Transfer Learning for Reinforcement Learning Domains: A Survey

Presentation by Takashi Nagata and J.B. Lanier

Paper by Matthew E. Taylor and Peter Stone

Journal of Machine Learning Research, volume 10, pp 1633–1685, 2009.

Reinforcement Learning is hard,

especially from a blank slate.

- Need ways to efficiently abstract over the state space so that the agent can generalize experience.
- Things we might try:

  - Use function approximation to abstract over the state space.

  - Deconstruct tasks into hierarchies of subtasks.

  - Learn several low level policies (i.e. "options") and

    learn a meta policy to switch between them.

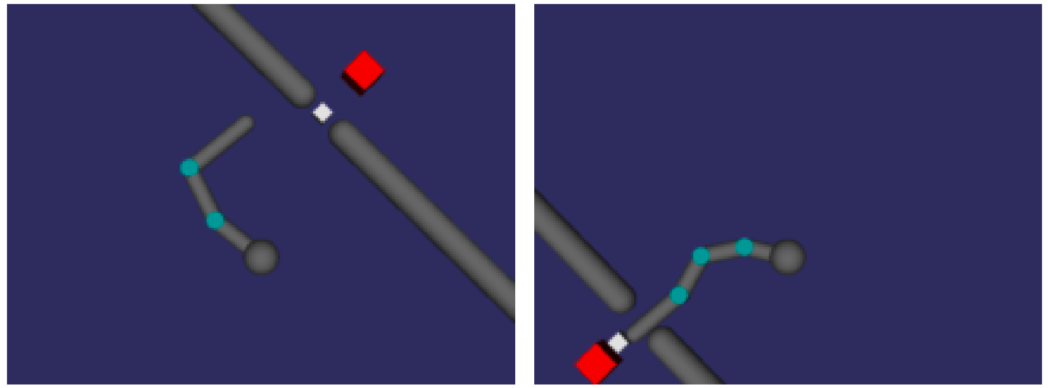
Transfer Learning

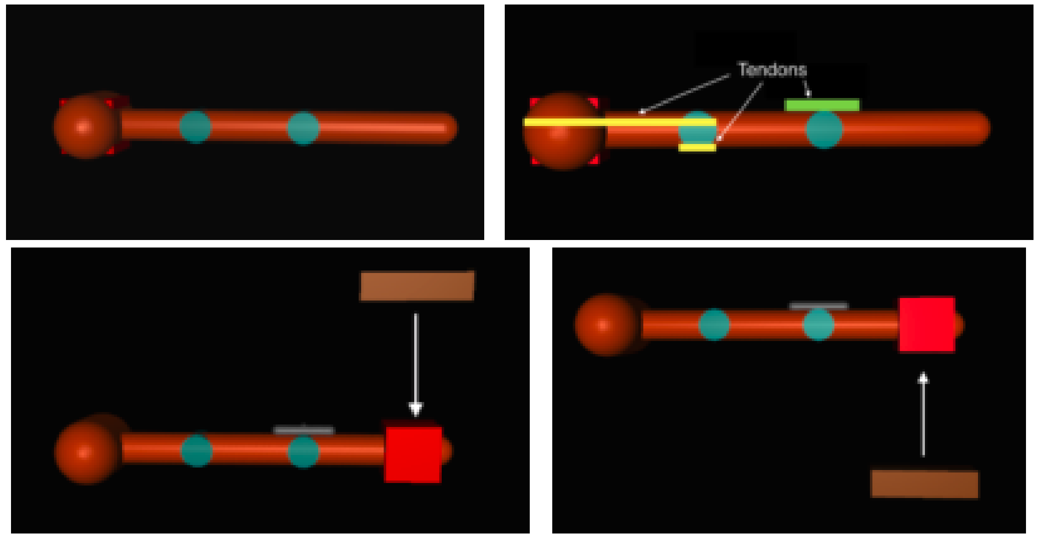
- Generalize *across* tasks.
- Use information learned in a ***source task*** to accelerate/improve learning in a ***target task***.

Intuitive example (*not from this paper)

- The 3 and 4 link robots performing the button pressing task

- Torque controlled and tendon controlled robot manipulation





A.Gupta et al. 2017

Evaluating Transfer Learning Methods

Agent would have to perform all of the following steps

1. Given a target task, select an appropriate source task or set of tasks from which to transfer
2. Learn how the source task(s) and target task are related
3. Effectively transfer knowledge from the source task(s) to the target task.

*Ideally… however

– TL research has focused on each independently

– no TL methods are currently capable of robustly accomplishing all three goals

Evaluating Transfer Learning Methods

1. Given a target task, select an appropriate source task or set of tasks from which to transfer
2. Learn how the source task(s) and target task are related
3. Effectively transfer knowledge from the source task(s) to the target task.

A challenge in TL research: <u>define evaluation metrics</u>

- there are many possible measurement options
- algorithms may focus on any of the three steps above

5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

Conclusion first…

- Jumpstart
- Asym
- Total
- Trans
- Time

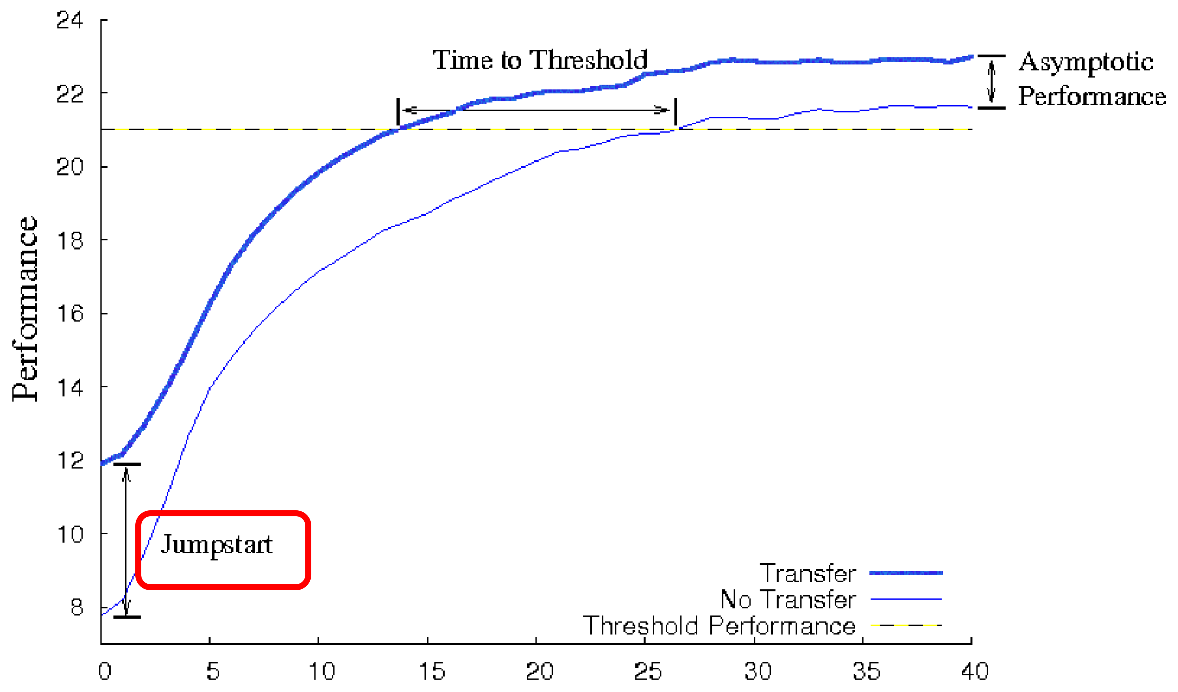
- Each metric has drawbacks and none are sufficient to fully describe the benefits of any transfer method
- It's impossible to create a total order ranking of different methods
- The authors instead suggest that a multi-dimensional evaluation with multiple metrics

