How much data is sufficient to learn high-performing algorithms?

Generalization guarantees for data-driven algorithm design

Nina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, **Ellen Vitercik**

Data-driven algorithm design

Algorithms often have **many tunable parameters**
Significant impact on runtime, solution quality, …

Hand-tuning is **time-consuming**, **tedious**, and **error prone**

Goal: Automate algorithm configuration via machine learning
Input: ***Training set*** of typical problem instances from application at hand
Sampled from unknown, **application-specific distribution**
Output: Configuration with strong **average empirical performance** on training set
Runtime, solution quality, etc.

Parameter setting should—**ideally**—be good on future inputs

Summary of contributions

Broadly applicable theory for deriving ***generalization bounds***:

| Algorithm's **average** performance on training set − Algorithm's **expected** performance on unknown distribution | ≤ **?**

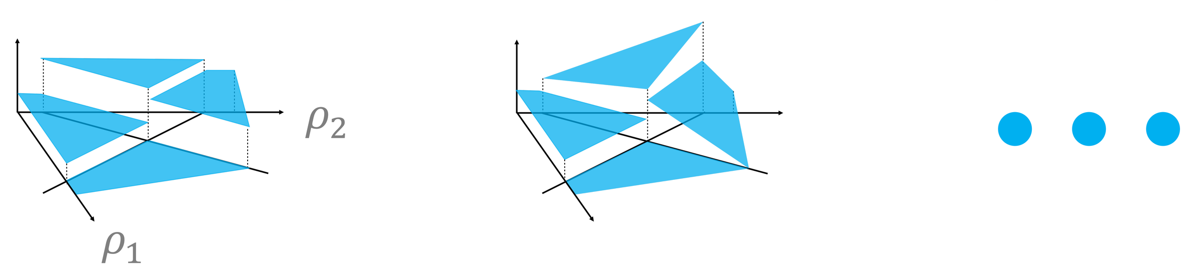
Prior research proved generalization bounds case-by-case
Gupta, Roughgarden, ITCS'16; Balcan, Nagarajan, **V**, White, COLT'17; Balcan, Dick, Sandholm, **V**, ICML'18; Balcan, Dick, White, NeurIPS'18; Balcan, Dick, Lang, ICLR'20; …

Clustering algorithm configuration | Integer programming algorithm configuration | Selling mechanism configuration | Greedy algorithm configuration | Computational biology algorithm configuration | Voting mechanism configuration

Recover bounds | Prove **novel** bounds

We uncover **overarching structure** linking these seemingly disparate domains

Guarantees apply to any parameterized algorithm where:
Performance is a ***piecewise-structured*** function of parameters



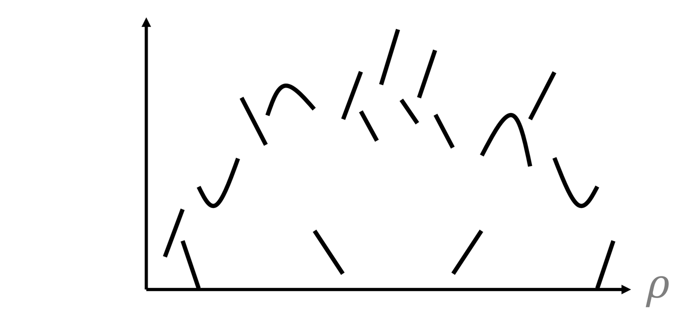
Piecewise constant | **Piecewise linear** | **Piecewise …**

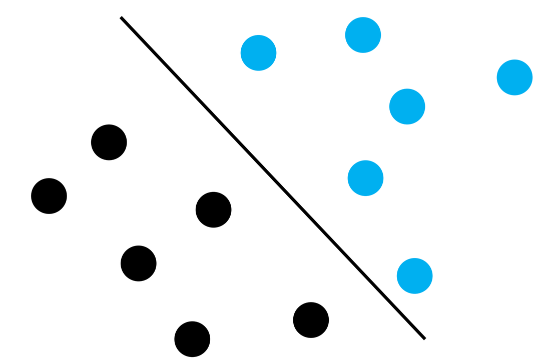
Additional references

- Book chapter by Balcan [Cambridge University Press '20]
- Online algorithm configuration:
  Exploited that the dual functions are piecewise Lipschitz to provide regret bounds
  [Balcan, Dick, **V**, FOCS'18; Balcan, Dick, Pegden, UAI'20; Balcan, Dick, Sharma, AISTATS'20]

Primary challenge

Performance is a **volatile** function of parameters
Complex connection between parameters and performance





Meanwhile, for well-understood functions in machine learning theory:
Simple connection between function parameters and value

Running example: Sequence alignment

```
          Gap
           ↓
    A  –  –  C  T  G
    –  G  T  C  A  –
          ↑  |  ↑
      Indel  |  Mismatch
           Match
```

Standard algorithm with parameters $\rho_1, \rho_2, \rho_3 \geq 0$:
Return alignment maximizing:
(# matches) − ρ_1 · (# mismatches) − ρ_2 · (# indels) − ρ_3 · (# gaps)

*"There is **considerable disagreement** among molecular biologists about the **correct choice** [of $\boldsymbol{\rho}$]"* [Gusfield et al. '94]

