Deep Learning

Lecture 1: Fundamentals of machine learning

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université

Outline

Goal: Set the fundamentals of machine learning.

- Why learning?
- Applications and success
- Statistical learning
  - Data
  - Empirical risk minimization
  - Under-fitting and over-fitting
  - Bias-variance dilemma

Why learning?

What do you see? How do we do that?!

Sheepdog or mop?

Chihuahua or muffin?

The automatic extraction of semantic information from raw signal is at the core of many applications, such as

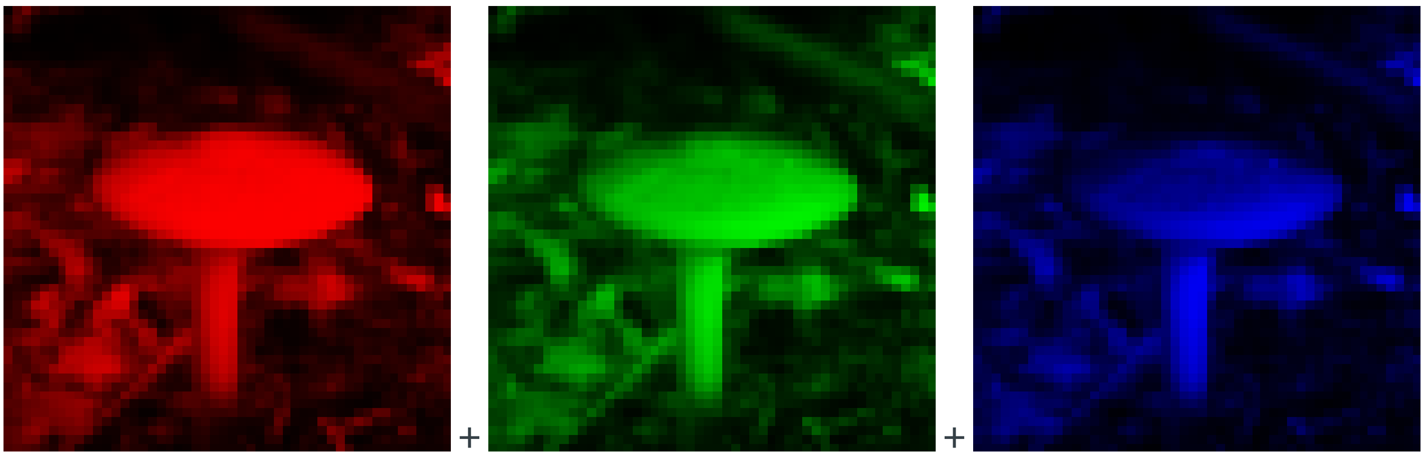
- image recognition
- speech processing
- natural language processing
- robotic control
- ... and many others.

How can we write a computer program that implements this mechanism?

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.



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, which cannot be designed by hand.

However, one can write a program that learns the task of extracting semantic information.

Techniques used in practice consist of:

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

This is similar to biological systems for which the model (e.g. brain structure) is DNA-encoded, and parameters (e.g. synaptic weights) are tuned through experiences.

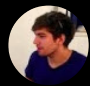
Deep learning encompasses software technologies to scale-up to billions of model parameters and as many training examples.

Applications and success

Segmentation (Hengshuang et al, 2017)

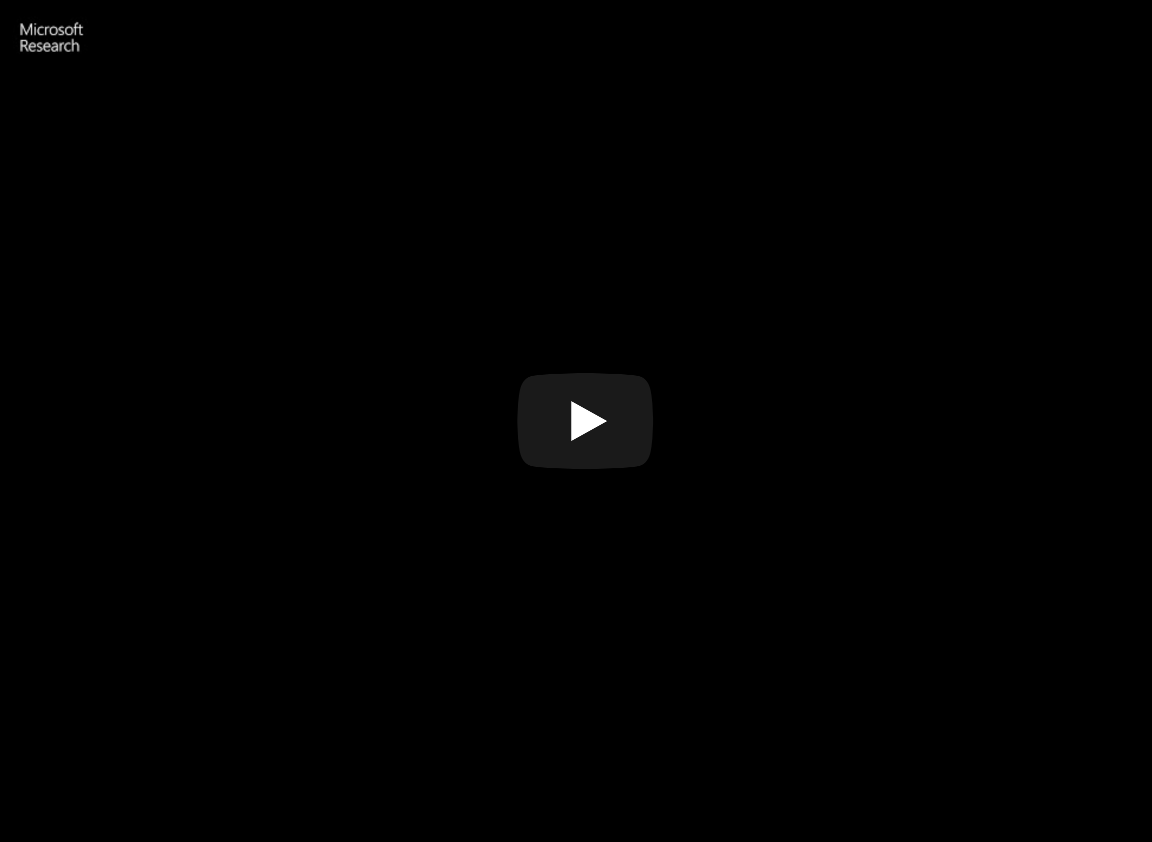
Pose estimation (Cao et al, 2017)

Reinforcement learning (Mnih et al, 2014)



Strategy games (Deepmind, 2016-2018)

Autonomous cars (NVIDIA, 2016)



Speech recognition, translation and synthesis (Microsoft, 2012)

Auto-captioning (2015)

Speech synthesis and question answering (Google, 2018)

