**НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ**

**БАКЪЛАВЪРСКИ ФАКУЛТЕТ**

**ДЕПАРТАМЕНТ "ИНФОРМАТИКА"**

**ПРОГРАМА** **ИНФОРМАТИКА**

**Проект по невронни мрежи**

**КУРСОВА РАБОТА**

**ТЕМА: Невронна мрежа за разпознаване и класифициране на ръкописни цифри**

**НА СТУДЕНТА: Илиян Делчев Додеков**

ФАК. № 50129

Пролетен Семестър

2014-2015 г.

Съдържание

[Въведение 4](#_Toc418809654)

[I. Задание 4](#_Toc418809655)

[II. Резултати 5](#_Toc418809656)

[Какво е невронна мрежа 6](#_Toc418809657)

[MNIST база от ръкописни изображения на цифри 7](#_Toc418809658)

[Типове изображения 9](#_Toc418809659)

[Реализация 10](#_Toc418809660)

[I. Изкуствен неврон 10](#_Toc418809661)

[ Суматор 11](#_Toc418809662)

[ Активационна функция 12](#_Toc418809663)

[II. Структура на мрежата 12](#_Toc418809664)

[III. Обучение на мрежата 13](#_Toc418809665)

[ Стъпка 1: Инициализиране на входовете 13](#_Toc418809666)

[ Стъпка 2: Намиране на изхода 13](#_Toc418809667)

[ Стъпка 3: Намиране на грешката (делтата) 14](#_Toc418809668)

[ Стъпка 4: Промяна тежестите и праговите коефиценти 15](#_Toc418809669)

[ Online стратегия 15](#_Toc418809670)

[ Batch стратегия 16](#_Toc418809671)

[ Момент 17](#_Toc418809672)

[ Стъпка 5: Проверка за претрениране (overfitting) 18](#_Toc418809673)

[ Стъпка 6: Повторение 19](#_Toc418809674)

[IV. Опериране с мрежата 19](#_Toc418809675)

[Опериране с приложението 20](#_Toc418809676)

[I. Файлове 20](#_Toc418809677)

[II. Конфигурация 21](#_Toc418809678)

[III. Опериране 21](#_Toc418809679)

[ Създаване на мрежа 23](#_Toc418809680)

[ Зареждане на вече създадена мрежа от файл 24](#_Toc418809681)

[ Записване на мрежа във файл 24](#_Toc418809682)

[ Обучение на мрежа 24](#_Toc418809683)

[ Тестване на мрежа 26](#_Toc418809684)

[ Пресмятане на процента на грешка 29](#_Toc418809685)

[ Квадратична грешка на тренировъчните данни 30](#_Toc418809686)

[ Квадратична грешка на валидационните данни 30](#_Toc418809687)

[Тестове 32](#_Toc418809688)

[I. Online трениране с момент и проверка за претрениране 32](#_Toc418809689)

[ 1 епоха 33](#_Toc418809690)

[ 2 епохи 35](#_Toc418809691)

[ 6 епохи 36](#_Toc418809692)

[II. Online трениране с проверка за претрениране и БЕЗ момент 38](#_Toc418809693)

[III. Online трениране БЕЗ проверка за претрениране и БЕЗ момент 39](#_Toc418809694)

[IV. Batch трениране с момент и проверка за претрениране 41](#_Toc418809695)

[ 4 епохи 42](#_Toc418809696)

[ 8 епохи 42](#_Toc418809697)

[V. Batch трениране с момент и БЕЗ проверка за претрениране 44](#_Toc418809698)

[Изпозлвани източници 47](#_Toc418809699)

# Въведение

## Задание

Проектът представлява реализирана невронна мрежа с три слоя за разпознаване и класифициране на ръкописни цифри. Използван е Java, като език за имплементацията.

Особености на мрежата:

* Възможност за подаване на входни данни от файлове. Всяко изображение е 28х28 пиксела със стойност от 0 до 255.
* За обучение и тестване на мрежата е ползвана MNIST (Modified National Institute of Standards and Technology) базата от ръкописни цифри.
* Три слоя – един входен, един скрит и един изходен. 784 неврона във входния слой; 387 неврона в скрития слой; 10 неврона в изходния слой. Възможност за промяна на броя на невроните посредством конфигурационен файл.
* Възможност за обучение на мрежата по алгоритъма за обратно разпространение на грешката (backpropagation).
* Възможност за избор на обучителна стратегия – batch или online
* Активационната функция е сигмоидална (f(x)=1/(1+e-х)))
* Коефицент на обучение – 0,5, с възможност за промяна посредством конфигурационен файл.
* Възможност за ползване на момент при обучение – 0,5, с възможност за промяна посредством конфигурационен файл.
* Възможност за записване на вече обучена мрежа във файл, за по-нататъшна употреба.
* Възможност за визуализиране на квадратична грешка
* Възможност за работа с тренировъчни, валидационни и тестови данни.
* Възможност за прекратяване на обучението против претрениране (overfitting), посредством валидационен набор от данни.
* Възможно за задаване на seed за случайни числа.

## Резултати

В MNIST базата от данни има 3 типа изображания – 60 000 броя, които се ползват за трениране на мрежата, 5 000 броя за тестване на мрежата и още 5 000 броя валидационни данни, които предпазват мрежата от претрениране. Важно е да се отбележи, че в трите типа изображения няма цифри, които се повтарят.

С горепосочената мрежа, след около 10 цикъла на обучение с 60 000 изображения, е реализиран коефициент на грешка **9,68%**. След този момент мрежата започва да се претренира. Ако проверката за претрениране се премахне, реализирания коефициент на грешка стига до около **1,94%**.

# Какво е невронна мрежа

Невронната мрежа е модел за обработка на информация, вдъхновен от изучаването на биоелектричните мрежи в мозъка на човека и животните, образувани от неврони и техните синапси. В наши дни учените често наричат изкуствените невронни мрежи просто невронни мрежи.

Математическия аналог на биологичната невронна мрежа представлява множество от взаимносвързани прости изчислителни елементи (неврони). Всеки неврон приема сигнали от другите (под формата на числа), сумира ги, като сумата минава през активационна функция (най-често използваната е сигмоидалната функция y=f(x)=1/(1+e-х)), и така определя своята активация (степен на възбуда), която се предава по изходящите връзки към другите неврони. Всяка връзка има тегло, което умножавайки се със сигнала, определя неговата значимост (сила). Теглата на връзките са аналогични на силата на синаптичните импулси, предавани между биологичните неврони. Отрицателна стойност на теглото съответства на подтискащ импулс, а положителна - на възбуждащ.

В невронната мрежа обикновено винаги съществуват входен и изходен слой от неврони, във входния се въвежда информацията към мрежата, след това сигналите от входните неврони преминават през един или няколко слоя от междинни (скрити) неврони, според топологията на невронната мрежа, като сигналите накрая стигат до изходния слой, откъдето се чете получената информация.

Математически е доказано, че всяка невронна мрежа с поне един скрит слой от достатъчно на брой неврони между входния и изходния слой, може да моделира поведението на всяка съществуваща функция.

Теглата на връзките между невроните определят функционалността и поведението на невронната мрежа. За да бъде една невронна мрежа използваема и приложима към даден проблем, тя трябва да бъде предварително изучена.

