Machine Learning Overview

AI for Quant Research
AiQR Academy

December 6, 2025

Contents

1 Introduction to Machine Learning

1.1 What is Machine Learning?

Machine Learning (ML) is a branch of artificial intelligence that allows a model to learn patterns from data without being explicitly programmed for a specific task. The goal is to learn a function f that maps inputs X to outputs Y, i.e., $f : X \to Y$.

2 Types of Machine Learning

2.1 Supervised Learning

Supervised learning involves learning from labeled data where each input is associated with a correct output. The goal is to learn a mapping function $f : X \to Y$ from the inputs X (features) to the outputs Y (targets), which can be continuous (regression) or categorical (classification).

2.1.1 Regression

Regression is used to predict a continuous output y based on input features $X = [x_1, x_2, \dots, x_n]$. A common model is the **linear regression** model, where the relationship between the inputs and the output is assumed to be linear.

The general form of the linear regression model is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

where β_0 is the intercept, $\beta_1, \beta_2, \dots, \beta_n$ are the coefficients (weights) for each input feature, and ϵ is the error term. The goal is to estimate the coefficients β that minimize the error between the predicted values $\hat{y}$ and the actual values y. The common objective function used is the **Mean Squared Error** (MSE):

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

where m is the number of training samples.

Example in Finance: - *Predicting stock prices or returns:* A linear regression model can be used to predict the price of a stock based on historical prices, trading volume, interest rates, and other financial ratios (e.g., P/E ratio). - Features (X): historical stock price, trading volume, interest rates. - Target (y): future stock price or return.

2.1.2 Classification

Classification is used to predict a categorical output, often a binary variable. A common model for binary classification is the **logistic regression** model. Despite its name, logistic regression is a classification algorithm, not a regression one.

The model predicts the probability $P(y = 1|x)$ of the output being class 1 (e.g., "default" or "non-default") given the input features X. The logistic regression model assumes the following relationship between the input features and the probability:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

The output is a probability, and a threshold (e.g., 0.5) is applied to decide the class label.

Example in Finance: - *Credit risk classification:* Logistic regression can be used to classify whether a loan applicant is likely to default based on their financial history. - Features (X): credit score, income, debt-to-income ratio, previous defaults. - Target (y): whether the applicant defaults (yes/no).

2.2 Unsupervised Learning

Unsupervised learning occurs without labeled data, and the model must find hidden structures in the data. Common tasks include clustering and dimensionality reduction.

2.2.1 Clustering

Clustering is a technique for grouping data points into clusters based on their similarities. One of the most common clustering algorithms is **K-Means**, where k is the number of clusters to find. The algorithm works by minimizing the sum of the squared distances between each data point and the centroid of its cluster.

The objective function of K-Means is:

$$\text{Minimize:} \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

where C_i is the i-th cluster, and μ_i is the centroid of the cluster.

Example in Finance: - *Grouping stocks based on performance or risk characteristics:* K-Means clustering can be used to group stocks or other financial assets that exhibit similar performance or risk patterns. - Features (X): historical returns, volatility, market capitalization, industry sector. - Output: clusters of stocks with similar characteristics.

2.2.2 Dimensionality Reduction

Dimensionality reduction is used to reduce the number of input features while preserving as much information as possible. A popular technique is **Principal Component Analysis** (PCA), which transforms the data into a set of orthogonal components that capture the most variance in the data.

Example in Finance: - *Reducing the number of factors for portfolio management:* PCA can be used to reduce a large number of economic and financial indicators into a smaller set of principal components, which can help in understanding the key factors driving asset returns. - Features (X): returns of individual assets, economic factors (e.g., GDP growth, inflation). - Output: principal components representing the main drivers of portfolio risk and return.

2.3 Reinforcement Learning

Reinforcement learning (RL) is a learning paradigm where an agent interacts with an environment and learns to take actions that maximize a cumulative reward over time. The agent receives feedback in the form of rewards or penalties based on the actions it takes. RL is commonly used in decision-making problems such as trading and portfolio optimization.

A common algorithm in RL is **Q-Learning**, which seeks to learn a policy $\pi(s)$ that maps each state s to the optimal action a that maximizes future rewards. The agent learns a **Q-function** that estimates the expected cumulative reward for each action in each state:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

where $Q(s,a)$ is the value of taking action a in state s, r is the reward, γ is the discount factor, and α is the learning rate.

Example in Finance: - *Algorithmic trading:* A reinforcement learning agent can learn an optimal trading strategy by buying, selling, or holding based on market conditions to maximize cumulative profits. - State (s): current market conditions (e.g., stock price, volume, volatility). - Action (a): buy, sell, or hold. - Reward (r): profit or loss from the trade.

3 Learning

Learning = Representation + Evaluation + Optimization

3.1 Representation (Hypothesis space)

The model is defined by its **parameters** (weights and biases). Different representations include decision trees, neural networks, SVMs, etc.

3.2 Evaluation (Loss function)

The **loss function** measures the quality of the model's predictions. For regression, the **MSE** is often used:

$$L(w) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

3.3 Optimization

The goal is to minimize the loss function using optimization techniques like **gradient descent**:

$$w \leftarrow w - \eta\frac{\partial L(w)}{\partial w}$$

where η is the learning rate.

4 Neural Networks

4.1 Representation

Neural networks consist of multiple layers of interconnected neurons. Each layer applies a transformation to the input it receives, and the output is passed to the next layer. These layers include:

- **Input layer**: Takes the input data (features) for the model.
- **Hidden layer(s)**: Applies transformations using learned weights and biases. The network can have multiple hidden layers.
- **Output layer**: Produces the final output (e.g., classification result, predicted value).

