

- 1 Die Smartwatch kann zum Internet der Dinge gezählt werden, da sie **über das Internet mit anderen Geräten vernetzt ist und über Sensoren und Aktoren verfügt**. Über Sensoren wie **Beschleunigungs- und Herzfrequenzsensoren** erfasst die Smartwatch Daten, die sie weiterverarbeitet und an verbundene Geräte sendet. **Display und Lautsprecher** sind Aktoren der Smartwatch, über die auf erfasste oder empfangene Daten reagiert werden kann. 3
- 2 Auf einem selbst gehosteten Server ist vermutlich eher mit Ausfällen zu rechnen, weshalb hier der Server von LovelaceLab überlegen ist. Bei einem privaten Server ist unter bestimmten Umständen die Vertraulichkeit gefährdet, wenn es es zu einem erfolgreichen Angriff auf das Unternehmen kommt. 2
- 3a ?-herstellerVon(berlinbyte,Y) 1
- 3b Ergebnisse der Anfrage: 2
Y = panzerglas
Y = cpu
Y = funkmodul.
k1 besteht aus einem Bauteil, das wiederum aus dem Bauteil k2 besteht.
- 3c enthältBauteil(X,Y):- 3
bestehtAus(X,Y);
(bestehtAus(X,Z), enthältBauteil(Z,Y)).
- 3d hatZulieferer(X,Y):- 2
herstellerVon(X,TeilX),
herstellerVon(Y,TeilY),
enthältBauteil(TeilX,TeilY).
- 4a Der Hersteller sammelt die Sensordaten zum Zeitpunkt eines Sturzes und verknüpft diese mit der ausgewählten Option des Nutzers. Dadurch entsteht ein gelabelter Datensatz. Diese Daten können zum Trainieren und Testen mittels überwachten maschinellen Lernens verwendet werden. 2
- 4b Das obere Neuron der verdeckten Schicht erhält zwei Eingabewerte: 2
– Die Beschleunigung (15) multipliziert mit 20 = 300
– Die Drehgeschwindigkeit (100) multipliziert mit 4 = 400
Diese beiden Werte werden addiert: $300 + 400 = 700$
Danach werden von diesem Wert 600 abgezogen: $700 - 600 = 100$
 $100 \geq 0$, also feuert dieses Neuron
- Das untere Neuron der verdeckten Schicht erhält zwei Eingabewerte:
– Die Beschleunigung (15) multipliziert mit 12 = 180
– Die Drehgeschwindigkeit (100) multipliziert mit 30 = 3000
Diese beiden Werte werden addiert: $180 + 3000 = 3180$
Danach werden von diesem Wert 3000 abgezogen: $3180 - 3000 = 180$
 $180 \geq 0$, also feuert dieses Neuron
- Das Neuron der Ausgabeschicht erhält zwei Eingabewerte:
– Das Feuern des oberen Neurons (1) multipliziert mit 1 = 1
– Das Feuern des unteren Neurons (1) multipliziert mit 1 = 1
Diese beiden Werte werden addiert: $1 + 1 = 2$
Danach werden von diesem Wert 1,5 abgezogen: $2 - 1,5 = 0,5$
 $0,5 \geq 0$, also feuert dieses Neuron und liefert die Ausgabe 1
- 4c Der Schwellenwert momentan ist 1,5. Die Neuronen können die Werte 0, 1 oder 2 liefern. Für 0 oder 1 ist der Ausgabewert momentan negativ, für 2 positiv. Diese Eigenschaft muss erhalten bleiben, und das bleibt sie durch Schwellenwerte zwischen]1;2] 1

