# Numerical Methods for Mathematical Finance

## Lecture 4: Energy Markets and Machine Learning for Finance

Matteo Garbelli

University of Verona

November 26, 2024

# Today:

1. Introduction to Energy Markets

2. Machine Learning for Finance
   - Supervised Learning
   - ML Application 1: SL for Portfolio Optimization
   - Reinforcement Learning
   - ML Application 2: RL for Bidding Strategy Optimization in Electricity Market

# Aims

1. How do Energy Markets work?
2. What is Machine Learning?
3. Why Machine Learning in Finance?

# Presentation Outline

# What are Energy Markets?

Energy markets deal with the buying and selling of energy products like electricity, natural gas, oil, and renewable energy.

Differently form finance:

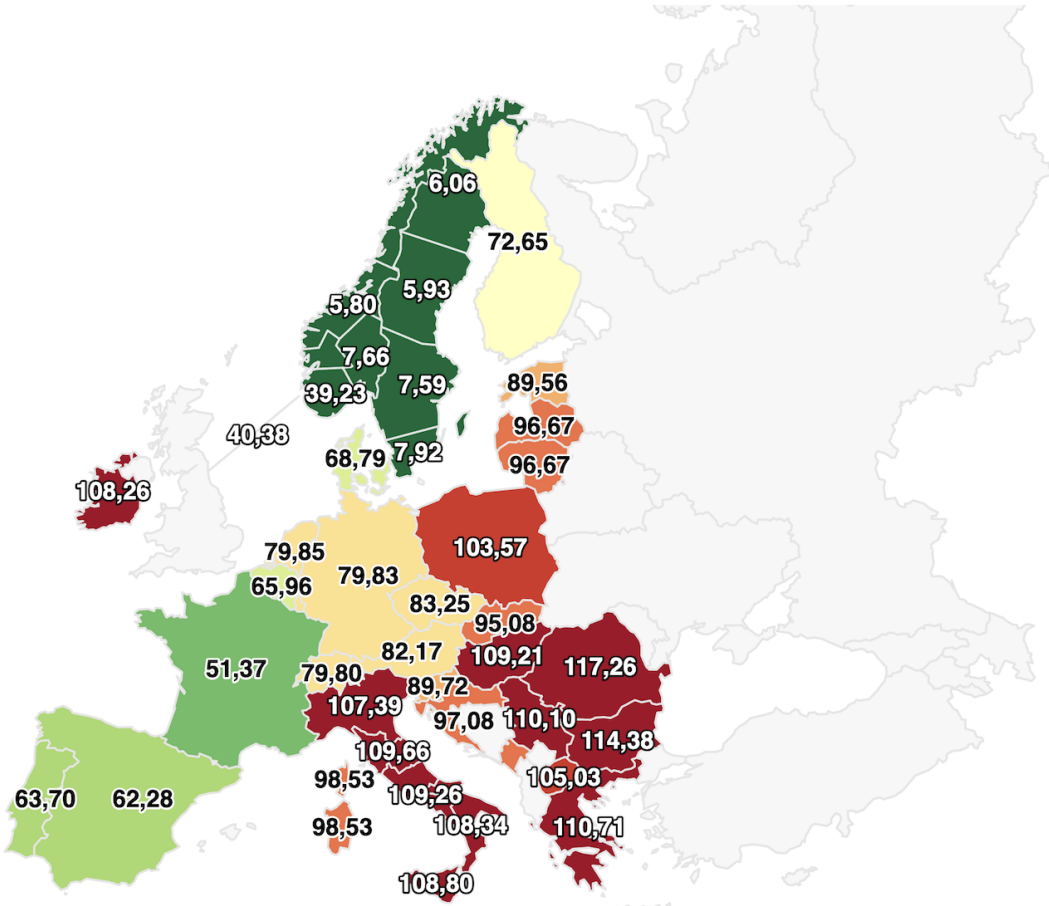
- High price volatility due to supply-demand imbalances.
- Regional markets with specific regulations (e.g., North America, Europe, Asia).

But still:

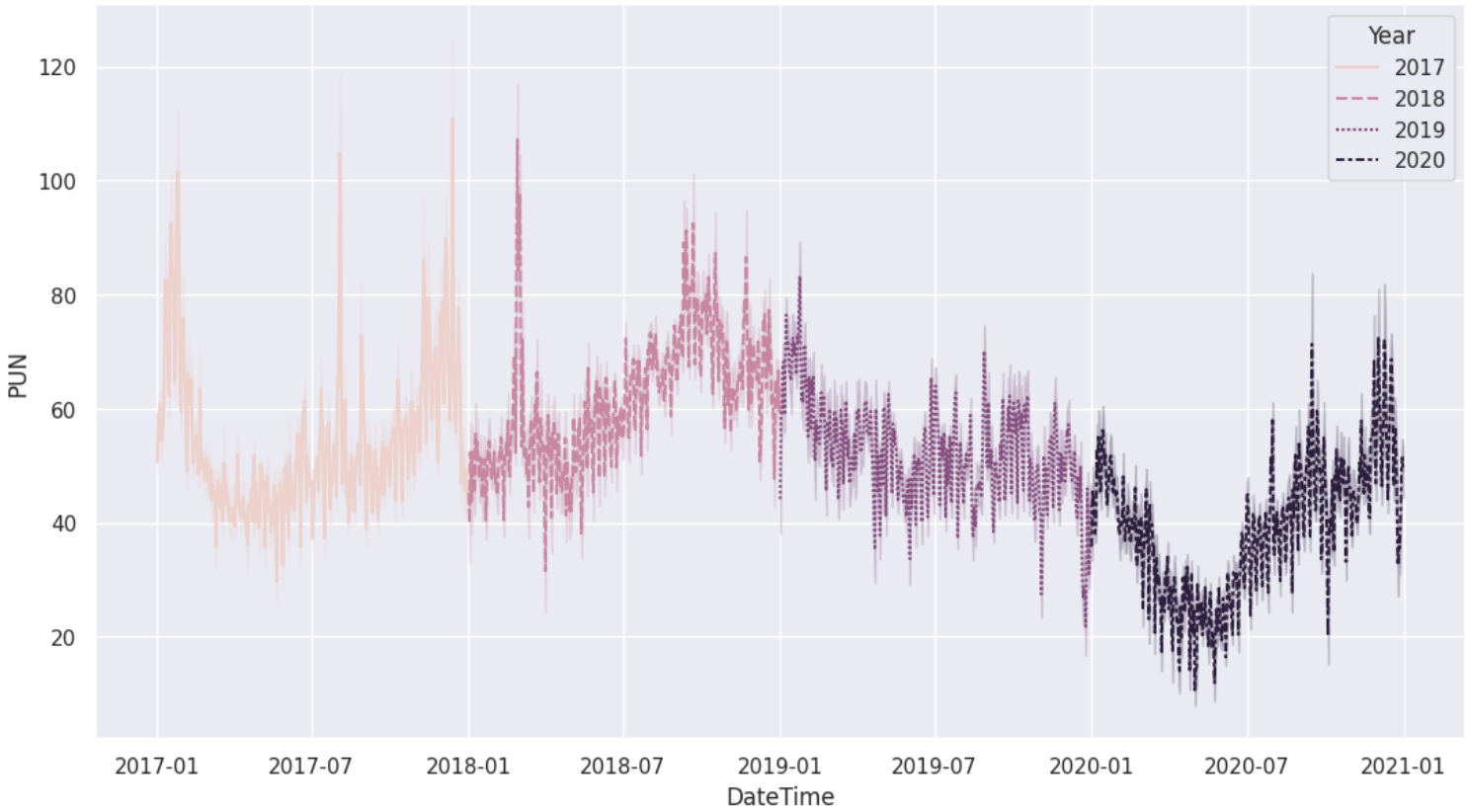
- Pricing and hedging of energy derivatives.
- Risk management for energy portfolios.
- Forecasting price and demand trends.

