

Deep Learning, Clustering, and Decision Process Approaches for Modeling Time Series Data

Shan Zhong
University of South Carolina
Advisor: David Hitchcock

May 19, 2022

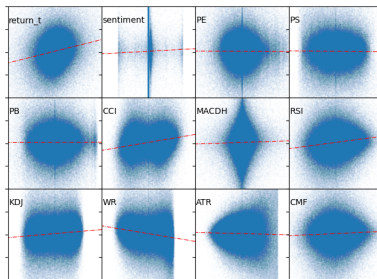
Overview

- ▶ Project 1: S&P 500 Stock Price Prediction Using Technical, Fundamental and Text Data
 - ▶ Ensemble Learning, Text Embedding, LSTM Deep Learning.
- ▶ Project 2: Functional Clustering of Fictional Narratives Using Vonnegut Curves
 - ▶ Bayesian MCMC, Time Warping, Curve Clustering.
- ▶ Project 3: Multi-agent value and policy learning to build prediction intervals for massive correlated stock data and construct portfolios under risk-measure
 - ▶ Hidden Markov Model, Multi-agent Order Dispatchment.

Project 1: Data and Preprocessing

- ▶ Left: Summary for the variables that we used in the model besides past return, from January 1, 2000, to December 31, 2019, for 518 S&P stocks.
- ▶ Right: Each predictor variable at t with the dependent variable return at $t + 1$, where these variables were transformed to standard Gaussian and missing values filled by 0.

Variable Name	Quick Introduction	Window period
CCI	The Commodity Channel Index can help to identify price reversals, price extremes and trend strength.	20 days
MACDH	The Moving Average Convergence Divergence Histogram is the difference between MACD and the MACD signals.	12, 26, and 9 days
RSI	The Relative Strength Index, which measures the average of recent upward movement versus the upward and downward movements combined, in a percentage form.	14 days
KDJ	The Stochastic Oscillator measures where the close is relative to the low and high.	14, 3, and 3 days
WR	The Williams %R index is another index for buy signals by measuring the current price in relation to the past N periods.	14 days
ATR	The Average True Range provides an indicator for the volatility of price. We converted ATR into percentages.	14 days
CMF	The Chaikin Money Flow provides an indicator related to the trading volume.	20 days
PE	The price to earning ratio is calculated as the share price divided by the earnings per share.	7 days
PB	The price to book ratio is calculated as the share price divided by the book value per share.	7 days
PS	The price to sales ratio is calculated as the share price divided by the revenues per share.	7 days
sentiment	The sentiment scores measure the positive or negative impact of news.	7 days



Project 1: Text Data processing

- ▶ Left: An example of NYT text data after selecting the variables for display and modifying the structure. We used a BERT model to infer article sentiment.
- ▶ Right: Performance of the pretrained BERT model that fine tuned on Financial PhraseBank data where 66% of the 16 people agreed in their judgments.

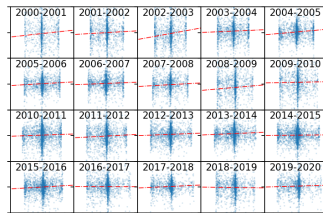
Snippet	"Chris Cox, who quit Facebook last year after differences with the company's chief executive, Mark Zuckerberg, is returning as chief product officer."
Web url	"https://www.nytimes.com/2020/06/11/technology/facebook-chris-cox.html"
Lead paragraph	"SAN FRANCISCO — Facebook said on Thursday that Chris Cox, a former top executive, was returning to the company as chief product officer."
Headline	{main: "Facebook Brings Back a Former Top Lieutenant to Zuckerberg", sub: None }
Keywords	{name: "subject", value: "Social Media", rank: 1 }, {name: "subject", value: "Mobile Applications", rank: 2 }, {name: "subject", value: "Appointments and Executive Changes", rank: 3 },
	{name: "organizations", value: "Facebook Inc", rank: 4 }, {name: "persons", value: "Cox, Chris (1982-)", rank: 5 }, {name: "persons", value: "Zuckerberg, Mark E", rank: 6 }
Other variables	{document type: "article", type of material: "News", news desk: "Business", section name: "Technology", word count: 370, publish date: "2020-06-11T19:16:33+0000" }

Training	positive	negative	neutral	
count	913	410	2050	
Testing	predicted positive	predicted negative	predicted neutral	observation
true positive	229	4	22	255
true negative	6	98	0	104
true neutral	48	16	421	485
accuracy	0.886			844

Project 1: Text Data extraction

- ▶ Left: Two examples for the LCS edit distance match and embedding distance match to find the associate companies with each news article.
- ▶ Right: Scatter plot of sentiment at t on x-axis versus the dependent variable return at $t + 1$ on y-axis, with the dashed line representing the simple fitted regression line for the two variables. These plots were generated every year with missing values excluded. Out of the 20 years, 18 have a positive slope.

Ticker	"FB"	
Company Name	"Facebook, Inc."	
NYT Organizations	"Flatbook Corp"	
Processed String	["Facebook Inc", "Flatbook Inc"]	
LCS edit distance	String Matched: "Fabook Inc", matched length: 10, percentage matched: 83.33%	
String	"Flatbook Corp"	"Facebook, Inc."
Embed Words	["Flat", "book", "Corp"]	["Facebook", "Inc"]
Embed vectors	[[-0.054, -0.1807, ..., -0.042, 0.133], [-0.022, 0.011, ..., 0.095, -0.033], [-0.062, 0.109, ..., 0.048, -0.098]]	[[-0.083, 0.094, ..., 0.086, 0.050], [-0.033, 0.078, ..., -0.027, -0.103]]
Combined vectors	[[-0.079, -0.035, ..., 0.159, -0.069], [-0.082, 0.121, ..., 0.042, -0.038]]	
Cosine Similarity	-0.36	



Project 1: Ensemble Model Results

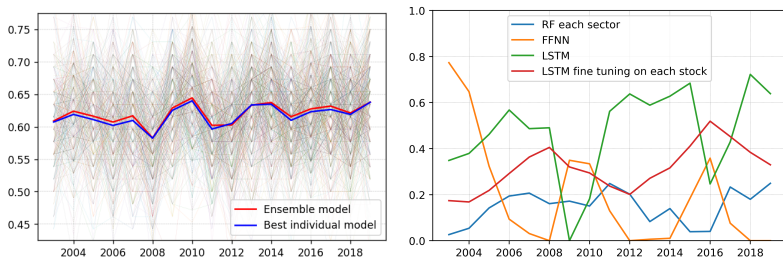
- Updated yearly, fitted using past years' prediction outputs as inputs to the ensemble model.

Table 1: Combined performance of several prediction models, as well our ensemble model, on all 518 current and past S&P 500 stocks during the period January 1, 2003 to December 31, 2019.

Model	Input data	Tuning parameters	MAE	RMSE	DA	UDA	DDA
Random Forest, for each sector ten years rolling, yearly update	return, sentiment, PE, PS, PB,CCI, MACDH, RSI, KDJ, WR, ATR, CMF from last period	max depth=8, max features=all, estimators=400, minimize MAE	0.0244	0.00144	60.35	74.90	42.20
Feed Forward Network, all stocks ten years rolling, monthly update		one forward layer with 48 nodes, relu activation, Dropout 0.6, minimize MAE	0.0243	0.00142	60.92	78.53	38.95
LSTM with all stocks together, fine tuning on each stock, all past data	return, sentiment, PE, PS, PB,CCI, MACDH, RSI, KDJ, WR, ATR, CMF from past three periods	1 LSTM layer, 32 nodes each, Dropout 0.6, minimize MAE, 3 outputs with one shift	0.0245	0.00146	61.07	67.75	52.61
LSTM, all stocks together, all past data, yearly update		2 LSTM layer, 32 nodes each, Dropout 0.6, minimize MAE, 3 outputs with one shift	0.0240	0.00140	61.69	74.07	46.13
Ensemble, yearly update, two year rolling	Individual Model outputs from above four model	Regression with constant fixed at 0, coefficient forced to be greater or equal than 0	0.0239	0.00138	62.09	73.71	47.61

Project 1: Ensemble Model Performance and Weights

- ▶ Left: The yearly DA comparison between ensemble model and the best individual model for that year. The thin lines represent the ensemble DA of different individual stocks by year.
- ▶ Right: The weights placed on different individual models for the ensemble model predicting 518 S&P stocks, by year.



