Technische Universität Ilmenau
Fakulät IA
Fachgebiet Neuroninformatik

Praktikum Neuroinformatik
WS 2021/22

Versuchsprotokoll

Neuroinformatik

20. Oktober 2021

1 Vorbereitung: Virtuelle Python Umgebung

Ordner erstellen mit mkdir

dann virtuelle Umgebung mit python3 −m venv env

Dieser Ordner enhält daraufhin:

- bin: Dateien die mit der virtuellen Umgebung interagieren
- include: C Header Dateien um Python Pakete zu kompilieren
- lib: eine Kopie der Python version mit Abhängigkeiten

Aktiviere die virtuelle paket isolation source env/bin/activate

Zum deaktivieren deactivate

Pip Pakete installieren:

```
pip install jupyterlab notebook spyder setuptools matplotlib scikit-learn num
```

NeuroInformatik Pakete runterladen:

pip install tui−ni−practical−course −−upgrade −−extra−index−url <URL−vom−Fachgebiet>‘

danach Anaconda herunterladen und installieren:

bash ~/Downloads/Anaconda3−2020.02−Linux−x86_64.sh

Aufgabe 1a: Kohonen Kette

Bash:tui−ni−kohonen−chain

1.

Wählen Sie „L-shape" als Datenverteilung aus (Dieser Input wird für die Aufgaben 1-5 verwendet). Starten Sie die Simulation („Start") und beobachten Sie die Bewegungen der Gewichtsvektoren im Inputraum. Stoppen Sie die Simulation, wenn sich die Gewichtsvektoren stabilisiert haben.

mit ca 300 Schritten passt sich das Kohonen Netz der Datenverteilung an

2.

Stoppen Sie die letzte Simulation („Stop") und initialisieren Sie die Kohonen-Kette neu („Reset"). Beobachten Sie die Bewegung der Gewichtsvektoren bei Nutzung des Einzelschrittbetriebs („Step") für ca. 20 Schritte. Beobachten Sie dabei, wie die Neuronen in Abhängigkeit des Inputs und der Nachbarschaftsfunktion aktiviert werden (rote Färbung).

- nähern sich immer weiter an
- bilden eine Linie
- ab ca Schritt 10 gleichen sie sich der L-Shape an

3.

Vergleichen Sie die Ergebnisse mit denen von Kohonen-Ketten, die die Lernraten $\eta = 0, \eta = 1, \eta = 2$ *sowie* $\eta = -0.5$ *nutzen (Nach jeder Änderung der Lernrate das Netzwerk zurücksetzen („Reset")). Worin bestehen die wesentlichen Unterschiede der entsprechenden Netze? Welche Lernrate ist in diesem Beispiel für einen praktischen Einsatz am geeignetsten und warum? Traten topologische Defekte auf: falls ja, wodurch wurden sie verursacht?*

- $\eta = 0$
  - das Kohonen Netz wird nicht angepasst/verändert
  - die Nachbarschaftsfunktion ändert sich
- $\eta = 1$
  - mit 6 schritten in einer reihe
  - Kohonen Kette springt immer wieder über L-Ecke hinaus
  - erst später richtige anpassung an L-Shape, ca 240 Schritte
- $\eta = 2$
  - sehr große Sprünge der gesamten Kette
  - viele Überkreuzungen der Nachbarschaften
  - springt aus L Shape heraus
- $\eta = -0.5$

  - Kohonen Karte wird immer größer
  - ab ca 20 Schritten Karte nicht mehr zu sehen

am geeignetsten hier: $\eta = 1$, da schnellste (überhaupt) anpassung

4.

Initialisieren Sie („Reset") eine Kohonen-Kette mit den folgenden Parametern: 20 Neuronen, Lernrate $\eta = 0.2$, Decay Lernrate $\delta\eta = 0.999$, Lernradius $r = 10$ und Decay Lernradius $\delta r = 0.99$. Starten Sie die Simulation im Einzelschrittbetrieb („Step"). Verfolgen Sie die Entwicklung der Gewichtsvektoren.

