Deep Learning

Lecture 1: Fundamentals of machine learning

Gilles Louppe
g.louppe@uliege.be

LIÈGE université

Course outline

Theory:

- Lecture 1: Fundamentals of machine learning
- Lecture 2: Neural networks
- Lecture 3: Convolutional networks
- Lecture 4: Variational auto-encoders
- Lecture 5: Generative adversarial networks
- Lecture 6: Adversarial attacks and defenses

Practice:

- Building and training neural networks with PyTorch

Outline

Goal: Set the fundamentals of machine learning.

- Why learning?
- Learning from data
- Empirical risk minimization
- Bias-variance dilemma

Why learning?

The automatic extraction of semantic information from raw signal is at the core of many applications (e.g., object recognition, speech processing, natural language processing, planning, etc).

Can we write a computer program that does that?

(ImageNet)

The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.

This is a mushroom.

In [1]:
```
from matplotlib.pyplot import imread
imread("mushroom-small.png")
```

Out[1]:
```
array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
        [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
        [0.4117647 , 0.34117648, 0.37254903, 1.        ],
        ...,
        [0.20392157, 0.23529412, 0.17254902, 1.        ],
        [0.16470589, 0.18039216, 0.12156863, 1.        ],
        [0.18039216, 0.18039216, 0.14117648, 1.        ]],

       [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
        [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
        [0.21176471, 0.2       , 0.20392157, 1.        ],
        ...,
        [0.1764706 , 0.24705882, 0.12156863, 1.        ],
        [0.10980392, 0.15686275, 0.07843138, 1.        ],
        [0.16470589, 0.20784314, 0.11764706, 1.        ]],

       [[0.14117648, 0.12941177, 0.10980392, 1.        ],
        [0.21176471, 0.1882353 , 0.16862746, 1.        ],
        [0.14117648, 0.13725491, 0.12941177, 1.        ],
        ...,
        [0.10980392, 0.15686275, 0.08627451, 1.        ],
        [0.0627451 , 0.08235294, 0.05098039, 1.        ],
        [0.14117648, 0.2       , 0.09803922, 1.        ]],

       ...,
```

This is a mushroom.

Extracting semantic information requires models of high complexity. Therefore one cannot write by hand a computer program that reproduces this process.

However, one can write a program that learns the task of extracting semantic information. A common strategy to solve this issue consists in:

- defining a parametric model with high capacity,
- optimizing its parameters, by "making it work" on the training data.

Learning tuning the many parameters of a model.

Learning from data

Data generative model

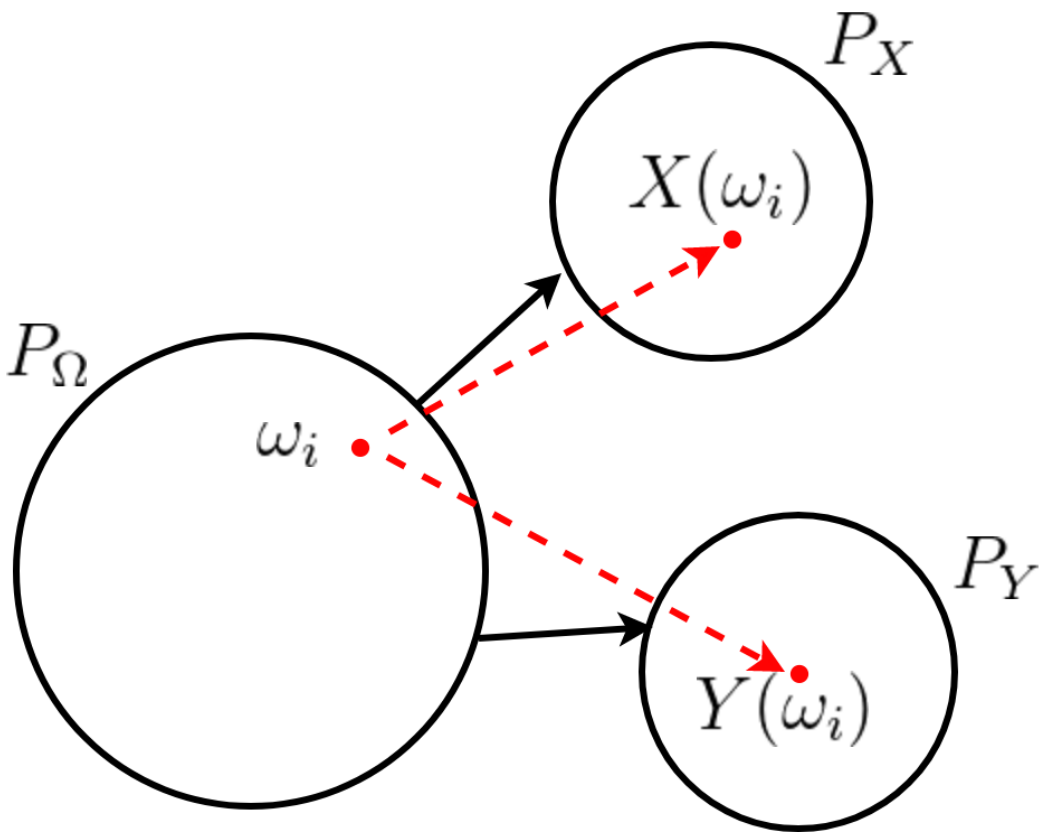
Consider an unknown joint probability distribution $(\quad , \quad)$ over observations or values of interest.