Project 1: Comparison

We used the excess of accuracy over a hypothetical model that always predicts increases for comparison. In another word, the excess of accuracy compares with the implied upward probability by assuming stock prices are unpredictable.

Table 2: Comparison of the improvement in DA among different models. Our model for the S&P 500 index was fitted using the same ensemble method but with the median of the individual stock model outputs.

Author	model refresh and predicting period	Predicting	Predicted accuracy	Percentage of time stock price increased in test set	Improved accuracy compared to model only predicting upwards
Jiang et al. for S&P 500 index	no refresh, 09/01/2012-04/01/2019	closing on next month	69.17%	67.35%	1.82%
Yu an Yan for S&P 500 index	yearly update, 01/01/2010-12/29/2017	closing on next day	58.07%	54.59%	3.48%
Ding et al. for S&P 500 index	no refresh, 02/22/2013- 11/21/2013	closing on next day	64.21%	59.68%	4.53%
Gorenc Novak and Velušček for 370 S&P stocks	20 days update, 10/27/2005-06/14/2013	high on next day	61.16%	57.07%	4.09%
Our model for S&P 500 index	monthly & yearly mixed 01/01/2003-12/31/2019	closing on next week	66.18%	60.65%	5.53%
Our model for 518 S&P stocks	monthly & yearly mixed 01/01/2003-12/31/2019	closing on next week	62.12%	55.49%	6.63%

Project 2: Functional Clustering of Fictional Narratives Using Vonnegut Curves, Motivation

Kurt Vonnegut, a famous American writer, during a public lecture in 2010, once proposed in a non-technical setting, “The fundamental idea is that stories have shapes which can be drawn on graph paper”. After Vonnegut came up with his idea, it drew some attention within the literary field and people have tried to use his curves to characterize the main character’s ill or good fortune for several popular novels. Such efforts were made by journalists, e.g., posted on the *Washington Post* website Swanson, 2015, using Vonnegut’s idea from a literary point of view rather than a mathematical or statistical one.

Project 2: Data

We chose 62 well-known novels which are available in the Project Gutenberg database from 1612 to 1925, from two lists of the 100 best novels ever written in English and in all languages, selected by The Guardian.

Table 3: Summary of the first ten selected novels.

Year	Title	Author	Main Character	First Tense
1612	Don Quixote	Miguel De Cervantes	Don Quixote	No
1678	Pilgrim's Progress	John Bunyan	None	Yes
1719	Robinson Crusoe	Daniel Defoe	Robinson Crusoe	Yes
1726	Gulliver's Travels	Jonathan Swift	Lemuel Gulliver	Yes
1749	Tom Jones	Henry Fielding	Tom Jones	No
1759	Tristram Shandy	Laurence Sterne	Tristram Shandy	Yes
1782	Dangerous Connections	Pierre Choderlos De Laclos	Epistolary format	Yes
1816	Emma	Jane Austen	Emma Woodhouse	No
1818	Frankenstein	Mary Shelley	Epistolary format	Yes
1818	Nightmare Abbey	Thomas Love Peacock	Scythrop	No

Project 2: Clean and turn .txt file into lists of sentences

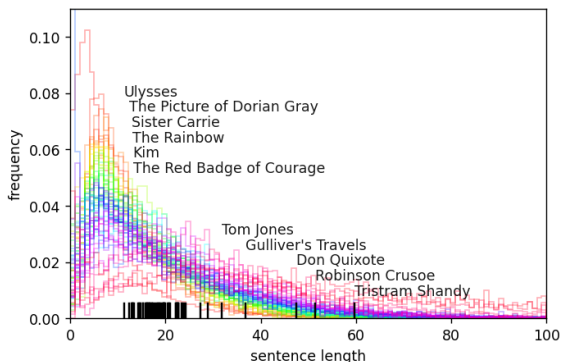


Figure 1: Histogram of how the length of sentences was distributed among different novels, by a percentage weighting. The black lines at the bottom indicate the average sentence length. We also noted the 5 shortest and 5 longest novels in terms of average sentence length, indicating the trend that the average sentence length became shorter over the 300 years of time.

Project 2: Sentiment extraction

- ▶ Average novel length of 7,173 sentences, average sentence length of 23.15 words. Longest sentence 256 words, shortest one word.
- ▶ Cleaned sentences as input, output sentiment
- ▶ Google API-based model to infer Sentence Sentiment and Entity Sentiment

Table 4: Example of sentence based sentiment

Sentence	Emma Woodhouse, handsome, clever, and rich, with a comfortable home and happy disposition, seemed to unite some of the best blessings of existence; and had lived nearly twenty-one years in the world with very little to distress or vex her.
Sentiment	<pre>{sentence sentiment: 0.70, entity sentiment: { 'entity name': 'Emma Woodhouse', 'type': 'PERSON', 'salience score': 0.52, 'sentiment score': 0.40}, { 'entity name': 'disposition', 'type': 'OTHER', 'salience score': 0.13, 'sentiment score': 0.30}, { 'entity name': 'home', 'type': 'LOCATION', 'salience score': 0.06, 'sentiment score': 0.40}, ... { 'entity name': 'distress', 'entity type': 'OTHER', 'salience score': 0.03, 'sentiment score': 0.30} }</pre>

Project 2: Smoothing sentiment with kernel smoothing

$$f_{i,h_i}(t) = \frac{1}{nh} \sum_{j=1}^n Y_i(t_j) K\left(\frac{t-t_j}{h_i}\right), h_i \leq t \leq 1 - h_i, \text{ otherwise NA}$$

$$\text{where } K(u) = \frac{15}{16}(1-u^2)^2, |u| \leq 1$$

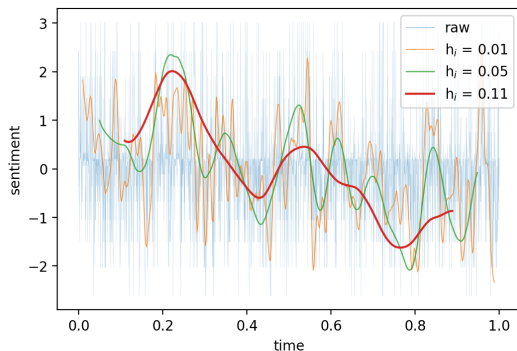
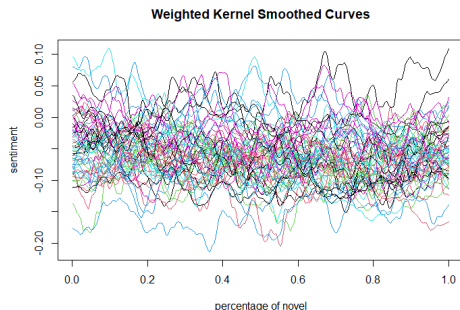
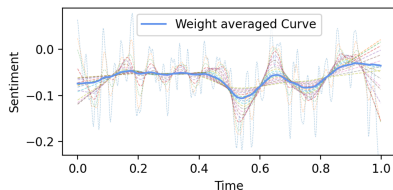


Figure 2: The raw and smoothed sentiment curve for *The Great Gatsby*.

Project 2: Smoothing sentiment with kernel smoothing

However, a large bandwidth would lead to the kernel weights being spread onto regions without data (outside of the left and right boundaries of the data).

- Method 1: Using a weighted average algorithm to average curves under different bandwidths.



