Accelerating data-driven algorithm design

November, 2024

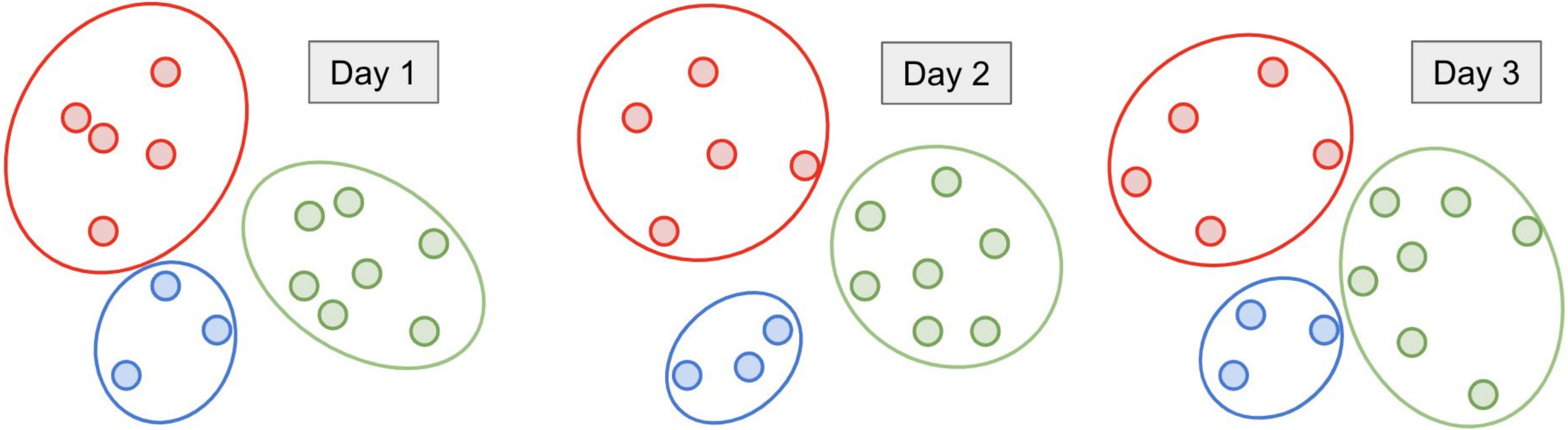
Nina Balcan, Chris Seiler, **Dravy Sharma**

NEURAL INFORMATION PROCESSING SYSTEMS

Data-driven algorithm design

Data-driven algorithm design is a framework for learning algorithms

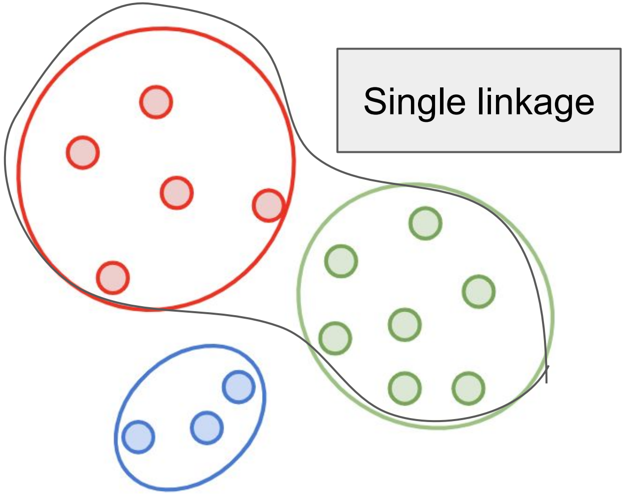
- Algorithms are concepts, and problem instances are data



