Robust pricing and hedging via neural SDEs

Lukasz Szpruch
University of Edinburgh, The Alan Turing Institute, London

joint work with: David Siska, Marc Sabate-Vilades @UoE
Zan Zuric and Antoine Jacquier @ICL

Outline

- Robust pricing and hedging with neural SDEs, [Gierjatowicz et al., 2020].
- Unbiased approximation of parametric path dependent PDEs, [Vidales et al., 2018].
- Neural ODEs via Relaxed Optimal Control. A perspective on deep recurrent neural networks, [Jabir et al., 2019].

Supervised Learning

- Data: $(\xi_i, \zeta_i)_{i=1}^n \sim \mathbb{P} \in \mathcal{P}(\mathbb{R} \times \mathbb{R}^d)$. $\mathbb{P}$ unknown

Supervised Learning

- Data: $(\xi_i, \zeta_i)_{i=1}^n \sim \mathbb{P} \in \mathcal{P}(\mathbb{R} \times \mathbb{R}^d)$. $\mathbb{P}$ unknown
- Goal: Find $f : \mathbb{R}^d \to \mathbb{R}$ that minimises population risk

Supervised Learning

- Data: $(\xi_i, \zeta_i)_{i=1}^n \sim \mathbb{P} \in \mathcal{P}(\mathbb{R} \times \mathbb{R}^d)$. $\mathbb{P}$ unknown
- Goal: Find $f : \mathbb{R}^d \to \mathbb{R}$ that minimises population risk
- Population Risk. Fix $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$

$$\mathcal{R}(f) := \mathbb{E}\left[\ell(\zeta, f(\xi))\right]$$

Classical view

- Fix $\mathcal{F} := \{f(\cdot, \theta) : \theta \in \Theta \in \mathbb{R}^p\}$
- Empirical Risk Minimisation

$$\min_{f \in \mathcal{F}} \mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(\zeta_i, f(\xi_i))$$

Classical view

- Fix $\mathcal{F} := \{f(\cdot, \theta) : \theta \in \Theta \in \mathbb{R}^p\}$
- Empirical Risk Minimisation

$$\min_{f \in \mathcal{F}} \mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(\zeta_i, f(\xi_i))$$

- Uniform convergence

$$\sup_{f \in \mathcal{F}} |\mathcal{R}_n(f) - \mathcal{R}(f)| \leq \epsilon(n, \mathcal{F})$$

Classical view

- Fix $\mathcal{F} := \{f(\cdot, \theta) : \theta \in \Theta \in \mathbb{R}^p\}$
- Empirical Risk Minimisation

$$\min_{f \in \mathcal{F}} \mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(\zeta_i, f(\xi_i))$$

- Uniform convergence

$$\sup_{f \in \mathcal{F}} |\mathcal{R}_n(f) - \mathcal{R}(f)| \leq \epsilon(n, \mathcal{F})$$

- convex optimisation . Take $\mathcal{F}$,ℓ so that ERM is convex

Classical view

- Fix $\mathcal{F} := \{f(\cdot, \theta) : \theta \in \Theta \in \mathbb{R}^p\}$
- Empirical Risk Minimisation

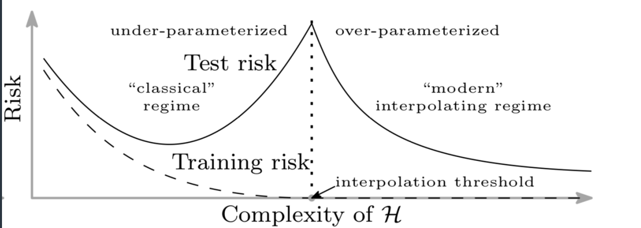
$$\min_{f \in \mathcal{F}} \mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(\zeta_i, f(\xi_i))$$

- Uniform convergence

$$\sup_{f \in \mathcal{F}} |\mathcal{R}_n(f) - \mathcal{R}(f)| \leq \epsilon(n, \mathcal{F})$$

- convex optimisation . Take $\mathcal{F}$,ℓ so that ERM is convex
- ... non of these works for deep learning....

New era of overparameterized statistical models ?



From Belkin. et.al. [Belkin et al., 2018].

- Need for new theory to study generalisation error. Classical Vapnik dimension and Rademacher complexity doesn't help.
- Overparametrised models can be optimal in the high signal-to-noise ratio regime Montanari et.al [Mei and Montanari, 2019]
- Implicit Regularisation [Heiss et al., 2019], [Neyshabur et al., 2017]

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.
- We can solve (a reasonable class of) high dimensional PDEs: [Ruf and Wang, 2020, Jacquier and Oumgari, 2019, Vidales et al., 2018]

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.
- We can solve (a reasonable class of) high dimensional PDEs: [Ruf and Wang, 2020, Jacquier and Oumgari, 2019, Vidales et al., 2018]
- [Horvath et al., 2019] showed that good approximation of the parametric pricing operator is all what's needed to calibrate

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.
- We can solve (a reasonable class of) high dimensional PDEs: [Ruf and Wang, 2020, Jacquier and Oumgari, 2019, Vidales et al., 2018]
- [Horvath et al., 2019] showed that good approximation of the parametric pricing operator is all what's needed to calibrate
- Deep hedging: [Buehler et al., 2019]

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.
- We can solve (a reasonable class of) high dimensional PDEs: [Ruf and Wang, 2020, Jacquier and Oumgari, 2019, Vidales et al., 2018]
- [Horvath et al., 2019] showed that good approximation of the parametric pricing operator is all what's needed to calibrate
- Deep hedging: [Buehler et al., 2019]

and can be used to build novel (econometrics) models:

- Market generators via (conditional) generative modelling [Kondratyev and Schwarz, 2019, Buehler et al., 2020, Henry-Labordere, 2019, Xu et al., 2020, Ni et al., 2020, Wiese et al., 2020]

Applications to Quant Finance

Deep nets enhance numerical approximation:

- [Hernandez, 2016] used neural networks to learn the calibration map from market data directly to model parameters.
- We can solve (a reasonable class of) high dimensional PDEs: [Ruf and Wang, 2020, Jacquier and Oumgari, 2019, Vidales et al., 2018]
- [Horvath et al., 2019] showed that good approximation of the parametric pricing operator is all what's needed to calibrate
- Deep hedging: [Buehler et al., 2019]

and can be used to build novel (econometrics) models:

- Market generators via (conditional) generative modelling [Kondratyev and Schwarz, 2019, Buehler et al., 2020, Henry-Labordere, 2019, Xu et al., 2020, Ni et al., 2020, Wiese et al., 2020]
- Stochastic local volatility models without Dupire's vol [Cuchiero et al., 2020]

Generative modelling

Model Selection

Until recently, models in finance and economics were mostly conceived in a three step fashion:

- gathering statistical properties of the underlying time-series or the so called stylized facts

Model Selection

Until recently, models in finance and economics were mostly conceived in a three step fashion:

- gathering statistical properties of the underlying time-series or the so called stylized facts
- handcrafting a parsimonious model, which would best capture the desired market characteristics without adding any needless complexity and

Model Selection

Until recently, models in finance and economics were mostly conceived in a three step fashion:

- gathering statistical properties of the underlying time-series or the so called stylized facts
- handcrafting a parsimonious model, which would best capture the desired market characteristics without adding any needless complexity and
- calibration and validation of the handcrafted model.

Model Selection

Until recently, models in finance and economics were mostly conceived in a three step fashion:

- gathering statistical properties of the underlying time-series or the so called stylized facts
- handcrafting a parsimonious model, which would best capture the desired market characteristics without adding any needless complexity and
- calibration and validation of the handcrafted model.

... model complexity was undesirable

Classical Risk Models:

Pros:

- Interpretable parameters
- Relatively easy to calibrate with relatively small amount of data
- Several decades of underpinning research

Classical Risk Models:

Pros:

- Interpretable parameters
- Relatively easy to calibrate with relatively small amount of data
- Several decades of underpinning research

Cons:

- Lack of systematic framework for model selection
- Knighting uncertainty (unknown unknowns)
- limited expressivity

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data
- A generative model is a transport map T from μ to ν i.e T is a map that "pushes μ onto ν". We write $T_{\#}\mu = \nu$.

