MASTER

Portfolio Analysis
Estimation of VaR Using Mixed Sum Product Networks

Einarsson, Arngrimur

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Uncertainty in Artificial Intelligence Group

Portfolio Analysis: Estimation of VaR Using Mixed Sum Product Networks

Supervisors:
Robert Peharz.

Final Version

Eindhoven, September 2021

Abstract

In this project we put forth a new approach to quantify a well known risk metric called Value-at-Risk (VaR). VaR is a statistical technique used to measure the amount of potential loss that could happen in an investment portfolio over a specified period of time. There are well established VaR models that have been used (in some form) in the industry for many years: the Normal Linear model, the Monte Carlo model and the Historical Simulation model. These models do however have drawbacks, the Normal Linear and Monte Carlo VaR have strong assumptions of normality for the percentage returns of the portfolio and the Historical Simulation depends heavily on data. In this project, a Mixed-Sum-Product Network (MSPN) was applied to estimate the VaR. The MSPN model is able to learn the joint probability distribution of the risk factor returns of a financial portfolio and capture the heavy tails without making any parametric assumptions of the daily returns distribution. However, the MSPN model does assume that the daily returns are independent and identically distributed meaning that each stock in the portfolio has the same probability distribution and all stocks are mutually independent.

By using the MSPN model to learn the joint probability distribution of the daily returns we could obtain possible realisations of the portfolio's returns and estimate the VaR. We executed two experiments to examine how well the MSPN model performed compared to the well established VaR models. In the first experiment, we used stock data from 03/07/2013-20/07/2021 to estimate a 1-day VaR value and the other experiment was a sliding window approach were we split our data into segments and estimated the VaR for each segment. Using statistical backtesting methodologies we were then able to see what VaR estimates were useful and which were not. The results indicate that the MSPN model performs well at high dimensions with abundance of data, especially at the highest (99%) VaR level. In addition, using MSPN inference we demonstrated that the MSPN model was able to capture the heavy tails of the daily returns and is therefore more capable of catching extreme events which is associated with high level VaR estimations. However, with smaller data and a lower VaR level (95%) there was no clear advantage of using the MSPN model. The results indicate that the MSPN is a promising alternative to VaR risk modelling, it is able to capture the heavy tails of the daily returns distribution with no parametric assumptions and provides useful VaR estimates. To conclude, the MSPN model is more sensitive to new market observations than the other models and is more capable of catching rare market movements providing more useful VaR estimates at the highest level (99%).

Acknowledgements

While working on this project I received great support and guidance from friends, colleagues and others.

First I would like to thank my supervisors, David Montalvan Hernandez and Robert Peharz whose expertise and knowledge was highly valued in designing the project, directions of research, gaining insight and interpreting results. Your feedback has pushed me into thinking more critically, putting my thoughts to words and organize my work so that it reaches its fullest potential.

In addition, I would like to thank my girlfriend, Marta Ýr Magnúsdóttir for her counsel, emotional support and sympathetic ear. Finally, I would like to thank my friends and family, who supported me throughout the whole process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For many years, mathematicians and other data scientists have dealt with the task of financial time series predictions. With increasing computing capabilities and progress in machine intelligence, one could assume that the design of accurate prediction models in finance is a manageable task. However, the future trend in the prices of financial instruments contains a lot of uncertainty and thus making much of the effort put in the prediction models of little value [Mariani et al., 2019]. Lately, market volatility has been growing as trading focuses on increasingly complex instruments whose risks are extremely difficult to assess. Risk management and financial securities have been around for centuries, yet we are only getting started to understand how the risks of complex financial products can be quantified realistically, even though these measures make all the difference between success and failure in the financial industry [Alexander, 2008b].

Despite a rapid growth in computing and artificial intelligence, modern portfolio management is largely based on linear models and the Markowitz framework [Markowitz, 1991], better known as Modern Portfolio Theory. The main drawbacks of the well established risk models in portfolio management are very strong and unrealistic assumptions, for example parametric assumptions about the future risk factor returns. To address the limitations of these models we will provide a new generative approach of quantitative risk assessment in portfolio theory by using a special type of a probabilistic graphical model to model the market uncertainty that ultimately drives future prices. More precisely, our model learns the joint probability distribution of the price trends for various sets of financial assets to match the probability distribution of the real market. By using this approach we eliminate the assumptions used in these well established linear risk models, i.e. we do not make any parametric assumptions of the risk factor returns.

In this project, we aim to assess a well known quantitative risk metric called Value-at-Risk by using a probabilistic model called Mixed Sum Product Networks. By using a model of this kind we can learn the behavior of a collection of complex financial instruments to acquire an estimate of the risk associated. To demonstrate how the model performs compared to older well established VaR models we plan to execute experiments on a wide range of stock data from 2013-2021. This data includes stock prices for the top 40 stocks of the S&P 500 index and we will provide VaR risk estimates with different sets of these stocks. We will look at different aspects of how the model performs both by retrieving VaR estimates using a sliding window based approach, where we split our data into smaller segments and estimate the VaR repeatedly with various segments sizes and we will also look at the performance of the model using as much data as possible with different collections of stocks. For further inspection, we aim to estimate very conservative and less conservative VaR to gain a better insight in how the Mixed Sum Product Network model performs at the extreme levels. To assess the VaR estimates retrieved from all models we will use three well established backtesting methodologies, two of these methodologies are statistical methodologies whereas one is a non-statistical method that is a regulatory framework used in the industry and is used for more conservative VaR measures.

This work is split as follows. First, we will motivate precisely why an approach of this kind is relevant and put forth our Research Questions that we aim to answer with this project. Then we will explain in detail all the financial and mathematical background that is relevant for this project and after that we will introduce previous work that is related to risk assessment and machine learning. Next, we will explain the design of our experiments, what data we used, explain the hyperparameters of our model, etc. After explaining our experimental setup we will present the results along with conclusions followed by future work, where we explain what are the next potential directions of research for this project.

Chapter 2

Problem Statement

In this chapter we will argue why and how this project is a promising new way to assess VaR in portfolio theory. We will define what a financial portfolio is, explain how the VaR risk measure is defined, present well known VaR models and their limitations and then explain what are Sum-Product-Networks and Mixed Sum Product Networks and how they are a promising candidate for helping calculating the VaR. Finally, we will present the research questions that we aim to answer in this project.

2.1 Motivation

Portfolios are a collection of financial investments like stocks, bonds, commodities and cash. Although these are the common financial instruments of portfolios, they can also include a wide range of assets including real estate, art and private investments. Due to the different number of assets that can be incorporated in a portfolio, we need a way to assess its behavior (riskiness) by using a summary which is called a risk measure. We can define a market risk measure as the measure of uncertainty in the future value of a portfolio, i.e. the measure of uncertainty in the profits and loss of a portfolio [Alexander, 2008b]. In this project, we focus in on one of the most popular risk measure called Value-at-Risk (VaR). Value-at-risk has been around since the mid 1990s, and almost all of financial institutions use some form of VaR [Alexander, 2008b]. It is a statistical measure of the riskiness of financial portfolios of assets and is defined as the maximum amount expected to be lost over a given time horizon, at a pre-defined confidence level. For example, if we have a 1-day 95% VaR of 1 million USD for a portfolio, then there is 95% confidence that over the next day the portfolio is not expected to lose more than 1 million USD.

The VaR risk metric can be calculated using different techniques. There are three widely known VaR models; the Normal Linear VaR model, the Historical Simulation model and the Monte Carlo VaR model. However, these models have their limitations. The Normal Linear VaR model, often called the Variance-Covariance model, can only be applied to linear portfolios and can only be generalized to a few simple parametric forms. The Historical Simulation VaR model assumes that all possible future variation has been experienced in the past which imposes a strict requirement on the data and the Monte Carlo VaR models require a lot of computational power and can yield considerable simulations errors if the sampling strategy is not well thought of.

From this we can imagine that there is room for alternative approaches to estimate the VaR for portfolios, methods that do not have similar assumptions and restrictions as the other three models, which are most often not realistic. In this study, we will investigate how a particular kind of Sum Product Networks, named Mixed Sum Product Networks, can be applied to help estimate the VaR. Sum Product Networks are a prominent type of deep probabilistic models as they are a flexible representation for high-dimensional distributions [Trapp et al., 2019]. They are related to other better known probabilistic graphical models like Bayesian networks. Sum Product Networks are similar to neural networks to some extent and can tackle the same types of problems, such as natural language understanding and image processing. Because of their ability to represent high-dimensional distributions they are a good candidate for estimating the probability distributions of portfolio returns. Mixed Sum Product Networks are a mixture of Sum Product Networks and nonparametric probability distributions that require no parametric assumptions to learn the portfolio's return distribution. By learning this distribution of returns we can acquire an estimation of the VaR measure. In addition, a Mixed Sum Product Network model can ideally capture the extreme events (high profits and low losses), which occur at the tails of the portfolio's return distribution, providing a better estimate of the VaR measure than the other models at higher confidence levels.

2.2 Research Questions

The aim of this thesis is to estimate VaR with Mixed Sum Product Networks, namely:

> *RQ1: Is it possible to calculate the 1-day VaR of a financial portfolio using an MSPN by learning the joint probability distribution of the portfolio's risk factor returns?*
>
> *RQ2: How does it compare with other well-known VaR models?*

Chapter 3

Background

In this chapter we will look at the concept of risk measures, how VaR is defined in detail, we will look at other concepts like backtesting and explain three well-known VaR models and how they calculate VaR. In addition, we will define and explain Sum Product Networks, in particular Mixed Sum Product Networks and why they are a good candidate for this kind of problem.

3.1 Basics of Portfolio Theory

A portfolio is defined as a collection of financial investments like stocks, bonds, commodities and cash. Let's consider a simple portfolio with k assets and denote the i-th asset price at time t by $p_{i,t}$. At time 0 a certain amount is invested in each of these assets, so that the portfolio contains a unit amount n_i of asset i, for $i = 1, ..., k$. The set $\{n_1, n_2, .., n_k\}$ is defined as the vector of portfolio holdings at time 0. If the portfolio is rebalanced, then the holdings in certain financial assets may change, but without that the portfolio holdings are constant over time.

If there is no rebalancing over time, then at any time $t > 0$, the value of the portfolio is the sum of the product of asset prices and holdings,

$$P_t = \sum_{i=1}^{k} n_i p_{i,t} \tag{3.1}$$

The proportion of capital invested in a certain asset i at time t is called the portfolio weight on this asset. The portfolio weight on asset i is

$$w_{it} = \frac{n_i p_{i,t}}{P_t} \tag{3.2}$$

The return of a portfolio is calculated with the formula,

$$R = \sum_{i=1}^{k} w_i R_i \tag{3.3}$$

where k is the number of assets, w_i is the portfolio weight on asset i and R_i denotes the one-period return on asset i.

In addition, we also need to define the profit and loss (P&L) on an asset, which is the change in price of an asset over a period of time,

$$\text{P\&L}_{t,t+h} = \frac{B_{t,t+h}P_{t+h} - P_t}{P_t} = \frac{P_{t+h} - P_t}{P_t} \tag{3.4}$$

where $B_{t,t+h}$ is the discount factor assumed to be 1 for $h = 1$. Our portfolio P&L returns therefore illustrate how the total value of our portfolio will change, based on a percent change in the underlying prices.

3.2 Risk Measures

This section is largely based on [Artzner et al., 1999], where desirable properties of market risks are discussed and justified. We will make a brief summary of the concepts presented in that paper.

We can think of a risk measure as a mapping from spaces of probability distributions to real numbers but it is important to present a formal definition of a risk measure and what are the properties they should satisfy.

There is a natural way to define a risk measure for a "reference instrument" by describing how close or far a position is from acceptance by the regulator. Let us define Ω as the set of states of nature and we make the assumption that it is finite. Then let $\mathcal{X}$ be the set of all risks, i.e. the set of all real valued functions $X \in \mathcal{X}$, which represent the final net worth of an instrument, or of a portfolio of instruments, for each element of Ω.

Definition 3.2.1 (Risk Measure)**.** A risk measure $\rho(X)$ is a mapping from $\mathcal{X}$ into $\mathbb{R}$.

A measure of risk enables us to express the riskiness of a position with a single number. As the positions are more risky, the higher its measure of risk is and vice versa. When $\rho(X)$ is positive, the number assigned by the measure ρ to the risk X will be interpreted as the amount of capital, m, that is needed to make the risky position X an acceptable position. Roughly, an acceptable position, is a position that meets the capital requirements established by some regulator. If the risk measure $\rho(X)$ is negative, the cash amount $-\rho(X)$ can be pulled out from the already acceptable position and invested in a more profitable way. Therefore, we can conclude that this concept of risk measure is related to acceptability.

We can state that sets of acceptable future net worths are of primary importance and need to be considered when describing allowance or rejection of a risky position. An acceptance set $\mathcal{A}$ is a class of final net worths accepted by a regulator. The families of acceptable positions depend on the tolerance of the regulator and each risk identifies an acceptance set of admissible positions.

Definition 3.2.2. The acceptance set associated with a risk measure ρ is the set denoted by $\mathcal{A}_\rho$ and defined by

$$\mathcal{A}_\rho = \{X \in \mathcal{X} | \rho(X) \leq 0\} \tag{3.5}$$

Definition 3.2.3. For a given rate of return r of a reference instrument, the risk measure associated with the acceptance set $\mathcal{A}$ is the mapping from $\mathcal{X}$ to $\mathbb{R}$, denoted by

$$\rho_{\mathcal{A},r}(X) = \inf\{m | m \cdot r + X \in \mathcal{A}\} \tag{3.6}$$

From Definition 3.2.3 we can conclude that a measure of risk for an unacceptable position is interpreted as the minimum extra capital, defined by m, that is needed to invest in the reference instrument, for letting the future value of the modified position to be acceptable.

Another concept introduced in the paper is the notion of coherent risk measures and is a fundamental concept related to the acceptability of risk measures.

Definition 3.2.4. A risk measure is coherent if it satisfies the following axioms:

Axiom 1. Translation Invariance For all $X \in \mathcal{X}$ and for all $m \in \mathbb{R}$, we have

$$\rho(X + m) = \rho(X) - m \tag{3.7}$$

Axiom 2. Sub-Additivity For all $X_1 \in \mathcal{X}$ and $X_2 \in \mathcal{X}$, we have

$$\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2) \tag{3.8}$$

Axiom 3. Positive Homogeneity For all $X \in \mathcal{X}$ and for all $\tau > 0$, we have

$$\rho(\tau \cdot X) = \tau \cdot \rho(X) \tag{3.9}$$

Axiom 4. Monotonicity For all $X_1 \in \mathcal{X}$ and $X_2 \in \mathcal{X}$ with $X_1 \leq X_2$, we have

$$\rho(X_1) \geq \rho(X_2) \tag{3.10}$$

The first Axiom of Translation Invariance illustrates that adding (or deducting) a risk-free amount to a portfolio and investing it in the reference instrument results in a decrease of the risk of the position by exactly the same amount. The Axiom of Sub-Additivity is a simple reminder of diversification theory where a portfolio made up by several assets is strictly less risky than a portfolio composed by a single instrument.

In this sense, we can assert that the sub-additivity sets an upper bound to the risk of a portfolio and thus to the amount of regulatory capital needed for allocation. The Axiom of Sub-Additivity captures the essence of how a risk measure should behave, especially in the aggregation of portfolios, and because of that it is the key feature of a risk metric. Axiom 3 of Positive Homogeneity explains that if (for instance) the exposure to a specific position doubles then the risk measure that is related to the position should double as well. However, in the case that the position size directly influences risk, we should account for any possible repercussion (e.g. difficulty in liquidate the position), and therefore we might expect the risk to more than double. To conclude, the Axiom of Monotonicity explains that if a position X_2 always performs better than X_1 then the risk associated to X_1 should be higher than that related to X_2 [Roccioletti, 2015].

