

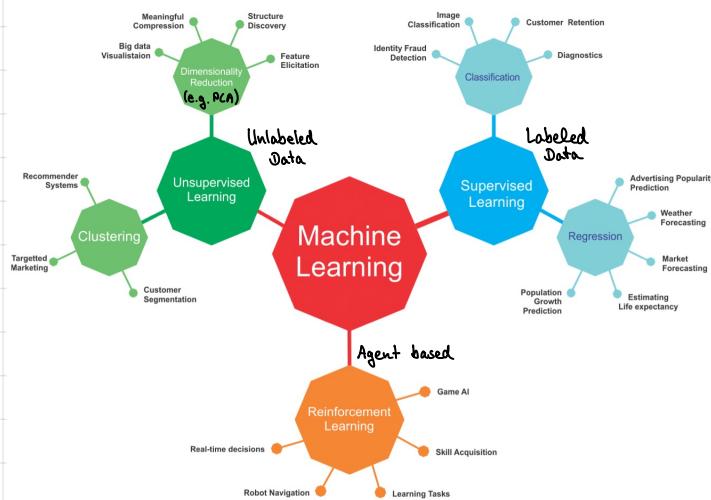
Introduction to Machine Learning

I2ML (oder MaLe)

Legende

- Neues Kapitel
- Unterkapitel
- Definition
- Idee / Setup / Motivation für den darauf folgenden Inhalt
- Wichtiger Teil einer Definition / Aussage / Theorem
- Notation
- Beispiele

Machine Learning Basics



Data in Supervised Learning

	Features x				Target y
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
labeled data	4.3	3.0	1.1	0.1	setosa
	5.0	3.3	1.4	0.2	setosa
	7.7	3.8	6.7	2.2	virginica
	5.5	2.5	4.0	1.3	versicolor
	5.9	3.0	5.1	1.8	?
unlabeled data	4.4	3.2	1.3	0.2	?

n -Zeilen \underline{x}_1 \underline{x}_2 \dots \underline{x}_n
 p -Merkmale

Target variable types:

- Numerical (\mathbb{R})
- Integer (\mathbb{Z})
- Categorical ($\{C_1, \dots, C_g\}$)
- Binary ($\{0, 1\}$)

Regression task
Classification task

Supervised task

Models & Parameters

In most Regression tasks $g = 1$. In Classification tasks $g = \# \text{Categories}$ and f is e.g. a score.

A function $f: \mathcal{X} \rightarrow \mathbb{R}^g$ is called a **Model** (or **Hypothesis**)

! ML requires constraining f to a certain type of function which we call **structural prior**.

The set $\mathcal{H} := \{f \mid f \text{ belongs to the class of the structural prior}\}$ is called **Hypothesis space / Model class**.

Usually \mathcal{H} is constructed as a parametrized family of curves with parameters $\theta := (\theta_1, \dots, \theta_d) \in \Theta$ **Parameter space**

$\Rightarrow \mathcal{H} = \{f_\theta \mid f_\theta \text{ belongs to a family of curves parametrized by } \theta\}$

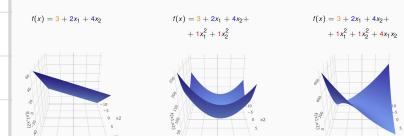
Therefore: ! Finding the optimal model is equivalent to finding the optimal parameters

! The parameter-to-model mapping can be non-injective, i.e. one model is described by different parameter vectors.

Example for Hypothesis / Parameter Spaces

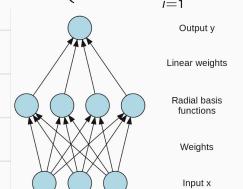
Bivariate quadratic function:

$$\mathcal{H} = \{f : f(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2, \theta \in \mathbb{R}^6\},$$



RBF Network.

$$\mathcal{H} = \left\{ f : f(\mathbf{x}) = \sum_{i=1}^k a_i \rho(\|\mathbf{x} - \mathbf{c}_i\|) \right\},$$



a_i := weight of i -th neuron

c_i := its center vector

$$\rho(\|\mathbf{x} - \mathbf{c}_i\|) := \exp(-\beta \cdot \|\mathbf{x} - \mathbf{c}_i\|^2)$$

is the i -th radial basis function with bandwidth $\beta \in \mathbb{R}$.

Notation

Input/Feature Space: \mathcal{X} All data sets of size n : $\mathbb{D}_n := (\mathcal{X} \times \mathcal{Y})^n$

Output/Target Space: \mathcal{Y} All finite data sets: $\mathbb{D} := \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$

For the actually observed data we denote:

i -th observation: $(\underline{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$

j -th feature vector: $\underline{x}_j := (x_j^{(1)}, \dots, x_j^{(n)})^T$

Observed data set: $\mathbb{D} := ((\underline{x}^{(1)}, y^{(1)}), \dots, (\underline{x}^{(n)}, y^{(n)})) \in \mathbb{D}_n$

Data generating process: $P_{xy} : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$

Encoding of Categorical Features

Let x be a nominal-scaled feature, i.e. $x \in \{C_1, \dots, C_k\}$

We transform the scalar x into a vector: $\sigma(x) := \begin{bmatrix} 1 & \mathbb{I}_{C_1}(x) \\ 1 & \mathbb{I}_{C_2}(x) \\ \vdots & \vdots \\ 1 & \mathbb{I}_{C_k}(x) \end{bmatrix} \in \{0, 1\}^k$

Each entry of $\sigma(x)$ is treated as a separate (binary) feature.

• One-hot-encoding: $k = k$ dummies. So exactly one entry of $\sigma(x)$ is 1 ("hot").

• Dummy-encoding: $k = k-1$ dummies. So at most one entry of $\sigma(x)$ is 1.

↳ cuts redundancy of one-hot-encoding

Necessary if non-singular-matrix is required (e.g. Lin. Reg.)

For an ordinal-scaled feature x we use an encoding that reflects the ordinality, e.g. a sequence of integer values.

Losses & Risk Minimization

Loss

The **Loss function** $L: \mathbb{Y} \times \mathbb{R}^d \rightarrow \mathbb{R}$, $(y, f(x)) \mapsto L(y, f(x))$ quantifies the quality of the prediction on a single observation x .

