Neural Networks with Asymptotics Control

Asymptotic Splines and Constrained Radial Layers

Alexandre Antonov, Michael Konikov, and Vladimir Piterbarg

June 2020

Goal

- Machine learning (ML), deep learning (DL), and neural networks (NN) are becoming a standard piece of kit
- NNs have been proposed to 1) speed up slow function calculations and 2) calculate conditional expectations (see eg [AK], [MG] for a typical approach)
- Other applications have being explored: [AS], [BH], [BGT], [GR], [PHL] etc
- Generally the “learning” points are concentrated in the main body of a distribution and not its tails. A typical NN fitted to observations extrapolates outside of the learning “cloud” in a totally uncontrolled manner
- Our goal is to incorporate extra information we have (asymptotics) into an NN construction in other to control extrapolation
- Many important reasons such as stress testing of financial models, explainability, etc.

NN I

An NN is a multi-dimensional function parametrization

$$\begin{aligned} x &\to x^{(1)} = \sigma\left(W^{(1)}x + b^{(1)}\right) \\ &\to x^{(2)} = \sigma\left(W^{(2)}x^{(1)} + b^{(2)}\right) \\ &\dots \\ F_\theta(x) &= x^{(N)} = W^{(N)}x^{(N-1)} \end{aligned}$$

where

- each line in the scheme above is called a layer
- non-linear functions σ are called activation functions and are normally fixed
- the adjustable parameters for an NN are the weight matrices $W^{(i)}$ and bias vectors $b^{(i)}$

NN II

- We denote the NN parametrization as $F_\theta(x)$ where θ is the collection of all adjustable parameters over the layers, $\theta = \left\{ W^{(n)}, b^{(n)} \right\}_{n=1}^{N}$
- The parameters θ are determined by the so-called learning process (non-linear optimization in the old-speak):

  Given input points $\left\{ x^{(p)}, y^{(p)} \right\}_{p=1}^{P}$ *and a loss function, eg*

$$L(\theta) = \sum_{p=1}^{P} \left(F_\theta \left(x^{(p)} \right) - y^{(p)} \right)^2$$

  get optimal parameters θ^* *that minimizes the loss function*

- The non-linear optimization is performed numerically

NN III

- The NN parametrization is numerically attractive – its derivatives, necessary for the optimization, can be rapidly calculated
- The resulting function $F_\theta^*(x)$ approximates a “conditional expectation” $\mathbb{E}\left[Y \mid X\right]$.
- If the points $y^{(p)} = f\left(x^{(p)}\right)$ for some function f, the NN approximates the function itself, $F_\theta \simeq f$

Advantages and Drawbacks

Advantages

- Highly efficient and customizable (free parameters control, bespoke layers, GPU acceleration, cloud etc) software
- Universality of the approximation
- Theoretical guarantee of success given sufficient data

Drawbacks

- Numerically solving a non-linear highly multi-dimensional problem is unstable – local minima, overfitting, etc.
- No asymptotic (extrapolation) control in standard NNs – **our main focus**
- No interpolation control in standard NNs – even if the training points are well fit, one can potentially observe an unintuitive behavior *between* them, esp. for multi-layer NN

Theoretical Basis

Two theorems form the theoretical basis for the NN construction:

- Cybenko [Cyb] and Hornik [HSW]
- Kolmogorov-Arnold (Hilbert's 13th problem extension)
  See a review of [BG]

In [AKP] we show that information about the asymptotics is not extractable from the above theorems (esp., the K-A)

Asymptotics Control: The Plan

- Start with a function $f(x)$ we want to approximate while preserving its asymptotics
- Prerequisite: Know the asymptotics!
- First step: find a *control variate* function that has the same asymptotics – either in all directions or partially
- Proposal: use a multidimensional spline $S(x)$ which has the same asymptotics as $f(x)$
- The rest: approximate the residual function $R(x) = S(x) - f(x)$ with a special NN having vanishing (zero) asymptotics

1D Spline Reminder

- Spline (or cubic spline) $S(x)$ is a piece-wise cubic polynomial between its *nodes* $\{h_i\}_{i=1}^N$, i.e.

