# Machine Learning
## *Theory*

Christian Rossi

Academic Year 2023-2024

**Abstract**

The course topics are:

- Introduction: basic concepts.
- Learning theory:
  - Bias/variance tradeoff. Union and Chernoff/Hoeffding bounds.
  - VC dimension. Worst case (online) learning.
  - Practical advice on how to use learning algorithms.
- Supervised learning:
  - Supervised learning setup. LMS.
  - Logistic regression. Perceptron. Exponential family.
  - Kernel methods: Radial Basis Networks, Gaussian Processes, and Support Vector Machines.
  - Model selection and feature selection.
  - Ensemble methods: Bagging, boosting.
  - Evaluating and debugging learning algorithms.
- Reinforcement learning and control:
  - MDPs. Bellman equations.
  - Value iteration and policy iteration.
  - TD, SARSA, Q-learning.
  - Value function approximation.
  - Policy search. Reinforce. POMDPs.
  - Multi-Armed Bandit.

# Contents

# CHAPTER 1

# Introduction

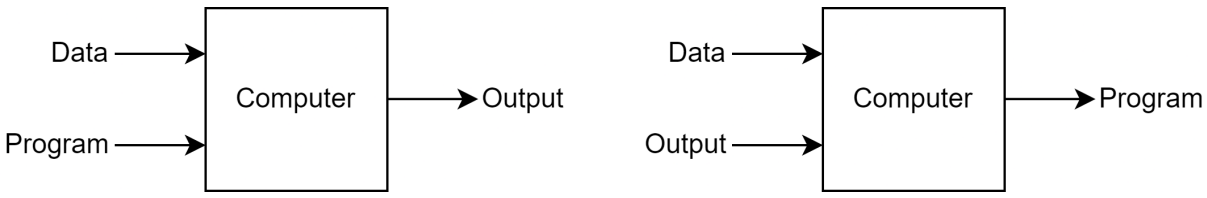
## 1.1 Machine learning

**Definition** (*Learning*)**.** A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, improves with experience $E$.

Machine learning, a subset of artificial intelligence, derives knowledge from experience and induction.

In machine learning, we depend on computers to make informed decisions using new, unfamiliar data. Designing a comprehensive set of meaningful rules can prove to be exceedingly difficult. Machine learning facilitates the automatic extraction of relevant insights from historical data and effectively applies them to new datasets.

The objective is to automate the programming process for computers, acknowledging the bottleneck presented by writing software. Instead, our aim is to utilize the data itself to accomplish the required tasks.




(a) Programming (b) Machine learning

Figure 1.1: Difference between programming and machine learning

Machine learning paradigms can be categorized into three main types:

- *Supervised learning*: involves labeled data and direct feedback, aiming to predict outcomes or future events.
- *Unsupervised learning*: operates without labeled data or feedback, focusing on discovering hidden structures within the data.
- *Reinforcement learning*: centers around a decision-making process, incorporating a reward system to learn sequences of actions.

### 1.1.1 Supervised learning

Supervised learning encompasses several distinct tasks:

- *Classification*: this involves assigning predefined categories or labels to data points based on their features. The model is trained on labeled data, learning patterns to predict the class labels of new data points.
- *Regression*: the goal here is to predict continuous numerical values based on input features, as opposed to discrete class labels in classification. The model learns a function mapping input features to output values.
- *Probability estimation*: this task predicts the likelihood of certain events or outcomes occurring, often used to gauge the confidence of model predictions.

Formally, in supervised learning, a model learns from data to map known inputs to known outputs. The training set is denoted as $\mathcal{D} = \{\langle x, t\rangle\}$, Where $t = f(x)$, with $f$ representing the unknown function to be determined using supervised learning techniques.

Various techniques can be employed for supervised learning, including linear models, artificial neural networks, support vector machines, and decision trees.

