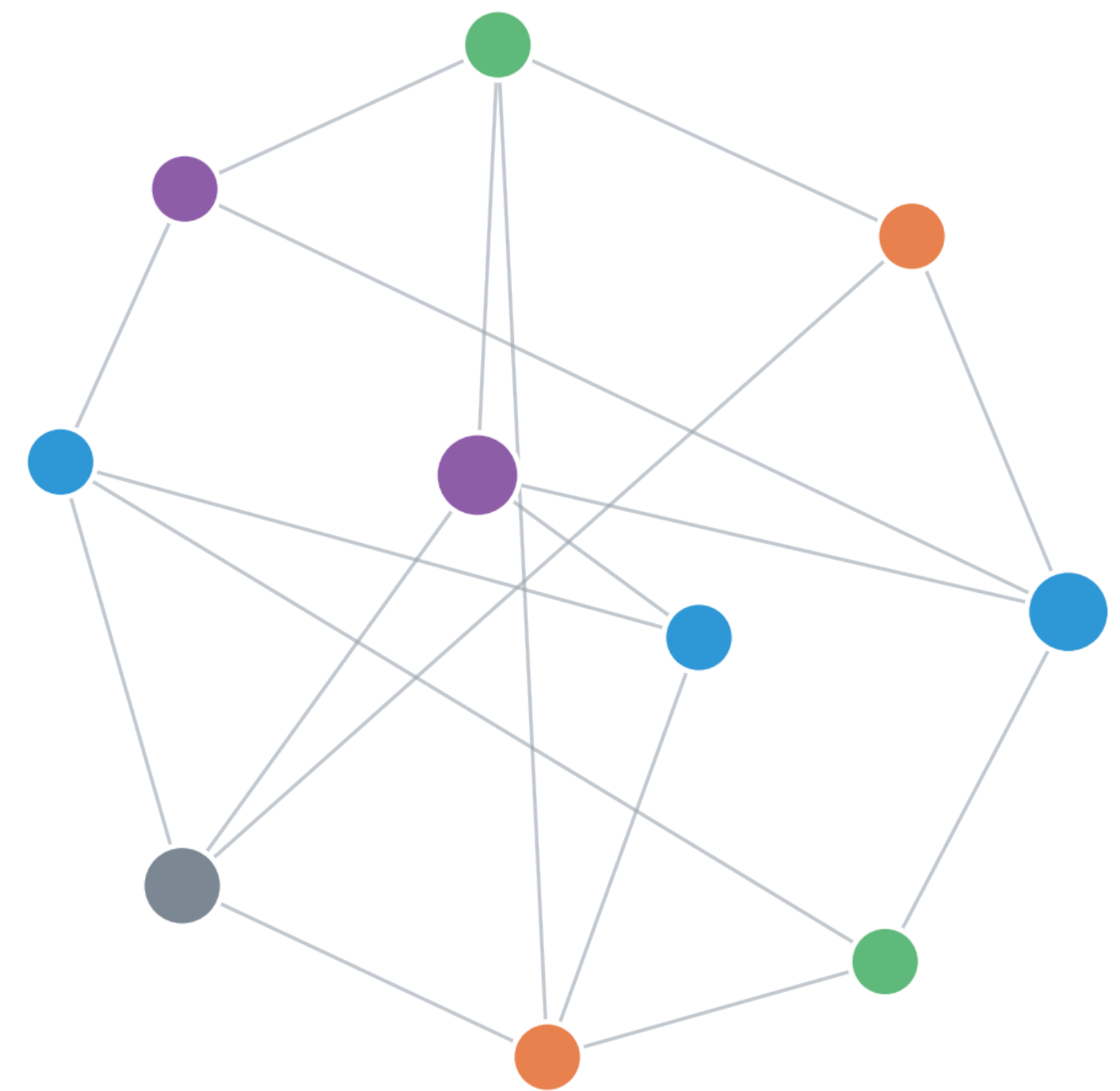


Reliable Machine Learning for Discrete Decision-Making

Ellen Vitercik · Stanford University



ML for combinatorial optimization

Combinatorial opt has long captured ML researchers' imaginations

- E.g., traveling salesman problem (TSP): iconic for theory and ML
 - 40 years from Hopfield networks [e.g., Hopfield, Tank, '86] to LLMs

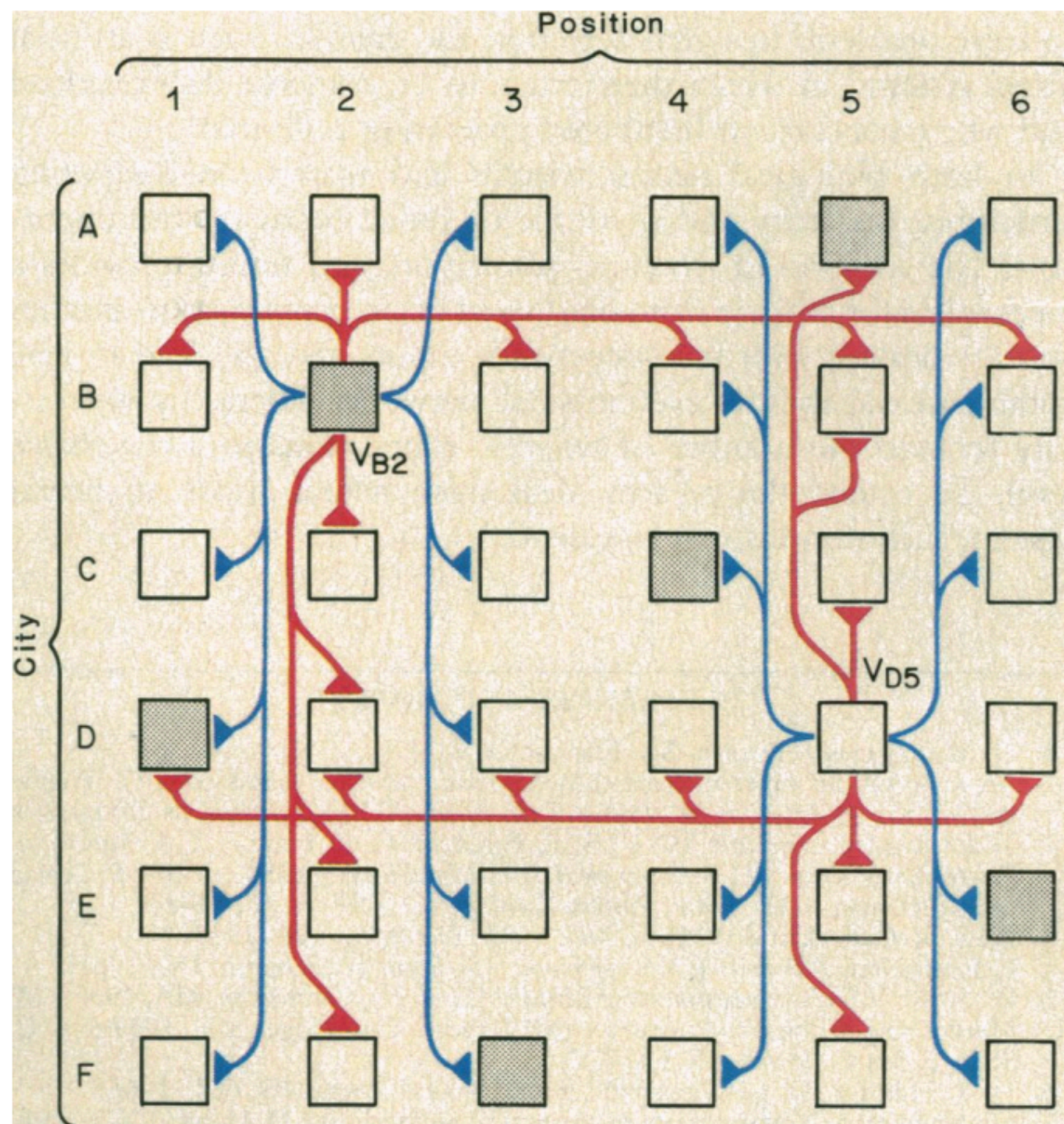


Fig. 5. A stylized picture of the syntax and connections of the **TSP neural circuit**. Each neuron is symbolically indicated by a square. The neurons are arranged in an n by n array. Each city is associated with n neurons in a row, and each position in the final tour is associated with n neurons in a column. A given neuron ($V_{X,j}$) represents the hypothesis that city X is in position j in the solution. The patterns of synaptic connection for two different neurons are indicated.

