An Online Learning Approach to Data-driven Algorithm Design

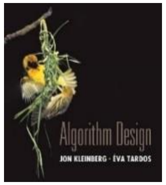
Maria-Florina (Nina) Balcan
Carnegie Mellon University

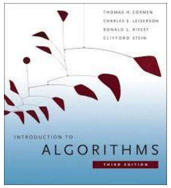
Analysis and Design of Algorithms

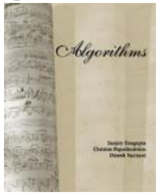
Classic algo design: solve a worst case instance.

- 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.

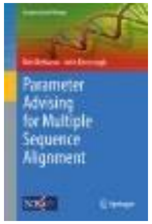
Data Driven Algorithm Design

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.

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

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

Algorithm Design as Distributional Learning

Goal: given large family of algos, sample of typical instances from domain, find an algo that performs well on new instances from same domain.

[Gupta-Roughgarden, ITCS'16 &SICOMP'17]

Large family F of algorithms

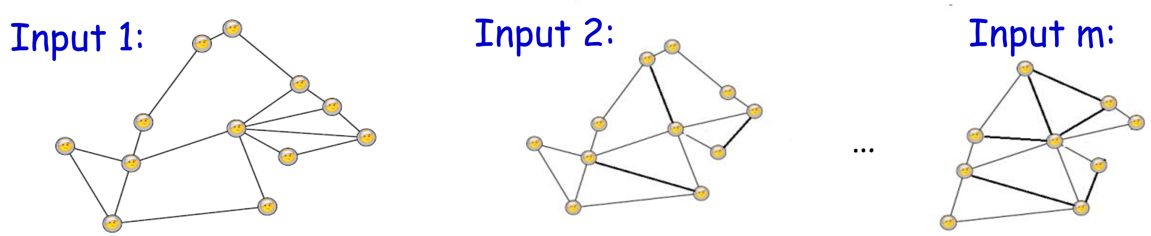
MST + Dynamic Programming

Greedy + Farthest Location

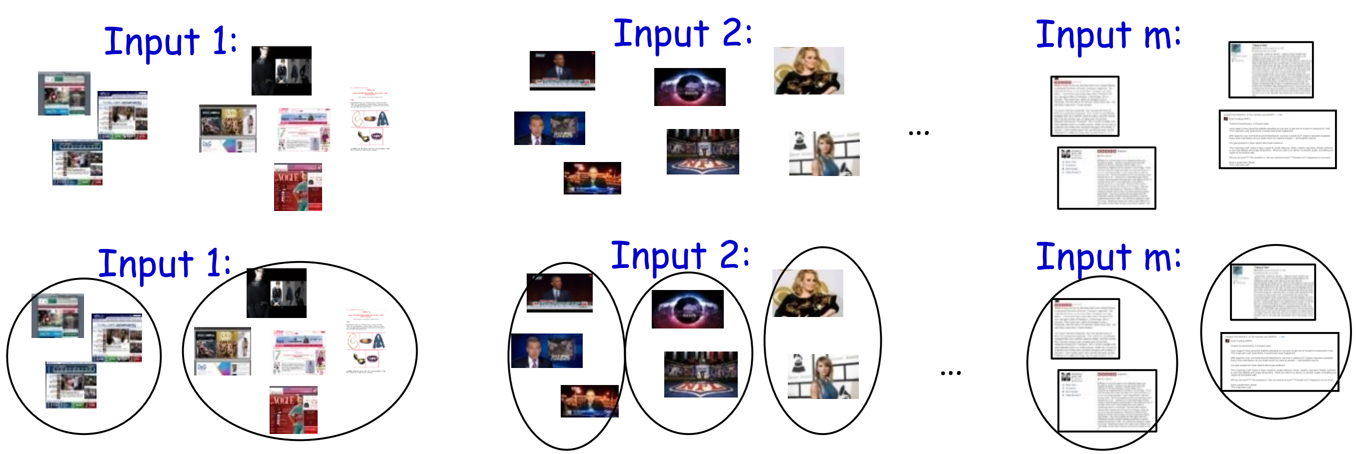
...

Sample of i.i.d. typical inputs

Facility location:



Clustering:

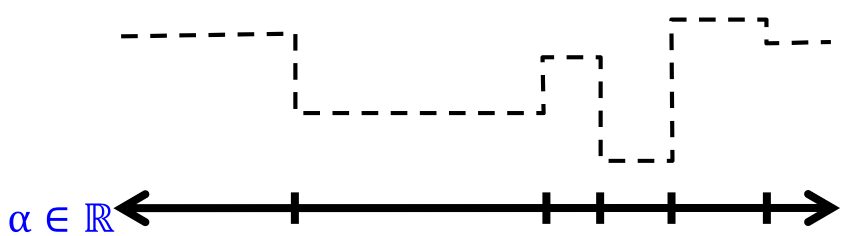


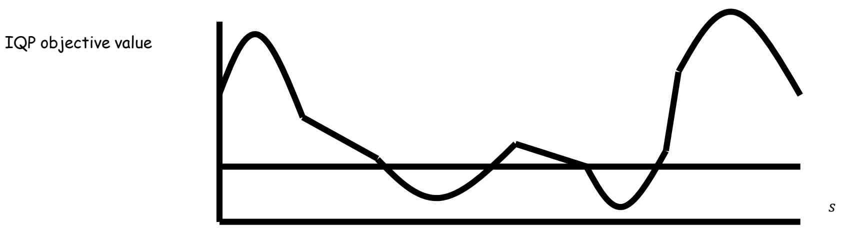
Statistical Learning Approach to AAD

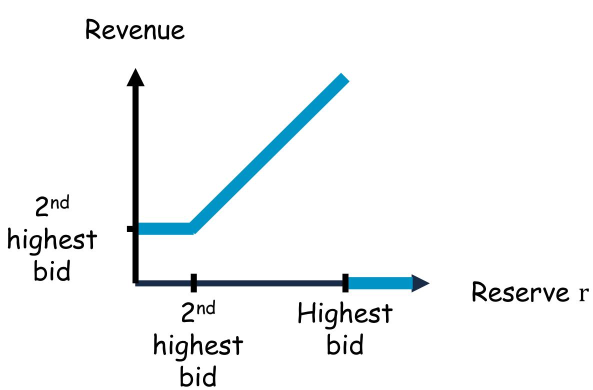
Sample Complexity: How large should our sample of typical instances be in order to guarantee good performance on new instances?

