GYMNASIUM BÄUMLIHOF

WETTBEWERBSARBEIT

Orthogonale Eigenschaftsfunktionen

Ein ergänzender Ansatz künstliche neuronale Netzwerke zu trainieren

---

Autor:
Lukas Florian Münzel

Betreuungsperson:
Bernhard Pfammatter

Basel, 31. Oktober 2021

Inhaltsverzeichnis

1 Einleitung

Schon seit den 1950er Jahren faszinierte das Konzept des maschinellen Lernens Forscher und Autoren zugleich [1]. Dank exponentiell wachsenden Rechenressourcen konnten diese frühen Träume in den 1980er Jahren und dann vor allem im 21. Jahrhundert in die Realität umgesetzt werden. Moderne Methoden können heutzutage von der Empfehlung von YouTube-Videos bis zur Früherkennung von Brustkrebs eingesetzt werden [2, 3].

Tiefes Lernen (englisch: *deep learning*) bewies sich im Verlaufe der Zeit als eine der vielversprechendsten Methoden des maschinellen Lernens. Inspiriert vom menschlichen Gehirn werden hierbei neuronale Netzwerke mit mehreren Schichten von Neuronen simuliert. In der englischen Fachliteratur werden diese Schichten als *layers* bezeichnet. Die Neuronen der ersten Schicht nehmen direkt eine Eingabe, wie beispielsweise ein Bild an, gewichten diese und geben sie an die Neuronen der nächsten Schicht weiter. Diese nächsten Neuronen könnten dann beispielsweise Kanten im gegebenen Bild erkennen. Die Neuronen der darauf folgenden Schicht könnten anschliessend, basierend auf der gewichteten Summe der Ausgaben der Kanten erkennenden Neuronen, komplexere Formen wie Kreise erkennen. Dieser Prozess des Gewichtens und Weitergebens an die Neuronen der nächsten Schicht wird dann wiederholt, bis die letzte Schicht, die Ausgabeschicht erreicht, ist. Ein Neuron dieser letzten Schicht könnte dann beispielsweise die Wahrscheinlichkeit, dass das ursprünglich gegebene Bild einen Hund abbildet, angeben.

Um festzustellen, wie gut das neuronale Netzwerk zu einem gegebenen Zeitpunkt seine Aufgabe erfüllt, wird der sogenannte *Verlust* (englisch: *loss*) berechnet. Der Verlust misst, wie verschieden die Ausgaben der Neuronen der letzten Schicht von der für sie erwünschten Werte sind. Die Gewichte zwischen den verschiedenen Schichten werden anschliessend so angepasst, dass dieser Verlust minimiert wird.

Unser Ansatz ist nun allerdings auch in der Lage, Informationen der Neuronen der vorletzten Schicht in Betracht zu ziehen. Genauer subtrahieren wir ein Mass der Informationsunabhängigkeit zwischen den *Eigenschaftsfunktionen* der Neuronen der vorletzten Schicht vom Verlust. Die *Eigenschaftsfunktionen* sind die Funktionen, welche aus den Eingaben in die Neuronen der ersten Schicht die Ausgabe der Neuronen der vorletzten Schicht generieren.

So wird das neuronale Netzwerk für die Generierung von unabhängigen respektive eben zueinander *orthogonalen* Eigenschaftsfunktionen belohnt. Orthogonale Eigenschaftsfunktionen haben wegen ihrer mathematischen Eigenschaften den grossen Vorteil, dass sie die Berechnung der optimalen Wahl der Gewichte zwischen der vorletzten und der letzten Schicht stark vereinfachen.

Wie weit die Eigenschaftsfunktionen davon entfernt sind orthogonal zu sein, wird mithilfe der *Kovarianz* bestimmt. Die Berechnung dieser *Kovarianz* für das Trainieren von künstlichen neuronalen Netzwerken ist keine gänzlich neue Idee. Allerdings wurde sie von Yann LeCun, einem Pionier des maschinellen Lernens, wegen der hohen für sie benötigten Rechenleistung abgelehnt, weshalb ihr bis dato wenig Aufmerksamkeit geschenkt wurde.

Den Effekt, den die Kovarianz zwischen den Eigenschaftsfunktionen auf die Leistung neuronaler Netzwerke hat, zu verstehen und beeinflussen zu können, könnte gemeinsam mit dem Ausnutzen der mathematischen Eigenschaften orthogonaler Eigenschaftsfunktionen künstliche neuronale Netzwerke schneller bessere Leistungen erbringen lassen. Da so potenziell finanzielle Mittel eingespart und Tausende unnötige Tode verhindert werden könnten, haben wir uns dazu entschieden, diese Methode auf einen ersten Prüfstand zu stellen.

1.1 Inhalt der Arbeit

In Abschnitt 2 werden die grundlegende Funktion künstlicher neuronaler Netzwerke und die mathematischen Hintergründe, welche zum Verständnis der Bedeutung der Orthogonalität der Eigenschaftsfunktionen erforderlich sind, erörtert. In Abschnitt 3 werden die zur Orthogonalisierung der Eigenschaftsfunktionen verwendeten Daten und Methoden erklärt. Die Vor- und Nachteile dieser Orthogonalisierung sowie das Verhalten eines Masses für die Kovarianz in konventionell trainierten neuronalen Netzwerken werden in Abschnitt 4 beschrieben. Eine Zusammenfassung unserer Ergebnisse ist in Abschnitt 5 zu lesen. Abschliessend beschreibt Abschnitt 6, wie zukünftige Arbeiten auf unseren Ergebnissen aufbauen könnten und wir danken den in Abschnitt 7 genannten Personen und Institutionen, welche diese Forschungsarbeit ermöglicht haben.

2 Notwendige Vorkenntnisse

2.1 Die Funktionsweise künstlicher neuronaler Netzwerke

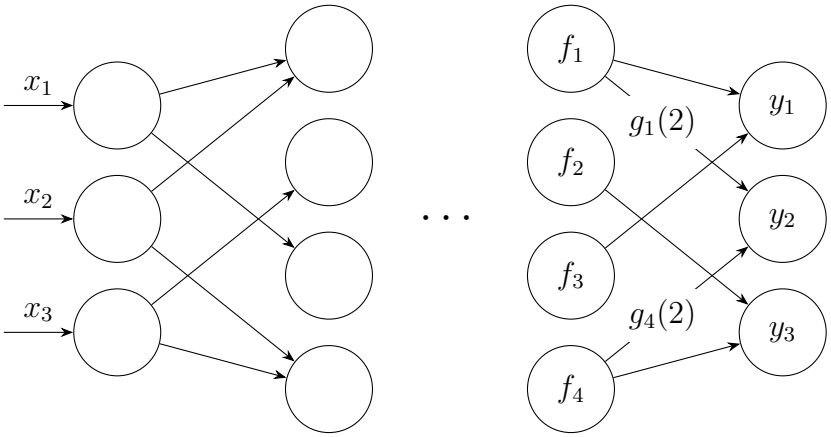

Abbildung 1: Ein künstliches neuronales Netzwerk mit Gewichten und Eigenschaftsfunktionen

Abbildung 1 veranschaulicht die Grundstruktur eines künstlichen neuronalen Netzwerkes. Die Kreise repräsentieren jeweils einzelne Neuronen. Die gezeigten Spalten von Neuronen werden jeweils als *Schicht* bezeichnet. Die *Eigenschaftsfunktionen* sind die Funktionen, welche die Ausgaben der vorletzten Neuronenschicht mit den Eingaben in die erste Schicht als Argument berechnen. In der Abbildungen und im Folgenden werden die Bezeichnungen $f_1, f_2, f_3 \cdots f_k$ für diese Funktionen verwendet. Die *Gewichte*, mit denen die Ausgabe der Neuronen der vorletzten Schicht gewichtet werden, sind in der Abbildung als $g_1(y), g_2(y) \cdots g_k(y)$ bezeichnet. Hierbei verbindet das Gewicht $g_i(j)$ die i-te Eigenschaftsfunktionen mit dem j-ten Ausgabeneuron. Zwischen allen anderen aufeinanderfolgenden Schichten befinden sich ebenfalls Gewichte. Mit allen Gewichten als Parameter und den Eingaben $x_1, x_2, \cdots x_n$ als Argumenten berechnet das neuronale Netzwerk die Ausgaben $y_1, y_2 \cdots y_m$.

