Machine Learning & Deep Learning

Artificial Intelligence

Jacopo Parretti

I Semester 2025-2026

Indice

Parte I

Introduction to Machine Learning

1 What is Machine Learning?

Machine Learning is a branch of Artificial Intelligence that enables software to use data to find solutions to specific tasks without being explicitly programmed to do so.

1.1 Machine Learning vs Statistical Modelling

In traditional statistical modelling, we follow a structured process:

1. Collect data
2. Verify and clean the data (correct or discard if not clean)
3. Use the clean data to test hypotheses, make predictions and forecasts

In contrast, Machine Learning takes a different approach:

- It is the **data** that determines which analytic techniques to use
- The computer uses data to **train** algorithms to find patterns and make predictions
- Algorithms are no longer **static** but become **dynamic**

1.2 Conventional Programming vs Machine Learning

The fundamental difference between conventional programming and machine learning can be understood through their inputs and outputs:

1.2.1 Conventional Programming

- **Input**: Program + Data
- **Output**: Result
- You write the program, give it data, and get results

1.2.2 Machine Learning

- **Input**: Data + Result
- **Output**: Program
- You give the computer data and desired results, and it learns to generate a program/algorithm

1.3 The Machine Learning Recipe

The ML recipe consists of 4 fundamental steps:

1. Collect data
2. Define a family of possible models
3. Define an objective (error) function to quantify how well a model fits the data
4. Find the model that minimizes the error function (training/learning a model)

1.3.1 The Key Ingredients

Every machine learning problem requires five essential ingredients:

- **Task**: The problem we want to solve
- **Data**: The information we use to learn
- **Model hypothesis**: The family of functions we consider
- **Objective function**: How we measure success
- **Learning algorithm**: How we find the best model

2 Machine Learning Tasks

A task represents the type of prediction being made to solve a problem on some data. We can identify a task with the set of functions that can potentially solve the problem. In general, it consists of functions assigning each **input** $x \in \mathcal{X}$ an **output** $y \in \mathcal{Y}$.

2.1 Classification Task

Definizione 2.1 (Classification)**.** Find a function $f \in \mathcal{Y}^{\mathcal{X}}$ assigning each input $x \in \mathcal{X}$ a **discrete** label, $f(x) \in \mathcal{Y} = \{c_1, \ldots, c_k\}$.

In classification, the output space is a finite set of discrete categories or classes. The goal is to learn a decision boundary that separates different classes.

2.2 Regression Task

Definizione 2.2 (Regression)**.** Find a function $f \in \mathcal{Y}^{\mathcal{X}}$ assigning each input $x \in \mathcal{X}$ a **continuous** label, $f(x) \in \mathcal{Y} = \mathbb{R}$.

In regression, the output is a continuous value, allowing for predictions of quantities such as prices, temperatures, or probabilities.

2.3 Density Estimation Task

Definizione 2.3 (Density Estimation)**.** Find a probability distribution $f \in \Delta(\mathcal{X})$ that best fits the data $x \in \mathcal{X}$.

There is no reference to an output space (as in classification and regression tasks). Instead, we make a reasoning on the **input** x itself, trying to model the underlying probability distribution that generated the data.

2.4 Clustering Task

Definizione 2.4 (Clustering)**.** Find a function $f \in \mathbb{N}^{\mathcal{X}}$ that assigns each input $x \in \mathcal{X}$ a **cluster index** $f(x) \in \mathbb{N}$.

All points mapped to the same index form a cluster. The goal is to group similar data points together without prior knowledge of the groups.

2.5 Dimensionality Reduction Task

Definizione 2.5 (Dimensionality Reduction)**.** Find a function $f \in \mathcal{Y}^{\mathcal{X}}$ that **maps** each (**high-dimensional**) input $x \in \mathcal{X}$ to a **lower-dimensional** embedding $f(x) \in \mathcal{Y}$, where $\dim(\mathcal{Y}) < \dim(\mathcal{X})$.

This task aims to reduce the number of features while preserving the essential information in the data.

3 Data in Machine Learning

3.1 Data Representation

We represent inputs of our algorithm as $x \in \mathcal{X}$ and outputs as $y \in \mathcal{Y}$. The dataset is the set of both, where each input is associated with an output.

Key considerations:

- The quality of the data is crucial for the performance of the algorithm
- It is equally important to have a large amount of data to learn effective models
- Poor quality or insufficient data can lead to poor model performance

3.2 Features

Definizione 3.1 (Features)**.** Features are the measurable properties or characteristics of the phenomenon being observed.

Examples:

- To identify a flower: color, petal length, petal width, sepal length, sepal width
- To identify a fruit: color, shape, size, texture, weight
- To classify images: pixel values, edges, textures, shapes

We can say that features are how our algorithm "sees" the data and are in general represented with (fixed-size) vectors.

