Practical RL: **R**epresentation, **I**nteraction, **S**ynthesis, and **M**ortality (**PRISM**)

Peter Stone

Director, Learning Agents Research Group (LARG)
Department of Computer Science
The University of Texas at Austin

Joint work with members of LARG
past and present

Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

- Autonomous agents
- Robotics
- Multiagent systems

Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

- Autonomous agents
- Robotics
- Multiagent systems
- Machine learning

Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

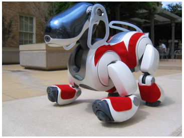
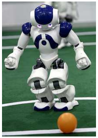
- Autonomous agents
- Robotics
- Multiagent systems
- Machine learning
  - **Reinforcement Learning**

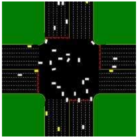
Research Question

To what degree can autonomous intelligent agents learn in the presence of teammates and/or adversaries in real-time, dynamic domains?

Research Areas

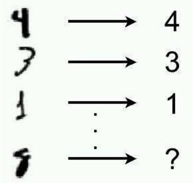
- Autonomous agents
- Robotics
- Multiagent systems
- Machine learning
  - **Reinforcement Learning**



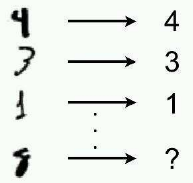
Reinforcement Learning

Supervised learning mature [WEKA]



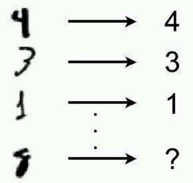
Reinforcement Learning

Supervised learning mature [WEKA]



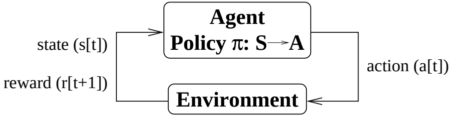
For agents, reinforcement learning most appropriate

Reinforcement Learning



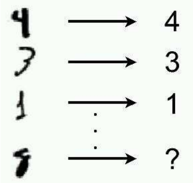
Supervised learning mature [WEKA]

For agents, reinforcement learning most appropriate

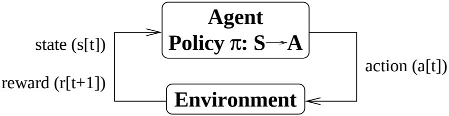


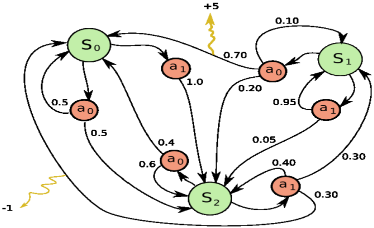
Reinforcement Learning

Supervised learning mature [WEKA]



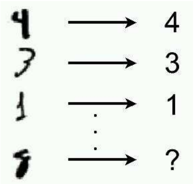
For agents, reinforcement learning most appropriate



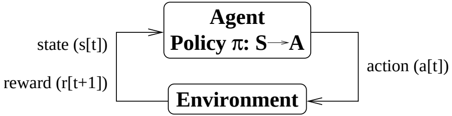


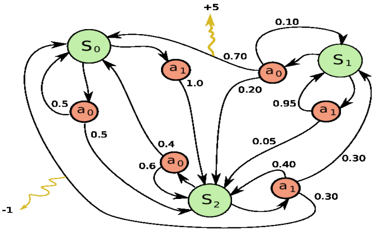
Reinforcement Learning

Supervised learning mature [WEKA]



For agents, reinforcement learning most appropriate

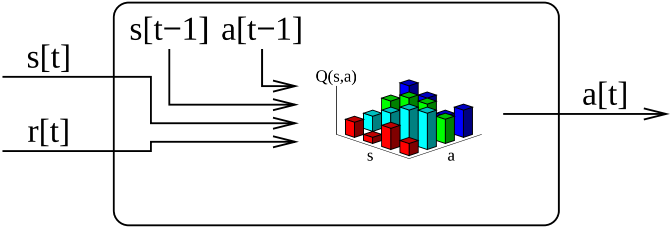




– Foundational theoretical results
– Applications require innovations to scale up

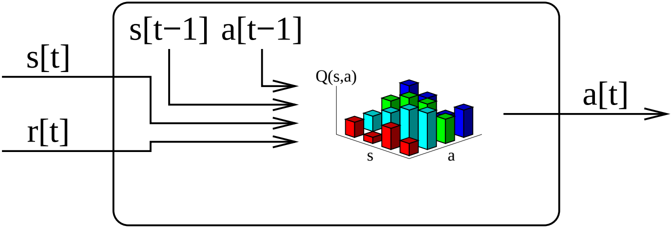
RL Theory

Success story: Q-learning converges to π^* [Watkins, 89]



RL Theory

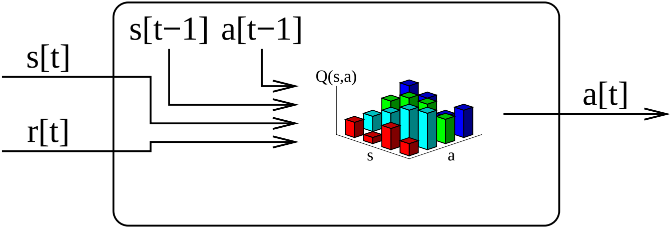
Success story: Q-learning converges to π^* [Watkins, 89]



– Table-based representation
– Visit every state infinitely often

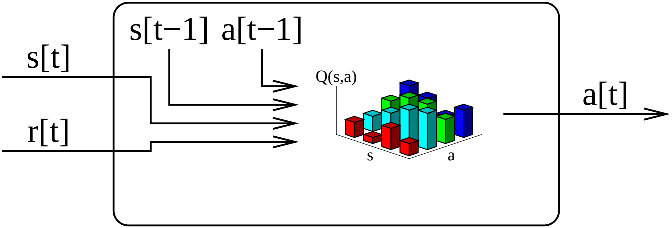
Function Approximation

In practice, visiting every state impossible

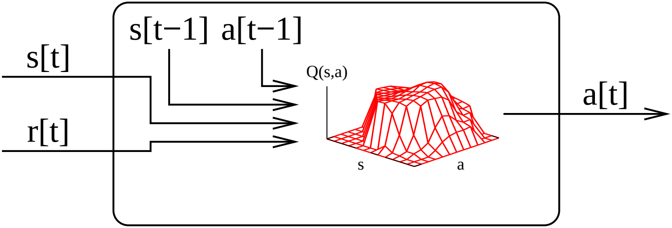


Function Approximation

In practice, visiting every state impossible



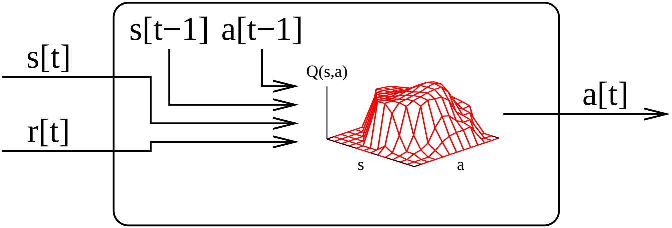
Function approximation of value function



Theoretical guarantees harder to come by

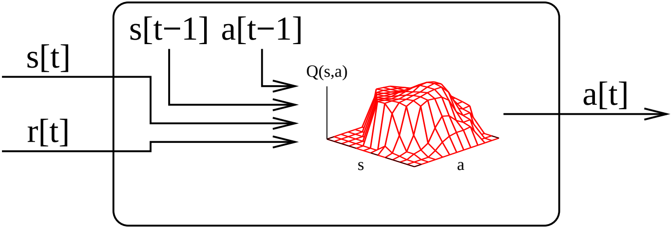
Batch Methods

In practice, often experience is scarce

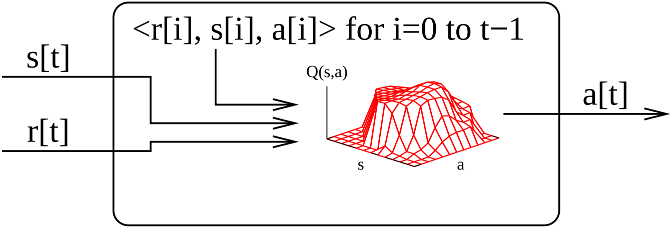


Batch Methods

In practice, often experience is scarce

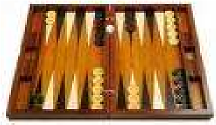


Save transitions:



