# On PDE approach to some machine learning problems

Hong Zhang

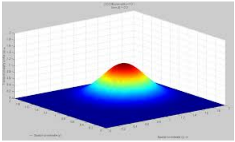
Applied Math, Brown University

April 8, 2017

Partial differential equations (PDEs) are used to describe many phenomenons. Many famous PDEs are from physics.

- Heat: heat equation, $u$ tempature

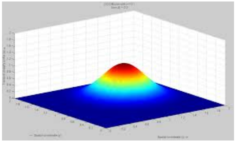
$$\partial_t u = \sum_{i=1}^{3} \frac{\partial^2 u}{\partial x_i^2} = \Delta u$$



Partial differential equations (PDEs) are used to describe many phenomenons. Many famous PDEs are from physics.

- Heat: heat equation, $u$ tempature

$$\partial_t u = \sum_{i=1}^{3} \frac{\partial^2 u}{\partial x_i^2} = \Delta u$$



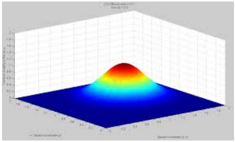
- Fluid mechanics: Naiver-Stokes equation

$$\partial_t u + (u \cdot \nabla) u - \Delta u = -\nabla p.$$

Partial differential equations (PDEs) are used to describe many phenomenons. Many famous PDEs are from physics.

- Heat: heat equation, $u$ tempature

$$\partial_t u = \sum_{i=1}^{3} \frac{\partial^2 u}{\partial x_i^2} = \Delta u$$



- Fluid mechanics: Naiver-Stokes equation

$$\partial_t u + (u \cdot \nabla) u - \Delta u = -\nabla p.$$

- Quantum mechanics, electrodynamics, image processing,. . . .

# Probability and PDEs

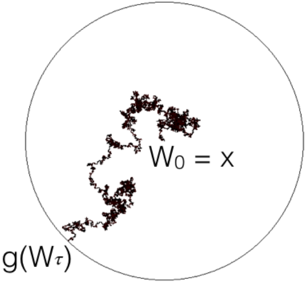



Figure: Brownian Motion

$W_t$: a Brownian motion with $W_0 = x$. $W$ hits boundary at $W_\tau$. $u(x) = \mathbb{E}^x[g(W_\tau)]$. Then

$$\Delta u = 0,$$

and

$$u = g \quad \text{on} \quad S_1.$$

# Online Learning: Example 1


Figure: Spam Detection

- emails arrive one by one

# Online Learning: Example 1


Figure: Spam Detection

- emails arrive one by one
- spam detector makes prediction for each of them: spam or not

# Online Learning: Example 1


Figure: Spam Detection

- emails arrive one by one
- spam detector makes prediction for each of them: spam or not
- sometimes get feedback from users

# Online Learning: Example 1


Figure: Spam Detection

- emails arrive one by one
- spam detector makes prediction for each of them: spam or not
- sometimes get feedback from users
- make better detection next time

# Online Learning: Example 2


Figure: Recommendation

- users visit Netflix /Amazon/...

# Online Learning: Example 2


Figure: Recommendation

- users visit Netflix /Amazon/...
- movies/products/... are recommended

# Online Learning: Example 2


Figure: Recommendation

- users visit Netflix /Amazon/. . .
- movies/products/. . . are recommended
- users click or not

# Online Learning: Example 2


Figure: Recommendation

- users visit Netflix /Amazon/...
- movies/products/... are recommended
- users click or not
- make better recommendation next time

# Why Online Learning?

- Batch learning:
  - distribution fixed over time (training and test)
  - i.i.d. assumptions
  - a fixed predictor

# Why Online Learning?

- Batch learning:
  - distribution fixed over time (training and test)
  - i.i.d. assumptions
  - a fixed predictor
- Online learning:
  - no distribution assumption
  - suitable for changing or adversarial environment
  - a natural model for many applications, mixing training and test
  - memory efficient

# Prediction With Expert Advices

- Need: to make a decision every day
  - which stock to invest in
  - which route to drive home
  - ...