8 AUGUST 1986

ML for combinatorial optimization

Combinatorial opt has long captured ML researchers' imaginations

- E.g., traveling salesman problem (TSP): iconic for theory and ML
 - 40 years from Hopfield networks [e.g., Hopfield, Tank, '86] to LLMs
- Hard problems recur on **structured instance distributions**
 - Data reveals patterns not leveraged by worst-case analysis
- ML can **guide decisions** at multiple interfaces
 - Policies, solver controls, heuristics, configurations
- Inputs are often **uncertain** or **partially observed**
 - Predictions help, but errors can amplify

ML for combinatorial optimization

Many different forms

- **Goal 1:** Improve algorithms on structured instance distributions
 - Neural combinatorial optimization
 - ML for exact combinatorial optimization
 - Data-driven algorithm design
 - Automated heuristic discovery
- **Goal 2:** Learn uncertainty before optimizing
 - Data-driven robust optimization
 - Algorithms with predictions
 - Decision-focused learning
 - Neural algorithmic reasoning
- Discover new algorithms with provable guarantees?

Outline

1. Survey of different approaches

i. **Goal 1: Improve algorithms on structured instance distributions**

ii. Goal 2: Learn uncertainty before optimizing

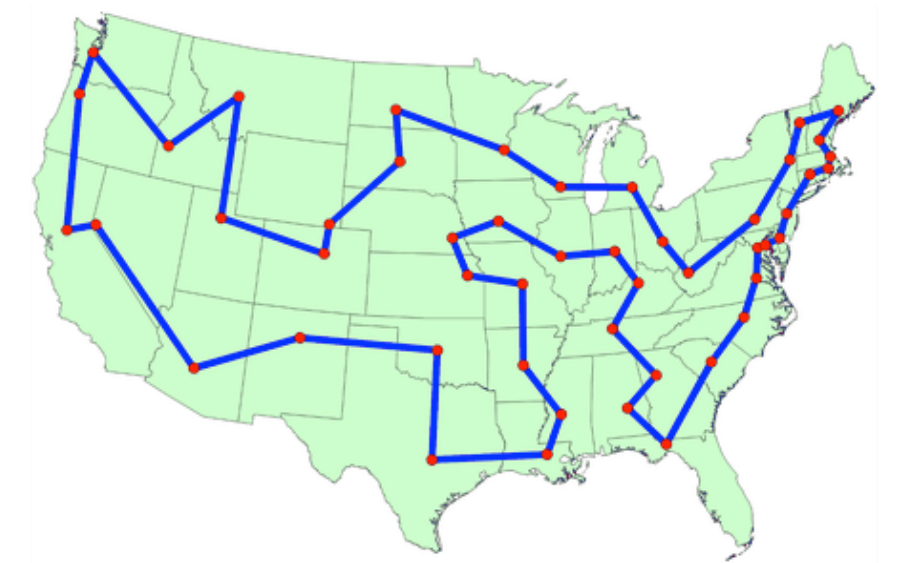
2. Algorithms with calibrated machine learning predictions [ICML'25]

3. Conclusions and future directions

Neural combinatorial optimization

Hopfield, Tank, '85; Durbin, Willshaw, '87; Vinyals et al., '15; Bello et al., '17; Dai et al., '17; Nazari et al., '18; ...

- Learn **end-to-end policies** for discrete optimization
 - Output solutions directly, without hand-designed heuristics
- Canonical problems: TSP, VRP, vertex cover, independent set, ...
- **Training:** imitation learning or reinforcement learning
 - **Decoding:** greedy, sampling, beam search
- **Core challenges:** feasibility, generalization, improvement over solvers
- **Theory angle:** what explains an architecture's strong performance?

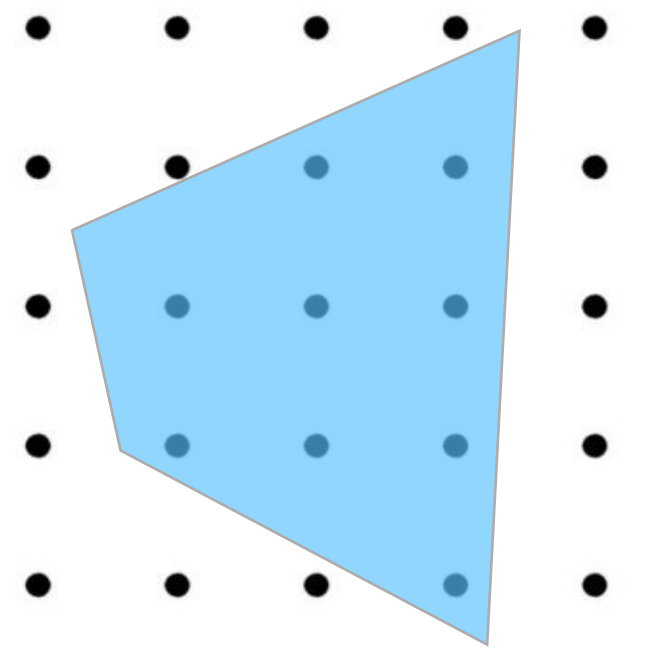


Hayderi, Saberi, **V**, Wikum, ICML'24

ML for exact combinatorial solvers

Gomes, Selman, '01; Nudelman et al., '04; Kadioglu et al., '11; Sandholm, '13; He et al., '14; Khalil et al., '16; ...

- **Keep solver**; learn the expensive choices
 - Branching, cutting, restarts, presolve, primal heuristics
- **Domains**: integer programming, SAT, CP
- **Objective**: reduce search tree nodes, wall-clock time
- ML guides; solvers certify solutions and proofs
- What's the right place to integrate LLMs?



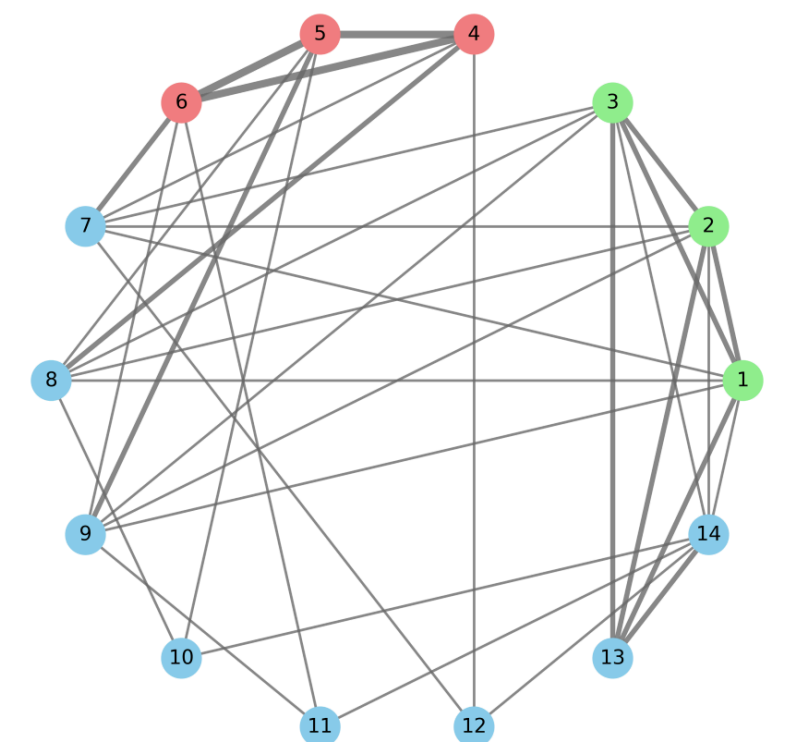
Lawless, Li, Wikum, Udell, [V](#), CPAIOR'25

LLMs for automated heuristic discovery

Liu et al., '23; Romera-Paredes et al., '24; Liu et al., '24; Ye et al., '24; Zhang et al., '24; Novikov et al., '25; ...