4 Data Distributions and Learning Types

In machine learning, the information about the problem we want to solve is represented in the form of a data distribution, usually denoted as p_{data}.

4.1 Supervised Learning

4.1.1 Classification and Regression

The data distribution is over pairs of inputs and outputs:

$$p_{\text{data}} \in \Delta(\mathcal{X} \times \mathcal{Y})$$

Here:

- $\mathcal{X}$ is the input space
- $\mathcal{Y}$ is the output space (labels or targets)
- The goal is to learn a mapping from inputs to outputs using labeled data

4.2 Unsupervised Learning

4.2.1 Density Estimation, Clustering, and Dimensionality Reduction

The data distribution is only over the input space:

$$p_{\text{data}} \in \Delta(\mathcal{X})$$

Here:

- We do not have explicit labels or targets
- The goal is to discover structure, patterns, or representations within the data itself

4.3 Summary of Learning Types

Task Type	**Data Distribution**	**Learning Type**
Classification, Regression	$p_{\text{data}} \in \Delta(\mathcal{X} \times \mathcal{Y})$	Supervised Learning
Density Estimation, Clustering, Dimensionality Reduction	$p_{\text{data}} \in \Delta(\mathcal{X})$	Unsupervised Learning

Tabella 1: Comparison of learning types based on data distribution

Nota. In **supervised learning**, each data point consists of an input and a corresponding output (label). In **unsupervised learning**, each data point consists only of the input, and the algorithm tries to find patterns or structure without explicit labels.

This distinction is fundamental in machine learning, as it determines the type of algorithms and approaches used to solve different problems.

5 Data Sampling and Dataset Splitting

5.1 The Data Distribution Problem

In both supervised and unsupervised learning, the true data distribution p_{data} is typically **unknown**. This presents a fundamental challenge:

- We do not have direct access to the underlying probability distribution that generates the data
- We only have access to a finite sample of data points drawn from this distribution

Solution: We perform **sampling** from the distribution. By collecting a representative sample of data, we can approximate the true distribution and use it to train our models.

5.2 Training, Validation, and Test Sets

To properly evaluate machine learning models, we split our data into three distinct sets. Each set serves a specific purpose in the machine learning pipeline.

Definizione 5.1 (Dataset Splitting)**.** A dataset is typically divided into three subsets:

- **Training set** (D_{train}): Used to train the model
- **Validation set** (D_{val}): Used to tune hyperparameters and select the best model
- **Test set** (D_{test}): Used to evaluate the final model performance

Nota. **Critical requirement**: All three sets (training, validation, and test) should be sampled from the **same probability distribution**. This ensures that the model's performance on the test set is a reliable indicator of its performance on new, unseen data.

5.3 How to Use the Three Sets

The three datasets serve different purposes in the machine learning workflow:

5.3.1 Learning/Training Phase

During the learning phase, the **training set** is used to train the machine learning algorithm, which learns patterns and relationships in the data to produce a **program** (model).

The **validation set** plays a crucial role in this phase:

- Evaluates different model configurations
- Tunes hyperparameters
- Selects the best performing model

This process involves iterative feedback between training and validation until we find the optimal configuration.

5.3.2 Testing/Model Evaluation Phase

Once the best model is selected, we use the **test set** to evaluate its performance. The trained model is applied to the test data, which it has never seen before. This gives us an unbiased estimate of how well the model will perform in real-world scenarios, telling us whether our model can truly generalize to new data.

5.4 The Importance of Distribution Similarity

5.4.1 Correct Scenario: Same Distribution

When training, validation, and test sets are sampled from the **same distribution**:

- The sets are **similar** in terms of statistical properties
- They represent the same underlying phenomenon
- However, they **do not overlap** (no data point appears in multiple sets)

This ensures that the model can generalize well from training to test data.

Example: If we're classifying fruits, all three sets might contain apples and bananas in similar proportions, but with different specific instances.

5.4.2 Incorrect Scenario: Different Distributions

When training/validation and test sets come from **different distributions**, the model learns patterns from one distribution but is tested on another. This problematic scenario typically leads to:

- Poor performance on the test set
- Unreliable predictions on new data
- Inability to generalize to real-world applications

Example: Training on apples and bananas, but testing on oranges and limes. The model cannot classify fruits it has never seen during training.

Nota. **Key principle**: The fundamental assumption in machine learning is that training and test data come from the same distribution. Violating this assumption leads to poor generalization and unreliable models.

6 Training and Testing: The Role of Features

6.1 Training (Learning, Induction)

During the training phase, the model learns to distinguish between different classes based on the features provided in the training data.

