Multi-criteria Optimization

Bayesian Optimization

Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Recap: Bayesian Optimization I

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 II

Bayesian optimization loop

Require: Search space $\boldsymbol{\Lambda}$, cost function c, acquisition function u, predictive model $\hat{c}$, maximal number of function evaluations T

Result : Best configuration $\hat{\boldsymbol{\lambda}}$ (according to $\mathcal{D}$ or $\hat{c}$)

```
Initialize data D^(0) with initial observations
for t = 1 to T do
    Fit predictive model ĉ^(t) on D^(t−1)
    Select next query point: λ^(t) ∈ arg max_{λ∈Λ} u(λ; D^(t−1), ĉ^(t))
    Query c(λ^(t))
    Update data: D^(t) ← D^(t−1) ∪ {⟨λ^(t), c(λ^(t))⟩}
```

Initialize data $\mathcal{D}^{(0)}$ with initial observations

for $t = 1$ **to** T **do**

- Fit predictive model $\hat{c}^{(t)}$ on $\mathcal{D}^{(t-1)}$
- Select next query point: $\boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} u(\boldsymbol{\lambda}; \mathcal{D}^{(t-1)}, \hat{c}^{(t)})$
- Query $c(\boldsymbol{\lambda}^{(t)})$
- Update data: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \boldsymbol{\lambda}^{(t)}, c(\boldsymbol{\lambda}^{(t)}) \rangle\}$

Multi-Criteria Bayesian Optimization

Goal: Extend Bayesian optimization to multiple cost functions

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\boldsymbol{\lambda}) \Leftrightarrow \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \left(c_1(\boldsymbol{\lambda}), c_2(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda})\right).$$

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

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda}) \qquad \text{with} \quad w_i \geq 0$$

- **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
  $\longrightarrow$ Bayesian optimization can be used without adaption of the general algorithm.

Scalarization: ParEGO [Knowles. 2006]

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

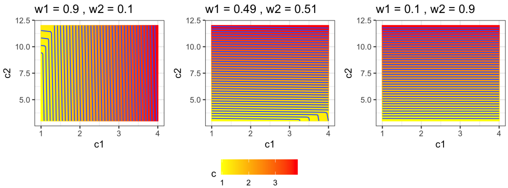
$$c = \max_{i=1,\ldots,m} (w_i c_i(\boldsymbol{\lambda})) + \rho \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda}),$$

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

$$W = \left\{ w = (w_1, \ldots, w_m) | \sum_{i=1}^{m} w_i = 1, w_i = \frac{l}{s} \wedge, l \in 0, \ldots, s \right\},$$

  with $|W| = \binom{s+m-1}{k-1} 1$.
- New weights are drawn in every BO iteration.
- ρ is a small parameter suggested to be set to 0.05.
- s selects the number of different weights to draw from.

Why the Tchebycheff norm?



$$c = \max_{i=1,\ldots,m} (w_i c_i(\boldsymbol{\lambda})) + \rho \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda}),$$

- The norm consists of two components:
  - $\max_{i=1,\ldots,m} (w_i c_i(\boldsymbol{\lambda}))$ takes only the maximum weighted cost into account.
  - $\sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda})$ is the weighted sum of all cost functions.
- ρ describes the trade-off between these components.
- By the randomized weights in each iteration and the usually small value of $\rho = 0.05$, this allows exploration of extreme points of single cost functions.
- One can prove: **Every solution of the scalarized problem is pareto-optimal!**

ParEGO Algorithm

ParEGO loop

Require: Search space $\boldsymbol{\Lambda}$, cost function c, acquisition function u, predictive model $\hat{c}$, maximal number of function evaluations T, ρ, l, s

Result : Best configuration $\hat{\boldsymbol{\lambda}}$ (according to $\mathcal{D}$ or $\hat{c}$)

```
Initialize data D^(0) with initial observations
for t = 1 to T do
    Sample w from {w = (w_1, ..., w_m) | Σ_{i=1}^m w_i = 1, w_i = l/s ∧, l ∈ 0, ..., s};
    Compute scalarization c^(t) = max_{i=1,...,m} (w_i c_i(λ)) + ρ Σ_{i=1}^m w_i c_i(λ);
    Fit predictive model ĉ^(t) on D^(t-1)
    Select next query point: λ^(t) ∈ arg max_{λ∈Λ} u(λ; D^(t-1), ĉ^(t))
    Query c(λ^(t))
    Update data: D^(t) ← D^(t-1) ∪ {⟨λ^(t), c(λ^(t))⟩}
```

Initialize data $\mathcal{D}^{(0)}$ with initial observations

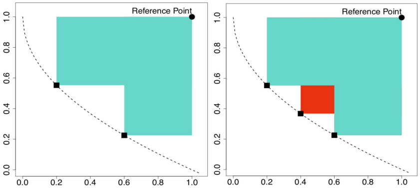
for $t = 1$ **to** T **do**

- Sample w from $\left\{w = (w_1, \ldots, w_m) \middle| \sum_{i=1}^m w_i = 1, w_i = \frac{l}{s} \wedge, l \in 0, \ldots, s\right\}$;
- Compute scalarization $c^{(t)} = \max_{i=1,\ldots,m}\left(w_i c_i(\boldsymbol{\lambda})\right) + \rho \sum_{i=1}^m w_i c_i(\boldsymbol{\lambda})$;
- Fit predictive model $\hat{c}^{(t)}$ on $\mathcal{D}^{(t-1)}$
- Select next query point: $\boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} u(\boldsymbol{\lambda}; \mathcal{D}^{(t-1)}, \hat{c}^{(t)})$
- Query $c(\boldsymbol{\lambda}^{(t)})$
- Update data: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \boldsymbol{\lambda}^{(t)}, c(\boldsymbol{\lambda}^{(t)}) \rangle\}$

Hypervolume based Acquisition Functions

Idea: Define acquisition function that directly models contribution to dominated HV.

