

# Notas

---

Carlos Daniel Contreras Quiroz

10 de abril de 2023

## To do list

1. Entender BSDE ✓
  2. Escribir conexión PDE-BSDE ✓
  3. Arreglar dimensiones en cap 1, para  $Y$  general
  4. Programar reflexión en frontera ✓
  5. Deep Fictitious Play
  6. Escribir Crowd motion
  7. Simular diferencias finitas
  8. Escribir simulaciones
  9. Escribir apendice Neural Networks
- 

## Problemas

1. El repositorio del review corre muy lento ✓
  2. Las condiciones de frontera no son iguales
    - Reflejar los caminos puede funcionar ✗  
Esto no funciona ni tiene sentido, qué pendejo, en 1d funciona porque  $d=n$ , pero en lo que estaba haciendo no.
    - Combinarlo con PINNS!
  3. Parece que DeepBSDE no funciona con dimensiones de más de 100
    - Podría usarse el deep Backward de Pham [1], se puede entrenar basado en modelos anteriores. No hay código
    - O tambien deep splitting de Beck [2]. Si hay código
- 

## Ideas

- Hacerlo en flux/equinox/jax
- Usar hopf formula?
- Buscar un modelo de Cucker Smale abierto
- Buscar un método para resolver problemas en horizonte infinito (actor-critic podría funcionar)(Tambien elliptic Feynman-Kac)
- Resolver MFG con deep learning
- Price of anarchy

---

## Preguntas

1. En que sentido converge a la solución?
  2. Qué pasaría si se cambia la discretización de Euler por una mejor? Afecta el modo de convergencia?
- 

## Probabilidad

### Preguntas

- Es  $E[\int_0^T |f|^2] < \infty$  equivalente a que  $\int_0^T |f|^2 < \infty$  a.s?
- Por que Feynman-Kac se parece a un problema de control?

### Para recordar

#### Recordar: Filtración aumentada

A una filtración  $\mathbb{F} = (\mathcal{F}_t)_t$  en un espacio de probabilidad  $(\Omega, \mathcal{A}, \mathbb{P})$  le corresponde una filtración continua a la derecha  $\mathbb{F}^+ = (\mathcal{F}_t)_t = \cap_{t < s} (\mathcal{F}_s)$ .  $\mathbb{F}$  se dice continua a la derecha si  $\mathbb{F}^+ = \mathbb{F}$ .

Sea  $\mathcal{N}_P = \{A \subseteq \omega | A \subseteq B \text{ para algún } B \in \mathbb{F} \text{ con } P(B) = 0\}$ .  $\mathbb{F}$  se dice completa si  $\mathcal{F}_t$  contiene  $\mathcal{N}_P$  para todo  $t$ .

Una filtración continua a la derecha y completa se llama filtración aumentada.

## Deep Learning schemes

### Preguntas

- Porqué se puede resolver hacia adelante la bsde? Qué significa eso?
- No hay forma de meter el parámetro  $t$  como entrada en la red neuronal?
- Como aseguramos que el proceso se quede adentro de la habitación?

### To do list

1. Programar Deep BSDE Flux
2. Programar DBDP y Deep Splitting
3. Programar Merged Deep BSDE
4. Programar PINNs interpolation
5. Comprobar con problemas de frontera
6. Programar Reflexion 1d
7. Calcular controles
8. Escribir

Vamos a intentar resolver ecuaciones del tipo

$$\begin{aligned} \frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \text{Tr}(\sigma \sigma^T(t, x) (\text{Hess}_x u)(t, x)) + \nabla u(t, x) \cdot \mu(t, x) \\ + f(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x)) = 0 \end{aligned} \quad (1)$$

con la condición final  $u(T, x) = g(x)$ .

