Multi-objective machine learning to predict Pareto fronts

Timo M. Deist

Centrum Wiskunde & Informatica (CWI), Amsterdam (until 2021)

May 6, 2022

Acknowledgments

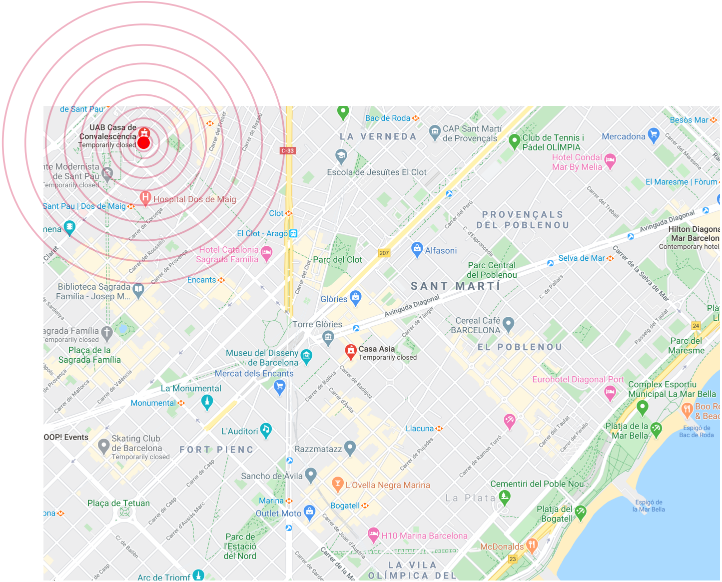
Co-authors:

Stef Maree	Monika Grewal	Frank Dankers	Tanja Alderliesten	Peter Bosman
				

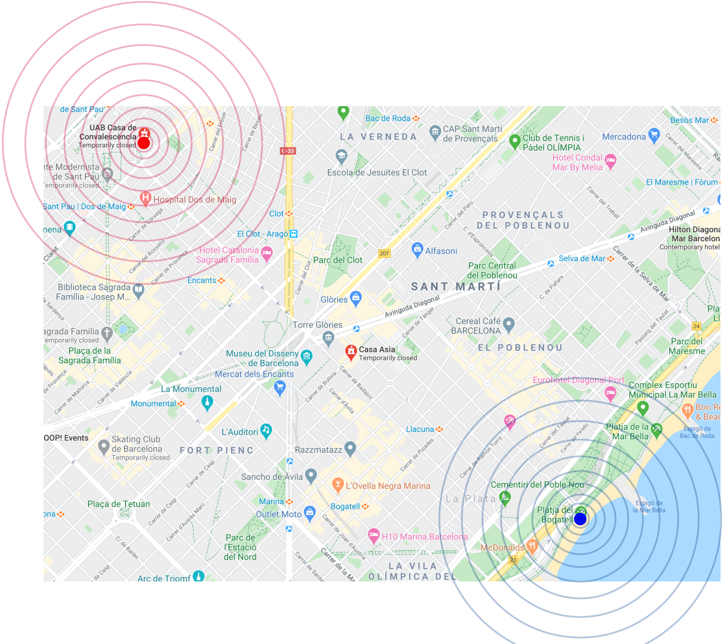
Feedback:

Marco Virgolin (CWI)

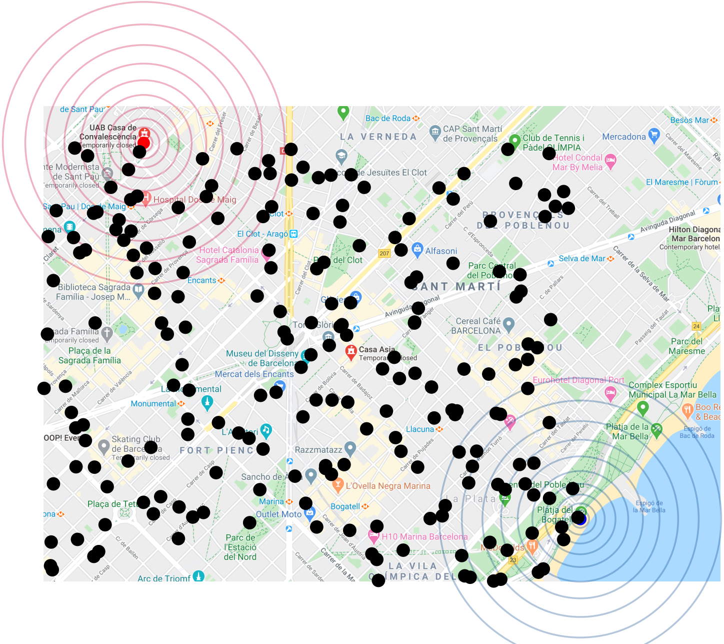
The Beach-Conference conundrum



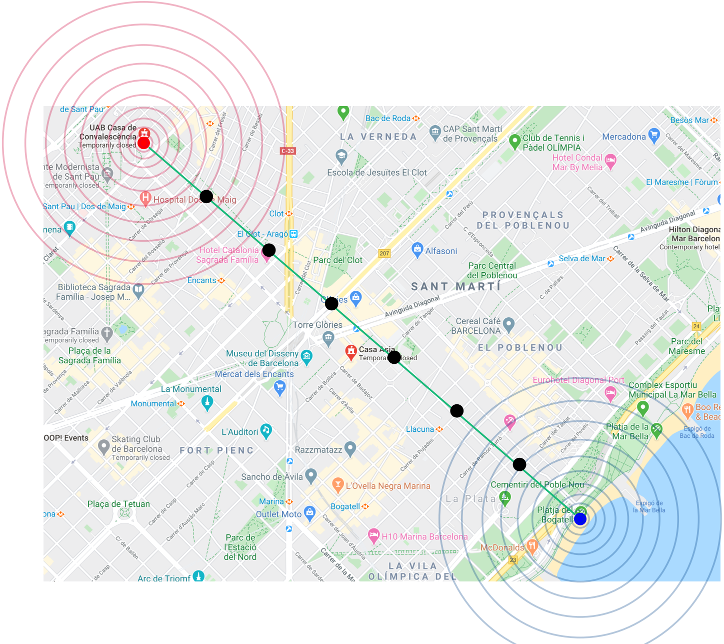
The Beach-Conference conundrum



The Beach-Conference conundrum



The Beach-Conference conundrum



Multi-objective (a posteriori) decision-making[1]

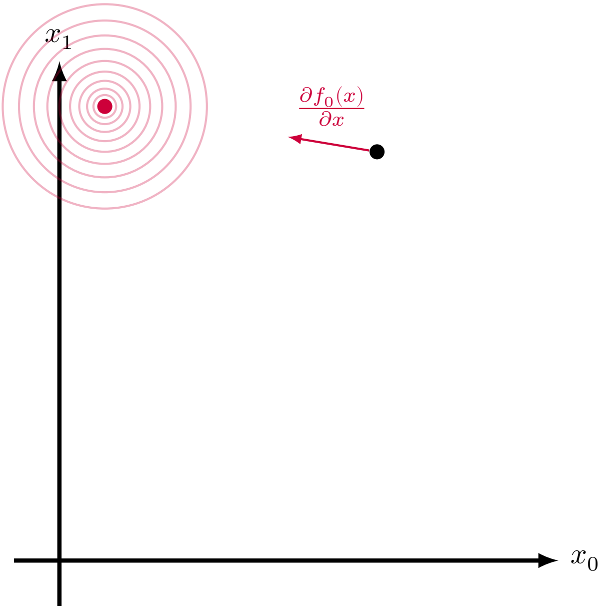
1. Optimize two well-defined, competing objectives
   - ↓ Distance to conference
   - ↓ Distance to beach
2. Present decision-maker with p solutions on the Pareto front
3. Decision-maker chooses solution

What is needed: The Pareto front.

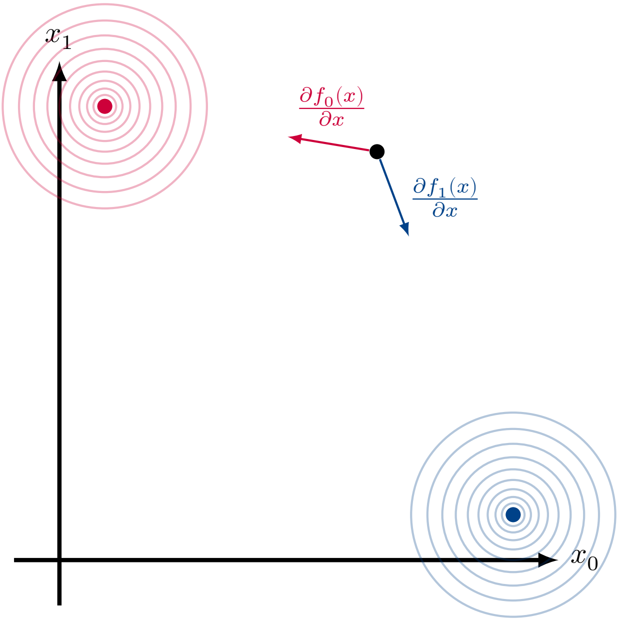
Problem statement

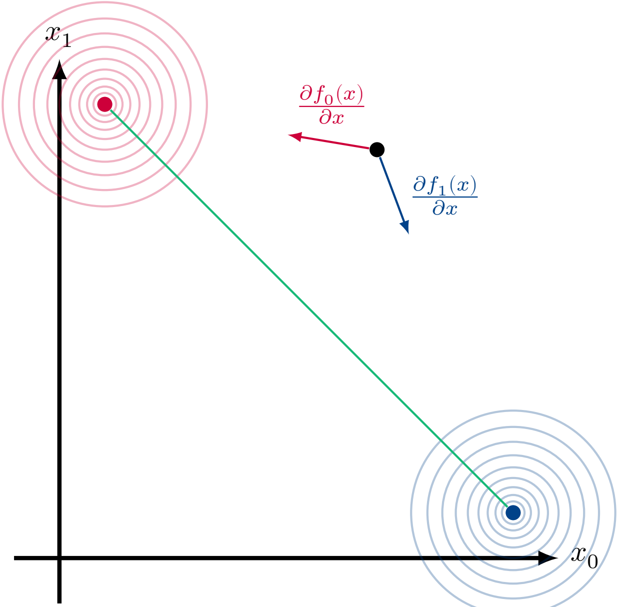
Find p solutions that span the Pareto front.

