Optimization of a Sequential Decision Making Problem for a Rare Disease Diagnostic Application.

Erwan Le Pennec

with

Rémi Besson (X) and Stéphanie Allassonnière (Paris Descartes)

TSE - 04/10/2018

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?

Outline

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

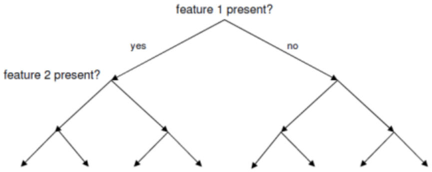
- Rare diseases: very few cases even in the world largest DB!

Excel Type Dataset

- Expert database build from literature (E. Spaggiari).
- 81 diseases, 307 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: $\mathbb{S} = \{P, A, U\}^{307}$ (presence,absence,not yet looked at) for each symptom.
- Action: $\mathbb{A} = \{1, \ldots, 307\}$ next symptom.
- Policy: $\pi : s \in \mathbb{S} \mapsto a \in \mathbb{A}$ next symptom given the state.

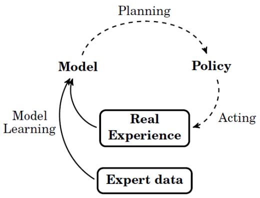
Probabilistic setting

- 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 $\pi^\star(\mathcal{S}_0) = \operatorname{argmin}_\pi \mathbb{E}[T|\mathcal{S}_0]$!

Problems to be Solved



Environment Learning with Maximum Entropy Principle

- We have $P[S_i \mid D]$ but we need to know $P[S_{i_1}, \ldots, S_{i_K} \mid D]$.
- Idea: add some expert knowledge and maximize uncertainty.

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 non adversarial game and find the optimal strategy.

Outline

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?

Outline

State, Action and Reward

State

- State: $\mathbb{S} = \{P, A, U\}^{307}$ (presence,absence,not yet looked at) for each symptom.
- Final state: state for which the disease entropy is below ϵ

Action

- Action: $\mathbb{A} = \{1, \ldots, 307\}$ next symptom.

Rewards

- Reward r on each action:
  - 0 if current state is terminal,
  - -1 otherwise.
- Not random!

Policy and Cumulative Reward

Policy

- Policy: $\pi : \mathcal{S} \in \mathbb{S} \mapsto a \in \mathbb{A}$ (Next symptom given the state)
- Can be deterministic or stochastic...

Policy Execution

- Initial state: $\mathcal{S}_0 = (U, \ldots, U)$
- At step t,
  - Select action $\pi(\mathcal{S}_{t-1})$
  - Observe reward r_t and new state $\mathcal{S}_t$
  - Stop if $\mathcal{S}_t$ is terminal.

Cumulative Reward

- With T the stopping time

$$\mathcal{R} = \sum_{t=1}^{T} r_t \ (= -T)$$

- Here $T \leq 307$...

Policy Quality

Policy and Cumulative Reward

- Policy: $\pi : s \in \mathbb{S} \mapsto a \in \mathbb{A}$ (Next symptom given the state)
- Initial state: $\mathcal{S}_0 = (U, \ldots, U)$
- Policy execution: $\mathcal{S}_t \to a_t = \pi(\mathcal{S}_t) \to r_t \to \mathcal{S}_{t+1}$.
- Cumulative reward:

$$\mathcal{R} = \sum_{t=1}^{T} r_t \; (= -T)$$

Policy Quality

- Cumulative reward is random!
- Quality measure by expected value given the initial state:

$$V_\pi(\mathcal{S}_0) = \mathbb{E}_\pi[\mathcal{R}|\mathcal{S}_0]$$

Policy Evaluation and Planning

Policy Quality

- Quality measured by the policy value:
$$V_\pi(\mathcal{S}) = \mathbb{E}_\pi[\mathcal{R}|\mathcal{S}]$$

Two natural problems

- Policy evaluation: compute v_π given π.
- Planning: determine $\pi^\star$ such that $V_{\pi^\star}(\mathcal{S}) \geq V_\pi(\mathcal{S})$ for all $\mathcal{S}$ and π.