Risk

Given a model $f(x)$ and a Loss function L , the (theoretical) **Risk** is the expected loss

$$R(f) := \mathbb{E}_{X,Y} [L(y, f(x))] = \int L(y, f(x)) dP_{XY}$$

Empirical Risk

Given a model $f(x)$, a Loss function L and an i.i.d. sample $x^{(1)}, \dots, x^{(n)}$ from P_{XY} , then the **Empirical Risk** is defined as

$$R_{\text{emp}}(f) := \sum_{i=1}^n L(y^{(i)}, f(x^{(i)})) \text{ or (parametrized) } R_{\text{emp}}(\theta) := \sum_{i=1}^n L(y^{(i)}, f(x^{(i)}|\theta))$$

Goal

The best model is the model with the smallest risk. We estimate that by computing $\hat{\theta} := \arg \min_{\theta \in \Theta} R_{\text{emp}}(\theta)$

Optimization

Components of Supervised Learning

Empirical Risk Minimization

The **ERM** problem is $\hat{\theta} := \arg \min_{\theta \in \Theta} R_{\text{emp}}(\theta)$

- ! $\hat{\theta}$ doesn't need to be unique.

Example

- Gradient Descent (if R_{emp} is continuous in θ).
- Normal-equation in case of linear regression

Learner/Inducer

A **Learner/Inducer** I is an algorithm which

- receives a training set $D \subseteq \mathbb{D}$ and
- for a given hypothesis space H of models $f: \mathbb{X} \rightarrow \mathbb{R}^d$
- uses a risk function $R_{\text{emp}}(f)$ or $R_{\text{emp}}(\theta)$ to evaluate $f \in H$ on D .
- uses an optimization procedure to find \hat{f} or $\hat{\theta}$

So the inducer is a mapping $I: D \times \Lambda \rightarrow H$ or $I: D \times \Lambda \rightarrow \Theta$

Hyperparameterspace for control settings λ

Framework

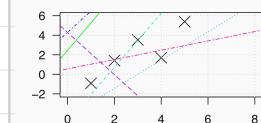
$$\begin{aligned} \text{Learning} &= H + \text{Risk} + \text{Optimization} \\ &= H + \text{Loss} (+\text{Regularization}) + \text{Optimization} \end{aligned}$$

Hypothesis Space :	<ul style="list-style-type: none"> Step functions Linear functions Sets of rules Neural networks Voronoi tessellation ... 	Risk / Loss :	<ul style="list-style-type: none"> Mean squared error Misclassification rate Negative log-likelihood Information gain ... 	Optimization :	<ul style="list-style-type: none"> Analytical solution Gradient descent Combinatorial optimization Genetic algorithms ...
--------------------	---	---------------	--	----------------	--

! Useful framework, but doesn't cover all special cases. E.g. some ML methods are not defined via risk minimization.

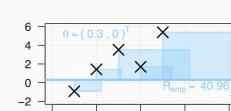
Example

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



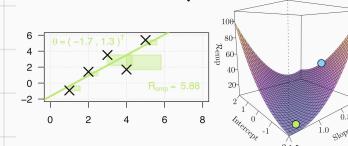
Alt: Polynomials, Splines

$$R_{\text{emp}}(\theta) = \sum_{i=1}^n (y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2$$



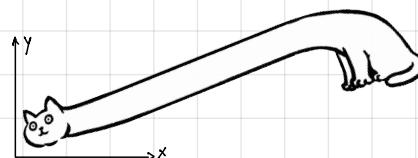
Alt: Absolute error, L1 Loss-function Alt: Stochastic gradient descent

Opt. method: Deriving $\hat{\theta}$ analytically.



$\theta = (-0.3, 0)^T$ $R_{\text{emp}} = 5.88$

Supervised Regression



Design matrix

Let $\underline{x}_1, \dots, \underline{x}_p$ be our p feature vectors, then we call the matrix

$$X = \begin{pmatrix} 1 & | & | & | \\ | & \underline{x}_1 & \dots & \underline{x}_p \\ 1 & | & | & | \end{pmatrix} \in \mathbb{R}^{n \times (p+1)}$$

Furthermore, we define $\underline{x} := (1, x_1, \dots, x_p)^T \in \mathbb{R}^{p+1}$ and $\underline{\theta} = (\theta_0, \theta_1, \dots, \theta_p) \in \mathbb{R}^{p+1}$

Setup

We predict $y \in \mathbb{R}$ as linear combinations of features

$$\hat{y} = f(\underline{x}) = \underline{\theta}^T \underline{x} = \theta_0 + \theta_1 x_1 + \dots + \theta_p x_p$$

This results in the hypothesis space $\mathcal{H} = \{f(\underline{x}) = \underline{\theta}^T \underline{x} \mid \underline{\theta} \in \mathbb{R}^{p+1}\}$

A typical choice for the Loss-function is L2-Loss.

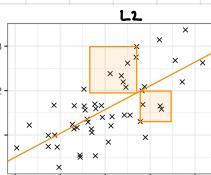
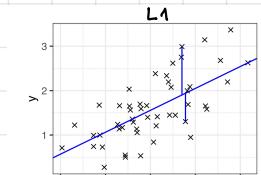
This results in $\mathcal{R}_{\text{emp}}(\underline{\theta}) = \sum_{i=1}^n (y^{(i)} - \underline{\theta}^T \underline{x}^{(i)})^2$.

L2-Loss / Squared Error

$$\mathcal{L}(y, f(\underline{x})) = \frac{1}{2} (y - f(\underline{x}))^2 \quad \text{or} \quad \mathcal{L}(y, f(\underline{x})) = (y - f(\underline{x}))^2$$

Properties: Convex; differentiable everywhere

Problem: Outliers have a huge effect on L2.

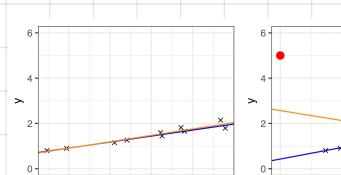


L1-Loss / Absolute Error

$$\mathcal{L}(y, f(\underline{x})) = |y - f(\underline{x})|$$

Properties: Convex; Robust against outliers

Problem: Not differentiable, slower to optimize



Optimization

We want to find $\hat{\underline{\theta}} = \arg \min_{\underline{\theta} \in \mathbb{R}^p} \sum_{i=1}^n (y^{(i)} - \underline{\theta}^T \underline{x}^{(i)})^2 = \arg \min_{\underline{\theta} \in \mathbb{R}^p} \|\underline{y} - X\underline{\theta}\|_2^2$

