Deep learning algorithms for stochastic control and applications to energy storage problems

Huyên PHAM*

*University Paris Diderot, LPSM

Joint work with
A. Bachrouf, University of Oslo
C. Huré, University Paris Diderot, LPSM
N. Langrené, CSIRO, Data61, Risklab Australia

CAESARS 2018
Saclay, september 3-5, 2018

Discrete-time stochastic control on finite horizon

Markov Decision Process (MDP)

- **State process** $X = (X_n)_n$ in $\mathcal{X} \subset \mathbb{R}^d$, $n = 0, \ldots, N$

- Controlled by $\alpha = (\alpha_n)_n$ **action/policy**: $\alpha_n = \pi_n(X_n)$ for some measurable sequence $\pi_n : \mathcal{X} \to \mathbb{A}$, $n = 0, \ldots, N-1$.

- **State dynamics** in a random environment: $X = X^\alpha$

$$X_{n+1} \quad = \quad F_n(X_n, \alpha_n, \varepsilon_{n+1})$$

$\leftrightarrow$ One-step transition probabilities:

$$\begin{aligned} P_n^a(x, dx') &= \mathbb{P}\big[X_{n+1}^\alpha \in dx' | X_n^\alpha = x, \alpha_n = a\big] \\ &= \mathbb{P}\big[F_n(x, a, \varepsilon_{n+1}) \in dx'\big] \end{aligned}$$

- **Reward**: running reward $f_n(x, a)$ and terminal reward $g(x)$

Performance criterion

$$J_n(x,\alpha) \;=\; \mathbb{E}\Big[\sum_{k=n}^{N-1} f_n(X_n^\alpha,\alpha_n) + g(X_N^\alpha)\Big|X_n^\alpha = x\Big]$$

▶ **Goal**: Find optimal performance V and optimal action/policy $\alpha^* \leftrightarrow \pi^* = (\pi_n^*)_n$ valued in $\mathbb{A}^{\mathcal{X}}$:

$$V_n(x) \;:=\; \sup_\alpha J_n(x,\alpha) \;=\; J_n(x,\alpha^*), \quad n = 0,\ldots,N,\; x \in \mathcal{X}.$$

Remark:

- MDP can also be viewed as time discretization of continuous-time stochastic control problem $\leftrightarrow$ Bellman PDE

Dynamic Programming (DP) Bellman equation

From global to local optimization: Backward recursion on $V = (V_n)$ (Value function iteration)

$$\begin{cases} V_N(x) & = & g(x) \\ V_n(x) & = & \sup\limits_{a \in \mathbb{A}} \Big\{ \underbrace{f_n(x,a) + \mathbb{E}\big[V_{n+1}(X^\alpha_{n+1}) \big| X^\alpha_n = x, \alpha_n = a\big]}_{Q_n(x,a) \,:=\, f_n(x,a) + P^a_n V_{n+1}(x)} \Big\}, \quad n = N-1, \ldots, 0. \end{cases}$$

$\longrightarrow$ Optimal policy: $\pi^* = (\pi^*_n)_n$ from the Q-value function

$$\pi^*_n(x) \quad \in \quad \arg\max_{a \in \mathbb{A}} Q_n(x,a), \quad n = N-1, \ldots, 0$$

Remark.
$\{V_n(X^*_n) + \sum_{k=0}^n f(X^*_n, \alpha^*_n),\ n = 0, \ldots, N\}$, is a martingale:

$$V_n(x) \quad = \quad f_n(x, \pi^*_n(x)) + P_n^{\pi^*_n(x)} V_{n+1}(x).$$

Numerical challenges

Two sources of curses of dimensionality:

- Computations of the conditional expectation operator $P_n^a V_{n+1}(x)$, $n = 0, \ldots, N-1$, for any $x \in \mathcal{X} \subset \mathbb{R}^d$, and $a \in \mathbb{A}$. Computational complexity in **high-dimension for the state space** $\mathbb{R}^d$ and also the control space $\mathbb{A}$!
- Computation of the optimal policy: Supremum in $a \in \mathbb{A}$ of $Q_n(x,a) = f_n(x,a) + P_n^a V_{n+1}(x)$, for each $x \in \mathcal{X}$: $\rightarrow$ optimal policy $\hat{\pi}(x)$. Computational complexity in **high dimension for the control space** $\mathbb{A}$!