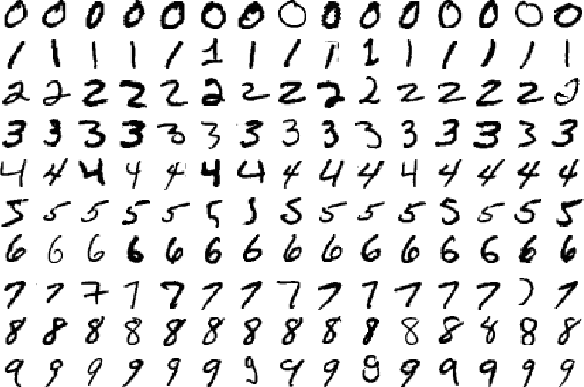
Изучаването на една невронна мрежа се постигне чрез промяна на теглата на връзките между невроните и се осъществява чрез правила, които определят как да се променят тези тегла. Най-разпространеното сред тях е метода на обратното разпространение на сигнал за грешка (back-propagation), където за всеки изходен неврон се изчислява разликата от желаното му поведение, като се формира сигнал за грешка, който се движи назад към входния слой и по пътя си променя теглата на връзките така, че при следващата активация на мрежата грешката да бъде по-малка от сегашната. Този начин на "обучение" на мрежата обаче води до "забравяне"- мрежата бъде обучена да разпознава един елемент и впоследствие той не се повтори във входните данни, мрежата "забравя" този елемент.

# MNIST база от ръкописни изображения на цифри

MNIST (Modified National Institute of Standards and Technology) базата от ръкописни изображения на цифри представлява набор от събрани цифри, написани с ръкописно от хора, за целите на машинното обучение. Базата представлява набор от 60 000 тернировъчни изображения и 10 000 тестови изображения. Всички тези изображения са нормализирани да бъдат с еднакъв размер и центрирани на базата на тежестта на изображението. Всяко от тях е 28х28 пиксела и представлява цифра от 0 до 9. Общо това прави 784 пиксела за всяко изображение. Важно е да се отбележи, че набора от тестови и тренировъчни цифри е различен и едното не е част от другото. Идеята е тренировъчният набор да се ползва за обучение, докато тестовият набор за проверка на точността и калкулиране на грешка, след като обучението приключи.

Повече за базата от изображения може да бъде прочетено на офицялния и сайт - <http://yann.lecun.com/exdb/mnist> .

Някой от изображенията в базата изглеждат по следния начин:



Всичките тестови и тренировъчни изображения са записани във 2 файла (общо 4). Така четенето става по-бързо и не се забавя допълнително обучителния процес. Във всеки комплект от файлове, единият съдържа пикселите на всички изображения, а другия – коя е цифрата. Това означава, че за всеки 784 пиксела в единия файл, има 1 цифра в другия файл (отношението е 1:784).

Всеки от файловете е бинарен и има следната структура:

**TRAINING SET LABEL FILE (train-labels-idx1-ubyte):**

[offset] [type]          [value]          [description]   
0000     32 bit integer  0x00000801(2049) magic number (MSB first)   
0004     32 bit integer  60000            number of items   
0008     unsigned byte   ??               label   
0009     unsigned byte   ??               label   
........   
xxxx     unsigned byte   ??               label

Стойностите варират между 0 и 9.

**TRAINING SET IMAGE FILE (train-images-idx3-ubyte):**

[offset] [type]          [value]          [description]   
0000     32 bit integer  0x00000803(2051) magic number   
0004     32 bit integer  60000            number of images   
0008     32 bit integer  28               number of rows   
0012     32 bit integer  28               number of columns   
0016     unsigned byte   ??               pixel   
0017     unsigned byte   ??               pixel   
........   
xxxx     unsigned byte   ??               pixel

Пикселите имат стойност от 0 до 255. 0 означава бяло а 255 означава черно.

**TEST SET LABEL FILE (t10k-labels-idx1-ubyte):**

[offset] [type]          [value]          [description]   
0000     32 bit integer  0x00000801(2049) magic number (MSB first)   
0004     32 bit integer  10000            number of items   
0008     unsigned byte   ??               label   
0009     unsigned byte   ??               label   
........   
xxxx     unsigned byte   ??               label

Стойностите варират между 0 и 9.

**TEST SET IMAGE FILE (t10k-images-idx3-ubyte):**

[offset] [type]          [value]          [description]   
0000     32 bit integer  0x00000803(2051) magic number   
0004     32 bit integer  10000            number of images   
0008     32 bit integer  28               number of rows   
0012     32 bit integer  28               number of columns   
0016     unsigned byte   ??               pixel   
0017     unsigned byte   ??               pixel   
........   
xxxx     unsigned byte   ??               pixel

Пикселите имат стойност от 0 до 255. 0 означава бяло а 255 означава черно.

Имплементираната мрежа не е задължително да работи точно с тези файлове, но е важно да се спазва структурата.

## Типове изображения

Както вече беше споменто, мрежата работи с 3 типа изображания – 60 000 броя, които се ползват за трениране на мрежата, 5 000 броя за тестване на мрежата и още 5 000 броя валидационни данни, които предпазват мрежата от претрениране.

**Тренировъчен тип**

Ползват се за регулиране на теглата в мрежата.

**Валидационен тип**

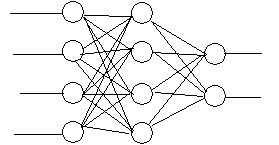
Ползват се за минимизиране на претренирането. С този набор не се регулират тежестите, а просто се проверява, че увеличението на точността на мрежата над тренировъчния набор също увеличава и точността над валидационния набор (с който мрежата не е тренирана). Ако точността на мрежата над тренировъчния набор расте, но точността на мрежата над валидационния набор намалява или остава същата, значи мрежата се претренира и обучението трябва да спре.

**Тестови тип**

Ползват се само за тестване на вече трениран продукт за да се потвърди, че мрежата работи добре.

# Реализация

Както вече беше споменато, една невронна мрежа се състои от няколко слоя (в случая 3), като всеки слой има брой неврони в себе си. Невроните от един слой са свързани с всъчки неврони от следващия слой. Входа на мрежата се вкарва във входния слой, а резултата от мрежата се взима от изходния слой.

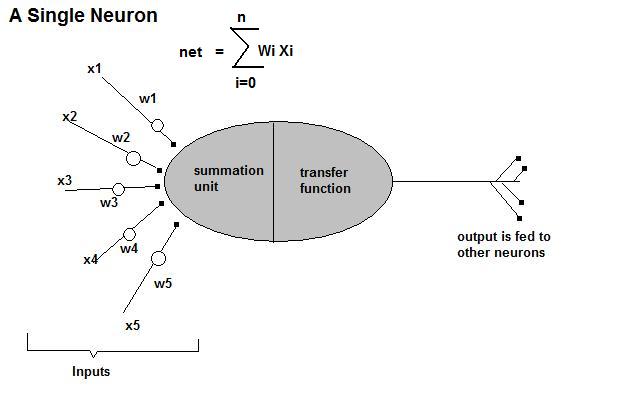


Една невронна мрежа може да бъде обучава с примери. Това вклучва подаване на вход и очавкан изход. На базата на грешката коефицентите се променят, така че да има по коректи резултати. Този процес се нарича трениране.

След като мрежата бъде обучена, за да се оперира с нея, е нужно да се подаде вход и мрежата да изчисли изхода. Изхода, разбира се, не винаги ще бъде коректен, но при правилно обучение, ще бъде сведен до минимум.