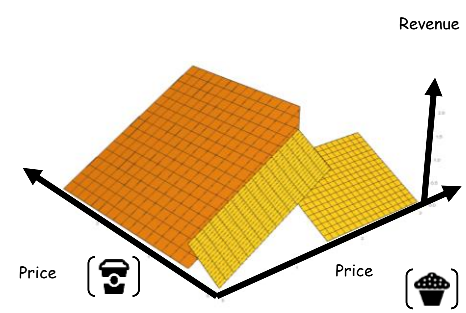
$m = O(\dim(F)/\epsilon)$ instances suffice to ensure generalizability

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









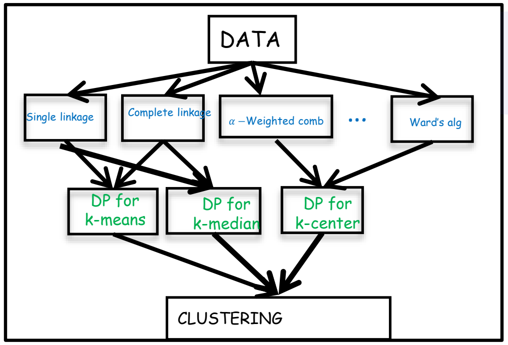
Algorithm Design as Distributional Learning

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

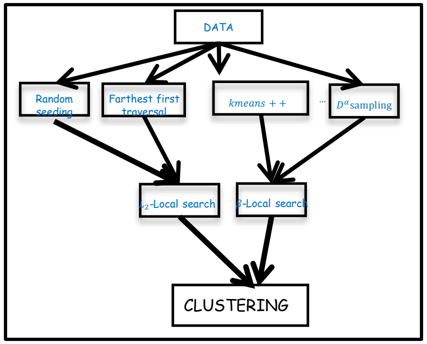
[Balcan-Nagarajan-Vitercik-White, COLT 2017]



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

Parametrized Lloyds

[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]

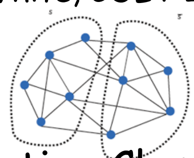
Algorithm Design as Distributional Learning

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

- **Partitioning pbs via IQPs:** SDP + Rounding

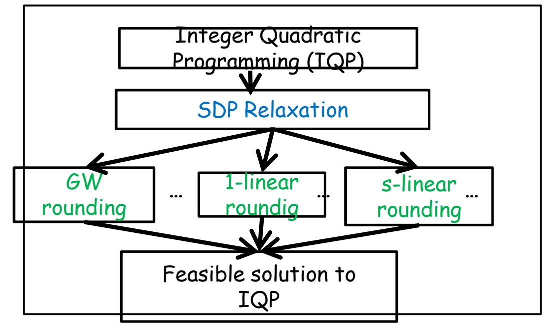
  [Balcan-Nagarajan-Vitercik-White, COLT 2017]

  E.g., Max-Cut,



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

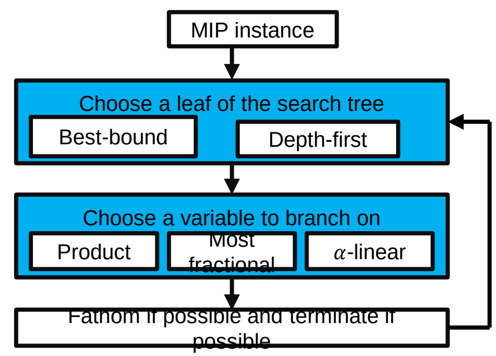
  Max-2SAT, Correlation Clustering



- **MIPs:** Branch and Bound Techniques

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

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



- **Automated mechanism design for revenue maximization**

  Parametrized VCG auctions, posted prices, lotteries.

  [Balcan-Sandholm-Vitercik, EC 2018]

Algorithm Design as Distributional Learning

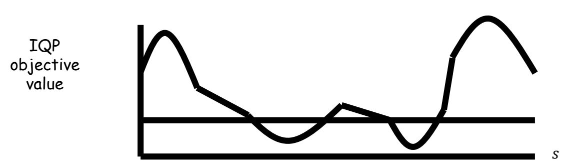
Our results: General sample complex. tools via structure of dual class

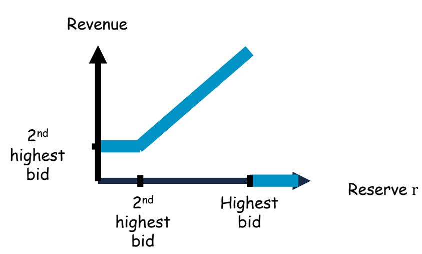
[Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, 2019]

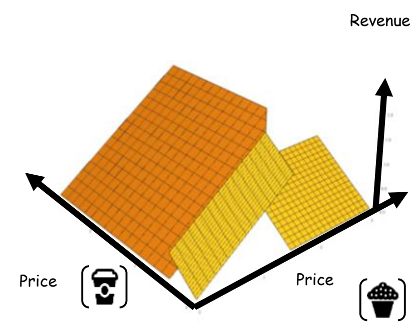
Thm: Suppose for each $\text{cost}_I(\alpha)$ there are $\leq N$ boundary fns $f_1, f_2, \ldots \in F$ s. t within each region defined by them, $\exists\, g \in G$ s.t. $\text{cost}_I(\alpha) = g(\alpha)$.

$$\text{Pdim}(\{\text{cost}_\alpha(I)\}) = O\big((d_{F^*} + d_{G^*}) \log(d_{F^*} + d_{G^*}) + d_{F^*} \log N\big)$$

d_{F^*} =VCdim of dual of F, d_{G^*} =Pdim of dual of G.



