Universal Static Hedging of Contingent Claims Using Neural Networks

Shashi Jain [1]

[1] Department of Management Studies,
Indian Institute of Science. Bangalore, India

Machine learning in quantitative finance and risk management
CWI, Amsterdam
2nd July 2020

Outline

Motivation

1. Can we use neural networks to automate the choice of basis functions used in the regression for American Monte Carlo ?
2. While working on (1) we stumbled on something else – in well functioning markets, with certain assumptions, path-dependent contingent claims can be hedged using portfolio of short maturity options.
3. Not the first to realize (2), however, to the best of our knowledge, our framework is the first to extend static hedging to high-dimensional derivatives.

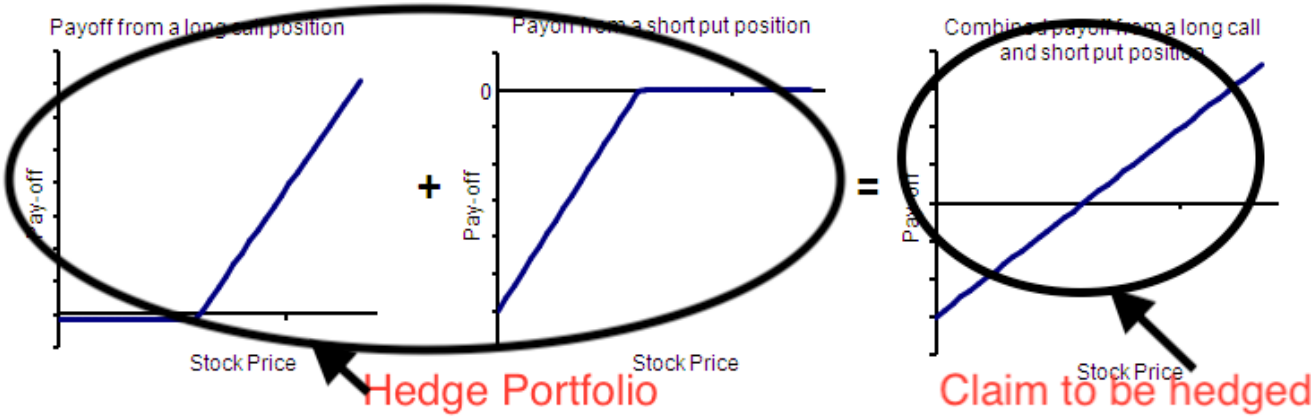
Structure

- Dynamic vs Static Hedging.
- Pricing Bermudan options, with focus on the regress later approach.
- Regress later with Neural Network (RLNN)
- Results

Dynamic Hedging

- Black, Scholes and Merton showed that, under certain assumptions, market risk of options can be eliminated by the setting up of a dynamic hedge account.
  - The hedge account consists of money market account and position in the underlyings on which the option is written.
  - The quantum of the positions in the underlyings is determined by the *Delta* values of the option being hedged.
  - The position should ideally be updated within short time intervals, in practice mostly EOD.
- Dynamic hedging often breaks down when there are sharp movements (jumps) in markets, or when the market faces liquidity issues.
- Unfortunately these are the precise moments where an effective hedge is highly desired.

Static Hedging



- Consider the payoff at T of a claim to be hedged (depicted in the right hand side of the figure below).
- A combination of long call and short put can replicate this payoff at T.
- If the two portfolios have the same value at T, then by no-arbitrage argument (under appropriate assumptions), at any time prior to T, including $t = 0$, the value of the two portfolios should be exactly the same. We have a perfect hedge, with no need for rebalancing!

