Algorithm configuration with piecewise constant objectives

Zijian Wang, Yujia Zhang

Quiz questions

1. [Vitercik20] In some algorithm configuration problems, performing uniform sampling in the configuration space may completely miss regions that lead to near-optimal behavior.
   a. True
   b. False
2. [BSV20] For an extremely simple algorithm selector, it would not overfit for a large portfolio.
   a. True
   b. False
   c. It depends on the dimension of feature vectors
3. [BSV20] The authors majorly apply Natarajan dimension and pseudo-dimension for the derivation of sample complexity bound.
   a. True
   b. False
   c. Neither

Outline

- Piecewise constant objectives
- An algorithm for selecting "good" parameter regions
- Generalization

Piecewise constant behavior

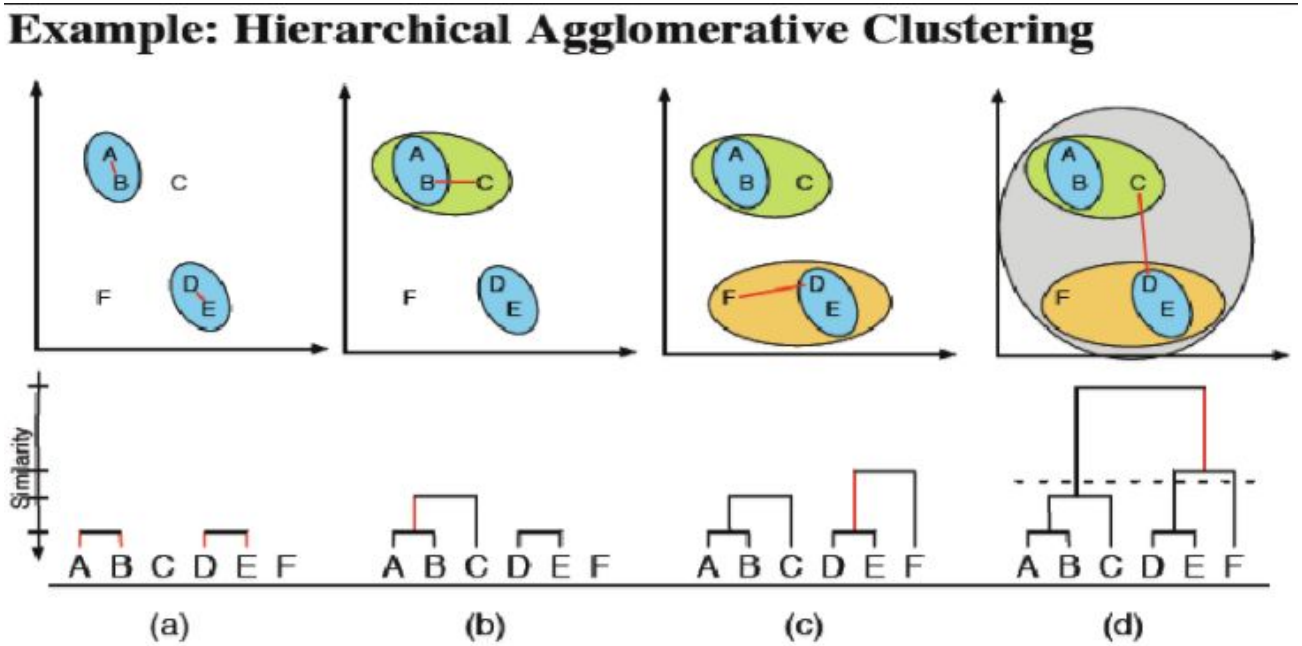
- In some applications, the performance of an algorithm is piecewise constant in its configuration parameters.
  - branch and bound
  - hierarchical clustering
- The number of pieces can grow with the data size.
- Small changes in the parameters of an algorithm might lead to big changes in the output

→ How do we find a set of good parameters that contain the "good pieces"?

Example: clustering

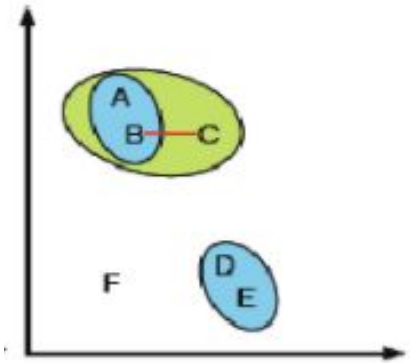
- The goal of clustering: partition the points into groups such that
  - distances within groups are minimized
  - distances between groups are maximized
- Objectives: k-means, k-median, k-center
- Hierarchical agglomerative clustering
  - start with each individual point as a cluster
  - sequentially **merge** smaller clusters based on a "linkage function"
  - then **prune** to decide how many clusters to keep

