15-887
Planning, Execution and Learning

Learning in Planning:
Learning Cost Function

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

A bit of terminology

- Imitation Learning/Apprenticeship Learning/Learning from Demonstrations/Robot Programming by Demonstrations
  - Methods for programming robot behavior via demonstrations [Schaal & Atkeson, ‘94], [Abbeel & Ng, ’04], [Pomerleau et al., ‘89], [Ratliff & Bagnell, ‘06], [Billard, Calinon & Dillmann, ’13], [Sammut et al., ‘92],…
- Major classes of Imitation Learning:
  - Learning policies directly from demonstrated trajectories or supervised learning [Schaal & Atkeson, ‘94], [Pomerleau et al., ‘89],…
  - Learning a cost function (or reward function) from demonstrations and then using it to generate plans (policies) [Abbeel & Ng, ’04], [Ratliff & Bagnell, ‘06], ...

A bit of terminology

- Imitation Learning/Apprenticeship Learning/Learning from Demonstrations/Robot Programming by Demonstrations
  - Methods for programming robot behavior via demonstrations [Schaal & Atkeson, '94], [Abbeel & Ng, '04], [Pomerleau et al., '89], [Ratliff & Bagnell, '06], [Billard, Calinon & Dillmann, '13], [Sammut et al., '92],…

- Major classes of Imitation Learning:
  - Learning policies directly from demonstrated trajectories or supervised learning [Schaal & Atkeson, '94], [Pomerleau et al., '89],…
  - Learning a cost function (or reward function) from demonstrations and then using it to generate plans (policies) [Abbeel & Ng, '04], [Ratliff & Bagnell, '06], ...

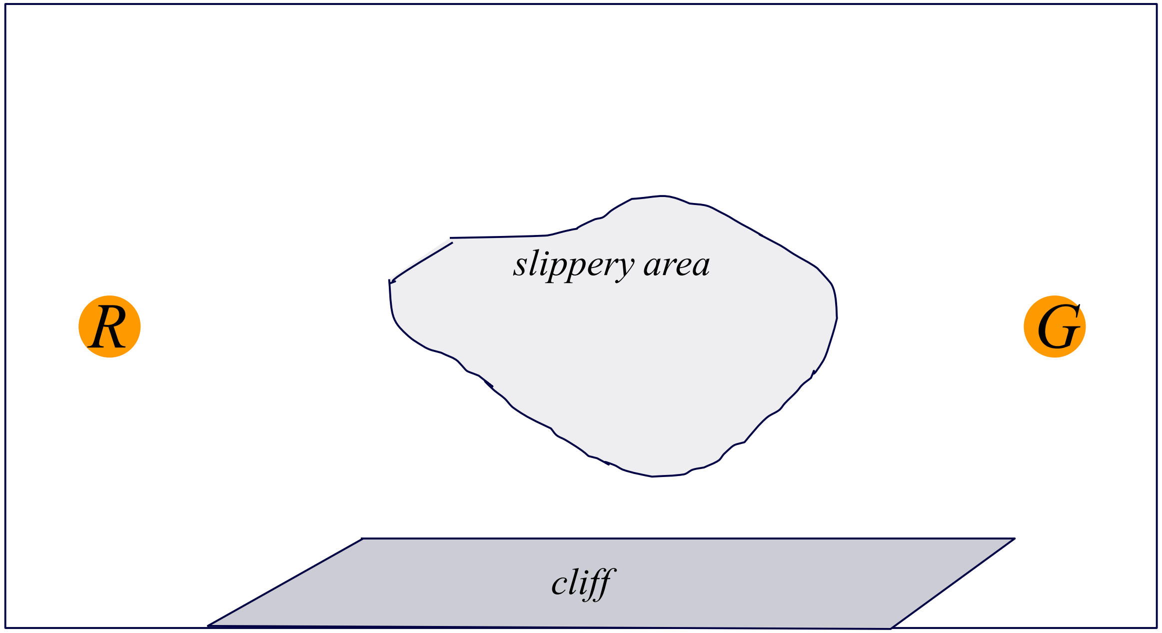
Inverse Reinforcement Learning (IRL), Inverse Optimal Control

Learning a cost function

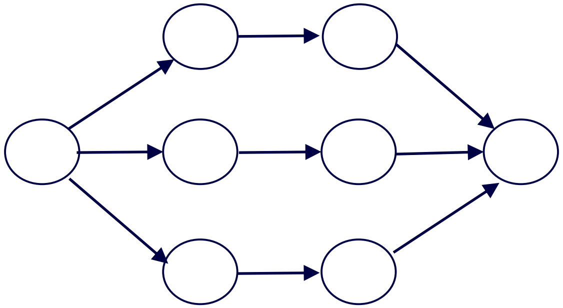
- **Recover a cost function that makes given demonstrations optimal plans** [Ratliff, Silver & Bagnell, ’09]

