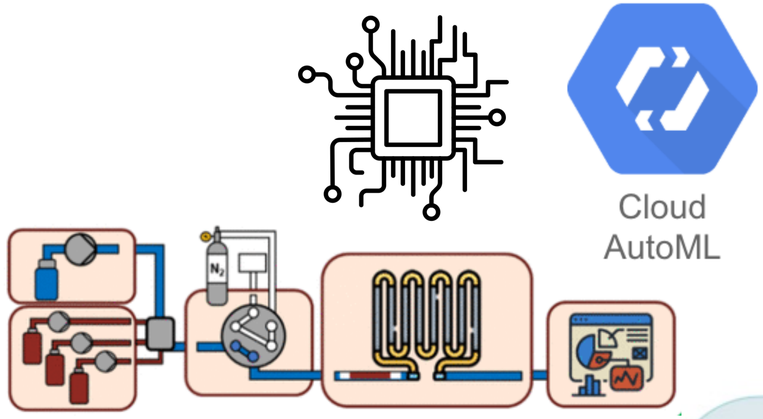
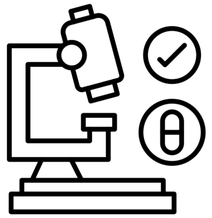
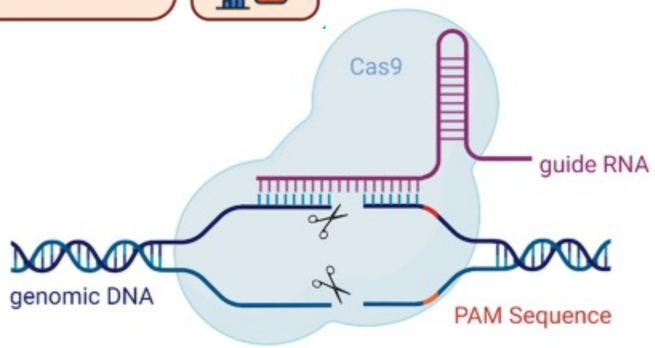
# Learning to Optimize in Structured Domains

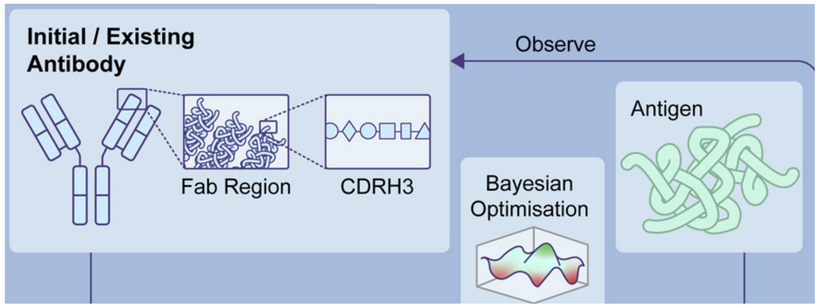
Parnian Kassraie
ETH Zurich

# Learning-based optimization



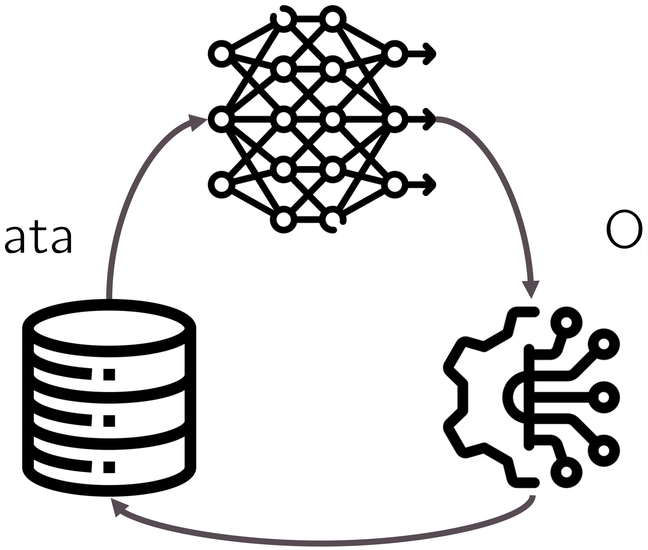





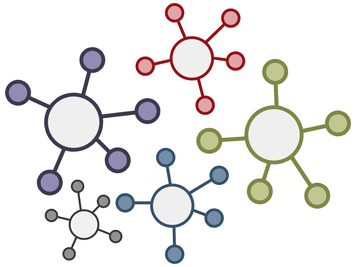

Learning the objective

Collecting data



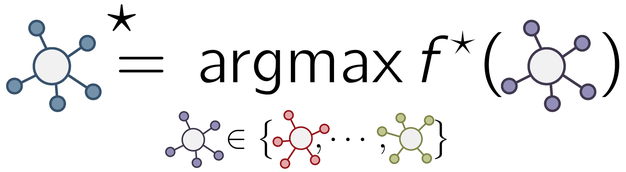
Optimized Solution

# Example: Drug Discovery



a galaxy of small molecules
(*action* domain)

binding to a protein
(*reward* function)




find good molecules
(*bandit* problem)

