Einleitung

Bei Algorithmen und Zufall Teil 1 haben wir Algorithmen kennengelernt und uns mit der Frage beschäftigt, wie Algorithmen mit Automaten realisiert werden können. Dabei haben wir gesehen, dass Algorithmen auch dazu genutzt werden können, Zufälle in Automaten abbilden zu können. Mit der Möglichkeit nichtdeterministische, also echte Zufallszahlen zu erzeugen wurde beispielhaft der Pfefferminzautomat zu einem stochastischen Automaten, der widrige Bedingungen wie Faustschläge von ungeduldigen Kindern, schlechte Witterungsbedingungen oder Wartungsbedarf abbilden kann.

Heute möchte ich gerne auf dem Gehörten aufbauen und ein bisschen tiefer in Computeralgorithmen einsteigen, die mit Zufällen umgehen.

Algorithmus

Eine Folge aus Arbeitsschritten die Kriterien erfüllen:

- Ausführbarkeit
- Eindeutigkeit
- Endlichkeit
- Speicherbarkeit
- Allgemeinheit
- Terminiertheit

Anhang (1)

Aber keine Angst, wenn Sie den ersten Vortrag zu diesem Thema versäumt haben: wo immer es hilfreich erscheint, werde ich wichtige Dinge in Erinnerung rufen und dazu, wie in diesem Beispiel zu den Eigenschaften des Algorithmus, graue Kästen auf der linken Seite einführen.

Die Basis allen Umgangs mit dem Zufall ist die Fähigkeit eines Algorithmus zufällige Ereignisse aufzunehmen, zu verarbeiten und darauf zu reagieren. Da dies einerseits in Interaktion mit einem Benutzer, der hohe Erwartungen an die Reaktionszeit hat, geschieht und andererseits z.B. in der Robotik auch autonom funktionieren soll, muss die Verarbeitung quasi in Echtzeit geschehen.

Ich lade Sie heute ein, einen grundlegenden Algorithmus zu verstehen der es erlaubt, mit uns Menschen zu interagieren und zu kommunizieren. Alle uns heute und in Zukunft noch weit vermehrt umgebenden „intelligenten“ Dinge benutzen ihn. Die Kenntnis dieses Algorithmus wird Ihnen helfen, das Internet der Dinge, die künstliche Intelligenz und die Robotik zu entmystifizieren und einen rationalen Zugang dazu zu gewinnen.

Die Rede ist von einem Algorithmus namens Scheduler.

1. Echtzeit Systeme – wie Algorithmen mit dem Zufall umgehen

Der Bedarf Realzeit Computeralgorithmen einzusetzen entstand Anfang der 1970 Jahre zunächst für die Simulation natürlicher Prozesse. Man begann für Simulationen über das Zeitverhalten von Algorithmen nachzudenken. Das Zeitverhalten sollte dem Zeitverhalten des simulierten Prozesses möglichst ähnlich sein und sich nicht an der Computertaktung orientieren. Zusammen mit dem gleichzeitig entstandenen militärischen Interesse an der Realzeitverarbeitung standen genügend Mittel zur Verfügung, um die ersten Realzeit Betriebssysteme auf den Markt zu bringen:

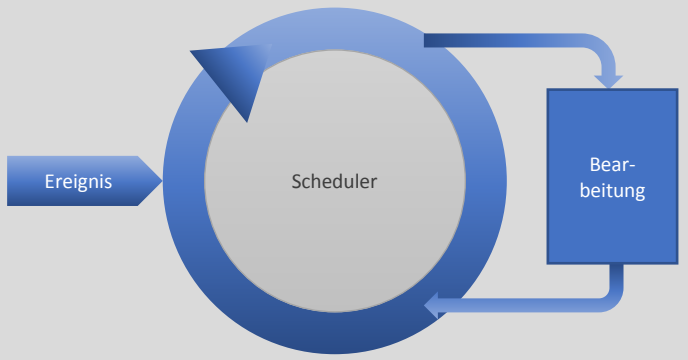
- RTOS (Real Time Operating System)
- RDOS (Realtime Disk Operating System) von Data General kam 1972 auf den Markt

Der Unterschied zwischen beiden Systemen liegt in der Pufferung von Prozess Daten. Ein RTOS System verarbeitet die Daten sofort, während ein RDOS System Zwischenspeicher nutzen kann. Generell werden heute RTOS Systeme für die sogenannten Embedded Systems, Computer ohne direkte Benutzerschnittstelle, eingesetzt, während alle Betriebssysteme mit Benutzerschnittstelle auf dem RDOS Ansatz aufgebaut sind.

1.2 Ein Realzeit Algorithmus (Realtime Kernel) – Zufälle mal 1 (physikalische Zufälle)

Das Ziel eines Realzeit Algorithmus ist es eingehende Daten aus zufällig eingetretenen Ereignissen innerhalb einer definierten Zeitspanne vollständig zu verarbeiten. Die definierte Zeitspanne wird durch die Anwendung definiert. Sie liegt in der Regel im Millisekunden-Bereich, kann aber auch bis in den Stunden-, oder Microsekunden-Bereich gehen. Immer besteht der Anspruch, keine Verzögerung zwischen der Verarbeitung der eingegangenen Daten und dem tatsächlichen Zeitverlauf wahrnehmen zu können; im Fachjargon heißt es sehr treffend „quasi simultan im Zeitmultiplex“.