Hierarchical agglomerative clustering



http://machinelearningstories.blogspot.com/2017/09/hierarchical-clustering-bottom-up.html

Merging, then prune



Commonly used "linkage functions":

- minimum pairwise distance
- maximum pairwise distance
- average pairwise distance

Can "interpolate" between these ideas by constructing an infinite family of merge functions

Families of linkage functions, parameterized by ϱ

$$\mathcal{A}_1 = \left\{ \xi_{1,\rho} : (A,B) \mapsto \left(\min_{u\in A, v\in B} (d(u,v))^\rho + \max_{u\in A, v\in B} (d(u,v))^\rho \right)^{1/\rho} \,\middle|\, \rho \in \mathbb{R} \cup \{\infty, -\infty\} \right\},$$

$$\mathcal{A}_2 = \left\{ \xi_{2,\rho} : (A,B) \mapsto \rho \min_{u\in A, v\in B} d(u,v) + (1-\rho) \max_{u\in A, v\in B} d(u,v) \,\middle|\, \rho \in [0,1] \right\},$$

$$\mathcal{A}_3 = \left\{ \xi_{3,\rho} : (A,B) \mapsto \left(\frac{1}{|A||B|} \sum_{u\in A, v\in B} (d(u,v))^\rho \right)^{1/\rho} \,\middle|\, \rho \in \mathbb{R} \cup \{\infty, -\infty\} \right\}.$$

A separate paper (Balcan et al. 2017) shows that:

- There is not a good value of rho that works well for all instances of clustering problems.
- The clustering objective is made up of piecewise constant components, i.e., there exists a partition of the ϱ space such that two points in the same partition result in identical sequences of merging.
- The number of pieces grows polynomially with the number of datapoints.

Configuration selection

- How do we find good configurations from an infinite space?
- High-level idea:
  - find a finite subset of the infinite space that contains a "good" configuration with high probability
  - run configuration selection algorithm in finite space
  - the selected configuration is "good" even in the infinite space

Defining "goodness" of configurations

- The beginning of the δ-tail of the loss function

$$t_\delta(\boldsymbol{\rho}) = \operatorname{argmax}_{\tau \in \mathbb{Z}} \{\mathbb{P}_{j \sim \Gamma}(\ell(\boldsymbol{\rho}, j) \geq \tau) \geq \delta\}$$

- The best expected $c\delta$-capped expected loss in the entire (infinite) parameter space over all problem instances

$$OPT_{c\delta} = \inf_{\boldsymbol{\rho} \in \mathcal{P}} \{\mathbb{E}_{j \sim \Gamma} [\min\{\ell(\boldsymbol{\rho}, j), t_{c\delta}(\boldsymbol{\rho})\}]\}$$

- Some configurations can have a huge loss on a few rare problem instances. Capping the loss allows us to work with a notion of "approximate optimality".

Defining "goodness" of configurations

- The beginning of the δ-tail of the loss function

$$t_\delta(\boldsymbol{\rho}) = \text{argmax}_{\tau \in \mathbb{Z}} \{\mathbb{P}_{j \sim \Gamma}(\ell(\boldsymbol{\rho}, j) \geq \tau) \geq \delta\}$$

- The best expected $c\delta$-capped expected loss in the entire (infinite) parameter space over all problem instances

$$OPT_{c\delta} = \inf_{\boldsymbol{\rho} \in \mathcal{P}} \{\mathbb{E}_{j \sim \Gamma} [\min\{\ell(\boldsymbol{\rho}, j), t_{c\delta}(\boldsymbol{\rho})\}]\}$$

Definition 2.2 ($(\epsilon, \delta, \mathcal{P})$-optimality)**.** A parameter vector $\hat{\boldsymbol{\rho}}$ is $(\epsilon, \delta, \mathcal{P})$-*optimal* if

$$\mathop{\mathbb{E}}_{j \sim \Gamma} \left[\min \left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_\delta\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq (1+\epsilon) \inf_{\boldsymbol{\rho} \in \mathcal{P}} \left\{ \mathop{\mathbb{E}}_{j \sim \Gamma} \left[\min \left\{\ell\left(\boldsymbol{\rho}, j\right), t_{\delta/2}(\boldsymbol{\rho})\right\}\right] \right\}.$$

