Deep Learning in Asset Pricing

Luyang Chen [1] Markus Pelger [1] Jason Zhu [1]

[1]Stanford University

November 17th 2018

Western Mathematical Finance Conference 2018

Hype: Machine Learning in Investment

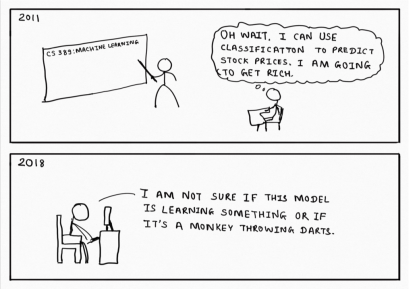
The U.S. Stock Market Belongs to Bots

Bloomberg, July 15, 2017

Same reporter 3 weeks later

The Quant Fund Robot Takeover Has Been Postponed

Bloomberg, Aug 9, 2017



- Efficient markets: Asset returns dominated by unforecastable news

$\Rightarrow$ Financial return data has very low signal-to noise ratio

$\Rightarrow$ This paper: Including financial constraints (no-arbitrage) in learning algorithm significantly improves signal

Motivation: Asset Pricing

The Challenge of Asset Pricing

- One of the most important questions in finance:

  Why are asset prices different for different assets?

- **No-Arbitrage Pricing Theory**: **Stochastic discount factor SDF** (also called pricing kernel or equivalent martingale measure) explains differences in risk and asset prices
- Fundamental question: What is the SDF?
- Challenges
  - SDF should depend on all available economic information: Very large set of variables
  - Functional form of SDF unknown and likely complex
  - SDF needs to capture time-variation in economic conditions
  - Risk premium in stock returns has a low signal-to-noise ratio

This paper

Goals of this paper:

General non-linear asset pricing model and optimal portfolio design

$\Rightarrow$ Deep-neural networks applied to all U.S. equity data and large sets of macroeconomic and firm-specific information.

Why is it important?

1. Stochastic discount factor (SDF) generates tradeable portfolio with highest risk-adjusted return
(Sharpe-ratio=expected excess return/standard deviation)
2. Arbitrage opportunities
    - Find underpriced assets and earn "alpha"
3. Risk management
    - Understand which information and how it drives the SDF
    - Manage risk exposure of financial assets

Contribution of this paper

Contribution

- This Paper: Estimate the SDF with deep neural networks
- Crucial innovation: Include no-arbitrage condition in the neural network algorithm and combine four neural networks in a novel way
- Key elements of estimator:
  1. Non-linearity: Feed-forward network captures non-linearities
  2. Time-variation: Recurrent (LSTM) network finds a small set of economic state processes
  3. Pricing all assets: Generative adversarial network identifies the states and portfolios with most unexplained pricing information
  4. Dimension reduction: Regularization through no-arbitrage condition
  5. Signal-to-noise ratio: No-arbitrage conditions increase the signal to noise-ratio

$\Rightarrow$ General model that includes all existing models as a special case

Contribution of this paper

Empirical Contributions

- Empirically outperforms all benchmark models.
- Optimal portfolio has out-of-sample annual Sharpe ratio of 2.15.
- Non-linearities and interaction between firm information matters.
- Most relevant firm characteristics are price trends, profitability, and capital structure variables.
- Shallow learning outperforms deep-learning.

Literature (partial list)

- Deep-learning for predicting asset prices
  - Gu, Kelly and Xiu (2018)
  - Feng, Polson and Xu (2018)
  - Messmer (2017)

  $\Rightarrow$ Predicting future asset returns with feed forward network
- Linear or kernel methods for asset pricing of large data sets
  - Lettau and Pelger (2018): Risk-premium PCA
  - Feng, Giglio and Xu (2017): Risk-premium lasso
  - Freyberger, Neuhierl and Weber (2017): Group lasso
  - Kelly, Pruitt and Su (2018): Instrumented PCA

The Model

No-arbitrage pricing

- $R^e_{i,t+1}$ = excess return (return minus risk-free rate) at time $t+1$ for asset $i = 1, ..., N$
- Fundamental no-arbitrage condition:
  for all $t = 1, ..., T$ and $i = 1, ..., N$

$$\mathbb{E}_t[M_{t+1}R^e_{i,t+1}] = 0$$

  - $\mathbb{E}_t[.]$ expected value conditioned on information set at time t
  - M_{t+1} stochastic discount factor SDF at time $t+1$.
- Conditional moments imply infinitely many unconditional moments

$$\mathbb{E}[M_{t+1}R^e_{t+1,i}I_t] = 0$$

for any $\mathcal{F}_t$-measurable variable I_t

The Model

No-arbitrage pricing

- Without loss of generality SDF is projection on the return space

$$M_{t+1} = 1 + \sum_{i=1}^{N} w_{i,t} R^e_{i,t+1}$$

$\Rightarrow$ Optimal portfolio $-\sum_{i=1}^{N} w_{i,t} R^e_{i,t+1}$ has highest conditional Sharpe-ratio

- Portfolio weights $w_{i,t}$ are a general function of macro-economic information I_t and firm-specific characteristics $I_{i,t}$:

$$w_{i,t} = w(I_t, I_{i,t}),$$

$\Rightarrow$ Need non-linear estimator with many explanatory variables!

