Asset Management: Class Project

Alessandro Stolfo[◇], Emanuele Palumbo[◇], Metod Jazbec[◇]

***Abstract*—Hidden Markov Model and Long-Short-Term-Memory are applied to portfolio optimization task. Performance of both approaches is analysed and compared. While LSTM is shown to be capable of producing on average higher returns, its results are found to be sensitive to changes in the optimization pipeline (length of investing period etc.). On the other hand, HMM produces more conservative returns, however those are found to be more robust with respect to changes in investing hyper-parameters. Additionally, sequence-to-sequence LSTM architecture is considered.**

I. Asset Allocation

A. Optimization

Modern portfolio theory (MPT), introduced by Harry Markowitz in [1], is based on two main concepts:

- Investors aim to maximize returns for given level of risk.
- Risk can be reduced by diversifying a portfolio through individual, unrelated securities.

These two principles lead to an efficient frontier of portfolios, among which the investor is free to choose, assumed that investors are risk-averse, preferring a portfolio with less risk for a given level of return.

Within this framework, portfolio optimization is the process of selecting the best portfolio, according to some objective. Such objective typically involves maximizing factors such as expected return, and minimizing costs like financial risk.
The traditional approach to portfolio optimization consists of weighing risk (expressed as variance) against expected return (expressed as historical mean of the past returns). In the following we present the problem using utility-maximization framework.
Let the vector of asset returns $R \in \mathbb{R}^n$ have a Gaussian distribution $R \sim \mathcal{N}(\mu, \Sigma)$. Mean and variance of the portfolio $\theta \in \mathbb{R}^n$ are given by

$$\mu_p = \theta^T \mu, \qquad \sigma_p^2 = \theta^T \Sigma \theta.$$

Since we want to derive the set of mean-variance efficient portfolios, we want to optimize

$$\theta^T \mu - \frac{\rho}{2} \theta^T \Sigma \theta,$$

where ρ is the constant coefficient of absolute risk aversion. As a result we obtain the family of portfolios described by

$$\theta = v_1(\mu_p) l_{22} \theta_{\text{MinVar}} + v_2(\mu_p) l_{21} \theta^*$$

where $l_{22} = 1^T \Sigma^{-1} 1$, $l_{21} = 1^T \Sigma^{-1} \mu$, $\theta^* = \frac{\Sigma^{-1}\mu}{1^T \Sigma^{-1} \mu}$ and $\theta_{\text{MinVar}} = \frac{\Sigma^{-1} 1}{1^T \Sigma^{-1} 1}$.
We can thus notice that all minimum variance portfolios can be constructed as a combination (depending on μ_p) of only two portfolios. This fact is know as *mutual fund* theorem.

B. Shrinkage of Σ

The standard statistical method to estimate Σ is to gather a history of past stock returns and compute their sample covariance matrix. However, when the number of stocks under consideration is large, especially relative to the number of historical return observations available (which is the usual case), this method might lead to an inaccurate estimation of such matrix. This kind of estimation error is most likely to perturb the mean-variance optimizer. In its place, we can think of using the matrix obtained from the sample covariance matrix through a transformation called shrinkage.
Denote by $\{\boldsymbol{x}_i\}_{i=1}^n$ a set of i.i.d. Gaussian p-dimensional random vectors. Our goal is to find an estimator $\hat{\boldsymbol{\Sigma}}(\{\boldsymbol{x}_i\}_{i=1}^n)$ which minimized the MSE

$$\mathbb{E}\left[\left\| \hat{\boldsymbol{\Sigma}}(\{\boldsymbol{x}_i\}_{i=1}^n) - \boldsymbol{\Sigma} \right\|_F^2 \right].$$

As stated above, the classical estimator of $\boldsymbol{\Sigma}$ is the sample covariance matrix

$$\hat{\boldsymbol{S}} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{x_i} \boldsymbol{x_i}^T.$$

This estimator is unbiased. However, it does not necessarily achieve low MSE due to its high variance and it is usually ill-conditioned for large problem. Now consider a naive but most well-conditioned estimate for $\boldsymbol{\Sigma}$

$$\hat{\boldsymbol{F}} = \frac{\text{Tr}(\hat{\boldsymbol{S}})}{p} \boldsymbol{I}.$$

this "structured" estimate will result in reduced variance with the expense of increasing the bias. A reasonable trade-off between low bias and low variance is achieved by shrinkage of $\hat{\boldsymbol{S}}$ towards $\hat{\boldsymbol{F}}$, resulting in the following estimator

$$\hat{\boldsymbol{\Sigma}} = (1 - \hat{\rho}) \hat{\boldsymbol{S}} + \hat{\rho} \hat{\boldsymbol{F}},$$

characterized by the shrinkage coefficient $\hat{\rho}$.
Our goal thus becomes to find a shrinkage coefficient $\hat{\rho}$ that minimizes the MSE.