(ϵ, δ)-optimal subsets

Definition 2.2 ($(\epsilon, \delta, \mathcal{P})$-optimality)**.** A parameter vector $\hat{\boldsymbol{\rho}}$ is $(\epsilon, \delta, \mathcal{P})$-*optimal* if

$$\mathop{\mathbb{E}}_{j\sim\Gamma}\left[\min\left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_{\delta}\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq (1+\epsilon)\inf_{\boldsymbol{\rho}\in\mathcal{P}}\left\{\mathop{\mathbb{E}}_{j\sim\Gamma}\left[\min\left\{\ell\left(\boldsymbol{\rho}, j\right), t_{\delta/2}(\boldsymbol{\rho})\right\}\right]\right\}.$$

- A finite subset containing a "good" configuration

Definition 3.1 ((ϵ, δ)-optimal subset)**.** A finite set $\bar{\mathcal{P}} \subset \mathcal{P}$ is an (ϵ, δ)-*optimal subset* if there is a vector $\hat{\boldsymbol{\rho}} \in \bar{\mathcal{P}}$ such that $\mathbb{E}_{j\sim\Gamma}\left[\min\left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_{\delta/2}\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq \sqrt{1+\epsilon}\cdot OPT_{\delta/4}$.

(ϵ, δ)-optimal subsets

Definition 2.2 ($(\epsilon, \delta, \mathcal{P})$-optimality)**.** A parameter vector $\hat{\boldsymbol{\rho}}$ is $(\epsilon, \delta, \mathcal{P})$-*optimal* if

$$\mathop{\mathbb{E}}_{j\sim\Gamma}\left[\min\left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_\delta\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq (1+\epsilon)\inf_{\boldsymbol{\rho}\in\mathcal{P}}\left\{\mathop{\mathbb{E}}_{j\sim\Gamma}\left[\min\left\{\ell\left(\boldsymbol{\rho}, j\right), t_{\delta/2}(\boldsymbol{\rho})\right\}\right]\right\}.$$

- A finite subset containing a "good" configuration

Definition 3.1 ((ϵ, δ)-optimal subset)**.** A finite set $\bar{\mathcal{P}} \subset \mathcal{P}$ is an (ϵ, δ)-*optimal subset* if there is a vector $\hat{\boldsymbol{\rho}} \in \bar{\mathcal{P}}$ such that $\mathbb{E}_{j\sim\Gamma}\left[\min\left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_{\delta/2}\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq \sqrt{1+\epsilon}\cdot OPT_{\delta/4}$.

- If you search within the finite subset containing a "good" configuration, you will find a configuration whose performance is near-optimal over the infinite space

Theorem 3.2. *Let $\bar{\mathcal{P}} \subset \mathcal{P}$ be an (ϵ, δ)-optimal subset and let $\epsilon' = \sqrt{1+\epsilon} - 1$. Suppose $\hat{\boldsymbol{\rho}} \in \bar{\mathcal{P}}$ is $\left(\epsilon', \delta, \bar{\mathcal{P}}\right)$-optimal. Then $\mathbb{E}_{j\sim\Gamma}\left[\min\left\{\ell\left(\hat{\boldsymbol{\rho}}, j\right), t_\delta\left(\hat{\boldsymbol{\rho}}\right)\right\}\right] \leq (1+\epsilon)\cdot OPT_{\delta/4}$.*

Learning (ϵ, δ)-optimal subsets

Algorithm 1 Algorithm for learning (ϵ, δ)-optimal subsets

Input: Parameters $\delta, \zeta \in (0,1), \epsilon > 0$.

Set $\eta \leftarrow \min\left\{\frac{1}{8}\left(\sqrt[4]{1+\epsilon}-1\right), \frac{1}{9}\right\}$, $t \leftarrow 1$, $T \leftarrow \infty$, and $\mathcal{G} \leftarrow \emptyset$.
while $2^{t-3}\delta < T$ **do**
    Set $\mathcal{S}_t \leftarrow \{j\}$, where $j \sim \Gamma$.
    while $\eta\delta < \sqrt{\frac{2d \ln|\textsc{Partition}(\mathcal{S}_t, 2^t)|}{|\mathcal{S}_t|}} + \sqrt{\frac{8}{|\mathcal{S}_t|} \ln \frac{8(2^t|\mathcal{S}_t|t)^2}{\zeta}}$ **do** Draw $j \sim \Gamma$ and add j to $\mathcal{S}_t$.
    Compute tuples $(\mathcal{P}_1, z_1, \boldsymbol{\tau}_1), \ldots, (\mathcal{P}_k, z_k, \boldsymbol{\tau}_k) \leftarrow \textsc{Partition}\left(\mathcal{S}_t, 2^t\right)$.
    for $i \in \{1, \ldots, k\}$ with $z_i \geq 1 - 3\delta/8$ **do**
        Set $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{P}_i\}$.
        Sort the elements of $\boldsymbol{\tau}_i$: $\tau_1 \leq \cdots \leq \tau_{|\mathcal{S}_t|}$.
        Set $T' \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{m=1}^{|\mathcal{S}_t|} \min\left\{\tau_m, \tau_{\lfloor|\mathcal{S}_t|(1-3\delta/8)\rfloor}\right\}$.
        if $T' < T$ **then** Set $T \leftarrow T'$.
    $t \leftarrow t + 1$.
For each set $\mathcal{P}' \in \mathcal{G}$, select a vector $\boldsymbol{\rho}_{\mathcal{P}'} \in \mathcal{P}'$.