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data
- A generative model is a transport map T from μ to ν i.e T is a map that "pushes μ onto ν". We write $T_{\#}\mu = \nu$.
- Parametrise transport map $T(\theta)$, $\theta \in \mathbb{R}^p$, e.g some network architecture or Heston model

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data
- A generative model is a transport map T from μ to ν i.e T is a map that “pushes μ onto ν”. We write $T_{\#}\mu = \nu$.
- Parametrise transport map $T(\theta)$, $\theta \in \mathbb{R}^p$, e.g some network architecture or Heston model
- Seek $\theta^\star$ s.t $T(\theta^\star)_{\#}\mu \approx \nu$.

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data
- A generative model is a transport map T from μ to ν i.e T is a map that "pushes μ onto ν". We write $T_{\#}\mu = \nu$.
- Parametrise transport map $T(\theta)$, $\theta \in \mathbb{R}^p$, e.g some network architecture or Heston model
- Seek $\theta^\star$ s.t $T(\theta^\star)_{\#}\mu \approx \nu$.
- Need to make the choice of the metric

$$D(T(\theta)_{\#}\mu, \nu) := \sup_{f \in \mathcal{K}} \| \int f(x)(T(\theta)_{\#}\mu)(dx) - \int f(x)\nu(dx) \|$$

- $\mathcal{K}$ could be set of options we want to calibrate to, could be neural network

Generative modelling

- Generative models such as GANs or VAEs demonstrated a great success in seemingly high dimensional setups.
- Input: Source distribution μ and target distribution ν i.e input-output data
- A generative model is a transport map T from μ to ν i.e T is a map that "pushes μ onto ν". We write $T_{\#}\mu = \nu$.
- Parametrise transport map $T(\theta)$, $\theta \in \mathbb{R}^p$, e.g some network architecture or Heston model
- Seek $\theta^\star$ s.t $T(\theta^\star)_{\#}\mu \approx \nu$.
- Need to make the choice of the metric

$$D(T(\theta)_{\#}\mu, \nu) := \sup_{f \in \mathcal{K}} \| \int f(x)(T(\theta)_{\#}\mu)(dx) - \int f(x)\nu(dx) \|$$

- $\mathcal{K}$ could be set of options we want to calibrate to, could be neural network
- The modelling choices are
  1. metric D
  2. parametrisation of T
  3. algorithm used for training!!!

Generative modelling in finance

Pros:

- Expressive and work in high dimensions
- By design data driven, adaptable to change in environment
- Provide new perspective on classical problems in finance

Generative modelling in finance

Pros:

- Expressive and work in high dimensions
- By design data driven, adaptable to change in environment
- Provide new perspective on classical problems in finance

Cons:

- Parameters are not interpretable - black box approach
- Training algorithms are data hungry
- Models might be hard to work with e.g how to go fro $\mathbb{Q}$ to $\mathbb{P}$?
- A field largely empirical, lack of standardised benchmarks, lack of theoretical guarantees

Robust pricing and hedging via neural SDEs

Model Calibration

Classical Calibration:

- Pick a parametric model $(S_t(\theta))_{t\in[0,T]}$ (e.g an Itô process) with parameters $\theta \in \mathbb{R}^p$

Model Calibration

Classical Calibration:

- Pick a parametric model $(S_t(\theta))_{t\in[0,T]}$ (e.g an Itô process) with parameters $\theta \in \mathbb{R}^p$
- Parametric model induces martingale measure $\mathbb{Q}(\theta)$

Model Calibration

Classical Calibration:

- Pick a parametric model $(S_t(\theta))_{t\in[0,T]}$ (e.g an Itô process) with parameters $\theta \in \mathbb{R}^p$
- Parametric model induces martingale measure $\mathbb{Q}(\theta)$
- Input Data: prices of traded derivatives $\mathfrak{p}(\Phi_i)_{i=0}^M$ with corresponding payoffs $(\Phi_i)_{i=0}^M$

Model Calibration

Classical Calibration:

- Pick a parametric model $(S_t(\theta))_{t\in[0,T]}$ (e.g an Itô process) with parameters $\theta \in \mathbb{R}^p$
- Parametric model induces martingale measure $\mathbb{Q}(\theta)$
- Input Data: prices of traded derivatives $\mathfrak{p}(\Phi_i)_{i=0}^M$ with corresponding payoffs $(\Phi_i)_{i=0}^M$
- Output : θ^* such that $\mathfrak{p}(\Phi_i) \approx \mathbb{E}_{\mathbb{Q}(\Theta^*)}[\Phi_i]$

Robust Price bounds

- There are infinitely many models that are consistent with the market

Robust Price bounds

- There are infinitely many models that are consistent with the market
- $\mathcal{M}$ - set of all martingale measures that are calibrated to data

Robust Price bounds

- There are infinitely many models that are consistent with the market
- $\mathcal{M}$ - set of all martingale measures that are calibrated to data
- Compute conservative bounds for the price

$$\sup_{\mathbb{Q}\in\mathcal{M}} E[\Psi] \quad \text{and} \quad \inf_{\mathbb{Q}\in\mathcal{M}} E[\Psi]$$

- Use duality theory to deduce (semi-static) hedging strategy

Robust Price bounds

- There are infinitely many models that are consistent with the market
- $\mathcal{M}$ - set of all martingale measures that are calibrated to data
- Compute conservative bounds for the price

$$\sup_{\mathbb{Q}\in\mathcal{M}} E[\Psi] \quad \text{and} \quad \inf_{\mathbb{Q}\in\mathcal{M}} E[\Psi]$$

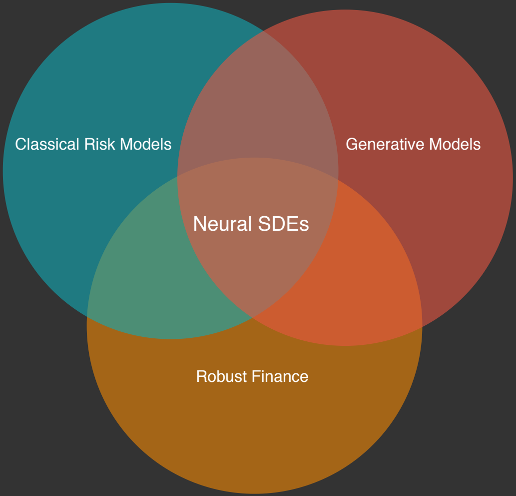
- Use duality theory to deduce (semi-static) hedging strategy
- The obtain bounds typically to wide to be of practical value

Robust Price bounds

- There are infinitely many models that are consistent with the market
- $\mathcal{M}$ - set of all martingale measures that are calibrated to data
- Compute conservative bounds for the price

$$\sup_{\mathbb{Q}\in\mathcal{M}} E[\Psi] \quad \text{and} \quad \inf_{\mathbb{Q}\in\mathcal{M}} E[\Psi]$$

- Use duality theory to deduce (semi-static) hedging strategy
- The obtain bounds typically to wide to be of practical value
- Challenges:
  a) Incorporate prior information to restrict a search space $\mathcal{M}$
  b) Design efficient algorithms for computing price bounds and corresponding hedges


Classical Risk Models
Generative Models
Neural SDEs
Robust Finance

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$\begin{aligned} dS_t^\theta &= rS_t^\theta \, dt + \sigma^S(t, X_t^\theta, \theta)\, dW_t \,, \\ dV_t^\theta &= b^V(t, X_t^\theta, \theta)\, dt + \sigma^V(t, X_t^\theta, \theta)\, dW_t \,, \\ X_t^\theta &= (S_t^\theta, V_t^\theta)\,, \end{aligned}$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$\begin{aligned} dS_t^\theta &= rS_t^\theta\,dt + \sigma^S(t, X_t^\theta, \theta)\,dW_t\,, \\ dV_t^\theta &= b^V(t, X_t^\theta, \theta)\,dt + \sigma^V(t, X_t^\theta, \theta)\,dW_t\,, \\ X_t^\theta &= (S_t^\theta, V_t^\theta)\,, \end{aligned}$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)
- The model induces a martingale probability measure $\mathbb{Q}(\theta)$

