TECHNISCHE
UNIVERSITÄT
WIEN

DIPLOMARBEIT

Neuronale Netze zur SCR Berechnung

ausgeführt am

Institut für
Finanz und Versicherungsmathematik
der Technischen Universität Wien

unter der Anleitung von

Univ.Prof. Dipl.-Math. Dr.rer.nat. Thorsten Rheinländer

durch

Maria-Theresa Baumgarten,BSc
Matrikelnummer: 01126282

Wien, am 12. August 2019

(Betreuer) (Verfasser)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Kurzfassung

Seit 1.1.2016 sind Versicherungsunternehmen, im Rahmen der Solvency II Richtlinien dazu verpflichtet die Solvenzkapitalanforderungen zu berechnen. Dafür ist die Berechnung der Marktwerte der Verpflichtungen zum Zeitpunkt Null sowie zum Zeitpunkt Eins notwendig, was mittels Nested Monte Carlo Methode durchführbar ist. Diese Methode benötigt jedoch erheblichen Rechenaufwand, weshalb nach einer geeigneten Alternative gesucht wird.
In dieser Arbeit wird, anhand zweier kleiner Portfolios aus sogenannten Select Produkten, die Möglichkeit behandelt, die inneren Schleife der Nested Monte Carlo Methode durch künstliche Neuronale Netze zu ersetzen. Der Fokus liegt dabei auf dem Aufbau eines solchen Netzes.
In diesem Rahmen wird getestet, wie sich zum einen die Verwendung verschiedener Aktivierungsfunktionen zum anderen die Kombination aus Knoten und Layern des Netzes auf die Performance des Modells auswirken. Dabei konnte beobachtet werden, dass jene Aktivierungsfunktionen, deren Definitionsbereich zwar beschränkt, aber sowohl negative als auch positive Werte beinhaltet, für beide Datensätze im Vergleich sehr gute Ergebnisse liefern. Für die Wahl der Anzahl der Knoten beziehungsweise Layer des Netzes ergibt sich, dass eine geringe Anzahl an Layern mit mehr Knoten präferiert werden sollte.
Der Einsatz Künstlicher Neuronaler Netze zur Berechnung des SCR, beschleunigt dies maßgeblich, da ein großter Teil der Monte Carlo Simulationen ersetzt wird.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Abstract

Due to the introduction of the Solvency II regulations, insurance businesses are obligated to calculate the so called Solvency Capital Requirement which represents a measure for the solvency of an insurer. This requires the evaluation of the market values of liabilities at time zero as well as at time one. A commonly method for computing this values is by so called Nested Monte Carlo Simulations which is very time consuming. Therefore it is necessary to find a suitable alternative.
In this thesis the idea is, to substitute the inner loop of the Nested Monte Carlo method by a feed forward neural network. The fokus thereby lies on the architecture of the used networks.
In that context different tests are computed on two sets of Data from simple portfolios of so called select products. Neural networks with different activation functions are computed and the resulting losses are compared. As result it can be said, that using those activation functions with bounded target set containing positive as well as negative values produce relatively low losses compared to others.
Furthermore the interaction of number of layers and units per layer is discussed. The conclusion of this test is that in general it is preferable to use models with more units per layer but lesser layers.
Using feed forward neural networks for calculating the Solvency Capital Requirement results in a relevant reduction of computational time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mir bei der Erstellung dieser Arbeit, sowie auf meinem Weg durch das Studium, sowohl fachlich als auch menschlich, zur Seite gestanden sind.
Ein besonderer Dank geht dabei an Prof. Dipl.-Math. Dr.rer.nat. Thorsten Rheinländer, für die ausgezeichnete Betreuung meiner Diplomarbeit. Ich konnte viel neues Wissen aus dieser Zeit mitnehmen.
Weiters bedanke ich mich bei meiner Familie, insbesondere bei meinen Eltern, die mich durch meine Zeit auf der TU laufend unterstützt und immer wieder aufgebaut haben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 12. August 2019

Maria-Theresa Baumgarten

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Inhaltsverzeichnis

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

1 Einleitung

Machine Learning und besonders sogenannte Künstliche Neuronale Netze stellen eine inovative Alternative zu statistischen Standardmethoden dar. Diese bringen besonders in Fällen von sehr großen Datensätzen einige Vorteile mit sich. Nicht nur konnte damit bereits in diversen Anwendungen bessere Ergebnisse erzielen werden ([2, Breiman]), sondern auch den Rechenaufwand deutlich verringern.

Die Finanzbranche ist dabei eine der Bereiche, in denen Machine Learning Anwendungen findet, besonders bei Vorhersagen von Entwicklungen am Markt. Dies liegt darin begründet, dass diese Methoden darauf basieren, aus bekannten Daten Muster zu erkennen um diese in Folge auf neue Daten anzuwenden.

Ein Teilgebiet der Finanzbranche, ist die Berechnung der Solvenzkapitalanforderungen (SCR) für Versicherungsunternehmen. Dies stellt ein Maß für die Solvabilität dieser Unternehmen. dar und muss seit der Einführung von Solvency II am 1.1.2016 von jedem Versicherungsunternehmen jährlich berechnet werden. Das SCR ist dabei jener Wert, der einem Unternehmen an Reserven zur Verfügung stehen sollte, um mit einer Sicherheit von 99.5% alle Verpflichtungen decken zu können, die während dem Folgejahr anfallen könnten. Damit ist das SCR jener Wert den das Eigenkapital jetzt haben sollte, damit es mit einer Wahrscheinlichkeit von 99.5% in einem Jahr noch positiv ist.
In dieser Arbeit wird zur Vereinfachung der Berechnung angenommen, dass das Versicherungsunternehmen eine passive Anlagestrategie verfolgt, wodurch sich das Problem auf die Bewertung des Marktwertes der Verpflichtungen des Unternehmens zum Zeitpunkt 0 und des Marktwertes der Verpflichtungen zum Zeitpunkt 1 einschränken lässt.
Eine genauere Beschreibung der Berechnung des SCR findet sich in Kapitel 2

Eine weit verbreitete Methode, die auf diese Problemstellung angewandt wird, ist die, in Kapitel 3.3 beschriebene, Nested Monte Carlo Methode. Diese Methode ist äußerst rechenaufwendig, wodurch laufend nach einer passenden Alternative gesucht wird.

Mit der Nested Monte Carlo Methode wird in der äußeren Schleife der Marktwert der Verpflichtungen zum Zeitpunkt 0 berechnet und mit einer inneren Schleife, für jeden Pfad der äußeren, der Marktwert zum Zeitpunkt 1. Damit entsteht unter Verwendung der inneren Schleife ein Vektor dessen Dimension gleich der Pfade der äußeren Schleife ist. Jeder Vektoreintrag stellt einen Marktwert der Verpflichtungen zum Zeitpunkt 1 dar, der ebenfalls durch eine Monte Carlo Simulation berechnet wurde. Diese wiederholten Monte Carlo Simulationen resultieren in einem sehr hohen Rechenaufwand für das SCR. Daher war die Idee dieser Arbeit, die innere Schleife durch einfeed forward Neuronales Netz zu ersetzen (Siehe 4.1.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Wie in Kapitel 6.1 beschrieben, wird also nur ein kleiner Teil der Marktwerte der Verpflichtungen zum Zeitpunkt 1 mit Hilfe Monte Carlo Simulationen berechnet. Diese werden in Folge für das Training des Netzes und zur Modellkalibrierung genutzt. Die restlichen Marktwerte zum Zeitpunkt 1 werden im Anschluss mit Hilfe des Neuronalen Netzes berechnet. Was im Vergleich zur Monte Carlo Methode einen kleinen Bruchteil der Rechenzeit in Anspruch nimmt.

Was jedoch potentiell zeitaufwendig sein könnte, ist die Suche nach der optimalen Architektur des Netzes, da hier die Wahl der Aktivierungsfunktion, die Anzahl der Layer, die Knotenzahl pro Layer und viele weitere Faktoren eine Rolle spielen.

Kapitel 6 beschreibt die Ergebnisse aus verschiedenen Tests. Dabei werden 2 kleine Portfolios aus sogenannten Select-Produkten betrachtet. Diese Produkte sind, besonders im derzeitigen Niedrigzinsumfeld, eine alternative zur klassischen Rentenversicherung, da sie die Möglichkeit einer garantierten Verzinsung mit der Spekulation auf einen zugrundeliegenden Index vereinen. Eine genauere Beschreibung dieser Produkte ist in Kapitel 5 zu finden.

Bei den Tests hinsichtlich der Netzwerkarchitektur werden zuerst verschiedene Aktivierungsfunktionen betrachtet. Danach wird die Frage behandelt, wie sich die Anzahl der Layer, sowie die Knotenzahl pro Layer auf die Performance auswirkt. Zuletzt wird kurz diskutiert, wie sich der quadratische Fehler für einen Validierungsdatensatz, also Daten die nicht zum Training des Netzes verwendet wurden, verhält, wenn sich zum einen die Zahl der Layer bei gleichbleibender Knotenzahl pro Layer zum anderen die Knotenzahl pro Layer bei gleichbleibender Anzahl der Layer verändert.
Zum Schluss werden für beide Portfolios jene Netze mit den zwei besten Aktivierungsfunktionen, resultierend aus Kapitel 6.2, und der jeweils besten Anzahl an Knoten beziehungsweise Layer, zur Berechnung des SCR verwendet.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2 Solvency Capital Requirement (SCR)

Mit 1.1.2016 traten die neuen Regelungen unter Solvency II in Kraft. Diese sollten eine risikobasierte, einheitliche Vorschrift zur Aufsicht europäischer Versicherungsunternehmen darstellen. Solvency II soll dabei einen Wandel einem regelbasierten System hin zu einer auf Prinzipien aufgebauten Regulierung darstellen.
Dieses ist, in Anlehnung an die Aufsichtsregelungen für Banken, Basel II&III, in drei Säulen aufgeteilt: Vorgaben zu den Kapitalanforderungen, Governance-Vorschriften und Veröffentlichungs- und Meldevorschriften. Im Rahmen der ersten Säule - Vorgaben zu den Kapitalanforderungen - sind zwei Größen von Bedeutung, die Mindestkapitalanfoderungen (MCR - Minimum Capital Requirements) und die Solvenzkapitalanforderungen (Solvency Capital Requirements). Während das MCR ein Maß für die finanzielle Lage eines Unternehmens darstellt, das der Aufsicht eine unmittelbare Reaktion auf dessen Veränderung ermöglichen soll, wird das SCR als Indikator für die Solvabilität eines Unternehmens gesehen. Es soll des Weiteren Unternehmen einen Anreiz bieten, ihre Risiken gewissenhaft zu managen[4, EIOPA].
Das SCR stellt dabei jenen Wert dar, den ein Unternehmen an Reserven zu bilden hat, um mögliche Verluste, welche innerhalb eines Jahres auftreten können mit einer Sicherheit von 99.5% abdecken zu können[14, Nilsson, Sandberg]. Das bedeutet das Unternehmen hat zu jeder Zeit Kapital in der Höhe des einjährigen 99.5% Value at Risk (VaR) zu halten, um eventuelle Verluste im kommenden Jahr zu vermeiden. Dies kann dargestellt werden als:

$$SCR = \underset{x}{argmin}\left(\mathbb{P}\left(AC_1 \geq 0 | AC_0 = x\right) \geq 0.995\right) \tag{2.1}$$

mit

t ... Zeit in Jahren
AC_t ... Eigenkapital (Available Capital)

[8, Hejazi et al.]

Um den Anforderungen einer risikobasierten Regelung gerecht zu werden, basiert die vorgegebene Standardformel für das SCR auf einer vollumfassenden Herangehensweise an (quantifizierbare) Risiken. Dies steht im Gegensatz zum bisherigen Faktor-Ansatz zur Berechnung der Risikomarge aus Solvency I und verzichtet des Weiteren auf die implizite Vorsicht in der Bestimmung der technischen Rückstellungen, aus Solvency I [4, EIOPA].
Die Bestimmung des SCR nach der Standardformel bringt jedoch für viele Unternehmen einige Schwierigkeiten mit sich, wodurch viele von ihnen die Möglichkeit, ein internes Modell dafür zu entwickeln, in Betracht ziehen[8, Hejazi et al.].
Eine Methode zu diesem Zweck ist die Nested Monte Carlo Methode, welche in [1, Bauer et al.] vorgestellt wird.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Da die Berechnung der bedingte Wahrscheinlichkeit aus Definition (2.1) für das SCR in der Praxis schwierig umzusetzen ist, schlagen [1, Bauer et al.] eine äquivalente Darstellung des SCR vor, die auf der Berechnung des einjährigen Verlustes Δ beruht.

$$\Delta = AC_0 - \frac{AC_1}{1+i} \tag{2.2}$$

Wobei i den risikolosen Zinssatz im ersten Jahr bezeichnet.
Damit gilt für das SCR:

$$SCR = \underset{x}{argmin}\left(\mathbb{P}\left(\Delta > x\right) \leq 0.5\%\right) \tag{2.3}$$

Aus beiden Formeln (2.1) und (2.3) für das Solvency Capital Requirement wird klar, dass bei der Berechnung das Eigenkapital (AC) eine wichtige Rolle spielt. Dieses ist die Differenz zwischen dem Marktwert der Assets und dem Marktwert der Verpflichtungen eines Versicherungsunternehmens. Das AC kann also zu jedem Zeitpunkt t dargestellt werden als:

$$AC_t = MVA_t - MVL_t$$

Die Bewertung eines typischen Asset Portfolios eines Versicherungsunternehmens ist in der Regel relativ simpel, da die Marktwerte entweder direkt verfügbar sind, oder mit etablierten Standardmodellen, unter Verwendung von verfügbaren Parametern, berechnet werden können[1, Bauer et al.].
Daher nehmen wir, in Anlehung an die Vorgehensweise in [8, Hejazi et al.], für den Rahmen dieser Arbeit an, dass unser Versicherungsunternehmen eine passive Anlagestrategie gewählt hat. Das bedeutet Kapital M_0 der Aktionäre wird so angelegt, dass nur risikolose Zinsen generiert werden. Damit ist $MVA_0 = M_0$ und $MVA_1 = M_0(1+i)$, womit wir für Δ aus (2.2) schreiben können:

$$\begin{aligned}\Delta &= (MVA_0 - MVL_0) - \frac{MVA_1 - MVL_1}{1+i} = \\ &= (M_0 - MVL_0) - \frac{M_0(1+i) - MVL_1}{1+i} = \\ &= \frac{MVL_1}{1+i} - MVL_0\end{aligned} \tag{2.4}$$

Da Versicherungsprodukte meist eine komplexe Form haben, die unter anderem oft Garantien und Optionen beinhalten kann diese Bewertung grundsätzlich nicht in einer geschlossenen Form geschehen. Die meisten Unternehmen verwenden daher einen Modellansatz, der auf Monte Carlo Simulationen beruht.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3 Simulationsmethode

3.1 Monte Carlo Methode

Vielen Lösungen quantitativer Probleme in Wirtschaft, Wissenschaft und Technik liegt computergestütztes statistisches Sampling zu Grunde. Diese Aufgabenstellungen können dabei in drei Kategorien eingeteilt werden: Das generieren von zufälligen Prozessen oder Objekten um ihr Verhalten zu analysieren, die Berechnung numerischer Größen durch wiederholtes Sampling und die Lösung von komplizierten Optimierungsverfahren durch die Verwendung von zufallsbasierten Algorithmen [12, Kroess, Rubinstein].
Da, wie weiter unten erläutert, die Berechnung des SCR auf Vorhersage zukünftiger Verpflichtungen und damit auf Erwartungswerten beruht, fällt das Problem dieser Arbeit unter die Kategorie der Berechnung numerischer Größen durch wiederholtes Sampling.