[1] Thiele et al. (2009)

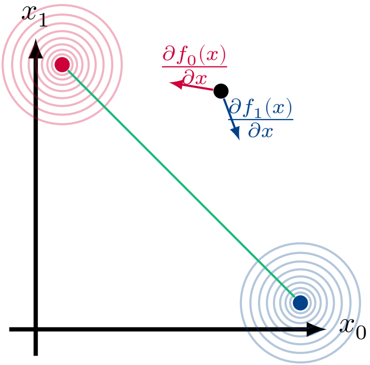


x_1
$\frac{\partial f_0(x)}{\partial x}$
x_0

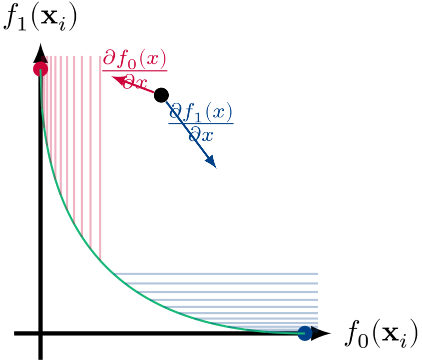

x_1
$\frac{\partial f_0(x)}{\partial x}$
$\frac{\partial f_1(x)}{\partial x}$
x_0

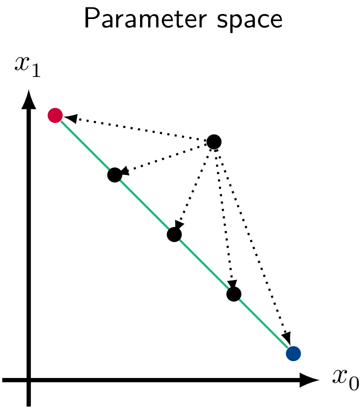

x_1
$\frac{\partial f_0(x)}{\partial x}$
$\frac{\partial f_1(x)}{\partial x}$
x_0

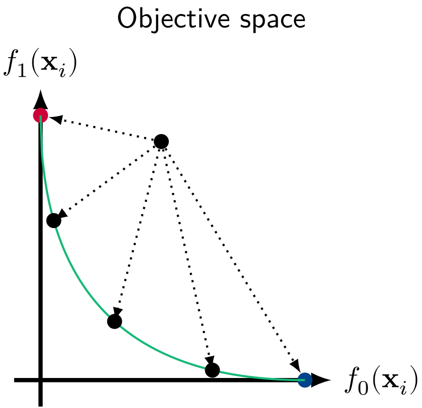
Parameter space



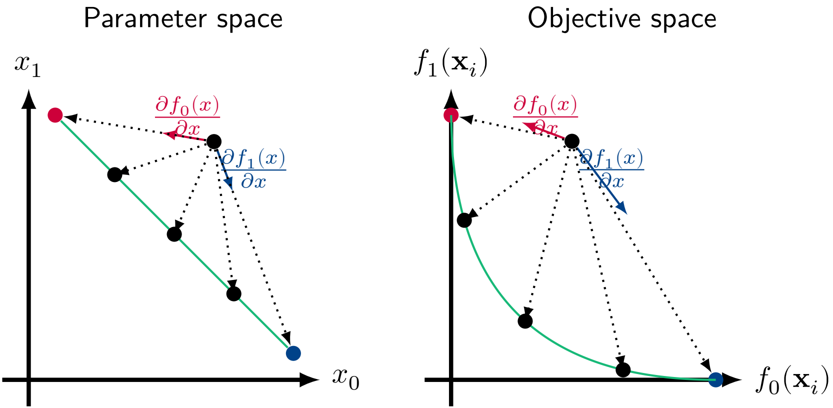
Objective space



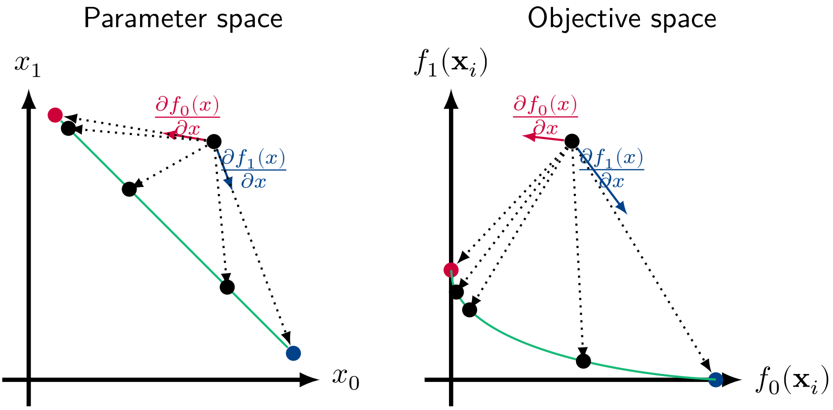




Goal: find a set of solutions evenly spread across the Pareto front



$$d = w_0 \frac{\partial f_0(x)}{\partial x} + w_1 \frac{\partial f_1(x)}{\partial x}$$

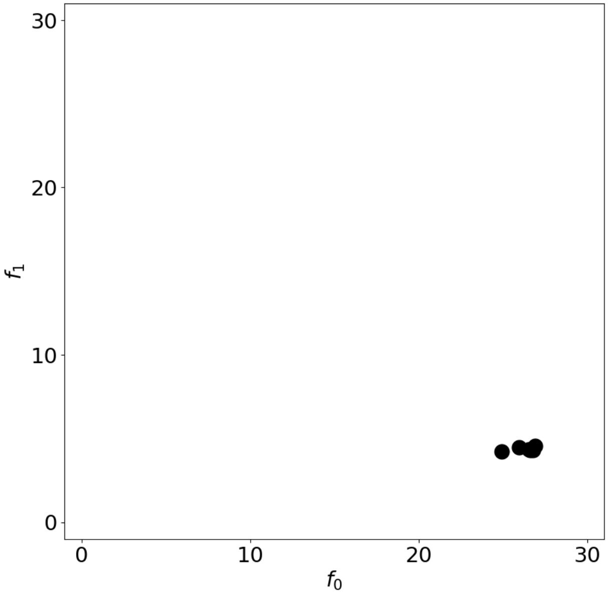


$$d_i = a\frac{\partial f_0(x)}{\partial x} + (1-a)\frac{\partial f_1(x)}{\partial x} \quad \forall a_i \in \{0.1, 0.2, \dots, 1\}$$

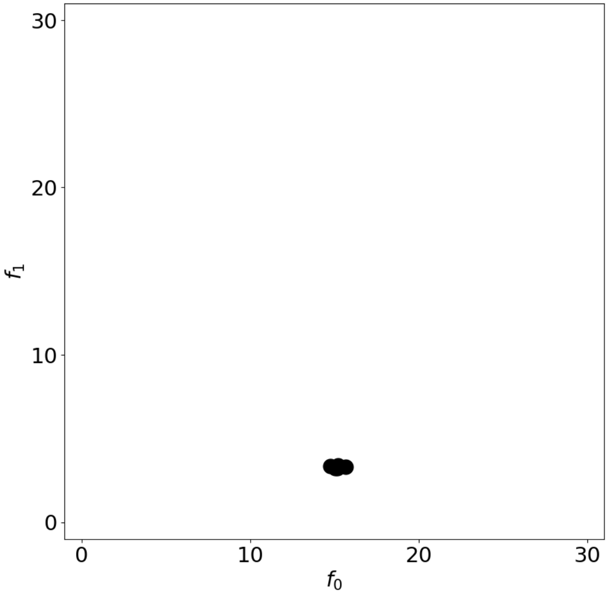
What can happen for some MO problems...[2]

[1] Das and Dennis (1997)