## Изкуствен неврон

Един неврон, който се ползва е една изкуствена невронна мрежа, много си прилича по структура на билогичен неврон.



Неврона се сътои от различен брой входове, суматор (summation unit), активационна функция (transfer function). Изхода на неврон от своя страна може да бъде предаден като вход на други неврони. Важно е да се отбележи и че за всеки входен сигнал има асоциирана съответна тежест.

### Суматор

За да се изчисли общия входен сигнал, суматора първо трябва да изчисли сбора на всеки входен сигнал, умножен със съответстващата му тежест. Също така обаче, всеки неврон има и прагов елемент, който променя общия входен сигнал. Праговия елемент поначно се инициализира със случайна стойност при създаване на невронната мрежа. Тежестите и праговия елемент се променят при тренирането на мрежата.

Където:

**yi**: сумиран входен сигнал

**Wij**: входна тежест

**Xj**: входен сигнал

### Активационна функция

Активационната функция представлява тривиална функция, която ползва общия входен сигнал, изчислен от суматора, за да генерира изходен сигнал. Изхода от активационната функция се предава като входен сигнал на невроните в следващия слой.

В тази имплементация е ползва на сигмоидалната функция. Тази функция приема стойността генерирана от суматора и произвежда стойност между 0 и 1.

Където:

**оut**: изход на неврон

**sum**: сумиран входен сигнал

## Структура на мрежата

Реализираната мрежа е разделена на 3 слоя – 1 входен, 1 скрит и 1 изходен. Невроните от входния слой са свързани с невроните от скрития слой, а невроните от скрития слой са свързани с невроните от изходния слой. Входния слой има 784 неврона, скрития слой има 387 неврона, а изходния слой има 10 неврона.

Броя на невроните във входния слой е равен на броя пиксели на всяко изображение. Входните неврони могат да имат стойност от 0.0 до 1.0. Тъй като стойността на всеки пиксел е между 0 и 255, преди стойността на пиксела да се подаде на входния неврон, тя е разделена на 255, за да може да се вмести в допустимите раници.

Броя на невроните в изходния слой е равен на броя възможни цифри (10). В зависимост от това коя е разпозната цифра, съответният неврон трябва да има стойност клоняща към 1.0, а останалите трябва да имат стойност клоняща към 0.0. Например, ако резултата е 0, това означава че първият неврон ще има стойност близка до 1.0.

Броян на невроните в скрития слой е избран произволно и може да бъде променян, но е следвано правилото:

брой неврони в скрития слой = (брой неврони във входния слоя – брой неврони в изходния слой)/2

## Обучение на мрежата

В тази секция се разглежда как се обучава реализираната невронна мрежа. В тази имплементация е ползван алгоритъма за обратно разпространение на грешката.

Важно е да се отбележи, че при първоначалното създаване на невронната мрежа, тежестите и праговите елементи се инциализират със случайни стойности.

Обучение на невронна мрежа представлява постъпковата промяна на тежестите и праговите елементи, за да може да се постигат по-акуратни резултати.

Един тренировъчен цикъл се състои от 6 стъпки.

### Стъпка 1: Инициализиране на входовете

За да се инициализа входа на мрежата е нужно просто статично да се инициализират изходите на входния слой.

### Стъпка 2: Намиране на изхода

По-горе е описано как се пресмята изхода от един неврон. Изхода на невроните от входния слой се ползва за вход на невроните в скрития слой. Аналогично, изхода на невроните от скрития слой се ползва за вход на невроните в изходния слой. Изхода на невроните от изходния слой е и изхода на мрежата.

Където:

**оut**: изход на неврон

**Wij**: входна тежест

**Xj**: входен сигнал

Както беше споменато, функцията updateOutput() първо се пресмята общата входна стойност и после минава през активационната функциия.

Функцията на суматора изглежда по следния начин:

Където:

**yi**: сумиран входен сигнал

**Wij**: входна тежест

**Xj**: входен сигнал

Активационната функция изглежда по следния начин:

Където:

**f(sum)**: изход на неврон

**sum**: сумиран входен сигнал

### Стъпка 3: Намиране на грешката (делтата)

В тази стъпка се пресмята грешката. Грешка (делта) може да се дефинира, като разликата между получения резултат и очаквания резултат. Например, когато за пръв път пресметнем изхода на мрежата, най-вероятно ще бъде тотално грешен.

За да се пресметне грешката:

1. Първо се пресмята грешката на всеки неврон в изходния слой
2. Пресметнатата делта се ползва за да се пресметне делтата на невроните в скрития слой.
3. Пресметнатата делта от скрития слой се ползва за да се пресметне делтата на невроните във входния слой.

За пресмятането на грешката на един неврон в изходния слой, ползваме следната формула:

Където:

: грешка на даден неврон

*:* изходен сигнал на даден неврон

: очакван резултат за даден неврон от изходния слой

**:** първа производна на активационната функция

За намирането на фактора на грешка в скрития слой обаче, сметките са малко по-различни:

1. Първо, делтата на всеки неврон, към който текущия неврон е свързанн, се умножава по тежестта на тази връзка
2. Тези произведения се събират за да се делтата за даден неврон в скрития слой.

Където:

: грешка на даден неврон

**:** първа производна на активационната функция

: делта на неврон от следващия слой.

*:* тежест на връзка до неврон от следващия слой

След края на стъпка 3, имаме делтата на всички неврони в мрежата.

### Стъпка 4: Промяна тежестите и праговите коефиценти

Както беше споменато, мрежата поддържа две стратегии на обучение – online и batch. При двете стратегии, тежестите се променят в различен етап.

### Online стратегия

При тази стратегия, тежестите се коригират след изчислението на грешката на всеки отделен пример от тренировъчния набор. Т.е.:

1. Пресмята се грешката при първия пример. Теглата се променят в съответствие с грешката.
2. Пресмята се грешката при втория пример. Теглата се променят в съответствие с грешката.
3. Пресмята се грешката при третия пример. Теглата се променят в съответствие с грешката.
4. И т.н.

След калкулирането на грешката на всички неврони във всички слоеве, трябва да бъдат коригирани тежестите и праговите коефиценти, на базата на същата тази грешка. Всеки неврон трябва да коригира повече от една тежест, тъй като има различна тежест за всяка връзка.

Функцията updateFreeParams() калкулира новите стойности на праговия елемент и тежестите, на базата на делтата която получихме в стъпка 3. Праговия елемент представлява тежест с коефициент на сигнала равен на 1.

За намиране на нов прагов елемент:

Където:

: грешка на даден неврон

*:* нова стойност на праговия елемент

: коефицент на обучение

*:* стара стойност на праговия елемент

За намиране на нова тежест на невроните:

Където:

: грешка на неврона

*:* нова стойност на тежестта

: стара стойност на тежестта

: коефицент на обучение

: изходна стойност на неврона

Online стратегията е доста по-бърза при обучение.

### Batch стратегия

При тази стратегия, тежестите се коригират след изчислението на грешката на всичките пример от тренировъчния набор. Т.е.:

1. Пресмята се грешката при първия пример. Теглата НЕ се променят, но промяната в теглата се пази в паметта.
2. Пресмята се грешката при втория пример. Теглата НЕ се променят, но промяната в теглата се събира с тази от предишната итерация и се пази в паметта.
3. Пресмята се грешката при третия пример. Теглата НЕ се променят, но промяната в теглата се събира с тази от предишните итерации и се пази в паметта.
4. И т.н.

