



Dec 2, 2025
San Diego



New Frontiers of Hyperparameter Optimization:

Recent advances and open challenges in theory and practice

Dravyansh (Dravy) Sharma
IDEAL, TTIC



Maria-Florina Balcan
CMU

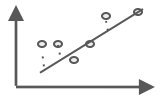
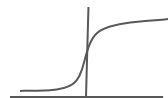

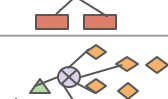

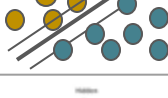


Colin White
Meta



What is a hyperparameter?

HP tuning is a special case of algorithm selection in Machine Learning

	Linear regression	Ridge/Lasso penalties
	Logistic regression	L1, L2 regularization penalties
	Decision Tree	splitting criterion, pruning cost, max depth
	k-Nearest neighbors	k, weights, metric, abstention threshold
	Support Vector Machines	C, kernel, gamma
	Neural Networks	activation function, learning schedule, ...

HPs define a collection of algorithms for learning a predictor

Why so common in ML?
Hard problems + role of data

Hyperparameter tuning and transfer

HP tuning is important across ML

- Data prep + HP tuning take up most of the applied ML researcher hours
- Takes up to 90% of the compute
- Critical in high-stakes and large-scale applications

HP transfer is crucial today!

- Unavoidable in LLMs where each of the above is magnified multifold!

Algorithm design for machine learning

- **Hyperparameter tuning** is poorly understood and yet of critical importance
 - why? ML works on data



- There is NO single best algorithm+hyperparameter!
 - Must tune/configure for the best performance on **domain-specific data**
- Current practices require incredible amounts of **compute** and **engineering** efforts, and yet with no guarantees!
- Understanding how the performance actually varies with the hyperparameter is crucial for **principled tuning**

Roadmap

- ❖ Introduction
- ❖ Major techniques used in practice
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ Data-driven algorithm design
 - Learning-theoretic guarantees
 - Distributional learning
 - Online learning
- ❖ Tuning core ML algorithms
 - Linear regression, decision trees
 - Semi-supervised learning, neural networks
- ❖ Conclusion

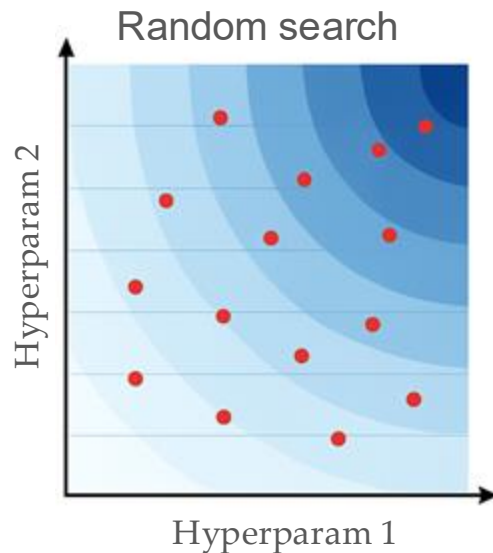
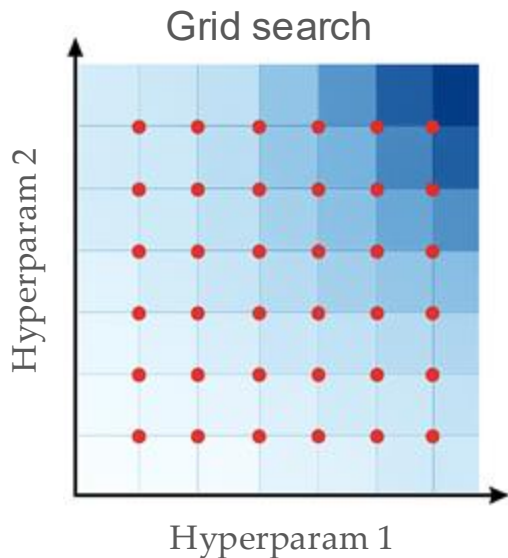


Roadmap

- ❖ Introduction
- ❖ **Major techniques used in practice**
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ Data-driven algorithm design
 - Learning-theoretic guarantees
 - Distributional learning
 - Online learning
- ❖ Tuning core ML algorithms
 - Linear regression, decision trees
 - Semi-supervised learning, neural networks
- ❖ Conclusion

Hyperparameter tuning setup

- Tune hyperparameters such as learning rate, batch size, weight decay
- $f(\alpha) = \text{val_loss}$
- Baselines: grid search, random search
- Black box optimization (zeroth order optimization)
 - no gradient info; treat function as a “black box”



Bayesian Optimization

- **Gaussian Process:**

- a collection of (infinitely many) random variables that are jointly Gaussian.
- a distribution over functions – models noisy evaluation of some $f(\alpha)$.
- given by a mean function $m(\alpha)$ and covariance $k(\alpha, \alpha')$.

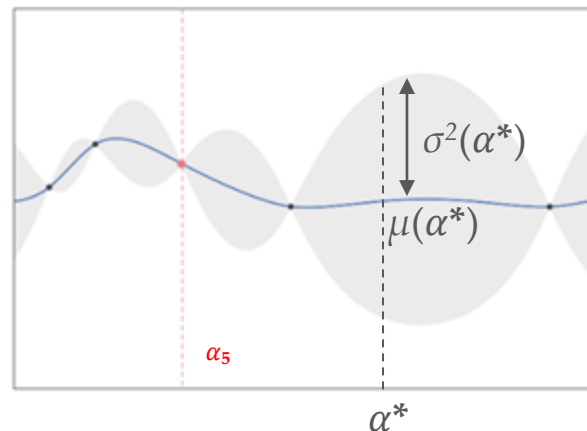
$$\mathbb{E}[f(\alpha)] = m(\alpha).$$

$$\mathbb{E}[(f(\alpha) - m(\alpha))(f(\alpha') - m(\alpha'))] = k(\alpha, \alpha').$$

- Since all finite collections of function values are assumed jointly Gaussian, the conditional distribution of any new point given the observed points is also Gaussian, i.e. posterior predictive mean and variance at α^* , given observed points A is

$$\mu(\alpha^*) = K(\alpha^*, A)K(A, A)^{-1}f(A).$$

$$\sigma^2(\alpha^*) = K(\alpha^*, \alpha^*) - K(\alpha^*, A)K(A, A)^{-1}K(A, \alpha^*).$$



$$A = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]$$

Bayesian Optimization

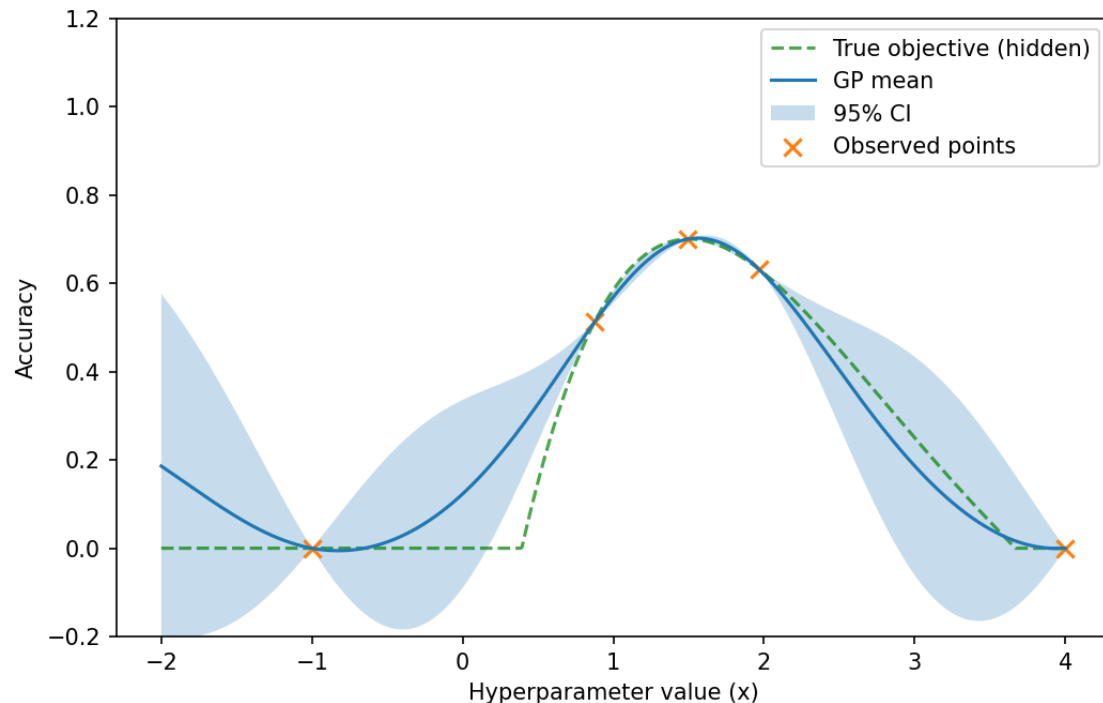
Acquisition function

- Trade off exploration vs. exploitation

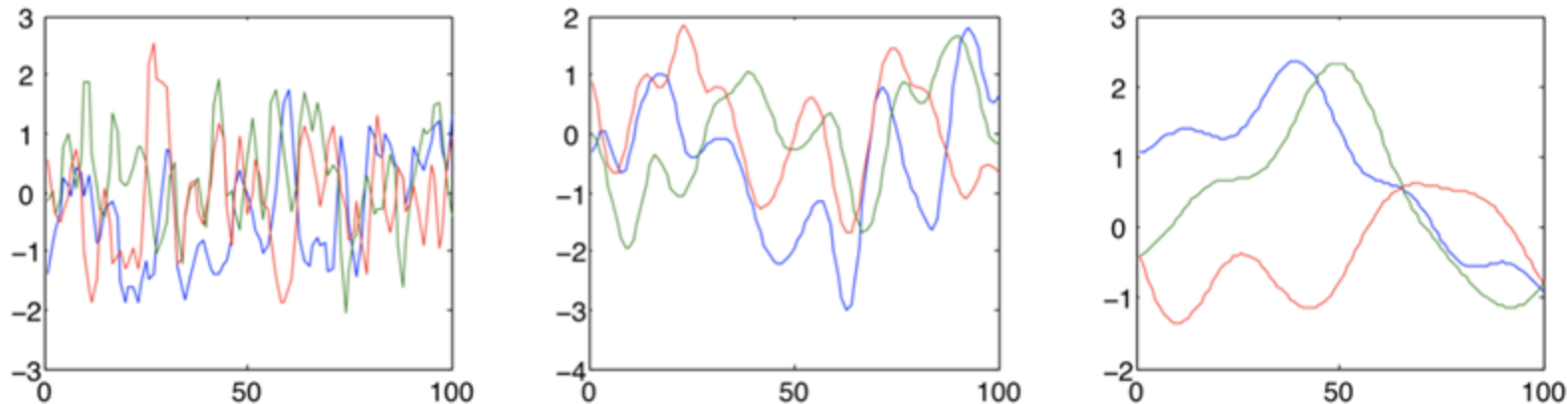
$$EI(\alpha) = E[\max(0, f_{\text{best}} - f(\alpha))]$$

Bayesian optimization:

```
for i in {0, ..., n}:  
    // use GP to compute EI  
    select  $\alpha^* = \max_{\alpha} EI(\alpha)$   
    compute val_loss of  $\alpha^*$ 
```



BO has its own hyperparameters!



[Frazier, 2022]

Assumption on the smoothness of the function $f(\alpha) = \text{val_loss}$
(without this assumption, convergence is slow)

[Berkenkamp, Schoellig, Krause JMLR 2019]

BO libraries: [Dragonfly: Kandasamy et al., JMLR 2020],
[SMAC3: Lindauer et al. JMLR 2022]

Bandit-based approaches

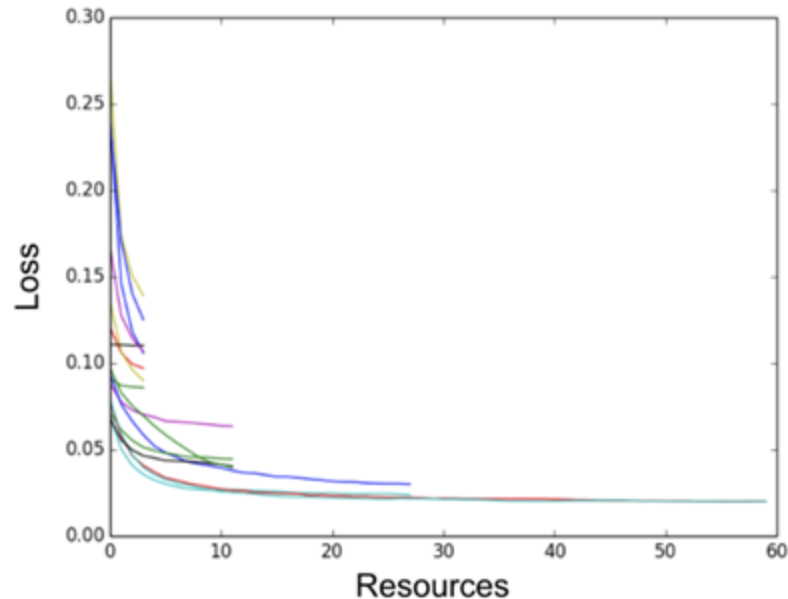
Random search with adaptive early stopping

Each arm has a noisy non-stationary reward that eventually converges to a limiting value

- **Successive halving**

```
Given sets of hyperparameters  $A$   
for  $i$  in  $\{0, \dots, 3\}$ :  
    run( $\alpha$ ,  $10 * 2^i$ ),  $\alpha \in A$   
     $A := \text{top\_k}(A, 16 * 2^{-i})$ 
```

- **Hyperband:** multiple runs of successive halving, across different hyper-hyperparameters



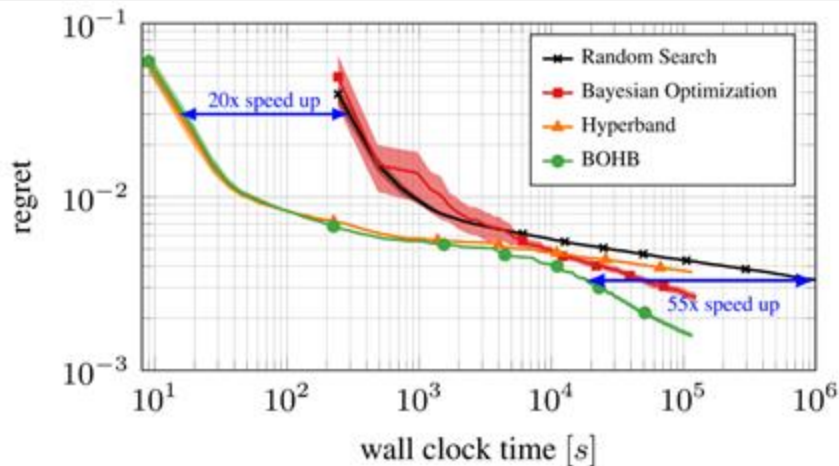
[Jamieson, Talwalkar (AISTATS 2016)]
[Li, Jamieson, DeSalvo, Rostamizadeh,
Talwalkar (JMLR 2018)]

Other approaches and speedups

BOHB (BO + Hyperband):

Run Hyperband, but replace the random selection of configurations at the beginning of each iteration by a model-based search

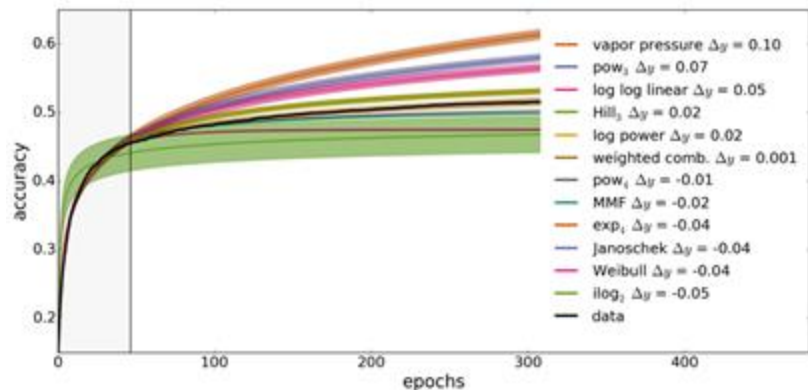
[Falkner, Klein, Hutter, ICML 2018]



Learning curve extrapolation:

Speed up HPO algorithms by extrapolating partial learning curves

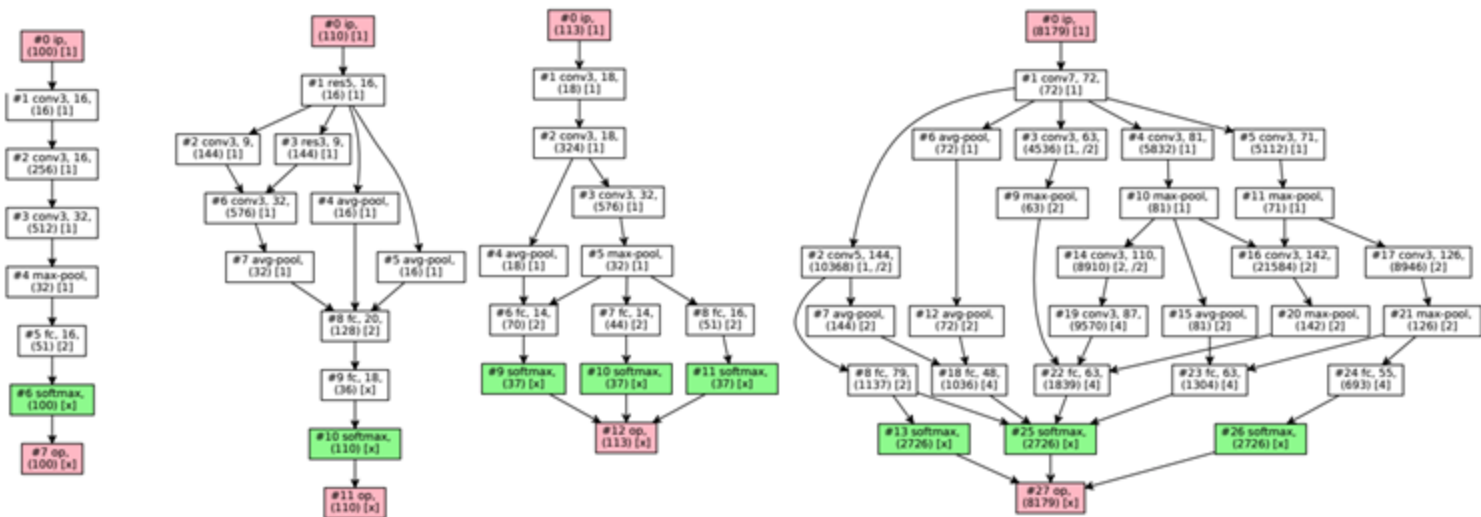
[Domhan, Springenberg, Hutter, IJCAI 2015]