At every step t

Choose a molecule

$_t$

Evaluate it

$y_t = f^\star(\quad_t) + \varepsilon_t$

Goal: solve it using the fewest samples.

estimate $f^\star$

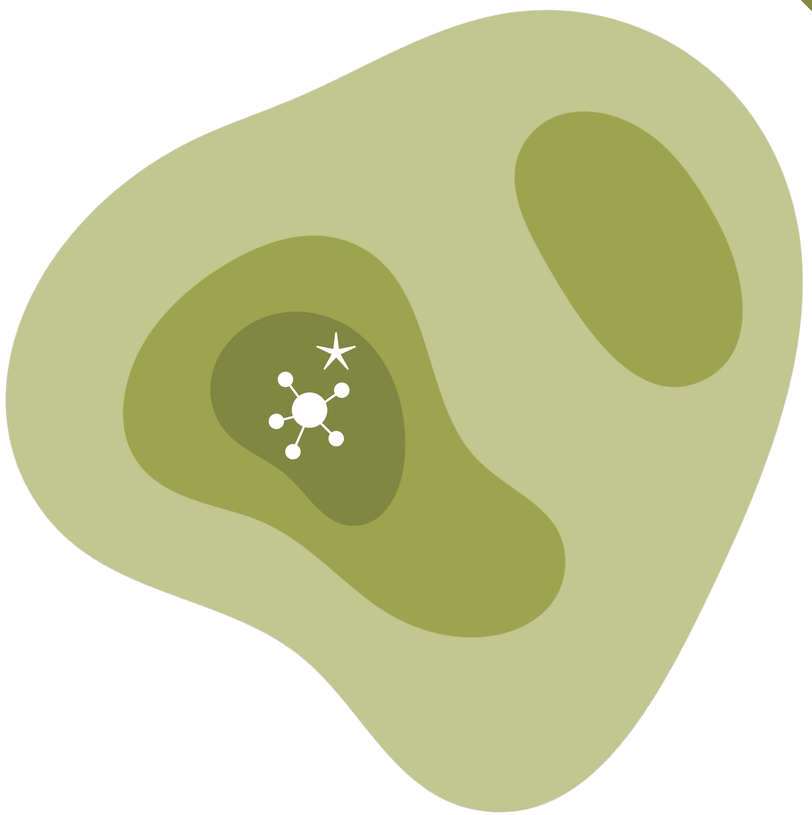
explore

maximize $f^\star$

exploit

Can we leverage the structure and symmetries of this domain of graphs?

# Sample Efficienty through structure



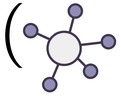
We narrow down the domain to high-reward areas.

Observing $f^\star($ $)$ gives information about its neighborhood

$\phi($ $)$ $\quad d($ $,$ $) < \Delta \quad |f^\star($ $) - f^\star($ $)| \leq \epsilon$

Knowing which **features** are relevant in calculation of the reward, yields a **notion of similarity**, helping us to find the solution.

A fine-grained notion of similarity accelerates the search.

$d_2($ $,$ $) \leq d_1($  $,$ $)$

- Uncover sparsity/low-dim manifold
- Respect Image, graph, or sequence structure

# Problem setting (Adaptive optimization)

The agent takes an action $x_t \in \mathcal{X}$ by looking at $\mathcal{H}_{t-1} = \{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$.

The environment responds with $y_t = f^\star(x_t) + \varepsilon_t$ sub-Gaussian $\mathcal{F}_{t-1}$measurable noise

$$f^\star(x) = \langle \phi^\star(x), \theta^\star \rangle \quad \theta^\star \in \ell_2(\mathbb{N})$$

linear or Hilbert-Schmidt operator

Goal: incur minimal expected regret

$$R(T) = \sum_{t=1}^{T} f(x^\star) - f(x_t)$$

When $\phi^\star$ is known, success criteria is any-time sub-linearity

$$\mathbb{P}\Big(\forall T \geq 1 : R(T) = o(T)\Big) \geq 1 - \delta \qquad \lim_{T \to \infty} \frac{R(T)}{T} \overset{\text{a.s.}}{=} 0$$

Many algorithm classes achieve this: e.g. optimistic or reduction-based.

We focus on $\phi^\star$ when is unknown.

# Learning the structure offline vs. online

With (plenty of) offline i.i.d. data we can learn the features well.