За намиране на нов прагов елемент:

Където:

: сбор от промените в теглата след всички тренировъчни примери за текущата епоха.

: грешка на даден неврон за текущия тренировъчен пример

*:* нова стойност на праговия елемент

: коефицент на обучение

*:* стара стойност на праговия елемент

За намиране на нова тежест на невроните:

Където:

: сбор от промените в теглата след всички тренировъчни примери за текущата епоха.

: грешка на неврона за текущия тренировъчен пример

*:* нова стойност на тежестта

: стара стойност на тежестта

: коефицент на обучение

: изходна стойност на неврона за текущия тренировъчен пример

Специфичното за batch стратегията е, че теглата се променят само веднъж за всяка епоха.

### Момент

Мрежата дава възможността за ползване на т. нар. момент. Моментът дава възможност на системата за избягване на локален минимум и за по-бързо обучение. Представлява коефициент който се добавя към текущата промяна на тежестта, на база предишната.

При много висок коефициент има опасност от пропускане на минимума. При много нисък коефициент има опасност от много бавно трениране на мрежата.

Формулите за обучение с момент изглеждат по следния начин:

За намиране на нов прагов елемент:

Където:

: грешка на даден неврон

*:* нова стойност на праговия елемент

: коефициент на обучение

*:* стара стойност на праговия елемент

: коефициент на момент

**:** промяната на тежестта на праговия елемент от предишната итерация

За намиране на нова тежест на невроните:

Където:

: грешка на неврона

*:* нова стойност на тежестта

: стара стойност на тежестта

: коефицент на обучение

: изходна стойност на неврона

: коефициент на момент

**:** промяната на тежестта на връзката от предишната итерация

### Стъпка 5: Проверка за претрениране (overfitting)

След края на всеки обучителен цикъл, се прави проверка за претрениране. Тази стъпка се извършва с валидационния набор от изображения. Проверява се дали точността на мрежата над валидационния набор от изображения расте. Ако намалява или остава същата значи мрежата се претренира и обучението ще спре до тук.

За да се намери точността на мрежата се изчислява квадратичната грешка. За тази цел се изчислява резултата от мрежата за валидационния набор, но без да се променят тежестите. След всяко изчисление се пресмята квадратичната грешка по следната формула:

Където:

Квадратична грешка

*:* изходен сигнал на неврон от изходия слой

: очакван резултат за даден неврон от изходния слой

**1/n:** деление по броя на неврони в изходния слой

Квадратичната грешка се смята за всяко изображение от валидационния набор. Всичките грешки се събират и се делят на броя изображения. С това се намира коефициентът на точност. Ако текущия коефициент на точност е по-голям или равен на този от предишния обучителен цикъл, значи мрежата е претренирана и обучението трябва да спре.

### Стъпка 6: Повторение

След изпълнението на тези 5 стъпки, най-вероятно имаме по-добра мрежа. За получаване на реално резултати обаче този процес трябва да бъде повторен многократно.

## Опериране с мрежата

Оперирането на мрежата вклучва следните 2 стъпки:

1. Подаване на входове, както е описано в стъпка 1 от миналата глава.
2. Калкулирането на изходите, както е описано в стъпка 2 от миналата глава.

Важно е да се отбележи, че за да може да се използва една мрежа, тя трябва да бъде добре обучена с достатъчно примери. Може да се твърди, че колкото и да бъде тренирана една мрежа, винаги ще има някакъв процент грешка.

# Опериране с приложението

Както беше споменато, приложението е имплементирано изцяло на Java, което означава, че за да се оперира с приложението е нужно да има инсталирана Java 7.

## Файлове

Самия пакет е архив, представляващ набор от следните файлове и директории:

├───doc

├───IliyanDodekovF50129.docx

└───IliyanDodekovF50129.pdf

├───lib

└───mnist-nn.jar

├───networks

└───\*.nn

├───resources

├───test-images.idx3-ubyte

├───test-labels.idx1-ubyte

├───train-images.idx3-ubyte

├───train-labels.idx1-ubyte

├───val-images.idx3-ubyte

└───val-labels.idx1-ubyte

├───src

└───com

└───nbu

└───cscb822

├───api

└───\*.java

├───exception

└───\*.java

├───impl

└───\*.java

├───mnist

└───\*.java

├───util

└───\*.java

└───\*.java

├───nn.bat

└───nn.properties

В следващата таблица са описани по-важните файлове:

|  |  |
| --- | --- |
| Име | Описание |
| doc/ | Директория съдържаща документацията за приложението |
| lib/ | Директория съдържаща компилирания Java код |
| networks/ | Директория съдържаща готови невронни мрежи, записани във файл |
| resources/ | Директория съдържаща MNIST базата от ръкописни изображения в специални файлове. |
| src/ | Директория съдържаща сорс кода на приложението |
| nn.bat | Скрипт за стартиране на приложението под Windows |
| nn.properties | Конфигурационен файл на приложението. |

## Конфигурация

Цялата конфигурация за приложението се прави във файла nn.properties. Той съдържа следните възможни конфигурационни опции:

|  |  |
| --- | --- |
| Име | Описание |
| learning.rate | Коефицент на обучение |
| neurons.input.layer | Брой неврони във входния слой. |
| neurons.hidden.layer | Брой неврони в скрития слой. |
| neurons.output.layer | Брой неврони в изходния слой. |
| file.training.label | Файл съдържащ 60000 имена на цифри за обучителните изображения. |
| file.training.image | Файл съдържащ пикселите на 60000 цифри за обучителните изображения. |
| file.testing.label | Файл съдържащ 5000 имена на цифри за тестовите изображения. |
| file.testing.image | Файл съдържащ пикселите на 5000 цифри за тестовите изображения. |
| file.validation.label | Файл съдържащ 5000 имена на цифри за валидационни изображения. |
| file.validation.image | Файл съдържащ пикселите на 5000 цифри за валидационни изображения. |
| image.pixels.x | Брой хоризонтални пиксели на изображенията. |
| image.pixels.y | Брой вертикални пиксели на изображенията. |
| overfitting.check.enabled | Дали мрежата да проверява за претрениране с валидационния набор от изображения. Възможните стойности са true или false. |
| random.seed | Seed за случайни числа за генерирането на тежестите на нова мрежа. |
| training.mode | Определя обучителната стратегия – online или batch. Възможните стойности са online и batch. |
| momentum.enable | Тази настроийна разрешава или забранява ползването на момент в изчислението на нови стойности на тежестите. Възможните стойности са true и false. |
| momentum.rate | Коефициент на момент. |

## Опериране

Реализираното приложение е изцяло конзолно, като всичките му възможности могат да бъдат достъпени с помощта на меню. За да бъде стартирано е нужно да се стартира файла **nn.bat** под Windows.

При стартиране на приложението се зареждат файловете с изображенията в паметта за по-бързо достъпване и на потребителя е представено меню:

E:\workspace\CSCB822NN>nn.bat

Please wait. Loading files...