Die Anwendung der Monte Carlo Methode zur Berechnung von Erwartungswerten wird insbesondere durch das *Starke Gesetz der Großen Zahlen von Kolmogorov* und damit verbunden dem *Satz zur fast sicheren Konvergenz* mathematisch gerechtfertigt.

Satz 3.1.1. *(fast sichere Konvergenz)*
Gibt es für eine Folge von Zufallsvariablen $(Z_n)_{n\in\mathbb{N}}$*, ein gegebenes* $c \in \mathbb{R}$ *und für alle* $\epsilon, \delta > 0$ *ein* $n_0 \in \mathbb{N}$*, sodass*

$$\mathbb{P}[|Z_n - c| > \epsilon, \forall n > n_0] < \delta$$

dann heißt die Folge $(Z_n)_{n\in\mathbb{N}}$ *fast sicher konvergent gegen* c *und wir schreiben*

$$Z_n \xrightarrow{f.s.} c.$$

Satz 3.1.2. *(Starkes Gesetz der Großen Zahlen von Kolmogorov)*
Für eine gegebene Folge von unabhängigen, gleichverteilten Zufallsvariablen Y_i *mit Erwartungswert*

$$\mathbb{E}[Y_i] = \mu$$

definiere

$$X_n := \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

Dann gilt: X_n *konvergiert fast sicher gegen* μ *für* $n \to \infty$ $(X_n \xrightarrow{f.s.} \mu)$ *[10, Jaeckel]* .

Zur Berechnung des Erwartungswertes mittels Monte Carlo simulieren wir also eine Folge von Zufallsgrößen. Mit denen wir dann mit Hilfe von Satz 3.1.2 den Erwartungswert näherungsweise ermittelt können. In Satz 3.1.2 sehen wir, dass die Genauigkeit der Annäherung größer wird, wenn die Anzahl der Simulationen steigt. [12, Kroess, Rubinstein]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.2 Nested Stochastic Simulation

Ist ein stochastischer Parameter zu berechnen, der von einem anderen solchen abhängt, so reicht eine einfache Monte Carlo Simulation nicht aus. In diesen Fällen wird dazu eine sogenannte Nested Stochastic Simulation verwendet. Dabei werden für den Sampling-Prozess typischerweise zwei Ebenen benutzt.

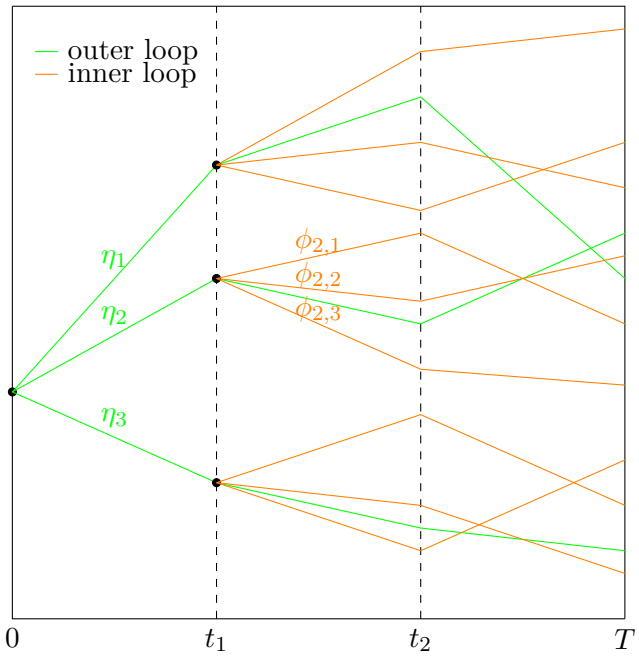


Abbildung 3.1: Nested Simulation mit 3 äußeren Pfaden η_1, η_2, η_3 und je 3 inneren Pfaden $\phi_{i,1}, \phi_{i,2}, \phi_{i,3}$ für $i \in \{1, \ldots, 3\}$.

Wie in Abbildung 3.1 zu sehen ist, werden im ersten Schritt die Werte in einer sogenannten *äußeren Schleife* (Outer Loop) simuliert. Diese Pfade der Simulation werden oft als Szenarien bezeichnet. Im zweiten Schritt werden für jedes Szenario der äußeren Schleife Werte durch Pfade in sogenannten *inneren Schleifen* (Inner Loops) simuliert. Damit hängen die innere Simulationen von den Werten der äußeren Szenarien zum Zeitpunkt t_1 ab.
Zusätzlich können die Zeitschritte als wichtige Parameter in den meisten Anwendungen der Nested Simulation angesehen werden da viele Berechnungen auf Rekursionen basieren. Die Zeitschritte sind dabei deren Perioden. [17, Runhuan Feng et al.].

3.3 Nested Monte Carlo Methode zur SCR Berechnung

Zur Berechnung des SCR mit den Vorgaben aus Kapitel 2 orientieren wir uns an der Vorgehensweise von [1, Bauer et al.] und an den Ausführungen von [6, Grosen and Jorgensen] zur Bewertung von Versicherungsverpflichtungen. Dazu betrachte einen vollständigen filtrierten Wahrscheinlichkeitsraum $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$, wobei T die Maturität des Lebensversicherungsvertrages mit der längsten Laufzeit im Portfolio des Versicherers bezeichnet.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Weiters ist Ω der Raum aller möglichen Zustände des Finanzmarktes, $\mathbb{P}$ ein Wahrscheinlichkeitsmaß und $\mathcal{F}_t$ kann als alle Informationen bis zum Zeitpunkt t verstanden werden. Bei der Bewertung eines Versicherungsvertrages resultieren Unsicherheiten aus der ungewissen Entwicklung bestimmter Einflussfaktoren wie Zinsen, Indizes, etc. Diese Einflussfaktoren werden durch eine d-dimensionale Markovkette $Y = (Y_t)_{t\in[0,T]} = (Y_{t,1}, \dots, Y_{t,d})_{t\in[0,T]}$ modelliert und damit der Wert jedes Produktes dargestellt. Es wird also angenommen, dass Funktionen $(f_1, \dots, f_T)$ existieren, mit denen der Wert der Verpflichtungen zum Ende der Laufzeit $X = (X_1, \dots, X_T)$ dargestellt werden kann als $X_t = f_t(Y_s, s \in [0,t])$ für $t \in \{1, \dots, T\}$. Betrachte zusätzlich das risikolose Bankkonto $(B_t)_{t\in[0,T]}$ mit $B_t = exp\left(\int_0^t r_u du\right)$ wobei $r_t = r(Y_t)$ der risikolose Zinssatz zur Zeit $t \in \{0, \dots, T\}$ bezeichnet. Auf dem Wahrscheinlichkeitsraum existiere weiters ein, zu $\mathbb{P}$ äquivalentes Wahrscheinlichkeitsmaß $\mathcal{Q}$ unter dem Vertragswerte durch ihren erwarteten diskontierten Wert bezüglich $(B_t)_{t\in[0,T]}$ dargestellt werden können.

Mit diesen Annahmen der Marktwert der zukünftigen Verpflichtungen zum Zeitpunkt $t = 0$ berechnet werden durch:

$$MVL_0 := \mathbb{E}_{\mathcal{Q}}\left[\sum_{t=1}^{T} exp\left(-\int_0^t r_u du\right) X_t\right]$$

Dieser Erwartungswert kann in den meisten Fällen nicht direkt berechnet werden, weswegen typischer Weise auf Monte-Carlo-Simulationen zurückgegriffen wird.

Ist M_0 die Anzahl der verwendeten Pfade der Monte-Carlo-Simulation, dann bezeichnet $(Y_t^{(m)})_{t\in[0,T]}$ mit $m \in \{1, \dots, M_0\}$ die unabhängigen Pfade des zugrundeliegenden Prozesses Y. Damit können für jeden Pfad die Vertragswerte $(X_t^{(m)})$ $(t \in \{1, \dots T\}, m \in \{1, \dots, M_0\})$ bestimmt werden. Durch Diskontierung mit den ensprechenden Faktoren und Mittelung über alle Pfade kann MVL_0 in Abhängigkeit der Anzahl der Monte-Carlo-Pfade approximiert werden:

$$\widetilde{MVL_0} = \frac{1}{M_0}\sum_{m=1}^{M_0}\sum_{t=1}^{T} \exp\left(-\int_0^t r_u^{(m)} du\right) X_t^{(m)} \tag{3.1}$$

wobei $r_t^{(m)}$ den risikolosen Zinssatz im m-ten Pfad zum Zeitpunkt t bezeichnet.

In (2.4) sehen wir, dass zur SCR Berechnung nicht nur der Marktwert der Verpflichtungen zum Zeitpunkt $t = 0$ zu bewerten ist, sondern auch zum Zeitpunkt $t = 1$. Hier kommt die Nested Stochastic Simulation zur Anwendung, da nun die Simulation von den Ergebnissen aus der Berechnung von MVL_0 abhängt. Der Marktwert zum Zeitpunk $t = 1$ kann dann wieder berechnet werden durch:

$$MVL_1 := \mathbb{E}_{\mathcal{Q}}\left[\sum_{t=2}^{T} exp\left(-\int_1^t r_u du\right) X_t \middle| \mathcal{F}_1\right] \tag{3.2}$$

Hier ist die Verteilung der MVL_1 gesucht, was wiederum mittels Monte Carlo Simulationen erreicht werden kann, da die einzelnen Simulationen unabhängig und gleichverteilt

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

sind.
Ist nun $n \in \mathbb{N}$ die Anzahl der Pfade aus der Simulation für MVL_0, dann sind $(Y_s^{(i)})_{s\in[0,1]}$ mit $i \in \{1,\dots,N\}$ die Entwicklungen des Marktes im ersten Jahr. Damit kann MVL_1 aus (3.2) für den i-ten Pfad berechnet werden durch

$$MVL_1^{(i)} = \mathbb{E}_{\mathcal{Q}}\left[\sum_{t=2}^{T} exp\left(-\int_1^t r_u du\right) X_t \middle| (Y_s^{(i)})_{s\in[0,1]})\right]$$

Auch hier ist es im Normalfall nicht möglich, den Erwartungswert direkt zu berechnen, womit auf Monte-Carlo Simulationen zurückgegriffen werden muss. Ist nun $M_1^{(i)}$ die Anzahl der Pfade zur Simulation der Erwartungwerte so sind wiederum $(Y_t^{(i,m)})$ mit $i \in \{1,\dots,N\}$ und $m \in \{2,\dots M_1\}$ die zugehörigen Prozesse ausgehend von $(Y_s^{(i)})_{s\in[0,1]}$ aus dem ersten Jahr. Daraus wird, wie vorhin, $(X_t^{(i,m)})$ bestimmt und mit den entsprechenden Faktoren diskontiert. Auch hier erhalten wir durch das Mitteln über alle M_1 Pfade wieder die entsprechenden Approximationen für $MVL_1^{(i)}$:

$$\widetilde{MVL_1}^{(i)} = X_1^{(i)} + \frac{1}{M_1}\sum_{m=1}^{M_1}\sum_{t=2}^{T} exp\left(-\int_1^t r_u^{(i,m)} du\right) X_t^{(i,m)}, \text{ für } i \in \{1,\dots,N\} \qquad (3.3)$$

Das SCR kann jetzt approximiert werden, indem (3.1) und (3.3) in die Formel (2.4) eingesetzt werden und davon das 99,5%-Quantil bestimmt wird.

Als Vereinfachung zum Zweck dieser Arbeit betrachten wir nur eine einzige Verpflichtung.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4 Machine Learning

Unter Machine Learning können die verschiedenen Computer-Algorithmen verstanden werden, welche bekannte (oft historische) Daten verwenden, um damit Vorhersagen für einen neuen Datensatz zu treffen. Dies kann so verstanden werden, dass in bekannten Daten, auf unterschiedliche Art und Weise, ein Muster gesucht wird, um damit unter einer gewissen Unsicherheit zukünftige Ergebnisse berechnen zu können[14, Nilsson, Sandberg].
Der Aufbau dieser Machine-Learning Algorithmen hängt von der zugrundeliegenden Fragestellung ab, also welche Form die Ergebnisse der Vorhersage haben sollen.
Machine-Learning Algorithmen werden hauptsächlich im Rahmen von Problemen wie Klassifikation, Regression, Transkription, maschinelle Übersetzung, Strukturierung, Auffinden von Anomalien, Synthese und Sampling, Rauschunterdrückung und Schätzung der Dichte bzw. Wahrscheinlichkeitsfunktion verwendet. Dabei sollen durch diese Aufzählung keine weiteren Anwendungsbereiche ausgeschlossen werden[5, Goodfellow et al.].
In dieser Arbeit wenden wir einen Algorithmus aus dem Umfeld des Machine Learning auf ein Regressionsproblem an. Das Programm soll also numerische Werte aus einem gegebenen Input vorhersagen. Dazu soll eine Funktion $F : \mathbb{R}^n \to \mathbb{R}$ gefunden werden, die einem Vektor $\mathbf{x}$ eine Zahl $y := F(\mathbf{x})$ zuweist.
Betrachtet man die Art des Lernens der Modelle, so können Machine Learning Algorithmen in drei verschiedene Kategorien eingeteilt werden. *Supervised Learning* verfolgt das Ziel eine vorgegebene Fehlerfunktion zu minimieren, die zwischen dem errechneten Output und den bekannten Ergebnissen ermittelt wird. Hier ist wichtig, dass Vergleichsdaten (sogenannte Labels) für den Output in der Trainingsphase bereitstehen. Bei *Unsupervised Learning* sind nur Input Daten vorhanden. Also kann Supervised Learning mit der bedingten Dichteschätzung und Unsupervised Learning mit der unbedingten Dichteschätzung verglichen werden. Die dritte Kategorie bildet das *Reinforcement Learning*, bei dem das Trainingsystem ununterbrochen mit seinem Umfeld zusammenhängt[7, Hastie et al.].