So ist beispielsweise y_2 definiert als die Summe der Ausgaben der Eigenschaftsfunktionen skaliert mit den jeweiligen Gewichten. Demzufolge ist $y_2 = \sum_{i=1}^{k} g_i(2) f_i(x_1, x_2, \cdots)$. Um y_2 zum Beispiel in eine Wahrscheinlichkeit umzuwandeln, wird y_2 anschliessend noch normalisiert. Allerdings werden nicht nur alle Ausgaben y_i der Neuronen der letzten Ebene mit diesem Verfahren berechnet, denn der noch nicht normalisierte Ausgabewert aller Neuronen nach der ersten Schicht ist ebenfalls durch die gewichtete Summe der Ausgaben der Neuronen der vorhergehenden Schicht gegeben.

Nun kann aus einer Eingabe und den Gewichten eine Ausgabe generiert werden. Mit zufälligen Gewichten hat diese Ausgabe allerdings noch keinerlei Verbindung mit dem, wofür wir diese gerne verwenden würden. Deshalb berechnet das neuronale Netzwerk Ausgaben für tausende von Beispielen und der sogenannte *Verlust* (englisch: *loss*) wird berechnet. Der *Verlust* misst, wie stark sich die Ausgabe des neuronalen Netzwerkes von der gewünschten Ausgabe in den gegebenen Beispielen unterscheidet. Eine für Klassifikationsprobleme übliche Definition des *Verlustes* ist die Kreuzentropie. Die Kreuzentropie kann für unseren Anwendungsfall vereinfacht folgendermassen definiert werden:

$$L_{Kreuzentropie} = -\frac{1}{|S|} \sum_{s \in S} \log P(s)$$

Hierbei sei S die Menge der richtigen Lösungen zu den gegebenen Beispielen und $P(s)$ die vom neuronalen Netzwerk angegeben Wahrscheinlichkeit, dass das Beispiel, welches als Klasse s klassifiziert werden sollte, ein Beispiel der Klasse s ist.

Stochastic gradient descent (deutsch: stochastischer Gradientenabstieg) oder komplexere Verfahren basierend auf ähnlichen Ideen [4] berechnen anschliessend, wie die Gewichte zur Minimierung des Verlustes angepasst werden müssen. Dieser Prozess ist iterativ, zum Finden einer immer besseren Lösung werden die Gewichte also schrittweise angepasst. Die Grösse dieser Schritte wird als *Lernrate* (englisch: *learning rate*) und eine Iteration über alle Eingabedaten als *Epoche* bezeichnet. Der Optimierungsprozess wird in diesem Zusammenhang oft *Training* genannt.

2.2 Was sind Vektoren

2.2.1 Einführung

Im Rahmen des gymnasialen Mathematikunterrichts werden Vektoren meist als Pfeile im zwei- oder dreidimensionalen Raum vorgestellt. Dies lässt sich verhältnismässig einfach vorstellen und kann beispielsweise in der Physik sehr gut angewandt werden. Wenn ein Tennisball durch die Luft fliegt, hat seine Geschwindigkeit eben eine Richtung und einen Betrag.

Es stellt sich nun aber heraus, dass die Erkenntnisse, die sich aus dem Studium von Vektoren ergaben, für mehr als lediglich Pfeile im Raum Anwendung finden. Deshalb wurde das Konzept von Vektoren generalisiert. So wurden bestimmte Eigenschaften, die für alle Vektoren gelten sollen, sogenannte *Axiome*, definiert. Wenn man in der Lage ist zu zeigen, dass ein mathematisches

Objekt diese Axiome erfüllt, so können alle für diese generalisierten Vektoren entwickelten Erkenntnisse auch auf dieses neue mathematische Objekt angewandt werden [5].

2.2.2 Generalisierte Definition eines Vektors

Definition 1 (Menge)**.** Eine *Menge* ist eine Zusammenfassung einzelner Objekte, welche als Elemente bezeichnet werden. So sind $\mathbb{N} = \{1, 2, 3, \cdots\}$ und $\mathbb{Z} = \{\cdots, -2, -1, 0, 1, \cdots\}$ Mengen von Zahlen. Folglich gilt beispielsweise, dass $1 \in \mathbb{N}$, sprich 1 ist ein Element der Menge $\mathbb{N}$ [6].

Definition 2 (Kardinalität)**.** Die Anzahl der Elemente einer Menge M wird als *Kardinalität* $|M|$ dieser Menge bezeichnet.

Definition 3 (Verknüpfung)**.** Eine *Verknüpfung* kann zwei Elemente zweier verschiedener Mengen oder derselben Menge auf ein Element einer Menge abbilden. Konkret bildet beispielsweise die Verknüpfung $+$ zwei natürliche Zahlen auf die natürlichen Zahlen ab, während die Verknüpfung $-$ zwei natürliche Zahlen auf die ganzen Zahlen abbildet. Letztere Verknüpfung würde als $-\colon \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ niedergeschrieben.

Definition 4 (Algebraische Struktur)**.** Eine Menge zusammen mit einer oder mehreren Verknüpfungen wird als *algebraische Struktur* bezeichnet. Als Beispiel ist $(\mathbb{R}, +, \cdot)$ die algebraische Struktur der reellen Zahlen zusammen mit den Verknüpfungen der Addition und der Multiplikation.

Definition 5 (Körper)**.** Ein *Körper* ist eine algebraische Struktur, für welche Multiplikation und Addition definiert sind und gewisse Bedingungen bezüglich der Eigenschaften dieser Verknüpfungen gelten. Von Relevanz ist hierbei vor allem, dass $(\mathbb{R}, +, \cdot)$ diese Bedingungen erfüllt und folglich einen Körper bildet.

Sei V eine Menge, K ein Körper $\mathbb{R}$, $\oplus\colon V \times V \to V$ eine Verknüpfung namens *Vektoraddition* und $\odot\colon K \times V \to V$ eine Verknüpfung namens *Skalarmultiplikation*

Nun wird die algebraische Struktur $(V, \oplus, \odot)$ als Vektorraum über dem *Körper* K, $v \in V$ als Vektor und $k \in K$ als Skalar bezeichnet, wenn folgende Axiome für alle $\vec{u}, \vec{v}, \vec{w} \in V$ und $a, b \in K$ gelten [7]:

1. $\vec{u} \oplus (\vec{v} \oplus \vec{w}) = (\vec{u} \oplus \vec{v}) \oplus \vec{w}$
2. Es existiert ein Vektor $\vec{0}$, sodass $\vec{v} \oplus \vec{0} = \vec{0}$

3. Für jeden Vektor $\vec{v} \in V$ existiert ein Vektor $-\vec{v} \in V$, sodass $\vec{v} \oplus (-\vec{v}) = \vec{0}$
4. $\vec{u} \oplus \vec{v} = \vec{v} \oplus \vec{w}$
5. $a \odot (\vec{u} + \vec{v}) = (a \odot \vec{u}) \oplus (a \odot \vec{v})$
6. $(a + b) \odot \vec{v} = (a \odot \vec{v}) \oplus (b \odot \vec{v})$
7. $(a \cdot b) \odot \vec{v} = a \odot (b \odot \vec{v})$
8. Es existieren ein Element $1_k \in K$, sodass $1_k \odot \vec{v} = \vec{v}$

Es kann einfach gezeigt werden, dass diese Axiome beispielsweise für Vektoren in $\mathbb{R}^3$, respektive präziser ausgedrückt für Vektoren im Vektorraum $\mathbb{R}^3$ über den reellen Zahlen, gelten.

