Deep Learning in Asset Pricing

Wu Zhu (Presenter)
University of Pennsylvania

December 4, 2019

Motivation

Deep learning is nothing but a powerful non-parametric (functional) approximation.

Incorporate these powerful functional approximation tools into finance and economics, but still remain basic economic intuitions

Two recent papers - will emphasize how to link economic sense with our toolbox

Background in Economics

Suppose we have N stocks or assets,

[1] Excess return vector: $R_t^e = (R_{1t}^e, ..., R_{Nt}^e)'$ with observations $t = 1, 2, ... T$.

[2] Economic constraint - agents try to make arbitrage using their information. As a result, there would be no arbitrage in equilibrium, i.e., there exists

$$E_t M_{t,t+1} R_{i,t+1}^e = 0 \Longrightarrow E_t R_{i,t+1}^e = -\frac{Cov_t(M_{t+1}, R_{t+1}^e)}{Var_t(M_{t+1})} \frac{Var_t(M_{t,t+1})}{E_t M_{t,t+1}}$$

with $E_t[.]$ the conditional expectation operator conditional on information up till period t.

$M_{t,t+1}$ is the SDF (Stochastic Discount Factor) measuring the relative prices of one dollar in different states

Background in Economics

Suppose agents optimize their portfolios, then there exist a time-varying vector w_t such that

$$M_{t,t+1} = 1 - w_t^T R^e_{t+1}$$

where w_t is measurable with respect to information set uptill t.

$$E[[1 - w_t^T R^e_{t+1}] R^e_{t+1} g(I_t, I_{i,t})] = 0$$

for $\forall$ measurable function g with respect to information set up till t

I_t are all available macro observable variables, and I_{it} are firms' individual characteristics uptill period t.

Deep Learning in Asset Pricing (Chen,Pelger, and Zhu, 2019)

Deep Learning in Asset Pricing (Chen, Pelger, and Zhu,2019)

Deep Learning in Asset Pricing (Chen,Pelger, and Zhu, 2019)

How do they estimate time varying w_t? Generative Adversarial Networks

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

Empirically, define loss function

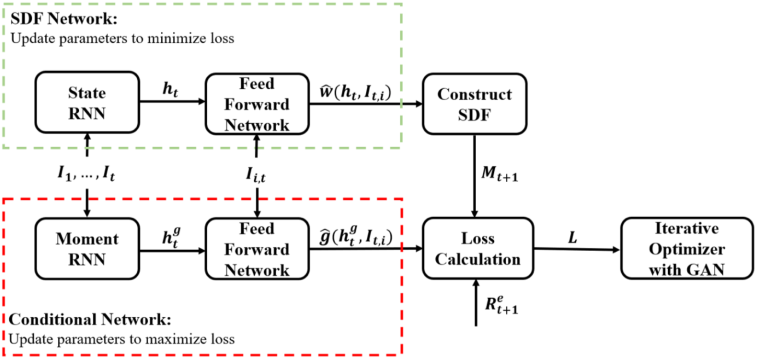
$$L(w, g, I_t, I_{t,i}) = \frac{1}{N} \sum_{i=1}^{N} \frac{T_i}{T} \| \frac{1}{T_i} \sum_{t_i} M_{t+1} R^e_{t+1,i} g(I_t, I_{t,i}) \|^2$$

Deep Learning in Asset Pricing (Chen,Pelger, and Zhu, 2019)

1. $w(I_t, I_{i,t})$ - approximate using RNN
2. $g(I_t, I_{i,t})$ - approximate using RNN
3. $w(I_t, I_{i,t})$ interact with $g(I_t, I_{i,t})$ in adversarial pattern - Given $w(I_t, I_{i,t})$, $g(I_t, I_{i,t})$ maximizes the loss, given $g(I_t, I_{i,t})$, $w(I_t, I_{i,t})$ minimizes the loss

Deep Learning in Asset Pricing (Chen,Pelger, and Zhu, 2019)

Figure 1. Model Architecture



Model architecture of GAN (Generative Adversarial Network) with RNN (Recurrent Neural Network) with LSTM cells.