### 1.1.2 Unsupervised learning

Unsupervised learning encompasses two main tasks:

- *Clustering*: in this task, the objective is to group similar data points together based on their features, without predefined labels. The goal is to uncover underlying patterns or structures within the data. Clustering algorithms segment the data into clusters or groups, where data points within the same cluster exhibit greater similarity compared to those in different clusters. Unlike supervised learning, where labeled data is provided, clustering algorithms explore data solely based on features to identify similarities.
- *Dimensionality reduction*: this task involves reducing the number of input variables or features in a dataset while retaining essential information. This is often done to address the curse of dimensionality, enhance computational efficiency, and mitigate overfitting risks in models. Dimensionality reduction techniques aim to transform high-dimensional data into a lower-dimensional representation while preserving most relevant information.

Formally, in unsupervised learning, computers learn previously unknown patterns and efficient data representations. The training set is defined as $\mathcal{D} = \{x\}$, where the goal is to find a function $f$ that extracts a representation or grouping of the data.

Various techniques are used for unsupervised learning, including k-means clustering, self-organizing maps, and principal component analysis.

### 1.1.3 Reinforcement learning

Reinforcement learning encompasses several key approaches:

- *Markov decision process*: a mathematical framework for modeling decision-making, involving states, actions, transition probabilities, and rewards. The goal is to find a policy that maximizes cumulative rewards while considering uncertainty.

- *Partially observable MDP*: an extension of MDP where the current state is uncertain and must be inferred from observations. The objective remains the same, but the agent maintains a belief over possible states based on observations.
- *Stochastic games*: models for decision-making with multiple agents, where outcomes depend on actions and random factors. Players aim to optimize strategies considering other players' actions and uncertainties.

In reinforcement learning, the computer learns the optimal policy based on a training set $\mathcal{D}$ containing tuples $\langle x, u, x', r \rangle$, where $x$ is the input, $u$ is the action, $x'$ is the resulting state after the action, and $r$ is the reward. The policy $Q^*$ is defined to maximize $Q^*(x, u)$ over actions $u$ for each state $x$ in the training set.

Various techniques such as Q-learning, SARSA, and fitted Q-iteration are used to find this optimal policy.

# CHAPTER 2

# Supervised learning

## 2.1 Introduction

Supervised learning stands as the predominant and well-established learning approach. Its core objective is to enable a computer, given a training set $\mathcal{D} = \{\langle x, t \rangle\}$, to approximate a function $f$ that maps an input $x$ to an output $t$. The input variables $x$, often referred to as features or attributes, are paired with output variables $t$, also known as targets or labels. The tasks undertaken in supervised learning are as follows:

- *Classification*: when $t$ is discrete.
- *Regression*: when $t$ is continuous.
- *Probability estimation*: when$t$ represents a probability.

Supervised learning finds application in scenarios where:

- Humans lack the capability to perform the task directly (e.g., DNA analysis).
- Humans can perform the task but lack the ability to articulate the process (e.g., medical image analysis).
- The task is subject to temporal variations (e.g., stock price prediction).
- The task demands personalization (e.g., movie recommendation).

### 2.1.1 Function approximation

The process of approximating a function $f$ from a dataset $\mathcal{D}$ involves several steps:

1. *Define a loss function* $\mathcal{L}$: this function calculates the discrepancy between $f$ and $h$, a chosen approximation.
2. *Select a hypothesis space* $\mathcal{H}$: this space consists of a set of candidate functions from which to choose an approximation $h$.
3. *Minimize* $\mathcal{L}$ *within* $\mathcal{H}$: the goal is to find an approximation $h$ within the hypothesis space $\mathcal{H}$ that minimizes the loss function $\mathcal{L}$.

The hypothesis space $\mathcal{H}$ can be expanded to theoretically achieve a perfect approximation of the function $f$. However, a significant challenge arises because the loss function $\mathcal{L}$ cannot be easily determined, primarily due to the absence of the actual function $f$.