2.2.3 Funktionen als Vektoren betrachten

Sei F die Menge aller Funktionen, sodass jedes $f \in F$ die reellen Zahlen zwischen null und eins auf die reellen Zahlen abbildet. Für den Vektorraum $(F, \oplus, \odot)$ über den reellen Zahlen wird das Skalarprodukt und die Vektoraddition definiert als [8]:

$$(f \oplus g)(x) = f(x) + g(x)$$
$$(k \odot g)(x) = kf(x)$$

Hierbei seien $f(x)$ und $g(x)$ jeweils Elemente der Menge F und $k \in \mathbb{R}$. Erneut können die acht Axiome gezeigt werden. So gilt beispielsweise für jedes $f(x) \in F$, dass $(f \oplus g)(x) = f(x)$, wenn $g(x) = 0$, womit das zweite Axiom bewiesen ist. Zudem kann das sogenannte *innere Produkt* $\langle \cdot, \cdot \rangle \colon F \times F \to \mathbb{R}$ definiert werden. Dieses innere Produkt stellt das Äquivalent zum Skalarprodukt dar. Einige Axiome für das Skalarprodukt gelten auch für das innere Produkt, es sind beispielsweise beide kommutativ. Wenn erneut $f, g \in F$ gilt, ist das innere Produkt $\langle f, g \rangle$ definiert als:

$$\langle f, g \rangle = \int_0^1 f(x)g(x)dx$$

2.3 Projektionen auf eine lineare Hülle

Definition 6 (Lineare Hülle)**.** Die *lineare Hülle* H von den Vektoren $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$ ist definiert als die Menge aller Vektoren, welche als $k_1\vec{v_1} + k_2\vec{v_2} + \cdots + k_n\vec{v_n}$ dargestellt werden können, wobei k_i jeweils ein Skalar sei. Im Folgenden sind diese Skalare stets reelle Zahlen. Abstrakter ausgedrückt ist also

$$H(\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}) = \left\{ \vec{h} = \sum_{i=1}^{n} k_i\vec{v_i} \;\middle|\; k_i \in \mathbb{R} \right\}$$

Definition 7 (Orthogonal)**.** Zwei Vektoren sind genau dann orthogonal zueinander, wenn ihr Skalarprodukt respektive ihr inneres Produkt null ist. In $\mathbb{R}^2$ ist dies gleichbedeutend damit, dass die beiden Vektoren in einem rechten Winkel zueinander stehen.

Für unsere Zwecke kann die *Projektion* des Vektors $\vec{w}$ auf eine lineare Hülle H als folgendes Problem verstanden werden:

$$\operatorname{proj}_H \vec{w} = \operatorname*{argmin}_{\vec{h} \in H} ||\vec{h} - \vec{w}||^2$$

Es wird folglich der Vektor der linearen Hülle mit minimaler Distanz zu $\vec{w}$ gesucht. Wenn der Vektor $\vec{w}$ Element der linearen Hülle ist, ist dementsprechend seine Projektion auf diese lineare Hülle $\vec{w}$ selbst.

Eine des Weiteren sehr wichtige Eigenschaft der Projektion ist, dass wenn die Vektoren $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$ orthogonal zueinander sind, die Projektion auf die lineare Hülle einfach in Projektionen auf die einzelnen Vektoren $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$ aufgeteilt werden kann.

Die Projektion eines Vektors $\vec{w}$ auf die lineare Hülle der orthogonalen Vektoren $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$ ist nämlich

$$\operatorname{proj}_H \vec{w} = \sum_{i=1}^{n} \operatorname{proj}_{\vec{v_i}} \vec{w} = \operatorname{proj}_{\vec{v_1}} \vec{w} \operatorname{proj}_{\vec{v_2}} \vec{w} + \cdots$$

Ein Beweis dieser Gleichung für Vektoren in $\mathbb{R}^n$ ist im Anhang A.3 zu finden.

Zudem ist dies auch in den Abbildungen 2 und 3 zu erkennen. Die grünen Vektoren stellen jeweils $v_i \operatorname{proj}_{\vec{v_i}} \vec{w}$ dar. Da $\vec{w}$, hier als der schwarze Punkt dargestellt, in beiden Abbildungen Element der linearen Hülle von $\vec{v_1}$ und $\vec{v_2}$ ist, sollte seine Projektion auf diese lineare Hülle $\vec{w}$ selbst sein. Es ist also erkenntlich, dass die Summe der Projektionen auf $\vec{v_1}$ und $\vec{v_2}$, nicht zwangsläufig gleich der Projektion auf die gesamte lineare Hülle ist, wenn die beiden Vektoren nicht orthogonal zueinander sind.

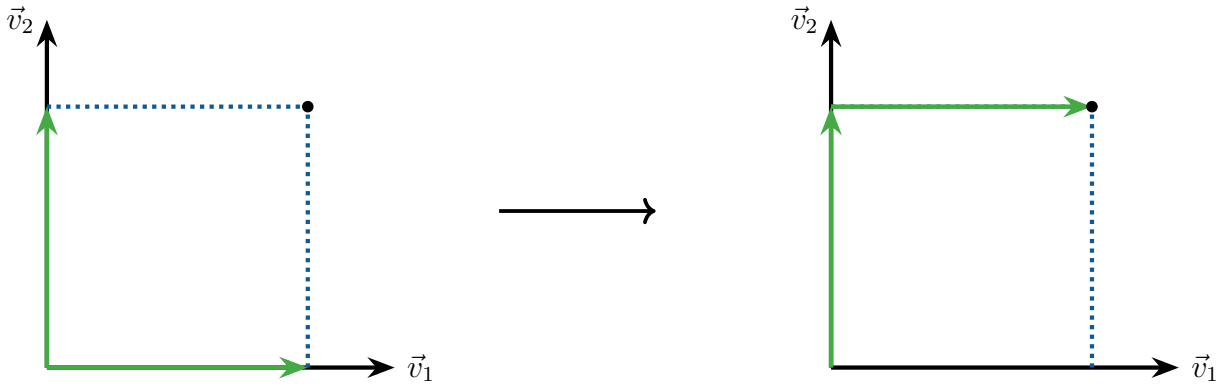

Abbildung 2: Projektion auf die lineare Hülle der orthogonalen Vektoren $\vec{v_1}$ und $\vec{v_2}$

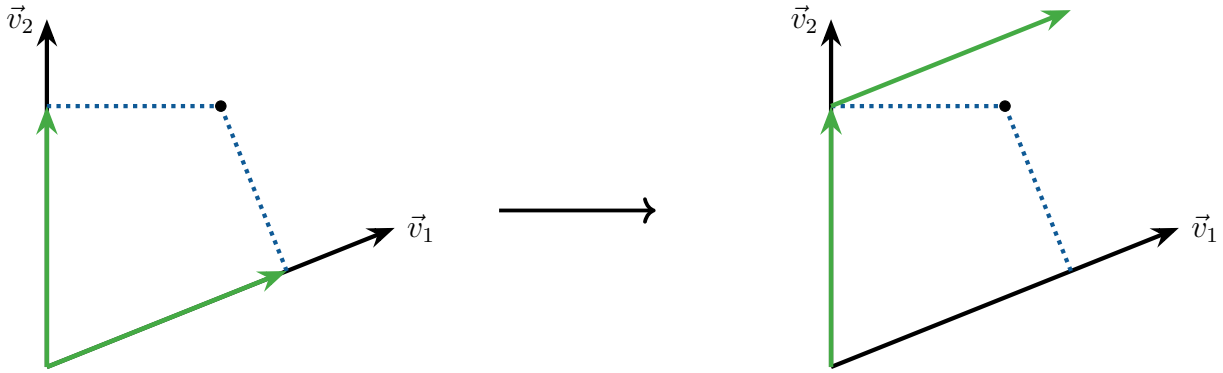

Abbildung 3: Projektion auf die lineare Hülle der nicht orthogonalen Vektoren $\vec{v_1}$ und $\vec{v_2}$

2.4 Projektion auf die Zielfunktion

Die Zielfunktion $T(x_1, x_2, \cdots, x_n)$ sei definiert als die Funktion, welche die Argumente $x_1, x_2 \cdots x_n$ auf den jeweils gewünschten Wert von y_j für ein spezifisches j abbildet. Für gegebene Eigenschaftsfunktionen ist es nun unser Ziel, die Gewichte zwischen den Eigenschaftsfunktionen und dem i-ten Ausgabeneuron so anzupassen, dass $y_j(x_1, x_2, \cdots, x_n) = \sum_i g_i f_i(x_1, x_2, \cdots, x_n)$ möglichst der Zielfunktion entspricht. Die Zielfunktion und die Eigenschaftsfunktionen können nun wie in Teilabschnitt 2.2.3 beschrieben als Vektoren interpretiert werden. Den Verlust zu minimieren ist also gleichbedeutend damit, die Distanz zwischen der Zielfunktion $\vec{T}(x_1, x_2, \cdots, x_n)$ und den Eigenschaftsfunktionen skaliert mit den jeweiligen Gewichten zu minimieren. Wir

suchen folglich die $g_1, g_2, \cdots g_k$, welche den folgenden Ausdruck minimieren:

$$||\vec{T} - \sum_i g_i \vec{f_i}||^2$$

Dieses Problem wurde bereits in Teilabschnitt 2.3 als die Projektion von $\vec{T}$ auf die lineare Hülle der Vektoren $\vec{f_1}, \vec{f_2}, \cdots, \vec{f_k}$ vorgestellt. Der rote Punkt in der Abbildung 4 soll diese Projektion repräsentieren.

