

Two Applications of ML in Finance

I) Estimating Default Risk

II) Pricing and Hedging American-Style Derivatives

Sebastian Becker and Patrick Cheridito

RiskLab, ETH Zurich

ETH Zurich, April 9, 2021

A Closer Look at Supervised ML

1 Classical example: *The Titanic dataset*

- **Label:** did a passenger survive the Titanic disaster? $y \in \mathcal{Y} = \{0, 1\} = \{\text{yes, no}\}$
 - **Features:** $x_1 = \text{1st/2nd/3rd class}$, $x_2 = \text{gender}$, $x_3 = \text{age}$
 - Split the data into **training data** and **test data**: random choice of holdout data, or cross-validation
- More information is available here: <https://www.kaggle.com/c/titanic>

2 Modern example: *Covid 19*

- **Label:** who is at risk of being hospitalized if infected? $y \in \mathcal{Y} = \{0, 1\} = \{\text{no, yes}\}$
- **Features:** $x_1 = \text{age}$, $x_2 = \text{gender}$, $x_3 = \text{blood pressure}$, $x_4 = \text{blood type}$

Note there are different types of data

- 1st/2nd/3rd class: discrete numerical data
- blood pressure: continuous numerical data
- age: discrete or continuous numerical data
- gender, blood type: categorical data

In general ...

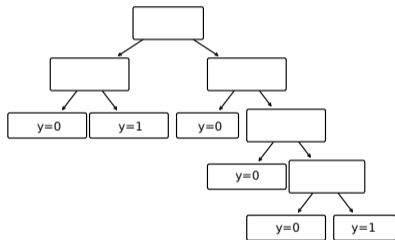
- there is a **feature set** $\mathcal{X} = \{(x_1, \dots, x_d) : x_1 \in \mathcal{X}_1, \dots, x_d \in \mathcal{X}_d\}$ and a **label set** $\mathcal{Y} \subseteq \mathbb{R}$
- there is **training data** $(x^j, y^j)_{j=1}^J \subseteq \mathcal{X} \times \mathcal{Y}$ and **test data** $(x^j, y^j)_{j=J+1}^{J+K} \subseteq \mathcal{X} \times \mathcal{Y}$
- **supervised learning** tries to find a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ parametrized by a **parameter** $\theta \in \Theta \subseteq \mathbb{R}^q$ that minimizes the **empirical loss**

$$\sum_{j=1}^J \ell(y^j, f_\theta(x^j)) \quad \text{for a given function } \ell : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty)$$

- **popular choice:** $\ell(y, z) = (y - z)^2$

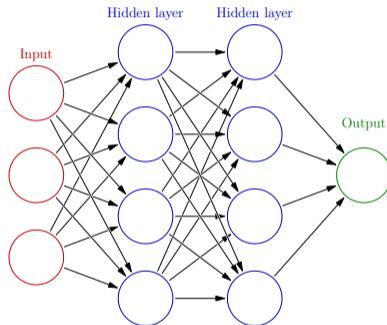
Popular models for $f_{\theta}: \mathcal{X} \rightarrow \mathcal{Y}$, $\theta \in \Theta \dots$

Decision trees



Grow the tree by iteratively splitting the *training set* based on relevant *features* and corresponding *thresholds*

Neural networks



Train the network on the *training data* with a *stochastic gradient descent* method

The quality of the results depends on ...

- the relevance and quality of the data ... *data collection and preparation*
- is the *training data* $(x^j, y^j)_{j=1}^J$ representative of the *test data* $(x^j, y^j)_{j=J+1}^{J+K}$? ... *generalization*
- is there enough *training data* $(x^j, y^j)_{j=1}^J$ compared to the *complexity* of $f_\theta, \theta \in \Theta$? ... *overfitting*

Advantages of trees

easy to understand/interpret
can be used with relatively little *training data*

Drawbacks of trees

often unstable
not suitable for large data sets

Advantages of neural networks

can find structure in large data sets
outcome is continuous in the *features*

Drawbacks of neural networks

needs a lot of *training data*
“black box”

Assessing the performance

- *Accuracy*: percentage of correct predictions on the test set

Class imbalance

In many applications, there is a large *negative class* and a small *positive class*

- Most people screened for a disease are not sick
- Most payments are not fraudulent