Case study: Neural Architecture Search

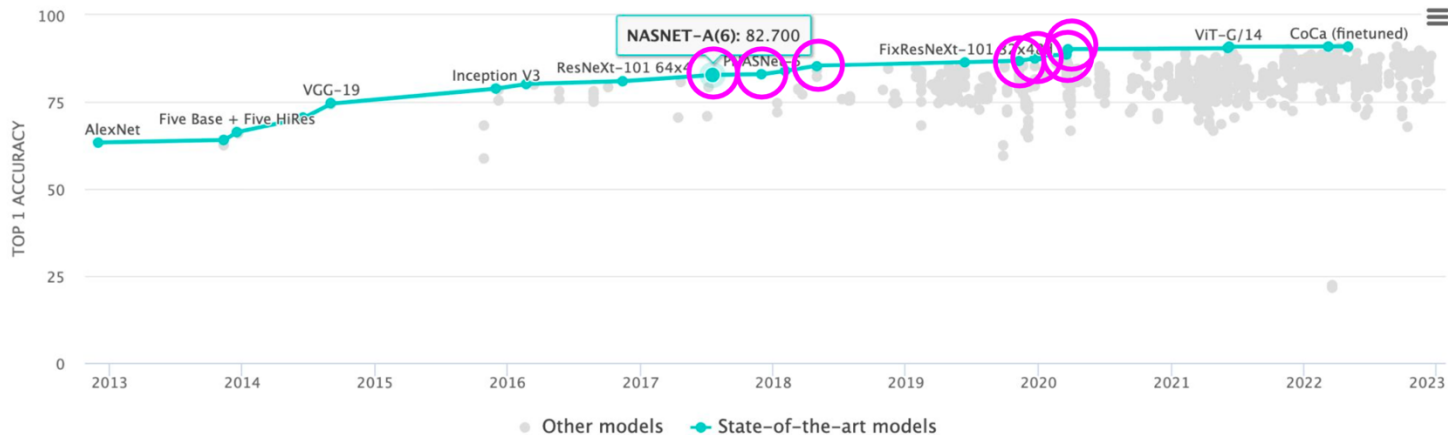
Define the search space as a DAG with architecture components (e.g. conv_3x3, conv_5x5, pool, fc)

- The search space is a critical decision [Li, Talwalkar, UAI 2019]



[Kandasamy, Neiswanger, Schneider, Póczos, Xing, NeurIPS 2018]

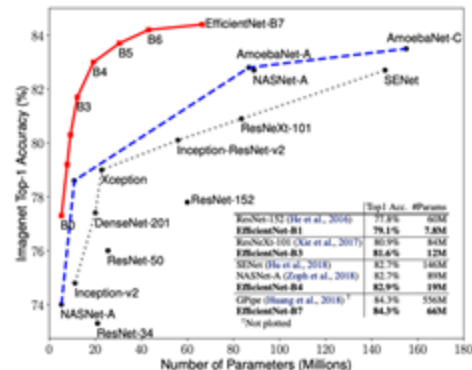
Case study: Neural Architecture Search (NAS)



[paperswithcode.com/sota/image-classification-on-imagenet, 2022]

- NAS has been used to achieve SotA on imagenet seven times since 2017
- NAS has also been used to discover efficient architectures such as EfficientNet

[Tan, Le, ICML 2019]

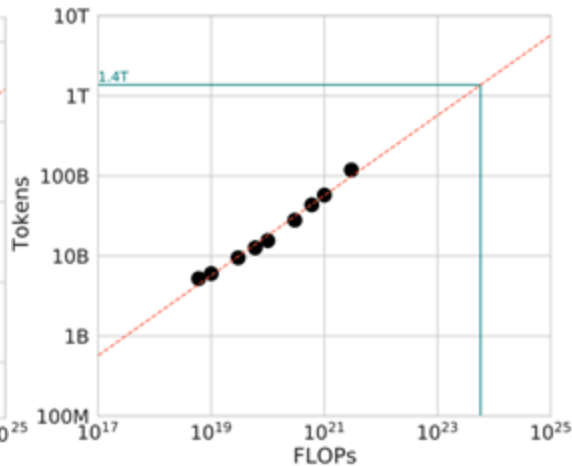
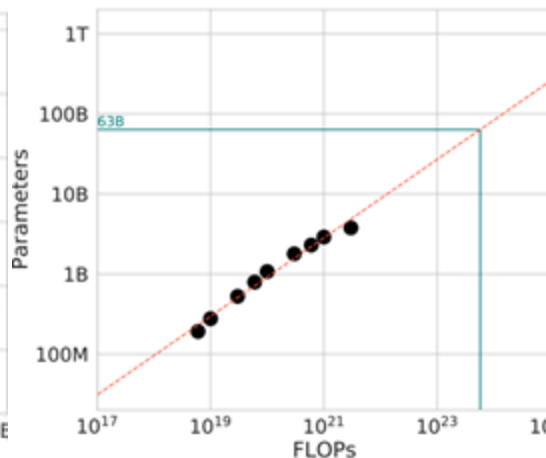
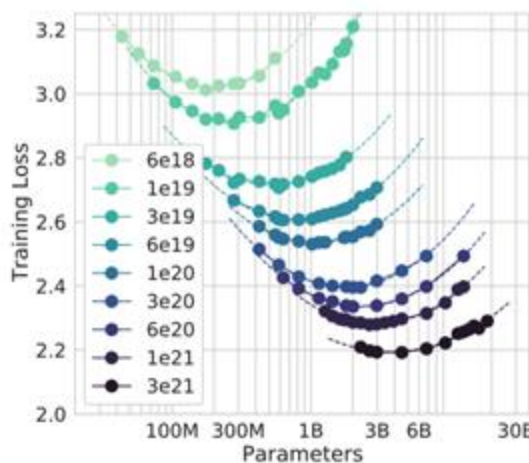


Large Language Models: Scaling Laws

Train large model once → Think in terms of scaling laws

Given a fixed computational budget, how to pick **#params** and **#tokens**?

- Isoflop analysis: sweep over model sizes for a fixed computational budget
- Fit an empirical trend across small models → scale up

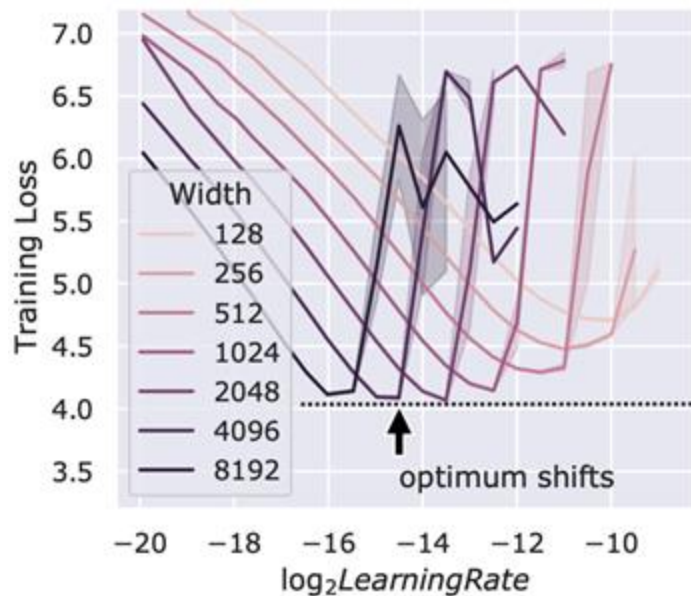


$$\text{FLOPs} \approx 6 * (\# \text{ params}) * (\# \text{ tokens})$$

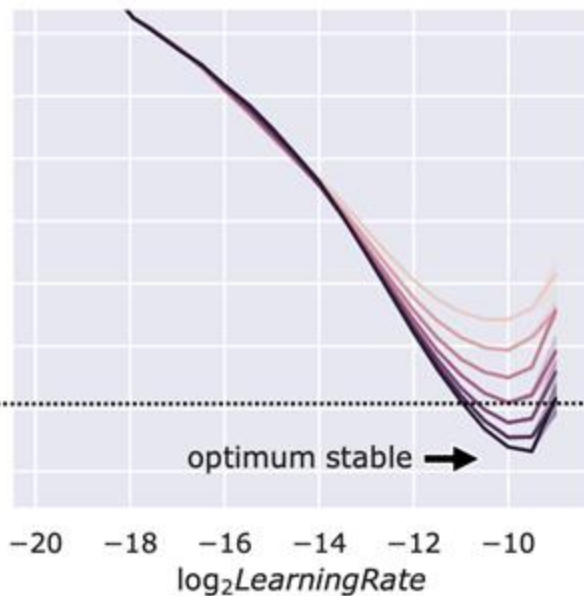
[Kaplan et al, 2020], [Hoffmann et al, 2022]

Large Language Models: selecting learning rate using muP

Traditional LR scaling law:



muP LR scaling law:



Maximum Update Parameterization (muP):

Parameterize the model such that the learning rate is same across all scales. Then tune LR once.

- Width-dependent LR
- Width-dependent weight initialization

[Yang, Hu, ICML 2021], [Yang et al., NeurIPS 2021]

Limitations of popular HPO approaches in practice

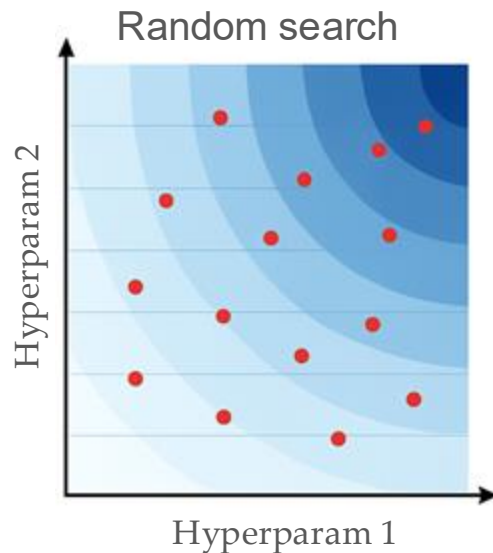
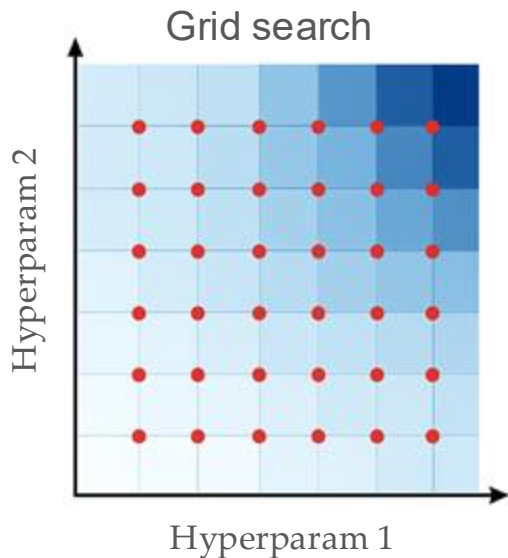
- Theoretical guarantees typically need strong assumptions
- Need to tune hyper-hyper-parameters
- Overtuning (val loss is a proxy for generalization) [Schneider, Bischl, Feurer, AutoML 2025]

Practitioners rely heavily on empirical findings!

All approaches are black-box!! (agnostic to the structure of the function)

Hyperparameter tuning setup

- Tune hyperparameters such as learning rate, batch size, weight decay
- $f(\alpha) = \text{val_loss}$
- Baselines: grid search, random search
- Black box optimization (zeroth order optimization)
 - no gradient info; treat function as a “black box”



Bayesian Optimization

- **Gaussian Process:**

- a collection of (infinitely many) random variables that are jointly Gaussian.
- a distribution over functions – models noisy evaluation of some $f(\alpha)$.
- given by a mean function $m(\alpha)$ and covariance $k(\alpha, \alpha')$.

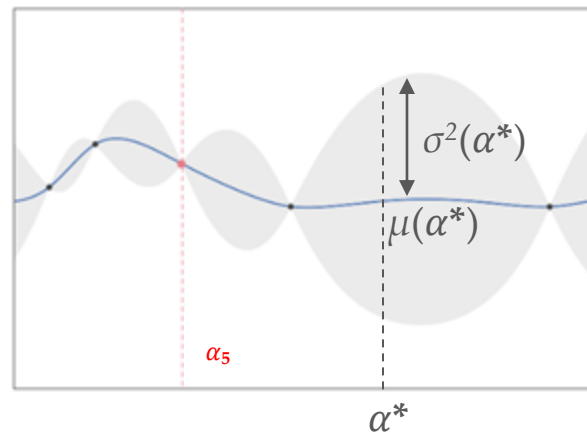
$$\mathbb{E}[f(\alpha)] = m(\alpha).$$

$$\mathbb{E}[(f(\alpha) - m(\alpha))(f(\alpha') - m(\alpha'))] = k(\alpha, \alpha').$$

- Since all finite collections of function values are assumed jointly Gaussian, the conditional distribution of any new point given the observed points is also Gaussian, i.e. posterior predictive mean and variance at α^* , given observed points A is

$$\mu(\alpha^*) = K(\alpha^*, A)K(A, A)^{-1}f(A).$$

$$\sigma^2(\alpha^*) = K(\alpha^*, \alpha^*) - K(\alpha^*, A)K(A, A)^{-1}K(A, \alpha^*).$$



$$A = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]$$

Bayesian Optimization

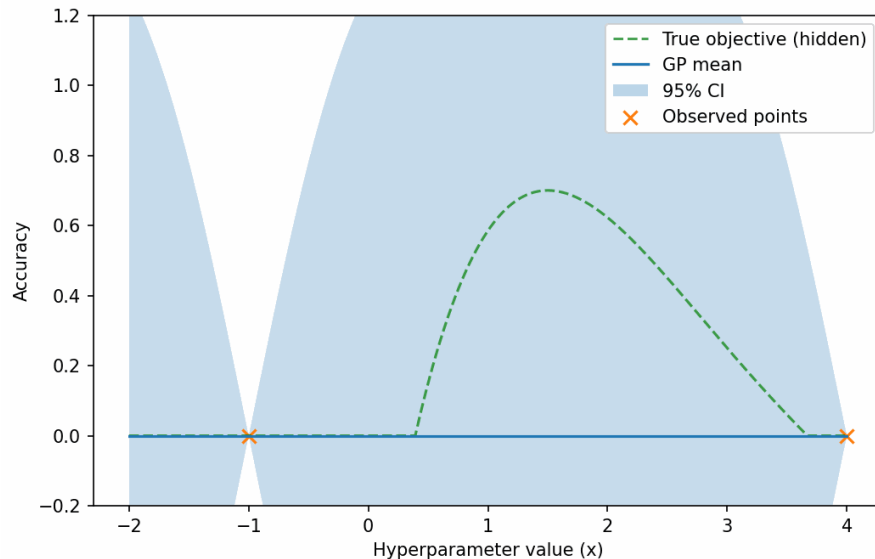
Acquisition function

- Trade off exploration vs. exploitation

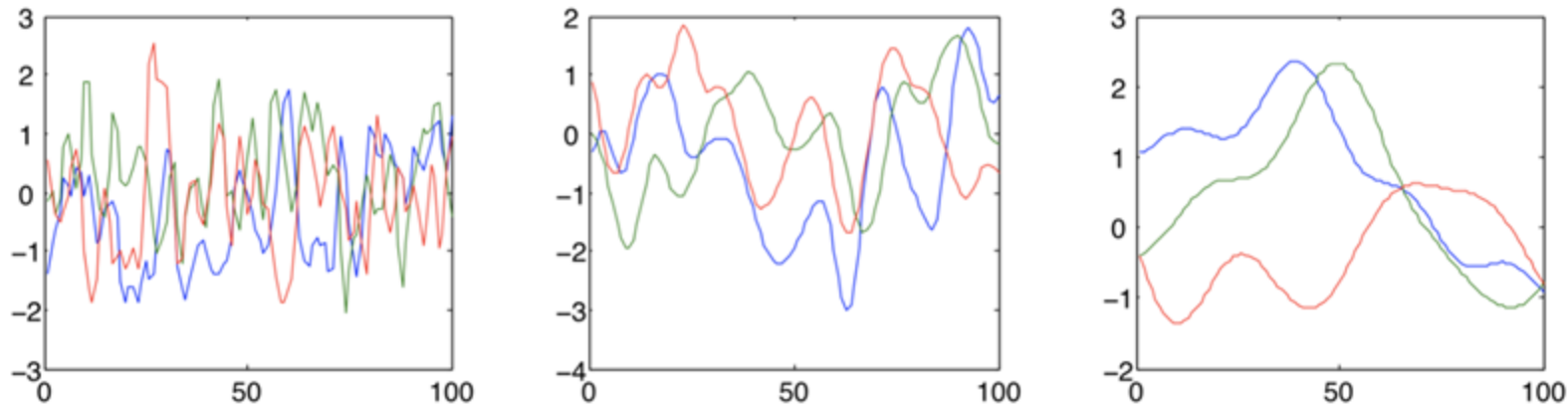
$$EI(\alpha) = E[\max(0, f_{\text{best}} - f(\alpha))]$$

Bayesian optimization:

```
for i in {0, ..., n}:  
    // use GP to compute EI  
    select  $\alpha^* = \max_{\alpha} EI(\alpha)$   
    compute val_loss of  $\alpha^*$ 
```



BO has its own hyperparameters!



