technische universität
dortmund

Bachelorarbeit

Interaktive Visualisierung verschiedener Erklärbarkeitsverfahren für Neuronale Netze

Lisa Salewsky
Mai 2021

Gutachter:
Prof. Dr. Katharina Morik
Matthias Jakobs

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS VIII)
https://www-ai.cs.uni-dortmund.de/

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Hintergrund

Mit zunehmendem Einsatz von maschinellen Lernverfahren wächst zugleich auch der Bedarf an guten Erklärungen. Die Zahl der Bereiche des täglichen Lebens, in denen Entscheidungen von Maschinen getroffen werden, wächst stetig. Gleichzeitig erhöht sich die Komplexität der Probleme sowie die Anzahl der nutzbaren Informationen. Die zur Entscheidungsfindung notwendigen Algorithmen arbeiten meist auf eine komplexe Weise und mit einer für Menschen ungreifbaren Menge von Daten und Eigenschaften [**?**].

Je stärker einzelne Personen von solchen Entscheidungen betroffen sind, umso wichtiger ist es, Vertrauen zu schaffen. Damit geht zugleich die Verbesserung des Verständnisses maschineller Entscheidungen einher[**?**].

Mittlerweile gibt es zahlreiche verschiedene Ansätze, die unterschiedlichste Lösungsvorschläge für besseres Verständnis bieten. Sie lassen sich in fünf grobe Kategorien unterteilen, für die jeweils eine große Menge konkreter Algorithmen existieren.

1.1.1 Definition. In dieser Arbeit werden mit der Bezeichnung „Ansatz "die entsprechenden Grundideen zum Öffnen der Black Box bezeichnet.

1.1.2 Definition. Algorithmus, (Erklärbarkeits-)modell oder - methode bezeichnen hingegen die konkreten Implementierungen, die einem Ansatz zugeordnet werden können.

Zum Einen ist es möglich vollständig auf die Implementierung von nicht-interpretierbare Algorithmen zu verzichten. Es können stattdessen Methoden verwendet werden, die nur eine begrenzte Anzahl von Eigenschaften betrachten und nur einfache Zusammenhänge beschreiben [**?**]. Auf diese Weise bleiben die Entscheidungen für Menschen greifbar und verständlich. Eine solche starke Einschränkung hat jedoch im alltäglichen Leben oftmals Grenzen. Diese zeigen sich in verminderter Genauigkeit, sobald aufgrund der gewählten Begrenzungen, zu wenig Eigenschaften gewählt oder komplexere Zusammenhänge nicht mehr dargestellt werden können [**?**].

Sollen jedoch komplexe Lernverfahren wie zum Beispiel Neuronale Netze verwendet werden, müssen diese erklärt werden, um Verständnis zu erzeugen. Diese Erklärbarkeitsmethoden öffnen die Black Box der komplexen Funktionen und verweden dafür verschiedene Methoden [**?**].

Bei dem Ansatz der Modell-Erklärung bleibt die eigentliche Funktion im Allgemeinen eine Black Box. Nach dem erfolgreichen Trainieren des komplexen Modells wird zusätzlich ein interpretierbarer Ansatz so trainiert, dass er die gleichen Ergebnisse wie die Black Box Funktion liefert. Anhand des weniger komplexen Modells kann so die Entscheidung für Menschen zugänglich und verständlich gemacht werden [**?**].

Wie gut solche Erklärungen sind lässt sich jedoch nicht immer trivial bestimmen [**?**]. Oft sind zudem weitere Faktoren von Interesse, beispielsweise die Laufzeit oder der Speicherbedarf, wenn nur begrenzte Zeit oder Ressourcen zur Verfügung stehen. Des Weiteren ist das Verstehen von Erklärungen in vielen Fällen sehr subjektiv und kann sich von Person zu Person unterscheiden, auch wenn die eigentliche Erklärung unverändert bleibt [**?**].

Zudem können die einzelnen Teile eines Black Box Modells untersucht und die jeweiligen Auswirkungen auf eine Entscheidung erkennbar gemacht werden. Bei diesem Ansatz wird zwischen Methoden, die nur den In- und Output betrachten und solchen, die die gesamte Funktion einbeziehen unterschieden. Erstere erklären die Ausgabe eines komplexen Modells indem sie sowohl eine Vorhersage als auch eine dazugehörige Erklärung liefern und stellen somit eine Black Box Outcome Explanation bereit [**?**].

Algorithmen, die nach dem Prinzip der Erklärung interner Vorgänge der Black Box aufgebaut sind, liefern Repräsentationen der Funktionalität des komplexeren Modells. Sie bilden eine Lösung für das Black Box Inspection Problems [**?**].Somit werden Begründungen für getroffene Entscheidungen anhand der Veranschaulichung der Black Box Funktion selbst gegeben. Bei diesen kann es sich zum Beispiel um Heat-Maps **To do** Heatmap Erklärung raussuchen + Quellen (**??**) handeln.

Insgesamt zeigt sich, dass es zwar viele verschiedene Erklärbarkeitsmethoden gibt, diese aber nur schwierig zu vergleichen sind. Daher scheint es praktisch, diesen direkten Vergleich zwischen verschiedenen Erklärungsarten für bestimmte Daten zu visualisieren.

1.2 Aufbau der Arbeit

Im zweiten Kapitel werden zunächst die Zielsetzung sowie das allgemeine Vorgehen diskutiert und anschließend Grundlagen kurz aufgegriffen. Zu diesen Gehört der Aufbau eines neuronalen Netzes, das die Basis für diese Arbeit bildet, sowie die Erklärung des generellen Vorgehens beim Trainieren eines solchen Netzes anhand des Beipiels Backpropagation. Als letzter Punkt werden die Ansätze- Surrogatmodelle, Counterfactuals und Feature Contribution - denen die verwendeten Methoden angehören genauer beschrieben.

Das dritte Kapitel beschäftigt sich mit den verwendeten Erklärbarkeitsmodellen im Speziellen. Dabei werden jeweils sowohl die verwendeten theoretischen Konzepte als auch die Umsetzung im Programmcode aufgegriffen und erläutert.

Das Design und die Umsetzung der interaktiven Benutzeroberfläche in Form einer Webseite bilden Kapitel vier. An dieser Stelle wird sowohl der Designprozess beschrieben als auch Einschränkungen und Alternativen erläutert sowie die verwendete Software und der selbst implementierte Code beschrieben.

Als letztes werden die Ergebnisse der Arbeit zusammengefasst und ein Fazit gezogen. Hier werden vor allem Probleme und deren Behebung besonders beleuchtet. Des Weiteren findet eine Evaluierung der Praktikabilität und des Mehrwertes statt.

1.3 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, einem Nutzer den direkten Vergleich verschiedener Erklärbarkeitsansätze zu ermöglichen. Es sollte einem Anwender somit möglich sein, mit der gewählten Oberfläche zu interagieren. Es sollte ihm möglich sein zu entscheiden, welche Erklärungen er für seinen individuellen Anwendungsfall als hilfreich empfindet. Dabei kann der Schwerpunkt auf der Erhöhung des Verständnisses und somit dem Vertrauen in eine Entscheidung liegen, er könnte jedoch ebenso auf einer besonders kurzen Laufzeit oder geringem Speicherverbrauch liegen.

Neben einer entsprechenden Visualisierung der Erklärung sollen somit auch Metadaten angezeigt werden. Der Nutzer sollte dabei einen Überblick über für ihn wichtige Informationen erhalten, diese ein- und ausblenden können sowie beliebig viele der vorhandenen Erklärbarkeitsmethoden wählen und anzeigen lassen können. Zudem soll die Auswahl verschiedener Datenpunkte möglich sein. Auf diese Weise soll die Oberfläche individuelle Bedürfnisse und Anforderungen abdecken können.

Des Weiteren sollen die verwendeten Erklärbarkeitsmethoden verschiedenen Ansätzen angehören, gleichzeitig aber auch einen Vergleich innerhalb eines Ansatzes ermöglichen. Daher wurden sechs konkrete Algorithmen, je zwei pro Ansatz gewählt.

Um eine Visualisierung zwecks eines Vergleichs verschiedener Erklärungsarten erstellen zu können wird zunächst ein Klassifizierer benötigt, dessen Entscheidungen erklärt werden können. Bei diesem handelt es sich um ein Neuronales Netz, das trainiert und für diese Arbeit bereitgestellt wird.

