TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Hauptseminar

Maschinelles Lernen auf FPGAs

Matthias Brinker

28.06.2018

Gliederung

1. Motivation
2. Konzepte
3. Multilayer perceptrons (MLP)
4. Random Forest
5. Q-Learning
6. Maschinelles Lernen auf FPGAs
7. Fazit & Ausblick

Motivation

Motivation

Problem:

— Herkömmliche Algorithmen benötigen Fachwissen

— Fachwissen muss formalisierbar sein

— Schwierig in einigen Themenbereichen (Spracherkennung, Geischtserkennung, ...)

Lösung:

— Algorithmus lernt selbstständig

Konzepte

Konzepte

- Unüberwachtes Lernen (unsupervised learning)
- Überwachtes Lernen (supervised learning)
  - Bestärktes Lernen (reinforcement learning)
- Seit 2000 fast nur noch Überwachtes Lernen

Unüberwachtes Lernen

Voraussetzung:
— Große Menge an unbeschriftete Daten

Aufgabe:
— Gruppenfindung innerhalb der Daten
— Dimensionale Reduktion

Ablauf:
— Algorithmus sucht Gemeinsamkeiten in den Daten und gruppiert oder reduziert diese

Überwachtes Lernen

Voraussetzung:
— Qualitativ hochwertige Daten für das Trainingsset

Aufgabe:
— Kategorisierung von Daten
— Voraussagen für Daten treffen

Ablauf:
— Ein Algorithmus wird anhand des Trainingsset auf ein spezielles Problem trainiert

Bestärktes Lernen

Voraussetzung:
— Prozess wird als Markow-Entscheidungsproblem formuliert

Aufgabe:
— Optimalen Prozessablauf finden

Ablauf:
— Agent definieren
— Agent hat Zustände
— Agent führt Aktionen aus
— Erhält Belohnungen für Aktionen
— Baut „Gedächtnis“ auf, welche Aktionen zielführend sind

Bestärktes Lernen

Markow-Entscheidungsproblem:

— Tupel (S, A, T, r, p_0)

— Mit S als eine Menge von Zuständen

— Mit A als eine Menge von Aktionen

— Mit T als die Übergangswahrscheinlichkeit von Zustand s ∈ S nach Zustand s‘∈ S bei der Aktion a ∈ A

