Learning to price and hedge path-dependent derivatives

David Šiška[1,2]

This is joint work with Marc Sabate-Vidales[1] and Lukasz Szpruch[2,3].

Conference on Machine Learning in Finance
17th September 2019, Oxford

[1] Universitiy of Edinburgh
[2] Vega http://vega.xyz/
[3] The Alan Turing Institute

Talk Outline

i) Motivation from Vega,

ii) Machine learning algorithms for path-dependent derivatives:
   - Projection solver,
   - Martingale representation solver - direct,
   - Martingale representation solver - variance,
   - Martingale representation solver - control variate.

iii) Covergence:
   - On-line black box quality estimation with control variates and CLT,
   - Numerical results.
   - Some theory.

Vega

Vision[4]:

i) Decentralized derivatives exchange,
ii) Anyone can design a derivative using smart language of economic primitives and open a market by committing (financially) to market make,
iii) Markets are opened by default but can be voted down during proposal period by stakeholders.

Risk management challenge: not a spot exchange, people are trading promises.

How to safely margin the trades, in particular

i) What risk models,
ii) Robust calibration,
iii) **Efficient risk calculation.**

[4]See Danezis, Hrycyszyn, Mannerings, Rudolph and Š [1].

Margin calculation

We need:

i) Model to evaluate derivative liabilities at time $\tau > 0$ *given a real world scenario* (pricing in risk-neutral measure $\mathbb{Q}$)

ii) Model to move one step to the next possible closeout run time $\tau > 0$ (real-world measure $\mathbb{P}$).

iii) A coherent risk-measure $\rho = \rho^{\mathbb{P}}$ to establish the risk in a given (discounted) position X.

Minimum margin is, for position Ξ,

$$m_t^{\text{min}} := \rho^{\mathbb{P}} \left(\mathbb{E}^{\mathbb{Q}} \left[\Xi | \mathcal{F}_{t+\tau} \right] \right) .$$

Nested simulations I

Simulations of the risk drivers (asset processes, vol processes etc. $i = 1, \ldots, d$) under $\mathbb{P}$ denoted $x^{i,j}$ for $j = 1, \ldots, N$

$$m_t^{\min} \approx \frac{1}{\lambda} \frac{1}{N} \sum_{j=1}^{N} \left(-p_{t+\tau}^{j} | x^{i,j} \mathbb{1}_{p_{t+\tau}^{j} | x^{i,j} < -\mathrm{VaR}_{\lambda}^{N} | x^{i,j}} \right) .$$

Here

$$p_{t+\tau}^{j} | x^{i,j} \approx x^{0,j} \frac{1}{\tilde{N}} \sum_{k=1}^{\tilde{N}} \frac{\xi_k^j | x^{i,j}}{x_k^{0,j} | x^{i,j}}$$

i.e. for each j we need to simulate $k = 1, \ldots \tilde{N}$ realizations of the discounted payoff $\xi_k^j | x^{i,j}$ under $\mathbb{Q}$.

Killer:

- Need $N\tilde{N}$ simulations.
- The faster this can be done the lower $\tau > 0$ and the lower margin i.e. higher leverage.

Nested simulations II

More efficient methods:

- Regression based methods, see [1].
- Multi-level approach, see [2].
- **Machine learning based approach.**

[1] M. Broadie and Y. Du and C. C. Moallemi. Risk Estimation via Regression. *Operations Research*, 63(5), 1077–1097, 2015.

[2] M. Giles and A.-L. Haji-Ali. Multilevel Nested Simulation for Efficient Risk Estimation. *arXiv*:1802.05016, 2018.

Feed-forward neural networks

Layers L, size of layer k given by $l_k \in \mathbb{N}$.

i) Space of parameters

$$\Pi = (\mathbb{R}^{l^1 \times l^0} \times \mathbb{R}^{l^1}) \times (\mathbb{R}^{l^2 \times l^1} \times \mathbb{R}^{l^2}) \times \cdots \times (\mathbb{R}^{l^L \times l^{L-1}} \times \mathbb{R}^{l^L}),$$

ii) The network *parameters*

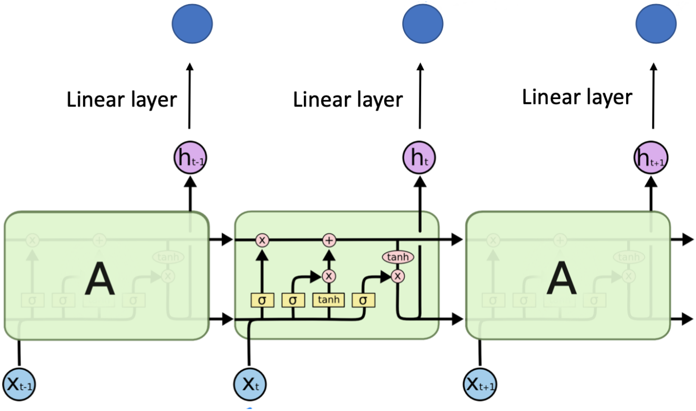
$$\Psi = ((\alpha^1, \beta^1), \ldots, (\alpha^L, \beta^L)) \in \Pi .$$