- Those objects may not exist in general! In our case, they exist.
- Can be traced back to the 50's!

Outline

Policy Evaluation by Bellman Backup

Fixed Point Property

- Policy value is the solution of a fixed point problem:
$$V_\pi(\mathcal{S}) = \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, \pi(\mathcal{S}))(V_\pi(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, \pi(\mathcal{S}), \mathcal{S}')))$$
- Bellman operator $\mathcal{T}^\pi$:
$$V(\mathcal{S}) \mapsto \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, \pi(\mathcal{S}))(V(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, \pi(\mathcal{S}), \mathcal{S}')))$$

Policy Evaluation by Dynamic Programming

- Iterative algorithm:
$$V_{n+1}(\mathcal{S}) = \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, \pi(\mathcal{S}))(V_n(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, \pi(\mathcal{S}), \mathcal{S}')))$$
- Convergence can be proved. (Finite time for finite horizon!)

Policy Iteration

Policy Enhancement by Bellman Backup

- π is enhanced by replacing it by
$$\pi'(\mathcal{S}) = \operatorname*{argmax}_a \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(V_\pi(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}')))$$

Policy Planning by Policy Iteration

- Policy iteration: alternate estimation of V_π and policy enhancement.
- Convergence can be proved. (Finite time for finite states!)

- Analysis much more complicated when estimation of V_π is only approximate.

Value Iteration

Policy Enhancement by Bellman Backup

- π is enhanced by replacing it by
$$\pi'(\mathcal{S}) = \operatorname*{argmax}_a \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(V_\pi(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}')))$$

Policy Planning by Dynamic Programming

- Direct clever iterative algorithm using Bellman operator $\mathcal{T}$:
$$V_{n+1}(\mathcal{S}) = \max_a \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(V_n(\mathcal{S}) + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}')))$$
- Convergence can be proved. (Finite time for finite states!)
- Optimal policy:
$$\pi^\star(\mathcal{S}) = \operatorname*{argmax}_a \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(V_\infty(\mathcal{S}) + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}')))$$

Q Value Iteration

Policy Enhancement by Bellman Backup

- Q-value function (action-value function):

$$Q_\pi(\mathcal{S}, a) = \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(V_\pi(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}'))$$

$$V_\pi(\mathcal{S}) = Q_\pi(\mathcal{S}, \pi(\mathcal{S}))$$

- π is enhanced by replacing it by

$$\pi'(\mathcal{S}) = \operatorname*{argmax}_a Q_\pi(\mathcal{S}, a)$$

Policy Planning by Dynamic Programming

- Direct clever iterative algorithm using Bellman operator $\mathcal{T}$:

$$Q_{n+1}(\mathcal{S}, a) = \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(\max_{a'} Q_n(\mathcal{S}', a') + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}')))$$

- Convergence can be proved. (Finite time for finite states!)
- Optimal policy:

$$\pi^\star(\mathcal{S}) = \operatorname*{argmax}_a Q_\infty(\mathcal{S}, a)$$

Problem Solved?

Two main issues

- Need to modify all states simultaneously.
- Need to known explicitly the transition probability (and the expected reward)

Asynchronous update

- Modify V or Q each time one consider a state.
- Different strategy to order the states:
  - fixed order,
  - Bellman equation error order,
  - current strategy play.

- Convergence results if all the couple states-actions are visited infinitely often...

Outline

Stochastic Approx. and Robbins-Monro

- Lots of fixed point in MDP: $h(V) = \mathcal{T}V - V = 0$, $h(Q) = \mathcal{T}Q - Q$...