— Mit r als Belohnung

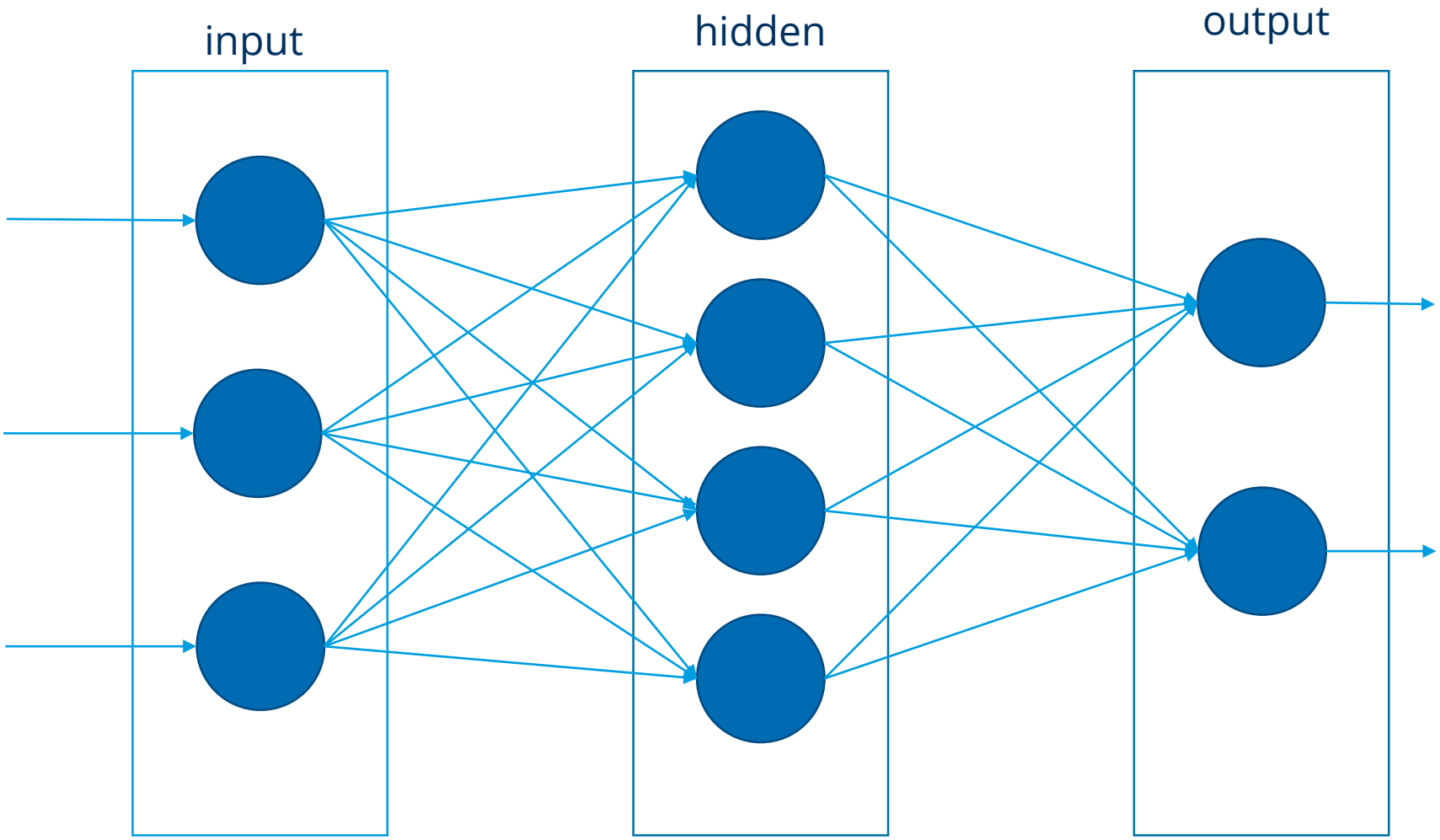
— Mit p_0 als Wahrscheinlichkeitsmatrix, welche für jeden Zustand s ∈ S angibt mit welcher Wahrscheinlichkeit es sich um den Startzustand handelt

Multilayer perceptrons (MLP)

Multilayer perceptrons (MLP)

- Umgangssprachlich „vanilla neural network“
- Besteht aus mindestens drei Layers (input, hidden, output)
- Alle Knoten sind mit allen Knoten des folgenden Layers verbunden
- Jede Kante hat ein Gewicht
- Jeder Knoten (außer Input-Knoten) hat eine Funktion, welche aus den gewichteten Eingängen einen Ausgang berechnet („activation function“)

Multilayer perceptrons (MLP)



Multilayer perceptrons (MLP)