Neben dem Netz werden zudem Implementierungen verschiedener Erklärbarkeitsansätze und die Erstellung einer Visualisierung in Form einer interaktiv verwendbaren Webseite benötigt. Hierbei werden die Methoden anhand der Verfügbarkeit passender Bibliotheken ausgewählt. Ein Algorithmus muss dabei folgende Kriterien erfüllen, um als passend eingestuft zu werden: Er sollte (hauptsächlich) in Python implementiert sein, der zu ver-

wendende Programmcode sollte frei verfügbar sein und sollte Erklärungen für neuronale Netze, sowie bestenfalls eine Umwandlung in eine interpretierbare Repräsentation liefern.

1.4 Vorgehen

Für diese Arbeit wird der Kreditdatensatz [?] [1] **To do** Datensatz raussuchen, Paper finden rausfinden wo und wie Referenzieren (
sowie ein darauf vortrainiertes Neuronales Netz[2] **To do** NN raussuchen, rausfinden wo und wie Referenzieren (??).

Alle folgenden Bezeichnungen werden in den jeweils referenzierten Kapitel im Detail mit der zugrundeliegenden Theorie, Berechnungen und der Implementierung erläutert.

Die verschiedenen Algorithmen, die implementiert werden , sind folgende: Für Surrogatmodelle wurde ein Entscheidungsbaum [?, ?, ?] und ein lineares Modell mit einem Lasso-Klassifizierer [?], für Counterfactuals ein Counterfactual-Modell [3] [?, 19, ?] und (DiCE)[14] und für den Feature-Contribution-Ansatz SHAP Deep Explainer [10] und Layerwise Relevance Propagation (LRP) gewählt [13, 5].

Die sechs gewählten Erklärbarkeitsmethoden werden dabei wie folgt umgesetzt:

- Entscheidungsbaum: ein Entscheidungsbaum wird auf Basis des sklearn DecisionTreeClassifiers **To do** Referenz zum Paper und Code, Doku (??) implementiert. Zudem wird die Ausgabe des jeweiligen Pfades für einen bestimmten Datenpunkt umgesetzt, auf deren Basis anschließend eine Visualisierung erstellt werden kann.

- Lasso: ebenfalls auf Basis von sklearn wird Lasso **To do** Referenz zum Paper und Code, Doku (??) eingebunden. Auch für diesen Ansatz wird eine Ausgabe in Form eines Barplots der Koeffizienten bestimmt, die über die Oberfläche visualisiert werden kann.

- FAT-Forensics-Counterfactuals: der CounterfactualExplainer **To do** Referenz zum Paper und Code, Doku (??) von FatForensics wird verwendet, um eine Counterfactual-Erklärung zu generieren. Diese kann über die entsprechende textualise-Methode **To do** footnote zu Code, Doku (??) zu einer interpretierbaren Erklärung umgewandelt und für die Visualisierung verwendet werden.

- DiCE: **To do** ordentlich formulieren (??) von DiCE Implementierung bereitgestellten Explainer nutzen außerdem bereitgestellte Mehtode für umwandeln in textuelle Ausgabe nutzen **To do** Referenz zum Paper und Code, Doku (??) **To do** richtig formulieren; nach implementationsversuchen (??)

- DeepSHAP: von DeepSHAP Implementierung bereitgestellten Explainer nutzen außerdem bereitgestellte Mehtode für umwandeln in Text nutzen **To do** Referenz zum Paper und Code, Doku (??) **To do**

[1] Link zum Datensatz? sonst Verweis auf Anhang

[2] Link zum NN Code? sonst Verweis auf Anhang

[3] `https://fat-forensics.org/generated/fatf.transparency.predictions.counterfactuals.CounterfactualExplainer.html#fatf.transparency.predictions.counterfactuals.CounterfactualExplainer`

- LRP: LRP selbst implementieren, erklärung als Text generieren, der die Beteiligung der einzelnen Feature angibt **To do** Referenz zu Code, Tutorial (**??**)**To do** richtig formulieren; nach implementationsve

Für alle Algorithmen werden zudem die Metadaten Laufzeit (in O-Notation), Speicherbedarf und Genauigkeit bestimmt.

Die interaktive Oberfläche wird mithilfe von Subplots realisiert und zeigt die Erklärbarkeitsmethoden, bietet dem Nutzer mehrere Möglichkeiten zu interagieren. Von den umgesetzten Erklärbarkeitsmethoden können beliebig viele ausgewählt und angezeigt werden. Ebenso ist es möglich die errechneten Metadaten zusätzlich oder einzeln für jede Methode einzublenden. Aus den Datenpunkten kann der Nutzer einen beliebigen auswählen und diesen detaillierter ansehen.

Kapitel 2

Grundlagen

2.1 Machine Learnings Grundbegriffe

Bei künstlicher Intelligenz handelt es sich um einen Teilbereich der Informatik, doch eine genaue, allgemeine Definition gibt es nicht. Der Begriff wird oft für bestimmte Szenarien beschrieben. Dabei überschneidet sich ein entscheidender Teil: Es werden Computer entwickelt, die in der Lage sind, sich menschenähnlich zu verhalten. Dieses schließt zum Beispiel die Fähigkeit zu Lernen oder zur Selbstverbesserung ein [] **To do** Referenzen (**??**) und ermöglicht es intelligenten Systemen, Entscheidungen zu treffen. Oftmals sind diese Klassifizierungen für Menschen schwierig zu entscheiden oder zu erkennen oder sehr komplex.

Doch es ist nicht nur der allgemeine Oberbegriff, der nicht allgemeingültig ist. Auch viele andere Bezeichnungen sind austauschbar oder unterschiedlich definiert, abhängig von einem speziellen Problem. Andere Begrifflichkeiten sind dagegen allgemein etabliert. Im folgenden werden die in dieser Arbeit verwendeten Benennungen und entsprechende Synonyme zur Klarheit definiert.

2.1.1 Definition. Künstliche Intelligenz (KI), intelligente Systeme

Die Bezeichnungen werden austauschbar verwendet und bezeichnen in dieser Arbeit Computer, die sich menschenähnlich verhalten können [].

2.1.2 Definition. Machine Learning (ML, maschninelles Lernen)

Mit dem Begriff Machine Learning wird der gesamte Lernprozess für ein intelligentes System bezeichnet. Dieser beinhaltet mehrere verschiedene Schritte, die abhängig von bestimmten Problemstellungen in ihrer konkreten Realisierung variieren können. Er Beschreibt das Entwerfen eines Algorithmus, der das gegebene Problem akkurat löst [12, 15, 2].

Die Grundlage für maschinelles Lernen bilden jedoch immer Daten, deren Art (Bilder, Texte, Tonaufzeichnungen, etc.) und Anzahl sowie die Menge und Vielfalt der einzelnen

vorhanden Feature können vollkommen unterschiedlich sein. Auf diesen werden dann verschiedenste Berechnungen ausgeführt. Heutzutage handelt es sich dabei oft um verschiedene Netze (wie zum Beispiel in 2.2 beschrieben).

Im Allgemeinen gibt es für bestimmte Probleme mehrere Optionen, einen festgelegten Algorithmus mithilfe von Parametern zu modifizieren, sodass die vorgegebene Berechnung möglichst gut an die Daten angepasst ist.

2.1.3 Definition. Training/ Lernen / Lernprozess

Eine künstliche Intelligenz wird trainiert, wenn die Parameter eines Algorithmus verändert werden, sodass die Ausgabe des Netzes (siehe 2.2) schrittweise verbessert und an die Datengrundlage angepasst wird. [2] **To do** Referenzen (**??**) Eine genauere Erläuterung des Lernprozesses ist in Kapitel 2.3 angegeben.

2.1.4 Definition. Eigenschaft (Feature)

Eine Eigenschaft beschreibt bestimmte Werte eines konkreten Objekts. Für das „Objekt"einer Person können Eigenschaften zum Beispiel der Name, das Geschlecht oder das Alter sein. [] **To do** Referenzen (**??**)

2.1.5 Definition. Datenpunkt

Ein Datenpunkt ist ein einzelnes Objekt aus allen vorhanden Daten, die für das Training, oder das Testen einer Funktion verwendet werden. Dabei wird zwischen Trainings- und Testdaten unterschieden. [] **To do** Referenzen (**??**)

2.1.6 Definition. Trainingsdatum/ Trainingsbeispiel/ Beispiel / Trainingsdatenpunkt

Ein Trainingsdatum wird zum Trainieren, also im Lernprozess eines Netzes (siehe Kapitel 2.2) verwendet. [20]

2.1.7 Definition. Testdatum/ Testbeispiel/ Beispiel / Testdatenpunkt

Ein Testbeispiel wird zum Testen der erlernten Klassifikationen genutzt. Es sind für das Netz noch unbekannte Daten, die dazu dienen, die Genauigkeit der Vorhersagen eines Netzes zu bestimmen. [20]

Neben den vielen Unterschieden in Definition und Verwendung bestimmter Begriffe haben sich jedoch einige allgemein durchgesetzt. Zu diesen gehören grundlegende Konzepte wie supervised und unsupervised (und semi-supervised learning) und die Unterscheidung in Klassifizierungs- und Regressionsprobleme. Letztere basiert zum Teil auf der vielen maschinellen Entscheidungen zugrundeliegenden Statistik und der Unterscheidung von Eigenschaften in qualitative und quantitative Merkmale.