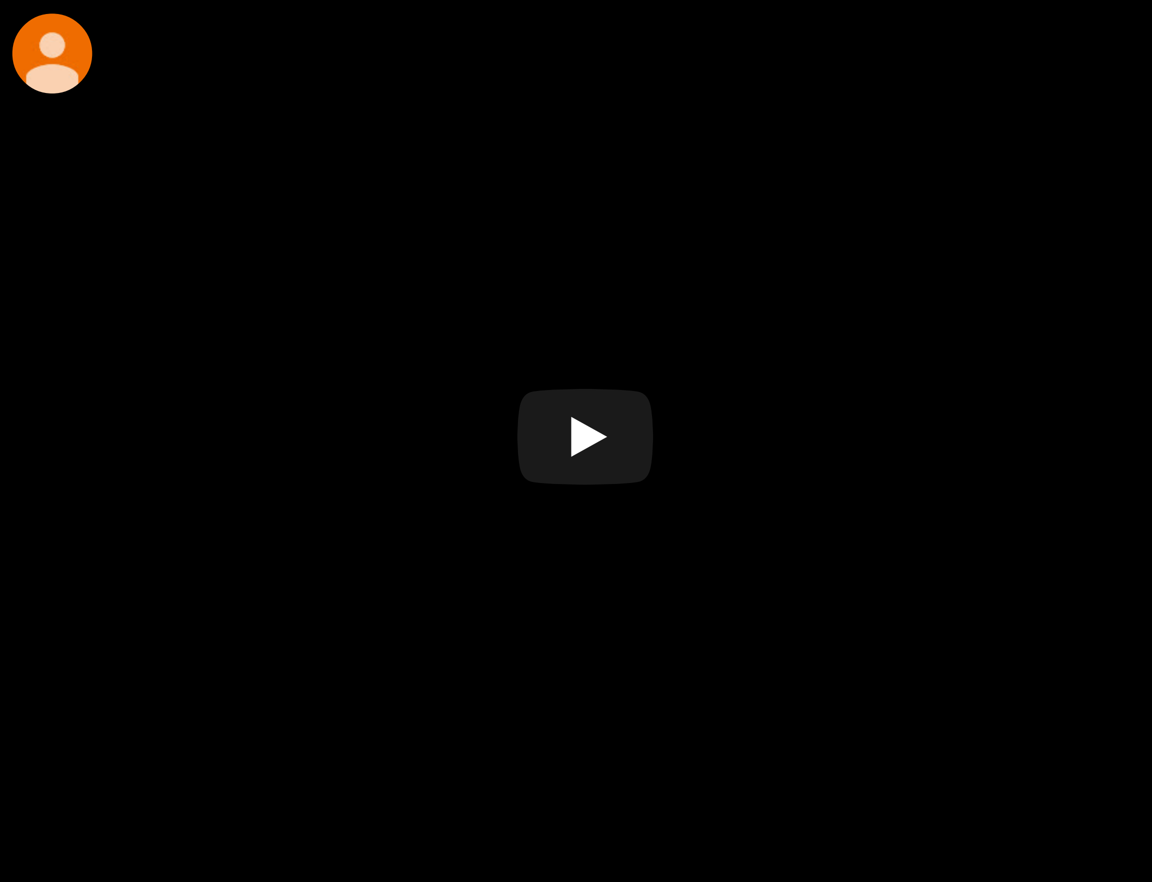
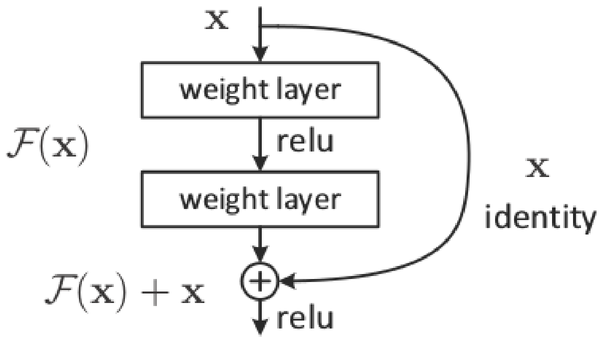


Image generation (Karras et al, 2018)

Music composition (NVIDIA, 2017)

Why does it work now?

Algorithms



Data

Software

Compute engines

Five decades of research in machine learning provided

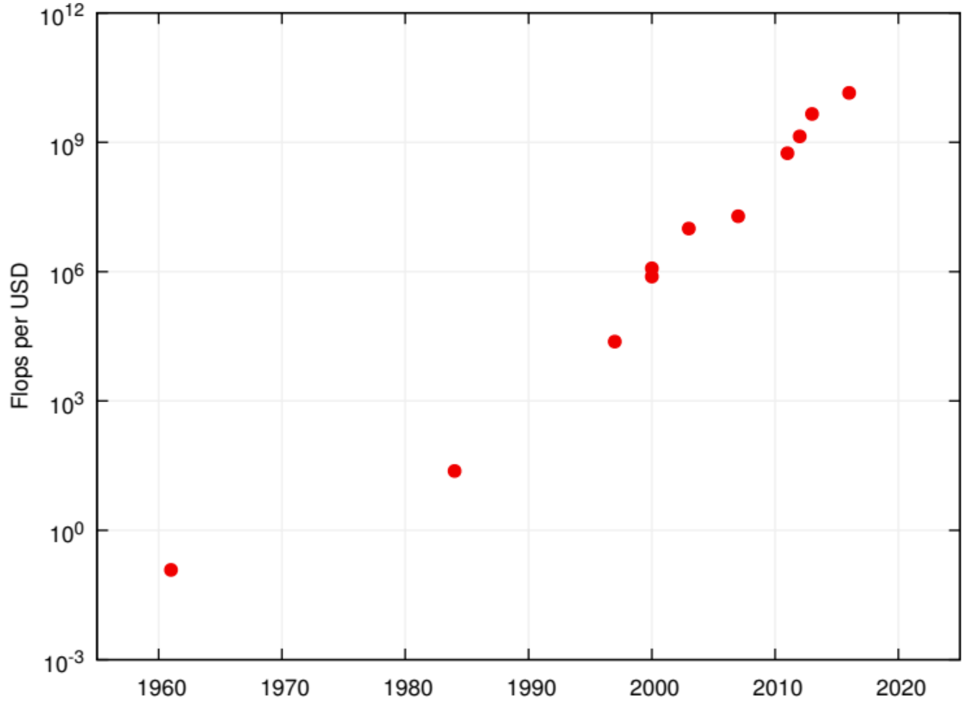
- a taxonomy of ML concepts (classification, generative models, clustering, kernels, linear embeddings, etc.),
- a sound statistical formalization (Bayesian estimation, PAC),
- a clear picture of fundamental issues (bias/variance dilemma, VC dimension, generalization bounds, etc.),
- a good understanding of optimization issues,
- efficient large-scale algorithms.

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

From a practical perspective, deep learning

- lessens the need for a deep mathematical grasp,
- makes the design of large learning architectures a system/software development task,
- allows to leverage modern hardware (clusters of GPUs),
- does not plateau when using more data,
- makes large trained networks a commodity.

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.