5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

Metric1: Jumpstart

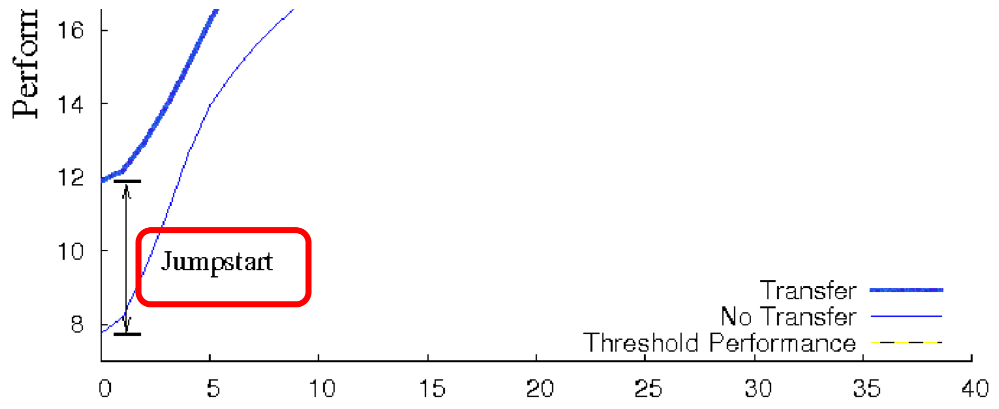
- The initial performance of an agent in a target task
  - “can transfer be used so that the initial performance is increased relative to the performance of an initial (random) policy?”



Metric1: Jumpstart

- The initial performance of an agent in a target task
    - "can transfer be used so that the initial performance is increased relative to the performance of an initial (random) policy?"

- fails to capture the behavior of learning in the target task
    - only focuses on the performance before learning occurs

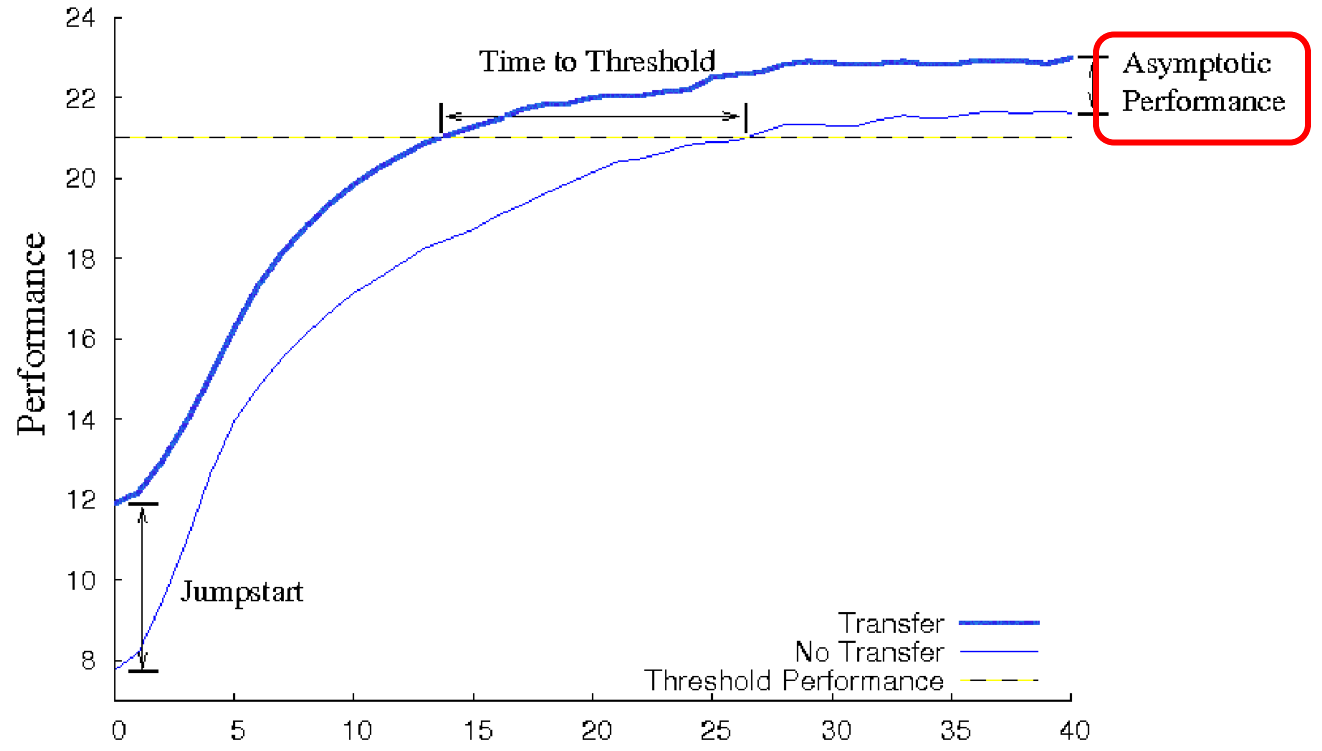


5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

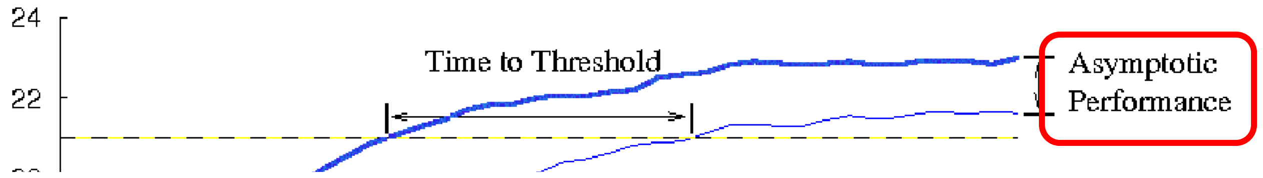
Metric2: Asymptotic Performance

- The final learned performance of an agent in the target task both with and without transfer

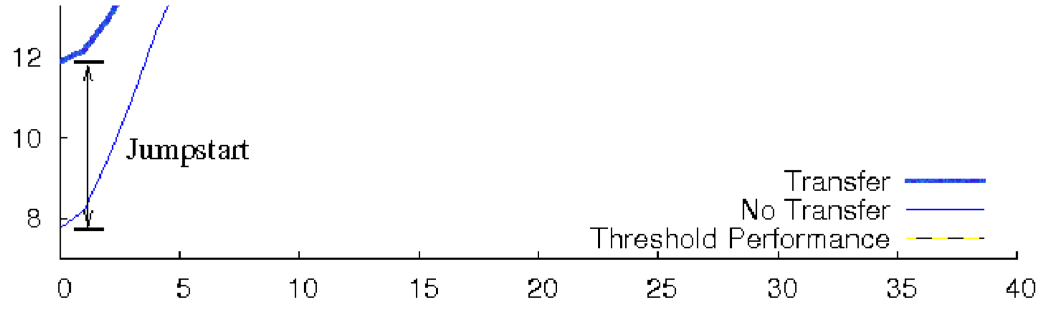


Metric2: Asymptotic Performance

- The final learned performance of an agent in the target task both with and without transfer