Applications: Towards a Useful Tool

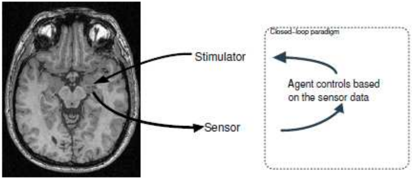
- Backgammon [Tesauro, '94]
- Helicopter control [Ng et al., '03]



Applications: Towards a Useful Tool

- Backgammon [Tesauro, '94]
- Helicopter control [Ng et al., '03]

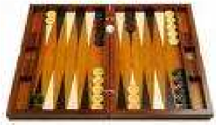
- Adaptive treatment of epilepsy [Pineau et al., '08]



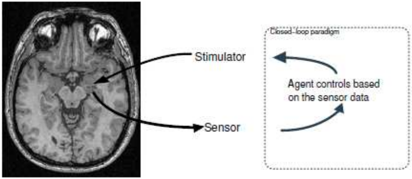
- Invasive species management,
  wildfire suppression [Dietterich et al., '13]

Applications: Towards a Useful Tool

- Backgammon [Tesauro, '94]
- Helicopter control [Ng et al., '03]

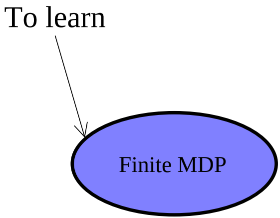


- Adaptive treatment of epilepsy [Pineau et al., '08]

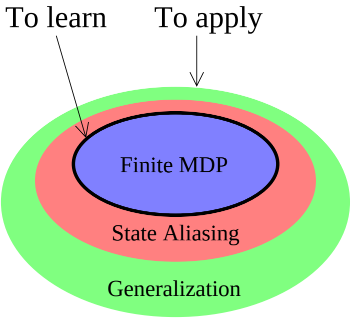


- Invasive species management, wildfire suppression [Dietterich et al., '13]
- Google Purchase of DeepMind, '14

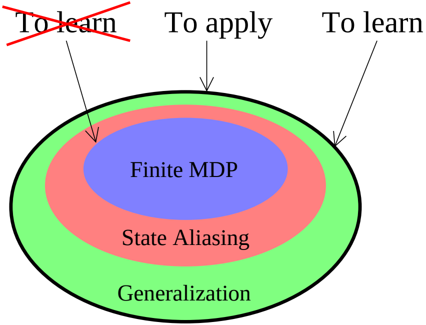
RL as a Tool



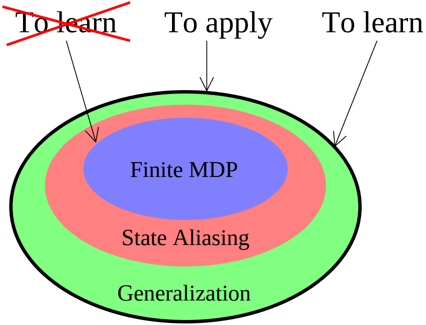
RL as a Tool


To learn
To apply
Finite MDP
State Aliasing
Generalization

RL as a Tool

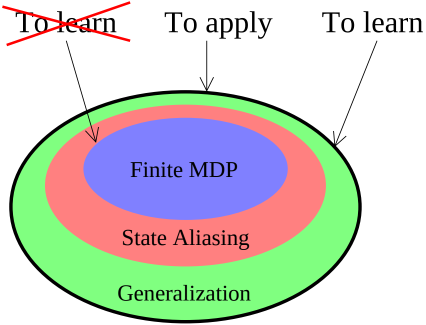

~~To learn~~
To apply
To learn
Finite MDP
State Aliasing
Generalization

RL as a Tool



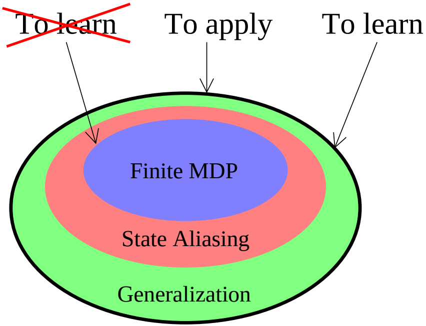
- Rather than "**Should** RL work?" . . .

RL as a Tool



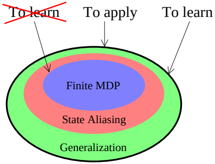
- Rather than "**Should** RL work?" . . .
- . . . "**Does** RL work?"

RL as a Tool



- Rather than "**Should** RL work?" . . .
- . . . "**Does** RL work?"
  - When not: "**How** can we make it work?"

Practical RL



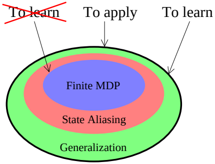
- **R**epresentation

- **I**nteraction

- **S**ynthesis

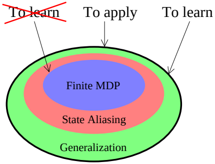
- **M**ortality

Practical RL



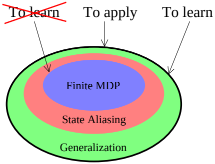
- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
- **S**ynthesis
- **M**ortality

Practical RL



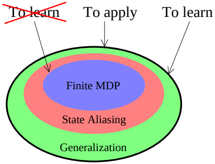
- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
- **M**ortality

Practical RL



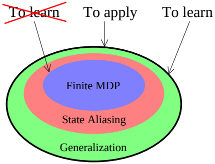
- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality

Practical RL



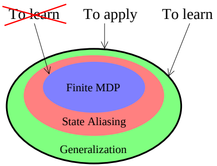
- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

Practical RL



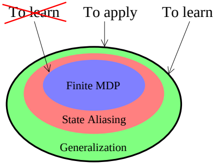
- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLES [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLES [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

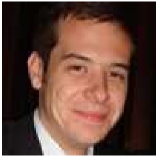
Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-maxq [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: texplore [Hester & S., MLJ 2013]

TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots

Todd Hester and Peter Stone



Machine Learning, 2013

Reinforcement Learning

Model-free Methods

- Learn a value function directly from interaction with environment
- Can run in real-time, but not very sample efficient

Reinforcement Learning

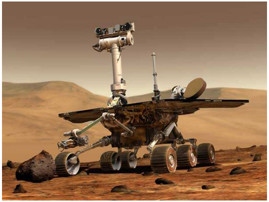
Model-free Methods

- Learn a value function directly from interaction with environment
- Can run in real-time, but not very sample efficient

Model-based Methods

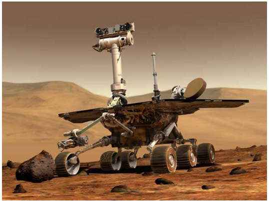
- Learn model of transition and reward dynamics
- Update value function using model (planning)
- Can update action-values without taking real actions in the world

Mortality



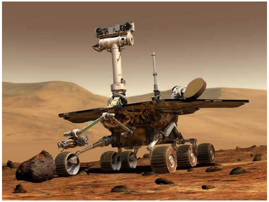
- Robot’s “lifetime” short compared to size of world

Mortality



- Robot's "lifetime" short compared to size of world
- (Still need to act in real time)