$\Rightarrow$ Use a feed forward network to estimate $w_{i,t}$

Loss Function

Objective Function for Estimation

- Estimate SDF portfolio weights $w(.)$ to minimize the no-arbitrage moment conditions
- For a set of conditioning variables $\hat{I}_t$ the loss function is

$$L(\hat{I}_t) = \frac{1}{N}\sum_{i=1}^{N}\frac{T_i}{T}\left(\frac{1}{T_i}\sum_{t=1}^{T_i} M_{t+1}R^e_{i,t+1}\hat{I}_t\right)^2.$$

- Allows unbalanced panel.
- How can we choose the conditioning variables $\hat{I}_t = f(I_t, I_{i,t})$ as general functions of the macroeconomic and firm-specific information?

$\Rightarrow$ Generative Adversarial Network (GAN) chooses $\hat{I}_t$!

Generative Adversarial Network (GAN)

Determining Moment Conditions

- Two networks play zero-sum game:
  1. one network creates the SDF M_{t+1}
  2. other network creates the conditioning variables $\hat{I}_t$
- Iteratively update the two networks:
  1. for a given $\hat{I}_t$ the SDF network minimizes the loss
  2. for a given SDF the conditional networks finds $\hat{I}_t$ with the largest loss (most mispricing)

$\Rightarrow$ Intuition: find the economic states and assets with the most pricing information

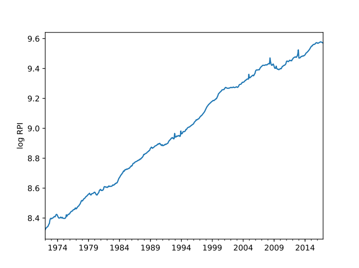
Recurrent Neural Network (RNN)

Transforming Macroeconomic Time-Series

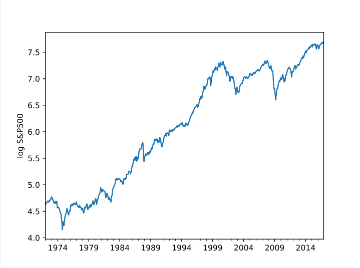
- **Problems** with economic time-series data
  - Time-series data is often non-stationary $\Rightarrow$ transformation necessary
  - Asset prices depend on economic states $\Rightarrow$ simple differencing of non-stationary data not sufficient
- **Solution**: Recurrent Neural Network (RNN) with Long-Short-Term Memory (LSTM) cells
- Transform all macroeconomic time-series into a low dimensional vector of stationary state variables

Example: Non-stationary Macroeconomic Variables

Macroeconomic Variables



(a) Log RPI



(b) Log S&P500

Macroeconomic state processes

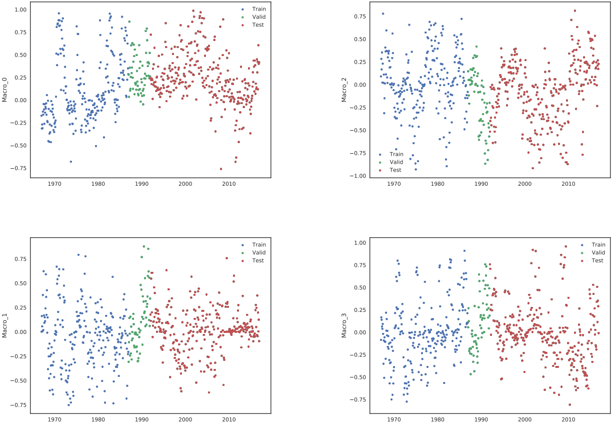
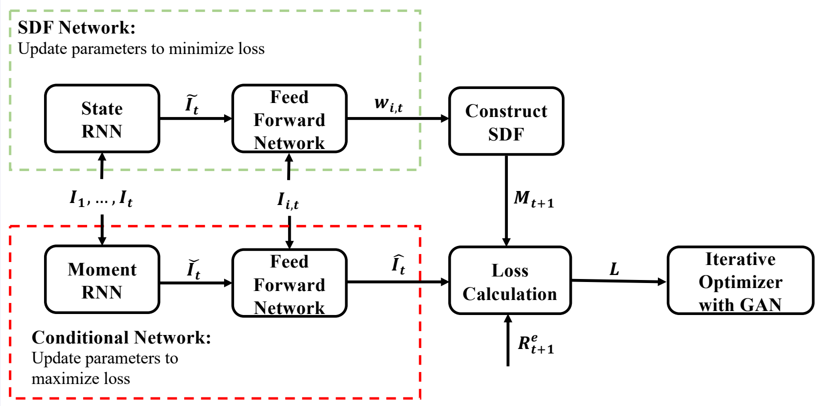


Figure: **Macroeconomic state processes (LSTM Outputs)** based on 178 macroeconomic time-series.

Neural Networks

Model Architecture



Data

Data

- 50 years of monthly observations: 01/1967 - 12/2016.
- Monthly stock returns for all U.S. securities from CRSP (around 31,000 stocks)
  Use only stocks with with all firm characteristics (around 10,000 stocks)
- 46 firm-specific characteristics for each stock and every month (usual suspects) $\Rightarrow I_{i,t}$
  normalized to cross-sectional quantiles
- 178 macroeconomic variables
  (124 from FRED, 46 cross-sectional median time-series for characteristics, 8 from Goyal-Welch) $\Rightarrow I_t$
- T-bill rates from Kenneth-French website
- Training/validation/test split is 20y/5y/25y

Benchmark models

Benchmark models

1. Linear factor models (CAPM, Fama-French 5 factors)
2. Instrumented PCA (Kelly et al. (2018): estimate SDF as linear function of characteristics: $w_{i,t} = \theta^\top I_{i,t}$
3. Deep learning return forecasting (Gu et al. (2018)):
   - Predict conditional expected returns $\mathbb{E}_t[R_{i,t+1}]$
   - Empirical loss function for prediction

$$\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(R_{i,t+1} - g(I_t, I_{i,t+1})\right)^2$$