### 2.1.2 Taxonomy

The taxonomy is as follows:

- *Parametric* or *nonparametric*: parametric methods are characterized by having a fixed and finite number of parameters, while nonparametric methods have a number of parameters that depend on the training set.
- *Frequentist* or *Bayesian*: frequentist approaches utilize probabilities to model the sampling process, whereas Bayesian methods use probability to represent uncertainty about the estimate.
- *Empirical risk minimization* or *structural risk minimization*: empirical risk refers to the error over the training set, while structural risk involves balancing the training error with model complexity.

The type of machine learning can be:

- *Direct*: This method involves learning an approximation of $f$ directly from the dataset $\mathcal{D}$.
- *Generative*: in this approach, the model focuses on modeling the conditional density $\mathrm{P}(t|x)$ and then marginalizing to find the conditional mean:

$$\mathbb{E}\left[t|x\right] = \int t \cdot \mathrm{P}(t|x)dt$$

- *Discriminative*: This method models the joint density $\mathrm{P}(x,t)$, infers the conditional density $\mathrm{P}(t|x)$, and then marginalizes to find the conditional mean:

$$\mathbb{E}\left[t|x\right] = \int t \cdot \mathrm{P}(t|x)dt$$

## 2.2 Linear regression

The goal of regression is to approximate a function $f(x)$ that maps input $x$ to a continuous output $t$ from a dataset $\mathcal{D}$:

$$\mathcal{D} = \{\langle x, t\rangle\} \implies t = f(x)$$

Here, $x$ is a vector. To perform regression, we assume the existence of a function capable of performing this mapping. The key components of constructing a linear regression problem include:

- The method used to model the function $f$ (the hypothesis space).
- The evaluation criteria for the approximation (the loss function).
- The optimization process for optimizing the model.

In linear regression, the function $f$ is modeled using linear functions. This choice is motivated by several factors:

- Linear models are easily interpretable, making them suitable for explanation.
- Linear regression problems can be solved analytically, allowing for efficient computation.
- Linear functions can be extended to model nonlinear relationships.
- More sophisticated methods often build upon or incorporate elements of linear regression.

**Hypothesis space** In mathematical terms, the approximation $y$ can be defined as:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{D-1} w_j x_j = \mathbf{w}^T \mathbf{x}$$

Here, $\mathbf{x} = (1, x_1, \ldots, x_{D-1})$ is a vector, and $w_0$ is called the bias parameter. It's important to note that the output $y$ is a scalar value.

In a two-dimensional space, our hypothesis space will be the set of all points in the plane $(w_0, w_1)$. The coordinates of each point will correspond to a line in the $(\mathbf{x}, y)$ space.

**Loss function** A commonly used error loss function for the linear regression problem is the sum of squared errors (SSE), defined as:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(y(x_n, \mathbf{w}) - t_n\right)^2$$

This sum is also referred to as the residual sum of squares (RSS) and can be expressed as the sum of squared residual errors:

$$RSS(\mathbf{w}) = \left\|\boldsymbol{\epsilon}_2^2\right\| = \sum_{i=1}^{N} \epsilon_i^2$$

This formulation of the loss function allows for obtaining a closed-form optimization solution.

**Optimization** For linear models, a closed-form optimization of the RSS, known as least squares, begins with the matrix representation of the loss function:

$$L(\mathbf{w}) = \frac{1}{2} RSS(\mathbf{w}) = \frac{1}{2} \left(\mathbf{t} - \boldsymbol{\Phi}_{\mathbf{w}}\right)^T \left(\mathbf{t} - \boldsymbol{\Phi}_{\mathbf{w}}\right)$$

Here, $\boldsymbol{\Phi} = \begin{bmatrix}\phi(x_1) & \ldots & \phi(x_N)\end{bmatrix}^T$ and $\mathbf{t} = \begin{bmatrix}t_1 & \ldots & t_n\end{bmatrix}^T$. To find the optimal $\mathbf{w}$, we compute the first derivative of $L(\mathbf{w})$ and set it to zero:

$$\widehat{\mathbf{w}}_{OLS} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \mathbf{t}$$

However, the inversion of the matrix $\boldsymbol{\Phi}^T \boldsymbol{\Phi}^{-1}$ can be computationally expensive, especially for large datasets, with a complexity of $O(nm^2 + m^3)$, assuming the matrix is non-singular (invertible).

