Inhalt

[1. Einleitung 3](#_Toc11601950)

[2. Generierung der FEM-basierten Daten 4](#_Toc11601951)

[2.1 Allgemeines von FEM 4](#_Toc11601952)

[2.2 Vorgehensweise einer FEM-Analyse 4](#_Toc11601953)

[2.3 Automatisierung der Vorgehensweise durch SolidWorks-API mittels C# Programmierung 7](#_Toc11601954)

[2.3.1 Einführung der C# Programmierung 7](#_Toc11601955)

[2.3.2 Einführung der SolidWorks-API 8](#_Toc11601956)

[2.3.3 Automatisierte Vorgehensweise 9](#_Toc11601957)

[2.3.4 Ansammlung der Trainingsdaten 10](#_Toc11601958)

[3 Daten erkunden 11](#_Toc11601959)

[3.1 Korrelation innerhalb des Datensatz 11](#_Toc11601960)

[3.2 Statistische Verteilung 12](#_Toc11601961)

[3.2.1 Verteilung des Eingangsfeatures 12](#_Toc11601962)

[3.2.2 Verteilung von Zielgrößen 13](#_Toc11601963)

[3.2.3 Einfluss des Ungleichgewichts von Zielgrößen auf das Vorhersagemodell 14](#_Toc11601964)

[3.3 Verstärkung bzw. Ausgleich des Datensatz 15](#_Toc11601965)

[3.3.1 Addieren einen zufälligen kleinen Wert zu Eingangsfeature der Minderheit 16](#_Toc11601966)

[3.3.2 Interpolation in Eingangsfeature der Minderheit 17](#_Toc11601967)

[3.3.3 Auswirkung der Verstärkung des Datensatz auf Verteilung 19](#_Toc11601968)

[3.3.4 Auswirkung der Verstärkung des Datensatz auf Modellgüte 20](#_Toc11601969)

[4 Daten vorbereiten (einschließlich Feature Scaling) 21](#_Toc11601970)

[4.1 Ziel und Motivation 21](#_Toc11601971)

[4.2 Umsetzung gängiger Methoden 22](#_Toc11601972)

[4.2.1 Datenbereinigung 22](#_Toc11601973)

[4.2.2 Ausreißererkennung 23](#_Toc11601974)

[4.3.3 Feature Scaling 24](#_Toc11601975)

[4.3 Hinweise zur Anwendung von Feature Scaling 25](#_Toc11601976)

[5 Vorhersagemodell erstellen 25](#_Toc11601977)

[5.1 Einführung des maschinellen Lernens 26](#_Toc11601978)

[5.1.1 Arten des maschinellen Lernens 27](#_Toc11601979)

[5.1.2 Zielstellungen des maschinellen Lernens 27](#_Toc11601980)

[5.1.3 Overfitting und Underfitting 29](#_Toc11601981)

[5.2 Methoden im maschinellen Lernen 30](#_Toc11601982)

[5.2.1 Support Vector Maschine (SVM) 30](#_Toc11601983)

[5.2.2 Neuronales Netz 32](#_Toc11601984)

# 1. Einleitung

Bis Mitte des 20. Jahrhunderts wurde zunächst die theoretische Grundlage der Finite-Elemente-Methode (FEM) gebildet. Dank der Fortschritte von Rechnern ist das Lösen das komplexe theoretische Gleichungssystem möglich. Heutzutage ist FEM in der festigkeitsmäßigen Auslegung von Bauteilen weit verbreitet. Die FEM basiert auf dem Lösen von unterschiedlichen Differenzialgleichungen mittels numerischen iterativen Verfahren. Ein Bauteil wird in endlich viele Elemente, die mit einfacher und nicht überlappender Form gekennzeichnet ist, aufgeteilt. Dazu spielen sowohl die Form der Elemente als auch die Größe der Elemente eine entscheidende Rolle. Deswegen erfordert FEM entsprechend qualifiziertes Fachpersonal. Das Lösen einer Variante von Bauteilen kann manchmal stundenlang dauern. Die benötigte Bearbeitungszeit ist abhängig von Geometrie und Rechenleistung. Vorteil der FEM liegt daran, dass nach einer Berechnung alle Spannungen und dazu entsprechende Verformungen herauskommen. Bei parametrisierbaren Geometriemodellen, welche individuell auf eine maximale Zielverformung hin, ausgelegt werden soll, ist die FEM aufwendig.

Das Interesse für künstliche neuronale Netze (KNN) setzte bereits in den frühen 1940er Jahren ein, also etwa gleichzeitig mit dem Einsatz programmierbarer Computer in angewandter Mathematik (David Kriesel, 2005, S. 27). Aber wegen die Beschränkung der Reichenleistung entwickelte sich die KNN sehr langsam. Bereits 1974 entwickelte Paul Werbos für seine Dissertation die Backpropagation bzw. die Fehlerrückführung (Paul Werbos, 1974). Das Modell war aber erst später von einer größeren Bedeutung. Neuronales Netz (NN) ist heutzutage ein sehr erfolgreicher Ansatz aus dem Bereich des maschinellen Lernens, welcher auf Basis von verfügbaren Wissen verallgemeinert. D.h. es lernt nicht das verfügbare Wissen auswendig, sondern erkennt. Auf Grund der Fortschritte die Rechenleistung von CPU und GPU entsteht die Möglichkeit, tieferes und tieferes Neuronales Netz zu antrainieren. Typische eingesetzte Beispiele von maschinellen Lernen sind Klassifikation von Bildern, Text- und Spracheübersetzung, und Empfehlungssystem. Dazu kommt spezielle Struktur des Netzes, z.B. Convolutional Neural Network (CNN) und Rekurrentes neuronales Netz (RNN).

Im Rahmen der Arbeit soll untersucht werden, ob wissensbasierte Methoden (Support Vector Maschine - SVM, Neuronale Netze - NN) ähnlich qualifizierte Aussagen treffen können, wie die numerische Simulation. Diese Arbeit konzentriert sich darauf, wie die maximale Verschiebung eines Bauteils unter bestimmten Lasten vorhergesagt werden kann. Dazu sind an mehreren parametrischen Bauteilen entsprechende SVM und NN aufzustellen, zu dimensionieren und zu validieren. Die benötigten Daten bzw. Wissen wird durch ein Addin-Modul „Simulation“ in einer CAD Software SolidWorks erzeugt. Um die Datengenerierung automatisch laufen zu können, ist die Anruf von SolidWorks durch „Application Programming Interface“ (API) in C# nötig. Die entsprechenden Wertebereiche für Bauteildimensionen und Lasten sind sinnvoll einzuschränken.

Um die vollständig Vorgehensweise zu klären, wird in der Arbeit 3 parametrischen Bauteile als Beispiele genommen, nämlich Balken, L-Profil und W-Profil (Siehe Abb. 1).

()

Die Arbeit ist anhand der Vorgehensweise der Erzeugung eines Vorhersagemodells eingegliedert. Zuerst müssen die benötigten Daten bzw. Wissen durch ein automatisiertes Skript erfassen werden. Danach wird die ansammelte Daten in einem bestimmten Regeln erkundet. Dann folgt das Erstellen eines Vorhersagemodells. Dazu sind mehre Modelle mit unterschiedlichen Strukturen und Parametern sich voneinander verglichen. Im letzten Teil ist die Vertiefungsrichtung dargestellt.

# 2. Generierung der FEM-basierten Daten

## 2.1 Allgemeines von FEM

Die Methode der finiten Elemente (FEM) ist ein durch numerische Iterationen berechnetes Verfahren, das in vielen Anwendungsbereiche des Maschinenbaus und Elektromagnets zum Einsatz kommt. Die Grundgleichungen zur Beschreibung strukturmechanischer Probleme wie Deformationen, Spannungen, Geschwindigkeiten, Druck, Temperaturen usw., sind gewöhnlich oder partielle Differenzialgleichungen (DGLn) bzw. Differenzialgleichungssysteme (Friedrich U. Mathiak, 2010, S 1.1).

Der Grundgedanke von FEM besteht darin, dass die uns interessierende Bereiche in eine endliche Anzahl einfacher Teilbereiche. In FEM wird dieser Prozess Diskretisierung genannt, in der die großen Bereiche zu zerkleinern. Die Differenzialgleichungen, die das physikalische und mechanische System beschreibt, werden auf der endlichen Anzahl von Elementen gelöst.

## 2.2 Vorgehensweise einer FEM-Analyse

Im Ingenieurbereich stehen vielseitige Simulationssoftwaren zur Verfügung, die eine FEM-Analyse durchführen zu können. Z.B. ABAQUS, ANSYS und SolidWorks (mit Simulation-AddIn). Obwohl es viele Auswahl im Markt gibt, haben alle sehr ähnliche Vorgehensweise. Die beispielsweise in SolidWorks-Simulation können in 7 Schritte eingegliedert (Michael Brand, 2016, S 7).

● Erstellen einer Studie

In der SolidWorks-Simulation Zusatzanwendung stehen nicht nur konventionale lineare Mechanikanalyse, sondern auch mehre Einsatztype zur Verfügung, z.B. Thermische Analyse, Frequenzanalyse, Knickenanalyse usw. (siehe Abb. 2).

In der Arbeit wird auf Festigkeitsanalyse (Static in Abb. 2) konzentriert.



Abb. 2: Einsatztype in SolidWorks Simulation

● Anwenden des Materials

Einige am häufigsten eingesetzte Materialien sind in der Bibliothek vordefiniert und mit dazu entsprechend Name gekennzeichnet. Die benötigten Kennwerte von Werkstoff, z.B. Elastizitätsmodul und Schubmodul, sind auch dazu verbunden.

In der Arbeit wird nur ein Material „AISI 1020“ (DIN C22) als Beispiel untersucht.

● Einspannungen definieren

Ziel davon ist die Bewegungen eines Bauteils im Raum ein oder mehr Freiheitsgrad einzuschränken. Im Allgemeine heißt es auch Randbedingungen Definieren. Der Grund liegt daran, dass die Differenzialgleichungen der Mechanik die Randbedingungen benötigen, um gelöst werden zu ermöglichen. Die Type der Einspannung ist je nach dem einzelnen Szenario der Anwendung.

In der Arbeit wird eine Stirnfläche als fixiert betrachtet.

● Lasten definieren

Im Ingenieurbereich gibt’s unterschiedliche Lasten. Davon sind die äußeren Kräfte und Momente als allgemeine Lasten betrachtet.

In der Arbeit sind nur Kräfte als bestimmte Lasten berücksichtigt.

● Modell vernetzen

Im Rahmen der FEM ist die Vernetzung eines Bauteils entscheidend. Damit entstehen diese endlichen Elemente. Je nach Form, Größe und Anzahl der Elemente können die entsprechenden Ergebnisse sich voneinander abweichen.

