
NEURAL NETS & DNS



presents:

Comparison of 1 year forecasting of yield curves



FEBRUARY 28, 2018

M&A SOLUTIONS

Hong Kong

Contents

Introduction..... 2

Methodology 3

Data regularization..... 3

Model 4

 Structure..... 4

 Nodes 4

 Activation functions..... 5

 Optimisers 5

Results..... 7

Additional analysis..... 9

Future work..... 10



Introduction

How to set the central assumption for the mean of nominal yields is a problem with wide ranging implications for regulatory capital calculations, risk management as well as ALM and determining the optimal investment strategy for insurance and pension undertakings. It is well understood and evidenced that three factors (shift, twist, curvature) are the principal drivers of yield curve dynamics¹. Building on this an interesting approach to forecasting yield curves using a dynamic form of the Nelson Siegel ('DNS') Model is proposed by Diebold and Li². While these techniques often lead to significantly lower forecasting errors than other methods (e.g. forward curve or analyst predictions) they still essentially embed dynamics similar to the historical window used for estimation.

In this material we investigate whether the use of technologies such as neural networks and tensor flow can improve the forecasting accuracy of the Dynamic Nelson Siegel Model.



¹ Litterman, R., & Scheinkman, J. (1991). Common Factors Affecting Bond Returns. *Journal of Fixed Income*, 54-61

² Diebold, F., Li, C., & Yue, V. Z. (2008). Global yield curve dynamics and interactions: A dynamic NelsonSiegel approach. *Journal of Econometrics*, 351-363.

Methodology

Our model is a hybrid of a wide and deep model. We mirror an approach used when forecasting the term structure using the DNS Model and only allow the model to look at N years (in our case the optimal amount for economies considered was 5 years - 60 monthly periods) to predict the yield curve that lies 12 months ahead. The width (*i.e. the number of experts*) of the model takes advantage of various aspects that are not captured within the ARIMA framework - long term memory switches that can detect a regime change, correlation between the shift (Beta 1), slope (Beta 2) and twist (Beta 3). Further, whereas ARIMA can only be assessed against the components decomposition of the future yield curve, the neural net can be assessed against actual yield curve.

In order to minimize the effect of noise in the historical yield curve data we've opted to mimic the compression of this data to three factors using the ARIMA framework. [Side note: We've also performed some analysis on whether the auto-encoders could beat PCA decomposition for this problem, and found that PCA remained the optimal solution at least in the mean time]. We've found that this provided optimal structure for enabling comparison: it had the same inputs and the same information compression of the process in both the neural network and DNS approaches. Removal of the 3 component (as per PCA) constraint would likely increase performance, but this was outside of the scope of our research.

To control for periods and economic cycles we've attempted to train on slightly different data for all economies. Additionally, we varied the structure slightly according to economy. In some cases a deeper network was more beneficial; in others a shallow one performed the best.

We trained the model on one set of 60 period segments, allowing for shuffling, validated it against 20 (or so) subsequent periods, and tested it against the next 20 periods. This means that between model training, and the actual predictive capacity for which it is being used there is around 2 years of gap. (We take Bloomberg month end interpolated data periods from say April '06 to February '14 to train the model, March '14 to October '15 to validate, and the rest of the data to June '17 to test the model). Although this is not reflective of reality of building these models (they may be used without the test period), it still provides a robust justification to the validity of our approach.

SOLUTIONS

Data regularization

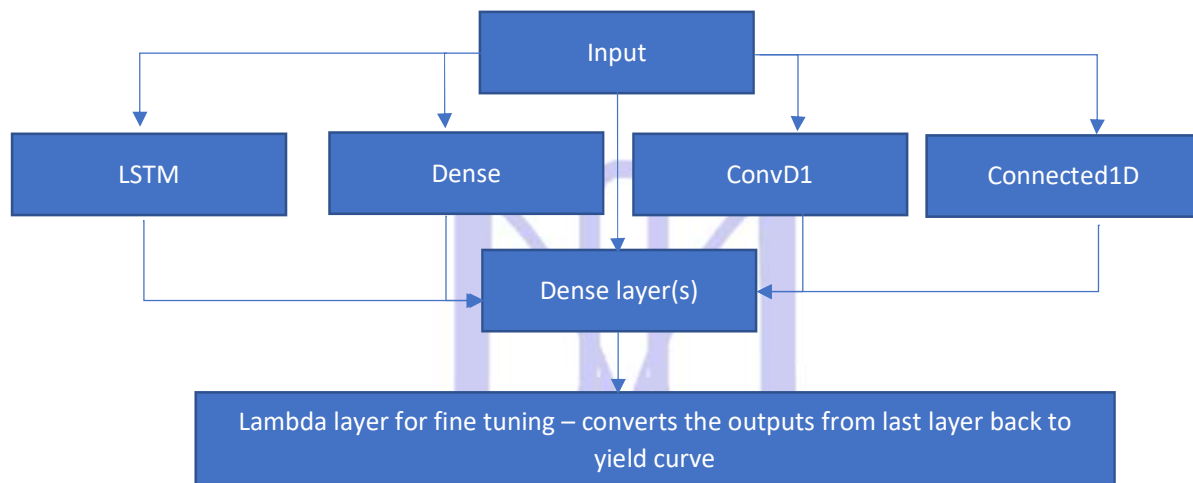
To enable the neural network to perform to the best of its efficiency we normalise the data prior to feeding it into the neural net (standardising by the history (60 periods) for the input). Although there are standardization functions readily available within the Sci-kit libraries in Python, our standardization was done manually (not recommended). The additional benefit of training directly to the yield curve means that no de-standardisation was needed.