	TFlops (10^{12})	Price	GFlops per $
Intel i7-6700K	0.2	$344	0.6
AMD Radeon R-7 240	0.5	$55	9.1
NVIDIA GTX 750 Ti	1.3	$105	12.3
AMD RX 480	5.2	$239	21.6
NVIDIA GTX 1080	8.9	$699	12.7

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

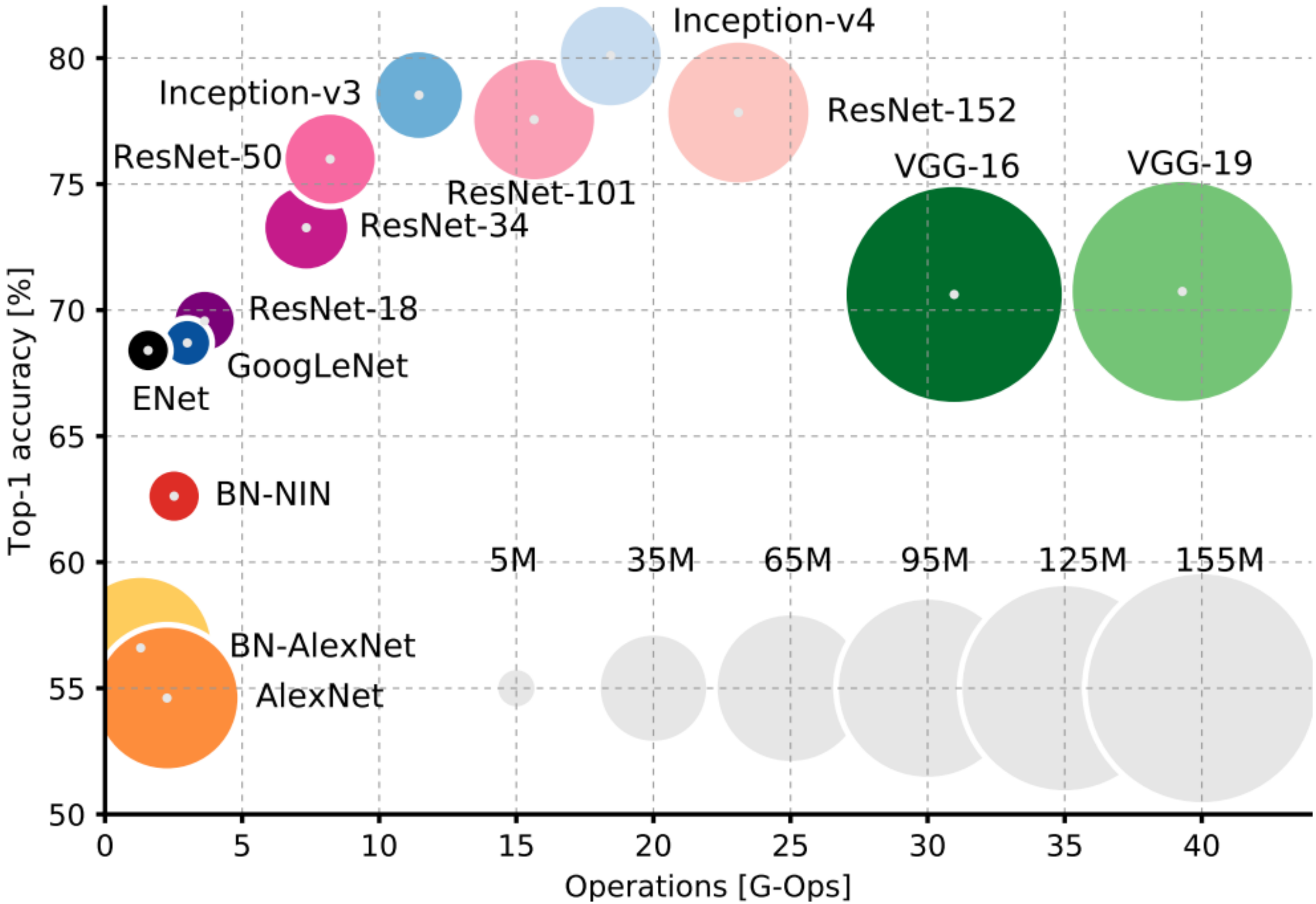


Image credits: Canziani et al, 2016, arXiv:1605.07678.

Statistical learning

Statistical learning

Consider an unknown joint probability distribution $P(X, Y)$.

Assume training data

$$(\mathbf{x}_i, y_i) \sim P(X, Y),$$

with $\mathbf{x}_i \in \mathcal{X}, y \in \mathcal{Y}, i = 1, ..., N$.

- In most cases,
  - $\mathbf{x}_i$ is a p-dimensional vector of features or descriptors,
  - y 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 N.
- In general, we do not have any prior information about $P(X, Y)$.

Inference

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

- Classification: Given $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, ..., C\}$, we want to estimate

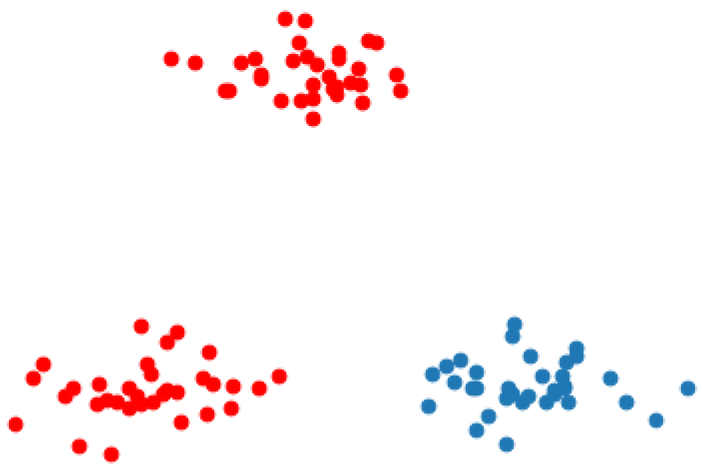
$$\arg\max_y P(Y = y | X = \mathbf{x}).$$

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

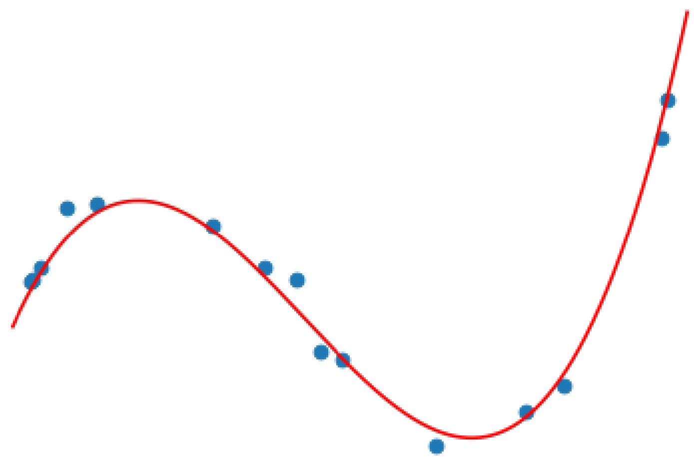
$$\mathbb{E}\left[Y | X = \mathbf{x}\right].$$

Or more generally, we want to estimate

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



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



Regression aims at estimating relationships among 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 is the prediction $f(\mathbf{x})$ from y.