# Prediction With Expert Advices

- Need: to make a decision every day
  - which stock to invest in
  - which route to drive home
  - ...
- Given: advice from a set of experts
  - financial adviser
  - Route 1, Route I-95, Route 295,...
  - ...

# Prediction With Expert Advices

- Need: to make a decision every day
  - which stock to invest in
  - which route to drive home
  - . . .

- Given: advice from a set of experts
  - financial adviser
  - Route 1, Route I-95, Route 295,. . .
  - . . .

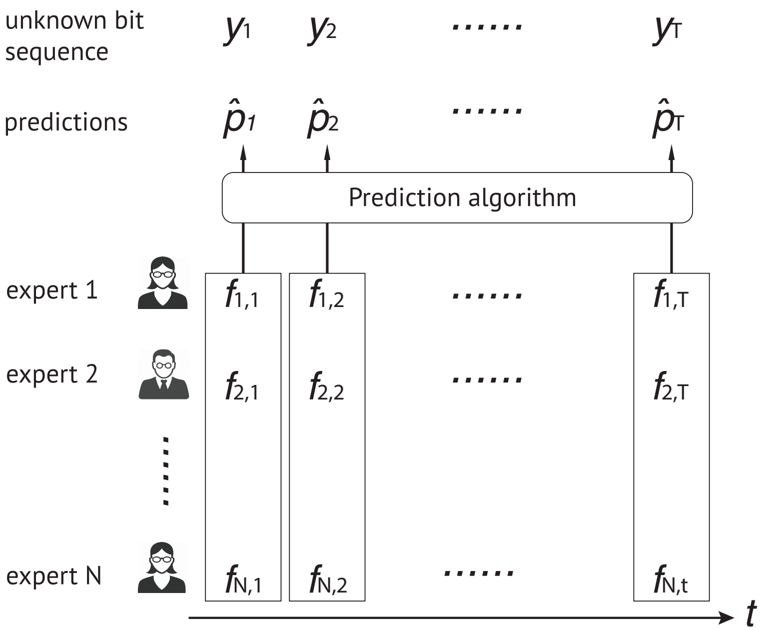
- Question: How to combine advice and make smart choices?

unknown bit sequence $y_1$ $y_2$ ...... $y_T$

$t$

| | | | | | |
|---|---|---|---|---|---|
| unknown bit sequence | | $y_1$ | $y_2$ | ...... | $y_T$ |
| expert 1 |  | $f_{1,1}$ | $f_{1,2}$ | ...... | $f_{1,T}$ |
| expert 2 |  | $f_{2,1}$ | $f_{2,2}$ | ...... | $f_{2,T}$ |
| | ⋮ | | | | |
| expert N |  | $f_{N,1}$ | $f_{N,2}$ | ...... | $f_{N,t}$ |

$\longrightarrow t$


unknown bit sequence
$y_1$
$y_2$
......
$y_T$
predictions
$\hat{p}_1$
$\hat{p}_2$
......
$\hat{p}_T$
Prediction algorithm
expert 1
expert 2
expert N
$f_{1,1}$
$f_{1,2}$
......
$f_{1,T}$
$f_{2,1}$
$f_{2,2}$
......
$f_{2,T}$
$f_{N,1}$
$f_{N,2}$
......
$f_{N,t}$
$t$

# Gentle Start: a simple example

- Set up:
  - Predict an unknown sequence $y_1, y_2, \ldots,$ where $y_i \in \{0, 1\}$.
  - $N$ experts provide advice $(f_{1,t}, f_{2,t}, \ldots, f_{N,t})$
  - A forecaster makes her guess $\hat{p}_t \in \{0, 1\}$ for $y_t$
  - True $y_t$ is revealed and find if $\hat{p}_t = y_t$.
  - Know that one of the experts makes no mistakes.
- Goal: bound the number of the mistakes made by the forescaster

# Halving Algorithm

Algorithm:

- at time $t$, expert $i$ has weight $w_i^t$

# Halving Algorithm

Algorithm:

- at time $t$, expert $i$ has weight $w_i^t$
- originally, $w_i = 1, \forall i \in [1, N]$