Example

- Consider a (simple) outdoor navigation example



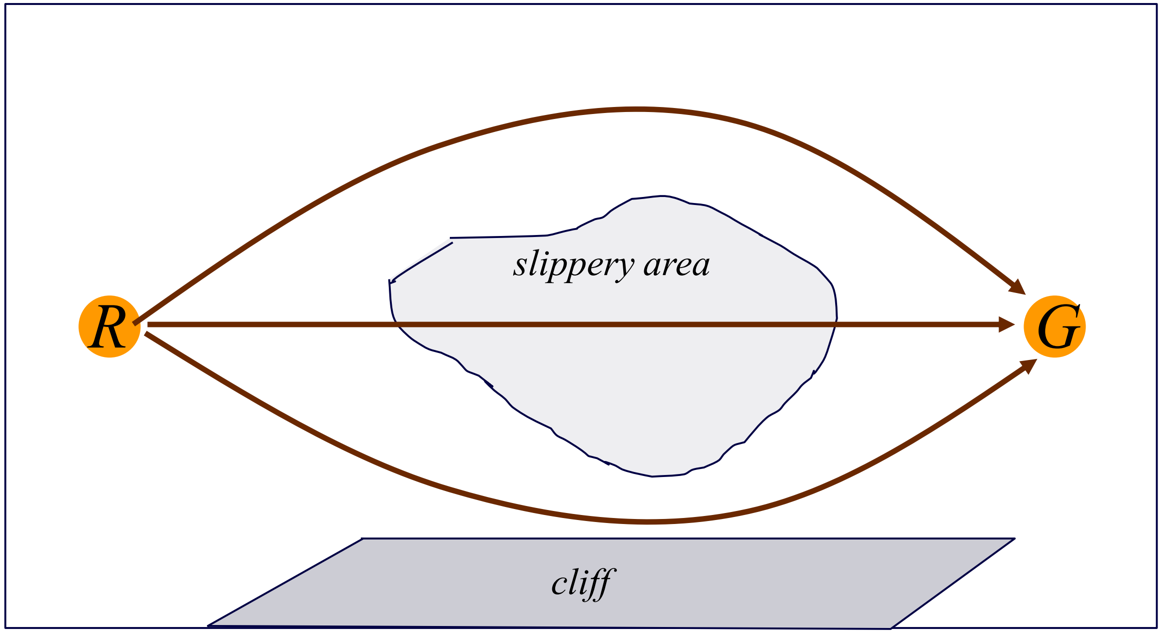
Modeled as graph search



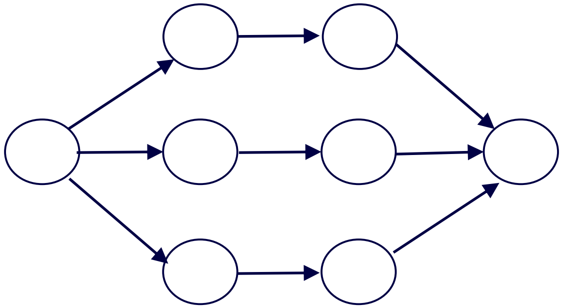
Example

- Consider a (simple) outdoor navigation example

Can we teach the planner to avoid slippery areas and driving close to the cliff (without manually tweaking a cost function)?



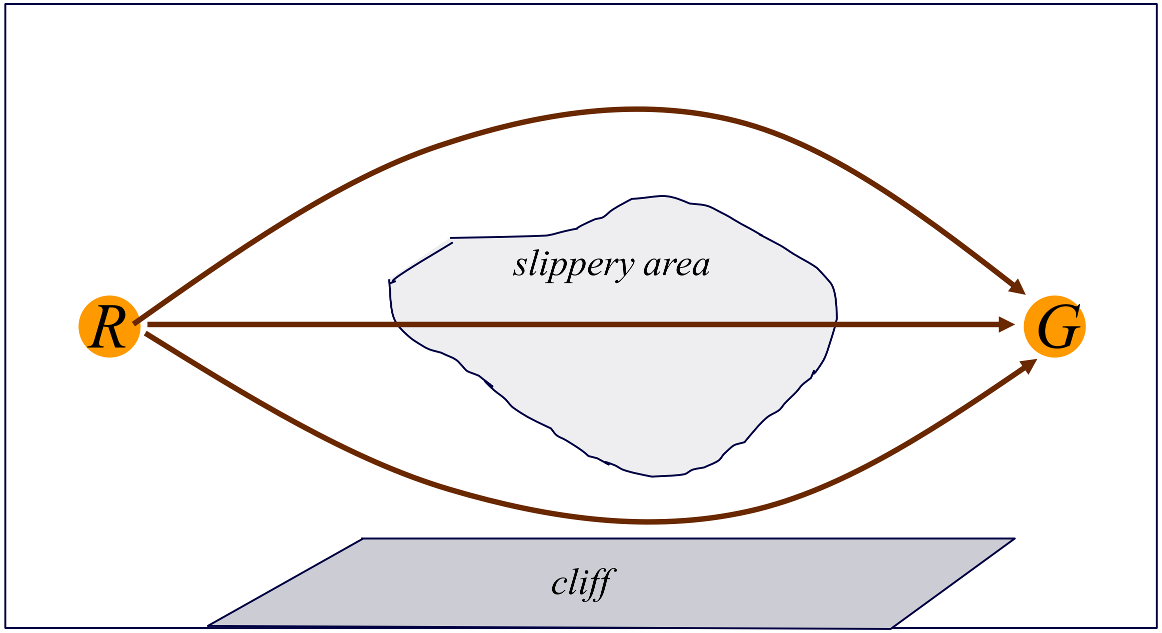
Modeled as graph search



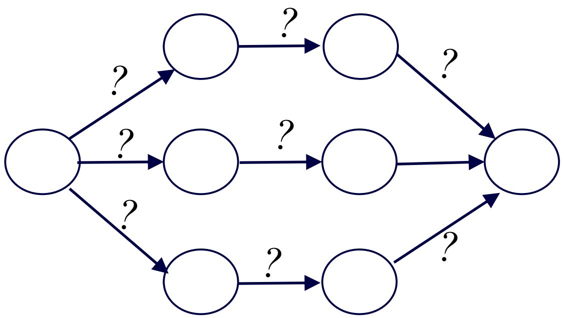
Example

- Consider a (simple) outdoor navigation example

Can we teach the planner to avoid slippery areas and driving close to the cliff (without manually tweaking a cost function)?