To mitigate this, stochastic gradient descent (SGD) can be employed. The algorithm known as least mean squares (LMS) uses the following update rule:

$$L(\mathbf{x}) = \sum_n L(x_n)$$

Expanding this, we get:

$$\begin{aligned}\mathbf{w}^{(n+1)} &= \mathbf{w}^{(n)} - \alpha^{(n)} \nabla L(x_n) \\ &= \mathbf{w}^{(n)} - \alpha^{(n)} \left( \mathbf{w}^{(n)^T} \phi(\mathbf{x}_n) - t_n \right) \phi(\mathbf{x}_n)\end{aligned}$$

Here, $\alpha$ is the learning rate, and convergence is guaranteed if $\sum_{n=0}^{\infty} = +\infty$ and $\sum_{n=0}^{\infty} = \alpha^{(n)^2} < +\infty$.

If the regression problem involves multiple outputs, meaning that $\mathbf{t}$ is not a scalar, we can solve each regression problem independently. However, we can still use the same set of basis functions. The solution for the weight vectors for all outputs can be expressed as:

$$\widehat{\mathbf{W}} = \left( \mathbf{\Phi}^T \mathbf{\Phi} \right)^{-1} \mathbf{\Phi}^T \mathbf{T}$$

Here, each column of matrix $\mathbf{T}$ and $\widehat{\mathbf{W}}$ corresponds to the target vector and the weight vector for each output, respectively. This solution can be easily decoupled for each output $k$:

$$\widehat{\mathbf{w}}_k = \left( \mathbf{\Phi}^T \mathbf{\Phi} \right)^{-1} \mathbf{\Phi}^T \mathbf{t}_k$$

An advantage of this approach is that $\left( \mathbf{\Phi}^T \mathbf{\Phi} \right)^{-1}$ only needs to be computed once, regardless of the number of outputs.

### 2.2.1 Basis function

While a linear combination of input variables may not always suffice to model data, we can still construct a regression model that is linear in its parameters. This can be achieved by defining a model using non-linear basis functions, expressed as:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

Here, the components of the vector $\boldsymbol{\phi}(\mathbf{x}) = (1, \phi_1(\mathbf{x}), \ldots, \phi_{M-1}(\mathbf{x}))^T$ are referred to as features. These features allow for a more flexible representation of the input data, enabling the model to capture non-linear relationships between the input variables and the output.

**Example:**
Let's reconsider a set of data regarding individuals' weight and height, along with their completion times for a one-kilometer run:

| Height (cm) | Weight (kg) | Completion time (s) |
|---|---|---|
| 180 | 70 | 180 |
| 184 | 80 | 220 |
| 174 | 60 | 170 |

We can model this problem using a dummy variable and introduce the Body Mass Index (BMI) as a new feature:

| **Dummy variable** | **Height (cm)** | **Weight (kg)** | **BMI** | **Completion time (s)** |
|---|---|---|---|---|
| $x_0$ | $x_1$ | $x_2$ | $x_3$ | $t$ |
| 1 | 180 | 70 | 21 | 180 |
| 1 | 184 | 80 | 23 | 220 |
| 1 | 174 | 60 | 20 | 170 |

Here, the dummy variable $x_0$ is always initialized to one. Now, we have the option to retain or discard the weight and height variables, considering only the BMI values for analysis.

The most commonly used basis functions in regression are:

- *Polynomial*:
$$\phi_j(x) = x^j$$
- *Gaussian*:
$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$
- *Sigmoidal*:
$$\phi_j(x) = \frac{1}{1+\exp\left(\frac{\mu_j - x}{\sigma}\right)}$$

Here, the constant $\mu_j$ is referred to as a hyperparameter, as its value needs to be determined through experimentation and depends on the user's experience.

