Risk publications

Risk

RISK MANAGEMENT • DERIVATIVES • REGULATION

Risk.net January 2021

Cutting edge
Machine learning

Deep asymptotics

Deep asymptotics

Artificial neural networks have recently been proposed as accurate and fast approximators in various derivatives pricing applications. Their extrapolation behaviour cannot be controlled due to the complex functional forms typically involved. Alexandre Antonov, Michael Konikov and Vladimir Piterbarg overcome this significant limitation and develop a new type of neural network that incorporates large-value asymptotics, allowing explicit control over extrapolation

Artificial neural networks (ANNs, or NNs), once trained, fit training values well and, with some care, interpolate between them in a reasonable way. Extrapolation beyond the range of training points, however, is not controllable in a typical NN.

The lack of control over extrapolation is a significant limitation of ANN utility in finance. Financial models often need to be evaluated at values of input variables that are significantly different from current market conditions. Changes of regime are common in financial markets. Input values in stress scenarios, required for sound risk management, routinely fall outside the range of a 'reasonable' training set, with unpredictable extrapolation. The coronavirus crisis is a live example of large market shocks.

One possible solution to this problem is sampling the input variable space widely enough so that any possible future value of the input falls within the sample range (interpolation) and never outside of it (extrapolation). It is clear this is not a fully satisfactory solution, as one does not know *a priori* what future values might be required.

Fortunately, for many financial applications, asymptotics for large values of parameters are known. The aim of this article is to demonstrate how the knowledge of asymptotics can be effectively translated into control over the extrapolating behaviour of NNs.

We propose a two-step approach to approximating a multidimensional function while preserving its asymptotics. First, we find a control variate function that has the same asymptotics as the initial function. Second, we approximate the residual with a special ANN that has vanishing asymptotics in all, or some, directions.

The apparent simplicity of the plan hides a number of complications that we overcome in this article. Specifically, we contribute two main technical results that make this program work. For step one, we show how to construct a universal control variate: a multidimensional spline that has the same asymptotics as the initial function. For step two, we design a custom NN layer that guarantees zero asymptotics in required directions, with fine control over the regions where the NN interpolation is used and where the asymptotics kick in.

Technical details and important theoretical background can be found in an online companion article (Antonov *et al* 2020).

Plan for controlling asymptotics

Define a function $f(x)$ that we want to approximate, while preserving its asymptotics. It goes without saying that the asymptotics need to be known.

In step one, we find a *universal* control variate function $S(x)$ that has the same asymptotics as $f(x)$. For multidimensional inputs – the most interesting case – these asymptotics could be known either in all directions or only in some; naturally, as much information about the asymptotics as is available should be incorporated into the control variate function $S(x)$. These functions – asymptotics-controlled splines – are described in the next section.

In step two, we approximate the residual function $R(x) = S(x) - f(x)$ with a special NN that has zero asymptotics in all, or some, directions. For this, later in the article we design a custom NN layer called a 'constrained radial layer' that guarantees zero asymptotics in all directions, with fine control over the regions where the NN interpolation is used and where the asymptotics kick in.

Asymptotics-controlled splines

A cubic spline $S(x)$ is a function given by a piecewise cubic polynomial between its nodes $\{h_i\}_{i=1}^N$. For a C^2 cubic spline, the values, derivatives and second derivatives are matched at the nodes. Two extra conditions are needed to fully specify the functional form. The classical 'natural' C^2 spline has zero second derivatives at the boundaries. In figure 1, we approximate the Black-Scholes option value, as a function of the log spot, with all other parameters fixed, using the natural spline. Clearly, the asymptotics are not captured.

Suppose we know the behaviour of the original, calculation-heavy function in its tails, ie:

$$f(x) \simeq \begin{cases} f_-(x) & \text{for } x < h_0 \\ f_+(x) & \text{for } x > h_{N+1} \end{cases}$$

for large negative h_0 and large positive h_{N+1}. To incorporate the asymptotics in the spline, we expand the set of spline nodes to include points h_0 and h_{N+1}, and we make sure the spline passes through them:

$$S(h_0) = f_-(h_0)$$
$$S(h_{N+1}) = f_+(h_{N+1})$$

To finish, we specify first-order derivatives at these new boundary points:

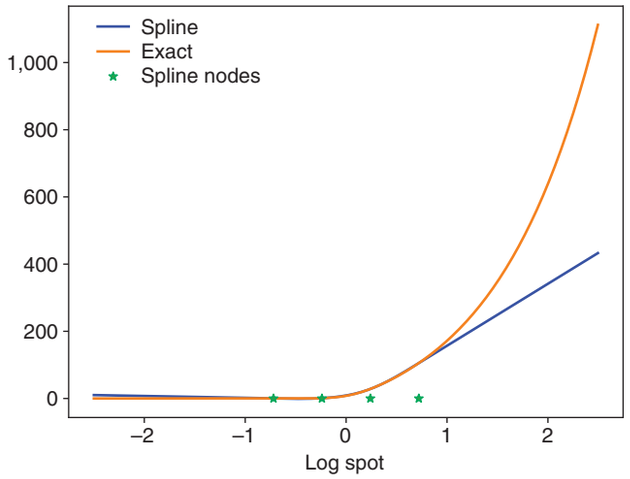
$$S'(h_0) = f'_-(h_0)$$
$$S'(h_{N+1}) = f'_+(h_{N+1})$$

We extend the approximating function $S(x)$ to outside of the spline range by specifying:

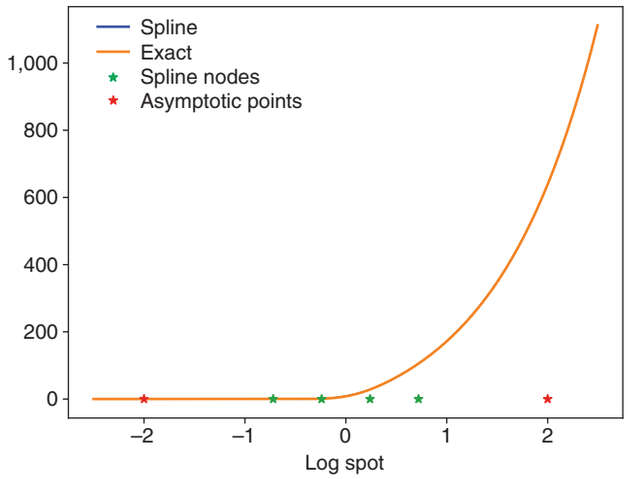
$$S(x) = \begin{cases} f_-(x) & \text{for } x \leqslant h_0 \\ f_+(x) & \text{for } x \geqslant h_{N+1} \end{cases}$$

This extension is C^1 but not C^2 at h_0 and h_{N+1}.

1 Fit of the natural spline to the Black-Scholes price as a function of log spot, fitted at green dots



2 Fit of the asymptotics-controlled spline to the Black-Scholes price as a function of log spot, fitted at green dots



It is possible – and, in fact, preferable – to construct a C^2 control variate spline-based function. Full details are available in Antonov *et al* (2020), but the idea is to pick a point $h_{1/2}$ between h_0 and h_1 and find, analytically, the value $S(h_{1/2})$ such that the second derivative at h_0 is matched, ie, $S''(h_0) = f''_-(h_0)$, with the same logic applied at the right end. Figure 2 demonstrates the improvements in the fit to the Black-Scholes function when we control the asymptotics and second derivatives at the end points.

A tensor product of one-dimensional asymptotics-controlled splines is used when multiple input dimensions are present. Details of the construction and calculation costs are presented in Antonov *et al* (2020). Here, we mention that the total multidimensional spline calculation cost per point is of the order:

$$O(T_S N_2 \cdots N_D + 2T_A N_3 \cdots N_D) \tag{1}$$

where T_A is the cost to calculate an asymptotic value, T_S is a one-dimensional spline calculation cost typically equal to several multiplications, and N_d is the number of nodes for those dimensions $d = 1, \ldots, D$ for which asymptotics are available.

Constrained radial layer (CRL)

After applying a control variate function, the problem reduces to fitting a function with vanishing, ie, zero, asymptotics. For that, we propose to use the following NN representation of the function with vanishing/zero asymptotics:

$$f_G(x) = \sum_{i=1}^{N} \lambda_i G(x \mid c^{(i)}, w^{(i)}) \tag{2}$$

where $G(x \mid c, w)$ is a Gaussian 'bell', or kernel, with a centre c (D-dimensional vector) and a width w ($D \times D$-dimensional matrix):

$$G(x \mid c, w) = \exp(-\tfrac{1}{2}(x-c)^{\mathrm{T}}(w^{-1})^{\mathrm{T}} w^{-1}(x-c)) \tag{3}$$

Sometimes these functions are called radial functions, so we call the NN layer with Gaussian kernels the (Gaussian) radial layer. We use the diagonal matrix:

$$w = \mathrm{diag}(w_1, \ldots, w_D)$$

so that (3) becomes:

$$G(x \mid c, w) = \exp\left(-\frac{1}{2}\sum_{d=1}^{D}(x_d - c_d)^2/w_d^2\right) \tag{4}$$

We choose Gaussian kernels as they are well understood by researchers, decay rapidly and ultimately seem to work well. Other families of radial basis functions could, of course, be used, but we do not explore this avenue further.