2.1.8 Definition. Qualitative Merkmale

Eine Eigenschaft ist qualitativ, wenn sie nominalskaliert (durch einen Begriff beschrieben) oder ordinalskaliert (durch Ziffern beschrieben, jedoch ohne Aussage über eine Größe

der Abstände). Auf diesen Werten kann in der Regel nicht sinnvoll gerechnet werden, da sie keine einheitlichen Skalen und/oder Abstände einhalten. Des weiteren sind qualitative Merkmale immer diskret. [7, 17]

Typische Beispiele sind Namen, das Geschlecht oder Schulnoten. Name und Geschlecht sind dabei nominalskaliert, Schulnoten ordnial.

2.1.9 Definition. Quantitative / metrische Merkmale

Ein quantitatives Feature lässt sich durch Zahlen angeben. Die Werte können sowohl diskret als auch stetig sein, ermöglichen aber in jedem Fall sinnvolle Rechnungen. Abstände sind auf diesen Merkmalen fest durch bestimmte Einheiten (z.B. durch die Einheit 1 für natürliche Zahlen) vorgegeben. [7, 17]

Hierzu zählen zum Beispiel die Größe, das Alter oder andere Messungen, wie die Temperatur.

Auf dieser Unterscheidung basiert die Differenzierung in Regressions- und Klassifizierungsprobleme. Hierbei ist jedoch die Ausgabe (siehe Definition **??**) entscheidend.

2.1.10 Definition. Regressionsprobleme

Handelt es sich bei der Ausgabe maschinellen Lernverfahrens um ein quantitives Merkmal, liegt ein Regressionsproblem vor. [**?**]

2.1.11 Definition. Klassifikationsprobleme

Gibt ein maschinelles Lernverfahren ein qualitatives Merkmal aus, handelt es sich um ein Klassifikationsproblem. [**?**]

Neben der Unterscheidung aufgrund der Ausgabeart eines Algorithmus wird auch die grundlegende Art des Lernens unterschieden in supervised, unsupervised und semi-supervised learning.

2.1.12 Definition. Supervised Learning / überwachtes Lernen

Beim Supervised Learning sind alle Trainingsdaten Klassifiziert. Die Ausgabe, die vom Algorithmus errechnet werden soll steht fest. Anhand dieses vorgegebenen korrekten Outputs kann somit das Ergebnis automatisch geprüft und eine Fehlerrate bestimmt werden. [1]

Zum Beispiel würde folgendes Szenario als supervised Learning kategorisiert werden:

Die vorhandenen Daten sind Personendaten, anhand derer Kreditwürdigkeit bestimmt wurde. Alle Personendaten wurden bereits beurteilt (Kredit erhalten/ Kredit nicht erhalten) und sind mit den entsprechenden korrekten Labeln versehen. Ein intelligentes System wird trainiert und lernt, Vorhersagen zu machen. Die Genauigkeit der Vorhersagen kann über die Fehlerrate anhand der anders klassifizierten Beispiele bestimmt werden.

2.1.13 Definition. Unsupervised Learning / unüberwachtes Lernen

Beim Unsupervised Learning sind die vorhandenen Trainingsdaten nicht vorher klassifiziert. Der Algorithmus teilt sie lediglich anhand verschiedener Kriterien in bestimmte Gruppen ein und gibt diese Gruppierung aus. Hierbei gibt es oft keine „korrekten "Antworten, da die zugrundeliegenden Daten keine besondere Einordnung enthalten. [1]

Zu unsupervised Learning gehört beispielsweise die Bestimmung von Ähnlichkeiten verschiedener User um Vorschläge (Filme, Produkte, etc.) zu generieren, die an die einzelnen Personen angepasst sind. Hierbei können die Nutzer nicht von vornherein in bestimmte Gruppen klassifiziert werden und eine Fehlerberechnung anhand falscher Ausgaben ist ebenfalls nicht möglich.

2.1.14 Definition. Semi-Supervised Learning

Beim Semi-Supervised Learning sind Teile der vorliegenden Daten klassifiziert und benannt, andere jedoch nicht. Für Beispiele, die ein Label haben kann eine Fehlklassifikation automatisch erkannt werden, für diejenigen, die keine bestimmte Klassifizierung haben jedoch nicht. [1]

Ein Beispiel für Semi-Supervised Learning ergibt sich aus jedem Problem, bei dem eine Dunkelziffer existiert und eventuell Daten vorliegen, die nicht erkannt wurden. Zum Beispiel können Daten von zahlreichen Patienten vorliegen, die einen Coronatest haben machen lassen. Von diesen werden alle positiv getesteten vermerkt. Es bleibt jedoch unklar, ob alle nicht positiven Tests korrekt waren oder ob eine Person zu früh getestet wurde, um ein positives Ergebnis zu erhalten. Somit sind nicht sicher alle positiven Datenpunkte mit einem Label versehen, jedoch auch nicht alle unklassifiziert.

2.2 Aufbau Neuronaler Netze

Für Berechnungen von Klassifikationen werden oftmals Neuronale Netze (NN) verwendet. Diese sind in ihrer Struktur dem menschlichen Gehirn nachempfunden und erhalten einige Bezeichnungen daher aus der Biologie. [18] **To do** Referenzen (richtig???) (**??**) Es kann es verschiedenste Varianten dieser Netze geben wie zum Beispiel (Tiefe) Neuronale Netze ((Deep) Neural Networks, (D)NN), Convolutional Neural Networks (CNN) oder Reccurrent Neural Networks (RNN). Jeder dieser Typen hat besondere Eigenschaften, wie einzelne Schichten (Definition **??** verbunden werden oder welche Schichtentypen (**??**) verwendet werden [18, 21]. Für diese Arbeit ist lediglich das verwendete Neuronale Netz wichtig und daher wird auch nur dieses definiert:

2.2.1 Definition. Neuronales Netz (NN)

Ein Neuronales Netz ist die Kombination vieler einzelner Funktionen. Es besteht aus Neuronen, die in Schichten angeordnet sind (siehe Abbildung 2.1). Dabei bestimmt die Anzahl der Schichten, ob es sich um ein Single-Layer NN (es gibt nur eine Schicht) oder

Mulit-Layer NN (es hat mehrere Schichten) handelt. Dabei hat ein Netz immer eine In- und Output-Schicht. Diese überschneiden sich im Falle eines Single-Layer NN. [?, 18]

2.2.2 Definition. Neuron

Jedes einzelne Neuron erhält einen Input aus allen eingehenden Kanten (den Ausgaben der vorherigen Neuronen, multipliziert mit den entsprechenden Gewichten, siehe 2.2 und ??). Auf diesen wird meist eine Aktivierungsfunktion (Definition ??) angewandt, die die Ausgabe für das Neuron zurückgibt. Diese Ausgabe beschreibt den Wert des Neurons. [?] **To do** Referenzen (??)

Ein Neuron wird im Folgenden mit a_j^k bezeichnet. Dabei bestimmt der Index k die Schicht in der sich ein Neuron befindet, j indiziert es innerhalb der Schicht k.

2.2.3 Definition. Bias

Neben „normalen "Neuronen gibt es zudem noch Bias-Neuronen. Diese haben keinen Input und keine Aktivierungsfunktion. Sie werden dazu genutzt, die Inputs zusätzlich zu modifizieren und anpassbar zu machen. [?] **To do** Referenzen (??)

Ein Bias wird mit b^k beschrieben.Auch hier gibt k die entsprechende Schicht an, in der der Bias sich befindet.

2.2.4 Definition. Schicht

Neuronen sind in verschiedenen Schichten angeordnet. Dabei gibt jede Schicht immer eine bestimmte Aktivierungsfunktion (Definition 2.2.7) der Neuronen vor. Diese bestimmte Funktion ist ausschlaggebend für die Benennung der Schicht (z.B. Convolution, Pooling, Fully Connected, etc.) **To do** Referenzen!!! (??)