Assume training data drawn from this distribution:

$$(\quad , \quad) \sim \quad (\quad , \quad),$$

with $\in \quad , \quad \in \quad , \quad = 1, \ldots, \quad .$

- In most cases,
  - is a -dimensional vector of features or descriptors,
  - is a scalar (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 .
- In general, we do not have any prior information about $(\quad , \quad)$.



Probability space and random variables interpretation of the data generative process.

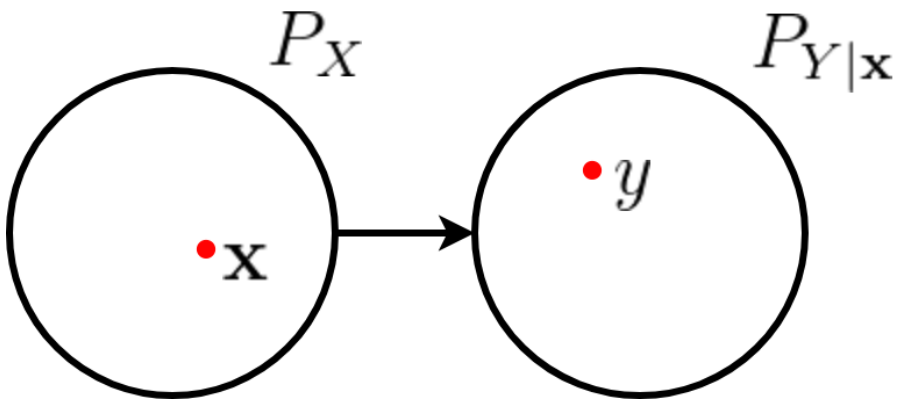
(,) ~ (,) ⇔ ~ Ω, = ( ), = ( )

Alternatively, the joint distribution can be interpreted as a two-step generative process such that

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X)$$

where

- for $P(X|Y)P(Y)$:
  - first, we draw $y \sim P(Y)$
  - then, generate $\mathbf{x} \sim P(X|Y = y)$.
- for $P(Y|X)P(X)$:
  - first, we draw $\mathbf{x} \sim P(X)$
  - then, generate $y \sim P(Y|X = \mathbf{x})$.



Two-step generative interpretation of $P(X, Y)$ as $P(X)P(Y|X)$.

e.g., $\mathbf{x} \sim P(X)$, $y = f(\mathbf{x}) + \epsilon$ for $\epsilon \sim \mathsf{N}$.

Inference

In supervised learning, we are usually interested in the two following inference problems:

- Classification:
  Given $(\mathbf{x}, y) \in \mathrm{X} \times \mathrm{Y} = \quad^{p} \times \{1, \ldots, C\}$, we want to estimate

$$\arg\max_{y} P(Y = y | X = \mathbf{x}).$$

- Regression:
  Given $(\mathbf{x}, y) \in \mathrm{X} \times \mathrm{Y} = \quad^{p} \times \quad$, we want to estimate

$$[Y | X = \mathbf{x}].$$

The boundary between these inference problems is fuzzy, as one often reduces to the other.

- Regression enables classification through class scores.
- Classification can be viewed as discretized regression.

These inference problems also closely relate to the more general (conditional) density estimation problem.

Empirical risk minimization

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

$$\ell : \mathsf{Y} \times \mathsf{Y} \to \mathsf{R}$$

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

For example,

- for classification:

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

- for regression:

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

Let us denote as F 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 $f \in \mathsf{F}$ with a small expected risk (or generalization error)

$$R(f) = \mathsf{E}_{(\mathbf{x},y)\sim P(X,Y)} \left[\ell(y, f(\mathbf{x})) \right].$$

This means that for a given data generating distribution and for a given hypothesis space, the optimal model is

$$f_* = \arg\min_{f \in \mathsf{F}} R(f).$$

Unfortunately, since $P(X, Y)$ is unknown, the expected risk cannot be evaluated and the optimal model cannot be determined.

However, given training data $\mathbf{d} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\}$, we can compute an estimate, the empirical risk (or training error)

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

This estimate can be used for finding a good enough approximation of f_*, giving rise to the empirical risk minimization principle:

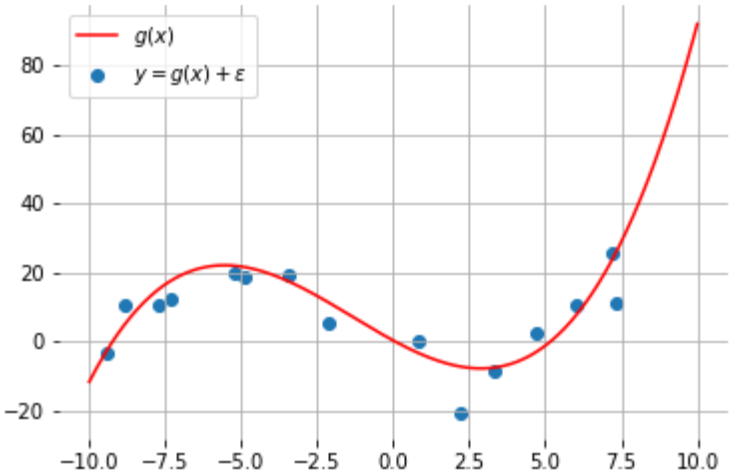
$$f_*^{\mathbf{d}} = \arg\min_{f \in \mathsf{F}} \hat{R}(f, \mathbf{d})$$

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

Under regularity assumptions, empirical risk minimizers converge:

$$\lim_{N \to \infty} f_*^{\mathbf{d}} = f_*$$

Polynomial regression



Consider the joint probability distribution $P(X, Y)$ induced by the data generating process

$$x, y \sim P(X, Y) \Leftrightarrow x \sim U[-10; 10], \epsilon \sim \mathrm{N}(0, \sigma^2), y = g(x) + \epsilon$$

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

Our goal is to find a function f that makes good predictions on average over $P(X, Y)$.

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

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

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 are the predictions.

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

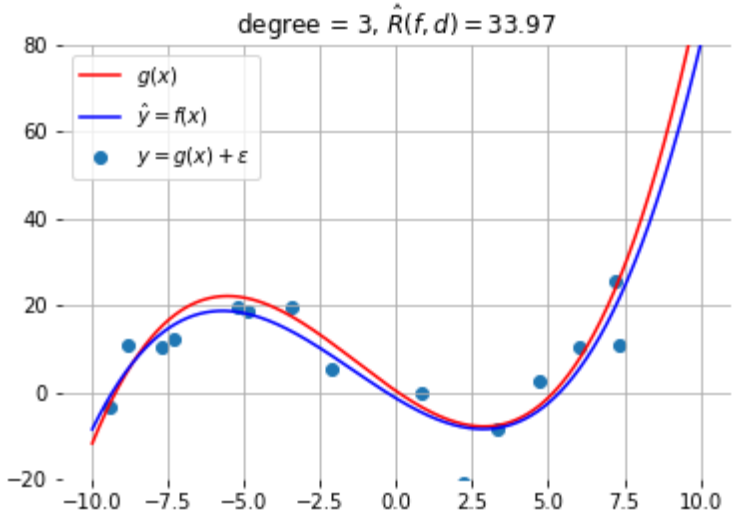
$$\begin{aligned} \mathbf{w}_* &= \arg\min_{\mathbf{w}} R(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \mathrm{E}_{(x,y)\sim P(X,Y)} \ (y - f(x; \mathbf{w}))^2 \end{aligned}$$

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

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

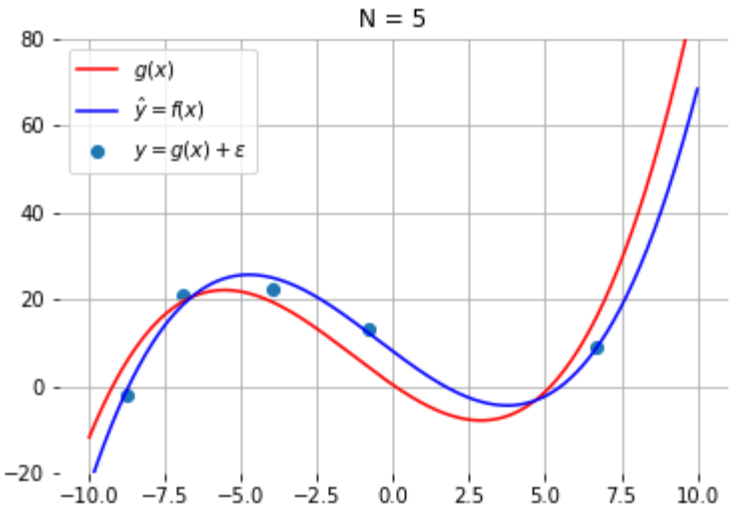
This is ordinary least squares regression, for which the solution is known analytically:

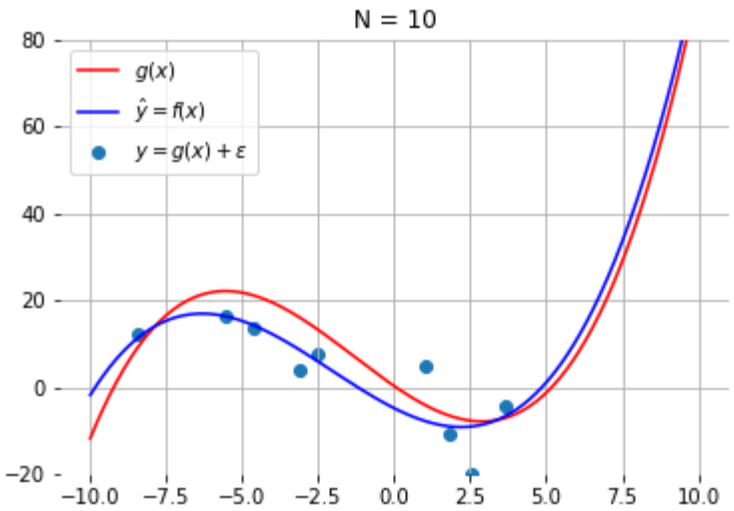
$$\mathbf{w}_*^{\mathbf{d}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

