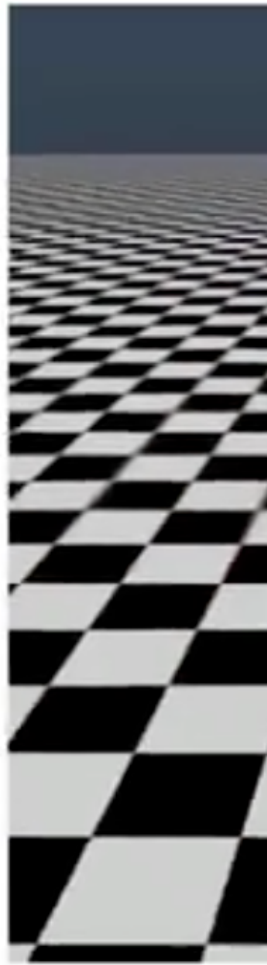
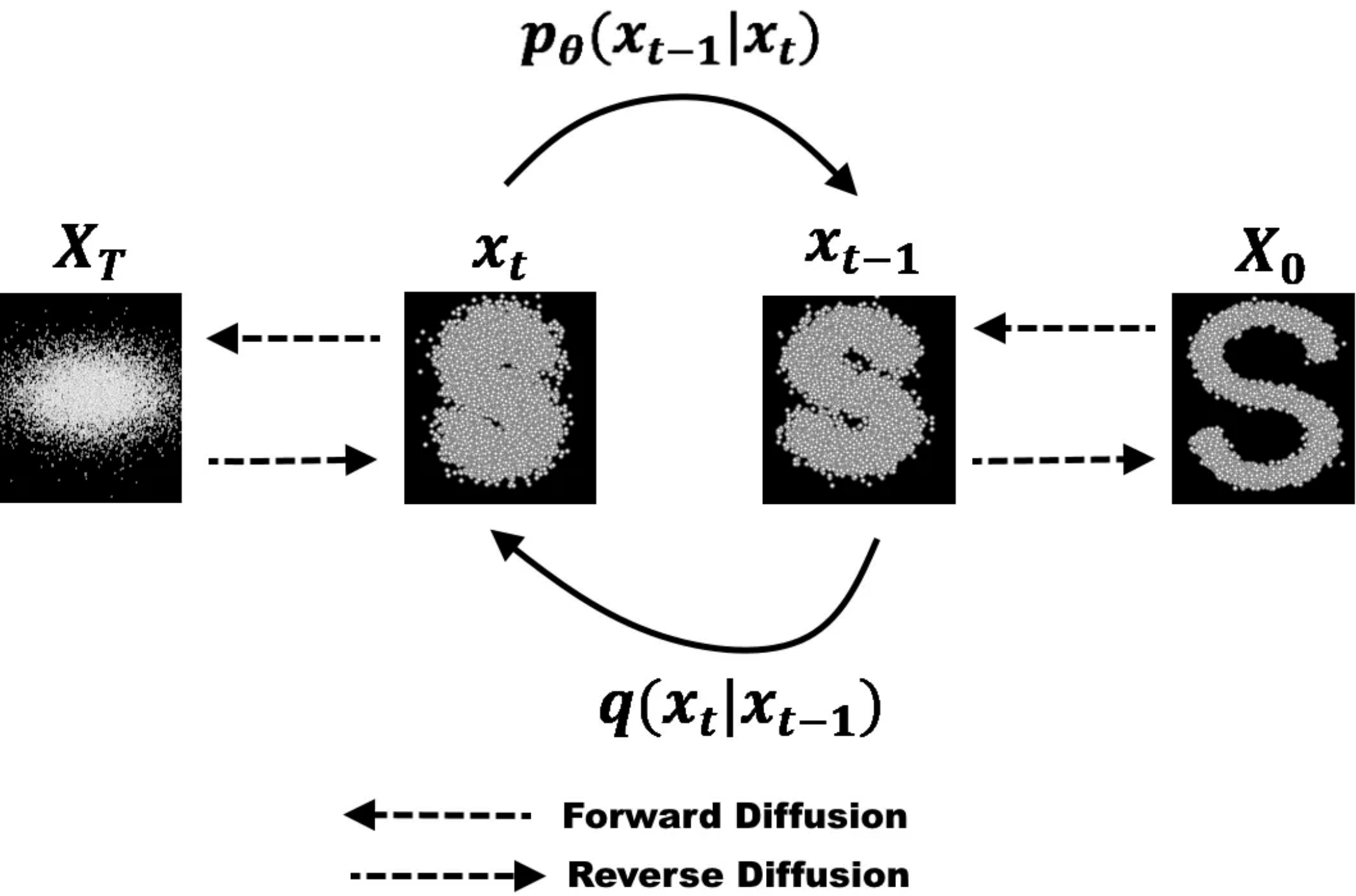