Static Hedging

Theorem

(Carr and Wu [2013]) Under the Markovian and no-arbitrage assumptions, the time-t value of a European option maturing at a fixed time $T \geq t$ relates to the time-t value of a continuum of European call options at a shorter maturity $u \in [t, T]$, by

$$V_t^T(S_t, K) = \int_0^\infty w(S_u) V_t^u(S_t, S_u)\, dS_u, \ u \in [t, T], \tag{1}$$

for any $S_t > 0$ and $t \leq u$. The weights $w(S_u)$ do not vary with S_t or t, and are given by

$$w(S_u) = \frac{\partial^2}{\partial S_u^2} V_u^T(S_u, K) \tag{2}$$

- Any payoff can then be represented by a portfolio of continuum of options with different strikes. For practical purposes you consider finite set of options, obtained using numerical methods like Gaussian quadrature

Proof

- For a single factor Markovian process , using the results of Breeden and Litzenberger [1978], who show the risk neutral density can be equated to the second strike derivative of the call option using the following relation:

$$\mathbb{P}\left(S_T \mid S_t\right) = e^{r(T-t)} \frac{\partial^2}{\partial S_T^2} V_t^T\left(S_t, S_T\right) \tag{3}$$

$$\begin{aligned} V_t^T(S_t, K) &= e^{-r(u-t)} \mathbb{E}\left[V_u^T(S_u, K) \mid \mathcal{F}_t\right], \ u \in [t, T] \\ &= e^{-r(u-t)} \int_0^\infty V_u^T(S_u, K) \mathbb{P}(S_u | S_t)\, dS_u \\ &= \int_0^\infty V_u^T(S_u, K) \frac{\partial^2}{\partial S_u^2} V_t^u\left(S_t, S_u\right)\, dS_u \qquad (4) \\ &= \int_0^\infty V_t^u(S_t, S_u) \frac{\partial^2}{\partial S_u^2} V_u^T(S_u, K), \qquad (5) \end{aligned}$$

The problem

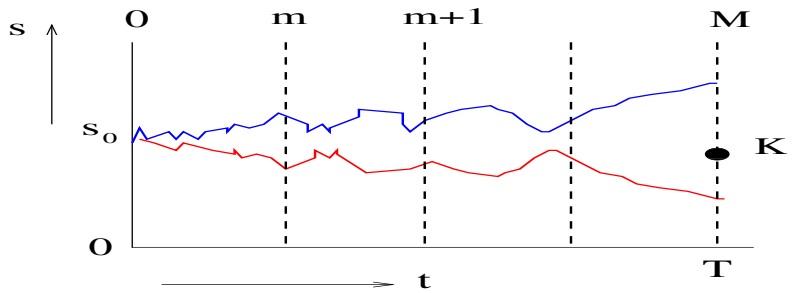
- Bermudan option value at t_0, is given by,

$$\frac{V_{t_0}(\mathbf{S}_{t_0})}{B_{t_0}} = \sup_{\tau \in \mathrm{T}} \mathbb{E}\left[\frac{h(\mathbf{S}_\tau)}{B_\tau}\right],$$

- where T is the set of admissible times $[t_0 = 0, \ldots, t_m, \ldots, t_M = T]$
- $h(\mathbf{X}_{t_m})$: payoff in t_m.
- $B_t = \exp(\int_0^t r_s\, ds)$, is risk-less saving accounts process, where r_t denotes the instantaneous risk-free rate of return.

Dynamic Programming Formulation

- In order to compute the option value, it is necessary to compute the early-exercise policy.



- At the terminal time the option value is given by,

$$V_T(\mathbf{S}_T) = \max(h(\mathbf{S}_T), 0). \tag{6}$$

- The Bermudan option value at time t_{m-1} and state $\mathbf{S}_{t_{m-1}}$ is given by

$$V_{t_{m-1}}(\mathbf{S}_{t_{m-1}}) = \max(h(\mathbf{S}_{t_{m-1}}), Q_{t_{m-1}}(\mathbf{S}_{t_{m-1}})). \tag{7}$$

- The continuation value $Q_{t_{m-1}}$, is :

$$Q_{t_{m-1}}(\mathbf{S}_{t_{m-1}}) = B_{t_{m-1}} \mathbb{E}\left[\frac{V_{t_m}(\mathbf{S}_{t_m})}{B_{t_m}} \mid \mathbf{S}_{t_{m-1}}\right]. \tag{8}$$

Regress Later With Neural Network (RLNN)

- The algorithm begins by generating N independent copies of sample paths, $\{\mathbf{S}_{t_0}, \ldots, \mathbf{S}_{t_M}\}$.
- The method then computes the value of the contingent claim at terminal time as $V_{t_M}(\mathbf{S}_{t_M}) = \max(h(\mathbf{S}_{t_M}), 0)$.
- The following steps are then employed for each monitoring date, t_m, $m \leq M$, recursively moving backwards in time, starting from t_M:
- Assume that $\tilde{V}_{t_m}(\mathbf{S}_{t_m}(n))$, $n = 1, \ldots, N$, our estimates for $V_{t_m}(\mathbf{S}_{t_m}(n))$, are known.