This results in the ordinary-least-squares estimator:

$$\hat{\underline{\theta}} = (X^T X)^{-1} X^T \underline{y} \quad (\text{MLE})$$

ANALYTICAL OPTIMIZATION – PROOF

$$\begin{aligned} \mathcal{R}_{\text{emp}}(\underline{\theta}) &= \sum_{i=1}^n (y^{(i)} - \underline{\theta}^T \underline{x}^{(i)})^2 = \|\underline{y} - X\underline{\theta}\|_2^2, \quad \underline{\theta} \in \mathbb{R}^p \text{ with } p := p+1 \\ 0 &= \frac{\partial \mathcal{R}_{\text{emp}}(\underline{\theta})}{\partial \underline{\theta}} \quad (\text{sum notation}) \\ 0 &= \frac{\partial}{\partial \underline{\theta}} \sum_{i=1}^n e_i^2 \quad | \text{ sum and chain rule} \\ 0 &= \sum_{i=1}^n \frac{\partial e_i^2}{\partial \underline{\theta}_i} \frac{\partial \underline{\theta}_i}{\partial \underline{\theta}} \\ 0 &= \sum_{i=1}^n 2e_i(-1)(\underline{x}^{(i)})^T \\ 0 &= \sum_{i=1}^n (y^{(i)} - \underline{\theta}^T \underline{x}^{(i)}) (\underline{x}^{(i)})^T \\ \sum_{i=1}^n \underline{\theta}^T \underline{x}^{(i)} (\underline{x}^{(i)})^T &= \sum_{i=1}^n y^{(i)} (\underline{x}^{(i)})^T \quad | \text{ transpose} \\ \underline{\theta}^T \sum_{i=1}^n (\underline{x}^{(i)})^T \underline{x}^{(i)} &= \sum_{i=1}^n \underline{x}^{(i)} y^{(i)} \\ \underline{\theta} &= \underbrace{\begin{pmatrix} \underline{x}^T \underline{x}^{-1} & \underline{x}^T & y \\ \hline \bar{p} \times \bar{p} & \bar{p} \times n & n \times 1 \end{pmatrix}}_{\bar{p} \times 1} \end{aligned}$$

Polynomial Regression

Setup General Linear Model

We predict $y \in \mathbb{R}$ as linear combinations of basis functions ϕ_j

$$\hat{y} = f(\underline{x}) = \theta_0 + \theta_1 \phi_1(x_1) + \dots + \theta_p \phi_p(x_p)$$

If we set $\phi_j \equiv \text{id}_x$ we get the usual linear regression model.

! This model is linear in parameters $\underline{\theta}$ but not in covariates/features \underline{x} .

$$f(ax + bx^*; \underline{\theta}) \neq a f(\underline{x}; \underline{\theta}) + b f(\underline{x}^*; \underline{\theta}) \text{ but } f(\underline{x}; a\underline{\theta} + b\underline{\theta}^*) = a f(\underline{x}; \underline{\theta}) + b f(\underline{x}; \underline{\theta}^*)$$

The design matrix now is defined as $X = \begin{pmatrix} 1 & | & | & | \\ | & \phi_1(x_1) & \dots & \phi_p(x_p) \\ 1 & | & | & | \end{pmatrix}$

Setup Polynomial Regression

In the Polynomial Regression we choose the basis functions $\phi^{(d)}: \mathbb{R} \rightarrow \mathbb{R}, x_j \mapsto \sum_{k=1}^d \beta_k x_j^k$

$$\text{This results in } f(\underline{x}) = \theta_0 + \theta_1 \phi^{(d)}(\underline{x}) = \theta_0 + \sum_{k=1}^d \theta_{1,k} x^k \quad \text{where } \theta_{1,k} := \theta_1 \cdot \beta_k$$

$$X = \begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^d \\ | & | & | & \vdots & | \\ 1 & x^{(n)} & (x^{(n)})^2 & \dots & (x^{(n)})^d \end{pmatrix}, \quad \underline{\theta} \in \mathbb{R}^{d+1}$$

Supervised Classification



Setup

Classification tasks aim at predicting a discrete output

$$y \in \mathcal{Y} = \{C_1, \dots, C_g\}$$

with $2 \leq g < \infty$ given Data D .

We encode \mathcal{Y} as:

- binary case $g=2$: $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$.

- multiclass case $g > 2$: $\mathcal{Y} = \{1, \dots, g\}$

! Our model $f: \mathcal{X} \rightarrow \mathbb{R}^g$ outputs scores/probabilities and not classes.

↳ Continuous function is easier to optimize than discrete-valued functions.

↳ Scores/Probabilities contain more information and can be transformed into classes. But this transformation is non-injective.

Approaches to construct classifiers

Generative Approach

Idea: Assume that the data generating process for x depends on y .

Then we want to answer "Which y tends to have x like that?"

$$\pi_k(x) = P(y=k|x) = \frac{P(x|y=k) \cdot P(y=k)}{P(x)} = \frac{P(x|y=k) \cdot \pi_k}{\sum_j P(x|y=j) \cdot \pi_j} \propto P(x|y=k) \cdot \pi_k$$

with π_j being a prior estimated from data, by using e.g. relative frequencies.

Discriminant Approach

Idea: Use empirical risk minimization with a suitable Loss-function

We want to answer "What is the best prediction for y given x ?"

$$\hat{f} = \arg \min_{f \in \mathcal{H}} R_{\text{emp}}(f) = \arg \min_{f \in \mathcal{H}} \sum_{i=1}^n L(y^{(i)}, f(x^{(i)}))$$

Examples

Logistic Regression (Discriminant)

$$\mathcal{H} = \{\pi: \mathcal{X} \rightarrow [0, 1] \mid \pi(x) = s(\theta^T x)\}$$

$$\text{with } s(\theta^T x) = \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)} = \frac{1}{1 + \exp(-\theta^T x)}$$

Logistic/Bernoulli Loss function:

$$L(y, \pi(x)) = -y \log(\pi(x)) + (1-y) \log(1-\pi(x))$$

Optimization is done numerically (GD).



Penalizes heavier the more confident we are in a wrong prediction.

Scoring classifier

Construct g discriminant/scoring functions $f_1, \dots, f_g: \mathcal{X} \rightarrow \mathbb{R}$.

We choose the class with the maximum score, i.e. $h(x) := \arg \max_{k \in \{1, \dots, g\}} f_k(x)$

Special case $g=2$:

One discriminant/scoring function is sufficient. $f(x) = f_1(x) - f_{-1}(x)$ (Note: $\mathcal{Y} = \{-1, 1\}$)

$$h(x) = \text{sgn}(f(x)) \text{ and } |f(x)| \text{ is called confidence.}$$

Probabilistic classifier

Construct g probability functions $\pi_1, \dots, \pi_g: \mathcal{X} \rightarrow [0, 1]$ with $\sum_{k=1}^g \pi_k = 1$.