3.3 Value-at-Risk

This section is largely based on [Alexander, 2008a]. In this book, she explains the concept of VaR thoroughly and the three VaR models: Normal Linear VaR, Monte Carlo VaR and Historical Simulation VaR.

We can define VaR as the loss we are fairly sure will not be exceeded if the current portfolio is held over some period of time. VaR has two basic parameters:

1. *Significance level* α (or confidence level $1 - \alpha$).
2. *Risk horizon* h, the period of time (that is most often measured in days) in which VaR is to be measured.

The significance level is often set by an external regulator, otherwise the significance level for the VaR depends on the attitude to risk of the user. As the user is more conservative, the lower the value of α he will use obtaining a higher confidence level.

The risk horizon is the period in which we measure the potential loss. Different risks are measured over different time periods according to their liquidity. Traders of liquid positions that are operating under VaR limits require real-time, intra-day VaR estimates to assess the effect of any proposed trade on their current level of VaR. The more liquid the risk, the shorter the time period over which the risk needs to be assessed. Liquid risks have the tendency to evolve rapidly and it is difficult to represent the dynamics of these risks over a long term. In high stressful and volatile markets, markets tend to lose liquidity and therefore the risk horizon should be increased when measuring VaR in stressful circumstances.

If we consider a portfolio of risky assets and a fixed risk horizon h and suppose that we have estimated the P&L distribution that is associated to this portfolio and denote by $F_L(l) = \mathbb{P}(X \leq l)$ its distribution function, we can then define a statistic based on F_L which evaluates the level of risk associated to the holding of our portfolio over the risk horizon.

Definition 3.3.1 (Value-at-Risk)**.** Given a significance level $\alpha \in [0; 1]$ and a predetermined time horizon h, the VaR of a portfolio at the significance level α is given by the smallest number l such that the probability that the loss X exceeds l, in the time horizon h, is no larger than α.

More formally, the VaR risk measure is defined as:

$$\mathrm{VaR}_h(\alpha) := \inf\{l \in \mathbb{R} : \mathbb{P}(X \leq l) \leq \alpha\} \tag{3.11}$$

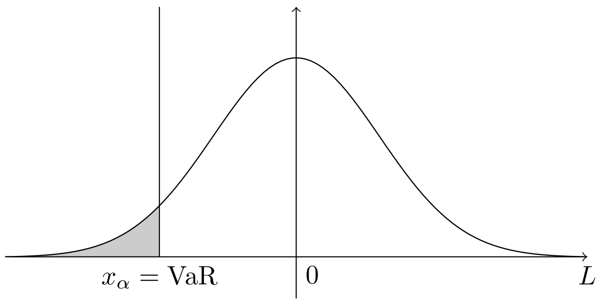


Figure 3.1: 5% VaR of a hypothetical P&L probability density function.

where X is a random variable representing the P&L of our portfolio at time horizon h. In Figure 3.1 we can see a visualisation of the VaR risk measure. The shaded area represents $\alpha \approx 0$ and the α quantile represents the VaR value, x_α.

VaR is thus simply the α quantile of the P&L distribution. In market risk management the risk horizon is usually one to ten days, while in credit risk management it is usually one year. The VaR risk measure is intuitive and popular because of its conceptual clarity. It gives us a rough idea about the extent of risk in our portfolio with only one single number making it quite manageable to interpret. The VaR measure can be measured for different types of assets, whether it is stocks, bonds, currencies, derivatives or any other assets with a price. Because of that, banks and financial institutions use it.

3.3.1 Normal Linear VaR model

The Normal Linear VaR model is only applicable when the return of the profits and losses of a portfolio is a linear function of its risk factor returns.

In addition, there is the assumption that risk factor returns are normally distributed, and that their joint distribution is multivariate normal. Because of that, the covariance matrix (which gives the covariance between each pair of elements) is all that is needed to capture the dependency between the risk factor returns. Therefore, if we write the return of a portfolio as X (excluding the dependence on time and risk horizon for simplicity) we assume

$$X \overset{i.i.d}{\sim} \mathcal{N}(\mu_p, \sigma_p^2) \tag{3.12}$$

where the parameter μ_p is the expected return of the portfolio and σ_p^2 is its standard deviation at time t.

The expected return of the portfolio is calculated with the formula,

$$\mu_p = \mathbb{E}[R_p] = \sum_{i=1}^{N} w_i \mathbb{E}[R_i] \approx \bar{R_p} = \sum_{i=1}^{N} \bar{R_{it}} w_i \tag{3.13}$$

where R_p is the return of the portfolio at time t, $\mathbb{E}[R_i]$ is the expected return of instrument i, $\bar{R_p}$ is the average return of the portfolio at time t, $\bar{R_{it}}$ is the average return of asset i.

The standard deviation is calculated by the formula,

$$\sigma_p^2 = \sqrt{w^T \Omega w} \tag{3.14}$$

where Ω is the covariance matrix of our financial assets in the portfolio,

$$\Omega = \begin{bmatrix} \sigma_1 & \dots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{m1} & \dots & \sigma_m \end{bmatrix} \tag{3.15}$$

The diagonal contains the variances of each stock while the non-diagonal elements is the covariance between each pair of stocks.

The goal is to find the α quantile return such that $\mathbb{P}(X < x_\alpha) = \alpha$. When the standard deviation and expected return has been calculated, the VaR can be found by calculating the inverse of the normal cumulative distribution.

3.3.2 Historical simulation VaR model

The Historical Simulation VaR model reorganizes historical daily returns, putting them in order from worst to best. From the re-organized historical daily returns we can then retrieve the VaR by calculating the x_α quantile from the distribution.

This model has the assumption that all possible variations have been experienced in the past, and that the historically simulated distribution is identical to the returns distribution over the forward looking horizon. One of the main problems of the Historical Simulation model approach is that it needs much data. To construct the historical distribution the number of data points needed is equal to the number of observations on each risk factor return in the simulation. This number of data points should be as high as possible to get good precision. Ideally, the historical data should be sampled at the daily frequency and for many years. The reason for this is that we need much data to estimate the quantiles of an empirical distribution. The accumulation of such data is a lot of work and can be difficult.

In Figure 3.2, we can see an example of 99% VaR of simulated P&L density.

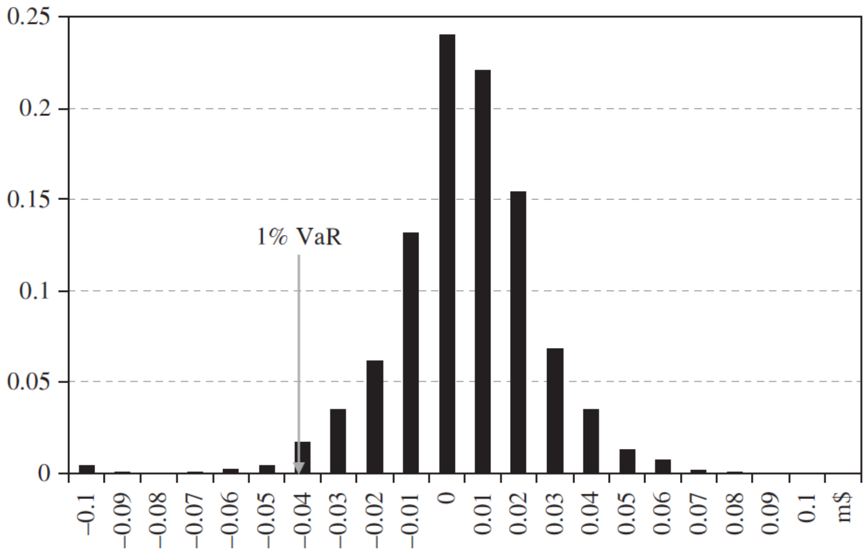


Figure 3.2: Simulated P&L density displaying 1% VaR[Alexander, 2008b].

The advantage of these models is that assumptions for distributions are few. There is no need to assume for any kind of distribution of the risk factor return distribution, least of all multivariate normality. There is only one distribution assumption and that is that the multivariate distribution of the risk factor returns over the risk horizon will be identical to the one in the past. In addition, if the historical VaR is scaled to a longer period of time, then we need to assume that the risk factor returns are i.i.d. They do not have to be normally distributed but they have to have a stable distribution of some sort so that it is possible to derive a scaling rule for the historical VaR.

One of the advantages of the Historical Simulation VaR model is that it bases the risk factor dependencies on experienced risk factor returns and co-movements between these rather than on a parametric model for their distribution like the Normal Linear VaR model. However, because of sample size constraints the historical VaR needs to be assessed initially at the daily horizon, and then scaled up to longer horizons. The scaling from a daily horizon to a long risk horizon requires a detailed investigation of the nature of the empirical return distribution.

3.3.3 Monte Carlo VaR model

The Monte Carlo VaR model, in its basic form, uses the same assumptions as the Normal Linear VaR model, i.e. that the risk factor returns are i.i.d. with a multivariate normal distribution. However, the Monte Carlo VaR model is more flexible and many different assumptions about the multivariate distribution of the risk factor returns can be accommodated.

Monte Carlo models that are i.i.d. multivariate, simulate independent standard normal vectors which are then transformed to multivariate normal vector using the Cholesky decomposition of the risk factor returns covariance matrix. From that we can obtain a simulated portfolio risk horizon, one for each simulated vector of correlated risk factor returns.

One other difference between the Monte Carlo VaR and the Normal Linear VaR is that the Monte Carlo VaR can and should only be applied to nonlinear portfolios, i.e. portfolios where the returns cannot be expressed by linear functions. If Monte Carlo models were applied to linear models we can obtain sampling errors that are not present in the Normal Linear VaR model.

Because of their similarities, the Monte Carlo and Normal Linear VaR models have similar estimations on VaR, but both approaches are common. If we compare the two methods, the Normal Linear VaR model is very precise but is built on assumptions that are unlikely to hold while the Monte Carlo VaR is subject to simulation errors. Thus, these two methods should give a similar value of VaR, if that is not the case then most likely there were too few simulations used in the Monte Carlo VaR execution. The Monte Carlo VaR models can be based on any kind of multivariate distribution for risk factor returns, whereas the normal linear VaR model have closed-form solutions for only a select distributions.

3.4 Backtesting

How should a model be assessed? It can be argued that a model should be assessed on the usefulness of its predictions, not the reasonableness of assumptions or analytical sophistication. From the very beginning of VaR in 1994, financial authorities immediately recognized that there was a need for VaR backtesting methodologies. The first research on this topic was published in 1995 [Kupiec, 1995] and 1996 [Kambhu et al., 1996].

Backtesting is a statistical process of assessing the usefulness of Value-at-Risk predictions when it is applied to a portfolio over a certain period of time. By collecting enough data, i.e. the realized profits and losses for the portfolio, statistical tests (and other kinds of tests) can be applied to assess how well the Value-at-Risk measurements capture the riskiness of a portfolio.

The backtesting concept is explained mathematically in [Roccioletti, 2015] where we consider a continuous loss distribution. By the definition of Value-at-Risk at a certain significance level α and a time horizon h, we have that the violation probability of the VaR number equals α:

$$\mathbb{P}(X < \mathrm{VaR}_{h,\alpha}(X)) = \alpha \tag{3.16}$$

We can describe the violation process with an indicator function as

$$I_{t+1}(\alpha) = \mathbb{1}_{\{X(t+1)<\mathrm{VaR}(X(t+1))\}} \tag{3.17}$$

This means that the hit sequence returns a value of 1, if the loss in day $t+1$ is larger than the predicted VaR number from the model, and 0 if not. To do backtesting we must build this sequence of zeros and ones $\{I_{t+1}\}_{t=1}^{T}$, where T is the number of days in the testing period.

From that, we can conclude that a VaR risk model has correct unconditional coverage if

$$\mathbb{P}(I_{t+1} = 1) = \mathbb{E}[I_{t+1}] = \alpha \tag{3.18}$$

In layman's terms, a correct unconditional coverage indicates that the proportion of VaR violations is not significantly different from α across the number days. A correct conditional coverage suggests that the model gives a VaR hit with the right probability on every day, provided all the information available the day before. Under these set of terms, VaR hits are independent and identically distributed Bernoulli random variables, with a success probability of α.

Therefore, we can think of VaR backtesting as questioning the following null hypothesis:

$$H_0 = I_{t+1} \sim \text{ i.i.d Bernoulli}(\alpha) \tag{3.19}$$

It is worth mentioning that the significance level choice is very important because that choice is directly related to the following errors:

1. Type I: probability of rejecting a correct model.
2. Type II: probability of not rejecting a wrong model.

As the significance level increases, the number of Type I errors increases but number of Type II errors decreases and vice versa.

3.4.1 Binomial Distribution Test

The Binomial Distribution test states that if we have a violation sequence, as described in Equation 3.17, that is independently and identically distributed and has correct unconditional coverage (see Equation 3.18), then the total number of violations x has a binomial distribution $\mathcal{B}(n,\alpha)$ with $\mathbb{E}[x] = [n\alpha]$ and $\text{Var}[x] = n\alpha(1-\alpha)$. If x is a realisation of binomial random variable X, then our null hypothesis is

$$H_0 : X \sim \mathcal{B}(n,\alpha) \tag{3.20}$$

If we have a large number of observations, the central limit theorem can approximate the binomial distribution by the normal distribution. Therefore for discussing the null hypothesis in Equation 3.20 we can use a simple mean test:

$$z = \frac{x - n\alpha}{\sqrt{n\alpha(1-\alpha)}} \approx \mathcal{N}(0,1) \tag{3.21}$$

3.4.2 Kupiec's Proportion of Failure Test

This coverage test is the oldest proposed VaR backtesting methodology and is concerned with whether or not the reported VaR is violated more than $100\alpha\%$ percent of the time.

Let's assume we have a number of observations n and a number of violations x. Let's define $q := 1-\alpha$ and we define the unconditional coverage $q^* = \mathbb{E}[I_t]$, the null hypothesis of the Kupiec's test is:

$$H_0 : 1 - q = q^* \tag{3.22}$$

We treat x as a realisation of a binomial random variable X. To test our null hypothesis at some significance level α, we must construct a non-rejection interval $[x_1, x_2]$ such that

$$\mathbb{P}([X \notin [x_1, x_2]) \leq \alpha \tag{3.23}$$

Rather than directly calculating the probabilities of $\mathcal{B}(n, 1-q)$, the Kupiec's Test uses this distribution to construct a likelihood ratio

$$\Lambda = \frac{q^{n-x}(1-q)^x}{\left[\frac{n-x}{n}\right]^{n-x}\left[\frac{x}{n}\right]^x} \tag{3.24}$$

If the value of the statistic in Equation 3.24 is higher than the critical value of the χ^2 distribution, H_0 will be rejected and we conclude that the model is incorrect. The power of the Kupiec's test grows as we have more data and therefore when we have a considerable amount of data we can reject an inaccurate model without much difficulty [Roccioletti, 2015].

However, it can be difficult to infer probabilities with Equation 3.24. A standard technique is to consider $-2\log(\Lambda)$ [Holton, 2003]. We consider

$$-2\log(\Lambda) = -2\log\left[\frac{q^{n-x}(1-q)^x}{\left[\frac{n-x}{n}\right]^{n-x}\left[\frac{x}{n}\right]^x}\right] = -2\log\left[\left[\frac{n-x}{qn}\right]^{n-x}\left[\frac{x}{(1-q)n}\right]^x\right] \tag{3.25}$$

According to [Lehmann and Romano, 2005] the expression in Equation 3.25 is approximately centrally chi-squared with one degree of freedom, i.e. $-2\log(\Lambda) \sim \chi^2(1,0)$ assuming H_0.