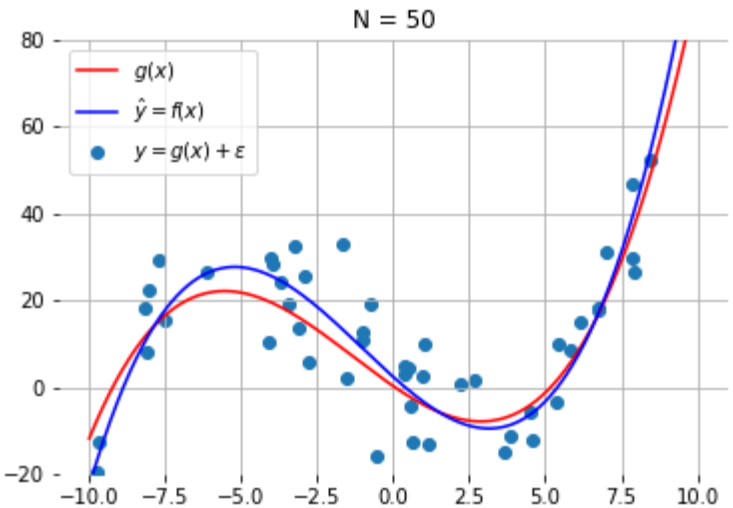


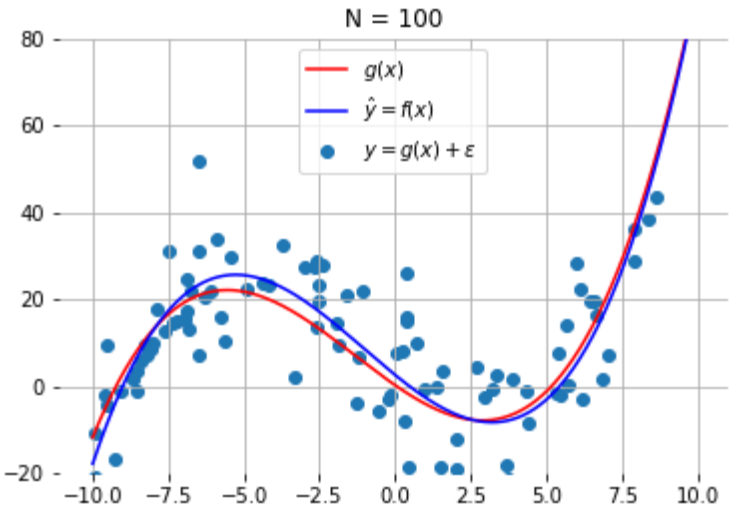
The expected risk minimizer within our hypothesis space is g itself.

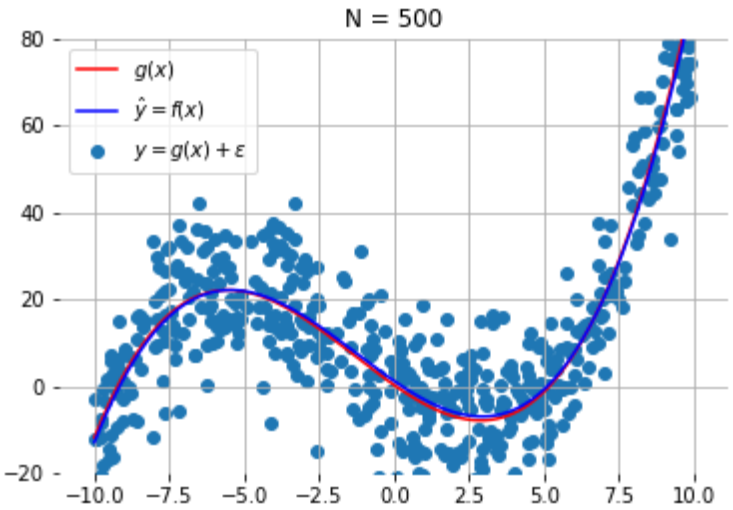
Therefore, on this toy problem, we can verify that
$f(x; \mathbf{w}_*^{\mathbf{d}}) \to f(x; \mathbf{w}_*) = g(x)$ as $N \to \infty$.


