Optimization of a sequential decision problem in prenatal ultrasound

E. Le Pennec
R. Besson - S. Allassonnière

Agro - 07/10/2019

Outline

Birth and Rare Diseases

A Few Numbers

- 780.000 births/year in France, 5 millions births/year in Europe
- 3 to 4% are affected by at least one congenital abnomalies
- Rare diseases: 3 millions patients in France, 30 millions in Europe.

Medical Setting

Prenatal Ultrasound Diagnosis

- **France:** three compulsory ultrasound tests during pregnancy.
- Some classical measures (e.g. Down syndrome).
- **No strict examination protocol.**

Necker Hospital Obstetrician

- Rare disease expertise.
- Among world largest medical database.
- Will to **systematize** their knowledge.

Proposed Tool

Ultrasound as a Sequential Process

- Ultrasound exam seen as a **sequence of measures**.
- **Goals:**
  - **Reduce the time** required to obtain a diagnosis
  - **Avoid to miss** a rare disease.

Diagnosis Assistance Tool

- **Propose** the next measure to make.
- **Show** the current most probable diseases.
- **Easy to use GUI** implemented in **R**!

What's inside this tool?

Charade

Ultrasound Diagnostic

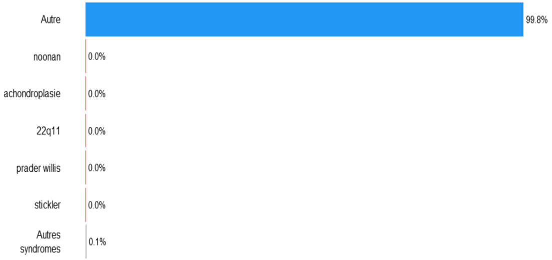
Reset Page

Choix d'une anomalie

ventricular septal defect (0.0361%)

not present | present

Historique



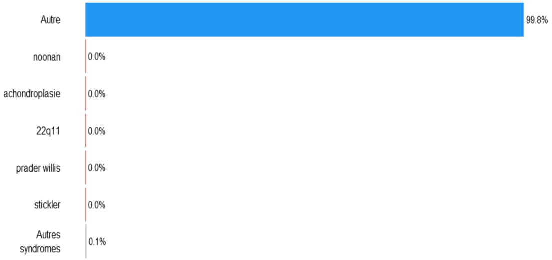
Reset Page

Choix d'une anomalie

ventricular septal defect (0.0361%)

not present present

Historique



Outline

Data at Hands

id disease	id symptom	probability of symptom knowing the disease
16	29	0.39
16	136	0.67
16	149	0.50
16	176	0.16
16	181	0.50
16	231	0.75

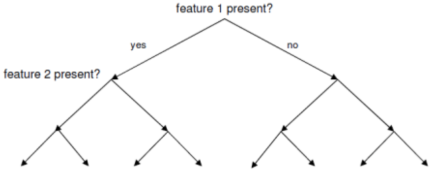
- Rare diseases: very few cases even in the world largest DB!

Excel Type Dataset

- Expert database build from OrphaData (E. Spaggiari).
- 81 diseases, 202 symptoms (signs visible with ultrasound):
  - Disease probability: $P[D = d_j]$
  - Symptom probability given each disease: $P[S_i = k \mid D = d_j]$.

- Database will be enriched from the future exams.

Our Goals



Medical Goals

- Guide a (non rare disease expert) sonographer to assess as fast as possible potential diseases.
- Propose her/him the next symptom to check.

Technical Goals

- Build a *good* decision tree (a *good* policy).
- Develop a GUI that can be easily used.

Markov Decision Process

State, Action and Policy

- State: $\mathbb{S} = \{P, A, U\}^{202}$ (presence,absence,not yet looked at) for each symptom.
- Action: $\mathbb{A} = \{1, \dots, 202\}$ next symptom.
- Policy: $\pi : s \in \mathbb{S} \mapsto a \in \mathbb{A}$ next symptom given the state.

