CS 747, Autumn 2020: Week 11, Lecture 1

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2020

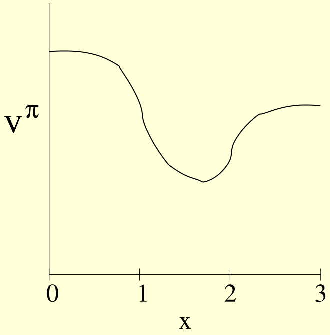
Reinforcement Learning

1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
   - Humanoid robot soccer
   - Railway scheduling

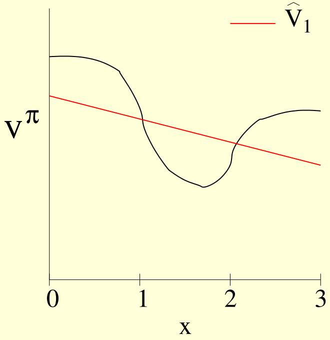
Reinforcement Learning

1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
    - Humanoid robot soccer
    - Railway scheduling

How Good is Linear Function Approximation?

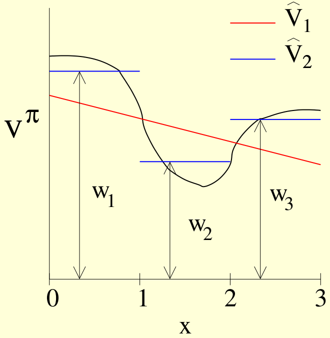


How Good is Linear Function Approximation?



$$\hat{V}_1(x) = w_1 x + w_2.$$

How Good is Linear Function Approximation?



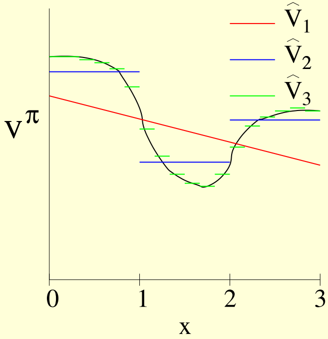
$$b_1 = \begin{cases} 1 & \text{if } 0 \leq x < 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_2 = \begin{cases} 1 & \text{if } 1 \leq x < 2, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_3 = \begin{cases} 1 & \text{if } 2 \leq x < 3, \\ 0 & \text{otherwise.} \end{cases}$$

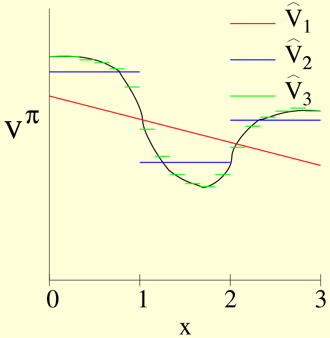
$$\hat{V}_2(x) = w_1 b_1 + w_2 b_2 + w_3 b_3.$$

How Good is Linear Function Approximation?



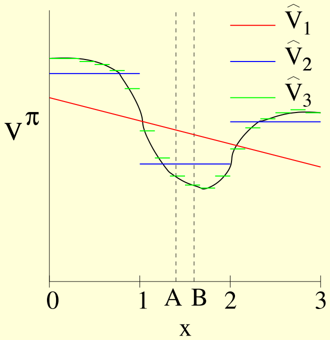
$\hat{V}_3(x)$: 18 piece-wise constants.

How Good is Linear Function Approximation?



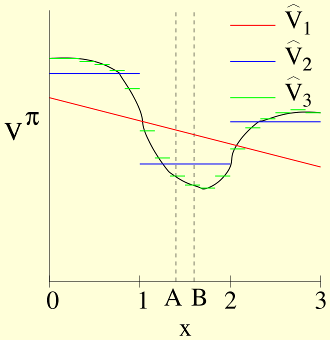
- Is $\hat{V}^3$ the obvious choice?

How Good is Linear Function Approximation?



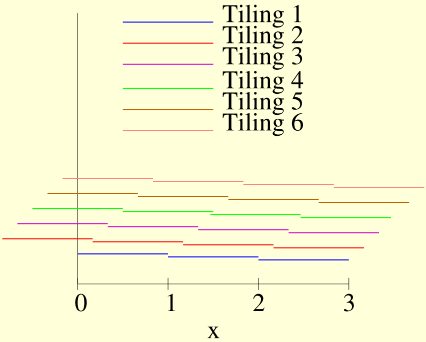
- Is $\hat{V}^3$ the obvious choice?
- $\hat{V}^3$ has the highest resolution, but does not generalise well.

How Good is Linear Function Approximation?



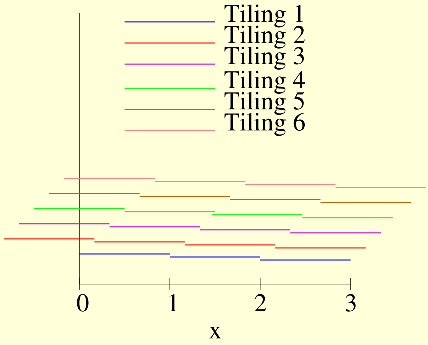
- Is $\hat{V}^3$ the obvious choice?
- $\hat{V}^3$ has the highest resolution, but does not generalise well.
- How to achieve high resolution along with generalisation?

Tile coding



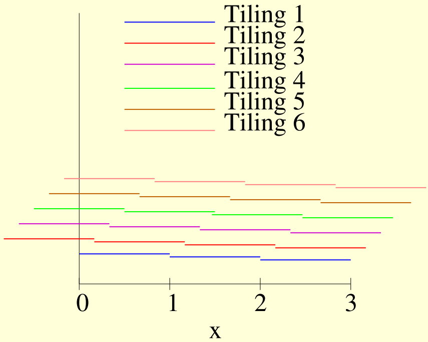
- A tiling partitions x into equal-width regions called tiles.

Tile coding



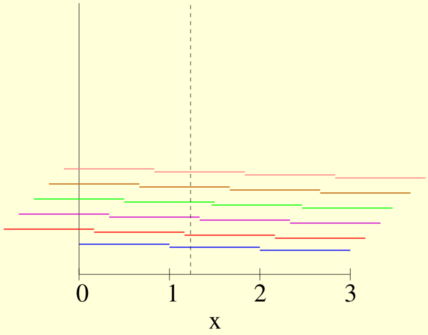
- A tiling partitions x into equal-width regions called tiles.
- Multiple tilings (say m) are created, each with an offset ($1/m$ tile width) from the previous.

Tile coding



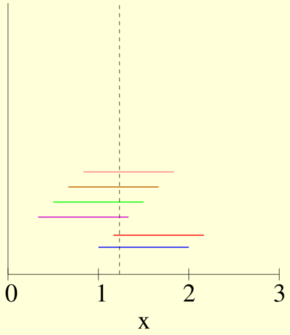
- A tiling partitions x into equal-width regions called tiles.
- Multiple tilings (say m) are created, each with an offset ($1/m$ tile width) from the previous.
- Each tile has an associated weight.