Sketched Robbins-Monro Theorem

- Goal: Solve $h(\theta) = 0$
- Assumption:
  - the minimizer $\theta^\star$ is such that $\forall \theta, \langle h(\theta), \theta - \theta^\star \rangle < 0$
  - it exists H_n such that $\mathbb{E}[H_n(\theta)] = h(\theta)$
- Algorithm:

$$\theta_{n+1} = \theta_n + \alpha_n H_n(\theta_n)$$

- Thm: θ_n converges toward $\theta^\star$

- Example: $H(\theta)$ is decreasing.
- Assumption can be relaxed (Lyapunov function...)
- Coordinatewise update possible if all coordinate are visited infinitely often.
- Can we capitalize on this?

Value Function and Bellman

Dynamic Programming

- Bellman equation:

$$\begin{aligned} V_\pi(\mathcal{S}) &= \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, \pi(\mathcal{S}))(V_\pi(\mathcal{S}') + \mathbb{E}(r(\mathcal{S}, \pi(\mathcal{S}), \mathcal{S}'))) \\ &= \mathcal{T}^\pi V_\pi \end{aligned}$$

- V_π is a zero of $h(V) = \mathcal{T}^\pi V - V$.
- Bellman approximation (Temporal Difference):

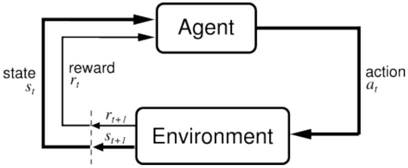
$$h(V)(\mathcal{S}_t) = \underbrace{V(\mathcal{S}_{t+1}) + r(\mathcal{S}_t, \pi(\mathcal{S}_t), \mathcal{S}_{t+1})}_{\text{Unb. est. of } \mathcal{T}(V)(\mathcal{S}_t)} - V(\mathcal{S}_t)$$

- Algorithm:

$$V_{n+1}(\mathcal{S}_t) = V_n(\mathcal{S}_t) + \alpha_n \left(V(\mathcal{S}_{t+1}) + r(\mathcal{S}_t, \pi(\mathcal{S}_t), \mathcal{S}_{t+1}) - V(\mathcal{S}_t)\right)$$

- No need to know explicitly the transitions. (model free)
- Only need to be able to *play* and observe the environment.
- Allow policy evaluation and approximate policy iteration.
- Policy should explore all states and all actions infinitely often!

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!

Value Function and Monte Carlo

Monte Carlo

- Value function

$$V_\pi(\mathcal{S}) = \mathbb{E}\left[\sum_0^{T-1} r(\mathcal{S}_{t+i}, \pi(\mathcal{S}_{t+i}), \mathcal{S}_{t+i+1})\right]$$

- V_π is a zero of $\mathbb{E}[\sum_{i=1}^{T} r_t] - V$
- MC approximation (Temporal Difference):

$$H(V)(\mathcal{S}_t) = \underbrace{\sum_0^{T-1} r(\mathcal{S}_{t+i}, \pi(\mathcal{S}_{t+i}), \mathcal{S}_{t+i+1})}_{G_t} - V(\mathcal{S}_t)$$

- Algorithm:

$$V_{n+1}(\mathcal{S}_t) = V_n(\mathcal{S}_t) + \alpha_n\left(G_t - V_n(\mathcal{S}_t)\right)$$

- No need to know explicitly the transitions.
- Only need to be able to *play* and observe the environment.
- TD(λ): Interpolation between Bellman and MC.

Q Learning

Action-value function Q

- Bellman fixed point:

$$\begin{aligned} Q(\mathcal{S}, a) &= \sum_{\mathcal{S}'} p(\mathcal{S}' \mid \mathcal{S}, a)(\max_{a'} Q(\mathcal{S}', a') + \mathbb{E}(r(\mathcal{S}, a, \mathcal{S}'))) \\ &= \mathcal{T}(Q)(\mathcal{S}, a) \end{aligned}$$

- Optimal Q is a zero of $\mathcal{T}(Q) - Q$.
- Bellman approximation:

$$H(Q)(\mathcal{S}_t, a_t) = \max_{a'} Q(\mathcal{S}_{t+1}, a') + r(\mathcal{S}, a_t, \mathcal{S}_{t+1}) - Q(\mathcal{S}_t, a_t)$$

- Algorithm:

$$\begin{aligned} Q_{n+1}(\mathcal{S}_t, a_t) &= Q_n(\mathcal{S}_t, a_t) \\ &\quad + \alpha_n \left(\max_{a'} Q(\mathcal{S}_{t+1}, a') + r(\mathcal{S}, a_t, \mathcal{S}_{t+1}) - Q(\mathcal{S}_t, a_t) \right) \end{aligned}$$

- Reinforcement learning setting.
- Explo. policy should explore every state/action infinitely often.
- Optimal solution does not have this restriction!

