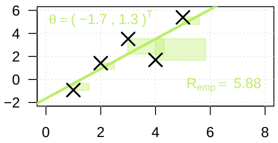
Introduction to Machine Learning

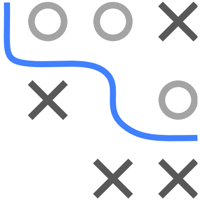
ML-Basics
Components of Supervised Learning



Learning goals

- Know the three components of a learner: Hypothesis space, risk, optimization
- Understand that defining these separately is the basic design of a learner
- Know a variety of choices for all three components

COMPONENTS OF SUPERVISED LEARNING

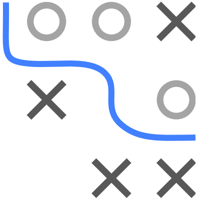


Summarizing what we have seen before, many supervised learning algorithms can be described in terms of three components:

Learning = Hypothesis Space + Risk + Optimization

- **Hypothesis Space:** Defines (and restricts!) what kind of model f can be learned from the data.
- **Risk:** Quantifies how well a specific model performs on a given data set. This allows us to rank candidate models in order to choose the best one.
- **Optimization:** Defines how to search for the best model in the **hypothesis space**, i.e., the model with the smallest **risk**.

COMPONENTS OF SUPERVISED LEARNING

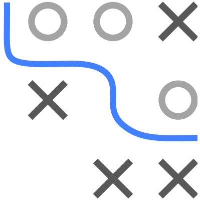


This concept can be extended by the concept of **regularization**, where the model complexity is accounted for in the risk:

Learning = Hypothesis Space + Risk + Optim
Learning = Hypothesis Space + Loss (+ Regularization) + Optim

- For now you can just think of the risk as sum of the losses.
- While this is a useful framework for most supervised ML problems, it does not cover all special cases, because some ML methods are not defined via risk minimization and for some models, it is not possible (or very hard) to explicitly define the hypothesis space.

VARIETY OF LEARNING COMPONENTS



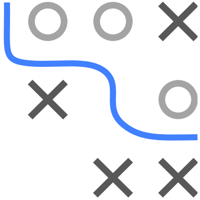
The framework is a good orientation to not get lost here:

Hypothesis Space :
- Step functions
- Linear functions
- Sets of rules
- Neural networks
- Voronoi tesselations
- ...

Risk / **Loss** :
- Mean squared error
- Misclassification rate
- Negative log-likelihood
- Information gain
- ...

Optimization :
- Analytical solution
- Gradient descent
- Combinatorial optimization
- Genetic algorithms
- ...

SUPERVISED LEARNING, FORMALIZED



A **learner** (or **inducer**) $\mathcal{I}$ is a *program* or *algorithm* which

- receives a **training set** $\mathcal{D} \in \mathbb{D}$, and,
- for a given **hypothesis space** $\mathcal{H}$ of **models** $f : \mathcal{X} \to \mathbb{R}^g$,
- uses a **risk** function $\mathcal{R}_{\text{emp}}(f)$ to evaluate $f \in \mathcal{H}$ on $\mathcal{D}$;
  or we use $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$ to evaluate f's parametrization $\boldsymbol{\theta}$ on $\mathcal{D}$
- uses an **optimization** procedure to find

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \mathcal{R}_{\text{emp}}(f) \qquad \text{or} \qquad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}).$$

So the inducer mapping (including hyperparameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$) is:

$$\mathcal{I} : \mathbb{D} \times \boldsymbol{\Lambda} \to \mathcal{H}$$

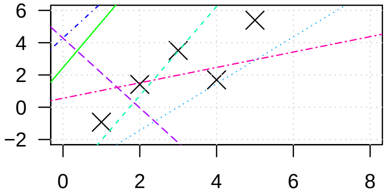
We can also adapt this concept to finding $\hat{\boldsymbol{\theta}}$ for parametric models:

$$\mathcal{I} : \mathbb{D} \times \boldsymbol{\Lambda} \to \Theta$$

EXAMPLE: LINEAR REGRESSION ON 1D

- The **hypothesis space** in univariate linear regression is the set of all linear functions, with $\boldsymbol{\theta} = (\theta_0, \theta_1)^\top$:

$$\mathcal{H} = \{f(\mathbf{x}) = \theta_0 + \theta \mathbf{x} : \theta_0, \theta_1 \in \mathbb{R}\}$$

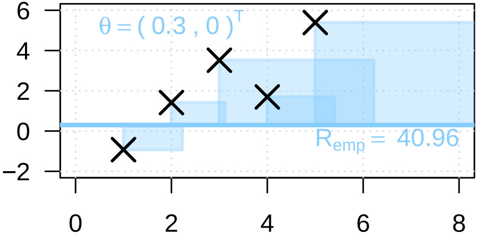


Design choice: We could add more flexibility by allowing polynomial effects or by using a spline basis.

EXAMPLE: LINEAR REGRESSION ON 1D / 2

- We might use the squared error as loss function to our **risk**, punishing larger distances more severely:

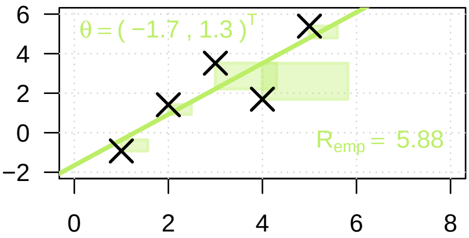
$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y^{(i)} - \theta_0 - \theta_1 \mathbf{x}^{(i)})^2$$

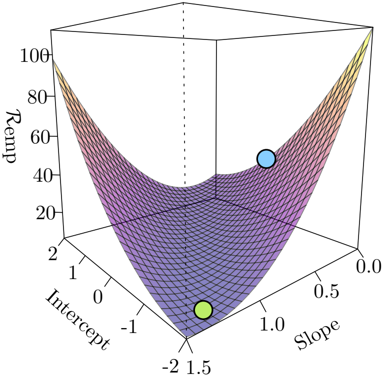


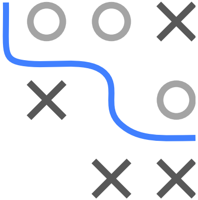
Design choice: Use absolute error / the $L1$ loss to create a more robust model which is less sensitive regarding outliers.

EXAMPLE: LINEAR REGRESSION ON 1D / 3

- **Optimization** will usually mean deriving the ordinary-least-squares (OLS) estimator $\hat{\theta}$ analytically.







Design choice: We could use stochastic gradient descent to scale better to very large or out-of-memory data.

SUMMARY

By decomposing learners into these building blocks:

- we have a framework to better understand how they work,
- we can more easily evaluate in which settings they may be more or less suitable, and
- we can tailor learners to specific problems by clever choice of each of the three components.

Getting this right takes a considerable amount of experience.