Master Recherche IAC
Robots et agents autonomes

Emanuel Aldea – Odalric Maillard – Michèle Sebag
TAO
CNRS – INRIA – LRI, Université Paris-Sud

Feb. 13th 2015
Approaches

1. Optimal control

2. Reactive behavior

3. Planning
   Reinforcement Learning
   Optimization
Case 3. Planning

Approaches

- Reinforcement learning
- Inverse reinforcement learning
- **Policy search (= optimize the controller)**
  - Gradient-based
  - Evolutionary robotics
  - Imitation-based
  - Preference-based RL

Challenges

- Design the objective function (define the optimization problem)
- Solve the optimization problem
- Assess the validity of the solution
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Policy search, formal background

Assumption

- We know the policy search space \( \pi : \text{State} \mapsto \text{Action} \)
  For instance: Neural Nets, Decision list
- This search space \( \Theta \) is parametric \( \equiv \mathbb{R}^d \)
- There exists a computable objective function to be optimized:

\[
\theta \mapsto \pi_\theta \mapsto \text{behavior} \mapsto F(\theta)
\]

An optimization problem

Find \( \theta^* = \arg\max \{ F(\theta) \} \)

Specificities

- Noisy optimization (actuators, motors) and partially observable setting
- Can (must) incorporate prior knowledge
  search space structure; initialization; objective function
Example:
swarm robots moving in column formation

Robot
Example, foll’d

<table>
<thead>
<tr>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1$ blind zone</td>
</tr>
<tr>
<td>$l_2$ sensor range</td>
</tr>
<tr>
<td>$\phi$ Vision angular range</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r(t), s(t)$ positions</td>
</tr>
<tr>
<td>$\theta(t)$ angular direction</td>
</tr>
</tbody>
</table>
Example of a (almost manual) controller

<table>
<thead>
<tr>
<th>Info. from the image sensors</th>
<th>Info. from the IR sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq x_{\text{image}} \leq \alpha$</td>
<td>$0 \leq x_{\text{IR}} &lt; \beta_0$</td>
</tr>
<tr>
<td>$\alpha &lt; x_{\text{image}} &lt; (19 - \alpha)$</td>
<td>$\beta_0 \leq x_{\text{IR}} &lt; \beta$</td>
</tr>
<tr>
<td>$\alpha \leq x_{\text{image}} \leq 19$</td>
<td>$\beta \leq x_{\text{IR}}$</td>
</tr>
<tr>
<td>preceding robot NOT FOUND</td>
<td>move backward or turn right</td>
</tr>
<tr>
<td></td>
<td>turn left</td>
</tr>
<tr>
<td></td>
<td>move backward or turn right</td>
</tr>
<tr>
<td></td>
<td>stop move forward</td>
</tr>
<tr>
<td></td>
<td>move backward or turn right</td>
</tr>
<tr>
<td></td>
<td>turn right</td>
</tr>
<tr>
<td></td>
<td>move backward or turn right</td>
</tr>
<tr>
<td></td>
<td>move forward</td>
</tr>
</tbody>
</table>
Toward defining $\mathcal{F}$

- The $i$-th robot follows the $k$-th robot at time $t$ iff the center of gravity of $k$ belongs to the perception range of $i$ ($s_k(t) \in A_i(t)$).
- The $i$-th robot is a leader if i) it does not follow any other robot; ii) there exists at least one robot following it.
- A column is a subset $\{i_1, \ldots, i_K\}$ such that robot $i_{k+1}$ follows robot $i_k$ and robot $i_1$ is a leader.
- A deadlock is a subset $\{i_1, \ldots, i_K\}$ such that robot $i_{k+1}$ follows robot $i_k$ and robot $i_1$ follows robot $i_K$. 
Milestones

1. From $\theta$ to $\pi_\theta$ trivial

2. From $\pi_\theta$ to the robot behavior

3. From the robot behavior to evaluating $\mathcal{F}(\theta)$

4. From trials $\{((\theta_t, \mathcal{F}(\theta_t))\}$ to $\theta^*$
Milestone 1
From the controller $\pi_\theta$ to the robot behavior

How

- In silico = in simulation
  - Main approach for evolutionary robotics
  - No way, says the roboticist

- In situ: embeds the policy on the robot, and sees.
  - The robot breaks before long
  - Makes it difficult to compute $\mathcal{F}(\theta)$

- Both

Hod Lipson & Bongard 2006
Milestone 1

Bottleneck: Accurate predictions

▶ World model: what is out there.
  SLAM, Simultaneous Localization and Mapping

Long term planning

▶ Forward model: what will happen if robot selects action $a$ in state $s$
  Local model of itself

Short term planning

▶ Uncertainties about e.g. sensors or actuators models, initial localization.
Milestone 1

**Bottleneck: Accurate predictions, follow’d,**

- Partially observable effects
  ex., in the case of swarms: there are many robots does robot Bob know robot Alice’s plans? If yes, centralized resolution
  Else, Alice’s behavior is impredictable (and Bob can’t predict with certainty what will be in his vision cone).
  → non deterministic model.
  thus, the behavior is a random variable; $\mathcal{F}(\theta)$ becomes an expectation,

$$\mathbb{E}_{\sim \pi_{\theta}}[\mathcal{F}(\text{behavior})]$$
Milestone 2
From the robot behavior to $F(\theta)$