Model

The model consists of 4 experts and a “look through” layer which enables the model to use the data directly. We will refer to experts as layers, as this is commonly accepted terminology in the sphere. By a layer we mean specific neural net with computational architecture suited to a purpose.

Structure

Previous work in this sphere condemned the use of deep networks as inaccurate for financial analysis³. In our work this did not prove to be absolutely true (and although we attempted to calibrate a Deep Boltzmann Machine for predictive purposes in MATLAB, in the end it didn't perform well), as with shorter iterations, and without re-conversion back into the yield curve it was a lot more difficult to correctly fit a deep network. However, we've found that dropout rate in addition to extra nodes typically improved the generalisation power of the network.



Nodes

Our network had a look through layer as per Hinton⁴, to enable it to assess which experts should be granted more credence. The four expert networks are as follows:

- 1) LSTM 120 nodes with dropout of 50% - Long-Short Term Memory is typically cited as the feature of recurrent neural networks most equivalent to ARIMA: the layer has the ability to open and close "gates" to retain information. There are additional features of retention that we didn't employ in this model (like 'Stateful' implementation - which enables to better tune the LSTM). Our original network included three layers of these, however as we started comparing with ARIMA they had poor performance.⁵
- 2) Dense 6 nodes with dropout of 50% - perhaps the simplest of the layers as it multiplies each unit of the input by the number of nodes contained in the layer and apply the activation function and bias (none in our case). We used dropout (when half the nodes randomly

³ Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 215-236.

⁴ <http://www.cs.toronto.edu/~fritz/absps/jjnh91.pdf>

⁵ <https://keras.io/layers/recurrent/#lstm>

disappear and bring the actual resulting network to 3 nodes), so as to amplify the predictive power of the network.⁶

- 3) Convolution Layer (1D) with 9 filters - this layer filters the information that comes in to the model - the filters applied are the same for all data - in our case we opted by a 3x3 structure to deflect the derivative aspects of the betas.⁷
- 4) Locally connected 1D – as above in 3), however each section of the data fed to the layer. This layer may "hear" specific noises coming in from different historical data points.⁸

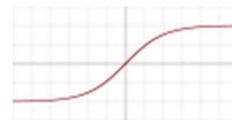
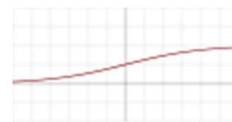
We've then proceeded to compress the information coming out of these experts into a single layer and feed into either another dense layer or directly to the last 3 node layer which is then taken to be the changes to the historical means and standard deviations to the betas.

As a side note, we are sure that as more and more examples of implementation of layers become prevalent, the easier the addition of layers becomes which is very likely to lead to direct improvement of the model performance. Although, at first blush overfitting may be a concern, techniques like dropout and kernel/bias constraints automatically reduce the models ability to overfit (they essentially mean that the model cannot always rely on a node being always available).

Activation functions

Although there are multiple activation functions (image source wiki⁹): we have focused on three main shapes that are simple to distinguish, and readily available within the Keras package.

- 1) Sigmoid (aka logistic or Soft step) – smooths out small variations as less sensitive to small changes, this function is sensible to use when you are trying to get to generalize to the “bigger picture” i.e. image recognition.
- 2) RELU – regularized linear unit works well for economies which have a positive trend to their yields. This was a small number.
- 3) Tanh – hyperbolic tangent, most sensitive to small changes. This made the function optimal for picking up the small changes in betas that sigmoid function "smoothed over".



Optimisers

Most of the problems have a complex space of solutions. Indeed, the more unknowns added the more complex the optimisation of problems of weights and biases becomes. Below is an overview of more frequently used optimisers with some of their features.