To achieve zero asymptotics outside a certain domain, or, equivalently, (near) zero values and first-order derivatives on the boundary of that domain for the fitting function, it is important to control the centres and widths of Gaussian kernels. In simple terms, they should be well inside the asymptotic region (as defined in the 'Experiments with SABR' section below and visualised in figure 3) so that all Gaussian kernels have small values at the boundary. We achieve such control by using mapping functions that keep kernels' centres and widths within the specified region.

For brevity, we denote $c = c^{(1)}$ to be the centre of the first kernel in the sum (2); all the others are handled identically. Here, $c = (c_1, \ldots, c_D)$, with D being the dimension of the problem, ie, the number of input arguments to the function we fit. Suppose our goal is to make sure c_1, the first coordinate of the centre, is always in the interval $[l, u]$ for $l < u$ (all the other coordinates are handled similarly). For that, we introduce the following mapping function:

$$c_1 = \tau_{l,u}(\bar{c}_1)$$

where:

$$\tau_{l,u}(x) = \frac{u\mathrm{e}^x + l\mathrm{e}^{-x}}{\mathrm{e}^x + \mathrm{e}^{-x}} \tag{5}$$

The function $\tau_{l,u}(x)$ is monotonically increasing and maps $\mathbb{R}$ to $[l, u]$. We apply similar transformations to all coordinates of all centres $c^{(i)}$, $i = 1, \ldots, N$, and replace the parametrisation (2) with:

$$f_G(x) = \sum_{i=1}^{N} \lambda_i G(x \mid \tau(\bar{c}^{(i)}), w^{(i)}) \tag{6}$$

The optimiser will now iterate over variables $\bar{c}^{(i)}$, $i = 1, \ldots, N$, with our mapping ensuring that Gaussian kernel centres $c^{(i)} = \tau(\bar{c}^{(i)})$, $i = 1, \ldots, N$, are always within the specified bounds.

Overfitting, an ever-present danger, needs to be controlled. The fitting procedure may try to place a very narrow Gaussian kernel centred at each fitting point with little regard for interpolation between the points. To discourage such behaviour and, ultimately, to control interpolation between the learning points, we introduce the minimum width $w_{\min} > 0$ and require that:

$$w_d^{(i)} \geqslant w_{\min}, \quad i = 1, \ldots, N, \; d = 1, \ldots, D \tag{7}$$

Constraining centres of Gaussian kernels to a certain domain is not sufficient to guarantee small values of kernels on a given asymptotic boundary as widths, if left unchecked during optimisation, can grow large enough for kernels to have non-vanishing values at the asymptotic boundary. Thus, we apply a similar mapping technique to control the widths in the context of the diagonal specification (4). Details are given in Antonov *et al* (2020).

Applications

The asymptotics-controlled CRL method is best suited for approximating slow-to-calculate multidimensional functions where asymptotics for some or all of the dimensions can be efficiently calculated.

The computational cost estimate (1) indicates that for the method to provide significant computational savings, the number of dimensions with asymptotics control should be in the single digits, provided that the asymptotic function calculation cost is equivalent to several multiplications and that three spline nodes per asymptotically controlled dimension, as we recommend, are used.

Our method is efficient for pricing functions in multifactor models that would normally require Monte Carlo or finite-difference calculations. It is typical that such functions have easily available asymptotics for at most 5–10 dimensions, such as spots and volatilities, thus fitting perfectly into our framework. Of course, the model should still be reasonably performant to produce several tens of thousands of learning points in a sensible amount of time, although alternatively this calculation can often be moved offline. For example, simulations of 10^5–10^6 paths for hundreds of time steps and tens of factors are a reasonable target for our method and will benefit greatly from the asymptotics-controlled NN approximation.

Presenting our method for such complex applications would burden the exposition with a large number of unnecessary details, so we consider a practically useful yet more manageable application in the next section.