Autoencoder Asset Pricing Models (Gu, Kelly, and Xiu, 2019)

Autoencoder Asset Pricing Models - 2019

Autoencoder Asset Pricing Models (Gu, Kelly, and Xiu, 2019)

Economic Backgrounds

Consider a standard factor model. Suppose the SDF takes factor structure, we have

$$r_{i,t+1} = \beta'_{i,t} f_{t+1} + \epsilon_{t+1}$$

where $\beta_{i,t}$ is the loading of stock i on factor f_{t+1}. Usually, both the loadings and factors are unobservable.

with economic constraint,

$$E_t[\epsilon_{i,t+1}] = 0, Cov_t(f_{t+1}, \epsilon_{i,t+1}) = 0$$

Autoencoder Asset Pricing Models (Gu, Kelly, and Xiu, 2019)

Loss function

$$L = \frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} (r_{it+1} - \beta'_{i,t} f_{t+1})^2$$

How to approximate for $\beta_{it} = \beta(z_{it})$ - use deep neural net

How to approximate for factors f_{t+1} - use autoencoder, that is, initially use r_{t+1}, and take a dimension reduction to recover lower dimensional factors.

Autoencoder Asset Pricing Models (Xiu et al 2019)

$$z_{i,t-1}^{(0)} = z_{i,t-1},$$

$$z_{i,t-1}^{(l)} = g\left(b^{(l-1)} + W^{(l-1)} z_{i,t-1}^{(l-1)}\right), \quad l = 1, \ldots, L_\beta,$$

$$\beta_{i,t-1} = b^{(L_\beta)} + W^{(L_\beta)} z_{i,t-1}^{(L_\beta)}.$$

Autoencoder Asset Pricing Models (Xiu et al 2019)

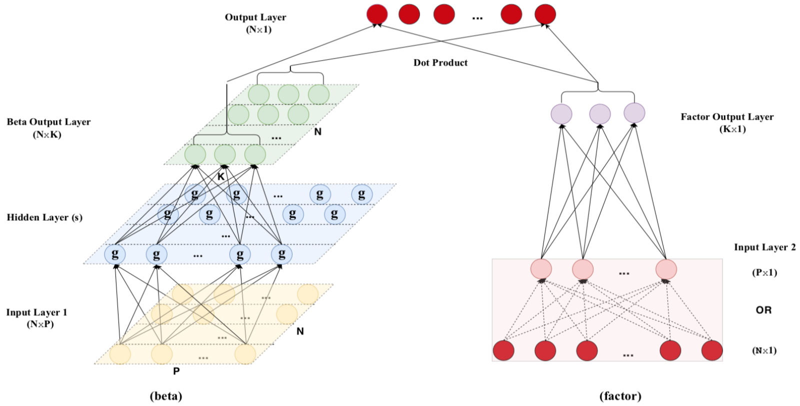
$$r_t^{(0)} = r_t,$$

$$r_t^{(l)} = \widetilde{g}\left(\widetilde{b}^{(l-1)} + \widetilde{W}^{(l-1)} r_t^{(l-1)}\right), \quad l = 1, \dots, L_f,$$

$$f_t = \widetilde{b}^{(L_f)} + \widetilde{W}^{(L_f)} r_t^{(L_f)}.$$

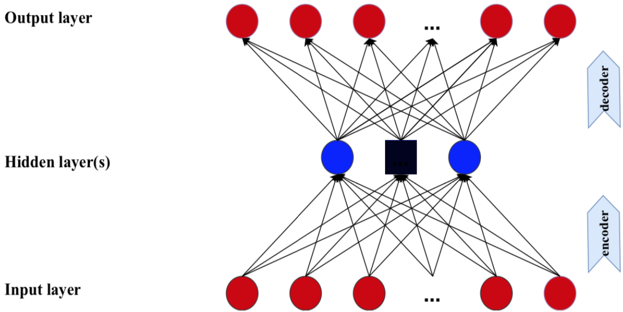
Autoencoder Asset Pricing Models (Xiu et al 2019)

Figure 2: Conditional Autoencoder Model



Autoencoder Asset Pricing Models (Xiu et al 2019)

Figure 1: Standard Autoencoder Model



Note: This figure describes a standard autoencoder with one hidden layer. The output and input layers are identical, while the hidden layer is a low dimensional compression of inputs variables into latent factors, which can be expressed as weighted linear combinations of input variables.