Die Input-Neuronen stellen die einzelnen Eigenschaften eines Objekts dar. Für ein Bild sind es die einzelnen Pixel, bei einer Person können es Merkmale wie zum Beispiel das Alter, die Größe und das Geschlecht sein. [?] **To do** Referenzen (??)

Die Output-Neuronen bestimmen die Klassifikation und somit die Ausgabe, die das Netz errechnet hat. Es kann hier einen einzelnen oder auch mehrere verschiedene Outputs geben. [?] **To do** Referenzen (??)

2.2.5 Definition. Gewicht

Die einzelnen Neuronen sind über Kanten miteinander verbunden. Diese beschreiben Gewichte (siehe Abbildungen 2.1 und 2.2). [?] **To do** Referenzen (??)

Diese Gewichte werden mit w_{ij}^k bezeichnet. Der Index k bestimmt hierbei die Schicht, die sich näher am Input bestimmt. Neuron i liegt in Schicht k, j in Schicht k+1.

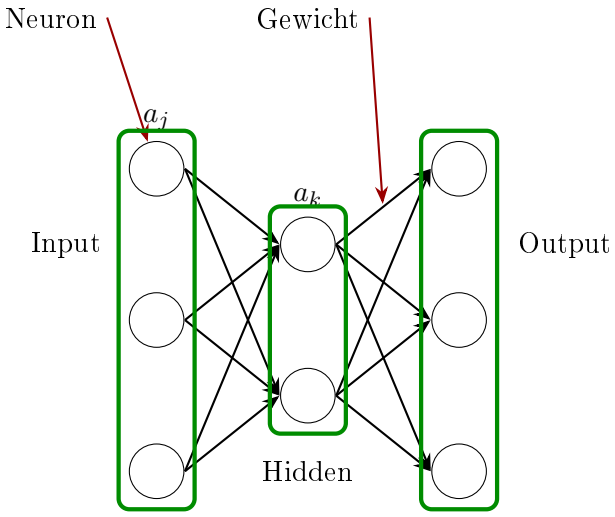


Abbildung 2.1: DNN Schichten

2.2.6 Definition. Neuroneninputs

Der Input eines Neurons setzt sich aus mehreren Teilen zusammen. Er wird im Folgenden mit z_j^k bezeichnet und besteht aus der Summe der Ausgaben vorheriger Neuronen, multipliziert mit den entsprechenden Kanten-Gewichten und einem eventuellen Bias: $z_j^k = \sum_i a_i \cdot w_{ij}^k + b^k$.

2.2.7 Definition. Aktivierungsfunktion

Die Aktivierungsfunktion wird innerhalb eines Neurons auf den jeweiligen Input angewandt und errechnet die entsprechende Ausgabe. Diese Funktionen können verschiedenste Formen haben (z.B. Sigmoid, ReLU, etc.), müssen jedoch immer differenzierbar sein. []

Wird eine Aktivierungsfunktion verwendet, so wird dies über die Notation σ gekennzeichnet.

Sigmoide Funktionen haben typischerweise eine S-Form. Sie sind beschränkt, differenzierbar und haben genau einen Wendepunkt. [?]

ReLu-Funktionen bilden das Maximum von 0 und einem weiteren Wert. Sie sorgen also dafür, alle negativen Eingaben auf 0 abzubilden. Alle anderen Werte werden auf positive Werte, z.B. die Identität abgebildet. [?, ?]

2.2.8 Definition. Neuronenoutput

Die Ausgabe eines Neurons errechnet sich aus der Anwendung der Aktivierungsfunktion auf den Neuroneninput: $a_j = \sigma(z_j^k) = \sigma(\sum_i a_i \cdot w_i j^k + b^k)$

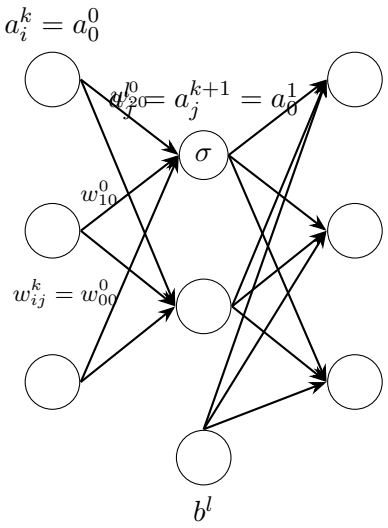


Abbildung 2.2: DNN Notationen

2.3 Der Lernprozess Neuronaler Netzwerke

Damit Neuronale Netze gute Ergebnisse liefern können, müssen sie zunächst trainiert werden. Die grundlegende Idee eines Lernprozesses für den Fall eines Regressionsproblems mit Supervised Learning wird im Folgenden anhand des Beispiels von Bilddaten als Input erläutert.

Als erstes muss ein Netz erstellt werden. Dies beinhaltet die Entscheidung über eine Anzahl von Schichten, die Auswahl dieser Schichten (-typen) und das Festlegen der Reihenfolge, in der die Schichten vorkommen sollen. Anschließend kann das erstellte Netz mit zufälligen Werten für Gewichtskanten und Biasneuronen initialisiert werden. [] **To do** Referenzen (**??**)

Es ist außerdem wichtig, eine geeignete Fehlerfunktion zu bestimmen, sodass anhand der Ergebnisse der Lernprozess stattfinden kann.

2.3.1 Definition. Fehlerfunktion/ Fehler/ Kostenfunktion

Eine Fehlerfunktion gibt meist prozentual an, wie gut oder schlecht eine Klassifizierung des Neuronalen Netzes war. Sie sollte zudem stetig sein. [] **To do** Referenzen (**??**) Sie beschreibt die „Distanz“des vom Netz bestimmten Ergebnisses zum korrekten. Sie gibt an, wie falsch das von Netz erlernte Ergebnis ist. Daher wird sie auch als Kostenfunktion bezeichnet. [**?**]

Zum Beispiel können Datenpunkte mit zwei Farben (rot, blau) als Labeln gegeben sein. Das Netz hat die Aufgabe, diese Punkte so zu unterteilen, dass alle Datenpunkte, die einen Bereich teilen, die gleiche Farbe haben. An dieser Stelle beschreibt die Fehlerfunktion die Distanz vom korrekten Ergebnis anhand der falsch klassifizierten Datenpunkte. Es muss beachtet werden, dass nicht ausschließlich die Anzahl ausgegeben wird, sondern diese sollte zusätzlich modifiziert werden. Eine Möglichkeit stellt die Berechnung der euklidischen

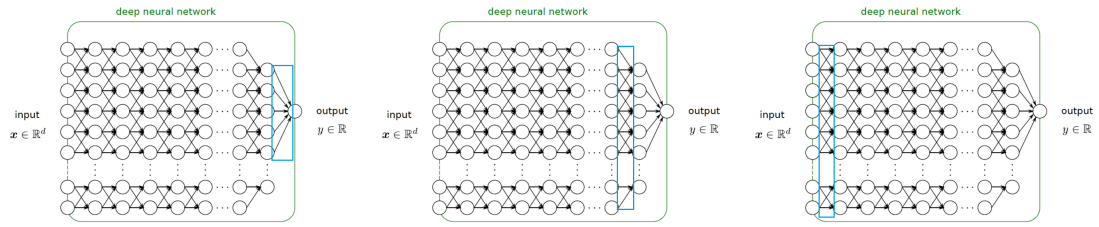
Distanz jedes Datenpunktes zur Trennung der bestimmten Bereiche dar. Punkte, die korrekt eingeordnet wurden, können eine positive Ausgabe erhalten, fehlerhaft zugeordnete hingegen eine negative. So kann die Fehlerrate des Netzes stetig und gewichtet bestimmt werden.

