Section 2: Transfer in Reinforcement Learning

Section Outline

• Introduction to RL

• The dimensions of transfer
  – task relatedness
  – transferred knowledge
  – learning algorithms

• Transfer between tasks with same state-action variables
  – From one source task to one target task
  – From many source tasks to one target task
  – Multitask learning: Learning a distribution of tasks

• Transfer between tasks with different state-action variables
  – No explicit mapping
  – Mapping state variables and actions between tasks
  – Learning the inter-task mapping

Introduction to RL

• See Part I of the tutorial
• Here we briefly recall basic concepts and notation
Introduction to RL

- Markov Decision Process

\[ \mathcal{M} = \langle S, A, R, P \rangle \]

\[ P(s_{t+1} \mid s_t, a_t, \ldots, s_0, a_0) = P(s_{t+1} \mid s_t, a_t) \]

Markov property

- (Deterministic) Policy \( \pi : S \rightarrow A \)

- Value functions

\[ V^\pi(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s \right] \]

\[ Q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s, a_0 = a \right] \]
Introduction to RL

• Optimal value functions

\[ V^*(s) = \max_{a \in A} \sum_{s'} P(s'|s, a) \left[ R(s, a) + \gamma V^*(s') \right] \]

\[ Q^*(s, a) = R(s, a) + \gamma V^*(s') \]

• Optimal policy

\[ \pi^*(s) = \arg \max_{a \in A} Q^*(s, a) \]

Introduction to RL

• On-line algorithms: learning as collecting samples

\[ Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left( R(s, a) + \gamma \max_{a' \in A} Q(s', a') \right) \]

Introduction to RL

• Batch algorithms (FQI)

\[ Q^0(\cdot, \cdot) = \arg \min_{Q \in F} \sum_{i=1}^{n} [Q(s_i, a_i) - R(s_i, a_i)]^2 \]

\[ Q^k(\cdot, \cdot) = \arg \min_{Q \in F} \sum_{i=1}^{n} [Q(s_i, a_i) - \left( R(s_i, a_i) + \gamma \max_{a' \in A} Q^{k-1}(s_{i+1}, a') \right)]^2 \]
Section Outline

- Introduction to RL
- The dimensions of transfer
  - task relatedness
  - transferred knowledge
  - learning algorithms
- Transfer between tasks with same state-action variables
  - From one source task to one target task
  - From many source tasks to one target task
  - Multitask learning: Learning a distribution of tasks
- Transfer between tasks with different state-action variables
  - No explicit mapping
  - Mapping state variables and actions between tasks
  - Learning the inter-task mapping

Task Differences

- Goal (reward function)
  \[ \mathcal{M}_1 = \langle S, A, R_1, P \rangle \quad \mathcal{M}_2 = \langle S, A, R_2, P \rangle \]
- Dynamics (transition model)
  \[ \mathcal{M}_1 = \langle S, A, R, P_1 \rangle \quad \mathcal{M}_2 = \langle S, A, R, P_2 \rangle \]
- Domain (state-action space / features)
  \[ \mathcal{M}_1 = \langle S_1, A_1, R, P \rangle \quad \mathcal{M}_2 = \langle S_2, A_2, R, P \rangle \]

Transferred Knowledge

- Task representation
  - Action space (e.g., options, task decomposition)
  - Reward function
  - Solution representation
    - Basis function

- Experience Transfer
  - Samples
    - Collected through direct exploration
  - Value function / policy
    - Solution initialization

Type of Learning Algorithm

- Online vs. Offline (batch)
  - Online: bias the learning/exploration process
  - Offline: bias the approximation of the value function
- Model based (model learning) vs. Model free
  - Model based: high-level common structure among the MDPs
  - Model free: low-level similarities among the MDPs
The Dimensions of Transfer

- 
  - The choice of the algorithm influences the knowledge that can be transferred.
  - The effectiveness of the transferred knowledge depends on the task differences/relatedness.

Transfer Metrics

- Domain Dependant:
  - Asymptotic performance
  - Jumpstart
  - Total reward
  - Learning time

- Domain Independent:
  - ?