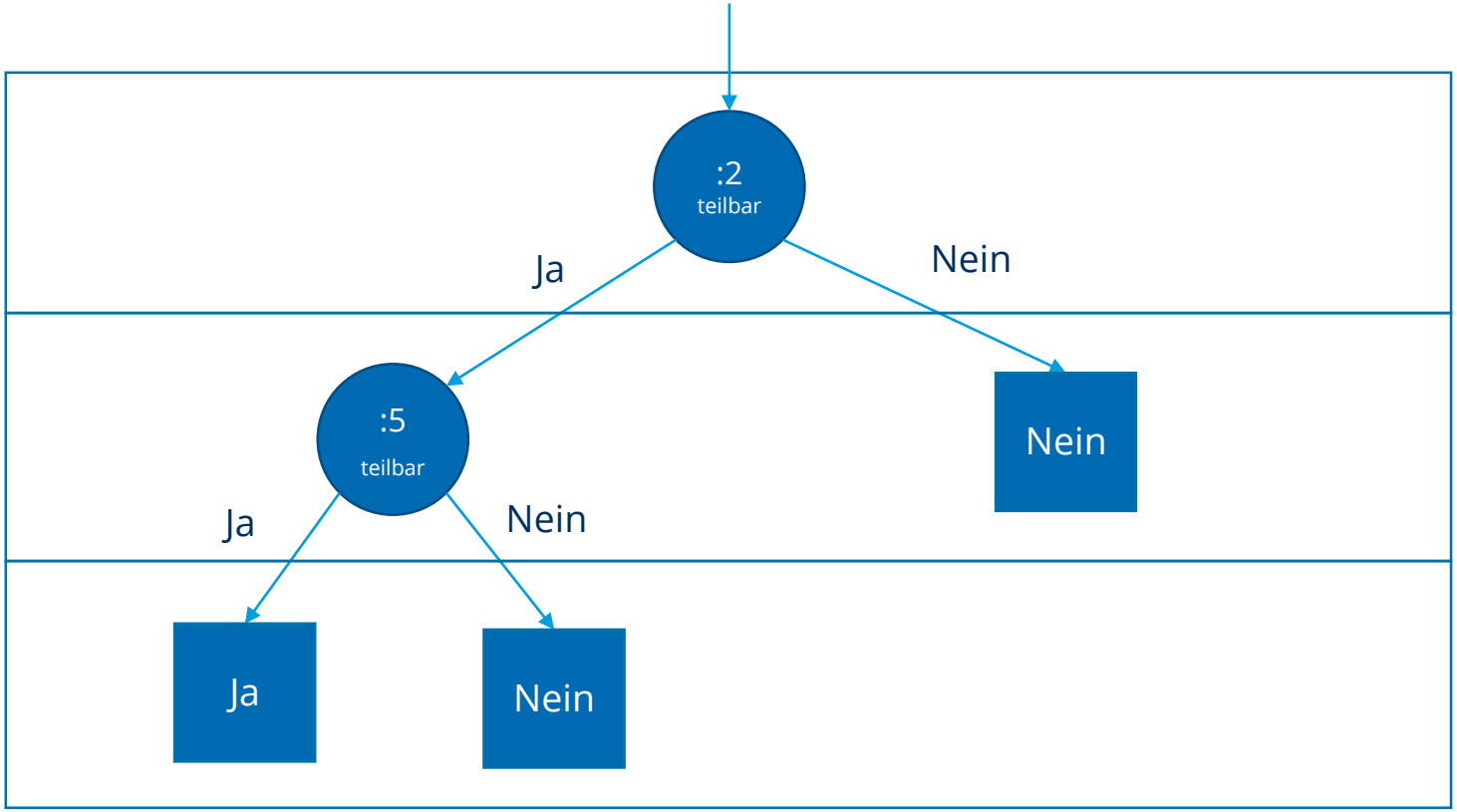
- „feed-forward“ Netzwerk
- Lernt durch „backpropagation“
  - Vergleich Ergebnis mit Soll-Wert aus dem Trainingsset
  - Berechnet Fehler
  - Passt die Gewichte an

Random Forest

Random Forest

- Besteht aus mehreren unkorrelierten Entscheidungsbäumen
- Jeder Baum entscheidet sich für ein Ergebnis/Klasse
- Der Durchschnitt, beziehungsweise die Klasse mit den meisten Stimmen wird als Ergebnis genommen

Binärer Entscheidungsbaum (Zahl durch 10 teilbar)



Random Forest

— Bäume werden anhand des Trainingsset aufgebaut

— Verschiedene Möglichkeiten die Bäume aufzubauen

Nach Breiman [4]:

1. Von den n Beobachtungen in der Trainingsmenge werden n Stück zufällig mit Zurücklegen gezogen.
2. Von M Merkmalen der Trainigsdaten werden an jedem Konten im Baum m << M Merkmale zufällig ausgewählt, die als Schnitt infrage kommen sollen. Aus diesen kann dann ein Merkmal nach einem Kriterium ausgesucht werden
3. Der Baum wird voll ausgebaut und nicht zurückgeschnitten

