Foundations of Data Driven Algorithm Design

Maria-Florina (Nina) Balcan
Carnegie Mellon University
Data Driven Algorithm Selection

Some domains we have polynomial time optimal algorithms:

- E.g., sorting, searching, shortest paths...

Some domains we don't:

- Different methods work better in different settings.
- Large family of methods - what’s best in our application?
- E.g., data clustering, partitioning problems, auction design, ...

Use ML to automate algo design in difficult domains.
Data Driven Algorithm Selection

Use ML to automate algo design in difficult domains.

- Large body of empirical work.
  - AI community: E.g., [Xu-Hutter-Hoos-LeytonBrown, JAIR 2008]
  - Computational Biology: E.g., [DeBlasio-Kececioglu, 2018]
  - Game Theory: E.g., [Likhodedov and Sandholm, 2004]

- This talk: formal guarantees for this approach.
Algorithm Selection as a Learning Problem

**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 $\mathcal{F}$ of algorithms

Sample of typical inputs

Facility location:

Clustering:
Sample Complexity of Algorithm Selection

**Goal:** given large family of algos, sample of typical instances from domain, find an algo that performs well on new instances from same domain.

**Approach:** ERM, find the algo that performs best over our sample.

**Key Question:** When do we generalize?

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

**Goal**: widely applicable techniques for analyzing the intrinsic complexity of families of algos and ensuring good generalizability.

Also design an efficient meta-algorithm.

**Natural Idea**: apply tools from learning theory.

\[ m = O(\dim(F)/\epsilon^2) \] instances suffice to ensure generalizability

**Challenge**: analyze \( \dim(F) \), due to combinatorial & modular nature, “nearby” programs/algos can have drastically different behavior.

Classic machine learning

Our work
Formal Guarantees for Algorithm Selection

Prior Work:

[Gupta-Roughgarden, ITCS 2016 & SICOMP 2017]: proposed learning theoretic model for analyzing algorithm selection; analyzed greedy procedures for subset selection problems (knapsack & independent set).
Formal Guarantees for Algorithm Selection

- **Our Work**: Distributional settings, new algo classes applicable for a wide range of problems.

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

- **Clustering**: Linkage + Dynamic Programming

- **Partitioning pbs via IQPs**: SDP + Rounding
  
  E.g., Max-Cut, Max-2SAT, Correlation Clustering
Formal Guarantees for Algorithm Selection

- **Our Work**: Distributional settings, new algo classes applicable for a wide range of problems.

[Balcan-Dick-Sandholm-Vitercik, ICML 2018]

- **Branch and Bound Techniques for solving MIPs**

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

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
Formal Guarantees for Algorithm Selection

- **Our Work**: Distributional settings, new algo classes applicable for a wide range of problems.
  - [Balcan-Nagarajan-Vitercik-White, COLT 2017]
  - [Balcan-Dick-Sandholm-Vitercik, ICML 2018]

- **Related Work**: guarantees for automated mechanism design in distributional settings.
  - [Balcan-Sandholm-Vitercik, EC 2018]
  - [Balcan-Sandholm-Vitercik, Tutorial ICML 2018]

- **Recent Work**: General results for private and online algorithm selection.
  - [Balcan-Dick-Vitercik, FOCS 2018]
**Clustering**

**Problem**: Given a set of $n$ objects (news articles, customer surveys, web pages, ...), organize into natural groups.

- **E.g., objective based clustering**
  - $k$-median: find centers $\{c_1, c_2, ..., c_k\}$ to minimize $\sum_p \min d(p, c_i)$
  - $k$-means: find centers $\{c_1, c_2, ..., c_k\}$ to minimize $\sum_p \min d^2(p, c_i)$
  - $k$-center: find centers to 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:** \[ \text{dist}(A, B) = \text{avg}_{x \in A, x' \in B'} \text{dist}(x, x') \]

- **\(\alpha\)-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.
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 algorithm can snowball and produce large changes later on.
**Clustering: Linkage + Dynamic Programming**

**Our Results:** $\alpha$-weighted linkage+DP

- Pseudo-dimension is $O(\log n)$, so small sample complexity.

**Key idea:**

- Break real line into a small number of intervals s.t. **on each instance:**

  $\alpha \in \mathbb{R}$

  - Two $\alpha$’s from one interval result in the same tree.
  - And therefore the same clustering.
  - And therefore the same performance cost.
**Clustering: Linkage + Dynamic Programming**

**Our Results:** $\alpha$-weighted linkage+DP

Pseudo-dimension is $O(\log n)$, so small sample complexity.

**Key idea:**
- Break real line into intervals s.t. on each instance same performance.
  $\alpha \in \mathbb{R}$

- For a clustering instance of $n$ points, $O(n^8)$ intervals.
  - Over any $\alpha$ interval, so long as order in which all pairs of nodes are merged is fixed, then resulting tree is invariant.
  - 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.

- Any interval boundary must be an equality for some such set of 8 points, so $O(n^8)$ interval boundaries. Order of merges is fixed between any two adjacent interval boundaries.
Clustering: Linkage + Dynamic Programming

Our Results: $\alpha$-weighted linkage + DP

Pseudo-dimension is $O(\log n)$, so small sample complexity.