- Use LLMs to automate the human loop of **heuristic design**
- Search over programs, rules, priorities
- Heuristic generation/refinement via LLM-guided **evolutionary search**
 - Fitness: solution quality, runtime
- Examples: Heuristics for
 - Canonical problems like TSP, bin packing, ...
 - Discovering lower bound constructions

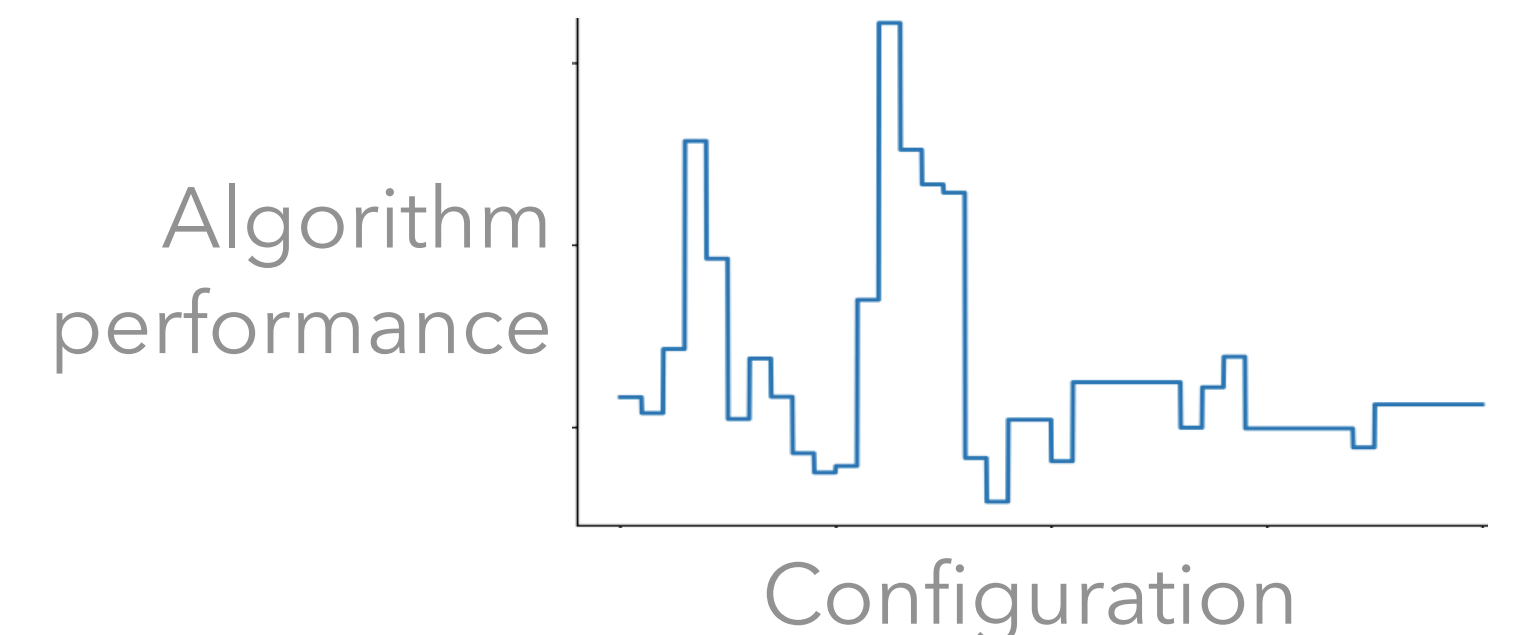
E.g., Nagda et al., '25



Data-driven algorithm design

Gupta, Roughgarden '16; Balcan, Nagarajan, **V**, White, '17; Kleinberg et al., '17; Balcan, Dick, **V**, '18; Garg, Kalai, '18; ...

- What **theory guarantees** can we give for learning algorithm configurations?
 - Examples: clustering, tree search, dynamic programming, ...
- Training data represents **historical** problem instances
 - Goal: strong expected performance – proxy for **future** performance
- Analyze generalization beyond sampled training set: **sample complexity**
- **Key challenge:** algorithm performance is volatile
 - Tweak parameters $\Rightarrow$ large changes in behavior



Goal 1: Improve algs on structured problems

Summary

Method	Where ML enters	What is learned/produced	Major focus/challenge
Neural CO	Solution construction	Neural policy	Feasibility, generalization
ML for exact solvers	Solver control	Branching/cutting/... rules	Distribution shift
LLM heuristic discovery	Heuristic design	Executable heuristic code	Benchmark overfitting
Data-driven algorithm design	Algorithm configuration	Tuned algorithm family	Sample complexity

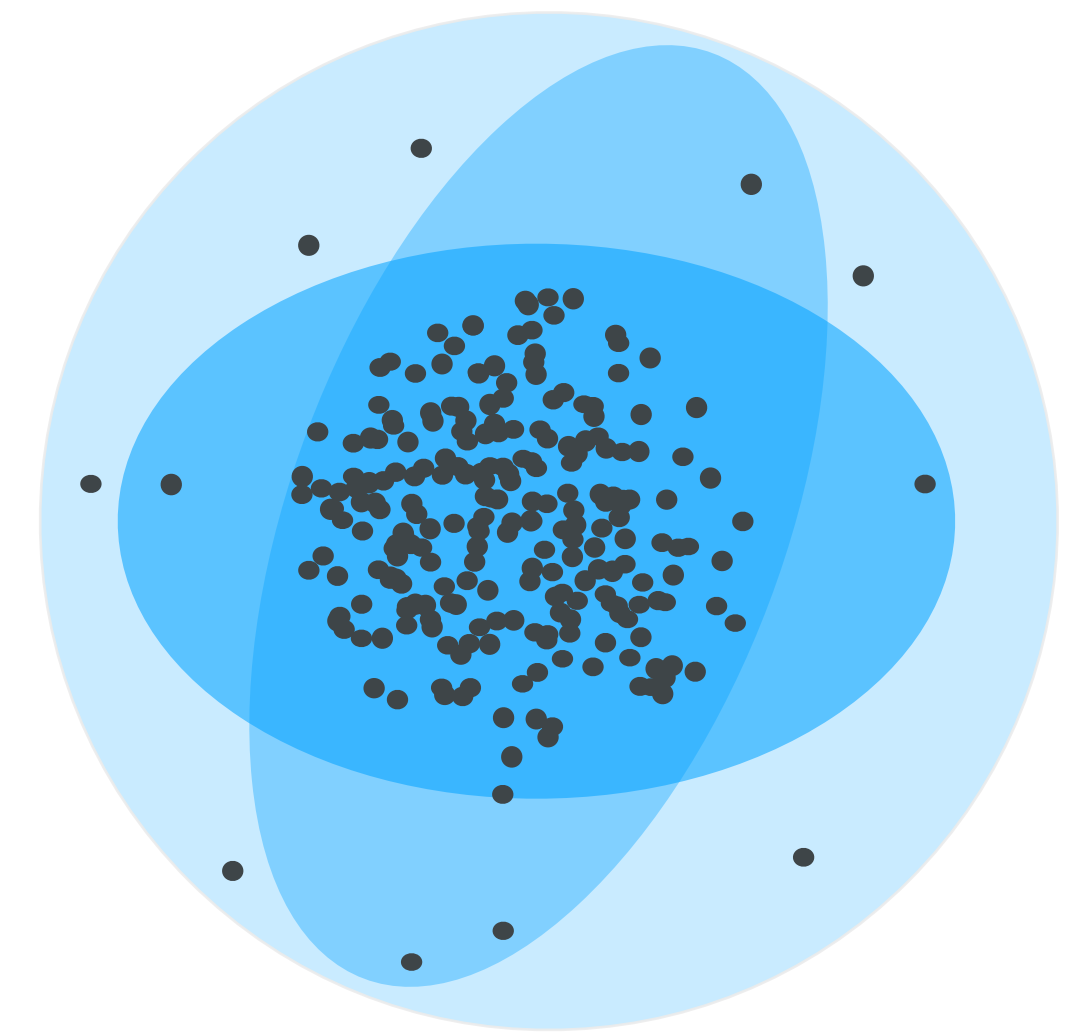
Outline

1. Survey of different approaches
 - i. Goal 1: Improve algorithms on structured instance distributions
 - ii. Goal 2: Learn uncertainty before optimizing**
2. Algorithms with calibrated machine learning predictions [ICML'25]
3. Conclusions and future directions

Data-driven robust optimization

Scarf, '58; Soyster, '73; El Ghaoui, Lebret, '97; Ben-Tal, Nemirovski, '98; Bertsimas, Sim, '04; Calafiore, Campi, '06; ...

