Machine learning and the expert in the loop

Michèle Sebag

TAO

ECAI 2014, Frontiers of AI
Computing Machinery and Intelligence

... the problem is mainly one of programming.

brain estimates: $10^{10}$ to $10^{15}$ bits

I can produce about a thousand digits of program lines a day

[Therefore] more expenditious method seems desirable.

⇒ Machine Learning
ML envisioned by Alan Turing

The process of creating a mind

- Initial state [the innate]  ML expert
- Education [environment, teacher]  Domain expert
- Other

The teaching process

... We normally associate punishments and rewards with the teaching process...

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.

This talk: formulating the Pleasure-and-Pain ML agenda
Overview

Preamble

Machine Learning: All you need is...
  ...logic
  ...data
  ...optimization
  ...rewards

All you need is expert’s feedback
  Interactive optimization
  Programming by Feedback

Programming, An AI Frontier
ML: All you need is logic

Perception → Symbols → Reasoning → Symbols → Actions

Let’s forget about perception and actions for a while...

Symbols → Reasoning → Symbols

Requisite

- Strong representation
- Strong background knowledge
- [ Strong optimization tool ] cf F. Fages if numerical parameters involved
Principle: generate hypotheses from background knowledge and experimental data, design experiments to confirm/infirm hypotheses

Adam: drug screening, hit conformation, and cycles of QSAR hypothesis learning and testing.

Eve: — applied to orphan diseases.
So efficient

- Search: Reuse constraint solving, graph pruning, ...

Requirement / Limitations

- Initial conditions: critical mass of high-order knowledge
- ... and unified search space
- Symbol grounding, noise

Of primary value: intelligibility

- A means: for debugging
- An end: to keep the expert involved.
Overview

Preamble

**Machine Learning: All you need is...**

...logic
...data
...optimization
...rewards

All you need is expert’s feedback
Interactive optimization
Programming by Feedback

Programming, An AI Frontier
ML: All you need is data

Old times: datasets were rare

- Are we overfitting the Irvine repository?
- [ current: Are we overfitting MNIST? ]

Fig. 4. Size-normalized examples from the MNIST database.

The drosophila of AI
ML: All you need is data

Now

- Sky is the limit!
- Logic $\rightarrow$ Compression
- Compression $\rightarrow$ symbols, distribution

Markus Hutter, 2004
Big data

IBM Watson defeats human champions at the quiz game Jeopardy

- $i$, 1000$^i$, kilo, mega, giga, tera, peta, exa, zetta, yotta, bytes

- Google: 24 petabytes/day
- Facebook: 10 terabytes/day; Twitter: 7 terabytes/day
- Large Hadron Collider: 40 terabytes/seconds
The Higgs boson ML Challenge

Balazs Kégl, Cécile Germain et al.

https://www.kaggle.com/c/higgs-boson September 2014, 15th
The ATLAS detector
An event in the ATLAS detector
The data

- Hundreds of millions of proton-proton collisions per second
  - hundreds of particles: decay products
  - hundreds of thousands of sensors (but sparse)
  - for each particle: type, energy, direction is measured
  - a fixed list of $\sim 30-40$ extracted features:
    \[ x \in \mathbb{R}^d \]
  - e.g., angles, energies, directions, number of particles
  - discriminating between signal (the particle we are looking for) and background (known particles)

- Filtered down to 400 events per second, still petabytes per year
  - real-time (budgeted) classification – a research theme on its own
  - cascades, cost-sensitive sequential learning
The analysis

• Highly unbalanced data:
  • in the $H \rightarrow \tau\tau$ channel we expect to see $< 100$ Higgs bosons per year in $400 \times 60 \times 60 \times 24 \times 356 \approx 10^{10}$ events
  • after pre-selection, we will have 500K background (negative) and 1K signal (positive) events

• The goal is not classification but discovery
  • a classifier is used to define a (usually tiny) selection region in $\mathbb{R}^d$
  • a counting test is used to determine whether the number of observed events selection region exceeds significantly the expected number of events predicted by an only-background hypothesis
Overview

Preamble

Machine Learning: All you need is...
...logic
...data
...optimization
...rewards

All you need is expert’s feedback
Interactive optimization
Programming by Feedback

Programming, An AI Frontier
Old times

- Find the best hypothesis
- Find the best optimization criterion
  - statistically sound
  - such that it defines a well-posed optimization problem
  - tractable
Episode 1

- NNs are universal approximators,…
- … but their training yields non-convex optimization problems
- … and some cannot reproduce the results of some others…

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.
Episode 2

At last, SVMs arrive!

Principle

- Min $||h||^2$
- subject to constraints on $h(x)$ (modelling data)
  
  $h(x_i) . y_i > 1$, $|h(x_i) - y_i| < \epsilon$, $h(x_i) < h(x'_i)$, $h(x_i) > 1$...

  classification, regression, ranking, distribution, ...