$$S(x) = a_i + b_i x + c_i x^2 + d_i x^3 \quad \text{for } x \in [h_i,\ h_{i+1})$$

- Values, derivatives and the second derivatives coincide at the nodes, i.e. $S(h_i - \varepsilon) = S(h_i + \varepsilon)$, $S'(h_i - \varepsilon) = S'(h_i + \varepsilon)$ and $S''(h_i - \varepsilon) = S''(h_i + \varepsilon)$
- These are almost sufficient to fix all the spline coefficients: we need 2 extra conditions — at the first node and the last one — to finish the job
- The classic spline has zero second derivatives at the boundaries (at the first node and the last one)

1D Spline without asymptotic control: BS approximation

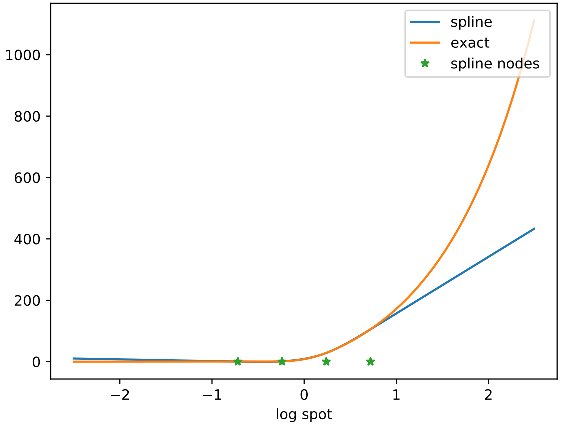

Figure: Classic spline fits the BS price at the four nodes (in green) and deviates at the tails

1D Spline with Asymptotics Control

- Instead of zeroing the second derivatives one can choose to fix the first derivatives at boundary points to arbitrary values
- This can be a way to fix *asymptotics*
- Suppose that we know the (calculation-heavy) original function behaviour in its tails, i.e. it can be analytically approximated: $f(x) \simeq f_-(x)$ for $x < h_0$ and $f(x) \simeq f_+(x)$ for $x > h_{N+1}$ for large negative h_0 and large positive h_{N+1}
- The recipe is simple:
  - Add the points h_0 and h_{N+1} to the spline nodes
  - Make the spline pass through them, $S(h_0) = f_-(h_0)$ and $S(h_{N+1}) = f_+(h_{N+1})$,
  - Also fix its first derivatives, $S'(h_0) = f'_-(h_0)$ and $S'(h_{N+1}) = f'_+(h_{N+1})$,
- This procedure does not guarantee the second derivatives fit

1D Spline with Asymptotics Control: Continuous Second Derivatives

- One can also assure the continuity of the second derivatives
- For this we pick up a point between h_0 and h_1, say $h_{\frac{1}{2}}$, and find (analytically) a value $S\left(h_{\frac{1}{2}}\right)$ such that we match the second derivative at h_0, $S''(h_0) = f''_{-}(h_0)$
- The same is valid for the right tail
- According to our experiments, the second derivatives control helps approximate "reasonable" functions (see [AKP] for details)

1D Spline with Asymptotics Control: BS Approximation

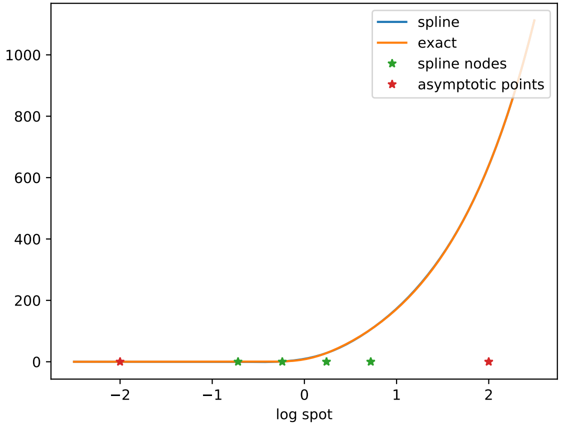

Figure: Spline fits the BS price for a fixed vol

Multi-Dimensional Splines with Asymptotics

- In a multi-dimensional setup we use a tensor product of 1D splines with asymptotics control in each direction (where they are available)
- Details can be found in [AKP]
- Below we demonstrate the asymptotics control effect using example of the BS price for $T = 1$, $K = 100$ in the log scale

