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

Ellen Vitercik
UC Berkeley

Purdue University, Theory Seminar


Nina Balcan


Dan DeBlasio


Travis Dick


Carl Kingsford


Tuomas Sandholm

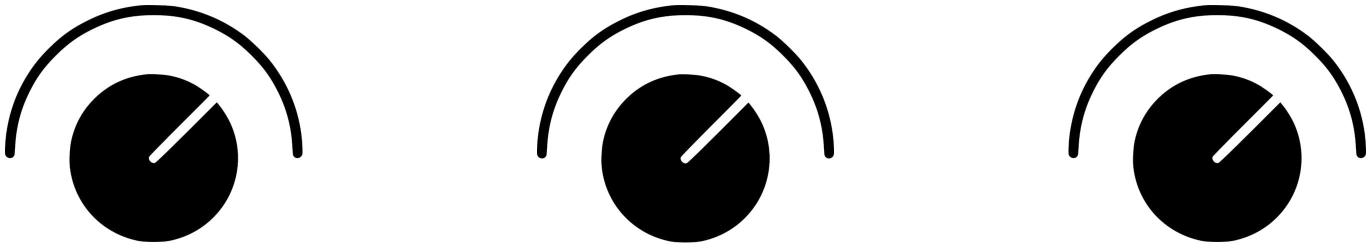
Appeared in STOC'21

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**



Data-driven algorithm design

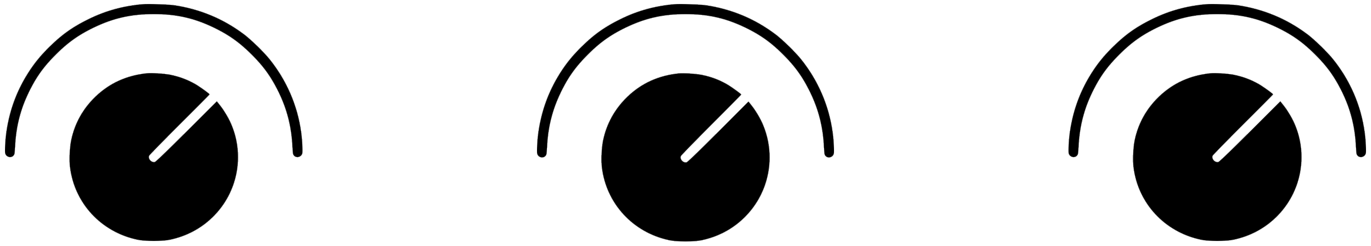
Goal: Automate algorithm configuration via machine learning

Algorithmically find good parameter settings

using a **set of “typical” inputs** from application at hand

Training set

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



Example: Sequence alignment

Goal: Line up pairs of strings

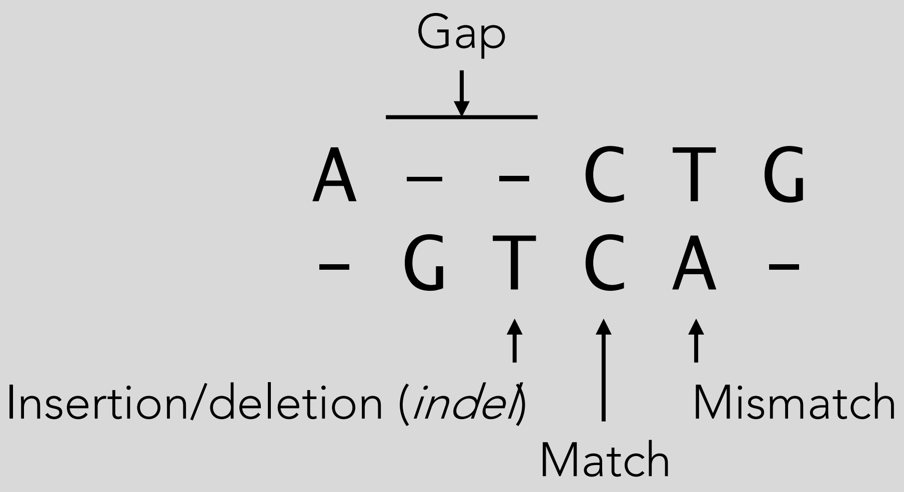
Applications: Biology, natural language processing, etc.

Sequence alignment algorithms

Input: Two sequences S and S'

Output: Alignment of S and S'

$$S = \mathrm{A\ C\ T\ G}$$
$$S' = \mathrm{G\ T\ C\ A}$$



Sequence alignment algorithms

Standard algorithm with parameters $\rho_1, \rho_2, \rho_3 \geq 0$:

Return alignment maximizing:

$$(\#\text{ matches}) - \rho_1 \cdot (\#\text{ mismatches}) - \rho_2 \cdot (\#\text{ indels}) - \rho_3 \cdot (\#\text{ gaps})$$

$S = $ A C T G

$S' = $ G T C A

Gap

```
A - - C T G
- G T C A -
```

Insertion/deletion (*indel*)

Match

Mismatch

Sequence alignment algorithms

Can sometimes access **ground-truth, reference** alignment

E.g., in computational biology: Bahr et al., Nucleic Acids Res.'01; Raghava et al., BMC Bioinformatics '03; Edgar, Nucleic Acids Res.'04; Walle et al., Bioinformatics'04

Requires extensive manual alignments

…rather just run parameterized algorithm

How to tune algorithm's parameters?

"There is ***considerable disagreement*** *among molecular biologists about the* ***correct choice****"* [Gusfield et al. '94]

```
A - - C T G
- G T C A -
```

Sequence alignment algorithms

```
-GRTCPKPDDLPFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYPAKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLGNWSAMPSC-KA
```

Ground-truth alignment of protein sequences

Sequence alignment algorithms

```
-GRTCPKPDDLPFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYPAKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLGNWSAMPSC-KA
```

Ground-truth alignment of protein sequences

```
GRTCP---KPDDLPFSTVVPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDN-GFVNYPAKPTLYYK-DKATFGCHDGY-SLDGPEEIECTKLGNWS-AMPSCKA
```

Alignment by algorithm with **poorly-tuned** parameters

Sequence alignment algorithms

```
-GRTCPKPDDLPFSTVVP-LKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
E-VKCPFPSRPDNGFVNYPAKPTLYYKDKATFGCHDGYSLDGP-EEIECTKLGNWSAMPSC-KA
```

Ground-truth alignment of protein sequences

```
GRTCP---KPDDLPFSTVVPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDN-GFVNYPAKPTLYYK-DKATFGCHDGY-SLDGPEEIECTKLGNWS-AMPSCKA
```

Alignment by algorithm with **poorly-tuned** parameters

```
GRTCPKPDDLPFSTV-VPLKTFYEPGEEITYSCKPGYVSRGGMRKFICPLTGLWPINTLKCTP
EVKCPFPSRPDNGFVNYPAKPTLYYKDKATFGCHDGY-SLDGPEEIECTKLGNWSA-MPSCKA
```

Alignment by algorithm with **well-tuned** parameters

Automated parameter tuning procedure

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs

Sequence S_1 Sequence S_1' Reference alignment A_1	Sequence S_2 Sequence S_2' Reference alignment A_2	…

3. Find parameters with good performance on average over T

Runtime, solution quality, etc.

Automated parameter tuning procedure

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs

Sequence S_1
Sequence S_1'
Reference alignment A_1

Sequence S_2
Sequence S_2'
Reference alignment A_2

…

3. Find parameters with good performance on average over T

Output alignment is close to reference alignment

Automated parameter tuning procedure

1. Fix parameterized algorithm
2. Receive training set T of "typical" inputs

Sequence S_1
Sequence S_1'
Reference alignment A_1

Sequence S_2
Sequence S_2'
Reference alignment A_2

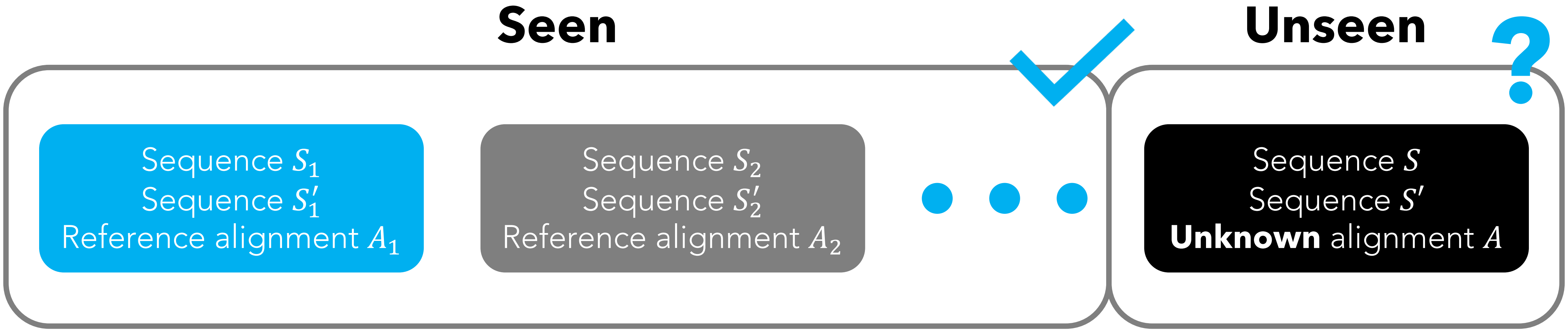
…

3. Find parameters with good performance on average over T

Key question (focus of talk):

Will those parameters have good **future** performance?