Um dieses Ziel zu erreichen, wird ein Algorithmus benutzt, der wie ein geschlossener Kreis funktioniert und die Aufgabe hat, auf eingehende Ereignisse zu warten. Tritt ein solches Ereignis ein, wird der Kreis verlassen, um die Aufgabe zu bearbeiten, anschließend wird in den Kreis zurückgekehrt. Ist gerade kein Ereignis zu bearbeiten tut der Algorithmus nichts. Einen solchen Algorithmus nennt man Scheduler.



Dieses einfache Modell mit einem Ereignis und einem Bearbeitungs-Algorithmus (Task) unterscheidet sich kaum von einem Algorithmus, der nach dem Start nur einfach abläuft und dann stoppt. Der Sinn des Aufbaus erschließt sich aber, wenn viele Ereignisse zufällig eintreten, die verschiedene Bearbeitungsprozesse erfordern.

1.3 Eigenschaften eines Schedulers

Zwischen dem Eintreten eines Ereignisses und dem Ende der Bearbeitung darf nur eine kurze Verzögerungszeit entstehen. Die maximale Verzögerungszeit ist dabei für eine Aufgabe definiert. Wird die maximale Verzögerungszeit verfehlt, entsteht ein fataler Fehler, der zu einer entsprechenden Fehlerreaktion führt.

Eigenschaften eines Schedulers

- kurze Verzögerung
- Determiniertheit
- Unterbrechbarkeit
- Autonome Ausführung von hardwarenahen Tasks

Alle Tasks, die in einem Scheduler bearbeitet werden können, müssen in ihrem Zeitverhalten bekannt sein.

Tasks können unterschiedliche maximale Verzögerungszeiten und unterschiedliche Prioritäten haben. Prinzipiell müssen Tasks daher unterbrechbar sein in dem Sinne, dass eine Task mitten in der

Bearbeitung durch eine Task höherer Priorität unterbrochen werden kann, und anschließend ohne Verlust von Information fortgesetzt wird.

Jeder Scheduler hat nicht zuletzt die Eigenschaft bestimmte Tasks autonom auszuführen – dazu gehören Power Management, Interrupt Management, Speichermanagement, Behandlung von Ausnahmesituationen.

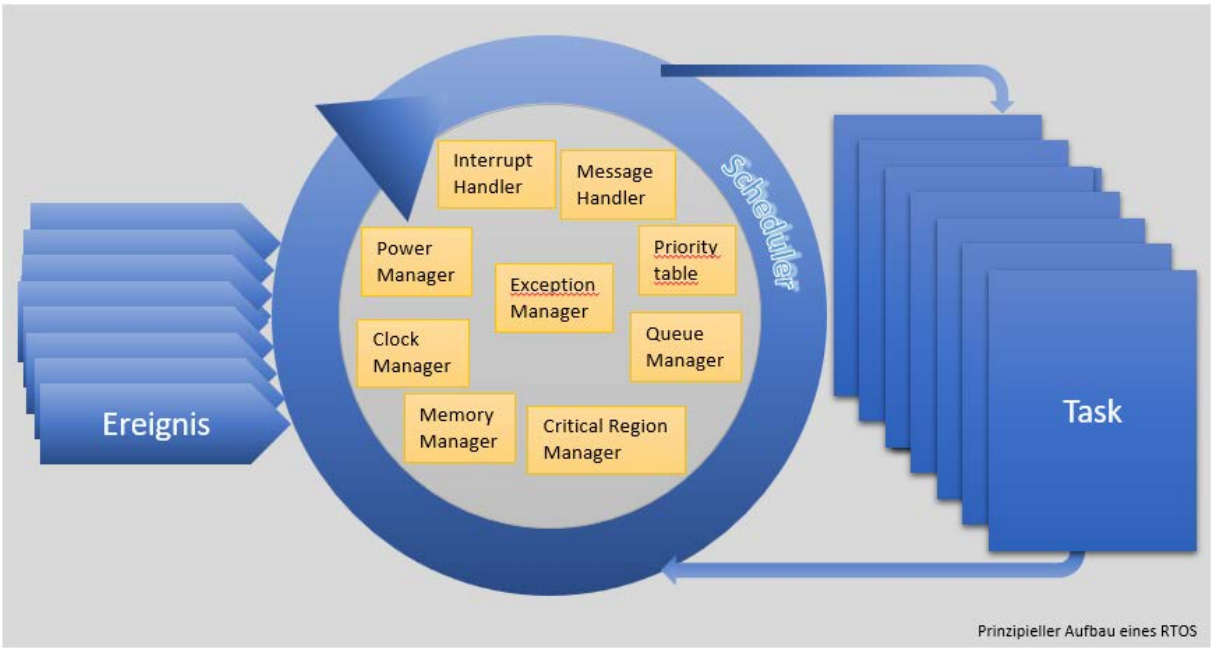
1.4 Komponenten eines Schedulers

- Ein Ereignis ist zufällig bezüglich des Zeitpunkts zu dem es eintritt. Das Ereignis liefert Daten und der Scheduler erzeugt einen Interrupt.
- Eine Task ist ein Algorithmus, der ein Ereignis verarbeitet. Eine Task ist grundsätzlich unterbrechbar und wird nach einer Unterbrechung fortgesetzt. Dafür wird der Status der unterbrochenen Task mit allen Zwischenergebnissen zunächst in einem Task Control Block (TCB) gespeichert und vor der Fortsetzung zurückgeholt. Eine Task ist in der Regel auch reentrant, kann also während sie unterbrochen ist erneut ausgeführt werden. Wichtig ist diese Eigenschaft, wenn eine Task für die Bearbeitung von unterschiedlichen Ereignissen ausgeführt wird. Alle grundlegenden Komponenten des Schedulers sind in diesem Sinne Tasks.

