Efficient Contextual Bandits with Continuous Actions

Maryam Majzoubi
New York University

Chicheng Zhang
University of Arizona

Rajan Chari
Microsoft Research

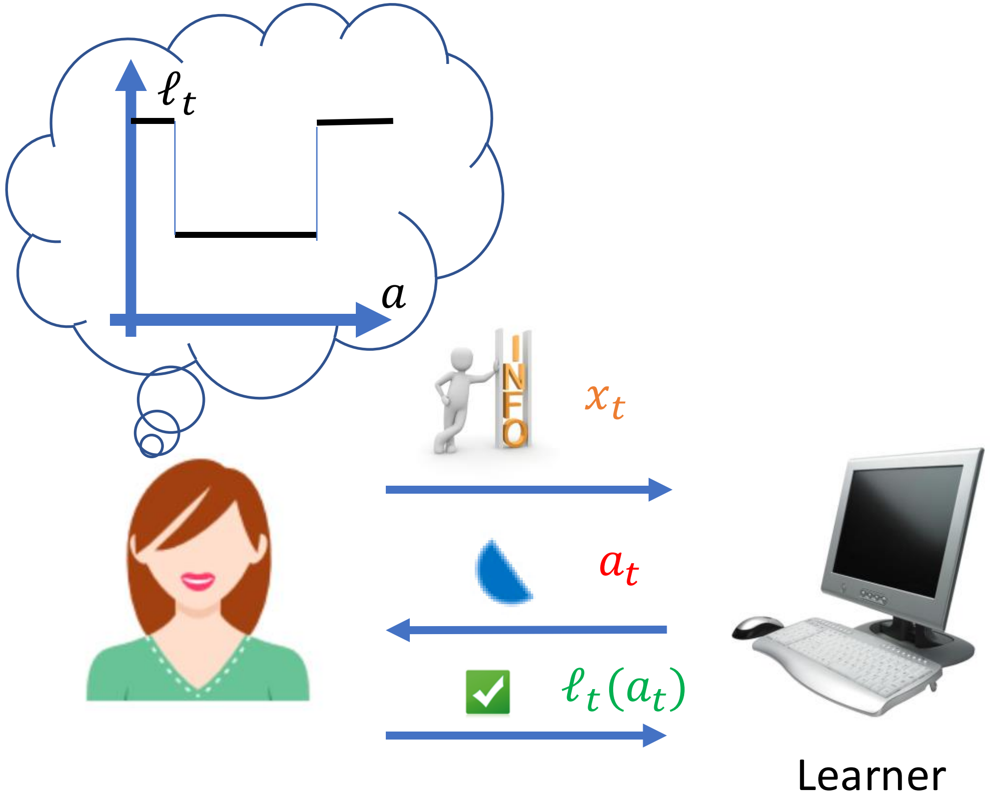
Akshay Krishnamurthy
Microsoft Research

John Langford
Microsoft Research

Alex Slivkins
Microsoft Research

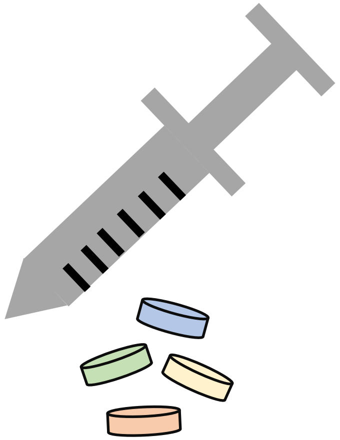
34th Conference on Neural Information Processing Systems, Vancouver, Canada
NeurIPS 2020 Workshop on Vowpal Wabbit

Contextual bandits (CB)