Automated parameter tuning procedure



Key question (focus of talk):

Will those parameters have good **future** performance?

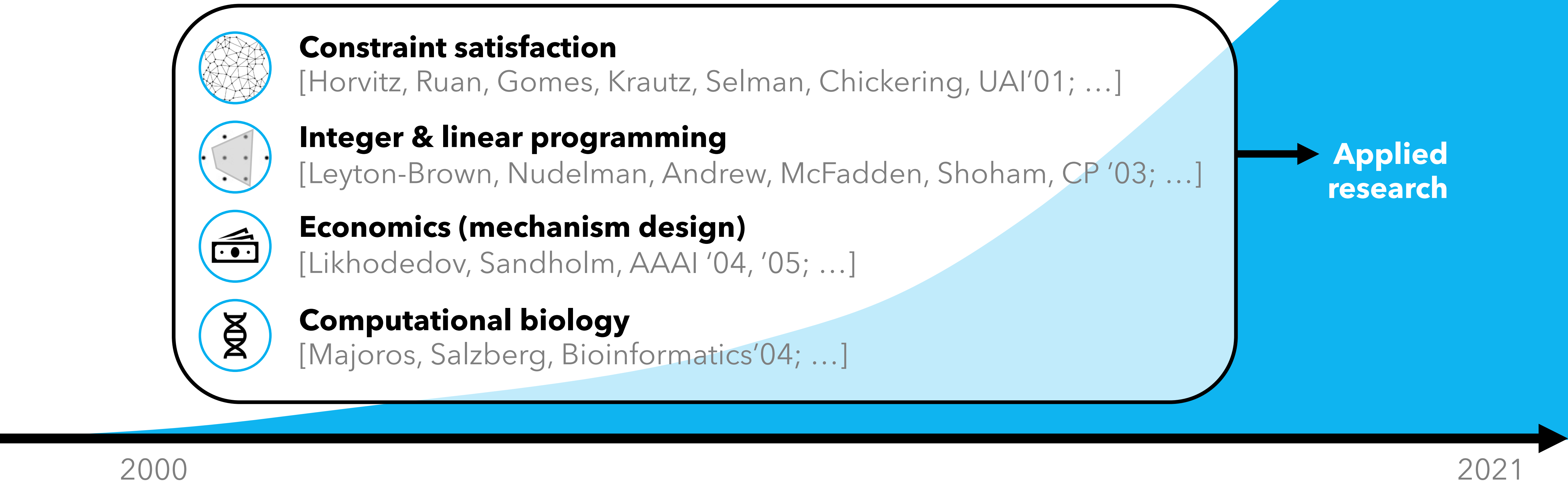
Existing research

Constraint satisfaction
[Horvitz, Ruan, Gomes, Krautz, Selman, Chickering, UAI’01; …]

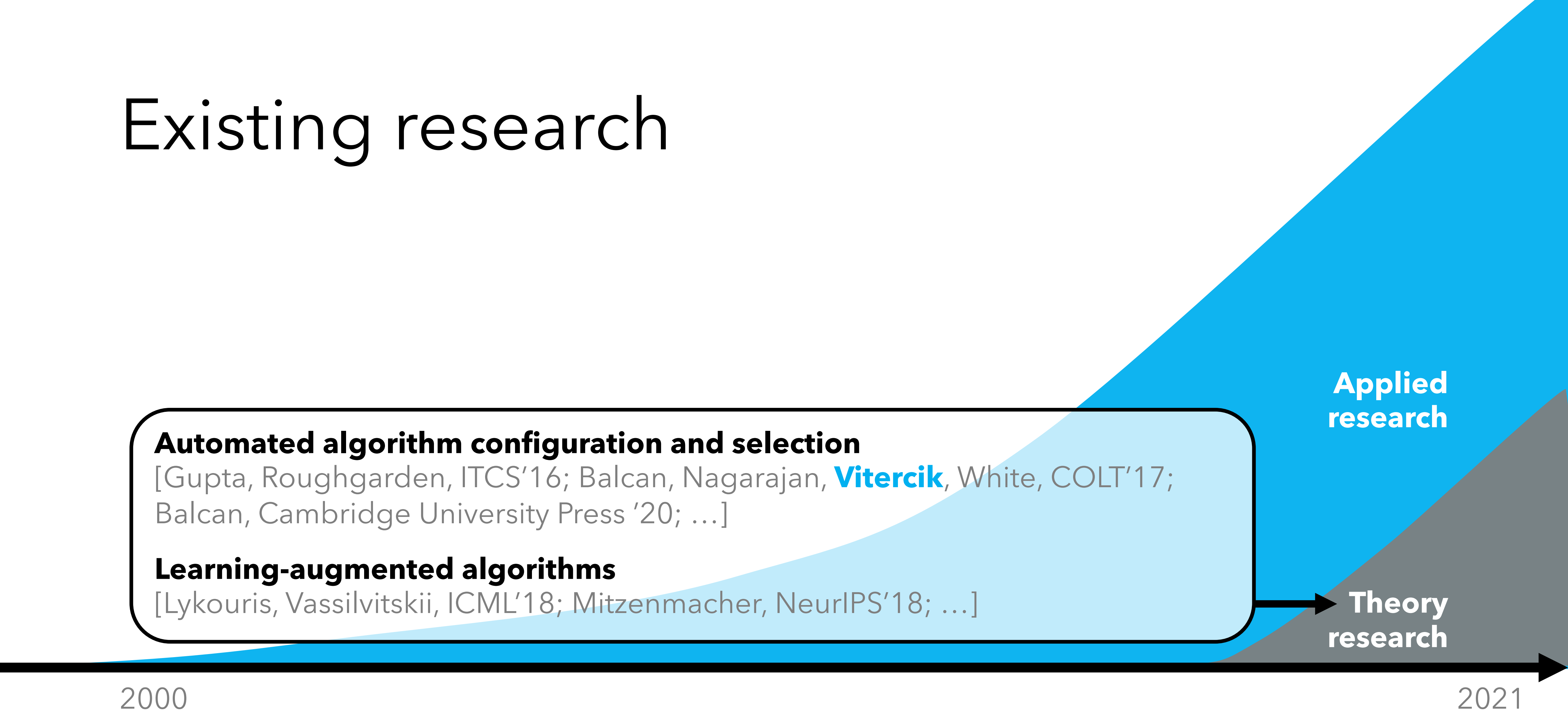
Integer & linear programming
[Leyton-Brown, Nudelman, Andrew, McFadden, Shoham, CP ’03; …]

Economics (mechanism design)
[Likhodedov, Sandholm, AAAI ’04, ’05; …]

Computational biology
[Majoros, Salzberg, Bioinformatics’04; …]



Existing research



Automated algorithm configuration and selection
[Gupta, Roughgarden, ITCS'16; Balcan, Nagarajan, **Vitercik**, White, COLT'17; Balcan, Cambridge University Press '20; …]

Learning-augmented algorithms
[Lykouris, Vassilvitskii, ICML'18; Mitzenmacher, NeurIPS'18; …]

Theoretical guarantees are needed to build firm foundations

This talk: Main result

Key question **(focus of talk):**

Good performance on **average** over **training set** implies good **future** performance?

Answer this question for any parameterized algorithm where:

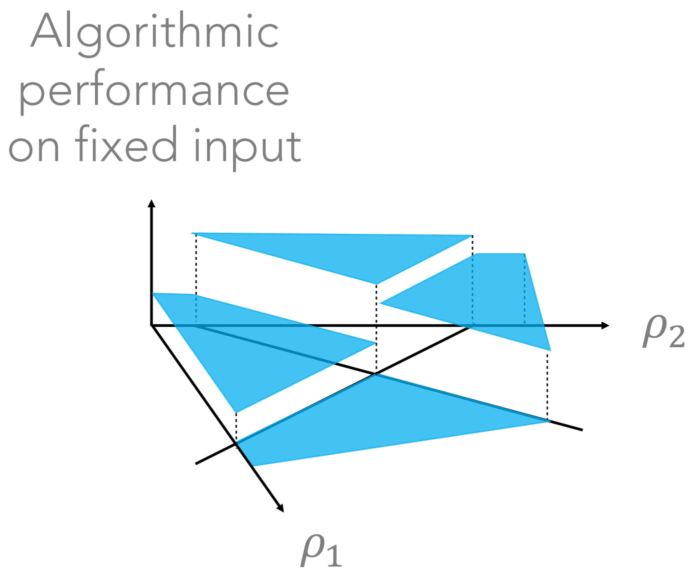
Performance is **piecewise-structured** function of parameters

