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.

Large family $F$ of algorithms

Sample of i.i.d. typical inputs

Facility location:

Clustering:

MST + Dynamic Programming
Greedy + Farthest Location
...
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?

\[ m = O(\text{dim}(F)/\epsilon) \] 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 algs for subset selection pbs (knapsack & independent set).

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

**Clustering: Parametrized Linkage**

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

```
DATA
\begin{align*}
\text{Single linkage} & \quad \text{Complete linkage} \\
& \quad \text{a - Weighted comb} \\
& \quad \ldots \\
& \quad \text{Ward's sig} \\
\text{DP for k-means} & \quad \text{DP for k-median} \\
& \quad \text{DP for k-center} \\
\end{align*}
```

\[ \dim(F) = O(\log n) \]

**Parametrized Lloyds**

[Balcan-Dick-White, NeurIPS 2018]

```
DATA
\begin{align*}
\text{Ran. seeds} & \quad \text{Farthest first} \\
& \quad \text{Local search} \\
\end{align*}
```

\[ \dim(F) = O(k \log n) \]

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

[Balcan-DeBlasio-Dick-Kingsford-Sandholm-Vitercik, 2019]
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, Max-2SAT, Correlation Clustering

  \[ \text{dim}(F) = O(\log n) \]

- **MIPs:** Branch and Bound Techniques

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

  \[
  \begin{align*}
  \text{Max } & \ c \cdot x \\
  \text{s.t. } & \ Ax = b \\
  & \ x_i \in \{0,1\}, \forall i \in I
  \end{align*}
  \]

- **Automated mechanism design for revenue maximization**

  Parametrized VCG auctions, posted prices, lotteries.

  [Balcan-Sandholm-Vitercik, EC 2018]
**Our results:** General sample complex tools via structure of dual class

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

**Thm:** Suppose for each \( \text{cost}_1(\alpha) \) 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 \( \text{cost}_1(\alpha) = g(\alpha) \).

\[
P\dim(\{\text{cost}_\alpha(I)\}) = O((d_{F^*} + d_{G^*}) \log(d_{F^*} + d_{G^*}) + d_{F^*} \log N)
\]

\( d_{F^*} = \text{VCdim of dual of } F, \ d_{G^*} = \text{Pdim of dual of } G. \)
Online Algorithm Selection

• So far, batch setting: collection of typical instances given upfront.


• **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 them 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 them 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:** $\min \sum_p \min_i d(p, c_i)$.

- **$k$-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.

\[ d(x,y) - \text{distance between } x \text{ and } y \]

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: \[ \text{dist}(A, B) = \frac{\text{avg}_{x \in A, x' \in B} \text{dist}(x, x')}{\text{# pairs}} \]

• Parametrized family, \( \alpha \)-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 pruning.

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

\( \alpha \in \mathbb{R} \)

**Key idea:**

- For a given \( \alpha \), which will merge first, \( N_1 \) and \( N_2 \), or \( N_3 \) and \( N_4 \)?

- 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


- **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 $t \in \{1, ..., T\}$:

1. Online learning algo chooses a parameter $\rho_t$
2. Adversary selects a piecewise Lipschitz function $u_t: \mathcal{C} \to [0, H]$  
   - corresponds to some pb instance and its induced scoring fnc
3. Get feedback: Full information: observe the function $u_t(\cdot)$ 
   Bandit feedback: observe only payoff $u_t(\rho_t)$.

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

\[\uparrow\] Performance of best parameter in hindsight
\[\uparrow\] Our cumulative performance
Online Regret Guarantees

Existing techniques (for finite, linear, or convex case): select $\rho_t$ probabilistically based on performance so far.

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

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

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 $\exists$ perfect $\rho$.