How

- In simulation: define computable $F$
  by trials and errors (fitness shaping)
  manual (see section evolutionary robotics)
- In situ:
  - Interactive
  - Manual
  - Measurements (e.g. data mining on the videos).
Milestone 3
Optimisation

How

- Gradient-based approaches
- Direct Policy Search
- Black-box optimization
- Evolutionary Robotics
- Surrogate optimization
- Preference reinforcement learning

What is optimized

- policy $\equiv \theta$
- Value function. (satisfies Bellman equation)
- Energy function $H(s, a)$ (same use, but without Bellman)

$$\pi(s) = \arg\max_a\{H(s.a)\}$$
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Direct policy search, formal background

Assumption

- Function $F(\theta)$, to be optimized

Pros

- No divergence, even under function approximation
- Policies are much simpler to represent (a neural net)
- Partial observability does not hurt convergence
  increases computational cost and harms long-term value

Cons

- Lost convergence to the globally optimal policy
- Lost the Bellman constraint $\rightarrow$ larger variance
Direct policy search, principles

Recall: Policy return estimate

\[ V(s) = \mathbb{E}\left[ \sum_t \gamma^t r(s_t) \mid s_0 = s \right] \]

or long term average reward

\[ V(s) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[ \sum_t r(s_t) \mid s_0 = s \right] \]

Assumption: ergodic Markov chain

(After a while, the initial state does not matter).

- \( V(s) \) does not depend on \( s \)
- One can estimate the percentage of time spent in state \( s \)

\[ q(\theta, s) = Pr_\theta(S = s) \]

Another policy return estimate

expected average reward

\[ V = \mathbb{E}_\theta[r(S)] = \sum_s r(s)q(\theta, s) \]
1. $\mathcal{F}(\theta) = \mathbb{E}_\theta[r(S)] = \sum_s r(s)q(\theta, s)$

2. Compute or estimate the gradient, $\nabla \mathcal{F}(\theta)$

3. Use it: (can do better)

$$\theta_{t+1} = \theta_t + \alpha \nabla \mathcal{F}(\theta)$$
Computing the derivative

\[ \nabla V = \nabla (\sum_s r(s)q(\theta, s)) = \sum_s r(s) \nabla q(\theta, s) \]

Then:

\[ \nabla V = \mathbb{E}_\theta [r(S) \frac{\nabla q(\theta, S)}{q(\theta, S)}] \]

**Unbiased estimate of the gradient** (integral = empirical sum)

\[ \hat{\nabla} V = \frac{1}{N} \sum_i r(s_i) \frac{\nabla q(\theta, s_i)}{q(\theta, s_i)} \]
Computing the derivative, foll’d

Using trajectories $((s_t, r(s_t)))$:

Given observations et rewards,

$$\nabla q(\theta, s_t) = \sum_{i=0}^{t-1} \frac{\nabla p_\theta(s_i, s_{i+1})}{p_\theta(s_i, s_{i+1})}$$

where $p_\theta(s_i, s_j)$ is the probability of going from $s_i$ to $s_j$ with $\pi_\theta$.

Eligibility trace

$$z_0 = 0; \quad z_t = z_{t-1} + \frac{\nabla p_\theta(s_{t-1}, s_t)}{p_\theta(s_{t-1}, s_t)}$$
Computing the derivative, foll’d

Approximations

\[ z_t = \sum_{k=t-n}^{t-1} \frac{\nabla p_\theta(s_k, s_{k+1})}{p_\theta(s_k, s_{k+1})} \]

or

\[ z_t = \beta z_{t-1} + \frac{\nabla p_\theta(s_{t-1}, s_t)}{p_\theta(s_{t-1}, s_t)} \]

Quality

\[ \hat{\nabla}_\beta V = \frac{1}{T} \sum_t r(s_t)z_t \]

Baxter Bartlett 2001

\[ \lim_{\beta \to 1} \hat{\nabla}_\beta V = \nabla V \]

Role of \( \beta \): tradedoff bias/variance.
Discussion

Pros

▶ Many achievements: fine manipulation (peg-in-hole), learning biped walking with integrated trajectory generation and execution, first results using a real humanoid robot.

Cons

▶ Finite state space
▶ Adversely affected by reward variance
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Evolutionary Robotics, Milestones

1. Select the search space $\Theta$
2. Define the objective function $\mathcal{F}(\theta)$
   Sky is the limit: controller; morphology of the robot; co-operation of several robots...
3. Define a computable objective function
   in simulation, in-situ, reality gap
4. Optimize: Evolutionary Computation (EC); variants thereof
5. Test the found solution
1. Search Space

Neural Nets

- Universal approximators; continuity; generalization hoped for.
- Fast computation
- Can include priors in the structure
- Feedforward: reactive; Recurrent, with internal state

Critical issues

- Find the structure;
  (structured EC much more difficult)

