Fragen zu künstlichen Neuronalen Netzen (KNN)

1. Welches Python package/ Module kann für KNN genutzt werden?

> Tensorflow/ Keras.

2. Definiere Netztopologien. Gib Beispiele (3)

> Struktur oder Layout neuronaler Netze.
Die Art und Weise wie Schichten miteinander verbunden sind, beschreibt die Netztopologie.
Bsp.
Forward Netzwerke
Recurrent Neuronale Netze
Convolutional Neural networks

3. Was ist ein KNN?

> Maschinelles Lernverfahren
> Supervised und unsupervised
> Klassifikation und Regression.
> Ansammlung an künstlichen Neuronen

4. Wie lernt ein KNN?

Lernt durch Anpassen von Gewichten zwischen Neuronen innerhalb des Netzes.
> Passt die Gewichte während Trainingsschritt an.
> **Lernt selbständig** an Daten.

5. Seit wann gibt es die Idee und seit wann sind KNN in der Realanwendung?

> Idee. 1943: McCulloh und Pitts beschreiben Informationsverarbeitung von biologischen Neuronen
> Anwendung seit ca. 2010.

6. Worauf basiert die Idee von KNN?

>Eine Abstraktion von Informationsverarbeitung bei biologischen Neuronen

7. Wann sind KNN gut anwendbar?

Anwendbar auf explizit schwer modellierbare Probleme.
> Das heißt, es ist gut anwendbar, wenn man schlecht Regeln für Modellierung verwenden kann.

8. Was sind in der Regel die Anpassbaren Elemente eines KNN?

Die Gewichtung (vor allem Am Anfang).

9. Aus welchen zwei wichtigen Teilen besteht ein Neuron?

> Gewichtung
>Aktivierungsfunktion

10. Definiere „Gewichtung“ bei KNN.

Grad des Einflusses der eingehenden Signale.
> Anpassbar.

11. Was dient als Analogon zu Dendriten in KNN?

> Ein KNN hat Inputs. Der Input ist ein empfangener Reiz.

12. Was dient als Axonhügel Analogon bei KNN und was macht es?

> Übertragungsfunktion
> Berechnet die Netzeingabe
> Inputs werden verrechnet über die Übertragungsfunktion.
> In der Regel die Summe der Gewichte wird benutzt.

13. Was dient als Analogon zum Aktivierungspotential bei KNN? Was macht es? Welche Funktion kann dafür verwendet werden?

> Aktivierungsfunktion
> Nimmt die Netzeingabe aus der Übertragungsfunktion und berechnet die Netzausgabe.
> Sigmoid-Funktion

14. Was dient als Constraint? Was ist die Analogie bei biologischen Neuronen?

> Schwellenwerte
> Wird von der Aktivierung abgezogen
> Bildet constraints, um anzugeben, wann Neuronen arbeiten/feuern sollen.

15. Wie sieht die Aufbauphase eines KNN aus? 3 Elemente

> Baue Topologie aus
> Entscheide, wieviele Neuronen und Schichten nötig sind.
> Wähle Aktivierungsfunktion.

16. Welche 3 Möglichkeiten des Trainings gibt es bei KNN?

> Supervised Learning
> Unsupervised Learning
> Reinforcement Learning

17. Was macht das Training bei KNN? Wie wird das Training initialisiert und wie werden Sie angepasst?

> Bgringt das Netz dazu aus den Eingaben die passenden Ausgaben zu generieren.
> Gewichte zufällig initialisiert
> Dann mittels Trial and Error angepasst, bis Klassifikationsgüte gut genug ist.

18. Was ist der Vorteil der gewählten Initialisierung?

In der Regel werden Gewichte zufällig initialisiert, weil gezeigt wurde, dass Random Initialisierung Trainingserfolg erhöht + man bleibt nicht in lokalen Minima hängen.

19. Was bedeutet ein Gewicht von 0?

0 bedeutet, dass man die Kante, die 2 Neuronen verbinden sollte, löscht.

20. Werden Neuronen gelöscht oder erzeugt beim Training?

In der Regel werden Neuronen nicht erzeugt oder gelöscht. Nur ihre Gewichte werden angepasst.

21. Welche Form hat das Ausgabeneuron?

> Kann eine Klasse sein (Klassifikation)
> Oder eine Funktion (Regression).

22. Was ist eine Epoche und in welchen Schritt finden wir es?

> Epoche: Komplette Iteration über gesamten Datensatz.
> Während des Trainings durchläuft KNN den Datensatz (Input) mehrmals.
> 100 Epochen während also 100 Trainingsiterationen.

23.Wie funktioniert Training beim überwachten Lernen?