Key idea:
- Break real line into intervals s.t. on each instance same performance.
- For $m$ clustering instances of $n$ points, $O(mn^8)$ intervals.

So, pseudo-dim is $O(\log n)$. 
Clustering: Linkage + Dynamic Programming

Our Results: \( \alpha \)-weighted linkage+DP

- Pseudo-dimension is \( O(\log n) \).

For \( m = \tilde{O}(\log n/\epsilon^2) \), w.h.p. expected performance cost of best \( \alpha \) over the sample is \( \epsilon \)-close to optimal over the distribution.

- Given sample \( S \), can find best algo from this family in poly time.

Algorithm (high level)

- Solve for all \( \alpha \) intervals over the sample

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

- Find the \( \alpha \) interval with the smallest empirical cost
Partitioning Problems via IQPs

**IQP formulation**

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

E.g., max-cut

\[
\text{Max } \sum_{(i,j) \in E} w_{ij} \left( \frac{1-v_i v_j}{2} \right) \\
\text{s.t. } v_i \in \{-1,1\}
\]

Many of these problems are NP-hard.
Partitioning Problems via IQPs

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

Algorithmic Approach: SDP + Rounding

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

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

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]
   - Choose a random hyperplane.
   - Random thresholding
     Set \( x_i \) to 1 w.p. \( \frac{1}{2} + \frac{1}{2} \varphi_s(\langle u_i, Z \rangle) \) and -1 w.p. \( \frac{1}{2} - \frac{1}{2} \varphi_s(\langle u_i, Z \rangle) \)

\[
\varphi_s(x) = -1_{x < -s} + \frac{x}{s} 1_{x \in [-s,s]} + 1_{x > s}
\]
Parametrized family of rounding procedures

**IQP formulation**

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

**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]
   
   - Choose a random hyperplane.
   - Random thresholding
     
     Set \( x_i \) to 1 w.p \( \frac{1}{2} + \frac{1}{2} \varphi_s(\langle u_i, Z \rangle) \)
     
     and -1 w.p \( \frac{1}{2} - \frac{1}{2} \varphi_s(\langle u_i, Z \rangle) \)
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 the best parameter overall.
Online Algorithm Selection

- So far, batch setting: collection of typical instances given upfront.
- Scoring functions non-convex, with lots of discontinuities, cannot use known techniques. They are piecewise Lipschitz.
- Online optimization with Piecewise Lipschitz functions.
- Identify a general structural property called dispersion that allows us to get good regret bounds and show this property holds for many alg. selection problems.
Recent Work: Online Algorithm Selection

Recent Work: [Balcan-Dick-Vitercik, FOCS 2018]

Online optimization

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

1. The online learning algorithm chooses a parameter $\rho_t$
2. The adversary chooses a piecewise Lipschitz function $u_t: \mathcal{C} \rightarrow [0, H]$ (corresponds to some problem instance and its induced scoring function)
   Receive the score of the parameter we selected $u_t(\rho_t)$.
3. Full information: Algorithm observes the function $u_t(\cdot)$
4. Bandit feedback: Algorithm only receives payout $u_t(\rho_t)$.

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

\[\uparrow\]\[\uparrow\]
Performance of best parameter in hindsight Our cumulative performance
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 \(u_i\).
Full information: exponentially weighted forecaster

**Full information: exponentially weighted forecaster** [Cesa-Bianchi and Lugosi 2006]

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

- Sample a vector \( \rho_t \) from a distribution \( p_t \) where
  \[
  p_t(\rho) \propto \exp\left( \lambda \sum_{s=1}^{t-1} u_s(\rho) \right)
  \]

**Our Results:**

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

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

Usual \( \sqrt{T} \) bound, but lose a \( \log(1/w) \) multiplicative term, and an additive \( kH \) term [for the \( k \) discontinuities that might be inside a ball of radius \( w \) around the optimal solution] and an additive \( TLw \) for the Lipschitz constant.
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 $O\left( H\left( \sqrt{T d \log \frac{1}{w} + k} \right) + TLw \right)$.

For most problems:

- Set $w \approx 1/\sqrt{T}$

- Get $k = \sqrt{T} \times$ (some function of problem)

- Overall, get regret $\tilde{O}(H\sqrt{Td})$. 

Full information: exponentially weighted forecaster
Example: rounding of SDP relaxation of IQP

Idea:

- Exploit randomness of algorithm to give a guarantee on dispersion.
- Prove that whp, for any $\alpha \geq \frac{1}{2}$, the set of $u_i$ are $(T^{\alpha-1}, O(nT^{\alpha}\sqrt{\log n}))$-dispersed.
- Lipschitz value depends on which class of rounding schemes.
- Setting $\alpha = \frac{1}{2}$ leads to regret of $\tilde{O}(Hn\sqrt{T})$. 
Discussion

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

• Exploit structure to provide good sample complexity and regret bounds. Also privacy guarantees.

• From a learning theory point of view, techniques of independent interest beyond algorithm configuration.

• Related in spirit to Hyperparameter tuning, AutoML, MetaLearning.

• **Future Work**: use our insights to analyze problems commonly studied in these settings (e.g., tuning hyper-parameters in deep nets)