[Frazier, 2022]

Assumption on the smoothness of the function $f(\alpha) = \text{val_loss}$
(without this assumption, convergence is slow)

[Berkenkamp, Schoellig, Krause JMLR 2019]

BO libraries: [Dragonfly: Kandasamy et al., JMLR 2020],
[SMAC3: Lindauer et al. JMLR 2022]

Bandit-based approaches

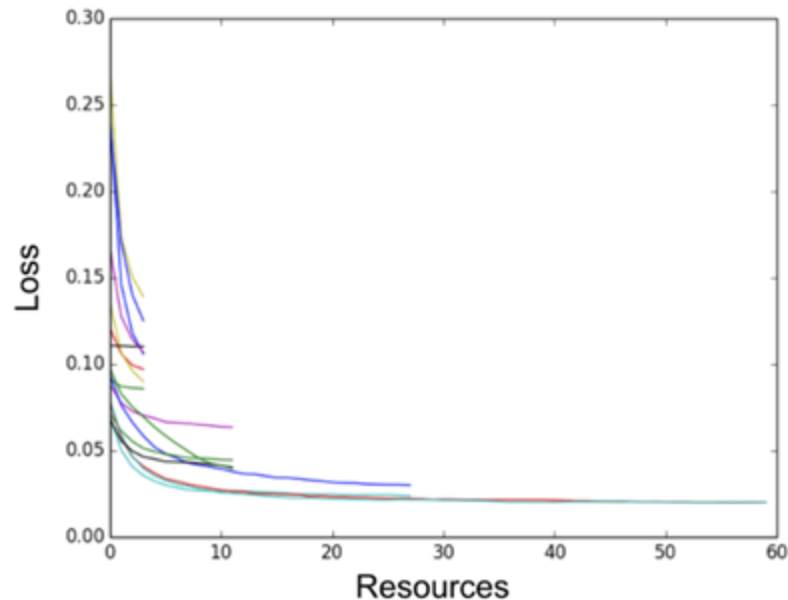
Random search with adaptive early stopping

Each arm has a noisy non-stationary reward that eventually converges to a limiting value

1. Successive halving

```
Given sets of hyperparameters  $A$   
for  $i$  in  $\{0, \dots, 3\}$ :  
    run( $\alpha$ ,  $10 * 2^i$ ),  $\alpha \in A$   
     $A := \text{top}_k(A, 16 * 2^{-i})$ 
```

2. **Hyperband**: multiple runs of successive halving, across different hyper-parameters



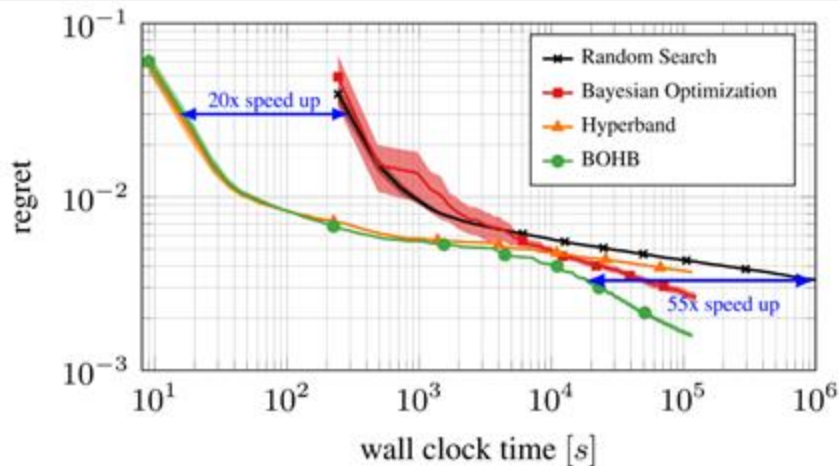
[Jamieson, Talwalkar (AISTATS 2016)]
[Li, Jamieson, DeSalvo, Rostamizadeh,
Talwalkar (JMLR 2018)]

Other approaches and speedups

BOHB (BO + Hyperband):

Run Hyperband, but replace the random selection of configurations at the beginning of each iteration by a model-based search

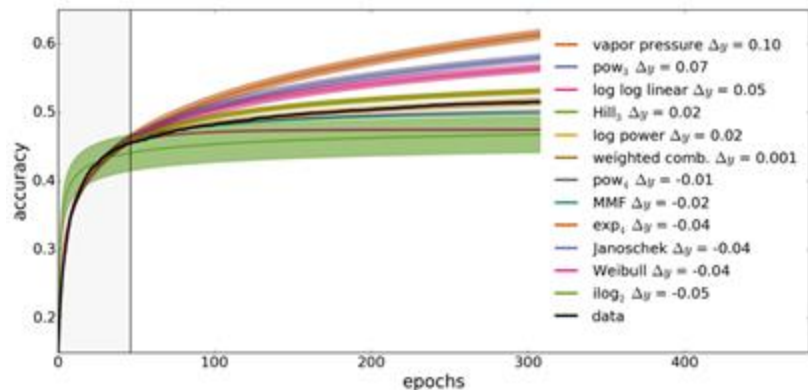
[Falkner, Klein, Hutter, ICML 2018]



Learning curve extrapolation:

Speed up HPO algorithms by extrapolating partial learning curves

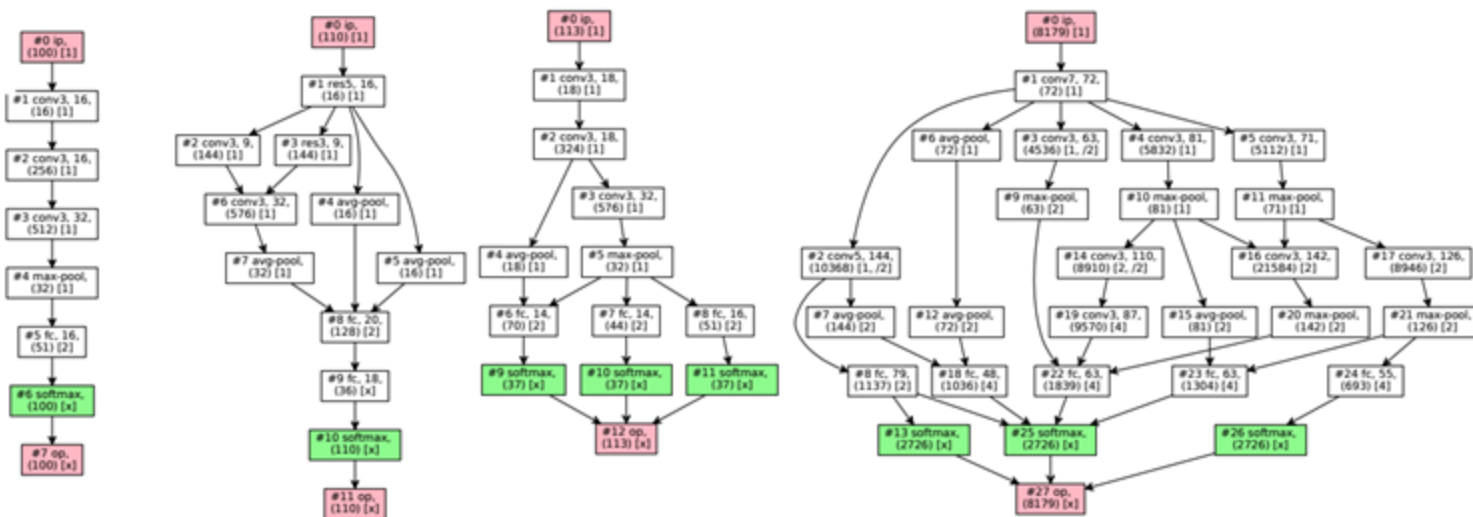
[Domhan, Springenberg, Hutter, IJCAI 2015]



Case study: Neural Architecture Search

Define the search space as a DAG with architecture components (e.g. conv_3x3, conv_5x5, pool, fc)

- The search space is a critical decision [Li, Talwalkar, UAI 2019]



[Kandasamy, Neiswanger, Schneider, Póczos, Xing, NeurIPS 2018]

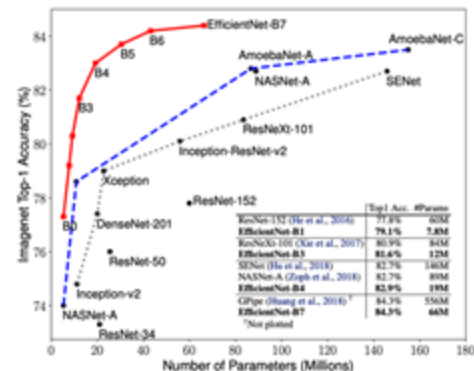
Case study: Neural Architecture Search (NAS)



[\[paperswithcode.com/sota/image-classification-on-imagenet, 2022\]](https://paperswithcode.com/sota/image-classification-on-imagenet, 2022)

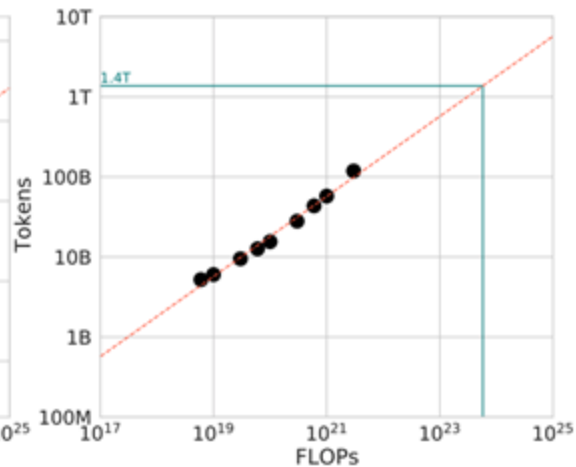
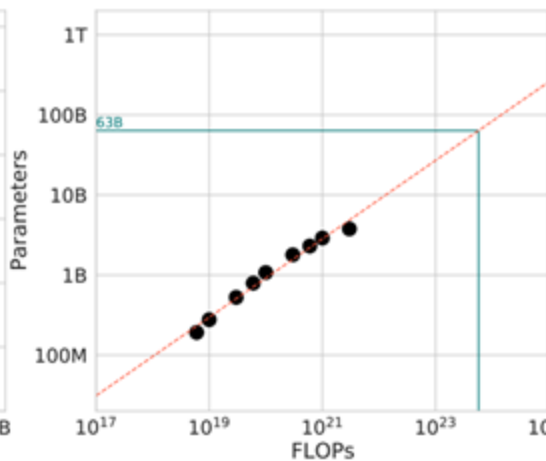
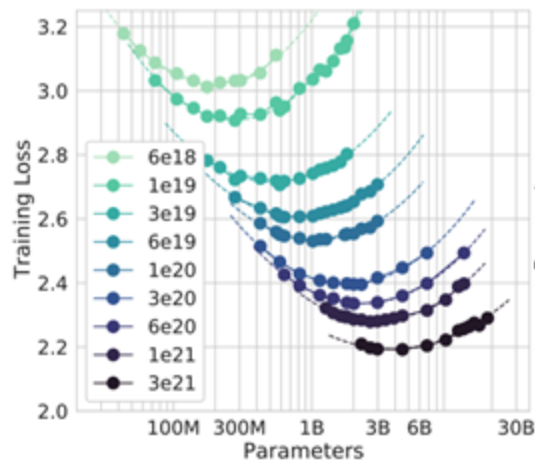
- NAS has been used to achieve SotA on imagenet seven times since 2017
- NAS has also been used to discover efficient architectures such as EfficientNet

[\[Tan, Le, ICML 2019\]](#)



Large Language Models: Scaling Laws

- **Scaling Laws for the ratio of tokens per parameter**
 - Isoflop analysis: given a fixed compute budget, sweep over model sizes
 - Fit an empirical trend across small models -> scale up



$$FLOPs \approx 6 * (\# \text{ params}) * (\# \text{ tokens})$$

[Kaplan et al, 2020], [Hoffmann et al, 2022]

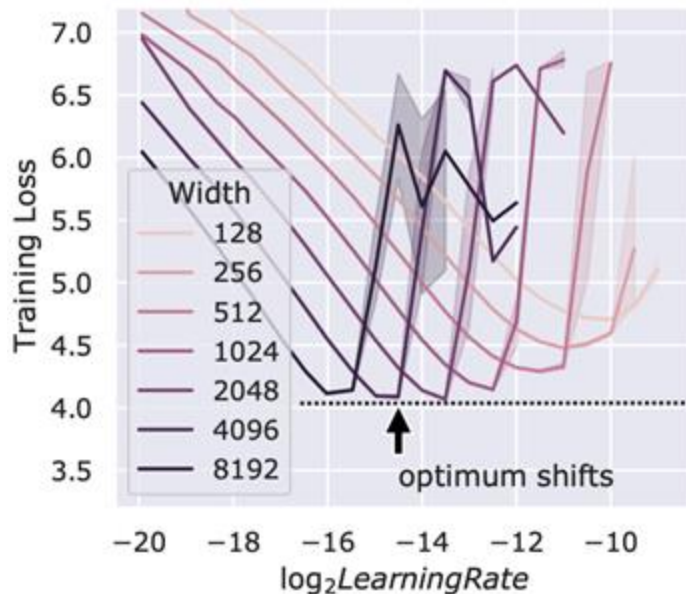
Large Language Models: selecting learning rate using muP

Maximum Update Parameterization (muP):

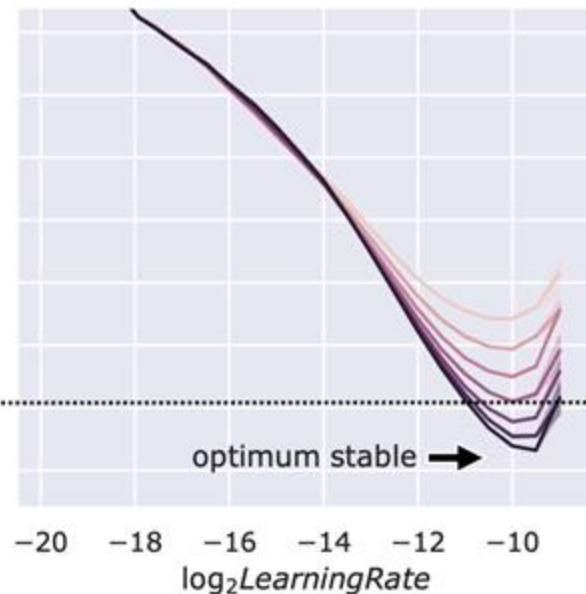
Parameterize the model such that the learning rate is same across all scales. Then tune LR once.

- Width-dependent LR
- Width-dependent weight initialization
- Width

Traditional LR scaling law:



muP LR scaling law:



Limitations of popular HPO approaches in practice

- Theoretical guarantees typically need strong assumptions
- Need to tune hyper-hyper-parameters
- Overtuning (val loss is a proxy for generalization) [Schneider, Bischl, Feurer, AutoML 2025]

Practitioners rely heavily on empirical findings!

All approaches are black-box!! (agnostic to the structure of the function)

Roadmap

- ❖ Introduction
- ❖ Major techniques used in practice
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ **Data-driven algorithm design**
 - Learning-theoretic guarantees
 - Distributional learning
 - Online learning
- ❖ Tuning core ML algorithms
 - Linear regression, decision trees
 - Semi-supervised learning, neural networks
- ❖ Conclusion

Machine Learning for Algorithm Design

Design and Analysis of Algorithms

Algorithm: (finite) sequence of precise step-by-step instructions to solve a well specified class of problems.

Algorithms for solving combinatorial problems. E.g.,


- Clustering: organize an input set of items into natural groups.
- Pricing: price a set of items to maximize revenue.
- Subset selection: output most valuable subset of items subject to capacity constraint.

Design and Analysis of Algorithms

Algorithms for solving combinatorial problems. E.g.,

- clustering, partitioning
- pricing, auction design
- subset selection

**Classic
Approach**

- 
1. Algorithm **hand-designed**, stroke of genius.
 2. **Worst-case analysis**, one-problem instance.

Many problems typically hard in classic frameworks.

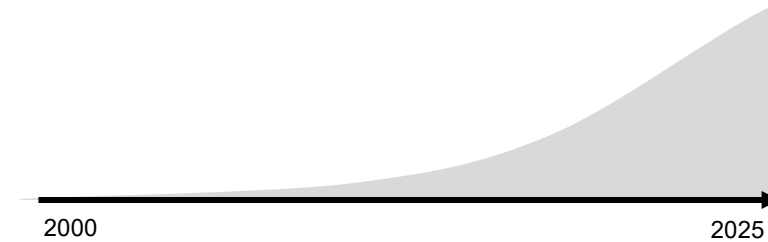
Machine Learning for Algorithm Design

Data-driven algo design: use learning & data for algo design.

- often repeatedly solve instances of the same algo problem.

Classic Work: largely empirical

AI, Computational Biology, Game Theory



[Horvitz-Ruan-Gomes-Kautz-Selman-Chickering, 2001] [DeBlasio-Kececioglu, 2018]

[Xu-Hutter-Hoos-LeytonBrown, 2008] [Likhodedov and Sandholm, 2004]

Recent Work: Data driven algos with **provable guarantees.**

“Data-driven algorithm design”, M.F. Balcan, chapter in “Beyond the Worst-Case Analysis of Algorithms book, 2020.

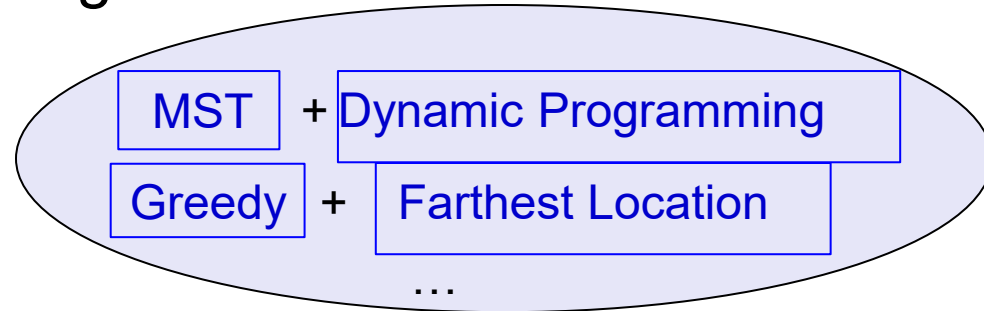
Interesting tools, with implications to Hyperparameter Tuning.

Algorithm Design as Distributional Learning

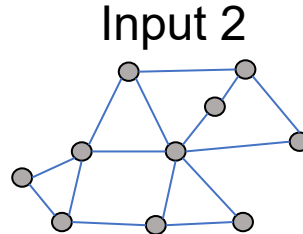
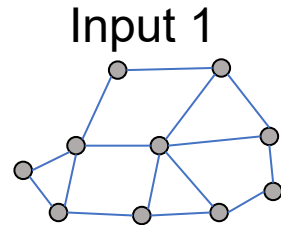
Data-driven algo design: directly learn an algorithm (from a parametric family of algos) that does well on instances from a given domain.

Large family F of algorithms

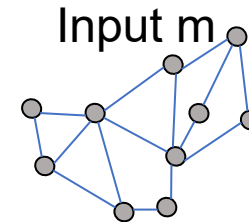
Sample of typical inputs



Clustering:



...



Pricing:

Input 1		
$v_1(C)$...	$v_n(C)$
$v_1(M)$		$v_n(M)$
$v_1(C \& M)$		$v_n(C \& M)$

...

Input m		
$v_1(C)$...	$v_n(C)$
$v_1(M)$		$v_n(M)$
$v_1(C \& M)$		$v_n(C \& M)$

Knapsack:

$(v_1 s_1), \dots, (v_n s_n), C$

...

