Foundations of Data Driven Combinatorial Algorithm Selection

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.

• Long history in the AI community.
  • E.g., [Xu-Hutter-Hoos-LeytonBrown, JAIR 2008]

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

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

**Large family $F$ of algorithms**

**Sample of typical inputs**

- Facility location:
  - Input 1:
  - Input 2:
  - Input $m$:

- Clustering:
  - Input 1:
  - Input 2:
  - Input $m$:...
**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.

**Why is it important?**

- For some problems we don’t know **any** algos that do well in the worst case, but we hope some algo in our family will do well in our domain.

  - Facility location and clustering:

- For others, we do have good worst-case algorithms, but still, some do better than others on different domains.

  - Shortest paths: Adaptive algorithms have better performance!
**Sample Complexity of Algorithm Selection**

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

**Approach:** 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.

**Idea:** apply tools from learning theory to analyze intrinsic complexity/dimension of families of algos.

\[ m = O(\text{dim}(F)/\epsilon) \] instances suffice to ensure generalizability

- Use this to design an efficient meta-algorithm

**Challenge:** due to combinatorial and modular nature, “nearby” programs/algos can have drastically different behavior.
Sample Complexity of Algorithm Selection


- **Our Work:** [Balcan-Nagarajan-Vitercik-White, COLT 2017]
  - **Clustering:** Linkage + Dynamic Programming
    - **Partitioning pbs via IQPs:** SDP + Rounding
      - E.g., Max-Cut,
      - Max-2SAT, Correlation Clustering

- **Recent Work:** [Balcan-Dick-Vitercik, 2017]
  - Private and online algorithm selection
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.

- Exhaustive search is too expensive, and efficient heuristics sometimes work and sometimes don't.
Clustering: Linkage + Dynamic Programming

Family of poly time 2-stage algorithms:

1. Use a 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.
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:**
  \[
  \text{dist}(A, B) = \frac{1}{\binom{|A|}{2} \cdot \binom{|B|}{2}} \sum_{x \in A, x' \in B} \text{dist}(x, x')
  \]

- **\( \alpha \)-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.
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, $\mathcal{N}_1$ 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.
  - 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 instance 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**

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

E.g., max-cut

\[
\text{Max } \sum_{(i,j) \in E} w_{ij} \left( \frac{1-v_i v_j}{2} \right) \\
\text{s.t. } v_i \in \{-1,1\}
\]
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. 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 \cite{Goemans95}
   - 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
\[
\begin{align*}
\text{Max } & x^T A x = \sum_{i,j} a_{i,j} x_i x_j \\
\text{s.t. } & x \in \{-1,1\}^n
\end{align*}
\]

Algorithmic Approach: SDP + Rounding

1. 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]
   - 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**

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

**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 \( \frac{1}{2} + \frac{1}{2} \psi_s(\langle u_i, Z \rangle) \)
     
     and -1 w.p \( \frac{1}{2} - \frac{1}{2} \psi_s(\langle u_i, Z \rangle) \)
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.
Recent Work: Online Private Alg. Selection

Recent Work: [Balcan, Dick, Vitercik’17]

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 $u_t: \mathcal{C} \rightarrow [0, H]$ (Equivalently, the adversary can choose a problem instance $x_t$ and sets $u_t(\rho_t) = u(x_t, \rho_t)$.)
3. Full information: Algorithm observes the function $u_t(\cdot)$ and receives payout $u_t(\rho_t)$.
   
   Bandit feedback: Algorithm only receives payout $u_t(\rho_t)$.

We want to minimize regret: $\max_{\rho \in \mathcal{C}} \sum_{t=1}^{T} u_t(\rho) - \mathbb{E}[\sum_{t=1}^{T} u_t(\rho_t)]$
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 [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 $O\left(H \left(\sqrt{T}d \log \frac{1}{w} + k\right) + TLw\right)$. 
Example: rounding of SDP relaxation of IQP

We consider s-linear rounding and “outward rotation” rounding.

Idea:

• Exploit randomness of algor to give a guarantee on dispersion.
• View random hyperplane Z as part of problem instance.
• Prove that whp, for any $\alpha \geq \frac{1}{2}$, the set of $u_i$ are

\[
\left(T^{1-\alpha}, O(nT^\alpha \sqrt{\log n})\right)-dispersed
\]

• Lipschitz value depends on which class of rounding schemes.
Example: Knapsack

A problem instance is collection of items $i$, each with a value $v_i$ and a size $s_i$, along with a knapsack capacity $C$.

Goal: select most valuable subset of items that fits. Find subset $T$ to maximize $\sum_{i \in T} v_i$ subject to $\sum_{i \in T} s_i \leq C$.

Natural family of algorithms: given $\rho$, select the better of:

- Greedily pack items in order of value (highest value first) subject to feasibility.
- Greedily pack items in order of $\frac{v_i}{s_i^\rho}$ subject to feasibility.

We show: under natural conditions on items (every pair of items has a bounded joint value distribution), can get $\tilde{O}(n^2\sqrt{T})$ regret.
Summary

• Strong performance guarantees for data driven algorithm selection.

• Exploit structure to overcome in-stability.

• Related work: sample complexity of automated mechanism design.
  
  [Balcan-Sandholm-Vitercik, NIPS’16]

Future Work: analyze other problems (e.g., tuning parameters in deep networks)