The structure of a basic feedforward neural network is illustrated below:
In this diagram:

- The green nodes represent the input layer.
- The blue nodes represent the hidden layer.
- The red node represents the output layer.

Each connection between nodes is associated with a weight, and each node applies a function (e.g., ReLU, Sigmoid) to the weighted sum of its inputs before passing the result to the next layer.

The mathematical formulation for a single neuron is:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

Where z is the output of the neuron, w_i are the weights, x_i are the inputs, and b is the bias.

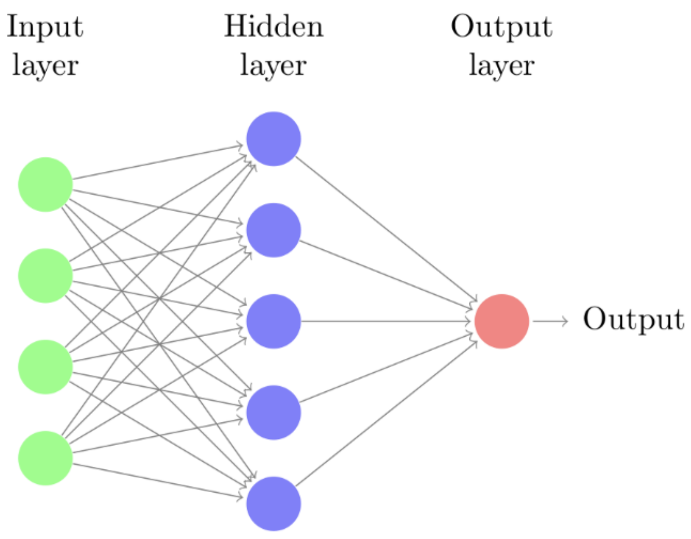

Figure 1: Neural network with one hidden layer.

4.2 Evaluation

The loss function is calculated to evaluate the quality of predictions. For regression, it is the **MSE**, and for classification, it is often the **cross-entropy**.

4.3 Optimization

The optimization algorithm adjusts the weights to minimize the loss function. **Gradient descent** is often used to do this.

5 Overfitting and Underfitting

5.1 Overfitting

Overfitting occurs when the model is too complex and fits the training data too well, leading to poor performance on new data.

Solutions:

- Use **regularization** techniques like L2:

$$L(w) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{n} w_j^2$$

- Collect more data.

5.2 Underfitting

Underfitting occurs when the model is too simple and fails to capture the complexity of the data.

Solution: Use a more complex model or add **features**.

6 Model Selection

6.1 Data splitting

The data should be divided into **training**, **validation**, and **test** sets.

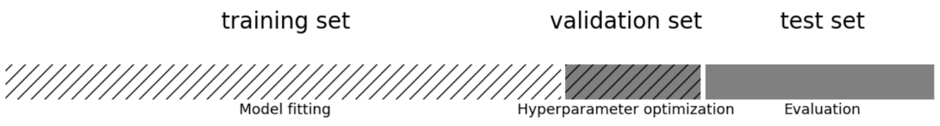

Figure 2: Data splitting.

6.2 Cross-validation

To avoid **data leakage** and get a robust evaluation, use **cross-validation**.

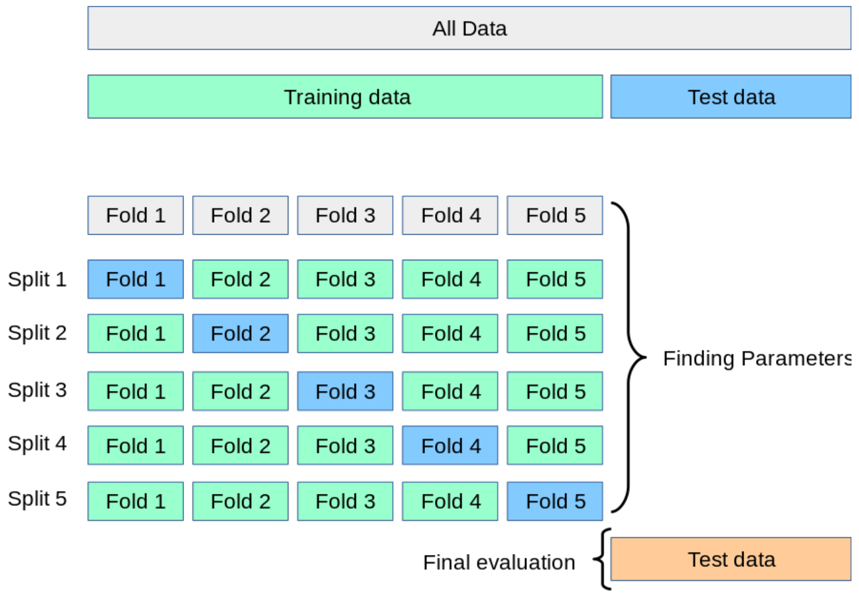

Figure 3: Cross-Validation.

6.3 Hyperparameter optimization

Once hyperparameters are optimized, the model should be evaluated on the test set for reliable results.

7 Feature Engineering

Feature engineering involves transforming raw data into a format suitable for machine learning.
Common techniques:

- **Feature selection**: Choosing the most important variables.
- **Dimensionality reduction**: Using techniques like **PCA**.
- **Encoding**: Transforming categorical variables into numerical ones via methods like **one-hot encoding**.

Importance: Good **feature engineering** can significantly improve a model's performance.