Projektarbeit

Thema:

Unterschiede zwischen SOA und Microservices

Stefan Kruk

geboren am 14.08.1992

Matr.-Nr.: 7084972

An der Fachhochschule Dortmund im Fachbereich Informatik erstellte

Projektarbeit

im Studiengang Softwaretechnik (Dual)

zur Erlangung des Grades akademischen Grades Bachelor/Master of Art

Betreuer: Prof. Dr. Johannes Ecke-Schüth

Fachhochschule
Dortmund
University of Applied Sciences and Arts

Fachbereich Informatik

Dortmund, 4. Januar 2016

Inhaltsverzeichnis

Überblick

Kurzfassung

Hier sollte eine halbseitige Kurzfassung der Arbeit stehen.

Abstract

Here, an abstract written in English should appear.

Kapitel 1

Einleitung

1.1 Motivation

WICHTIGER HINWEIS: Alles in diesem **Entwurf** den Aspekt **wissenschaftliches Arbeiten** bei der Anfertigung einer Bachelor-/Master-Arbeit Betreffende ist absolut **verbindlich** und muss uneingeschränkt berücksichtigt werden!

Alles in diesem **Entwurf** das **Layout** einer Bachelor-/Master-Arbeit Betreffende gibt die **persönliche** Meinung des Verfassers wieder! Jede Studentin und jeder Student mag eigene Vorstellungen entwickeln! Hier findet man eventuell erste hilfreiche Anhaltspunkte.

Nun zum Überblick! Was sollte er u. a. enthalten:

- Erläuterung der Problemstellung
- Motivation für die Beschäftigung mit dem Problem
- Hinweis auf eventuell schon vorhandene Entwicklungen
- Abgrenzung der eigenen Arbeit von eventuell schon vorhandenen Entwicklungen
- kurzer Abriss über den Inhalt der Arbeit

1.2 Notation

In diesem Abschnitt wird die in dieser Arbeit verwendete Notation vorgestellt und erläutert, falls sie relativ aufwendig ist.

1.3 Grundlagen

Wohlbekannte, aber für diese Arbeit besonders wichtige Resultate finden sich hier, wenn sie denn benötigt werden. Wichtig in diesem und in allen folgenden Kapiteln: **Zitate angeben!!!** Wann immer etwas aus einem Buch, einer Veröffentlichung, einem Vortrag oder einer www-Page entnommen ist, **muss** dies durch eine entsprechende Angabe der Quelle im Text (z.B. [vgl. 9, S. 23ff]) sowie einer Angabe der vollständigen Quelle im Literaturverzeichnis kenntlich gemacht werden! Es ist absolut unzulässig, längere Passagen **wörtlich** oder **sinngemäß und nahezu wörtlich** aus einem anderen Dokument zu übernehmen, ohne dies präzise zu zitieren, auch wenn man die Referenz pauschal im Literaturverzeichnis angibt (Plagiat, nicht bestanden, keine Wiederholung möglich). Im Rahmen der Erklärung am Ende der Bachelor-/Master-Arbeit verpflichtet sich die Studentin bzw. der Student, dieser, im Rahmen wissenschaftlicher Arbeit fundamentalen Pflicht, alle Quellen angegeben zu haben und Zitate kenntlich gemacht zu haben, nachgekommen zu sein.
Überwiegt bei einer Arbeit der Anteil korrekt zitierter, aber mehr oder weniger wörtlich übernommener Passagen, ist sie zwar im engeren Sinne kein Plagiat, allerdings auch kein Nachweis einer gemäß Prüfungsordnung zu erbringenden selbstständigen wissenschaftlichen und fachpraktischen Leistung (nicht bestanden, Wiederholung möglich).

Kapitel 2

Theorie neuronaler Netze

2.1 Die formalen Neuronen

In diesen Abschnitten geht es um die mathematischen Grundlagen der zu behandelnden Problemstellung. Hier können Definitionen und Sätze auftauchen, wobei die Sätze teils bewiesen werden sollten (falls einfach und für das weitere Verständnis wesentlich) oder ihre Beweise sauber zitiert werden. Beispiele:

Definition 2.1.1 (Formales Neuron)
Ein formales Neuron ist eine Funktion $\kappa : \mathbf{R}^n \to \mathbf{R}^m$*, die erzeugt wird durch die Verkettung einer sogenannten Transferfunktion* $\sigma : \mathbf{R} \to \mathbf{R}$ *mit einer sogenannten Aktivierungsfunktion* $A : \mathbf{R}^n \to \mathbf{R}$*:*

$$\begin{aligned} \kappa : \quad & \mathbf{R}^n \to \mathbf{R}^m \,, \\ & \vec{x} \mapsto \big(\sigma(A(\vec{x})), \sigma(A(\vec{x})), \,\ldots\, , \sigma(A(\vec{x}))\big) \,. \end{aligned} \tag{2.1}$$