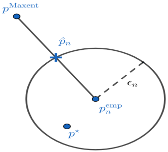
Probabilistic setting

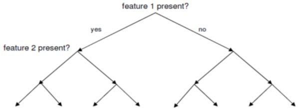
- Natural Markovian modeling: $\mathcal{S}_{t+1}$ depends only on $\mathcal{S}_t$ and a_t!

Markovian Decision Process

- Any strategy π defines a law on $(\mathcal{S}_t)$ starting from $\mathcal{S}_0$.
- Let T be the stopping time before a diagnosis can be posed.
- We need to find $\pi^\star$ such that $\pi^\star(\mathcal{S}_0) = \text{argmin}_\pi \, \mathbb{E}[T|\mathcal{S}_0]$!

Problems to be Solved





Environment Learning with Maximum Entropy Principle

- We have $P[S_i \mid D]$ but we need to know $P[S_{i_1}, \ldots, S_{i_K} \mid D]$.
- We need to take into account future exams.
- **Idea:** add some expert knowledge and maximize uncertainty, interpolate between the expert model and the data.

- Yields a simulator rather than the MDP transition proba...

Diagnostic Strategy Optimization by Reinforcement Learning

- Find a policy that allows to detect the disease while minimizing the average duration.
- **Idea:** recast the problem as a planning issue and find the optimal strategy.

Diagnostic Strategy Optimization

Diagnostic Strategy Optimization.

- Find a policy that allows to detect the disease while minimizing the average duration.

Measure of Performance

- Number of questions before being able to diagnose a disease.

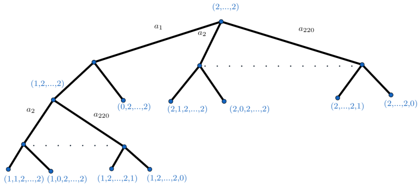
Alternative Formulations

- Trade-off: cost of misdiagnosis/cost of medical tests to perform.
- Reach the lowest uncertainty under fixed budget constraint (time, money).

Non Adversarial Game

- The disease and symptoms do not change during the exam.
- Strategy: given what has been seen, what is the next symptom to look at?

Stochastic Shortest Path



Stochastic Shortest Path

- T is a stooping time at some *final states*
- How to minimize the expectation of T?

Final States

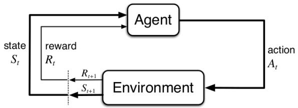
- Entropy based criterion: $H(D \mid \mathcal{S}) \leq \epsilon$

MDP

- Rewards: $\forall \mathcal{S}_t, a_t,\ r(\mathcal{S}_t, a_t) = -1$

Outline

Reinforcement Learning



Reinforcement Learning Setting

- **Env.:** provides a reward and a new state for any action.
- **Agent policy π:** choice of an action A_t from the state S_t.
- **Total reward:** (discounted) sum of the rewards.

Questions

- **Policy evaluation:** how to evaluate the expected reward of a policy knowing the environment?
- **Planning:** how to find the best policy knowing the environment?
- **Reinforcement Learning:** how to find the best policy without knowing the environment?

Outline

The Agent-Environment Interface

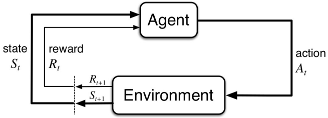


Figure 3.1: The agent–environment interaction in a Markov decision process.

MDP

- At time step $t \in \mathcal{N}$:
  - State $S_t \in \mathcal{S}$: representation of the environment
  - Action $A_t \in \mathcal{A}(S_t)$: action chosen
  - Reward $R_{t+1} \in \mathcal{R}$: instantaneous reward
  - New state S_{t+1}
- Dynamic entirely defined by

$$\mathbb{P}\left(S_t = s', R_r = r \middle| S_{t-1} = s, A_{t-1} = a\right) = p(s', r|s, a)$$

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite.

Returns ans Episodes

Return

- (Discounted) Return:

$$G_t = \sum_{t'=t+1}^{T} \gamma^{t'} R_{t'}$$

- Recursive property

$$G_t = R_{t+1} + \gamma G_{t+1}$$

- Finiteness if $|R| \leq M$

$$|G_t| \leq \begin{cases} (T - t - +1)M & \text{if } T < \infty \\ M\frac{1}{1-\gamma} & \text{otherwise} \end{cases}$$

- Not well defined if $T = \infty$ and $\gamma = 1$.