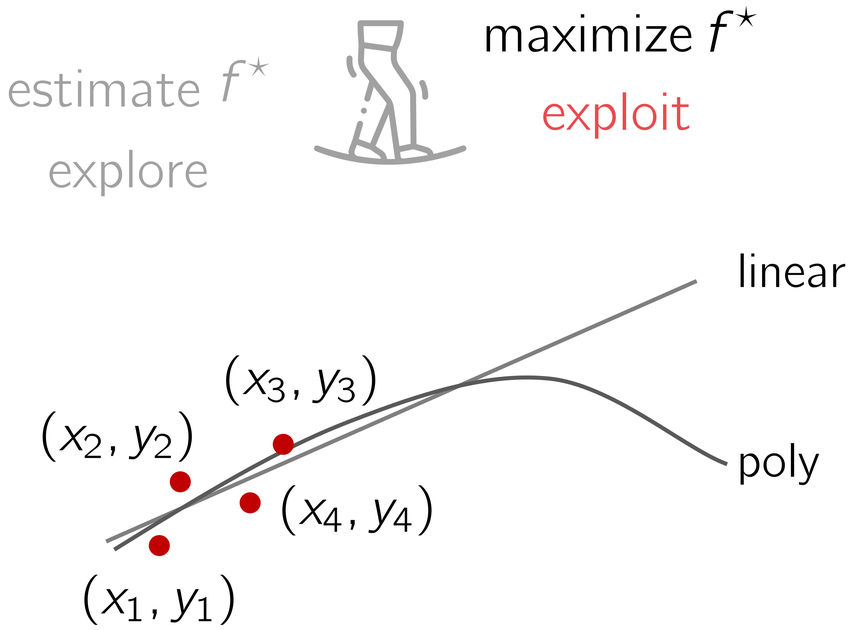
- Sparsity recovery and representation learning
- Manifold learning
- Adapting network architecture to data type

$$f^\star(x) = \langle \phi^\star(x), \theta^\star \rangle$$

This is more complicated in adaptive setting.

- Samples are non-i.i.d.
- The may not even be diverse.

We need to assign uncertainty to the model and adapt is sequentially as we observe new evidence.



# Research question

Can we design (online) optimization algorithms that uncover and adapt to the problem structure?

Linear Solutions

Online Model Selection

$$\phi^\star \in \{\phi_1, \dots, \phi_M\} \quad M \gg T$$

Non-Linear Solutions

Adaptive Representation Learning

$\phi^\star$ is learnable with a neural network

# Online Model Selection (Problem)

The reward is linearly parametrizable

$$f^\star(x) = \langle \phi^\star(x), \theta^\star \rangle \quad \theta^\star \in \mathbb{R}^d \quad \|\theta^\star\|_2 \leq B$$

$$\phi^\star \in \{\phi_1, \ldots, \phi_M\} \quad M \gg T$$

Maximize $f^\star$ without knowing $\phi^\star$

$$R(T) = \sum_{t=1}^{T} f(x^\star) - f(x_t)$$

- Sublinear in $T$
- $\log M$

# A basic solution: Explore then Commit

For $T_0$ steps, take i.i.d. samples following a uniform, or "diverse" distribution

$$\{(x_1, y_1), \ldots, (x_{T_0}, y_{T_0})\}$$

Incur high regret of $2BT_0$

Use Group Lasso for implicit model selection $\quad \Phi(x) = (\phi_1(x), \ldots, \phi_M(x)) \in \mathbb{R}^{dM}$

$$\hat{\theta} = \arg\min \frac{1}{T_0} \sum_{s \leq T_0} (y_s - \langle \Phi(x_s), \boldsymbol{\theta} \rangle)^2 + \lambda \sum_{j \leq M} \|\theta_j\|_2$$

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M) \in \mathbb{R}^{dM}$$

Relies on Lasso variable selection

For the remaining steps, always do greedy

$$x_t = \arg\max \langle \Phi(x), \hat{\theta} \rangle$$

Is not any-time: only works if horizon T is known in advance

Under a good choice of $T_0$ and $\lambda$ satisfies

$$R(T) = \mathcal{O}\left(\sqrt[3]{T^2 \log M}\right) \qquad \text{w.h.p.}$$

# Our solution: Probabilistic Aggregation of Models

Instead of commiting to a single model,

Randomly iterate over the models and at each step choose one

Instatiate M "agents"

Agent j uses $\phi_j$ to model the reward

Has is own action of choice $x_{t,j}$

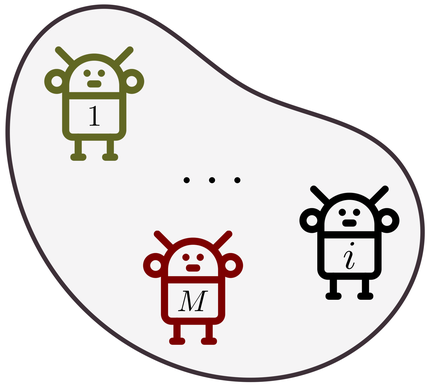
Maintain a distribution over agents

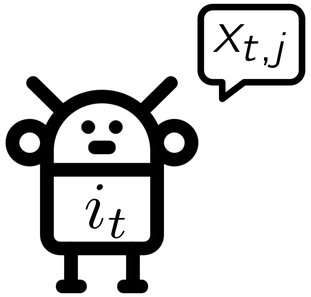
Reflects our belief on how good an agent is