2.3.1 Backpropagation

Nach der ersten Erstellung und Initialisierung des Netzes muss der eigentliche Lernprozess gestartet werden. Dieser kann zum Beispiel über Backpropagation, also das Rückpropagieren von Werten durch das Netz, geschehen. [] **To do** Referenzen (**??**)

Im Lernprozess wird dabei für jeden Trainingsdatenpunkt der gleiche Prozess durchgeführt: Das Trainingsbeispiel (siehe Definition 2.1.6) wird vom Netz klassifiziert und der Fehler berechnet. Um diesen zu verbessern muss nun rückwärts das Netz durchlaufen und einzelne Neuronenoutputs angepasst werden. Um dies zu erreichen müssen die Inputs, die die Neuronen erhalten verändert werden, da die Aktivierungsfunktionen durch die entsprechende Schicht fest sind. [**?**, **?**]

Um die Inputs zu ändern können entweder die passenden Gewichte oder aber die Neuronenoutputs aus der vorherigen Schicht angepasst werden. Durch das Modifizieren von Neuronenoutputs aus vorherigen Schichten wird der gesamte Prozess eine Schicht näher zum Input verlagert. Dies geschieht, bis der Input-Layer erreicht ist und die Neuronenoutputs nicht mehr verändert werden können, da es die Eingabewerte sind. Auf diese Weise werden die Werte der einzelnen Neuronen und die Gewichte im Netz Schicht für Schicht so verändert, dass sich der errechnete Fehler verringert. [**?**, **?**]



(a) Backpropagation 1 **(b)** Backpropagation 2 **(c)** Backpropagation N

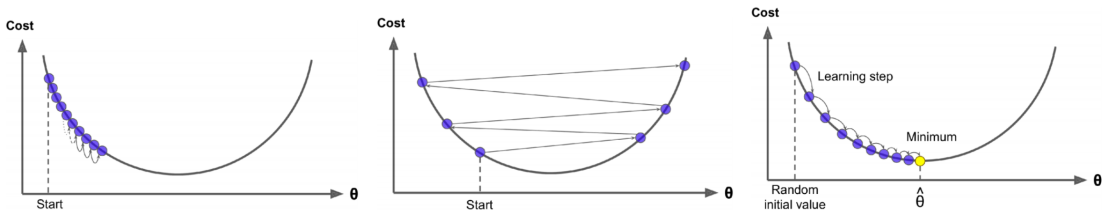
Abbildung 2.3: Backpropagation Visualisierung

Anschließend wird dieser Prozess für alle weiteren Trainingsdaten durchgeführt. Dabei kann zusätzlich durch die sogenannte Learningrate bestimmt werden, wie stark die Optimierungen sich auswirken.

2.3.2 Definition. Learningrate

Bei der Learningrate handelt es sich um einen Parameter, der in der Optimierung wichtig ist, um zu verhindern, dass sich dem Optimum in zu kleinen oder zu großen Schritten

genähert wird (siehe Grafik 2.4a). Zu kleine Schritte würden bedeuten, dass der Lernprozess sehr lange dauert, zu große Schritte können dafür sorgen, dass das Optimum im schlimmsten Fall nie erreicht werden kann (siehe Grafik 2.4b). [?]



(a) Zu kleine Learningrate **(b)** Zu große Learningrate **(c)** Optimale Learningrate

Abbildung 2.4: Visualisierung Learningrate

Nachdem der Lernprozess abgeschlossen wurde sind alle Gewichte fest und dürfen nicht mehr angepasst werden. [?] **To do** Referenzen (??)

Für die Validierung werden dann die Testdaten genutzt. Diese werden ebenfalls dem Netz als Input übergeben und klassifiziert. An dieser Stelle dürfen jedoch keine Anpassungen mehr vorgenommen werden, da der Lernprozess beendet ist. Für alle Testdaten werden die jeweiligen Fehlerraten bestimmt und ein gesamter Fehler z.B. über die Berechnung des Durchschnitts gebildet. Anhand dieses Fehlers kann die Genauigkeit der Klassifizierung des Netzes für fremde, vom Netz zuvor nicht gesehene Daten bestimmt werden. **To do** Referenzen (??)

Für die Validierung gibt es mehrere Ansätze, mit denen eine möglichst genaue Aussage über die Accuracy eines Intelligenten Systems getroffen werden kann. Zu diesen zählt zum Beipsiel Cross-Validation. Es ist jedoch auch möglich, mit je einem Trainings- und einem Testset das Netz zu validieren. **To do** Referenzen (??)

2.3.3 Definition. Cross-Validation

Cross-Validation wird für das validieren von Trainingsergebnissen verwendet. Hierbei werden die vorhandenen Daten mehrfach in verschiedene Trainings- und Testsets (Mengen aus Trainings- und Testdaten) eingeteilt. Anschließend wird das Netz mit allen bestimmten Trainingssets trainiert und mit den zugehörigen Testsets validiert. Von den bestimmten Genauigkeitswerten wird anschließend der Durchschnitt als repräsentative Accuracy des Netzes berechnet. [20]

2.4 Das gegebene Neuronale Netz

Das verwendete Neuronale Netz wurde bereits auf dem German Credit Datensatz trainiert zur Verfügung gestellt. Es besteht aus den folgenden fünf Schichten in der aufgezählten Reihenfolge: Linear, ReLU, Dropout, Linear und Softmax.

2.4.1 Definition. Linear Layer

Ein Linear Layer erhält den Namen aus der Funktion, die in den entsprechenden Knoten in diesem Layer als Aktivierungsfunktion verwendet wird. Es handelt sich hierbei somit um eine lineare Aktivierungsfunktion. [16]

2.4.2 Definition. ReLU Layer

Knoten in einem ReLU Layer nutzen als Aktivierungsfunktion eine Rectifier Linear Unit Funktion. [3]

2.4.3 Definition. Dropout Layer

Der Dropout Layer wird genutzt, um Overfitting vorzubeugen. Dabei werden im Trainingsprozess zufällig Knoten der Schicht auf Null gesetzt, um so Informationen auszuschließen. [6]

2.4.4 Definition. Softmax Layer

Der Softmax Layer wird oft in Klassifikationsproblemen mit mehreren möglichen Ausgaben verwendet. Er bestimmt die Wahrscheinlichkeiten für alle Ausgabeknoten. Aus diesen wird dann derjenige mit der höchsten Wahrscheinlichkeit als Klassifizierung des Netzes ausgewählt. [4, 9, 8]

2.5 Ansätze

Die gewählten Ansätze entsprechen jeweils einem der in der Motivation erläuterten zum Öffnen der Black Box, die von komplexe Neuronale Netze gebildet wird. Eine Erklärung mithilfe von Surrogatmodellen passt zu der Idee der Modell-Erklärung. Counterfactuals entsprechen einer Erklärung der Entscheidung lediglich anhand der Ausgabe und sind damit Black Box Outcome Explanations. Der Ansatz der Erklärung durch die Bestimmung von Feature Contributions erklärt die innere Funktionalität des Netzes und bietet daher eine mögliche Lösung für das Black Box Inspection Problem. [**?**]

Im folgenden wird die genaue Grundidee der drei Ansätze näher erläutert. In Kapitel 3 folgt zudem die Theorie und Implementierung für die konkreten Algorithmen.

2.5.1 Surrogatmodelle

Surrogatmodelle sind einfache, leicht interpretierbare, kostengünstige Approximationen von komplexen Funktionen. Ein solches Modell wird so trainiert, dass es genau wie die BlackBox-Funktion klassifiziert, inklusive der Fehler. Auf diese Weise können sie für Erklärungen von Black Box Funktionen genutzt werden. Bei der Approximation handelt es sich immer um ein leicht interpretierbares Modell, das sich weiterhin wie die zu erklärende Funktion verhält. [**?**, **?**]

Hierfür können alle interpretierbaren Modelle genutzt werden, zum Beispiel Entscheidungsbäume [?, ?] oder lineare Modelle **To do** Referenzen (??). Für diese Arbeit werden genau diese beiden Methoden implementiert.

Welche Modelle tatsächlich als leicht interpretierbar eingestuft werden können ist jedoch eine subjektive Entscheidung. Im Allgemeinen können hierzu aber diejenigen Algorithmen gezählt werden, die zu einem gewissen Grad oder vollständig transparent sind. [?] Die Aufstellung einer allgemeinen, alles umfassenden Definition für leicht interpretierbare Modelle ist nicht möglich, da je nach Anwendungsgebiet verschiedene Eigenschaften greifen. [?]

Alle entsprechenden Modelle haben jedoch gemein, dass sie für einen Nutzer Sinn ergeben, sich also in einer für Menschen verständlichen Domäne befinden. [13]

2.5.2 Counterfactuals

Counterfactuals geben Erklärungen anhand von beispielhaften Änderungen gegeben und sind damit ein besipielbasierter Ansatz. [?] Die Erklärung entsteht aus einer Wahl gut passender Attribute und einer Beschreibung der möglichen Wertänderung. Gut passend bedeutet hier, dass die Veränderung der Eigenschaften für das gesamte Objekt minimal ist, so wenig Eigenschaften, wie möglich gewählt werden und zugleich eine Änderung der Vorhergesagten Klasse erreicht wird. [?]