Output: The (ϵ, δ)-optimal set $\{\boldsymbol{\rho}_{\mathcal{P}'} \mid \mathcal{P}' \in \mathcal{G}\}$.

Learning (ϵ, δ)-optimal subsets

Algorithm 1 Algorithm for learning (ϵ, δ)-optimal subsets

Input: Parameters $\delta, \zeta \in (0,1), \epsilon > 0$.

Set $\eta \leftarrow \min\left\{\frac{1}{8}\left(\sqrt[4]{1+\epsilon}-1\right), \frac{1}{9}\right\}$, $t \leftarrow 1$, $T \leftarrow \infty$, and $\mathcal{G} \leftarrow \emptyset$.

while $2^{t-3}\delta < T$ **do**

  Set $\mathcal{S}_t \leftarrow \{j\}$, where $j \sim \Gamma$.

  while $\eta\delta < \sqrt{\frac{2d \ln |\textsc{Partition}(\mathcal{S}_t, 2^t)|}{|\mathcal{S}_t|}} + \sqrt{\frac{8}{|\mathcal{S}_t|} \ln \frac{8(2^t |\mathcal{S}_t| t)^2}{\zeta}}$ **do** Draw $j \sim \Gamma$ and add j to $\mathcal{S}_t$.

  Compute tuples $(\mathcal{P}_1, z_1, \boldsymbol{\tau}_1), \ldots, (\mathcal{P}_k, z_k, \boldsymbol{\tau}_k) \leftarrow \textsc{Partition}\left(\mathcal{S}_t, 2^t\right)$.

  for $i \in \{1, \ldots, k\}$ with $z_i \geq 1 - 3\delta/8$ **do**

    Set $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{P}_i\}$.

    Sort the elements of $\boldsymbol{\tau}_i$: $\tau_1 \leq \cdots \leq \tau_{|\mathcal{S}_t|}$.

    Set $T' \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{m=1}^{|\mathcal{S}_t|} \min\left\{\tau_m, \tau_{\lfloor |\mathcal{S}_t|(1-3\delta/8) \rfloor}\right\}$.

    if $T' < T$ **then** Set $T \leftarrow T'$.

  $t \leftarrow t+1$.

For each set $\mathcal{P}' \in \mathcal{G}$, select a vector $\boldsymbol{\rho}_{\mathcal{P}'} \in \mathcal{P}'$.

Output: The (ϵ, δ)-optimal set $\{\boldsymbol{\rho}_{\mathcal{P}'} \mid \mathcal{P}' \in \mathcal{G}\}$.

T: estimate of near-opt. loss

$\mathcal{G}$: set of good configurations

$\vec{\tau}_i$: 2^t-capped loss for each $j \in \mathcal{S}_t$

z_i: fraction of instances $j \in \mathcal{S}_t$ with loss $< 2^t$

← sorted 2^t-capped loss of $j \in \mathcal{S}_t$ under configs in $\mathcal{P}_i$

← approximate $\mathrm{OPT}_{\delta/4}$

← budget cap 2^t doubles every round

recall $\mathrm{OPT}_{\delta/4} = \inf_{\rho \in \mathcal{P}} \mathbb{E}_{j \sim \mathcal{P}}\left[\min\left(\ell(\rho, j), t_{\delta/4}(\rho)\right)\right]$

where $t_{\delta/4}(\rho) = \operatorname{argmax}_{\tau \in \mathbb{Z}}\left\{\mathbb{P}_{j \sim \mathcal{P}}\left(\ell(\rho, j) \geq \tau\right) \geq \delta\right\}$

Algorithm complexity analysis

Theorem 4.1. *With probability* $1-\zeta$*, the following conditions hold, with*

$$\eta = \min\left\{\frac{\sqrt[4]{1+\epsilon}-1}{8}, \frac{1}{9}\right\} \quad \textit{and} \quad c = \frac{16\sqrt[4]{1+\epsilon}}{\delta}:$$