See NEAT and HyperNEAT

Stanley Miikkulainen, 2002

NeuroEvolution of Augmented Topology
1. Search Space, foll’d

Classifier Systems

if (true) then leftSpeed=2; rightSpeed=2;
if (leftSensor > threshold) then rightSpeed=0;
if (rightSensor > threshold) then leftSpeed=0;
if (leftSensor > threshold) and (rightSensor > threshold) then leftSpeed=-2; rightSpeed=-2;

Finite State Automata
1. Search Space, foll’d

**Genetic Programming**: trees made of

- Nodes (operators) $\mathcal{N}$
- Leaves (operands) $\mathcal{T}$

$$\text{Search space } \Omega = \text{Trees}(\mathcal{N}, \mathcal{T})$$

**Examples**:

- $\mathcal{N} = \{+, \times\}$
- $\mathcal{T} = \{X, R\}$
  - $\Omega =$ Polynoms of $X$.
- $\mathcal{N} =$ {if-then-else, while-do, repeat-until,..}
- $\mathcal{T} =$ {expressions, instructions}
  - $\Omega =$ Programs

**Key issues**:

- Variable length genomes
- MORE ≠ BETTER
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
2. Objective

The promise: no need to decompose the goal

- Behavioral robotics
  - Manipulations
  - Construction d’une carte
  - Exploration
  - Evitement d’obstacles
  - Deplacement

- Evolutionary robotics
  - ?
  - ?
  - ?
  - ?

Hand crafted decomposition

Emergence of a structure

Capteurs ➔ Moteurs
In practice: bootstrap

- All initial (random) individuals are just incompetent
- Fitness landscape: Needle in the Haystack? (doesn’t work)
- Start with something simple
- Switch to more complex *during evolution*
- Example: visual recognition
2. Objective, foll’d

▶ Functional vs behavioral
  state of controller vs distance walked
▶ Implicit vs explicit
  Survival vs Distance to socket
▶ Internal vs external information
  Sensors, ground truth
▶ Co-evolution: e.g. predator/prey
  performance depends on the other robots

State of art
▶ Standard: function, explicit, external variables
▶ In-situ: behavioral, implicit, internal variables
▶ Interactive: behavioral, explicit, external variables
2. Objective, foll’d

Fitness shaping

▶ Obstacle avoidance

▶ Obstacle avoidance, and move!

▶ Obstacle avoidance, and (non circular) move!!

Finally

\[ \mathcal{F} = \int_{T_{\text{exp.}}} V(1 - \sqrt{\Delta v})(1 - i) \]

▶ \( V \) sum of wheel speed \( r_i \in [-0.5, 0.5] \) → move

▶ \( \Delta v = |r_1 + r_2| \) → ahead

▶ \( i \) maximum (normalised) of sensor values → obstacle avoidance

Behavioral, internal variables, explicit
Result analysis

- **First generations**
  - Most rotate
  - Best ones slowly go forward
  - No obstacle avoidance
  - Perf. depends on starting point

- After $\approx 20$ gen.
  - Obstacle avoidance
  - No rotation

- Thereafter, gradually speed up
Result analysis

- Max. speed 48mm/s (true max = 80)
  Inertia, bad sensors
- Never stuck in a corner
  contrary to Braitenberg

Going further

- Changing environment
- Changing robotic platform
- From simulation to real-world

Fast adaptation
Explore and recharge

Not a reactive behavior

- Battery gets empty in 20s in white zone
- Recharges in black zone
- But no reward in black zone
Explore and recharge, 2

- A ground sensor
  → sees whether the ground is white or black
- 2 sensors passive mode
  → ambient light

Search space: Elman network

- Optimize weights
- Recurrent NN, thus with internal state
- Optimize in situ
Explore and recharge, 2

Performance

\[ \mathcal{F} = \int_{\text{White zone}} V(1 - i) \]

- Lifetime requires a good recharge strategy
- \( V \) cumulative wheel speed \( r_i \in [-0.5, 0.5] \)
  \( \rightarrow \) move
- \( i \) maximum (normalised) of sensor values
  \( \rightarrow \) obstacle avoidance

Behavioral, internal, explicit + implicit
Result analysis

During evolution

Fitness (best and average)

Lifetime

Inspecting best behavior

methods inspired from neurophysiology/ethology

Instrumenting the robot

Battery and motor state along lifetime
Overview

Situation of the problem

Policy search

Direct policy search

**Evolutionary Robotics**
- Search space
- Objective
- Reality Gap
- Co-evolution
- Evolution of morphology

Intrinsic and interactive rewards
- Intrinsic rewards
- Interactive rewards
- Programming by feedback
Reality gap

- What if simulator does not reflect the robot or the environment?
- Optimizes the wrong function
Reality gap, 2
Against in-situ

Finally
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Co-Evolution

Competitive co-evolution

- Goal: survival
- Model: predator-prey

\[
\frac{\partial N_1}{\partial t} = N_1(r_1 - b_1 N_2), \quad \frac{\partial N_2}{\partial t} = N_2(-r_1 + b_2 N_1)
\]

- \( \rightarrow \) population sizes oscillate
- Simulation: fixed population size, performance varies
- Fitness computed by turnament
  
  global, random, with best individuals, \ldots
Predator-prey

- Predator: sees; is slow
- Prey: is blind; is twice as fast

Floreano et Nolfi, 97-99

RN 8+5 → 2 recurrent

RN 8 → 2 recurrent
Fitness

- Round robin tournament, all predators and preys
- Stops when predator catches the prey (ad hoc sensor)
- Or after 500 cycles, ≈ 50s
- Performance (each) + = duration of tournament

Predators must minimize performance Preys must maximize performance

Behavioral, implicit, internal/external
First results

- First predators very bad
- Beware of the Red Queen!
- The final best can be caught by previous best ones!