Example: Consider a fruit classification task with the following training data:

- *red, round, leaf, 3oz, ...* → apple
- *green, round, no leaf, 4oz, ...* → apple
- *yellow, curved, no leaf, 8oz, ...* → banana
- *green, curved, no leaf, 7oz, ...* → banana

The model learns to associate patterns in the features (color, shape, presence of leaf, weight) with the corresponding labels (apple or banana). **What distinguishes apples and bananas is based on the features!** The model identifies which feature combinations are characteristic of each class.

6.2 Testing

Once trained, the model can classify new examples based on the same features it learned during training. When presented with a new example described by the same semantic features, the model makes a prediction.

Example: Given a new fruit with features *red, round, no leaf, 4oz, ...*, the model uses the learned patterns to predict whether it is an apple or a banana.

Nota. **Critical assumption**: The features of the training and testing data must be the same! The model can only make predictions based on the features it was trained on. If the test data uses different features or feature representations, the model cannot function properly.

6.3 Learning and Generalization

Learning is fundamentally about **generalizing** from the training data. The goal is not simply to memorize the training examples, but to learn patterns that apply to new, unseen data.

However, the failure of a machine learning algorithm is often caused by a **bad selection of training samples**. The key issue is that we might introduce **unwanted correlations** from which the algorithm derives **wrong conclusions**.

Example: Consider a model trained to recognize objects in images. If all training images were captured on sunny days, the model might learn to associate sunny weather conditions with the objects. When tested on images taken on cloudy days, the model might fail to recognize the same objects because it never encountered cloudy conditions during training. The model incorrectly learned that sunny weather is a relevant feature for object recognition.

Osservazione. The quality and diversity of training data are crucial. Training data should be representative of all conditions the model will encounter in real-world applications, avoiding spurious correlations that lead to poor generalization.

7 The Complete Machine Learning Recipe

We can now revisit the complete machine learning workflow with all its essential ingredients:

7.1 The Five Ingredients of Machine Learning

Every machine learning problem requires five fundamental components that work together:

1. **Task**: The specific problem we want to solve (e.g., classify birds vs. non-birds in images)
2. **Data**: The collection of examples used to train and evaluate the model. Data quality and quantity are critical for success.
3. **Model Hypothesis**: The family of functions or models we consider as potential solutions. This defines the space of possible models the learning algorithm can explore.
4. **Objective Function**: A mathematical function that quantifies how well a model performs on the data. The learning algorithm aims to optimize this function.
5. **Learning Algorithm**: The method used to search through the model hypothesis space and find the model that optimizes the objective function.

7.2 The Interplay of Ingredients

These five ingredients are interconnected:

- The **task** determines what kind of **data** we need and what **model hypothesis** is appropriate
- The **data** influences which features are available and how we define the **objective function**
- The **model hypothesis** constrains what the **learning algorithm** can discover
- The **objective function** guides the **learning algorithm** toward better models

Understanding and carefully selecting each ingredient is essential for building effective machine learning systems.

8 Model and Hypothesis Space

8.1 What is a Model?

Definizione 8.1 (Model)**.** A **model** is the implementation of a function $f \in \mathcal{F}_{\text{task}}$ that can be tractably computed.

In practice, when searching for our function f, we don't look at everything in $\mathcal{F}_{\text{task}}$. Instead, we look at a subset called the **hypothesis space**: $\mathcal{H} \subset \mathcal{F}_{\text{task}}$, which is composed of a set of models (e.g., neural networks, decision trees, linear models).

The learning algorithm seeks a **solution within the hypothesis space**. In other words, a model is a simplified representation of the world that we can actually compute and optimize.

Osservazione. The choice of hypothesis space is crucial:

- If $\mathcal{H}$ is too small, it may not contain a good solution

- If $\mathcal{H}$ is too large, finding the best model becomes computationally intractable
- The hypothesis space defines the types of patterns the model can learn

8.2 Types of Learning Paradigms

Machine learning can be categorized into different paradigms based on the type and availability of labeled data:

8.2.1 Supervised Learning

In supervised learning, we have labeled data where each input is paired with its corresponding output. This includes:

- **Classification**: Predicting discrete labels
- **Regression**: Predicting continuous values

8.2.2 Unsupervised Learning

In unsupervised learning, we only have input data without labels. The goal is to discover structure in the data. This includes:

- **Clustering**: Grouping similar data points
- **Density Estimation**: Modeling the probability distribution of data
- **Dimensionality Reduction**: Finding lower-dimensional representations

8.2.3 Semi-Supervised Learning

Semi-supervised learning addresses scenarios where we have a large amount of unlabeled data and only some labeled examples. This paradigm has gained significant popularity in recent years to leverage large datasets through self-supervised learning techniques.

