Data Driven Algorithm Design

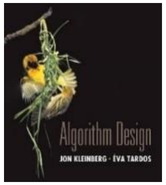
Maria-Florina (Nina) Balcan
Carnegie Mellon University

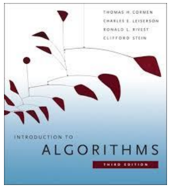
Analysis and Design of Algorithms

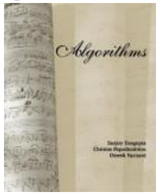
Classic algo design: solve a worst case instance.

- Easy domains, have optimal poly time algos.

  E.g., sorting, shortest paths







- Most domains are hard.

  E.g., clustering, partitioning, subset selection, auction design, …

Data driven algo design: use learning & data for algo design.

- Suited when repeatedly solve instances of the same algo problem.

Data Driven Algorithm Design

Data driven algo design: use learning & data for algo design.

- Different methods work better in different settings.
- Large family of methods – what's best in our application?

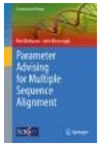
Prior work: largely empirical.

- Artificial Intelligence:

  [Horvitz-Ruan-Gomes-Kautz-Selman-Chickering, UAI 2001]

  [Xu-Hutter-Hoos-LeytonBrown, JAIR 2008]

- Computational Biology: E.g., [DeBlasio-Kececioglu, 2018]



- Game Theory: E.g., [Likhodedov and Sandholm, 2004]

Data Driven Algorithm Design

Data driven algo design: use learning & data for algo design.

- Different methods work better in different settings.
- Large family of methods – what's best in our application?

Prior work: largely empirical.

Our Work: Data driven algos with **formal guarantees.**

- Several cases studies of widely used algo families.
- General principles: push boundaries of algo design and ML.

Related to: Hyperparameter tuning, AutoML, MetaLearning.

Program Synthesis (Sumit Gulwani's talk on Mon).

Structure of the Talk

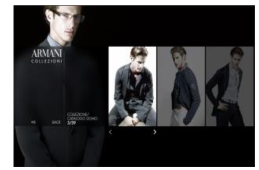
- Data driven algo design as batch learning.
    - A formal framework.
    - Case studies: clustering, partitioning pbs, auction pbs.
    - General sample complexity theorem.
- Data driven algo design as online learning.

Example: Clustering Problems

Clustering: Given a set objects organize then into natural groups.

- E.g., cluster news articles, or web pages, or search results by topic.





- Or, cluster customers according to purchase history.



- Or, cluster images by who is in them.

Often need do solve such problems repeatedly.

- E.g., clustering news articles (Google news).

Example: Clustering Problems

Clustering: Given a set objects organize then into natural groups.

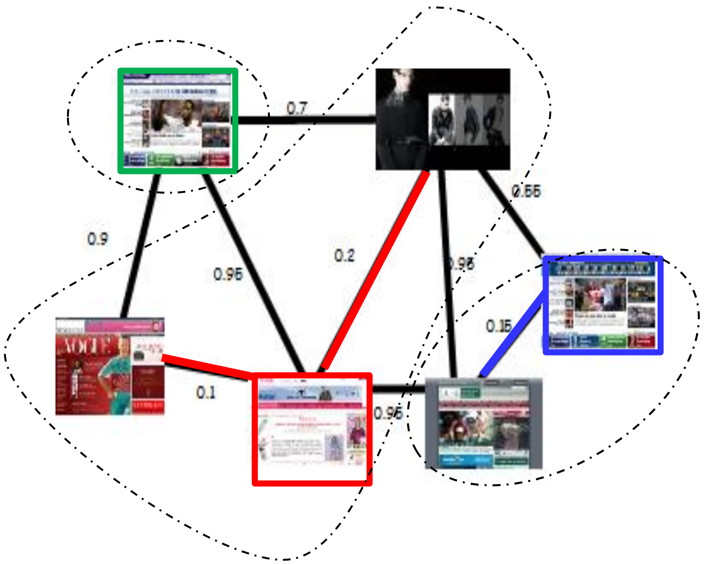
Objective based clustering

***k*-means**

Input: Set of objects S, d

Output: centers $\{c_1, c_2, \ldots, c_k\}$

To minimize $\sum_p \min_i d^2(p, c_i)$



k-median: min $\sum_p \min d(p, c_i)$.

k-center/facility location: minimize the maximum radius.

- Finding OPT is NP-hard, so no universal efficient algo that works on all domains.

Algorithm Design as Distributional Learning

Goal: given family of algos **F**, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Large family F of algorithms

MST + Dynamic Programming

Greedy + Farthest Location

...

Sample of typical inputs

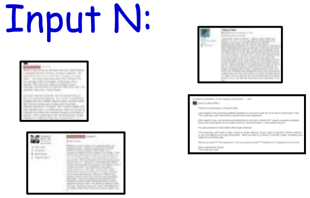
Clustering:

Input 1:

Input 2:

...

Input N:

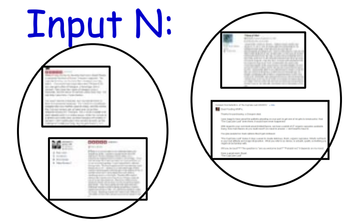


Input 1:

Input 2:

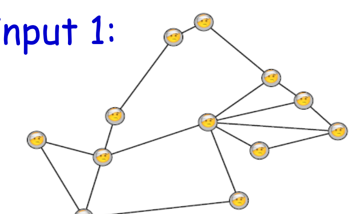
...

Input N:

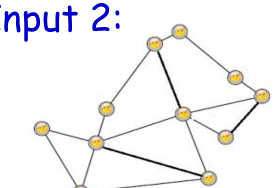


Facility location:

Input 1:

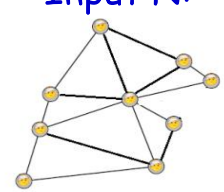


Input 2:



...

Input N:



Sample Complexity of Algorithm Selection

Goal: given family of algos **F**, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Approach: ERM, find $\hat{A}$ near optimal algorithm over the set of samples.

Key Question: Will $\hat{A}$ do well on future instances?

Sample Complexity: How large should our sample of typical instances be in order to guarantee good performance on new instances?

Sample Complexity of Algorithm Selection

Goal: given family of algos $\mathbf{F}$, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Approach: ERM, find $\hat{\mathbf{A}}$ near optimal algorithm over the set of samples.

Key tools from learning theory

- Uniform convergence: for any algo in F, average performance over samples "close" to its expected performance.

  - Imply that $\hat{\mathbf{A}}$ has high expected performance.

  - $N = O(\dim(\mathbf{F}) / \epsilon^2)$ instances suffice for ϵ-close.

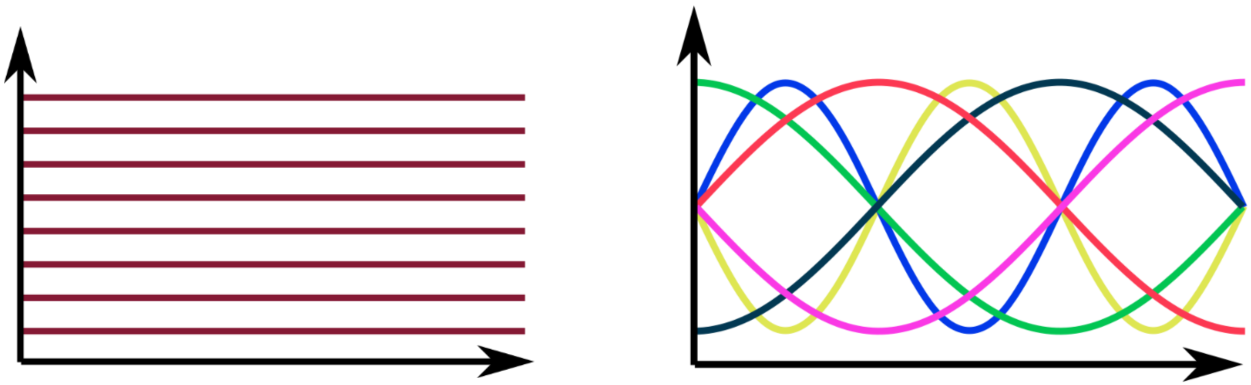
Sample Complexity of Algorithm Selection

Goal: given family of algos $\mathbf{F}$, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Key tools from learning theory

$N = O(\dim(\mathbf{F})/\epsilon^2)$ instances suffice for ϵ-close.