Paredis 97
Hall of fame

Intuition
Also compete with best ancestors
Hall of fame, 2

Tournament among all individuals in all generations
Black $\equiv$ predator wins, white $\equiv$ prey wins

Ideal situation / Without Hall of Fame / With Hall of Fame

Final best are better than (almost) all ancestors.
**Goal**

- Evolve both morphology and controller
- using a grammar (oriented graph)
- Heavy computational cost  
  simulation, several days on Connection Machine – 65000 proc.
- Evolving locomotion (walk, swim, jump)
- and competitive co-evolution (catch an object)
The creatures, Karl Sims

more?

http://www.youtube.com/watch?v=JBgG_VSP7f8
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Contexte
I. Getting motivated. Internal rewards

Delarboulas et al., PPSN 2010

Requirements

1. No simulation
2. On-board training
   ▶ Frugal (computation, memory)
   ▶ No ground truth
3. Providing “interesting results”
   “Human – robot communication”

Goal: self-driven Robots: Defining instincts
Starting from (almost) nothing

Robot $\equiv$ a data stream

$t \rightarrow x[t] = (\text{sensor}[t], \text{motor}[t])$

Trajectory $= \{x[t], t = 1 \ldots T\}$

Robots trajectory
Starting from (almost) nothing

Robot ≡ a data stream

\[ t \rightarrow x[t] = (\text{sensor}[t], \text{motor}[t]) \]

Trajectory = \{x[t], t = 1 \ldots T\}

Computing the quantity of information of the stream

Given \(x_1, \ldots x_n\), visited with frequency \(p_1 \ldots p_n\),

\[
\text{Entropy}(\text{trajectory}) = - \sum_{i=1}^{n} p_i \log p_i
\]

Conjecture

Controller quality \(\propto\) Quantity of information of the stream
Building sensori-motor states

Avoiding trivial solutions...
If sensors and motors are continuous / high dimensional
▶ then all vectors $x[t]$ are different
▶ then $\forall i, p_i = 1/T; \quad \text{Entropy} = \log T$

... requires generalization
From the sensori-motor stream to clusters

Clusters in sensori-motor space ($\mathbb{R}^2$)
Clustering

**k-Means**

1. Draw \( k \) points \( x[t_i] \)
2. Define a partition \( C \) in \( k \) subsets \( C_i \)

\[
C_i = \{ x / d(x, x[t_i]) < d(x, x[t_j]), j \neq i \}
\]

**\( \epsilon \)-Means**

1. Init : \( C = \{ \} \)
2. For \( t = 1 \) to \( T \)
   
   - If \( d(x[t], C) > \epsilon \), \( C \leftarrow C \cup \{x[t]\} \)
Curiosity Instinct

Search space

- Neural Net, 1 hidden layer.

Definition

- Controller $F +$ environment $\rightarrow$ Trajectory
- Apply Clustering on Trajectory
- For each $C_i$, compute its frequency $p_i$

$$\mathcal{F}(F) = - \sum_{i=1}^{n} p_i \times \log(p_i)$$
Curiosity instinct: Maximizing Controller IQ

Properties

- Penalizes inaction: a single state $\rightarrow$ entropy $= 0$
- Robust w.r.t. sensor noise (outliers count for very little)
- Computable online, on-board (use $\epsilon$-clustering)
- Evolvable onboard

Limitations: does not work if

- Environment too poor
  
  (in desert, a single state $\rightarrow$ entropy $= 0$)

- Environment too rich
  
  (if all states are distinct, $Fitness(\text{controller}) = \log T$)

both under and over-stimulation are counter-effective.
From curiosity to discovery

Intuition

- An individual learns sensori-motor states ($x[t_i]$ center of $C_i$)
- The SMSs can be transmitted to offspring
- giving the offspring an access to “history”
- The offspring can try to “make something different”

fitness(offspring) = Entropy(Trajectory(ancestors $\cup$ offspring))

NB: does not require to keep the trajectory of all ancestors. One only needs to store $\{C_i, n_i\}$
From curiosity to discovery

Cultural evolution transmits genome + “culture”

1. parent = (controller genome, (C₁, n₁), . . . (Cₖ, nₖ))
2. Perturb parent controller → offspring controller
3. Run the offspring controller and record x[1], . . . x[T]
4. Run ϵ-clustering variant.

\[
Fitness(\text{offspring}) = - \sum_{i=1}^{\ell} p_i \log p_i
\]
\( \epsilon \)-clustering variant

**Algorithm**

1. Init: \( C = \{(C_1, n_1), \ldots, (C_K, n_K)\} \)
2. For \( t = 1 \) to \( T \)
   - If \( d(x[t], C) > \epsilon \), \( C \leftarrow C \cup \{x[t]\} \)
3. Define \( p_i = \frac{n_i}{\sum_j n_j} \)

\[
\text{Fitness\ (offspring)} = - \sum_{i=1}^{\ell} p_i \log p_i
\]
Validation

Experimental setting
Robot = Cortex M3, 8 infra-red sensors, 2 motors.
Controller space = ML Perceptron, 10 hidden neurons.