Task Control Block (TCB)

- Adresse zur Fortsetzung der Ausführung
- Registerstände
- Status von Flags und Semaphoren
- …

- Der Scheduler sorgt für den sequentiellen Ablauf der Tasks und teilt dafür die Computer Ressourcen der jeweils aktiven Task zu. Um sicherzustellen, dass alle Tasks ihre zeitlichen Restriktionen einhalten können, werden verschiedene Strategien eingesetzt, die die Abfolge der Taskabschnitte festlegen. Besonders einleuchtend sind „Earliest Deadline First (EDF) und Round Robin (RR) bei dem jede Task zyklisch für ein kurzes Zeitintervall an die Reihe kommt. Der Scheduler bedient sich einiger Mechanismen, die als grundlegende Tasks implementiert werden:



Prinzipieller Aufbau eines RTOS

  - Ein Interrupt wird durch ein Ereignis oder den Clock Manager ausgelöst. Der Scheduler unterbricht in diesem Falle die laufende Task und benutzt die Prioritätentabelle um zu entscheiden welche Task als nächstes zu starten ist.

- Task können untereinander kommunizieren und Daten austauschen. Dafür werden Messages mit Statusinformationen und Daten an eine andere Task versendet. Jede Task, die Messages erhalten kann, prüft bei jedem Aufruf, ob Messages eingegangen sind und bearbeitet diese.
- In der Prioritätentabelle ist festgeschrieben, welchen Rang die einzelnen Ereignisse haben.
- Warteschlangen können bei gleichzeitigem Eintreffen mehrerer Ereignisse entstehen; auch Messages können bei jeder Task in einer Warteschlage gespeichert sein.
- Der Clock Manager kennt die zeitlichen Restriktionen und sorgt dafür, dass sie eingehalten werden. Dafür kann er einen Interrupt erzeugen um eine niederpriore Task auszuführen, wenn sie droht die Restriktion zu verletzen. Diese Art von Interrupt wird regelmäßig in Zeitintervallen von wenigen Millisekunden erzeugt schon allein deshalb, um die Einhaltung von Restriktionen zu überprüfen.
- Ein Critical Region Manager kann Teile eine Task für die Unterbrechung durch einen Interrupt sperren. Das ist zum Beispiel sinnvoll für Zeit in der bei einem Interrupt der Task Status im TCB gespeichert wird.
- Der Exception Manager ist eine Task, die mit unerwarteten Ereignissen umgeht und eine entsprechende Reaktion auslöst.
- Der Memory Manager hat den Überblick über Speicherbereiche in den die TCB's und Zwischenergebnisse der Tasks abgelegt werden.

Viel Theorie erfordert konkrete Beispiele.

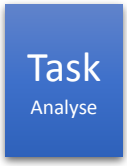
1.5 Beispiele

Ab 1973 war ich als Marineoffizier Teil eines sehr kleinen Kommandos mit der Aufgabe Führungssysteme für Schiffe zu entwickeln. Diese Systeme sollten in Realzeit alle Informationen der Radaranlagen und anderer Messeinrichtungen eines Schiffes oder einer Flottille analysieren können, um auf der anderen Seite ebenso in Echtzeit Abwehrmaßnahmen und Waffen zu steuern. Das war eine Zeit, in der man mit Rechnerkapazität noch sorgsam umgehen musste und so war ein direkter Zugang zu den Komponenten des RTOS erforderlich. Damals wurde auf den neu entwickelten Schnellbooten der Klasse S143 der Rechner 1830B von Univac eingesetzt – ein 30-Bit Rechner mit einer Speicherkapazität von 64kb. Dieser Rechner, obwohl Schrank-groß, war als einziger verfügbarer Rechner klein genug, um auf einem Schnellboot eingesetzt werden zu können.

Das Beispiel ist die Erfassung und Analyse von Radarpulsen einer anfliegenden Rakete und der Störung der Zielerfassung dieser Rakete und elektronische Gegenmaßnahmen. Hier wird unmittelbar deutlich, dass nur wenig Zeit für die Bearbeitung zur Verfügung steht.

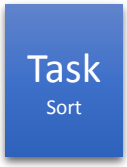
Das Radarsystem hat bei einem Umlauf Signale aufgefangen, die als Daten ein Ereignis markieren. Die Daten enthalten zu jedem Puls charakteristische Informationen z.B. Frequenz und Dauer

Der Scheduler erkennt das eingegangene Ereignis speichert mit seinem Memorymanager die Daten und setzt ein Zeichen „neue Daten". Über den Interrupt Handler erhält eine Task namens Analyse die Ressourcen.

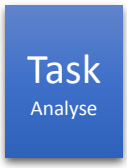


Die Task erkennt neue Daten und sendet zunächst eine Message an eine Task „Sort“, denn die Analyse ist mit sortierten Daten erheblich schneller.

Der Scheduler erkennt mit seinem Message Handler den Bedarf und gibt die Kontrolle an die Task Sort.



Die Daten werden nach verschiedenen Kriterien sortiert und eine Fertigmeldung als Message abgesetzt.



Jetzt kann die Analyse, nachdem Sie vom Scheduler die Kontrolle bekommen hat, die neuen Daten mit Daten aus den vorhergehenden Zyklen vergleichen und daraus Muster erkennen. Ein einfaches Muster wäre z.B. eine konstante Folge von Pulsen die aufgrund bekannter Charakteristiken von Flugkörpern einem bestimmten Raketentyp zugeordnet werden können. In diesem Falle sendet die Analyse eine Message mit den Daten zu dem erkannten Flugkörper an eine Task zur Steuerung eines Störsenders und fordert den Scheduler auf, einen Interrupt mit höchster Priorität zu erzeugen.

