Data Driven Algorithm Design

Maria-Florina (Nina) Balcan

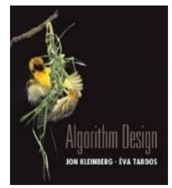
Carnegie Mellon University

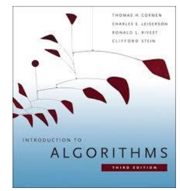
Analysis and Design of Algorithms

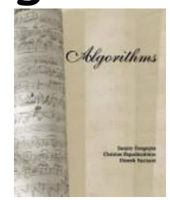
Classic algo design: solve a worst case instance.

- 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.

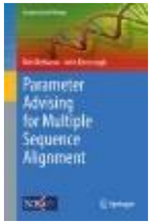
Data Driven Algorithm Design

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: E.g., [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.

This talk: Data driven algos with **formal guarantees.**

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

Related in spirit to Hyperparameter tuning, AutoML, MetaLearning.

Structure of the Talk

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

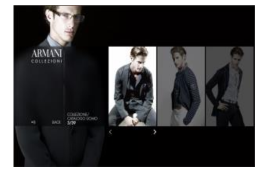
- Data driven algo design via online learning.
  - Online learning of non-convex (piecewise Lipschitz) fns.

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.

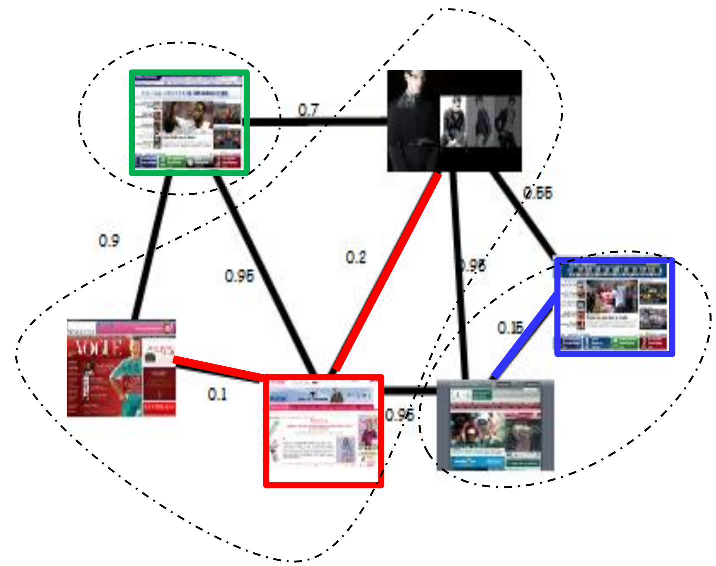
Objective based clustering

***k*-means**

Input: Set of objects S, d

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

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 Selection as a Learning Problem

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

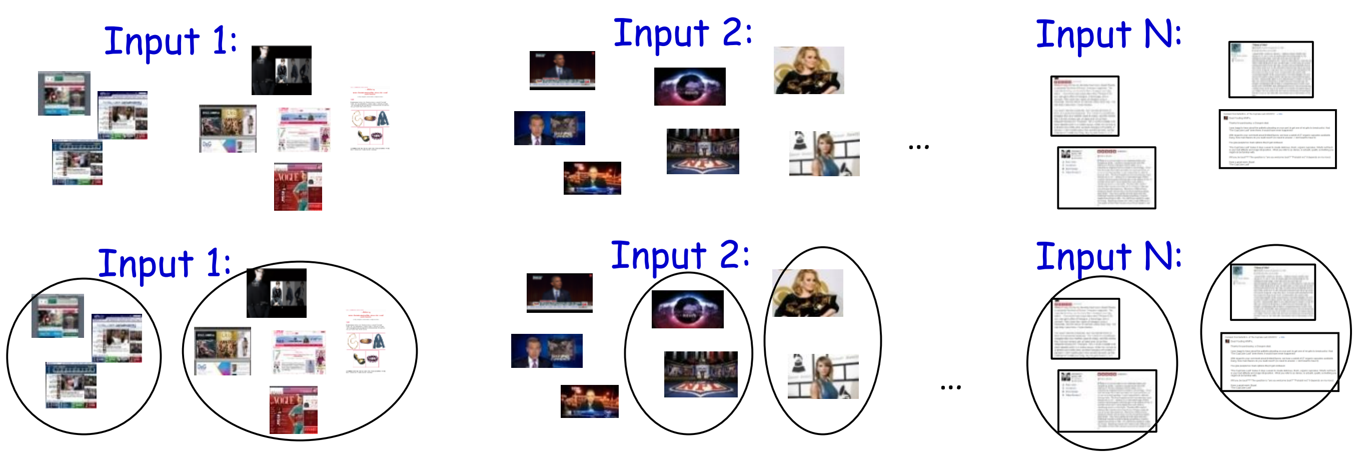
MST + Dynamic Programming

Greedy + Farthest Location

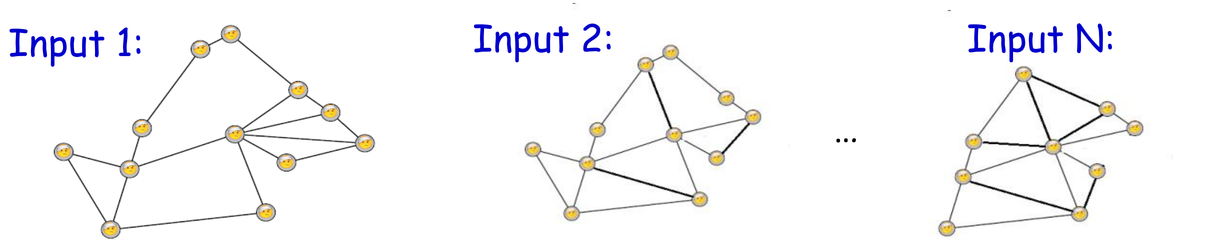
...

Sample of typical inputs

Clustering:



Facility location:



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: 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: 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.