Mortality

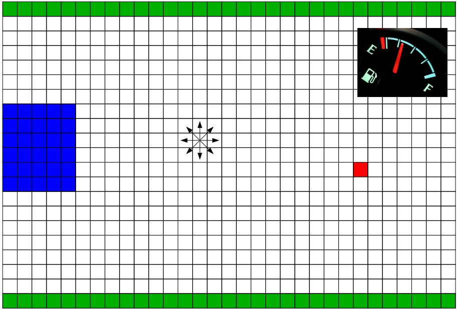


- Robot's "lifetime" short compared to size of world
- (Still need to act in real time)

Problem: Cannot explore everywhere

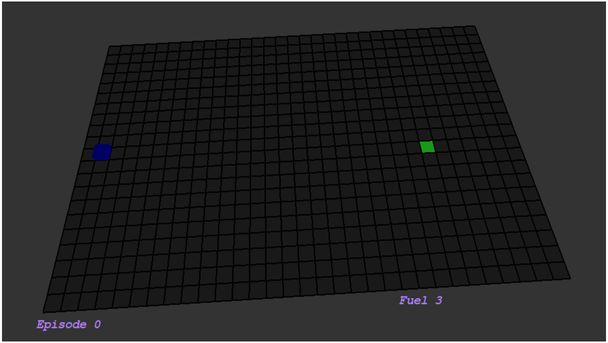
- Choose where not to explore
- Idea: Learn multiple possible models and compare them
- Only explore states that are both uncertain in model and promising for final policy

Fuel World



- Most of state space is very predictable
- But fuel stations have varying costs
- Want to explore mainly fuel stations, and particularly ones on short path to goal

Fuel World Behavior



- Agent explores randomly at first
- Agent focuses its exploration on fuel stations near the shortest path to the goal, trying a different fuel station each episode.
- Agent finds near-optimal policies.

Velocity Control: Real-Time Need

- Vehicle upgraded to run autonomously by adding shift-by-wire, steering, and braking actuators.
- 10 second episodes (at 20 Hz: 200 samples / episode)

Velocity Control

- State:
  - Current Velocity
  - Desired Velocity
  - Accelerator Pedal Position
  - Brake Pedal Position
- Actions:
  - Do nothing
  - Increase/decrease brake position by 0.1
  - Increase/decrease accelerator position by 0.1
- Reward: -10.0 * velocity error (m/s)

Desiderata

1. Algorithm must learn in very few actions (be **sample efficient**)
2. Algorithm must act **continually** in real-time (while learning)
3. Algorithm must handle **continuous** state
4. Algorithm must handle **delayed** actions

Desiderata

1. Algorithm must learn in very few actions (be **sample efficient**)
2. Algorithm must act **continually** in real-time (while learning)
3. Algorithm must handle **continuous** state
4. Algorithm must handle **delayed** actions

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-Max	Brafman 2001	Yes	No	No	No
Q-Learning	Watkins 1989	No	Yes	No	No
with F.A.	Sutton & Barto 1998	No	Yes	Yes	No
SARSA	Rummery & Niranjan 1994	No	Yes	No	No
GPRL	Deisenroth & Rasmussen 2011	Yes	No	Yes	No
BOSS	Asmuth et al 2009	Yes	No	No	No
Bayesian DP	Strens 2000	Yes	No	No	No
MBBE	Dearden et al 1999	Yes	No	No	No
MBS	Walsh et al 2009	Yes	No	No	Yes
Dyna	Sutton 1990	No	Yes	No	No

Common Approaches

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-Max	Brafman 2001	Yes	No	No	No
Q-Learning	Watkins 1989	No	Yes	No	No
with F.A.	Sutton & Barto 1998	No	Yes	Yes	No
SARSA	Rummery & Niranjan 1994	No	Yes	No	No
GPRL	Deisenroth & Rasmussen 2011	Yes	No	Yes	No
BOSS	Asmuth et al 2009	Yes	No	No	No
Bayesian DP	Strens 2000	Yes	No	No	No
MBBE	Dearden et al 1999	Yes	No	No	No
MBS	Walsh et al 2009	Yes	No	No	Yes
Dyna	Sutton 1990	No	Yes	No	No

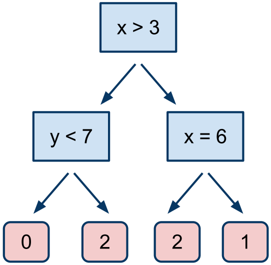
The TEXPLORE Algorithm

1. Limits exploration to be sample efficient
2. Selects actions continually in real-time
3. Handles continuous state
4. Handles actuator delays

Available publicly as a ROS package:

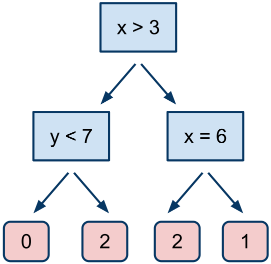
www.ros.org/wiki/rl-texplore-ros-pkg

Challenge 1: Sample Efficiency



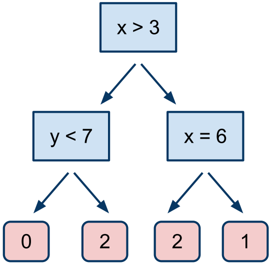
Challenge 1: Sample Efficiency

- Treat model learning as a supervised learning problem
  - Input: State and Action
  - Output: Distribution over next states and reward



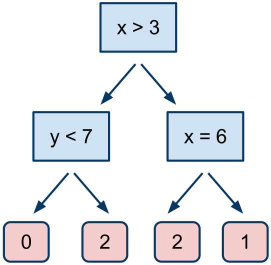
Challenge 1: Sample Efficiency

- Treat model learning as a supervised learning problem
  - Input: State and Action
  - Output: Distribution over next states and reward
- Factored model: Learn a separate model to predict each next state feature and reward



Challenge 1: Sample Efficiency

- Treat model learning as a supervised learning problem
  - Input: State and Action
  - Output: Distribution over next states and reward
- Factored model: Learn a separate model to predict each next state feature and reward
- Decision Trees: Split state space into regions with similar dynamics



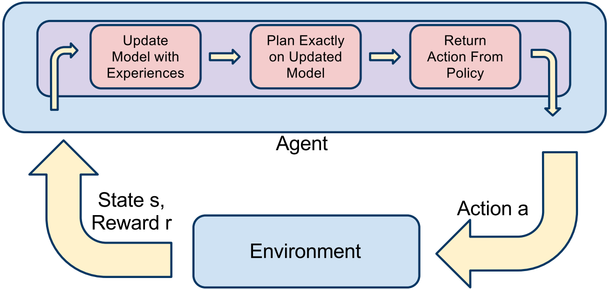
Random Forest Model

- Average predictions of m different decision trees
- Each tree represents a hypothesis of the true dynamics of the domain

Random Forest Model

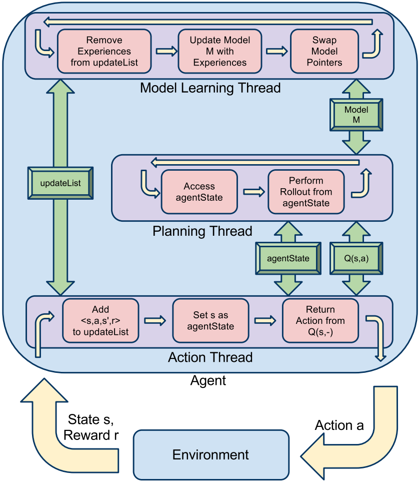
- Average predictions of m different decision trees
- Each tree represents a hypothesis of the true dynamics of the domain
- Acting greedily w.r.t. the average model balances predictions of optimistic and pessimistic models
- Limits the agent's exploration to state-actions that appear promising, while avoiding those which may have negative outcomes