Diese beispielhaften Erklärungen sind oftmals von folgender Form: Wäre Attribut $X = x_i$ und $Z = z_i$ gewesen, dann wäre die Entscheidung als y_i anstelle von y_j ausgefallen. Konkret könnte dies im Fall von Kreditwürdigkeit zu der Aussage „Wäre das Einkommen in Höhe von $x_i = 5000$ €monatlich und die Schulden nicht höher als $z_i = 10000$ €gewesen, dann wäre die Kreditwürdigkeit als hoch genug eingestuft worden (y_i = Kredit wird bewilligt statt y_j = Kredit wird abgelehnt)“.

Auf diese Weise ist es möglich, ausschlaggebende Input-Features, sowie Handlungshinweise und Ähnliches bereitzustellen. Counterfactuals können so einen Einblick in die für eine Entscheidung wichtigen Eigenschaften liefern und zugleich Probleme bei der Entscheidungsfindung aufdecken. [19]

2.5.3 Feature Contribution

Die Erklärung durch das bestimmen von Feature Contributions beschreibt einen Ansatz, bei dem der Beitrag einzelner Komponenten des Modells zu einer Klassifizierung betrachtet wird. [?, ?] Hierfür wird anhand der Input-Features eine Erklärung bestimmt, die Wichtigkeit einzelner Eigenschaften für eine Entscheidung und somit den Beitrag dieser Inputs für das Ergebnis errechnet. [] **To do** Referenzen (??)

Auf diese Weise ist es möglich, auch einen Einblick in die inneren Schichten des Modells zu geben, da Contributions nicht nur auf die Input-Schicht beschränkt sein müssen. So kann mehr Transparenz geschaffen und die Funktionsweise der Black Box Funktion

verständlich gemacht werden. Auch dieser Ansatz ermöglicht zudem das Erkennen möglicher Probleme des Modells anhand der bestimmten Wichtigkeiten für die Entscheidung. [] **To do** Referenzen (**??**)

Der Unterschied zu den vorherigen Ansätzen ist hier, dass auch alle inneren Schichten des neuronalen Netzes, die hidden layer, Teil der Erklärung sind. Das Verständnis wird nicht, wie bei Counterfactuals ausschließlich aus Ein- und Ausgaben bestimmt und hat im Gegensatz zur Erstellung vollständig neuer Modelle wie bei der Verwendung von Surrogatmodellen direkten Bezug zur zugrundeliegenden Funktion. [] **To do** Referenzen (**??**)

Zu Algorithmen für diesen Ansatz gehören zum Beispiel die Layerwise Relevance Propagation (LRP) [13] oder eine Kombination aus Shapley Values und dem deep LIFT Verfahren durch DeepSHAP. [] **To do** Referenzen (**??**)

Kapitel 3

Erklärungen in der Praxis

3.1 Surrogatmodelle

3.1.1 Entscheidungsbaum

Ein Entscheidungsbaum wird zu den leichter interpretierbaren Ansätzen [?] gezählt. Somit kann für das Surrogatmodell ein Entscheidungsbaum verwendet werden.

Bei diesen handelt es sich um ein Modell, welches aus inneren Knoten, Blättern und diese verbindenden Zweigen besteht. Die Blätter enthalten konkrete Klassifizierungen, die inneren Knoten repräsentieren Verzweigungen, die bestimmte Abfragen (Tests [?]) enthalten. Anhand dieser Abfragen kann der entsprechende Zweig, eines zu klassifizierenden Datenpunktes, verfolgt werden [?]. Das Blatt der Pfades gibt die Klassifizierung an. Eine Erklärung für diese konkrete Entscheidung entspricht hierbei der Ausgabe des Pfades, den der zu erklärende Datenpunkt genommen hat [?].

Für diese Arbeit wird ein Entscheidungsbaum trainiert, der das Neuronale Netz approximiert und auf den Trainingsdaten die gleichen Klassifizierungen vornimmt, wie das Netz.

Theorie

Der implementierte Entscheidungsbaum nutzt den DecisionTreeClassifier und zugehörige Methoden von sklearn **Referenz, Doku, Code** (??).Dieser Classifier nutzt eine modifizierte Version des CART (Classification and regression trees) Algorithmus **Referenzen ml book etc** (??) zur Erstellung der Entscheidungsbäume. Hierbei ist es möglich, mehrere Parameter individuell zu bestimmen. Beispielsweise ist eine Unterscheidung zwischen der Verwendung der Gini-Unreinheit oder Entropie als Maß der Unreinheit der einzelnen Knoten möglich. []

Im weiteren Verlauf gelten für die Beschreibung von Entscheidungsbäumen folgende Namenskonventionen:

Knoten des Entscheidungsbaums werden mit $\mathbf{m}$ indiziert, Trainingsvektoren haben die Bezeichnung $\mathbf{x_i} \in \mathbf{R^n}$ mit $i \in \{1, ..., l\}$, $\mathbf{y} \in \mathbf{R^l}$ beschreiben die zu x_i zugehörigen Labelvektoren. []

Ein Split bezeichnet das Aufteilen der Datenpunkte in einem Knoten in zwei Kind-Knoten. Hierfür muss ein Splitkandidat $\boldsymbol{\theta(j, t_m)}$ mit Feature $\mathbf{j}$ und Threshold $\mathbf{t_m}$ für die entsprechende Eigenschaft in Knoten m gewählt werden. Für alle Datenpunkte in dem Knoten wird dann der Wert des Features mit dem Threshold verglichen und so ein Split bestimmt. []

$\mathbf{N_m} = |Q_m|$ bestimmt die Gesamtzahl aller Beispiele in der Menge $\mathbf{Q_m}$ in Knoten m. Die Subsets dieser Datenpunkte werden jeweils mit $\mathbf{Q_m^{left}} = \{(x, y)|x_i^j \leq t_m\}$ für diejenigen, die nicht größer als der gewählte Threshold an Knoten m sind, und $\mathbf{Q_m^{right}} = Q_m \backslash Q_m^{left} = \{(x, y)|x_i^j > t_m\}$ für die verbleibenden Datenpunkte, bezeichnet. Die Gesamtzahl der Feature in Q_m^{left} und Q_m^{right} werden analog zur Bezeichnung von Q_m mit $N_m^{left} = |Q_m^{left}|$ und $N_m^{right} = |Q_m^{right}|$ []

Eine Funktion $\mathbf{H(Q_m)}$ ist die Unreinheits- bzw. Loss-Funktion, mithilfe derer die Qualität $\boldsymbol{G(Q_m, \theta)} = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta))$ eines Splits bestimmt wird. []

3.1.1 Definition. Gini-Unreinheit

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Hierbei entspricht $\mathbf{p_{mk}}$ der Proportion der Observationen in Knoten m, die Klasse $\mathbf{k}$ angehören.

$$p_{mk} = \frac{1}{N_m} \sum_{y \in Q_m} I(y = k)$$

I ist die Identitätsfunktion, die wie folgt definiert ist:

$$Iy = k = \begin{cases} 0 & \text{falls } y \neq k \\ 1 & \text{falls } y = k \end{cases}$$

3.1.2 Definition. Entropie

$$H(Q_m) = -\sum_k p_{mk} log(p_{mk})$$

Der Algorithmus bestimmt dann mit einer der Unreinheitsfunktionen als Maß eine optimale Lösung, indem

$$\theta^\star = argmin_\theta G(Q_m, \theta)$$

rekursiv für $Q_m^{left}(\theta^\star)$ und $Q_m^{right}(\theta^\star)$ berechnet wird, bis die maximale Tiefe erreicht wurde. Auch diese Tiefe ist ein Parameter, der in sklearn der Funktion übergeben werden kann. []

Code

3.1.2 Lineare Modelle (Logistische Regression?)

Ähnlich wie Entscheidungsbäume können auch lineare Modelle als Surrogat verwendet werden. Zur Erklärung des Black Box Algorithmus werden sie ebenfalls so trainiert, dass sie exakt die gleichen Klassifikationen bestimmen, wie das zu erklärende Modell.

Um eine Ausgabe zu errechnen werden verschiedene Prädiktoren in eine (meist lineare) Gleichung der Form $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$ verbunden. Y bestimmt die Klassifizierung und β_j mit $j \in \{1, ...p\}$ sind entsprechende Koeffizienten, die die Prädiktoren X_j modifizieren. β_0 ist ein von den Prädiktoren unabhängiger Koeffizient. [?]