Der Scheduler tut was er tun muss, er speichert die Message mit seinem Message Handler und gibt über den Interrupt Handler die Kontrolle an die Task „Send“.

Diese Task liest die Message und steuert daraufhin einen Sender der mit einer auf den Flugkörper abgestimmten Kennung Störpulse aussendet.

Da eine anfliegende Rakete nur wenige Pulse braucht, um ein Ziel zu erfassen, wird deutlich, dass der ganze Ablauf in wenigen Sekunden abgeschlossen sein muss.

Heute umgibt uns eine Vielzahl von Dingen, die uns Informationen geben, für unsere Sicherheit sorgen, uns lästige Arbeiten abnehmen oder uns unterhalten. Diese Dinge kennen unseren Aufenthaltsort, unser Verhalten in einem bestimmten Kontext und agieren/reagieren entsprechend. Alle diese Dinge können das nur, weil in Ihnen ein Scheduler die Kontrolle über die Rechnerleistung einer entsprechenden Task übergibt und mit dem Mechanismus der Interrupts dabei fähig bleibt weitere Ereignisse aufzunehmen und zu verarbeiten:

- Airbag
- GPS
- Temperaturregelungen und Füllstandsüberwachungen
- Fahrassistenzsysteme oder autonomes Fahren
- Interaktive Computeranwendungen (Tastatur, Maus, Controller aller Art)
- Produktionsprozesse z.B. in der chemischen Industrie oder bei der Stahl- und Aluminium Produktion

- Fahrgast-Informationssysteme im öffentlichen Nahverkehr
- Schiffs- oder Flugzeug-Positionssysteme
- Börsenkurse
- Auktionen im Netz
- Streaming von Filmen oder Musik
- Stromverteilung und smart meetering
- Echtzeit MRT
- Wissenschaftliche Anwendungen z.B. in der Experimentalphysik

2. Zufall und Robotik

Roboter begegnen uns täglich – es reicht eine Zeitung aufzuschlagen oder sich im Internet zu bewegen um aktuelle Informationen dazu zu erhalten. Da es verschiedene Definitionen des Begriffs Roboter gibt, hilft es zunächst sich klar zu machen, dass sehr unterschiedliche technische Apparaturen darunter sind, die nur eines gemeinsam haben, nämlich dem Menschen Arbeit abzunehmen.

Die Japan Robot Association (JARA) hat eine nützliche Kategorisierung vorgeschlagen. Manuelle Manipulatoren werden direkt von einem Bediener geführt, der z.B. Bewegungen ausführt, die der Roboter eins zu eins nachmacht. Alle höheren Kategorien bewegen sich nach Algorithmen, die feste oder variable Sequenzen nachvollziehen. Nur die höchste Kategorie der intelligenten Roboter verfügt über eine eigene Wahrnehmungsfähigkeit und kann Bewegungen danach steuern.

- Manual Manipulator
- Fixed Sequence Robot
- Variable Sequence Robot
- Playback Robot
- Numerical Control Robot
- Intelligent Robot

Quelle: JARA

Jede Robotik, die sich in Raum und Zeit bewegt, muss in der Lage sein, Zufälle zu bearbeiten. Jedes Hindernis im vorgesehenen Weg sollte erkannt werden, um schädliche Kollisionen zu vermeiden. Um das Ziel der Bewegung zu erkennen, müssen Zufälle in der Beleuchtung und der optischen Sensorik bewältigt werden. Zufällige Unregelmäßigkeiten in der Lage von Werkstücken oder der Beschaffenheit der Oberfläche sollte der Roboter ausgleichen können und besondere Zufälligkeiten in der Interaktion mit Menschen müssen mit höchster Priorität bewältigt werden. Es ist daher nicht verwunderlich, dass wir auch in intelligenten Robotern einen Scheduler wiederfinden.

2.1 Sensoren und Effektoren

Sensoren sind alle technischen Messeinrichtungen, die Daten liefern können wie z.B.

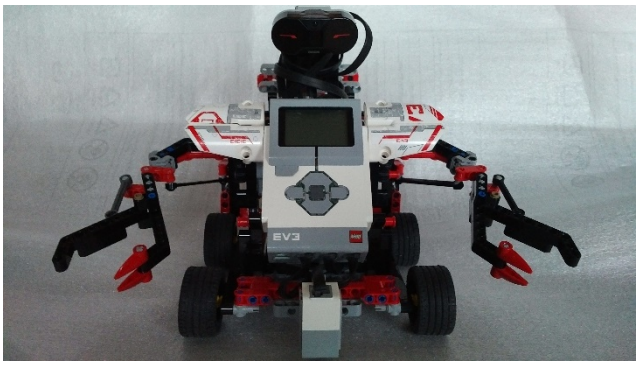
- Wärmemenge, Temperatur, Feuchtigkeit,
- Druck, Schallfeldgrößen, Helligkeit, Farbe, Beschleunigung
- Elektromagnetische Wellenpulse
- pH-Wert, Ionenstärke, elektrochemisches Potential

Effektoren (synonym auch Aktoren) sind unter anderem

- Elektromotoren
- Bimetalle
- Hydraulik, Pneumatik
- Elektrochemische und elektromechanische Aktoren
- Bildschirme