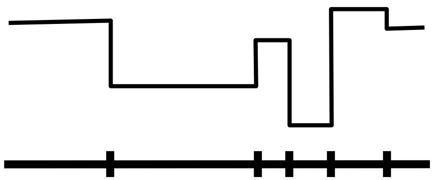


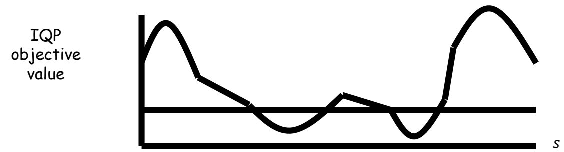


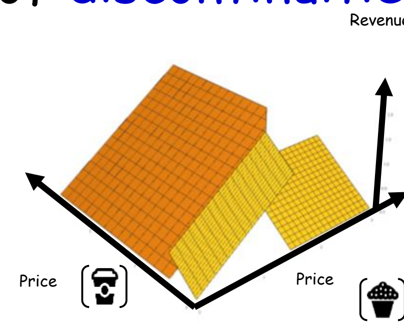
Online Algorithm Selection

- 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.

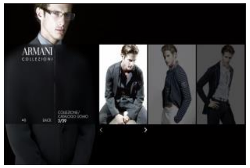
Structure of the Talk

- Overview. Data driven algo design as batch learning.
- An example: clustering via parametrized linkage.
- 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.

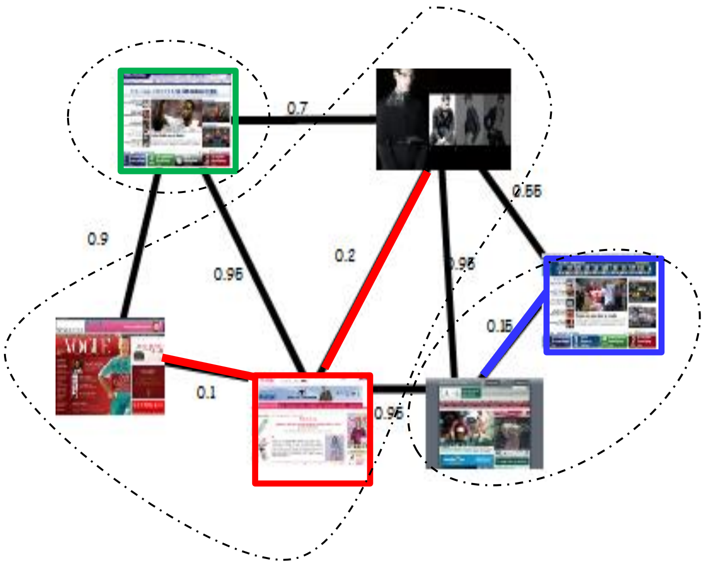
Objective based clustering

***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 $\sum_p \min d(p, c_i)$.

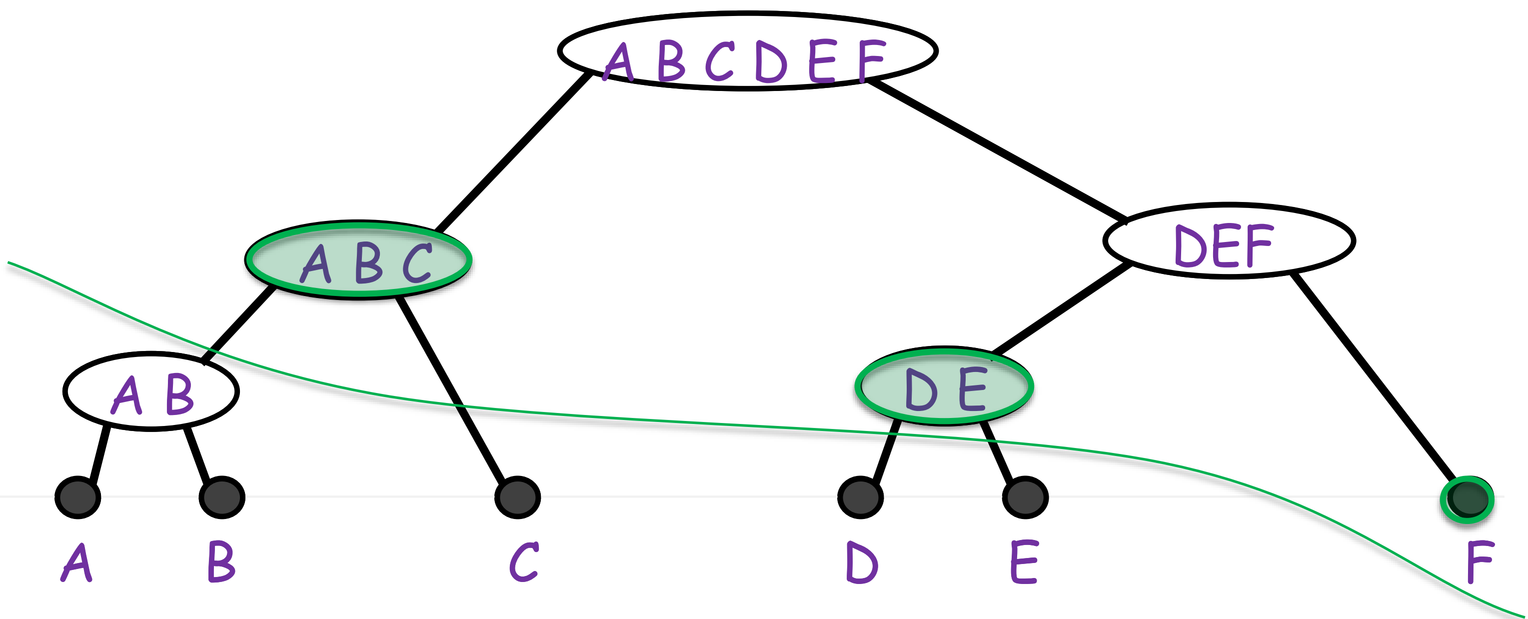
k-center/facility location: minimize the maximum radius.

