Deep learning for mean-field control with common noise and jumps

Nacira Agram
joint with Jan Rems

18 januari 2024

Table of Contents

1. Conditional McKean-Vlasov
   - Pontryagin maximum principle
   - Dynamic programming
2. Interbank Systemic Risk Model - Revisited
3. An optimal consumption/harvesting problem
4. Signatures and Deep learning

Control of conditional McKean-Vlasov

A problem of conditional mean-field control or control of conditional McKean-Vlasov equation consists in:

Dynamics (drift α, diffusion β^0, β, jump coef. γ^0, γ, Brownian motion B,common noise B^0, compensated Poisson r.m. $\tilde{N}$, common jump $\tilde{N}^0$)

Let X^u be the solution of the controlled conditional McKean-Vlasov dynamics

$$dX(t) = dX^u(t) = \alpha(t, X(t), \mu_t, u(t))dt + \beta^0(t, X(t), \mu_t, u(t))dB^0(t)$$
$$+ \beta(t, X(t), \mu_t, u(t))dB(t) + \int_{\mathbb{R}^*} \gamma^0(t, X(t^-), \mu_{t^-}, u(t), \zeta)\widetilde{N}^0(dt, d\zeta)$$
$$+ \int_{\mathbb{R}^*} \gamma(t, X(t^-), \mu_{t^-}, u(t), \zeta)\widetilde{N}(dt, d\zeta), \quad X(0) \sim \mu_0,$$
$$\mu_t^u = \mu_t = \mathcal{L}(X(t) \mid B^0)$$

Cost function (profit rate f, bequest function g, control u, time horizon T)

$$J(u) = \mathbb{E}\Big[\int_0^T f(t, X(t), \mu_t, u(t))dt + g(X(T), \mu_T)\Big]$$

LQ MFG with Common Noise

Example (Interbank borrowing/lending)

X =log-monetary reserve, $u(t)$=rate of borrowing/lending to central bank, population state

$$dX(t) = [a(\mathbb{E}[X(t)|\mathcal{G}_t] - X(t)) + u(t)]dt + \sigma\rho dB^0(t) + \sigma\sqrt{1-\rho^2}dB(t) \\ + \int_{\mathbb{R}} \zeta \widetilde{N^0}(dt, d\zeta) + \int_{\mathbb{R}} \zeta \widetilde{N^1}(dt, d\zeta)$$

$$J(u(t)) = \mathbb{E}\Big[\int_0^T \Big(\frac{1}{2}u(t)^2 - qu(t)\,(\mathbb{E}[X(t)|\mathcal{G}_t] - X(t)) \\ + \frac{\epsilon}{2}(\mathbb{E}[X(t)|\mathcal{G}_t] - X(t))^2\Big)dt + \frac{c}{2}(\mathbb{E}[X(t)|\mathcal{G}_T] - X(T))^2\Big]$$

Continuous case [Carmona et al.,2015]

Pontryagin maximum principle

Metric space of measures

- Wasserstein space: $\mathcal{P}_2(\mathbb{R}^d) := \{\mu \in \mathcal{P}(\mathbb{R}^d) : \int_{\mathbb{R}^d} |x|^2 \mu(dx) < +\infty\}$
- 2-Wasserstein metric:

$$W_2(\mu_1, \mu_2) = \inf\{(\int_{\mathbb{R}^d} |x-y|^2 \mu(dx, dy))^{\frac{1}{2}} : \mu \in \mathcal{P}_2(\mathbb{R}^d \times \mathbb{R}^d) \text{ with } \mu(\cdot \times \mathbb{R}^d) := \mu_1,\ \mu(\mathbb{R}^d \times \cdot) := \mu_2\}$$

Hamiltonian

$$
\begin{aligned}
&H(t,x,\mu,u,p,q,q^0,r(\cdot),r^0(\cdot)) \\
&= f(t,x,\mu,u) + \alpha(t,x,\mu,u)p + \beta^0(t,x,\mu,u)q^0 + \beta(t,x,\mu,u)q \\
&+ \int_{\mathbb{R}} \gamma^0(t,x,\mu,u,\zeta)r^0(\zeta)\nu(d\zeta) + \int_{\mathbb{R}} \gamma(t,x,\mu,u,\zeta)r(\zeta)\nu(d\zeta)
\end{aligned}
$$

