
CERN

Automatic performance optimization and first steps towards reinforcement learning at the CERN Low Energy Ion Ring

Workshop: 2nd ICFA Workshop on Machine Learning for Charged Particle Accelerators
27 Feb. 2019

Simon Hirlaender,
Verena Kain, Michael Schenk

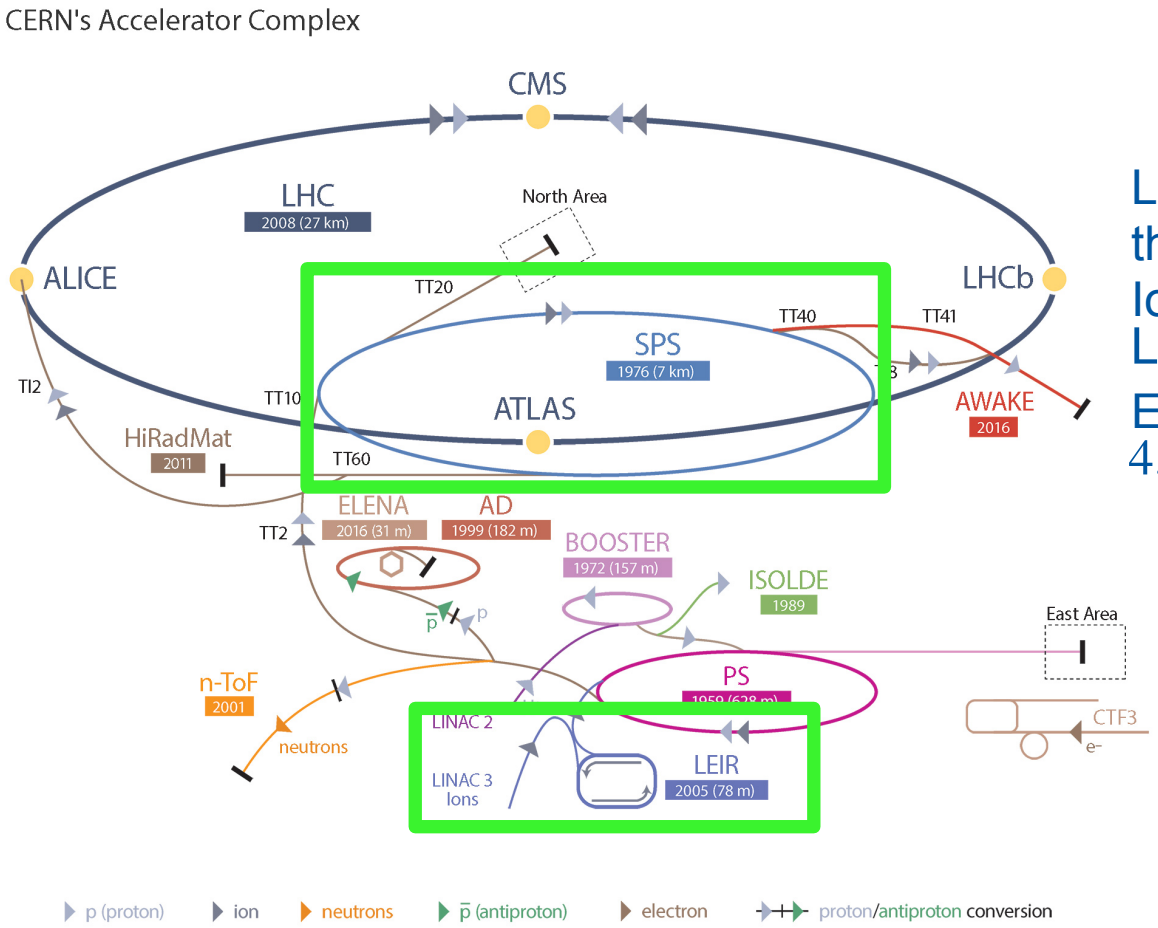
Overview

- Part 1: Introduction
- Part 2: Applications - optimization, reinforcement learning
- Summary/Outlook

Part 1

Introduction

LEIR/SPS in the CERN complex



LEIR: first synchrotron in the LHC ion chain

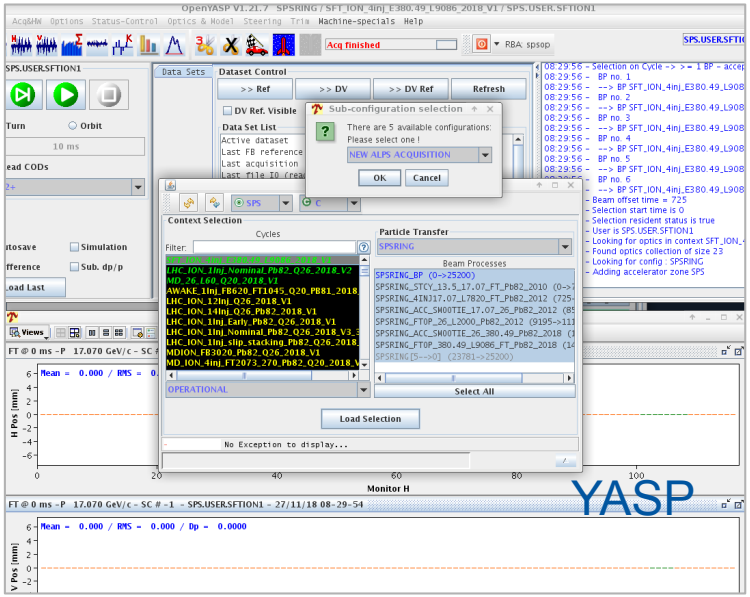
Ion species:
Lead 54^{+}

Energy range:
4.2 MeV/u – 72 MeV/u

Operating accelerators = correcting parameters

Observable, model and correctors….

- Well established CERN approach from LHC, SPS,.. - work with higher level parameters and deterministic algorithms, models
- Example SPS: improve transmission = reduce losses; e.g. reduce RMS orbit
  - Observable: 216 beam position monitors
  - Correctors: ~ 216 horizontal and vertical dipole corrector magnets
- Model based optimization based - deterministic algorithms
  - Correction done in 2 - 3 clicks
  - MICADO, SVD,.. YASP (CERN - Yet Another Steering Program)



But…

Some of the accelerators and beam characteristics difficult to model

Examples:

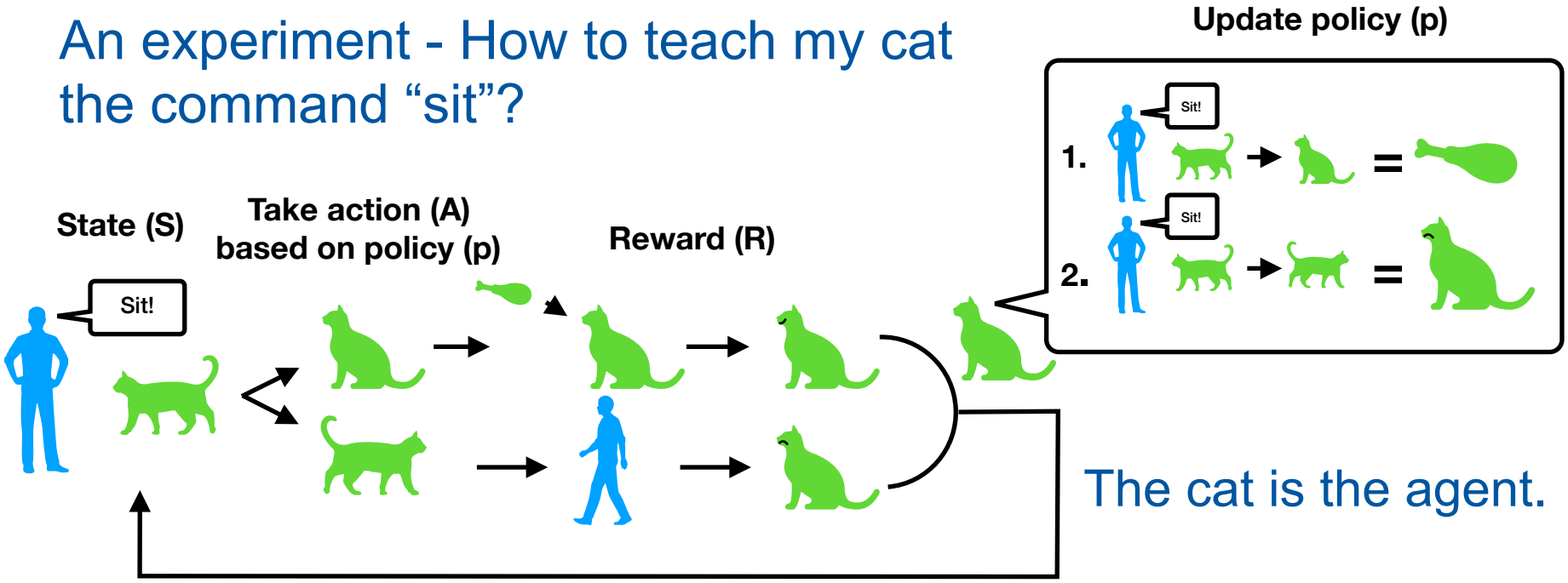
- Space charge dominated beam dynamics in LINACs
- E-cooling setting-up
- Transmission optimization during transition crossing
- Alignment of collimators, electrostatic septa with many degrees of freedom
- …

What can be done to avoid manual scans and trial-and-error?