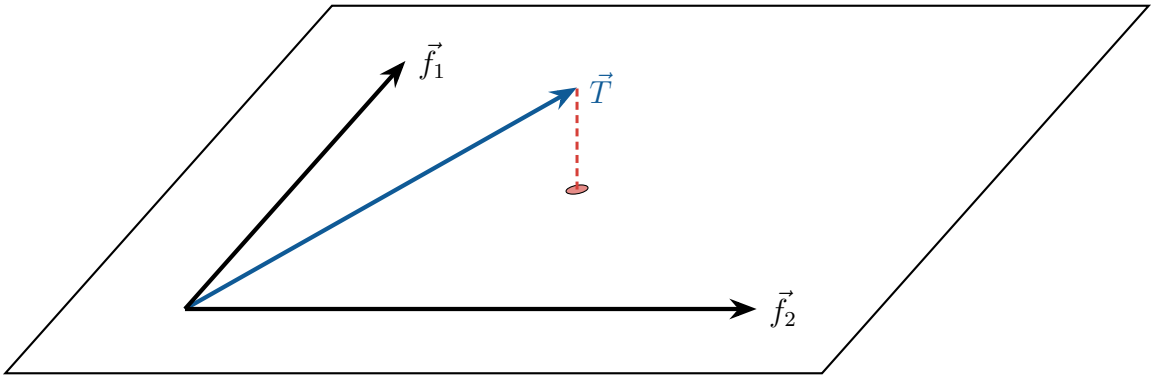

Abbildung 4: Projektion von $\vec{T}$ auf die lineare Hülle der Vektoren $\vec{f_1}$ und $\vec{f_2}$

Konventionelle Methoden, welche entsprechend mit nicht orthogonalen Eigenschaftsfunktionen operieren, approximieren diese Projektion iterativ, siehe Teilabschnitt 2.1. Dieses Problem könnte allerdings, wie in Teilabschnitt 2.3 beschrieben, für orthogonale Eigenschaftsfunktionen in mehrere bedeutend einfacher zu lösende Probleme aufgeteilt werden. Wenn die Eigenschaftsfunktionen also zueinander orthogonal wären, so könnte die optimale Gewichtskonfiguration zwischen der vorletzten und der letzten Neuronenschicht effizienter gefunden werden, als konventionelle Methoden eine Approximation für dieses Optimum finden.

3 Methoden

3.1 Generierung der Eingabedaten

Das benutzte Datenset besteht aus Punkten in Clustern von mehrdimensionalen Normalverteilungen. Jedem Cluster ist eine Farbe zugeordnet, welche das neuronale Netzwerk basierend auf den Koordinaten eines Punktes versucht vorherzusagen. Die Farben werden in diesem Kontext als Klassen bezeichnet. Man spricht bei solchen, mit einer vorgegebenen Lösung verbundenen Klassifikationsproblemen von *überwachtem Lernen*, respektive in der englischen Fachliteratur von *supervised learning*.

Ein so gewähltes Datenset hat den Vorteil, dass es wenig Rechenleistung benötigt und dennoch repräsentativ für viele komplexere Probleme ist. Zudem kann die Komplexität des Problems beliebig angepasst werden, da wir die Eingabedaten selbst generieren. Ein Datenpunkt dieser Eingabedaten mit den Koordinaten der Position im Raum und der vorgegebenen Lösung wird als *Beispiel* (englisch: *sample*) bezeichnet.

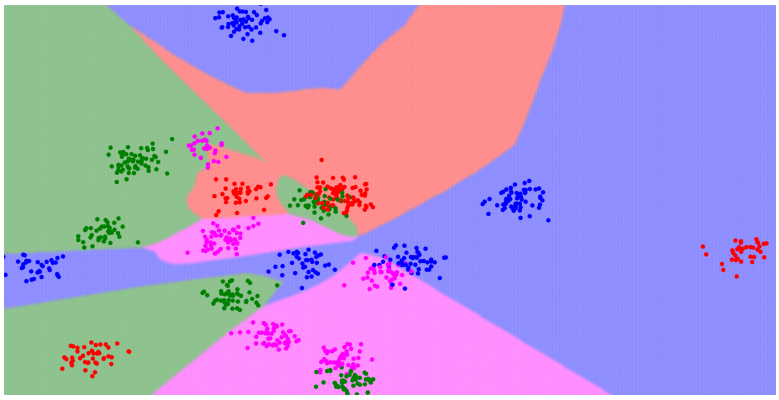

Abbildung 5: Generierte Eingabedaten mit zwei Dimensionen, vier Klassen und fünf Clustern pro Klasse

In Abbildung 5 repräsentiert die Farbe der Punkte eines Clusters die mit diesen Punkten assoziierte Klasse, während die blassere Hintergrundfarbe die von einem neuronalen Netzwerk für den jeweiligen Punkt als am wahrscheinlichsten eingeschätzte Farbe darstellt.

3.2 Messen der Orthogonalität

Orthogonale Eigenschaftsfunktionen sind wünschenswert, da sie das Finden der optimalen Gewichte zwischen der vorletzten und der letzten Neuronenschicht bedeutend einfacher machen würden. Da wir generell nicht in der Lage sind, orthogonale Eigenschaftsfunktionen zu generieren, wollen wir nun messen, wie weit die Eigenschaftsfunktionen davon entfernt sind, orthogonal zu sein.

Zwischen zwei Vektoren im euklidischen Raum ist uns dazu das Skalarprodukt bekannt, denn das Skalarprodukt zwischen $\vec{v}$ und $\vec{w}$ ist definiert als:

$$\vec{v} \cdot \vec{w} = \sum_i v_i \cdot w_i = |\vec{v}|\,|\vec{w}|\,\cos(\varphi)$$

Hierbei ist φ der Winkel zwischen $\vec{v}$ und $\vec{w}$. Je weiter die Vektoren davon entfernt sind, orthogonal zu sein, desto höher ist das Skalarprodukt.

Nun können die Eigenschaftsfunktionen jeweils wie in Teilabschnitt 2.2.3 beschrieben als Vektoren interpretiert werden und das Äquivalent zum Skalarprodukt, das innere Produkt kann verwendet werden. Dabei eröffnet sich das Problem, dass sich der Definitionsbereich der Eigenschaftsfunktionen sowohl im Positiven als auch im Negativen bis ins Unendliche erstreckt. So wäre es beispielsweise sinnvoll, wenn das Produkt der Eigenschaftsfunktionen am Punkt $P(100|100)$ nur wenig gewichtet werden würde, wenn 99 % der Eingabedaten weniger als drei Einheiten vom Ursprung entfernt sind. Um diese Probleme zu lösen, können wir anstelle des Integrals des Produktes der beiden Eigenschaftsfunktionen deshalb den Erwartungswert ihres Produktes verwenden. Formeller ausgedrückt sind wir also an dem Wert des folgenden Ausdrucks interessiert:

$$\mathbb{E}[f(x_1, x_2, \cdot, x_n) \cdot g(x_1, x_2, \cdot, x_n)]$$

Hierbei werden $x_1, x_2, \cdots, x_n$ aus der Wahrscheinlichkeitsverteilung der Punkte der Eingabedaten gezogen. So wird das Produkt der beiden Eigenschaftsfunktionen proportional zur Wahrscheinlichkeit, dass sich der Punkt, an dem sie evaluiert werden, überhaupt in den Eingabedaten befindet, gewichtet.

