Machine Learning I

Supervised learning framework

Souhaib Ben Taieb

University of Mons

Table of contents

Table of contents

Input and output variables

The **input variables**[1] are typically denoted using the symbol X. If we observe p different variables, we write $X = (X_1, X_2, \dots, X_p)$. The inputs belong to an *input space* $\mathcal{X}$.

- Examples: $\mathcal{X} \subseteq \mathbb{R}^p$ or $\mathcal{X} = \{0, 1\}^p$.

The **output variable**[2] is typically denoted using the symbol Y. The output belongs to an *output space* $\mathcal{Y}$.

- Regression: $\mathcal{Y} \subseteq \mathbb{R}$
- Classification (with K categories): $\mathcal{Y} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$
  - Binary classification ($K = 2$): $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{0, 1\}$

[1] also called *predictors*, *independent variables*, *features*, *variables* or just *inputs*.
[2] also called the *response* or *dependent variable*.

Joint distribution

We assume $(X, Y) \sim p_{X,Y}$ where $p_{X,Y}$ is a fixed *unknown* distribution which can be factorized as

$$p_{X,Y}(x, y) = p_X(x)p_{Y|X}(y|x),$$

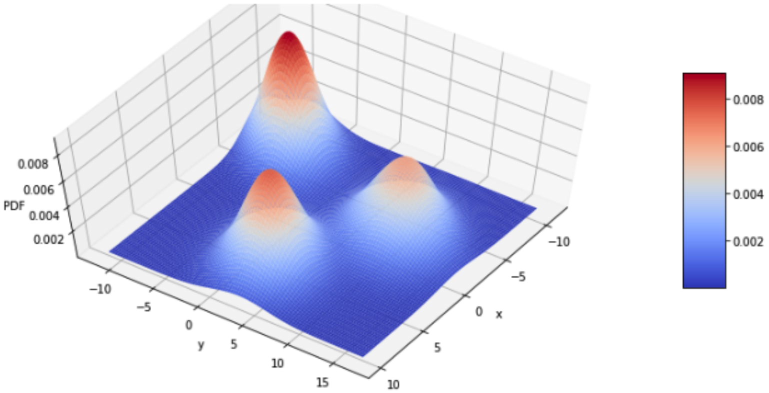
where

- the marginal distribution p_X models uncertainty in the sampling of the inputs.
- the conditional distribution $p_{Y|X}$ describes a stochastic (non-deterministic) relation between inputs and output.

Equivalently, we have

$$X \sim p_X \text{ and } Y|X = x \sim p_{Y|X}(\cdot|x).$$

Joint distribution



Source: `https://tinyurl.com/19bdt531`

Optimal predictions

Define a **loss function** $\mathrm{L} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$. Given a prediction $\hat{y} \in \mathcal{Y}$ and the true (observed) value $y \in \mathcal{Y}$, $L(y, \hat{y})$ measures how far $\hat{y}$ is from y.

Examples include the *squared error loss* $L(y, \hat{y}) = (y - \hat{y})^2$, the *absolute error loss* $L(y, \hat{y}) = |y - \hat{y}|$, and the *zero-one loss* $L(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$, where $\mathbb{1}\{\}$ is the indicator function.

The **optimal prediction function** which minimizes the **expected error** (or **expected risk**) is given by

$$f = \underset{h:\mathcal{X}\to\mathcal{Y}}{\operatorname{argmin}} \ \mathbb{E}_{x,y}[L(y, h(x))].$$

In practice, we cannot compute f since we **do not know** $p_{X,Y}$.

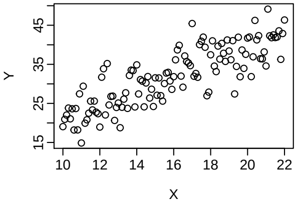
The dataset

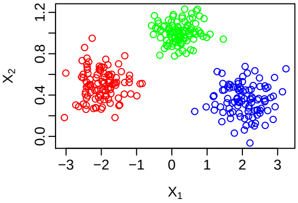
The **dataset**, also called *training set*, is a set of n input-output pairs

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} = \{(x_i, y_i)\}_{i=1}^{n},$$

where the data points (x_i, y_i) are assumed to be i.i.d. realizations of $p_{X,Y}$.

Each pair, also called an *example* or a *data point*, belongs to the *data space* $\mathcal{X} \times \mathcal{Y}$.





- Left figure: $\mathcal{X} \subseteq \mathbb{R}$ ($p = 1$) and $\mathcal{Y} \subseteq \mathbb{R}$
- Right figure: $\mathcal{X} \subseteq \mathbb{R}^2$ ($p = 2$) and $\mathcal{Y} = \{R, G, B\}$

The supervised learning problem

Let $\mathcal{H}$ be a **hypothesis set**, i.e. a set of prediction function (hypotheses) under consideration. An example is the linear hypothesis set $\mathcal{H} = \{h(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p : \beta_0, \beta_1, \ldots, \beta_p \in \mathbb{R}\}$.