Model and problem formulation

$\mathbb{R}^d$: Set of all parameters
$\mathcal{X}$: Set of all inputs (e.g., sequence pairs)

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
Runtime, solution quality, …

Assume $u_{\boldsymbol{\rho}}(x) \in [-1,1]$

Standard assumption: Unknown distribution $\mathcal{D}$ over inputs
Models specific application domain at hand

Generalization bound: Given samples $x_1, \dots, x_N \sim \mathcal{D}$, for any $\boldsymbol{\rho}$,

$$\left|\frac{1}{N}\sum_{i=1}^{N} u_{\boldsymbol{\rho}}(x_i) - \mathbb{E}_{x\sim\mathcal{D}}[u_{\boldsymbol{\rho}}(x)]\right| \leq \textbf{?}$$

Empirical average utility | **Expected utility**

Main result

$\mathcal{U} = \{u_{\boldsymbol{\rho}} : \mathcal{X} \to \mathbb{R} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$ **"Primal" function class**

Typically, prove generalization guarantees by bounding the ***complexity*** of $\mathcal{U}$
VC dimension, Rademacher complexity, …

Challenge: $\mathcal{U}$ is gnarly. E.g., in sequence alignment:
- Each domain element is a pair of sequences
- Unclear how to plot/visualize functions $u_{\boldsymbol{\rho}}$
- No obvious notions of Lipschitzness or smoothness to rely on

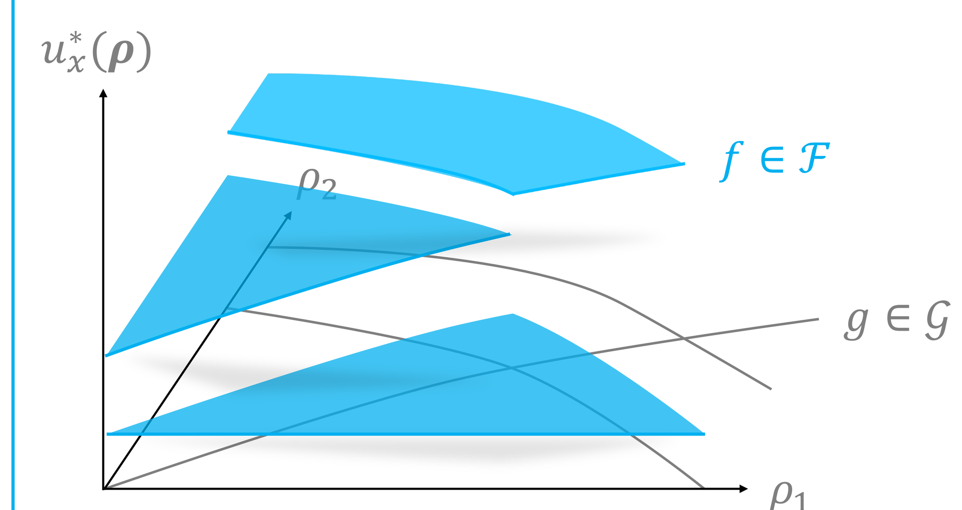
This is where **dual functions** come in handy!
$u_x^*(\boldsymbol{\rho})$ = utility as function of parameters
$u_x^*(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$
$\mathcal{U}^* = \{u_x^* : \mathbb{R}^d \to \mathbb{R} \mid x \in \mathcal{X}\}$ **"Dual" function class**

Across algorithm configuration, ubiquitously, the duals are ***piecewise-structured***

Theorem
With high probability, for all $\boldsymbol{\rho}$:

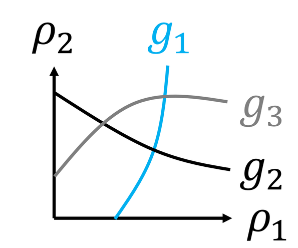
|**Average** utility on training set − **expected** utility| $= \tilde{O}\left(\sqrt{\frac{\mathcal{P}\text{-dim}(\mathcal{F}^*) + \text{VC-dim}(\mathcal{G}^*)\ln k}{N}}\right)$

\# boundary functions (k); Training set size (N)



Lemma: Given k boundaries, how many sign patterns do they make?

$$\left|\left\{\begin{pmatrix} g_1(\boldsymbol{\rho}) \\ \vdots \\ g_k(\boldsymbol{\rho}) \end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq (ek)^{\text{VCdim}(\mathcal{G}^*)}$$



Proof idea: Transition to dual and apply Sauer's lemma: for any $\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_k$

$$\left|\left\{\begin{pmatrix} g(\boldsymbol{\rho}_1) \\ \vdots \\ g(\boldsymbol{\rho}_k) \end{pmatrix} : g \in \mathcal{G}\right\}\right| \leq (ek)^{\text{VCdim}(\mathcal{G})}.$$

Example application: Sequence alignment

With high probability, for any $\boldsymbol{\rho} \in \mathbb{R}^3$,

|**avg** utility on training set − **expected** utility| $= \tilde{O}\left(\sqrt{\frac{\ln(\text{seq. length})}{N}}\right)$

Distance between algorithm's output and ground-truth alignment