Theoretisch stehen in der FEM vielseitige Elementtypklasse zur Verfügung, nämlich für Linienelemente gibt’s Stab und Balken, für Flächenelemente gibt’s Dreieck und Viereck, für Volumenelemente gibt’s Hexaeder und Tetraeder.

Bei SolidWorks Simulation sind drei grundlegende Vernetzungstype verfügbar: shell mesh, beam mesh und solid mesh (Glenn Whyte, 2018). Folgendes steht die Eigenschaft der typischen Vernetzungstype (siehe Tabelle 1).

Tabelle 1: Vergleich der Vernetzungstype bei SolidWorks Simulation

|  |  |  |  |
| --- | --- | --- | --- |
|  | Solid Mesh | Shell Mesh | Beam Mesh |
| Elementtype | Tetraeder | Dreieck | Balken |
| Einsätze | Für alles andere  (als standardmäßige Voreinstellung) | Dünne Bauteile | Bauteile mit konstanter Abschnitt |

Für Solid Mesh bei SolidWorks Simulation bittet der Lieferant keine andere Elementtype, sondern nur Tetraeder. D.h. in der Arbeit spielt der Elementtyp keine Rolle.

Die Größe und die Anzahl der Elemente sind sich voneinander abhängig. Je kleiner die Elemente sind, desto größere Anzahl der Elemente ist.

In der Abb. 3 wird Einfluss der Elementsgröße auf FEM Ergebnisse (maximale Verschiebung) gezeigt. Elementsgröße 0,2mm ist als angenommene Richtige bzw. Referenz betrachtet. Es weist eine Tendenz auf, dass mit absteigenden Elementsgröße neigt es dazu, größere Ergebnisse zu erzielen. Im Vergleich zu großen Elementsgröße besitzen die Vernetzungen, die mit kleinen Elementsgröße versehen sind, einen deutlichen Unterschied zur Referenz, dazu zählen sowohl die Verteilung als auch die Differenz.

In der Arbeit wird der Unterschied nicht weiter qualitativ untersucht, sondern nehmen wir eine quasi vernünftige Elementsgröße 0.2mm als Beispiel. Die ist nach der Abwägung zwischen Genauigkeit und Rechenzeit.



Abb. 3: Einfluss der Elementsgröße auf FEM Ergebnisse

(maximale Verschiebungen)

● Studie ausführen

Wenn alle benötigten Eingangsdaten (Geometrie, Material, Lasten, Vernetzung usw.) zugewiesen sind, dann kann das Berechnungsverfahren selbst automatisch durchlaufen.

● Ergebnisse analysieren

Beim Standard stehen 3 verformungsrelevante Ergebnisse zur Verfügung, nämlich vonMises Spannungen, resultierende Verschiebungen und äquivalente Dehnungen.

In der Arbeit uns interessieren die vonMises Spannungen und resultierende Verschiebungen.

## 2.3 Automatisierung der Vorgehensweise durch SolidWorks-API mittels C# Programmierung

2.3.1 Einführung der C# Programmierung

C# ist eine objektorientierte Programmierungssprache, die erste Version C# 1.0 war in 2002 von Microsoft veröffentlicht. Nach mehr als zehnjährige Entwicklungszeit hat C# große Fortschritte gemacht. Die mit der C# entwickelte Programme laufen auf der Microsoft .NET-Plattform. Dabei handelt es sich um eine virtuelle Maschine, die Programmen, die für diese Plattform entwickelt wurden, einen Prozessor vorgaukelt, der so in Form eines echten Computerchips gar nicht existiert.

In der Arbeit wird die Version C# 7.5.2 eingesetzt.

2.3.2 Einführung der SolidWorks-API

API heißt auch Programmierschnittstelle, mit der von einem Softwaresystem die anderen Programme zur Anbindung an das System zur Verfügung gestellt wird. Es gibt zwei Type von API (A.T. Mathew, 2010). Eine ist mit Abhängigkeit von Programmierungssprache, damit können sowohl die Syntax als auch die Elemente von dieser bestimmten Programmierungssprache zugegriffen werden. Im Gegensatz ist die andere ohne Abhängigkeit von Programmierungssprache. D.h. die kann von mehreren Programmierungssprachen angerufen werden.

SolidWorks ist ein kommerzielles CAD System von Dassault Systemes. Das umfassen vollständige 3D-Modellierung und umfangreiche zusätzliche Anwendungen (auf Englisch Add-In-Module), SolidWorks Simulation ist eine davon. Neben dem bietet Dassault Systemes SolidWorks-APIs in C#, VB und C++ für die Automatisierung und Anpassung der Vorgehensweise. In der Arbeit wird C# als API-Programmierungssprache eingesetzt.

Im SolidWorks gibt’s drei grundlegende Dokumente, nämlich Bauteil, Komponente und Zeichnungen. Entsprechend stehen Drei dazugehörten Objekte in SolidWorks-API, nämlich PartDoc, AssemblyDoc und DrawingDoc. Die Struktur der Objekte in SolidWorks API ist in Abb. 4 dargestellt. SldWorks funktioniert als Wurzel und alle sind davon deriviert. Jede Objekte hat seine eigenen Funktionen, damit die Manipulationen in hinterlegten Applikationen erledigt werden. Darauffolgende ist ein Beispiel.

„*swModel.SketchManager.CreateLine(0, 0, 0, 1, 0, 0);*“

Eine Instanz von ModelDoc2 heißt „*swModel*“. SketechManager ist eine von dazugehörten Objekten. Die fasst alle Manipulationen um, damit der Sketch erstellt und/oder ergänzt werden können. CreateLine ist eine Funktion davon, die eine Linie anhand der vorgegebenen Koordinaten im Sketch hinzufügen kann.

In der Arbeit wird SolidWorks 2016 und dazu entsprechende API verwendet.



Abb. 4: Struktur der Objekte in SolidWorks API

2.3.3 Automatisierte Vorgehensweise

Für das Training eines wissensbasierten Modells (z.B. Neuronales Netz) ist i.d.R. eine hohe Anzahl von Daten unerlässlich. Bei manchen Anwendungen wie Bilderarbeitung und Spracheerkennung können die benötigten Daten im Internet herausgefunden und heruntergeladen werden. Aber bei dieser Arbeit steht keine bestimmten bereiten Daten zur Verfügung. Trotzdem ist es tausende Daten manuell anzusammeln unrealistisch. Deswegen spielt die Automatisierung der Vorgehensweise eine große Rolle.

Je nach der Geometrie und Lasten unterscheidet die Anzahl des Eingangsfeatures (siehe xxx) sich. Hauptsache ist es mit dieser Eingangsfeature der Bauteil und dessen äußeren Lasten vollständig definiert werden zu können. Tabelle 2 zeiget die Beispiele (siehe Abb. 1) und dazugehörte Anzahl des Eingangsfeatures.

Tabelle 2: Anzahl des Eingangsfeatures bei unterschiedlichen Szenen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Balken | L-Profil | W-Profil | W-Profil  (Planare Deformation) |
| Anzahl des Eingangsfeatures | 6 | 7 | 9 | 8 |

Für eine gute Verallgemeinerung eines Vorhersagemodells sollen die gefütterte Traningsdaten möglichst zufällig sein. D.h. die Daten dürfen keine subjektiven Präferenzen enthalten. In C# ist Random-Funktion eine der Möglichkeiten, Zufallszahl zu erzeugen. Man muss darauf beachten, dass Random-Funktion in C# keine mathematisch echte Zufallszahl erzeugen kann, sondern Pseudozufallszahl.

In der Arbeit wird die Pseudozufallszahl verwendet. Der Unterschied der Wirkung auf Vorhersagemodell zwischen Pseudozufallszahl und mathematisch echten Zufallszahl wurde nicht tiefer untergesucht.

Das Wertebereich von Eingangsfeature beschränkt sich innerhalb einer sinnvollen Grenz. Der Grund liegt daran, dass die Komplexität des Vorhersagemodells sich auch in einem gewissen Niveau befindet. Dazu entsprechen es ein relativ kleiner Umfang von Trainingsdatensatz und eine quasi niedrige Rechenleistung. In Tabelle 3 steht das Wertbereich des Eingangsfeatures von bespielweise W-Profil-Bauteile.

Tabelle 3: Wertbereich des Eingangsfeatures von W-Profil-Bauteile

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | m | width | thickness | L1 | L2 | L3 | F1 | F2 | F3 |
| Wertbereich | [2, 4] | [3, 10] | [1, 10] | [24, 32] | [15, 21] | [14, 22] | [0, 50] | [0, 50] | [0, 50] |

Die manuelle Vorgehensweise einer FEM-Studie ist schon in Kapitel 2.2 gezeigt. Bevor jeder räumlichen FEM-Studie muss 3D-Modell vorhanden sein. D.h. zuerst muss man ein geometrisches 3D-Modell anhand der durch die Random-Funktion generierten Eingangsdaten erstellen. Danach läuft eine entsprechende FEM-Studie durch. Um vollständige automatische Durchführungen zu realisieren, wird eine for-Schleife im Programm geschrieben. Der konkrete C# Quellecode wird im Anhang beigefügt.

2.3.4 Ansammlung der Trainingsdaten

Am Ende jeder for-Schleife können alle FEM-Ergebnisse durch dazu entsprechende API-Funktion abgelesen werden. Ziel der Arbeit ist es die maximale Verschiebung eines Bauteils unter äußeren Lasten vorherzusagen, deswegen ist die maximale Verschiebung im Trainingsdatensatz unerlässlich. Darüber hinaus funktioniert die entsprechende maximale Spannung als ein zusätzlicher Faktor, der sich um die Streckgrenze behandelt (siehe Kapitel xxx). Nach erfolgreicher Ablesen von FEM-Ergebnissen wird die Zielgröße durch Microsoft-Excel-API in einer Tabelle (.xlsx) exportiert.



Abb.5: Blockdiagramm der Datenstruktur



Abb. 6: Aufbau der Datenstrukturen von W-Profil

(räumliche Deformation)

I.d.R. ist ein Datensatz wie Abb. 5 von vier Teilen aufgebaut, nämlich Geometrie, Lasten, Zielgröße und Zusatz (wenn nötig). In der Arbeit wird eine Kombination von die 4 (o. 3) Eigenschaften eine Datei genannt. Z.B. eine Zeile in Abb.6 heißt eine Datei.

In Abb. 6 wird Aufbau der Datenstrukturen von W-Profil mit räumlicher Deformation dargestellt. Davon sind die zwei „maxStress(MPa)“ und „maxDisp(mm)“ die obere genannte zwei Zielgröße. Rechts liegen die durch Zielgröße abgeleitet Eigenschaften, „class“ und „out“, die als Zusatz betrachtet und in Kapitel xxx besprecht werden.