Problem solved?

- Almost...

Dimension of the problem

- State of dimension: 3^{307}
- Much larger than the memory of any computer on earth...
- Previous methods intractable in our case!

Dimension reduction

- Parameterization of the policy?
- Parameterization of the value function?
- Parameterization of the action-value function?

Outline

Policy Parameterization

Parameterization

- $\pi_\theta(\mathcal{S}) = f_\theta(\Phi(\mathcal{S}))$ with $\theta \in \mathbb{R}^d$.
- Example: f_θ is a logit model depending on s only on $P(A|\mathcal{S})$ and $H(D|\mathcal{S})$
- Values functions:

$$V_\theta(\mathcal{S}) = \mathbb{E}_{\pi_\theta}[\sum r_t|\mathcal{S}]$$

$$Q_\theta(\mathcal{S}, a) = \mathbb{E}_{\pi_\theta}[\sum r_t|\mathcal{S}, a]$$

Parametric Optimization

- Optimization in θ by stoch. *gradient* descent?
- Issue: neither V or Q are known...

REINFORCE

Parametric Policy Gradient

- Value function gradient as an expectation of policy gradient!

$$\nabla V_\theta(\mathcal{S}_0) \propto \mathbb{E}_{\pi_\theta}\left[\sum_a \nabla \pi_\theta(a|\mathcal{S}) Q_\pi(\mathcal{S}, a)\right]$$

- Action sampling:

$$\nabla V_\theta(\mathcal{S}_0) \propto \mathbb{E}_{\pi_\theta}\left[\frac{\nabla \pi_\theta(a_t|\mathcal{S}_t)}{\pi_\theta(a_t|\mathcal{S}_t)} Q_\pi(\mathcal{S}_t, a_t)\right]$$

REINFORCE Algorithm

- Episodic MC play with

$$\theta_{t+1} = \theta_t + \alpha_t G_t \nabla \ln \pi_\theta(a_t|\mathcal{S}_t)$$

- Episodic MC play with baseline

$$\theta_{t+1} = \theta_t + \alpha_t \left(G_t - V_t(\mathcal{S}_t)\right) \nabla \ln \pi_\theta(a_t|\mathcal{S}_t)$$

where V_t is any function independent of a...

Value Function Parameterization

Parametric V approximation

- V approximated by a function parameterized by w: $V_\pi \approx V_w$
- Quality measured by
$$J(w) = \mathbb{E}_\pi \left[\left(V_\pi(\mathcal{S}) - V_w(\mathcal{S}) \right)^2 \right]$$
- Gradient:
$$\nabla J(w) = -\mathbb{E}_\pi \left[\left(V_\pi(\mathcal{S}) - V_w(\mathcal{S}) \right) \nabla V_w(\mathcal{S}) \right]$$
- Optimal w: $\nabla J(w) = 0$...

MC algorithm playing policy π

- Update:
$$w_{t+1} = w_t + \alpha_t \left(G_t(\mathcal{S}_t) - v_w(\mathcal{S}_t) \right) \nabla V_w(\mathcal{S}_t)$$

- Convergence results for linear approximations.
- Similar algorithm for the Q function.

Approximate Bellman Backup

Parametric Value Function

- V function approx. by a function parameterized by some w:
$$V_\pi(\mathcal{S}) \approx V_w(\mathcal{S})$$
- Quality measured by
$$J(w) = \mathbb{E}_\pi\left[\left(V_\pi(\mathcal{S}) - V_w(\mathcal{S})\right)^2\right]$$
- Gradient:
$$\nabla J(w) = -\mathbb{E}_\pi\left[\left(V_\pi(\mathcal{S}) - V_w(\mathcal{S})\right)\nabla V_w(\mathcal{S})\right]$$
- Optimal w: $\nabla J(w) = 0$...