We choose the class with the maximum score, i.e. $h(x) := \arg \max_{k \in \{1, \dots, g\}} \pi_k(x)$

Special case $g=2$:

One discriminant/scoring function is sufficient. $\pi(x) = 1 - \pi_0(x)$ (Note: $\mathcal{Y} = \{0, 1\}$)

$$h(x) = \mathbb{1}[\pi(x) \geq c]$$

with c being a specified threshold.

Linear Classifier

If the discriminant/scoring functions f_1, \dots, f_g can be specified as linear functions, i.e. $f_k(x) = w_k^T x + b_k$ with g being a rank-preserving, monotone transformation, then we call f_1, \dots, f_g linear classifiers.

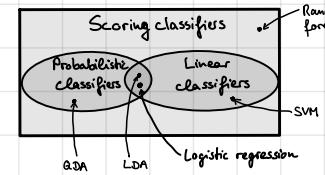
The decision boundaries are hyperplanes.

A decision region for class k is defined as $\mathcal{X}_k := \{x \in \mathcal{X} \mid h(x) = k\}$

The decision boundaries are defined as hypersurfaces in \mathcal{X} with tied maximal score:

$$\{x \in \mathcal{X} \mid \exists i, j \in \{1, \dots, g\}: f_i(x) = f_j(x) \wedge \forall k \neq i, j: f_i(x) \geq f_k(x) \wedge f_j(x) \geq f_k(x)\}$$

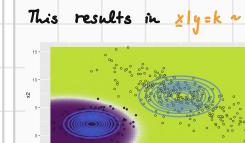
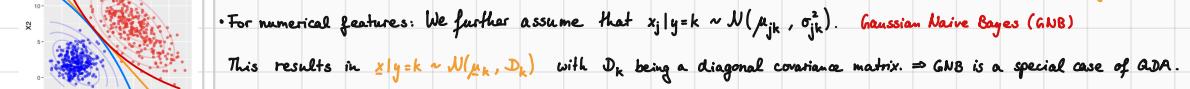
Special case $g=2$ with threshold c the decision boundary is $\{x \in \mathcal{X} \mid f(x) = c\}$



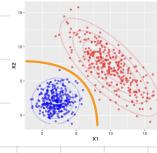
Naive Bayes (Generative)

In Naive Bayes we assume that given class y the features are conditionally independent, i.e. $p(x|y=k) = \prod_{j=1}^m p(x_j|y=k)$

For numerical features: We further assume that $x_j|y=k \sim N(\mu_{jk}, \sigma_{jk}^2)$. Gaussian Naive Bayes (GNB)



This results in $x|y=k \sim N(\mu_k, \Sigma_k)$ with Σ_k being a diagonal covariance matrix. \Rightarrow GNB is a special case of QDA.



For categorical features: We further assume that $x_j|y=k$ is multinomial distributed, i.e. $p(x_j|y=k) \propto \prod_{m=1}^M p_{kj}^{n_m}$

with n_m / p_{kj} being the absolute/relative frequency of value m in feature j restricted to class k . Multinomial Naive Bayes

Note: The multinomial Naive Bayes classifier is a linear classifier.

? If $\exists m: p_{kj} = 0$, then we replace the 0 with a small value (Laplace Smoothing).

Performance Evaluation



Generalization Error

Generalization Error for a Fixed Model

Let $(x, y) \sim P_{xy}$, \hat{f} a fixed model and L a Loss-function.

We define the Generalization error of a fixed model \hat{f} as

$$GE(\hat{f}, L) := E_{xy} [L(y, \hat{f}(x))]$$

If there exists a dedicated test set $D_{\text{test}} \subseteq D_n$ then we can estimate $\widehat{GE}(\hat{f}, L) := \frac{1}{m} \sum_{(x,y) \in D_{\text{test}}} L(y, \hat{f}(x))$

! Rarely possible, because at the end we fit our model on the complete data set. So no data is left available for D_{test} .

Example

$MSE(\hat{f}) = E[(y - \hat{f}(x))^2]$ is a GE of \hat{f} with L2-Loss.

$$E[(y - \hat{f}(x))^2] = \text{Var}(\hat{f}(x)) + \underbrace{\text{Bias}(\hat{f}(x))^2}_{\text{irreducible noise}} + \text{Var}(E[\cdot])$$

Setup

We are interested in evaluating our (final) model \hat{f} . But because this is rarely possible we evaluate the next best thing, i.e. the inducer. To do so, we might need to define specific measures.

Inner and Outer Loss

Sometimes we want to use a different performance measure to evaluate the model/inducer, then we used to construct the model, i.e. the Loss-function.

We call the Loss-function that we used for ERM inner loss.

We call the performance evaluation function outer loss.

! The outer loss doesn't need to be a pointwise Loss-function

Set-based Performance Metrics

Sometimes we evaluate a model performance based on the set of predictions as a whole and not on observation based losses. We use a metric

$$\rho: \bigcup_{m \in \mathbb{N}} (Y^m \times \mathbb{R}^{m \times g}) \rightarrow \mathbb{R}, \quad (y, F) \mapsto \rho(y, F), \quad \text{where } F \text{ is defined as } F_{D_{\text{test}}, \hat{f}} := \begin{bmatrix} \hat{f}(x^{(1)}) \\ \vdots \\ \hat{f}(x^{(m)}) \end{bmatrix} \in \mathbb{R}^{m \times g} \quad \text{and } |D_{\text{test}}| = m.$$

Pointwise loss L can be seen as a special case of a metric ρ

$$\text{with } \rho_L(y, F) = \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, F^{(i)}) = \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, \hat{f}(x^{(i)})) = \underbrace{\frac{1}{m} \sum_{i=1}^m L(y^{(i)}, \hat{f}(x^{(i)}))}_{= \widehat{GE}(\hat{f})}$$

Example

In Logistic Regression we use the Logistic/Bernoulli Loss-function as inner Loss. For performance evaluation, i.e. outer loss, we can use the AUC.

Connection between both GE's

We can only estimate GE with $n_{\text{train}} = |D_{\text{train}}| < |D| = n$ which could result in a pessimistic bias for \widehat{GE} , i.e. we overestimate the GE.

Using a pointwise losses ρ_L we can rewrite the GE for an inducer as

$$GE(I, \lambda, n_{\text{train}}, \rho_L) = E_{D_{\text{train}}, I(D_{\text{train}}, \lambda)} [L(y, I(D_{\text{train}}, \lambda)(x))]$$

! No limit, b.c. that was only needed for set-based-performance-metric.