- It may be difficult to tell when the learner has indeed converged
    - (particularly in tasks with infinite state spaces) convergence may take prohibitively long



5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

Metric3: Total reward

- the total reward accumulated during training

  => Improving initial performance and achieving a faster learning rate will help agents accumulate more on-line reward

Metric3: Total reward

- It's possible that
  - a method which achieves a high performance relatively quickly will have less total reward by converging to sub-optimal level

    on the other hand

  - a method which learns very slowly but eventually plateaus at a slightly higher performance level will get more reward

"Quick and less total reward vs. Slow and higher total reward"

This may happen
=> The metric is most appropriate for tasks that have a <u>well-defined duration</u>

5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

Metric4: Transfer Ratio

- The ratio of
  the total reward accumulated by the transfer learner and
  the total reward accumulated by the non−transfer learner

$$r = \frac{\textit{area under curve with transfer - area under curve without transfer}}{\textit{area under curve without transfer}}$$

Metric4: Transfer Ratio

- The ratio c
  the total r
  the total r

$$r = \frac{area\ ur \quad out\ transfer}{}$$

Not scale invariant:

e.g.
the area under the transfer curve were 1000 units
the area under the non-transfer curve were 500
=> the transfer ratio would be 1.0

If all rewards were multiplied by a constant, this ratio would not change.
But if an offset were added, the ratio would change.

$$(1010 - 510) / 510 = 0.98$$

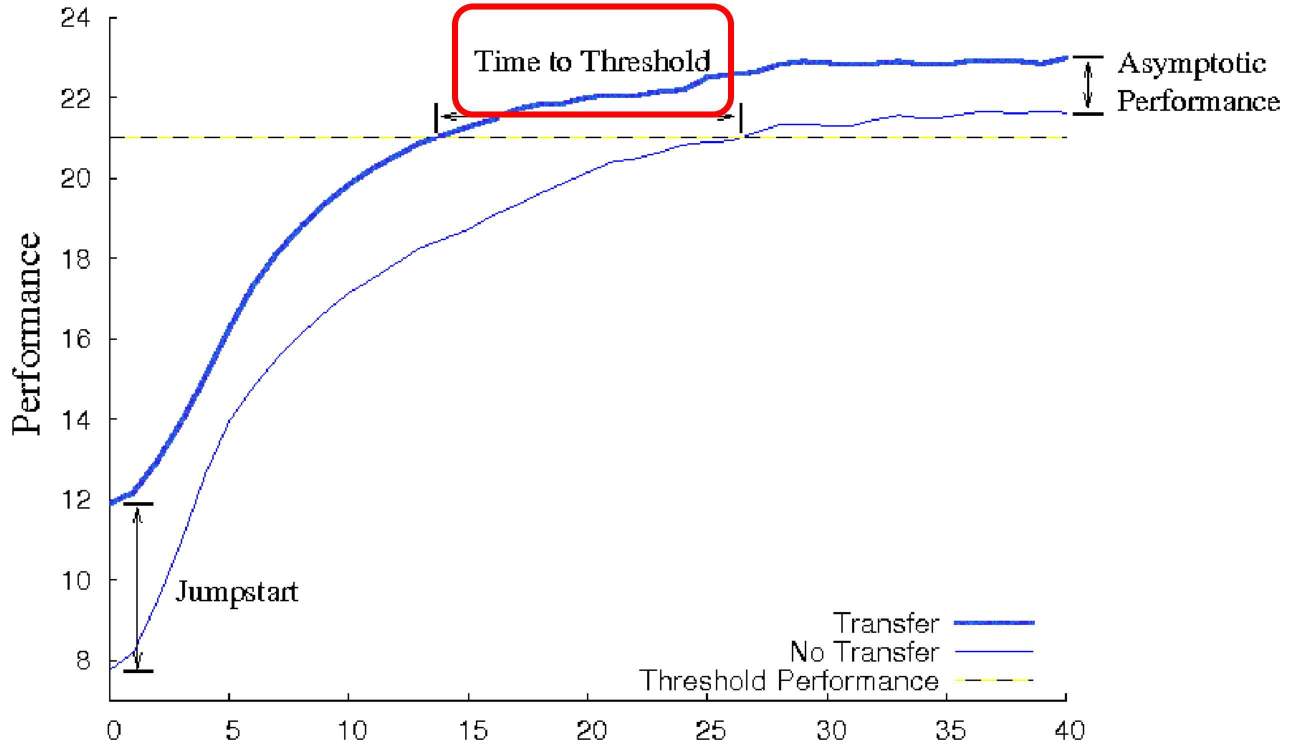
The evaluation of a TL algorithm with the transfer ratio is therefore <u>closely related to the reward structure of the target task being tested</u>

5 metrics

- Jumpstart
- Asymptotic Performance
- Total Reward
- Transfer Ratio
- Time to Threshold

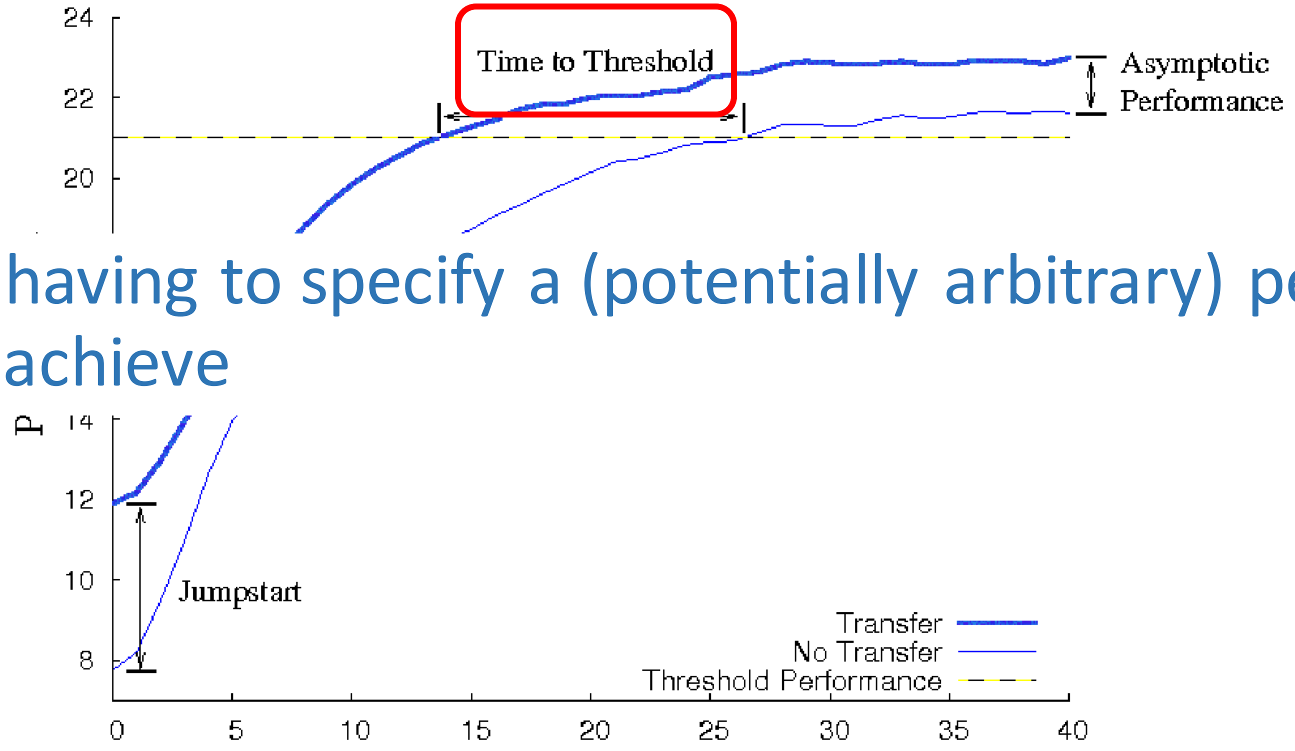
Metric5: Time to Threshold

- The learning time needed by the agent to achieve a pre-specified performance level



