Reinforcement Learning in Feedback Control

Mach Learn (2011) 84:137–169
DOI 10.1007/s10994-011-5235-x

Reinforcement learning in feedback control

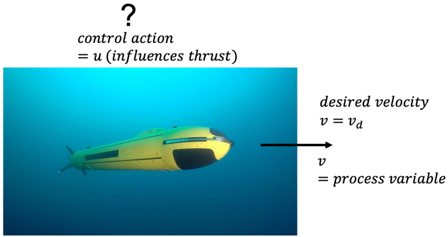
Challenges and benchmarks from technical process control

Roland Hafner · Martin Riedmiller

Overview

Technical Process Control

Example : Underwater vehicle control



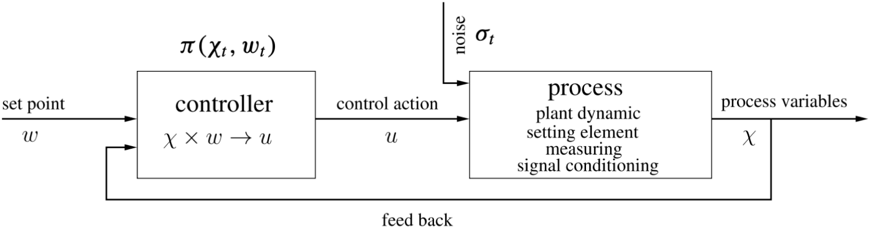
Application areas : Aircraft control, chemical plants, air conditioners, magnetic levitation trains, etc.

Shortcomings of Classical Design Process

- Tedious and Demanding
- Involves assumptions and simplification of systems

Solution: Learning controllers by interacting with the process using RL

Feedback Control



$$\chi_{t+1} = f(\chi_t, u_t, \sigma_t)$$

General form of time discrete dynamic systems

Overview

Prior Work

- System Identification
  (Ljung 1999; Goodwin and Payne 1977;Nelles 2001)
- Parametric Models - based on the physical properties of the process
- Non-Parametric Models e.g. neural networks
  (Sjberg et al. 1995)
- Robust Control
  (Dullerud 2000)

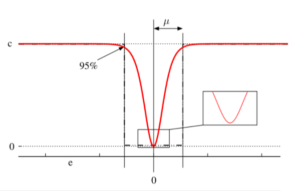
Why Reinforcement Learning Controllers ?

- Controller learns from real process behaviour
- does not suffer from model inaccuracy or simplifications in design process
- Data sampling incorporated within learning process

RL formulation for feedback control

- MDP State : $x_t = [\chi_t, e_t]$
- χ_t : process variables
- $e_t = w_t - y_t$
- w_t = reference set point
- y_t = controlled process variables
- Actions : $u_t = \pi([\chi_t, w_t - y_t])$
- Observed Transitions : $([\chi_t, w_t - y_t], u_t, [\chi_{t+1}, w_t - y_{t+1}])$ does not include change in set point

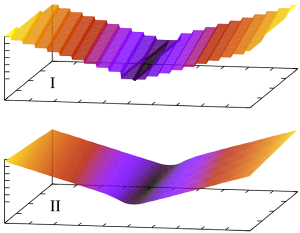
Choice of immediate cost function



$$c(x,u) = c(e) = \begin{cases} 0 & |e| < \mu \\ \mathcal{C} & \text{else} \end{cases}$$

$$c(x,u) = c(e) = \tanh^2(|e| * w) * \mathcal{C}$$

$$w = \tanh^{-1}\left(\frac{\sqrt{0.95}}{\mu}\right)$$



Overview

NFQCA Algorithm

Recap : NFQ

- Q-function as neural network
- $\hat{Q}_{x,u} = c(x,u) + \min_b Q_k(x',b)$ (for every iteration)
- Train using new set of $((x,u), \hat{Q}_{x,u})$