# Electricity Prices

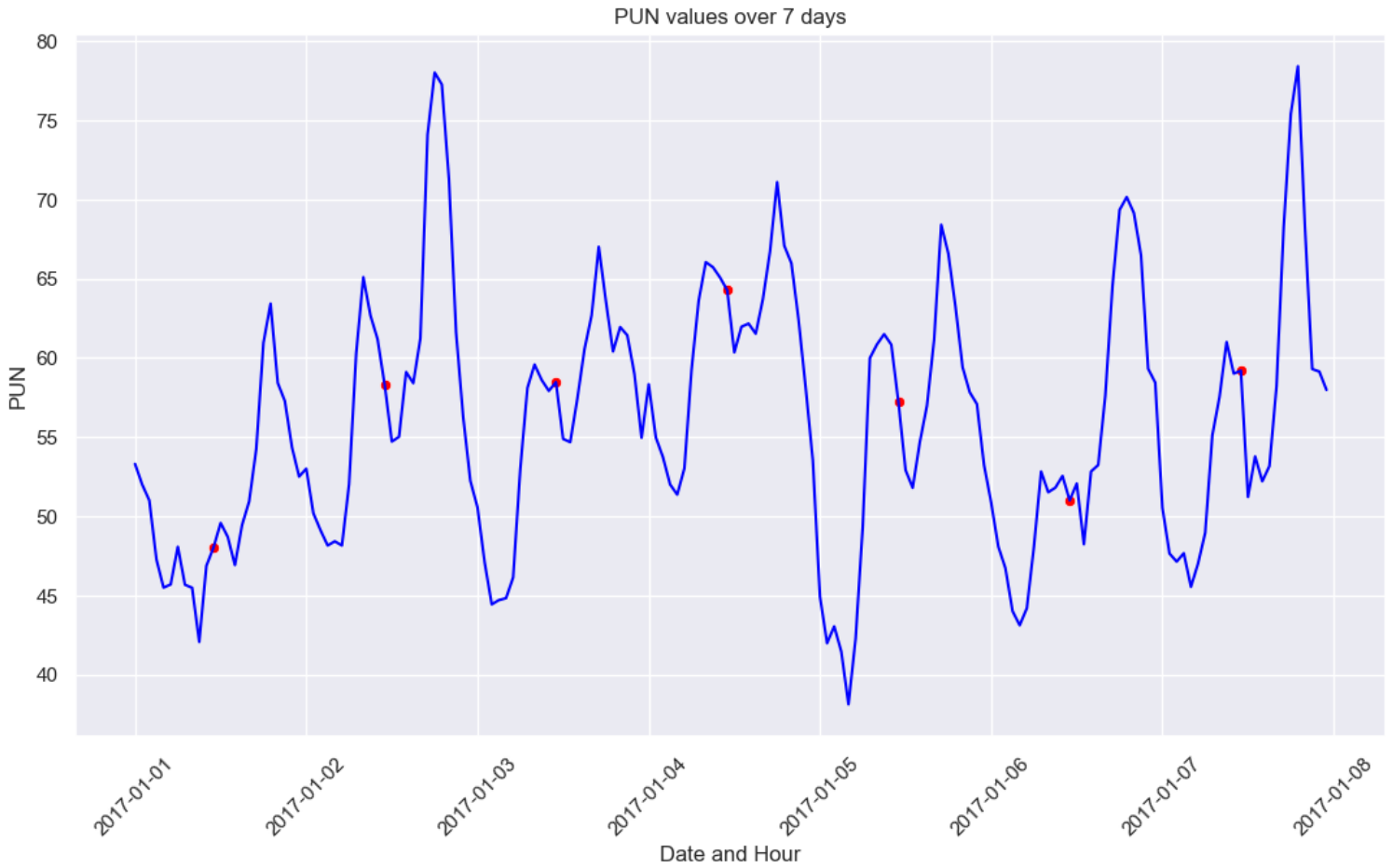
Average prices of the electricity spot market (day-ahead) in week 37 of 2024 in EUR/MWh



# 4 years PUN


Year
2017
2018
2019
2020
PUN
120
100
80
60
40
20
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01
DateTime

# 7 days PUN


PUN values over 7 days
PUN
80
75
70
65
60
55
50
45
40
2017-01-01
2017-01-02
2017-01-03
2017-01-04
2017-01-05
2017-01-06
2017-01-07
2017-01-08
Date and Hour

# Day-Ahead Markets

Day-ahead markets allow market participants to trade energy for delivery on the following day. Prices are determined through an auction process based on supply and demand forecasts.

- Trades are made in hourly intervals
- Provides a reference price for electricity, often used as a benchmark.
- Relies heavily on accurate demand and generation forecasts.

**Pros:**

- Helps stabilize the grid by enabling early scheduling of power generation.
- Allows participants to hedge against price fluctuations in real-time markets.

# Intraday Markets

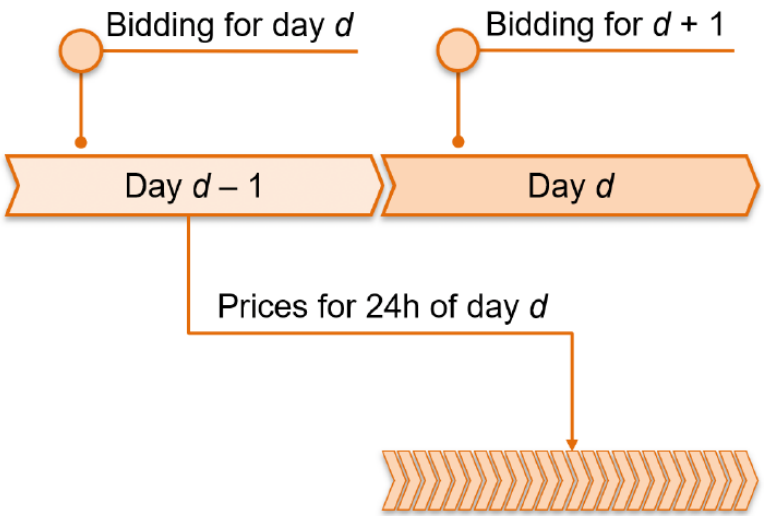
Intraday markets allow participants to trade energy close to real time, with delivery schedules as short as 15 minutes ahead. Used to balance unforeseen imbalances in the grid caused by forecasting errors or unexpected demand/supply changes.

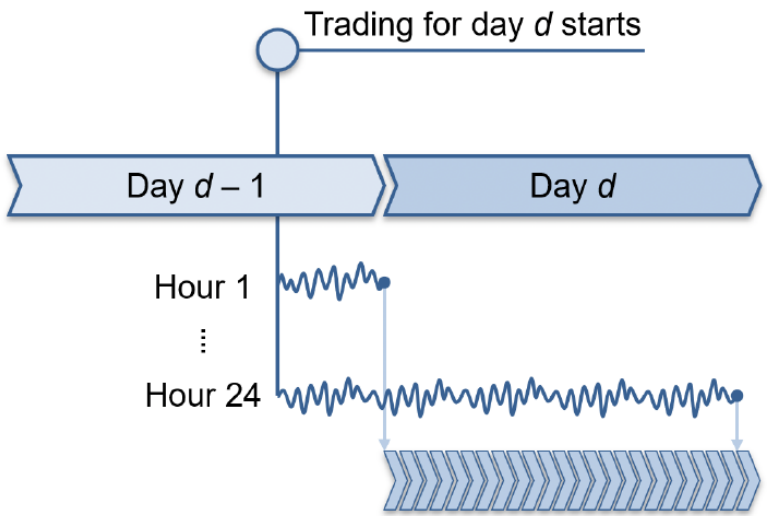
- Trades occur continuously or in auctions, depending on the market design.
- Prices tend to be more volatile than in the day-ahead market.
- Often influenced by weather, renewable energy variability, and grid constraints.

**Pros:**

- Increases flexibility for grid operators and participants.
- Essential for balancing markets, especially in systems with high renewable energy penetration.

# Day–Ahead vs Intraday



# The EUPHEMIA Algorithm

EUPHEMIA (Pan-European Hybrid Electricity Market Integration Algorithm) is the optimization algorithm used to calculate market prices in Europe's day-ahead market.