   - Use only simple feedforward network for forecasting
   - Optimal portfolio: Long-short portfolio based on deciles of highest and lowest predicted returns

Results - Sharpe Ratio

Sharpe Ratios of Benchmark Models

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

Results - Sharpe Ratio

Table: Performances of our approach sorted by validation Sharpe ratio

SR (Train)	SR (Valid)	SR (Test)	SMV	CSMV	HL	CHL	HU	CHU
1.80	1.01	0.62	4	32	4	0	64	4
1.30	1.01	0.54	4	32	2	1	64	8
2.13	0.97	0.61	4	32	4	0	64	16
2.49	0.96	0.51	4	32	4	0	64	16

$\Rightarrow$ Optimal model: 4 moments, 4 macro states, 4 layers, 64 hidden units

Optimal Portfolio Performance

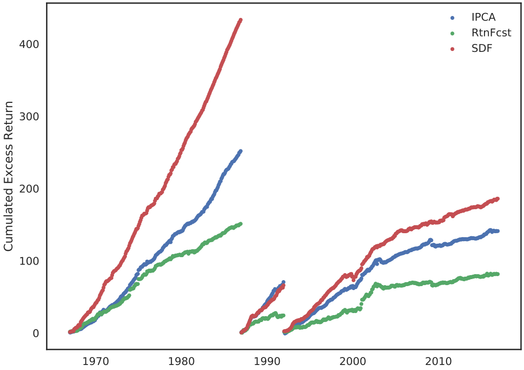


Figure: **Cumulated Normalized SDF Portfolio.**

Results - Sharpe Ratio for Forecasting Approach

Performances with Return Forecast Approach

Macro	Neurons	SR (Train)	SR (Valid)	SR (Test)
Y	[32, 16, 8]	0.16	0.24	-0.00
Y	[128, 128]	1.30	0.10	0.04
N	[32, 16, 8]	0.63	0.41	0.27
N	[128, 128]	0.67	0.51	0.37

IPCA: Number of Factors

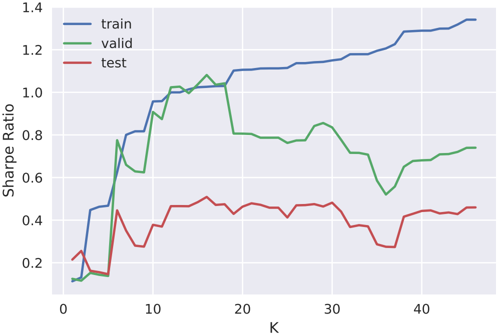


Figure: Sharpe ratio as a function of the number of factors for IPCA

Results - Sharpe Ratio

Performance of Benchmark Models

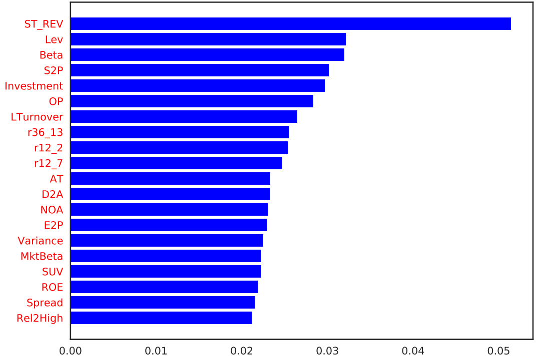
Table: **SDF Portfolio vs. Fama-French 5 Factors**

	Mkt-RF	SMB	HML	RMW	CMA	intercept
coefficient	0.06	0.00	0.01	0.17	0.05	0.47
correlation	0.02	-0.14	0.25	0.33	0.16	-

$\Rightarrow$ Conventional factors do no span SDF

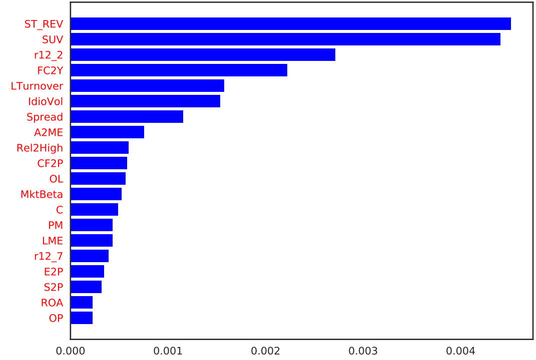
Results - Variable Importance

Variables Ranked by Average Absolute Gradient (Top 20) for SDF network



Results - Variable Importance

Variables Ranked by Reduction in R^2 for RtnFcst (Top 20)



Size Effect

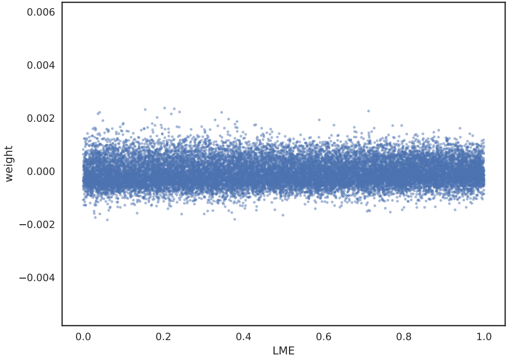
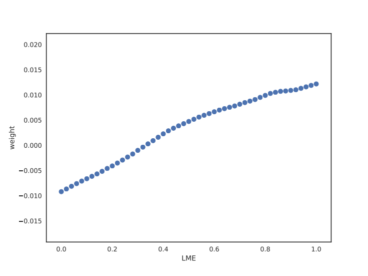


Figure: **SDF weight and market capitalization in test data**

Results - SDF Weights

Relationship between Weights and Characteristics



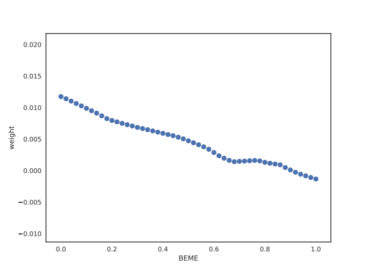


Figure: Weight as a function of size (LME) and book-to-market (BEME).