It is worth mentioning that the utility of asymptotically controlled CRL NNs is not limited to approximating explicitly calculable functions. It is straightforward to apply the same method to the problem of calculating conditional expectations, a calculation underlying the American Monte Carlo and, more generally, Monte Carlo-based methods for solving high-dimensional partial differential equations and optimal control problems in finance. The details can be found in Antonov *et al* (2020, section 6), along with a comparison of the NNs with regression-based methods traditionally used in the American Monte Carlo. In brief, regressing against known basis functions is, of course, a much faster operation than fitting a non-linear CRL layer. What is gained in speed, however, is often lost in flexibility, and the merits of the two approaches should be judged against the specific requirements of a particular problem.

Experiments with SABR

In this section, we consider the benchmark problem of fitting the implied volatility function of the stochastic alpha beta rho (SABR) model: see McGhee (2018) and Horvath *et al* (2019) (a 'canonical' problem of approximating a Black-Scholes pricing function is presented in Antonov *et al* (2020)). We show that incorporating asymptotics into the approximation as we suggest dramatically improves the fit in the extrapolation region.

The standard parametrisation of the SABR model is given by:

$$\mathrm{d}S(t) = \alpha(t) S(t)^{\beta} \, \mathrm{d}W_1(t) \tag{8}$$

$$\mathrm{d}\alpha(t) = \gamma \alpha(t) \, \mathrm{d}W_2(t) \tag{9}$$

$$S(0) = S_0, \quad \alpha(0) = \alpha_0, \quad \mathrm{d}W_1 \, \mathrm{d}W_2 = \rho \, \mathrm{d}t \tag{10}$$

where S_0 is the spot, α_0 is the initial level of volatility, β is the skew, ρ is the correlation between the spot process and the volatility process, and γ is the volatility of volatility. We denote by $\sigma_{\mathrm{N}}(S_0, \alpha_0, \beta, \rho, \gamma, K, T)$ the implied normal volatility for a European call (or put) option with strike K and expiry T in the SABR model (8)–(10) with the parameters $(S_0, \alpha_0, \beta, \rho, \gamma)$. Our goal is to fit the function $\sigma_{\mathrm{N}}(\cdot)$ with an ANN using known asymptotics.

The focus of this article is on how to incorporate the known behaviour of the approximated function for large values of (some of) the parameters into the fitting procedure, and not on deriving/revisiting asymptotics for the SABR model (see Antonov *et al* (2019) for the latter). Hence, for brevity, we consider a case that can be treated exactly without approximations. This also helps us rigorously quantify errors of the NN approximations. Specifically, we set $\rho = 0$ and use the exact analytical formulas for European option values in the SABR model with zero correlation from Antonov *et al* (2019).

We apply a number of transformations to the original SABR parameters to reduce the dimensionality of the problem and normalise parameter ranges. First, we define:

$$v = \alpha_0 S_0^{\beta - 1}$$

and introduce normalised processes:

$$\bar{S}(t) = \frac{S(t)}{S_0}$$

$$\bar{\alpha}(t) = \frac{\alpha(t)}{\alpha_0}$$

European option prices in the original model are recovered from the normalised one via:

$$\mathbb{E}[(S(T) - K)^+] = S_0 \mathbb{E}[(\bar{S}(T) - \bar{K})^+]$$

where:

$$\bar{K} = K / S_0$$

so the two are equivalent for our purposes. As we explain later, we approximate SABR implied volatility for one fixed T at a time. It then follows that for a fixed T (and with ρ fixed at 0) the implied normal volatility is a function of four arguments (v, β, γ, K):

$$\sigma_{\mathrm{N}}(\cdot) = \sigma_{\mathrm{N}}(v, \beta, \gamma, K)$$

■ **Parameters.** For testing, we set time to expiry to one, so $T = 1$. We focus our testing on a configuration where we control asymptotics in the dimensions of the at-the-money volatility v and the strike K only. We let the approximating NN extrapolate in β and γ dimensions as it may. This configuration corresponds to the typical requirements of being able to apply spot and volatility stress scenarios while keeping the skew and volatility of volatility as well as the slope and curvature of the volatility smile near the 'normal' levels.

Some tests also include a configuration where the asymptotics in the γ direction is also controlled to show that, unsurprisingly, a better fit is achieved.