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$\begin{aligned} dS_t^\theta &= rS_t^\theta \, dt + \sigma^S(t, X_t^\theta, \theta)\, dW_t \,, \\ dV_t^\theta &= b^V(t, X_t^\theta, \theta)\, dt + \sigma^V(t, X_t^\theta, \theta)\, dW_t \,, \\ X_t^\theta &= (S_t^\theta, V_t^\theta)\,, \end{aligned}$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)
- The model induces a martingale probability measure $\mathbb{Q}(\theta)$
- Solution map is an instance of causal transport

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$\begin{aligned} dS_t^\theta &= rS_t^\theta\, dt + \sigma^S(t, X_t^\theta, \theta)\, dW_t\,, \\ dV_t^\theta &= b^V(t, X_t^\theta, \theta)\, dt + \sigma^V(t, X_t^\theta, \theta)\, dW_t\,, \\ X_t^\theta &= (S_t^\theta, V_t^\theta)\,, \end{aligned}$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)
- The model induces a martingale probability measure $\mathbb{Q}(\theta)$
- Solution map is an instance of causal transport
- See [Cuchiero et al., 2020] for neural SDEs with a prior on vol process.

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$\begin{aligned} dS_t^\theta &= rS_t^\theta\, dt + \sigma^S(t, X_t^\theta, \theta)\, dW_t\,, \\ dV_t^\theta &= b^V(t, X_t^\theta, \theta)\, dt + \sigma^V(t, X_t^\theta, \theta)\, dW_t\,, \\ X_t^\theta &= (S_t^\theta, V_t^\theta)\,, \end{aligned}$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)
- The model induces a martingale probability measure $\mathbb{Q}(\theta)$
- Solution map is an instance of causal transport
- See [Cuchiero et al., 2020] for neural SDEs with a prior on vol process.
- See [Arribas et al., 2020] for Sig-SDEs (neural SDE in a signature feature space)

Neural SDEs

- We build an Itô process $(X_t^\theta)_{t\in[0,T]}$, with parameters $\theta \in \mathbb{R}^p$

$$
\begin{aligned}
dS_t^\theta &= rS_t^\theta \, dt + \sigma^S(t, X_t^\theta, \theta) \, dW_t \,, \\
dV_t^\theta &= b^V(t, X_t^\theta, \theta) \, dt + \sigma^V(t, X_t^\theta, \theta) \, dW_t \,, \\
X_t^\theta &= (S_t^\theta, V_t^\theta) \,,
\end{aligned}
$$

  where σ^S, b^V, σ^V are given by neural networks (can be path-depedend)
- The model induces a martingale probability measure $\mathbb{Q}(\theta)$
- Solution map is an instance of causal transport
- See [Cuchiero et al., 2020] for neural SDEs with a prior on vol process.
- See [Arribas et al., 2020] for Sig-SDEs (neural SDE in a signature feature space)
- Neural SDEs are easy to work with e.g consistent change from $\mathbb{Q}$ to $\mathbb{P}$.

Neural SDEs

i) **Calibration to market prices** Find model parameters θ^* such that model prices match market prices:

$$\theta^* \in \arg\min_{\theta \in \Theta} \sum_{i=1}^{M} \ell(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)) \,.$$

Neural SDEs

i) **Calibration to market prices** Find model parameters θ^* such that model prices match market prices:

$$\theta^* \in \arg\min_{\theta\in\Theta} \sum_{i=1}^{M} \ell(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i))\,.$$

ii) **Robust pricing** Find model parameters $\theta^{l,*}$ and $\theta^{u,*}$ which provide robust arbitrage-free price bounds for an illiquid derivative, subject to available market data:

$$\theta^{l,*} \in \arg\min_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi]\,, \quad \text{subject to} \quad \sum_{i=1}^{M} \ell(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)) = 0\,,$$

$$\theta^{u,*} \in \arg\max_{\theta\in\Theta} \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi], \quad \text{subject to} \quad \sum_{i=1}^{M} \ell(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi_i], \mathfrak{p}(\Phi_i)) = 0\,.$$

where $\ell : \mathbb{R}\times\mathbb{R} \to [0,\infty)$ is a convex loss function such that $\min_{x\in\mathbb{R}, y\in\mathbb{R}} \ell(x,y) = 0$.

Stochastic Optimisation

Let $M = 1$ and the loss function

$$h(\theta) = \ell\Big(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi], \mathfrak{p}(\Phi)\Big)\,.$$

Then in the gradient step update we have

$$\partial_\theta h(\theta) = \partial_x \ell\left(\mathbb{E}^{\mathbb{Q}}[\Phi(X^\theta)], \mathfrak{p}(\Phi)\right) \mathbb{E}^{\mathbb{Q}}[\partial_\theta \Phi(X^\theta)]\,,$$

Since ℓ is typically not an identity function, a mini-batch estimator of $\partial_\theta h(\theta)$, obtained by replacing $\mathbb{Q}$ with $\mathbb{Q}^N$ given by

$$\partial_\theta h^N(\theta) := \partial_x \ell\left(\mathbb{E}^{\mathbb{Q}^N}[\Phi(X^\theta)], \mathfrak{p}(\Phi)\right) \mathbb{E}^{\mathbb{Q}^N}[\partial_\theta \Phi(X^\theta)]\,,$$

is a biased estimator of $\partial_\theta h$.

Stochastic Optimisation

Let $M = 1$ and the loss function

$$h(\theta) = \ell\Big(\mathbb{E}^{\mathbb{Q}(\theta)}[\Phi], \mathfrak{p}(\Phi)\Big).$$

Then in the gradient step update we have

$$\partial_\theta h(\theta) = \partial_x \ell\left(\mathbb{E}^{\mathbb{Q}}[\Phi(X^\theta)], \mathfrak{p}(\Phi)\right) \mathbb{E}^{\mathbb{Q}}[\partial_\theta \Phi(X^\theta)],$$

Since ℓ is typically not an identity function, a mini-batch estimator of $\partial_\theta h(\theta)$, obtained by replacing $\mathbb{Q}$ with $\mathbb{Q}^N$ given by

$$\partial_\theta h^N(\theta) := \partial_x \ell\left(\mathbb{E}^{\mathbb{Q}^N}[\Phi(X^\theta)], \mathfrak{p}(\Phi)\right) \mathbb{E}^{\mathbb{Q}^N}[\partial_\theta \Phi(X^\theta)],$$

is a biased estimator of $\partial_\theta h$.

