IASD M2 at Paris Dauphine

# Deep Reinforcement Learning

## 10: Applying Deep RL in practice with Kaggle

Eric Benhamou David Saltiel

# Regime changes are ubiquitous


Very challenging for backward looking portfolio methods

# Standard portfolio theory

**Markowitz and beyond optimizes a risk criteria based on past data**

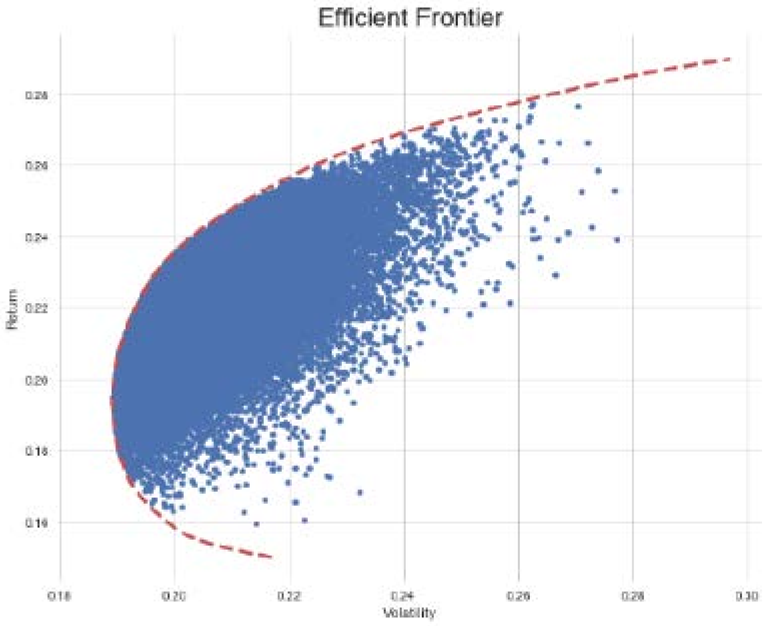



Figure 1: Markowitz efficient frontier for the GAFA: returns taken from 2017 to end of 2019

denote by $w = (w_1, ..., w_l)$ the allocation weights

$\mu = (\mu_1, ..., \mu_l)^T$ be the expected returns

$\Sigma$ the matrix of variance covariances

$r_{min}$ be the minimum expected return

$$\underset{w}{\text{Minimize}} \quad w^T \Sigma w \qquad (1)$$

$$\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0$$

# Why standard portfolio methods fail?

- **Standard portfolio methods rely:**
  - **on simplified settings (easy solving for convex optimization),**
  - **on few parameters (returns, volatility, correlation).**

- **By construction, these methods:**
  - **uses past data that are non stationary,**
  - **do not connect market data and context to allocation,**
  - **do not aim at capturing regime change.**

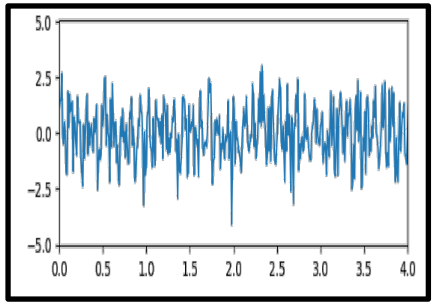
Standard methods do not perform so well in out of sample data. The question is: can we do better?

A modern approach to Markowitz portfolios

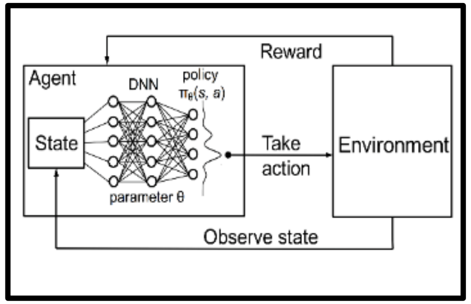
# 2. Deep Learning for portfolio allocation

# Deep Learning Portfolios

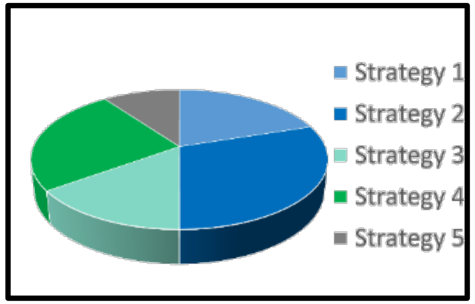
1.Data analysis




2. Model free DRL




3. Portfolio allocation



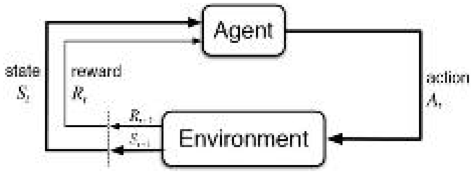

- Enriching the data set
  - Portfolio data : prices series of assets
  - Contextual data : volatility, sentiment, risk aversion, economic variables..