RLNN continued

- **Regress Later**: In this step a parametrized value function $\tilde{G} : \mathbb{R}^d \times \mathbb{R}^{N_p} \mapsto \mathbb{R}$, is computed, where $\beta_{t_m} \in \mathbb{R}^{N_p}$ is a vector of free parameters.
  The objective is to choose for each t_m, a parameter vector β_{t_m}, so that
  $$\tilde{G}(\mathbf{S}_{t_m}, \beta_{t_m}) = \tilde{V}_{t_m}(\mathbf{S}_{t_m}).$$
- **Compute Continuation Value**: The continuation values for the sample points $\mathbf{X}_{t_{m-1}}(n),\ n = 1, \ldots, N$ are then approximated by,

$$\begin{aligned} \widehat{Q}_{t_{m-1}}\left(\mathbf{X}_{t_{m-1}}(n)\right) &= B_{t_{m-1}} \mathbb{E}\left[\frac{\tilde{V}_{t_m}(\mathbf{S}_{t_m})}{B_{t_m}} \mid \mathbf{S}_{t_{m-1}}(n)\right] \\ &\approx B_{t_{m-1}} \mathbb{E}\left[\frac{\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})}{B_{t_m}} \mid \mathbf{S}_{t_{m-1}}(n)\right] \end{aligned} \tag{9}$$

RLNN continued

- We need to ensure that $\tilde{G}(\mathbf{S}_{t_m}, \beta_{t_m})$
  a. Is flexible enough to approximate $\tilde{V}_{t_m}(\mathbf{S}_{t_m})$ to arbitrary accuracy
  b. A closed form conditional expectation $\mathbb{E}\left[\frac{\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})}{B_{t_m}} \mid \mathbf{S}_{t_{m-1}}(n)\right]$ can be obtained.
- By the *universal approximation theorem*, a neural network with a single hidden layer and sufficiently large number of neurons can approximate any continuous function over a compact set arbitrarily well (see Hornik et al. [1989]). We therefore choose as our approximation architecture at t_m a feed-forward network $\tilde{G}^{\beta} : \mathbb{R}^d \to \mathbb{R}$ with single hidden layer.
- While the single hidden layered neural network satisfies the criterion (a), (b) is ensured by a specific choice of the activation function.

The network architecture

- We choose as our approximation architecture at t_m a feed-forward network $\tilde{G}^{\beta} : \mathbb{R}^d \to \mathbb{R}$ of the form

$$\tilde{G}^{\beta_{t_m}} := \psi \circ A_2 \circ \varphi \circ A_1$$

- where $A_1 : \mathbb{R}^d \to \mathbb{R}^p$ and $A_2 : \mathbb{R}^p \to \mathbb{R}$ are affine functions of the form,

$$A_1(\mathbf{x}) = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1 \quad \text{for } \mathbf{x} \in \mathbb{R}^d,\ \mathbf{W}_1 \in \mathbb{R}^{p\times d}, \mathbf{b}_1 \in \mathbb{R}^p,$$

and

$$A_2(\mathbf{x}) = \mathbf{W}_2\mathbf{x} + b_2 \quad \text{for } \mathbf{x} \in \mathbb{R}^p,\ \mathbf{W}_2 \in \mathbb{R}^{1\times p}, b_2 \in \mathbb{R}.$$

- p is the number of neurons used in the hidden layer.