$q_t \in \Delta_M$

M-dimensional probability simplex

$i_t \sim q_t$



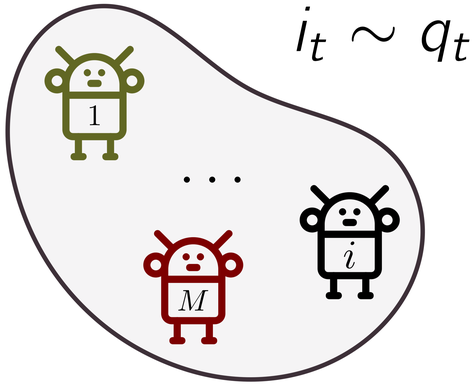

$y_t = f^\star(x_t) + \varepsilon_t$

Update $q_t$

Pass $(x_t, y_t)$ to all agents

# Method

Use Exponential Weighting to update the probability over agents




In the next iterates, increase probability of the agent, if it yielded high rewards

True reward obtained by agent j so far

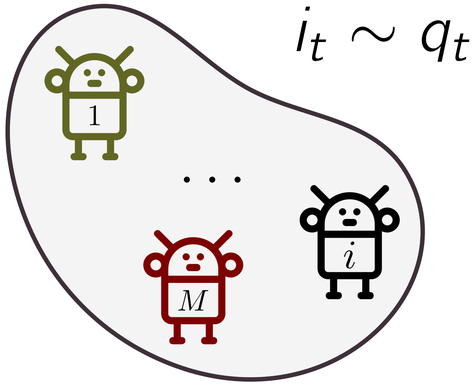
$$q_{t,j} = \frac{\exp\left(\eta_t \sum_{s=1}^{t-1} f^\star(x_{s,j})\right)}{\sum_{i=1}^{M} \exp\left(\eta_t \sum_{s=1}^{t-1} f^\star(x_{s,i})\right)}$$

sensitivity of updates

But reward of unplayed agents is not observed.

# Method

Use Exponential Weighting to update the probability over agents




In the next iterates, increase probability of the agent, if it could have yielded high rewards

Estimate of the reward obtained by agent j so far

$$q_{t,j} = \frac{\exp\left(\eta_t \sum_{s=1}^{t-1} \hat{f}(x_{s,j})\right)}{\sum_{i=1}^{M} \exp\left(\eta_t \sum_{s=1}^{t-1} \hat{f}(x_{s,i})\right)}$$

Reward of unplayed agents is not observed? Hallucinate it.

Turn group lasso into a sparse online regression oracle

$$\hat{\theta}_t = \arg\min \frac{1}{t} \sum_{s \le t} (y_s - \langle \Phi(x_s), \boldsymbol{\theta} \rangle)^2 + \lambda_t \sum_{j \le M} \|\theta_j\|_2$$

$$\hat{f}(x) = \langle \Phi(x), \hat{\theta}_t \rangle$$

Cost of going time uniform is loglog d!

**Theorem (Anytime Lasso Oracle)**

*For appropriate choice of* $(\lambda_t)_{t \ge 1}$,

$$\mathbb{P}\left(\forall t \ge 1 : \left\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t\right\|_2 \lessapprox \sqrt{\frac{\log(M/\delta)}{t}}\right) \ge 1 - \delta$$

# Result: ALEXP

Anytime **Exp**onential weighting algorithm with **L**asso reward estimates

**Algorithm 1** ALEXP

**for** $t \geq 1$ **do**

Draw $i_t \sim q_t$ and let it choose the action $x_t$

Observe $y_t = f^\star(x_t) + \varepsilon_t$.

Pass $\{(\mathbf{x}_t, y_t)\}$ to all agents $j = 1, \ldots, M$.

Hallucinate the expected reward of every agent.

Update selection probabilities

$$q_{t+1,j} \leftarrow \frac{\exp(\eta_t \sum_{s=1}^{t} \hat{f}_{s,j})}{\sum_{i=1}^{M} \exp(\eta_t \sum_{s=1}^{t} \hat{f}_{s,i})}$$

**end for**

* Algorithm has been simplified.