Die einzelnen Prädiktoren des Modells können kombiniert und zu Polynomen verbunden werden. So ist es auch möglich, quadratische, kubische oder höhere polynomielle Funktionen in linearen Modellen umzusetzen. [?] Anhand der erstellten Gleichung lassen sich durch die Koeffizienten die Gewichtungen der Feature oder Featurekombinationen, die die Prädikatoren bilden, ablesen. [?] Diese Gewichtungen können beispielsweise durch ein Balkendiagramm visualisiert werden.

Um eine gute Interpretierbarkeit zu gewährlreisten sollten sowohl die Anzahl verwendeten Feature als auch die Anzahl der Kombinationen und somit der Höhe des Funktionsgrades möglichst gering gehalten werden. [?, ?]

Theorie

Für die Umsetzung eines linearen Modells wird Logistische Regression zur Klassifizierung verwendet. Auch für diese Umsetzung bilden die vorhandenen sklearn Implementierungen die Grundlage.

Code

3.2 Counterfactuals

3.2.1 Fat Forensics Counterfactuals

CF umschreiben! (??) Counterfactuals bieten dem Nutzer textuelle, beispielhafte Angaben zur Auswirkung bestimmter Merkmale. Sie beschreiben nach gewissen Kriterien minimale Änderungen, die zu einer anderen Klassifizierung führen würden und stellen somit nächste mögliche Welten dar, in denen eine Entscheidung anders ausgefallen wäre. [19] Sie geben somit eine Erklärung für eine Klassifizierung an und können ein hilfreiches Werkzeug sein, um schlechte Trainingsdaten, die einen gewissen Bias haben, zu erkennen. [19]

Für diese Arbeit werden Counterfactuals verwendet, um nächste mögliche Welten als Erklärungen für Klassifizierungen anzugeben.

Für die Verwendung von Counterfactuals als Erklärung kann die Implementierung von FAT Forensics genutzt werden. [**?**] [1] [2] Diese stellt Funktionen zur Errechnung und zu textuellen Ausgabe der bestimmten Counterfactuals bereit. So können dem Nutzer passende Erklärungen generiert und präsentiert werden.

Theorie

Code

3.2.2 DiCE

Dice umschreiben! D (**??**)iCE (Diverse Counterfactual Explanations) ist ein Ansatz, der darauf fokussiert ist, möglichst diverse Counterfactuals zu erzeugen, die trotzdem möglichst genau lokale Entscheidungsgrenzen bestimmen. [14] Um dies zu erreichen wird neben der Diversität auch die Nähe zum Input in das Optimierungsproblem mit einbegriffen.

Der größte Unterschied zur Generierung einfacher Counterfactuals ergibt sich aus der Erstellung einer Menge von möglichen Erklärungen [14], statt nur einer einzigen. [11] So kann die Diversität für das Set sichergestellt und optimiert werden. [14]

Für die Implementierung von DiCE wird eine zu den Papern [14, 11] zugehörige Bibliothek von verwendet.[3] Diese stellt den benötigten Sourcecode bereit und kann ebenfalls auf neuronalen Netzen verwendet werden.

Theorie

Code

3.3 Feature Contribution

3.3.1 DeepSHAP

DeepSHAP bissl umschreiben! B (**??**)ei dem SHAP (SHapley Additive exPlanations) Deep Explainer handelt es sich um einer abgewandelte und modifizierte Version des DeepLIFT Algorithmus. [4]

DeepLIFT erklärt die Beteiligung einzelner Neuronen, gemessen an ihren Inputs. Um diesen Wert errechnen zu können werden für jeden Knoten Attributregeln aufgestellt, die sich aus der Aktivierung auf dem Referenzinput ergeben. Mithilfe dieser Regeln und der

[1] `https://fat-forensics.org/generated/fatf.transparency.predictions.counterfactuals.CounterfactualExplainer.html#fatf.transparency.predictions.counterfactuals.CounterfactualExplainer`

[2] `https://github.com/fat-forensics/fat-forensics`

[3] `https://github.com/interpretml/dice`

[4] `https://shap.readthedocs.io/en/latest/generated/shap.DeepExplainer.html#shap.DeepExplainer`

Verbindung mit der sogenannten Kettenregel für Multiplizierer [?] lassen sich die Beteiligungen mittels Backpropagation bestimmen. [?]

Diese Berechnungen lassen sich deutlich beschleuningen, wenn nur Annäherungen von Shapley Values für die DeepLIFT Attribut-Regeln bestimmt werden. Der Deep Explainer nutzt aus, dass diese Approximation aus den DeepLIFT-Regeln errechnet werden kann. Die für jeden Knoten bestimmten Attribut-Regeln können dabei so gewählt werden, dass sie eine Abschätzung bilden. Durch das Integrieren über viele Hintergrundbeispiele wird eine Annäherung ermittelt, sodass die Summe der SHAP Values die Differenz zwischen dem erwarteten ($E(f(x))$)und dem aktuellen ($f(x)$) Output des Modells ergeben: $f(x) - E(f(x))$. [10]

Shapley Values sind ein Lösungskonzept, das Shapley für die Game Theory entwickelt hat. Bei diesem werden alle Feature als „Spieler“, die Vorhersagen oder Klassifizierungen als „Gewinn“angesehen. Aus dem Zusammenspiel der einzelnen Variablen kann dann die Beteiligung am Output errechnet und somit eine Erklärung für Klassifizierungen bestimmt werden. [?] Dabei stellt die Berechnug der Shapley Values ein NP-vollständiges Problem dar, das durch die Verwendung von Approximationen der eigentlichen Values polinomiell und somit effizient lösbar wird. [?]

Für die Umsetzung von Deep Explainer kann die SHAP Bibliothek verwendet werden. [5] Diese stellt die Impementation eines Algorithmus bereit, der SHAP auf neuronalen Netzen anwendet und der für die Erklärung von Klassifizierungen genutzt werden kann. [6]

Theorie

Code

3.3.2 LRP

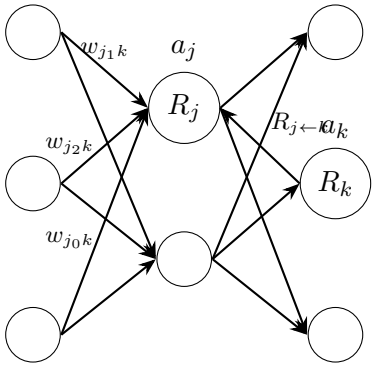
LRP umschreiben! B (??)ei LRP handelt es sich um einen Algorithmus, der Neuronales Netz rückwärts, also entgegen der Lern- und Klassifizierungsrichtung durchläuft und anhand von gelernten Gewichten und Aktivierungsfunktionen eine Erklärung des Inputs erzeugt. Für Bilddaten ist dies die Erstellung einer Heatmap, die farblich kennzeichnet, welche Wichtigkeit bestimmte Pixel für die Klassifizierung hatten. Dies entspricht der allgemeinen Grundidee der Feature Importance zur Erklärung, Gewichte pro Eingabedimension auszugeben. Diese Gewichte geben für jeden konkreten Datenpunkt an, wie stark die entsprechende Belegung für bzw. gegen die Entscheidung des Modells spricht. [13] [7]

LRP wird anhand eines Tutorials selbst implementiert. Bei diesem handelt es sich um das zum Paper [13] Zugehörige, in dem der notwendige Code erläutert wird.