Wenn wir nun noch den Erwartungswert der jeweiligen Eigenschaftsfunktion von ihr abziehen, bevor wir sie mit der anderen Eigenschaftsfunktion multiplizieren, resultiert die sogenannte *Kovarianz*. Diese ist ein in der Statistik bekanntes Mass für die Korrelation zwischen zwei Zufallsvariablen, in diesem Fall $f_i(x_1, x_2, \cdots, x_n)$ und $f_j(x_1, x_2, \cdots, x_n)$. Die Kovarianz $k_{i,j}$

zwischen f_i und f_j wird folglich berechnet als:

$$k_{i,j} = \mathbb{E}[(f_i - \mathbb{E}[f_i])(f_j - \mathbb{E}[f_j])]$$

Sei $\mathbf{x}$ die Menge aller Koordinaten der Punkte der Eingabedaten und $\mathbf{x_s} = x_1, x_2, \cdots x_n$ die Koordinaten des s-te Punkt der Eingabedaten. Daraus ergibt sich:

$$\mathbb{E}[f_i] \approx \frac{1}{|\mathbf{x}|} \sum_{s=1}^{|\mathbf{x}|} f_i(\mathbf{x_s})$$

$$k_{i,j} \approx \frac{1}{|\mathbf{x}|} \sum_{s=1}^{|\mathbf{x}|} [f_i(\mathbf{x_s}) - \mathbb{E}[f_i]] \cdot [f_j(\mathbf{x_s}) - \mathbb{E}[f_j]]$$

Wenn nun in diesem oder den folgenden Kapiteln f_i und f_j als zueinander orthogonal bezeichnet werden, so ist damit gemeint, dass $k_{i,j} = 0$ ist. Je weiter die beiden Eigenschaftsfunktionen unter diese Definition davon entfernt sind, orthogonal zu sein, desto höher ist also der Betrag der Kovarianz zwischen ihnen.

Nun wollen wir ein sinnvolles Mass dafür definieren, wie nah alle Paare zweier unterschiedlicher Eigenschaftsfunktionen dazu sind, orthogonal zu sein. Wir schlagen zu diesem Zweck folgendes Mass vor:

$$\frac{\sum_i k_{i,i}^2}{\sum_{i,j} k_{i,j}^2}$$

Diese Definition hat den Vorteil, dass alle resultierende Werte reelle Zahlen zwischen null und eins inklusive sind. Wenn dieses *Kovarianzmass* einen Wert von eins hat, sind alle Paare zweier verschiedenen Eigenschaftsfunktionen orthogonal. Je weiter sie davon entfernt sind, orthogonal zu sein, desto näher ist das Mass an null. Für einen Teiler von null definieren wir das Kovarianzmass als null. Der grosse Nachteil dieser Methode ist ihre Laufzeit von $\mathcal{O}(|\mathbf{x}|k^2)$, wobei k die Anzahl der Eigenschaftsfunktionen darstellt. Dies ist der Fall, da k^2 Paare der k Eigenschaftsfunktionen existieren und pro Beispiel für jedes dieser Paare eine konstante Anzahl Operationen ausgeführt werden muss.

3.3 Kovarianzmass und Genauigkeit am Ende regulärer Trainingsprozesse messen

Um herauszufinden, was das Kovarianzmass und die Genauigkeit beeinflusst, wurden beide am Ende normaler Trainingsprozesse gemessen. Die *Genauigkeit* (englisch: *accuracy*) ist hierbei definiert als der Quotient zwischen der Anzahl Beispiele, für welche das neuronale Netzwerk die richtige Klasse vorhersagte, und der Gesamtanzahl der Beispiele.

Zu diesem Zweck wurden für jeden erhobenen Datenpunkt hunderte von neuronalen Netzwerken für eine konstante Anzahl Epochen trainiert. Die neuronalen Netzwerke verschiedener Datenpunkte unterschieden sich jeweils nur in einem Parameter, wie z.B. der Anzahl der Eigenschaftsfunktionen, von einem Basisfall. Da die Daten, wie in Teilabschnitt 3.1 erwähnt, künstlich generiert wurden, konnte zudem der Einfluss der Anzahl Dimensionen, der Anzahl Klassen sowie der Anzahl Cluster pro Klasse in Erfahrung gebracht werden.

3.4 Das Kovarianzmass vom Verlust subtrahieren

Das Kovarianzmass gibt an, wie nah die Eigenschaftsfunktionen daran sind, orthogonal zu sein. Da das neuronale Netzwerk, respektive etwas spezifischer *Stochastic gradient descent*, versucht, den Verlust zu minimieren, kann nun einfach das Kovarianzmass vom Verlust subtrahiert werden. Dadurch wird das Kovarianzmass maximiert und die Eigenschaftsfunktionen werden dementsprechend so verändert, dass sie orthogonaler zueinander sind.

Wie gut das neuronale Netzwerk bezüglich des Kovarianzmasses und der Genauigkeit abschneidet, wurde für verschieden Gewichtungen des Maximierens des Kovarianzmasses am Ende jeder Epoche gemessen. Erneut wurden hunderte von neuronalen Netzwerken trainiert, damit statistisch relevante Daten erhoben werden konnten.

Um diese Gewichtung zu quantifizieren, definieren wir die Kovarianzmassgewichtung w, $0 \leq w \leq 1$. Je höher diese Kovarianzmassgewichtung w, desto wichtiger ist das Kovarianzmass im Verhältnis zum regulären Verlust für die Berechnung des Gesamtverlustes.

Der Gesamtverlust wird dementsprechend definiert als:

$$L = (1 - w)L_{Kreuzentropie} - w\frac{\sum_i k_{i,i}^2}{\sum_{i,j} k_{i,j}^2}$$

