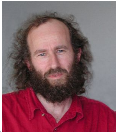
Learning with the human in the loop

Michèle Sebag

Riad Akrour



Marc Schoenauer

TAO

Constructive Machine Learning Wshop, ICML 2015

Evolution of Computer Science

1970s Specifications — Languages & thm proving

1990s Programming by Examples — Pattern recognition & ML

2010s Interactive Learning and Optimization

Evolution of Computer Science

1970s Specifications — Languages & thm proving

1990s Programming by Examples — Pattern recognition & ML

2010s Interactive Learning and Optimization

Motivations

- no explicit specification
- open world — $P(x)$ changes
- under-specified goal

Summary

- Machine Learning needs logics, data, optimization....
- Machine Learning needs feedback: the human in the loop.
- Co-evolution of the human in the loop and the learner.

If the computer could read the user's mind

Shannon's Mind Reading Machine

http://cs.williams.edu/ bailey/applets/MindReader/index.html

If the computer could read the user's mind

Shannon's Mind Reading Machine

http://cs.williams.edu/ bailey/applets/MindReader/index.html

The 20Q game

$2^{20} \approx 10^6 > \#\text{words} \approx 10^5$

20Q.net Inc. http://www.20q.net/

Overview

Interactive Learning and Optimization in Search

Reinforcement Learning

Programming by Feedback

Interactive learning and optimization

Optimizing the coffee taste — Herdy et al., 96

Black box optimization:

$$\mathcal{F} : \Omega \rightarrow \mathbb{R} \qquad \text{Find arg max } \mathcal{F}$$

The user in the loop replaces $\mathcal{F}$

Optimizing visual rendering — Brochu et al., 07

Optimal recommendation sets — Viappiani & Boutilier, 10

Information retrieval — Shivaswamy & Joachims, 12

Interactive optimization

Features

- Search space $X \subset \mathbb{R}^d$ (recipe x: 33% arabica, 25% robusta, etc)
- hardly available features; unknown objective
- Expert emits preferences: $x \prec x'$.

Iterative scheme

1. At step t, Alg. generates candidates $x_t^{(1)}, x_t^{(2)}$
2. Expert emits preferences $x_t^{(1)} \succ x_t^{(2)}$
3. $t \rightarrow t+1$

Issues

- Asking as few questions as possible $\neq$ active ranking
- Modelling the expert's preference

  surrogate optimization objective
- Enforce the exploration vs exploitation trade-off

Optimal Bayesian Recommendation Sets

Boutilier Viappiani 2010

Notations

- Objects in a finite domain $Y \subset \{0, 1 \ldots\}^d$
- Generalized additive independent model $U(y) = \langle w, y \rangle$
- Belief $P(w, \theta)$

Algorithm

For $t = 1 \ldots T$ do
$*$ Propose a set $y_1 \ldots y_k$ (Selection criterion, see next)
$*$ Observe preferred $\bar{y}$
$*$ Update θ

Selection criterion

Expected utility of solution y

$$EU(y,\theta) = \int_W \langle w, y \rangle dP(w,\theta)$$

Maximum expected utility

$$EU^*(\theta) = max_y EU(y,\theta)$$

Selection Criterion: return solution with maximum

- Expected utility
- Maximum expected posterior utility given y^* the best solution so far

$$\begin{aligned} EPU(y,\theta) = \quad & Pr(y > y^*;\theta) EU^*(\theta | y > y^*) \\ & + Pr(y < y^*;\theta) EU^*(\theta | y < y^*) \end{aligned}$$

- Maximum expected utility of selection

$$\begin{aligned} EUS(y,\theta) = \quad & Pr(y > y^*;\theta) EU(y,\theta | y > y^*) \\ & + Pr(y < y^*;\theta) EU(y^*,\theta | y < y^*) \end{aligned}$$

Optimal Bayesian Recommendation Sets, 2

Comments

- Max. expected utility = greedy choice

- Max expected posterior utility: greedy with 1-step look-ahead (maximizes the expected utility of the solution found after the user will have expressed her preference). But computing $EPU(y)$ requires solving two optimization problems.

