

Raport końcowy

Sieć CNN
do klasyfikacji
obrazów znaków

Piotr Heinzelman
Wydział Elektryczny

1. Opis struktury i głównych funkcji

Teacher - klasa nauczyciela

Do pracy potrzebuje źródła danych (klasy `FileReader`), która dostarcza dwuwymiarowy wektor - czyli tablicę wartości pikseli obrazu. Przy czym pierwsza wartość (`X[0][0]`) jest numerem klasy. Klasa `FileReader` po uruchomieniu konstruktora wczytuje dane, a następnie buduje wewnętrzną tablicę obrazów do wykorzystania. W momencie gdy `Teacher` otrzyma obraz, jego indeks jest usuwany z dostępnych obrazów, dzięki temu nauczone wzory nie powtarzają się w epoce.

`Teacher` wczytuje wektor uczący wołając metodę: `fileReader.getNextTrainX()`;

W aplikacji, (a także podczas testów) wykorzystywane są statyczne metody narzędziowe takich klas jak:

- **Tools** - dostarczających m.in. metod wyświetlających zawartości wektorów i tablic w konsoli;
- **Mat** - dostarczających metod związanych z arytmetyką macieżową, czyli m.in. mnożeniem przez skalar (skalowaniem) mnożeniem przez macierzy, dodawaniem, obliczaniem wyznaczników i innymi;

Klasa `Teacher` tworzy i konfiguruje potrzebne do pracy warstwy sieci po wywołaniu metody `prepare(filters)`; jako parametr przekazujemy liczbę filtrów w warstwie spłotowej. Metoda ta powoduje m.in. wylosowanie nowych wag dla wszystkich warstw sieci (wych które te wagi mają). Wagi te będą modyfikowane w procesie uczenia, a będą wykorzystywane w procesie wyliczania wartości wektora wyjściowego. Wyliczony wektor jest wykorzystywany w procesie uczenia, oraz w procesie pracy aplikacji polegającej na rozpoznawaniu znaków.

W aplikacji używana jest jedna warstwa spłotowa zawierająca 8 filtrów o wymiarach 3x3. Warstwa spłotowa po wysłaniu jako parametru wektora wejściowego `X` o wymiarach 28x28 pikseli czyli obrazu wylicza w procesie spłatania 8 "obrazów", czy tablic wartości wynikowych.

tablice te podawane są kolejnej warstwie `Pool` - która wylicza maksymalne wartości pól 2x2 pikseli i zwraca jako "piksel" wyjściowy zmniejszając wymiar danych z 8x26x26 pixeli do 8x13x13 pikseli. (zakładając że podaliśmy jako liczbę filtrów liczbę 8).

Kolejna warstwa (Softmax)

- dokonuje spłaszczenia bloku danych do postaci jednowymiarowego wektora
- blok wejściowy o wymiarach 8 x 13 x 13 liczb wymnaża przez swoją wewnętrzną warstwę wag `W` o wymiarach ([8x13x13] na [10]) oraz i dodaje do wyniku wartości Bias. Otrzymujemy do kolejnej operacji wektor `A` o wymiarach [10].
- przeliczenia wektora `A` na wektor rozkładu prawdopodobieństwa `P`

Po przeliczeniu wektora `A` wg. wzoru na `P` rozkład prawdopodobieństwa otrzymujemy wektor wyjściowy, który opisuje prawdopodobieństwo przynależności obrazu do kolejnych klas.

Proces uczenia

Jeśli “nauczyciel” wie do jakiej klasy faktycznie należał obraz, to może policzyć “Błąd” tej operacji. A jeśli tak, to może te informacje w postaci wektora różnicy wartości oczekiwanych i wartości otrzymanych i przekazać wyjściowej. W warstwie wyjściowej zajdzie proces korygowania Wagi, a także wyliczenia “błędu” jaki wnośła warstwa bezpośrednio poprzedzająca. Warstwa poprzedzająca także otrzyma informacje liczbową w postaci wektora o swoim udziale. Na tej podstawie skoryguje swoje wagi, o ile je posiada. Warstwą korygującą wagi będzie warstwa splotowa, która skoryguje - albo inaczej dostroi swoje filtry tak - by w przyszłości dokładniej reagowały na pokazany obraz. Taki proces - wstecznego przekazywania wielkości błędu to proces nauczania.

