Learning a Transfer Function for Reinforcement Learning Problems

Tom Croonenborghs
Biosciences and Technology Department, KH Kempen University College, Geel, Belgium

Kurt Driessens, Maurice Bruynooghe
Declarative Languages and Artificial Intelligence, Katholieke Universiteit Leuven, Leuven, Belgium

K.U.Leuven Association

AAAI'08: Transfer Learning for Complex Tasks
July 14 2008, Chicago
Inductive Transfer for Reinforcement Learning

Problem

Reinforcement Learning (RL) often approached as *tabula rasa* learning technique

- forced to perform *random exploration*
- quickly becomes infeasible or impractical in complex domains

Solution: Use Transfer Learning!

- learn in a *source task*
- leverage knowledge in *target task*

A lot of different approaches, e.g. see Lisa’s talk

- often a hand-coded mapping is needed to relate features from a source task to a target task
Inductive Transfer for Reinforcement Learning

Problem
Reinforcement Learning (RL) often approached as *tabula rasa* learning technique

- forced to perform *random exploration*
- quickly becomes infeasible or impractical in complex domains

Solution: Use Transfer Learning!

- learn in a *source task*
- leverage knowledge in *target task*

A lot of different approaches, e.g. see Lisa’s talk

- often a hand-coded mapping is needed to relate features from a source task to a target task
How to get rid of the expert?

**Desiderata**

Transfer knowledge from a source task to a target task with a different state (and action) representation *without* the need for a human-designed mapping.

**Some Existing Work**

**Liu & Stone 2006** Use a graph-matching algorithm to find similarities between state variables

- need for a complete and correct transition function

**Kuhlmann & Stone 2007** Similar approach tailored for GGP

**Taylor et al. 2007** Learn a classifier that predicts the index of a particular state variable given a transition

- state variables needs to be arranged into task-independent clusters
- similarities based on state transition might not give the entire answer
How to get rid of the expert?

Desiderata

Some Existing Work

Liu & Stone 2006 Use a graph-matching algorithm to find similarities between state variables
- need for a complete and correct transition function

Kuhlmann & Stone 2007 Similar approach tailored for GGP

Taylor et al. 2007 Learn a classifier that predicts the index of a particular state variable given a transition
- state variables needs to be arranged into task-independent clusters
- similarities based on state transition might not give the entire answer
- learning a mapping between states versus state variables
Outline of our approach

1. Using exploration to transfer knowledge
2. Which “transfer action” to choose?
3. Learn a transfer function
Outline of our approach

1. Using exploration to transfer knowledge

2. Which “transfer action” to choose?

3. Learn a transfer function
sometimes choose a “transfer action”

Outline of our approach

1. Using exploration to transfer knowledge

2. Which “transfer action” to choose?

3. Learn a transfer function
Sometimes choose a “transfer action”

Outline of our approach

1. Using exploration to transfer knowledge
2. Which “transfer action” to choose?
3. Learn a transfer function
Outline of our approach

1. Using exploration to

2. Which “transfer action” to choose?

3. Learn a transfer function

sometimes choose a “transfer action” at least as good as the best action according to current utility function.
Outline of our approach

1. Using exploration to sometimes choose a “transfer action” at least as good as the best action according to current utility function

2. Which “transfer action” to choose?

3. Learn a transfer function
Inductive Transfer for RL

Learning a Transfer Function

Preliminary Experiment

Conclusions

Outline of our approach

1. Using exploration

2. Which “transfer action” at least as good as the best action according to current utility function

3. Learn a transfer function

sometimes choose a “transfer action” predicts probability that transfer action is at least as good as that action
This talk as a single algorithm
This talk as a single algorithm