Probabilistic numerical methods based on DP

(i) Approximate the Q-value function (conditional expectation) by Monte-Carlo regression on: basis functions, neural networks, SVM, etc

- MC regression in the spirit of Longstaff-Schwartz (for optimal stopping problems)
- Main issue: simulation of the endogenous controlled process

(ii) Optimal control is then "computed" from $\arg\max_{a \in \mathbb{A}} \hat{Q}_n(x, a)$ where $\hat{Q}$ is an approximation of the Q-value function. Typically:

- $\mathbb{A}$ finite set, or discretize $\mathbb{A}$
- Newton method for the search of extremum

Numerical methods by direct approximation (without DP)

• **Control approximation**: Focus directly on the (parametric) approximation $\pi = (\pi_n)$ of the policy on the whole period

$$\pi_n(x) \quad = \quad A(x;\theta_n), \quad n = 0,\dots,N-1,$$

for some given function $A(.,\theta)$ with parameters $\theta = (\theta_0,\dots,\theta_{N-1}) \in \mathbb{R}^{q\times N}$ $\rightarrow$ maximize over θ

$$J_0\big(x_0, A(.;\theta_0),\dots,A(.;\theta_{N-1})\big) \quad = \quad \mathbb{E}\Big[\sum_{n=0}^{N-1} f_n(X_n, A(X_n;\theta_n)) + g(X_N)\Big].$$

- Kou, Peng, Xu (16): E-M algorithm with basis functions for A
- J. Han, W. E, A. Jentzen (17): Deep neural network (DNN) for A and global optimization by stochastic gradient descent (SGD), see also P. Henry-Labordère (18)

Our approach and contributions

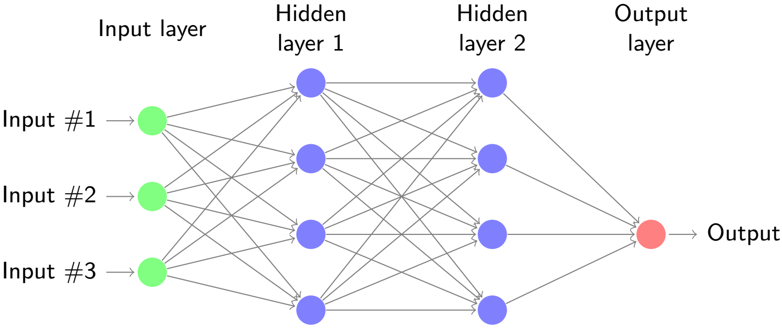
- Combine different ideas from maths (numerical probability) and computer science communities (reinforcement learning) to propose (and compare) three algorithms based on:
  - Dynamic programming (DP)
  - Deep Neural Networks (DNN) for the approximation/learning of
    - (i) Optimal policy
    - (ii) Value function
  - Monte-Carlo regressions with different characteristics:
    - Performance/policy iteration (PI) or hybrid iteration (HI)
    - Now or later/quantization

- Convergence analysis

- Numerical tests and an application to energy storage problems

Deep Neural networks (DNN): multilayer perceptron

Architecture of a DNN: composed of layers and neurons (units)



(Feedforward artificial NN)

Mathematical representation of DNN

- DNN: composition of simple functions to approximate complicated ones $\neq$ usual additive approximation theory

▶ Represented by parametrized function:

$$
\begin{aligned}
x \in \mathbb{R}^d \quad \mapsto \quad \Phi(x;\theta) &= \mathcal{L}^{out} \circ \mathcal{L}^L \circ \ldots \mathcal{L}^1(x), \\
\Phi_\ell \quad = \quad \mathcal{L}^\ell \Phi_{\ell-1} &:= \sigma(w_\ell \Phi_{\ell-1} + b_\ell) \in \mathbb{R}^{d_\ell},
\end{aligned}
$$

with L hidden layers (layer ℓ with d_ℓ units), activation function σ (Sigmoid, ReLu, ELU, etc), and weights $\theta = (w_\ell, b_\ell)_\ell$.