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

Sample Complexity of Algorithm Selection

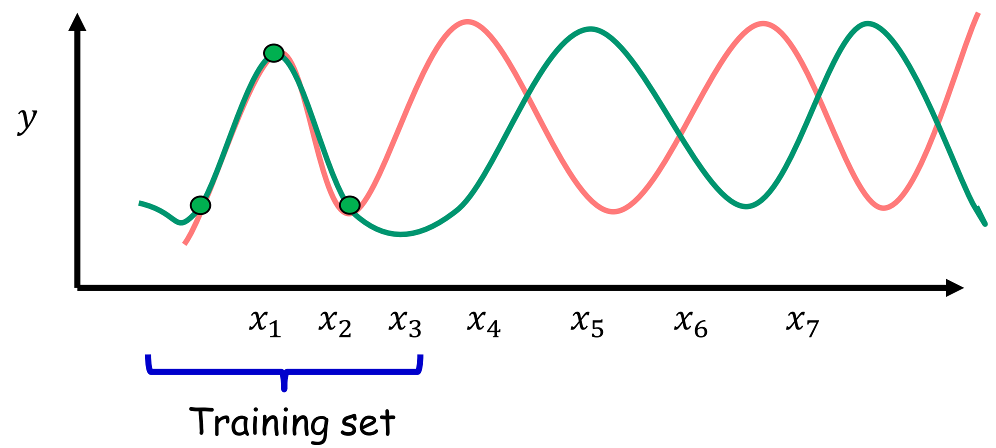
Goal: given family of algos $\mathbf{F}$, sample of typical instances from domain (unknown distr. D), find algo that performs well on new instances from D.

Key tools from learning theory

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

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

Overfitting



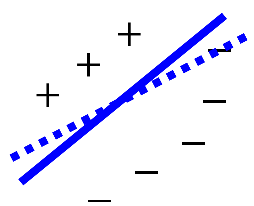
Sample Complexity of Algorithm Selection

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.

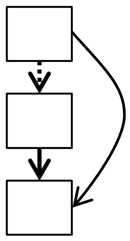
Key tools from learning theory

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

Challenge: analyze dim(F), due to combinatorial & modular nature, "nearby" programs/algos can have drastically different behavior.



Classic machine learning



Our work

Challenge: design a computationally efficient meta-algorithm.

Formal Guarantees for Algorithm Selection

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

Our results:

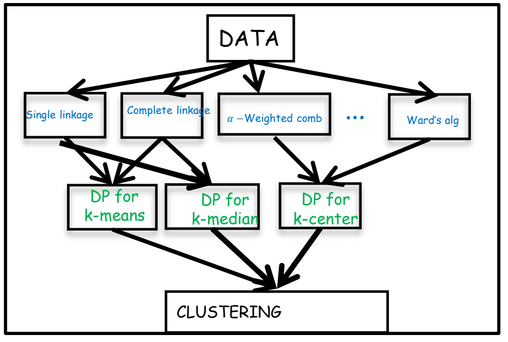
- New algorithm classes applicable for a wide range of problems (e.g., clustering, partitioning, auctions).
- General techniques for sample complexity based on properties of the dual class of fns.

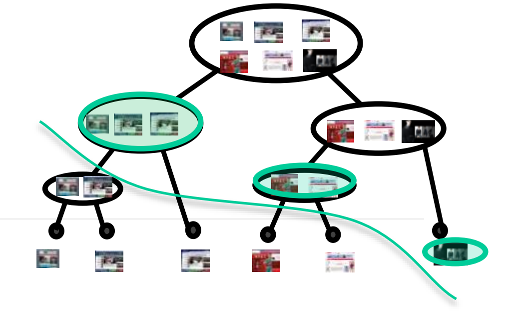
Formal Guarantees for Algorithm Selection

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

- Clustering: Linkage + Dynamic Programming

  [Balcan-Nagarajan-Vitercik-White, COLT'17]

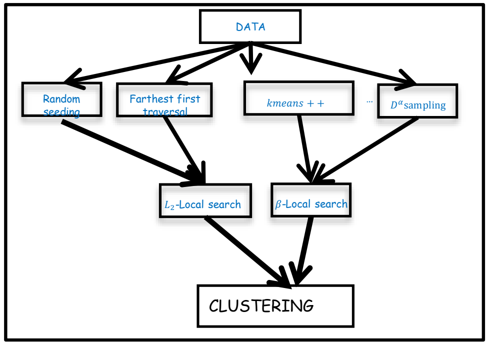




- Clustering: Greedy Seeding + Local Search

  [Balcan-Dick-White, NeruIPS'18]

  Parametrized Lloyds methods



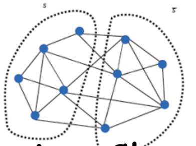
Formal Guarantees for Algorithm Selection

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

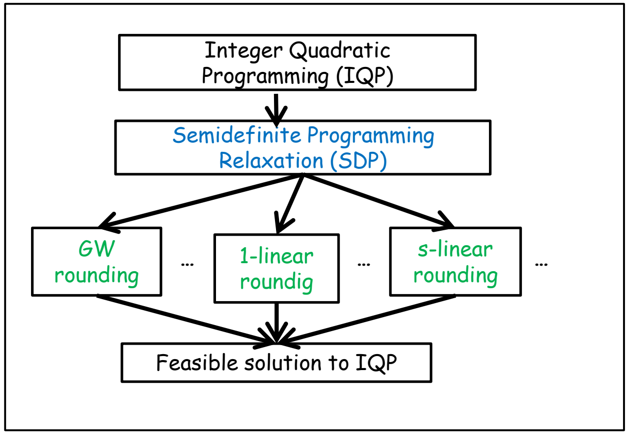
- Partitioning pbs via IQPs: SDP + Rounding

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

  E.g., Max-Cut,

  Max-2SAT, Correlation Clustering



- Automated mechanism design

  [Balcan-Sandholm-Vitercik, EC 2018]

  Generalized parametrized VCG auctions, posted prices, lotteries.