To achieve low regret, need structural condition.
Dispersion, Sufficient Condition for No-Regret

\{u_1(\cdot), \ldots, u_T(\cdot)\} \text{ is } (w, k)\text{-dispersed if any ball of radius } w \text{ contains boundaries for at most } k \text{ 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 $\rho_t$ from distr. $p_t$: $p_t(\rho) \propto \exp \left( \lambda \sum_{s=1}^{t-1} u_s(\rho) \right)$

Our Results:

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

Density of $\rho$ exponential in its performance so far.
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 $\rho_t$ from distr. $p_t$: $p_t(\rho) = \frac{\exp(\lambda U_t(\rho))}{W_t}$

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

$$W_t = \int \exp(\lambda U_t(\rho)) \, d\rho$$

Our Results:

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

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

For most problems:

- Set $w \approx 1/\sqrt{T}$, $k = \sqrt{T} \times (\text{fnc of problem})$
Dispersion, Sufficient Condition for No-Regret

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

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

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

The expected regret is $O\left(H \left( \sqrt{T d \log \frac{R}{w}} + k \right) + TLw \right)$.

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

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

$$\frac{W_{t+1}}{W_t} \leq \exp\left(\frac{(e^{H\lambda} - 1)P_t}{H}\right) \quad P_t = \text{expected alg payoff at time } t.$$  

$$\frac{W_{T+1}}{W_1} \leq \exp\left(\frac{(e^{H\lambda} - 1)P_{ALG}}{H}\right) \quad P_{ALG} = \text{expected alg total payoff.}$$
Dispersion, Sufficient Condition for No-Regret

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

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

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

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

**Key idea:** analyze \( W_t = \int \exp(\lambda U_t(\rho)) \, d\rho \) (norm. fac tor)

- **Lower bound** (uses dispersion + Lipschitz):

  Let \( \rho^* \) be the optimal parameter vector in hindsight.

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

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

  \( \leq k \) fn cs \( u_s \) have disp. in \( B(\rho^*, w) \) & range is \([0, H]\)
Dispersion, Sufficient Condition for No-Regret

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

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

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

The expected regret is $O\left( H \left( \sqrt{T d \log \frac{R}{w}} + k \right) + TLw \right)$.

**Key idea:** analyze $W_t = \int \exp(\lambda U_t(\rho)) d\rho$ (norm. factor)

- **Lower bound**
  
  For all $\rho \in B(\rho^*, w)$, $U_{T+1}(\rho) \geq U_{T+1}(\rho^*) - Hk - TLw$.
  
  $$W_{T+1} \geq \text{Vol}(B(\rho^*, w)) \exp(\lambda(U_{T+1}(\rho^*) - Hk - TLw))$$

  Get
  
  $$\frac{W_{T+1}}{W_1} \geq \left( \frac{w}{R} \right)^d \exp(\lambda(U_{T+1}(\rho^*) - Hk - TLw))$$
Dispersion, Sufficient Condition for No-Regret

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

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

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

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

**Key idea:** analyze $W_t = \int \exp(\lambda U_t(\rho)) d\rho$ (norm. factor)

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

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

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

Set $\lambda = \frac{1}{H} \sqrt{\frac{d}{T} \log \left( \frac{R}{w} \right)}$  

Regret $O\left( H \left( \sqrt{Td \log \frac{R}{w}} + k \right) + TLw \right)$
Example: Clustering with $\rho$-weighted linkage

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

**Theorem:** If $T$ instances with distances selected in $[0, B]$ from $\kappa$-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 $O(H\sqrt{T\log(Tn\kappa B)})$
Example: Clustering with $\rho$-weighted linkage

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

**Theorem:** If $T$ instances with distances in $[0, B]$ from $\kappa$-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 $\frac{d_1 - d_3}{(d_1 - d_3) + (d_4 - d_2)}$.

  Given $\rho$, 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 $O((\kappa B)^2)$-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 $\geq 1 - \delta$:

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

**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 $O(\log N)$

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

**Proof idea**:

How many ways can $\mathcal{F}$ label $d$ instances ($d$ sets of $\leq N$ pts) $s_1, \ldots, s_d$?

- $d$ instances have at most $Nd$ points in their union.
- For $f_I, f_{I'}$ 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 $(Nd)^2 + 1$ different subsets of them that can be covered using intervals.

To shatter $d$ instances, must have $2^d \leq (Nd)^2 + 1$. $VC$-dim $d = O(\log N)$. 
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 $v_i/s_i^\rho$ 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 $k = O\left(\frac{wTn^2b^2 \log(C)}{\sqrt{T \log(n/\delta)}}\right)$. 
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.