Lemma 1

For $\ell(x, y) = |x - y|^2$, we have

$$\left|\mathbb{E}^{\mathbb{Q}}\left[\partial_\theta h^N(\theta)\right] - \partial_\theta h(\theta)\right| \leq \frac{2}{N}\left(\mathbb{V}ar^{\mathbb{Q}}[\Phi(X^\theta)]\right)^{1/2}\left(\mathbb{V}ar^{\mathbb{Q}}[\partial_\theta \Phi(X^\theta)]\right)^{1/2}.$$

Learning PDEs

Let

$$X_t^\beta = \sigma(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}, \beta)dW_t,$$

Learning PDEs

Let

$$X_t^\beta = \sigma(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}, \beta) dW_t,$$

$$F_t^\beta := F^\beta(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}) = \mathbb{E}\left[\Phi((X_s^\beta)_{s\in[0,T]}) \middle| (X_{s\wedge t}^\beta)_{s\in[0,T]}\right]$$

Learning PDEs

Let

$$X_t^\beta = \sigma(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}, \beta)dW_t,$$

$$F_t^\beta := F^\beta(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}) = \mathbb{E}\left[\Phi((X_s^\beta)_{s\in[0,T]})|(X_{s\wedge t}^\beta)_{s\in[0,T]}\right]$$

Martingale representation theorem via functional Itô calculus

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

Learning PDEs

Let

$$X_t^\beta = \sigma(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}, \beta)dW_t,$$

$$F_t^\beta := F^\beta(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}) = \mathbb{E}\left[\Phi((X_s^\beta)_{s\in[0,T]}) | (X_{s\wedge t}^\beta)_{s\in[0,T]}\right]$$

Martingale representation theorem via functional Itô calculus

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

$$\mathbb{V}\left[\Phi\big((X_s^\beta)_{s\in[0,T]}\big) - \int_t^T \nabla_\omega F_t^\beta\big((X_{r\wedge s}^\beta)_{r\in[0,T]}\big)\, dX_s \mid (X_{s\wedge t}^\beta)_{s\in[0,T]}\right] = 0$$

Learning PDEs

Let

$$X_t^\beta = \sigma(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}, \beta) dW_t,$$

$$F_t^\beta := F^\beta(t, (X_{s\wedge t}^\beta)_{s\in[0,T]}) = \mathbb{E}\left[\Phi((X_s^\beta)_{s\in[0,T]}) | (X_{s\wedge t}^\beta)_{s\in[0,T]}\right]$$

Martingale representation theorem via functional Itô calculus

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

$$\mathbb{V}\left[\Phi\big((X_s^\beta)_{s\in[0,T]}\big) - \int_t^T \nabla_\omega F_t^\beta\big((X_{r\wedge s}^\beta)_{r\in[0,T]}\big)\, dX_s \mid (X_{s\wedge t}^\beta)_{s\in[0,T]}\right] = 0$$

- Can learn (parametric) path dependent PDEs
- We have unbiased approximation to the PDE by hybrid Monte Carlo/deep learning, see [Vidales et al., 2018]

Neural SDEs - Algorithm

Input: $\pi = \{t_0, t_1, \ldots, t_{N_{\text{steps}}}\}$ time grid for numerical scheme.
Input: $(\Phi_i)_{i=1}^{N_{\text{prices}}}$ option payoffs.
Input: Market option prices $\mathfrak{p}(\Phi_j)$, $j = 1, \ldots, N_{\text{prices}}$.
for epoch : 1 : N_{epochs} **do**

Generate N_{trn} paths $(x_{t_n}^{\pi,\theta,i})_{n=0}^{N_{\text{steps}}} := (s_{t_n}^{\pi,\theta,i}, v_{t_n}^{\pi,\theta,i})_{n=0}^{N_{\text{steps}}}$, $i = 1, \ldots, N_{\text{trn}}$ using Euler scheme.

During one epoch: Freeze ξ, use Adam to update θ, where

$$\theta = \widehat{\operatorname{argmin}_{\theta}} \sum_{j=1}^{N_{\text{prices}}} \left(\mathbb{E}^{N_{\text{trn}}} \left[\Phi_j \left(X^{\pi,\theta} \right) - \sum_{k=0}^{N_{\text{steps}}-1} \bar{\mathfrak{h}}(t_k, \tilde{X}_{t_k}^{\pi,}, \xi_j) \Delta \tilde{\bar{S}}_{t_k}^{\pi,} \right] - \mathfrak{p}(\Phi_j) \right)^2$$

During one epoch: Freeze θ, use Adam to update ξ, by optimising the sample variance

$$\xi = \widehat{\operatorname{argmin}_{\xi}} \sum_{j=1}^{N_{\text{prices}}} \mathbb{V}\mathrm{ar}^{N_{\text{trn}}} \left[\Phi_j \left(X^{\pi,\theta} \right) - \sum_{k=0}^{N_{\text{steps}}-1} \bar{\mathfrak{h}}(t_k, X_{t_k}^{\pi,\theta}, \xi_j) \Delta \tilde{\bar{S}}_{t_k}^{\pi,\theta} \right]$$

end for
return θ, ξ_j for all prices $(\Phi_i)_{i=1}^{N_{\text{prices}}}$.

Results

We calibrate (local) Stochastic Volatility model

$$
\begin{aligned}
dS_t &= rS_t dt + \sigma^S(t, S_t, V_t, \nu)S_t\, dB_t^S, \quad S_0 = 1,\\
dV_t &= b^V(V_t, \phi)\, dt + \sigma^V(V_t, \varphi)\, dB_t^V, \quad V_0 = v_0,\\
d\langle B^S, B^V\rangle_t &= \rho dt
\end{aligned}
$$

to European option prices

$$
\mathfrak{p}(\Phi) := \mathbb{E}^{\mathbb{Q}(\theta)}[\Phi] = e^{-rT}\mathbb{E}^{\mathbb{Q}(\theta)}\left[(S_T - K)_+ \mid S_0 = 1\right]
$$

for maturities of $2, 4, \ldots, 12$ months and typically 21 uniformly spaced strikes between in $[0.8, 1.2]$.

Results

We calibrate (local) Stochastic Volatility model

$$
\begin{aligned}
dS_t &= rS_t dt + \sigma^S(t, S_t, V_t, \nu) S_t \, dB_t^S, \quad S_0 = 1, \\
dV_t &= b^V(V_t, \phi)\, dt + \sigma^V(V_t, \varphi)\, dB_t^V, \quad V_0 = v_0, \\
d\langle B^S, B^V \rangle_t &= \rho dt
\end{aligned}
$$

to European option prices

$$
\mathfrak{p}(\Phi) := \mathbb{E}^{\mathbb{Q}(\theta)}[\Phi] = e^{-rT} \mathbb{E}^{\mathbb{Q}(\theta)} \left[(S_T - K)_+ \,|\, S_0 = 1 \right]
$$

for maturities of $2, 4, \ldots, 12$ months and typically 21 uniformly spaced strikes between in $[0.8, 1.2]$.

As an example of an illiquid derivative for which we wish to find robust bounds we take the lookback option

$$
\mathfrak{p}(\Psi) := \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi] = e^{-rT} \mathbb{E}^{\mathbb{Q}(\theta)} \left[\max_{t \in [0,T]} S_t - S_T \middle| X_0 = 1 \right].
$$

Results

We calibrate (local) Stochastic Volatility model

$$
\begin{aligned}
dS_t &= rS_t dt + \sigma^S(t, S_t, V_t, \nu) S_t \, dB_t^S, \quad S_0 = 1, \\
dV_t &= b^V(V_t, \phi) \, dt + \sigma^V(V_t, \varphi) \, dB_t^V, \quad V_0 = v_0, \\
d\langle B^S, B^V \rangle_t &= \rho dt
\end{aligned}
$$

to European option prices

$$
\mathfrak{p}(\Phi) := \mathbb{E}^{\mathbb{Q}(\theta)}[\Phi] = e^{-rT} \mathbb{E}^{\mathbb{Q}(\theta)} \left[(S_T - K)_+ \,|\, S_0 = 1 \right]
$$

for maturities of $2, 4, \ldots, 12$ months and typically 21 uniformly spaced strikes between in $[0.8, 1.2]$.

As an example of an illiquid derivative for which we wish to find robust bounds we take the lookback option

$$
\mathfrak{p}(\Psi) := \mathbb{E}^{\mathbb{Q}(\theta)}[\Psi] = e^{-rT} \mathbb{E}^{\mathbb{Q}(\theta)} \left[\max_{t \in [0,T]} S_t - S_T \middle| X_0 = 1 \right].
$$

We generate synthetic data using Heston model.

Calibration to market prices

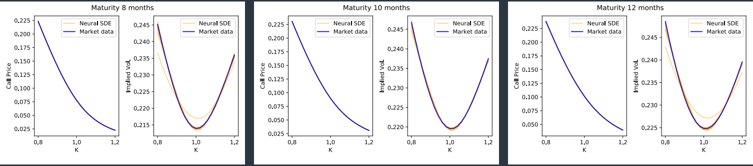


Figure: Vanilla option prices and implied volatility curves of the 10 calibrated Neural SDEs vs. the market data for different maturities.