Piecewise constant, linear, quadratic, …

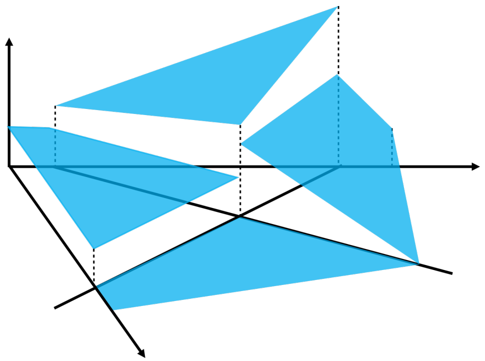
This talk: Main result

Performance is **piecewise-structured** function of parameters

Piecewise constant, linear, quadratic, …



Piecewise constant



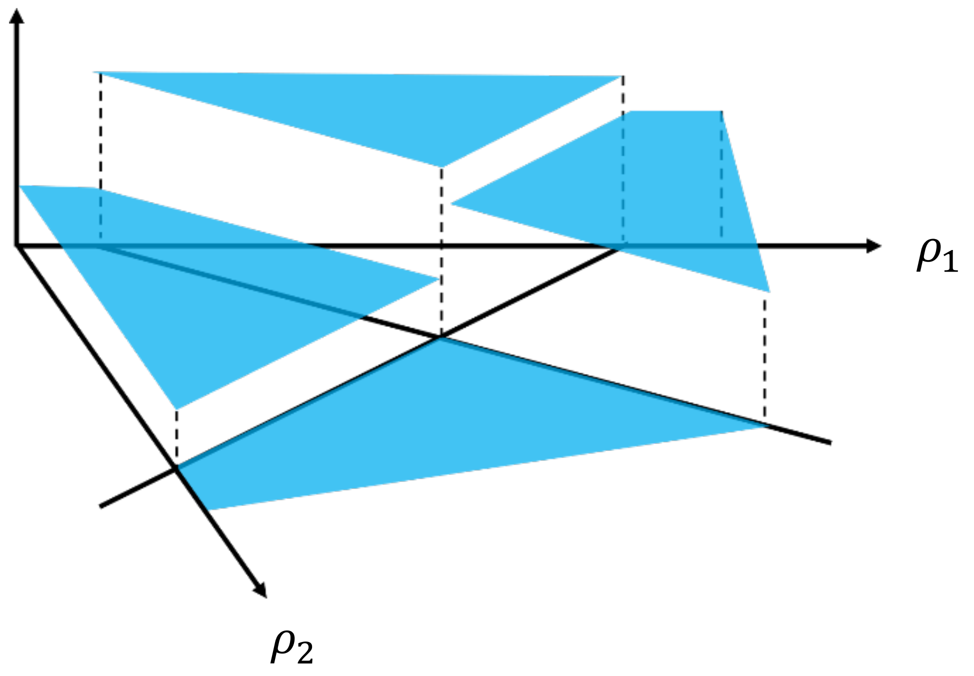
Piecewise linear

Piecewise …

Example: Sequence alignment

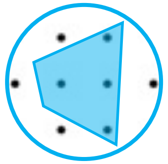
Distance between **algorithm's output** given S, S'

and **ground-truth** alignment is p-wise constant



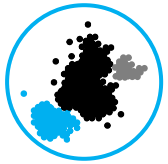
Piecewise structure

Piecewise structure unifies **seemingly disparate** problems:



Integer programming

Balcan, Dick, Sandholm, **V**, ICML'18
Balcan, Nagarajan, **V**, White, COLT'17

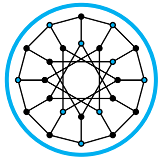


Clustering

Balcan, Nagarajan, **V**, White, COLT'17
Balcan, Dick, White, NeurIPS'18
Balcan, Dick, Lang, ICLR'20

Computational biology

Balcan, DeBlasio, Dick, Kingsford, Sandholm, **V**, STOC'21



Greedy algorithms

Gupta, Roughgarden, ITCS'16

Mechanism configuration

Balcan, Sandholm, **V**, EC'18

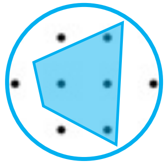
Online configuration [Gupta, Roughgarden, ITCS'16, Cohen-Addad and Kanade, AISTATS'17]

Exploited piecewise-Lipschitz structure to provide regret bounds

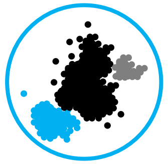
[Balcan, Dick, **V**, FOCS'18; Balcan, Dick, Pegden, UAI'20; Balcan, Dick, Sharma, AISTATS'20]

Piecewise structure

Piecewise structure unifies **seemingly disparate** problems:

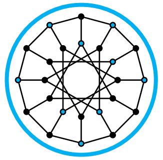


Integer programming
Balcan, Dick, Sandholm, **V**, ICML'18
Balcan, Nagarajan, **V**, White, COLT'17



Clustering
Balcan, Nagarajan, **V**, White, COLT'17
Balcan, Dick, White, NeurIPS'18
Balcan, Dick, Lang, ICLR'20

Computational biology
Balcan, DeBlasio, Dick, Kingsford, Sandholm, **V**, STOC'21



Greedy algorithms
Gupta, Roughgarden, ITCS'16

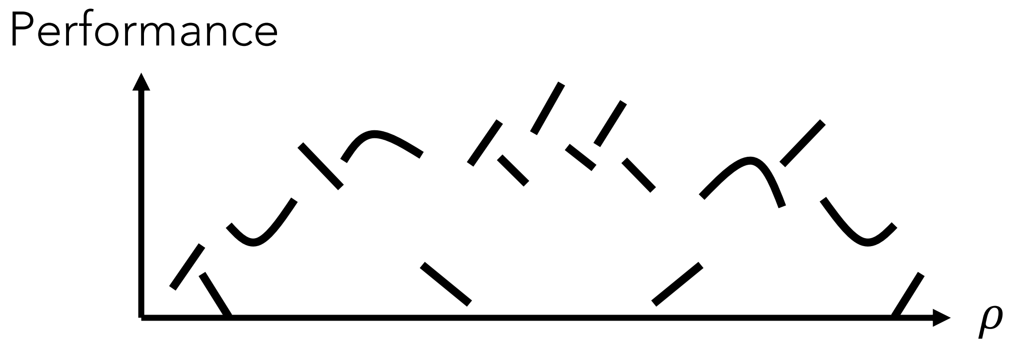
Mechanism configuration
Balcan, Sandholm, **V**, EC'18

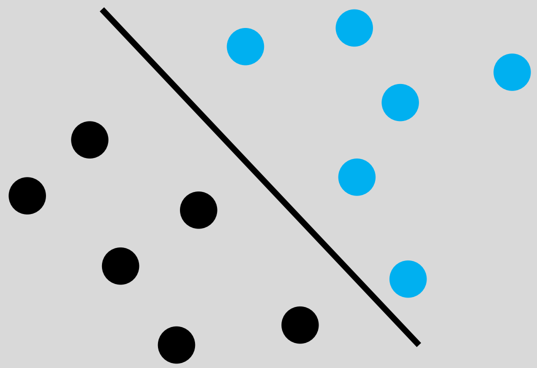
Ties to a long line of research on machine learning for **revenue maximization**
Likhodedov, Sandholm, AAAI'04, '05; Balcan, Blum, Hartline, Mansour, FOCS'05; Elkind, SODA'07; Cole, Roughgarden, STOC'14; Mohri, Medina, ICML'14; Devanur, Huang, Psomas, STOC'16; …

Primary challenge:

Algorithmic performance is a **volatile** function of parameters

Complex connection between parameters and performance





For well-understood functions in machine learning theory:

Simple connection between function parameters and value

Outline

1. Introduction
2. **Model and problem formulation**
3. Our guarantees
4. Conclusion and future directions

Model

$\mathbb{R}^d$: Set of all parameters

$\mathcal{X}$: Set of all inputs

Example: Sequence alignment

$\mathbb{R}^3$: Set of alignment algorithm parameters

$\mathcal{X}$: Set of sequence pairs

$$
\begin{aligned}
S &= \mathrm{A}\ \ \mathrm{C}\ \ \mathrm{T}\ \ \mathrm{G} \\
S' &= \mathrm{G}\ \ \mathrm{T}\ \ \mathrm{C}\ \ \mathrm{A}
\end{aligned}
$$

One sequence pair $x = (S, S') \in \mathcal{X}$

Algorithmic performance

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

E.g., runtime, solution quality, distance to ground truth, …

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

Can be generalized to $u_{\boldsymbol{\rho}}(x) \in [-H, H]$

Algorithmic performance

$u_{\boldsymbol{\rho}}(x)$ = distance between algorithm's output and ground-truth

S = A C T G
S' = G T C A

A – – C T G
– G T C A –

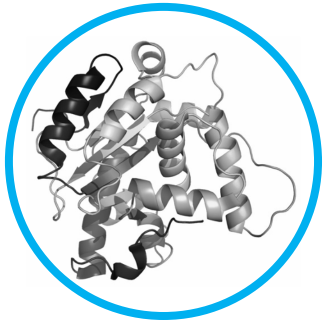
One sequence pair $x = (S, S') \in \mathcal{X}$

Model

Standard assumption: Unknown distribution $\mathcal{D}$ over inputs

Distribution models specific application domain at hand

E.g., distribution over pairs of DNA strands



E.g., distribution over pairs of protein sequences

Generalization bounds

Key question: For any parameter setting $\boldsymbol{\rho}$,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \ldots, x_N \sim \mathcal{D}$, for any $\boldsymbol{\rho}$,

Generalization bounds

Key question: For any parameter setting $\boldsymbol{\rho}$,
is **average** utility on training set close to **expected** utility?