2D Spline with Asymptotics Control: BS Approximation

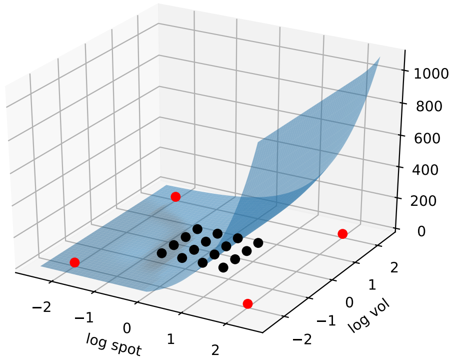

Figure: Blue surface: approximation; red dots: corners of the asymptotic region; dark blue dots: spline nodes (a 4×4 grid)

2D Spline with Asymptotics Control: Difference vs BS

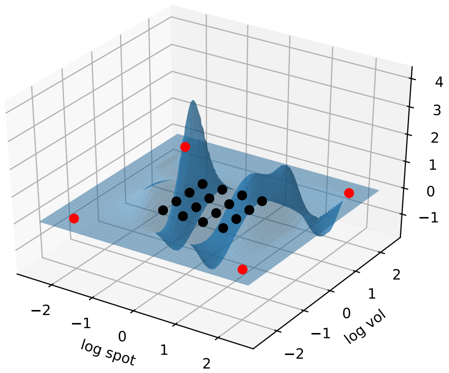

Figure: Blue surface: difference; red dots: corners of the asymptotic region; dark blue dots: spline nodes

2D Spline without Asymptotics Control: BS Approximation

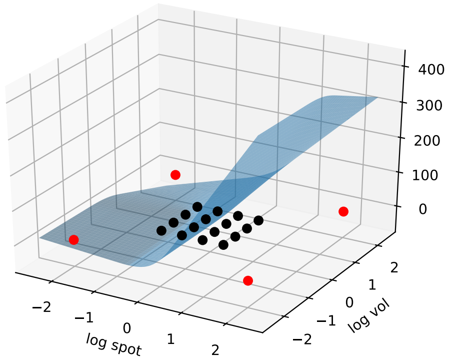

Figure: Blue surface: approximation; red dots: corners of the asymptotic region; dark blue dots: spline nodes

2D Spline without Asymptotics Control: Difference vs BS

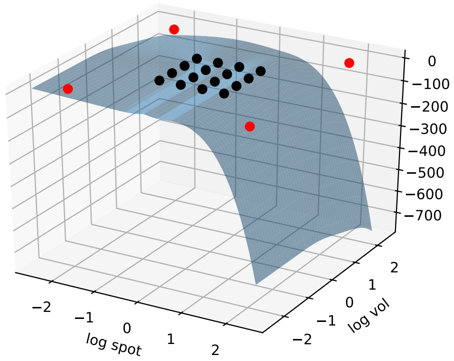

Figure: Blue surface: difference; red dots: corners of the asymptotic region; dark blue dots: spline nodes

NN with Vanishing Asymptotics I

- We propose to use the following representation of the function with vanishing/zero asymptotics

$$f_G(x) = \sum_{i=1}^{N} \lambda_i \, G\left(x \,|\, c^{(i)}, w^{(i)}\right)$$

where $G\,(x \,|\, c, w)$ is a Gaussian "bell" with a centre c (D-dim vector) and a width w ($D \times D$-dim matrix)[1]:

$$G\,(x \,|\, c, w) = \exp\left(-\frac{1}{2}(x-c)^T \left(w^{-1}\right)^T w^{-1}(x-c)\right)$$

- A numerical solver[2] changes the parameters of the function $f_G(x)$ to match the target values at the learning points

[1] In our numerical experiments we use a diagonal matrix

[2] The standard ADAM solver from Keras in our experiments

Points of Note

- We do not let the bells have arbitrary centres and widths: to guarantee zero asymptotics at a specific boundary we need to restrict them
- First idea: have a custom loss function that penalizes the bells that get close or over the asymptotic bounds
- According to our experiments, it is better to use a mapping function to make sure bells' centres and widths are where we want them to be
- A few hundreds of nodes/basis functions is enough to achieve excellent precision in our tests

Numerical experiments

We test our ideas on:

- the Black-Scholes function of spot and vol – described in details in [AKP]
- more complicated 4-dimensional SABR model – described below

4-dimensional SABR test

We test the normalized SABR model

$$\begin{aligned} dS(t) &= v\,\alpha(t)S(t)^{\beta}\; dW_1(t), \\ d\alpha(t) &= \gamma\,\alpha(t)\; dW_2(t), \\ S(0) &= 1, \quad \alpha(0) = 1. \end{aligned}$$

where v is an (approximate) ATM vol.