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)

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

This means that for a given data generating distribution $P(X, Y)$ and for a given hypothesis space $\mathcal{F}$, the optimal model is

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

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

However, if we have i.i.d. training data $\mathbf{d} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\}$, 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 is unbiased and can be used for finding a good enough approximation of f_*. This results into the empirical risk minimization principle:

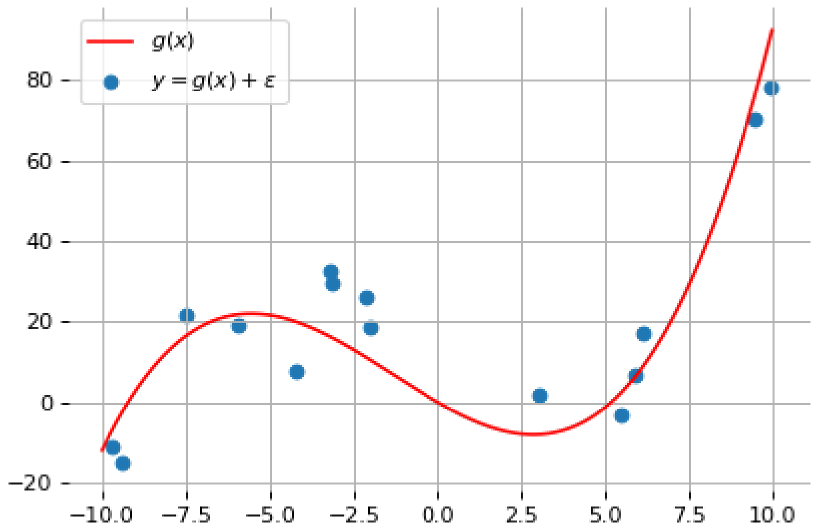
$$f_*^{\mathbf{d}} = \arg\min_{f \in \mathcal{F}} \hat{R}(f, \mathbf{d})$$

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

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

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

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

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

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

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

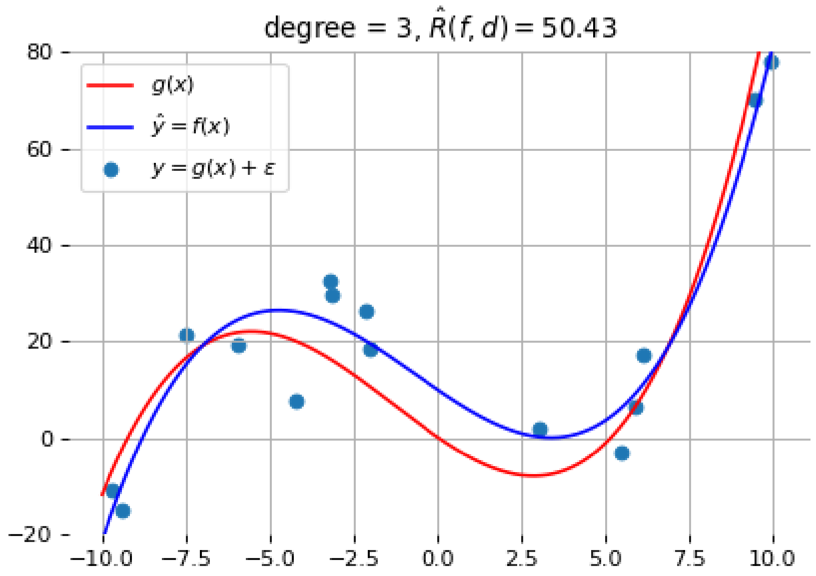
$$\begin{aligned} \mathbf{w}_* &= \arg\min_{\mathbf{w}} R(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \mathbb{E}_{(x,y)\sim P(X,Y)} \left[(y - f(x; \mathbf{w}))^2 \right] \end{aligned}$$

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:

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

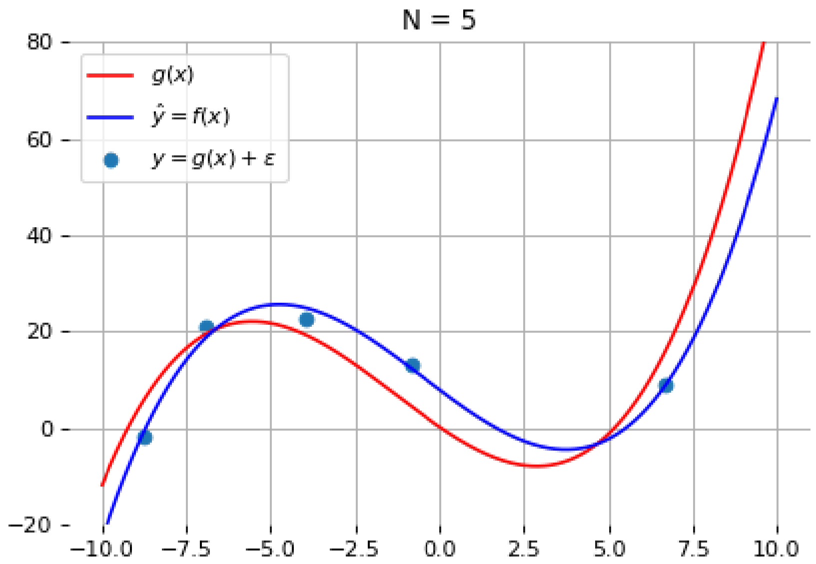
This is ordinary least squares regression, for which the solution is known analytically:

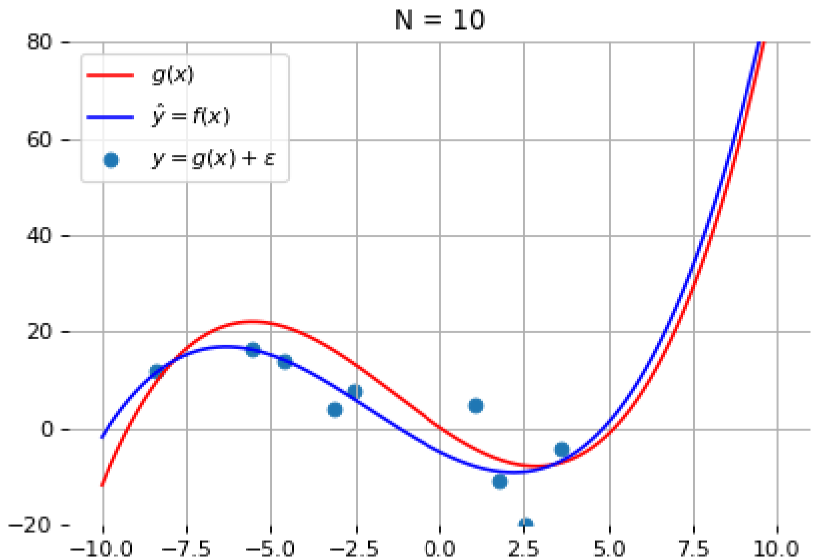
$$\mathbf{w}_*^{\mathbf{d}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

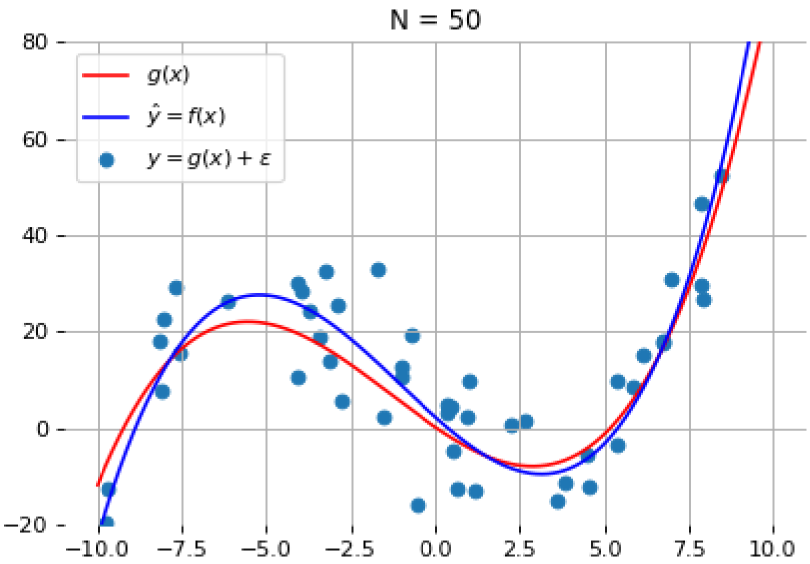


The expected risk minimizer within our hypothesis space is g itself.