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

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

Generalization bounds

Key question: For any parameter setting $\boldsymbol{\rho}$,
is **average** utility on training set close to **expected** utility?

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

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

Generalization bounds

Key question: For any parameter setting $\boldsymbol{\rho}$,
is **average** utility on training set close to **expected** utility?

Formally: Given samples $x_1, \ldots, x_N \sim \mathcal{D}$, for any $\boldsymbol{\rho}$,

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

Good **average empirical** utility ➡ Good **expected** utility

Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
   a. **Example of piecewise-structured utility function**
   b. Piecewise-structured functions more formally
   c. Main theorem
   d. Application: Sequence alignment
4. Conclusion and future directions

Sequence alignment algorithms

Standard algorithm with parameters $\rho_1, \rho_2, \rho_3 \geq 0$:

Return alignment maximizing:

$$(\#\text{ matches}) - \rho_1 \cdot (\#\text{ mismatches}) - \rho_2 \cdot (\#\text{ indels}) - \rho_3 \cdot (\#\text{ gaps})$$

$S\ =$ A C T G
$S' =$ G T C A

Gap

```
A - - C T G
- G T C A -
```

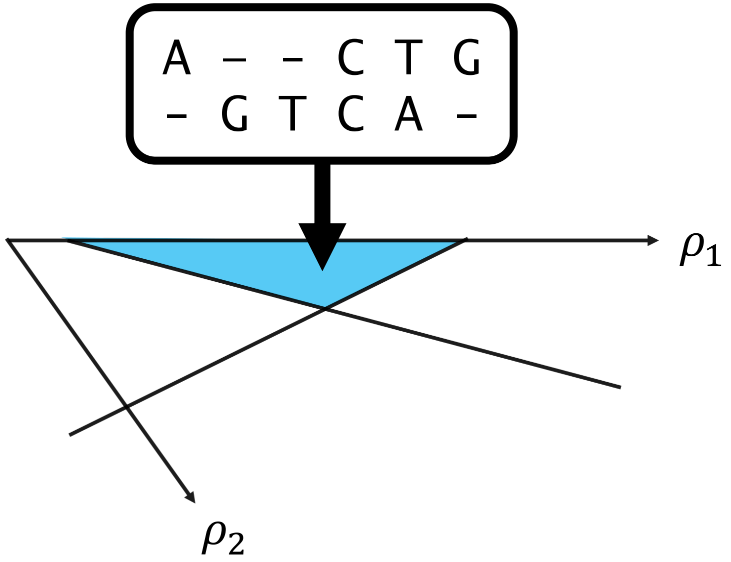
Mismatch
Match
Insertion/deletion (*indel*)

Sequence alignment algorithms

Lemma:

For any pair S, S', there's a small partition of $\mathbb{R}^3$ s.t. in any region, algorithm's output is fixed across all parameters in region

S = A C T G
S' = G T C A

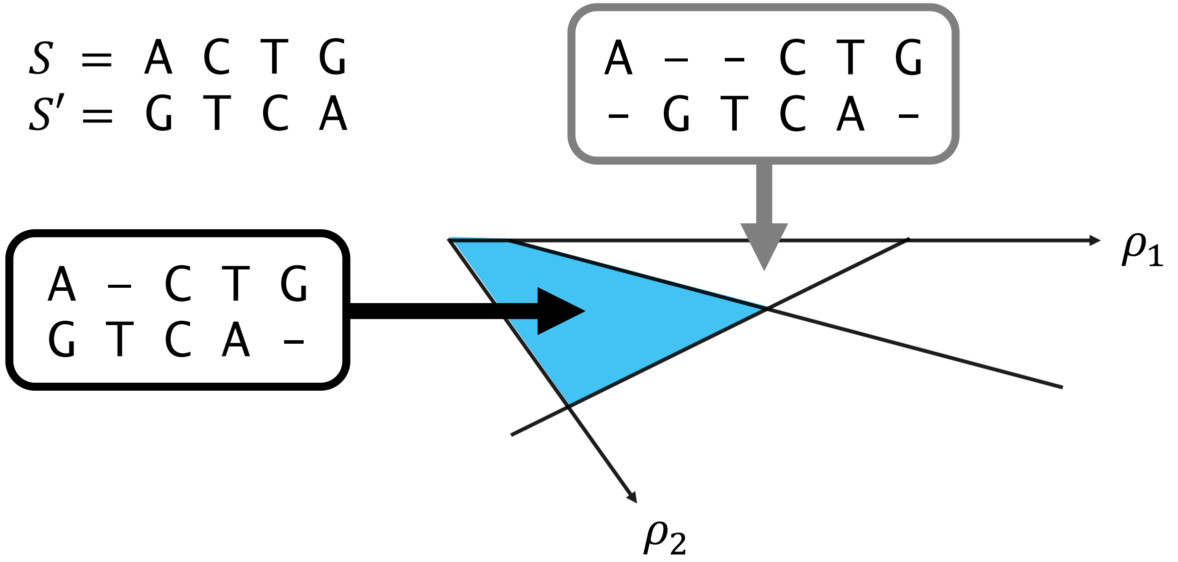


Gusfield et al., Algorithmica '94; Fernández-Baca et al., J. of Discrete Alg. '04

Sequence alignment algorithms

Lemma:

For any pair S, S', there's a small partition of $\mathbb{R}^3$ s.t. in any region, algorithm's output is fixed across all parameters in region



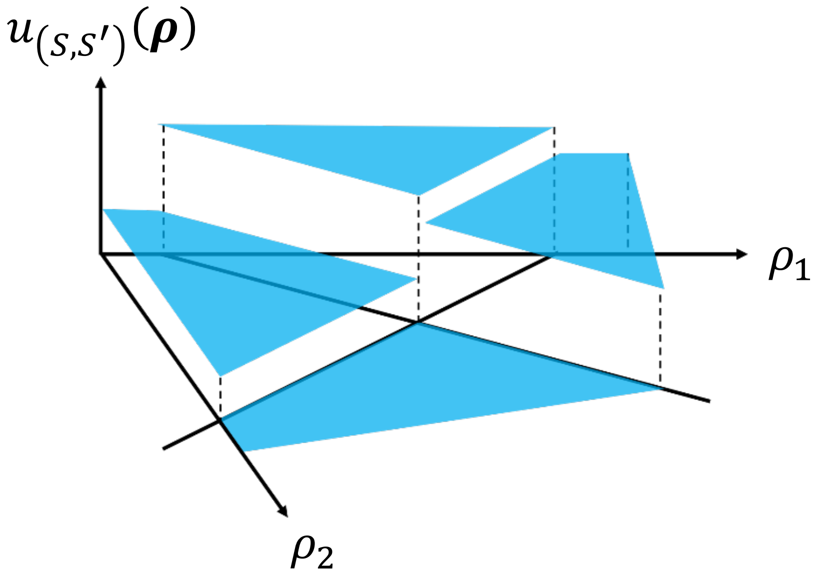
Gusfield et al., Algorithmica '94; Fernández-Baca et al., J. of Discrete Alg. '04

Piecewise-constant utility function

Corollary:

Utility is piecewise constant function of parameters

Distance between algorithm's output and ground-truth alignment



Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
   a. Example of piecewise-structured utility function
   b. **Piecewise-structured functions more formally**
   c. Main theorem
   d. Application: Sequence alignment
4. Conclusion and future directions

Primal & dual classes

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
$\mathcal{U} \; = \left\{u_{\boldsymbol{\rho}}: \mathcal{X} \to \mathbb{R} \,\middle|\, \boldsymbol{\rho} \in \mathbb{R}^d\right\}$ **"Primal" function class**

Typically, prove guarantees by bounding ***complexity*** of $\mathcal{U}$

VC dimension, pseudo-dimension, Rademacher complexity, …

Primal & dual classes

$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
$\mathcal{U} = \left\{u_{\boldsymbol{\rho}} : \mathcal{X} \to \mathbb{R} \,\middle|\, \boldsymbol{\rho} \in \mathbb{R}^d\right\}$ **"Primal" function class**

Typically, prove guarantees by bounding ***complexity*** of $\mathcal{U}$

Challenge: $\mathcal{U}$ is gnarly

E.g., in sequence alignment:

- Each domain element is a pair of sequences
- Unclear how to plot or visualize functions $u_{\boldsymbol{\rho}}$
- No obvious notions of Lipschitz continuity or smoothness to rely on

