Deep learning for financial derivative pricing.

Name: An Shen
Department of Computer Science
Stanford University
anshen21@stanford.edu

1 Introduction

How to quickly and accurately price a financial derivative is critical to the profitability of a bank. From trading perspective, as the market moves, an instantaneous re-price of financial derivative can help traders make critical decisions. From risk management perspective, the more risk scenarios you can look into, the better control you may have.

However, as financial derivatives are getting more and more complicated, the pricing tends to be slower and slower. How to price financial derivatives using deep learning method is now a very active area for both the academics and practitioners. In this paper, we will use deep learning methods to price a variety of exotic financial products:

- Basket Losses
- Bermudan Swaption
- Callable Bonds
- CDS
- Convertible Bonds
- CVA Swap
- Equity Option
- Replication

More over, feature engineering is not very common with financial data, as most data are well structured and well understood. In this paper, we will use SVI[1] parametrization of volatility surface as a novel way to see if feature engineering can improve the DNN's performance for financial data. I believe this is the first time we are trying this approach.

2 Related Work

Many previous work has been done in this area since 1990[2][3][4][5]. However, most of the academic research are focused on simple derivatives such as vanilla stock option. Or, they tend to use simplified input data, such as a flat volatility surface (one number for the whole surface).

In this paper, we will focus on using DNN to price complicated financial products with more comprehensive data and also with feature engineered data. (Such as interest rate curve, volatility surface and hazard rate.)

3 Dataset and Features

Due to license issue, for this exercise I cannot use official vendor data. Quantlib (`https://www.quantlib.org/`) is used as the tool to generate the data. Quantlib is one of the best financial derivative libraries in the public domain. It's been used as a standard benchmark in many settings in the financial industry.

When use Quantlib to generate the data (simulate the market data, calculate the derivative price), a very important decision is what parameters to ignore, as to train on all the parameters is not practical

for this project. Here are the major input data features I chose to generate, based on the importance to pricing and the relevance to this project:

- Recover Rate
- Interest Rate
- Interest Rate Volatility
- Equity Volatility
- Moneyness
- CDS Spread
- Hazard Rate

The output is the derivative price.

Here is a sample of the data(for CDS_E2):

Derivative Price, Calculation Time, IR 1M, IR 2M, IR 3M, IR 6M, IR 9M, IR 1Y, IR 2Y, IR 3Y, IR 4Y, IR 5Y, IR 6Y, IR 7Y, IR 8Y, IR 9Y, IR 10Y, IR 12Y, IR 15Y, IR 20Y, IR 30Y, Hazard Rate, Recover Rate

816298, 0, 0.0003, 0.000225, 0.000405, 0.00092, 0.00128, 0.001685, 0.001115, 0.00138, 0.001765, 0.00226, 0.00286, 0.003525, 0.00421, 0.00486, 0.00545, 0.006435, 0.007485, 0.0085, 0.009105, 0.02, 0.41

4 Baseline

The main baseline model is the one in [5], It has 4 hidden layers of 100 neurons each. The activation functions are LeakyReLU, ELU, ReLU and ELU respectively for each layer. The final output layer comprises a single output neuron set to be the standard exponential function so the output is non-negative. At each hidden layer a dropout rate of 25% is applied. The loss function used for optimization is mean-squared error (MSE). The batch size is 64. The model is trained with 10 epochs.

I do feel that training with 10 epochs was a limitation of hardware at the paper's time. I tried 500 in my effort to replicate the baseline result. Please notice that the main baseline model is only run for the Equity Option product, as it was designed for in [5].

A second baseline of using Linear Regression is also conducted, as a comparison to other methods in general.

5 Methods

We will use MLP here and try to optimize the hyper parameters.

Due to limited resource, a few hyper parameters are chosen based on past experience: The activation function used is Swish. Past experience showed that Swish is normally no worse than other popular activation functions. The batch size is set to 2 for small data set, 8 for medium data set and 32 for large data set, as I want to take the most advantage of the regularization effect of Mini-Batch . The optimization function is Adam.

The data set is split 60/20/20 (Train/Dev/Test). A random search is used to find the number of DNN layers (2 to 10) and the number of neurons (8 to 1024) at each layer, per different financial product.

Each run will stop after certain amount of epochs based on the observation of convergence (200, 300 or 500, depends on the derivative). The model which provides the best Dev result will be used for the Test. I will not use any explicit regularization method as I believe the DNN size is by itself a good regularization.

Mean absolute percentage error(MAPE) is used to measure the accuracy of the result. Mean absolute error(MAE) is provided as a reference. The loss function is still MSE.

One particular test (No Feature_VOL v.s. Feature_SVI) is designed to see if SVI parametrization feature engineering can improve the result. SVI parametrization is basically using a function with 5 parameters to represent a slice of volatility surface, which is a curve at a particular expiry date. The original data may have 20 or more actual volatility data points for each slice. We use original data points or the 5 SVI parameters as two different input data sets(plus other data); and see which one gives the better result.