1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5: \hspace{1em} $Ex_Q \leftarrow \emptyset$
6: \hspace{1em} $Ex_P \leftarrow \emptyset$
7: \hspace{1em} generate a starting state $s_0$
8: \hspace{1em} $i \leftarrow 0$
9: \hspace{1em} repeat {for each step of episode}
10: \hspace{2em} if \text{rand()} < \epsilon_t \text{ then} \{\text{select transfer action}\}
11: \hspace{2em} $a_i = \pi_s(\text{argmax}_t \ p(t, s))$
12: \hspace{1em} $t_i = \text{argmax}_t \ p(t, s) \{\text{remember transfer state}\}$
13: \hspace{1em} else \{\text{select exploration action}\}
14: \hspace{2em} $a_i = a \text{ with prob.} \frac{e^{\frac{Q_e(s, a)}{\tau}}}{\sum_{b \neq a} e^{\frac{Q_e(s, b)}{\tau}}}$
15: \hspace{1em} end if
16: \hspace{1em} take action $a_i$, observe $r_i$ and $s_{i+1}$
17: \hspace{1em} $i \leftarrow i + 1$
18: \hspace{1em} until $s_i$ is terminal
19: \hspace{1em} $q_{i-1} \leftarrow r_{i-1}$
20: \hspace{1em} for $j = i - 2$ to 0 do
21: \hspace{2em} $q_j \leftarrow r_j + \gamma q_{j+1}$
22: \hspace{2em} generate example $x_q = (s_j, a_j, q_j)$
23: \hspace{2em} $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$
24: \hspace{2em} if $a$ was selected as transfer action, i.e. $a_j = \pi_s(\text{argmax}_t \ p(t, s_j))$ then
25: \hspace{3em} if $q_j \geq \text{max}_a \hat{Q}_e(s_j, a)$ then
26: \hspace{4em} generate example $x_p = (t_j, s_j, \text{`transfer'})$
27: \hspace{3em} else
28: \hspace{4em} generate example $x_p = (t_j, s_j, \text{`no transfer'})$
29: \hspace{3em} end if
30: \hspace{2em} end if
31: \hspace{2em} $Ex_P \leftarrow Ex_P \cup \{x_p\}$
32: \hspace{1em} end if
33: \hspace{1em} Update $\hat{Q}_e$ using $Ex_Q$ to produce $\hat{Q}_{e+1}$
34: \hspace{1em} Update $\hat{P}_e$ using $Ex_P$ to produce $\hat{P}_{e+1}$
35: \hspace{1em} $e \leftarrow e + 1$
36: \hspace{1em} until no more episodes
1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5: $E_{\text{Q}} \leftarrow \emptyset$
6: $E_{\text{P}} \leftarrow \emptyset$
7: generate a starting state $s_0$
8: $i \leftarrow 0$
9: repeat {for each step of episode}
10: ... 
11: until $s_i$ is terminal
12: ... 
13: until no more episodes
This talk as a single algorithm