Given a significance level α, we can calculate the α quantile of the $\chi^2(1,0)$ distribution and setting that equal to Equation 3.24 we can solve for x. From that we will get two solutions which can be obtained using numerical methods. Rounding the lower one down and the higher one up gives us the $[x_1, x_2]$ non-rejection interval.

3.4.3 The Basel Committee's Traffic Light Coverage Test

The final backtesting framework we want to cover is a regulatory framework established by the Basel Committee in 2013 and is based on a regular comparison of the bank's daily risk measure with the realized profit or loss. The use of a proprietary VaR measure requires approval of regulators. A bank is obligated to have an independent risk management function and satisfy regulators that it was following acceptable risk management practices.

As with other backtesting methodologies, we look at how often the trading outcomes exceeds the risk measures (or look at how often the risk measures were larger than the trading outcome). Then the fraction covered can be compared with the intended level of coverage to gauge the performance of the bank's risk model [Grody et al., 2013]. This backtesting methodology is based on the VaR measures at the 99% percentile confidence level. The banks were required to produce 250 risk measures forecasts. According to the Basel Committee, an exception occurs when the loss of a trading desk registered in a day of the backtesting period is higher than the corresponding VaR measure from our model. If a risk measure or the profit (or loss) is unavailable for some reason it will count as an exception.

We can define a risk capital requirement as a minimum capital requirement for banks set by regulators. In this backtesting framework, this capital requirement depends not only on the portfolio risk but also on the outcome of the backtesting procedure

$$\mathrm{CR}_t = mf_t \cdot \rho(L_t) \tag{3.26}$$

where CR stands for Capital Requirement, mf for the multiplication factor and ρ is the risk measure computed on the loss distribution L. The subscript t indicates that all of this is calculated at time t, using the information we have up to time $t-1$. The mf_t term is determined from the backtesting results and the Basel Committee ranks the backtesting according to three categories: green, yellow and red zones. This approach is called the The Basel Traffic Light Approach.

The number of VaR exceptions that land in the green zone are not of concern and those that fall in the yellow zone require monitoring. The Basel Committee recommended that the VaR measures from the yellow zone be weighted more heavily in calculating the banks' capital charges for market risk and are implicated by the multiplier factors - these multipliers are indicated in Table 3.1.

Zone	Number of Exception	Plus Factor	Cumulative Probability
Green Zone	0	0.00	8.11
	1	0.00	28.58
	2	0.00	54.32
	3	0.00	75.81
	4	0.00	89.22
Yellow Zone	5	0.40	95.88
	6	0.50	98.63
	7	0.65	99.60
	8	0.75	99.89
	9	0.85	99.97
Red Zone	≥ 10	1.00	99.99

Table 3.1: The Basel Committee defined the green, yellow and red zones for backtesting proprietary 1-day 99% VaR measures, assuming 250 observations.

Table 3.1 shows the plus factors we need to calculate the multiplication factor

$$mf_t = 3 + \text{ Plus Factor} \tag{3.27}$$

Note that for a model providing 99% coverage it is quite likely to produce as many as four exceptions and therefore there is little reason to be concerned by backtesting results that fall in this range [Grody et al., 2013]. As the number of exceptions go from five to nine, we fall in the yellow zone in which outcomes suggest both accurate and inaccurate models. Obviously, as the number of violations grow the validity of our model become smaller as the number of violations approaches nine. This can be reflected by the scaling factor. If the backtesting results fall into the red zone that clearly indicates that there is something wrong in our model: a well specified model should not display ten or more violations for 250 observations.

3.5 Sum Product Networks

The goal of this study, as mentioned, is to use a probabilistic model to represent probability distributions, i.e. compute their marginals and modes efficiently and learn them accurately. Sum Product Networks (SPNs) are one kind of these probabilistic models. They are directed graphs that represent a probability distribution that is resulted from a hierarchy of distributions combined in the form of mixtures (sum nodes) and factorizations (product nodes). The main advantage of SPNs is that they can perform inference task in time proportional to the number of edges in a graph [Sánchez-Cauce et al., 2021].

SPNs are similar to other probabilistic graphical models (PGMs), such as Bayesian Networks (BNs) and Markov Networks, in their ability to represent probability distributions but the main difference is that in PGMs, each node represents a variable and edges represent probabilistic dependencies, while in SPNs, each node represents a probability distribution. PGMs can be compiled into SPNs and arithmetic circuits [Chavira and Darwiche, 2007]. As stated, SPNs compute marginals and conditional probabilities proportional to the number of edges in the graph while in BNs inference is NP-hard.

We can look at SPNs as a particular type of feed-forward neural networks because there is a flow of information from the input nodes to the output nodes. However, the main difference between SPNs and neural networks is that SPNs have a probabilistic interpretation while standard neural networks do not. Inference is also different between the two: to compute a posterior probability requires two passes, and the most probable explanation (MPE) - in most SPNs - can be found by backtracking from the root to the leaves. In addition, SPNs can do inference with partial information, i.e. when some variables are unknown, while in neural networks it is necessary to assign a value to each of the input nodes. When looking at parameter learning, neural networks are usually trained with gradient descent or variations of that, while SPNs can be trained with probabilistic algorithms, such as EM and Bayesian methods.

Definition 3.5.1 (Sum Product Networks)**.** An SPN $\mathcal{S}$ is a rooted acyclic directed graph such that:

1. Each leaf node represents a probability distribution for a finite-states variable, V.
2. All the other nodes are either of type sum or product.
3. Every edge $n_i \to n_j$ outgoing from a sum nodes has an associated weight, $w_{ij} > 0$.

An SPN can be built bottom-up beginning with sub-SPNs of one node and joining them with sum and product nodes.

We can define the scope of a node n_i, with $sc(n_i)$. For a leaf node, it is the set of variables on which the probability distribution is defined. The scope of a non-terminal node n_i is the union of scopes of its children:

$$sc(n_i) = \bigcup_{j \in ch(i)} sc(n_j) \tag{3.28}$$

The scope of an SPN $\mathcal{S}$, denoted by $sc(\mathcal{S})$, is the scope of its root, $sc(n_r)$. A sum node is complete if all its children have the some scope. An SPN is complete if all its sum nodes are complete. Finally, a product node is decomposable if all its children have pairwise disjoint scopes. An SPN is decomposable if all its product nodes are decomposable.

SPNs are defined recursively, as weighted sums and products of smaller SPNs, with univariate distributions as the base case. These networks also have other important classes of probabilistic models as special cases, including mixture models, thin junction trees, non-recursive probabilistic context-free grammars, and more [Gens and Pedro, 2013].

In [Poon and Domingos, 2011], they described and defined an SPN over Boolean variables. They defined the negation of a random variables X_i as $\bar{X}_i$ and the indicator function [.] has a value 1 when its argument is true and 0 otherwise. They abbreviate $[X_i]$ as X_i and $[\bar{X}_i]$ as $\bar{X}_i$.

They defined the unnormalized probability distribution $\Phi(x) \geq 0$. The network polynomial of $\Phi(x)$ is $\sum_x \Phi(x)\Pi(x)$ where $\Pi(x)$ is the product of the indicators that have value 1 in state x. This network polynomial is a multilinear function of the indicator variables. The unnormalized probability of the evidence e is the value of the network polynomial when all indicators compatible with e are set to 1 and the remainder are set to 0. For any evidence e, the cost of computing $P(e) = \frac{\Phi(e)}{Z}$ is linear in the size of the network polynomial. The network polynomial $\sum_x \Phi(x)\Pi(x)$ has size exponential in the number of variables, however we may be able to represent and evaluate it in polynomial space and time using an SPN.

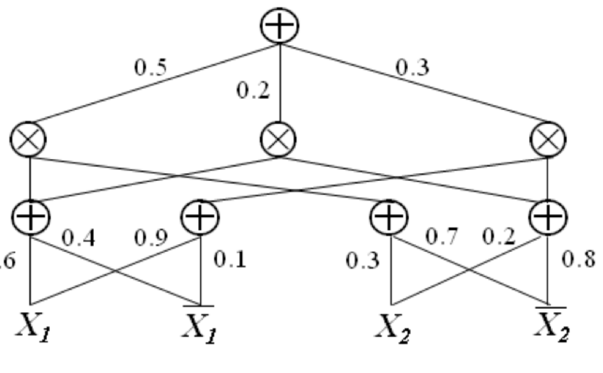


Figure 3.3: SPN implementing a naive Bayes mixture model

For a given structure of SPNs the weights can be learned in two ways; generatively and discriminatively. In the generative case, a well-known algorithm, the EM algorithm, was proposed by [Poon and Domingos, 2011] where each example is presented in turn to the SPN. The authors demonstrated that this algorithm was capable of learning very deep SPNs but they used a predefined SPN structure that was both expensive to learn (because there were a lot of nodes) and insufficiently flexible. For the other case, the discriminative one, [Gens and Domingos, 2012] proposed a backpropagation-style gradient descent algorithm which obtained state-of-the-art results on image classification problems, but again faced a trade-off between flexibility and cost of learning [Gens and Pedro, 2013].

In [Gens and Pedro, 2013] paper they proposed a method for learning an SPN structure. Their algorithm, named LearnSPN, inputs an i.i.d. sample of a vector-valued variable that is in the form of a matrix of instances by variables. If the vector is of unit length, their algorithm returns the corresponding univariate distribution. In the case there are no dependencies detected among the variables, the algorithm returns a fully factorized distribution.

Another recent approach for learning the structure of SPNs was proposed by [Peharz et al., 2020]. In this paper, they introduced a simple and scalable method to construct random and tensorized SPNs, named RAT-SPNs. First a random region is constructed, which is then populated with array of SPN nodes. This particular strategy essentially dictates a random hierarchical tensorial decomposition, learning SPNs with reduced sparsity. These types of SPNs map well into existing deep learning frameworks like Tensorflow and scales to a million parameters.

In the case of density estimation, [Peharz et al., 2020] used the classic EM algorithm previously mentioned. The EM algorithm is free of tuning-parameters and rapidly increases the likelihood, and is therefore a good choice for this particular task. Their results showed that the RAT-SPNs yielded test-likelihoods very close to ID-SPN which is one of the most sophisticated SPN learners available.

For a given training set, $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ of i.i.d. samples, which is drawn from an unknown distribution $P^*(\mathbf{X})$, we want to approximate this distribution. The canonical approach to generative learning is maximizing the log-likelihood

$$\mathrm{LL}(w) = \frac{1}{N}\sum_{n=1}^{N} \log \mathcal{S}(\mathbf{x}_N) \tag{3.29}$$

where w denotes all the parameters of the SPN, i.e. sum weights and parameters of the input distributions [Peharz et al., 2020].

To optimize Equation 3.29, the EM algorithm is applied. This algorithm rapidly and monotonically increases the likelihood, is free of tuning-parameters and can be implemented via simple and backward evaluations to compute the required statistics.

3.6 Mixed Sum-Product Networks

There is a particular type of Sum-Product Networks that appears to be applicable in risk assessment in portfolio theory; an architecture called Mixed Sum Product Networks (MSPNs). When structuring probabilistic models we sometimes have to spend a considerable amount of time in identifying the parametric form of the random variables (Gaussian, Poisson, etc.). One approach to make this easier is to use MSPNs, a deep architecture for hybrid domains that features tractable queries [Molina et al., 2018]. They are a general class of mixed probabilistic models, by combining Sum-Product Networks and piecewise polynomials, and they allow for a broad range of exact and tractable inference without having to make assumptions of distribution. To learn MSPNs from data we are required to do different conditioning and decomposition steps for SPNs that is tailored towards nonparametric distributions, i.e. we need to describe methods so that we can model hybrid domains without making parametric assumptions. These steps are described in [Molina et al., 2018], namely three factors are explained: Rényi Decomposition, Rényi Conditioning and Nonparametric Univariate Leave Distributions.

They approached the problem of seeking independent subsets of random variables of mixed but unknown types as a dependency discovery problem. According to [Rényi, 1959], a measure of maximum dependence $\rho^* : V_i \times V_j \to [0,1]$ between random variables V_i and V_j should satisfy several fundamental properties like symmetry, transformation invariance and that the following expression should hold $\rho^*(V_i, V_j) = 0$ if and only if V_i and V_j are statistically independent. Rényi also explained how the Hirschfeld-Gebelein Maximum Correlation Coefficient (HGR) covers all of these conditions.

Definition 3.6.1 (Hirschfeld-Gebelein-Rényi Maximum Correlation coefficient)**.** Given random variables X and Y, the Hirschfeld-Gebelein-Rényi maximal correlation of (X, Y) is defined as follows [Anantharam et al., 2013]:

$$\rho_m(X;Y) := \max_{(f(X),g(Y))\in\mathcal{S}} \mathbb{E}[f(X)g(Y)], \tag{3.30}$$

where $\mathcal{S}$ is the collection of pair of real-valued random variables $f(X)$ and $g(Y)$ such that

$$\mathbb{E}f(X) = \mathbb{E}g(Y) = 0, \text{ and } \mathbb{E}f^2(X) = \mathbb{E}g^2(Y) = 1 \tag{3.31}$$

In a paper [Lopez-Paz et al., 2013] they presented an estimator for the HGR ρ^* coefficient, the randomized dependency coefficient (RDC). This estimator is quite useful for hybrid domains as it can be applied to both multivariate, continuous and discrete random variables. In addition, it runs in $\mathcal{O}(M \log M)$ running time, with M being the number of instances making it one of the fastest nonlinear dependency measure. The idea behind the RDC is to find the linear correlation between the representations of two random samples that have undergone a series of non-linear transformations. The samples are concluded to be independent if and only if the transformed samples are linearly uncorrelated.

If we consider two random samples $\mathcal{D}_{V_i} = \{v_i^m | v_i^m \sim V_i\}_{m=1}^M$ and $\mathcal{D}_{V_j} = \{v_j^m | v_j^m \sim V_i\}_{m=1}^M$ drawn from random variables V_i and V_j, we conclude that V_i and V_j are independent if and only if $\rho(\mathcal{D}_{V_i}, \mathcal{D}_{V_j}) = 0$, where ρ is the RDC. In order to do achieve invariance against scaling and shifting data transformations, we compute their *empirical copula transformations* [Póczos et al., 2012], $\mathcal{C}_{V_i}$ and $\mathcal{C}_{V_j}$.

$$\mathcal{C}_{V_i} = \left[\frac{1}{M} \sum_{r=1}^{M} \mathbb{1}\{v_i^r \le v_i^m\} | v_i^m \in \mathcal{D}_{V_i} \right]_{m=1}^{M} \tag{3.32}$$

After that, we apply a random linear projection on the obtained samples to a k-dimensional space, finally passing them through a non-linear function σ,

$$\phi(\mathcal{C}_{V_i}) = \sigma(\mathbf{w} \cdot \mathcal{C}_{V_i}^T + b), (\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k\times k}) \tag{3.33}$$

Note that $\mathbf{w} \in \mathbb{R}^{k\times 1}, b \in \mathbb{R}$ and that random sampling $\mathbf{w}$ from a zero-mean k-dimensional Gaussian is analogous to the use of a Gaussian kernel [Lopez-Paz et al., 2013]. In [Molina et al., 2018] they used $k = 20$, σ to be the sine function and $s = \frac{1}{6}$ because they have been proven to be useful heuristics according to [Lopez-Paz et al., 2013]. Then, we can compute the canonical correlations (CCA) ρ^2 for $\phi(\mathcal{C}_{V_i})$ and $\phi(\mathcal{C}_{V_j})$ as the solutions for the eigenproblem:

$$\begin{bmatrix} 0 & \sum_{ii}^{-1}\sum_{ij} \\ \sum_{jj}^{-1}\sum_{ji} & 0 \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \rho^2 \begin{bmatrix} \beta \\ \gamma \end{bmatrix} \tag{3.34}$$

From these solutions, we conclude the RDC coefficient to be the largest canonical correlation coefficient:

$$\text{RDC}(V_i, V_j) = \sup_{\beta,\gamma} \rho(\beta^T \phi(\mathcal{C}_{V_i}), \gamma^T \rho(\mathcal{C}_{V_j})) \tag{3.35}$$

In the case of independent random variables, we are looking for the RDC to be zero.