# Preference Learning with Response Time: Robust Losses and Guarantees

Ayush Sawarni, Vasilis Syrgkanis

# Learning from Human Preferences



Advanced Data Analysis

¿Qué respuesta prefieres?  
Tu elección ayudará a mejorar ChatGPT.

Respuesta 1

Certainly! Here are some exhaustive tests and edge tests for the 'SHR' operation:

1. **Basic Test:** Ensuring that a general right shift works.

2. **VF Test:** Ensuring that the VF register is correctly set based on the least-significant bit of Vx.

3. **Zero Test:** Making sure that shifting a zero register doesn't produce any unexpected results.

4. **All Ones Test:** Testing a value of all ones to ensure correct shifting.

5. **Edge Test:** Shifting the smallest and largest possible values.

6. **Repeated Shift Test:** Repeatedly shifting until the register becomes zero.

rust

Copy code

```
#[test]
fn test_shr_basic() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0x08); // 0b1000
    cpu.execute(0x8006); // SHR V0
    assert_eq!(cpu.memory.read_reg(0), 0x04); // 0b0100
}

#[test]
fn test_shr_vf() {
    let mut cpu = CPU::new();

    // Least-significant bit is 1
    cpu.memory.write_reg(0, 0x03); // 0b0011
    cpu.execute(0x8006); // SHR V0
    assert_eq!(cpu.memory.read_reg(0xF), 1);

    // Least-significant bit is 0
    cpu.memory.write_reg(0, 0x02); // 0b0010
    cpu.execute(0x8006); // SHR V0
    assert_eq!(cpu.memory.read_reg(0xF), 0);
}

#[test]
fn test_shr_zero() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0x00);
    cpu.execute(0x8006); // SHR V0
    assert_eq!(cpu.memory.read_reg(0), 0);
    assert_eq!(cpu.memory.read_reg(0xF), 0);
}

#[test]
fn test_shr_all_ones() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0xFF);
    cpu.execute(0x8006); // SHR V0
    assert_eq!(cpu.memory.read_reg(0), 0x7F);
    assert
```

Respuesta 2

To create exhaustive tests for the 'SHR' (shift right) opcode '8xy6', we need to consider a few things:

1. **The least-significant bit (LSB):** This bit will determine if 'VF' is set to '1' or '0'.

2. **Division by 2:** After the operation, 'Vx' should be half of its original value.

3. **Edge cases:** These include situations like 'Vx' being '0' or '0xFF' and other boundary values.

Let's design the tests based on the original interpretation (where 'Vy' is ignored):

Test 1: Basic operation with LSB as 1

rust

Copy code

```
#[test]
fn test_shr_vx_lsb_1() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0b1010_1011); // V0 = 0b10101011

    cpu.execute(0x8006); // SHR V0

    assert_eq!(cpu.memory.read_reg(0xF), 1); // VF is set
    assert_eq!(cpu.memory.read_reg(0), 0b1010_1011 >> 1);
}

Test 2: Basic operation with LSB as 0
```

rust

Copy code

```
#[test]
fn test_shr_vx_lsb_0() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0b1010_1010); // V0 = 0b10101010

    cpu.execute(0x8006); // SHR V0

    assert_eq!(cpu.memory.read_reg(0xF), 0); // VF is not set
    assert_eq!(cpu.memory.read_reg(0), 0b1010_1010 >> 1);
}

Test 3: Edge case with 'Vx' as '0'
```