Approximate Bellman Backup

- Iterate
$$w_{t+1} = w_t + \alpha_t\left(r_t(\mathcal{S}_t) + V_w(\mathcal{S}_{t+1}) - v_w(\mathcal{S}_t)\right)\nabla V_w(\mathcal{S}_t)$$
- Biased estimate of $V_\pi(\mathcal{S})$...
- Some convergence results...

Approximate Q Learning

Parametric Action-Value

- Q function approx. by a function parameterized by some w:
$$Q(s,a) \approx Q_w(s,a)$$
- Almost zero characterization of the optimal: $\mathcal{T}Q_w - Q_w \simeq 0$
- More precisely: minimizer of
$$L(w) = \mathbb{E}\left[(\mathcal{T}Q_w(s,a) - Q_w(s,a))^2\right]$$

Approximate Q Learning Algorithm

- Iterate
$$w_{t+1} = w_t + \alpha_t \left(r_t(\mathcal{S}_t) + \max_{a'} Q_w(\mathcal{S}_{t+1}, a')) - Q_w(\mathcal{S}_t, a_t) \right) \times \nabla Q_w(\mathcal{S}_t, a_t)$$
- Not stable!
- Is the derivation correct?

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 games
- Frozen Q: use the previous weights in the max
- Clip/normalize rewards...

Actor/Critic

- Learn simultaneously the optimal policy and its value function!

Actor/Critic

- Actor: policy
  - Action sampling with baseline:
    $$\nabla V_\theta(\mathcal{S}_0) \propto \mathbb{E}_{\pi_\theta}\left[\frac{\nabla \pi_\theta(a_t|\mathcal{S}_t)}{\pi_\theta(a_t|\mathcal{S}_t)}\left(Q_\pi(\mathcal{S}_t, a_t) - V_t(\mathcal{S}_t)\right)\right]$$
- Critic: estimate of the quality
  - $V_t = V_w$ a good parametric estimate of V_{π_θ}.
  - Bellman backup: $V_{w_t}(\mathcal{S}_t) \simeq r_t + V_{w_t}(\mathcal{S}_{t+1})$

Algorithm

- Simultaneous update:

$$w_{t+1} = w_t + \alpha_t\left(r_t(\mathcal{S}_t) + V_{w_t}(\mathcal{S}_{t+1}) - V_{w_t}(\mathcal{S}_t)\right)\nabla V_w(\mathcal{S}_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t\left(r_t(\mathcal{S}_t) + V_{w_t}(\mathcal{S}_{t+1}) - V_{w_t}(\mathcal{S}_t)\right)\nabla \ln \pi_\theta(a_t|\mathcal{S}_t)$$

- Can also param. $Q(\mathcal{S}, a)$ and use $r_t + Q_{w_t}(\mathcal{S}_{t+1}, \pi_{\theta_t}(\mathcal{S}_{t+1}))$...
- And tricks...

Outline

Parameterization

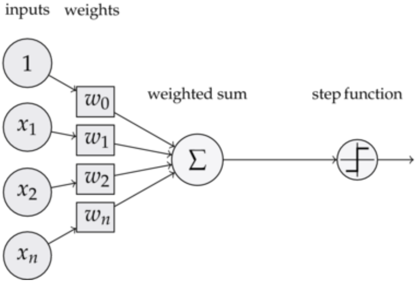
Linear

- V, Q or $\log(\pi)$ are linear with respect to some feature.
- Examples:
  - Tabular setting,
  - Logit model for the policy,
  - kernel decomposition...
- Some theoretical guarantees.

(Deep) Neural Network

- Much more freedom in the functions.
- Quite easy to try when one has a GPU.
- Almost no theoretical results!

