Dokumentation Projekt: Neuronales Netzwerk

Aufgabe:

Ziel ist es ein neuronales Netzwerk selbst zu implementieren, welches den MNIST Datensatz für handgeschriebene Ziffern identifiziert.  
Hierfür soll das Netzwerk zunächst mit einem Trainingsdatensatz trainiert und anschließend mit dem Testdatensatz getestet werden.  
Das neuronale Netz liest nacheinander Bildinformationen ein und ändert anhand des herauskommenden Fehlers seine Gewichte in der Backpropagation. Zum Schluss werden die Gewichte gespeichert.

Vorbereitung:

Im Zuge der Vorbereitung habe ich ein Klassendiagramm erstellt, mit dem ich dann in die Implementierung gegangen bin. Das Diagramm ist im Anhang zu finden.

Die Klasse NeuralNetwork steht hier im Zentrum. Es nutzt die Klassen Matrix, ActivationFunction, SignalProcessor, BackPropagator und WeightUpdater um die nötigen Schritte zu managen und den übernimmt die Übergaben der einzelnen Arbeitseinheiten aneinander.

Die Main-Methode liest am Anfang den gesamten Datensatz mit dem IdxFileReader ein und speichert ihn in der Klasse DataSet und übergibt dieses an das Neurale Netzwerk, bevor es gestartet wird. Als Programmargumente sind die Pfade zu den nötigen Dateien sowie ein Modusargument, welches das Netzwerk auf Training oder Test setzt.

Die Idee ist es, das Neuronale Netzwerk aus den Neuronenlayern und den Gewichten dazwischen zusammenzusetzen. Die Aufteilung in die drei Arbeitseinheiten SignalProcessor, BackPropagator und WeightUpdater geschah, um die Klasse nicht zu groß werden zu lassen.

Unterschiede zum Klassendiagramm:

Das Diagramm befindet sich im Anhang.

Bei der Implementierung haben sich einige Änderungen ergeben. Die Neuronenstruktur (layers) wird nicht mehr in einem zweidimensionalen Neuronen-Array repräsentiert, sondern in einem Matrix-Array – die Klasse Neuron wurde hierbei auch gelöscht. Grund dafür war, dass die Neuronen keine Funktionalität benötigten, da alles mit Matrixoperationen berechnet werden kann.

DataSet wurde zu einem Record umgewandelt, da es, wie der Name schon verrät, sich hierbei nur um eine Datensammlung hält.

Die Klasse Matrix wurde um benötigte Funktionen erweitert.

Die Arbeitseinheiten haben ein Interface dazwischengeschaltet bekommen, um besseren Programmierstandards zu entsprechen.

Ablauf:

Nach dem Start des Neuronalen Netzwerkes mit dem entsprechenden Datensatz wird es mit den entsprechenden Parametern initialisiert. Folgende Parameter nimmt das Netzwerk entgegen:

Funktion: Die übergebene Funktion wird als Aktivierungsfunktion bei der Signalverarbeitung verwendet.

Modus: Die möglichen Modi ist „train“ und „test“. Beim Training wird die Backpropagation durchgeführt und die Gewichtsveränderungen gespeichert.

Anzahl der Neuronen pro Layer: Legt die Anzahl der Neuronen pro Hidden Layer fest.

Lernrate: Legt die Lernrate für die Backpropagation fest.

Wiederholungen: Bestimmt, wie oft der komplette Datensatz durch das Netzwerk gehen soll.

Schritt für Schritt wird ein MNIST-Bild als Signal ausgelesen und dieses an den Signalprozessor übergeben. Dieser übernimmt die Hinführung von der Inputschicht bis hin zur Outputschicht.

Die Backpropagation vergleicht das Ergebnis der Outputschicht mit dem MNIST-Label und berechnet den Fehler. Dieser Fehler wird nun auf die einzelnen Gewichte zurückgeführt und es werden entsprechende Änderungen an ihnen vorgenommen.

Das Gewichtsmanagement speichert zum Schluss die neuen Gewichte, sofern sich das Netzwerk im Training befindet.

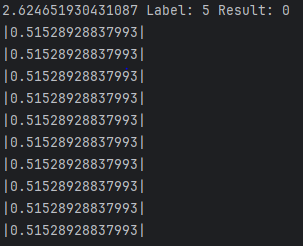
In jedem Schritt gibt die Backpropagation das Ergebnis der Fehlerfunktion, der erwartete Wert, das tatsächliche Ergebnis und die komplette Outputschicht aus.

Ziel ist es, den Output an der Labelzahl möglichst nah an 1 und alle anderen möglichst nah an 0 zu bekommen.

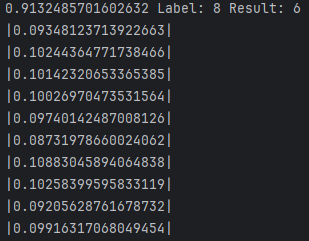
Ergebnis:

Das Netzwerk arbeitet wie vorgesehen, der Fehler verringert sich mit jedem Schritt:

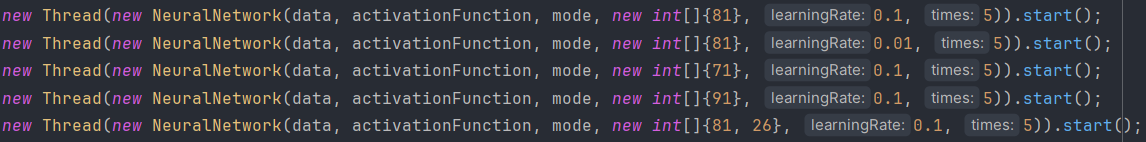
Fehler nach dem ersten Bild (2,625):



Fehler nach dem letzten Bild (0,913):



Nach jedem Durchlauf wird die Trefferrate besser. Verschiedene Voreinstellungen bringen verschiedene Fortschritte:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thread-Nr. | Hiddenlayer-  Neuronen | Lernrate | Trefferrate nach 1. Durchlauf in % | Treff. 2. D. in % | Treff. 3. D. in % | Treff. 4. D. in % | Treff. 5. D. in % |
| 1 | 81 | 0,1 | 92,68 | 96,01 | 96,90 | 97,51 | 97,88 |
| 2 | 81 | 0,01 | 86,76 | 92,25 | 93,57 | 94,31 | 94,88 |
| 3 | 71 | 0,1 | 92,44 | 95,70 | 96,62 | 97,15 | 97,63 |
| 4 | 91 | 0,1 | 92,78 | 96,22 | 97,10 | 97,65 | 98,03 |
| 5 | 81, 26 | 0,1 | 88,52 | 90,79 | 91,29 | 91,29 | 91,66 |

Die Gewichtsanpassungen werden in Binärdateien gespeichert. Mit den Testbildern erreichen die 5 Netzwerke folgende Ergebnisse:

Netzwerk 1: 96,76 %  
Netzwerk 2: 96,76 %  
Netzwerk 3: 96,26 %  
Netzwerk 4: 97,03 %  
Netzwerk 5: 91,63 %

Alle Netzwerke erzielen mit den Training- und Testdaten ähnliche Trefferraten von über 90 %.  
Das Programm ist somit ein Erfolg.