$\Rightarrow$ Size and value have close to linear effect

Results - SDF Weights

Relationship between Weights and Characteristics

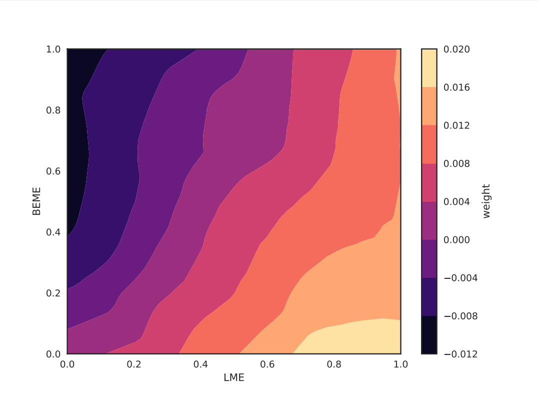


Figure: Weight as a function of size (LME) and book-to-market (BEME).

$\Rightarrow$ Size and value have non-linear interaction!

Results - SDF Weights

Relationship between Weights and Characteristics

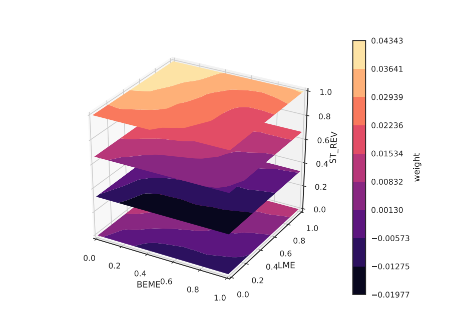
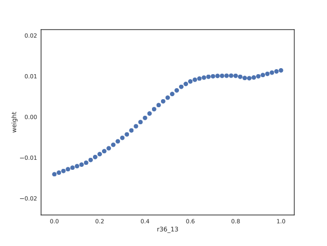


Figure: Weight as a function of size, book-to-market and ST-reversal.

⇒ Complex interaction between multiple variables!

Results - SDF Weights

Relationship between Weights and Characteristics



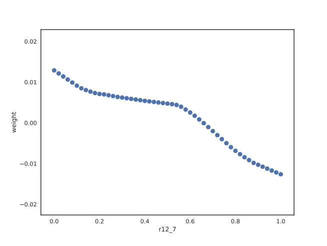


Figure: Weight as a function of reversal (r36-13) or momentum (r12-7).

$\Rightarrow$ Non-linear effect!

Results - Weights

Relationship between Weights and Characteristics

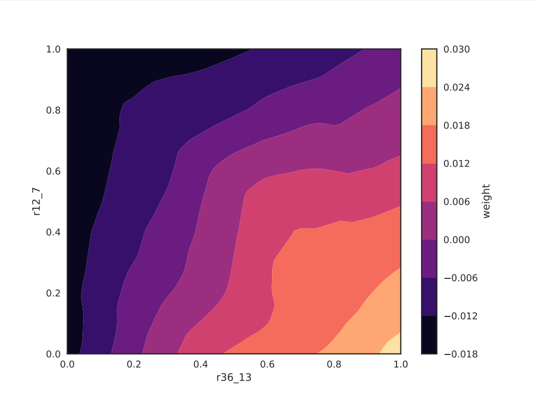


Figure: Weight as a function of momentum (r12-7) and reversal (r36-13).

$\Rightarrow$ Complex interaction!

Results - Weights

Relationship between Weights and Characteristics

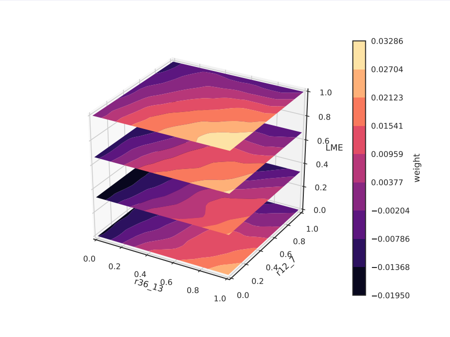


Figure: Weight as a function of momentum (r12-7), reversal (r36-13) and size (LME).

⇒ Complex interaction between multiple variables!

Simulation

Setup

- Consider a single factor model

$$R_{i,t+1} = \beta_{i,t} F_{t+1} + \varepsilon_{i,t+1}$$

- The only factor is sampled from $\mathcal{N}(\mu_F, \sigma_F^2)$.
- The loadings are $\beta_{i,t} = C_{i,t}$ with $C_{i,t}$ i.i.d $\mathcal{N}(0,1)$.
- The residuals are i.i.d $\mathcal{N}(0,1)$.
- $N = 500$ and $T = 600$. Define training/validation/test $= 250,\ 100,\ 250$.
- Consider $\sigma_F^2 \in \{0.01, 0.05, 0.1\}$.
- Sharpe Ratio of the factor $SR = \mu_F/\sigma_F = 0.3$ or $SR = 1$.