= learning the "right" cost function

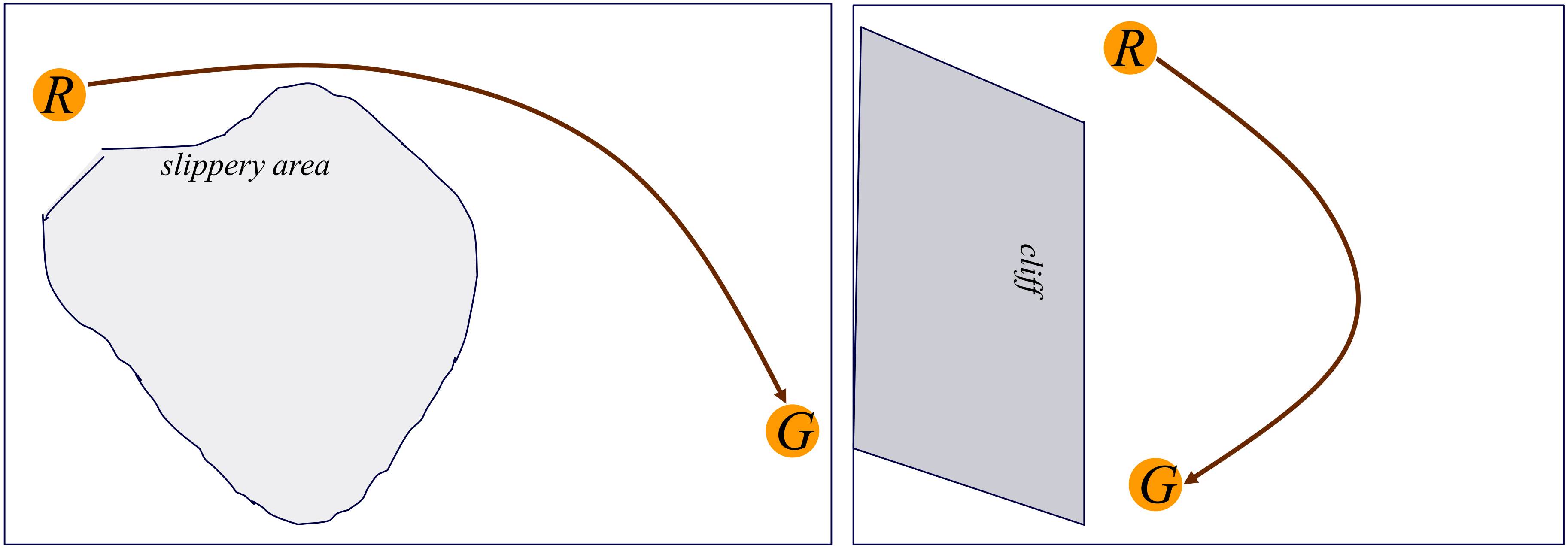


Example

- Consider a (simple) outdoor navigation example

Can we teach the planner to avoid slippery areas and driving close to the cliff (without manually tweaking a cost function)?

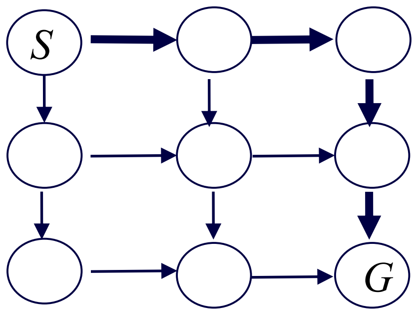
A user gives N demonstrations of what paths are good.
We want a cost function for which these demonstrated trajectories are least-cost plans



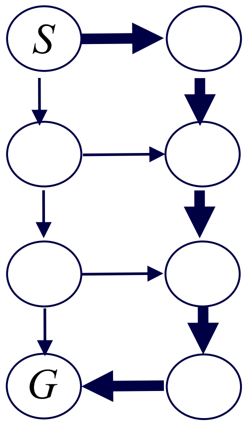
Example

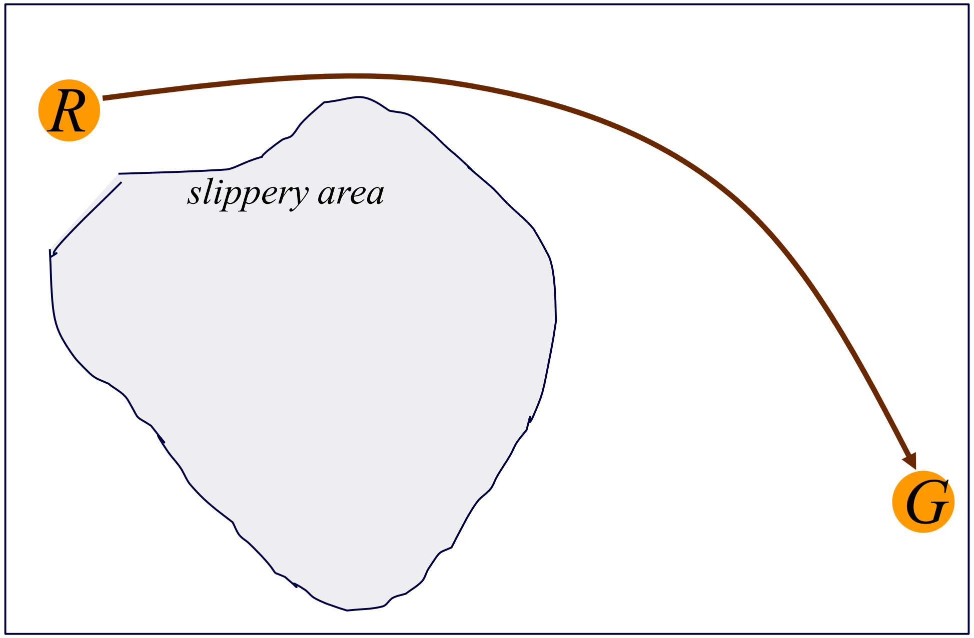
- Consider a (simple) outdoor navigation example