For the clustering of hybrid data [Molina et al., 2018] proposed to cluster multivariate hybrid samples after the RDC pipeline has processed them. By doing that, we can produce a feature space in which clusters may be more easily seperable and no assumption of distributions has to be made.

Lastly, to realize MSPNs, they adapted piecewise polynomial approximations of the univariate leaf densities. One straightforward approach to approximate the univariate leaf densities is to use piecewise constant functions, i.e. histograms. They adapted a scheme proposed in [Rozenholc et al., 2010], offering an adaptive binning, i.e. with regular intervals, that is learned from data by optimizing a penalized likelihood function. By doing this, we allow MSPNs to model both multimodal and skewed univariate distributions without further assumptions. They also applied Laplacian smoothing by a factor of Δ to cope with unseen values and the natural overfitting of histograms. As expected, if we increase the degree of our leaf polynomial approximations, we can get more expressive models. To find the right balance between complexity of learning inference and expressiveness they restrained to piecewise linear approximations. From these three factors of consideration, we can construct an MSPN.

To summarize, MSPNs are a combination of nonparametric probability distributions and deep probabilistic models. These networks provide effective learning, tractable inference and enhanced probability. They allow us to train multivariate mixed distributions more easily than previous methods across a wide range of domains. Because there is no need for parametric assumptions in the MSPN framework it seemed to be a good fit for learning the high-dimensional joint probability distribution of the P&L distribution (Equation 3.4) in portfolios and estimate the VaR.

Chapter 4

Previous Work

In this chapter we will look at a few machine learning studies in the field of finance, some regarding the estimation of VaR.

4.1 Portfolio Analysis Using General Adversarial Networks

The previous mentioned VaR models are well recognized and have been used for quite some time in the industry. A recent study [Mariani et al., 2019], used an alternative approach to model the market uncertainty. They proposed a generative model to learn the joint probability distribution of the real market. This model learned the joint probability distribution of the price trends for a particular portfolio (or set of assets) to match the probability distribution of the real market. When the model was then trained they optimized the portfolio by deciding the best diversification to minimize the risk and maximize the expected returns observed over the execution of several simulations.

In their work they modeled the probability distribution of the asset-price trends for the future f days given the current market situation represented by the latest observed b days. They also pointed out that since they are dealing with time-series data they have to count in the factor of time. By using a 1D convolutional neural network (CNN) they had an effective way to process the time series as pointed out by [Bai et al., 2018]. Using the CNN they ended up with a matrix M with k rows (financial assets) and w columns (days), $M \in \mathbb{R}^{k \times w}$. The matrix M spanned the whole analysis length and contained the known past called M_b and the unknown future M_f. A generative deep-neural network G was then applied to learn the probability distribution of future price trends M_f within the target future horizon f given the known recent past M_b, and a prior distribution of a random latent vector.

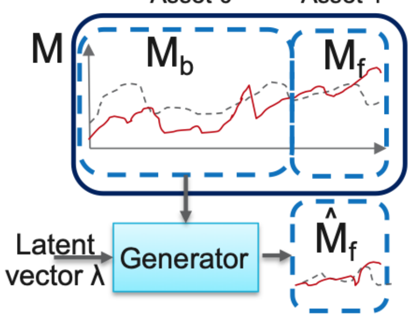


Figure 4.1: Conceptual overview of the proposed PAGAN generator [Mariani et al., 2019]

A graphical interpretation is displayed in Figure 4.1, the matrix M represents the input and the output of the generator G. The generative model returns a possible realisation future matrix M_f as a function:

$$\hat{M}_f = G(M_b, \lambda) \tag{4.1}$$

where λ is the latent vector from a prior distribution. The λ vector represents the unknown future events and phenomena the impacts the marketplace. The known past M_b was used to condition the probability distribution of the future $\hat{M}_f$ based on the most updated market situation.

Their result was that their approach was able to expose to the final user the possibility of selecting a target risk level and come up with a suggestion of diversification given the current market situation. By comparing their results with the Markowitz modern portfolio optimization approach[1] they achieved better performance in terms of expected return maximization and risk minimization.

4.2 Forecasting Exhange Rate VaR Using Deep Belief Network Ensemble Based Approach

In a paper from 2018, [He et al., 2018] proposed a new VaR estimate that is based on Deep Belief Network ensemble model with Empirical Mode Decomposition (EMD) technique. With the EMD-BDN ensemble model they attempted to capture multi-scale features with the aim to predict the risk movement accurately. Individual data components are extracted using the EMD model while individual forecasts can be calculated at different scales using the ARMA-GARCH model. The DBN model searches for the optimal nonlinear ensemble weights to combine the individual forecasts at different scales into the ensembled exchange rate VaR forecasts.

Deep Belief Network (DBN) is a generative model in the general deep learning framework. The DBNs have activation functions and process the information and as the number of layers increases the number of parameters involved in training the network increases exponentially. To cope with this problem, they used Boltzmann machines in the DBNs (RBMs) for pre-training. RBM is used as the pre-training model to learn the hidden data feature in an unsupervised learning process. They are neural networks and consists of two layers called visible layer and hidden layer. In RBMs, neurons in the different levels of layers have mutual undirected connections while the neurons in the same level of the layers are independent. In the structure of RBM, v and h represent the state of the neurons of visible layers and hidden layers respectively. The structure of the DBN is displayed in Figure 4.2. The training process of the DBN model can be divided into pre-training phase with an unsupervised learning process and fine tuning phase with a supervised learning process.

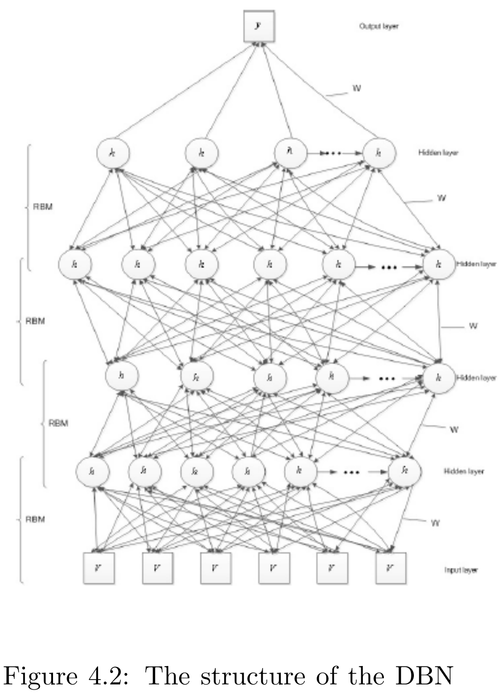


Figure 4.2: The structure of the DBN model [He et al., 2018].

They constructed an ensemble VaR model based on EMD and the DBN model that consisted of a series of steps to transform and model the data characteristics. The first step was to assume that there was a multi scale risk structure in the exchange rate movement. Given the exchange rate time series data, $x_i, i = 1, 2, .., N$, EMD model was used to calculate the decomposed components IMF at different scales up to the maximal scale N and the residuals e_i, $x_i = \sum_{i=1}^{N} \text{IMF}_{i,t} + e_t$.

[1]Markowitz, H. M. (1991). Foundations of portfolio theory. The journal of finance, 46(2), 469-477.

This transforms the exchange rate data into a series of data components, then there is further ensemble modelling performed. Secondly, they estimated the ARMA-GARCH model for individually extracted data components using the model training data. The conditional mean and conditional standard deviation were then forecasted using ARMA-GARCH model, however with the normal assumption. Thirdly, they assumed that there was a nonlinear relationship between the individual risk factors and the total risk estimate. Finally, they calculated the VaR with the conditional mean and standard deviation with $\text{VaR}_\alpha = -\mu + \sigma Z_a$, where $a = 1 - \text{cl}$, Z_a was the normal variate at quantile a, μ and σ were the conditional mean and standard deviation.

They used an extensive market dataset across major exchange markets, including Australian Dollar against Dollar, Dollar against Canadian Dollar, Dollar against Swiss Franc, and Euro against Dollar. In addition, they backtested their backtesting result using a conditional coverage test, looked at the number of exceedances for each model and deciding if the model was useful by looking at the p-values. In terms of average p value the DBN model performed better than the ARMA-GARCH model and was more conservative at the higher VaR levels (97.5% and 99%).

Their results was that the DBN model produces generally more optimal forecasts. However, the DBN model assumes that the risk factor returns of exchange rates followed a (multivariate) normal distribution which is most often unrealistic in the real-world. In our MSPN approach, there is no need for an assumption of this kind, the MSPN main strength is that it can learn high-dimensional joint distributions with no parametric assumptions for the returns.

4.3 Bayesian LSTM model for Value at Risk and Expected Shortfall Joint Forecasting

In a study, [Li et al., 2020] proposed using a hybrid model based on the Asymmetric Laplace quasi-likelihood and employed Long Short-Term Memory (LSTM) time series modelling technique to capture the dynamics of VaR and ES (alternative to VaR, more sensitive to the shape of the tail of the loss distribution) efficiently. They referred to this model as LSTM-AL and used the Markov chain Monte Carlo (MCMC) algorithm for Baysian inference in the LSTM-AL model.

The LSTM-AL model is an innovative model that is a combination of the LSTM structure and the ES-CAViaR framework [Engle and Manganelli, 2004], which is a framework that specifies the evolution of the quantile over time using an autoregressive process and estimates the parameters with regression quantiles. The model has 18 parameters and they used MCMC to sample from their posterior distribution. A graphical representation of the model is shown in Figure 4.3.

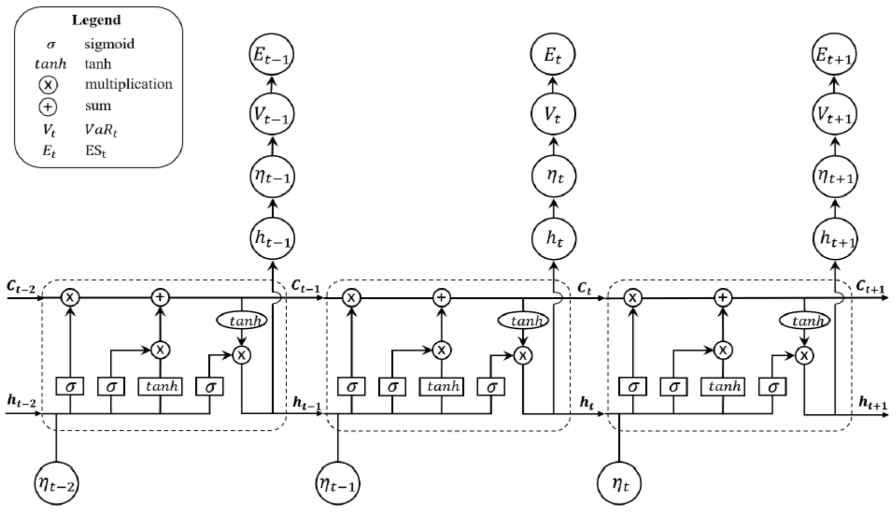


Figure 4.3: A graphical representation of the LSTM-AL model [Li et al., 2020].

They did a simulation study to compare both the in-sample and out-of-sample properties of the proposed Bayesian LSTM-AL model, in comparison with other competing models. With each set of the simulated data, which were simulated using a non-linear stochastic volatility model, they used the first 1000 observations for Baysian model estimation (in-sample analysis) and the last 1000 observations for model evaluations (out-of-sample analysis). They then compared the performance of the LSTM-AL model with two ES-CAViaR family models. Their conclusion was that the LSTM-AL model outperformed two benchmark models in all their simulated datasets in terms of quantile loss and AL score.

In addition, they did an empirical study where the applied the LSTM-AL model into four financial time series to test its ability in capturing VaR and ES dynamics. The results showed that the LSTM-AL model consistently outperforms the other models across all stocks index markets. They concluded that that indicates that the LSTM-AL model can capture non-linear and long-term serial dependence properties exhibited by the selected financial stocks markets.

To conclude, the LSTM-AL model generated favourable results in both simulation study and empirical study, in terms of the VaR quantile loss function and VaR-ES joint score function, especially in the financial stocks markets where the long-term dependence and non-linearity properties are present.

4.4 VaR Exception Classification Model With Support Vector Machines

The final study related to the application of machine learning on VaR which we will present is [XIONG, 2018] where they looked at VaR exceptions with a machine learning model. More precisely, they built an SVM classifier that classifies VaR exceptions into "market move" and "VaR model issue". They used TSNE data visualization, which is a method for visualising high dimensional data, to study the separability of the two categories. In addition, they proposed a numerical method to approximate VaR model predicted P&L and proved an asymptotic convergence property.

They defined X_{t+1} as a vector of portfolio returns between the end of day t and the end of day $t+1$, and Δ_t as the risk captured by a VaR model at the beginning of day $t+1$. Then according to the VaR model, daily P&L between the end of day t and the end of day $t+1$ equals the product of risk and return,

$$\text{VaR PL}_{t+1} = \Delta_t \cdot X_{t+1} \tag{4.2}$$

To be able to calculate VaR of date t, a predictive model calculated a risk-synchronized P&L according to some distribution D_t. For example, historical VaR models use historical returns back to T days and the computable risk of day t:

$$D_t = \{\Delta_t X_t, \Delta_t X_{t-1}, ..., \Delta_t X_{t-T}\} \tag{4.3}$$

In addition, they defined the clean P&L, which excludes the intraday trades), denoted by $clnPL$. Compared with VaR-predicted P&L, the clean P&L may contain a part that is not explained by the VaR model, denoted as $unxPL$. Then the clean $P\%L$ can be decomposed in the following way

$$clnPL_{t+1} = \Delta_t \cdot X_{t+1} + unxPL_{t+1} \tag{4.4}$$

Thus, we can define the VaR P&L p-value as:

$$\text{VaR PV}_{t+1} = D_t(\text{VaR PL}_{t+1}) \tag{4.5}$$

The p space ranges from 0 to 1 and therefore the P&Ls are normalized automatically when projected onto the p-values.

They achieved high accuracy rate by the RBF Kernel SVM model. The model successfully predicted the exception points which do not obey exactly the p-value criterion.

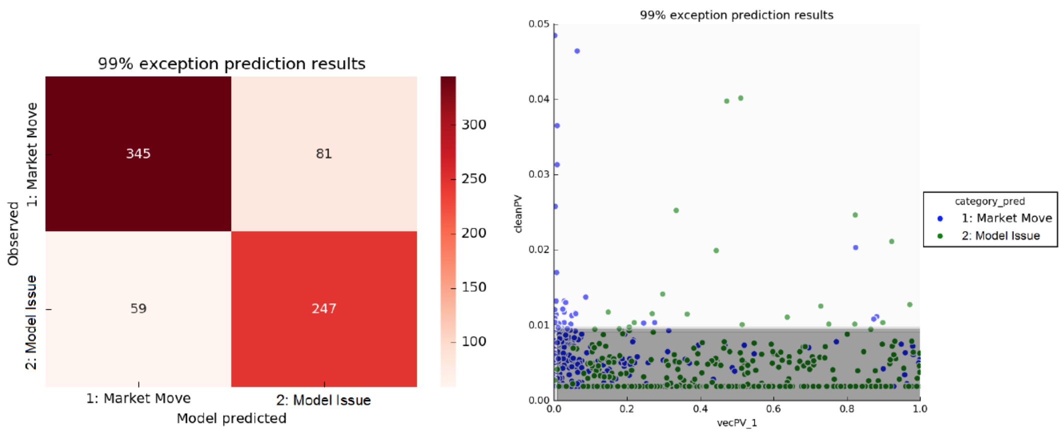


Figure 4.4: Prediction results on 99% VaR exceptions [XIONG, 2018].