**Inputs:**

- Supply and demand bids from participants in all connected markets.
- Transmission capacities between countries or regions.

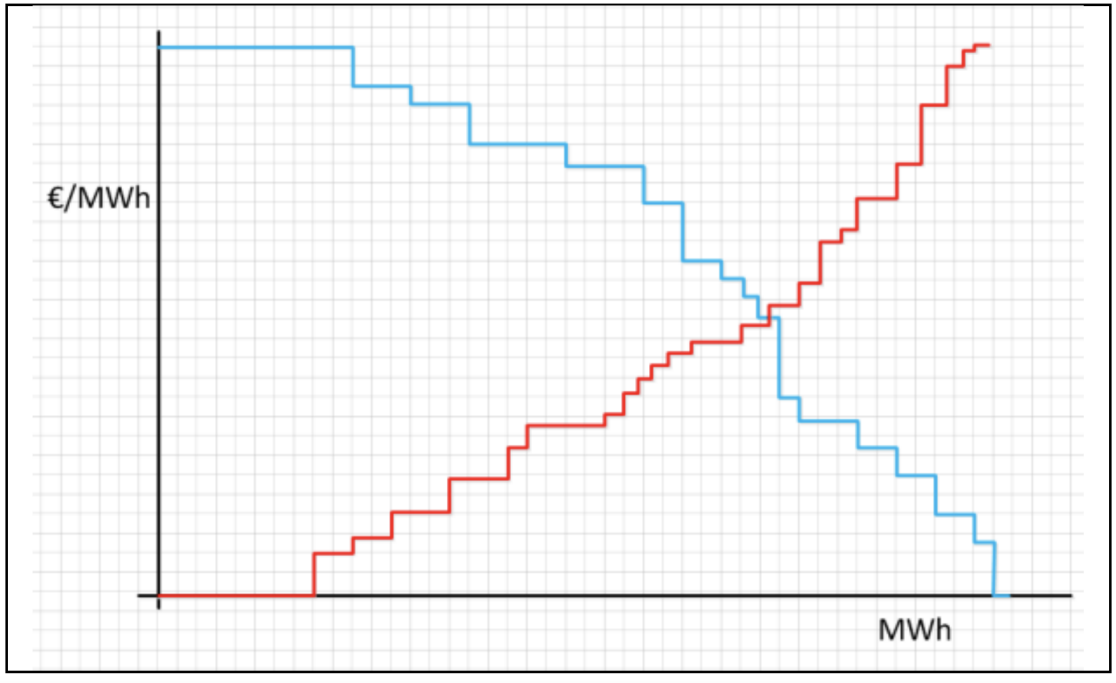
**Outputs:**

- Hourly market prices for each region.
- Scheduled cross-border electricity flows.

**Impact:**

- Increases market efficiency and liquidity.
- Facilitates integration of renewable energy sources by optimizing transmission capacity usage.

# Intersection of supply and demand curves




- Supply Curve: Represents the available quantity of electricity at various prices (€/MWh).
- Demand Curve: Represents the quantity of electricity demanded at different price levels.

# Price Formation Mechanism

- The Market-Clearing Price (**MCP**) is determined where the two curves intersect, indicating the equilibrium price and quantity.
- **Market Equilibrium**: At this point, the price balances supply and demand, resulting in efficient allocation.
- **Stepwise Structure**: The stepwise nature of the curves reflects the aggregated offers from multiple suppliers and buyers, characteristic of energy markets with discrete price-quantity bids.

# Use Cases for Data-Driven Algorithms in Energy Markets

1. **Price Forecasting:**
   - Build models to predict future energy prices using historical data.
   - Incorporate seasonality and market-specific factors.
2. **Risk Management:**
   - Use Monte Carlo simulations to assess portfolio risk.
3. **Demand Response Optimization:**
   - Predict consumer demand using machine learning techniques.
4. **Market Arbitrage:**
   - Identify arbitrage opportunities between spot and futures markets.

# Pubblic Available Datasets

- For the Italian market: Gestore dei Mercati Energetici `https://www.mercatoelettrico.org/`
- Greenhouse Gas Emissions from Energy Annual time series of GHG emissions from energy, a major source of anthropogenic emissions.`https://www.iea.org/data-and-statistics/data-product/greenhouse-gas-emissions-from-energy`
- Open energy information, data and resources `https://openei.org/wiki/Main_Page`

# Key Takeaways

- Energy markets are a vital area in mathematical finance with unique challenges and opportunities.
- Data-driven approaches are essential for building predictive and risk-management tools.
- Use the suggested data sources to explore real-world applications.
- **What can you do?**
  - Explore the suggested data repositories.
  - Implement simple forecasting models using historical data.
  - Investigate seasonality and volatility patterns.

# Presentation Outline

1. Introduction to Energy Markets

2. Machine Learning for Finance
   - Supervised Learning
   - ML Application 1: SL for Portfolio Optimization
   - Reinforcement Learning
   - ML Application 2: RL for Bidding Strategy Optimization in Electricity Market

# Data-Driven vs Model-Driven Algorithms

**Data-Driven**: Rely heavily on data to uncover patterns and make predictions.
*Examples*: Machine learning

- ✅ Adapt well to complex, nonlinear relationships.
- ✅ Effective for tasks with large datasets.
- ❌ Require substantial data to perform well.
- ❌ Often act as "black boxes" with limited interpretability.

# Data-Driven vs Model-Driven Algorithms

**Data-Driven**: Rely heavily on data to uncover patterns and make predictions. *Examples*: Machine learning

- ✅ Adapt well to complex, nonlinear relationships.
- ✅ Effective for tasks with large datasets.
- ❌ Require substantial data to perform well.
- ❌ Often act as "black boxes" with limited interpretability.

**Model-Driven**: Based on predefined mathematical or physical models. *Examples*: Physics-based modeling, control systems.

- ✅ Often interpretable and grounded in domain knowledge.
- ✅ Useful when theoretical understanding is strong.
- ❌ May struggle with high-dimensional data.
- ❌ Limited in capturing complex patterns without adjustment.

# Introduction to Machine Learning in Finance

## What is Machine Learning?

- A subset of artificial intelligence focused on building systems that learn from data to make predictions or decisions.
- Relies on algorithms that identify patterns and relationships in large datasets.

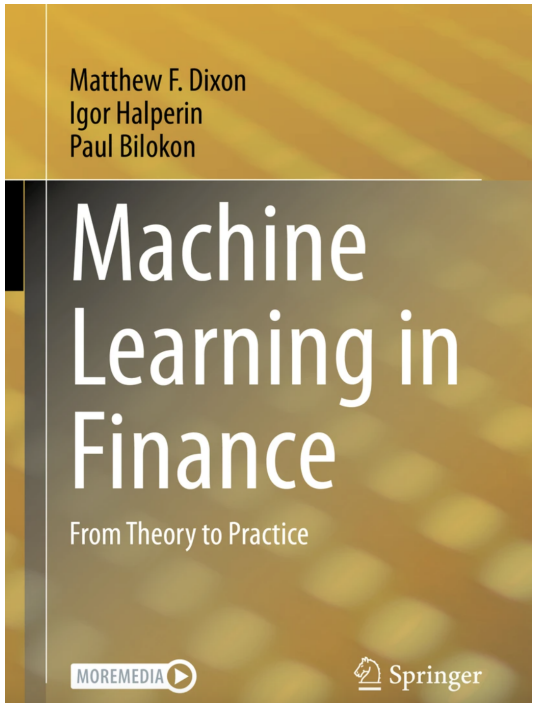
## Why Machine Learning in Finance?

- Financial markets generate massive amounts of data daily.
- ML excels at handling high-dimensional and non-linear problems.
- Key applications in finance:
  1. Risk management and fraud detection.
  2. Algorithmic trading and portfolio optimization.
  3. Price prediction and market forecasting.