Simulation Results: Intuition

Intuition: Better noise diversification with our approach

- Simple return prediction

$$\frac{1}{TN}\sum_{i=1}^{N}\sum_{t=1}^{N}(R^e_{i,t+1}-f(I_t))^2$$

- SDF estimator

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}R^e_{i,t+1}M_{t+1}g(C_{i,t})\right)^2$$

$\Rightarrow$ SDF estimator averages out the noise over the time-series

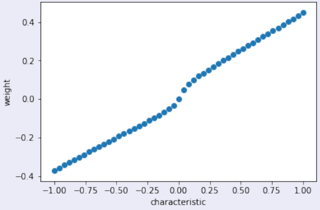
Simulation Results

Sharpe Ratio on Test Dataset

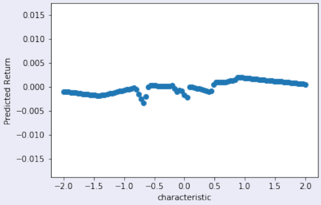
σ_F^2	RtnFcst	SDF estimator
SR=0.3		
0.01	0.03	0.22
0.05	0.20	0.33
0.10	0.35	0.35
SR=1		
0.01	0.63	0.96
0.05	0.92	0.97
0.10	1.03	1.03

Simulation Results

Estimated loadings and SDF weights



(a) Our SDF estimator



(b) Return forecasting

$\Rightarrow$ Our approach detects SDF and loading structure.

$\Rightarrow$ Simple forecasting approach fails.

Conclusion

Methodology

- Novel combination of deep-neural networks to estimate the pricing kernel
- Key innovation: Use no-arbitrage condition as criterion function
- Time-variation explained by macroeconomic states and firm characteristics
- General asset pricing model that includes all other models as special cases

Empirical Results

- Outperforms benchmark models
- Non-linearities and interactions are important

Number of Stocks

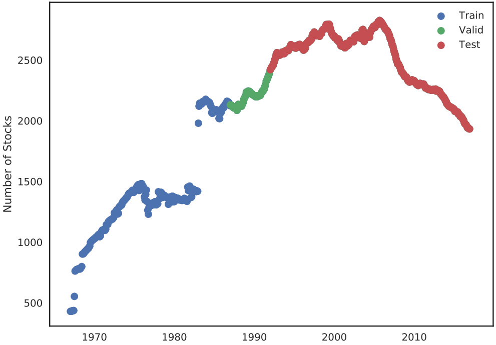


Figure: **Number of Stocks**

Results

Performance of Benchmark Models

Table: **Max 1 Month Loss & Max Drawdown**

	Max 1 Month Loss	Max Drawdown
IPCA	-6.711	5
RtnFcst (Equally Weighted)	-4.005	4
RtnFcst (Value Weighted)	-3.997	4
SDF	-5.277	4

$\Rightarrow$ Optimal portfolio has desirable properties

IPCA: Number of Factors

Table: Performance with IPCA

Number of Factors	SR (Train)	SR (Valid)	SR (Test)
1	0.113	0.117	0.206
2	0.121	0.100	0.226
3	0.483	0.205	0.184
4	0.498	0.200	0.176
5	0.507	0.196	0.164
6	0.685	0.843	0.485
12	1.049	1.174	0.470

Results - Sharpe Ratio for Forecasting Approach

Performances with Return Forecast Approach

Macro	Neurons	Value Weighted	SR (Train)	SR (Valid)	SR (Test)
Y	[32, 16, 8]	N	0.21	0.09	0.03
		Y	0.16	0.24	-0.00
Y	[128, 128]	N	1.51	0.20	0.15
		Y	1.30	0.10	0.04
N	[32, 16, 8]	N	1.13	1.34	0.68
		Y	0.63	0.41	0.27
N	[128, 128]	N	1.22	1.25	0.67
		Y	0.67	0.51	0.37

Optimal Portfolio Performance

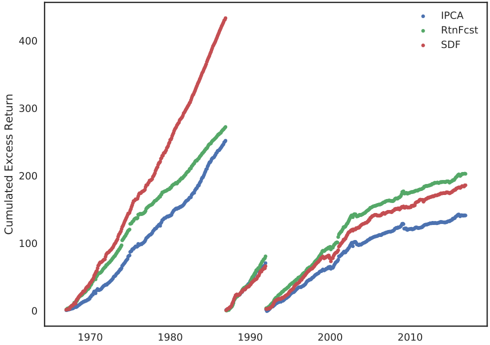


Figure: **Cumulated Normalized SDF Portfolio.** Use equal weighting for return forecast approach.

Optimal Portfolio Performance

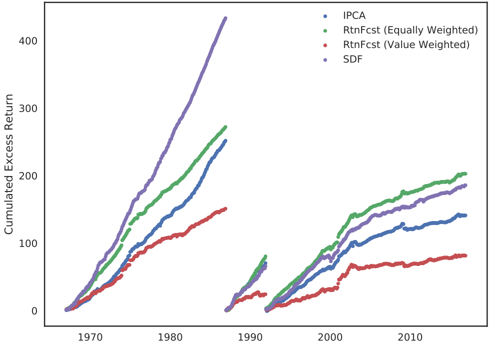


Figure: **Cumulated Normalized SDF Portfolio.** Include both value weighting and equal weighting for return forecast approach.

Hyper-Parameter Search

Search Space

- *CV* number of conditional variables: 4, 8, 16, 32
- *SMV* number of macroeconomic state variables: 4, 8, 16, 32
- *HL* number of fully-connected layers: 2, 3, 4
- *HU* number of hidden units in fully-connected layers: 32, 64, 128
- *D* dropout rate (keep probability): 0.9, 0.95
- *LR* learning rate: 0.001, 0.0005, 0.0002, 0.0001

- Choose best configuration of all possible combinations (1152) of hyper-parameters on validation set.
- Use ReLU activation function $ReLU(x)_i = \max(x_i, 0)$.

