Sieć neuronowa MLP w problemie klasyfikacji ręcznie pisanych cyfr.

Piotr Grzybowski

09 listopada 2017

1 Opis problemu

Klasyfikacją możemy nazwać przypisanie każdego elementu z danego zbioru X do dokładnie jednego z k rozłącznych zbiorów Y_k. Przygotowane dane były reprezentowane przez ręcznie pisanych cyfr w postaci binarnych obrazów o rozmiarach $(10x7)$ pikseli, które były podzielone na dziesięć klas od zera do dziewiątki.

2 Proponowane rozwiązanie

Powyżej zaprezentowany problem zostanie rozwiązany przy użyciu sieci neuronowej wielowarstwowej, użytej jako klasyfikator.

2.1 Sieć neuronowa

Sieć neuronowa wielowarstwowa składa się z warstwy wejściowej, przynajmniej jednej warstwy ukrytej, warstwy wyjściowej. Każdy z neuronów takiej warstwy składa się z:

- Wektora wag o długości liczbie sygnałów wejściowych. Każdemu sygnałowi wejściowemu x_i, odpowiada dokładnie jedna kolejna waga w_i. Symbolicznie $W = [x_1, ..., x_N]$, gdzie N to liczba sygnałów wejściowych do neuronu.
- Stałej zwanej "biasem". Liczba rzeczywista.
- Funkcji aktywacji, według której obliczana jest wartość wyjścia neuronów w sieci neuronowej.

Wartość wyjścia neuronu jest liczona w sposób następujący (g - funkcja aktywacji)

$$Output = g(x^T w + b) \tag{1}$$

Poniżej krótki opis każdej z warstw uwzględnionych powyżej:

- **Warstwa wejściowa:** posłuży tylko jako miejsce do którego będziemy ładować dane, na których sieć będzie przeprowadzała operacje. Warstwa wejściowa charakteryzuje się tym, że ma takie samo wyjście jak wejście. W zakładanym modelu posłuży ona jako *'data loader (ang.)*, czyli będzie przyjmować dane ze świata wewnętrznego, przechowywać je w trakcie przeprowadzania operacji oraz przesyłać wyście do kolejnej warstwy. Warstwa wejściowa składa się z N neuronów, gdzie N zależy od wymiarowości danych wejściowych. Przykładowo w naszym wypadku warstwa wejściowa będzie składała się z 70 neuronów wejściowych, gdyż binarny obraz o rozmiarze $(10x7)$ możemy zapisać za pomocą 70-cioelementowego wektora.
- **Warstwa ukryta:** Składa się z K neuronów, gdzie ich liczba jest zależna od użytkownika
- **Warstwa wyjściowa:** Warstwa wyjściowa zawiera tyle neuronów, do ilu możliwych klas możemy przypisać nasze dane. Działa ona jako klasyfikator. Neuron z największą wartością aktywacji wskazuje na klasę, którą sieć zaklasyfikowała daną wejściową.

3 Zbiór danych

Zbiór danych został przygotowany przez studentów. Zawiera on 1744 przykłady ręcznie pisanych cyfr na czarno białym obrazie binarnym o rozmiarze (10x7). W procesie uczenie zostanie on podzielony na podzbiory rozłączne: treningowy, walidacyjny, testowy.

4 Badania

Podstawowym problemem w poprawności działania sieci neuronowej jest dobór odpowiednich hiperparametrów. W badaniach zbadane zostaną hiperparametry pod względem jakości predykcji jak i szybkości uczenia. Kolejnym problemem jest wybór sposobu optymalizacji funkcji kosztu. Kolejnym problemem jej przeciwdziałanie "overfittingowi", zbadana zostanie regularyzacja.

4.1 Szybkość uczenia w przypadku liczby neuronów w warstwie ukrytej

W części tej została zbadana wymagana liczba epok do wyuczenia sieci neuronowej. Wyniki badań przedstawia poniższa tabela:

Tabela 1:

Liczba neuronów	Liczba epok	Odchylenie	Skuteczność	Odchylenie
1	14	2	30,25%	7,12%
2	12	1	30,36%	5,09%
4	9	2	47,17%	6,22%
8	6	1	68,10%	3,44%
16	7	1	72,00%	2,19%
32	7	1	78,26%	3,96%
64	5	1	80,47%	1,13%
128	5	1	80,01%	2,11%
256	3	1	75,36%	4,32%
512	7	2	69,23%	2,76%
1024	nie uczy	nie uczy	nie uczy	nie uczy

Tabela1: Szybkośc i skuteczność uczenia sieci w zależności od liczby neuronów w warstwie ukrytej

Na podstawie powyższej tabeli można stwierdzić, że zbyt mała liczba neuronów w warstwie ukrytej wymaga większej liczby epok do wyuczenia modelu. Charakteryzuje się także tym, że skuteczność predykcji nie jest wysoka, mała liczba neuronów zaczyna dokładnie odzwierciedlać zbiór treningowy (ang. overfitting), natomiast nie są one w stanie w dobry sposób reprezentować danych w zbiorze, który nie uczestniczył w procesie uczenia. Po drugie zbyt duża liczba neuronów w warstwie ukrytej powoduje, że liczba wymaganych epok się zwiększa, jeżeli natomiast będzie ich zbyt dużo do wielkości posiadanego zbioru uczącego możemy zauważyć zjawisko nie douczenia (ang. underfitting), model nie jest w stanie zaprezentować dobrze nawet danych treningowych jeżeli jest ich niewystarczająco dużo.