So, only predicting negatives trivially accomplishes high accuracy

- *Precision*: percentage of positive predictions that were correct
- *Recall*: percentage of actual positives that were predicted correctly

Application I

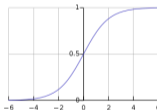
Estimating Default Risk

- **Label:** default probability $p \in (0, 1)$
- **Features:** $x_1 = \text{age}$, $x_2 = \text{income}$, $x_3 = \text{salaried/self-employed}$

We consider two different approaches

- ① **Logistic regression** $p(x) = f_{\theta}(x) = \psi(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$,

where $\psi(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$



(logistic function)

- ② **Neural network with two hidden layers**

$$p(x) = f_{\theta}(x) = \psi \circ A_3 \circ \rho \circ A_2 \circ \rho \circ A_1,$$

where

$$A_1: \mathbb{R}^3 \rightarrow \mathbb{R}^m, A_2: \mathbb{R}^m \rightarrow \mathbb{R}^n, A_3: \mathbb{R}^n \rightarrow \mathbb{R} \quad \text{are affine and} \quad \rho(x) = \max\{x, 0\} \quad (\text{ReLU})$$

- *Likelihood* of a Bernoulli(p) random variable to take the value $y \in \{0, 1\}$:

$$p^y(1-p)^{1-y} = \begin{cases} p & \text{for } y = 1 \\ 1-p & \text{for } y = 0 \end{cases}$$

- *Likelihood* of J i.i.d. Bernoulli(p) random variables to take the values y^1, \dots, y^J :

$$\prod_{j=1}^J p^{y^j} (1-p)^{1-y^j}$$

- *Log-likelihood*

$$\sum_{j=1}^J y^j \log p + (1 - y^j) \log(1 - p)$$

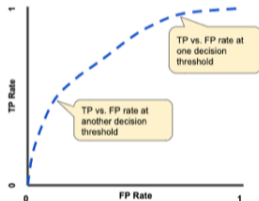
- *Training*: try to minimize the *total deviance (negative conditional log-likelihood)*

$$\theta \mapsto \sum_{j=1}^J -y^j \log f_{\theta}(x^j) - (1 - y^j) \log (1 - f_{\theta}(x^j))$$

on the *training data*

Evaluating the performance of the default model

- Is the *normalized total deviance* on the *test data* larger than on the *training data*? \rightsquigarrow overfitting!
- Conditional distribution of the test data (contingency table)
which percentage of the *test data* x^j with prediction $p(x^j) \in (a\%, b\%]$ is positive?
- A ROC (receiver operating characteristic) curve plots the *true positive rate* TP/P against the *false positive rate* FP/N for different *decision thresholds*



- What is important in this particular application? (compared to e.g. radar detection of incoming missiles)
 - A false positive is a “false alarm” or *lost business opportunity*
 - A false negative is a “miss” or *DEFAULTED LOAN!*
- Calculate estimates of the *expected P&L* and the *99%-Value-at-Risk* on the *test data*

Application II

Pricing and Hedging American-Style Derivatives

- Let X_0, X_1, \dots, X_N be a d -dimensional Markov process on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, i.e. $X_0, X_1, \dots, X_N: \Omega \rightarrow \mathbb{R}^d$ are random vectors such that

$$\mathbb{P}[X_{n+1} \in B \mid X_n] = \mathbb{P}[X_{n+1} \in B \mid X_0, \dots, X_n]$$

every random sequence can be made Markov by adding enough past information to the current state

- $g: \{0, 1, \dots, N\} \times \mathbb{R}^d \rightarrow \mathbb{R}$ a measurable function such that $\mathbb{E} g(n, X_n)^2 < \infty$ for all n
- **Optimal Stopping Problem**