Primal & dual classes

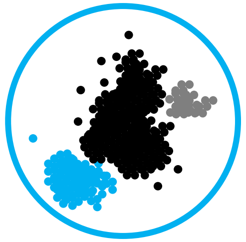
$u_{\boldsymbol{\rho}}(x)$ = utility of algorithm parameterized by $\boldsymbol{\rho} \in \mathbb{R}^d$ on input x
$\mathcal{U} \;\; = \left\{u_{\boldsymbol{\rho}}: \mathcal{X} \to \mathbb{R} \,\middle|\, \boldsymbol{\rho} \in \mathbb{R}^d\right\}$ **"Primal" function class**

$u_x^*(\boldsymbol{\rho})$ = utility as function of parameters
$u_x^*(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$
$\mathcal{U}^* = \left\{u_x^*: \mathbb{R}^d \to \mathbb{R} \,\middle|\, x \in \mathcal{X}\right\}$ **"Dual" function class**

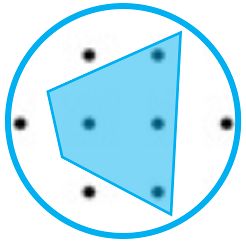
- Dual functions have simple, Euclidean domain
- Often have ample structure can use to bound complexity of $\mathcal{U}$

Piecewise-structured functions

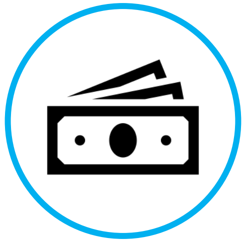
Dual functions $u_x^*: \mathbb{R}^d \to \mathbb{R}$ are **piecewise-structured**



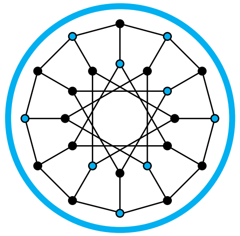
Clustering
algorithm
configuration



Integer
programming
algorithm
configuration



Selling
mechanism
configuration



Greedy
algorithm
configuration

Computational
biology
algorithm
configuration

Voting
mechanism
configuration

Piecewise-structured functions

Online algorithm configuration

Gupta, Roughgarden, ITCS'16, Cohen-Addad and Kanade, AISTATS'17

Problem instances arrive online, not necessarily i.i.d.

Exploited piecewise-Lipschitz structure to give regret bounds

Balcan, Dick, **V**, FOCS'18; Balcan, Dick, Pegden, UAI'20; Balcan, Dick, Sharma, AISTATS'20

Regret bounds require additional structure:

Boundaries between pieces don't concentrate

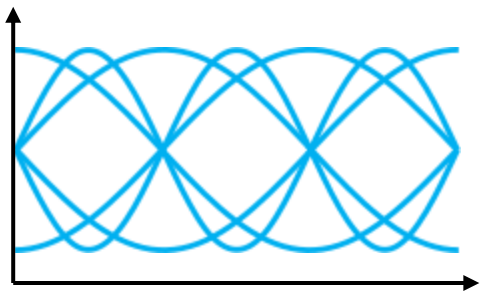
Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
    a. Example of piecewise-structured utility function
    b. Piecewise-structured functions more formally
    c. **Main theorem**
    d. Application: Sequence alignment
4. Conclusion and future directions

Intrinsic complexity

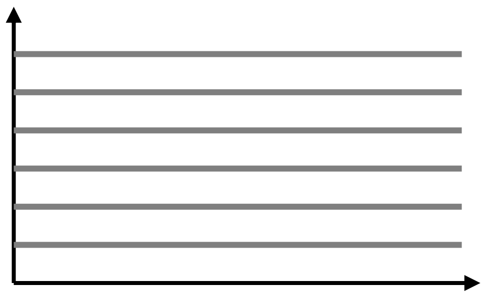
"Intrinsic complexity" of function class $\mathcal{G}$

- Measures how well functions in $\mathcal{G}$ fit complex patterns
- Specific ways to quantify "intrinsic complexity":
  - VC dimension
  - Pseudo-dimension



More complex

Less complex

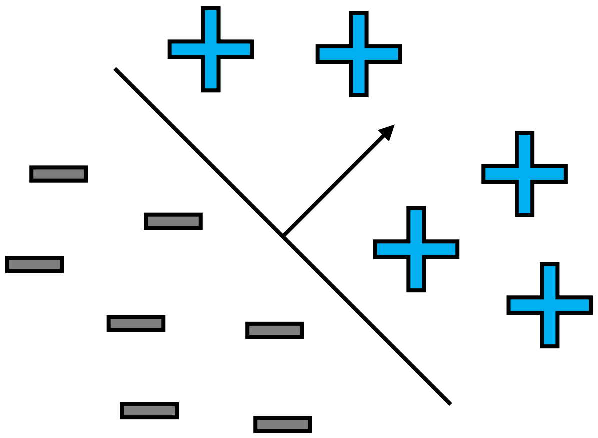


VC dimension

Complexity measure for binary-valued function classes $\mathcal{F}$

(Classes of functions $f: \mathcal{Y} \to \{-1,1\}$)

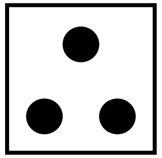
E.g., linear separators



VC dimension of $\mathcal{F}$

Size of the largest set $\mathcal{S} \subseteq \mathcal{Y}$
that can be labeled in all $2^{|\mathcal{S}|}$ ways by functions in $\mathcal{F}$

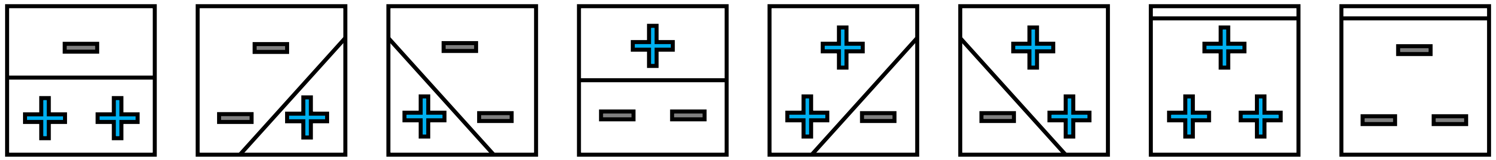
Example: $\mathcal{F}$ = Linear separators in $\mathbb{R}^2$ VCdim($\mathcal{F}$) ≥ 3



VC dimension of $\mathcal{F}$

Size of the largest set $\mathcal{S} \subseteq \mathcal{Y}$
that can be labeled in all $2^{|\mathcal{S}|}$ ways by functions in $\mathcal{F}$

Example: $\mathcal{F}$ = Linear separators in $\mathbb{R}^2$ VCdim($\mathcal{F}$) ≥ 3



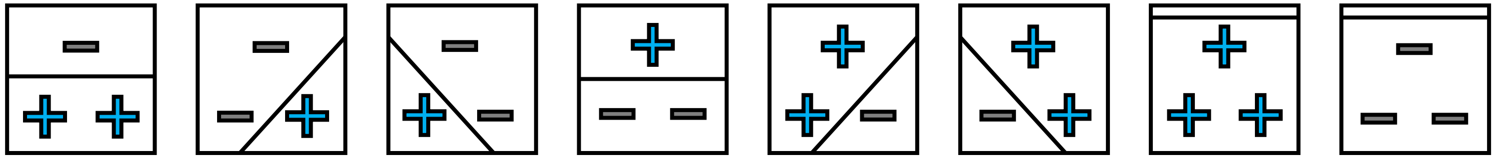
VCdim($\mathcal{F}$) ≤ 3

VCdim({Linear separators in $\mathbb{R}^d$}) $= d + 1$

Sample complexity using VC dimension

Theorem [Vapnik, Chervonenkis, '71]: For any dist. $\mathcal{D}$ over $\mathcal{Y}$, given $N = \tilde{O}\left(\frac{\mathrm{VCdim}(\mathcal{F})}{\epsilon^2}\right)$ samples $y_1, \ldots, y_N \sim \mathcal{D}$, WHP $\forall f \in \mathcal{F}$,

$$\left|\frac{1}{N}\sum_{i=1}^{N} f(y_i) - \mathbb{E}_{y\sim\mathcal{D}}[f(y)]\right| \leq \epsilon$$

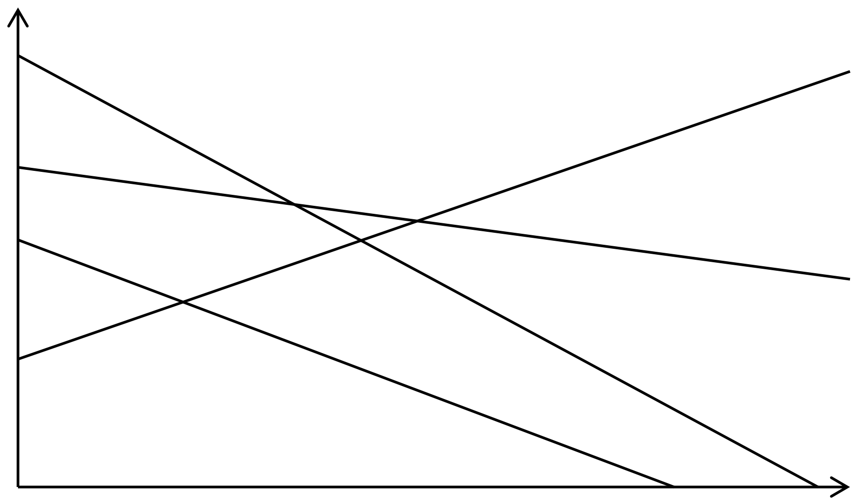


Pseudo-dimension

Complexity measure for real-valued function classes $\mathcal{G}$

(Classes of functions $g: \mathcal{Y} \to [0,1]$)