Policies and Value Functions

Policy and Value Functions

- Policy: $\pi(a|s)$
- Value function:
$$v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$
- Action value function:
$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right]$$

Two natural problems

- Policy evaluation: compute v_π given π.
- Planning: find $\pi^\star$ such that $v_{\pi^\star}(s) \geq v_\pi(s)$ for all s and π.

- Those objects may not exist in general!
- Can be traced back to the 50's!

Outline

Policy Evaluation by Bellman Backup

Fixed Point Property

- Bellman Equation
  $$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma v_\pi(s')\right] = \mathcal{T}_\pi(v_\pi)(s)$$

- Linear equation that can be solved.

Policy Evaluation by Dynamic Programming

- Fixed point iterative algorithm: $v_{k+1}(s) = \mathcal{T}_\pi(v_k)(s)$

- Converge if $T < \infty$ or $\gamma < 1$.

Planning by Policy Improvement

Policy Improvement Property

- If π' is such that $\forall s, q_\pi(s, \pi'(s)) \geq v_\pi(s)$ then $v_{\pi'} \geq v_\pi$.

- ϵ-greedy improvement among ϵ-policy: classical improvement degraded by picking uniformly the action with probability ϵ

Policy Iteration Algorithm

- Compute v_{π_k}
- Greedy update:

$$
\begin{aligned}
\pi_{k+1}(s) &= \operatorname*{argmax}_a q_{\pi_k}(s, a) \\
&= \operatorname*{argmax}_a \sum_{s', r} p(s', r | s, a) \left(r + \gamma v_{\pi_k}(s') \right)
\end{aligned}
$$

- If $\pi' = \pi$ after a greedy update $v_{\pi_{k+1}} = v_{\pi_k} = v_*$.

- Convergence in finite time in the finite setting.

Planning by Bellman Backup

Fixed Point Property

- Bellman Equation
$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma v_*(s')\right] = \mathcal{T}_*(v_*)(s)$$

- Linear programming problem that can be solved.

Policy Evaluation by Dynamic Programming

- Iterative algorithm: $v_{k+1}(s) = \mathcal{T}_*(v_k)(s)$

- Converge if $T < \infty$ or $\gamma < 1$.
- Amount to improve the policy after only one step of policy evaluation.

Planning by Bellman Backup

Q-value and enhancement

- Q-value:

$$q_\pi(s,a) = \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right]$$

- Easy policy enhancement: $\pi'(s) = \operatorname*{argmax}_a q(s,a)$

Fixed Point Property

- Bellman Equation

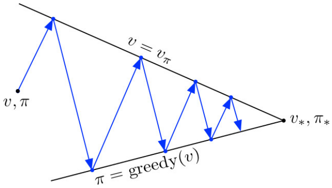
$$q_*(s,a) = \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] = \mathcal{T}_*(q_*)(s,a)$$

- Linear programming problem that can be solved.

Policy Evaluation by Dynamic Programming

- Iterative algorithm: $q_{k+1}(s,a) = \mathcal{T}_*(q_k)(s,a)$

Generalized Policy Iteration



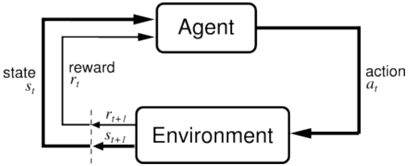
Generalized Policy Iteration

- Consists of two simultaneous interacting processes:
  - one making a value function consistent with the current policy (policy evaluation)
  - one making the policy greedy with respect to the current value function (policy improvement)

- Stabilizes only if one reaches the optimal value/policy pair.
- Asynchronous update are possible provided every state(/action) is visited infinitely often.
- Very efficient but requires the knowledge of the transition probabilities.