iii) Reconstruction $\mathcal{R}\Psi : \mathbb{R}^{l^0} \to \mathbb{R}^{l^L}$ given recursively, for $x_0 \in \mathbb{R}^{l^0}$, by $z_0 \in \mathbb{R}^{l^0}$, by

$$(\mathcal{R}\Psi)(z^0) = \alpha^L z^{L-1} + \beta^L , \quad z^k = \varphi^{l^k}(\alpha^k z^{k-1} + \beta^k) , k = 1, \ldots, L-1 .$$

LTSM neural networks

Diagram[5]:



We will still denote its parameters $\Psi \in \Pi$ and think of LSTM net as

$$(\mathcal{R}\Psi) : \{0, 1, \dots, N_{\text{steps}}\} \times (\mathbb{R}^d)^{1+N_{\text{steps}}} \to (\mathbb{R}^{d'})^{1+N_{\text{steps}}} .$$

[5]From Christopher Olah https://colah.github.io/

General pricing / hedging setup

Path-dependent (discounted) payoff:

$$\Xi_T := G\Big((X_s)_{s\in[0,T]}\Big), \quad G : C([0,T];\mathbb{R}^d) \to \mathbb{R} \text{ given.}$$

Price in (some) r.n. measure is

$$V_t = \mathbb{E}\left[\Xi_T|\mathcal{F}_t\right] = \mathbb{E}\left[G((X_s)_{s\in[0,T]})|(X_s)_{s\in[0,t]}\right]$$

Assume $\mathbb{F} := (\mathcal{F}_s)_{s\in[0,t]}$ is generated by d'-dim Wiener process and $\sigma((X_s)_{s\le t}) = \mathcal{F}_t$.

Take a partition of $[0, T]$ denoted

$$\pi := \{t = t_0 < \cdots < t_{N_{\text{steps}}} = T\}$$

and consider a discretization of $(X_s)_{s\in[0,T]}$ by $(X^\pi_{t_i})_{i=0}^{N_{\text{steps}}}$.

Learning L^2-orthogonal projection

Theorem 1

Let $\mathcal{X} \in L^2(\mathcal{F})$. Let $\mathcal{G} \subset \mathcal{F}$ be a sub σ-algebra. There exists a unique random variable $\mathcal{Y} \in L^2(\mathcal{G})$ such that $\mathcal{Y} = \mathbb{E}[\mathcal{X}|\mathcal{G}]$ and

$$\mathbb{E}[|\mathcal{X} - \mathcal{Y}|^2] = \inf_{\eta \in L^2(\mathcal{G})} \mathbb{E}[|\mathcal{X} - \eta|^2].$$

Take $\mathcal{X} := \Xi_T$, $\mathcal{G} := \mathcal{F}_t = \sigma((X_s)_{s\in[0,t]})$ so

$$V_t = \operatorname*{argmin}_{\eta \in L^2(\mathcal{F}_t)} \mathbb{E}[|\Xi_T - \eta|^2].$$

Doob–Dynkin Lemma implies that for any $\eta \in L^2\left(\sigma((X_{s\wedge t})_{s\in[0,T]})\right)$ there is $h : [0, T] \times C([0, T]; \mathbb{R}^d) \to \mathbb{R}$ measurable s.t. $\eta = h(t, (X_{s\wedge t})_{s\in[0,T]})$.

So

$$\mathbb{E}[|\Xi_T - V_t|^2] = \inf_h \mathbb{E}[|\Xi_T - h(t, (X_{s\wedge t})_{s\in[0,T]})|^2]$$

Infimum over measurable functions $h : [0, T] \times C([0, T]; \mathbb{R}^d) \to \mathbb{R}$

Learning L^2-orthogonal projection II

Network:

$$\mathbb{E}[|\Xi_T - V_t|^2] \approx \inf_{\Psi \in \Pi} \mathbb{E}[|\Xi_T^\pi - (\mathcal{R}\Psi)(t, (X^\pi_{t_i \wedge t})_{i=0,1,\ldots,N_{\text{steps}}})|^2]$$

and so

$$V_t \approx \widehat{\operatorname{argmin}}_{\Psi \in \Pi} \mathbb{E}[|\Xi_T^\pi - (\mathcal{R}\Psi)(t, (X^\pi_{t_i \wedge t})_{i=0,1,\ldots,N_{\text{steps}}})|^2]\,.$$

Here:

i) Ξ_T^π, X^π denote numerical approximations.

ii) Hat over arg min denotes the we will use SGD (and so won't necessarily find actual minimum).