- Ab ca 400 Schritten komplette Anpassung an L-Shape
- langsame anpassung an Shape
- ab 50 schritten schon ungefähr L-Shape, nur nicht ausgefüllt

Vergleichen Sie die Ergebnisse mit denen von Kohonen-Ketten, die die Lernradien $r = 100$ sowie $r = 0.001$ nutzen! Worin bestehen die wesentlichen Unterschiede der entsprechenden Netze? Welcher Lernradius ist in diesem Beispiel für einen praktischen Einsatz am geeignetsten und warum? Traten topologische Defekte auf: falls ja, wodurch wurden sie verursacht?

- $r = 100$
  - bildet erst punkt
  - ab ca 100 schritten bildet sich linie
  - ab ca 400 anpassung an L Shape, ab ca 600 fertig
- $r = 0,001$
  - passt nur gewichtsvektoren an die schon im L Shape sind, weiter entfernte bleiben außerhalb des Netzes
  - viele Überkreuzungen möglich

hier am praktischsten: $r = 10$, hat schon früh eine relaitv gute anpassung an L-Shape

5.

Initialisieren Sie („Reset") eine Kohonen-Kette mit den folgenden Parametern:

- *20 Neuronen, Lernrate* $\eta = 0.2$*, Decay Lernrate* $\delta\eta = 0.9$*, Lernradius* $r = 10$ *und Decay Lernradius* $deltar = 0.99$ *sowie*
- *20 Neuronen, Lernrate* $\eta = 0.2$*, Decay Lernrate* $\delta\eta = 0.999$*, Lernradius* $r = 10$ *und Decay Lernradius* $\delta r = 0.5$*.*

Starten Sie die Simulation im Einzelschrittbetrieb („Step"). Beobachten Sie die Adaptation der Gewichtsvektoren! Worin liegen die Ursachen der aufgetretenen Probleme?

1. nach ca 100 Schritten bewegt sich nichts mehr
2. sehr schneller abfall der anpassung, nach weniger als 100 Schritten keine anpassung mehr
3. zu schneller abbau der Lernrate; Lernradius zu klein, sodass weiter entfernte Gewichtsvektoren nicht mehr beachtet werden

6.

Initialisieren Sie („Reset") eine Kohonen-Kette mit 20 Neuronen sowie den Parametern Lernrate $\eta = 0.2$*, Decay Lernrate* $\delta\eta = 0.999$*, Lernradius* $r = 10$ *und Decay Lernradius* $\delta r = 0.99$*. Wählen Sie als Inputpattern „equally distributed" und starten Sie die Simulation („Start"). Welches grundsätzliche Problem ist bei der Nutzung einer Kohonen-Kette in einem zweidimensionalen Inputraum zu berücksichtigen?*

passt sich dem 2D Raum an, jedoch jedes mal anders, da es keinen richtigen orientierungspunkt hat.

7.

Initialisieren Sie („Reset") eine Kohonen-Kette mit 20 Neuronen sowie den Parametern Lernrate $\eta = 0.5$*, Decay Lernrate* $\delta\eta = 0.999$*, Lernradius* $r = 10$ *und Decay Lernradius* $\delta r = 0.99$*. Wählen Sie als Inputpattern „Omega-shape" und starten Sie die Simulation 3 mal („Reset"+„Start"). Überlegen Sie, warum sich die Nachbarschaftsbeziehung der Neuronen in der Kette in diesem Fall als hinderlich für den Lernprozess erweist?*

durch die verkettung benötigt es mehr schritte und passt sich nicht genau an

Aufgabe 1b: Neural Gas

tui−ni−neural−gas

1.

Wählen Sie „Omega-shape" als Datenverteilung aus. Starten Sie die Simulation („Start") und beobachten Sie den Lernprozess. Welche Unterschiede ergeben sich im Vergleich zu dem Lernprozess der Kohonen-Kette. Betrachten Sie dabei auch die Aktivierung der Neuronen im Einzelschrittbetrieb.

- nach ca 200 Schritten fertig, wesentlich schneller als Kohonen Kette - zieht sich Anfangs nicht so stark zusammen wie kohonen kette

2.

Wählen Sie „Travelling Salesman" als Datenverteilung und vergegenwärtigen Sie sich welches Problem dabei gelöst werden soll: Problem des Handlungsreisenden. Initialisieren Sie („Reset") ein Neural Gas mit 20 Neuronen sowie den Parametern Lernrate $\eta = 0.2$*, Decay Lernrate* $\delta\eta = 0.999$*, Lernradius* $r = 10$ *und Decay Lernradius* $\delta r = 0.99$*. Können Sie mit dem Netzwerk eine Lösung für das Problem finden?*

- ab 400 Schritten kaum mehr bewegung, die meisten (bis auf vier) sind auf die „Orte" verteilt
- kürzester Weg kann so nicht gefunden werden da es keine verknüpfung/reihenfolge zwischen den orten gibt

3.