# Halving Algorithm

Algorithm:

- at time $t$, expert $i$ has weight $w_i^t$
- originally, $w_i = 1, \forall i \in [1, N]$
- prediction according to the weighted majority

# Halving Algorithm

Algorithm:

- at time $t$, expert $i$ has weight $w_i^t$
- originally, $w_i = 1, \forall i \in [1, N]$
- prediction according to the weighted majority
- update expert's weight with wrong prediction $w_i = 0$

# Halving Algorithm

Algorithm:

- at time $t$, expert $i$ has weight $w_i^t$
- originally, $w_i = 1, \forall i \in [1, N]$
- prediction according to the weighted majority
- update expert's weight with wrong prediction $w_i = 0$

Conclusion: the number of mistakes made by the forecaster is bounded by $\lfloor \log_2 N \rfloor$.

# More general setting:

- Do not know there is one expert who makes no mistakes.

# More general setting:

- Do not know there is one expert who makes no mistakes.
- Weighted Majority Algorithm:
  - at any time, expert $i$ has weight $w_i^t$.
  - originally, $w_i^t = 1, \forall i \in [1, N]$.
  - prediction according to weighted majority.
  - weight of each wrong expert updated ($\beta \in (0, 1)$)

$$w_i^{t+1} \leftarrow w_i^t \beta.$$

# More general setting:

- Do not know there is one expert who makes no mistakes.
- Weighted Majority Algorithm:
  - at any time, expert $i$ has weight $w_i^t$.
  - originally, $w_i^t = 1, \forall i \in [1, N]$.
  - prediction according to weighted majority.
  - weight of each wrong expert updated ($\beta \in (0, 1)$)

$$w_i^{t+1} \leftarrow w_i^t \beta.$$

- Theorem: let $m_i^t$ be the number of mistakes made by expert $i$ till time $t$ and $m^t$ be the number of mistakes made by the WM algorithm. Then

$$m_t \leq \frac{\log_2 N + m_{\text{best}}^t \log_2 \frac{1}{\beta}}{\log_2 \frac{2}{1+\beta}}.$$

  - $m_t \leq O(\log_2 N) + \text{constant} \times \text{best expert}.$

# Remarks:

$$m_t \leq O(\log_2 N) + \text{constant} \times \text{best expert}$$

- Linear dependence on the best expert

# Remarks:

$$m_t \leq O(\log_2 N) + \text{constant} \times \text{best expert}$$

- Linear dependence on the best expert
- Independent of the number of predictions

# Remarks:

$$m_t \leq O(\log_2 N) + \text{constant} \times \text{best expert}$$

- Linear dependence on the best expert
- Independent of the number of predictions
- Independent of the choice of sequence of outcomes

# Remarks:

$$m_t \leq O(\log_2 N) + \text{constant} \times \text{best expert}$$

- Linear dependence on the best expert
- Independent of the number of predictions
- Independent of the choice of sequence of outcomes
- Mild dependence on the number of the experts

## General Setting

- For $t = 1$ to $T$ do
  - feature $x_t \in X$ and advice $\hat{y}_{t,i} \in Y$, $i \in [1, N]$
  - predict $\hat{y}_t \in Y$
  - true label $y_t \in Y$
  - loss $L(y_t, \hat{y}_t)$

# General Setting

- For $t = 1$ to $T$ do
  - feature $x_t \in X$ and advice $\hat{y}_{t,i} \in Y$, $i \in [1, N]$
  - predict $\hat{y}_t \in Y$
  - true label $y_t \in Y$
  - loss $L(y_t, \hat{y}_t)$
- Objective: minimize regret, i.e., difference of total loss incurred and that of best exeprt:

$$Regret(T) = \sum_{t=1}^{T} L(y_t, \hat{y}_t) - \min_{i \in [1,N]} \sum_{t=1}^{T} L(y_t, \hat{y}_{t,i}).$$