Learning L^2-orthogonal projection III

Algorithm: Projection solver

Initialisation: $\Psi^0 \in \Pi$, $N_{\text{trn}} \in \mathbb{N}$ large

for $i : 1 : N_{\text{trn}}$ **do**

Generate $(x^i_{t_n})_{n=0}^{N_{\text{steps}}}$ from $(X_s)_{s\in[0,T]}$.

Compute $\Xi^{\pi,i}_T := G((X^{\pi,i}_{t_k})_{k=0,1,\dots,N_{\text{steps}}})$.

end for

Starting with Ψ^0, use SGD to find $\Psi^{\diamond,N_{\text{trn}}}$, where

$$\Psi^{\diamond,N_{\text{trn}}} = \widehat{\underset{\theta}{\operatorname{argmin}}} \frac{1}{N_{\text{trn}}} \sum_{i=1}^{N_{\text{trn}}} \sum_{k=0}^{N_{\text{steps}}-1} [|\Xi^{\pi,i}_T - (\mathcal{R}\Psi)(t_k, (x^i_{t_k \wedge t_j})_{j=0,1,\dots,N_{\text{steps}}})|^2]$$

return $\Psi^{\diamond,N_{\text{trn}}}$.

- Works with incomplete markets but no (direct) access to hedging strategy.
- In Markovian setting automatic differentiation gives hedging strategy.

Learning martingale representation I

Assume complete market ($d = d'$). Then (classical) martingale representation: $\exists Z$ which is $\mathbb{F}$ adapted and

$$V_t = \Xi_T - \int_t^T Z_s \, dW_s \, .$$

With functional Itô calculus

$$V_t = G\big((X_s)_{s\in[0,T]}\big) - \int_t^T \nabla_\omega G\big((X_{r\wedge s})_{r\in[0,T]}\big) \, dX_s \, .$$

See Cont and Fournié [2].

Learning martingale representation II

Aim: use Monte Carlo and Machine Learning to obtain Z. Why?

i) You also get V.
ii) You get the hedging strategy.

With the partition π of $[0,T]$ in mind

$$V_{t_m} = V_{t_{m+1}} + \int_{t_m}^{t_{m+1}} Z_s \, dW_s \;\; \text{for } m = 0, 1, \ldots, N_{\text{steps}} - 1,$$
$$V_{t_{N_{\text{steps}}}} = \Xi_T \, .$$

Approximate by **two** networks with (possibly) different size / architecture $\Psi \in \Pi^\Psi$, $\Phi \in \Pi^\Phi$:

$$V_{t_m} \approx (\mathcal{R}\Psi)\left(t_m, \left(X^\pi_{s\wedge t_m}\right)_{s\in[0,T]}\right) \;\text{and}\; Z_{t_m} \approx (\mathcal{R}\Phi)\left(t_m, \left(X^\pi_{s\wedge t_m}\right)_{s\in[0,T]}\right) .$$

Get consistency condition:

$$0 \approx (\mathcal{R}\Psi)\left(t_{m+1}, \left(X^\pi_{s\wedge t_{m+1}}\right)_{s\in[0,T]}\right) - (\mathcal{R}\Psi)\left(t_m, \left(X^\pi_{s\wedge t_m}\right)_{s\in[0,T]}\right)$$
$$+ (\mathcal{R}\Phi)\left(t_m, \left(X^\pi_{s\wedge t_m}\right)_{s\in[0,T]}\right)\left(W_{t_{m+1}} - W_{t_m}\right) =: \mathcal{E}^\pi_{m+1}(\Psi, \Phi) \, .$$

Learning martingale representation III

Initialisation: Ψ^0, Φ^0, N_{trn}

for $i : 1 : N_{\text{trn}}$ **do**

Generate samples $(w^i_{t_n})_{n=0}^{N_{\text{steps}}}$, use these to generate $(x^{\pi,i}_{t_n})_{n=0}^{N_{\text{steps}}}$ by approximating $X = (X_s)_{s\in[0,T]}$, generate $\Xi^{\pi,i}_T$.

end for

Starting with Ψ^0, Φ^0, use SGD to find $(\theta^{\diamond,N_{\text{trn}}}, \Psi^{\diamond,N_{\text{trn}}})$ where

$$
\begin{aligned}
(\Psi^{\diamond,N_{\text{trn}}}, \Phi^{\diamond,N_{\text{trn}}}) :=& \widehat{\operatorname{argmin}}_{(\Psi,\Phi)} \frac{1}{N_{\text{trn}}} \sum_{i=1}^{N_{\text{trn}}} \Bigg[\left| \Xi^{\pi,i}_T - (\mathcal{R}\Psi)(T, (x^{\pi,i}_{t_k})_{k=0,1,\ldots,N_{\text{steps}}}) \right|^2 \\
& + \sum_{m=0}^{N_{\text{steps}}-1} \left| \mathcal{E}^{\pi,i}(\Psi,\Phi)_{m+1} \right|^2 \Bigg] , \\
\mathcal{E}^{\pi,i}(\Psi,\Phi)_{m+1} :=& (\mathcal{R}\Psi)\left(t_{m+1}, \left(x^{\pi,i}_{s\wedge t_{m+1}}\right)_{s\in[0,T]}\right) - (\mathcal{R}\Psi)\left(t_m, \left(x^{\pi,i}_{s\wedge t_m}\right)_{s\in[0,T]}\right) \\
& + (\mathcal{R}\Phi)\left(t_m, \left(X^{\pi,i}_{s\wedge t_m}\right)_{s\in[0,T]}\right)\left(w^i_{t_{m+1}} - w^i_{t_m}\right) .
\end{aligned}
$$