Initialisieren Sie („Reset") ein Neural Gas mit den folgenden Parametern: 20 Neuronen, Lernrate $\eta = 0.5$*, Decay Lernrate* $\delta\eta = 0.999$*, Lernradius* $r = 10$ *und Decay Lernradius* $\delta r = 0.99$*. Wählen Sie als Inputpattern „slowly moving rectangle" und starten Sie die Simu-*

lation („Start"). Welche Parameter müssen Sie adaptieren, damit sich die Gewichtsneuronen an den sich ständig ändernden Input anpassen können?

- anfangs gute anpassung an viereck, jedoch mit vortschreitenden schritten keine anpassung mehr
- Lernrate/Lernradius Decay muss auf „1" gesetzt werden, um stetige anpassung aller Punkte zu garantieren

Aufgabe 2a: Transferfunktion

tui−ni−transfer−function

1.

Setzen Sie bitte die Gewichte auf die Werte $w_0 = 0.5$ und $w_1 = 0.5$. Machen Sie sich durch unterschiedliche Wahl von Aktivierungs- und Ausgabefunktion mit der Bedienung des Applets vertraut. Variieren Sie dabei auch den Parameter „Classification threshold" auf der rechten Seite der Visualisierung, um die Auswirkung der virtuellen Trennebene (Schwellwert über die Ausgabe des Neurons) für die Lösung eines Zweiklassenproblems zu untersuchen.

2.+3.

Aktivieren Sie die Checkbox „Dataset classification" um den ersten Datensatz in die Visualisierung einzublenden. Wählen Sie als Aktivierungsfunktion „Dot product" und als Ausgabefunktion „linear". Variieren Sie die Gewichte des Neurons (w0,w1,bias) und den Klassifikationsschwellwert („Classification threshold") um eine korrekte Klassifikation der Datenpunkte zu erreichen. Bei dem grünen Datenpunkten muss die Ausgabe des Neurons oberhalb der Klassifikationsschwelle liegen und bei den roten Datenpunkten unterhalb der Schwelle. Korrekt klassifizierte Datenpunkte werden vergrößert dargestellt. Wählen Sie nacheinander die Datensätze 2 bis 5 und kombinieren Sie jeweils geeignete Aktivierungs- und Ausgabefunktionen, um die Daten, mit dem einen realisierten Neuron, jeweils korrekt in zwei Klassen zu trennen.

Datensatz	w_0	w_1	bias	Threshold	Aktivierungsfunktion	Ausgabefunktio
Point Set 1	$0,5$	$0,5$	0	0	Dot Product	Linear
Point Set 2	0,5	0,5	0	0,18	Dot Product	Gaussian
Point Set 3	0,5	0,5	0	0	Min Distance	Linear
Point Set 4	0,5	0,5	0	0,27	Mahalanobis, $\phi = 143$, $\sigma_0 = 0$, $\sigma_1 = 12.5$	Symetric Sigmoi
Point Set 5	0,5	0,5	0	0	Mahalonobis, $\phi = 143$, $\sigma_0 = \infty$, $\sigma_1 = 3.57$	Gaussian, $\sigma = 0.$

4.

Wenn alle Datensätze korrekt klassifiziert wurden, wird der „Submit"-Button aktiviert. Klicken Sie auf den „Submit"-Button und geben Sie in dem sich öffnenden Fenster Ihre E-Mail-Adresse ein. Klicken Sie anschließend auf „Generate!" und kopieren Sie den angezeigten, kodierten (nicht lesbaren) Text, welcher die von Ihnen bei der Bearbeitung eingestellten Parameter enthält. Diesen Text reichen Sie in der dazugehörigen Aufgabe in Moodle ein. Bitte beachten Sie dabei die Einreichungsfrist im Moodle-Kurs.

Aufgabe 2b: Delta Regel

tui−ni−delta−rule

1.