rust

Copy code

```
#[test]
fn test_shr_vx_zero() {
    let mut cpu = CPU::new();
    cpu.memory.write_reg(0, 0x00); // V0 = 0x00

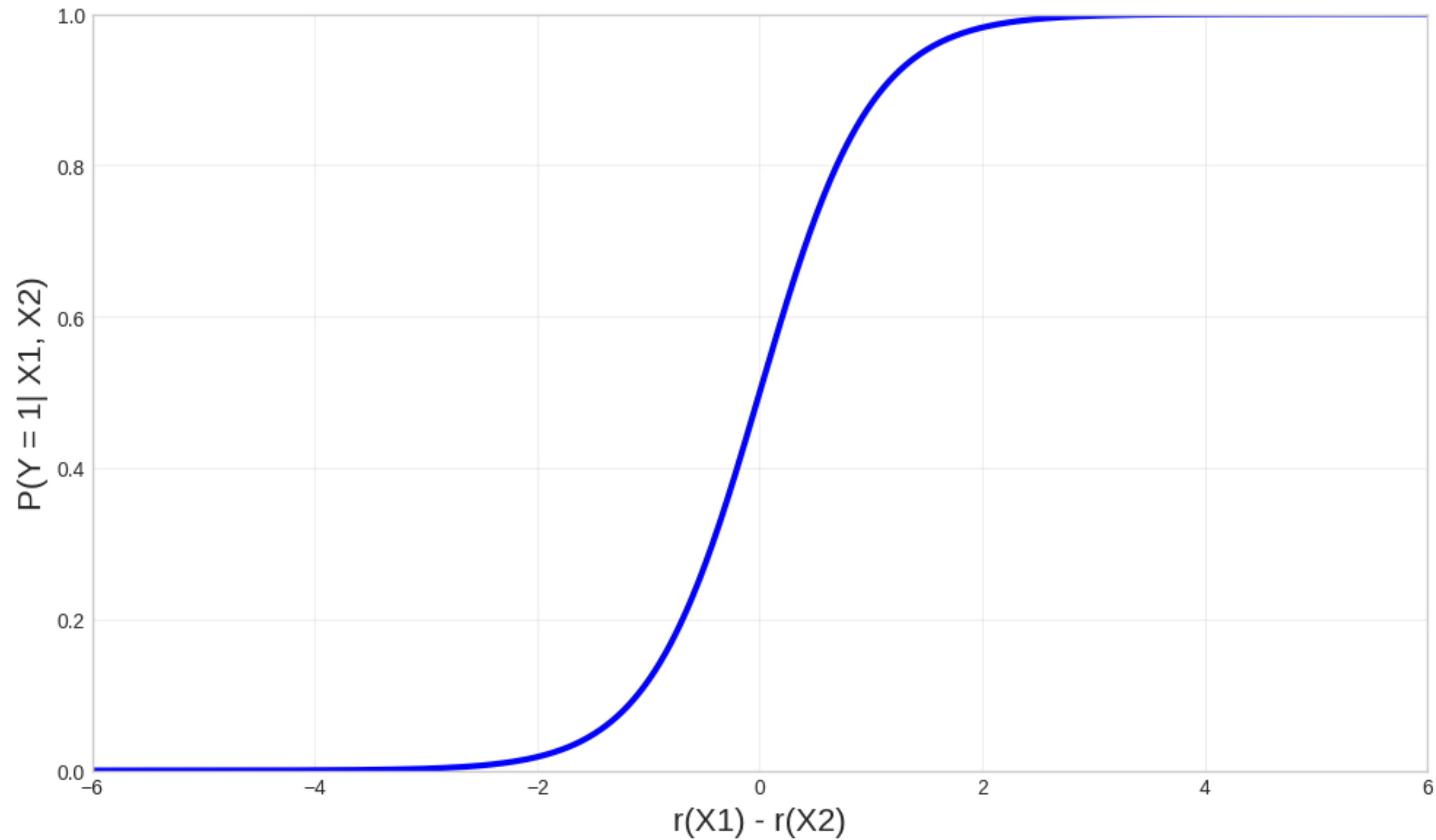
    cpu.execute(0x8006); // SHR V0
```

Detener generación

Envía un mensaje

ChatGPT may produce inaccurate information about people, places, or facts. [ChatGPT September 25 Version](#)

# Preference Learning

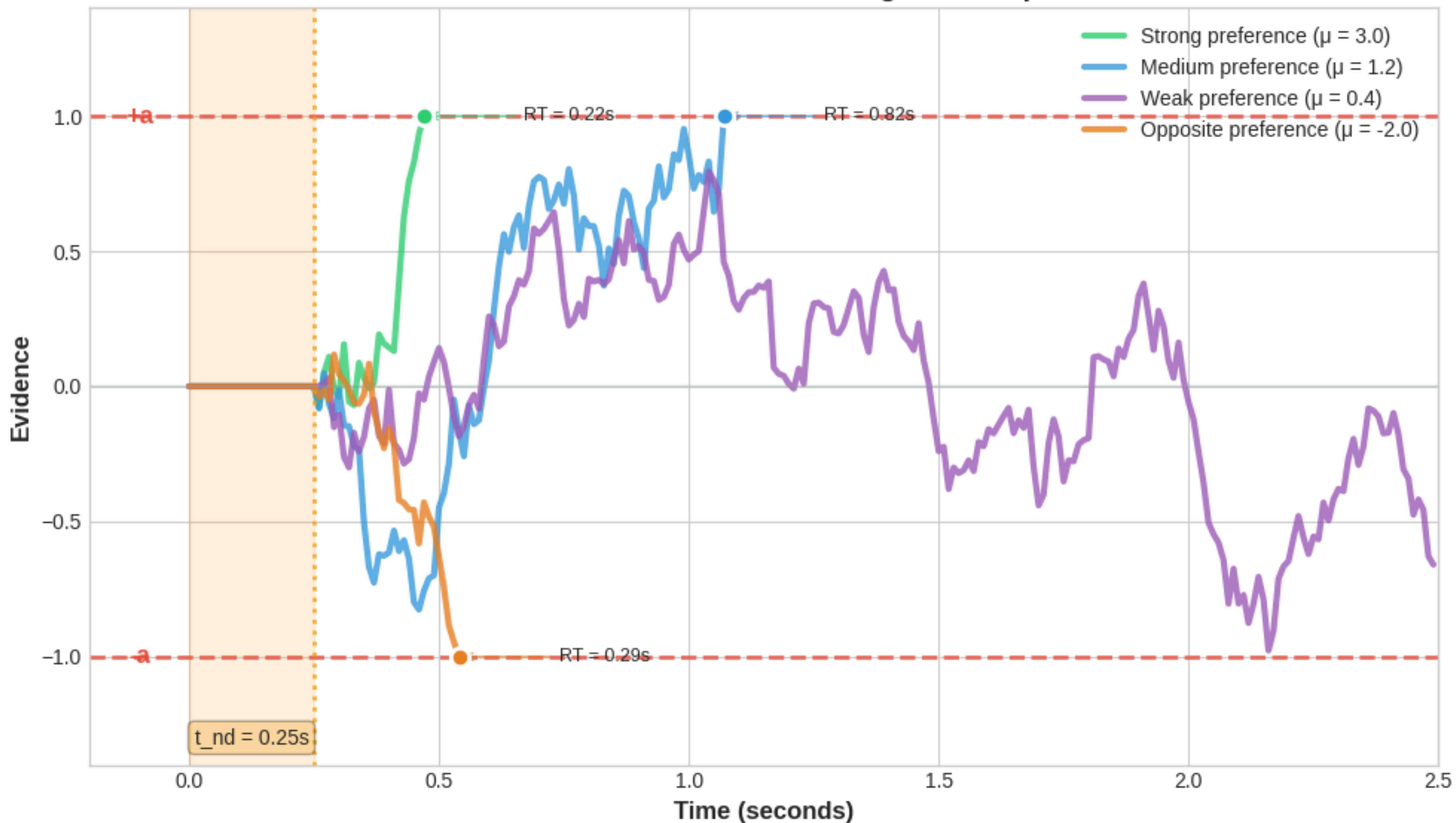


**Observe:** Signal strength reduces as  $r(X1) - r(X2)$  grows. **Vanishing** gradient problem.  
Learning reward model gets harder.