- Develop online models as much as possible
- **Advanced optimization algorithms and reinforcement learning**

Basic terminology of RL

An experiment - How to teach my cat the command “sit”?



The cat is the agent.

The Reward Hypothesis: Goals and purposes = maximization of expected value of the cumulative sum of a received scalar signal - the reward.

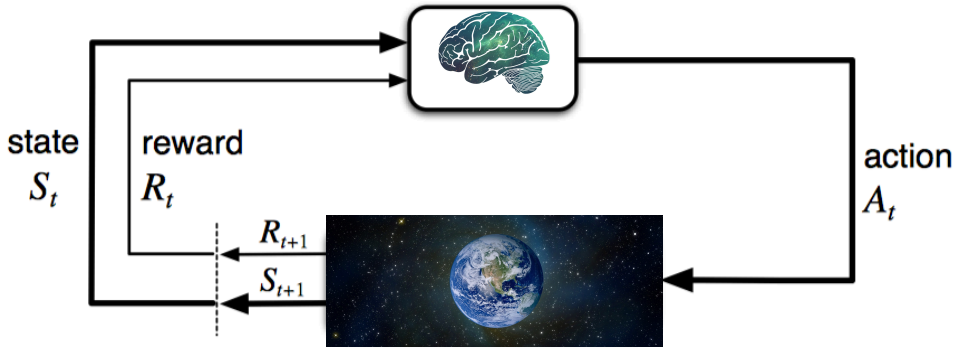
RL - mathematical formalism

- **Markov decision process (MDP)** of the first order:
  Current state s completely characterizes the state of the world.
  The world is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, which denotes the state-, action-, reward-spaces and the **transition probabilities** $\mathbb{P}$, R is the **current reward** and γ **the discount factor**.

- Objective: Find the policy, which maximizes the (expected) cumulative discounted reward for a given state:

$$G_t = \sum_{t \geq 0} \gamma^t R_t$$

$$\gamma \in [0,1]$$



Important terms

- The (stochastic) policy π: What is the strategy to take actions?

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- The **q** function (for a given π): How good is an action in a specific state following the given strategy **p** with respect to all future rewards?

$$q^{\pi}(s, a) = \mathbb{E}^{\pi}[G_t \mid S_t = s, A_t = a], \quad G_t := \sum_{t \geq 0} \gamma^t R_t$$

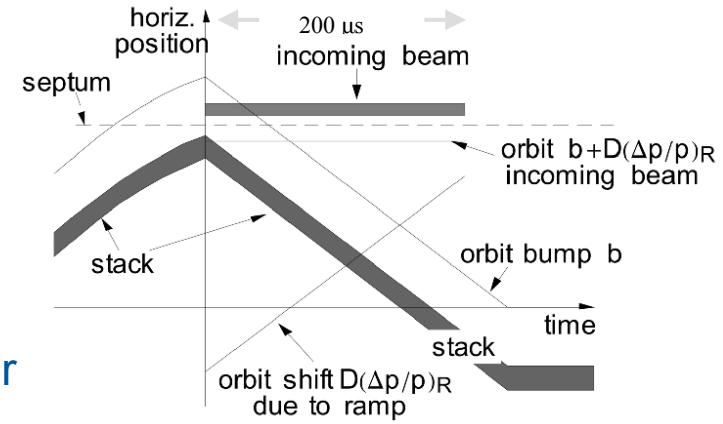
- Either one tries to find a good estimate for π or for **q** $\rightarrow$ or for both…

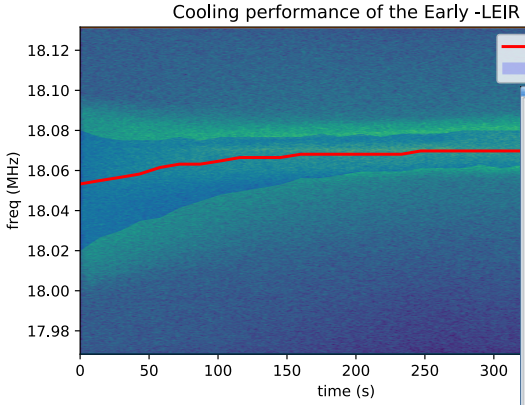
Part 2

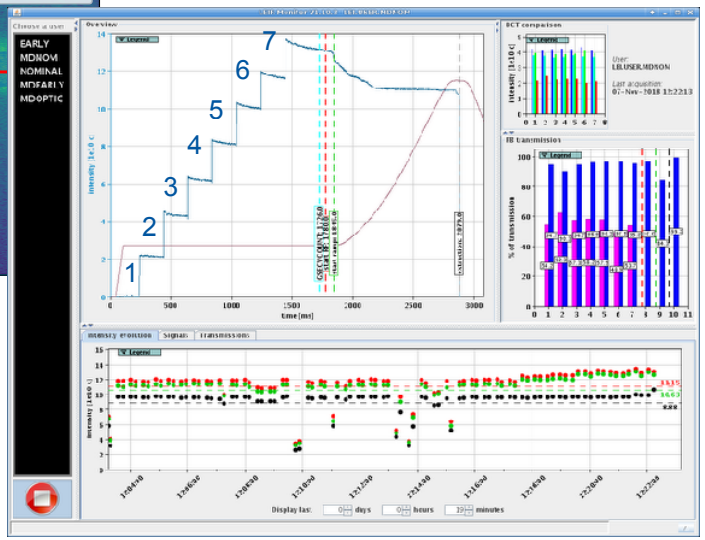
Applications - optimization, reinforcement learning

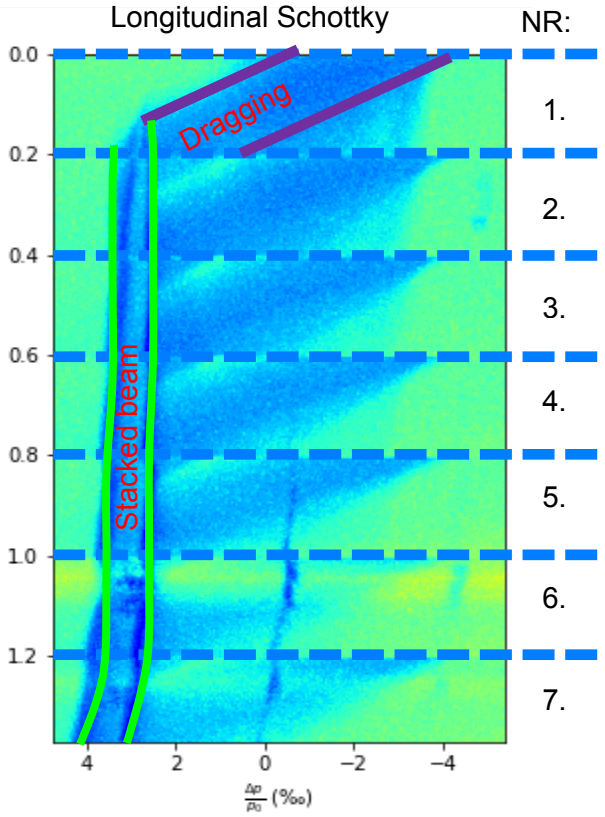
LEIR multi-turn injection with stacking

- 6D phase space painting — limited diagnostics
- E-cooling between injections to free space for next injection
- Sensitive to several interdependent parameters (e.g. injected intensity)





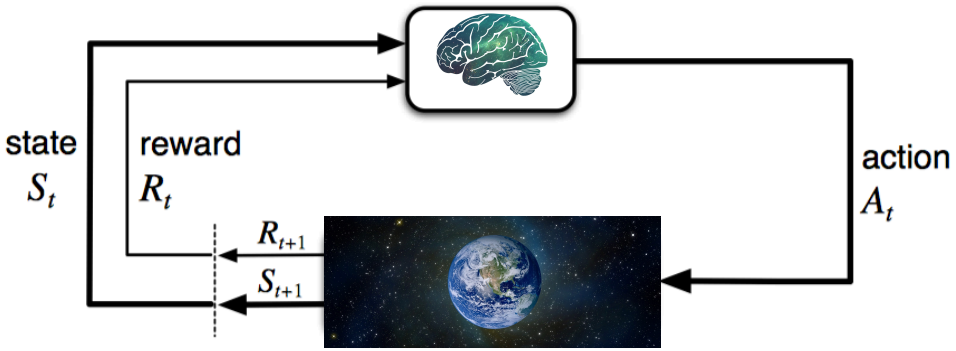




Model free agent - optimization algorithms
First test implementation

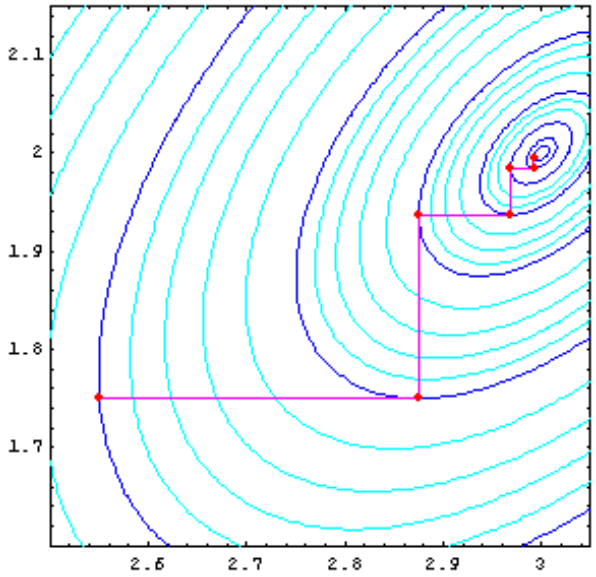
Scenario: maximize LEIR multi-turn injected intensity by optimizing LEIR orbit at injection