4.1 Künstliche Neuronale Netze (KNN)

Neuronale Netze finden Anwendung in allen Kategorien von Machine Learning Algorithmen, sowie für jedes Problem zu dessen Lösung diese eingesetzt werden.
Wie bereits erwähnt, wird Machine Learning in dieser Arbeit auf ein Regressionsproblem angewandt, weswegen wir hier künstliche Neuronale Netze in diesem Kontext betrachten werden.

Das Modell besteht aus einem Input X, welcher durch eine sogenannte Aktivierungsfunktion $\varrho(.)$, angewandt auf eine affine Transformation, in neue Variable $S_n, n \in \mathbb{N}$ transformiert wird, welche auch "Knoten" genannt werden. Dabei kann $n \in \mathbb{N}$ beliebig gewählt werden, wobei n standardmäßig eine Potenz von 2 ist. Eine genauere Beschreibung dieses

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Vorgangs liefert Definition 4.1.1.
Diese Transformation kann mehrmals geschehen, wodurch mehrere "Layers" entstehen. Der Teil zwischen Input- und Output-Layer wird als "hidden Layers" bezeichnet, da der genaue Wert der Transformationen nicht sichtbar ist. Ein KNN mit mehreren hidden Layers wird *Deep Neural Network* genannt [14, Nilsson, Sandberg].
Mathematisch kann dieser Vorgang durch die folgende Definition beschrieben werden:

Definition 4.1.1. *Seien $N, L_0, \dots, L_N \in \mathbb{N}$ und $\varrho : \mathbb{R} \to \mathbb{R}$ eine Funktion. Weiters sei für alle $n \in \{1, \dots N\}$, $S_n : \mathbb{R}^{L_{n-1}} \to \mathbb{R}^{L_n}$ eine affine Funktion. Dann heißt die Funktion $F : \mathbb{R}^{L_0} \to \mathbb{R}^{L_N}$, definiert durch*

$$F(x) = S_N \circ F_{N-1} \circ \cdots \circ F_1 \text{ mit } F_n = \varrho \circ S_n \text{ für } n \in \{1, \dots, N-1\}$$

(feed forward) Neuronales Netz.
Die Aktivierungsfunktion ϱ wird dabei komponentenweise angewandt.
N bezeichnet die Anzahl der Layers, $L_1, \dots, L_{N-1}$ die Dimensionen der hidden Layers und L_0, L_N die Dimensionen des Input- bzw. Output-Layers.
Für alle $n \in \{1, \dots, N\}$ ist die affine Funktion S_n gegeben als $S_n = W^n x + b^n$ für alle $W^n \in \mathbb{R}^{L_n \times L_{n-1}}$ und $b^n \in \mathbb{R}^{L_n}$.
Für $i \in \{1, \dots, L_n\}$, $j \in \{1, \dots, L_{n-1}\}$ wird W^n_{ij} als das Gewicht jener Kante interpretiert, die den i-ten Knoten des $(n-1)$-ten Layers mit dem j-ten Knoten des n-ten Layers verbindet.
Die Menge aller Neuronalen Netze mit Aktivierungsfunktion ϱ die $\mathbb{R}^{d_0}$ auf $\mathbb{R}^{d_1}$ abbildet, wird als $\mathcal{NN}^{\varrho}_{\infty,d_0,d_1}$ bezeichnet [3, Buehler et al.].

Einige Beispiele für die Aktivierungsfunktion sind[1]:

Sigmoid Aktivierungsfunktion	$\varrho(x) = \frac{1}{1+e^{-x}}$	$\in (0, 1)$
Rectified Linear Unit (ReLU)	$\varrho(x) = max(0, x) = (x)_+$	$\in [0, \infty)$
Tangens Hyperbolicus Aktivierungsfunktion	$\varrho(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\in (-1, 1)$
Softsign Aktivierungsfunktion	$\varrho(x) = \frac{x}{\lvert x \rvert + 1}$	$\in (-1, 1)$
Exponential Linear Unit	$\varrho(x) = \begin{cases} x & , x > 0 \\ e^x - 1 & , x < 0 \end{cases}$	$\in [-1, \infty)$
lineare Aktivierungfunktion	$\varrho(x) = id_x$	

Die lineare Aktivierungsfunktion wird für gewöhnlich im Output-Layer verwendet.

Da in dem Modell aus Definition 4.1.1 die Informationen nur in eine Richtung und von einem Layer zum unmittelbar nächsten laufen, wird diese Art von Neuronalen Netzen feed

[1] siehe https://keras.io/activations/

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

forward genannt.

Die Aufgabenstellung bei der Verwendung von Neuronalen Netzen besteht darin, für alle $n \in \{1, \dots, N\}$ die Werte der Gewichte W_{ij}^n für $i \in \{1, \dots L_n\}$, $j \in \{1, \dots, L_{n-1}\}$ und des Bias-Vektors $b^n \in \mathbb{R}^{L_n}$ so zu wählen, dass eine, im Vorhinein gewählte, Fehlerfunktion zwischen den Outputs und den Labels minimal wird. Diesen Vorgang nennt man Training des künstlichen Neuronalen Netzes. Bei Regressionsproblemen wird dafür standardmäßig der quadratische Fehler verwendet [14, Nilsson,Sandberg].
Ein Multi-Layer Feed Forward Neuronales Netz können wir uns wie folgt vorstellen:

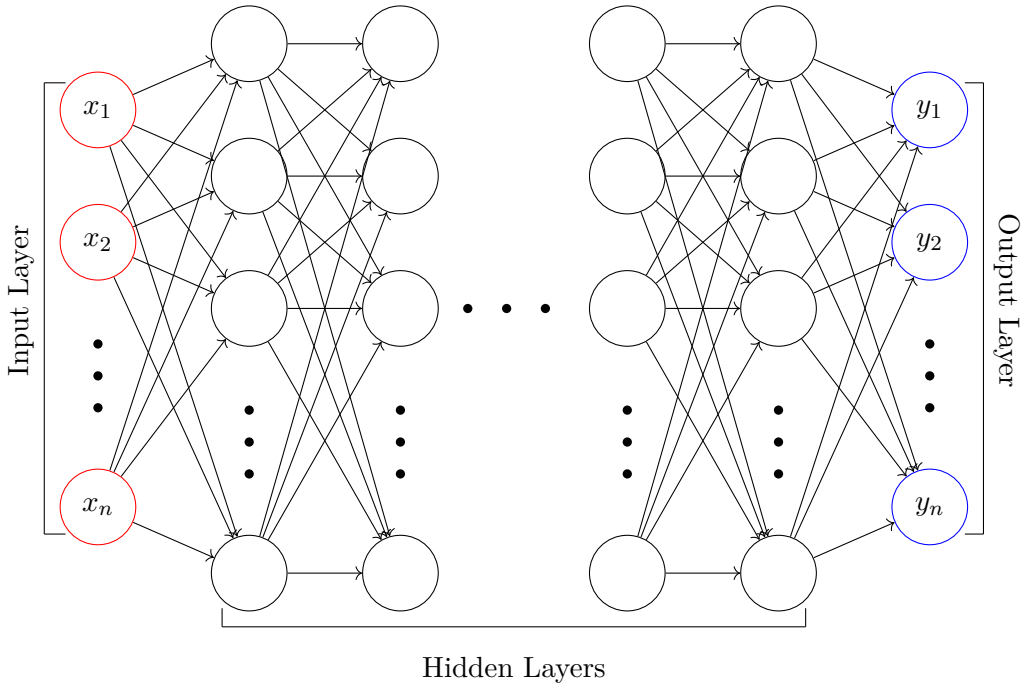
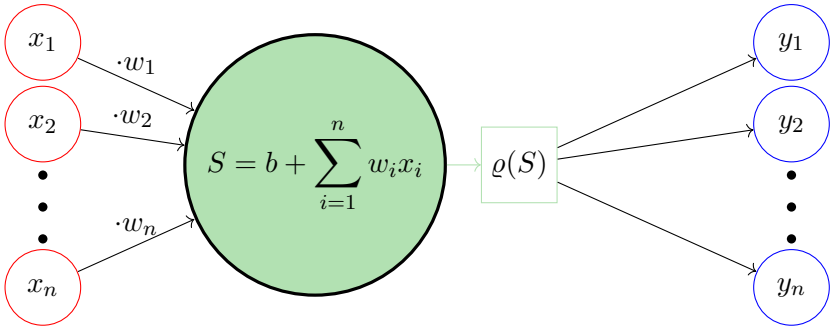


Abbildung 4.1: Multi-Layer Neuronales Netz einem n-dimensionalen Input x und n Outputs y



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Abbildung 4.2: Darstellung der Vorgänge in einem Knoten; n-dim. Input und n Outputs; Aktivierungsfunktion ϱ, bias b und Gewichte w_i

Die Anwendung eines Neuronalen Netzes wird durch das folgendes Resultat gerechtfertigt, da dieses im Grunde besagt, dass jede Funktion beliebig genau durch ein künstliches Neuronales Netz approximiert werden kann.

Satz 4.1.1. *(Universal Approximation)*
Ist die Aktivierungfunktion ϱ beschränkt und nicht konstant, dann gelten die folgenden Aussagen:

- *Für jedes endliche Maß μ auf $(\mathbb{R}^{d_0}, \mathcal{B}(\mathbb{R}^{d_0}))$ und $1 \leq p < \infty$ ist die Menge $\mathcal{NN}^{\varrho}_{\infty,d_0,1}$ dicht in $L^p(\mathbb{R}^{d_0}, \mu)$.*
- *Ist zusätzlich $\varrho \in C(\mathbb{R})$, dann ist $\mathcal{NN}^{\varrho}_{\infty,d_0,1}$ dicht in $C(\mathbb{R}^{d_0})$ bezüglich der Topologie der gleichmäßigen Konvergenz auf kompakten Mengen.*

Da jede Komponente eines d_1-dimensionalen Neuronalen Netzes ein eindimensionales Neuronales Netz ist, können diese Ergebnisse leicht auf ein $d-1$-dimensionales Neuronales Netz ausgeweitet werden [9, Hornik].

4.2 Gradienten Abstiegsverfahren

Wie bereits erwähnt, soll, während der Trainingsphase, der quadratische Fehler minimiert werden. Für dieses Optimierungsproblem wird üblicherweise ein sogenanntes Gradienten Abstiegsverfahren verwendet.

Für die Übersichtlichkeit betrachten wir hier ein Neuronales Netzwerk mit nur einem Hidden Layer, da das Gradienten Abstiegsverfahren besonders im letzten Layer zu Anwendung kommt und damit die Theorie auf ein mehrdimensionales Netz leicht erweitert werden kann. Unter Verwendung der Definition 4.1.1 entsteht folgendes Gleichungssystem:
Sei $Y := F_1$, $Z := S_2$ und bezeichne ω die Gewichte des Hidden-Layers und κ die des Output-Layers. Zusätzlich definieren wir mit b den Bias des Hidden-Layers und mit β den des Output-Layers. Damit ist ein feed forward Neuronales Netz, mit einem Hidden- Layer, für Input-Vektoren X gegeben durch:

$$\begin{aligned} Y_m &= \varrho(b_m + \omega_m^T X) && m \in \{1, \dots, L_1\} \\ Z_k &= \beta_k + \kappa_k^T Y && k \in \{1, \dots, L_2\} \\ F_k(X) &= g_k(Z) && k \in \{1, \dots, L_2\} \end{aligned} \tag{4.1}$$

Dabei ist ω_m bzw. κ_k der Vektor der Gewichte des m-ten bzw. k-ten Knoten im Hidden bzw. Output Layer.
Da $F(X)$ eine Funktion der Knoten aus dem Output-Layer ist, gilt die letzte Gleichung aus (4.1), wobei g die Aktivierungsfunktion des letzten Layers bezeichnet. Weiters ist hier L_2 die Anzahl der möglichen Ergebnisse. Im Rahmen eines Regressionsproblems wird diese Funktion im Normalfall als Identität und $L_2 = 1$ angenommen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Wie wir bereits wissen, ist die Aufgabe eines Neuronalen Netzes die Gewichte und die Bias-Werte so zu wählen, dass der quadratische Fehler möglichst klein wird.
Bei Klassifikationsproblemen wird hier, statt dem quadratischen Fehler, oft die Kreuzentropie verwendet.

Für eine Trainingsmenge $(\boldsymbol{x}_i, y_i)_{i=1}^M$ wird der quadratische Fehler in Abhängigkeit von der Parametermenge $\boldsymbol{\theta}$, wie folgt definiert:

$$R(\boldsymbol{\theta}) = \sum_{k=1}^{L_2} \sum_{i=1}^{M} (y_{ik} - F_k(x_i))^2$$

Für die einzelnen Datenpunkte aus der Trainingsmenge definiere weiters mit dem Gleichungssystem aus (4.1) $y_{mi} := \varrho(b_m + w_m^T x_i)$ und damit $y_i := (z_{1i}, \ldots, z_{L_1 i})$. Wodurch der quadratischen Fehler geschrieben werden kann als:

$$R(\boldsymbol{\theta}) = \sum_{i=1}^{M} R_i = \sum_{i=1}^{M} \sum_{k=1}^{L_2} (y_{ik} - F_k(x_i))^2$$

mit den Differentialen

$$\frac{\partial R_i}{\partial \kappa_{km}} = -2(y_{ik} - F_k(x_i)) g_k'(\kappa_k^T z_i) z_{mi} \tag{4.2}$$

$$\frac{\partial R_i}{\partial \omega_{mu}} = -\sum_{i=1}^{L_2} 2(y_i k - F_k(x_i)) g_k'(\kappa_k^T z_i) \kappa_{km} \varrho'(\omega_m^T x_i) x_{iu} \tag{4.3}$$

Damit sind die Iterationen für das Gradienten-Abstiegsverfahren gegeben durch:

$$\kappa_{km}^{(r+1)} = \kappa_{km}^{(r)} - \eta_r \sum_{i=1}^{M} \frac{\partial R_i}{\partial \kappa_{km}^{(r)}}$$

$$\omega_{mu}^{(r+1)} = \omega_{mu}^{(r)} - \eta_r \sum_{i=1}^{M} \frac{\partial R_i}{\partial \omega_{mu}^{(r)}}$$

wobei mit η_r die r-te Lernrate bezeichnet wird.
Die "Fehler" δ_{ki} und s_{mi} in den Knoten des Output- bzw. Hidden Layers, können dabei aus den Ableitungen (4.2) und (4.3) abgelesen werden:

$$\frac{\partial R_i}{\partial \kappa_{km}} = \delta_{ki} z_{mi} \tag{4.4}$$

$$\frac{\partial R_i}{\partial \omega_{mu}} = s_{mi} x_{iu} \tag{4.5}$$

Damit kann s_{mi} auch geschrieben werden als:

$$s_{mi} = \sum_{k=1}^{L_2} \delta_{ki} \kappa_{km} \varrho'(\omega_m^T x_i) \tag{4.6}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

was als "Back-Propagation" bekannt ist.