4 Resultate und Diskussion

4.1 Kovarianzmass und Genauigkeit am Ende regulärer Trainingsprozesse messen

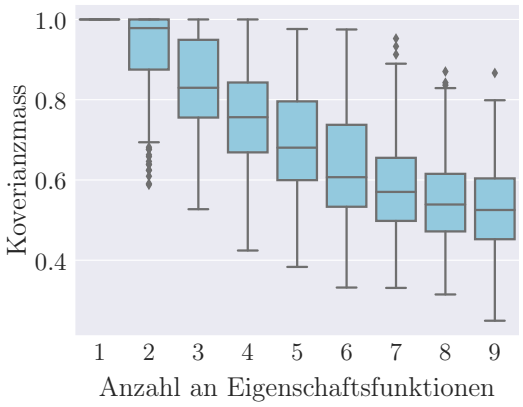

Abbildung 6: Das Kovarianzmass für verschiedene Anzahl an Eigenschaftsfunktionen

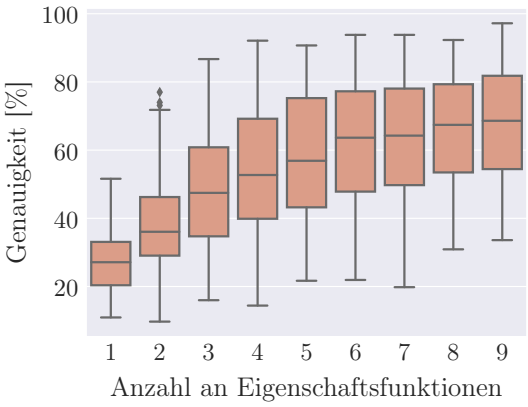

Abbildung 7: Die Genauigkeit für verschiedene Anzahl an Eigenschaftsfunktionen

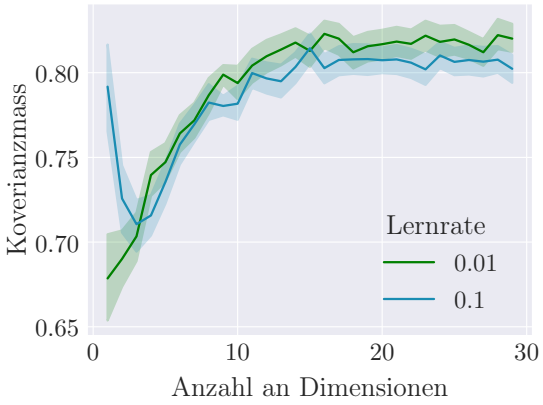

Abbildung 8: Das Kovarianzmass für verschiedene Anzahl an Dimensionen

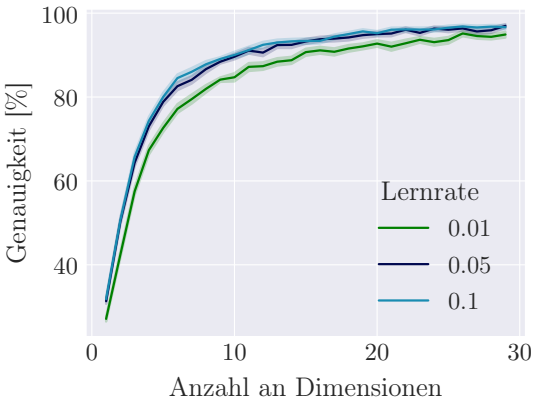

Abbildung 9: Die Genauigkeit für verschiedene Anzahl an Dimensionen

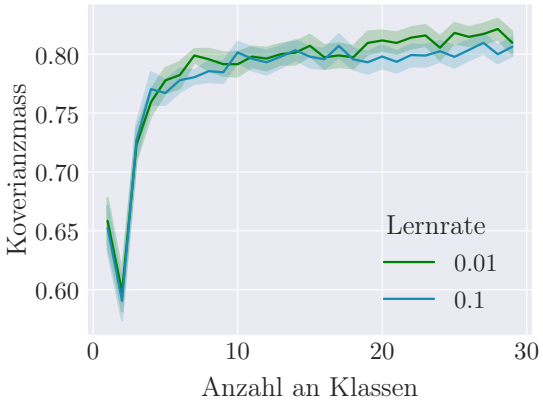

Abbildung 10: Das Kovarianzmass für verschiedene Anzahl an Klassen

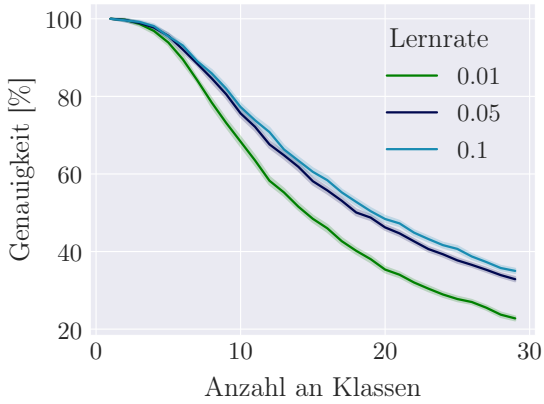

Abbildung 11: Die Genauigkeit für verschiedene Anzahl an Klassen

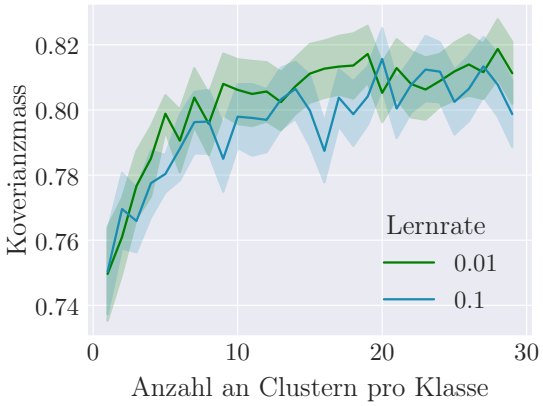

Abbildung 12: Das Kovarianzmass für verschiedene Anzahl an Clustern pro Klasse

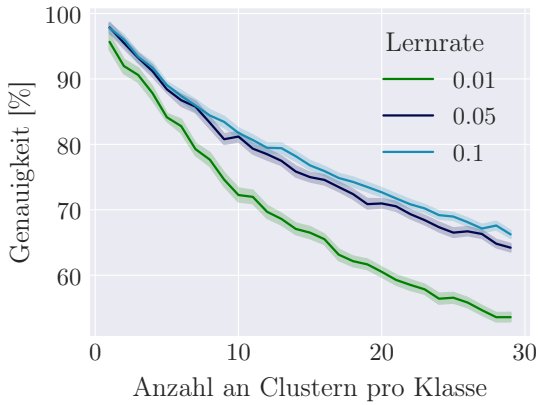

Abbildung 13: Die Genauigkeit für verschiedene Anzahl an Clustern pro Klasse

Die Abbildungen 6, 8, 10 und 12 zeigen klar eine positive Korrelation zwischen der Komplexität des den neuronalen Netzwerken gegebenen Problems und dem Kovarianzmass. Je komplexer das Problem, desto höher das Kovarianzmass. Zudem zeigt sich in Abbildungen 7, 11 und 13, dass diese schwierigere Probleme mit geringerer Genauigkeit gelöst werden, was konsistent mit unserer Intuition ist. Im Anhang A.2 wird diskutiert, weshalb dies bei Abbildung 9 nicht der Fall sein könnte. Unsere Hypothese für diese positive Korellation ist, dass bei komplexeren Problemen weniger Information zwischen den Eigenschaftsfunktionen redundant sein darf. Dies hat zur Konsequenz, dass die Eigenschaftsfunktionen orthogonaler zueinander sind.

4.2 Das Kovarianzmass vom Verlust subtrahieren

Da *Stochastic gradient descent* versucht, die Gewichte zur Minimierung des Verlustes anzupassen, nimmt das vom Verlust subtrahierte Kovarianzmass zu, was in Abbildung 15 erkannt werden kann. Eine Kovarianzmassgewichtung von zehn Prozent resultierte nach 40 Epochen in einem Kovarianzmass von mehr als 99%, im Vergleich zu einem Kovarianzmass von etwas mehr als 80% bei regulärem Training. Mit regulärem Training ist hierbei das Training mit der üblichen Definition des Verlustes, hier also der Kreuzentropie, und dementsprechend einer Kovarianzmassgewichtung von null gemeint.

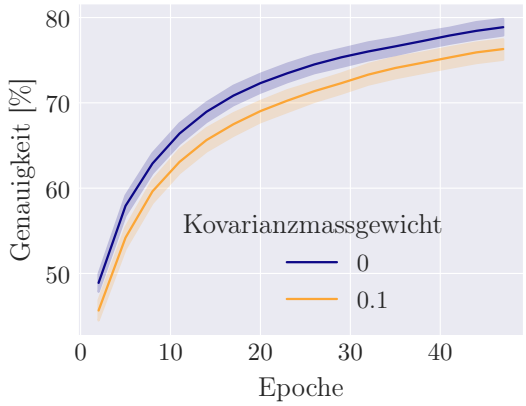

Abbildung 14: Entwicklung der Genauigkeit während des Trainingsprozesses bei einer Lernrate von 0.05

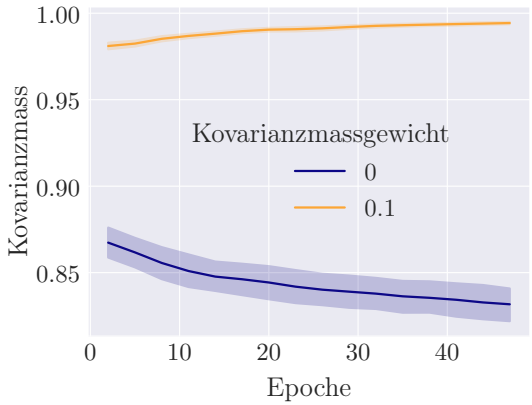

Abbildung 15: Entwicklung des Kovarianzmasses während des Trainingsprozesses bei einer Lernrate von 0.05

Allerdings mussten wir bei der gewählten Kovarianzmassgewichtung für die orthogonaleren Eigenschaftsfunktionen, wie in Abbildung 14 ersichtlich, Genauigkeit einbüssen. Alle in dieser Arbeit gezeigten Intervalle sind 95% Konvidenzintervalle, womit dieses Ergebnis statistisch signifikant ist. Unsere Erklärung für dieses Verhalten ist, dass *Stochastic gradient descent* mit höherem Kovarianzmass einen Kompromiss zwischen Orthogonalität und Eigenschaftsfunktionen, mit welchen die Kreuzentropie minimiert werden kann, eingehen muss und dementsprechend die Genauigkeit sinkt. Dies heisst aber nicht zwangsläufig, dass unsere Definition des Verlustes mit einer Kovarianzmassgewichtung von zehn Prozent strikt schlechtere Resultate produziert, denn mit den in Abschnitt 2 diskutierten mathematischen Methoden könnte in einem nächsten Schritt die optimale Gewichtskonfiguration zwischen der letzten und der vorletzten Schicht berechnet werden.