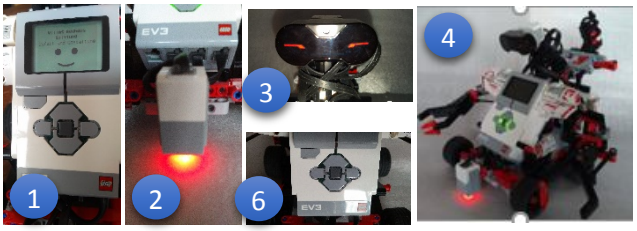
- Spracheinheiten

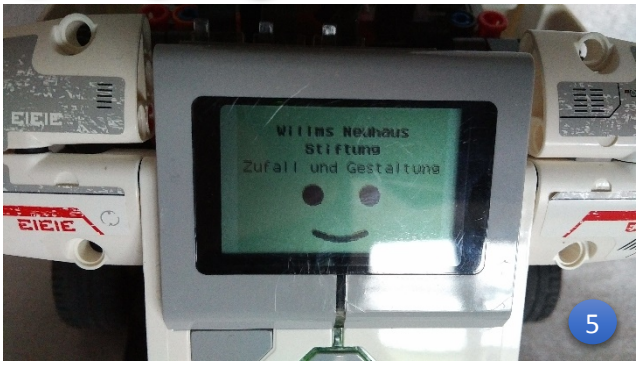
2.2 Beispiel AlZ2 - ein kleiner Roboter aus Lego-Bausteinen



Wie seine großen Kollegen, besteht auch dieser kleine Beispielroboter aus

- einem zentralen Computer (1),
- Sensoren, in diesem Falle einem Farbsensor (2) und einem Infrarotsensor (3),
- Effektoren, in diesem Falle 3 Motoren (4), einem Bildschirm (5), ein paar LEDs (6) und einer Spracheinheit.





Im zentralen Computer verrichtet, wie könnte es anders sein, ein Scheduler seine Arbeit und für die Verarbeitung der Sensordaten und die Steuerung der Effektoren stehen entsprechende Tasks bereit. Der Programmierer hat Tasks programmiert, die den Roboter befähigen, entlang einer schwarzen Linie seinen Weg autonom zu finden und stehen zu bleiben, falls ein Hindernis hierbei im Wege steht.
Das Modell funktioniert wie folgt:

Der Scheduler wird gestartet, führt eine Initialisierung für Bildschirm und Sprachausgabe durch und startet Sensoren und Effektoren.

Ereignis — Der Farbsensor meldet eine erkannte Farbe

Die Task für die Motorsteuerung wird aufgerufen.

Task Motor — Die Task stellt fest, welche Farbe gemeldet wurde, und korrigiert den Kurs durch Ein- oder Ausschalten des linken oder rechten Motors. Bei rot wird nach links gesteuert, bei weiß nach rechts, bei schwarz laufen beide Motoren

Ereignis — Der Infrarotsensor meldet ein Hindernis

Der Scheduler erzeugt einen Interrupt, stellt anhand seiner Prioritätentabelle fest, dass alle anderen ´Tasks unterbrochen werden müssen und gibt die Kontrolle der Task für die Motorsteuerung

Task Motor

Unabhängig davon, wo die Motorsteuerung unterbrochen wurde, fängt sie eine neue Bearbeitung an und stoppt in diesem Falle beide Motoren. Dann schickt sie eine Message an die Task für die Sprachausgabe

Der Scheduler prüft den Status der Sensoren, bemerkt, dass das Hindernis noch da ist und gibt die Kontrolle an eine niederpriore Task für die Sprachausgabe.

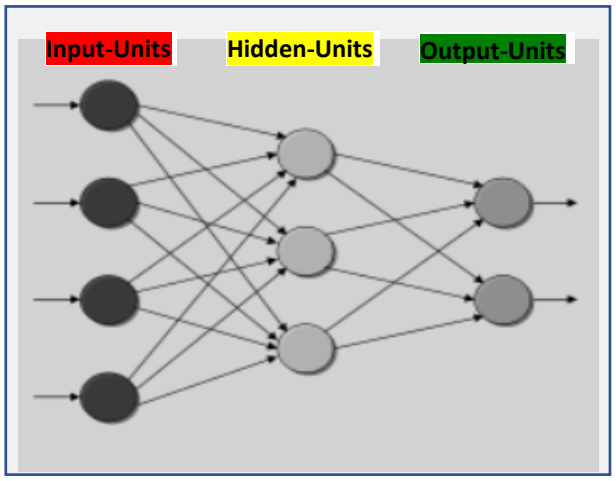
Task Sprich

Die Sprachausgabe sagt „Outsch“

Sobald das Hindernis beseitigt ist, erhält die Motorsteuerung wieder die Kontrolle.

Für einen guten Überblick über den Stand in der Robotik lohnt ein Blick auf die Messen Cebit in Frankfurt und IREX in Tokio des Jahres 2017, deren Schwerpunkt die Robotik waren. Im Vordergrund stand insbesondere die Verbindung mit künstlicher Intelligenz (KI).