From Figure 4.4 we can see that there is no clear boundary between predicted market moves and model issues, which indicates that the real datasets are for portfolios where the risk profile significantly changes. The parameters of the RBF Kernel SVM was achieved through cross-validation. The prediction power of the SVM is also validated by the fact that it reproduces the exact classification rule on an ideally simulated dataset.

Chapter 5

Experimental Setup

In this chapter, we will explain the data that was used for the VaR estimation, how the data was preprocessed, provide details about how we structured the MSPN model, how we used the outputs from the MSPN model to calculate the VaR and finally we'll explain the hyperparameters of our model.

5.1 Portfolio Data

The data that was used for the portfolio analysis was the top 40 companies, sorted by weight, listed in the S&P 500 index[1]. The S&P (Standard & Poor's) 500 index is a market-capitalization-weighted index of the 500 largest publicly-traded companies in the United States. The S&P 500 uses this market-capitalization weighting method, giving a higher percentage allocation to the companies that have the largest market capitalizations (market cap), which is the market value of a publicly traded company's outstanding shares:

$$\text{Company Weighting in S\&P 500} = \frac{\text{Company market cap}}{\text{Total of all market caps}} \tag{5.1}$$

The weighting of each component of the S&P 500 begins with summing the total market cap for the index. The companies used in this project are listed in Table 5.1.

The financial data that was used in this project ranged from 03/07/2013-20/07/2021 and contained the opening, high, low, closing and adjusted closing price, along with the volume traded for each stock. We considered only the closing price for each asset to train our models and calculate the VaR. The closing price is the raw price or cash value of the last transacted price of an asset before the market officially closes for normal trading. The closing prices are often used as a reference point for investors to compare a stock's performance since the previous day and are often used to construct graphs displaying historical price changes over a period of time.

We decided to normalize the data for each asset, i.e. we normalize the prices for each asset so it fits in range $[-1, 1]$. Thus, we receive the daily percentage changes in closing prices for each asset i at time t by applying the transformation

$$\bar{p}_{i,t} = \frac{p_{i,t} - p_{i,t-1}}{p_{i,t-1}} \tag{5.2}$$

By normalizing our data we achieve a common scale for all of the financial assets in the portfolio and our models are exposed to data within a reasonable range. In general, normalization is common in machine learning when we do not know the distribution of our data and it has varying scales which fits our data characteristics.

[1]Top list was retrieved from https://www.slickcharts.com/sp500

#	Company	Symbol	#	Company	Symbol
1	Apple Inc.	AAPL	21	Exxon Mobil Corporation	XOM
2	Microsoft Corporation	MSFT	22	Netflix Inc.	NFLX
3	Amazon.com Inc.	AMZN	23	Verizon Communications Inc.	VZ
4	Facebook Inc. Class A	FB	24	Intel Corporation	INTC
5	Alphabet Inc. Class A	GOOGL	25	salesforce.com inc.	CRM
6	Alphabet Inc. Class C	GOOG	26	Cisco Systems Inc.	TMO
7	Berkshire Hathaway Inc. Class B	BRK-B	27	Pfizer Inc.	PFE
8	JPMorgan Chase & Co.	JPM	28	Coca-Cola Company	KO
9	Tesla Inc	TSLA	29	Abbott Laboratories	ABT
10	Johnson & Johnson	JNJ	30	AbbVie Inc.	ABBV
11	NVIDIA Corporation	NVDA	31	PepsiCo Inc.	PEP
12	Visa Inc. Class A	V	32	NIKE Ince. Class B	NKE
13	UnitedHealth Group Incorporated	UNH	33	AT&T Inc.	T
14	Home Depot Inc.	HD	34	Thermo Fisher Scientific Inc.	TMO
15	Procter & Gamble Company	PG	35	Chevron Corporation	CVX
16	Bank of America Corp	BAC	36	Accenture Plc Class A	ACN
17	Walt Disney Company	DIS	37	Broadcom Inc.	AVGO
18	Mastercard Incorporated Class A	MA	38	Merck & Co. Inc.	MRK
19	Adobe Inc.	ADBE	39	Walmart Inc.	WMT
20	Comcast Corporation Class A	CMCSA	40	Eli Lilly and Company	LLY

Table 5.1: Top 40 stocks of the S&P 500 index on 28/07/2021.

For visualization of multiple stocks we applied the transformation from closing prices to base 100. This is used in industry to normalize percentages because the percentage changes makes it difficult to compare multiple stocks. We apply the transformation for each stock i

$$\{p_{i,t}\} \to \left\{ I_t = \frac{I_0}{p_{i,0}} p_{i,t} \right\} \tag{5.3}$$

where $I_0 = 100$ is the base case. In Figure 5.1 we can see the historical price changes after doing the index transformation.

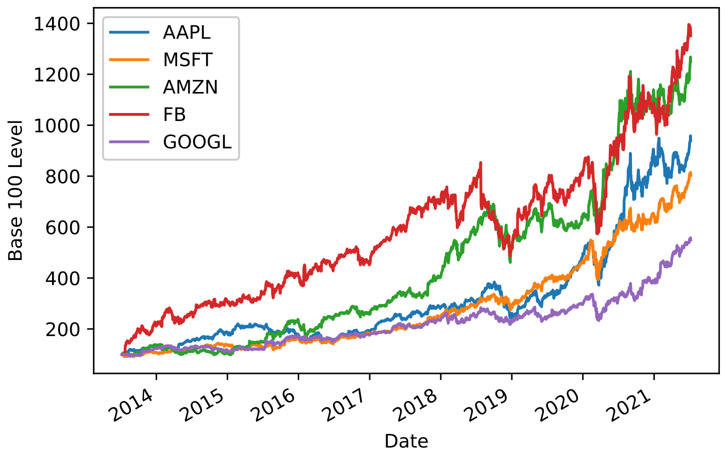


Figure 5.1: Historical prices in base 100 of the top 5 companies in the S&P index from 2013-2021.

From Figure 5.1 we can see closing prices for the top 5 companies in the S&P index. These are daily percentage changes for five companies: Amazon, Microsoft, Amazon, Facebook and Google. Notice that these five assets follow an upward trend from 2013-2021 but at a different growth rate.

For further research of our data, it is useful to see how the historical daily percentage returns are distributed for each stock, for example if we want to see if it is realistic to make the assumption that the P&L distribution is normally distributed. In Figure 5.2, we can see on a historical basis the daily return distribution for the top five companies.

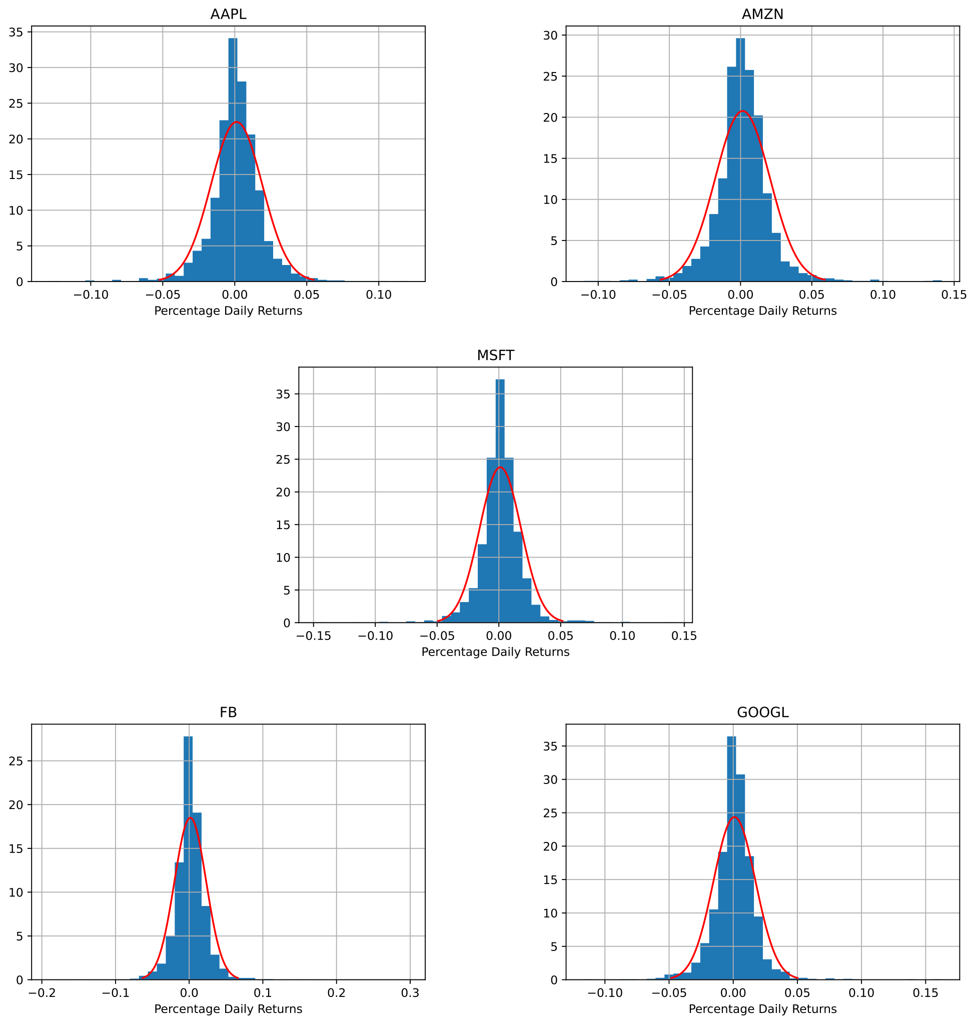


Figure 5.2: Historical daily percentage distribution of returns of the top 5 companies in the S&P index from 2013-2021.

In Figure 5.2 we can see that the daily returns of all assets violate the assumption of normality. The normal curve does not cover the heavy tails of the distribution, which are the extreme events, in all cases. Thus, the underlying assumptions of the Normal Linear VaR model and Monte Carlo VaR are violated while the MSPN and the Historical Simulation model make no parametric assumptions of the daily returns.

5.2 Architecture of the MSPN model

The MSPN was implemented in Python using the library SPFlow, which is an open-source Python library that provides an interface to inference, learning and manipulation routines for deep and tractable probabilistic SPN models. We can build the MSPN model directly from data and there are a few hyperparameters that needs to be calibrated, these are

1. Context - here we set up how we are to model our data, in our case we specify that we want to model 5, 10, 20 or 40 real valued variables (daily percentage returns of stocks). In other types problems this can be a list of parametric types (Gaussian, Poisson, etc.).
2. Correlation measure - We need to decide how we want to calculate the correlations of our data which is an essential part to learn the tree structured MSPN. The default value is RDC which is the estimator for the HGR coefficient (see Equation 3.35).
3. Clustering method - This is the task of Rényi Decomposition (see Section 3.6) where we choose how we want to cluster our data samples and depends on the choice of a metric space.
4. Minimum number of instances (η) - A number representing how often we split our input data into independent components where each component is either partitioned into clusters, resulting in a sum node, or each component is recursively induced into a product node. The default value for this parameter is 200.
5. Threshold - edges between the random variables have to pass a certain threshold of significance. The default value is 0.3.
6. Random Generator - exposes a number of methods for generating random numbers drawn from a variety of probability distributions.

From the MSPN model we can get possible realisations of daily percentage changes in returns and use that to acquire the VaR. More precisely, once we have trained our MSPN model, from time $[0, t]$, we can sample from our learnt joint distribution of daily returns of our portfolio many times and multiply each realisation with its corresponding weight to get our portfolio return at time $t+1$ from Equation 3.3. When that is finished, we can calculate the significance (α) percentile to get our VaR estimation. This procedure is described in Algorithm 1.

Algorithm 1 Calculate α100% 1-day SPN VaR

```
1: procedure CalculateMSPNVaR(Scenario, $\alpha$)    ▷ Specify a number of simulations & $\alpha$
2:     for each scenario do
3:         Sample a possible realisation of the joint distribution for time $t + 1$.
4:         Multiply each instrument with its corresponding weight.
5:         Calculate portfolio return realisation at time $t + 1$ with Equation 3.9
6:     Calculate the $\alpha$ percentile of all possible portfolio returns to get the 100$\alpha$% 1-day VaR estimation.
```

As explained in Section 3.5, we want to maximize the log-likelihood $LL(w)$ values when training our MSPN. Therefore, we strive to choose the value for each hyperparameter that gives us the highest log-likelihood values. We ran experiments on the entire dataset for 5, 10, 20 and 40 stock portfolios to pick the value for the minimum number of instances and threshold that yielded the highest average log-likelihood value. Therefore by training the MSPN with different hyperparameter values we were able to decide on using the ones that yielded the highest average log-likelihood value, this was executed for all of our portofolio compositions of 5, 10, 20 and 40 stocks.

In Tables 5.2-5.5 we can see how the exponential of the log-likelihood varies with different values of the hyperparameters where the highest values are marked in bold text. Reasonable

Threshold Min Instances	0.2	0.3	0.4	0.5	0.6
50	1.81e+07	1.73e+07	1.28e+07	1.23e+07	9.98e+06
75	1.79e+07	1.74e+07	1.28e+07	1.23e+07	**9.98e+07**
100	1.75e+07	1.80e+07	1.28e+07	1.23e+07	9.98e+06

Table 5.2: Average $\exp(LL(w))$ values for a five stock portfolio.

Threshold Min Instances	0.2	0.3	0.4	0.5	0.6
50	8.2e+14	1.75e+15	1.96e+15	1.83e+15	8.22e+14
75	8.97e+14	1.58e+15	**1.97e+15**	1.65e+15	8.16e+14
100	9.38e+14	1.70e+15	1.94e+15	1.94e+15	1.72e+15

Table 5.3: Average $\exp(LL(w))$ values for a ten stock portfolio.

Threshold Min Instances	0.2	0.3	0.4	0.5	0.6
50	7.56e+30	**4.54e+31**	1.36e+31	3.68e+30	1.41e+30
75	6.76e+30	3.29e+31	1.29e+31	3.20e+30	1.40e+30
100	3.96e+30	3.19e+31	1.26e+31	3.47e+30	1.37e+30

Table 5.4: Average $\exp(LL(w))$ values for a 20 stock portfolio.

Threshold Min Instances	0.2	0.3	0.4	0.5	0.6
50	1.13e+62	3.62e+63	1.27e+63	7.71e+61	5.41e+60
75	1.86e+62	1.92e+63	8.33e+62	7.71e+61	5.34e+60
100	1.37e+62	**5.81e+63**	8.67e+62	7.68e+61	5.32e+60

Table 5.5: Average $\exp(LL(w))$ values for a 40 stock portfolio.

values for the hyperparameters varies for different sizes of portfolios, for example a good value for minimum instances for ten stock and 20 stock portfolios is 75 and 50. In this project, we used the randomized dependency coefficient (RDC) as described in Chapter 3.6 for our measure of correlation and k-means as a clustering method. In addition, experiments were executed to select the 'Random Generator' parameter for each type of portfolio. We looked at the log-likelihood values from models that had Random generator ranging from 17, 25, 50, 75, 100, 125, 150 and 175. For each type of portfolio a Random Generator parameter value was selected which was associated with the highest average log-likelihood value, the assigned hyperparameter value for each portfolio type are displayed in Table 5.6.