6 Experiments/Results/Discussion

Here is the DNN result:

Product Name (_ Numerical Method)	Layer	Neuron	Batch Size	MAPE	MAE
BasketLosses_BaseCorrelationGLHP	7	8	8	0.042755	0.012496
BasketLosses_GaussianBinomial	4	256	8	0.13112	0.036747
BasketLosses_GInhomogeneous	4	256	8	0.140218	0.040784
BasketLosses_GLHP	8	8	8	0.038944	0.011354
BasketLosses_RandomG	7	32	8	0.10879	0.031611
BasketLosses_RandomLossG	6	256	8	0.037375	0.010441
BasketLosses_RandomT	2	1024	8	0.098458	0.028586
BasketLosses_TBinomial	2	1024	8	0.086039	0.023834
BermudanSwaption_BK	7	64	32	1.294388	0.11244
BermudanSwaption_G2FDM	4	128	32	1.175972	0.084476
BermudanSwaption_G2Tree	5	1024	32	1.064222	0.092518
BermudanSwaption_HW2FDM	9	32	32	1.077003	0.072749
BermudanSwaption_HW2Tree	9	512	32	1.023689	0.088785
BermudanSwaption_HWFDM	8	64	32	0.944994	0.066299
BermudanSwaption_HWTree	6	128	32	1.743009	0.107169
CallableBonds	4	16	2	0.042959	0.035312
CDS_E1_1Y	3	512	2	0.052219	39.04919
CDS_E1_2Y	3	128	2	0.049734	72.51462
CDS_E1_3M	5	32	2	0.051927	12.71989
CDS_E1_6M	5	32	2	0.038149	14.25715
CDS_E2	6	256	8	0.166011	23043.64
ConvertibleBonds_AdditiveEquiprobabilities_AM	5	32	2	0.129612	0.156451
ConvertibleBonds_AdditiveEquiprobabilities_EU	4	64	2	0.104203	0.12691
ConvertibleBonds_CoxRossRubinstein_AM	7	512	2	0.159052	0.19051
ConvertibleBonds_CoxRossRubinstein_EU	6	128	2	0.091093	0.110924
ConvertibleBonds_JarrowRudd_AM	5	64	2	0.121032	0.147271
ConvertibleBonds_JarrowRudd_EU	7	16	2	0.088149	0.108797
ConvertibleBonds_Joshi_AM	10	16	2	0.105794	0.130677
ConvertibleBonds_Joshi_EU	6	1024	2	0.09917	0.118956
ConvertibleBonds_LeisenReimer_AM	4	32	2	0.14115	0.168813
ConvertibleBonds_LeisenReimer_EU	7	64	2	0.092445	0.111749
ConvertibleBonds_Tian_AM	10	8	2	0.095824	0.118909
ConvertibleBonds_Tian_EU	8	512	2	0.107463	0.12911
ConvertibleBonds_Trigeorgis_AM	9	256	2	0.116447	0.143198
ConvertibleBonds_Trigeorgis_EU	8	64	2	0.096126	0.116057
CVASwap_10	4	32	32	0.09869	0.000391
CVASwap_15	10	32	32	0.100703	0.000785
CVASwap_20	4	128	32	0.057521	0.000721
CVASwap_25	5	128	32	0.049069	0.000811
CVASwap_30	6	32	32	0.069239	0.001363
CVASwap_5	5	128	32	0.391234	5.61E-05
Replication_BarrierOption	6	256	8	0.540374	0.004269
EquityOption_BinomialCoxRossRubinstein_AM	5	16	8	0.397571	0.027521
EquityOption_BlackScholes_EU	9	16	8	0.417976	0.012054

Product Name (_ Numerical Method)	Layer	Neuron	Batch Size	MAPE	MAE
EquityOption_BlackVasicek_EU	8	8	8	0.143927	0.009882
EquityOption_FiniteDifferences_AM	7	16	8	0.331846	0.01263
EquityOption_HestonSemiAnalytic_EU	8	8	8	0.39605	0.008421
EquityOption_MCCrude_EU	8	32	8	0.400269	0.012334
EquityOption_MCLongstaffSchwartz_AM	7	1024	8	0.36481	0.020632
EquityOption_QMCSobol_EU	7	32	8	0.436067	0.00796

Based on the result above, except Bermudan Swaption, all other products have a MAPE less than 1%. And the MAPE of Bermudan Swaption is less than 2%. This means DNN can be used to approximate the pricing function.

The result for the main base line (EquityOption_BlackScholes_EU) is MAPE 22.1487 and MAE 0.405890. Which is much higher than the new architect (MAPE 0.417976 and MAE 0.012054). The main reason to me is the dropout layer. Based on my past experience, with regression, dropout seems always make things a lot worse.

The result for the second base line is in the Attachment. On average, the MAPE for DNN is 0.303017597, and the MAPE for Linear Regression is 0.539843749. Not surprisingly, DNN is better than Linear Regression.