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, the goal of supervised learning is to learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ to map new/unseen examples with minimal **prediction error**.

We can compute the **in-sample error** or **training error**[3]

$$E_{\text{in}}(h) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i)),$$

and solve the following optimization problem:

$$g_{\mathcal{D}} = \operatorname*{argmin}_{h \in \mathcal{H}} E_{\text{in}}(h),$$

[3] Also called the **empirical risk**.

The supervised learning problem

Ideally, we would like to select the hypothesis $h \in \mathcal{H}$ which minimizes the **out-of-sample error**

$$E_{\text{out}}(h) = \mathbb{E}_{x,y}[L(y, h(x))], \tag{1}$$

and compute

$$g^* = \underset{h \in \mathcal{H}}{\text{argmin}}\ E_{\text{out}}(h).$$

In summary, there are three different prediction functions:

$$f = \underset{h:\mathcal{X} \to \mathcal{Y}}{\text{argmin}}\ E_{\text{out}}(h),$$

$$g^* = \underset{h \in \mathcal{H}}{\text{argmin}}\ E_{\text{out}}(h),$$

and

$$g_{\mathcal{D}} = \underset{h \in \mathcal{H}}{\text{argmin}}\ E_{\text{in}}(h).$$

Summary

The **dataset** $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is composed of n input-output pairs (x_i, y_i), which are i.i.d. realizations from an **unkown joint distribution** $p_{X,Y}$ where $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$.

The **loss function** $\mathrm{L} : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$ allows us to measure the error we incur in predicting $\hat{y}$ in place of y.

The **hypothesis set** $\mathcal{H}$ is a set of prediction function under consideration. Each hypothesis $h \in \mathcal{H}$ has an **in-sample error** $E_{\text{in}}(h)$, computed on $\mathcal{D}$, and an **out-of-sample error** $E_{\text{out}}(h)$ which depends on $p_{X,Y}$.

Given $\mathcal{H}$ and using $\mathcal{D}$, the **learning algorithm** $\mathcal{A}$ picks the best hypothesis g from $\mathcal{H}$ according to the loss function L.

Together, the hypothesis set and the learning algorithm are referred to as the **learning model**.

Table of contents

Linear models

Let us consider a regression problem where $x \in \mathbb{R}^p$.

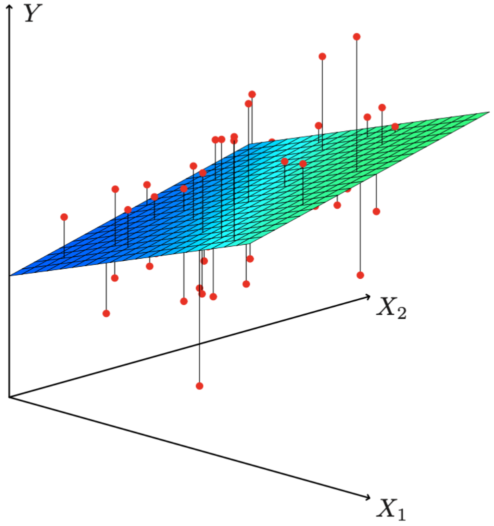
- The **hypothesis set** for linear models is given by

$$\mathcal{H} = \{h(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p : \beta_0, \beta_1, \ldots, \beta_p \in \mathbb{R}\}$$

- One **learning algorithm** is the *(ordinary) least squares* method.

Linear models

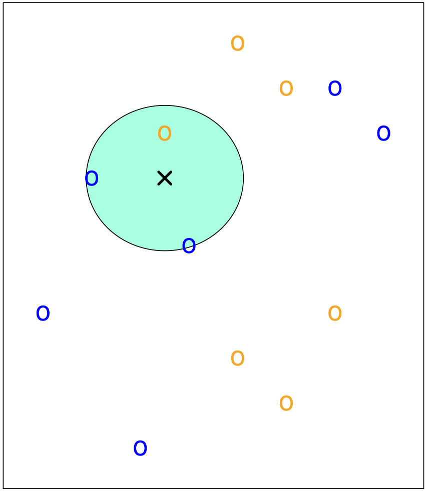
Example with $p = 2$.