The key idea is to provide the model with tasks different from the one to be solved, but that allow the model to learn the data distribution from the unlabeled data. Once the model understands the general structure of the data, it can then specialize on the few labeled examples to solve the target task.

Example: Training a language model on large amounts of unlabeled text to learn general language patterns, then fine-tuning it on a small labeled dataset for sentiment analysis.

9 Model-Based vs. Instance-Based Learning

Machine learning algorithms can be further categorized based on how they make predictions:

9.1 Model-Based (Parametric) Learning

Definizione 9.1 (Parametric Model)**.** A learning model that summarizes data with a **set of parameters of fixed size** (independent of the number of training examples) is called a **parametric model**.

Key characteristics:

- The model learns a fixed set of parameters from the training data
- Once trained, the original training data can be discarded

- No matter how much data you give to a parametric model, it won't change its mind about how many parameters it needs
- The model makes predictions using only the learned parameters

Examples: Naive Bayes, Linear Regression, Logistic Regression, Neural Networks

In a model-based approach, the algorithm learns a function (e.g., a decision boundary) described by parameters. For instance, a linear classifier might learn the curve $\alpha x_1 + \beta x_2 + \gamma$ that separates two classes, where α, β, γ are the parameters.

9.2 Instance-Based (Nonparametric) Learning

Definizione 9.2 (Nonparametric Model)**.** These algorithms **do not learn parameters** but instead use the data itself to compare a new example through similarity metrics.

Key characteristics:

- The model stores (some or all of) the training data
- Predictions are made by comparing new examples to stored training examples
- The model's complexity can grow with the amount of training data
- No explicit training phase that produces a fixed set of parameters

Examples: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest

In an instance-based approach, when a new example arrives, it is compared with nearby training examples and classified according to a similarity policy. For example, in KNN, a new point is classified based on the majority class of its k nearest neighbors in the training data.

9.3 Comparison

Aspect	**Model-Based**	**Instance-Based**
Parameters	Fixed number of parameters	No fixed parameters
Training data	Can be discarded after training	Must be retained for predictions
Prediction speed	Fast (only uses parameters)	Slower (compares with stored data)
Memory	Low (stores only parameters)	High (stores training examples)
Flexibility	Fixed model complexity	Complexity grows with data

Tabella 2: Comparison between model-based and instance-based learning

10 Error Function (Objective Function)

To allow an algorithm to learn, we need to provide it with a metric, a measure that helps it understand the direction of the optimal solution we are seeking. For this reason, **performance measures** are defined to enable the algorithm to learn.

10.1 What is an Error Function?

In machine learning, **training a model** means finding a **function** which maps a set of values x to a value y. We can calculate how well a predictive model is doing by comparing the **predicted values** with the **true values** for y.

Definizione 10.1 (Training Error vs. Test Error)**.**
- If we apply the model to the data it was trained on, we are calculating the **training error**
- If we calculate the error on data which was **unknown** in the training phase, we are calculating the **test error**

The test error is the most important metric because it tells us how well the model generalizes to new, unseen data.

10.2 Types of Error Functions

There are different types of error functions (also called loss functions or objective functions), each suited for different tasks:

- **Mean Absolute Error (MAE)**: Measures the average absolute difference between predictions and true values
- **Mean Squared Error (MSE)**: Measures the average squared difference between predictions and true values
- **Binary Cross-Entropy**: Used for binary classification problems
- **Mean Average Precision**: Used for ranking and information retrieval tasks

The choice of error function depends on the task and the desired properties of the model.

11 Example: Polynomial Fitting

Let's consider a concrete example to understand how models, hypothesis spaces, and error functions work together.

11.1 Problem Setup

Consider a regression problem with the following setup:

- **Data**: $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Data generated from $\sin(2\pi x) + \text{noise}$
- Training set with $n = 10$ points
- **Goal**: Learn a function (shown as the purple line) that fits the data

The correct solution should capture the underlying sinusoidal pattern while being robust to the noise in the training data.

11.2 Model: Polynomial Functions

We choose to model the data using polynomial functions:

$$f_w(x) = \sum_{j=0}^{M} w_j x^j$$

where:

- M is the **degree of the polynomial** (a hyperparameter)
- $w = \{w_0, \ldots, w_M\}$ are the **parameters** to be learned