- Actions: trimmable high level parameters x, y, x', y' @ injection point
  - The state space is defined by the actions x, y, x', y'
- Reward: injected intensity

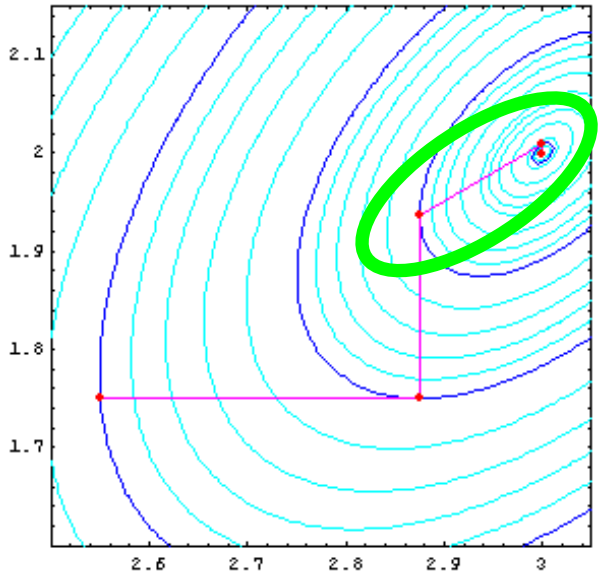


The policy

- The policy of first agent: **Powell optimization algorithm**
  - Learns from the first few moves and estimates the best new action (= direction)



Classical taxi-cab policy - search along fixed directions (human approach)

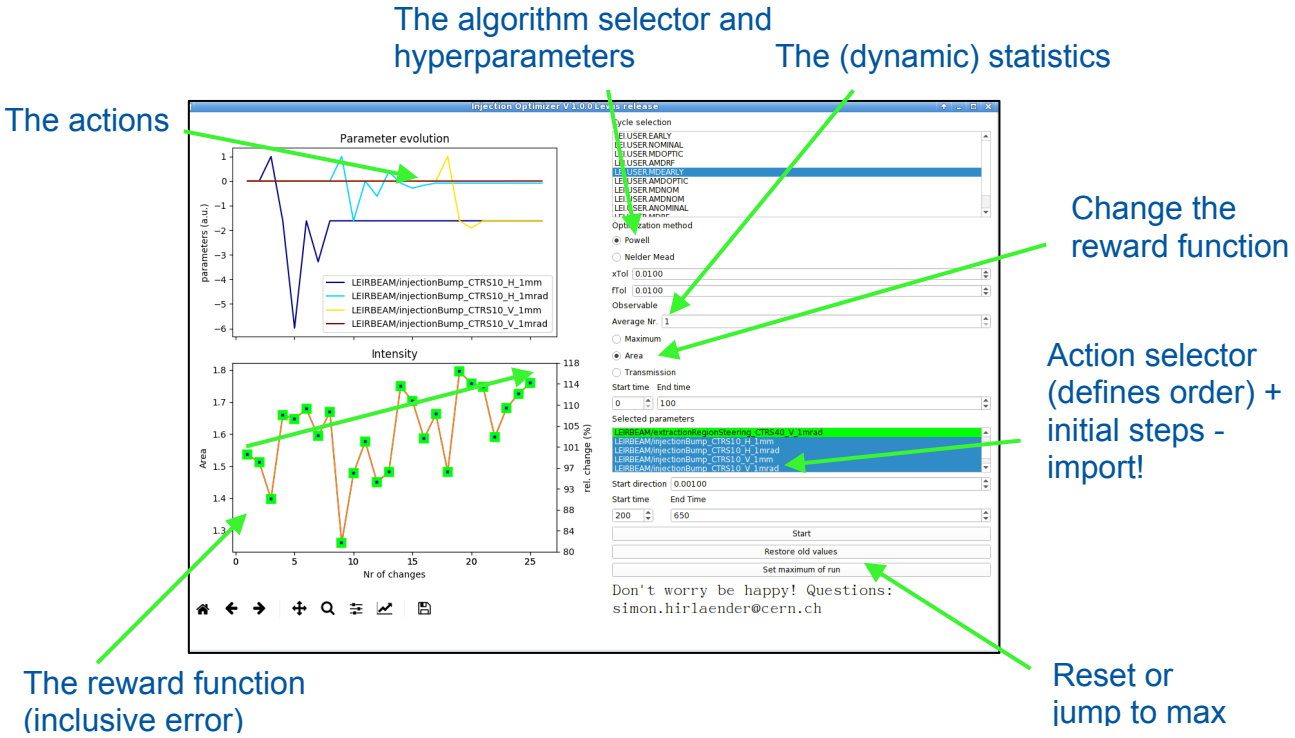


Powell's policy - take the direction of the average change

Pictures taken from: http://mathfaculty.fullerton.edu/mathews/n2003/PowellMethodMod.html

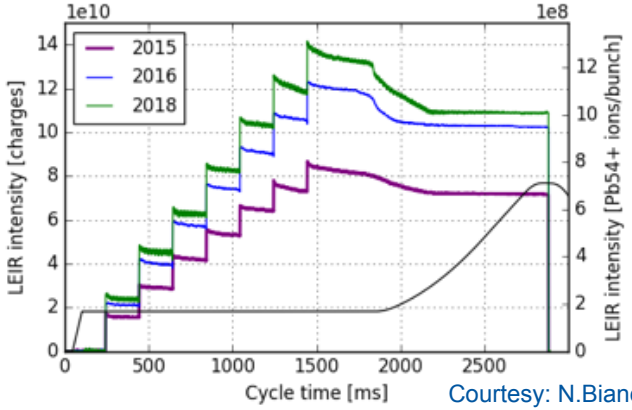
Implementation

- Python integrated in CERN control system
  - CERN pyJAPC to interface with hardware and settings DB
  - Can optimize scalar settings as well as function settings
  - Generic, operational framework - extendable to other machines (within minutes…)



Achievements - LEIR

- 2018: record injected intensity into LEIR (and LHC)
- Fast recovery after LEIR machine stops and drifts
- Reproducible performance

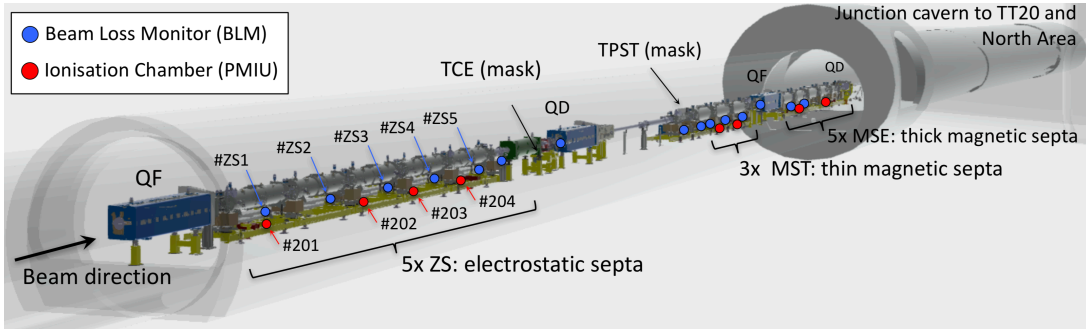


Courtesy: N.Biancacci

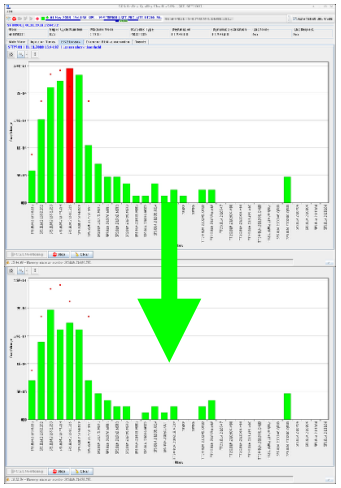
Result LHC 2018 for LEIR extracted intensity