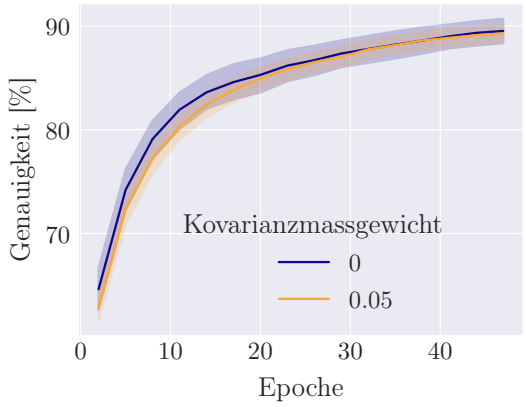

Abbildung 16: Entwicklung der Genauigkeit während des Trainingsprozesses bei einer Lernrate von 0.1

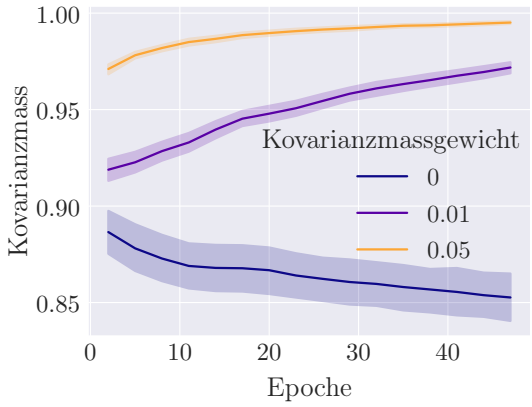

Abbildung 17: Entwicklung des Kovarianzmasses während des Trainingsprozesses bei einer Lernrate von 0.1

Wie in Abbildungen 16, 17, 18 und 19 gezeigt, kann jedoch mit einer besseren Wahl der Kovarianzmassgewichtung dieselbe Genauigkeit wie in regulärem Training mit nahezu orthogonalen Eigenschaftsfunktionen erreicht werden. Erneut könnte allerdings, da die Eigenschaftsfunktionen nahezu orthogonal sind, die optimale Gewichtskonfiguration zwischen der vorletzten und der letzten Neuronenschicht gefunden werden. Somit ist es durchaus möglich, dass unsere Methode nach diesem Schritt eine höhere Genauigkeit als Methoden mit einer konventionellen Definition des Verlustes erreichen würde. Zudem könnte eine neue Klasse auch mit wenig Rechenaufwand für die gegebenen Eigenschaftsfunktionen optimal klassifiziert werden.

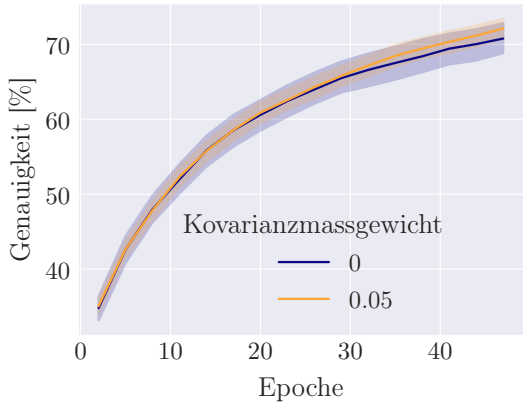

Abbildung 18: Entwicklung der Genauigkeit während des Trainingsprozesses bei einer Lernrate von 0.01

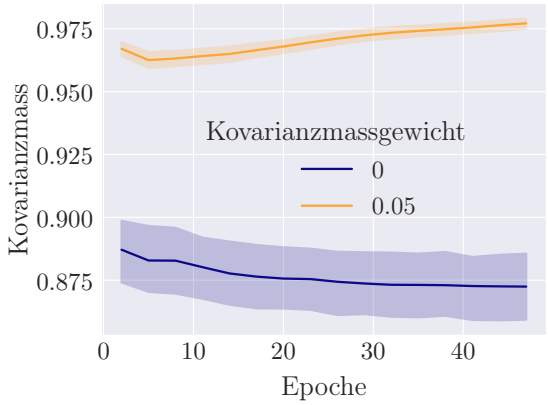

Abbildung 19: Entwicklung des Kovarianzmasses bei einer Lernrate von 0.01

5 Zusammenfassung

Zusätzlich zu Informationen bezüglich der Qualität der Vorhersagen der letzten Schicht nimmt unsere Methode Information der Funktionen der vorletzten Neuronenschicht in Betracht. Spezifisch messen wir mit einem *Kovarianzmass*, wie nah diese *Eigenschaftsfunktionen* daran sind, orthogonal zu sein. Zuerst zeigten wir, dass für anspruchsvollere Probleme dieses Kovarianzmass im Durchschnitt höher ist. Demzufolge orthogonalisieren neuronale Netzwerke ihre Eigenschaftsfunktionen bereits bis zu einem gewissen Grad, ohne dafür explizit durch den Verlust belohnt zu werden. Um das neuronale Netzwerk anschliessend für orthogonalere Eigenschaftsfunktionen zu belohnen, wurde das Kovarianzmasses vom Verlust subtrahiert. Da die resultierenden Eigenschaftsfunktionen nahezu orthogonal sind, könnte in einem nächsten Schritt die optimale Gewichtskonfiguration zwischen der letzten und der vorletzten Neuronenschicht gefunden werden. Zum einen könnten dadurch potentiell schneller bessere Resultate erreicht werden. Zum anderen hat diese Methodik auch den grossen Vorteil, dass ein neues Klassifikationsproblem, welches mit bereits trainierten orthogonalen Eigenschaftsfunktionen gelöst werden kann, nun effizient für die gegebenen Eigenschaftsfunktionen optimal gelöst werden kann.

6 Ausblick

Ein erster Schritt, um unsere Methodik auf den Prüfstand zu stellen, wäre es, zu versuchen unsere Resultate mit komplexeren Datensets wie beispielsweise MNIST [9] zu replizieren. MNIST ist ein Standardproblem des maschinellen Lernens, bei dem ein neuronales Netzwerk handgeschriebene Ziffern erkennen soll.

Zudem könnte die Projektion der Zielfunktion auf die Eigenschaftsfunktionen berechnet werden, auch da nicht sicher ist, wie gut dieser Schritt auf nicht vollständig orthogonalen Eigenschaftsfunktionen anwendbar ist. Es wäre durchaus möglich, dass so für Szenarien mit einer kleinen Anzahl an Eigenschaftsfunktionen mit weniger Rechenaufwand höhere Genauigkeit erzielt werden könnte.

Eine in Abbildungen 15, 17 und 19 ersichtliche Beobachtung, welche für weitere Forschung von Interesse sein könnte und von uns noch nicht definitiv erklärt werden konnte, ist, dass das Kovarianzmass in regulären Trainingsprozessen von Epoche zu Epoche abzunehmen scheint.

7 Danksagung

Der Grossteil dieser Arbeit wurde während dem Research Science Institute 2021 unter Mentoring von Professor Lizhong Zheng und Dr. Erixhen Sula ausgeführt und in englischer Sprache verfasst. Ich möchte dementsprechend zuerst vor allem Professor Lizhong Zheng, Dr. Erixhen und Herrn Bernhard Pfammatter herzlich danken. Ich bin auch der Koreferentin für diese Arbeit, Frau Aline Steiner zu Dank verpflichtet. Die Organisation meiner Teilnahme an dem oben erwähnten Forschungsprogramm verdanke ich dem Massachusetts Institute of Technology, dem Center of Excellence in Education und der FBK Bern. Zudem möchte ich mich für ihre finanzielle Unterstützung bei der René & Susanne Braginsky Stiftung und der Fritz-Gerber-Stiftung bedanken.

8 Literatur

[1] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(October), 1950.

[2] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

[3] Ioannis Sechopoulos, Jonas Teuwen, and Ritse Mann. Artificial intelligence for breast cancer detection in mammography and digital breast tomosynthesis: State of the art. *Seminars in Cancer Biology*, 72:214–225, 2021. Precision Medicine in Breast Cancer.

[4] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[5] Grant Sanderson. Abstract vector spaces | chapter 16, essence of linear algebra. `https://www.youtube.com/watch?v=TgKwz5Ikpc8`, 2016.

[6] Wikipedia. Menge (Mathematik) — Wikipedia, die freie enzyklopädie. `http://de.wikipedia.org/w/index.php?title=Menge\%20(Mathematik)&oldid=216106618`, 2021. [Online; zugegriffen am 13. Oktober 2021].

[7] Wikipedia. Vektorraum — Wikipedia, die freie enzyklopädie. `http://de.wikipedia.org/w/index.php?title=Vektorraum&oldid=215372159`, 2021. [Online; zugegriffen am 13. Oktober 2021].