Q-Learning

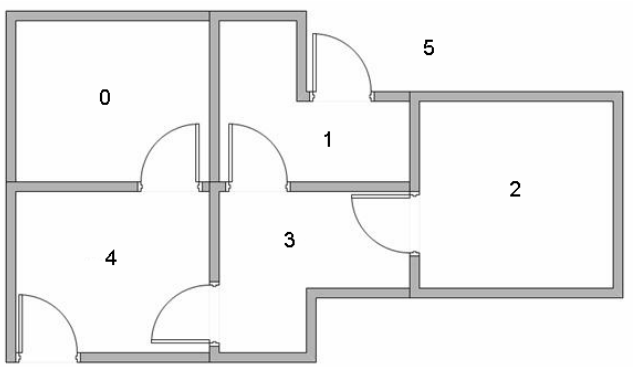
Q-Learning

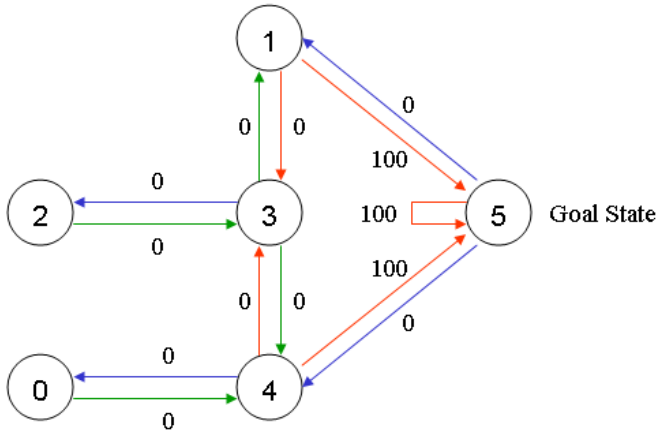
- Bestärktes Lernen
- Nicht Modelbasiert
- „off-policy“
- „Q-Matrix“ speichert für jeden Zustandsübergang eine Wertigkeit

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{old\ value} + \underbrace{\alpha}_{learning\ rate} \cdot \overbrace{\left(\underbrace{r_t}_{reward} + \underbrace{\gamma}_{discount\ factor} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{estimate\ of\ optimal\ future\ value} \right)}^{learned\ value}$$

Quelle [5]

Q-Learning Beispiel [6]