- Learn an **uncertainty model** from historical data
 - Demands, costs, travel times
- Uncertainty enters the **optimization problem** via confidence/ambiguity sets, calibrated intervals
- Solver chooses decisions safe across plausible futures
- **Goal:** protect against sampling error and distribution shift
- **Tradeoff:** robustness versus conservatism



Algorithms with predictions

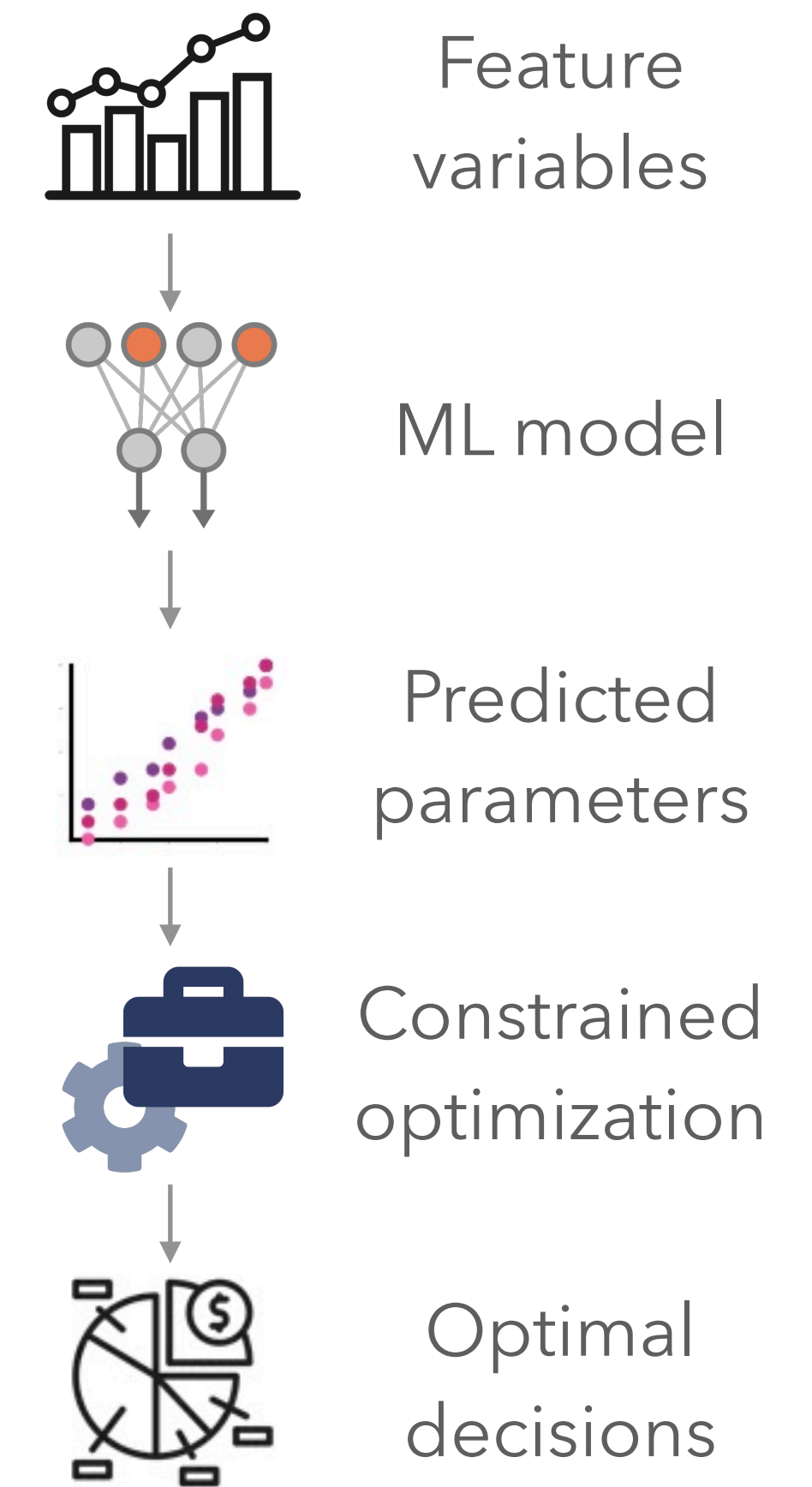
Mahdian, et al., '07; Devanur, Hayes, '09; Emek et al., '11; Boyar et al., '17; Lykouris, Vassilvitskii, '18; ...

- Use point predictions (typically) while preserving **worst-case guarantees**
- Add predictions to **classical algorithms** (future arrivals, sizes, ranks, optima)
- Predictions guide decisions (e.g., thresholds, evictions, matchings)
- **Goal:** exploit ML advice without over-trusting it
- **Robustness:** Retain worst-case guarantees when predictions fail
- **Consistency:** Guarantees scale with prediction error

Decision-focused learning

Bengio, '97; Amos, Kolter, '17; Donti et al., '17; Elmachoub, Grigas, '17; Wilder et al., '19; Agrawal et al., '19; ...

- Train predictors for **downstream optimization quality**
- Loss measures **final decision cost**, not prediction error
- **Optimization layer** maps predictions to decisions
 - Differentiable solvers or gradient surrogates
- Useful when prediction accuracy misaligns with value
- **Tradeoff:** decision quality versus training complexity



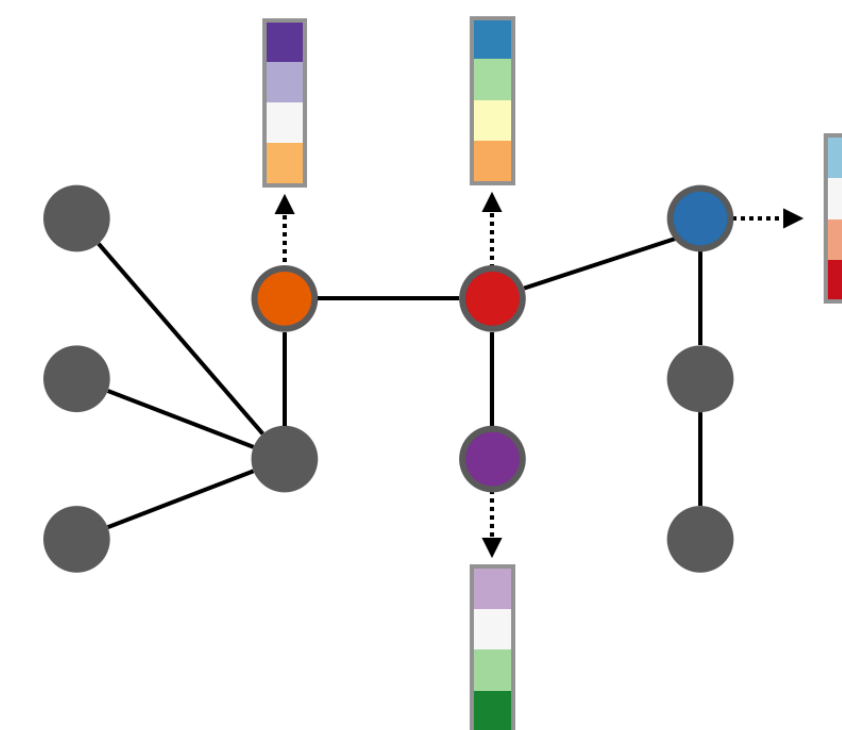
Mandi et al., '24

Neural algorithmic reasoning

Zaremba, Sutskever, '14; Graves et al., '14; Reed, de Freitas, '16; Veličković et al., '20; Veličković, Blundell, '21; ...