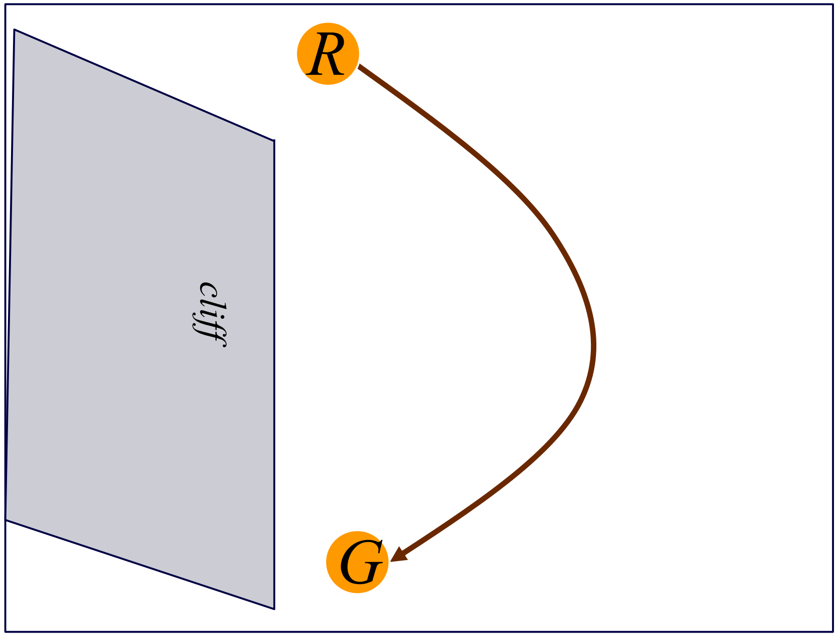
Demonstration d_1 on graph G_1



Demonstration d_2 on graph G_2



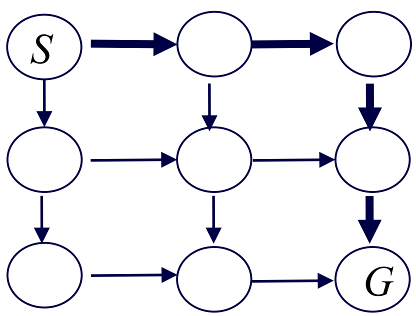




Example

- Consider a (simple) outdoor navigation example

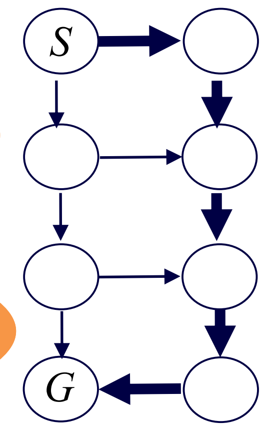
Demonstration d_1 on graph G_1



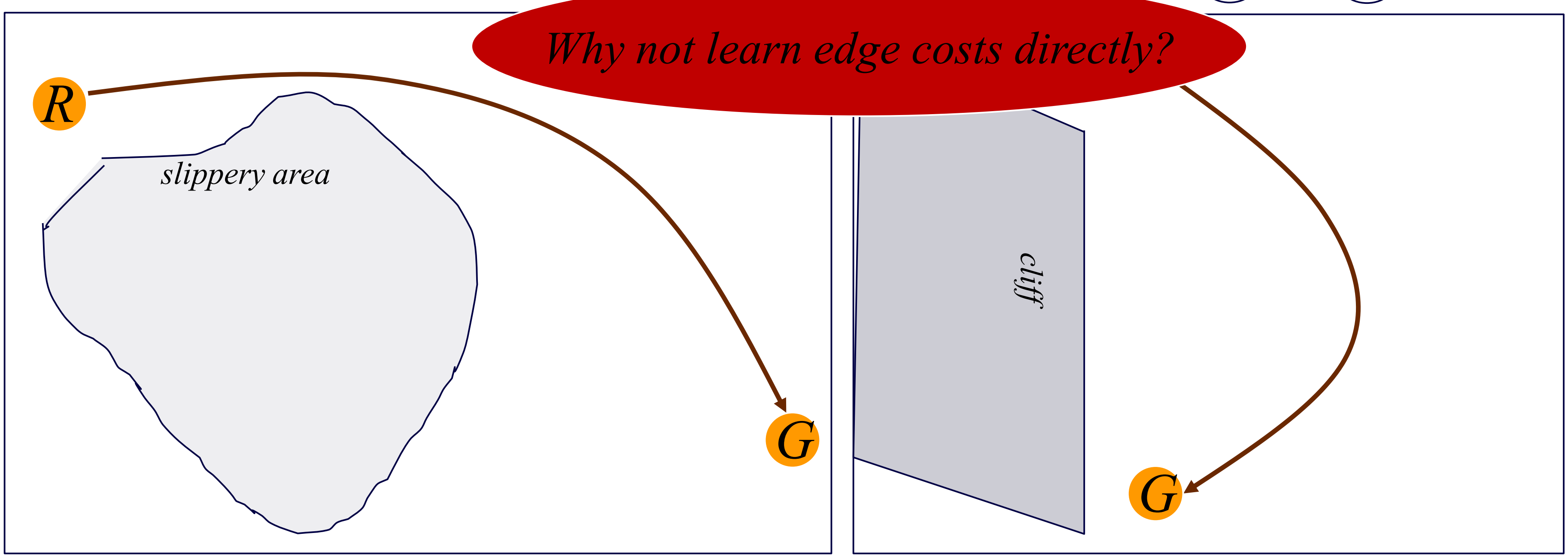
Compute cost function that makes these demonstrations optimal paths