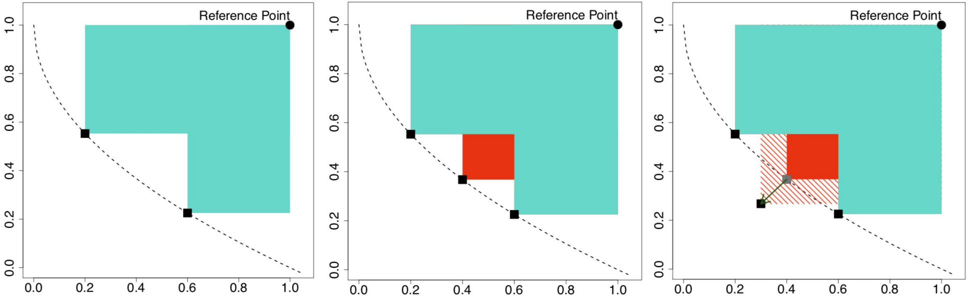
$$\max(0, S(\mathcal{P} \cup \boldsymbol{\lambda}, R) - S(\mathcal{P}, R))$$



- Fit m single-objective surrogate models $\hat{c}_1, \ldots, \hat{c}_m$
- Acquisition function takes all surrogate models into account.
- Single-criteria optimization of acquisition function.

S-Metric Selection-based EGO I

Using the Lower Confidence bound $u_{\text{LCB},1}(\boldsymbol{\lambda}), \ldots, u_{\text{LCB},m}(\boldsymbol{\lambda})$, an optimistic estimate of hypervolume contribution can be calculated.



S-Metric Selection-based EGO II

Problem: Based on the way the hypervolume contribution is measured large plateaus of zero improvement are present.

- These make optimization much harder.
- An adaptive penalty is added to regions in which the lower confidence bound is dominated.

This method is referred to as SMS-EGO [Ponweiser et al. 2008].

Further Hypervolume based Acquisition Functions

Expected Hypervolume Improvement (EHI) [Yang et al. 2019]

$$u_{EI,\mathcal{H}}(\boldsymbol{\lambda}) = \int_{-\infty}^{\infty} p(c \mid \boldsymbol{\lambda}) \times \mathcal{H}(\boldsymbol{\lambda}) \; dc,$$

with $\mathcal{H}(\boldsymbol{\lambda}) = S(\mathcal{P} \cup \boldsymbol{\lambda}, R) - S(\mathcal{P}, R))$.

- Direct extension of u_{EI} to the hypervolume.
- $p(c \mid \boldsymbol{\lambda})$ is the joint density of the surrogate model predictions at $\boldsymbol{\lambda}$.
- As the surrogates are GPs and modeled independently of each other, this is just an integral over m univariate normal distributions.
- Efficient computations for $m \leq 3$ exist, beyond that expensive simulation-based computation is required.

Further hypervolume based acquisition functions:

- **Stepwise Uncertainty Reduction** (SUR) based on the probability of improvement.
- **Expected Maximin Improvement** (EMI) based on the ϵ-indicator.

Hypervolume based BO Algorithm

Hypervolume based Bayesian optimization loop

Require: Search space $\boldsymbol{\Lambda}$, cost function c, acquisition function u, predictive model $\hat{c}$, maximal number of function evaluations T

Result : Best configuration $\hat{\boldsymbol{\lambda}}$ (according to $\mathcal{D}$ or $\hat{c}$)

Initialize data $\mathcal{D}^{(0)}$ with initial observations

for $t = 1$ **to** T **do**

- Fit predictive models $\hat{c}_1^{(t)}, \dots, \hat{c}_m^{(t)}$ on $\mathcal{D}^{(t-1)}$
- Select next query point: $\boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} u(\boldsymbol{\lambda}; \mathcal{D}^{(t-1)}, \hat{c}_1^{(t)}, \dots, \hat{c}_m^{(t)})$
- Query $c(\boldsymbol{\lambda}^{(t)})$
- Update data: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \boldsymbol{\lambda}^{(t)}, c(\boldsymbol{\lambda}^{(t)}) \rangle\}$

Multi-criteria Optimization

Introduction

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Introductory example I

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 II

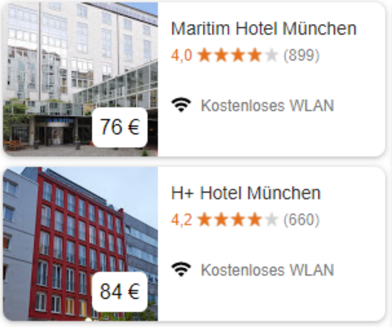
Example:
Choose the best hotel to stay at by maximizing ratings subject to a maximum price per night.

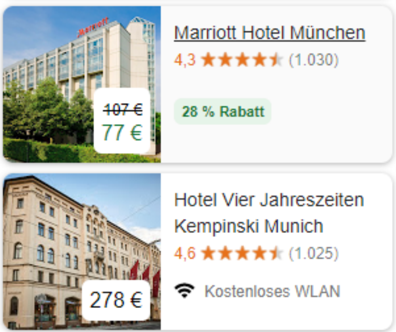
Problems:

- The result depends on how we select the maximum price and usually returns different solutions for different maximum price values.
- We could also choose a minimum rating and optimize the price per night.
- The more objectives we optimize, the more difficult such a definition becomes.

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

Introductory example III





When booking a hotel: find the hotel with

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

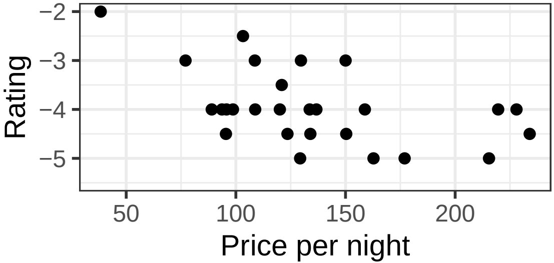
Since our standard is to minimize objectives, we minimize negative ratings.

Introductory example IV

The objectives often conflict with each other:

- Lower price $\rightarrow$ usually lower hotel rating.
- Better rating $\rightarrow$ usually higher price.

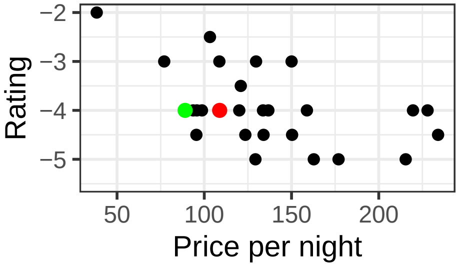
Example: (negative) average rating by hotel guests (1 - 5) vs. average price per night (excerpt).