$(v_1 s_1), \dots, (v_n s_n), C$

Algorithm Design as Distributional Learning

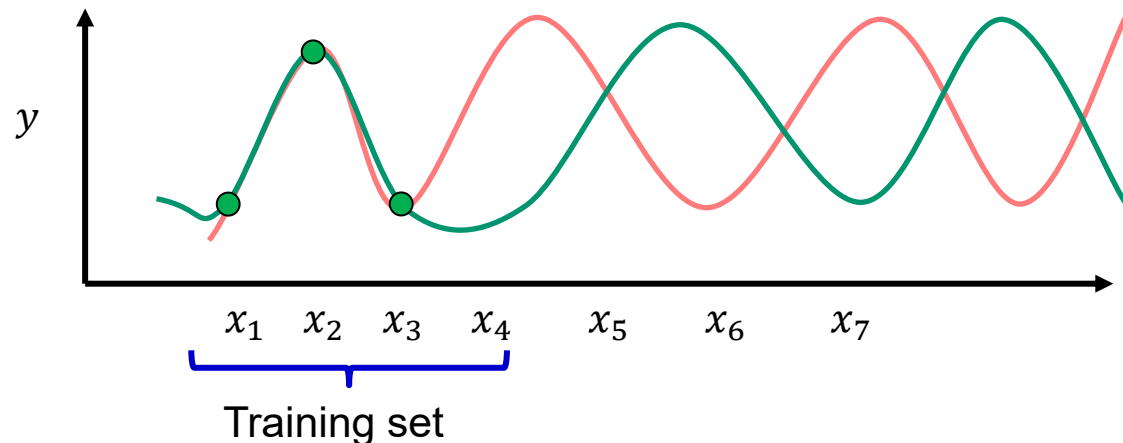
Sample Complexity: How large should training set be to guarantee that algos that do well over training set do well on new instances?

Tools from statistical learning theory

$m = O(\dim(\mathbf{F}) / \epsilon^2)$ instances suffice for uniform convergence.

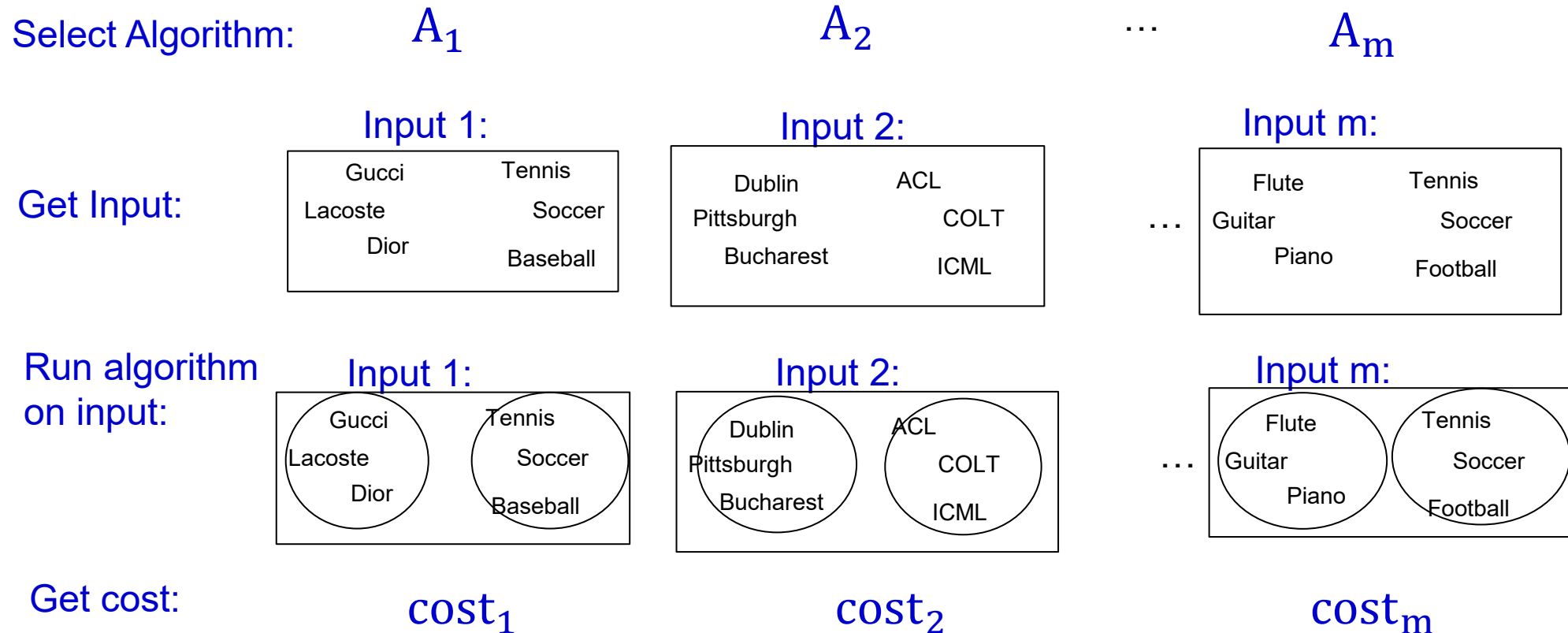
$\dim(\mathbf{F})$ (e.g. pseudo-dimension): ability of fns in \mathbf{F} to fit complex patterns

Overfitting



Online Algorithm Selection

Online alg. selection: instances arrive online, one by one

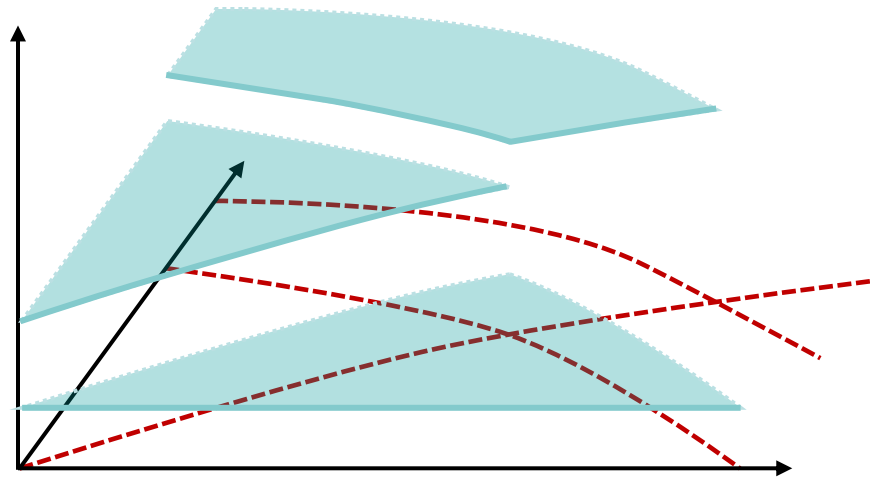


Guarantee: no regret - our cumulative performance comparable to performance of best parameter setting (algorithm) in hindsight.

Provable Data-Driven Algo Design, Challenges

Learnability of more complex objects.

Key Challenge: much more volatile losses.



Recent work: case studies and general principles.

Key new techniques: structure of dual function classes.

Data-driven algorithm design: Problem Setup

[Gupta-Roughgarden, ITCS'16 & SICOMP'17] [Balcan, book chapter, 2020].

- Fix an algorithmic pb (e.g., subset selection or clustering).
- Let Π be the set of problem instances for this problem.

Let Alg be a family of algos, parameterized by set $P \subseteq \mathbb{R}^d$; A_α the algo in Alg parametrized by $\alpha \in P$.

- Fix a utility function $u: \Pi \times P \rightarrow [0, H]$ where $u(I, \alpha)$ measures the performance of algo A_α on problem instance I .
 - $u_\alpha: \Pi \rightarrow [0, H]$ induced by A_α , where $u_\alpha(I) = u(I, \alpha)$.

Data-driven algorithm design: Problem Setup

[Gupta-Roughgarden, ITCS'16 & SICOMP'17] [Balcan, book chapter, 2020].

- Specific domain: unknown input distribution \mathcal{D} over Π .
- Learning algo uses m i.i.d. samples $I_1, I_2, \dots, I_m \sim \mathcal{D}$ to find an algo $A_\alpha \in \text{Alg}$ for future inputs from \mathcal{D} .

(can measure $u_\alpha(I)$ of each algo $A_\alpha \in \text{Alg}$ on each input I_i)

- Goal: output an algo of Alg that performs almost as well as the optimal algorithm $A_{\alpha^*} \in \text{Alg}$ for \mathcal{D} that maximizes

$$E_{I \sim \mathcal{D}}[u_\alpha(I)] \text{ over } A_\alpha \in \text{Alg}.$$

- Typical approach: pick \hat{A} that does well over the sample.

Sample Complexity: How large should training set be to guarantee that algos that do well over training set do well on new instances?

Data-driven algorithm design. Example: Knapsack Problem

[Gupta-Roughgarden, ITCS'16 & SICOMP'17]

Input: An instance I consists of n items (each item i has a value v_i and a size s_i), and knapsack capacity C .

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

Alg : greedy algos parametrized by $P = R$.

For $\alpha \in P$, algo A_α :

- Set score of item i to be v_i/s_i^α .
- In decreasing order of score, we add each item to the knapsack if there is enough capacity left.

$u(I, \alpha)$ = value of items chosen by the algo param. by α on I .

Data-driven algorithm design. Example: Partitioning Problems

[Balcan-Nagarajan-Vitercik-White, COLT'17] [Balcan-Dick-Lang, ICLR'20]

Input: set of objects S , d .

Output: centers $\{c_1, c_2, \dots, c_k\}$

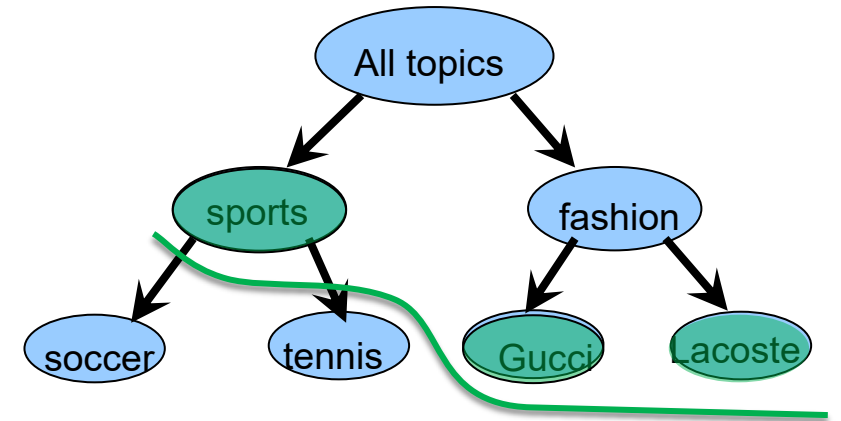
k-means clustering: $\min \sum_p \min_i d^2(p, c_i)$

k-center (facility location): $\min \max \text{radius}$.

Alg : greedy algos parametrized by $P = R$.

1. Greedy linkage-based, get hierarchy.

$$\text{E.g., } \text{dist}_\alpha(A, B) = (1 - \alpha)\text{SL} + \alpha \max_{x \in A, x' \in B} d(x, x')$$



2. Fixed algo (e.g., DP or last k-merges) to select a good pruning.

$u(I, \alpha)$ = clustering objective chosen by the algo α on I .

Uniform Convergence

Uniform convergence: for any algo in Alg , average performance over samples “close” to its expected performance.

- Imply that $\hat{\mathbf{A}}$ that does best over the sample has high expected performance.

Learning theoretic notion of dimension, e.g., pseudo-dimension

Theorem

$m = O(\text{dim}(\mathbf{F})/\epsilon^2)$ suffices so that for any distribution D over Π , with prob. at least $1 - \delta$ over the draw $\{I_1, \dots, I_m\} \sim D$, for all algos $A_\alpha \in \text{Alg}$,

$$\left| \mathbb{E}_{I \sim D}[u_\alpha(I)] - \frac{1}{m} \sum_{i=1}^m u_\alpha(I_i) \right| \leq \epsilon$$

Algorithm Design as Distributional Learning

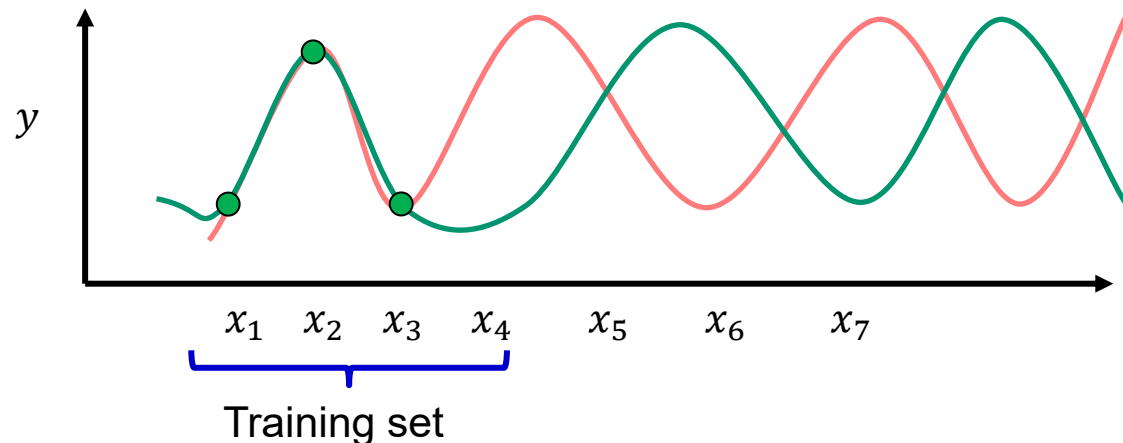
Sample Complexity: How large should training set be to guarantee that algos that do well over training set do well on new instances?

Tools from statistical learning theory

$m = O(\dim(\mathbf{F}) / \epsilon^2)$ instances suffice for uniform convergence.

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

Overfitting



General Sample Complexity via Dual Classes

Theorem (informally) [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

Technique for analyzing $\dim(\{u_\alpha(\cdot) : \text{param } \alpha\})$ that takes advantage of the structure of **dual class** $\{u_I(\cdot) : \text{instances } I\}$.

Given $u: \Pi \times P \rightarrow [0, H]$, $u(I, \alpha)$ = performance of A_α on I .

- $u_\alpha: \Pi \rightarrow [0, H]$ induced by A_α , $u_\alpha(I) = u(I, \alpha)$.
- $u_I: P \rightarrow [0, H]$ induced by I , $u_I(\alpha) = u(I, \alpha)$.

$\{u_I: \text{instances } I\}$ dual class for $\{u_\alpha(\cdot) : \text{param } \alpha\}$

General Sample Complexity via Dual Classes

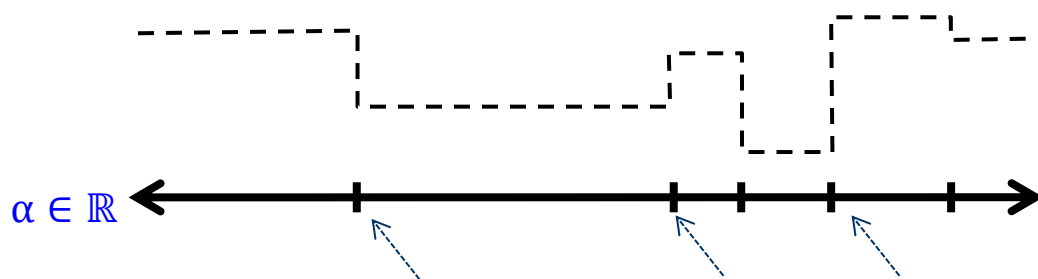
Theorem (informally) [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

Technique for analyzing $\dim(\{u_\alpha(\cdot) : \text{param } \alpha\})$ that takes advantage of the structure of **dual class** $\{u_I(\cdot) : \text{instances } I\}$.

Key motivation: can often show u_I is structured.

Example: knapsack, greedy family Alg, u_I piece-wise linear:

[Gupta-Roughgarden, ITCS'16 & SICOMP'17]



Critical points of the form $\frac{\log\left(\frac{v_i}{v_j}\right)}{\log\left(\frac{s_i}{s_j}\right)}$, so $O(n^2)$ pieces.

General Sample Complexity via Dual Classes

Theorem (informally) [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

Technique for analyzing $\dim(\{u_\alpha(\cdot) : \text{param } \alpha\})$ that takes advantage of the structure of **dual class** $\{u_I(\cdot) : \text{instances } I\}$.

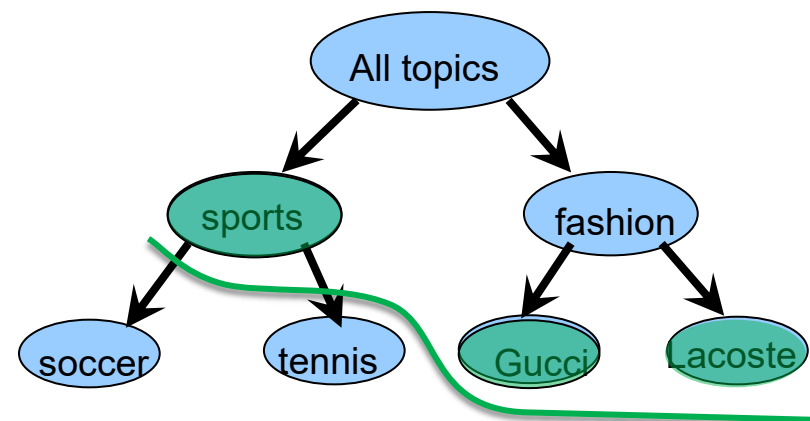
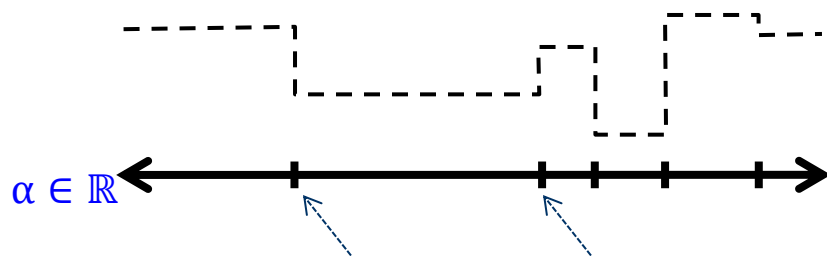
Example: clustering, parametrized-linkage, u_I piece-wise linear

[Balcan-Nagarajan-Vitercik-White, COLT'17] [Balcan-Dick-Lang, ICLR'20]

1. Greedy linkage-based, get hierarchy.

$$\text{E.g., } \text{dist}_\alpha(A, B) = (1 - \alpha)\text{SL} + \alpha \max_{x \in A, x' \in B} d(x, x')$$

2. Fixed algo to select a good pruning.



General Sample Complexity via Dual Classes

Theorem (informally) [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

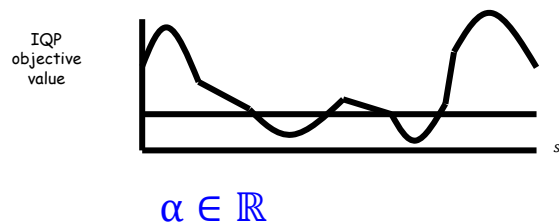
Technique for analyzing $\dim(\{u_\alpha(\cdot) : \text{param } \alpha\})$ that takes advantage of the structure of **dual class** $\{u_I(\cdot) : \text{instances } I\}$.