The result for Feature Engineering v.s. No Feature Engineering is below. It shows that Feature Engineering actually made things worse.

Feature Engineered/ No Feature Engineer	Layer	Neuron	Batch Size	MAPE	MAE
Feature_SVI	5	8	2	0.722857	0.620889
No Feature_VOL	10	32	8	0.207944	0.106228

7 Conclusion/Future Work

Based on the result, we can say that DNN can be used to price financial derivatives. Future work would include using vendor data to conduct the final test, advocate the result in financial community and complete an open source solution to this problem.

8 Contributions

An Shen

9 References

[1] Jim Gatheral, Antoine Jacquier. Arbitrage-free SVI volatility surfaces. 2013.

[2] M. Malliaris and L. Salchenberger. A neural network model for estimating option prices. *Journal of Applied Intelligence*.3(3):193–206, 1993b.

[3] Marco J. Morellia, Guido Montagna, Oreste Nicrosinib, Michele Treccani, Marco Farina, Paolo Amato. Pricing Financial derivatives with neural networks. Physica A 338 (2004) 160 – 165.

[4] Ralf Herrmanny and Alexander Narr. Neural Networks and the Valuation of Derivatives Some Insights into the Implied Pricing Mechanism of German Stock Index Options. 1996 meeting of the German Finance Association.

[5] Robert Culkin & Sanjiv R. Das. Machine Learning in Finance: The Case of Deep Learning for Option Pricing. Santa Clara University. 2017.

10 Attachment

Here is the Linear Regression result:

Product Name (_ Numerical Method)	MAPE	MAE
BasketLosses_BaseCorrelationGLHP	0.000601532	0.017714
BasketLosses_GaussianBinomial	0.008737159	0.248782
BasketLosses_GInhomogeneous	0.00747137	0.214686
BasketLosses_GLHP	0.000601532	0.017714
BasketLosses_RandomG	0.007721531	0.221249
BasketLosses_RandomLossG	0.008564095	0.240277
BasketLosses_RandomT	0.004003282	0.116539
BasketLosses_TBinomial	0.008569334	0.242658
BermudanSwaption_BK	0.305201791	2.434856
BermudanSwaption_G2FDM	0.279433899	2.267207
BermudanSwaption_G2Tree	0.281529108	2.354132
BermudanSwaption_HW2FDM	0.347518299	2.395602
BermudanSwaption_HW2Tree	0.341299604	2.384304
BermudanSwaption_HWFDM	0.364128963	2.420076
BermudanSwaption_HWTree	0.357556313	2.413801
CallableBonds	0.006742506	0.53191
CDS_E1_1Y	0.053495462	2277.981
CDS_E1_2Y	0.088860028	6707.733
CDS_E1_3M	0.019215533	274.4301
CDS_E1_6M	0.031455321	756.4332
CDS_E2	1.150928166	3873359
ConvertibleBonds_AdditiveEquiprobabilities_AM	0.001926878	0.232074
ConvertibleBonds_AdditiveEquiprobabilities_EU	0.00257342	0.298468
ConvertibleBonds_CoxRossRubinstein_AM	0.003050999	0.371314
ConvertibleBonds_CoxRossRubinstein_EU	0.003982303	0.472256
ConvertibleBonds_JarrowRudd_AM	0.002729203	0.329506
ConvertibleBonds_JarrowRudd_EU	0.003647749	0.42928
ConvertibleBonds_Joshi_AM	0.003169582	0.386793
ConvertibleBonds_Joshi_EU	0.004066814	0.483292
ConvertibleBonds_LeisenReimer_AM	0.003169587	0.386794
ConvertibleBonds_LeisenReimer_EU	0.004066797	0.48329
ConvertibleBonds_Tian_AM	0.002712681	0.327446
ConvertibleBonds_Tian_EU	0.003635481	0.427431
ConvertibleBonds_Trigeorgis_AM	0.003020617	0.367709
ConvertibleBonds_Trigeorgis_EU	0.003935586	0.466668
CVASwap_10	0.369289977	0.103546
CVASwap_15	0.274443157	0.201693
CVASwap_20	0.22851758	0.280259
CVASwap_25	0.204133249	0.332474
CVASwap_30	0.191392562	0.365715
CVASwap_5	1.45782471	0.013403
Replication_BarrierOption	1.085713793	0.820648
EquityOption_BinomialCoxRossRubinstein_AM	2.294670029	0.880491
EquityOption_BlackScholes_EU	2.739269937	0.909345
EquityOption_BlackVasicek_EU	0.437071874	0.854498
EquityOption_FiniteDifferences_AM	2.290314272	0.880702
EquityOption_HestonSemiAnalytic_EU	2.738992904	0.909346
EquityOption_MCCrude_EU	2.910880444	0.910994
EquityOption_MCLongstaffSchwartz_AM	3.332728584	0.870196
EquityOption_QMCSobol_EU	2.717621867	0.909338