1. *Algorithm 1 terminates after* $\bar{t} = O\left(\log\left(c \cdot OPT_{\delta/4}\right)\right)$ *iterations.*
2. *Algorithm 1 returns an* (ϵ, δ)*-optimal set of parameters of size at most*

$$\sum_{t=1}^{\bar{t}} \left|\textsc{Partition}\left(\mathcal{S}_t, c \cdot OPT_{\delta/4}\right)\right|.$$

3. *The sample complexity on round* $t \in [\bar{t}]$*,* $|\mathcal{S}_t|$*, is*

$$\tilde{O}\left(\frac{d \ln\left|\textsc{Partition}\left(\mathcal{S}_t, c \cdot OPT_{\delta/4}\right)\right| + c \cdot OPT_{\delta/4}}{\eta^2\delta^2}\right).$$

References

Balcan, Maria-Florina, et al. "Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems." *Conference on Learning Theory*. PMLR, 2017. Link

Balcan, Maria-Florina, Tuomas Sandholm, and Ellen Vitercik. "Learning to optimize computational resources: Frugal training with generalization guarantees." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 04. 2020. Link

Balcan, Maria-Florina, Tuomas Sandholm, and Ellen Vitercik. "Generalization in portfolio-based algorithm selection." *arXiv preprint arXiv:2012.13315* (2020). Link

Generalization in Portfolio-Based Algorithm Selection

Zijian Leonard Wang

Environmental Process Concentration
School of Civil & Environmental Engineering, COE
Center for Research on Programmable Plant Systems
Cornell University,USA

- First provable guarantees for portfolio-based algorithm selection
  - How large the training set to ensure resulting algorithm selector's average performance over training set is close to its expected performance.

$$|\frac{1}{N}\sum_{z\in\mathbb{S}} u_{f(z)}(z) - \mathop{\mathbb{E}}_{z\in\mathbb{D}}[u_{f(z)}(z)]|$$

- If the portfolio is large, overfitting is inevitable, even with an extremely simple algorithm selector.
  - Experiments showed that, for every possible problem instance, there must exist a well-suited parameter setting as we increase portfolio size, which would definitely result in overfitting.
- It provided a tight (up to log factors) bound on the number of samples sufficient and necessary to ensure that the selector's average performance on the training set generalizes to its expected performance.

Three Key Concepts

- **Portfolio**: a set $\mathcal{P} = \{\rho_1, ..., \rho_k\} \subseteq \mathbb{R}$ of k parameter settings.
- **Algorithm selector**: a mapping $f : \mathcal{Z} \to \mathcal{P}$ from problem instances $z \in \mathcal{Z}$ to parameter settings $f(z) \in \mathcal{P}$. $\mathcal{F} = \{f_i\}_{i \in I}$ denotes collections of algorithm selectors, where $f_i = \cup_{n=1}^{N} f_n$ and N is an arbitrary integer.
- **Algorithm's performance as a function of its parameters**: Given an instance $z \in \mathcal{Z}$, the performance of parameter setting selected by $\hat{f}$ is $u_{\hat{f}(z)}(z)$. Expected quality is $\mathbb{E}_{z \sim \mathcal{D}}[u_{\hat{f}(z)}(z)]$
- Instance $z \in \mathcal{Z} \xrightarrow{\text{Selector } f}$ Portfolio $\rho \in \mathcal{P} \subseteq \mathbb{R} \xrightarrow{\text{Function } u}$ Performance $u_{\hat{f}(z)}(z)$

Notation

- Unknown distribution $\mathcal{D}$ over problem instances in $\mathcal{Z}$. Training set $\mathcal{S} = \{z_1, ..., z_N\} \sim \mathcal{D}^N$
- For every parameter $\rho \in \mathbb{R}$ of algorithm given an input $z \in \mathcal{Z}$, $\exists$ mapping $u_\rho : \mathcal{Z} \to [0, H]$ that measures the **performance** of algorithm. E.g., runtime or quality of algorithm's output.
- $\mathcal{U} = \{u_\rho : \rho \in \mathbb{R}\}$ denotes the set of all performance functions

Sample Complexity Bounds

Formally, we bound

$$|\frac{1}{N}\sum_{z\in\mathbb{S}} u_{f(z)}(z) - \mathop{\mathbb{E}}_{z\in\mathbb{D}}[u_{f(z)}(z)]|$$

Notation Assumption

- Primal function u_ρ and dual function $u_z^\star : \mathbb{R} \to \mathbb{R}$ for a fixed input, and $u_z^\star(\rho) = u_\rho(z)$
- **Assumption**: algorithmic performance is a piecewise-constant function of the parameters. That is, each function $u_z^\star$ is a piecewise constant with at most t pieces, for some $t \in \mathbb{Z}$

Definition

To apply multi-class dimension tools to analyze the **algorithm selectors** $\mathcal{F}$, the authors firstly defined a **multi-class function** $\bar{f}$ by partition, and then analyzed **pseudo-dimension bound** using definition of Natatarajan dimension as key tool.