- 4d Die Eingabeschicht hat 4 Knoten (4 Eingabesensoren) 3
 Die verdeckte Schichten hat 9 Knoten (4 + 5)
 Die Ausgabeschicht hat 3 Knoten (3 Fälle, die entschieden werden müssen)
 Die Anzahl der Gewichte ist $4 \cdot 5 + 5 \cdot 4 + 4 \cdot 3 = 52$ (Jeder Knoten einer Schicht ist verbunden mit jedem Knoten der darauffolgenden Schicht)
 Die Anzahl der Schwellenwerte ist $5 + 4 + 3 = 12$ (alle Knoten außer Eingabeknoten)
- 4e Ein neuronales Netz mit mehr verdeckten Schichten und mehr Neuronen kann sich 2
 tendenziell besser an komplexe Muster anpassen, jedoch kann dadurch auch zu overfitting kommen, also einer zu starken Anpassung an die Trainingsdaten. Es generalisiert dann schlechter bei neuen, unbekannten Daten. Gerade bei wenigen Trainingsdaten ist die Gefahr von overfitting groß. Die Aussage ist deshalb nicht allgemeingültig.
- 4f 1. Forward Propagation: 4
 Für jedes Element der Trainingsdaten wird die Ausgabe des Netzes berechnet
2. Kostenfunktion
 Mittels einer Kostenfunktion wird die Abweichung der berechneten Ausgabewerte von den jeweiligen Labeln der Eingabewerte (also den tatsächlichen Zielwerten) berechnet (Fehler).
3. Fehlerrückführung
 Der Wert der Kostenfunktion, und damit gleichzeitig der Fehler, soll minimiert werden. Das geschieht indem ausgehend von der Ausgabeschicht die Schwellenwerte und Gewichte der Schichten verändert werden. Diese Anpassung wird nach dem Einfluss der Gewichte und Schwellenwerte auf den Fehler durchgeführt.
- Die Schritte 1-3 werden z.B. für eine gewisse Zahl an Durchläufen ausgeführt
- 4g Ein gewisser Anteil der Daten sollte nicht zum Trainieren verwendet werden, um damit das 1
 neuronale Netz testen zu können, um zu schauen, wie gut das Netz generalisiert (also für unbekannte Daten funktioniert). Wenn die Daten bereits im Training verwendet wurden ist klar, dass das Netz für sie gut funktioniert, und es könnte keine Aussage darüber getroffen werden, wie gut das Netz generalisiert.
- 5a K-Means ist ein Beispiel für **unsupervised learning**, da er noch unbekannte Muster in 2
 Rohdaten erkennen soll (ohne vorheriges Label).
- 5b Durch Pythagoras berechnet sich der Abstand zu den Zentren: 3
 $|PZ_1| = \sqrt{(1^2 + 1^2)} \approx 1,41$ $|PZ_2| = \sqrt{(0,83^2 + 1,12^2)} \approx 1,39$
 Aufgrund des geringeren Abstandes gehört P zu Cluster 2.
- 5c Da der neue Punkt N deutlich weiter von allen anderen Punkten entfernt ist als diese 3
 untereinander, werden zwei Cluster entstehen. Eines enthält alle anderen Punkte und eines nur Punkt N.
 Bei der konkreten Berechnung wird N seinem Cluster zugeordnet. Alle anderen Punkte dem Cluster von Punkt P.
 Im zweiten Schritt werden die Zentren aktualisiert. N bleibt Zentrum, für alle anderen Punkte wird das Zentrum neu berechnet. In weiteren Schritten ändert sich nichts mehr, da sich die Zuordnung durch das neue Zentrum nicht ändert.
- 5d Python: 2

```
def clusterzuordnungAktualisieren(self):

    geändert = False

    for i in range(len(self.punkte)):
        bester_cluster = 0
        kleinster_abstand = self.abstandGeben(self.punkte[i],
self.zentren[0])

        for j in range(len(self.zentren)):
            abstand = self.abstandGeben(self.punkte[i],
self.zentren[j])
```

```
        if abstand < kleinster_abstand:
            kleinster_abstand = abstand
            bester_cluster = j

    if self.zuordnung[i] != bester_cluster:
        self.zuordnung[i] = bester_cluster
        geaendert = True

return geaendert
```

40

- 1a L_1 = Palindrome mit L änge von mindestens 2 und gerader Anzahl Buchstaben

3

Jedes mal, wenn ein Buchstabe hinzugefügt werden soll, geschieht dies zwei Mal; einmal am Anfang und einmal am Ende des Wortes. Da die Wortlänge nicht begrenzt ist, können theoretisch unendlich viele Buchstaben hinzugefügt werden.

Um ein Palindrom mit einem Automaten darzustellen, wird für jeden Buchstaben ein eigener Zustand benötigt, sonst kann man nicht gewährleisten, dass die Worthälften gespiegelt sind.