Challenge 2: Real-Time Action Selection



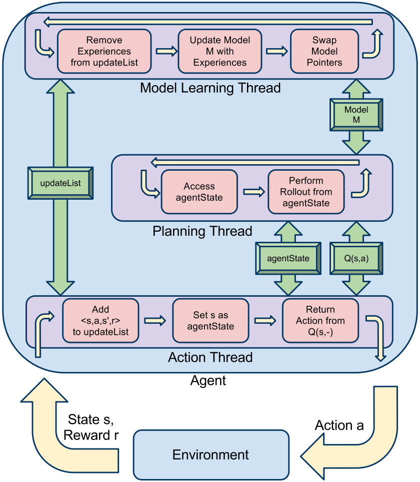
- Model update can take too long
- Planning can take too long

Real-Time Model Based Architecture (RTMBA)



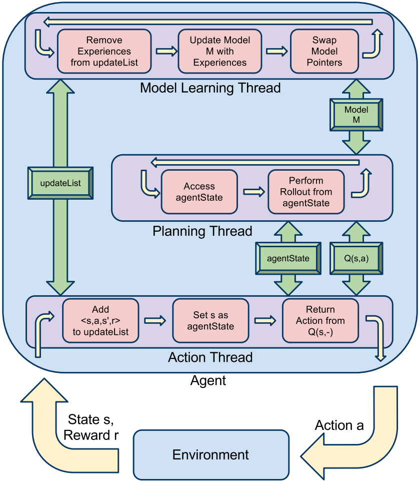
- Model learning and planning on parallel threads

Real-Time Model Based Architecture (RTMBA)



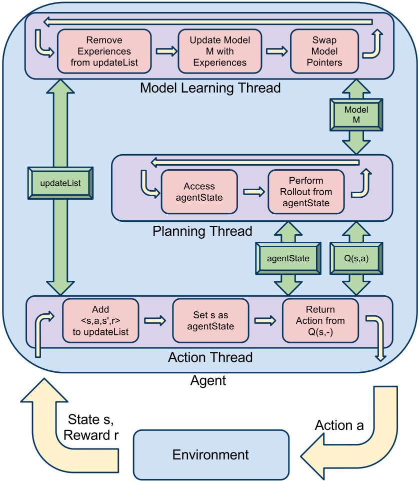
- Model learning and planning on parallel threads
- Action selection is not restricted by their computation time

Real-Time Model Based Architecture (RTMBA)



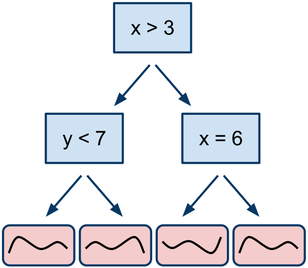
- Model learning and planning on parallel threads
- Action selection is not restricted by their computation time
- Use sample-based planning (anytime)

Real-Time Model Based Architecture (RTMBA)



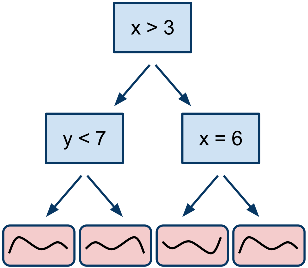
- Model learning and planning on parallel threads
- Action selection is not restricted by their computation time
- Use sample-based planning (anytime)
- Mutex locks on shared data

Challenge 3: Continuous State



- Use regression trees to model continuous state
- Each tree has a linear regression model at its leaves

Challenge 3: Continuous State



- Use regression trees to model continuous state
- Each tree has a linear regression model at its leaves
- Discretize state space for value updates from UCT, but still plan over continuously valued states

Challenge 4: Actuator Delays

- Delays make domain non-Markov, but k-Markov

Challenge 4: Actuator Delays

- Delays make domain non-Markov, but k-Markov
- Provide model with previous *k* actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant

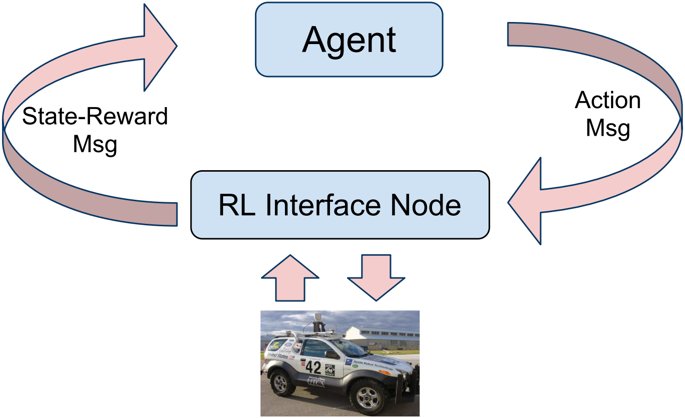
Challenge 4: Actuator Delays

- Delays make domain non-Markov, but k-Markov
- Provide model with previous *k* actions (Similar to U-Tree [McCallum 1996])
- Trees can learn which delayed actions are relevant
- UCT can plan over augmented state-action histories easily

Autonomous Vehicle

- Upgraded to run autonomously by adding shift-by-wire, steering, and braking actuators.
- Vehicle runs at 20 Hz.
- Agent must provide commands at this frequency.

Uses ROS [Quigley et al 2009]



- http://www.ros.org/wiki/rl_msgs

Simulation Experiments

Exploration Approaches

- Epsilon-Greedy
- Boltzmann Exploration
- Use merged BOSS-like model
- Use random model each episode

Sample Efficient Methods

- BOSS [Asmuth et al 2009]
- Bayesian DP [Strens 2000]
- Gaussian Process RL [Deisenroth & Rasmussen 2011]

Simulation Experiments

Continuous Models

- Tabular Models
- Gaussian Process RL [Deisenroth & Rasmussen 2011]
- KWIK linear regression [Strehl & Littman 2007]

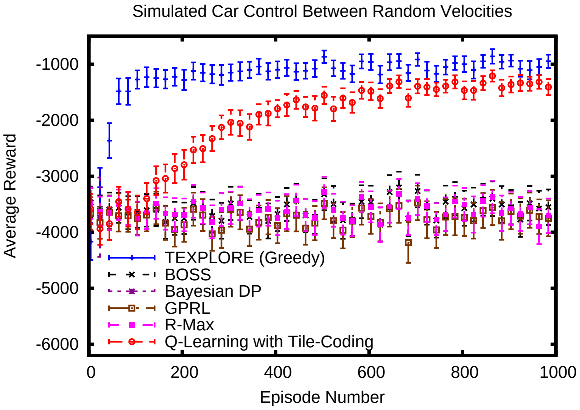
Real-Time Architectures

- Real Time Dynamic Programming [Barto et al 1995]
- Dyna [Sutton 1990]
- Parallel Value Iteration

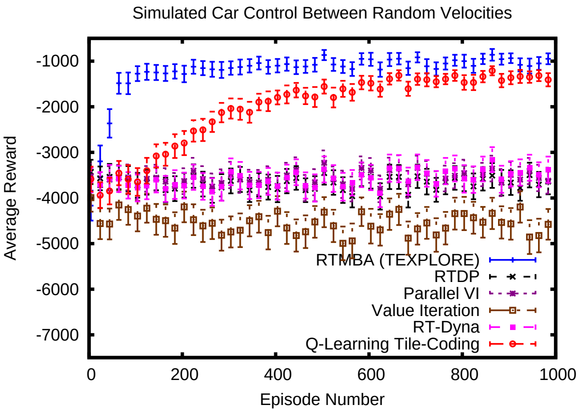
Actuator Delays

- Model Based Simulation [Walsh et al 2009]