Models for Comparison

Loss Functions for Different Models

1. Simple return prediction

$$\frac{1}{TN}\sum_{i=1}^{N}\sum_{t=1}^{N}(R^e_{i,t+1} - f(I_t))^2$$

2. Unconditional moment

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}R^e_{i,t+1}M_{t+1}\right)^2$$

3. GAN conditioned on the firm characteristics (benchmark approach)

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}R^e_{i,t+1}M_{t+1}g(C_{i,t})\right)^2$$

4. GAN network based on moment portfolios

$$\left(\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}R^e_{i,t+1}g(C_{i,t})\right)M_{t+1}\right)^2$$

5. Price decile portfolios

$$\frac{1}{10}\sum_{i=1}^{10}\left(\frac{1}{T}\sum_{t=1}^{T}R^e_{i,t+1}M_{t+1}\right)^2$$

Simulation Results

Sharpe Ratio on Test Dataset (SR=1)

σ_F^2	RtnFcst	UNC	GAN	PortGan	Decile
0.01	0.627	0.98	0.964	0.978	0.983
0.05	0.924	0.957	0.969	0.957	0.953
0.1	1.031	1.023	1.033	1.003	1.039

Simulation Results (Continue)

Sharpe Ratio on Test Dataset (SR=0.3)

σ_F^2	RtnFcst	UNC	GAN	PortGan	Decile
0.01	0.03	0.22	0.221	0.222	0.215
0.05	0.199	0.33	0.331	0.319	0.328
0.1	0.353	0.368	0.353	0.366	0.36

Economic Significance of Variables

Sensitivity

- We define the sensitivity of a particular variable as the magnitude of the derivative of weight w with respect to this variable (averaged over the data):

$$\text{Sensitivity}(x_j) = \frac{1}{C}\sum_{i=1}^{N}\sum_{t}\left|\frac{\partial w(\tilde{I}_t, I_{i,t})}{\partial x_j}\right| \tag{1}$$

  with C a normalization constant. The analysis is performed with the feed-forward network and we only consider the sensitivity of firm characteristics and state macro variables.
- A sensitivity of value z for a given variable means that the weight w will approximately change (in magnitude) by $z\Delta$ if that variable is changed by a small amount Δ.

Interactions between Variables

Significance of Interactions

We might also want to understand how the output simultaneously depends upon multiple variables. We can measure the economic significance of the interaction between variables x_i and x_j by the derivative:

$$\text{Sensitivity}(x_i, x_j) = \frac{1}{C} \sum_{i=1}^{N} \sum_{t} \left| \frac{\partial^2 w(\tilde{I}_t, I_{i,t})}{\partial x_i \partial x_j} \right| \tag{2}$$

This derivative can be generalized to measure higher-order interactions.

Finite Difference Schemes

First-Order and Second-Order Finite Difference Schemes

Suppose we have some multivariate function f. Without actually measuring gradients, we can approximate them with finite difference methods

$$\frac{\partial f}{\partial x_j} \approx \frac{f(x_j + \Delta) - f(x_j)}{\Delta} \tag{3}$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{f(x_i + \Delta, x_j + \Delta) - f(x_i + \Delta, x_j) - f(x_i, x_j + \Delta) + f(x_i, x_j)}{\Delta^2} \tag{4}$$

Leave-One-Out Methods

Leave-One-Out Analysis

Leave-one-out analysis is another method to explain the explanatory power of the variables. For each variable, the variable is removed from the model and the Sharpe Ratio is evaluated on the test dataset in the absence of this covariate. Specifically, the leave-one-out variable is set to 0 for all data samples in the test dataset and the Sharpe Ratio is calculated using the reduced variable vector. Then, the variable is replaced in the model, and a leave-one-out test is performed on a new variable.

SDF Network

Summary

1. **GOAL:** Generate portfolio weight $w_{i,t}$.
2. **Input:** Macro-economic information history $\{I_1, \ldots, I_t\}$ and firm characteristics $I_{i,t}$.
3. **Output:** Weight $w_{i,t}$.
4. **Architecture:**
    - The history of macro variables is transformed via a Recurrent Neural Network (RNN). The transformed macro variables extract predictive information and summarize macro history.
    - The transformed macro variables and firm characteristics are passed through a Feed Forward Network to generate weights.

Feed Forward Network

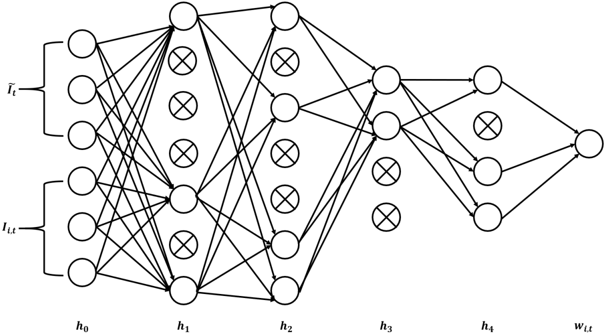


Figure: Feed Forward Network with Dropout

Feed Forward Network

Network Structure

- The input layer accepts the raw predictors (or features).

$$h_0 = x_{i,t} = [I_{i,t}, \tilde{I}_t] \tag{5}$$

- Each hidden layer takes the output from the previous layer and transforms it into an output as

$$h_k = f(h_{k-1}W_k + b_k) \qquad k = 1, \ldots, K \tag{6}$$

  In our implementation, we use ReLU activation function.