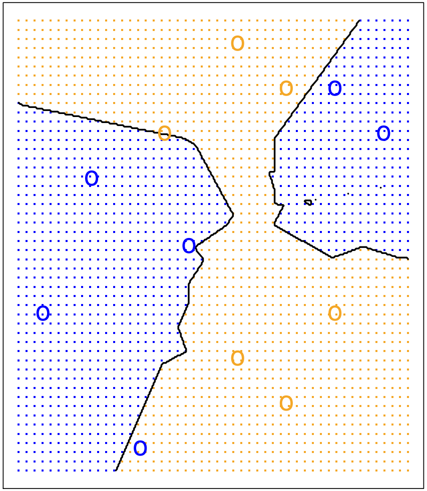


K-Nearest Neighbors (KNN) model

- **Input**: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, number of neighbors $K \leq n$,, and new input x_*
- **Classification** with M categories ($y \in \{C_1, \dots, C_M\}$)
  - Find the K nearest points to x_* in $\mathcal{D}$, denoted $\mathcal{N}_*$.
  - $$\hat{p}(C_m \mid x = x_*) \approx \frac{1}{K} \sum_{i \in \mathcal{N}_*} \mathbb{1}\{y_i = C_m\} \quad (m = 1, 2, \dots, M)$$
  - $$h(x_*) = C_{m^*} \text{ where } m^* = \text{argmax}_m \, \hat{p}(y = C_m \mid x = x_*)$$
- **Regression**
  - Find the K nearest points to x_* in $\mathcal{D}$, denoted $\mathcal{N}_*$.
  - $$h(x_*) = \frac{1}{K} \sum_{i \in \mathcal{N}_*} y_i$$

K-Nearest Neighbours (KNN) model





K-Nearest Neighbours (KNN) is one of the simplest machine learning model for both classification and regression.

Parametric and non-parametric models

- In a **parametric model**, every hypothesis is uniquely defined by a **fixed number of parameters**.
- In a **non-parametric model**, we can not describe a hypothesis with a fixed number of parameters. Usually the number of “parameters” **grows with the size of the dataset**.
- Both parametric and non-parametric models have **hyper-parameters** (structural parameters), while parametric models also have **parameters**
  - For KNN, K is a **hyper-parameter**.
  - For linear models, p is a **hyper-parameter**, and the coefficients β_j are **parameters**.
- We also make a distinction between **linear** and **non-linear** models.

Table of contents

E_{in} for MLE

Recall that, given a set of possible distributions

$$\mathcal{H} = \{p(y;\boldsymbol{\theta}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\},$$

and a dataset $y_1, y_2, \ldots, y_n$, the maximum likelihood estimator is given by

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_{i=1}^{n} \log p(y_i;\boldsymbol{\theta}) \qquad = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} -\log p(y_i;\boldsymbol{\theta})}_{E_{\text{in}}(\boldsymbol{\theta})}. \tag{2}$$

Since each hypothesis h is completely characterized by $\boldsymbol{\theta}$, this is equivalent to

$$g = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\ E_{\text{in}}(h).$$

E_{out} for MLE

By (strong) law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n} -\log p(y_i;\boldsymbol{\theta}) \overset{n\to\infty}{\longrightarrow} \mathbb{E}[-\log p(y;\boldsymbol{\theta})]$$

In other words, we can think of maximum likelihood estimation as trying to minimize

$$E_{\text{out}}(\boldsymbol{\theta}) = \mathbb{E}[-\log p(y;\boldsymbol{\theta})]$$

E_{out} for MLE

$$\begin{aligned}
\operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} E_{\text{out}}(\boldsymbol{\theta}) &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}[-\log p(y;\boldsymbol{\theta})] && (3)\\
&= \operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}\left[\log p(y) - \log p(y;\boldsymbol{\theta})\right] && (4)\\
&= \operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}\left[\log \frac{p(y)}{p(y;\boldsymbol{\theta})}\right] && (5)\\
&= \operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \int \log \frac{p(y)}{p(y;\boldsymbol{\theta})} p(y)\ \mathrm{d}y && (6)\\
&= \operatorname*{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathrm{KL}(p_\theta, p), && (7)
\end{aligned}$$

where $KL(q, p)$ is the KL-divergence between two distributions q and p, which measures the discrepancy between the two distributions. Note that KL-divergence is not a distance measure (not symmetric).

Table of contents

How does E_{out} relates to E_{in}?

$$E_{\text{out}}(h) = E_{\text{in}}(h) + [E_{\text{out}}(h) - E_{\text{in}}(h)].$$

To obtain a small $E_{\text{out}}(h)$, we want

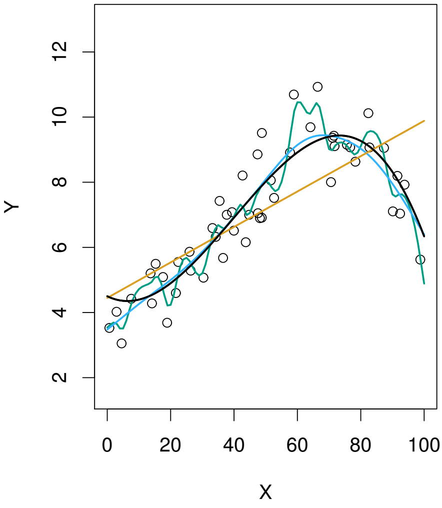
1. small $E_{\text{in}}(h)$
2. small $[E_{\text{out}}(h) - E_{\text{in}}(h)]$

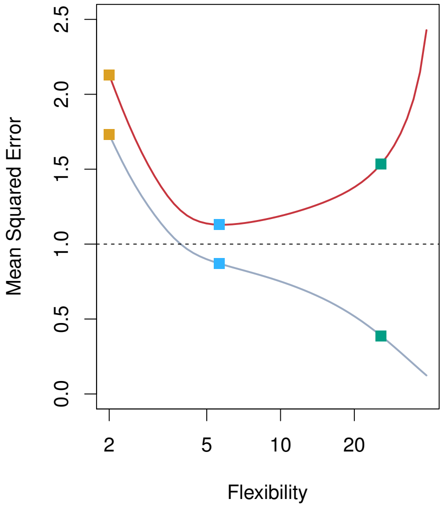
Selecting the best hypothesis by minimizing $E_{\text{in}}(h)$ only can be misleading.