75 ns	Mean /10^{10}c	Typical/10^{10}c	LIU/10^{10}c
LHC run	8.9	9.4	8.8

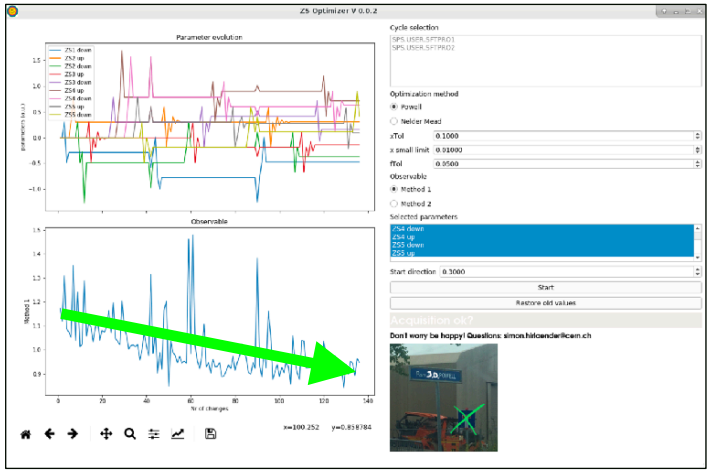
Achievements - other CERN machines



Normalized losses



Example: automatic alignment of electro-static septum for slow extraction at the SPS



- 5 3.5 m long tanks with moveable anodes
  - 9 degrees of freedom to optimize; goal: minimize losses in extraction channel
  - Constrained to protect the hardware
- **Reduced alignment time** from ~ 8 h (quasi- manual scans) to ~ 45 minutes

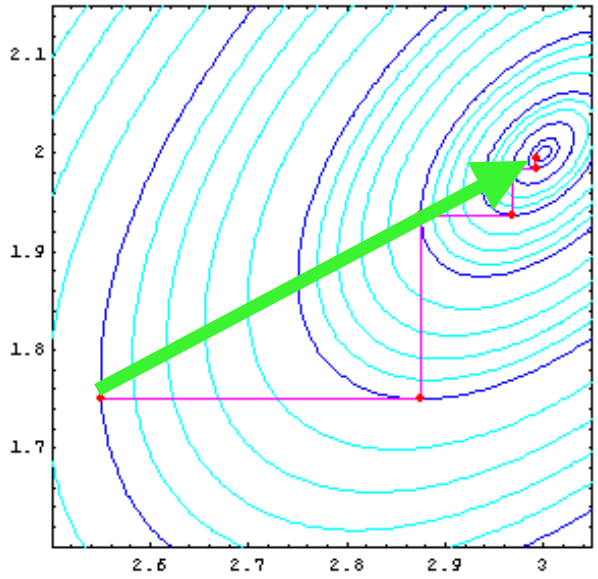
What's next?

- Successful application of Powell (and other) algorithms across CERN injector chain
  - Simple, noise resistant, scalable, flexible
- "Limitation": needs to re-learn each time → slow

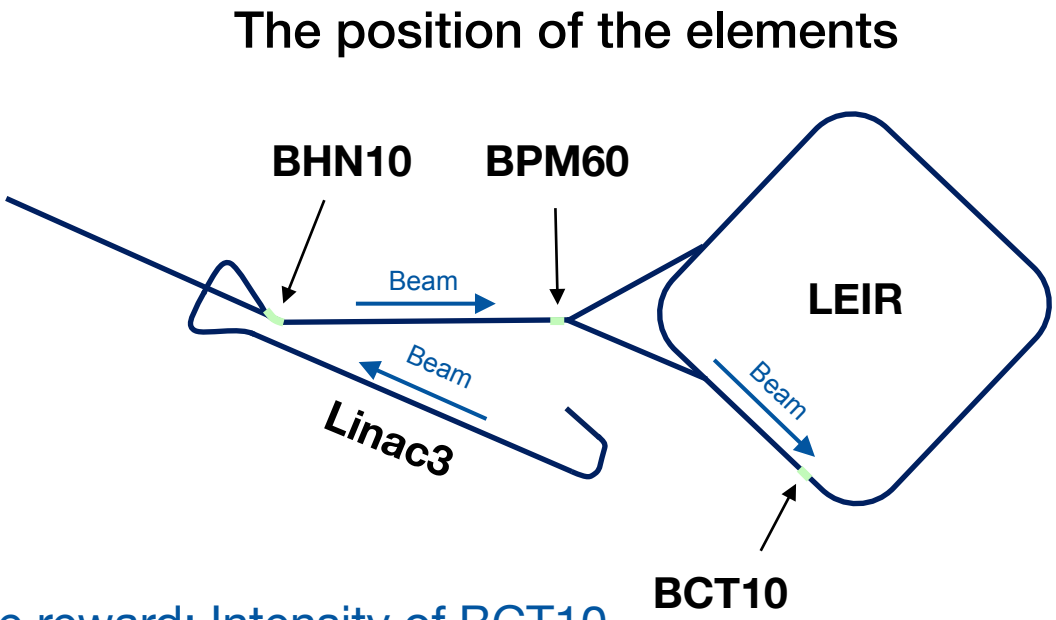
Next step:

Reinforcement learning

- Find the optimal policy



Reinforcement Learning - A proof of principle experiment (pop):

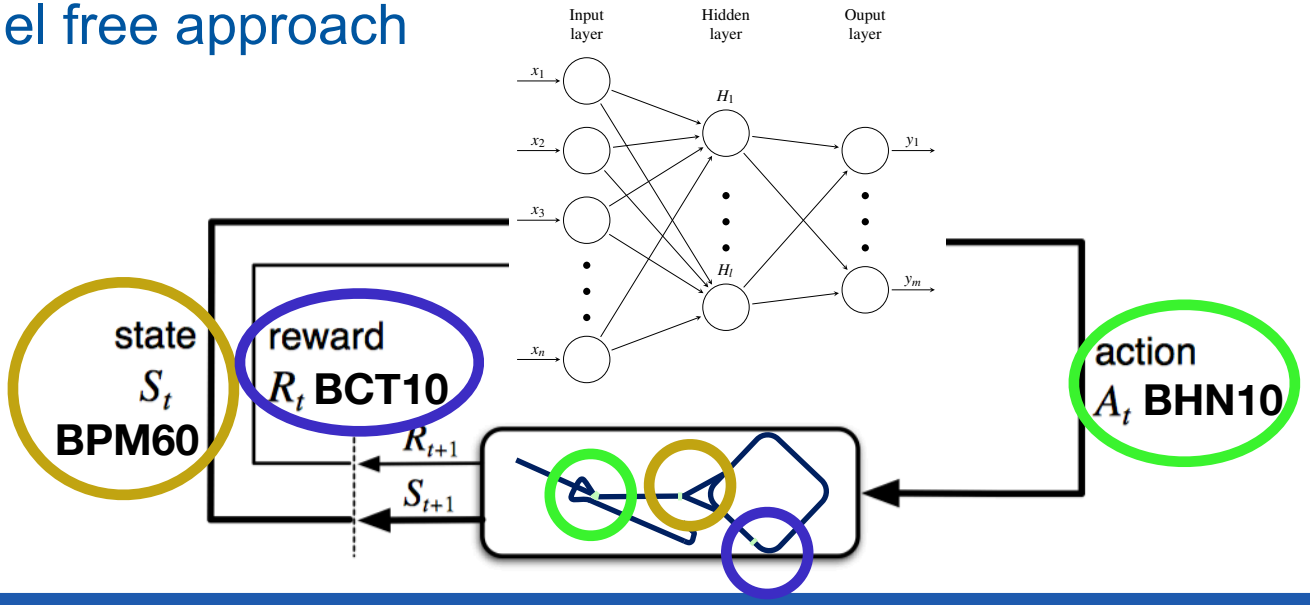


- The reward: Intensity of BCT10
- The state: Position of the beam at BPM60
- The action: Change by $\pm\Delta$ or hold the value of dipole BHN10

Used to create a test case to learn to handle the algorithm, hyper-parameters, artificial neural network architecture

Find the optimal policy - the approach

- Two main strategies in RL:
  - Find the policy directly (policy gradient) → computational intense - many iterations
  - Find the Q function iteratively and from that a better policy -> **Q learning**
- The used algorithm: DQN learning - (Deep Q learning)
- Guaranty: Q learning converges to the optimal policy
- Model free approach



The ingredients in more detail:

- What is Q learning?
- **Temporal difference** (TD) iteration:
  - Updating the **Q** - function iteratively - fast prediction with low number of iterations (bootstrapping) - online training possible
  - Offline policy update rule:

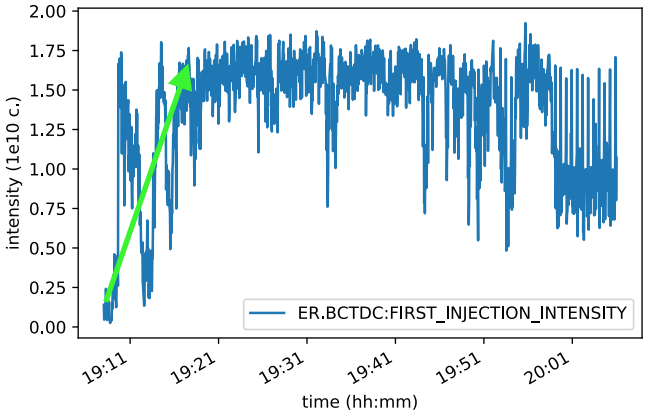
$$Q^{\pi}(S,A) \leftarrow Q^{\pi}(S,A) + \alpha \Big(\overbrace{R_{t+1} + \gamma \mathbf{max}_{a'} Q^{\pi}(S_{t+1}, a')}^{\textbf{Estimate including new information}} - Q^{\pi}(S,A) \Big)$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\textbf{TD-term}}$$