Magic number for file resources\val-labels.idx1-ubyte is 2049

Entry size for file resources\val-labels.idx1-ubyte is 5000

Magic number for file resources\test-labels.idx1-ubyte is 2049

Entry size for file resources\test-labels.idx1-ubyte is 5000

Magic number for file resources\test-images.idx3-ubyte is 2051

Entry size for file resources\test-images.idx3-ubyte is 3920000

Magic number for file resources\val-images.idx3-ubyte is 2051

Entry size for file resources\val-images.idx3-ubyte is 3920000

Magic number for file resources\train-labels.idx1-ubyte is 2049

Entry size for file resources\train-labels.idx1-ubyte is 60000

Magic number for file resources\train-images.idx3-ubyte is 2051

Entry size for file resources\train-images.idx3-ubyte is 47040000

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise:

След избор на дадена опция и изпълнението и, потребителят винаги бива доведен до това меню до изход от приложението.

Всяка от опциите има следното действие:

* **Create a new neural network** – Създава нова невронна мрежа със случайни стойности на тежестите и праговите коефиценти. Приложението може да работи само с една невронна мрежа и при повторно извикване на тази опция, старата мрежа ще бъде изтрита.
* **Load a neural network from a file** – Зарежда вече създадена и потенциално обучена невронна мрежа от файл. Формата на файла е специфичен за приложението и могат да бъдат зареждани само файлове, които са създадени от същото това приложение. Приложението може да работи само с една невронна мрежа и при повторно извикване на тази опция, старата мрежа ще бъде изтрита.
* **Save a neural network to a file** – Запазва обучена мрежа във файл. Формата на файла е специфичен за приложението и могат да бъдат зареждани само файлове, които са създадени от същото това приложение.
* **Train the neural network** – Започва обучение на невронна мрежа. Задава се колко обучителни цикъла да се направят. Възможно е и предварително прекратяване на тренирането при достигане на даден процент на грешка – тази опция може да се конфигурира в nn.properties файла.
* **Test the neural network** – Демонстрира се работата на невронната мрежа – подава се цифра и мрежата познава коя е тя. Цифрата се подава, като пореден номер от тестовия сет от MNIST базата. За добри резултати е нужно мрежата да бъде обучена достатъчно.
* **Print average error rate** – Калкулира се процента на грешка, спрямо всички изображения от тестовия сет от MNIST базата. Реално всичките 10000 изображения се прекарват през мрежата и излиза процента на сбъркани категоризации.
* **Print average mean square history for training data** – визуализира квадратичната грешка изчислена след всеки цикъл на обучение на мрежата.
* **Print average mean square history for validation data** - визуализира квадратичната грешка на валидационния набор от изображения изчислена след всеки цикъл на обучение на мрежата.
* **Quit** – Изход от приложението.

### Създаване на мрежа

За да се създаде мрежа:

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 1

Neural Network successfully initialized!

### Зареждане на вече създадена мрежа от файл

За да се зареди вече създадена мрежа:

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 2

Enter file to load the network from: E:\workspace\CSCB822NN\networks\nn5-1\_94.nn

Neural Network successfully loaded!

### Записване на мрежа във файл

За да се запиша невронна мрежа във файл:

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 3

Enter file to save the network to: E:\workspace\CSCB822NN\networks\test.nn

### Обучение на мрежа

За да бъде обучена мрежа:

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 4

Finished adding training data.

Finished adding validation data.

Enter the number of epochs: 5

Training is in process. Please wait.

Training mode is set to: online

Training data MSE for round [1] is [0.03328079540739458].

Validation data MSE for round [1] is [0.0336537310321584].

Epoch [1] is done in 18.73735 minutes.

Training data MSE for round [2] is [0.02339336776163224].

Validation data MSE for round [2] is [0.01630062776223768].

Epoch [2] is done in 18.625983333333334 minutes.

Training data MSE for round [3] is [0.014923125409202685].

Validation data MSE for round [3] is [0.015142307748459996].

Epoch [3] is done in 18.652933333333333 minutes.

Training data MSE for round [4] is [0.014149090772041207].

Validation data MSE for round [4] is [0.01493014154354006].

Epoch [4] is done in 18.62825 minutes.

Training data MSE for round [5] is [0.013676553937716985].

Validation data MSE for round [5] is [0.014629638115677301].

Epoch [5] is done in 18.43585 minutes.

Training done in 93.10598333333333 minutes.

В зависимост от броя обучителни итерации, времето за обучение може да бъде доста голямо. Ако е конфигурирано, тренировката може да спре преждевременно при претрениране.

Претрениране на мрежата изглежда по следния начин:

MNIST Neural Network for handwritten digit recognition. Choose the desired optio

n:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 4

Finished adding training data.

Finished adding validation data.

Enter the number of epochs: 5

Training is in process. Please wait.

Training mode is set to: shuffle

Training data MSE for round [1] is [0.0134509350214862].

Validation data MSE for round [1] is [0.014896809038974735].

Average MSE of validation data is bigger than in previous iteration. The network

is overfitting - training will stop now.

Training done in 18.647316666666665 minutes.

### Тестване на мрежа

Тестването на мрежата става посредством избор на поредна цифра от тестовия сет на MNIST базата. Мрежата се опитва да познае е коя е цифрата и съответно се принтира дали наистина е това. На конзолата се изрисува и как изглежда самото изображение, като всички пиксели, които не са 0 се изобразяват със знака „\*“. Принтира се също и вектор с изхода на мрежата (стойността на 10-те неврона в изходния слой).

MNIST Neural Network for handwritten digit recognition. Choose the desired optio

n:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 5

Enter the desired letter number from the testing set or [b] to return: 371

+++

++++++++

++++++++++

+++++++++++

++++++

++++

+++

++++

+++++

+++++

+++++

+++++

+++++

+++++

++++ ++++++++

+++++++++++++++++

+++++++++++++++++

+++++++++++++

++++++++++

++++

Testing with number: 2

This looks like the number: 2

Neural Network output - [0:0.0000 ,1:0.0000 ,2:1.0000 ,3:0.0000 ,4:0.0000 ,5:0.0

000 ,6:0.0000 ,7:0.0000 ,8:0.0000 ,9:0.0000]

Enter the desired letter number from the testing set or [b] to return: 982

++++++++

+++++++++++

+++++++++++

+++++++++++

++++ +++++

+++++

+++++

++++++

+++++++

+++++++

++++++

+++++

++++++

+++++

+++++++

+++++++

++++++

+++++++++++

+++++++++++

++++++++

Testing with number: 3

This looks like the number: 3

Neural Network output - [0:0.0000 ,1:0.0000 ,2:0.0000 ,3:1.0000 ,4:0.0000 ,5:0.0

000 ,6:0.0000 ,7:0.0000 ,8:0.0000 ,9:0.0000]

Enter the desired letter number from the testing set or [b] to return: 2345

++++++

++++++++++

+++++ ++++++

+++ +++

+++ ++++

+++ ++++

++++ +++++

++++ +++

++++++++++

++++++++

+++

+++

+++

+++

+++

+++

+++

+++

+++

+++

Testing with number: 9

This looks like the number: 9

Neural Network output - [0:0.0000 ,1:0.0000 ,2:0.0000 ,3:0.0000 ,4:0.0000 ,5:0.0

000 ,6:0.0000 ,7:0.0000 ,8:0.0000 ,9:1.0000]

Enter the desired letter number from the testing set or [b] to return:

За да се върне в главното меню потребителя просто трябва да натисне “b”.

### Пресмятане на процента на грешка

При пресмятане на грешката се тестват целите тестов, валидационни и тренировъчни набори на MNSIT базата. Процента на грешка представлява процента на сбъркани цифри спрямо целия брой:

MNIST Neural Network for handwritten digit recognition. Choose the desired option:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 6

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [25507] from [60000]. That's [42.51166666666666%].