Unter Verwendung dieser Ergebnisse geht der Algorithmus des Gradienten Absiegsverfahrens folgendermaßen vor:
Im ersten Schritt wird F_k durch (4.1) mit zufällig gewählten Gewichten sowie Bias-Werten berechnet. Wodurch im nächsten Schritt der Fehler δ_{ki} aus (4.4) gemeinsam mit (4.2) ermittelt werden kann und für die Berechnung des Fehlers s_{mi} kann "Back-Propagation", also die Formel (4.6) benützt werden. Mit s_{mi} und δ_{ki} sowie den Formeln (4.4) und (4.5) können am Ende jedes Durchlaufs (Epoche) die Gewichte sowie die Bias-Werte angepasst werden [7, Hastie et al.].

4.3 Momentum Methode

Im Gradienten Abstiegsverfahren werden keine zweiten Ableitungen und damit auch nicht die Krümmung der Funktion betrachtet. Dies kann besonders im Mehrdimensionalen zu einem Problem führen. In diesem Fall existieren nämlich verschiedene zweiten Ableitungen in jede Richtung an einem bestimmten Punkt. Unterscheiden sich diese maßgeblich, bedeutet dies, dass die Ableitung in eine Richtung deutlich schneller fällt als in eine andere Richtung. Dies wird im Gradienten Abstiegsverfahren nicht mit einbezogen, wodurch das Verfahren nicht beachtet, dass hier vorzugsweise jene Richtung verfolgt werden sollte, in welcher der Gradient länger negativ bleibt, die Funktion also länger fällt. Zusätzlich wird damit die Wahl der Schrittweite zu einer großen Herausforderung, da das Risiko besteht, dass der Algorithmus über das Minimum hinwegläuft, wenn diese zu groß gewählt wird. Andererseits gilt, je kleiner die Schrittweite, desto langsamer das Verfahren.

Eine Methode, welche die Krümmung der Funktion einbezieht, und besonders hinsichtlich stark gekrümmter Funktionen, kleiner konsistenter Gradienten oder stark schwankender Gradienten entwickelt wurde, ist die sogenannte Momentum Methode. Diese verwendet ein exponentiel fallendes, kumuliertes gleitendes Mittel von Gradienten aus der Vergangenheit und bewegt sich in dessen Richtung.
Bei dieser Methode wird als eine Variable v eingeführt, welche die Richtung und Geschwindigkeit angibt, in die die Parameter angepasst werden. Diese Variable v ist wie folgt definiert:

$$v = \alpha v - \epsilon \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)}) \right)$$

Dabei ist θ der Parametervektor und $L(f(x^{(i)}, \theta), y^{(i)})$ die zu optimierende Funktion, ϵ die Lernrate, und $\alpha \in [0, 1)$ jener Hyperparameter, der angibt, wie schnell der Einfluss der vorherigen Gradienten exponentiell abnimmt.
Die neuen Parameter im jeweiligen Schritt, sind dann die Parameter des vorhergegangenen Optimierungsschrittes plus v.
Je größer der Hyperparameter α im Bezug zur Lernrate ϵ ist, desto mehr wird die aktuelle Anpassung durch die vorherigen Gradienten beeinflusst.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Beim Basisalgorithmus für stochastic Gradient Descent ist die Schrittweite ganz einfach gegeben durch die Norm des Gradienten multipliziert mit der Lernrate. Wird das Momentum in die Optimierungsmethode mit einbezogen so hängt die Schrittweite von der gesamten Folge an Gradienten ab. Insbesondere von den Werten selbst, aber auch davon wie sehr die Gradienten in jedem Schritt von einander abhengen [5, Goodfellow et al.].

4.4 Anpassung der Lernrate

Wie bereits erwähnt ist eine optimale Wahl der Lernrate bei der Verwendung des Basisalgorithmus des Gradienten Absiegsverfahren elementar. Dies ist eine der schwierigsten Aufgaben, da es deutliche Auwirkungen auf die Performance des Modells haben kann. Der Grund dafür ist, dass der Fehler auf manche Veränderungen in den Parametern sehr stark reagieren kann, auf andere wiederum nur minimal. Die in 4.3 diskutierte Momentum Methode kann dieses Problem verringern, jedoch indem ein weiterer Hyperparameter eingeführt wird, der wiederum optimal zu wählen ist.
Zwei Methoden zur Anpassung der Lernrate, die häufig verwendet werden sind AdaGrad und RMSProp. Beide skalieren die Lernrate unter Verwendung der historischen Parameter. Während AdaGrad die Lernrate alle Modellparameter invers proportional zur Quadratwurzel der Summe alle historischen Parameter skaliert, werden ibei RMSProp die kumulierten Gradienten durch einen exponentiell gewichteten gleitenden Mittelwert ersetzt.
In beiden Fällen sinkt die Lernrate für Parameter, für die die partielle Ableitung der Fehlerfunktion größer ist, schneller als für jene Parameter mit kleiner partieller Ableitung der Fehlerfunktion. Dies resultiert in einem schnelleren Fortschritt in der flacheren Richtung.
In beiden Fällen ist die Anpassung der Parameter gegeben durch

$$\Delta\theta = \frac{\epsilon}{\delta + \sqrt{r}} \cdot \left(\frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}, \theta), y^{(i)}) \right)$$

Dabei bezeichnet θ den Parametervektor, ϵ die initialisierte Lernrate, r die Anpassung der Lernrate und L die zu optimierende Fehlerfunktion. Durch die Konstante δ, welche sehr klein gewählt wird (beispielsweise $\delta = 10^{-7}$), wird die Division durch 0 vermieden.

AdaGrad und RMSProp unterscheiden sich nur in der Berechnung der Anpassung der Lernrate r:

$$r_t^{AdaGrad} = r_{t-1}^{AdaGrad} + \left(\frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}, \theta), y^{(i)}) \right)^2$$

$$r_t^{RMSProp} = \rho\, r_{t-1}^{RMSProp} + (1-\rho) \left(\frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}, \theta), y^{(i)}) \right)^2$$

Dabei bezeichnet ρ die sogenannte Abfallrate [5, Goodfellow et al.].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.5 Adam Optimizer

Der Basisalgorithmus des Gradienten Abstiegsverfahren verwendet Gradienten direkt. Was zu Problemen führen kann, da diese für beliebige Parameter nahe bei Null sein können. Dies kann dazu führen, dass der Algorithmus bei einem lokalen Minimum stecken bleibt, obwohl ein globales Minimum gesucht ist. Um dieses Problem zu umgehen empfehlen [16, Patterson et al.] die sogenannte Adam Methode. Diese wird derzeit ohnehin standardmäßig verwendet, da sie einige Vorteile mit sich bringt. So beeinflusst das Ausmaß der Anpassung der Parameter nicht die Dimension der Gradienten, die Anzahl der Schritte ist ungefähr durch den Parameter für die Schrittzahl beschränkt, Stationarität wird für das Verfahren nicht vorausgesetzt, der Optimizer funktioniert auch für schwach besetzte Gradienten und die automatische Reduktion der Schrittweite.

Der Grundgedanke des Adaptive Moment Optimizer (Adam) besteht in der Berechnung des Schätzers des ersten und zweiten Moments unter Verwendung des Gradienten der Funktion bezüglich der Parameter in jedem Schritt. Damit entsteht folgendes Gleichungssystem als Grundlage für den Algorithmus:

$$\begin{aligned} g_t &= \nabla_\theta f_t(\theta_{t-1}) \\ m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned}$$

Dabei bezeichnet g_t den Gradient der Funktion f im t-ten Schritt bezüglich der Parameter θ, m_t den Schätzer für das erste Moment und v_t den für das zweite. Mit $\beta_1, \beta_2 \in [0, 1)$ kann die exponentielle Abfallrate der Momentenschätzer geregelt werden.
Im Algorithmus für den Adam Optimizer werden die beiden Schätzer als 0 Vektoren initialisiert, was jedoch dazu führt dass die Schätzer in Folge in Richtung Null verzerrt sind. Besonders kommt dies zu Beginn des Verfahrens und wenn die Abfallrate gering ist zu tragen. Daher werden in einem ersten Schritt β_1 und β_2 nahe bei 1 gewählt. Weiters kann dem entgegen gewirkt werden indem m_t und v_t wie folgt angepasst werden:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned}$$

Damit erfolgt die Anpassung der Parameter über folgende Iteration:

$$\theta_t = \theta_{t-1} - \frac{\gamma \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

wobei γ die Schrittweite bezeichnet [11, Kingma et al.].

Für die Anwendung des Adam Optimizers im Rahmen Neuronaler Netze empfehlen [11, Adam und Ba] die Parameter des Algorithmus als $\gamma = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ zu wählen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Adam kann als eine Art der Kombination von RMSProp und der Momentum Methode gesehen werden. Dabei wird das Momentum direkt als Schätzer des ersten Moments des Gradienten mit exponentiellen Gewichten in den Algorithmus eingebaut. RMSProp wird in Folge auf dieser Schätzer angewandt [5, Goodfellow et al.].

4.6 Implementierung künstlicher Neuronaler Netze in Python [2]

Die benutzerfreundlicheste Methode des Programmierens künstlicher Neuronaler Netze in Python ist die Verwendung des Keras Paketes. Dieses basiert auf dem, von Google entwickelten, Paket TensorFlow. Als direkter Nachfolger von Googles erstem Deep Learning Tools DistBelief, ist dieses Paket für jeden Nutzer als Open Source frei zugänglich. Dem Nutzer ist es damit leicht möglich Modelle selbst zu entwickeln und zu editieren.
Die Besonderheit von TensorFlow ist dabei, dass jedes neuronale Netz als gerichteter, zyklenfreier Graph dargestellt werden kann. Dabei stellen die Kanten die In- und Outputs der einzelnen Rechenschritte dar, die Knoten sind für die Verarbeitung dieser Inputs zu Outputs zuständig. Dies bedeutet, dass in den Knoten eines solchen Graphen die mathematischen Operationen erfolgen, die Kanten stellen mehrdimensionale Datenarrays, sogenannte Tensoren, dar, wodurch zwischen den Knoten kommuniziert wird.
Wie erwähnt ist Keras die benutzerfreundliche Methode zur Programmierung Neuronaler Netze in Python, da dieses Paket bereits mit dem Ziel programmiert wurde, dem Benutzer einfache und schnelle Experimente zu ermöglichen. Die Entwickler folgten dabei folgendem Leitbild:

"Being able to go from idea to result with the least possible delay is key to doing good research".

Bei der Entwicklung des Paketes waren folgende Prinzipien maßgeblich, welche bei der Nutzung auch spürbar sind.

- Benutzerfreundlichkeit: Da das Paket für Menschen und nicht für Maschinen entworfen wurde, steht das Erlebnis des Benutzers im Mittelpunkt. Dafür bietet Keras eine konsistente und einfache Programmierschnittstelle welche für die Anwendung auf Standardprobleme nur eine minimale Anzahl von Nutzeraktionen notwendig macht.

- Modularität: Hier wird ein Modell verstanden als Vereinigung mehrerer, für sich allein stehender, Module. Diese können einfach angepasst und mit wenig Aufwand zusammengefügt werden.

- Einfache Erweiterungsmöglichkeit: Neue Module sind, in Form von Klassen oder Funktionen, einfach hinzuzufügen. Dabei besteht bereits eine Vielzahl von vorimplementierten Beispielen.

- Arbeit mit Python: Es werden keine separaten Modelle benötigt und die Konfiguration von Datein erfolgt in einer selbsterklärenden Form. Weiters sind alle Modelle in Python implementiert, wodurch der Code kompakt und einfach zu erweitern ist.

[2] Informationen stammen von der offiziellen Keras Website: https://keras.io

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.6.1 Ein kleines Beispiel

```
]
model=Sequential()
model.add(Dense(32,input_dim=input_data.shape[1],activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(1))

model.compile(loss='mse', optimizer='adam',
                    metrics=['mean_absolute_error'])

history=model.fit(input_data, labels, epochs=500, callbacks=es,
                            verbose=0, validation_split=0.2)

labels_new=model.predict(scaler.transform(input_data_predict))
```

Zur Erstellung eines Modells wird `Sequential()` verwendet, welches eine lineare Anordnung von Layern darstellt.

Das Hinzufügen und damit Stapeln von Layern erfolgt einfach durch den Python-Befehl `.add()`. Dieser Keras Befehl bietet dabei diverse Optionen zur Anpassung der Architektur jedes einzelnen Layers. Dabei aktiviert `Dense()` die verschiedenen Optionen. Da im Input Layer die Daten meistens mehrdimensional sind, muss im ersten Layer des Keras Modells die Dimension der Input Daten durch `input_dim=input_data.shape[1]` angegeben werden. Dabei bezeichnet `input_data` die Matrix der Input-Daten für das Neuronale Netz. Danach wird in jedem Layer des Modells die Anzahl der Knoten, sowie die Aktivierungsfunktion angegeben. Standardmäßig wird hier die lineare Aktivierungsfunktion verwendet und dahier muss in dem Fall diese nicht explizit angegeben werden.

Ist das Modell aufgebaut, so kann der Lernprozess durch `.compile()` konfiguriert werden. Hier ist zu definieren welche Fehlerfunktion `loss` betrachtet, welcher Optimierer `optimizer` angewandt und welche Metrik `metrics` berechnet werden soll.

Das Training läuft nun iterativ ab, wobei die Anzahl der Iterationen im Vorhinein durch die Initialisierung der `epochs` angegeben wird. Dieses Training erfolgt mit `.fit()`, dem die Trainingsmatrix, der Labels-Vektor, die Anzahl der Iterationen und die Aufteilung der Trainingsmenge in Training und Validierung übergeben werden muss. Hier existieren noch weitere Möglichkeiten, wie die Angabe einer verfrühten Beendigung des Trainings bei einer geringen Veränderung des Fehlers. Nach einigen Test dieser Option wurde jedoch festgestellt, dass dies dazu führen kann, dass eine plötzliche starke Veränderung damit nichtmehr mit einbezogen wird.

Zur Anwendung des trainierten Modelles auf einen neuen Datensatz wird der Python/Keras Befehl `.predict()` verwendet. Wobei hier nur die neuen Input-Daten zu übergeben sind.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5 Select Produkte

In diesem Kapitel orientieren wir uns maßgeblich an [15, Pasztor].