BSDE

$$
\begin{aligned}
dp(t) &= -[\partial_x H(t) + \tilde{E}[\partial_\mu H(t)\tilde{X}(t)]dt + q^0(t)dB^0(t) + q(t)dB(t) \\
&\quad + \int_{\mathbb{R}} r^0(t,\zeta)\tilde{N}^0(dt,d\zeta) + \int_{\mathbb{R}} r(t,\zeta)\tilde{N}(dt,d\zeta), \\
p(T) &= \partial_x g(X(T),\mu(T)) + \tilde{E}[\partial_\mu g(\tilde{X}(T),\mu(T))\tilde{X}(T)]
\end{aligned}
$$

Theorem (Sufficient maximum principle)

Let $\hat{u}$ be an admissible control with corresponding controlled state and adjoint processes. Suppose that for each $t \in [0, T]$

1. *(Convexity) The functions*

$$\begin{aligned} (x, \mu, u) &\mapsto H(t) \\ (x, \mu) &\mapsto g(x, \mu), \end{aligned}$$

   are convex $dt \otimes \mathbb{P}$ a.e.

2. *(Minimum conditions)*

$$\mathbb{E}[\hat{H}(t)] = \underset{u \in \mathcal{A}}{ess \inf} \mathbb{E}[H(t)],$$

$dt \otimes \mathbb{P}$ a.e. Then $\hat{u}$ is an optimal control for our problem.

Adjoint equation

$$
\begin{aligned}
dY(t) = -[&aY(t) + b\mathbb{E}[Y(t)|B^0] + cZ^0(t) + d\mathbb{E}[Z^0(t)|B^0] + mZ(t) \\
&+ n\mathbb{E}[Z(t)|B^0] + \gamma(t)]dt + Z^0(t)dB^0(t) + Z(t)dB(t) \\
Y(T) = \xi&
\end{aligned}
$$

Question

What is the closed formula for linear BSDE with common noise?

Non-common noise [Agram et al. 2022]

Weighted Sobolev space $\mathbb{M}$ is the pre-Hilbert space of random measures μ on $\mathbb{R}^n$ equipped with the norm

$$\|\mu\|_{\mathbb{M}}^2 := \mathbb{E}[\int_{\mathbb{R}^n} |\hat{\mu}(y)|^2 e^{-y^2} dy],$$

$$\hat{\mu}(y) := \int_{\mathbb{R}^n} e^{-ixy} \mu(dx); \quad y \in \mathbb{R}^n,$$

where $xy = x \cdot y = x_1 y_1 + x_2 y_2 + ... + x_n y_n$ is the scalar product in $\mathbb{R}^n$.
If $\mu, \eta \in \mathbb{M}$:

$$\langle \mu, \eta \rangle_{\mathbb{M}} = \mathbb{E}[\int_{\int_{\mathbb{R}^n}} \mathrm{Re}(\bar{\hat{\mu}}(y)\hat{\eta}(y))|y|^2 e^{-y^2} dy],$$

where, $\mathrm{Re}(z)$ denotes the real part and $\bar{z}$ denotes the complex conjugate of the complex number z.

Fokker Planck SPIDE

Fokker Planck equation:

$$d\mu_t = A_0^*\mu_t dt + A_1^*\mu_t dB^0(t) + \int_{\mathbb{R}^k} A_2^*\mu_t \widetilde{N}^0(dt, d\zeta)$$

where

$$A_0^*\mu = -D[\alpha\mu] + \frac{1}{2}D[((\beta^0)^2 + \beta^2)\mu]$$
$$+ \sum_{\ell=1}^{2} \int_{\mathbb{R}} \left\{ \mu^{(\gamma^{(\ell)})} - \mu + D[\gamma^{(\ell)}(s,\cdot,\zeta)\mu] \right\} \nu_\ell(d\zeta)$$