◇ Data Science Masters, ETH Zurich

The optimal ("oracle") estimate of $\hat{\mathbf{\Sigma}}$, and thus the optimal estimate of $\hat{\rho}$ is the solution to

$$\begin{aligned}&\text{minimize } \mathbb{E}\left[\left\|\hat{\mathbf{\Sigma}}(\{\boldsymbol{x}_i\}_{i=1}^n) - \mathbf{\Sigma}\right\|_F^2\right]\\&\text{subject to } \hat{\mathbf{\Sigma}} = (1-\rho)\hat{\boldsymbol{S}} + \rho\hat{\boldsymbol{F}},\end{aligned}$$

and therefore cannot be generally implemented, as it depends on the unknown $\mathbf{\Sigma}$.
Let us then consider two different approaches to approximate the optimal shrinkage coefficient.

1) The Rao-Blackwell Ledoit-Wolf (RBLW)

The Rao-Blackwell LW (RBLW) estimator is

$$\hat{\mathbf{\Sigma}}_{RBLW} = (1-\hat{\rho}_{RBLW})\hat{\boldsymbol{S}} + \hat{\rho}_{RBLW}\hat{\boldsymbol{F}},$$

where

$$\hat{\rho}_{RBLW} = \frac{\frac{n-2}{2}\operatorname{Tr}(\hat{\boldsymbol{S}}^2) + \operatorname{Tr}^2(\hat{\boldsymbol{S}})}{(n+2)[\operatorname{Tr}(\hat{\boldsymbol{S}}^2) - \frac{\operatorname{Tr}^2(\hat{\boldsymbol{S}})}{p}]}.$$

2) The OAS Estimator

The idea behind the OAS is to approximate this oracle via an iterative procedure. We initialize the iterations with an initial guess of $\mathbf{\Sigma}$ (e.g. the sample covariance) and iteratively refine it. The limit, is the the following OAS solution

$$\hat{\mathbf{\Sigma}}_{OAS} = (1-\hat{\rho}^*_{OAS})\hat{\boldsymbol{S}} + \hat{\rho}^*_{OAS}\hat{\boldsymbol{F}},$$

where

$$\hat{\rho}^*_{OAS} = min\left(\frac{\frac{1-2}{2}\operatorname{Tr}(\hat{\boldsymbol{S}}^2) + \operatorname{Tr}^2(\hat{\boldsymbol{S}})}{\frac{n+1-2}{p}[\operatorname{Tr}(\hat{\boldsymbol{S}}^2) - \frac{\operatorname{Tr}^2(\hat{\boldsymbol{S}})}{p}]}, 1\right).$$

C. Estimation of μ and the James-Stein Estimator

While the last section focused on the estimation of the covariance Σ, this section will deal with estimating the mean vector μ. This is a delicate part of the Portfolio Selection problem as the composition of the optimal portfolio is extremely sensitive to changes in the means. As the data are supposed to come from d independent Gaussian distributions, the sample mean:

- is the ML estimator for the mean
- is an unbiased estimator for the mean
- has minimum variance among all unbiased estimators for the mean

It seems then reasonable to choose the sample mean as our estimator for μ. Note that we evaluate an estimator $\hat{\mu}$ of μ in terms of its risk R, that is computed as the mean squared error $MSE(\hat{\mu})$:

$$R = MSE(\hat{\mu}) = \mathbb{E}\left\|\hat{\mu} - \mu\right\|_2^2$$

However, in a remarkable paper [2], Stein showed that, when dealing with a d-variate Gaussian distribution, with $d > 2$, estimating each mean with its own average (sample mean) is not an admissible choice. For the purpose of the explanation we consider a setting with a d-dimensional Gaussian vector $(X_1, \ldots, X_d)$ and a single observation X. It is easy to extend the result to n observations. The ML, unbiased, minimum variance estimator in this case would be X itself. Note $R(\hat{\mu}_{ML}) = R(X) = \sigma^2 d$
Suppose we have an estimate $\hat{\mu}(X)$ of μ of the form