Let $\mathcal{D}' = \{(x_i', y_i')\}_{i=1}^{n'}$ be another sample (**independent** of $\mathcal{D}$) where $(x_i', y_i') \overset{\text{i.i.d.}}{\sim} p_{X,Y}$. We define the **testing/test error** as $E_{\text{test}}(h) = \frac{1}{n'} \sum_{i=1}^{n'} L(y_i', h(x_i'))$.

Let us compare the training error $E_{\text{in}}(h)$ and the test error $E_{\text{test}}(h)$ (a proxy for $E_{\text{out}}(h)$) for different hypotheses h.

Training and test errors in regression

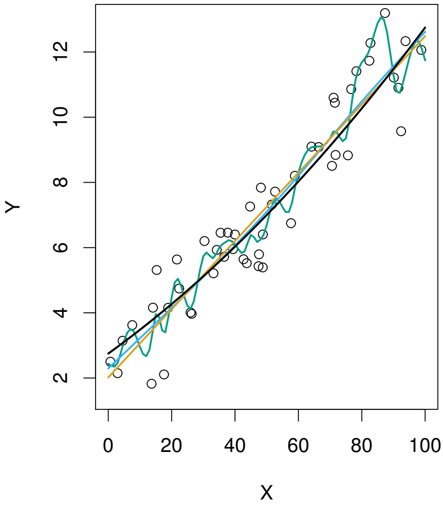


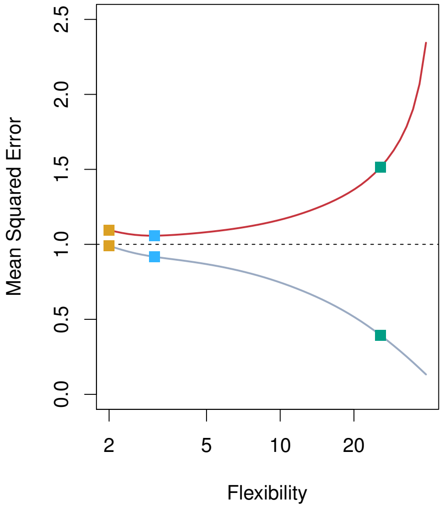


Black: true curve
Orange: linear regression
Blue/green: nonlinear regression

Grey: Training MSE
Red: Test MSE
Dashed: Minimum test MSE

Training and test errors in regression

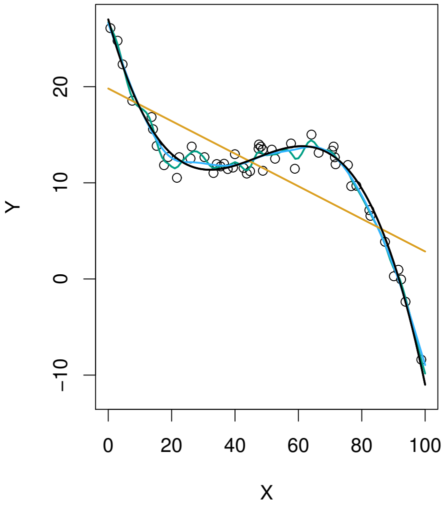


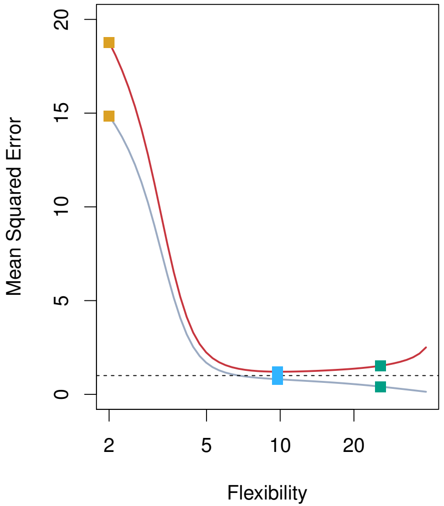


Black: true curve
Orange: linear regression
Blue/green: nonlinear regression

Grey: Training MSE
Red: Test MSE
Dashed: Minimum test MSE

Training and test errors in regression

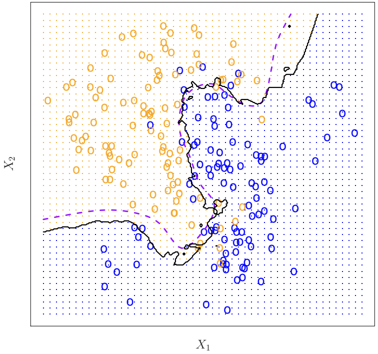


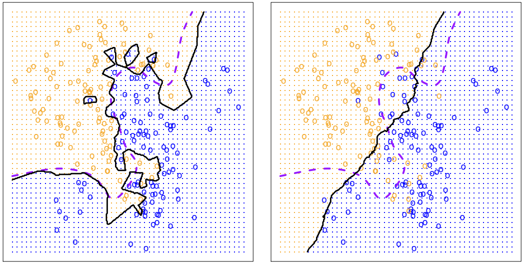


Black: true curve
Orange: linear regression
Blue/green: nonlinear regression

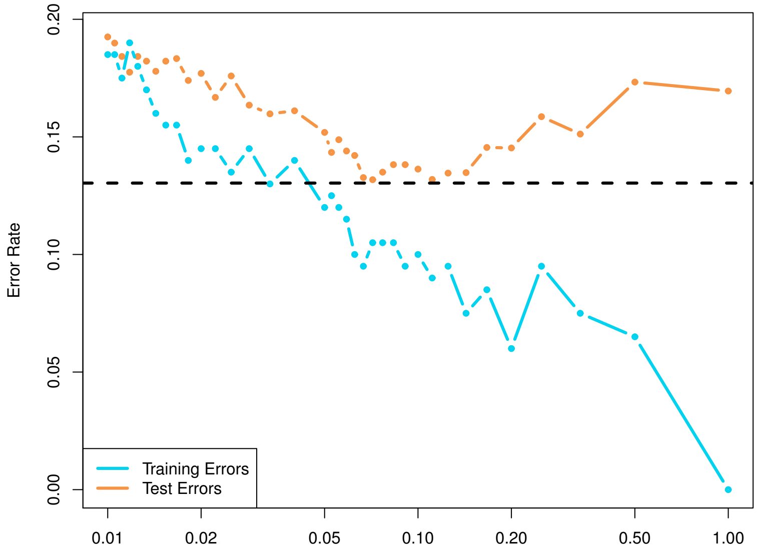
Grey: Training MSE
Red: Test MSE
Dashed: Minimum test MSE

