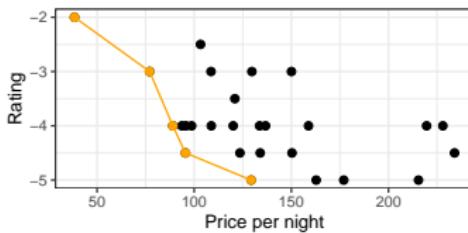


Optimization in Machine Learning

Multi-criteria Optimization Introduction

28.01.25



Learning goals

- What is multi-criteria optimization?
- Motivating examples
- Pareto optimality

find cheap room
with good rating

INTRODUCTORY EXAMPLE

Often we want to solve optimization problems concerning several goals.

General applications:

- Medicine: maximum effect, but minimum side effect of a drug.
- Finances: maximum return, but minimum risk of an equity portfolio.
- Production planning: maximum revenue, but minimum costs.
- Booking a hotel: maximum rating, but minimum costs.

In machine learning:

- Sparse models: maximum predictive performance, but minimal number of features.
- Fast models: maximum predictive performance, but short prediction time.
- ...



INTRODUCTORY EXAMPLE

Example: Choose the best hotel by maximizing ratings subject to a maximum price per night.

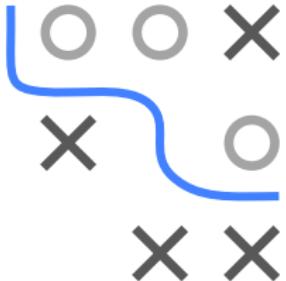
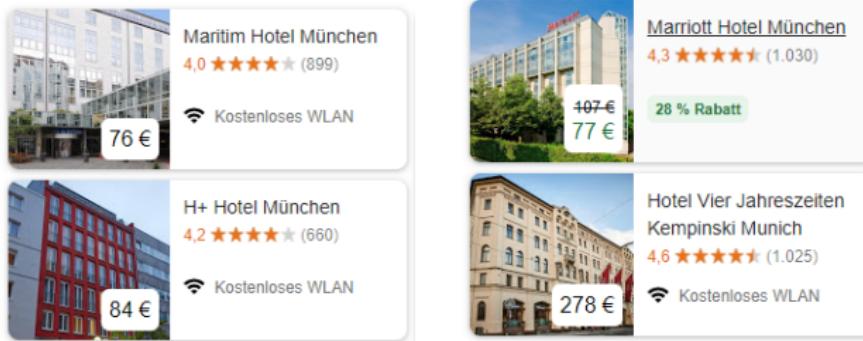
Problems:

- The result depends on how we select the maximum price; different price bounds give different solutions.
- We could also choose a minimum rating and optimize the price.
- The more objectives we include, the harder it gets to fix cutoffs like these.

Goal: Find a more general approach to solve multi-criteria problems.



INTRODUCTORY EXAMPLE



When booking a hotel: find the hotel with

- minimum price per night (**costs**), and
- maximum user rating (**performance**).

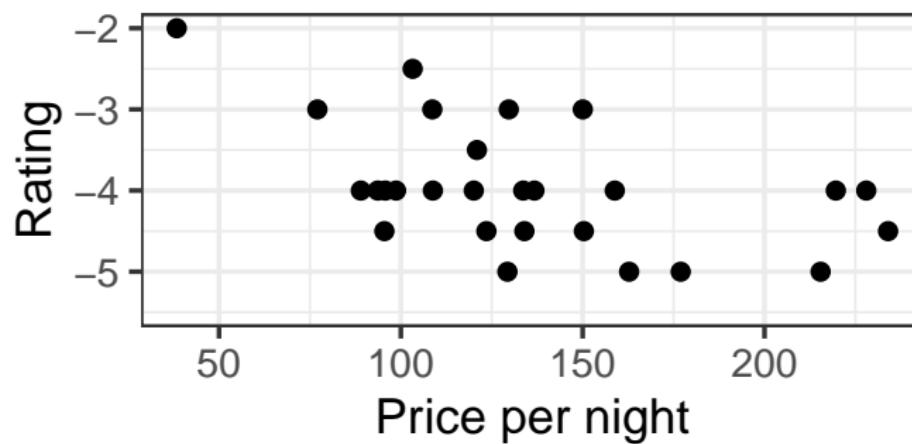
Since our standard is to minimize objectives, we minimize negative ratings.

INTRODUCTORY EXAMPLE

The objectives often conflict with each other:

- Lower price → usually lower hotel rating.
- Better rating → usually higher price.

Example: (negative) average rating by hotel guests (1–5) vs. average price per night.

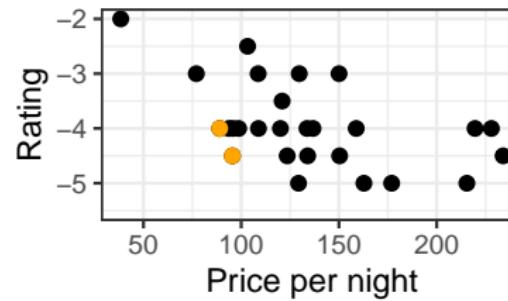
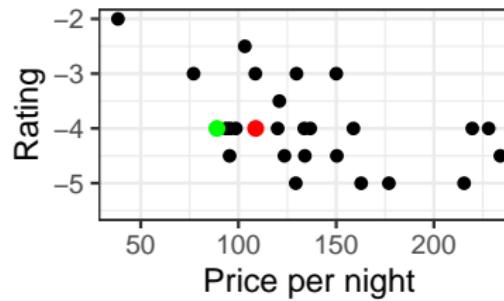
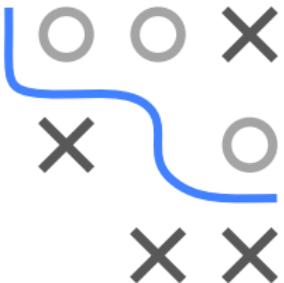


Because we want to have "Cost sol = minimum"

INTRODUCTORY EXAMPLE

We cannot directly establish a total order on the objective space:

- Left: A hotel with rating 4 for 89 Euro ($f^{(1)} = (89, -4.0)$) is better than a hotel for 108 Euro with the same rating ($f^{(2)} = (108, -4.0)$).
- Right: How to decide if $f^{(1)} = (89, -4.0)$ is better or worse than $f^{(2)} = (95, -4.5)$?
- How much is one *rating point* worth?



Let $f = (\text{price}, -\text{rating})$. The candidate $f^{(1)} = (89, -4.0)$ dominates $f^{(2)} = (108, -4.0)$. That is, $f^{(1)}$ is at least as good in every dimension, and strictly better in one dimension. Hence $f^{(2)} \prec f^{(1)}$. But for $f^{(1)} = (89, -4.0)$ and $f^{(2)} = (95, -4.5)$ we cannot say any is strictly better.

HOW TO DEFINE OPTIMALITY?

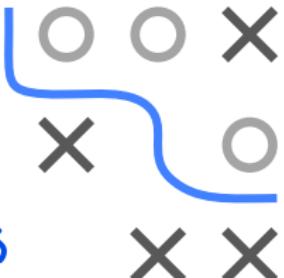
- We call these **incomparable** (or *equivalent* in the Pareto sense):

$$f^{(1)} \not\prec f^{(2)} \quad \text{and} \quad f^{(2)} \not\prec f^{(1)}$$

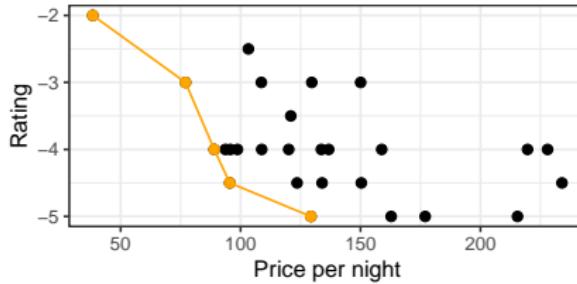
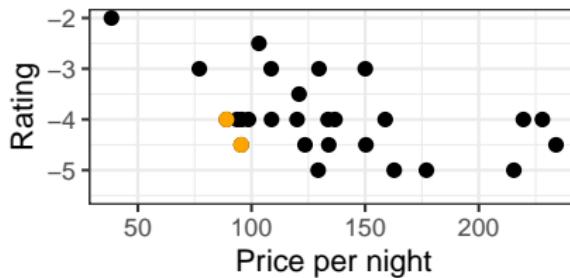
non of the

points dominates

the other.



- All such non-dominated points form the **Pareto front**.



DEFINITION: MULTI-CRITERIA OPTIMIZATION

A multi-criteria optimization problem is defined by

*I f's not
always clean
what min.
means*

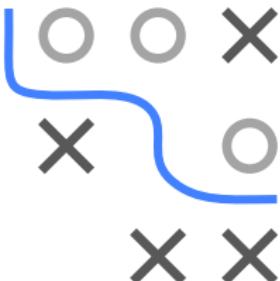
$$\min_{\mathbf{x} \in \mathcal{S}} \mathbf{f}(\mathbf{x}) \Leftrightarrow \min_{\mathbf{x} \in \mathcal{S}} (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})),$$

Bold f

with $\mathcal{S} \subset \mathbb{R}^n$ and a multi-criteria objective function $\mathbf{f} : \mathcal{S} \rightarrow \mathbb{R}^m$, $m \geq 2$.

