# Introduction to machine learning

## Supervised discriminative learning

Simon Leglaive

CentraleSupélec

This presentation is adapted from INFO8010 - Deep Learning course by Gilles Louppe at ULiège.

# In the last episode...

We want to infer some latent information from observations.

- Data: Get **unlabeled** data $\mathcal{D} = \left\{ \mathbf{x}_i \overset{i.i.d}{\sim} p^\star(\mathbf{x}) \right\}_{i=1}^N$.
- Modeling: Define a model that relates the latent variable of interest to the observations $p(\mathbf{x}, z; \theta) = p(\mathbf{x} \mid z; \theta) p(z; \theta)$.
- Inference: Compute the posterior distribution $p(z \mid \mathbf{x}; \theta)$, which can then be used in many different ways.
- Learning: Estimate the unknown model parameters $\theta$ by maximizing the log-marginal likelihood $\ln p(\mathbf{x}; \theta)$ averaged over the dataset.

This corresponds to a form of unsupervised learning, using a generative modeling approach.

# Today

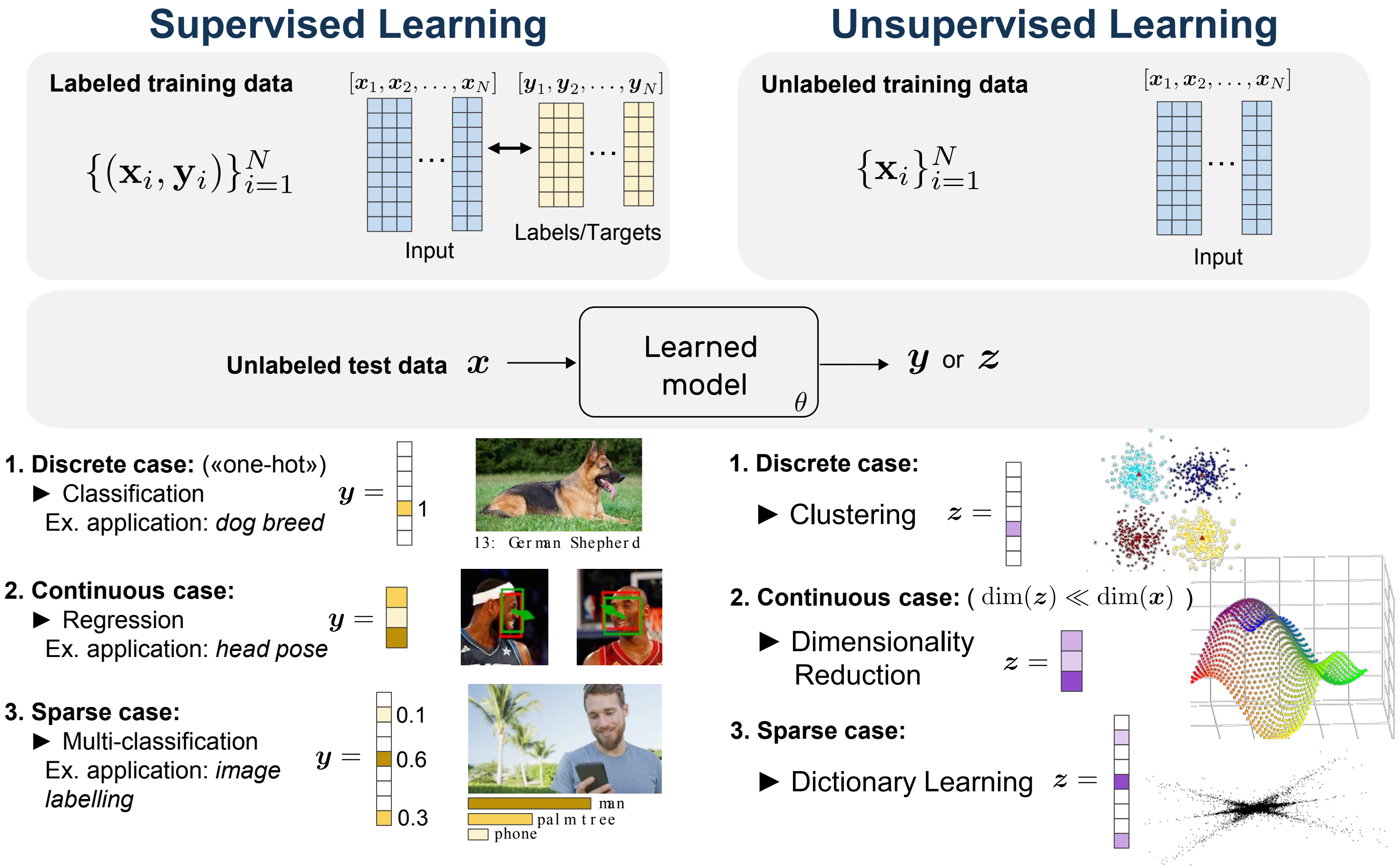
We will focus on supervised learning with discriminative models.

The key concepts you should be familiar with at the end of this course are the following:

- Supervised learning
- Generative vs. discriminative model
- Empirical risk minimization
- Multinomial logistic regression (lab session)

# Supervised learning

The term supervised means that, at training time, the input data samples $\mathbf{x}_i \in \mathcal{X}$ are labeled with the information we will try to infer at test time. The labels are represented by $y_i \in \mathcal{Y}$.



## Supervised Learning

**Labeled training data**

$\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$

$[\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N]$ $\longleftrightarrow$ $[\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N]$

Input

Labels/Targets

## Unsupervised Learning

**Unlabeled training data**

$\{\mathbf{x}_i\}_{i=1}^N$

$[\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N]$

Input

**Unlabeled test data** $\boldsymbol{x}$ → Learned model $\theta$ → $\boldsymbol{y}$ or $\boldsymbol{z}$

**1. Discrete case:** («one-hot»)
▶ Classification
Ex. application: *dog breed*

$\boldsymbol{y} =$ 1

**2. Continuous case:**
▶ Regression
Ex. application: *head pose*

$\boldsymbol{y} =$

**3. Sparse case:**
▶ Multi-classification
Ex. application: *image labelling*

$\boldsymbol{y} =$ 0.1 0.6 0.3

**1. Discrete case:**
▶ Clustering $\boldsymbol{z} =$

**2. Continuous case:** ( $\dim(\boldsymbol{z}) \ll \dim(\boldsymbol{x})$ )
▶ Dimensionality Reduction $\boldsymbol{z} =$

**3. Sparse case:**
▶ Dictionary Learning $\boldsymbol{z} =$

Image credits: Antoine Deleforge, Inria, course given at Télécom Physique Strasbourg.

# Training time

At training time, our observations consist of (input, label) pairs, which are assumed to be i.i.d according to some distribution $p^\star(\mathbf{x}, y)$:

$$\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \overset{i.i.d}{\sim} p^\star(\mathbf{x}, y) \right\}_{i=1}^{N}.$$

For instance, $\mathbf{x}_i$ is a $D$-dimensional input feature vector and $y_i$ is a scalar label (e.g., a category or a real value).

# Test time

At test time, we only observe a new input data sample $\mathbf{x}$, from which we want to infer $y$.

This means computing the posterior distribution $p(y \mid \mathbf{x}; \theta)$, which depends on a set of parameters $\theta$ that have been estimated during the learning stage, at training time.

In supervised learning, we are often only interested in a point estimate $\hat{y}$, for instance:

- classification task:

$$\hat{y} = \arg\max_{k \in \{1,...,C\}} p(y = k \mid \mathbf{x}; \theta) \in \{1, ..., C\};$$

- regression task:

$$\hat{y} = \mathbb{E}_{p(y|\mathbf{x};\theta)}\,[y] \in \mathbb{R}.$$

# Supervised learning with generative modeling

We have access to a **labeled** dataset $\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} p^\star(\mathbf{x}, y) \right\}_{i=1}^N$.

- Generative modeling - we define the joint distribution to explain how the input data $\mathbf{x}$ are generated from the label $y$:

$$p(\mathbf{x}, y; \theta) = p(\mathbf{x} \mid y; \theta) p(y; \theta).$$

- Inference - we "invert" this generative model using Bayes theorem:

$$p(y \mid \mathbf{x}; \theta) = \frac{p(\mathbf{x} \mid y; \theta) p(y; \theta)}{p(\mathbf{x}; \theta)}.$$

- Learning - we estimate the unknown model parameters $\theta$ by maximizing the log-likelihood $\ln p(\mathbf{x}, y; \theta)$ averaged over the dataset $\mathcal{D}$:

$$\max_\theta \mathbb{E}_{p^\star(\mathbf{x},y)} [\ln p(\mathbf{x}, y; \theta)].$$

No need to marginalize over $y$ as it is observed in a supervised setting!