3. Roboter mit neuronalen Netzen – Zufälle mal 2 (semantische Zufälle)



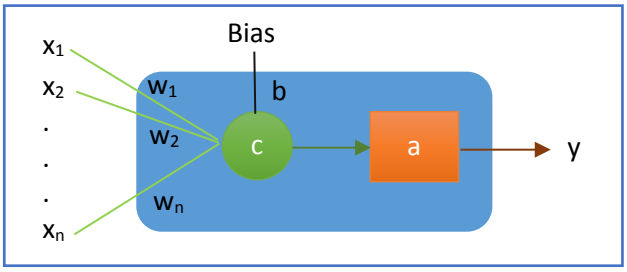
Die Verbindung mit den neuronalen Netzen, deren Prinzip Teil des ersten Vortrags zum Thema Algorithmen und Zufall war, ist der entscheidende Schritt für die Autonomie der Roboter. Im Unterschied zu Robotern, die auch ohne KI Zufälle in vorgegebenen Zusammenhängen bearbeiten (siehe AlZ2), können Roboter mit KI lernen sich in der Umgebung zurechtzufinden und sich an die Umgebung anzupassen. Vor allem können sie kommunizieren und sind damit Zufällen anderer Art, semantischen Zufällen, ausgesetzt.

Wie neuronale Netze im Prinzip funktionieren, war bereits Thema des ersten Vortrags zu diesem Thema. Daher werde ich dieses Mal das Trainieren der neuronalen Netze in den Vordergrund stellen und einige besondere Beispiele erläutern. Zuvor soll aber die Funktion der einzelnen Units in einem neuronalen Netz verständlich werden.

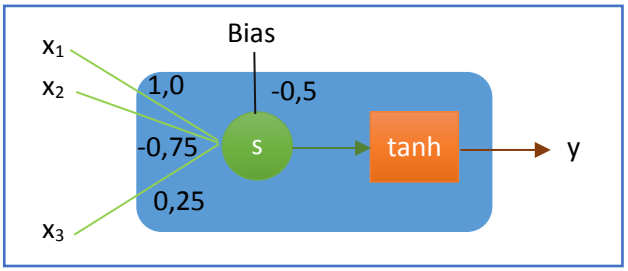
3.1 Der Neuronen-Algorithmus

Natürlich ist die einzelne Unit in der obigen Darstellung auch ein Algorithmus und ein einfacher dazu. Dieser Algorithmus wird im Kontext der neuronalen Netze Neuron oder Perceptron genannt. Ich verwende, der Einfachheit halber, den Begriff Neuron. Die Vorstellung allerdings, dass mit einem solchen Neuronen-Algorithmus die Wirklichkeit biologischer Neuronen vollkommen nachgebildet sei, musste wieder aufgegeben werden. Im biologischen Umfeld spielen noch andere Faktoren wie Nahrungsangebot, Stresslevel oder die Anwesenheit von freien Radikalen eine Rolle.

Der Neuronen-Algorithmus ist in diesem Sinne ein Modell und so funktioniert er:



Wie in der Grafik dargestellt, empfängt das Neuron Signale (Inputs $x_1 – x_n$) denen ein Gewicht ($w_1 – w_n$) zugeordnet ist. Zusätzlich geht ein Bias, eine Stör- oder Steuerungsgröße, ein, die eine Verstärkung oder Abschwächung des Ausgangssignals des Neurons bewirkt. Die Einstellung des Bias wird in der Lernphase optimiert. Die Funktion c kombiniert alle Eingangswerte. Mit der kombinierten Eingangsfunktion wird dann die Aktivierungsfunktion a ausgeführt um den Ausgangswert y zu erzeugen. Aktivierungsfunktionen sind überwiegend linear (dann ist sie gleich der Kombinationsfunktion), logarithmisch oder hyperbolisch.



Betrachten wir zum Beispiel ein Neuron mit 3 Eingangswerten x = (-0,8, 0,2, -0,4), den Gewichten w = (1,0, -0,75, 0,25) und dem Bias b = -0,5. Die Kombinationsfunktion soll schlicht die Summe aller Eingangswerte sein. Das bedeutet, dass jeder Eingangswert mit seinem Gewicht multipliziert und dann aufaddiert wird – in mathematischer Notation

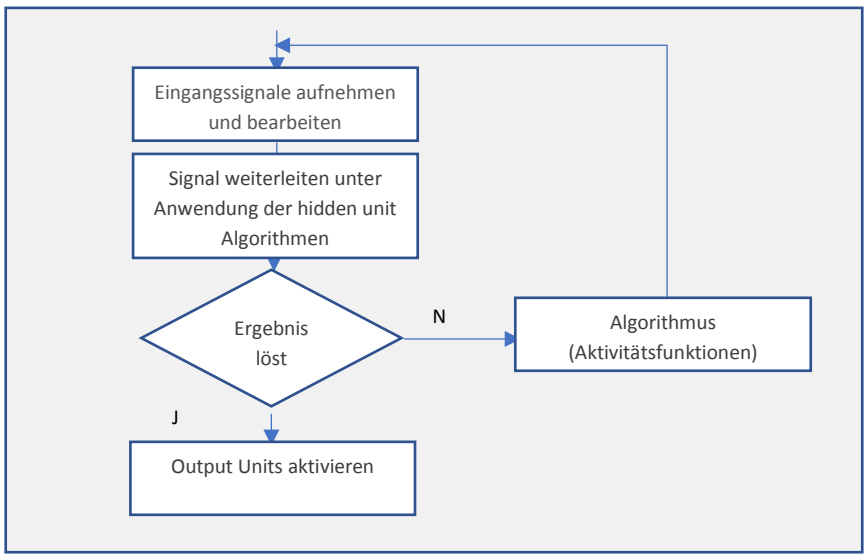
$$(c = b + \Sigma\, w_i\, x_i).$$

Für die Beispielwerte lautet die Rechnung s = -0,5 + (1,0 x -0,8) + (-0,75 x 0,2) + (0,25 x -0,4) = -1,55. Die Aktivierungsfunktion ist in diesem Beispiel hyperbolisch und das Ergebnis ist

$$y = \tanh(-1{,}55) = -0{,}91$$

Das Neuron ist damit ein mathematisches Modell für das Verhalten eines einzelnen biologischen Neurons.