- Usually, data is fed directly into a DL algorithm, and allocation is decided, based on some reward function. Lots of choices :
  - Auto encoders
  - RNN
  - Convolution network…

# Dynamic allocation based on Deep Reinforcement Learning

## Reinforcement Learning (RL)

RL consists in **finding the optimal action** (the portfolio allocation) **according to state** (financial information) **given a reward** (the best net portfolio final performance).




$$A_t^* = \pi(S_t)$$ ?

RL **learns** and **finds the optimal portfolio allocation** in an interactive environment **by trial and error** using feedback from actions and rewards.

## Deep RL (DRL)

DRL uses deep learning in RL to find the function that maps states to optimal action.

$$A_t^* = \pi(S_t)$$

Deep networks can represent complex functions thanks to non-linear activation and multiple layers (universal approximation).

DRL has been **successfully applied in challenging problems** (games such as Go (Google), Poker (Facebook), or autonomous driving) to find solutions that perform better than human.

# The mathematical problem

We have a portfolio whose daily NAV is $(P_t)$. Portfolio is composed of $n$ assets whose daily returns are $R_t = \left[r_t^0, ..., r_t^n\right]$. Transaction assumed proportional to portfolio size and denoted by $b$

- **States** $S_t$: strategies past returns + volatility + correlation + additional contextual variables (other markets, macro, options, etc..)
- **Action** Reallocation at the end of each day $W_t(S_t) = (w_t^0, ..., w_t^n)$
- **Reward** Sharpe ratio at the end of the training period $SR(T) = \frac{Return(P)}{Volatility(P)}$

Find best function: $\pi : S_t \mapsto W_t$ to maximize $SR(T)$. We parametrize the deep function by $\pi(\theta)$

# The DRL portfolio manager

We hire a smart portfolio manager- Mr. Deep Reinforcement Learning. **Mr. DRL** will observe multiple variables and give us daily advice on **the portfolio weights**

---

**Algorithm 1** Portfolio allocation using Deep Reinforcement Learning (DRL)

---

**Input:** $S_t$, state space including
    past performance, volatility and other multiple financial data
**Output:** Daily portfolio allocation decisions $\pi : S_t \mapsto W_t$

Initialize policy function $\pi_0$
**while** Not converged **do**
    Initialize $P_0 = \$1,000,000$
    **for** t=0,...,T **do**
        Portfolio manager DRL observes $S_t$ and outputs $W_t = \pi_\theta(S_t)$
        Calculate next day portfolio value
$$P_{t+1} = P_t : \times \left(1 + W_t^T R_T - \mu^T |W_t - W_{t-1}|\right)$$
    Compute Training Sharpe ratio $SR(T) = \frac{(P_T/P_0-1)^{250/T}}{Std(P_t/P_{t-1}-1)\times 250}$
    Update policy weight $\theta_\pi$

---

# Link with previous portfolio methods

## Similarities

It has a reward that takes into account both risk and returns.
It is an optimization.
It includes Markowitz and Black Litterman. Why?
    Simply by taking very simple states like returns and variance.

## Differences

Conceptually, we try to find a function rather just parameters.
Optimization is strongly non convex.
By design, it connects allocation to multiple financial data.

**Stability and sensitivity to noisy data is an issue**

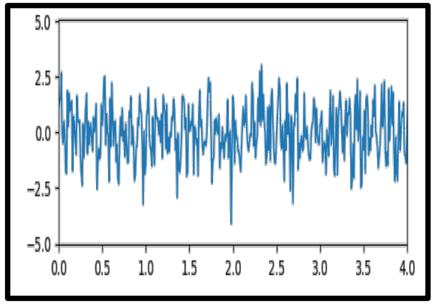
Using supervised predictions is closer to human decision and accelerate training

