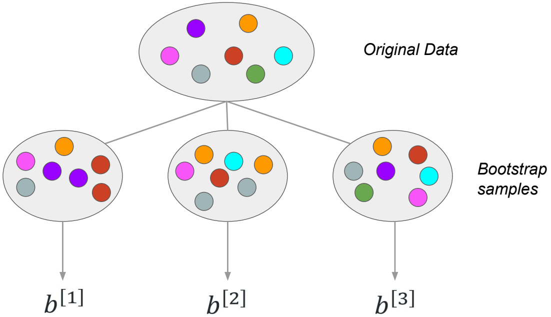
Common Machine Learning Algorithms



CONTENTS

LINEAR MODELS

LINEAR MODELS – FUNCTIONALITY

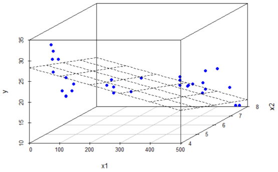
SUPERVISED PARAMETRIC WHITE-BOX

General idea Represent target as function of linear predictor $\boldsymbol{\theta}^T\mathbf{x}$

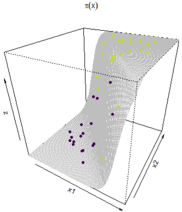
Hypothesis space

$\mathcal{H} = \{f : \mathcal{X} \to \mathbb{R} \mid f(\mathbf{x}) = \phi(\boldsymbol{\theta}^\top \mathbf{x})\}$, with suitable transformation $\phi(\cdot)$, e.g.,

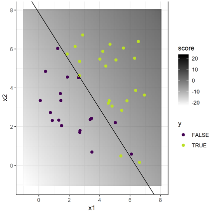
- Identity $\phi(\boldsymbol{\theta}^\top\mathbf{x}) = \boldsymbol{\theta}^\top\mathbf{x}$ $\Rightarrow$ **linear regression**
- Logistic sigmoid function $\phi(\boldsymbol{\theta}^T\mathbf{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T\mathbf{x})} =: \pi(\mathbf{x} \mid \boldsymbol{\theta})$ $\Rightarrow$ **(binary) logistic regression**
  - Probability $\pi(\mathbf{x} \mid \boldsymbol{\theta}) = \mathbb{P}(y = 1 \mid \mathbf{x})$ of belonging to one of two classes
  - Separating hyperplane via decision rule (e.g., $\hat{y} = 1 \Leftrightarrow \pi(\mathbf{x}) > 0.5$)



Linear regression hyperplane



Logistic function for bivariate input



Separating hyperplane for bivariate logistic regression

LINEAR MODELS – FUNCTIONALITY

Empirical risk

- **Linear regression**
  - Typically, based on **quadratic** loss: $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)^2$

    $\Rightarrow$ corresponding to ordinary-least-squares (OLS) estimation
  - Alternatives: e.g., **absolute** or **Huber** loss (both more robust)
- **Logistic regression:** based on **Bernoulli/log/cross-entropy** loss

  $\Rightarrow \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} -y^{(i)} \log\left(\pi\left(\mathbf{x}^{(i)}\right)\right) - (1 - y^{(i)}) \log\left(1 - \pi\left(\mathbf{x}^{(i)}\right)\right)$

Optimization

- For **OLS**: analytically with $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
  (with $\mathbf{X} \in \mathbb{R}^{n \times p}$: matrix of feature vectors)
- For **other loss functions**: numerical optimization

Hyperparameters None

LINEAR MODELS – PRO'S & CON'S

Advantages

+ **simple and fast** implementation; cheap computational costs
+ intuitive **interpretability**: mean influence of features on the output and feature importance
+ fits **linearly** separable data sets very well
+ works well **independent of data size**
+ basis for many machine learning algorithms

Disadvantages

– not suitable for data based on a **non-linear** data generating process $\rightarrow$ **strong simplification** of real-world problems

– **strong assumptions**: data is independent (multi-collinearity must be removed)

– tend to **overfit** (can be reduced by regularization)

– **sensitive to outliers and noisy data**

Simple method with good interpretability for linear problems, but strong assumptions and simplification of real-world problems

LINEAR MODELS – PRACTICAL HINTS

Check assumptions??

This model is very effective if the following assumptions are fulfilled:

- **linearity**: The expected response is a linear combination of the features.
- **homoscedasticity**: The variance of residuals is equal for all features.
- **independence**: All observations are independent of each other.
- **normality**: Y is normally distributed for any fixed value of the features

Implementation

- **R:** `mlr3` learner `LearnerRegrLM`, calling `stats::lm()`
- **Python:** `LinearRegression` from package `sklearn.linear_model`, package for advanced statistical parameters `statsmodels.api`

Regularization

In practice, we often use regularized models in order to **prevent overfitting** or perform feature selection. More details will follow in the subsequent chapter.