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 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 to 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 $\{c_1, c_2, ..., c_k\}$
  - 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 $\mathcal{F}$, sample of typical instances from domain (unknown distr. $\mathcal{D}$), find algo that performs well on new instances from $\mathcal{D}$.

**Large family $\mathcal{F}$ of algorithms**

**Sample of typical inputs**

- Clustering:
  - Input 1:
  - Input 2:
  - Input N:

- Facility location:
  - Input 1:
  - Input 2:
  - Input N:
Sample Complexity of Algorithm Selection

**Goal:** Given a family of algorithms $F$, a sample of typical instances from domain (unknown distribution $D$), find an algorithm that performs well on new instances from $D$.

**Approach:** Find $\hat{A}$, 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 $F$, 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 tools from learning theory**

- **Uniform convergence:** for any algo in $F$, average performance over samples “close” to its expected performance.
  - Imply that $\hat{A}$ has high expected performance.
  - $N = O(\text{dim}(F) / \epsilon^2)$ instances suffice for $\epsilon$-close.
**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 $D$.

**Key tools from learning theory**

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

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

![Diagram of overfitting](image-url)
Sample Complexity of Algorithm Selection

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

**Key tools from learning theory**

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

**Challenge:** Analyze $\text{dim}(\mathbf{F})$, due to combinatorial & modular nature, “nearby” programs/algos can have drastically different behavior.

**Challenge:** Design a computationally efficient meta-algorithm.
Formal Guarantees for Algorithm Selection


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.
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, NeurIPS’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

  \[ \text{Feasible solution to IQP} \]

  \[ \text{Integer Quadratic Programming (IQP)} \]

  \[ \text{Semidefinite Programming Relaxation (SDP)} \]

  \[ \text{GW rounding} \]

  \[ \text{1-linear rounding} \]

  \[ \text{s-linear rounding} \]

• **Automated mechanism design**

  [Balcan-Sandholm-Vitercik, EC 2018]

  **Generalized parametrized VCG auctions, posted prices, lotteries.**
Our results: New algo classes applicable for a wide range of pbs.

- Branch and Bound Techniques for solving MIPs

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

Formal Guarantees for Algorithm Selection

[Balcan-Dick-Sandholm-Vitercik, ICML’18]
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 $\{c_1, c_2, ..., c_k\}$

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

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

$k$-center: 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:** 
  \[ \text{dist}(A, B) = \text{avg}_{x \in A, x' \in B} \text{dist}(x, x') \]

- **Parametrized family, }\alpha\text{-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 algs 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: $\alpha$-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.
Claim: Pseudo-dimension of $\alpha$-weighted linkage + DP is $O(\log n)$, so small sample complexity.

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

**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, \( \mathcal{N}_1 \) 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.
Claim: Pseudo-dimension of $\alpha$-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 $2^m \leq m \cdot n^8$. Pseudo-dimension is $O(\log n)$. 

$\alpha \in \mathbb{R}$

Diagram showing clustering instances and patterns.
Clustering: Linkage + Dynamic Programming

Claim: Pseudo-dimension of $\alpha$-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 $\alpha$ over the sample is $\epsilon$-close to optimal over the distribution.

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

Algorithm
- Solve for all $\alpha$ intervals over the sample
  $\alpha \in \mathbb{R}$
- Find the $\alpha$ interval with the smallest empirical cost
Claim: Pseudo-dimension of $\alpha$-weighted linkage + DP is $O(\log n)$, so small sample complexity.

High level learning theory bit

- Want to prove that for all algorithm parameters $\alpha$:
  \[
  \frac{1}{|S|} \sum_{I \in S} \text{cost}_\alpha(I) \text{ close to } \mathbb{E}[\text{cost}_\alpha(I)].
  \]

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

$\text{cost}_1(\alpha) = \text{cost}_\alpha(I)$

$\alpha \in \mathbb{R}$
Partitioning Problems via IQPs

IQP formulation

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

Many of these pbs are NP-hard.

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

\textbf{Input:} Weighted graph \( G, w \)

\textbf{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

\( \text{var } v_i \) for node \( i \), either +1 or -1
Partitioning Problems via IQPs

**IQP formulation**

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

**Algorithmic Approach: SDP + Rounding**

1. Semi-definite programming (SDP) relaxation:
   - Associate each binary variable \(x_i\) with a vector \(u_i\).
   \[
   \text{Max } \sum_{i,j} a_{i,j} \langle u_i, u_j \rangle \\
   \text{subject to } \|u_i\| = 1
   \]

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 \(u_i\) falls on.
**Parametrized family of rounding procedures**

**IQP formulation**

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

**Algorithmic Approach: SDP + Rounding**

1. **SDP relaxation:**
   Associate each binary variable \( x_i \) with a vector \( u_i \).
   \[
   \text{Max } \sum_{i,j} a_{i,j} \langle u_i, u_j \rangle \\
   \text{subject to } ||u_i|| = 1
   \]

2. **s-Linear Rounding**
   \[\text{[Feige&Landberg'06]}\]
   - Inside margin, randomly round
   - Outside margin, round to -1.

---

**Diagram:**

- Integer Quadratic Programming (IQP)
- Semidefinite Programming Relaxation (SDP)
- GW rounding
- 1-linear rounding
- s-linear rounding
- Feasible solution to IQP
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 $\alpha$ 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

- **Key insight**: dual function sufficiently structured.

- For a fixed set of bids, revenue is **piecewise linear fnc** of parameters.

- Analyze pseudo-dim of \( \{ \text{revenue}_M : M \in \mathcal{M} \} \) 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.

---

**2nd-price auction with reserve**

```
Revenue
```

**Posted price mechanisms**

```
Revenue
```

```
Price($)  
2nd highest bid
```

```
Price($)  
2nd highest bid
```

```
Reserve $r$
```

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

1. **Online learning algo** chooses a parameter $\rho_t$

2. **Adversary** selects a piecewise Lipschitz function $u_t: C \rightarrow [0, H]$
   - 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 $u_t(\rho_t)$.

**Goal**: minimize regret:

$$\max_{\rho \in C} \sum_{t=1}^{T} u_t(\rho) - \mathbb{E}\left[\sum_{t=1}^{T} u_t(\rho_t)\right]$$

\[\uparrow\]

Performance of best parameter in hindsight \hspace{2cm} 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: $\max_{\rho \in C} \sum_{t=1}^{T} u_t(\rho) - \mathbb{E}[\sum_{t=1}^{T} u_t(\rho_t)] = \tilde{O}(\sqrt{T} \times \ldots )$

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)\} is \((w,k)\)-dispersed if any ball of radius \(w\) contains boundaries for at most \(k\) 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 \( \rho_t \) 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 \( \tilde{O}(\sqrt{Td \text{ fnc of problem}}) \).
Dispersion, Sufficient Condition for No-Regret

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

On each round $t \in \{1, ..., T\}$:
- Sample a vector $\rho_t$ from distr. $p_t$: $p_t(\rho) \propto \exp\left(\lambda \sum_{s=1}^{t-1} u_s(\rho)\right)$

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

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

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

For most problems:
- Set $w \approx 1/\sqrt{T}$, $k = \sqrt{T} \times (\text{fnc of problem})$
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 Lloyd’s 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)