1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5: $\text{Ex}_Q \leftarrow \emptyset$
6: $\text{Ex}_P \leftarrow \emptyset$
7: generate a starting state $s_0$
8: $i \leftarrow 0$
9: repeat {for each step of episode}
10: if $\text{rand}() < \epsilon_t$ then {select transfer action}
11: $a_i = \pi_s(\text{argmax}_t p(t, s))$
12: $t_i = \text{argmax}_t p(t, s)$ {remember transfer state}
13: else {select exploration action}
14: $a_i = a$ with prob. $\frac{e^{-\frac{1}{\tau} \hat{Q}_e(s, a)}}{\sum_{b \neq a} e^{-\frac{1}{\tau} \hat{Q}_e(s, b)}}$
15: end if
16: take action $a_i$, observe $r_i$ and $s_{i+1}$
17: $i \leftarrow i + 1$
18: until $s_i$ is terminal
19: $q_{i-1} \leftarrow r_{i-1}$
20: for $j = i - 2$ to 0 do
21: $q_j \leftarrow r_j + \gamma q_{j+1}$
22: generate example $x_q = (s_j, a_j, q_j)$
23: $\text{Ex}_Q \leftarrow \text{Ex}_Q \cup \{x_q\}$
24: if $a$ was selected as transfer action, i.e. $a_j = \pi_s(\text{argmax}_t p(t, s_j))$ then
25: if $q_j \geq \max a \hat{Q}_e(s_j, a)$ then
26: generate example $x_p = (t_j, s_j, \text{‘transfer’})$
27: else
28: generate example $x_p = (t_j, s_j, \text{‘no transfer’})$
29: end if
30: $\text{Ex}_P \leftarrow \text{Ex}_P \cup \{x_p\}$
31: end if
32: end for
33: Update $\hat{Q}_e$ using $\text{Ex}_Q$ to produce $\hat{Q}_{e+1}$
34: Update $\hat{P}_e$ using $\text{Ex}_P$ to produce $\hat{P}_{e+1}$
35: $e \leftarrow e + 1$
36: until no more episodes
1. \texttt{initialize the Q-function hypothesis} $\hat{Q}$
2. \texttt{initialize the transfer function} $\hat{P}$
3. $\epsilon \leftarrow 0$
4. repeat
5. \{for each episode\}
6. \texttt{Ex} $\leftarrow \emptyset$
7. \texttt{Ex} $\leftarrow \emptyset$
8. \texttt{generate a starting state} $s_0$
9. \{for each step of episode\}
10. if $\text{rand}() < \epsilon$ then \{select transfer action\}
11. \hspace{1cm} $a_i = \pi_s(\text{argmax}_t \ p(t, s))$
12. \hspace{1cm} $t_i = \text{argmax}_t \ p(t, s)$ \{remember transfer state\}
13. else\{select exploration action\}
14. \hspace{1cm} $a_i = a$ with prob. $\frac{\hat{Q}_e(s, a)}{\sum_{b \neq a} \hat{Q}_e(s, b)}$
15. \hspace{1cm} $t_i = \text{argmax}_t \ p(t, s)$ \{remember transfer state\}
16. \hspace{1cm} $\epsilon \leftarrow \epsilon + 1$
17. \hspace{1cm} $i \leftarrow i + 1$
18. \until $s_i$ is terminal
19. $r_i \leftarrow r_{i-1}$
20. for $j = i - 2 \text{ to } 0$ do
21. \hspace{1cm} $q_j \leftarrow r_j + \gamma q_{j+1}$
22. \hspace{1cm} generate example $x_q = (s_j, a_j, q_j)$
23. \hspace{1cm} $\text{Ex} \leftarrow \text{Ex} \cup \{x_q\}$
24. \hspace{1cm} if $a_i$ was selected as transfer action, i.e. $a_i = \pi_s(\text{argmax}_t \ p(t, s_j))$ then
25. \hspace{2cm} $\text{Ex} \leftarrow \text{Ex} \cup \{x_t \ p(t, s_j)\}$
26. \hspace{2cm} if $a_i = \hat{Q}_e(s_j, a)$ then
27. \hspace{3cm} generate example $x_p = (t_j, s_j, 'transfer')$
28. \hspace{2cm} else
29. \hspace{3cm} generate example $x_p = (t_j, s_j, 'no transfer')$
30. \hspace{1cm} $\text{Ex} \leftarrow \text{Ex} \cup \{x_p\}$
31. \until no more episodes
Transfer knowledge during exploration

The policy during exploration

- exploration policy $\pi_e(s) : S \rightarrow A$
  - guarantees that each state-action combination has a non-zero probability of being visited
- alter exploration policy
  - motivated by previous transfer learning approaches and guidance
  - currently very simple
    - $\epsilon$-based transfer knowledge exploration
    - with a probability of $\epsilon$ choose a “transfer action”
Transfer actions?

The transfer function \( p(s, t): S_s \times S_t \rightarrow [0, 1] \)

- for each state \( s \)
  - probability that the best action for \( s \) (according to the source policy)
  - is at least as good for \( t \)
  - as the best action according to the current \( Q \)-function

- transfer policy \( \pi_t(t) = \pi_s(\arg\max_s p(s, t)) \)

What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task
- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action
Transfer actions?

The transfer function $p(s, t): S_s \times S_t \rightarrow [0, 1]$

- for each state $s$
  - probability that the best action for $s$ (according to the source policy)
  - is at least as good for $t$
  - as the best action according to the current $Q$-function

- transfer policy $\pi_t(t) = \pi_s(\arg\max_s p(s, t))$

What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task

- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action
Transfer actions?