We non-linearly normalise the parameters:

$$(v, \beta, \gamma, K) \to (z_1, \ldots, z_4)$$

such that they most likely 'live' inside the interval $[-3, 3]$. The training set is thus generated by random sampling of a four-dimensional vector of uncorrelated standard Gaussian random numbers for the normalised variables z, and then by mapping them back into reasonable ranges for our initial parameters (v, β, γ, K). Our mapping is described in detail in Antonov *et al* (2020).

Let us now describe the training and testing regions for the parameters. As all the parameters are normalised to be of the same magnitude, we use the same bounds in all dimensions:

- the learning bound $L_\mathrm{l} = 1.0$ that defines the region where we train the NN on non-asymptotic values;
- the asymptotic bound $L_\mathrm{a} = 1.5$ that defines the region outside of which we use asymptotics; and
- the measurement bound $L_\mathrm{m} = 2.0$ that defines the region where we measure performance/errors.

We then use $[-L_\mathrm{l}, L_\mathrm{l}]^4$, $[-L_\mathrm{a}, L_\mathrm{a}]^4$ and $[-L_\mathrm{m}, L_\mathrm{m}]^4$ as the learning, asymptotic and measurement regions, respectively. Asymptotic bounds are only relevant for asymptotics-controlled dimensions, which, for most of the tests, are the volatility v and the strike K.

In addition, we label the following layers for ease of reference later: the layer $[-L_\mathrm{a}, L_\mathrm{a}]^4 \setminus [-L_\mathrm{l}, L_\mathrm{l}]^4$ is called a 'no-man's land' region, and the layer $[-L_\mathrm{m}, L_\mathrm{m}]^4 \setminus [-L_\mathrm{a}, L_\mathrm{a}]^4$ is called a 'peripheral' layer.

The regions (or, rather, their two-dimensional representations) are visualised in figure 3.

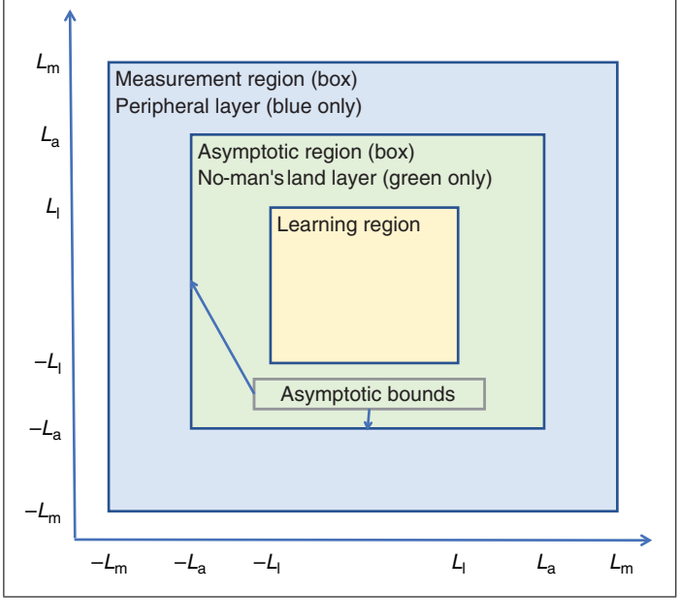
■ **Asymptotic splines and constrained radial layers.** Asymptotically controlled splines are constructed as they were earlier in the article. We use a rather sparse grid of three nodes in each of the four dimensions, at the normalised values of $\{-1, 0, 1\}^4$; we remind the reader that the point here is to control the asymptotics and not necessarily have a close fit inside the learning region, as that will be taken care of by the NN.

Having removed the asymptotics in the v and K dimensions, we train the CRL NN to the residual.

■ **Testing.** The training set is comprised of randomly generated standard Gaussian four-dimensional samples of $(z_1, \ldots, z_4)$ that are then converted to (v, β, γ, K) using mappings from the 'Parameters' subsection above. Concrete configurations of the training and validation sets are given below.

The CRL NN we use to fit the de-asymptotised values has 300 nodes (Gaussian kernels) with 2,700 parameters to train. For the dimensions where we aim to control asymptotics, we make sure the centres and widths of the Gaussian kernels are within the learning bounds, as described in the 'Constrained radial layer' section. For uncontrolled dimensions, the bounds for the centres and widths are set very wide. Note that our model has a fairly modest number of parameters relative to the size of our generated dataset, so we feel there is no danger of overfitting and decide not to use the dropout regularisation feature.