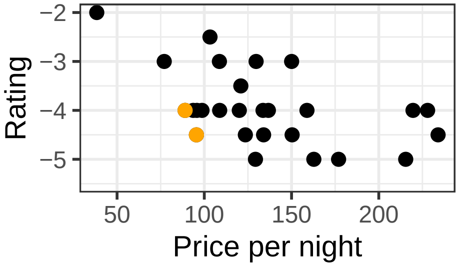


Introductory example V

Often, objectives are not directly comparable as they are measured on different scales:

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





Definition: multi-criteria optimization problem

A **multi-criteria optimization problem** is defined by

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\boldsymbol{\lambda}) \Leftrightarrow \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \left(c_1(\boldsymbol{\lambda}), c_2(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda})\right),$$

with $\boldsymbol{\Lambda} \subset \mathbb{R}^n$ and multi-criteria objective function $c : \boldsymbol{\Lambda} \to \mathbb{R}^m$, $m \geq 2$.

- **Goal:** minimize multiple target functions simultaneously.
- $(c_1(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda}))^\top$ maps each candidate $\boldsymbol{\lambda}$ into the objective space $\mathbb{R}^m$.
- Often no clear best solution, as objective are usually conflicting and we cannot totally order in $\mathbb{R}^m$.
- W.l.o.g. we always minimize.
- Alternative names: multi-criteria optimization, multi-objective optimization, Pareto optimization.

Pareto sets and Pareto optimality

Definition:
Given a multi-criteria optimization problem

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \left(c_1(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda})\right), \quad c_i : \boldsymbol{\Lambda} \to \mathbb{R}.$$

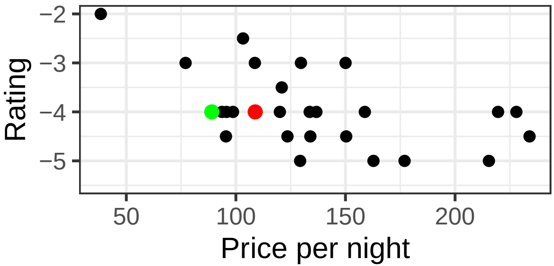
- A candidate $\boldsymbol{\lambda}^{(1)}$ **(Pareto-) dominates** $\boldsymbol{\lambda}^{(2)}$, if $c(\boldsymbol{\lambda}^{(1)}) \prec c(\boldsymbol{\lambda}^{(2)})$, i.e.
  1. $c_i(\boldsymbol{\lambda}^{(1)}) \leq c_i(\boldsymbol{\lambda}^{(2)})$ for all $i \in \{1, 2, ..., m\}$ and
  2. $c_j(\boldsymbol{\lambda}^{(1)}) < c_j(\boldsymbol{\lambda}^{(2)})$ for at least one $j \in \{1, 2, ..., m\}$
- A candidate $\boldsymbol{\lambda}^*$ that is not dominated by any other candidate is called **Pareto optimal**.
- The set of all Pareto optimal candidates is called **Pareto set**
  $\mathcal{P} := \{\boldsymbol{\lambda} \in \boldsymbol{\Lambda} | \; \nexists \; \tilde{\boldsymbol{\lambda}}$ with $c(\tilde{\boldsymbol{\lambda}}) \prec c(\boldsymbol{\lambda})\}$
- $\mathcal{F} = c(\mathcal{P}) = \{c(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \mathcal{P}\}$ is called **Pareto front**.

How to define optimality? I

Let $c = (\text{price}, -\text{rating})$. For some cases it is *clear* which point is the better one:

- The candidate $c^{(1)} = (89, -4.0)$ dominates $c^{(2)} = (108, -4.0)$: $c^{(1)}$ is not worse in any dimension and is better in one dimension. Therefore, $c^{(2)}$ gets **dominated** by $c^{(1)}$

$$c^{(2)} \prec c^{(1)}.$$

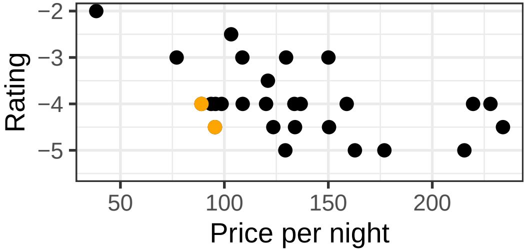


How to define optimality? II

For the points $c^{(1)} = (89, -4.0)$ and $c^{(2)} = (95, -4.5)$ we cannot say which one is better.

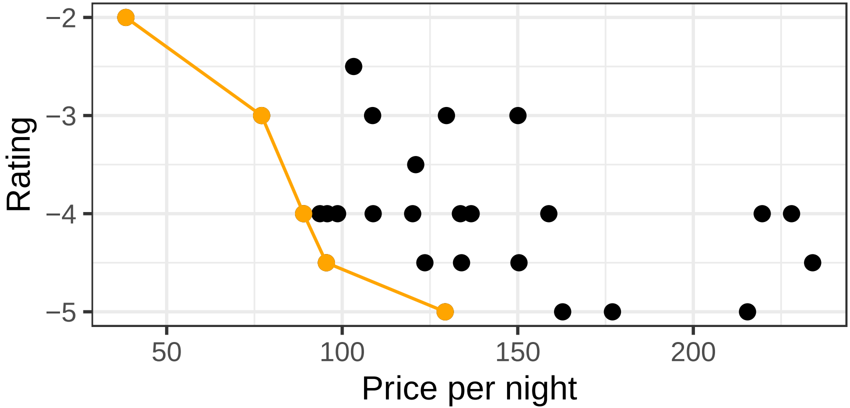
- We define the points as **equivalent** and write

$$c^{(1)} \not\prec c^{(2)} \text{ and } c^{(2)} \not\prec c^{(1)}.$$



How to define optimality? III

- The set of all equivalent points that are not dominated by another point is called the **Pareto front**.

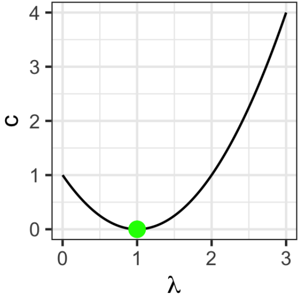


Example: One objective function

We consider the minimization problem

$$\min_{\boldsymbol{\lambda}} c(\boldsymbol{\lambda}) = (\boldsymbol{\lambda} - 1)^2, \qquad 0 \leq \boldsymbol{\lambda} \leq 3.$$

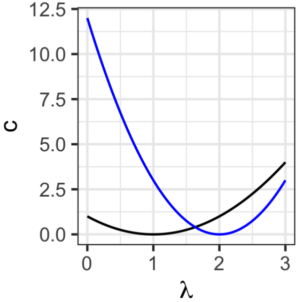
The optimum is at $\boldsymbol{\lambda}^* = 1$.