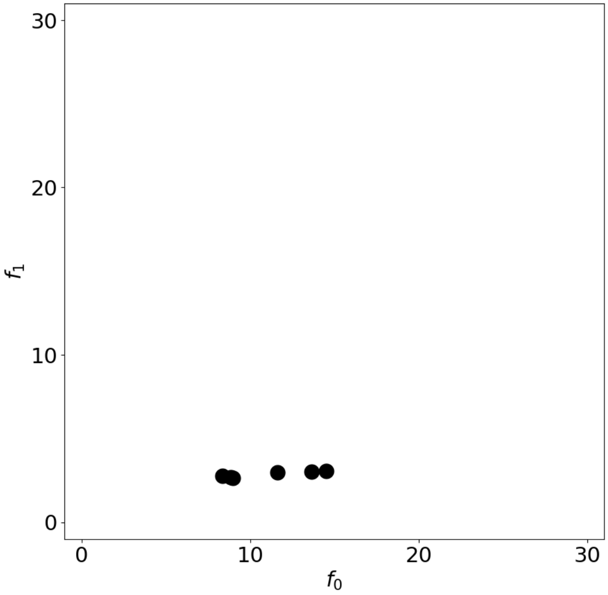
MO optimization example



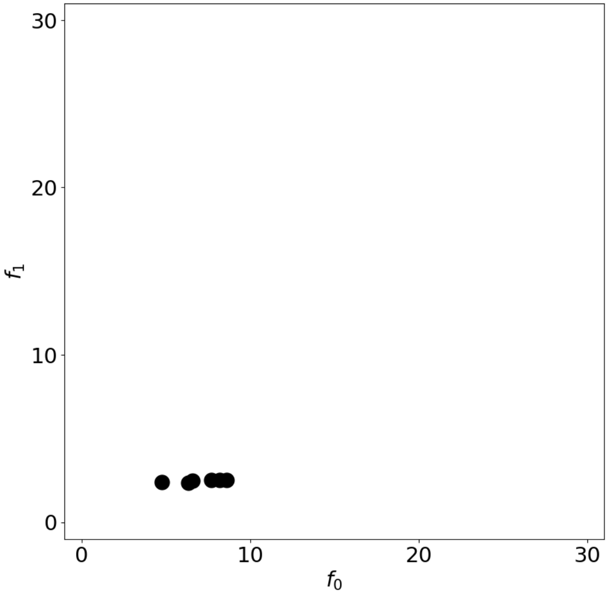
MO optimization example



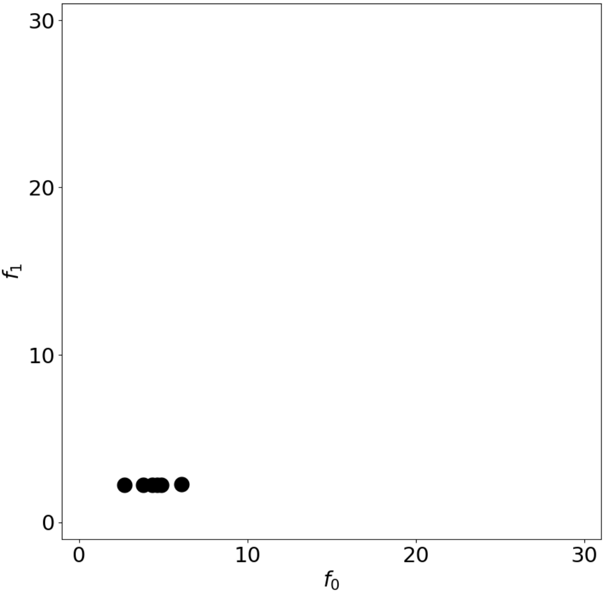
MO optimization example



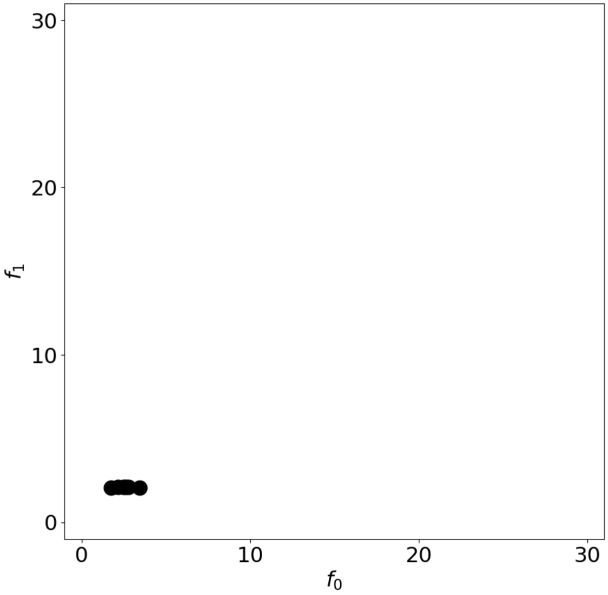
MO optimization example



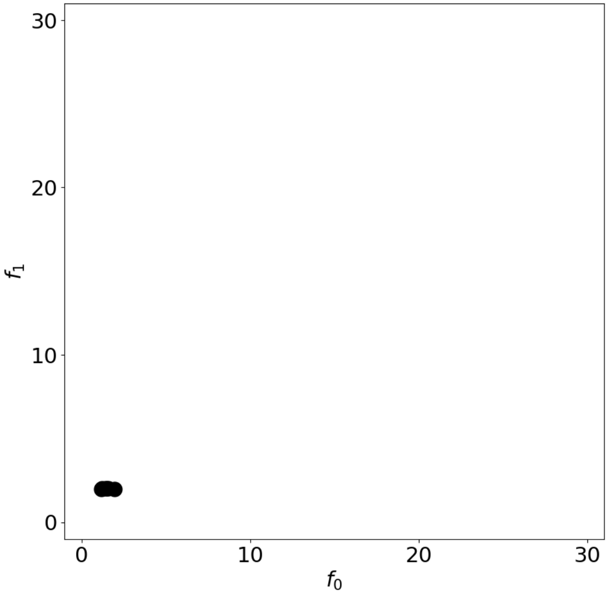
MO optimization example



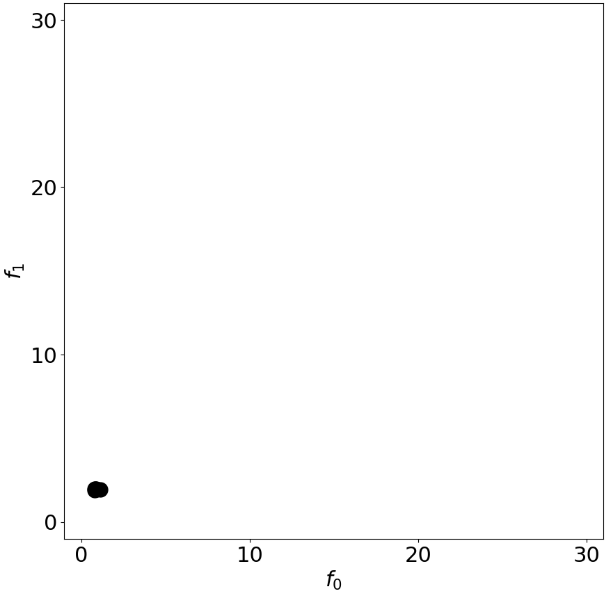
MO optimization example



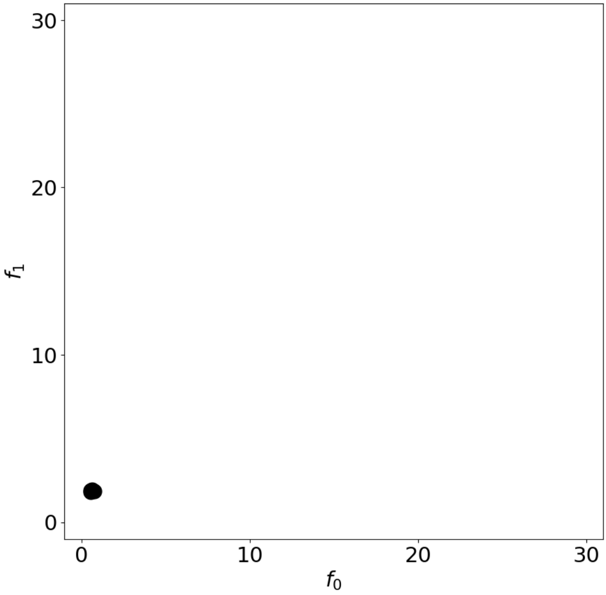
MO optimization example



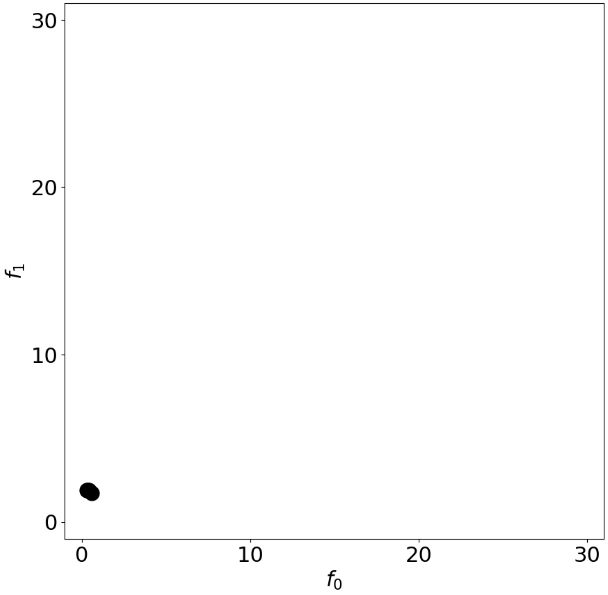
MO optimization example



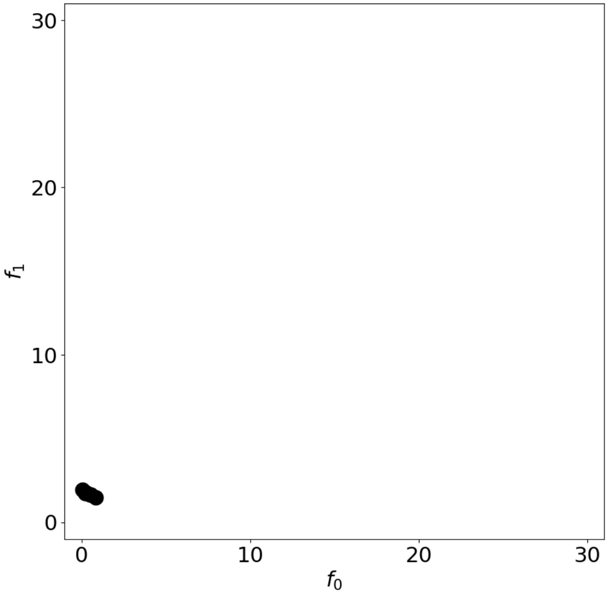
MO optimization example



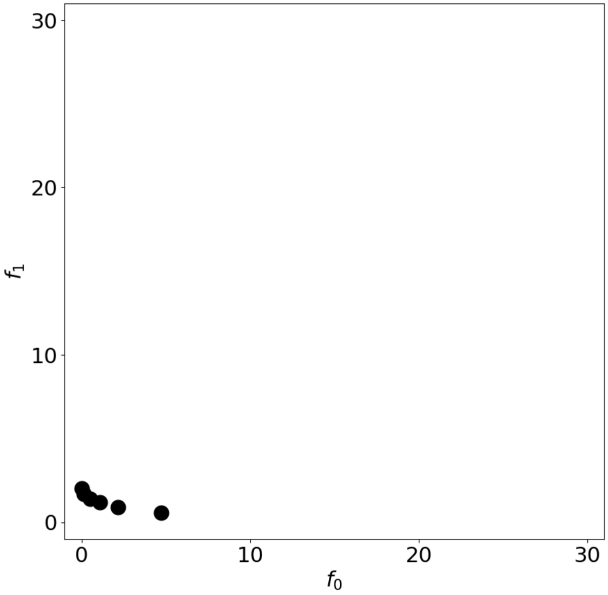
MO optimization example



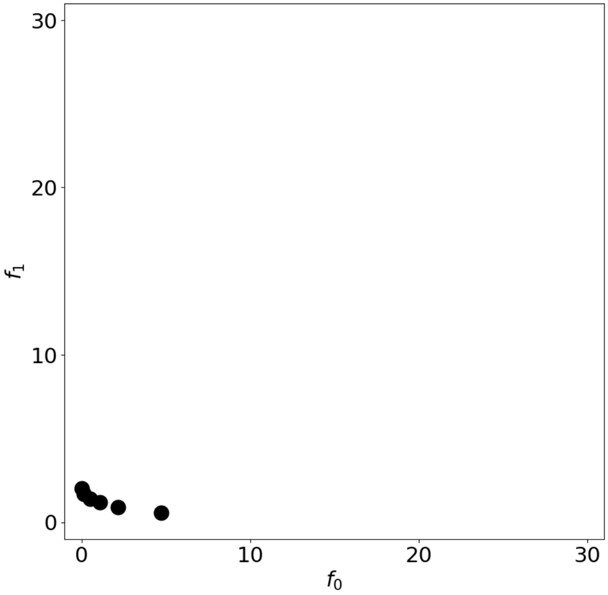
MO optimization example



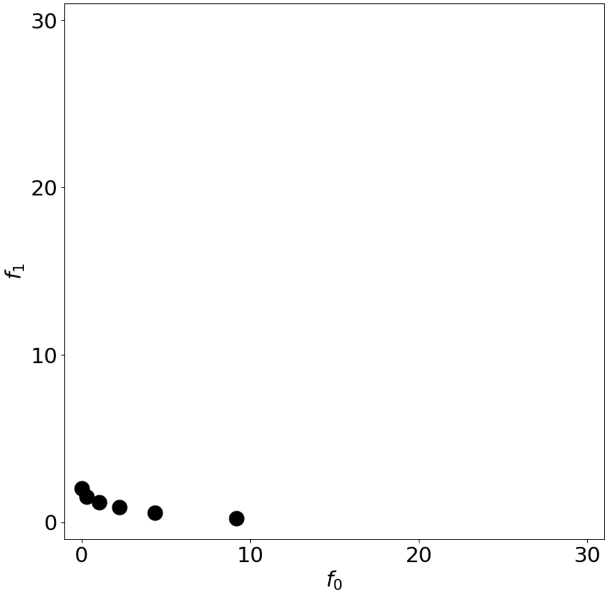
MO optimization example



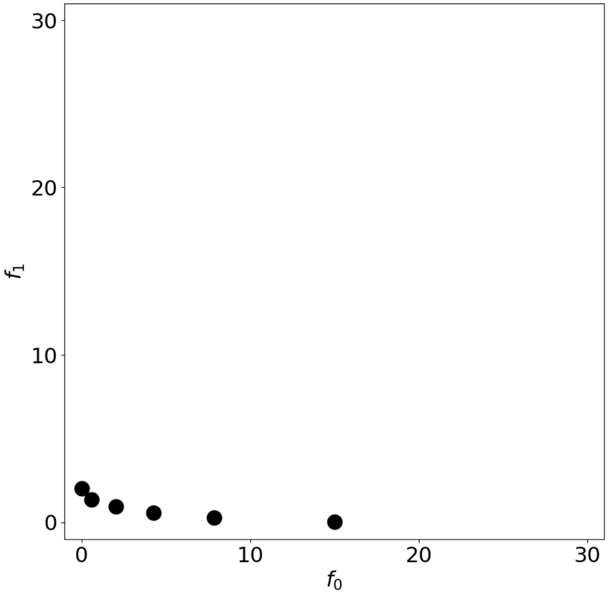
MO optimization example