# 3 Daten erkunden

Nach der automatisierten Datenerfassung steht eine Tabelle zur Verfügung, die alle Eingangsparameter und Zielgröße enthaltet. Beim Erkunden von Daten geht es um Statistik. D.h. in diesem Schritt werden alle Eigenschaften, die nicht direkt entdeckt werden können, mit Hilfe der statistischen Beschreibungen einfach und explizit interpoliert. Dazu zählen sowohl die Verteilung als auch die sich miteinander verbundenen Korrelationsindex. Ziel davon ist es, Universalität von Daten zu überprüfen, die bei der Verallgemeinerung des Vorhersagemodells eine entscheidende Rolle spielt.

## 3.1 Korrelation innerhalb des Datensatz

Eine Korrelation beschreibt eine Beziehung zwischen zwei oder mehreren Merkmalen, Ereignissen, Zuständen oder Funktionen. Die Maßzahlen der Korrelation liegen betragsmäßig meist in einem Bereich von -1 (ein vollständig negativer linearer Zusammenhang) bis 1 (ein vollständig positiver linearer Zusammenhang). Für zwei quadratisch integrierbare Zufallsvariablen *X* und *Y* mit jeweils positiver Standardabweichung bzw. und Kovarianz Cov(*X*, *Y*) ist der Korrelationskoeffizient (Pearsonscher Maßkorrelationskoeffizient) definiert durch:

In Abb. 7 wird die Korrelation des Eingangsfestures punktweise dargestellt. Die Diagonale steht die sogenannte Kerndichteschätzung, die wird im Kapitel 3.2.1 erläutert. Außer der Diagonale sind die andere Zelle paarweise Diagramm, in dem wird ein Feature zu alle anderen punktweise aufgezeichnet. Die entsprechenden pearsonscher Korrelationskoeffizienten befinden sich in Abb. 8. Davon liegen alle Werte (außer der Diagonale) in der Nähe von Null, deswegen weißt das Eingangsfeature kein Zusammenhang sich miteinander auf. D.h. bei solchem Datensatz importiert man keine objektive Präferenz, die eine wichtige Voraussetzung beim Trainieren des Vorhersagemodells ist. Trotzdem bei manchen Situationen würden einige richtige Vorwissen zum Datensatz geführt (Siehe Kapitel 3.3).



Abb. 7: Scatter-Matrix des Eingangsfeatures



Abb. 8: Korrelationskoeffizienten des Eingangsfeatures

## 3.2 Statistische Verteilung

3.2.1 Verteilung des Eingangsfeatures

Die Kerndichteschätzung für jedes einzelne Feature ergibt sich in Diagonale in Abb. 7, die ist ein statistisches Verfahren zur Schätzung der Wahrscheinlichkeitsverteilung einer Zufallsvariable. Im Histogramm wird die Häufigkeit einer Zufallsvariable innerhalb einem bestimmten Abstand dargestellt. Deswegen sind die Werte von Häufigkeiten unstetig, die Anzahl der Häufigkeiten ist vom Abstand abhängig. Im Vergleich dazu weist die Stetigkeit in Kerndichteschätzung auf. Durch die auf der Diagonale liegenden Kerndichteschätzungskennlinien kann man sagen, dass die Verteilungen des Eingangsfeatures angenähert als Gleichverteilung angesehen werden kann. Der Grund dahinter ist es, dass die Kerndichteschätzung am meisten Falle quasi als konstante betrachtet werden kann. Für die Vereinfachung besteht eine Anforderung, dass das auf wissensbasierte Vorhersagemodell gegen Ungleichgewicht robust sein muss (Siehe Kap. 3.3).

3.2.2 Verteilung von Zielgrößen

Wenn ein System linear wäre, würde es die Überlagerungseigenschaft erfüllen. D.h. wären alle Variable mit Gleichverteilung zufällig verteilt, würden die Funktion, die durch lineare Manipulationen von diesen Variablen erzeugt, auch eine Gleichverteilung aufweisen.

Wie in Kap. 2.1 gezeigt ist eine FEM-Analyse ein System von partiellen Differentialgleichungen, sondern nicht als lineare System berücksichtigt werden kann. Im Hinblick auf Verteilung besitzen die Zielgröße i.d.R. andere Profile als die von Eingangsfeature. Zusammenfassend führt die Gleichverteilung des Eingangsfeatures nicht zu einer Gleichverteilung einer FEM-Analyse. Die entsprechende Auswirkung wird in Abb. 9 gezeigt.

Links oben und rechts unteren steht die Kerndichteschätzung, jeweils für maximale Spannung und maximale Verschiebung. Die Beide verfügen sich über einem Ungleichgewicht, insbesondre bei max. Verschiebung. Die meisten Daten von max. Verschiebungen liegen innerhalb einem Bereich von 0 bis etwa 15mm. Trotzdem befindet die Maximale sich über 250mm. Das Ungleichgewicht wirkt sich maßgeblich auf die Genauigkeit des Vorhersagemodells. Darüber wird es in Kap. 3.2.3 untersucht.

Die restliche zwei Diagramme zeigen die Korrelation zwischen die zwei Zielgrößen. Davon kann es direkt ausgelesen, dass die max. Spannung und max. Verschiebung einen positiven linearen Zusammenhang sich miteinander haben, weil die Datenpunkte auf eine mit positive Steigerung kenngezeichnete schräge Linie zusammenziehen.

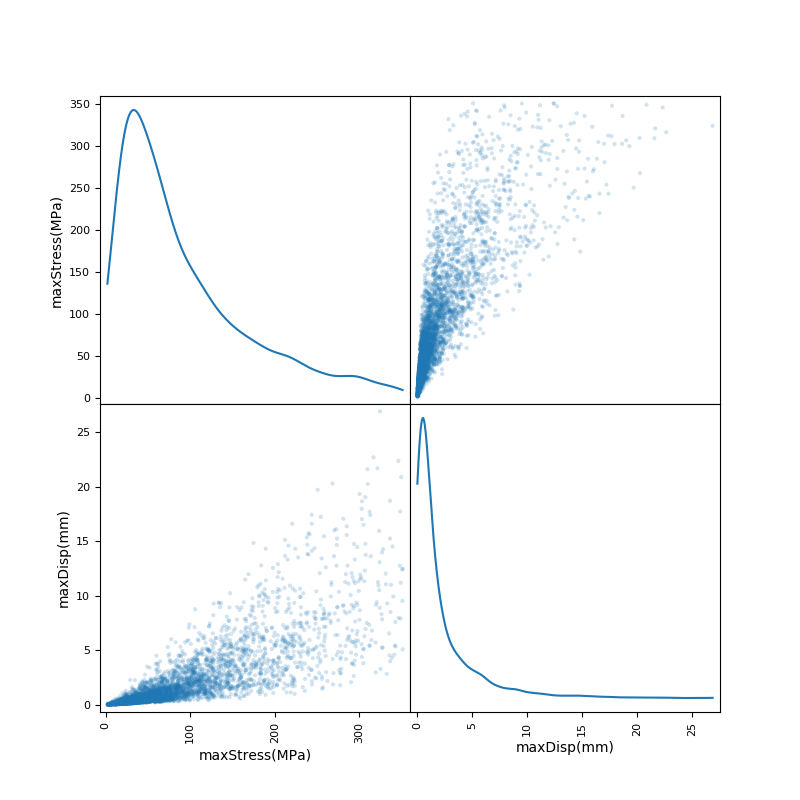


Abb. 9: Scatter-Matrix von Zielgrößen

3.2.3 Einfluss des Ungleichgewichts von Zielgrößen auf das Vorhersagemodell

Zusammenfassend könnte das Vorhersagemodell, das durch einen unausgeglichenen Datensatz trainiert wäre, auch das Ungleichgewicht vom Datensatz erlernen. Z.B. bei dem Datensatz in Abb. 9 konzentrieren die max. Verschiebungen auf kleiner als quasi 15mm. Dazu entsprechende Vorhersagemodell neigt auch zum Vorhersagen kleiner als 15mm. Wegen des Ungleichgewichts des Datensatz ist das Vorhersagemodell nicht in der Lage, die Minderheit mit hoher Genauigkeit vorherzusagen.

In Abb. 10 ist ein Verlauf der Modellgüte des Vorhersagemodells, die durch ausgeglichen Datensatz trainiert ist. Aus diesem Diagramm kann die Ungenauigkeit bei Minderheit (max. Verschiebungen großer als etwa 15mm) deutlich herausgelesen werden.



Abb. 10: Modellgüte bei einem unausgeglichenen Datensatz.

## 3.3 Verstärkung bzw. Ausgleich des Datensatz

Wie in Kap. 3.1 gesagt ein richtiger Datensatz sollte keine objektive Präferenz hereinlassen. Trotzdem zählt Ausgleich des Datensatz nicht dazu, d.h. es ist wissenschaftlich subjektive Verstärkung.

„Studies have shown that for several base classifiers, a balanced data set provides improved overall classification performance compared to an imbalanced data set (H. He, Learning from Imbalanced Data)”

In einer Arbeit von H. He „Learning from Imbalanced Data“ wurden Drei Methoden gegen Ungeleichgewicht vorgeschlagen, nämlich „Sampling Methods for Imbalanced Learning“, „Cost-Senstive Methods for Imbalanced Learning“ und „Kernel-Based Methods and Active Learning Methods for Imbalanced Learning“. Davon sind Oversampling und Undersampling in der künstlichen Intelligenz weitverbreitete. Zusammenfassend bei Oversampling werden einige zufälligen Daten von der Minderheit dupliziert, im Gegensatz dazu bei Undersampling werden einige zufälligen Daten von der Mehrheit entfernt. Deswegen haben die zwei Methoden eindeutige Nachteile. Bei Oversampling entsteht eine Overfitting-Gefahr (Siehe Kap. xxx) und Undersampling führt zur Gefahr, manche wichtigen Informationen sich verloren zu lassen.

Eine Voraussetzung hinter der obengenannten Methode ist es, dass das Datensatz fixiert ist. D.h. das Wissen für Trainieren eines Modells ist vorgegeben, das könnte schwer oder unmöglich noch mehr angesammelt werden. Dieses Problem bei der Minderheit ist noch schlimmer, deswegen ist weitere Ansammlung der Minderheit im Praxis normalerweise sehr aufwändig. Ein typisches Beispiel ist „Mammography Data Set“, eine Sammlung von Bildern aus einer Reihe von Mammographie-Untersuchungen, die an verschiedenen Patienten durchgeführt wurden. Dieses Datensatz beinhaltet 10923 „Negative“ (gesund) und 260 „Positive“ (krebsartig), dazu lässt sich die Minderheit „Positive“ im Praxis quasi unmöglich gleich wie „Negative“ anwachsen.

Die Drei von H. He vorgeschlagenen Methoden könnten so zusammengefasst, das vorgegebene fixierte Datensatz möglichst vernünftig einzusetzen oder durch Algorithmus zu ergänzen.

Bei uns in der Arbeit ist die Situation anders, d.h. im Gegensatz dazu ist unser FEM-Datensatz flexibel, sondern nicht fixiert und vorgegeben. Deswegen kommt eine neue aber einfache und direkte Lösung gegen Ungleichgewicht, die Minderheit mehr zu generieren.