N = 5
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$


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


N = 50
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

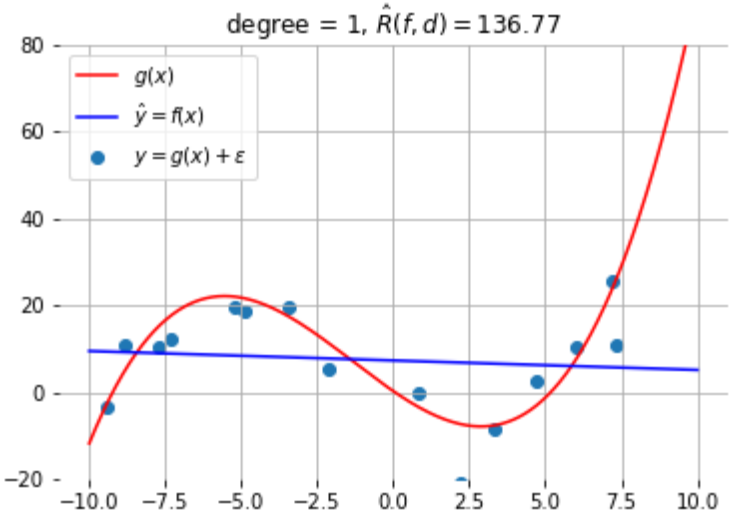

N = 100
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


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

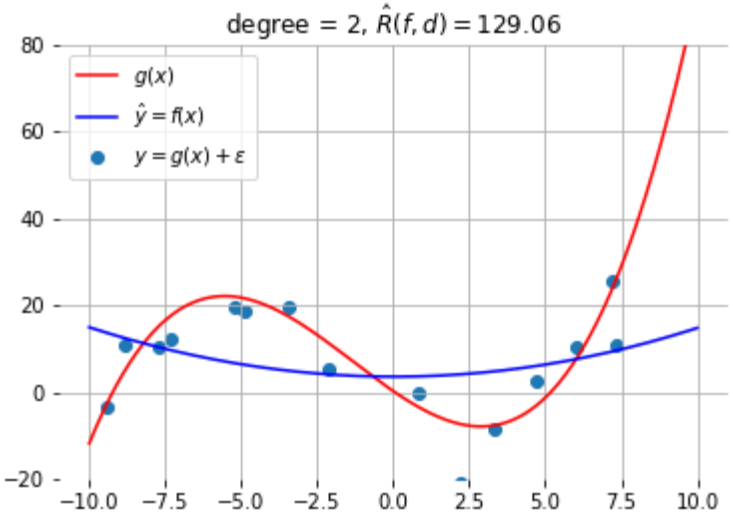
Under-fitting and over-fitting

Under-fitting and over-fitting

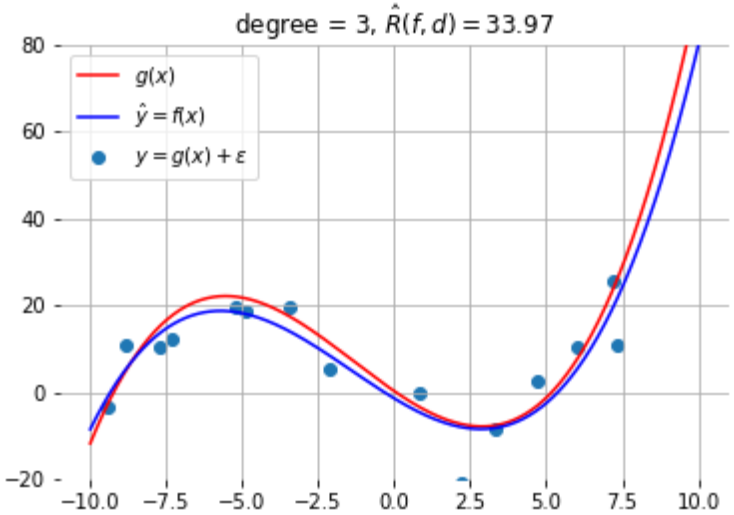
What if we consider a hypothesis space F in which candidate functions f are either too "simple" or too "complex" with respect to the true data generating process?