# Supervised learning with discriminative modeling

We have access to a **labeled** dataset $\mathcal{D} = \left\{(\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} p^\star(\mathbf{x}, y)\right\}_{i=1}^{N}$.

- Discriminative modeling - we define the joint distribution to directly explain how the label $y$ is obtained from the input data $\mathbf{x}$:

  $$p(\mathbf{x}, y; \theta) = p(y \mid \mathbf{x}; \theta)p(\mathbf{x}),$$

  where $p(\mathbf{x})$ is non-informative (e.g., uniform after normalization) and does not depend on $\theta$.

- Inference - super easy, just evaluate the model!

- Learning - we estimate the unknown model parameters $\theta$ by maximizing the log-likelihood $\ln p(\mathbf{x}, y; \theta) = \ln p(y \mid \mathbf{x}; \theta) + cst$ averaged over the dataset $\mathcal{D}$:

  $$\max_{\theta} \mathbb{E}_{p^\star(\mathbf{x},y)} \left[\ln p(\mathbf{x}, y; \theta)\right] \quad \Leftrightarrow \quad \min_{\theta} \mathbb{E}_{p^\star(\mathbf{x},y)} \left[- \ln p(y \mid \mathbf{x}; \theta)\right].$$

  In supervised discriminative learning, $-\ln p(y \mid \mathbf{x}; \theta)$ is often called the negative log-likelihood (NLL) function, which can be confusing (e.g., torch.nn.NLLLoss).

# 3 fundamental examples

# Binary classification example

- Training data: $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Y} = \{0, 1\}$.
- Model: We assume there exists a function $f(\cdot; \theta) : \mathbb{R}^D \mapsto [0, 1]$ such that

  $p(y = 1|\mathbf{x}; \theta) = f(\mathbf{x}; \theta)$ and $p(y = 0|\mathbf{x}; \theta) = 1 - f(\mathbf{x}; \theta)$.

  The model can be compactly rewritten using the following probability mass function (pmf) for all $y \in \mathcal{Y}$:

$$p(y \mid \mathbf{x}; \theta) = \Big(f(\mathbf{x}; \theta)\Big)^y \Big(1 - f(\mathbf{x}; \theta)\Big)^{(1-y)}.$$

- The NLL function is called the binary cross-entropy:

$$-\ln p(y \mid \mathbf{x}; \theta) = -y \ln \Big(f(\mathbf{x}; \theta)\Big) - (1 - y) \ln \Big(1 - f(\mathbf{x}; \theta)\Big).$$

# $C$-class classification example

- Training data: $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Y} = \{1, ..., C\}$.
- Model: We assume there exists a function $f(\cdot; \theta) : \mathbb{R}^D \mapsto [0, 1]^C$ such that

  $$p(y = k \mid \mathbf{x}; \theta) = f_k(\mathbf{x}; \theta) \text{ for all } k \in \{1, ..., C\} \text{ and } \sum_{k=1}^{C} f_k(\mathbf{x}; \theta) = 1.$$

  It can be compactly rewritten using the following pmf for all $y \in \mathcal{Y}$:

  $$p(y \mid \mathbf{x}; \theta) = \prod_{k=1}^{C} p(y = k \mid \mathbf{x}; \theta)^{\mathbf{1}_{y=k}} = \prod_{k=1}^{C} f_k(\mathbf{x}; \theta)^{\mathbf{1}_{y=k}}.$$

- The NLL function is called the cross-entropy:

  $$-\ln p(y \mid \mathbf{x}; \theta) = -\sum_{k=1}^{C} \mathbf{1}_{y=k} \ln \Big( f_c(\mathbf{x}; \theta) \Big).$$

# Logistic regression vs. naive Bayes classifiers

- When $f(\mathbf{x}; \theta)$ is an affine function of $\mathbf{x}$, this model is called (multinomial) logistic regression.

  It is one of the simplest discriminative models for supervised classification.

- The most popular generative model for supervised classification is called the naive Bayes classifier.

- For a comparison between the generative and discriminative approches to supervised classification, see (Ng and Jordan, 2002).

Ng, Andrew Y.; Jordan, Michael I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. Advances in Neural Information Processing Systems (NIPS).

# Regression example

- Training data: $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^P$ and $\mathcal{Y} = \mathbb{R}^Q$.
- Model: We assume there exists a function $f(\cdot; \theta) : \mathbb{R}^P \mapsto \mathbb{R}^Q$ such that

$$p(\mathbf{y} \mid \mathbf{x}; \theta) = \mathcal{N}\Big(\mathbf{y}; f(\mathbf{x}; \theta), \mathbf{I}\Big) = (2\pi)^{-Q/2} \exp\Big( -\tfrac{1}{2} \parallel \mathbf{y} - f(\mathbf{x}; \theta) \parallel_2^2 \Big).$$

- The NLL gives the squared error:

$$-\ln p(\mathbf{y} \mid \mathbf{x}; \theta) = \frac{1}{2} \parallel \mathbf{y} - f(\mathbf{x}; \theta) \parallel_2^2 + cst.$$

# Wrap-up

- Data: Get the labeled training dataset $\mathcal{D} = \left\{(\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} p^\star(\mathbf{x}, y)\right\}_{i=1}^{N}$.
- Modeling/inference: Define $p(y \mid \mathbf{x}; \theta)$.
- Learning: Estimate $\theta$ by minimizing $-\ln p(y \mid \mathbf{x}; \theta)$ averaged over $\mathcal{D}$.

Having to define a model as the expression of a probability mass/density function $p(y \mid \mathbf{x}; \theta)$ can be restrictive, and is actually not necessary.

A more general approach to supervised discriminative learning is based on the principle of empirical risk minimization.

# Empirical risk minimization

# Empirical risk minimization

Consider a function $f : \mathcal{X} \to \mathcal{Y}$ produced by some learning algorithm. The predictions of this function can be evaluated through a loss

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R},$$

such that $\ell(y, f(\mathbf{x})) \geq 0$ measures how close the prediction $f(\mathbf{x})$ from $y$ is.

## Examples of loss functions

Classification: $\ell(y, f(\mathbf{x})) = \mathbf{1}_{y \neq f(\mathbf{x})}$

Regression: $\ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

Let $\mathcal{F}$ denote the hypothesis space, i.e. the set of all functions $f$ than can be produced by the chosen learning algorithm.

We are looking for a function $f \in \mathcal{F}$ with a small expected risk (or generalization error)

$$R(f) = \mathbb{E}_{p^\star(\mathbf{x},y)}\left[\ell(y, f(\mathbf{x}))\right] = \mathbb{E}_{p^\star(\mathbf{x})}\left[\mathbb{E}_{p^\star(y|\mathbf{x})}\left[\ell(y, f(\mathbf{x}))\right]\right].$$

This means that for a given data generating distribution $p^\star(\mathbf{x}, y)$ and for a given hypothesis space $\mathcal{F}$, the optimal model is

$$f^\star = \arg\min_{f \in \mathcal{F}} R(f).$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

Unfortunately, since $p^\star(\mathbf{x}, y)$ is unknown, the expected risk cannot be evaluated and the optimal model cannot be determined.

However, if we have i.i.d. training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, we can compute an estimate, the empirical risk (or training error)

$$\hat{R}(f, \mathcal{D}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \ell(y_i, f(\mathbf{x}_i)).$$

This estimate is unbiased and can be used for finding a good enough approximation of $f^\star$. This results into the empirical risk minimization principle:

$$f^\star_{\mathcal{D}} = \arg\min_{f \in \mathcal{F}} \hat{R}(f, \mathcal{D})$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

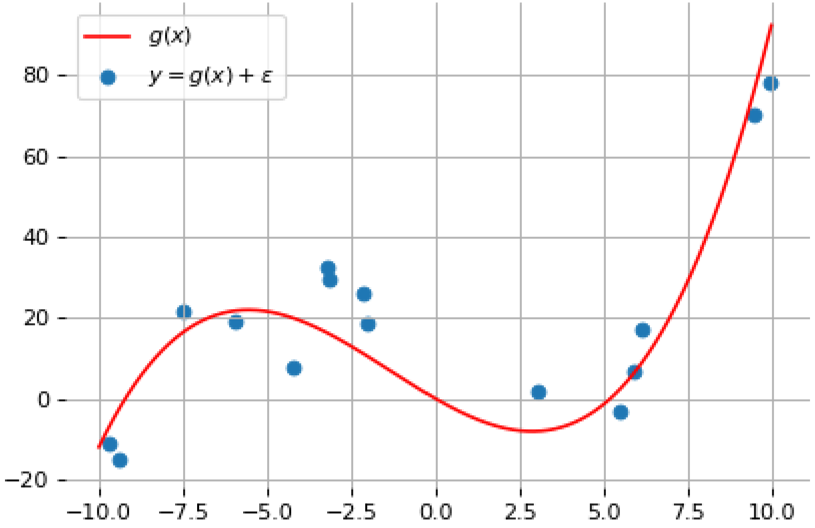
Most supervised machine learning algorithms, including neural networks, implement empirical risk minimization.