Using that, we can rewrite the GE for a fixed model \hat{f} that was trained on $D_{\text{train}} = D$, i.e. $\hat{f} = I(D_{\text{train}}, \lambda)$, as

$$GE(\hat{f}, L) = GE(I, \lambda, n_{\text{train}}, \rho_L | D_{\text{train}}) \quad \text{or equivalently}$$

$$E_{x,y} [L(y, \hat{f}(x))] = E_{D_{\text{train}}, I(D_{\text{train}}, \lambda)} [L(y, I(D_{\text{train}}, \lambda)(x)) | D_{\text{train}}]$$

Generalization Error for Inducer

Let I_λ be an Inducer on n_{train} points from P_{xy} and let ρ be a performance metric (outer loss).

We define the Generalization Error for the Inducer I as

$$GE(I, \lambda, n_{\text{train}}, \rho) := \lim_{n_{\text{test}} \rightarrow \infty} E_{D_{\text{train}}, D_{\text{test}}, I(D_{\text{train}}, \lambda)} [\rho(y, F_{D_{\text{test}}, I(D_{\text{train}}, \lambda)})]$$

with $n_{\text{train}} = |D_{\text{train}}|$.

Idea: Quality of models when fitted with I_λ on n_{train} points from P_{xy} .

! Expected value over D_{train} and D_{test} sampled independently.

Rarely possible, therefore we use resampling methods to obtain an estimate for $GE(I, \lambda, n_{\text{train}}, \rho)$.

! Some ρ might only converge for $n_{\text{test}} \rightarrow \infty$, which is why we need the limit and why we take the expectation over $D_{\text{test}} \in \mathcal{D}_{n_{\text{test}}}$ instead of $(x, y) \sim P_{xy}$

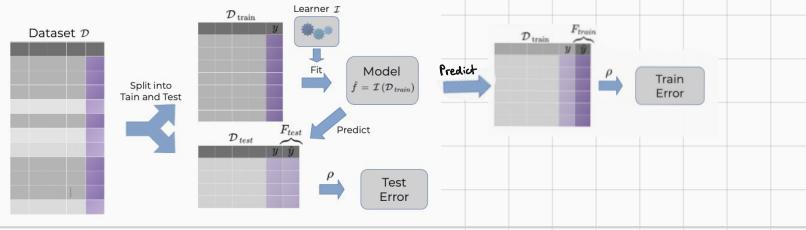
Training- and Test-Error

Training Error and Test Error

Let $\hat{f} = \mathcal{I}(D_{\text{train}}, \lambda)$ and let p be a performance metric. We define the

- Training Error as $p(y_{\text{train}}, F_{D_{\text{train}}, \hat{f}})$
- Test Error as $p(y_{\text{test}}, F_{D_{\text{test}}, \hat{f}})$

! Choose D_{test} and D_{train} , s.t. $D_{\text{test}} \cap D_{\text{train}} = \emptyset$ to avoid optimistic Bias.



Train Error vs. Test Error

Goodness-of-fit measures - like R^2 , \mathcal{L} , AIC, BIC, deviance - are based on training error but are based on distributional assumptions and limited data - therefore hard to use for high-dimensional or more complex data.

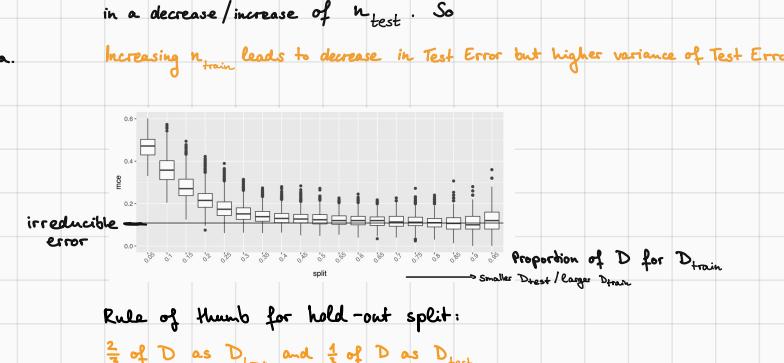
- Decrease of n_{train} \Rightarrow Increase of Test Error (in general) b.c. model generalizes better with more training data and worse with less training data.
- Increase of complexity \Rightarrow Decrease of Training Error (in general) b.c. it becomes easier to learn all patterns on small training data sets or with more flexibility.
- Decrease of n_{test} \Rightarrow Increase of variance of test error.

Bias-Variance - Tradeoff

The GE of the inducer can be estimated by the Test Error.

The quality of this estimate is based on the Bias(ϵ) and the Variance of the Test Error. (ϵ) Smaller Test Error means less Bias.

These are influenced by the complexity and amount of training data.



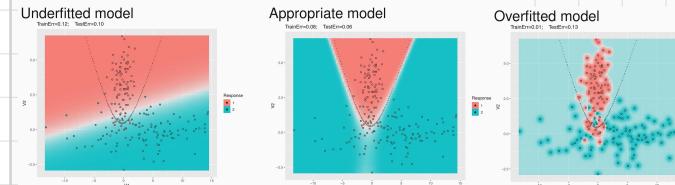
Overfitting and Underfitting

Underfitting occurs when a model can't reflect the true shape of underlying function (given the data)

\hookrightarrow High Test Error and high Training Error.

Overfitting occurs when a model reflects noise or artifacts from D_{train} which do not generalize.

\hookrightarrow Small Train Error and high Test Error.



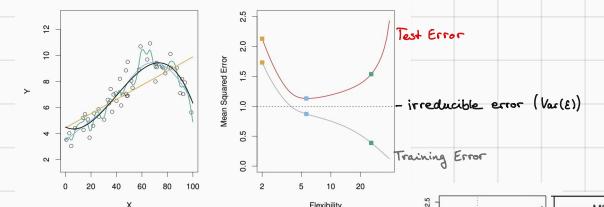
Overfitting is influenced by

- Complexity of Model, i.e. $\dim(\mathcal{H})$
- Amount of Training Data, i.e. n_{train}
- Dimensionality of Feature Space, i.e. $\dim(X)$
- Irreducible Noise, i.e. aleatoric uncertainty

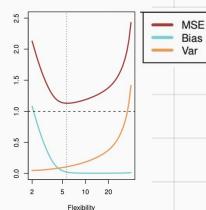
Overfitting can be avoided by regularization methods:

- Constrain \mathcal{H} by using less complex model classes
- Higher amount or better quality of training data
- Use Feature Selection/Engineering and keep only the features that carry a lot of useful information.
- Occam's razor: If two models have similar GE, prefer the simpler one.