Project 2: Method 2 B-spline smoothing

$$Y_i(t) = f_i(t) + \epsilon_{it}, \text{ for some } f_i : [0, 1] \rightarrow \mathbb{R}$$

where ϵ is independent of t with $\mathbb{E}(\epsilon) = 0$ and $\mathbb{E}(\epsilon^2) = \sigma^2$, and $f_i(t)$ was approximated with the spline basis vector $\phi(t)$:

$$f_i(t) \approx \phi(t)^T \beta_i = \sum_{j=1}^p \beta_{ij} \phi_j(t)$$

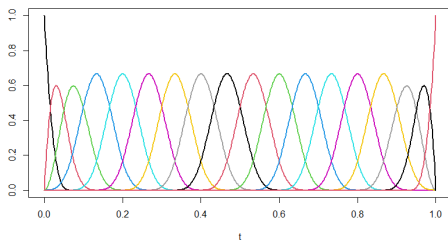


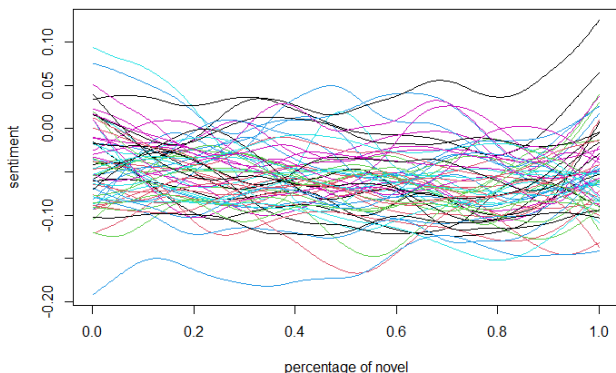
Figure 3: An example for the basis function with order 3 and 15 equally spaced knots; 100 Knots would have more basis curves superimposed.

Project 2: Penalized B-spline smoothing

B-spline smoothing still has unwanted behavior at the two sides.
The penalized B-spline objective function to minimize is:

$$\hat{f}_i(t) = \arg \min_{f_i(t)} \left(\sum [Y_i - f_i(t)]^2 + \lambda \int_0^1 [f_i''(t)]^2 dt \right)$$

Penalized B-spline Smoothed Curves



Project 2: Validate sentiment results

Term Frequency-Inverse Document Frequency (TF-IDF):

$$\text{TF-IDF}(term, novel_i) = \frac{\text{count}(term, novel_i)}{\sum_{term} \text{count}(term, novel_j)} \cdot \log\left(\frac{\text{sum of available novels}}{1 + \sum_j (\text{sum of novels containing } term)} + 1\right)$$

A Passage to India			
Keyword	TF-IDF	Appearances	Correlation with sentiment
echo	0.000238	26	-0.68
British	0.000238	26	-0.60
evidence	0.000176	24	-0.80
mistake	0.000162	24	-0.59

Keyword	TF-IDF	Appearances	Correlation with sentiment
Hindu	0.000537	23	0.30
club	0.000455	57	0.38
Professor	0.000579	38	0.57
pleasant	0.000180	26	0.65

Project 2: Validate sentiment results

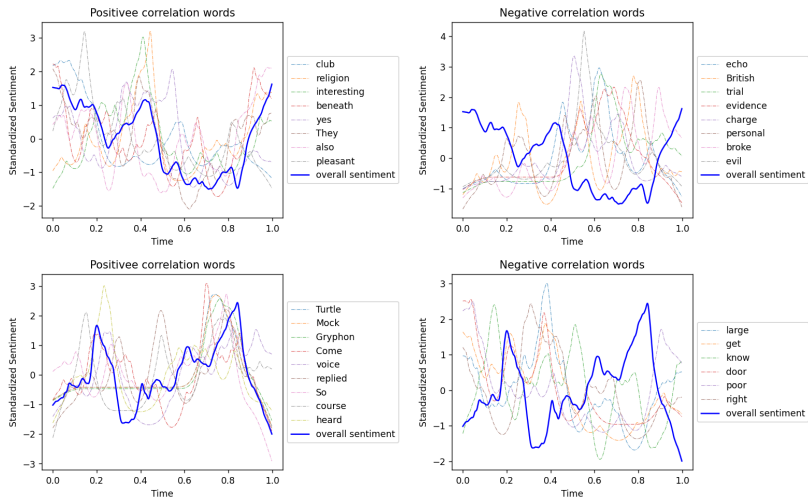
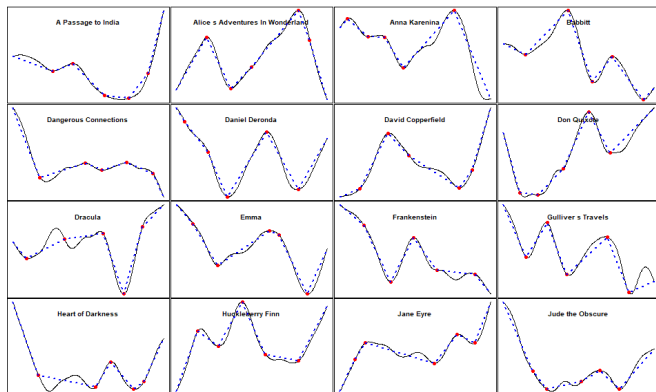


Figure 5: The sentiment curve for and the frequency of selected keywords, for *A Passage to India* and *Alice's Adventures In Wonderland*.

Project 2: Finding the most prominent Landmarks

Examples of curves with fixed 5 landmarks indicated with dots connected by the dashed lines; this may be helpful for interpreting patterns across the novels:



Warping functions: $\Gamma = \{\gamma : [0, 1] \rightarrow [0, 1], \gamma(0) = 0, \gamma(1) = 1\}$
Transfer curves from $f(t)$ to $f(\gamma(t))$

Project 2: Finding the most prominent Landmarks

Assume the candidate landmark $\mathbf{l}_k = (l_1, l_2, \dots, l_k)$ are generated from the order statistics of k iid uniform $[0, 1]$ random variables, so that $l_1 \leq l_2 \leq \dots \leq l_k$; this represents our noninformative prior knowledge about the \mathbf{l}_k 's. Then (l_1, l_2, \dots, l_k) is distributed uniformly over the simplex:

$$\{(t_1, \dots, t_k) \in \mathbb{R}^k | 0 \leq t_1 \leq t_2 \leq \dots \leq t_k \leq 1\}$$

Then let $p_i = l_i - l_{i-1}$, where $1 \leq i \leq k+1$, $l_0 = 0$, $l_{k+1} = 1$:

$$p_1, p_2, \dots, p_{k+1} \sim \text{Dirichlet}(\alpha), \text{ with } \alpha = 1$$

Define $f_{\mathbf{l}_k}(t)$ denote the linear interpolation connecting the left endpoint, landmarks, and right endpoint. We assume for $(t_1 = 1/n, t_2 = 2/n, \dots, t_n = n/n)$:

$$[f(t_1) - f_{\mathbf{l}_k}(t_1), \dots, f(t_n) - f_{\mathbf{l}_k}(t_n)] \sim \mathcal{N}_n(0_n, \Sigma_{n \times n})$$

Project 2: Finding the most prominent Landmarks

We modeled $\Sigma_{n \times n}$ with a scaling factor ℓ^2 such that $\Sigma_{n \times n} = \ell^2 I_{n \times n}$.

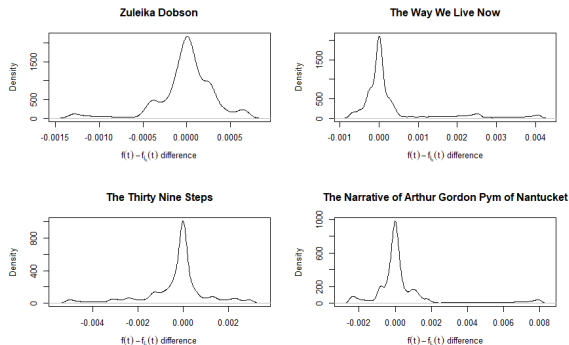


Figure 6: The distribution of $[f(t_1) - f_k(t_1), \dots, f(t_n) - f_k(t_n)]$, when we chose \mathbf{l}_k to be all local maximum and minimum of the original curve and those of the first and second derivatives as candidate landmarks.