Error count for number [0] is 5870

Error count for number [1] is 145

Error count for number [2] is 416

Error count for number [3] is 6127

Error count for number [4] is 652

Error count for number [5] is 887

Error count for number [6] is 5354

Error count for number [7] is 5534

Error count for number [8] is 274

Error count for number [9] is 248

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [2085] from [5000]. That's [41.699999999999996%].

Error count for number [0] is 510

Error count for number [1] is 10

Error count for number [2] is 26

Error count for number [3] is 510

Error count for number [4] is 38

Error count for number [5] is 68

Error count for number [6] is 446

Error count for number [7] is 453

Error count for number [8] is 7

Error count for number [9] is 17

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [2091] from [5000]. That's [41.82%].

Error count for number [0] is 451

Error count for number [1] is 7

Error count for number [2] is 50

Error count for number [3] is 500

Error count for number [4] is 71

Error count for number [5] is 86

Error count for number [6] is 410

Error count for number [7] is 452

Error count for number [8] is 33

Error count for number [9] is 31

-----

### Квадратична грешка на тренировъчните данни

Визуализира квадратичната грешка изчислена след всеки цикъл на обучение на мрежата.

MNIST Neural Network for handwritten digit recognition. Choose the desired optio

n:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 7

Training epoch [1] has finished with MSE value of [0.07434192700569693].

Training epoch [2] has finished with MSE value of [0.05444115540430707].

Training epoch [3] has finished with MSE value of [0.044637232831067346].

Training epoch [4] has finished with MSE value of [0.03406269781808672].

Training epoch [5] has finished with MSE value of [0.03328079540739458].

Training epoch [6] has finished with MSE value of [0.02339336776163224].

Training epoch [7] has finished with MSE value of [0.014923125409202685].

Training epoch [8] has finished with MSE value of [0.014149090772041207].

Training epoch [9] has finished with MSE value of [0.013676553937716985].

Training epoch [10] has finished with MSE value of [0.0134509350214862].

### Квадратична грешка на валидационните данни

Визуализира квадратичната грешка на валидационния набор от изображения изчислена след всеки цикъл на обучение на мрежата.

MNIST Neural Network for handwritten digit recognition. Choose the desired optio

n:

1. Create a new neural network.

2. Load a neural network from a file.

3. Save neural network to a file.

4. Train the neural network.

5. Test the neural network.

6. Print average error rate.

7. Print average mean square error history for training data.

8. Print average mean square error history for validation data.

99. Quit.

Your choise: 8

Training epoch [1] has finished with MSE value of [0.055963184571055105].

Training epoch [2] has finished with MSE value of [0.05323948756883251].

Training epoch [3] has finished with MSE value of [0.034858559903565].

Training epoch [4] has finished with MSE value of [0.03420022121822513].

Training epoch [5] has finished with MSE value of [0.0336537310321584].

Training epoch [6] has finished with MSE value of [0.01630062776223768].

Training epoch [7] has finished with MSE value of [0.015142307748459996].

Training epoch [8] has finished with MSE value of [0.01493014154354006].

Training epoch [9] has finished with MSE value of [0.014629638115677301].

Training epoch [10] has finished with MSE value of [0.014896809038974735].

Тук може да се види, че при последната обучителна итерация квадратичната грешка е по-голяма от при пред-последната обучителна итерация. Съответно обучението е спряло преждевременно.

# Тестове

Тази секция описва впечатленията ми от работата с мрежата. Изложени са различни тестове с различни параметри на мрежата, както и резултатите.

## Online трениране с момент и проверка за претрениране

Мрежата се инициализира със следните параметри:

training.mode=online

momentum.enable=true

momentum.rate=0.5

overfitting.check.enabled=true

learning.rate=0.5

random.seed=

Training Data Images: 60000

Testing Data Images: 5000

Validation Data Images: 5000

За 6 епохи успява да се обучи до **28%** грешка на тестовия набор, преди да започне да се претренира.

При първоначалното генериране на тежестите на мрежата тя има следния процент на грешка за тестовите, валидационните и тренировъчните данни (над 90% и на трите):

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [55303] from [60000]. That's [92.17166666666667%].

Error count for number [0] is 5908

Error count for number [1] is 5910

Error count for number [2] is 5649

Error count for number [3] is 6085

Error count for number [4] is 5841

Error count for number [5] is 2945

Error count for number [6] is 5154

Error count for number [7] is 6265

Error count for number [8] is 5599

Error count for number [9] is 5947

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [4597] from [5000]. That's [91.94%].

Error count for number [0] is 519

Error count for number [1] is 477

Error count for number [2] is 481

Error count for number [3] is 509

Error count for number [4] is 482

Error count for number [5] is 243

Error count for number [6] is 417

Error count for number [7] is 516

Error count for number [8] is 464

Error count for number [9] is 489

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [4595] from [5000]. That's [91.9%].

Error count for number [0] is 457

Error count for number [1] is 496

Error count for number [2] is 504

Error count for number [3] is 495

Error count for number [4] is 500

Error count for number [5] is 239

Error count for number [6] is 393

Error count for number [7] is 512

Error count for number [8] is 479

Error count for number [9] is 520

-----

### 1 епоха

След една епоха се наблюдава рязко подобрение – грешката (mean square error) е **0.07668308144304185** на тренировъчния набор и **0.06644655196707798** на валидационния.

Training is in process. Please wait.

Training mode is set to: online

Training data MSE for round [1] is [0.07668308144304185].

Validation data MSE for round [1] is [0.06644655196707798].

Epoch [1] is done in 44.00143333333333 minutes.

Training done in 44.02778333333333 minutes.

Има следния процент на грешка:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [34271] from [60000]. That's [57.11833333333334%].

Error count for number [0] is 5539

Error count for number [1] is 88

Error count for number [2] is 1102

Error count for number [3] is 6115

Error count for number [4] is 333

Error count for number [5] is 5318

Error count for number [6] is 5729

Error count for number [7] is 4565

Error count for number [8] is 1249

Error count for number [9] is 4233

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [2737] from [5000]. That's [54.74%].

Error count for number [0] is 465

Error count for number [1] is 1

Error count for number [2] is 73

Error count for number [3] is 510

Error count for number [4] is 23

Error count for number [5] is 425

Error count for number [6] is 479

Error count for number [7] is 354

Error count for number [8] is 64

Error count for number [9] is 343

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [2784] from [5000]. That's [55.67999999999999%].

Error count for number [0] is 434

Error count for number [1] is 5

Error count for number [2] is 121

Error count for number [3] is 500

Error count for number [4] is 36

Error count for number [5] is 442

Error count for number [6] is 446

Error count for number [7] is 351

Error count for number [8] is 129

Error count for number [9] is 320

-----

### 2 епохи

След две епохи отново се наблюдава рязко подобрение – грешката (mean square error) е **0.05387129600441176** на тренировъчния набор и **0.04841927969152609** на валидационния.

Training is in process. Please wait.

Training mode is set to: online

Training data MSE for round [1] is [0.05387129600441176].

Validation data MSE for round [1] is [0.04841927969152609].

Epoch [1] is done in 46.11386666666667 minutes.

Training done in 46.23345 minutes.

Има следния процент на грешка:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [25507] from [60000]. That's [42.51166666666666%].