- 1) RMS Prop– optimiser that works exceptionally well on Recurrent Neural Nets (as per our modelling starting point) and the technique behind the optimiser is smoothing out variations in convergence by dividing them by average over number of immediately previous periods.¹⁰

⁶ <https://keras.io/layers/core/#dense>

⁷ <https://keras.io/layers/convolutional/#conv1d>

⁸ <https://keras.io/layers/local/#locallyconnected1d>

⁹ <https://keras.io/optimizers/#parameters-common-to-all-keras-optimizers>

¹⁰ www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

- 2) Adadelta¹¹ – adaptive learning rate method - the optimiser speeds up or slows down the gradient descent depending on first order information emerging from computation.
- 3) SGD – stochastic gradient descent, has fewer specific designs than the two above, but essentially provides ability to fit the optimisation to any problem, as the underlying approach is supremely simple: the weights of the network are adjusted in accordance with the error rate relative to the targets. That intuition although computationally intensive allows confidence in convergence of solution to the problem over time.

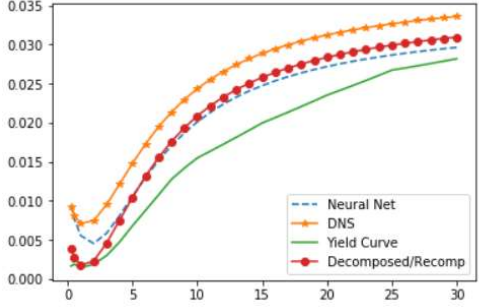
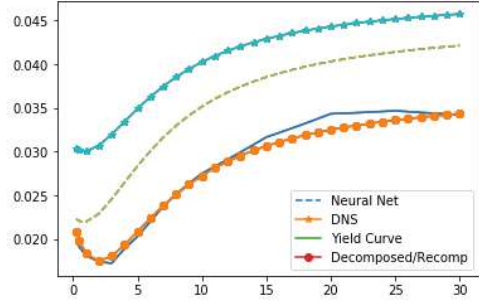
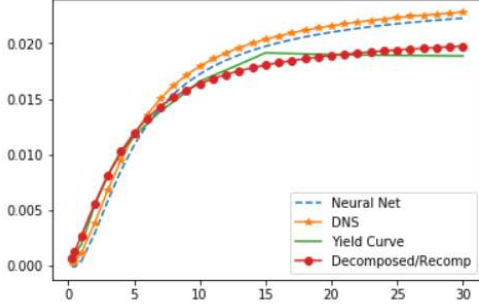


¹¹ <https://arxiv.org/abs/1212.5701>

Results

Based on the following model we were able to obtain the following results for these economies:

Economy	DNS sum abs validation error	NN sum abs validation error	Sample plot
US	4.58341216	4.39226404	<p>Comparison between actual and predicted for two methods US</p>
UK	4.47859626	4.06658903	<p>Comparison between actual and predicted for two methods UK</p>
DE	7.10978443	5.58480586	<p>Comparison between actual and predicted for two methods DE</p>
JP	1.41906627	0.92467229	<p>Comparison between actual and predicted for two methods JP</p>

SG	3.49509066	2.79304426	<p>Comparison between actual and predicted for two methods for SG</p> 
AU	6.26584138	4.57419012	<p>Comparison between actual and predicted for two methods AU</p> 
HK	0.77290543	0.66875669	<p>Comparison between actual and predicted for two methods HK</p> 

Additional analysis

We also had a look at possibilities of using US data to infer HK curves and AU curves, since the original data provided short history to work with.

These networks require further amendments to the structure, and hence the performance on existing structure is cited below. As it is obvious the information does get lost between the two countries, but we are working to investigate a) renormalisation via HK/AU rather than US as is currently done b) adding an additional information into the model about the difference between the economies

Economy	DNS sum abs validation error	NN sum abs validation error	NN x US sum abs val error	Sample plot
HK	2.07945826	2.02418896	2.00145044	<p>Comparison between actual and predicted for two methods HK-US</p>
AU	6.26584138	5.14440047	4.68136195	<p>Comparison between actual and predicted for two methods AU-US</p>

Future work

This work can be extended to incorporate market indicators, commentary and even central bank announcements.

Residual neural nets and regression analysis can also be a powerful enhancement, as some of the errors are trending in their nature across the training or validation or testing periods.

Below you can see an example of such an occurrence, which chronological results of residuals from one of our models. The errors show a consistent trend across time. Hence, corrections could be made to further enhance the model accuracy via residual modelling.