3.2 Netz-Training – Zufälle mal 3 (Trainingszufälle)



Im ersten Vortrag zum Thema Algorithmen und Zufall war bereits ein einfaches Blockdiagramm zu sehen, dass die Lernfunktion eines neuronalen Netzes darstellt. Mit dem Instrumentarium aus Abschnitt 3.1 wissen wir nun etwas genauer, was ein hidden unit Algorithmus ist und die Aktivierungsfunktion bewirkt.

Der Algorithmus, der die Lernfunktion abbildet hat prinzipiell 4 Stellschrauben:

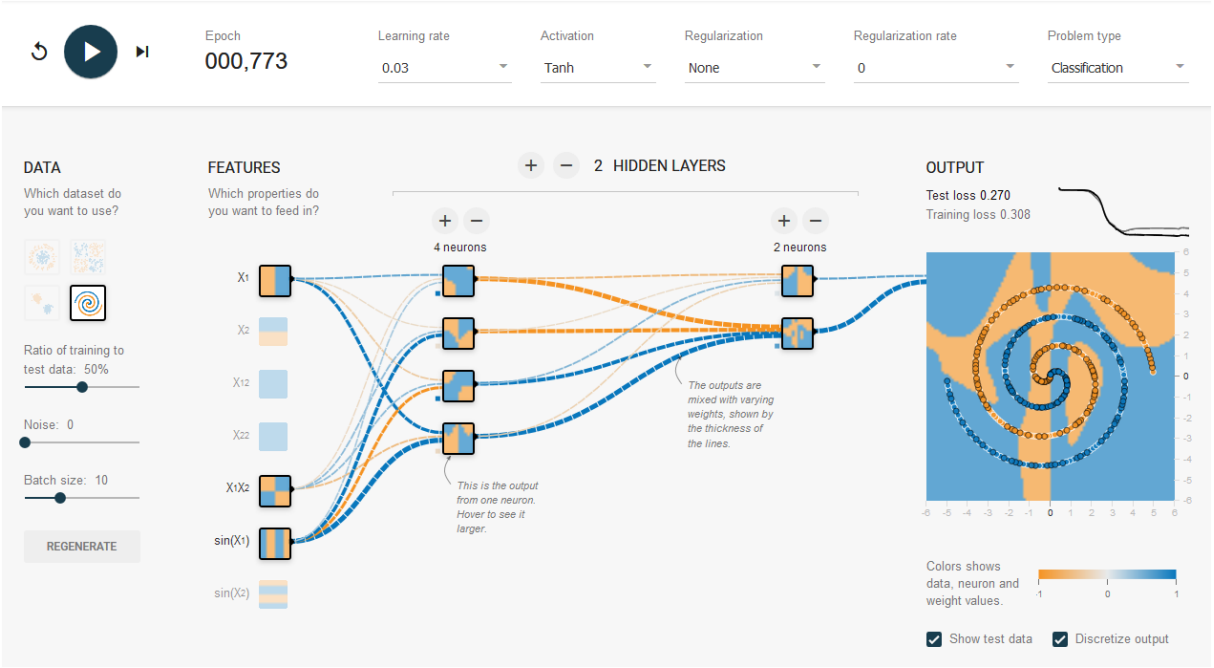
- Das Gewicht w, mit dem die Inputwerte x gewichtet werden
- Den Bias Wert b
- Die Kombinationsfunktion c
- Die Aktivierungsfunktion a

In der Regel begnügt man sich w und b zu modifizieren und c und a als Bestandteil des neuronalen Netzes zu betrachten. Der Lern-Algorithmus ist unter diesen Voraussetzungen als Minimierung einer Kostenfunktion realisiert. Man kombiniert dafür die Gewichte aller Eingangswerte und den Bias zu einem n-dimensionalen Gewichtsvektor. Die mathematischen Verfahren (und Algorithmen) zur Bildung des Minimums der Kostenfunktion sind wohlbekannt. Das Minimum bildet den im Sinne des Trainingserfolgs optimalen Wert für die Gewichte der Eingangswerte und den Bias. Leider ist die Kostenfunktion im Allgemeinen nicht linear. Daher wird zusätzlich ein Stopp Kriterium gebraucht, das definiert, wann der Lernerfolg erreicht ist.

Das Lernen startet mit einem zufälligen Gewichtsvektor. Dann wird eine große Folge von Eingangswerten generiert mit denen das neuronale Netz gefüttert wird. Dabei wird der Gewichtsvektor bei jedem Durchlauf angepasst. Nach jedem Durchlauf wird die Reduktion der Kostenfunktion betrachtet. Der Lernprozess stoppt, wenn für die Reduktion ein bestimmter Schwellwert unterschritten wird.