$$\hat{\mu}(X) = f(X) + X$$

where $f : \mathbb{R}^d \to \mathbb{R}^d, f \in \mathcal{C}^1$.
Recall that X here is the ML estimator for μ, and as we will see later, we can interpret f(X) as an *adjusting* or *shrinking* factor.
We can find then the following decomposition for the risk R:

$$\begin{aligned}R = MSE(\hat{\mu}) = \mathbb{E}\left\|\hat{\mu} - \mu\right\|_2^2 = \mathbb{E}\left\|f(X) + X - \mu\right\|_2^2 =\\= \mathbb{E}\left\|f(X)\right\|_2^2 + \sigma^2 d + 2\sum_{i=1}^d f_i(X)(X_i - \mu_i) \qquad (1)\end{aligned}$$

From multivariate Stein's lemma [3], we have that for $X \in \mathbb{R}^d$, $X \sim N(\mu, \sigma)$ and $f : \mathbb{R}^d \to \mathbb{R}^d, f \in \mathcal{C}^1$ with $\mathbb{E}f'(X) < \infty$ it holds:

$$\forall i \qquad \mathbb{E}[f_i(X)(X-\mu)] = \sigma^2\mathbb{E}[\nabla f_i(X)].$$

Note that to each i term in the sum in (1) we can apply Stein's lemma and we get

$$R = \mathbb{E}\left\|f(X)\right\|_2^2 + \sigma^2 d + 2\sigma^2\sum_{i=1}^d \mathbb{E}[\frac{\partial f_i(X)}{\partial X_i}]. \qquad (2)$$

Let the **James-Stein estimator**[4] be

$$\hat{\mu}_{JS} = -\frac{(d-2)\sigma^2}{\left\|X\right\|^2}X + X$$

Using (2) we get

$$R(\hat{\mu}_{JS}) = \sigma^2 - \sigma^4(d-2)\mathbb{E}\left[\frac{1}{\left\|X\right\|^2}\right] \quad < \sigma^2 d$$

We showed then the inadmissibility of $\hat{\mu}_{ML}$: $\hat{\mu}_{JS}$ should instead be used.
Note again that using James-Stein estimator can be seen as taking the natural estimator X and use shrinking. In addition note that each estimate $\hat{\mu}_{JSi}$ of μ_i is computed using the entire vector X, i.e. the realizations of all the d variables, which is surprising and counterintuitive as the d variables are assumed to be independent.

D. Bayesian approach in estimating μ

One can also have a Bayesian updating approach in estimating μ. For the purpose of the explanation, we will consider a single asset (not a vector) and we will estimate its mean μ. Namely, we have a (Gaussian) prior for μ with mean μ_0 and variance σ_0 and some data, i.e. N observations x_i that come from $X \sim N(\mu, \sigma^2)$, where μ, σ^2 are the true mean and variance. Note that σ is supposed to be known. Let $\bar{X} = \frac{1}{N}\sum_{i=1}^N x_i$ be the average of the observations. The Bayes approach consists of using as estimator the mean of the

posterior distribution. In this case the expression for the **Bayes estimator** for μ is:

$$\hat{\mu}_B = \mu_0 \frac{\frac{1}{\sigma_0^2}}{\frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}} + \bar{X} \frac{\frac{N}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}}$$

Note that the resulting estimator is a weighted sum of the prior and the average of the observations. A Bayesian approach could be a smart choice as the prior can be thought of as a regularization term.

II. Financial Markets

Working in the financial market means dealing with a system that is characterized by a lot of parameters, on whose values there is a considerable amount of uncertainty. People that work in the financial market try to gain information on it by developing convenient ways to estimate parameters from past data. There are other examples of dynamical systems governed in (at least a partial amount) by non-deterministic laws, in which parameter estimation has an important role: one example could be climate systems. There is one big difference though when dealing with financial market, instead of a climate system. Indeed, learning and estimating parameters from a climate system will obviously not affect the dynamic of the system itself. Conversely, the agents that act in the finance market and influence it, are at the same time learning and estimating parameters. It is obvious then that learning influences the dynamic of financial markets in a sort of 'self-referential' way. The effects of learning in the financial market is the main topic in [5] paper. An example of one, at first sight puzzling, phenomenon in financial markets, that is presented in the paper as an effect of learning, is the difference between the stock returns volatility and the volatility of the underlying dividends.