Mathematical representation of DNN

• DNN: composition of simple functions to approximate complicated ones $\neq$ usual additive approximation theory

▶ Represented by parametrized function:

$$\begin{aligned} x \in \mathbb{R}^d \quad \mapsto \quad \Phi(x;\theta) &= \mathcal{L}^{out} \circ \mathcal{L}^L \circ \ldots \mathcal{L}^1(x), \\ \Phi_\ell \quad = \quad \mathcal{L}^\ell \Phi_{\ell-1} &:= \sigma(w_\ell \Phi_{\ell-1} + b_\ell) \in \mathbb{R}^{d_\ell}, \end{aligned}$$

with L hidden layers (layer ℓ with d_ℓ units), activation function σ (Sigmoid, ReLu, ELU, etc), and weights $\theta = (w_\ell, b_\ell)_\ell$.

- Theoretical justification by universal approximation theorem (Hornick 91). Rate of convergence not yet well understood (partial results in the case of one hidden layer, see Bach 17).
- Key feature: automatic differentiation for computing derivatives of Φ used in SGD to find the "optimal" parameters.

Algo NNContPI: control learning by performance iteration

A combination of DP and Han, E, Jentzen algo:

- For $n = N-1, \ldots, 0$: keep track of the approximated optimal policies $\hat{\pi}_k$, $k = n+1, \ldots, N-1$, and compute

$$\hat{\pi}_n \quad \in \quad \arg\max_{\pi} \mathbb{E}\Big[f_n(X_n, \pi(X_n)) + \underbrace{\sum_{k=n+1}^{N-1} f_k(\hat{X}_k, \hat{\pi}_k(\hat{X}_k)) \; + g(\hat{X}_N)}_{\hat{Y}^{\pi}_{n+1}} \Big]$$

where $X_n \rightsquigarrow \mu$ (probability distribution on $\mathcal{X}$), $(\hat{X}_k)_{k=n+1,\ldots,N}$, generated from X_n, with control $(\pi, \hat{\pi}_k)_{k=n+1,\ldots,N-1}$. $\rightarrow$ Practical implementation:

- DNN for policy: $\pi(x) = A(x; \beta)$ $\rightarrow$ optimization over parameter β
- SGD based on training samples $X_n^{(m)}$, $(\hat{X}_k^{(m)})_k$, $m = 1, \ldots, M$ $\rightarrow$ $\hat{\pi}_n^M = A(.; \hat{\beta}_n^M)$.

Algo NNContPI: control learning by performance iteration

A combination of DP and Han, E, Jentzen algo:

- For $n = N-1, \ldots, 0$: keep track of the approximated optimal policies $\hat{\pi}_k$, $k = n+1, \ldots, N-1$, and compute

$$\hat{\pi}_n \quad \in \quad \arg\max_{\pi} \mathbb{E}\Big[f_n(X_n, \pi(X_n)) + \underbrace{\sum_{k=n+1}^{N-1} f_k(\hat{X}_k, \hat{\pi}_k(\hat{X}_k)) \; + g(\hat{X}_N)}_{\hat{Y}^{\pi}_{n+1}} \Big]$$

where $X_n \rightsquigarrow \mu$ (probability distribution on $\mathcal{X}$), $(\hat{X}_k)_{k=n+1,\ldots,N}$, generated from X_n, with control $(\pi, \hat{\pi}_k)_{k=n+1,\ldots,N-1}$. $\rightarrow$ Practical implementation:

- DNN for policy: $\pi(x) = A(x;\beta) \rightarrow$ optimization over parameter β
- SGD based on training samples $X_n^{(m)}$, $(\hat{X}_k^{(m)})_k$, $m = 1, \ldots, M \rightarrow \hat{\pi}_n^M = A(.;\hat{\beta}_n^M)$.