- Learn **algorithmic procedures internally** (no external solver call)
- Neuralize algorithmic **process** (e.g., dynamic programming, search)
- Encode algorithmic priors as **inductive biases**
- Desired wins: extrapolation (e.g., size generalization), robustness
- **Example:** GNNs can exactly simulate primal-dual algorithm

He, **V**, ICML'25



Goal 2: Learn uncertainty before optimizing

Summary

Method	What ML provides	Where ML enters	Output
Data-driven robust optimization	Uncertainty model	Optimization formulation	Robust decision
Algorithms with predictions	Point advice	Algorithm decision rule	Guided classical algorithm
Decision-focused learning	Decision-aware predictor	Training objective	Decision-aware predictor
Neural algorithmic reasoning	Learned computation	Neural architecture/process	Internal algorithm executor

Outline

1. Survey of different approaches
 - i. Goal 1: Improve algorithms on structured instance distributions
 - ii. **Goal 2: Learn uncertainty before optimizing**
- 2. Algorithms with calibrated machine learning predictions** [ICML'25]
3. Conclusions and future directions



Judy Hanwen Shen

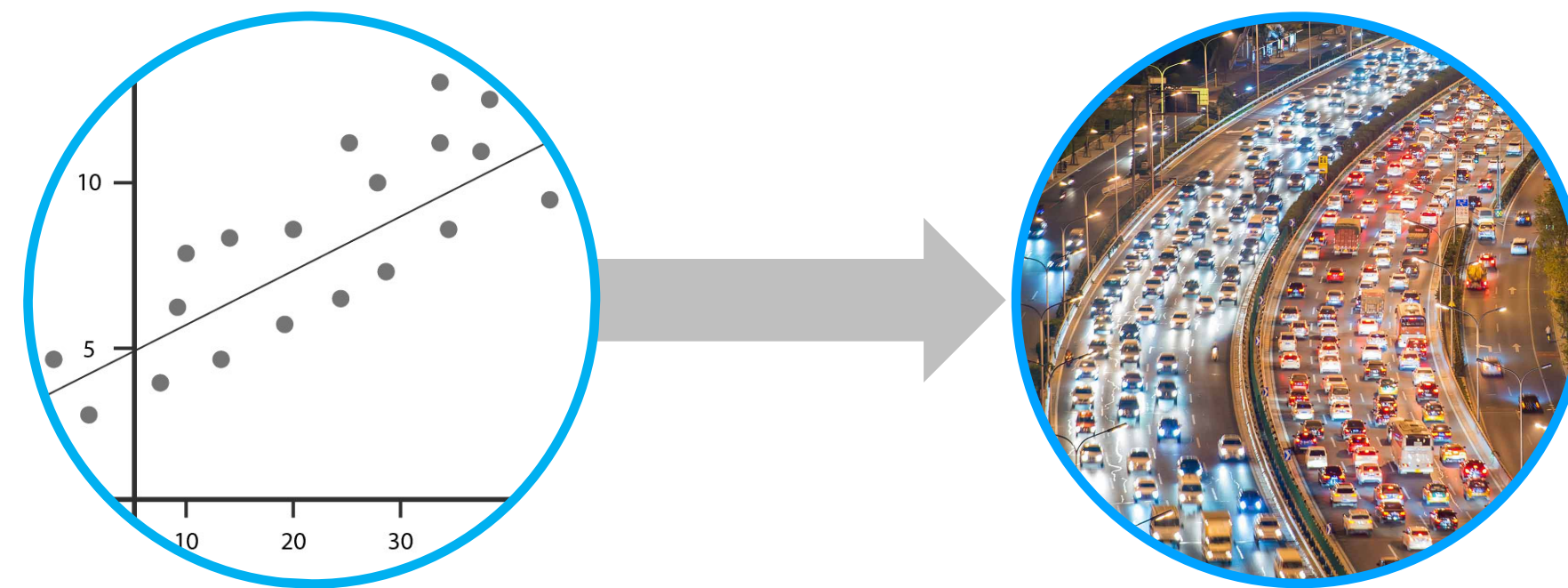


Anders Wikum

Algorithms and prediction uncertainty

Challenge: prediction **errors can amplify** in decision-making

Don't blindly trust predictions



Jumping off point: ML models can estimate uncertainty **automatically**

- Examples: calibration and conformal predictions
- Well-defined, statistical notion of whether a prediction can be trusted

Our contributions

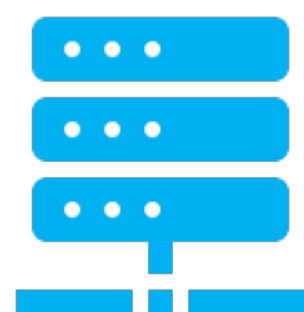
Demonstrate calibration's utility for online algorithms

Two case-studies:



Online rent-or-buy problems

- E.g., cloud vs equipment purchase, subscription vs lifetime licenses, ...
- Algorithm with guarantees that improve with accuracy and calibration error



Online job scheduling

- E.g., processing radiology diagnostic images
- Calibrated predictions yield better schedules than prior work [Cho et al., '22]

Validate methods on real-world datasets

Outline

1. Survey of different approaches
2. Algorithms with calibrated machine learning predictions [ICML'25]
 - i. **Background: Calibration**
 - ii. Calibration for online algorithms
3. Conclusions and future directions

Calibration (binary target)

- Random variables (X, Y) with support $\mathcal{X} \times \{0,1\}$
- $f: \mathcal{X} \rightarrow [0,1]$ is **calibrated** if $\mathbb{P}[Y = 1 \mid f(x) = p] = p$
- E.g., rain prediction: Weather is rainy
 - 50% of days where $f(X) = 0.5$
 - 100% of days where $f(X) = 1$
- Let $T(X) = \mathbb{P}[Y = 1 \mid f(X)]$ (equals $f(X)$ if perfectly calibrated)



- Calibrated, **unsharp**:
 $f(X) = \mathbb{P}[Y = 1]$ for all X
- Calibrated, **sharp**:
 $f(X) = \mathbb{P}[Y = 1 \mid X]$ for all X

$$\underbrace{\mathbb{E} \left[(Y - f(X))^2 \right]}_{\ell_2 \text{ error}} = \underbrace{\text{Var}(Y)}_{\text{Uncertainty}} - \underbrace{\text{Var}(T(X))}_{\text{Sharpness}} + \underbrace{\mathbb{E} \left[(T(X) - f(X))^2 \right]}_{\text{Calibration error}}$$

ℓ_2 error

Uncertainty

Sharpness

Calibration error

Outline

1. Survey of different approaches
2. Algorithms with calibrated machine learning predictions [ICML'25]
 - i. Background: Calibration
 - ii. Calibration for online algorithms
 - a. Rent-or-buy**
 - b. Job scheduling
3. Conclusions and future directions

Online rent-or-buy

- Models buy-vs-rent under unknown horizon
- Balances upfront cost against ongoing rental expenses



Cloud:
on-demand vs
reserved instances



Equipment:
lease machinery vs
purchase outright



Mobility:
transit passes,
bike share

Ski rental problem

Prototypical online **rent-or-buy** decision-making problem

- Skier will ski for some **unknown** number of days $Z \in \mathbb{R}_+$
- Each day, decide to rent skis for \$1 or buy for one-time cost \$ b
- **Goal:** Minimize total skiing cost
- Worst-case "breakeven" strategy:
Rent for b days, and if still want to ski, buy [Karlin et al. '01]

$$\text{Competitive ratio (CR)} := \frac{\text{ALG}}{\text{OPT}} = \frac{\text{amount algorithm pays}}{\min\{Z, b\}} \leq 2$$

Ski rental with predictions: Prior work

Algorithm with prediction of # skiing days Z [Kumar et al. NeurIPS'18]