If the discout rate *r* and the average divident growth *g* are constant and known the **Gordon growth model** gives the stock price P. In this case, the stock return volatility is the volatility of the underlying dividend growth. In the real case though, *g* is not known but estimated: this means that high dividends not only affect directly the stock price via their current value, but they also raise expectation on future dividends (affecting the estimated value of *g*). Pastor, Veronesi [5] describe this as 'double kick' to the stock price and highlight how this explains how stock returns volatility tends to be higher (even by a significant percentage) compared to the volatility of the underlying dividends.

III. Machine Learning

Quality of estimation/prediction of expected return vector μ has a profound impact on the performance of Mean Variance Optimization (MVO) procedure. In the following we present two techniques that were disscused in class. Denote by $\mathcal{X} = \{x_1, \dots, x_T\}$ dataset of asset returns. [1]

A. Hidden Markov Model

We introduce latent first-order Markov process $z_{1:T}$. that is assumed to generate observed data points $x_{1:T}$. Additional key assumption is that x_t is conditionally independent of the past, i.e. $x_{1:t-1}$, given current hidden state z_t. Joint probability distribution can thus be restated as

$$P(x_{1:T}, z_{1:T}) = P(z_1) \prod_{t=2}^{T} P(z_t|z_{t-1}) \prod_{t=1}^{T} P(x_t|z_t).$$

In our case we can think of hidden states as market regimes, thus we model z_t as discrete random variable with support $\{1, \dots, K\}$. If we define transition matrix [2] as

$$A := \begin{bmatrix} P(z_t = 1|z_{t-1} = 1) & \dots & P(z_t = K|z_{t-1} = 1) \\ \vdots & \ddots & \\ P(z_t = 1|z_{t-1} = K) & \dots & P(z_t = K|z_{t-1} = K) \end{bmatrix},$$

we see that state space equation can be written as $P(z_t) = A^T P(z_{t-1})$. For so-called emission probabilities $P(x_t|z_t)$, Gaussian distributions are most often used. To learn the parameters of the model $\theta = \{\Pi, A, \mu_1, \Sigma_1, \dots, \mu_K, \Sigma_K\}$[3] expectation-maximization (EM) algorithm is deployed, where the objective is to maximize the data log-likelihood $\log P(\mathcal{X}, \mathcal{Z}|\theta)$ w.r.t. θ. In the E-step we determine membership probabilities $P(z_t|x_t)$, which we then use in M-step to find optimal θ (e.g. using MLE techniques). We iterate between the two steps untill convergence[4]

Idea behind this is to segment the data into K-clusters (market regimes) and then only use data points withing a single cluster to update given parameters μ_j and Σ_j. MLE does this implicitly by taking weighted averages of all data points (weighted w.r.t. membership probabilities obtained in E-step).

One way to come up with the final estimate of the return vector $\hat{\mu}$ for portfolio optimization[5], would be to use learned parameters of the model to first predict distribution over the next hidden state $P(z_{T+1}) = A^T P(z_T)$ and then obtain $\hat{\mu}$ as

$$\hat{\mu} = \sum_{j=1}^{K} \mu_j P(z_{T+1} = j).$$

By doing so, we are putting most weight to the mean vector of the market regime that we believe is most likely to take place in the economy next. One possible caveat here is

[1] Note that each $x_i \in \mathbb{R}^N$, where N represents number of assets in our portfolio.

[2] Such matrix is row stochastic, i.e. all elements are non-negative and entries in each row sum up to 1.

[3] $\Pi = P(z_1)$ is the prior distribution.

[4] Note that multiple local maxima exists.

[5] Approach where we model $P(x_{T+1}|\mathcal{X})$ directly is ommited here for the sake of brevity (to explain equation on the intuitive level we would have to briefly introduce forward-backward algorithm, which we used in the lectures to perform smooting inference in HMM, i.e. calculating $P(z_t|\mathcal{X}), \forall t$).

that we are predicting one time-step into the future, whereas difference between re-balancing dates in our MVO pipeline can be much bigger (`window` parameter in the code skeleton). Hence we are making assumption that predicted distribution over market regimes for the next time-step is representative for the entire period up to the next re-balancing date. To get around this issue, we could for example model instead $P(z_{T+N}) = A^{T^N} P(Z_T)$ for some $1 < N \leq$ `window`.