Convex optimization! (well, except for hyper-parameters)

More sophisticated optimization (alternate, upper bounds)...

Boyd & Vandenberghe 04; Bach 04; Nesterov 07; Friedman & al. 07; ...
SVMs and Deep Learning

Episode 3

▶ Did you forget our AI goal? (learning ↔ learning representation)
▶ At last Deep learning arrives!

Principle

▶ We always knew that many-layered NNs offered compact representations

\[ 2^n \text{ neurons on 1 layer} \quad \text{versus} \quad n \text{ neurons on log } n \text{ layers} \]
SVMs and Deep Learning

Episode 3

▶ Did you forget our AI goal? (learning ↔ learning representation)
▶ At last Deep learning arrives!

Principle

▶ We always knew that many-layered NNs offered compact representations
▶ But, so many poor local optima!
Episode 3

▶ Did you forget our AI goal?
   (learning ↔ learning representation)
▶ At last Deep learning arrives!

Principle

▶ We always knew that many-layered NNs offered compact representations
   Hasted 87
▶ But, so many poor local optima!
▶ Breakthrough: unsupervised layer-wise learning
   Hinton 06; Bengio 06
SVMs and Deep Learning

From prototypes to features

- $n$ prototypes $\rightarrow n$ regions
- $n$ features $\rightarrow 2^n$ regions
SVMs and Deep Learning

Last Deep news

- Supervised training works, after all  
  Glorot Bengio 10
- Does not need to be deep, after all
  Ciresan et al. 13, Caruana 13
SVMs and Deep Learning

Last Deep news

- Supervised training works, after all  
  Glorot Bengio 10
- Does not need to be deep, after all  
  Ciresan et al. 13, Caruana 13
  - Ciresan et al: use prior knowledge (non linear invariance operators) to generate new examples
  - Caruana: use deep NN to label hosts of examples; use them to train a shallow NN.
SVMs and Deep Learning

Last Deep news

▶ Supervised training works, after all
▶ Does not need to be deep, after all

Glorot Bengio 10

Ciresan et al. 13, Caruana 13

▶ SVMers’ view: the deep thing is linear learning complexity

Take home message

▶ It works
▶ But why?
▶ Intelligibility?
SVMs and Deep Learning

Last Deep news

▶ Supervised training works, after all  
   Glorot Bengio 10
▶ Does not need to be deep, after all  
   Ciresan et al. 13, Caruana 13
▶ SVMers’ view: the deep thing is **linear learning complexity**

Take home message

▶ It works
▶ But why ?
▶ Intelligibility ?

no doubt you recognize a cat  
Le &al. 12
Overview

Preamble

Machine Learning: All you need is...
   ...logic
   ...data
   ...optimization
   ...rewards

All you need is expert’s feedback
   Interactive optimization
   Programming by Feedback

Programming, An AI Frontier
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** $S$
- **Action space** $A$
- **Transition** $p(s, a, s') \mapsto [0, 1]$
- **Reward** $r(s)$
- **Discount** $0 < \gamma < 1$

Goal: a policy $\pi$ mapping states onto actions

$$\pi : S \mapsto A$$

s.t.

Maximize $E[\pi | s_0] = \text{Expected \ discounted \ cumulative \ reward}$

$$= r(s_0) + \sum_t \gamma^{t+1} p(s_t, a = \pi(s_t), s_{t+1})r(s_{t+1})$$
Find the treasure

Single reward: on the treasure.
Wandering robot

Nothing happens...
The robot finds it
Robot updates its value function

\[ V(s, a) \equiv \text{“distance“ to the treasure on the trajectory.} \]
Reinforcement learning

* Robot most often selects $a = \text{arg max } V(s, a)$
* and sometimes explores (selects another action).
* Lucky exploration: finds the treasure again
Updates the value function

* Value function tells how far you are from the treasure given the known trajectories.
* Value function tells how far you are from the treasure
Finally

Let’s be greedy: selects the action maximizing the value function
Reinforcement learning

Three interdependent tasks
- Learn value function
- Learn transition model
- Explore the world

Issues
- Exploration / Exploitation dilemma
- Representation, approximation, scaling up
- REWARDS designer’s duty
Reinforcement learning and the expert

Input needed from expert

▶ A reward function
  standard RL
  Sutton & Barto 08; Szepesvári 10

▶ An expert demonstrating an “optimal“ behavior
  inverse RL
  Abbeel & al. 04-12; Billard & al. 05-13
  Lagoudakis & Parr 03; Konaridis & al. 10

▶ A reliable teacher
  preference-based RL
  Akrou & al. 11, Wilson & al. 12, Knox & al. 13, Saxena & al. 13

▶ A teacher
  Akrou & al. 14
Strong expert jumps in and demonstrates behavior

Inverse reinforcement learning

From demonstration to rewards
From \((s_t, a_t, s_{t+1})\), learn a reward function \(r\) s.t.

\[
Q(s_i, a_i) \geq Q(s_i, a) + 1, \forall a \neq a_i
\]

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

Then apply standard RL
Inverse Reinforcement Learning

issues

▶ An informed representation (speed; bumping in a pedestrian)
▶ A strong expert  

Kolter et al. 07; Abbeel 08
Inverse Reinforcement Learning

Issues

- An informed representation (speed; bumping in a pedestrian)
- A strong expert

Kolter et al. 07; Abbeel 08

In some cases there is no expert

Swarm-bot (2001-2005)

Swarm Foraging, UWE

Symbion IP, 2008-2013; http://symbion.org/
Expert jumps in and provides preferences...