Remarks.
1) **No value function iteration**: $\hat{V}_n^M$ is simply computed as the gain functional associated to controls $(\hat{\pi}_k^M)_{k=n,\ldots,N-1}$.
2) Low bias estimate, but possibly high variance estimate and large complexity, especially when N is large.

Algo Hybrid: control learning by hybrid iteration

- Initialization: $\hat{V}_N = g$

- For $n = N-1, \ldots, 0$:

  (i) Compute the approximated optimal policy

$$\hat{\pi}_n \quad \in \quad \arg\max_{\pi} \mathbb{E}\big[f_n(X_n, \pi(X_n)) + \hat{V}_{n+1}(X^{\pi}_{n+1})\big]$$

  where $X_n \rightsquigarrow \mu$, $X^{\pi}_{n+1} \rightsquigarrow P_n^{\pi(X_n)}(X_n, dx')$. Implemented by

  - DNN for policy: $\pi(x) = A(x;\beta) \rightarrow$ optimization over parameter β
  - SGD method based on training samples $X_n^{(m)}$, $m = 1, \ldots, M \rightarrow \hat{\pi}_n^M = A(.;\hat{\beta}_n^M)$.

Algo Hybrid: control learning by hybrid iteration

- Initialization: $\hat{V}_N = g$

- For $n = N-1, \ldots, 0$:

(i) Compute the approximated optimal policy

$$\hat{\pi}_n \quad \in \quad \arg\max_{\pi} \mathbb{E}\big[f_n(X_n, \pi(X_n)) + \hat{V}_{n+1}(X^{\pi}_{n+1})\big]$$

where $X_n \rightsquigarrow \mu$, $X^{\pi}_{n+1} \rightsquigarrow P_n^{\pi(X_n)}(X_n, dx')$. Implemented by

- DNN for policy: $\pi(x) = A(x; \beta) \rightarrow$ optimization over parameter β
- SGD method based on training samples $X_n^{(m)}$, $m = 1, \ldots, M \rightarrow \hat{\pi}_n^M = A(.; \hat{\beta}_n^M)$.

(ii) Updating: compute the approximated value function

$$\begin{aligned} \hat{V}_n(x) &= \mathbb{E}\Big[f_n(X_n, \hat{\pi}_n(X_n)) + \hat{V}_{n+1}(X^{\hat{\pi}_n}_{n+1})\Big| X_n = x\Big] \\ &= f_n(x, \hat{\pi}_n(x)) + P_n^{\hat{\pi}_n(x)} \hat{V}_{n+1}(x) \end{aligned}$$

by Monte-Carlo regression: now or later

Algo Hybrid-Now

Regress now on a set $\mathcal{F}$ of functions on $\mathcal{X}$ (from $n+1$ to n)

$$\hat{V}_n \quad \in \quad \arg\min_{\Phi \in \mathcal{F}} \mathbb{E}\Big| f_n(X_n, \hat{\pi}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{\pi}_n}) \; - \; \Phi(X_n)\Big|^2$$

- For instance, $\mathcal{F}$ class of DNN: $x \mapsto \Psi(x; \theta)$
- Optimization over θ by SGD based on training samples $X_n^{(m)} \rightsquigarrow \mu$, $m = 1, \ldots, M$, $\rightarrow$ $\hat{V}_n^M = \Psi(.; \hat{\theta}_n^M)$.