- Discount factor γ
- Learn rate α

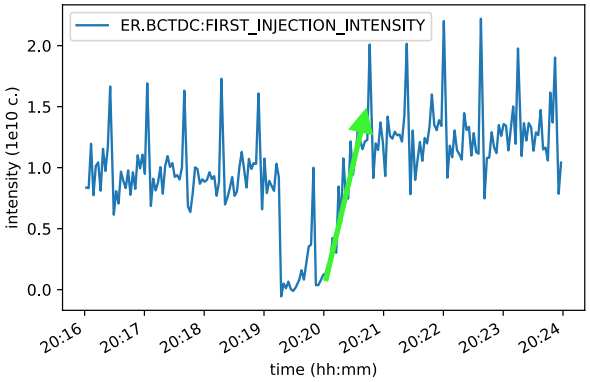
Pop results(1): Initial agent training

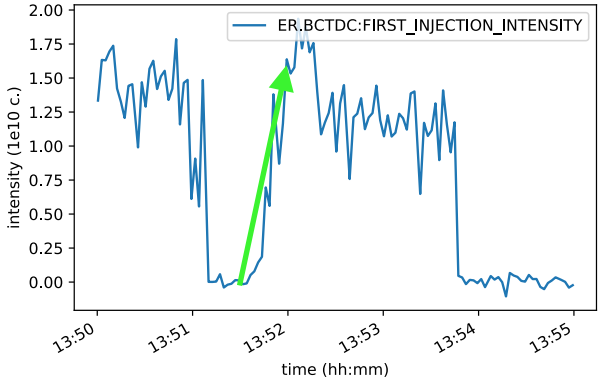
Training from scratch on LEIR approx. 600-iterations (measurement):



- The agent was able to learn the task without any initial knowledge and applied it successfully to different situations
- The drawback - large number of iterations

Apply the learned: real life test on different situations:recovered after a few moves

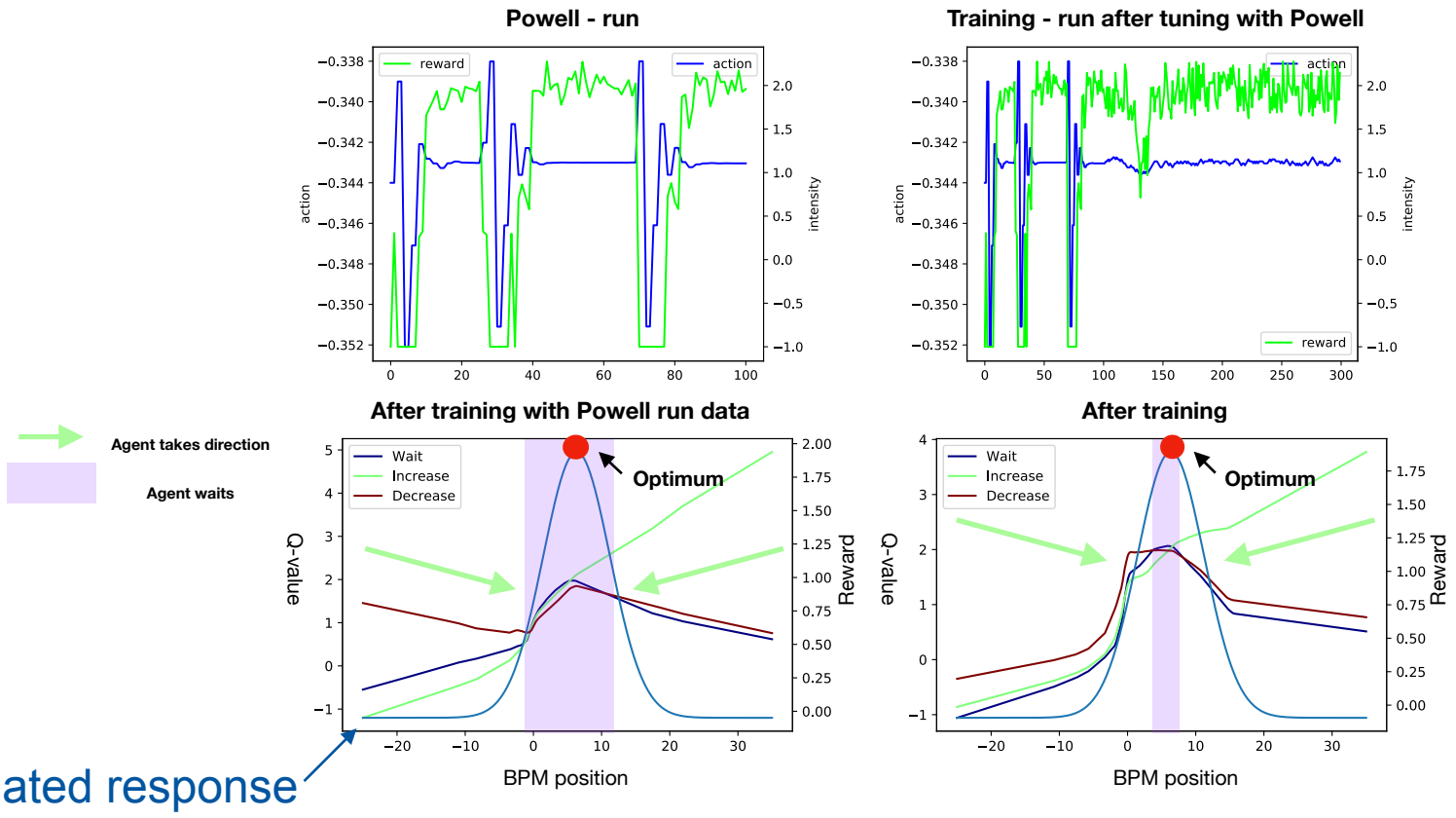




Projects on: https://gitlab.cern.ch/RL-group

Pop results(2): New hybrid training

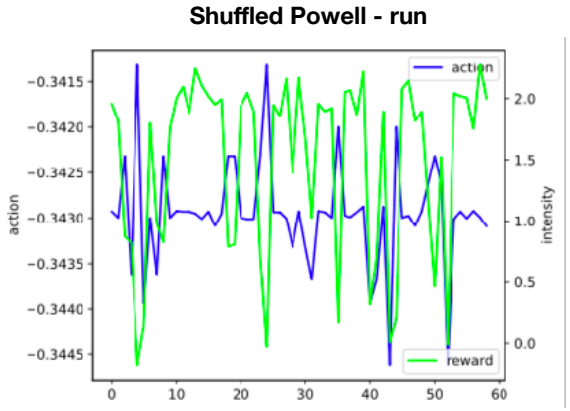
- Number of iterations reduced due to Powell training by a factor of two

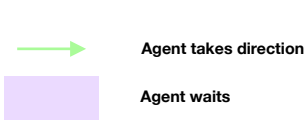


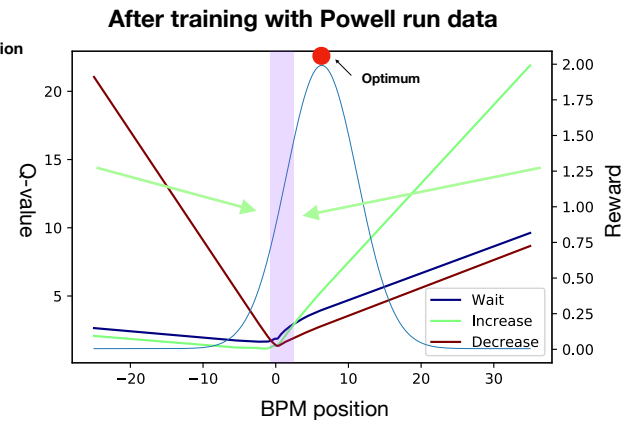
Simulated response

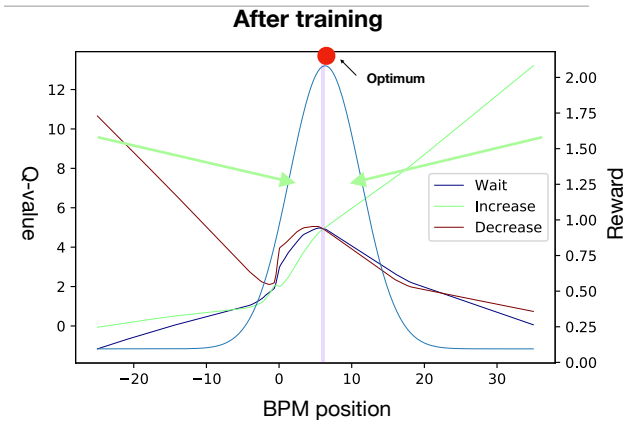
Pop results(3): Further tuning

- Using better parameters and deep double Q learning (DDQN) method
  Two independent networks to avoid a positive bias
- Number of iterations further reduced by a **factor of three**









The implementation (python)