Project 2: Find Landmarks Using MCMC

$$\pi(\mathbf{l}_k, \ell^2 | f(t_1), \dots, f(t_n)) \propto \pi(f(t_1), \dots, f(t_n) | \mathbf{l}_k, \ell^2) \pi(\ell^2, \mathbf{l}_k)$$

Here we set our proposal distribution $Q(\mathbf{l}_k' | \mathbf{l}_k) = Q(p_1', p_2', \dots, p_{k+1}' | p_1, p_2, \dots, p_{k+1})$ to draw from a Dirichlet Distribution with concentration parameter α . Also to prevent points getting too close to each other, we implicitly assumed the minimum distance between two landmarks to be above b :

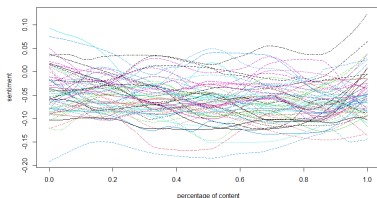
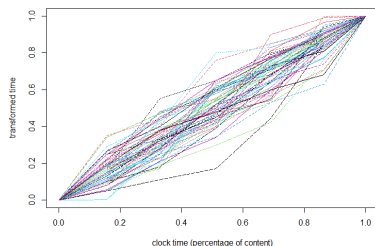
$$p_1', p_2', \dots, p_{k+1}' \sim \text{Dirichlet}(\alpha(b + p_1), \alpha(b + p_2), \dots, \alpha(b + p_{k+1}))$$

And the proposal distribution $Q(\ell^{2'} | \ell^2)$ is chosen to be a Log-normal distribution with mean ℓ^2 and standard deviation parameter σ^2 :

$$\ell^{2'} \sim \text{Log-normal}(\log(\ell^2) - \frac{\sigma^2}{2}, \sigma^2)$$

Project 2: Landmark Registration

An example showing a warping function by linearly warping the landmarks from each curve to the mean landmark location of all the curves. The mean landmark locations are plotted on the x-axis, and locations of corresponding individual curves' landmarks are on the y-axis.



Similar with minimizing $\|(f(t), f_{l_k}(t))\|^2$ to find landmarks, we can find a set of points representing the warping function by solving $\inf_{\gamma \in \Gamma} \|(f_i \circ \gamma_i) - f_1\|$.

Project 2: SRVF Framework

We introduce the Square Root Velocity Function (SRVF), commonly used in elastic shape analysis:

$$q(t) = \{\dot{f}(t)/\sqrt{|\dot{f}(t)|} \text{ when } \dot{f}(t) \neq 0 \text{ and } 0 \text{ when } \dot{f}(t) = 0\}$$

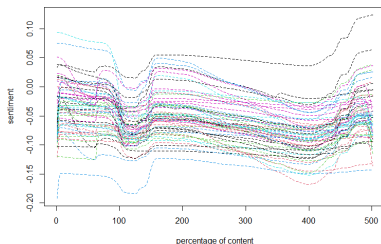
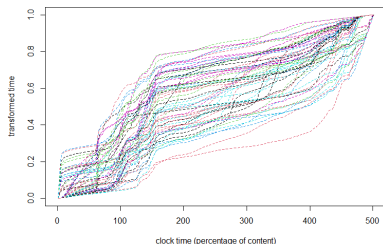
An important property of the SRVF approach is that the \mathbb{L}_2 distance $\|q\|^2 = \{\int_0^1 q(t)^2 dt\}$ between any $q_1, q_2 \in \mathbb{L}_2$ is invariant to any warping $\gamma \in \Gamma$ (Srivastava et al., 2011):

$$\|(q_1, \gamma) - (q_2, \gamma)\|^2 = \|q_1 - q_2\|^2$$

The fact that this distance is invariant to warping yields a unique center, the Karcher mean u_Q , for q_1, q_2, \dots, q_n , regardless of the warping:

$$\hat{u}_Q = \arg \inf_{u \in Q} \sum_{i=1}^N \inf_{\gamma_i \in \Gamma} \|u_Q - (q_i, \gamma_i)\|_2^2$$

Project 2: SRVF Framework



Directly minimizing the distance to Karcher mean would often over smooth, thus Srivastava et al. (2011) suggested adding a penalizing term to find a proper γ_i , with \mathcal{R} a smoothness penalty to keep curve close to $\gamma_i(t) = t$:

$$\arg \inf_{\gamma_i \in \Gamma} (\|u_Q - (q_i, \gamma_i)\|_2^2 + \lambda \mathcal{R}(\gamma_i))$$

Project 2: SRVF Framework

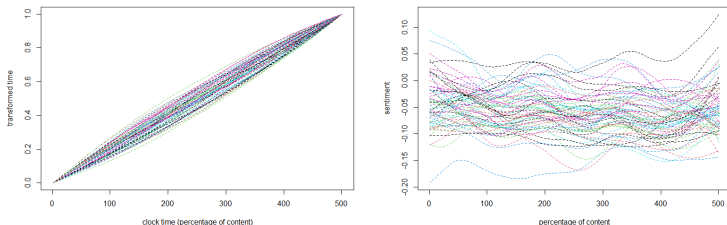


Figure 7: Warping functions and warped curves under $\lambda = 0.1$

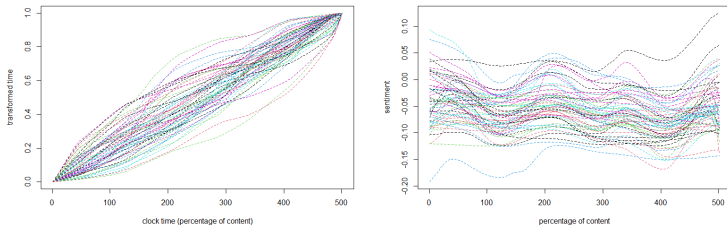


Figure 8: Warping functions and warped curves under $\lambda = 0.01$

Project 2: Optimal number of clusters

The clustering can be performed by minimizing metrics like the L_2 norm, i.e., $\|f_1(t), f_2(t)\|^2 = \sqrt{\int_{t=0}^1 [f_1(t) - f_2(t)]^2 dt}$:

$$\arg \sum_{i=1}^k \inf_{u_i \in U} \sum_{j \in \text{cluster}_i} \inf_{a, b \in \mathbb{R}} \|(f_j(t), u(t))\|_2^2$$

or by maximizing a functional version of the Pearson correlation ρ , where $\rho(f_1(t), f_2(t)) = \frac{\int_{t=0}^1 f_1(t)f_2(t) dt}{\|f_1(t)\|^2 \|f_2(t)\|^2}$:

$$\arg \sum_{i=1}^k \max_{u_i \in U} \sum_{j \in \text{cluster}_i} \max_{a, b \in \mathbb{R}} \rho(f_j(t), u(t))$$

We utilized the elbow method to check the improvement of the chosen metric attained by adding one more cluster, to determine the optimal number of clusters.

Project 2: Optimal number of clusters

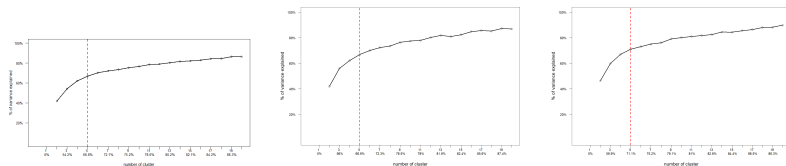


Figure 9: Percentage of variance explained under each cluster number k , for k-mean clustering based on the L_2 distance metric for linear landmark, lambda 0.1, and lambda 0.01 warping.

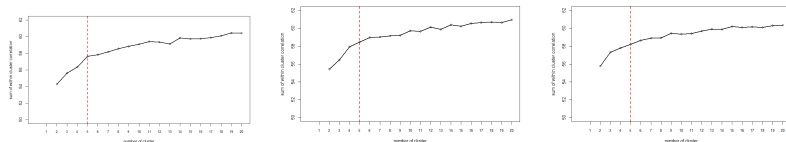


Figure 10: Sum of within-cluster correlation for each cluster number k , for k-mean clustering based on the Pearson correlation metric for linear landmark, lambda 0.1, and lambda 0.01 warping.