and

$$A_1^*\mu = -D[\beta_0\mu], \quad A_2^*\mu = \mu^{(\gamma^{(0)})} - \mu$$

Dynamic programming

$$dY(t) = F(Y(t))dt + G(Y(t))dB(t) + \int_{\mathbb{R}^k} H(Y(t^-),\zeta)\widetilde{N}(dt,d\zeta)$$

$$:= \begin{bmatrix} dt \\ dX(t) \\ d\mu_t \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha(Y(t)) \\ A_0^*\mu_t \end{bmatrix} dt + \begin{bmatrix} 0_{1\times m} \\ \beta(Y(t)) \\ A_1^*\mu_t, 0, 0..., 0 \end{bmatrix} dB(t)$$

$$+ \int_{\mathbb{R}^k} \begin{bmatrix} 0_{1\times k} \\ \gamma(Y(t^-),z) \\ A_2^*\mu_t, 0, 0, ..., 0 \end{bmatrix} \widetilde{N}(dt,dz)$$

$$J(y) = \mathbb{E}^y\Big[\int_0^T f(s+t, X(t), \mu_t, u(t))dt + g(X(T),\mu_T)\Big]$$

More details [Agram et al. 2024]

Hamilton-Jacobi-Bellman equation

$$f(y,\widehat{u}(y)) + L_{\widehat{u}(y)}\widehat{\varphi}(y) = 0, \quad \widehat{\varphi}(T,x,\mu) = g(x,\mu)$$

Here

$$L\varphi = \frac{\partial\varphi}{\partial s} + \sum_{j=1}^{d}\alpha_j\frac{\partial\varphi}{\partial x_j} + \langle\nabla_\mu\varphi, A_0^*\mu\rangle + \tfrac{1}{2}\sum_{j,n=1}^{d}(\beta\beta^T)_{j,n}\frac{\partial^2\varphi}{\partial x_j\partial x_n}$$

$$+ \tfrac{1}{2}\sum_{j=1}^{d}\beta_{j,1}\frac{\partial}{\partial x_j}\langle\nabla_\mu\varphi, A_1^*\mu\rangle + \tfrac{1}{2}\langle A_1^*\mu, \langle D_\mu^2\varphi, A_1^*\mu\rangle\rangle$$

$$+ \int_{\mathbb{R}^k}\Big\{\varphi(s,x+\gamma^{(1)},\mu+A_2^*\mu) - \varphi(s,x,\mu) - \sum_{j=1}^{d}\gamma_j^{(1)}\tfrac{\partial}{\partial x_j}\varphi(s,x,\mu) - \langle A_2^*\mu, D_\mu\varphi\rangle\Big\}\nu$$

$$+ \sum_{\ell=2}^{k}\int_{\mathbb{R}^k}\Big\{\varphi(s,x+\gamma^{(\ell)},\mu)) - \varphi(s,x,\mu) - \sum_{j=1}^{d}\gamma_j^{(\ell)}\tfrac{\partial}{\partial x_j}\varphi(s,x,\mu)\Big\}\nu_\ell(dz)$$

Table of Contents

1. Conditional McKean-Vlasov
   - Pontryagin maximum principle
   - Dynamic programming
2. Interbank Systemic Risk Model - Revisited
3. An optimal consumption/harvesting problem
4. Signatures and Deep learning

Interbank Systemic Risk Model - borrowing/lending

Example (Interbank borrowing/lending)

X =log-monetary reserve, $u(t)$=rate of borrowing/lending to central bank, population state

$$dX(t) = [a(\mathbb{E}[X(t)|B^0] - X(t)) + u(t)]dt + \sigma\rho dB^0(t) + \sigma\sqrt{1-\rho^2}dB(t) + \gamma\int_{\mathbb{R}} \zeta \widetilde{N}(dt, d\zeta)$$

The goal is to minimize

$$J(u) = \mathbb{E}\Big[\int_0^T \Big(\Big(\frac{1}{2}u(t)^2 - qu(t)\Big)\big(\mathbb{E}[X(t)|B^0] - X(t)\big) + \frac{\epsilon}{2}(\mathbb{E}[X(t)|B^0] - X(t))^2\Big)dt + \frac{c}{2}(\mathbb{E}[X(t)|B^0] - X(T))^2\Big]$$