Under regularity assumptions, empirical risk minimizers converge:

$$\lim_{N \to \infty} f^\star_{\mathcal{D}} = f^\star$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Regression example

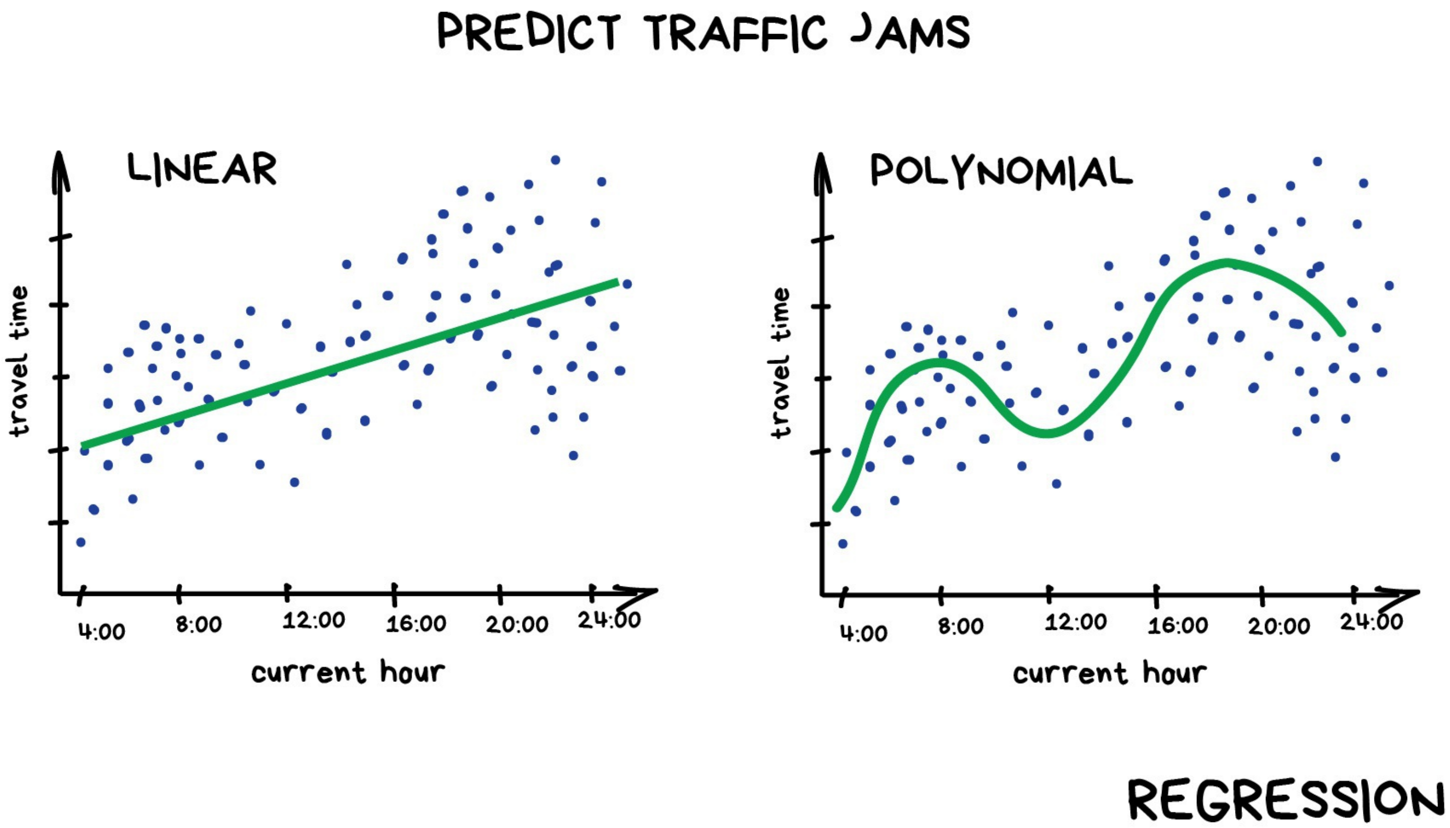


Consider the joint probability distribution $p^\star(x, y)$ induced by the data generating process

$$(x, y) \sim p^\star(x, y) \Leftrightarrow x \sim \mathcal{U}([-10; 10]), \epsilon \sim \mathcal{N}(0, \sigma^2), y = g(x) + \epsilon$$

where $x \in \mathbb{R}$, $y \in \mathbb{R}$ and $g$ is an unknown polynomial of degree 3.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Regression is used to study the relationship between two continuous variables.

Of course, it can be extended to higher dimensions.

Image credit: https://noeliagorod.com/2019/05/21/machine-learning-for-everyone-in-simple-words-with-real-world-examples-yes-again/

# Step 1: Defining the model

Our goal is to find a function $f$ that makes good predictions on average over $p^\star(x, y)$.

Consider the hypothesis space $f \in \mathcal{F}$ of polynomials of degree 3 defined through their parameters $\mathbf{w} \in \mathbb{R}^4$ such that

$$\hat{y} \triangleq f(x; \mathbf{w}) = \sum_{d=0}^{3} w_d x^d$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Step 2: Defining the loss function

For this regression problem, we use the squared error loss

$$\ell(y, f(x; \mathbf{w})) = (y - f(x; \mathbf{w}))^2$$

to measure how wrong the predictions are.

Therefore, our goal is to find the best value $\mathbf{w}^\star$ such

$$\begin{aligned} \mathbf{w}^\star &= \arg\min_{\mathbf{w}} R(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \mathbb{E}_{p^\star(x,y)} \left[ (y - f(x; \mathbf{w}))^2 \right] \end{aligned}$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Step 3: Training

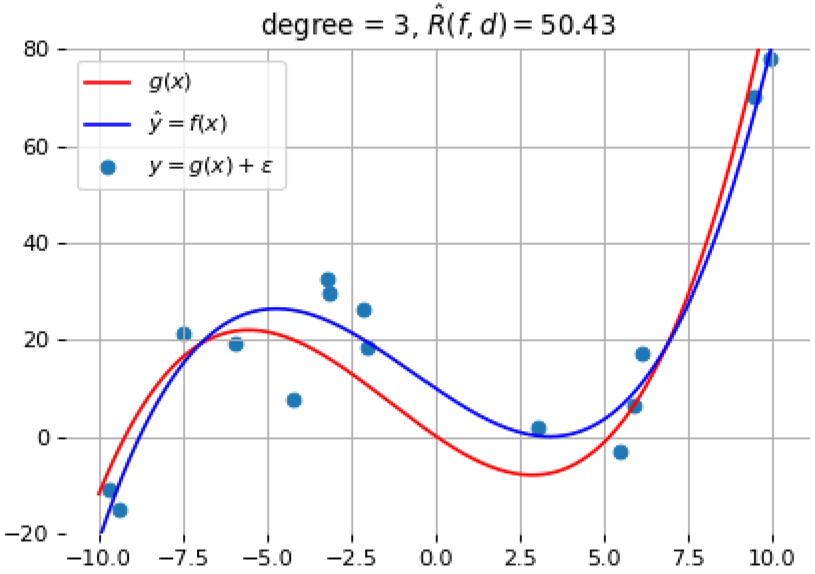
Given a large enough training set $\mathcal{D} = \{(x_i, y_i) | i = 1, \ldots, N\}$, the empirical risk minimization principle tells us that a good estimate $\mathbf{w}^\star_\mathcal{D}$ of $\mathbf{w}^\star$ can be found by minimizing the empirical risk:

$$
\begin{aligned}
\mathbf{w}^\star_\mathcal{D} &= \arg\min_\mathbf{w} \hat{R}(\mathbf{w}, \mathcal{D}) \\
&= \arg\min_\mathbf{w} \frac{1}{N} \sum_{(x_i, y_i) \in \mathcal{D}} (y_i - f(x_i; \mathbf{w}))^2 \\
&= \arg\min_\mathbf{w} \frac{1}{N} \sum_{(x_i, y_i) \in \mathcal{D}} \Big(y_i - \sum_{d=0}^{3} w_d x_i^d \Big)^2 \\
&= \arg\min_\mathbf{w} \frac{1}{N} \left\lVert \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}}_{\mathbf{y}} - \underbrace{\begin{pmatrix} x_1^0 \ldots x_1^3 \\ x_2^0 \ldots x_2^3 \\ \ldots \\ x_N^0 \ldots x_N^3 \end{pmatrix}}_{\mathbf{X}} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} \right\rVert^2
\end{aligned}
$$