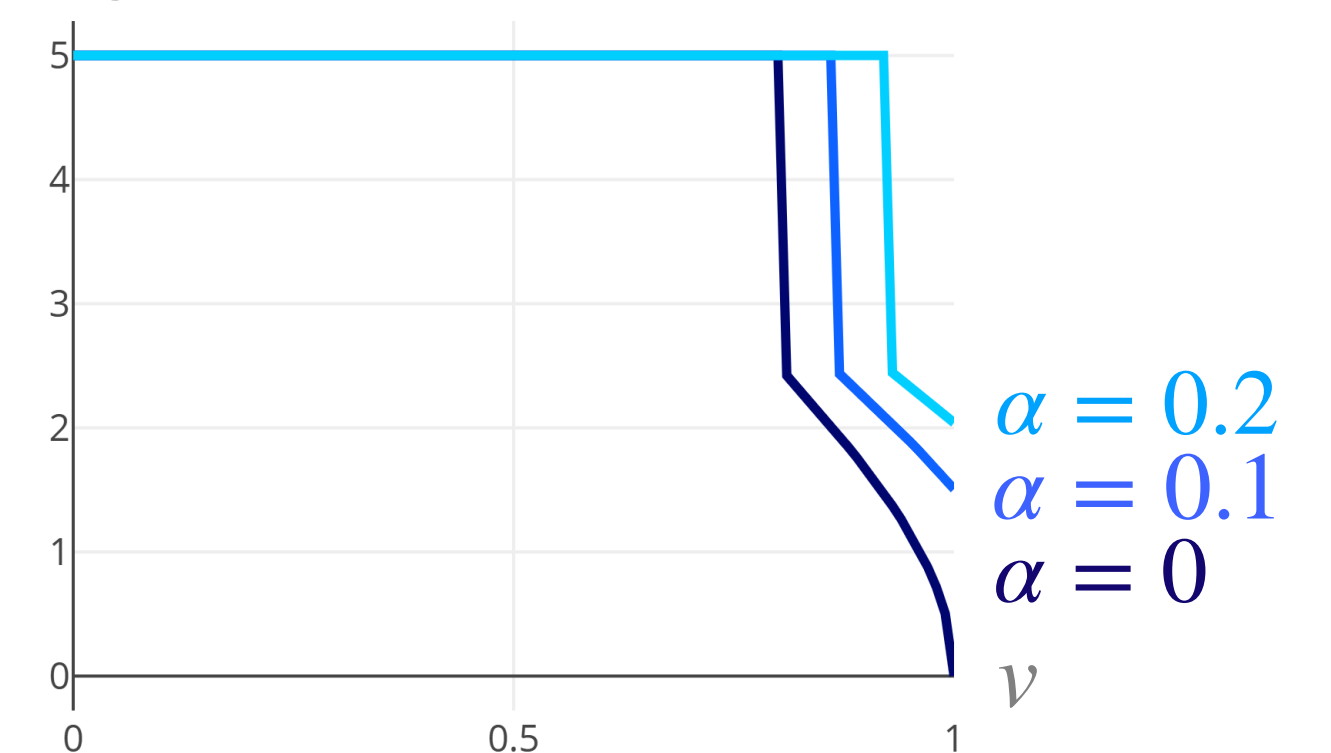
- Uses “trust” parameter $\lambda \in [0,1]$
 - $\lambda = 0$: fully trust predictor; $\lambda = 1$: don’t trust predictor at all
- **Algorithm:** If prediction $\geq b$: buy on day $\lceil \lambda b \rceil$. Else, buy on day $\lceil b/\lambda \rceil$.
- **Consistency guarantee:** Perfect prediction yields $CR \leq 1 + \lambda$
 - If you knew to set $\lambda = 0$, recover optimal $CR = 1$
- **Robustness guarantee:** Any prediction yields $CR \leq 1 + \frac{1}{\lambda}$
 - If you knew to set $\lambda = 1$, recover best worst-case $CR = 2$
- **Our goal:** leverage calibration to encode trust/uncertainty

Algorithm with calibrated prediction

- $\mathcal{X}$ = skier features
- Predictor $f(X)$ of target $Y = 1(Z > b)$
Max calibration error $\alpha = \max_{v \in R(f)} |v - \mathbb{P}[Y = 1 \mid f(X) = v]|$
- **Algorithm:** given prediction $f(X) = v$ rent for $k(v)$ days

$$k(v) = \begin{cases} b & \text{if } v \leq \frac{4 + 3\alpha}{5} \\ b\sqrt{\frac{1 - v + \alpha}{v + \alpha}} & \text{else} \end{cases}$$

Days rented $k(v)$, ($b = 5$)

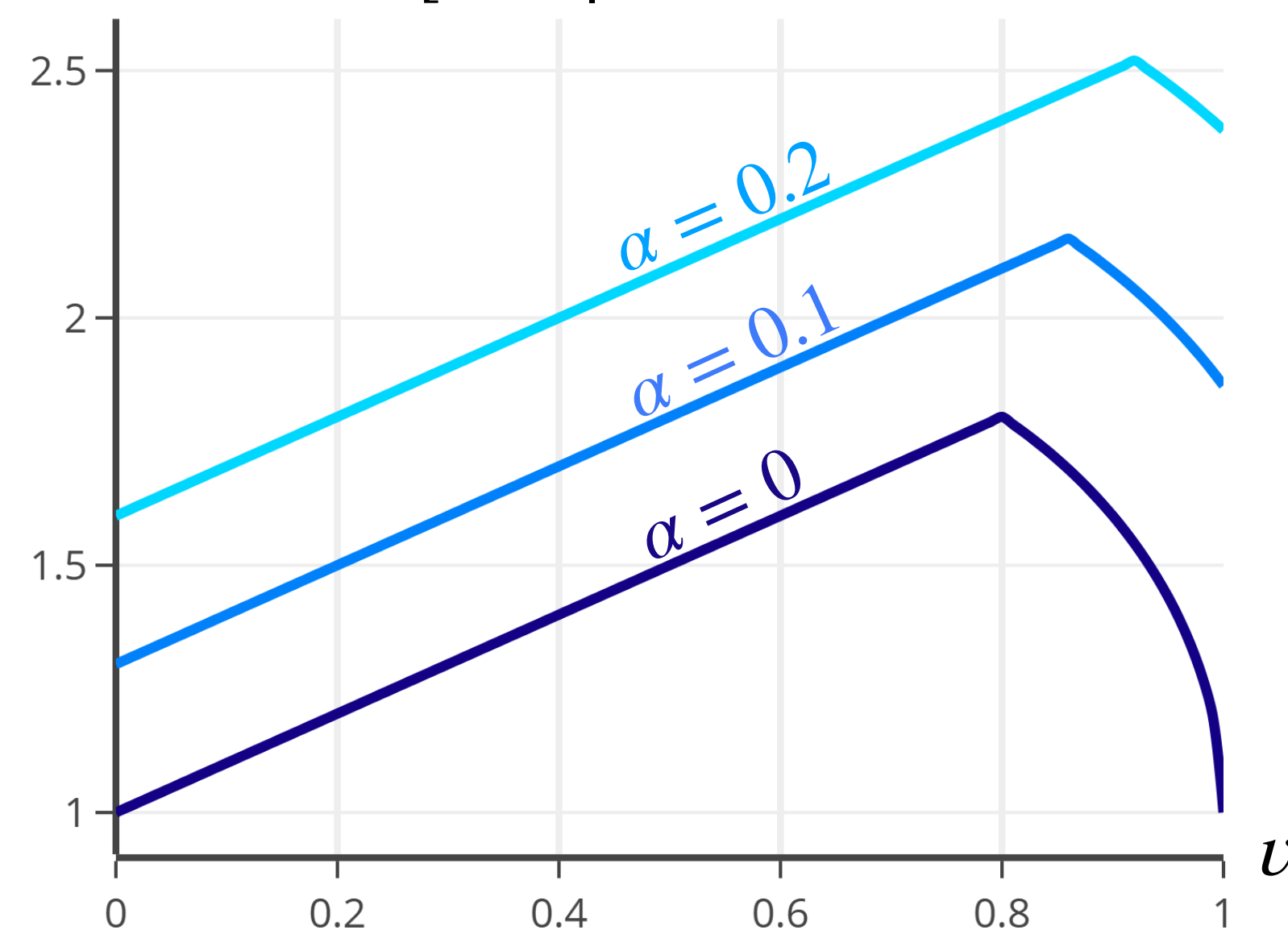


Ski rental: Main results

Prediction-wise bound:

$$\mathbb{E}[\text{CR} \mid f(X) = v] \leq 1 + 2\alpha + \min\left(v + \alpha, 2\sqrt{(v + \alpha)(1 - v + \alpha)}\right)$$

$\mathbb{E}[\text{CR} \mid f(X) = v]$



Ski rental: Main results

Prediction-wise bound:

$$\mathbb{E}[\text{CR} \mid f(X) = v] \leq 1 + 2\alpha + \min\left(v + \alpha, 2\sqrt{(v + \alpha)(1 - v + \alpha)}\right)$$

Lower bound: $\forall v$, exists distribution & calibrated predictor s.t.

$$\mathbb{E}[\text{CR} \mid f(X) = v] \geq 1 + \min\left(v, 2\sqrt{v(1 - v)}\right)$$

Global bound:

$$\mathbb{E}[\text{CR}] \leq 1 + 3\alpha + \min\left(\mathbb{P}[Z > b], 2\sqrt{\text{MSE}(f) + 3\alpha}\right)$$