Outline

Reinforcement Learning



Reinforcement Learning - Sutton (98)

- An agent takes actions in a sequential way, receives rewards from the environment and tries to maximize his long-term (cumulative) reward.

Reinforcement Learning

- MDP setting with cumulative reward.
- Planning problem.
- Environment known only through interaction, i.e. some sequences $\cdots S_t A_t R_{t+1} S_{t+1} A_{t+1} \cdots$.

Monte Carlo

MC Methods

- Back to $v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right]$.
- Monte Carlo:
  - Play several episodes using policy π.
  - Average the returns obtained after any state s.

- Good theoretical properties provided every states are visited asymptoticaly *infinitely often*.

Extensions

- Extension to off-policy setting (behavior policy $b \neq$ target policy π) with importance sampling.
- Extension to planning with policy improvement steps

- No theoretical results for the last case.
- Need to wait until the end of an episode to update anything...

Bootstrap and TD Prediction

Bootstrap and TD

- Rely on

$$\begin{aligned} v_\pi(s) &= \mathcal{T}_\pi v_\pi(s) \\ &= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right] \end{aligned}$$

- Temporal Difference: stochastic approximation scheme

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

- Update occurs at each time step.
- Can be proved to converge (under some assumption on α)!

- Combine the best of Dynamic Programing and MC.
- Can be written in term of Q:
  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$

SARSA and Q Learning

- How to use this principle to obtain the best policy?

SARSA: Planning by Prediction and Improvement (online)

- Update Q following the current policy π

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

- Update π by policy improvement.

- May not converge if one use a greedy policy update

Q Learning: Planning by Bellman Backup (off-line)

- Update Q following the behavior policy b

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$$

- No need to use importance sampling correction for depth 1 update.

- Proof of convergence in both cases.

Variations

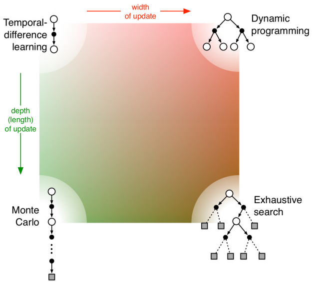


Figure 8.11: A slice through the space of reinforcement learning methods, highlighting the two of the most important dimensions explored in Part I of this book: the depth and width of the updates.

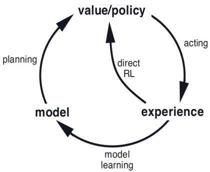
Depth

- Number of steps in the update. x

Width

- Number of states/actions considered at each step.

Planning and Learning



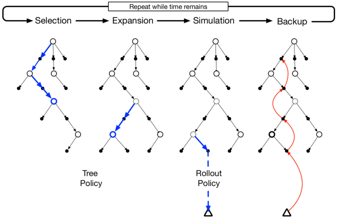


Figure 8.10: Monte Carlo Tree Search. When the environment changes to a new state, MCTS executes as many iterations as possible before an action needs to be selected, incrementally building a tree whose root node represents the current state. Each iteration consists of the four operations **Selection**, **Expansion** (though possibly skipped on some iterations), **Simulation**, and **Backup**, as explained in the text and illustrated by the bold arrows in the trees. Adapted from Chaslot, Bakkes, Szita, and Spronck (2008).

Planning and Models

- Planning can combine a model estimation (DP) and direct learning (RL).

Real Time Planning

- Planning can be made online starting from the current state.

- Curse of dimensionality: methods are hard to use when the cardinality of the states and the actions are large!

Outline

Value Function Approximation

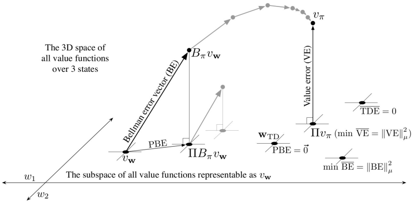
Value Function Approximation

- **Idea:** replace $v(s)$ by a parametric $\hat{v}(s, \boldsymbol{w})$.
- **Issues:**
  - Which approximation functions?
  - How to define the quality of the approximation?
  - How to estimate $\boldsymbol{w}$?