Po każdym cyklu (epoce) nauczania wypisywany na konsoli jest podsumowujący komunikat:

```
filters: 8, average accuracy:- 0.823%
filters: 8, average accuracy:- 0.881%
filters: 8, average accuracy:- 0.8884%
filters: 8, average accuracy:- 0.8958%
filters: 8, average accuracy:- 0.9052%
filters: 8, average accuracy:- 0.9178%
```

zestawienie wyników dla różnych ilości masek przy ekspozycji 1000 obrazów dla 5 epok.
5 x 1000 images in ep.

```
filters: 4, average accuracy:- 0.8156%
filters: 4, average accuracy:- 0.8796%
filters: 4, average accuracy:- 0.8884%
filters: 4, average accuracy:- 0.905%
filters: 4, average accuracy:- 0.8988%
```

```
filters: 5, average accuracy:- 0.8222%
filters: 5, average accuracy:- 0.8774%
filters: 5, average accuracy:- 0.9036%
filters: 5, average accuracy:- 0.8968%
filters: 5, average accuracy:- 0.9162%
```

```
filters: 6, average accuracy:- 0.8066%
filters: 6, average accuracy:- 0.8852%
filters: 6, average accuracy:- 0.8924%
filters: 6, average accuracy:- 0.9016%
filters: 6, average accuracy:- 0.9154%
```

```
filters: 7, average accuracy:- 0.8304%
filters: 7, average accuracy:- 0.8928%
filters: 7, average accuracy:- 0.9076%
filters: 7, average accuracy:- 0.9046%
filters: 7, average accuracy:- 0.9162%
```

```
filters: 8, average accuracy:- 0.8126%
filters: 8, average accuracy:- 0.891%
filters: 8, average accuracy:- 0.896%
filters: 8, average accuracy:- 0.9016%
filters: 8, average accuracy:- 0.9098%
```

Testowanie i użycie

Natomiast jeśli nauczyciel nie wie, lub chce przetestować sieć do jakiej klasy należy obraz i chciałby taką informację uzyskać od sieci - to jest to proces zwykłego wykorzystania sieci - „w przód”, i nawet nie ma podstaw by korygować stan sieci.

Wyliczanie wektora wyjściowego używamy zarówno jako krok wstępny przed procesem uczenia, a także w czasie testu stanu nauczania sieci.

proces przekazywania (propagacji) danych „do wyjścia”:

- 1) Obraz -----> Conv
- 2) Conv - splot -> Pool
- 3) Pool - max -> Softmax
- 4) Softmax - flat -> Softmax
- 5) Softmax - *W - +bias -> Softmax
- 6) Softmax - rozkład -> wektor wyjściowy Z

Funkcja klasy **Teacher**, **test()** wylicza odpowiedź sieci na pokazany obraz, czyli prawdopodobieństwa przynależności obrazu do wszystkich klas. Suma wszystkich prawdopodobieństw wynosi 1. po sprawdzeniu sekwencji obrazów w losowej kolejności klasa testowa pokazuje błąd międzyklasowy błąd klasyfikacji, oraz ilość bezwzględnie źle przyporządkowanych obrazów.

Wynik działania cykli uczenie i testowania dla obrazów z bazy MNIST

(do projektu załączyłem pliki z bazy MNIST - nieco jednak przykrojone z uwagi na ich objętość i ograniczenie wielkości)

Dodatkowe baza obrazów
autor: Kamil Heinzelman



każdy cykl uczenia kończy się wygenerowaniem raportu z podsumowaniem błędu międzyklasowego, aplikacja jest skonfigurowana na pokazanie 1000 obrazów w cyklu testu.