return $(\Psi^{\diamond,N_{\text{trn}}}, \Phi^{\diamond,N_{\text{trn}}})$.

Learning martingale representation - minimizing variance I

Recall

$$V_t = \Xi_T - \int_t^T Z_s \, dW_s \, .$$

Monte Carlo: say Z^i, W^i, Ξ^i_T are iid samples of Z, W, Ξ. Then for

$$\mathcal{V}_t^N := \frac{1}{N} \sum_{i=1}^{N} \left(\Xi^i_T - \int_t^T Z^i_s \, dW^i_s \right)$$

we have

i) Unbiased estimator: $\mathbb{E}\left[\mathcal{V}_t^N | \mathcal{F}_t\right] = V_t$,

ii) Zero variance estimator: $\mathbb{V}\mathrm{ar}\left[\mathcal{V}_t^N | \mathcal{F}_t\right] = 0$.

Use variance as objective in learning.

Learning martingale representation - minimizing variance II

Initialisation: Φ^0, N_{trn}

for $i : 1 : N_{\text{trn}}$ **do**

Generate the samples of Wiener process increments $(\Delta w_{t_n})_{n=1}^{N_{\text{steps}}}$.

Use $(\Delta w_{t_n})_{n=1}^{N_{\text{steps}}}$ to generate samples $(x^i_{t_n})_{n=0}^{N_{\text{steps}}}$ by simulating the process X.

Use these to compute Ξ^i_T.

end for

Starting with Φ^0 use SGD to approximate $\Phi^{\diamond,N_{\text{trn}}}$ with objective

$$\Phi^{\diamond,N_{\text{trn}}} := \widehat{\operatorname{argmin}_{\Phi\in\Pi}} \frac{1}{N_{\text{trn}}} \sum_{i=1}^{N_{\text{trn}}} \left(\Xi^i_T - \mathcal{V}^{\pi,N_{\text{steps}},i}(\Phi) \right)^2 ,$$

where

$$\mathcal{V}^{\pi,N_{\text{steps}},i}(\Phi) := \sum_{m=0}^{N_{\text{steps}}-1} (\mathcal{R}\Phi)\left(t_m, (x^i_{t_k \wedge t_m})_{k=0,1,\ldots,N_{\text{steps}}}\right) \left(w^i_{t_{m+1}} - w^i_{t_m}\right).$$

return $\Phi^{\diamond,N_{\text{trn}}}$.

Learning martingale representation - control variate I

Recall that with

$$\mathcal{V}_t^N := \frac{1}{N}\sum_{i=1}^{N}\left(\Xi_T^i - \int_t^T Z_s^i\, dW_s^i\right)$$

we have $\mathbb{E}\left[\mathcal{V}_t^N | \mathcal{F}_t\right] = V_t$, $\mathbb{V}\mathrm{ar}\left[\mathcal{V}_t^N | \mathcal{F}_t\right] = 0$.

With

$$\mathcal{V}^{N,\pi} := \frac{1}{N}\sum_{i=1}^{N}\left(\Xi_T^i - \sum_{k=0}^{N_{\text{steps}}-1} Z_{t_k}^i \left(W_{t_{k+1}}^i - W_{t_k}\right)\right)$$

we have $\mathbb{E}\left[\mathcal{V}^{N,\pi} | \mathcal{F}_t\right] \approx V_0$, $\mathbb{V}\mathrm{ar}\left[\mathcal{V}^{N,\pi} | \mathcal{F}_t\right] > 0$ but small.

Aim: Use stochastic integral as control variate.