Satz 2.1.1 (Dichtheitssatz für hyperbolische Aktivierung)
Es sei $f : \mathbf{R}^n \to \mathbf{R}$ *stetig,* $K \subset \mathbf{R}^n$ *kompakt und* $\sigma : \mathbf{R} \to \mathbf{R}$ *eine stetige Sigmoidalfunktion. Dann gibt es für alle* $\varepsilon > 0$ *Parameter*

$$\begin{aligned} & q \in \mathbf{N}, \\ & d_{kp} \in \mathbf{R}, \;\; 1 \le k \le n, \; 1 \le p \le q, \\ & \rho_p \in \mathbf{R}, \quad 1 \le p \le q, \\ & g_p \in \mathbf{R}, \quad 1 \le p \le q, \end{aligned} \tag{2.2}$$

so dass für alle $\vec{x} \in K$ gilt

$$\left| f(\vec{x}) - \sum_{p=1}^{q} g_p \sigma \left(\rho_p \prod_{k=1}^{n} (x_k - d_{kp}) \right) \right| < \varepsilon. \tag{2.3}$$

Definiert man nun

$$\sigma_\pi(\vec{x}) := \sigma(\prod_{k=1}^{n} x_k) \,, \tag{2.4}$$

so lässt sich dies auch kurz schreiben als

$$H_\sigma := \overline{span} \left\{ \sigma_\pi(\rho(\vec{x} - \vec{d})) \,:\, \vec{d} \in \mathbf{R}^n,\, \rho \in \mathbf{R} \right\} = C(\mathbf{R}^n) \,, \tag{2.5}$$

wobei $\overline{span}$ den Abschluss bezüglich der Topologie der gleichmäßigen Konvergenz auf kompakten Mengen bezeichnet.

Beweis: Vergleiche die Originalarbeiten [8, 17] oder [10, S. 163ff]. ■

Auch Zeichnungen (vgl. Abbildung 2.1) können sinnvoll sein und eingebunden werden:

Abbildung 2.1: Reales Neuron (Schematische Skizze)

2.2 Die Topologie neuronaler Strukturen

2.3 Der Ausführ-Modus

2.4 Der Lern-Modus

Um Algorithmen zu erläutern kann es sinnvoll sein Pseudocode zu verwenden. Ein Beispiel dafür ist der Pseudocode 2.4.1.

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
1: $y \Leftarrow 1$
2: **if** $n < 0$ **then**
3: $X \Leftarrow 1/x$
4: $N \Leftarrow -n$
5: **else**
6: $X \Leftarrow x$
7: $N \Leftarrow n$
8: **end if**
9: **while** $N \neq 0$ **do**
10: **if** N ist gerade **then**
11: $X \Leftarrow X \times X$
12: $N \Leftarrow N/2$
13: **else**
14: $y \Leftarrow y \times X$
15: $N \Leftarrow N - 1$
16: **end if**
17: **end while**

Pseudocode 2.4.1: Berechne $y = x^n$

Kapitel 3

Implementierung neuronaler Netze

3.1 Die Wahl der Programmiersprache: Java

Kurze Begründung für die Auswahl der Programmiersprache. Mit welchem Entwicklungstool wurde gearbeitet und warum? Oder wurde direkt mit dem JDK gearbeitet? Wenn ja, warum? **Keine** detaillierte Einführung in Java; das ist inzwischen Standard. Allerdings: Neue und spezielle Bibliotheken, Packages oder Klassen, die benutzt werden, müssen begründet und erläutert werden.

3.2 Details zum Entwurf und zur Entwicklung

Klassische Vorgehensweise bei dem Entwurf und der Entwicklung eines Anwendungsprogramms (OOA, OOD, OOP, etc.). Insbesondere sollten hier (oder – falls zu umfangreich – spätestens im Anhang) die entsprechenden Diagramme eingebunden werden.