**Theorem (Online Model Selection)**

*For appropriate choices of parameters, ALEXP with UCB agents satisfies*

$$R(T) = \mathcal{O}\left(\sqrt{T \log^3 M} + T^{3/4}\sqrt{\log M}\right)$$

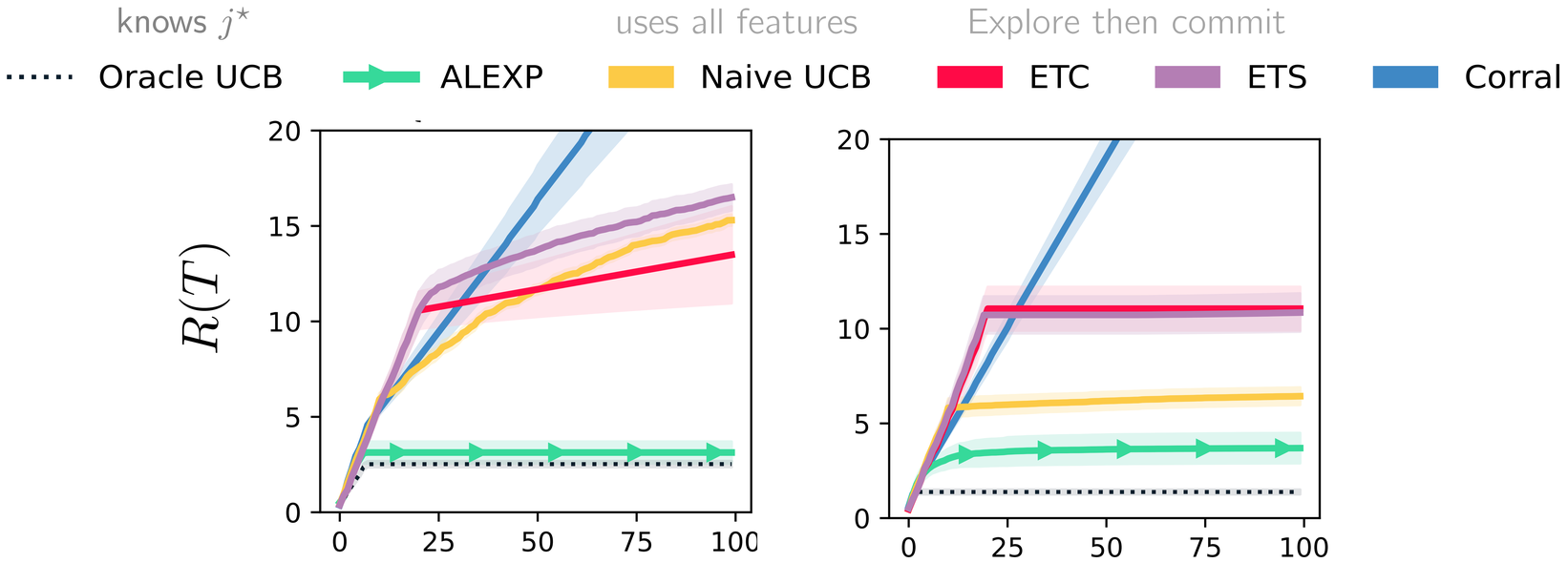
*w.h.p. simultaneously for all* $T \geq 1$.

[Solves the open problem of Agarwal et al. 2017 in the Linear case]

# Performance and intuition

Each feature map consists of d polynomials chosen out of degree-p Legendre basis $M = \binom{p+1}{d}$

$\theta^\star$ is drawn uniformly at random from the unit sphere




$M = 55$

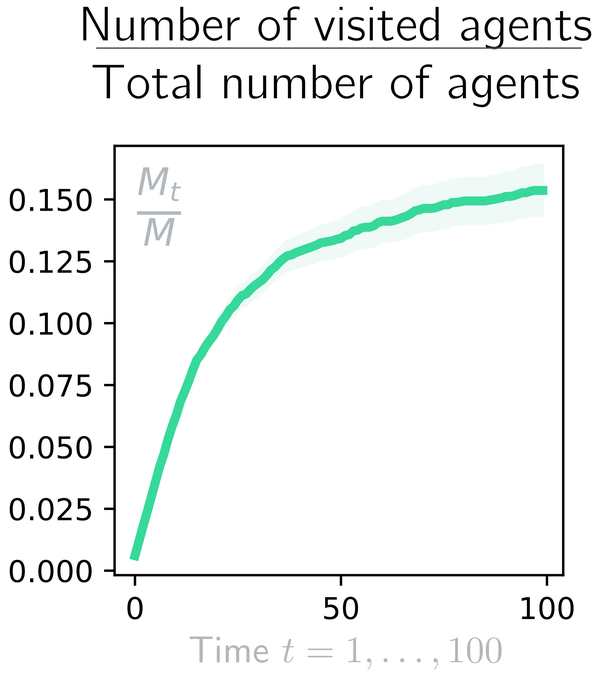
Correlated Models

$p = 10, d = 8$

$M = 165$

Many Models

$p = 10, d = 3$




Rapidly recognizes top agents and whp selects among them

# Outline

Intro

What is learning-based optimization?
Why structure-aware?