3 Regions of interest (see text for details)



We compare our results with standard feed-forward NN methods with and without coverage of the asymptotics region with learning points. For these, we use the same NN with two hidden sigmoid activation layers, one with 300 nodes and another with 4 nodes.[1] The number of training parameters, 2,708, 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, focusing on the deviation of the fit from the known exact results in various regions.

More details on the optimisation parameters and model configurations are available in Antonov *et al* (2020, section 8.5).

■ **Testing errors.** Our tests show that asymptotics control significantly improves the NN fit to the SABR implied volatilities. Table A summarises the results. Errors reported are mean-squared absolute errors against the exact solution. For comparison, the exact SABR normal volatilities have an average of 20% and a standard deviation of around 12% for the training set simulations, but they could be as high as 800% at the edges.

We test four models with different configurations. The first three models are fit to 10,000 learning points inside the learning region $[-L_\mathrm{l}, L_\mathrm{l}]^4$. The 'model 4' learning set consists of 10,000 points inside the learning region $[-L_\mathrm{l}, L_\mathrm{l}]^4$ and 5,000 points in the peripheral layer $[-L_\mathrm{m}, L_\mathrm{m}]^4 \setminus [-L_\mathrm{a}, L_\mathrm{a}]^4$ where asymptotics are available.

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

A. Absolute mean-squared errors of SABR volatility fitting

Absolute error	Model 1	Model 2	Model 3	Model 4	Num pts
Learning error	0.026%	0.029%	0.029%	0.98%	N/A
Validation error	0.75%	1.56%	15.16%	1.26%	10,000
Learning region error	0.09%	0.08%	0.07%	0.59%	2,600
No-man's land layer error	0.49%	1.25%	2.69%	0.94%	4,200
Peripheral layer error	1.2%	2.3%	26.7%	1.89%	3,200

All of the models share the same validation set of 10,000 points in the measurement region $[-L_{\mathrm{m}}, L_{\mathrm{m}}]^4$, split across regions as per table A.

Learning and validation points $(z_1, \ldots, z_4)$ are generated using a Gaussian distribution scaled to ensure ample coverage of the relevant regions. Points outside of the prescribed regions are discarded, so that the resulting distribution of the points is effectively a truncated Gaussian.

'Model 1' is a CRL NN where we control the asymptotics in the volatility, strike and volatility of volatility (γ) dimensions. 'Model 2' is a CRL NN where only the volatility and strike dimensions are controlled. We include these results to show the improvements achieved by controlling asymptotics in more dimensions. 'Model 3' refers to our baseline standard NN without asymptotics control. Finally, 'model 4' is a standard NN with learning points inside the learning region (10,000) and in the peripheral layer (5,000), which we use as a simple but realistic alternative to CRL NNs with asymptotics control (models 1 and 2). Models 3 and 4 share the same NN configuration described earlier.

'Learning error' is the mean-squared error over all learning points. 'Validation error' is the mean-squared error over all validation points: these cover the measurement region $[-L_{\mathrm{m}}, L_{\mathrm{m}}]^4$. As explained earlier, inside the measurement region we look at the results separately in the learning region $[-L_{\mathrm{l}}, L_{\mathrm{l}}]^4$ and in the two layers, the no-man's land layer and the peripheral layer. The learning, no-man's land and peripheral region/layer errors are mean-squared absolute errors over validation points in the respective regions/layers. We also report the number of validation points in each set in table A.

Models 1–3 fit the inputs equally well when measured at all the learning points ('Learning error' row) and over validation points in the learning region ('Learning region error' row).

Model 4 performs much worse in the learning region. Its validation error is 7–9 times larger than those for models 1–3. This is typical of an NN that is trying to balance the fit in different regions (the peripheral region versus the learning region in this case). We recall that all four models have the same number of trainable parameters.

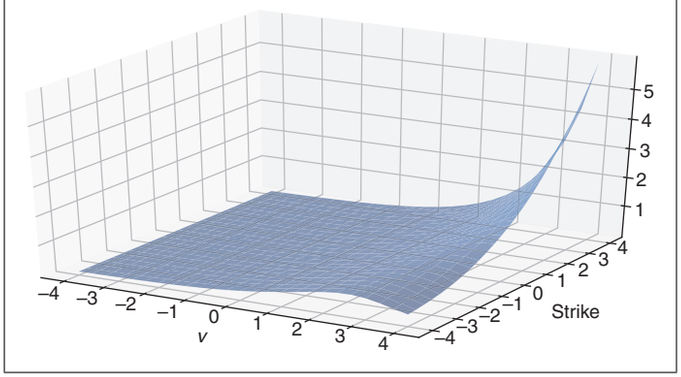
Clearly, model 1 and model 2 outperform model 3 significantly (in some cases by an order of magnitude) outside the learning region, as seen from the other rows. Also, not surprisingly, model 1 outperforms model 2 measurably outside of the learning region, as it has more dimensions under asymptotic control. The model 4 errors in the no-man's land and peripheral regions are somewhere between those for models 1 and 2. An improvement over model 3 is naturally explained by the presence of learning points in the peripheral region for model 4 (and their absence for model 3).