- **Goal:** minimize multiple target functions simultaneously.
- $(f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top$ maps each candidate \mathbf{x} into the objective space \mathbb{R}^m .
- Typically no single best solution exists, as the objectives are often in conflict and cannot be totally ordered in \mathbb{R}^m .
- W.l.o.g. we always minimize.
- Also called multi-objective optimization, Pareto optimization, etc.

without loss of gen.



PARETO SETS AND PARETO OPTIMALITY

Definition:

Given

$$\min_{\mathbf{x} \in \mathcal{S}} (f_1(\mathbf{x}), \dots, f_m(\mathbf{x})), \quad f_i : \mathcal{S} \rightarrow \mathbb{R}.$$

- A candidate $\mathbf{x}^{(1)}$ (**Pareto**)-dominates $\mathbf{x}^{(2)}$ if $f(\mathbf{x}^{(1)}) \prec f(\mathbf{x}^{(2)})$, i.e.:
 - ➊ $f_i(\mathbf{x}^{(1)}) \leq f_i(\mathbf{x}^{(2)})$ for all i , *for all obj. functions at least as good*
 - ➋ $f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)})$ for at least one j . *as good + for at least one obj.*
- A candidate \mathbf{x}^* not dominated by any other candidate is **Pareto optimal**. *if we can't find a point that dominate the given point, then it's optimal*
- The set of all Pareto optimal candidates is the **Pareto set**
$$\mathcal{P} = \{\mathbf{x} \in \mathcal{S} \mid \nexists \tilde{\mathbf{x}} : f(\tilde{\mathbf{x}}) \prec f(\mathbf{x})\}.$$
- $\mathcal{F} = f(\mathcal{P})$, the **Pareto front**, is the image of that set in objective space.

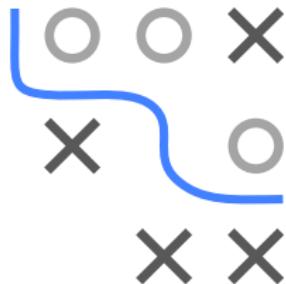
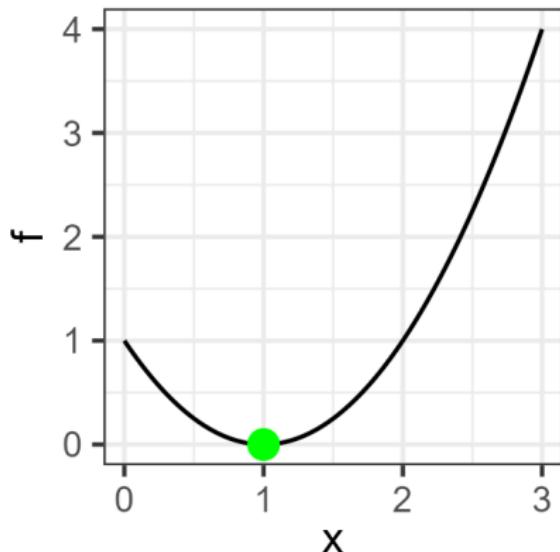


EXAMPLE: ONE OBJECTIVE FUNCTION

We consider

$$\min_x f(x) = (x - 1)^2, \quad 0 \leq x \leq 3.$$

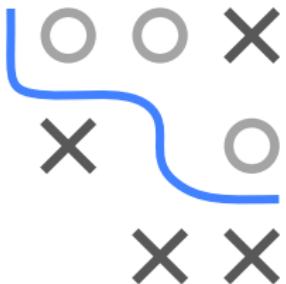
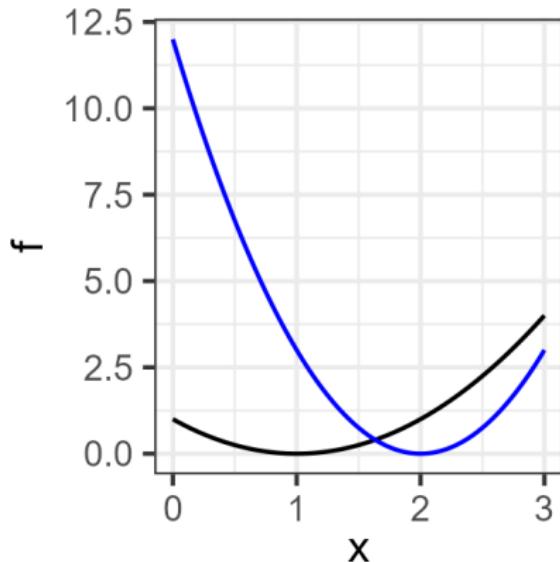
The optimum is at $x^* = 1$.



EXAMPLE: TWO TARGET FUNCTIONS

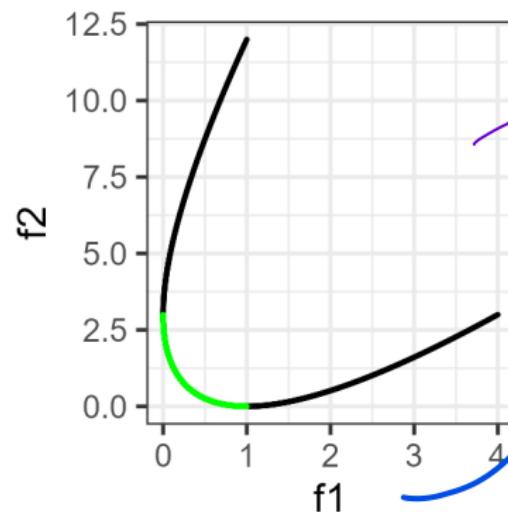
We extend the above problem to two objectives $f_1(x) = (x - 1)^2$ and $f_2(x) = 3(x - 2)^2$. Thus

$$\min_x f(x) = (f_1(x), f_2(x)), \quad 0 \leq x \leq 3.$$

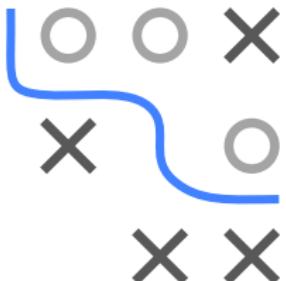


EXAMPLE: TWO TARGET FUNCTIONS

Visualizing both objectives: $(f_1(x), f_2(x))$ for all $x \in [0, 3]$ in the objective space:



The Pareto front is shown in green.



A-PRIORI VS. A-POSTERIORI

- The Pareto set is a set of equally valid solutions.
- Often, we only want a **single** solution in practice.
- Without extra info, there is no unique best solution: one must choose based on further criteria (cost constraints, business preference, etc.).



Two overall strategies:

- **A-priori approach:** incorporate user preferences *before* the optimization.
- **A-posteriori approach:** first find (an approximation of) the entire Pareto set, then let the user choose.

A-PRIORI PROCEDURE

Example: Weighted total

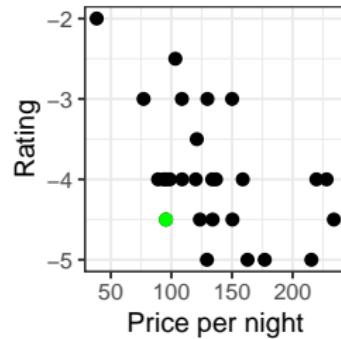
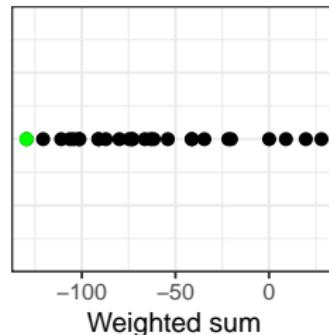
$$\min_{\mathbf{x}} \sum_{i=1}^m w_i f_i(\mathbf{x})$$

boil down
to single criteria



Prior knowledge: "One rating point is worth 50 Euro to this customer."

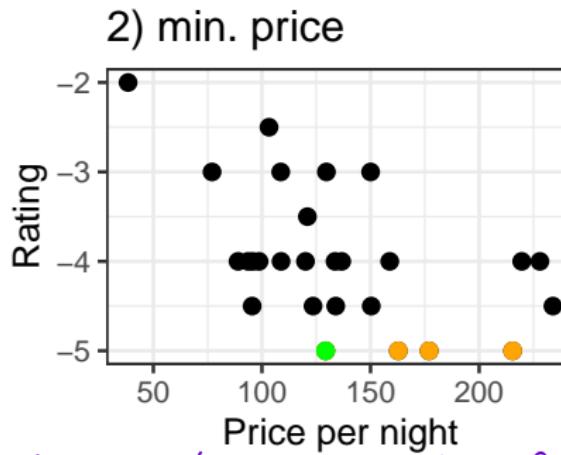
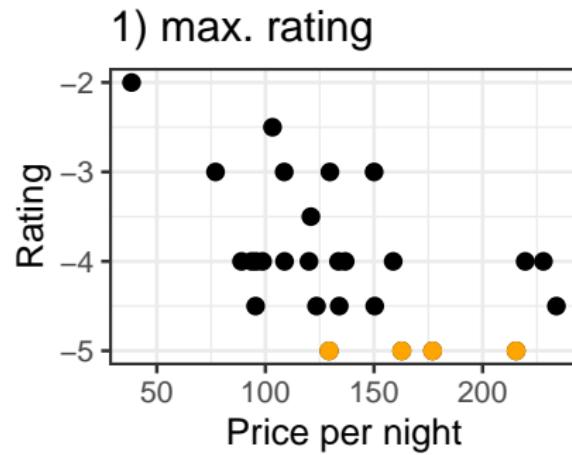
$$\min_{\text{Hotel}} (\text{Price}) - 50 \times (\text{Rating}).$$