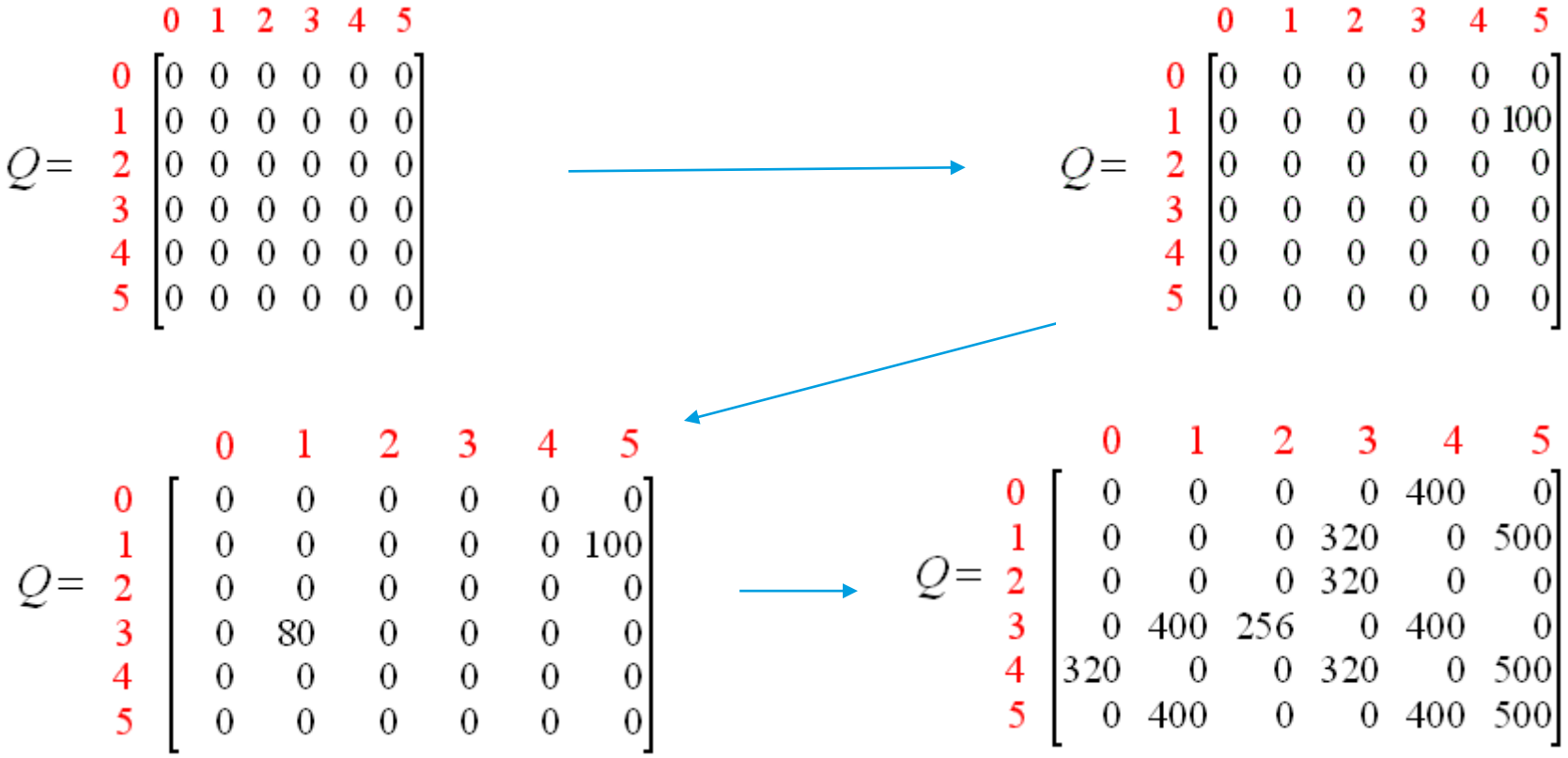




$$R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Q-Learning Beispiel [6]



Maschinelles Lernen auf FPGAs

Maschinelles Lernen für FPGAs

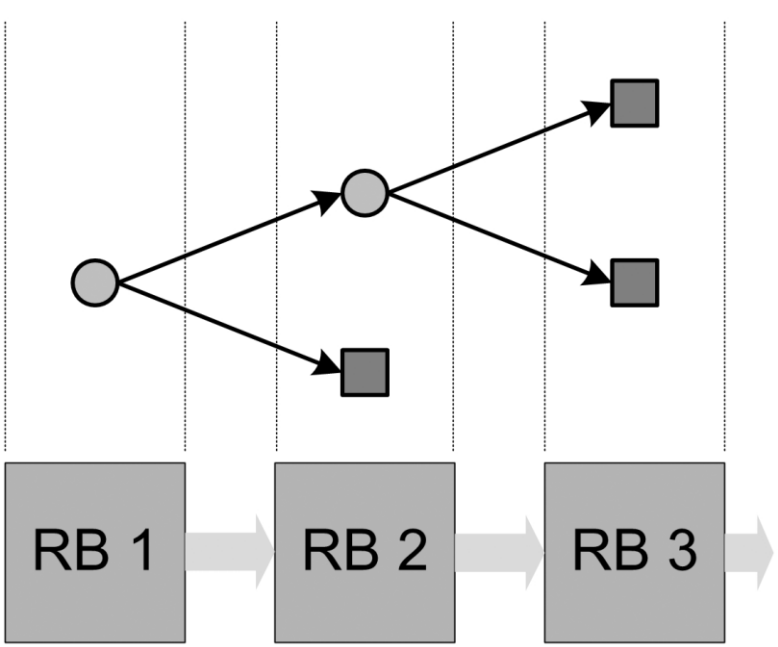
- Kaum in realen Applikationen vertreten
- Datenanalysten können High-Level-Programmiersprachen, aber keine Hardwarebeschreibung
- Lange Iterationszyklen, compile (place & route)

- Strukturelle Vorteile gegenüber Grafikkarten
- Pipeline
- (Neue FPGAs) zur Laufzeit rekonfigurierbare

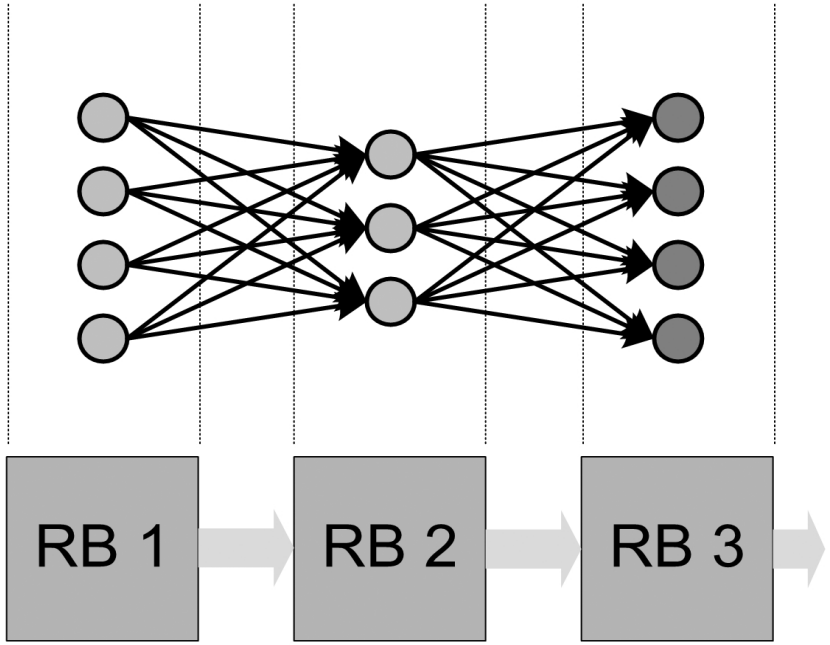
Reconfigureable machine learning classifier (RMLC) [3]