# General Setting

- For $t = 1$ to $T$ do
  - feature $x_t \in X$ and advice $\hat{y}_{t,i} \in Y$, $i \in [1, N]$
  - predict $\hat{y}_t \in Y$
  - true label $y_t \in Y$
  - loss $L(y_t, \hat{y}_t)$
- Objective: minimize regret, i.e., difference of total loss incurred and that of best exeprt:

$$Regret(T) = \sum_{t=1}^{T} L(y_t, \hat{y}_t) - \min_{i \in [1,N]} \sum_{t=1}^{T} L(y_t, \hat{y}_{t,i}).$$

- Goal: obtain sublinear regret to the best expert. (top 10 experts, any competitor, etc. )

$$\lim_{T \to \infty} \frac{Regret(T)}{T} = 0.$$

# An Optimal Bound

- Exponential weighted average
  - weight update:
$$w_i^{t+1} \leftarrow w_i^t e^{-\eta_t L_{t,i}},$$
    where $L_{t,i}$ is total loss incurred by expert $i$ till time $t$
  - prediction:
$$\hat{y}_t = \frac{\sum_{i=1}^N w_i^t y_{t,i}}{\sum_{i=1}^N w_i^t}.$$

# An Optimal Bound

- Exponential weighted average
  - weight update:
    $$w_i^{t+1} \leftarrow w_i^t e^{-\eta_t L_{t,i}},$$
    where $L_{t,i}$ is total loss incurred by expert $i$ till time $t$
  - prediction:
    $$\hat{y}_t = \frac{\sum_{i=1}^N w_i^t y_{t,i}}{\sum_{i=1}^N w_i^t}.$$
- Theorem: Assume $L$ is convex in its first argument and takes values in $[0,1]$. Then for any $T$

$$Regret(T) \leq \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{(T/2)\log_2 N} + \sqrt{\log_2 N/2}.$$

# Remarks on the theorem

- When the loss is between $[0, 1]$

$$Regret(T) = O(\sqrt{(\log N)T}).$$

## Remarks on the theorem

- When the loss is between $[0, 1]$

$$Regret(T) = O(\sqrt{(\log N)T}).$$

- The bound can be improved if more information is known, e.g., if the loss functions are special, $L_2$ loss, $L_1$ loss,

## Remarks on the theorem

- When the loss is between $[0, 1]$

$$Regret(T) = O(\sqrt{(\log N)T}).$$

- The bound can be improved if more information is known, e.g., if the loss functions are special, $L_2$ loss, $L_1$ loss,
- In general, the bound is asymptotically optimal.

# Active research area: literatures

- Littlestone and Warmuth developed weighted majority algorithm
- Cesa-Bianchi et al. obtain minimax regret bounded by $\sqrt{(T/2)\log N}$.
- Cesa-Bianchi and Lugosi connected the regret with the Rademacher Complexity of the set of experts.
- Even-Dar et al. considered regret to the average
- Rakhlin et al. developed Sequential Rademacher Complexity to study online learning.
- ...

# A naive prediction problem (T.Cover, 1965)

- Predict $y_1, y_2, \ldots$, where $y_t \in \{-1, 1\}$ (stock goes up or down).

# A naive prediction problem (T.Cover, 1965)

- Predict $y_1, y_2, \ldots$, where $y_t \in \{-1, 1\}$ (stock goes up or down).
- Forecaster makes prediction $\hat{p}_t \in [-1, 1]$ (buy/sell $\hat{p}_1$ shares of stocks) according to some algorithm $\hat{P}$.

# A naive prediction problem (T.Cover, 1965)

- Predict $y_1, y_2, \ldots$, where $y_t \in \{-1, 1\}$ (stock goes up or down).
- Forecaster makes prediction $\hat{p}_t \in [-1, 1]$ (buy/sell $\hat{p}_1$ shares of stocks) according to some algorithm $\hat{P}$.
- At each time $t$, investor loss (or gain) $-\hat{p}_t y_t$. Cumulative loss (or gain)

$$L(\hat{P}, y_T) := -\sum_{t=1}^{T} \hat{p}_t y_t$$

# A naive prediction problem (T.Cover, 1965)

- Predict $y_1, y_2, \ldots,$ where $y_t \in \{-1, 1\}$ (stock goes up or down).
- Forecaster makes prediction $\hat{p}_t \in [-1, 1]$ (buy/sell $\hat{p}_1$ shares of stocks) according to some algorithm $\hat{P}$.
- At each time $t$, investor loss (or gain) $-\hat{p}_t y_t$. Cumulative loss (or gain)

$$L(\hat{P}, y_T) := -\sum_{t=1}^{T} \hat{p}_t y_t$$

- Two naive experts: one predicts 1 (always goes up); the other one predicts -1(always goes down).