3.3.1 Addieren einen zufälligen kleinen Wert zu Eingangsfeature der Minderheit

Aus dem unausgeglichenen Datensatz werden die Datenpunkte der Minderheit angefasst. Die direkt vom Datensatz ausgelesenen Datenpunkte sind als Originale in Abb. 11 mit roten Punkten angesehen. Danach wird eine Zufallsvariante zu deren Eingangsfeature addiert. Die dazu addierte Zufallsvariante besitzt eine Standardnormalverteilung, deren Dichtefunktion sich ergibt durch:

Die neue erzeugte Eingangsfeature wird durch folgende Gleichung gegeben und in Abb. 11 mit blauen Dreiecken dargestellt:



Abb. 11: Ausgleich des Datensatz durch Addieren

Die neue Eingangsfeature liegt in der Nähe von Originale und wird in der FEM-Analyse durch automatisiertes Verfahren durchgeführt. Zum Schluss kommen die dazugehörte FEM-Ergebnisse (max. Spannung und max. Verschiebung) heraus. Das Verbinden von Eingangsfeature und die zwei FEM-Ergebnisse bzw. Zielgrößen wird als ein ergänzter Datensatz angesehen und im originalen Datensatz hinzugefügt. Damit wächst die Minderheit an und glicht der originale Datensatz aus.

Würde die Minderheit in einem bestimmten Raum sich befinden, könnte das Addieren einer Zufallsvariante dazu das Raum erweitern.

3.3.2 Interpolation in Eingangsfeature der Minderheit

Interpolation ist ein Begriff in der numerischen Mathematik. Zu gegebenen diskreten Daten soll eine stetige Funktion gefunden werden, die diese Daten abbildet. Man sagt dann, die Funktion interpoliert die Daten.

Im Gegensatz zum Addieren einer Zufallsvariante begrenzt die sich bei der Interpolation auf das bestimmte Raum, dessen Grenz durch die Minderheit ausgebildet ist. Wie in Abb. 11 und 13 gezeigt steht die Originale (Minderheit) punktweise diskret im Raum zur Verfügung. In der Arbeit entspricht die Interpolation nicht 100% der mathematischen Definition. D.h. uns interessieren die Zielgrößen (max. Spannung und Verschiebung) in diesem Schritt nicht. Hauptsache ist es die neuen Datenpunkte von Eingangsfeature zu erzeugen. Danach gehen die neuen erzeugten Datenpunkte von Eingangsfeature in der automatisierten FEM-Analyse herein und kommen die entsprechenden FEM-Ergebnisse bzw. Zielgrößen (max. Spannung und Verschiebung) aus. Damit wird das Anwachsen der Minderheit realisiert.

Die konkrete Vorgehensweise läuft so (Abb. 12):

1) Für jeden Datenpunkt von Eingangsfeature (roter Punkt in Abb. 13)werden andere zufälligen Punkte bestimmen.

2) Für jede Verbindung von und wird der Mittelpunk berechnet. Es ergibt sich durch:

Neben dem Mittelpunkt könnte auch Variante z.B. ein Dritte ausgewählt werden.

In Abb.13 werden die interpolierten Datenpunkte von Eingangsfeature dargestellt (bei n=2). Davon sind die grünen Rechtecke die erzeugten Datenpunkte. Die dazu entsprechende Lage ist gleich wie erwartet, d.h. innerhalb des bestimmten Raum von Minderheit.

Durch Addieren eine Zufallsvariante (siehe 3.3.1) würde das Raum von Minderheit erweitert. Im Gegensatz dazu würde es durch Interpolation das Raum von Minderheit verfeinert.

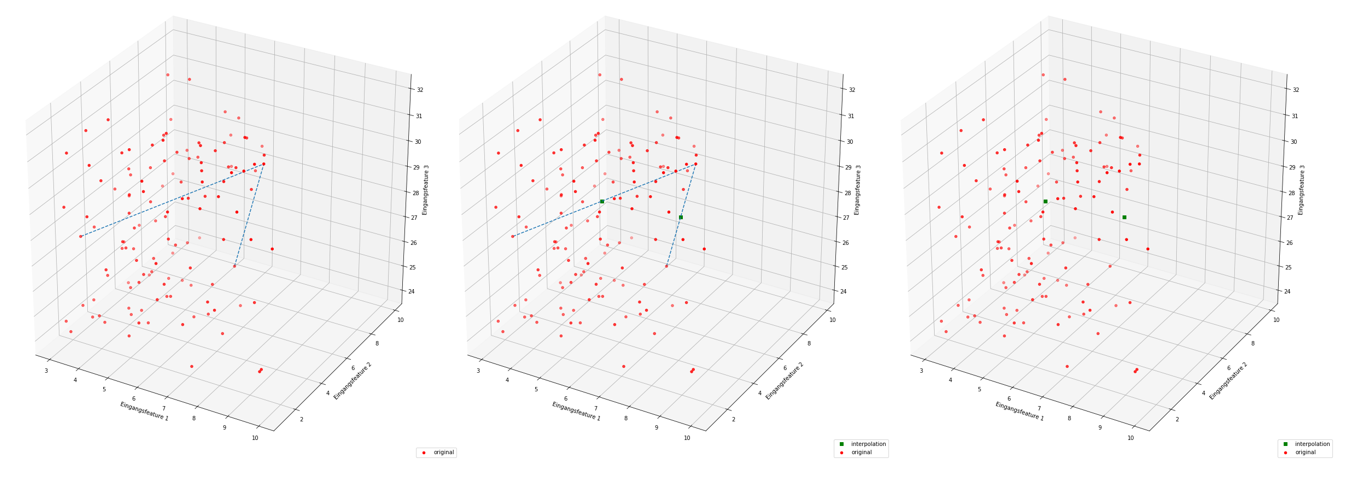


Abb. 12: Vorgehensweise der Interpolation



Abb. 13: Interpolation von Eingangsfeature der Minderheit

3.3.3 Auswirkung der Verstärkung des Datensatz auf Verteilung

In Abb. 14 ist die statistische Verteilung mit Verstärkung des Datensatz gezeigt. Solches Diagramm wurde schon in Kap. 3.2 erläutert. Im Vergleich zu Abb. 9 in Kap. 3.2.2 hat die Kerndichteschätzung von max. Verschiebung eine Verbesserung des Gleichgewichts, weil zwei Bogen im Verlauf der Kerndichtschätzung (rechts unteren) auslesbar sein können.

Trotzdem besitzt das Datensatz noch kein optimales Gleichgewicht, d.h. nach der Verstärkung zählen die Datenpunkte, die mit großen Werte versehen, auch zu Minderheit. Der Grund liegt daran, dass obwohl das Datensatz nicht 100% optimal verstärkt wird, damit kann auch eine deutliche Verbesserung der Genauigkeit bzw. Modellgüte des Vorhersagemodells geschafft werden. Darüber wird es im Folgendes bespricht.

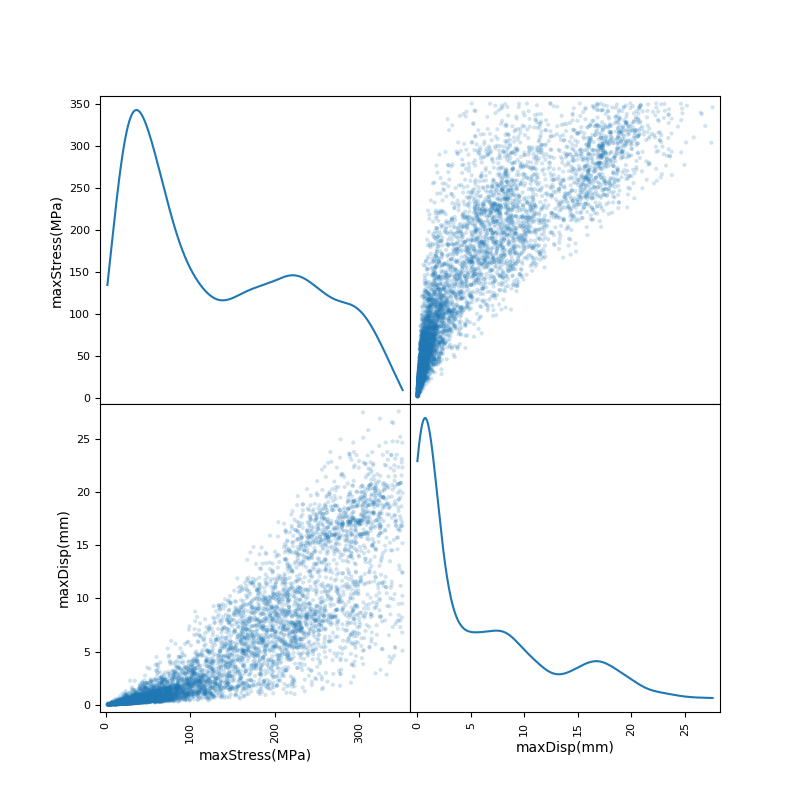


Abb. 14: Verteilung mit Verstärkung des Datensatz

3.3.4 Auswirkung der Verstärkung des Datensatz auf Modellgüte

In Abb. 15 werden zwei Diagramme für Modellgüte dargestellt, nämlich links mit Verstärkung und rechts ohne Verstärkung des Datensatz. Die X-Achse steht für die realen Werte, die direkt aus der automatisierten FEM-Analyse. Die Y-Achse steht für die Vorhersage, die direkt aus dem Vorhersagmodell.

Die Abweichung zwischen realer Wert und Vorhersage könnte mit vielseitigen Kennwerten kenngezeichnet werden (siehe Kap. Xxx), MAPE und MAE sind zwei davon. MAPE lieget im Titel des Diagramms und MAE ist durch die zwei grünen Linien kenngezeichnet.

Im rechts ist ein Trend erkennbar, dass mit steigenden realen Werte die Abweichung dazwischen sich vergrößert. Der Trend passt zum Aussage von Kap. 3.2.3. D.h. bei Minderheit und Mehrheit entsteht eine inhomogene Abweichung. Im Gegensatz dazu bei der Verstärkung des Datensatz trifft eine homogene Modellgüte im ganzen Wertebereich auf. Die Kennwerte MAPE und MAE vom links sind deutlich besser als die vom rechts.

Zusammenfassend kann man sagen, dass die Verstärkung bzw. Ausgleich des Datensatz eine Rolle als Booster des Modells spielt.

Theoretisch und mathematisch ist die positive Auswirkung der Verstärkung auf Modellgüte schwer zu beweisen und begründen. In der Arbeit wurde nur eine praktische Validierung geschafft.