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

Clustering: Linkage + Dynamic Programming

Family of poly time 2-stage algorithms:

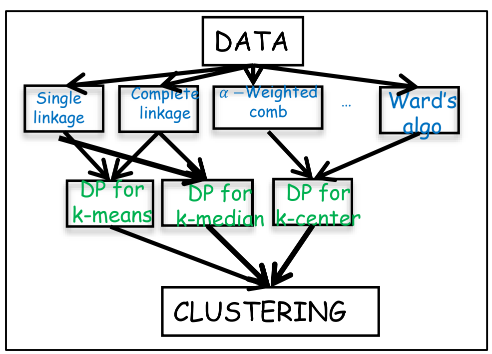
1. Use a greedy linkage-based algorithm to organize data into a hierarchy (tree) of clusters.
2. Dynamic programming over this tree to identify pruning of tree corresponding to the best clustering.

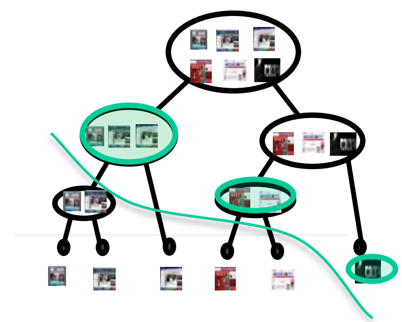


Clustering: Linkage + Dynamic Programming

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

Both steps can be done efficiently.

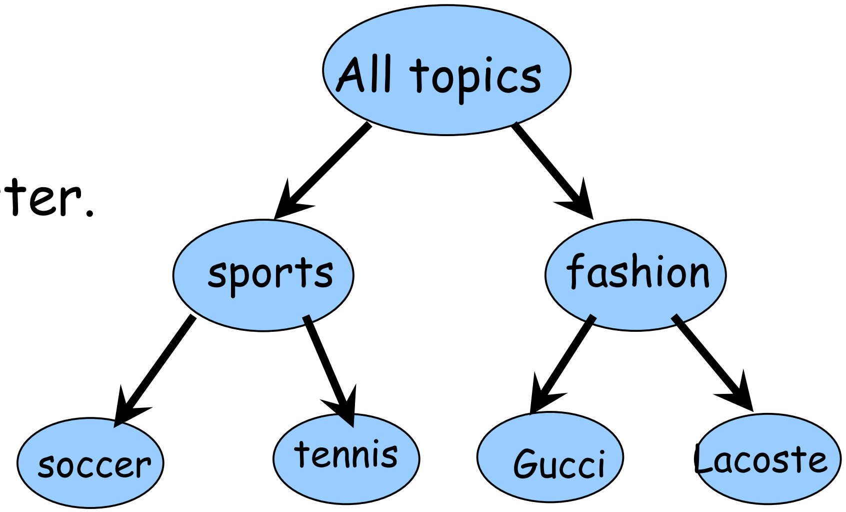




Linkage Procedures for Hierarchical Clustering

Bottom-Up (agglomerative)

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



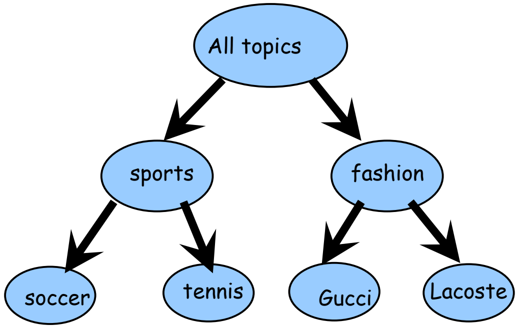
Different defs of "closest" give different algorithms.

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

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

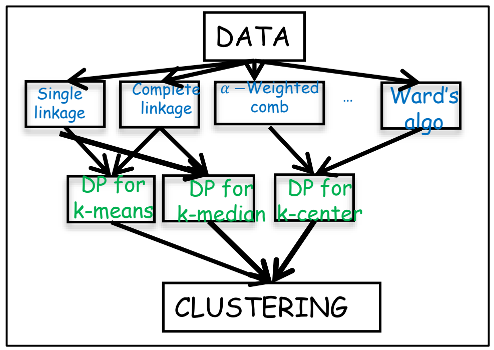
- Average linkage: $\mathrm{dist}(A,B) = \operatorname{avg}_{x \in A, x' \in B} \mathrm{dist}(x, x')$

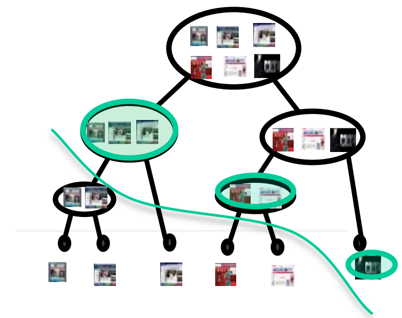
- Parametrized family, α-weighted linkage:

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

Clustering: Linkage + Dynamic Programming

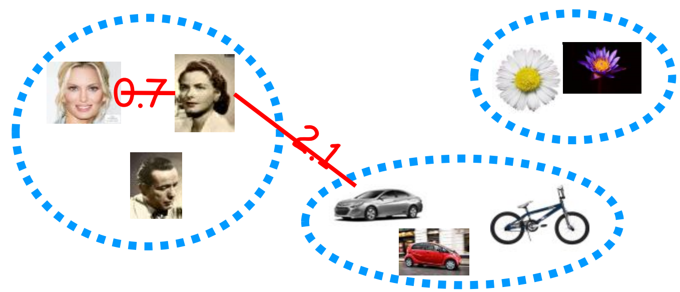
1. Use a linkage-based algorithm to get a hierarchy.
2. Dynamic programming to the best prunning.





- Used in practice.

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



- Strong properties.