$\dim(\mathbf{F})$ (e.g. pseudo-dimension): ability of fns in $\mathbf{F}$ to fit complex patterns



More complex patterns can fit, more samples needed for UC and generalization

Sample Complexity of Algorithm Selection

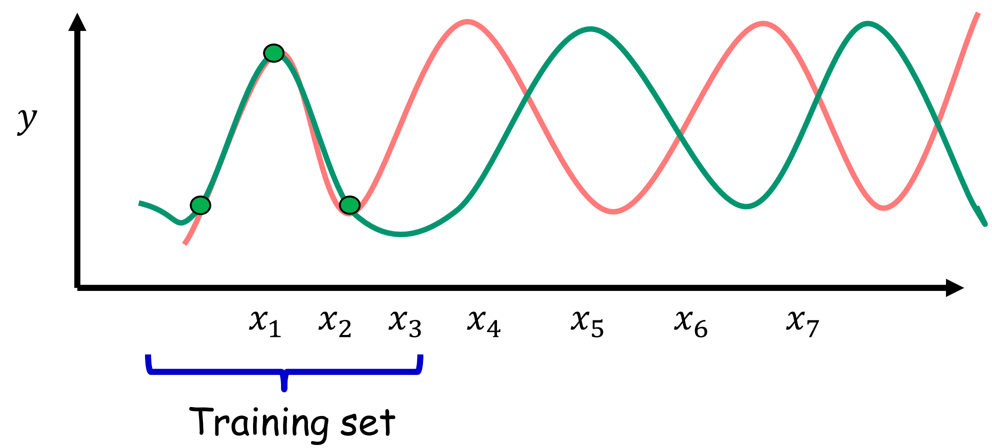
Goal: given family of algos $\mathbf{F}$, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Key tools from learning theory

$N = O(\dim(\mathbf{F})/\epsilon^2)$ instances suffice for ϵ-close.

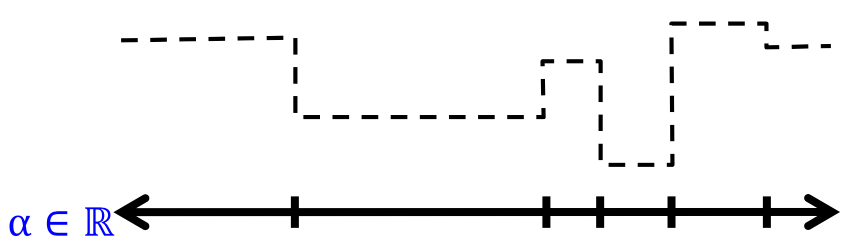
$\dim(\mathbf{F})$ (e.g. pseudo-dimension): ability of fns in $\mathbf{F}$ to fit complex patterns

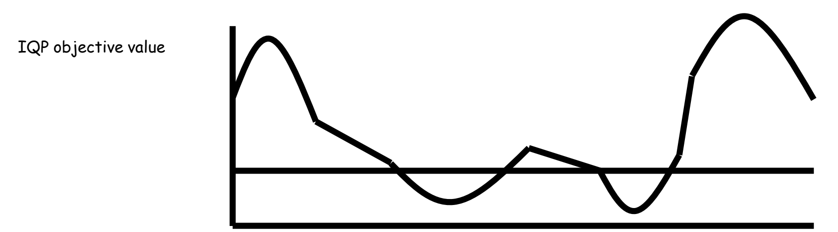
Overfitting

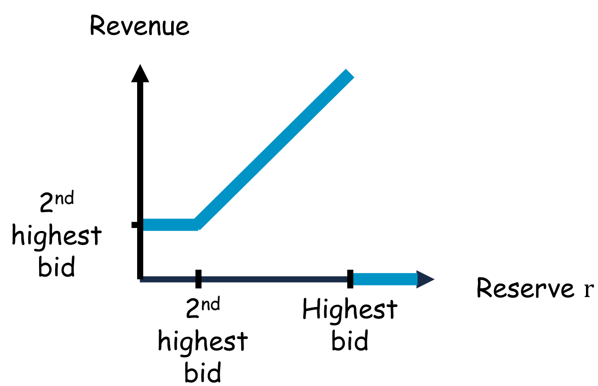


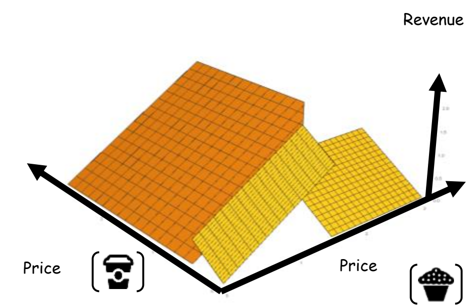
Statistical Learning Approach to AAD

Challenge: "nearby" algos can have drastically different behavior.









Challenge: design a computationally efficient meta-algorithm.

Algorithm Design as Distributional Learning

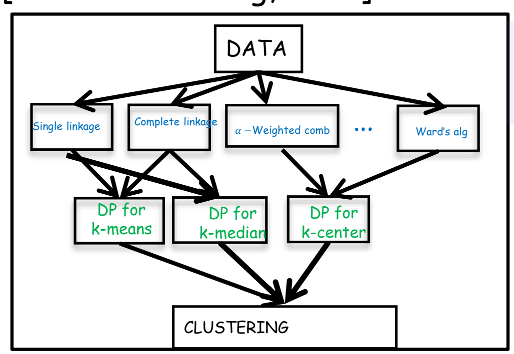
Prior Work: [Gupta-Roughgarden, ITCS'16 &SICOMP'17] proposed model; analyzed greedy algos for subset selection pbs (knapsack & independent set).

Our results: New algorithm classes for a wide range of problems.

Clustering: Parametrized Linkage

[Balcan-Nagarajan-Vitercik-White, COLT 2017]

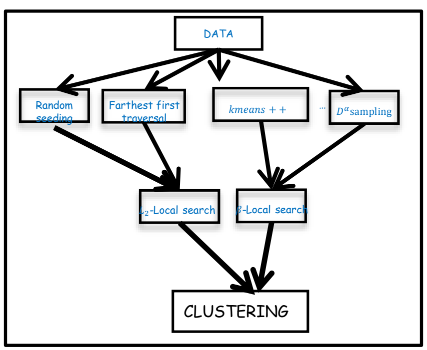
[Balcan-Dick-Lang, 2019]



$\dim(F) = O(\log n)$

Parametrized Lloyd's

[Balcan-Dick-White, NeurIPS 2018]



$\dim(F) = O(k \log n)$

Alignment pbs (e.g., string alignment): parametrized dynamic prog.

[Balcan-DeBlasio-Dick-Kingsford-Sandholm-Vitercik, 2019]

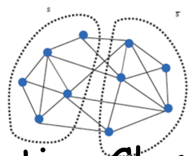
Algorithm Design as Distributional Learning

Our results: New algo classes applicable for a wide range of pbs.

- Partitioning pbs via IQPs: SDP + Rounding

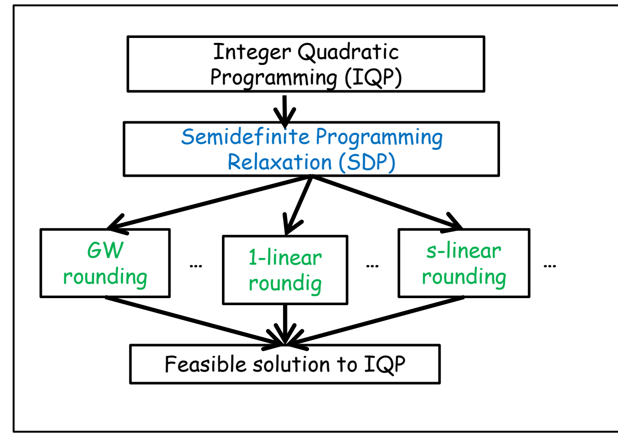
  [Balcan-Nagarajan-Vitercik-White, COLT 2017]

  E.g., Max-Cut,

  $\dim(F) = O(\log n)$

  Max-2SAT, Correlation Clustering



- Automated mechanism design

  [Balcan-Sandholm-Vitercik, EC 2018]

  Generalized parametrized VCG auctions, posted prices, lotteries.