Sogenannte „Select-Produkte“, als Konzept für Lebensversicherungen beruhen auf der Partizipation an einem Aktienindex. Dadurch werden die Möglichkeiten des Aktienmarktes mit den Vorteilen einer Rentenversicherungen verbunden. Der Versicherungsnehmer hat jährlich die Möglichkeit, sich zwischen einer Partizipation am Index und einer garantierten Verzinsung zu entscheiden. Dies beeinflusst in Folge die Höhe der Jahresrendite.

Entscheidet sich der Versicherungsnehmer für die Indexpartizipation, so wird der eingezahlte Betrag bis zu Beginn des ersten Indexjahres mit dem, zu Jahresanfang veröffentlichten, Zins für unterjährig angesammelte Beträge verzinst. Der Gewinn aus der Verzinsung wird zum Kauf einer einjährigen Option auf die Kursentwicklung eines zugrundeliegenden Aktienindex verwendet.
Bei dieser Strategie wird die Jahresrendite folgendermaßen ermittelt: Die monatlichen Kursgewinne werden unter Berücksichtigung einer Obergrenze (Cap) aufsummiert und, falls das Endergebnis positiv ist, nach dem „Lock-in“ Prinzip gesichert. Das Vorsorgevermögen erhöht sich damit um die ermittelte Rendite aus der Anlage. Ist die Summe der monatlichen Gewinne jedoch negativ, so wird sie 0 gesetzt und das Vorsorgevermögen für das Folgejahr bleibt gleich. Diese Untergrenze (auch Floor) bei 0 schützt dabei den Vertrag vor Kapitalverlust. Die Höhe des Caps wird jährlich im vorhinein festgelegt und hängt unter anderem von aktuellen Kapitalmarktdaten, wie zum Beispiel der Volatilität oder dem aktuellen Zinsniveau ab. Zusätzlich hängt von der Höhe der Verzinsung des Versicherers auch die Höhe des Betrags zum Kauf der Option ab, was wiederum die Höhe des Caps beeinflusst. Je höher der Betrag, desto höher der Cap. Umgekehrt gilt aber auch je höher der Cap, desto höher der Optionspreis. Dass der Cap nicht zu niedrig ausfällt wird durch den Wettbewerb zwischen den Emittenten am Kapitalmarkt geregelt.

Während dem laufenden Indexjahres, bis zum folgenden Indexstichtag, werden die eingezahlten Beiträge abzüglich der anfallenden Kosten angesammelt und mit dem Zinssatz für unterjährige Beiträge verzinst. Damit wird die Bezugsgröße für die Indexpartizipation des Folgejahres erhöht.

Wie anfangs angemerkt besteht jährlich vor dem folgenden Indexstichtag die Möglichkeit aus der Indexpartizipation auszusteigen, falls die Entwicklung des Aktienmarktes weniger positiv erscheint. Dabei kann der Versicherungsnehmer entscheiden, vollständig in eine sichere Verzinsung zu investieren oder das Vorsorgevermögen flexibel auf beide Varianten aufzuteilen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.1 Mathematische Beschreibung

Wird die Indexpartizipation gewählt so kann die Rendite des Vertrages wie folgt berechnet werden:

Sei $T \in \mathbb{N}$ die Dauer der Ansparphase in Jahren und definiere $\mathbb{T} := \{1, 2, \dots T\}$. Dann ist die monatliche Indexentwicklung gegeben durch:

$$\forall t \in \mathbb{T}: \; r_{t,\tau+1} = \frac{S_{t,\tau+1} - S_{t,\tau}}{S_{t,\tau}}$$

wobei $S_{t,\tau}$ den Wert des Aktienindex im Monat $\tau \in \{1, 2, \dots, 12\}$ des Jahres t bezeichnet.

Die Jahresrendite eines Select-Produktes ist, wie bereits erwähnt, die Summe der monatlichen Kursgewinne des Index unter Beachtung des Caps c_t. Damit kann die Jahresrendite des Produktes r_t^{Sel} geschrieben werden als:

$$\forall t \in \mathbb{T}: \; r_t^{Sel} = max\left(\sum_{\tau=1}^{12} min(r_{t,\tau+1}; c_t); 0\right) \tag{5.1}$$

Wie bereits angemerkt, hat der Versicherungsnehmer, bei der Anlage in ein Select-Produkt, neben der 100%-igen Veranlagung in den Aktienindex oder der 100%-igen sicheren Verzinsung, die Möglichkeit die beiden Varianten zu mischen. Dabei kann diese Aufteilung für ein Jahr im Voraus gewählt werden. Um dies mathematisch allgemein auszudrücken, wird angenommen, dass diese Aufteilung beliebig gewählt werden kann. Dafür wird ein $s_t \in [0, 1]$ definiert, das jenen Anteil darstellen soll, der in die sichere Verzinsung r_t^{Sic} veranlagt wird. Diese sichere Verzinsung setzt sich dabei aus der, vom Unternehmen garantierten Verzinsung und der Gewinnbeteiligung zusammen. Damit kann r_t^{Sic} dargestellt werden als:

$$\forall t \in \mathbb{T}: \quad r_t^{Sic} = r_t^{Garantie} + r_t^{Gewinn}$$

Der Rest des Vorsorgekapitals wird mit r_t^{Sel} aus (5.1) verzinst. Damit gilt für die Mischung der beiden Varianten r_t^{Mix}, die zur Verzinsung am Ende des Jahres verwendet wird, die folgende Darstellung:

$$\forall t \in \mathbb{T}: \quad r_t^{Mix} = s_t r_t^{Sic} + (1 - s_t) r_t^{Sel}.$$

Die Formel für r_t^{Mix} kann zur Einbeziehung der laufenden Kosten eines Vertrages, wie Verwaltungskosten γ und Inkassokosten β, erweitert werden. Dafür sei δ die Summe dieser laufenden Kosten pro Jahr die direkt proportional zur Beitragssumme ist. Diese Kosten werden durch δ^{Sic} und δ^{Sec} auf die beiden Komponenten aufgeteilt. Daher gilt:

$$\forall t \in \mathbb{T}: \quad r_t^{Mix} = s_t(r_t^{Sic} - \delta^{Sic}) + (1 - s_t)(r_t^{Sel} - \delta_t^{Sel}). \tag{5.2}$$

Die Vertragsentwicklung (=account value) AV_t eines Select-Produktes gegen **Einmalerlag** (EB) lässt sich damit berechnen durch:

$$\begin{aligned} AV_0 &= EB - (\alpha \cdot EB) \\ AV_t &= AV_0 \cdot \prod_{k=1}^{t} (1 + r_k^{Mix}) \quad \forall t \in \mathbb{T} \end{aligned} \tag{5.3}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

wobei α die Abschlusskosten zur Zahlung der Vermittlerprovision bezeichnet.
Die Formel (5.3) für den Wert des Vertrages kann auch rekursiv dargestellt werden:

$$AV_t = AV_{t-1} \cdot (1 + r_t^{Mix})$$

Zusätzlich gilt für Select-Produkte, dass mindestens $EB > AV_0$ als Garantie zum Zeitpunkt T am Versicherungskonto vorhanden sein muss, da diese Produkte eine Beitragsgarantie zum Ende der Ansparphase beinhalten.

Wählt der Versicherungsnehmer anstatt eines Einmalerlags **laufende Prämienzahlungen** setzt sich der Vertragswert wie folgt zusammen:
Beiträge, welche bis zum ersten Indexstichtag einbezahlt wurden, werden verzinst angesammelt und und bilden, zusammen mit dem, zum Indexstichtag fälligen, Beitrag die Bezugsgröße zur Indexparizipation des beginnenden Indexjahres. Der in Folge investierte Beitrag wird bis zum nächsten Indexstichtag des Folgejahres verzinst angesammelt.
Unterjährig bezahlte Beiträge werden nicht in die Bezugsgröße zur Indexpartizipation eiberechnet und werden getrennt mit dem Zins für unterjährig bezahlte Beiträge verzinst. Diese Verzinsung wird jährlich neu für ein Jahr festgelegt und stimmt nicht zwangsläufig mit dem sicheren Zinssatz überein. Bei der Festlegung wird die Gesamtverzinsung des Versicherers als Grundlage herangezogen.
Die Vertragsentwicklung lässt sich als für laufende Prämienzahlungen P_t und dem Zinssatz r_t^m für unterjährige Beträge nach Kostenabzug durch Zahlweise m ($m = 1, 2, 4, 12$) rekursiv berechnen durch:

$$\begin{aligned} AV_0 &= P_t - (\alpha \cdot P_0) \\ AV_t &= AV_{t-1}(1 + r_t^{Mix}) + P_t(1 + r_t^1) \quad \forall t \in \mathbb{T} \end{aligned} \tag{5.4}$$

wobei hier nur die einjährige Zahlweise betrachtet wird.

5.1.1 Modelle

Bei Select-Produkten hängt die Höhe der Jahresrendite von der Entwicklung des Aktienindex am Finanzmarkt ab. Daher ist zur Bewertung eines solchen Produktes ein geeignetes Modell für dessen Verlauf notwendig.
Zusätzlich bilden Zinsstrukturkurven von Staatsanleihen für die Anlage eines Versicherungsunternehmens die Grundlage für den risikolosen Zinssatz. Bei der klassischen Lebensversicherung erhalten Versicherungsnehmer zusätzlich eine garantierte Verzinsung über eine lange Laufzeit. Daher ist die Verwendung von guten Modellen zur Beschreibung von Zinsentwicklungen maßgeblich für diese Unternehmen.
Infolge werden wir sowohl für den Verlauf des Aktienindex, als auch für den Zinsverlauf jeweils ein geeignetes und in der Praxis oft genutzes Modell vorstellen.

Modellierung des Aktienkurses

Nach dem, von Paul Samuleson entwickelten, stochastischen Modell zur Beschreibung der Entwicklung von Wertpapieren sind Renditen normalverteilt und Aktienkurse implizit

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

log-normalverteilt. Dieses Modell wird als geometrische Brown'sche Bewegung bezeichnet.

Der Kurs einer Aktie zum Zeitpunkt t (S_t) ist zum Zeitpunkt t am Aktienmarkt beobachtbar. Der zukünftige Verlauf des Aktienkurses ist zu diesem Zeitpunkt jedoch noch mit Unsicherheiten behaftet.

Ist nun $(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{T}, \mathbb{F}, S)$ ein Marktmodell mit $\mathbb{T} = [0, T], T > 0$ auf dem nur ein Risikopapier $(S_t)_{t\in\mathbb{T}}$ exisitiert. Weiters sei $r > 0$ eine feste Zinsrate und existiere auf dem Markt eine risikofreie Anlagemöglichkeit S^0, welche die Differentialgleichung

$$dS_t = rS_t^0 dt$$

für den Startwert S_0^0 erfüllt.
Dann kann für die Beschreibung des Aktienkursbewegung die folgende stochastische Differentialgleichung mit Startwert S_0 verwendet werden:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \qquad \text{mit} \qquad \mu \in \mathbb{R} \qquad \text{und} \qquad \sigma > 0 \qquad (5.5)$$

Hier bezeichnet der Parameter μ die erwartete mittlere Kursentwicklung der Aktie pro Zeiteinheit dt und σ die Standardabweichung des Preises.

Definition 5.1.1. *Für eine Standard Brown'sche Bewegung $(W_t)_{t\in\mathbb{T}}$ ist*

$$S_t = S_0 \; exp\left[\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right]$$

als die Lösung von (5.5) eine geometrische Brown'sche Bewegung.

Vasicek Zinsmodell

Auch die Wahl eines geeigneten Modells zur Beschreibung der Zinsentwicklung spielt für Versicherungsunternehmen eine Große Rollte. Ein Modell, welches zu diesem Zweck oft herangezogen wird, ist das Short-Rate-Modell von Vasicek. Dieses ist ein sogenanntes Ein-Faktor-Modell, was bedeutet, dass die Differenetialgleichung, zur Beschreibung der Veränderung, auf einer eindimensionalen Standard Brown'schen Bewegung basiert.
Die Zinsrate ist dabei gegeben als die Lösung der folgenden Differentialgleichung:

$$dr_t = \kappa(\theta - r_t)dt + \sigma dW_t \qquad \text{mit} \qquad r_0 > 0$$

wobei die Parameter $\kappa, \theta, \sigma > 0$ sind. Mit κ wird hier die sogenannte "Mean-Reversion-Rate" bezeichnet, also die Geschwindigkeit mit welcher der Prozess zum "Mean-Reversion-Level" θ zurückkehrt. Der Parameter σ steht für die Volatilität.
Dieses Modell hat die Form eines sogenannten Ornstein-Uhlenbeck Prozesses welcher wie folgt definiert ist:

Definition 5.1.2. *Ist für $c > 0$ und eine Standard Borwn'sche Bewegung W_t der Prozess X die Lösung einer stochastischen Differentialgleichung der Form*

$$dX_t = -cX_t dt + dW_t$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

so heißt X Ornstein-Uhlenbeck Prozess. Diese haben die Eigenschaft, dass sie immer versuchen zum "Mean-Reversion-Level" θ zurückzukehren, wobei gilt, dass je weiter X von θ entfernt ist, desto stärker ist der Drift zu θ.

Die wichtigesten Eigenschaften der Zinsrate nach dem Vasicek Modell sind die Normalverteilung der r_t bezüglich dem risikolosen Wahrscheinlichkeitsmaß. Daraus folgt, dass r_t mit positiver Wahrscheinlichkeit auch negative Werte annehmen kann. Dies wird als Nachteil dieses Modells angesehen. Jedoch kann dies, besonders in Krisensituationen sogar als sehr nützlich angesehen werden. So war beispielsweise im April 2014 ein negativer 3-Monats Euribor beobachtbar.
Eine weitere nachteilige Eigenschaft dieses Modells ist, dass diese nur von drei verschiedenen Parametern abhängt und damit die Möglichkeit, diese an Marktdaten zu kalibieren nicht vorhanden ist.

Da Zinssätze durch eine Vielzahl an unvorhersehbaren Ereignissen beeinflusst werden können die aus Faktoren entstehen, welche nicht vollständig modelliert werden können, sind Zinssatzmodelle im Allgemeinen als ungenau anzusehen.

5.2 Implementierung in R

Der hier verwendete Code zur Simulation der Wertentwicklung der Select-Produkte basiert auf der Arbeit [15, Pasztor].