B. LSTM

Long-short-term-memory (LSTM) is an instance of the recurrent neural network (RNN) architecture. In contrast to feed-forward networks, RNNs have additional input/output in the form of the hidden state at each time step, with which relevant time dependent information is propagated through the network. This makes RNNs more suitable to process temporal sequence data, be it text, videos or in our case vector of asset returns. The difference between different version of RNNs (vanilla RNN, LSTM, GRU) is in the architecture of the cell unit.

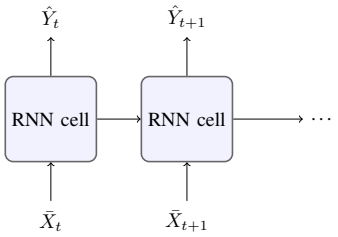

Fig. 1. Many-to-many RNN architecture.

Since dimension of the outputs of the cell depends on the number of hidden nodes in the cell, we project those to the desired dimension using linear layer[6]. Our objective thus becomes to find $\underset{\theta}{argmin} \sum_{i=1}^{N} loss(y_i, \hat{y_i})$, where θ denotes the parameters of the neural network and y_i are true labels. This is done using back-propagation through time (gradient-based technique especially suitable for recurrent architectures).

Note that contrary to HMM, we have a supervised setting here, so we need to somehow come up with the labels y. It is also not that straightforward what to feed to the network as input x at each time step. Since in MVO our goal is to find $\hat{\mu}$ for the next investing period, one reasonable choice would be $y_t = mean(y_t, \ldots, y_{t+\texttt{future}})$, where `future` is the period between the two re-balancing dates, for labels and $\bar{x}_t = [x_{t-N}, \ldots, x_t]$, i.e. concatenation of last N data points, as inputs for each t in our training set.

One of main challenges in fitting LSTM (or other deep learning approaches) is the high number of design choices that we have to make (number of layers, how to come up with x and y, whether to include linear layers or not, choice of an optimizer, loss function etc.). How well we make those decisions, can sometimes make a difference between a successful and unsuccessful method. Also since in our case we do not have plenty of training data (single price trajectory for each asset), we should be aware that we are prone to over-fitting.

[6]Same can be done for inputs, i.e. we can first embed the inputs to the hidden dimension with linear layer and then pass those to the RNN cell.

IV. Experiments

A. Evaluation pipeline

For backtesting/comparing different methods we used the rolling window approach, i.e. at each rebalancing date past data was used to calculate new weights for the next investing period (instead of doing the more traditional train-validation-test data split).

We used three different metrics for comparison:

- **Sharpe ratio:** The Sharpe Ratio for the portfolio p is defined as
$$\mathbf{SR}(p) = \frac{R_p - R_f}{\sigma_p}$$
where $\bar{R}_p$ is the average portfolio return, R_f is the risk-free asset (assumed to be zero in our case) and σ_p is the standard deviation in portfolio's return.
- **Annualized return:** Annualized return of a portfolio p is calculated as
$$\mathbf{AR}(p) = (1 + CR(p))^{\frac{252}{D}} - 1$$
where D is the number of days the portfolio was held and $CR(p)$ is the cumulative return over the D days.
- **Maximum drawdown:** Maximum drawdown metric for portfolio p up to time T is calculated as
$$\mathbf{MD}(p) = \max_{t_1 \in (0,T)} \left[\max_{t_2 \in (0,t_1)} R_p(t_2) - R_p(t_1) \right]$$
where $R_p(t)$ is the portfolio return at time t. It is the maximum difference between rolling peak and current value.

B. Baselines

The following two baselines were also considered:

- **Equal Weights.** Each available asset is allocated an equal fraction of wealth. This naive diversification rule is often use as a baseline investment model and, in spite of its simplicity, has been shown to be capable of decent performances. In particular, in presence of high estimation error of parameters, it might perform better than more sophisticated models.
- **Sample Mean.** The mean vector used in the optimization process is computed as sample mean of the data points belonging to the previous window.

C. Approach

Our approach was based on the testing and evaluation of the different models we have seen during the course, focusing more closely on HMM and LSTM.

1) Hidden Markov Model: Two different versions of HMM were tested. First variant is an instance of the plain algorithm, where we fit the model on the given data of the 14 assets. In the second approach, we first fit the model on dataset consisting of 4 assets, that we thought are more representative of the regimes present in the economy: volatility index ("VIX"), the "S&P500" index, the "DAX" index and the price of the U.S. 5-year Treasury bond ("US5Y"). Obtained annotations of the past dates are used to segment initial dataset (14 assets) into separate clusters. In both versions, only data belonging to the next predicted most likely state/regime is used to compute estimate of the mean vector.