Robust pricing

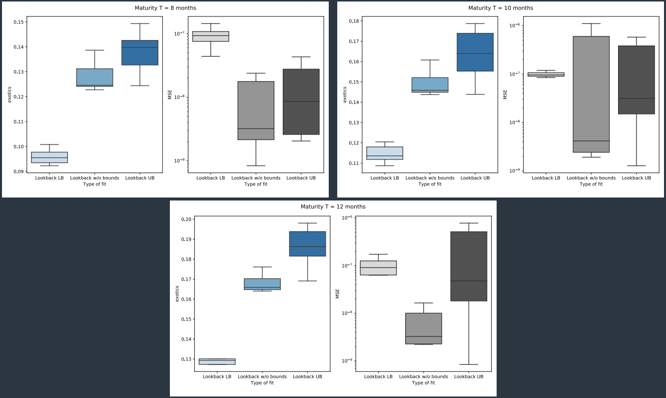


Figure: Exotic option price are in blue; Calibration error i in grey. The three box-plots in each group arise respectively from aiming for a lower bound, ad hoc and upper bound price of illiquid derivative. Each box plot comes from 10 different runs of Neural SDE calibration.

Control Variate effect on training

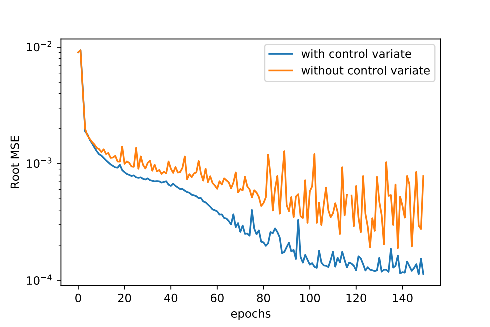


Figure: Root Mean Squared Error of calibration to Vanilla option prices with and without hedging strategy parametrisation

Joint SPX and VIX calibration with neural SDEs

Consider the Neural SDE

$$\begin{aligned} dS_t^\theta &= S_t^\theta \sigma(t, V_t^\theta; \theta)\, dW_t\,, \\ dV_t^\theta &= a(t, V_t^\theta; \theta)\, dt + b(t, V_t^\theta; \theta)\, dB_t\,, \\ \rho &= \langle dW, dB \rangle_t\ . \end{aligned}$$

It can be shown that the VIX dynamics at time $t \in [0, T]$ can be expressed as

$$\mathrm{VIX}_t^2 := \frac{1}{\Delta\tau} \mathbb{E}\left[\int_t^{t+\Delta\tau} \sigma_s^2 \mathrm{d}s \middle| \mathcal{F}_t \right] = -\frac{2}{\Delta\tau} \mathbb{E}\left[\log\left(\frac{S_{t+\Delta\tau}}{S_t} \right) \middle| \mathcal{F}_t \right], \quad \Delta\tau = \frac{30}{365}$$

The VIX future with maturity maturity T is then given by

$$F_{t,T}^{\mathrm{VIX}} := \mathbb{E}\left[\mathrm{VIX}_T | \mathcal{F}_t \right]$$

VIX options are defined as

$$C_t^{\mathrm{VIX}}(T, K) := \mathbb{E}\left[(\mathrm{VIX}_T - K)^+ \middle| \mathcal{F}_t \right], \ P_t^{\mathrm{VIX}}(T, K) := \mathbb{E}\left[(K - \mathrm{VIX}_T)^+ \middle| \mathcal{F}_t \right].$$

joint work with: Antoine Jacquier, Marc Sabate Vidales, David Siska, Zan Zuric.

Calibration to market data

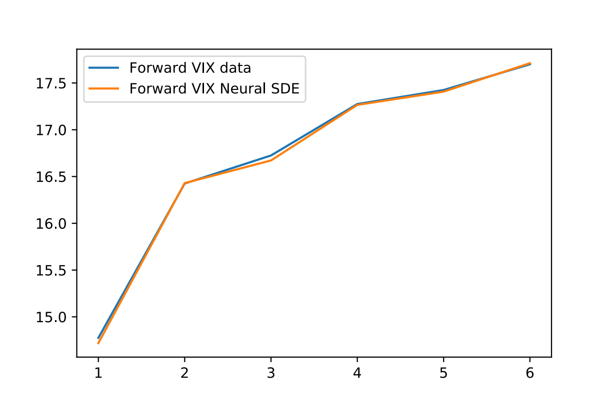


Figure: Calibration to market data (data source: OptionMetrics) containing SPX options, VIX options an VIX future for $T = 1, \ldots, 6$ months

Calibration to market data

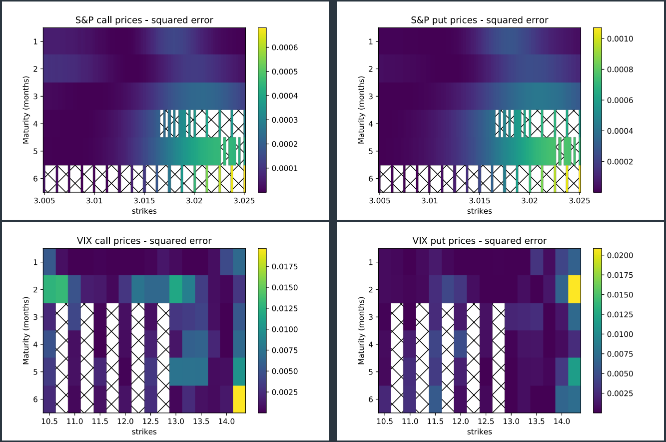


Figure: Calibrated neural SDE errors on SPX options and VIX options. Hatches correspond to combinations of Maturity/Strike for which there was not market data available

Extensions

- Neural SDE model in real-world measure $\mathbb{P}(\theta)$

Extensions

- Neural SDE model in real-world measure $\mathbb{P}(\theta)$
- Let $\zeta : [0, T] \times \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^n$ be another parametric function
- Let

$$b^{S,\mathbb{P}}(t, X_t^\theta, \theta) := rS_t^\theta + \sigma^S(t, X_t^\theta, \theta)\zeta(t, X_t^\theta, \theta)\,,$$
$$b^{V,\mathbb{P}}(t, X_t^\theta, \theta) := b^V(t, X_t^\theta, \theta) + \sigma^V(t, X_t^\theta, \theta)\zeta(t, X_t^\theta, \theta)\,.$$

- We now define a real-world measure $\mathbb{P}(\theta)$ via the Radon–Nikodym derivative

$$\frac{d\mathbb{P}(\theta)}{d\mathbb{Q}(\theta)} := \exp\left(\int_0^T \zeta(t, X_t^\theta, \theta)\, dW_t + \frac{1}{2}\int_0^T |\zeta(t, X_t^\theta, \theta)|^2\, dt\right)\,.$$

Extensions

- Neural SDE model in real-world measure $\mathbb{P}(\theta)$
- Let $\zeta : [0, T] \times \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^n$ be another parametric function
- Let

$$
\begin{aligned}
b^{S,\mathbb{P}}(t, X_t^\theta, \theta) &:= rS_t^\theta + \sigma^S(t, X_t^\theta, \theta)\zeta(t, X_t^\theta, \theta)\,, \\
b^{V,\mathbb{P}}(t, X_t^\theta, \theta) &:= b^V(t, X_t^\theta, \theta) + \sigma^V(t, X_t^\theta, \theta)\zeta(t, X_t^\theta, \theta)\,.
\end{aligned}
$$