# Reference book

**Machine Learning in Finance: From Theory to Practice**
by Matthew F. Dixon, Igor Halperin, Paul Bilokon. Springer (2020)

`https://link.springer.com/book/10.1007/978-3-030-41068-1`



# Machine Learning Paradigms

1. **Supervised Learning:**
   - The algorithm learns a mapping from input (features) to output (labels).
   - Data: Labeled dataset with input-output pairs.
   - Applications: Price prediction, credit scoring, fraud detection.

# Machine Learning Paradigms

1. **Supervised Learning:**
   - The algorithm learns a mapping from input (features) to output (labels).
   - Data: Labeled dataset with input-output pairs.
   - Applications: Price prediction, credit scoring, fraud detection.
2. **Unsupervised Learning:**
   - The algorithm finds patterns or structures in data without labeled outputs.
   - Data: Unlabeled dataset.
   - Applications: Customer segmentation, anomaly detection, clustering.

# Machine Learning Paradigms

1. **Supervised Learning:**
   - The algorithm learns a mapping from input (features) to output (labels).
   - Data: Labeled dataset with input-output pairs.
   - Applications: Price prediction, credit scoring, fraud detection.
2. **Unsupervised Learning:**
   - The algorithm finds patterns or structures in data without labeled outputs.
   - Data: Unlabeled dataset.
   - Applications: Customer segmentation, anomaly detection, clustering.
3. **Reinforcement Learning:**
   - The algorithm learns to make decisions by interacting with an environment.
   - Data: Sequential actions, rewards, and states.
   - Applications: Algorithmic trading, portfolio optimization, energy market bidding.

# Presentation Outline

# Supervised Learning

In supervised learning, we are given **labeled data**, consisting of pairs:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), \quad \text{where } x_i \in X,\ y_i \in Y.$$

The goal is to learn the underlying **relationship** between the input space $X$ (features) and the output space $Y$ (labels or responses).

Each observation $x_i$ is referred to as a **feature vector**, which represents the input variables, and $y_i$ is the **label** or **response**, representing the target output.

The learned relationship can then be used to predict the response $y$ for new, unseen feature vectors $x$.

# Neural Networks in Financial Prediction

Neural networks approximate non-linear functions using multiple layers.

- **Types of Networks**:
  - Feedforward Network: $Y = f_\theta(X)$
  - Recurrent Network such as Long Short-Term Memory (LSTM)
- **Activation Functions**: Allows for non-linear transformations:

$$Y(X) = \sigma(WX + b)$$

where $\sigma$ is an activation function, such as ReLU or $\tanh$.

# The Universal Approximation Property

The Universal Approximation Theorem states that:

> *A feedforward neural network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$, given an appropriate activation function.*

# Presentation Outline

# Problem Statement and Data Collection

- **Objective:** Minimize the risk (volatility) of a portfolio.
- **Supervised Learning Framework:**
  - Inputs: Historical asset returns (rolling window).
  - Outputs: Optimal portfolio weights.
- 1. Simulate GBMs;
  2. Use `yfinance` to fetch real stock prices (e.g., AAPL, MSFT, GOOGL, AMZN, TSLA).
- Compute daily logarithmic returns:

$$R_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

# Portfolio Optimization: Problem Setting

The portfolio consists of $N$ risky assets with returns $r_1, r_2, \ldots, r_N$ and one risk-free asset with interest rate $r_f$.

Portfolio weights are $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$, where $\pi_0$ is the proportion in the risk-free asset and $\pi_i$ $(i \geq 1)$ are proportions in risky assets.

The weights satisfy $\sum_{i=0}^{N} \pi_i = 1$.

Objective: Minimize risk while achieving a target volatility $\sigma_\Pi$ for the portfolio.

# The Optimization Problem for $\pi_i$

Targets $\pi^*$ are computed as:

$$\min_{\pi} \quad \sqrt{\pi^\top \Sigma \pi},$$

subject to:

$$\sqrt{\pi^\top \Sigma \pi} \leq \sigma_{\text{target}},$$

$$\sum_{i=1}^{N} \pi_i = 1, \qquad \pi_i \geq 0, \quad \forall i.$$

- $\Sigma$: Covariance matrix of asset returns.
- $\sigma_{\text{target}}$: Target portfolio volatility level.
- $\pi$: Portfolio weights to be optimized.

**Target:** Optimal weights $\pi^*$ minimizing risk while ensuring the portfolio volatility does not exceed the $\sigma_{\text{target}}$.

# Supervised Learning Scheme

Objective: Optimize portfolio allocation using a neural network model.

1. Data Acquisition and Preprocessing
2. Feature Engineering $\rightarrow$ Rolling window (e.g., 60 days):
   - Compute covariance matrix of returns for each window and optimize to find weights.
3. Model Definition and Training in a Supervised Learning Perspective:
   - Inputs: Historical returns (features).
   - Outputs: Optimal weights (target).
4. Performance Comparison:
   - Optimized portfolio vs. equally weighted portfolio.

# Step 1: Data Acquisition

1. Simulations of 5 geometric Brownian motions;
2. 
   - Download historical stock data from Yahoo Finance.
   - Selected tickers: AAPL, MSFT, GOOGL, AMZN, TSLA
   - Calculate daily returns from the adjusted closing prices.

# Step 2: Feature Engineering

- Use a rolling window to extract relevant features for training.
- Features extracted:
  - Expected Returns (mean over window).
  - Volatility (standard deviation).
  - Momentum (average return of the last 30 days).
- Prepare training and testing datasets.

# Step 3: Model Training

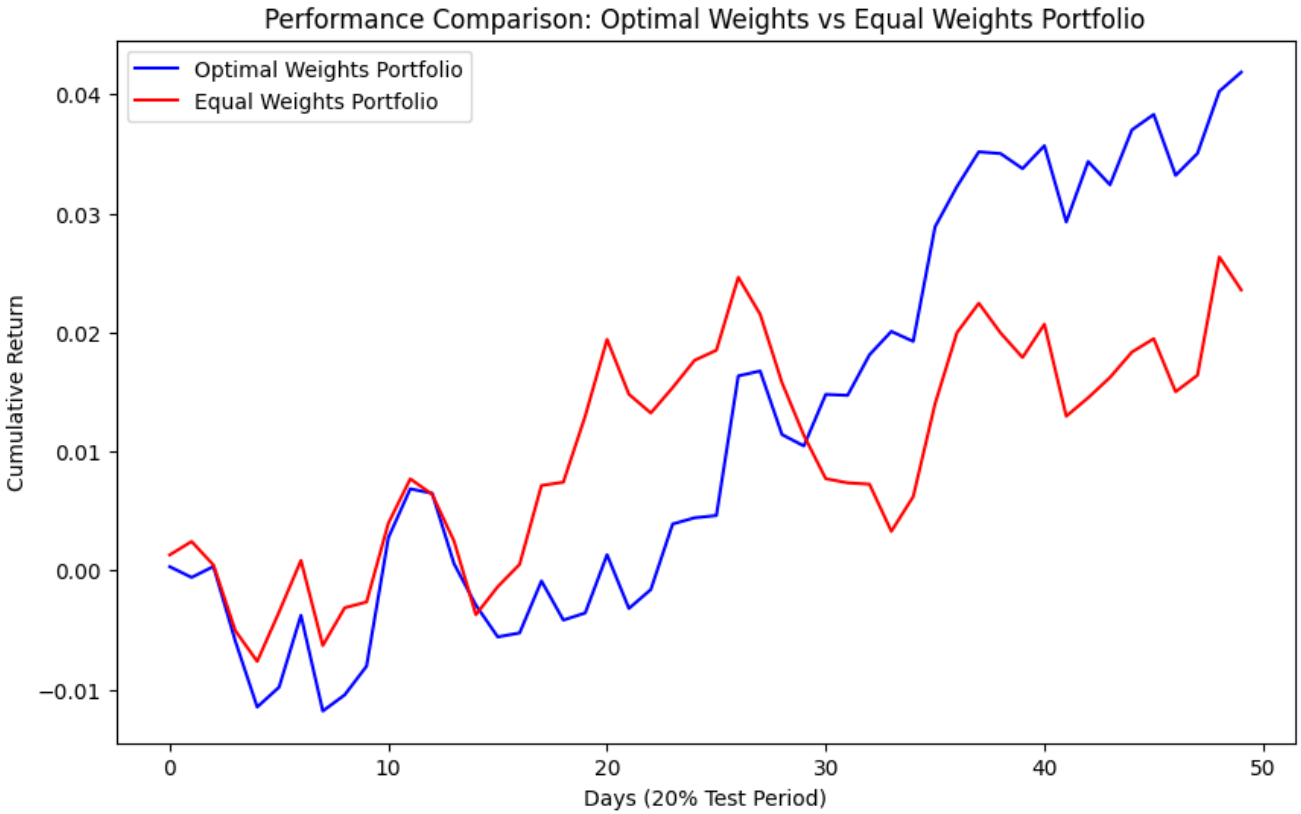
- Define a neural network to predict asset allocation.
- Features represent expected prices, volatility, and momentum.
- Train the network to predict optimal weights for each asset.