Cover proved

$$Regret(T) = \Theta(\sqrt{T}).$$

# PDE approach

This problem was considered by Kohn using PDE technique.

- Instead of proposing an algorithm and then proving the asymptotic bound, PDE method focuses on the optimal strategy

# PDE approach

This problem was considered by Kohn using PDE technique.

- Instead of proposing an algorithm and then proving the asymptotic bound, PDE method focuses on the optimal strategy
- Typical Goal: minimize the worst-case regret with respect to the best performing expert.

# PDE approach

This problem was considered by Kohn using PDE technique.

- Instead of proposing an algorithm and then proving the asymptotic bound, PDE method focuses on the optimal strategy
- Typical Goal: minimize the worst-case regret with respect to the best performing expert.
- More general goal: minimize the worst case value of

$$\phi(\text{regret w.r.t. expert "1", regret w.r.t. expert "-1"})$$

at time $T$. ("Typical goal" is $\phi(x_1, x_2) = \max\{x_1, x_2\}$.)

# Two very simple experts

Essentially, deterministic two-player games (optimal control):

# Two very simple experts

Essentially, deterministic two-player games (optimal control):

- state space:

$$(x_1, x_2) = (\text{regret w.r.t. expert "1", regret w.r.t. expert "-1"}).$$

# Two very simple experts

Essentially, deterministic two-player games (optimal control):

- state space:

$$(x_1, x_2) = (\text{regret w.r.t. expert "1", regret w.r.t. expert "-1"}).$$

- control: prediction $\hat{p}_t \in [-1, 1]$ (investor's purchase bounded by 1)

# Two very simple experts

Essentially, deterministic two-player games (optimal control):

- state space:

$$(x_1, x_2) = (\text{regret w.r.t. expert “1”, regret w.r.t. expert “-1”}).$$

- control: prediction $\hat{p}_t \in [-1, 1]$ (investor's purchase bounded by 1)
- value function: $v(x, t)$ = optimal time $T$ result, starting from relative regrets $x = (x_1, x_2)$ at time $t$.

dynamic programming principle:

$$\begin{aligned} v(x_1, x_2, t) &= \min_{|p|\leq 1} \max_{b=\pm 1} v(\text{new position}, t+1) \\ &= \min_{|p|\leq 1} \max_{b=\pm 1} v(x_1 + b(1-p), x_2 - b(1+p), t+1) \end{aligned}$$

for $t < T$, with final-time condition $v(x, T) = \phi(x)$.

# dynamic Programming Principle

Recall: $(x_1, x_2) =$ (regret w.r.t. "'1" expert, regret w.r.t. "-1" expert), where regret = (investor's loss) - (expert's loss).

If the investor buys $p$ shares and the market goes up, investor loss $-p$, the "+" expert loss $-1$, the "-1" expert loss 1. So the state moves from $(x_1, x_2)$ to $(x_1 - (1-p), x_2 + (1+p))$.

Similarly, if investor buys $p$ shares and market goes down, state move from $(x_1, x_2)$ to $(x_1 + (1-p), x_2 - (1+p))$.

Hence the dynamic programming principle:

$$v(x_1, x_2, t) = \min_{|p|\le 1} \max_{b=\pm 1} v(x_1 + b(1-p), x_2 - b(1+p), t+1).$$

- Want to know how the regret accumulates as $T \to \infty$.

- Want to know how the regret accumulates as $T \to \infty$.
- To access this question, it is natural to rescale the problem and look for a continuum limit.