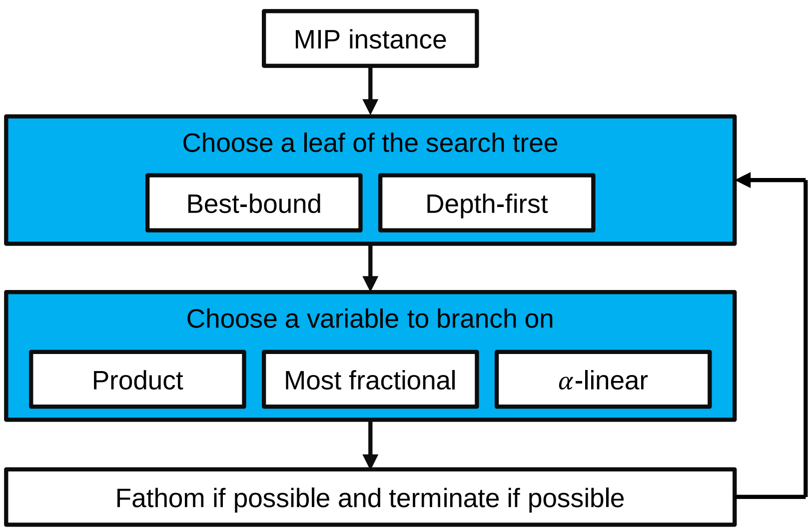
Formal Guarantees for Algorithm Selection

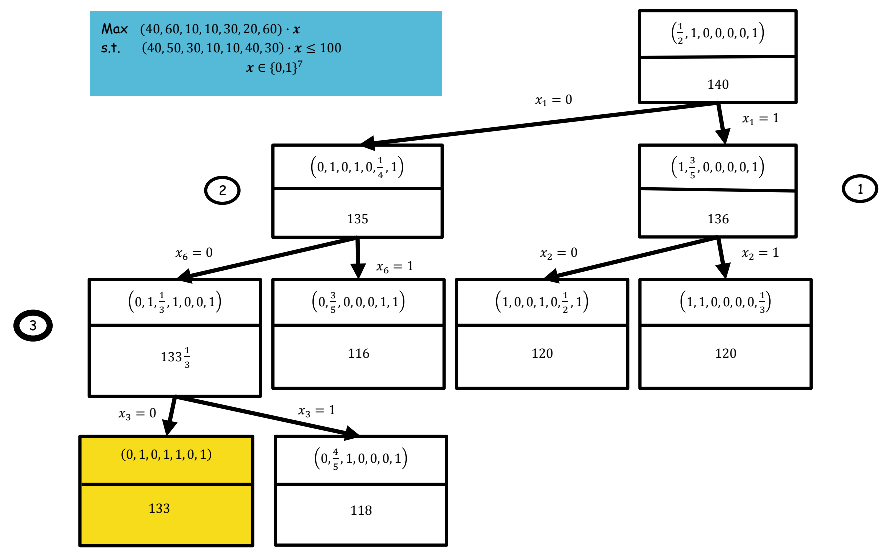
Our results: New algo classes applicable for a wide range of pbs.

- Branch and Bound Techniques for solving MIPs

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

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





Clustering Problems

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

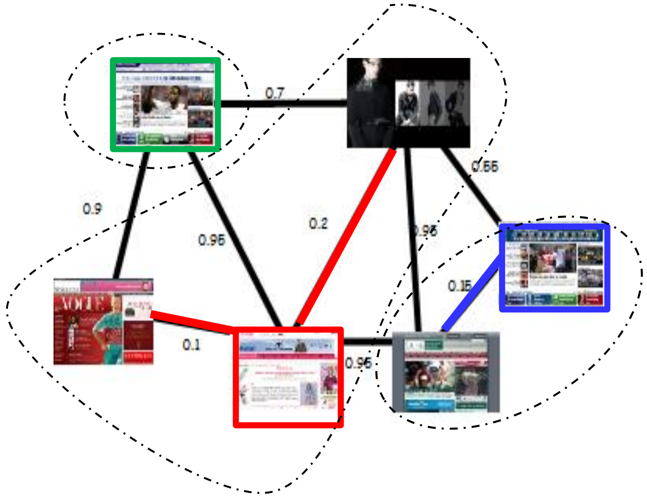
Objective based clustering

***k*-means**

Input: Set of objects S, d

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

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



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

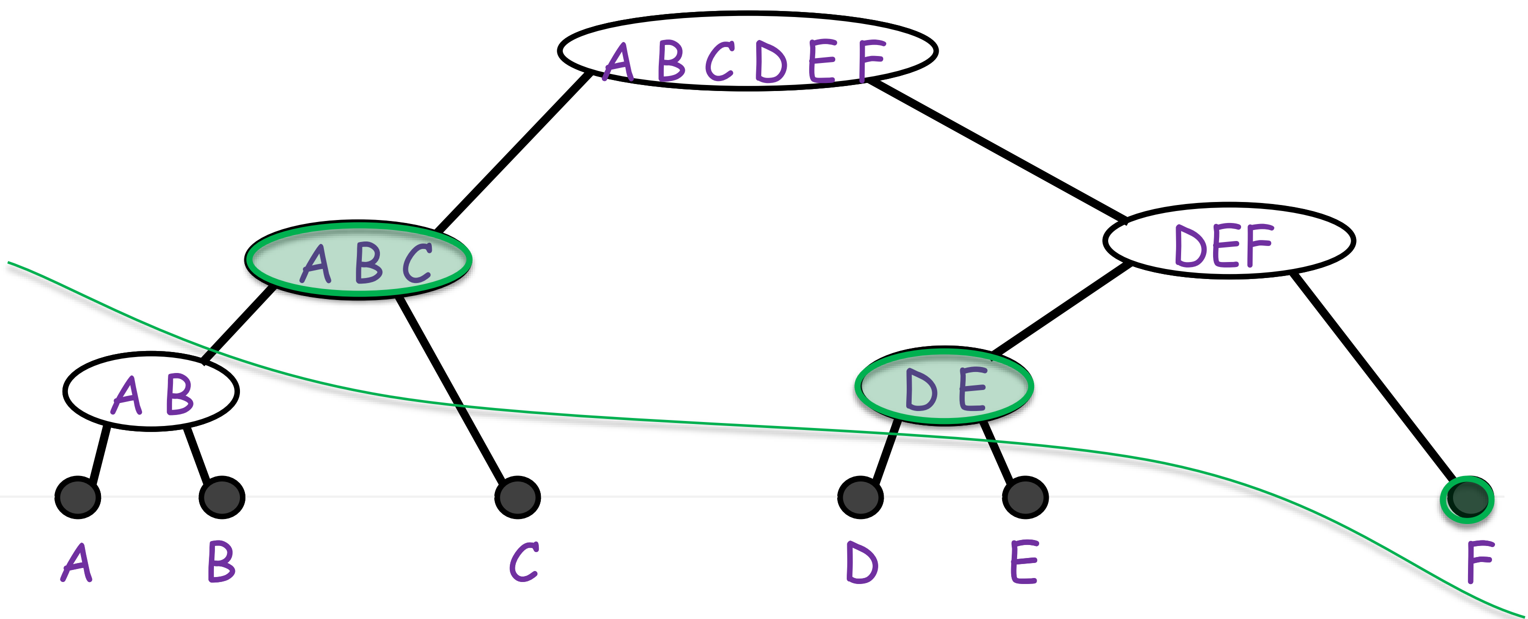
k-center: minimize the maximum radius.