Metric5: Time to Threshold

- The learning time needed by the agent to achieve a pre-specified performance level



- Suffers from having to specify a (potentially arbitrary) performance agents must achieve

5 metrics

- Jumpstart
- Asym
- Total
- Trans
- Time

- Each metric has drawbacks and none are sufficient to fully describe the benefits of any transfer method
- It's impossible to create a total order ranking of different methods
- The authors instead suggest that a multi-dimensional evaluation with multiple metrics

Dimensions of Comparison

- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

Dimensions of Comparison

- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

Task Difference Assumptions

- How are the source and target tasks allowed to differ?
- Different actions, transitions, state spaces?
- Methods that allow less similar source and target tasks allow more flexibility, and are often more able to apply past knowledge to novel target tasks.

Dimensions of Comparison

- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

Source Task Selection

- How was the source task selected, by a human or as part of the algorithm?

- Can we learn from more than one source task?

- Hard to guarantee that a source task is actually useful.

Dimensions of Comparison

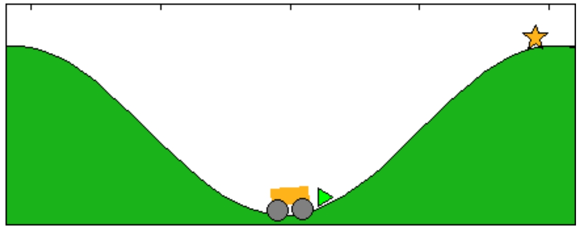
- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

Inter-task Mappings

- If tasks have different states and actions, a mapping from the source task to the target task will either need to be learned or provided in order to leverage past experience.

Inter-task Mappings

- If tasks have different states and actions, a mapping from the source task to the target task will either need to be learned or provided in order to leverage past experience.



{forward, neutral, back} -> {left, brake, right}

Inter–task Mappings

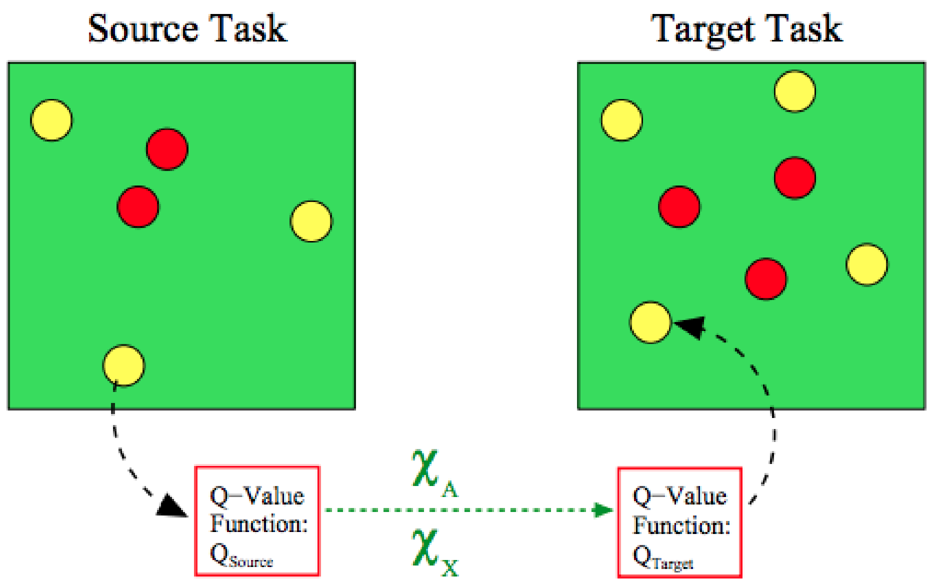

Source Task
Target Task
Q–Value Function: Q_{Source}
Q–Value Function: Q_{Target}
χ_A
χ_X

Figure 7: In Value Function Transfer, learned Q-values from the source task are transformed via inter-task mappings. These transformed Q-values are then used to initialize agents in the target task, biasing their learning with low-level knowledge from the source task.

χ_A: 4 vs. 3 Keepaway to 3 vs. 2 Keepaway

4 vs. 3 Action	3 vs. 2 Action
Hold Ball	Hold Ball
Pass_1	Pass_1
Pass_2	Pass_2
Pass_3	Pass_2

Table 3: This table describes the (full) action mapping from 4 vs. 3 to 3 vs. 2.

Image source: Taylor (2007)

Dimensions of Comparison

- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

Transferred Knowledge

- What information do we rememeber from source tasks?

- Low level information: {s,a,r,s'} instances, action–value functions, policies, models, prior distributions, etc.
  - Can be directly leveraged

- High level information: partial policies or options, important features, subtask definitions
  - Usually indirectly leveraged

Dimensions of Comparison

- Task Difference Asumptions
- Source Task Selection
- Inter-task Mappings
- Transferred Knowledge
- Allowed Learners