A-PRIORI PROCEDURE

Example: Lexicographic method

Prior knowledge: The rating is more important than the price. Hence we optimize the rating first, then the price among those best ratings:



opt. wrt to 1st obj., then 2nd . . .
(depends on how we order)

A-PRIORI PROCEDURE

Lexicographic approach in general:

$$f_1^* = \min_{\mathbf{x} \in \mathcal{S}} f_1(\mathbf{x}),$$

$$f_2^* = \min_{\substack{\mathbf{x} \in \mathcal{S}: \\ f_1(\mathbf{x}) = f_1^*}} f_2(\mathbf{x}),$$

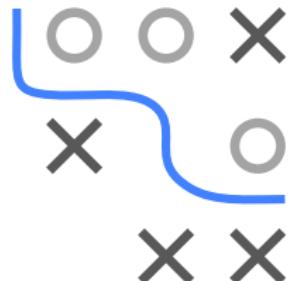
$$f_3^* = \min_{\substack{\mathbf{x} \in \mathcal{S}: \\ f_1(\mathbf{x}) = f_1^*, f_2(\mathbf{x}) = f_2^*}} f_3(\mathbf{x}),$$

⋮

But different orderings yield different solutions.

Summary (a-priori):

- Implicit assumption: single-objective optimization is “easy.”
- We get only *one* solution, which depends strongly on the chosen weighting/order/etc.
- Varying these parameters systematically can produce several solutions, but may miss large parts of the non-dominated set.



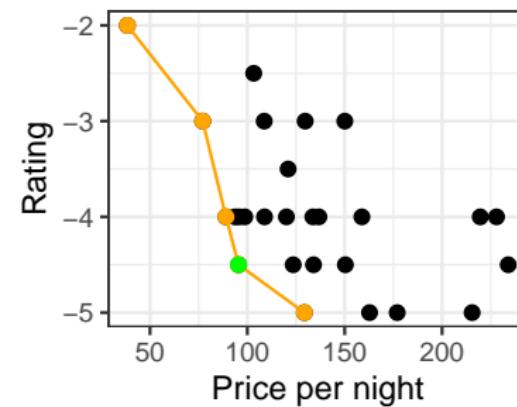
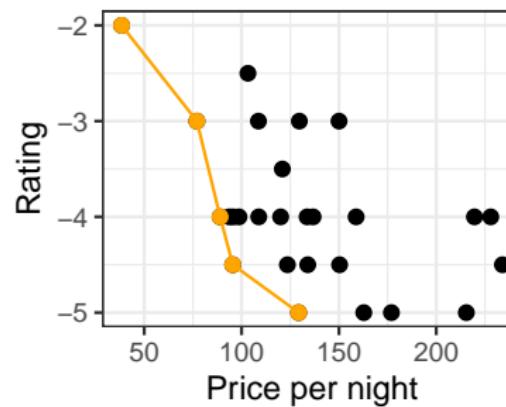
A-POSTERIORI PROCEDURE

A-posteriori methods aim to

- find *all* optimal candidates (or a good approximation of them),
- let the user pick a single solution based on personal preferences or additional info (e.g. hotel location).

Hence, a-posteriori methods are more general.

Example: All Pareto-optimal hotels are shown on the left. User then picks from that subset on the right, based on hidden preferences (location, brand loyalty, etc.):

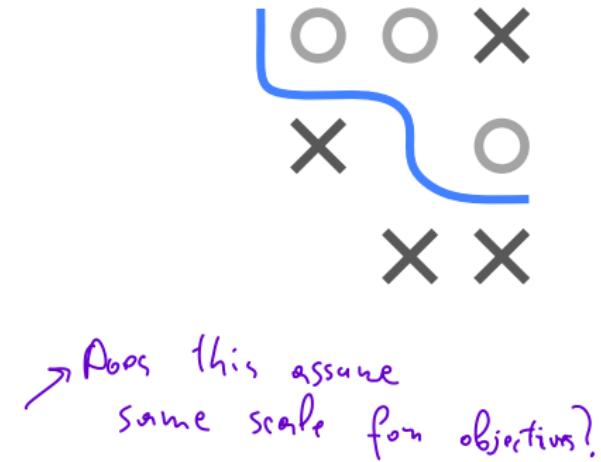
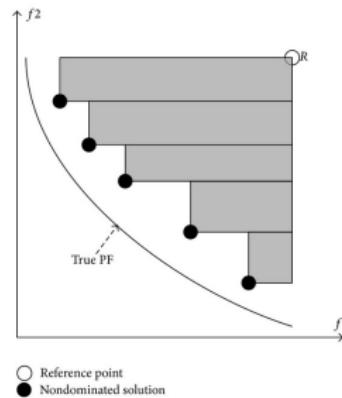


EVALUATION OF SOLUTIONS

A common measure for the performance of a set of candidates $\mathcal{P} \subset \mathcal{S}$ is the **dominated hypervolume**: (HV)

$$S(\mathcal{P}, R) = \lambda \left(\bigcup_{\tilde{x} \in \mathcal{P}} \{x \mid \tilde{x} \prec x \prec R\} \right),$$

where $\lambda(\cdot)$ is the Lebesgue measure and R a pre-specified reference point every point in the set must dominate



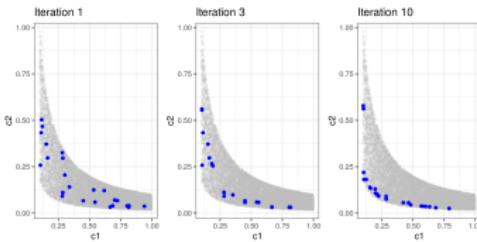
→ Does this assume same scale for objectives?

- HV is measured w.r.t. the reference point R , often chosen so each component is a known “worst” (upper bound) for that objective.
- HV is also called the **S-metric**.
- Its computation is $\mathcal{O}(n^{m-1})$ in the number of points and dimension m .
- Efficient algorithms exist for small m . In ML, we rarely exceed $m = 3$ in practice.

Optimization in Machine Learning

Multi-criteria Optimization Evolutionary Approaches

28.01.25

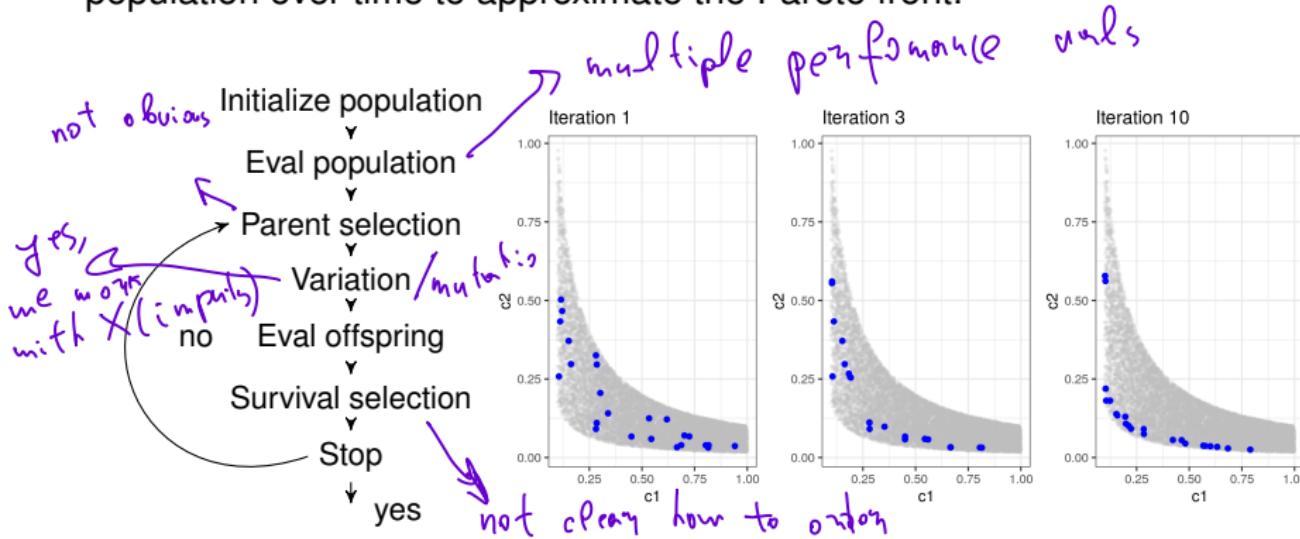


Learning goals

- EMOAs and population-based approach
- NSGA-II
- SMS-EMOA

A-POSTERIORI METHODS AND EVOLUTIONARY ALGORITHMS

Evolutionary multi-objective algorithms (EMOAs) evolve a diverse population over time to approximate the Pareto front.

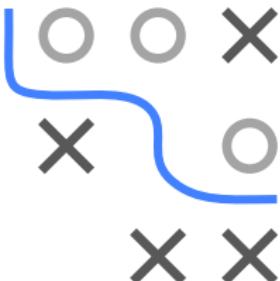


- Population-based search approximates a whole Pareto set in one run.
- The main steps (initialize, select, vary, survive, repeat) mirror standard EAs but must handle multiple objectives.