Table of Contents

1 Conditional McKean-Vlasov
    Pontryagin maximum principle
    Dynamic programming

2 Interbank Systemic Risk Model - Revisited

3 An optimal consumption/harvesting problem

4 Signatures and Deep learning

An optimal consumption/harvesting problem

$$dX(t) = \mathbb{E}[X(t)|B^0]\Big[(\rho(t) - c(t))\,dt + \theta dB^0(t) + \sigma_0(t)\,dB(t) + \int_{\mathbb{R}} \gamma_0(t,\zeta)\,\widetilde{N}(dt,d\zeta)\Big]$$

$$J(c) = \mathbb{E}\left[\int_0^T \ln\big(c(t)\mathbb{E}[X(t)|B^0]\big)\,dt + \lambda \ln\Big(\mathbb{E}\left[X(T)|B^0\right]\Big)\right]$$

- **The value function has a special form (ansatz):**

$$\varphi(s, x, \mu) = \kappa_0(s) + \kappa_1(s) \ln \mu$$

$\kappa_0(s), \kappa_1(s)$ are C^1 deterministic functions

- **Hamilton-Jacobi-Bellman equation:**

$$\mu + \kappa_0'(s) + \kappa_1'(s) \ln \mu + \kappa_1(s)(\rho(t) - c) - \tfrac{1}{2}\kappa_1(s)\theta^4 \Big\} = 0$$

Consequence: the MFC solution is given by:

- **Value function:**

$$\varphi(s, x, \mu) = \kappa_0(s) + \kappa_1(s) \ln \mu$$

- **Control:**

$$\widehat{c}(s) = \frac{1}{\lambda + T - s}$$

with

$$\begin{cases} \kappa_0'(s) = & \frac{1}{2}\kappa_1(s)\theta^4 + \ln \kappa_1(s) - (\rho(s) + \theta)\kappa_1(s) \\ \kappa_0(T) = & 0, \end{cases}$$

More details [Agram et al. 2022]

Table of Contents

1. Conditional McKean-Vlasov
   - Pontryagin maximum principle
   - Dynamic programming
2. Interbank Systemic Risk Model - Revisited
3. An optimal consumption/harvesting problem
4. Signatures and Deep learning

Signatures of Paths

Consider a tensor algebra $T(\mathbb{R}^m) = \bigoplus_{k=0}^{\infty} (\mathbb{R}^m)^{\otimes k}$.

Definition

Let $(X(t))_{t\in[0,T]}$ be a stochastic process with values in $\mathbb{R}^m$ and finite p-variation. The signature $S(X)_{a,b}$ of X on an interval $[a, b] \subseteq [0, T]$ is an element of $T(\mathbb{R}^m)$ defined by $S(X)_{a,b} = (1, X^1, \dots, X^k, \dots)$, where

$$X^k = \int_{a<t_1<\cdots<t_k<b} dX(t_1) \otimes \cdots \otimes dX(t_k).$$

We denote by $S^D_{a,b}(X) = (1, X^1, \dots, X^D)$ the truncated signature of depth D. It has dimension $\frac{m^{D+1}-1}{m-1}$.

Signature characterizes the path up to tree-like equivalence and is useful as a feature set when working with paths

Deep learning algorithm - continuous case

In the benchmark example, we set $\gamma = 0$. Algorithm is composed of three main components

1. For SDE approximation, we use the Euler-Maruyama method.
2. The conditional expectation is estimated using signatures and Ridge regression.
3. Control is learned with LSTM networks and stochastic gradient descent

Deep learning algorithm - continuous case

Algorithm Optimal control for common noise

Require: Learning rate η, signature depth D, LSTM networks $\{\varphi_n\}_{0\leq k\leq N-1}$ and Brownian motions $\{B_k^{0,j}\}_{0\leq k\leq N}^{1\leq j\leq M}$, $\{B_k^j\}_{0\leq k\leq N}^{1\leq j\leq M}$, $\{X_0^j\}^{1\leq j\leq M}$.