Example: Two target functions I

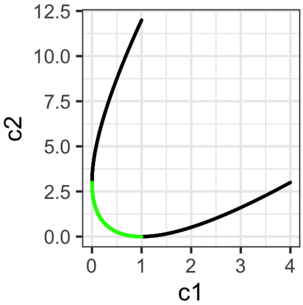
We extend the above problem to two objective functions $c_1(\boldsymbol{\lambda}) = (\boldsymbol{\lambda} - 1)^2$ and $c_2(\boldsymbol{\lambda}) = 3(\boldsymbol{\lambda} - 2)^2$, thus

$$\min_{\boldsymbol{\lambda}} c(\boldsymbol{\lambda}) = \left(c_1(\boldsymbol{\lambda}), c_2(\boldsymbol{\lambda})\right), \qquad 0 \leq \boldsymbol{\lambda} \leq 3.$$



Example: Two target functions II

We consider the functions in the objective function space $c(\boldsymbol{\Lambda})$ by drawing the objective function values $(c_1(\boldsymbol{\lambda}), c_2(\boldsymbol{\lambda}))$ for all $0 \leq \boldsymbol{\lambda} \leq 3$.



The Pareto front is shown in green. The Pareto front cannot be *left* without getting worse in at least one objective function.

A-priori vs. A-posteriori

- The Pareto set is a set of equally optimal solutions.
- In many applications one is often interested in a **single** optimal solution.
- Without further information no unambiguous optimal solution can be determined.
  $\rightarrow$ The decision must be based on other criteria.

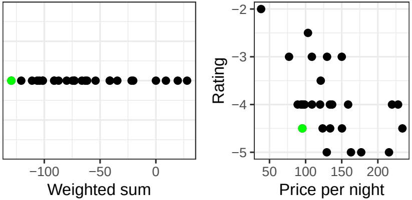
There are two possible approaches:

- **A-priori approach**: User preferences are considered **before** the optimization process
- **A-posteriori approach**: User preferences are considered **after** the optimization process

A-priori procedure I

Example: Weighted total
Prior knowledge: One rating point is worth 50 Euro to a customer.
$\rightarrow$ We optimize the weighted sum:

$$\min_{\text{Hotel}} (\text{Price / Night}) - 50 \cdot \text{Rating}$$



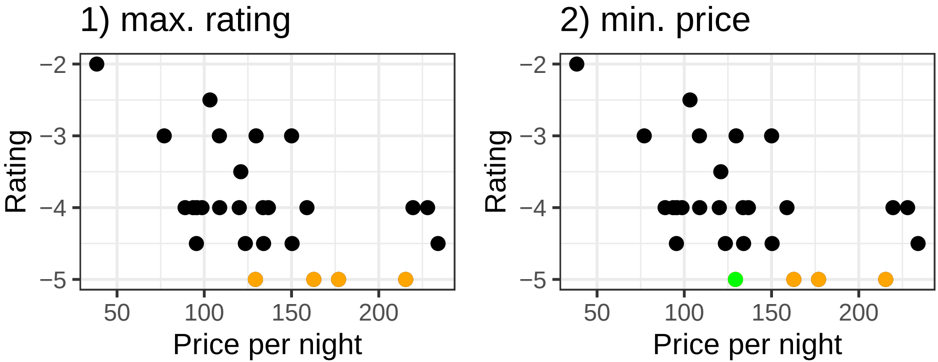
Alternative a weighted sum: $\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda})$ with $w_i \geq 0$

A-priori procedure II

Example: Lexicographic method

Prior knowledge: Customer prioritizes rating over price.

$\rightarrow$ Optimize target functions one after the other.



A-priori procedure III

A-priori approach: Lexicographic method

$$
\begin{aligned}
c_1^* &= \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c_1(\boldsymbol{\lambda}) \\
c_2^* &= \min_{\boldsymbol{\lambda} \in \{\boldsymbol{\lambda} \mid c_1(\boldsymbol{\lambda}) = c_1^*\}} c_2(\boldsymbol{\lambda}) \\
c_3^* &= \min_{\boldsymbol{\lambda} \in \{\boldsymbol{\lambda} \mid c_1(\boldsymbol{\lambda}) = c_1^* \wedge c_2(\boldsymbol{\lambda}) = c_2^*\}} c_3(\boldsymbol{\lambda}) \\
&\vdots
\end{aligned}
$$

But: Different sequences provide different solutions.

A-priori procedure IV

Summary a-priori approach:

- Implicit assumption: Single-objective optimization is *easy*.
- Only one solution is obtained, which depends on a-priori weights, order, etc.
- Several solutions can be obtained if weights, order, etc. are systematically varied.
- Usually not all non-dominated candidates can be found by these methods.

A-posteriori procedure I

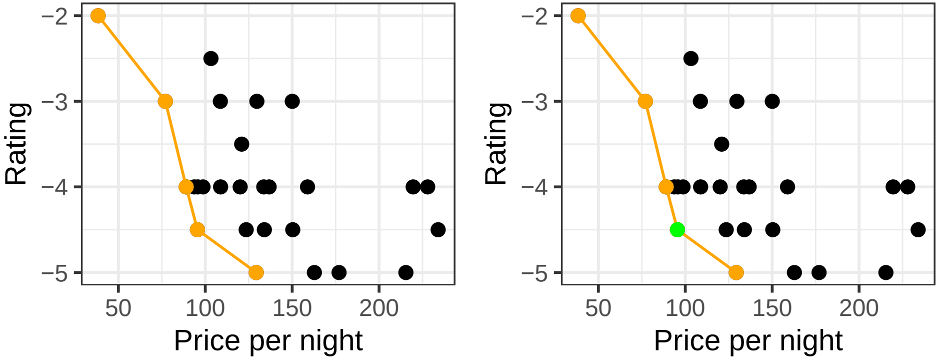
A-posteriori methods try to

- find the set of **all** optimal candidates (the Pareto set),
- select (if necessary) an optimal candidate based on prior knowledge or individual preferences.
- Implicit assumption: Specifying your hidden preferences / making a selection from a pool of candidates is easier, if you see the non-dominated solutions.

