I2ML :: EVALUATION AND TUNING

Set-Based Performance Metrics

$J \in \{1, \dots, n\}^m$: m-dimensional index vector for a dataset $\mathcal{D} \in \mathbb{D}_n$, which also induces $\mathcal{D}_J = \left(\mathcal{D}^{(J^{(1)})}, \dots, \mathcal{D}^{(J^{(m)})}\right) \in \mathbb{D}_m$

$\mathbf{y}_J = \left(y^{(J^{(1)})}, \dots, y^{(J^{(m)})}\right) \in \mathcal{Y}^m$: vector of labels

$\mathbf{F}_{J,f} = \left(f(\mathbf{x}^{(J^{(1)})}), \dots, f(\mathbf{x}^{(J^{(m)})})\right) \in \mathbb{R}^{m \times g}$: matrix of prediction scores yielded by a model f

General **performance measure**: $\rho : \bigcup_{m \in \mathbb{N}} (\mathcal{Y}^m \times \mathbb{R}^{m \times g}) \to \mathbb{R}$ maps every m-dimensional label vector $\mathbf{y}_J$ and a matrix of prediction scores $\mathbf{F}_{J,f}$ to a scalar performance value.

$\rho_L(\mathbf{y}, \mathbf{F}) = \sum_{i=1}^m L(\mathbf{y}^{(i)}, \mathbf{F}^{(i)})$: performance measure induced by an arbitrary point-wise loss L

Generalization Error

The **generalization error** GE is the performance of a model induced by $\mathcal{I}_{\boldsymbol{\lambda}}$ from datasets $\mathcal{D}_{\text{train}} \sim (\mathbb{P}_{xy})^{n_{\mathrm{train}}}$ evaluated with performance measure ρ over a dataset $\mathcal{D}_{\text{test}} \sim (\mathbb{P}_{xy})^{n_{\mathrm{test}}}$ when $n_{\text{test}} \to \infty$, i.e.,

$$\mathrm{GE}(\mathcal{I}, \boldsymbol{\lambda}, n_{\mathrm{train}}, \rho) = \lim_{n_{\mathrm{test}} \to \infty} \mathbb{E}\left[\rho\left(\mathbf{y}, \mathbf{F}_{J_{\mathrm{test}}, \mathcal{I}(\mathcal{D}_{\mathrm{train}}, \boldsymbol{\lambda})}\right)\right],$$

where the expectation is taken over both datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$.

Data Splitting and Resampling

$\mathcal{J} = ((J_{\mathrm{train},1}, J_{\mathrm{test},1}), \dots, (J_{\mathrm{train},B}, J_{\mathrm{test},B}))$: **resampling strategy** consisting of B train-test-splits $(J_{\mathrm{train},i}, J_{\mathrm{test},i})$

Estimator of the generalization error $\mathrm{GE}(\mathcal{I}, \boldsymbol{\lambda}, n_{\mathrm{train}}, \rho)$:

$$\widehat{\mathrm{GE}}(\mathcal{I}, \mathcal{J}, \boldsymbol{\lambda}, \rho) = \mathrm{agr}\Big(\rho\Big(\mathbf{y}_{J_{\mathrm{test},1}}, \mathbf{F}_{J_{\mathrm{test},1}, \mathcal{I}(\mathcal{D}_{\mathrm{train},1}, \boldsymbol{\lambda})}\Big), \; \vdots \; \rho\Big(\mathbf{y}_{J_{\mathrm{test},B}}, \mathbf{F}_{J_{\mathrm{test},B}, \mathcal{I}(\mathcal{D}_{\mathrm{train},B}, \boldsymbol{\lambda})}\Big)\Big),$$

where the aggregating function agr is often the mean and $n_{\mathrm{train}} \approx n_{\mathrm{train},1} \approx \dots \approx n_{\mathrm{train},B}$

Resampling Strategies

K-fold cross-validation : splits the data into K roughly equally-sized partitions. Uses each part once as test set and joins the others for training

Leave-one-out cross validation : n-fold cross-validation

Repeated **subsampling** / Monte Carlo cross-validation : for $p \in (0, 1)$ randomly draws K training sets of size $\lfloor p \cdot n \rfloor$ without replacement from the data and uses the data not drawn as the corresponding K test sets

Bootstrap sampling : Similar to repeated subsampling but the training data is randomly drawn with replacement n times