Vamos a realizar una aproximación con la fórmula de Feynman-Kac. Esto es, la solución de la ecuación anterior viene dada por

$$\begin{aligned} u(t, X_t) - u(0, X_0) \\ = - \int_0^t f(s, X_s, u(s, X_s), \sigma^T(s, X_s) \nabla u(s, X_s)) ds \\ + \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s) dW_s. \end{aligned} \quad (2)$$

donde  $X$  resuelve la ecuación diferencial estocástica

$$X_t = \xi + \int_0^t \mu(s, X_s) ds + \int_0^t \sigma(s, X_s) dW_s \quad (3)$$

Vamos a estudiar primero el caso de HJB del repositorio. Acá intentamos controlar el proceso 100-dimensional

$$dX_t = 2\sqrt{\lambda} m_t dt + \sqrt{2} dt \quad X(0) = x \quad t \in (0, T) \quad (4)$$

a través del control  $m_t$ , con el funcional de costo

$$J(m_t) = \mathbb{E} \left[ \int_0^T \|m_t\|^2 dt + g(X_T) \right]. \quad (5)$$

Recordar: Hamilton-Jacobi-Bellman

Para un proceso controlado

$$\begin{aligned} dX_x &= \mu(X, u)dt + \sigma(X, u)dW \\ X(0) &= x \end{aligned}$$

con función de costo

$$J(t, x, u) = \mathbb{E} \left[ \int_t^T f(X_x, u)dt + g(X_x) \right].$$

la ecuación de HJB para la función de valor

$$V(t, x) = \sup_{u \in \mathcal{A}} J(t, x, u)$$

es

$$\frac{\partial V}{\partial t} + \sup_{u \in \mathcal{A}} \{ \mathcal{L}^u(V) + f(x, u) \} = 0$$

donde

$$\mathcal{L}^u(V)(t, x) = \mu(x, u) \cdot \nabla V(t, x) + \frac{1}{2} \text{Tr}(\sigma(t, x) \sigma^T(t, x) \nabla^2 V(t, x))$$

con la condición final

$$u(T, x) = g(x)$$

Esto también puede escribirse con el Hamiltoniano

$$H(t, x, p, M) = \sup_{u \in \mathcal{A}} \{ \mu(x, u) \cdot p + \frac{1}{2} \text{Tr}[\sigma(t, x) \sigma^T(t, x) M] + f(x, u) \}$$

y asumiendo que el control no se aplica a la volatilidad se escribe

$$\frac{\partial u}{\partial t} + H(t, x, \nabla u) + \frac{1}{2} \text{Tr}[\sigma(t, x) \sigma^T(t, x) \nabla^2 u] = 0$$

```
1  import numpy as np
2
3  def incmatrix(genl1, genl2):
4      m = len(genl1)
5      n = len(genl2)
6      M = None #to become the incidence matrix
7      VT = np.zeros((n*m, 1), int) #dummy variable
8
9      #compute the bitwise xor matrix
10     M1 = bitxormatrix(genl1)
11     M2 = np.triu(bitxormatrix(genl2), 1)
12
13     for i in range(m-1):
14         for j in range(i+1, m):
15             [r, c] = np.where(M2 == M1[i, j])
16             for k in range(len(r)):
17                 VT[(i)*n + r[k]] = 1;
18                 VT[(i)*n + c[k]] = 1;
19                 VT[(j)*n + r[k]] = 1;
20                 VT[(j)*n + c[k]] = 1;
21
22     if M is None:
23         M = np.copy(VT)
24     else:
25         M = np.concatenate((M, VT), 1)
26
27     VT = np.zeros((n*m, 1), int)
28
29     return M
```

## N-agent games and Mean field modeling

### Preguntas

- Cual es la diferencia entre los tipos de control abierto, cerrado y markoviano? En qué afecta acá?
- cuales son exactamente las dimensiones?

### To do list

1. Programar deep fictious play
2. Programar diferencias finitas
3. Escribir contexto
4. Escribir Mean field
5. Escribir deep fictious

## Referencias

- [1] Côme Huré, Huyên Pham y Xavier Warin. “Deep backward schemes for high-dimensional nonlinear PDEs”. En: *Mathematics of Computation* 89.324 (31 de ene. de 2020), págs. 1547-1579. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/mcom/3514. URL: <https://www.ams.org/mcom/2020-89-324/S0025-5718-2020-03514-5/> (visitado 28-02-2023).

- [2] Christian Beck et al. “Deep Splitting Method for Parabolic PDEs”. En: *SIAM Journal on Scientific Computing* 43.5 (ene. de 2021), A3135-A3154. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/19M1297919. URL: <https://epubs.siam.org/doi/10.1137/19M1297919> (visitado 14-02-2023).