# Step 4: Model Evaluation

- Use the trained model to predict portfolio weights on test data.
- Compare the performance of the optimized portfolio with:
  - Equal-weighted portfolio as a benchmark.
- Calculate cumulative returns over the test period.

# Performance Comparison

- Visualize the performance of the optimized portfolio against an equal-weighted portfolio.
- The comparison highlights the effectiveness of the learned allocation strategy.



# Presentation Outline

# Reinforcement Learning

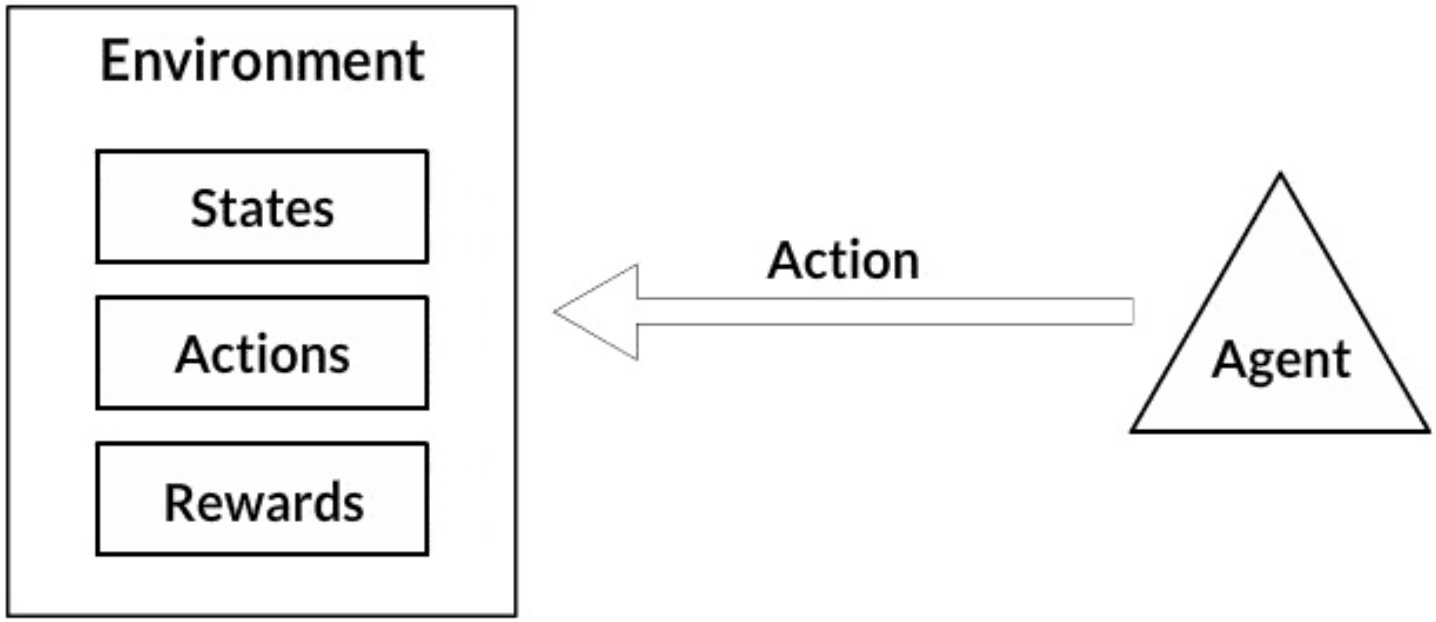
RL is a learning paradigm where a numerical reward signal is maximized through repeated experience by interactions with the environment.



The agent's objective is to develop a strategy that maximizes the expected cumulative reward by learning a policy that maps states to actions.

# Reinforcement Learning

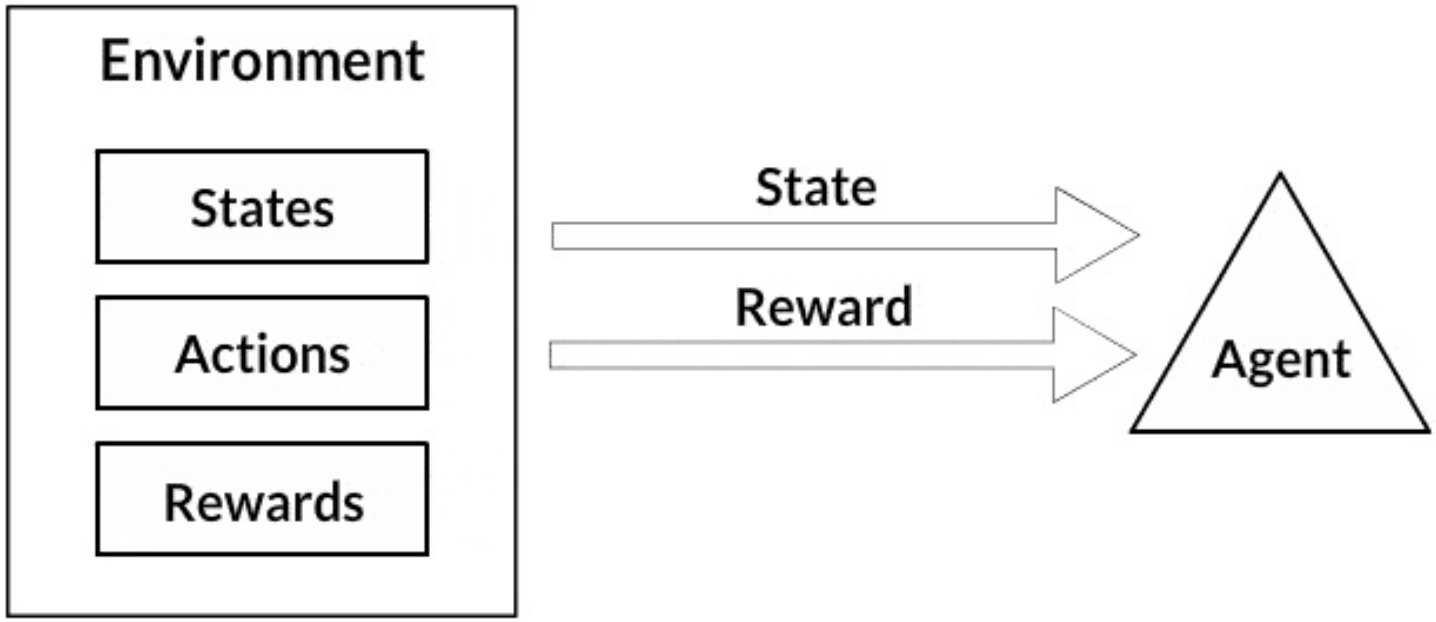
RL is a learning paradigm where a numerical reward signal is maximized through repeated experience by interactions with the environment.




The agent's objective is to develop a strategy that maximizes the expected cumulative reward by learning a policy that maps states to actions.