Multi-class function $\bar{f}$

Given the selector $f : \mathcal{Z} \to \rho$, we let $\rho_1 < \rho_2 < \rho_3 ... < \rho_{\bar{\kappa}}$ for $\bar{\kappa} \leq \kappa$, and $\mathcal{Z}_1, ..., \mathcal{Z}_{\bar{\kappa}}$ partitioned by f where if $f(z) = \rho_i$, then $z \in \mathcal{Z}_i$. Hence, corresponding multi-class function $\bar{f} : \mathcal{Z} \to [\kappa]$ to show which instance $z \in \mathcal{Z}_i$ partition is mapped to which ρ_i.

Natarajan Dimension (to quantify intrinsic complexity of class of selectors)

Set $\bar{\mathcal{F}}$ multi-class **shatters** set $\mathcal{Z}$ with N instances if $\exists$ labels $y_i \in [\kappa]$ and $y_i' \in [\kappa]$, $i = 1, 2, ..., N$ such that: 1) $y_i \neq y_i'$ for $\forall i \in [N]$, 2)
$\forall$ subset $C \subseteq [N], \exists \bar{f} \in \bar{\mathcal{F}}$ s.t. $\bar{f}(z_i) = y_i$ if $i \in C$ and $\bar{f}(z_i) = y_i'$ otherwise.
Natarajan dimension of $\bar{\mathcal{F}} :=$ cardinality of largest set that can be multi-class shattered by $\bar{\mathcal{F}}$.

Definition

Pseudo-dimension

Similar to Natarajan Dimension, authors introduced pseudo-dimension for performance $\mathcal{U}_{\mathcal{F}}$. Set $\mathcal{U}_{\mathcal{F}}$ shatters N instances if exists N witness w_i such that $\forall$ subset $C \subseteq [N]$, $\exists f \in \mathcal{F}$ such that $u_{f(z_i)}(z_i) \leq w_i$ if $i \in C$ and $u_{f(z_i)}(z_i) < w_i$ otherwise. $Pdim(\mathcal{U}_{\mathcal{F}}) :=$ size of largest set of instances that can be shattered by $\mathcal{U}_{\mathcal{F}}$.

Recall the VC dimension, we have

$$\mathbb{P}(\text{test error} \leq \text{train error} + \text{distance}) = 1 - \delta$$

, where

$$\text{distance} = \sqrt{\frac{1}{N}[D(log\frac{2N}{D} + 1) - log\frac{\delta}{4}]}$$

. Similarly, we have

$$|\frac{1}{N}\sum_{z \in \mathbb{S}} u_{f(z)}(z) - \mathop{\mathbb{E}}_{z \in \mathbb{D}}[u_{f(z)}(z)]| = O(H\sqrt{\frac{1}{N}(Pdim(\mathcal{U}_{\mathcal{F}}) + log\frac{1}{\delta})})$$

Theorem

Theorem 1

Suppose each dual function $u_z^\star$ is piecewise-constant with at most t pieces. Let $\bar{d}$ be the Natarajan dimension of $\bar{\mathcal{F}}$. Then $Pdim(\mathcal{U}_\mathcal{F}) = \tilde{\mathcal{O}}(\bar{d} + \kappa logt)$. For N instances and κ parameters, $N = O((\kappa + \bar{d})log(\kappa + \bar{d}) + \kappa logt)$

Theorem 2

For any $\kappa, \bar{d} \geq 2$, there is a class of functions $\mathcal{U} = \{u_\rho : \rho \in \mathbb{R}\}$ and a class of selectors $\mathcal{F}$ such that:

- Each selector $f \in \mathcal{F}$ maps to $\leq k$ parameter settings.
- Each dual function $u_z^\star$ is piecewise-constant with 1 discontinuity.
- The Natarajan dimension of $\bar{\mathcal{F}}$ is at most $\bar{d}$, and
- The pseudo-dimension of $\mathcal{U}_\mathcal{F}$ is $\Omega(k + \bar{d})$

Linear Performance Models

For each problem case, $\exists$ feature mapping $\phi : \mathcal{Z} \to \mathbb{R}^m$ that assigns feature vectors $\phi(z) \in \mathbb{R}^m$ to problem instances $z \in \mathcal{Z}$.