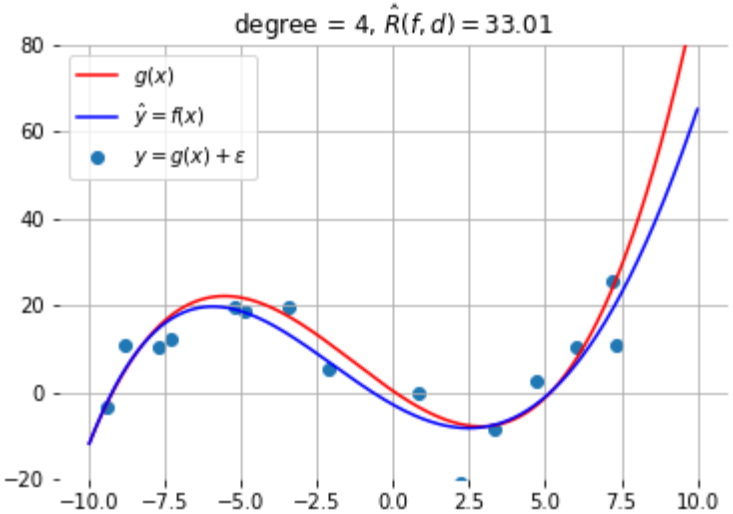
F = polynomials of degree 1



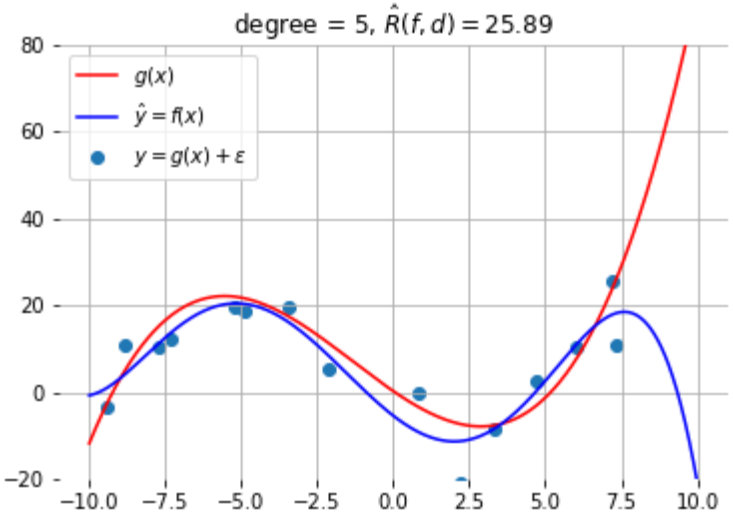
F = polynomials of degree 2



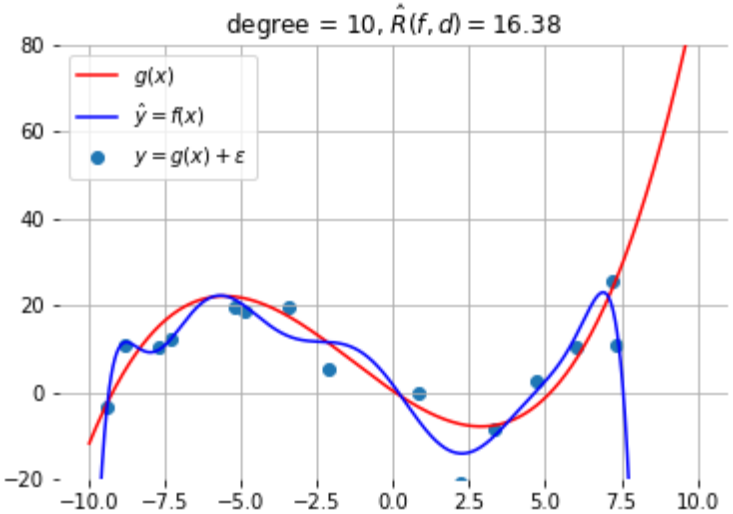
F = polynomials of degree 3



F = polynomials of degree 4



F = polynomials of degree 5



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

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.

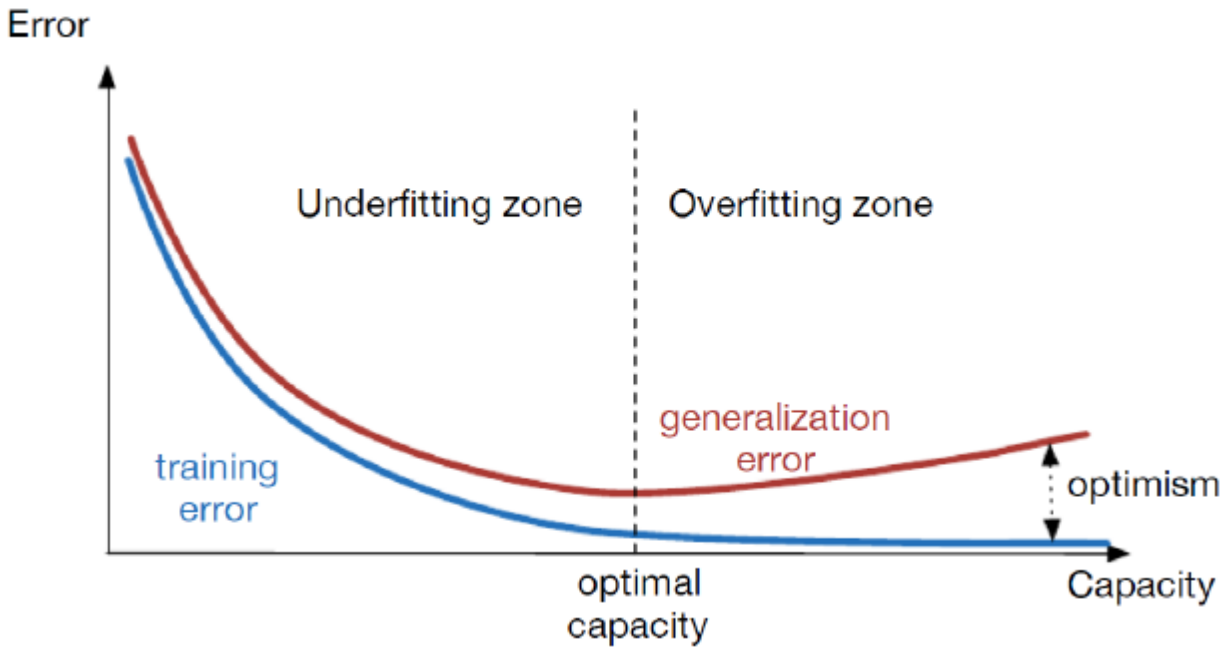
The capacity of an hypothesis space induced by a learning algorithm intuitively represents the ability to find a good model $f \in \mathcal{F}$ for any function, regardless of its complexity.