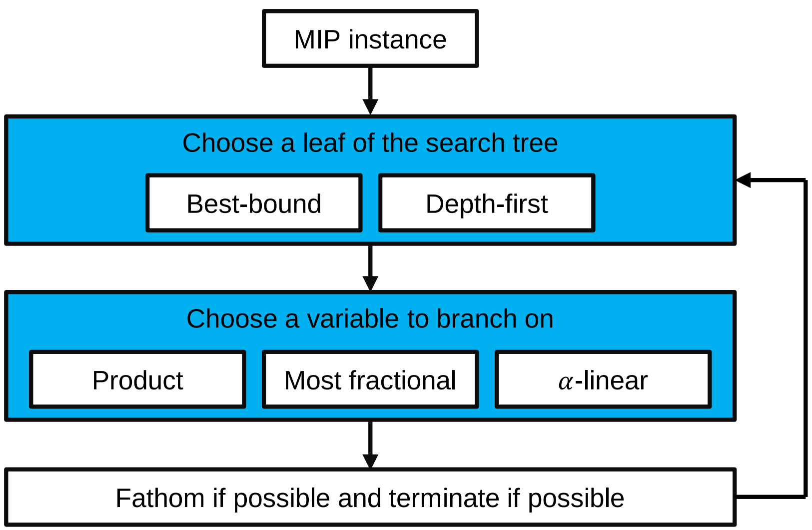
Algorithm Design as Distributional Learning

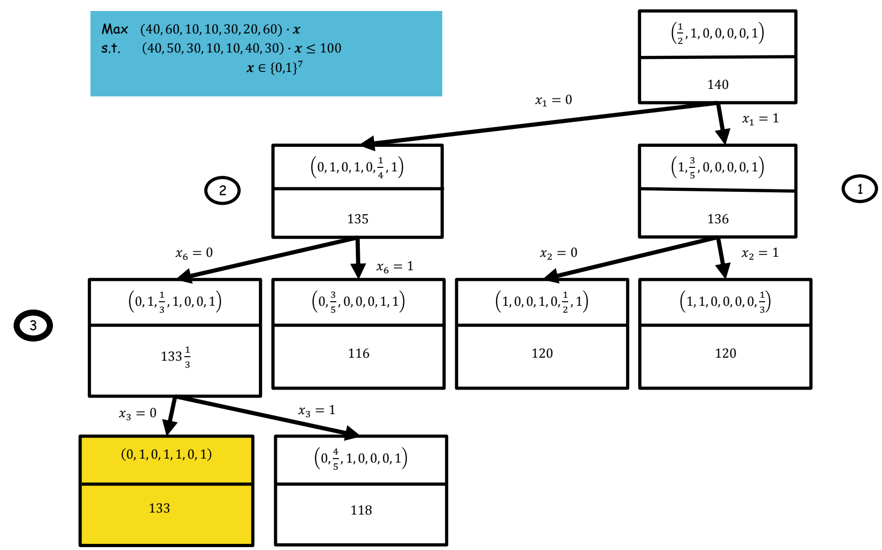
Our results: New algo classes applicable for a wide range of pbs.

- Branch and Bound Techniques for solving MIPs

  [Balcan-Dick-Sandholm-Vitercik, ICML'18]

Max $\boldsymbol{c} \cdot \boldsymbol{x}$
s.t. $\quad A\boldsymbol{x} = \boldsymbol{b}$
$\qquad x_i \in \{0,1\}, \forall i \in I$





Clustering Problems

Clustering: Given a set objects (news articles, customer surveys, web pages, …) organize then into natural groups.

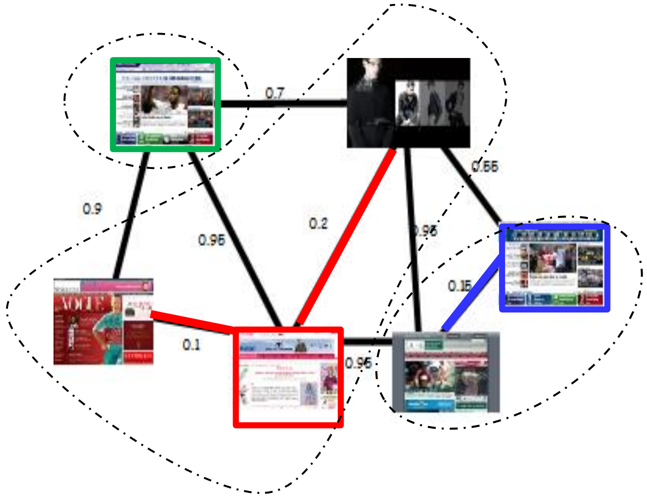
Objective based clustering

***k*-means**

Input: Set of objects S, d

Output: centers $\{c_1, c_2, \ldots, c_k\}$

To minimize $\sum_p \min_i d^2(p, c_i)$



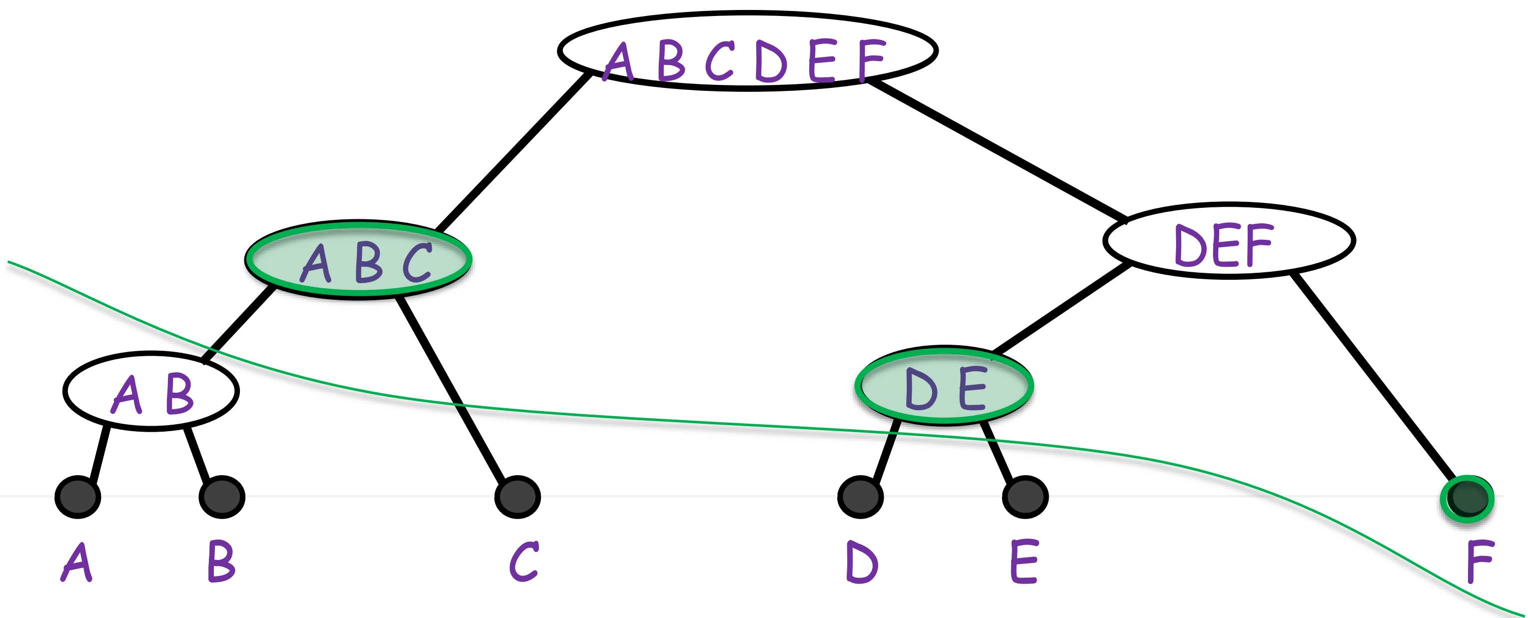
Or minimize distance to ground-truth

Clustering: Linkage + Post-processing

Family of poly time 2-stage algorithms:

[Balcan-Nagarajan-Vitercik-White, COLT 2017]

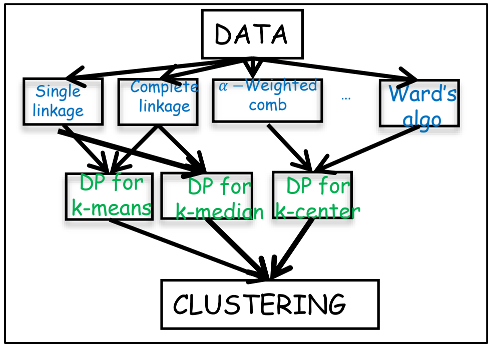
1. Greedy linkage-based algo to get hierarchy (tree) of clusters.
2. Fixed algo (e.g., DP or last k-merges) to select a good pruning.

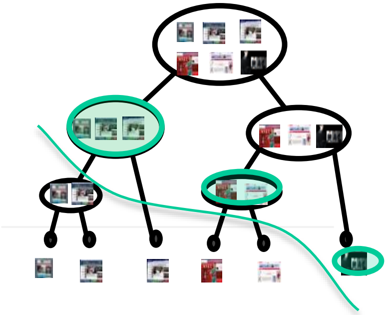


Clustering: Linkage + Post-processing

1. Linkage-based algo to get a hierarchy.
2. Post-processing to identify a good pruning.

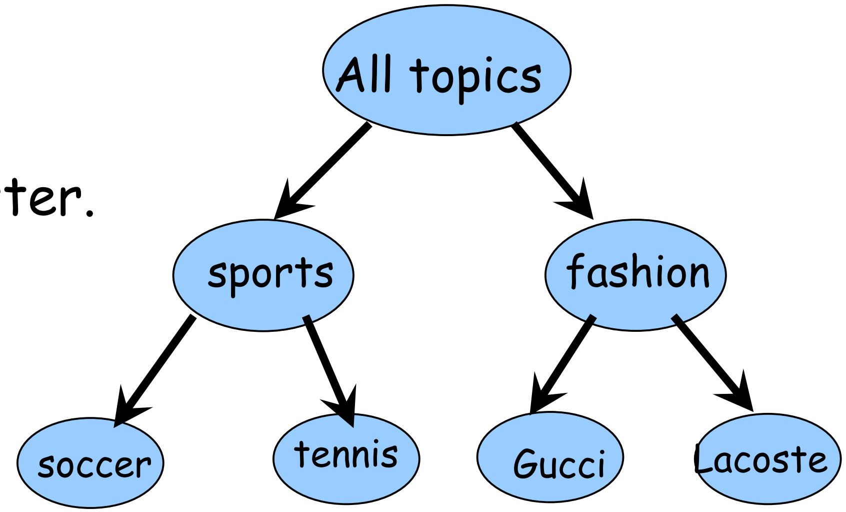
Both steps can be done efficiently.





Linkage Procedures for Hierarchical Clustering

Bottom-Up (agglomerative)



- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

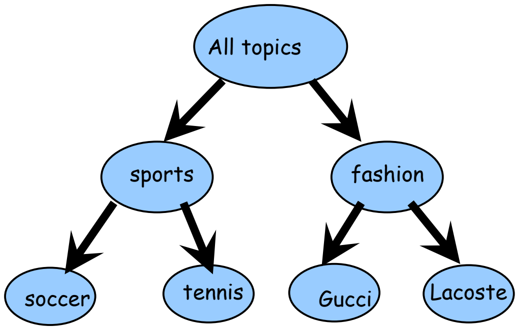
Different defs of "closest" give different algorithms.

Linkage Procedures for Hierarchical Clustering

Have a distance measure on pairs of objects.

$d(x,y)$ – distance between x and y

E.g., # keywords in common, edit distance, etc



- Single linkage: $\mathrm{dist}(A,B) = \min_{x \in A, x' \in B} \mathrm{dist}(x,x')$

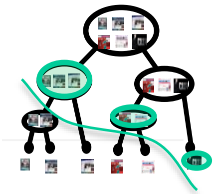
- Complete linkage: $\mathrm{dist}(A,B) = \max_{x \in A, x' \in B} \mathrm{dist}(x,x')$

- Parametrized family, α-weighted linkage:

$$\mathrm{dist}_\alpha(A,B) = (1-\alpha) \min_{x \in A, x' \in B} d(x,x') + \alpha \max_{x \in A, x' \in B} d(x,x')$$

Clustering: Linkage + Post Processing

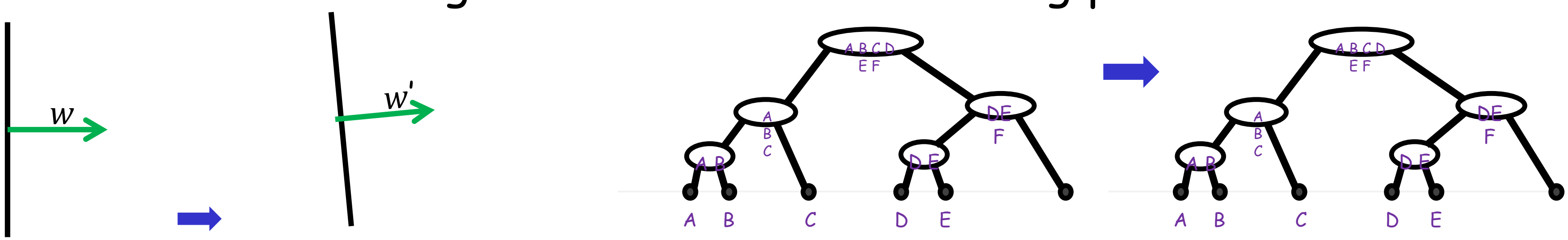
Our Results: **α-weighted linkage** **+ Post-processing**



- Pseudo-dimension is O(log n), so small sample complexity.

- Given sample S, find best algo from this family in poly time.

Key Technical Challenge: small changes to the parameters of the algo can lead to radical changes in the tree or clustering produced.

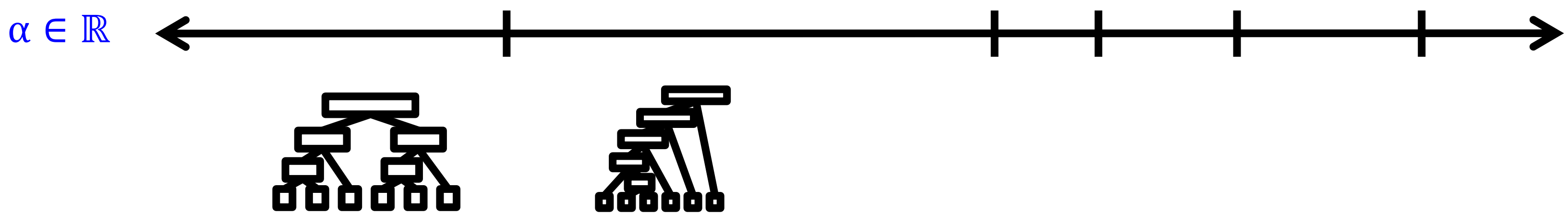


Problem: a single change to an early decision by the linkage algo, can snowball and produce large changes later on.

Clustering: Linkage + Post Processing

Claim: Pseudo-dim of α-weighted linkage + Post-process is O(log n).