NFQCA Algorithm

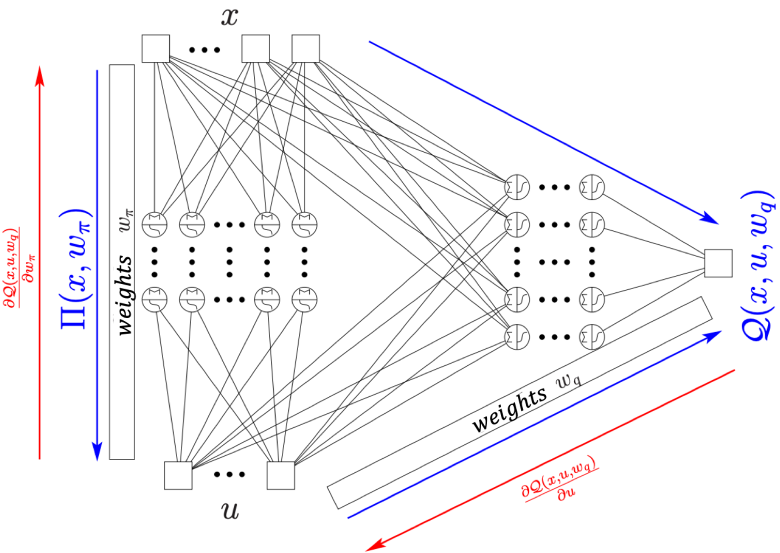
Recap : NFQ

- Q-function as neural network
- $\hat{Q}_{x,u} = c(x,u) + \min_b Q_k(x', b)$ (for every iteration)
- Train using new set of $((x,u), \hat{Q}_{x,u})$

Neural Fitted Q-Iteration with Continuous Actions

- Fitted Actor-Critic algorithm
- Value function based
- continuous state and action space
- Neural policy function $\pi(x, w_\pi)$
- $\pi_k(x) \approx \mathrm{argmin}_u(Q_k(x,u))$
- $\hat{Q}(x,u) = c(x,u) + Q_k(x', \pi_k(x'))$

NFQCA Actor-Critic Network



Actor-Fitted Training

Algorithm 1: $AktorFittedTraining(\mathcal{Q}, \mathfrak{D})$

Require: $\mathcal{Q}$: *Approximation of the Q function* $\mathfrak{D}$: *set of observed transitions*
Ensure: $\Pi(x)$: *neural representation of the greedy strategy regarding* $\mathcal{Q}$
  $\Pi_0 \Leftarrow$ *neural network with randomly initialized weights*
  $k \Leftarrow 0$
  repeat
    for $i = 0, \ldots, \sharp\mathfrak{D}$ **do**
      $u = forward\_propagate(\Pi_k, x^i)$
      $q = forward\_propagate(\mathcal{Q}, x^i, u)$
      $(\frac{\partial q}{\partial x_1^i}, \ldots, \frac{\partial q}{\partial x_n^i}, \frac{\partial q}{\partial u_1}, \ldots, \frac{\partial q}{\partial u_m}) = backward\_propagate(\mathcal{Q}, 1)$
      $backward\_propagate(\Pi, (\frac{\partial q}{\partial u_1}, \ldots, \frac{\partial q}{\partial u_m}))$ *cumulate* $\frac{\partial q}{\partial w_\pi}$ *for all weights*
    end for
    $\Pi_{k+1} \Leftarrow update\_weights(\Pi_k)$*with*RProp *gradient descent*
    $k \Leftarrow k + 1$
  until *converged*
  return Π_k

Batch QCA Update

Algorithm 2 : BatchQCAUpdate($\mathcal{Q}, \Pi, \mathfrak{D}$)

Require: $\mathcal{Q}$: *Approximation of the Q function*
Require: Π: *Neural approximation of the greedy strategy with respect to Q*
Require: $\mathfrak{D}$: *Amount of observed transitions*
Ensure: $\mathcal{P}$: *Training data set with updated Q values*

$\mathcal{P} \Leftarrow \emptyset$
for $i = 1, \ldots, \sharp\mathfrak{D}$ **do**
with $d_i = (x_i, u_i, x'_i, c_i)$ i^{th} Element $\in \mathfrak{D}$
$\hat{Q} = (1 - \alpha) * \mathcal{Q}(x_i, u_i) + \alpha * (c_i + \gamma \mathcal{Q}(x'_i, \Pi(x'_i))$
$\mathcal{P} \Leftarrow \mathcal{P} \cup ((x_i, u_i), \hat{Q})$
end for
return $\mathcal{P}$

NFQCA Update

***Algorithm* 3 :** NFQCAUpdate($\mathcal{Q}_k$, Π_k, $\mathfrak{D}$)

Require: $\mathcal{Q}_k$: *neural approximation of the Q function* , Π_k: *neural approximation of the strategy*