> Eingabe und Ausgabe sind bei Supervised Learning bekannt.
> Man trainiert Modell über mehrere Epochen
> Vergleicht Ausgabe des Netzes jedes mal mit echten Lösungen/Labeln.
> Gewicht von jedem Input wird bei jeder Iteration/Epoche angepasst.
Wichtig: Iterative Anpassung der Gewichte.
Benötigt Aktivierungsmuster der Ausgabeneuronen.

24. KNN gibt doch die 100 % korrekte Lösung an, oder?

Nein. Keine exakte Lösung. Nur eine Annäherung
> gibt KNN eine Fehlertoleranz.

25. Wozu dient die Percepton-Lernregel/ Delta-Regel? Wann ist es anwendbar? Wie funktioniert es?

> Ein Verfahren zur Anpassung der Gewichte in einem Perceptron
> Algorithmus zum Training der Gewichte in der simpelsten Form des KNN.

Annahmen:
> Nur Supervised Learning, da Vergleich zu Ground Truth nötig ist.
> nur für linear separierbare Daten (siehe SVM)
> Maxover approximiert bei nicht linear separierbaren Daten
> Perceptron: Einfachte KNN, die nur Eingabe und Ausgabeneuronen haben, aber keine Hidden Layer. → Man hat also nur eine trainierbare Schicht.

Funktion:
Verändert Gewichte entsprechend dem Beitrag der Gewichte zum Netzfehler.
> Je schlimmer der Fehler eines Neurons, desto stärker wird es angepasst.
> Anpassung des Neurons davon abhängig, wie sehr er zum Fehler beiträgt.
Netzfehler: Differenz zwischen tatsächlichen Neuron (bekannt, da supervised Learning) und estimated Neuronwert.
> **Iterativ**, bis gewünschte Verhalten erreicht ist.

26. Was ist die Formel für Delta-Regel? Wie erfolgt die Berechnung der Gewichte? Was gibt die Lernrate an?

Neues Gewicht = Altes Gewicht (am Anfang zufällig initialisiert) + Lernrate * error * Input

Lernrate: Gibt an, wie stark Gewichtung im Netz bei jeder Aktualisierung verändert werden soll.

1. Berechne Initiale Netz über Summenfunktion/ Übertragungsfunktion

2.Aktivierungsfunktion nutzen
3. Berechne Error → Echter Label – berechneter Label aus Schritt 1.
4. In Formel für neues Gewicht einsetzen und für jedes Inputneuron ein neues Gewicht berechnen
5. Iterieren.

27. Was ist die Verallgemeinerung der Delta-Regel für mehrschichtige Netze? Wie funktioniert es? Welches Verfahren nutzt es und wie funktioniert es?

> Backpropagation: Man beginnt bei der Ausgangsschicht und geht rückwärts zur Eingabeschicht und ändert dabei Gewichte.
> Verallgemeinerung der Delta-Regel
> Für mehrschichtige Netze
> Schichtenweise Veränderung der Gewichte angefangen beim Ausgangsneuron.
> Fehlöer der Ausgabe eines Neurons ist eien Funktion der Gewichte aller eingehenden Ntzverbindungen.
> **Gradientenabstiegverfahren**: Suche Gewicht, damit Fehlerfunktion minimiert wird.
Optimierungsregel heiß also: Suche alle Gewichte, so dass die Fehlerfunktion minimiert wird.

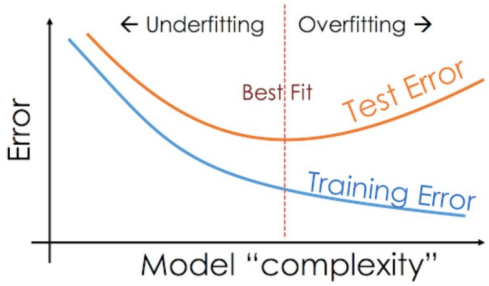
28. Was ist ein Gradient und Gradientenabstieg? Welche Aufgabe hat hier die Lernrate? Wieso muss ich die richtige Lernrate wählen. Was ist das Ziel des Gradientenabstiegverfahrens?

> Gradient: Man hat eine hyperdimensionale Funktion mit Höhen und Tiefen.
> Die Achsen des Hyperdimensionalen Raums ist gegeben durch die Gewichte und die Höhe durch die Fehlerrate.
> Funktion wird durch Gewichte beeinflusst
> Gradient gibt an, wo etwas steigt oder fällt. Wo muss ich mich hinbewegen im Raum, um ein Minimum (minimaler Fehler) zu erreichen?

Lernrate: Schrittweise entlang des Gradienten durch Lernrate definiert.
> Wir wollen Gradientennetz schnell absuchen, um globale Minima zu finden und die Lernrate gibt an, wie groß die Schritte entlang des Netzes sind.
> Muss es richtig wählen, da ich bei zu großen Schritten das Minimum überspringen könnte.