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

Clustering: Linkage + Dynamic Programming

Family of poly time 2-stage algorithms:

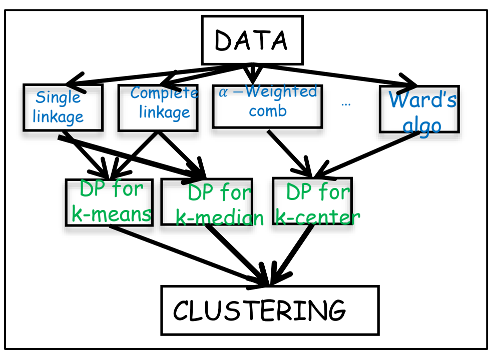
1. Use a greedy linkage-based algorithm to organize data into a hierarchy (tree) of clusters.
2. Dynamic programming over this tree to identify pruning of tree corresponding to the best clustering.

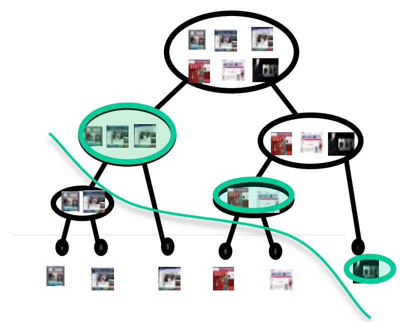


Clustering: Linkage + Dynamic Programming

1. Use a linkage-based algorithm to get a hierarchy.
2. Dynamic programming to the best pruning.

Both steps can be done efficiently.

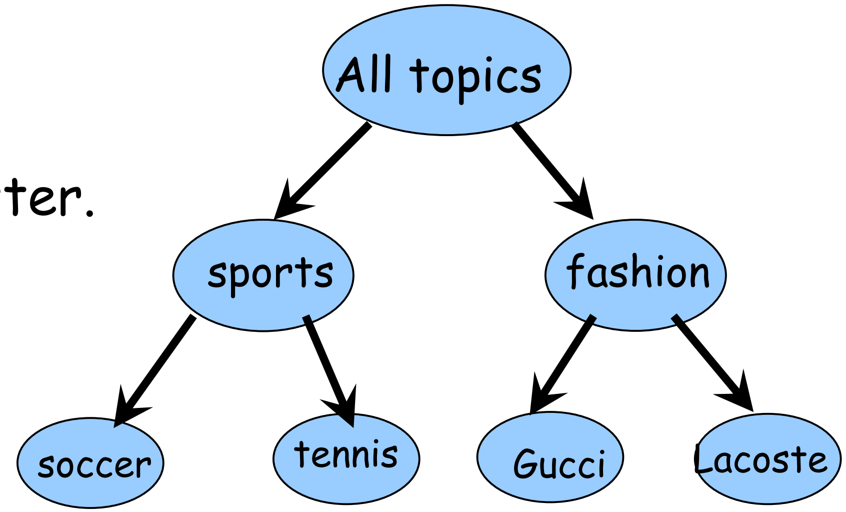




Linkage Procedures for Hierarchical Clustering

Bottom-Up (agglomerative)

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



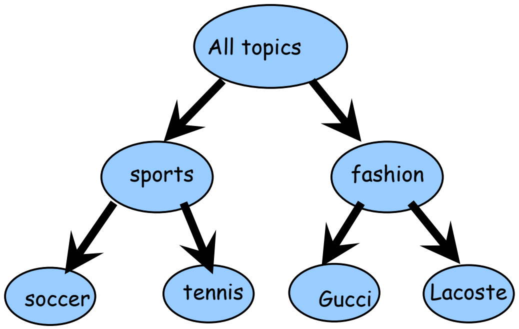
Different defs of "closest" give different algorithms.

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

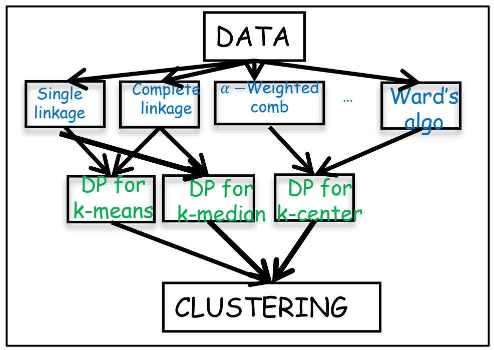


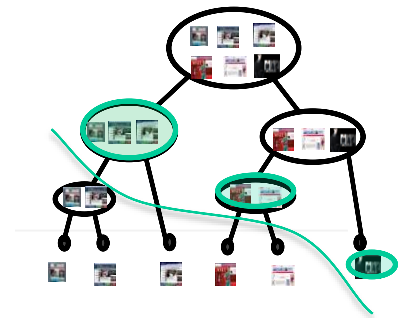
- Single linkage: $\text{dist}(A,B) = \min_{x \in A, x' \in B} \text{dist}(x,x')$
- Complete linkage: $\text{dist}(A,B) = \max_{x \in A, x' \in B} \text{dist}(x,x')$
- Average linkage: $\text{dist}(A,B) = \operatorname{avg}_{x \in A, x' \in B} \text{dist}(x,x')$
- Parametrized family, α-weighted linkage:

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

Clustering: Linkage + Dynamic Programming

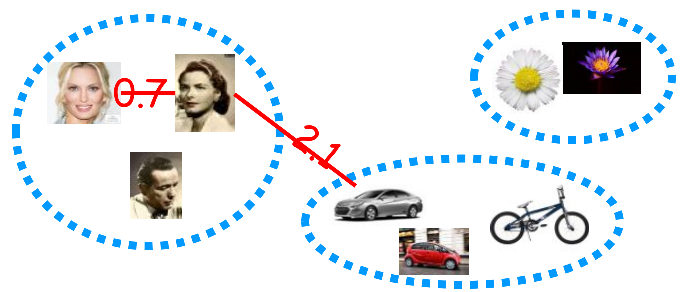
1. Use a linkage-based algorithm to get a hierarchy.
2. Dynamic programming to the best prunning.