Learning martingale representation - control variate II

With

$$\mathcal{V}_t^{N,\pi,\Phi,\lambda} := \frac{1}{N}\sum_{i=1}^{N}\left(\Xi_T^i - \lambda \underbrace{\sum_{k=0}^{N_{\text{steps}}-1} (\mathcal{R}\Phi)(t_k, (X^i_{t_j\wedge t_k})_{j=0,1,\dots,N_{\text{steps}}})\,\Delta W^i_{t_{k+1}}}_{=:M^\Phi}\right).$$

The optimal coefficient $\lambda^{*,\Phi}$ that minimises the variance is

$$\lambda^{*,\Phi} = \frac{\mathbb{C}\text{ov}[\Xi_T, M^\Phi]}{\mathbb{V}\text{ar}[M^\Phi]}.$$

Variance reduction factor is $\frac{1}{1-(\rho^{\pi,\Phi})^2}$ where

$$\rho^{\pi,\Phi} = \frac{\mathbb{C}\text{ov}(\Xi_T, M^\Phi)}{\sqrt{\mathbb{V}\text{ar}[\Xi_T]\mathbb{V}\text{ar}[M^\Phi]}}.$$

Objective:

$$\Phi^{\diamond,\pi} := \operatorname*{argmin}_{\Phi\in\Pi}\left[1 - \left(\rho^{\pi,\Phi}\right)^2\right].$$

Learning martingale representation - control variate III

Initialisation: Φ^0, N_{trn}

for $i : 1 : N_{\mathrm{trn}}$ **do**

Generate the samples of Wiener process increments $(\Delta w_{t_n})_{n=1}^{N_{\mathrm{steps}}}$.
Use $(\Delta w_{t_n})_{n=1}^{N_{\mathrm{steps}}}$ to generate samples $(x^i_{t_n})_{n=0}^{N_{\mathrm{steps}}}$ by simulating the process X. Use $(x^i_{t_n})_{n=0}^{N_{\mathrm{steps}}}$ to compute Ξ^i_T.

end for

Starting with Φ^0 use SGD to find

$$\Phi^{\diamond,N_{\mathrm{trn}}} := \underset{\Phi\in\Pi}{\widehat{\mathrm{argmin}}} \left[1 - \left(\frac{\sum_{i=1}^{N_{\mathrm{trn}}}(M^{i,\Phi} - \overline{M^{\Phi}})(\Xi^i_T - \overline{\Xi_T})}{\left(\sum_{i=1}^{N_{\mathrm{trn}}}(\Xi_T - \overline{\Xi_T})^2 \sum_{i=1}^{N_{\mathrm{trn}}}(M^{i,\Phi} - \overline{M^{\Phi}})^2\right)^{1/2}}\right)^2\right],$$

where $\overline{\Xi_T} := \sum_{i=1}^{N_{\mathrm{trn}}} \Xi^i_T$, $\overline{M^{\Phi}} := \sum_{i=1}^{N_{\mathrm{trn}}} M^{i,\Phi}$ and

$$M^{i,\Phi} := \sum_{i=1}^{N_{\mathrm{trn}}} \sum_{k=0}^{N_{\mathrm{steps}}-1} (\mathcal{R}\Phi)(t_k, (X^i_{t_j\wedge t_k})_{j=0,1,\ldots,N_{\mathrm{steps}}})\,\Delta W^i_{t_{k+1}}\,.$$

return $\Phi^{\diamond,N_{\mathrm{trn}}}$.

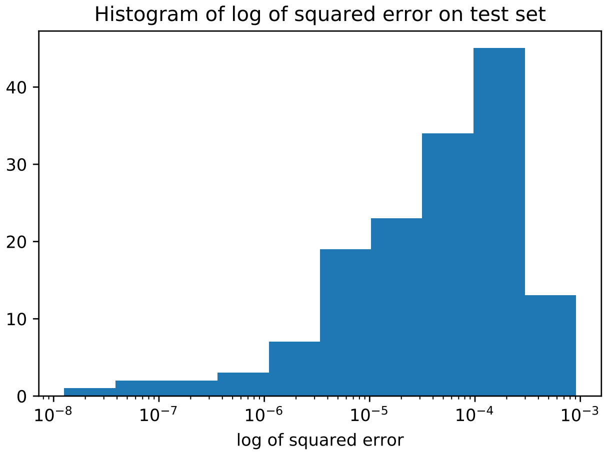
Sanity check: exchange option

Method	Confidence Interval Variance	Confidence Interval Estimator
Monte Carlo	$[5.95 \times 10^{-7}, 1.58 \times 10^{-6}]$	$[0.1187, 0.1195]$
Martingale rep. - var. min.	$[4.32 \times 10^{-9}, 1.14 \times 10^{-8}]$	$[0.11919, 0.11926]$
Martingale rep. - corr. max.	$[2.30 \times 10^{-9}, 6.12 \times 10^{-8}]$	$[0.11920, 0.11924]$
Martingale rep. - two networks	$[4.13 \times 10^{-9}, 1.09 \times 10^{-8}]$	$[0.11919, 0.11926]$
MC + CV Margrabe	$[3.10 \times 10^{-9}, 8.23 \times 10^{-9}]$	$[0.11919, 0.11925]$

Error is time discretization arising even when exact form of martingale representation term is used.

Robustness example

Exchange option (Markovian) in $d = 5$ with random covariance matrix (parametric approximation).



Clearly, there are input combinations where error is 10^{-3} rather than 10^{-6}.