Algo Hybrid-LaterQ

Quantization + **Regress later** on a set $\mathcal{F}$ of functions on $\mathcal{X}$:

- Approximate analytically by quantization the conditional expectation

$$\tilde{V}_n(X_n) := f_n(X_n, \hat{\pi}_n(X_n)) + \tilde{P}_n^{\hat{\pi}_n(X_n)} \hat{V}_{n+1}(X_n)$$
$$:= f_n(X_n, \hat{\pi}_n(X_n)) + \sum_{j=1}^{J} p_j \hat{V}_{n+1}\big(F_n(X_n, \hat{\pi}_n(X_n), e_j)\big)$$

  where $\tilde{\varepsilon}_{n+1} \rightsquigarrow \sum_{j=1}^{J} p_j \delta_{e_j}$ is a J-quantizer of ε_{n+1}.

- Regress $\tilde{V}_n$ on a set $\mathcal{F}$ of functions on $\mathcal{X}$ (e.g. DNN):

$$\hat{V}_n \in \arg\min_{\Phi \in \mathcal{F}} \mathbb{E}\Big[\ell\big(\tilde{V}_n(X_n) - \Phi(X_n)\big)\Big]$$

  for some loss function ℓ on $\mathbb{R}$, e.g., $\ell(y) = y^2$.

Remark. Compared to Regress now, Regress Later MC reduces the variance of the estimated $\hat{V}_n^M$.

Case of finite control space: classification

- Card($\mathbb{A}$) $= L < \infty$: $\mathbb{A} = \{a_1, \ldots, a_L\}$

- Randomize the control: given a state value x, the controller chooses a_ℓ with a probability $p_\ell(x)$
  - (Deep) Neural Network for the probability vector $p = (p_\ell)_\ell$ with softmax output layer:

$$z \quad \longmapsto \quad p_\ell(z;\beta) \;=\; \frac{\exp(\beta_\ell.z)}{\sum_{\ell=1}^L \exp(\beta_\ell.z)}, \quad \ell = 1, \ldots, L.$$

  - Optimization over the probability vector p via the parameter β

Remark. In practice, we then use pure control strategies: given a state value x, choose $a_{\ell^*(x)}$ with

$$\ell^*(x) \quad \in \quad \arg\max_{\ell=1,\ldots,L} p_\ell(x).$$

Convergence of the algo NNcontPI

- M number of training samples
- Neural Network for policy:
  - $\mathcal{A}_K^\gamma$: class of NN with one hidden layer, K **neurons**, and **total variation norm** smaller than γ

Theorem. Under suitable conditions, and assuming the existence of an optimal policy $(\pi_k^*)_k$, we have for all $n = 0, \ldots, N-1$:

$$\mathbb{E}_M \big| V_n(X_n) - \hat{V}_n^M(X_n) \big| \quad = \quad \mathcal{O}_{\mathbb{P}}\Big(\gamma \sqrt{\frac{\ln M}{M}} + \underbrace{\sup_{k=n,\ldots,N-1} \inf_{A \in \mathcal{A}_K^\gamma} \big\| A(X_k) - \pi_k^*(X_k) \big\|_{L^1}}_{\varepsilon_n^{NN}(A)} \Big),$$

where $\mathbb{E}_M$ stands for the expectation conditioned on the training set used for computing the approximated optimal policies $\hat{\pi}_k^M$, and $(X_k)_k$ is the corresponding controlled process starting from $X_n \rightsquigarrow \mu$.

Proof. Arguments from statistical learning theory: Györfi et al. (02)

Convergence of the algo Hybrid

- M number of training samples
- Neural Network for policy and value function:
  - $\mathcal{A}_K^\gamma$: class of NN (valued in $\mathbb{A}$) with one hidden layer, K **neurons**, and **total variation norm** smaller than γ
  - $\mathcal{F}_K^\gamma$: class of NN (valued in $\mathbb{R}$) with one hidden layer, K **neurons**, and **total variation norm** smaller than γ