- If the capacity of $\mathcal{F}$ is low, then $f_B \notin \mathcal{F}$ and $R(f) - R_B$ is large for any $f \in \mathcal{F}$, including f_* and $f_*^{\mathbf{d}}$. Such models f are said to underfit the data.
- If the capacity of $\mathcal{F}$ is high, then $f_B \in \mathcal{F}$ or $R(f_*) - R_B$ is small. However, because of the high capacity of the hypothesis space, the empirical risk minimizer $f_*^{\mathbf{d}}$ could fit the training data arbitrarily well such that

$$R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0.$$

  In this situation, $f_*^{\mathbf{d}}$ becomes too complex with respect to the true data generating process and a large reduction of the empirical risk (often) comes at the price of an increase of the expected risk of the empirical risk minimizer $R(f_*^{\mathbf{d}})$. In this situation, $f_*^{\mathbf{d}}$ is said to overfit the data.

Therefore, our goal is to adjust the capacity of the hypothesis space such that the expected risk of the empirical risk minimizer gets as low as possible.



In practice, the capacity of the hypothesis space can be controlled through hyper-parameters of the learning algorithm. For example:

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

Bias-variance decomposition

Consider a fixed point x and the prediction $\hat{Y} = f_*^{\mathbf{d}}(x)$ of the empirical risk minimizer at x.

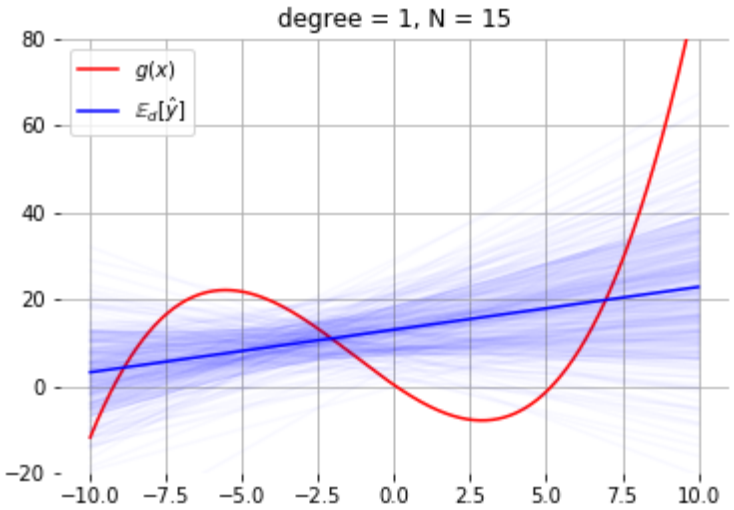
Then the local expected risk of $f_*^{\mathbf{d}}$ is

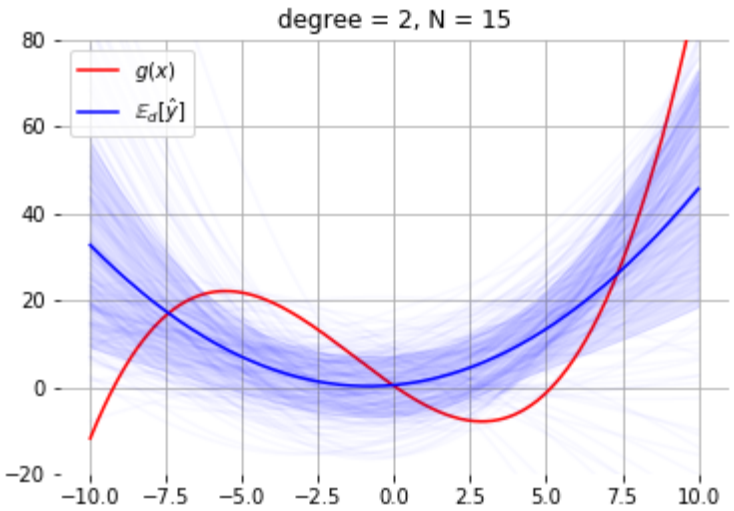
$$\begin{aligned} R(f_*^{\mathbf{d}}|x) &= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_*^{\mathbf{d}}(x))^2\right] \\ &= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_B(x) + f_B(x) - f_*^{\mathbf{d}}(x))^2\right] \\ &= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_B(x))^2\right] + \mathbb{E}_{y\sim P(Y|x)}\left[(f_B(x) - f_*^{\mathbf{d}}(x))^2\right] \\ &= R(f_B|x) + (f_B(x) - f_*^{\mathbf{d}}(x))^2 \end{aligned}$$