	5 Stock Portfolio	10 Stock Portfolio	20 Stock Portfolio	40 Stock Portfolio
Random Generator	50	125	175	100

Table 5.6: Random Generators values that were assigned for each MSPN model.

The Random Generator hyperparameter did have a noticeable impact on the log-likelihood values which was to be expected because this value has influence in how the both the correlation and decompositioning is calculated. By using a higher number for the Random Generator we are training our model with different decompositions of the data.

5.3 Sliding Window Approach

In this study, we designed an approach to get an in depth understanding for how the MSPN model performs. Instead of using the whole data set from $[0, t]$ to acquire a single VaR estimate we split our data into many segments or windows and used the data in each window to calculate the VaR. The main goal of this approach is to get as many VaR values as possible and assess these values using the backtesting methodologies to find out how the MSPN model performs.

First, we need to decide how many windows, W, we want to have. Preferably, we want as many windows as possible to gain insight in how the MSPN model is performing. When that is done, we train an MSPN model and calculate the VaR value for each window. Each VaR value is then assessed using the Kupiec's Coverage Test where we look at the VaR exceedances in the next window. If a particular VaR value passes the Kupiec's Test then we increase a counter D by one, else the counter is not increased. In total there are N VaR values and we calculate the average

$$\frac{1}{N}\sum_{i=1}^{N} D_i \tag{5.4}$$

The procedure is explained in Algorithm 2 below,

Algorithm 2 Sliding Window Assessment of MSPN model

```
1: procedure SlidingWindowAssessment($\alpha$, $W$)    ▷ Specify the number of windows $W$ and significance $\alpha$
2:     Split the data into $W$ windows
3:     Calculate the non-rejection interval with Kupiec's Test.
4:     for each window do
5:         Train MSPN on data.
6:         Calculate VaR from the MSPN.
7:         Calculate number of VaR exceedances in the next window, $W + 1$.
8:         if VaR estimate is in non-rejection interval then
9:             $D = D + 1$
```

We did the Sliding Window assessment for each portfolio type (5, 10, 20 and 40 stocks) for the Normal Linear model, Historical Simulation model and the MSPN model and compared the results. We examined how the VaR varies across time for each model, how the number of exceedances varied for each model and compared the averages.

Chapter 6

Results

In this chapter we will present the results from our experiments in detail. First, we will look at how well the MSPN captures the heavy tails of the probability distribution of the risk factor returns for one, five, ten, 20 and 40 stock portfolios by looking at the densities from the MSPN provided with inference. Then we will look at how well the MSPN model performs on the entire dataset from 2013-2021 both at the 95% and 99% level and backtest those results. Finally, we will use the Sliding Windows Approach to gain further insight in how well the MSPN performs compared to the other models i.e. how much effect data size and dimensions has on the performance of the MSPN model.

6.1 Density Estimation and Heavy Tails

One important property of the MSPN model that was important to examine was how well the model captures the heavy tails of the P&L distribution. This is important because we want to capture what happens at the extremes, i.e. for high losses and profits. With the assumption of normality in portfolio daily returns these events are considered highly unlikely while the likelihood that one encounters significant deviations from the mean in heavy-tailed distributions is much greater. Therefore, it is now commonly accepted that financial asset returns are in fact, heavy-tailed [Bradley and Taqqu, 2003]. For example, 3σ events (three standard deviations from the mean) may occur with much greater probability when the returns are heavy tailed and for quantile based measures like VaR the heavy tails have a large effect. This is especially true for the highest quantiles (for example 99% VaR) of the daily returns because that is associated with very rare market movements.

The Historical Simulation VaR models use historical returns and therefore it allows for heavy tails without making any assumptions on the probability distributions of the daily returns in the portfolio. However, Historical Simulation VaR models need much data to capture the heavy tails. For example, if we are to calculate a 99% 1-day VaR with 100 data points there is only a single observation in the tail. Finally, the Monte Carlo VaR model allows for heavy tails because of its flexibility in P&L distribution assumptions but capturing the heavy tails requires a lot of computational power (more simulations) and a reasonable assumptions about the P&L distribution.

In order to see how well the MSPN grasps the heavy tails of the P&L distribution we used MSPN inference. After training the MSPN model in each case, we sampled many possible realisations of the portfolio using simulations, i.e. we trained our MSPN model from $[0, t]$ and obtained realisations for the portfolio daily returns at $t + 1$. From the realisations we then calculated the MSPN inference, that means we can calculate the density for a given portfolio return realisation. Using these densities we can then visualize how efficiently the MSPN model grasps the heavy-tailed portfolio data.

In Figure 6.1, we can see the density acquired from MSPN inference in possible portfolio realisations for 5, 10, 20 and 40 stock portfolios. In all cases the MSPN model seems to grasps the

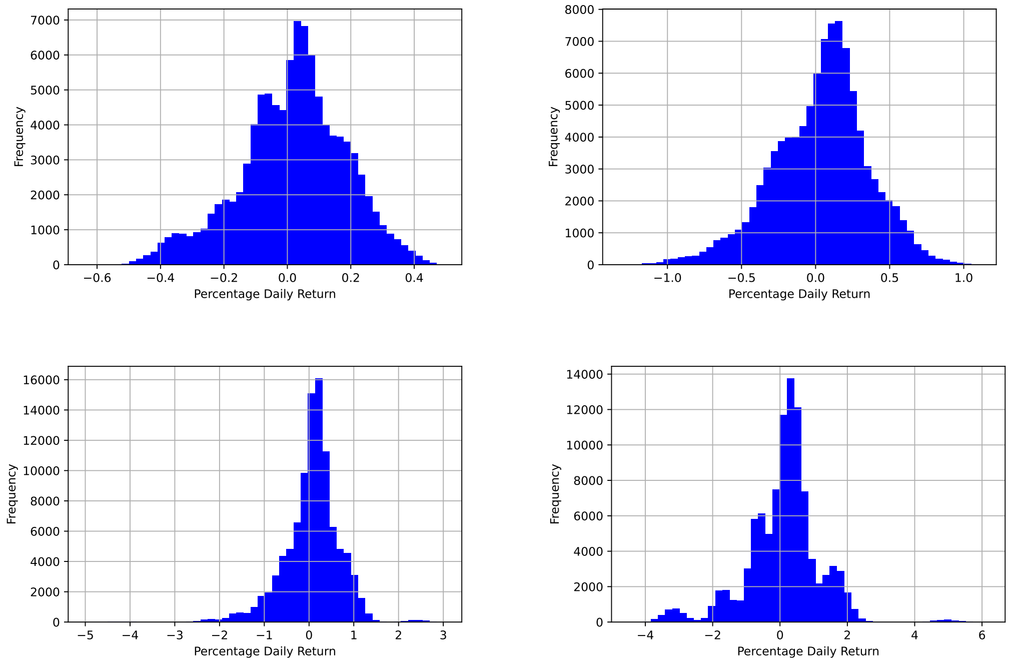


Figure 6.1: Density estimation for 5, 10, 20 and 40 stock portfolios obtained by MSPN inference.

heavy tails of the P&L distribution. This indicates that the MSPN models are capable of catching the extreme events and provide more useful VaR estimates at the higher levels.

6.2 VaR Estimation of Stock Data From 2013-2021

For this section, we calculated the VaR for four different portfolio compositions containing 5, 10, 20 and 40 stocks. All of these stocks were from the S&P 500 stock index and we used the entire data set ranging from 03/07/2013-20/07/2021. Using the entire dataset we produced a single VaR value for all of the three models: MSPN model, Normal Linear VaR model and Historical Simulation VaR model. After acquiring the VaR estimates for the entire period we backtested them using the three different backtesting methodologies explained in Chapter 3. Note that the weights are divided equally for all kinds of portfolios.

The 95% and 99% VaR results are displayed in Table 6.1 and 6.2, one value for each model. For all types of portfolios we executed 100000 simulations to retrieve the possible realisations.

	95% 1-day VaR [%]			
	5 Stock Portfolio	10 Stock Portfolio	20 Stock Portfolio	40 Stock Portfolio
Normal Linear	-2.31	-2.04	-1.91	-1.73
Historical Simulation	-2.32	-1.98	-1.74	-1.56
MSPN	-2.28	-1.96	-1.73	-1.52

Table 6.1: 95% 1-day VaR results for all models.

In Tables 6.1 and 6.2 we can see that for the 95% 1-day VaR the Normal Linear VaR produces the lowest VaR value in all portfolio compositions. However, the MSPN model gives the lowest estimate of VaR for the 99% VaR. Then, we backtested these VaR values over the entire period with three methodologies, the results are displayed in Table 6.6-6.5. From a financial point of view, a risk-averse investor (one who chooses preservation of capital over potential higher-than-

	99% 1-day VaR [%]			
	5 Stock Portfolio	10 Stock Portfolio	20 Stock Portfolio	40 Stock Portfolio
Normal Linear	-3.31	-2.92	-2.75	-2.47
Historical Simulation	-4.21	-3.71	-3.63	-3.27
MSPN	-4.41	-3.49	-3.45	-3.44

Table 6.2: 99% 1-day VaR results for all models.

average return) would choose the most conservative model, which is the MSPN model at the 99% level. However, being conservative comes at a price, the MSPN VaR value at the 99% level is approximately 30% higher than the VaR from the Normal Linear model and therefore would require 30% more capital.

	5 Stock Portfolio: 99% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[9, 33]$	Pass?	Pass?
Normal Linear	46 Exceedances	No	Yes
Historical Simulation	21 Exceedances	Yes	Yes
MSPN	14 Exceedances	Yes	Yes

Table 6.3: Backtesting results for a 99% 1-day VaR of a 5 stock portfolio.

	10 Stock Portfolio: 99% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[9, 33]$	Pass?	Pass?
Normal Linear	33 Exceedances	Yes	Yes
Historical Simulation	60 Exceedances	No	Yes
MSPN	28 Exceedances	Yes	Yes

Table 6.4: Backtesting results for a 99% 1-day VaR of a 10 stock portfolio.

	20 Stock Portfolio: 99% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[9, 33]$	Pass?	Pass?
Normal Linear	45 Exceedances	No	Yes
Historical Simulation	21 Exceedances	Yes	Yes
MSPN	24 Exceedances	Yes	Yes

Table 6.5: Backtesting results for a 99% 1-day VaR of a 20 stock portfolio.

	40 Stock Portfolio: 99% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[9, 33]$	Pass?	Pass?
Normal Linear	41 Exceedances	No	Yes
Historical Simulation	21 Exceedances	Yes	Yes
MSPN	16 Exceedances	Yes	Yes

Table 6.6: Backtesting results for a 99% 1-day VaR of a 40 stock portfolio.

From Tables 6.3-6.6 we can see the entire backtesting results at the 99% VaR level. The MSPN model passes the Kupiec's Test for all portfolio compositions at this level while the Historical

Simulation model fails for a 10 stock portfolio and the Normal Linear model fails for all portfolio types except the 10 stock portfolio. However, all of the models pass the Binomial Distribution test.

	5 Stock Portfolio: 95% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[82, 121]$	Pass?	Pass?
Normal Linear	102 Exceedances	Yes	Yes
Historical Simulation	101 Exceedances	Yes	Yes
MSPN	106 Exceedances	Yes	Yes

Table 6.7: Backtesting results for a 95% 1-day VaR of a 5 stock portfolio.

	10 Stock Portfolio: 95% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[82, 121]$	Pass?	Pass?
Normal Linear	99 Exceedances	Yes	Yes
Historical Simulation	101 Exceedances	Yes	Yes
MSPN	103 Exceedances	Yes	Yes

Table 6.8: Backtesting results for a 95% 1-day VaR of a 10 stock portfolio.

	20 Stock Portfolio: 95% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[82, 121]$	Pass?	Pass?
Normal Linear	87 Exceedances	Yes	Yes
Historical Simulation	101 Exceedances	Yes	Yes
MSPN	103 Exceedances	Yes	Yes

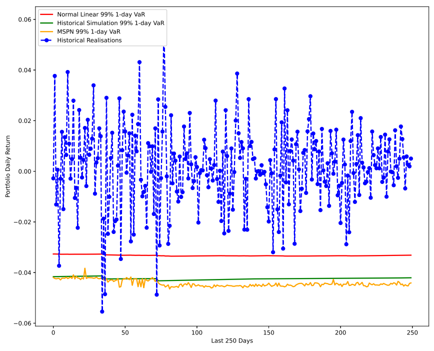
Table 6.9: Backtesting results for a 95% 1-day VaR of a 20 stock portfolio.

	40 Stock Portfolio: 95% 1- day VaR		
	Kupiec's Test		Binomial Distribution Test
	Non-rejection Interval: $[82, 121]$	Pass?	Pass?
Normal Linear	83	Yes	Yes
Historical Simulation	103	Yes	Yes
MSPN	110	Yes	Yes

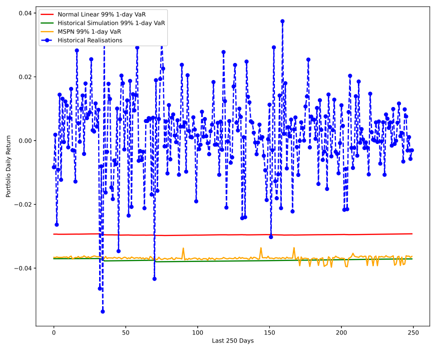
Table 6.10: Backtesting results for a 95% 1-day VaR of a 40 stock portfolio.

For the 95% VaR, we can see that the MSPN also passes the two coverages tests despite having more violations than the other three models. The traditional models do not provide more useful VaR estimates at the 95% VaR level, all of the models have useful estimates and pass the coverage tests. The MSPN model does seem to provide more useful VaR estimates at the 99% level by never failing the Kupiec's Test while the other two VaR models fail in some occasions. The Normal Linear VaR model produces more optimistic VaR estimates at this level resulting in too many exceedances.

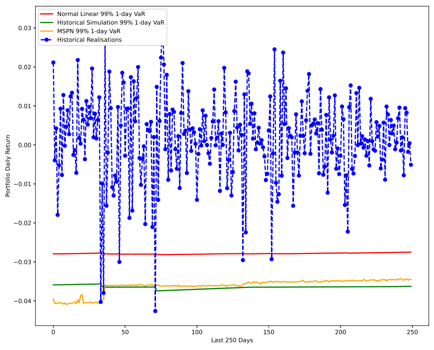
The final backtesting methodology we executed was the Basel Traffic Light Coverage Test which requires the most current 250 day VaR measurements and from these estimates we look at the number of violations, i.e. how often portfolio returns exceeds the VaR. We plotted the latest 250 VaR measurements both for the 95% and 99% level and compared the MSPN with the other models. Note that this test is only used at the 99% VaR level and therefore we cannot conclude anything about the usefulness of the VaR values at the 95% level. However, it is interesting to see multiple predictions, from all models both at the 95% and 99% levels to see how well the models respond to new market movements.



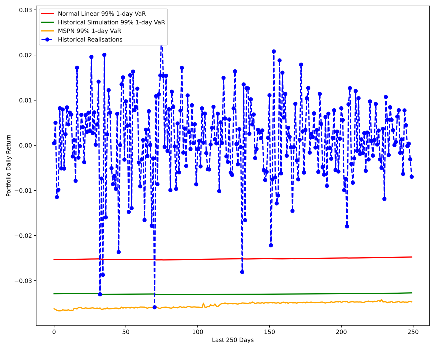
(a) 5 Stock Portfolio.



(b) 10 Stock Portfolio.



(c) 20 Stock Portfolio.



(d) 40 Stock Portfolio.