Tuning

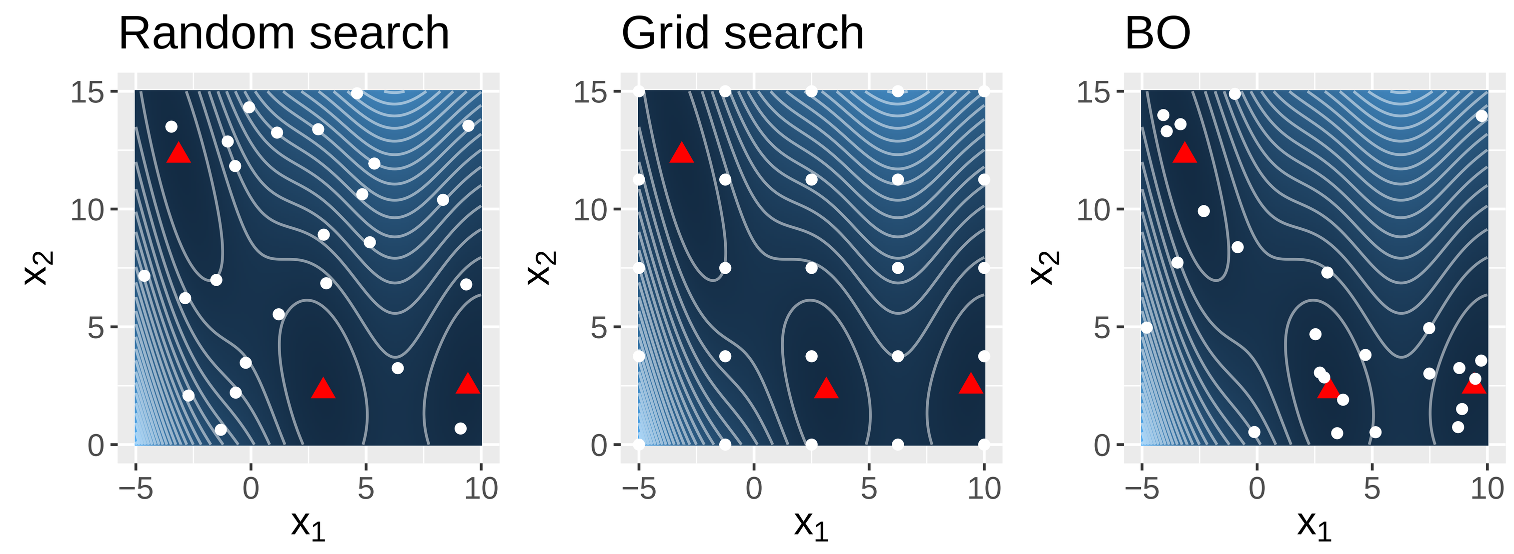
Tuner τ is a mapping which takes a dataset $\mathcal{D} \in \mathbb{D}$, an inducer $\mathcal{I}$, a search space $\tilde{\boldsymbol{\Lambda}}$, a performance measure ρ and a resampling strategy $\mathcal{J}$ and return a hyperparameter configuration $\hat{\boldsymbol{\lambda}} \in \tilde{\boldsymbol{\Lambda}}$ such that

$$\tau(\mathcal{D}, \mathcal{I}, \boldsymbol{\lambda}, \rho, \mathcal{J}) = \hat{\boldsymbol{\lambda}} \approx \boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \tilde{\boldsymbol{\Lambda}}} \widehat{\mathrm{GE}}(\mathcal{I}, \mathcal{J}, \rho, \boldsymbol{\lambda}).$$

The search space $\tilde{\boldsymbol{\Lambda}}$ is a bounded subspace of the hyperparameter space. If the inducer $\mathcal{I}$, the seach space $\tilde{\boldsymbol{\Lambda}}$, the performance measure ρ and the sampling strategy $\mathcal{J}$ are fixed, a self-tuning learner $\mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}} : \mathbb{D} \to \mathcal{H}$, can be derived from a tuner τ such that

$$\mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}} = \mathcal{I}_{\hat{\boldsymbol{\lambda}}}, \text{ i.e., } \mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}}(\mathcal{D}) = \mathcal{I}_{\tau(\mathcal{D}, \mathcal{J}, \mathcal{I}, \rho, \tilde{\boldsymbol{\Lambda}})}(\mathcal{D}).$$

Black-Box Optimization Techniques



Random search : uniformly samples candidates from the search space

Grid search : (uniformly) discretizes the search space and samples candidates from it without replacment

Bayesian optimization : continuously learns a surrogate model of the objective function and leverages it to sample new candidates from the search space while balancing exploration and exploitation

Evolutionary algorithms : are stochastic optimization methods that aim to solve optimization problems by using ideas of natural evolution

Hyperband : is a multi-fidelity method which tries to allocate more budget to promising candidates based on Successive Halving

Nested Resampling

$\mathcal{J}_{B_{\mathrm{outer}}, B_{\mathrm{inner}}} = \left(\mathcal{J}_{\mathrm{outer}}, \left(\mathcal{J}^{(1)}_{\mathrm{inner}}, \dots, \mathcal{J}^{(B_{\mathrm{outer}})}_{\mathrm{inner}}\right)\right)$: **nested resampling strategy** where the outer resampling strategy $\mathcal{J}_{\mathrm{outer}}$ is defined on $\mathcal{D}$ and the inner resampling strategy $\mathcal{J}^{(i)}_{\mathrm{inner}}$ defined on $\mathcal{D}_{\mathrm{outer,train},i}$.

Estimator of the generalization error $\mathrm{GE}(\mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}}, n_{\mathrm{train}})$:

$$\widehat{\mathrm{GE}}(\mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}}, \mathcal{J}_{B_{\mathrm{outer}}, B_{\mathrm{inner}}}) = \mathrm{agr}\Big(\rho\Big(\mathbf{y}_{J_{\mathrm{outer,test},1}}, \mathbf{F}_{J_{\mathrm{outer,test},1}, f_{\mathcal{D}_{\mathrm{outer,train},1}}}\Big), \; \vdots \; \rho\Big(\mathbf{y}_{J_{\mathrm{outer,test},B_{\mathrm{outer}}}}, \mathbf{F}_{J_{\mathrm{outer,test},B_{\mathrm{outer}}}, f_{\mathcal{D}_{\mathrm{outer,train},B_{\mathrm{outer}}}}}\Big)\Big),$$

where $f_{\mathcal{D}_{\mathrm{outer,train},i}} = \mathcal{T}_{\mathcal{I}, \tilde{\boldsymbol{\Lambda}}, \rho, \mathcal{J}^{(i)}_{\mathrm{inner}}}(\mathcal{D}_{\mathrm{outer,train},i})$ and $\mathcal{J}^{(i)}_{\mathrm{inner}}$ has the same type of resampling strategy as $\mathcal{J}$ and $n_{\mathrm{train}} \approx n_{\mathrm{outer,train},1} \approx \dots \approx n_{\mathrm{outer,train},B_{\mathrm{outer}}}$