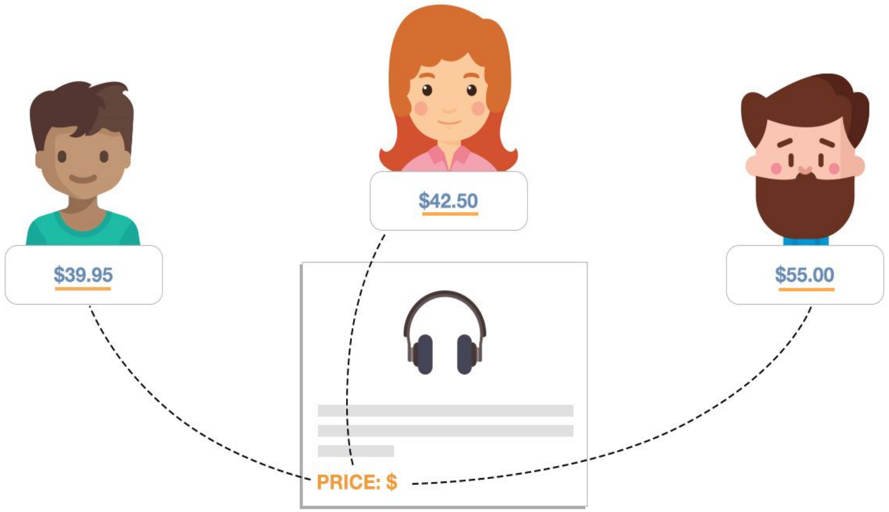


- For time step $t = 1, 2, \ldots, T$:
    - Receives context x_t
    - Takes an action $a_t \in A$
    - Receives loss $\ell_t(a_t) \in [0,1]$
- Learner's goal: minimize cumulative loss $\sum_{t=1}^{T} \ell_t(a_t)$
- In many practical settings the **action** chosen is actually ***continuous***.

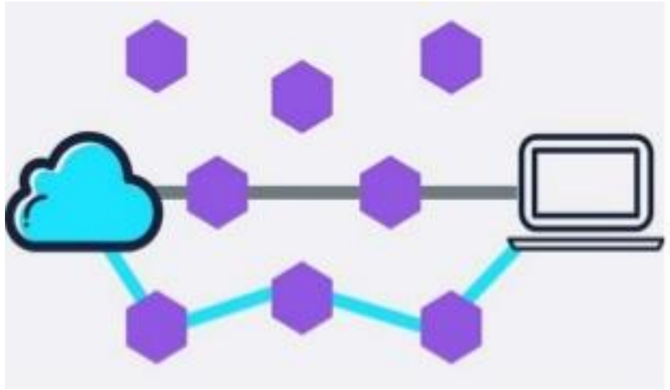
Continuous-action CB applications



Precision Medicine:
dosage



Dynamic pricing:
price [BM18]

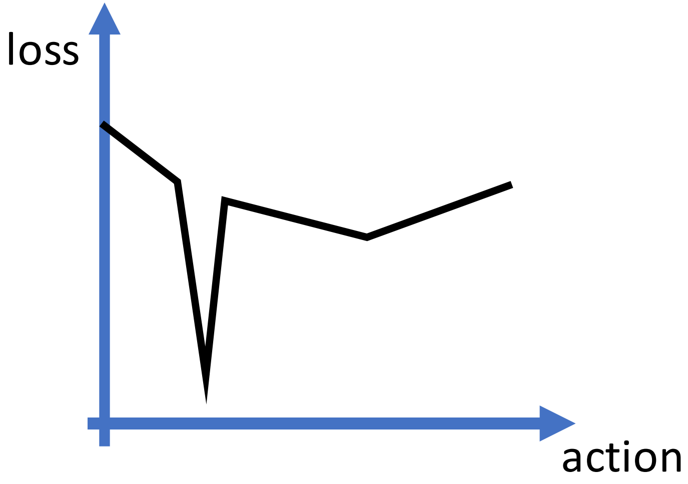


Networking:
packet sending rate [JRG19]

Challenges

- Discrete action spaces:
  - Can afford trying all possible actions through ``exploration”

- Continuous action spaces:
  - Need extra geometric assumptions to compete with best arm/policy.

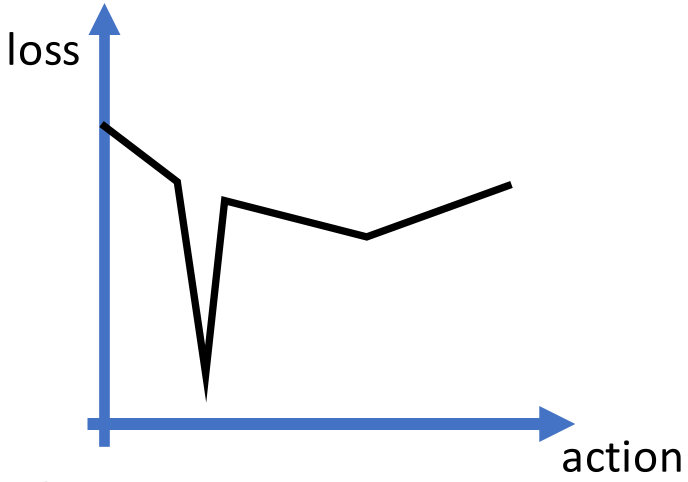


Formal setup

- Action space $A = [0, 1]$
- Policy: mapping from context to action (e.g. linear policy $\pi_w(x) = \langle w, x \rangle$)
- Policy class Π: a structured collection of policies (e.g. $\Pi = \{\pi_w : w \in \mathrm{R}^d\}$)

- Regret:

$$\mathrm{Reg}(T, \Pi) = \underbrace{\boxed{\sum_{t=1}^{T} \ell_t(a_t)}}_{\text{Loss of learner}} - \underbrace{\boxed{\min_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(\pi(x))}}_{\text{Loss of best policy}}$$