Tile coding



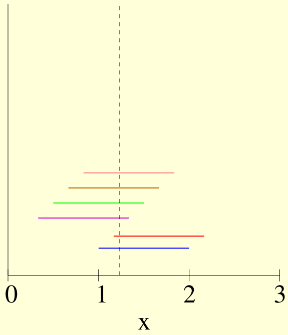
- A tiling partitions x into equal-width regions called tiles.
- Multiple tilings (say m) are created, each with an offset ($1/m$ tile width) from the previous.
- Each tile has an associated weight.
- The function value of a point is the sum of the weights of the tiles intersecting it (one per tiling).

Tile coding



- A tiling partitions x into equal-width regions called tiles.
- Multiple tilings (say m) are created, each with an offset ($1/m$ tile width) from the previous.
- Each tile has an associated weight.
- The function value of a point is the sum of the weights of the tiles intersecting it (one per tiling).

Tile coding



- Each tile is a binary feature.
- Tile width and the number of tilings determine generalisation, resolution.
- Observe that two points more than (tile width / number of tilings) apart can be given arbitrary function values.

Representing $\hat{Q}$

- Given a feature value x as input, the corresponding set of tilings $T : \mathbb{R} \to \mathbb{R}$ returns the sum of the weights of the tiles activated by x.

Representing $\hat{Q}$

- Given a feature value x as input, the corresponding set of tilings $T : \mathbb{R} \to \mathbb{R}$ returns the sum of the weights of the tiles activated by x.
- The usual practice is to have a separate set of tilings $T_{aj} : \mathbb{R} \to \mathbb{R}$ for each action a and state feature $j \in \{1, 2, \ldots, d\}$. Hence

$$\hat{Q}(s, a) = \sum_{j=1}^{d} T_{aj}(x_j(s)).$$

Representing $\hat{Q}$

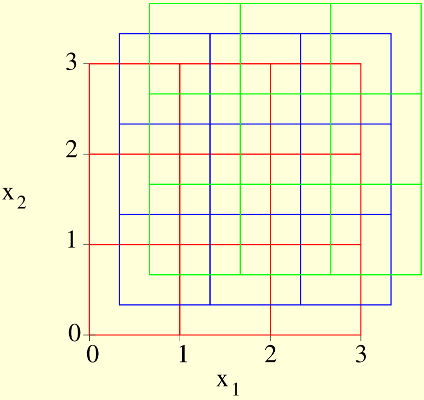
- Given a feature value x as input, the corresponding set of tilings $T : \mathbb{R} \to \mathbb{R}$ returns the sum of the weights of the tiles activated by x.
- The usual practice is to have a separate set of tilings $T_{aj} : \mathbb{R} \to \mathbb{R}$ for each action a and state feature $j \in \{1, 2, \ldots, d\}$. Hence

$$\hat{Q}(s, a) = \sum_{j=1}^{d} T_{aj}(x_j(s)).$$

- Usually, tile widths and the number of tilings are configured specifically for each feature. For example, in soccer, could use $2m$ as tile width for "distance" features, and 10° as tile width for "angle" features.

2-*d* Tile coding

- For representing more complex functions, can also have tilings on conjunctions of features (see below for 2 features).



- Introduces more parameters—which could help or hurt.

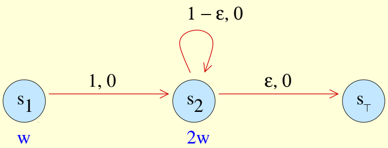
Tile Coding: Summary

- Linear function approximation does not restrict us to a representation that is linear in the given/raw features.
- Tile coding a standard approach to discretise input features and tune both resolution and generalisation.
- Enjoys many empirical successes, especially in conjunction with Linear Sarsa(λ).
- Common to store weights in a hash table (collisions don't seem to hurt much), whose size is set based on practical constraints.
- 1-d tilings most common; rarely see conjunction of 3 or more features.

Reinforcement Learning

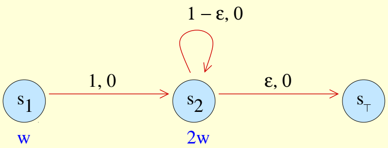
1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
   - Humanoid robot soccer
   - Railway scheduling

A Counterexample



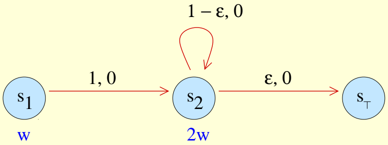
- Prediction problem (policy π).
- Episodic, start state is s_1.
- Observe that $V^{\pi}(s_1) = V^{\pi}(s_2) = 0$.
- Linear function approximation with single parameter w: $x(s_1) = 1, x(s_2) = 2$; hence $\hat{V}(s_1) = w$, $\hat{V}(s_2) = 2w$.

A Counterexample



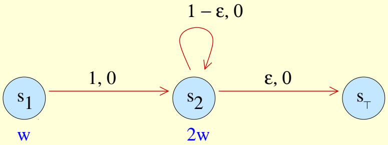
- Prediction problem (policy π).
- Episodic, start state is s_1.
- Observe that $V^\pi(s_1) = V^\pi(s_2) = 0$.
- Linear function approximation with single parameter w: $x(s_1) = 1, x(s_2) = 2$; hence $\hat{V}(s_1) = w$, $\hat{V}(s_2) = 2w$.
- What's the optimal setting of w?

A Counterexample



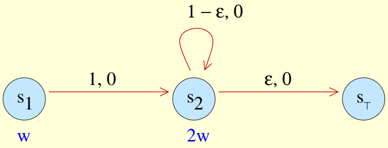
- Prediction problem (policy π).
- Episodic, start state is s_1.
- Observe that $V^\pi(s_1) = V^\pi(s_2) = 0$.
- Linear function approximation with single parameter w: $x(s_1) = 1, x(s_2) = 2$; hence $\hat{V}(s_1) = w$, $\hat{V}(s_2) = 2w$.
- What's the optimal setting of w?
- $w = 0$ gives the exact answer!

A Counterexample



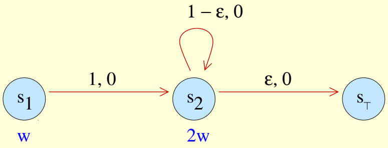
- Prediction problem (policy π).
- Episodic, start state is s_1.
- Observe that $V^\pi(s_1) = V^\pi(s_2) = 0$.
- Linear function approximation with single parameter w: $x(s_1) = 1, x(s_2) = 2$; hence $\hat{V}(s_1) = w$, $\hat{V}(s_2) = 2w$.
- What's the optimal setting of w?
- $w = 0$ gives the exact answer!
- We design an iteration $w_0 \to w_1 \to w_2 \to \ldots$, and see if it converges to 0 (due to Tsitsiklis and Van Roy, 1996).

A Counterexample



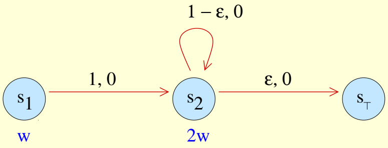
- From state s, let s', r be the (random) next state, reward.