Architecture continued

- $\varphi : \mathbb{R}^j \to \mathbb{R}^j, j \in \mathbb{N}$ is the component-wise ReLU activation function given by:

$$\varphi(x_1, \ldots, x_j) := \left(\max(x_1, 0), \ldots, \max(x_j, 0)\right),$$

- while $\psi : \mathbb{R}^j \to \mathbb{R}^j, j \in \mathbb{N}$ is the component-wise linear activation function given by:

$$\psi(x_1, \ldots, x_j) := \left(x_1, \ldots, x_j\right).$$

- $\beta_{t_m} \in \mathbb{R}^{N_p}$ is chosen to minimize the following mean squared error.

$$\beta_{t_m} = \underset{\beta_{t_m}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{n=1}^{N} \left(\tilde{V}_{t_m}(\mathbf{S}_{t_m}(n)) - \tilde{G}^{\beta_{t_m}} \left(\mathbf{S}_{t_m}(n)\right) \right)^2 \right), \qquad (10)$$

- where $\mathbf{S}_{t_m}(n),\ n = 1, \ldots, N$ now constitute the set of training points.

Interpretation of the first layer

- The outcome of the first hidden layer with a choice of p neurons can be represented as

$$\mathbf{o} = \varphi\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right),\ \mathbf{x} \in \mathbb{R}^d,\ \mathbf{W}_1 \in \mathbb{R}^{p\times d}, \mathbf{b}_1 \in \mathbb{R}^p,$$

  where $\varphi : \mathbb{R}^p \to \mathbb{R}^p, p \in \mathbb{N}$ is the component-wise ReLU activation function.
- Each of the p elements of $\mathbf{o} := \{o_1, \ldots, o_p\}$ therefore has the form,

$$o_i = \max\left(\sum_{j=1}^{d} w_{ij}x_j + b_{i1}, 0\right),$$

- the ith neuron, where $i = 1, \ldots, p$, has the form of the payoff of an arithmetic basket option with weights $w_{ij},\ j = 1, \ldots, d$ and strike b_{i1}, written on the underlying $\mathbf{x} := \{x_1, \ldots, x_d\}$.

Interpretation of the second layer

- The second hidden layer performs the following operation:

$$y = \left(\sum_{i=1}^{p} \omega_i o_i \right) + b_2, \text{ where, } \mathbf{W}_2 := \{\omega_1, \ldots, \omega_p\}, \tag{11}$$

- which can be seen as determining the weights of the p basket options you need to hold in your portfolio.
- The amount you need to invest in the risk free asset is given by the bias of the second hidden layer, i.e. b_2.

Continuation Value

- Once $\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})$, has been estimated the continuation value at any time t between $[t_{m-1}, t_m]$, can be approximated as the expectation of $\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})$, give $\mathcal{F}_t$.
- As $\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})$, is the aggregated payoff of portfolio of arithmetic basket options, its conditional expectation at t is the corresponding aggregated value of the corresponding European basket options.
- In the case of GBM process, if the objective is pricing (rather than hedging), a neat closed form solution for the continuation value is obtained if we work with $\tilde{G}^{\beta_{t_m}}(\log(\mathbf{S}_{t_m}))$, rather than $\tilde{G}^{\beta_{t_m}}(\mathbf{S}_{t_m})$.

Universal Static Hedging

- $\tilde{G}(\mathbf{S}_{t_m}, \beta_{t_m})$, is the aggregated payoff of portfolio of p arithmetic basket options that mature at t_m.
- By *universal approximation theorem*, for sufficiently large value of p, $\tilde{G}(\mathbf{S}_{t_m}, \beta_{t_m})$, can fit any continuous compact function in $\mathbb{R}^d$, including $\tilde{V}_{t_m}(\mathbf{S}_{t_m})$.
- As $\tilde{G}(\mathbf{S}_{t_m}, \beta_{t_m})$, can replicate the option value at V_{t_m}, under Markovian and no-arbirtage assumption, G should replicate V any time between $(t_{m-1}, t_m]$.
- The convergence of the method is discussed in detail in Lokeshwar et al. [2019].