Approximation functions

- Any parametric (or kernel based) approximation could be used.
- Most classical choice:
  - Linear approximation.
  - Deep Neural Nets...

Approximation Quality



- How define when $\hat{v}(\cdot, \boldsymbol{w})$ is close to v_π (or v_*)

Prediction(/Control)

- Prediction objective:
$$\sum_s \mu(s)(v_\pi(s) - \hat{v}(s, \boldsymbol{w}))^2$$
- Bellman Residual:
$$\sum_s \mu(s)(\mathcal{T}_\pi \hat{v}(s, \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w}))^2$$
or its projection...

- **Issue:** Neither v_π or $\mathcal{T}_\pi$ are known...

Online Gradient and Semi-Gradient

Online Prediction

- SGD algorithm on $\boldsymbol{w}$:
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(v_\pi(S_t) - \hat{v}(S_t, \boldsymbol{w}) \right) \nabla \hat{v}(S_t, \boldsymbol{w})$$
- MC approximation (still SGD):
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(G_t - \hat{v}(S_t, \boldsymbol{w}) \right) \nabla \hat{v}(S_t, \boldsymbol{w})$$
- TD approximation (not SGD anymore):
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}) \right) \nabla \hat{v}(S_t, \boldsymbol{w})$$

- Deeper or wider scheme possible.

Online Control

- SARSA-like algorithm:
    - Prediction step as previously with the current policy
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}) - \hat{q}(S_t, A_t, \boldsymbol{w}) \right) \nabla \hat{q}(S_t, A_t, \boldsymbol{w})$$
    - ϵ-greedy update of the current policy

Offline Control with Approximation

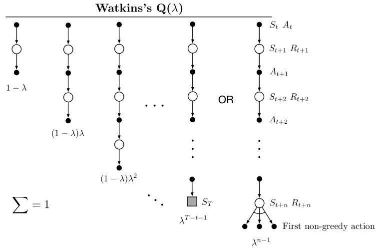


Figure 12.12: The backup diagram for Watkins's Q(λ). The series of component updates ends either with the end of the episode or with the first nongreedy action, whichever comes first.

Offline Control

- Q-Learning like algorithm:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \boldsymbol{w}) - \hat{q}(S_t, A_t, \boldsymbol{w})\right) \times \nabla \hat{q}(S_t, A_t, \boldsymbol{w})$$

  with an arbitrary policy b.
- Deeper formulation using importance sampling possible.

- **Issue:** Hard to make it converge in general!

Deadly Triad

Sutton-Barto's Deadly Triad

- **Function Approximation**
- **Bootstrapping**
- **Off-policy training**

Stabilization Tricks

- (Back to policy iteration),
- Memory replay: sample from a set of episodes
- Frozen Q: use the previous weights in the max
- Clip/normalize rewards...

Actor-Critic

- Other approach with a **parametric policy**.

Actor-Critic

- Simultaneous parameterization of
  - the policy π by $\boldsymbol{\theta}$,
  - the value function s by $\boldsymbol{w}$
- Simultaneous update:

$$\delta_t = R_t + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w})$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_{t+1} + \alpha \delta_t \frac{\nabla \pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_{t+1} + \alpha \delta_t \nabla \hat{v}(S_t, \boldsymbol{w})$$

- Online approach
- Can be adapted to continuous actions.

Outline

Numerical Experiments

Naive approach: Breiman CART

- Greedy policy that optimize the expectation of next step entropy.

Baseline: Actor-critic with REINFORCE

- Linearly parametrized policy using next step entropy expectation and other simple features

Deep Q-Learning

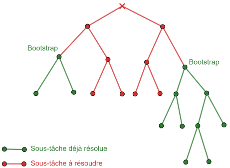
- Q-Learning with Neural Networks.

- Nothing specific for the first two approaches...

Very High Dimension Case!

Issues

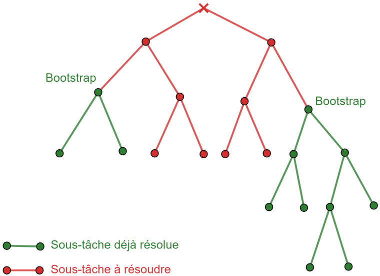
- DQN is unstable with TD in our setting (too slow to backpropagate the rewards?)
- Much better results using MC!
- Still hard to optimize everything from the beginning!