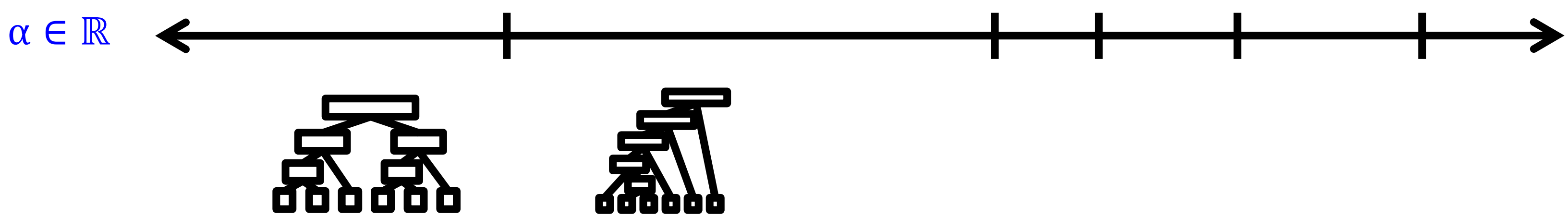
E.g., best known algos for perturbation resilient instances for k-median, k-means, k-center.

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

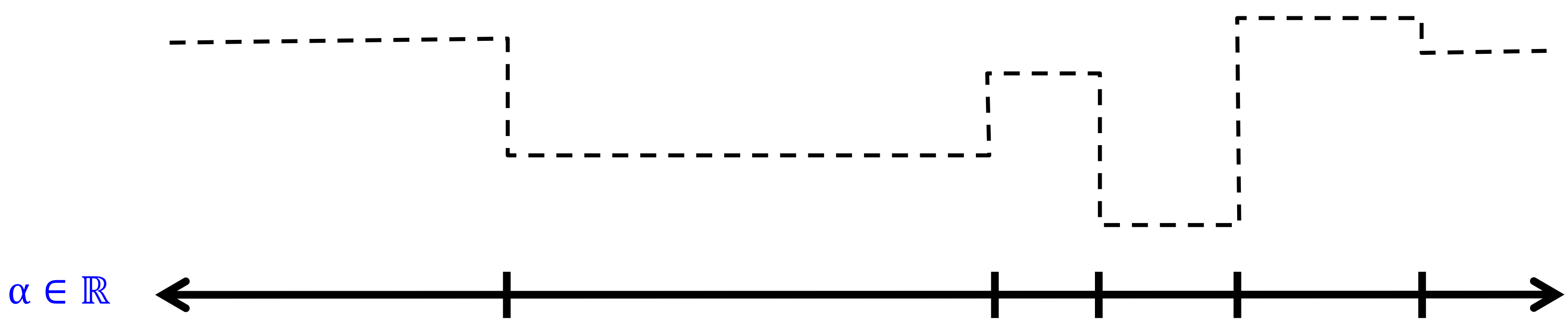
PR: small changes to input distances shouldn't move optimal solution by much.

Clustering: Linkage + Dynamic Programming

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.

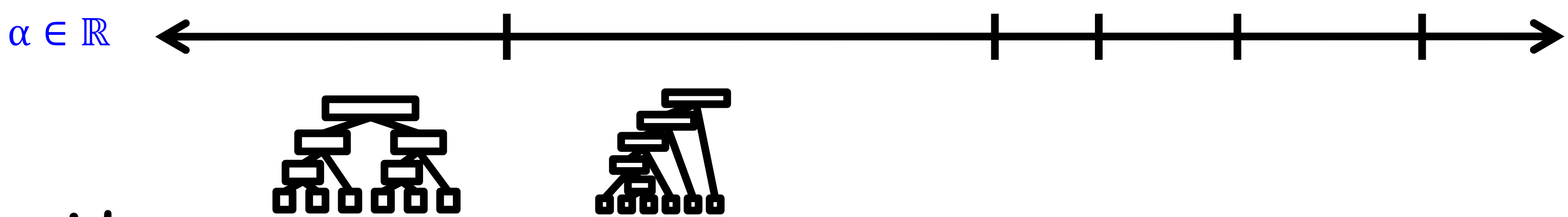


So, the cost function is piecewise-constant with at most $O(n^8)$ pieces.



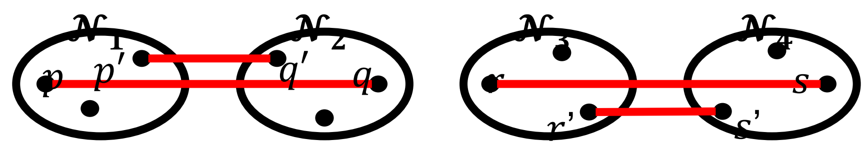
Clustering: Linkage + Dynamic Programming

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 $\mathcal{N}_4$?

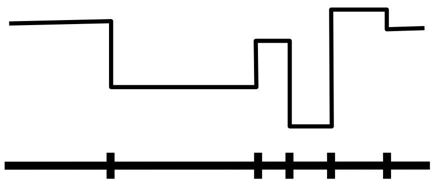


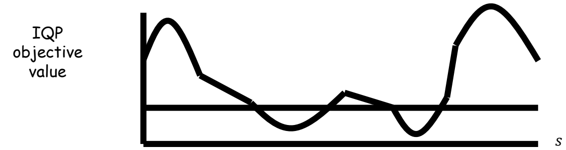
- Depends on which of $(1-\alpha)d(p,q) + \alpha d(p',q')$ or $(1-\alpha)d(r,s) + \alpha d(r',s')$ is smaller.

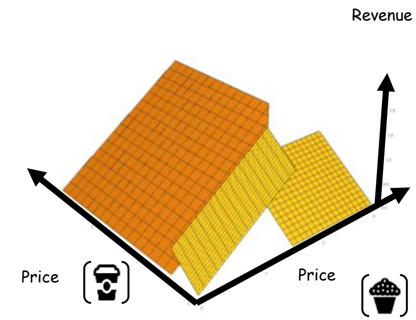
- An interval boundary an equality for 8 points, so $O(n^8)$ interval boundaries.