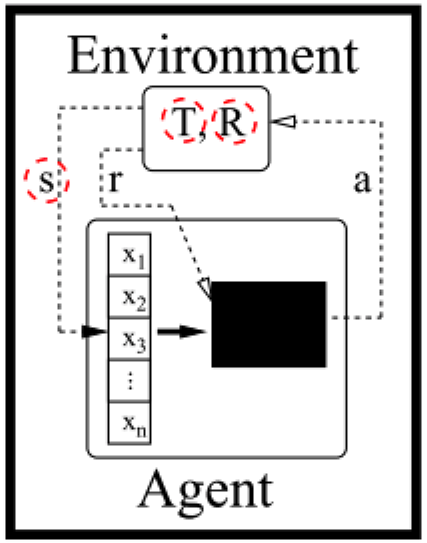
Allowed Learners

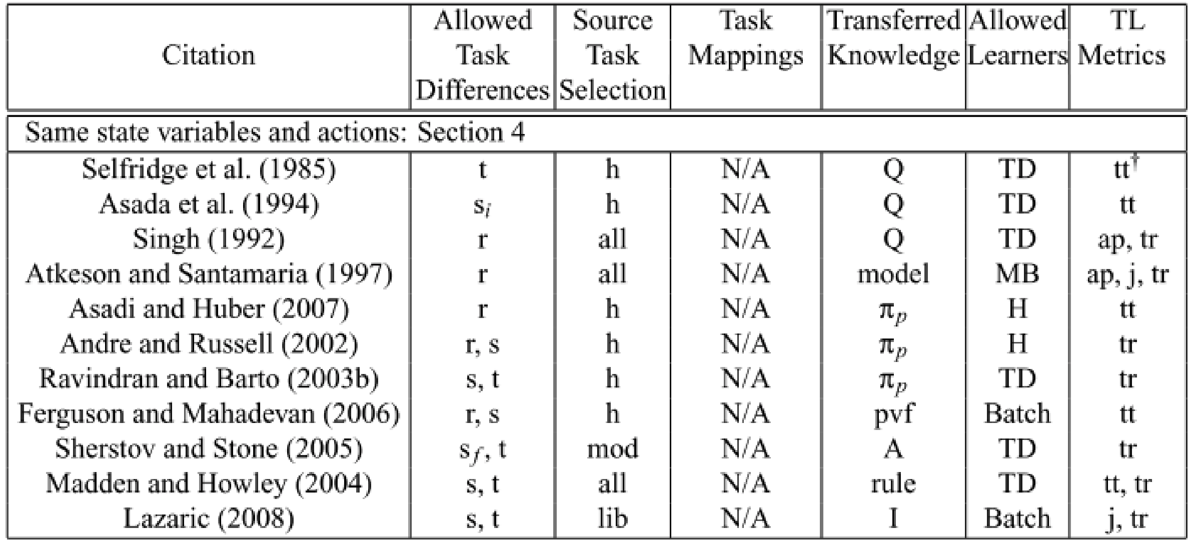
- What types of RL learning algorithms are actually compatible with the transferred knowledge?
- Some need the type of learner to stay the same across source and target task, others have no restrictions.
- Most common example in this paper is temporal difference learning.

Transfer Methods for

Fixed State Variables and Actions

- the source and target tasks use the same state variables and
- agents in both tasks have the same set of actions





Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Same state variables and actions: Section 4						
Selfridge et al. (1985)	t	h	N/A	Q	TD	tt[†]
Asada et al. (1994)	s_i	h	N/A	Q	TD	tt
Singh (1992)	r	all	N/A	Q	TD	ap, tr
Atkeson and Santamaria (1997)	r	all	N/A	model	MB	ap, j, tr
Asadi and Huber (2007)	r	h	N/A	π_p	H	tt
Andre and Russell (2002)	r, s	h	N/A	π_p	H	tr
Ravindran and Barto (2003b)	s, t	h	N/A	π_p	TD	tr
Ferguson and Mahadevan (2006)	r, s	h	N/A	pvf	Batch	tt
Sherstov and Stone (2005)	s_f, t	mod	N/A	A	TD	tr
Madden and Howley (2004)	s, t	all	N/A	rule	TD	tt, tr
Lazaric (2008)	s, t	lib	N/A	I	Batch	j, tr

Transfer Methods for <u>Fixed State Variables and Actions</u> contd.

- Some approaches
    - Changing a task over time
        - Selfridge et al. (1985)
    - Breaking a task into a series of smaller tasks
        - Singh (1992)
    - Transferring between a progression of tasks of increasing difficulty
        - Asada et al. (1994)
    - Using "partial policies" or 'options' between different tasks
        - Asadi and Huber (2007)
    - Transfer *instances* (*observed <s, a, r, s'> tuples)
        - Lazaric (2008)

Transfer Methods for

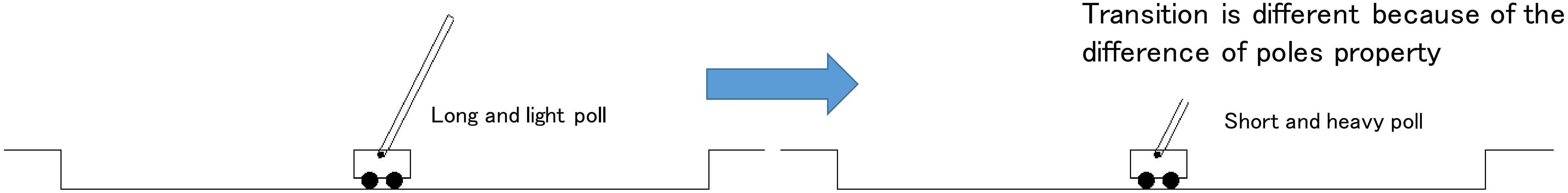
Fixed State Variables and Actions

- the source and target tasks use the same state variables
  and
- agents in both tasks have the same set of actions

State and actions are the same:

- Cart velocity
- Pole angle
- Left or Right

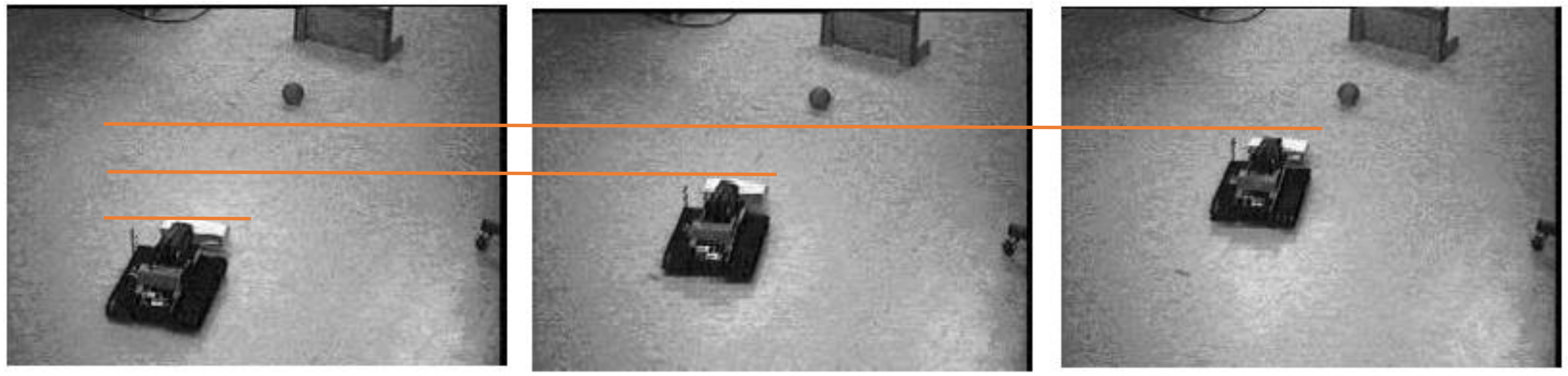
Transition is different because of the difference of poles property



Example from Selfridge et al. (1985)
*Figure is not from the paper

Asada et al., (1994)

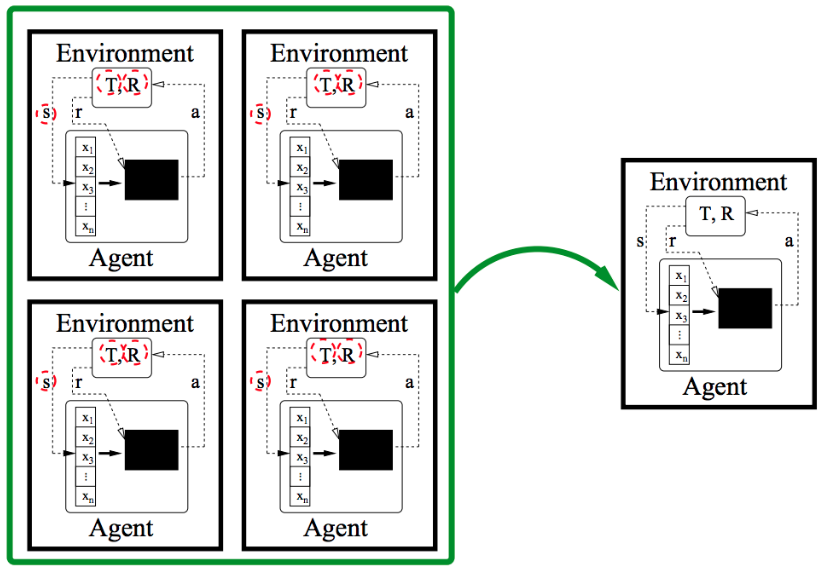
- learning from easy missions
- the task is made incrementally harder
    - by moving the agent's initial state further and further from the goal state
    - (not by changing the dynamics of the task)