We point out that for model 4 the learning and validation errors are similar in magnitude, unlike those for the other models. This is due to the fact that for model 4 the learning and validation points span the same region, while that is not the case for the other models.

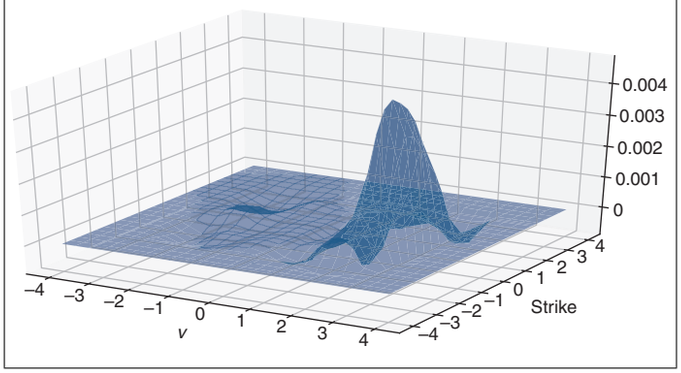
B. Training time in seconds on a standard desktop PC

	Model 1	Model 2	Model 3	Model 4
Training time (s)	726	753	578	1,361

4 SABR implied volatility ($\beta = 0.5$ and $\gamma = \sqrt{3}$)



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



To conclude, we present training times for the four models in table B. They show that adding extra asymptotic points to the learning set in model 4 significantly increases computational cost.

■ **Plots.** To visualise our approximations behaviour we plot them as functions of normalised ATM volatility v and normalised strike K while fixing the other normalised parameters to 0, ie, $z_2 = 0$ and $z_3 = 0$. This corresponds to the initial $\beta = 0.5$ and $\gamma = \sqrt{3}$. In Antonov *et al* (2020), one can find a series of plots for different fixed β and γ.

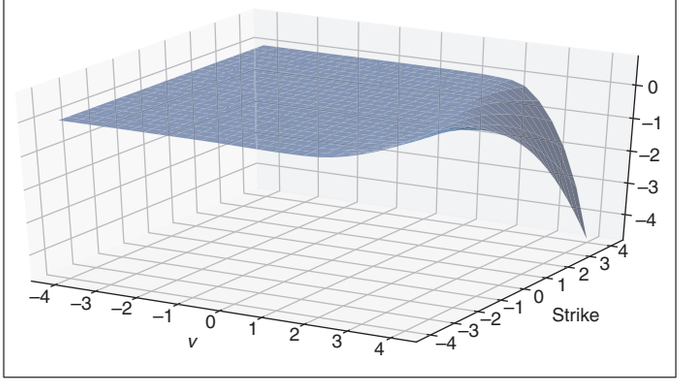
Figure 4 plots SABR volatilities that we are trying to fit. Note quite extreme values towards the edges of the region of interest.

In figure 5, we plot the fitting error of the NN versus exact SABR values using our method of asymptotics control.

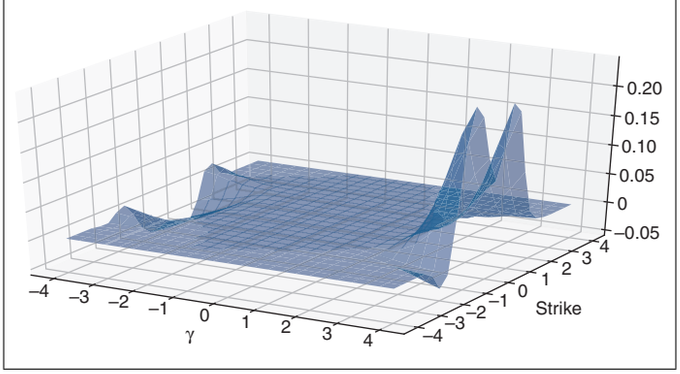
The same in the non-asymptotics-controlled case is plotted in figure 6.