A-posteriori methods are therefore the more generic approach to solving a multi-criteria optimization problem.

A-posteriori procedure II

Example: A user is displayed all Pareto optimal hotels (left) and chooses an optimal candidate (right) based on his hidden preferences or additional criteria (e.g. location of the hotel).

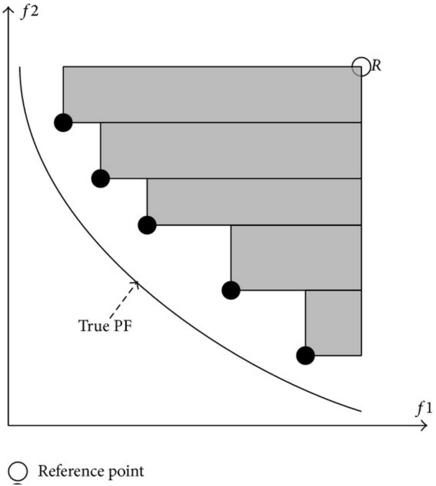


Evaluation of solutions I

A common metric for evaluating the performance of a set of candidates $\mathcal{P} \subset \boldsymbol{\Lambda}$ is the **dominated hypervolume**

$$S(\mathcal{P}, R) = \Lambda\left(\bigcup_{\tilde{\boldsymbol{\lambda}} \in \mathcal{P}} \left\{\boldsymbol{\lambda} | \tilde{\boldsymbol{\lambda}} \prec \boldsymbol{\lambda} \prec R\right\}\right),$$

where Λ is the Lebuesge measure.



Evaluation of solutions II

- HV is calculated w.r.t the reference point R, which often reflects in each component the natural maximum of the respective objective – if possible
- The dominated hypervolume is also often called **S-Metric**.
- Computation of HV scales exponentially in the number of objective functions $\mathcal{O}(n^{m-1})$.
- Fast approximations exist for small values of m and especially for machine learning applications we rarely optimize $m > 3$ objectives.

Multi-criteria Optimization

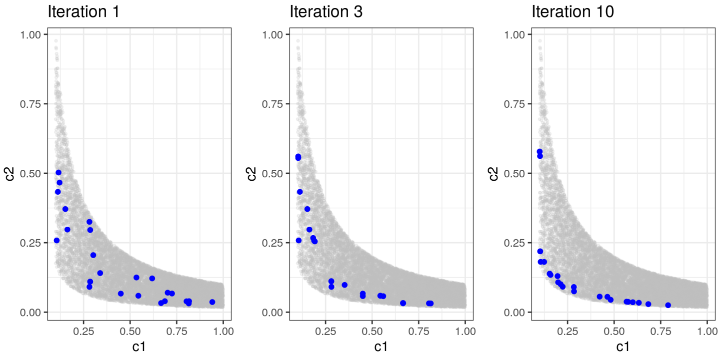
Evolutionary Approaches

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

A-posteriori methods and evolutionary algorithms I

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

Initialize population
↓
Eval population
↓
Parent selection
↓
Variation
↓
no   Eval offspring
↓
Survival selection
↓
Stop
↓ yes



A-posteriori methods and evolutionary algorithms II

Algorithm 1 Basic EA template loop

```
Init and eval population $\mathcal{P}_0 \subset \mathbf{\Lambda}$ with $|\mathcal{P}| = \mu$
$t \leftarrow 0$
repeat
    Select parents and generate offspring $\mathcal{Q}_t$ with $|\mathcal{Q}_t| = \lambda$
    Select $\mu$ survivors $\mathcal{P}_{t+1}$
    $t \leftarrow t + 1$
until Stop criterion fulfilled
```

- Note that (as in the EA lecture unit) we are using somewhat non-standard notation here.
- Nearly all steps in the above template work also for EMOAs but both parent and survival selection are now less obvious. How do we rank under multiple objectives?

NSGA-II

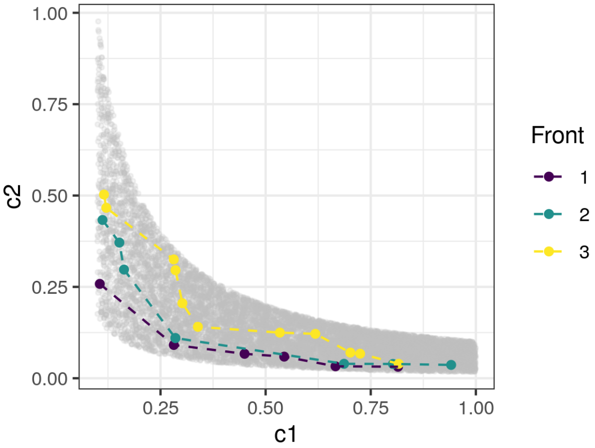
The **non-dominated sorting genetic algorithm (NSGA-II)** was published by [Dep et al. 2002].

- Follows a $(\mu + \lambda)$ strategy.
- All previously discussed variation strategies can be used; the original paper uses tournament selection, polynomial mutation and simulated binary crossover.
- Parent and survival selection rank candidates by
  1. **Non-dominated sorting** as main criterion
  2. **Crowding distance assignment** as tie breaker

NSGA-II: non-dominated sorting I

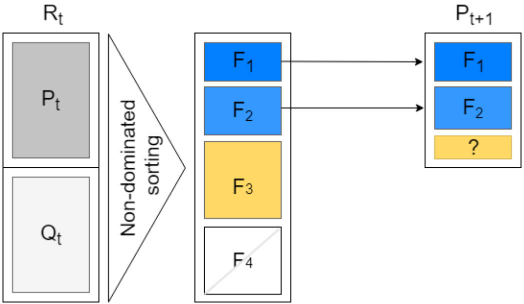
NDS partitions an objective space set into fronts $\mathcal{F}_1 \prec \mathcal{F}_2 \prec \mathcal{F}_3 \prec \dots$.

- $\mathcal{F}_1$ is non-dominated, each $\boldsymbol{\lambda} \in \mathcal{F}_2$ is dominated, but only by points in $\mathcal{F}_1$, each $\boldsymbol{\lambda} \in \mathcal{F}_3$ 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.



NSGA-II: non-dominated sorting II

How does survival selection now work? We fill μ *places* one by one with $\mathcal{F}_1, \mathcal{F}_2, \ldots$ until a front can no longer **fully** survive (here: $\mathcal{F}_3$).

