A Review of Factor Modeling with Machine Learning

Presented by: Ivy Yang

Department of Economics
University of Miami

March 22, 2022

Purpose of this presentation

In this presentation, I want:

- Summarize the factor model using machine learning methods in recent years.
- Discuss the Neuron Network approach.
- Go through financial instruments that worth to be studied in my future research.

Literature

Reference paper:

- "A Factor Model for Option Returns" Buchner and Kelly (2022)
- "Empirical Asset Pricing via Machine Learning"
  Gu Kelly and Xiu (2020)
- "Deep Learning in Asset Pricing" Chen Pelger and Zhu (2019)
- "Characteristics are Covariances: A Unified Model of Risk and Return"
  Kelly, Pruitt and Su (2019)
- "A Neural Network-Based Framework for Financial Model Calibration"
  Liu Borovykh Grzelak and Oosterlee (2019)
- "Cross Section of Option Returns and Idiosyncratic Stock Volatility"
  Cao and Han (2013), Only theoretical part

Roadmap

1. Background Knowledge
2. Machine Learning Algorithms
3. Potential Extension

Background Knowledge

Theoretical base

All of our modeling methods are based on the No-Arbitrage pricing theory by Lucas (1987)

$$\mathbb{E}_t\left[M_{t+1}R^e_{t+1,i}\right]=0 \Leftrightarrow \mathbb{E}_t\left[R^e_{t+1,i}\right]=\underbrace{\left(-\frac{\mathrm{Cov}_t\left(R^e_{t+1,i},M_{t+1}\right)}{\mathrm{Var}_t\left(M_{t+1}\right)}\right)}_{\beta^{\mathrm{SDF}}_{t,i}}\cdot\underbrace{\frac{\mathrm{Var}_t\left(M_{t+1}\right)}{\mathbb{E}_t\left[M_{t+1}\right]}}_{\lambda_t},$$

where the excess return: $R^e_{t+1,i}=R_{t+1,i}-R^f_{t+1}$

The SDF (M_{t+1}) could be pinned down by the tangent portfolio:

$$M_{t+1}=1-\sum_{i=1}^{N}\omega_{i,t}R^e_{i,t+1}=1-\omega_t^T R^e_{t+!}$$

Theoretical base

Target of machine learning

- **For stocks:** Maximize the mis-pricing. By Chen et. al (2020)

$$\min_{\omega} \max_{g} \frac{1}{N} \sum_{j=1}^{N} \left\| \mathbb{E}\left[\left(1 - \sum_{i=1}^{N} \omega\left(I_t, C_{t,i}\right) R^e_{t+1,i} \right) R^e_{t+1,j} g\left(I_t, C_{t,j}\right) \right] \right\|^2$$

- **For options:** Using the delta-hedging strategy. By Cao and Han (2013)
  Delta-hedged gain measures the change in the value of a self-financing portfolio consisting of a long call position, hedged by a short position in the underlying stock.

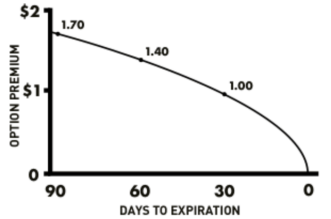
$$\Pi_{[1,T]} = \sum_{t=1}^{T-1} \left(F_{t+1} - F_t\right) - \sum_{t=1}^{T-1} \Delta_t \left(S_{t+1} - S_t\right) - \sum_{t=1}^{T-1} \frac{a_{t,t+1} r_t}{365} \left(F_t - \Delta_t S_t\right)$$

Here $\Pi_{[t,t+\tau]}$ is the excess dollar return of delta-hedged call option.
Normalize it to make it comparable across stocks: $\Pi(t, t+\tau) / \left(\Delta_t S_t - C_t\right)$

Option Greeks:

- **[**δ**:]**Delta is the amount an option price is expected to move based on a 1 change in the underlying stock.
  As a general rule, in-the-money options will move more than out-of-the-money options, and short-term options will react more than longer-term options to the same price change in the stock.
- **[**θ**:]**Time decay:

Figure 2: Time decay of an at-the-money call option



This graph shows how an at-the-money option's value will

Option Greeks:

- **[ν:]**Vega is the amount call and put prices will change, in theory, for a corresponding one-point change in implied volatility.
- **[ρ:]**Rho is the amount an option value will change in theory based on a one percentage-point change in interest rates.
- **[γ:]**Gamma is a measure of sensitivity to jump risk, which is an important driver of option's factor betas.