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

This is ordinary least squares regression, for which the solution is known analytically:

$$\mathbf{w}^\star_{\mathcal{D}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



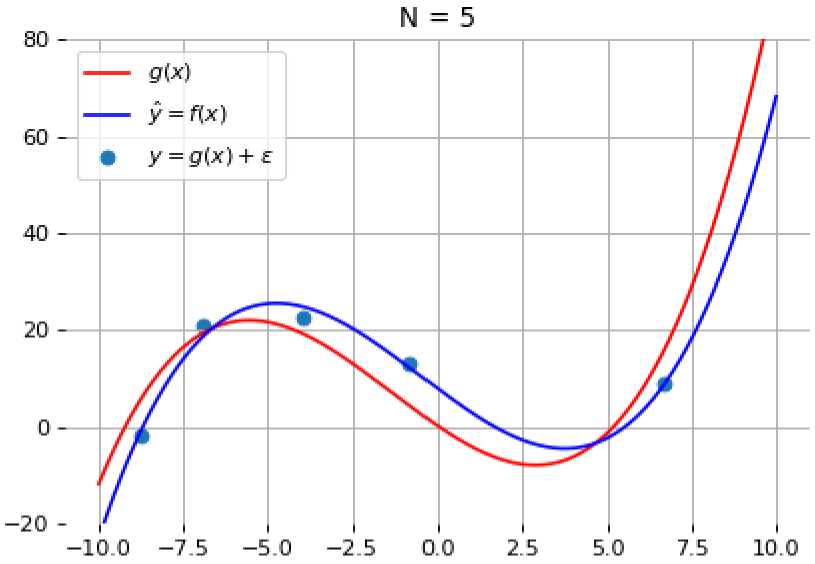
In many situations, the problem is more difficult and we cannot find the solution analytically. We resort to **iterative optimization algorithms**, such as (variants of) gradient descent.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

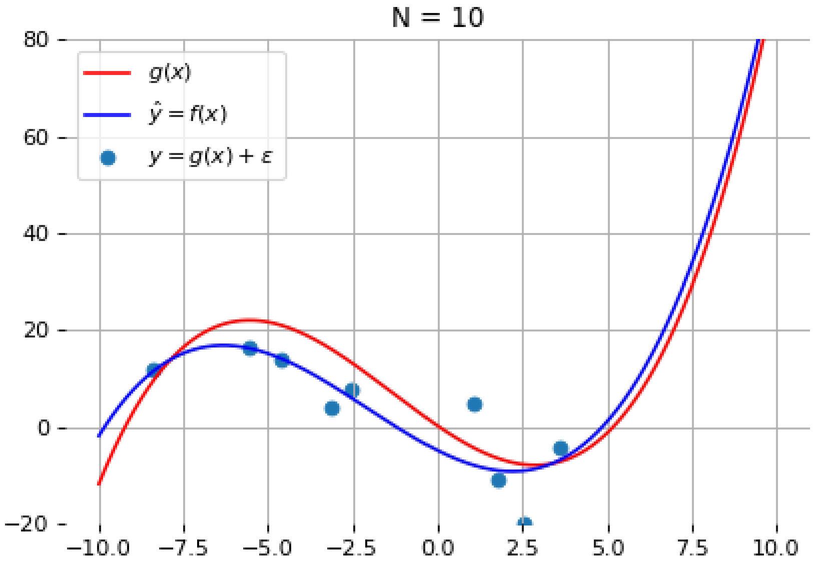
The expected risk minimizer $f(x;\mathbf{w}^\star)$ within our hypothesis space $\mathcal{F}$ (polynomials of degree 3) is $g(x)$ itself (i.e. the polynomial of degree 3 with the true parameters).

Therefore, on this toy problem, we can verify that $f(x;\mathbf{w}^\star_\mathcal{D}) \to f(x;\mathbf{w}^\star) = g(x)$ as $N \to \infty$.

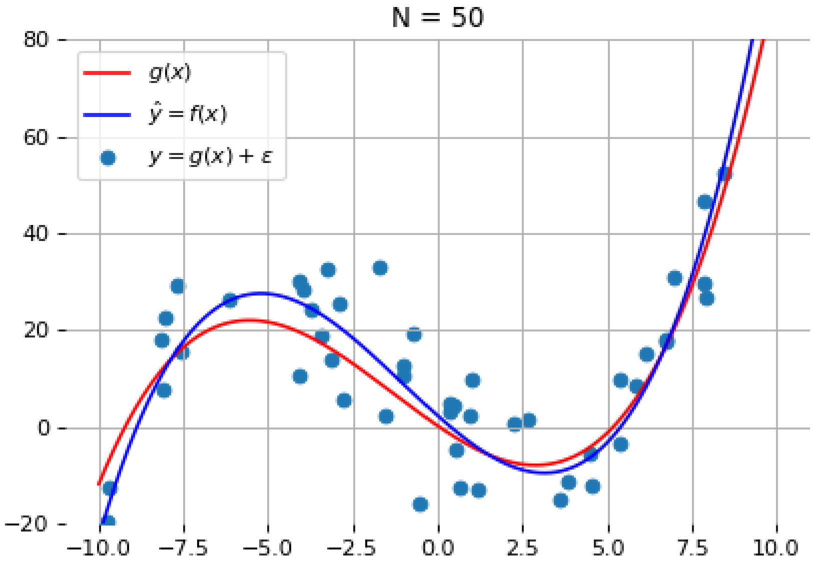
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



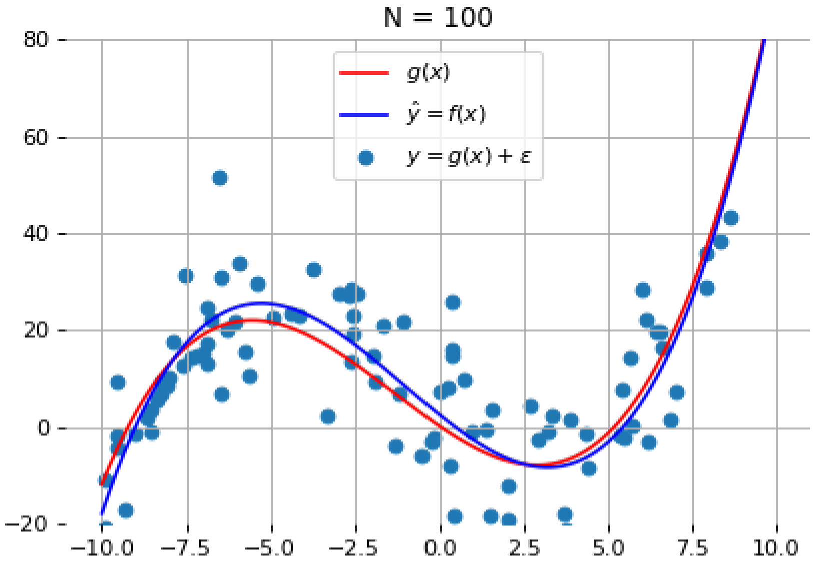
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



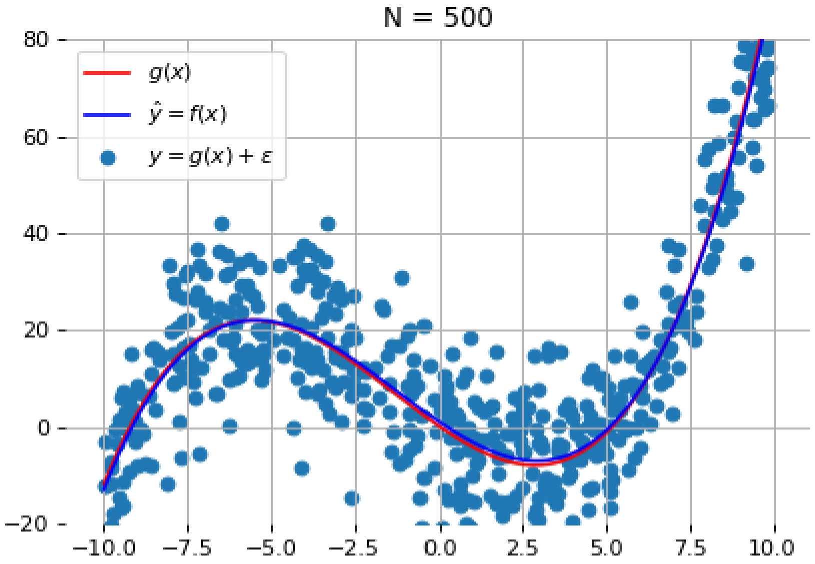
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

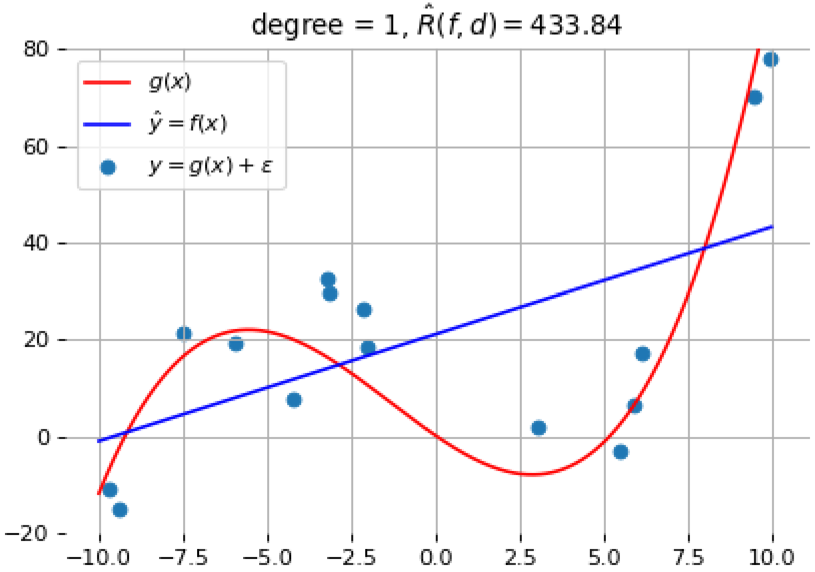


Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

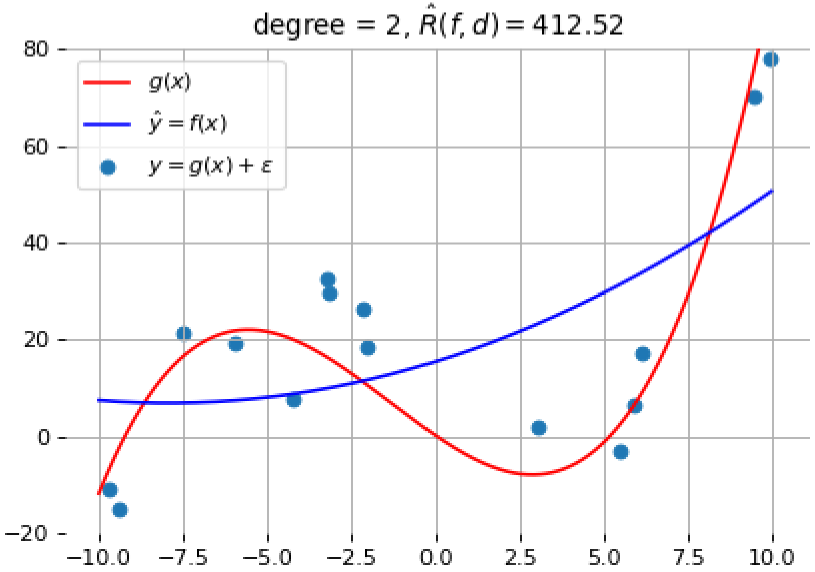
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 1

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

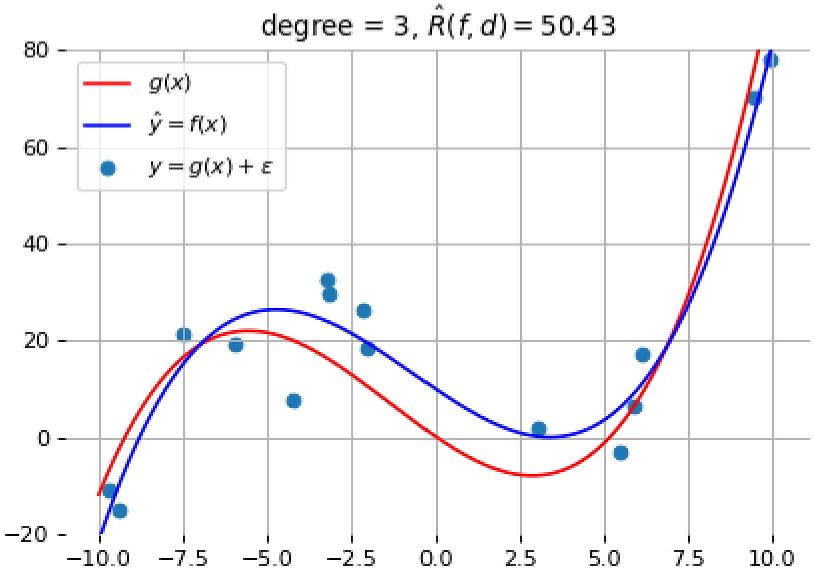
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 2

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

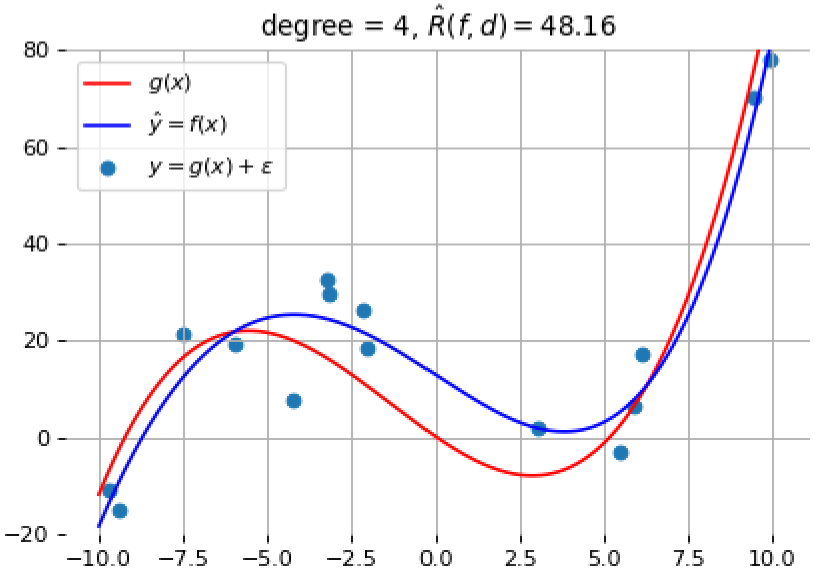
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 3

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

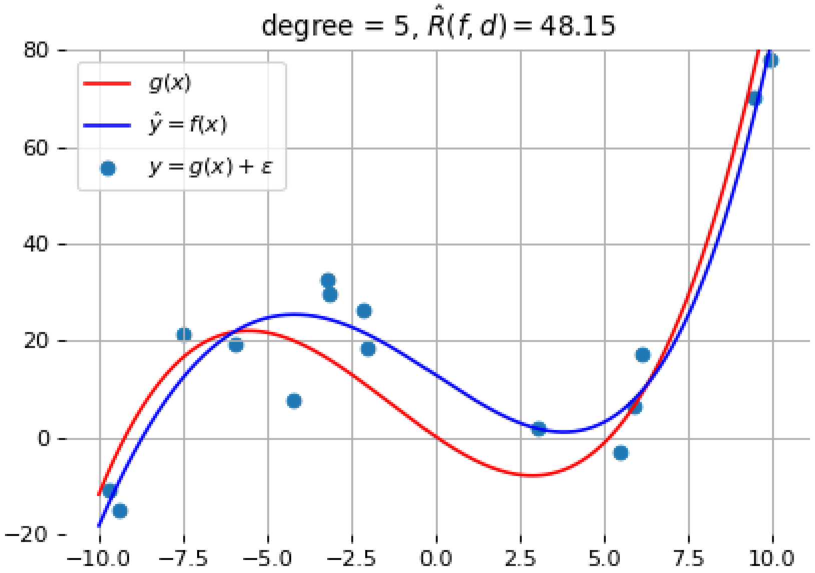
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 4

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

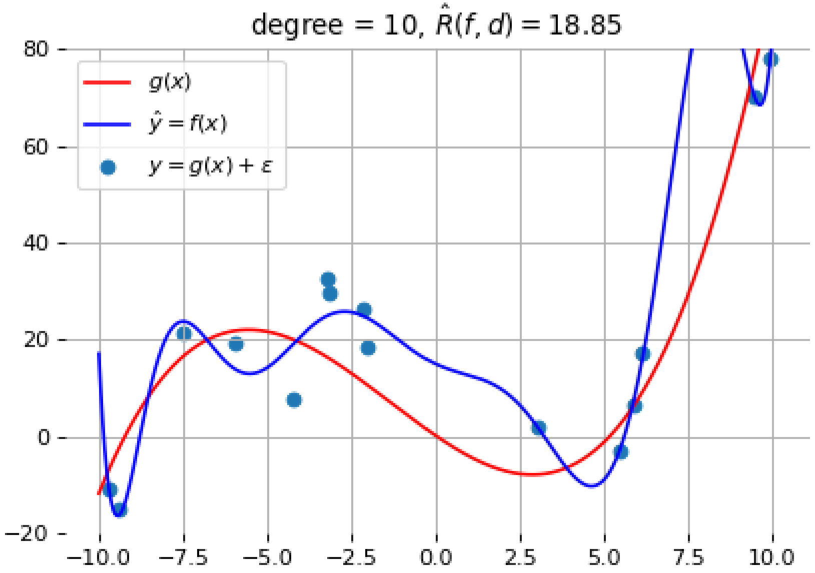
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 5

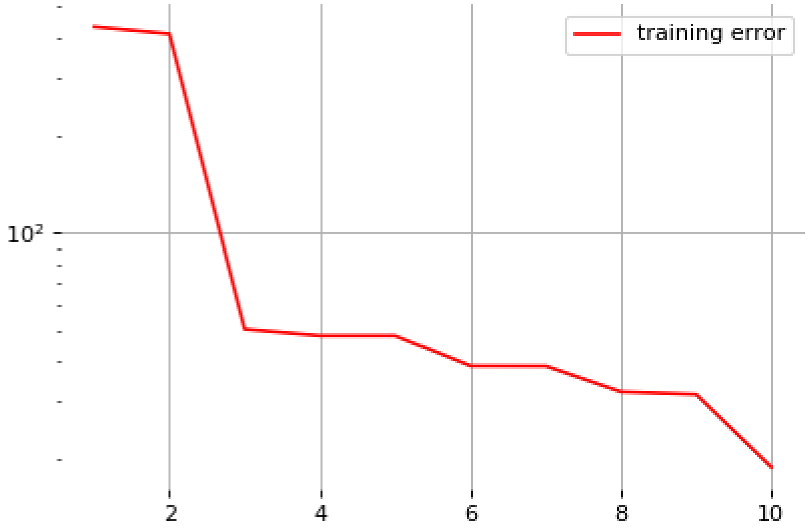
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions $f$ are either too "simple" or too "complex" with respect to the true data generating process?



$\mathcal{F}$ = polynomials of degree 10

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Error vs. degree $d$ of the polynomial.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Bayes risk and estimator

Let $\mathcal{Y}^{\mathcal{X}}$ be the set of all functions $f : \mathcal{X} \to \mathcal{Y}$.

We define the Bayes risk as the minimal expected risk over all possible functions,

$$R_B = \min_{f \in \mathcal{Y}^{\mathcal{X}}} R(f),$$

and call Bayes estimator the model $f_B$ that achieves this minimum.

**No model $f$ can perform better than $f_B$.**

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

The capacity of an hypothesis space $\mathcal{F}$ induced by a learning algorithm intuitively represents the ability to find a good model $f \in \mathcal{F}$ that can fit any function, regardless of its complexity.

If the capacity is infinite, we can fit any function, but in practice the capacity is always finite.

The capacity can be controlled through hyper-parameters of the learning algorithm. For example:

- The degree of the family of polynomials;
- The number of layers in a neural network;
- The number of training iterations;
- Regularization terms.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

## Underfitting and overfitting

- If the capacity of $\mathcal{F}$ is too low, then $f_B \notin \mathcal{F}$ and $R(f) - R_B$ is large for any $f \in \mathcal{F}$, including $f^\star$ and $f^\star_{\mathcal{D}}$. Such models $f$ are said to underfit the data.

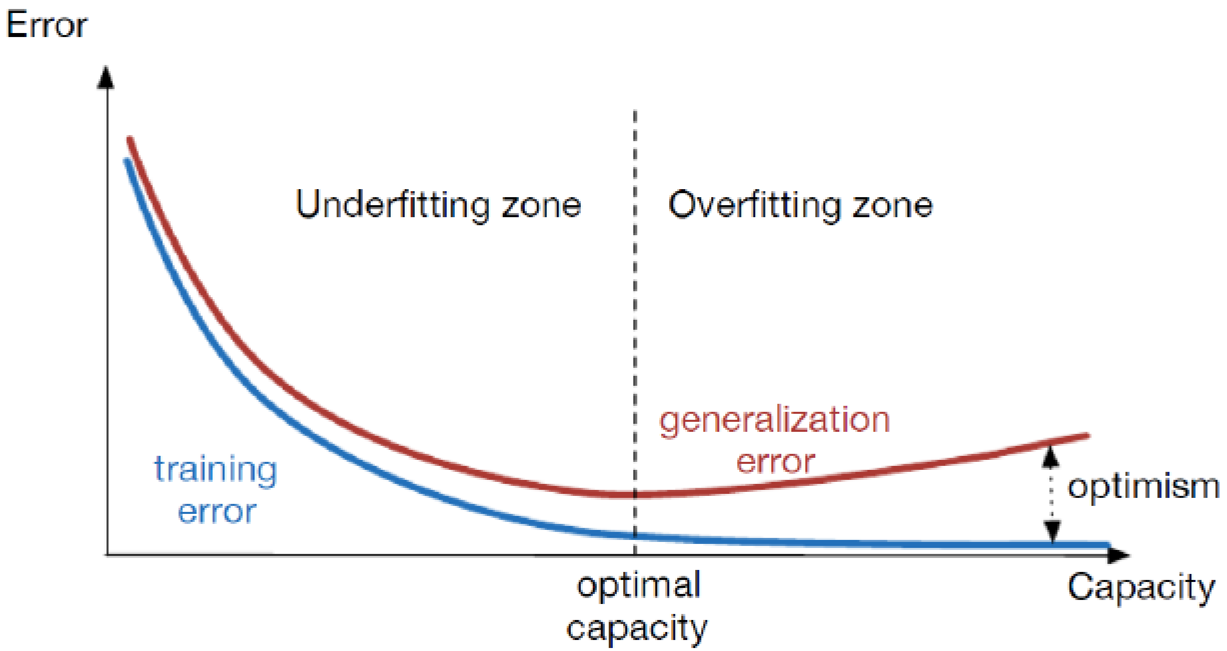
- If the capacity of $\mathcal{F}$ is too high, then $f_B \in \mathcal{F}$ or $R(f^\star) - R_B$ is small.
  However, because of the high capacity of the hypothesis space, the empirical risk minimizer $f^\star_{\mathcal{D}}$ could fit the training data arbitrarily well such that

$$R(f^\star_{\mathcal{D}}) \geq R_B \geq \hat{R}(f^\star_{\mathcal{D}}, \mathcal{D}) \geq 0.$$

  This indicates that the empirical risk $\hat{R}(f^\star_{\mathcal{D}}, \mathcal{D})$ is a poor estimator of the expected risk $R(f^\star_{\mathcal{D}})$. In this situation, $f^\star_{\mathcal{D}}$ becomes too specialized with respect to the true data generating process, $f^\star_{\mathcal{D}}$ is said to overfit the data.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

Therefore, our goal is to adjust the capacity of the hypothesis space such that the expected risk of the empirical risk minimizer (the generalization error) $R(f^\star_\mathcal{D})$ gets as low as possible, and not simply the empirical risk of the empirical risk minimizer (training error) $\hat{R}(f^\star_\mathcal{D}, \mathcal{D})$.