Section Outline

- Introduction to RL
  - The dimensions of transfer:
    - task relatedness
    - transferred knowledge
    - learning algorithms
  - Transfer between tasks with same state-action variables:
    - From one source task to one target task
    - From many source tasks to one target task
    - Multitask learning: Learning a distribution of tasks
  - Transfer between tasks with different state-action variables:
    - No explicit mapping
    - Mapping state variables and actions between tasks
    - Learning the inter-task mapping
Section Outline

- Introduction to RL
- The dimensions of transfer
  - task relatedness
  - transferred knowledge
  - learning algorithms
- Transfer between tasks with the same state-action variables
  - From one source task to one target task
  - From many source tasks to one target task
  - Multitask learning: Learning a distribution of tasks
- Transfer between tasks with different state-action variables
  - No explicit mapping
  - Mapping state variables and actions between tasks
  - Learning the inter-task mapping

1-to-1: the Scenario

- One source task
  - Collect some knowledge (e.g., samples, solution, abstraction, …)
- One target task
  - Very few information is available
- Assumption: same state-action space
1-to-1: Example

\[ \mathcal{M}_2 \]

1-to-1: Example

\[ \mathcal{M}_2 \]

1-to-1: Example

\[ \mathcal{M}_2 \]

1-to-1: Formalization

- MDPs  \( \mathcal{M}_1 = (S, A, R_1, P_1) \)  \( \mathcal{M}_2 = (S, A, R_2, P_2) \)
- Knowledge  \( \mathcal{K}_\mathcal{M} \)  (e.g.,  \( \mathcal{K}_\mathcal{M} = \{ (s_i, a_i, r_i, s'_i) \} \leq n \))

\[ A(\mathcal{K}_\mathcal{M}) = \begin{cases} \mathcal{V}/\mathcal{Q} \\ \pi \end{cases} \]
- Learning Algorithm

\[ T(\mathcal{K}_\mathcal{M}) = \mathcal{K}'_\mathcal{M} \]
- Transfer function
1-to-1: Formalization

- Transfer process
  1. Collect $\mathcal{K}_{\mathcal{M}_1}$ from the source task
  2. Collect $\mathcal{K}_{\mathcal{M}_2}$ from the target task
  3. Transfer $\mathcal{I}(\mathcal{K}_{\mathcal{M}_1} | \mathcal{K}_{\mathcal{M}_2}) = \mathcal{K}_{\mathcal{M}_2}'$
  4. Learn $\mathcal{A}(\mathcal{K}_{\mathcal{M}_2} \cup \mathcal{K}_{\mathcal{M}_2}')$
  5. Evaluate the performance

Points 2. 3. 4. can be reiterated

1-to-1: Challenges

- Which knowledge to transfer?
  - The choice depends on the task relatedness (e.g., similar optimal policy, similar optimal value function, etc.) and on the learning algorithm (e.g., batch algorithms cannot be initialized)

- How to transfer the knowledge?
  - Direct transfer: use source knowledge in the target task as it is (e.g., Q-table initialization)
  - Transformation of source knowledge according to target structure

1-to-1: A Representative Algorithm (1)

- “Proto-Transfer Learning in Markov Decision Processes Using Spectral Methods” (Mahadevan, Ferguson, 2006)
- The idea: extract basis functions from the source task and reuse them in tasks with similar “graph”
- Task difference: goal and dynamics (and domain)
- Transferred knowledge: solution representation
- Learning algorithm: model-free batch
- Metric: generalization
1-to-1: A Representative Algorithm (1)

- **Pros**
  - Proto-value functions can be reused in many different tasks independently from how similar the optimal value functions are

- **Cons**
  - The “shape” of the optimal value function depends also on the reward function (see (Ferrante et al., 2008))

### Table

<table>
<thead>
<tr>
<th></th>
<th>Exp 1a (pure)</th>
<th>Exp 1b (pure)</th>
<th>Exp 1b (transfer)</th>
<th>Exp 1c (pure)</th>
<th>Exp 1c (transfer)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prob. of success</strong></td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Avg. # of steps</strong></td>
<td>14.8 ± 2.1</td>
<td>13.6 ± 2.1</td>
<td>14.9 ± 3.0</td>
<td>7.3 ± 1.2</td>
<td>7.4 ± 1.2</td>
</tr>
<tr>
<td><strong>Min/Max steps</strong></td>
<td>[5, 27]</td>
<td>[4, 22]</td>
<td>[5, 24]</td>
<td>[3, 13]</td>
<td>[2, 11]</td>
</tr>
<tr>
<td><strong>Avg. total discounted rew.</strong></td>
<td>26.2 ± 5.6</td>
<td>30.0 ± 7.1</td>
<td>29.2 ± 8.8</td>
<td>53.5 ± 6.5</td>
<td>53.1 ± 7.3</td>
</tr>
<tr>
<td><strong>Iterations to convergence</strong></td>
<td>19</td>
<td>16</td>
<td>11</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>
1-to-1: A Representative Algorithm (2)

