Data Driven Algorithm Design

Maria-Florina (Nina) Balcan
Carnegie Mellon University
Analysis and Design of Algorithms

Classic algo design: solve a worst case instance.

• Easy domains, have optimal poly time algos.
  E.g., sorting, shortest paths

• Most domains are hard.
  E.g., clustering, partitioning, subset selection, auction design, ...

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

• Suited when repeatedly solve instances of the same algo problem.
Data Driven Algorithm Design

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

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

Prior work: largely empirical.

- Artificial Intelligence:
  [Horvitz-Ruan-Gomes-Kautz-Selman-Chickering, UAI 2001]
  [Xu-Hutter-Hoos-LeytonBrown, JAIR 2008]
- Computational Biology: E.g., [DeBlasio-Kececioglu, 2018]
- Game Theory: E.g., [Likhodedov and Sandholm, 2004]
Data Driven Algorithm Design

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

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

Prior work: largely empirical.

Our Work: Data driven algos with formal guarantees.

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

Related to: Hyperparameter tuning, AutoML, MetaLearning.

Program Synthesis (Sumit Gulwani’s talk on Mon).
Structure of the Talk

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

**Clustering**: Given a set objects organize 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 of objects organize them 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 Design as Distributional Learning

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

**Large family $F$ of algorithms**

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

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

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

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

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

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

**Key tools from learning theory**

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

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

**Key tools from learning theory**

\[ N = O\left( \frac{\dim(\mathcal{F})}{\epsilon^2} \right) \] instances suffice for $\epsilon$-close.

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

More complex patterns can fit, more samples needed for UC and generalization
Sample Complexity of Algorithm Selection

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

**Key tools from learning theory**

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

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

---

Overfitting

$y$

$\mathcal{F}$

Training set
Statistical Learning Approach to AAD

**Challenge:** “nearby” algos can have drastically different behavior.

**Challenge:** design a computationally efficient meta-algorithm.
Algorithm Design as Distributional Learning


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

Clustering: Parametrized Linkage
[Balcan-Nagarajan-Vitercik-White, COLT 2017]
[Balcan-Dick-Lang, 2019]

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

Parametrized Lloyd’s
[Balcan-Dick-White, NeurIPS 2018]

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

Alignment pbs (e.g., string alignment): parametrized dynamic prog.
[Balcan-DeBlasio-Dick-Kingsford-Sandholm-Vitercik, 2019]
Algorithm Design as Distributional Learning

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

- **Partitioning pbs via IQPs: SDP + Rounding**
  
  [Balcan-Nagarajan-Vitercik-White, COLT 2017]

  E.g., Max-Cut,
  Max-2SAT, Correlation Clustering

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

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

Max $c \cdot x$

s.t. $A x = b$

$x_i \in \{0,1\}, \forall i \in I$

```
MIP instance

Choose a leaf of the search tree

Best-bound  Depth-first

Choose a variable to branch on

Product  Most fractional  $\alpha$-linear

Fathom if possible and terminate if possible
```
Clustering Problems

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

Objective based clustering

$k$-means

Input: Set of objects $S, d$

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

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

Or minimize distance to ground-truth
Clustering: Linkage + Post-processing

Family of poly time 2-stage algorithms:

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

1. **Greedy linkage-based algo to get hierarchy (tree) of clusters.**

2. **Fixed algo (e.g., DP or last k-merges) to select a good pruning.**
Clustering: Linkage + Post-processing

1. Linkage-based algo to get a hierarchy.

2. Post-processing to identify a good pruning.

Both steps can be done efficiently.
Linkage Procedures for Hierarchical Clustering

**Bottom-Up (agglomerative)**

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

Different defs of “closest” give different algorithms.
Linkage Procedures for Hierarchical Clustering

Have a distance measure on pairs of objects.

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

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

• Single linkage: \[ \text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x') \]

• Complete linkage: \[ \text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x') \]

• Parametrized family, \( \alpha \)-weighted linkage:

\[
\text{dist}_\alpha(A, B) = (1 - \alpha) \min_{x \in A, x' \in B} d(x, x') + \alpha \max_{x \in A, x' \in B} d(x, x')
\]
**Clustering: Linkage + Post Processing**

**Our Results:**  \( \alpha \)-weighted linkage + Post-processing

- Pseudo-dimension is \( O(\log n) \), so small sample complexity.
- Given sample \( S \), find best algo from this family in poly time.

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

Problem: a single change to an early decision by the linkage algo, can snowball and produce large changes later on.
Claim: Pseudo-dim of $\alpha$-weighted linkage + Post-process is $O(\log n)$.

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-dim of $\alpha$-weighted linkage + Post-process is $O(\log n)$.

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.