- Want to know how the regret accumulates as $T \to \infty$.
- To access this question, it is natural to rescale the problem and look for a continuum limit.
- Our scaling is like the passage from random walk to Brownian motion.
  - Let $\xi_1, \xi_2, \ldots$ be i.i.d. random variable with mean 0 and variance 1. For each $n$, define a continuous-time stochastic processes $\{W_n(t)\}$

$$W_n(t) = \frac{1}{\sqrt{n}} \sum_{1 \le j \le [nt]} \xi_j.$$

Then $W_n(t) \to W_t$, where $W_t$ is a standard Brownian motion.

- So we consider a scaled version of the problem: stock moves $\pm\epsilon$, time steps are $\epsilon^2$, and the scaling of regret is $\epsilon^2$. The value function is still the optimal time-T result. The dynamic programming principle becomes

$$w^{\epsilon}(x_1, x_2, t) = \min_{|p|\leq 1} \max_{b=\pm 1} w^{\epsilon}(x_1 + \epsilon b(1-p), x_2 - \epsilon b(1+p), t + \epsilon^2)$$

- So we consider a scaled version of the problem: stock moves $\pm\epsilon$, time steps are $\epsilon^2$, and the scaling of regret is $\epsilon^2$. The value function is still the optimal time-T result. The dynamic programming principle becomes

$$w^{\epsilon}(x_1, x_2, t) = \min_{|p|\leq 1} \max_{b=\pm 1} w^{\epsilon}(x_1 + \epsilon b(1-p), x_2 - \epsilon b(1+p), t + \epsilon^2)$$

- We expect as $\epsilon \to 0$, $w^{\epsilon}(x, t) \to w(x, t)$ and $w$ solves some PDEs.

1. Use the Taylor's expansion to expand

$$w(x_1 + \epsilon b(1-p), x_2 - \epsilon(1+p), t + \epsilon^2),$$

then plug into the dynamic programming equation.

2. Investor choose $p$ to make $O(\epsilon)$ terms vanishes; this gives

$$p = \frac{\partial_1 w - \partial_2 w}{\partial_1 w + \partial_2 w}.(\text{optimal strategy})$$

3. The $O(\epsilon^2)$ are insensitive to $b = \pm 1$; they give the nonlinear PDE

$$w_t + 2 < D^2 w \frac{\nabla^\perp w}{\partial_1 w + \partial_2 w}, \frac{\nabla^\perp w}{\partial_1 w + \partial_2 w} >= 0,$$

where $\nabla^\perp w = (\partial_2 w, -\partial_1 w)$. Prescribe the final value $w = \phi$ at $T$.

# Conclusion

- if $\phi$ smooth

$$\|w^{\epsilon}(x,t) - w(x,t)\| \le C\epsilon,$$

where $C$ independent of $t$.

- if $\phi(x) = \max\{x_1, x_2\}$

$$\|w^{\epsilon}(x,t) - w(x,t)\| \le C\epsilon \log \epsilon.$$

- Equation can be solved explicitly.

## Discussion

- Stock prediction problem has a continuous-time limit. The PDE has a rather explicit solution.

## Discussion

- Stock prediction problem has a continuous-time limit. The PDE has a rather explicit solution.
- Problem is considered as a deterministic two-person game.

# Discussion

- Stock prediction problem has a continuous-time limit. The PDE has a rather explicit solution.
- Problem is considered as a deterministic two-person game.
- One can consider more complex situation, e.g., more experts, more intelligent experts, etc.