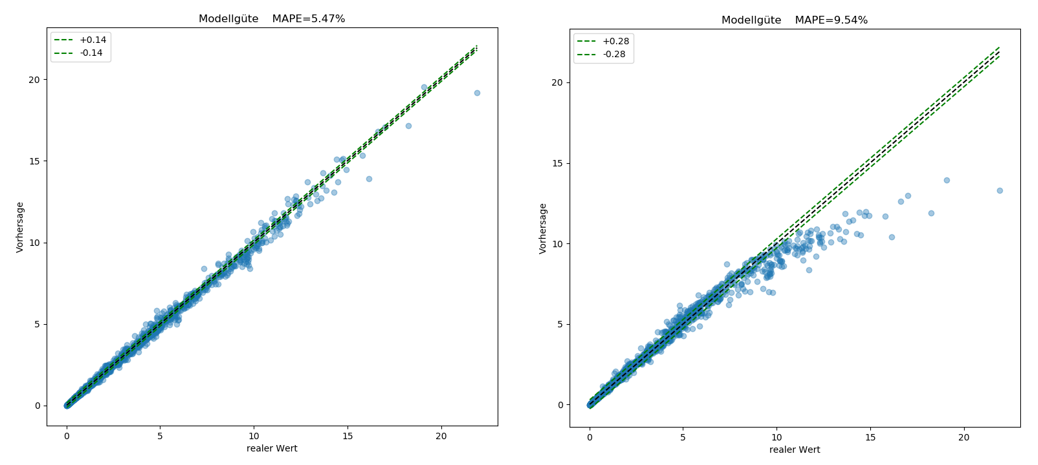


Abb. 15: Vergleich zwischen mit und ohne Verstärkung des Datensatz bei demselben Vorhersagemodell. (links) Modellgüte mit Verstärkung. (rechts) Modellgüte ohne Verstärkung.

# 4 Daten vorbereiten (einschließlich Feature Scaling)

## 4.1 Ziel und Motivation

Im Praxis enthält das Datensatz normalerweise viele Werte, die das lernfähige Algorithmus nicht erkennen kann. Z.B. NULL-Wert und Text-Wert. Dafür müssen derartige Werte beim Datenvorbereiten bereinigt werden.

Würde ein Eingangsfeature lediglich aus einen festen Wert bestehen, hätte dieses Eingangsfeature keine Auswirkungen auf Trainieren des Vorhersagmodells, d.h. kein Positive und kein Negative, sondern nur relativ mehrere Rechenzeiten. Dazu entspricht in der Arbeit beispielhaft die Elementsgröße der Vernetzung bei FEM-Analyse. Die ist auf 0,2mm festgestellt und daher im Datensatz zum Trainieren entfernt.

Die Werte eines Eingangsfeatures, die deutlich größer als die meisten sind, zählen zum Ausreißer. Die sind i.d.R. durch ein falsches Verfahren und/oder defekte Sensoren ermittelt. Da müssen die beim Datenvorbereiten herausgefunden und entfernt werden.

I.d.R. verfügen Eingangsfeature unterschiedlichen Maßstab. Z.B. in Tabelle 3 geht das Eingangsfeature „thickness“ von 1 bis 10, aber das Eingangsfeature „F1“ von 0 bis 50. Die Empfindlichkeit der Kostfunktion (Siehe Kap. xxx) gegen Änderung des Gewichts unterschiedet sich wegen unterschiedliches Maßstabs.

In Abb. 16 wird die Auswirkung von Feature Scaling auf Gradientenabstiegsverfahren (Siehe Kap. xxx) dargestellt, nämlich links mit Feature Scaling und rechts ohne Feature Scaling. Von Rot bis Grün nimmt die Kostenfunktion (Siehe Kap.) ab. Weil Eingangsfeature „thickness“ kleiner ist, braucht es eine größere Änderung von , die Kostenfunktion zu verändern. Deswegen ist die Hauptachse der Ellipse entlang der x-Achse, darauf Eingangsfeature „thickness“ steht.

Im Links geht das Gradientenabstiegsverfahren direkt nach dem Minimum der Kosten, deswegen kommt es schneller in Minimum an. Obwohl kann das Gradientenabstiegsverfahren auch in Minimum ankommen, geht es fest senkrecht zum Minimum am Anfang. Solche Laufbahn verursacht Zeitaufwand und mehre Rechenleistung.

Ziel von Feature Scaling ist es, dass den Maßstab von allen Eingangsfeature zu vereinigen. Es gibt Sonderfälle, z.B. für Entscheidungsbaum ist Feature Scaling nicht benötigt.

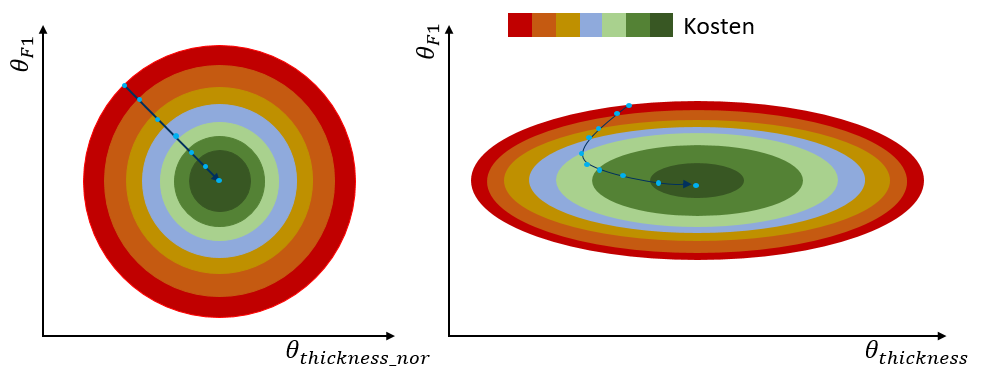


Abb. 16: Gradientenabstiegsverfahren

mit (links) und ohne (rechts) Feature Scaling

## 4.2 Umsetzung gängiger Methoden

4.2.1 Datenbereinigung

Die fehlende Werte (NULL und NaN) werden häufig durch falsch Kommunikation während der Datenerfassung erzeugt. Die sind schwer erneut genau zu bestimmen. Die direkte Lösung liegt daran, dass die die fehlende Werte enthaltende Daten ganz entfernt werden können. Die explizite Methode ist mit Mittelwert die fehlende Werte zu ersetzen. Jonathan Sterne wurde eine Methode für die fehlenden Werten in 2009 aufstellen, die heißt Multiple Imputation.

In Kap. 2.2 wird das Material in der Arbeit auf „AISI 1020“ festgelegt. Falls mehrere Varianten des Materials untersucht werden wollen, dann geht das Eingangsfeature „Material“ im Datensatz herein (siehe Tabelle 4 links). Trotzdem versteht das Vorhersagemodell die zwei Wörter „AISI 1020“ und „AISI 304“ nicht. Es kann nur mit Zahlwerte arbeiten. Eine Lösung dazu ist für jedes Material ein Eingangsfeature zu erzeugen (siehe Tabelle 4 rechts). Die Zahlwerte 1 und 0 werden zur Charakterisierung des Materials eingesetzt. Wäre ein großes Projekt mit hunderten Materialien, würde diese Lösung zur Dimensionskatastrophe führen.

Tabelle 4: Eine Lösung zur Text-Behandlung





4.2.2 Ausreißererkennung

Ausreißererkennung funktioniert als ein Filter, mit dem die Ausreißer herausgefiltert und die Normale bleibt werden können. Mittels des Box-Diagramms in Abb. 17 sind Ausreißer schematisch dargestellt. Die Datenpunkte, die entfernt von die meisten liegen, können als Ausreißer betrachtet werden. Damit kann eine feste Grenz bestimmt werden. Innerhalb dieser Grenz stehen alle normale Datenpunkte. Die Grenz ist als so genannte Filter angesehen. Beispielsweise in Abb. 17 wird die Grenz auf 150 festgelegt. Dann die Werte, die größer als xx sind, gehören zum Ausreißer und werden vom Datensatz entfernt.

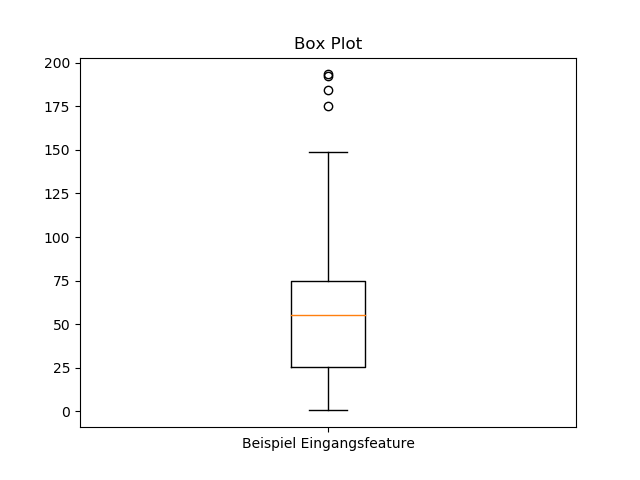


Abb. 17: Boxplot zum Erkennen des Ausreißers

4.3.3 Feature Scaling

● Normalisierung (Min-Max-Scaler)

Alle numerischen Werte des Eingangsfeatures werden auf dem Bereich von 0 bis 1 projiziert. Abb. 18 zeigt dieses Verfahren. Die mathematische Formel ergibt sich durch:

Da *m* ist die Anzahl der Trainingsdaten.

Damit können einheitsspezifische Werte zu reinen numerischen Zahlenwerte umwandeln. Weiterhin sind die vom Maßstab unabhängig.

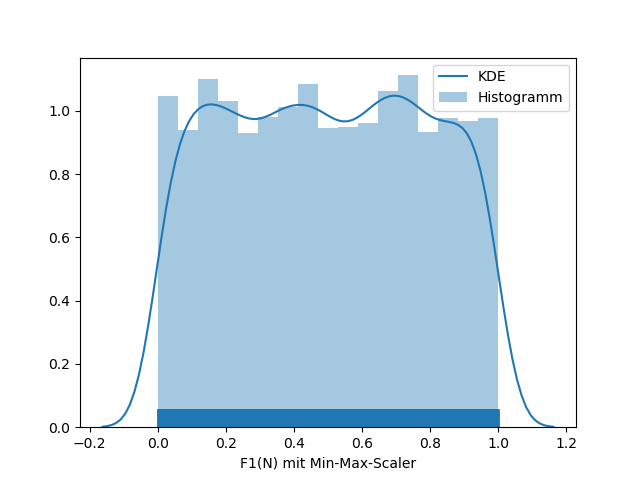


Abb. 18: Verfahren der Normalisierung,

(Links) Originale im Wertebereich von 0 bis 50,

(Rechts)Mit Min-Max-Scaler im Wertebereich von 0 bis 1.

● Standardisierung

Im Vergleich zur Normalisierung projiziert es bei der Standardisierung all originalen Werte auf einem bestimmten Bereich, wo in der Nähe von 0 liegt. Die Transformation wird in Abb. 19 dargestellt und ergibt sich durch:

Da steht für den Mittelwert, steht für die Standardabweichung, *m* ist die Anzahl der Trainingsdaten.

