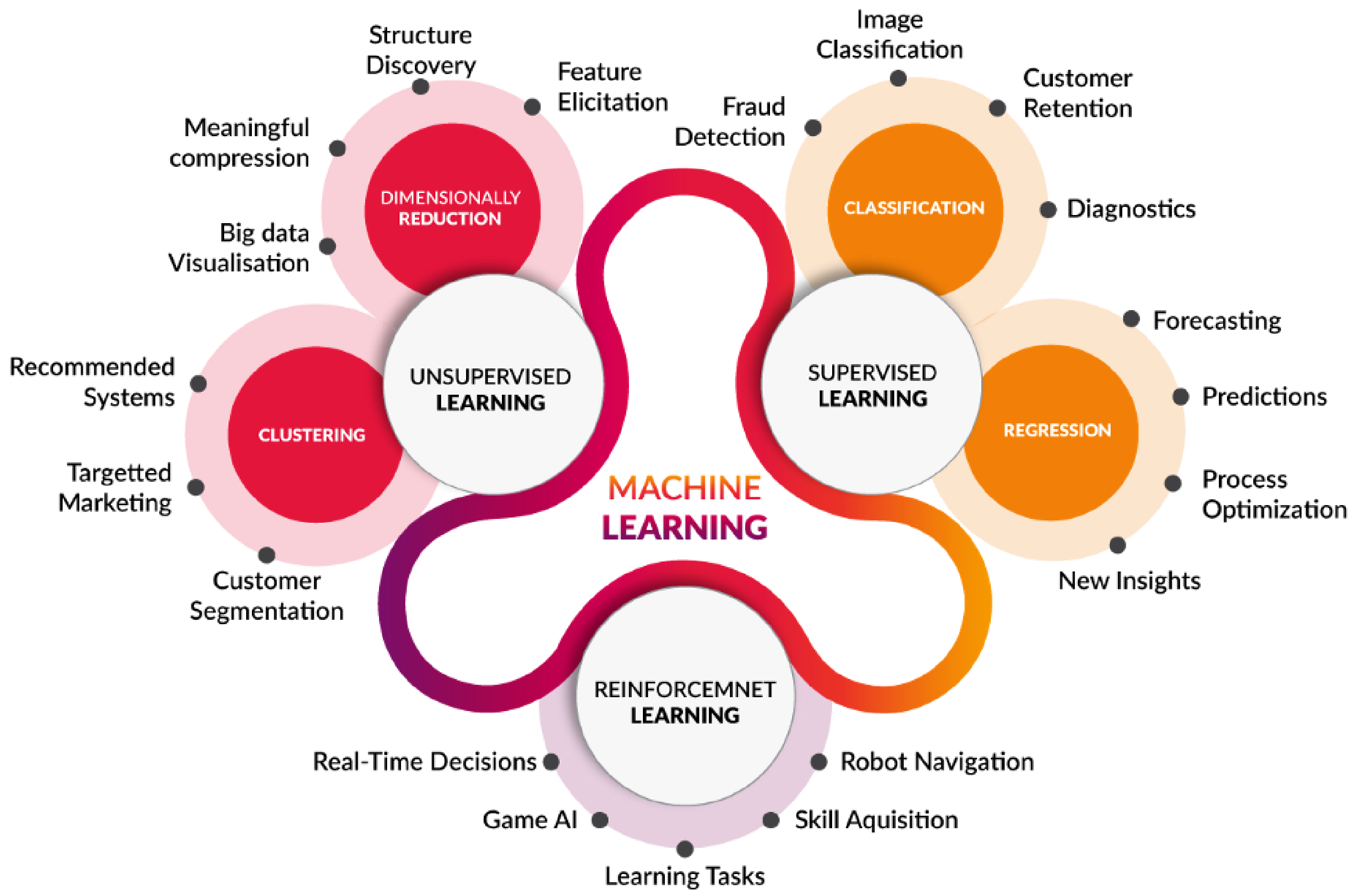
Bayesian Methods for Machine Learning

Lecture 2 - Fundamentals of machine learning

Simon Leglaive

CentraleSupélec, 2020-2021



Today, we will focus on supervised learning.

Image credits: Kamil Krzyk, Types of Machine Learning

Supervised learning

Consider an unknown joint probability distribution $p^\star(\mathbf{x}, y)$.

Assume training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1,...,N}$ where

$$(\mathbf{x}_i, y_i) \sim p^\star(\mathbf{x}, y).$$

- For instance $\mathbf{x}_i$ is a p-dimensional input feature vector and y_i is a scalar label (e.g., a category or a real value).
- The training data is generated i.i.d.
- The training data can be of any finite size N.
- In general, we do not have any prior information about $p^\star(\mathbf{x}, y)$.

Inference

Supervised learning is usually concerned with the two following inference problems:

- Classification: Given $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, ..., C\}\}_{i=1,...,N}$, we want to estimate for any **new** input $\mathbf{x}$:

$$\arg\max_y p(y|\mathbf{x}).$$

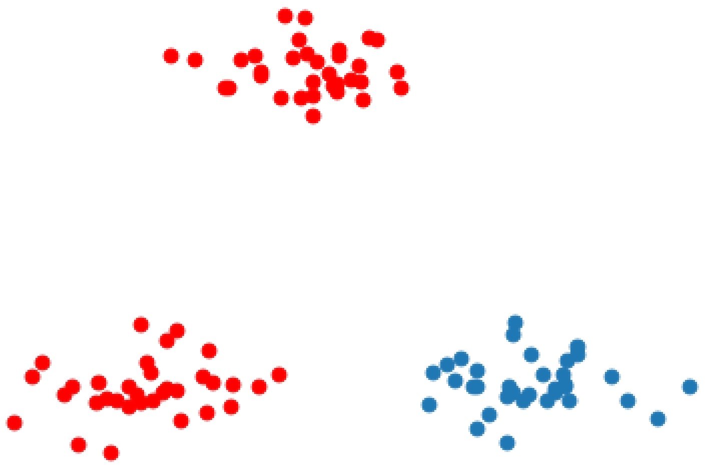
- Regression: Given $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \mathbb{R}\}_{i=1,...,N}$, we want to estimate for any **new** input $\mathbf{x}$:

$$\mathbb{E}_{p(y|\mathbf{x})}\left[y\right] = \int_{\mathcal{Y}} y\, p(y|\mathbf{x}) dy.$$

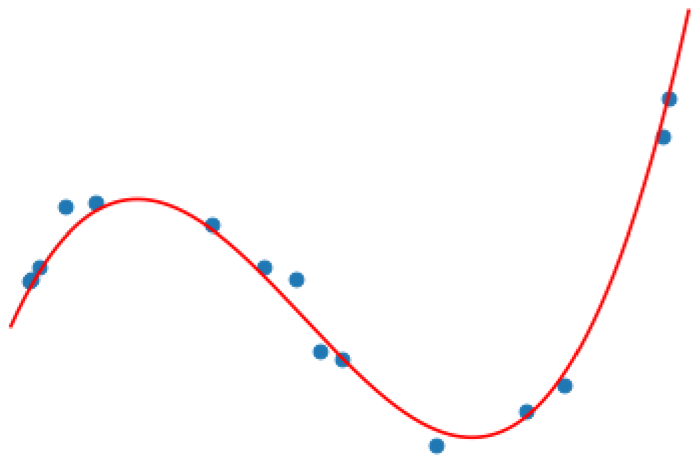
Or more generally, inference consists in computing the posterior distribution

$$p(y|\mathbf{x})$$

for any new $\mathbf{x}$ (i.e. that was not part of the training dataset).