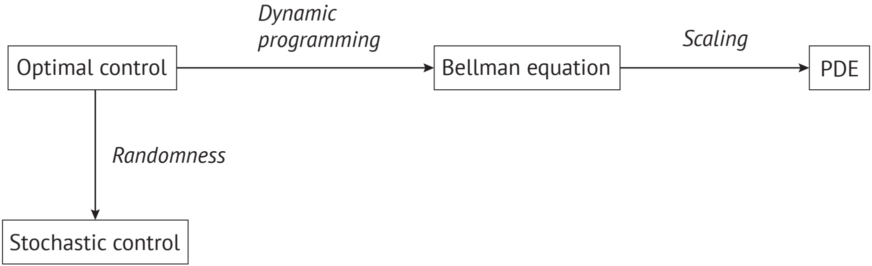
# Discussion

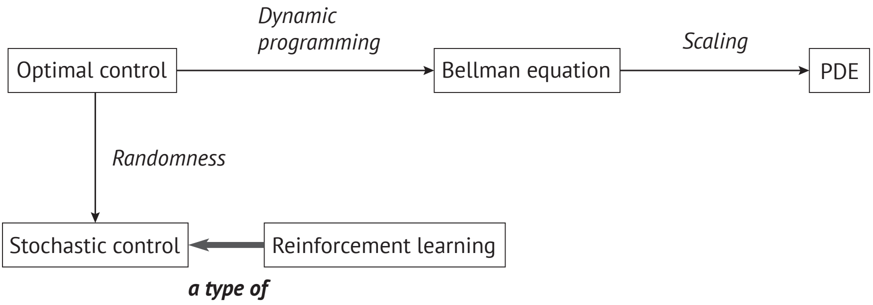
- Stock prediction problem has a continuous-time limit. The PDE has a rather explicit solution.
- Problem is considered as a deterministic two-person game.
- One can consider more complex situation, e.g., more experts, more intelligent experts, etc.
- dynamic programming principle can be hard to derive.

$\boxed{\text{Optimal control}}$

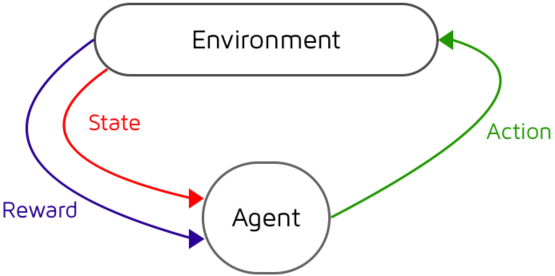
$$\boxed{\text{Optimal control}} \xrightarrow{\textit{Dynamic programming}} \boxed{\text{Bellman equation}}$$

Optimal control $\xrightarrow{\text{\textit{Dynamic programming}}}$ Bellman equation $\xrightarrow{\text{\textit{Scaling}}}$ PDE


Optimal control
Dynamic programming
Bellman equation
Scaling
PDE
Randomness
Stochastic control


Dynamic programming
Scaling
Optimal control
Bellman equation
PDE
Randomness
Stochastic control
Reinforcement learning
a type of

# Reinforcement Learning




S: Set of states

A: Set of actions

R: Reward function. $\forall s_t, a_t,\ r_t = R(s_t, a_t) \in \mathbb{R}$

- Markov decision process (MDP) is a discrete time stochastic control process. The value function $v : S \to \mathbb{R}$ satisfies a discrete Bellman equation.

- Markov decision process (MDP) is a discrete time stochastic control process. The value function $v : S \to \mathbb{R}$ satisfies a discrete Bellman equation.
  - Given a policy $\pi$ with infinite horizon for any state $x \in X$

$$V_\pi(x) = R(x, \pi(x)) + \gamma \sum_{x' \in X} P(x'|x, \pi(x)) V_\pi(x')$$

  - Optimal policy

$$V^*(x) = \max_{a \in A} \{R(x, a) + \gamma \sum_{x' \in X} P(x'|x, a) V_\pi(x')\}$$

- Continuous time and state: PDEs.
  - Dynamics of the state

$$dX_s = b(X_s, a_s)\, ds + \sigma(X_s, a_s)\, dW_s, \tag{1}$$

where $W_s$ is Brownian motion and $a_s$ is control process.

- Continuous time and state: PDEs.
  - Dynamics of the state

$$dX_s = b(X_s, a_s)\, ds + \sigma(X_s, a_s)\, dW_s, \tag{1}$$

where $W_s$ is Brownian motion and $a_s$ is control process.
  - Reward

$$R(t, x, a) = \mathbb{E}\Big[ \int_t^T f(X_s^{t,x}, a_s)\, ds + g(X_T^{t,x}) \Big],$$

where $X^{t,x}$ is the solution of (1) starting from $X_t = x$.