Medium and Hard Arenas
Validation, 2

Plot points in hard arena visited 10 times or more by the 100 best individuals.

Nolfi & Floreano  Lehman & Stanley  Curiosity  Discovery

PPSN 2010
Partial conclusions

Entropy-minimization
- computable on-board; no need of prior knowledge/ground truth
- yields “interesting” behavior
- needs stimulating environment

See also
- Robust Intrinsic Motivation
  Baranes & Oudeyer 05,07; Oudeyer, NIPS 2012
Overview

Situation of the problem

Policy search

Direct policy search

Evolutionary Robotics
  Search space
  Objective
  Reality Gap
  Co-evolution
  Evolution of morphology

Intrinsic and interactive rewards
  Intrinsic rewards
  Interactive rewards
  Programming by feedback
Prior knowledge in RL

- In the form of a **Reward** function $\mathcal{R} : S \times A \mapsto \mathbb{R}$
- Find **Policy** $\pi$ Maximizing $\mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(s_t)) \right]$
Reinforcement Learning and Rewards

Prior knowledge in RL

- In the form of a **Reward** function \( R : S \times A \mapsto \mathbb{R} \)
- Find **Policy** \( \pi \) Maximizing \( \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \right] \)

Bottlenecks

- Rewards \( \equiv \) ground truth challenges in-situ
- In a swarm context \( R \) can be
  - Centralized: \( R : (S \times A) \times \cdots \times (S \times A) \mapsto \mathbb{R} \)
    (global vision, tractability issues)
  - Decentralized: \( R_1 : (S \times A) \mapsto \mathbb{R}, \ldots, R_N : (S \times A) \mapsto \mathbb{R} \)
    - Tractable: Every robot optimize its own reward
    - Trials and Errors process to tune it
Inverse Reinforcement Learning?

Ng Russell 00, Abbeel Ng 04, Kolter et al. 07, ..

Prior knowledge in Inverse Reinforcement Learning

- Expert demonstrates a **good behavior** \( \{s_t, a_t, s_{t+1}\} \)

Abbeel & Ng 04
Inverse Reinforcement Learning?

Ng Russell 00, Abbeel Ng 04, Kolter et al. 07, ..

Prior knowledge in Inverse Reinforcement Learning

- Expert demonstrates a **good behavior** \( \{ s_t, a_t, s_{t+1} \} \)

![Driving demonstration](image1)

![Learned behavior](image2)

Abbeel & Ng 04

- From this, learn a reward function \( \mathcal{R} \)

\[
\forall a \neq a_t, \text{Action Value}(s_t, a_t) \geq \text{Action Value}(s_t, a)
\]

- Then apply standard RL!
What if no idea about a good behavior

Alan Winfield & Wenguo Liu 08

each point is a robot
Preference-based Policy Learning

Step 1: use expert’s feedback to learn the goal (PPL)

- **Prior knowledge:** pairwise preferences over behaviors
- **Expert** become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - **Expert:** Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

Akrour et al. 2011
Preference-based Policy Learning

**Step 1: use expert’s feedback to learn the goal (PPL)**  
Akrour et al. 2011

- **Prior knowledge:** pairwise preferences over behaviors
- **Expert** become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - **Expert:** Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

**Step 2: reduce expert’s burden (APRIL)**  
Akrour et al. 2012

- A **hundred** of demonstrations to find a satisfying $\pi$ in our exp.
- How can we reduce ”Expert Sample Complexity”?
Preference-based Policy Learning

Step 1: use expert’s feedback to learn the goal (PPL)

- **Prior knowledge:** pairwise preferences over behaviors
- **Expert** become a *critic* instead of a *performer*
- **Iterate**
  - **Agents:** Demonstrate a behavior
  - **Expert:** Compare behavior with previous ones (better/worse)
  - **Agents:** Optimize expert preferences model + exploration term

Step 2: reduce expert’s burden (APRIL)

- A *hundred* of demonstrations to find a satisfying $\pi$ in our exp.
- How can we reduce ”Expert Sample Complexity”?
- Active Learning!?
Step 1. Preference-based Policy Learning

1. Demonstrate two policies
2. Ask the user her preference
3. Train a preference model $J_t$ SVM ranking
4. Self-train: find a policy $\pi$ maximizing $J_t$
5. ... $+\alpha_t$ Novelty adaptive exploration wrt archive
6. Demonstrate $\pi$, iterate

- $\alpha_t$ increases when success
Which space?

Environment helps!