Training and test errors in classification

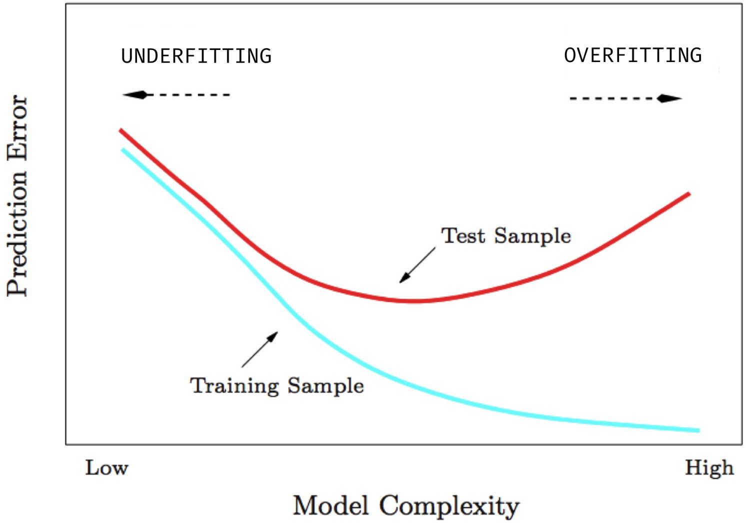




Training and test errors in classification



A fundamental picture



Training and test errors

Consider

$$f = \underset{h:\mathcal{X}\to\mathcal{Y}}{\text{argmin}}\ E_{\text{out}}(h),$$

$$g^* = \underset{h\in\mathcal{H}}{\text{argmin}}\ E_{\text{out}}(h),$$

and

$$g = \underset{h\in\mathcal{H}}{\text{argmin}}\ E_{\text{in}}(h).$$

How different are $E_{\text{out}}(g)$ and $E_{\text{out}}(f)$?

The approximation-generalization tradeoff

The difference between the out-of-sample error of g and f can be decomposed as follows

$$E_{\text{out}}(g) - E_{\text{out}}(f) = \underbrace{[E_{\text{out}}(g^*) - E_{\text{out}}(f)]}_{\text{Approximation error}} + \underbrace{[E_{\text{out}}(g) - E_{\text{out}}(g^*)]}_{\text{Estimation error}}$$

- **Approximation error** is how far the entire hypothesis set is from f. Larger hypothesis sets have lower approximation error.
- **Estimation error** is how good g is with respect to the best in the hypothesis set. Larger hypothesis sets have higher estimation error because it is harder to find a good prediction function based on limited data.

This is called the **approximation-generalization** tradeoff.

