

## Abschlusspräsentation der Java-Gruppe

Parallelisierung des Neuronale Netze Simulators Neuroph Markus Braun, Daniel Hammann, Dominik Messinger, Dominic Rausch | 11. Februar 2013

INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD), LEHRSTUHL FÜR PROGRAMMIERSYSTEME

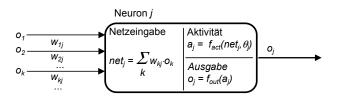


#### **Neuronale Netze**



#### Einleitung

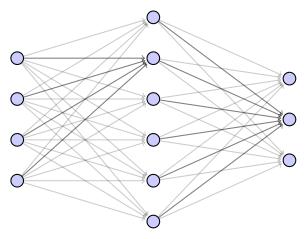
- Bestehen aus Neuronen (Verarbeitungseinheiten)
- Gewichtete Eingabeverbindungen werden zusammenfasst (Propagierungsfunktion)
- Aktivierungsfunktion mit Schwellwert
- Aus Aktivierung folgt mittels Ausgabefunktion die Ausgabe



[Quelle: Lehr- und Übungsbuch künstliche Intelligenz; Lämmel, Cleve; 2012]



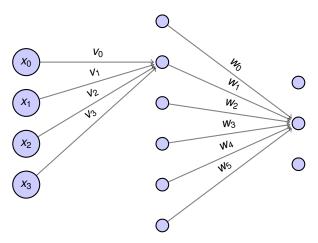




Multilayerperceptron-Netz



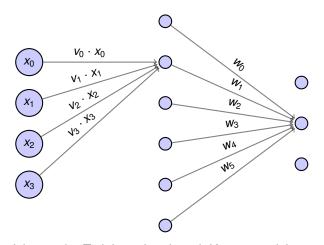




Trainingseingabe und Kantengewichte



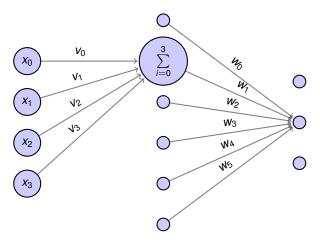




Gewichtung der Trainingseingabe mit Kantengewicht



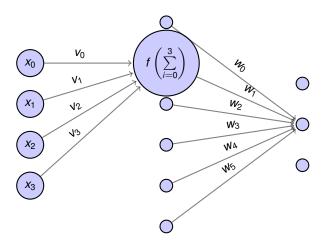




Summe der Produkte ergibt Netzeingabe in der Hiddenschicht



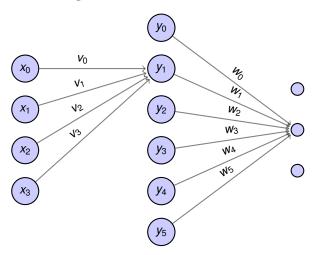




Netzeingabe der restlichen Neuronen der Hiddenschicht



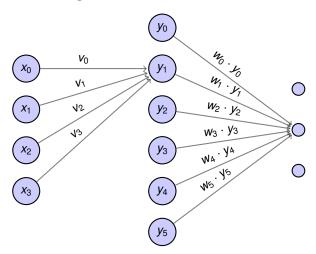




Umrechnung der Netzeingabe mit der Übertragungsfunktion



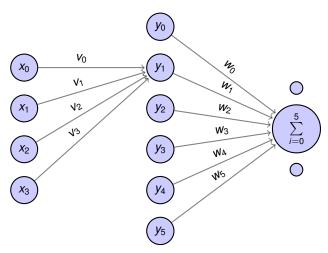




Gewichtung der Ausgabewerte mit Kantengewicht



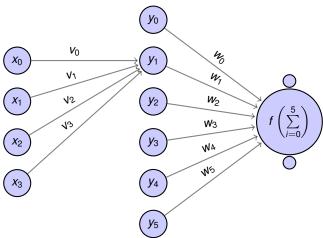




Summe der Produkte ergibt Netzeingabe in der Ausgabeschicht