** TEST ** errors 124

incorrect class ->		[0]		[1]		[2]		[3]		[4]		[5]		[6]		[7]		[8]		[9]
True class	(0)		1		.		.		1
True class	(1)	.		.		2		.		.		.		1		1		.		.
True class	(2)	3		.		.		.		1		.		2		1		2		1
True class	(3)	1		1		8		.		.		3		1		4		.		5
True class	(4)	.		2		2			1		.		6
True class	(5)	1		3		2		1		3		.		2		.		3		5
True class	(6)	3			1		.		1		1		.
True class	(7)	1		1		4		.		3			8
True class	(8)	2		3		8		2		.		1		.		.		.		3
True class	(9)	1		2		1		.		4		.		1		3		.		.

Uruchomienie i konfiguracja aplikacji

Aplikacja wymaga zainstalowania środowiska Java w wersji min. 19
<https://jdk.java.net/19/>

Uruchomienie następuje po wydaniu komendy
java -jar CNN.jar

pliki z danymi powinny być umieszczone w tym samym folderze,
oraz mieć przypisane odpowiednie nazwy tj:

testImages
testLabel
trainImage
trainLabel

wynik działania:

G:\SieciNeuronoweProj\Projekt\CNN->C:\opt\jdk-19\bin\

java -jar CNN.jar

filters: 8, average accuracy:- 0.8154%

** TEST ** errors 135

incorrect class ->	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
True class (0)	.	.	1	2	.
True class (1)	1
True class (2)	2	1	.	.	1	.	3	1	1	3
True class (3)	3	.	4	.	.	1	2	.	3	2
True class (4)	4	.	.	13
True class (5)	6	.	3	2	1	.	8	.	12	2
True class (6)	.	1	1
True class (7)	1	2	4	.	1	.	.	.	2	17
True class (8)	2	2	1	.	.	1	2	.	.	9
True class (9)	1	1	.	1	1	.	.	2	1	.

filters: 8, average accuracy:- 0.882%

** TEST ** errors 124

incorrect class ->	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
True class (0)	1	.	.	1
True class (1)	.	.	2	.	.	.	1	1	.	.
True class (2)	3	.	.	.	1	.	2	1	2	1
True class (3)	1	1	8	.	.	3	1	4	.	5
True class (4)	.	2	2	1	.	6
True class (5)	1	3	2	1	3	.	2	.	3	5
True class (6)	3	1	.	1	1	.
True class (7)	1	1	4	.	3	8
True class (8)	2	3	8	2	.	1	.	.	.	3
True class (9)	1	2	1	.	4	.	1	3	.	.

filters: 8, average accuracy:- 0.896%

...

Uwaga: w plikach Obrazów pierwszych 6 bitów jest pomijanych , w plikach klas pomijanych jest pierwszych 8 bitów. Pliki odpowiadają strukturze plików bazy MNIST i sa plikami binarnymi odczytywanymi jako wartości od 0-255 na piksel.

podgląd klas pliku (przy wykorzystaniu klasy użytkowej Tools):