- Continuous time and state: PDEs.
  - Dynamics of the state

$$dX_s = b(X_s, a_s)\, ds + \sigma(X_s, a_s)\, dW_s, \tag{1}$$

    where $W_s$ is Brownian motion and $a_s$ is control process.
  - Reward

$$R(t, x, a) = \mathbb{E}\Big[ \int_t^T f(X_s^{t,x}, a_s)\, ds + g(X_T^{t,x}) \Big],$$

    where $X^{t,x}$ is the solution of (1) starting from $X_t = x$.
  - Value function

$$v(t, x) = \sup_{a \in \mathcal{A}} R(t, x, a).$$

- Continuous time and state: PDEs.
  - Dynamics of the state

$$dX_s = b(X_s, a_s)\, ds + \sigma(X_s, a_s)\, dW_s, \tag{1}$$

    where $W_s$ is Brownian motion and $a_s$ is control process.
  - Reward

$$R(t, x, a) = \mathbb{E}\Big[\int_t^T f(X_s^{t,x}, a_s)\, ds + g(X_T^{t,x})\Big],$$

    where $X^{t,x}$ is the solution of (1) starting from $X_t = x$.
  - Value function

$$v(t, x) = \sup_{a \in \mathcal{A}} R(t, x, a).$$

  - HJB equation

$$\frac{\partial v}{\partial t} + \sup_{a \in \mathcal{A}} \big(L_a v + f(x, a)\big) = 0 \quad \text{in} [0, T] \times \mathbb{R}^d,$$

    where $L_a$ is the infinitesimal generator of the process (1).

# Second-order Linear Elliptic PDEs

A classical example is

$$\Delta u(x) = \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} = 0.$$

Let $W_t$ be a Brownian motion starting from $x$. Then

$$\lim_{t \to 0} \frac{E^x u(W_t) - u(x)}{t} = \frac{1}{2} \Delta u.$$

General diffusion process

$$L_a u = \frac{1}{2} Tr(\sigma \sigma^T D^2 u) = \frac{1}{2} \sum_{i,j=1}^{d} a_{ij} D_{ij} u,$$

where $\{a_{ij}\}$ is some positive-definite matrix.

# HJB equation

Fully nonlinear equation ($v$ depends on $(t, x)$)

$$\partial_t v + \sup_{a \in \mathcal{A}} (L_a v + f(x, a)) = 0,$$

where $L_a$ is a second-order linear elliptic operator.

After deriving the HJB equation formally, the equation holds in a weak sense. The regularity of the solution is very low.

- Easy to find solution in the space of continuous function
- Equation holds in a very weak sense (viscosity solution).
- Important problem in the PDE community to prove the existence of classical solution.

# Mathematical Result

Recall:

$$dX_s = b(X_s, a_s)\,ds + \sigma(X_s, a_s)\,dW_s.$$

Under some regularity assumption on the coefficients $\sigma$, $b$, then the solution to the HJB

$$\partial_t u + \sup_{a \in \mathcal{A}} \big( L_a u + f(x, a) \big) = 0,$$

is a classical solution. Rigorously, $u \in C^{2,\alpha}$ for some $\alpha > 0$.

The theoretical results helps to build numerical scheme to solve the HJB.

# Jump processes and Nonlocal equations

Brownian motion is a continuous stochastic process. Consider more general process: Lévy processes (jump processes)

$$dX_s = b(X_s, a_s)\, ds + \sigma(X_s, a_s)\, dY_s,$$

where $Y_s$ is a Lévy process. The infinitesimal generator of such processes is nonlocal

$$L_a u = \int_{\mathbb{R}^d} (u(x+y) + u(x-y) - 2u(x)) \frac{K_a(y)}{|y|^{d+\sigma}}\, dy,$$

and $\sigma \in (0, 2)$.

**Theorem**

*Let $v$ be a weak solution to*

$$v_t + \sup_a (L_a v + f_a) = 0,$$

*where $L_a$ is a nonlocal operator defined previously. Then the solution $v$ is a classical solution and derivatives of $v$ are bounded.*

Thank you.