Key motivation: can often show u_I is structured.

Partitioning Pbs via IQPs

SDP + s-linear rounding

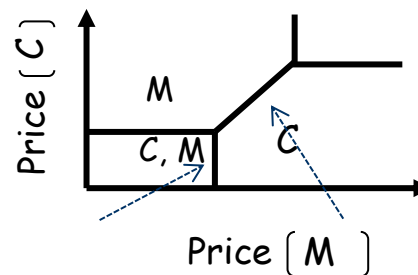
[Balcan-Nagarajan-Vitercik-White, COLT 2017]



Posted Pricing, Two-Part Tariffs, Parametrized VCG auctions, etc.

[Balcan-Sandholm-Vitercik, EC'18]

[Balcan-Beyhaghi, TMLR'24]



Decision boundary where the buyer prefers one bundle over the other, is a hyperplane.

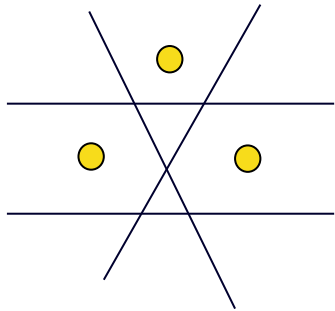
VC-dimension (for binary valued classes)

VC-dimension of a function class H is the cardinality of the largest set S that can be labeled in all possible ways $2^{|S|}$ by H .

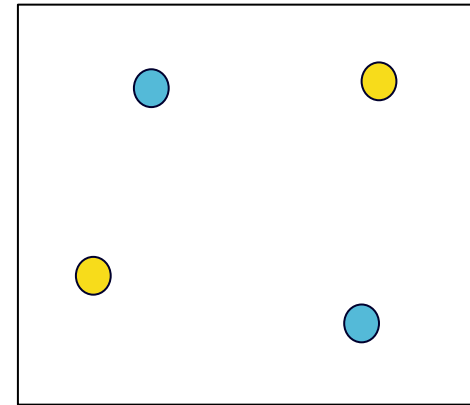
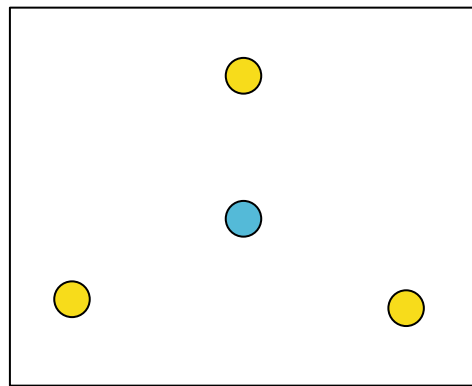
[If arbitrarily large finite sets can be shattered by H , then $\text{VCdim}(H) = \infty$]

E.g., H = linear separators in \mathbb{R}^2 : $\text{VCdim}(H) = 3$

$\text{VCdim}(H) \geq 3$



$\text{VCdim}(H) < 4$



E.g., H = linear separators in \mathbb{R}^d : $\text{VCdim}(H) = d+1$

Pseudo-dimension (for real valued classes)

The **pseudo-dimension** [Pollard 1984] of a function class F is the cardinality of the largest set $S = \{x_1, \dots, x_m\}$ and thresholds y_1, \dots, y_m s.t. all 2^m above/below patterns can be achieved by functions $f \in F$.

- E.g., $m = 2$, there should exist $f_1 \in F$ s.t. $f_1(x_1) < y_1, f_1(x_2) < y_2$; $f_2 \in F$ s.t. $f_2(x_1) > y_1, f_2(x_2) < y_2$; $f_3 \in F$ s.t. $f_3(x_1) < y_1, f_3(x_2) > y_2$, and $f_4 \in F$ s.t. $f_4(x_1) > y_1, f_4(x_2) > y_2$

Equivalently, the **pseudo-dimension** of F is the VC dim of the class of “below-the-graph” indicator functions $\{B_f(x, y) = \text{sgn}(f(x) - y) : f \in F\}$

Pseudo-dimension, Uniform Convergence

The **pseudo-dimension** [Pollard 1984] of a function class \mathcal{F} is the cardinality of the largest set $S = \{x_1, \dots, x_m\}$ and thresholds y_1, \dots, y_m s.t. all 2^m above/below patterns can be achieved by functions $f \in \mathcal{F}$.

Uniform convergence guarantees [Pollard'84; Dudley '67]

For any $\delta \in (0,1)$ and any distribution \mathcal{D} over \mathcal{X} , with probability $1 - \delta$ over the draw $\{x_1, \dots, x_m\} \sim \mathcal{D}^m$, for all functions $f \in \mathcal{F}$,

$$\left| \mathbb{E}_{x \sim \mathcal{D}}[f(x)] - \frac{1}{m} \sum_{i=1}^m f(x_i) \right| = O \left(U \sqrt{\frac{\mathbf{Pdim}(\mathcal{F})}{m}} + U \sqrt{\frac{\log(1/\delta)}{m}} \right),$$

where U is the maximum $f(x)$ for any $f \in \mathcal{F}$ and x in the support of \mathcal{D} .

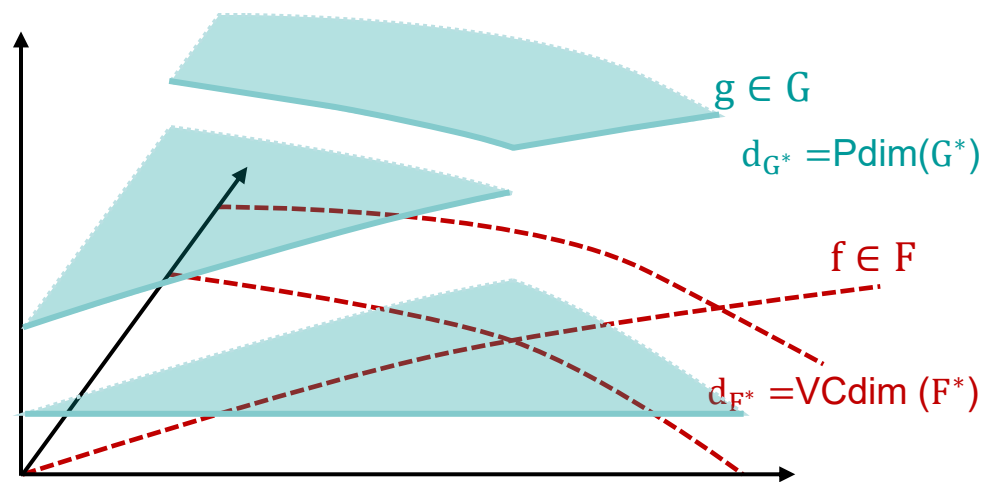
General Sample Complexity via Dual Classes

Theorem [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

Suppose for each $u_I(\alpha)$ there are $\leq N$ boundary fns $f_1, f_2, \dots \in F$ s. t within each region defined by them, $\exists g \in G$ s.t. $u_I(\alpha) = g(\alpha)$.

$$\text{Pdim}(\{u_\alpha(I)\}) = \tilde{O}((d_{F^*} + d_{G^*}) + d_{F^*} \log N)$$

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



General Sample Complexity via Dual Classes

Theorem [Balcan-DeBlasio-Kingsford-Dick-Sandholm-Vitercik, STOC 2021&JACM 2024]

Suppose for each $u_I(\alpha)$ there are $\leq N$ boundary fns $f_1, f_2, \dots \in F$ s. t within each region defined by them, $\exists g \in G$ s.t. $u_I(\alpha) = g(\alpha)$.

$$\text{Pdim}(\{u_\alpha(I)\}) = \tilde{O}((d_{F^*} + d_{G^*}) + d_{F^*} \log N)$$

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

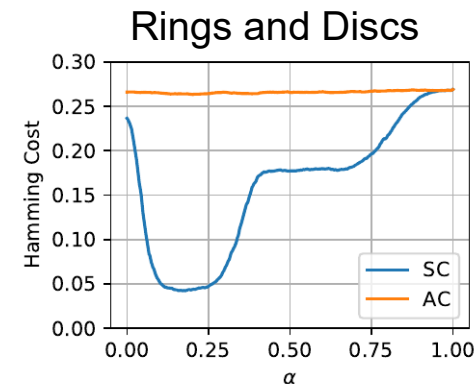
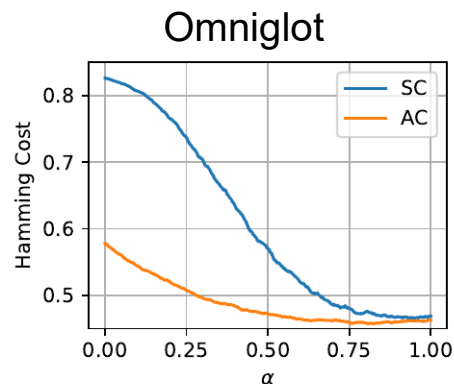
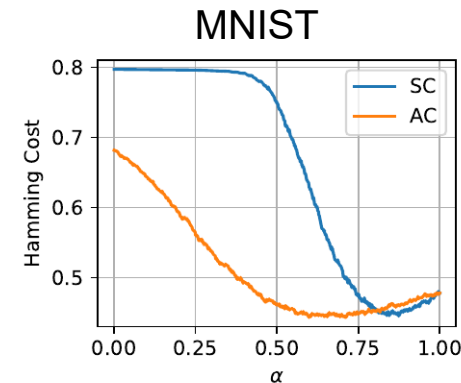
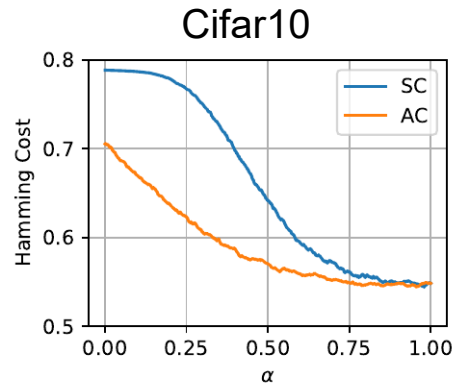
Proof insights:

- Fix D instances I_1, \dots, I_D and D thresholds z_1, \dots, z_D . Bound # sign patterns $(u_\alpha(I_1), \dots, u_\alpha(I_D))$ ranging over all α . Equivalently, $(u_{I_1}(\alpha), \dots, u_{I_D}(\alpha))$.
- Use VCdim of F^* , bound # of regions induced by $u_{I_1}(\alpha), \dots, u_{I_D}(\alpha) : (eND)^{d_{F^*}}$.
- On a region, exist g_{I_1}, \dots, g_{I_D} s.t., $(u_{I_1}(\alpha), \dots, u_{I_D}(\alpha)) = (g_{I_1}(\alpha), \dots, g_{I_D}(\alpha))$, which equals $(\alpha(g_{I_1}), \dots, \alpha(g_{I_D}))$. 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.

Different Algos Work in Different Settings

[Balcan-Dick-Lang, ICLR'20]

- Optimal parameters vary across different distributions.
- Choosing the parameter can give large improvements to loss; improvements over SL and CL by interpolating between them.

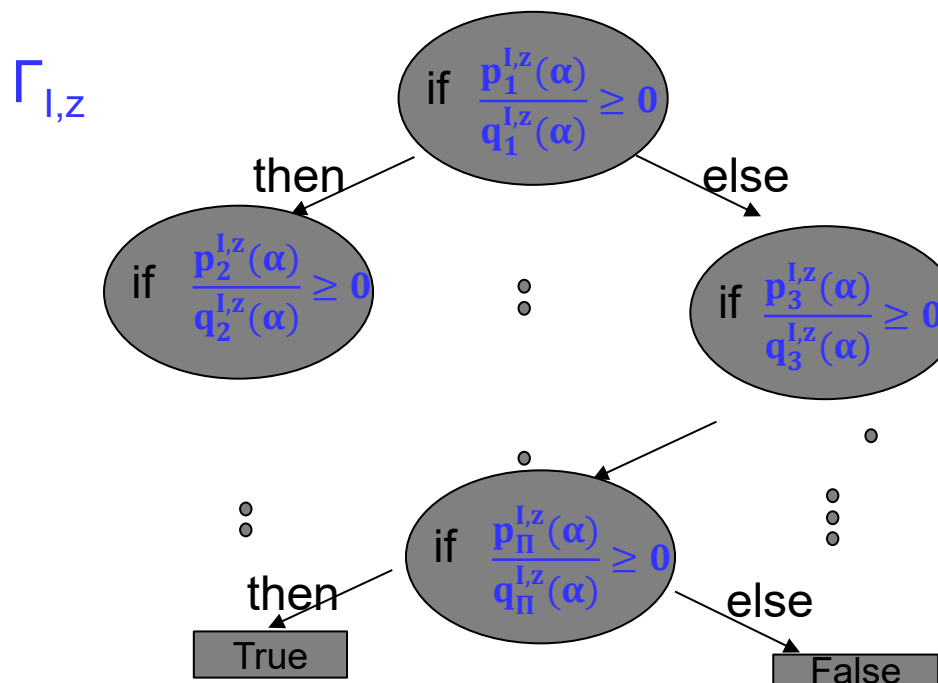


Goldberg-Jerrum (GJ) Framework

Theorem [Bartlett, Indyk, Wagner. COLT'22] Assume $\alpha \in \mathbb{R}^n$, i.e. each $A_\alpha \in \text{Alg}$ has n real param. For any l and z , there is a GJ procedure $\Gamma_{l,z}$ that determines for all α if $u_l(\alpha) \geq z$ by evaluating Π distinct predicates (ratios of polys) with max. degree Δ . Then:

$$\text{Pdim}(\{u_\alpha(l)\}) = O(n \log(\Delta \Pi))$$

GJ (95) Procedure



Goldberg-Jerrum (GJ) Framework

Theorem [Bartlett, Indyk, Wagner. COLT'22] Assume $\alpha \in \mathbb{R}^n$, i.e. each $A_\alpha \in \text{Alg}$ has n real param. For any I and z , there is a GJ procedure $\Gamma_{I,z}$ that determines for all α if $u_I(\alpha) \geq z$ by evaluating Π distinct predicates (ratios of polys) with max. degree Δ . Then:

$$\text{Pdim}(\{u_\alpha(I)\}) = O(n \log(\Delta \Pi))$$

Theorem [Balcan, Ngyuen, Sharma, TMLR'25] Assume $\alpha \in \mathbb{R}^n$, i.e. each $A_\alpha \in \text{Alg}$ has n real param. For any instance I and threshold z , there is a Pfaffian GJ algorithm $\Gamma_{I,z}$ that determines for all α if $u_I(\alpha) \geq z$ by evaluating Π distinct Pfaffian predicates with Pfaffian chain length q , degree Δ , and Pfaffian degree M .

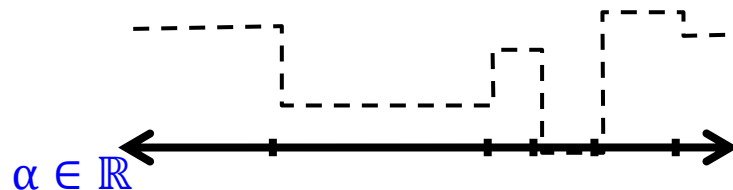
$$\text{Pdim}(\{u_\alpha(I)\}) = O(n^2 q^2 + n q \ln(\Delta + M) + n \ln \Pi)$$

Online Algorithm Selection

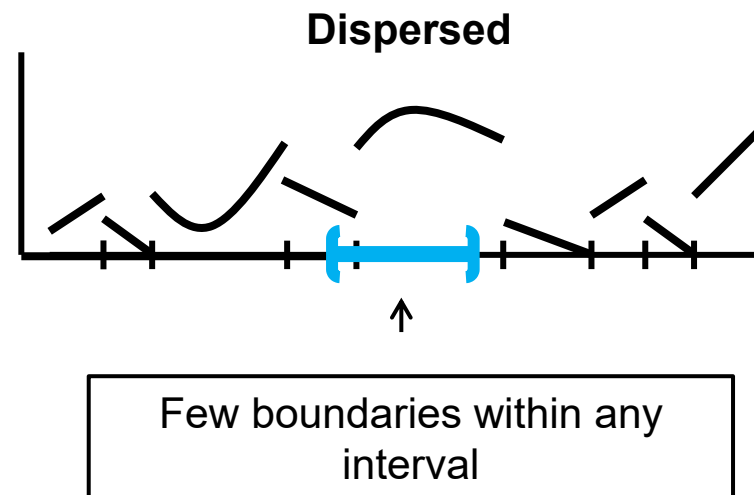
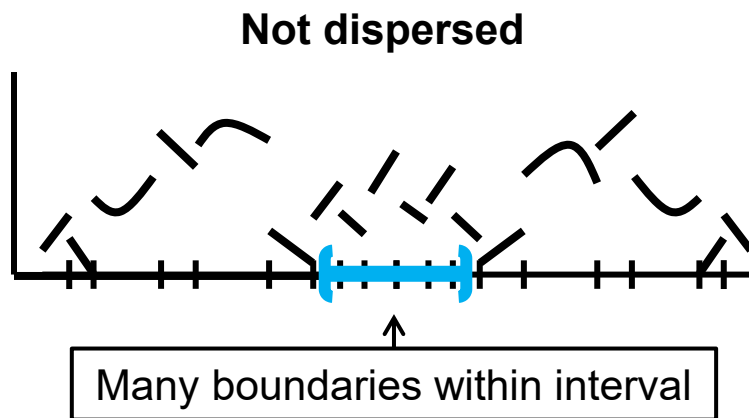
Instances arrive online, one by one. [Balcan-Dick-Vitercik, FOCS'18], [Balcan-Dick-Pedgen, UAI'20]

Guarantee: no regret - cumulative performance of learner comparable to performance of best algorithm from the family in hindsight.