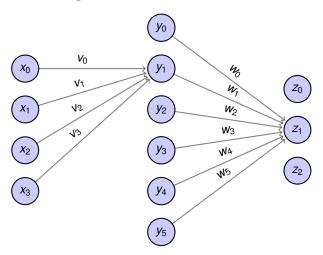




Netzeingabe der restlichen Neuronen der Ausgabeschicht







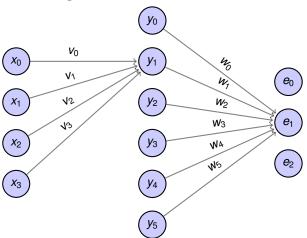
Umrechnung der Netzeingabe mit der Übertragungsfunktion



Einleitung

000000





 Berechnung des Ausgabefehlers anhand Ausgabewert und gewünschtem Ausgabewert

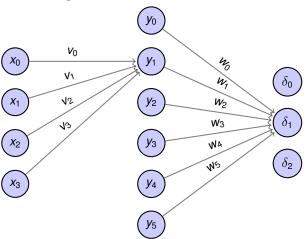


Parallelisierungsansätze

Evaluierung

000000

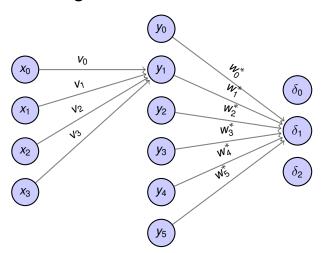




 Berechnung von delta-Werten anhand Netzeingabe, Übertragungsfunktion und Ausgabefehler



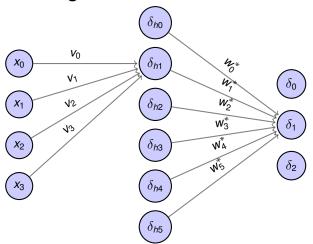




Anpassung der Kantengewichte



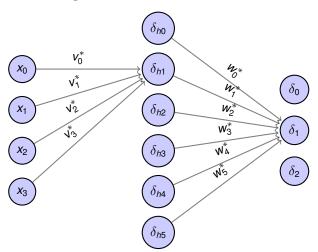




Berechnung von delta-Werten in Hiddenschicht (benötigt u.a. ausgehende Kantengewichte und delta-Werte der Ausgabeschicht)

Einleitung 000000



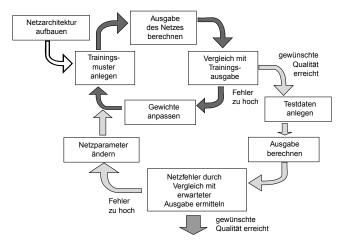


Anpassung der Kantengewichte



#### **Neuronale Netze**





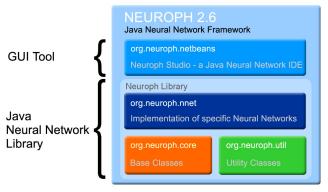
[Quelle: Lehr- und Übungsbuch künstliche Intelligenz; Lämmel, Cleve; 2012]



## Neuroph



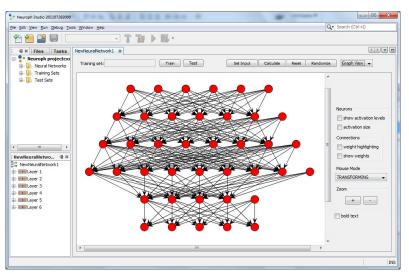
- Simulationsumgebung für Neuronale Netze
- 42751 LOC
- Apache 2.0 Lizenz
- http://neuroph.sourceforge.net/





## Neuroph

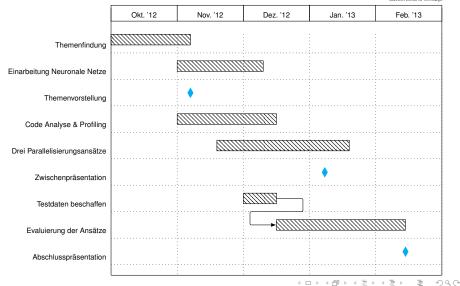






### Überblick





### Überblick

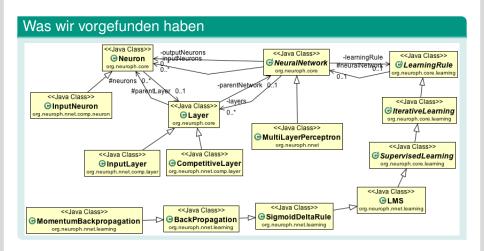


- Einleitung √
- Neuroph auf Code-Ebene
- Drei Parallelisierungsansätze
- Evaluierung
- Fazit