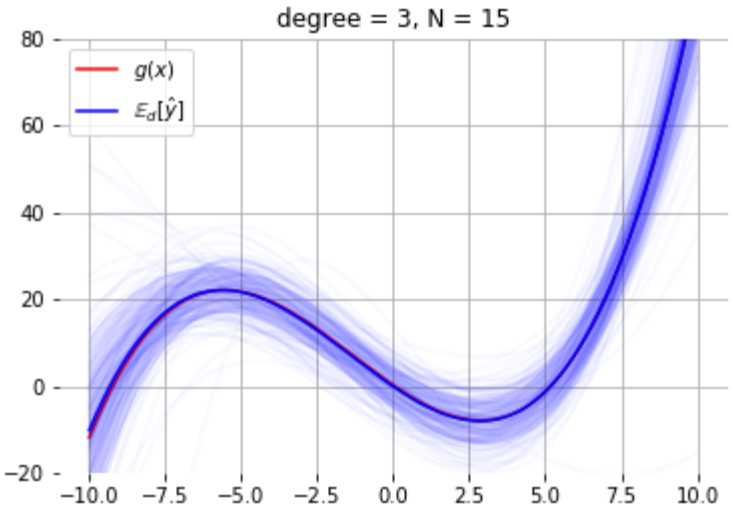
where

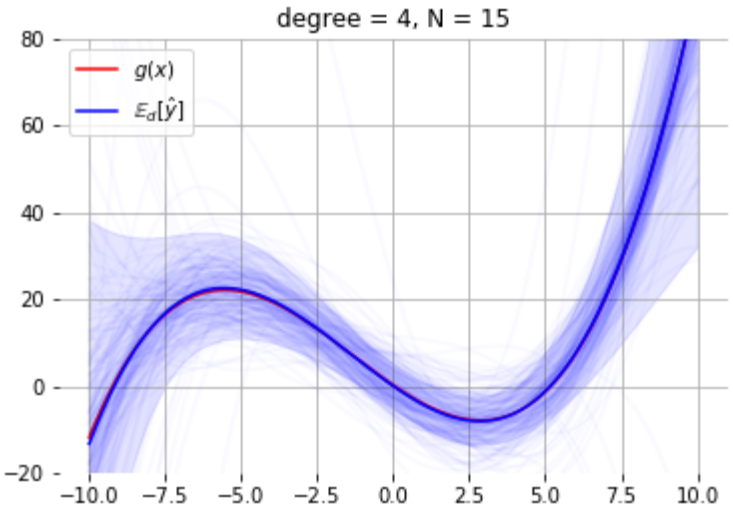
- $R(f_B|x)$ is the local expected risk of the Bayes model. This term cannot be reduced.
- $(f_B(x) - f_*^{\mathbf{d}}(x))^2$ represents the discrepancy between f_B and $f_*^{\mathbf{d}}$.

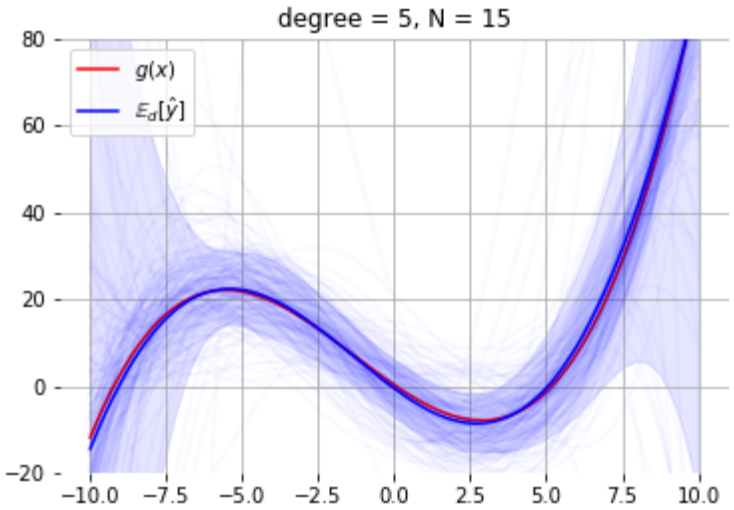
If $\mathbf{d} \sim P(X, Y)$ is itself considered as a random variable, then $f_*^{\mathbf{d}}$ is also a random variable, along with its predictions $\hat{Y}$.


degree = 1, N = 15
$g(x)$
$\mathbb{E}_{\mathcal{D}}[\hat{y}]$
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
$g(x)$
$\mathbb{E}_{\mathcal{D}}[\hat{y}]$
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
$g(x)$
$\mathbb{E}_{\mathcal{D}}[\hat{y}]$
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
$g(x)$
$\mathbb{E}_{\mathcal{D}}[\hat{y}]$
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
$g(x)$
$\mathbb{E}_{\mathcal{D}}[\hat{y}]$
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:

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

This decomposition is known as the bias-variance decomposition.

- The noise term quantities 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.

Typically,

- Models of low capacity have low variance but high bias.
- Models of high capacity how high variance but low bias.

The end.

References

- EE-559 Deep learning (Francois Fleuret, EPFL)
- Understanding Random Forests: From Theory to Practice (Louppe, 2014)