Key fact: If we fix a clustering instance of n pts and vary α, at most $O(n^8)$ switching points where behavior on that instance changes.

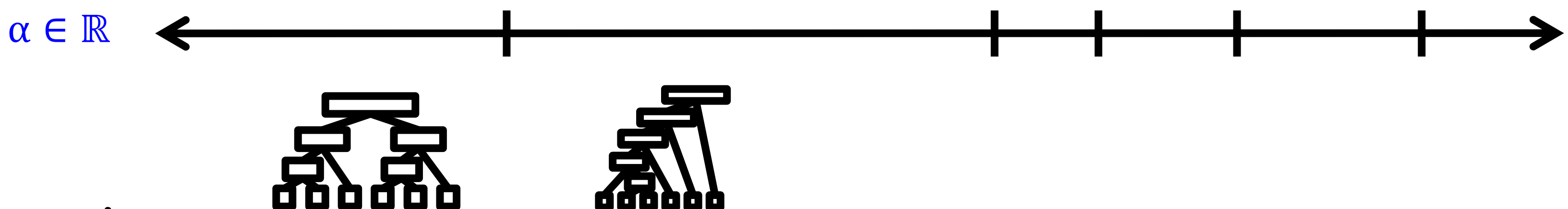


So, the cost function is piecewise-constant with at most $O(n^8)$ pieces.

Clustering: Linkage + Post Processing

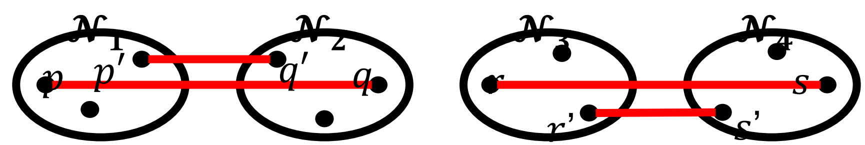
Claim: Pseudo-dim of α-weighted linkage + Post-process is $O(\log n)$.

Key fact: If we fix a clustering instance of n pts and vary α, at most $O(n^8)$ switching points where behavior on that instance changes.



Key idea:

- For a given α, which will merge first, $\mathcal{N}_1$ and $\mathcal{N}_2$, or $\mathcal{N}_3$ and $\mathcal{N}_4$?



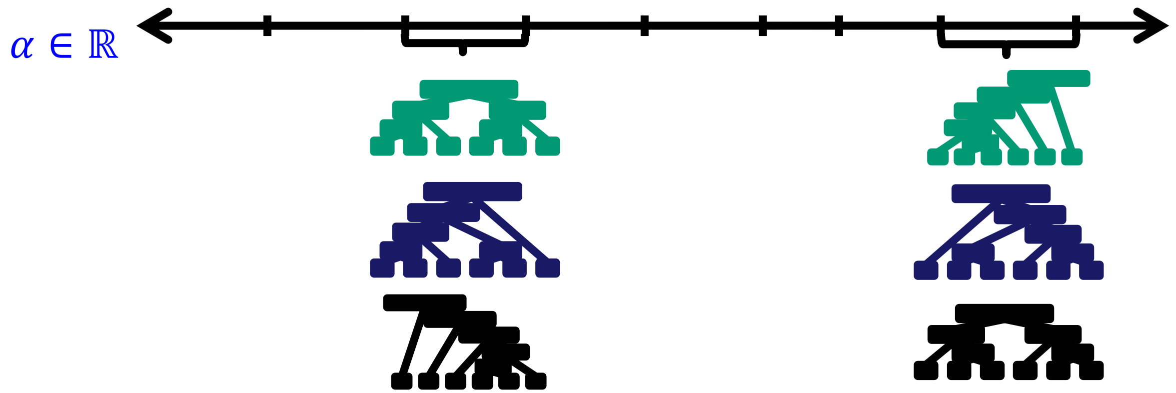
- Depends on which of $\alpha d(p,q) + (1-\alpha) d(p',q')$ or $\alpha d(r,s) + (1-\alpha) d(r',s')$ is smaller.

- An interval boundary an equality for 8 points, so $O(n^8)$ interval boundaries.

Clustering: Linkage + Post Processing

Claim: Pseudo-dim of α-weighted linkage + Post-process is O(log n).

Key idea: For m clustering instances of n points, $O(mn^8)$ patterns.



- Pseudo-dim largest m for which 2^m patterns achievable.
- So, solve for $2^m \leq m\, n^8$. Pseudo-dimension is O(log n).

Clustering: Linkage + Post Processing

Claim: Pseudo-dim of α-weighted linkage + Post-process is $O(\log n)$.

For $N = O(\log n / \epsilon^2)$, w.h.p. expected performance cost of best α over the sample is ϵ-close to optimal over the distribution

Claim: Given sample S, can find best algo from this family in poly time.

- Solve for all α intervals over the sample.

$\alpha \in [0,1]$

- Find α interval with smallest empirical cost.

Learning Both Distance and Linkage Criteria

[Balcan-Dick-Lang, 2019]

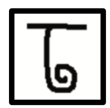
- Often different types of distance metrics.
  - Captioned images, d_0 image info, d_1 caption info.

"Black Cat"

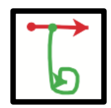
"Bobcat"

  - Handwritten images: d_0 pixel info (CNN embeddings), d_1 stroke info.

Character Image



Stroke Data



Family of Metrics: Given d_0 and d_1, define

$$d_\beta(x, x') = (1 - \beta) \cdot d_0(x, x') + \beta \cdot d_1(x, x')$$

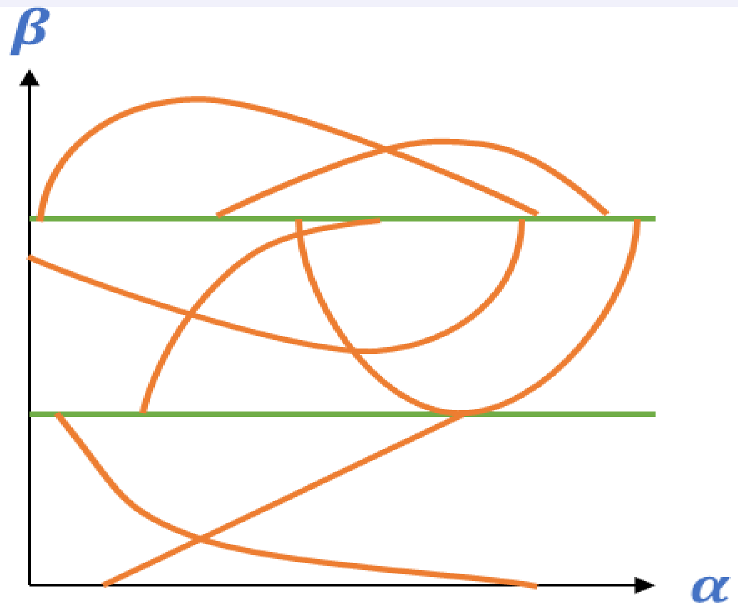
Parametrized **(α, β)-weighted linkage** (α interpolation between single and complete linkage and β interpolation between two metrics):

$$\mathrm{dist}_\alpha(A, B; d_\beta) = (1 - \alpha) \min_{x \in A, x' \in B} d_\beta(x, x') + \alpha \max_{x \in A, x' \in B} d_\beta(x, x')$$

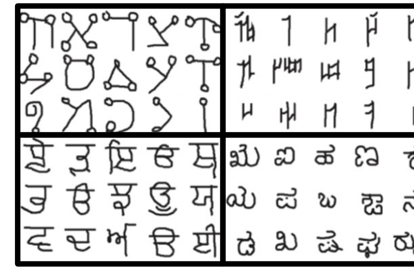
Learning Both Distance and Linkage Criteria

Claim: Pseudo-dim. of (α, β) -weighted linkage is O(log n).

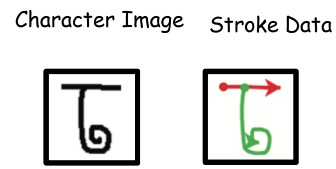
Key fact: Fix instance of n pts; vary α, β, partition space with $O(n^8)$ linear, quadratic equations s.t. within each region, same cluster tree.