BASIC EA TEMPLATE LOOP

Algorithm 1: Basic EA template loop

- 1 Initialize and evaluate population $\mathcal{P}_0 \subset \mathcal{S}$ with $|\mathcal{P}_0| = \mu$
 - 2 $t \leftarrow 0$
 - 3 **repeat**
 - 4 Select parents from \mathcal{P}_t and generate offspring \mathcal{Q}_t , with $|\mathcal{Q}_t| = \lambda$
 - 5 Select μ survivors to form \mathcal{P}_{t+1}
 - 6 $t \leftarrow t + 1$
 - 7 **until** *Stop criterion is fulfilled*
-

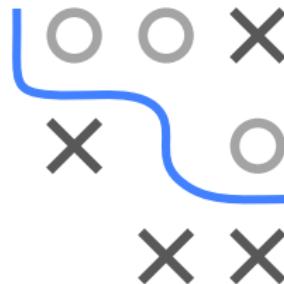


- As in standard EAs, we have parent selection, variation, and survival selection.
- For multi-objective EAs (EMOAs), ranking solutions by “fitness” is no longer straightforward. We need special selection strategies (non-dominated sorting, etc.).

NSGA-II

The **non-dominated sorting genetic algorithm (NSGA-II)** was published by Deb et al. 2002.

- Follows a $(\mu + \lambda)$ strategy for survival.
- Any standard variation operators (like polynomial mutation, SBX) can be used.
- Parent and survival selection both use:
 - ❶ **Non-dominated sorting (NDS)** as the main ranking criterion,
 - ❷ **Crowding distance** as a tie-break.

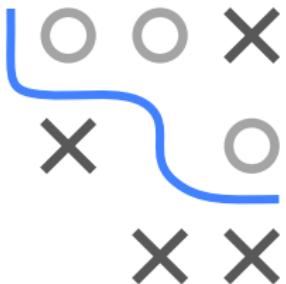
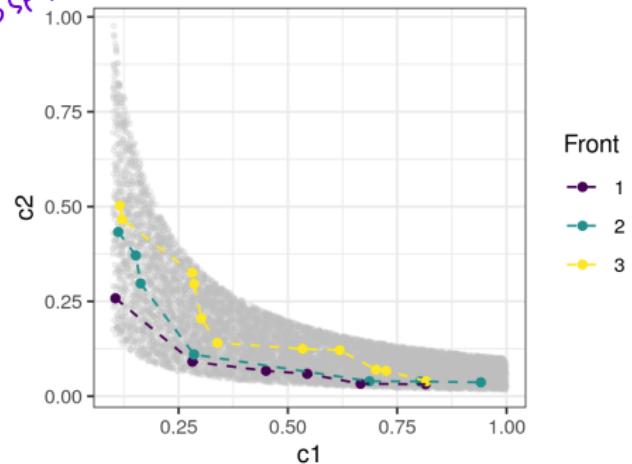


NSGA-II: NON-DOMINATED SORTING (I)

NDS partitions an objective space set into fronts $\mathcal{F}_1 \prec \mathcal{F}_2 \prec \mathcal{F}_3 \prec \dots$

- \mathcal{F}_1 is non-dominated, each $x \in \mathcal{F}_2$ is dominated, but only by points in \mathcal{F}_1 , each $x \in \mathcal{F}_3$ is dominated, but only by points in \mathcal{F}_1 and \mathcal{F}_2 , and so on.
- We can easily compute the partitioning by computing all non-dominated points \mathcal{F}_1 , removing them, then computing the next layer of non-dominated points \mathcal{F}_2 , and so on.

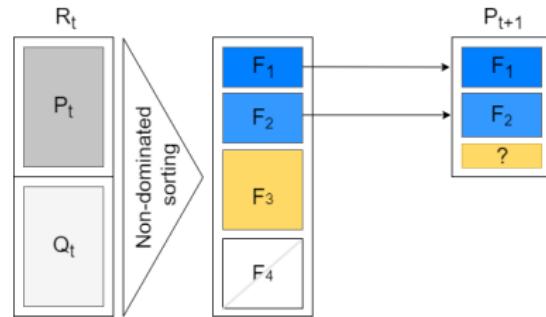
partition into subf. (layers)



NSGA-II: NON-DOMINATED SORTING (II)

For **survival** in $(\mu + \lambda)$ selection:

- We rank all $\mu + \lambda$ individuals by front index.
- We start filling the next population from \mathcal{F}_1 upward.
- If front \mathcal{F}_i cannot fully fit (because we would exceed μ), we partially fill with a subset of \mathcal{F}_i .

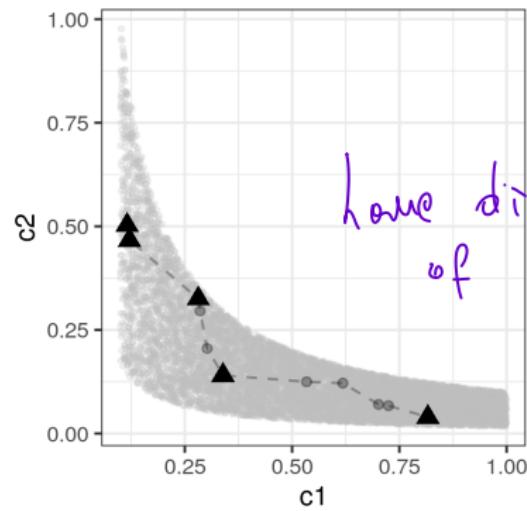
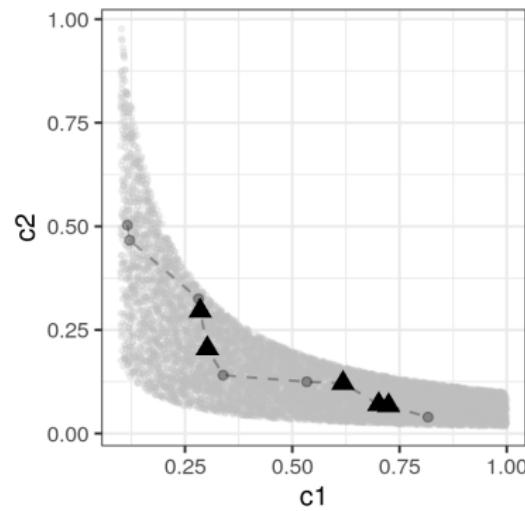


But: Which individuals survive from this partially included front? →
crowding distance sorting.

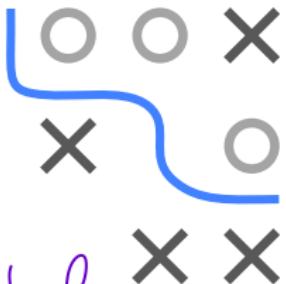
(NDS is also used for parent selection with binary tournaments.)

NSGA-II: CROWDING DISTANCE (I)

Idea: Prefer individuals that are more “spread out” (less crowded).



Left: not good, solutions bunched up. Right: better spread.



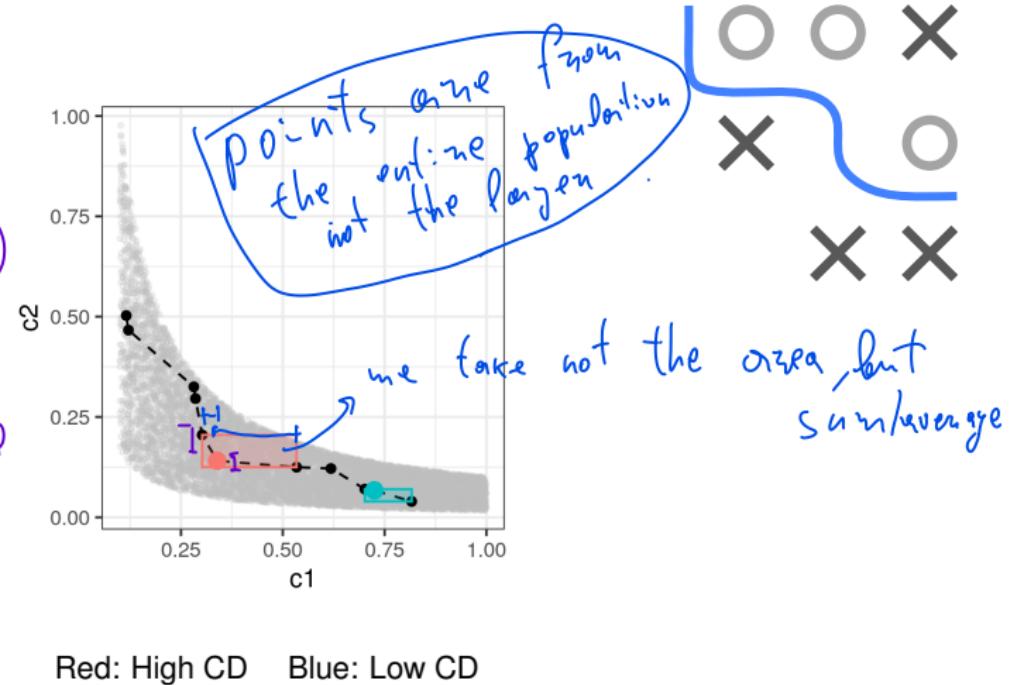
NSGA-II: CROWDING DISTANCE (II)

Crowding distance:

For each objective f_j :

- Sort points by f_j (in ascending order).
- Normalize f_j values to $[0, 1]$. (in max)
- “Border” points (lowest/highest f_j) get infinite distance (ensuring they always survive if possible). (always keep those)
- For internal points, distance is difference in f_j -values between next neighbors.