for $1 \leq \text{epoch} \leq P$ **do**

  for $0 \leq k \leq N-1$ **do**

    Compute optimal f^* by ridge regression for pairs $S_{0,t_k}^D(t, B^0)$ and X_k

    Set $\mu_k^j = f^*\left(S^D_{0,t_k(t,B^{0,j})}\right)$

    Set $u_k^j = \varphi_k(X_k^j, \mu_k^j; \theta)$

    Set $X_{k+1}^j = X_k^j + [a(\mu_k^j - X_k^j) + u_k^j]\Delta_t + \sigma\rho\Delta B_k^{0,j} + \sigma\sqrt{1-\rho^2}\Delta B_k^j$

  end for

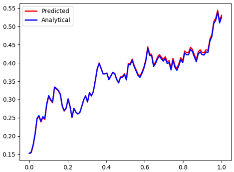
  Using Monte Carlo obtain $\hat{J} = \frac{1}{M}\sum_{j=1}^M J(u^j, X^j, \mu^j)$

  Update $\theta = \theta - \eta\nabla\hat{J}$

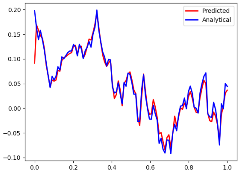
end for

return u, X, μ

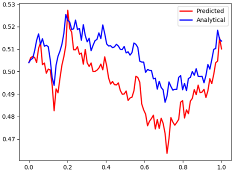
Deep learning algorithm - continuous case



(a) State process X



(b) Control process u

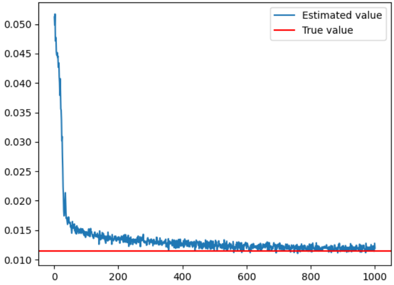


(c) Cond. expectation μ

Figur: Comparison between predicted and analytical solutions

Deep learning algorithm - continuous case

The loss function also converges nicely towards the theoretical value of 0.011, as seen in the graph below.



Figur: Convergence of loss function towards its theoretical value

Deep learning algorithm - with jump

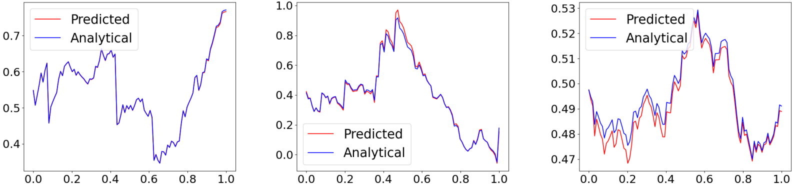
Here we assume jumps do influence dynamics.The Euler-Maruyama discretization changes into

$$X_{k+1}^j = X_k^j + [a(\mu_k^j - X_k^j) + u_k^j]\Delta_t + \sigma\rho\Delta B_k^{0,j} + \sigma\sqrt{1-\rho^2}\Delta B_k^j \\ + \gamma J_k^j - \lambda\gamma\nu\Delta_t, \tag{1}$$

where $J_k^j \sim \mathcal{N}(\nu, \beta^2)$ if Poisson process with rate λ has jump on the interval $[t_k, t_k + 1]$.

We use a different deep learning approach that fixes the estimation of conditional expectation and only updates it on every few rounds of training.

Deep learning algorithm - with jump

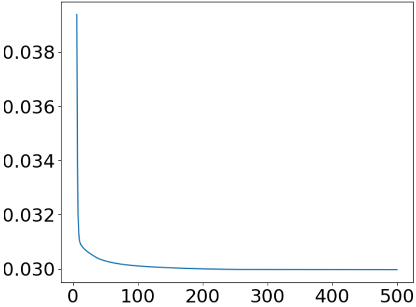


(a) State process X (b) Control process u (c) Cond. expectation μ

Figur: Comparison between predicted and analytical solutions

Deep learning algorithm - with jump

The loss function still converges nicely but towards a bigger value due to jumps



Figur: Convergence of the loss function

Thank you for your attention