A Counterexample



- From state s, let s', r be the (random) next state, reward.
- If our current estimate of V^{π} is $\hat{V}$, the bootstrapping idea suggests $\mathbb{E}_{\pi}[r + \gamma \hat{V}(s')]$ as a "better estimate" of $V^{\pi}(s)$.

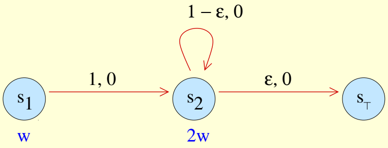
A Counterexample



- From state s, let s', r be the (random) next state, reward.
- If our current estimate of V^π is $\hat{V}$, the bootstrapping idea suggests $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$ as a "better estimate" of $V^\pi(s)$.
- We update w so it best-fits the bootstrapped estimate in terms of squared error on the states.
- We begin with arbitrary w_0, and for $k \geq 0$ set

$$w_{k+1} \leftarrow \underset{w \in \mathbb{R}}{\operatorname{argmin}} \sum_s \left(\mathbb{E}_\pi[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

A Counterexample

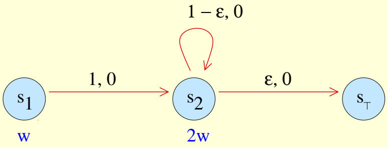


- From state s, let s', r be the (random) next state, reward.
- If our current estimate of V^{π} is $\hat{V}$, the bootstrapping idea suggests $\mathbb{E}_{\pi}[r + \gamma \hat{V}(s')]$ as a "better estimate" of $V^{\pi}(s)$.
- We update w so it best-fits the bootstrapped estimate in terms of squared error on the states.
- We begin with arbitrary w_0, and for $k \geq 0$ set

$$w_{k+1} \leftarrow \underset{w \in \mathbb{R}}{\operatorname{argmin}} \sum_{s} \left(\mathbb{E}_{\pi}[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

- Is $\lim_{k \to \infty} w_k = 0$?

A Counterexample

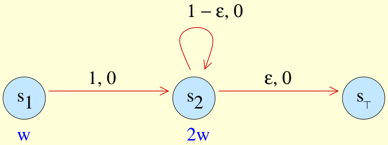


- From state s, let s', r be the (random) next state, reward.
- If our current estimate of V^{π} is $\hat{V}$, the bootstrapping idea suggests $\mathbb{E}_{\pi}[r + \gamma \hat{V}(s')]$ as a "better estimate" of $V^{\pi}(s)$.
- We update w so it best-fits the bootstrapped estimate in terms of squared error on the states.
- We begin with arbitrary w_0, and for $k \geq 0$ set

$$w_{k+1} \leftarrow \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{s} \left(\mathbb{E}_{\pi}[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

- Is $\lim_{k \to \infty} w_k = 0$? Let's see.

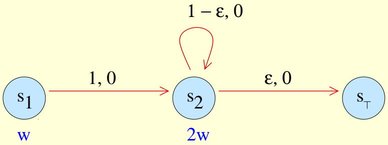
A Counterexample



$$w_{k+1} = \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{s} \left(\mathbb{E}_{\pi}[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2$$

$$= \operatorname*{argmin}_{w \in \mathbb{R}} \left((2\gamma w_k - w)^2 + (2\gamma(1-\epsilon) w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k.$$

A Counterexample

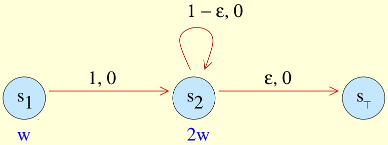


$$w_{k+1} = \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{s} \left(\mathbb{E}_\pi[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2$$

$$= \operatorname*{argmin}_{w \in \mathbb{R}} \left((2\gamma w_k - w)^2 + (2\gamma(1-\epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k.$$

- For $w_0 = 1$, $\epsilon = 0.1$, $\gamma = 0.99$, $\lim_{k\to\infty} w_k = \infty$; divergence!

A Counterexample

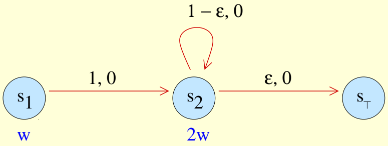


$$w_{k+1} = \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{s} \left(\mathbb{E}_{\pi}[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2$$

$$= \operatorname*{argmin}_{w \in \mathbb{R}} \left((2\gamma w_k - w)^2 + (2\gamma(1-\epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k.$$

- For $w_0 = 1$, $\epsilon = 0.1$, $\gamma = 0.99$, $\lim_{k \to \infty} w_k = \infty$; divergence!
- The failure owes to the combination of three factors: off-policy updating, generalisation, bootstrapping.

A Counterexample



$$w_{k+1} = \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{s} \left(\mathbb{E}_{\pi}[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2$$

$$= \operatorname*{argmin}_{w \in \mathbb{R}} \left((2\gamma w_k - w)^2 + (2\gamma(1-\epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k.$$

- For $w_0 = 1$, $\epsilon = 0.1$, $\gamma = 0.99$, $\lim_{k \to \infty} w_k = \infty$; divergence!
- The failure owes to the combination of three factors: off-policy updating, generalisation, bootstrapping.
- But these are almost always used together in practice!

Summary of Theoretical Results*

Method	Tabular	Linear FA	Non-linear FA
TD(0)	C, O	C	NK
TD(λ), $\lambda \in (0, 1)$	C, O	C	NK
TD(1)	C, O	C, “Best”	C, Local optimum
Sarsa(0)	C, 0	Chattering	NK
Sarsa(λ), $\lambda \in (0, 1)$	NK	Chattering	NK
Sarsa(1)	NK	NK	NK
Q-learning(0)	C, 0	NK	NK

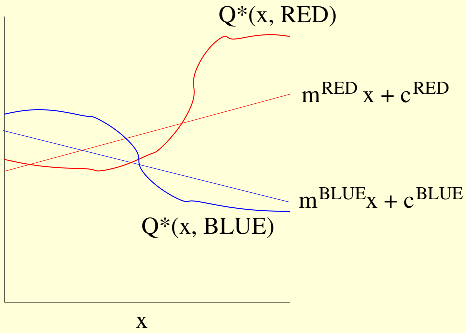
(C: Convergent; O: Optimal; NK: Not known.)

$\star$: to the best of your instructor's knowledge.

Reinforcement Learning

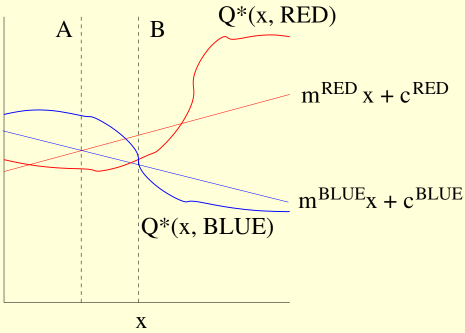
1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
   - Humanoid robot soccer
   - Railway scheduling

So Near, Yet So Far



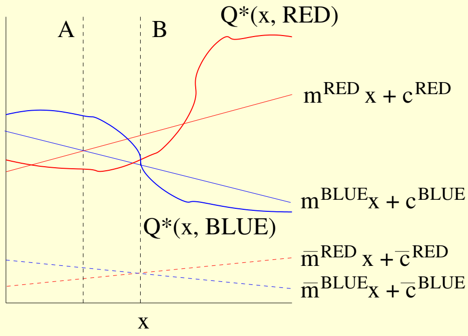
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$ a "good" approximation of $Q^\star$.