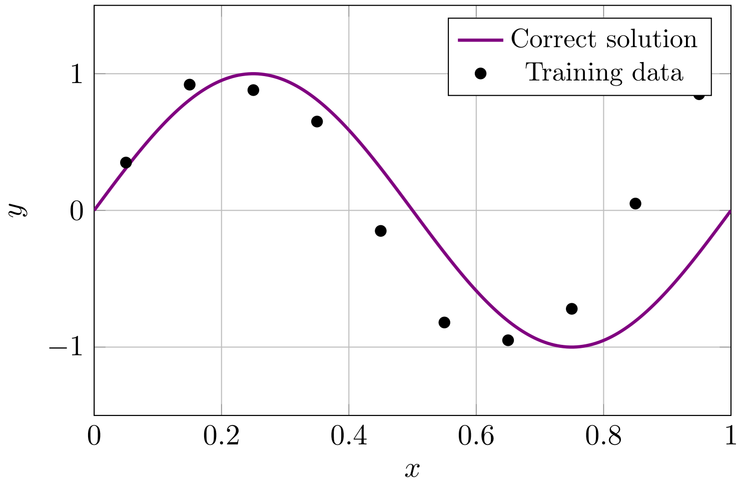


Figura 1: Polynomial Fitting Problem: Data generated from $\sin(2\pi x)$ with noise

- The **hypothesis space** for a fixed degree M is: $\mathcal{H}_M = \{f_w : w \in \mathbb{R}^{M+1}\}$

Different values of M give us different hypothesis spaces:

- $M = 0$: Constant functions (horizontal lines)
- $M = 1$: Linear functions (straight lines)
- $M = 2$: Quadratic functions (parabolas)
- $M = 3$: Cubic functions
- Higher M: More complex curves

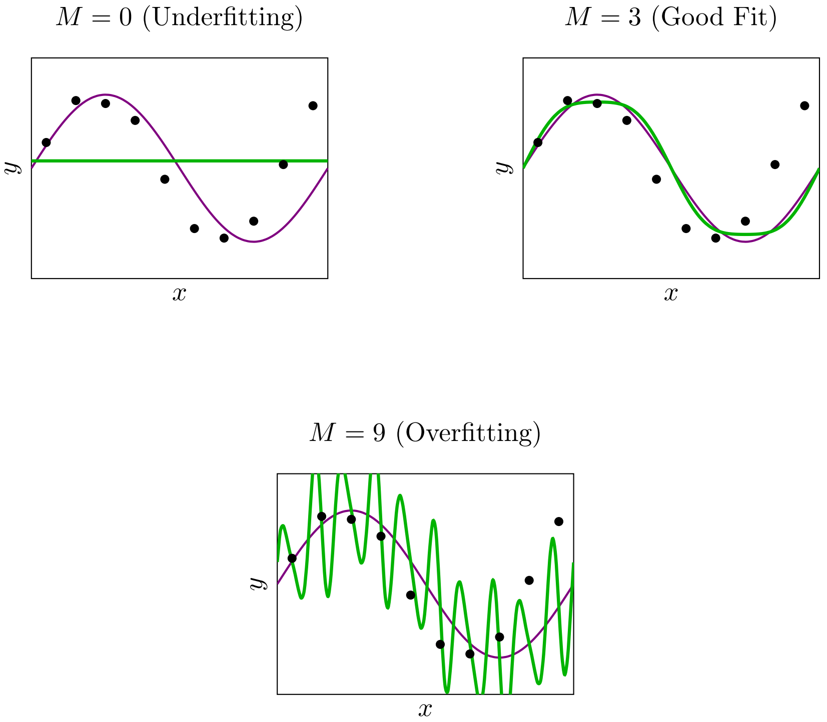


Figura 2: Polynomial fits with different degrees: $M = 0$ (underfitting), $M = 3$ (good fit), $M = 9$ (overfitting). Purple: true function, Green: fitted polynomial, Black dots: training data.

11.3 Error Function: Mean Squared Error

To measure how well our polynomial fits the data, we use the Mean Squared Error (MSE):

$$E(f; \mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^{n} [f(x_i) - y_i]^2$$

This error function:

- Computes the squared difference between the prediction $f(x_i)$ and the ground-truth label y_i for each point
- Averages these squared differences over all n training points
- Is also called a **pointwise loss** because it measures the error at each individual data point

The learning algorithm's goal is to find the parameters w that minimize this error function:

$$w^* = \arg \min_{w \in \mathbb{R}^{M+1}} E(f_w; \mathcal{D}_n)$$

Osservazione. The choice of polynomial degree M is crucial:

- Too small M (e.g., $M = 0$ or $M = 1$): The model is too simple and cannot capture the sinusoidal pattern (**underfitting**)
- Appropriate M (e.g., $M = 2$ or $M = 3$): The model captures the underlying pattern well
- Too large M: The model fits the noise in the training data and doesn't generalize well (**overfitting**)

12 Underfitting and Overfitting

Two fundamental problems can occur when training machine learning models: underfitting and overfitting. Understanding these concepts is crucial for building models that generalize well.

12.1 Definitions

Definizione 12.1 (Underfitting)**. Underfitting** is a scenario where a model is unable to capture the relationship between the input (x) and output (y) variables accurately, generating a high error rate on both the training set and unseen data (e.g., testing set).