Mit Hilfe dieser Transformation treten die meisten sich in der Nähe von 0 und sich über eine Standardabweichung von 1 verfügbar.

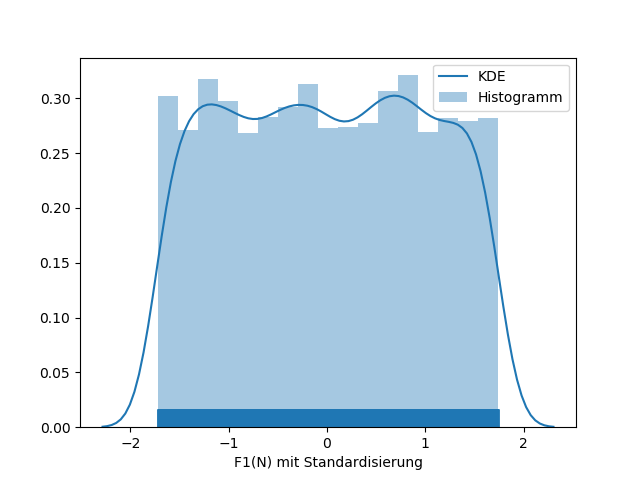


Abb. 19: Verfahren der Standardisierung,

(Links) Originale im Wertebereich von 0 bis 50,

(Rechts)Mit Standardisierung in der Nähe von 0.

## 4.3 Hinweise zur Anwendung von Feature Scaling

Die obere genannte zwei Scaler sind lediglich linear proportionale Manipulation, d.h. die eingesetzten Koeffizienten (Minimum, Maximum, Mittelwert und Standardabweichung) sind für die Scaler entscheidet. Diese Koeffizienten stammen aus den Trainingsdatensatz. Beim Testen und Anwenden des Vorhersagemodells müssen dieselbe Koeffizienten aus Trainingsdatensatz immer eingesetzt werden. D.h. man sollte nicht für Testdatensatz und Anwendungsfälle neue Scaler-Koeffizienten einstellen.

# 5 Vorhersagemodell erstellen

Ziel der Arbeit ist es ein wissensbasiertes Vorhersagemodell mit ausreichender Genauigkeit die maximalen Verschiebungen eines Bauteils unter bestimmten Lasten zu erstellen. Normalerweise ist sogenannte wissensbasierte Modell ein obenliegender Begriff. Darunter stehen Big Data, Künstliche Intelligenz, usw. Abb. 20 zeigt die Gliederung des Begriffs im Bereich eines wissensbasierten Modells. Darauf stehen Support Vector Machine (SVM) und Neuronales Netz (NN) im Fokus, weil die in der Arbeit zum Einsatz kommen.

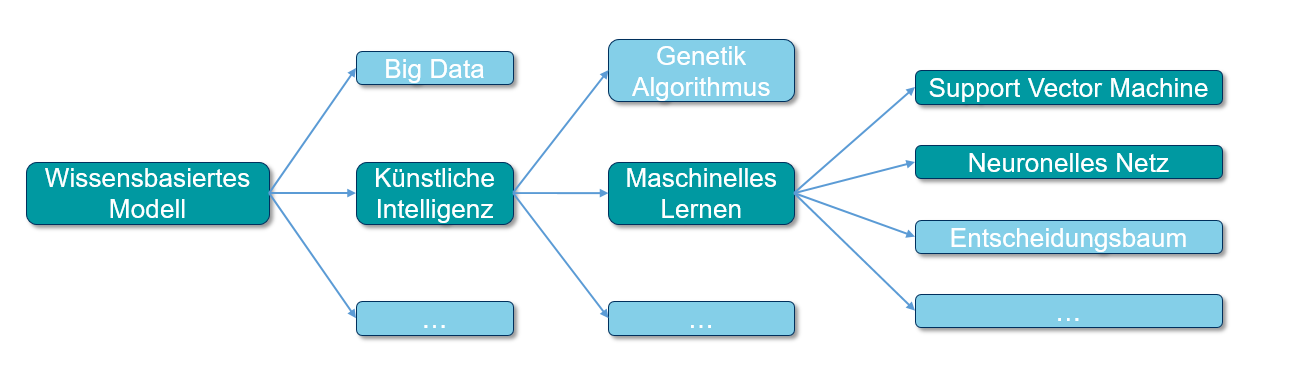


Abb. 20: Gliederung des Begriffs

Im Ingenieurbereich wirken sich die in Abb. 20 stehende Techniken als Werkzeuge, wodurch können Probleme mit einem kreativen Lösungsweg gelöst werden. Dazu verwendet man im Wesentlichen die vordefinierten Bibliotheken, welche von Informatiker und Mathematiker zusammengestellt werden. Einige typische Beispiele dazu sind TensorFlow (von Google), Sckite-Learn, PyTorch (von Facebook).

TensorFlow ist ein Framework zur datenstromorientierten Programmierung. Es wird aus Python-Programmen heraus benutzt und in Python und C++ implementiert. Populäre Anwendung findet TensorFlow im Bereich des maschinellen Lernens. In der Arbeit wird TensorFlow auch angewendet, das Vorhersagemodell zu erstellen.

Unter TensorFlow steht ein Teil Keras zur Verfügung. Keras vereinfacht die Programmierung und ermöglicht einen schnellen Anfang bzw. Keras verhält sich wie modulare Bauteile, damit kann man unterschiedliche Modelle bauen. Deswegen spielt es sowohl im Ingenieurbereich als auch in der Forschung eine wesentliche Rolle.

Scikit-Learn ist eine freie Software-Bibliothek zum maschinellen Lernen für die Programmiersprache Python. In der Arbeit wird die als Hilfsfunktion zum Trainieren und Testen des Vorhersagemodells eingesetzt.

Neben die TensorFlow und Scikit-Learn sind noch ein paar Bibliotheken benötigt, z.B. Numpy, Pandas, Seaborn usw. Die behandeln sich um sowohl Datenstruktur und Datenvisualisierung als auch Datenbearbeitung. Großer Vorteil davon ist es, dass die typischen Bibliotheken zum maschinellen Lernen dazu schon angepasst werden.

## 5.1 Einführung des maschinellen Lernens

Begriff von maschinellen Lernen ist die künstliche Generierung von Wissen aus Erfahrung. Ein künstliches System lernt aus Beispielen (gefütterte Wissen) und kann die nach Beendigung der Lernphase verallgemeinern.

Alpaydin (2016) sagte so, maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz. Durch das Erkennen von Mustern und Gesetzmäßigkeiten generieren Computersysteme selbständig Wissen aus vorliegenden Datenbeständen.

Zusammenfassend kann man sagen, dass maschinelles Lernen auf Herausfinden einer Funktion abzielen, die die Zusammenhänge zwischen Aus- und Eingangsdaten beschreiben (siehe Abb. 21).

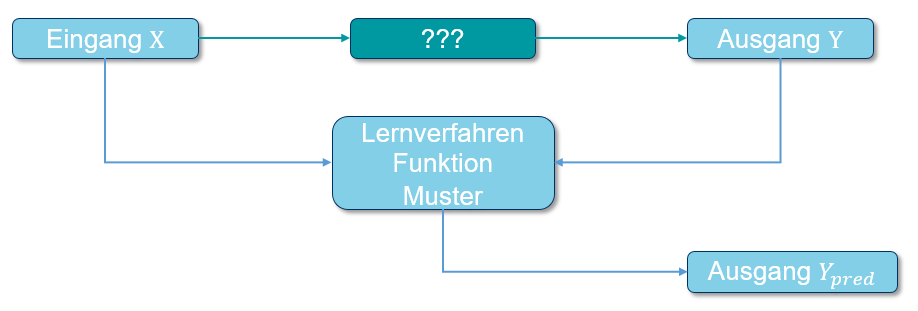


Abb. 21: Maschinelles Lernen

5.1.1 Arten des maschinellen Lernens

● Überwachtes Lernen (Supervised Learning)

Ziel besteht darin, dass eine Abbildung von Eingabewerte auf Ausgabewerte gelernt werden soll, wobei die korrekten Ausgabewerte für die dazugehörten Eingabewerte im Datensatz bekannt sind. Die Ergebnisse des Lernprozesses können mit den bekannten richtigen Ergebnissen verglichen, also „überwacht“, werden.

In der Arbeit sind alle Lernprozesse überwachtes Lernen, weil wie in Abb. 5 und 6 zeigt, die Zielgrößen (maximale Verschiebungen) sind schon im Trainingsdatensatz bekannt.

● Unüberwachtes Lernen (Unsupervised Learning)

Im Gegensatz zum überwachten Lernen sind die Ausgabewerte im Voraus nicht bekannt und uns stehen nur die Eingabewerte zur Verfügung. Der Lernprozess versucht, in den Eingabedaten Muster und Regelmäßigkeiten zu erkennen. In anderen Worten, beim unüberwachten Lernen soll die in Eingangsdaten versteckten Ähnlichkeiten herausgefunden und mit einem mathematischen Modell beschreibt werden soll.

● Verstärkendes Lernen (Reinforment Learning)

Gleich wie die beim unüberwachten Lernen uns stehen keine Ausgabewerte zur Verfügung, aber beim verstärkenden Lernen enthält das Algorithmus mit jedem Beispiel positives bzw. negatives Feedback zu seiner vorschlagenden Lösung. Das Feedback heißt also Belohnung. Der Algorithmus heißt also Agent. Ein Agent entscheidet selbstständig, welche Aktion in welcher Situation die beste ist, womit die beste Belohnung erzielen kann.

5.1.2 Zielstellungen des maschinellen Lernens

● Klassifikation – Modell zum Vorhersage der Klassenzugehörigkeit

Klassifikationsverfahren sind Methoden zur Klassifizierung von Objekten oder Situationen in Klassen. Mithilfe der Klassifikation werden Objekte oder Datensätze einer vorgegebenen Anzahl von Klassen zugeordnet. Hierzu wird ein Modell auf Grundlage von Datensätzen erzeugt, deren Klassenzugehörigkeit bekannt ist. D.h. das Klassifikationsverfahren gehört zum überwachten Lernen.

● Regression – Modell zum Vorhersage der stetigen Werten

Im Gegensatz zur Klassifikation nimmt die Zielvariable stetige Zahlwerte an. Trotzdem stehen die richtigen realen Zielvariable auch im Datensatz zur Verfügung. Daher zählt Regression zum überwachten Lernen. Mithilfe der Regression wird ein funktionaler Zusammenhang zwischen Zielvariable und Einflussvariable eines Datensatzes hergestellt.

Dies entspricht die Anforderung in der Arbeit, die maximalen Verschiebungen (stetige Zahlwerte) vorherzusagen. Deswegen ist Regression einen wesentlichen Teil des Vorhersagemodells in der Arbeit.

