameter

\[ Z \]

\[ \eta \]
Neural Processes

Task = Function on 1D space.

Given training points, use neural processes to predict mean and std of function values at other locations.
Cart Pole
Image Super-resolution

Task = Image = Function on 2D space.
Image Super-resolution

context

target prediction at different resolutions x 3

context

target prediction

Interpolation baselines

4 x 4

8 x 8

16 x 16

32 x 32

8 x 8

16 x 16

32 x 32

256 x 256
Adversarial testing of RL agents

\[ M, p_s, p_g \]

\[ r(M, p_s, p_g | A) \]

Powerful RL Agent
Adversarial testing of RL agents

\[ M, p_s, p_g \]

\[ r(M, p_s, p_g | A) \]

Bayesian Optimization

\[ \min_{M, p_s, p_g} r(M, p_s, p_g | A) \]

\[ (M, p_s, p_g, A) \sim p(\mathcal{T}) \] - training & holdout samples (agents, mazes, positions)

Yee Whye Teh
Bayesian Optimization performance

![Graph showing performance of different optimization methods over iterations. The x-axis represents iterations, and the y-axis represents the normalized minimum. The graph compares Alpha-Div, Random Search, BBB, DKL, Neural Process, and GP.]
Examples of performance decrease
Summary

- Prior knowledge/inductive biases are necessary for fast learning.
  - Knowledge about what is in the environment
  - Knowledge about how to solve tasks
- Sources of prior knowledge:
  - Features, losses, architectures
  - Data augmentation
  - Data from other modalities
  - Related tasks
Thank You!

Recommender systems

Objective:

For given user \( u \), approximate its rating function \( f_u : \mathcal{I} \rightarrow \mathbb{R} \) given the observed context \( \mathcal{I}_u \subset I \)
Recommender systems

Objective:

For given user $u$, approximate its rating function $f_u : \mathcal{I} \rightarrow \mathbb{R}$ given the observed context $\mathcal{I}_u \subset \mathcal{I}$

$$(\mathcal{I}_u, \mathcal{R}_u) \sim p(\mathcal{T})$$ - training & holdout samples

$\mathcal{IG}(\mathcal{I}_i) := \mathcal{H}(p(r_{\setminus i}|\mathcal{C})) - \mathbb{E}_{p(r_{\setminus i}|\mathcal{C})}[\mathcal{H}(p(r_{\setminus i}|\mathcal{C}'))] \quad \mathcal{C}' = \mathcal{C} \cup \{\mathcal{I}_i, \hat{r}_i\}$

Bootstrapping from the model’s predictions
## Results on MovieLens: RMSE

<table>
<thead>
<tr>
<th>Model</th>
<th>MovieLens 100k</th>
<th>20% of user data</th>
<th>50%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD++</td>
<td>1.0517</td>
<td>–</td>
<td>1.0217</td>
<td>–</td>
</tr>
<tr>
<td>Multitask MLP</td>
<td>0.9831</td>
<td>–</td>
<td>0.9679</td>
<td>–</td>
</tr>
<tr>
<td>MAML</td>
<td>0.9593</td>
<td>–</td>
<td>0.9441</td>
<td>–</td>
</tr>
<tr>
<td>NP (random)</td>
<td>0.9359</td>
<td>–</td>
<td>0.9215</td>
<td>–</td>
</tr>
<tr>
<td>NP (Info gain)</td>
<td>0.9288</td>
<td>–</td>
<td>0.8829</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>MovieLens 20m</th>
<th>20%</th>
<th>50%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD++</td>
<td>0.9454</td>
<td>–</td>
<td>0.9454</td>
<td>–</td>
</tr>
<tr>
<td>Multitask MLP</td>
<td>0.8570</td>
<td>–</td>
<td>0.8401</td>
<td>–</td>
</tr>
<tr>
<td>MAML</td>
<td>0.8142</td>
<td>–</td>
<td>0.7852</td>
<td>–</td>
</tr>
<tr>
<td>NP (random)</td>
<td>0.7982</td>
<td>–</td>
<td>0.7684</td>
<td>–</td>
</tr>
<tr>
<td>NP (Info gain)</td>
<td>0.7932</td>
<td>–</td>
<td>0.7366</td>
<td>–</td>
</tr>
</tbody>
</table>