So Near, Yet So Far



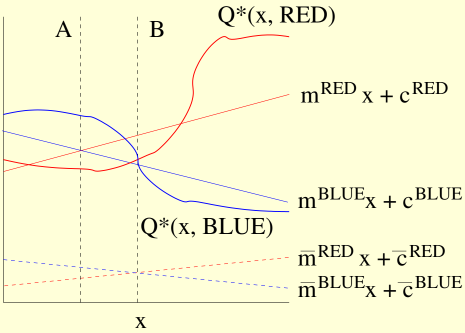
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$ a "good" approximation of $Q^\star$. But induces non-optimal actions for $x \in (A, B)$.

So Near, Yet So Far



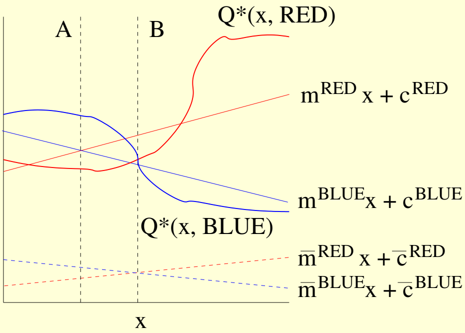
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$ a "good" approximation of $Q^\star$. But induces non-optimal actions for $x \in (A, B)$.
- $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$ a "bad" approximation of $Q^\star$. But induces optimal actions for all x!

So Near, Yet So Far



- $(m^{\mathrm{RED}}, c^{\mathrm{RED}}, m^{\mathrm{BLUE}}, c^{\mathrm{BLUE}})$ a "good" approximation of $Q^\star$. But induces non-optimal actions for $x \in (A, B)$.
- $(\bar{m}^{\mathrm{RED}}, \bar{c}^{\mathrm{RED}}, \bar{m}^{\mathrm{BLUE}}, \bar{c}^{\mathrm{BLUE}})$ a "bad" approximation of $Q^\star$. But induces optimal actions for all x!
- Perhaps we found $(m^{\mathrm{RED}}, c^{\mathrm{RED}}, m^{\mathrm{BLUE}}, c^{\mathrm{BLUE}})$ by Q-learning.

So Near, Yet So Far



- $(m^{\mathrm{RED}}, c^{\mathrm{RED}}, m^{\mathrm{BLUE}}, c^{\mathrm{BLUE}})$ a "good" approximation of $Q^\star$.
  But induces non-optimal actions for $x \in (A, B)$.
- $(\bar{m}^{\mathrm{RED}}, \bar{c}^{\mathrm{RED}}, \bar{m}^{\mathrm{BLUE}}, \bar{c}^{\mathrm{BLUE}})$ a "bad" approximation of $Q^\star$.
  But induces optimal actions for all x!
- Perhaps we found $(m^{\mathrm{RED}}, c^{\mathrm{RED}}, m^{\mathrm{BLUE}}, c^{\mathrm{BLUE}})$ by Q-learning.
- How to find $(\bar{m}^{\mathrm{RED}}, \bar{c}^{\mathrm{RED}}, \bar{m}^{\mathrm{BLUE}}, \bar{c}^{\mathrm{BLUE}})$?

Black Box Optimisation

- An abstract model:

  Input $w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow$ Output $f(w)$.

Black Box Optimisation

- An abstract model:

  Input $w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow$ Output $f(w)$.

  For what input w is output $f(w)$ maximum?

Black Box Optimisation

- An abstract model:
  Input $w = (w_1, w_2, \dots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow$ Output $f(w)$.
  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

Black Box Optimisation

- An abstract model:
  Input $w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow$ Output $f(w)$.
  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

- System: Your MDP!
  w: parameters defining your policy.
  $f(w)$: expected long-term reward (as a scalar).

Black Box Optimisation

- An abstract model:
  Input $w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow$ Output $f(w)$.
  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

- System: Your MDP!
  w: parameters defining your policy.
  $f(w)$: expected long-term reward (as a scalar).

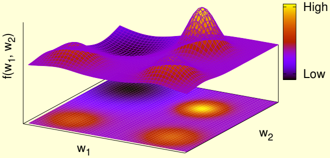
- Is finding the optimal w easy? Why is this approach called black box optimisation?

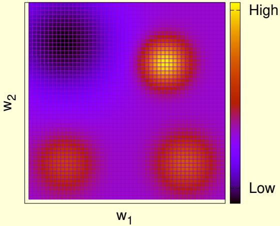
Typical Context for Black Box Optimisation

- Little/nothing known/assumed about f—can be discontinuous, non-linear, “erratic”.
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

Typical Context for Black Box Optimisation

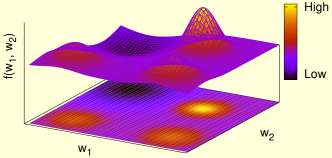
- Little/nothing known/assumed about f—can be discontinuous, non-linear, “erratic”.
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

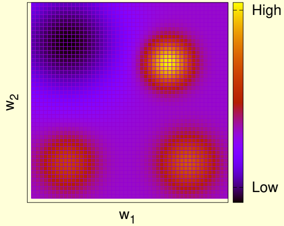




Typical Context for Black Box Optimisation

- Little/nothing known/assumed about f—can be discontinuous, non-linear, "erratic".
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

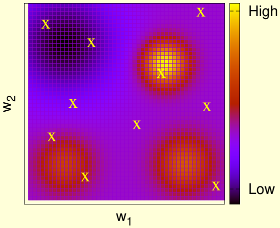




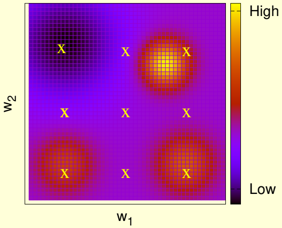
- How to find a "relatively good" w?

Some Natural Approaches

Random weight guessing

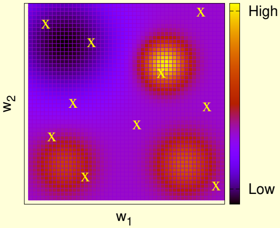


Grid search

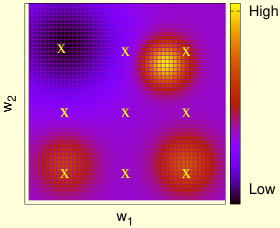


Some Natural Approaches

Random weight guessing



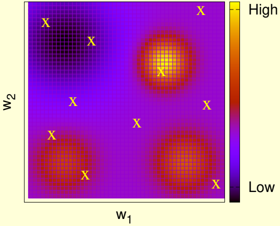
Grid search



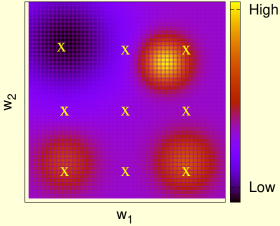
- These approaches work for small dimensions d (say $d \leq 5$).