Influence of complexity:



We can deconstruct the MSE of the Test Error into its Variance and Bias:
We can see that we need to trade-off in order to minimize the MSE.



Resampling



Setup

We want to estimate $GE(I, \lambda, n, p_c) = GE(\hat{f}, L)$ but doing so with a single hold-out-split results in a high pessimistic Bias or high Variance.

So instead we split repeatedly and average the result. This way we reduce the variance from small test sets.

Resampling Strategies

We choose $J_{\text{train}} \in \{1, \dots, n\}^{n_{\text{train}}}$ and $J_{\text{test}} \in \{1, \dots, n\}^{n_{\text{test}}}$.
"Choose J_{train} indices from $\{1, \dots, n\}$ with replacing"

We define $J := (J_{\text{train},1}, J_{\text{test},1}), \dots, (J_{\text{train},B}, J_{\text{test},B})$

with $B \in \mathbb{N}$ and $n_{\text{train},1} \approx n_{\text{train},2} \approx \dots \approx n_{\text{train},B} \approx n_{\text{train}}$

Our estimate is then

$$\widehat{GE}(I, \lambda, J, p_c) = \frac{1}{B} \sum_{k=1}^B p(y_{J_{\text{test},k}}, F_{J_{\text{test},k}, I(D_{\text{train},k}, \lambda)})$$

Cross Validation

1. Split the data into k equally-sized partitions.

1.1. Stratification. For unbalanced target classes perform the split per class and rejoin the data. Preserves the target distr.

2. Each set is test set once, the rest is used for training \rightarrow Results in k test errors (and training errors)

3. Average the test errors

LOO-CV

! 5 or 10 fold CV's are common. n -fold CV is also called Leave-one-out-CV \rightarrow Nearly unbiased but high variance

! The test errors are not independent b.c. the splits aren't independent \rightarrow There is no unbiased estimate for $V[\widehat{GE}(I, \dots)]$.

Leave-one-Object-Out

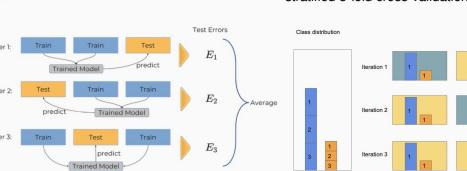
This method is used when there are multiple observations per object \rightarrow Data is not i.i.d. anymore.

The data of one object should either be in the training set or test set ! Not in both \rightarrow Otherwise \widehat{GE} is biased.

This can be achieved by using CV on objects, i.e. split the objects into k groups...

Example CV

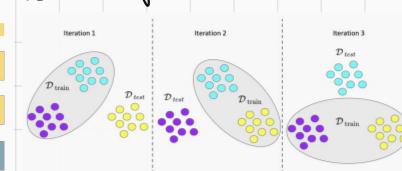
3-fold CV



stratified 3-fold cross-validation



Leave-One-Object-Out



Bias-Variance Analysis in Resampling

If there exists a dedicated Test set that is not used to train \hat{f} , then $\widehat{GE}(\hat{f}, L) = \frac{1}{m} \sum_{(y_i, z_i) \in D_{\text{test}}} L(y_i, \hat{f}(z_i))$ with the m samples being i.i.d. and $E[\widehat{GE}(\hat{f}, L)] = E[L(y, \hat{f}(z))] = GE(\hat{f}, L)$ and $V[\widehat{GE}(\hat{f}, L)] = \frac{1}{m} V[L(y, \hat{f}(z))]$

We can apply the Central Limit Theorem to approx. the distr. of $\widehat{GE}(\hat{f}, L)$ and compute Confidence intervals.

If \hat{f} is trained on D we estimate $GE(I, \lambda, n, p_c)$ instead of $GE(\hat{f}, L)$ using a resampling based estimate $\widehat{GE}(I, \lambda, J, p_c)$.
 $E[\widehat{GE}(I, \lambda, J, p_c)] \approx E[p(y_{J_{\text{test}}}, F_{J_{\text{test}}, I(D_{\text{train}}, \lambda)})] = E\left[\frac{1}{m} \sum_{(y_i, z_i) \in D_{\text{test}}} L(y_i, I(D_{\text{train}})(z_i))\right] = GE(I, \lambda, n_{\text{train}}, p_c)$

! So our estimate for $GE(I, \lambda, n, p_c)$ is in expectation nearly correct, i.e. n_{train} instead of n . B.c. $n_{\text{train}} < n$ our estimate is pessimistically biased - on n it would perform better.
↑ i.e. unbiased

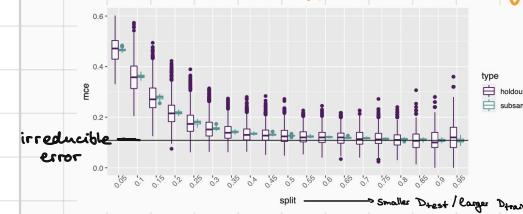
Subsampling/Monte Carlo CV

1. Split the data into two partitions
 2. Use one set for training and the other for testing. hold-out split \rightarrow Results in one test error
 3. Repeat step 1. & 2. many times
 4. Average over all test errors
- ! $\frac{8}{10}$ or $\frac{9}{10}$ of the data for training is common.

Bootstrap

1. Draw a training sets of size n with replacement from the full data set D . The data of D that is not drawn is the test set.
 2. Train and evaluate the model \rightarrow Results in a test error
 3. Repeat 1. & 2. many times
 4. Average over all test errors
- ! A training set contains about 3g unique points (for $n \rightarrow \infty$)

Bias-Variance-Trade-off in Subsampling

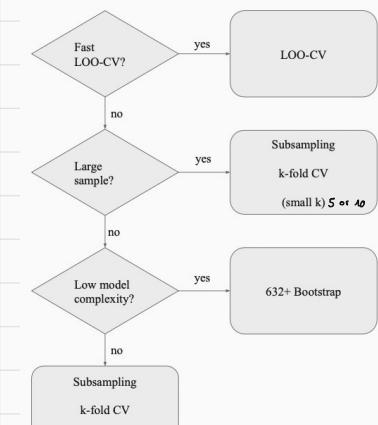


! Bias(Subsampling | split-rate) \neq Bias(Hold-out | split-rate)

Variance (Subsampling | split-rate) $<$ Variance (Hold-out | split-rate)

\Rightarrow "optimal" split-rate is higher in Subsampling compared to one Hold-out-split.

