Learning with the human in the loop

Michèle Sebag

Riad Akrour

Marc Schoenauer

TAO

Constructive Machine Learning Wshop, ICML 2015
<table>
<thead>
<tr>
<th>Era</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970s</td>
<td>Specifications Languages &amp; thm proving</td>
</tr>
<tr>
<td>1990s</td>
<td>Programming by Examples Pattern recognition &amp; ML</td>
</tr>
<tr>
<td>2010s</td>
<td>Interactive Learning and Optimization</td>
</tr>
</tbody>
</table>
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
- under-specified goal

$P(x)$ changes
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

Think of something and 20Q will read your mind by asking a few simple questions. The object you think of should be something that most people would know about, but not a proper noun or a specific person, place, or thing. Click the ? in the upper right corner for help.

Q1. Is it classified as Animal, Vegetable or Mineral?
Animal, Vegetable, Mineral, Concept, Unknown

Suggestions

If you would like some suggestions of what to think about, 20Q recommends the following:

Some things 20Q has chosen at random . . .
- jacks (child’s game),
- talcum powder,
- anmitsu (bean paste with honey),
- a hot tub,
- an apricot.

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
Black box optimization:

\[ \mathcal{F} : \Omega \rightarrow \mathbb{R} \quad \text{Find} \ \arg \max \mathcal{F} \]

The user in the loop replaces \( \mathcal{F} \)

Optimizing visual rendering

Optimal recommendation sets

Information retrieval

Herdy et al., 96

Brochu et al., 07

Viappiani & Boutilier, 10

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 $x_t^{(1)} \succeq x_t^{(2)}$
3. $t \rightarrow t + 1$

Issues
- Asking as few questions as possible $\neq$ active ranking
- Modelling the expert’s preference
- Enforce the exploration vs exploitation trade-off
Optimal Bayesian Recommendation Sets

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 $\theta$
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

$$EPU(y, \theta) = Pr(y > y^*; \theta)EU^*(\theta|y > y^*) + Pr(y < y^*; \theta)EU^*(\theta|y < y^*)$$

- Maximum expected utility of selection

$$EUS(y, \theta) = Pr(y > y^*; \theta)EU(y, \theta|y > y^*) + Pr(y < y^*; \theta)EU(y^*, \theta|y < y^*)$$
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 \( y_t = \arg\max_y \{ \langle w_t, \phi(x_t, y) \rangle \} \)
- Get feedback \( \tilde{y}_t \) from user (swapping items in \( y \))
- Update utility model:
  \[
  w_{t+1} = w_t + \phi(x_t, \tilde{y}_t) - \phi(x_t, y_t)
  \]

Difference wrt multi-class perceptron
- Feedback: \( \tilde{y}_t \) is a rearrangement of \( y_t \) (not true label)
- Criterion: regret (not misclassification)
Interactive Intent Modelling

The vocabulary issue in human-machine interaction

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

![Table 1: Word-Object Data](image)

- 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

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

Ruotsalo et al. 15
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?

A policy = strategy = \{ state \mapsto 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 $\pi$ mapping states onto actions

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

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

Goal states are given a terminal value of 1

<table>
<thead>
<tr>
<th>4 rooms</th>
<th>4 hallways</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 unreliable primitive actions</td>
<td></td>
</tr>
</tbody>
</table>

8 multi-step options (to each room's 2 hallways)

Given goal location, quickly plan shortest route

All rewards zero, $\gamma = .8$
Using expert demonstrations to train a classifier $s \rightarrow \pi(s)$
Using expert demonstrations

to train a classifier $s \rightarrow \pi(s)$ ... yields brittle policies

Inverse Reinforcement Learning

Infer the reward function explaining the expert behavior

Russell Ng 00, Abbeel Ng 04
Sidestepping numerical rewards

Medical prescription

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

\[ s, a, \pi \prec s, a', \pi \]

Co-Active Learning

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 \( \text{Inverse RL} \)
- Compares and revises agent demonstrations \( \text{Co-Active L} \)
- Compares demonstrations \( \text{Preference RL, PF} \)

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 \( \text{IRL, CAL} \)
- Learns, and selects demonstrations \( \text{CAL, PRL, PF} \)
- Accounts for the expert’s mistakes \( \text{PF} \)
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
\( \mathcal{Y} \) Evaluation space, behavior space

controllers, \( \mathbb{R}^D \)
trajectories, \( \mathbb{R}^d \)

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

Utility function

\[ U^* : \mathcal{Y} \mapsto \mathbb{R} \quad U^*(y) = \langle w^*, y \rangle \]

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

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

The probability of error is

$\begin{cases} 
0 & \text{if the absolute margin is } > \text{threshold } \delta \\
\text{piecewise linear for } -\delta < z < \delta. 
\end{cases}$

Where $\delta$ 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 $w^*$

Sensitivity study
$M_E$ Expert inconsistency
$M_A > M_E$ Computer estimate of expert’s inconsistency

True utility of $x_t$ 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


A. Tversky and D. Kahneman.


Programming by feedback

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)

Akrour et al. 14
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 \( \mathcal{W} \)

\( \mathcal{W} = \) linear fns on \( \mathcal{Y} \)

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

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

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 \((y, y')\)

after Viappiani Boutilier, 10

EUS: Expected utility of selection

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

EPU: Expected posterior utility

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

Therefore

\[
\text{argmax } EPU(y, y') \leq \text{argmax } EUS(y, y')
\]
Optimization in demonstration space

NL: noiseless

N: noisy

Proposition

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

Proposition

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

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 $\Phi$: 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 $x_1 = \arg\max \\{\langle w_0, \mathbb{E}_\Phi[\Phi(x)] \rangle\}$
   ▶ Iteratively, find $x_{i+1} = \arg\max \\{\langle \mathbb{E}_{\theta_i}[w], \mathbb{E}_\Phi[\Phi(x)] \rangle\}$, 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 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

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 robot

**Nao:** true utility of $x_t$

**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