The model is too simple and has not yet learned from the data. It performs poorly on both training and test data.

Definizione 12.2 (Overfitting)**. Overfitting** occurs when the model gives accurate predictions for training data (lower training error) but not for testing data (higher testing error).

The model is too sensitive to the training data and has essentially memorized it, including its noise and peculiarities, rather than learning the underlying pattern.

12.2 The Bias-Variance Trade-off

The relationship between model complexity, training error, and test error can be visualized as follows:

- **Left side of the graph (low complexity)**: High error rate in both training and testing → **Underfitting**

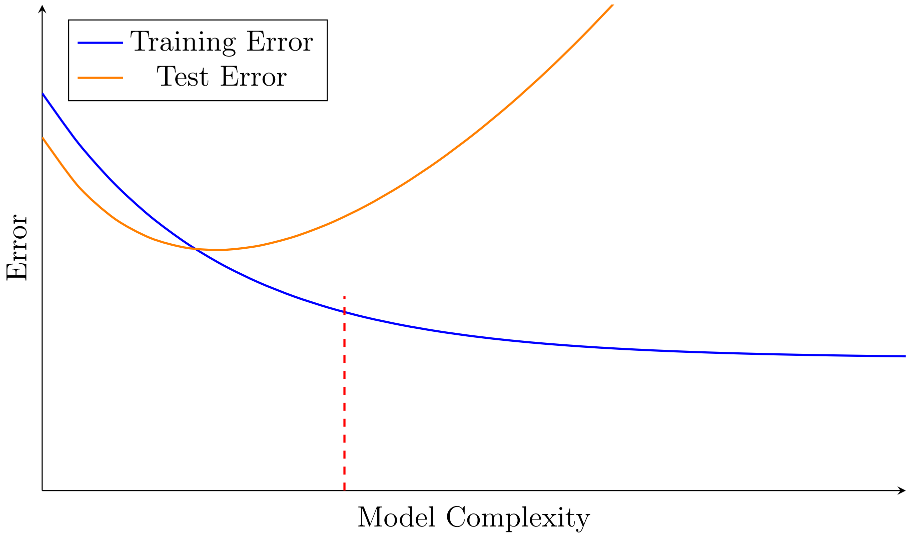


Figura 3: Bias-Variance Trade-off: Training and Test Error vs. Model Complexity

- **Middle region (optimal complexity)**: Low error rate in both training and testing → **Best Fit**
- **Right side of the graph (high complexity)**: Low error rate in training but high error rate in testing → **Overfitting**

12.3 Diagnostic Matrix

We can diagnose the state of our model by examining both training and test errors:

	Low Training Error	**High Training Error**
Low Test Error	OK (Good fit)	Underfitting
High Test Error	Overfitting	Underfitting

Tabella 3: Diagnostic matrix for model performance

12.4 How to Handle Overfitting

When the model is too sensitive to training data and overfits, we can apply several strategies:

1. **Try a simpler model**: Use a model with fewer parameters or lower complexity
2. **Try a less powerful model**: Choose a model architecture with reduced capacity
3. **Increase regularization impact**: Apply techniques that penalize model complexity:
   - Early stopping: Stop training before the model overfits
   - L1/L2 regularization: Add penalty terms to the loss function
4. **Use a smaller number of features**:
   - Remove some features that may be causing overfitting
   - Apply feature selection techniques to identify the most relevant features
5. **Get more data**: Increasing the training set size helps the model learn more generalizable patterns

12.5 How to Handle Underfitting

When the model has not yet learned from the data and underfits, we can try:

1. **Try more complex models**: Use models with a larger number of parameters that can capture more intricate patterns
    - Ensemble learning: Combine multiple models
2. **Less regularization**: Reduce or remove regularization constraints that may be limiting the model's capacity
3. **A larger quantity of features**: Add more features or engineer new features that better capture the underlying relationships (get more features)

12.6 Polynomial Fitting Example Revisited

Let's examine how different polynomial degrees affect the fit:

- $M = 0$ (constant function): Severe underfitting. The horizontal line cannot capture any of the sinusoidal pattern. Both training and validation errors are high.
- $M = 1$ (linear function): Still underfitting. A straight line cannot represent the curved pattern. Both errors remain high.
- $M = 3$ (cubic polynomial): Good fit. The model captures the underlying sinusoidal pattern well. Both training and validation errors are low, and the curves are similar.
- $M = 9$ (9th-degree polynomial): Overfitting. The model passes through all training points (very low training error) but oscillates wildly between them. The validation error is high because the model has learned the noise rather than the signal.