Guidelines



Measures for Regression

Pointwise outer-Loss

Mean squared error (MSE):

$$\rho_{\text{MSE}}(\hat{y}, F) = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2 \in [0, +\infty) \rightarrow L_2 \text{ loss}$$

Similar: sum of squared errors (SSE)

$$\text{root MSE (RMSE)} \quad \rho_{\text{RMSE}} = \sqrt{\rho_{\text{MSE}}}$$

Mean absolute error (MAE):

$$\rho_{\text{MAE}}(\hat{y}, F) = \frac{1}{m} \sum_{i=1}^m |y^{(i)} - \hat{y}^{(i)}| \in [0, +\infty) \rightarrow L_1 \text{ loss}$$

Similar: median absolute error ρ_{MedAE} (even more robust)

Mean absolute percentage error (MAPE):

$$\rho_{\text{MAPE}}(\hat{y}, F) = \frac{1}{m} \sum_{i=1}^m \left| \frac{y^{(i)} - \hat{y}^{(i)}}{y^{(i)}} \right| \in [0, +\infty)$$

↳ Small $|y|$ influence more, but can't handle $y=0$

Set-based outer-Loss

$$R^2: \rho_R(\hat{y}, \bar{F}) = 1 - \frac{\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^m (y^{(i)} - \bar{y})^2} = 1 - \frac{\text{SSE}_{\text{linModel}}}{\text{SSE}_{\text{Constant Model}}} \\ = 1 - \frac{\rho_{\text{MSE}}(\hat{y}, LM(\bar{y}))}{\rho_{\text{MSE}}(\hat{y}, \bar{y})}$$

! Higher $R^2 \Rightarrow$ Better fit.

But R^2 is invariant w.r.t. linear scaling of y . MSE is not.

Generalized R^2 :

$$1 - \frac{\text{Loss}_{\text{ComplexModel}}}{\text{Loss}_{\text{SimpleModel}}} \quad \text{E.g. model vs. constant, linear vs. non-linear, tree vs. forest, ...}$$

! Usually $R^2 \in [0, 1]$, but this is only true if we evaluate on Training Data. On Test Data $R^2 < 0$ is possible \rightarrow Overfitting

Spearman's ρ :

$$\rho_{\text{Spearman}}(\hat{y}, F) = \frac{\text{Cov}(rg(\hat{y}), rg(\hat{y}))}{\sqrt{\text{Var}(rg(\hat{y})) \cdot \text{Var}(rg(\hat{y}))}} \in [-1, 1]$$

↳ Very robust against outliers and invariant under monotone transformations of \hat{y} .

Measures for Classification



Confusion Matrix

For multiclass classification

		True classes			n	
		setosa	versicolor	virginica	error	n
Predicted classes	setosa	50	0	0	0	50
	versicolor	0	46	4	4	50
		virginica	0	4	46	4
		error	0	4	4	8
		n	50	50	50	-
			50	50	50	150

For binary classification:

		True class		
		y positiv	y negativ	
Pred.	\hat{y} pos.	True positiv	False positive	#Pred. pos.
	\hat{y} neg.	False negative	True negative	#Pred. neg.
		#Class positiv	#Class negativ	n

Set-based outer-Loss / ROC Analysis

True condition		
Total population	Condition positive	Condition negative
Predicted condition	True positive, Power	False positive, Type I error
	False negative, Type II error	True negative
Predicted condition negative	True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\text{True positive}}{\text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\text{False positive}}{\text{Condition negative}}$
	False negative rate (FNR), Miss rate = $\frac{\text{False negative}}{\text{Condition positive}}$	True negative rate (TNR), Specificity (SPC), Selectivity, True negative rate = $\frac{\text{True negative}}{\text{Condition negative}}$
Prevalence = $\frac{\text{Condition positive}}{\text{Total population}}$		
Accuracy (ACC) = $\frac{\text{True positive} + \text{True negative}}{\text{Total population}}$		
Positive predictive value (PPV) = $\frac{\text{True positive}}{\text{Predicted condition positive}}$		
False discovery rate (FDR) = $\frac{\text{False positive}}{\text{Predicted condition positive}}$		
Negative predictive value (NPV) = $\frac{\text{True negative}}{\text{Predicted condition negative}}$		
Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$		
Diagnostic odds ratio (DOR) = $\frac{\text{TPR}}{\text{FPR}}$		
F1 score = $\frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$		
Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$		

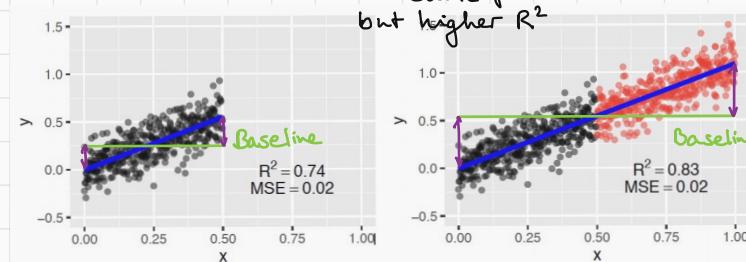
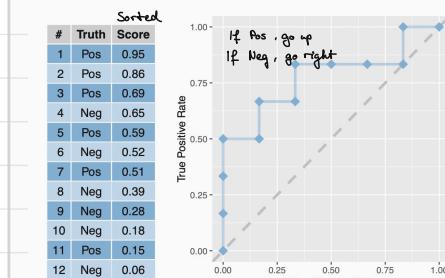
→ Not a good performance measure when label distr. is unbalanced. Accuracy paradox

→ Usually preferable in situations with imbalanced data.

→ Balances TPR and PPV but tends to the smaller value.

Different metrics emphasize different aspects of performance. Choice requires Domain Knowledge.

DRAWING ROC CURVES



Pointwise outer-Loss using class-labels

Misclassification error rate (MCE):

$$\rho_{\text{MCE}} = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \neq \hat{y}^{(i)}] = \frac{\text{False pos.} + \text{False neg.}}{\text{Total population}}$$

Accuracy (ACC):

$$\rho_{\text{ACC}} = \frac{1}{m} \sum_{i=1}^m [y^{(i)} = \hat{y}^{(i)}] = \frac{\text{True pos.} + \text{True neg.}}{\text{Total population}}$$

! No information about how good/lousy prob's are.