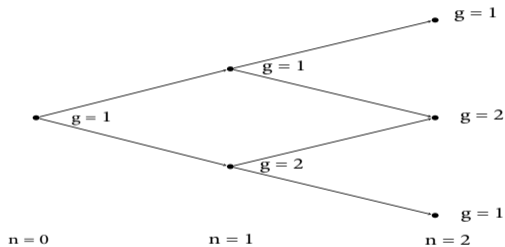
$$\sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau)$$

where \mathcal{T} is the set of all X -stopping times $\tau: \Omega \rightarrow \{0, 1, \dots, N\}$

that is, $1_{\{\tau=n\}} = h_n(X_0, \dots, X_n)$ for all n

Toy Example

$$X_0 = 0, \quad \mathbb{P}[X_1 = \pm 1] = \frac{1}{2}, \quad \mathbb{P}[X_2 = X_1 \pm 1 \mid X_1] = \frac{1}{2}$$



$$\tau^* = \begin{cases} 2 & \text{if } X_1 = 1 \\ 1 & \text{if } X_1 = -1 \end{cases}$$

$$\mathbb{E} g(\tau^*, X_{\tau^*}) = \frac{1}{4} \times 1 + \frac{1}{4} \times 2 + \frac{1}{2} \times 2 = 1.75$$

Deriving an Optimal Stopping Time

- Set

$$\tau_N^* = N$$

$$C_n = \mathbb{E} \left[g \left(\tau_{n+1}^*, X_{\tau_{n+1}^*} \right) \mid X_n \right], \quad n \leq N-1, \quad (\text{continuation value})$$

$$\tau_n^* = n \mathbf{1}_{\{g(n, X_n) \geq C_n\}} + \tau_{n+1}^* \mathbf{1}_{\{g(n, X_n) < C_n\}}, \quad n \leq N-1$$

- Then

$$V_n = \sup_{\tau \in \mathcal{T}_n} \mathbb{E} g(\tau, X_\tau) = \mathbb{E} g(\tau_n^*, X_{\tau_n^*})$$

where \mathcal{T}_n is the set of all X -stopping times τ such that $n \leq \tau \leq N$

- In particular,

$$V_0 = \sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau) = \mathbb{E} g(\tau_0^*, X_{\tau_0^*})$$

So τ_0^* is an optimal stopping time!

Stopping Decisions

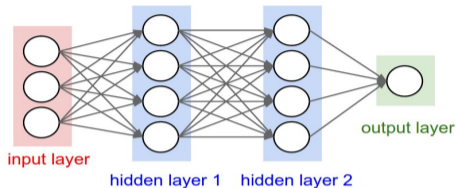
- Let $f_n, f_{n+1}, \dots, f_N : \mathbb{R}^d \rightarrow \{0, 1\}$ be measurable functions such that $f_N \equiv 1$. Then

$$\tau_n = \sum_{m=n}^N m f_m(X_m) \prod_{j=n}^{m-1} (1 - f_j(X_j)) \quad \text{with} \quad \prod_{j=n}^{n-1} (1 - f_j(X_j)) := 1$$

is a stopping time in \mathcal{T}_n

- $$\tau_n = n f_n(X_n) + \tau_{n+1} (1 - f_n(X_n)) \quad \text{where} \quad \tau_{n+1} = \sum_{m=n+1}^N m f_m(X_m) \prod_{j=n+1}^{m-1} (1 - f_j(X_j))$$

Neural Network Approximation



Idea Recursively approximate f_n by a neural network $f^\theta: \mathbb{R}^d \rightarrow \{0, 1\}$ of the form

$$f^\theta = 1_{[0, \infty)} \circ a_3^\theta \circ \rho \circ a_2^\theta \circ \rho \circ a_1^\theta,$$

where

- q_1 and q_2 are positive integers specifying the number of nodes in the two hidden layers,
- $a_1^\theta: \mathbb{R}^d \rightarrow \mathbb{R}^{q_1}$, $a_2^\theta: \mathbb{R}^{q_1} \rightarrow \mathbb{R}^{q_2}$ and $a_3^\theta: \mathbb{R}^{q_2} \rightarrow \mathbb{R}$ are affine functions given by

$$a_i^\theta(x) = A_i x + b_i, \quad i = 1, 2, 3,$$

- for $j \in \mathbb{N}$, $\rho: \mathbb{R}^j \rightarrow \mathbb{R}^j$ is the component-wise ReLU activation function given by $\rho(x_1, \dots, x_j) = (x_1^+, \dots, x_j^+)$

The components of θ consist of the entries of A_i and b_i , $i = 1, 2, 3$

More precisely,

- assume parameter values $\theta_{n+1}, \theta_{n+2}, \dots, \theta_N \in \mathbb{R}^q$ have been found such that $f^{\theta_N} \equiv 1$ and the stopping time

$$\tau_{n+1} = \sum_{m=n+1}^N m f^{\theta_m}(X_m) \prod_{j=n+1}^{m-1} (1 - f^{\theta_j}(X_j))$$

produces an expectation $\mathbb{E} g(\tau_{n+1}, X_{\tau_{n+1}})$ close to the optimal value V_{n+1}