- Used in practice.

E.g., [Filippova-Gadani-Kingsford, BMC Informatics]



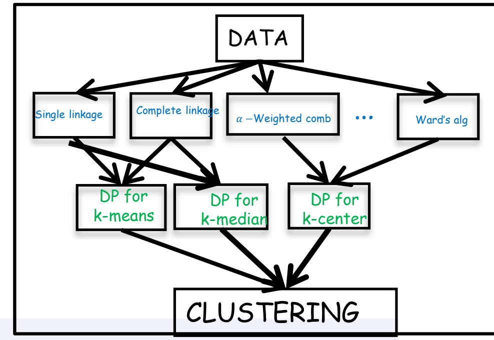
- Strong properties.

E.g., best known algos for perturbation resilient instances for k-median, k-means, k-center.

[Balcan-Liang, SICOMP 2016] [Awasthi-Blum-Sheffet, IPL 2011]
[Angelidakis-Makarychev-Makarychev, STOC 2017]

PR: small changes to input distances shouldn't move optimal solution by much.

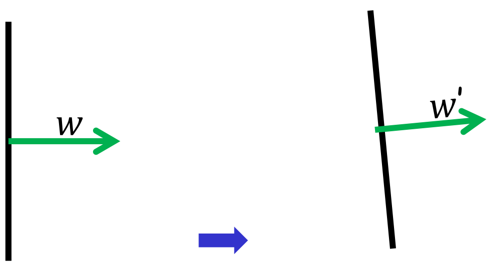
Clustering: Linkage + Dynamic Programming

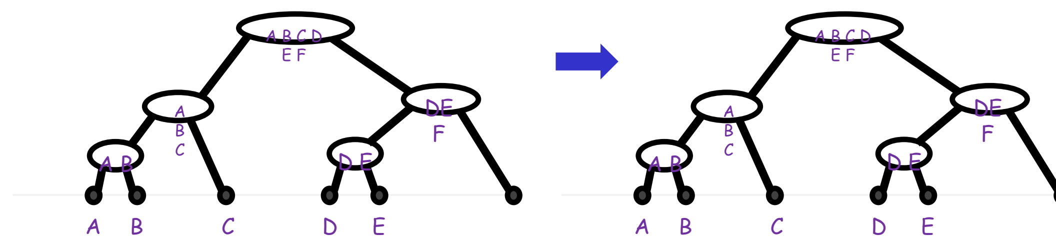


Our Results: α-weighted linkage+DP

- 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.



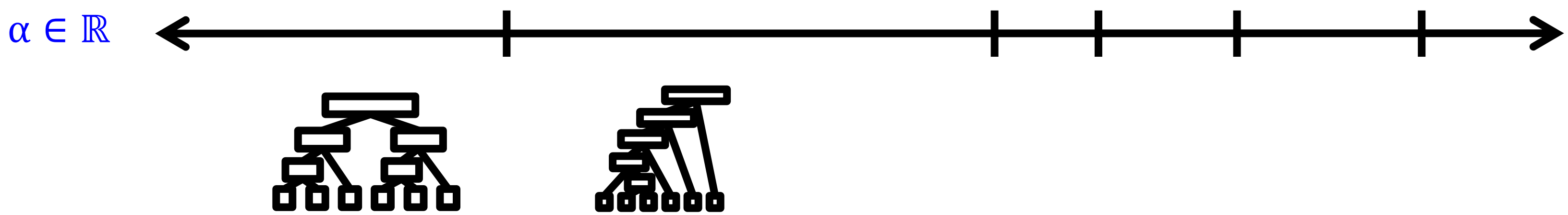


Problem: a single change to an early decision by the linkage algo, can snowball and produce large changes later on.

Clustering: Linkage + Dynamic Programming

Claim: Pseudo-dimension of α-weighted linkage + DP is $O(\log n)$, so small sample complexity.

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.

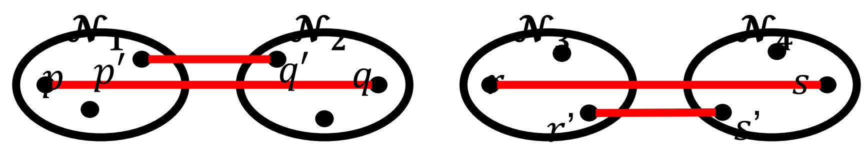
Clustering: Linkage + Dynamic Programming

Claim: Pseudo-dimension of α-weighted linkage + DP is $O(\log n)$, so small sample complexity.

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.

$\alpha \in \mathbb{R}$

Key idea:

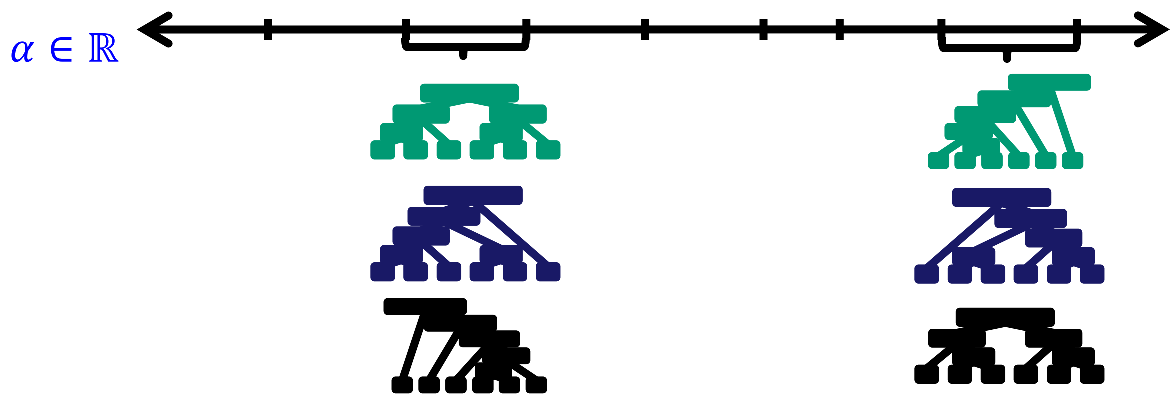


- For a given α, which will merge first, $\mathcal{N}_1$ and $\mathcal{N}_2$, or $\mathcal{N}_3$ and $\mathcal{N}_4$?
- Depends on which of $(1-\alpha)d(p,q)+\alpha d(p',q')$ or $(1-\alpha)d(r,s)+\alpha d(r',s')$ is smaller.
- An interval boundary an equality for 8 points, so $O(n^8)$ interval boundaries.