[5] `https://shap.readthedocs.io/en/latest/index.html`
[6] `https://shap.readthedocs.io/en/latest/generated/shap.DeepExplainer.html`
[7] `http://heatmapping.org/tutorial/`

Theorie

Für die Verwendung von LRP werden im Folgenden Relevanzneuronen erstellt und alle Relevanzwerte entgegen der Lernrichtung durch das Netz propagiert. Diese Relevanzneuronen sind mit R_k bezeichnet und repräsentieren in der Output-Schicht zunächst den tatsächlichen Neuronen-Output a_k. [**?**]



Code

3.4 Taxanomie

Metadaten sammeln

Kapitel 4

Webpage

Kapitel 5

Ergebnisse

5.1

5.2 Todos

- Einleitung: Motivation, Ziele Einleitung: Ziele umschreiben
- Was passiert in dieser Arbeit? Einleitung: Vorgehen erläutern Kapitel 2 benennen
- Erläuterung der Ansätze (Oberthemen; Einführung in das Kapitel) Kapitel 2: Grundlagen Kapitel 2: NN Aufbau **To do** Kapitel 2: Backpropagation (**??**) Kapitel 2: Ansatzerklärungen übernehmen Kapitel 2: Verbindung zu Blackbox öffnenden Ansätzen herstellen
  - Surrogat Kapitel 2.1: Surrogat detaillierter
  - Counterfactual Kapitel 2.2: Counterfactual detaillierter
  - Feature Contribution Kapitel 2.3: Feature Contribution detaillierter
- Erklärung im speziellen: Surrogat Kapitel 3.1: Surrogatmodell anlegen
  - DT Theorie Kapitel 3.1.1: DT übernehmen Kapitel 3.1.1: DT verwendete Formeln erklären **To do** Kapitel 3.1.1: Grafiken ergänzen (**??**)
  - DT Implementierung –> selfmade **To do** Kapitel 3.1.2: Implementierung beenden (**??**) **To do** Kapitel 3.1.2: Code erlä
  - Lineares Modell Theorie (Optionen?) **To do** Kapitel 3.1.3: Lineares Modell übernehmen (**??**)
    To do Kapitel 3.1.3: verwendete Umsetzung und Formeln erklären (**??**) **To do** Kapitel 3.1.3: Grafiken ergänzen (**??**)
  - Lineares Modell Implementierung –> selfmade **To do** Kapitel 3.1.4: Implementierung beenden (**??**)
    To do Kapitel 3.1.4: Code erläutern (**??**)
- Erklärung im speziellen Counterfactual **To do** Kapitel 3.2: Counterfactual anlegen (**??**)
  - Coutnerfactuals FAT-Forensics Theorie **To do** Kapitel 3.2.1: Counterfactuals übernehmen (**??**)
    To do Kapitel 3.2.1: Counterfactuals verwendete Formeln (+ brute force statt Loss function) erklären (**??**)
    To do Kapitel 3.2.1: Grafiken ergänzen (**??**)
  - Coutnerfactuals FAT-Forensics Implementierung –> Übernahme und Verwendung **To do** Kapitel 3.2.2: Implementierung beenden (**??**) **To do** Kapitel 3.2.2: Code erläutern (**??**)
  - DiCE Theorie **To do** Kapitel 3.2.3: DiCE übernehmen (**??**) **To do** Kapitel 3.2.3: DiCE verwendete Formeln erklären (**??**)
  - DiCE Implementierung –> Übernahme und Verwendung **To do** Kapitel 3.2.4: Implementierung beenden (**??**)
    To do Kapitel 3.2.4: Code erläutern (**??**)
- Erklärung im speziellen Feature Contribution **To do** Kapitel 3.3: Feature Contribution anlegen (**??**)
  - DeepSHAP Theorie **To do** Kapitel 3.3.1: Shap, 'DeepLIFT übernehmen (**??**) **To do** Kapitel 3.3.1: Shap, DeepLIFT, Deep
    To do Kapitel 3.3.1: Grafiken ergänzen (**??**)
  - DeepSHAP Implementierung –> Übernahme und Verwendung **To do** Kapitel 3.3.2: Implementierung beenden
    To do Kapitel 3.3.2: Code erläutern (**??**)

  - LRP Theorie **To do** Kapitel 3.3.3: LRP übernehmen (Proposal & Proseminar) (**??**) **To do** Kapitel 3.3.3: LRP Form
    To do Kapitel 3.3.3: Grafiken ergänzen (**??**)
  - LRP Implementierung –> selfmade **To do** Kapitel 3.3.4: Implementierung beenden (**??**)
    To do Kapitel 3.3.4: Code erläutern (**??**)

- Metadaten **To do** Kapitel 3.4: Metadaten ergänzen (**??**)
- Webpage **To do** Kapitel 4: Webpage Ein- und Überleitung, Begründung für Nutzung (**??**)
  - Design Idee **To do** Kapitel 4: Webpage entwerfen (**??**) **To do** Kapitel 4: Webpage einbauen (**??**)
    To do Kapitel 4: Webpage erläutern (**??**)
  - Theorie, verwendete Grundlagen **To do** Kapitel 4: verwendete Code Basis / Theorie erkläutern (**??**)
  - Implementierung / Umsetzung **To do** Kapitel 4: eigenen Code erklären (**??**)
- Ergebnis **To do** Kapitel 5: Ergebnis formulieren (**??**)
- Fazit **To do** Kapitel 5: Fazit (und Ausblick ?) (**??**)

  To do Allgemein: Referenzen sortieren, aufhübschen und einbinden (**??**)
- Abbildungen
- Algorithmen
- Code
- Literatur

Anhang A

Weitere Informationen

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

-DE *Welche Arten von Maschinellem Lernen gibt es?*, Januar 2021. *Wie lernen Maschinen? - Grundlegende Begriffe - ML-Blog • Grundlagen*, Januar 2021. Agarap, Abien Fred: *Deep Learning using Rectified Linear Units (ReLU).* arXiv:1803.08375 [cs, stat], Februar 2019. Alabassy, B., M. Safar und M. W. El-Kharashi: *A High-Accuracy Implementation for Softmax Layer in Deep Neural Networks.* In: *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, Seiten 1–6, April 2020. Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller und Wojciech Samek: *On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation.* PLOS ONE, 10(7), Juli 2015. Choe, Junsuk und Hyunjung Shim: *Attention-Based Dropout Layer for Weakly Supervised Object Localization.* In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seiten 2214–2223, Long Beach, CA, USA, Juni 2019. IEEE. Cramer, E. und U. Kamps: *Grundlagen der wahrscheinlichkeitsrechnung und statistik: Ein skript für studierende der informatik, der ingenieur- und wirtschaftswissenschaften.* Springer-lehrbuch. Springer Berlin Heidelberg, 2014. Du, Gaoming, Chao Tian, Zhenmin Li, Duoli Zhang, Yongsheng Yin und Yiming Ouyang: *Efficient Softmax Hardware Architecture for Deep Neural Networks.* In: *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, Seiten 75–80, Tysons Corner VA USA, Mai 2019. ACM. Hu, R., B. Tian, S. Yin und S. Wei: *Efficient Hardware Architecture of Softmax Layer in Deep Neural Network.* In: *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, Seiten 1–5, November 2018. Lundberg, Scott M. und Su-In Lee: *A Unified Approach to Interpreting Model Predictions.* Advances in Neural Information Processing Systems, 30:4765–4774, 2017. Mahajan, Divyat, Chenhao Tan und Amit Sharma: *Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers.* arXiv:1912.03277 [cs, stat], Juni 2020. Mohri, Mehryar, Afshin Rostamizadeh und Ameet Talwalkar: *Foundations of Machine Learning, second edition.* MIT Press, Dezember 2018. Montavon, Grégoire, Wojciech Samek und Klaus-Robert Müller: *Methods for interpreting and understanding deep neural networks.* Digital Signal Processing, 73:1–15, Februar 2018. Mothilal, Ramaravind Kommiya, Amit Sharma und Chenhao Tan: *Explaining Machine Learning Classifiers through Diverse Coun-*

terfactual Explanations. arXiv:1905.07697 [cs, stat], Dezember 2019. Naqa, Issam El, Ruijiang Li und Martin J. Murphy: *Machine Learning in Radiation Oncology: Theory and Applications.* Springer, Juni 2015. Rajaguru, Harikumar und Sunil Kumar Prabhakar: *E-health Design with Spectral Analysis, Linear Layer Neural Networks and Adaboost Classifier for Epilepsy Classification from EEG Signals.* In: Hemanth, D. Jude und S. Smys (Herausgeber): *Computational Vision and Bio Inspired Computing*, Lecture Notes in Computational Vision and Biomechanics, Seiten 634–640, Cham, 2018. Springer International Publishing. Sachs, Michael: *Wahrscheinlichkeitsrechnung und Statistik: für Ingenieurstudierende an Hochschulen: mit 35 Bildern, 93 Beispielen und 71 Aufgaben.* Mathematik-Studienhilfen. Fachbuchverlag Leipzig im Carl Hanser Verlag, München, 5. Auflage Auflage, 2018. Sazli, Murat H: *A BRIEF REVIEW OF FEED-FORWARD NEURAL NETWORKS.* Seite 7. Wachter, Sandra, Brent Mittelstadt und Chris Russell: *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR.* SSRN Electronic Journal, März 2018. Wurm, Christian: *Neuronale Netze und Tiefe Architekturen.* Seite 139. Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do und Kaori Togashi: *Convolutional neural networks: an overview and application in radiology.* Insights into Imaging, 9(4):611–629, August 2018.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 22. Februar 2021

Lisa Salewsky