Then sum (or average) across all m objectives to get the final crowding score.



HYPERVOLUME CONTRIBUTION

(SMS)

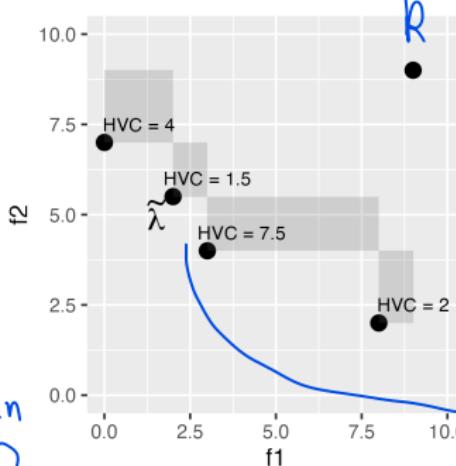
S-Metric-Selection uses the *hypervolume contribution* $\Delta s(\mathbf{x}, \mathcal{P})$ of each individual:

(HVC)

$$\Delta s(\mathbf{x}, \mathcal{P}) = S(\mathcal{P}, R) - S(\mathcal{P} \setminus \{\mathbf{x}\}, R),$$

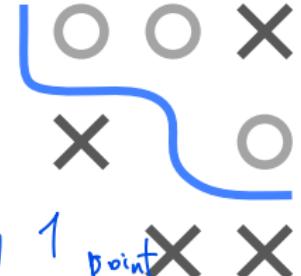
where $S(\cdot, R)$ is the dominated HV w.r.t. reference point R .

- Dark rectangles: unique HV part dominated only by that dot
- Gray cross: reference point R .
- The individual with the smallest HV contribution is “least important” for covering objective space.
- This is used by the SMS-EMOA to remove individuals in survival selection.

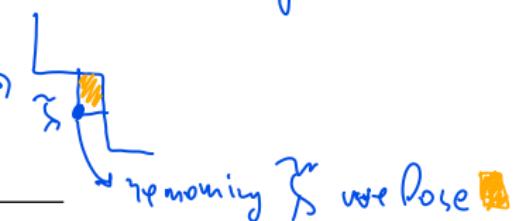


(Geman
alg.)

$\mu + 1$
only.
produces only 1 point



★ contribution of
one point to the front
(how much volume it
brings)

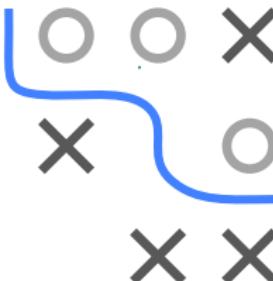


SMS-EMOA ALGORITHM

Algorithm 2: SMS-EMOA

```
1 Generate initial population  $\mathcal{P}_0$  of size  $\mu$ 
2  $t \leftarrow 0$ 
3 repeat
4   Create exactly one offspring  $\mathbf{q}$  by recombination + mutation
5   Perform non-dominated sorting on  $(\mathcal{P}_t \cup \{\mathbf{q}\})$ 
6   Let  $\mathcal{F}_k$  be the last front
7    $\tilde{\mathbf{x}} \leftarrow \arg \min_{\mathbf{x} \in \mathcal{F}_k} \Delta s(\mathbf{x}, \mathcal{F}_k)$ 
8    $\mathcal{P}_{t+1} \leftarrow (\mathcal{P}_t \cup \{\mathbf{q}\}) \setminus \{\tilde{\mathbf{x}}\}$ 
9    $t \leftarrow t + 1$ 
10 until Termination
```

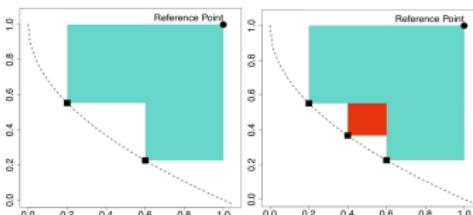
- As soon as we exceed size $\mu + 1$, remove the worst HV contributor from the *last front*.
- That way, each iteration keeps exactly μ individuals.
- Ranking is primarily by front membership; hypervolume contrib. is the tiebreak.



for last layer we use HVC
as a tiebreaker
Adding more than 1 point will make
deciding which points to remove
an NP hard problem

Optimization in Machine Learning

Multi-criteria Optimization Bayesian Optimization



Learning goals

- Single-criteria recap
- Multi-criteria introduction
- Scalarization vs. HV-based methods

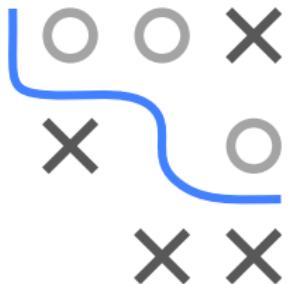


RECAP: BAYESIAN OPTIMIZATION

Advantages of BO

- Sample efficient
- Can handle noise
- Native incorporation of priors
- Does not require gradients
- Theoretical guarantees

We will now extend BO to multiple cost functions.



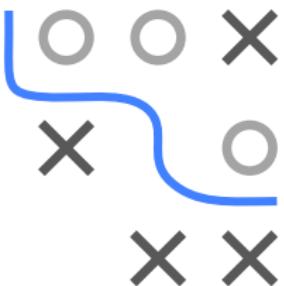
RECAP: BAYESIAN OPTIMIZATION

Algorithm 0: Bayesian optimization loop

Require: Search space \mathcal{S} , cost function f , acquisition function u , predictive model \hat{f} , maximal number of function evaluations T

Result : Best configuration $\hat{\mathbf{x}}$ (based on \mathcal{D} or \hat{f})

- 1 Initialize data $\mathcal{D}^{(0)}$ with initial observations
 - 2 **for** $t \leftarrow 1$ **to** T **do**
 - 3 Fit predictive model $\hat{f}^{(t)}$ on $\mathcal{D}^{(t-1)}$
 - 4 Select next query point:
$$\mathbf{x}^{(t)} \in \arg \max_{\mathbf{x} \in \mathcal{S}} u(\mathbf{x}; \mathcal{D}^{(t-1)}, \hat{f}^{(t)})$$
 - 5 Query $f(\mathbf{x}^{(t)})$
 - 6 Update data:
$$\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \mathbf{x}^{(t)}, f(\mathbf{x}^{(t)}) \rangle\}$$
-



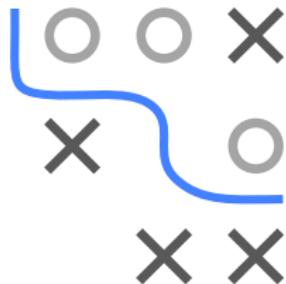
MULTI-CRITERIA BAYESIAN OPTIMIZATION

Goal: Extend Bayesian optimization to multiple cost functions

$$\min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) \Leftrightarrow \min_{\mathbf{x} \in \mathcal{S}} (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})).$$

There are two basic approaches:

- ① Simplify the problem by scalarizing the cost functions, or
- ② define acquisition functions for multiple cost functions.

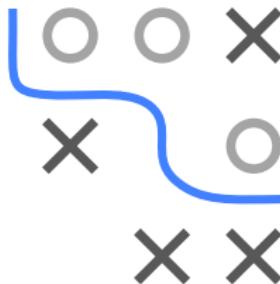


SCALARIZATION

Idea: Aggregate all cost functions

$$\min_{\mathbf{x} \in \mathcal{S}} \sum_{i=1}^m w_i f_i(\mathbf{x}) \quad \text{with } w_i \geq 0$$

- **Obvious problem:** How to choose w_1, \dots, w_m ?
 - Expert knowledge?
 - Systematic variation?
 - Random variation?
- If expert knowledge is not available a-priori, we need to ensure that different trade-offs between cost functions are explored.
- Simplifies multi-criteria optimization problem to single-objective
→ standard Bayesian optimization can be used without adapting the general algorithm.