### Code Analyse I



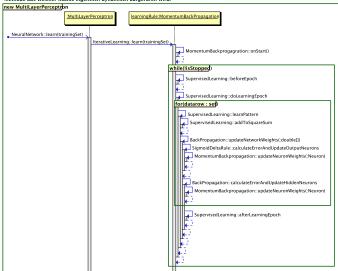




## Code Analyse II



Dieses Diagram beschreibt den Ablauf eines 'learn' Aufrufes auf einen Multilayerperceptron. Die Syntax :: zeigt an, die Methode aus welcher Klasse eigentlich dynamisch aufgerufen wird.





### Überblick



- Einleitung √
- Neuroph auf Code-Ebene ✓
- Drei Parallelisierungsansätze
- Evaluierung
- Fazit



## Parallelisierungsansätze

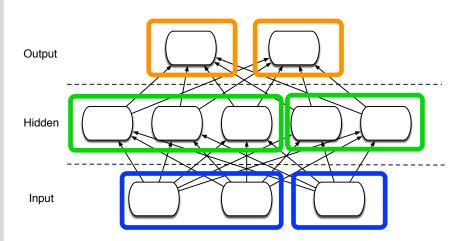


- Layer-Partitionierung
- Batch Learning Parallelisierung
- Clonebased Parallelisierung



## Layer-Partitionierung I







## Layer-Partitionierung II



#### Layer::calculate()

```
for (Neuron n : this.neurons)
    n.calculate();
```

#### ParallelLayer::calculate()

```
ExecutorService service = getExecutor();
Queue < Future <?>> futures = new LinkedList <>();
for (NeuronJob j : jobs)
    futures.add(service.submit(j));
waitForAll(futures);
```

M. Braun, D. Hammann, D. Messinger, D. Rausch - Paralellisierung neuronaler Netze

Parallelisierungsansätze

00000000000000

Evaluierung

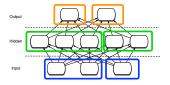
## Layer-Partitionierung III



- Neuer Netzwerktyp: Paralleles neurales Netzwerk
  - Profitiert von großen Netzwerken
  - Nicht von großen Datenmengen
- Fast alle gemessenen Speedups < 1</li>

#### Erster naiver Ansatz ineffektiv

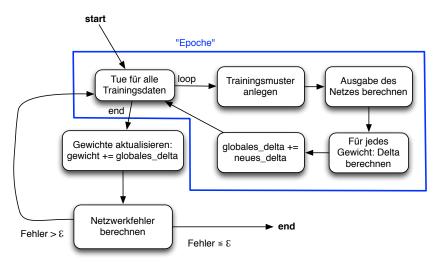
- Auswertung des NN sehr schnell
- Hotspot ist eine get-Methode!





#### **Exkurs: Batch lernen**

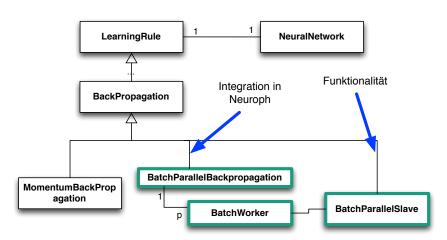






# **Batch Learning Parallelisierung I**





Paralleles Lernen statt parallelem Netzwerk



## **Batch Learning Parallelisierung II**



- Paralleler Lernvorgang (Netzwerk sequentiell)
- Vorteil:
  - Saubere, konsistente Schnittstelle
  - Zugriff auf protected Funktionalität
- Nachteil:
  - Lernalgorithmus fest
    - Backpropagation
  - Softwaretechnisch fragwürdig
    - Beispiel: Override mit leerem Body