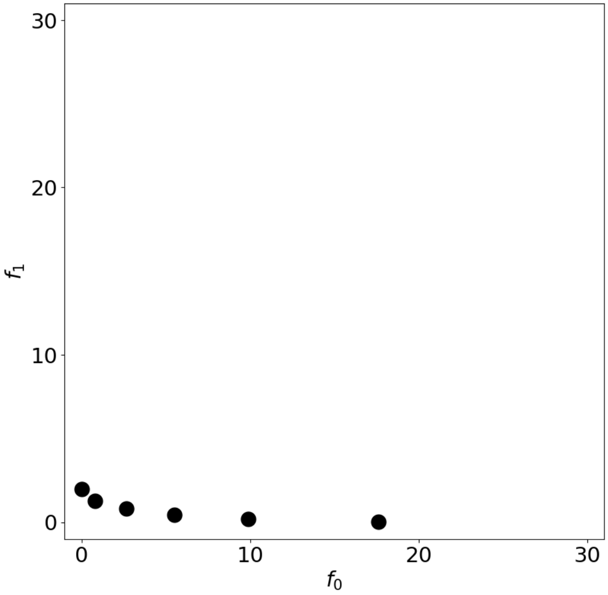
MO optimization example



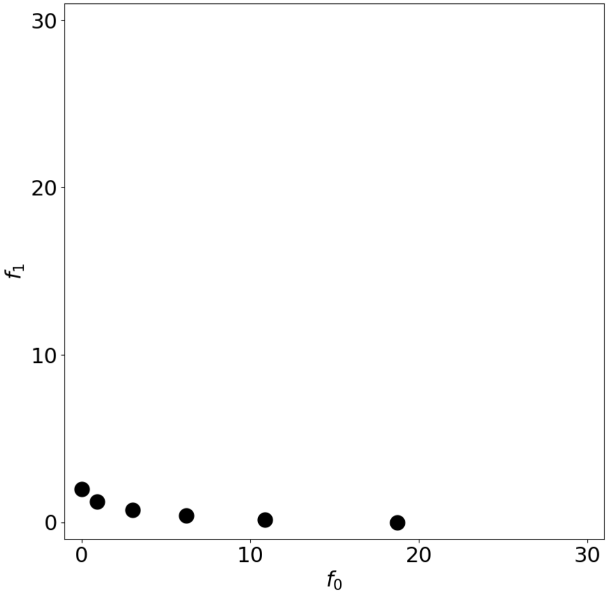
MO optimization example



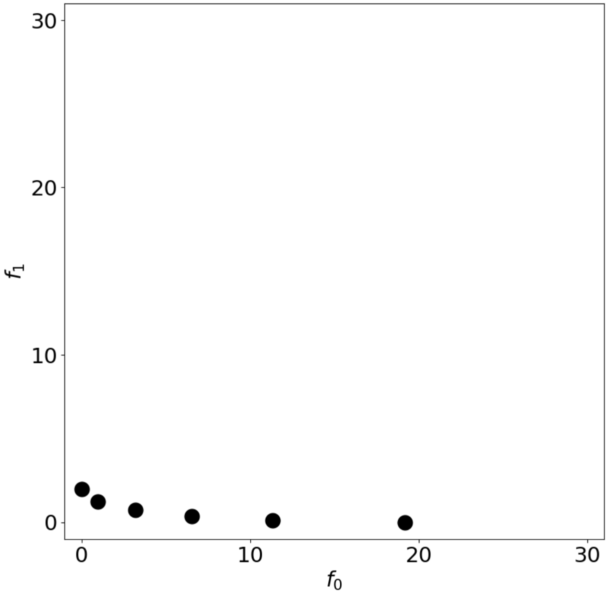
MO optimization example



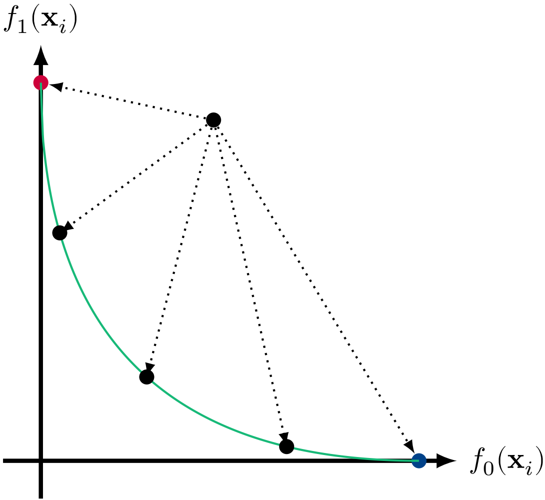
MO optimization example

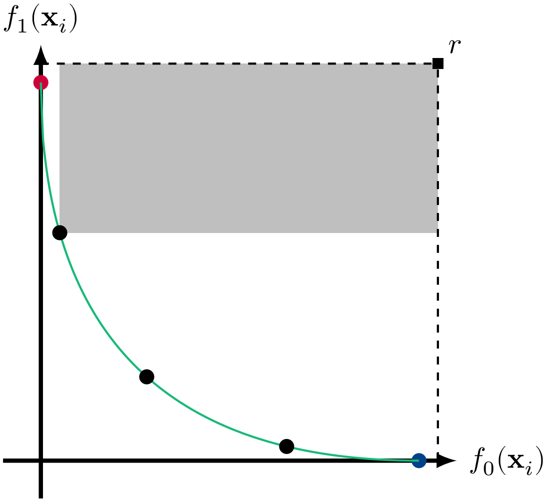


MO optimization example

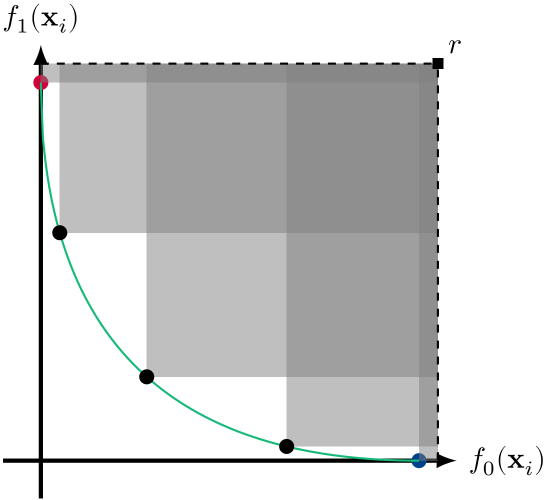


Objective space

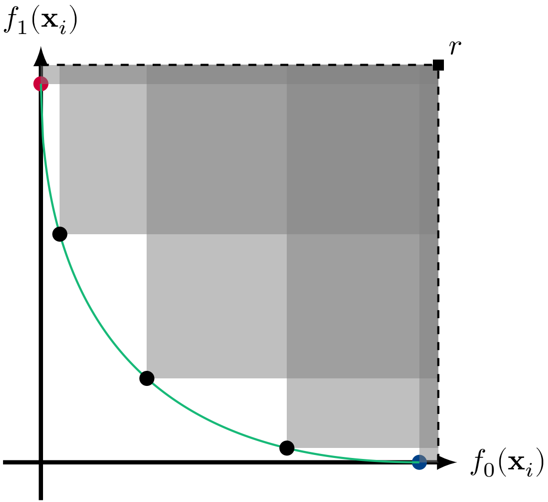




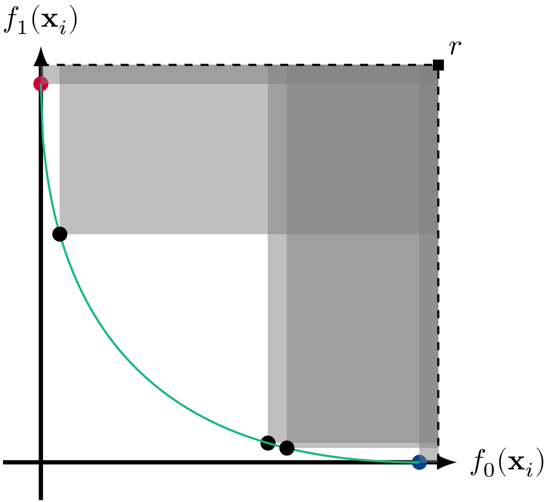
Dominated **hypervolume (HV)**



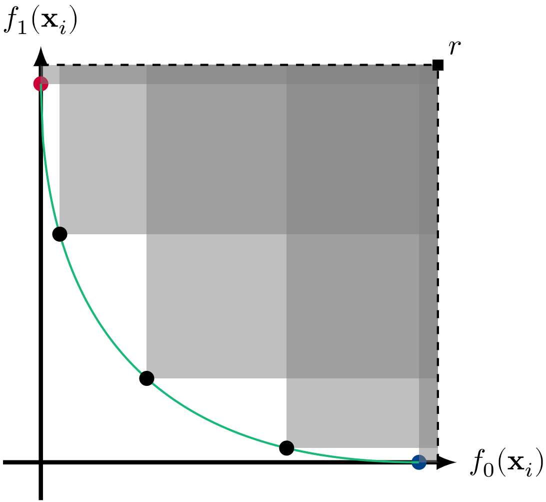
Dominated **hypervolume (HV)**



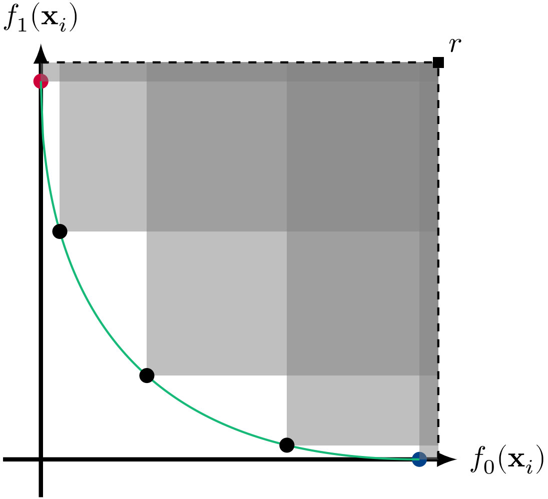
The dominated **hypervolume (HV)** is maximal in solutions that are spread over the front.



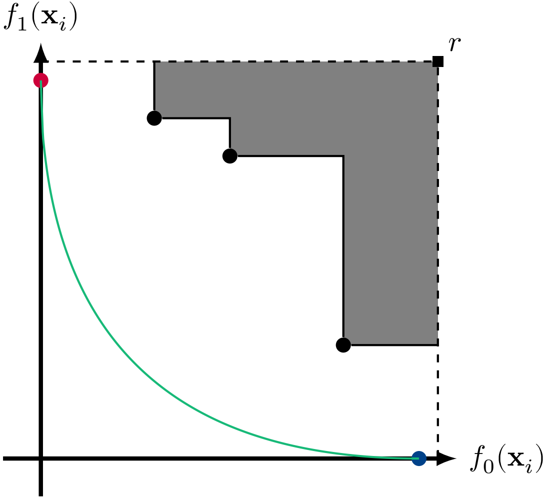
The dominated **hypervolume (HV)** is maximal in solutions that are spread over the front.