Project 2: Clustering

To cluster our sentiment curves, we employ the following objective function, which permits clustering and warping at the same time:

$$\arg \sum_{i=1}^k \inf_{u_i \in U} \sum_{i \in \text{cluster}_j} \left(\inf_{\gamma_i \in \Gamma} \|u_i - (q_i, \gamma_i)\|_2^2 + \lambda \mathcal{R}(\gamma_i) \right)$$

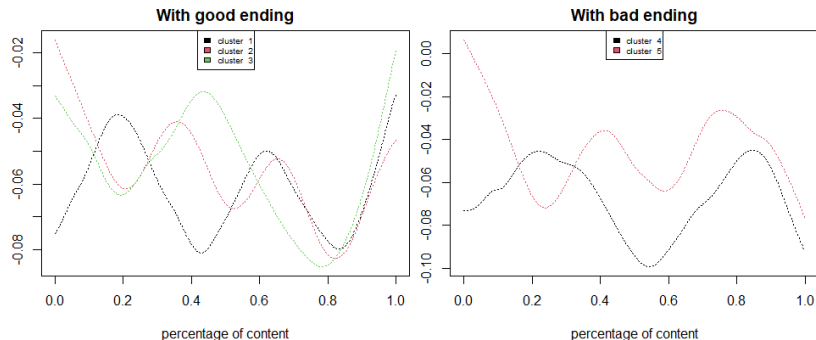
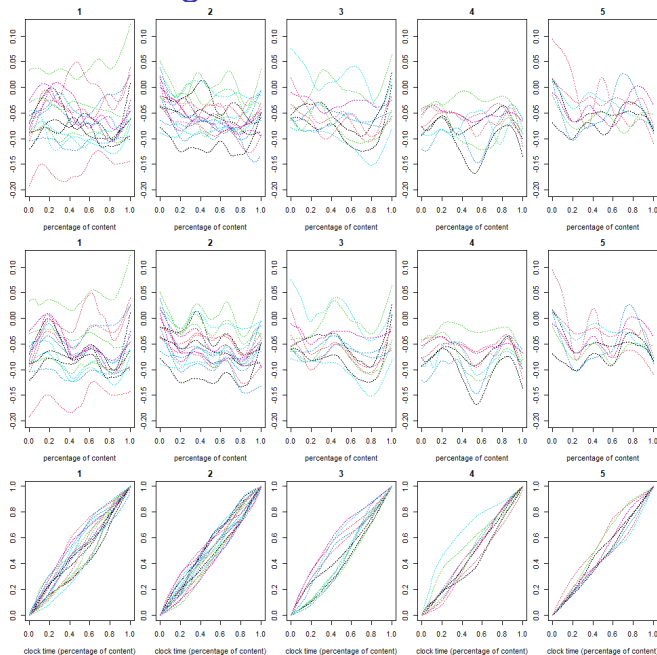


Figure 11: Here we show the result of clustering under $k = 5$, $\lambda = 0.01$ for the template curves after warping, using k-means clustering.

Project 2: Clustering



Project 2: Conclusion

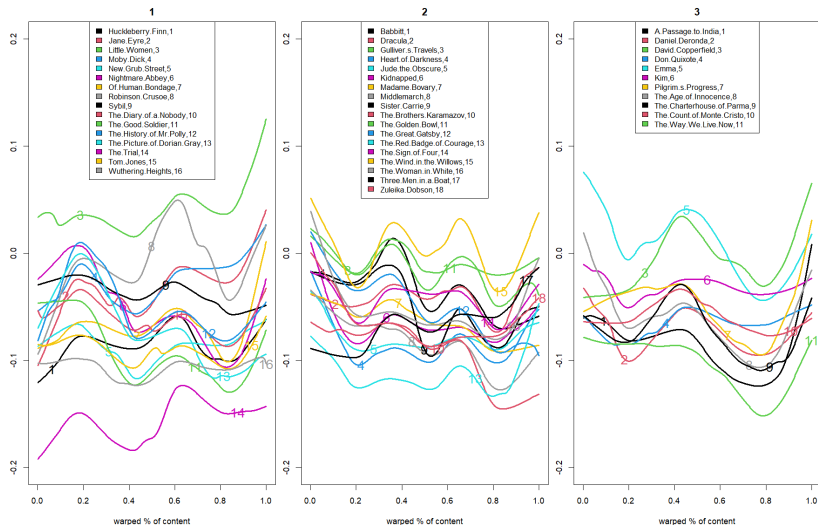


Figure 12: The curves membership, using k-means clustering.

Project 2: Conclusion

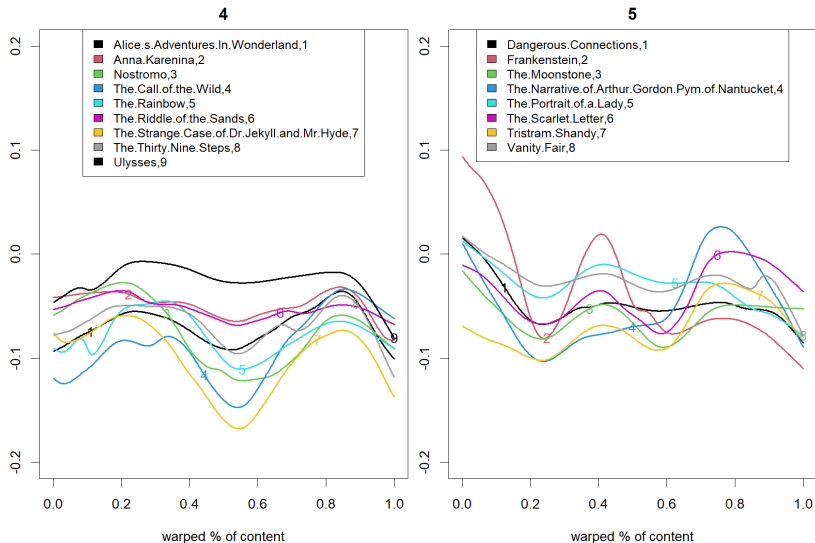


Figure 13: The curves membership, using k-means clustering.

Project 2: Future Research

- ▶ Apply the techniques to the main characters' data.
- ▶ Use these patterns to simulate new short stories with similar patterns which adhere to the sentiment curve categories.
- ▶ Functional form of a Dirichlet process mixture (DPM) model, which can adaptively learn the number of clusters.

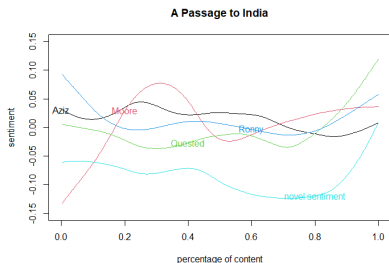


Figure 14: The sentiment curve for the novel *A Passage to India* and four major characters in the novel. The characters' names appear irregularly across the novel, but still fit nicely into the B-spline form.

Project 3: Data and Environment Setup

The environment accepts an input of a specific year, and then the environment will assign all previous years' data to the training set and the specified year's data into the test set. Next the environment applies the Yeo-Johnson transformation operator first to the training dataset and then to both the training and testing dataset. Finally, the environment will start to obtain sample data by randomly selecting a starting date and generating the r_t , *CCI*, *MACDH*, *RSI*, *KDJ*, *WR*, *ATR*, *CMF* information for each stock, for further tuning in the action, observation, and states space.

Indicator Name	Quick Introduction	Window period
CCI	The Commodity Channel Index can help to identify price reversals, price extremes and trend strength.	20 days
MACDH	The Moving Average Convergence Divergence Histogram is the difference between MACD and the MACD signals.	12, 26, and 9 days
RSI	The Relative Strength Index, which measures the average of recent upward movement versus the upward and downward movements combined, in a percentage form.	14 days
KDJ	The Stochastic Oscillator measures where the close is relative to the low and high.	14, 3, and 3 days
WR	The Williams %R index is another index for buy signals by measuring the current price in relation to the past N periods.	14 days
ATR	The Average True Range provides an indicator for the volatility of price. We converted ATR into percentages.	14 days
CMF	The Chaikin Money Flow provides an indicator related to the trading volume.	20 days

Project 3: Hidden Markov Model (HMM) Under RL framework

Our observations \mathbf{o}_t can be defined as the stock we were holding at that time, plus the industry, price and indicator information of all the stocks we are considering.