Linear Solutions

Online Model Selection

- PK et al. ICML '22
- PK et al. NeurIPS '23
- FS*, PK* et al. UAI '23

Non-Linear Solutions

Adaptive Representation Learning

- PK, AK AISTATS '22
- PK et al. NeurIPS '22

# Adaptive Representation Learning with NNs

The linear model class, no matter how large M is, can not characterize complex rewards.

Can we parametrize the feature map and learn it from data using an MLP? $f_{\mathrm{NN}}(x) = \langle \phi_{\mathrm{NN}}(x), \theta \rangle$

At step t using the data $\{(x_1, y_1), \ldots, (x_t, y_t)\}$

Network width

$$\mathcal{L}_t(\theta) = \frac{1}{t} \sum_{s \leq t} \left( f_{\mathrm{NN}}(x_s; \theta) - y_s \right)^2 + m\lambda \|\theta - \theta_0\|_2^2$$

L-layer MLP

Parameters at initialization

Run SGD for long enough to obtain $f_{\mathrm{NN}}(x; \theta_t)$.

? Is $f_{\mathrm{NN}}(x; \theta_t)$ a good estimate of $f^\star$?
It is trained on non-i.i.d data.

? How do we use $f_{\mathrm{NN}}(x; \theta_t)$ for action selection?
Greedy can fail.

$\mathrm{argmax}\, f_{\mathrm{NN}}(x; \theta_t)$

# Neural any-time confidence sequence

The trained network may not be aligned with the reward. However,

**Theorem (Anytime Neural Confidence Sets)**

*Under a mild regularity assumption on $f^\star$, if the learning rate is small enough and the network is wide enough, with probability greater than $1-\delta$*

$$\forall t \geq 1,\ x \in \mathcal{X}: \quad \left| f^*(x) - f_{NN}(x;\theta_t) \right| \leq \beta_t \sigma_t(x)$$

uncertainty of the reward estimates

$$\sigma_t(x) = g_t^\top (m\lambda I + g_t g_t^\top)^{-1} g_t$$

$$g_t = \sum_{s \leq t} \nabla_\theta f_{\mathrm{NN}}(x_s;\theta_s)$$

The theorem gives an upper confidence bound (UCB) on the true reward values.

We take an "optimistic" strategy for selectinng actions:

$$x_{t+1} = \operatorname{argmax} f_{\mathrm{NN}}(x;\theta_t) + \beta_t \sigma_t(x)$$

# Neural UCB Algorithms

Putting everything together:

**Algorithm 2** NN-UCB

**Input:** Initialization $\theta_0$, GD learning rate
**for** $t \geq 1$ **do**
  Play $x_t = \arg\max f_{\text{NN}}(x; \theta_{t-1}) + \beta_t \sigma_{t-1}(x)$.
  Observe $y_t$ and calculate the loss on $(x_t, y_t)$.
  Update $\theta_t$ with a step of GD.
  Update $\sigma_t(x)$ using the gradient vector.
**end for**

In practice: data is batched, we use ADAM, and a plug in a computationally light approximation of the uncertainty term.

**Theorem (Neural Bandit No Regret Guarantee)**

*Under the same assumptions and with probability greater than $1 - \delta$, NN-UCB satisfies*

$$\forall T \geq 1 : R(T) = \tilde{\mathcal{O}}\left(C(d, L) T^{1-1/2d}\right)$$

Marks the first no-regret guarantee for a neural bandit algorithm.

Prior work:

$$R(T) = \tilde{\mathcal{O}}\left(I(y_{1:T}; f_{1:T})\sqrt{T}\right)$$

[Zhou et al. ICML 20]
[Zhang et al. ICLR 21]
[Yang et al. arXiv 20]