Ziel: ideale Gewicht mit dem kleinsten Fehler finden, indem man das globale Minimum im Gradienten findet.

29. Wie kann man basierend auf einem Plot, der die Loss-Funktion gegen die Epochen mit Training und Validation-Daten erkennen, ob es sich um Overfitting/ Underfitting beim KNN-Modell handelt? Wie sollte die Linie verlaufen (idealerweise)?



Overfitting:
- Wenn Training deutlich unter Validation/test liegt (siehe Abbildung).
Underfitting:
- Wenn Test/Validation unter Training fällt.
- Hoher Error.

Schlecht, wenn:
Training und Test Linien **auseinanderlaufen (Testlinie steigt, während Error sinkt)**.
Wenn beide **stagnieren/konstant** bleiben (aber ok, wenn die Accuracy bereits sehr hoch ist).

30. Muss ich bei KNN auf die Evaluation warten, bis KNN komplett durchläuft?

Nein: Ich kann schon während des Trainings sehen, ob Training erfolgreich ist.
>Kann pro Iteration Fehler bestimmen.
> Kann nach jeder Epoche Performancemetriken rausgeben lassen.
Ergebnisse in „History“ Objekt gespeichert.

31. Wie erfolgt Reinforcement Learning bei KNN? Welches Verfahren gibt es? Wie effinzient ist es in vergleich mit Supervised Learning?

Man teilt dem Netz nach der Klassifikation mit, ob es richtig oder falsch lag (mit z.B. Scores).
> Q-Learning.
> Weniger effizient.
> Braucht kein klares Label (Vorteil).

32. Wie erfolgt unsupervised Learning? Welche Lernregel/ Algorithmus gibt es?

> Nur Eingabe bekannt, nicht Ausgabe.
> Netz versucht selbst Muster zu finden.
> Hebbsche Lernregel: What fires together, wires together.
> Autoencoder.

33. Was sind Feed Forward Netze? Erkläre es mit eigenen Worten? Wie funktioniert es? Welche Spezialform gibt es (erkläre es)?

> Eine Netztopologie
> Simpelster Fall eines Netzes.
> Die Verbindungen zeigen Richtung Ausgangsneuron/Schicht.
> Eine Schicht ist immer nur mit einer der nächsten Schichten verbunden
Man hat also eine klare Richtung.
> Kann aber auch Schichten überspringen.

Sonderform: **Fully connected FF-Netze** → Dense-Netze, wo Neuronen einer Schicht mit allen Neuronen der Folgenden Schicht verbunden sind.

34. Wie viele Inputs gibt es im Imputlayer und wie viele Outputneuronen gibt es?

Input: Anzahl Inputs enspricht Anzahl an Variablen.
Output: Enspricht Anzahl der Vorhersagen/Klassen.

35. Was ist Deep Leaning? Wodurch unterscheidet es sich von anderen ML-Ansätzen

> Teilgebiet des Machine Learnings.
> Verwendet KNN
> Name kommt daher, dass man vielschichtig arbeitet bei den KNN.
> Unterschied: Kein Feature Engineering notwendig.

36. Obwohl Perceptrons schon im letzten Jahrhundert existiert haben, gibt es Deep Learning erst seit ca. 2009. Welche 2 wichtigen Beiträge haben die heutige Nutzung von DL ermöglicht?

> NVIDIA GPUs mit Tensorcores, die hochparalleles Arbeiten ermöglichen, wurden 2009 veröffentlicht.
> Zugriff auf Big Data.

37. Welche 2 Klassen an Deep Learning Modellen unterscheidet man + erkläre diese? Kenne zu beiden Beispiele

Discriminative Models:
Schließen von beobachteten Variablen auf unbeobachtete Variablen.
> Klassifikation und Regression
> Recurrent Neural Networks
> Convolutional Neural Networks

Generative Models:
Kann auch die beobachteten Werte erstellen (Deep Fakes).
Oft zwei Netze benutzt, wobei erstes Netz Output generiert und zweites Netz Output evaluiert.
Besonderheit: Beide Netze zusammen traniert.
> Generative Adversarial Networks
> Restricted Bolzmann Machines

38. Was ist RNN (Wofür steht die Abkürzung)? Welches Problem löst es? Zum welchem der zwei Modelle gehört es? Können FFN mit EKG Daten arbeiten? Was erlaubt RNN?

> Recurrent neural Networks
> Discriminative Networks
>Löst Problem bei Feed Forward Netzen.
Problem von FFN: Nutzt fixe Anzahl an Eingabeneuronen, hat nur eine Richtung und hat eine fixe Anzahl an Layern.

EKG: Muss EKG Fenstern für FFN oder ich nutze RNN.