$$\begin{aligned} \mathbf{o}_t = & \{ o_{stock_1} : \text{ticker}_1, pos_t^1, \mathbf{X}_{t,t-1,t-2}^{stock_1} = [r_{t,t-1,t-2}^{stock_1}, CCI_{t,t-1,t-2}^{stock_1}, \dots, CMF_{t,t-1,t-2}^{stock_1}] \in [-4.5, 4.5]^p, \\ & o_{stock_2} : \text{ticker}_2, pos_t^2, \mathbf{X}_{t,t-1,t-2}^{stock_2} = [r_{t,t-1,t-2}^{stock_2}, CCI_{t,t-1,t-2}^{stock_2}, \dots, CMF_{t,t-1,t-2,t-3}^{stock_2}] \in [-4.5, 4.5]^p, \\ & \dots \\ & o_{stock_d} : \text{ticker}_d, pos_t^d, \mathbf{X}_{t,t-1,t-2}^{stock_d} = [r_{t,t-1,t-2}^{stock_d}, CCI_{t,t-1,t-2}^{stock_d}, \dots, CMF_{t,t-1,t-2}^{stock_d}] \in [-4.5, 4.5]^p, \\ & cash : pos_t^{cash}, \text{Restricted to: } \underbrace{pos_t^1 + pos_t^2 + \dots + pos_t^{cash}}_{n=d+1} = 1 \} \end{aligned}$$

Thereafter the hidden state \mathbf{s} was defined as the hidden mean trend movement $\boldsymbol{\mu}_{t+1}$ of return \mathbf{r}_{t+1} , and the hidden volatility $\boldsymbol{\sigma}_{t+1}$ for the next period of the d stocks, which later helps us to make buy, sell, or hold decisions though policy $\pi(\mathbf{a}_t | \mathbf{o}_{1:t})$, with state space \mathcal{S} :

$$\mathbf{s}_t = \{ [\mu_{t+1}^{stock_1}, \sigma_{t+1}^{stock_1}], [\mu_{t+1}^{stock_2}, \sigma_{t+1}^{stock_2}], \dots, [\mu_{t+1}^{stock_d}, \sigma_{t+1}^{stock_d}] \} \in \mathcal{S} = \{ [\mathbb{R}, \mathbb{R}^+]^d \}$$

Project 3: Hidden Markov Model (HMM) Under RL framework

Our Hidden Markov Model $\mathbb{P}(\mathbf{s}_t|\mathbf{o}_t)$ is fitted by calculating the conditional probability distribution of $\mathbf{r}_{t+1} = [r_{t+1}^{stock_1}, \dots, r_{t+1}^{stock_d}]$ based on \mathbf{s}_t

$$\mathbb{P}_{\theta}(\mathbf{r}_{t+1}|\mathbf{o}_t) = \mathbb{P}_{\theta}(\mathbf{r}_{t+1}|\mathbf{s}_t)\mathbb{P}_{\theta}(\mathbf{s}_t|\mathbf{o}_t)$$

Which we modeled as a correlated multivariate Gaussian Distribution, parameterized by $\mu_{t+1} = \phi(\mathbf{o}_t)^{\top}\theta_{\mu}$, and $\Sigma_{t+1} = \phi(\mathbf{o}_t)^{\top}\theta_{\sigma}$:

$$\mathbb{P}_{\theta}(\mathbf{r}_{t+1}|\mathbf{o}_t) \sim \mathcal{N}(\mu_{t+1}, \Sigma_{t+1}) | \mu_{t+1} = \phi(\mathbf{o}_t)^{\top}\theta_{\mu}, \Sigma_{t+1} = \phi(\mathbf{o}_t)^{\top}\theta_{\sigma}$$

where $\phi(\mathbf{o})$ are the basis functions evaluated at \mathbf{o} , learned via LSTM networks.

Project 3: LSTM Basis function

For each $\mathbf{X}_{t,t-1,t-2} = [\mathbf{X}_{t,t-1,t-2}^{stock_1}, \mathbf{X}_{t,t-1,t-2}^{stock_2}, \dots, \mathbf{X}_{t,t-1,t-2}^{stock_d}]$ in \mathbf{o}_t , we share the LSTM units among stocks:

$$h_{i,t-2}, c_{i,t-2} = \text{LSTM}_i(\mathbf{X}_{t-2}, h_0, c_0), \text{ for } i \text{ in } 1, 2, \dots, 128$$

$$h_{i,t-1}, c_{i,t-1} = \text{LSTM}_i(\mathbf{X}_{t-1}, h_{t-2}, c_{t-2}), \text{ for } i \text{ in } 1, 2, \dots, 128$$

$$h_{i,t}, c_{i,t} = \text{LSTM}_i(\mathbf{X}_t, h_{t-1}, c_{t-1}), \text{ for } i \text{ in } 1, 2, \dots, 128$$

and

$$\mu_{t+1}^{stock_j} = \theta_{\mu}^{1,stock_j} h_{1,t} + \theta_{\mu}^{2,stock_j} h_{2,t} + \dots + \theta_{\mu}^{128,stock_j} h_{128,t} + \theta_{\mu}^{constant,stock_j}$$

$$\sigma_{t+1}^{stock_j} = \theta_{\sigma}^{1,stock_j} h_{1,t} + \theta_{\sigma}^{2,stock_j} h_{2,t} + \dots + \theta_{\sigma}^{128,stock_j} h_{128,t} + \theta_{\sigma}^{constant,stock_j}$$

Project 3: Model Under RL framework

We standardized \mathbf{X}_t^{stock} so they are distributed with mean zero and standard deviation one, so we assumed them to be stationary. The Σ_{t+1} were then modeled by a stationary covariance kernel with a squared exponential (SE) kernel function:

$$\Sigma_{t+1}^{jk} = K_{SE}(\sigma_{t+1}^{stock_j}, \sigma_{t+1}^{stock_k}) = \sigma_{t+1}^{stock_j} \sigma_{t+1}^{stock_k} \cdot \tau^2 \exp\left(-\frac{\|\mathbf{x}_t^{stock_j} - \mathbf{x}_t^{stock_k}\|_2^2}{2\ell^2}\right)$$

where $\log \sigma$ are learnt by neural network, τ^2, ℓ^2 are the hyperparameters predefined, and $\|\cdot\|_2$ the L_2 norm. In this setting, we have $\exp\left(-\frac{\|\mathbf{x}_t^{stock_j} - \mathbf{x}_t^{stock_k}\|_2^2}{2\ell^2}\right) \in [0, 1)$. In practice, we applied a small nugget term (Chen et al., 2021) v to the diagonal of the matrix, to avoid the covariance matrix being close to singular, for numerical stability.

$$\Sigma_{t+1}^{jj} = (\sigma_{t+1}^{stock_j})^2 [\tau^2 + v]$$

Project 3: Modeling Correlation Among Stocks

Our objective function to maximize is the log-likelihood of our model parameters θ_μ , θ_σ , and the parameters for ϕ , denoted \mathbf{W}_ϕ , given the true return $\mathbf{r}_{t+1}^{true} = [r_{t+1,true}^{stock_1}, \dots, r_{t+1,true}^{stock_d}]$ at $t+1$, and the observation \mathbf{o}_t at t , for the d chosen stocks:

$$\text{Likelihood} = \mathbb{P}[\mathbf{W}_\phi, \theta_\mu, \theta_\sigma | \mathbf{o}_t, \mathbf{r}_{t+1}^{true}]$$

The gradient of θ_μ can be calculated as follows:

$$\begin{aligned}\nabla_{\theta_\mu} \log \text{Likelihood} &= \nabla_{\theta_\mu} \log \left\{ \det(2\pi\Sigma)^{\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)^\top \Sigma^{-1}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)\right] \right\} \\&= \nabla_{\theta_\mu} \left\{ \log(\det(2\pi\Sigma)^{\frac{1}{2}}) + \log(\exp\left[-\frac{1}{2}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)^\top \Sigma^{-1}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)\right]) \right\} \\&= \nabla_{\theta_\mu} \left(-\frac{1}{2}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)^\top \Sigma^{-1}(\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu) \right), \text{ Let } \mathbf{u} = (\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu) \\ \nabla_{\theta_\mu} \log \text{Likelihood} &= -\frac{1}{2} \nabla_{\theta_\mu} (\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)^\top \Sigma^{-1} \mathbf{u} \\&= -\frac{1}{2} (-\phi(\mathbf{o})) \left[\mathbf{u}^\top \frac{\partial \Sigma^{-1} \mathbf{u}}{\partial \theta_\mu} + (\Sigma^{-1} \mathbf{u})^\top \frac{\partial \mathbf{u}}{\partial \theta_\mu} \right] \\&= \frac{1}{2} \phi(\mathbf{o}) [\mathbf{u}^\top (\Sigma^{-1} + (\Sigma^{-1})^\top)] \\&= \frac{1}{2} \phi(\mathbf{o}) (\mathbf{r} - \phi(\mathbf{o})^\top \theta_\mu)^\top [\Sigma^{-1} + (\Sigma^{-1})^\top]\end{aligned}$$