Online Algorithm Selection

- [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 $u_t(\boldsymbol{\rho}_t)$.

3. Get **feedback:** Full information: observe the function $u_t(\cdot)$

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

Goal: **minimize regret**: $\underbrace{\max_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho})}_{\text{Performance of best parameter in hindsight}} - \mathbb{E}\Big[\underbrace{\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)}_{\text{Our cumulative performance}}\Big]$

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: $\max_{\boldsymbol{\rho}\in\mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] = \tilde{\mathbf{O}}\left(\sqrt{\mathbf{T}} \times \cdots\right)$

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

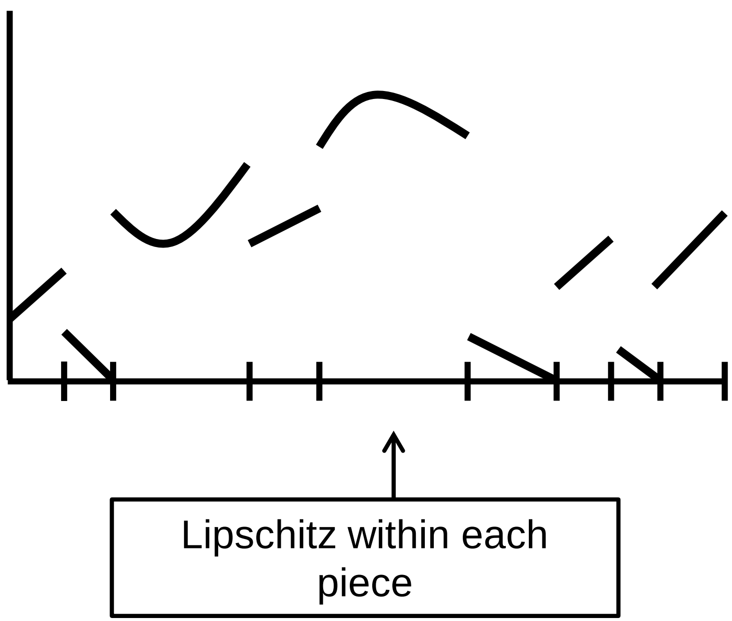
Challenge: if discontinuities, cannot get no-regret.

- 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 ρ.

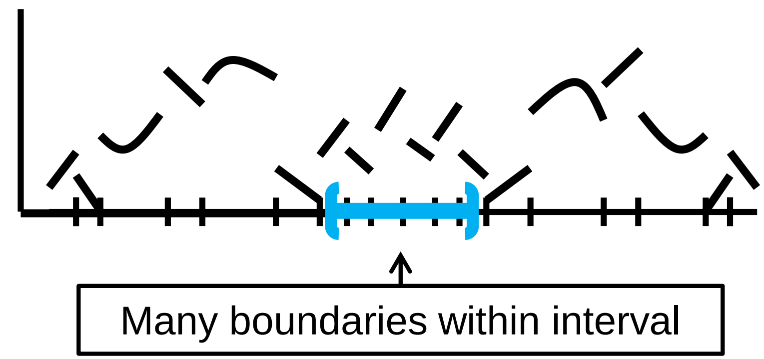
To achieve low regret, need structural condition.

Dispersion, Sufficient Condition for No-Regret

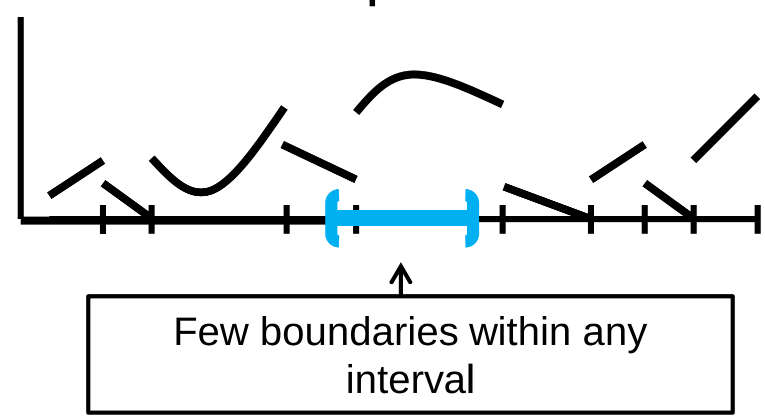
Piecewise Lipschitz function



Not disperse



Disperse



$\{u_1(\cdot),\ldots,u_T(\cdot)\}$ 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

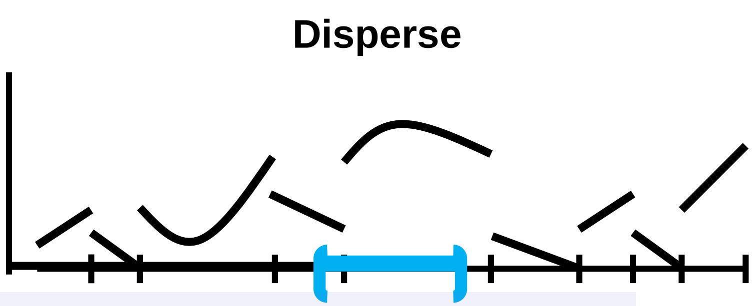
Full info: exponentially weighted forecaster [Cesa-Bianchi-Lugosi 2006]

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

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

density of $\boldsymbol{\rho}$ exponential in its performance so far

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 $t \in \{1, \dots, T\}$:

- Sample a $\boldsymbol{\rho}_t$ from distr. p_t: $p_t(\boldsymbol{\rho}) = \frac{\exp(\lambda U_t(\boldsymbol{\rho}))}{\boldsymbol{W}_t}$

$U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ total payoff of $\boldsymbol{\rho}$ up to time t

$W_t = \int \exp\left(\lambda U_t(\boldsymbol{\rho})\right) d\boldsymbol{\rho}$

Our Results:

Assume $\{u_1(\cdot), \dots, u_T(\cdot)\}$ are piecewise L-Lipschitz and $(\mathbf{w}, \mathbf{k})$-dispersed.