Some Natural Approaches

Random weight guessing



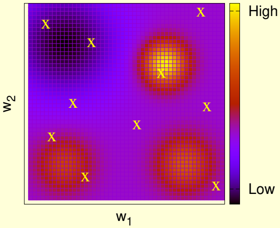
Grid search



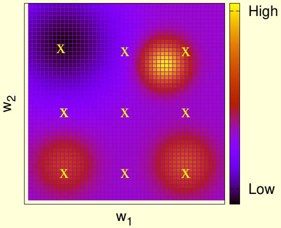
- These approaches work for small dimensions d (say $d \leq 5$).
- No method can be expected to work well for very large d (1000's or higher).

Some Natural Approaches

Random weight guessing



Grid search



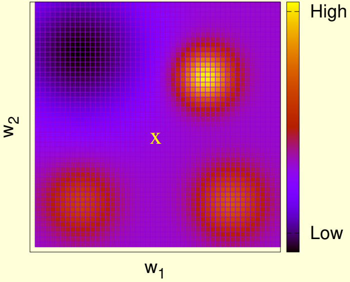
- These approaches work for small dimensions d (say $d \leq 5$).
- No method can be expected to work well for very large d (1000's or higher).
- Local search works for intermediate d (10's, 100's).

Local Search

- Illustrative method: Hill Climbing.

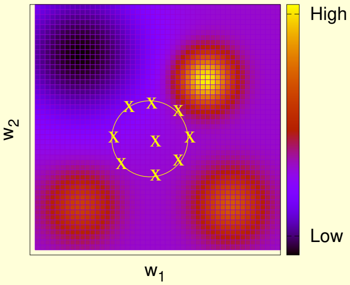
Local Search

- Illustrative method: Hill Climbing.



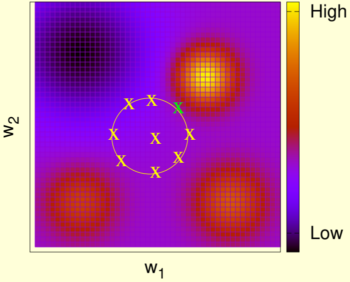
Local Search

- Illustrative method: Hill Climbing.



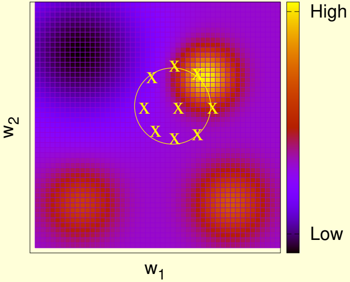
Local Search

- Illustrative method: Hill Climbing.



Local Search

- Illustrative method: Hill Climbing.



Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...
  No decided winner among these (depends on problem, not much guidance).

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...
  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...
  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.
  − Ignores the structure of f.
  + Highly parallelisable.
  + Allow easy integration of domain knowledge.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...
  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.
  − Ignores the structure of f.
  + Highly parallelisable.
  + Allow easy integration of domain knowledge.
  Validation/successes primarily empirical; not much theoretical justification.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...
  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.
  $-$ Ignores the structure of f.
  $+$ Highly parallelisable.
  $+$ Allow easy integration of domain knowledge.
  Validation/successes primarily empirical; not much theoretical justification.

- Called Policy search when applied on the RL problem.

Reinforcement Learning

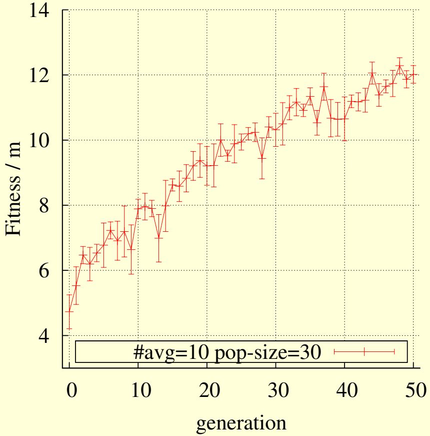
1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
    - Humanoid robot soccer
      Joint work with Patrick MacAlpine, Yinon Bentor, Daniel Urieli, and Peter Stone (UT Austin Villa robot soccer team).
    - Railway scheduling

Gait Optimisation: Policy Parameters

Notation	Description
maxStep_i^*	Maximum step sizes allowed for x, y, and θ
y_{shift}^*	Side to side shift amount with no side velocity
z_{torso}^*	Height of the torso from the ground
z_{step}^*	Maximum height of the foot from the ground
f_g^*	Fraction of a phase that the swing foot spends on the ground before lifting
f_a	Fraction that the swing foot spends in the air
f_s^*	Fraction before the swing foot starts moving
f_m	Fraction that the swing foot spends moving
ϕ_{length}^*	Duration of a single step
δ^*	Factors of how fast the step sizes change
y_{sep}	Separation between the feet
x_{offset}^*	Constant offset between the torso and feet
x_{factor}^*	Factor of the step size applied to the forwards position of the torso
err_{norm}^*	Maximum COM error before the steps are slowed
err_{max}^*	Maximum COM error before all velocity reach 0

Design and Optimization of an Omnidirectional Humanoid Walk: A WinningApproach at the RoboCup 2011 3D Simulation Competition. Patrick MacAlpine, Samuel Barrett, Daniel Urieli, Victor Vu, and Peter Stone. In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012), AAAI Press, 2012.

Progress of Forward Speed Optimisation



On Optimizing Interdependent Skills: A Case Study in Simulated 3D Humanoid Robot Soccer. Daniel Urieli, Patrick MacAlpine, Shivaram Kalyanakrishnan, Yinon Bentor, and Peter Stone. In Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), pp. 769–776, IFAAMAS, 2011.

RoboCup 2011 3D Simulation Competition

- UT Austin Villa combined score: 136–0 (over 24 games).

Rank	Team	Goal Difference
3	apollo3d	1.45 (0.11)
5-8	boldhearts	2.00 (0.11)
5-8	robocanes	2.40 (0.10)
2	cit3d	3.33 (0.12)
5-8	fcportugal3d	3.75 (0.11)
9-12	magmaoffenburg	4.77 (0.12)
9-12	oxblue	4.83 (0.10)
4	kylinsky	5.52 (0.14)
9-12	dreamwing3d	6.22 (0.13)
5-8	seuredsun	6.79 (0.13)
13-18	karachikoalas	6.79 (0.09)
9-12	beestanbul	7.12 (0.11)

. . .

UT Austin Villa 2011: A Champion Agent in the RoboCup 3D Soccer Simulation Competition. Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Ştiurcă, Victor Vu, and Peter Stone. In Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 129–136, IFAAMAS, 2012.

Reinforcement Learning

1. Tile coding
2. Issues in control with function approximation
3. Policy search
4. Case studies
   - Humanoid robot soccer
   - Railway scheduling
     Joint work with Rohit Prasad and Harshad Khadilkar.