- Impossible to get nontrivial regret guarantees in general

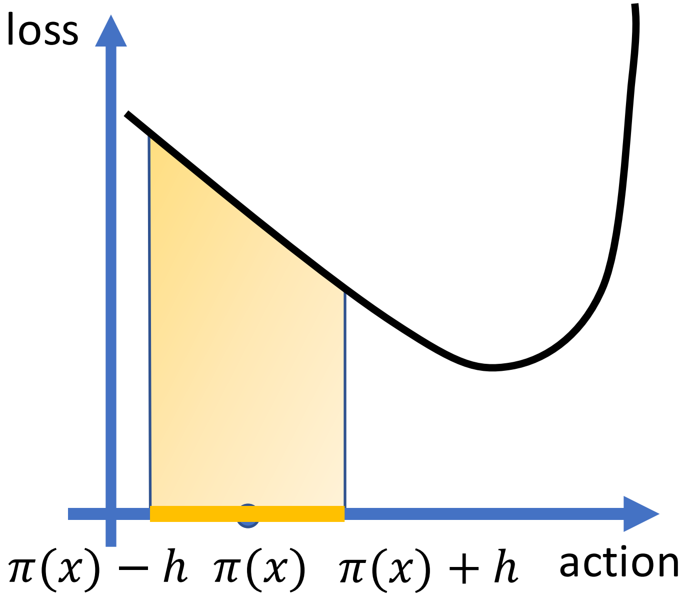
Smoothed regret

- Smoothed regret:

$$\mathrm{SReg}(T,\Pi,h) = \sum_{t=1}^{T} \ell_t(a_t) - \min_{\pi\in\Pi} \sum_{t=1}^{T} \mathrm{E}_{a\sim\pi_h(\cdot|x)}\ell_t(a)$$

where $\pi_h(\cdot\,|x)$ = uniform($[\pi(x) - h, \pi(x) + h]$)

and h is a fixed parameter (*bandwidth*).



  - Admits **assumption-free** nontrivial guarantees
  - Recovers many existing results in contextual bandits with smooth loss assumptions, e.g. Lipschitz losses

- **Problem**: the existing algorithms with sublinear smoothed regret are not computationally efficient.

CATS: Continuous Action Trees with Smoothing

- **ϵ-greedy with smoothing**: it chooses an action with probability
  ϵ : uniformly at random from [0, 1]
  $1 - \epsilon$: based on the prediction of the h-smoothing of the learned policy

- **Key idea 1:** Reduce CB learning to importance-weighted (IW) multiclass learning

- **Key idea 2:** Using tree policies to reduce IW multiclass learning to binary classification → computationally and statistically efficient

CATS: Continuous Action Trees with Smoothing

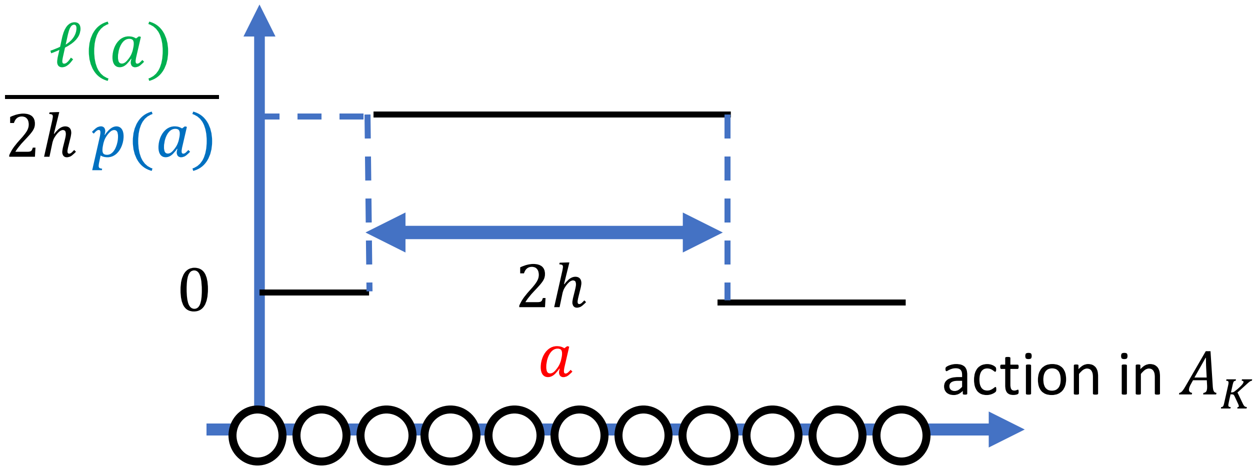
- **Key idea 1**: Reduce CB learning to importance-weighted (IW) multiclass learning

- Input: interaction log $S = \{(x, a, \ell(a), p(a))\}$,
  where $p(a)$ = probability density for action a.