- We now define a real-world measure $\mathbb{P}(\theta)$ via the Radon–Nikodym derivative

$$
\frac{d\mathbb{P}(\theta)}{d\mathbb{Q}(\theta)} := \exp\left(\int_0^T \zeta(t, X_t^\theta, \theta)\, dW_t + \frac{1}{2}\int_0^T |\zeta(t, X_t^\theta, \theta)|^2\, dt\right)\,.
$$

- Under appropriate assumption on ζ (e.g. bounded) the measure $\mathbb{P}(\theta)$ is a probability measure and by using Girsanov theorem we can find Brownian motion $(W_t^{\mathbb{P}(\theta)})_{t\in[0,T]}$ such that

$$
\begin{aligned}
dS_t^\theta &= b^{S,\mathbb{P}}(t, X_t^\theta, \theta)\, dt + \sigma^S(t, X_t^\theta, \theta)\, dW_t^{\mathbb{P}(\theta)}\,, \\
dV_t^\theta &= b^{V,\mathbb{P}}(t, X_t^\theta, \theta)\, dt + \sigma^V(t, X_t^\theta, \theta)\, dW_t^{\mathbb{P}(\theta)}\,.
\end{aligned}
$$

Extensions

- We can incorporate additional market information e.g bound on realised variance by adding additional constrain during training

Extensions

- We can incorporate additional market information e.g bound on realised variance by adding additional constrain during training
- We can use neural SDEs to adversarially train hedging strategies using ideas from distributionally robust optimisation

Extensions

- We can incorporate additional market information e.g bound on realised variance by adding additional constrain during training
- We can use neural SDEs to adversarially train hedging strategies using ideas from distributionally robust optimisation
- We can simplify the learned models using ideas from explainable machine learning

Neural SDEs

Pros:

- Expressive yet consistent with classical framework
- By design data driven, adaptable to changes in environment
- Provide consistent models for calibrating under $\mathbb{Q}$ and $\mathbb{P}$
- Provide systematic framework for model selection
- Ability to learn in law data regime (due to good prior)
- (Some) Theoretical guarantees for the generalisation error

Neural SDEs

Pros:

- Expressive yet consistent with classical framework
- By design data driven, adaptable to changes in environment
- Provide consistent models for calibrating under $\mathbb{Q}$ and $\mathbb{P}$
- Provide systematic framework for model selection
- Ability to learn in law data regime (due to good prior)
- (Some) Theoretical guarantees for the generalisation error

Cons:

- Parameters are not interpretable, but the models are.
- Computationally more intense than classical models, but because we train with gradient descent recalibration (typically) is cheap.

Neural ODEs - perspective on recurrent neural network

Example

- Recurrent neural networks can be written

$$X^{l+1} = X^l + \phi(X^l, \theta^l) \quad X^0 = \xi \in \mathbb{R}^d$$

Example

- Recurrent neural networks can be written

$$X^{l+1} = X^l + \phi(X^l, \theta^l) \quad X^0 = \xi \in \mathbb{R}^d$$

- Infinite network (useful when fitting time-series data)

$$dX_t^\xi(\theta) = \phi(X_t^\xi(\theta), \theta_t)\, dt\,, \;\; t \in [0,1]\,, \;\; X_0 = \xi \in \mathbb{R}^d.$$

Example

- Recurrent neural networks can be written

$$X^{l+1} = X^l + \phi(X^l, \theta^l) \quad X^0 = \xi \in \mathbb{R}^d$$

- Infinite network (useful when fitting time-series data)

$$dX_t^\xi(\theta) = \phi(X_t^\xi(\theta), \theta_t)\, dt\,, \;\; t \in [0,1]\,, \;\; X_0 = \xi \in \mathbb{R}^d.$$

- Take input-output data $(\xi, \zeta) \sim \mathcal{M}$. Our objective is to minimize

$$J\Big((\theta)_{t\in[0,T]}\Big) := \int_{\mathbb{R}^d \times \mathbb{R}^d} |\zeta - X_T^\xi(\theta)|^2 \, \mathcal{M}(d\xi, d\zeta)\,.$$

Example

- Recurrent neural networks can be written

$$X^{l+1} = X^l + \phi(X^l, \theta^l) \quad X^0 = \xi \in \mathbb{R}^d$$

- Infinite network (useful when fitting time-series data)

$$dX_t^{\xi}(\theta) = \phi(X_t^{\xi}(\theta), \theta_t)\, dt\,, \quad t \in [0,1]\,, \quad X_0 = \xi \in \mathbb{R}^d.$$

- Take input-output data $(\xi, \zeta) \sim \mathcal{M}$. Our objective is to minimize

$$J\Big((\theta)_{t\in[0,T]}\Big) := \int_{\mathbb{R}^d\times\mathbb{R}^d} |\zeta - X_T^{\xi}(\theta)|^2\, \mathcal{M}(d\xi, d\zeta)\,.$$

- Goal: Find $\tilde{\theta}$ such that

$$\frac{d}{d\varepsilon} J\Big(\theta + \varepsilon(\tilde{\theta} - \theta)\Big)\Big|_{\varepsilon=0} \leq 0\,.$$

Relaxed Stochastic Control and Deep Learning

- Mean-field perspective on neural networks

$$\frac{1}{n}\sum_{i=1}^{n}\beta_{n,i}\varphi(\alpha_{n,i}\cdot z+\rho_{i,n}\cdot\zeta)=\int_{\mathbb{R}^d}\beta\varphi(\alpha\cdot z+\rho\cdot\zeta)\,\nu^n(d\beta,d\alpha,d\rho)\,.$$

Relaxed Stochastic Control and Deep Learning

- Mean-field perspective on neural networks

$$\frac{1}{n}\sum_{i=1}^{n}\beta_{n,i}\varphi(\alpha_{n,i}\cdot z+\rho_{i,n}\cdot\zeta)=\int_{\mathbb{R}^d}\beta\varphi(\alpha\cdot z+\rho\cdot\zeta)\,\nu^n(d\beta,d\alpha,d\rho)\,.$$

- Let $\phi(z,a,\zeta)=\beta\varphi(\alpha\cdot z+\rho\cdot\zeta)$, and consider

$$X_t^{\nu,\xi,\zeta}=\xi+\int_0^t\int\phi(X_r^{\nu,\xi,\zeta},a,\zeta)\,\nu_r(da)\,dr$$

Relaxed Stochastic Control and Deep Learning

- $$J^{\sigma,\mathcal{M}}(\nu) := \int_{\mathbb{R}^d \times \mathcal{S}} \left[\int_0^T \int f_t(X_t^{\nu,\xi,\zeta}, a, \zeta)\, \nu_t(da)\, dt + g(X_T^{\nu,\xi,\zeta}, \zeta) \right] \mathcal{M}(d\xi, d\zeta)$$
$$+ \frac{\sigma^2}{2} \int_0^T \mathrm{Ent}(\nu_t)\, dt\,.$$

- $$\mathrm{Ent}(m) := \begin{cases} \int_{\mathbb{R}^d} m(x) \log\left(\frac{m(x)}{g(x)}\right) dx & \text{if } m \text{ is a.c. w.r.t. Lebesgue measure} \\ \infty & \text{otherwise} \end{cases}$$

  and Gibbs measure g:

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

- See work by Weinan E [Weinan, 2017]; Cruchiero, Larsson, Teichmann,[Cuchiero et al., 2019]; Hu, Kazeykina, Ren [Hu et al., 2019]

Relaxed Stochastic Control and Deep Learning

- The goal is to find, for each $t \in [0, T]$ a vector field flow $(b_{s,t})_{s\geq 0}$ such that the measure flow $(\nu_{s,t})_{s\geq 0}$ given by

$$\partial_s \nu_{s,t} = \mathrm{div}(\nu_{s,t}\, b_{s,t})\,, \;\; s \geq 0\,,\; \nu_{0,t} = \nu_t^0 \in \mathcal{P}_2(\mathbb{R}^p)\,,$$

satisfies that $s \mapsto J^\sigma(\nu_{s,\cdot})$ is decreasing.