5.2.1 Zinsentwicklung

Wie in 5.1 ersichtlich, basiert die Verzinsung des Vertrages auf einem Basis-Zinssatz. Dieser kann mit dem Vasicek Modell aus Kapitel 5.1.1 beschrieben, mit Hilfe einer Standard Brown'schen Bewegung simuliert werden. Da Brown'sche Bewegungen normalverteilt sind, wird im R-Code diese mittels einer zufälligen Wahl aus einer Normalverteilung simuliert.
Im Code muss weiters die Anzahl der Simulationen, die Laufzeit des Vertrages, der Anfangswert des Zinsverlaufes, sowie die Parameter, im Vorhinein, initialisiert werden. Dafür werden hier folgende Werte verwendet:

r_0	0.00
θ	0.01
k	0.5
σ	0.01

Diese Werte sind analog zu [**?**,]asztorEva gewählt.

]

```
## Simuliere Zinsen
n = 100   # Anzahl der Simulationen
T = 10    # Zeit
m = 20   # Laufzeit
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
dt = T/m  #Abstand der Zeitintervalle
r = matrix(0,nrow=m+1,ncol=n)  # Zinsmatrix
r[1,] = r0  #1. Zeile der Zinsmatrix sind die Eintraege = r0
for(j in 1:n){
  for(i in 2:(m+1)){
    dr = k*(theta-r[i-1,j])*dt + sigma*sqrt(dt)*rnorm(1,0,1)
      #rnorm erzeugt eine standrad normalverteile ZV W_t
    r[i,j] = r[i-1,j] + dr
  }
```

5.2.2 Indexentwicklung

Wie die Zinsentwicklung basiert auch die Indexentwicklung auf einer Standard Brown'schen Bewegung. Bei der Simulation wird hier die Monatsrendite betrachtet, weshalb (12· Laufzeit), Werte berechnet werden müssen.
Hier sind, wie bei der Simulation der Zinsentwicklung, der Anfangswert des Index sowie die Parameter wie folgt initialisiert:

s_0	10
μ	0.01
σ	0.05

]

```
#Funktion
s0 = as.vector(s0)
nsteps=length(periods)
dt = c(periods[1],diff(periods))
drift = mu - 0.5*sigma^2

temp = matrix(exp(drift*dt + sigma*sqrt(dt)*rnorm(nsteps*n)), ncol=n)
for(i in 2:nsteps) temp[i,]=temp[i,]*temp[(i-1),]
```

5.2.3 Cap

Der in Kapitel 5 beschrieben Cap wird durch das sogenannte Bisektionsverfahren bestimmt. Dieses stellt eine numerische Methode zum Auffinden von Nullstellen stetiger Funktionen dar, welches aus dem Beweis des Zwischenwertsatzes von Bolzano folgt.
Aus diesem Satz folgt, dass für eine, auf einem abgeschlossenen Intervall $[a, b]$, stetige Funktion f mit $f(a) < 0$ und $f(a) > 0$ gilt, dass sie mindestens einmal die x Achse in diesem Intervall kreuzt, da wegen der Stetigkeit Sprünge ausgeschlossen sind. Das bedeutet, dass f zumindest eine Nullstelle auf dem offenen Intervall (a, b) hat.

Das Bisektionsverfahren wird wie folgt in R implementiert:

]

```
bisek <- function (f, a, b, num = 50, eps = 1e-05){
  h = abs(b - a)/num
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
  i=0
  j=0
  a1 = b1 = 0
  while (i <= num) {
    a1 = a + i * h
    b1 = a1 + h
    if (f(a1) == 0) {
      return(a1)
    }
    else if (f(b1) == 0) {
      return(b1)
    }
    else if (f(a1) * f(b1) < 0) {
      repeat {
        if (abs(b1 - a1) < eps)
          break
        x <- (a1 + b1)/2
        if (f(a1) * f(x) < 0)
          b1 <- x
        else a1 <- x
      }
      j=j+1
      return(((a1 + b1)/2))
    }
    i=i+1 }
  if(j==0){return("Nullstelle konnte nicht gefunden werden")}
}# Funktionsende bisek
```

Diese Funktion wird dann zu jedem Zeitpunkt während der Laufzeit zur Berechnung des jeweiligen Caps aufgerufen.

```
]
a <- function(cap)
        {r.sicher[1,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                  [((12*(1))-11):(12*(1)),],2,sum),0)))}
b <- function(cap)
        {r.sicher[2,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                  [((12*(2))-11):(12*(2)),],2,sum),0)))}
u <- function(cap)
        { r.sicher[3,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                  [((12*(3))-11):(12*(3)),],2,sum),0)))}
d <- function(cap)
        { r.sicher[4,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                  [((12*(4))-11):(12*(4)),],2,sum),0)))}
.
.
.
w <- function(cap)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
        { r.sicher[14,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                [((12*(14))-11):(12*(14)),],2,sum),0)))}
o <- function(cap)
        { r.sicher[15,4]-
        -(mean(pmax(apply(pmin(m_perform,cap)
                [((12*(15))-11):(12*(15)),],2,sum),0)))}

A=bisek(a,0.01,0.05,50,0.00001)
B=bisek(b,0.01,0.05,50,0.00001)
U=bisek(u,0.01,0.05,50,0.00001)
D=bisek(d,0.01,0.05,50,0.00001)
.
.
.
W=bisek(w,0.01,0.05,50,0.00001)
O=bisek(o,0.01,0.05,50,0.00001)
cap.hoehe=c(A,B,U,D,E,F,G,H,I,J,K,L,V,W,O)
```

Dabei wird angenommen, dass die Höhe des Cap bei Verträgen mit Einmalerlag gleich ist wie bei prämienpflichtigen Verträgen. Dies kann damit begründet werden, dass die Höhe der sicheren Verzinsung und auf der Optionspreise trotz verschiedener Zahlweisen gleich bleiben. Daher wird angenommen, dass der Betrag `AVo*r.sicher` investiert wird.
Der Cap ist dann die Nullstelle jener Funktion f, die für einen bestimmten Cap durch die Differenz aus sicherer Veranlagung und Optionpreis definiert wird.
Im Vektor `cap.hoehe` werden die gefundenen Nullstellen gespeichert und zur Berechnung des Select Zinses verwendet.
Besitzt die Funktion keine Nullstelle, so wird "Nullstelle konnte nicht gefunden werden" ausgegeben, was beispielsweise der Fall sein kann, wenn die Gesamtverzinsung 0% beträgt. In diesem Fall wird mit dem garantierten Zinssatz verzinst, und das investierte Kapital ist gleich 0 EUR, wodurch keine Option gekauft werden kann.

5.2.4 Select-Zins

Für die Simulation des Select-Zinses wird der, nach dem Vasicek-Modell, simulierte Zins in die Formel (5.1) zu dessen Berechnung eingesetzt. Die Simulation ergibt eine Matrix, welche die Jahresrendite aus der Indexpartizipation enthält. Dabei wird der Zins auf 0 gesetzt, falls der Index negative Werte annimmt.

```
]
y_perform=matrix(0,m,n)
l=0
while(l<=m-1){
   l=l+1
   y_perform[l,]=apply(m_perform[((12*l)-11):(12*l),],2,sum)
}
y_perform

r.select=pmax(y_perform,0)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
r.select  #Jahresrendite bei Indexpartizipation
```

Dieser Code zeigt die Berechnung der Jahresrendite bei Indexpartizipation ohne Einbeziehung eines Cap. Wird ein solcher in die Verzinsung mit einbezogen, so wird dies folgender Maßen impelentiert:

```
]

cap.neu=matrix(0,m2,n)
cap.matrix=cap.neu[rep(1:nrow(cap.neu),each=12),]

capped_perform=matrix(0,(12*m2),n)
l=0
while(l<=((12*m2)-1)){
   l=l+1
   capped_perform[l,]=as.numeric(pmin(m_perform[l,],cap.matrix[l,]))
}

y_perform=matrix(0,m2,n)
l=0
while(l<=m2-1){
   l=l+1
   y_perform[l,]=apply(capped_perform[((12*l)-11):(12*l),],2,sum)
}

r.select=pmax(y_perform,0)
```