- Max expected utility of selection: limited loss of performance compared to max EPU; much less computationally expensive.

Co-active Learning

Shiwasvamy Joachims 2012

Context

Refining a search engine. Given query x, propose ordered list y.

Notations

- User utility $U(y|x)$
- Search space of linear models $U(y|x) = \langle w, \phi(x, y)\rangle$

Algorithm

For $t = 1 \ldots T$

∗ Given x_t, Propose $y_t = \text{argmax}_y \{\langle w_t, \phi(x_t, y)\rangle\}$

∗ Get feedback $\bar{y}_t$ from user (swapping items in y)

∗ Update utility model:

$$w_{t+1} = w_t + \phi(x_t, \bar{y}_t) - \phi(x_t, y_t)$$

Difference wrt multi-class perceptron

- Feedback: $\bar{y}_t$ is a rearrangement of y_t (not true label)
- Criterion: regret (not misclassification)

Interactive Intent Modelling

The vocabulary issue in human-machine interaction

Furnas et al. 87

- Single access term chosen by a single designer will provide very poor access:

TABLE I. Word-Object Data

(a) Sample data from the text-editing study

Words	Objects: "Insert"	"Delete"	"Replace"	"Move"	"Transpose"	...
Change	30	22	60	30	41	
Remove	0	21	12	17	5	
Spell	4	14	13	12	10	
Reverse	0	0	0	0	27	...
Leave	10	0	0	1	0	
Make into	0	4	0	0	1	
⋮	⋮	⋮	⋮	⋮	⋮	

- Humans are likely to use different vocabularies to encode and decode their intended meaning.

Two translation tasks

...not equally difficult

A From mother tongue to foreign language: one has to know vocabulary and grammar

B From foreign language to mother tongue: desambiguation from context, by guessing, etc

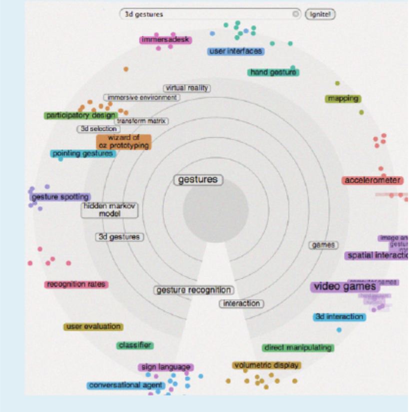
Search

- Writing a query: An *A*-task
- Assessing relevance: A *B*-task

Interactive Intent Modelling, 2

A human-in-loop approach   Ruotsalo et al. 15

- Show candidate documents
- Ask user's preferences
- Focus the query



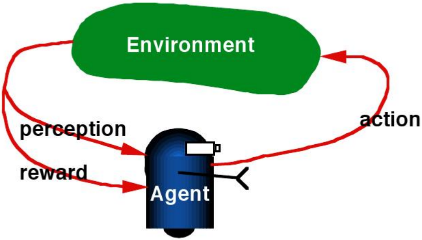
Overview

Interactive Learning and Optimization in Search

Reinforcement Learning

Programming by Feedback

Reinforcement Learning



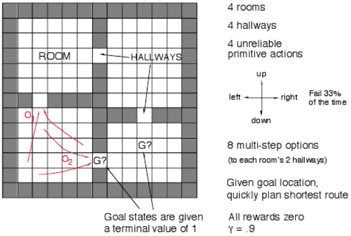
Generalities

- An agent, spatially and temporally situated
- Stochastic and uncertain environment
- Goal: select an action in each time step,
- ... in order maximize expected cumulative reward over a time horizon

What is learned ?

$$\text{A policy} = \text{strategy} = \{ \text{ state} \mapsto \text{action } \}$$

Reinforcement Learning, formal background



Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition $p(s, a, s') \mapsto [0, 1]$
- Reward $r(s)$
- Discount $0 < \gamma < 1$