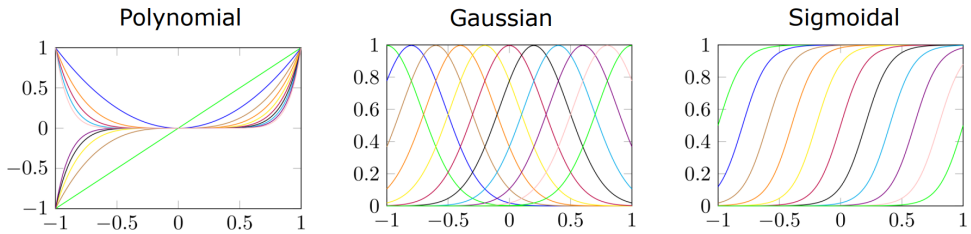


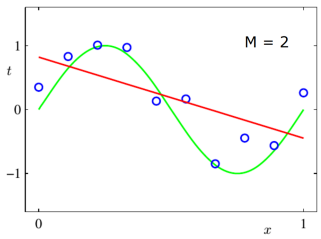

Figure 2.1: Some possible basis functions shapes

It's noteworthy that the Gaussian basis function allows for a local approximation by omitting values that are close to zero. This approach enables capturing the relationship between the input and output in a reduced input space area. As we move away from the mean, approaching zero, the values become negligible.

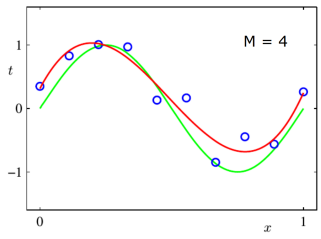
### 2.2.2 Regularization

A function can achieve a better approximation by increasing the degree of the polynomial used in the regression.

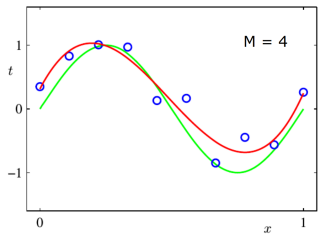
**Example:**
Consider a function generating a set of points with some noise:




Using a second-order polynomial instead of a linear one provides a better approximation:




Further improving the approximation can be achieved with a higher-degree polynomial (e.g., ninth grade):




However, increasing the polynomial degree also increases the complexity of the model parameters. To address this complexity, adjustments are needed in the loss function:

$$L(\mathbf{w}) = L_D(\mathbf{w}) + \lambda L_W(\mathbf{w})$$

Here, $L_D(\mathbf{w})$ represents the usual loss function, $L_W(\mathbf{w})$ reflects model complexity (a hyperparameter), and $\lambda$ is the regularization coefficient. $L_W(\mathbf{w})$ can be tailored using ridge regression or lasso methods.

**Ridge regression** In ridge regression, the regularization term $L_W(\mathbf{w})$ is defined as:

$$L_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\left\|\mathbf{w}\right\|_2^2$$

Thus, the overall loss function becomes:

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\phi(x_i)\right)^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|_2^2$$

Despite the regularization term, the loss function remains quadratic with respect to $w$, allowing for closed-form optimization:

$$\widehat{\mathbf{w}}_{ridge} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathrm{t}$$

The term $\lambda\mathbf{I}$ is crucial in solving the singularity problem, as it transforms a non-singular matrix into a singular one with an appropriate choice of $\lambda$.

**Lasso** Another common regularization method is lasso, where the regularization term $L_W(\mathbf{w})$ is defined as:

$$L_W(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_1 = \frac{1}{2} \sum_{j=0}^{M-1} |w_j|$$

Thus, the overall loss function becomes:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(t_i - \mathbf{w}^T \phi(x_i)\right)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_1$$

In this case, closed-form optimization is not possible. However, lasso typically leads to sparse regression models: when the regularization coefficient $\lambda$ is large enough, some components of $\widehat{\mathbf{w}}$ become equal to zero. Regularization can be seen as equivalent to minimizing $L_D(\mathbf{w})$ subject to the constraint:

$$\sum_{j=0}^{M-1} |w_j| \leq \eta$$