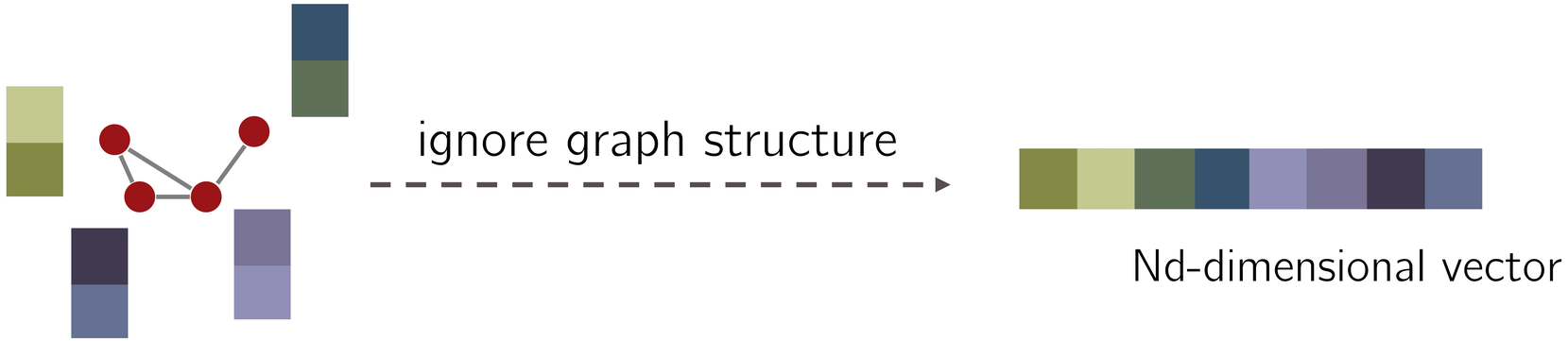
But we proved:

$$I(y_{1:T}; f_{1:T}) = \tilde{\mathcal{O}}(T^{1-1/d})$$

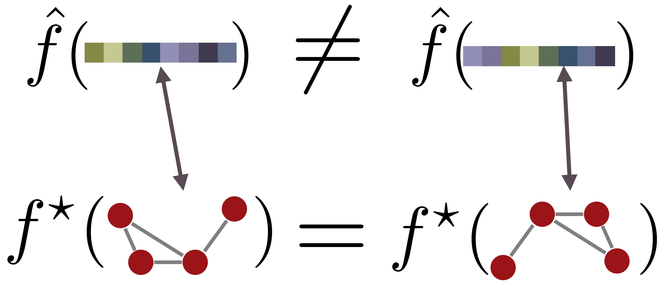
# Graph domain symmetries

We showed that neural networks are a great tool to sequentially learn important features of our actions. We can push this further if the action domain has more structure.

Domain:
N-node graphs
with d-dimensional
node features




Structure-agnostic reward estimator




Use a model invariant to node permutation.

A GNN with 1 conv layer meets this structure!

$$f_{\text{GNN}}(G;\theta) = f_{\text{GNN}}(\pi \cdot G;\theta)$$

Permutation operator

# Fast rates by leverging structure

We can construct confidence sets using GNNs, and create an optimistic algorithm: GNN-UCB

**Theorem (Fast Rates for Graphs)**

*Under regularity assumptions and with probability greater than $1-\delta$, GNN-UCB satisfies*

$$\forall T \geq 1 : R(T) = \tilde{\mathcal{O}}\left(C(d, L) \log N T^{1-1/2d}\right)$$

Naively using NN-UCB gives

$$R(T) = \tilde{\mathcal{O}}\left(C(d, L) T^{1-\frac{1}{2Nd}}\right)$$

T: the bandit horizon,
N: the number of nodes in each graph,
d: the dimension of node features

CNNs uncover important features in images much better than neural networks.

For 1D images, this is in part credited to invariance of the convolution operator to certain transforms (e.g. circular shifts).

**Theorem (Fast Rates for Images)**

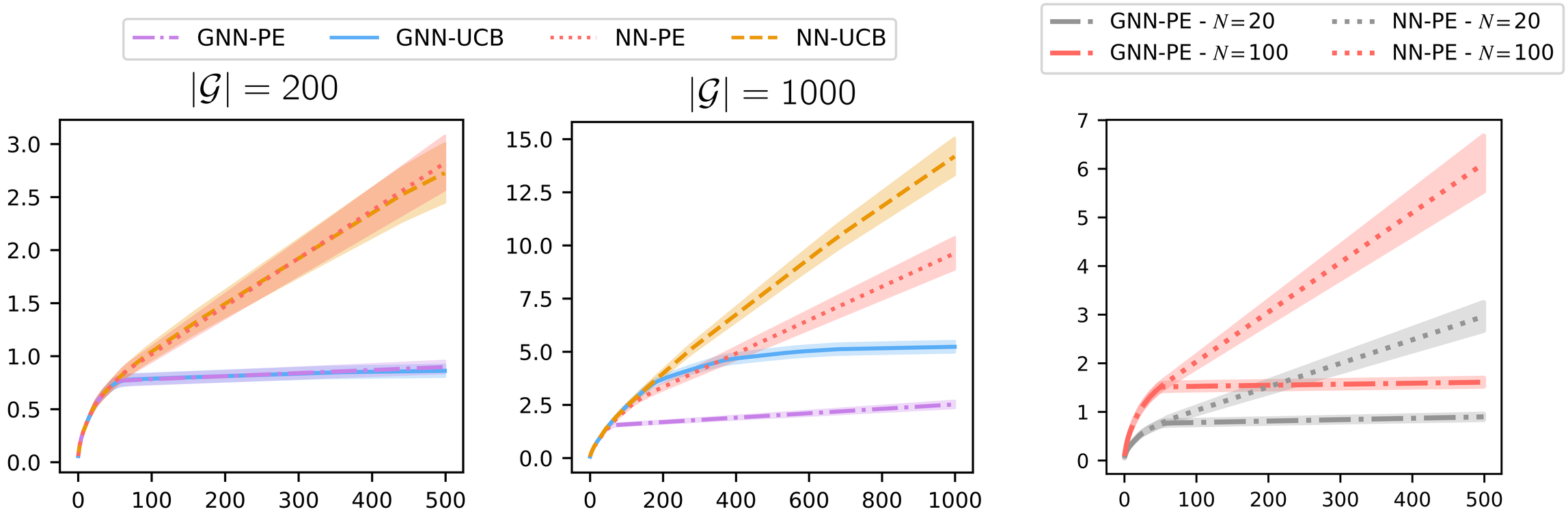
*Under regularity assumptions and with probability greater than $1-\delta$, CNN-UCB satisfies*