- Used library: **TensorFlow** with the **API-KERAS**
- Democratic machine learning: **open ai gym**!
  - Fit into this framework to have a standard and learn from existing environments - visualization…
- Specific RL libraries:
  - **keras-rl** https://github.com/keras-rl/keras-rl
  - Advanced users: TRFL- Reinforcement Learning Building Blocks

Projects on: https://gitlab.cern.ch/RL-group

Summary

- Started to extensively use optimization algorithms in the CERN injector chain in 2018
  - Mainly in LEIR, PS booster, ISOLDE, SPS
- -> game changers for performance, ease of operation and reproducibility
- Next step: reinforcement learning
  - First successful proof-of-principle test for simple process at LEIR injection
  - Sample efficiency is crucial, i.e. online training currently long
  - Operational validity of training to be studied

Outlook

- Collaboration with the University of Malta
- Attack the problem of the sample efficiency - e.g. embedding priors about the world, e.g., intuitive physics
- Develop relevant simulations - model based training
- Upcoming tests on:
    - Linac4
    - AWAKE

Resources

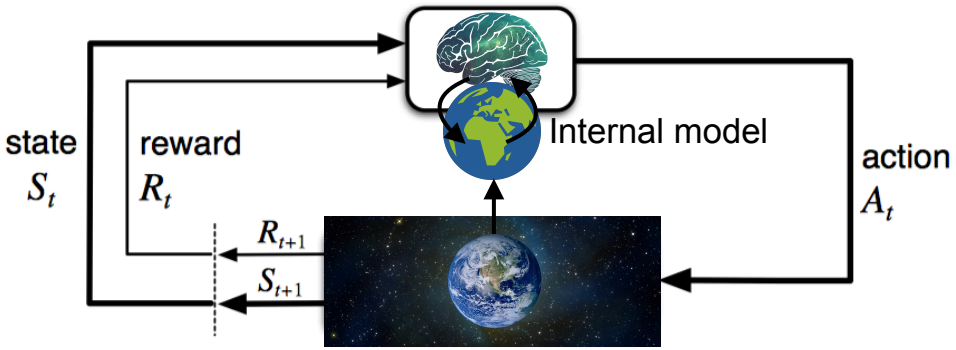
- The classic: Reinforcement Learning: An Introduction (Barto & Sutton 2018)
- A more mathematically rigorous approach: https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf
- Deepmind: https://deepmind.com

Thank you for your time

Appendices

What means model free (in this context)?

- Model free approach: Not treated explicitly probabilities to get from a state **s** to a state **s'** by taking the action **a**
- Interesting if the underlying dynamics changes frequently (e.g. LEIR) - partially observable Markov processes
- Model - based approach → takes information of $\mathbb{P}$ into account and draws simulated samples from an internal constructed model and updates it before acting (the agent can plan - extended MDP - information state included)

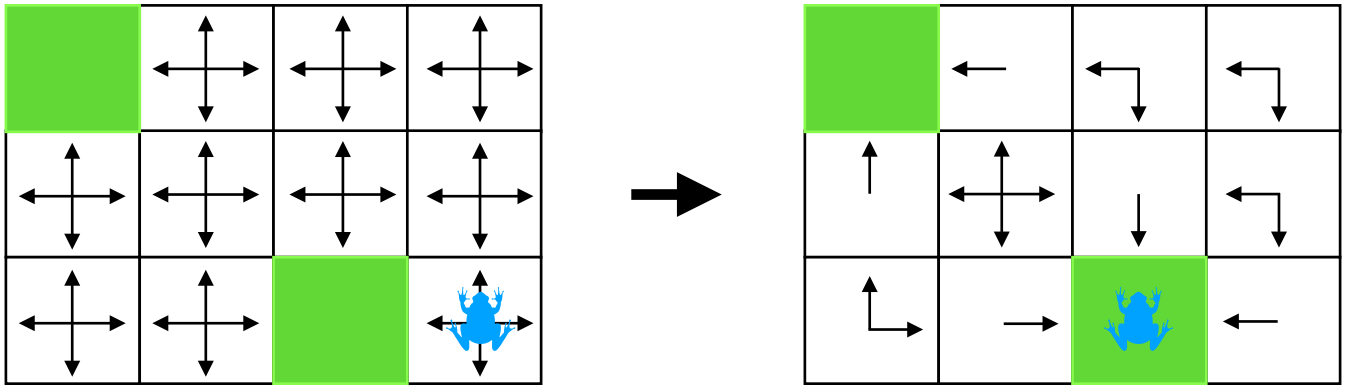


An example (1)

What is a state? What is an action? What is a trajectory?
What is Q?

- Objective: reach the green squares as fast as possible
- The reward is -1 for each step! - short trajectories (sequence of states and actions) and on the green square 0 (could loop forever).

Random policy **Best policy**

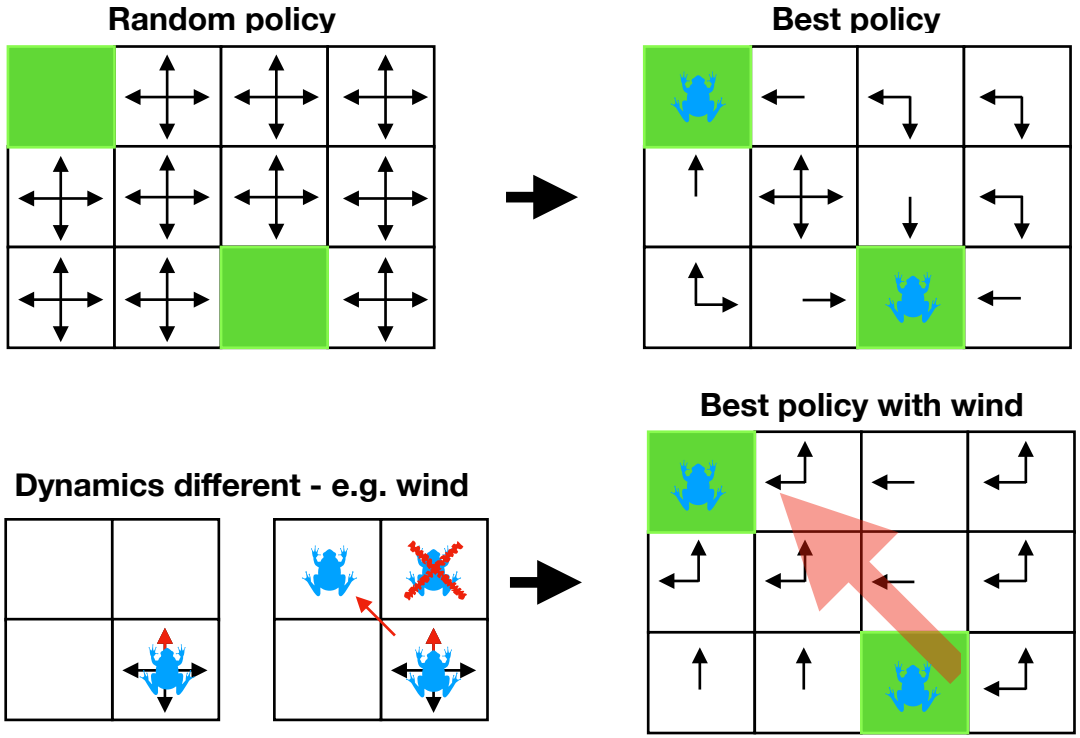


- Objective: Find the policy, which maximizes the (expected) cumulative discounted reward $G_t = \sum_{t \geq 0} \gamma^t R_t$ for all states.

An example (2)

What is the dynamics?

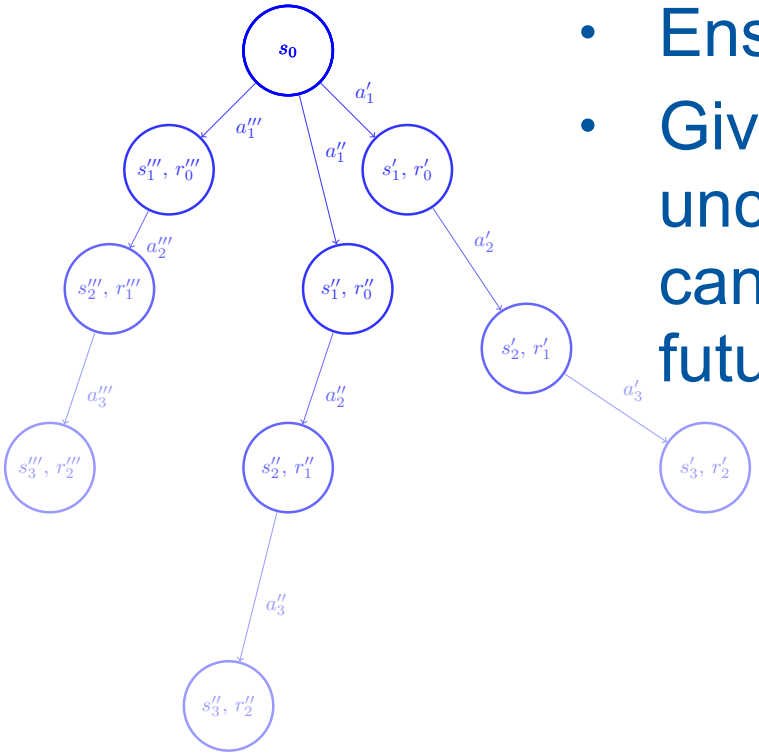
The probability $\mathbb{P}$ to end in a state **s'** after taking an action **a** in state **s**



If the approach is model free - the dynamics is not explicitly taken into account.