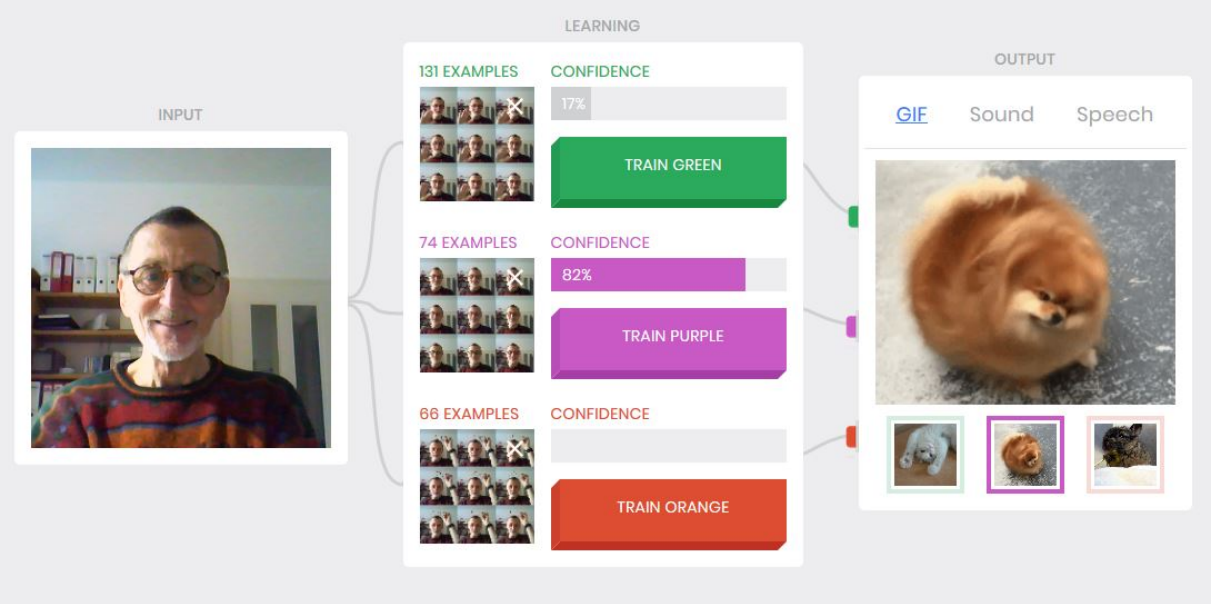
Beispiel

Google ist ganz vorne dabei, wenn es darum geht künstliche Intelligenz für viele Programmierer verfügbar zu machen. Eine eigene Plattform ermöglicht die Beteiligung.



Eine Demonstration des Lernens lässt sich direkt im Browser ausführen. Man wählt ein Datenformat Input- Hidden- und Output Units. Die Anzahl der Units auf einer Ebene lässt sich genauso verändern wie die Anzahl der Hidden Unit Ebenen. Die Farben deuten negative und positive Werte an. Im Laufe der Iteration sieht man, wie bestimmte Pfade verstärkt und andere abgeschwächt werden und das Ergebnisbild sich verändert.

Wem das noch zu abstrakt ist, kann sich vielleicht mit einem weiteren Beispiel anfreunden. Es geht darum, ein neurales Netz darauf zu trainieren, Selfies in unterschiedlichen Posen zu erkennen. Das geht wieder mit der KI von Google indem man zunächst viele Bilder von sich selbst in 3 verschiedenen Posen machen lässt. Anschließend ist die KI in der Lage zwischen den Posen zu unterscheiden.



Nach der Lernphase kann man verschiedene Posen ausprobieren und der Algorithmus reagiert mit einem der 3 zugeordneten Ergebnisbilder oder einer Sprachausgabe.

Das Prinzip Bilder, oder allgemeiner zufällige Konstellationen, wiederzuerkennen und daraus Handlungen abzuleiten, benutzen künstlicher Intelligenzen die sich frei bewegen können wie autonome Fahrzeuge (an Land, zu Wasser oder in der Luft) oder tierähnliche oder humanoide Roboter.

Schlussbetrachtung

Die Kenntnis grundlegender Algorithmen wie dem Scheduler und dem Neuronen-Algorithmus kann helfen, die durch die Informatik angestoßenen gesellschaftlichen Veränderungen zu verfolgen und einzuordnen. Mit diesem Wissen lässt sich die Diskussion um künstliche Intelligenz und Robotik besser verstehen und technische Schwärmerei lässt sich besser unterscheiden von realen Fortschritten.

Es lohnt sich, die weitere Entwicklung aktiv zu verfolgen.

Quellenverzeichnis

Klaus Mainzer	Künstliche Intelligenz
Klaus Mainzer	Der kreative Zufall
Heinrich Niemann	Methoden der Mustererkennung
Sigurd Hess	Als die Computer lernten zur See zu fahren
Mickael Launay	Der große Roman der Mathematik
http://vipclubmn.org	Information Technology Pioneers

https://en.m.wikipedia.org/wiki/Real-time_computing

https://www.neuraldesigner.com/blog

Google	https://teachablemachine.withgoogle.com

Weitere öffentliche Quellen aus Wikipedia und Youtube

Veröffentlichungen der FAZ

Anhang

(1) Algorithmus

Der Algorithmus beschreibt einen Prozess in einzelnen Schritten, die zum Ergebnis führen.

- Ausführbarkeit – es muss einen Automaten geben, der jeden Schritt des Algorithmus ausführen kann
- Eindeutigkeit – nach jedem Schritt muss eindeutig klar sein, welcher Schritt als nächstes folgt
- Endlichkeit – der Algorithmus darf nur endliche viele Schritte umfassen
- Speicherbarkeit – für jeden Schritt darf nur ein endlich großer Arbeitsspeicher eingesetzt werden
- Allgemeinheit – der Algorithmus löst nicht nur einen Spezialfall, sondern eine Klasse von Problemen
- Terminiertheit – der Algorithmus soll nach endlich vielen Schritten stoppen