0

3

9

4

7

```
,!;2
~@! .
,$@@@,
6=@@@@,
7=@@*=@! .
;@@; . ;@;
-$@; #!
,$@*, #$.
;@- ;@;
,$@; , $=
.#@$. $=
-@! $=
=@; . ~@=
=@: :@=
=@: #@,
=@: ;@! 3
=@$: ,=@@:
:@@#! ;=@@; 3
.=@@@@@@@#: 2
1, #@@@! : 4

;*, , , ---: ,
~@@$###@@@@@:
;$@@@@@###$@~
, ~! : ~ =@#
. * @ $
! @ @ !
: @ @ # ,
: @ @ # ,
# @ # ,
, @ $ : .
; @ @ * -
; @ @ * -
, 1 ; $ @ @ = .
6 ! ! , : ! @ ! ,
; @ , 2 ; @ :
$ * : @ *
$ $ , @ @ *
; @ ! , ~ @ @ ~
3 ! @ # ; ; ! @ @ ! -
; $ @ @ @ $ =
- * @ # : ,

. . # = * ~ 7
. : # @ @ @ @ @ * ~
, # @ @ @ * ~ = $ @ $ ,
, # @ @ # : . 4 * @ @ -
, = @ @ : . ; @ @ @ :
~ @ @ : . ; @ @ @ -
2 * @ # 6 3 * @ @ @ -
- @ @ ~ * @ @ @ @ -
= @ @ , ~ @ @ @ @ @ -
- @ @ - - @ @ = @ @ *
. ! @ $ 9 ~ # @ ~ . @ @ =
~ @ @ ! # @ $ - 4 - @ @ ~
, * @ @ @ @ ! - ; @ @ ;
2 ; ; ; 5 2 ; @ @ ;
; @ @ ~
~ @ @ ~
~ @ @ ~
~ @ @ ~
~ @ @ ~
8 # @ ;
: @ ~

. $ =
= @ =
: @ @ ;
. $ @ : 9
= @ =
* @ -
- @ =
= @ :
; @ : , ; ~
~ @ : , ! @ !
# $ * $ @ @ ;
: $ @ @ @ =
, ~ ! @ :
; @ ,
; $
; #
; =
; =
; =
; =
; =

~ # @ @ *
. ; @ @ $ @ ! 4
- # @ $ ~ - * @ ,
5 ; @ @ = 4 ! @ ,
- # @ * : * @ ,
. ~ ! @ # : 4 * $ ,
, $ @ @ @ = . * *
, # # , 2 , @ :
: @ :
$ $ 5
, $ ,
~ @ -
! @ -
5 # @ -
. @ @ ,
. @ #
. @ *
. @ ;
: @ ;
, @ ~
```

Podsumowanie:

Przy niewielkich próbkach w epoce widać dokładnie jak sieć uczy się obrazów, w pierwszych cyklach jej trafność bardzo szybko wzrasta od 15-20% do 70-80%. Później proces spowalnia i zbliża się do 90% i oscyluje wokół granicy 90%.

filters: 4, average accuracy:- 0.8156%

filters: 4, average accuracy:- 0.8796%

filters: 4, average accuracy:- 0.8884%

filters: 4, average accuracy:- 0.905%

filters: 4, average accuracy:- 0.8988%

Sieci neuronowe typu z warstwami splotowymi oraz z wyjściową warstwą softmax bardzo dobrze radzą sobie z problemami rozpoznawania i klasyfikacji obrazów. warstwy splotowe ułatwiają wyszukiwanie wzorców nawet jeśli sa one przesunięte lub nawet obrócone.

Zaprojektowana sieć dobrze radzi sobie z klasyfikowaniem obrazów do 10 podanych klas.

Działa bardzo szybko - zwłaszcza jeśli ma do dyspozycji jakąś kartę graficzną. nawet stosunkowo prosta karta Radeon RX 6600 przyspiesza w sposób zauważalny obliczenia, albo może inaczej, jej brak powoduje zauważalne spowolnienie pracy aplikacji.

Mimo że język Java nie jest specjalnie szybki, i nie ma wielu specjalizowanych bibliotek przyspieszających operacje arytmetyczne, aplikacja daje sobie świetnie radę.

Próbowałem dla porównania wykorzystać klasyczną sieć MLP do rozwiązania tego zadania, przy wektorze wejściowym o rozmiarze 784 uczenie sieby było bardzo czasochłonne. Sieć MLP często wypłasczała wszystkie wagi, i przestawała się "uczyć".

Możliwe, że wprowadzenie momentum poprawiłoby nieco sytuację, ale różnica pomiędzy MLP a CNN jest kolosalna. Oczywiście, nie należy lekceważyć sieci MLP, jednak do tak postawionego zadania CNN będzie po prostu dużo lepsza.