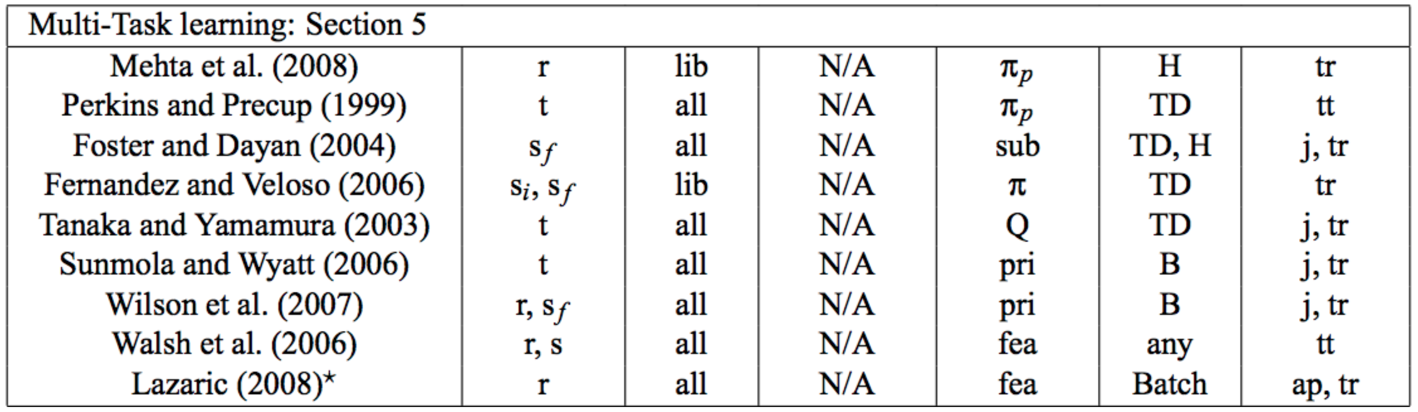


Transfer Methods for

Multi-Task Learning

- The source and target tasks have the same states and actions as before.
- But now we learn from multiple source tasks.





Multi-Task learning: Section 5						
Mehta et al. (2008)	r	lib	N/A	π_p	H	tr
Perkins and Precup (1999)	t	all	N/A	π_p	TD	tt
Foster and Dayan (2004)	s_f	all	N/A	sub	TD, H	j, tr
Fernandez and Veloso (2006)	s_i, s_f	lib	N/A	π	TD	tr
Tanaka and Yamamura (2003)	t	all	N/A	Q	TD	j, tr
Sunmola and Wyatt (2006)	t	all	N/A	pri	B	j, tr
Wilson et al. (2007)	r, s_f	all	N/A	pri	B	j, tr
Walsh et al. (2006)	r, s	all	N/A	fea	any	tt
Lazaric (2008)*	r	all	N/A	fea	Batch	ap, tr

Transfer Methods for

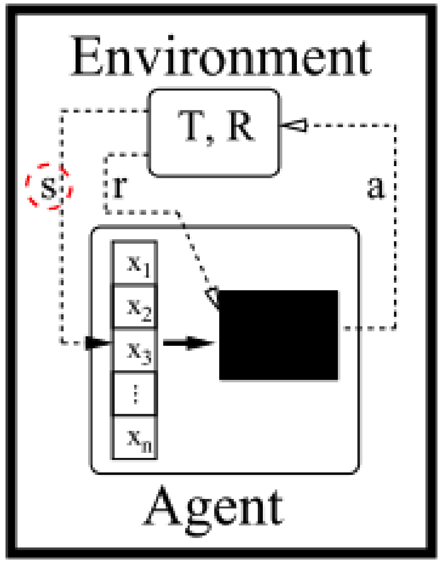
<u>Multi-Task Learning</u> contd.

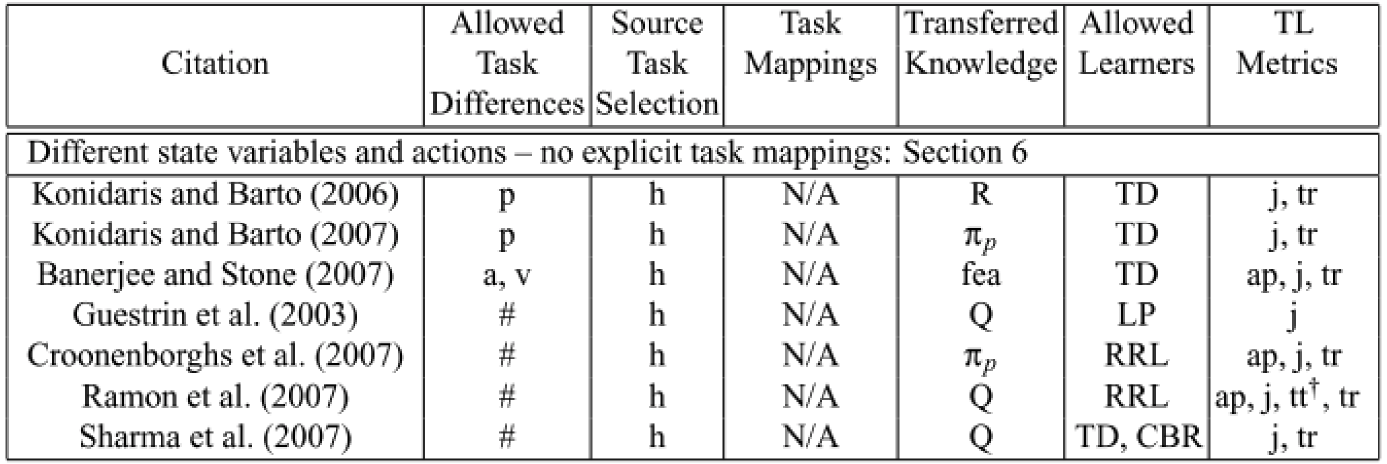
- Some approaches
  - Learn weighted combinations of reward features, match policies that weight reward features similarly.
    - Mehta et al. (2008)
  - Fragment source tasks into sub-tasks, use learned sub-tasks to solve target.
    - Foster and Dayan (2004)
  - At each step, randomly explore, or exploit previously learned policies (with higher chance to choose policies currently performing better).
    - Fernandez and Veloso (2006)
  - Use average Q-values from previous tasks as starting point for target task. Let standard deviation control priorities in exploration.
    - Tanaka and Tamamura (2003)

Transferring Task–Invariant Knowledge Between Tasks

with Differing State Variables and Actions

- allow the source task and target task to have different state variables and actions
- no explicit mapping between the tasks is needed





Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Different state variables and actions – no explicit task mappings: Section 6						
Konidaris and Barto (2006)	p	h	N/A	R	TD	j, tr
Konidaris and Barto (2007)	p	h	N/A	π_p	TD	j, tr
Banerjee and Stone (2007)	a, v	h	N/A	fea	TD	ap, j, tr
Guestrin et al. (2003)	#	h	N/A	Q	LP	j
Croonenborghs et al. (2007)	#	h	N/A	π_p	RRL	ap, j, tr
Ramon et al. (2007)	#	h	N/A	Q	RRL	ap, j, tt†, tr
Sharma et al. (2007)	#	h	N/A	Q	TD, CBR	j, tr

Transferring Task-Invariant Knowledge Between Tasks with Differing State Variables and Actions

- Some approaches
  - agent-space and problem-space representations
    - Task-specific item (features and actions) may change, but learned something from unchanging agent-space is useful
    - Konidaris and Barto (2006)
  - a relational MDP
    - Rather than learning a standard value function, an agent-centered value function for each class of agents is calculated in a source task
    - Guestrin et al. (2003)

idea of the method

- knowledge can be transferred between tasks with relative ease by framing the task in an agent−centric space
- problem representations do not change from the learner’s perspective

Konidaris and Barto (2006)

- separated the standard RL problem into two spaces representation
  - agent-space
    - determined by the agent's capabilities, which remain fixed
    - e.g. physical sensors and actuators
  - problem-space
    - may change between source and target problems

Konidaris and Barto (2006) cont.

- learns a shaping reward on-line while learning a source task in agent-space
- If a target task has <u>a similar reward structure</u> and action set, the learned shaping reward will help the agent achieve a jumpstart and higher total reward

- Example

  one of the agent' s sensors measures the distance between it and a particular important state (beacon located near the goal state)

  The agent may learn a shaping reward that assigns reward when the state variable describing its distance to the beacon is reduced

  it can be learned even in the absence of an environmental reward

Konidaris and Barto (2006) cont.

- Introduce *c*: an agent-space sensation

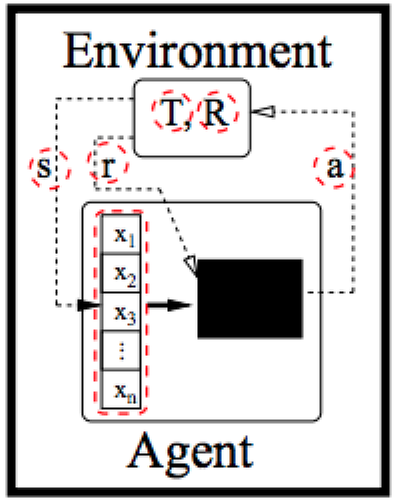
$$s_i^j = (d_i^j, c_i^j, r_i^j, v_i^j)$$

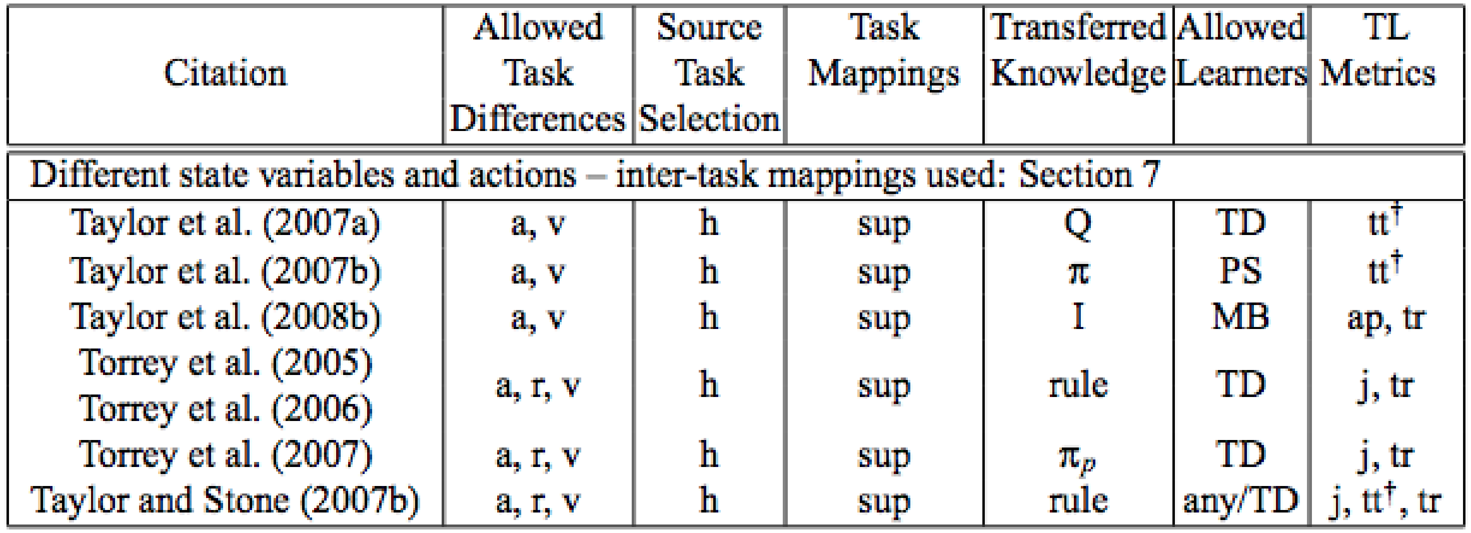
- V (expected cumulative reward) is not portable between tasks
  - the form and meaning of d (as a problem-space descriptor) may change from one task to another
- Learn expected return (L) for novel states rather V

$$L : c_i^j \mapsto v_i^j.$$

Transfer between Different Actions and State Representations with Explicit Mappings

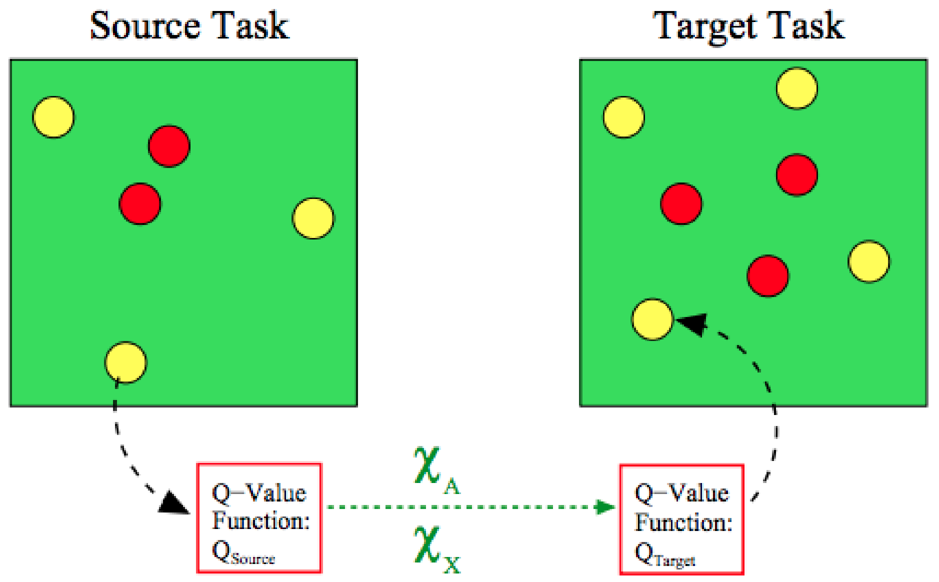
- allow the source task and target task to have different state variables and actions
- An explicit mapping between the tasks is given.