Table of contents

The model selection problem

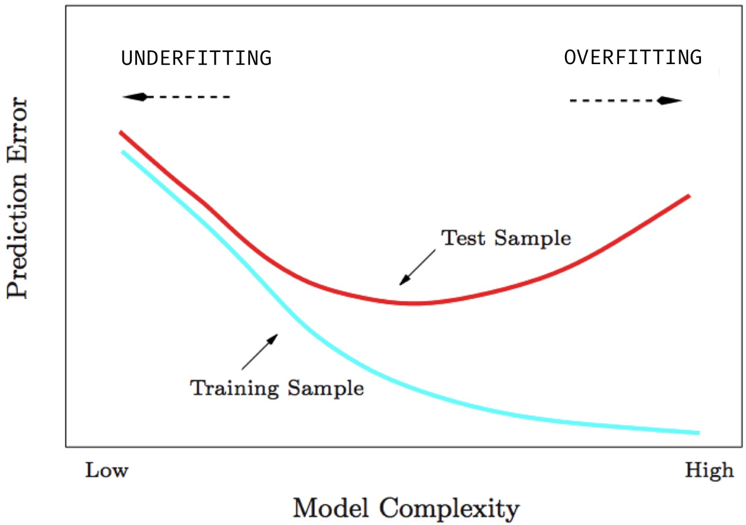
- How many and which input variables should we choose?
- How do we choose the number of neighbours K in KNN?
- ...
- More generally, how do we choose the **hyper-parameters** of a machine learning model?

$\rightarrow$ We need a way of **assessing** and **selecting** the best model among multiple competing models.

The model selection/structural identification procedure

1. **Model generation** (hypothesis set)
Generate a set of candidate model structures among which the best one is to be selected. If applicable, estimate the parameters of each candidate model, i.e. fit the model by minimizing the in-sample (training) error (*parametric identification*).
2. **Model assessment/validation**
Evaluate/validate the model's performance by computing a validation error, i.e. an estimate of the out-of-sample error.
3. **Model selection**
Select the final model structure in the set that has been proposed by **model generation** and assessed by **model validation**. We typically select the model structure that minimizes the validation error.

Why not use the in-sample (training) error for model selection?



How to estimate the out-of-sample error?

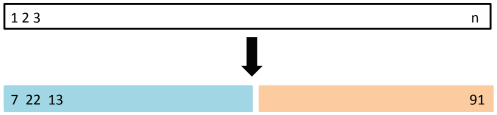
$$E_{\text{out}}(h) = E_{\text{in}}(h) + \underbrace{[E_{\text{out}}(h) - E_{\text{in}}(h)]}_{\textbf{overfit penalty}}, \quad h \in \mathcal{H}.$$

1. **Directly estimate it** using resampling methods, i.e. by resampling the data set. Examples include **the validation set**, **cross-validation** and **bootstrap**.
2. **Estimate the overfiit penalty** and **add it to the in-sample (training) error**.

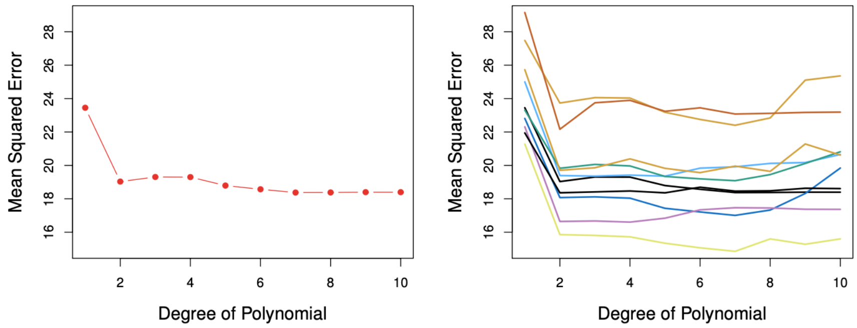
Table of contents

Validation-set approach

- We randomly divide the dataset into two parts: a **training set** and a **validation set**.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set
- The resulting **validation-set error** provides an estimate of the **out-of-sample error**.
- This estimate of the out-of-sample error can be highly variable, depending on precisely which observations are included in the training set and the validation set.