5.2.5 Mix-Zins

Da der Versicherungsnehmer auch jährlich die Wahlmöglichkeit hat zum Teil am Index zu partizipieren und den restlichen Teil mit dem garantierten Zins zu verzinsen, muss auch darauf in der Simulation rücksicht genommen werden. Wie bereits in Formel (5.2) ersichtlich, wird das mit Hilfe eines Parameters s umgesetzt. Hier wird dies durch einen, im Vorhinein, initialisierten Vektor mit der Dimension gleich der Laufzeit, umgesetzt, da anderenfalls die Entscheidungstheorie mit einbezogen werden müsste. Dies würde jedoch den Rahmen dieser Arbeit übersteigen.

```
]
r.mix=pmax(r.sicher,0)*s+(1-s)*pmax(r.select,0)
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

6 Praktische Umsetzung

6.1 Daten

Zum Training der Künstlichen Neuronalen Netze und folgenden Berechnung des SCR wurden zwei verschiedene Datensätze mittels Nested Monte Carlo Simulation simuliert. Diese sollen zwei verschiedene Portfolios aus Select Produkten darstellen.
Das erste besteht aus nur einem Produkt mit Einmalerlag von 10000,- und ohne Cap. Dieses sollte dazu dienen erste Daten zu liefern.
Das Produkt hat eine Laufzeit von 20 Jahren und der Entscheidungsvektor hatte folgende Form:

t	1	2	3	4	5	6	7	8	9	10
s	0	0.5	0.5	0.25	0.5	0	0.5	0	0.5	0.25
t	11	12	13	14	15	16	17	18	19	20
s_1	0.5	0	0.25	1	0.25	0.5	0	0.5	0.75	1

Tabelle 6.1: Entscheidungsvektor Produkt 1

Für den zweiten Datensatz wurden 2 weitere Produkte zu dem Portfolio hinzugefügt. Zum einen ein Select Produkt mit einem Einmalerlag von 10500,- und einem Cap für den Zins bei Indexpartizipation. Zum anderen ein Produkt mit einer jährlichen Prämie von 900, jedoch ohne Cap.
Das Produkt mit Cap hat eine Laufzeit von 15 Jahren, das mit jährlicher Prämienzahlung eine Laufzeit von 25 Jahren.
Die beiden Entscheidungsvektoren sehen folgendermaßen aus:

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_2	0.25	0.25	1	0.5	0.75	0.75	0.25	0.75	0.25	1	0.25	0.75	0	0.5	0.25

Tabelle 6.2: Entscheidungsvektor Produkt 2

Für beide dieser Portfolios wurde der Marktwert zum Zeitpunkt 0, unter der Verwendung

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_3	0.5	1	0	0.75	0.25	0.5	0	0.25	1	0.75	0	1	0.25	0.25	0.75
t	16	17	18	19	20	21	22	23	24	25					
s_3	0.5	1	0.75	0.5	0.75	1	0.75	0	0.5	1					

Tabelle 6.3: Entscheidungsvektor Produkt 3

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

von 10000 Simulationen berechnet, sowie für die ersten 1000 Pfade dieser äußeren Schleife der Marktwert zum Zeitpunkt 1 unter der Verwendung von jeweil 1000 Simulationen.
Die Trainingsmatrix für die Neuronalen Netze bestand aus den Werten aus der Simulation des Zinssatzes zum Zeitpunkt 1, den Werten aus der Simulation des Index zum Zeitpunkt 1, den Portfoliowerten zum Zeitpunkt 1, jeweils aus der äußeren Schleife, und den zugehörigen Werten des Marktwertes zum Zeitpunkt 1 aus der jeweiligen inneren Schleife. Die Marktwerte zum Zeitpunkt 1 wurden als sogenannte Labels oder Vergleichswerte während dem Training verwendet.

Zum Training wurden die Daten mit dem sogenannten MinMaxScaler skaliert. Dieser zieht von jedem Wert das Minimum ab und dividiert das Resultat durch die Differenz aus Minimum und Maximum.

6.2 Aktivierungsfunktionen

Da die Aktivierungsfunktion in den Neuronalen Netzen eine Skalierung der Input Daten für jeden Knoten vornimmt ist die Wahl dieser ebenfalls zu beachten. Dabei wird hier angenommen, dass die Wahl der Aktivierungsfunktion von der Skalierung der gewünschten Daten und damit der Labels im Zusammenhang stehen sollte.
Um die Auswirkung der Aktivierungsfunktion auf die hier verwendeten Daten zu vergleichen, wurden, für jede der in Kapitel 4.1 angeführten Funktionen, 5 Netze mit jeweils 4 hidden Layer und 16 Knoten pro Layer trainiert. Dazu wurde in 500 Trainingsläufen die Parameter angepasst und der Fehler berechnet. Zum Vergleich der Resultate wurde der Durchschnitt der Fehler der 5 Netze mit der selben Architektur berechnet.
Die folgende Tabelle zeigt die Ergebnisse daraus für den Test mit nur **einem Produkt**:

	loss
sigmoid	0.002556346
relu	0.000475663
tanh	0.000494536
softsign	0.000403909
elu	0.000424179
linear	0.000711799

Tabelle 6.4: Vergleich der Aktivierungsfunktionen für nur ein Produkt

Hier sehen wir, dass besonders der Fehler bei der Verwendung der sigmoid Aktivierungsfunktion relativ groß ist. Was eventuell daran liegen könnte, dass diese Funktion die Werte auf einen Bereich mit der kleinsten Breite projiziert.

Die folgende Tabelle zeigt die Ergebnisse daraus für den Test mit den **drei verschiedenen Produkten**:
Hier sehen wir, dass jene Aktivierungsfunktionen, deren Definitionsbereich sowohl nach unten als auch noch oben beschränkt ist, die besseren Resultate liefern.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

	loss
sigmoid	0.000017755
relu	0.000055412
tanh	0.000018418
softsign	0.000007654
elu	0.000084541
linear	0.000176215

Tabelle 6.5: Vergleich der Aktivierungsfunktionen für 3 Produkte

In beiden Fällen kann gesagt werden, dass der Tangens Hyperbolicus und die Softsign Aktivierungsfunktion gute Ergebnisse liefern. Dies könnte damit begründet werden, dass diese Aktivierungsfunktionen jene sind, deren Definitionsbereich zwar beschränkt ist, jedoch sowohl negative, als auch positive Werte beinhaltet.

6.3 Layer vs. Knoten pro Layer

In diesem Teil der Arbeit wurden 54 verschiedene Netze mit den selben Trainingsdaten darauf untersucht, wie sich die Wahl der Knotenzahl und der Layerzahl auf den quadratischen Fehlern des Validierungssets auswirken. Die Frage war hier, ob eher eine größere Anzahl an Layern oder an Knoten pro Layer in einem geringeren Fehler resultieren.
Zu diesem Zweck wurden für jede der in Kapitel 4.1 vorgestellten Aktivierungsfunktionen 5 mal die aus den verschiedenen Netzen resultierenden Fehler, nach 500 Trainingsläufen, berechnet und daraus der Durchschnitt gebildet. Die Anzahl der Layer und Knoten sind dabei immer Potenzen von 2. Die Maximalanzahl der Knoten des Neuronalen Netzes insgesamt waren dabei 512.

Für das **Portfolio mit einem Produkt** sind die Durchschnitte der 5 Durchläufe für die unterschiedlichen Aktivierungsfunktionen in den folgenden Tabellen dargestellt.

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,0029279	0,00237127	0,00176571	0,00092013	0,00066345	0,00050138	0,00043247	0,0004531	0,00048395	0,00050915
	2	0,00237709	0,00121599	0,00131075	0,00076167	0,00060998	0,00046077	0,00041946	0,00053474	0,00048754	
	4	0,01925614	0,00165179	0,00057348	0,00050247	0,00041057	0,00047685	0,00047839	0,0005321		
	8	0,01389825	0,00164837	0,00099914	0,00040754	0,00061514	0,00072601	0,00039776			
	16	0,00763118	0,00152033	0,00074154	0,00086433	0,00088842	0,0006148				
	32	0,01402984	0,00844003	0,00094603	0,00071981	0,00049838					
	64	0,03051481	0,02183104	0,0192083	0,01597501						
	128	0,03058465	0,030548	0,03049454							
	256	0,03050853	0,03057107								

Tabelle 6.6: Ergebnisse für Aktivierungsfunktion tanh (1 Produkt)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00227105	0,00211722	0,00164536	0,00125513	0,00044179	0,00039608	0,00038431	0,00044957	0,00060818	0,00049921
	2	0,00258581	0,00212288	0,0022369	0,00041843	0,00040689	0,0004099	0,00048007	0,00045412	0,00065432	
	4	0,00229648	0,0021572	0,00081047	0,00041146	0,00049574	0,0006402	0,00053531	0,00051327		
	8	0,00239712	0,00135341	0,00087249	0,00050306	0,00040643	0,00048837	0,00043819			
	16	0,00261815	0,00166193	0,00170986	0,00058981	0,00057767	0,00054826				
	32	0,02716951	0,00707603	0,00399313	0,00046489	0,00056625					
	64	0,03050985	0,03052106	0,02878929	0,03015085						
	128	0,0305029	0,03050621	0,0305668							
	256	0,030565	0,03061346								
	512	0,03054251									

Tabelle 6.7: Ergebnisse für Aktivierungsfunktion softsign (1 Produkt)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00172009	0,00237115	0,00186833	0,00115134	0,00106473	0,00045653	0,00059099	0,00046561	0,00058403	0,00055331
	2	0,00278342	0,00125411	0,00088972	0,00045634	0,00039106	0,00044746	0,0006412	0,00044884	0,00042534	
	4	0,00234164	0,00090168	0,00046743	0,00041309	0,00055153	0,00040754	0,00054476	0,00047328		
	8	0,0017447	0,00142197	0,00053522	0,00042038	0,00054772	0,00046286	0,00044316			
	16	0,00772247	0,00191894	0,00049909	0,00062779	0,00053259	0,00042963				
	32	0,01400499	0,00205532	0,00070828	0,00061984	0,00053424					
	64	0,03047588	0,02414618	0,02097658	0,00328348						
	128	0,0305515	0,03051491	0,03059853							
	256	0,03046434	0,030543								

Tabelle 6.8: Ergebnisse für Aktivierungsfunktion elu (1Produkt)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00282506	0,0025474	0,00257838	0,00252737	0,00233761	0,00222676	0,00224295	0,0019815	0,00197565	0,00199539
	2	0,00275372	0,00263951	0,00261992	0,00257646	0,00252629	0,00217507	0,00210326	0,00260557	0,0030983	
	4	0,00377921	0,0026465	0,00250865	0,00252688	0,00254819	0,00244585	0,00226633	0,00222209		
	8	0,01553226	0,0022844	0,00226417	0,00217994	0,00236448	0,00254338	0,00248068			
	16	0,03055531	0,03056714	0,03058543	0,03048123	0,03061537	0,03055434				
	32	0,03054405	0,03053477	0,03057419	0,03071186	0,03074388					
	64	0,03052516	0,0305931	0,03064705	0,03056368						
	128	0,03053418	0,03056108	0,03051858							
	256	0,03052719	0,03049374								

Tabelle 6.9: Ergebnisse für Aktivierungsfunktion sigmoid (1 Produkt)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,01957279	0,01475113	0,00212476	0,00192213	0,00162359	0,00097652	0,00052687	0,00042772	0,00060563	0,00046722
	2	0,03052899	0,01333354	0,00783155	0,00100375	0,00117418	0,00038906	0,00048011	0,00052434	0,00060264	
	4	0,02450337	0,01367132	0,0074578	0,00648823	0,00053513	0,00046504	0,00051073	0,00054886		
	8	0,03051579	0,01285283	0,01350267	0,00683772	0,00055243	0,00045058	0,00051045			
	16	0,03054644	0,03051673	0,03053216	0,0009394	0,00070376	0,00052763				
	32	0,0305494	0,03054482	0,03051707	0,02488587	0,00670192					
	64	0,03055038	0,03052586	0,03054905	0,03060314						
	128	0,0305336	0,03052539	0,03052118							
	256	0,03052435	0,03054943								

Tabelle 6.10: Ergebnisse für Aktivierungsfunktion relu(1 Produkt)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00315570	0,0041585	0,00271687	0,00185647	0,00087168	0,00041301	0,00041715	0,00064579	0,00042669	0,00044135
	2	0,00344146	0,00310407	0,00165597	0,00058208	0,0004339	0,00047928	0,00046185	0,00052766	0,00058744	
	4	0,00394833	0,00289328	0,00050362	0,00050011	0,00061499	0,00043149	0,00057916	0,00042552		
	8	0,00383341	0,00306233	0,00039503	0,00051922	0,00049721	0,0004876	0,00043345			
	16	0,0036274	0,00202118	0,00044194	0,00079321	0,0005421	0,00061286				
	32	0,00920528	0,00131496	0,00063468	0,00048619	0,00050081					
	64	0,02534996	0,00273806	0,00132843	0,00059973						
	128	0,03054312	0,03066392	0,02445899							
	256	0,03049895	0,0305377								

Tabelle 6.11: Ergebnisse für Aktivierungsfunktion linear(1 Produkt)

In diesen Tabellen wurden die 10 größten Fehler orange und rot makiert, wobei rot das Maximum ist. Die 10 kleinsten Fehler sind blau und grün markiert wobei grün der kleinste resultierende Fehler ist.
Aus diesen Tabellen ist gut ersichtlich, dass eine höhere Anzahl an Layer, auch wenn die Anzahl der Knoten pro Layer klein ist, schlechtere Resultate erzielen als eine kleinere Anzahl an Layern mit mehr Knoten.
Weiters kann in den Diagonalen der Tabellen der Fehler jener Netze mit der selben Gesamtanzahl an Knoten abgelesen werden. Was hier besonders in Tabelle 6.7 ins Auge fällt ist, dass auch bei der gleichen Anzahl an Knoten eine geringere Layerzahl von Vorteil ist. So liegt hier der Fehler für ein Neuronales Netz mit 8 Layern mit jeweils 64 Knoten bei 0.00039776 jedoch für ein Netz mit 64 Layern und nur jeweils 8 Knoten liegt der Fehler bei 0.01597501. Bei beiden Netzen liegt die Gesamtanzahl der Knoten bei 512.

Auf die Frage, ob die Anzahl der zu trainierenden Parameter der Netze einen Einfluss auf den Fehler haben könnte kann abermals mit Hilfe von Tabelle 6.7 gut beantwortet werden. So besitzt ein Neuronales Netz mit 128 Layern mit je einem Knoten 260 zu trainierende Parameter, ein Netz mit 4 Layern und je 8 Knoten hat 257 zu trainierende Parameter, also eine sehr ähnliche Anzahl. Wie wir aus der Tabelle ablesen können liegt der Fehler des Netzes mit mehr Layern jedoch bei 0.03058176, der mit weniger Layern bei 0.00050248.

Für das **Portfolio mit 3 unterschiedlichen Produkten** zeigen die folgenden Tabellen den Durchschnitt der 5 Durchläufe, für die verschiedenen Aktivierungsfunktionen:

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,01353174	0,02028812	0,00678175	0,00003983	0,00002525	0,00001606	0,00002596	0,00679362	0,00005817	0,00006332
	2	0,00677467	0,00678807	0,00689658	0,00000733	0,00002820	0,00677662	0,00001989	0,00681986	0,00008038	
	4	0,01353096	0,00678975	0,00001271	0,00002397	0,00001936	0,00004557	0,00000800	0,00002394		
	8	0,02028862	0,01353034	0,00001295	0,00002947	0,00002395	0,00005459	0,00008403			
	16	0,03379365	0,02030465	0,00678746	0,00005898	0,00001364	0,00010680				
	32	0,03380721	0,03379353	0,01353909	0,00679732	0,00010559					
	64	0,03379935	0,03379457	0,03380332	0,03379401						
	128	0,03380803	0,03379749	0,03379580							
	256	0,03379482	0,03385684								