✓ Lower MSE and calibration error lead to near-optimal CR

Outline

1. Survey of different approaches
2. Algorithms with calibrated machine learning predictions [ICML'25]
 - i. Background: Calibration
 - ii. Calibration for online algorithms
 - a. Rent-or-buy
 - b. Job scheduling**
3. Conclusions and future directions

Online job scheduling

Motivation [Cho et al. '22]: **radiologist's scheduling problem**

- One radiologist, backlog of imaging cases
- True case **urgency** initially **unknown**
 - Partial review reveals true urgency
- ML model provides **noisy** urgency probabilities
- Must choose processing order under uncertainty
- Goal: develop a simple, **robust scheduling policy**

Online job scheduling

- 1 machine to process n unit-length jobs
- Each job i has **unknown** high ($y_i = 1$) or low ($y_i = 0$) **priority**
 - Processing a θ -fraction of a job reveals its priority
- Jobs can be stored after partial processing
- Objective: Minimize weighted sum of completion times

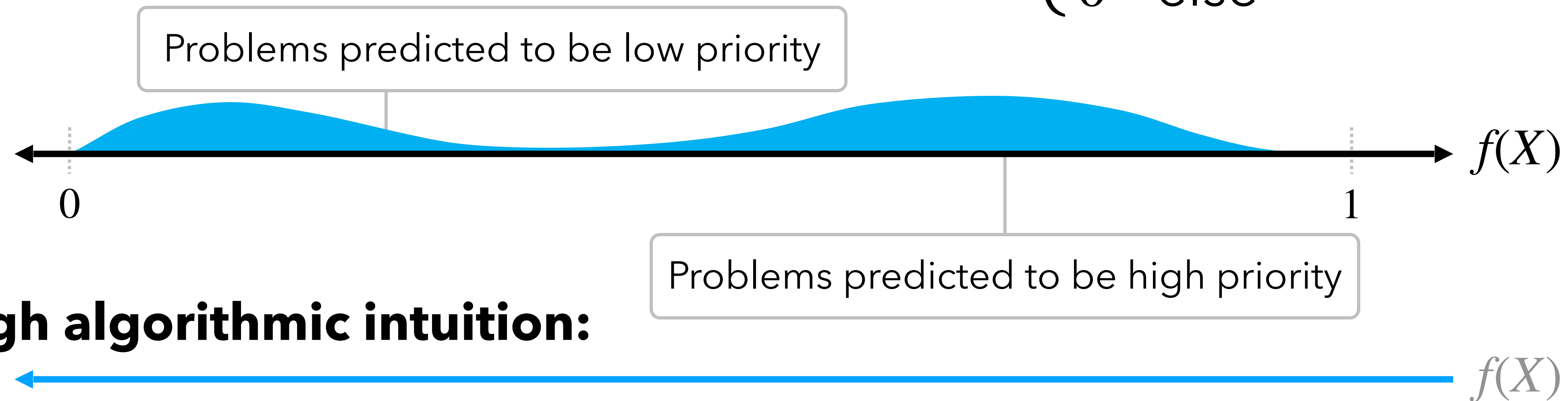
$$\sum C_i \cdot \omega_{y_i}$$

Completion time of job i

Cost per unit delay, with $\omega_1 > \omega_0 > 0$

Algorithmic intuition

Given features X , calibrated $f(X) \in [0,1]$ of $Y = \begin{cases} 1 & \text{if job is high priority} \\ 0 & \text{else} \end{cases}$



Rough algorithmic intuition:

Run jobs predicted to be high priority first, then low priority later, preemptively

If discover current is low-priority, stop, move to end of queue, and start new job

Importance of predictor sharpness

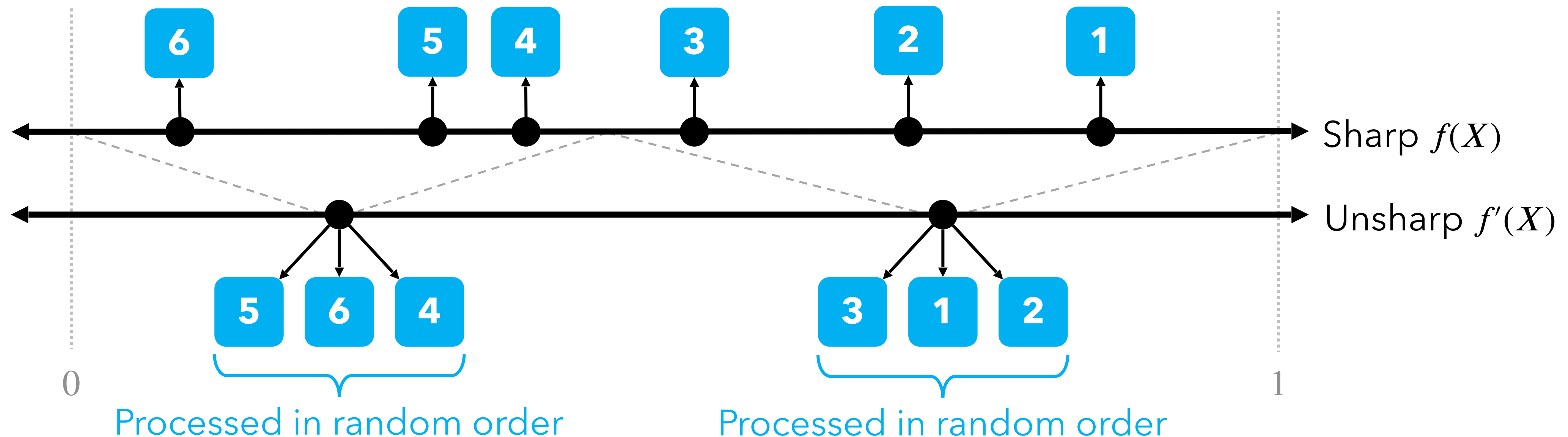
Key insight: **interchanges** are the primary source of **regret**

Weighted sum of completion times compared to optimal in hindsight

Importance of predictor sharpness

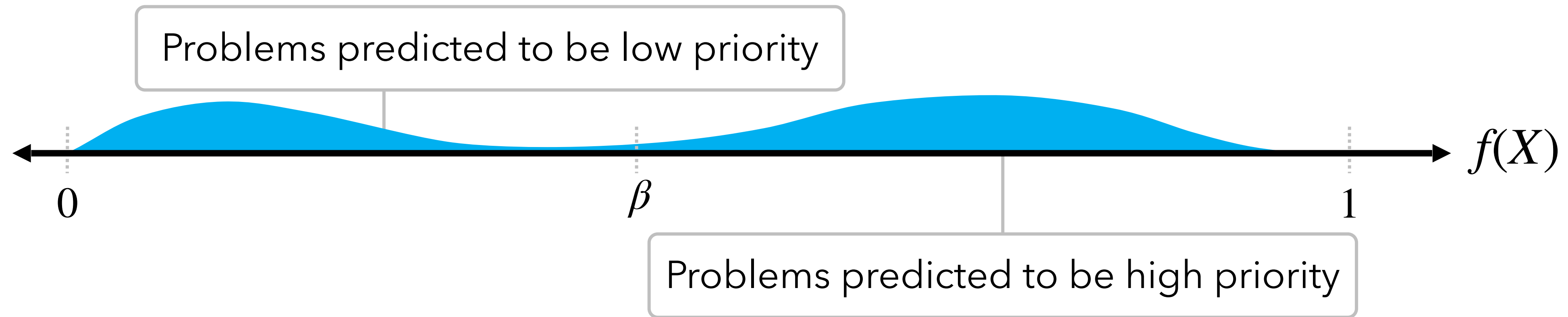
Key insight: **interchanges** are the primary source of **regret**

- Low priority job (partially) processed before high priority job
- **Sharp predictors** lead to **fewer interchanges**