Clustering: Linkage + Dynamic Programming

Claim: Pseudo-dimension of α-weighted linkage + DP is O(log n), so small sample complexity.

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 + Dynamic Programming

Claim: Pseudo-dimension of α-weighted linkage + DP is $O(\log n)$, so small sample complexity.

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.

Algorithm

- Solve for all α intervals over the sample

- Find the α interval with the smallest empirical cost

Clustering: Linkage + Dynamic Programming

Claim: Pseudo-dimension of α-weighted linkage + DP is O(log n), so small sample complexity.

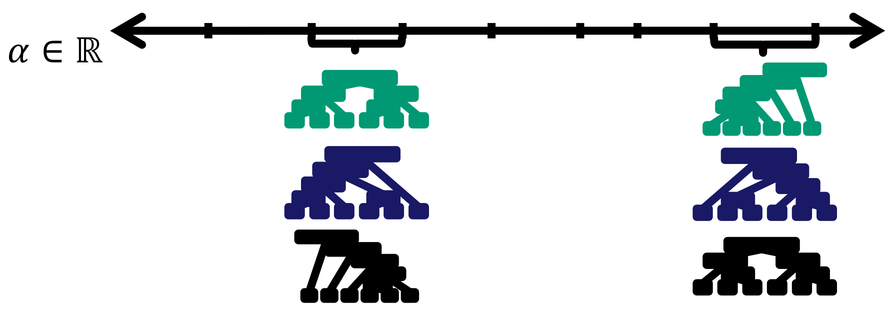
High level learning theory bit

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

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

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

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



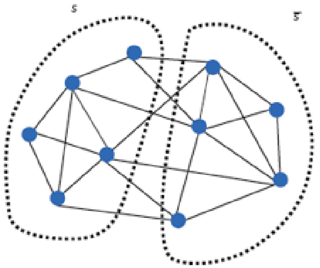
Partitioning Problems via IQPs

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

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

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

Partitioning Problems via IQPs

IQP formulation

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

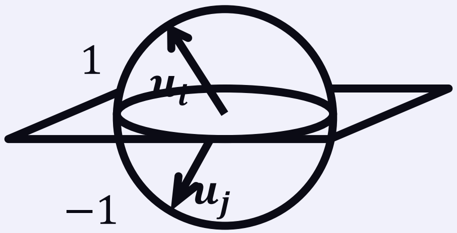
Algorithmic Approach: SDP + Rounding

1. Semi-definite programming (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. Rounding procedure [Goemans and Williamson '95]



- Choose a random hyperplane.
- (Deterministic thresholding.) Set x_i to -1 or 1 based on which side of the hyperplane the vector $\mathbf{u}_i$ falls on.

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

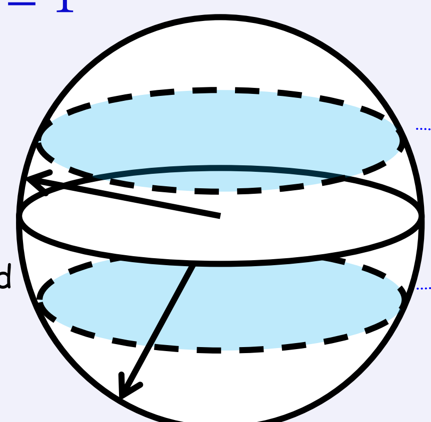
1. SDP relaxation:

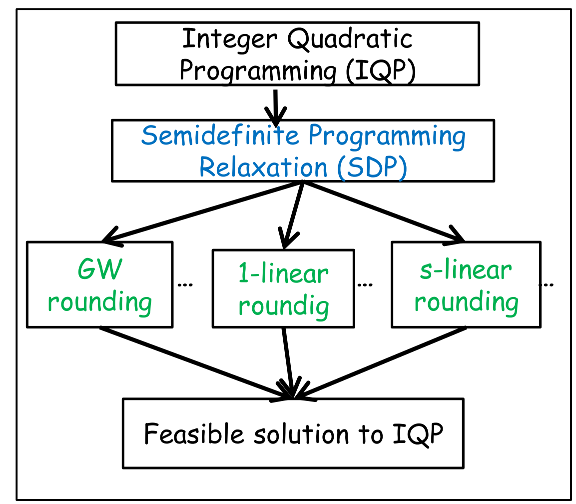
Associate each binary variable x_i with a vector $\mathbf{u}_i$.

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

2. s-Linear Rounding

[Feige&Landberg'06]



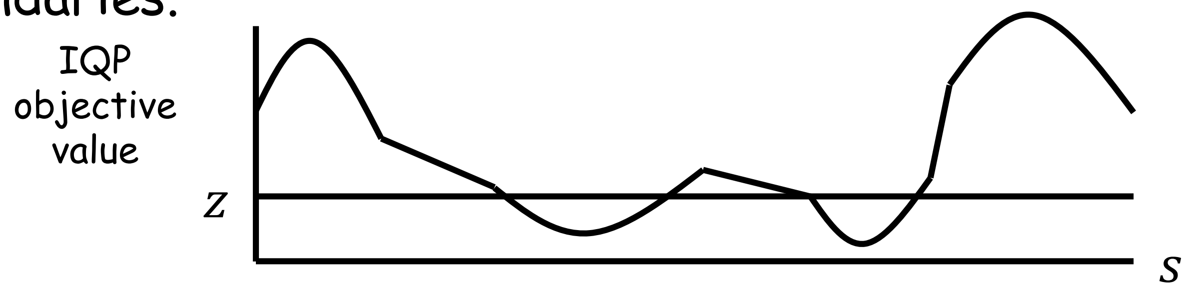


Partitioning Problems via IQPs

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.