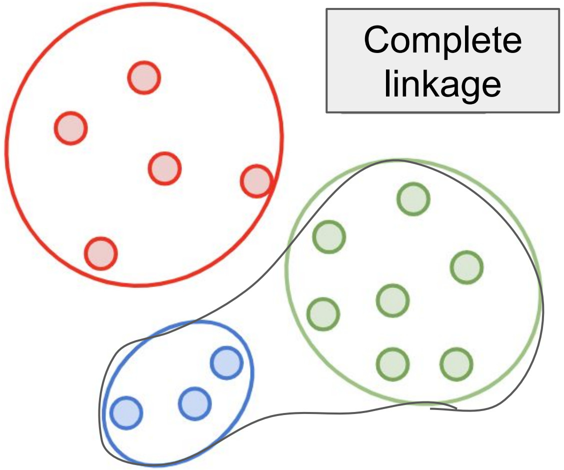
Data-driven algorithm design

Data-driven algorithm design is a framework for learning algorithms

- Algorithms are concepts, and problem instances are data
- Typically parameterized algorithm families over continuous space C



Merge cluster pairs A,B minimizing $\min_{a \in A, b \in B} d(a,b)$



Merge cluster pairs A,B minimizing $\max_{a \in A, b \in B} d(a,b)$

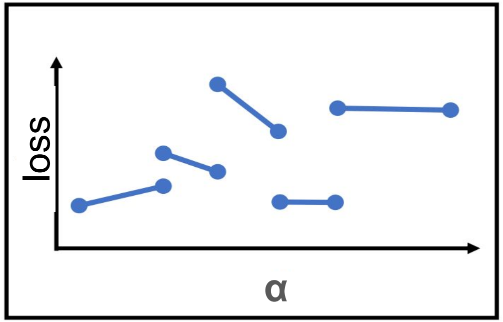
Family of heuristics:

Merge cluster pairs A,B minimizing $\boldsymbol{\alpha} \min_{a \in A, b \in B} d(a,b) + \mathbf{(1-\alpha)} \max_{a \in A, b \in B} d(a,b)$

Data-driven algorithm design

Data-driven algorithm design is a framework for learning algorithms

- Algorithms are concepts, and problem instances are data
- Typically parameterized algorithm families over continuous space C
- Loss function is often piecewise-structured



Prior work

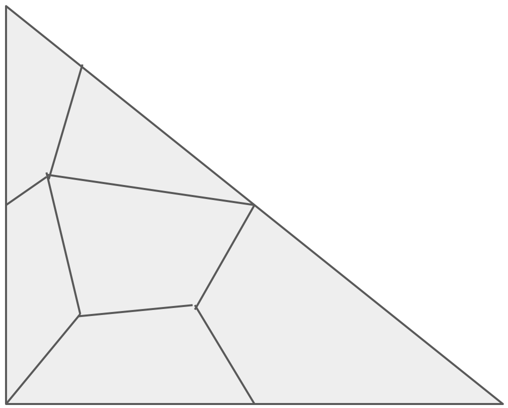
Bounded **sample complexity**:

- Poly number of instances needed to learn the best algorithm parameter
- Typically achieved by ERM (Empirical Risk Minimization)
  - Minimize loss on training samples

Computational complexity: ??

Challenge: Computing the pieces of the piecewise loss function efficiently

Linkage-based clustering



We have a collection of linkage heuristics:

- Single linkage
- Complete linkage
- Median linkage

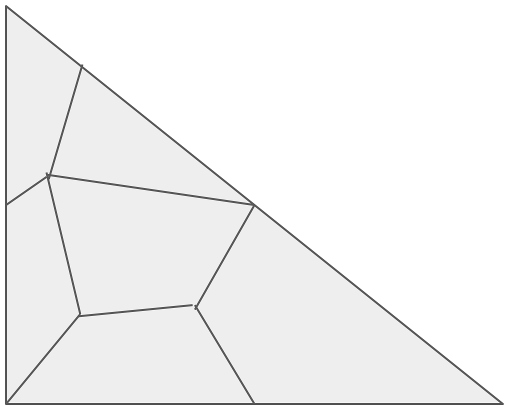
Loss is a piecewise constant function of interpolation parameters

Worst-case: number of pieces can be exponential in number of parameters!