The transfer function $p(s, t): S_s \times S_t \rightarrow [0, 1]$

- for each state $s$
  - probability that the best action for $s$ (according to the source policy)
  - is at least as good for $t$
  - as the best action according to the current $Q$-function
- transfer policy $\pi_t(t) = \pi_s(\text{argmax}_s p(s, t))$

What about action mappings?

Sorry, that is still part of future work

- assume that actions in the source task are executable in the target task
- if this is not the case, we generate a negative reward when the agent tries to execute a non-available action
This talk as a single algorithm

1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5: $Ex_Q \leftarrow \emptyset$
6: $Ex_P \leftarrow \emptyset$
7: generate a starting state $s_0$
8: $i \leftarrow 0$
9: repeat {for each step of episode}
10: if $\text{rand}() < \epsilon_t$ then {select transfer action}
11: $a_i = \pi_s(\text{argmax}_t p(t, s))$
12: $t_i = \text{argmax}_t p(t, s)$ {remember transfer state}
13: else {select exploration action}
14: $a_i = a$ with prob. $\frac{e^{Q_e(s,b)}}{\sum_{b \neq a} e^{Q_e(s,b)}}$
15: end if
16: take action $a_i$, observe $r_i$ and $s_{i+1}$
17: $i \leftarrow i + 1$
18: until $s_i$ is terminal
19: $q_{i-1} \leftarrow r_{i-1}$
20: for $j = i - 2$ to 0 do
21: $q_j \leftarrow r_j + \gamma q_{j+1}$
22: generate example $x_q = (s_j, a_j, q_j)$
23: $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$
24: if $a$ was selected as transfer action, i.e. $a_j = \pi_s(\text{argmax}_t p(t, s_j))$ then
25: if $q_j \geq \text{max}_a \hat{Q}_e(s_j, a)$ then
26: generate example $x_p = (t_j, s_j, \text{‘transfer’})$
27: else
28: generate example $x_p = (t_j, s_j, \text{‘no transfer’})$
29: end if
30: $Ex_P \leftarrow Ex_P \cup \{x_p\}$
31: end if
32: end for
33: Update $\hat{Q}_e$ using $Ex_Q$ to produce $\hat{Q}_{e+1}$
34: Update $\hat{P}_e$ using $Ex_P$ to produce $\hat{P}_{e+1}$
35: $e \leftarrow e + 1$
36: until no more episodes
Learning a Transfer Function for Reinforcement Learning Problems

9: repeat {for each step of episode}
10: if rand() < $\epsilon_t$ then {select transfer action}
11: \[ a_i = \pi_s(\arg\max_t p(t, s)) \]
12: \[ t_i = \arg\max_t p(t, s) \] {remember transfer state}
13: else {select exploration action}
14: \[ a_i = a \text{ with prob. } \frac{\hat{Q}_e(s,a)}{\sum_{b \neq a} e^{\frac{\hat{Q}_e(s,b)}{\tau}}} \]
15: end if
16: take action $a_i$, observe $r_i$ and $s_{i+1}$
17: $i \leftarrow i + 1$
18: until $s_i$ is terminal

Learning a Transfer Function for Reinforcement Learning Problems
This talk as a single algorithm