Wählen Sie bitte die „Teacher function" „y=x" und die „Output function" „Linear"! Starten Sie die Simulation im Schrittbetrieb („Step")! 1. Beobachten Sie dabei den Verlauf der Fehler E(t) bzw. E(w) sowie des Gradienten von E(w) in Abhängigkeit der präsentierten Inputs! Wodurch entsteht der periodische Verlauf der beiden Fehlerfunktionen? der Input pendelt periodisch zwischen ± 1, das Neuron versucht entgegenzusteuern

2. Initialisieren Sie das Netz neu („Reset") und wählen Sie eine „Batch size" von „11 samples"! Beobachten Sie erneut den Verlauf der Fehler E(t) bzw. E(w) sowie des Gradienten von E(w) im Einzelschrittbetrieb („Step")! Starten Sie nach etwa 30 Lernschritten den Trainingsprozess („Start"). Welcher Unterschied ergibt sich für den Verlauf des Fehlers E(t)?

- Fehler fallen exponentiell ab
- kein Periodischer Verlauf der Fehler mehr

- Neuron passt sich immer weiter den Input-Werten an

2.

Stoppen Sie bitte die letzte Simulation und initialisieren Sie das Netz neu („Reset")! Wählen Sie „$y = |x|$" als „Teacher function" und starten Sie den Lernprozess („Start")! Warum konvergiert der Fehler nicht gegen den Wert Null? Wie kann dieses Problem behoben werden?

- Fehler E_t konvergiert gegen $0,4$
- Fehler E_w konvergiert gegen $0,6$
- Ausgabefunktion ReL nähert den Fehler E_w an $0,2$ und E_t immer weiter an 0 an

3.

Wählen Sie die lineare Ausgabefunktion und aktivieren Sie die Checkbox „Use own output implementation". Da diese Ausgabefunktion die Aktivierung identisch weiterleitet, müssen Sie dem y-Wert einfach die Aktivierung zuweisen. Tragen Sie dazu bitte: $y = z$ ein. Klicken Sie anschließend auf „Check code". Bei einer korrekten Umsetzung wird die Textbox grün hinterlegt.

4.

Aktivieren Sie die Checkbox „Use own derivative". Wenn Sie die Funktion $y = z$ nach z ableiten, erhalten Sie die Ableitung von 1. Weisen Sie daher der Variablen $dy_{dz} = 1$ zu. Klicken Sie wieder auf „Check code" um Ihre Umsetzung zu testen.

5.

Wählen Sie „Sigmoid" als Ausgabefunktion. Aktivieren Sie die Checkbox „Use own output implementation" und geben Sie die Fermi-Ausgabefunktion ein (siehe Skript Neuroinformatik). Die erfolgreiche Umsetzung wird durch einen Haken hinter „Sigmoid output function" im unteren Bereich gekennzeichnet. Geben Sie anschließend auch die Gleichung für

die Ableitung der Fermi-Funktion ein (Hinweis: die Ableitung enthält y, welches in diesem Code-Fenster noch nicht berechnet vorliegt).

- Implementation: $y = 1/(1 + exp(-c * z))$
- derivate: $dy_dz = c * y(1 - y) = c * (1/(1 + exp(-c * z)))(1 - (1/(1 + exp(-c * z))))$

6.

Wählen Sie „ReLU" als Ausgabefunktion und setzen Sie die Ausgabefunktion und deren Ableitung um. Beachten Sie für die Ableitung, dass sich die ReL-Funktion, abhängig von z, wie eine Konstante oder wie eine lineare Ausgabefunktion verhält. Die Umsetzung von If-Abfragen finden Sie am Ende dieser Seite.

- Implementation: $y = max(0, z)$
- derivate:

```
if z<= 0:
    dy_dz=0
else:
    dy_dz=1
```

7.

Wenn alle Funktionen korrekt umgesetzt wurden, wird der „Submit"-Button aktiviert. Klicken Sie auf den „Submit"-Button und geben Sie in dem sich öffnenden Fenster Ihre E-Mail-Adresse ein. Klicken Sie anschließend auf „Generate!", kopieren Sie den angezeigten, kodierten (nicht lesbaren) Text, welcher die von Ihnen bei der Bearbeitung eingestellten Parameter. Reichen Sie diesen Text über die dazugehörige Aufgabe in Moodle ein. Bitte beachten Sie dabei die Einreichungsfrist im Moodle-Kurs.