Important terms (1)

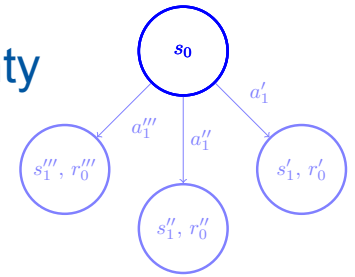
large γ The discount factor: $\gamma \in [0,1]$ $G_t = \sum_{t \geq 0} \gamma^t R_t$



- Ensures convergence in any case
- Gives the possibility to take uncertainty into account - how far can/should we look into the future?

small γ

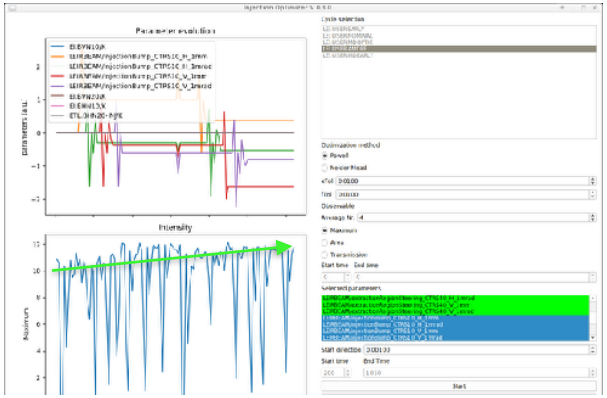
= large uncertainty



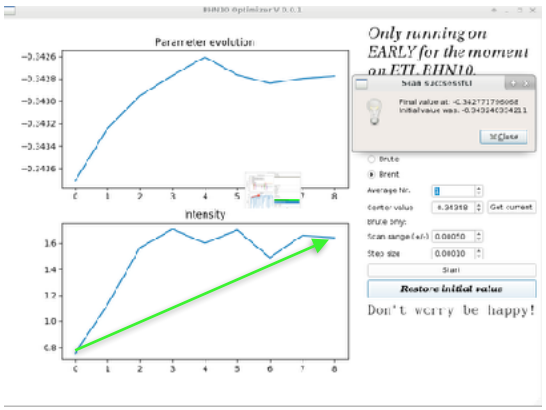
Our operational tool set…

- The methods are implemented - using scipy optimize
- Easy to use by everybody
- Framework easily adaptable to other machines

LEIR: High dimensional optimization of MTI

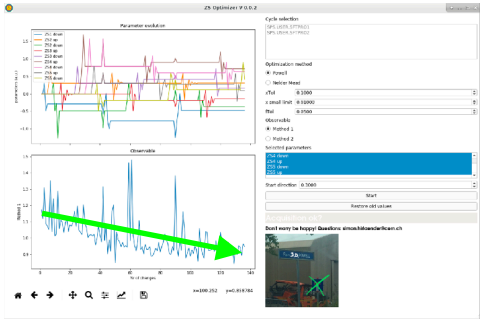


Single parameter constrained optimization (Brent)



Linac3: ion source

SPS: electrostatic septum-alignment



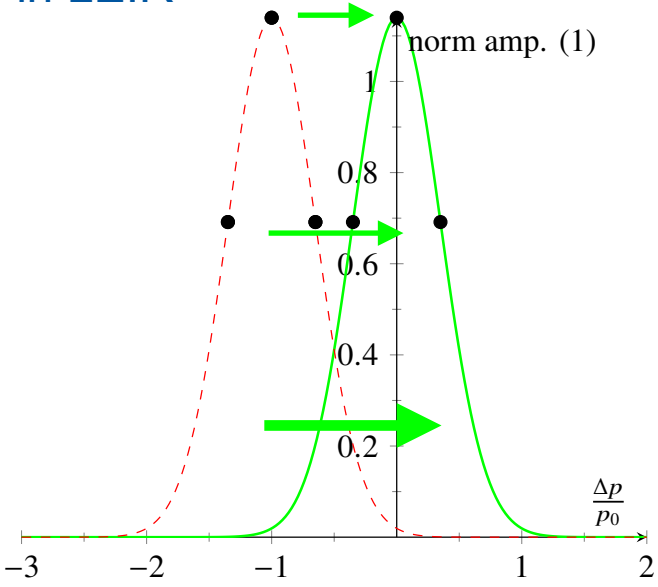
Projects on: https://gitlab.cern.ch/PythonicOperationSoftware

Further reading on scipy optimize
https://www.scipy-lectures.org/advanced/mathematical_optimization/

What else can be done in this way?

- Target function can be a χ^2 function (eg. to a reference) - or moments of a distribution
- Model free extremum seeking stabilization
- …

Energy distribution correction in LEIR



What algorithms to use for optimization in operations

The situation (measurements) is delicate due to two facts:

A. The data is noisy
B. The number of evaluations is limited

- B ⇒ Excludes highly iterative codes
  (simulated annealing, genetic algorithms, differential evolution, particle swarm…)
- A∧B ⇒ Zero order methods

- Consequently:
  - Downhill simplex method (Nelder Mead)
  - Powell method
  - Back up:
    - Basin hopping (non convex — multiple minima)
    - Brute scan - usually non feasible

Enhanced Powell's method - *n* dimensions

- The start is to perform a line-search (e.g. Fibonacci search) in *n* linearly independent directions, taking the optimum along one directions as starting point for the next direction.
- After *n*-steps a check is done if the average direction gives improvement and is not dominated by a specific direction (avoid to loose a specific direction) → if it is satisfied the direction of the greatest decrease is replaced by the average direction.
- The procedure is repeated until the convergence criteria are met.

- The agent learns a part of the system response each time from scratch.
- The start policy improved dynamically.
- A smart scanner.

A constrained policy

- How to protect the hardware/how to handle constrains (borrowed from optimization theory)?
  - Barrier Methods:
  - Penalize for reaching the boundary of an inequality constraint
  - Penalty Methods:
  - **Penalize for violating a constraint — not send to HW**
- How can we handle changes not affecting the hardware?
  - If changes are smaller as a specific minimum the change is not send to HW and the change is treated as indifferent by Powell's method
  - Gives also a reduction of time consumption!

The ingredients in more detail (2):

- **Exploration/Exploitation problem:** Avoid to get stuck in a local optimum - contemporary mathematical issue…
- **Q learning -** off policy (we can choose the policy - still converges):

$$Q^{\pi}(S,A) \leftarrow Q^{\pi}(S,A) + \underbrace{\alpha(R_{t+1} + \gamma \mathbf{max}_{a'} Q^{\pi}(S_{t+1},a') - Q^{\pi}(S,A))}_{\textbf{TD-term}}$$

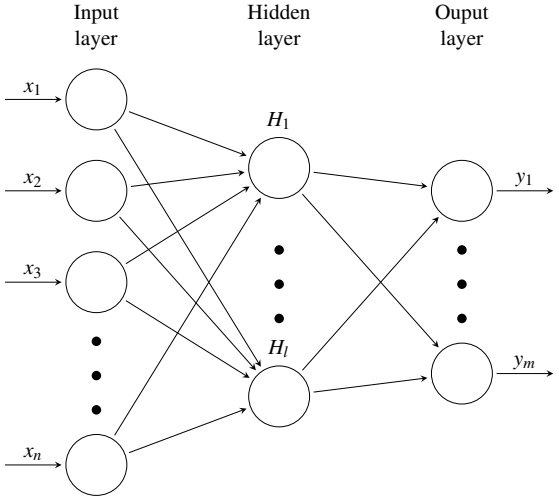
  - An action is taken with a probability $(1-\varepsilon)$ randomly - otherwise greedy (ε-greedy) - random selections slowly excluded.

The ingredients in more detail (3):

- **Approximation:**
  - To handle the number of limited iterations: Use a function approximator to beat “the curse of dimensionality” - generalize!
  - Q is approximated by a (d)eep (n)etwork: (D)Q(N) - deep learning
    - The TD term is used to update the network
- **Stability**:
  - The data is correlated due to the fact that it is generated by trajectories - stability issues
  - Solution: Experienced Replay:
    - The trajectory is stored and shuffled - correlation is reduced → better stability

$$f(x_1, x_2, x_3 \ldots, x_n) = (y_1, y_2, y_3 \ldots, y_m)$$

$$\approx$$



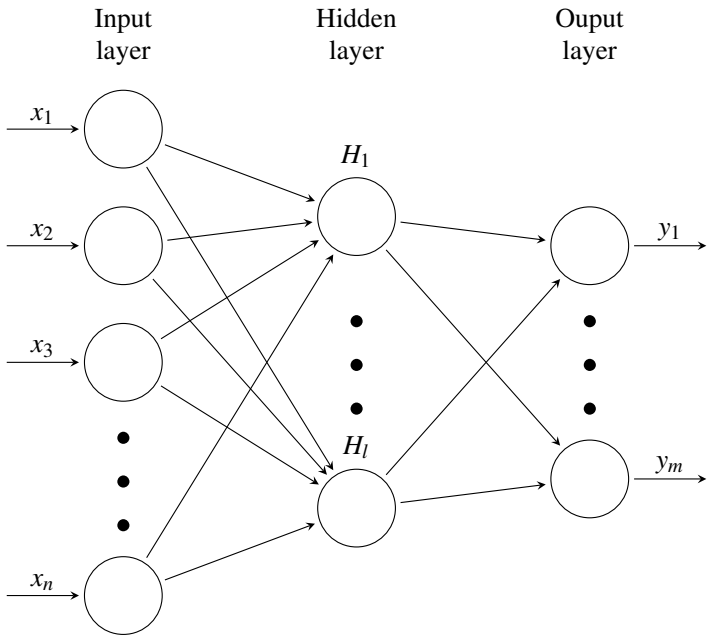
The ingredients in more detail (4):

Recap - Why a neural network? Generalization!