Upper Bounds

- Haugh and Kogan Haugh and Kogan [2004] proposed the dual formulation for pricing Bermudan options. For an arbitrary adapted super-martingale process $\mathcal{M}_t$, it follows that,

$$\begin{aligned} V_{t_0}(\mathbf{X}_{t_0}) &= \sup_{\tau} \mathbb{E}\left[\frac{h_\tau}{B_\tau}\right] \\ &= \sup_{\tau} \mathbb{E}\left[\frac{h_\tau}{B_\tau} + \mathcal{M}_\tau - \mathcal{M}_\tau\right] \\ &\leq \mathcal{M}_{t_0} + \sup_{\tau} \mathbb{E}\left[\frac{h_\tau}{B_\tau} - \mathcal{M}_\tau\right] \\ &\leq \mathcal{M}_{t_0} + \mathbb{E}\left[\max_t \left(\frac{h_t}{B_t} - \mathcal{M}_t\right)\right]. \end{aligned}$$

- The dual problem is to minimize the upper bound with respect to all adapted super-martingale processes, i.e.,

$$\overline{V}_{t_0}(\mathbf{X}_{t_0}) = \inf_{\mathcal{M}\in\Pi} \left(\mathcal{M}_{t_0} + \mathbb{E}\left[\max_t \left(\frac{h_t}{B_t} - \mathcal{M}_t\right)\right]\right), \tag{12}$$

Upper bound continued

- With, $\mathcal{M}_{t_0} = 0$, we construct a martingale process as:

$$\mathcal{M}_{t_m}(n) = \sum_{i=0}^{m-1} \left[\frac{\tilde{G}^{\beta_{t_{i+1}}}\left(\mathbf{S}_{t_{i+1}}(n)\right)}{B_{t_{i+1}}} - \frac{\widehat{Q}_{t_i}\left(\mathbf{X}_{t_i}(n)\right)}{B_{t_i}} \right] \tag{13}$$

- The upper bound, $\overline{V}_{t_0}$, is then given by

$$\begin{aligned} \overline{V}_{t_0}(\mathbf{X}_{t_0}) &= \mathbb{E}\left[\max_t \left(\frac{h_t}{B_t} - \mathcal{M}_t \right) \right] \\ &= \frac{1}{N_L} \sum_{n=1}^{N_L} \max_{t_m} \left(\frac{h(\mathbf{X}_{t_m}(n))}{B_{t_m}} - \mathcal{M}_{t_m}(n) \right), \; t_m \in [t_0, \ldots, t_M] \end{aligned}$$

- Unlike regress now schemes, we don't need sub-simulation to estimate $\widehat{Q}_{t_i}$, as we know the closed form conditional expectation of $\tilde{G}^{\beta_{t_{i+1}}}$ is $\widehat{Q}_{t_i}$.

Hyperparameters

- We use Adam Kingma and Ba [2014] with initial leaning rate as 10^{-3}, as the optimizer for the weights update in the mini-batch gradient ascent algorithm.
- The batch size is chosen as one tenth of the total training points.
- For training the neural network corresponding to the monitoring date $t_M = T$, we initialize the weights and biases randomly with uniform random variables .

Hyper parameters continued

- As $\tilde{V}_{t_{m-1}} \approx \tilde{V}_{t_m}$ we transfer the final weights obtained from training the network for monitoring date t_m as the initial weights for training the network at t_{m-1}.
- In order to avoid over-fitting, we divide the training points into training set and validation set, in the ratio 0.7 to 0.3 respectively and use the mean squared error of the validation set as the early-stopping criteria with a patience of 6 epochs.
- We normalize the initial asset price to 1 and appropriately adjust the strike. This restricts the domain in which the network needs to be trained, something we found especially beneficial while training the network for max options.
- We use 50,000 training points generated using the GBM process under the risk-neutral measure.

Max Option (GBM Process)

S_0	RLNN Direct est. (s.e.)	RLNN Lower Bound. (s.e.)	RLNN Upper Bound (s.e.)	RLNN 95% CI	Literature 95% CI	Binomial
d=2 assets:						
90	8.078 (0.016)	8.071 (0.025)	8.086 (0.0003)	[8.062, 8.086]	[8.053, 8.082]	8.075
100	13.902 (0.017)	13.905 (0.028)	13.924 (0.0004)	[13.894, 13.924]	[13.892, 13.934]	13.902
110	21.346 (0.010)	21.352 (0.029)	21.353 (0.0002)	[21.341, 21.353]	[21.316, 21.359]	21.345
d=3 assets:						
90	11.282 (0.017)	11.295 (0.030)	11.303 (0.001)	[11.287, 11.303]	[11.265, 11.308]	11.29
100	18.702 (0.022)	18.688 (0.025)	18.715 (0.001)	[18.677, 18.715]	[18.661, 18.728]	18.69
110	27.572 (0.021)	27.554 (0.023)	27.592 (0.001)	[27.545, 27.592]	[27.512, 27.663]	27.58
d=5 assets:						
90	16.680 (0.063)	16.636 (0.044)	16.743 (0.003)	[16.624, 16.744]	[16.620, 16.653]	
100	26.177 (0.062)	26.141 (0.034)	26.268 (0.002)	[26.125, 26.270]	[26.115, 26.164]	
110	36.815 (0.042)	36.760 (0.045)	36.909 (0.003)	[36.744, 36.909]	[36.710, 36.798]	