[8] Wikipedia. Function space — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Function\%20space&oldid=1029300370`, 2021. [Online; accessed 14-October-2021].

[9] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[10] Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Commun. Inf. Theory*, 1(4):417–528, December 2004.

A Anhang

A.1 Anmerkung zum Begriff der Orthogonalität

Zum Teil wurde in dieser Arbeit angemerkt, dass die Eigenschaftsfunktionen orthogonaler (zueinander) sind. Mathematisch gesehen ist Orthogonalität binär, entweder die Eigenschaftsfunktionen sind orthogonal oder sie sind es nicht. Da die strenggenommen richtige Formulierung, „Eigenschaftsfunktionen, welche weniger weit davon entfernt sind, orthogonal zu sein“, allerdings etwas umständlich ist und keine grossen Vorteile zum Verständnis des Texts hat, wurde zum Teil auf die einfachere Formulierung zurückgegriffen.

A.2 Anmerkung zur Abbildung 9

In Teilabschnitt 4.1 wird die Abbildung 8 als Beispiel für das mit der Komplexität des gegebenen Problems wachsende Kovarianzmass angegeben. Allerdings zeigt Abbildung 9 klar, dass komplexere Probleme zum Teil auch mit höherer Genauigkeit gelöst werden. Unsere Hypothese für dieses Verhalten ist, dass die Eigenschaftsfunktionen mit mehr Dimensionen mehr Informationen abspeichern, da in die Eingabeschicht mehr Information eingespiesen wird und so mehr abzuspeichernde Information vorhanden ist. Da die Eigenschaftsfunktionenen mehr Information abspeichern, sind sie weniger redundant und haben ein höheres Kovarianzmass. Mit diesen zusätzlichen Informationen kann das Problem besser gelöst werden, weshalb die Genauigkeit steigt.

A.3 Beweis zur Projektion auf die lineare Hülle orthogonaler Vektoren

Theorem A.1. *Für die Projektion eines Vektors w auf die lineare Hülle zueinander orthogonaler Vektoren $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$ gilt*

$$\text{proj}_H \, \vec{w} = \sum_{i=1}^{n} \text{proj}_{\vec{v_i}} \, \vec{w} = \text{proj}_{\vec{v_1}} \, \vec{w} + \text{proj}_{\vec{v_2}} \, \vec{w} + \cdots$$

Wir treffen dabei folgende Annahmen

$$\vec{v_i} \in \mathbb{R}^m, m \in \mathbb{N}$$
$$\vec{w} \in \mathbb{R}^o, o \in \mathbb{N}$$
$$||\vec{v_i}||^2, ||\vec{w}||^2 > 0$$
$$\vec{v_i} \cdot \vec{v_j} = 0 \; \forall i \neq j$$

Im Folgenden stellt $v_{i,j}$ das j-te Element des i-ten Vektors und w_i das i-te Element des Vektors w dar. Zudem definieren wir $v_{i,j} = 0 \; \forall j > m$ und $w_i = 0 \; \forall i > o$. Dies bietet eine nützliche Definition für die Distanz zweier Vektoren in Räumen mit einer unterschiedlichen Anzahl Dimensionen. So beträgt beispielsweise die Distanz zwischen dem Vektor $\begin{bmatrix} 2 \\ 5 \end{bmatrix}$ und dem Vektor $\begin{bmatrix} 2 \end{bmatrix}$ genau 5.

Lemma A.2. *Die quadrierte euklidische Distanz zwischen $\vec{w}$ und einem Vektor $\vec{h} \in H = \{\sum_i k_i \vec{v_i}\}$ ist mit obigen Annahmen genau dann minimal, wenn für jedes $k_i \in \{k_1, k_2, \cdots, k_n\}$ gilt, dass*

$$k_i \vec{v_i} = \text{proj}_{\vec{v_i}} \vec{w} = \frac{\vec{v_i} \cdot \vec{w}}{||\vec{v_i}||^2} \vec{v_i}$$

Aus Lemma A.2 folgt nach der in Teilabschnitt 2.3 beschrieben Definition der Projektion direkt das Theorem A.1.

Sei D nun die quadrierte euklidische Distanz zwischen $\vec{w}$ und $\sum_i k_i \vec{v_i}$

$$D = ||\sum_{i=1}^{n}(k_i\vec{v_i}) - \vec{w}||^2 \tag{1}$$

$$= (k_1 v_{1,1} + k_2 v_{2,1} \cdots - w_1)^2 + (k_1 v_{1,2} + k_2 v_{2,2} \cdots - w_2)^2 \cdots \tag{2}$$

$$\frac{\partial D}{\partial k_i} = 2v_{i,1}(k_1 v_{1,1} + k_2 v_{2,1} \cdots - w_1) + 2v_{i,2}(k_1 v_{1,2} + k_2 v_{2,2} \cdots - w_2) \cdots \tag{3}$$

$$\frac{1}{2}\frac{\partial D}{\partial k_i} = k_i v_{i,1}^2 + v_{i,1}\sum_{j\neq i}(k_j v_{j,1}) - v_{i_1} w_1 + k_i v_{i,2}^2 + v_{i,2}\sum_{j\neq i}(k_j v_{j,2}) - v_{i_2} w_2 \cdots \tag{4}$$

$$= k_i(\sum_{j=1}^{m} v_{i,j}^2) + \sum_{j\neq i}(k_j \sum_{k=1}^{m} v_{i,k} v_{j_k}) - \sum_{j=1}^{\max(m,o)} w_j v_{i,j} \tag{5}$$

$$= k_i||\vec{v_i}||^2 + \sum_{j\neq i}(k_j \vec{v_i} \cdot \vec{v_j}) - \vec{w} \cdot \vec{v_i} \tag{6}$$

$$\frac{\partial D}{\partial k_i} = 2k_i||\vec{v_i}||^2 - 2\vec{w} \cdot \vec{v_i} \tag{7}$$

$$\frac{\partial D^2}{\partial^2 k_i} = 2||\vec{v_i}||^2 > 0 \tag{8}$$

Aus Zeile 8 folgt, dass D als Funktion von $k_1, k_2, \cdots k_n$ strikt konvex ist. Daraus lässt sich schliessen, dass die Funktion $D(k_1, k_2, \cdots k_n)$ genau dann minimal ist, wenn diese ein lokales Extremum erreicht respektive, wenn die Ableitung von D nach $k_1, k_2, \cdots$ und k_n null ist. Daraus würde Lemma A.2 und somit Theorem A.1 folgen. Man beachte, dass Zeilen 9 bis 12 für jedes $i \in [1, n]$ gelten.

$$\frac{\partial D}{\partial k_i} = 2k_i||\vec{v_i}||^2 - 2\vec{w} \cdot \vec{v_i} \tag{9}$$

$$0 = 2k_i||\vec{v_i}||^2 - 2\vec{w} \cdot \vec{v_i} \tag{10}$$

$$k_i = \frac{\vec{w} \cdot \vec{v_i}}{||\vec{v_i}||^2} \tag{11}$$

$$k_i\vec{v_i} = \frac{\vec{w} \cdot \vec{v_i}}{||\vec{v_i}||^2}\vec{v_i} = \text{proj}_{\vec{v_i}}\,\vec{w} \tag{12}$$

$$\text{proj}_H\,\vec{w} = \sum_{i=1}^{n} k_i\vec{v_i} = \sum_{i=1}^{n} \text{proj}_{\vec{v_i}}\,\vec{w} \qquad \text{Q. E. D.}$$

Ehrlichkeitserklärung

Die eingereichte Arbeit ist das Resultat meiner/unserer persönlichen, selbstständigen Beschäftigung mit dem Thema. Ich habe/wir haben für sie keine anderen Quellen benutzt als die in den Verzeichnissen aufgeführten. Sämtliche wörtlich übernommenen Texte (Sätze) sind als Zitate gekennzeichnet.

Ort, Datum	Unterschrift

Anmerkung zur Ehrlichkeitserklärung

Wie in der Danksagung erwähnt, wurde die in dieser Arbeit diskutierten Resultate sowie eine in englischer Sprache verfasste, noch nicht gänzlich ausgereifte Version dieser Arbeit unter Mentoring von Professor Lizhong Zheng und Dr. Erixhen Sula des Massachusetts Institute of Technology im Rahmen eines Forschungsprogrammes namens *Research Science Institute 2021* erarbeitet.