1: initialize the Q-function hypothesis \( \hat{Q}_0 \)
2: initialize the transfer function \( \hat{P}_0 \)
3: \( e \leftarrow 0 \)
4: repeat {for each episode}
5: \( \text{Ex}_Q \leftarrow {} \)
6: \( \text{Ex}_P \leftarrow {} \)
7: generate a starting state \( s_0 \)
8: \( i \leftarrow 0 \)
9: repeat {for each step of episode}
10: if \( \text{rand()} < \epsilon_t \) then {select transfer action}
11: \( a_i = \pi_s(\arg\max_t p(t, s)) \)
12: \( t_i = \arg\max_t p(t, s) \) {remember transfer state}
13: else {select exploration action}
14: \( a_i = a \) with prob. \( \frac{e^{\frac{-\hat{Q}_e(s, a)}{\tau}}}{\sum_{b \neq a} e^{\frac{-\hat{Q}_e(s, b)}{\tau}}} \)
15: end if
16: take action \( a_i \), observe \( r_i \) and \( s_{i+1} \)
17: \( i \leftarrow i + 1 \)
18: until \( s_i \) is terminal
19: \( q_{i-1} \leftarrow r_{i-1} \)
20: for \( j = i - 2 \) to 0 do
21: \( q_j \leftarrow r_j + \gamma q_{j+1} \)
22: generate example \( x_q = (s_j, a_j, q_j) \)
23: \( \text{Ex}_Q \leftarrow \text{Ex}_Q \cup \{x_q\} \)
24: if \( a \) was selected as transfer action, i.e. \( a_j = \pi_s(\arg\max_t p(t, s_j)) \) then
25: if \( q_j \geq \max_a \hat{Q}_e(s_j, a) \) then
26: generate example \( x_p = (t_j, s_j, 'transfer') \)
27: else
28: generate example \( x_p = (t_j, s_j, 'no transfer') \)
29: end if
30: \( \text{Ex}_P \leftarrow \text{Ex}_P \cup \{x_p\} \)
31: end if
32: end for
33: Update \( \hat{Q}_e \) using \( \text{Ex}_Q \) to produce \( \hat{Q}_{e+1} \)
34: Update \( \hat{P}_e \) using \( \text{Ex}_P \) to produce \( \hat{P}_{e+1} \)
35: \( e \leftarrow e + 1 \)
36: until no more episodes
Learning the transfer function

- need to learn $p(s, t) : S_s \times S_t \rightarrow [0, 1]$
- at the end of every learning episode:
  - for every executed transfer action $a_t$
  - compare 2 utility-values
    - MC based $Q_{MC}(s, a_t)$
    - learned generalized $\hat{Q}(s, a)$
  - learning examples
    - “transfer” if $Q_{MC}(s, a_t) \geq \max_a \hat{Q}(s, a)$
    - and “no transfer” otherwise
This talk as a single algorithm

1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5:   $Ex_Q \leftarrow \emptyset$
6:   $Ex_P \leftarrow \emptyset$
7:   generate a starting state $s_0$
8:   $i \leftarrow 0$
9: repeat {for each step of episode}
10:   if $\text{rand}() < \epsilon_t$ then {select transfer action}
11:     $a_i = \pi(s(\arg\max_t p(t, s)))$
12:     $t_i = \arg\max_t p(t, s)$ {remember transfer state}
13:   else {select exploration action}
14:     $a_i = a$ with prob. $\frac{e^{-\frac{\hat{Q}_e(s, a)}{\tau}}}{\sum_{b \neq a} e^{-\frac{\hat{Q}_e(s, b)}{\tau}}}$
15: end if
16: take action $a_i$, observe $r_i$ and $s_{i+1}$
17: $i \leftarrow i + 1$
18: until $s_i$ is terminal
19: $q_{i-1} \leftarrow r_{i-1}$
20: for $j = i - 2$ to 0 do
21:   $q_j \leftarrow r_j + \gamma q_{j+1}$
22:   generate example $x_q = (s_j, a_j, q_j)$
23:   $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$
24: if $a$ was selected as transfer action, i.e. $a_j = \pi(s(\arg\max_t p(t, s_j)))$ then
25:   if $q_j \geq \max_a \hat{Q}_e(s_j, a)$ then
26:     generate example $x_p = (t_j, s_j, \text{‘transfer’})$
27:   else
28:     generate example $x_p = (t_j, s_j, \text{‘no transfer’})$
29: end if
30: $Ex_P \leftarrow Ex_P \cup \{x_p\}$
31: end if
32: end for
33: Update $\hat{Q}_e$ using $Ex_Q$ to produce $\hat{Q}_{e+1}$
34: Update $\hat{P}_e$ using $Ex_P$ to produce $\hat{P}_{e+1}$
35: $e \leftarrow e + 1$
36: until no more episodes
This talk as a single algorithm