Bound

Let $\rho = [\rho_1, ..., \rho_\kappa] \in \mathbb{R}^\kappa$ with κ distinct parameter settings. Define weight vector $w_i \in \mathbb{R}^m$ for each $i \in [\kappa]$. $< \phi(z), w_i >$ estimates the performance of algorithm parameterized by ρ_i on instance z. Thus, we pick parameter with best performence

$$f_{\rho,W}(z) = \rho_i, \text{ where } W = [w_1, w_2, ..., w_\kappa], \ i = argmax_{j \in [\kappa]}\{< w_j, \phi(z) >\}$$

To apply Natarajan dimension, m-dimensional linear class was defined. Rearrange above,

$$\bar{\mathcal{F}}_L = \{g_W : W \in \mathbb{R}^{m \times \kappa}\}, \text{ where } g_W(z) = argmax_{i \in [\kappa]}\{< w_i, \phi(z) >\}$$

Then, such m-dimensional linear class has Natarajan dimension of $\bar{d} = O(m\kappa)$.

Corollary

Suppose dual functions are piecewise-constant with at most t pieces.The instance number of $\mathcal{U}_{\mathcal{F}_L} = \{z \to u_{f(z)} : f \in \mathcal{F}_\mathcal{L}\}$ is $O(\kappa m log(\kappa m) + \kappa log t)$

Regression tree performance models

Regression tree T's leaf nodes partition the **feature space** $\mathbb{R}^m$ into disjoint regions $R_1, ..., R_\ell$. Each R_i corresponds to a constant c_i.

Notation Formulation

$\rho = (\rho_1, .., \rho_\kappa)$ is a set of κ distinct parameter settings. Each tree T_i corresponds to a ρ_i. $T = (T_1, ..., T_\kappa)$ is set of κ trees.
Define: Algorithm selector $f_{\rho,T}(z) = \rho_i$, where $i = argmax_{j \in \kappa}\{h_{T_j}(z)\}$, $h_{T_j}(z)$ represents the j-th tree's prediction of the algorithm's performance on instance z.

Lemma

Suppose we limit ourselves to building regression trees with at most ℓ leaves. Then the Natarajan dimension of $\bar{\mathcal{F}}_R$ is $O(\ell\kappa log(\ell\kappa m))$

Corollary

Suppose the dual functions are piecewise-constant with at most pieces and we limit ourselves to building regression trees with at most ℓ leaves. Then $Pdim(\mathcal{U}_{\mathcal{F}_R}) = O(\ell\kappa log(\ell\kappa m) + \kappa log t)$

Clustering-based algorithm selectors

Feature vectors $\phi(z_1), ..., \phi(z_N) \in \mathbb{R}^m$ and each cluster needs one good parameter setting. For each new instance z, the algorithm selector determines which cluster center is closest to $\phi(z)$.

Formulation

Let parameter settings $\rho = (\rho_1, ..., \rho_\kappa)$ and vector set $x_1, ..., x_\kappa \in \mathbb{R}^m$. Define algorithm selector $f_{\rho,X}(z) = \rho_i$ where $i = argmin_{j\in[\kappa]}\{\|x_j - \phi(z)\|_p\}$ for some ℓ_p-norm with $p \geq 1$. The class of algorithm selector $\mathcal{F}_C = \{f_{\rho,X} : \rho \in \mathbb{R}^\kappa, X \in \mathbb{R}^{m\times\kappa}\}$

Lemma

For any $p \in [1, \infty)$, *the Natarajan dimension of* $\bar{\mathcal{F}}_C$ *is* $O(m\kappa log(m\kappa p))$

Corollary

Suppose the dual functions are piecewise-constant with at most t pieces. Then $Pdim(\mathcal{U}_{\mathcal{F}_C}) = \tilde{O}(m\kappa + \kappa logt)$.

Learning procedure with guarantees

Procedures

1. Draw a training set of problem instances $\mathcal{S}$ $\mathcal{D}^N$.
2. Use the training set $\mathcal{S}$ to select a set of κ or fewer parameter settings $\hat{\mathcal{P}} \subseteq \mathbb{R}$.
3. Use $\mathcal{S}$ to learn an algorithm selector $\hat{f} \in \mathcal{F}$ that maps problem instances $z \in \mathcal{Z}$ to parameter settings $\hat{f}(z) \in \hat{\mathcal{P}}$.