# 3. Boosting DRL

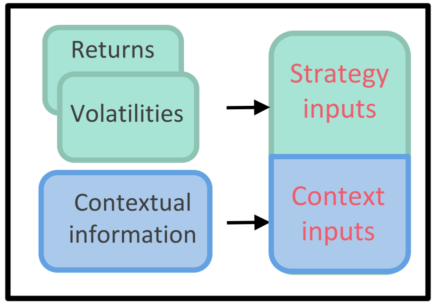
Combining Supervised and DRL fastens training

# Methodology with boosting DRL

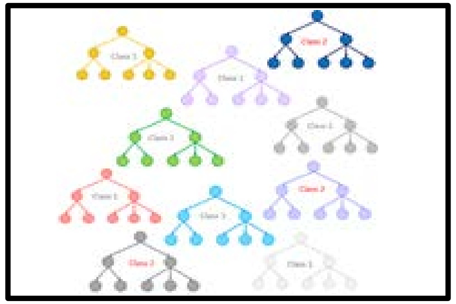
1.Data analysis



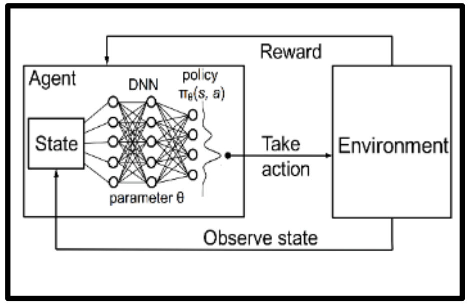

**2. State augmentation**



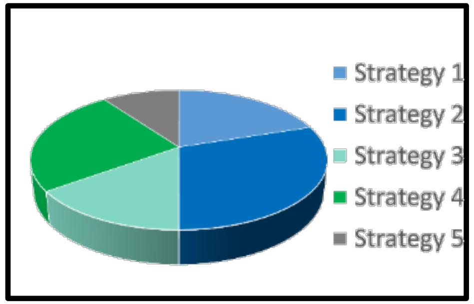

**3. Regime prediction**



4. Model free RL



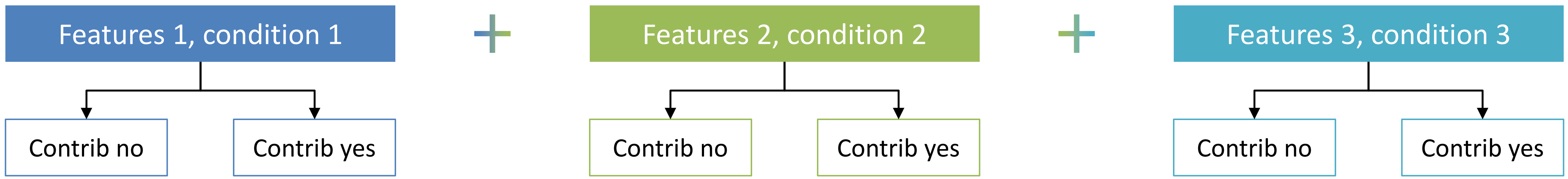

5. Portfolio allocation




- Data is prepared to create list of features, that are used to predict regimes
- We identify two regimes:
  - Bull regime: quantile 65% over 2 months
  - Bear regime: quantile 15% over 2 months
- The regimes are predicted independently with two distinct models

# Regime prediction

Estimating the likelihood of market regimes is a data classification problem.
We use Gradient-Boosting-Tree models (e.g. XGBoost, LightGBM) over other approaches (e.g. deep learning) since they dominate many ML competitions for classification problems and Kaggle competitions



# Robust filtering of features

**01** **Initial dataset & transformations**

150 factors under continuous examination

**02** **Gradient boosting tree**

Ranking and selecting features by importance

**03** **Variance Inflation Factor**

Removing colinear features

**04** **Final-specific dataset**

Around  33% of features selected

But interpretability and transparency of the model is still an issue

# Interpretability with Shapley values

- Researchers at Microsoft Research have proposed an approach to interpret model predictions using the Shapley values.
- The Shapley value is a concept based on game theory and is based on the work from Lloyd Shapley, who won the Nobel Prize in Economics for it in 2012.
- Back in the 1950s, Lloyd Shapley tried to fairly allocate the total gain to players in a cooperation game. The concept of fairness in this instance was defined by three main axioms:
  - efficiency : the sum of the gains attributed to each players should be equal to the total gains,
  - symmetry : if one player can replace another one then they should get the same reward,
  - linearity - a player that has a bigger impact than all other players in all games should receive a bigger reward.

# Kaggle challenge