Background knowledge about options

Review the Black-Scholes Model:

Based on:

$dS_t = \mu S_t dt + \sigma S_t dz_t$

European call pricing:

$$C = N(d_1) S_t - N(d_2) K e^{-rt}$$

$$\text{where } d_1 = \frac{\ln \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right) t}{\sigma \sqrt{t}}$$

$$\text{and } d_2 = d_1 - \sigma \sqrt{t}$$

- C: call option price
- N: CDF if the normal distribution
- S_t: spot price of an asset
- K: strike price
- r: risk-free interest rate
- t: time to maturity
- δ volatility of the asset

Auxiliary variables are:

- $D = e^{-r\tau}$: discount factor
- $F = e^{r\tau} S = \frac{S}{D}$ is the forward delta of the underlying asset, and $S = DF$ ($\Delta_{it}^{Fwd} = \Delta_{i,t} \cdot e^{d_{i,t}(T_i - t)/365}$)

Background knowledge about options

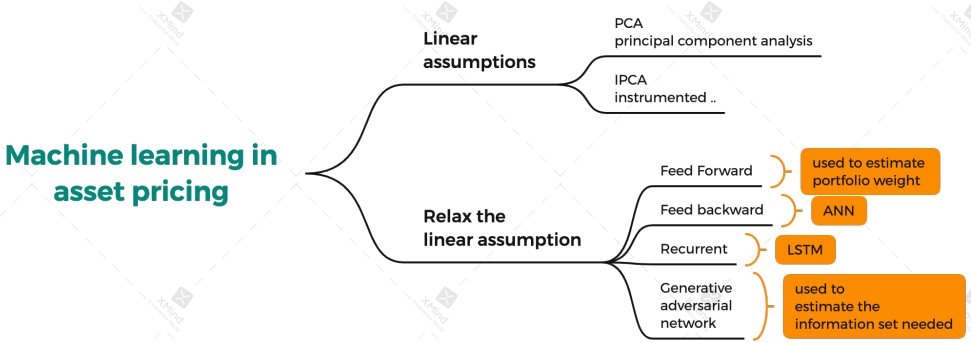
Risk sources:

This will be used in the IPCA interpretation.

- **Overall risk of the volatility**
- **Maturity risk**
- **Risk from moneyness skew of the volatility surface**

Machine Learning Algorithms

Summary about machine learning methods



General information

Why Machine Learning?

Including the no-arbitrage constraint in the learning algorithm significantly improves the risk premium signal/noise ratio, and makes it possible to better explain individual stock returns.

Importance of the loss function
(`https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/`**):**

Loss function is an evaluation of a machine learning model.

Where we can find loss functions?
There are many forms of loss function, here are two examples:

- **in Chen et.al (2019):** $L\left(\omega \mid g, h_t^g, h_t, C_{t,i}\right)$ is the average of $\min_\omega \max_g \frac{1}{N} \sum_{j=1}^{N} \left\| \mathbb{E}\left[\left(1 - \sum_{i=1}^{N} \omega\left(I_t, C_{t,i}\right) R_{t+1,i}^e\right) R_{t+1,j}^e g\left(I_t, C_{t,j}\right)\right]\right\|^2$
- **in Liu et.al (2019):** $L(\boldsymbol{\theta}) := D(f(\mathbf{x}), F(\mathbf{x} \mid \boldsymbol{\theta}))$ with target: $\arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta} \mid (\mathbf{X}, \mathbf{Y}))$

PCA

PCA and IPCA are discussed in Kelly et.al (2019), Buchner and Kelly (2022).
They use the PCA to estimate SDF as a linear function of characteristics: $w_{i,t} = \theta^\top I_{i,t}$

PCA and IPCA are based on the assumption that loadings (β) are linear in characteristics.