Definition

1. Portfolio $\hat{\mathcal{P}}$ is nearly optimal over the training set in the sense that

$$\frac{1}{N}\sum_{z\in\mathcal{S}} \max_{\rho\in\hat{\mathcal{P}}} u_\rho(z) \geq \alpha \max_{\mathcal{P}\subset\mathbb{R}:|\mathcal{P}|\leq\kappa} \frac{1}{N}\sum_{z\in\mathcal{S}} \max_{\rho\in\mathcal{P}} u_\rho(z) - \beta$$

2. Algorithm selector $\hat{f}$ returns high-performing parameter settings from the set $\hat{\mathcal{P}}$ in the sense that

$$\frac{1}{N}\sum_{z\in\mathcal{S}} u_{\hat{f}(z)}(z) \geq \frac{1}{N}\sum_{z\in\mathcal{S}} \max_{\rho\in\mathcal{P}} u_\rho(z) - \epsilon$$

Learning procedure with guarantees continued

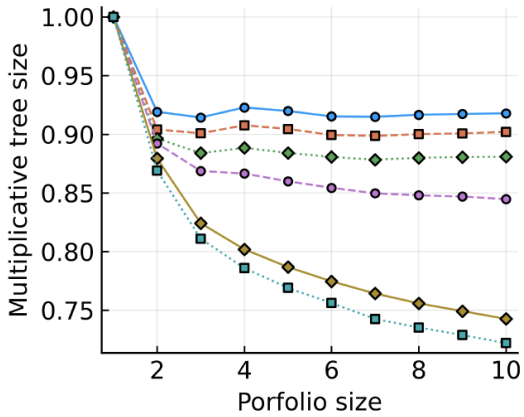
Theorem

Suppose each dual function $u_z^\star$ is piecewise constant with at most t pieces. Given a training set $\mathcal{S} \subseteq \mathcal{Z}$ of size N, suppose we learn an $(\alpha, \beta, \epsilon)$-optimal portfolio $\hat{\mathcal{P}} \subset \mathbb{R}$ and algorithm selector $\hat{f} : \mathcal{Z} \to \hat{\mathcal{P}}$ in $\mathcal{F}$. With probability $1 - \delta$ over the draw of the training set $\mathcal{S}$ $\mathcal{D}^N$,

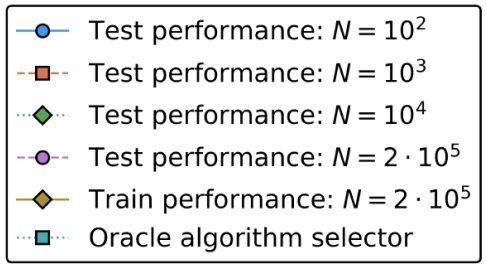
$$\mathbb{E}_{z\ \mathcal{D}}[u_{\hat{f}(z)}(z)] \geq \alpha \max_{\mathcal{P}:|\mathcal{P}|\leq\kappa} \mathbb{E}[\max_{\rho\in\mathcal{P}} u_\rho(z)] - \epsilon - \beta - \tilde{O}(H\sqrt{\frac{\bar{d}+\kappa}{N}})$$

Experiments

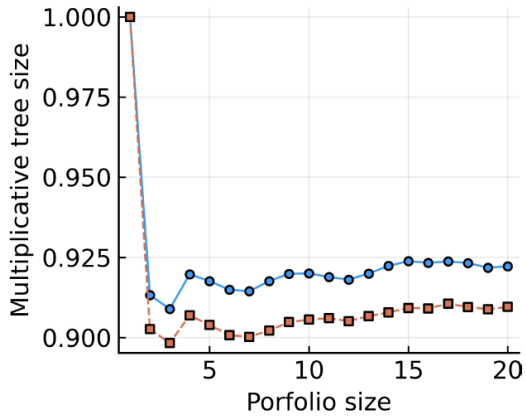
They run the experiments using CPLEX software to measure the performance (here, multiplicative tree size) over the portfolio size of 10 and 20.



(a) Plot with portfolio sizes 1 through 10.



(b) Legend for Figures 1a and 1c.



(c) Plot with portfolio sizes 1 through 20.

Figure 1: In Figures 1a and 1c, we plot the multiplicative tree size improvement we obtain as we increase both the portfolio size along the horizontal axis and the size of the training set, denoted N. Fixing a training set size and letting $\hat{v}_\kappa$ be the average tree size we obtain over the test set using a portfolio of size κ (see Equation (5)), we plot $\hat{v}_\kappa / \hat{v}_1$. In Figure 1a, the portfolio size ranges from 1 to 10 and the training set size N ranges from 100 to 200,000. In Figure 1c, the portfolio size ranges from 1 to 20 and the training set size ranges from 100 to 1000. In Figure 1a, we also plot a similar curve for the test performance of the oracle algorithm selector, as well as the training performance of the learned algorithm selector when $N = 2 \cdot 10^5$.