● Clustering

Clustering ist ähnlich zur Klassifikation, die Beispiele in Klassen bzw. Gruppen einzuordnen. Der Unterschied liegt daran, dass bei Clustering die Klassenzugehörigkeit (Zielgrößen) nicht im Datensatz bekannt sind. Dies entspricht genau die Definition des unüberwachten Lernen. Diese Verfahren haben die Aufgabe, verständliche Muster und Zusammenhänge zur Beschreibung eines komplexen Datensatzes zu generieren, ohne dass vorher bekannt ist, welche Zusammenhänge innerhalbe der Daten existieren.

● Assoziation

Die Assoziation bezeichnet die Suche nach Abhängigkeit zwischen unterschiedlichen Merkmalswerten innerhalb eines Datensatzes. In anderen Worten heißt es, die relevanten Korrelationen zwischen Variablen in großen Datensatz aufzudecken.

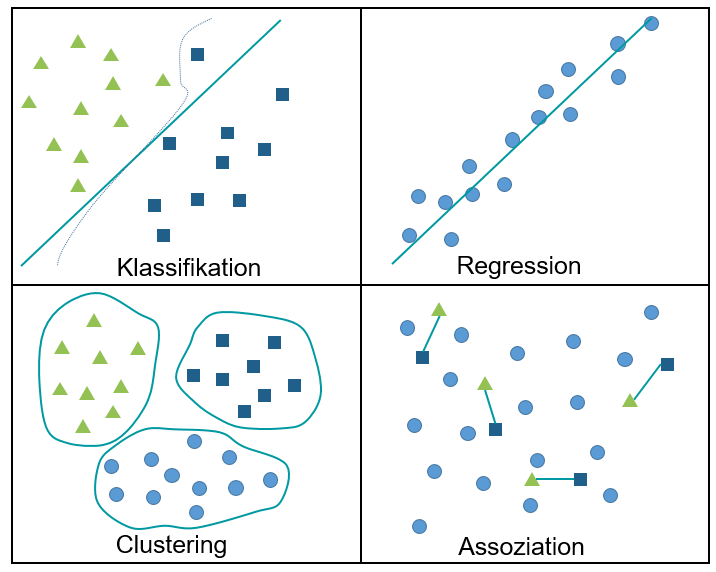


Abb. 22: Zielstellungen des maschinellen Lernens

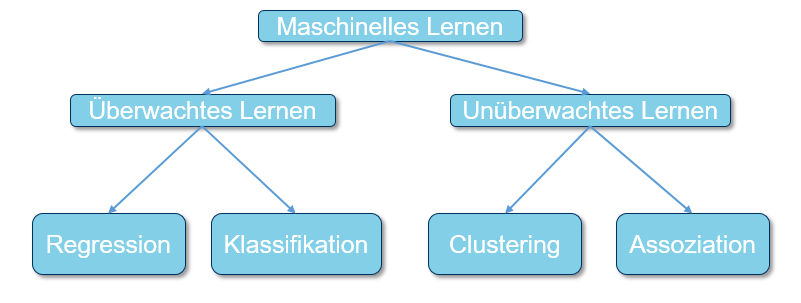


Abb. 23: Gliederung der Definitionen von maschinellen Lernen

5.1.3 Overfitting und Underfitting

Beim Overfitting eines Modells heißt es also, dass das Modell die Datenpunkte überangepasst ist. Eine zu komplexe Modell erlernt nicht nur die zugrundliegende Funktion, sondern auch Rauschen in den Daten (siehe Abb. 24 links).

Im Gegensatz dazu ist ein Modell beim Underfitting weniger komplex als die zugrundeliegende Funktion, welche damit nicht vollständig beschreibt werden kann. Z.B. Zielgröße hängt quadratisch von Einflussgrößen ab, aber es wurde nur ein lineares Modell gewählt (siehe Abb. 24 rechts).

Eine einfache Methode im Praxis die Anzahl des Neurons und der Schichten in einem neuronalen Netz auszuwählen ist „stretchhose“\*\*, wobei wird ein Modell mit mehr Schichten und Neuronen als es eigentlich braucht, dann lässt sich Regularizationstechnik (z.B. early stopping, dropout) einsetzen, das Auftreten von Overfitting zu vermeiden.

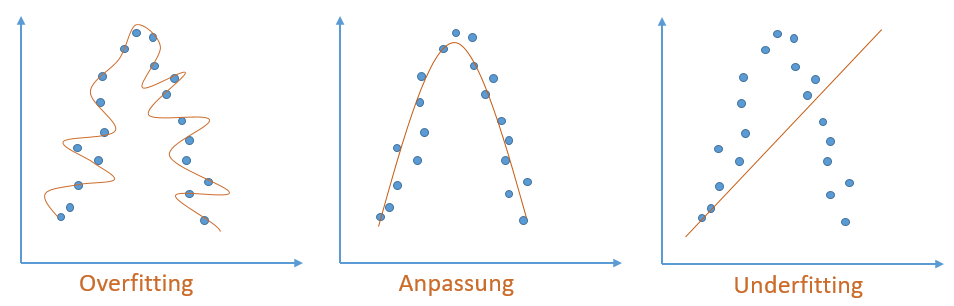


Abb. 24: (Links) Overfitting, (Rechts) Underfitting

\*\*auf engl. Stretch pants, instead of wasting time looking for pants that perfectly match your size, just use large stretch pants that will shrink down to the right size.

## 5.2 Methoden im maschinellen Lernen

Es gibt umfangreiche Methoden im maschinellen Lernen. Die enthalten lineare Regression, LASSO, Support Vector Maschine (SVM), Entscheidungsbaum, Neuronales Netz (NN) usw. Je nach den Eigenschaften des Problems wird dazu geeignete Methode eingesetzt. D.h. man muss zunächst die umfangreichen Methoden bekannt sein, in welcher Situation eine angepasste Methode eingesetzt werden könnte. Nach der Abwägung von Komplex und Reichenleistung werden SVM und NN in der Arbeit genommen.

5.2.1 Support Vector Maschine (SVM)

Eigentlich ist SVM ein 2-Klassen-Klassifikator (zu mehren Klassen erweiterbar), dessen Strategie zum Lernen ist der Abstand (auf Englisch Margin) zwischen die zwei Klassen im Featureraum zu maximieren.

Abb. 26 zeigt ein Beispiel von SVM. Die Trennebene verhält sich als Kriterium zur Klassifizierung der Datenpunkte.

Falls , dann Zuordnung zu positiven Klasse, Zielgröße .

Falls , dann Zuordnung zu negativen Klasse, Zielgröße .

Im Nachfolgendes wird die mathematischen Hintergründe dargestellt, wodurch wird der max. Margin erreicht.

● Funktioneller Margin

Der Absolutwert ist der relative Abstand zwischen Datenpunkt und Trennebene\*. Weiterhin ergibt sich den gleichen Zahlenwert wie den Absolutwert.

Damit kommt die Definition von funktionellen Margin :

Wobei, ist ein Datenpunkt im Featureraum, ist Label im Datensatz.

Weiterhin heißt ) der funktionelle Margin einer Trennebene bezogen auf vorgegebenen Datensatz.

Falls die Koeffizienten sich verdoppeln würden, würde die Trennebene sich nicht bewegen aber der funktionelle Margin sich verdoppeln. In andren Worten reicht es nicht, den Abstand vollständig zu beschreiben.

● Geometrischer Margin

Daher kommt die Definition von geometrischen Margin.

Wobei, ist 2-Norm eines Vektors .

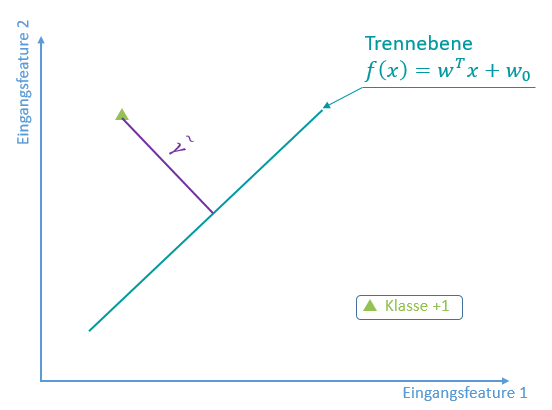


Abb. 25: Geometrischer Margin

Einen Absolutwert zu erwerben wird nochmal zu Gl. xxx multipliziert. Danach ergibt der vollständige geometrische Margin durch:

Davon entspricht der Zählerpolynom zum funktionellen Margin .

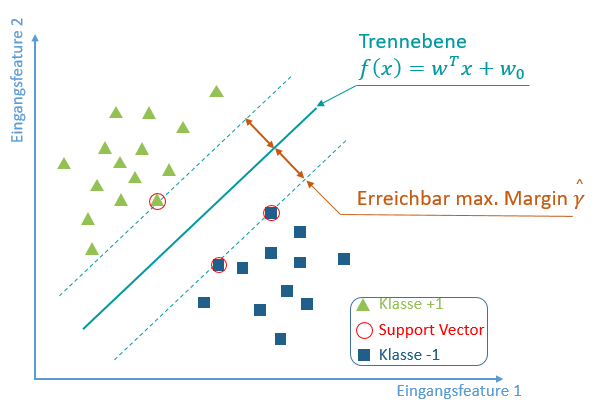


Abb. 26: Support Vector Maschine

Wie in Abb. 26 gezeigt, Ziel von uns ist es, der geometrische Margin zu maximieren:

xxx

unter Bedingung (Definition vom funktionellen Margin ):

Annahme:

\*

Dann Gl. Xxx ändert sich zu:

kkk

\*Grund der Annahme: weil die Koeffizienten proportional sich verändern können und damit bewegt die Trennebene nicht, ist die Annahme logisch.

Damit ist der Prozess, wodurch Gl. Kkk gezeigt, die Suche nach der besten Trennebene. Die Datenpunkte, die genau in Abb.26 auf den gestrichelten Linien liegen, heißen als Support Vector.

● Kernel Funktion

Das obengenannte Verfahren besitzt eine Voraussetzung, dass die Datenpunkte im Featureraum linear trennbar sein müssen, sonst weißt es eine schlechte Modellgüte beim Einsatz auf. Eine Möglichkeit dies zu lösen ist es, das nicht linear trennbare Datensatz in einen höherdimensionalen Raum zu transformieren, in welchem man sich eine bessere lineare Trennbarkeit erhofft.

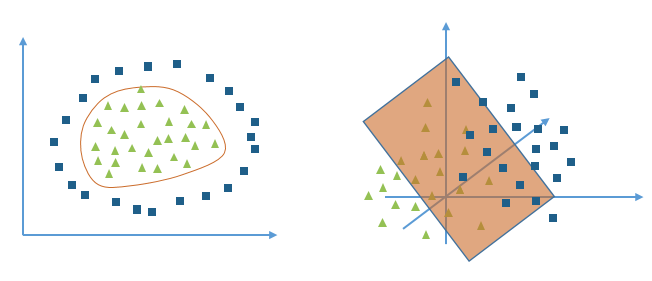


Abb. Xx: Datensatz in höherdimensionalen Raum transformieren

Abb. XX zeigt das Transformationsprozess, wobei mittels Kernel Funktion ein Datensatz sich in linear trennbare umwandeln lässt. Die üblichen Kernels sind:

lineare Kernel,

polynomiale Kernel,

wobei einer ganzen Zahl.