- Parametric Representation
  policy $\pi$ in $\mathbb{R}^D$ 
  NN weight vector

- Behavioral Representation
  $\pi \rightarrow$ trajectory $\rightarrow$ histogram of sensorimotor states $\mathbb{R}^d$

Comments

- Expert interested in robot behavior (not in NN weights)
- Mapping $\mathbb{R}^D \mapsto \mathbb{R}^d$ non Lipschitz

  small variations in $\mathbb{R}^D$ $\rightarrow$ large variations in $\mathbb{R}^d$

$\rightarrow$ Learn the expert's preference model in $\mathbb{R}^d$
Modelling the expert’s preferences

A system of values $V$

- For $i$-th sensorimotor state, a weight $v[i]$
- Map $\pi$ onto its sms histogram $p_\pi[i]$

$$V(\pi) = \langle v, p_\pi \rangle$$

Rank-based learning

Given $\pi^{(1)} \prec \ldots \prec \pi^{(k)}$, minimize

$$\frac{1}{2} ||w||^2$$

subject to

$$\langle w, p_\pi^\ell \rangle < \langle w, p_\pi^{\ell+1} \rangle + 1 \quad \ell = 1 \ldots k - 1$$
Validation

Getting out of a maze

Comments

- \(PPL_d\) reaches the goal after 39 interactions (saves 3/4 interactions)
- \(PPL_D\) inefficient; Novelty search (Stanley 2010) inefficient.
Coordinated exploration of an arena
Two independent robots, operated with same controller; goal is to maximize the number of zones simultaneously visited by both robots.
Comments

▶ More challenging goal
  no visual primitive (see other robot, see an obstacle
▶ PPL\(_d\) efficient (saves 9/10 interactions)
▶ PPL\(_D\) inefficient; Novelty search (Stanley 2010) very inefficient (large search space).
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose \( u = \arg \max J_w(u) \)?
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $u = \arg \max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose \( u = \text{arg max} \ J_w(u) \)?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select \( u \) maximizing *Expected Utility of Selection* (EUS) of candidate \( u \) w.r.t \( U_t \)

Viappiani & Boutilier 10
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose \( u = \arg \max J_w(u) \)?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select \( u \) maximizing \textit{Expected Utility of Selection (EUS)} of candidate \( u \) w.r.t \( U_t \)

\[
EUS(u; U_t) = \mathbb{E}_w[\max(\langle w, u \rangle, \langle w, u^*_t \rangle)]
\]
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose \( u = \arg \max J_w(u) \)?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal

Select \( u \) maximizing Expected Utility of Selection (EUS) of candidate \( u \) w.r.t. \( U_t \)

\[
EUS(u; U_t) = \mathbb{E}_w [\max (\langle w, u \rangle, \langle w, u^* \rangle)]
\]

\( u > u^* \)

\[
\mathbb{E}_{w \in W^+} [\langle w, u \rangle]
\]
Step 2. Active Preference-based Reinforcement Learning

Why an active component

What if we choose $u = \arg \max J_w(u)$?

- Does not favor discovery of novel sensori-motor states
- No notion of Information Gain

Proposal
Select $u$ maximizing Expected Utility of Selection (EUS) of candidate $u$ w.r.t $U_t$

$$EUS(u; U_t) = \mathbb{E}_w[\max(\langle w, u \rangle, \langle w, u^* \rangle)]$$

$$\begin{align*}
& u > u^* \\
& \mathbb{E}_{w \in W^+}[\langle w, u \rangle] + \mathbb{E}_{w \in W^-}[\langle w, u^* \rangle]
\end{align*}$$
Implementation

**EUS Intractable** (in practice, $\text{dim}(u) > 1000$)

- All preference constraints define a version space

Version space of consistent estimates

\[
\text{AEUS}(u; U_t) = \langle w^+, u \rangle_F^+ + \langle w^-, u^*_t \rangle_F^-
\]
Implementation

\textbf{EUS Intractable} (in practice, $dim(\mathbf{u}) > 1000$)

- All preference constraints define a version space
- A candidate behavior $\mathbf{u}$ splits the VS in two

Version space of consistent estimates
Implementation

**EUS** Intractable (in practice, \( \dim(u) > 1000 \))

- All preference constraints define a version space
- A candidate behavior \( u \) splits the VS in two
- \( w^+ \) and \( w^- \) solution of the associated ranking problem
Implementation

**EUS Intractable** (in practice, \( \text{dim}(u) > 1000 \))

- All preference constraints define a version space
- A candidate behavior \( u \) splits the VS in two
- \( w^+ \) and \( w^- \) solution of the associated ranking problem

Approximated Expected Utility of Selection

\[
AEUS(u; \mathcal{U}_t) = \frac{\langle w^+, u \rangle}{F^+} + \frac{\langle w^-, u^*_t \rangle}{F^-}
\]

Policy selection criteria

\[
\pi_t = \arg \max_{\pi} \mathbb{E}_{u \sim \pi}[AEUS(u)]
\]
APRIL Algorithm

- $\pi_0 \leftarrow \text{random}$
- $u_0 = \text{demonstration of } \pi_0$
- Archive $\mathcal{U}_0 = \{u_0\}$
- FOR $t = 0 \rightarrow T$ (while Expert cooperates)
  - (R) Select $\pi_{t+1} = \arg \max \{\mathbb{E}_{u \sim \pi}[AEUS(u; \mathcal{U}_t)]\}$
  - (R) Demonstrate $u_{t+1}$ from policy $\pi_{t+1}$ to the expert
  - (E) Expert ranks $u_{t+1}$ and archive $\mathcal{U}_t$ is updated.
ENDFOR
Experimental Validation of **AEUS**