Comparing the two plots in figures 5 and 6, we note the error is much smaller in the asymptotics-controlled case for all values of parameters, with the highest error coming from the non-controlled behaviour in the no-man's land region $[-L_{\mathrm{a}}, L_{\mathrm{a}}]^4 \setminus [-L_{\mathrm{l}}, L_{\mathrm{l}}]^4$, with the expected behaviour within the learning region $[-L_{\mathrm{l}}, L_{\mathrm{l}}]^4$ and the peripheral region $[-L_{\mathrm{m}}, L_{\mathrm{m}}]^4 \setminus [-L_{\mathrm{a}}, L_{\mathrm{a}}]^4$. However, it is still dwarfed by the error in the non-controlled

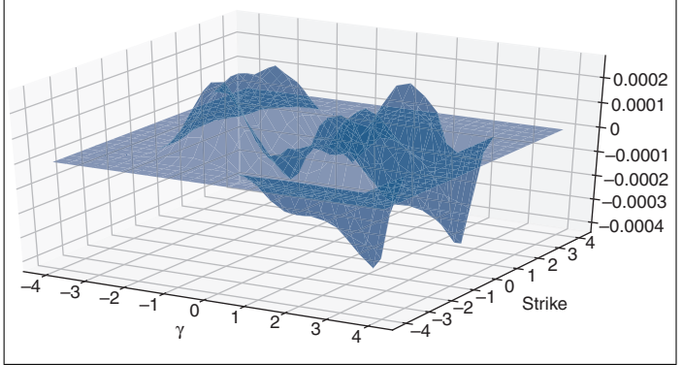
6 Volatility fitting error for NN with no asymptotics control for fixed $\beta = 0.5$ and $\gamma = \sqrt{3}$



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



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



case outside of $[-L_1, L_1]^4$. See figure 3 for a visual representation of these regions.

Next, let us look more closely at the effect of controlling asymptotics in more dimensions. In figure 7, we plot the fitting error for the NN where the strike and volatility dimensions are asymptotics-controlled but the skew and volatility of volatility are not, ie, our main test case. These results are for the same model as in figure 4, but we plot them for fixed volatility and skew so that we can explore the behaviour of the error for large strikes and volatility of volatilities.

Figure 8 demonstrates what happens to this error if we add volatility of volatility γ to the dimensions we control. Comparing this with figure 7, we see a much smaller error approaching 0 for now-controlled large γ values.

Conclusions

Uncontrollable extrapolation outside the training set range is widely acknowledged to be a serious weakness of traditional ANNs. The problem is particularly acute in financial applications given that financial models need to deal with rapid regime changes, stress tests and regulatory requirements on explainability. In this article, we develop a method to control ANN extrapolation by incorporating asymptotics of financial functions into the ANN construction. Such asymptotics are often already available for common financial functions, or could be derived using traditional techniques. ■

Alexandre Antonov is the chief analyst at Danske Bank in Copenhagen; Michael Konikov is the head of quantitative development at Numerix in New York; and Vladimir Piterbarg is the head of quantitative analytics and quantitative development at NatWest Markets in London.
Email: antonov22@gmail.com,
mkonikov1@gmail.com,
vladimir.piterbarg@natwestmarkets.com.

REFERENCES

Antonov A, M Konikov and M Spector, 2019
Modern SABR Analytics
Springer

Antonov A, M Konikov and V Piterbarg, 2020
Neural networks with asymptotics control
SSRN preprint, February, available at https://ssrn.com/abstract=3544698

Buehler H, L Gonon, J Teichmann and B Wood, 2018
Deep hedging
arXiv preprint

Ferguson R and AD Green, 2018
Deeply learning derivatives
SSRN preprint, available at https://ssrn.com/abstract=3244821

Henry-Labordere P, 2019
CVA and IM: welcome to the machine
Risk March, pages 76–81

Horvath B, A Muguruza and M Tomas, 2019
Deep learning volatility
SSRN preprint, available at https://ssrn.com/abstract=3322085

Kondratyev A, 2018
Curve dynamics with artificial neural networks
Risk June, pages 74–79

Kondratyev A and C Schwarz, 2019
The market generator
SSRN preprint, available at https://ssrn.com/abstract=3384948

McGhee W, 2018
An artificial neural network representation of the SABR stochastic volatility model
SSRN preprint, available at https://ssrn.com/abstract=3288882

Ritter G, 2017
Machine learning for trading
Risk October, pages 84–89