Table: Bermudan option values for a call on the maximum of 2, 3 and 5 assets. For the two asset case we use 256 hidden units, for three assets 512 hidden units, while for five assets we use 1024 hidden units. The reference confidence interval for the two and three asset case are taken from Andersen and Broadie (2004), and for the five asset case from Broadie and Cao (2009)

Convergence with neurons

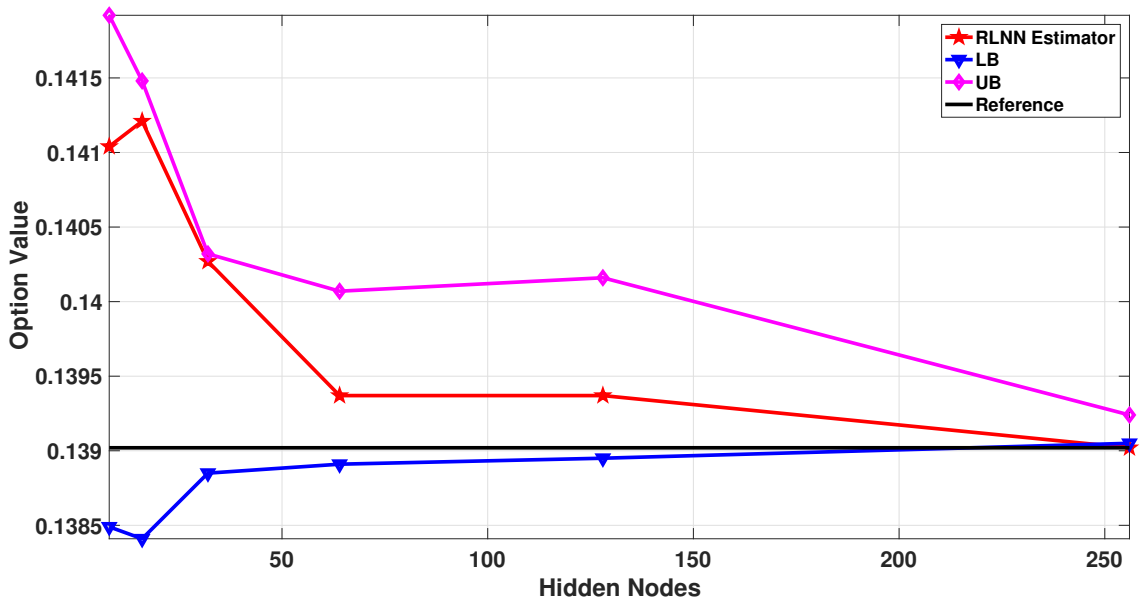


Figure: The upper and lower bound values for a Bermudan call on the maximum of two assets when an increasing number of hidden units are used in the first layer of RLNN. The reference value is obtained from a binomial tree and is equal to 13.902. Also plotted are the corresponding direct RLNN estimator values, i.e. $\tilde{V}_{t_0}$.

Questions

- Details on some of the aspects covered here can be found in Lokeshwar et al. [2019].

References

Douglas T Breeden and Robert H Litzenberger. Prices of state-contingent claims implicit in option prices. *Journal of business*, pages 621–651, 1978.

Peter Carr and Liuren Wu. Static hedging of standard options. *Journal of Financial Econometrics*, 12(1):3–46, 2013.

Martin B Haugh and Leonid Kogan. Pricing american options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vikranth Lokeshwar, Vikram Bhardawaj, and Shashi Jain. Neural network for pricing and universal static hedging of contingent claims. *Available at SSRN 3491209*, 2019.