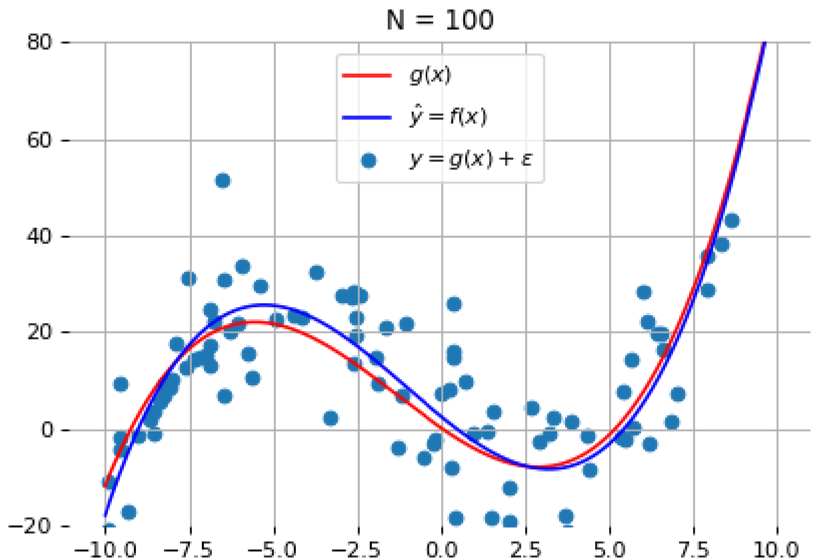
Therefore, on this toy problem, we can verify that $f(x; \mathbf{w}_*^{\mathbf{d}}) \rightarrow f(x; \mathbf{w}_*) = g(x)$ as $N \rightarrow \infty$.

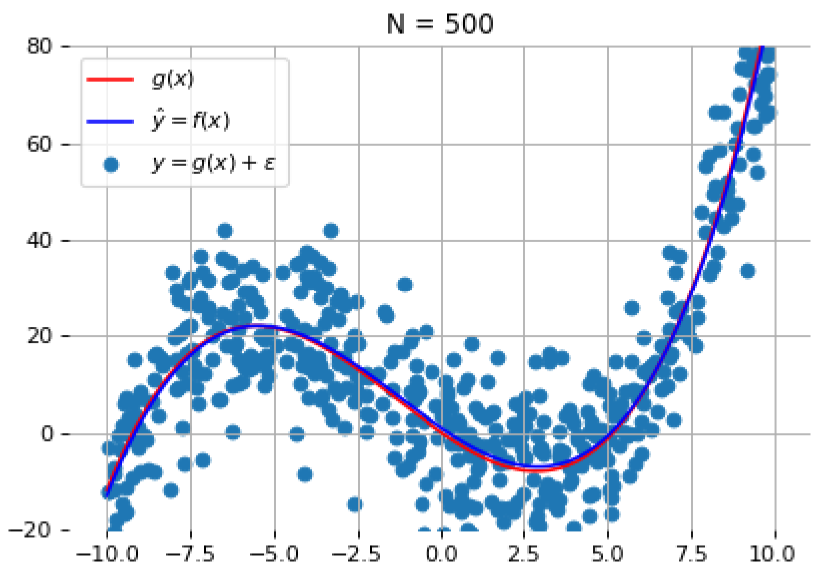

N = 5
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


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



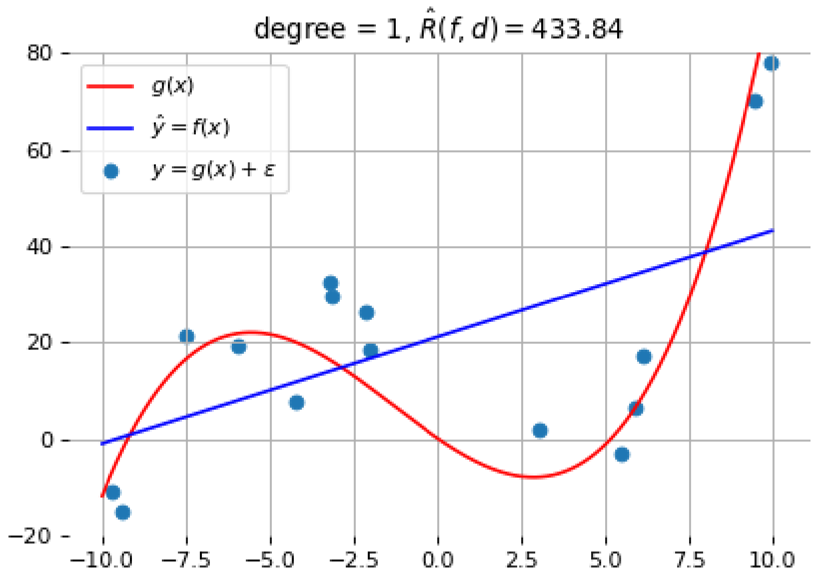
N = 50
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$


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

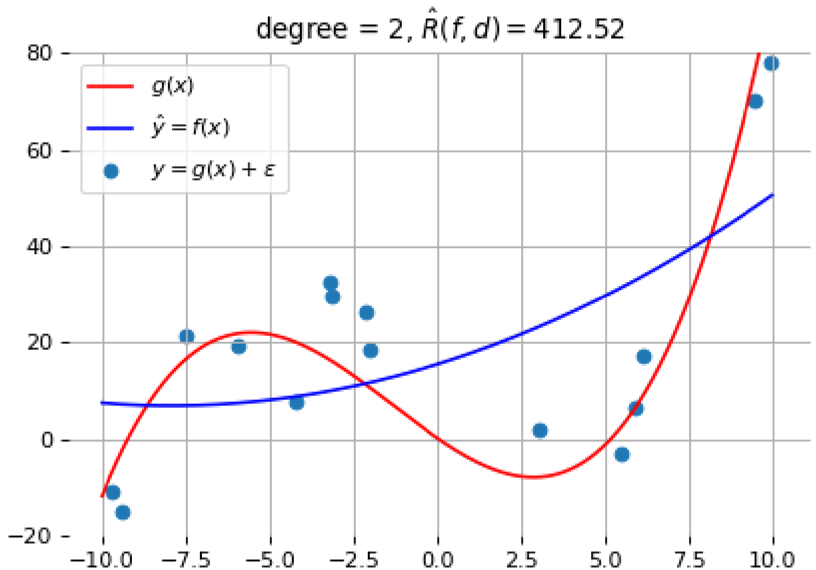

N = 500
80
60
40
20
0
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
$g(x)$
$\hat{y} = f(x)$
$y = g(x) + \varepsilon$