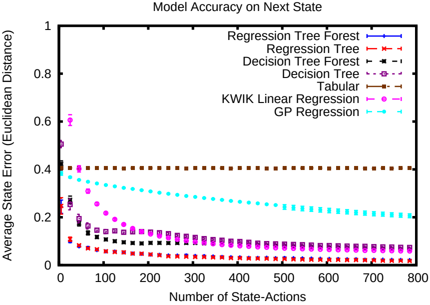
Challenge 1: Sample Efficiency

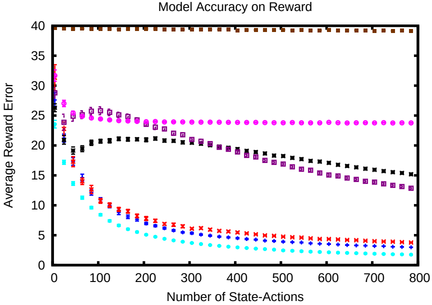


Challenge 2: Real-Time Action Selection

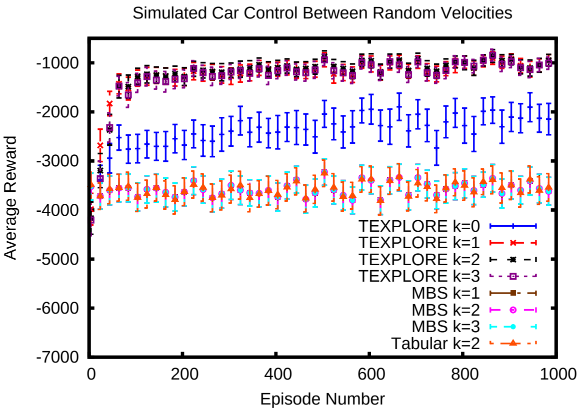


Challenge 3: Modeling Continuous Domains





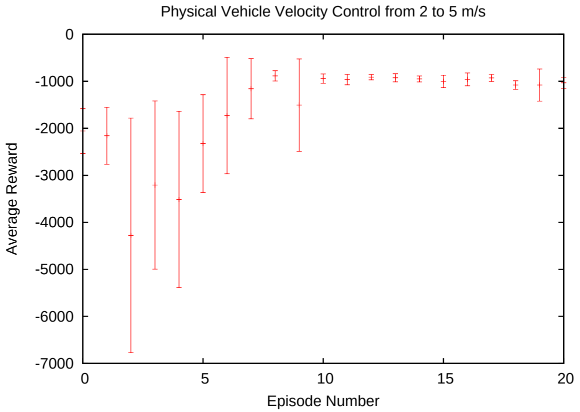
Challenge 4: Handling Delayed Actions



On the physical vehicle

- But, does it work on the actual vehicle?

On the physical vehicle

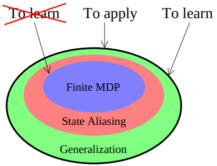


- Yes! It learns the task within 2 minutes of driving time

TEXPLORE Summary

- TEXPLORE can:
  1. Learn in few samples
  2. Act continually in real-time
  3. Learn in continuous domains
  4. Handle actuator delays
- TEXPLORE code has been released as a ROS package: www.ros.org/wiki/rl-texplore-ros-pkg

Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

UT Austin Villa 2014
RoboCup 3D Simulation League Champion via Overlapping Layered Learning

Patrick MacAlpine, Mike Depinet, and Peter Stone

AAAI, 2015

Layered Learning

- For domains too complex for tractably mapping state features $S \longmapsto$ outputs O
- Hierarchical subtask decomposition given: $\{L_1, L_2, \ldots, L_n\}$

Layered Learning

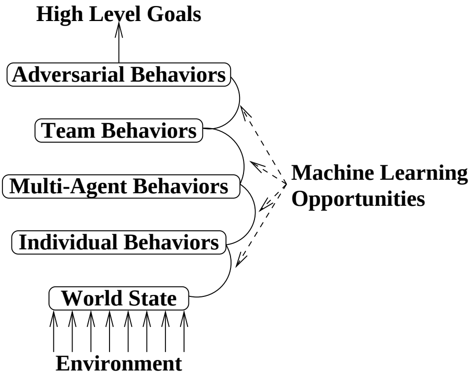
- For domains too complex for tractably mapping state features S $\longmapsto$ outputs O
- Hierarchical subtask decomposition given: $\{L_1, L_2, \ldots, L_n\}$
- Machine learning: exploit data to train, adapt

Layered Learning

- For domains too complex for tractably mapping state features $S \longmapsto$ outputs O
- Hierarchical subtask decomposition given: $\{L_1, L_2, \ldots, L_n\}$
- Machine learning: exploit data to train, adapt
- **Synthesis:** Learning in one layer feeds into next layer

Layered Learning

- For domains too complex for tractably mapping state features S $\longmapsto$ outputs O
- Hierarchical subtask decomposition given: $\{L_1, L_2, \ldots, L_n\}$
- Machine learning: exploit data to train, adapt
- **Synthesis:** Learning in one layer feeds into next layer



Layered Learning in Practice

First applied in simulated robot soccer [Stone & Veloso, ’97]

	Strategic Level	Example
L_1	individual	ball interception
L_2	multiagent	pass evaluation
L_3	**team**	pass selection

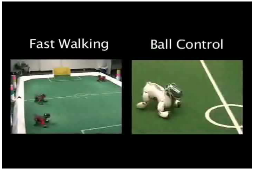
Layered Learning in Practice

First applied in simulated robot soccer [Stone & Veloso, ’97]

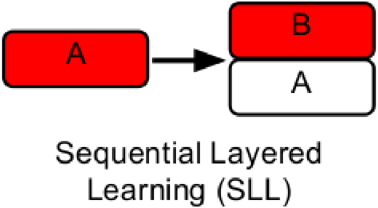
	Strategic Level	Example
L_1	individual	ball interception
L_2	multiagent	pass evaluation
L_3	**team**	pass selection

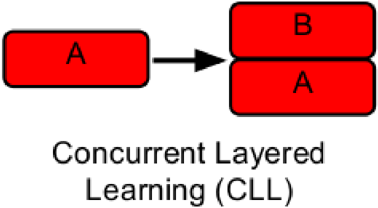
Later applied on real robots [Stone, Kohl, & Fidelman, ’06]

	Strategic Level	Example
L_1	individual	fast walking
L_2	individual	ball control



Layered Learning Paradigms



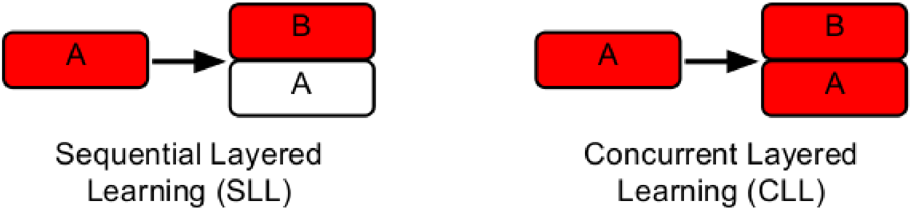


DESCRIPTIONS:

Sequential Layered Learning: Freeze parameters of layer after learning before learning of the next layer

Concurrent Layered Learning: Keep parameters of layer open during learning of the next layer

Layered Learning Paradigms

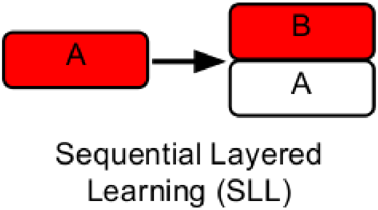


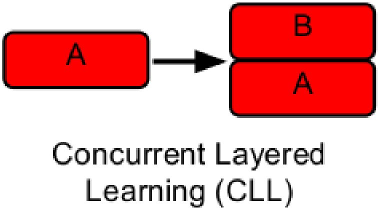
PROBLEMS:

Sequential Layered Learning: Can be too limiting in the joint layer policy seach space

Concurrent Layered Learning: The increased dimensionality can make learning harder or intractible