- now try to find a maximizer $\theta_n \in \mathbb{R}^q$ of

$$\theta \mapsto \mathbb{E} \left[g(n, X_n) f^{\theta}(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}}) (1 - f^{\theta}(X_n)) \right]$$

- **Goal** find an (approximately) optimal $\theta_n \in \mathbb{R}^q$ with a *stochastic gradient ascent method*

- **Problem** for $x \in \mathbb{R}^d$, the θ -gradient of

$$f^\theta(x) = 1_{[0, \infty)} \circ a_3^\theta \circ \rho \circ a_2^\theta \circ \rho \circ a_1^\theta(x)$$

is 0 or does not exist

- As an **intermediate step** consider a neural network $F^\theta: \mathbb{R}^d \rightarrow (0, 1)$ of the form

$$F^\theta = \psi \circ a_3^\theta \circ \rho \circ a_2^\theta \circ \rho \circ a_1^\theta \quad \text{for} \quad \psi(x) = \frac{e^x}{1 + e^x}$$

- Use **stochastic gradient ascent** to find an approximate optimizer $\theta_n \in \mathbb{R}^q$ of

$$\theta \mapsto \mathbb{E} \left[g(n, X_n) F^\theta(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}}) (1 - F^\theta(X_n)) \right]$$

- **Approximate** $f_n \approx f^{\theta_n} = 1_{[0, \infty)} \circ a_3^{\theta_n} \circ \rho \circ a_2^{\theta_n} \circ \rho \circ a_1^{\theta_n}$

- **Repeat the same steps** at times $n - 1, n - 2, \dots, 0$

Training the Networks

- Let $(x_n^k)_{n=0}^N$, $k = 1, 2, \dots$ be independent simulations of $(X_n)_{n=0}^N$
- Let $\theta_{n+1}, \dots, \theta_N \in \mathbb{R}^q$ be given, and consider the corresponding stopping time

$$\tau_{n+1} = \sum_{m=n+1}^N m f^{\theta_m}(X_m) \prod_{j=n+1}^{m-1} (1 - f^{\theta_j}(X_j))$$

- τ_{n+1} is of the form $\tau_{n+1} = l_{n+1}(X_{n+1}, \dots, X_{N-1})$ for a measurable function

$$l_{n+1} : \mathbb{R}^{d(N-n-1)} \rightarrow \{n+1, n+2, \dots, N\}$$

- Denote

$$l_{n+1}^k = \begin{cases} N & \text{if } n = N-1 \\ l_{n+1}(x_{n+1}^k, \dots, x_{N-1}^k) & \text{if } n \leq N-2 \end{cases}$$

- The realized reward

$$r_n^k(\theta) = g(n, x_n^k) F^\theta(x_n^k) + g(l_{n+1}^k, x_{l_{n+1}^k}^k) (1 - F^\theta(x_n^k))$$

is continuous and almost everywhere differentiable in θ

Stochastic Gradient Ascent

- **Initialize** $\theta_{n,0}$ typically random; e.g. Xavier initialization

- **Standard updating** $\theta_{n,k+1} = \theta_{n,k} + \eta \nabla r_n^k(\theta_{n,k})$

- **Variants**

- Mini-batches

- Batch normalization

- Momentum

- Adagrad

- RMSProp

- AdaDelta

- Adam

- Decoupled weight decay

- Warm restarts

- ...