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 $\alpha d(p, q) + (1 - \alpha)d(p', q')$ or $\alpha d(r, s) + (1 - \alpha)d(r', s')$ is smaller.
- An interval boundary an equality for 8 points, so $O(n^8)$ interval boundaries.
Claim: Pseudo-dim of \( \alpha \)-weighted linkage + Post-process is \( O(\log n) \).

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

- Pseudo-dim largest \( m \) for which \( 2^m \) patterns achievable.

- So, solve for \( 2^m \leq m n^8 \). Pseudo-dimension is \( O(\log n) \).
Claim: Given sample $S$, can find best algo from this family in poly time.

- Solve for all $\alpha$ intervals over the sample.
  $\alpha \in [0,1]$
- Find $\alpha$ interval with smallest empirical cost.

Clustering: Linkage + Post Processing

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

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.
Learning Both Distance and Linkage Criteria

[Balcan-Dick-Lang, 2019]

- Often different types of distance metrics.
  - Captioned images, $d_0$ image info, $d_1$ caption info.
  - Handwritten images: $d_0$ pixel info (CNN embeddings), $d_1$ stroke info.

Family of Metrics: Given $d_0$ and $d_1$, define

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

Parametrized $(\alpha, \beta)$-weighted linkage (\(\alpha\) interpolation between single and complete linkage and \(\beta\) interpolation between two metrics):

$$\text{dist}_\alpha(A, B; d_\beta) = (1 - \alpha) \min_{x \in A, x' \in B} d_\beta(x, x') + \alpha \max_{x \in A, x' \in B} d_\beta(x, x')$$
Learning Both Distance and Linkage Criteria

Claim: Pseudo-dim. of \((\alpha, \beta)\)-weighted linkage is \(O(\log n)\).

Key fact: Fix instance of \(n\) pts; vary \(\alpha, \beta\), partition space with \(O(n^8)\) linear, quadratic equations s.t. within each region, same cluster tree.
Learning Distance for Clustering Subsets of Omniglot

- Written characters from 50 alphabets, each character 20 examples. [Lake, Salakhutdinov, Tenenbaum ’15]

- Image & stroke (trajectory of pen)

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

- $d_0$ uses character images.
  - Cosine distance between CNN feature embeddings
  - CNN trained on MNIST.

- $d_1$ Hand-designed Stroke.
  - Average distance from points on each stroke to nearest point on other stroke.
Clustering Subsets of Omniglot

\[ \beta^* = 0.514 \]
\[ \text{Error} = 33.0\% \]
\[ \text{Improvement of 9.1\%} \]

\[ \beta = 1 \]
\[ \text{Error} = 42.1\% \]
Partitioning Problems via IQPs

**Input:** Weighted graph $G, w$

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

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

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.

**IQP formulation**

$$\text{Max } x^T A x = \sum_{i,j} a_{i,j} x_i x_j$$

s.t. $x \in \{-1, 1\}^n$
Parametrized family of rounding procedures

**IQP formulation**

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

**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**
   [Feige&Landberg'06]
   Inside margin, randomly round
   Outside margin, round to -1.

---

[Diagram of the algorithmic approach with a flowchart illustrating the connections between SDP relaxation, rounding procedures, and the 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.
Data-driven Mechanism Design

- **Mechanism design** for revenue maximization.
  
  [Balcan-Sandholm-Vitercik, EC'18]

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

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

2nd-price auction with reserve

Posted price mechanisms
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 $\{\text{cost}_I: \text{instances } I\}$.

\[
\text{cost}_I(\alpha) = \text{cost}_\alpha(I)
\]

$\alpha \in \mathbb{R}$
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 $t \in \{1, ..., T\}$:

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

\[\uparrow\]
Performance of best parameter in hindsight

\[\uparrow\]
Our cumulative performance
Dispersion, Sufficient Condition for No-Regret

Piecewise Lipschitz function

Lipschitz within each piece

Not disperse

Many boundaries within interval

Disperse

Few boundaries within any interval

\{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 \(u_i\).
Full info: exponentially weighted forecaster \cite{Cesa-Bianchi-Lugosi2006}

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 $\tilde{O}(\sqrt{Td \text{ fnc of problem}})$. 
Summary and Discussion

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

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

- Machine learning: techniques of independent interest beyond algorithm selection.
Many Exciting Open Directions

- Analyze other widely used classes of algorithmic paradigms.
  - Branch and Bound Techniques for MIPs [Balcan-Dick-Sandholm-Vitercik, ICML’18]
  - Parametrized Lloyd’s methods [Balcan-Dick-White, NeurIPS’18]
  - Other algorithmic paradigms relevant to data-mining pbs.

- Other learning models (e.g., one shot, domain adaptation, reinforcement learning).

- Explore connections to program synthesis; automated algo design.

- Connections to Hyperparameter tuning, AutoML, Meta-learning.

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