4.2 Przebadanie różnych współczynników uczenia

W części tej została zbadana wielkość stałej uczenia w trakcie wyuczenia sieci neuronowej. Wyniki badań przedstawia poniższa tabela:

Tabela 2:

Stała uczenia	Liczba epok	Odchylenie	Skuteczność	Odchylenie
0,001	20	1	10,32%	5,60%
0,01	10	2	40.,44%	3,45%
0,1	9	1	72,12%	3,17%
0,5	5	1	80,47%	1,13%
1	5	1	84,36%	2,36%
1.3	7	1	86,66%	3,52%
1,5	7	1	82,36%	4,32%
2	10	2	81,15%	4,12%
5	3	1	74,33%	5,47%

Tabela1: Szybkośc i skuteczność uczenia sieci w zależności od stałej uczenia

Na podstawie powyższej tabeli można stwierdzić, że zbyt mała stała uczenia powoduje, że model może utknąć w minimum lokalnym jak i potrzebuje więcej epok aby wyuczyć się na danych treningo-

wych. Z kolei zbyt duża stałą uczenia powoduje, że możemy oscylować wokół minimum funkcji kosztu i nigdy do niego nie trafimy, powoduje to, że nasz model osiąga gorsze wyniki niż oczekujemy. Kiedy stałą uczenia jest dużo za dużo model nie uczy się w cale, podobnie w przypadku zbyt małej dążącej do zera stałej uczenia.

4.3 Wpływ liczebności zbioru uczącego

W części tej została zbadana wielkość zbioru treningowego w trakcie wyuczenia sieci neuronowej. Wyniki badań przedstawia poniższa tabela:

Tabela 3:

Zbiór Tr	Walidacyjny	Odchylenie
100	10,32%	5,60%
200	41.,44%	2,45%
400	65,12%	3,17%
800	80,47%	1,13%
1600	84,36%	2,36%

Tabela1: Skuteczność uczenia sieci w zależności od wielkości zbioru treningowego

Większy zbiór treningowy przeciwdziała overfittingowi. W naszym przypadku wymagana jest większa liczba danych do wyuczenia modelu do predykcji na wyższym poziomie. Powinno się zastosować proces Data Augumentation to generowania większej liczby danych, jednakże wykraczało to za ramy tego zadania. Z doświadczenia, zbyt duża liczba danych może spowodaować że model nie będzie się w stanie wyuczyć w akceptowalnym czasie bądź nigdy.

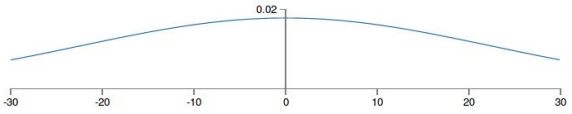
4.4 Metody optymalizacji

Przebadane zostały moteody optimalizacji gradientu prostego z dodatkiem momentum i bez. Momentum zapewnia nam szybsze zbieganie do minimum funkcji kosztu. W porównaniu do poprzednich eksperymentów, prrzy zastosowaniu momentum wystarczało do trzech epok przy 64 neuronach i współczynniku uczenia 1.3 aby szybko osiągac wyniki powyżej 80 procent na zbiorze walidacyjnym.

4.5 Wpływ inicjalizacji wartości wag początkowych

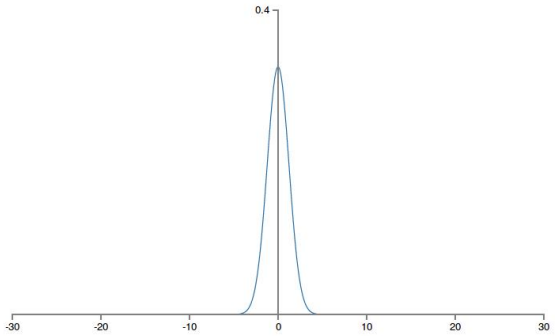
W zadaniu zostały zbadane dwa sposoby initializowania wag. Pierwszym z nich było zastosowanie Rozkładu Normalnego o średniej równej 0.0 oraz odchyleniu standardowym 1.0. Wiąże się to jednak z tym, że wagi nie będą mocno zróżnicowane. Rozkład losowanych zmiennych będzie płaski:

Tabela 4:



Lepszym sposobem będzie użycie Rozkładu Normalnego, z przeskalowanym odchyleniem standardowym o pierwiastek liczby neuronów w initializowanej warstwie. Spowoduje to, że rozkład losowanych zmiennych będzie bardziej zróżnicowany:

Tabela 5:



W przypadku drugiego rozkładu initializowanych wag uczenie przebiegało szybciej.

5 Podsumowanie

Uczenie sieci neuronowych jest pewną rodzaju sztuką. Można bardzo skrupulatnie dobierać hiperparametry, stosować różne funkcje kosztu, funkcji aktywacji, różnych metod optymalizacji co sprawia, że zadanie jest trudne. Jednakże warto korzystać także z pewnych wzorców i dobrych praktych, które będą działać poprawnie nawet dla naszych nowych problemów.