Lower Bound

- The candidate optimal stopping time

$$\tau^\Theta = \sum_{n=1}^N n f^{\theta_n}(X_n) \prod_{j=0}^{n-1} (1 - f^{\theta_j}(X_j))$$

yields a lower bound

$$L = \mathbb{E} g(\tau^\Theta, X_{\tau^\Theta}) \quad \text{for the optimal value} \quad V_0 = \sup_{\tau} \mathbb{E} g(\tau, X_{\tau})$$

- Let $(y_n^k)_{n=0}^N$, $k = 1, 2, \dots, K_L$, be a new set of independent simulations of $(X_n)_{n=0}^N$
- τ^Θ can be written as $\tau^\Theta = l(X_0, \dots, X_{N-1})$ for a measurable function $l : \mathbb{R}^{dN} \rightarrow \{0, 1, \dots, N\}$
- Denote $l^k = l(y_0^k, \dots, y_{N-1}^k)$

- Use the Monte Carlo approximation

$$\hat{L} = \frac{1}{K} \sum_{k=1}^{K_L} g(l^k, y_{l^k}^k) \quad \text{as an estimate for } L$$

Lower Confidence Bound

- Consider the *sample variance*

$$\hat{\sigma}_L^2 = \frac{1}{K_L - 1} \sum_{k=1}^{K_L} \left(g(l^k, y_{l^k}^k) - \hat{L} \right)^2$$

- By the *central limit theorem*,

$$\left[\hat{L} - z_\alpha \frac{\hat{\sigma}_L}{\sqrt{K_L}}, \infty \right)$$

is an asymptotically valid $1 - \alpha$ confidence interval for L

where z_α is the $1 - \alpha$ quantile of the standard normal distribution

- As a consequence,

$$\left[\hat{L} - z_\alpha \frac{\hat{\sigma}_L}{\sqrt{K_L}}, \infty \right)$$

is also an asymptotically valid $1 - \alpha$ confidence interval for the true optimal value V_0

Dual Problem

- The **value process** $H_n = g(n, X_n) \vee C_n$ is a *super-martingale*
- Let $H_n = H_0 + M_n^H - A_n^H$ be the *Doob decomposition*, that is,
 $(M_n^H)_{n=0}^N$ is a *martingale* and $(A_n^H)_{n=0}^N$ a *non-decreasing predictable process* such that $M_0^H = A_0^H = 0$

Theorem

and

$$V_0 = \mathbb{E} \left[\max_{0 \leq n \leq N} \{g(n, X_n) - M_n^H\} \right]$$

$$V_0 \leq \mathbb{E} \left[\max_{0 \leq n \leq N} \{g(n, X_n) - M_n - \varepsilon_n\} \right]$$

for every martingale $(M_n)_{n=0}^N$ with $M_0 = 0$ and **estimation errors** $(\varepsilon_n)_{n=0}^N$ satisfying $\mathbb{E}[\varepsilon_n \mid \mathcal{F}_n^X] = 0$

Approximate Upper Bound

- Let $(M_n^\Theta)_{n=0}^N$ be the *martingale part of the value process* generated by $f^{\theta_0}, \dots, f^{\theta_{N-1}}$
- Use *nested simulation* to generate realizations M_n^k of $M_n^\Theta + \varepsilon_n$ (*unbiased estimation errors*) along simulated paths z_n^k of $X_n, n = 0, \dots, N$

•

$$U = \mathbb{E} \left[\max_{0 \leq n \leq N} \left(g(n, X_n) - M_n^\Theta - \varepsilon_n \right) \right] \quad \text{is an upper bound for } V_0$$

- Use the Monte Carlo approximation

$$\hat{U} = \frac{1}{K_U} \sum_{k=1}^{K_U} \max_{0 \leq n \leq N} \left(g(n, z_n^k) - M_n^k \right) \quad \text{as an estimate for } U$$

Our point estimate of V_0 is $\hat{V} = \frac{\hat{L} + \hat{U}}{2}$

Confidence Interval for V_0

- By the *central limit theorem*,

$$\left(-\infty, \hat{U} + z_\alpha \frac{\hat{\sigma}_U}{\sqrt{K_U}} \right]$$

is an asymptotically valid $1 - \alpha$ confidence interval for U , where $\hat{\sigma}_U$ is the corresponding *sample standard deviation*

- So,

$$\left[\hat{L} - z_\alpha \frac{\hat{\sigma}_L}{\sqrt{K_L}}, \hat{U} + z_\alpha \frac{\hat{\sigma}_U}{\sqrt{K_U}} \right]$$

is an asymptotically valid $1 - 2\alpha$ confidence interval for V_0 .