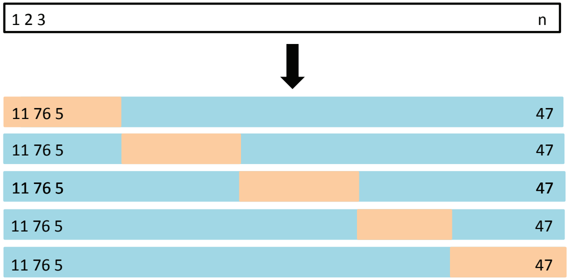


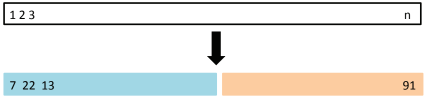
Validation-set approach

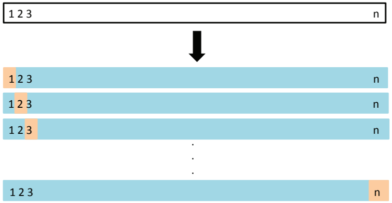


Left panel shows single split; right panel shows multiple splits

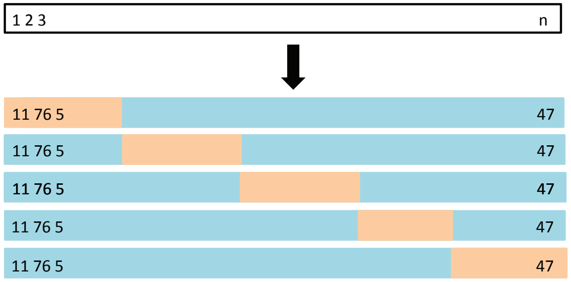
K-fold Cross-validation

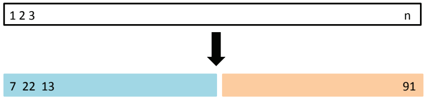


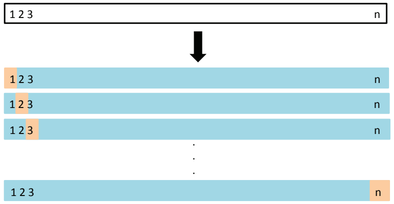




K-fold Cross-validation







K-fold Cross-validation

- Divide the data set into K different parts.
- Remove one part, fit the model on the remaining $K-1$ parts, and compute the prediction error on the omitted part.
- Repeat K times taking out a different part each time
- By averaging the K prediction errors we obtain an estimate of the out-of-sample error, i.e. the prediction error for new observations (not used in training).
- Setting $K=n$ yields n-fold or **leave-one out cross-validation** (LOOCV).

K-fold Cross-validation

Let the K parts be $A_1, A_2, \ldots, A_K$, where A_k denotes the indices of the observations in part k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.
Compute

$$\text{CV} = \sum_{k=1}^{K} \frac{n_k}{n} \text{Err}_k$$

where, for example,

$$\text{Err}_k = \sum_{i \in A_k} \frac{1}{n_k} (y_i - \hat{y}_i^{(-k)})^2 \quad \text{(squared error loss)},$$

or

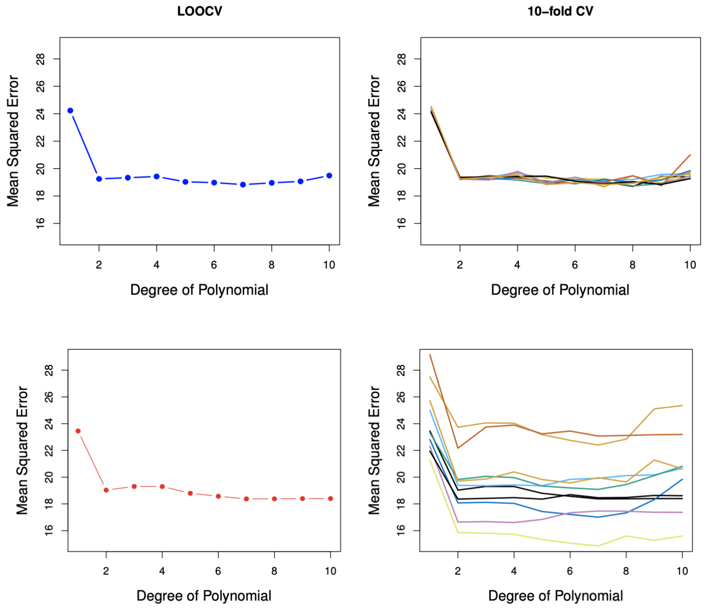
$$\text{Err}_k = \sum_{i \in A_k} \frac{1}{n_k} \mathbb{1}\{y_i \neq \hat{y}_i^{(-k)}\} \quad \text{(zero-one loss)},$$

and $\hat{y}_i^{(-k)}$ is the prediction for observation i, obtained from the model fit with data where part k is removed.

K-fold Cross-validation - Bias and variance tradeoff

- Each training set is only $(K-1)/K$ as big as the original data set. So the estimates of prediction error will be biased upwards.
- Bias minimized when $K = n$ (LOOCV).
- But variance increases with K since the estimates from each fold are more correlated (as there are more overlapping observations in each part) and hence their average can have higher variance.
- Empiricial observation: $K = 5$ or $K = 10$ provide a good compromise for this bias-variance tradeoff.