- Rechnerische Gemeinsamkeiten in verschiedenen Algorithmen
- Strukturelle Gemeinsamkeiten in Hardwareimplementierungen
- Rekonfigurierbare Blöcke entwickeln, welche für verschiedene Machine-Learning-Algorithmen genutzt werden können

Reconfigureable machine learning classifier (RMLC) [3]



Mapping für Entscheidungsbäume



Mapping für ein MLP-Netzwerk

Reconfigureable machine learning classifier (RMLC) [3]

Für einige Algorithmen nutzbar:

— Axis parallel decision trees (AP-DT)

— Suport Vector Machine with polynomial kernel (SVM-P)

— Suport Vector Machine with radial kernel (SVM-R)

— Multilayer perceptrons artificial neural network (MLP-ANN)

— Radial basis function artificial neural network (RBF-ANN)

Reconfigureable machine learning classifier (RMLC) [3]

Ergebnis:

— Vergleich R project vs 1D RMLC
— Vergleich R project vs 2D RMLC (15 parallele 1D RMLC)
— Vertex 7 (3500-4000€) gegen AMD Phenom II 1090T (3,2GHz) (gebraucht 100€)

Reconfigureable machine learning classifier (RMLC) [3]

Dataset	AP-DT	SVM-P	SVM-R	MLP-ANN	RBF-ANN
bc	17.489	4.597	6.526	5.125	2.597
bcw	11.448	4.613	9.917	15.868	7.166
ca	9.672	5.454	9.450	3.860	1.696
cg	9.514	7.655	9.035	3.899	1.965
col	11.776	7.540	11.385	5.847	2.924
cvr	19.687	4.882	8.690	8.011	3.711
hd	19.694	5.726	11.263	10.641	4.858
hep	10.917	11.397	17.597	11.568	5.244
hs	6.781	4.006	7.248	7.901	3.633
ion	5.009	4.140	6.218	5.607	2.637
ld	5.651	3.819	8.048	1.347	0.664
lr	10.571	5.861	8.959	6.664	3.268
mus	17.146	4.299	4.479	29.481	13.676
pid	7.112	3.663	6.998	8.986	4.069
rvp	21.293	9.559	12.225	23.274	10.356
sb	4.260	3.747	4.117	4.486	2.317
son	4.774	3.486	4.075	6.086	2.906
ttt	4.204	3.748	5.857	3.947	1.829
Average speedup	10.94	5.46	8.45	9.03	4.20

Klassifizierungs-Speedup 1D RMLC im Vergleich zu R project

Reconfigureable machine learning classifier (RMLC) [3]

Dataset	AP-DT	SVM-P	SVM-R	MLP-ANN	RBF-ANN
bc	237.573	68.385	97.076	75.813	38.424
bcw	83.611	54.992	120.024	202.478	92.514
ca	114.562	80.571	139.597	55.871	24.578
cg	111.268	113.875	134.39	56.367	28.4
col	142.676	111.084	167.729	84.969	42.546
cvr	240.891	69.734	124.453	116.264	53.939
hd	236.532	82.988	163.465	153.82	70.41
hep	98.482	148.709	230.709	157.93	71.869
hs	46.467	55.383	100.598	99.544	46.192
ion	33.729	50.379	76.083	71.297	33.684
ld	48.471	55.055	116.338	17.879	8.914
lr	114.461	72.768	111.854	94.804	46.547
mus	232.05	64.307	66.999	436.108	202.315
pid	63.193	53.525	102.535	121.015	55.281
rvp	256.855	141.012	180.354	337.79	150.507
sb	23.216	55.583	61.158	53.471	27.71
son	25.942	40.516	47.507	72.543	34.755
ttt	58.051	55.831	87.246	58.547	27.125
Average speedup	120.446	76.372	118.229	125.917	58.651