Goal: a policy π mapping states onto actions

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

s.t.

$$\begin{aligned} \text{Maximize } E[\pi|s_0] &= \text{Expected discounted cumulative reward} \\ &= r(s_0) + \sum_t \gamma^{t+1} p(s_t, a = \pi(s_t), s_{t+1}) r(s_{t+1}) \end{aligned}$$

Reinforcement learning

Tasks (model-based RL)

- Learn value function
- Learn transition model
- Explore

Algorithmic & Learning issues

- Representation of the state/action space
- Approximation of the value function
- Scaling w.r.t. state-action space dimension
- Exploration / Exploitation

Expert's duty: design the reward function, s.t.

- optimum corresponds to desired behavior
- tractable (approximate) optimization.

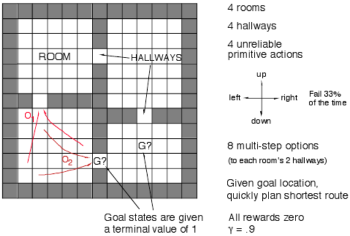
Designing the reward function

Sparse

- only reward on the treasure: a Needle in the Haystack optimization problem

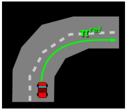
Informed

- Significant expertise (in the problem domain, in RL) required



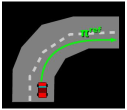
Using expert demonstrations

to train a classifier $s \to \pi(s)$

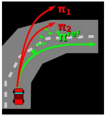


Using expert demonstrations

to train a classifier $s \rightarrow \pi(s)$



... yields brittle policies



Inverse Reinforcement Learning Russell Ng 00, Abbeel Ng 04

Infer the reward function explaining the expert behavior

Sidestepping numerical rewards

Medical prescription — Furnkranz et al., 2012

Avoid quantifying the cost of a fatal event: comparing the effects of actions.

$$s, a, \pi \prec s, a', \pi$$

Co-Active Learning — Shivaswamy Joachims, 15

The user responds by (slightly) improving the machine output.

Relaxing Expertise Requirements in RL

Expert

- Associates a reward to each state — RL
- Demonstrates a (nearly) optimal behavior — Inverse RL
- Compares and revises agent demonstrations — Co-Active L
- Compares demonstrations — Preference RL, PF

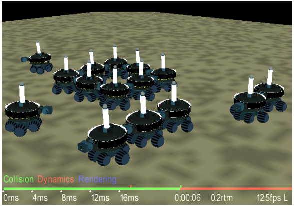
Ex-
per-
tise
↘

Agent

- Computes optimal policy based on rewards — RL
- Imitates verbatim expert's demonstration — IRL
- Imitates and modifies — IRL
- Learns the expert's utility — IRL, CAL
- Learns, and selects demonstrations — CAL, PRL, PF
- Accounts for the expert's mistakes — PF

Au-
ton-
omy
↗

Motivating application: Swarm Robotics



Swarm-bot (2001-2005)

Swarm Foraging, UWE

Symbrion IP, 2008-2013; http://symbrion.org/

Inverse RL not applicable: target individual behavior unknown.

Programming by feedback

Akrour et al. 14

Loop

1. Computer presents the expert with a pair of behaviors y_1, y_2
2. Expert emits preferences $y_1 \succ y_2$
3. Computer learns expert's utility function $\langle w, y \rangle$
4. Computer searches for behaviors with best utility

Key issues

- Asks few preference queries

Not active preference learning: Sequential model-based optimization

- Accounts for human noise

Human noise

Human beings often are

- irrational
- inconsistent
  - they make errors
  - they adapt themselves
  - they are kind...