- “Metrics for finite Markov decision processes” (Ferns et al., 2005)
- The idea: define a metric on the MDPs that can be used to bound the transfer performance
- Task difference: goal and dynamics
- Transferred knowledge: (optimal) policy
- Learning algorithm: model-based
- Metric: learning time (in terms of computational cost)

1-to-1: A Representative Algorithm (2)

- Assumption: both models are available but they are computationally expensive to solve
- Compute a (nearly-optimal) policy on the source task and reuse it in the target task
- How far is the transfer performance from the optimal one given the (low-level) difference between the two MDPs?

1-to-1: A Representative Algorithm (2)

- MDP distance
  \[ d(s) = \max_{a \in A} \left( |R_1(s, a) - R_2(s, a)| + cT_K(d)(P_1(s, a), P_2(s, a)) \right) \]

  Distance in state s

  Kantorovich distance

- Transfer performance
  \[ \|V_{2\pi_1} - V_{2}\| \leq \frac{2}{1-c} \max_{a \in A} d(s) + \frac{1+c}{1-c} \|V_{1\pi_1} - V_{1}\| \]

  Performance of \( \pi_1 \) in \( M_2 \)

1-to-1: A Representative Algorithm (2)

- Pros
  - Given the model difference provides a bound over the transfer performance
- Cons
  - It is not a transfer algorithm (direct transfer of the policy)
  - The MDP metric can be computationally expensive
1-to-1: A Representative Algorithm (3)

• “Improving Action Selection in MDP’s via Knowledge Transfer” (Sherstov and Stone, 2005)

• The Idea: in problems with large/infinite number of actions, only few are really necessary (e.g., the Baker Task), then transfer of the action set from source to target

• Task differences: goal and dynamics

• Learning algorithm: model-free, online (any?)

• Metric: learning time

1-to-1: A Representative Algorithm (3)

• Few actions are really useful to solve the problem

1-to-1: A Representative Algorithm (3)

• The source task could be not representative enough

• Random Task Perturbation (RTP)
  – Generates series of source tasks
  – Guard against misleading source tasks

• Extended by Leffler et al. (2007) to speed up single task learning

1-to-1: A Representative Algorithm (3)

• Optimal policies in the perturbed sources
  \( K_{M_1} = \{ \pi_i^* \} \)

• Extract an optimal action space
  \( T(K_{M_1}) = A' \)
1-to-1: A Representative Algorithm (3)

• Pros
  – Bias the learning towards “useful” actions
  – Can be used with any learning algorithm

• Cons
  – Removing actions could prevent from learning the optimal policy (but the loss could be bounded)

1-to-1: Conclusion

• Most straightforward type of transfer
• The transfer mechanism is strictly related with the learning algorithm

Open Problems
  – How task similarity influences the performance of transfer
  – Proof of transfer advantage over learning from scratch
  – Connections with domain adaptation in (semi-)supervised learning

Section Outline

• Introduction to RL

• The dimensions of transfer
  – task relatedness
  – transferred knowledge
  – learning algorithms

• Transfer between tasks with same state-action variables
  – From one source task to one target task
  – From many source tasks to one target task
  – Multitask learning: Learning a distribution of tasks

• Transfer between tasks with different state-action variables
  – No explicit mapping
  – Mapping state variables and actions between tasks
  – Learning the inter-task mapping

N-to-1: the Scenario

• Set of source tasks
  – Collect knowledge from each of them

• One target task
• Selectively transfer source knowledge to the target task

• Assumption: same state-action space
N-to-1: Example

N-to-1: Challenges

- Merge different sources of knowledge
- Select sources similar to the target task
- Avoid negative transfer