2) LSTM: We used plain many-to-many LSTM architecture and kept the design of inputs and labels as proposed in the lectures (i.e. $\bar{x}_t = [x_{t-N}, \ldots, x_t]$, $y_t = mean(y_t, \ldots, y_{t+\texttt{future}})$). To combat small size of the training data in the beginning of our evaluation pipeline, we used "warm-up" period of approximately 10 years. This way we ensured that LSTM had enough data to learn something meaningful already at first re-balancing date. Also, we did not retrain the model from the scratch at every re-balancing date, but we only fine-tuned the weights using the data from the last window.

After fine-tuning the hyper-parameters (number of layers, cell size, parameters from inputs/labels design) we noticed that LSTM is capable of producing promising results, however those seem to be very dependent not only on LSTM's own hyper-parameters, but also on parameters from the optimization pipeline, namely `window` (i.e. difference between two re-balancing dates) and length of the "warm-up" period.

Another thing that caught our attention (and that perhaps raises some concerns regarding the interpretability of the model) was lack of connection between `window` parameter from optimization pipeline and `future` parameter from labels design. Intuitively, values for both should be approximately the same (since the goal is to obtain an estimate of mean returns over entire period until next re-balancing date), however we observed that smaller values for `future` parameter yield better results.

In addition we tried out sequence-to-sequence model using encoder-decoder LSTM architecture. Here, input sequence $x_{t-M}, ..., x_t$ is first fed to the encoder in order to obtain the last hidden state, which can be thought of as representation of the history up to now. This state is then used to initialize decoder, with which we generate predictions $\hat{x}_{t+1}, ..., \hat{x}_{t+N}$. To come up with the final estimation of the mean vector, we simply take the mean of predictions generated by decoder. During training, the so called "teacher-forcing" is deployed, where we feed decoder ground-truths $x_t, ..., x_{t+N-1}$ instead of its own predictions.

Model	Sharpe Ratio	Max. D.	Ann. Return
Sample mean	0.085	0.961	2.1%
Equal weights	0.426	0.356	3.8%
James-Stein	0.441	0.254	2.7%
HMM	0.719	0.112	2.8%
HMM (with reg.)	0.738	**0.105**	2.8%
LSTM	**0.842**	0.150	**7.9%**
LSTM (seq2seq)	0.275	0.949	3.5%

TABLE I
INVESTING PERIOD FROM 2004-12-31 TO 2015-12-28, WEEKLY DATA.

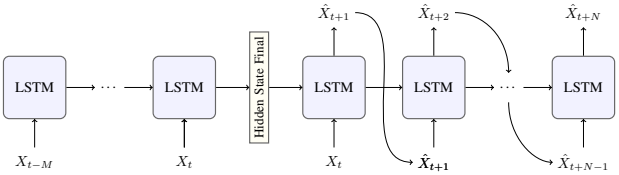

Fig. 2. Sequence-to-sequence architecture (prediction pipeline). M denotes the length of encoding sequnce, N the length of prediction/decoding interval.

Even though this approach was shown to achieve state-of-the-art results when it comes to multivariate time series forecasting [6] (which is essentially the goal of the proposed pipeline), the results we obtained were worse than the ones with many-to-many architecture. Often the model was not even able to outperform baselines (i.e. sample mean, equal weights). Reason for poor performance might be small dimension of the considered time process (14 trade-able assets, whereas successful implementations often consider series with more than 1000 features, i.e. there are more connections that model can exploit), as well as lack of training and fine-tuning skills on our side (when it comes to neural network architectures).

Implementation details for reported models: To design inputs and labels for plain LSTM we used values $N = 10$ and $\texttt{future} = 20$, whereas for sequence-to-sequence model length of encoder input sequence was $M = 50$ and length of decoder sequence was $N = 10$. Both models were trained using ADAM optimizer, with initial learning rate of 0.0001. Cell size was set to 15 (number of hidden nodes). In plain LSTM we also tried increasing number of layers, and we found that having 2 stacked layers helps improving the overall performance.

D. Results

In table I results for different models for investing period from 2004-12-31 to 2015-12-28 are shown. Data was considered on a weekly basis. Weights are recalculated every 52 weeks, using data from the past 52 weeks. For LSTM models, we additionally used aforementioned "warm-up" period, i.e. at first re-balancing date the entire past data was used to fit the model.