$$\forall T \geq 1 : R(T) = \tilde{\mathcal{O}}\left(C(d, 2)\left(\frac{T}{d}\right)^{1-1/2d}\right)$$

# Improved performance by leveraging structure

Domain $\mathcal{G}$: Erdos-Rényi random graphs
Objective: Sampled from GP$(0, k_{GNN})$

- Outperforms NN methods
- Scales well to large domains of graphs
- Scales well to domains of large graphs



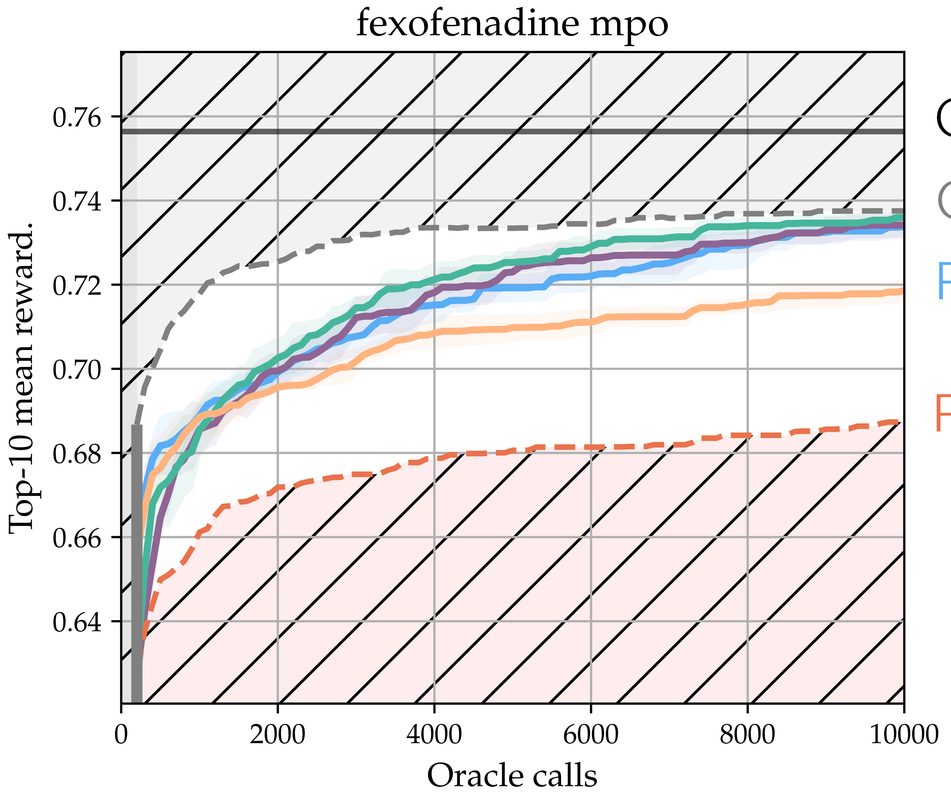
# Application: Virtual Screening for Drug Discovery

Virtual screening serves as the first step in DD pipelines.

Domain: Zinc dataset with 250k small organic molecules

Rewards: 23 objectives on qantitative estimates of drug-likeness

$$G_t = \text{argmax} f_{\text{GNN}}(G; \theta_{t-1}) + \beta_t \sigma_{t-1}(G)$$




Draw 500 graphs from Zinc

Oracle

Computationally Limited Oracle

Practical Variants of GNN-UCB

Pure exploration



[MW, BS, PK, AK, IB, AI4D 2024]

# Outlook

With language models reaching human capacity, learning-based optimization becomes integral to the LLM training stack.

Learning-based Data-collection

Active Reward modeling

Personalization (contextual bandits)

# Thank you!



All icons from flaticon.com, colors adapted by ChatGPT.