Theorem. Under suitable conditions, and assuming the existence of an optimal policy $(\pi_k^*)_k$, we have for all $n = 0, \ldots, N-1$:

$$\mathbb{E}_M \big| V_n(X_n) - \hat{V}_n^M(X_n) \big| \quad = \quad \mathcal{O}_{\mathbb{P}}\Big(\gamma^2 \sqrt{K \frac{\ln M}{M}} + \underbrace{\sup_{k=n,\ldots,N-1} \inf_{A \in \mathcal{A}_K^\gamma} \big\| A(X_k) - \pi_k^*(X_k) \big\|_{L^1}}_{\varepsilon_n^{NN}(A)}$$

$$+ \underbrace{\sup_{k=n,\ldots,N} \inf_{\Psi \in \mathcal{F}_K^\gamma} \big\| \Psi(X_k) - V_k(X_k) \big\|_{L^2}}_{\varepsilon_n^{NN}(V)} \Big).$$

A semi-linear PDE with quadratic gradient term

$$\begin{cases} \frac{\partial v}{\partial t} + \Delta_x v - |D_x v|^2 = 0, & (t,x) \in [0,T) \times \mathbb{R}^d \\ v(T,x) = g(x) \end{cases}$$

This PDE can be written as an HJB equation associated to a stochastic control problem whose discrete-time version (time step $h = T/N$) is:

$$\begin{aligned} V_0(x_0) &= \inf_\alpha \mathbb{E}\Big[\sum_{n=0}^{N-1} |\alpha_n|^2 h \; + \; g(X_N^\alpha) \Big] \\ X_{n+1}^\alpha &= X_n^\alpha + 2\alpha_n h + \sqrt{2}\, \Delta W_{(n+1)h}, \quad X_0^\alpha = x_0. \end{aligned}$$

$\rightarrow$ Explicit solution (via Hopf-Cole transformation):

$$V_0(x_0) = -\ln\Big(\mathbb{E}\Big[\exp\big(-g(x_0 + \sqrt{2} W_T) \big) \Big] \Big).$$

Implementation

Algo Hybrid-Now

- $N = 20$ time steps, $T = 1$, $h = 1/30$.

- DNN for policy (resp. value function): function from $\mathcal{X} = \mathbb{R}^d$ into $\mathbb{A} = \mathbb{R}^d$ (resp. $\mathbb{R}$):
  - Input layer with d neurons
  - 3 hidden layers with $d + 10$ neurons each
  - Output layer with d neurons (resp. 1 neuron)

- Exponential Linear Unit (ELU) activation function

- Optimization with Adam method in TensorFlow
  - Training distribution $\mu \rightsquigarrow \mathcal{N}\big(x_0, \sqrt{2h} I_d\big)$.

Test 1: from Han, E, Jentzen (17)

- $d = 100$, $g(x) = \ln\left(\frac{1}{2}(1 + |x|^2)\right)$.

Figure: Relative error of the Algo Hybrid-Now for $V_0(x_0 = 0)$.
RelError $= 0.092\%$; Standard deviation of $\hat{V}_0^M(0) = 0.00191\%$

Model setup

Real-options valuation of gas storage (discrete-time version of the Carmona-Ludkovski model)

- Gas (random) price $(P_n)_n$
- Gas inventory $(C_n)_n$ controlled by the decision α_n to inject, do nothing, or withdraw gas:

$$C_{n+1} = \begin{cases} C_n + b_{in} & \text{if } \alpha_n = +1 \quad \text{(injection/buy gas)} \\ C_n & \text{if } \alpha_n = 0 \quad \text{(do nothing)} \\ C_n - s_{out} & \text{if } \alpha_n = -1 \quad \text{(withdraw/sell gas)} \end{cases}$$

with b_{in}, $s_{out} > 0$.

- Physical inventory constraint:

$$C_n \in [C_{min}, C_{max}].$$

Control problem

- Maximize over α on finite horizon N:

$$\mathbb{E}\Big[\sum_{k=0}^{N-1} f(P_n, C_n, \alpha_n) \; + \; g(P_N, C_N)\Big]$$

with

- Revenue at any time n:

$$f(p, c, a) \quad = \quad \begin{cases} -b_{in}p - \kappa c & \text{if } a \; = \; +1 \quad \text{(injection/buy gas)} \\ -\kappa c & \text{if } a \; = \; 0 \quad \text{(do nothing)} \\ s_{out}p - \kappa c & \text{if } a \; = \; -1 \quad \text{(withdraw/sell gas)} \end{cases}$$

  with storage cost $\kappa > 0$.