3.3 Details zur konkreten Implementierung

Hier sollten ausgewählte Teile des Source-Codes, die für die Funktionalität des Programms fundamental sind, im Detail erläutert werden. Neben dem Aufzeigen der generellen Konzepte zur Umsetzung des mathematischen Kalküls in Programm-Code geht es hier auch um Fragen wie Effizienz, Parallelisierbarkeit, numerische Stabilität, etc..

```java
public final class HelloWorld
{
    /*
     * Ein Umlaut Test: Ä
     */
    public static void main(final String[] arg)
    {
        System.out.println("Hallo Welt!"); // Ausgabe: Hallo Welt!
    }
}
```

Quellcode 3.3.1: Ein Hallo-Welt-Programm in der Programmiersprache Java.

3.3.1 XML

Beispiel für XML-Code siehe Quelltext

```xml
<!-- Ein Kommentar in XML -->
<xs:element name="UsernameToken">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Username"/>
            <xs:element ref="Password" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Id" type="xs:ID"/>
        <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
</xs:element>
```

Quellcode 3.3.2: Beispiel für XML-Code

3.3.2 JAVA

Beispiel für Java-Code siehe Quelltext

```java
/**
 * JavaDoc
 */
public class JavaBeispiel implements garNichts {

/*
 * Das ist ein plumper Kommentar
 * der über zwei Zeilen geht
 */
    public void macheWas throws LatexException {
        for (int i = 0; i < 666; i++) {              //Schleife
    durchlaufen
            System.out.println("Mache was...");
        }
    }
}
```

Quellcode 3.3.3: Beispiel für Java-Code

3.4 Probleme bei der Implementierung

Klar!

3.5 Zeichnungen

Die folgenden Zeichnungen wurden mit den LaTeX-Zusatzpaketen pgf und tikz erstellt. Sie stellen sehr mächtige Werkzeuge zur Verfügung um Diagramme und Grafiken aller Art zu erstellen. Die Ergebnisse sind professionell und können, falls nötig, mit wenig Aufwand geändert werden. Es erfordert natürlich eine gewisse Einarbeitung, aber diese wird durch die Resultate schnell wieder aufgewogen. Eine umfangreiche Anleitung mit vielen weiteren Beispielen findet sich auf
http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf

Es folgen einige Beispiele.

3.5.1 Zustandsdiagramm

Das Zustandsdiagramm (englisch: state diagram) der UML ist eine der dreizehn Diagrammarten dieser Modellierungssprache für Software und andere Systeme. Es stellt einen endlichen Automaten in einer UML-Sonderform grafisch dar und wird benutzt, um entweder das Verhalten eines Systems oder die zulässige Nutzung der Schnittstelle eines Systems zu spezifizieren.

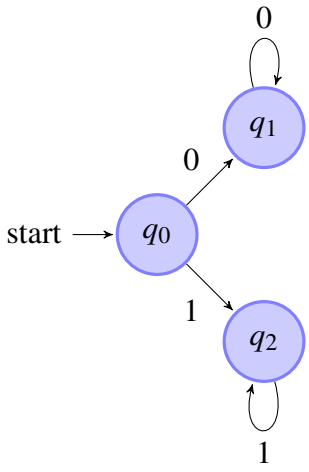

Abbildung 3.1: Zustandsdiagramm

3.5.2 Petrinetz

Ein Petri-Netz ist ein mathematisches Modell von nebenläufigen Systemen. Es ist eine formale Methode der Modellierung von Systemen bzw. Transformationsprozessen. Die ursprüngliche Form der Petri-Netze nennt man auch Bedingungs- oder Ereignisnetz. Petri-Netze wurden durch Carl Adam Petri in den 1960er Jahren definiert. Sie verallgemeinern wegen der Fähigkeit, nebenläufige Ereignisse darzustellen, die Automatentheorie.

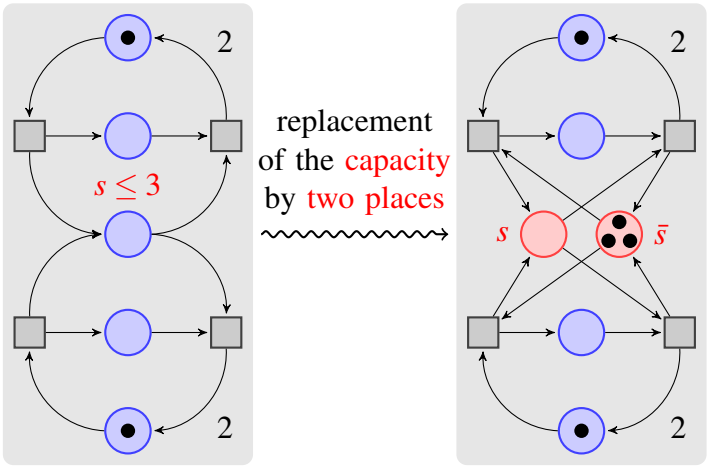

Abbildung 3.2: Petrinetz

3.5.3 Graph

Ein Graph besteht in der Graphentheorie anschaulich aus einer Menge von Punkten, zwischen denen Linien verlaufen. Die Punkte nennt man Knoten oder Ecken, die Linien nennt man meist Kanten, manchmal auch Bögen. Auf die Form der Knoten und Kanten kommt es im allgemeinen dabei nicht an. Knoten und Kanten können auch mit Namen versehen sein, dann spricht man von einem benannten Graphen.