Layered Learning Paradigms



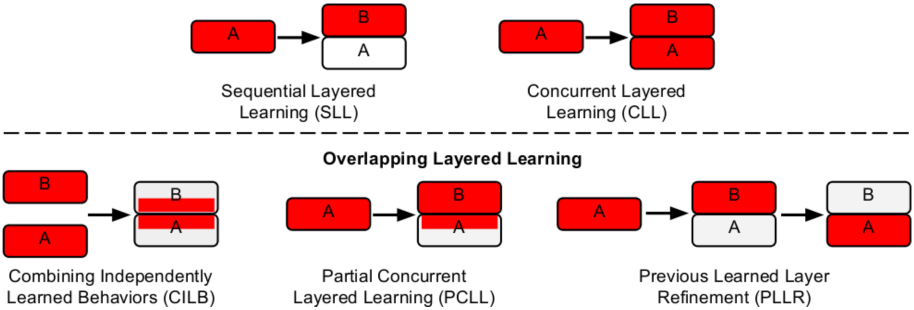


SOLUTION:

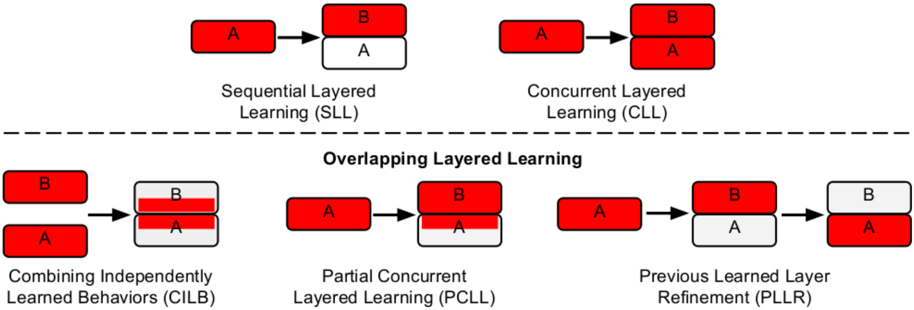
Overlapping Layered Learning: Tradeoff between freezing or keeping open previous learned layers

Optimizes "seam" or overlap between behaviors: keeps some parts of previously learned layers open during subsequent learning

Overlapping Layered Learning

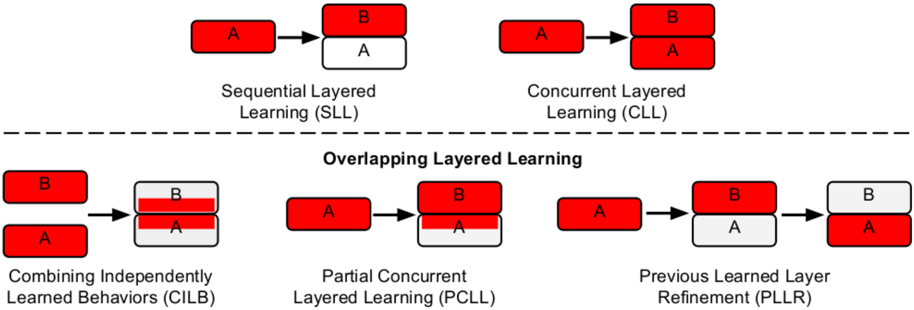


Overlapping Layered Learning



Combining Independently Learned Behaviors: Behaviors learned indpendently and then combined by relearning subset of behaviors' parameters

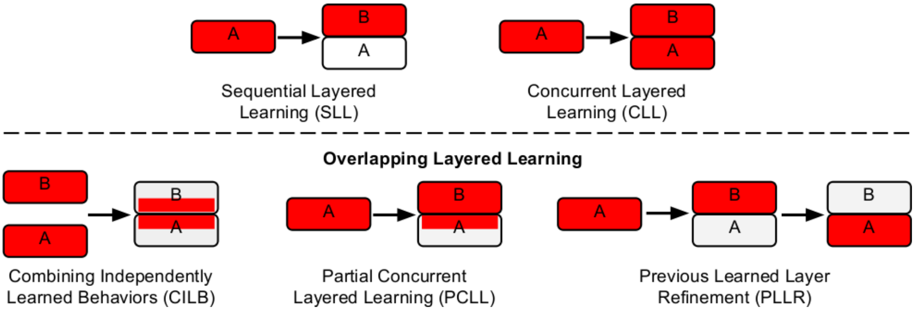
Overlapping Layered Learning



Combining Independently Learned Behaviors: Behaviors learned indpendently and then combined by relearning subset of behaviors' parameters

Partial Concurrent Layered Learning: Part, but not all, of a previously learned layer's behaviors are left open

Overlapping Layered Learning



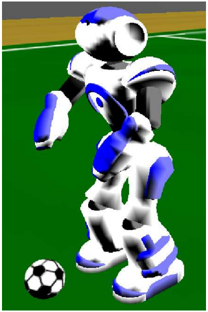
Combining Independently Learned Behaviors: Behaviors learned indpendently and then combined by relearning subset of behaviors' parameters

Partial Concurrent Layered Learning: Part, but not all, of a previously learned layer's behaviors are left open

Previous Learned Layer Refinement: After a pair of layers is learned, part or all of the initial layer is unfrozen

RoboCup 3D Simulation Domain

- Teams of 11 vs 11 autonomous robots play soccer
- Realistic physics using Open Dynamics Engine (ODE)
- Simulated robots modeled after Aldebaran Nao robot
- Robot receives noisy visual information about environment
- Robots can communicate over limited bandwidth channel



RoboCup Champions 2011, 2012

RoboCup Champions 2011, 2012

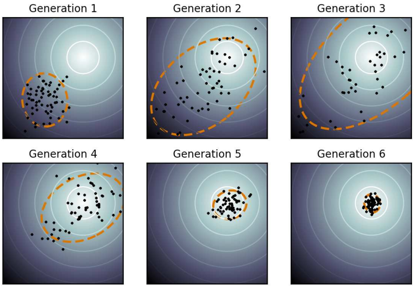
Humanoid Walk Learning via Layered Learning and CMA-ES

- Parameterized **double linear inverted pendulum model**

RoboCup Champions 2011, 2012

Humanoid Walk Learning via Layered Learning and CMA-ES

- Parameterized **double linear inverted pendulum model**



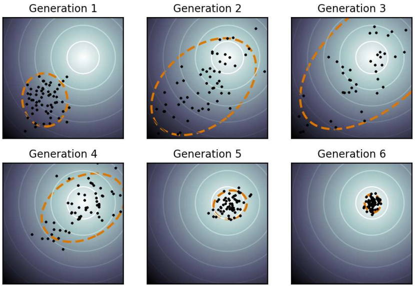
CMA-ES
[Hansen, '09]

- Stochastic, derivative-free, numerical optimization method
- Candidates sampled from multidimensional Gaussian

RoboCup Champions 2011, 2012

Humanoid Walk Learning via Layered Learning and CMA-ES

- Parameterized **double linear inverted pendulum model**



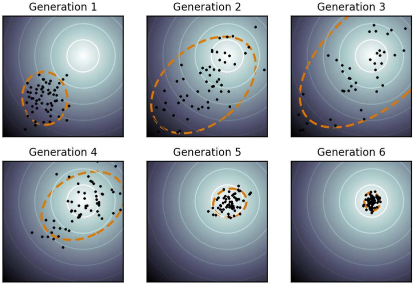
CMA-ES
[Hansen, '09]

- Stochastic, derivative-free, numerical optimization method
- Candidates sampled from multidimensional Gaussian
  - Mean maximizes likelihood of previous successes
  - Covariance update controls search step sizes

RoboCup Champions 2011, 2012

Humanoid Walk Learning via Layered Learning and CMA-ES

- Parameterized **double linear inverted pendulum model**

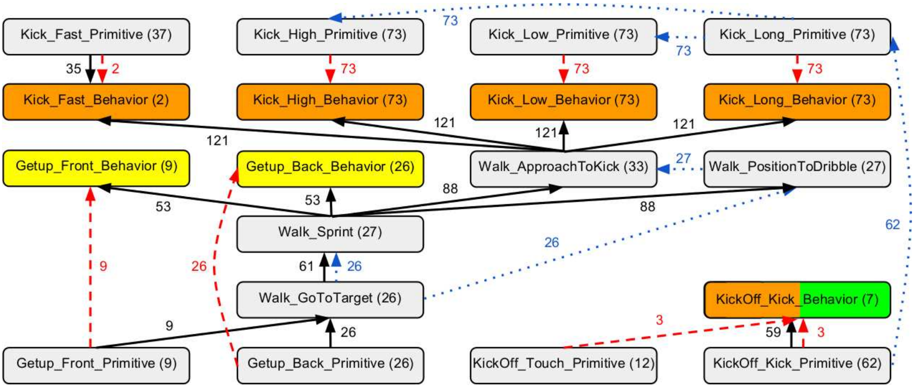


CMA-ES
[Hansen, ’09]

- Stochastic, derivative-free, numerical optimization method
- Candidates sampled from multidimensional Gaussian
  - Mean maximizes likelihood of previous successes
  - Covariance update controls search step sizes

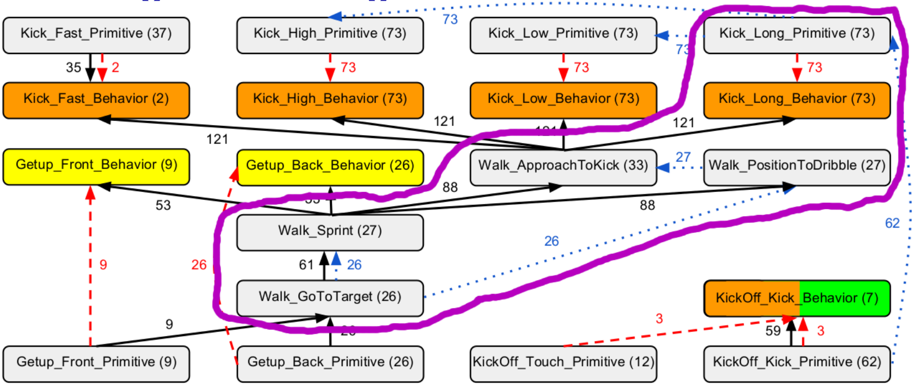
Initial walk **No layered learning** **2 layers**
3 layers **Final walk** **Champs*2**

Learned Layers



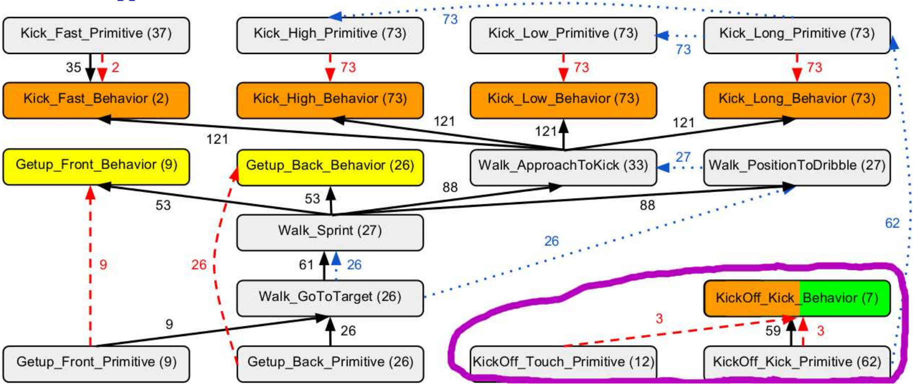
- 19 learned behaviors for standing up, walking, and kicking
  - CILB, PCLL, PLLR
- Over 500 parameters optimized during the course of learning
  - frozen, passed, seeded

Dribbling and Kicking the Ball in the Goal



- Four different walk parameter sets
  - Target/sprint/position + **approach ball to kick**
- Learn **fixed kick**
- Combine **kick with walk**: combine independent layers (CILB)
  - Overlap kick parameters for positioning
- Final **walk and kick**

Scoring on a Kickoff



- Kickoffs indirect (2 players must touch to score)
- Learn **fixed kick**
- Learn **touch** behavior **interferes**
- Combine **kick with touch**
  - Relearn position patterns: combine independent layers (CILB)
  - Learn new timing parameter: partial concurrent (PCLL)

Impact of Overlapping Layered Learning

1000 games vs. top 3 teams from 2013

Impact of Overlapping Layered Learning

1000 games vs. top 3 teams from 2013

	Average Goal Difference		
Opponent	**Full Team**	**No Kickoff**	**Dribble Only**
apollo3d	2.703 (0.041)	2.062 (0.038)	1.861 (0.034)
UTAustinVilla2013	1.589 (0.036)	1.225 (0.033)	0.849 (0.025)
fcportugal3d	3.991 (0.051)	3.189 (0.048)	1.584 (0.030)

No Kickoff: On kickoff, kick ball deep into opponent's end

Dribble Only: No kicking

Repetition on Different Robot Types

Type 0: Standard Nao model

Type 1: Longer legs and arms

Type 2: Quicker moving legs

Type 3: Wider hips and longest legs and arms

Type 4: Added toes to foot

Repetition on Different Robot Types

Type 0: Standard Nao model

Type 1: Longer legs and arms

Type 2: Quicker moving legs

Type 3: Wider hips and longest legs and arms

Type 4: Added toes to foot

	Avg. Goal Difference per Robot Type				
Opponent	**Type 0**	**Type 1**	**Type 2**	**Type 3**	**Type 4**
apollo3d	1.787	1.819	1.820	1.543	2.827
UTAustinVilla2013	0.992	0.892	1.276	0.573	1.141
fcportugal3d	2.423	3.025	3.275	2.678	4.033

Repetition on Different Robot Types

Type 0: Standard Nao model

Type 1: Longer legs and arms

Type 2: Quicker moving legs

Type 3: Wider hips and longest legs and arms

Type 4: Added toes to foot

	Avg. Goal Difference per Robot Type				
Opponent	**Type 0**	**Type 1**	**Type 2**	**Type 3**	**Type 4**
apollo3d	1.787	1.819	1.820	1.543	2.827
UTAustinVilla2013	0.992	0.892	1.276	0.573	1.141
fcportugal3d	2.423	3.025	3.275	2.678	4.033

Computation per type

$\approx$ 700k parameter sets evaluated

$\approx$ 1.5 years compute time ($\approx$ 50 hours on condor cluster)

RoboCup 2014

Won competition with undefeated record: outscored opps 52–0

Opponent	Avg. Goal Diff.	Record (W-L-T)	Goals (F/A)	KO Score %
BahiaRT	2.075 (0.030)	990-0-10	2092/17	96.2
FCPortugal	2.642 (0.034)	986-0-14	2748/106	83.4
magmaOffenburg	2.855 (0.035)	990-0-10	2864/9	88.3
RoboCanes	3.081 (0.046)	974-0-26	3155/74	69.4
FUT-K	3.236 (0.039)	998-0-2	3240/4	96.3
SEU_Jolly	4.031 (0.062)	995-0-5	4034/3	87.6
KarachiKoalas	5.681 (0.046)	1000-0-0	5682/1	87.5
ODENS	7.933 (0.041)	1000-0-0	7933/0	92.1
HfutEngine	8.510 (0.050)	1000-0-0	8510/0	94.7
Mithras3D	8.897 (0.041)	1000-0-0	8897/0	90.4
L3M-SIM	9.304 (0.043)	1000-0-0	9304/0	93.7