1. Consider policy class Π taking actions in $A_K = \left\{0, \frac{1}{K}, \dots, \frac{K-1}{K}\right\}$.
2. For every input, generate cost-sensitive label using importance weighted loss estimate:

$$\hat{L}(\pi_h) = \frac{1}{|S|} \sum_{S} \frac{\pi_h(a \mid x)}{p(a)} \ell(a)$$
$$= \frac{1}{|S|} \sum_S \tilde{c}\left(\pi(x)\right)$$

where cost vector $\tilde{c}$ is:



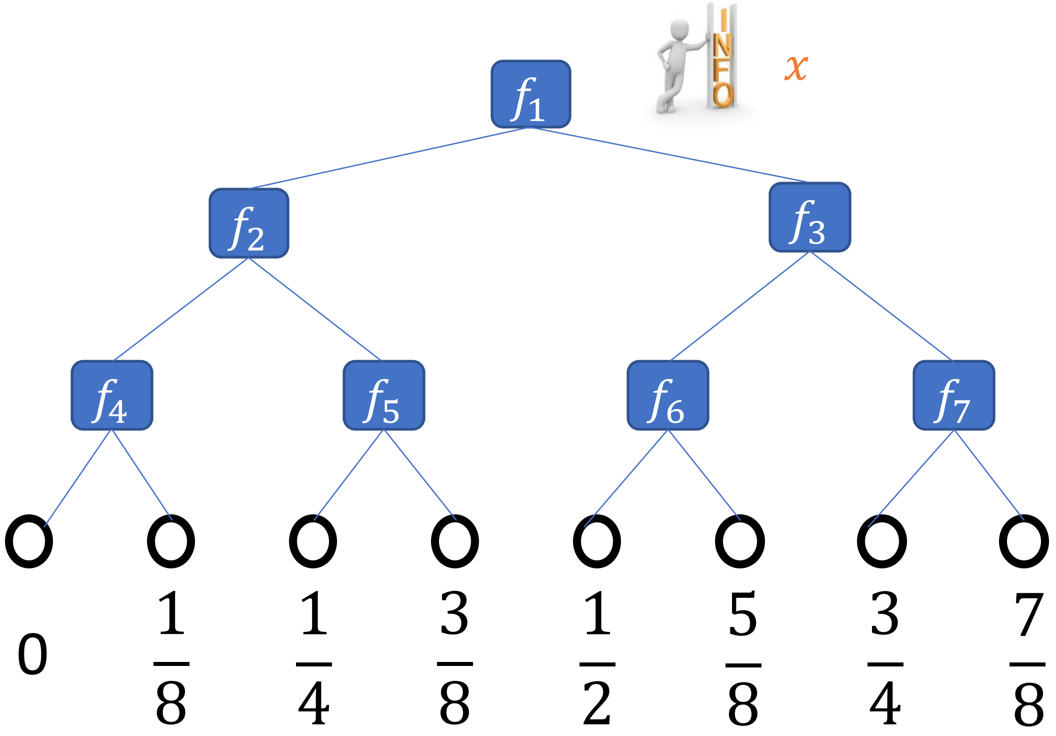
CATS: Continuous Action Trees with Smoothing

- **Key idea 2**: Using tree policies to reduce IW multiclass learning to binary classification

Tree policy: special form of decision tree with leaves associated with fixed action labels in A_K