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tabelle 6.12: Ergebnisse für Aktivierungsfunktion tanh (3 Produkte)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00020482	0,00008109	0,00007224	0,00003463	0,00003205	0,00001663	0,0000146	0,00001888	0,00003835	0,00018705
	2	0,00679342	0,00003968	0,00006736	0,0000262	0,00001738	0,00001167	0,00002668	0,00003125	0,00006082	
	4	0,01353399	0,00001718	0,00002129	0,0000124	0,00001284	0,00002309	0,00001318	0,00004537		
	8	0,00005546	0,00000917	0,00001835	0,00002218	0,00004717	0,00002883	0,00001169			
	16	0,01354926	0,00002014	0,00003177	0,00001858	0,00006283	0,00002884				
	32	0,03379164	0,03380144	0,0203026	0,01357885	0,00681364					
	64	0,03379878	0,03379907	0,03381556	0,03380250						
	128	0,03379919	0,03380891	0,03380789							
	256	0,03380518	0,03379903								

Tabelle 6.13: Ergebnisse für Aktivierungsfunktion softsign(3 Produkte)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,01430977	0,00043423	0,00010053	0,00001782	0,00004571	0,00008482	0,00016606	0,00057405	0,00082028	0,00050167
	2	0,00676576	0,00013397	0,00002352	0,00003721	0,00002303	0,00074624	0,00022148	0,00200751	0,00027406	
	4	0,0270384	0,00003452	0,00001865	0,00005327	0,00002252	0,00001958	0,00002697	0,00004215		
	8	0,01358621	0,01353905	0,00002772	0,00000996	0,00003824	0,00003591	0,00001306			
	16	0,03380078	0,00003476	0,00000677	0,00004795	0,00004978	0,00002107				
	32	0,03380239	0,00687612	0,00001690	0,00005067	0,0000479					
	64	0,03379477	0,03371185	0,02705879	0,00681313						
	128	0,03379225	0,03381140	0,03381986							
	256	0,03380197	0,03379704								

Tabelle 6.14: Ergebnisse für Aktivierungsfunktion elu (3 Produkte)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,01703459	0,03391216	0,00010747	0,00005591	0,00002676	0,00002927	0,00003323	0,00005799	0,00019118	0,00031007
	2	0,06766348	0,03390637	0,0000758	0,00004054	0,00004321	0,00004277	0,00001366	0,00009473	0,00018335	
	4	0,06782374	0,01752381	0,00017358	0,00006414	0,00003433	0,00002273	0,00003676	0,00018765		
	8	0,05647704	0,05191284	0,03392627	0,00011903	0,00004814	0,00001301	0,00003621			
	16	0,08447676	0,08448739	0,08455458	0,0847412	0,08490089	0,08508155				
	32	0,08449085	0,08450722	0,08451441	0,08461576	0,08504664					
	64	0,08448068	0,08447786	0,08450158	0,08473140						
	128	0,08449952	0,08449524	0,08451427							
	256	0,0844771	0,08449518								

Tabelle 6.15: Ergebnisse für Aktivierungsfunktion sigmoid (3 Produkte)

		Knoten pro Layer									
		1	2	4	8	16	32	64	128	256	512
Layer	1	0,00568151	0,02216797	0,00079522	0,00002785	0,00003925	0,00001965	0,00004268	0,00032503	0,00003206	0,00007032
	2	0,04055104	0,03389234	0,0067717	0,00001919	0,00015566	0,00005284	0,00001027	0,00030338	0,00003575	
	4	0,03379357	0,0337920	0,00677826	0,00001223	0,0000179	0,00003345	0,00001596	0,00001439		
	8	0,0405494	0,04054928	0,02027629	0,00005377	0,00001445	0,00001565	0,00001067			
	16	0,04054973	0,04055236	0,03379809	0,00676351	0,00001098	0,00000747				
	32	0,04054909	0,04054893	0,04055067	0,02027873	0,00001457					
	64	0,04055225	0,04054993	0,04055178	0,04055778						
	128	0,04055001	0,04054964	0,04054924							
	256	0,04055081	0,04055136								

Tabelle 6.16: Ergebnisse für Aktivierungsfunktion relu (3 Produkte)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Auch hier ist ein ähnliches Muster wie in Tabelle 6.6 bis Tabelle 6.11 erkennbar. So sind, auch bei diesem Test, jene Werte, die grün oder gelb markiert sind, und damit die 10 kleinsten resultierenden Fehler darstellen, immer im oberen, rechten Teil der Tabelle zu finden. Also dort, wo die die Werte für jene Modelle mit weniger Layern und dafür mehr Knoten pro Layer zu finden sind. Andererseits sind auch hier die rot markierten Werte, und damit die 10 größten resultierenden Fehler, unten links zu finden, also in jenem Bereich, in dem der Fehler für die Modelle mit weniger Knoten pro Layer, dafür aber mehr Layern notiert ist.

Allgemein ist die Architektur eines Neuronalen Netzes im bezug auf die Anzahl der Layer beziehungsweise Knoten, die in einem Modell verwendet werden ist ein wichtiges Thema bei der Anwendung von Künstlichen Neuronalen Netzen auf verschiedenste Probleme. Theoretische Beweise allgemein gültiger Eigenschaften von Neuronalen Netzen sind jedoch sehr spezifisch. So zeigen Shaham et. all in Ihrem Paper [18, U.Shaham et al.], dass es einen Aufbau eines Neuronalen Netzes mit 4 Layer (also mit 3 Hidden Layer+einem Output Layer) gibt, sodass eine Funktion aus dem L^2 einer Manigfaltigkeit, beliebig genau approximiert werden kann. Dabei hängt die Anzahl der Knoten in jedem Layer, die dazu benötigt werden von der sogenannten Wavelet [1] (also von ihrem wellenartigen Charakter) der Funktion, ihrer Krümmung und der Dimension der Mannigfaltigkeit ab, auf der die Funktion definiert ist. Weiters ist die Anzahl der Knoten in jedem Layer unterschiedlich. Die angewandten Aktivierungsfunktionen sind dabei im ersten Hidden Layer die lineare, im zweiten und dritten die Rectified Linear Unit und im Output Layer wieder die lineare Aktivierungsfunktion.
Wie hier kurz dargestellt, bedeutet dies zwar, dass ein optimales Neuronales Netz mit 4 Layers erstellt werden kann, jedoch wiederum unter der Voraussetzung dass bestimmte Eigenschaften der zu approximierenden Funktion bekannt sind.

6.4 Capacity

Der Unterschied zwischen dem Optimierungsproblem im Machine Learning Setting zu einem Standard-Optimierungsproblem ist, dass das Machine Learning Modell nicht nur für die Trainingsdaten optimal sein soll, sondern auch für eine Datenmenge, die im Vorhinein noch unbekannt ist. Das bedeutet, der Fehler soll nicht nur für die Trainingsmenge sondern zusätzlich für eine Testmenge möglichst klein sein [5, Goodfellow et al.].
In diesem Abschnitt betrachten wir für die Aktivierungsfunktionen Tangens Hyperbolicus und Softsign, wie sich einerseits die Variation der Knotenzahl bei einer gleichbleibenden 4 Layer, andererseits die Variation der Anzahl der Layer bei gleichbleibender Knotenzahl von je 4 pro Layer, im Laufe des Trainings verändert. Die betrachteten Werte sind dabei der quadratische Fehler einer Test-Datenmenge, berechnet mit dem Modell für eine separate Trainingsmenge.
Die folgenden Tabellen zeigen zuerst die Ergebnisse für das Portfolio mit nur einer Verpflichtung und danach jene für das Portfolio bestehend aus drei Verpflichtungen:

[1] Für die Theorie zu Wavelet siehe zum Beispiel [13, Louis et al.]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

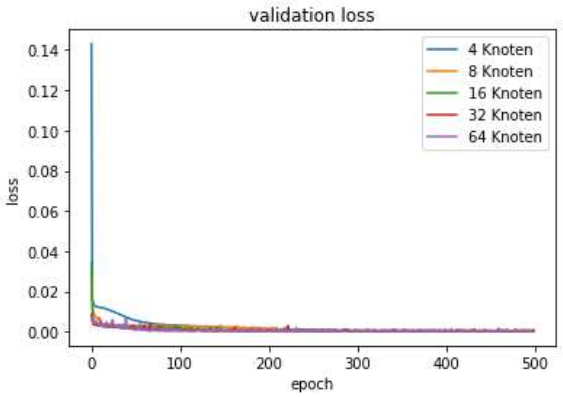


Abbildung 6.1: tanh Aktivierungsfkt. (1 Produkt); var. Knotenzahl

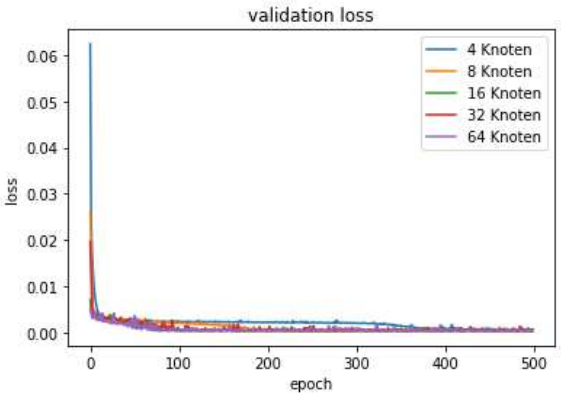


Abbildung 6.2: softsign Aktivierungsfkt. (1 Produkt); var. Knotenzahl

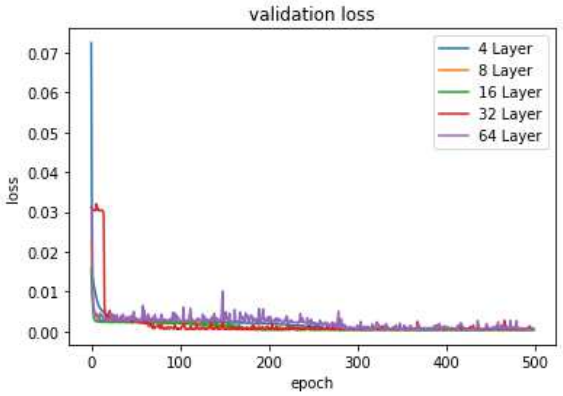


Abbildung 6.3: tanh Aktivierungsfkt. (1 Produkt); var. Layer

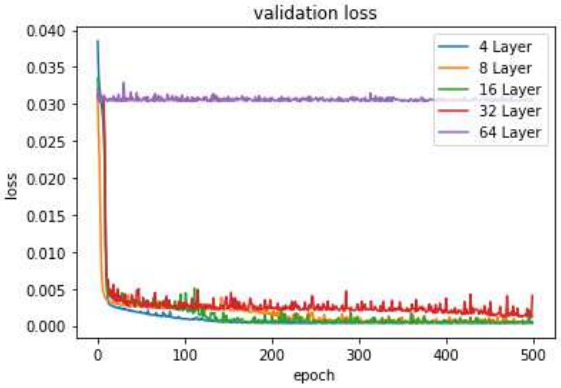


Abbildung 6.4: softsign Aktivierungsfkt. (1 Produkt); var. Layer

Hier sieht man bereits, dass eine Veränderung der Knotenzahl bei einer gleichbleibenden Anzahl der Layer in keiner großen Veränderung resultiert. Die Veränderung der Anzahl der Knoten bei gleichbleibender Layerzahl ist jedoch ausschlaggebend. So sieht man hier, wie bereits in Tabellen 6.6 bis 6.11 ersichtlich, dass eine hohe Anzahl an Layern in einem schlechteren Fehler resultiert als eine kleinere Anzahl an Layern. Besonders bemerkenswert ist hier bei der softsign Aktivierungsfunktion, dass es bei der Verwendung von 64 Layern einen großen Sprung macht.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

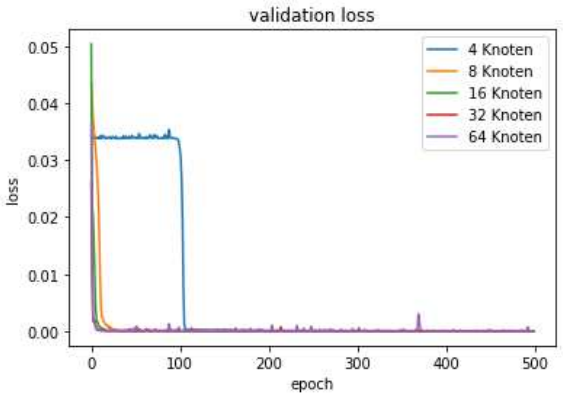


Abbildung 6.5: tanh Aktivierungsfkt. (3 Produkte); var. Knotenzahl

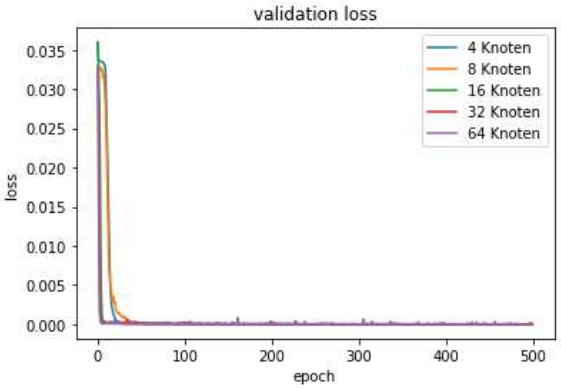


Abbildung 6.6: softsign Aktivierungsfkt. (3 Produkte); var. Knotenzahl

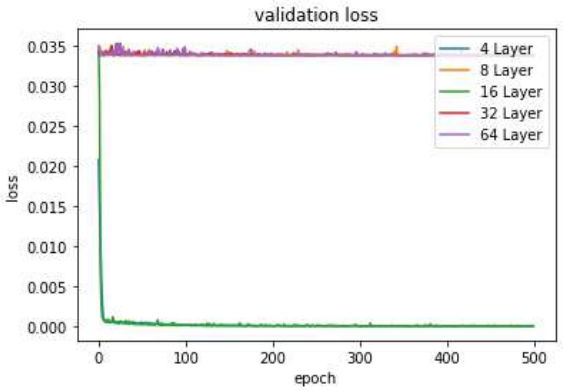


Abbildung 6.7: tanh Aktivierungsfkt. (3 Produkte); var. Layer

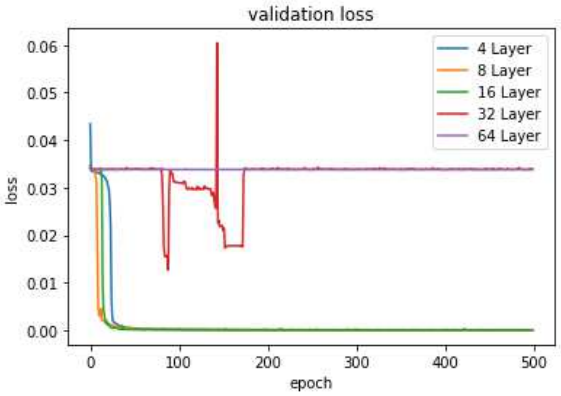


Abbildung 6.8: softsign Aktivierungsfkt. (3 Produkte); var. Layer

Auch beim zweiten Datensatz kann eine ähnliche Aussage wie beim ersten getroffen werden. Hier ist jedoch bei der Veränderung der Anzahl der Layer bereits bei 32 Layern, ein Sprung im Resultat zu sehen. In Abbildung 6.8 ist weiters zu sehen, dass bei einer Verwendung von 32 Layern mit jeweils 4 Knoten die Trainingszeit bei der Höhe des Fehlers eine große Rolle spielen kann, so ändert sich hier der Fehler zwischen dem 50. und dem 200. Traininsschritt stark.

In Abbildung 6.5 ist ebenfalls ersichtlich, dass ein zu früher Abbruch des Trainings zu einem hohen Fehler führen kann. So ist bei der Verwendung von 4 Knoten in 4 Layern der Fehler noch hoch und fällt erst nach dem 200. Trainingsdurchgang stark ab.

Zusammenfassend kann hier gesagt werden, dass besonders die Wahl der Layerzahl ausschlaggebend für die Performance des Modells ist. Zusätzlich sollte darauf geachtet werden,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

dass während der Optimierung genug Schritte zur Anpassung der Parameter durchgeführt werden.

6.5 SCR Berechnung

Zur Vervollständigung des praktischen Teils der Arbeits wurde zum Schluss das SCR berechnet. Dafür wurden 4 Modelle mit den Trainingsdaten trainiert und mit diesen der Marktwert zum Zeitpunkt 1 des Portfolios für die restlichen Daten berechnet. Dies sind jeweil 2 Modelle für die beiden Portfolios. Dabei wurden die beiden besten Aktivierungsfunkionen aus Kapitel 6.2 verwendet. Die zugehörige Anzahl der Knoten und Layer wurden ensprechend der Tabellen aus Kapitel 6.3 gewählt.
Die Labels aus der Trainingsmenge und die Resultate aus den Predictions mit den Modellen ergeben dann die Werte für das MVL_1 für jedes Szenario der äußeren Schleife.
Für das SCR selbst wird das $99,5\%$-Quantil der MVL_1 und das, mit der Nested Monte Carlo Methode berechnete MVL_0 in die Formel 2.4 eingesetzt. Daraus ergeben sich folgende Werte:

Aktivierungsfunktion	Layer	Knoten pro Layer	SCR
softsign	1	64	19978.06
elu	2	16	19978.06

Tabelle 6.17: SCR für das Portfolio aus einem Produkt

Aktivierungsfunktion	Layer	Knoten pro Layer	SCR
softsign	8	2	15873.41
tanh	2	8	15873.41

Tabelle 6.18: SCR für das Portfolio aus drei Produkten

Die Werte sind hier, in beiden Fällen, die selben. Wie weit diese von den, durch die Nested Monte Carlo Methode berechneten Werte entfernt liegen und inwiefern das problematisch ist, wäre eine Fragestellung für eine weiterführende Arbeit.

Die Größe dieser Werte, im Vergleich zu den bezahlten Beiträge des Versicherungsnehmers, lässt sich daraus begründen, dass nur bei einem der Produkte aus dem größeren Portfolio ein Cap für die Verzinsung bei Indexpartizipation einbezogen wird. Jedoch liegt für alle Produkte der Floor bei 0. Damit kann es dazu kommen dass die Verzinsung sehr hoch ausfällt.

Da in dieser Arbeit besonders die Performance der Neuronalen Netze im Vordergrund steht spielt die Höhe des SCR jedoch sowieso nur eine kleinere Rolle und wird hier Vollständigkeitshalber angegeben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tabellenverzeichnis

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abbildungsverzeichnis

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Literaturverzeichnis

[1] D. Bauer, A. Reuss, and D. Singer. On the calculation of the solvency capital requirement based on nested simulations. *Astin Bull.*, 42(2):453–499, 2012.

[2] L. Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 08 2001.

[3] H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, pages 1–21, 02 2019.

[4] EIOPA. Eiopa report on the fifth quantitative impact study (qis5) for solvency ii.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[6] A. Grosen and P. Jørgensen. Fair valuation of life insurance liabilities: The impact of interest rate guarantees, surrender options, and bonus policies. *Insurance: Mathematics and Economics*, 26(1):37–57, 2000.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistial Learning*.

[8] S. A. Hejazi and K. R. Jackson. A neural network approach to efficient valuation of large portfolios of variable annuities. *Insurance Math. Econom.*, 70:169–181, 2016.

[9] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, Mar. 1991.

[10] P. Jaeckel. *Monte Carlo Methods in Finance*.

[11] D. Kingma and J. Ba. Adam: a method for stochastic optimization (2014). *arXiv preprint arXiv:1412.6980*, 15, 2015.

[12] D. P. Kroese and R. Y. Rubinstein. Monte carlo methods. *WIREs Comp. Stat.*, 4:48–58, 2012.

[13] A. Louis, P. Maaß, and A. Rieder. *Wavelets: Theorie und Anwendungen*. Teubner Studienbücher Mathematik. Vieweg+Teubner Verlag, 2013.

[14] M. Nilsson and E. Sandberg. Application and evaluation of artificial neural networks in solvency capital requirement estimations for insurance products.

[15] E. Pasztor. Simulation und analyse von select produkten im niedrigzinsumfeld.

[16] J. Patterson and A. Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly, Beijing, 2017.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[17] P. Runhuan Feng, P. Zhenyu Cui, and M. Peng Li. Nested stochastic modeling for insurance companies.

[18] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *CoRR*, abs/1509.07385, 2015.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN