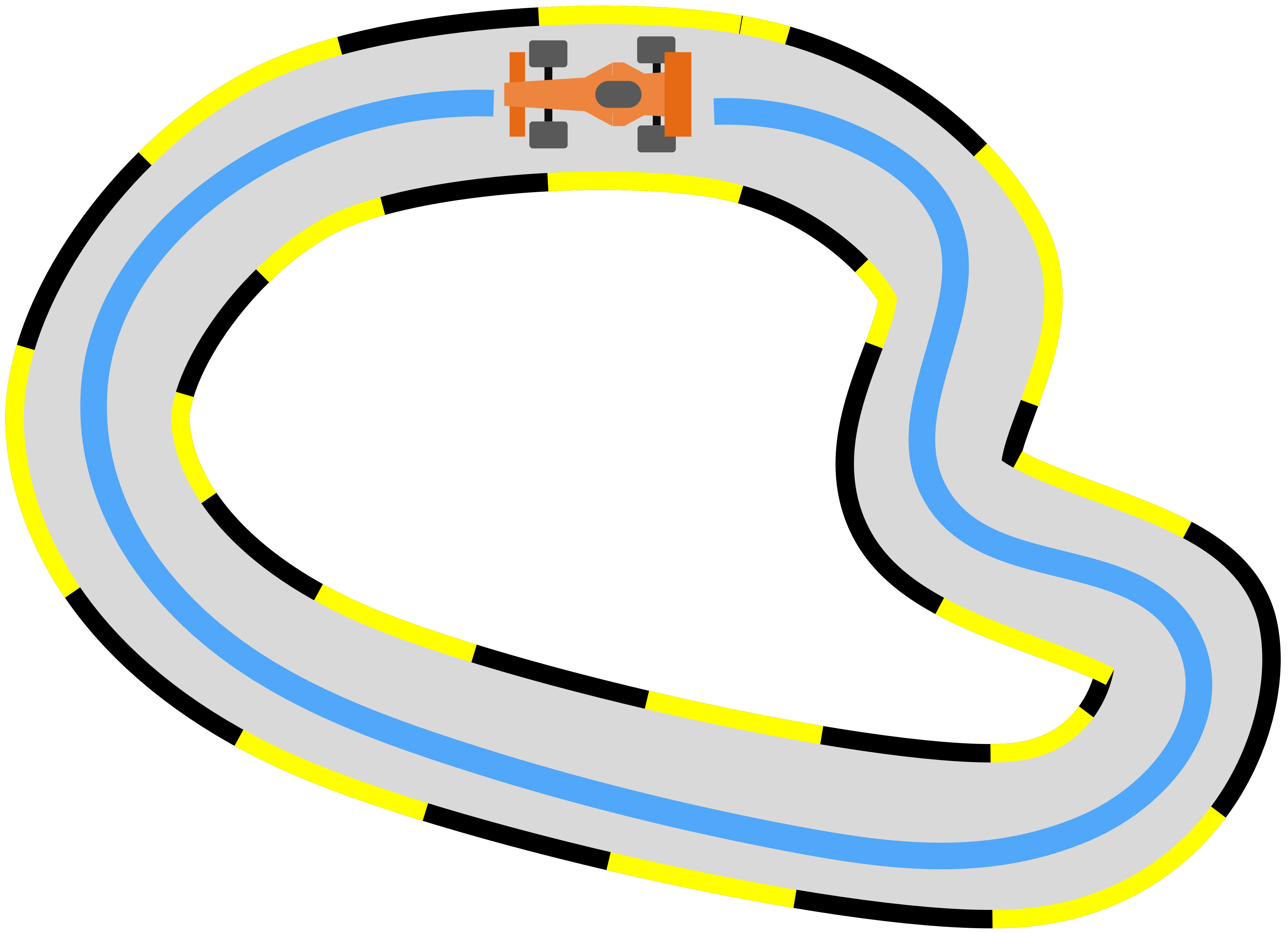
DAgger: Taming Covariate Shift with No Regret (Part 2!)

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

Behavior Cloning

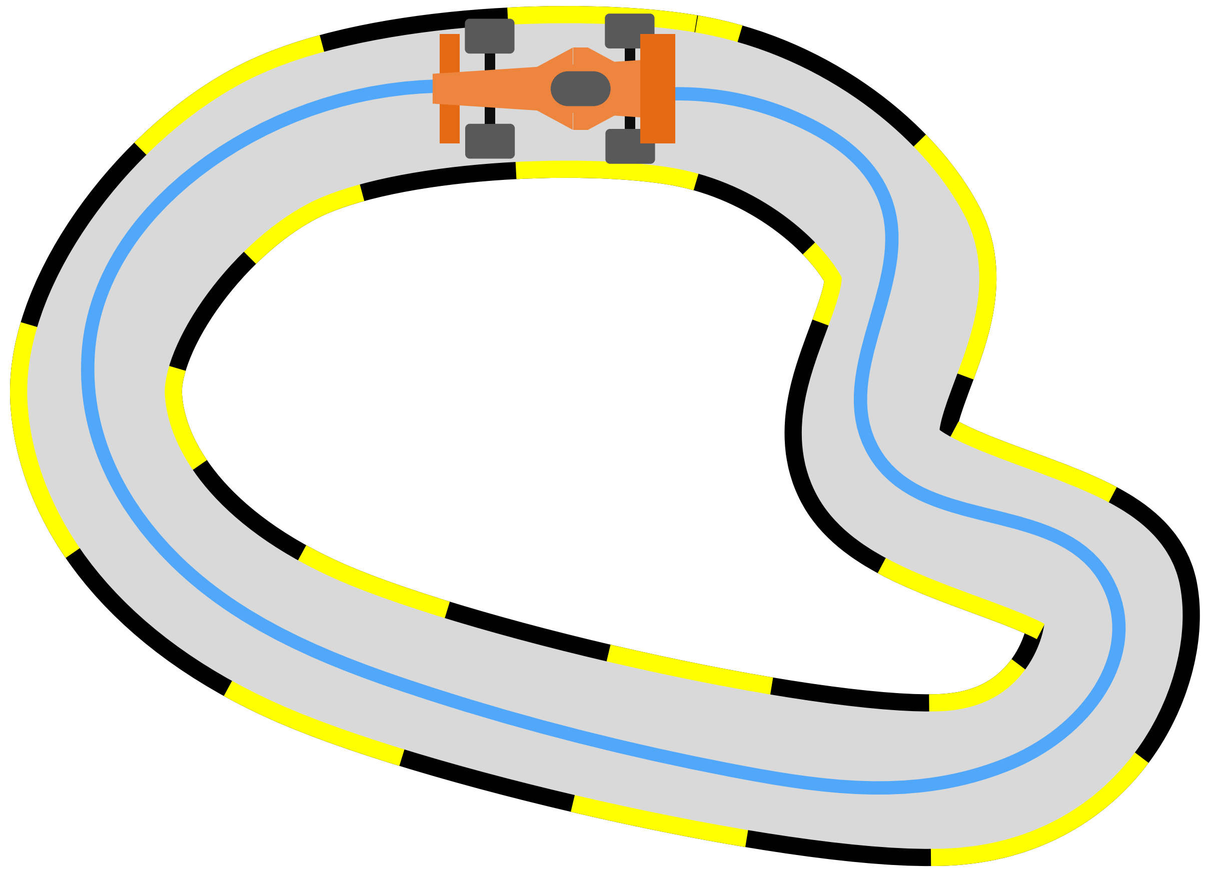


Expert runs away after demonstrations

The Big Problem with BC

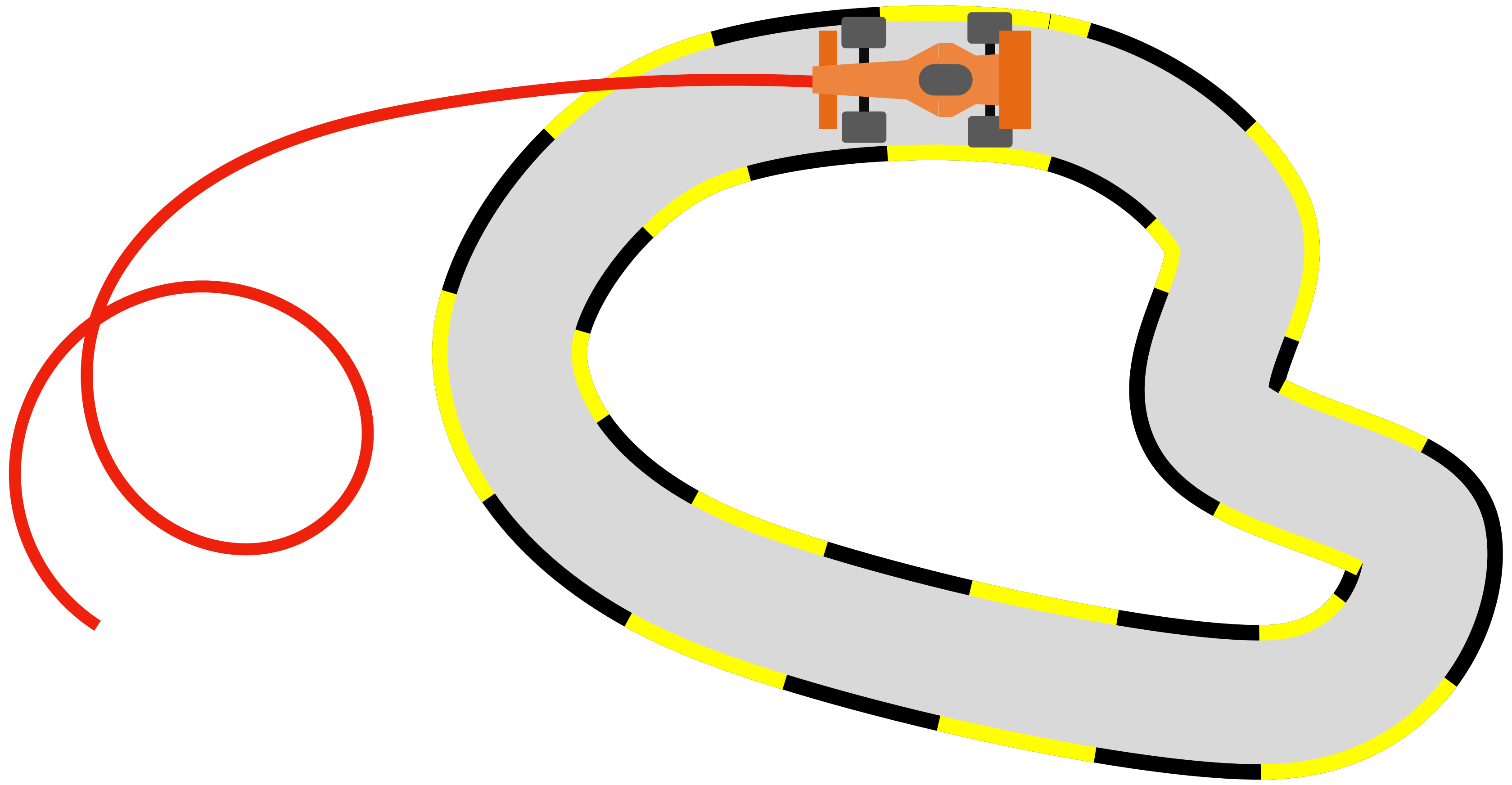
Train

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi^\star}}[\ell(s_t, \pi(s_t))]$$



Test

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi}}[\ell(s_t, \pi(s_t))]$$



The Goal

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi}}[\ell(s_t, \pi(s_t))]$$

Can we bound this to $O(\epsilon T)$?

DAGGER: A meta-algorithm for imitation learning

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

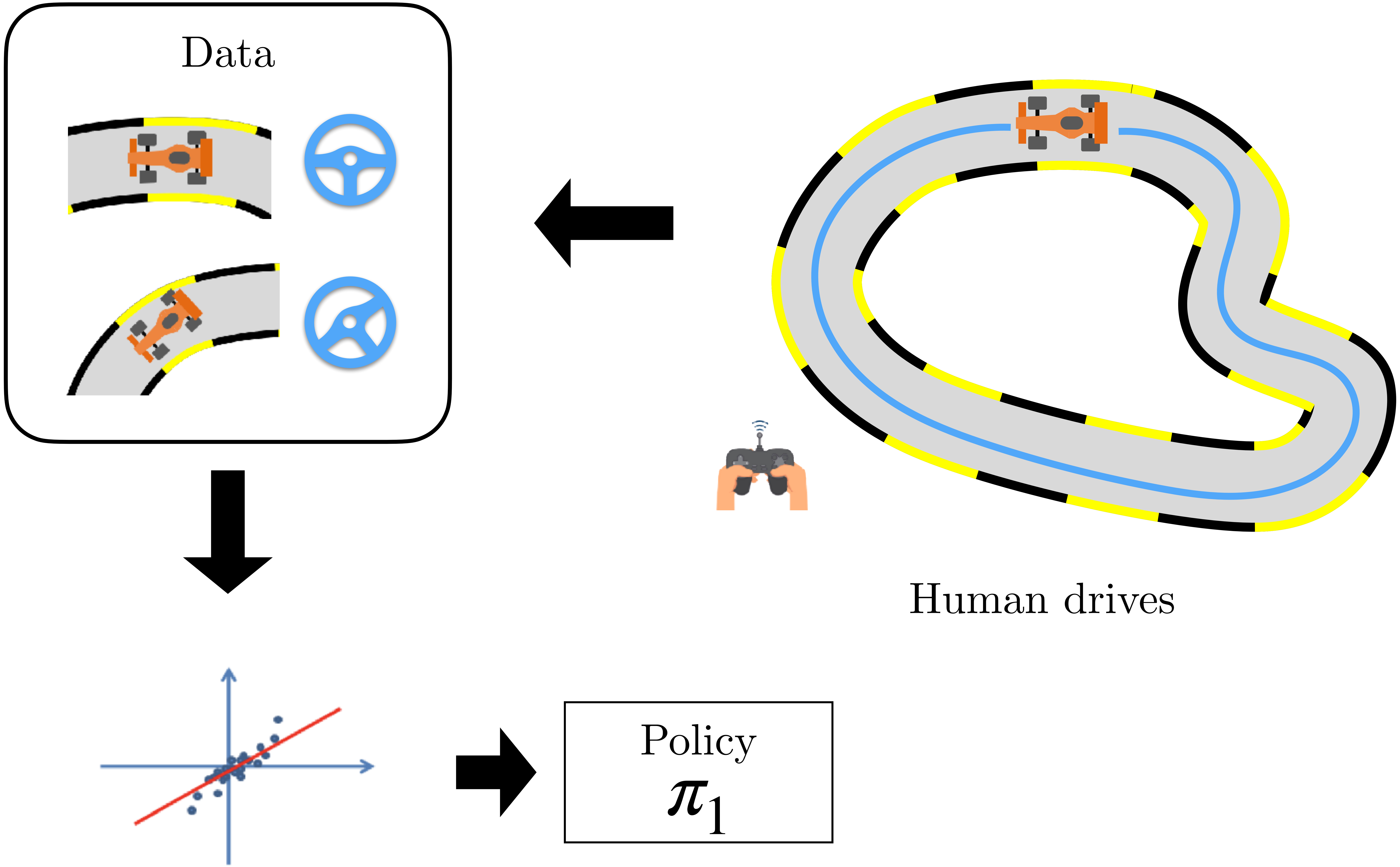
Stéphane Ross
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu

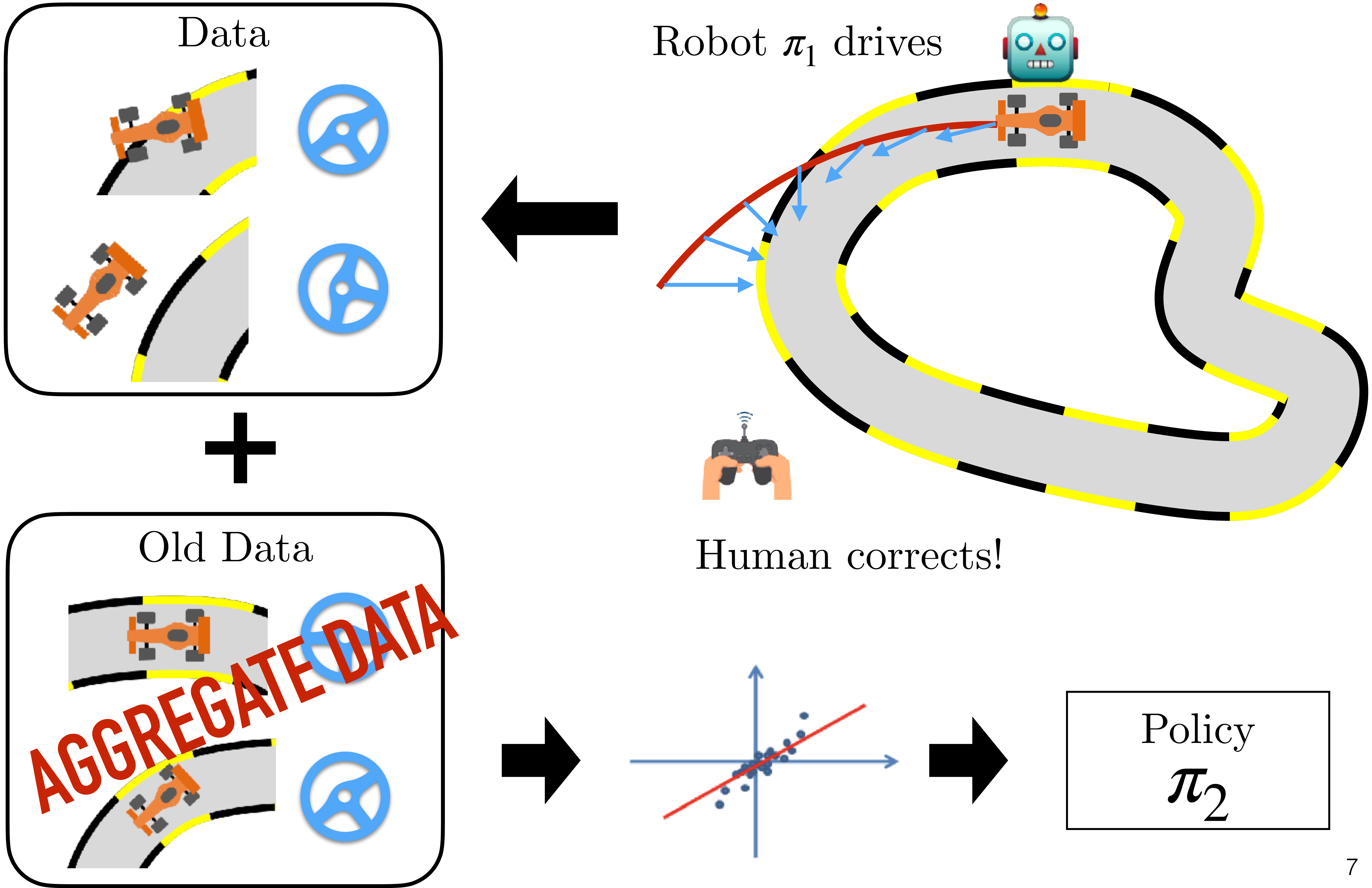
[Ross et al'11]

DAgger: Initializations



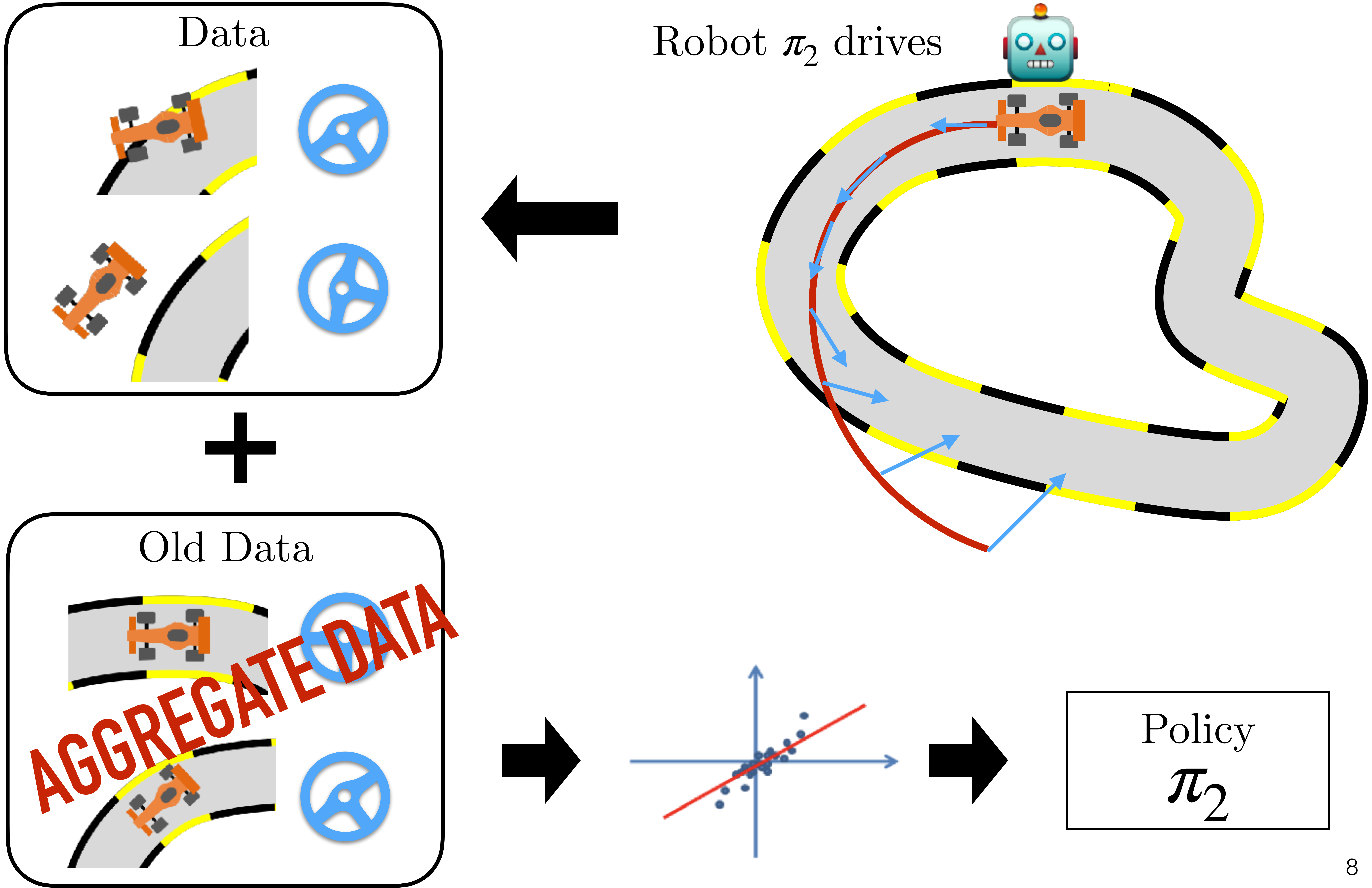
[Ross et al'11]

DAgger: Iteration 1



[Ross et al'11]

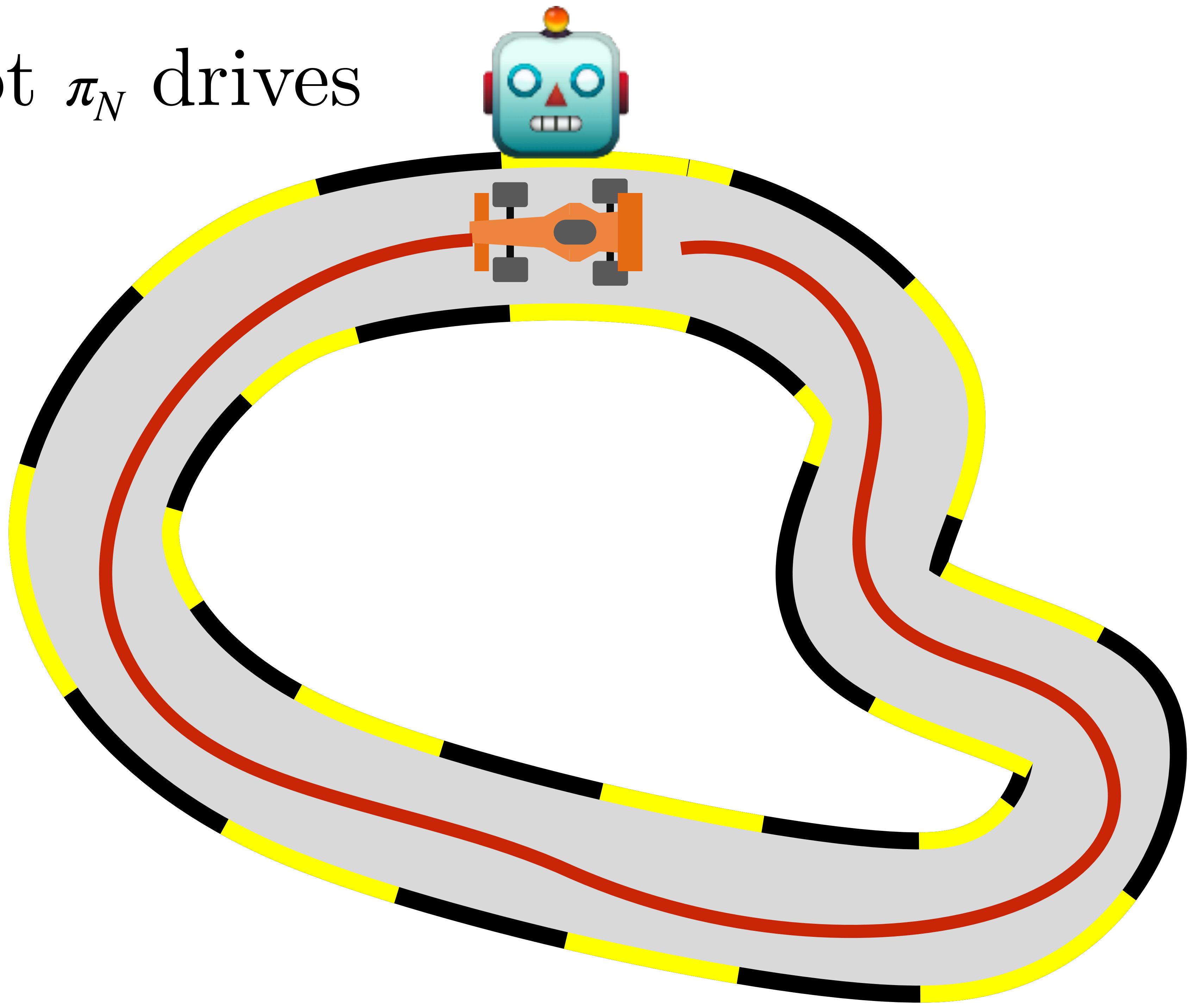
DAgger: Iteration 2



DAgger: Iteration N

[Ross et al'11]

Robot π_N drives



After many iterations ....
we are able to drive like a human!

DAgger (Dataset Aggregation)

Initialize with a random policy π_1 # Can be BC

Initialize empty data buffer $\mathcal{D} \leftarrow \{\}$

For $i = 1, \ldots, N$

Execute policy π_i in the real world and collect data

$$\mathcal{D}_i = \{s_0, a_0, s_1, a_1, \ldots\}$$ # Also called a rollout

Query the expert for the optimal action on learner states

$$\mathcal{D}_i = \{s_0, \pi^\star(s_0), s_1, \pi^\star(s_1), \ldots\}$$

Aggregate data $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

Train a new learner on this dataset $\pi_{i+1} \leftarrow \text{Train}(\mathcal{D})$

Select the best policy in $\pi_{1:N+1}$

The DAGGER Guarantee

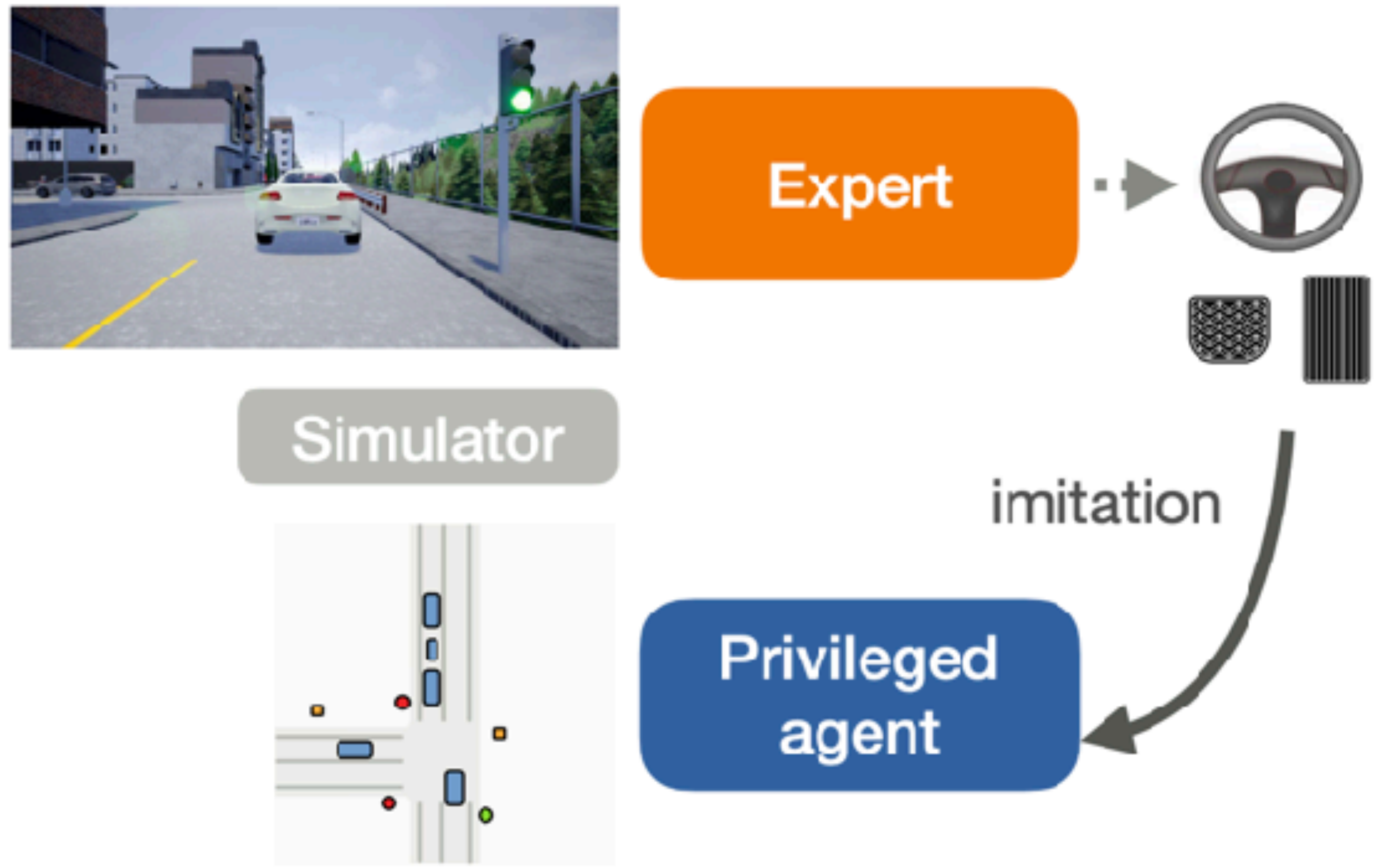
DAGGER returns a policy π such that

$$J(\pi) - J(\pi^*) \leq O(\epsilon H T)$$

H is the recoverability coefficient that says if I make a mistake, how much does an expert have to pay to recover

Many cool applications of DAGGER in robotics

Lee et al, Learning quadrupedal locomotion over challenging terrain (2020)



Chen et al Learning by Cheating(2020)

Choudhury et al, Data Driven Planning via Imitation Learning (2018)

Pan et al Imitation learning for agile autonomous driving (2019)

How do we actually apply DAGGER in practice?

Asking a *human* expert to label every state the robot visits is hard

Option 1: Extend DAGGER to different degrees of human feedback

Can we extend DAGGER to handle easier forms of human feedback preferences, interventions, etc?

Yes (*Future lectures!)

Option 2: Use an algorithmic oracle

What if we had a powerful algorithm
that we can run in train time
but not at test time?

With audio

Learning quadrupedal locomotion over challenging terrain

Joonho Lee[1], Jemin Hwangbo[1,2 †], Lorenz Wellhausen[1], Vladlen Koltun[3], Marco Hutter[1]

[1] Robotic Systems Lab, ETH Zurich
[2] Robotics & Artificial Intelligence Lab, KAIST
[3] Intelligent Systems Lab, Intel

†Substantial part of the work was carried out during his stay at 1

ETH zürich

RSL Robotic Systems Lab

intel Intelligent Systems Lab

But why does *aggregating* data work?

From Imitation Learning to Interactive No-Regret Learning

Interactive Learning

Learner

Adversary

$\min_{\pi} l(\pi)$

Choose $l(\pi)$

Interactive Learning

Learner		Adversary
Initialize policy	π_1 [policy] →	
	← $l_1(\,.\,)$ [loss]	Chooses loss
Update policy	π_2 →	
	← $l_2(\,.\,)$	Chooses loss
⋮		⋮

What is the best that I can do in such an adversarial setting?

From Imitation Learning to Interactive No-Regret Learning

How do we design algorithms that are no-regret?

$$\text{Regret} = \sum_{t=1}^{T} l_t(\pi_t) - \min_{\pi^*} \sum_{t=1}^{t} l_t(\pi^*)$$

(Learner) (Best in hindsight)

At every round t, choose the best policy in hindsight

$$\pi_t = \arg\min_{\pi} \sum_{i=1}^{t-1} l_i(\pi)$$

(lowest total loss)

$\sum l_t$ l_1

Policy 1

- -

1.0

Policy 2

- -

0.2

Policy 3

- -

0.5

Avg. Regret: - -

$\sum l_t$		l_1	l_2
1.0	Policy 1 	1.0	0.5
0.2	Policy 2 	0.2	0.5
0.5	Policy 3 	0.5	0.2

Avg. Regret: 0.80

$\sum l_t$		l_1	l_2	l_3
1.5	Policy 1 	1.0	0.5	0.5
0.7	Policy 2 	0.2	0.5	1.0
0.7	Policy 3 	0.5	0.2	0.2

Avg. Regret: 0.40

$\sum l_t$		l_1	l_2	l_3	l_4
2.0	Policy 1 	1.0	0.5	0.5	1.0
1.7	Policy 2 	0.2	0.5	1.0	0.2
0.9	Policy 3 	0.5	0.2	0.2	0.5

Avg. Regret: 0.53

$\sum l_t$		l_1	l_2	l_3	l_4	l_5
3.0	Policy 1 	1.0	0.5	0.5	1.0	0.5
1.9	Policy 2 	0.2	0.5	1.0	0.2	1.0
1.4	Policy 3 	0.5	0.2	0.2	0.5	0.2

Avg. Regret: 0.40

$\sum l_t$		l_1	l_2	l_3	l_4	l_5	l_6
3.5	Policy 1 	1.0	0.5	0.5	1.0	0.5	1.0
2.9	Policy 2 	0.2	0.5	1.0	0.2	1.0	0.5
1.6	Policy 3 	0.5	0.2	0.2	0.5	0.2	0.2

Avg. Regret: 0.32

$\sum l_t$		l_1	l_2	l_3	l_4	l_5	l_6
4.5	Policy 1 	1.0	0.5	0.5	1.0	0.5	1.0
3.4	Policy 2 	0.2	0.5	1.0	0.2	1.0	0.5
1.8	Policy 3 	0.5	0.2	0.2	0.5	0.2	0.2

Avg. Regret: 0.26

Is FTL no-regret?

FTL is no-regret if

1. We are in the continuous setting

2. Loss is strongly convex

Back to the proof!

Let's recap!

We can frame interactive imitation learning as online learning

FTL is no-regret if the loss is strongly convex

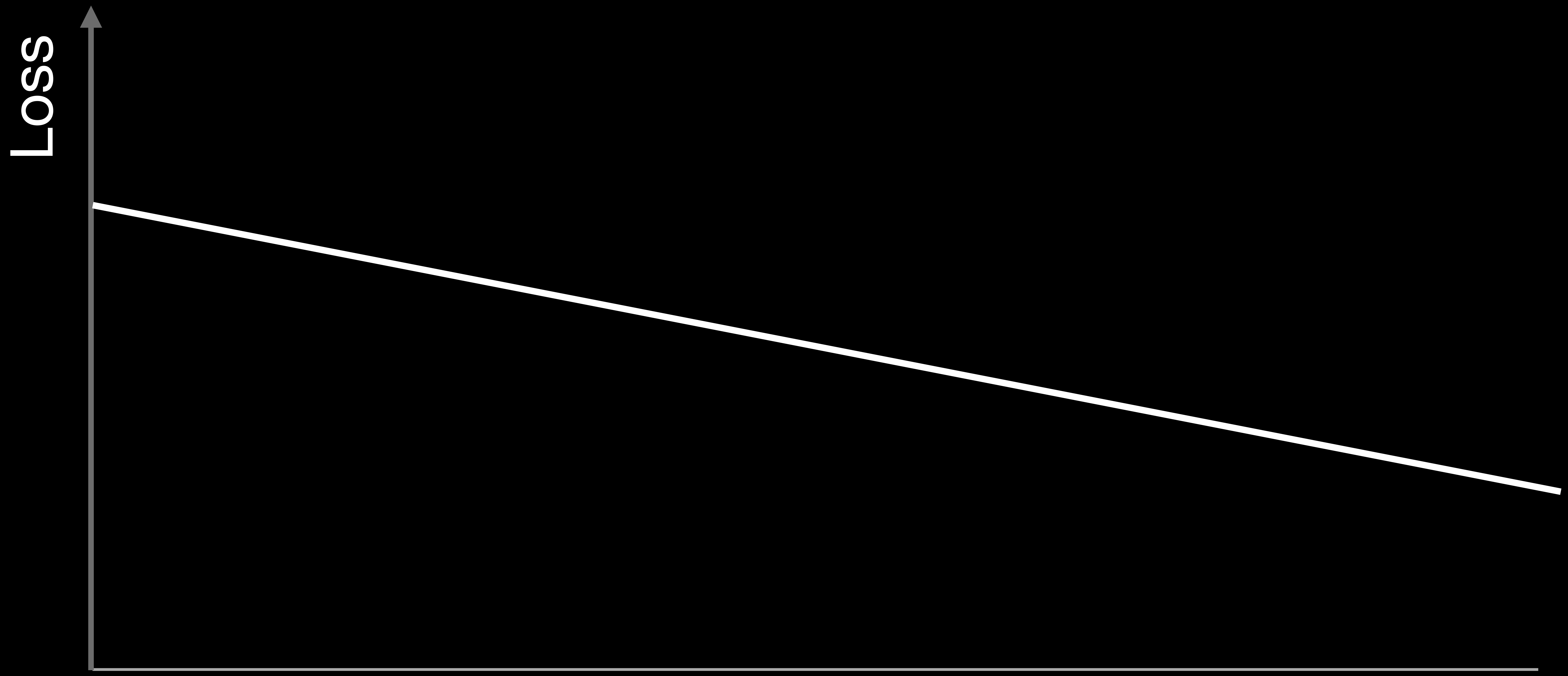
DAGGER is FTL

No-regret implies $O(\epsilon HT)$

The rabbit hole of online learning

When does FTL break?

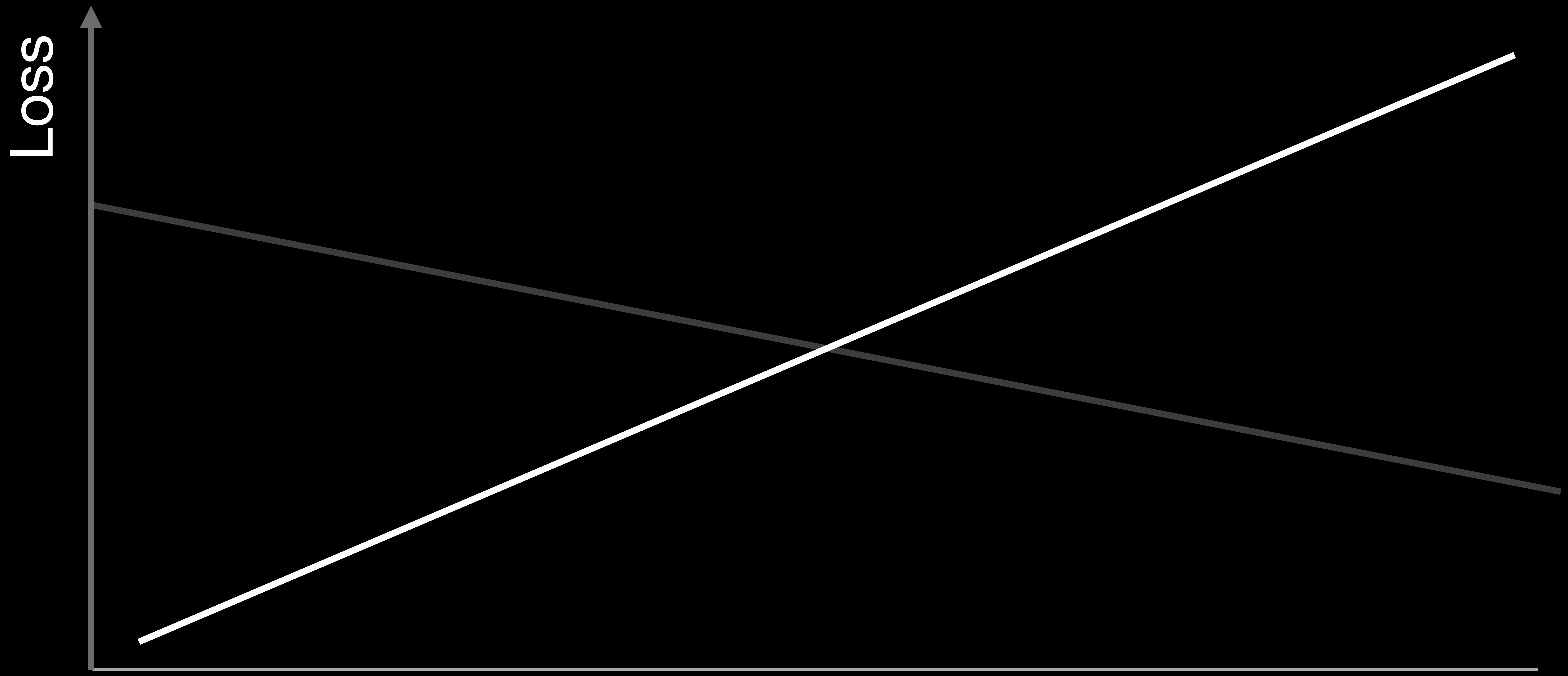
Loss = 0.75 Avg. Regret = 0.5



Choose π^1

Choose π^2

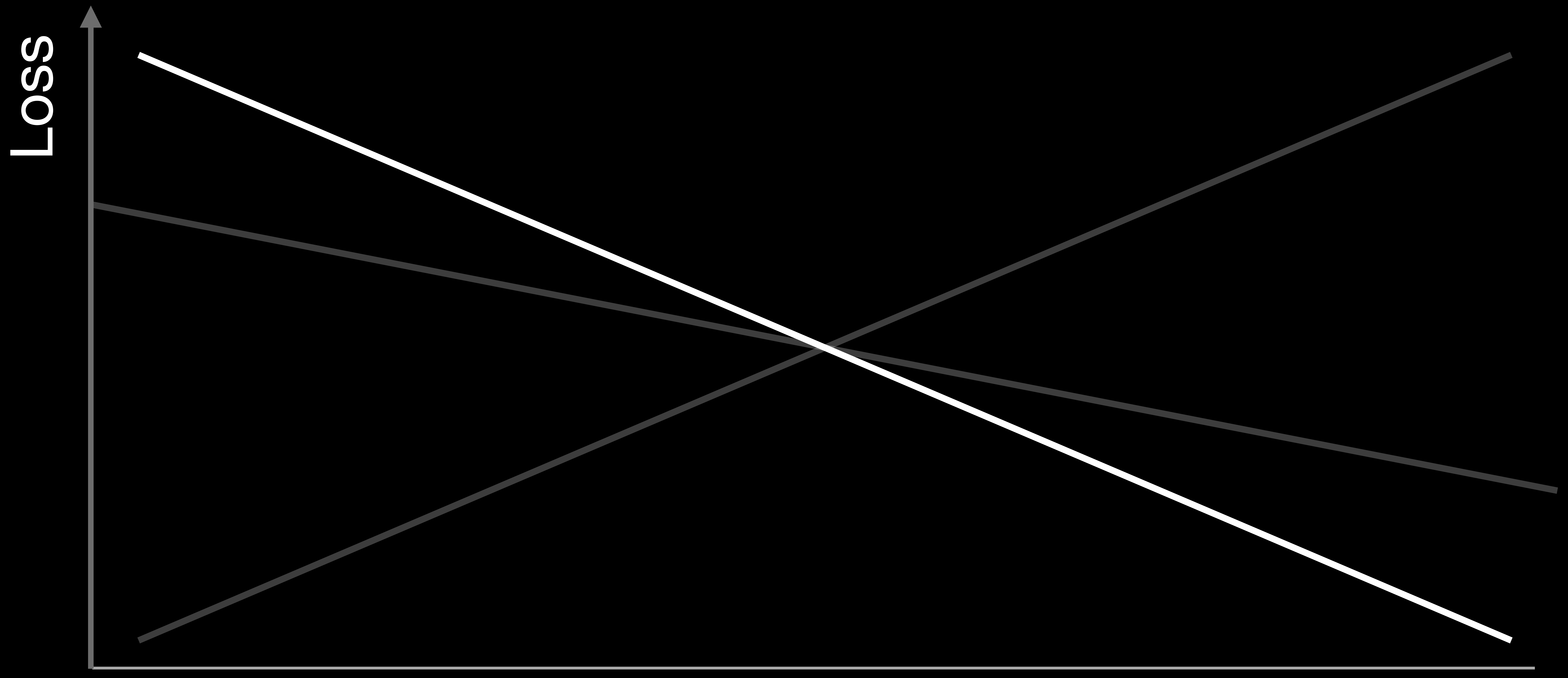
Loss = 1.0 Avg. Regret = 0.5



Choose π^1

Choose π^2

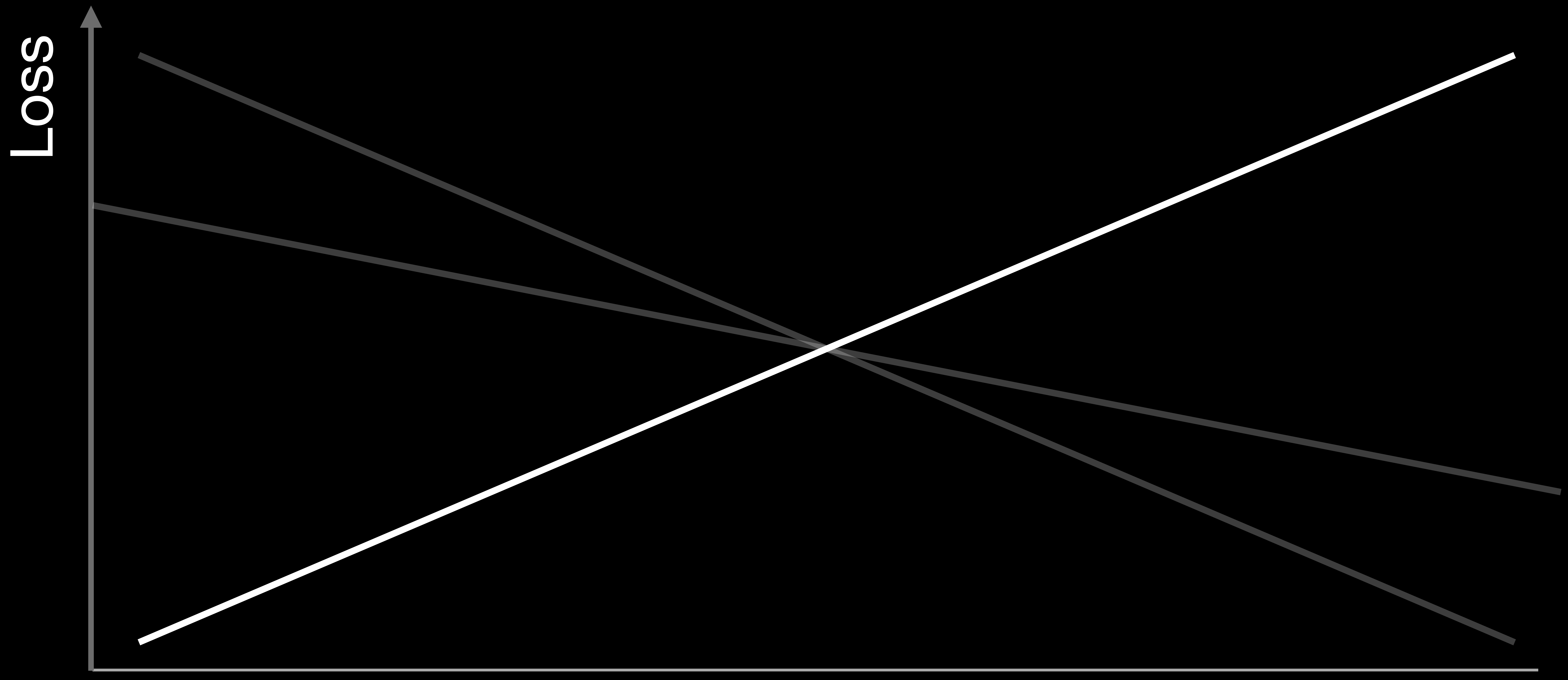
Loss = 1.0 Avg. Regret = 0.5



Choose π^1

Choose π^2

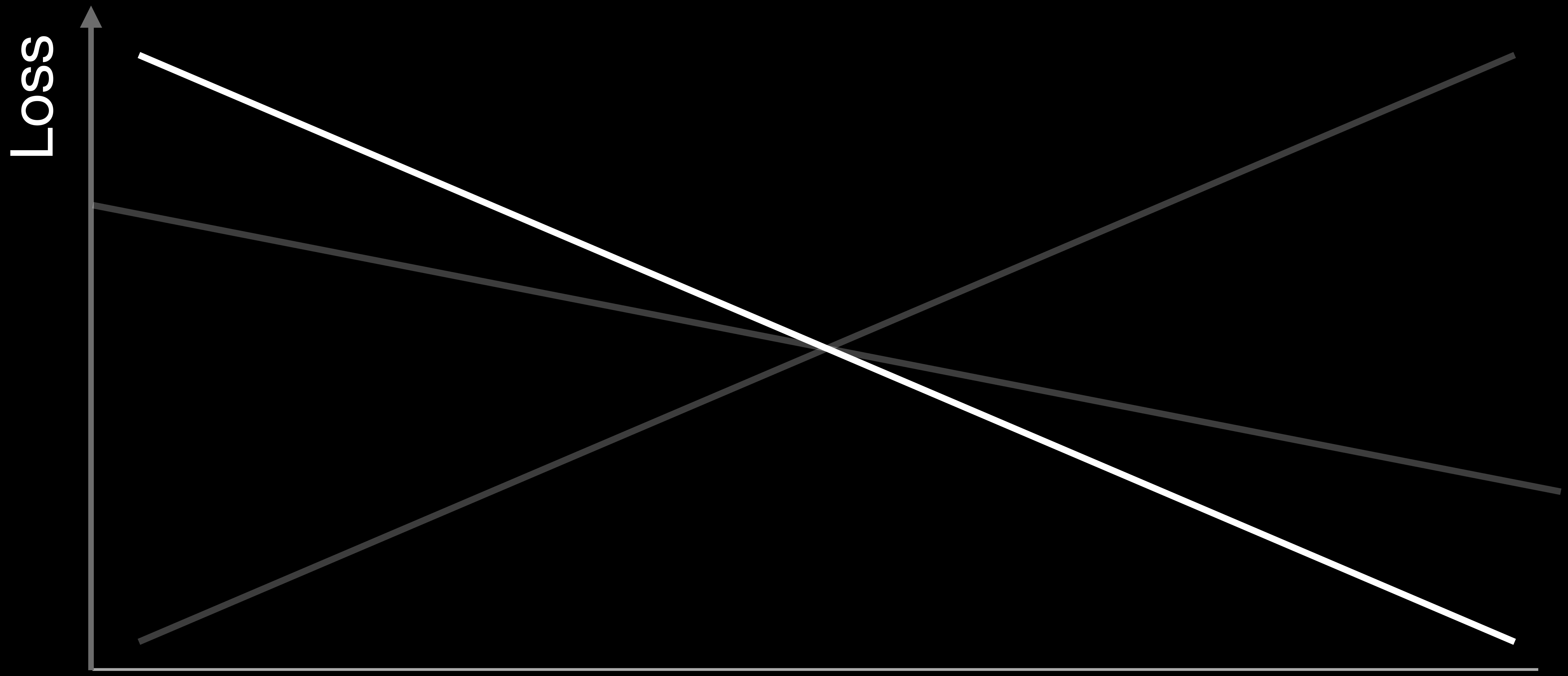
Loss = 1.0 Avg. Regret = 0.5



Choose π^1

Choose π^2

Loss = 1.0 Avg. Regret = 0.5



Choose π^1

Choose π^2

Be stable

Slowly change predictions

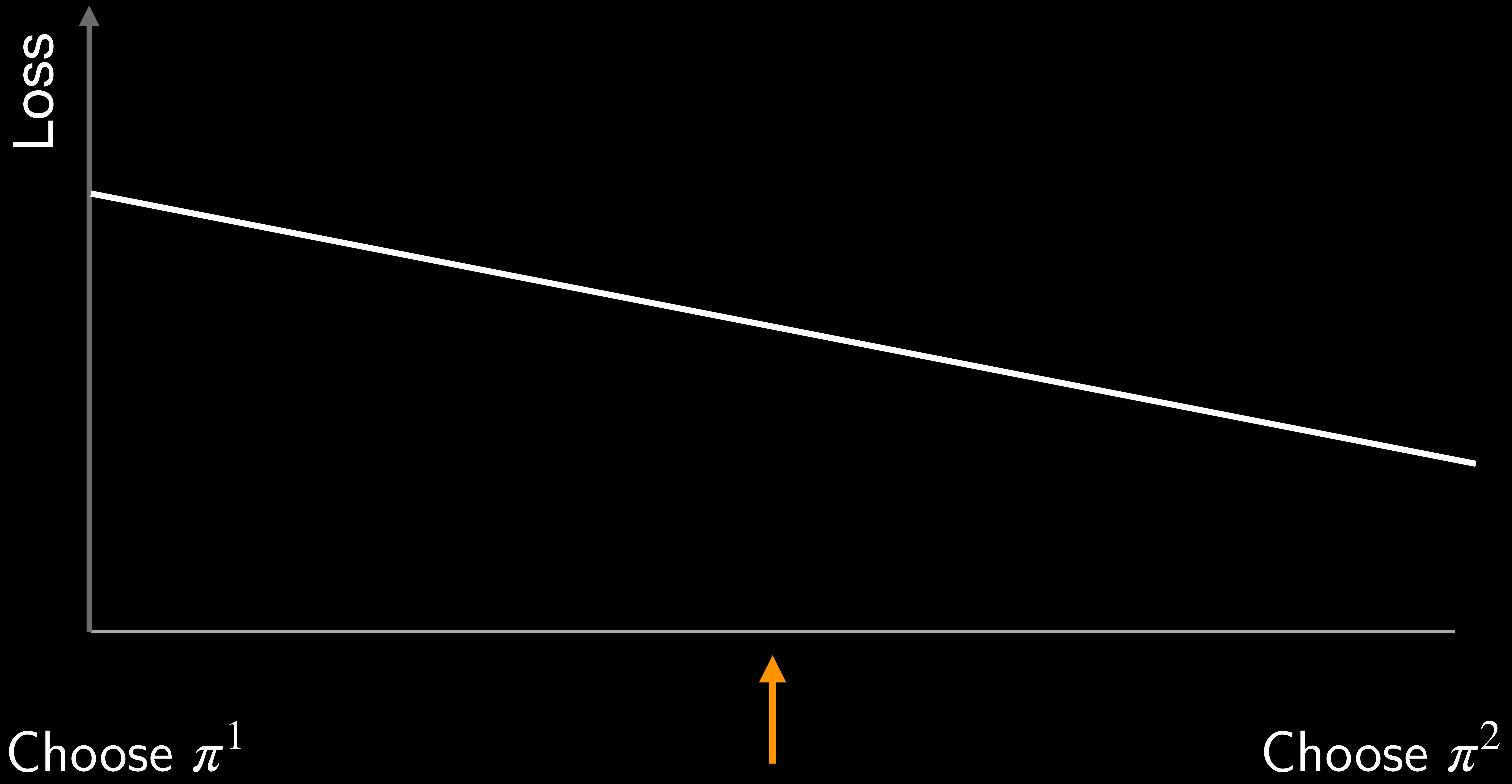
Achieve no-regret

Follow the Regularized Leader

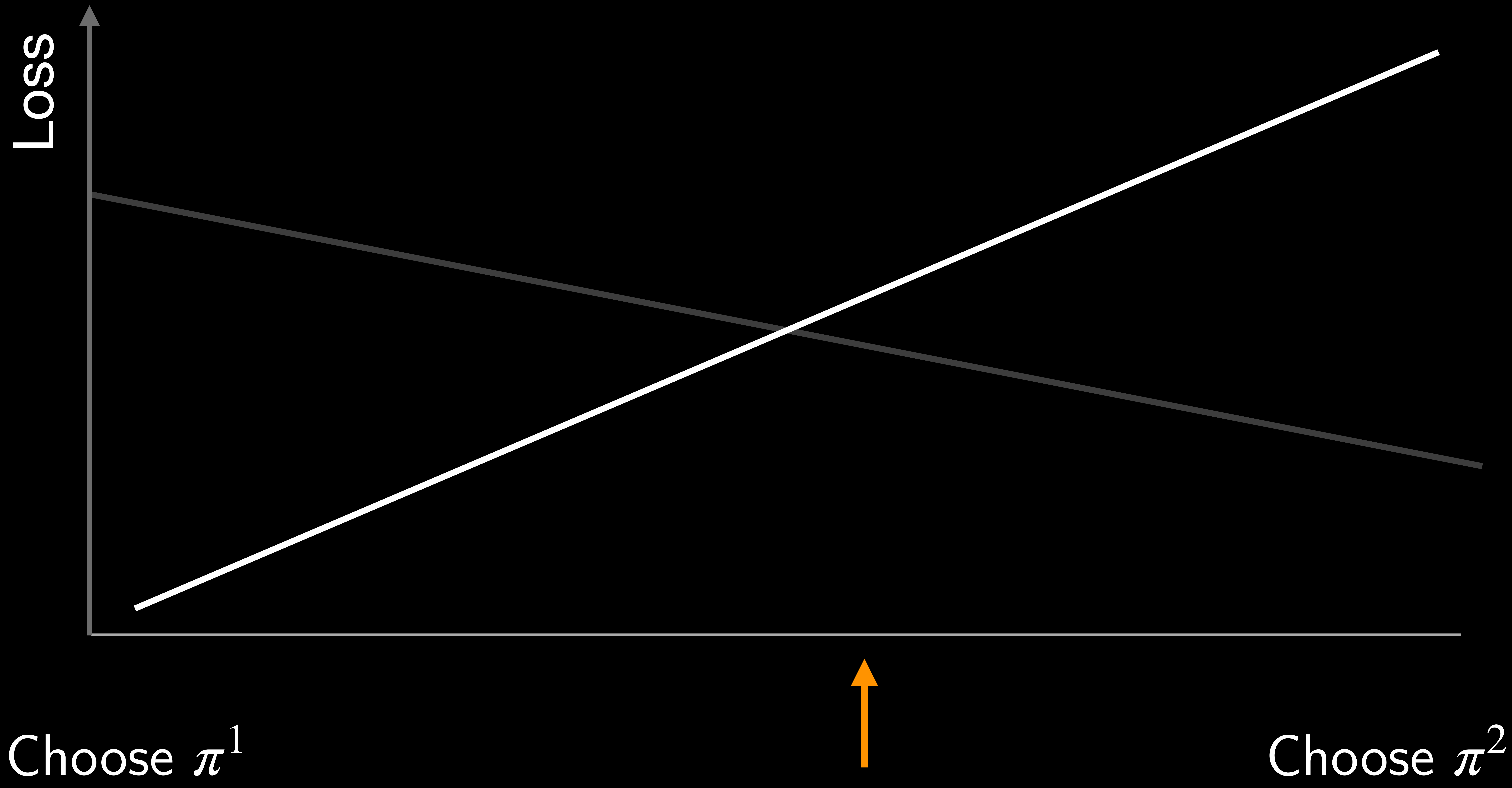
$$\pi_t = \arg\min_{\pi} \sum_{i=1}^{t-1} l_i(\pi) \quad + \eta_t R(\pi)$$

Strong regularization!