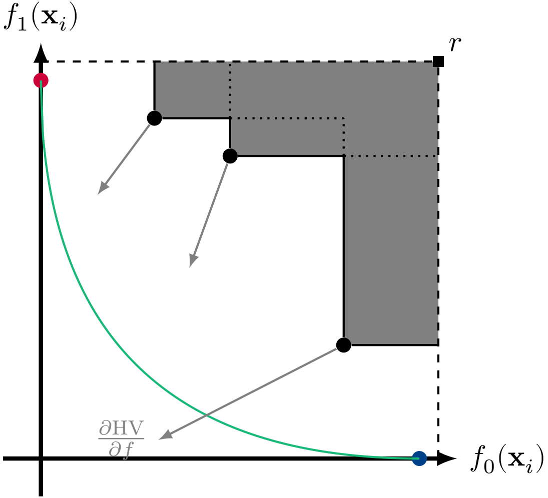


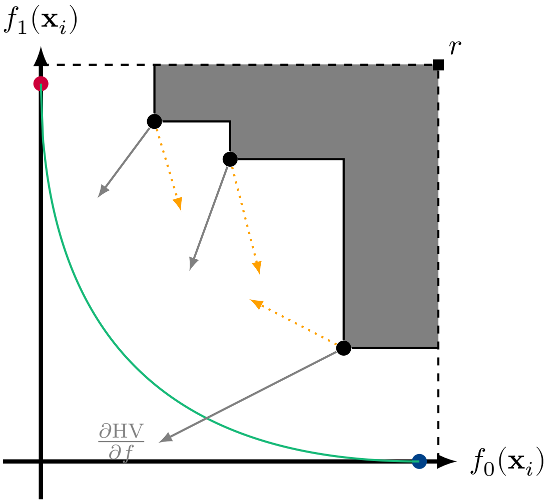
The dominated **hypervolume (HV)** is maximal in solutions that are spread over the front.



HV selects good sets of solutions. Can we compute HV gradients to help finding those sets?

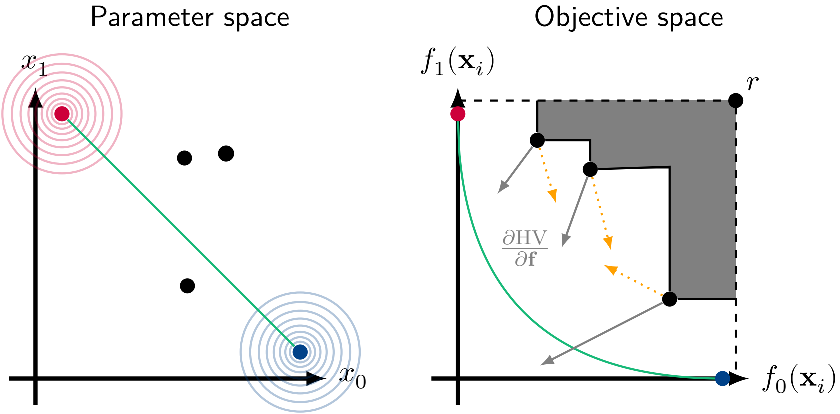

$f_1(\mathbf{x}_i)$
r
$f_0(\mathbf{x}_i)$


$f_1(\mathbf{x}_i)$
r
$\frac{\partial \mathrm{HV}}{\partial f}$
$f_0(\mathbf{x}_i)$

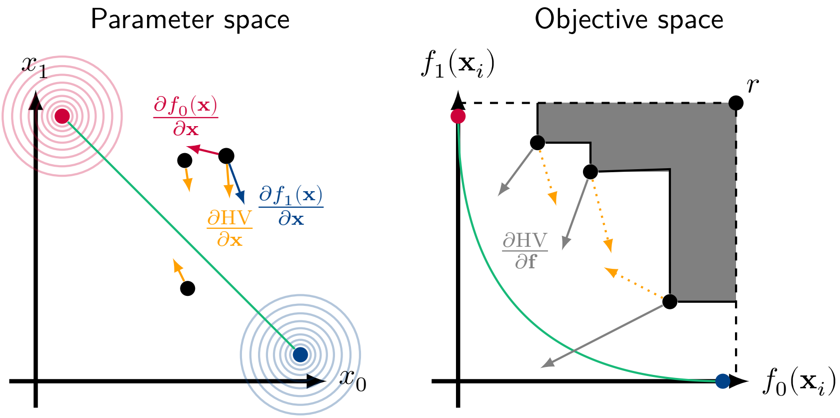


$$\frac{\partial \mathrm{HV}}{\partial \mathbf{x}_i} = \frac{\partial \mathrm{HV}}{\partial f_0(\mathbf{x}_i)} \frac{\partial f_0(\mathbf{x}_i)}{\partial \mathbf{x}_i} + \frac{\partial \mathrm{HV}}{\partial f_1(\mathbf{x}_i)} \frac{\partial f_1(\mathbf{x}_i)}{\partial \mathbf{x}_i}$$

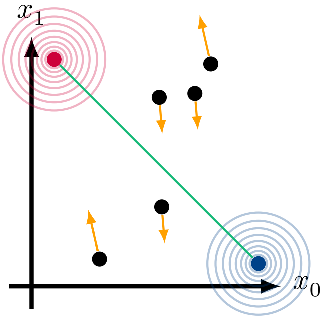
[2] Emmerich and Deutz (2014)



$$\frac{\partial \mathrm{HV}}{\partial \mathbf{x}_i} = \frac{\partial \mathrm{HV}}{\partial f_0(\mathbf{x}_i)} \frac{\partial f_0(\mathbf{x}_i)}{\partial \mathbf{x}_i} + \frac{\partial \mathrm{HV}}{\partial f_1(\mathbf{x}_i)} \frac{\partial f_1(\mathbf{x}_i)}{\partial \mathbf{x}_i}$$

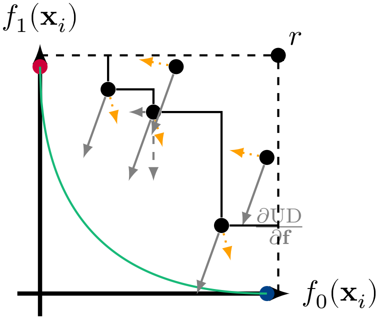


$$\frac{\partial \mathrm{HV}}{\partial \mathbf{x}_i} = \frac{\partial \mathrm{HV}}{\partial f_0(\mathbf{x}_i)} \frac{\partial f_0(\mathbf{x}_i)}{\partial \mathbf{x}_i} + \frac{\partial \mathrm{HV}}{\partial f_1(\mathbf{x}_i)} \frac{\partial f_1(\mathbf{x}_i)}{\partial \mathbf{x}_i}$$



Parameter space

1) Initialize p solutions



Objective space

2) Compute $\frac{\partial HV}{\partial \mathbf{f}}$ for non-dominated solutions

3) Compute $\frac{\partial UD}{\partial \mathbf{f}}$ for dominated solutions

4) Compute update direction $\frac{\partial UHV}{\partial \mathbf{x}}$

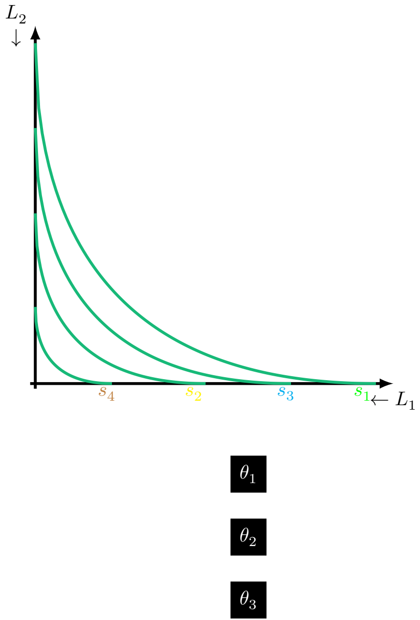
5) Move solutions

6) Repeat until convergence

Notation

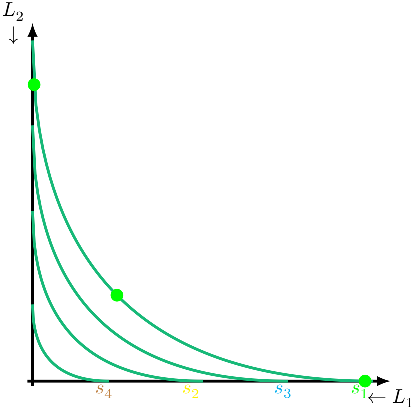
MO optimization		MO learning
Optimization instance	$\rightarrow$	Samples s_k
Variables $\mathbf{x}_i$	$\rightarrow$	Model parameters θ_i
Objectives f_j	$\rightarrow$	Losses L_j

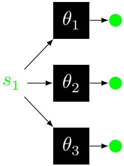
MO learning problem



- Each sample has its own Pareto front

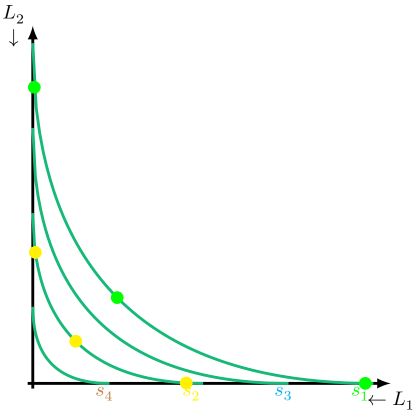
MO learning problem

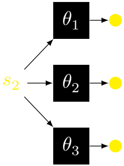




- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample

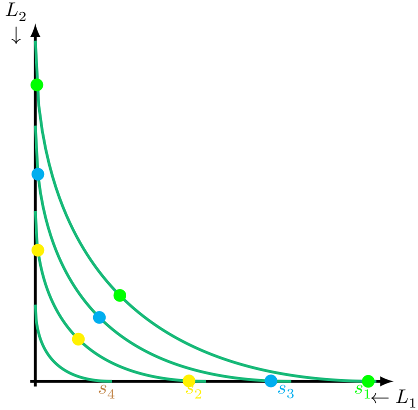
MO learning problem

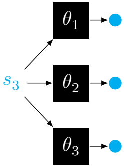




- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample

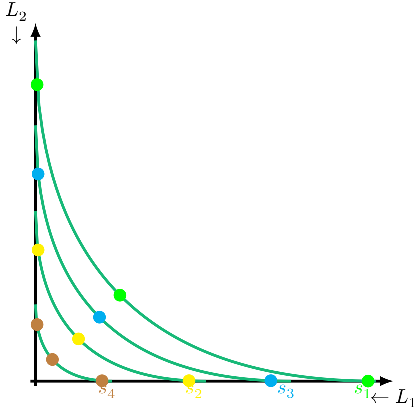
MO learning problem

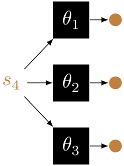




- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample

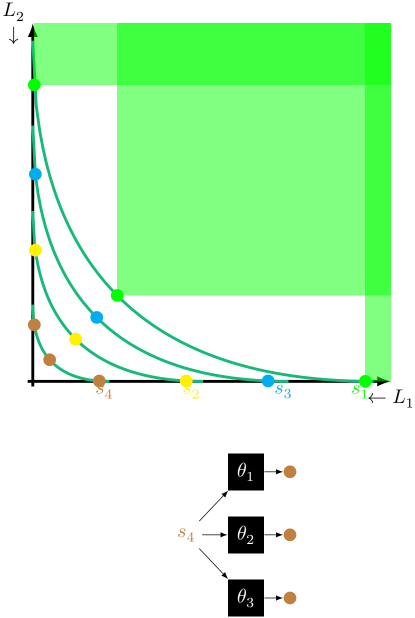
MO learning problem





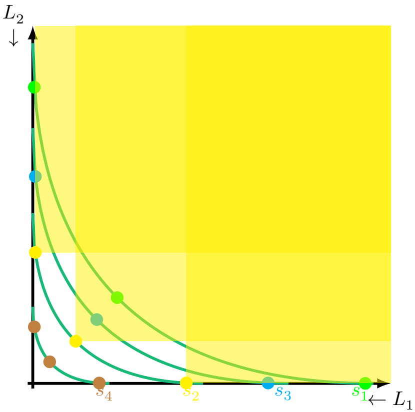
- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample

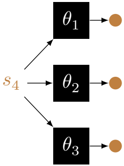
MO learning problem



- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample
- How? Train learners so that the average HV is maximal for all samples

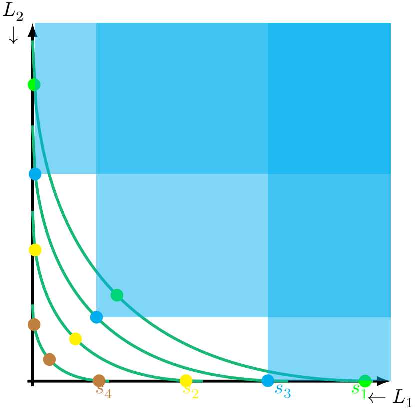
MO learning problem

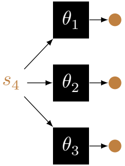




- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample
- How? Train learners so that the average HV is maximal for all samples

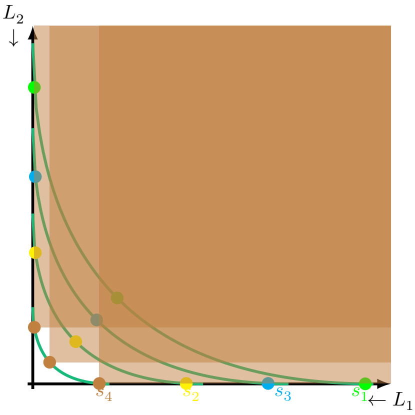
MO learning problem

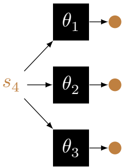




- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample
- How? Train learners so that the average HV is maximal for all samples

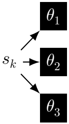
MO learning problem

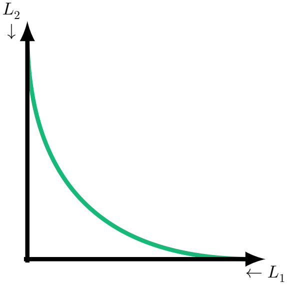




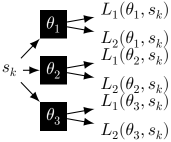
- Each sample has its own Pareto front
- What we want: learners generating Pareto optimal points for each sample
- How? Train learners so that the average HV is maximal for all samples

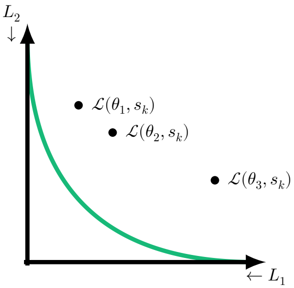
Forward pass





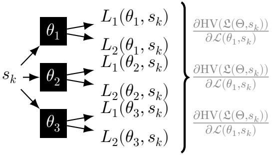
Forward pass

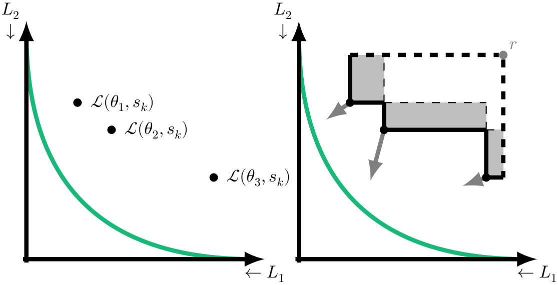




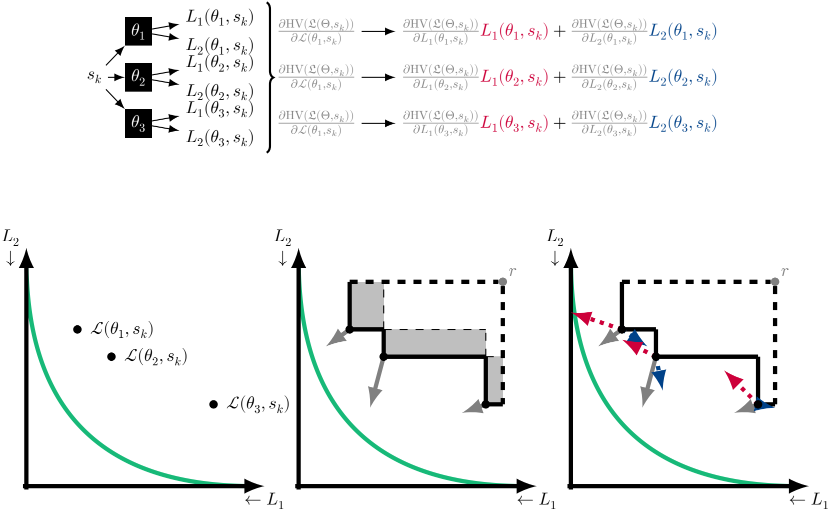
Forward pass

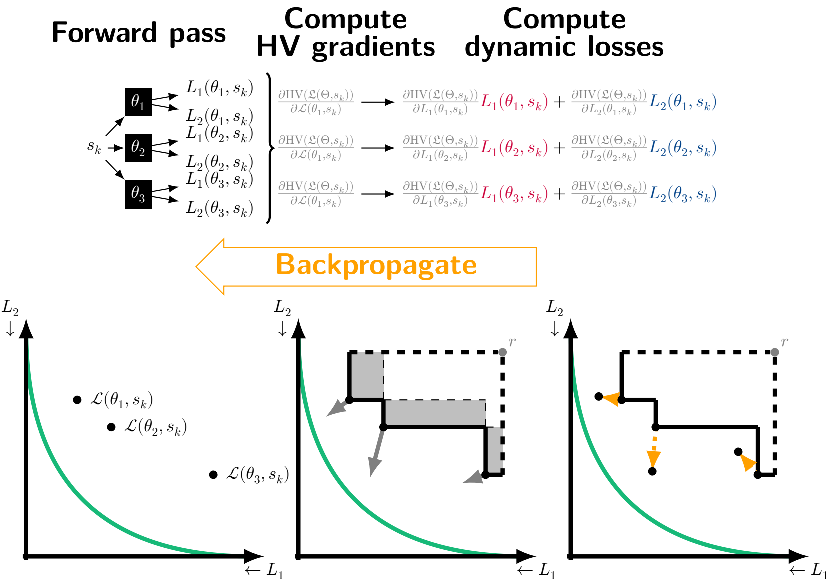
Compute HV gradients





Forward pass — **Compute HV gradients** — **Compute dynamic losses**





Forward pass
Compute HV gradients
Compute dynamic losses
s_k
θ_1
θ_2
θ_3
$L_1(\theta_1, s_k)$
$L_2(\theta_1, s_k)$
$L_1(\theta_2, s_k)$
$L_2(\theta_2, s_k)$
$L_1(\theta_3, s_k)$
$L_2(\theta_3, s_k)$
$\frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial \mathcal{L}(\theta_1,s_k)} \longrightarrow \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_1(\theta_1,s_k)} L_1(\theta_1, s_k) + \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_2(\theta_1,s_k)} L_2(\theta_1, s_k)$
$\frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial \mathcal{L}(\theta_1,s_k)} \longrightarrow \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_1(\theta_2,s_k)} L_1(\theta_2, s_k) + \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_2(\theta_2,s_k)} L_2(\theta_2, s_k)$
$\frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial \mathcal{L}(\theta_1,s_k)} \longrightarrow \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_1(\theta_3,s_k)} L_1(\theta_3, s_k) + \frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta,s_k))}{\partial L_2(\theta_3,s_k)} L_2(\theta_3, s_k)$
Backpropagate
$L_2 \downarrow$
$\leftarrow L_1$
$\mathcal{L}(\theta_1, s_k)$
$\mathcal{L}(\theta_2, s_k)$
$\mathcal{L}(\theta_3, s_k)$
r

MO regression

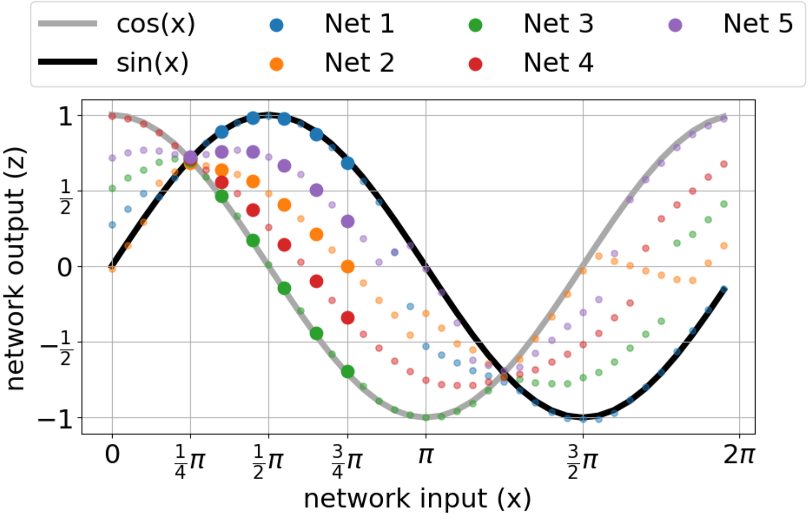
Given: $x_k,\ X \in [0, 2\pi]$

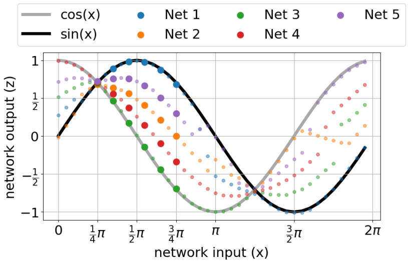
Predict: z_k that matches $y_k^{(j)}$

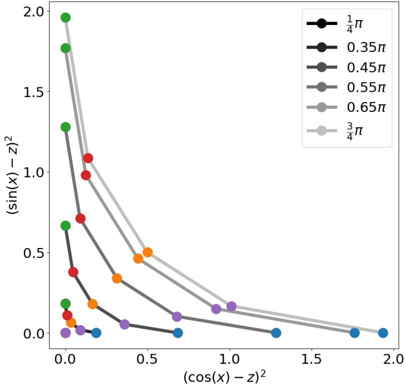
$$Y_1 = \cos(X), \quad Y_2 = \sin(X)$$

Use Mean Square Error (MSE) for learning:

$$L_j = \mathrm{MSE}_j = \frac{1}{|S|} \sum_{k=1}^{|S|} (y_k^{(j)} - z_k)^2$$

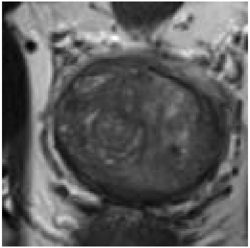

cos(x)
sin(x)
Net 1
Net 2
Net 3
Net 4
Net 5
network output (z)
1
$\frac{1}{2}$
0
$-\frac{1}{2}$
−1
0
$\frac{1}{4}\pi$
$\frac{1}{2}\pi$
$\frac{3}{4}\pi$
π
$\frac{3}{2}\pi$
2π
network input (x)


cos(x)
sin(x)
Net 1
Net 2
Net 3
Net 4
Net 5
network output (z)
network input (x)
1
$\frac{1}{2}$
0
$-\frac{1}{2}$
−1
0
$\frac{1}{4}\pi$
$\frac{1}{2}\pi$
$\frac{3}{4}\pi$
π
$\frac{3}{2}\pi$
2π

