Sample Complexity for 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.

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.
  - General sample complexity theorem.
Example: Clustering Problems

**Clustering:** Given a set of objects organize them into natural groups.

- E.g., cluster news articles, or web pages, or search results by topic.
  ![Web pages](image1.png) ![Vogue](image2.png) ![Web page](image3.png) ![Image](image4.png)

- Or, cluster customers according to purchase history.
  ![Customers](image5.png) ![Customers](image6.png) ![Customers](image7.png)

- Or, cluster images by who is in them.
  ![Images](image8.png) ![Images](image9.png) ![Images](image10.png)

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 $\{c_1, c_2, \ldots, c_k\}$

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

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

**k-center/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 algorithms $\mathcal{F}$, sample of typical instances from domain (unknown distr. $D$), find algo that performs well on new instances from $D$.

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

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

**Key Question:** Will $\tilde{\mathbf{A}}$ 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. $\mathbf{D}$), find algo that performs well on new instances from $\mathbf{D}$.

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

**Key tools from learning theory**

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

**Challenge:** analyze $dim(F)$, 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, alignment, 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 2017] [Balcan-Dick-Lang, 2019]

- **Clustering: Greedy Seeding + Local Search**
  [Balcan-Dick-White, NeurIPS 2018]

  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

- Computational biology (e.g., string alignment, RNA folding): parametrized dynamic programming.
  [Balcan-DeBlasio-Dick-Kingsford-Sandholm-Vitercik, 2019]
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]

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

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

- **General techniques for sample complexity based on properties of the dual class of fns.**
  
  [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, 2019]

- **Automated mechanism design for revenue maximization**
  
  [Balcan-Sandholm-Vitercik, EC 2018]

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

- Online and private algorithm selection.

[Balcan-Dick-Vitercik, FOCS 2018]  [Balcan-Dick-Pedgen, 2019]

[Balcan-Dick-Sharma, 2019]
Clustering Problems

Clustering: Given a set objects (news articles, customer surveys, web pages, ...) organize them 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 \( \sum_p \min_i d^2(p, c_i) \)

Or minimize distance to ground-truth
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: \[ \text{dist}(A, B) = \frac{1}{|A|} \sum_{x \in A} \frac{1}{|B|} \sum_{x' \in B} \text{dist}(x, x') \]

• 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 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.

\[ \alpha \in \mathbb{R} \]

\[ \mathcal{N}_1 \]
\[ \mathcal{N}_2 \]
\[ \mathcal{N}_3 \]
\[ \mathcal{N}_4 \]

**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 $2^m \leq m n^8$. Pseudo-dimension is $O(\log n)$.
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
**Clustering: Linkage + Dynamic Programming**

**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** $\{\text{cost}_I: \text{instances } I\}$.

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

$\alpha \in \mathbb{R}$
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., **Max cut**: partition a graph into two pieces to maximize weight of edges crossing the partition.

**Input**: Weighted graph \( G, w \)

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

\( v_i \in \{-1,1\} \)

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

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

Algorithmic Approach: SDP + Rounding

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

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
\[
\begin{align*}
\text{Max } & \sum_{i,j} a_{i,j} x_i x_j \\
\text{s.t. } & x \in \{-1,1\}^n
\end{align*}
\]

Algorithmic Approach: SDP + Rounding

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

2. s-Linear Rounding
   [Feige&Landberg'06]

Inside margin, randomly round
Outside margin, round to -1.
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

- 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.

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

2nd-price auction with reserve

<table>
<thead>
<tr>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd highest bid</td>
</tr>
<tr>
<td>Highest bid</td>
</tr>
</tbody>
</table>

Posted price mechanisms

<table>
<thead>
<tr>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserve r</td>
</tr>
<tr>
<td>Price</td>
</tr>
</tbody>
</table>

2nd-price auction with reserve

<table>
<thead>
<tr>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd highest bid</td>
</tr>
<tr>
<td>Highest bid</td>
</tr>
</tbody>
</table>

Posted price mechanisms

<table>
<thead>
<tr>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
</tr>
<tr>
<td>Price</td>
</tr>
</tbody>
</table>
General Sample Complexity via Dual Classes

• 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 \( \{\text{cost}_I: \text{instances } I\} \).