Railways and the Economy

[1]

- Indian Railways, 20000+ trains, 9.1 billion yearly ridership.
- Delay incurs economic costs

[1] `https://pixabay.com/photos/transportation-system-travel-vehicle-3351330/`.

Railway Rescheduling Problem

Train	Station	Timetable Arrival Time	Timetable Departure Time	Minimum Halt Time	Minimum Run Time	Scheduled Arrival Time	Scheduled Departure Time	Delay
101	Alpha	4:30	4:40	10	30			
102	Alpha	0:00	0:05	5	10			
601	Echo	9:15	9:16	1	60			
401	Delta	3:20	3:35	15	20			

Given an initial timetable
compute a feasible timetable
subject to resource allocation and time constraints
minimising

Priority-weighted departure delay

$$= \sum_{\text{Train } t, \text{ Resource } r} \frac{\text{Delay of train } t \text{ at resource } r}{\text{Priority of train } t}.$$

Railway Rescheduling Problem

Train	Station	Timetable Arrival Time	Timetable Departure Time	Minimum Halt Time	Minimum Run Time	Scheduled Arrival Time	Scheduled Departure Time	Delay
101	Alpha	4:30	4:40	10	30			
102	Alpha	0:00	0:05	5	10			
601	Echo	9:15	9:16	1	60			
401	Delta	3:20	3:35	15	20			

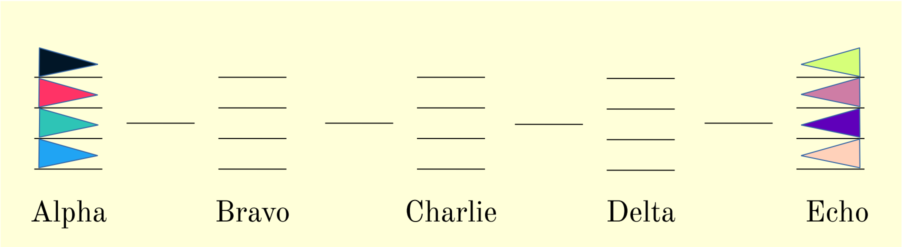
Given an initial timetable
compute a feasible timetable
subject to resource allocation and time constraints
minimising

Priority-weighted departure delay

$$= \sum_{\text{Train } t, \text{ Resource } r} \frac{\text{Delay of train } t \text{ at resource } r}{\text{Priority of train } t}.$$

Illustration

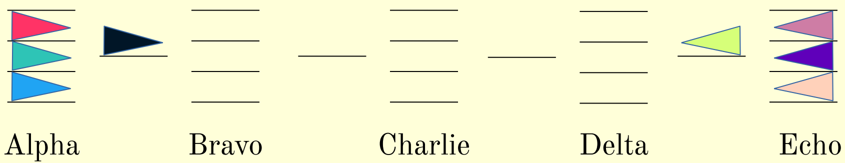
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

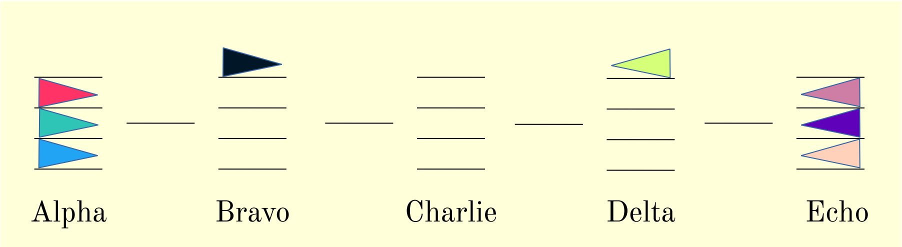
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

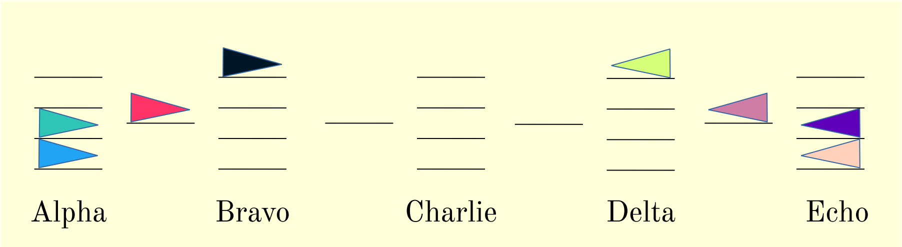
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

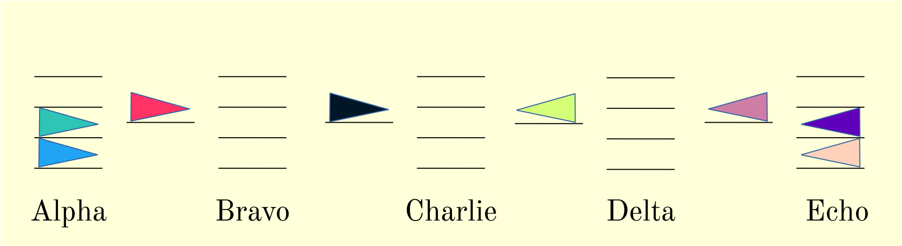
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

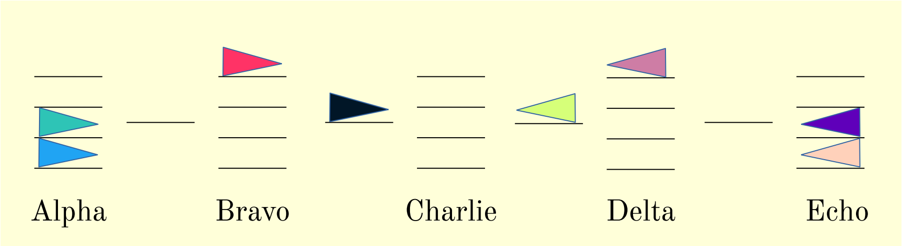
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

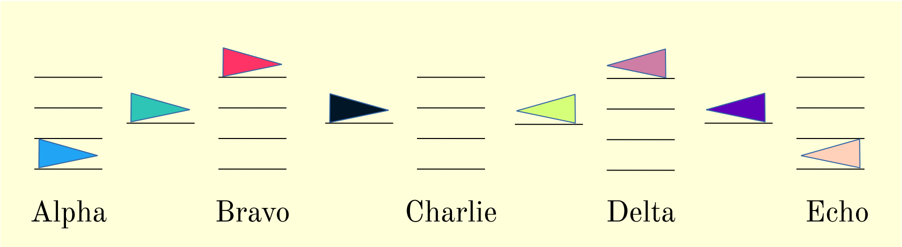
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

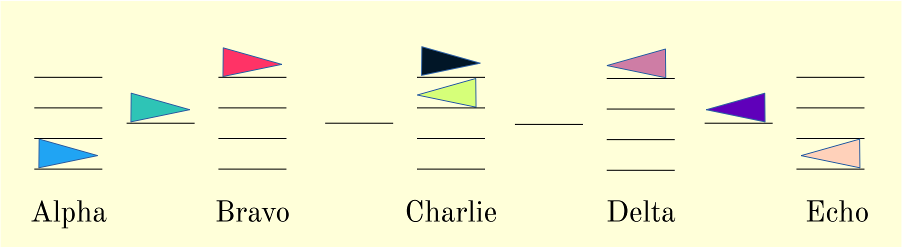
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