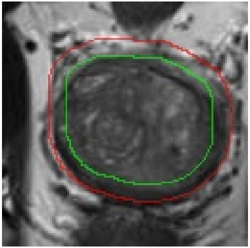

$\frac{1}{4}\pi$
0.35π
0.45π
0.55π
0.65π
$\frac{3}{4}\pi$
$(\sin(x)-z)^2$
$(\cos(x)-z)^2$
0.0
0.5
1.0
1.5
2.0

Organ segmentation

Prostate MRI



2 segmentations



Given: MRI scan[a]

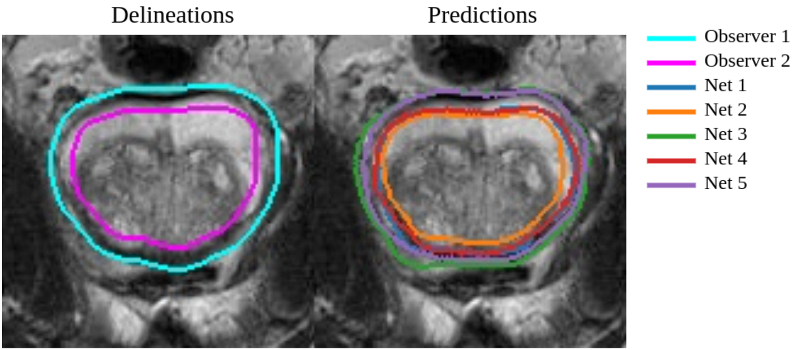
Predict: Prostate segmentation trading off **both** segmentations.

L_1 = Cross Entropy w.r.t. segmentation 1

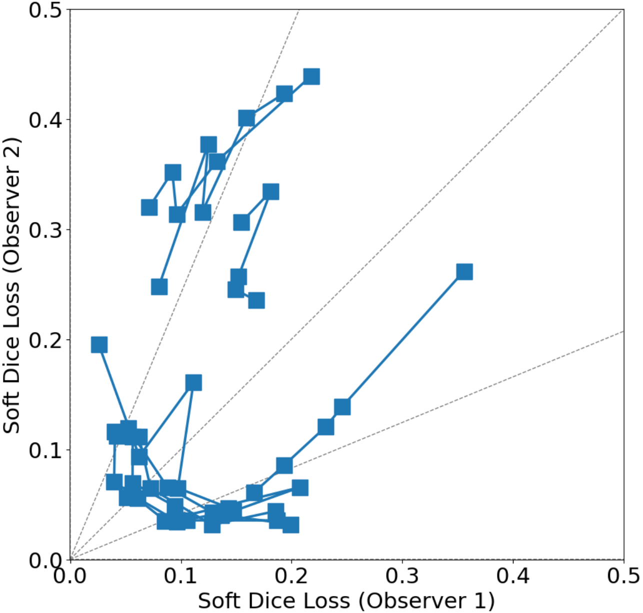
L_2 = Cross Entropy w.r.t. segmentation 2

[a]Data described in Dushatskiy et al. (2020)

Organ segmentation



Organ segmentation



Neural style transfer

Photo

Style

Popularized by Gatys et al. (2016)

Given: Photo & style image

Optimize: Image trading off photo content and style match

$$L_1 = \text{Content loss}$$
$$L_2 = \text{Style loss}$$

Photo by J.C.M. Dankers; Style by R. Lichtenstein, *Drowning Girl*

Neural style transfer (Gatys et al., 2016)

Neural style transfer

Photo

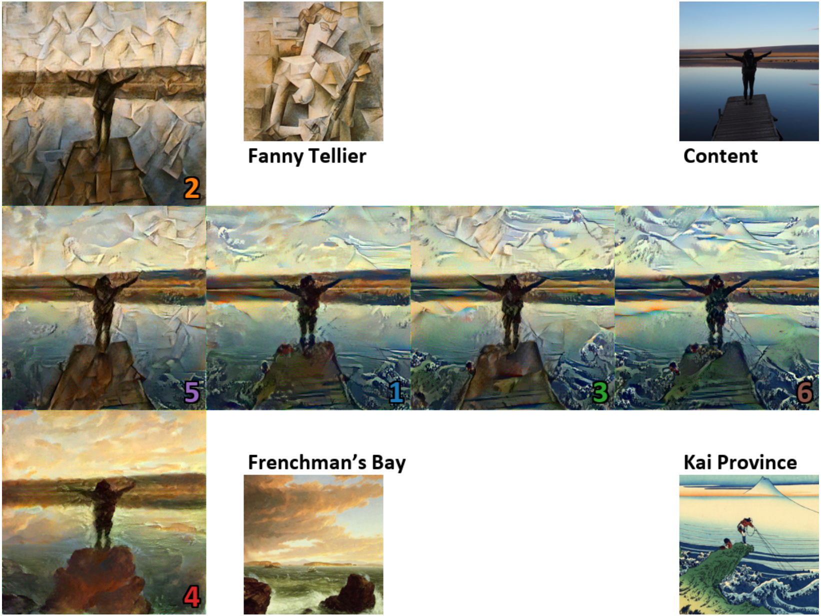
Style

How much style do you want?
10%?
20%?
100%?

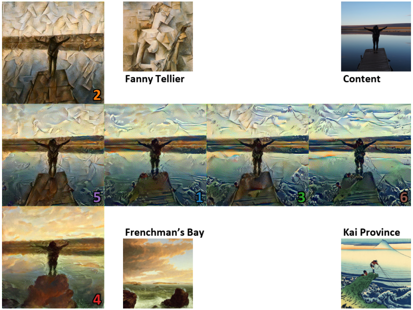
Neural style transfer

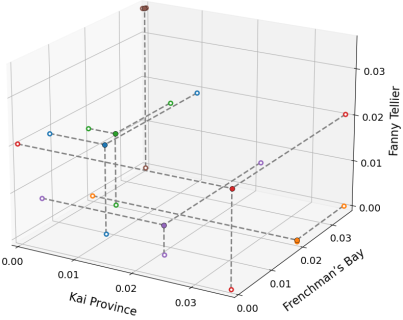
Neural style transfer

Neural multi-style transfer



Neural multi-style transfer





Conclusions

MO optimization to guide decision-making with conflicting goals

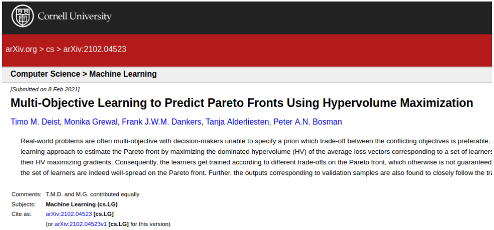
HV gradient ascent offers

- single objective & gradient-based search
- finding diverse sets of solutions close to Pareto front

We proposed MO learning based on HV-maximization

- generates Pareto front approximations per sample
- implemented in PyTorch with prototypes for
  - MO regression
  - medical imaging
  - neural style transfer
- also works for asymmetric Pareto fronts

MO learning literature


Cornell University

arXiv.org > cs > arXiv:2102.04523

Computer Science > Machine Learning

[Submitted on 8 Feb 2021]

Multi-Objective Learning to Predict Pareto Fronts Using Hypervolume Maximization

Timo M. Deist, Monika Grewal, Frank J.W.M. Dankers, Tanja Alderliesten, Peter A.N. Bosman

Real-world problems are often multi-objective with decision-makers unable to specify a priori which trade-off between the conflicting objectives is preferable. learning approach to estimate the Pareto front by maximizing the dominated hypervolume (HV) of the average loss vectors corresponding to a set of learners their HV maximizing gradients. Consequently, the learners get trained according to different trade-offs on the Pareto front, which otherwise is not guaranteed the set of learners are indeed well-spread on the Pareto front. Further, the outputs corresponding to validation samples are also found to closely follow the tr

Comments: T.M.D. and M.G. contributed equally

Subjects: **Machine Learning (cs.LG)**

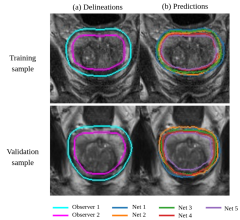
Cite as: arXiv:2102.04523 **[cs.LG]**

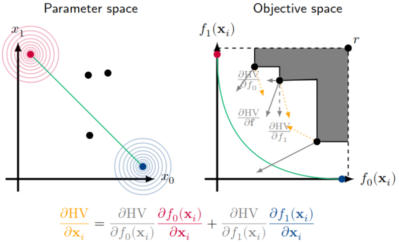
(or arXiv:2102.04523v1 **[cs.LG]** for this version)

- The full method
- Link to PyTorch code
- Comparison to existing methods
- Why one should not learn on average losses.

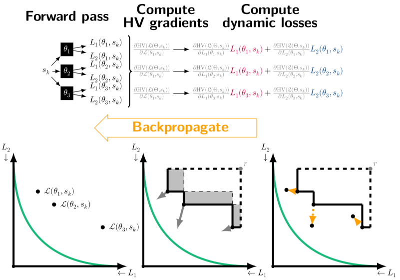
`https://arxiv.org/abs/2102.04523`

Questions?





UHV optimization



MO learning

Appendix

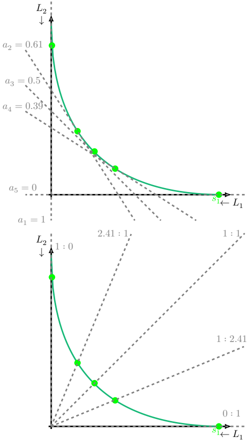
Existing methods

We compared to:

- Linear scalarization

  minimize $\quad a_i L_1(\theta_i, s_k) + (1 - a_i) L_2(\theta_i, s_k)$

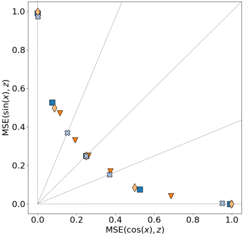
- Pareto MTL (Lin et al., 2019)
- EPO (Mahapatra and Rajan, 2020)



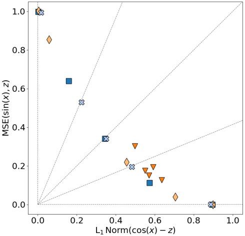
All these methods require knowing the desired trade-offs **before** training.