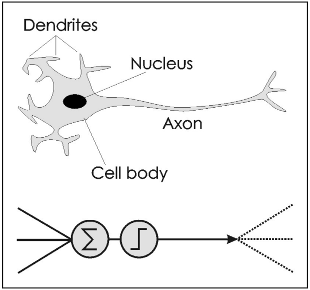
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

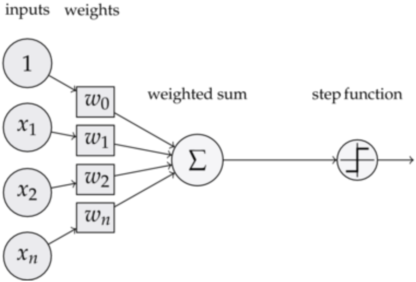
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

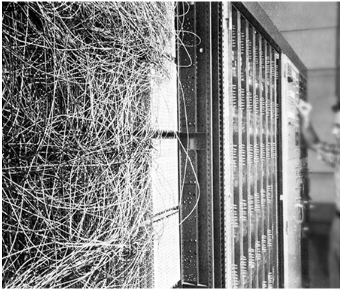
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

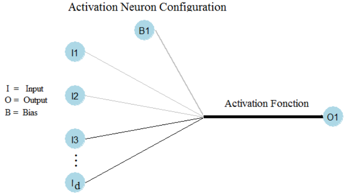
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Artificial Neuron and Logistic Regression



Artificial neuron

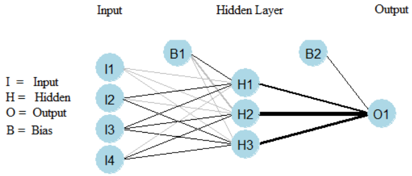
- Structure:
  - Mix inputs with a weighted sum,
  - Apply a (non linear) activation function to this sum,
  - Eventually threshold the result to make a decision.
- Weights learned by minimizing a loss function.

Logistic unit

- Structure:
  - Mix inputs with a weighted sum,
  - Apply the logistic function $\sigma(t) = e^t/(1+e^t)$,
  - Threshold at $1/2$ to make a decision!
- Logistic weights learned by minimizing the -log-likelihood.

- Equivalent to linear regression when using a linear activation function!

Multilayer Perceptron



MLP (Rumelhart, McClelland, Hinton - 1986)

- Multilayer Perceptron: cascade of layers of artificial neuron units.
- Optimization through a gradient descent algorithm with a clever implementation (**Backprop**)

- Construction of a function by composing simple units.
- MLP corresponds to a specific direct acyclic graph structure.
- Non convex optimization problem!

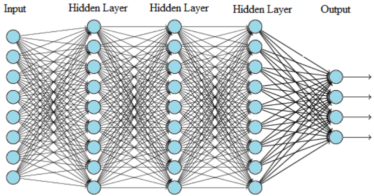
Universal Approximation Theorem

Universal Approximation Theorem (Hornik, 1991)

- A single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units

- Valid for most activation functions.
- A single hidden layer is sufficient but more may be more efficient.
- No bounds on the number of required units... (Asymptotic flavor)

Deep Neural Network

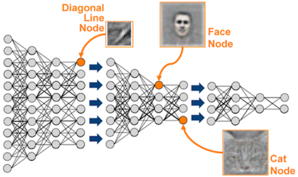


Deep Neural Network structure

- Deep cascade of layers!

- No conceptual novelty...
- But a lot of details that enabled to obtain a good solution: clever initialization, better activation function, weight regularization, accelerated stochastic gradient descent, early stopping...
- Use of GPU...
- Very impressive results!

Deep Learning



Family of Machine Learning algorithm combining:

- a (deep) multilayered structure,
- a clever optimization including initialization and regularization.

- Examples: Deep Neural Network, Deep (Restricted) Boltzman Machine, Stacked Encoder, Recursive Neural Network...
- Transfer learning: use as initialization a pretrained deep structure.
- Appears to be very efficient but lack of theoretical foundation!