Error count for number [0] is 5870

Error count for number [1] is 145

Error count for number [2] is 416

Error count for number [3] is 6127

Error count for number [4] is 652

Error count for number [5] is 887

Error count for number [6] is 5354

Error count for number [7] is 5534

Error count for number [8] is 274

Error count for number [9] is 248

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [2085] from [5000]. That's [41.699999999999996%].

Error count for number [0] is 510

Error count for number [1] is 10

Error count for number [2] is 26

Error count for number [3] is 510

Error count for number [4] is 38

Error count for number [5] is 68

Error count for number [6] is 446

Error count for number [7] is 453

Error count for number [8] is 7

Error count for number [9] is 17

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [2091] from [5000]. That's [41.82%].

Error count for number [0] is 451

Error count for number [1] is 7

Error count for number [2] is 50

Error count for number [3] is 500

Error count for number [4] is 71

Error count for number [5] is 86

Error count for number [6] is 410

Error count for number [7] is 452

Error count for number [8] is 33

Error count for number [9] is 31

-----

### 6 епохи

След още 4 епохи (общо 6), мрежата започва да се претренира и обучението спира.

Training is in process. Please wait.

Training mode is set to: online

Training data MSE for round [1] is [0.05021654235036695].

Validation data MSE for round [1] is [0.04808809736302044].

Epoch [1] is done in 37.91566666666667 minutes.

Training data MSE for round [2] is [0.041602297478713864].

Validation data MSE for round [2] is [0.03876231091359376].

Epoch [2] is done in 37.691266666666664 minutes.

Training data MSE for round [3] is [0.04043449517266307].

Validation data MSE for round [3] is [0.03796245944783562].

Epoch [3] is done in 37.81531666666667 minutes.

Training data MSE for round [4] is [0.03935100239311882].

Validation data MSE for round [4] is [0.036877558917042474].

Epoch [4] is done in 37.82018333333333 minutes.

Training data MSE for round [5] is [0.038960858293630635].

Validation data MSE for round [5] is [0.0370746927494557].

Average MSE of validation data is bigger than in previous iteration. The network

is overfitting - training will stop now.

Training done in 189.07436666666666 minutes.

Коефициентът на грешка отново е намалял драстично на 28% и това е крайното състояние на мрежата. 60% подобрение в точността за 6 епохи.

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [16941] from [60000]. That's [28.235%].

Error count for number [0] is 3492

Error count for number [1] is 113

Error count for number [2] is 478

Error count for number [3] is 255

Error count for number [4] is 209

Error count for number [5] is 382

Error count for number [6] is 5353

Error count for number [7] is 5721

Error count for number [8] is 609

Error count for number [9] is 329

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [1365] from [5000]. That's [27.3%].

Error count for number [0] is 298

Error count for number [1] is 7

Error count for number [2] is 30

Error count for number [3] is 16

Error count for number [4] is 14

Error count for number [5] is 37

Error count for number [6] is 442

Error count for number [7] is 471

Error count for number [8] is 30

Error count for number [9] is 20

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [1407] from [5000]. That's [28.139999999999997%].

Error count for number [0] is 282

Error count for number [1] is 6

Error count for number [2] is 57

Error count for number [3] is 7

Error count for number [4] is 27

Error count for number [5] is 52

Error count for number [6] is 382

Error count for number [7] is 481

Error count for number [8] is 72

Error count for number [9] is 41

-----

## Online трениране с проверка за претрениране и БЕЗ момент

Мрежата се инициализира със следните параметри:

training.mode=online

momentum.enable=**false**

momentum.rate=0.5

overfitting.check.enabled=true

learning.rate=0.5

random.seed=

Training Data Images: 60000

Testing Data Images: 5000

Validation Data Images: 5000

Този тест е правен докато мрежата все още нямаше възможност за трениране с момент. За 10 епохи успява да се обучи до **9,68%** грешка на тестовия набор, преди да започне да се претренира:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [4630] from [60000]. That's [7.716666666666666%].

Error count for number [0] is 3178

Error count for number [1] is 101

Error count for number [2] is 79

Error count for number [3] is 298

Error count for number [4] is 171

Error count for number [5] is 125

Error count for number [6] is 75

Error count for number [7] is 217

Error count for number [8] is 234

Error count for number [9] is 152

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [389] from [5000]. That's [7.779999999999999%].

Error count for number [0] is 284

Error count for number [1] is 8

Error count for number [2] is 7

Error count for number [3] is 23

Error count for number [4] is 16

Error count for number [5] is 11

Error count for number [6] is 2

Error count for number [7] is 13

Error count for number [8] is 14

Error count for number [9] is 11

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [484] from [5000]. That's [9.68%].

Error count for number [0] is 257

Error count for number [1] is 9

Error count for number [2] is 15

Error count for number [3] is 26

Error count for number [4] is 33

Error count for number [5] is 19

Error count for number [6] is 20

Error count for number [7] is 45

Error count for number [8] is 34

Error count for number [9] is 26

-----

## Online трениране БЕЗ проверка за претрениране и БЕЗ момент

Мрежата се инициализира със следните параметри:

training.mode=online

momentum.enable=**false**

momentum.rate=0.5

overfitting.check.enabled=**false**

learning.rate=0.5

random.seed=

Training Data Images: 60000

Testing Data Images: 5000

Validation Data Images: 5000

Този тест е правен докато мрежата все още нямаше възможност за трениране с момент, нито за проверка от претрениране. Това е и най-добрия резултат който мрежата е изкарвала. След 15 дневно трениране на мрежата (между 300-500 епохи), процентът на грешка над тестовия набор е **2.6%**, **1.28%** на валидационния и **0.46%** на тренировъчния! Важно е да се каже че за този тест, обучителния коефициент постепенно беше намален до 0.01, от 0.5:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [277] from [60000]. That's [0.46166666666666667%]

Error count for number [0] is 9

Error count for number [1] is 15

Error count for number [2] is 33

Error count for number [3] is 49

Error count for number [4] is 18

Error count for number [5] is 39

Error count for number [6] is 14

Error count for number [7] is 23

Error count for number [8] is 37

Error count for number [9] is 40

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [64] from [5000]. That's [1.28%].

Error count for number [0] is 4

Error count for number [1] is 6

Error count for number [2] is 8

Error count for number [3] is 5

Error count for number [4] is 7

Error count for number [5] is 9

Error count for number [6] is 2

Error count for number [7] is 7

Error count for number [8] is 7

Error count for number [9] is 9

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [130] from [5000]. That's [2.6%].

Error count for number [0] is 6

Error count for number [1] is 6

Error count for number [2] is 9

Error count for number [3] is 11

Error count for number [4] is 13

Error count for number [5] is 10

Error count for number [6] is 15

Error count for number [7] is 17

Error count for number [8] is 21

Error count for number [9] is 22

-----

## Batch трениране с момент и проверка за претрениране

Мрежата се инициализира със следните параметри:

training.mode=batch

momentum.enable=true

momentum.rate=0.5

overfitting.check.enabled=true

learning.rate=0.5

random.seed=

Training Data Images: 5000

Testing Data Images: 5000

Validation Data Images: 5000

Този тест показва, че мрежата не се справя добре при batch трениране. Това са най-лошите резултати. Тренировъчния набор е намален на 5000, защото при тест с 60000, тежестите се променят почти незабележимо. За целта валидационните и тестовите данни са еднакви за примера. След 9 обучителни цикъла, процента грешка на тренировъчния набор намалява с 2% от 93% на 91,28%, а на тестовия набор се увеличава от 90,36% на 90,88%. След 9-тата итерация мрежата започва да се претренира.