- Terminal condition: penalization for having less gas than initially

$$g(p, c) \quad = \quad -\mu p(C_0 - c)_+.$$

  with $\mu > 0$.

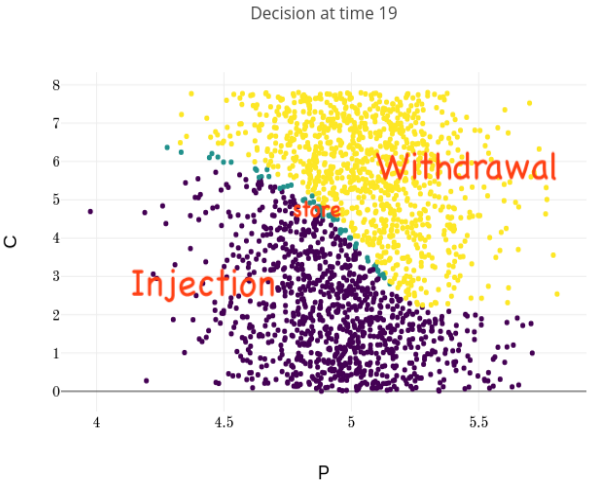
Numerical results

- **Model parameters**:
  - Mean-reverting gas price around $\bar{p} = 5$, with rate $\beta = 0.5$

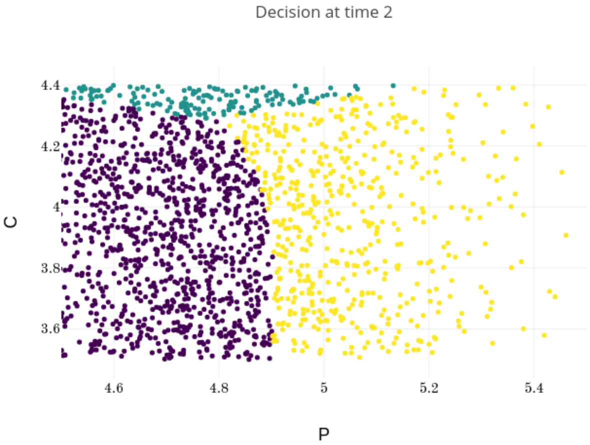
$$P_{n+1} = \bar{p}(1-\beta) + \beta P_n + \xi_{n+1}, \quad \xi_n \rightsquigarrow \mathcal{N}(0, \sigma^2 = 0.05), \quad P_0 = 4,$$

  - $b_{in} = 0.06$, $s_{out} = 0.25$, $\kappa = 0.01$,
  - $N = 30$, $\mu = 2$, $C_0 = 4$, $C_{min} = 0$, $C_{max} = 8$.
- Implementation by Algo NNContPI with DNN classification:
  - 3 hidden layers with $15 + 15 + 5$ neurons, output layer with 3 neurons
  - ELU activation function
  - Training samples of size $M = 250000$

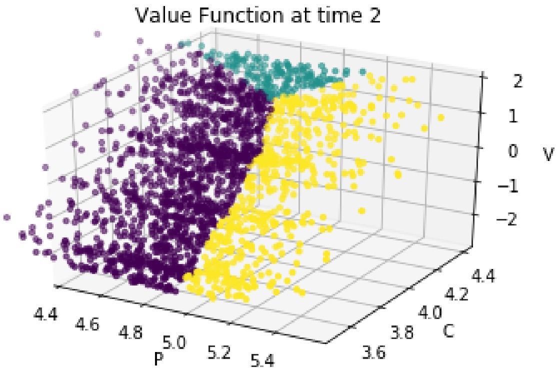
Optimal policy regions



Optimal policy regions varying in time



Value function varying in time



Model description

Discrete-time version of model in Heynmann et al. (17)