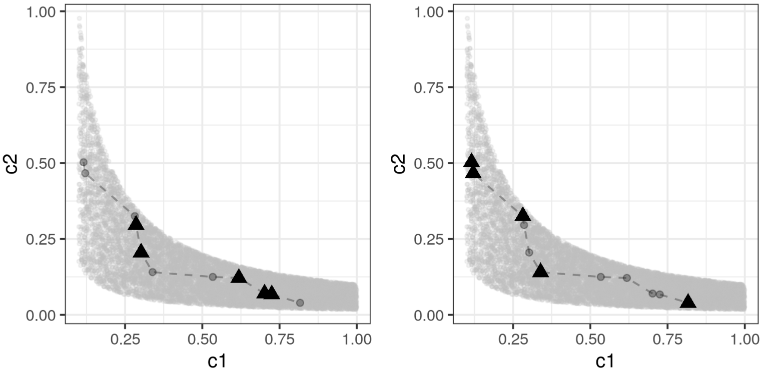


Which individuals survive from $\mathcal{F}_3$? $\rightarrow$ **crowding sort**

NB: the same principle to rank individuals is applied in tournament selection in parent selection.

NSGA-II: crowding distance I

Idea: Add *good* representatives of front $\mathcal{F}_3$, define this as points of "low density" in c-space.



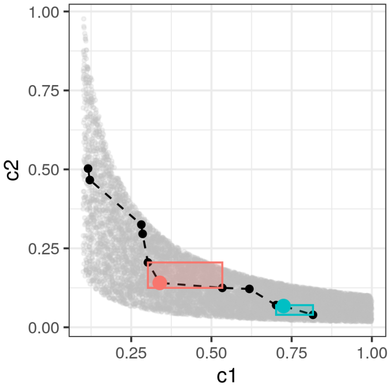
Left: Not good, points very close together. Right: better.

NSGA-II: crowding distance II

For each objective c_j

- Sort points by c_j
- Normalize scores to [0,1]
- Assign border points (which have score 0 or 1) a CD of ∞ (they should always be selected, if possible)
- Each point gets a distance score, which is the distance between its 2 next-neighbors w.r.t. the sorting of c_j

For each point, all of its m distance scores are summed up (or averaged) and points are ranked w.r.t. to this overall score.



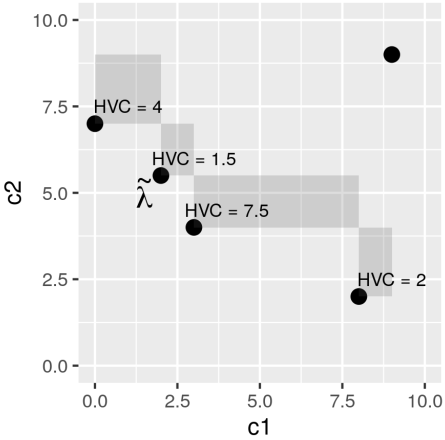
Red: Point with high CD. Blue: Low CD.

Selection criteria: contribution to the hypervolume I

SMS-EMOA
(S-Metric-Selection-EMOA) [Beume et al. 2007]
is a $(\mu + 1)$ EMOA and evaluates fitness of an individual $\boldsymbol{\lambda} \in \mathcal{P} \subset \boldsymbol{\Lambda}$ based on its contribution to the dominated HV:

$$\Delta s(\boldsymbol{\lambda}, \mathcal{P}) = S(\mathcal{P}, R) - S(\mathcal{P} \setminus \{\boldsymbol{\lambda}\}, R).$$

- Dark rectangles: HV contribution of dots.
- Grey point: reference point.
- The HVC contribution is the volume of space that is dominated only by $\boldsymbol{\lambda}$, and nothing else.
- $\tilde{\lambda}$ has lowest S-metric contribution.



SMS-EMOA algorithm I

Algorithm 2 SMS-EMOA

1: Generate start population $\mathcal{P}_0$ of size μ
2: $t \leftarrow 0$
3: **repeat**
4: Generate **one** individual $\mathbf{q}$ by recombination and mutation of $\mathcal{P}_t$
5: $\{\mathcal{F}_1, ..., \mathcal{F}_k\} \leftarrow \text{NDS}(\mathcal{P}_t \cup \{\mathbf{q}\})$
6: $\tilde{\boldsymbol{\lambda}} \leftarrow \text{argmin}_{\boldsymbol{\lambda} \in \mathcal{F}_k} \Delta s(\boldsymbol{\lambda}, \mathcal{F}_k)$
7: $\mathcal{P}_{t+1} \leftarrow (\mathcal{P}_t \cup \{\mathbf{q}\}) \setminus \{\tilde{\boldsymbol{\lambda}}\}$
8: $t \leftarrow t + 1$
9: **until** Termination criterion fulfilled

- L5: the set of temporary $(\mu + 1)$ individuals is partitioned by NDS into k fronts $\mathcal{F}_1, ..., \mathcal{F}_k$.
- L6-7: In last front, find $\tilde{\boldsymbol{\lambda}} \in \mathcal{F}_k$ with smallest HV contribution - and kill it.
- Fitness of an individual is mainly the rank of its front and HV contribution as tie-breaker.

Multi-criteria Optimization

Practical Applications

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Practical Applications in Machine Learning I

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

- Typically unbalanced classification tasks with unspecified costs.
- Could also use 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: Models in production need to predict fast.
- Size / Energy consumption: Models should be deployed on a mobile/edge device and not use much power.

Sparse Models: Balance *predictive performance* and *number of used features*, either for cost effieciency, but often also for interpretability.

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

- Model has to be fair regarding subgroups in the data, e.g. gender.
- Many different approaches to quantify fairness exist.

ROC Optimization - Setup

Again, we want to train a *spam detector* on the popular Spam dataset[1].

- Learning algorithm: SVM with RBF kernel.
- Hyperparameters to optimize:

`cost`	$[2^{-15}, 2^{15}]$
γ	$[2^{-15}, 2^{15}]$
Threshold t	$[0, 1]$

- Objective: *minimize* false positive rate (FPR) and *maximize* true positive rate (TPR), evaluated through 5-fold CV
- Optimizer: Multi-criteria Bayesian optimization:
  - ParEGO with $\rho = 0.05$, $s = 100000$.
  - Acquisition function u: *Confidence Bound* with $\alpha = 2$.
  - Budget: 100 evaluations
- Tuning is conducted on a training holdout and all hyperparameter configurations on the estimated Pareto front are validated on an outer validation set.