$$ReLU(x)_i = \max(x_i, 0) \tag{7}$$

- The output layer is simply a linear transformation of the output from the last hidden layer to a scaler

$$w_{i,t} = h_K W_{K+1} + b_{K+1} \tag{8}$$

Feed Forward Network

Model Complexity

- Number of hidden layers: K.
- Let's denote p_k to be the number of neurons (or hidden units) in the layer k. The parameters in the layer k are

$$W_k \in \mathbb{R}^{p_{k-1} \times p_k} \quad \text{and} \quad b_k \in \mathbb{R}^{p_k} \tag{9}$$

  with $p_0 = \dim(I_{i,t}) + \dim(\tilde{I}_t)$ and $p_{K+1} = 1$.
- Number of parameters: $\sum_{k=1}^{K+1}(p_{k-1} + 1)p_k$.
  e.g. A 4-hidden-layer network with hidden units $[128, 128, 64, 64]$ has 39105 parameters.

State RNN

Two Reasons for RNN

Instead of directly passing macro variables I_t as features to the feed forward network, we apply a nonlinear transformation to them with an RNN.

- Many macro variables themselves are not stationary and have trends. Necessary transformations of I_t are essential in generating a statble model.
- Using RNN allows us to encode all historical information of the macro economy. Intuitively, RNN summarizes all historical macro information into a low dimensional vector of state variables in a data-driven way.

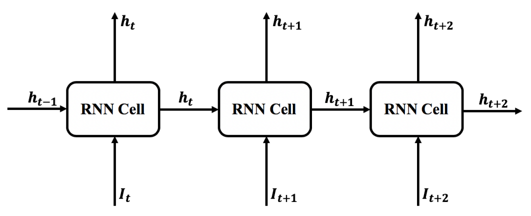
Transform Macro Variables via RNN

Properties of RNN

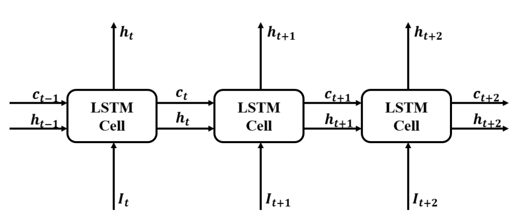
- For any $\mathcal{F}_t$-measurable sequence I_t, the output sequence $\tilde{I}_t$ is again $\mathcal{F}_t$-measurable. The transformation creates no look-ahead bias.
- $\tilde{I}_t$ contains all the macro information in the past, while I_t only uses current information.
- RNN helps create a stationary macro inputs for the feed forward network.

Recurrent Network

Recurrent Network with RNN Cell

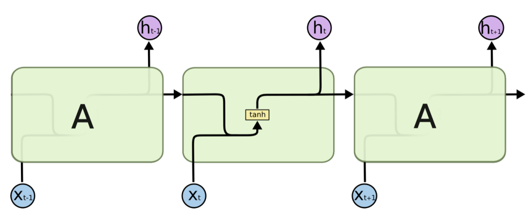


Recurrent Network with LSTM Cell



Recurrent Network

RNN Cell



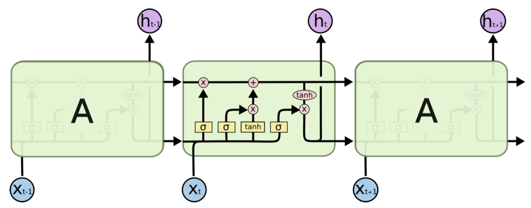
RNN Cell Structure

A vanilla RNN model takes the current input variable $x_t = I_t$ and the previous hidden state h_{t-1} and performs a nonlinear transformation to get the current hidden state h_t

$$h_t = f(h_{t-1}W_h + x_t W_x). \tag{10}$$

Recurrent Network

LSTM Cell



Recurrent Network

LSTM Cell Structure

An LSTM model creates a new memory cell $\tilde{c}_t$ with current input x_t and previous hidden state h_{t-1}:

$$\tilde{c}_t = \tanh(h_{t-1}W_h^{(c)} + x_t W_x^{(c)}). \tag{11}$$

An input gate i_t and a forget gate f_t are created to control the final memory cell:

$$i_t = \sigma(h_{t-1}W_h^{(i)} + x_t W_x^{(i)}) \tag{12}$$

$$f_t = \sigma(h_{t-1}W_h^{(f)} + x_t W_x^{(f)}) \tag{13}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t. \tag{14}$$

Finally, an output gate o_t is used to control the amount of information stored in the hidden state:

$$o_t = \sigma(h_{t-1}W_h^{(o)} + x_t W_x^{(o)}) \tag{15}$$

$$h_t = o_t \circ \tanh(c_t). \tag{16}$$

Motivation

Our Approach

We work with the fundamental pricing equation to obtain estimates of pricing kernel (or Stochastic Discount Factor or SDF).

1. Few additional assumptions.
2. Nonlinear in underlying predictors.
3. Time-varying portfolio weights.
4. Theoretically most profitable portfolio.

Motivation

Our Approach

We work with the fundamental pricing equation to obtain estimates of pricing kernel (or Stochastic Discount Factor or SDF).

1. **Few additional assumptions.**

$$\text{APT:} \quad \mathbb{E}_t[M_{t+1} R^e_{i,t+1}] = 0 \tag{17}$$

$$\text{Projection:} \quad M_{t+1} = 1 + \sum_{i=1}^{N} w_{i,t} R^e_{i,t+1} \tag{18}$$

2. Nonlinear in underlying predictors.
3. Time-varying portfolio weights.
4. Theoretically most profitable portfolio.

Motivation

Our Approach

We work with the fundamental pricing equation to obtain estimates of pricing kernel (or Stochastic Discount Factor or SDF).