Under-fitting and over-fitting

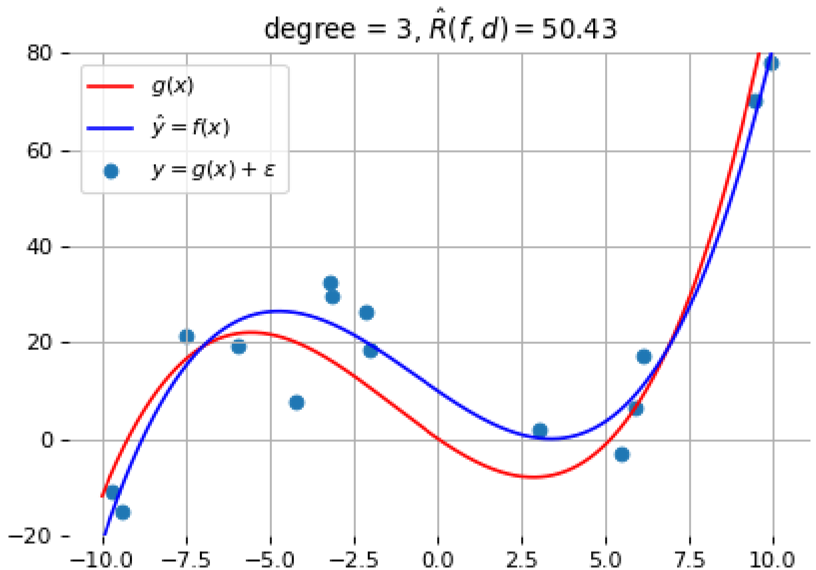
What if we consider a hypothesis space $\mathcal{F}$ in which candidate functions f are either too "simple" or too "complex" with respect to the true data generating process?



