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.

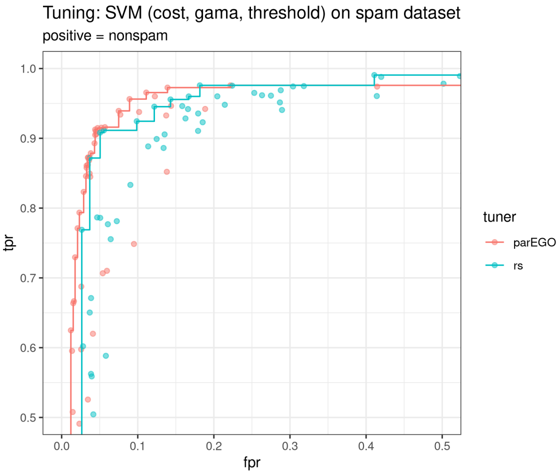
[1] `https://archive.ics.uci.edu/ml/datasets/spambase`

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:

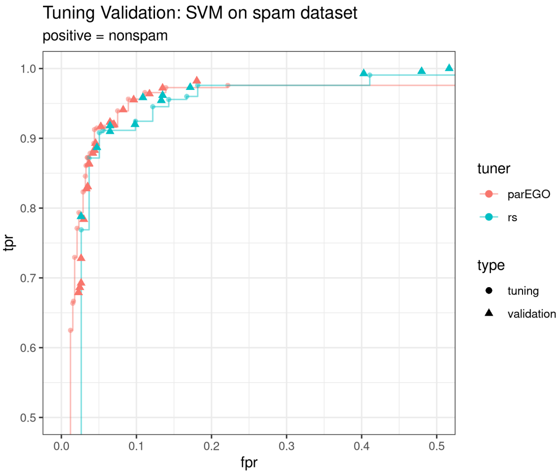
ParEGO:	0.965
random search:	0.959



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.

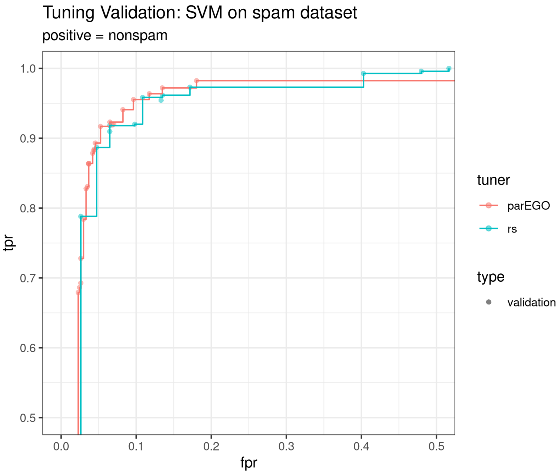


ROC Optimization - Result II

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.

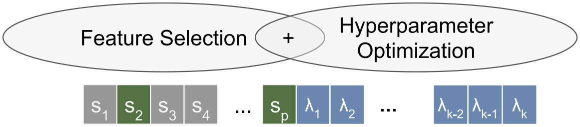
Efficient Models - Example: Feature Selection I

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**.

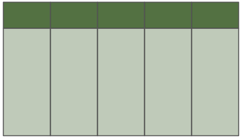


Efficient Models - Example: Feature Selection II

Idea: *Multi-Objective Hyperparameter Tuning and Feature Selection using Filter Ensembles* [Moosbauer, 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

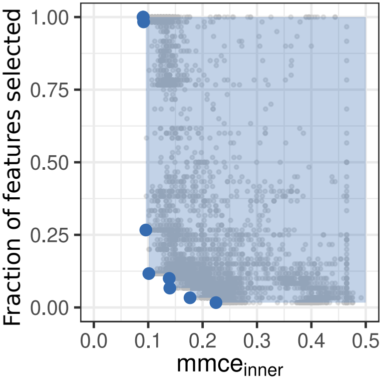
Efficient Models - Example: Feature Selection III

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.

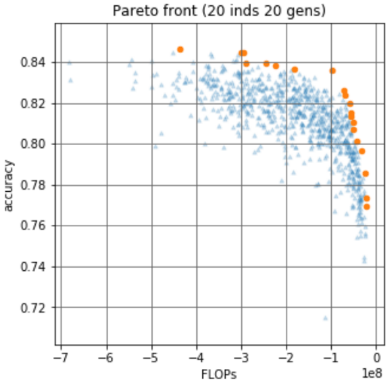
Training: 300 Epochs with a batch size of 128 and initial learning rate of 0.1.

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.

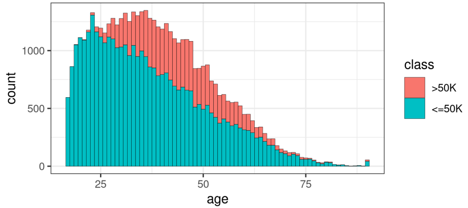


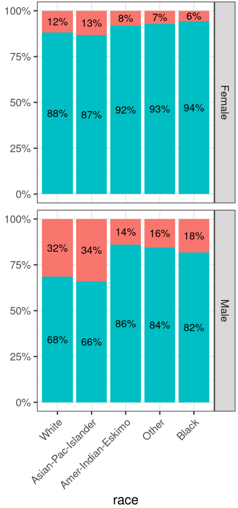
The `growth rate` is the number of output feature maps in each layer of a block

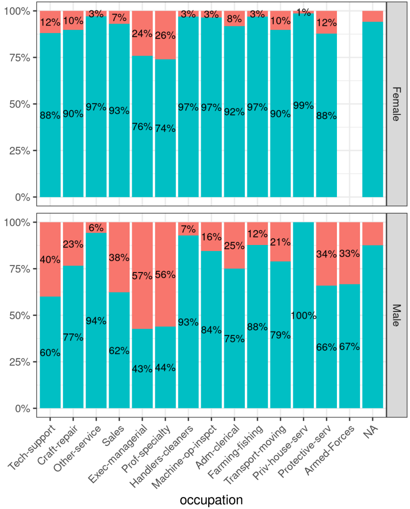
Fair Models - The Adult dataset

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`	$\{2, \ldots, 20\}$
`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

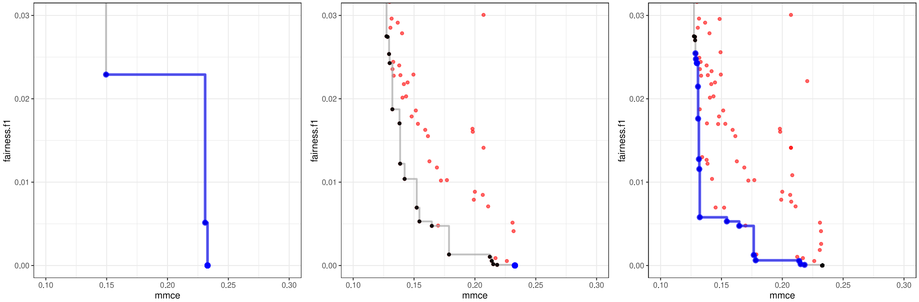


Figure: Pareto fronts after 20, 70 and 120 tuning iterations.

- 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.