- Internal nodes are binary classifiers

- Inference time: $O(\log K)$ per round

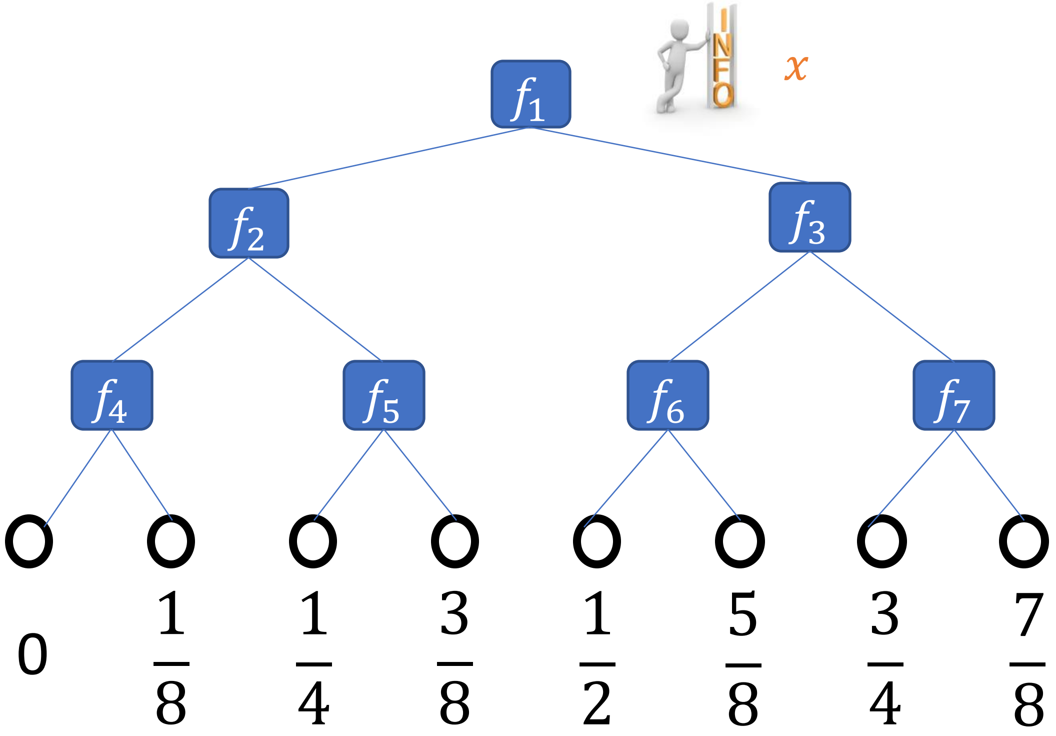


CATS: Continuous Action Trees with Smoothing

- **Key idea 2**: Using tree policies to reduce IW multiclass learning to binary classification

Training tree policies: we tailor the filter tree algorithm [BLR09], and show:

1. it can be implemented with $\boldsymbol{O}(\log \boldsymbol{K})$ time per example
2. it achieves statistical consistency under realizability



Provable guarantees

Base class F: {possible binary classifiers for internal tree nodes}
$\Pi_{K,F}$: {all tree policies with K leaves & base class F}

Theorem: *CATS* with input tree policy class $\Pi_{K,F}$ and bandwidth h:

- Computation: training time of $\boldsymbol{O(log\ K)}$ per example,
- Smoothed regret

$$\mathrm{SReg}(T, \Pi_{K,F}, h) \leq O\left(\left(\frac{K^2\ T^2 \ln|F|}{h}\right)^{1/3}\right)$$

under certain realizability assumptions

Experiment

- Evaluation: regression-based CB simulation

$$(x_t, y_t) \rightarrow (x_t, \ell_t), \text{ where } \ell_t = |a - y_t|$$

Baselines: perform naïve discretization with ϵ-greedy

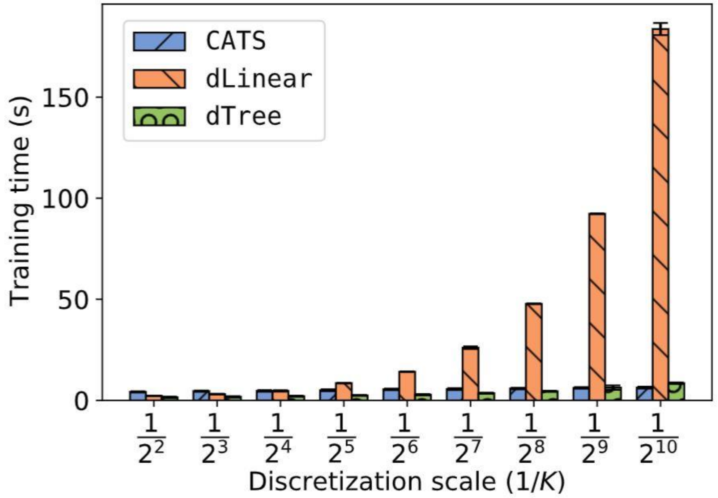
1. *dLinear*: reduces policy training to cost-sensitive one-versus-all multiclass classification

2. *dTree:* filter tree algorithm [BLR09]

Experiment: online contextual bandit learning

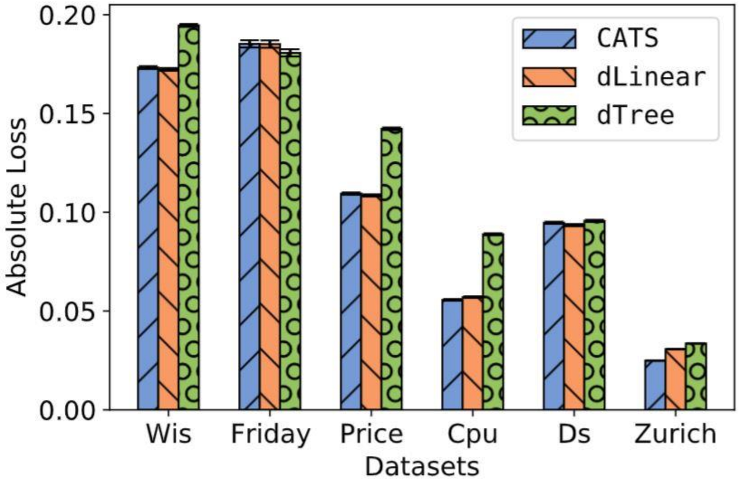
- Time cost comparison:

CATS and *dTree* have much better scalability with respect to K compared to *dLinear*



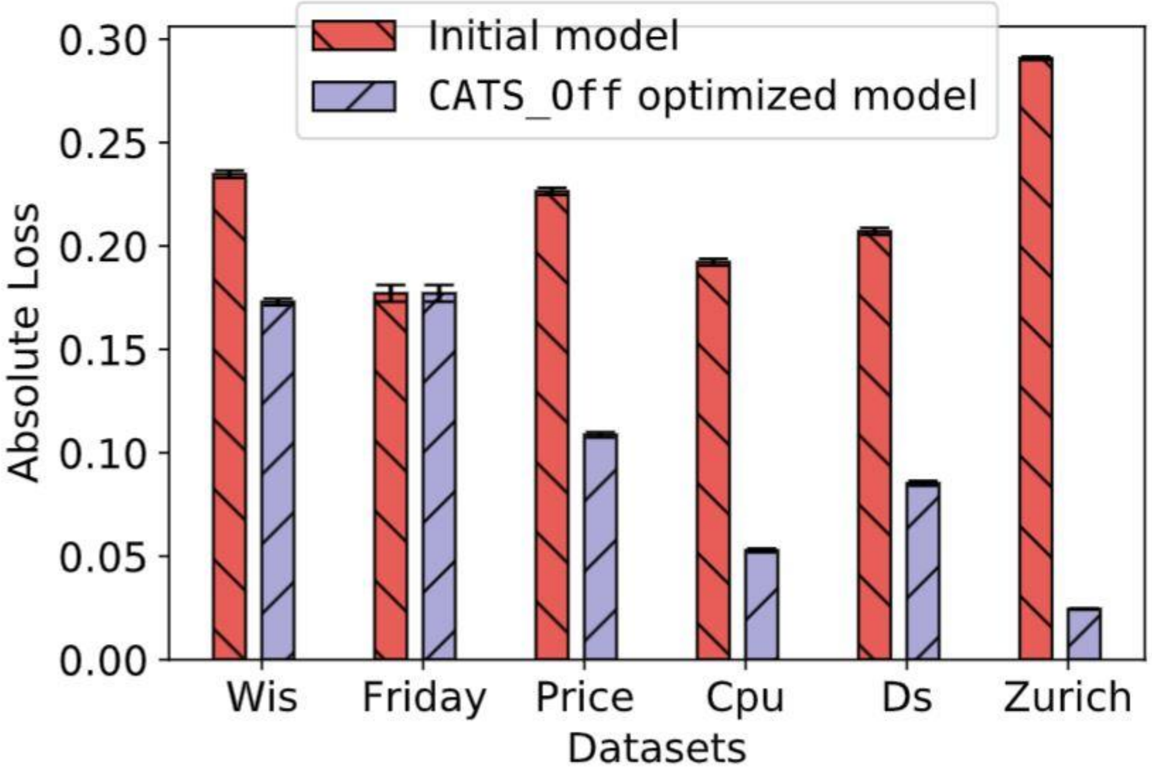
- Online loss comparison:

CATS and *dLinear* have lower average losses than *dTree*



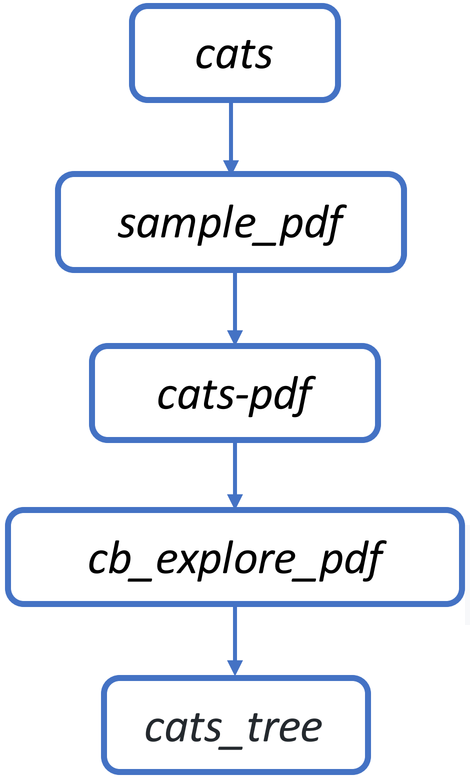
Experiment: off-policy optimization

- **Key advantage over naïve discretization**: it can use interaction log collected by one policy to do off-policy optimization over h and K.