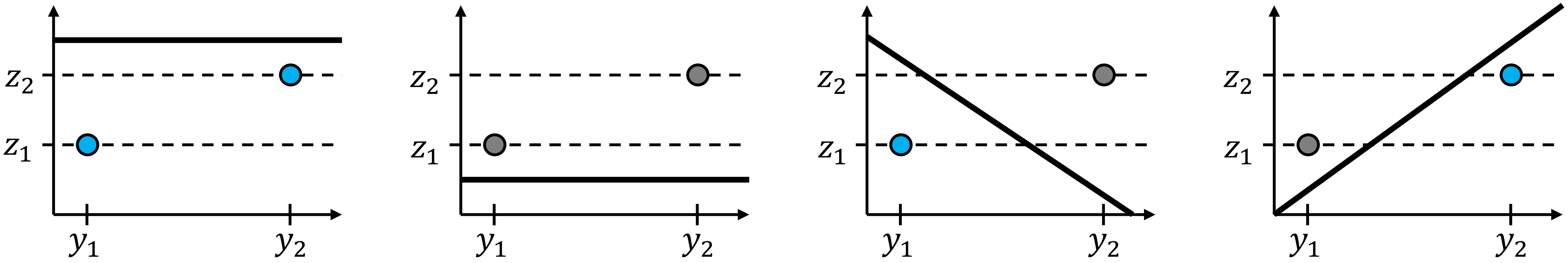
E.g., affine functions



Pseudo-dimension of $\mathcal{G}$

Size of the largest set $\{y_1, \dots, y_N\} \subseteq \mathcal{Y}$ s.t.:
for some *targets* $z_1, \dots, z_N \in \mathbb{R}$,
all 2^N above/below patterns achieved by functions in $\mathcal{G}$

Example: $\mathcal{G}$ = Affine functions in $\mathbb{R}$ $\qquad$ $\mathrm{Pdim}(\mathcal{G}) \geq 2$

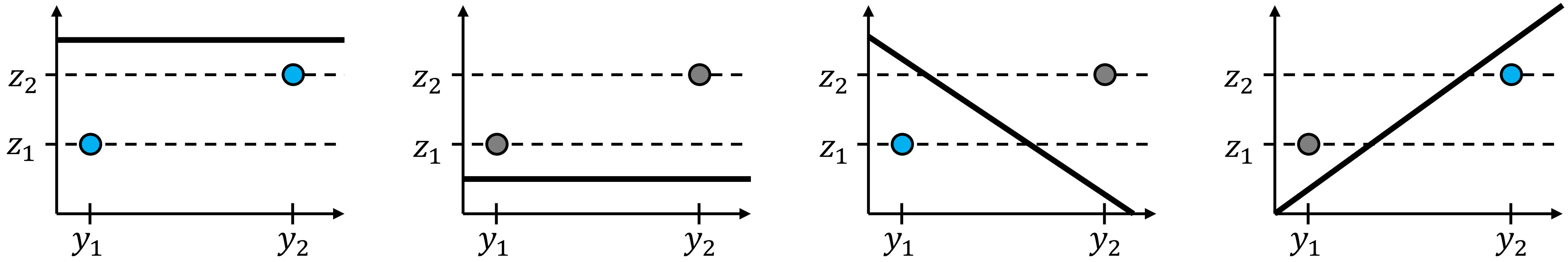


Sample complexity using pseudo-dim

Theorem [Pollard, '84]: For any dist. $\mathcal{D}$ over $\mathcal{Y}$,

given $N = \tilde{O}\left(\frac{\mathrm{Pdim}(\mathcal{G})}{\epsilon^2}\right)$ samples $y_1, \dots, y_N \sim \mathcal{D}$, WHP $\forall g \in \mathcal{G}$,

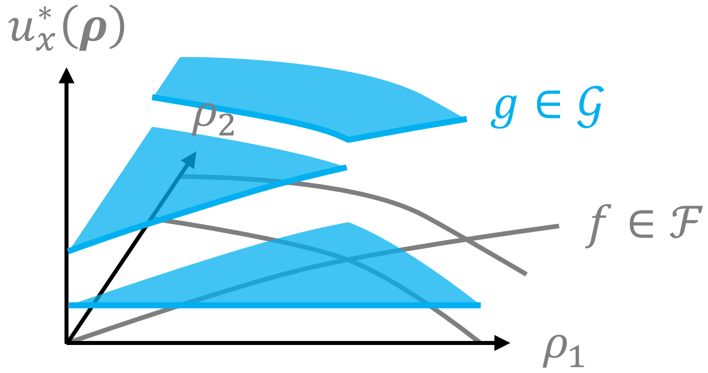
$$\left|\frac{1}{N}\sum_{i=1}^{N} g(y_i) - \mathbb{E}_{y\sim\mathcal{D}}[g(y)]\right| \le \epsilon$$



Main result (informal)

Boundary functions $f_1, \dots, f_k \in \mathcal{F}$ partition $\mathbb{R}^d$ s.t. in each region,
$u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

Training set of size $\tilde{O}\left(\frac{1}{\epsilon^2}\left(\mathrm{VCdim}(\mathcal{F}^*) + \mathrm{Pdim}(\mathcal{G}^*)\right)\log k\right)$ implies
WHP $\forall\boldsymbol{\rho}$, |**avg** utility over training set – **exp** utility| $\leq \epsilon$



Main result (informal)

Boundary functions $f_1, \dots, f_k \in \mathcal{F}$ partition $\mathbb{R}^d$ s.t. in each region,
$u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

Training set of size $\tilde{O}\left(\frac{1}{\epsilon^2}\left(\mathrm{VCdim}(\mathcal{F}^*) + \mathrm{Pdim}(\mathcal{G}^*)\right)\log k\right)$ implies
WHP $\forall \boldsymbol{\rho}$, |**avg** utility over training set - **exp** utility| $\leq \epsilon$

$\mathcal{F}, \mathcal{G}$ are typically very well structured

- $\mathcal{G}$ = set of all **constant** functions $\Rightarrow$ $\mathrm{Pdim}(\mathcal{G}^*) = O(1)$
- $\mathcal{G}$ = set of all **linear** functions in $\mathbb{R}^d$ $\Rightarrow$ $\mathrm{Pdim}(\mathcal{G}^*) = O(d)$

Main result (informal)

Boundary functions $f_1, \ldots, f_k \in \mathcal{F}$ partition $\mathbb{R}^d$ s.t. in each region,
$u_x^*(\boldsymbol{\rho}) = g(\boldsymbol{\rho})$ for some $g \in \mathcal{G}$.

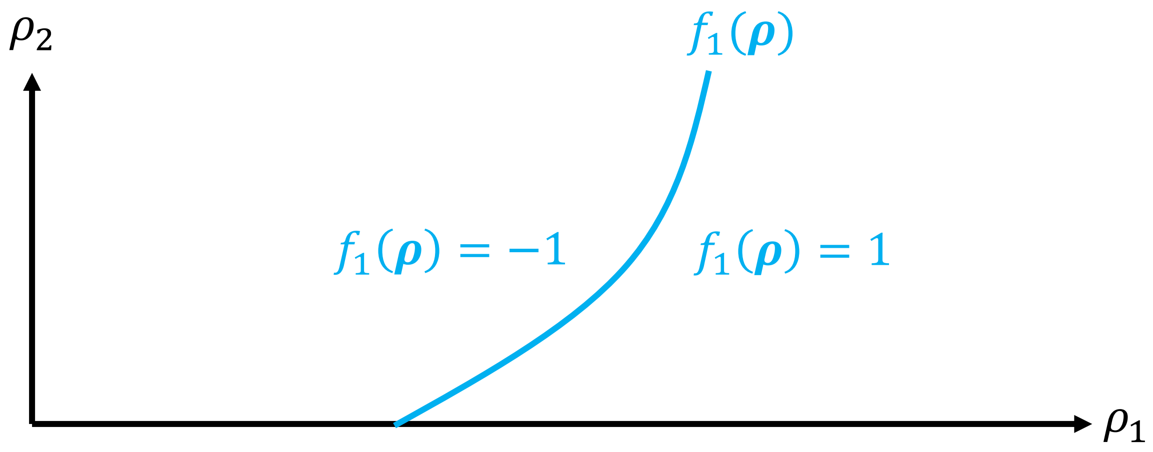
Theorem:

$$\mathrm{Pdim}(\mathcal{U}) = \tilde{O}\big((\mathrm{VCdim}(\mathcal{F}^*) + \mathrm{Pdim}(\mathcal{G}^*)) \log k\big)$$