Preferences often

- do no pre-exist
- are constructed on the fly

D. Kahneman, Thinking, fast and slow, 2011

Formal setting

$\mathcal{X}$ Search space, solution space — controllers, $\mathbb{R}^D$
$\mathcal{Y}$ Evaluation space, behavior space — trajectories, $\mathbb{R}^d$

$$\Phi : \mathcal{X} \mapsto \mathcal{Y}$$

Utility function

$$U^* \quad \mathcal{Y} \mapsto \mathbb{R} \quad U^*(\mathrm{y}) = \langle \mathbf{w}^*, \mathrm{y} \rangle$$ behavior space

Requisites

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

Programming by Feedback

Ingredients

- Learning the expert's utility
  to avoid asking too many preference queries
- Modelling the expert's competence
  to accommodate expert inconsistencies
- Selecting the next best behaviors to be demonstrated:
  - Which optimization criterion
  - How to optimize it

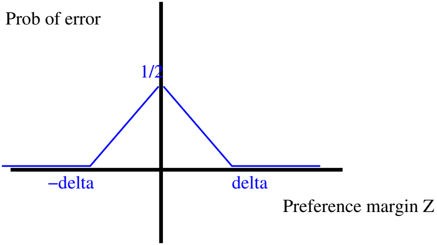
algorithmic details at the end

Modelling the expert's competence: Noise model

Given two solutions y and y′, for $\mathbf{w}^*$ the true utility

$$\text{Preference margin } z = \langle \mathbf{w}^*, \mathrm{y} - \mathrm{y}' \rangle$$

The probability of error is



- 0 if the absolute margin is $> $ threshold δ
- piecewise linear for $-\delta < z < \delta$.

Where δ is uniform in $[0, M]$ and M is the expert's inconsistence / incompetence

the lower, the most consistent the expert.

Experimental validation

- Sensitivity to expert competence
  Simulated expert, grid world

- Other benchmarks details at the end
  - Continuous case, no generative model
    The cartpole
  - Continuous case, generative model
    The bicycle
  - Training in-situ
    The Nao robot

The learner and the (simulated) human in the loop

Grid world: discrete case, no generative model

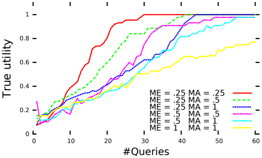
25 states, 5 actions, horizon 300, 50% transition motionless

The true $\mathbf{w}^*$

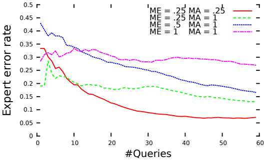
	...	1/4	1/2	1
			1/4	1/2
1/64				1/4
1/128	1/64			⋮
1/256	1/128	1/64		

Sensitivity study

M_E	Expert inconsistency
$M_A > M_E$	Computer estimate of expert's inconsistency



True utility of $\mathbf{x}_t$



expert's mistakes

The learner and the (simulated) human in the loop, 2

Findings

- The learner estimate M_A of the expert's inconsistency (M_E) does influence the number of mistakes done by the expert.
- No psychological effects though: this is a simulated expert.
- In the short run, a learner trusting a (mildly) incompetent expert does better than a learner distrusting a (more) competent expert.

Interpretation

- The higher M_A, the smoother the learned preference model, the more often the learner presents the expert with pairs of solutions with low margin;
- The lower the margin, the higher the mistake probability
- A cumulative (dis)advantage phenomenon

For low M_A, the computer learns faster, submits more relevant demonstrations to the expert, thus priming a virtuous educational process.

Partial conclusion

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.

CS + learning from the human in the loop

- No need to debug if you can just say: No !
  and the computer reacts (appropriately).
- I had a dream: a world where I don't need to read the manual.

Learning and Optimization with the Human in the Loop

Knowledge-constrained

Computation, memory-constrained

Bibliography

B. Akgun, K. Subramanian, J. Shim, and A. Lockerd Thomaz. Learning tasks and skills together from a human teacher. In W. Burgard and D. Roth, editors, *AAAI*. AAAI Press, 2011.

R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by feedback. In *ICML*, volume 32 of *JMLR Proceedings*, pages 1503–1511. JMLR.org, 2014.

E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS 20*, pages 409–416, 2008.

C. Furtlehner, M. Sebag, and Z. Xiangliang. Scaling Analysis of Affinity Propagation. *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, 81:066102, 2010.