- Sample $w^* \in d$-dimensional $L_2$-unit-sphere
- $S = \{u_1, \ldots, u_{1000}\}$ sampled unif. from $L_1$-unit-sphere
- Find $\arg \max_{u \in S} \langle w^*, u \rangle$ using minimal number of pairwise comparisons
- Compare **AEUS** with **SEUS** (SEUS = sample 10,000 $w$ in the VS to approx. EUS)
- **Result**: **AEUS** matches closely **SEUS**!
Policy Learning Tasks

APRIL vs IRL

- Two RL benchmarks: Mountain Car and Cancer Treatment
- What’s the cost of not having a demonstration as input?
- 15 pairwise comparisons!

APRIL vs PPL

- Huge gain compared to non-active variant
Relaxing Expertise Requirements
The art of programming

1970s Specifications

1990s Programming by Examples

2010s Interactive Learning and Optimization
  ▶ Optimizing coffee taste
  ▶ Visual rendering
  ▶ Choice query
  ▶ Information retrieval
  ▶ Robotics
    Akrou et al., 12; Wilson et al., 12; Knox et al. 13; Saxena et al. 13

ICML 14, Active Detection via Adaptive Submodularity, Chen et al.
Programming by feedback

Knowledge-constrained Computation, memory-constrained

1. Computer presents the expert with a pair of behaviors $y_{t1}, y_{t2}$
2. Expert emits preferences $y_{t1} \succ y_{t2}$
3. Computer learns expert’s utility function
4. Computer searches for behaviors with best utility
5. Goto 1
Relaxing Expertise Requirements: The RL trend

**Expert**
- Associates a reward to each state \( RL \)
- Demonstrates a (nearly) optimal behavior \( \text{Inverse RL} \)
- Compares and revises agent demonstrations \( \text{Co-active PL} \)
- Compares demonstrations \( \text{Preference PL, PF} \)

**Agent**
- Computes optimal policy based on rewards \( RL \)
- Imitates verbatim expert’s demonstration \( \text{IRL} \)
- Imitates and modifies \( \text{IRL} \)
- Learns the expert’s utility \( \text{IRL, CPL} \)
- Learns, and selects demonstrations \( \text{CPL, PPL, PF} \)
- Accounts for the expert’s mistakes \( \text{PF} \)
Programming by feedback

Lessons learned from early work

- Asks few preference queries
- Not active preference learning: Sequential model-based optimization

- Accounts for preference noise
  - Expert changes his mind
  - Expert makes mistakes
  - ...especially at the beginning

An alternative

- Agent demonstrates sub-behaviors
- Demonstrations start in interesting starting points $\sim \pi^*$

Wilson et al., 2012
Formal setting

\[ \mathcal{X} \text{ Search space, solution space} \]
\[ \mathcal{Y} \text{ Evaluation space, behavior space} \]

\[ \Phi : \mathcal{X} \mapsto \mathcal{Y} \]

Utility function

\[ U^* : \mathcal{Y} \mapsto \mathbb{R} \quad U^*(y) = \langle w^*, y \rangle \]
\[ U^*_\mathcal{X} : \mathcal{X} \mapsto \mathbb{R} \quad U^*_\mathcal{X}(x) = \mathbb{E}_{y \sim \Phi(\mathcal{X})}[U^*(y)] \]

Requisites

- Evaluation space: simple to learn from few queries
- Search space: sufficiently expressive
Programming by Feedback

Ingredients

- Modelling the expert’s competence
- Learning the expert’s utility
- Selecting the next best behaviors
  - Which optimization criterion
  - How to optimize it
Modelling the expert’s competence

Noise model

Given preference margin \( z = \langle w^*, y - y' \rangle \)

\[
P(y \prec y' \mid w^*, \delta) = \begin{cases} 
0 & \text{if } z < -\delta \\
1 & \text{if } z > \delta \\
\frac{1+z}{2} & \text{otherwise}
\end{cases}
\]

\( \delta \sim U[0, M] \)

![Diagram](attachment:image.png)

Prob of error

\(-\text{delta}\)  \(\frac{1}{2}\)  \(\text{delta}\)

Preference margin Z
Learning the expert’s utility function

**Data** \( \mathcal{U}_t = \{y_0, y_1, \ldots; (y_{i_1} \succ y_{i_2}), i = 1 \ldots t\} \)

- trajectories \( y_i \)
- preferences \( y_{i_1} \succ y_{i_2} \)

**Learning**: find \( \theta_t \) posterior on \( W \) \( W = \) linear fns on \( \mathcal{Y} \)

**Proposition**: Given \( \mathcal{U}_t \),

\[
\theta_t(w) \propto \prod_{i=1,t} P(y_{i_1} \succ y_{i_2} | w) \\
= \prod_{i=1,t} \left( \frac{1}{2} + \frac{w_i}{2M} \left( 1 + \log \frac{M}{|w_i|} \right) \right)
\]

with \( w_i = \langle w, y_{i_1} - y_{i_2} \rangle \), capped to \([-M, M]\).