Relaxed Stochastic Control and Deep Learning

- The goal is to find, for each $t \in [0, T]$ a vector field flow $(b_{s,t})_{s\geq 0}$ such that the measure flow $(\nu_{s,t})_{s\geq 0}$ given by

$$\partial_s \nu_{s,t} = \mathrm{div}(\nu_{s,t}\, b_{s,t})\,, \;\; s \geq 0\,,\; \nu_{0,t} = \nu_t^0 \in \mathcal{P}_2(\mathbb{R}^p)\,,$$

  satisfies that $s \mapsto J^\sigma(\nu_{s,\cdot})$ is decreasing.
- Relaxed Hamiltonian

$$H_t^\sigma(x,p,m,\zeta) := \int h_t(x,p,a,\zeta)\, m(da) + \frac{\sigma^2}{2}\mathrm{Ent}(m)$$
$$h_t(x,p,a,\zeta) := \phi_t(x,a,\zeta)p + f_t(x,a,\zeta)$$

Relaxed Stochastic Control and Deep Learning

- The goal is to find, for each $t \in [0, T]$ a vector field flow $(b_{s,t})_{s\geq 0}$ such that the measure flow $(\nu_{s,t})_{s\geq 0}$ given by

$$\partial_s \nu_{s,t} = \mathrm{div}(\nu_{s,t}\, b_{s,t})\,, \quad s \geq 0\,,\ \nu_{0,t} = \nu_t^0 \in \mathcal{P}_2(\mathbb{R}^p)\,,$$

  satisfies that $s \mapsto J^\sigma(\nu_{s,\cdot})$ is decreasing.
- Relaxed Hamiltonian

$$H_t^\sigma(x,p,m,\zeta) := \int h_t(x,p,a,\zeta)\, m(da) + \frac{\sigma^2}{2}\mathrm{Ent}(m)$$
$$h_t(x,p,a,\zeta) := \phi_t(x,a,\zeta)p + f_t(x,a,\zeta)$$

- The adjoint process $P_t^{\xi,\zeta}(\nu) = (\nabla_x g)(X_t^{\xi,\zeta}(\nu),\zeta)$,

$$dP^{\xi,\zeta}(\nu)_t = -(\nabla_x H_t)(X^{\xi,\zeta}(\nu)_t, P_t^{\nu,\xi,\zeta}(\nu), \nu_t)\, dt$$

Pontryagin's principle

Theorem 2

If $\nu \in \mathcal{V}_2$ is (locally) optimal then it must solve the following system:

$$
\begin{aligned}
\nu_t &= \operatorname*{argmin}_{\mu \in \mathcal{P}_2(\mathbb{R}^p)} \int_{\mathbb{R}^d \times \mathcal{S}} H_t^\sigma(X_t^{\xi,\zeta}, P_t^{\xi,\zeta}, \mu, \zeta)\, \mathcal{M}(d\xi, d\zeta)\,, \\
dX_t^{\xi,\zeta} &= \Phi(X_t^{\xi,\zeta}, \nu_t, \zeta)\, dt\,, \quad , X_0^{\xi,\zeta} = \xi \in \mathbb{R}^d \\
dP_t^{\xi,\zeta} &= -(\nabla_x H_t)(X_t^{\xi,\zeta}, P_t^{\xi,\zeta}, \nu_t, \zeta)\, dt\,,\ P_T^{\xi,\zeta} = (\nabla_x g)(X_T^{\xi,\zeta}, \zeta)\,.
\end{aligned}
$$

Gradient Flow Dynamics

$$d\theta_{s,t} = -\left(\int_{\mathbb{R}^d\times\mathcal{S}} (\nabla_a h_t)(X^{\xi,\zeta}_{s,t}, P^{\xi,\zeta}_{s,t}, \theta_{s,t}, \zeta)\,\mathcal{M}(d\xi, d\zeta) + \frac{\sigma^2}{2}(\nabla_a U)(\theta_{s,t})\right) ds + \sigma dB_s$$

where for $t \in [0, T]$

$$\begin{cases} \nu_{s,t} = \mathcal{L}(\theta_{s,t}) \\ X^{\xi,\zeta}_{s,t} = \xi + \displaystyle\int_0^t \Phi_r(X^{\xi,\zeta}_{s,r}, \nu_{s,r}, \zeta)\,dr \\ P^{\xi,\zeta}_{s,t} = (\nabla_x g)(X^{\xi,\zeta}_T, \zeta) + \displaystyle\int_t^T (\nabla_x H_r)(X^{\xi,\zeta}_{s,r}, P^{\xi,\zeta}_{s,r}, \nu_{s,r}, \zeta)\,dr\,. \end{cases}$$

Main Result

Theorem 3

Assume that $\sigma > 0$. *Then*

i) *if* $\nu^\star \in \operatorname{argmin}_{\nu \in \mathcal{V}_2} J^\sigma(\nu)$ *then* $\nu^\star$ *is an invariant measure given by*

$$\nu^\star(a) = e^{-\frac{2}{\sigma^2}\mathbf{h}_t(a,\nu^\star,\mathcal{M})} g(a)\,,$$

Main Result

Theorem 3

Assume that $\sigma > 0$. Then

i) *if $\nu^\star \in \operatorname{argmin}_{\nu \in \mathcal{V}_2} J^\sigma(\nu)$ then $\nu^\star$ is an invariant measure given by*

$$\nu^\star(a) = e^{-\frac{2}{\sigma^2}\mathbf{h}_t(a,\nu^\star,\mathcal{M})} g(a)\,,$$

ii) *if $\sigma^2\kappa - 4L > 0$ then $\nu^\star$ is unique and for all $s \geq 0$ any any $\mathcal{L}(\theta_{0,\cdot})$*

$$\mathcal{W}_2^T(\mathcal{L}(\theta_{s,\cdot}),\nu^\star)^2 \leq e^{-\lambda s}\mathcal{W}_2^T(\mathcal{L}(\theta_{0,\cdot}),\nu^\star)^2\,.$$

- $$\mathcal{W}_q^T(\mu,\nu) := \left(\int_0^T \mathcal{W}_q(\mu_t,\nu_t)^q\,dt\right)^{1/q}.$$

- $$\mathbf{h}_t(a,\mu,\mathcal{M}) := \int_{\mathbb{R}^d\times\mathcal{S}} h_t(X_t^{\xi,\zeta}(\mu), P_t^{\xi,\zeta}(\mu), a, \zeta)\mathcal{M}(d\xi,d\zeta)\,,$$

Generalisation Error

- Recall the cost function

$$J^{\sigma,\mathcal{M}}(\nu) := \int_{\mathbb{R}^d\times\mathcal{S}} \left[\int_0^T \int f_t(X_t^{\nu,\xi,\zeta}, a, \zeta)\,\nu_t(da)\,dt + g(X_T^{\nu,\xi,\zeta},\zeta)\right] \mathcal{M}(d\xi, d\zeta)$$
$$+ \frac{\sigma^2}{2}\int_0^T \mathrm{Ent}(\nu_t)\,dt\,.$$

Generalisation Error

- Recall the cost function

$$J^{\sigma,\mathcal{M}}(\nu) := \int_{\mathbb{R}^d \times \mathcal{S}} \left[\int_0^T \int f_t(X_t^{\nu,\xi,\zeta}, a, \zeta)\, \nu_t(da)\, dt + g(X_T^{\nu,\xi,\zeta}, \zeta) \right] \mathcal{M}(d\xi, d\zeta) + \frac{\sigma^2}{2} \int_0^T \mathrm{Ent}(\nu_t)\, dt\,.$$

- In practice, one does not have access to population distribution $\mathcal{M}$, but works with finite sample $\mathcal{M}^{N_1} := \frac{1}{N_1} \sum_{j_1}^{N_1} \delta_{(\xi^{j_1}, \zeta^{j_1})}$