RBF-Kernel,

● Soft Margin

Beim Soft Margin heißt es, dass SVM eine Toleranz für Ausreißer hat, somit kann ein größere Margin erzielen. Dazu wird die Nebenbedingung in Gl. Kkk erweitert:

Steuerung dieses weichen Fehlers durch Hyperparameter in der Kostenfunktion:

Ist groß, so ergeben sich eher enge Margins, die eher selten verletzt werden, d.h. der Klassifikator passt sich sehr stark an die Daten an. Ist klein, so ergeben sich eher breitere Margins, die öfter verletzt werden, d.h. der Klassifikator passt sich den Daten nicht im Detail an.

Mit Strafterm Kompromiss zwischen Komplexität (Anzahl der Stützvektoren) und Datenfehlanpassung (Anzahl an nicht trennbaren Punkten).

5.2.2 Neuronales Netz

● Perzeptron

Das Perzeptron ist eine von einfachsten künstlichen neuronalen Netzen. Es wurde in 1957 von Frank Rosenblatt aufgestellt. Die Knotenwerte der eingehenden Schicht werden gewichtet und anschließend aufsummiert. Zum Schluss lässt die Summe sich mittels Aktivierungsfunktion transformieren, eine Zielgröße vorherzusagen. Abb. 27 ist die schematische Darstellung dieser Funktionsweise. Davon Einflussgröße , Zielgröße , Gewichte , Aktivierungsfunktion

Falls , dann Zuordnung zu positiven Klasse, Zielgröße .

Falls , dann Zuordnung zu negativen Klasse, Zielgröße

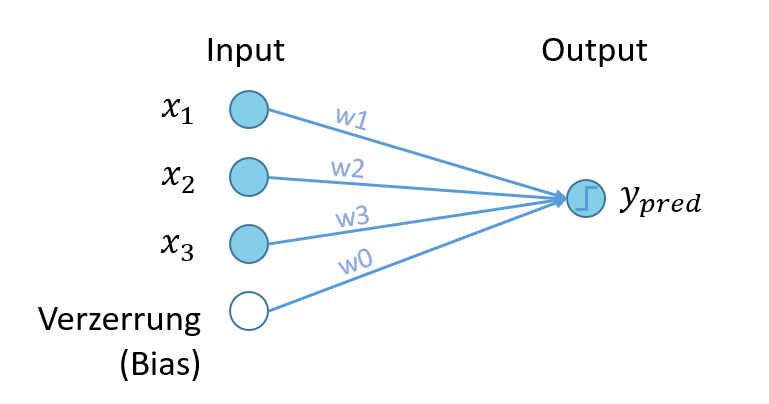


Abb. 27: Perzeptron

Trainieren eines Perzeptrons ist es so, die richtigen Zahlenwerte für die Gewichte herauszufinden. Es fangt mit bestimmten Initialwerten an Gewichte an, dann nach jeder Iteration aktualisieren sich die Gewichte, um die richtigen Zielgrößen nachzufolgen. Die Iteration könnte beendet werden, wenn die hinreichende Genauigkeit erreicht.

Zur Charakterisierung der Abweichung zwischen Richtige und Vorhersage kommt eine Kostenfunktion im Einsatz. Je nach den Arten und Zielstellungen eines maschinellen Lernens ist das Formel der Kostenfunktion unterschiedlich. Ziel des Trainierens im Allgemeinen ist es, die Kostenfunktion sich möglichsten verkleinern zu lassen.

Die Gewichte ergeben sich wieder durch eine Minimierung der Fehler zwischen Richtige und Vorhersage, also der Kostenfunktion des Perzeptrons, für ein Trainingsbeispiel mit Klasse und Vorhersage anhand der Kreuzentropie:

Fehler- bzw. Kostenminimierung wird mittels Gradientenabstiegsverfahren erledigt, damit die Gewichteupdate durch:

wobei

mit der Lernrate.

● Mehrlagige Schichten von Perzeptronen

Die mehrlagigen Schichten von Perzeptronen heißen also das sogenannte neuronale Netz. Außer den Ein- und Ausgangsschichten sind die dazwischenliegenden Schichten von Perzeptronen Zwischenschichten (auf Engl. Hidden Layer). Je mehr Zwischenschichten es gibt, desto tiefer ist das neuronale Netz, im Englischen spricht man daher auch von Deep Learning.

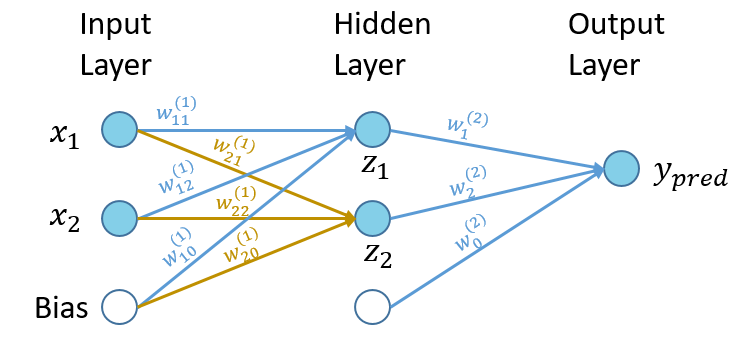


Abb. 28: Neuronales Netz

Abb. 28 zeigt ein Beispiel vom neuronalen Netz. Jede Verbindung besitzt ein eigenes Gewicht. Auf Grund der komplexen Strukturen wird die Notation für das Gewicht auch erweitert, nämlich:

Jede Einheit der Zwischenschichten berechnet sich wie das Perzeptron mit einer Aktivierungsfunktion . Damit könnte Output sich durch die Gl. Xx ergeben:

GL.xx

mit , der Aktivierungsfunktion jeweils für Hidden Layer und Output Layer. Typische Aktivierungsfunktionen sind logistische Funktion, rectified linear unit (ReLU), exponential linear unit (ELU) usw. Die Kurven und Formal dazu werden in Abb.28 dargestellt. Bei den Anwendungsfällen werden unterschiedliche Ergebnisse geliefert, wenn dieselbe neuronale Netz mit unterschiedlicher Aktivierungsfunktion verseht ist. Aber wie Anrelion Georon in seinem Buch „Hands-On Maschine Learning with Scikit-Learn and Tensorflow“ gesagt: ELU ist normalerweise besser als die andere.

In Kap. 5.1.2 sind die Zielstellungen des maschinellen Lernens vorgestellt, davon uns interessieren in der Arbeit Regression und Klassifikation. Die neuronalen Netze sind in der Lage, sowohl Klassifikation als auch Regression zu erledigen. Ob ein neuronales Netz für Klassifikation oder Regression ist, hängt es im Wesentlichen von der Aktivierungsfunktion des Output Layers ab. Für ein Klassifikationsproblem wird normalerweise eine logistische Funktion als Aktivierungsfunktion im Output Layer eingesetzt, welche die Aufsummierung der Verbindungen zu einer Wahrscheinlichkeit umrechnen kann. Damit wird die Klasse, die über höchster Wahrscheinlichkeit verfügt, als die vorhergesagte Klasse angesehen. Im Gegensatz dazu sind die Zielgrößen einer Regression stetige Zahlwerte. D.h. eine Aktivierungsfunktion im Output Layer bei einer Regression ist unnötig, wobei eine Aufsummierung der Verbindungen reicht, eine stetige Zielgröße bereitzustellen.

● Kostenfunktion

Die Definition der Kostenfunktion und deren Funktionsweise sind schon in der Vorstellung des Perzeptrons dargestellt. Für neuronales Netz ist die Situation mehr komplex als die für ein einzelne Perzeptron.

Anhand des Beispiels eines neuronalen Netzes in Abb. 27sich unterscheiden die Kostenfunktion für Klassifikation und Regression, nämlich:

für Regression, quadratischen Abweichung für ein Datenpunkt mit Vorhersage :

für Klassifikation, kreuzentropie für ein Datenpunkt mit Vorhersage :

Aktivierung der Gewichte und Berechnung der Aktivierung ergeben sich durch:

wobei m der Schichtnummer.

Nutze Kettenregel, um Einfluss der Gewichte am Anfang des Netzes zu bestimmen:

Der Einsatz von Kettenregel wird beim Trainieren eines neuronalen Netzes Backpropagation genannt, nämlich zwei Phasen: Vor- und Rückwärtsprozess.

Beim Vorwärtsprozess lässt sich ein Datenbeispiel ins Netz hereingehen, dadurch werden die Ergebnisse vom allen Neuronen ausgerechnet inklusive der Vorhersage von Output Layer. Danach wird die Vorhersage mit Richtige verglichen, die Abweichung lässt sich mit Kostenfunktion interpretieren.

Beim Rückwärtsprozess fließt die Abweichung von Output Layer zum Input Layer durch, wobei die Einflüsse von allen Gewichten der Verbindungen quantifiziert werden.

Zum Schluss lassen sich die Gewichte von allen Verbindungen mittels Gradientsabstiegsverharen so verändern, die Abweichung zu minimieren.

## 5.3 SVM Model für ein Balkenbauteil unter Lasten

Der Aufbau des Trainingsdatensatz ist in Kap. 2.3.4 gezeigt. In der öffentlichen Bibliothek Scikit-Learn ist eine SVM Funktion bereit verfügbar. Nach benötigten Datenvorbereitungen (siehe Kap. 4) lauft SVM Modell in der Gittersuche iterative durch, um ein mit besten Performance verfügtes Modell zu bestimmen.

Ein vordefiniertes SVM Modell stellt die veröffentlichte Bibliothek Scikte-Learn bereit, nämlich:

*SVR(kernel=’rbf’, degree=3, gamma=’auto\_deprecated’, coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache\_size=200, verbose=False, max\_iter=-1)*

Literaturverzeichnis

David Kriesel, Ein kleiner Überblick über Neuronale Netze, www.dkriesel.com, 2005

Paul Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Dissertation, Harvard University, 1975

Michael Brand, FEM-Praxis mit SolidWorks, Springer Vieweg, 2016

Glenn Whyte, Ins and outs on meshing elements for SOLIDWORKS SIMULATION, https://hawkridgesys.com/blog/ins-and-outs-on-meshing-in-solidworks-simulation, 2018

A.T. Mathew, C.S.P. Rao, A Novel Method of Using API to Generate Liaison Relationships from an Assembly, J. Software Engineering & Applications, 2010

Jonathan Sterne, Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls, https://www.bmj.com/content/338/bmj.b2393, 2009