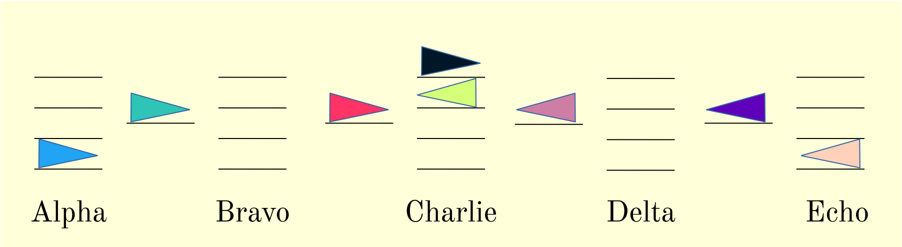
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

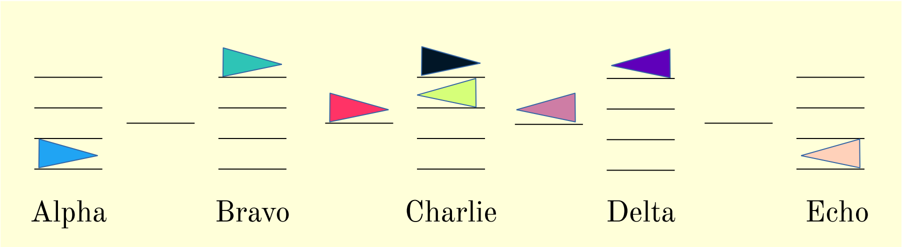
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

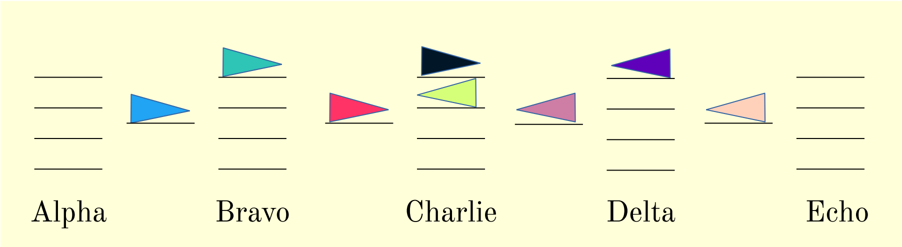
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

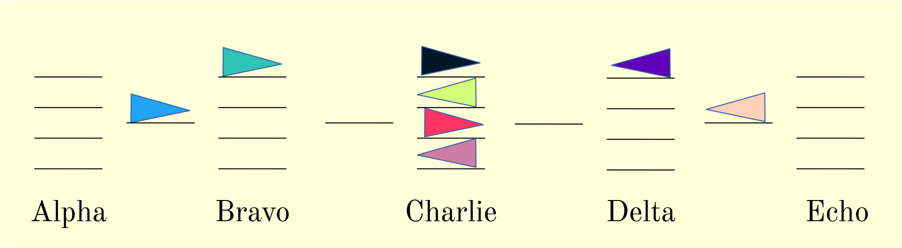
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

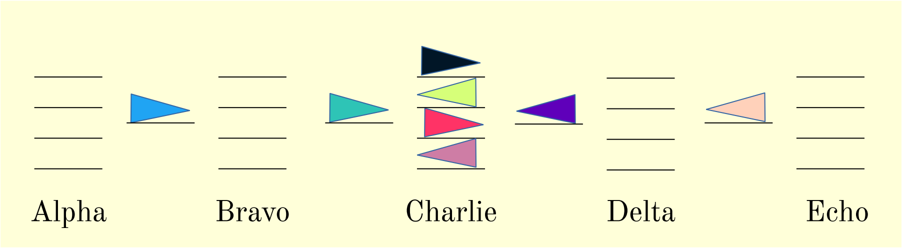
A hypothetical railway line with 8 trains and 5 stations



Intermediate state

Illustration

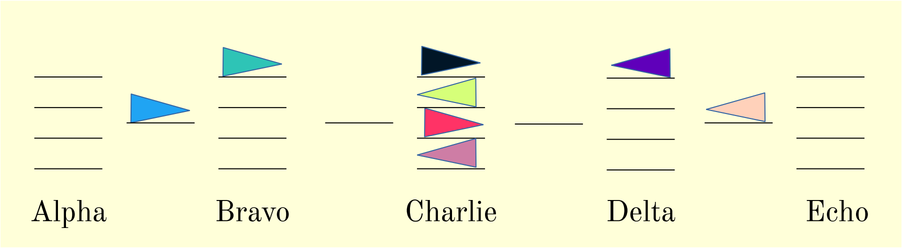
A hypothetical railway line with 8 trains and 5 stations



Deadlock!

Illustration

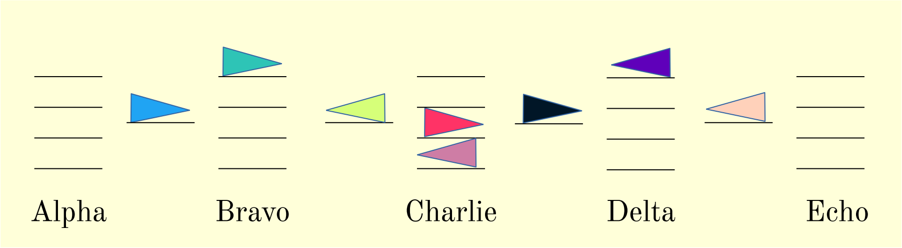
A hypothetical railway line with 8 trains and 5 stations