Challenge: loss functions volatile.



Our contribution: identify general properties (piecewise Lipschitz fns with dispersed discontinuities) sufficient for no regret guarantees.



Dispersion, Sufficient Condition for No-Regret

[Balcan-Dick-Vitercik, FOCS'18], [Balcan-Dick-Pedgen, UAI'20]

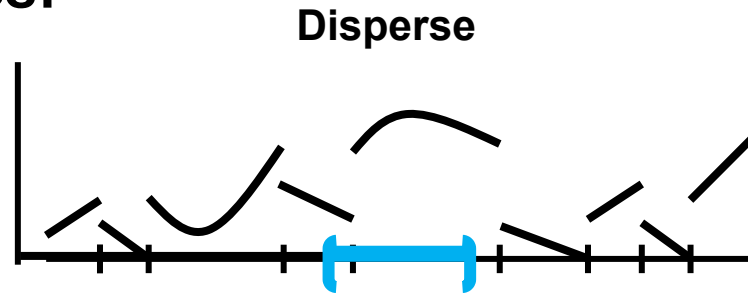
Full info: exponentially weighted forecaster [Cesa-Bianchi-Lugosi 2006]

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

- Sample α_t from distr. p_t : $p_t(\alpha) \propto \exp\left(\lambda \sum_{s=1}^{t-1} u_s(\alpha)\right)$

density of α exponential in its performance so far

No Regret Guarantees:



Disperse fns, regret $\tilde{O}(\sqrt{Td} \text{ fnc of problem})$.

Key Questions

- What are interesting tunable families of algos?
- How do we tune algos, to achieve best performance for a given domain, with provable guarantees?
- **Data driven algo design as distributional learning**

Suffices to show that **dual class** $\{u_I(\cdot): \text{instances } I\}$ is structured.

- **Data driven algo design as online learning**

[Balcan-Dick-Vitercik, FOCS'18], [Balcan-Dick-Pegden, UAI'20], [Balcan-Sharma, NeurIPS 2021]

Disperse fns, regret $\tilde{O}(\sqrt{Td} \text{ fnc of pb})$.

Who designs good machine learning algorithms?

- Often hand-designed, with tunable parameters.
- How do we tune machine learning algorithms with provable guarantees?
- What are interesting tunable families of algos?

Roadmap

- ❖ Introduction
- ❖ Major techniques used in practice
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ Data-driven algorithm design
 - Learning-theoretic guarantees
 - Distributional learning
 - Online learning
- ❖ **Tuning core ML algorithms**
 - **Linear regression, decision trees**
 - Semi-supervised learning, neural networks
- ❖ Future research

Regularized linear regression

Given instance $(X, y) \in \mathbb{R}^{m \times p} \times \mathbb{R}^m$

m : number of examples

X_{i*} : regressor, p features

y_i : regressand or dependent variable

Regularized linear regression

L1 (LASSO):

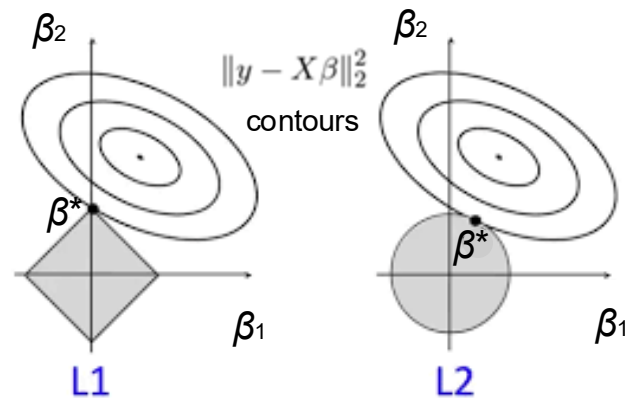
$$\beta_{L1} = \operatorname{argmin}_{\beta \in \mathbb{R}^{m \times p}} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

[Tibshirani 96]

L2 (Ridge):

$$\beta_{L2} = \operatorname{argmin}_{\beta \in \mathbb{R}^{m \times p}} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

[Hoerl & Kennard 70]



bad if $m < p$

not sparse

Regularized linear regression

Given dataset

$$(X, y) \in \mathbb{R}^{m \times p} \times \mathbb{R}^m$$

m : number of examples

X_{i*} : regressor, p features

y_i : regressand or dependent variable

Regularized linear regression

Elastic net:

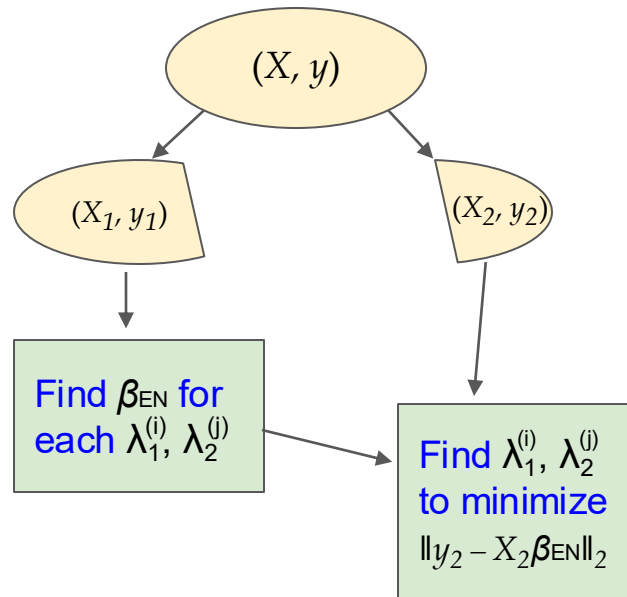
$$\beta_{\text{EN}} = \operatorname{argmin}_{\beta \in \mathbb{R}^{m \times p}} \|y - X\beta\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$$

[Zou & Hastie 05]

Objective
selection

Q: How to set hyperparameter $\alpha = (\lambda_1, \lambda_2)$ for given dataset?

A: Use a hold out dataset, a grid of parameter values, minimize sq-error on held-out set

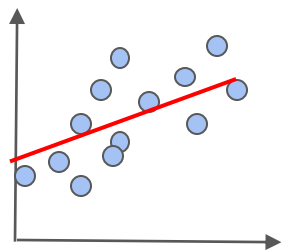


Multiple instances

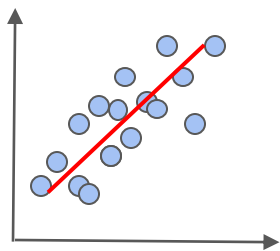
n instances of the regression problem

$$I_i = (X^{(i)}, y^{(i)}, X_v^{(i)}, y_v^{(i)}) \in \mathbb{R}^{m_i \times p} \times \mathbb{R}^{m_i} \times \mathbb{R}^{m'_i \times p} \times \mathbb{R}^{m'_i}$$

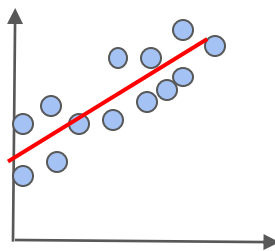
$$m_i, m'_i \leq m, p_i \leq p.$$



I_1



I_2



I_3

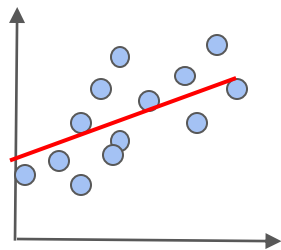


Multiple instances

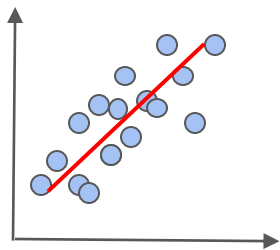
n instances of the regression problem

$$I_i = (X^{(i)}, y^{(i)}, X_v^{(i)}, y_v^{(i)}) \in \mathbb{R}^{m_i \times p} \times \mathbb{R}^{m_i} \times \mathbb{R}^{m'_i \times p} \times \mathbb{R}^{m'_i}$$

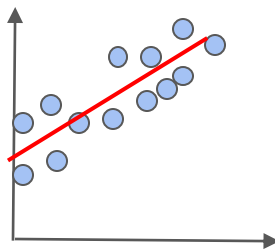
$$m_i, m'_i \leq m, p_i \leq p.$$



I_1



I_2



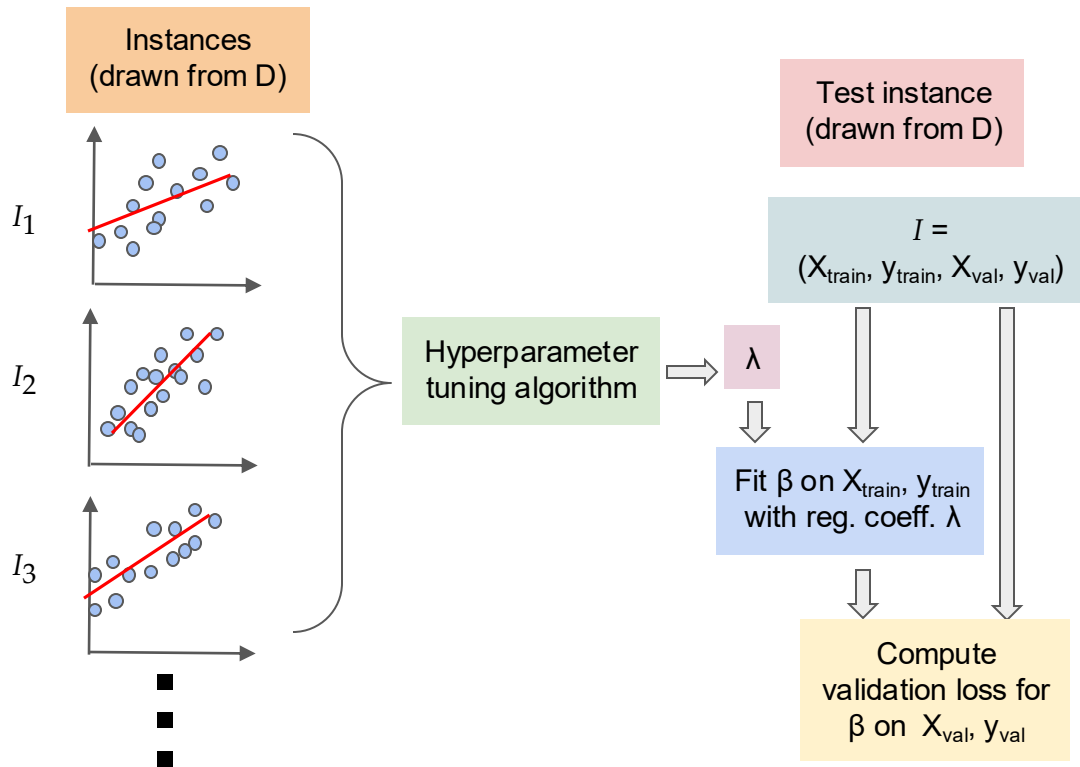
I_3

$$\text{loss}(P) = \frac{1}{m'} \|y_{\text{val}} - X_{\text{val}} \hat{\beta}\|^2$$

fit on $(X_{\text{train}}, y_{\text{train}})$
using predicted $(\hat{\lambda}_1, \hat{\lambda}_2)$



Multiple instances



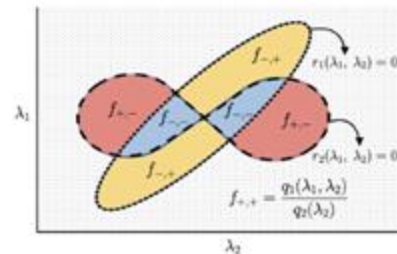
Elastic Net [Balcan, Khodak, Sharma and Talwalkar, NeurIPS'22, Balcan, Nguyen, Sharma, NeurIPS'23]

Example application: Tuning Elastic Net coefficients.

$$\min_w \|Xw - y\|^2 + \lambda \|w\|^2 + \lambda' \|w\|_1$$

Input: Training data X, y and validation data X', y' .

Goal: Tune λ, λ' to minimize *dual* validation loss + L0 terms (AIC/BIC).



Lemma: The dual validation loss is piecewise decomposable in the λ, λ' space with

- at most $d3^d$ algebraic boundaries of degree at most d ,
- at most 3^d distinct piece functions, each a rational function with degree at most $2d$.

Challenge: sharp transition boundaries, due to L0 terms in AIC/BIC validation loss.

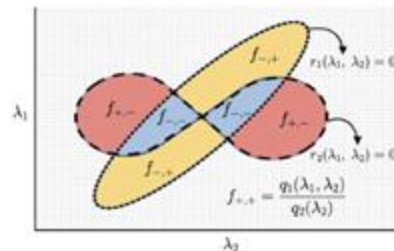
Elastic Net [Balcan, Khodak, Sharma and Talwalkar, NeurIPS'22, Balcan, Nguyen, Sharma, NeurIPS'23]

Lemma: The dual validation loss is piecewise decomposable in the λ, λ' space with

- at most $d3^d$ algebraic boundaries of degree at most d ,
- at most 3^d distinct piece functions, each a rational function with degree at most $2d$.

Proof sketch:

1. Elastic net is equivalent to a lasso for some modified datasets X', y' that depend on the ridge coefficient.
2. Lasso has a piecewise linear solution in terms of the L1 penalty with known conditions for critical points.
3. 1+2 gives polynomial boundary functions and rational piece functions in terms of both the coefficients.



Theorem: Sample complexity of tuning λ, λ' is $O(d/\varepsilon^2)$.

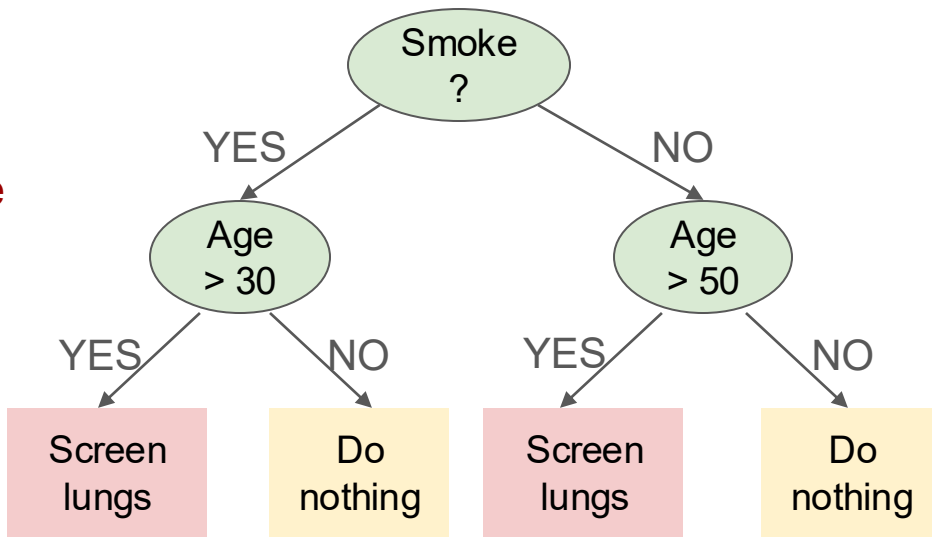
Decision Trees

Trees for classification:

- Each internal node \Leftrightarrow Splitting rule
- Each leaf node \Leftrightarrow Single Class

Interpretable ML models

- axis-parallel decision boundaries
- Neural nets are hard to interpret



Hard to learn optimal trees, but several useful heuristics!

Learning optimal decision trees is hard!

Hardness of DT learning

- NP-complete. [Laurent and Rivest (1976)]
- *Superconstant Inapproximability of Decision Tree Learning.*

[Koch et al. COLT 2024] [Koch and Strassle FOCS 2023, FOCS 2024]

Faster optimal decision trees (speed up the exp time branch-and-bound algorithm)

- [Hu et al. NeurIPS 2019]
- [McTavish et al. AAAI 2022]
- [Babbar et al. ICML 2025] (combines greedy with branch-and-bound)

Alternative: data-driven formulation,

instances I : labeled datasets, **utility** $u(I, \alpha)$: avg. acc. on instance I

Splitting criterion



Top-down decision tree learning

Inputs: Node function class \mathcal{F} , tree size \mathbf{t} ,
splitting criterion \mathbf{G}

- Start with leaf node

Splitting criterion



Top-down decision tree learning

Inputs: Node function class \mathcal{F} , tree size \mathbf{t} ,
splitting criterion \mathbf{G}

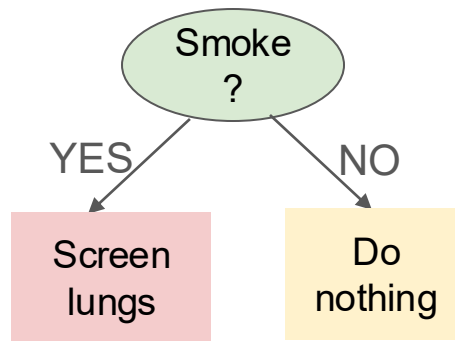
- Start with leaf node
- While at most \mathbf{t} leaf nodes
 - Split leaf node \mathbf{l} using node function \mathbf{f} which maximizes “splitting criterion”

Splitting criterion

Top-down decision tree learning

Inputs: Node function class \mathcal{F} , tree size t ,
splitting criterion G

- Start with leaf node
- While at most t leaf nodes
 - Split leaf node l using node function f which maximizes “splitting criterion”



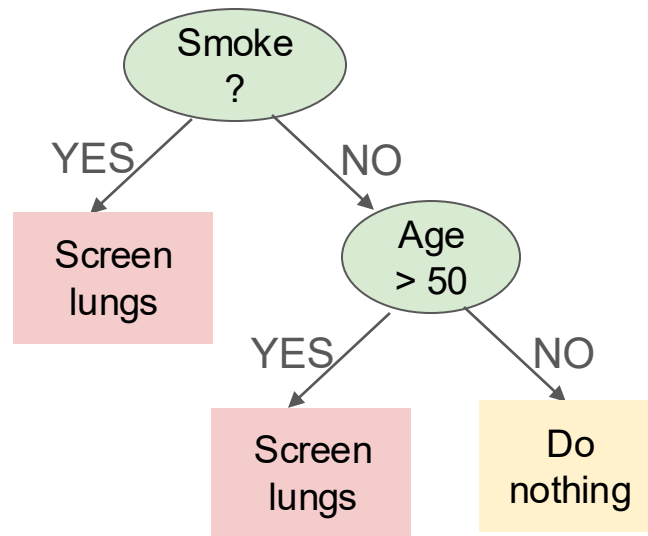
$$\mathcal{F} = \{\text{Smoke}, \text{Age} > 30, \text{Age} > 50\}$$

Splitting criterion

Top-down decision tree learning

Inputs: Node function class \mathcal{F} , tree size t ,
splitting criterion G

- Start with leaf node
- While at most t leaf nodes
 - Split leaf node l using node function f which maximizes “splitting criterion”



Key decision: Which node to split next
and how? \Rightarrow splitting criterion

$\mathcal{F} = \{\text{Smoke}, \text{Age} > 30, \text{Age} > 50\}$

Splitting criterion (a greedy approach)

$$G(T) = \sum_{l \in \text{leaves}(T)} w(l)g(\{p_1(l), \dots, p_c(l)\})$$

Top-down decision tree learning

Inputs: Node function class \mathcal{F} , tree size \mathbf{t} ,
splitting criterion \mathbf{G}

$w(l)$: number of datapoints that map to leaf l
 $p_i(l)$: fraction of them labeled i

\mathcal{F} : binary functions for labeling internal nodes: features $\rightarrow \{\text{left}, \text{right}\}$

\mathbf{G} : uses how the current tree partitions the data into different classes to determine which node to split next and using with function in \mathcal{F}

Overall algorithm: Greedy approach to growing a decision tree top-down (from the root to leaves by repeatedly replacing an existing leaf with an internal node based on a “splitting criterion”).