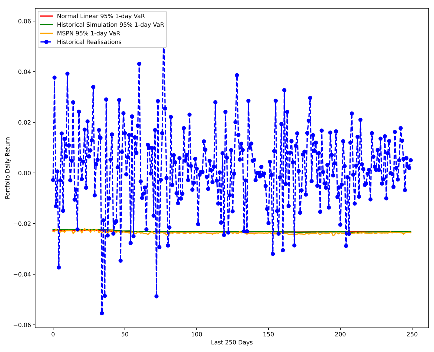
Figure 6.2: Basel Traffic Light Coverage Test 99% VaR results for 5, 10, 20 and 40 stock portfolios.

In Figure 6.2 we can see all 250 VaR estimates for the Normal Linear VaR model, the MSPN model and the Historical Simulation VaR model at the 99% VaR level. We can see that the MSPN model is more sensitive to new data than the other two models, the VaR values in the Historical Simulation model and the Normal Linear VaR model change slowly with each data point added while the MSPN is much more sensitive to newly observed market movements. For example, in Figure 6.2c we can see how the model grasps the two larger portfolio return losses at the beginning of the period and then the VaR values becomes quite lower after these events. The other two models do not seem to grasp this behavior. The results for the test is displayed in Table 6.11. From the table we can see that for the five stock portfolio both the Historical Simulations VaR model and the MSPN model fall into the green zone of the Basel Test while the Normal Linear model falls in the yellow zone. That is also the case for 10 and 20 stock portfolios and finally all models pass the Basel Test for the 40 stock portfolio. It is interesting to see that as the portfolios got larger and more complex the MSPN model had fewer and fewer exceedances.

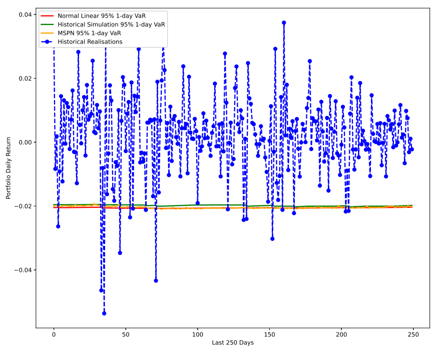
Finally, to gain further insight in the behavior of the models we plotted the 250 most recent VaR values in Figure 6.3 at the 95% level. In this case, the VaR values of the MSPN are more

	Number of Exceedances			
	5 Stock Portfolio	10 Stock Portfolio	20 Stock Portfolio	40 Stock Portfolio
Normal Linear	5	5	6	4
Historical Simulation	3	3	3	2
MSPN	3	3	1	0

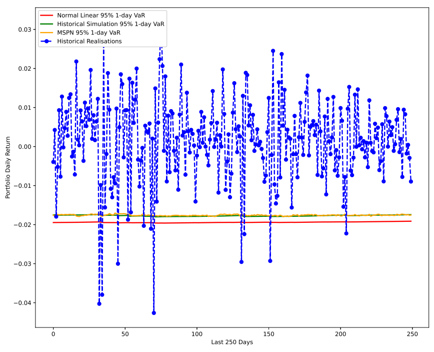
Table 6.11: Basel Traffic Light Results at the 99% level for all models.



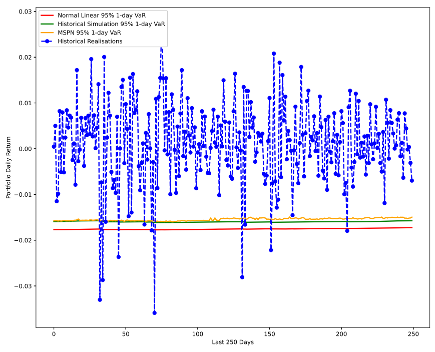
(a) 5 Stock Portfolio.



(b) 10 Stock Portfolio.



(c) 20 Stock Portfolio.



(d) 40 Stock Portfolio.

Figure 6.3: Basel Traffic Light Coverage Test 95% VaR results for 5, 10, 20 and 40 stock portfolios.

similar to the other models than at the 99% level. The Normal Linear model is more conservative then the other two for 20 and 40 stock portfolios. As with the 99% level, the MSPN seems to be more sensitive to new market movements.

6.3 VaR Estimation of Stock Data Using Sliding Window Approach

In this section we are going to present the results obtained by the Sliding Window Approach described in Chapter 5 to get a better look at how the MSPN model performs. For this experiment we looked at a wide range of window lengths, i.e. 93, 119, 156 and 253 days to calculate the VaR. The goal of this approach is to see how well the MSPN model performs with various amounts of data to train on, look at the effect of dimensionality (number of stocks in the portfolio) and compare the performance of all three models. This Sliding Window Approach was only performed at the 95% level because it proved to be too difficult to calculate the non-rejection interval at the 99% level for the smaller window sizes. The reason for that is that to be able calculate the non-rejection interval of the Kupiec's Test, we essentially need to solve a nonlinear equation 3.25 with numerical methods where ideally the equation has two solutions giving us the upper and lower bound of the non-rejection interval. However, at a high level of 99% with limited data (93, 119, 156, 253 observations) we could not calculate the non-rejection interval because the numerical method did not find one. In this setting we are dealing with a nonlinear setting that does not have an analytical solution and we are simply trying to find solutions without knowing if these solutions even exist.

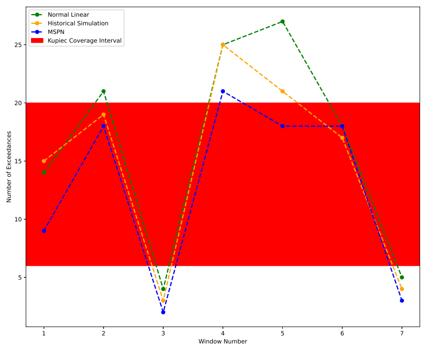
	Accuracy of 95% 1-day VaR estimations for all window sizes for all models.											
	Historical Simulation				Normal Linear				MSPN			
	Number of Stocks				Number of Stocks				Number of Stocks			
$\|W\|$	5	10	20	40	5	10	20	40	5	10	20	40
93	76.1	**66.7**	52.3	57.1	**85.7**	**66.7**	**66.7**	52.4	71.4	61.9	52.3	0.0
119	75.0	43.8	56.3	**62.5**	**81.3**	81.3	**62.5**	43.8	**81.3**	43.8	50.0	43.8
156	58.3	58.3	50.0	41.2	**75.0**	58.3	50.0	33.3	**75.0**	**66.6**	**58.3**	**50.0**
253	42.8	**28.2**	28.6	**28.5**	28.6	14.3	14.3	0.0	**57.1**	14.3	**42.8**	14.9
Average of scores	63.1	49.3	46.8	**47.3**	67.7	**55.2**	48.4	32.4	**71.2**	46,7	**50,9**	27.2

Table 6.12: Sliding Window Approach accuracy results for all models.

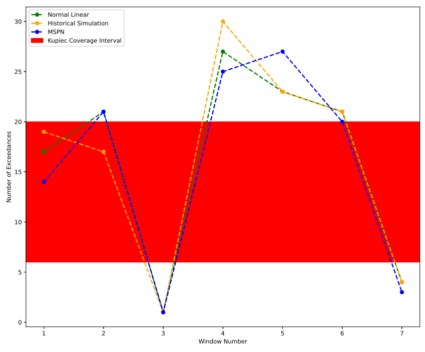
In Table 6.12, the accuracy results described with Equation 5.4 are displayed for each model where the bolded values are the highest accuracies achieved for each scenario. We can see that the MSPN model does not perform well with such limited data and with the smallest window size $|W| = 93$, the MSPN performs the worst of all three models. For $|W| = 119$ the MSPN performs best for the 5 stock portfolio and for $|W| = 156$ the MSPN model achieves the best accuracy results for all portfolio types stocks. Finally at $|W| = 253$, the MSPN performs the best out of the three. From this we can see that as the window got larger, i.e. the MSPN model predicts with more data, the MSPN achieved better and better results. Therefore as the window gets larger the MSPN achieves better results. In Figure 6.4 we can see the number of exceedances for each model along with the Kupiec's non-rejection interval. This interval remains constant for all windows as it depends merely on the data size and the significance level α of interest. The Normal Linear VaR model does worse as the portfolio contained more stocks (higher dimensions) and the window sizes got bigger. In addition, we can see that exceedances become less similar as the portfolio becomes more complex while being similar on the 5 and 10 stock portfolios indicating that it is more difficult to assess the VaR at higher dimensions. All of the three models do worst on the highest dimension portfolios, as the dimension goes up more data is needed to get more useful VaR estimates. To conclude, the MSPN model does not perform well at the smallest window size because by using such limited data we are not using the full power of the MSPN. For the MSPN model to learn joint distributions of high dimensionality there are more parameters to estimate and therefore it needs an abundance of data and 93 observations appears to be too limited. However, by simply looking at the average of scores from Table 6.12 the MSPN model performed the best of all three models in the Sliding Window Approach.

In Figure 6.5 the VaR values for the largest window size of each model is plotted for each portfolio type. One thing to notice is that the exceedances behavior becomes more dissimilar as the number of stocks gets larger demonstrating the effect of dimensionality and this effect impacts the MSPN model most of all three models.

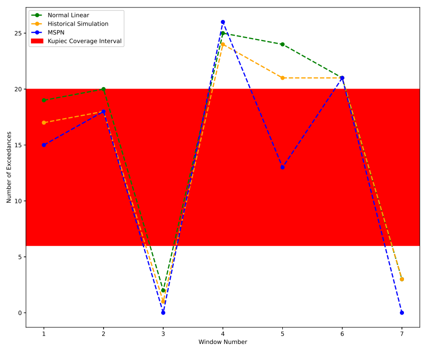
When we look at the smallest window size of $|W| = 93$ for all portfolio compositions we can see the MSPN shortcomings clearly. The VaR values obtained by the MSPN for the 40 stock portfolio



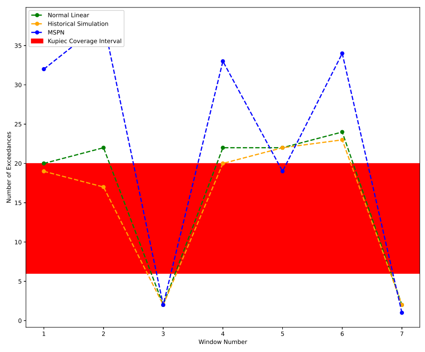
(a) Exceedances For a 5 Stock Portfolio.



(b) Exceedances For a 10 Stock Portfolio.



(c) Exceedances For a 20 Stock Portfolio.



(d) Exceedances For a 40 Stock Portfolio.

Figure 6.4: Number of exceedances along with the Kupiec's non-rejection interval for each window and portfolio types for all three models with window size $|W| = 253$.

is very dissimilar than the other two models being too optimistic leading to a higher amount of exceedances. The MSPN model has an accuracy of 0.0% with the smallest window size yet again indicating it heavily depends on larger amounts of data. However, the MSPN model does have an accuracy above 50% for 5, 10 and 20 stock portfolios and the VaR prediction behavior is similar to the other two at this level. In Figure 6.6 we can see how the VaR estimates of the three models behave similarly for all models with 5 and 10 stock portfolios but that is not the case for the 40 stock portfolio, there the MSPN has more optimistic VaR estimates.

If we compare Figures 6.5 and 6.6 we can see more clearly that the MSPN most often has more optimistic predictions as the dimensionality goes up, for example for the 40 stock portfolio in Figures 6.5d and 6.6d the MSPN has in almost all cases the most optimistic VaR estimate.

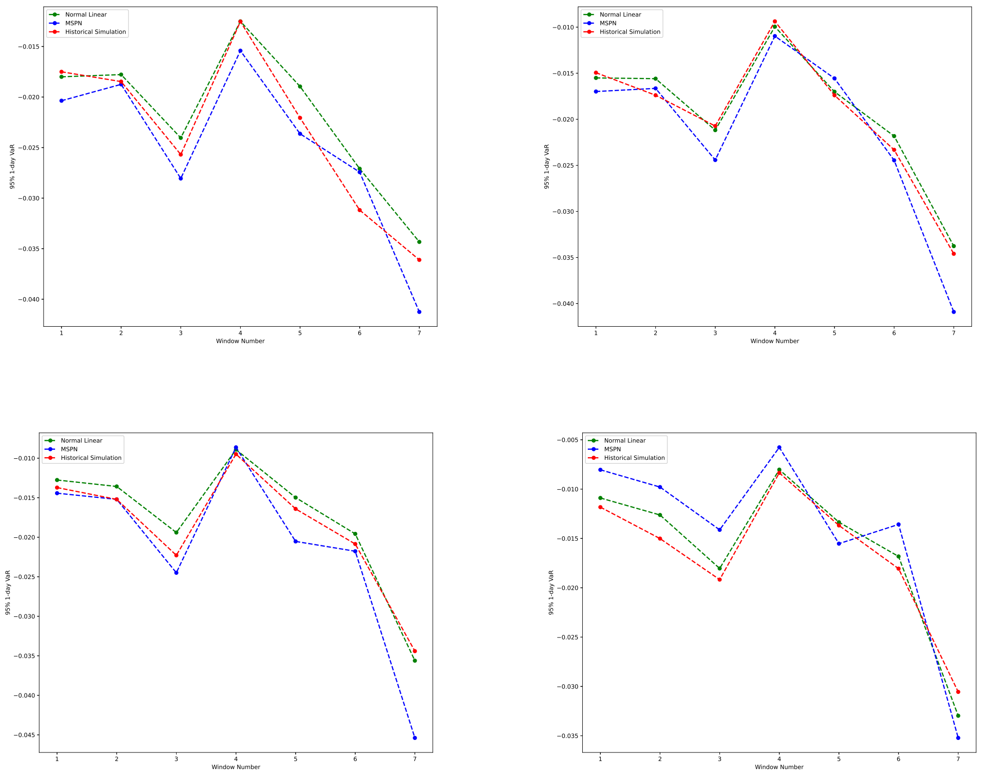


Figure 6.5: Estimated VaR values of each window for the largest window size $|W| = 253$.

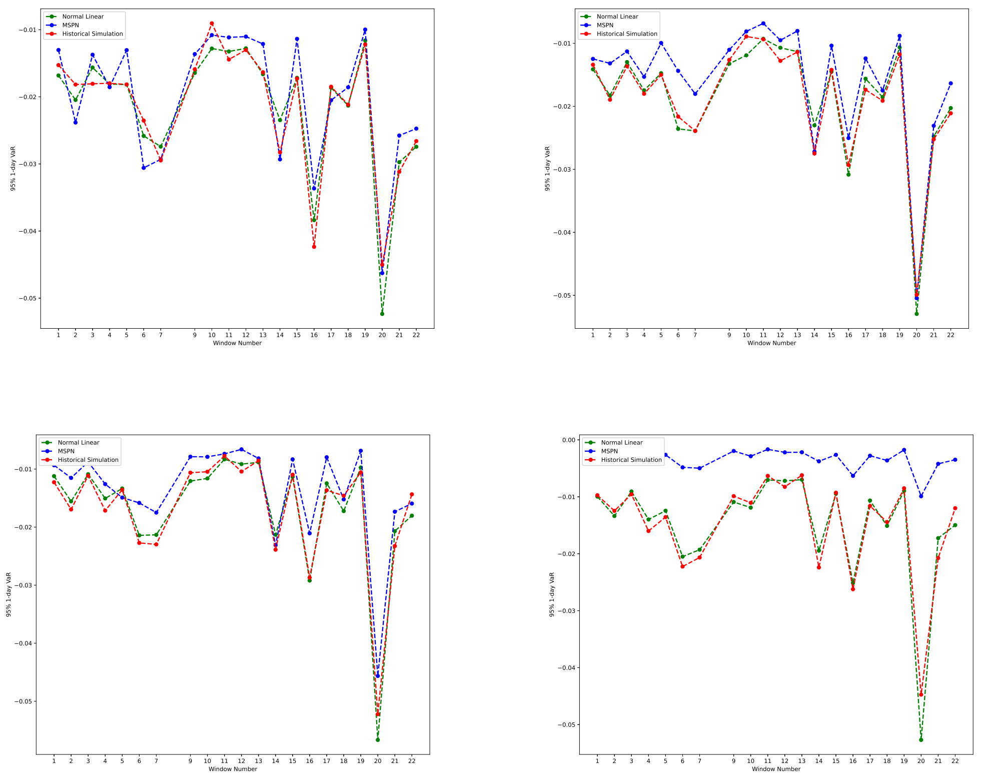


Figure 6.6: Estimated VaR values of each window for the smallest window size $|W| = 93$.