Learning Distance for Clustering Subsets of Omniglot



- Written characters from 50 alphabets, each character 20 examples. [Lake, Salakhutdinov, Tenenbaum '15]
- Image & stroke (trajectory of pen)



<u>Instance Distribution</u>

- Pick random alphabet. Pick 5 to 10 characters.
- Use all 20 examples of chosen characters (100 - 200 points)
- Target clusters are characters.

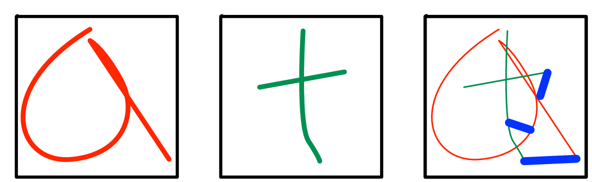
- d_0 uses character images.

  Cosine distance between CNN feature embeddings

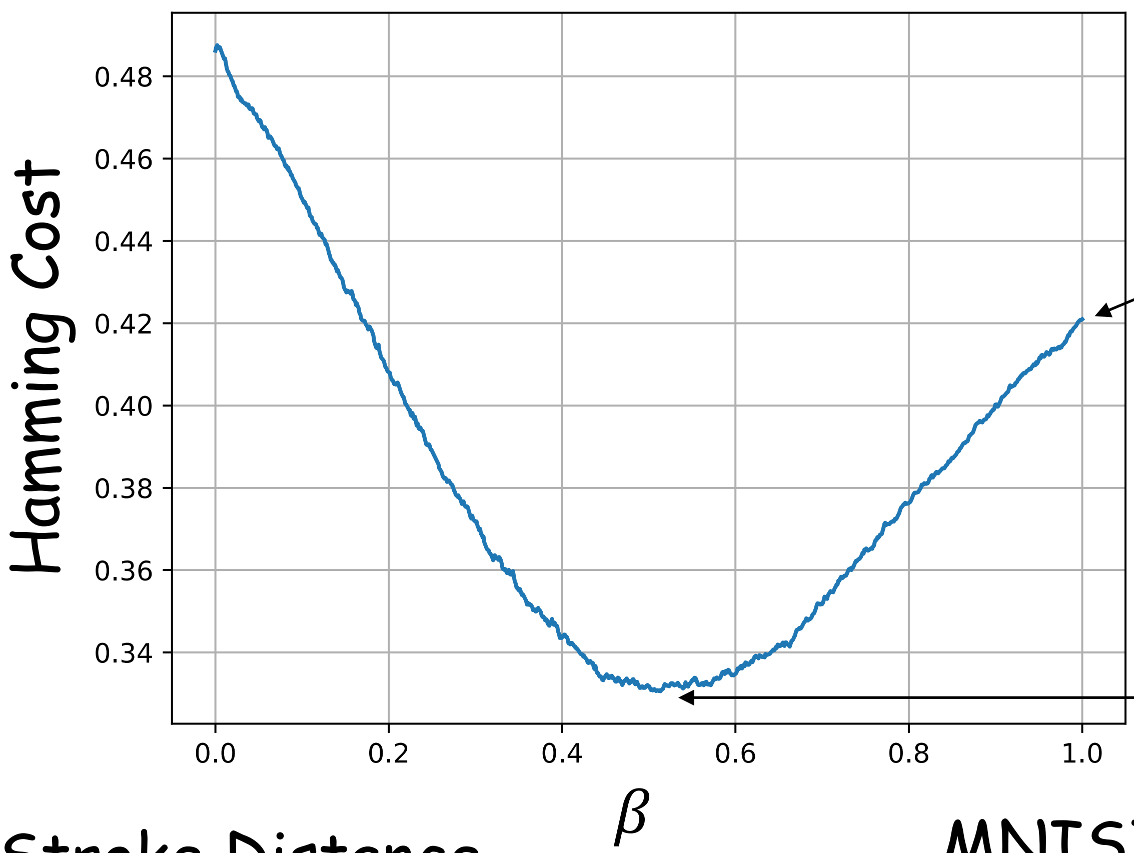
  CNN trained on MNIST.

- d_1 Hand-designed Stroke.

  Average distance from points on each stroke to nearest point on other stroke.



Clustering Subsets of Omniglot



$\beta = 1$
Error = 42.1%

$\beta^* = 0.514$
Error = 33.0%
Improvement of 9.1%

Stroke Distance

MNIST Features

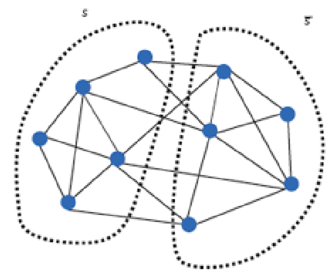
Partitioning Problems via IQPs

IQP formulation

$$\text{Max } \mathbf{x}^T A \mathbf{x} = \sum_{i,j} a_{i,j} x_i x_j$$
$$\text{s.t. } \mathbf{x} \in \{-1,1\}^n$$

Many of these pbs are NP-hard.

E.g., **Max cut**: partition a graph into two pieces to maximize weight of edges crossing the partition.



Input: Weighted graph G, w

Output: $$\text{Max } \sum_{(i,j)\in E} w_{ij} \boxed{\left(\frac{1-v_i v_j}{2}\right)}$$
$$\text{s.t. } v_i \in \{-1,1\}$$

1 if v_i, v_j opposite sign,
0 if same sign

var v_i for node i, either +1 or -1

Parametrized family of rounding procedures

IQP formulation

$$\text{Max } \mathbf{x}^T A \mathbf{x} = \sum_{i,j} a_{i,j} x_i x_j$$
$$\text{s.t. } \mathbf{x} \in \{-1,1\}^n$$

Algorithmic Approach: SDP + Rounding

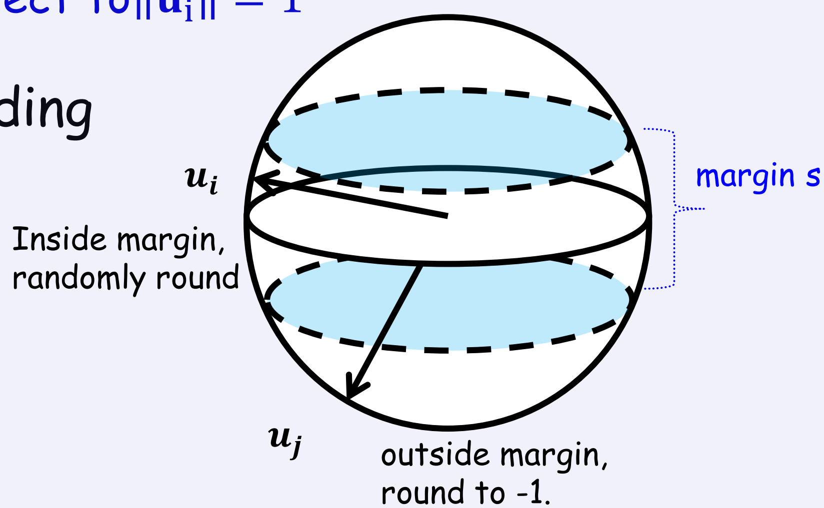
1. SDP relaxation:

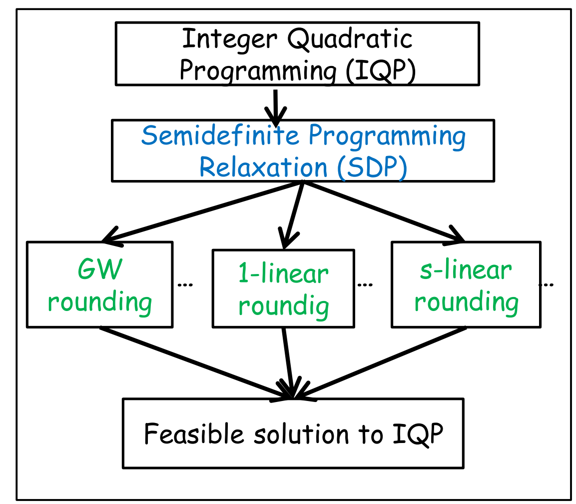
Associate each binary variable x_i with a vector $\mathbf{u}_i$.