MO regression

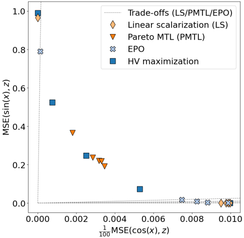
Symmetric losses



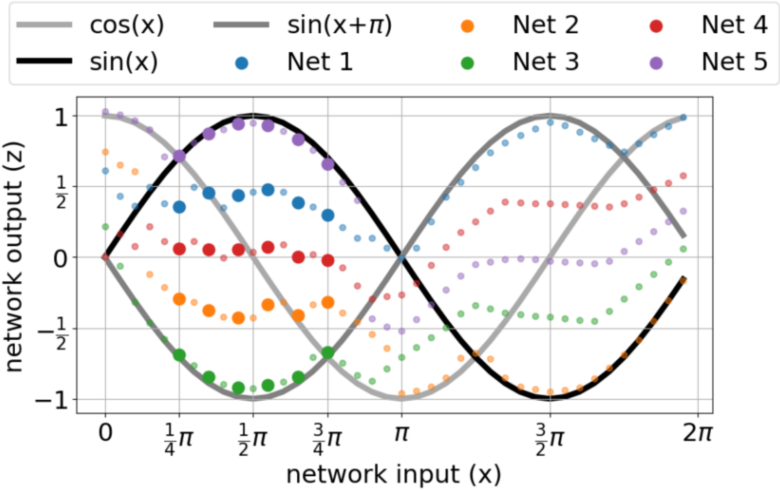
Different losses



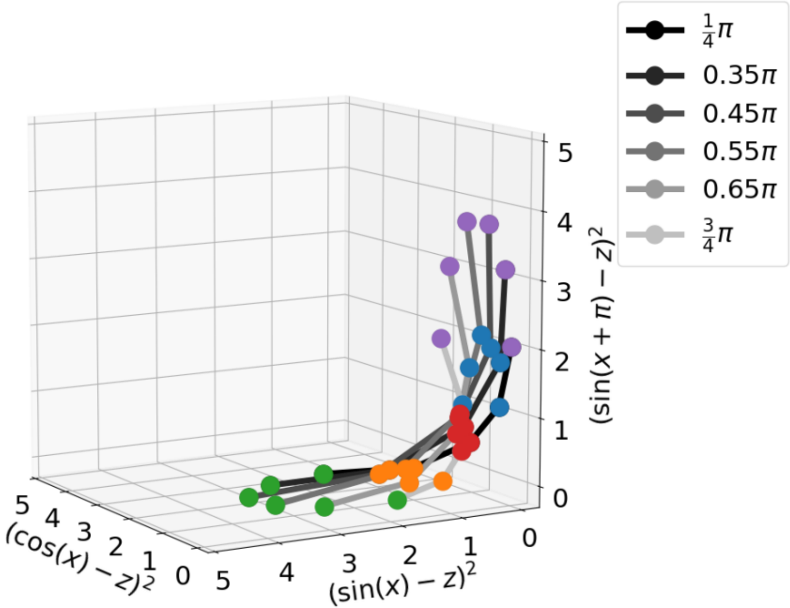
Different scales



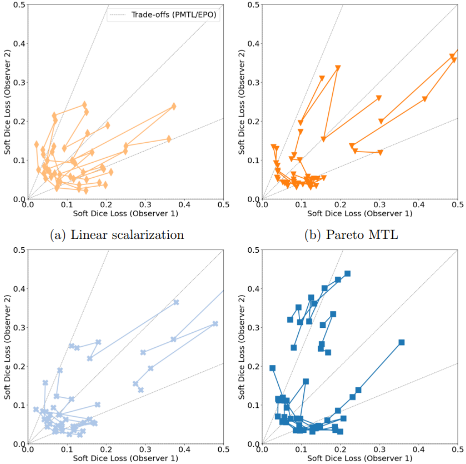
3D MO regression



3D MO regression



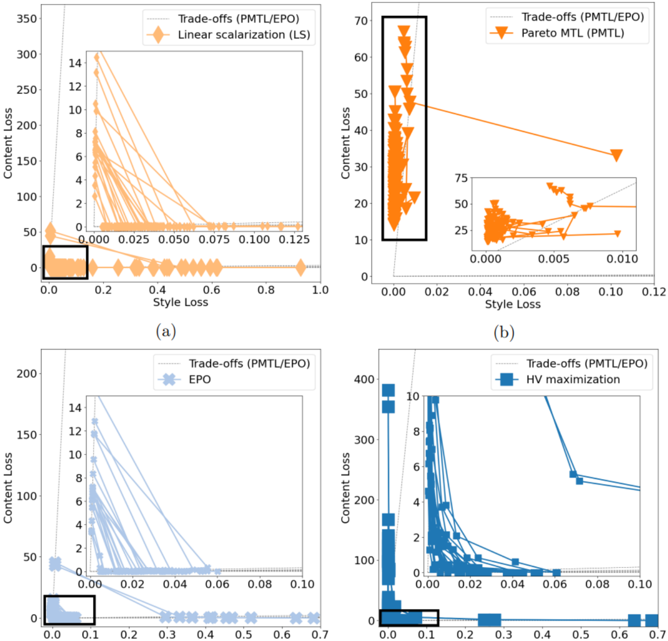
Organ segmentation

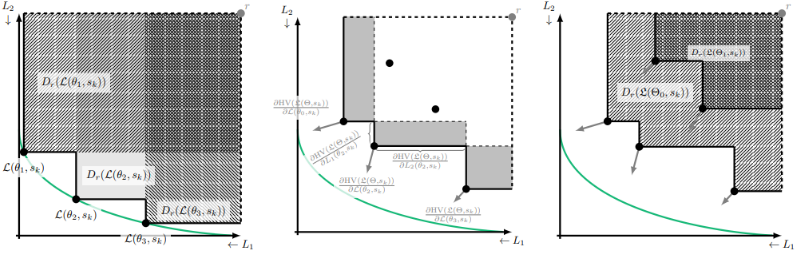


(a) Linear scalarization (b) Pareto MTL

(c) EPO (d) HV maximization

Neural style transfer





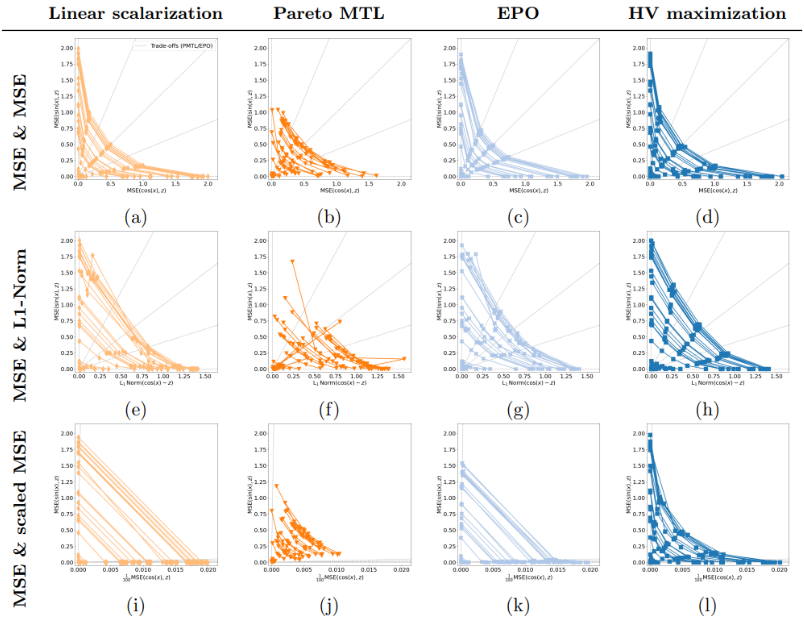
(a) Dominated subspaces (b) HV gradients (c) Domination-ranked fronts

MO learning algorithm

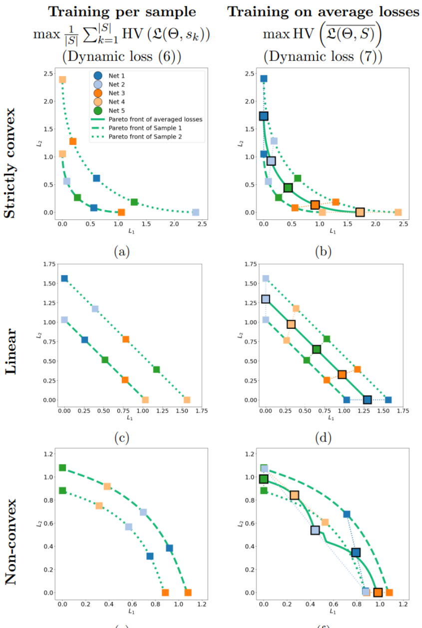
Algorithm 1 Training networks Θ for Pareto front prediction by HV maximization of domination-ranked fronts

Initialize p networks $\Theta = \{\theta_1, \ldots, \theta_p\}$
for each batch $\tilde{S}$ **do**
    for each network θ_i **do**
        for each sample $s_k \in \tilde{S}$ **do**
            Compute loss vector $\mathcal{L}(\theta_i, s_k)$
        end for
    end for
    for each sample $s_k \in \tilde{S}$ **do**
        Stack loss vectors $\mathcal{L}(\theta_i, s_k)$ into $\mathfrak{L}(\Theta, s_k)$
        Sort $\mathfrak{L}(\Theta, s_k)$ into multiple fronts $\mathfrak{L}(\Theta_l, s_k)$ by domination ranking (Section 3.2)
        for each front l **do**
            Compute loss weights $\frac{\partial \mathrm{HV}(\mathfrak{L}(\Theta_{q(i)}, s_k))}{\partial L_j(\theta_i, s_k)} \forall i, j$ using algorithm by Emmerich and Deutz (2014)
        end for
    end for
    for each network θ_i **do**
        Backpropagate on joint loss from Equation (6)
    end for
    Update Θ by stepping into gradient direction
end for

Comparison on asymmetric fronts



Learning per sample vs. on mean loss



Counterexample: asymmetric front when learning on mean loss

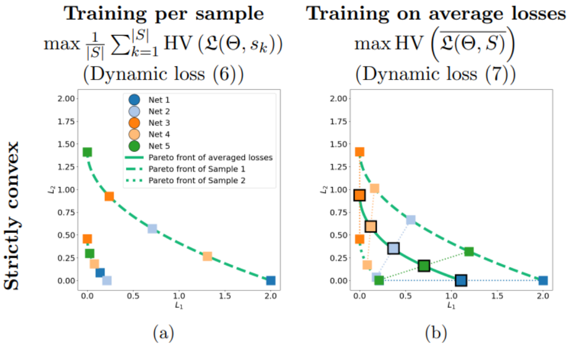


Figure 4: An example of a learning problem with strictly convex Pareto fronts in which HV maximization of average losses does not result in well-distributed outputs on both samples' fronts. HV maximization of each sample's losses (left) and of average losses (right).

Comparison: rescaling losses