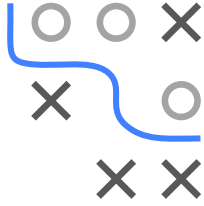
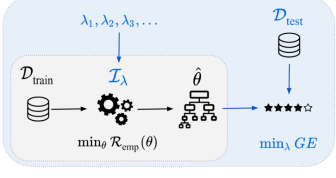
# Introduction to Machine Learning



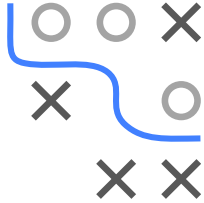
## Hyperparameter Tuning
## Problem Definition




**Learning goals**

- Definition of HPO objective and components
- Understand its properties
- What makes tuning challenging

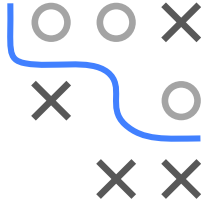
# HYPERPARAMETER OPTIMIZATION



**Hyperparameters (HP)** $\boldsymbol{\lambda}$ are parameters that are *inputs* to learner $\mathcal{I}$ which performs ERM on training data set to find optimal **model parameters** $\boldsymbol{\theta}$. HPs can influence the generalization performance in a non-trivial and subtle way.

**Hyperparameter optimization (HPO)** / **Tuning** is the process of finding a well-performing hyperparameter configuration (HPC) $\boldsymbol{\lambda} \in \tilde{\Lambda}$ for an learner $\mathcal{I}_{\boldsymbol{\lambda}}$.

# OBJECTIVE AND SEARCH SPACE



Search space $\tilde{\Lambda} \subset \Lambda$ with all optimized HPs and ranges:

$$\tilde{\Lambda} = \tilde{\Lambda}_1 \times \tilde{\Lambda}_2 \times \cdots \times \tilde{\Lambda}_I$$

where $\tilde{\Lambda}_i$ is a bounded subset of the domain of the i-th HP $\Lambda_i$, and can be either continuous, discrete, or categorical.

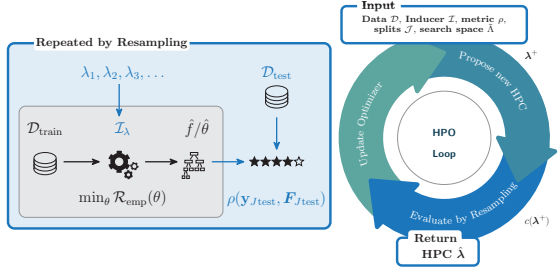
The general HPO problem is defined as:

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

with $\boldsymbol{\lambda}^*$ as theoretical optimum, and $c(\boldsymbol{\lambda})$ is short for estim. gen. error when $\mathcal{I}$, resampling splits $\mathcal{J}$, performance measure $\rho$ are fixed.

# OBJECTIVE AND SEARCH SPACE

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




- Evals are stored in **archive**
  $\mathcal{A} = ((\boldsymbol{\lambda}^{(1)}, c(\boldsymbol{\lambda}^{(1)})), (\boldsymbol{\lambda}^{(2)}, c(\boldsymbol{\lambda}^{(2)})), \ldots)$, with
  $\mathcal{A}^{[t+1]} = \mathcal{A}^{[t]} \cup (\boldsymbol{\lambda}^+, c(\boldsymbol{\lambda}^+))$.
- We can define tuner as function $\tau : (\mathcal{D}, \mathcal{I}, \tilde{\Lambda}, \mathcal{J}, \rho) \mapsto \hat{\boldsymbol{\lambda}}$

# WHY IS TUNING SO HARD?

- Tuning is usually **black box**: No derivatives of the objective are availabe. We can only eval the performance for a given HPC via a computer program (CV of learner on data).
- Every evaluation can require multiple train and predict steps, hence it's **expensive**.
- Even worse: the answer we get from that evaluation is **not exact, but stochastic** in most settings, as we use resampling.
- **Categorical and dependent hyperparameters** aggravate our difficulties: the space of hyperparameters we optimize over can have non-metric, complicated structure.
- Many standard optimization algorithms cannot handle these properties.