$$\text{Max } \sum_{i,j} a_{i,j} \langle \mathbf{u}_i, \mathbf{u}_j \rangle$$
$$\text{subject to } \|\mathbf{u}_i\| = 1$$

2. s-Linear Rounding

[Feige&Landberg'06]



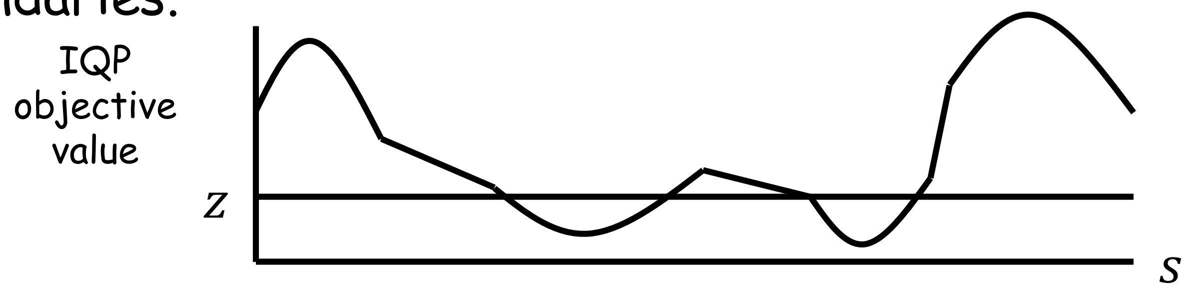


Partitioning Problems via IQPs

Our Results: SDP + s-linear rounding

Pseudo-dimension is O(log n), so small sample complexity.

Key idea: expected IQP objective value is piecewise quadratic in $\frac{1}{s}$ with n boundaries.



Given sample S, can find best algo from this family in poly time.

Data-driven Mechanism Design

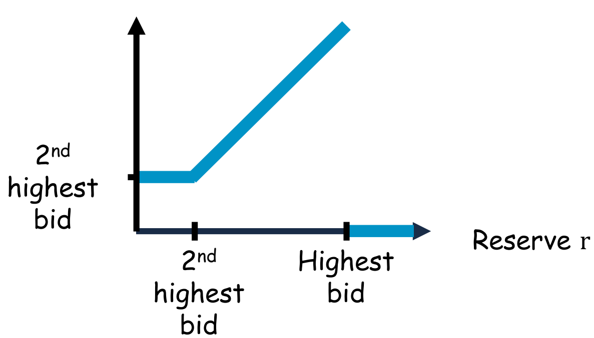
- **Mechanism design** for revenue maximization.

[Balcan-Sandholm-Vitercik, EC'18]

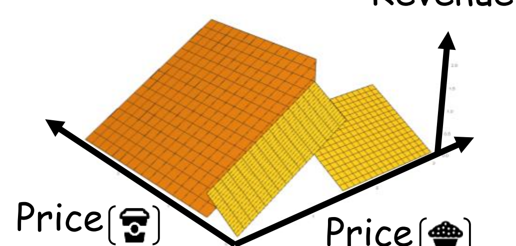
- Pseudo-dim of $\{\text{revenue}_M : M \in \mathcal{M}\}$ for multi-item multi-buyer settings.
  - Many families: second-price auctions with reserves, posted pricing, two-part tariffs, parametrized VCG auctions, etc.

- Key insight: dual function sufficiently structured.
  - For a fixed set of bids, revenue is piecewise linear fnc of parameters.

2nd-price auction with reserve



Posted price mechanisms



General Sample Complexity via Dual Classes

[Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, 2019]

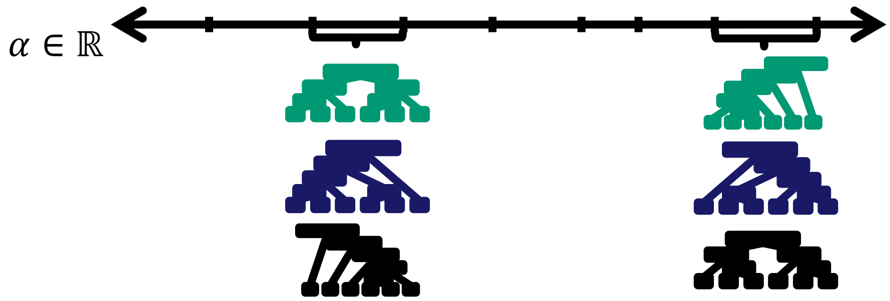
High level learning theory bit

- Want to prove that for all algorithm parameters $\boldsymbol{\alpha}$:

  $\frac{1}{|\mathcal{S}|}\sum_{\mathrm{I}\in\mathcal{S}} \mathrm{cost}_{\boldsymbol{\alpha}}(\mathrm{I})$ close to $\mathbb{E}[\mathrm{cost}_{\boldsymbol{\alpha}}(\mathbf{I})]$.

- Function class whose complexity want to control: $\{\mathrm{cost}_{\boldsymbol{\alpha}}: \text{parameter } \boldsymbol{\alpha}\}$.
- Proof takes advantage of structure of **dual class** $\{\mathrm{cost}_{\mathrm{I}}: \text{instances } \mathbf{I}\}$.

$\mathrm{cost}_{\mathbf{I}}(\boldsymbol{\alpha}) = \mathrm{cost}_{\boldsymbol{\alpha}}(\mathbf{I})$



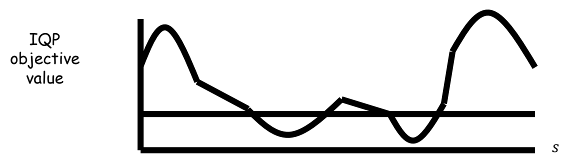
Structure of the Talk

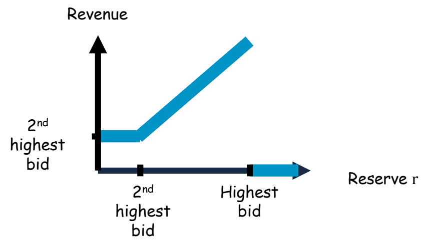
- Data driven algo design as batch learning.
  - A formal framework.
  - Case studies: clustering, partitioning pbs, auction problems.
- Data driven algo design via online learning.

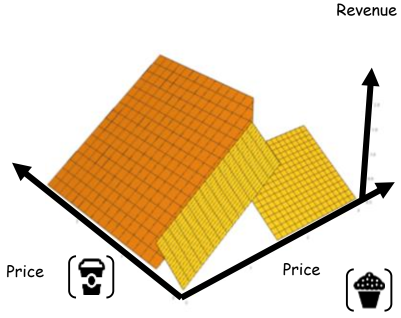
Online Algorithm Selection

- So far, batch setting: collection of typical instances given upfront.
- [Balcan-Dick-Vitercik, FOCS 2018], [Balcan-Dick-Pedgen, 2019] online alg. selection.

- **Challenge**: scoring fns non-convex, with lots of discontinuities.







Cannot use known techniques.

- Identify general properties (piecewise Lipschitz fns with dispersed discontinuities) sufficient for strong bounds.
    - Show these properties hold for many alg. selection pbs.