Dimension Reduction Trick

- State space partitioning to solve several smaller sub-problems.

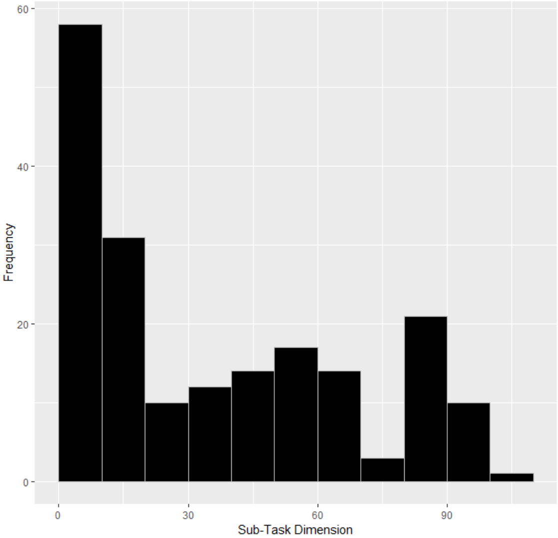
State Partition and MC



- Partition obtained by solving the problem starting from an abnomalies and falling back to previously computed strategy as soon as one reach a common state.
- Similar to a *n*-step bootstrapping!
- Works well with MC as *n* is not too large.

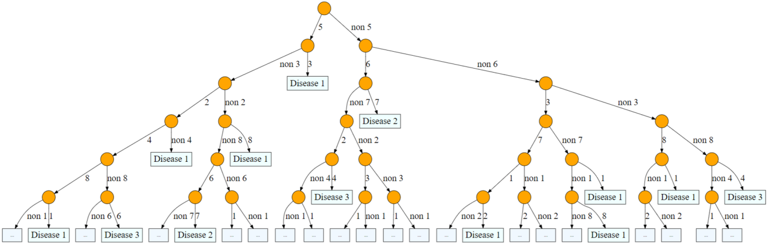
Subtask Dimension

Back to Prenatal Ultrasound

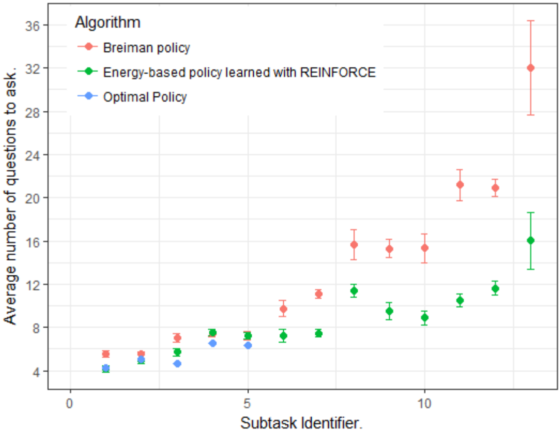


Optimal Decision Tree for a Small Subtask

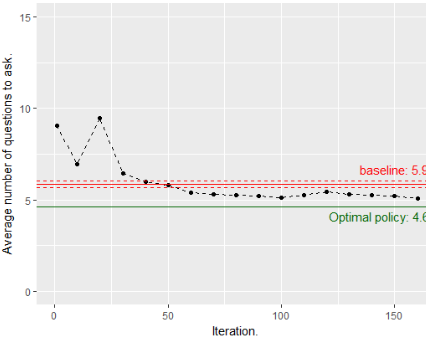
Back to Prenatal Ultrasound



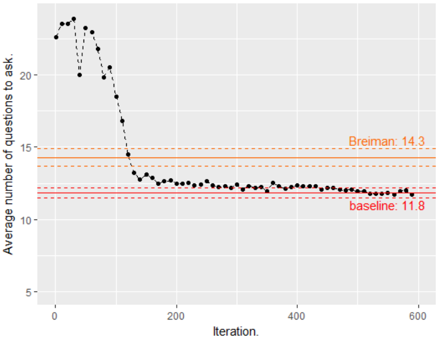
Optimal Policy for Small Subsets?



