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 (for distributional & online learning): 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.
- 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 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.
Clustering Problems

**Clustering**: Given a set 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
Algorithm Design as Distributional Learning

**Goal:** given family of algs $\mathcal{F}$, 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 $N$:

- **MST**
- **Greedy**
- **Dynamic Programming**
- **Farthest Location**
...
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?
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 \( \hat{\mathbf{A}} \) 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 \( \hat{\mathbf{A}} \) 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 $\mathbf{F}$, 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**

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

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

Overfitting
**Challenge:** "nearby" 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 functions.
Formal Guarantees for Algorithm Selection

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

### Clustering: Parametrized Linkage

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

**[Balcan-Dick-Lang, 2019]**

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

### Parametrized Lloyds

**[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]**
Formal Guarantees for Algorithm Selection

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]
Formal Guarantees for Algorithm Selection

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

• **General sample complexity theorem via structure of dual fns.**
  
  \[ \text{Pdim}(\{\text{cost}_\alpha(I)\}) = O((d_{F^*} + d_{G^*}) \log(d_{F^*} + d_{G^*}) + d_{F^*} \log N) \]
  
  [Balcan-DeBlasio-Dick-Kingsford-Sandholm-Vitercik, 2019]

• **Data driven algorithm design via online learning.**

  [Balcan-Dick-Vitercik, FOCS 2018] [Balcan-Dick-Pedgen, 2019] [Balcan-Dick-Sharma, 2019]
Clustering: Linkage + Post-processing

Family of poly time 2-stage algorithms:

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.
\[ d(x, y) - \text{distance between } x \text{ and } y \]
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:** \[ \text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x') \]
- **Parametrized family, } \alpha\text{-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

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

- Pseudo-dimension is \(O(\log n)\), so small sample complex.
- 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, \( N_1 \) and \( N_2 \), or \( N_3 \) and \( N_4 \)?
- 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: 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.

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.
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):

$$\text{dist}_\alpha(A, B; d_\beta) = (1 - \alpha) \min_{x \in A, x' \in B} d_\beta(x, x') + \alpha \max_{x \in A, x' \in B} d_\beta(x, x')$$
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.
Clustering Subsets of Omniglot

\[ \beta = 1 \]
Error = 42.1%

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

Stroke Distance $\beta$ MNIST Features

Hamming Cost

0.0 0.2 0.4 0.6 0.8 1.0
0.34 0.36 0.38 0.40 0.42 0.44 0.46 0.48
Partitioning Problems via IQPs

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

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: Max \(\sum_{(i,j) \in E} w_{ij} \left(1 - v_i v_j \right) / 2\) \\
\text{s.t. } v_i \in \{-1,1\} \\
var v_i \text{ for node } i, \text{ either } +1 \text{ or } -1 \\
1 \text{ if } v_i, v_j \text{ opposite sign, } 0 \text{ if same sign}
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** for sample complex of data-driven 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.

![Graphs showing revenue models](image)
General Sample Complexity via Dual Classes

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

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\}.

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

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

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

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

\[\text{IQP objective value}\]

\[\text{Revenue}\]

\[\text{2nd highest bid}\]

\[\text{Highest bid}\]

\[\text{Revenue}\]

\[\text{Price}\]

\[\text{Price}\]
General Sample Complexity via Dual Classes

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

\[
\text{cost}_i(\alpha) = g(\alpha).
\]

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

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

\( \text{VCdim}(F) \): 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 fns. Just use Sauer’s lemma in terms of \( \text{VCdim}(F^*) \).
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. $\text{cost}_i(\alpha) = g(\alpha)$.

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

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

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 $\text{cost}_{I_1}(\alpha), \ldots, \text{cost}_{I_D}(\alpha) : (eND)^{d_F^*}$.

- 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 $\left(\alpha(g_{I_1}), \ldots, \alpha(g_{I_D})\right)$. These are fns in dual class of $G$. Sauer's lemma on $G^*$, bounds $\#$ of sign patterns in that region by $(eD)^{d_G^*}$.

- Combining, total of $(eND)^{d_F^*}(eD)^{d_G^*}$. Set to $2^D$ and solve.
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 $t \in \{1, \ldots, T\}$:

1. **Online learning algo** chooses a parameter $\rho_t$
2. **Adversary selects** a piecewise Lipschitz function $u_t: C \to [0, H]$
   - 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:**

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

↑ \hspace{1cm} ↑

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

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

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

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

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 $w \approx 1/\sqrt{T}$, $k = \sqrt{T} \times (\text{fnc of problem})$
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 $k = O(wn^8\kappa^2B^2T) + O(\sqrt{T\log(n/\delta)})$.

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

From a uniform convergence argument

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

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

• Analyze other widely used classes of algorithmic paradigms.
  • Branch and Bound Techniques for MIPs [Balcan-Dick-Sandholm-Vitercik, ICML’18]
  • Parametrized Lloyds methods [Balcan-Dick-White, NIPS’18]

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

• Explore connections to program synthesis; automated algo design.

• Connections to Hyperparameter tuning, AutoML, MetaLearning.
  Use our insights for pbs studied in these settings (e.g., tuning hyper-parameters in deep nets)