Require: $\mathfrak{D}$: *Amount of observed transitions*

Ensure: $\mathcal{Q}_{k+1}$: *updated neuronal approximation of the Q function*

Ensure: Π_{k+1}: *updated neuronal approximation of the strategy*

$\mathcal{P} = BatchQCAUpdate(\mathcal{Q}_k, \Pi_k, \mathfrak{D})$

$\mathcal{Q}_{k+1} = RpropTraining(\mathcal{Q}_0, \mathcal{P})$

$\Pi_{k+1} = AktorFittedTraining(\mathcal{Q}_{k+1}, \mathfrak{D})$

return $\mathcal{Q}_{k+1}$, Π_{k+1}

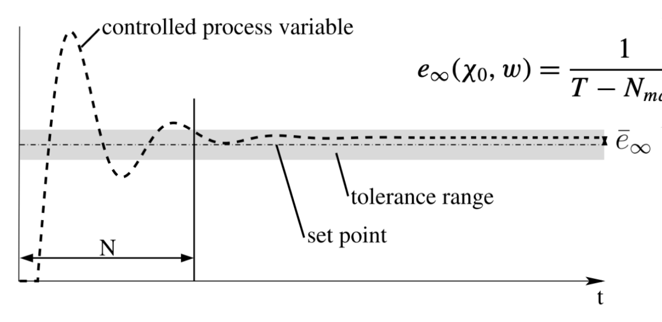
Overview

Benchmarks

Table 1 Proposed benchmark tasks and to what extent they shed light on the respective properties

Property	Underwater Vehicle	Pitch Control	Magnetic Levitation	Heating Coil
nonlinear dynamics	+++		+++	++
long-range dynamics		+++	+	+
precise control	++	++	+++	+
changing setpoints	+++	+++	+++	+++
external variables				+++

Performance Evaluation



$$e_\infty(\chi_0, w) = \frac{1}{T - N_{max}} \sum_{t=N_{max}}^{T} |y_t - w|$$

$$\bar{e}_\infty = \tfrac{1}{J} \textstyle\sum_{j=1}^{J} e_\infty(\chi_0^j, w^j) \text{ and } \bar{N} = \tfrac{1}{J} \textstyle\sum_{j=1}^{J} N(\chi_0^j, w^j).$$

$$e_T = \frac{1}{T_{traj}} \sum_{t=0}^{T_{traj}} |y_t - w(t)|$$

Results - Underwater Vehicle

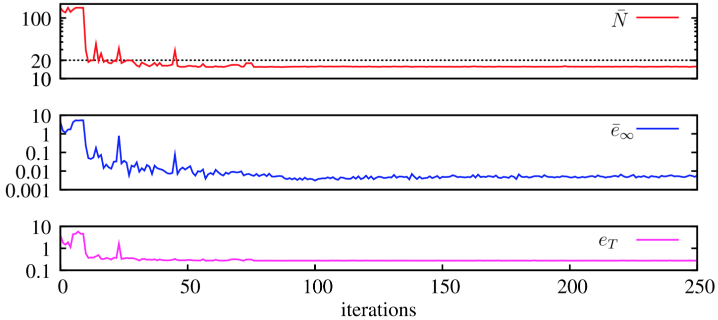
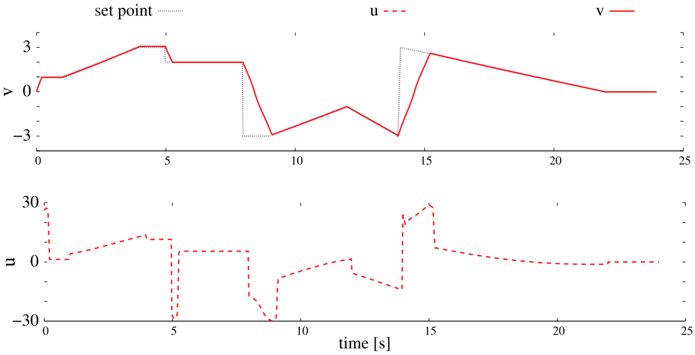


Table 5 Benchmark evaluation for the underwater vehicle control challenge. Smaller values in the evaluation criteria mean better performance of the controller. For comparison, the results of a bang-bang controller with minimal and maximal actions are shown

Controller	$\bar{N}$	$\bar{e}_\infty$	e_T
bang-bang	20.04	0.131	0.65
NFQ	18.22	0.054	0.34
NFQCA	15.68	0.003	0.27

NFQCA on set-point trajectory of underwater vehicle task



Results - Underwater Vehicle

Learning Performance:

NFQCA

$\approx 100 iterations = 5000 interactions \approx 2.5 minutes$

NFQ with 5 discrete actions

$\approx 140 iterations = 7000 interactions \approx 3.5 minutes$

Overview

Discussion

Main Contributions :

- Four benchmarking scenarios in process control
- Quantitative performance measures for controller and learning performance
- NFQCA as a baseline for feedback control RL problems
- Learn high quality, continuous, non-linear control laws in real time for real world applications

References

Hafner, R. (2009) Dateneffiziente selbstlernende neuronale Regler
PhD thesis, University of Osnabrueck, 2009

Questions?