Errors on all classes weighted equally, which is often inappropriate

Cost matrix:

$$\text{Costs} = \frac{1}{n} \sum_{i=1}^n C[y^{(i)}, \hat{y}^{(i)}] \\ = \frac{1}{n} \langle \text{Cost Matrix}, \text{Confusion Matrix} \rangle_F$$

Example:

Cost matrix		Confusion matrix	
		True classes	True classes
Predicted classes	solvent	not solvent	solvent
solvent	0	10000	100
not solvent	100	0	10
solvent	75	7	20

$$\text{Costs} = \frac{1}{n} \langle \text{Cost Matrix}, \text{Confusion Matrix} \rangle_F \\ = \frac{1}{100} \text{tr} \left(\begin{pmatrix} 0 & 10000 \\ 100 & 0 \end{pmatrix} \begin{pmatrix} 75 & 7 \\ 7 & 20 \end{pmatrix} \right) = \frac{1}{100} \text{tr} \left(\begin{pmatrix} 7500 & 700 \\ 700 & 200 \end{pmatrix} \right) = 307$$

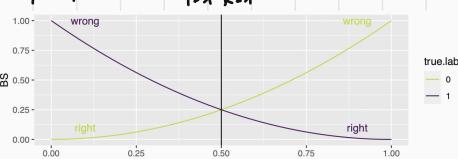
Pointwise outer-Loss using probabilities

Brier Score:

$$\rho_{\text{BS}} = \frac{1}{m} \sum_{i=1}^m (y^{(i)} - \hat{p}^{(i)})^2 \quad (\text{MSE for probabilities}) \\ \hookrightarrow \in [0, 1]$$

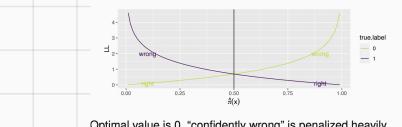
Multi-Class Brier Score:

$$\rho_{\text{BS, MC}} = \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K (y_k^{(i)} - \hat{p}_k^{(i)})^2$$



Log-Loss:

$$\rho_{\text{LL}} = \frac{1}{m} \sum_{i=1}^m (-y^{(i)} \log(\hat{p}^{(i)}) - (1-y^{(i)}) \log(1-\hat{p}^{(i)}))$$



ROC - Curves

ROC-Kurve

Die ROC-Kurve zeigt die Zuverlässigkeit der Vorhersagen für alle möglichen Schwellenwerte c an.

Verbindet die Punkte $(FPR(c), TPR(c)) \forall c \in [x_{(1)}, x_{(n)}]$

Für $c < x_{(1)} \Rightarrow \hat{y}_i = 1 \forall i \Rightarrow (FPR(c), TPR(c)) = (1, 1)$. Für $c > x_{(n)} \Rightarrow \hat{y}_i = 0 \forall i \Rightarrow (FPR(c), TPR(c)) = (0, 0)$

AUC-Maß

Maß zur Bewertung der ROC Kurve. $AUC = \frac{N_c + N_E}{2}$

\hookrightarrow Fläche unter ROC-Kurve.

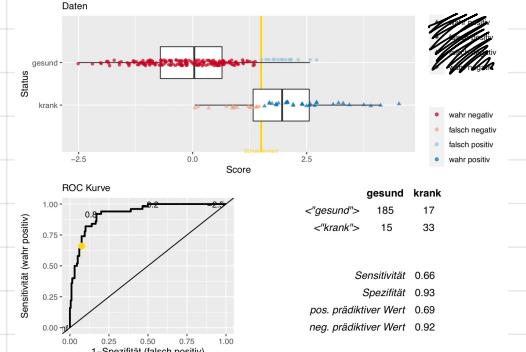
> perfekte Trennschärfe der Y-Gruppen durch c bedeutet $AUC = 1$

> X, y unabhängig $\Rightarrow AUC \approx 0$

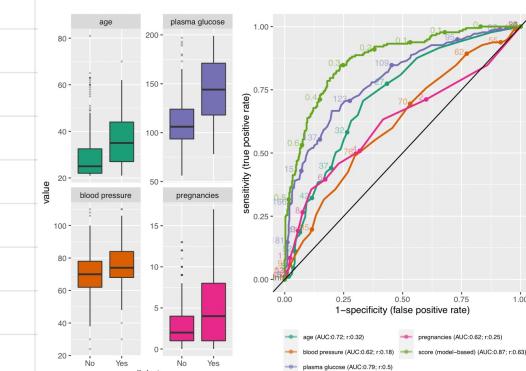
! Behandelt Spezifität & Sensitivität gleichwertig

und beachtet positiv/negativ prädikaten Wert nicht.

Beispiel



Beispiel



Partial AUC

If a certain value for TPR or FPR is not acceptable, then it might be useful to fix TPR or FPR to a required value and optimize the other.

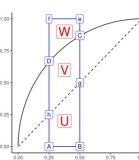
We then don't want to account for the cut-off region in our AUC. Hence, **partial AUC**.

- Normalize to $[0, 1]$:

$$pAUC_{\text{corrected}} = \frac{1}{2} \left(1 + \frac{pAUC - pAUC_{\min}}{pAUC_{\max} - pAUC_{\min}} \right),$$

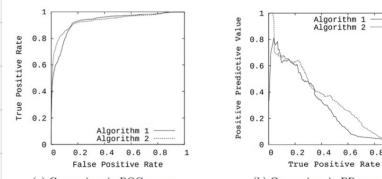
! Skipped Multi-class AUC

- pAUC is $V+U = "A-B-C-D"$
- pAUC_{min} is pAUC of random classifier, so $U = "A-B-g-h"$
- pAUC_{max} is $U+W = "A-B-e-f"$
- Compute percentage of V in $V+W$
- Rescale so random=0.5; optimal=1



Precision-Recall Curves

Create a ROC-like plot, but with TPR and PPV instead of TPR and FPR. might be better for highly imbalanced data ($n_- \gg n_+$)

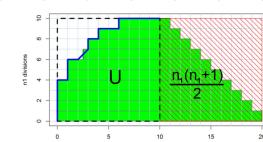
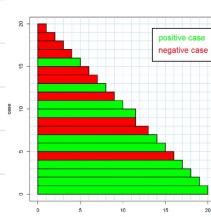


- > Both learners seem to perform well under ROC.
- > But under PR there is visible room for improvement.
- > PR reveals better, that Algo 2 is superior to Algo 1.
- > PR is more sensitive to imbalanced classes.

Curve dominates fully in ROC \Leftrightarrow Curve dominates fully in PR.

AUC & Mann-Whitney-U Test

Next, keep only the green ones, and slide them horizontally to get a nice even staircase on the right edge:



- U : area of the green bars on the left.
- $n_+ \cdot n_-$: area of the dashed rectangle.

\Rightarrow AUC is U normalized to the unit square:

$$AUC = \frac{U}{n_+ \cdot n_-}.$$