Cheng et al. 11, Furnkranz et al. 12

Context

- Medical prescription
- What is the cost of a death event?

Approach

\[ a \prec_{\pi,s} a' \text{ iff following } \pi \text{ after } (s, a) \text{ is better than after } (s, a') \]
Overview

Preamble

Machine Learning: All you need is...
  ...logic
  ...data
  ...optimization
  ...rewards

All you need is expert’s feedback
  Interactive optimization
  Programming by Feedback

Programming, An AI Frontier
The 20 Question game

- First a society game
  19th century

- Then a radio game
  end 40s, 50s

- Then an AI game
  Burgener, 88
  http://www.20q.net/

20 questions → 20 bits → discriminates among $2^{20} \approx 10^6$ words.
Interactive optimization

Optimizing the coffee taste

Black box optimization:

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

The user in the loop replaces \( F \)

Herdy et al., 96
Interactive optimization

Optimizing the coffee taste
Black box optimization:

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

The user in the loop replaces \( F \)

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)
- A non-computable objective
- Expert can (by tasting) emit preferences $x \prec x'$.

Scheme

1. Alg. generates candidates $x, x', x''$, ...
2. Expert emits preferences
3. goto 1.

Issues

- Asking as few questions as possible $\neq$ active ranking
- Modelling the expert’s taste surrogate model
- Enforce the exploration vs exploitation trade-off
Overview

Preamble

Machine Learning: All you need is...
  ...logic
  ...data
  ...optimization
  ...rewards

All you need is expert’s feedback
  Interactive optimization
  Programming by Feedback

Programming, An AI Frontier
Programming by feedback

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 trend

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

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

Critical issues

- Asks few preference queries
- Not active preference learning: Sequential model-based optimization
  
  cf H. Hoos’ talk

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

\( \mathcal{X} \) Search space, solution space
\( \mathcal{Y} \) 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
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 $z = \langle \mathbf{w}^*, y - y' \rangle$

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

$\delta \sim U[0, M]$
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 $\mathcal{W}$ \hspace{1cm} $\mathcal{W} =$ 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 \((y, y')\) 

**EUS: Expected utility of selection**

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

**EPU: Expected posterior utility**

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

Therefore

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

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

Proposition

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

Proposition

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

Limited loss incurred

\[ (L \sim \frac{M}{20}) \]
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 \( \Phi: \text{search} \mapsto \text{demonstration space} \),

\[
\mathbb{E}_\Phi [EUS^{NL}(\Phi(x), y^*_t)] \geq EUS^{NL}(\mathbb{E}_\Phi [\Phi(x)], y^*_t)
\]

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

- 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 incompetence

$M_A > M_E$  
Computer estimate of expert’s incompetence

<table>
<thead>
<tr>
<th></th>
<th>$\ldots$</th>
<th>$1/4$</th>
<th>$1/2$</th>
<th>$1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1/64$</td>
<td>1/4</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1/128$</td>
<td>1/64</td>
<td></td>
<td>1/4</td>
<td></td>
</tr>
<tr>
<td>$1/256$</td>
<td>1/128</td>
<td>1/64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

True $w^*$ on gridworld
Sensitivity to (simulated) expert incompetence, 2

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.
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 $x_t$

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
Overview

Preamble

Machine Learning: All you need is...
  ...logic
  ...data
  ...optimization
  ...rewards

All you need is expert’s feedback
  Interactive optimization
  Programming by Feedback

Programming, An AI Frontier
Partial Conclusion

Feasibility of Programming by Feedback for simple tasks

Back on track:

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.

Expert says: it's better
Expert says: it's worse

pleasure
pain
Revisiting 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

toward Programming by ML & Optimization
Programming by Feedback

About the numerical divide

C.A. : *Once things can be done on desktops they can be done by anyone.*

?? : Well ... not everyone is a digital native..

About interaction

as designer No need to debug if you can just say: No! and the computer reacts (appropriately).

as user I had a dream: a world where I don’t need to read the manual...
Future: Tackling the Under-Specified

Knowledge-constrained

Computation, memory-constrained
Acknowledgments

Riad Akrour
Marc Schoenauer
Alexandre Constantin
Jean-Christophe Souplet