# **Batch Learning Parallelisierung III**

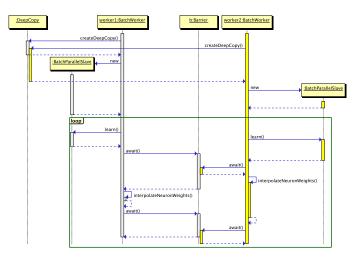


- Trainingsdaten auf Worker verteilt
- Worker lernen unabhängig
- Gewichtänderungen aufaddieren
  - Jeder Thread bearbeitet Subset
  - Dadurch keine Abhängigkeiten



# **Batch Learning Parallelisierung IV**







# Clonebased Parallelisierung I



#### Charakteristika

- Container für ein neuronales Netz mit überwachter Lernmethodik
- Unterteilt Trainingssatz und erlernt diese Teile parallel
- Abweichende Ergebnisgewichte verglichen mit sequentiellem Lernen



## Clonebased Parallelisierung II



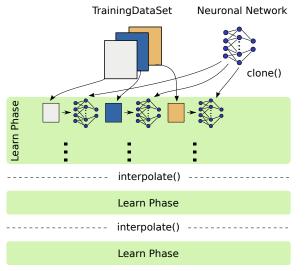
#### Vorgehen

- Aufteilung des Trainingssatzes auf Worker-Threads
- 2 Jeder Worker klont sich das Original-Netz
- 3 Jeder Worker lernt eine Iteration
- Gewichts-Interpolation unter allen Worker-Klonnetzen
- goto (3) bis maximale Iteration erreicht oder Fehler klein genug



# Clonebased Parallelisierung III







# Clonebased Parallelisierung IV



## Interpolationstypen

 $w = f(w_1, w_2, w_3, ..., w_n)$ 

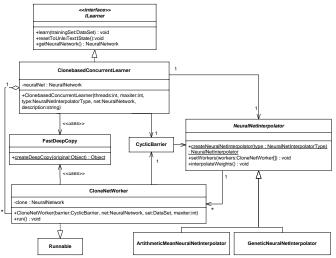
Gesucht: gute Interpolationsfunktion f

#### Kandidaten:

- Arithmetisches Mittel
- Minimum, Maximum
- Bestes bisheriges Gewicht ("Genetischer Ansatz")

## Clonebased Parallelisierung V

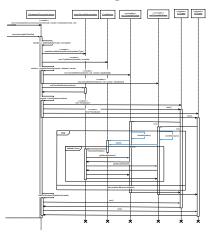






# **Clonebased Parallelisierung VI**





Siehe Handout



# Technischer Vergleich der Strategien



#### Klonbasiertes Lernen

- Kapselt vorhandene Funktionialität
  - Verwendung mit verschiedenen Lernstrategien und Netzen möglich

#### Paralleles Batchlernen

- Neue LearningRule in Vererbungshierarchie
  - saubere Schnittstelle, einfache Verwendung

Kategorie	Layerpartitionierung	Batch	Clonebased
Was ist parallel?	Netzwerk	Lernen	Lernen
Selbes Ergebnis?	ja	ja (batch)	nein
Integration	Modifikation	Erweiterung	Kapselung



# Technischer Vergleich der Strategien



#### Klonbasiertes Lernen

- Kapselt vorhandene Funktionialität
  - Verwendung mit verschiedenen Lernstrategien und Netzen möglich

#### Paralleles Batchlernen

- Neue LearningRule in Vererbungshierarchie
  - saubere Schnittstelle, einfache Verwendung

Kategorie	Layerpartitionierung	Batch	Clonebased
Was ist parallel?	Netzwerk	Lernen	Lernen
Selbes Ergebnis?	ja	ja (batch)	nein
Integration	Modifikation	Erweiterung	Kapselung



## Überblick



- Einleitung √
- Neuroph auf Code-Ebene √
- Drei Parallelisierungsansätze √
- Evaluierung
- Fazit



### **Testdaten**



#### Erste Versuche

- StockExchange Börsenvorhersage
- IrisScan Datensatz

#### Teilchenkollision (Cern)

- 15k Datensätze
- Eingabe: 2853 Sensorwerte
- Ausgabe: Ist das Ereignis interessant oder nicht?