# Reinforcement Learning

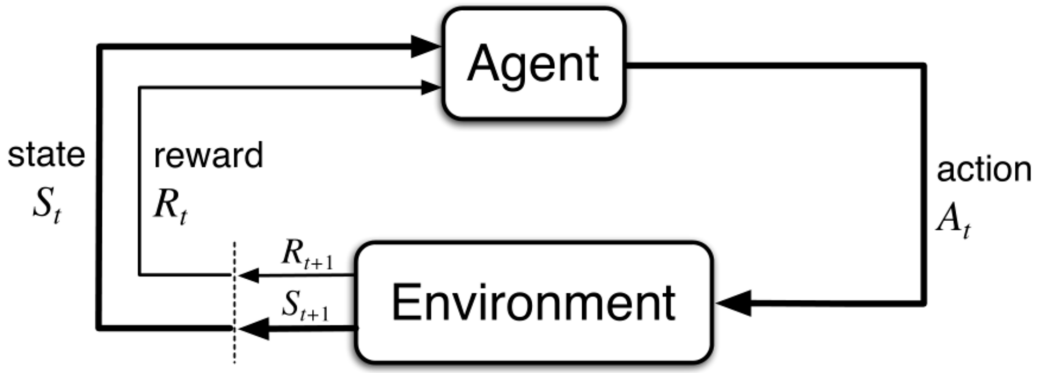
RL is a learning paradigm where a numerical reward signal is maximized through repeated experience by interactions with the environment.




The agent's objective is to develop a strategy that maximizes the expected cumulative reward by learning a policy that maps states to actions.

# RL Framework

- **Agent**: Learns and makes decisions.
- **Environment**: Provides feedback in the form of rewards or penalties.
- **Reward Function** $r(s, a)$: Defines the reward received after taking action $a$ in state $s$.
- **Policy** $\pi(a|s)$: Strategy that defines the action $a$ to take in state $s$.



# Markov Decision Process

RL problems are often modeled as MDPs, where the goal is to find the optimal policy $\pi^*$ that maximizes cumulative reward.

- a set of environment and agent states, $S$;
- a set of actions, A, of the agent;
- $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability of transition (at time $t$ from state $s$ to state $s'$ under action $a$
- $r_a(s, s')$ is the immediate reward after transition from $s$ to $s'$ with action $a$

# The Value Function

A basic RL agent interacts with its environment in discrete time steps.

At each time t, the agent receives the current state $s_t$ and reward $r_t$. It then chooses an action $a_t$ from the set of available actions, which is subsequently sent to the environment. The environment moves to a new state $s_{t+1}$ and the reward $r_{t+1}$ associated with the transition $(s_t, a_t, s_{t+1})$ is determined.

The goal of a RL agent is to learn a policy: $\pi : A \times S \to [0, 1] \subset P(A)$

$$\pi(a, s) = \Pr(a_t = a \mid s_t = s)$$

which outputs a strategy profile by maximizinig the expected cumulative reward.

# RL in Finance: Problem Formulation

The **Return Value** $R_t$ represents the long-term profit

$$R_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{T-t-1} r_T = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1}$$

with discount factor $\gamma$.

# RL in Finance: Problem Formulation

The **Return Value** $R_t$ represents the long-term profit

$$R_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{T-t-1} r_T = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1}$$

with discount factor $\gamma$.

The agent aims to optimize an **Objective** depending on $R_t$

$$J = \mathbb{E}_{a_t \sim \pi}[R_t],$$

that corresponds to learning a policy that maximizes the cumulative future payoff to be received starting from any given time $t$ until $T$.

# RL in Finance: Problem Formulation

The **Return Value** $R_t$ represents the long-term profit

$$R_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{T-t-1} r_T = \sum_{i=0}^{T-t-1} \gamma^i r_{t+i+1}$$

with discount factor $\gamma$.

The agent aims to optimize an **Objective** depending on $R_t$

$$J = \mathbb{E}_{a_t \sim \pi}[R_t],$$

that corresponds to learning a policy that maximizes the cumulative future payoff to be received starting from any given time $t$ until $T$.

The **Value Function** associated with such a control problem is

$$V_\pi(s_t) = \max_{a_t \in A} \mathbb{E}_\pi \left[ R(s_t, a_t, s_{t+1}) \right].$$

# The Bellman Equation

The value function $V_\pi(s)$ satisfies (recursively) the Bellman equation:

$$V_\pi(s) = \mathbb{E}_\pi \Big[ r_{t+1} + \gamma \cdot V_\pi(S_{t+1}) \;\Big|\; S_t = s \Big],$$

where:

- $r_{t+1}$: Immediate reward after taking an action.
- $\gamma$: Discount factor $(0 \leq \gamma \leq 1)$ for future rewards.
- $S_{t+1}$: Next state reached after taking an action in state $S_t$.

# Reinforcement Learning Applications in Finance

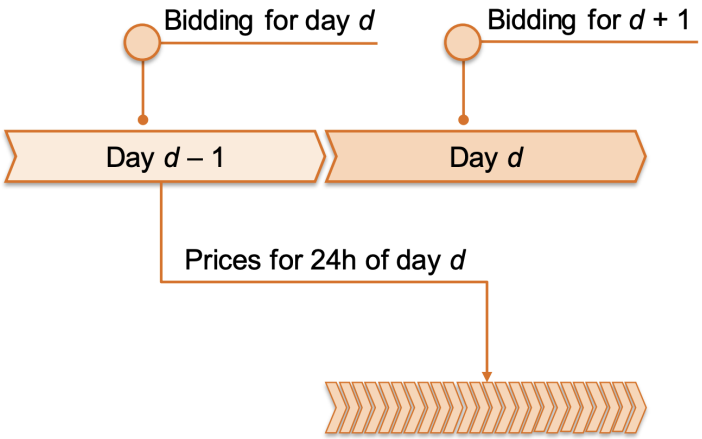
- **Algorithmic Trading:**
  - Optimizing trading strategies through sequential decision-making.
  - Example: A Deep Q-Network for dynamic portfolio allocation.
- **Portfolio Optimization:**
  - Balancing risk and return in portfolio management.
  - Dynamic rebalancing of asset allocations.
- **Option Pricing and Hedging:**
  - Learning optimal pricing strategies in incomplete markets.
  - Hedging dynamically in response to market conditions.
- **Energy Markets:**
  - Optimizing bidding strategies in intraday or day-ahead electricity markets.

# Presentation Outline

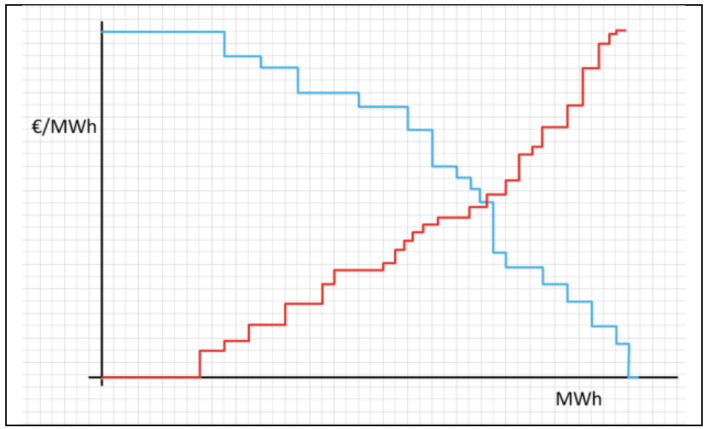
# Introduction to the Problem

- **Context**: Day-ahead electricity markets
  - Buyers and sellers submit bids for next-day delivery.
  - Prices determined by the intersection of supply and demand curves.
- **Objectives**:
  - Optimize bidding strategy for electricity suppliers.
  - Maximize profit using historical data and market feedback.
- **Challenge**: High-dimensional, continuous action space.