# Using response time

- Intuitively: stronger preference would lead to a shorter response time.
- Revealed strength of preferences. (*Chabris et al., 2008*), (*Konovalov, Krajbich 2019*), (*Bavard et al., 2024*).
- Cognitive Psychology, Neuroscience, Economics - Drift Diffusion Models.  
EZ-Diffusion Model (Wagenmakers et. al. 2007) - computationally efficient parameter estimation.
- Based the reward difference  $r(X)$  the decision making happens via the following process.
  - A. User spends  $t_{\text{nondec}}$  time encoding the choices.
  - B. Makes a decision  $Y$  after and “evidence accumulation” phase of duration  $T$
  - C. Evidence accumulation is modelled as a diffusion process with drift  $r(X)$

Drift Diffusion Model: Preference Strength → Response Time



# Learning $r$

- MLE of response time?
- Likelihood is **not** tractable. :(

$$f_T(t) = \frac{a}{\sqrt{2\pi t^3}} \sum_{k=-\infty}^{\infty} (2k+1) \exp \left[ -\frac{((2k+1)a - t \cdot r(X))^2}{2t} \right], \quad t > 0.$$

- Maybe use the moment expressions. Can show the following relation.

$$r(X) = \frac{\mathbb{E}[Y \mid X]}{\mathbb{E}[T \mid X]}.$$

- Loss function using the above relation.

# Goal: Construct a loss function to learn $r$

- **Objective:** given  $Z_i = (X_i^1, X_i^2, Y_i, T_i)$   $i = 1, \dots, n$  - learn the reward function  $r$
- With **only** preferences, the following loss function naturally follows from BT model.

- $\mathcal{L}^{\text{logloss}}(r) = \mathbb{E} [\log(1 + \exp(-2 Y r(X)))]$ .

- **Naive loss (with response time):** We may start with the following loss function

- $\mathcal{L}^{\text{non-ortho}} = \mathbb{E} \left[ (Y - r(X) \mathbb{E}[T | X])^2 \right]$

$\mathbb{E}[T | X]$  is unknown, learn  $t_o(X) = \mathbb{E}[T | X]$  Loss function is sensitive to the estimate of  $t_o$ .

- Learning  $t_o$  can be hard.

- $t_o$  may be highly non linear. **Example** - linear  $r(X) = \langle \theta, x \rangle$ ,  $t(X) = \frac{\tanh(r(X))}{r(X)}$  (non-linear, MSE non-convex in  $\theta$ ).

# Learning with Nuisances - Orthogonal Statistics

Neyman orthogonality Loss functions that are less sensitive to errors in Nuisance estimate.

Directional derivative (Gateaux derivative) of a function  $F: \mathcal{F} \rightarrow \mathbb{R}$  at  $f$  direction  $h$  is defined by

$$D_f F(f)[h] = \left. \frac{d}{dt} F(f + th) \right|_{t=0}$$

**Goal:** Orthogonal loss function with respect to response time function  $t(X)$ .

# Orthogonal Loss

$$\mathcal{L}(r; \mathbf{r}, t) = \mathbb{E} \left[ (Y - (T - t(X))\mathbf{r}(X) - r(X)t(X))^2 \right]$$

- $t$  : an estimate of  $t_o(X)$
- $\mathbf{r}$  : a crude estimate of reward function (minimizing log-loss for example)
- Orthogonal wrt to nuisance pair  $g = (t, \mathbf{r})$

## First stage:

- Minimize the log-loss function to estimate reward model  $\mathbf{r}$ .
- Can fit a neural network to estimate  $\mathbf{t}(X)$ .