- Solve for all α intervals over the sample, find best parameter over each interval, output best parameter overall.

Data driven mechanism design

- **Similar ideas** to provide sample complexity guarantees for **data-driven mechanism design** for revenue maximization for multi-item multi-buyer scenarios.

  [Balcan-Sandholm-Vitercik, EC'18]

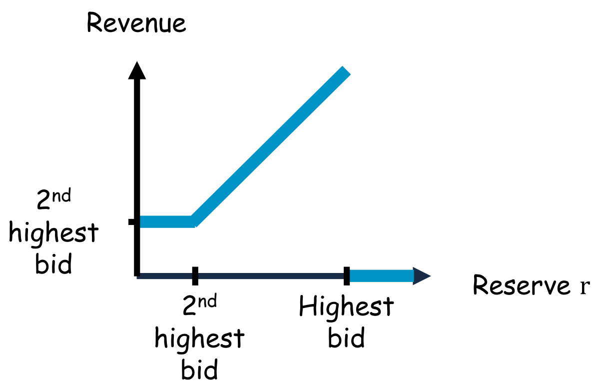
- Analyze pseudo-dim of $\{\text{revenue}_M : M \in \mathcal{M}\}$ for multi-item multi-buyer scenarios.
  - Many families: second-price auctions with reserves, posted pricing, two-part tariffs, parametrized VCG auctions, lotteries, etc.

Sample Complexity of data driven mechanism design

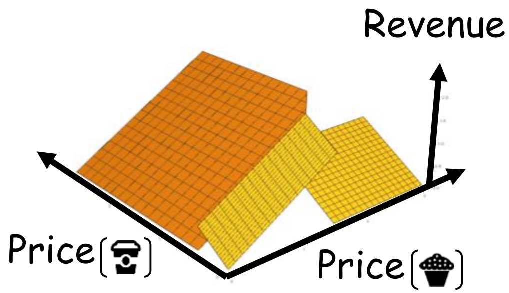
- Analyze pseudo-dim of $\{\mathrm{revenue}_M : M \in \mathcal{M}\}$ for multi-item multi-buyer scenarios. [Balcan-Sandholm-Vitercik, EC'18]
  - Many families: second-price auctions with reserves, posted pricing, two-part tariffs, parametrized VCG auctions, lotteries, etc.

- 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

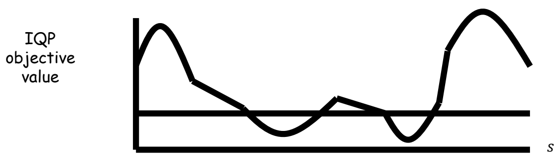


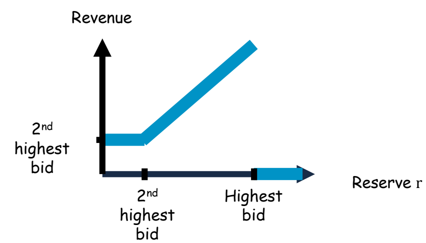
Structure of the Talk

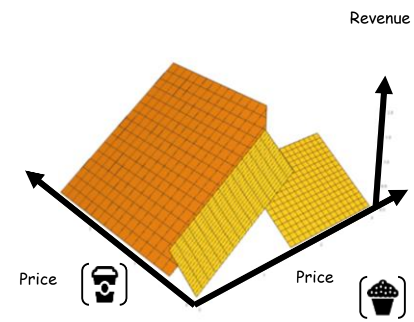
- Data driven algo design as batch learning.
    - A formal framework.
    - Case studies: clustering, partitioning pbs, auction problems.
- Data driven algo design via online learning.

Online Algorithm Selection

- So far, batch setting: collection of typical instances given upfront.
- [Balcan-Dick-Vitercik, FOCS 2018] online and private 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

Online Regret Guarantees

Existing techniques (for finite, linear, or convex case): select $\boldsymbol{\rho}_t$ probabilistically based on performance so far.

- Probability exponential in performance [Cesa-Bianchi and Lugosi 2006]
- Regret guarantee: $\max_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] = \tilde{\boldsymbol{O}}(\sqrt{\boldsymbol{T}} \times \cdots)$

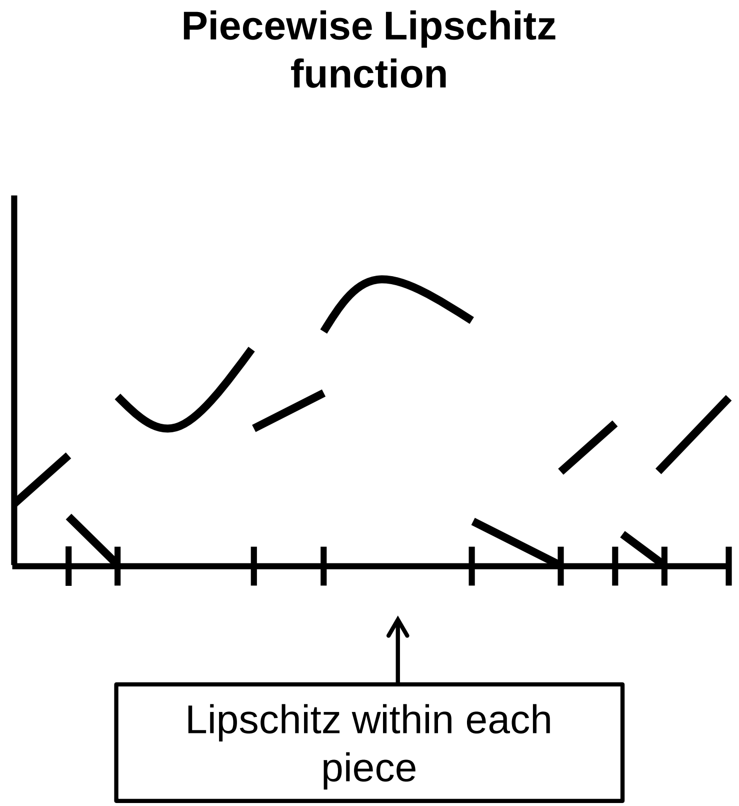
No-regret: per-round regret approaches 0 at rate $\tilde{\boldsymbol{O}}(\mathbf{1}/\sqrt{\boldsymbol{T}})$.