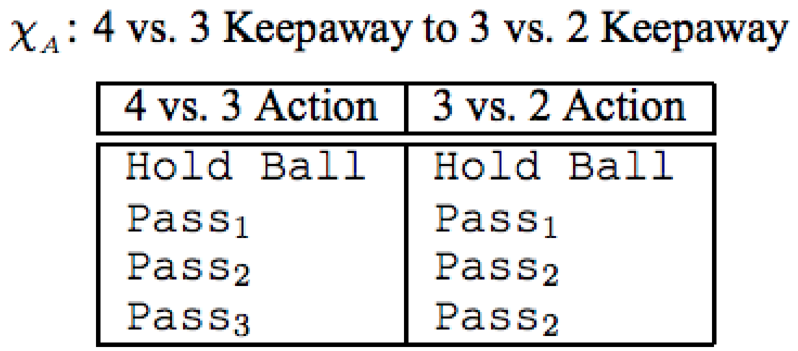
Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Different state variables and actions – inter-task mappings used: Section 7						
Taylor et al. (2007a)	a, v	h	sup	Q	TD	tt†
Taylor et al. (2007b)	a, v	h	sup	π	PS	tt†
Taylor et al. (2008b)	a, v	h	sup	I	MB	ap, tr
Torrey et al. (2005)	a, r, v	h	sup	rule	TD	j, tr
Torrey et al. (2006)						
Torrey et al. (2007)	a, r, v	h	sup	π_p	TD	j, tr
Taylor and Stone (2007b)	a, r, v	h	sup	rule	any/TD	j, tt†, tr

Inter–task Mappings



Source Task
Target Task
Q–Value Function: Q_{Source}
χ_A
χ_X
Q–Value Function: Q_{Target}

Figure 7: In Value Function Transfer, learned Q-values from the source task are transformed via inter-task mappings. These transformed Q-values are then used to initialize agents in the target task, biasing their learning with low-level knowledge from the source task.



χ_A: 4 vs. 3 Keepaway to 3 vs. 2 Keepaway

4 vs. 3 Action	3 vs. 2 Action
Hold Ball	Hold Ball
Pass_1	Pass_1
Pass_2	Pass_2
Pass_3	Pass_2

Table 3: This table describes the (full) action mapping from 4 vs. 3 to 3 vs. 2.

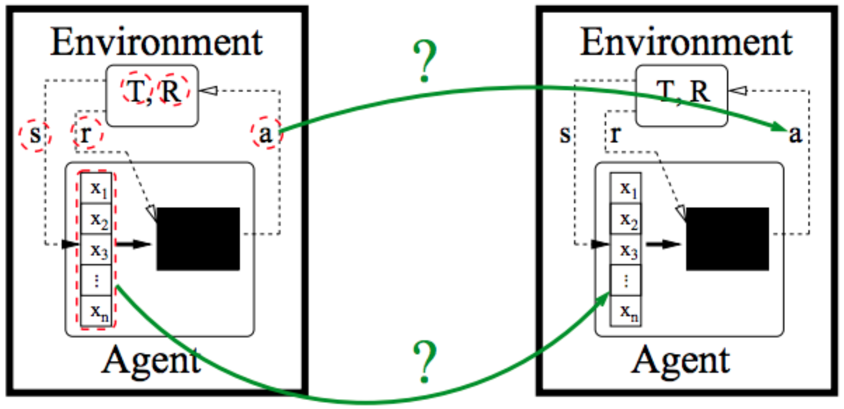
Image source: Taylor (2007)

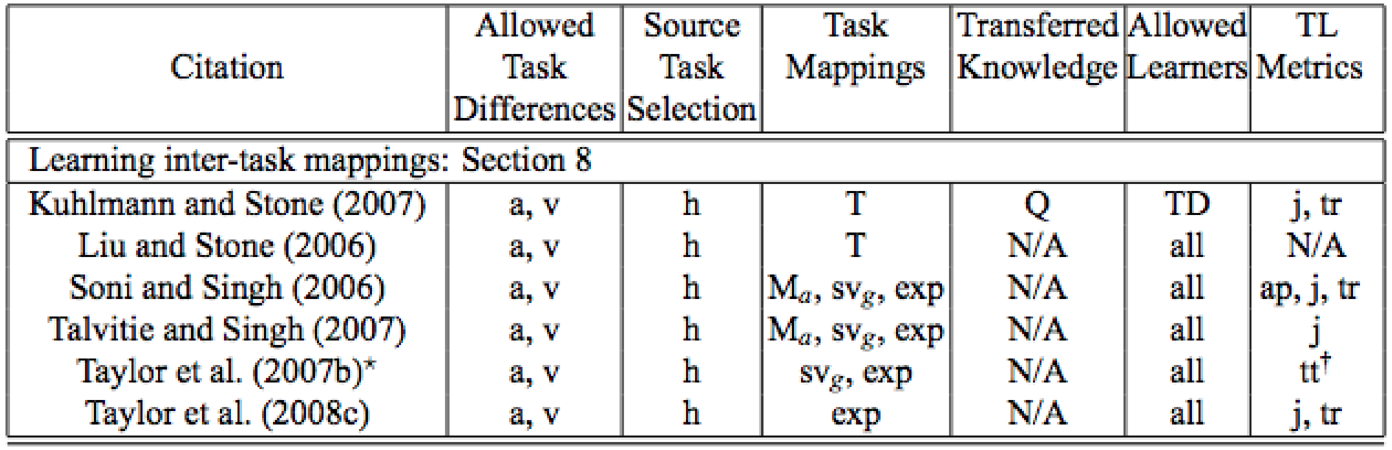
Transfer between Different Actions and State Representations with Explicit Mappings contd.

- Some approaches
    - Provide explicit mapping from source task, initialize target Q-values according to mapped source task values.
        - Taylor et al. (2007a)
    - Extract conditional "advice" rules from source task to serve as soft constraints to target policy (SVM learning Q-values).
        - Torry et al. (2005, 2006)
    - Extract chains of "advice" rules from source tied together as finite state machines to guide early exploration in target training.
        - Torry et al. (2007)

Learning Task Mappings

- In the previous section, task mappings were manually provided.
- Often the human cannot be involved.
- How can we automatically generate these mappings?





Citation	Allowed Task Differences	Source Task Selection	Task Mappings	Transferred Knowledge	Allowed Learners	TL Metrics
Learning inter-task mappings: Section 8						
Kuhlmann and Stone (2007)	a, v	h	T	Q	TD	j, tr
Liu and Stone (2006)	a, v	h	T	N/A	all	N/A
Soni and Singh (2006)	a, v	h	M_a, sv_g, exp	N/A	all	ap, j, tr
Talvitie and Singh (2007)	a, v	h	M_a, sv_g, exp	N/A	all	j
Taylor et al. (2007b)*	a, v	h	sv_g, exp	N/A	all	tt†
Taylor et al. (2008c)	a, v	h	exp	N/A	all	j, tr

Learning Task Mappings

- Some approaches
  - Given a complete description of any games (R,S, & T – the full MDP), produce an abstract representation – a ″rule graph″. Match the target's rule graph to the most isomorphic source graph (among multiple sources). Transfer the matching states.
    - Kuhlmann and Stone (2007)
  - Explore and exploit from a pool of candidate state mappings (assuming source and target states correspond to each other)
    - Soni and Singh (2006), Talvitie and Singh (2007)
  - Save all source episodes, build transition model of target with a small set of runs. Test model's prediction error against source episodes offline using possible mappings.
    - Taylor et al. (2008c)