On-line quality estimation with control variates and CLT

Say SGD converges to $\Phi^{\diamond,\pi,N}$.

Approximation of martingale representation term gives access to control variate with 0 expectation:

$$\sum_{k=0}^{N_{\text{steps}}-1} (\mathcal{R}\Phi)(t_k, (X^i_{t_j \wedge t_k})_{j=0,1,\ldots,N_{\text{steps}}})\, \Delta W^i_{t_{k+1}} =: M^{i,\Phi}\,.$$

i) n-samples, evaluate $\frac{1}{n}\sum_{i=1}^n M^{i,\Phi}$. If "far" from 0 things went wrong,

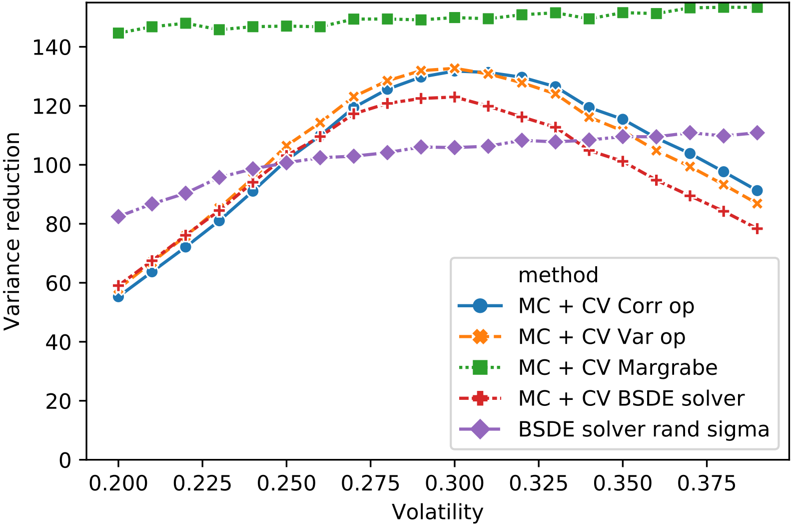
ii) evaluate

$$\rho^{\pi,\Phi,n} = \frac{\sum_{i=1}^{N_{\text{trn}}}(M^{i,\Phi} - \overline{M^{\Phi}})(\Xi^i_T - \overline{\Xi_T})}{\left(\sum_{i=1}^{N_{\text{trn}}}(\Xi_T - \overline{\Xi_T})^2 \sum_{i=1}^{N_{\text{trn}}}(M^{i,\Phi} - \overline{M^{\Phi}})^2\right)^{1/2}}\,.$$

If "far" from 1 things went wrong.

Variance reduction - outside parameter range

Training for fixed volatility of 30% versus parametric (constant cost).



High dimensional Markovian example

Take $d = 100$,

$$g(S) := \max\left(0, S^1 - \frac{1}{d-1}\sum_{i=2}^{d} S^i\right).$$

Method	Confidence Interval Variance	Confidence Interval Estimator
Monte Carlo	$[2.03 \times 10^{-7}, 5.41 \times 10^{-7}]$	$[0.0845, 0.0849]$
Martingale rep. - var. min.	$[4.13 \times 10^{-9}, 1.09 \times 10^{-8}]$	$[0.08484, 0.08490]$
Martingale rep. - corr. max.	$[3.80 \times 10^{-9}, 1.0 \times 10^{-8}]$	$[0.08487, 0.08493]$
Martingale rep. - two networks	$[5.32 \times 10^{-9}, 1.41 \times 10^{-8}]$	$[0.08485, 0.8492]$

Table: Results on exchange option problem on 100 assets, Monte Carlo vs. control variate with 10^6 samples.

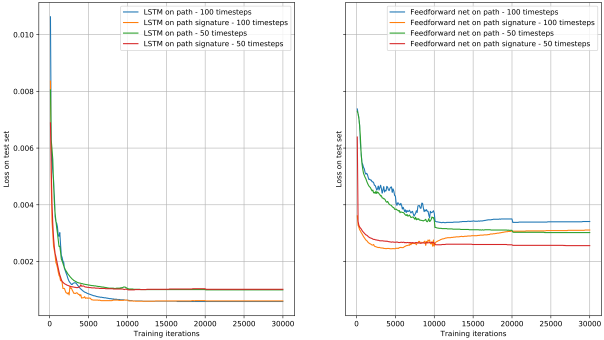
Model trained for any initial asset price (log-normal)[6].

All results from Sabate-Vidales, Š, Szpruch [3].

[6] With all parameters fixed we get variance reduction factor $5 \cdot 10^5$ - too easy.

LSTM vs FFN

Using the "martingale representation - minimizing variance" method:



i) "Lookback option" $[\max_{t \le T}(X_t^1 + X_t^2) - (X_T^1 + X_T^2)]_+$.