1. Few additional assumptions.
2. **Nonlinear in underlying predictors.**
   We model portfolio weights $w_{i,t}$ as some general function of macro-economic information I_t and firm-specific characteristics $I_{i,t}$:
   $$w_{i,t} = w(I_t, I_{i,t}; \theta), \tag{19}$$
   which can be highly nonlinear in input variables and high dimensional parameter θ. (Ans: Neural Networks)
3. Time-varying portfolio weights.
4. Theoretically most profitable portfolio.

Motivation

Our Approach

We work with the fundamental pricing equation to obtain estimates of pricing kernel (or Stochastic Discount Factor or SDF).

1. Few additional assumptions.
2. Nonlinear in underlying predictors.
3. **Time-varying portfolio weights.**
   We construct infinite number of moment conditions from pricing formula (17). For any $\mathcal{F}_t$-measurable variable $\hat{I}_t$,

$$\mathbb{E}[M_{t+1} R^e_{i,t+1} \hat{I}_t] = 0. \quad (20)$$

4. Theoretically most profitable portfolio.

Motivation

Our Approach

We work with the fundamental pricing equation to obtain estimates of pricing kernel (or Stochastic Discount Factor or SDF).

1. Few additional assumptions.
2. Nonlinear in underlying predictors.
3. Time-varying portfolio weights.
4. **Theoretically most profitable portfolio.** With (17) and (18), $w_{i,t}$ defines a portfolio with the highest Sharpe Ratio.

Comparison with Gu et al. (2018)

Target - Difference

- **Gu et al. (2018):** Given current available information, what is the best guess of asset's future return $\mathbb{E}[r_{i,t+1}|\mathcal{F}_t]$?
- **[Chen et al., 2018]:** Given current available information, what is the best guess of SDF (projected on asset span) that prices all the assets?

Target - Connection

- The (conditional) expectation and Sharpe Ratio of SDF is related to the estimation of $\mathbb{E}[r_{i,t+1}|\mathcal{F}_t]$ and $\mathbb{E}[r_{i,t+1}r_{j,t+1}|\mathcal{F}_t]$.

$$\mathbb{E}[SDF_{t+1}|\mathcal{F}_t] = 1 + w_t^\top \mathbb{E}[R^e_{t+1}|\mathcal{F}_t] \tag{21}$$

$$\mathbb{E}[SDF^2_{t+1}|\mathcal{F}_t] = 1 + 2w_t^\top \mathbb{E}[R^e_{t+1}|\mathcal{F}_t] + w_t^\top \mathbb{E}[R^e_{t+1}R^{e\top}_{t+1}|\mathcal{F}_t]w_t \tag{22}$$

Comparison with Gu et al. (2018)

Objective Function (Loss Function) - Difference

- **Gu et al. (2018):** For any $\mathcal{F}_t$-measurable variable $g(z_{i,t};\theta)$, $\mathbb{E}[r_{i,t+1}|\mathcal{F}_t]$ is the one such that $\mathbb{E}[(r_{i,t+1} - g(z_{i,t};\theta))^2]$ is minimized. The empirical loss function reads as

$$\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(r_{i,t+1} - g(z_{i,t};\theta))^2 \tag{23}$$

- **[Chen et al., 2018]:** SDF_{t+1} is a process such that for any asset and any conditional variable $\hat{I}_t$, $\mathbb{E}[SDF_{t+1}R^e_{i,t+1}\hat{I}_t] = 0$. There are infinite number of moment conditions and unconditional expectation $\mathbb{E}[SDF_{t+1}R^e_{i,t+1}] = 0$ is only one of them. Therefore, the empirical loss function based on unconditional expectation

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}SDF_{t+1}R^e_{i,t+1}\right)^2 \tag{24}$$

might not be enough.

Comparison with Gu et al. (2018)

Model Architecture - Difference

- **Gu et al. (2018):** Concatenate macro variables and firm characteristics as inputs of a fully-connected network to model $\mathbb{E}[r_{i,t+1}|\mathcal{F}_t]$.
- **[Chen et al., 2018]:** Encode macro variables to state macro variables with an RNN, which are then concatenated with firm characteristics as inputs of a fully-connected network to model w_t.

Comparison with Gu et al. (2018)

Optimal Portfolio - Difference

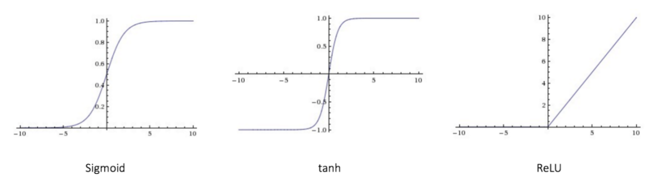
- **Gu et al. (2018):** The stocks are sorted into deciles based on model's forecasts. A zero-net-investment portfolio is constructed that buys the highest expected return stocks (decile 10) and sells the lowest (decile 1) with equal weights.
- **[Chen et al., 2018]:** The portfolio weights are given by the model. The optimal portfolio $-w_t^\top R^e_{t+1}$ is obtained by shorting SDF portfolio.

Model Architecture

Activation Function

The function f is nonlinear and is called the activation function. Common activation functions are Sigmoid, tanh and ReLU.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \tanh(x) = 2\sigma(2x) - 1 \quad \mathrm{ReLU}(x) = \max(0, x) \tag{25}$$



Gu, S., Kelly, B. T., and Xiu, D. (2018). Empirical asset pricing via machine learning.