Erlaubt die Verarbeitung von Sequenzen (wie EKG-Daten, Text, Genomdaten…)
> Kann **zeitlich codierte Informationen erkennen**.
Haben ein „Gedächtnis“: Rückgerichtete Schleifen

39. Was ist das Vanishing Gradient Problem? Wodurch wird es verursacht und wie löst man es?

Wenn man Sigmoid-Funktion nutzt, werden KNN-Modelle mit steigender Anzahl an Schichten schlechter, weil der Fehlergradient am Anfang bereits kleins ein kann.
> Verursacht durch Sigmoid
> Wird schlimmer mit jeder Schicht.
> Sigmoid reicht zwischen 0 und 1. Bei einem Wert von z.b. 0.985 gibt es nicht mehr viel Spielraum bis zur 1. Man beobachtet also **sehr kleine Gewichtsveränderungen**.
KNN kann längere Zusammenhänge nicht mehr erlenen.
Lösung:
> Statt Sigmoid-Funktion als Aktivierungsfunktion nutzt man ReLU (Rectified Linear Unit)
> Alle negativen Werte werden 0 gesetzt und alle positiven Werte verlaufen linear ins unendliche.

40. Welches Problem verursacht die Lösung für das Vanishing Gradienten Problem und wie löst man es?

> ReLU verläuft ins unendliche.
> Große Werte dominieren also über kleine, wodurch diese nicht ins Gewicht fallen.
> Lösung: Gradient Clipping. Setze einen Maximalwert und schaue mir höhere Werte nicht an.

41. Wo werden RNN eingesetzt? 6 Beispiele.

> Natural Language Processing (NLP)
- Amazon Alexa, Siri
> Übersetzer
> Wortvorhersage bei Tastaturen
> Zeitreihenvorhersagen (Börse)
>Gensequenzen
> Clinical Pathway vorhersagen.

42. Wie funktionieren CNN (und wofür steht die Abkürzung)? Wofür verwendet?

> CNN = Convolutional Neural Network
> Verwendet für Bilderkennung

> Features werden durch Faltung extrahiert.
> In Bildanalyse haben wir Filter kennengelernt wie den Gauss-Filter für Blur. **CNNs erlernen die idealen Filter selbst**.

43. Was macht Pooling Layer bei CNN?

> **Pooling Layer**: Verkleinert die Dimension der Featuremap. Verkleinere also Bildmatrix, indem ich über das gesamte Bild gehe und mehrere Werte in einem Wert zusammenfasse (zb. Median, mean oder max nehmen).
> **Pooling wird nach Faltung durchgeführt**!

44. Wieso müssen wir Schichten stapeln (stacking Layers)?

> Mit jeder Schicht steigt der Abstaktionsgrad.
>ermöglicht eine schrittweise Erhöhung des Abstraktionsgrads der gelernten Merkmale.

45. Welche Form von Feed Forward Netzen wird bei CNN benötigt?

> Dense Netze/ Fully Connected FF-Netze.
Brauch es, um aus Schichten Muster/ Klassen zu lernen.

46. Gib 3 Beispiele, wo CNN benutzt werden?

> Bilderkennung
> Spracherkennung (text2speech)
> Alpha Go.

47. Was sind Generative Adversarial Networks? Zu welcher Klasse an Deep Learning Modellen gehören Sie? Wozu werden Sie benutzt? Was ist besonders an GAN?

> Modell: Generative Modelle.
> Verfahren zum unüberwachten Lernen.
> Besteht aus zwei neuronalen Netzen, die gegeneinander lernen.
Generator-Netz erstellt Kandidaten/output und Diskriminator bewertet Output.

48. Welches Modell des GAN kennen Sie? Wozu dienen Sie und wie funktionieren Sie? Wie kann ich es auf Fehler evaluieren?

> Autoencoder
> Wird genutzt, um effiziente codierungen zu lernen.
> Unsupervised.
> **Lernt Embedding** für spätere Dimensionsreduktion.
> Besteht aus Encoder und Decoder

Encoder:
Der Encoder ist dafür verantwortlich, die Eingabedaten in eine komprimierte oder codierte Darstellung zu transformieren. Es nimmt die Eingabedaten auf und reduziert die Dimensionalität, indem es wichtige Merkmale extrahiert.
> D.h. die Schichten werden immer kleiner mit jeder Schicht.

Decoder:
Der Decoder nimmt die codierte Darstellung, die vom Encoder erzeugt wurde, und rekonstruiert die ursprünglichen Eingabedaten. Ziel ist es, eine Rekonstruktion zu erstellen, die so nah wie möglich an den Originaldaten liegt.
> Wird also größer, bis es die Größe der Eingangsschicht am Anfang hat.

Kriege Rekonstruktionsfehler, indem ich Eingabe und Ausgabe vergleiche.