## Second stage:

- Use  $t(\cdot)$  and  $\mathbf{r}(\cdot)$  in second stage with ortho-loss function

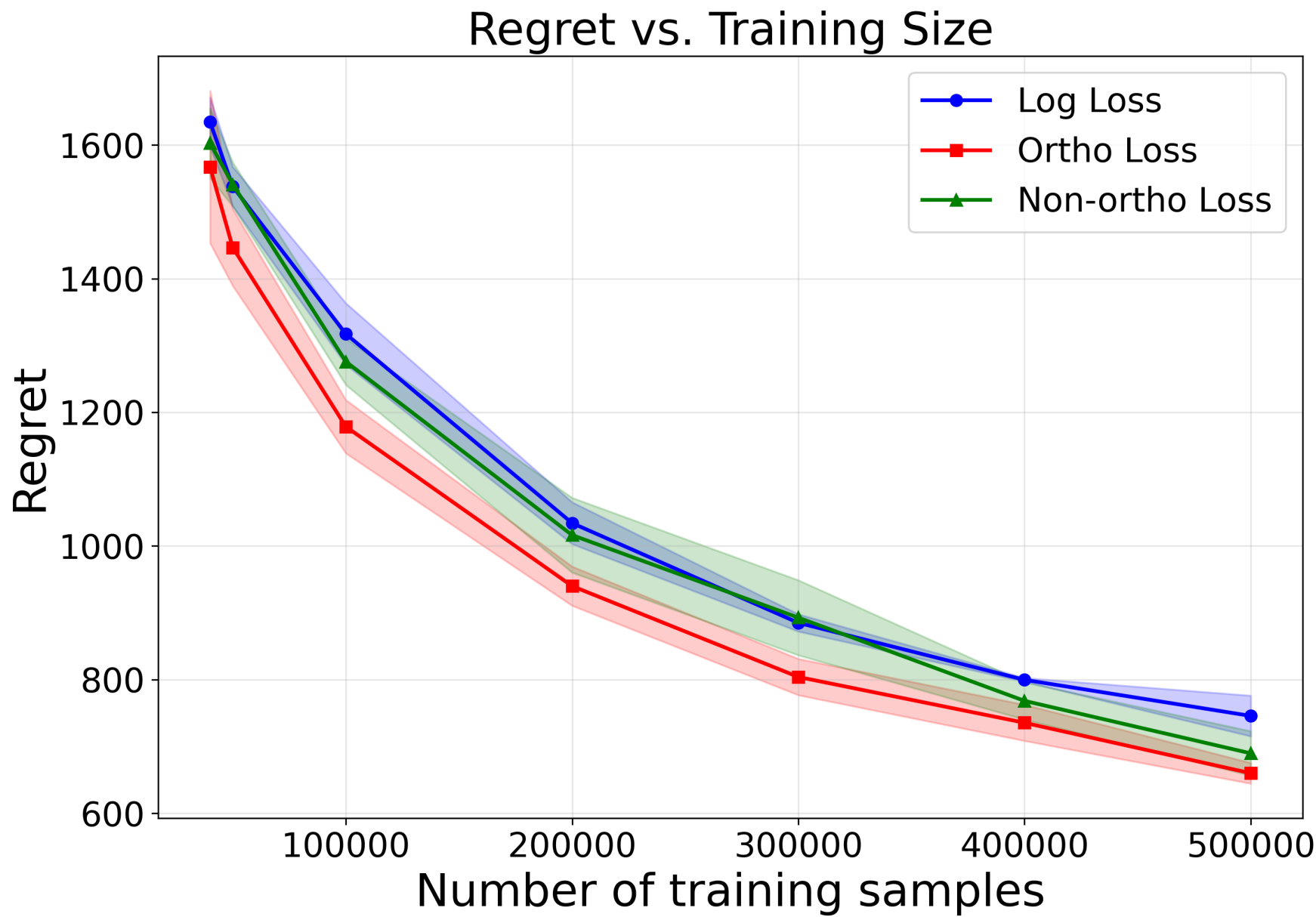
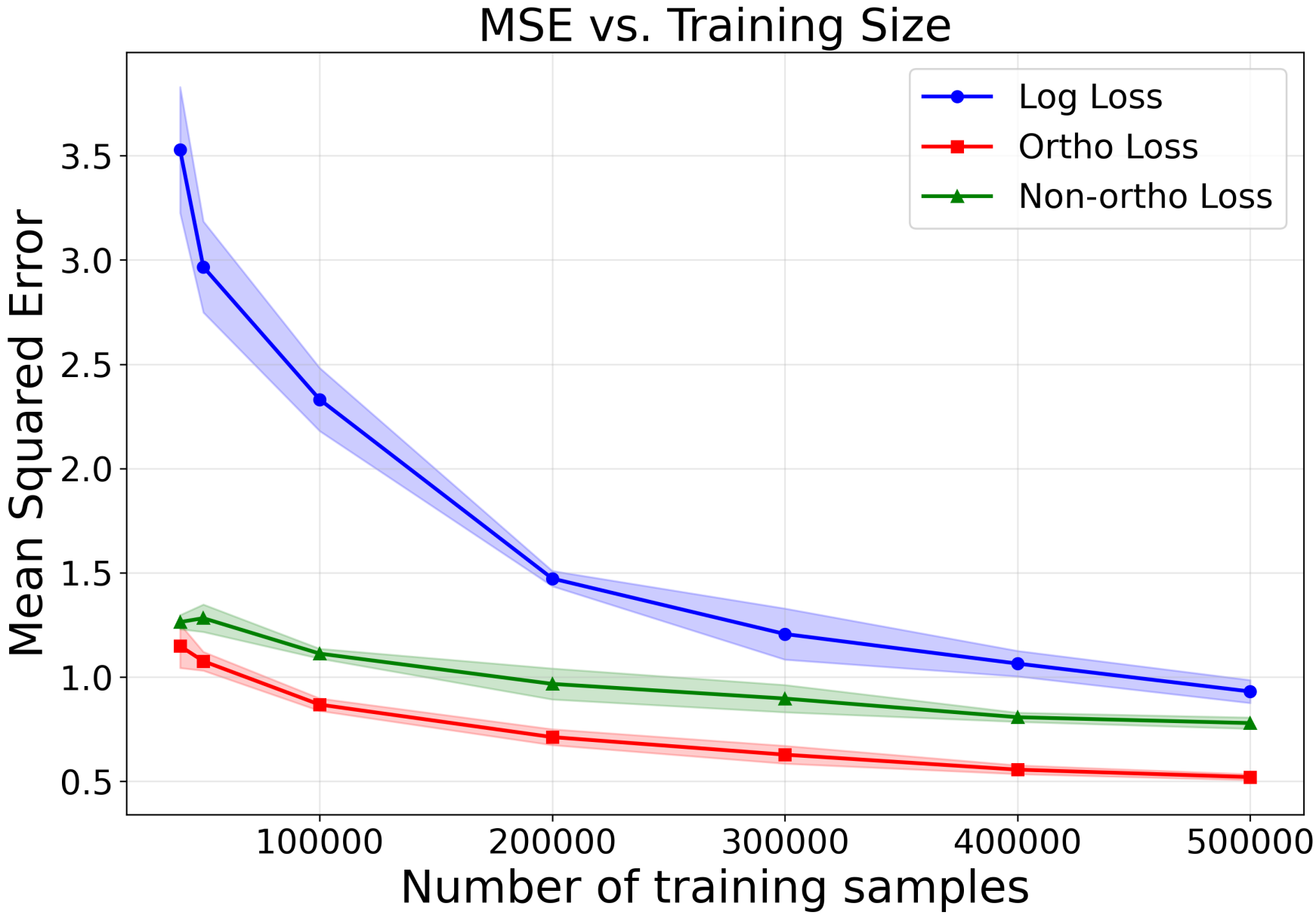
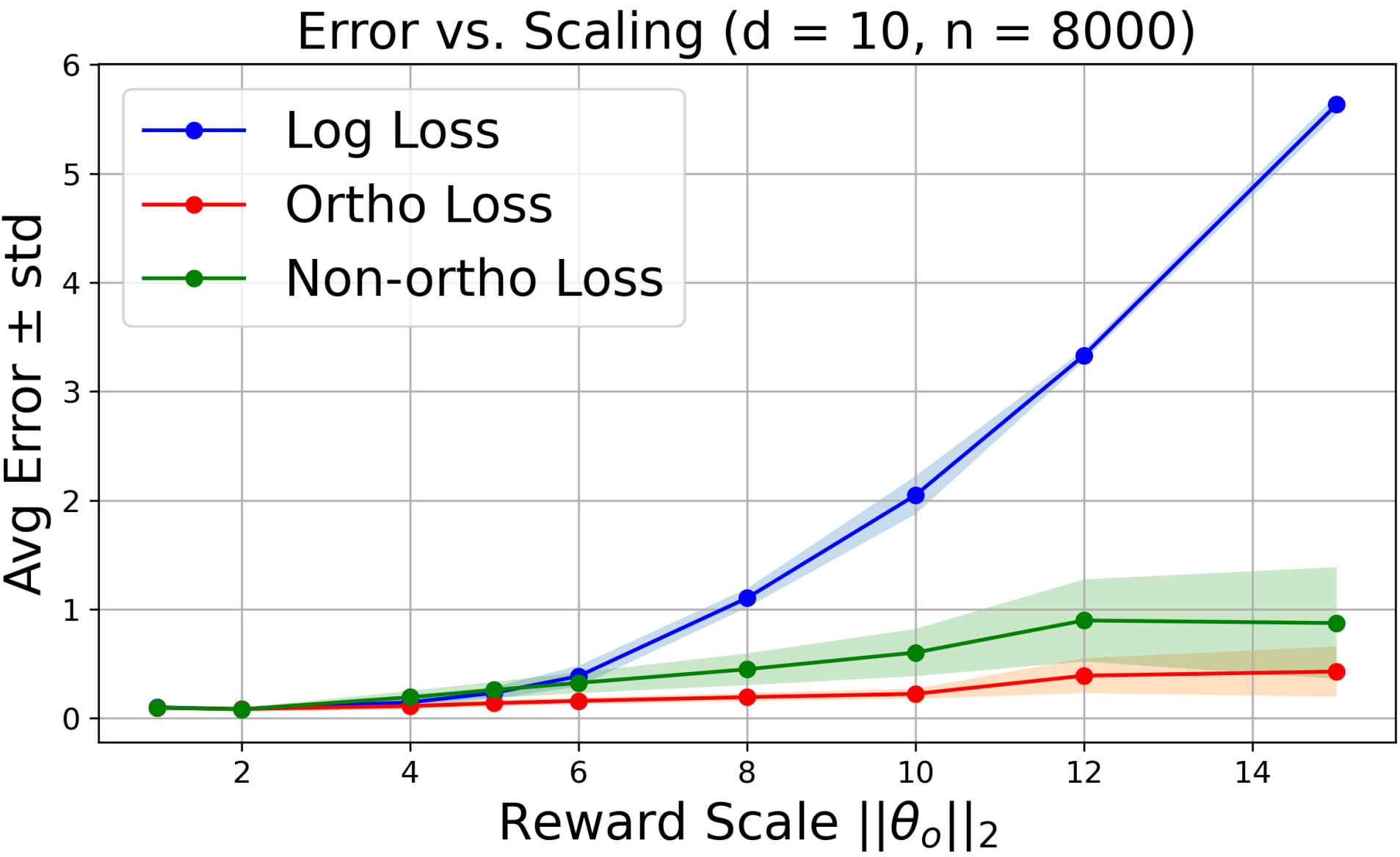
# Theoretical results:

- Linear reward  $r_o(X) := \langle \theta_o, X \rangle$  **Goal:** Estimate  $\theta_o$ . Assume  $|\theta_o| \leq S$  and  $X$  to be uniformly bounded by 1.

## Key result:

- Variance of the **choice-only estimator** grows *exponentially* with  $S$ .
- Variance of the **ortho-loss estimator** grows only *polynomially* with  $S$ .

# Experiments



# Future work - Implicit reward learning

DPO style objective: two stage learning

First stage (learn policy  $\phi$  standard DPO loss) -  $\mathbf{r}(X) = c \left( \log \frac{\phi(X^1)}{\pi_{\text{ref}}(X^1)} - \log \frac{\phi(X^2)}{\pi_{\text{ref}}(X^2)} \right)$

- Second stage

$$\mathcal{L}^{\text{ortho}}(r; \mathbf{r}) = \mathbb{E} \left[ \left( Y - T \mathbf{r}(X) + \tanh(\mathbf{r}(X)) - r(X) \frac{\tanh(\mathbf{r}(X))}{\mathbf{r}(X)} \right)^2 \right]$$

# Thank you!

<https://arxiv.org/abs/2505.22820>