Code reference:

`https://mpelger.people.stanford.edu/data-and-code`
`https://github.com/bkelly-lab/ipca`

PCA

Algorithm of PCA

PCA computes a set of input variables, and transforming them into a set of uncorrelated variables, called PCs.

This transformation behaves in way such that the first PC explains the largest possible variance, so on so forth.

Algorithm of IPCA:

On top of PCA model, IPCA explicitly accounts for time variation in individual asset behavior by allowing risk factor loadings, $\beta_{i,t}$, to depend on observable asset characteristics.

The characteristics serve as instrumental variables for conditional betas, which avoids the limitations of static betas in time series regression.

It can be interpreted as capturing three flavors of risk:

1. Overall level of volatility surface.
2. Maturity risk that summarized by the term structure slope of the volatility surface.
3. Moneyness Skew of the volatility surface.

Algorithm of IPCA:

Model for an excess return $r_{i,t+1}$

$$r_{i,t+1} = \alpha_{i,t} + \beta_{i,t} f_{t+1} + \epsilon_{i,t+1},$$
$$\alpha_{i,t} = z'_{i,t}\Gamma_\alpha + v_{\alpha,i,t}, \quad \beta_{i,t} = z'_{i,t}\Gamma_\beta + v_{\beta,i,t}.$$

- f_{t+1}: Latent factors
- $\beta_{i,t}$:
  1. Instrumenting the estimation of latent factor loadings with observable factors.
  2. Incorporating time-varying instruments to estimate dynamic factor loadings.
- Γ_β: The matrix mapping from a potentially large characteristics to limited factors.
- Any behavior of dynamic loadings that is orthogonal to instruments will absorbed by $\nu_{\beta,i,t}$

PCA

The target of IPCA

Will be recaped on page 32.

PCA

Pros and Cons

- **PCA:**

  Pros: Requiring no ex ante knowledge of the structure of average returns.

  Cons:

  1. PCA is inapt for estimating conditional versions of $r_{i,t+1} = \alpha_{i,t} + \beta'_{i,t} f_{t+1} + \epsilon_{i,t+1}$ since it can only cope with static loadings.
  2. PCA lacks the flexibility for a researcher to incorporate other data beyond returns to help identify a successful asset pricing model.

- **IPCA:**

  Pros: Allow dynamics in the model, and brings information beyond just returns into estimation of factors and betas

  Cons: Compared to RNN models, this model cannot depict the information learning process.

Feed forward Algorithm - ANN

- **ANN (CaNN)**
  The basic ANN is the multi-layer perceptron (MLP), which can be written as a composite function:

$$F(\mathbf{x} \mid \boldsymbol{\theta}) = f^{(L)}\left(\ldots f^{(2)}\left(f^{(1)}\left(\mathbf{x}; \boldsymbol{\theta}^{(1)}\right); \boldsymbol{\theta}^{(2)}\right); \ldots \boldsymbol{\theta}^{(L)}\right)$$

  where $\boldsymbol{\theta}^{(i)} = (\mathbf{w}_i, \mathbf{b}_i)$, $^{\mathbf{b}}\,\mathbf{w}_i$ is a weight matrix and b_i is a bias vector.
  Layers in this model could be written as:

$$\begin{cases} y(\mathbf{x}) = \varphi^{(2)}\left(\sum_j w_j^{(2)} z_j^{(1)} + b^{(2)}\right) \\ z_j^{(1)} = \varphi^{(1)}\left(\sum_i w_{ij}^{(1)} x_i + b_j^{(1)}\right) \end{cases}$$

  where $\varphi\left(w_{1j} x_j + b_{1j}\right)$ is the neuron's basis function, m is the number of neurons in a hidden layer. This form also allows a backward learning process.