Online Algorithm Selection via Online Optimization

Online optimization of general piecewise Lipschitz functions

On each round $t \in \{1, \dots, T\}$:

1. Online learning algo chooses a parameter $\boldsymbol{\rho}_t$
2. Adversary selects a piecewise Lipschitz function $u_t: \mathcal{C} \to [0, H]$
    - corresponds to some pb instance and its induced scoring fnc

   Payoff: score of the parameter we selected $u_t(\rho_t)$.
3. Get **feedback:** Full information: observe the function $u_t(\cdot)$

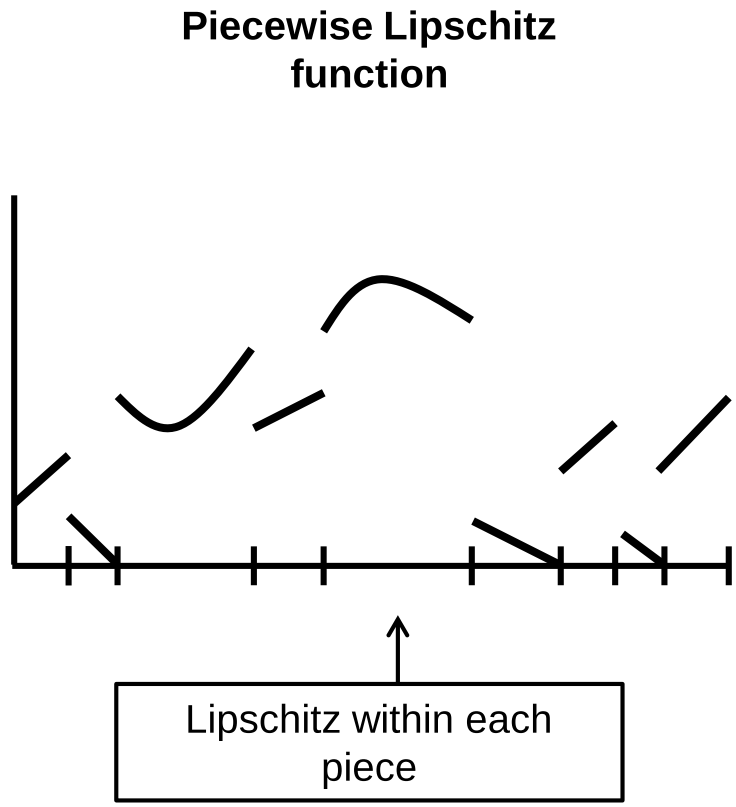
   Bandit feedback: observe only payoff $u_t(\boldsymbol{\rho}_t)$.

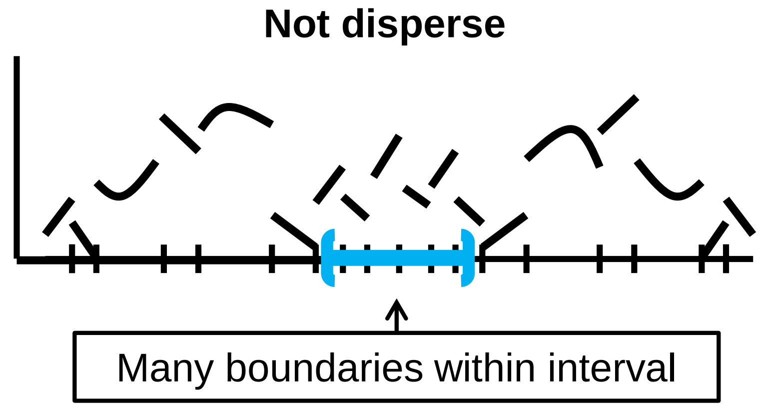
Goal: **minimize regret**: $\max_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right]$

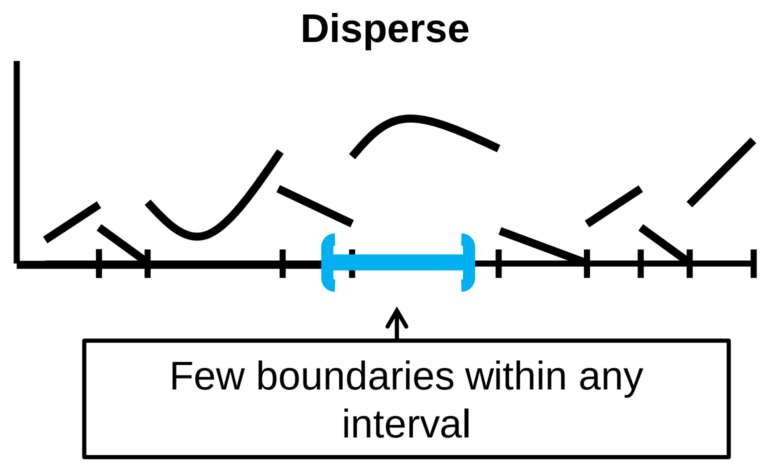
Performance of best parameter in hindsight

Our cumulative performance

Dispersion, Sufficient Condition for No-Regret







$\{u_1(\cdot), \dots, u_T(\cdot)\}$ is $(\mathbf{w}, \mathbf{k})$-dispersed if any ball of radius $\mathbf{w}$ contains boundaries for at most $\mathbf{k}$ of the u_i.

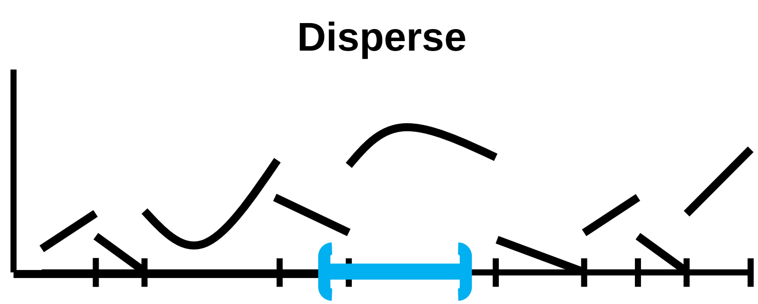
Dispersion, Sufficient Condition for No-Regret

Full info: exponentially weighted forecaster [Cesa-Bianchi-Lugosi 2006]

On each round $t \in \{1, \ldots, T\}$:

- Sample a vector $\boldsymbol{\rho}_t$ from distr. p_t: $$p_t(\boldsymbol{\rho}) \propto \exp\left(\lambda \sum_{s=1}^{t-1} u_s(\boldsymbol{\rho})\right)$$

Our Results:

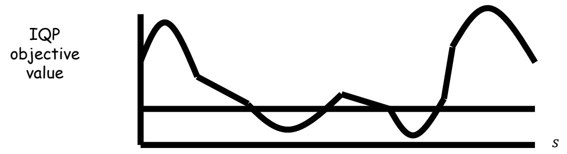


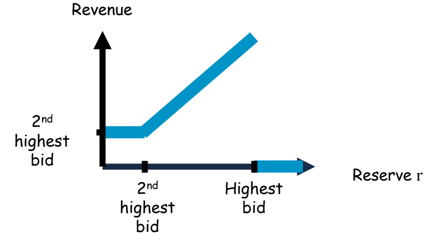
Disperse fns, regret $\tilde{O}\left(\sqrt{Td}\ \text{fnc of problem}\right)$.

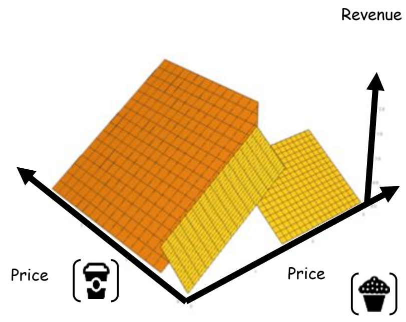
Summary and Discussion

- Strong performance guarantees for data driven algorithm selection for combinatorial problems.

- Provide and exploit structural properties of dual class for good sample complexity and regret bounds.







- Machine learning: techniques of independent interest beyond algorithm selection.

Many Exciting Open Directions

- Analyze other widely used classes of algorithmic paradigms.
  - Branch and Bound Techniques for MIPs [Balcan-Dick-Sandholm-Vitercik, ICML'18]
  - Parametrized Lloyd's methods [Balcan-Dick-White, NeurIPS'18]
  - Other algorithmic paradigms relevant to data-mining pbs.
- Other learning models (e.g., one shot, domain adaptation, reinforcement learning).
- Explore connections to program synthesis; automated algo design.
- Connections to Hyperparameter tuning, AutoML, Meta-learning.

  Use our insights for pbs studied in these settings (e.g., tuning hyper-parameters in deep nets)