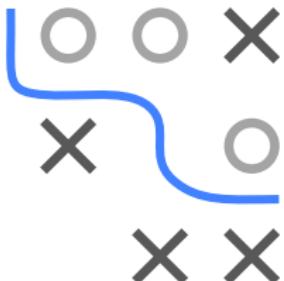


SCALARIZATION: PAREGO ▶ Knowles 2006

Scalarize the cost functions using the augmented Chebychev norm / achievement function

EGO → expensive
global
optim

?



$$f = \max_{i=1,\dots,m} (w_i f_i(\mathbf{x})) + \rho \sum_{i=1}^m w_i f_i(\mathbf{x}),$$

- The weights $w \in W$ are drawn from

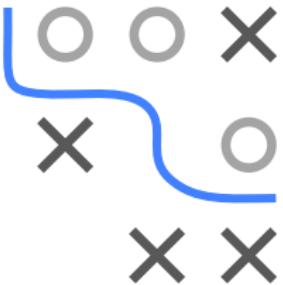
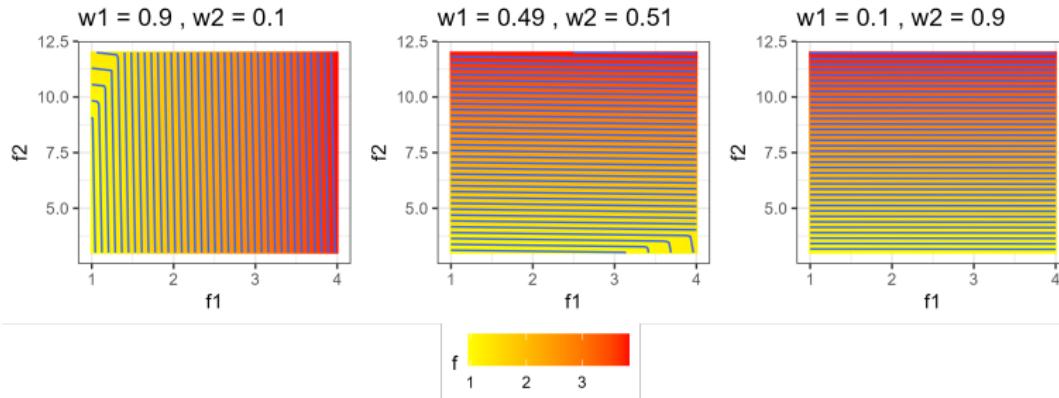
$$W = \{w = (w_1, \dots, w_m) \mid \sum_{i=1}^m w_i = 1, w_i = \frac{l}{s}, l \in \{0, \dots, s\}\},$$

with $|W| = \binom{s+m-1}{m-1}$.

- New weights are drawn in every BO iteration.
- ρ is a small parameter (e.g. 0.05).
- s controls how many distinct weight vectors exist.

Weights are randomly varied
We do one iter of single point
BO

WHY THE CHEBYCHEV NORM?



$$f = \max_{i=1,\dots,m} (w_i f_i(\mathbf{x})) + \rho \sum_{i=1}^m w_i f_i(\mathbf{x}),$$

- The norm consists of two components:
 - $\max_i (w_i f_i(\mathbf{x}))$ considers only the maximum weighted cost.
 - $\sum_i w_i f_i(\mathbf{x})$ is the weighted sum of all costs.
- ρ controls the trade-off between these parts.
- By randomizing the weights w in each iteration (and keeping ρ small), extreme points in single cost functions can be explored.
- One can prove that every solution of this scalarized problem is Pareto-optimal. (Hm)

But not every point in the front can be found from β_0 with epsilon

PAREGO ALGORITHM

ParEGO loop

Require: Search space \mathcal{S} , cost function f , acquisition function u , predictive model \hat{f} , maximal number of function evaluations T , ρ , l , s

Result : Best configuration $\hat{\mathbf{x}}$ (according to \mathcal{D} or \hat{f})

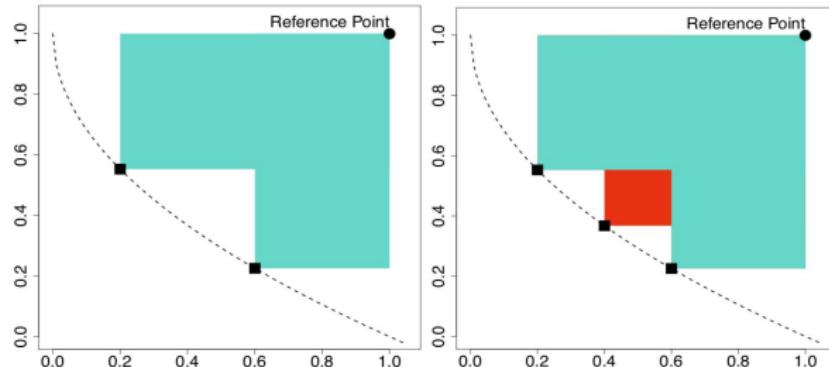
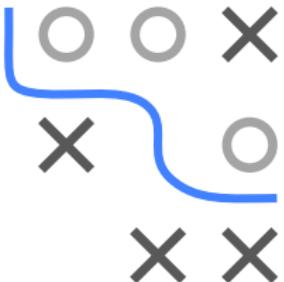
- 1 Initialize data $\mathcal{D}^{(0)}$ with initial observations
- 2 **for** $t = 1$ **to** T **do**
- 3 Sample $w \in W = \{(w_1, \dots, w_m) \mid \sum w_i = 1, w_i = \frac{l}{s}, l \in \{0, \dots, s\}\}$
- 4 Compute scalarization: $c^{(t)} = \max_i(w_i f_i(\mathbf{x})) + \rho \sum_i w_i f_i(\mathbf{x})$
- 5 Fit predictive model $\hat{f}^{(t)}$ on $\mathcal{D}^{(t-1)}$
- 6 Select next query point: $\mathbf{x}^{(t)} \in \arg \max_{\mathbf{x} \in \mathcal{S}} u(\mathbf{x}; \mathcal{D}^{(t-1)}, \hat{f}^{(t)})$
- 7 Query $f(\mathbf{x}^{(t)})$
- 8 Update data:
$$\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \mathbf{x}^{(t)}, f(\mathbf{x}^{(t)}) \rangle\}$$



HV BASED ACQUISITION FUNCTIONS

Idea: Define acquisition function that directly models the contribution to the dominated hypervolume (HV):

$$\max(0, S(\mathcal{P} \cup \mathbf{x}, R) - S(\mathcal{P}, R))$$

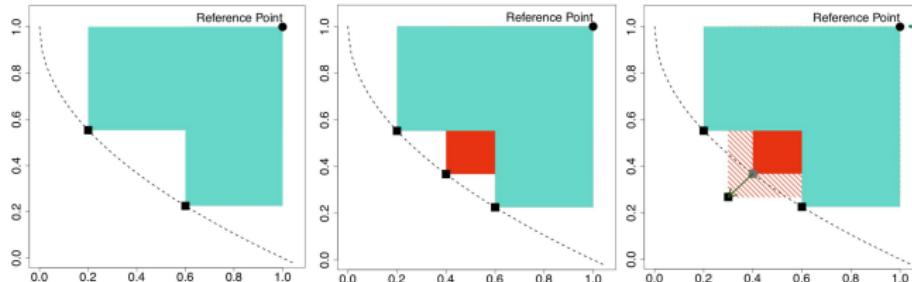


- Fit m single-objective surrogate models $\hat{f}_1, \dots, \hat{f}_m$.
- Acquisition function uses all surrogate models simultaneously.
- Still a single-objective optimization problem (of acquisition itself).

S-METRIC SELECTION-BASED EGO

Using the Lower Confidence Bound $u_{LCB,1}(\mathbf{x}), \dots, u_{LCB,m}(\mathbf{x})$, one can form an *optimistic* estimate of the hypervolume contribution:

we want to incorporate the uncertainty



if points are in the back, then we won't have any HVC so we need to figure out what to do

Problem: HV-contribution can be zero across large regions where the lower confidence bound is already dominated.

- Large zero-plateaus make optimization of the acquisition harder.
- SMS-EGO adds an adaptive penalty in such dominated regions.

This method is referred to as ▶ Ponweiser et al. 2008

FURTHER HV ACQUISITION FUNCTIONS

Expected Hypervolume Improvement (EHI)

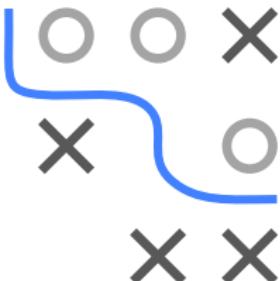
▶ Yang et al. 2019 :

$$u_{EI,\mathcal{H}}(\mathbf{x}) = \int_{-\infty}^{\infty} p(f \mid \mathbf{x}) \times [S(\mathcal{P} \cup \mathbf{x}, R) - S(\mathcal{P}, R)] df$$

- Direct extension of u_{EI} to the hypervolume criterion.
- $p(f \mid \mathbf{x})$ is the joint predictive density of all surrogate models at \mathbf{x} .
- With independent GPs per objective, this factorizes over m univariate normals.
- Exact integral feasible for $m \leq 3$, else simulation-based.

Further HV-based acquisitions:

- **Stepwise Uncertainty Reduction (SUR)** w.r.t. the probability of improvement.
- **Expected Maximin Improvement (EMI)** w.r.t. the ϵ -indicator.