Cost function – a function of features Φ: c(s,s') = f(ϕ(s,s'))

Demonstration d_2 on graph G_2



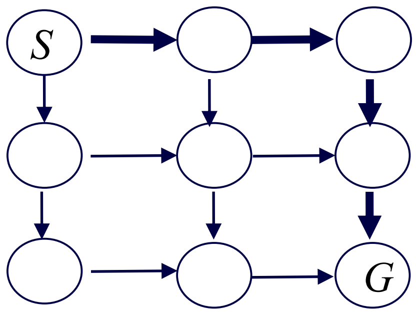
Why not learn edge costs directly?



Example

- Consider a (simple) outdoor navigation example

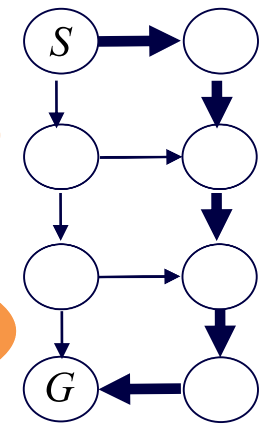
Demonstration d_1 on graph G_1



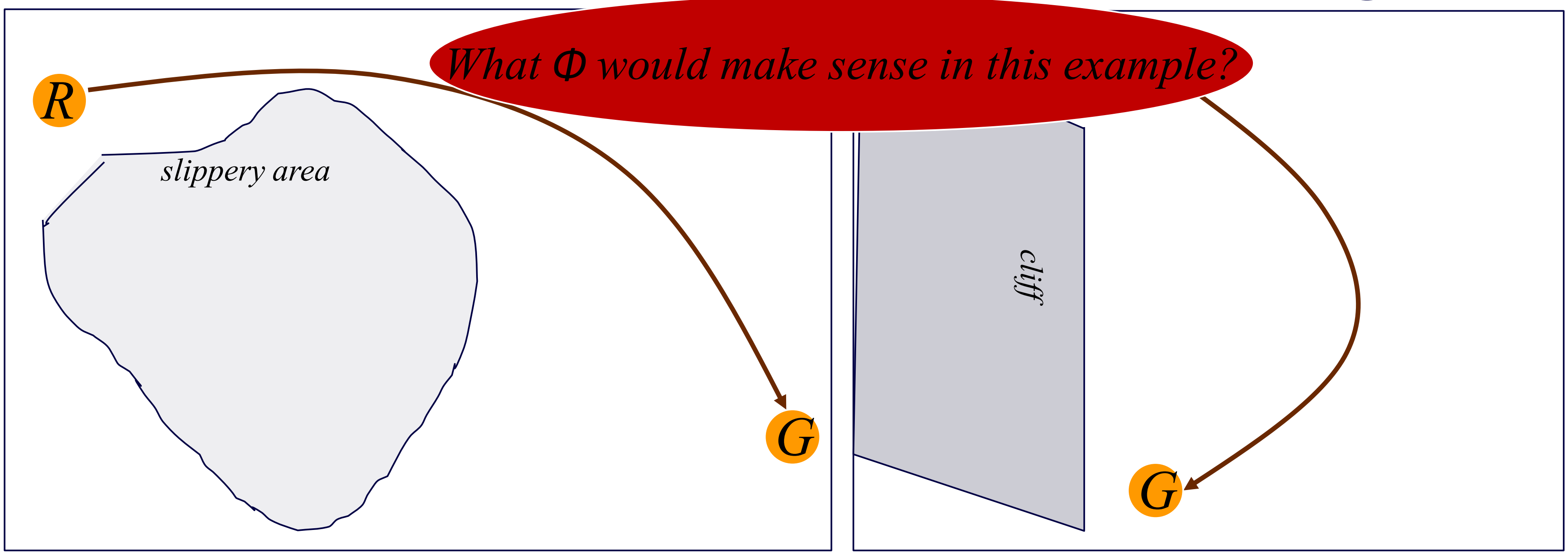
Compute cost function that makes these demonstrations optimal paths

Cost function – a function of features Φ: $c(s,s') = f(\phi(s,s'))$

Demonstration d_2 on graph G_2



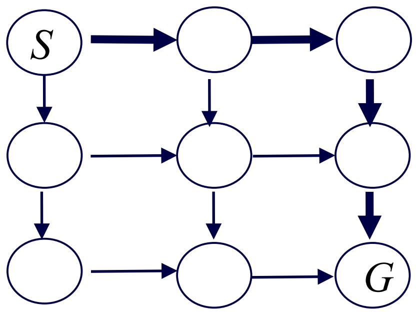
What Φ would make sense in this example?