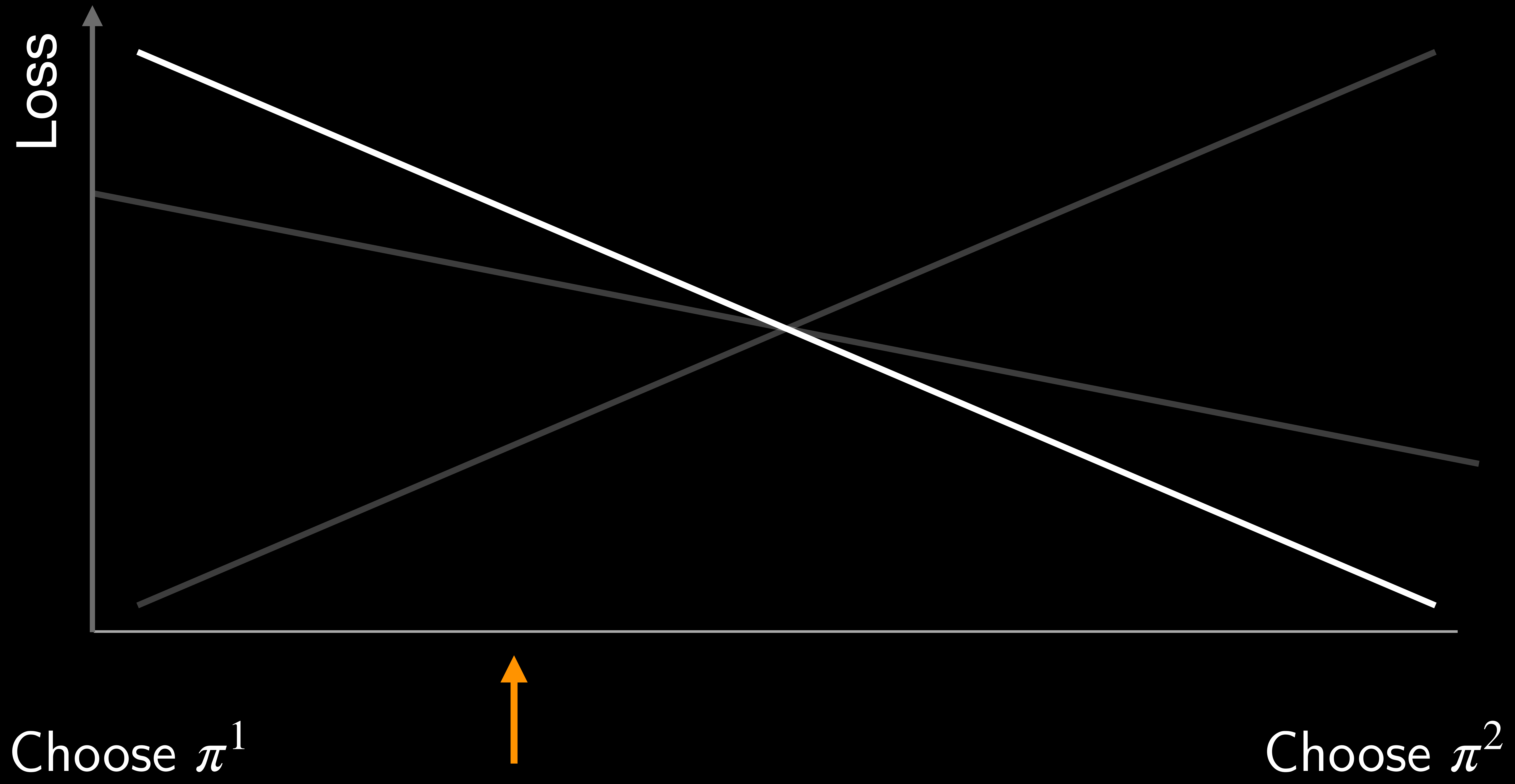
Loss = 0.5 Avg. Regret = 0.25



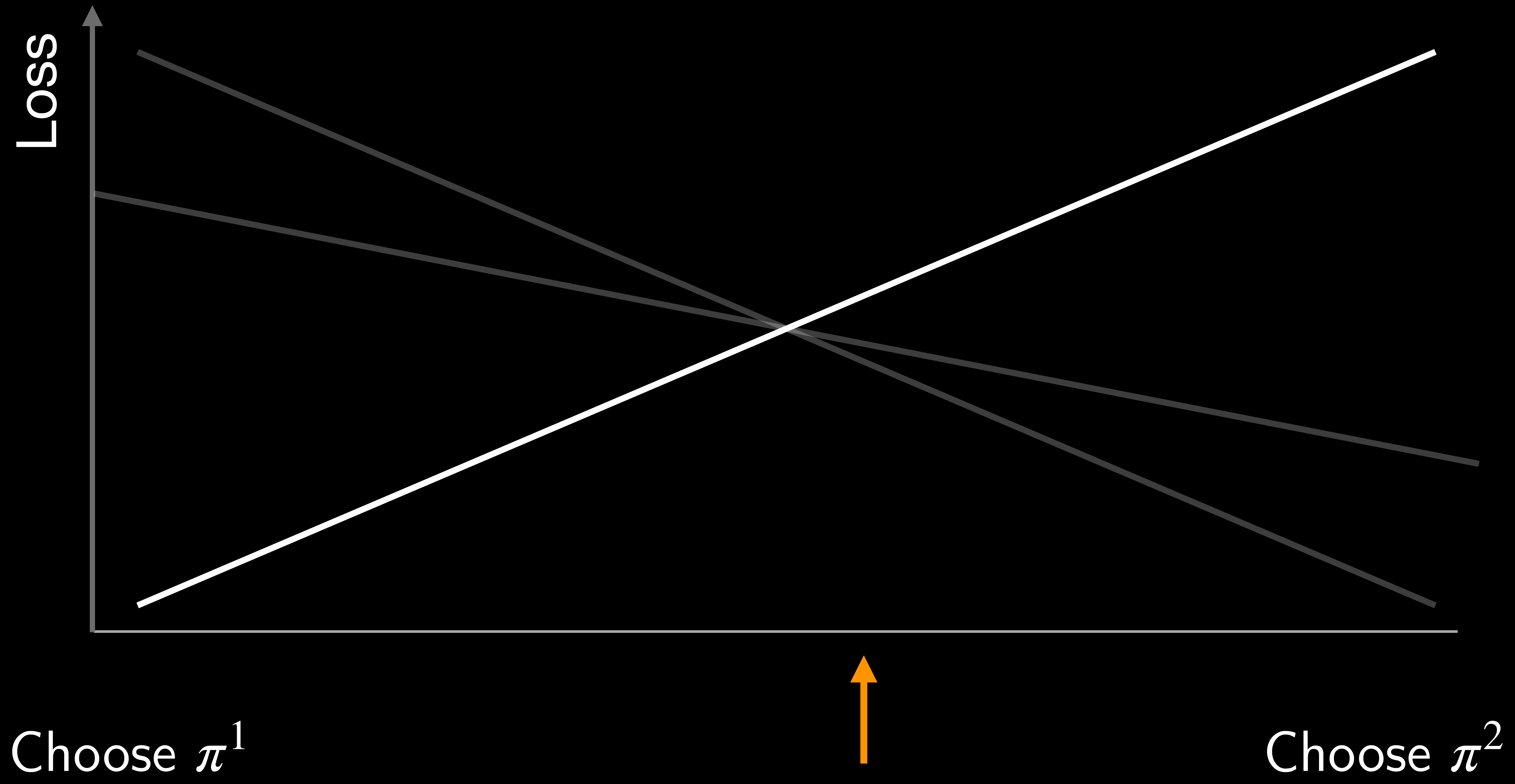
Loss = 0.6 Avg. Regret = 0.17



Loss = 0.78 Avg. Regret = 0.21



Loss = 0.6 Avg. Regret = 0.18



Loss = 0.78 Avg. Regret = 0.2