\[
U_t(y) = \mathbb{E}_{w \sim \theta_t}[\langle w, y \rangle]
\]
Best demonstration pair \((y, y')\)

inspiration, Viappiani Boutilier, 10

**EUS: Expected utility of selection** (greedy)

\[
EUS(y, y') = \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle > 0] \cdot U_{w \sim \theta_t, y > y'}(y) \\
+ \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle < 0] \cdot U_{w \sim \theta_t, y < y'}(y')
\]

**EPU: Expected posterior utility** (lookahead)

\[
EPU(y, y') = \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle > 0] \cdot \max_y U_{w \sim \theta_t, y > y'}(y'') \\
+ \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle < 0] \cdot \max_y U_{w \sim \theta_t, y < y'}(y'') \\
= \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle > 0] \cdot U_{w \sim \theta_t, y > y'}(y^*) \\
+ \mathbb{E}_{\theta_t}[\langle w, y - y' \rangle < 0] \cdot U_{w \sim \theta_t, y < y'}(y'^*)
\]

Therefore

\[
\arg\max EPU(y, y') \leq \arg\max EUS(y, y')
\]
Optimization in demonstration space

NL: noiseless  \hspace{2cm} N: noisy

Proposition

\[ EUS^{NL}(y, y') - L \leq EUS^N(y, y') \leq EUS^{NL}(y, y') \]

Proposition

\[ \max EUS_t^{NL}(y, y') - L \leq \max EPU_t^N(y, y') \leq \max EUS_t^{NL}(y, y') + L \]

Limited loss incurred  \hspace{2cm} (L \sim \frac{M}{20})
Optimization in solution space

1. Find best $y, y'$ → Find best $y$
   to be compared to best behavior so far $y^*_t$
   The game of hot and cold

2. Expectation of behavior utility → utility of expected behavior
   Given the mapping $\Phi$: search $\mapsto$ demonstration space,
   
   $$\mathbb{E}_\Phi[EUS^{NL}(\Phi(x), y^*_t)] \geq EUS^{NL}(\mathbb{E}_\Phi[\Phi(x)], y^*_t)$$

3. Iterative solution optimization
   
   ▶ Draw $w_0 \sim \theta_t$ and let $x_1 = \text{argmax} \ \{ \langle w_0, \mathbb{E}_\Phi[\Phi(x)] \rangle \}$
   
   ▶ Iteratively, find $x_{i+1} = \text{argmax} \ \{ \langle \mathbb{E}_{\theta_i}[w], \mathbb{E}_\Phi[\Phi(x)] \rangle \}$, with $\theta_i$
     posterior to $\mathbb{E}_\Phi[\Phi(x_i)] > y^*_t$.

Proposition. The sequence monotonically converges toward a local optimum of $EUS^{NL}$
Experimental validation

- Sensitivity to expert competence
  Simulated expert, grid world

- Continuous case, no generative model
  The cartpole

- Continuous case, generative model
  The bicycle

- Training in-situ
  The Nao robot
Sensitivity to simulated expert incompetence

Grid world: discrete case, no generative model
25 states, 5 actions, horizon 300, 50% transition motionless

$M_E$  Expert competence
$M_A > M_E$  Computer estimate of expert’s competence

<table>
<thead>
<tr>
<th>$M_E$</th>
<th>$M_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>1/64</td>
<td>1/64</td>
</tr>
<tr>
<td>1/128</td>
<td>1/128</td>
</tr>
</tbody>
</table>

True $w^*$ on gridworld  True utility of $x_t$  expert’s mistakes

A cumulative (dis)advantage phenomenon

The number of expert’s mistakes increases as the computer underestimates the expert’s competence.

For low $M_A$, the computer learns faster, submits more relevant demonstrations to the expert, thus priming a virtuous educational process.
Continuous Case, no Generative Model

The cartpole
State space \( \mathbb{R}^2 \), 3 actions
Dem. space \( \mathbb{R}^9 \), dem. length 3,000

Cartpole

True utility of \( x_t \) fraction in equilibrium

Two interactions required on average to solve the cartpole problem.
No sensitivity to noise.
Continuous Case, with Generative Model

**The bicycle**

Solution space $\mathbb{R}^{210}$ (NN weight vector)
State space $\mathbb{R}^4$, action space $\mathbb{R}^2$, dem. length $\leq 30,000$.

**True utility**

Optimization component: CMA-ES \( \text{Hansen et al., 2001} \)

15 interactions required on average to solve the problem for low noise.
versus 20 queries, with discrete action in state of the art.
Discussion and Perspectives

**Feasibility** of Programming by Feedback for simple tasks

An old research agenda

One could carry through the organization of an intelligent machine with only two interfering inputs, one for pleasure or reward, and the other for pain or punishment.

Next

- Identifying the sub-behaviors responsible for the expert’s like/dislikes (options)
- Accounting for the variance of $U_{y \sim \Phi(x)}(y)$