How to use *CATS*?

- Vowpal Wabbit (VW): https://github.com/VowpalWabbit/vowpal_wabbit



cats

sample_pdf

cats-pdf

cb_explore_pdf

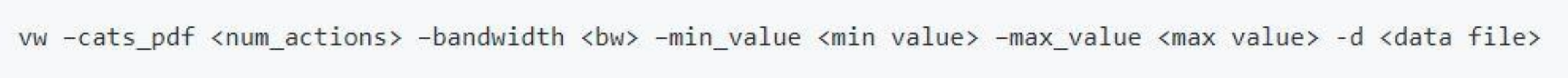
cats_tree

Return a single chosen action, along with its probability density

```
vw -cats <num_actions> -bandwidth <bw> -min_value <min value> -max_value <max value> -d <data file>
```

Returns action PDF:

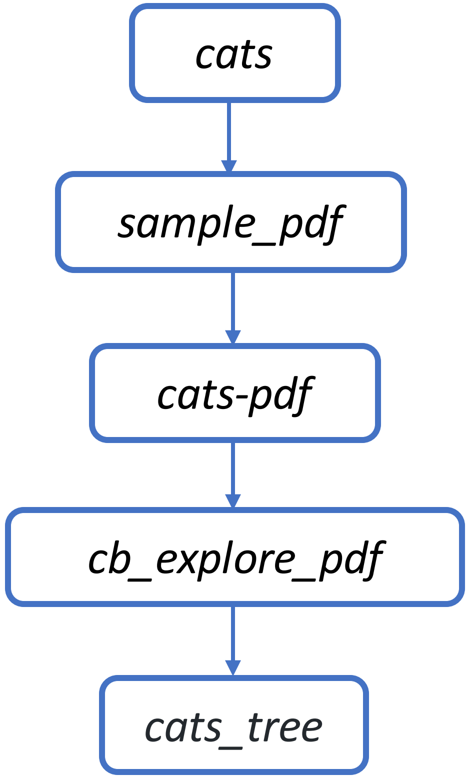
- Chosen action range with density = exploit probability
- The remaining range(s) with density = explore probability



```
vw -cats_pdf <num_actions> -bandwidth <bw> -min_value <min value> -max_value <max value> -d <data file>
```

How to use *CATS*?

- Prediction type:



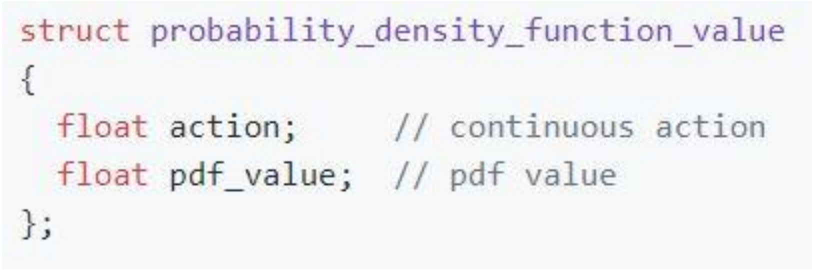
cats

sample_pdf

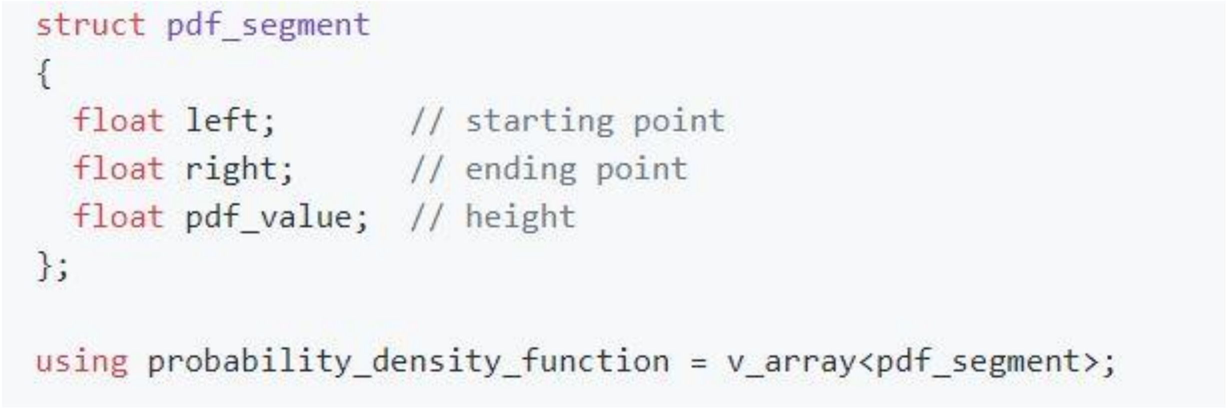
cats-pdf

cb_explore_pdf

cats_tree



```
struct probability_density_function_value
{
  float action;      // continuous action
  float pdf_value;   // pdf value
};
```