- Universal function approximator theorem: Any continuous function can be approximated with a finite ANN with one hidden layer using a nonlinear activation function → projection in the vector space of nodes.
- The state space might be intractable - we cannot visit every state - the function is "fitted".

$$f(x_1, x_2, x_3 \ldots, x_n) = (y_1, y_2, y_3 \ldots, y_m)$$

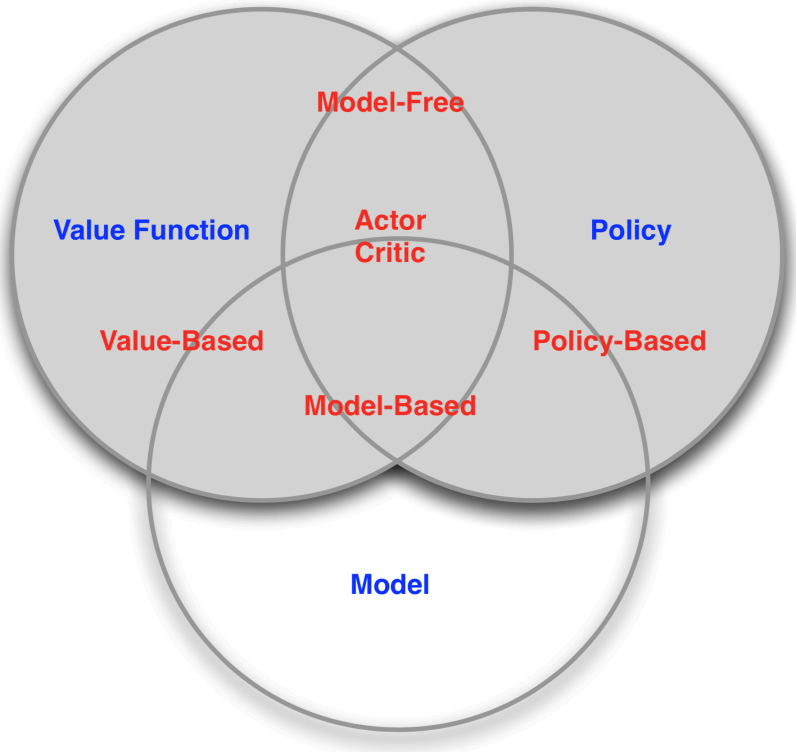
$$\approx$$



A network with more than one hidden layer is called deep network

Back to theory: Global vs. local strategies to find the best policy

- **Finding the best global policy directly - policy gradients:**
  - Monte Carlo - non biased but suffers from high variance so requires a lot of samples.
  - Not scalable + must compute V or Q for every state (iteratively to the end).
  - **Challenge**: sample-efficiency
- **Finding the best policy iteratively temporal difference learning:**
  - The **Hamilton–Jacobi–Bellman** equation is used as an update. What is the problem with this?
    - Biased - the estimates of estimates…
    - **Challenge**: bias
- Alternatives: Genetic algorithms, swarm optimization, etc…



Taken from http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching_files/pg.pdf

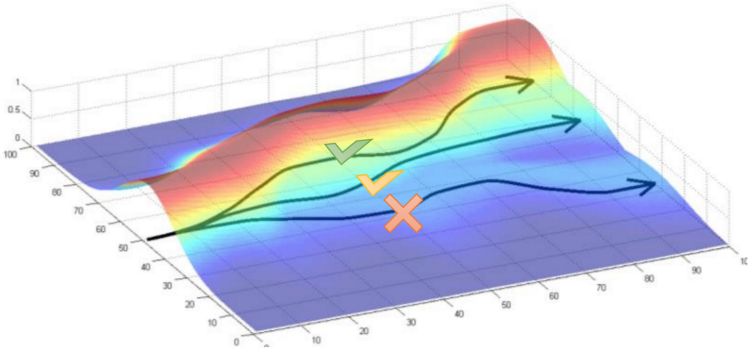
Some words about policy gradient

$$J(\theta) = E_{\tau \sim \pi_\theta}[\sum_t r_t(s_t, a_t)]$$

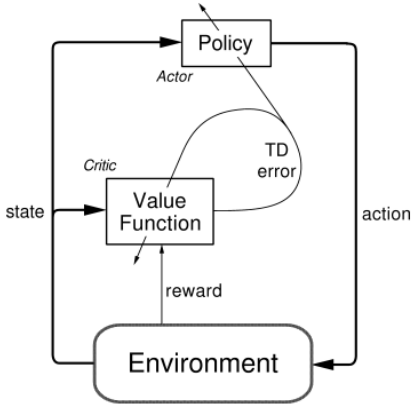
- The policy is parametrized directly and the gradient is taken as an update:

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_i^N \sum_t \nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) \sum_t r_t(s_{i,t}, a_{i,t})$$

  - Challenges: High variance
- Attempts to solve this:
  - Baseline - Advantage function
- Unifying TD and policy gradient - actor-critic small bias/variance
- Used to solve continuous problems



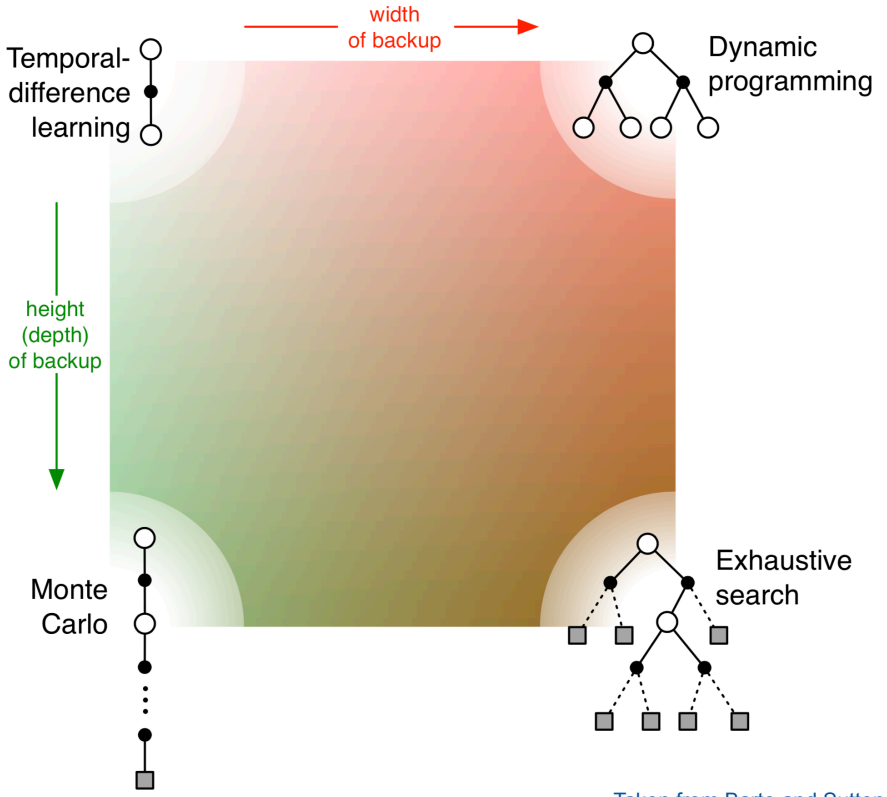
Taken from http://rll.berkeley.edu/deeprlcourse/f17docs/lecture_4_policy_gradient.pdf



Taken from https://cs.wmich.edu/~trenary/files/cs5300/RLBook/node66.html

A unified view - best of both worlds

- There is a continuum of one step - temporal difference (TD($\lambda = 0$)) to TD(λ) - to complete episodes ($\lambda = 1$) - Monte Carlo
- From myopic to farsighted
- To be tested to find optimum in simple efficiency



Taken from Barto and Sutton

A new forum focus the efforts on machine learning and advanced controls is needed:

- Meeting to share experiences and get advices and learn new technologies
- Provide a general framework to use the technology
- Centralize most popular tools as the optimizers — some features are distributed to all applications, some features stay individual — collecting data for models…
- Extend the functionality of the optimizers to learn specific task using reinforcement learning


CERN