Theorem: 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. \( \text{cost}_I(\alpha) = g(\alpha) \).

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

\( d_F^* = \text{VCdim of dual of } F \), \( d_G^* = \text{Pdim of dual of } G \).
Theorem: 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. $\text{cost}_1(\alpha) = g(\alpha)$.

$$\text{Pdim}(\{\text{cost}_1(I)\}) = O((d_F^* + d_G^*) \log(d_F^* + d_G^*) + d_F^* \log N)$$

$d_F^* = \text{VCdim of dual of F}$, $d_G^* = \text{Pdim of dual of G}$. 
General Sample Complexity via Dual Classes

Theorem: Suppose for each $\text{cost}_1(\alpha)$ there are $\leq N$ boundary functions $f_1, f_2, \ldots \in F$ such that within each region defined by them, $\exists g \in G$ s.t. $\text{cost}_1(\alpha) = g(\alpha)$.

$$\text{Pdim} \{\text{cost}_\alpha(I)\} = O((d_F^* + d_G^*) \log(d_F^* + d_G^*) + d_F^* \log N)$$

$d_F^* = \text{VCdim of dual of } F$, $d_G^* = \text{Pdim of dual of } G$.

$\text{VCdim}(F)$: fix $N$ pts. Bound # of labelings of these pts by $f \in F$ via Sauer’s lemma in terms of $\text{VCdim}(F)$.

$\text{VCdim}(F^*)$: fix $N$ fns, look at # regions. In the dual, a point labels a function, so direct correspondence between the shattering coefficient of the dual and the number of regions induced by these functions. Just use Sauer’s lemma in terms of $\text{VCdim}(F^*)$. 

\[
\begin{align*}
&f_1 \
&\quad \text{cost}_1 = g_1 \
&\quad \text{cost}_1 = g_2 \
&f_2 \
&\quad \text{cost}_1 = g_3 \
&\quad \text{cost}_1 = g_4 \
&\quad \text{cost}_1 = g_5 \
&f_3 \
&\quad \text{cost}_1 = g_6 \
&\quad \text{cost}_1 = g_7
\end{align*}
\]
General Sample Complexity via Dual Classes

Theorem: 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.

$$\text{cost}_1(\alpha) = g(\alpha).$$

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

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

Proof:

- Fix $D$ instances $I_1, \ldots, I_D$ and $D$ thresholds $z_1, \ldots, z_D$. Bound # sign patterns ($\text{cost}_\alpha(I_1), \ldots, \text{cost}_\alpha(I_D)$) ranging over all $\alpha$. Equivalently, $(\text{cost}_{I_1}(\alpha), \ldots, \text{cost}_{I_D}(\alpha))$.

- Use $\text{VCdim of } F^*$, bound # of regions induced by $\text{cost}_{I_1}(\alpha), \ldots, \text{cost}_{I_D}(\alpha) : (\text{eND})^{d_{F^*}}$.

- On a region, exist $g_{I_1}, \ldots, g_{I_D}$ s.t. $(\text{cost}_{I_1}(\alpha), \ldots, \text{cost}_{I_D}(\alpha)) = (g_{I_1}(\alpha), \ldots, g_{I_D}(\alpha))$, which equals $(\alpha(g_{I_1}), \ldots, \alpha(g_{I_D}))$. These are fns in dual class of $G$. Sauer’s lemma on $G^*$, bounds # of sign patterns in that region by $(\text{eD})^{d_{G^*}}$.

- Combining, total of $(\text{eND})^{d_{F^*}}(\text{eD})^{d_{G^*}}$. Set to $2^D$ and solve.
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.**

• **Learning theory: techniques of independent interest beyond algorithm selection.**
Discussion, Open Problems

• Analyze other widely used classes of algorithmic paradigms.

• Other learning models (e.g., one shot, domain adaptation, RL).

• Explore connections to program synthesis; automated algo design.

• Explore connections to Hyperparameter tuning, AutoML, MetaLearning.

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