The error plot shows:

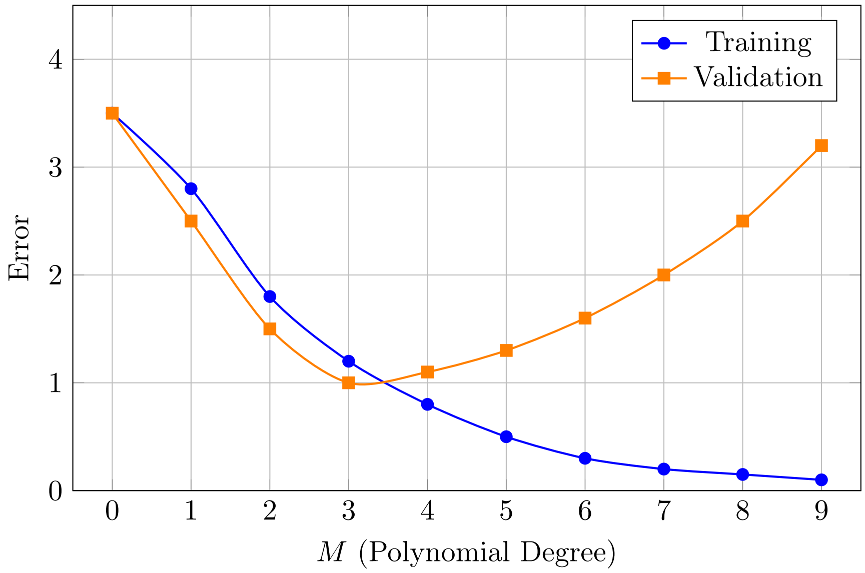


Figura 4: Training and Validation Error vs. Polynomial Degree M

- **Training error** (blue line): Decreases monotonically as M increases
- **Validation error** (orange line): Decreases initially, reaches a minimum around $M = 3$, then increases again

- The optimal model complexity is where the validation error is minimized

Nota. The key insight is that minimizing training error alone is not sufficient. We must monitor validation/test error to ensure the model generalizes well to new data. The best model is the one that balances fitting the training data with generalizing to unseen data.

13 Regularization

Regularization is one of the most important techniques in machine learning to prevent overfitting and improve model generalization.

13.1 What is Regularization?

Definizione 13.1 (Regularization)**.** Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

In practice, regularization is a modification of the training error function by appending a term $\Omega(f)$ that typically penalizes complex solutions.

13.2 The Regularized Objective Function

The regularized objective function has the following form:

$$E_{\text{reg}}(f; \mathcal{D}_n) = E(f; \mathcal{D}_n) + \lambda_n \Omega(f)$$

where:

- $E(f; \mathcal{D}_n)$ is the **training error function** (e.g., MSE)
- $\Omega(f)$ is the **regularization term** that penalizes model complexity
- λ_n is the **tradeoff hyperparameter** that controls the strength of regularization

The regularization term acts as a penalty that discourages the model from becoming too complex. By minimizing this combined objective, we find models that fit the data well while remaining relatively simple.

13.3 Effect of Regularization

The regularization term $\Omega(f)$ typically measures some notion of model complexity. When we minimize the regularized objective:

- We balance fitting the training data (low $E(f; \mathcal{D}_n)$) with keeping the model simple (low $\Omega(f)$)
- The hyperparameter λ_n controls this tradeoff
- Higher λ_n means stronger regularization (simpler models)
- Lower λ_n means weaker regularization (more complex models allowed)

13.4 Regularization in Polynomial Fitting

For the polynomial fitting example, we can regularize by penalizing polynomials with large coefficients. The regularized error function becomes:

$$E_{\text{reg}}(f_w; \mathcal{D}_n) = \frac{1}{n}\sum_{i=1}^{n}[f_w(x_i) - y_i]^2 + \frac{\lambda}{n}\|w\|^2$$

where:

- The first term is the training error (MSE)
- $\frac{\lambda}{n}\|w\|^2 = \frac{\lambda}{n}\sum_{j=0}^{M} w_j^2$ is the regularization term (L2 regularization)
- λ is the tradeoff hyperparameter

13.4.1 Effect of the Regularization Parameter λ

The choice of λ significantly affects the model:

- **$\lambda \approx 10^{-18}$ (very small)**: Almost no regularization. The model can have large coefficients and may overfit. The polynomial fits the training data very closely, including noise.
- **$\lambda = 1$ (moderate)**: Balanced regularization. The model is penalized for having large coefficients, leading to a smoother curve that generalizes better. This often provides the best tradeoff between training and validation error.
- **λ very large**: Strong regularization. The penalty for large coefficients is so high that the model becomes too simple and may underfit. The curve becomes nearly flat.