Linkage-based clustering

We have a collection of linkage heuristics:

- Single linkage
- Complete linkage
- Median linkage



Loss is a piecewise constant function of interpolation parameters

Worst-case: number of pieces can be exponential in number of parameters!

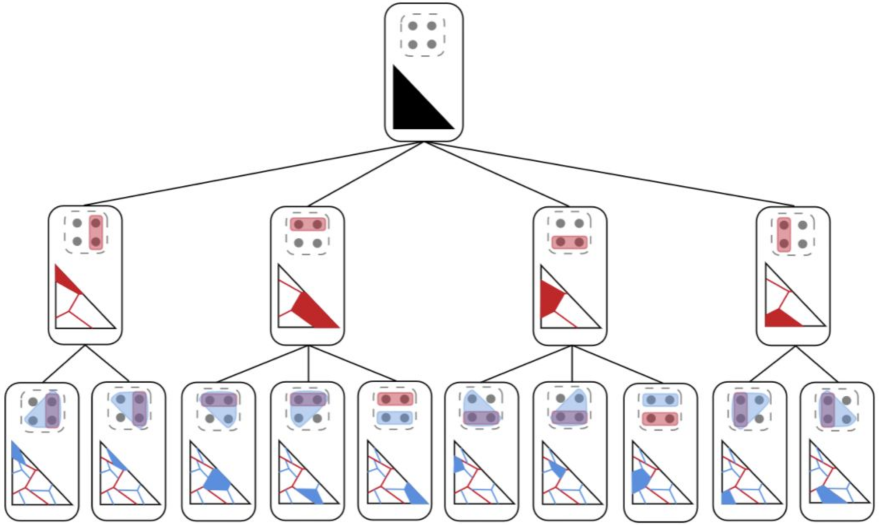
Our result: Loss can be computed efficiently whenever number of pieces is small