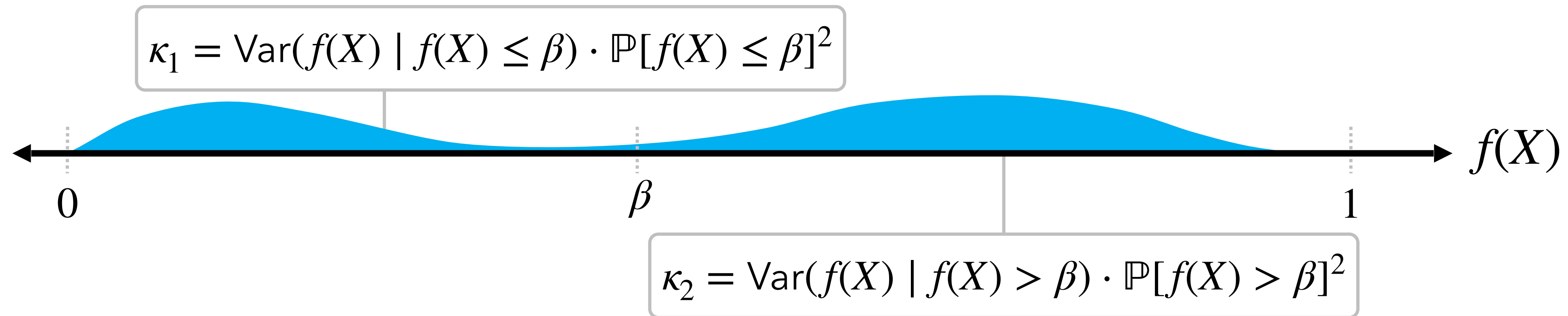
Importance of predictor sharpness

More formally



Importance of predictor sharpness

More formally



Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0$, $\mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

$$\mathbb{E}[\text{fraction of interchanged jobs}] \leq \underbrace{\text{Var}(Y)}_{\text{Inherent uncertainty}} (\underbrace{\epsilon_0 + \epsilon_1}_{\text{False positive/negative}}) - \underbrace{(\kappa_1 + \kappa_2)}_{\text{Sharpness}}$$

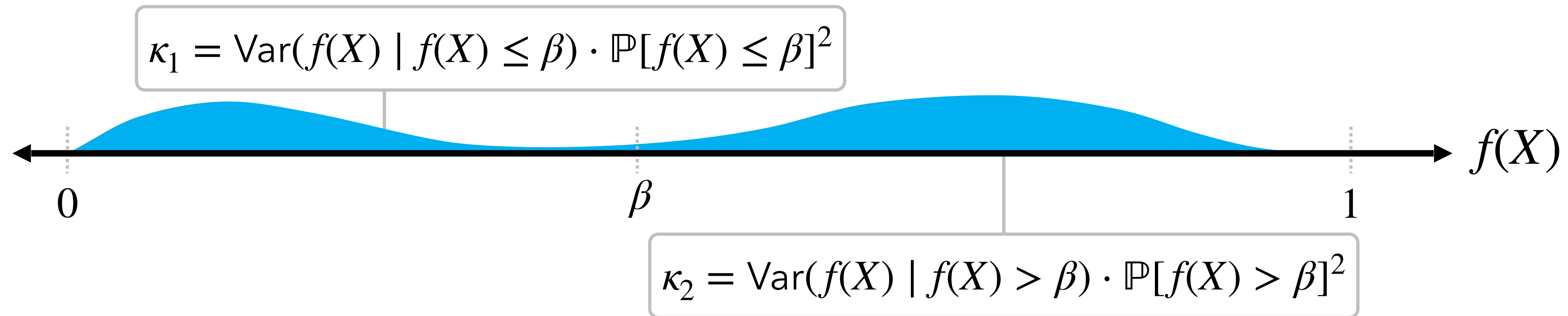
Inherent
uncertainty

False
positive/negative

Sharpness

Importance of predictor sharpness

More formally



Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0$, $\mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

$$\mathbb{E}[\text{fraction of interchanged jobs}] \leq \text{Var}(Y)(\epsilon_0 + \epsilon_1) - (\kappa_1 + \kappa_2)$$

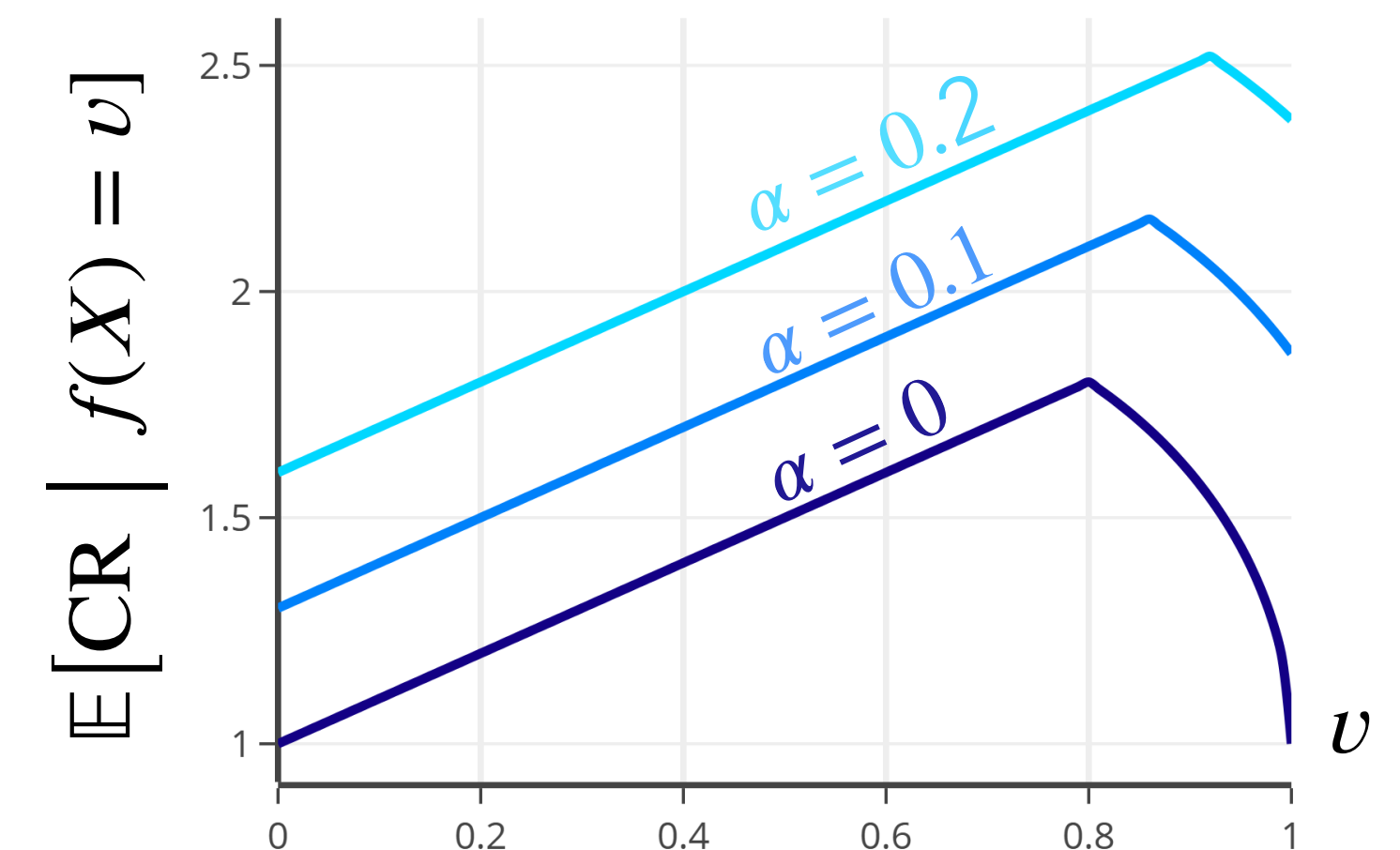
(Eventual) corollary: bound on algorithm's regret (see paper)

Outline

1. Introduction
2. Survey of different approaches
3. Algorithms with calibrated machine learning predictions [ICML'25]
- 4. Conclusions and future directions**

Conclusions

- **Calibration:** valuable tool for integrating ML into algorithms
- **Case studies:** ski rental and job scheduling
- Performance guarantees improve with calibration error
- Validated methods on real-world datasets



Ongoing directions

Connections to data-driven algorithm design

- Central question of **data-driven algorithm design**:
 - What components of algorithmic pipelines are learnable?
 - Algorithm parameters, uncertain aspects of the problem instance, ...
- **Connection to talk**: *max calibration error* is not efficient to optimize
- **Question**: what UQ methods are **learnable** and **algorithmically useful**?
- Connections to different ways to measuring mis-calibration
[E.g., Błasiok et al., '23; Kleinberg et al., '23; ...]

Reliable Machine Learning for Discrete Decision-Making

Ellen Vitercik · Stanford University