# The European Price Formation Mechanism



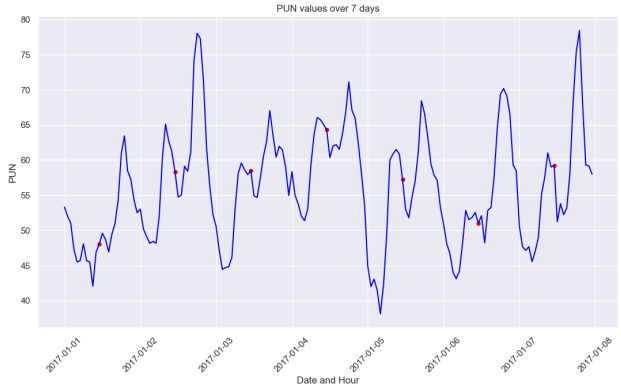

Bidding settlement in day-ahead auctions




Demand/Offer aggregated curve

# RL for an Electricity Supplier

State: Italian hourly PUN from 01-Jan-2017 to 12-Dec-2020




Action: energy/price curve

# RL for electricity market

- The seller selects and executes an action $a_t \in \mathcal{A}$;
- The system progresses from state $s_t$ under the action $a_t$ to $s_{t+1}$;
- the agent receives reward $r_{t+1}(s_t, a_t)$ based on the obtained profit when taking the offering curve $a_t$ in state $s_t$.

# RL for electricity market

- The seller selects and executes an action $a_t \in \mathcal{A}$;
- The system progresses from state $s_t$ under the action $a_t$ to $s_{t+1}$;
- the agent receives reward $r_{t+1}(s_t, a_t)$ based on the obtained profit when taking the offering curve $a_t$ in state $s_t$.

Agent goal: learn the best policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ to maximize its expected performance over the long term,

$$R_t = r_{t+1}(s_t, a_t) + \sum_{i=t+1}^{T-1} \gamma^i r_{i+1}(s_i, a_i)$$

with discount factor $\gamma \in [0, 1]$.

# Bellman Equation

The agent's value function is

$$V(s_n) = \max_{a_n \in \mathcal{A}} \mathbb{E}\left[R(s_n, a_n, s_{n+1})\right]$$

The DPP implies that $V$ satisfies a Bellman equation

$$V(s_n) = \max_{a_n \in \mathcal{A}} \mathbb{E}\left[r(s_n, a_n, s_{n+1}) + \gamma V(s_{n+1})\right] \tag{1}$$

# Bellman Equation

The agent's value function is

$$V(s_n) = \max_{a_n \in \mathcal{A}} \mathbb{E}\left[R(s_n, a_n, s_{n+1})\right]$$

The DPP implies that $V$ satisfies a Bellman equation

$$V(s_n) = \max_{a_n \in \mathcal{A}} \mathbb{E}\left[r(s_n, a_n, s_{n+1}) + \gamma V(s_{n+1})\right] \tag{1}$$

We define the Q-function $Q$, an action–valued function to measure the "quality" of taking a specific action

$$Q(s_n, a_n) = R(s_n, a_n) + \gamma \max_{a} Q(s_{n+1}, a)$$

that solves the following Bellman equation

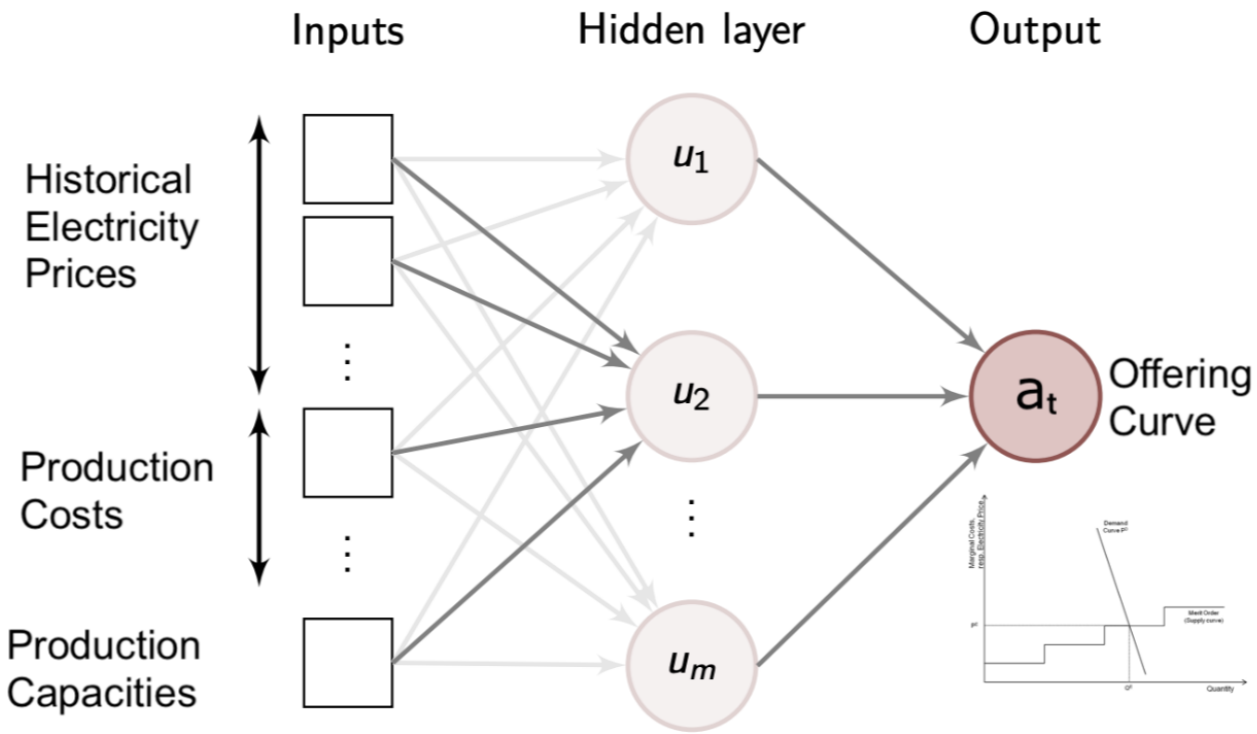
$$Q(s_n, a_n) = R(s_n, a_n) + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1})$$

# Deep Deterministic Policy Gradient

- DDPG is an off-policy algorithm that uses a NN to approximate the policy and another NN to approximate the value function.
- We employ NNs to approximate policy $\pi$ and value function V that solves Eq. (1) in a high dimensional, continuous actions space.
- We apply DDPG to learn optimal bidding strategies

# The Actor Network

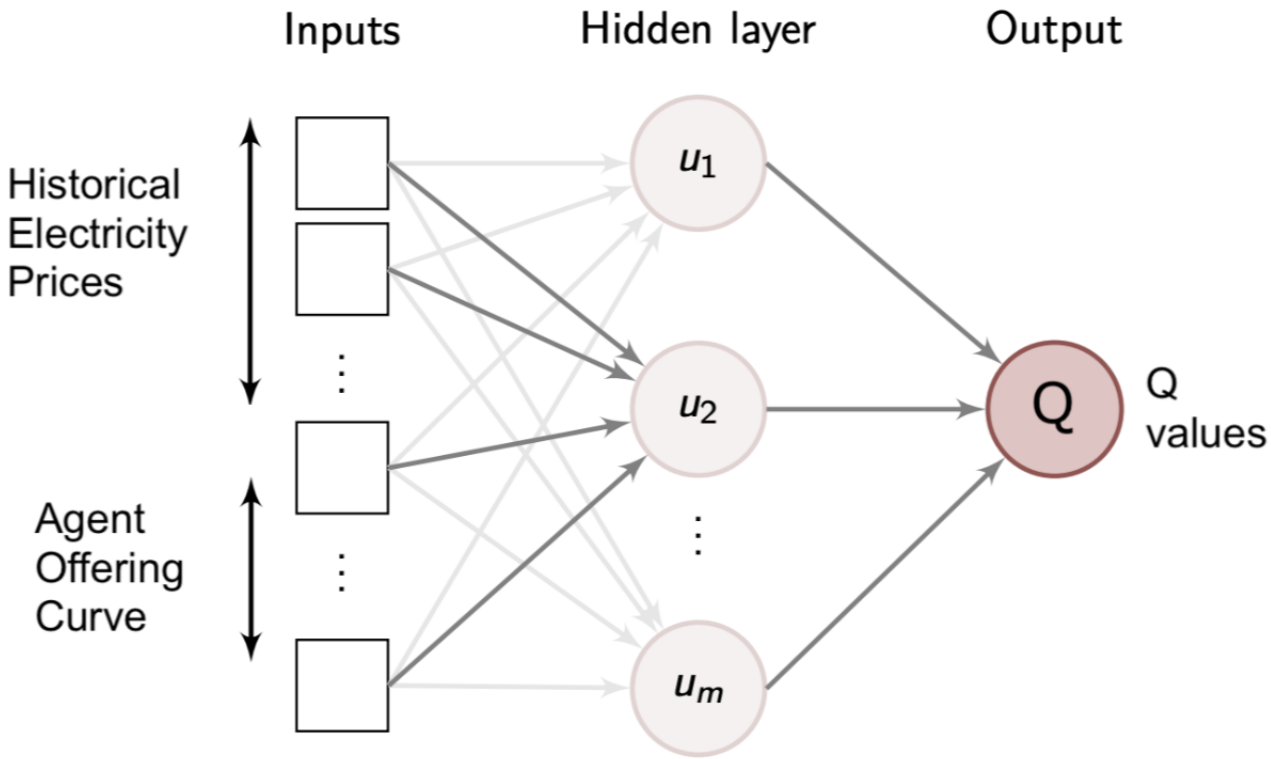
The AN approximates the current policy $\mu(s|\theta^{\mu})$.