In L_1 können die Wörter theoretisch unendlich lang werden, also würden auch unendlich viele Zustände benötigt, um alle erlaubten Wörter darzustellen. Ein endlicher Automat darf

aber nur eine endliche Menge an Zuständen haben. Die Sprache L_1 kann also nicht durch einen endlichen Automaten akzeptiert werden und ist deshalb nicht regulär.

- 1b Eine Grammatik besteht aus Eingabealphabet Σ , einem Vokabular V , einem Startsymbol S

3

und einer Menge an Produktionsregeln P . Das 4-Tupel ist $G = (\Sigma, V, S, P)$.

Das Alphabet enthält alle Terminalsymbole, also Zeichen, die nicht weiter ersetzt werden:

$\Sigma = A; C; G; T$.

Das Vokabular besteht hier auch nur einem Nichtterminalsymbol, dem Startsymbol. $V = S$

Durch die Produktionsregeln muss immer ein Palindrom erzeugt werden. Das Startsymbol wird also so ersetzt, dass immer links und rechts der gleiche Buchstabe eingefügt wird, oder aber nur zweimal ein Buchstabe.

$P = S \rightarrow ASA; S \rightarrow CSC; S \rightarrow GSG; S \rightarrow TST; S \rightarrow AA; S \rightarrow CC; S \rightarrow GG; S \rightarrow TT$

- 1c

8

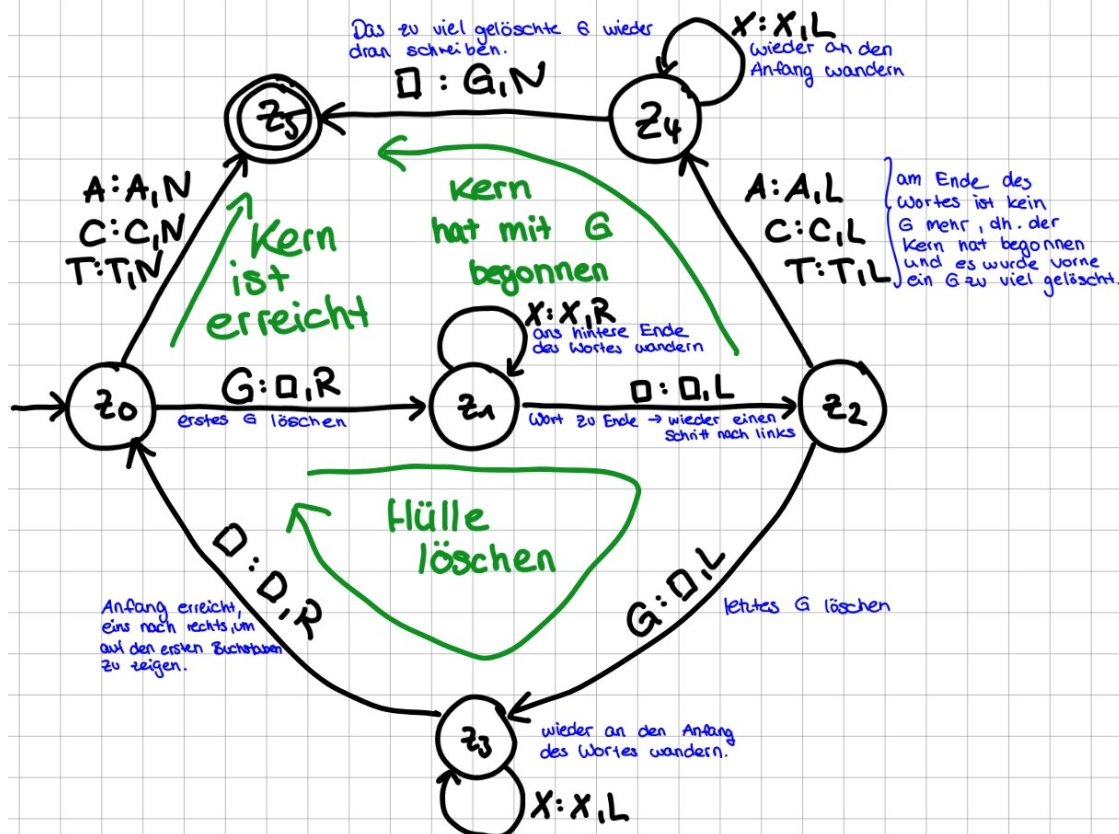


Bild einfügen

- 1d Der Übergang direkt vom Start- zum Endzustand, wenn einer der anderen Buchstaben eingelesen wird, muss entfernt werden.