Intermediate state

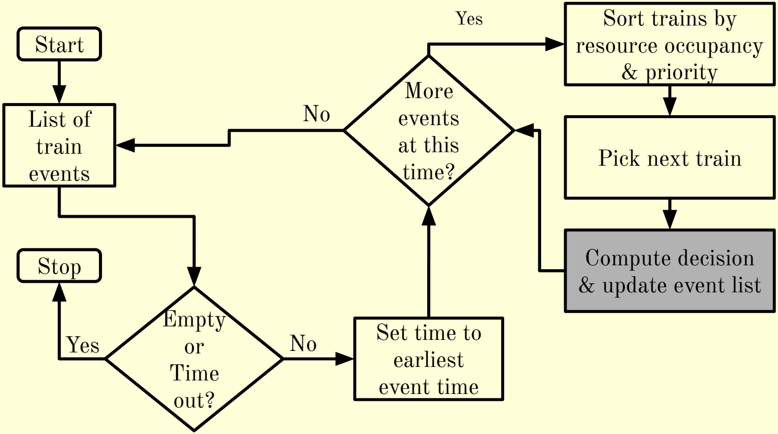
Illustration

A hypothetical railway line with 8 trains and 5 stations



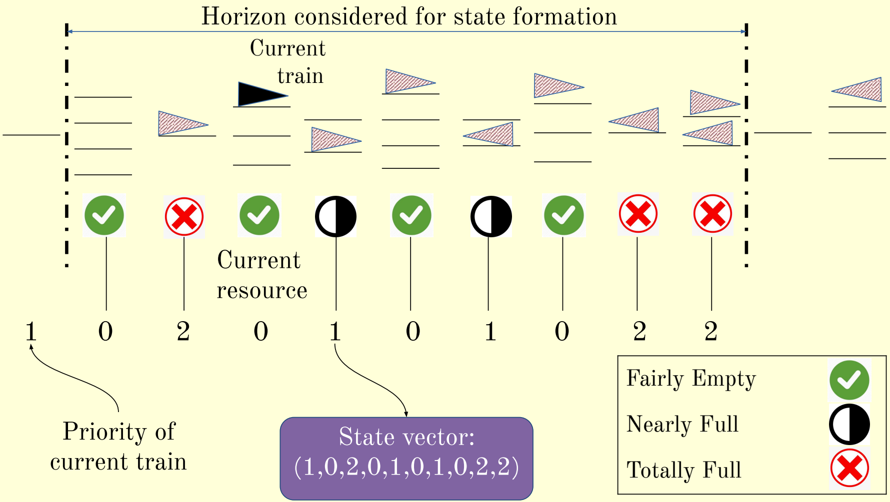
No Deadlock!

Our Solution



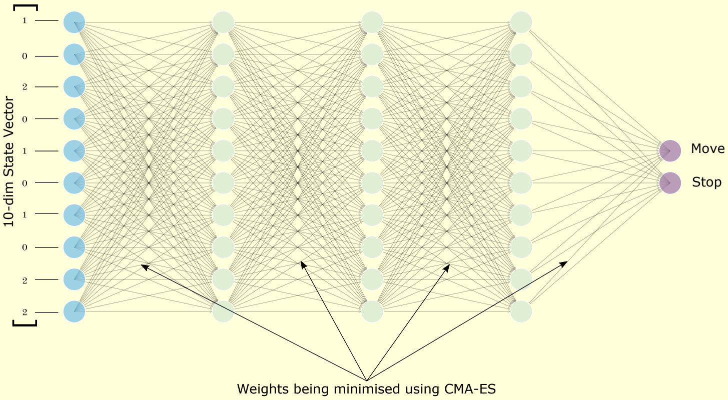
- The wrapper layer picks a potential train to move.
- We optimise the module that decides MOVE / STOP.

State Space



Horizon considered for state formation
Current train
Current resource
1 0 2 0 1 0 1 0 2 2
Priority of current train
State vector: (1,0,2,0,1,0,1,0,2,2)
Fairly Empty
Nearly Full
Totally Full

Policy Representation

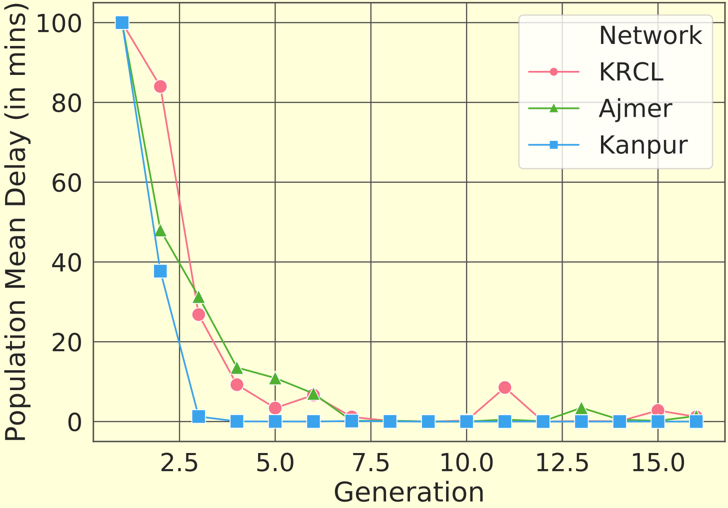


$$d = 352.$$

Benchmark Railway Lines

Scenario	Type	Stations	Trains	Events	Timetable span
HYP-2	Line	11	60	1320	4 hours
HYP-3	Line	11	120	2640	7 hours
KRCL	Line	59	85	5418	3 days
Kanpur	Line	27	190	7716	3 days
Ajmer	Line	52	444	26258	7 days

Results



Population Mean Delay (in mins)
100
80
60
40
20
0
2.5
5.0
7.5
10.0
12.5
15.0
Generation
Network
KRCL
Ajmer
Kanpur

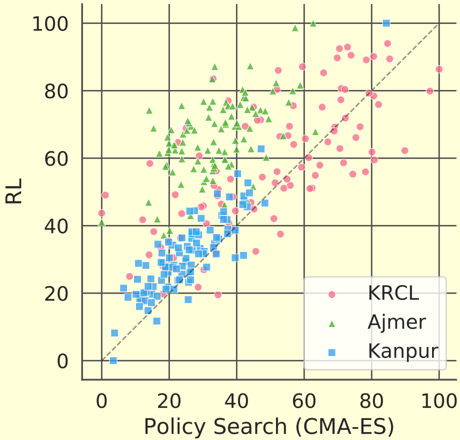
Summary of Results

Priority-weighted departure delay.

	Policy search	RL	TAH-FP	TAH-CF	Naive	GWP	PTD
HYP-2	**4.28 (0)**	4.78 (0)	4.58 (0)	5.93 (0)	11.16 (2)	4.35 (0)	714.00 (0)
HYP-3	**15.50 (0)**	18.54 (0)	61.89 (97)	140.14 (95)	- (100)	16.35 (0)	2003.98 (0)
KRCL	**42.34 (0)**	43.04 (0)	46.41 (8)	47.02 (0)	- (100)	42.40 (0)	4714.08 (0)
Ajmer	**3.92 (0)**	4.65 (0)	10.76 (3)	5.99 (0)	9.25 (76)	3.99 (3)	8304.84 (0)
Kanpur	**1.54 (0)**	1.66 (0)	2.19 (0)	2.28 (0)	1.85 (0)	1.54 (0)	313.60 (0)

Comparison with RL (Q-learning)

Priority-weighted departure delay.



Summary

- Initial lectures assumed finite state spaces.

  Seldom seen in practice!
- Need to generalise over states (sometimes actions).

  Function approximation has many empirical successes, yet is often problematic, especially for control.
- Quality of features/representation determines the validity of Markovian assumption.
- Policy search ignores Markovian structure—which sometimes works to its advantage!

  Conceptually simple; also has many empirical successes.

Summary

- Initial lectures assumed finite state spaces.

  Seldom seen in practice!
- Need to generalise over states (sometimes actions).

  Function approximation has many empirical successes, yet is often problematic, especially for control.
- Quality of features/representation determines the validity of Markovian assumption.
- Policy search ignores Markovian structure—which sometimes works to its advantage!

  Conceptually simple; also has many empirical successes.

- Next week: policy gradient methods.