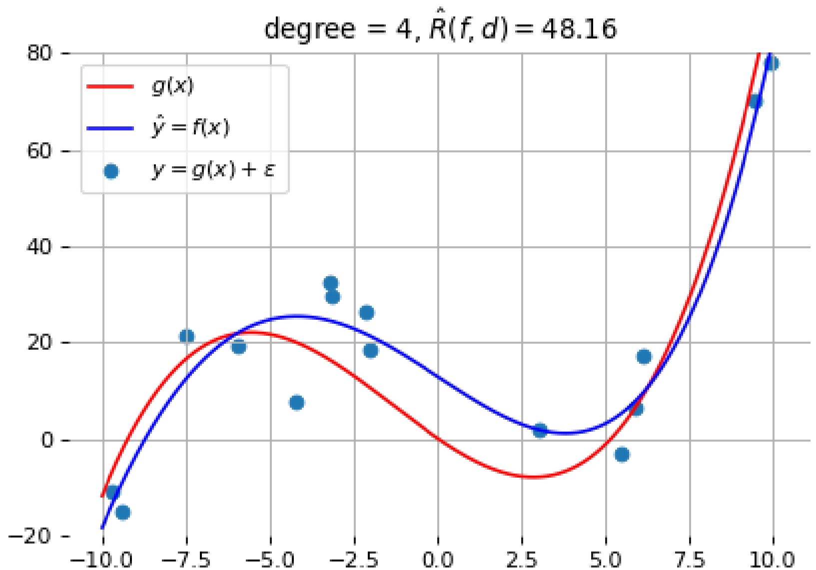
$\mathcal{F}$ = polynomials of degree 1



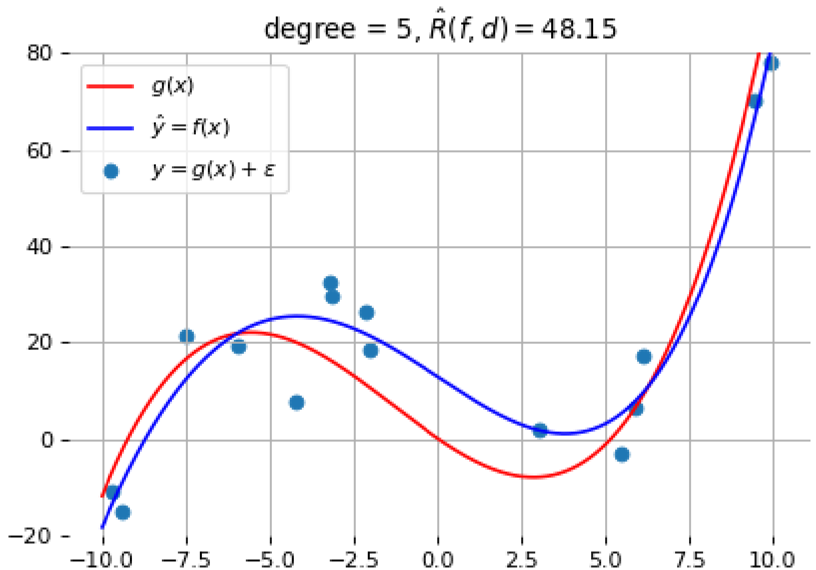
$\mathcal{F}$ = polynomials of degree 2



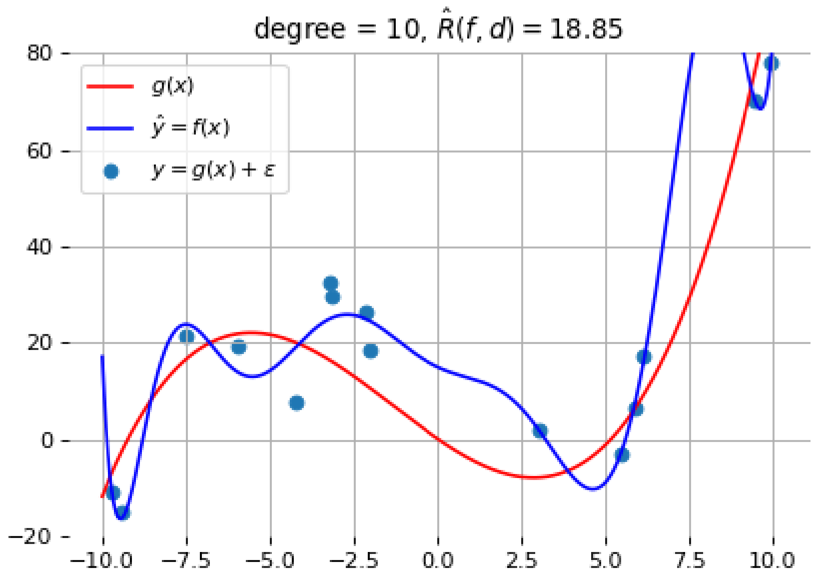
$\mathcal{F}$ = polynomials of degree 3



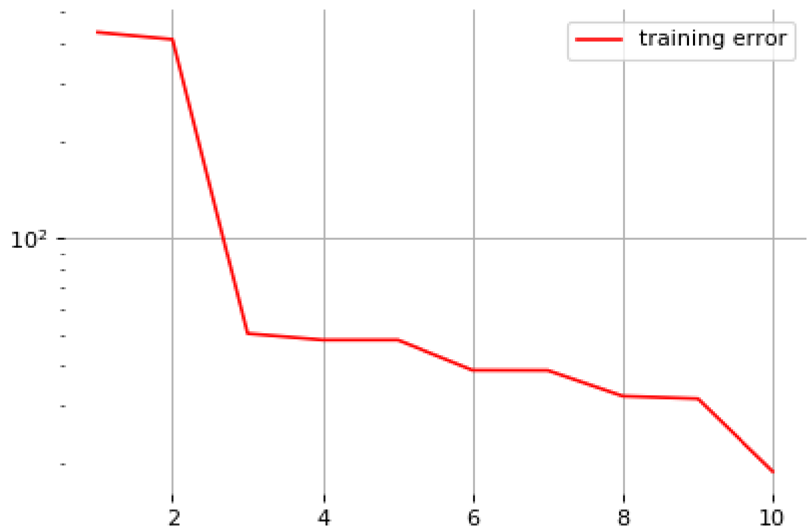
$\mathcal{F}$ = polynomials of degree 4



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



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



Degree of the polynomials VS. error.

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.

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

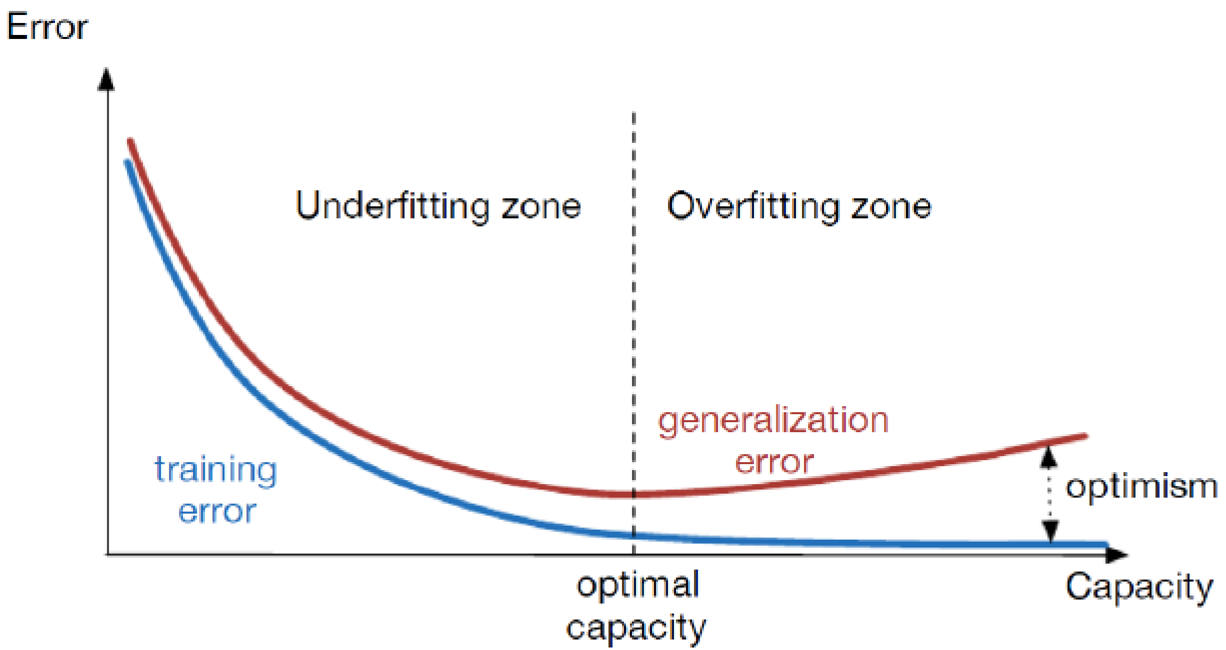
- The degree of polynomials;
- The number of layers in a neural network;
- The number of training iterations;
- Regularization terms.

- 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

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

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.



When overfitting,

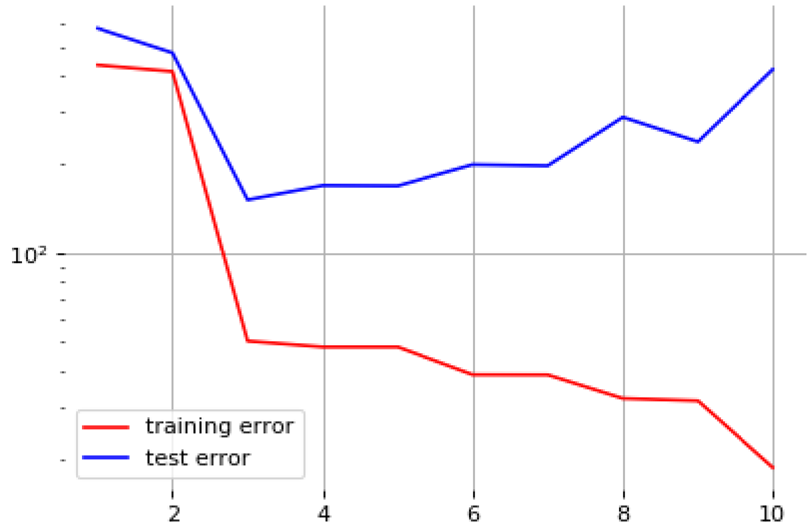
$$R(f_*^{\mathbf{d}}) \geq R_B \geq \hat{R}(f_*^{\mathbf{d}}, \mathbf{d}) \geq 0.$$

This indicates that the empirical risk $\hat{R}(f_*^{\mathbf{d}}, \mathbf{d})$ is a poor estimator of the expected risk $R(f_*^{\mathbf{d}})$.

Nevertheless, an unbiased estimate of the expected risk can be obtained by evaluating $f_*^{\mathbf{d}}$ on data $\mathbf{d}_{\text{test}}$ independent from the training samples $\mathbf{d}$:

$$\hat{R}(f_*^{\mathbf{d}}, \mathbf{d}_{\text{test}}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{d}_{\text{test}}} \ell(y_i, f_*^{\mathbf{d}}(\mathbf{x}_i))$$

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



Degree of the polynomials VS. error.