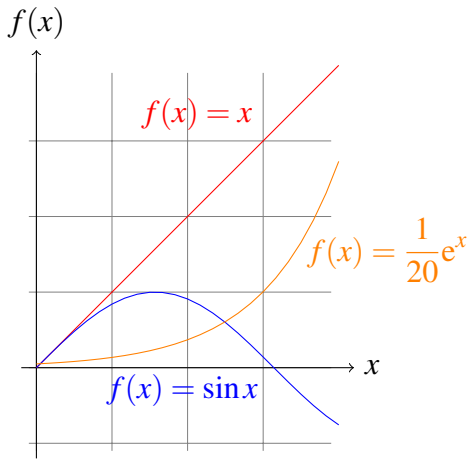

Abbildung 3.3: Graph

Kapitel 4

Anwendung neuronaler Netze

4.1 Anforderungen an Hard- und Software

- Hardware-Plattform
- Besonderheiten der Rechner-Architektur
- Prozessor-Typ bzw. -Typen
- Taktfrequenz
- Hauptspeicher-Größe
- Cache
- Swapspace (falls verwendet)
- Betriebssystem und Versionsnummer
- Entwicklungsumgebung
- Programmiersprache
- Compiler bzw. Interpreter und Versionsnummer
- Compiler- bzw. Interpreteroptionen
- verwendete Bibliotheken
- alle relevanten Implementationsdetails (insbesondere gesetzte Parameter u. ä.)
- usw., usw.

Alle diese Angaben sind wichtig, da sich die Ergebnisse sonst nur bedingt reproduzieren lassen. Nicht reproduzierbare Anwendungen sind aber wertlos!

4.2 Anwendung des Java-Programms

Hier sollte eine Beispiel-Anwendung mit dem entwickelten Programm durchgespielt werden und möglichst mit Screenshots dokumentiert werden.

Kapitel 5

Zusammenfassung und Ausblick

Noch einmal wird kurz erläutert, was in der Arbeit eigentlich gemacht wurde. Außerdem wird gesagt, was nach Meinung der Autorin bzw. des Autors noch alles gemacht werden könnte. Dadurch kann man zeigen, dass man auch etwas über den Tellerrand der eigentlichen Problemstellung hinweg gesehen hat. Insgesamt sollten dazu 1–2 Seiten genügen. Am Ende dieses Kapitels sollte die Bachelor-/Master-Arbeit etwa **40/80 Seiten** umfassen! Natürlich ist auch dies nur ein grober, aber im Auge zu behaltender Anhaltspunkt! Lieber gute 40 Seiten als redundante und langweilige 60 Seiten!

Literaturverzeichnis

[1] AUTOR, P.P.: *Titel der www-Seite*. http-Adresse. – Zugriffsdatum

[2] AUTOR, P.P.: *Titel des Buchs*. Erscheinungsort : Verlag, Erscheinungsjahr

[3] AUTOR, P.P.: Titel des Artikels. In: *Titel der Zeitschrift* Band (Jahr), S. ersteSeite–letzteSeite

[4] CHUI, C K. ; LI, X: Realization of neural networks with one hidden layer," Center for Approximation Theory. In: *Dept. of Mathematics, Texas A&M Univ., Tech. Rep* (1991), Nr. 244

[5] CHUI, Charles K. ; LI, Xin: Approximation by ridge functions and neural networks with one hidden layer. In: *Journal of Approximation Theory* 70 (1992), Nr. 2, S. 131 – 141. – URL http://www.sciencedirect.com/science/article/pii/002190459290081X. – ISSN 0021-9045

[6] DEVORE, RonaldA. ; HOWARD, Ralph ; MICCHELLI, Charles: Optimal nonlinear approximation. In: *manuscripta mathematica* 63 (1989), Nr. 4, S. 469–478. – URL http://dx.doi.org/10.1007/BF01171759. – ISSN 0025-2611