- Microgrid:
  - Generator (G) $\rightarrow$ power $G_n = \pi_g$ when turn on
  - Photovoltaic (PV) $\rightarrow$ electricity production $(P_n)_n$ in $[P_{min}, P_{max}]$
  - Battery storage with capacity (C_n) in $[C_{min}, C_{max}]$
- Power demand (D_n)
- Manager decisions: $\alpha = (\alpha^g, \alpha^S)$
  - $\alpha_n^g = 1$ (turn on), $\alpha_n^g = 0$ (turn off)
  - $\alpha_n^S \in [0, P_n]$: amount of (PV) energy transferred to satisfy demand
  - Excess (resp. lack) of energy from (G) and (PV)/Demand for charging (resp. discharging) battery

Control problem

Minimize over $\alpha = (\alpha_n^g, \alpha_n^S)_n$ valued in $\{0,1\} \times [0, P_n]$

$$\mathbb{E}\Big[\sum_{n=0}^{N-1} |G_n|^2\Big] \quad \text{subject to the physical constraints on } \ (C_n),$$

(We ignore here switching cost for turning on/off the generator).

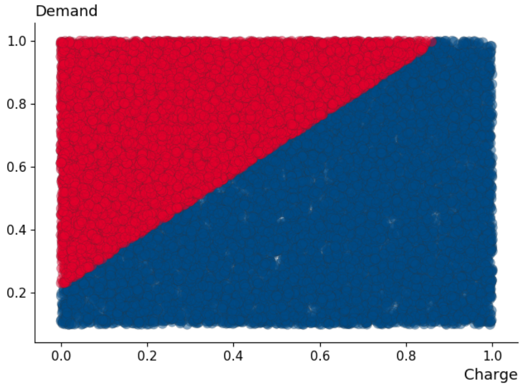
Remark. The constraint on (C_n) are dealt with by penalization in the objective functional.

Numerical results

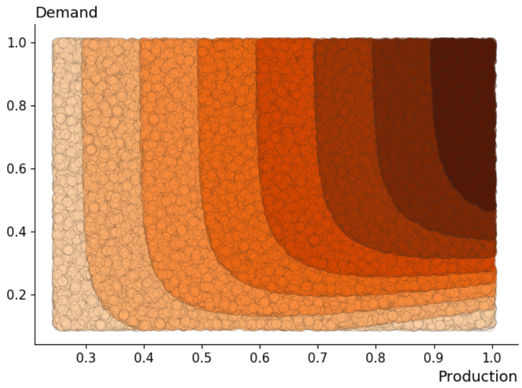
- **Model parameters**
  - Additive model for (P_n^S) valued in $[0.25, 1]$, $P_0 = 0.5$
  - Mean-reverting process for (D_n) around $\bar{D} = 0.5$, and valued in $[0.1, 1]$, $D_0 = 0.4$
  - $\pi_g = 0.8$, $C_{min} = 0$, $C_{max} = 1$
  - $N = 5$, penalty parameter for the constraints $= 10000$

- Implementation by Algo NNContPI with DNN classification for generator policy:
  - 3 hidden layers with $80 + 50 + 30$ neurons, output layer with $2 + 1$ neurons
  - ELU activation function
  - Training samples of size $M = 2^{16}$

Turn on (red)/Turn off (blue) policy regions when increasing PV production



Optimal transfer from PV when increasing battery charge



Concluding remarks

- Machine learning meets stochastic control
  - Neural network regression
  - Control learning
- We analyzed and tested three algorithms

Algo	Bias estimate	Variance	Complexity	Dimension
NNContPI	+	-	-	+
Hybrid-Now	-	+	+	+
Hybrid-LaterQ	-	++	+	-

- Future work:
  - Extension to mean-field control problems ...