N-to-1: Formalization

- Source MDPs: $\mathcal{M}_i = (S, A, R_i, P_i), \ 1 \leq i \leq N$
- Target MDP: $\mathcal{M}_t = (S, A_t, R_t, P_t)$
- Selection function: $\mathcal{F}(\{\mathcal{K}_{\mathcal{M}_i}\}) = \{\mathcal{K}'_{\mathcal{M}_i}\}$
- Transfer function: $\mathcal{T}(\mathcal{K}'_{\mathcal{M}_i}) = \mathcal{K}_{\mathcal{M}_t}$
- Learning algorithm:

$$\mathcal{A}\left(\bigcup_{i=1}^{N} \mathcal{K}'_{\mathcal{M}_i} \cup \mathcal{K}_{\mathcal{M}_t}\right)$$

N-to-1: Formalization

- Transfer process
  1. Collect $\mathcal{K}_{\mathcal{M}_i}, \ 1 \leq i \leq N$
  2. Collect $\mathcal{K}_{\mathcal{M}_t}$
  3. Select sources and knowledge $\mathcal{F}(\{\mathcal{K}_{\mathcal{M}_i}\}) = \{\mathcal{K}'_{\mathcal{M}_i}\}$
  4. Transfer $\mathcal{T}(\mathcal{K}'_{\mathcal{M}_i}) = \mathcal{K}'_{\mathcal{M}_t}$
  5. Learn $\mathcal{A}\left(\bigcup_{i=1}^{N} \mathcal{K}'_{\mathcal{M}_i} \cup \mathcal{K}_{\mathcal{M}_t}\right)$

The process can be reiterated
N-to-1: A Representative Algorithm

- "Transfer of samples in batch reinforcement learning" (Lazaric et al., 2008)
- The idea: selectively reuse samples on the basis of their likelihood in the target task
- Task difference: goal and dynamics
- Transferred knowledge: samples
- Learning algorithm: model-free batch
- Metric: learning time

N-to-1: A Representative Algorithm

- Knowledge $\mathcal{K} = \{s_j, a_j, r_j, s'_j\}$
- Collect $\mathcal{K}_M$, $1 \leq i \leq N$
- Collect $\mathcal{K}_s$
- Compute compliance/relevance for each source
- Select knowledge $\mathcal{F}(\mathcal{K}_M) = \mathcal{K}'_M$
- Transfer samples as they are $\mathcal{K}'_s = \mathcal{K}'_M$

- Run $\mathcal{A}\left(\bigcup_{i=1}^{N} \mathcal{K}'_M \cup \mathcal{K}'_s\right)$

N-to-1: A Representative Algorithm

- Source tasks selection
- Likelihood of target samples to be generated by the source tasks (compliance)

$$\lambda_j = P(M_i | \tau_j) \propto P(\tau_j | M_i) P(M_i) = P_{M_i}(a_j' | s_j, a_j) R_{M_i}(r_j | s_j, a_j) P(M_i)$$

where $\tau_j = \langle s_j, a_j, a_j', r_j \rangle \in \mathcal{K}_M$

$$\lambda_{M_i | \mathcal{K}_M} = \frac{1}{|\mathcal{K}_M|} \sum_{j=1}^{K_{M_i}} \lambda_j P(M_i)$$
N-to-1: A Representative Algorithm

- **Compliance**: task similarity in terms of likelihood of target samples to be generated by source tasks

\[
\lambda_{M_t} = \frac{1}{|\mathcal{K}_{M_s}|} \sum_{j=1}^{\mathcal{K}_{M_s}} \lambda_j P(M_i)
\]

- The higher the compliance (probability of target samples to be generated by the source task), the higher the probability to be transferred

N-to-1: A Representative Algorithm

- Source samples selection
- Among source samples select those which are more similar/informative to the target task

N-to-1: A Representative Algorithm

- Pros
  - Effective method to select sources and samples
  - Avoid negative transfer
- Cons
  - Difficult to relate the difference between the samples and the difference between the solutions
  - Tasks may have different models but similar solutions
N-to-1: Conclusions

• The selection of source tasks is critical
• Not all the types of knowledge can be easily merged among different tasks
• Open problems
  – Towards an open-ended transfer process
  – Tasks with different state-action space
  – Transfer from very different tasks may result in positive transfer

Section Outline

• Introduction to RL
• The dimensions of transfer
  – task relatedness
  – transferred knowledge
  – learning algorithms
• Transfer between tasks with same state-action variables
  – From one source task to one target task
  – From many source tasks to one target task
  – Multitask learning: Learning a distribution of tasks
• Transfer between tasks with different state-action variables
  – No explicit mapping
  – Mapping state variables and actions between tasks
  – Learning the inter-task mapping