Based on the observation of the last 7 days prices + production cost + available capacity, the Actor NN outputs the best offering curve

# The Critic Network

The CN approximates the Q function, for a given state-action pair by approximating the Bellman Equation



# Summary

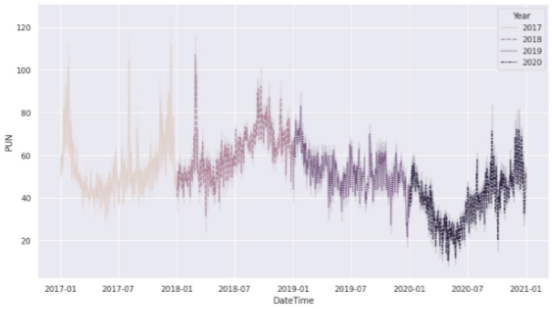
- **State**: Historical electricity prices, production costs, and capacities.
- **Action**: Offering curves defined by stepwise curves (volume-price pairs).
- Custom **Reward** function: Profit based on accepted bids (revenue minus production costs).
- **Adaptation of DDPG**: Actor proposes offering curves and Critic evaluates profitability of actions using the Q-value function.
  1. **Initialize**: Actor and Critic networks
  2. **Action Generation**: Actor proposes actions with added noise.
  3. **Environment Interaction**: Compute reward and observe next state.
  4. **Update Networks**:
     - Actor: Adjust policy to improve cumulative rewards.
     - Critic: Minimize loss between predicted and actual Q-values.

# Summary

## DATA

Historical Hourly Prices over 4 *years*

**STATES**: 7 days batches of unitary electricity prices

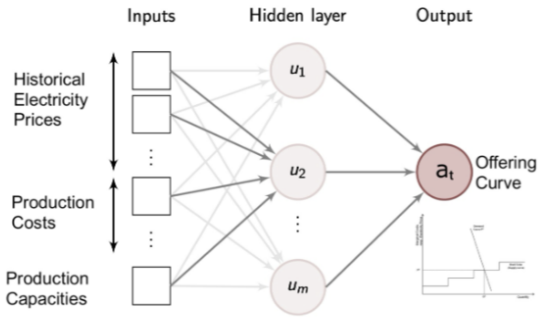



Two kind of producers:

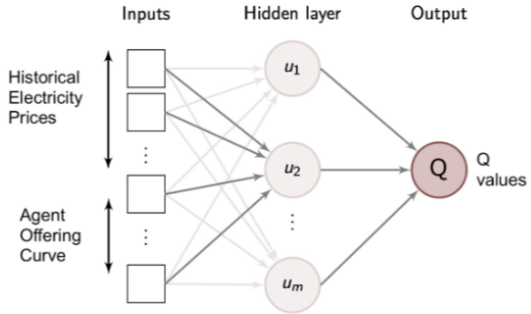
- Green producers (e.g. wind):
  Low Production Cost, Low Capacity
- Conventional producer (e.g. gas):
  High Production Cost, High Capacity

## TRAINING

Actor Network minimizes **REWARD**:




Critic Network approximates Bellman Eq.



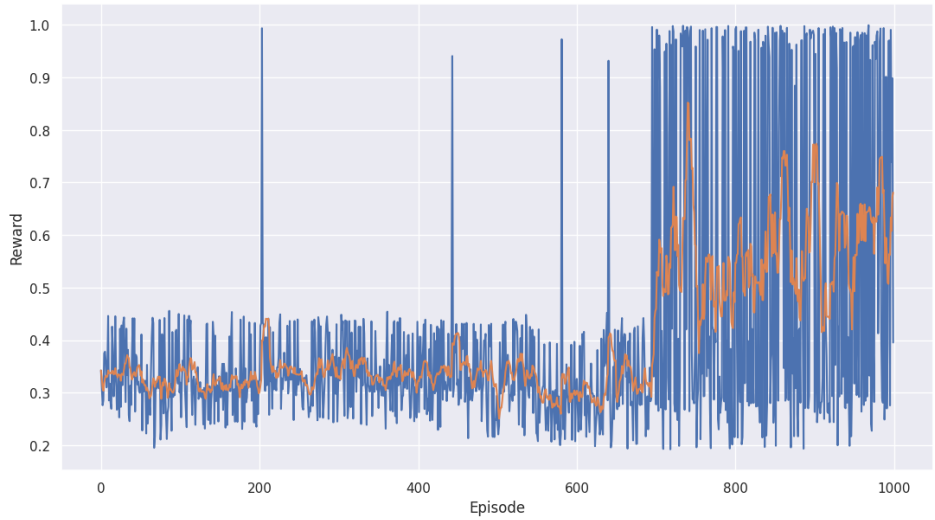

## TESTING

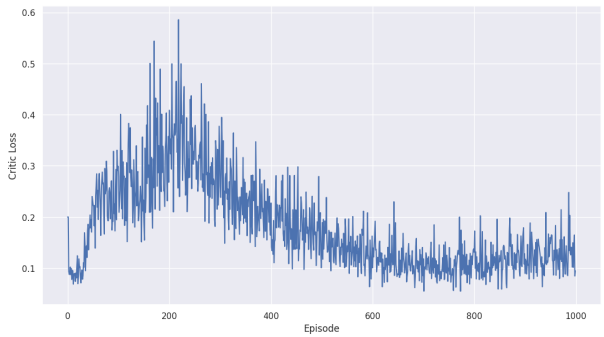
Given a 7-days observation of prices

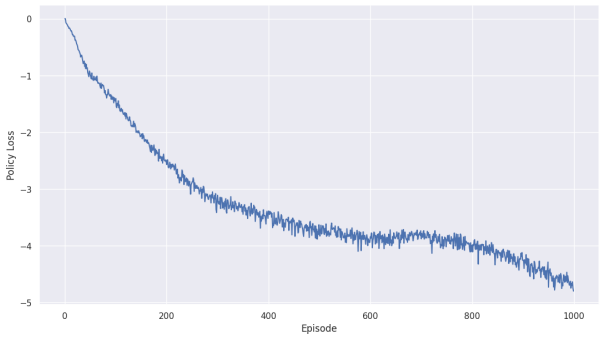

The trained Actor Network outputs **ACTIONS**

# The reward function

Episodes = 1000, Length of n episode = 30 days



# Possible extensions...

- **Model Improvements**:
  - Advanced neural network architectures (e.g., LSTM for time series).
- **Multi-Agent Systems**:
  - Competitive environments with multiple agents.
  - Decentralized coordination for market integration.
- **Real-World Integration**:
  - Incorporate live market data for actions for adaptive learning.