Primal function class $\mathcal{U} = \{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$

Key lemma

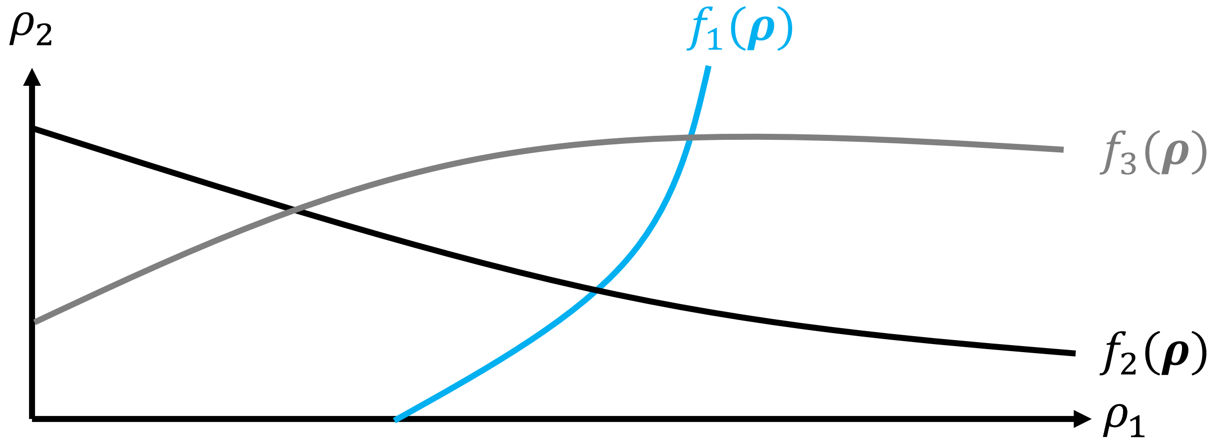
Each boundary function $f: \mathbb{R}^d \to \{-1,1\}$ splits $\mathbb{R}^d$ into 2 regions



Key lemma

Given D boundaries, how many sign patterns do they make?

$$\left|\left\{\begin{pmatrix} f_1(\boldsymbol{\rho}) \\ \vdots \\ f_D(\boldsymbol{\rho}) \end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq \text{?}$$



Key lemma

Given D boundaries, how many sign patterns do they make?

$$\left|\left\{\begin{pmatrix} f_1(\boldsymbol{\rho}) \\ \vdots \\ f_D(\boldsymbol{\rho}) \end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq \textbf{?}$$

Note: Sauer's lemma tells us that for any D points $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_D \in \mathbb{R}^d$

$$\left|\left\{\begin{pmatrix} f(\boldsymbol{\rho}_1) \\ \vdots \\ f(\boldsymbol{\rho}_D) \end{pmatrix} : f \in \mathcal{F}\right\}\right| \leq (eD)^{\mathrm{VCdim}(\mathcal{F})}$$

This is where transitioning to the dual comes in handy!

Key lemma

Given D boundaries, how many sign patterns do they make?

$$\left|\left\{\begin{pmatrix} f_1(\boldsymbol{\rho}) \\ \vdots \\ f_D(\boldsymbol{\rho}) \end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq (eD)^{\mathrm{VCdim}(\mathcal{F}^*)}$$

Proof ideas

For any problem instances $x_1, \dots, x_N$ and *targets* $z_1, \dots, z_N \in \mathbb{R}$,

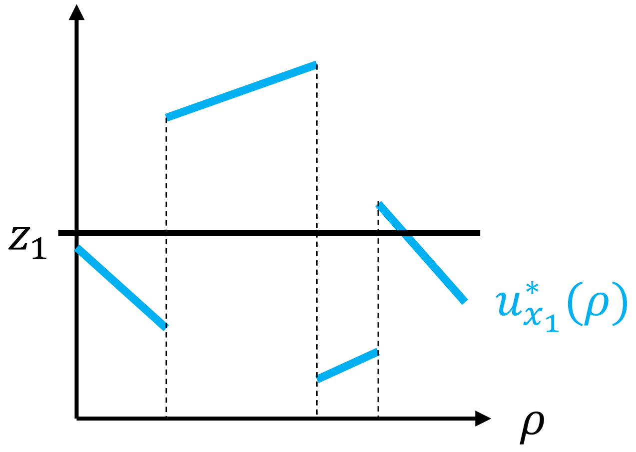
$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\big(u_{\boldsymbol{\rho}}(x_1) - z_1\big) \\ \vdots \\ \operatorname{sgn}\big(u_{\boldsymbol{\rho}}(x_N) - z_N\big)\end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq \textbf{?}$$

Switching to the dual functions,

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\big(u^*_{x_1}(\boldsymbol{\rho}) - z_1\big) \\ \vdots \\ \operatorname{sgn}\big(u^*_{x_N}(\boldsymbol{\rho}) - z_N\big)\end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq \textbf{?}$$

Proof ideas

$$\left|\left\{\begin{pmatrix} \operatorname{sgn}\left(u^*_{x_1}(\boldsymbol{\rho}) - z_1\right) \\ \vdots \\ \operatorname{sgn}\left(u^*_{x_N}(\boldsymbol{\rho}) - z_N\right) \end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d \right\}\right| \leq \textbf{?}$$

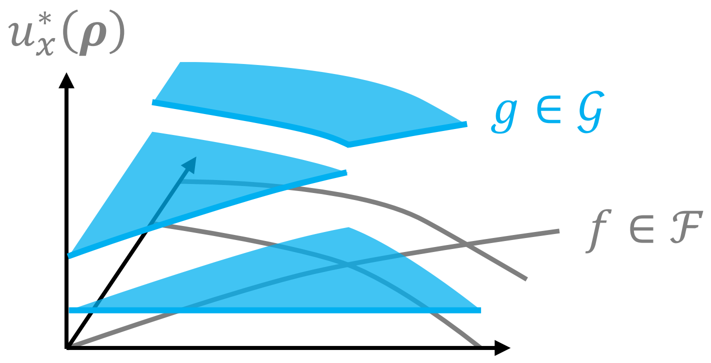


Proof ideas

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\left(u^*_{x_1}(\boldsymbol{\rho}) - z_1\right) \\ \vdots \\ \operatorname{sgn}\left(u^*_{x_N}(\boldsymbol{\rho}) - z_N\right)\end{pmatrix} : \boldsymbol{\rho} \in \mathbb{R}^d\right\}\right| \leq \text{?}$$

The duals $u^*_{x_1}, \ldots, u^*_{x_N}$ correspond to Nk boundary functions in $\mathcal{F}$

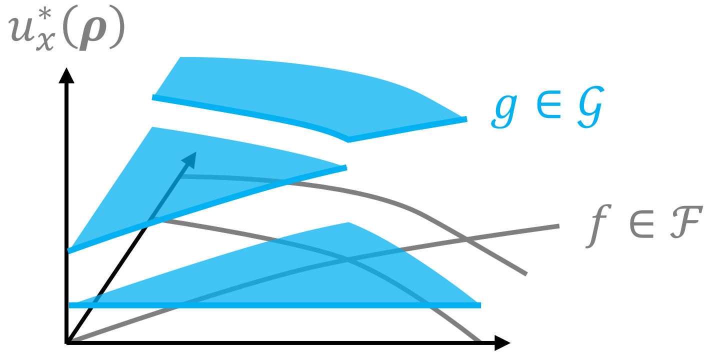
How many regions $R_1, \ldots, R_M$ in $\mathbb{R}^d$? $M \leq (eNk)^{\mathrm{VCdim}(\mathcal{F}^*)}$



Proof ideas

$$\left|\left\{\begin{pmatrix}\mathrm{sgn}\left(u^*_{x_1}(\boldsymbol{\rho}) - z_1\right)\\ \vdots \\ \mathrm{sgn}\left(u^*_{x_N}(\boldsymbol{\rho}) - z_N\right)\end{pmatrix} : \boldsymbol{\rho} \in R_j\right\}\right| \leq \text{?}$$

$\forall \boldsymbol{\rho} \in R_j$, duals are simultaneously structured: $u^*_{x_i}(\boldsymbol{\rho}) = g_i(\boldsymbol{\rho}), \forall i$



Proof ideas

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\left(u^*_{x_1}(\boldsymbol{\rho}) - z_1\right)\\ \vdots \\ \operatorname{sgn}\left(u^*_{x_N}(\boldsymbol{\rho}) - z_N\right)\end{pmatrix} : \boldsymbol{\rho} \in R_j\right\}\right| \leq \text{?}$$

$\forall \boldsymbol{\rho} \in R_j$, duals are simultaneously structured: $u^*_{x_i}(\boldsymbol{\rho}) = g_i(\boldsymbol{\rho}), \forall i$