Algorithm family: interpolation of popular splitting criteria.

Splitting criterion



DecisionTreeClassifier

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini',
splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None,
ccp_alpha=0.0, monotonic_cst=None)
```

[\[source\]](#)

A decision tree classifier.

Read more in the [User Guide](#).

Parameters:

criterion : {"gini", "entropy", "log_loss"}, default="gini"

Splitting criterion [Balcan and Sharma (UAI 2024)]

Empirical research suggests different criteria work best on different data [Mingers 1989]

- Entropy criterion (CART) $g(\{p_1(l), \dots, p_c(l)\}) = - \sum_{i=1}^c p_i \log p_i$
- Gini impurity (ID3) $g(\{p_1(l), \dots, p_c(l)\}) = \sum_i p_i(1 - p_i)$
- Kearns Mansour 96

(α, β) -Tsallis entropy

Family of top-down DT learning algorithms

A single criterion which interpolates all three!

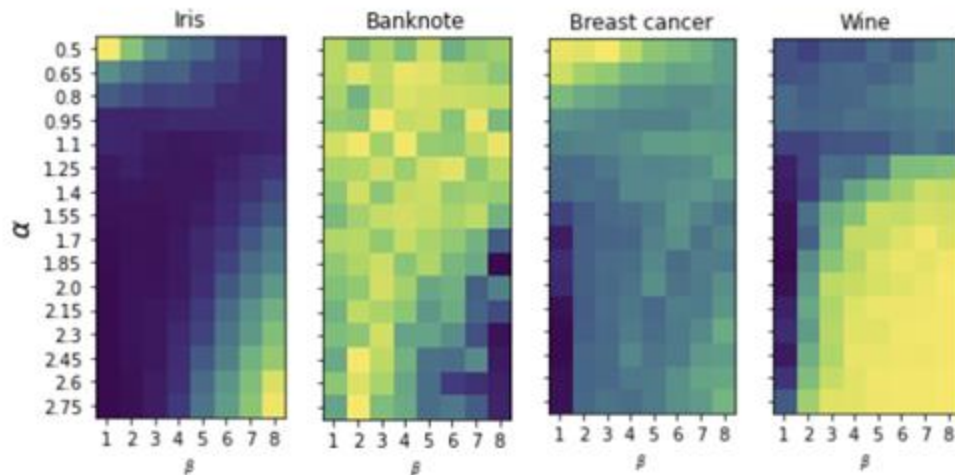
$$g_{\alpha, \beta}^{\text{TSALLIS}}(P) := \frac{C}{\alpha - 1} \left(1 - \left(\sum_{i=1}^c p_i^\alpha \right)^\beta \right)$$

Splitting criterion [\[Balcan and Sharma \(UAI 2024\)\]](#)

$$g_{2,1}^{\text{TSALLIS}}(P)$$

$$g_{\frac{1}{2},2}^{\text{TSALLIS}}(P)$$

$$\lim_{\alpha \rightarrow 1} g_{\alpha,1}^{\text{TSALLIS}}(P)$$



Theorem: We can learn to tune (α, β) using $O\left(\frac{t \log |\mathcal{F}| t}{\epsilon^2}\right)$ problem samples.

Splitting criterion [Balcan and Sharma (UAI 2024)]

Theorem: We can learn to tune (α, β) using $O\left(\frac{t \log |\mathcal{F}| t}{\epsilon^2}\right)$ problem samples.

Proof insights:

- Uses dual function (accuracy as a function of (α, β) on a fixed instance (X, y)) analysis
 - Dual function is piecewise-constant with boundaries given by exponential equations in (α, β) :
- Induction over top-down rounds, bounding the number of distinct behaviors (which node is split and how) in each round
- Over t rounds, $\tilde{O}(|\mathcal{F}|^{2t} t^{2t})$ distinct behaviors, which implies pseudo-dimension is $O(t \log |\mathcal{F}| t)$.

Gradient-boosted decision trees [\[Balcan and Sharma \(ArXiv 2025\)\]](#)

Regularized objective over a collection of K trees (size at most t),

$$L(\{T_i\}, D) = l(\{T_i\}, D) + \frac{1}{2} \lambda \sum_k \|\text{weights of leaves in } T_k\|^2$$

Splitting-criterion in XGBOOST [\[Chen and Guestrin \(2016\)\]](#):

- Across all nodes of all trees in the ensemble, split the one that maximizes a score based on first and second order gradients $\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}$

State-of-the-art approach for tabular datasets!

[\[McElfresh et al. \(NeurIPS 2023\), Jayawardhana et al. \(2025\)\]](#)

Key idea: show **piecewise-constant dual with polynomial boundaries**

Gradient-boosted decision trees [Balcan and Sharma (ArXiv 2025)]

Regularized objective over a collection of K trees (size at most t),

$$L(\{T_i\}, D) = l(\{T_i\}, D) + \frac{1}{2} \lambda \sum_k \|\text{weights of leaves in } T_k\|^2$$

Splitting-criterion in XGBOOST [Chen and Guestrin (2016)]:

- Across all nodes of all trees in the ensemble, split the one that maximizes a score based on first and second order gradients $\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda}$

There are at most $tK|\mathcal{F}|$ different candidate splits, or at most $t^2K^2|\mathcal{F}|^2$ pairs

Also over the course of XGBOOST, we have at most tK splits.

⇒ Computable using a GJ algorithm with at most $(t^2K^2|\mathcal{F}|^2)^{tK}$ predicates (degree 6)

⇒ $\text{Pdim}(U) = O(tK \log(tK|\mathcal{F}|))$

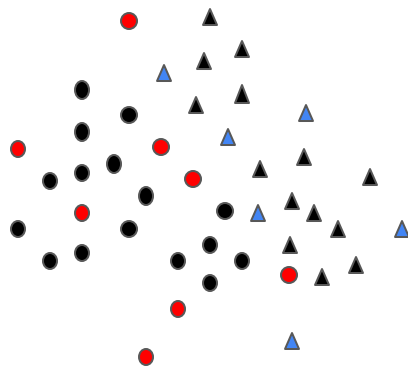
Roadmap

- ❖ Introduction
- ❖ Major techniques used in practice
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ Data-driven algorithm design
 - Learning-theoretic guarantees
 - Distributional learning
 - Online learning
- ❖ **Tuning core ML algorithms**
 - Linear regression, decision trees
 - **Semi-supervised learning, neural networks**
- ❖ Conclusion

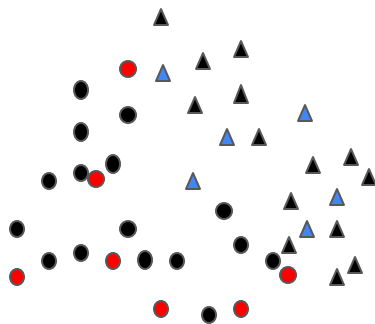
Example: Semi-Supervised Learning [Balcan and Sharma (NeurIPS 2021)]

- ★ Repeated problems e.g. emails on an email server, spam vs. non-spam

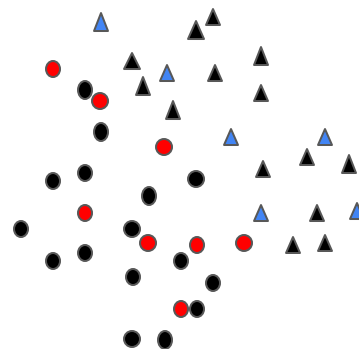
Goal: learn how to connect points using a graph s.t. a (hard or soft) min-cut yields accurate predictions



Day 1



Day 2

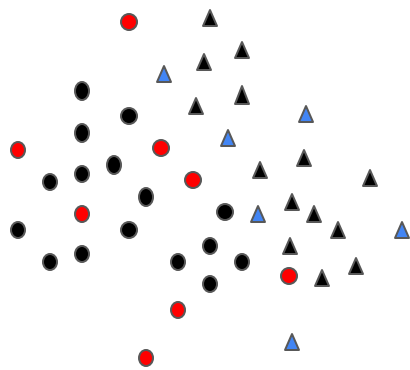


Day 3

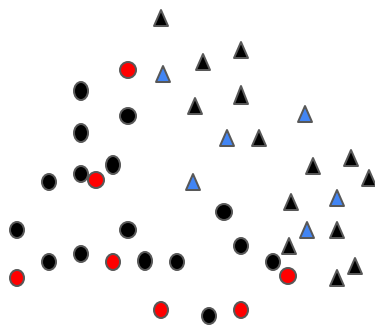
Example: Semi-Supervised Learning [Balcan and Sharma (NeurIPS 2021)]

- ★ Graph edges are set using some kernel with hyperparameters
 - Polynomial kernel: $(\langle f(u), f(v) \rangle + \alpha)^d$
 - RBF kernel: $\exp(-d(u, v)^2/\sigma^2)$
- ★ Instances I: partially labeled datasets; Utility: average accuracy of graph SSL

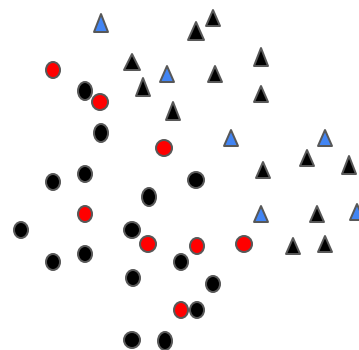
Piecewise-constant dual function with boundaries given by poly/exp equations



Day 1



Day 2

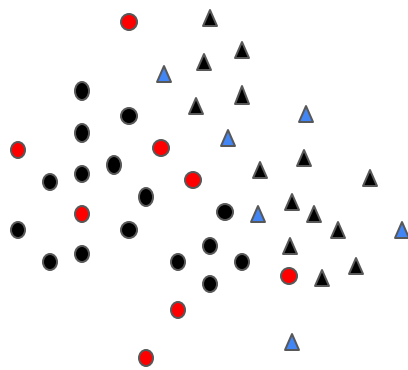


Day 3

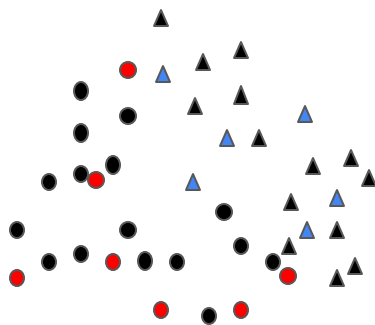
Example: Semi-Supervised Learning [Balcan and Sharma (NeurIPS 2021)]

Goal: learn how to connect points using a graph s.t. a (hard or soft) min-cut yields accurate predictions

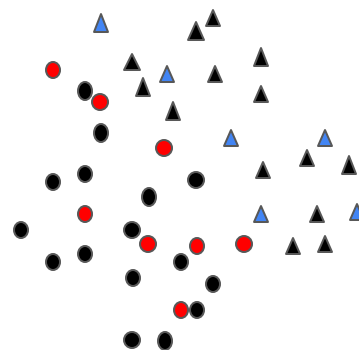
- **statistical learning**: tight upper+lower bounds on learning-theoretic complexity
- **online learning**: no regret by showing critical points are dispersed; primal-dual algorithm for computing pieces exactly;
- faster approx pieces using conjugate gradient method [Sharma and Jones, UAI 2023]



Day 1



Day 2



Day 3

Refined GJ Framework [Bartlett, Indyk, Wagner, COLT'22]

Example application: Low-rank approximation.

Inst, I : Given a sparse matrix $A \in \mathbb{R}^{n \times n}$ with $\|A\|_F = 1$, target rank $k < n$.

Goal: Sparse matrix \tilde{A} with rank k that minimizes (approximates A well).

Exact algorithm based on SVD (singular value decomposition) is inefficient!

Faster algorithm IVY [Indyk, Vakilian, Yuan '19] is family of parameterized heuristics uses a $m \times n$ auxiliary matrix (runtime nearly linear in #non-zero entries!).

Theorem: Sample complexity of tuning IVY is $O(mn/\varepsilon^2)$.

Applications [ML, stats, optimization]

Low-rank approximation [Bartlett, Indyk, Wagner, COLT 2022]

Regularizing linear (Elastic Net) and logistic regression [BKST NeurIPS 2022, BNS NeurIPS 2023, BGS 2025]

Simulated Annealing [Blum, Dan, Seddighin, AISTATS 2021]

Learning to branch and cut [Balcan, Dick, Sandholm, Vitercik, ICML 2018, JACM 2024]

Clustering (both k-center and hierarchical) [BNVW COLT 2017, BDW NeurIPS 2018, BDL ICLR 2020]

Gradient descent [Gupta and Roughgarden, ITCS 2016]

Integer and Linear Programming [Balcan et al., Khodak et al., Cheng and Basu, Sakaue and Oki (2024)]

More applications [CS theory, Comp bio, Mech design, Energy ...]

Knapsack, Maximum Weighted Independent Set [Gupta and Roughgarden, ITCS 2016, Balcan et al., FOCS 2018, Sun et al. 2022]

Max cut, Max 2-SAT [Balcan et al., COLT 2017]

Dynamic Programming, Sequence Alignment [Balcan et al., COLT 2017, STOC 2021, NeurIPS 2024]

Mechanism and game design [Balcan et al., EC18, NeurIPS 24, Jin et al. NeurIPS 24, Dütting et al. EC 2025]

Energy and climate science [Mathioudaki et al., 2023, Bostandoost et al. 2024]

Tuning deep networks: parameters and hyperparameters

- **Hyperparameter space** $A = [\alpha_{\min}, \alpha_{\max}] \subset \mathbb{R}$ (hyperparameter α)

fixed during training

- **Model parameter space** $W \subset \mathbb{R}$ (parameters/weights w)

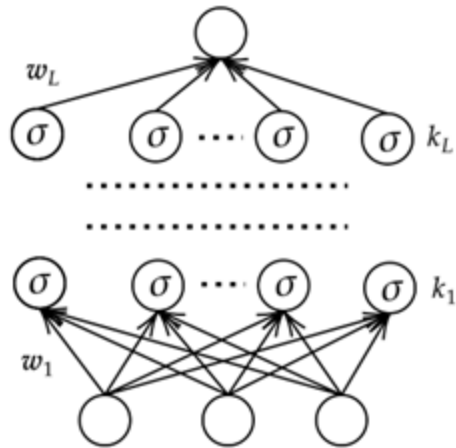
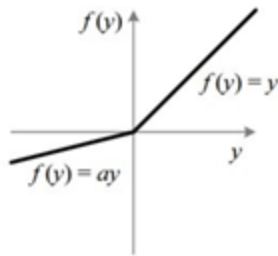
updated during training

- Example (learning activation functions):

- Consider a DNN $\tau_{\alpha, w}$ with model weights $w = (w_1, \dots, w_L)$

- Parametric ReLU activation function

$$\text{PReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{otherwise} \end{cases}$$



- More generally, one can interpolate* any activation functions

$$\sigma(z) = \alpha o_1(z) + (1 - \alpha) o_2(z)$$

where o_1, o_2 are common activation functions, α is interpolation hyperparameter

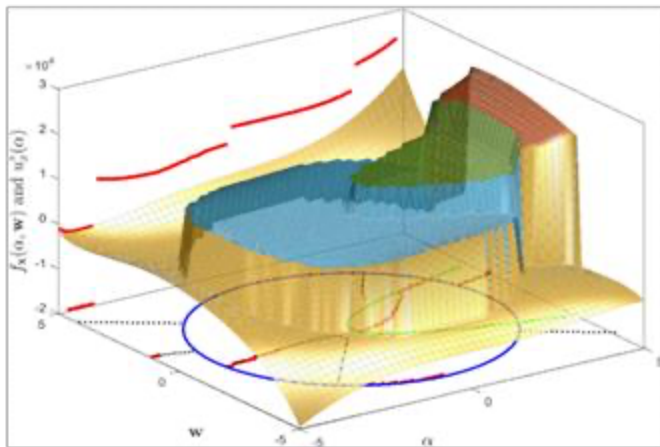
Model vs optimization hyperparameters

	“Model” or “architectural” hyperparameters	“Optimization” hyperparameters
Impact learned weights w	YES	YES
Are part of:	learned deep network $\tau_{\alpha, w}$	optimization algorithm
Examples	activation function hyperparameters, kernel parameters	learning rate, learning schedule, momentum

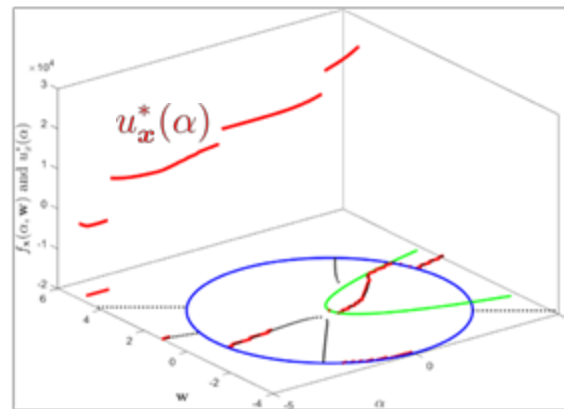
Tuning deep network hyperparameters [Balcan, Nguyen, Sharma, 2025]

Instances I : labeled datasets; utility: avg acc

Sample complexity of **data-driven tuning** of model hyperparameters (e.g. activation fns, GNN kernels) with p/w poly *parameter-dependent dual fn*



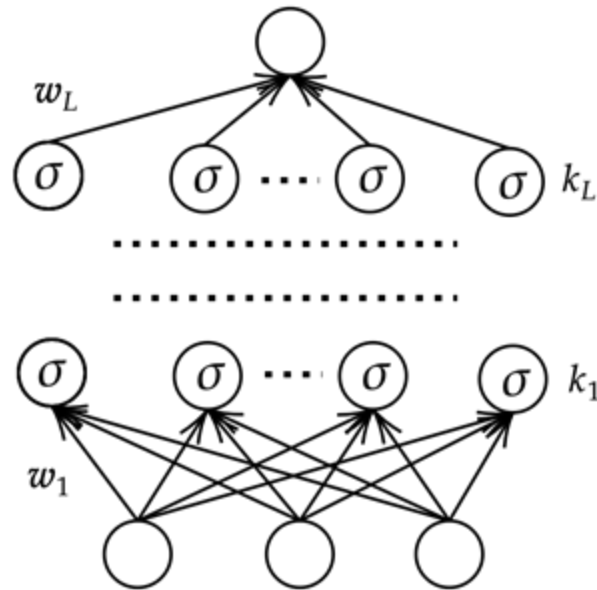
Poly surface depicting parameter-dependent dual $f_x(\alpha, w)$ and piecewise structure of dual



$u_x^*(\alpha) := u_\alpha(x) = \sup_w f_x(\alpha, w)$
new techniques to bound discontinuities and oscillations of dual

Learning the interpolated activation function

- DNN $\tau_{\alpha, w}$ with L layers
- Layer i : W_i params (total W), k_i nodes (total k)
- $\sigma(z) = \alpha o_1(z) + (1 - \alpha) o_2(z)$, where o_1, o_2 piecewise poly. with max degree Δ , p breakpoints
- T samples (not assumed iid) in each problem instance



Learning the interpolated activation function [Balcan, Nguyen, Sharma, 2025]

Theorem (informal): $\text{Pdim}(U) = O(\log M + d \log(\Delta N))$, where

M is the number of connected components

N is the number of boundaries

d is the dimension of w

Δ is the maximum polynomial degree

Application:

For the activation function interpolation:
 $\text{Pdim}(U) = O(L^2 W \log \Delta + L W \log(Tpk))$

Open Q: Improve?

Tuning learning rate in gradient descent [Sharma, 2025]

Gradient descent algorithm: Instance is (x, f) , loss = num steps till convergence.

Inputs: initial point x , iterations H , threshold θ . Hyperparameter: learning rate η