Klassifizierungs-Speedup 2D RMLC im Vergleich zu R project

„Multi-core, GP-GPU, or FPGA?“ [2]

- Compact Random Forest
  - Voll ausgelasteter Entscheidungsbaum
  - Feste Laufzeit
- „Clumps“ für FPGAs
  - Breite Daten-Pipeline und viele Empfänger → Routing Probleme
  - Gruppe von Entscheidungsbäumen die eine eigene Daten-Pipeline teilen

„Multi-core, GP-GPU, or FPGA?“ [2]

CPU:

— 2-socket Intel X5660 Westmeresystem with 12 cores running at 2.8 GHz

GP-GPU:

— 2-socket Intel X5660 Westmeresystem with 12 cores running at 2.8 GHz, 96 GB DRAM,
— NVIDIA Tesla M2050 that had 3 GB GDDR5

FPGA:

— Global HTG-V6-PCIE-L240-1 Board mit einem XC6VLX240T-1FFG1759 Virtex 6

„Multi-core, GP-GPU, or FPGA?“ [2]

Ergebnis:

Accelerator	No. Clumps/ Total No. Trees Per FPGA	# FPGAs, GPUs, or CPUs	Classification Rate (Ksamples/s)	Accel. Power (W)	Accel. Cost ($)	CRF Performance/ Power ((Ksamples/s)/Watt)	CRF Performance/Cost ((Ksamples/s)/$)
XC6VLX240T-1	1/8	4	31,250	12	7765	2604	4.0
XC6VLX240T-1	2/8	4	31,250	13	7765	2404	4.0
XC6VLX550T-2	2/16	2	31,250	11	10889	2841	2.9
XC6VLX550T-2	4/16	2	31,250	12	10889	2604	2.9
Tesla M2050	N/A	1	20,398	225	2699	91	7.6
Intel X5660	N/A	2	9,291	190	2440	49	3.8

Fazit & Ausblick

Fazit

Vorteil:
— Anstieg in der Performance
— Sehr ökonomisch im Energieverbrauch

Nachteil:
— Teuer in der Anschaffung
— Für Software-Programmierer schwierig zu programmieren
— Lange Iterationszyklen

Ausblick

- Probleme wachsen rasant, FPGAs müssen skalieren
- FPGA Technologie passt sich dem Trend an Größere Speicherkapazitäten
  - Kleinere Strukturgrößen
  - Größere Speicherkapazitäten
  - Verbesserte Querverbindungen für Multi-FPGA-Konfigurationen
- Design Tools mit höherer Abstraktion (High-Level)
- Übernahme von Altera durch Intel und Partnerprogramme zwischen IBM und Xilinx wird die FPGA-Landschaft stark verändert und in naher Zukunft in Datenzentren einsatzbereit [1]

Vielen Dank für eure Aufmerksamkeit!

Quellen

[1] Griffin Lacey, Graham Taylor and Shawki Areibi (2016). „Deep Learning on FPGAs: Past, Present, and Future“. **arXiv:1602.04283**

[2] Brian Van Essen, Chris Macaraeg, Maya Gokhale and Ryan Prenger (2012). „Accelerating a random forest classifier: multi-core, GP-GPU, or FPGA?“. 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines

[3] Vuk S. Vranjkovic, Rastislav J. R. Struharik and Ladislav A. Novak (2015). „Reconfigureable Hardware for Machine Learning Applications“, Journal of Circuits, Systems and Computers

[4] Breiman L., Random forests. In Machine Learning, Seiten 5–32, 2001

[5] Wikipedia contributors. (2018, June 15). Q-learning. In *Wikipedia, The Free Encyclopedia*. Retrieved 08:22, June 15, 2018, from https://en.wikipedia.org/w/index.php?title=Q-learning&oldid=846509589

Quellen

[6] McCullock, John. „A Painless Q-Learning Tutorial“. http://mnemstudio.org/path-finding-q-learning-tutorial.htm (15.06.2018)