Bermudan Max-Call Options

Consider d assets with prices evolving according to a multi-dimensional Black–Scholes model

$$S_t^i = s_0^i \exp\left([r - \delta_i - \sigma_i^2/2]t + \sigma_i W_t^i\right), \quad i = 1, 2, \dots, d,$$

for

- initial values $s_0^i \in (0, \infty)$
- a risk-free interest rate $r \in \mathbb{R}$
- dividend yields $\delta_i \in [0, \infty)$
- volatilities $\sigma_i \in (0, \infty)$
- and a d -dimensional Brownian motion W with constant correlation ρ_{ij} between increments of different components W^i and W^j

A Bermudan max-call option has time- t payoff $(\max_{1 \leq i \leq d} S_t^i - K)^+$ and can be exercised at one of finitely many times $0 = t_0 < t_1 = \frac{T}{N} < t_2 = \frac{2T}{N} < \dots < t_N = T$

$$\text{Price:} \quad \sup_{\tau \in \{t_0, t_1, \dots, T\}} \mathbb{E} \left[e^{-r\tau} \left(\max_{1 \leq i \leq d} S_\tau^i - K \right)^+ \right] = \sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau)$$

Numerical Results

for $s_0^i = 100$, $\sigma_i = 20\%$, $r = 5\%$, $\delta = 10\%$, $\rho_{ij} = 0$, $K = 100$, $T = 3$ years, $N = 9$:

# Assets	Point Est.	Comp. Time	95% Conf. Int.	Bin. Tree	Broadie–Cao 95% Conf. Int.
2	13.899	28.7s	[13.880, 13.910]	13.902	
3	18.690	28.9s	[18.673, 18.699]	18.69	
5	26.159	28.1s	[26.138, 26.174]		[26.115, 26.164]
10	38.337	30.5s	[38.300, 38.367]		
20	51.668	37.5s	[51.549, 51.803]		
30	59.659	45.5s	[59.476, 59.872]		
50	69.736	59.1s	[69.560, 69.945]		
100	83.584	95.9s	[83.357, 83.862]		
200	97.612	170.1s	[97.381, 97.889]		
500	116.425	493.5s	[116.210, 116.685]		

Hedging

- *Price evolution*

$$S_t^i = s_0^i \exp([r - \delta_i - \sigma_i^2/2]t + \sigma_i W_t^i)$$

- *Exercise times*

$$t_n = \frac{nT}{N}, \quad n = 0, 1, \dots, N$$

- *Hedging rebalancing times*

$$u_m = \frac{mT}{NM}, \quad m = 0, 1, \dots, NM$$

- *Discounted dividend-adjusted prices* $P_{u_m}^i = p_m^i(W_{u_m}^i) = s_0^i \exp(\sigma_i W_{u_m}^i - \sigma_i^2 u_m/2)$

- *Hedging portfolio*

$$(h \cdot P)_{u_m} = \sum_{j=0}^{m-1} \sum_{i=1}^d h_j^i(P_{u_j}) (P_{u_{j+1}}^i - P_{u_j}^i)$$

- *Simulate paths* $(w_m^k)_{m=0}^{NM}$, $k = 1, \dots, K_H$, of $(W_{u_m})_{m=0}^{NM}$

- *Train neural networks* $h^{\lambda_m} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to *minimize*

$$\sum_{k=1}^{K_H} \left(\hat{V} + \sum_{m=0}^{\tau^\theta M - 1} h^{\lambda_m}(w_m^k) \cdot (p_{m+1}(w_{m+1}^k) - p_m(w_m^k)) - g^k \right)^2$$

where

$$g^k = \exp \left(-r \frac{\tau^\Theta T}{N} \right) \left(\max_{1 \leq i \leq d} s_0^i \exp \left(\left[r - \delta_i - \frac{\sigma_i^2}{2} \right] \frac{\tau^\theta T}{N} + \sigma_i w_{\tau^\theta M}^{k,i} \right) - K \right)^+ \quad (\text{discounted payoff})$$

- *Evaluate*

$$\hat{V} + \sum_{m=0}^{\tau^\theta M - 1} h^{\lambda_m}(\tilde{w}_m^k) \cdot (p_{m+1}(\tilde{w}_{m+1}^k) - p_m(\tilde{w}_m^k)) - \tilde{g}^k$$

along independent samples $(\tilde{w}_m^k)_{m=0}^{NM}$, $k = 1, \dots, K_T$, of $(W_{u_m})_{m=0}^{NM}$

\rightsquigarrow yields an empirical distribution of the hedging error