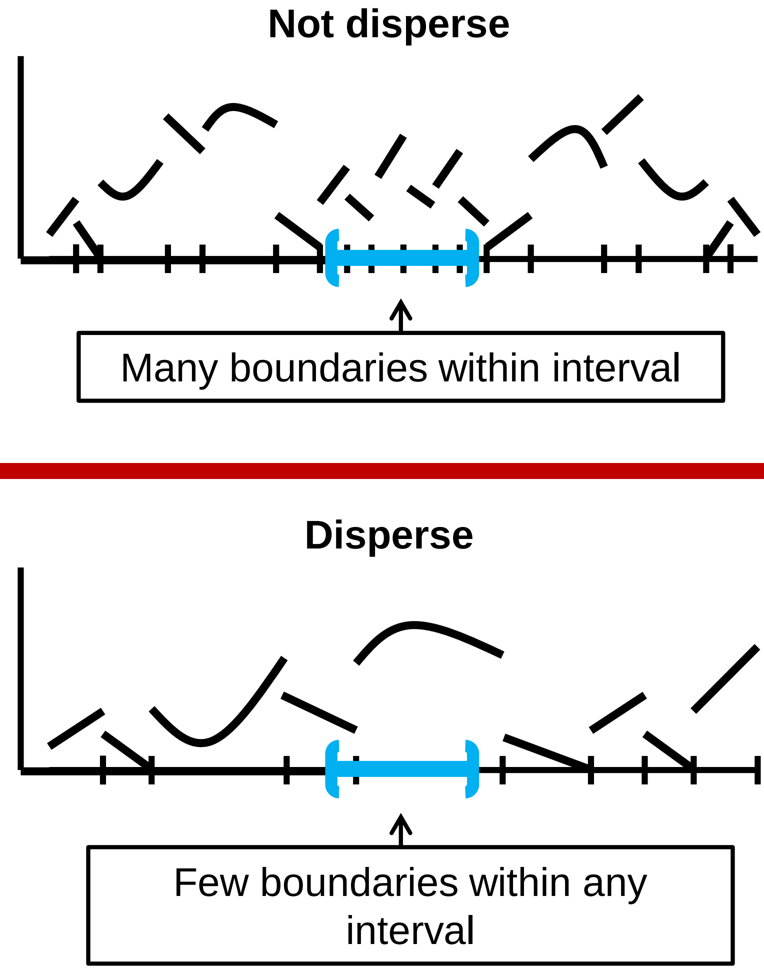
Challenge: if discontinuities, cannot get no-regret.

- Adversary can force online algo to "play 20 questions" while hiding an arbitrary real number.
  - Round 1: adversary splits parameter space in half and randomly chooses one half to perform well, other half to perform poorly.
  - Round 2: repeat on parameters that performed well in round 1. Etc.
  - Any algorithm does poorly half the time in expectation but ∃ perfect $\boldsymbol{\rho}$.

To achieve low regret, need structural condition.

Dispersion, Sufficient Condition for No-Regret





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

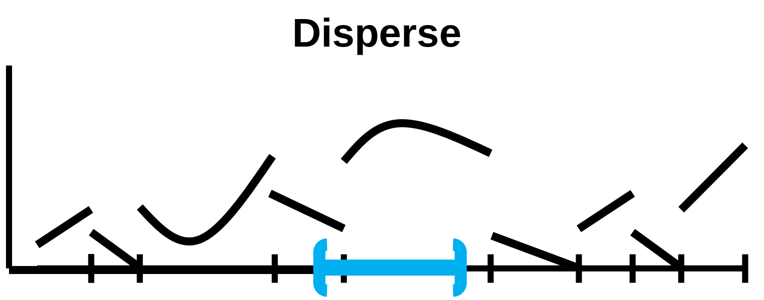
Dispersion, Sufficient Condition for No-Regret

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

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

- 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 $\tilde{O}\left(\sqrt{Td}\ \text{fnc of problem}\right)$.

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: Regret $\tilde{O}(\sqrt{Td}\text{ fnc of problem})$.

If $\sum_{t=1}^{T} u_t(\cdot)$ piecewise L-Lipschitz, $\{u_1(\cdot), \ldots, u_T(\cdot)\}$ is $(\mathbf{w}, \mathbf{k})$-dispersed.

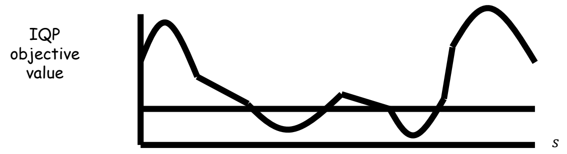
The expected regret is $O\left(H\left(\sqrt{Td\log\frac{1}{\mathbf{w}}} + \mathbf{k}\right) + TL\mathbf{w}\right)$.

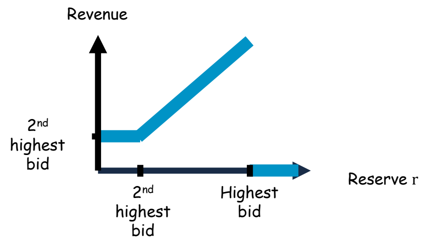
For most problems:

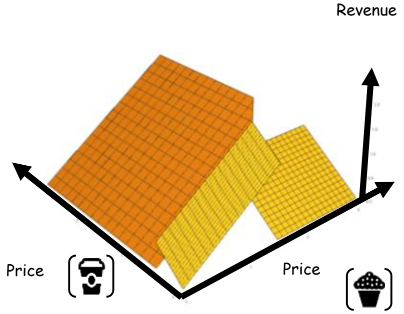
- Set $\mathbf{w} \approx 1/\sqrt{T}$, $\mathbf{k} = \sqrt{T} \times (\text{fnc of problem})$

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.







- Also differential privacy bounds.
- Learning theory: techniques of independent interest beyond algorithm selection.

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.

Future Work:

- Analyze other widely used classes of algorithms.
  - Branch and Bound Techniques for MIPs [Balcan-Dick-Sandholm-Vitercik, ICML'18]
  - Parametrized Lloyds methods [Balcan-Dick-White, NIPS'18]
- Related in spirit to Hyperparameter tuning, AutoML, MetaLearning.

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