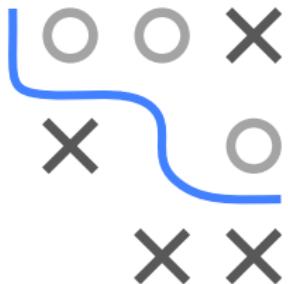
HYPERVOLUME BASED BO ALGORITHM

Hypervolume-based Bayesian optimization loop

Require: Search space \mathcal{S} , cost function f , acquisition function u , predictive model(s) $\hat{f}_1, \dots, \hat{f}_m$, maximal number of function evaluations T

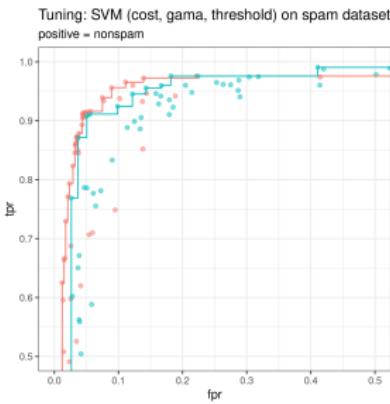
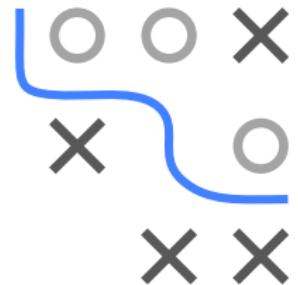
Result : Best configuration $\hat{\mathbf{x}}$

- 1 Initialize data $\mathcal{D}^{(0)}$ with initial observations
- 2 **for** $t = 1$ to T **do**
- 3 Fit predictive models $\hat{f}_1^{(t)}, \dots, \hat{f}_m^{(t)}$ on $\mathcal{D}^{(t-1)}$
- 4 Select next query point:
$$\mathbf{x}^{(t)} \in \arg \max_{\mathbf{x} \in \mathcal{S}} u(\mathbf{x}; \mathcal{D}^{(t-1)}, \hat{f}_1^{(t)}, \dots, \hat{f}_m^{(t)})$$
- 5 Query $f(\mathbf{x}^{(t)})$
- 6 Update data:
$$\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{(\mathbf{x}^{(t)}, f(\mathbf{x}^{(t)}))\}$$



Optimization in Machine Learning

Multi-criteria Optimization Practical Applications



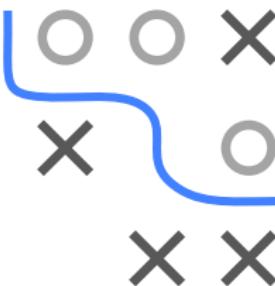
Learning goals

- ROC optimization
- Efficient models
- Fairness considerations

PRACTICAL APPLICATIONS IN MACHINE LEARNING

ROC Optimization: Balance *true positive* and *false positive* rates

- Typically used in unbalanced classification with unspecified costs.
- Could also consider other ROC metrics, e.g., *positive predicted value* or *false discovery rate*



Efficient Models: Balance *predictive performance* with *prediction time*, *energy consumption*, and/or *model size*.

- Time: production models need to predict fast.
- Size / Energy: model might be deployed on a mobile/edge device with power constraints

Sparse Models: Balance *predictive performance* and *number of used features*—for cost efficiency or interpretability.

Fair Models: Balance *predictive performance* and *fairness*.

- Model should not disadvantage certain subgroups (e.g. by gender).
- Multiple approaches exist to quantify fairness

ROC OPTIMIZATION - SETUP

Again, we want to train a *spam detector* on the popular Spam dataset¹.

- Learner: SVM with RBF kernel

- Hyperparameters:

$$\text{cost} \quad [2^{-15}, 2^{15}] \rightarrow ?$$

$$\gamma \quad [2^{-15}, 2^{15}]$$

$$\text{Threshold } t \quad [0, 1]$$

- Objective: *minimize* false positive rate (FPR) and *maximize* true positive rate (TPR), via 5-fold CV

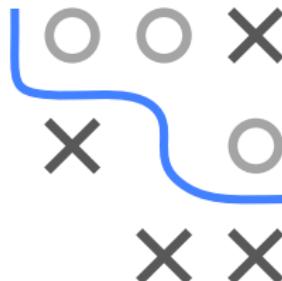
- Optimizer: multi-criteria Bayesian optimization

- ParEGO with $\rho = 0.05$, $s = 100000$

- Acquisition: Confidence Bound ($\alpha = 2$)
- Budget: 100 evaluations

- Tuning is done on a training holdout, and the hyperparameter configs on the estimated Pareto front are validated on external test set.

The threshold t could also be optimized post-hoc separately.



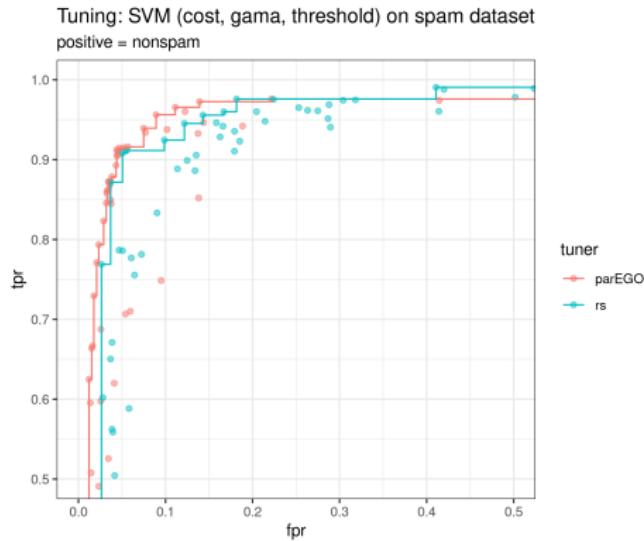
¹<https://archive.ics.uci.edu/ml/datasets/spambase>

ROC OPTIMIZATION - RESULT I

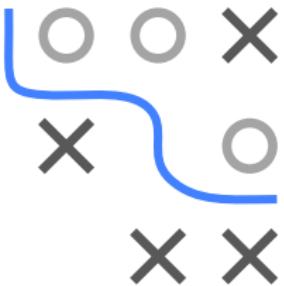
- Compared to *random search*, many *ParEGO* evaluations lie on the Pareto front.
- The *ParEGO* Pareto front dominates most random-search solutions.
- Dominated hypervolume (w.r.t. reference point $(0, 1)$):

ParEGO: 0.965

random: 0.959



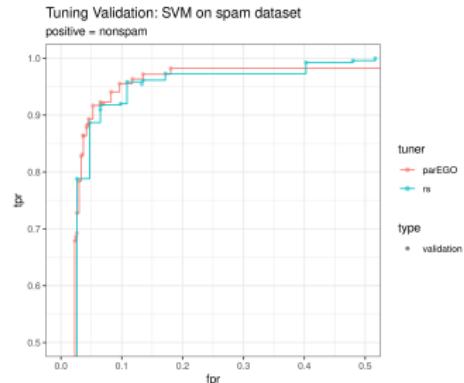
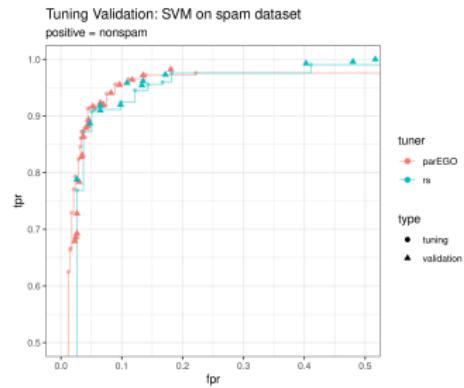
shouldn't this be
(1,0)?



ROC OPTIMIZATION - RESULT II

Validate configurations on the estimated Pareto front:

- The performance changes slightly on a new holdout set.
- TPR improved a bit, but FPR got slightly worse.
- Some configs become dominated once tested on the new holdout.
- Dominated hypervolume on the validation set:
ParEGO: 0.960
random: 0.961



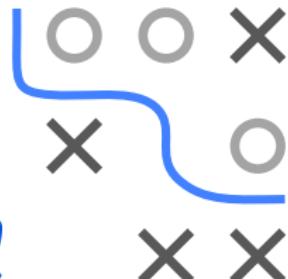
EFFICIENT MODELS - OVERVIEW

- “Efficiency” can refer to:
 - Memory consumption
 - Training or prediction time
 - Number of features used
 - Energy consumption
 - ...
- Some hyperparameters strongly impact efficiency:
 - Number of trees (RF, GBDT)
 - Number, size, type of layers (NN)
 - L1 regularization strength
- Others might have no direct influence on efficiency.
- Typical scenario: searching over multiple algorithms with different cost–performance trade-offs.



EFFICIENT MODELS - EXAMPLE: FEATURE SELECTION I

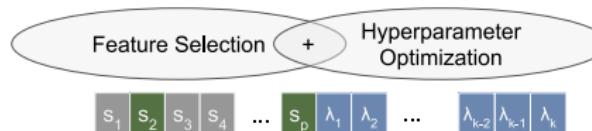
Goal: Identify an informative subset of features with minimal drop in performance vs. using all features.



$$\min_{\mathbf{x} \in \mathcal{S}, s \in \{0,1\}^p} \left(\widehat{GE}(\mathcal{I}(\mathcal{D}, \mathbf{x}, s)), \frac{1}{p} \sum_{i=1}^p s_i \right).$$

fraction of feat. used

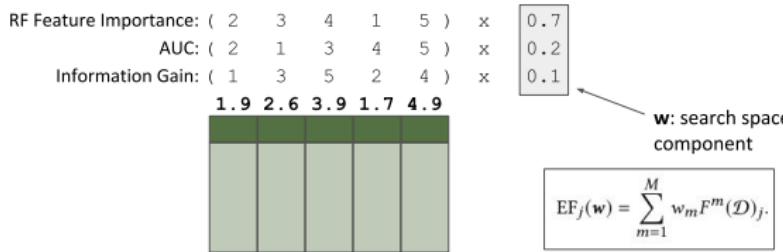
- Typically, feature selection and hyperparameter tuning happen in separate steps.
- Alternatively, search a good subset s and hyperparams \mathbf{x} simultaneously.