- Exact option price $\mathbb{E}\left[(S(T)-K)^{+}\right]$ analytics are available in [AKS] for our case of *zero* correlation.
- We fix the time $T = 1$ and study the normal implied volatility of the options above, a function of 4 arguments

$$\sigma_N(v, \beta, \gamma, K)$$

Remark. All the experiment details can be found in our support paper [AKP].

Parameters

- To demonstrate the flexibility of our approach, we control asymptotics in some but not all dimensions.
  We perform two experiments:
  - control asymptotics in the dimensions of the ATM vol v and the strike K only, and let the NN extrapolate in β and γ
  - control asymptotics in the dimensions of v, K and γ while β is our of control
- We sample four-dimensional vector of uncorrelated standard Gaussian random numbers, $(z_1, \ldots, z_4)$, and map those into some reasonable ranges for our variables.

Bounds

Since all the parameters are normalized (all z_j are standard Gaussian), we use the same bounds in all dimensions

- The learning bound $L_l = 1.0$ that defines the region where we train the NN on non-asymptotic values;
- The asymptotic bound $L_a = 1.5$ that defines the region outside of which we use asymptotics;
- The measurement bound $L_m = 2.0$ that defines the region where we measure performance/errors.

We then use $[-L_l, L_l]^4$, $[-L_a, L_a]^4$, $[-L_m, L_m]^4$ as the learning, asymptotic and measurement regions.

Remark. Asymptotic bounds are only relevant for asymptotics-controlled dimensions.

Tables below show the corresponding *standard*, i.e *non-normalized,* SABR parameter bounds

	lower bound	upper bound
v	0.034	0.677
β	0.023	0.977
γ	0.369	2.421
K	0.538	1.605

Table: Learning bounds for standard SABR parameters.

	lower bound	upper bound
v	0.016	1.432
β	0.001	0.999
γ	0.09	2.448
K	0.073	3.595

Table: Asymptotic bounds for standard SABR parameters.

	lower bound	upper bound
v	0.008	3.032
β	0	1
γ	0.01	2.5
K	0	13.723

Table: Measurement bounds for standard SABR parameters.

Note that the bounds for the strike depend on the ATM vol (see [AKP]) and the value in the table above correspond to the largest vol.

Asymptotic Splines and Constrained Radial Layers

- In the asymptotic spline we use a rather sparse grid of 3 nodes in each of the four dimensions of the normalized values of $\{-1, 0, 1\}^4$
- The goal here is to control the asymptotics and not necessarily have a close fit inside the learning region – that will be taken care of by the NN.
- Having removed the asymptotics in the v, K or γ dimension, we train the Constrained Radial Layer (CRL) NN to the residual.

Testing: Configuration

- The training set consists of $10,000$ standard Gaussian 4-dimensional samples of $(z_1, \ldots, z_4)$ that are then converted to (v, β, γ, K) using mappings from [AKS].
- To ensure a high sampling granularity and force the points to stay within their prescribed regions we sample enough points to get $10,000$ of them inside the appropriate region while rejecting those that are outside.
- The Constrained Radial Layer (CRL) used to fit the de-asymptotized values has 300 nodes (Gaussian kernels) with $2,700$ parameters to train.
  - For controlled dimensions, the centers and widths of the Gaussian kernels are within the bounds
  - For uncontrolled dimensions, the bounds for the centers and widths do not have constrains

- We compare our results with the standard NN method without any asymptotics control:
  - We use a NN with two hidden sigmoid[3] activation layers, one with 300 nodes and another with 4.
  - The number of training parameters, 2708, is very close to the CRL network.
- Once the approximation to the implied volatility function is learned, we generate a validation sample that covers the larger measurement region so that we can calculate various statistics, measuring the deviation of the fit from the known exact results in various regions.