Chapter 7

Conclusions

In this project we provided a new way to estimate the VaR risk of portfolios. Other well established model architectures like the Normal Linear, Historical Simulation and Monte Carlo VaR models have underlying assumptions that are most often not realistic while the MSPN model designed in this project does not. The MSPN model proved to be a realistic alternative to the other models, providing useful VaR estimates while having no underlying assumptions of the P&L distribution. However, the MSPN model has the underlying assumption that the portfolio's daily returns are independent and identically distributed which means that the observations are independent of each other and that potential returns for each stock has the same probability distribution. This means that our model does not include the time factor and has no "memory" of previous observations. In addition the MSPN model is not the ideal model for every scenario, especially for small amount of observations and requires heavy computational power for the higher dimensions.

We applied an MSPN architecture, a special combination of SPNs and piecewise polynomials, which allows for tractable inference without any assumptions for the P&L distribution. This architecture provides a new way to train and learn multivariate mixed distributions, which is represented as risk factor returns in portfolio theory, in a more effective way than other methods. Our results indicate that the MSPN model is capable of capturing the heavy tails of the risk factor returns and by doing that we are more capable of capturing the extreme events.

First, we provided VaR estimates using the entire dataset from 2013-2021 both at the 95% and 99% level. At the 95% level the MSPN provided useful and competitive VaR estimates passing the Kupiec and Binomial Distribution Test for all portfolio compositions. However, at the 95% level, there is no clear advantage of the MSPN model. In addition, we estimated the 99% 1-day VaR with all three models and used the two previously mentioned backtesting strategies and the Basel Traffic Light Test. The results from the 99% VaR estimates and the Basel Traffic Light Test indicate that the MSPN model is able to capture extreme events with better performance than the other models, which can be seen from the exceedances in Table 6.6 and the MSPN behavior in Figure 6.2. The MSPN VaR estimations appears to be more sensitive to new observations (market movements) than the other models and catches extreme events more frequently than the other models, this behavior is demonstrated in Figure 6.2c.

In addition, we conducted a Sliding Window approach to examine more carefully how the MSPN model performs by backtesting many VaR estimates with various window sizes and different portfolios compositions. On average, the MSPN model does not perform better than the other models with the smallest window size but performs best of all three with the largest window size. The MSPN model performs considerably worse than the other models for high dimensional portfolios with the smallest window size, failing every Kupiec Test. For the 40 stock portfolio, the MSPN model provided too optimistic VaR estimations leading to frequent exceedances. However, as the window size gets larger, the MSPN model provides more useful VaR estimations for all portfolio compositions. It proved to be very difficult to calculate acceptable VaR estimations for the 40 stock portfolio with such small data. Overall, the MSPN model had the highest average score of all three models, it had the highest average of scores two out of four times for all portfolio types as we can see in the bottom row of Table 6.12.

We can conclude that the MSPN model is a promising alternative for VaR estimation. It provides a way to calculate the VaR without using any parametric assumptions and is much more sensitive to new market movements when dealing with large data, especially at the 99% level. The Basel Traffic Light Test demonstrated that the MSPN was able to capture the extreme events better than the traditional models which is desirable feature because these events happen frequently with heavy-tailed distributions. This property is also demonstrated in Figures 6.1 where we used MSPN inference to see how well the MSPN captures the heavy tails by using possible realisations of the portfolio returns. However, the MSPN model does not seem to provide useful estimations when using smaller amounts of data at a time like in the Sliding Window Approach. The MSPN model needs to train on a considerable amount of data to fully capture the joint distribution, especially at the higher dimensions. This can be problematic because as observations gets older they should have less and less effect on tomorrow's realisations. This means, that we would not expect observations from many years ago to have a large effect on tomorrow's realisation of some stock, that should depend on more recent data. The MSPN model does not take this factor into consideration, an observation that is for example a few years old has as much impact on how the MSPN model is trained as a day old observation.

Chapter 8

Future Work

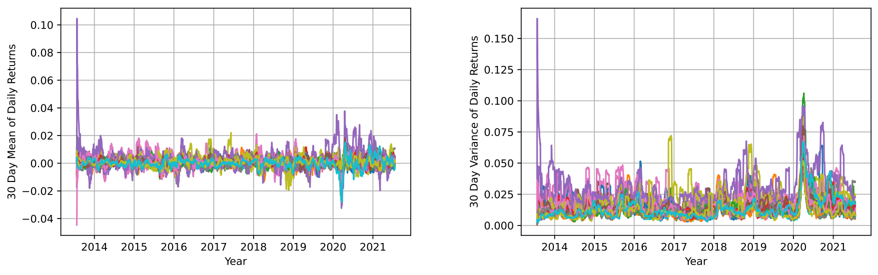
In this chapter we will discuss and present ideas that are possible extensions of this project. We will discuss how to incorporate time-dependent MSPNs, called Conditional Mixed Sum Product Networks, and discuss underlying assumptions that we use to deal with the financial time-series in our project and how we can deal with these assumptions.

8.1 Stationary Time Series

Dealing with financial time-series is different from the more traditional classification and regression predictive modeling problems. The temporal aspect adds an order to the observations that we are handling, e.g. the order of realisations for a particular stock has a strict temporal order.

We can define a stationary time series a series whose properties do not depend on the time at which the series is observed. More precisely, a stationary time series is a series whose statistical properties such as the mean, variance, etc. are all constant over time. Time series can be rendered as approximately stationary with mathematical transformations. The predictions retrieved from the stationarized series (with transformations) can then be untransformed by reversing the transformation previously used resulting in predictions for the original series.

In this project, we assumed that the financial series data was stationary, i.e. that the summary statistics remained constant over time (2013-2021). In Figure 8.1 we can see the mean and variance of our financial stocks where we chose a window size of 30 days, i.e. we calculate the mean and standard deviation with 30 days intervals.



(a) 30 day mean of stocks (b) 30 day standard deviation of stocks

Figure 8.1: Mean and standard deviation of all 40 stocks with a 30 day window.

From Figure 8.1 we can see that the mean and standard deviation fluctuate to some extent from 2014-2021 and especially in the extreme events, e.g. Covid-19 in the beginning of 2020. To conclude, a potential direction of research for this project would therefore be to figure out how to incorporate non-stationary time series for MSPN models.

8.2 Conditional MSPNs

Using the MSPN model we were able to learn the joint probability distribution of our portfolio daily returns to sample possible realisations and calculate the 1-day VaR. However, when we trained our MSPN on our financial data we did not take into account the time factor, i.e. we assumed that the daily returns of all stocks were i.i.d. and the temporal order of the observations had no effect on the training of our model. The natural next direction of this project would therefore be to include conditionality. For example, if we want to predict the VaR for one stock A, we would learn the conditional probability $\mathbb{P}(A(t)|A(t-1), A(t-2), ..., A(t-N))$, where N is a predetermined number of have much data we should take into account, instead of learning $\mathbb{P}(A(t))$ like we did in this project. By doing this we would be adding the temporal aspect and the model would take into account the order of our observations when training. The next step would therefore be to design a Conditional Mixed Sum Product Network (CMSPN).

In [Shao et al., 2019], they stated that in order to include the time factor of a time series into an SPN we can either use an existing SPN structure and condition on its parameters on the input (e.g. neural networks) or use a structure-learning approach that they proposed that derives both the structure and parameters of Conditional Sum Product Networks (CSPNs) from data. As explained in [Molina et al., 2018], this means that to learn a CMSPN $\mathbb{P}(Y|X)$ we can use a neural network g to make the CMSPN functional of its parameters.

For experimentation, we tried to include the time factor of our financial data by conditioning on its parameters on the input using a convolutional neural network. In [Mariani et al., 2019], they processed their financial time series with a 1-D convolutional neural network. These networks are an effective tool to process time series according to [Bai et al., 2018]. The deep-learning network processes the time information by convolving along the time dimension.

	1-day 95% VaR	Number of Exceedances	Kupiec's Test
CMSPN	-2.19%	82	No
MSPN	-2.98%	41	No
Historical Simulation	-2.98%	41	No
Normal Linear VaR	-2.97%	42	No

Table 8.1: 95% VaR and Kupiec's Test results with a non-rejection interval of [12,30] for all four models.

We designed a 1-D convolutional neural network with a windows size of 2, four convolutional layers with 64 filters and *ReLu* as an activation function and tested it on a 10 stock portfolio. The output of the convolutional neural network is then used as parameters for the CMSPN which is then trained and VaR derived as before on the realisations at time $t + 1$. By using 60/40 train and test split we were able to train the CMSPN on 403 observations to learn the joint probability distribution of the daily percentage returns.

In Table 8.1 we can see the 95% 1-day VaR and backtesting results for a 10 stock portfolio. The CMSPN model produced a slightly more optimistic VaR value while the other three produce a similar VaR estimation. However, all of models fail the Kupiec's Test with a non-rejection interval of [12;30] for 403 observations at the 95% level. In [Shao et al., 2019], it was mentioned that this approach of using a convolutional network might misrepresent the conditional independence present in the data. Therefore, we can conclude that in our case using convolutional neural network data for preprocessing did not seem to get a more useful 1-day 95% VaR result. To conclude, a possible direction of research for this project is to find a way to design a CMSPN and by doing that we can add the temporal aspect and exclude the assumption of independent and identically distributed returns in addition to parametric assumptions of the P&L distribution.

Bibliography

[Alexander, 2008a] Alexander, C. (2008a). *Market Risk Analysis, Quantitative Methods in Finance*, volume 1. John Wiley & Sons. 7

[Alexander, 2008b] Alexander, C. (2008b). Market Risk Analysis, Value at Risk Models (Volume IV). vii, 1, 3, 9

[Anantharam et al., 2013] Anantharam, V., Gohari, A., Kamath, S., and Nair, C. (2013). On maximal correlation, hypercontractivity, and the data processing inequality studied by erkip and cover. *arXiv preprint arXiv:1304.6133*. 16

[Artzner et al., 1999] Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). Coherent Measures of Risk. *Mathematical finance*, 9(3):203–228. 6

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*. 19, 44

[Bradley and Taqqu, 2003] Bradley, B. O. and Taqqu, M. S. (2003). Financial risk and heavy tails. In *Handbook of heavy tailed distributions in finance*, pages 35–103. Elsevier. 31

[Chavira and Darwiche, 2007] Chavira, M. and Darwiche, A. (2007). Compiling bayesian networks using variable elimination. In *IJCAI*, volume 2443. 13

[Engle and Manganelli, 2004] Engle, R. F. and Manganelli, S. (2004). Caviar: Conditional autoregressive value at risk by regression quantiles. *Journal of business & economic statistics*, 22(4):367–381. 21

[Gens and Domingos, 2012] Gens, R. and Domingos, P. (2012). Discriminative learning of sum-product networks. *Advances in Neural Information Processing Systems*, 25:3239–3247. 15

[Gens and Pedro, 2013] Gens, R. and Pedro, D. (2013). Learning the structure of sum-product networks. In *International conference on machine learning*, pages 873–880. 14, 15

[Grody et al., 2013] Grody, A. D., Fernandes, K. J., Hughes, P. J., and Steven Toms, J. (2013). Basel committee's fundamental review of the trading book: A commentary. *Journal of Risk Management in Financial Institutions*, 6(1):6–9. 12, 13

[He et al., 2018] He, K., Ji, L., Tso, G. K., Zhu, B., and Zou, Y. (2018). Forecasting exchange rate value at risk using deep belief network ensemble based approach. *Procedia computer science*, 139:25–32. vii, 20

[Holton, 2003] Holton, G. A. (2003). *Value-at-risk*. Acad. press. 12

[Kambhu et al., 1996] Kambhu, J., Keane, F., Benadon, C., Remolona, E. M., Bassett, W., Geoum, I. S., and Hendricks, D. (1996). Economic policy review. 10

[Kupiec, 1995] Kupiec, P. (1995). Techniques for verifying the accuracy of risk measurement models. *The J. of Derivatives*, 3(2). 10

[Lehmann and Romano, 2005] Lehmann, E. and Romano, J. P. (2005). Multiple testing and simultaneous inference. *Testing Statistical Hypotheses*, pages 348–391. 12

[Li et al., 2020] Li, Z., Tran, M.-N., Wang, C., Gerlach, R., and Gao, J. (2020). A bayesian long short-term memory model for value at risk and expected shortfall joint forecasting. *arXiv preprint arXiv:2001.08374*. vii, 21

[Lopez-Paz et al., 2013] Lopez-Paz, D., Hennig, P., and Schölkopf, B. (2013). The randomized dependence coefficient. *arXiv preprint arXiv:1304.7717*. 16

[Mariani et al., 2019] Mariani, G., Zhu, Y., Li, J., Scheidegger, F., Istrate, R., Bekas, C., and Malossi, A. C. I. (2019). PAGAN: Portfolio Analysis with Generative Adversarial Networks. *arXiv preprint arXiv:1909.10578*. vii, 1, 19, 44

[Markowitz, 1991] Markowitz, H. M. (1991). Foundations of portfolio theory. *The journal of finance*, 46(2):469–477. 1

[Molina et al., 2018] Molina, A., Vergari, A., Di Mauro, N., Natarajan, S., Esposito, F., and Kersting, K. (2018). Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. 15, 16, 44

[Peharz et al., 2020] Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., and Ghahramani, Z. (2020). Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, pages 334–344. PMLR. 15

[Póczos et al., 2012] Póczos, B., Ghahramani, Z., and Schneider, J. (2012). Copula-based kernel dependency measures. *arXiv preprint arXiv:1206.4682*. 16

[Poon and Domingos, 2011] Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE. 14, 15

[Rényi, 1959] Rényi, A. (1959). On measures of dependence. *Acta Mathematica Academiae Scientiarum Hungarica*, 10(3-4):441–451. 16

[Roccioletti, 2015] Roccioletti, S. (2015). *Backtesting value at risk and expected shortfall*. Springer. 7, 10, 12

[Rozenholc et al., 2010] Rozenholc, Y., Mildenberger, T., and Gather, U. (2010). Combining regular and irregular histograms by penalized likelihood. *Computational statistics & data analysis*, 54(12):3313–3323. 17

[Sánchez-Cauce et al., 2021] Sánchez-Cauce, R., París, I., and Díez, F. J. (2021). Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 13

[Shao et al., 2019] Shao, X., Molina, A., Vergari, A., Stelzner, K., Peharz, R., Liebig, T., and Kersting, K. (2019). Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures. *arXiv preprint arXiv:1905.08550*. 44

[Trapp et al., 2019] Trapp, M., Peharz, R., Ge, H., Pernkopf, F., and Ghahramani, Z. (2019). Bayesian learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 6347–6358. 3

[XIONG, 2018] XIONG, W. (2018). Machine learning in financial market risk: Var exception classification model. vii, 22, 23