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

For most problems:

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

Dispersion, Sufficient Condition for No-Regret

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

- Sample a $\boldsymbol{\rho}_t$ from distr. p_t: $p_t(\boldsymbol{\rho}) = \frac{\exp(\lambda U_t(\boldsymbol{\rho}))}{\mathbf{W}_t}$ $U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ total payoff of $\boldsymbol{\rho}$ up to time t

Thm: $\{u_1(\cdot), \dots, u_T(\cdot)\}$ are piecewise L-Lipschitz and $(\mathbf{w}, \mathbf{k})$-dispersed.

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

Key idea: Provide upper and lower bounds on $\frac{W_{T+1}}{W_1}$, where W_t norm. factor $W_t = \int \exp\left(\lambda U_t(\boldsymbol{\rho})\right) d\boldsymbol{\rho}$

- Upper bound: classic, relate algo perfor. with magnitude of update:

$$\frac{W_{t+1}}{W_t} \leq \exp\left(\frac{(e^{H\lambda} - 1)P_t}{H}\right)$$

P_t = expected alg payoff at time t.

$$\frac{W_{T+1}}{W_1} \leq \exp\left(\frac{(e^{H\lambda} - 1)P_{ALG}}{H}\right)$$

P_{ALG} = expected alg total payoff.

Dispersion, Sufficient Condition for No-Regret

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

- Sample a $\boldsymbol{\rho}_t$ from distr. p_t: $p_t(\boldsymbol{\rho}) = \frac{\exp(\lambda U_t(\boldsymbol{\rho}))}{W_t}$ $\quad U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ total payoff of $\boldsymbol{\rho}$ up to time t

Thm: $\{u_1(\cdot), \dots, u_T(\cdot)\}$ are piecewise L-Lipschitz and $(\mathbf{w}, \mathbf{k})$-dispersed.

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

Key idea: analyze $W_t = \int \exp\left(\lambda U_t(\boldsymbol{\rho})\right) d\boldsymbol{\rho}$ (norm. factor)

- Lower bound (uses dispersion + Lipschitz):

Let $\boldsymbol{\rho}^*$ be the optimal parameter vector in hindsight.

For all $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$, $\quad \sum_{t=1}^{T} u_t(\boldsymbol{\rho}) \geq \sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*) - Hk - TLw.$

remaining fns L-Lipschitz

$\leq k$ fncs u_s have disc. in $B(\boldsymbol{\rho}^*, w)$ & range is $[0, H]$

For all $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$, $\quad U_{T+1}(\boldsymbol{\rho}) \geq U_{T+1}(\boldsymbol{\rho}^*) - Hk - TLw.$

Dispersion, Sufficient Condition for No-Regret

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

- Sample a $\boldsymbol{\rho}_t$ from distr. p_t: $p_t(\boldsymbol{\rho}) = \frac{\exp(\lambda U_t(\boldsymbol{\rho}))}{\mathbf{W}_t}$ $U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ total payoff of $\boldsymbol{\rho}$ up to time t

Thm: $\{u_1(\cdot), \ldots, u_T(\cdot)\}$ are piecewise L-Lipschitz and $(\mathbf{w}, \mathbf{k})$-dispersed.

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

Key idea: analyze $W_t = \int \exp\left(\lambda U_t(\boldsymbol{\rho})\right) d\boldsymbol{\rho}$ (norm. factor)

- Lower bound

For all $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$, $U_{T+1}(\boldsymbol{\rho}) \geq U_{T+1}(\boldsymbol{\rho}^*) - Hk - TLw$.

$$W_{T+1} \geq \mathrm{Vol}\left(B(\boldsymbol{\rho}^*, w)\right) \exp(\lambda(U_{T+1}(\boldsymbol{\rho}^*) - Hk - TLw))$$

Get $$\frac{W_{T+1}}{W_1} \geq \left(\frac{w}{R}\right)^d \exp(\lambda(U_{T+1}(\boldsymbol{\rho}^*) - Hk - TLw))$$

Dispersion, Sufficient Condition for No-Regret

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

- Sample a $\boldsymbol{\rho}_t$ from distr. p_t: $p_t(\boldsymbol{\rho}) = \frac{\exp(\lambda U_t(\boldsymbol{\rho}))}{\mathbf{W}_t}$ $\quad U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ total payoff of $\boldsymbol{\rho}$ up to time t

Thm: $\{u_1(\cdot), \dots, u_T(\cdot)\}$ are piecewise L-Lipschitz and $(\mathbf{w}, \mathbf{k})$-dispersed.

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

Key idea: analyze $W_t = \int \exp\left(\lambda U_t(\boldsymbol{\rho})\right) d\boldsymbol{\rho}$ (norm. factor)

$$\exp\left(\frac{(e^{H\lambda} - 1)P_{ALG}}{H}\right) \geq \left(\frac{w}{R}\right)^d \exp(\lambda(U_{T+1}(\boldsymbol{\rho}^*) - Hk - TLw))$$

Take logs, use std approx of e^z and fact that $P_{ALG} \leq HT$:

$$U_{T+1}(\rho^*) - P_{ALG} \leq H^2 T\lambda + \frac{d\log R/w}{\lambda} + Hk + TLw$$

Set $\lambda = \frac{1}{H}\sqrt{\frac{d}{T}\log\left(\frac{R}{w}\right)}$ $\qquad$ Regret $O\left(H\left(\sqrt{Td\log\frac{R}{\mathbf{w}}} + \mathbf{k}\right) + TL\mathbf{w}\right)$

Example: Clustering with ρ-weighted linkage

ρ-weighted linkage: $\mathrm{dist}(A,B) = \rho \min_{x\in A, x'\in B} \mathrm{dist}(x,x') + (1-\rho) \max_{x\in A, x'\in B} \mathrm{dist}(x,x')$

Theorem: If T instances with distances selected in $[0,B]$ from κ-bounded densities, then for any w, with prob $\geq 1-\delta$, we get (w,k)-dispersion for $k = O(wn^8\kappa^2B^2T) + O\left(\sqrt{T\log(n/\delta)}\right)$.