1: initialize the Q-function hypothesis \( \hat{Q}_0 \)
2: initialize the transfer function \( \hat{P}_0 \)
3: \( e \leftarrow 0 \)
4: repeat {for each episode}
5: \( E_x Q \leftarrow \emptyset \)
6: \( E_x P \leftarrow \emptyset \)
7: generate a starting state \( s_0 \)
8: \( i \leftarrow 0 \)
9: repeat {for each step of episode}
10: if \( \text{rand}() < \epsilon \)
11: \( \{ \text{select transfer action} \} \)
12: \( a_i = \pi(s, \arg\max_t p(t, s)) \)
13: \( t_i = \arg\max_t p(t, s) \) \{remember transfer\}
14: else \{select exploration action\}
15: \( a_i = a \) with prob. \( \epsilon \)
16: \( a_i \neq a \) \( \hat{Q}(s_i, a_i) \)
17: \( \hat{Q}(s_i, a_i) \)
18: take action \( a_i \), observe \( r_i \) and \( s_{i+1} \)
19: \( q_i = q_{i-1} \)
20: for \( j = i - 2 \) to 0 do
21: \( q_j = r_j + \gamma q_{j+1} \)
22: generate example \( x_q = (s_j, a_j, q_j) \)
23: \( E_x Q \leftarrow E_x Q \cup \{x_q\} \)
24: if \( a_j \) was selected as transfer action, i.e.
25: \( a_j = \pi(s, \arg\max_t p(t, s)) \) then
26: if \( q_j \geq \max_a \hat{Q}_e(s_j, a) \) then
27: generate example \( x_p = (t_j, s_j, \text{`transfer'}) \)
28: else
29: generate example \( x_p = (t_j, s_j, \text{`no transfer'}) \)
30: \( E_x P \leftarrow E_x P \cup \{x_p\} \)
31: end if
32: end for
This talk as a single algorithm

1: initialize the Q-function hypothesis \( \hat{Q}_0 \)
2: initialize the transfer function \( \hat{P}_0 \)
3: \( e \leftarrow 0 \)
4: repeat \{for each episode\}
5: \( \text{Ex}_Q \leftarrow \emptyset \)
6: \( \text{Ex}_P \leftarrow \emptyset \)
7: generate a starting state \( s_0 \)
8: \( i \leftarrow 0 \)
9: repeat \{for each step of episode\}
10: if \( \text{rand}() < \epsilon_t \) then \{select transfer action\}
11: \( a_i = \pi_s(\text{argmax}_t p(t, s)) \)
12: \( t_i = \text{argmax}_t p(t, s) \) \{remember transfer state\}
13: else \{select exploration action\}
14: \( a_i = a \) with prob. \( \frac{e^{\frac{\hat{Q}_e(s,a)}{\tau}}}{\sum_{b \neq a} e^{\frac{\hat{Q}_e(s,b)}{\tau}}} \)
15: end if
16: take action \( a_i \), observe \( r_i \) and \( s_{i+1} \)
17: \( i \leftarrow i + 1 \)
18: until \( s_i \) is terminal
19: \( q_{i-1} \leftarrow r_{i-1} \)
20: for \( j = i - 2 \) to 0 do
21: \( q_j \leftarrow r_j + \gamma q_{j+1} \)
22: generate example \( x_q = (s_j, a_j, q_j) \)
23: \( \text{Ex}_Q \leftarrow \text{Ex}_Q \cup \{x_q\} \)
24: if \( a \) was selected as transfer action, i.e. \( a_j = \pi_s(\text{argmax}_t p(t, s_j)) \) then
25: if \( q_j \geq \text{max}_a \hat{Q}_e(s_j, a) \) then
26: generate example \( x_p = (t_j, s_j, \text{‘transfer’}) \)
27: else
28: generate example \( x_p = (t_j, s_j, \text{‘no transfer’}) \)
29: end if
30: \( \text{Ex}_P \leftarrow \text{Ex}_P \cup \{x_p\} \)
31: end if
32: end for
33: Update \( \hat{Q}_e \) using \( \text{Ex}_Q \) to produce \( \hat{Q}_{e+1} \)
34: Update \( \hat{P}_e \) using \( \text{Ex}_P \) to produce \( \hat{P}_{e+1} \)
35: \( e \leftarrow e + 1 \)
36: until no more episodes
Preliminary Experiment