Project 3: Model Stock Portfolio

Next we prove that for any d -dimensional multivariate normal distribution $\mathbf{r} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu} = (\mu_1, \dots, \mu_d)^\top$ and $\Sigma_{jk} = \text{cov}(r_j, r_k)$, $j, k = 1, \dots, d$. Define $Z = \mathbf{pos}^\top \mathbf{r} = \sum_{j=1}^d \text{pos}_j r_j$, and then the characteristic function of Z is given by:

$$\begin{aligned}\varphi_Z(t) &= E[\exp(itZ)] = E[\exp(it\mathbf{pos}^\top \mathbf{r})] = \varphi_{\mathbf{r}}(t\mathbf{pos}) \\ &= \exp\left(it \sum_{j=1}^d \text{pos}_j \mu_j - \frac{1}{2} t^2 \sum_{j=1}^d \sum_{k=1}^d \text{pos}_j \text{pos}_k \Sigma_{jk}\right)\end{aligned}$$

which is normal with

$$\mu_Z = \sum_{j=1}^d \text{pos}_j \mu_j, \quad \sigma_Z^2 = \sum_{j=1}^d \sum_{k=1}^d \text{pos}_j \text{pos}_k \Sigma_{jk}$$

Project 3: Model Stock Portfolio

Also, we can decompose the standard deviation σ_Z^2 of the portfolio into parts contributed by $stock_i$ and the parts contributed by other stocks, which later helps our stock selecting agents in deciding whether to include $stock_i$ in the portfolio:

$$\sigma_Z^2 = \sum_{j=1}^d \sum_{k=1}^d pos_j pos_k \Sigma_{jk} = [2 \cdot pos_i \sum_{k \neq i}^d pos_k \Sigma_{ik} + pos_i^2 \Sigma_{ii}] + \sum_{j \neq i}^d \sum_{k \neq i}^d pos_j pos_k \Sigma_{jk}$$

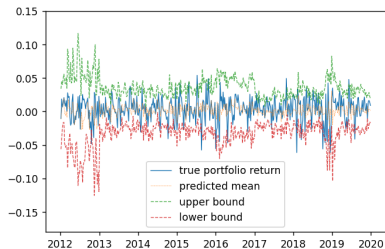
We define $\Sigma_i^{\text{contrib}}$ as the volatility contributed by $stock_i$:

$$\Sigma_i^{\text{contrib}} = pos_i \sum_{k=1}^d pos_k \Sigma_{ik}$$

When fixing other parts unchanged, this contribution can be viewed as an increasing function of pos_i .

Project 3: Model Stock Portfolio - Preliminary Results

- ▶ Left: The training and testing likelihood improvement during the first 100 epochs for the sampled year 2012, when setting d to be 5, for a feed forward network basis function (left, three layers with each layer 128 units).
- ▶ Right: The 95% prediction intervals of equal weighted 5 stock portfolios with AAPL, AMZN, IBM, GOOGL, MSFT for the testing data set under the feed forward network basis from 2012 to 2020. The coverage rate is 92.33%.



Project 3: MULTI-AGENT Order Dispatchment

First assume the money we invest in each stock is forever equally weighted, and we always hold d stocks for each period, then:

$$\mathbf{o}_t = \{o_{stock_1} : \text{ticker}_1, \mathbf{X}_{t,t-1,t-2}^{stock_1}, o_{stock_2} : \text{ticker}_2, \mathbf{X}_{t,t-1,t-2}^{stock_2}, \dots, o_{stock_n} : \text{ticker}_n, \mathbf{X}_{t,t-1,t-2}^{stock_n}\}$$

and the state \mathbf{s}_t inferred from the behavior model:

$$\mathbf{s}_t = \{s_{stock_1} : \text{ticker}_1, [\mu_{t+1}^{stock_1}, \sigma_{t+1}^{stock_1}, \Sigma_1^{\text{contrib}}] \dots, s_{stock_d} : \text{ticker}_d, [\mu_{t+1}^{stock_d}, \sigma_{t+1}^{stock_d}, \Sigma_d^{\text{contrib}}]\}$$

We define $\{o^{(i)}\}$ indicating the order information for $stock_i$.

around 500 orders with $\{o_t^{(i)}\} = \{\text{ticker}_i, [\mu_{t+1}^{stock_i}, \sigma_{t+1}^{stock_i}, \Sigma_i^{\text{contrib}}]\}$

Project 3: MULTI-AGENT Order Dispatchment

For each episode, a trading agent with the current stock holding as ticker_j will start at state

$$\{s_t = s_{\text{stock}_j} : \text{ticker}_j, [\mu_{t+1}^{\text{stock}_j}, \sigma_{t+1}^{\text{stock}_j}, \Sigma_j^{\text{contrib}}]\},$$

complete an order $o^{(i)}$, then change to state

$$\{s_{t+1} = s_{\text{stock}_i} : \text{ticker}_i, [\mu_{t+1}^{\text{stock}_i}, \sigma_{t+1}^{\text{stock}_i}, \Sigma_i^{\text{contrib}}]\}$$

During this transaction, a $J_k(\pi)$ is generated for the k_{th} agent to calculate our reward. We still assume a 1% commission cost each time we buy and sell.

$$J_t^k(\pi) = \begin{cases} \mu_{t+1}^{\text{stock}_j}, & \text{if idle} \\ \mu_{t+1}^{\text{stock}_i} - 0.01/d, & \text{if take order } o^{(i)} \end{cases}$$

Then our optimization problem can be set as:

$$\operatorname{argmax}_{\pi} J_t(\pi) = \frac{\sum_{k=1}^d J_t^k(\pi)/d - r_f}{\sigma_p}, \text{ evaluation period} = H$$

where r_f is the risk-free interest rate $(1/\gamma - 1)$, and

$$\sigma_p = \sum_{j=1}^d \sum_{k=1}^d (1/d)(1/d) \Sigma_{jk} = \sum_{j=1}^d \Sigma_j^{\text{contrib}}$$

Then the episode trajectory for one agent is:

$$\{o_0, s_0, J_0, o_1, s_1, J_1, \dots, o_{H-1}, s_{H-1}, J_{H-1}\}$$

and the Q-value function can be defined as:

$$Q^{\pi}(\mathbf{s}, \mathbf{o}) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t J_t(\pi) | \mathbf{s}, \mathbf{o}\right]$$

Within this setting, and set H to be 1, $Q^{\pi}(\mathbf{s}, \mathbf{o})$ can be calculated directly without any need for further modeling.

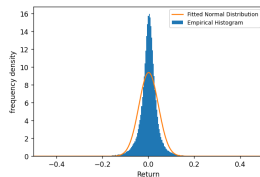
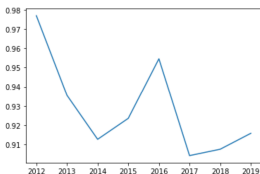
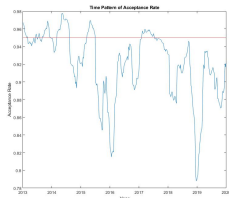
Project 3: MULTI-AGENT Decision Making Process

Algorithm 1 Decision making under fixed deterministic model

```
for each episode do
  for  $t$  in ( 0 to  $(H - 1)$  ) do
    for agent in ( 1 to  $d$  ) do
      for  $i$  in ( 1 to  $n$  ) do
        Calculate  $o^{(i)} = \{\text{tiker}_i, [\mu_{t+1}^{stock_i}, \sigma_{t+1}^{stock_i}, \Sigma_i^{\text{contrib}}]\}$ 
        Calculate  $Q^\pi(s, o^{(i)})$  with  $o^{(i)}$  and the other  $d - 1$ 
        stocks in  $\mathbf{s}_t$ 
      end for
      Choose  $o$  by by maximizing  $Q^\pi(s, o)$ 
      Update state information  $\mathbf{s}_t$ 
    end for
  end for
end for
```

Project 3: More To Do

The coverage rate for the 95% prediction interval for individual stocks for the Brownian motion methods (left), equal weighted portfolio with forward network (middle), and the empirical histogram for return versus normal distribution fitted (right).