R. Garnett, Y. Krishnamurthy, X. Xiong, J. G. Schneider, and R. Mann. Bayesian optimal active search and surveying. In *ICML*. Omnipress, 2012.

S. Gulwani. Automating string processing in spreadsheets using input-output examples. In T. Ball and M. Sagiv, editors, *POPL*, pages 317–330. ACM, 2011.

A. Jain, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *NIPS*, 2013.

W. B. Knox, P. Stone, and C. Breazeal. Training a robot via human feedback: A case study. In *Int. Conf. on Social Robotics*, volume 8239 of *LNCS*, pages 460–470. Springer, 2013.

Bibliography, 2

P. Liang, M. I. Jordan, and D. Klein. Learning programs: A hierarchical bayesian approach. In J. Fürnkranz and T. Joachims, editors, *ICML*, pages 639–646. Omnipress, 2010.

S. H. Muggleton and D. Lin. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In Francesca Rossi, editor, *Proc. 23rd IJCAI*. IJCAI/AAAI, 2013.

P.-Y. Oudeyer, A. Baranes, and F. Kaplan. Intrinsically motivated exploration for developmental and active sensorimotor learning. In *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 107–146. Springer Verlag, 2010.

A. Pease and S. Colton. Computational creativity theory: Inspirations behind the face and idea models. In *Proc. Intl Conf. on Computational Creativity*, pages 72–77, 2011.

P. Shivaswamy and T. Joachims. Online structured prediction via coactive learning. In *ICML*, 2012.

A. Tversky and D. Kahneman.

P. Viappiani and C. Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In J. D. Lafferty et al., editor, *NIPS 23*, pages 2352–2360. Curran Associates, Inc., 2010.

Bibliography, 3

A. Wilson, A. Fern, and P. Tadepalli. A Bayesian approach for policy learning from trajectory preference queries. In P. L. Bartlett et al., editor, *NIPS 25*, pages 1142–1150, 2012.

Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In A.P. Danyluk et al., editor, *Proc. 26th ICML*, volume 382 of *ACM Intl Conf. Proc. Series*, pages 1169–1176. ACM, 2009.

Xiangliang Zhang, Cyril Furtlehner, Cécile Germain-Renaud, and Michèle Sebag. Data stream clustering with affinity propagation. *IEEE Trans. Knowl. Data Eng.*, 26(7):1644–1656, 2014.

Programming by feedback

Akrour et al. 14

Algorithm

1. Learning the expert's utility function given the preference archive
2. Finding the best pair of demonstrations (y, y') (expected posterior utility under the noise model)
3. Achieving optimization in demonstration space (e.g. trajectory space)
4. Achieving optimization in solution space (e.g. neural net)

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 θ_t posterior on W $\qquad$ W = linear fns on $\mathcal{Y}$

Proposition: Given $\mathcal{U}_t$,

$$\begin{array}{rcl} \theta_t(\mathbf{w}) & \propto & \prod_{i=1,t} P(y_{i_1} \succ y_{i_2} \mid \mathbf{w}) \\ & = & \prod_{i=1,t} \left(\frac{1}{2} + \frac{\mathbf{w}_i}{2M} \left(1 + \log \frac{M}{|\mathbf{w}_i|} \right) \right) \end{array}$$

with $\mathbf{w}_i = \langle \mathbf{w}, y_{i_1} - y_{i_2} \rangle$, capped to $[-M, M]$.

$$U_t(\mathrm{y}) = \mathbb{E}_{w \sim \theta_t}[\langle \mathbf{w}, \mathrm{y} \rangle]$$

Best demonstration pair (y, y')

after Viappiani Boutilier, 10

EUS: Expected utility of selection (greedy)

$$\begin{aligned} EUS(\mathrm{y}, \mathrm{y}') &= \quad \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle > 0] \;.\; U_{w \sim \theta_t, y > y'}(\mathrm{y}) \\ &+ \; \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle < 0] \;.\; U_{w \sim \theta_t, y < y'}(\mathrm{y}') \end{aligned}$$