Generalisation Error

- Recall the cost function

$$J^{\sigma,\mathcal{M}}(\nu) := \int_{\mathbb{R}^d \times \mathcal{S}} \left[\int_0^T \int f_t(X_t^{\nu,\xi,\zeta}, a, \zeta)\, \nu_t(da)\, dt + g(X_T^{\nu,\xi,\zeta}, \zeta) \right] \mathcal{M}(d\xi, d\zeta)$$
$$+ \frac{\sigma^2}{2} \int_0^T \mathrm{Ent}(\nu_t)\, dt\,.$$

- In practice, one does not have access to population distribution $\mathcal{M}$, but works with finite sample $\mathcal{M}^{N_1} := \frac{1}{N_1} \sum_{j_1}^{N_1} \delta_{(\xi^{j_1}, \zeta^{j_1})}$
- Practioner use $J^{\mathcal{M}^{N_1}}(\nu)$ and NOT $J^{\sigma,\mathcal{M}_{N_1}}(\nu)$ to set stopping criteria for learning

Generalisation Error

- Recall the cost function

$$J^{\sigma,\mathcal{M}}(\nu) := \int_{\mathbb{R}^d \times \mathcal{S}} \left[\int_0^T \int f_t(X_t^{\nu,\xi,\zeta}, a, \zeta)\, \nu_t(da)\, dt + g(X_T^{\nu,\xi,\zeta}, \zeta) \right] \mathcal{M}(d\xi, d\zeta)$$
$$+ \frac{\sigma^2}{2} \int_0^T \mathrm{Ent}(\nu_t)\, dt\,.$$

- In practice, one does not have access to population distribution $\mathcal{M}$, but works with finite sample $\mathcal{M}^{N_1} := \frac{1}{N_1} \sum_{j_1}^{N_1} \delta_{(\xi^{j_1}, \zeta^{j_1})}$
- Practioner use $J^{\mathcal{M}^{N_1}}(\nu)$ and NOT $J^{\sigma,\mathcal{M}_{N_1}}(\nu)$ to set stopping criteria for learning
- The entropy term can be viewed as implicit regularisation

Generalisation Error

Theorem 4

Let $\sigma^2\kappa >> 0$. There is $c > 0$ independent of λ, S, N_1, N_2, d, p s.t

$$\mathbb{E}\left[\left|J^{\mathcal{M}}(\nu^{\star,\sigma}) - J^{\mathcal{M}}(\nu_{S,\cdot}^{\sigma,N_1,N_2,\Delta s})\right|^2\right] \leq c\left(e^{-\lambda S} + \frac{1}{N_1} + \frac{1}{N_2} + h\right),$$

The generalisation error is given by

$$J^{\mathcal{M}}(\nu_{S,\cdot}^{\sigma,N_1,N_2,\Delta s})$$
$$= J^{\mathcal{M}}(\nu_{S,\cdot}^{\sigma,N_1,N_2,\Delta s}) - J^{\mathcal{M}}(\nu^{\star,\sigma}) - \frac{\sigma^2}{2}\int_0^T Ent(\nu^{\star,\sigma}) + \min_{\mu\in\mathcal{V}_2} J^{\sigma,\mathcal{M}}(\mu).$$

- N_1 - size of the training data
- N_2 - proxy to the the number of parameters
- γ - learning rate
- S/γ - proxy for training time
- By discretising ODEs we can get estimateson the number of layers

Assumption 1

Fix $\epsilon > 0$ and $N_1 > 0$. Assume that $\forall \mathcal{M}^{N_1}$ $J^{M_1}(\nu^{\star,\sigma,N_1}) \leq \epsilon$.

Theorem 5

There is $c > 0$ independent of λ, S, N_1, N_2, d, p s.t

$$\mathbb{E}\left[\left|J^{\mathcal{M}}(\nu_{S,\cdot}^{\sigma,N_1,N_2,\Delta s})\right|^2\right] \leq \epsilon^2 + c\left(e^{-\lambda S} + \frac{1}{N_1} + \frac{1}{N_2} + h\right).$$

Outlook

We have full analysis of convergence of regularised gradient descent algorithm for deep networks modelled by ODEs.

Outlook

We have full analysis of convergence of regularised gradient descent algorithm for deep networks modelled by ODEs.

Key messages:

- Training of neural nets should be viewed as sampling rather then optimisation problem
- Wasserstein Gradient flow provides framework to study convergence of training algorithms
- Probabilistic numerical analysis provides quantitative bounds that do not suffer from curse of dimensionality

References I

[Arribas et al., 2020] Arribas, I. P., Salvi, C., and Szpruch, L. (2020). Sig-sdes model for quantitative finance. *arXiv preprint arXiv:2006.00218.*

[Belkin et al., 2018] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2018). Reconciling modern machine learning and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118.*

[Buehler et al., 2019] Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291.

[Buehler et al., 2020] Buehler, H., Horvath, B., Lyons, T., Perez Arribas, I., and Wood, B. (2020). Generating financial markets with signatures. *Available at SSRN.*

[Cuchiero et al., 2020] Cuchiero, C., Khosrawi, W., and Teichmann, J. (2020). A generative adversarial network approach to calibration of local stochastic volatility models. *arXiv preprint arXiv:2005.02505.*

[Cuchiero et al., 2019] Cuchiero, C., Larsson, M., and Teichmann, J. (2019). Deep neural networks, generic universal interpolation, and controlled odes. *arXiv preprint arXiv:1908.07838.*

[Gierjatowicz et al., 2020] Gierjatowicz, P., Sabate-Vidales, M., Siska, D., Szpruch, L., and Zuric, Z. (2020). Robust pricing and hedging via neural sdes. *Available at SSRN 3646241.*

[Heiss et al., 2019] Heiss, J., Teichmann, J., and Wutte, H. (2019). How implicit regularization of neural networks affects the learned function–part i. *arXiv preprint arXiv:1911.02903.*

[Henry-Labordere, 2019] Henry-Labordere, P. (2019). Generative models for financial data. *Available at SSRN 3408007.*

[Hernandez, 2016] Hernandez, A. (2016). Model calibration with neural networks. *Available at SSRN 2812140.*

References II

[Horvath et al., 2019] Horvath, B., Muguruza, A., and Tomas, M. (2019). Deep learning volatility. *Available at SSRN 3322085.*

[Hu et al., 2019] Hu, K., Kazeykina, A., and Ren, Z. (2019). Mean-field langevin system, optimal control and deep neural networks. *arXiv preprint arXiv:1909.07278.*

[Jabir et al., 2019] Jabir, J.-F., Šiška, D., and Szpruch, Ł. (2019). Mean-field neural odes via relaxed optimal control. *arXiv preprint arXiv:1912.05475.*

[Jacquier and Oumgari, 2019] Jacquier, A. J. and Oumgari, M. (2019). Deep ppdes for rough local stochastic volatility. *Available at SSRN 3400035.*

[Kondratyev and Schwarz, 2019] Kondratyev, A. and Schwarz, C. (2019). The market generator. *Available at SSRN 3384948.*

[Mei and Montanari, 2019] Mei, S. and Montanari, A. (2019). The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355.*

[Neyshabur et al., 2017] Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. (2017). Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071.*

[Ni et al., 2020] Ni, H., Szpruch, L., Wiese, M., Liao, S., and Xiao, B. (2020). Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421.*

[Ruf and Wang, 2020] Ruf, J. and Wang, W. (2020). Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance, Forthcoming.*

[Vidales et al., 2018] Vidales, M. S., Šiška, D., and Szpruch, L. (2018). Martingale functional control variates via deep learning. *arXiv:1810.05094.*

References III

[Weinan, 2017] Weinan, E. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11.

[Wiese et al., 2020] Wiese, M., Knobloch, R., Korn, R., and Kretschmer, P. (2020). Quant gans: Deep generation of financial time series. *Quantitative Finance*, pages 1–22.

[Xu et al., 2020] Xu, T., Wenliang, L. K., Munn, M., and Acciaio, B. (2020). Cot-gan: Generating sequential data via causal optimal transport. *arXiv preprint arXiv:2006.08571*.