MTL: the Scenario

• A set of tasks is given (e.g., drawn from a fixed distribution)
• Compute a solution for each of them trying to exploit their similarity

MTL: Example
MTL: Challenges

- Definition of similarity/relatedness
  - Similar solutions (e.g., weights of the linear function approximator)
  - Similar structure (e.g., similar reward functions)
  - Common generative model
- Definition of an algorithm able to exploit the relatedness (e.g., if the tasks are G-related then the algorithm is able to improve the performance)

MTL: Formalization

- MDPs: $\mathcal{M}_i = (S, A, R_i, P_i), \ 1 \leq i \leq N$
- Similarity function (the definition is highly dependent on the algorithm): $\mathcal{G}(\{\mathcal{M}_i\})$
- Joint learning algorithm: $\mathcal{A}(\{\mathcal{K}_i\}|\mathcal{G})$

MTL: Formalization

- Transfer process
  1. Collect $\mathcal{K}_{\mathcal{M}_i}, \ 1 \leq i \leq N$
  2. Compute similarity $\mathcal{G}(\{\mathcal{M}_i\})$ using $\{\mathcal{K}_i\}$
  3. Learn $\mathcal{A}(\{\mathcal{K}_i\}|\mathcal{G})$

The process can be reiterated

MTL: A Representative Algorithm (1)

- “Multi-Task Reinforcement Learning: A Hierarchical Bayesian Approach” (Wilson et al., 2007)
- The idea: tasks belong to different classes drawn from a fixed distribution
- Task difference: goal and dynamics
- Transferred knowledge: task structure
- Learning algorithm: model-based batch
- Metric: learning time
MTL: A Representative Algorithm (1)

• Similarity function $G$
  – Hierarchical generative model
  – Define a prior over the distribution of the (parameters of the) tasks

• Algorithm
  – Use all the samples to refine $G$
  – Use task-specific samples to learn the model

MTL: A Representative Algorithm (1)

• Given a suitable parameterization of the MDPs
• Given the hierarchical model parameters
• Collect enough samples from each
• Compute the parameters and the MDP with an EM-like algorithm
  – E-step $\tilde{\mathcal{M}}_i \leftarrow \text{SampleMAP}(Pr(\mathcal{M}|\mathcal{K}_i, \psi))$
  – M-step $\psi \leftarrow \text{SampleMAP}(Pr(\psi|\tilde{\mathcal{M}}_1, \ldots, \tilde{\mathcal{M}}_N))$
MTL: A Representative Algorithm (1)

- **Pros**
  - Once the hyper-parameters are tuned, it can be used also in the N-to-1 scenario
  - Tasks can belong to different classes

- **Cons**
  - The complexity of the generative model requires many samples to estimate the hyper-parameters
  - Focus on the MDPs but does not relate their solutions

MTL: A Representative Algorithm (2)

- "Knowledge transfer in Reinforcement Learning" (Lazaric, 2008)
- The idea: tasks share the same underlying feature space
- Task difference: goal and dynamics
- Transferred knowledge: solution representation
- Learning algorithm: model-free batch
- Metric: generalization

MTL: A Representative Algorithm (2)

- Multi-task feature learning (Argyriou, 2008)
  \[ \varepsilon(W, U) = \sum_{t=1}^{T} \sum_{k=1}^{m_k} \text{loss}(z_k, \langle w_k, U^T \varphi(z_k) \rangle) + \lambda ||W||^2_{2,1} \]
- Learn features and weights such that each task share the same feature space
- Integration into a FQI algorithm at each iteration
MTL: A Representative Algorithm (2)

Colored Grid World Problem

MTL: A Representative Algorithm (2)

Boat Problem

MTL: A Representative Algorithm (2)

Colored Grid World Problem

MTL: A Representative Algorithm (2)

Boat Problem

MTL: A Representative Algorithm (2)

• Pros
  – Automatically change the feature space in order to take advantage the most the task similarity
  – Improve the generalization capabilities

• Cons
  – The feature space may be different at each iteration

MTL: Conclusions

• Many possible models of relatedness
• Most common perspective in supervised learning
• Open problems
  – Difference between similarity of models and of solutions
  – Find and exploit relationships with supervised learning literature
  – Definition of algorithms provably able to exploit task relatedness and to avoid negative transfer