EPU: Expected posterior utility (lookahead)

$$\begin{aligned} EPU(\mathrm{y}, \mathrm{y}') &= \quad \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle > 0] \;.\; max_{y''} U_{w \sim \theta_t, y > y'}(\mathrm{y}'') \\ &+ \; \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle < 0] \;.\; max_{y''} U_{w \sim \theta_t, y < y'}(\mathrm{y}'') \\ &= \quad \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle > 0] \;.\; U_{w \sim \theta_t, y > y'}(\mathrm{y}^*) \\ &+ \; \mathbb{E}_{\theta_t}[\langle \mathbf{w}, y - y' \rangle < 0] \;.\; U_{w \sim \theta_t, y < y'}(\mathrm{y}'^*) \end{aligned}$$

Therefore

$$\text{argmax } EPU(\mathrm{y}, \mathrm{y}') \leq \text{argmax } EUS(\mathrm{y}, \mathrm{y}')$$

Optimization in demonstration space

NL: noiseless N: noisy

Proposition

$$EUS^{NL}(\mathrm{y}, \mathrm{y}') - L \leq EUS^{N}(\mathrm{y}, \mathrm{y}') \leq EUS^{NL}(\mathrm{y}, \mathrm{y}')$$

Proposition

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

Limited loss incurred $(L \sim \frac{M}{20})$

Optimization in solution space

1. Find best $y, y' \rightarrow$ Find best y

to be compared to best behavior so far y^*_t

The game of hot and cold

2. Expectation of behavior utility $\rightarrow$ utility of expected behavior

Given the mapping Φ: 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 $\mathbf{w}_0 \sim \theta_t$ and let $\mathbf{x}_1 = \operatorname{argmax} \{\langle \mathbf{w}_0, \mathbb{E}_\Phi[\Phi(\mathbf{x})]\rangle\}$
- Iteratively, find $\mathbf{x}_{i+1} = \operatorname{argmax} \{\langle \mathbb{E}_{\theta_i}[\mathbf{w}], \mathbb{E}_\Phi[\Phi(\mathbf{x})]\rangle\}$, with θ_i posterior to $\mathbb{E}_\Phi[\Phi(\mathbf{x}_i)] > y^*_t$.

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

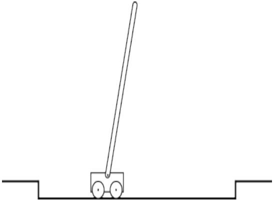
Experimental validation of Programming by Feedback

Continuous Case, no Generative Model

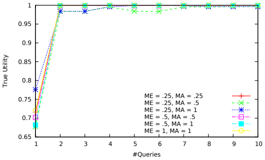
The cartpole

State space $\mathbb{R}^2$, 3 actions

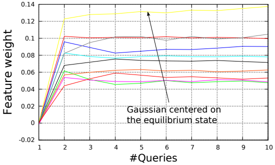
Dem. space $\mathbb{R}^9$, dem. length 3,000



Cartpole



True utility of $\mathbf{x}_t$ fraction in equilibrium



Estimated utility of features

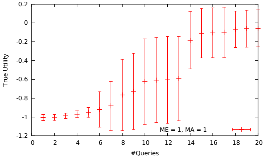
Two interactions required on average to solve the cartpole problem.

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

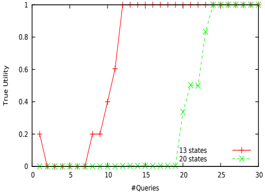


Optimization component: CMA-ES Hansen et al., 2001
15 interactions required on average to solve the problem for low noise.
versus 20 queries, with discrete action in state of the art.

Training *in-situ*

The Nao

The Nao robot



Nao: true utility of $\mathbf{x}_t$

Goal: reaching a given state.
Transition matrix estimated from 1,000 random (s, a, s') triplets.
Dem. length 10, fixed initial state.
12 interactions for 13 states
25 interactions for 20 states