Example

- Consider a (simple) outdoor navigation example

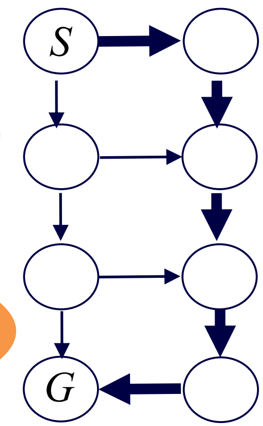
Demonstration d_1 on graph G_1



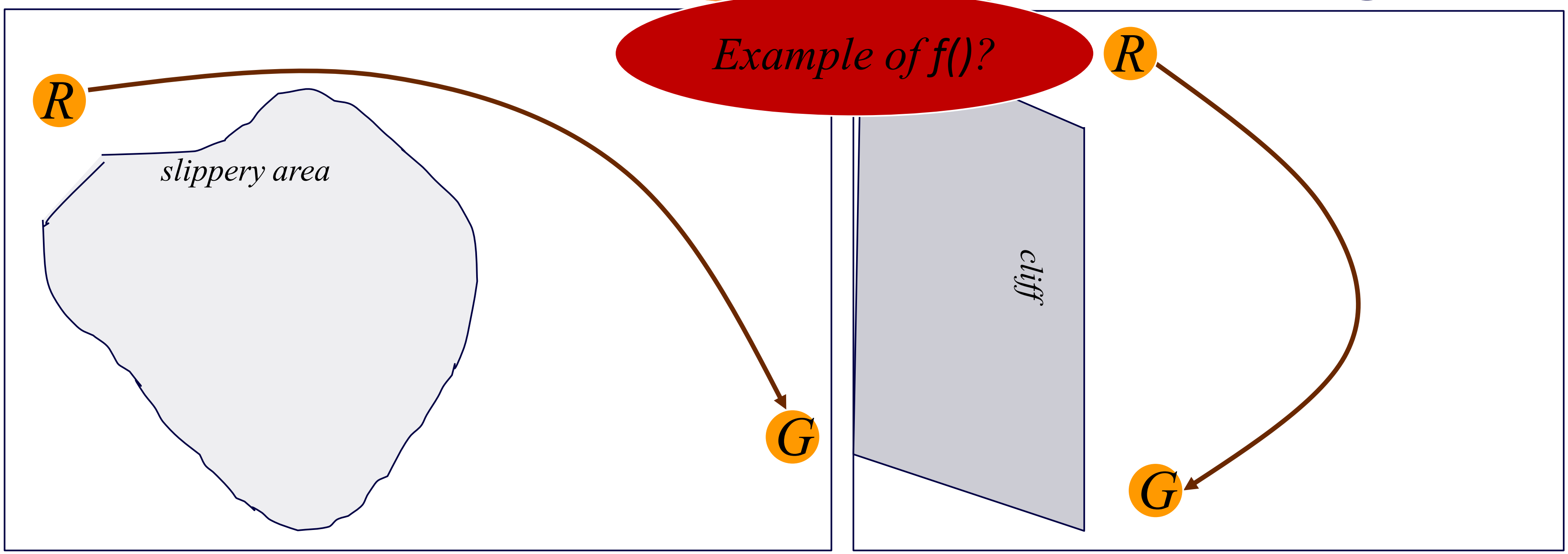
Compute cost function that makes these demonstrations optimal paths

Cost function – a function of features Φ: c(s,s') = f(ϕ(s,s'))

Demonstration d_2 on graph G_2



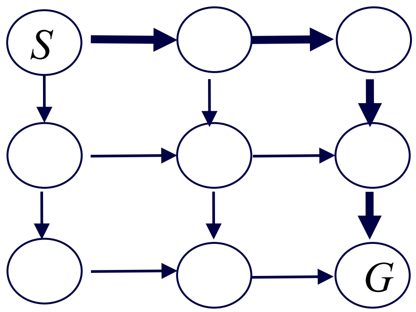
Example of f()?