Наблюдава се изключително бавно обучение.

Грешка при първоначално генериране на тежестите:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [4665] from [5000]. That's [93.30000000000001%].

Error count for number [0] is 514

Error count for number [1] is 564

Error count for number [2] is 494

Error count for number [3] is 293

Error count for number [4] is 482

Error count for number [5] is 436

Error count for number [6] is 411

Error count for number [7] is 516

Error count for number [8] is 485

Error count for number [9] is 470

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [4665] from [5000]. That's [93.30000000000001%].

Error count for number [0] is 514

Error count for number [1] is 564

Error count for number [2] is 494

Error count for number [3] is 293

Error count for number [4] is 482

Error count for number [5] is 436

Error count for number [6] is 411

Error count for number [7] is 516

Error count for number [8] is 485

Error count for number [9] is 470

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [4518] from [5000]. That's [90.36%].

Error count for number [0] is 456

Error count for number [1] is 571

Error count for number [2] is 520

Error count for number [3] is 205

Error count for number [4] is 500

Error count for number [5] is 456

Error count for number [6] is 357

Error count for number [7] is 511

Error count for number [8] is 489

Error count for number [9] is 453

-----

### 4 епохи

След 4 епохи се наблюдава минимално подобрение в грешката:

Training is in process. Please wait.

Training mode is set to: batch

Training data MSE for round [1] is [0.09999995555319206].

Epoch [1] is done in 3.450616666666667 minutes.

Training data MSE for round [2] is [0.09999994456468737].

Epoch [2] is done in 3.8093833333333333 minutes.

Training data MSE for round [3] is [0.09999992869957912].

Epoch [3] is done in 3.792666666666667 minutes.

Training data MSE for round [4] is [0.09999990144250792].

Epoch [4] is done in 3.818783333333333 minutes.

Training done in 14.934 minutes.

### 8 епохи

След още 4 епохи подобрението остава минимално, докато при последната епоха рязко се покачва грешката и тренировката спира заради претрениране:

Training is in process. Please wait.

Training mode is set to: batch

Training data MSE for round [1] is [0.09999984583143728].

Epoch [1] is done in 3.1206833333333335 minutes.

Training data MSE for round [2] is [0.09999968966873754].

Epoch [2] is done in 3.09655 minutes.

Training data MSE for round [3] is [0.09999873293247649].

Epoch [3] is done in 3.08085 minutes.

Training data MSE for round [4] is [0.09973598983054967].

Epoch [4] is done in 3.0790166666666665 minutes.

Training data MSE for round [5] is [0.1825600000000141].

Average MSE of training data is bigger than in previous iteration. The network i

s overfitting - training will stop now.

Training done in 15.5189 minutes.

Крайния процент на грешка е:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [4564] from [5000]. That's [91.28%].

Error count for number [0] is 520

Error count for number [1] is 564

Error count for number [2] is 502

Error count for number [3] is 510

Error count for number [4] is 482

Error count for number [5] is 0

Error count for number [6] is 496

Error count for number [7] is 516

Error count for number [8] is 485

Error count for number [9] is 489

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [4564] from [5000]. That's [91.28%].

Error count for number [0] is 520

Error count for number [1] is 564

Error count for number [2] is 502

Error count for number [3] is 510

Error count for number [4] is 482

Error count for number [5] is 0

Error count for number [6] is 496

Error count for number [7] is 516

Error count for number [8] is 485

Error count for number [9] is 489

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [4544] from [5000]. That's [90.88000000000001%].

Error count for number [0] is 460

Error count for number [1] is 571

Error count for number [2] is 530

Error count for number [3] is 500

Error count for number [4] is 500

Error count for number [5] is 0

Error count for number [6] is 462

Error count for number [7] is 512

Error count for number [8] is 489

Error count for number [9] is 520

-----

## Batch трениране с момент и БЕЗ проверка за претрениране

Мрежата се инициализира със следните параметри:

training.mode=batch

momentum.enable=true

momentum.rate=0.5

overfitting.check.enabled=false

learning.rate=0.5

random.seed=

Training Data Images: 5000

Testing Data Images: 5000

Validation Data Images: 5000

Въпреки, че за 20 епохи при този тест резултатите са малко по-добри от миналия, пак обучение е много бавно и процента грешка не може да се смъкне под 90%.

Начален процент грешка:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [4636] from [5000]. That's [92.72%].

Error count for number [0] is 510

Error count for number [1] is 556

Error count for number [2] is 362

Error count for number [3] is 510

Error count for number [4] is 477

Error count for number [5] is 436

Error count for number [6] is 296

Error count for number [7] is 515

Error count for number [8] is 485

Error count for number [9] is 489

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [4636] from [5000]. That's [92.72%].

Error count for number [0] is 510

Error count for number [1] is 556

Error count for number [2] is 362

Error count for number [3] is 510

Error count for number [4] is 477

Error count for number [5] is 436

Error count for number [6] is 296

Error count for number [7] is 515

Error count for number [8] is 485

Error count for number [9] is 489

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [4678] from [5000]. That's [93.56%].

Error count for number [0] is 449

Error count for number [1] is 569

Error count for number [2] is 394

Error count for number [3] is 500

Error count for number [4] is 482

Error count for number [5] is 455

Error count for number [6] is 318

Error count for number [7] is 512

Error count for number [8] is 489

Error count for number [9] is 510

-----

Краен процент грешка след 20 епохи:

-----

Calculating average error rate on TRAINING data...

Finished testing the data.

Error count is [4515] from [5000]. That's [90.3%].

Error count for number [0] is 520

Error count for number [1] is 564

Error count for number [2] is 502

Error count for number [3] is 510

Error count for number [4] is 482

Error count for number [5] is 436

Error count for number [6] is 496

Error count for number [7] is 516

Error count for number [8] is 0

Error count for number [9] is 489

-----

-----

Calculating average error rate on VALIDATION data...

Finished testing the data.

Error count is [4515] from [5000]. That's [90.3%].

Error count for number [0] is 520

Error count for number [1] is 564

Error count for number [2] is 502

Error count for number [3] is 510

Error count for number [4] is 482

Error count for number [5] is 436

Error count for number [6] is 496

Error count for number [7] is 516

Error count for number [8] is 0

Error count for number [9] is 489

-----

-----

Calculating average error rate on TESTING data...

Finished testing the data.

Error count is [4511] from [5000]. That's [90.22%].

Error count for number [0] is 460

Error count for number [1] is 571

Error count for number [2] is 530

Error count for number [3] is 500

Error count for number [4] is 500

Error count for number [5] is 456

Error count for number [6] is 462

Error count for number [7] is 512

Error count for number [8] is 0

Error count for number [9] is 520

-----

# Изпозлвани източници

За изготвянето на курсовата работа са ползвани следните източници:

Курсовете по CSCB822 Невронни Мрежи

<http://wikipedia.org>

<http://yann.lecun.com/exdb/mnist/>

<http://www.codeproject.com/>

<http://neuralnetworksanddeeplearning.com/>

<http://pages.cs.wisc.edu/>

<http://statweb.stanford.edu/~tibs/ElemStatLearn/data.html>