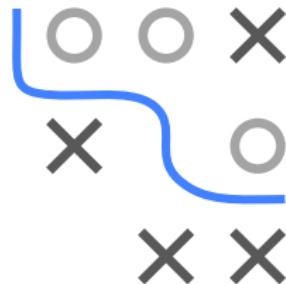
EFFICIENT MODELS - EXAMPLE: FEATURE SELECTION II

Idea: Multi-Objective Hyperparameter Tuning and Feature Selection using Filter Ensembles ▶ Binder et al. 2020.

- Pre-calculate multiple *feature-filter* rank vectors.



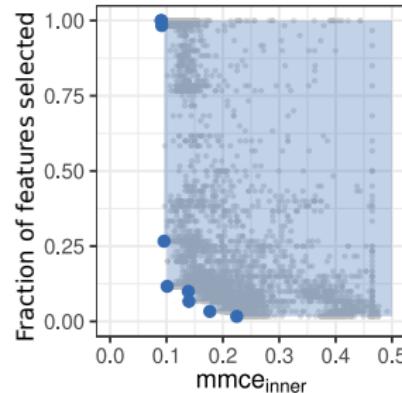
- New hyperparameter vector: $\mathbf{x} = (\tilde{\mathbf{x}}, w_1, \dots, w_p, \tau)$
 - $\tilde{\mathbf{x}}$: learner hyperparameters
 - (w_1, \dots, w_p) : weights for each filter ranking
 - $\tau \in [0, 1]$: fraction of features to keep



EFFICIENT MODELS - EXAMPLE: FEATURE SELECTION III

Joint FS + HP search on Sonar data²:

- Learner: SVM (RBF kernel).
- Hyperparams:
 - cost $[2^{-10}, 2^{10}]$
 - γ $[2^{-10}, 2^{10}]$
 - (w_1, \dots, w_p) $[0, 1]^p$
 - τ $[0, 1]$
- Objective: minimize misclassification + fraction of selected features.
- Optimizer: ParEGO + random forest surrogate + LCB acquisition, 15 batch proposals, budget 2000 evals.



²Only tuning error is shown

EFFICIENT MODELS - EXAMPLE: FLOPS

Goal: Optimize *accuracy* vs. *FLOPS* (floating-point ops)

► Wang et al. 2019

Data: CIFAR-10 (image classification).

Learner: *DenseNet* ► Huang et al. 2017 :

- 4 dense blocks, each with multiple convolution layers that feed into each other.
- Blocks connected by convolution + max-pooling layers.

Training: 300 epochs, batch size 128, initial LR 0.1.



EFFICIENT MODELS - EXAMPLE: FLOPS (CONT.)

- Objectives: *accuracy vs. FLOPS per observation.*

- Search space:

growth rate (k) [8, 32]

layers in 1st block [4, 6]

layers in 2nd block [4, 12]

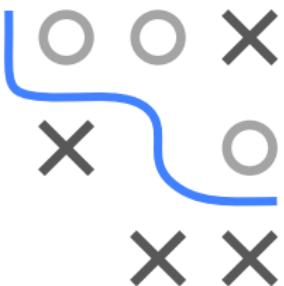
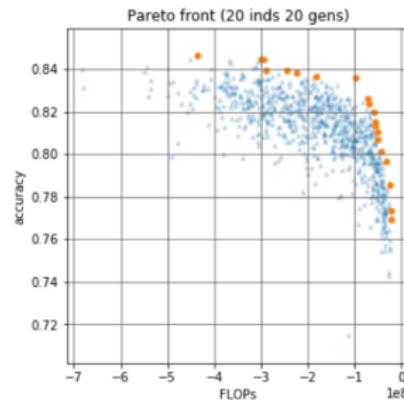
layers in 3rd block [4, 24]

layers in 4th block [4, 16]

- Tuner: *Particle Swarm Optimization,*

population 20, budget 400.

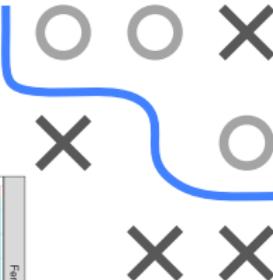
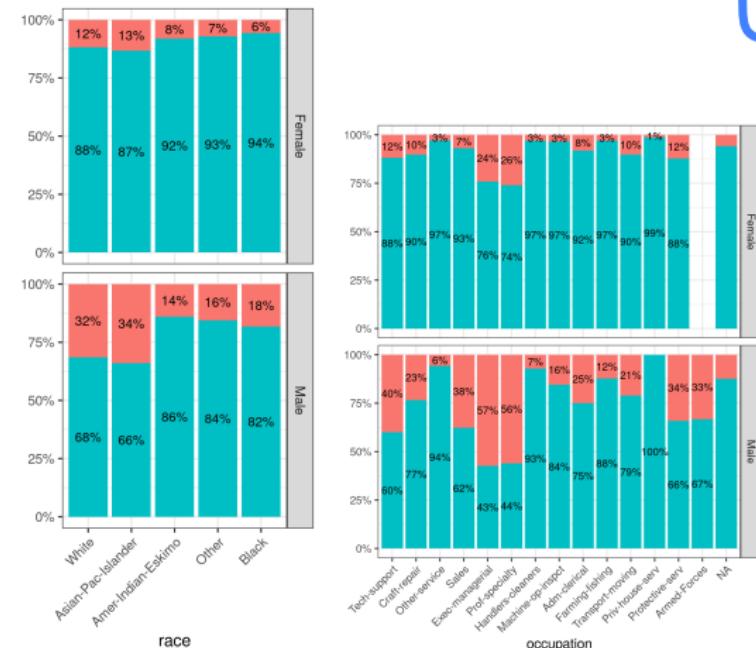
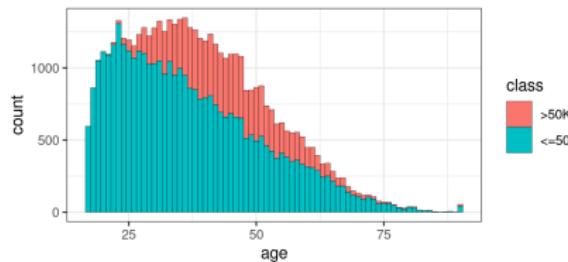
“Growth rate” = # of feature maps added each layer.



FAIR MODELS - THE ADULT DATASET

Dataset: Adult

- Source: US Census database, 1994, <https://www.openml.org/d/1590>.
- 48842 observations
- Target: binary, income above 50k
- 14 features: age, education, hours.per.week, marital.status, native.country, occupation, race, relationship, sex, ...



FAIR MODELS - SETUP

A toy example: build a “fair” model for predicting income (binary).

- Learner: XGBoost

eta	[0.01, 0.2]
gamma	$[2^{-7}, 2^6]$
max_depth	{2, ..., 20}
colsample_bytree	[0.5, 1]
colsample_bylevel	[0.5, 1]
lambda	$[2^{-10}, 2^{10}]$
alpha	$[2^{-10}, 2^{10}]$
subsample	[0.5, 1]

- Hyperparams:

colsample_bytree	[0.5, 1]
colsample_bylevel	[0.5, 1]
lambda	$[2^{-10}, 2^{10}]$
alpha	$[2^{-10}, 2^{10}]$
subsample	[0.5, 1]

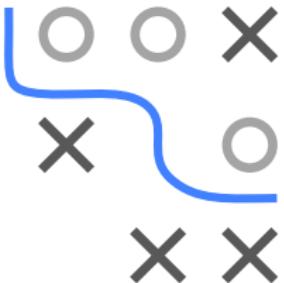
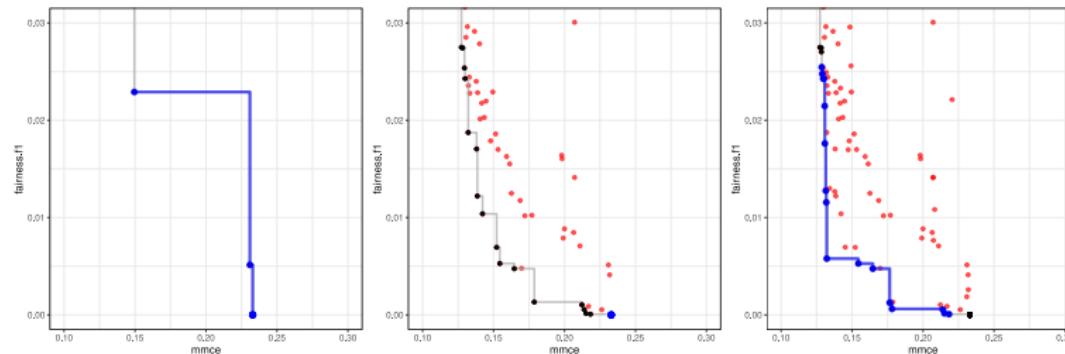
- Objective: *misclassification error vs. unfairness.*
- We define “unfairness” as the absolute difference in F1-scores between male/female sub-populations:

$$L_{\text{fair}} := \left| \text{F1}(y_f, \hat{f}(\mathbf{x}_f)) - \text{F1}(y_m, \hat{f}(\mathbf{x}_m)) \right|.$$

Note: This is a simplistic example. Real fairness questions often need more careful definitions and domain knowledge.



FAIR MODELS - RESULTS



- Optimizer: ParEGO + random forest surrogate, restricting $[w_i]$ to $[0.1, 0.9]$ if we only care about moderate fairness/performance ranges.
- In this example, hyperparams do indeed influence the fairness measure.
- But in practice, changing standard HPs often *won't* suffice to ensure fairness.