```
1: Initialize  $x_1 \leftarrow x$ 
2: for  $i = 1, \dots, H$  do
3:   if  $\|\nabla f(x_i)\| < \theta$  then
4:     Return  $x_i$ 
5:    $x_{i+1} = x_i - \eta \nabla f(x_i)$ 
```

Output: x_i

Prior work by Gupta and Roughgarden (2016):

Assumes: f is convex and smooth

Sample complexity of tuning learning rate is $O(H^3/\varepsilon^2)$

We get $O(H^3/\varepsilon^2)$ sample complexity even for non-convex non-smooth functions in deep networks!

Open Q: Improve?

Learning from small-samples

Can we figure out how to find the good hyperparameters for larger datasets/models based on learning good hyperparameters for smaller datasets/models?

[Chatziafratis, Karmarkar, Li and Vitercik, 2025] give some initial answers for algorithm selection in clustering

Open Q:
Neural Networks?
LLMs scaling laws?

Algorithms with Predictions

A new approach to designing algorithms, with predictions from machine learning

- Clustering [Ergun et al. ICLR 2022, Silwal et al. ICLR 2023, Braverman et al. arXiv 2025]
- Graph algorithms [Dinitz et al. 2021, Chen et al. ICML 2022, Aamand et al. arXiv 2025]
- Many, many more [350+ recent papers <https://algorithms-with-predictions.github.io/>]

Hot research topic, but how do we actually learn the predictions? [Khodak et al. NeurIPS 2022]

Open Q:
Learning-augmented
hyperparameter tuning?

Roadmap

- ❖ Introduction
- ❖ Major techniques used in practice
 - Bayesian Optimization
 - Bandit-based methods
 - Case studies: NAS and LLMs
- ❖ Data-driven algorithm design
 - Distributional learning
 - Online learning
- ❖ Tuning core ML algorithms
 - Linear regression, decision trees
 - Semi-supervised learning, neural networks
- ❖ **Conclusion**

Conclusion

- Last ~20 years:
 - Explosive growth in powerful ML algorithms and their range of applications
 - New practically successful approaches to algorithm design
- Last 10 years:
 - Machine learning for algorithm design acquired solid foundations in learning theory
- Last 5 years:
 - Hyperparameter tuning is rapidly transforming from an art to a principled science

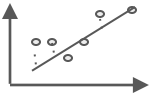
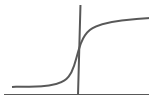
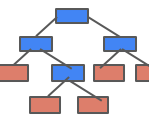
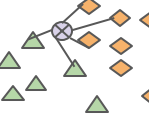
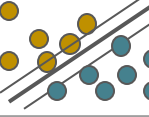
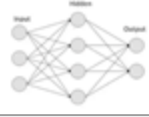
Next five years and beyond ...

- Other applications to tuning important hyperparameters and algorithms
- Focus on statistical complexity \longrightarrow computationally efficient methods?
- Making currently used approaches in practice more structure-aware
- Beyond the worst-case complexity: distribution-dependent bounds
- More challenging high-dimensional and distributed settings
 - E.g. extend our model hyperparameter tuning result to multiple hyperparameters

Next five years and beyond ...

- For essentially all ML algorithms of interest, we will know how to provably configure hyperparameters
- ML can be used to solve its own problems of robustness, interpretability and trustworthiness
- Reliable and safe use of AI is going to be critical
 - We require more from generative AI
 - Data-driven approaches are promising candidates
- ML-designed algorithms for ML can unlock as-yet untapped potential

Questions and transition to panel...

	Linear regression	Ridge/Lasso penalties
	Logistic regression	L1, L2 regularization penalties
	Decision Tree	splitting criterion, pruning cost, max depth
	k-Nearest neighbors	k, weights, metric, abstention threshold
	Support Vector Machines	C, kernel, gamma
	Neural Networks	activation function, learning schedule, ...

References

- [1] Mockus, Jonas. "The Bayesian approach to local optimization." In *Bayesian approach to global optimization: Theory and applications*, pp. 125-156. Dordrecht: Springer Netherlands, 1989.
- [2] Linial, Nathan, Yishay Mansour, and Noam Nisan. "Constant depth circuits, Fourier transform, and learnability." *Journal of the ACM (JACM)* (1993).
- [3] Srinivas, Niranjana, Andreas Krause, Sham Kakade, and Matthias Seeger. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design." In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015-1022. Omnipress, 2010.
- [4] Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization." *Advances in neural information processing systems* 24 (2011).
- [5] Maclaurin, Dougal, David Duvenaud, and Ryan Adams. "Gradient-based hyperparameter optimization through reversible learning." In *International conference on machine learning*, pp. 2113-2122. PMLR, 2015.
- [6] Domhan, Tobias, Jost Tobias Springenberg, and Frank Hutter. "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves." In *IJCAI*, vol. 15, pp. 3460-8. 2015.
- [7] Gupta, Rishi, and Tim Roughgarden. "A PAC approach to application-specific algorithm selection." In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pp. 123-134. 2016.
- [8] Jamieson, Kevin, and Ameet Talwalkar. "Non-stochastic best arm identification and hyperparameter optimization." In *Artificial intelligence and statistics*, pp. 240-248. PMLR, 2016.
- [9] Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In *International conference on machine learning*, pp. 1126-1135. PMLR, 2017.

References

- [10] Frazier, Peter I. "A tutorial on Bayesian optimization." *arXiv preprint arXiv:1807.02811* (2018).
- [11] Balcan, Maria-Florina, Travis Dick, and Ellen Vitercik. "Dispersion for data-driven algorithm design, online learning, and private optimization." In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 603-614. IEEE, 2018.
- [12] Franceschi, Luca, Paolo Frasconi, Saverio Salzo, Riccardo Grazi, and Massimiliano Pontil. "Bilevel programming for hyperparameter optimization and meta-learning." In *International conference on machine learning*, pp. 1568-1577. PMLR, 2018.
- [13] Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A novel bandit-based approach to hyperparameter optimization." *Journal of Machine Learning Research* 18, no. 185 (2018): 1-52.
- [14] Falkner, Stefan, Aaron Klein, and Frank Hutter. "BOHB: Robust and efficient hyperparameter optimization at scale." In *International conference on machine learning*, pp. 1437-1446. PMLR, 2018.
- [15] Balcan, Maria-Florina, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. "Learning to branch." In *International conference on machine learning*, pp. 344-353. PMLR, 2018.
- [16] Kandasamy, Kirthevasan, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. "Neural architecture search with bayesian optimisation and optimal transport." *Advances in neural information processing systems* 31 (2018).
- [17] Hazan, Elad, Adam Klivans, and Yang Yuan. "Hyperparameter Optimization: A Spectral Approach." *ICLR* (2018).
- [18] Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "DARTS: Differentiable Architecture Search." In *International Conference on Learning Representations*, 2019.

References

- [19] Berkenkamp, Felix, Angela P. Schoellig, and Andreas Krause. "No-regret Bayesian optimization with unknown hyperparameters." *Journal of Machine Learning Research* 20, no. 50 (2019): 1-24.
- [20] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In *International conference on machine learning*, pp. 6105-6114. PMLR, 2019.
- [21] Feurer, Matthias, and Frank Hutter. *Hyperparameter optimization*. Springer International Publishing, 2019.
- [22] Li, Liam, and Ameeth Talwalkar. "Random search and reproducibility for neural architecture search." In *Uncertainty in artificial intelligence*, pp. 367-377. PMLR, 2020.
- [23] Maria-Florina Balcan. *Data-Driven Algorithm Design*. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.
- [24] Kandasamy, Kirthevasan, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R. Collins, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. "Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly." *Journal of Machine Learning Research* 21, no. 81 (2020): 1-27.
- [25] Balcan, Maria-Florina, Travis Dick, and Manuel Lang. "Learning to Link." In *International Conference on Learning Representation*. 2020.
- [26] Lin, Jimmy, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. "Generalized and scalable optimal sparse decision trees." In *International conference on machine learning*, pp. 6150-6160. PMLR, 2020.
- [27] Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

References

- [28] Balcan, Maria-Florina, Travis Dick, and Manuel Lang. "Learning to Link." In *International Conference on Learning Representation*. 2020.
- [29] Parker-Holder, Jack, Vu Nguyen, and Stephen J. Roberts. "Provably efficient online hyperparameter optimization with population-based bandits." *Advances in neural information processing systems* 33 (2020): 17200-17211.
- [30] Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. "Learning piecewise Lipschitz functions in changing environments." In *International Conference on Artificial Intelligence and Statistics*, pp. 3567-3577. PMLR, 2020.
- [31] Balcan, Maria-Florina, and Dravyansh Sharma. "Data driven semi-supervised learning." *NeurIPS* (2021): 14782-14794.
- [32] Balcan, Maria-Florina, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. "How much data is sufficient to learn high-performing algorithms? Generalization guarantees for data-driven algorithm design." *Symposium on Theory of Computing (STOC)*, 2021.
- [33] Blum, Avrim, Chen Dan, and Saeed Seddighin. "Learning complexity of simulated annealing." In *International conference on artificial intelligence and statistics*, pp. 1540-1548. PMLR, 2021.
- [34] Yang, Greg, and Edward J. Hu. "Tensor programs iv: Feature learning in infinite-width neural networks." In *International Conference on Machine Learning*, pp. 11727-11737. PMLR, 2021.
- [35] Balcan, Maria-Florina, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. "Learning-to-learn non-convex piecewise-Lipschitz functions." *Advances in Neural Information Processing Systems* 34 (2021): 15056-15069.
- [36] Bartlett, Peter, Piotr Indyk, and Tal Wagner. "Generalization bounds for data-driven numerical linear algebra." In *Conference on Learning Theory*, 2022.
- [37] Balcan, Maria-Florina, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. "Provably tuning the ElasticNet across instances." *Advances in Neural Information Processing Systems* 35 (2022): 27769-27782.

References

- [38] Sun, Bo, Lin Yang, Mohammad Hajiesmaili, Adam Wierman, John CS Lui, Don Towsley, and Danny HK Tsang. "The online knapsack problem with departures." *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, no. 3 (2022): 1-32.
- [39] Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas et al. "Training compute-optimal large language models." arXiv preprint arXiv:2203.15556 (2022).
- [40] Lindauer, Marius, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. "SMAC3: A versatile Bayesian optimization package for hyperparameter optimization." *Journal of Machine Learning Research* 23 (2022): 1-9.
- [41] Balcan, Maria-Florina, Anh Nguyen, and Dravyansh Sharma. "New bounds for hyperparameter tuning of regression problems across instances." *Advances in Neural Information Processing Systems* 36 (2023): 80066-80078.
- [42] Sharma, Dravyansh, and Maxwell Jones. "Efficiently learning the graph for semi-supervised learning." In *Uncertainty in Artificial Intelligence*, 2023.
- [43] Silwal, Sandeep, Sara Ahmadian, Andrew Nystrom, Andrew McCallum, Deepak Ramachandran, and Seyed Mehran Kazemi. "KwikBucks: Correlation Clustering with Cheap-Weak and Expensive-Strong Signals." In *The Eleventh International Conference on Learning Representations, ICLR (2023)*.
- [44] Balcan, Maria-Florina, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. "An analysis of robustness of non-Lipschitz networks." *Journal of Machine Learning Research* 24, no. 98 (2023): 1-43.
- [45] Koch, Caleb, Carmen Strassle, and Li-Yang Tan. "Properly learning decision trees with queries is NP-hard." In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 2383-2407. IEEE, 2023.
- [46] Mathioudaki, Angeliki, Georgios Tsaousoglou, Emmanouel Varvarigos, and Dimitris Fotakis. "Data-Driven Optimization of Electric Vehicle Charging Stations." In *2023 International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1-6. IEEE, 2023.

References

- [47] Sharma, Dravyansh. "Data-driven algorithm design and principled hyperparameter tuning in machine learning." PhD dissertation, CMU (2024).
- [48] Balcan, Maria-Florina, and Dravyansh Sharma. "Learning Accurate and Interpretable Decision Trees." In *Uncertainty in Artificial Intelligence*, pp. 288-307. PMLR (2024). Extended version "Learning Accurate and Interpretable Tree-based Models" arXiv preprint arXiv:2405.15911 (2025).
- [49] Franceschi, Luca, Michele Donini, Valerio Perrone, Aaron Klein, Cédric Archambeau, Matthias Seeger, Massimiliano Pontil, and Paolo Frasconi. "Hyperparameter Optimization in Machine Learning." *arXiv preprint arXiv:2410.22854* (2024).
- [50] Sharma, Dravyansh. "No internal regret with non-convex loss functions." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 13, pp. 14919-14927. 2024.
- [51] Cheng, Hongyu, and Amitabh Basu. "Learning cut generating functions for integer programming." *Advances in Neural Information Processing Systems* 37 (2024): 61455-61480.
- [52] Balcan, Maria-Florina, Christopher Seiler, and Dravyansh Sharma. "Accelerating ERM for data-driven algorithm design using output-sensitive techniques." *Advances in Neural Information Processing Systems* 37 (2024): 72648-72687.
- [53] Sakaue, Shinsaku, and Taihei Oki. "Generalization bound and learning methods for data-driven projections in linear programming." *Advances in Neural Information Processing Systems* 37 (2024): 12825-12846.
- [54] Elias, Marek, Haim Kaplan, Yishay Mansour, and Shay Moran. "Learning-augmented algorithms with explicit predictors." *Advances in Neural Information Processing Systems* 37 (2024): 97972-98008.
- [55] Sambharya, Rajiv, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. "Learning to warm-start fixed-point optimization algorithms." *Journal of Machine Learning Research* 25, no. 166 (2024): 1-46.

References

- [56] Dumouchelle, Justin, Esther Julien, Jannis Kurtz, and Elias B. Khalil. "Neur2bilo: Neural bilevel optimization." *Advances in Neural Information Processing Systems* 37 (2024): 86688-86719.
- [57] Xie, Yaqi, Will Ma, and Linwei Xin. "VC theory for inventory policies." *arXiv preprint arXiv:2404.11509* (2024).
- [58] Bostandoost, Roozbeh, Walid A. Hanafy, Adam Lechowicz, Noman Bashir, Prashant Shenoy, and Mohammad Hajiesmaili. "Data-driven Algorithm Selection for Carbon-Aware Scheduling." *ACM SIGENERGY Energy Informatics Review* 4, no. 5 (2024): 148-153.
- [59] Sharma, Dravyansh, and Arun Suggala. "Offline-to-online hyperparameter transfer for stochastic bandits." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 19, pp. 20362-20370. 2025.
- [60] Balcan, Maria-Florina, Anh Tuan Nguyen, and Dravyansh Sharma. "Sample complexity of data-driven tuning of model hyperparameters in neural networks with structured parameter-dependent dual function." *Advances in Neural Information Processing Systems* 38 (2025).
- [61] Schneider L, Bischl B, Feurer M. "Overtuning in Hyperparameter Optimization." 4th International Conference on Automated Machine Learning AutoML (2025).
- [62] Balcan, Maria-Florina, Anh Tuan Nguyen, and Dravyansh Sharma. "Algorithm Configuration for Structured Pfaffian Settings." *TMLR* (2025).
- [63] Jiao, Xianqi, Jia Liu, and Zhiping Chen. "Learning complexity of gradient descent and conjugate gradient algorithms." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 17, pp. 17671-17679. 2025.
- [64] Dütting, Paul, Michal Feldman, Tomasz Ponitka, and Ermis Soumalias. "The pseudo-dimension of contracts." In *Proceedings of the 26th ACM Conference on Economics and Computation*, pp. 514-539. 2025.

References

- [65] Blum, Avrim, and Vaidehi Srinivas. "Competitive strategies to use "warm start" algorithms with predictions." In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 3775-3801. Society for Industrial and Applied Mathematics, 2025.
- [66] Iwata, Tomoharu, and Shinsaku Sakaue. "Learning to Generate Projections for Reducing Dimensionality of Heterogeneous Linear Programming Problems." In *Forty-second International Conference on Machine Learning (2025)*.
- [67] Balcan, Maria-Florina, Saumya Goyal, and Dravyansh Sharma. "Distribution-dependent Generalization Bounds for Tuning Linear Regression Across Tasks." *arXiv preprint arXiv:2507.05084* (2025).
- [68] Du, Ally Yalei, Eric Huang, and Dravyansh Sharma. "Tuning Algorithmic and Architectural Hyperparameters in Graph-Based Semi-Supervised Learning with Provable Guarantees." In *The 41st Conference on Uncertainty in Artificial Intelligence (2025)*.