DQN vs REINFORCE vs Breiman

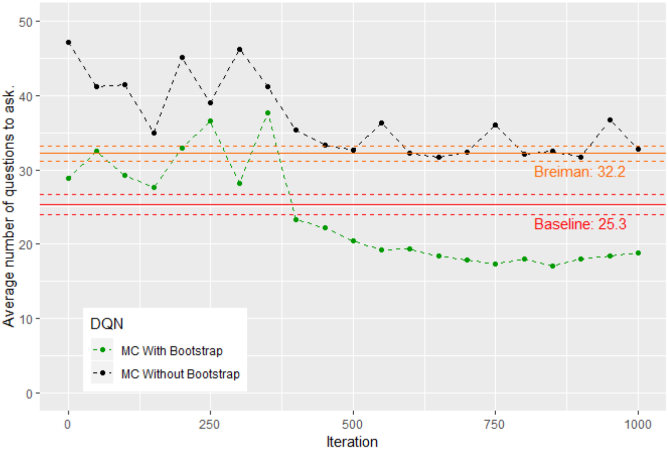


Task Dimension: 10



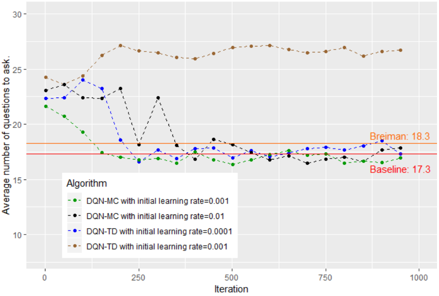
Task Dimension: 26

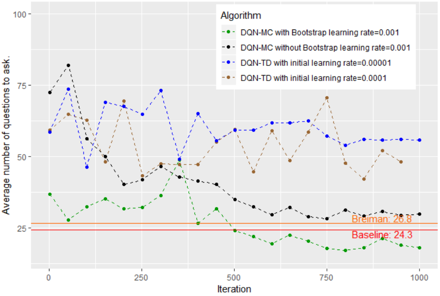
DQNs vs REINFORCE vs Breiman



Task Dimension: 70

DQNs vs REINFORCE vs Breiman





Task Dimension: 29

Task Dimension: 104

Take Away Message

Medical Goals

- Help obstetricians by improving/systematizing ultrasonic diagnostic **(MDP modeling)**
- Guide a (non rare disease expert) sonographer to assess as fast as possible potential diseases **(first prototype at Necker)**

Technical Goals

- Build an optimized decision tree:
    - Need to learn the environment **(MaxEnt and data assim.)**
    - Reinforcement learning **(Param. policy and MC vs Deep Q)**
- **Not yet** (theoretical) guarantees.

Take Away Message

- Reinforcement learning (or MDP) is an interesting tool.
- Formalization requires a true dialog between the mathematicans and the practicians.
- First prototype already tested by Necker.

The Team

Cliniciens

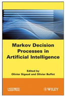
Statisticiens

Expert base de données

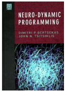
Outline

References

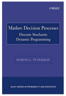
R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018



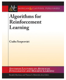
O. Sigaud and O. Buffet.
Markov Decision Processes in Artifical Intelligence.
Wiley, 2010



D. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, 1996



M. Puterman.
Markov Decision Processes. Discrete Stochastic Dynamic Programming.
Wiley, 2005



Cs. Szepesvári.
Algorithms for Reinforcement Learning.
Morgan & Claypool, 2010

R. Besson, E. Le Pennec, S. Allassonnière, J. Stirnemann, E. Spaggiari, A. Neuraz, "A Model-Based Reinforcement Learning Approach for a Rare Disease Diagnostic Task", arXiv preprint, 2018

Licence and Contributors

Contributors

- Main contributor: E. Le Pennec
- Contributor: R. Besson