10-fold Cross-validation



[illegible]

K-fold Cross-validation

For K-fold cross-validation, it's very helpful to assign a quantitative notion of variability to the cross-validation error estimate. Assuming $n_k = n/K$, we can write

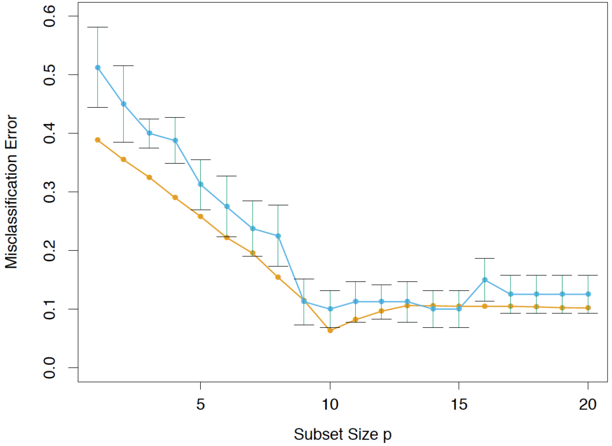
$$\mathrm{Var}(CV) = \mathrm{Var}\left(\sum_{k=1}^{K} \frac{1}{K}\mathrm{Err}_k\right) \approx \frac{1}{K^2}\sum_{k=1}^{K}\mathrm{Var}(\mathrm{Err}_1) = \frac{1}{K}\mathrm{Var}(\mathrm{Err}_1)$$

- This is an approximation since $\mathrm{Err}_1, \ldots, \mathrm{Err}_K$ are not i.i.d.
- This approximation is valid for small K (e.g., $K = 5$ or 10) but not really for big K (e.g., $K = n$), because then the quantities $\mathrm{Err}_1, \ldots, \mathrm{Err}_K$ are highly correlated.
- We can compute the standard deviation or standard error of the cross-validation error estimate as

$$\frac{1}{\sqrt{K}}\mathrm{sd}\{\mathrm{Err}_1, \mathrm{Err}_2, \ldots, \mathrm{Err}_K\},$$

where sd denotes the *empirical standard deviation*.

The one standard error rule



Choose the simplest model whose CV error is no more than one standard error above the model with the lowest CV error

Cross-validation: right and wrong

Consider a simple regression procedure applied to a dataset with 500 predictors and 50 samples:

1. Find the 5 predictors having the largest correlation with the response
2. Apply linear regression using only these 5 predictors

How to use cross-validation to estimate the out-of-sample error of this procedure?

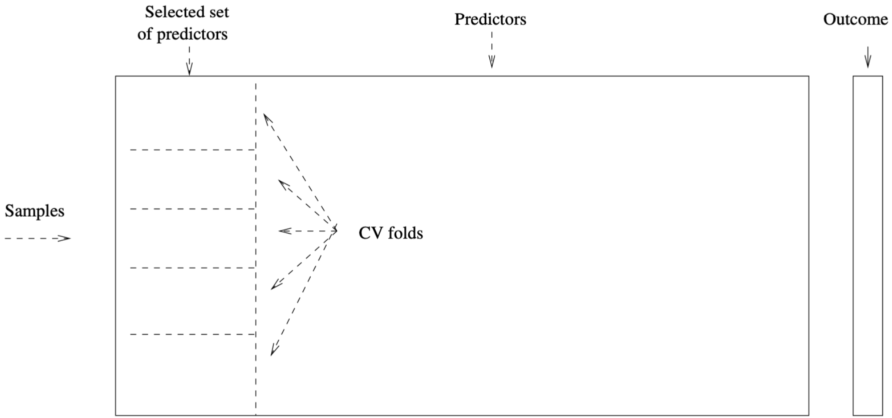
Cross-validation: right and wrong

Consider a simple regression procedure applied to a dataset with 500 predictors and 50 samples:

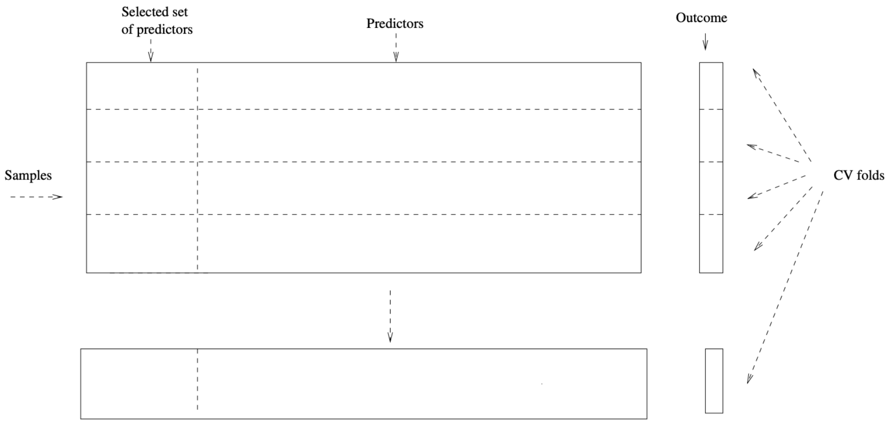
1. Find the 5 predictors having the largest correlation with the response
2. Apply linear regression using only these 5 predictors

How to use cross-validation to estimate the out-of-sample error of this procedure?

Wrong way


Selected set of predictors
Predictors
Outcome
Samples
CV folds

Right way


Selected set of predictors
Predictors
Outcome
Samples
CV folds

Cross-validation: right and wrong

Apply cross-validation to steps 1 **and** 2 (not just step 2)
$\rightarrow$ Every aspect of the procedure that involves using the data — variable selection, scaling, etc — must be cross-validated.