Classification consists in identifying
a decision boundary between objects of distinct classes.



Regression aims at estimating relationships among (usually continuous) variables.

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

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

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

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:

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

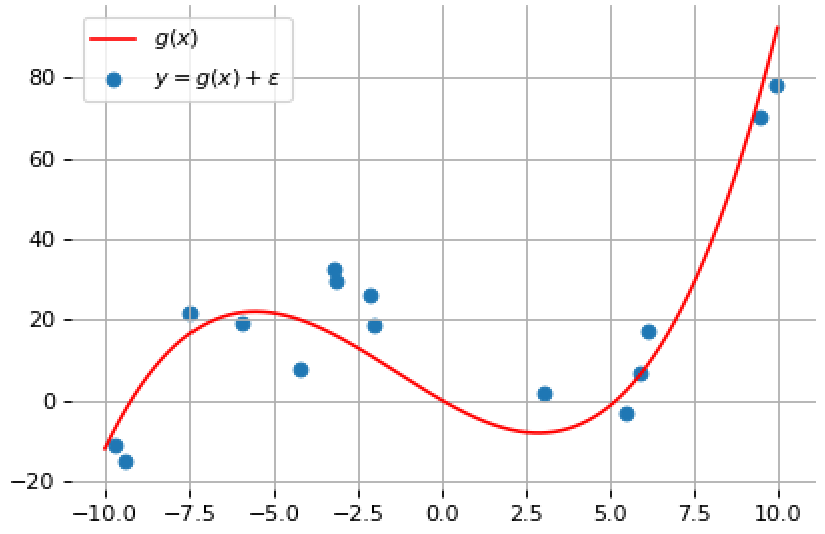
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

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

Under regularity assumptions, empirical risk minimizers converge:

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

Regression example

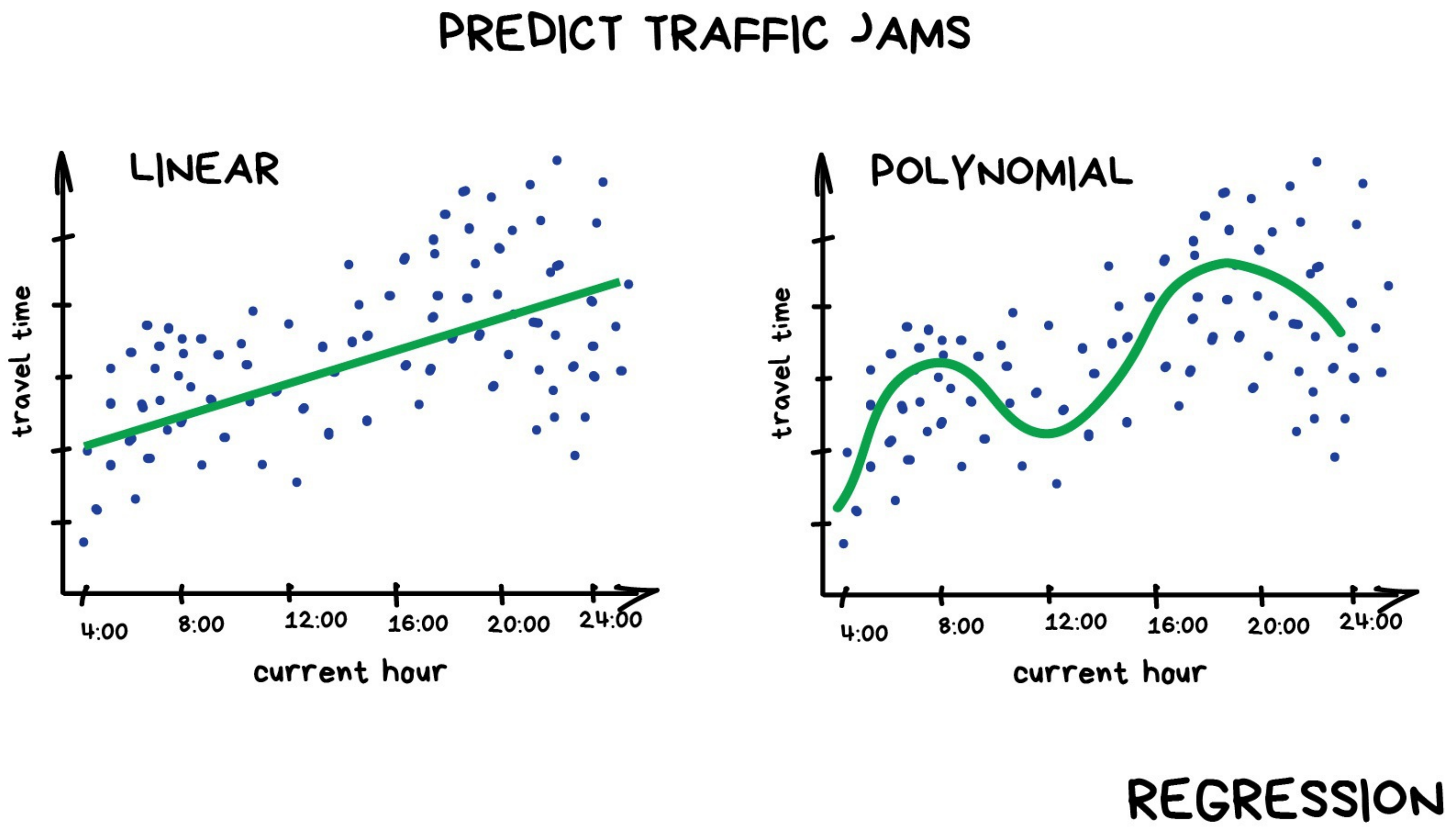


Consider the joint probability distribution $p^\star(x, y)$ induced by the data generating process

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

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

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

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

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

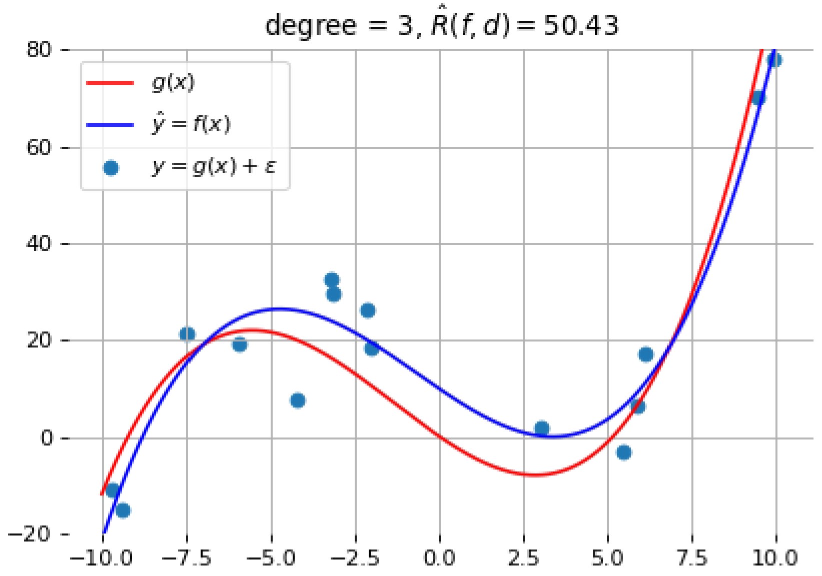
Step 3: Training

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:

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

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

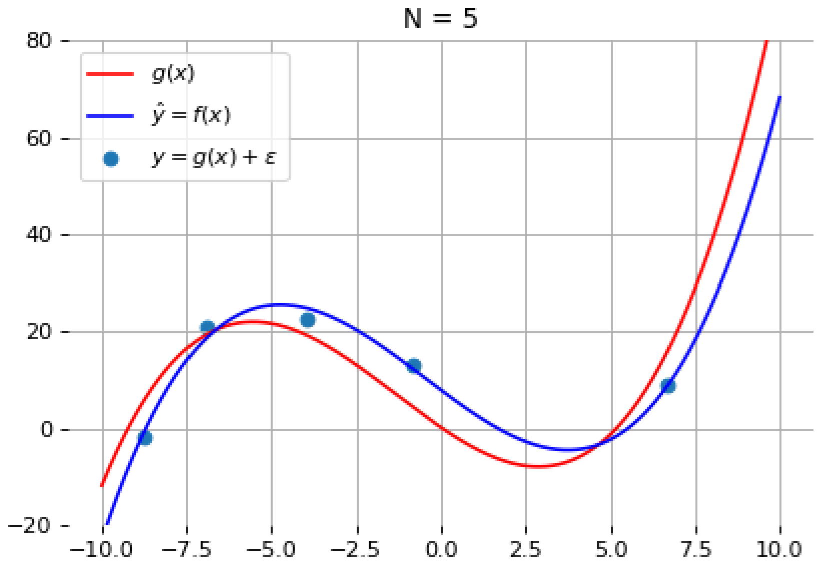
$$\mathbf{w}^\star_{\mathcal{D}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



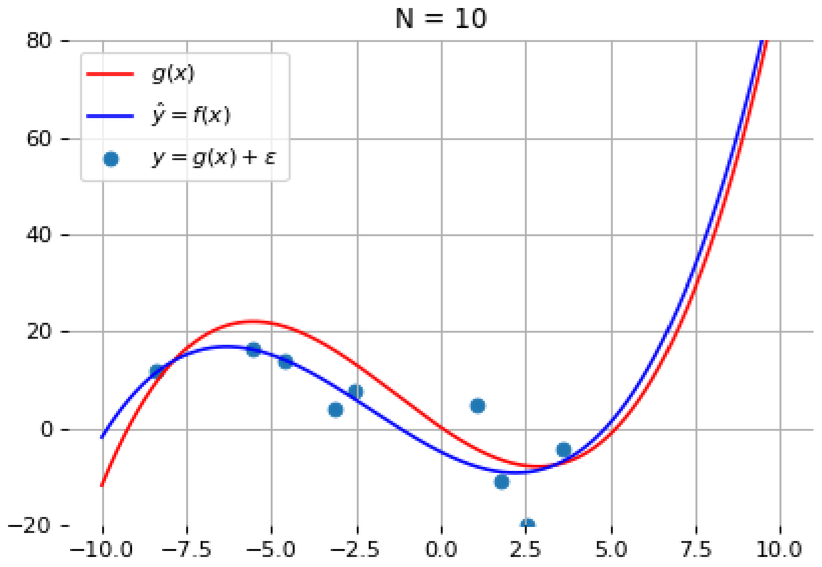
In many situations, the problem is more difficult and we cannot find the solution analytically. We resort to **iterative optimization algorithms**, such as (variants of) gradient descent.

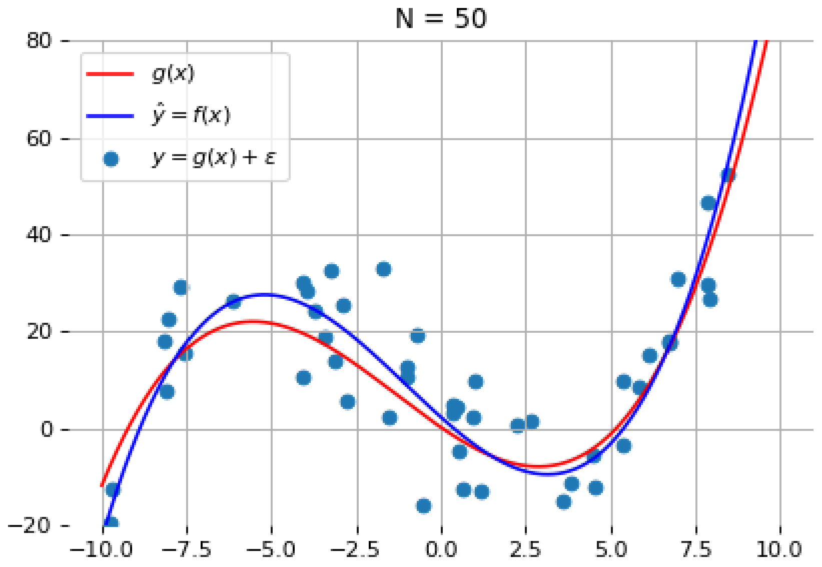
The expected risk minimizer $f(x;\mathbf{w}^\star)$ within our hypothesis space $\mathcal{F}$ (polynomials of degree 3) is $g(x)$ itself (i.e. the polynomial of degree 3 with the true parameters).

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 $N \to \infty$.