## **Evaluationsframework I**



#### Score

wird bestimmt durch

- Fehler (auf Testdaten)
- Laufzeit



### **Evaluationsframework II**



## Vorgehen

- Parse Experiment-Konfigurationsdatei
- 2 Bereite Testläufe vor
- g repeat
  - Permutation der Daten
  - Aufteilung in Trainings- und Testdaten
  - foreach ILearner L do
    - Lerne Trainingsdaten und messe Ausführungszeiten
    - ② Berechne Fehler auf Testdaten
- until alle Durchgänge absolviert



## **Evaluationsframework III**



java -jar runexperiment.jar -cf testconfig.txt -o results/ -v -d -csv

#### **Experiment-Konfiguration**

threads: 4

dataset: data/preparedData/15000rows.txt

training\_to\_test\_ratio: 0.5 input neurons: 2853

hidden\_neurons: 100

output\_neurons: 1

runs: 10 max iteration: 2

sync frequency:0.25

learners: clonebased-arithmeticmean, clonebased\_revised-arithmeticmean, mlp



NAME; NUMTHREADS; RUNS; AVGER; CONFERLO; CONFERLI; AVGTIME; CONFTIMELO; CONFTIMEHI; SUMER; SUMTIME Clonebased Concurrent Learner; 4;10;0,179106; 0,08943; 10,26878; 211762; 1343939; 288584; 1,791057; 2117624 Clonebased Concurrent Learner Revised; 4;10;0,158935; 0,105018; 0,212852; 223762; 187061; 260463; 1,589349; 2237627 Neural Network Wrapper; 4;10;0,163642; 0,064823; 0,262460; 528646; 502374; 554917; 1,636419; 2286460

#### Ergebnis als CSV



## **Testaufbau**



#### Poolrechner

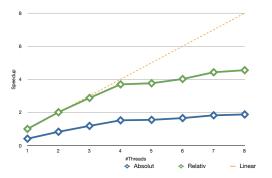
- Intel Core2Quad Q6600, 4 Kerne à 2,4 GHz, 8MB L2 Cache
- 8GB RAM
- JDK 7 (-Xms4096m -Xmx8000m)



# Messwerte: Speedup



- Ausführung des sequentiellen und des parallelen Batch-Learners
- 1000 Datenreihen aus CERN-Testdaten, 1:1 Trainings-/Testdaten, 10 Wiederholungen

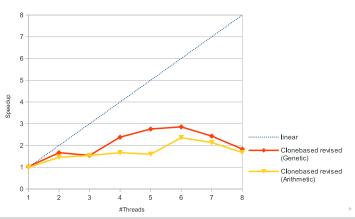




# Messwerte: Speedup



- Ausführung des sequentiellen MLPs und der klonbasierten Ansätze
- 5000 Datenreihen aus CERN-Testdaten, 1:1 Trainings-/Testdaten, 3 Wiederholungen



Einleitung

Neuroph auf Code-Ebene

Parallelisierungsansätze

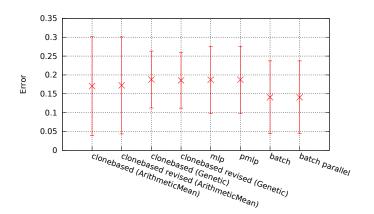
Evaluierung

90 Q

## **Messwerte: Error**



 1000 Datenreihen aus CERN-Satz, 1:1 Trainings-/Testdaten, 10 Wiederholungen





## Überblick



- Einleitung ✓
- Neuroph auf Code-Ebene ✓
- Drei Parallelisierungsansätze √
- Evaluierung √
- Fazit

## **Fazit**



- Neuronale Netze geeignet f
  ür Parallelisierung
- Trivialer Lösungsansatz nicht möglich
- Codebasis (Neuroph) nicht vorbereitet
- Fremden Code bearbeiten sehr aufwändig