[7] HECHT-NIELSEN, R.: *Neurocomputing*. Addison-Wesley Publishing Company, 1990 (New Horizons in Technology Series). – URL http://books.google.de/books?id=6YRQAAAAMAAJ. – ISBN 9780201093551

[8] LENZE, B.: Note on a density question for neural networks. In: *Numerical Funct. Analysis and Optimiz.* 15 (1994), S. 909–913

[9] LENZE, B.: *Einführung in die Fourier-Analysis*. zweite Auflage. Berlin : Logos Verlag, 2000. – ISBN 9783931216467

[10] LENZE, B.: *Einführung in die Mathematik neuronaler Netze*. zweite Auflage. Berlin : Logos Verlag, 2003. – ISBN 9783897220218

[11] LESHNO, Moshe ; LIN, Vladimir Y. ; PINKUS, Allan ; SCHOCKEN, Shimon: *Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function*. Bd. 6. Logos Verlag, 1993. – 861–867 S

[12] MHASKAR, H.N.: Approximation properties of a multilayered feedforward artificial neural network. In: *Advances in Computational Mathematics* 1 (1993), Nr. 1, S. 61–80. – URL http://dx.doi.org/10.1007/BF02070821. – ISSN 1019-7168

[13] MHASKAR, H.N.: Neural networks for optimal approximation of smooth and analytic functions. In: *Neural Computation* 8 (1996), S. 164–177

[14] MHASKAR, H.N ; MICCHELLI, Charles A.: Approximation by superposition of sigmoidal and radial basis functions. In: *Advances in Applied Mathematics* 13 (1992), Nr. 3, S. 350 – 373. – URL http://www.sciencedirect.com/science/article/pii/019688589290016P. – ISSN 0196-8858

[15] MHASKAR, H.N ; MICCHELLI, Charles A.: Degree of Approximation by Neural and Translation Networks with a Single Hidden Layer. In: *Advances in Applied Mathematics* 16 (1995), Nr. 2, S. 151 – 183. – URL http://www.sciencedirect.com/science/article/pii/S0196885885710081. – ISSN 0196-8858

[16] MÜLLER, Berndt ; REINHARDT, Joachim ; STRICKLAND, Michael T.: *Neural Networks*. 2. Springer-Verlag Berlin Heidelberg, 1995 (Physics of Neural Networks). – ISBN 9783540602071

[17] PINKUS, Allan: TDI-Subspaces of $C(\mathbf{R}^d)$ and Some Density Problems from Neural Networks. In: *Journal of Approximation Theory* 85 (1996), Nr. 3, S. 269 – 287. – URL http://www.sciencedirect.com/science/article/pii/S0021904596900428. – ISSN 0021-9045

Anhang A

Diagramme und Tabelle

Ein oder mehrere Anhänge können, müssen aber nicht vorhanden sein. Als Faustregel gilt: Alles was den Lesefluss stört, kann in einen Anhang, also insbesondere Programmlistings (länger als eine Seite), umfangreiches Tabellen-Material, etc.. Die Listings der Programme, also der **Original-Source-Code**, sollten auf jeden Fall – außer eventuell im Anhang – auch auf CD-ROM der Bachelor-/Master-Arbeit in einem Einsteckfach beigelegt werden. Als Zugabe kann dort auch noch direkt ausführbarer Maschinen-Code für verschiedene Plattformen hinterlegt werden. Ebenfalls sollte es eine Selbstinstallationsroutine für mindestens ein gängiges Betriebssystem auf dem Datenträger geben!
Im Gegensatz zu normalen Kapiteln werden Anhänge zur besseren Unterscheidung nicht mit „1“, „2“, „3“,… durchnummeriert, sondern mit „A“, „B“, „C“,…. Ist nur ein Anhang vorhanden, kann die Nummerierung „A“ entfallen.

Am Ende des Anhangs sollte der Umfang der Bachelor-/Master-Arbeit etwa **60/100 Seiten** betragen!

Anhang B

UML-Diagramme

Hier könnten Klassen- oder UML-Diagramme stehen!

Anhang C

Quellcode

Hier könnten konkrete Teile des Java-Quellcodes stehen!

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt und mich keiner fremden Hilfe bedient sowie keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 4. Januar 2016 Stefan Kruk

Erklärung

Mir ist bekannt, dass nach § 156 StGB bzw. § 163 StGB eine falsche Versicherung an Eides Statt bzw. eine fahrlässige falsche Versicherung an Eides Statt mit Freiheitsstrafe bis zu drei Jahren bzw. bis zu einem Jahr oder mit Geldstrafe bestraft werden kann.

Dortmund, den 4. Januar 2016 Stefan Kruk