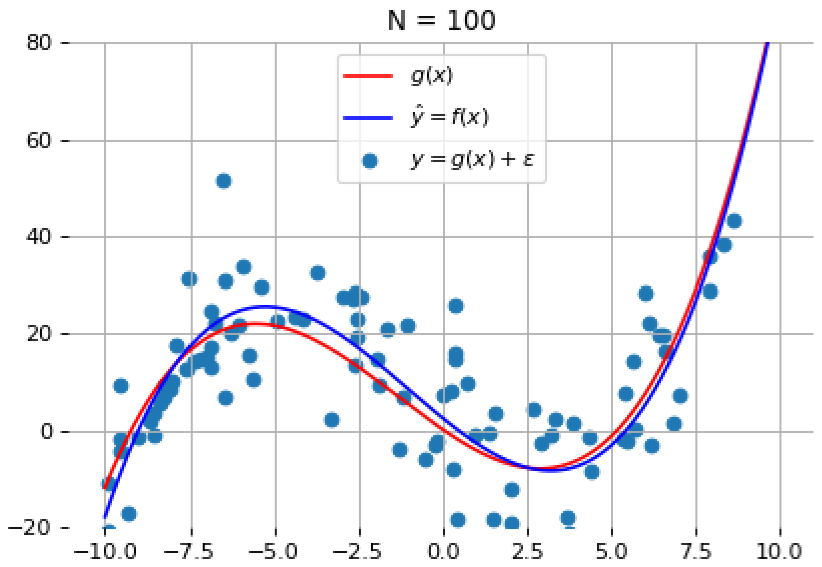


Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

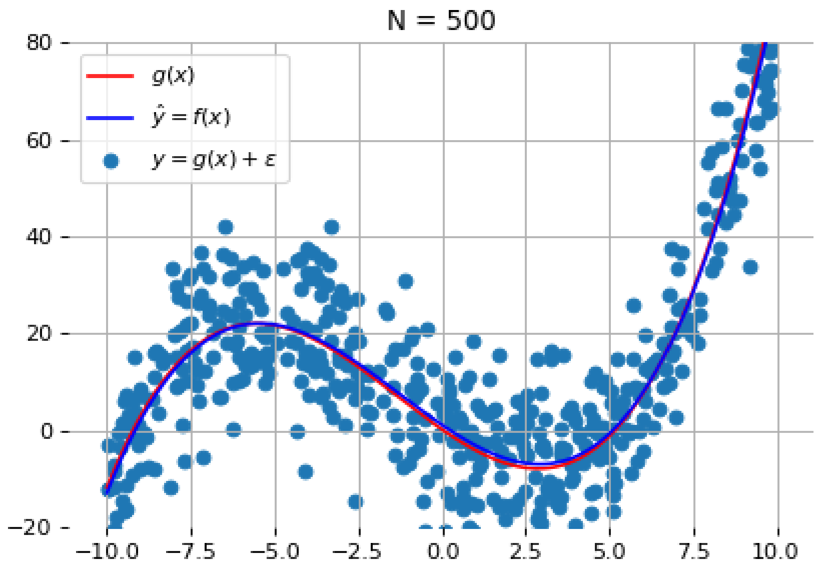

N = 10
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \epsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



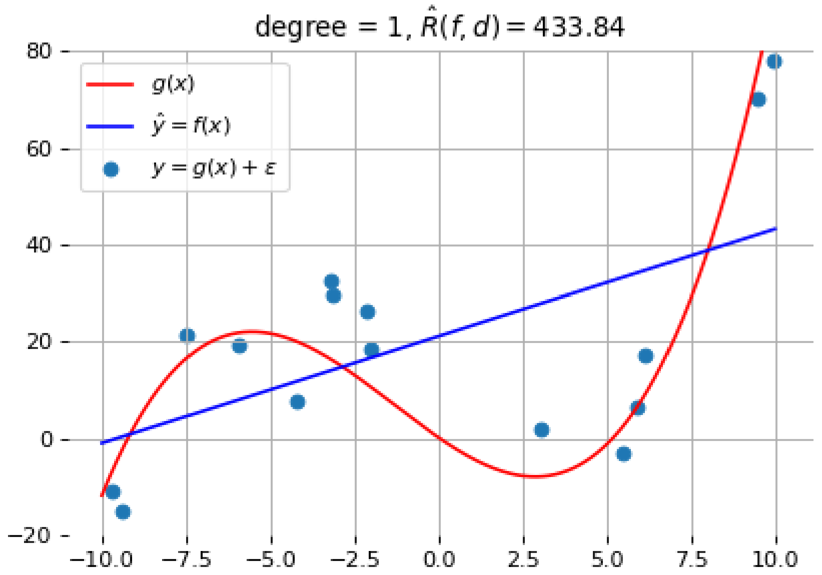
Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.



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

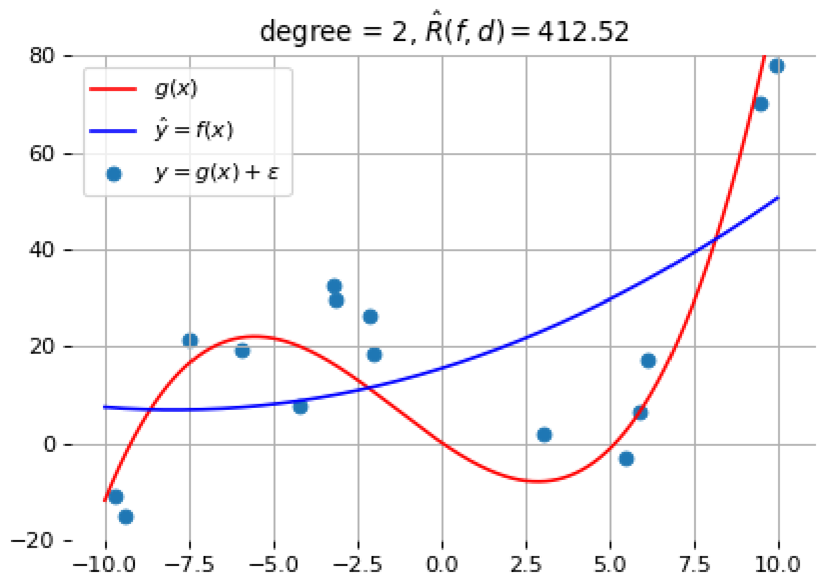

N = 500
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \epsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

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?



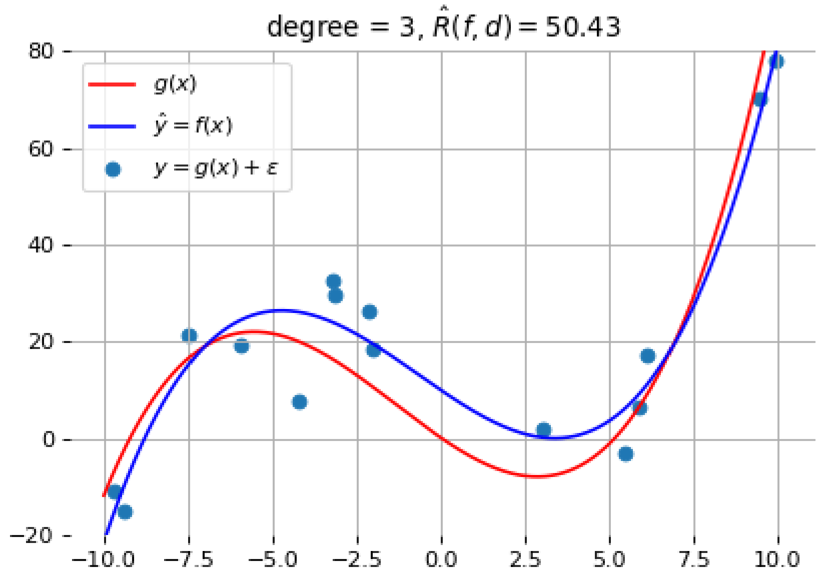
$\mathcal{F}$ = polynomials of degree 1

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?



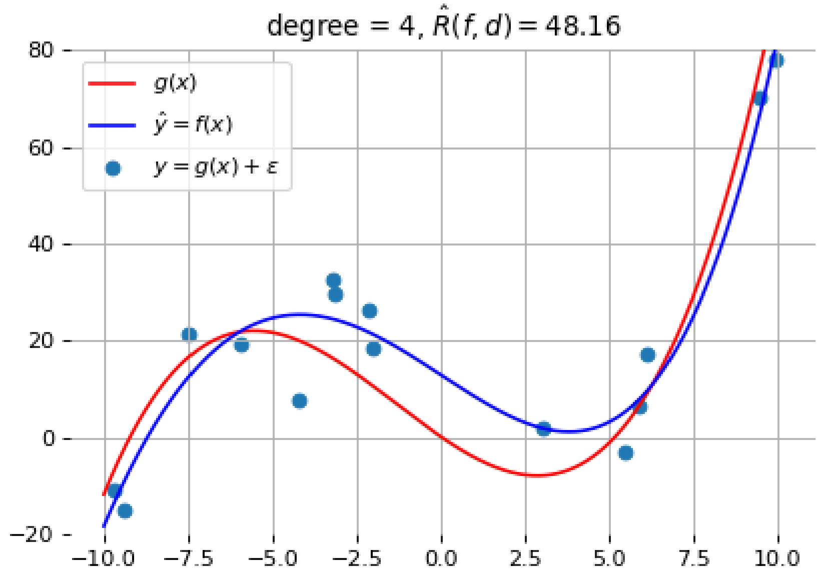
$\mathcal{F}$ = polynomials of degree 2

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

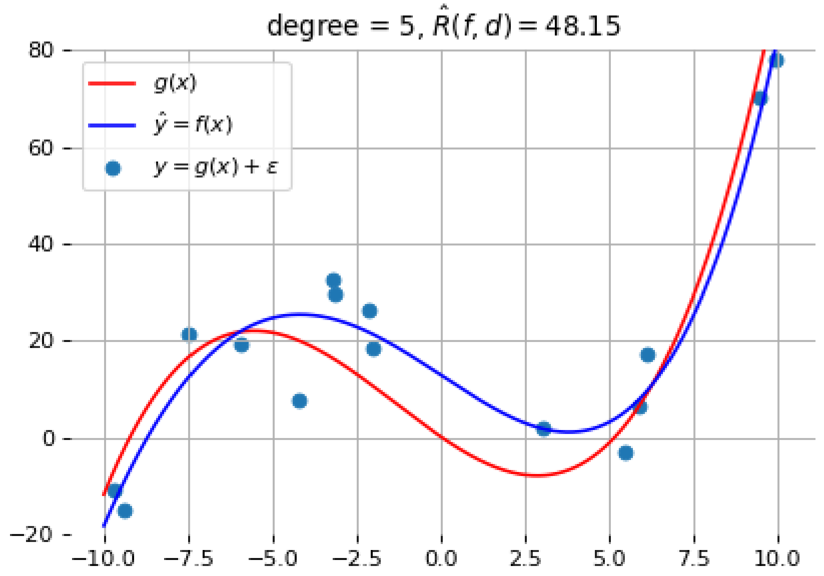
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions f are either too "simple" or too "complex" with respect to the true data generating process?



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

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

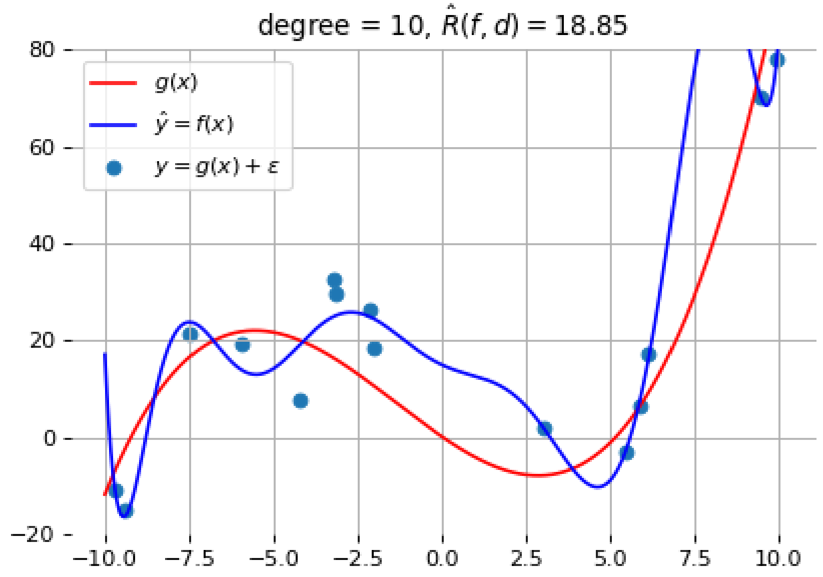
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions f are either too "simple" or too "complex" with respect to the true data generating process?



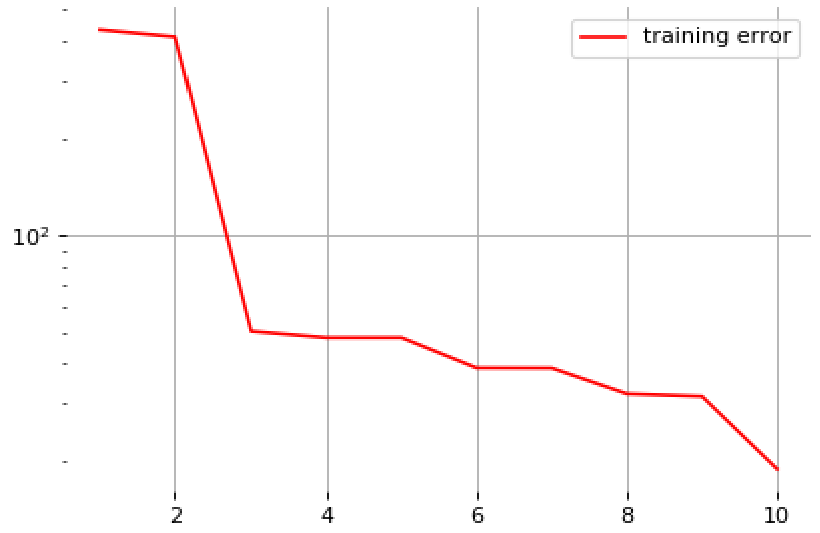
$\mathcal{F}$ = polynomials of degree 5

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



Error vs. degree d of the polynomial.

Bayes risk and model

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 model 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.

In practice, 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.

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.

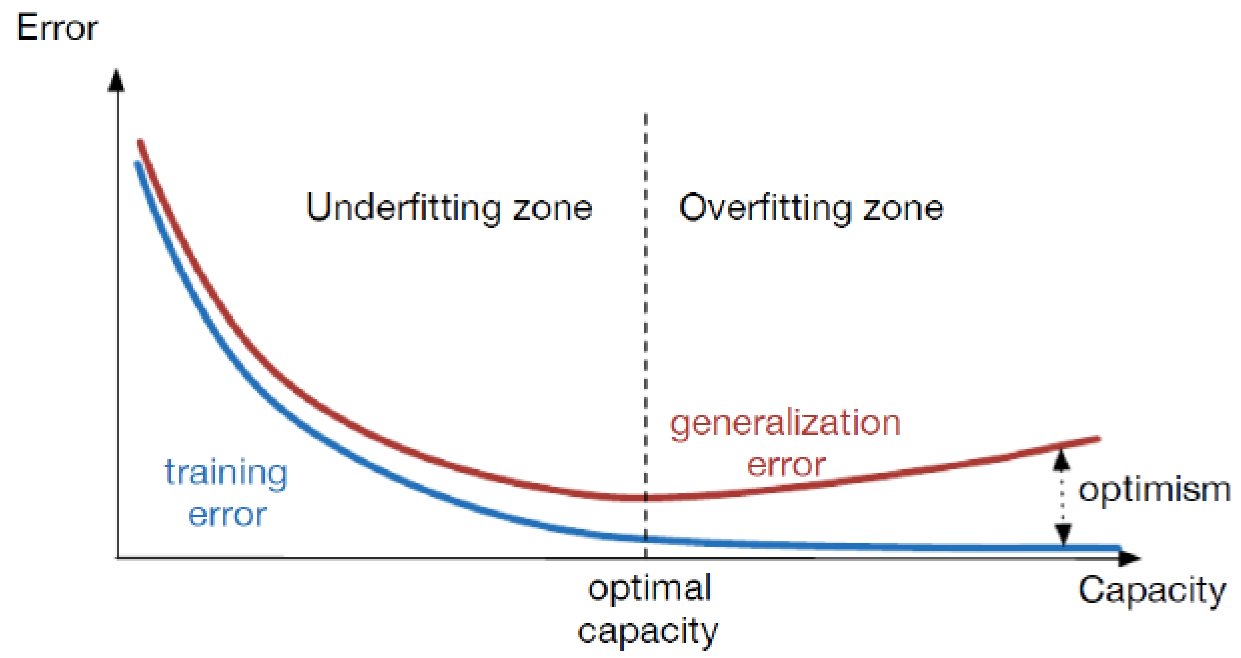
- If the capacity of $\mathcal{F}$ is too high, then $f_B \in \mathcal{F}$ or $R(f^\star) - R_B$ is small.
  However, because of the high capacity of the hypothesis space, the empirical risk minimizer $f^\star_\mathcal{D}$ could fit the training data arbitrarily well such that

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

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

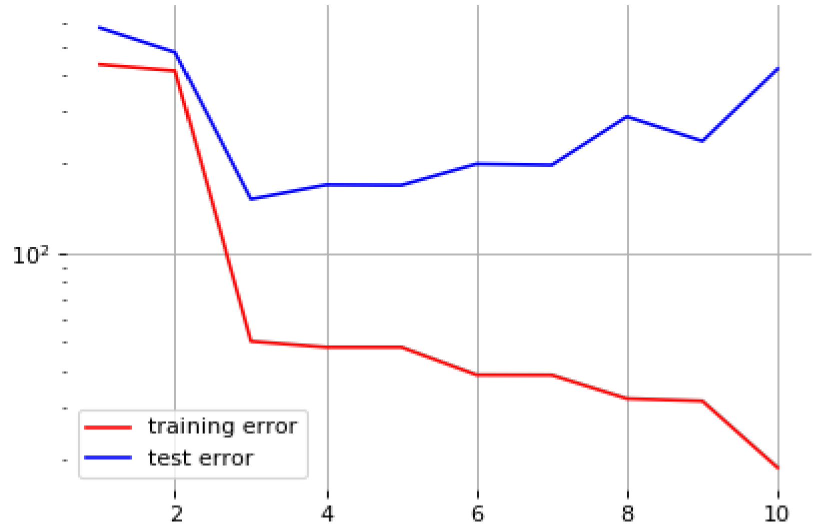


Credits: Gilles Louppe, INFO8010 - Deep Learning, ULiège.

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 $\mathcal{D}$:

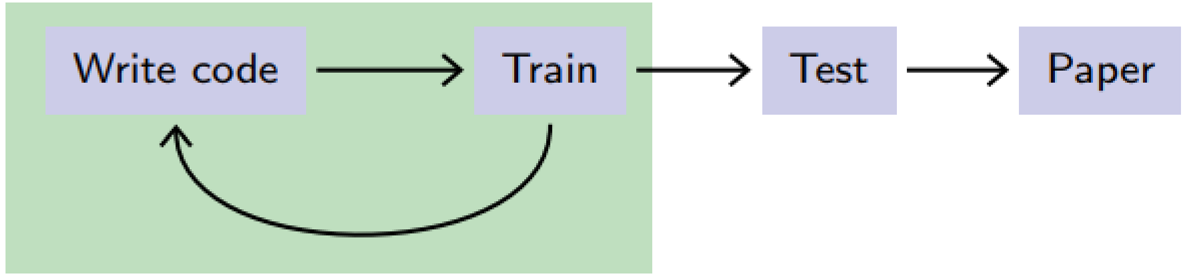
$$\hat{R}(f^\star_{\mathcal{D}}, \mathcal{D}_\text{test}) = \frac{1}{N_\text{test}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_\text{test}} \ell(y_i, f^\star_{\mathcal{D}}(\mathbf{x}_i))$$

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.



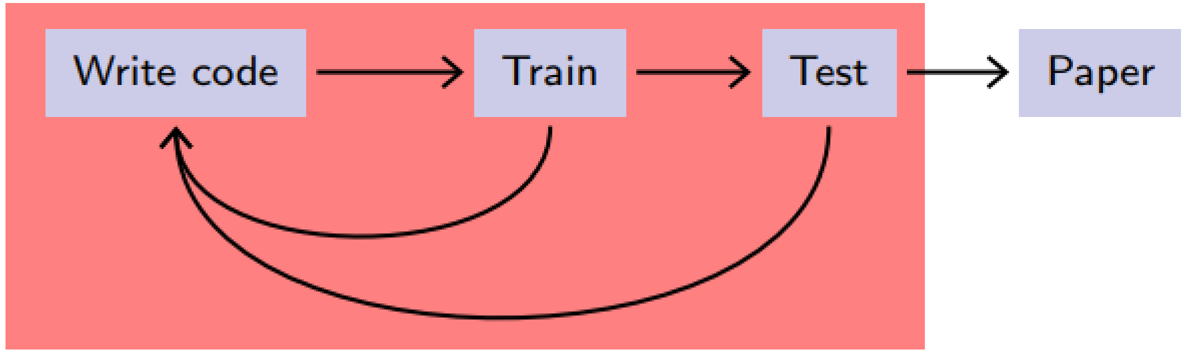
Error vs. degree d of the polynomial.

(Proper) evaluation protocol



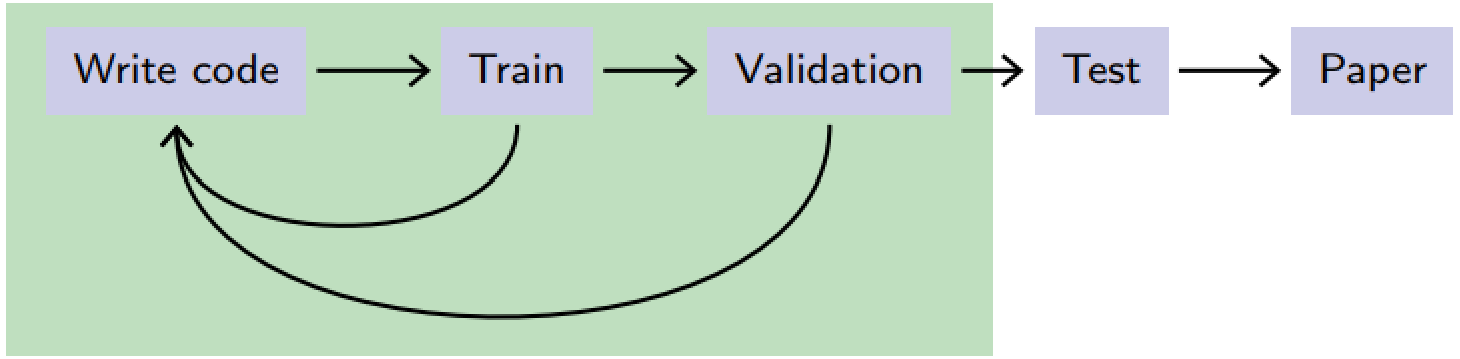
There may be over-fitting, but it does not bias the final performance evaluation.

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.



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

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

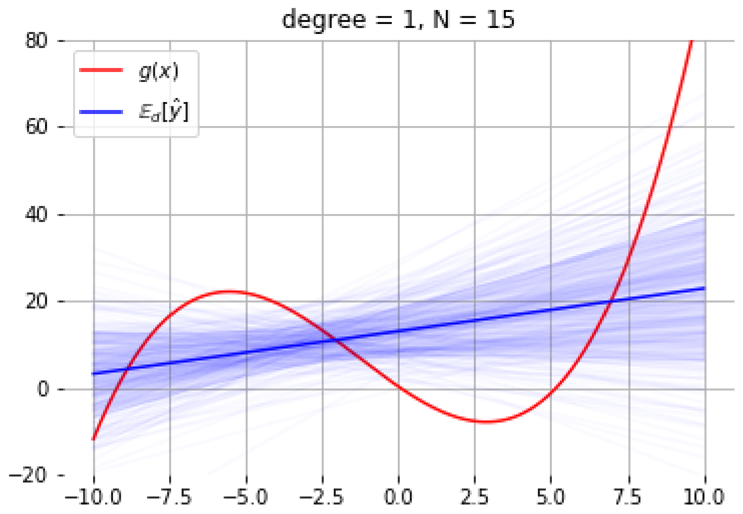
$$\begin{aligned}
R(f^\star_{\mathcal{D}}|x) &= \mathbb{E}_{p^\star(y|x)}\left[(y - f^\star_{\mathcal{D}}(x))^2\right] \\
&= \mathbb{E}_{p^\star(y|x)}\left[(y - f_B(x) + f_B(x) - f^\star_{\mathcal{D}}(x))^2\right] \\
&= \mathbb{E}_{p^\star(y|x)}\left[(y - f_B(x))^2\right] + \mathbb{E}_{p^\star(y|x)}\left[(f_B(x)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

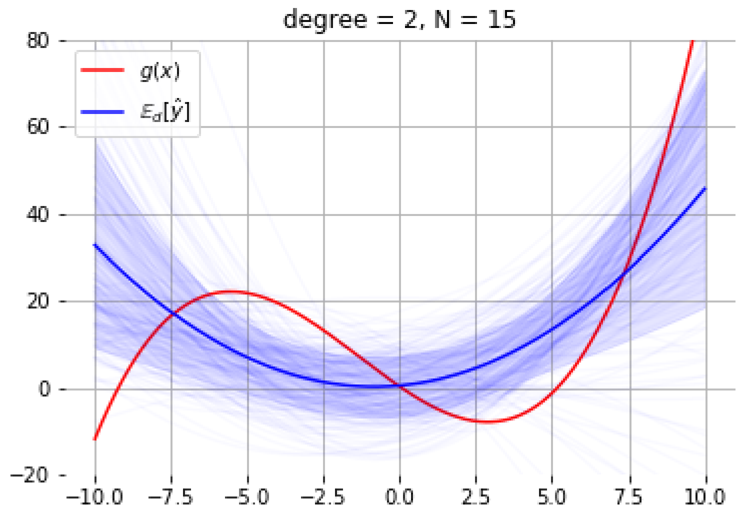
where

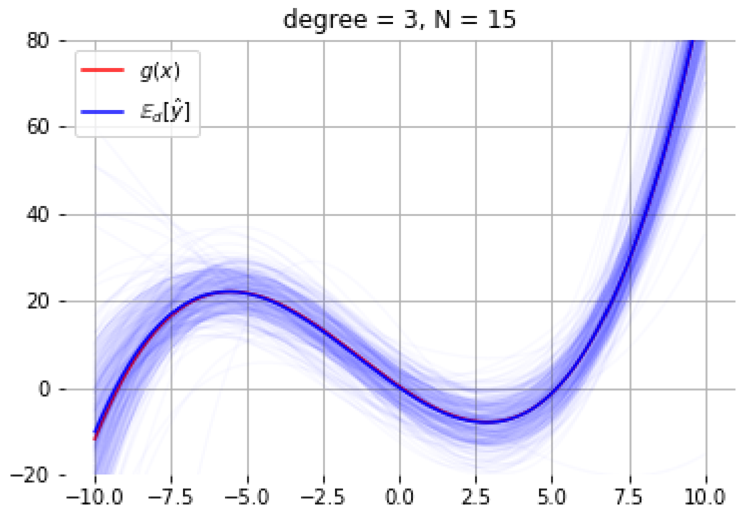
- $R(f_B|x)$ is the local expected risk of the Bayes model. This term cannot be reduced.
- $(f_B(x) - f^\star_{\mathcal{D}}(x))^2$ represents the discrepancy between f_B and $f^\star_{\mathcal{D}}$.

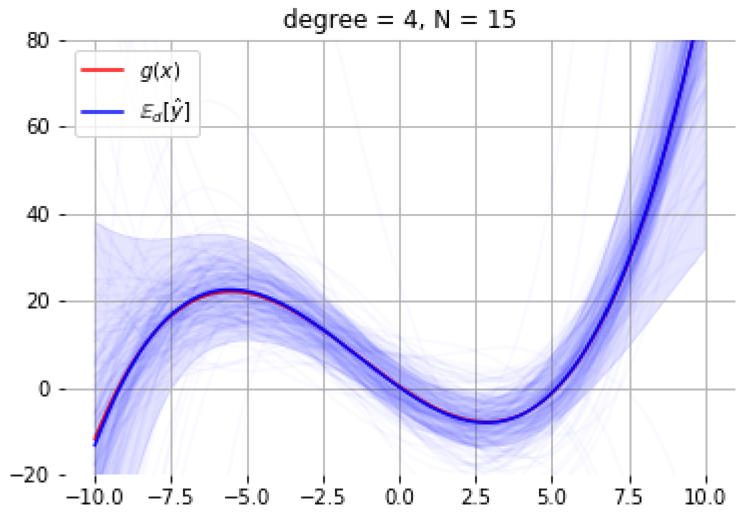
Note that: $R(f) = \mathbb{E}_{p^\star(\mathbf{x},y)}\left[\ell(y, f(\mathbf{x}))\right] = \mathbb{E}_{p^\star(\mathbf{x})}\left[\mathbb{E}_{p^\star(y|\mathbf{x})}\left[\ell(y, f(\mathbf{x}))\right]\right] = \mathbb{E}_{p^\star(\mathbf{x})}\left[R(f|x)\right]$

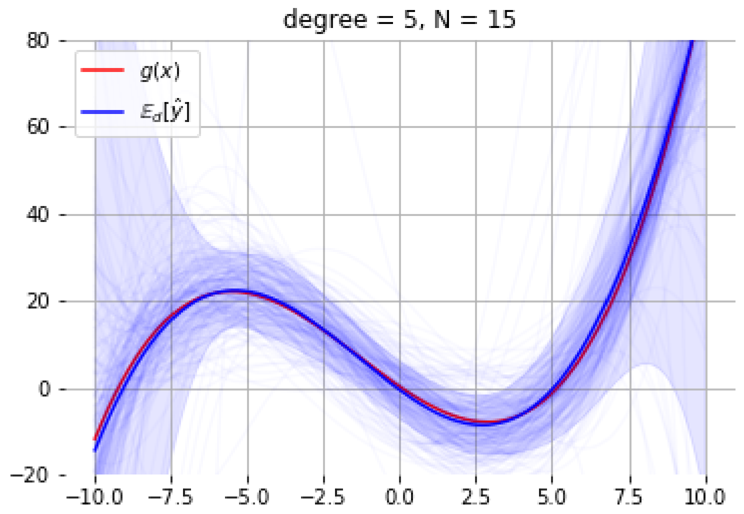
If $\mathcal{D}$ is itself considered as a random variable, then $f^{\star}_{\mathcal{D}}$ is also a random variable, along with its predictions $\hat{y}$.


degree = 1, N = 15
$g(x)$
$E_d[\hat{y}]$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


degree = 2, N = 15
$g(x)$
$E_d[\hat{y}]$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


degree = 3, N = 15
$g(x)$
$E_d[\hat{y}]$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


degree = 4, N = 15
$g(x)$
$E_d[\hat{y}]$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


degree = 5, N = 15
$g(x)$
$E_d[\hat{y}]$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Formally, the expected local expected risk yields to:

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

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 model.
- The variance term quantities the variability of the predictions.

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.

Maximum Likelihood and maximum a posteriori

Maximum Likelihood

Following the principle of empirical risk minimization, let $\mathcal{L}(\theta)$ denote a loss function defined over the model parameters by:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \ell\Big(y_i, f(\mathbf{x}_i; \theta)\Big).$$

For both classification and regression, we can interpret $f(\mathbf{x}; \theta)$ as defining a model of the posterior distribution $p(y|\mathbf{x}; \theta)$.

Therefore, we can define the loss $\ell(\cdot, \cdot)$ as the negative log-likelihood (NLL):

$$\begin{aligned} \ell(y, f(\mathbf{x}; \theta)) &= -\ln p(\mathbf{x}, y; \theta) \\ &= -\ln p(y|\mathbf{x}; \theta) - \ln p(\mathbf{x}) \\ &= -\ln p(y|\mathbf{x}; \theta) + cst(\theta) \end{aligned}$$

Maximum a posteriori

We could also treat θ as a random variable and define the loss $\ell(\cdot, \cdot)$ as the negative log-posterior:

$$\begin{aligned} \ell(y, f(\mathbf{x}; \theta)) &= -\ln p(\theta|\mathbf{x}, y) \\ &= -\ln p(\mathbf{x}, y|\theta) - \ln p(\theta) + cst(\theta) \\ &= -\ln p(y|\mathbf{x}; \theta) - \ln p(\theta) + cst(\theta) \end{aligned}$$

It results in the negative log-likelihood plus a regularization term over θ which is the negative prior.

Binary classification

- Training data: $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$.
- Model: $p(y = 1|\mathbf{x}; \theta) = f(\mathbf{x}; \theta)$ and $p(y = 0|\mathbf{x}; \theta) = 1 - f(\mathbf{x}; \theta)$.

  It can be compactly rewritten as follows for all $y \in \mathcal{Y}$:

$$p(y|\mathbf{x}; \theta) = \left(f(\mathbf{x}; \theta)\right)^y \left(1 - f(\mathbf{x}; \theta)\right)^{(1-y)}.$$

- Constraint: $f(\mathbf{x}; \theta) \in [0, 1]$.
- The NLL gives the binary cross-entropy loss:

$$\begin{aligned} \ell(y, f(\mathbf{x}; \theta)) &= -\ln p(\mathbf{x}, y; \theta) \\ &= -\ln p(y|\mathbf{x}; \theta) - \ln p(\mathbf{x}) \\ &= -y \ln \left(f(\mathbf{x}; \theta)\right) - (1 - y) \ln \left(1 - f(\mathbf{x}; \theta)\right) + cst(\theta) \end{aligned}$$

C-class classification

- Training data: $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{1, \ldots, C\}$.
- Model: $p(y = c|\mathbf{x}; \theta) = f_c(\mathbf{x}; \theta)$ for all $c \in \{1, \ldots, C\}$.

  It can be compactly rewritten as follows for all $y \in \mathcal{Y}$:

$$p(y|\mathbf{x}; \theta) = \prod_{c=1}^{C} p(y = c|\mathbf{x}; \theta)^{\mathbf{1}_{y=c}} = \prod_{c=1}^{C} f_c(\mathbf{x}; \theta)^{\mathbf{1}_{y=c}}.$$

- Constraint: $f(\mathbf{x}; \theta) \in [0, 1]^C$ and $\sum_{c=1}^{C} f_c(\mathbf{x}; \theta) = 1$ where $f_c(\mathbf{x}; \theta)$ is the c-th entry of $f(\mathbf{x}; \theta)$.
- The NLL gives the cross-entropy loss:

$$\ell(y, f(\mathbf{x}; \theta)) = -\sum_{c=1}^{C} \mathbf{1}_{y=c} \ln\Big(f_c(\mathbf{x}; \theta)\Big) + cst(\theta).$$

Regression

- 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: $p(\mathbf{y}|\mathbf{x};\theta) = \mathcal{N}\Big(\mathbf{y}; f(\mathbf{x};\theta), \mathbf{I}\Big) = (2\pi)^{-q/2} \exp\Big(-\frac{1}{2} \parallel \mathbf{y} - f(\mathbf{x};\theta) \parallel_2^2\Big)$.
- Constraint: $f(\mathbf{x};\theta) \in \mathbb{R}^q$.
- The NLL gives the squared error loss:

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

Lab session on multinomial logistic regression