The threshold t could be separately optimized post-hoc.

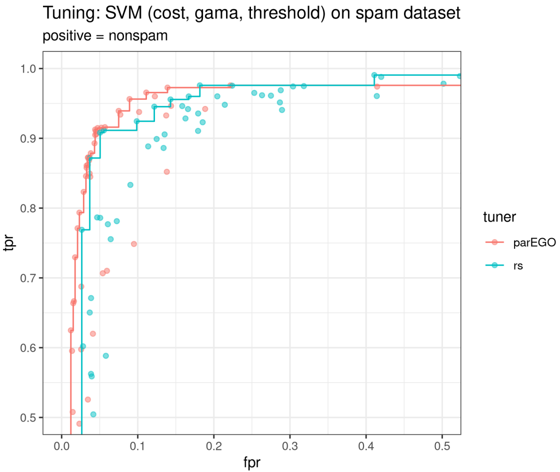
[1] `https://archive.ics.uci.edu/ml/datasets/spambase`

ROC Optimization - Result I

We notice:

- Compared to *random search*: Many *ParEGO* evaluations are on the Pareto front.
- The Pareto front of *ParEGO* dominates most points from the *random search*.
- The dominated hypervolume to the reference point $(0, 1)$ is:

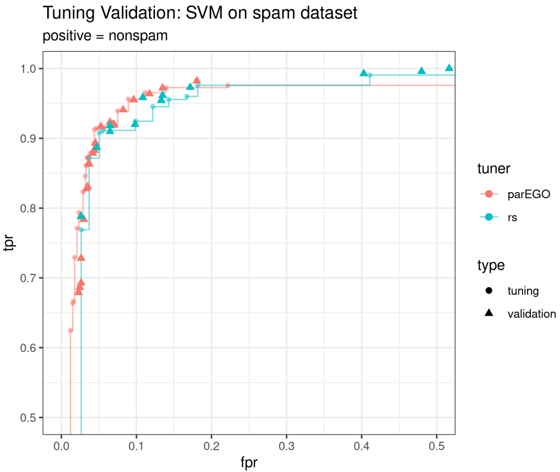
ParEGO:	0.965
random search:	0.959



ROC Optimization - Result II

We validate the configurations on the estimated Pareto front on a holdout:

- The performance on the validation set varies slightly.
- The TPR got slightly better but the FPR got slightly worse.
- On the validation set, some configurations get dominated by others.

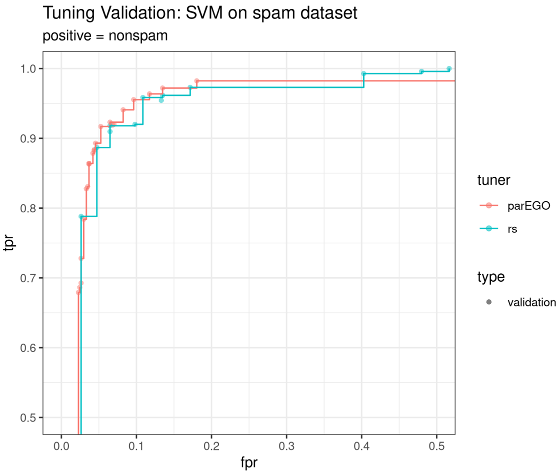


ROC Optimization - Result II

We validate the configurations on the estimated Pareto front on a holdout:

- The performance on the validation set varies slightly.
- The TPR got slightly better but the FPR got slightly worse.
- On the validation set, some configurations get dominated by others.
- The dominated hypervolume of the validation set is:

ParEGO:	0.960
random search:	0.961



Efficient Models - Overview

- "Efficiency" can be:
  - Memory consumption of the model
  - Training or prediction time
  - Number of features needed
  - Energy consumption for prediction
  - . . .
- Some hyperparameters have a strong impact on the efficiency of a model, e.g.,
  - Number of trees in *random forests* or *gradient tree boosting*,
  - Number, size and type of layers in *neural networks*,
  - L1 regularization penalties,
  - ...
- Other hyperparameters might have no influence on efficiency.
- Typical scenario: Optimize jointly over multiple algorithms of varying efficiency.

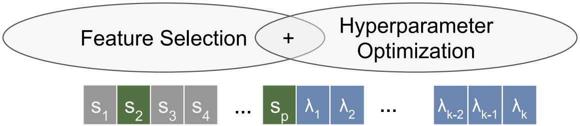
Efficient Models - Example: Feature Selection I

Goal of *feature selection*: Identify an informative feature subset with only a small drop in predictive performance compared to all features.

Find optimal hyperparameter setting $\boldsymbol{\lambda}$ and minimal feature subset s

$$\min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}, s\in\{0,1\}^p} \left(\widehat{GE}\left(\mathcal{I}(\mathcal{D}, \boldsymbol{\lambda}, s)\right), \frac{1}{p}\sum_{i=1}^{p} s_i\right)$$

- Problem: Feature selection and hyperparameter tuning are usually two separate steps.
- Solution: Identify an informative subset of features and a good hyperparameter configuration **simultaneously**.

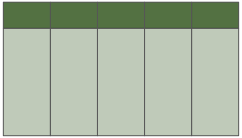


Efficient Models - Example: Feature Selection II

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

- Pre-calculate multiple ranked *feature filter values* .

RF Feature Importance:	(2	3	4	1	5)	x	0.7
AUC:	(2	1	3	4	5)	x	0.2
Information Gain:	(1	3	5	2	4)	x	0.1
	1.9	**2.6**	**3.9**	**1.7**	**4.9**		



w: search space component

$$\mathrm{EF}_j(\mathbf{w}) = \sum_{m=1}^{M} w_m F^m(\mathcal{D})_j.$$

- New joint hyperparameter vector: $\boldsymbol{\lambda} = (\tilde{\boldsymbol{\lambda}}, w_1, \ldots, w_p, \tau)$
  - Hyperparameters of learner: $\tilde{\boldsymbol{\lambda}}$
  - Weight of each *feature filter value* vector: $(w_1, \ldots, w_p)$
  - Fraction of features to keep τ

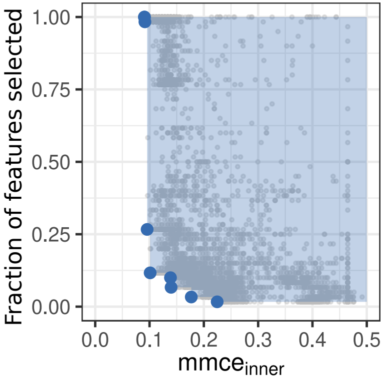
Efficient Models - Example: Feature Selection III

Combined feature selection and hyperparameter optimization on `Sonar` dataset[2].

- Learning algorithm: SVM with RBF kernel.
- Hyperparameters to optimize:

$$\begin{array}{rl} \texttt{cost} & [2^{-10}, 2^{10}] \\ \gamma & [2^{-10}, 2^{10}] \\ (w_1, \ldots, w_p) & [0,1]^p \\ \tau & [0,1]^p \end{array}$$

- Objective: minimize *misclassification* and *fraction of features selected*
- Optimizer: *ParEGO* with *random forest* surrogate, LCB acquisition function, 15 batch proposals, budget: 2000 evaluations



[2]Only the tuning error is shown here

Efficient Models - Example: FLOPS

Goal: Optimize prediction accuracy and number of floating point operations (FLOPs) [Wang et al. 2019].

Data: Image Classification on CIFAR-10.

Learner: *DenseNet* - Densely Connected Convolutional Network [Huang et al. 2018].

- Composed out of 4 *dense blocks*.
- A dense block consists of multiple convolutional layers where the inputs for each layer are all feature maps of all preceding layers in the block.
- Dense blocks are connected via convolutional and max pooling layer.

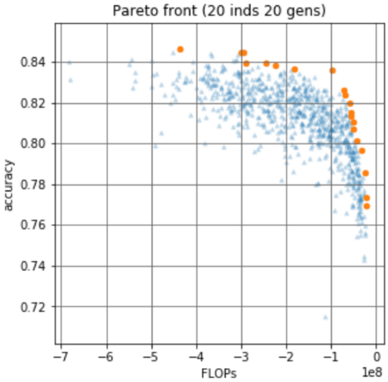
Training: 300 Epochs with a batch size of 128 and initial learning rate of 0.1.

Efficient Models - Example: FLOPS

- Objective: *accuracy* vs. *FLOPS* (floating point operations, per observation)
- Search Space:

`growth rate` (k)	$[8, 32]$
`layers in first block`	$[4, 6]$
`layers in second block`	$[4, 12]$
`layers in third block`	$[4, 24]$
`layers in fourth block`	$[4, 16]$

- Tuner: *Particle Swarm Optimization* with a population size of 20 and 400 evaluations.

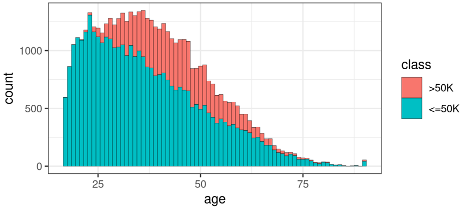


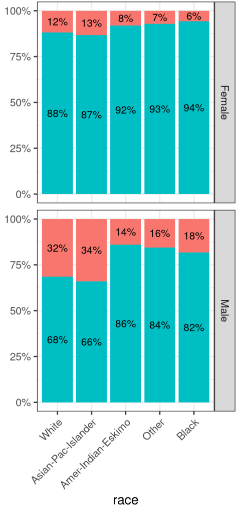
The `growth rate` is the number of output feature maps in each layer of a block

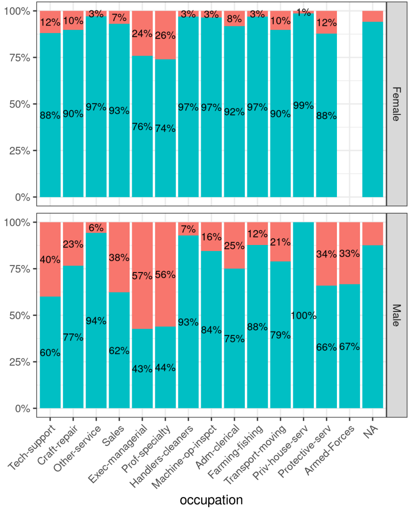
Fair Models - The `Adult` dataset

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 I

A fair model for income prediction on binarized target.

- Learner: *eXtreme Gradient Boosting*
- Hyperparameters to optimize:

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

- Objective: minimize *misclassification error* and *unfairness*

Fair Models - Setup II

- Careful: Usually this data would be used to model the relation between person characteristics and income, then to **discuss and study** by careful inference - to **figure out if** something like e.g. a "gender pay gap" exists.
- Here, in our toy example we **pretend** now that we would like to create a automatic "assignment algorithm" for salary - maybe not totally unrealistic nowadays? In **such** a scenario, biasing the prediction by **incorporating fairness** might be of interest.
- Here, a simplified proxy for fairness is defined as the absolute difference in F1-Scores between female (f) and male (m) population (low is good):

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

Fair Models - Results

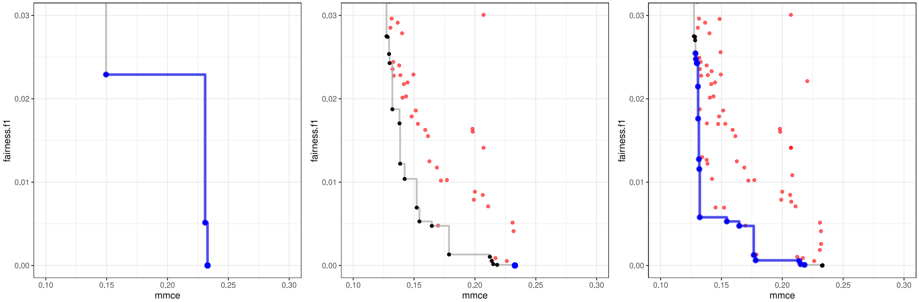


Figure: Pareto fronts after 20, 70 and 120 tuning iterations.

- Optimizer: ParEGO with random forest surrogate and restricted range of projections to $[0.1, 0.9]$ (No interest in very unfair or bad configurations).
- Here, the hyperparameters actually have an effect on the defined *fairness measure*.
- However, this is often not the case or not enough to ensure a fair model.