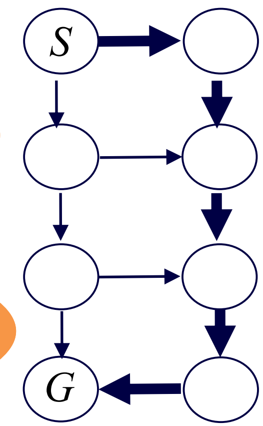
Example

- Consider a (simple) outdoor navigation example

Demonstration d_1 on graph G_1



Demonstration d_2 on graph G_2

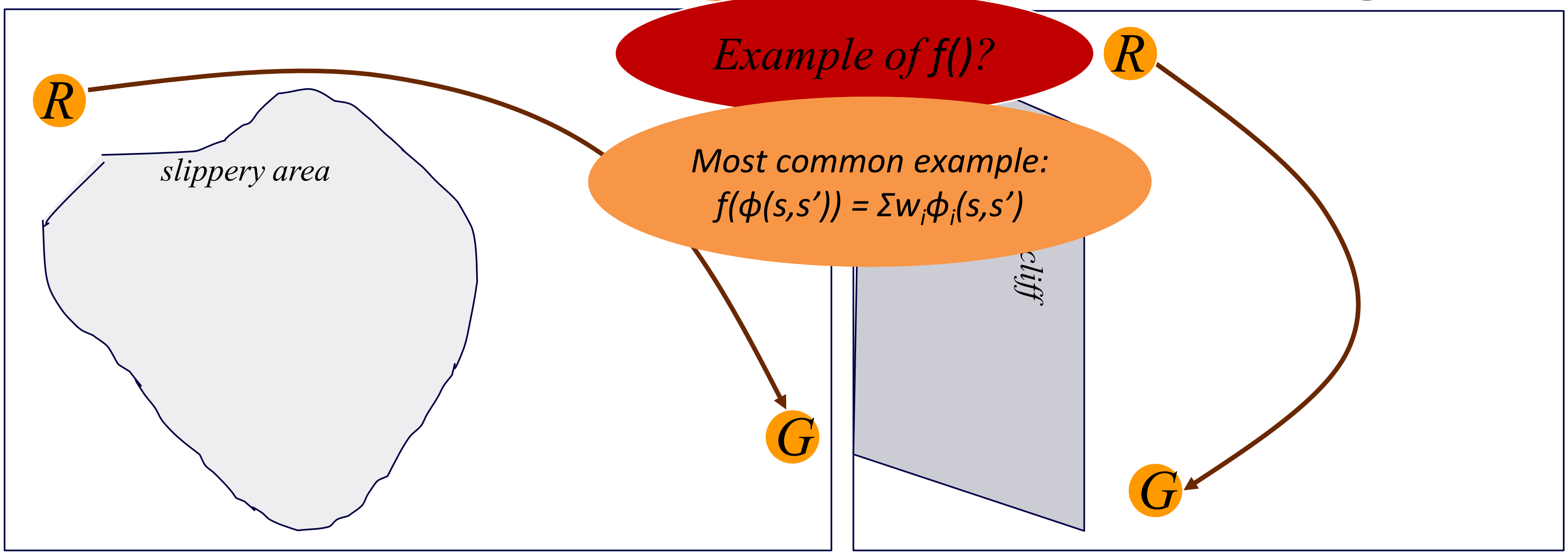


Compute cost function that makes these demonstrations optimal paths

Cost function – a function of features Φ: $c(s,s') = f(\phi(s,s'))$

Example of f()?

Most common example: $f(\phi(s,s')) = \Sigma w_i \phi_i(s,s')$



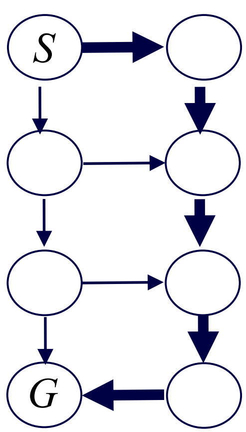
Example

- Consider a (simple) outdoor navigation example

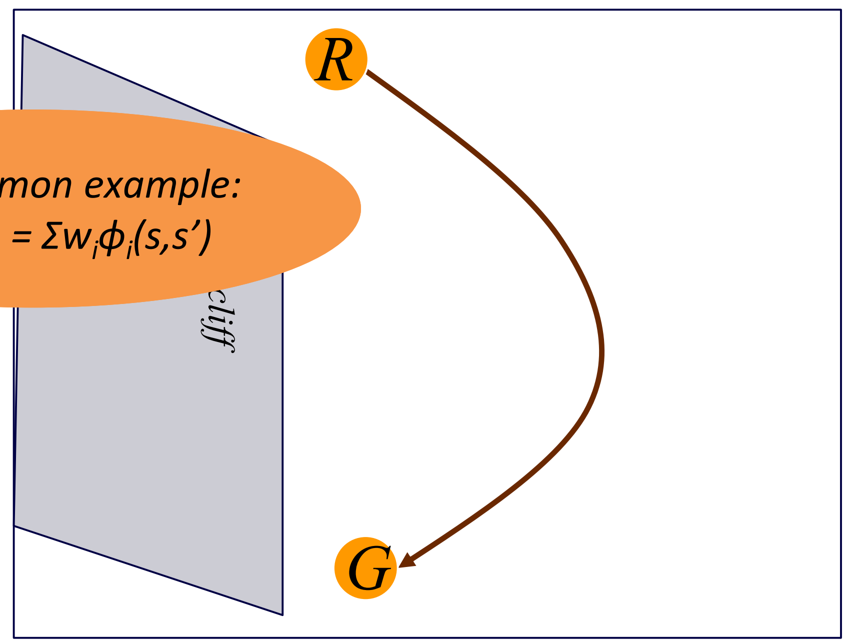
For example:
ϕ_0 : 1/(distance to slippery area)
ϕ_1 : 1/(distance to cliff)
ϕ_2 : length of the transition

Need to compute (learn) w_0, w_1, w_2 based on demonstrations

Demonstration d_2 on graph G_2



Most common example:
$f(\phi(s,s')) = \Sigma w_i \phi_i(s,s')$



LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1…N

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} c(s_k, s_{k+1})$

increase $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND (u,v) not in* $d_i\}$

decrease $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND (u,v) in* $d_i\}$

LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1…N

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} c(s_k, s_{k+1})$

increase $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND (u,v) not in* $d_i\}$

decrease $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND (u,v) in* $d_i\}$

Is π_i^* *always guaranteed to converge to* d_i*?*

LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1…N

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} c(s_k, s_{k+1})$

increase $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND (u,v) not in* $d_i\}$

decrease $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND (u,v) in* $d_i\}$

Any problem with arbitrary decrease of $f(\phi(,))$*?*

Any solutions?

LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1 …N

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} c(s_k, s_{k+1})$

increase ***log*** $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND (u,v) not in* $d_i\}$

decrease ***log*** $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND (u,v) in* $d_i\}$

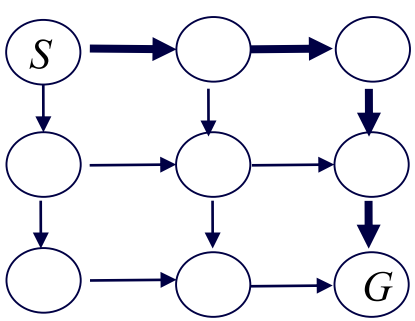
Example

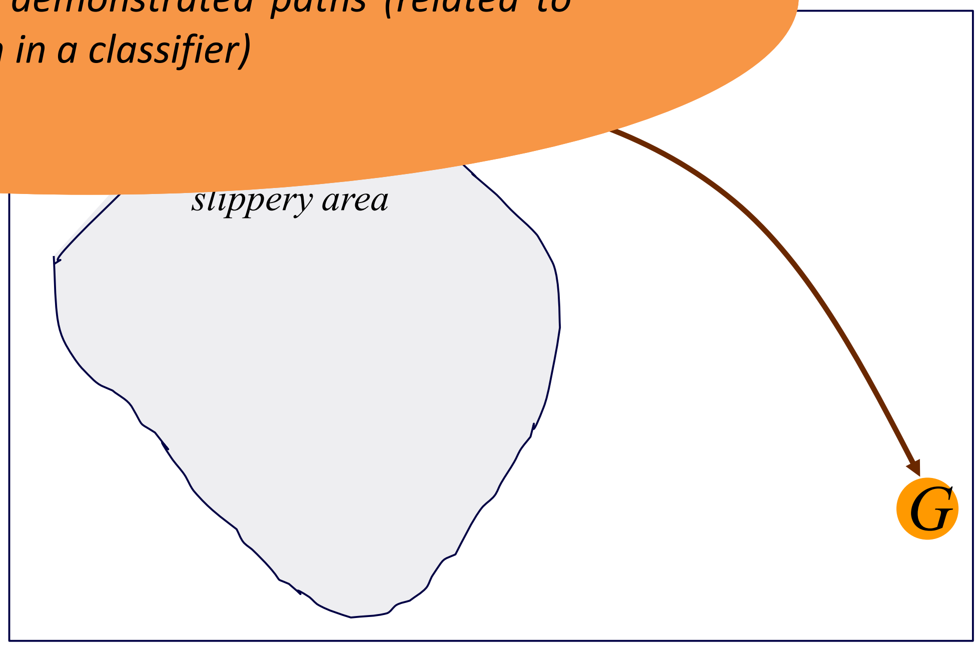
- Consider a (simple) outdoor navigation example

Suppose initial $w_0 = 0$. Any problem learning W?

Need a loss function that makes the algorithm learn harder to stay on the demonstrated paths (related to maximizing the margin in a classifier)

Demonstration d_1 on graph G_1





LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1…N

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} \{c(s_k, s_{k+1}) - \boldsymbol{l}(\mathbf{s_k}, \mathbf{s_{k+1}})\}$

increase log $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND (u,v) not in* $d_i\}$

decrease log $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND (u,v) in* $d_i\}$

Loss function penalizes being NOT on a demonstration path.
For example, $l(s,s')=0$ *if* (s,s') *on* d_i *and* $l(s,s')>1$ *otherwise*

LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1, \ldots d_N\}$ *on graphs* $\{G_1, \ldots, G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converged)

for i=1…N

How do we decide how to increase/decrease f(φ(,))?

update edge costs in graph G_i *using the current function* $f(\phi(,))$

plan an optimal path $\pi_i^* = \arg\min_{\pi_i} \sum_{k=0}^{length(\pi_i)-1} \{c(s_k, s_{k+1}) - l(s_k, s_{k+1})\}$

increase log $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *in* π_i^* *AND* (u,v) *not in* $d_i\}$

decrease log $f(\phi(,))$ *for edges (u,v) s.t.* $\{(u,v)$ *not in* π_i^* *AND* (u,v) *in* $d_i\}$

LEARCH (LEArning to searCH)

[Ratliff, Silver, Bagnell, 09]

Given demonstrations $\{d_1,\ldots d_N\}$ *on graphs* $\{G_1,\ldots,G_N\}$ *and features function* Φ

Need to compute $c(s,s') = f(\phi(s,s'))$ *s.t.* $d_i = \arg\min_{\pi_i} \sum_{i=1}^{N} c(\pi_i)$

While (Not Converg

for i=1…N

update edge costs in

plar

increase log

decreas

How do we decide how to increase/decrease $f(\phi(,))$*?*

Set dC vector as: +1 for all edges that need to be increased,
and -1 for all edges that need to be decreased.
Recompute $f(\phi(,))$ *to make a step in the direction of dC*

For example, if $f(\phi(s,s')) = \Sigma w_i \phi_i(s,s') = \Phi W$*, then:*

1. *Solve for vector dW from* $\Phi dW = dC$ *(e.g.,* $dW = (\Phi^T\Phi)^{-1}\Phi^T dC$ *)*
2. *Update W:* $W = W + \eta dW$

Learning cost in graphs vs. Learning rewards in MDPs

- Learning cost framework can be generalized to learning rewards in MDPs (typical Inverse Reinforcement Learning)

- Two broad frameworks to Inverse Reinforcement Learning in MDPs:

  – Max-margin [Ratliff & Bagnell, '06] – equivalent to the learning cost framework we just learned

  – Feature expectation matching [Abbeel & Ng, '04]

Summary

- Learning cost function is a way of learning from demonstrations

- Works by learning a cost function that makes demonstrations to be optimal solutions to planning problems

- Performance depends on the design of the features used to map states onto the cost function that is being learned