RoboCup 2014

Won competition with undefeated record: outscored opps 52–0

Opponent	Avg. Goal Diff.	Record (W-L-T)	Goals (F/A)	KO Score %
BahiaRT	2.075 (0.030)	990-0-10	2092/17	96.2
FCPortugal	2.642 (0.034)	986-0-14	2748/106	83.4
magmaOffenburg	2.855 (0.035)	990-0-10	2864/9	88.3
RoboCanes	3.081 (0.046)	974-0-26	3155/74	69.4
FUT-K	3.236 (0.039)	998-0-2	3240/4	96.3
SEU_Jolly	4.031 (0.062)	995-0-5	4034/3	87.6
KarachiKoalas	5.681 (0.046)	1000-0-0	5682/1	87.5
ODENS	7.933 (0.041)	1000-0-0	7933/0	92.1
HfutEngine	8.510 (0.050)	1000-0-0	8510/0	94.7
Mithras3D	8.897 (0.041)	1000-0-0	8897/0	90.4
L3M-SIM	9.304 (0.043)	1000-0-0	9304/0	93.7

- After: 11,000 games: won all by 67 (no losses)

RoboCup 2014

Won competition with undefeated record: outscored opps 52–0

Opponent	Avg. Goal Diff.	Record (W-L-T)	Goals (F/A)	KO Score %
BahiaRT	2.075 (0.030)	990-0-10	2092/17	96.2
FCPortugal	2.642 (0.034)	986-0-14	2748/106	83.4
magmaOffenburg	2.855 (0.035)	990-0-10	2864/9	88.3
RoboCanes	3.081 (0.046)	974-0-26	3155/74	69.4
FUT-K	3.236 (0.039)	998-0-2	3240/4	96.3
SEU_Jolly	4.031 (0.062)	995-0-5	4034/3	87.6
KarachiKoalas	5.681 (0.046)	1000-0-0	5682/1	87.5
ODENS	7.933 (0.041)	1000-0-0	7933/0	92.1
HfutEngine	8.510 (0.050)	1000-0-0	8510/0	94.7
Mithras3D	8.897 (0.041)	1000-0-0	8897/0	90.4
L3M-SIM	9.304 (0.043)	1000-0-0	9304/0	93.7

- After: 11,000 games: won all by 67 (no losses)
- **Highlights** from Final vs. RoboCanes (University of Miami)

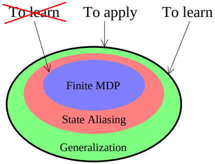
RoboCup 2014

Won competition with undefeated record: outscored opps 52–0

Opponent	Avg. Goal Diff.	Record (W-L-T)	Goals (F/A)	KO Score %
BahiaRT	2.075 (0.030)	990-0-10	2092/17	96.2
FCPortugal	2.642 (0.034)	986-0-14	2748/106	83.4
magmaOffenburg	2.855 (0.035)	990-0-10	2864/9	88.3
RoboCanes	3.081 (0.046)	974-0-26	3155/74	69.4
FUT-K	3.236 (0.039)	998-0-2	3240/4	96.3
SEU_Jolly	4.031 (0.062)	995-0-5	4034/3	87.6
KarachiKoalas	5.681 (0.046)	1000-0-0	5682/1	87.5
ODENS	7.933 (0.041)	1000-0-0	7933/0	92.1
HfutEngine	8.510 (0.050)	1000-0-0	8510/0	94.7
Mithras3D	8.897 (0.041)	1000-0-0	8897/0	90.4
L3M-SIM	9.304 (0.043)	1000-0-0	9304/0	93.7

- After: 11,000 games: won all by 67 (no losses)
- **Highlights** from Final vs. RoboCanes (University of Miami)
- More info: www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/

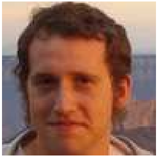
Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLES [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

Making Friends on the Fly: Advances in Ad Hoc Teamwork

Samuel Barrett, Katie Genter, and Peter Stone



AAAI, 2015; **AAMAS**, 2015

Ad Hoc Teamwork [Stone et al., AIJ 2013]

- Only in control of a single agent or subset of agents
- Unknown teammates
- Shared goals
- No pre-coordination

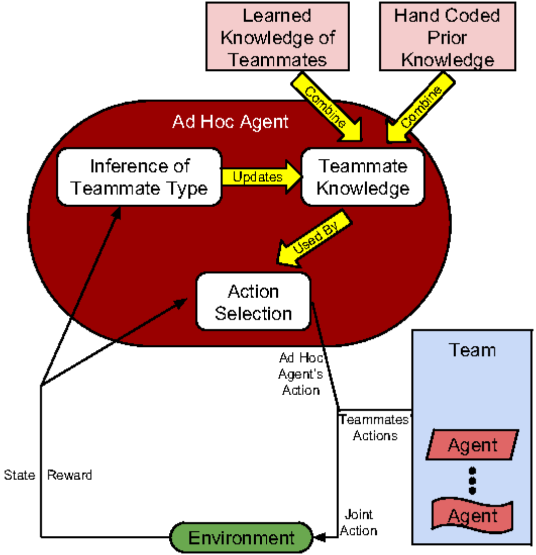
Ad Hoc Teamwork [Stone et al., AIJ 2013]

- Only in control of a single agent or subset of agents
- Unknown teammates
- Shared goals
- No pre-coordination

Examples in humans:

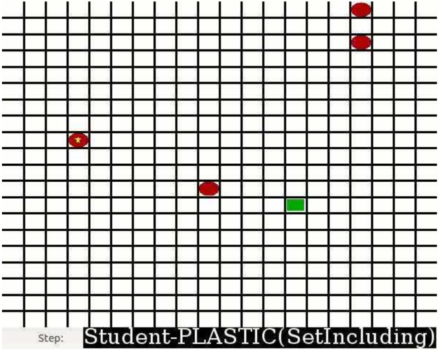
- Pick up soccer
- Accident response

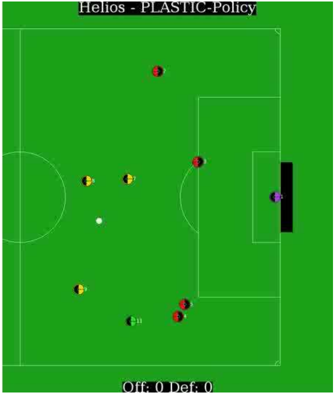
PLASTIC: Planning and Learning to Adapt Swiftly to Teammates to Improve Cooperation



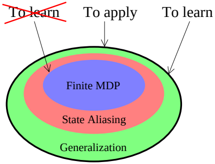
Testbed Domains

- Agent replaces single teammate in otherwise coherent team
- Adapts based on knowledge learned from previous teammates





Practical RL



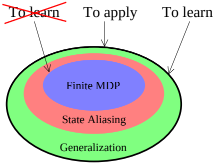
- **R**epresentation

- **I**nteraction

- **S**ynthesis

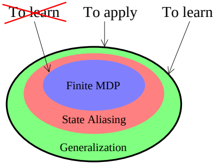
- **M**ortality

Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]

Practical RL



- **R**epresentation
  - Selecting the Algorithm: parameterized domains [K.&S., MLJ 2011]
  - Adapting Representation: NEAT+Q [Whiteson & S., JMLR 2006]
- **I**nteraction
  - With adversaries: CMLeS [Chakraborty & S., ICML 2010]
  - With ad hoc teammates: PLASTIC [Barrett, thesis 2014]
  - With people: TAMER [Knox & S., AAMAS 2010]
- **S**ynthesis
  - Of Algorithms: Layered Learning [S., MIT Press 2000]
  - Of Concepts: Fitted R-MAXQ [Jong & S., ECML 2009]
- **M**ortality
  - Leverage the Past: Transfer Learning [Taylor, S., & Liu, JMLR 2007]
  - Acknowledge a Finite Future: TEXPLORE [Hester & S., MLJ 2013]