For any **given** interval I, expected #instances with discontinuities in I is at most this

From a uniform convergence argument

Implies expected regret of $O\left(H\sqrt{T\log(Tn\kappa B)}\right)$

Example: Clustering with ρ-weighted linkage

ρ-weighted linkage: $\mathrm{dist}(A,B) = \rho \min_{x\in A, x'\in B} \mathrm{dist}(x,x') + (1-\rho) \max_{x\in A, x'\in B} \mathrm{dist}(x,x')$

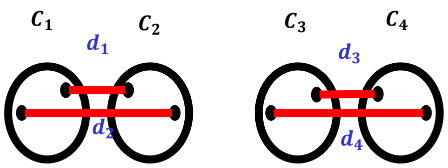
Theorem: If T instances with distances in $[0,B]$ from κ-bounded densities, then for any w, with prob $\geq 1-\delta$ get (w,k)-dispersion for $k = O(wn^8\kappa^2B^2T) + O\left(\sqrt{T\log(n/\delta)}\right)$.

Key ideas:

- Each instance $O(n^8)$ discontinuities of form $\dfrac{d_1 - d_3}{(d_1 - d_3) + (d_4 - d_2)}$

Given ρ, merge first, C_1 and C_2, or C_3 and C_4?

Critical value when $(1-\rho)d_1 + \rho d_2 = (1-\rho)d_3 + \rho d_4$



- Disc from $O\left((\kappa B)^2\right)$-bounded density.

- So, for any given interval I of width w, expected # instances with discontinuities in I is $O(wn^8\kappa^2B^2T)$. Uniform convergence on the top.

Proving Dispersion

- Use functional form of discontinuities to get distribution of discontinuity locations, and expected number in any given interval.
    - Randomness either from input instances (smoothed model) or from randomness in the algorithms themselves
- Use uniform convergence result on the top.

Lemma: Let $u_1, \ldots, u_T: \mathbb{R} \to \mathbb{R}$ be independent piecewise-Lipschitz, each with $\leq N$ discontinuities. Let interval I, let $D(I)$ be the # of u_i with at least one discontinuity in I. Then w.h.p $\geq 1 - \delta$:

$$\sup_I \left(D(I) - E[D(I)]\right) \leq O\left(\sqrt{T \log(N/\delta)}\right)$$

Proof Sketch:

- For interval I, let $f_I(i) = 1$ iff u_i has discontinuity in I.

Then $D(I) = \sum_i f_I(i)$

- Show VC-dim of $\mathcal{F} = \{f_I\}$ is $O(\log N)$, then use uniform convergence.

Proof: VC-dimension of F is $O(\log \mathrm{N})$

Lemma: For interval I, set s of $\leq \mathrm{N}$ points, $\mathrm{f}_{\mathrm{I}}(\mathrm{s}) = 1$ iff s has a point inside I. VC-dimension of class $\mathcal{F} = \{\mathrm{f}_{\mathrm{I}}\}$ is $\mathrm{O}(\log \mathrm{N})$.

Proof idea:

How many ways can $\mathcal{F}$ label d instances (d sets of $\leq \mathrm{N}$ pts) $\mathrm{s}_1, \ldots, \mathrm{s}_{\mathrm{d}}$?

- d instances have at most Nd points in their union.
- For $\mathrm{f}_{\mathrm{I}}, \mathrm{f}_{\mathrm{I}'}$ to label the instances differently, minimal req. is that intervals I and I' can't cover the exact same points.
- But, given Nd points, at most $(\mathrm{Nd})^2 + 1$ different subsets of them that can be covered using intervals.

To shatter d instances, must have $2^{\mathrm{d}} \leq (\mathrm{Nd})^2 + 1$. VC-dim $\mathrm{d} = \mathrm{O}(\log \mathrm{N})$.

Example: Knapsack

A problem instance is collection of items i, each with a value v_i and a size s_i, along with a knapsack capacity C.

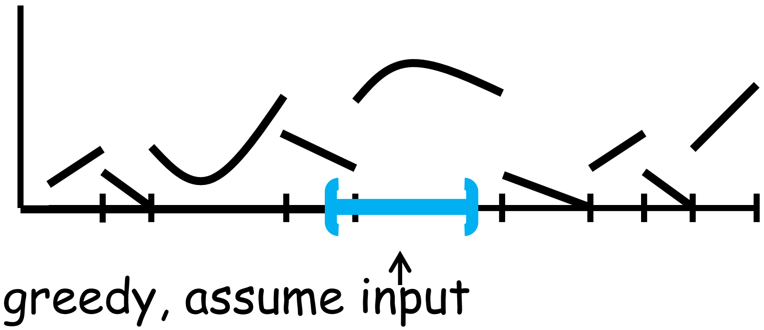
Goal: select most valuable subset of items that fits. Find subset V to maximize $\sum_{i \in V} v_i$ subject to $\sum_{i \in V} s_i \leq C$.

Greedy family of algorithms:

- Greedily pack items in order of value (highest value first) subject to feasibility.
- Greedily pack items in order of v_i / s_i^{ρ} subject to feasibility.

Thm: *Adversary chooses knapsack instances with n items, capacity C, sizes s_i in $[1, C]$, values v_i in $(0,1]$, and item values drawn independently from b-bounded distributions. Utilities $u_1, \ldots, u_T$ are piecewise constant and w.h.p $\geq 1 - \delta$, (w, k)-dispersed for $k = O\left(wTn^2b^2 \log(C) + \sqrt{T \log(n/\delta)}\right)$.*

Online Learning Summary



- Exploit dispersion to get no-regret.
  - For clustering & subset selection via greedy, assume input instances smoothed to show dispersion.
  - For partitioning pbs via IQPs, SDP + Rounding, exploit smoothness in the algorithms themselves.
- Can bound Rademacher complexity as a function of dispersion.
- Also bandit, and semi-bandit feedback.
- Also differential privacy bounds.
- Learning theory: techniques of independent interest beyond algorithm selection.