Feed forward Algorithm - ANN

Algorithm graph in Liu et.al (2019)

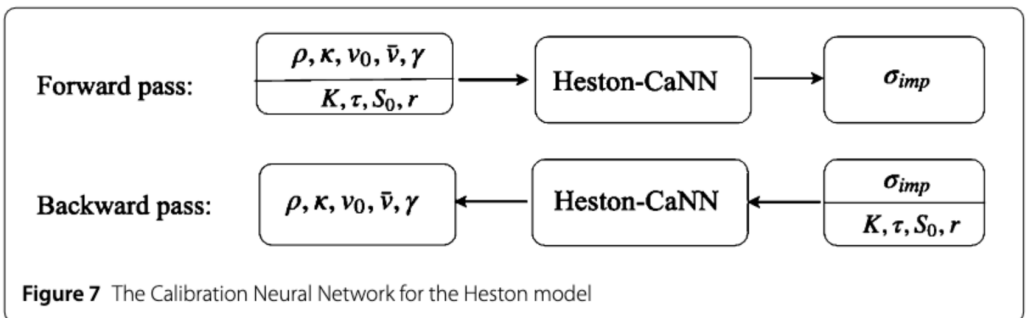


Figure 7 The Calibration Neural Network for the Heston model

Feed forward Algorithm - FFN

The target is to setup a function from input x to y: $y = f(x)$

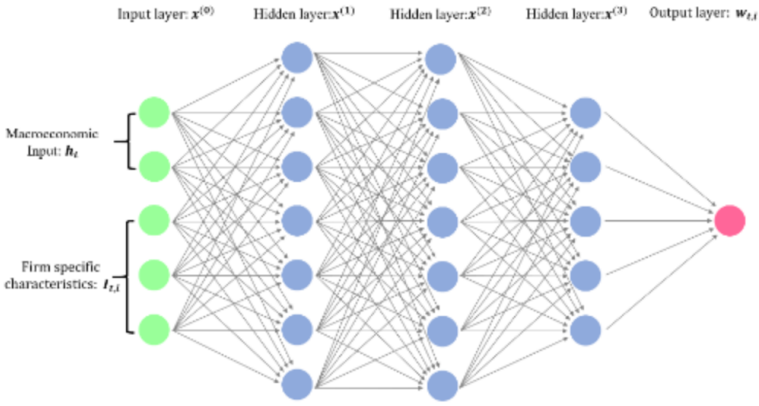
- They choose the rectified linear unit (ReLU): $\text{ReLU}(x_k) = \max(x_k, 0)$
- The hidden layers: $x^{(1)} = \left(x_1^{(1)}, \ldots, x_{K^{(1)}}^{(1)}\right)$ and it depends on the parameters: $W^{(0)} = \left(w_1^{(0)}, \ldots, w_{K^{(0)}}^{(0)}\right)$ and a bias term $w_0^{(0}$
- The output layer is simply a linear transformation of the output from the hidden layer:

$$x^{(1)} = \text{ReLU}\left(W^{(0)\top} x^{(0)} + w_0^{(0)}\right) = \text{ReLU}\left(w_0^{(0)} + \sum_{k=1}^{K^{(0)}} w_k^{(0)} x_k^{(0)}\right)$$

$$y = W^{(1)\top} x^{(1)} + w_0^{(1)} \quad \text{with } x^{(1)} \in \mathbb{R}^{K^{(1)}}, W^{(0)} \in \mathbb{R}^{K^{(1)} \times K^{(0)}}, W^{(1)} \in \mathbb{R}^{K^{(1)}}$$

Feed forward Algorithm - FFN

Figure 3. Feedforward Network with 3 Hidden Layers



Feed forward Algorithm - FFN

Steps in Chen et.al (2019) They applied FFN in estimating $x = [I_t, I_{t,i}]$:

- They acquired the optimal weights in GAN.
- Get the optimal instrumrnts for the moment conditions in GAN.
- Calculate the conditional mean return.
- Get the second moment ($y = R^e_{t+1,i} F_{t+1}$)

Feed forward Algorithm

Where it was applied to

- **ANN (Artificial Neural Network or Calibration using ANNs):** In Liu et.al (2019)
- **FFN:** In Chen et.al (2019)

Pros and Cons

- This model outperforms tree learning approaches and other linear and nonlinear prediction models
- **Cons:** In the real trading process, there will be dynamics between the last term's output and this term's input. FFN cannot depict the dynamic by itself.

Recurrent Neural Network

RNNs are a family of neural networks for processing sequences of data. They estimate non-linear time-series dependencies for vector valued sequences in a recursive form.

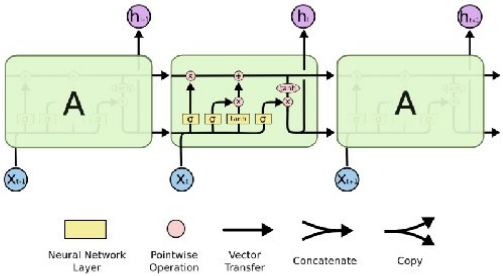
- **LSTM:** The LSTM is designed to deal with lags of unknown and potentially long duration in the time series, which makes it well-suited to detect business cycles.
- **GAN:** The GAN includes two models: A generative model G (detects the data distribution) and a discriminative model D (shows the probability that sample coming out of G. They play a two-player game and reach out to a unique solution that maximize the probability that D makes mistakes. Hence no Markov chain is needed.

Recurrent Neural Network - LSTM

Algorithm of LSTM (Long-Short Term Memory): relevant code: `https://github.com/topics/long-short-term-memory`

LSTM: Long Short Term Memory

- The basic structure of LSTM and some symbols to aid understanding



Recurrent Neural Network - LSTM

Where it was applied to

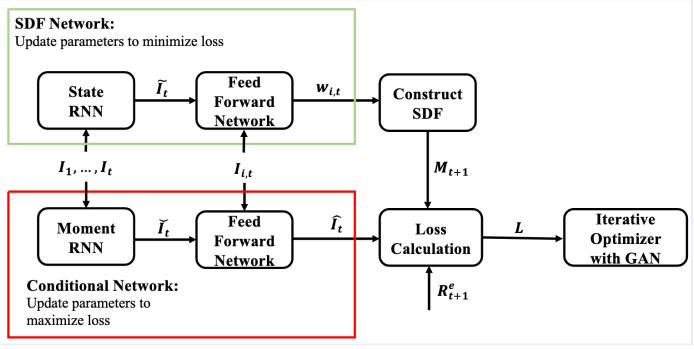
- **LSTM:** Chen Pelger and Zhu (2019)

Pros and Cons

- **Pros:** The LSTM approach can deal with both the large dimensionality of the system and a very general functional form of the states while allowing for long-term dependencies.
- **Cons:** This type of structure is powerful if only the immediate past is relevant, but it is not suitable if the time series dynamics are driven by events that are further back in the past.
- In Chen et.al, they constructed LSTM that deals with unknown and potentially long duration in the time series, so that they can detect business cycles.

Recurrent Neural Network - GAN

Algorithm of GAN (Generative Adversarial Network):



Code reference:

https://github.com/LouisChen1992/Deep_Learning_in_Asset_Pricing

https://github.com/goodfeli/adversarial

Recurrent Neural Network - GAN

Steps:

- Obtain an initial guess of the SDF by updating the SDF network to minimize the unconditional loss.
- For a given SDF network, maximize the loss by varying the parameters in the conditional network.
- Fix the parameters in the conditional network and train the SDF network to minimize the conditional loss

instead of directly using I_t as an input each network summarizes the whole macroeconomic time series information in the state process h_t (respectively h_t^g for the conditional network):

$$\left\{\hat{\omega}, \hat{h}_t, \hat{g}, \hat{h}_t^g\right\} = arg\,\min_{\omega,h_t}\max_{g,h_t^g} L\left(\omega \mid \hat{g}, h_t^g, h_t, I_{t,i}\right)$$

Recurrent Neural Network - GAN

Where it was applied to

- **GAN:** Chen Pelger and Zhu (2019)

Pros and Cons - Haven't discussed yet

I may detect the weakness of this kind of setting and improve it as a topic.