Key novel ideas

- Execution tree
- Clarkson’s algorithm

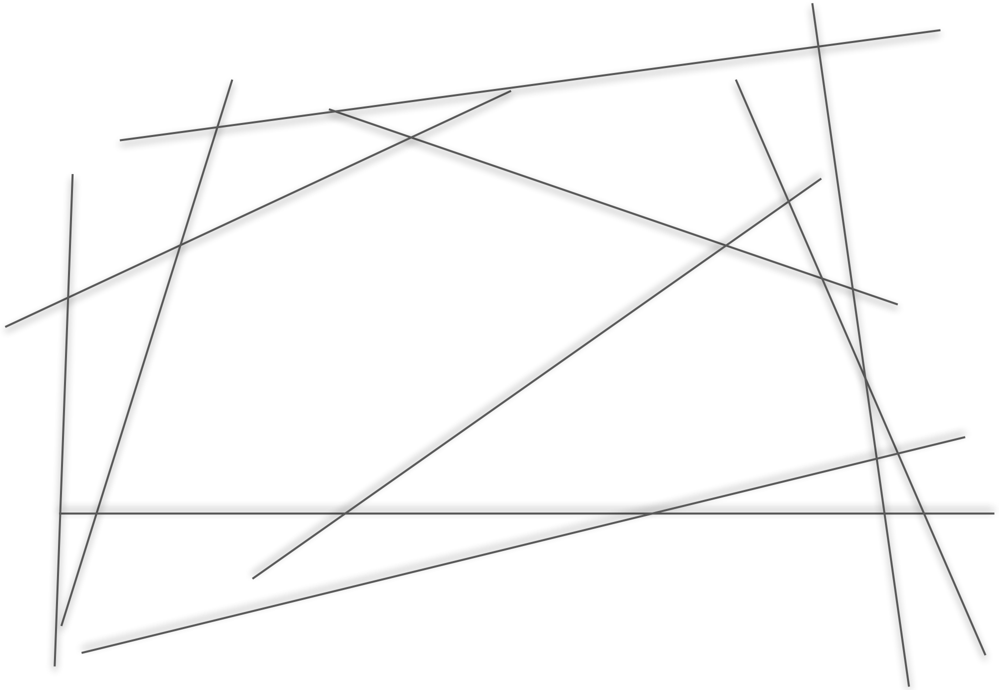
Execution tree

Compute the refinement of pieces induced by each merge step



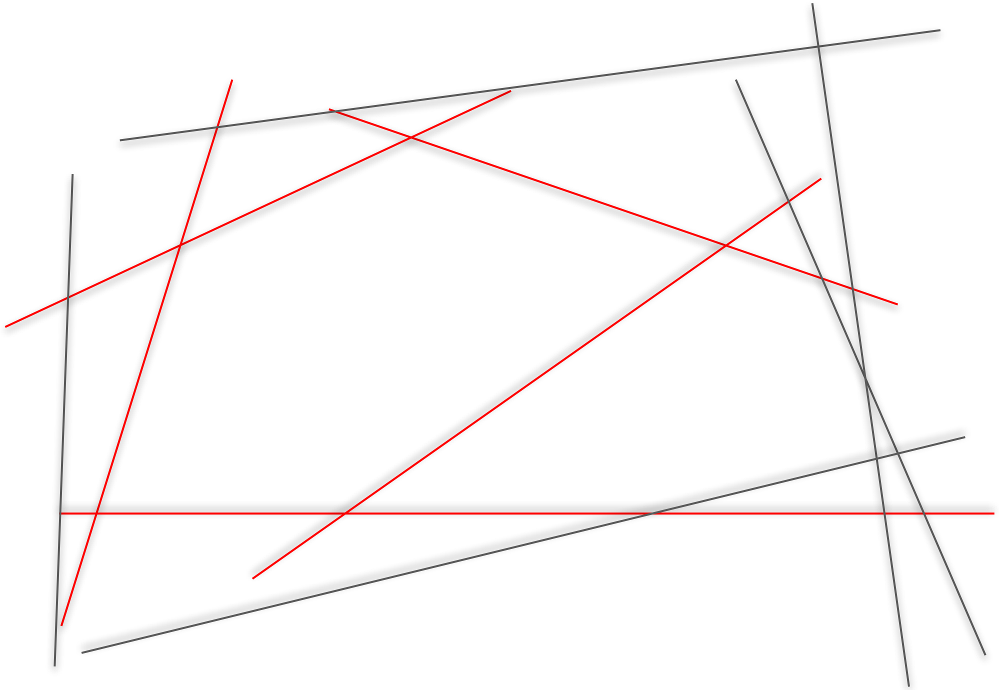
Clarkson’s algorithm

Compute the set of non-redundant hyperplanes in a linear system in output-sensitive time



Clarkson's algorithm

Compute the set of non-redundant hyperplanes in a linear system in output-sensitive time



Key result (informal)

Suppose the loss (as a function of the hyperparameter, on a fixed instance) is piecewise-structured with linear boundaries.

Then ERM can be implemented by solving R linear programs, where R is the number of pieces that actually appear in the loss function.

Applications

Problem	Dimension	Prior work (one instance)	T_S (one instance)
Two-part tariff pricing	$\ell = 1$	$O(K^3)$ [BPS20]	$\tilde{O}(R + K)$,
	any ℓ	$K^{O(\ell)}$ [BPS20]	$\tilde{O}(R^2 K)$
Linkage-based clustering	$d = 2$	$O(n^{18} \log n)$ [BDL20]	$O(Rn^3)$
	any d	$O(n^{8d+2} \log n)$ [BDL20]	$\tilde{O}(R^2 n^3)$
DP-based sequence alignment	$d = 2$	$O(R^2 + RT_{\text{DP}})$ [GBN94]	$O(RT_{\text{DP}})$
	any d	$s^{O(sd)} T_{\text{DP}}$ [BDD+21]	$\tilde{O}(\tilde{R}^{2L+1} T_{\text{DP}})$