4

Für jeden möglichen Buchstaben muss ein eigener Zyklus bestehend aus den Zuständen $z1X, z2X, z3X$ und $z4X$ (wobei X wieder für einen beliebigen Buchstaben steht) eingefügt

werden, der versucht den Buchstaben vorne und hinten am Wort zu entfernen. Jeder dieser Zyklen muss auch separat den Fall händeln, dass der zugehörige Buchstabe am Ende des Wortes nicht wieder auftaucht (also der Kern erreicht wurde). Dann muss die Turingmaschine, wie über z2 und z4, wieder an den Anfang des Wortes gehen und den Buchstaben, der gerade entfernt wurde, wieder hinschreiben.

2a Python: 2

```
def pell(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return 2 * pell(n-1) + pell(n-2)
```

Java:

```
int pell(int n){
    if ( n == 0){
        return 0;
    }
    else if ( n == 1){
        return 1;
    } else {
        return 2 * pell(n-1) + pell(n-2) ;
    }
}
```

2b $Pell(4) = 2 \cdot pell(3) + pell(2)$ 3

$$= 2 \cdot (2 \cdot pell(2) + pell(1)) + 2 \cdot pell(1) + pell(0)$$

$$= 2 \cdot (2 \cdot (2 \cdot pell(1)) + pell(0)) + 1) + 2 \cdot 1 + 0$$

$$= 2 \cdot (2 \cdot (2 \cdot 1 + 0) + 1) + 2 \cdot 1 + 0$$

$$= 12$$

Es sind 8 rekursive Aufrufe nötig!

2c Bei `anzahlAufrufeGeben(n)` wird bei beiden Abbruchbedingungen 0 zurückgegeben. 2

Ansonsten wird `anzahlAnrufeGeben(n-1) + anzahlAnrufeGeben(n-2) + 2` zurückgegeben.

2d Die Methode hat eine exponentielle Laufzeit, da bei jedem Aufruf ($n=0$ und $n=1$ 4

ausgeschlossen) zwei rekursive Aufrufe stattfinden.

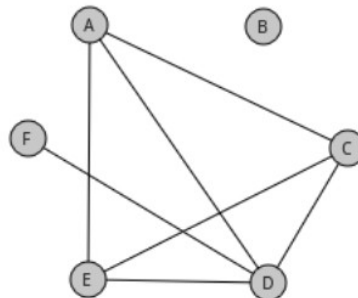
Würde man iterativ die n -te pell-Zahl berechnen, wäre die Laufzeit linear. Mit einer Zählwiederholung kann man ausgehend von den unteren beiden Pell-Zahlen die höheren Pell-Zahlen berechnen. Dafür benötigt man $n-1$ Durchläufe.

3a $S=\{A,D,C,E\}$. Im Sachzusammenhang ist das die größtmögliche Gruppe, in der niemand 2

miteinander befreundet ist.

3b Da in der stabilen Gruppe mit k Knoten keine 3

Knoten miteinander verbunden sind, sind im komplementären Graphen ebenfalls alle k Knoten miteinander verbunden und damit eine Clique.



3c Ein Problem liegt genau dann in NP, wenn eine gegebene Lösung für das Problem in 3

Polynomialzeit geprüft werden kann. Sei S eine solche Lösung, also angeblich eine stabile Menge. Es gibt $|S|$ über 2 Paare von Knoten in S , für die jeweils überprüft werden muss ob eine Kante zwischen ihnen existiert. Dass passiert in $O(n^2)$, da $n \geq |S|$. Damit ist es in Polynomialzeit prüfbar und damit in NP.

- 3d Wenn ein solcher Algorithmus existieren, dann könnte man damit auch das Cliquesproblem in polynomialer Zeit lösen (indem man den Graphen vorher invertiert). 3
Da das Cliquesproblem NP-vollständig ist, bedeutet eine polynomiale Lösung für dieses Problem, dass sich alle Probleme in der Klasse NP auf dieses Problem reduzieren und somit ebenfalls in polynomialer Zeit lösen ließen.
Damit würde $P = NP$ gelten.