13.4.2 Regularization Path

The plot of error vs. ln(λ) shows:

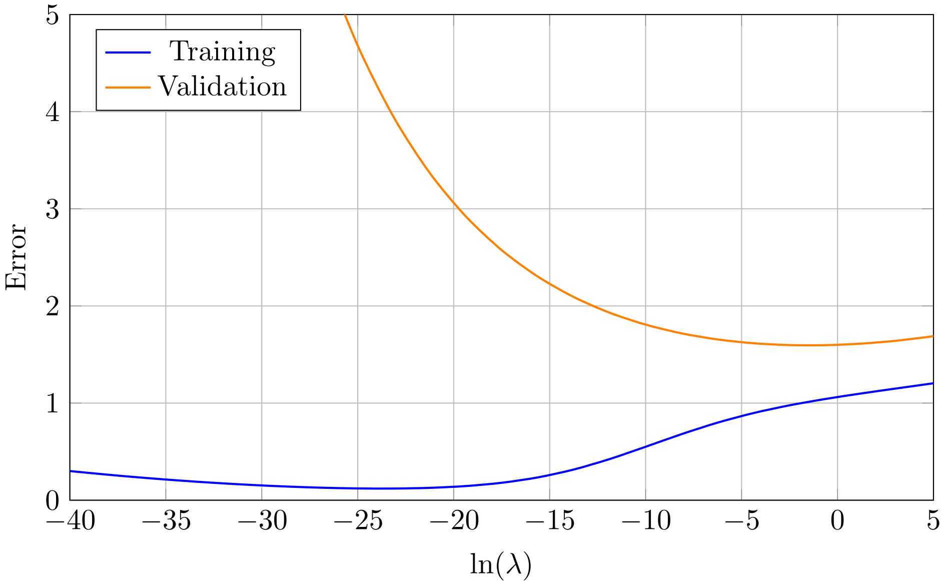


Figura 5: Training and Validation Error vs. ln(λ) (Regularization Strength)

- **Left side (small λ)**: Low training error, high validation error → Overfitting
- **Middle region**: Both errors are low → Good fit
- **Right side (large λ)**: Both errors increase → Underfitting

The optimal λ is found where the validation error is minimized.

13.5 Generalization and Data Size

An important theoretical result in machine learning relates training error to generalization error:

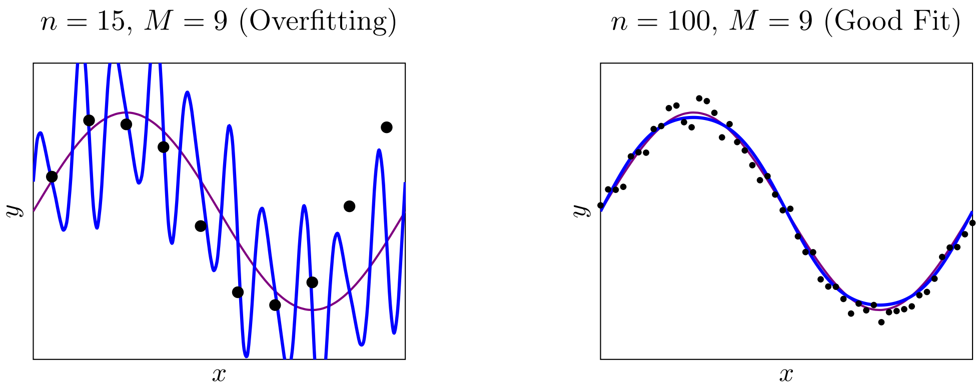
Teorema 13.1 (Generalization with Infinite Data)**.** *As the number of training samples approaches infinity, the training error approximates the generalization error:*

$$E(f; \mathcal{D}_n) \to E(f; p_{data}) \quad as \quad n \to \infty$$

This means:

- With a small dataset ($n = 15$), even a complex model ($M = 9$) may overfit because there isn't enough data to constrain the parameters
- With a large dataset ($n = 100$), the same complex model can generalize well because the abundant data prevents overfitting
- More data allows us to use more complex models without overfitting

Visual example:


$n = 15$, $M = 9$ (Overfitting)

$n = 100$, $M = 9$ (Good Fit)

Figura 6: Effect of data size on generalization: With few data points ($n = 15$), a complex model ($M = 9$) overfits. With many data points ($n = 100$), the same model generalizes well. Purple: true function, Blue: fitted polynomial, Black dots: training data.

- **$n = 15$, $M = 9$**: The 9th-degree polynomial overfits, oscillating wildly between the few training points
- **$n = 100$, $M = 9$**: With 100 training points, the same 9th-degree polynomial fits smoothly and generalizes well

Osservazione. This illustrates why "get more data" is often the most effective solution to overfitting. With sufficient data, even complex models can learn to generalize well. However, collecting more data is not always feasible, which is why regularization and other techniques remain important.