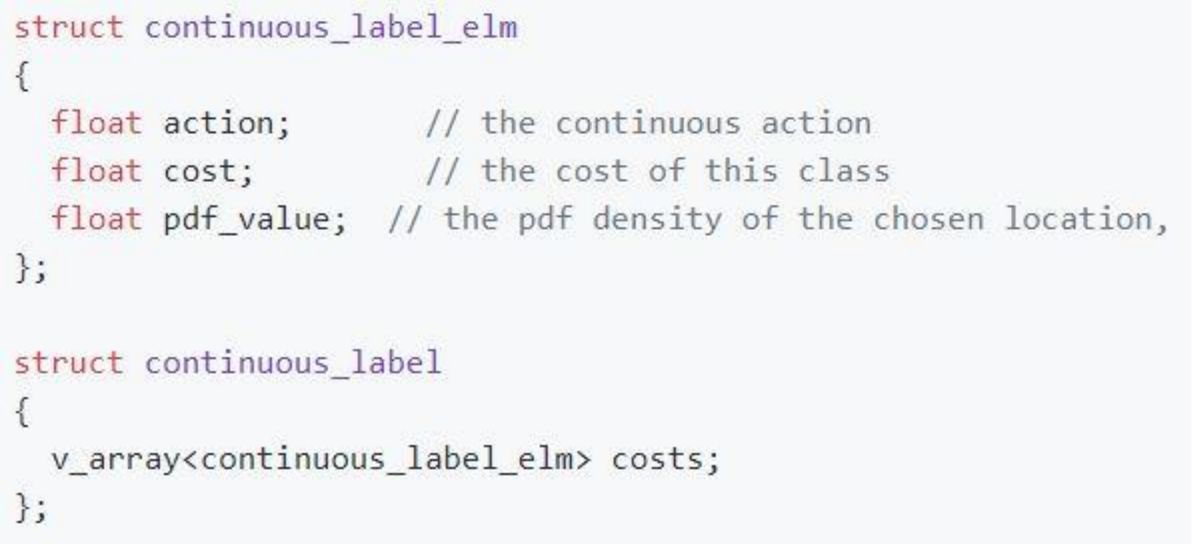


```
struct pdf_segment
{
  float left;        // starting point
  float right;       // ending point
  float pdf_value;   // height
};

using probability_density_function = v_array<pdf_segment>;
```

How to use *CATS*?

- Label type:

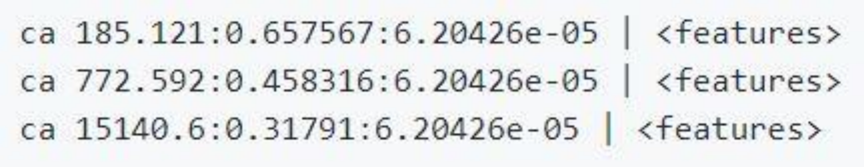


```
struct continuous_label_elm
{
  float action;       // the continuous action
  float cost;         // the cost of this class
  float pdf_value;  // the pdf density of the chosen location,
};

struct continuous_label
{
  v_array<continuous_label_elm> costs;
};
```

- Labelled example

```
ca action:cost:pdf_value |[namespace] <features>
```



```
ca 185.121:0.657567:6.20426e-05 | <features>
ca 772.592:0.458316:6.20426e-05 | <features>
ca 15140.6:0.31791:6.20426e-05 | <features>
```

Conclusion

- We propose a contextual bandit learning algorithm for continuous-action with unknown structure
- Theoretically and empirically: computationally and statistically efficient
- *CATS* code is available at Vowpal Wabbit

References

- [BM18] Dimitris Bertsimas and Christopher McCord. Optimization over continuous and multi-dimensional decisions with observational data. *NeurIPS 2018*
- [JRG19] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. A deep reinforcement learning perspective on internet congestion control. *ICML 2019*
- [KLSZ19] Akshay Krishnamurthy, John Langford, Aleksandrs Slivkins, and Chicheng Zhang. Contextual bandits with continuous actions: smoothing, zooming, and adapting. *COLT 2019*
- [BLR09] Alina Beygelzimer, John Langford, and Pradeep Ravikumar. Error-correcting tournaments. *ALT 2009*

Thank you!

Code: https://github.com/VowpalWabbit/vowpal_wabbit

arXiv: 2006.06040