As already mentioned, the promising results of LSTM should be taken with the grain of salt. We noticed that they are extremely sensitive to the start of investing period, see table II.

In HMM we also considered data on a daily basis and with shorter re-balancing period (since the assumption that next

Start of investing period	Sharpe Ratio	Max. D.	Ann. Return
2008	0.803	0.139	9.7 %
2010	0.232	0.302	3.2%
2012	0.452	0.161	5.2 %
2013	-0.189	0.256	-2.9 %
2014	0.402	0.053	3.7%

TABLE II
RUNS OF LSTM MODEL FOR DIFFERENT STARTS OF INVESTMENT PERIODS.

predicted state will persist over the entire span of next 52 weeks might be a bit unrealistic), see table III.

For HMM we mostly reused the code skeleton given to us at the end of the course. While running the experiments we noticed that HMM returns were often low, despite Sharpe ratio being quite high. Looking closer at the code we noticed that the issue was in the function `by_mean`, that was used for calculating means of different states. In the function, Bayesian approach for estimating mean with Gaussian prior is implemented, however prior seems to be too strong (mean of the prior `m_0` is usually around 0.1, whereas most of the returns have values around 0.001). This manifested in HMM producing estimates for mean vector, where all components were almost the same (and equal to prior `m_0`). After accounting for this by either adjusting the mean of the prior or by using sample mean instead, the returns obtained with HMM got higher.

Model	Sharpe Ratio	Max. D.	Ann. Return
Sample mean	0.149	1.136	3.3%
Equal weights	0.334	0.387	3.0%
HMM	0.252	0.932	5.2%
HMM (with reg.)	**0.369**	0.585	**5.6%**

TABLE III
RESULTS FOR DIFFERENT MODELS FOR INVESTING PERIOD FROM 2004-12-31 TO 2015-12-28. DATA CONSIDERED ON DAILY BASIS. WEIGHTS ARE RECALCULATED EVERY 12 WEEKS (60 BUSINESS DAYS), USING DATA FROM THE PAST 104 WEEKS (520 BUSINESS DAYS)

V. CONCLUSION

After extensive experiments with variants of both LSTM and HMM models, we decided to stick with the latter as the final model. Even though LSTM usually obtained higher annualized returns, we deemed the results too dependent on the parameters of the mean-variance optimization pipeline to consider the model fully trustworthy. Hence we decided to go with more conservative, yet more stable approach of HMM.

We would point of two possible areas of improvement. First one would be coming up with the better evaluation/backtesting pipeline, with the goal of getting results that are less susceptible to changes in the parameters of the optimization engine. Second one would be to put more attention into making LSTM produce more stable results (performing multiple train-validation data splits, deploying techniques to combat possible over-fitting issues like drop-out or regularization, using exponential decay for learning rate, standardizing data etc.).

CODE

The final model we have chosen is implemented in the "`final_model.py`" file. The file also comes with a `main` that runs the whole optimization pipeline, prints the computed metrics and plots the pnl curve. In order to execute it from the command line, run

```
python final_model.py [filename] [start_date] [end_date]
```

where `filename` is the .csv file containing the stock prices and should have the same format as the file "markets_new.csv", and `start_date` and `end_date`, which have the format YYYY-MM-DD, are respectively the dates of the first and of the last day you want to take into account in the optimization process (note that the actual first trading day is `window` days after `start_date`).
The file `optimize_pf.py` contains the pipeline the we used to run the different models on the provided data.

REFERENCES

[1] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, pp. 77–91, 1952.
[2] C. Stein, "Inadmissibility of the usual estimator of the mean of a multivariate normal distri- bution," *Proceeds of the Fourth Berkeley Symposium*, vol. 1, p. 197206, 1956.
[3] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *Ann. Statist.*, vol. 9, no. 6, pp. 1135–1151, 1981.
[4] C. James, W. Stein, "Estimation with quadratic loss," *Proceeds of the Fourth Berkeley Symposium*, vol. 1, p. 361380, 1961.
[5] L. Pastor and P. Veronesi, "Learning in financial markets," *Annu. Rev. Financ. Econ.*, vol. 1, no. 1, pp. 361–381, 2009.
[6] F. . G. J. Salinas, D. Flunkert, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *arXiv*, 2019.