Evaluation of fitness

Target of the literature

- **Minimize the sum of squared composite model errors. By Kelly et.al (2019), Gu et.al (2020)**.

  $\min_{\Gamma_\beta, F} \sum_{t=1}^{T-1} (r_{t+1} - Z_t \Gamma_\beta f_{t+1})' (r_{t+1} - Z_t \Gamma_\beta f_{t+1})$

  with FOC:

  $$\hat{f}_{t+1} = \left(\hat{\Gamma}'_\beta Z'_t Z_t \hat{\Gamma}_\beta\right)^{-1} \hat{\Gamma}'_\beta Z'_t r_{t+1}, \quad \forall t$$

  $$\text{vec}\left(\hat{\Gamma}'_\beta\right) = \left(\sum_{t=1}^{T-1} Z'_t Z_t \otimes \hat{f}_{t+1} \hat{f}'_{t+1}\right)^{-1} \left(\sum_{t=1}^{T-1} \left[Z_t \otimes \hat{f}'_{t+1}\right]' r_{t+1}\right)$$

Evaluation of fitness

Target of the literature

- **Explained variation (Left) and cross-sectional mean (Right). By Chen et.al (2020)**

$$EV = 1 - \frac{\left(\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t}\left(\hat{e}_{t+1,i}\right)^2\right)}{\left(\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t}\left(R^e_{t+1,i}\right)^2\right)} \qquad \mathrm{XS}-R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}\frac{T_i}{T}\left(\frac{1}{T_i}\sum_{t\in T_i}\hat{e}_{t+1,i}\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\frac{T_i}{T}\left(\frac{1}{T_i}\sum_{t\in T_i}\hat{R}_{t+1,i}\right)^2}$$

Goodness-of-fit measure. By Kelly,Pruitt and Su (2019), Buchner and Kelly (2022), Chen et.al (2020):

$$R^2_{\text{total}} = 1 - \frac{\sum_{i,t}\left(r_{i,t+1} - z'_{i,t}\left(\hat{\Gamma}_\alpha + \hat{\Gamma}_\beta \hat{f}_{t+1}\right)\right)^2}{\sum_{i,t} r^2_{i,t+1}}.$$

Empirically:

$$R^2_{\text{pred}} = 1 - \frac{\sum_{i,t}\left(r_{i,t+1} - z'_{i,t}\left(\hat{\Gamma}_\alpha + \hat{\Gamma}_\beta \hat{\lambda}\right)\right)^2}{\sum_{i,t} r^2_{i,t+1}}$$

where $\hat{\lambda}$ denotes the unconditional time-series mean of the factors.

Overall evaluation

This result is summarized in Chen (2020)

Model	SR (Train)	SR (Valid)	SR (Test)
FF-3	0.27	−0.09	0.19
FF-5	0.46	0.37	0.22
IPCA	1.05	1.17	0.47
RtnFcst	0.63	0.41	0.27

Potential Extension

Recurrent Neural Network and dynamics

Here the basic intuition is to use the dynamic between the return on the underlying assets of last term as an input of the derivative's return of the current term.

- Apply models used in options pricing to other instruments.
  CDO or CDS index: reference Wang et.al (2009) "Pricing Tranches of a CDO and a CDS Index: Recent Advances and Future Research"
- Chen et.al didn't apply their RNN into option pricing model, but it of worth to do so.

The End