Convolutional Network

PROC. OF THE IEEE, NOVEMBER 1998 7

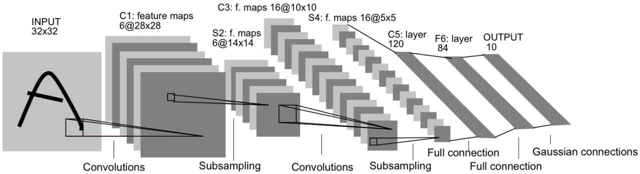
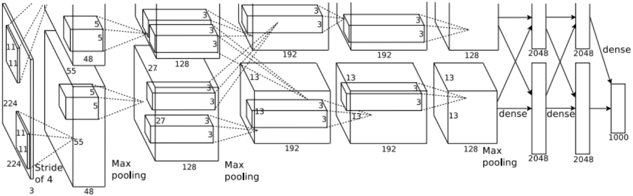


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

1989: 6 Hidden layer architecture (Yann LeCun)

- Drastic reduction of the number of parameters through a translation invariance principle (convolution)
- Requires 3 days of training for 60 000 examples!
- Tremendous improvement.
- Representation learned through the task.

Deep Convolutional Networks



2012: Alexnet (A. Krizhevsky, I. Sutskever, G. Hinton)

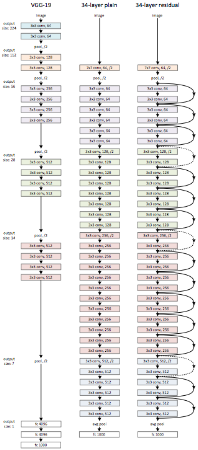
- Bigger layers and thus more parameters.
- Clever intialization scheme, RELU, renormalization and use of GPU.
- 6 days of training for 1.2 millions images.

Deep Convolutional Networks

[1]Inception 5 (GoogLeNet)

Inception 7a

[1]Going Deeper with Convolutions, [C. Szegedy et al, CVPR 2015]



- Deeper and deeper networks! (GoogLeNet / Residual Neural Network)

Outline

High-dimensional issues.

Issue

- DQN algorithm is not tractable for the main task: to find the best path starting from $s_0 = (2, ..., 2)$.

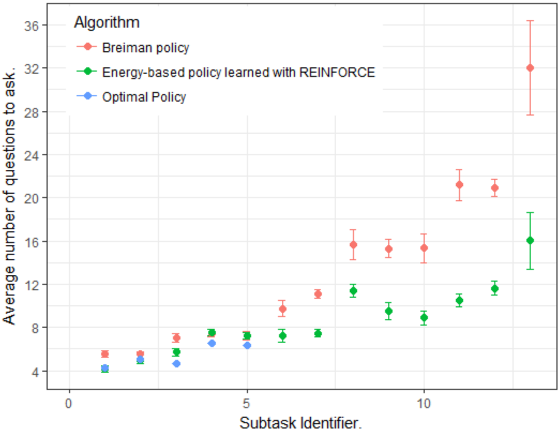
Dimension reduction

- Idea: Create subproblems of lower dimension.
- Learn a strategy starting from each $s_0^{(i)} = (2, ..., 2, 1, 2, ..., 2)$.
- Assumption: this first observed symptom is relevant (we can focus on the diseases for which this initial symptom is typical $\rightarrow$ reduce dimension).

Transfer Learning

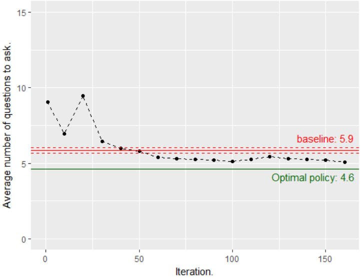
- Learn the strategy for the global task from what have been learned on subtasks: transfer learning.
- Ongoing research: promising results.

Some Results

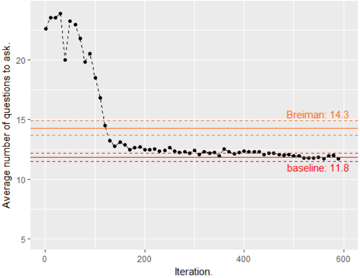


- Comparison between optimal policy, REINFORCE policy and Breiman policy.