[3]According to our experiments, the sigmoid activation functions are the most efficient for our application.

Testing: Errors

- Our tests show that controlling asymptotics significantly improves the NN fit to the SABR implied volatilities.
- We output mean-squared errors – the NN against the exact results
- We work in units of *log-normal* volatilities: $\sim$20% in the bulk but could be as high as 800% at the edges.
- "Learning error" is the mean-squared error over all learning points (that are all in the learning region).
- "Validation error" is the mean-squared error over all validation points.
- Their learning, asymptotic, and measurement region errors are mean-squared errors over validation points in the respective regions.

- "Model 1" is an a CRL NN where we control the vol, the strike and the vol-of-vol dimensions.
- "Model 2" is a CRL NN where we control the vol and the strike.
- "Model 3" refers to our baseline NN without asymptotics control – a deep NN with two hidden layers

Absolute error	Model 1	Model 2	Model 3
Learning error	0.026%	0.029%	0.029%
Validation error	0.75%	1.56%	15.16%
Learning region error	0.09%	0.08%	0.07%
Asymptotic region error	0.49%	1.25%	2.69%
Measurement region error	1.2%	2.3%	26.7%

Table: Summary results for SABR fitting.

Testing: Observations

- All three models fit the inputs equally well when measured at all the learning points ("Learning error" row) and over validation points in the learning region ("Learning error" row).
- Model 1 and Model 2 outperform Model 3 significantly (in some cases by an order of magnitude) outside the learning region
- Model 1 outperforms Model 2 measurably outside of the learning region as it has more dimensions under asymptotic control.

For further insight, we visualize test results in the following figures. As far as we work with four-dimensional functions, we fix two arguments and plot the rest two. More plots can be found in [AKS].

Testing: Plots

Fix β and γ and plot *errors* as function of atm vol and strike

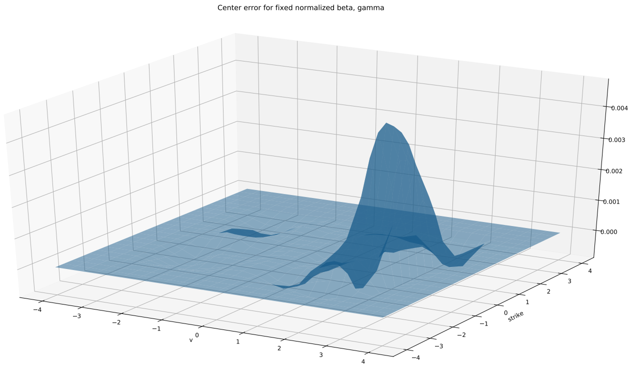

Figure: Volatility fitting error for NN with strike and volatility asymptotics control for fixed $\beta = 0.5$ and $\gamma = \sqrt{3}$.

Below we plot the approximation error for the NN with no asymptotics control – the learning region is significantly worse

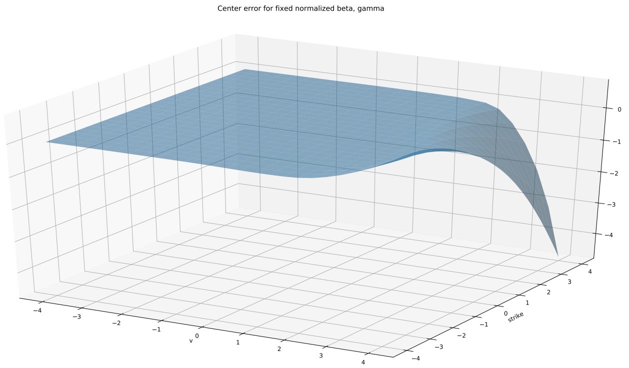

Figure: Volatility fitting error for NN with no asymptotics control for fixed $\beta = 0.5$ and $\gamma = \sqrt{3}$.

To show the effect of controlling asymptotics, we visualize the CRL error with the control on the strike and vol but *not* the skew and vol-of-vol.