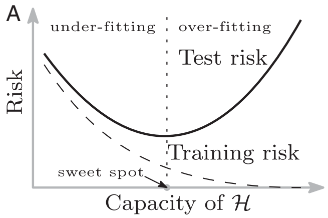
ii) LSTM training objective converges to minimum error due to time discretization.

iii) Signatures help training for LSTM (but not decisive).

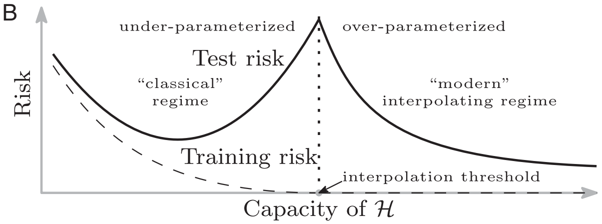
iv) FFN don't learn in this setup with SGD.

Choosing network sizes

Classical view:



Modern view from Belkin, Hsu, Ma and Mandal [4]:



Conclusion: Network should have many more parameters than training data points.

Representation theorems

- Hornik [5]: "any level of accuracy in approximation of a continuous function on a compact set is achievable by sufficiently wide one hidden layer feedforward network **with appropriate parameters**."
- Hornung et al. [6] and related: "solutions to many PDEs (e.g. Black–Scholes) can be approximated to any accuracy by a sufficiently deep and wide feedforward network **with appropriate parameters** without suffering from curse of dimensionality ."
- But does SGD reach such parameters? Supervised learning is **non-convex**.

Non-covex minimization problem

With $\hat{\varphi}(x,z) = \beta\varphi(\alpha\cdot z)$ for $x = (\alpha,\beta) \in (\mathbb{R}\times\mathbb{R}^D)^n$, we should minimize,

$$(\mathbb{R}\times\mathbb{R}^D)^n \ni x \mapsto \underbrace{\int_{\mathbb{R}\times\mathbb{R}^D} \Phi\left(y - \frac{1}{n}\sum_{i=1}^{n}\hat{\varphi}(x^i,z)\right)\nu(dy,dz)}_{=:F(x)} + \frac{\bar{\sigma}^2}{2}\underbrace{|x|^2}_{=:U(x)},$$

which is non-convex.

Supervised learning:

i) $\Phi : \mathbb{R} \to \mathbb{R}^+$ given, convex, e.g. $\Phi(x) = |x|^2$

ii) sample learning data from measure $\nu \in \mathcal{P}(\mathbb{R}\times\mathbb{R}^D)$ i.e. "big data"

iii) aim is to find optimal network parameters w.r.t. Φ.

Mean-field limit and convexity

Assume that x^i are i.i.d. samples from some measure $m \in \mathcal{P}(\mathbb{R}^d)$. Due to law of large numbers, for each fixed $z \in \mathbb{R}^D$ we have

$$\frac{1}{n}\sum_{i=1}^{n} \hat{\varphi}(x^i, z) \to \int_{\mathbb{R}^d} \hat{\varphi}(x,z)\, m(dx) \text{ as } n \to \infty\,.$$

The search for the optimal measure $m^* \in \mathcal{P}(\mathbb{R}^d)$ amounts to minimizing

$$\mathcal{P}(\mathbb{R}^d) \ni m \mapsto \int_{\mathbb{R}\times\mathbb{R}^D} \Phi\left(y - \int_{\mathbb{R}^d} \hat{\varphi}(x,z)\, m(dx)\right) \nu(dy,dz) =: F(m),$$

which is convex as long as Φ is i.e. for any $m, m' \in \mathcal{P}(\mathbb{R}^d)$, $\alpha \in [0,1]$ we have

$$F((1-\alpha)m + \alpha m') \leq (1-\alpha)F(m) + \alpha F(m')\,.$$

Observed in the pioneering works of Mei, Misiakiewicz and Montanari [7], Chizat and Bach [8] as well as Rotskoff and Vanden-Eijnden [9].

Study $V^\sigma(m) := F(m) + \frac{\sigma^2}{2} H(m)$**.**

Mean-field Langevin dynamics

For some $F : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ and a Gibbs measure g:

$$g(x) = e^{-U(x)} \text{ with } U \text{ s.t. } \int_{\mathbb{R}^d} e^{-U(x)}\, dx = 1$$

consider mean-field Langevin equation:

$$\begin{cases} dX_t = -\left(D_m F(m_t, X_t) + \frac{\sigma^2}{2} \nabla U(X_t)\right) dt + \sigma dW_t\,, \; t \in [0, \infty)\,, \\ m_t = \text{Law}(X_t)\,, \; t \in [0, \infty)\,. \end{cases} \tag{1}$$

Corresponding gradient flow:

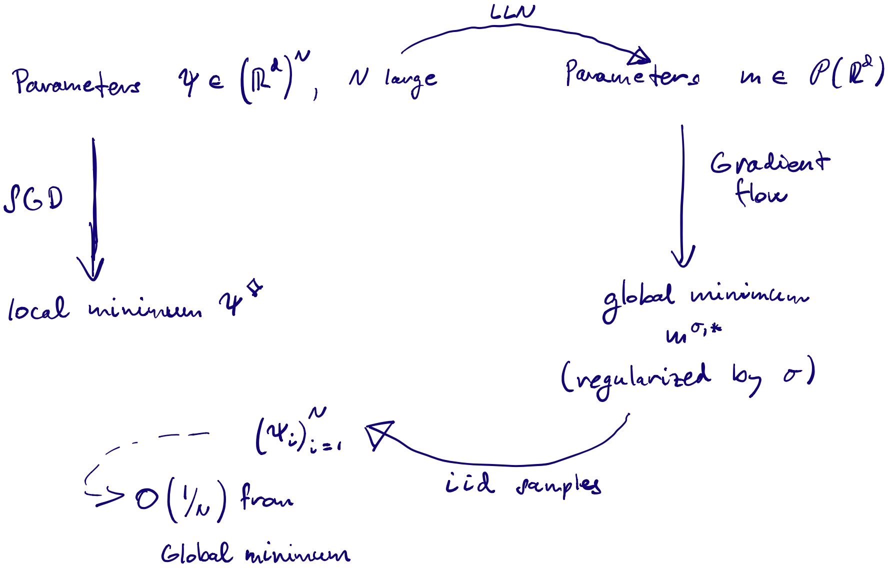
$$\partial_t m = \nabla \cdot \left(\left(D_m F(m, \cdot) + \frac{\sigma^2}{2} \nabla U \right) m + \frac{\sigma^2}{2} \nabla m \right) \text{ on } (0, \infty) \times \mathbb{R}^d\,.$$

If $m' \in \mathcal{I}_\sigma$ where

$$\mathcal{I}_\sigma := \left\{ m \in \mathcal{P}(\mathbb{R}^d) : \frac{\delta F}{\delta m}(m, \cdot) + \frac{\sigma^2}{2} \log(m) + \frac{\sigma^2}{2} U \text{ is a constant} \right\}$$

then $m' = \arg\min_{m \in \mathcal{P}(\mathbb{R}^d)} V^\sigma$.

Mean-field results



Details is Hu, Ren, Š, Szpruch [10].

Conclusions:

- Machine learning can approximate high dimensional and parametric models of pricing and hedging.
- To get learning convergence requires careful design and tuning.
- Unanswered questions about convergence and robustness.
- Can be partially mitigated by on-line performance tests based on control variates.
- LSTM for path dependent.
- Separate data generation from network training.

Outlook - Research

- Do machine learning models in finance suffer from adversarial attacks? i.e. can we find inputs where trained network fails spectacularly?
- Signatures in high dimension as dimension reduction method .
- Comparisons with existing algorithms for path-dependent derivatives e.g. Cont, Lu [11].

Outlook - Vega

- Invitation only "Nicenet" launching Q4 2019 (get in touch to get access).
- Working on smart product language coming in 2020.
- Public "Testnet" in 2020 (still no real money).
- Ongoing research: distributed model calibration, better liquidity pricing, market making stake modelling, . . .

References I

Code available: `https://github.com/marcsv87/Deep-PDE-Solvers`.

[1] Danezis, G., Hrycyszyn, D., Mannerings, B., Rudolph, T., and Šiška, D. Vega Protocol. `https://vegaprotocol.io/papers/vega-protocol-whitepaper.pdf` (2018).

[2] Cont, R. and Fournié, D.-A. (2013) Functional Itô calculus and stochastic integral representation of martingales. *Ann. Probab.,* **41**(1), 109–133.

[3] Sabate-Vidales, M., Šiška, D., and Szpruch, L. (2018) Unbiased deep solvers for parametric PDEs. *arXiv:1810.05094,*.

[4] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019) Reconciling modern machine-learning practice and the classical bias-variance trade-off. *PNAS,* **116**(32), 15849–15854.

[5] Hornik, K. (1991) Approximation capabilities of multilayer feedforward networks. *Neural Networks,* **4**(2), 251–257.

[6] Grohs, P., Hornung, F., Jentzen, A., and von Wurstemberger, P. (2018) A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. *arXiv:1809.02362,*.

[7] Mei, S., Montanari, A., and Nguyen, P.-M. (2018) A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences,* **115**(33), E7665–E7671.

[8] Chizat, L. and Bach, F. (2018) On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems* pp. 3040–3050.

[9] Rotskoff, G. M. and Vanden-Eijnden, E. (2018) Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv:1805.00915,*.

[10] Hu, K., Ren, Z., Šiška, D., and Szpruch, L. (2019) Mean-Field Langevin Dynamics and Energy Landscape of Neural Networks. *arXiv:1905.07769,*.

[11] Cont, R. and Lu, Y. (2016) Weak approximation of martingale representations. *Stochastic Process. Appl.,* **126**(3), 857–882.