Some Results



Task dimension: 10.



Task dimension: 26.

- Evolution of the performance of the neural network during the training phase with DQN-MC.

Some Results

	Average number of questions to ask.	
Subtask identifier	Meta-network	Specialized smaller network.
1	5.8	4.7
2	7.4	7
3	13.8	13.7
4	7.05	6.9
5	13.9	12.1

- Meta-Network initialized with tasks 1-4 and then trained with all tasks.

Outline

Uncertainty and Entropy

Environment Learning

- We have $P[S_i \mid D]$ but we need to know $P[S_{i_1}, \ldots, S_{i_K} \mid D]$.
- Idea: add some expert knowledge and maximize uncertainty.

Expert knowledge

- Some symptoms can not occur simultaneously...
- Need at least a certain number of symptoms to talk about a syndrome.

Uncertainty

- General idea: choose a solution that maximize the uncertainty while respecting the constraints (probability/impossibility).
- Uncertainty measured by entropy.

MaxEnt Principle

Environment Learning

- We have $P[S_i \mid D]$ but we need to know $P[S_{i_1}, ..., S_{i_K} \mid D]$.

- Naive idea: $P[S_{i_1}, ..., S_{i_K} \mid D] = P[S_{i_1} \mid D] \times ... \times P[S_{i_K} \mid D]$ (Conditional independence)

Data and Expert Knowledge

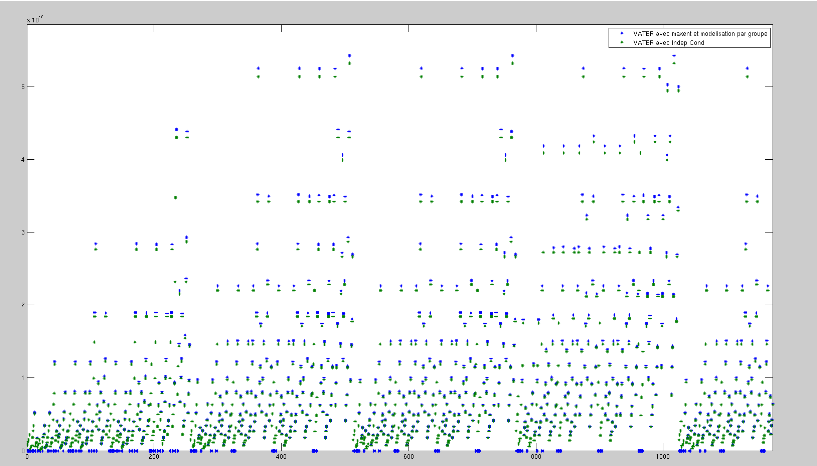
- Conditional probabilities: $P[S_i \mid D]$
- Medical constraints: $P[S_{i_k}, S_{i_{k'}} \mid D] = 0...$
- Mathematical constraints: P should be a probability...

MaxEnt Principle

- Maximize the entropy of the distribution $P[S_{i_1}, ..., S_{i_k} \mid D]$ under mathematical and medical constraints.

- Numerical scheme available.
- WIP on the interp. between maxent and maximum likelihood.

MaxEnt Estimate and Naive One



Medical Modeling Effect

- Difference not that large but important from the medical point of view.

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.

Acknowledgment

Thesis Team (CMAP/Paris Descartes)

- Rémi Besson, Erwan Le Pennec, Stéphanie Allassonnière

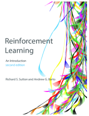
Yves Ville Team (Hôpital Necker Enfants Malades)

- Julien Stirneman, Emmanuel Spaggiari

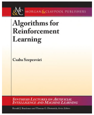
Anita Burgun Team (Paris Descartes/INSERM/Institut Imagine)

- Nicolas Garcelon, Anne-Sophie Jannot, Antoine Neuraz, Anita Burgun

References



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



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