Bias-variance decomposition

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

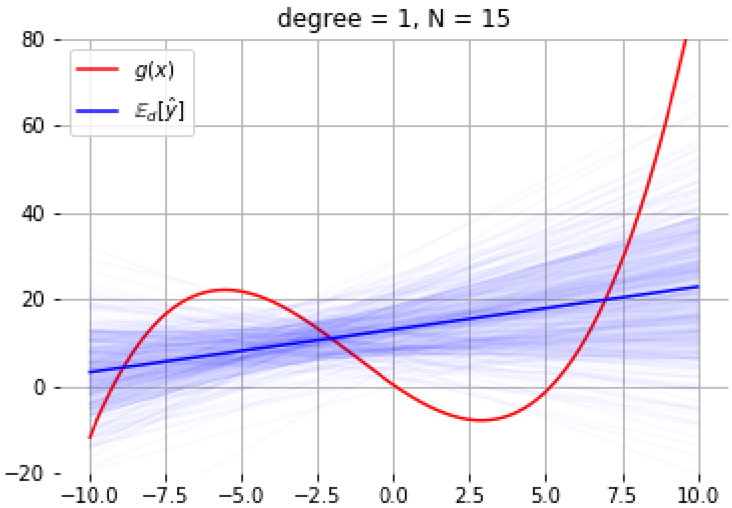
Then the local expected risk of $f_*^{\mathbf{d}}$ is

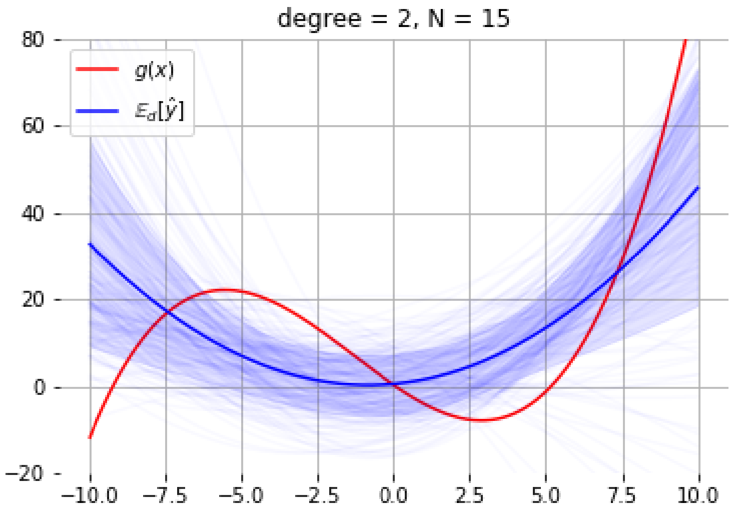
$$\begin{aligned}
R(f_*^{\mathbf{d}}|x) &= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_*^{\mathbf{d}}(x))^2\right] \\
&= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_B(x) + f_B(x) - f_*^{\mathbf{d}}(x))^2\right] \\
&= \mathbb{E}_{y\sim P(Y|x)}\left[(y - f_B(x))^2\right] + \mathbb{E}_{y\sim P(Y|x)}\left[(f_B(x) -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

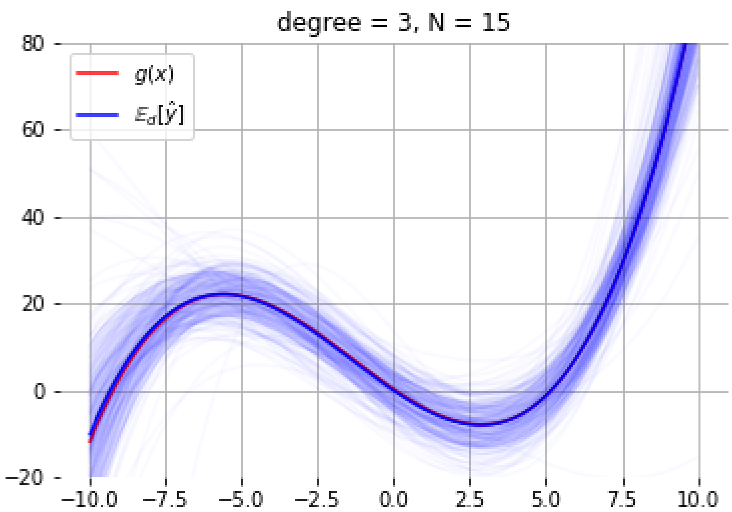
where

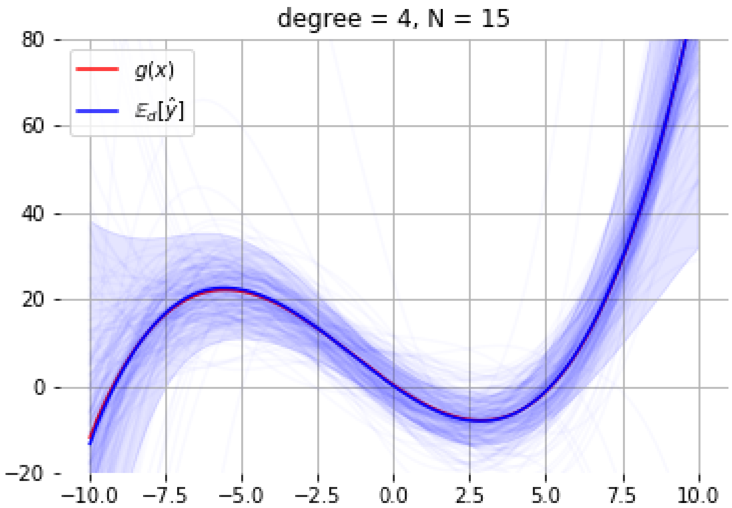
- $R(f_B|x)$ is the local expected risk of the Bayes model. This term cannot be reduced.
- $(f_B(x) - f_*^{\mathbf{d}}(x))^2$ represents the discrepancy between f_B and $f_*^{\mathbf{d}}$.

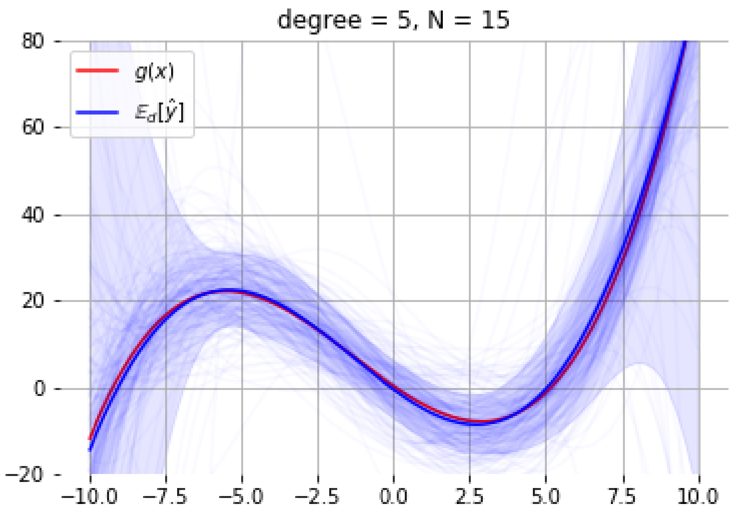
If $\mathbf{d} \sim P(X, Y)$ is itself considered as a random variable, then $f_*^{\mathbf{d}}$ is also a random variable, along with its predictions $\hat{Y}$.


degree = 1, N = 15
$g(x)$
$\mathbb{E}_d[\hat{y}]$
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
$g(x)$
$\mathbb{E}_d[\hat{y}]$
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
$g(x)$
$\mathbb{E}_d[\hat{y}]$
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
$g(x)$
$\mathbb{E}_d[\hat{y}]$
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
$g(x)$
$\mathbb{E}_d[\hat{y}]$
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.

Bias-variance trade-off

- Reducing the capacity makes f fit the data less on average, which increases the bias term.
- Increasing the capacity makes f vary a lot with the training data, which increases the variance term.

The end.

References

XXX