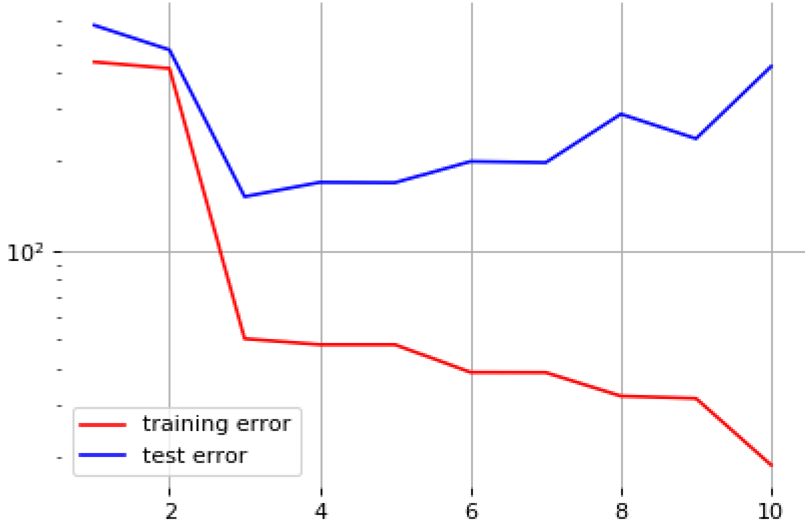
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

An unbiased estimate of the expected risk can be obtained by evaluating $f^\star_{\mathcal{D}}$ on data $\mathcal{D}_\text{test}$ independent from the training samples $\mathcal{D}$:

$$\hat{R}(f^\star_{\mathcal{D}}, \mathcal{D}_\text{test}) = \frac{1}{N_\text{test}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_\text{test}} \ell(y_i, f^\star_{\mathcal{D}}(\mathbf{x}_i))$$

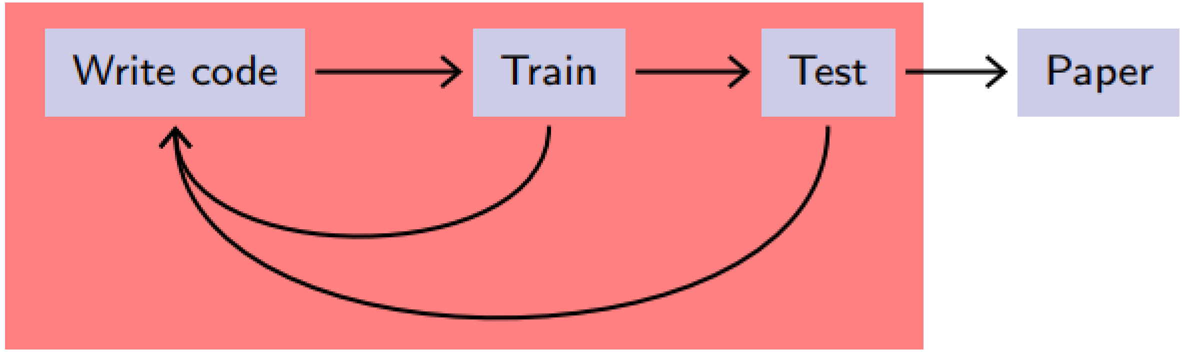
This test error estimate can be used to evaluate the actual performance of the model. However, it should not be used, at the same time, for model selection.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



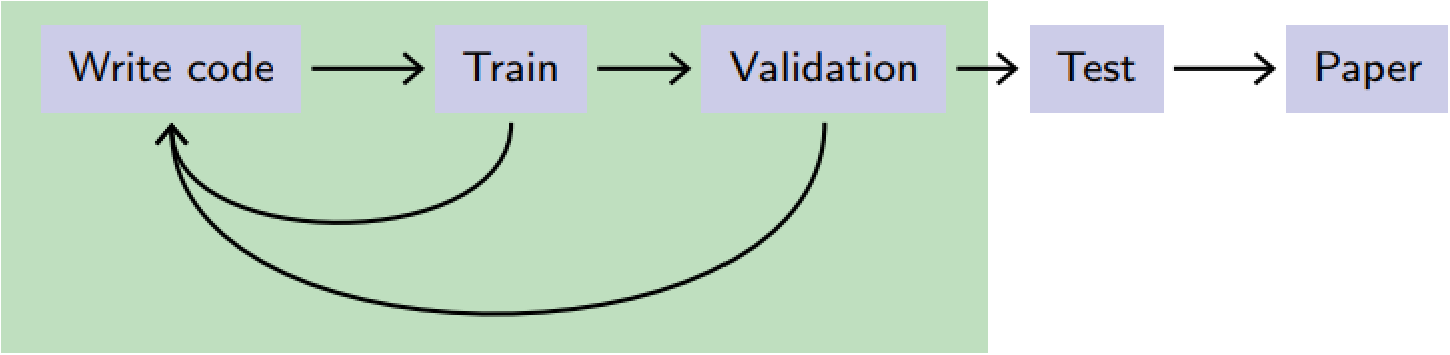
Error vs. degree $d$ of the polynomial.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



This should be avoided at all costs!

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.



Instead, keep a separate validation set for tuning the hyper-parameters.

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

# Bias-variance decomposition

Consider a fixed point $\mathbf{x}$ and the prediction $\hat{y} = f^\star_{\mathcal{D}}(\mathbf{x})$ of the empirical risk minimizer at $\mathbf{x}$.

Then the local expected risk of $f^\star_{\mathcal{D}}$ is

$$\begin{aligned} R(f^\star_{\mathcal{D}}|\mathbf{x}) &= \mathbb{E}_{p^\star(y|\mathbf{x})}\left[(y - f^\star_{\mathcal{D}}(\mathbf{x}))^2\right] \\ &= \mathbb{E}_{p^\star(y|\mathbf{x})}\left[(y - f_B(\mathbf{x}) + f_B(\mathbf{x}) - f^\star_{\mathcal{D}}(\mathbf{x}))^2\right] \\ &= \mathbb{E}_{p^\star(y|\mathbf{x})}\left[(y - f_B(\mathbf{x}))^2\right] + \mathbb{E}_{p^\star(y|\mathbf{x})}\left[(f_B(\mathbf{x}) - f^\star_{\mathcal{D}}(\mathbf{x}))^2\right] \\ &= R(f_B|\mathbf{x}) + (f_B(\mathbf{x}) - f^\star_{\mathcal{D}}(\mathbf{x}))^2 \end{aligned}$$

- $R(f_B|\mathbf{x})$ is the local expected risk of the Bayes estimator. This term cannot be reduced.
- $(f_B(\mathbf{x}) - f^\star_{\mathcal{D}}(\mathbf{x}))^2$ represents the discrepancy between $f_B$ and $f^\star_{\mathcal{D}}$.

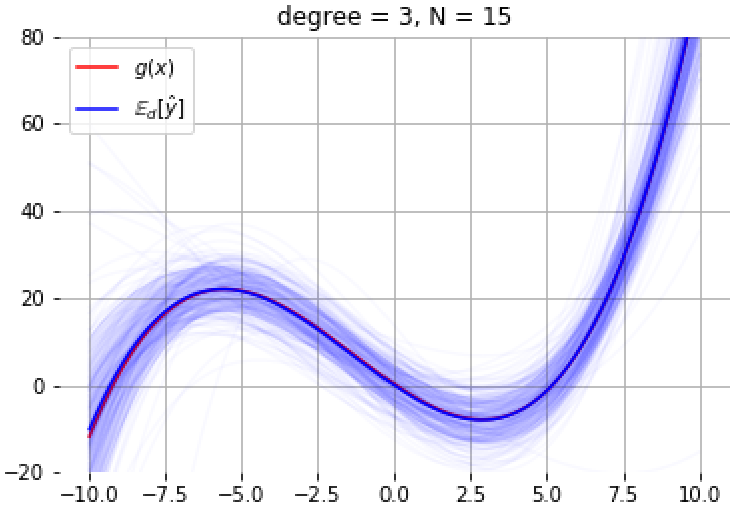
Remarks:

- $R(f) = \mathbb{E}_{p^\star(\mathbf{x},y)}\left[\ell(y, f(\mathbf{x}))\right] = \mathbb{E}_{p^\star(\mathbf{x})}\left[\mathbb{E}_{p^\star(y|\mathbf{x})}\left[\ell(y, f(\mathbf{x}))\right]\right] = \mathbb{E}_{p^\star(\mathbf{x})}\left[R(f|\mathbf{x})\right]$
- To go from the second to third line we used the fact that $f_B(\mathbf{x}) = \arg\min_{f \in \mathcal{Y}^\mathcal{X}} R(f) = \mathbb{E}_{p^\star(y|\mathbf{x})}[y]$.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

If $\mathcal{D}$ is itself considered as a random variable, then $f^\star_{\mathcal{D}}$ is also a random variable, along with its predictions $\hat{y} = f^\star_{\mathcal{D}}(\mathbf{x})$.

Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

Formally, the expected local expected risk yields to:

$$\begin{aligned}
&\mathbb{E}_\mathcal{D}\left[R(f^\star_\mathcal{D}|\mathbf{x})\right] \\
&= \mathbb{E}_\mathcal{D}\left[R(f_B|\mathbf{x}) + (f_B(\mathbf{x}) - f^\star_\mathcal{D}(\mathbf{x}))^2\right] \\
&= R(f_B|\mathbf{x}) + \mathbb{E}_\mathcal{D}\left[(f_B(\mathbf{x}) - f^\star_\mathcal{D}(\mathbf{x}))^2\right] \\
&= \underbrace{R(f_B|\mathbf{x})}_{\text{noise}} + \underbrace{(f_B(\mathbf{x}) - \mathbb{E}_\mathcal{D}\left[f^\star_\mathcal{D}(\mathbf{x})\right])^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_\mathcal{D}\left[(f^\star_\mathcal{D}(\mathbf{x}) - \mathbb{E}_\mathcal{D}\left[f^\star_\mathcal{D}(\mathbf{x})\right])^2\right]}_{\text{var}}
\end{aligned}$$

This decomposition is known as the bias-variance decomposition.

- The noise term quantity is the irreducible part of the expected risk.
- The bias term measures the discrepancy between the average model and the Bayes estimator.
- The variance term quantities the variability of the predictions.

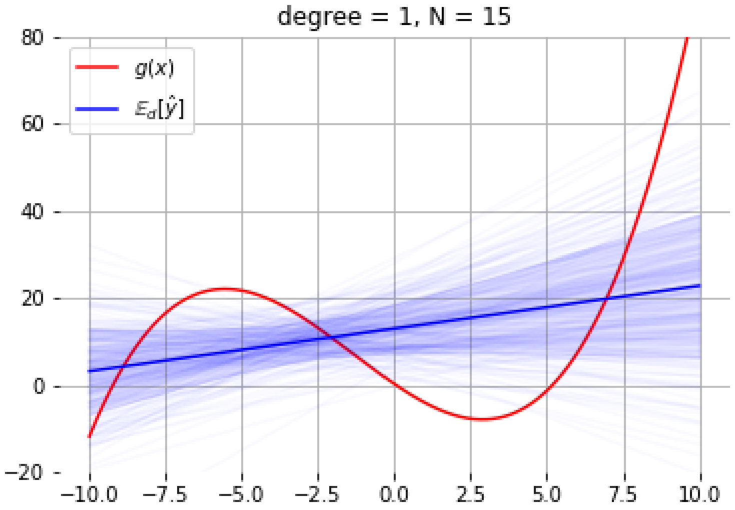
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Bias-variance trade-off

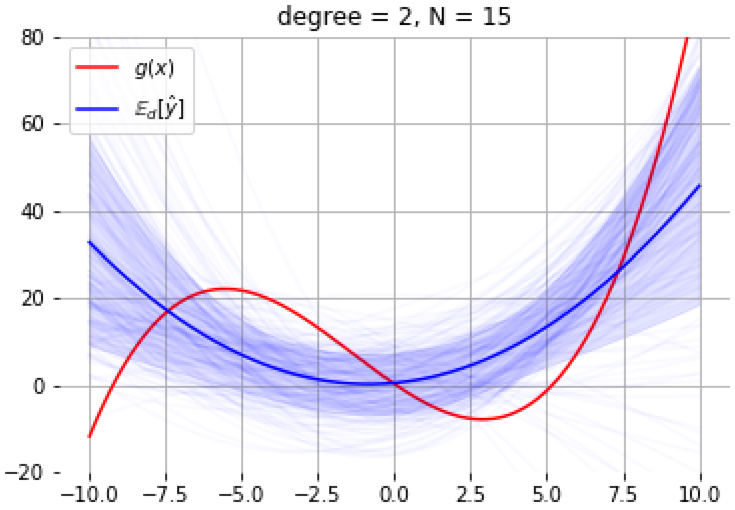
- Reducing the capacity makes $f^\star_{\mathcal{D}}$ fit the data less on average, which increases the bias term.

- Increasing the capacity makes $f^\star_{\mathcal{D}}$ vary a lot with the training data, which increases the variance term.

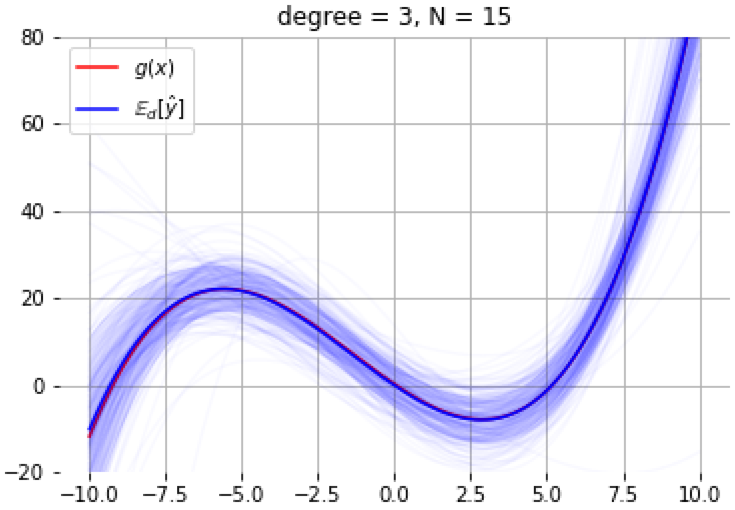
Credits: Francois Fleuret, EE559 Deep Learning, EPFL.



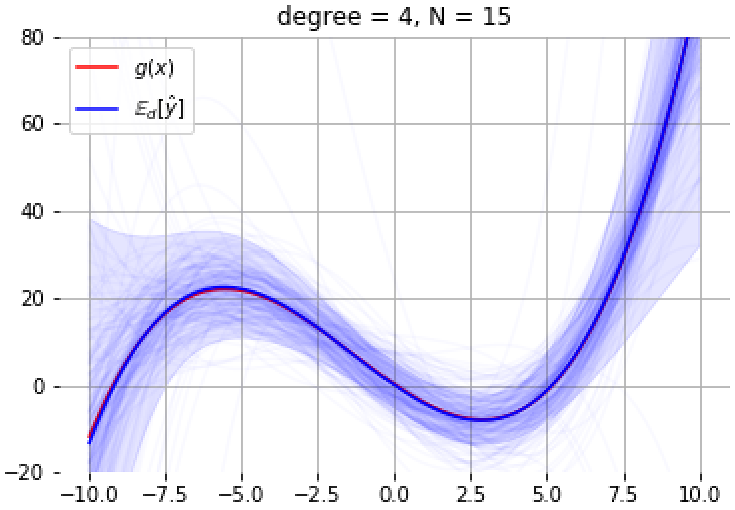
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



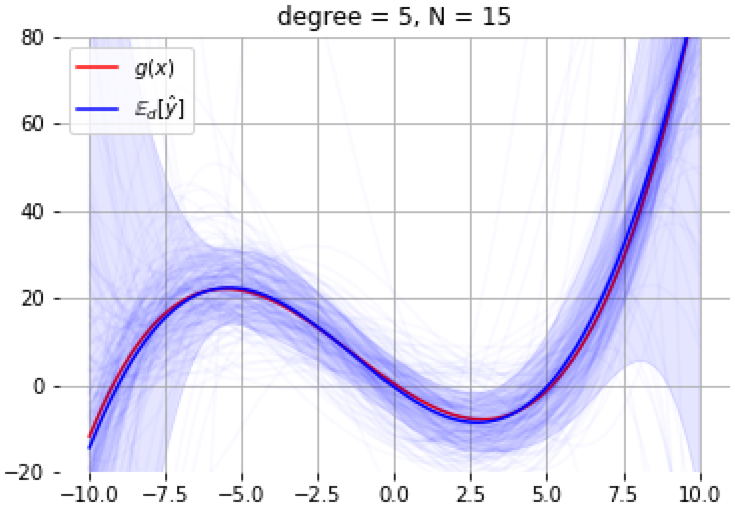
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

# Lab session on multinomial logistic regression