Propose to use a Model that can take input of $\{o_t^{(i)}\} = \{\text{ticker}_i, [\mu_{t+1}^{\text{stock}_i}, \sigma_{t+1}^{\text{stock}_i}, \Sigma_i^{\text{contrib}}]\}$, calculate pos_i to maximize $Q^\pi(s, o^{(i)}, pos_i)$. If pos_i is smaller than the current position, then open a new trading agent.

Appendix

The conditional distribution of \mathbf{l}_k given $f(t_1), \dots, f(t_n), \ell^2$ is:

$$\begin{aligned}\pi(\mathbf{l}_k | f(t_1), \dots, f(t_n), \ell^2) &\propto \pi(f(t_1), \dots, f(t_n) | \mathbf{l}_k, \ell^2) \pi(\mathbf{l}_k) \\ &\propto \pi(f(t_1), \dots, f(t_n) | \mathbf{l}_k, \ell^2) \pi(\mathbf{l}_k | p_1, \dots, p_{k+1}) \pi(p_1, \dots, p_{k+1}) \\ &\propto \pi(f(t_1) - f_{\mathbf{l}_k}(t_1), \dots, f(t_n) - f_{\mathbf{l}_k}(t_n) | p_1, \dots, p_{k+1}, \ell^2) \pi(p_1, \dots, p_{k+1})\end{aligned}$$

and the conditional distribution of ℓ^2 given $f(t_1), \dots, f(t_n), \mathbf{l}_k$ is:

$$\begin{aligned}\pi(\ell^2 | f(t_1), \dots, f(t_n), \mathbf{l}_k) &\propto \pi(f(t_1), \dots, f(t_n) | \ell^2) \\ &\propto \pi(f(t_1) - f_{\mathbf{l}_k}(t_1), \dots, f(t_n) - f_{\mathbf{l}_k}(t_n) | \ell^2)\end{aligned}$$

Algorithm 2 Algorithm to generate samples for ℓ^2, \mathbf{l}_k

Input: $[f(t_1), f(t_2), \dots, f(t_n)]$, where $t_1 = 1/n, t_2 = 2/n, \dots, t_n = (n)/n$

Initialize: \mathbf{l}_k, ℓ^2

for repeat in $1, \dots$, iteration **do**

 Sample from $\pi(\mathbf{l}_k | f(t_1), \dots, f(t_n), \ell^2)$ using Metropolis-Hastings with proposal distribution $Q(\mathbf{l}_k' | \mathbf{l}_k)$

 Find nearest \mathbf{l}_k , from all candidate landmarks \mathbf{l}_m , based on samples

 Sample ℓ^2 from $\pi(\ell^2 | f(t_1), \dots, f(t_n), \mathbf{l}_k)$ using Metropolis-Hastings with proposal distribution $Q(\ell^{2'} | \ell^2)$

 Record \mathbf{l}_k, ℓ^2

end for

```

# Define model
class my_basis_function(nn.Module):
    def __init__(self):
        super(my_basis_function, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(5*8, 128),
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        flatten_x = self.flatten(x)
        mean_and_logstd = self.linear_relu_stack(flatten_x)
        return mean_and_logstd

class my_policy(nn.Module):
    def __init__(self, basis_function):
        nn.Module.__init__(self)
        self.basis_function = basis_function
        self.basis_function.to(device)

    def action_distribution(self, x):
        r_and_logstd = self.basis_function(x)
        r_and_logstd = torch.split(r_and_logstd, 5, dim = 1)
        r = r_and_logstd[0]
        std = torch.exp(r_and_logstd[1])

        cov_matrix = []
        for i in range(x.shape[0]):
            cov_matrix.append( torch.matmul(torch.matmul( torch.diag(std[i]), se_kernel_fast(x[i],x[i]) ),torch.diag(std[i]) ) ) )

        distribution = ptd.multivariate_normal.MultivariateNormal(loc = r,covariance_matrix = torch.stack(cov_matrix) )
        return distribution

```

```

# Define model
class my_basis_function(nn.Module):
    def __init__(self):
        super(my_basis_function, self).__init__()
        self.lstm = nn.LSTM(input_size=len(_input_val)*env.d, hidden_size = 128, num_layers = 1, batch_first=True)
        self.linear = nn.Linear(128, env.d*2)

    def forward(self, x):
        flatten_x = x.flatten(start_dim=2).float()
        output, _ = self.lstm(flatten_x)
        mean_and_logstd = self.linear(output)
        return mean_and_logstd

class my_policy(nn.Module):
    def __init__(self, basis_function):
        nn.Module.__init__(self)
        self.basis_function = basis_function
        self.basis_function.to(device)

    def action_distribution(self, x):
        r_and_logstd = self.basis_function(x[:,env.look_back_period-1,:].float())

        r_and_logstd = r_and_logstd[:,env.look_back_period-1,:]

        r_and_logstd = torch.split(r_and_logstd, env.d, dim = 1)
        r = r_and_logstd[0]
        std = torch.exp(r_and_logstd[1])

        cov_matrix = []
        for i in range(x.shape[0]):
            cov_matrix.append( torch.matmul( torch.matmul( torch.diag(std[i]), se_kernel_fast(x[i,env.look_back_period-1,:],:),
                x[i,env.look_back_period-1,:]) ), torch.diag(std[i]) ) )

        distribution = ptd.multivariate_normal.MultivariateNormal(loc = r, covariance_matrix = torch.stack(cov_matrix) )
        return distribution

```

```

class StockEnv(object):
    """
    ### Description

    """
    def __init__(self):
        self.d = 5
        self.observation_space = [-4.5,4.5]
        self.look_back_period = 3
        self.state = None
        self.year = 2010
        self.transformer = None
        self.available_years = available_years
        self.available_days_train = available_days
        self.available_days_test = available_days
        self.train_df = None
        self.test_df = None
        self.original_dat = _dat_

    def update_year(self,year):
        self.year = year
        look_back_period = self.look_back_period
        _dat_ = self.original_dat

        train_df = _dat_[pd.DatetimeIndex(_dat_['date']).year < year ].copy()
        available_days_train = list(set(train_df['date']))
        available_days_train.sort()
        cut_date_ = available_days_train[-look_back_period+1]

        test_df = _dat_[_dat_['date']>= cut_date_].copy()
        test_df = test_df[ pd.DatetimeIndex(test_df['date']).year <= year ].copy()

        _train_X = np.asarray(train_df[_input_val])
        X_transformer = PowerTransformer(method='yeo-johnson', standardize=True).fit(_train_X)
        _train_X = X_transformer.transform(_train_X)
        # train_X = np.nan_to_num(_train_X)

```

```

available_days_train = list(set(train_df['date']))
available_days_train.sort()

available_days_test = list(set(test_df['date']))
available_days_test.sort()

self.transformer = X_transformer
self.train_df = train_df
self.test_df = test_df

self.available_days_train = available_days_train
self.available_days_test = available_days_test

def reset(self, training):
    if training:
        available_days = self.available_days_train
        _df = self.train_df
        # _df = train_df
    else:
        available_days = self.available_days_test
        _df = self.test_df
        # _df = test_df
    look_back_period = self.look_back_period
    d = self.d




    _day = np.random.choice(range(len(available_days)-look_back_period))
    _day = available_days[_day:(_day+look_back_period)]

    available_stocks = _df[_df['date'].isin(_day)].copy()
    _len_stocks = list(set(available_stocks['ticker']))
    _selected_stocks = np.random.choice(_len_stocks, d)
    available_stocks = available_stocks[available_stocks['ticker'].isin(_selected_stocks)].copy()

    output = []
    for i in range(look_back_period):
        day_available_stocks = available_stocks[available_stocks['date'] == _day[i]].copy()
        day_available_stocks.index = day_available_stocks['ticker']

```

References I

-  Chen, Yifan et al. (2021). “Solving and learning nonlinear pdes with gaussian processes”. In: *Journal of Computational Physics* 447, p. 110668.
-  Srivastava, Anuj et al. (2011). “Registration of functional data using Fisher-Rao metric”. In: *arXiv preprint arXiv:1103.3817*.
-  Swanson, Ana (2015). “Kurt Vonnegut graphed the world’s most popular stories”. In: *The Washington Post* 9.