Let us plot them for fixed volatility and skew

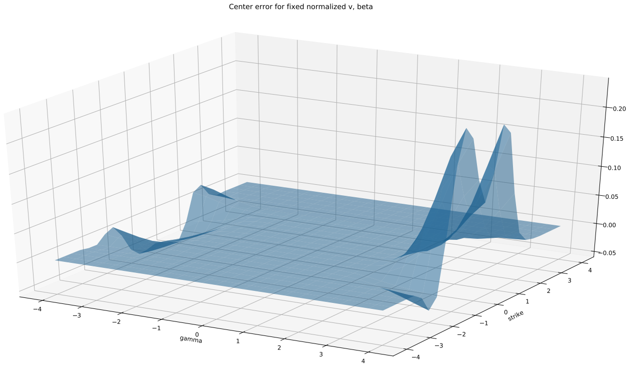

Figure: Volatility fitting error for NN with strike and volatility asymptotics control for fixed $v = 0.2$ and $\beta = 0.5$.

Let us see what happens to this error if we add vol-of-vol γ to the controlled dimensions we control – the error for large γ values is much smaller

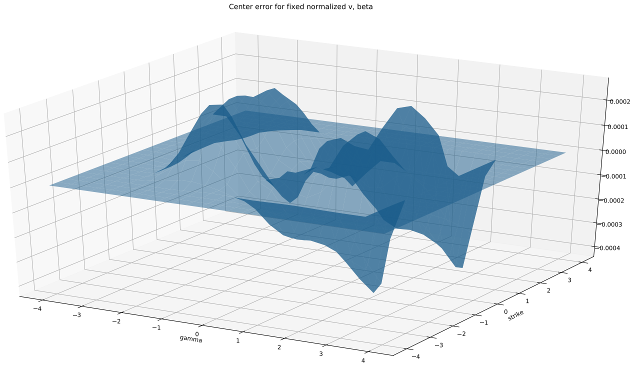

Figure: Volatility fitting error for NN with strike, volatility and volatility-of-volatility asymptotics control for fixed $v = 0.2$ and $\beta = 0.5$.

Conclusions

- While (deep) NNs are good in fitting/interpolation, they are not good in extrapolation
- This has profound consequences for their usefulness in math finance
  - Performance in regimes not seen before (happens all the time)
  - Stress testing
  - Explainability
- We propose a way to incorporate extra knowledge of the asymptotics into an NN
  - A special spline to "remove" the asymptotics
  - A special layer of Gaussian kernel functions with restrictions to approximate the reminder, a function with zero asymptotics at the boundaries
- Our results confirm our theoretical conclusions

References I

- A. Antonov, M. Konikov and M. Spector, ”Modern SABR Analytics”, 2019, Springer
- A. Antonov, M. Konikov and V. Piterbarg ”Neural Networks with Asymptotics Control” (2020), SSRN working paper: https://papers.ssrn.com/abstract=3544698
- J. Braun and M. Griebel “’On a constructive proof of Kolmogorov’s superposition theorem”
- B. Horvath, A. Muguruza and M. Tomas, “Deep Learning Volatility” (2019) SSRN working paper: https://ssrn.com/abstract=3322085
- G. Cybenko ”Approximation by superpositions of a sigmoidal function” Math. Control Signal Systems (1989) 2: 303.

References II

- H. Buehler, L. Gonon, J. Teichmann, B. Wood. "Deep Hedging", 2018, arxiv
- Kurt Hornik, Maxwell Stinchcombe and Halbert White "Multilayer feedforward networks are universal approximators", Neural Networks Volume 2, Issue 5, 1989, Pages 359-366
- A. Kondratyev, "Curve dynamics with artificial neural networks", 2018, Risk
- A. Kondratyev, C. Schwarz. "The Market Generator", 2019, SSRN working paper, https://ssrn.com/abstract=3384948
- W. McGhee, "An Artificial Neural Network Representation of the SABR Stochastic Volatility Model" (2018), SSRN working paper https://ssrn.com/abstract=3288882

References III

- P. Henry-Labordere. "CVA and IM: Welcome to the Machine", March 2019, Risk Magazine
- G. Ritter. "Machine learning for trading". October 2017, Risk Magazine, pp. 84-89