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}(g_1(\boldsymbol{\rho}) - z_1)\\ \vdots \\ \operatorname{sgn}(g_N(\boldsymbol{\rho}) - z_N)\end{pmatrix} : \boldsymbol{\rho} \in R_j\right\}\right| \leq \text{?}$$

Proof ideas

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\left(u^*_{x_1}(\boldsymbol{\rho})-z_1\right)\\ \vdots \\ \operatorname{sgn}\left(u^*_{x_N}(\boldsymbol{\rho})-z_N\right)\end{pmatrix} : \boldsymbol{\rho}\in R_j\right\}\right| \leq \textbf{?}$$

$\forall \boldsymbol{\rho} \in R_j$, duals are simultaneously structured: $u^*_{x_i}(\boldsymbol{\rho}) = g_i(\boldsymbol{\rho}), \forall i$

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}(g_1(\boldsymbol{\rho})-z_1)\\ \vdots \\ \operatorname{sgn}(g_N(\boldsymbol{\rho})-z_N)\end{pmatrix} : \boldsymbol{\rho}\in R_j\right\}\right| \leq \underbrace{(eN)^{\mathrm{Pdim}(\mathcal{G}^*)}}_{\textbf{Follows from key lemma}}$$

Proof ideas

$$\left|\left\{\begin{pmatrix}\operatorname{sgn}\left(u^*_{x_1}(\boldsymbol{\rho}) - z_1\right)\\ \vdots \\ \operatorname{sgn}\left(u^*_{x_N}(\boldsymbol{\rho}) - z_N\right)\end{pmatrix} : \boldsymbol{\rho}\in\mathbb{R}^d\right\}\right|$$

$$\le \underbrace{(eNk)^{\mathrm{VCdim}(\mathcal{F}^*)}}_{\text{Number of regions}}\underbrace{(eN)^{\mathrm{Pdim}(\mathcal{G}^*)}}_{\text{Number of sign patterns within each region}}$$

Pdim($\mathcal{U}$) equals largest N s.t. $2^N \le (eNk)^{\mathrm{VCdim}(\mathcal{F}^*)}(eN)^{\mathrm{Pdim}(\mathcal{G}^*)}$,
so Pdim($\mathcal{U}$) $= \tilde{O}\left((\mathrm{VCdim}(\mathcal{F}^*) + \mathrm{Pdim}(\mathcal{G}^*))\log k\right)$

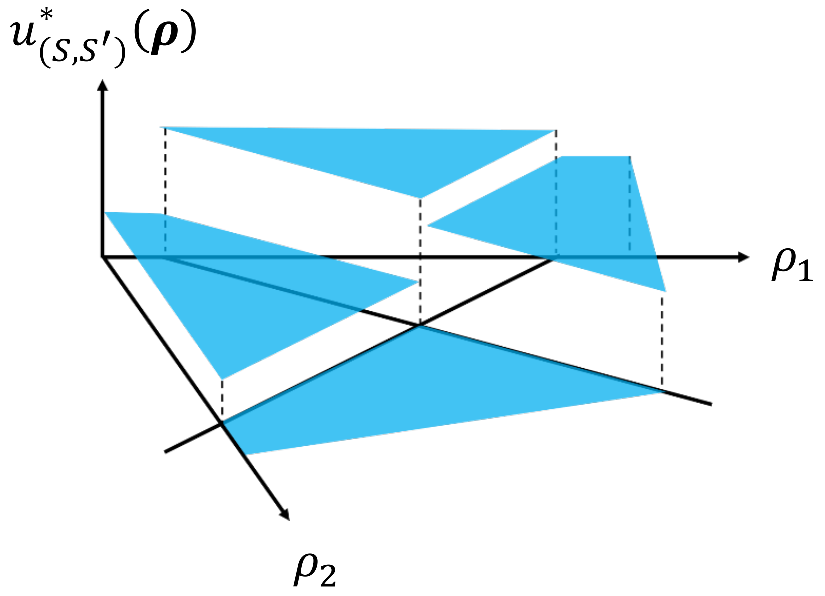
Outline

1. Introduction
2. Model and problem formulation
3. Our guarantees
    a. Example of piecewise-structured utility function
    b. Piecewise-structured functions more formally
    c. Main theorem
    d. **Application: Sequence alignment**
4. Conclusion and future directions

Piecewise constant dual functions

Lemma:

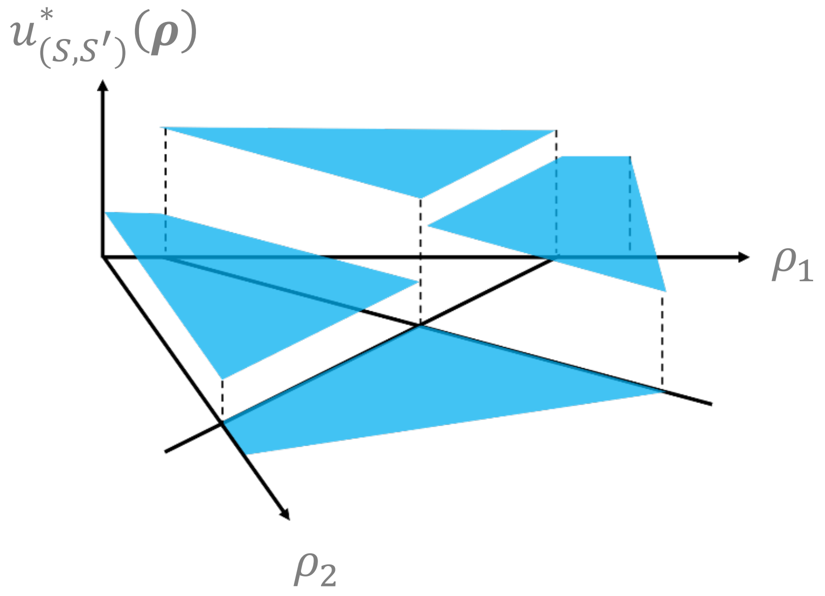
Utility is piecewise constant function of parameters



Sequence alignment guarantees

Theorem: Training set of size $\tilde{O}\left(\frac{\log(\text{seq. length})}{\epsilon^2}\right)$ implies WHP $\forall \boldsymbol{\rho}$,

$|$**avg** utility over training set – **exp** utility$| \leq \epsilon$



Outline

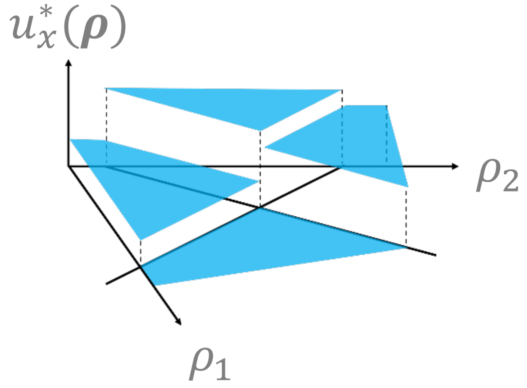
Conclusion

Guarantees for learning high-performing algorithm parameters

- Apply if performance is piecewise-structured function of parameters
- Proved by exploiting connections between primal and dual classes

Algorithm families from diverse domains exhibit this structure

- Clustering
- Economics (mechanism design)
- Integer programming
- Computational biology



Future research: Algorithms

This talk: Generalization guarantees

Apply to any configuration procedure (approximate, heuristic, optimal)

How to quickly find **provably** good parameters?

Growing body of work, but still many open questions

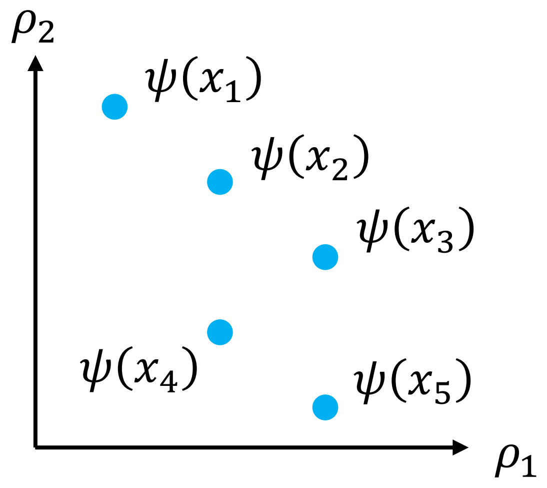
Kleinberg, Leyton-Brown, Lucier	IJCAI'17
Weisz, György, Szepesvári	ICML '18, '19
Kleinberg, Leyton-Brown, Lucier, Graham	NeurIPS'19
Weisz, György, Lin, Graham, Leyton-Brown, Szepesvári, Lucier	NeurIPS'20

Future research: Mapping to parameters

This talk: Learn **1** parameter setting that's good in expectation

Future research: More flexible approach

Learn **mapping** from inputs to parameter settings



Future research: Data-dependent bound

Strength of our results: **Input-distribution agnostic**

Apply to **any** distribution over instances

Can be loose for non-worst-case distributions

Guarantees that improve based on "niceness" of distribution?

- Covering-style analysis
- Rademacher complexity