A prototype of the suggested approach

- just a proof of principle
  - SARSA-algorithm
  - \( TG \) to learn utility function

- no continuous learning of the transfer function
  - learned a transfer function once with \( \text{TILDE} \)
  - based on 100 episodes in the target task and actions transferred from random states in the source task
Environments

**Source task**

<table>
<thead>
<tr>
<th>1,1</th>
<th>1,2</th>
<th>1,3</th>
<th>1,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,1</td>
<td>2,2</td>
<td>2,3</td>
<td>1,2</td>
</tr>
<tr>
<td>3,1</td>
<td>3,2</td>
<td>3,3</td>
<td>1,3</td>
</tr>
<tr>
<td>4,1</td>
<td>4,2</td>
<td>4,3</td>
<td>G</td>
</tr>
</tbody>
</table>

**Target task**

- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal
Environments

Source task
- 4 rooms
- 4 doors
- 4 keys
- Goal (G)

Target task
- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal
Environments

Source task

Target task

- at most 500 actions per episode
- receives a reward of 1 if he exits the last room and 0 otherwise
- state representation includes dimensions of rooms, locations and colors of doors and keys, the keys the agent possesses, his location and the location of the goal
The learned transfer function

\[ y(t) = y(s) + 8? \]
\[ x(t) = x(s) + 8? \]
\[ y(t) = y(s) + 11? \]
\[ y(t) = y(s) + 9? \]

**Able to learn both**
- the shift in coordinates
- transfer only valuable if the agent has the key to leave current room
The learned transfer function

Able to learn both
- the shift in coordinates
- transfer only valuable if the agent has the key to leave current room
Convergence results

Average reward

Convergence results (reward) grid world

- Circles: with transfer
- Triangles: without transfer

Episode

Average reward
Convergence results

Average number of actions

Convergence results (actions) grid world

- Red circles: with transfer
- Blue triangles: without transfer

 Episode

Average #actions

0 50 100 150 200 250 300 350 400 450

0 100 200 300 400 500 600 700 800 900 1000
Convergence results

Average number of actions

Note that currently negative transfer is not avoided.
Conclusions and Future Work

Conclusions

- proposed a first step towards automatically learning a mapping function
- transfer through guided exploration
- proof of principle with promising results

Future Work

- evaluate in more detail
- incorporate quality of possible transfer in the policy
  - avoid negative transfer
- incorporate incremental learning of the transfer function
- incorporating action descriptions into the transfer function
Conclusions and Future Work

**Conclusions**

- proposed a first step towards automatically learning a mapping function
- transfer through guided exploration
- proof of principle with promising results

**Future Work**

- evaluate in more detail
- incorporate quality of possible transfer in the policy
  - avoid negative transfer
- incorporate incremental learning of the transfer function
- incorporating action descriptions into the transfer function
Questions?

Let me start with a few....

Questions!

- is it really a good idea to base transfer on $Q$-values?
- avoiding negative transfer: does $p(s, t)$ give us enough information?
  - how low is too low?
  - measure relatively?
- substitute batch learning by a continuous incremental approach
  - what is the effect of not learning a transfer function based on random states?
  - how to deal with concept drift?
    - how quickly do you want to forget earlier examples?
- how to incorporate action descriptions into the transfer function?
Let me start with a few....

Questions!

- is it really a good idea to base transfer on $Q$-values?
- avoiding negative transfer: does $p(s, t)$ give us enough information?
  - how low is too low?
  - measure relatively?
- substitute batch learning by a continuous incremental approach
  - what is the effect of not learning a transfer function based on random states?
  - how to deal with concept drift?
    - how quickly do you want to forget earlier examples?
- how to incorporate action descriptions into the transfer function?