WSI - Raport z zadania 5

Wojciech Zieziula, Michał Pędziwiatr

Treść polecenia

1. Zaimplementować sieć neuronową - perceptron wielowarstwowy, oraz mechanizm jej uczenia algorytmem propagacji wstecznej gradientu.
2. Sprawdzić jakość działania algorytmu dla zadania regresji na zbiorze danych Wine Quality https://archive.ics.uci.edu/dataset/186/wine+quality.
3. Spróbować znaleźć konfigurację sieci, która pozwala osiągać dobre rezultaty.

Uwagi:

- Zbiór danych należy podzielić losowo na podzbiory uczący, i testowy (75-25). Podział musi być stały dla wszystkich eksperymentów, tj. należy ustawiać to samo ziarno generatora liczb losowych.
- Przed rozpoczęciem uczenia należy przyjrzeć się danym i odpowiednio je przygotować. W szczególności zwrócić uwagę na typ i zakres wartości atrybutów wejściowych oraz na brakujące wartości w zbiorze.
- Do wszystkich powyższych operacji, poza impementacją modelu, można używać gotowych funkcji bibliotecznych (polecam głównie `scikit-learn`), ale należy rozumieć funkcje, z których się korzysta.

Instrukcja

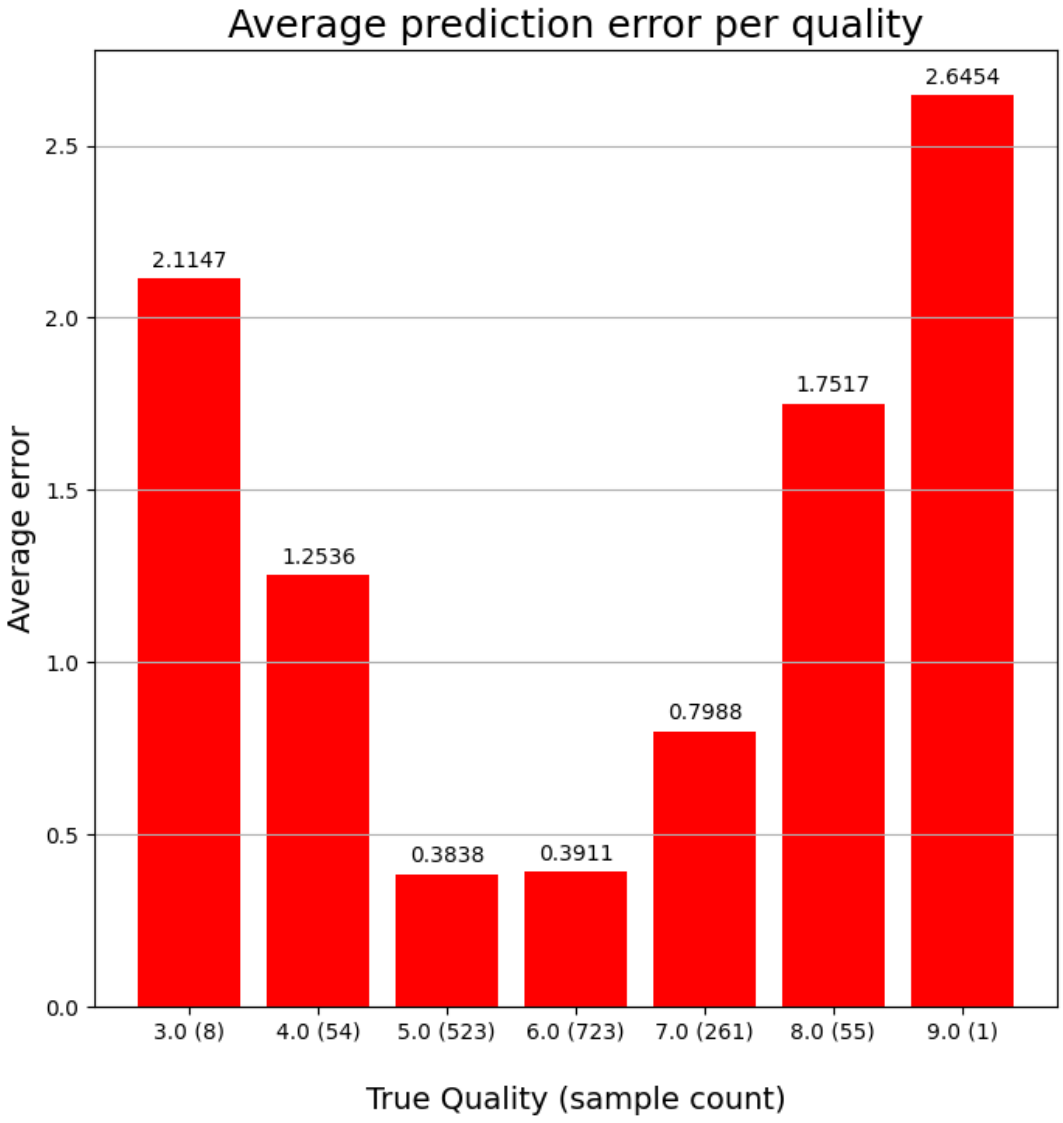
Argumenty przyjmowane przez skrypt:
--learning_rate <float> - długość kroku uczącego, domyślnie wynosi 0.0001
--epochs <int> - liczba iteracji algorytmu, domyślnie wynosi 500.
--hidden_layers <int …> - kolejne liczby wprowadzone wraz z tym argumentem będą traktowane jako ilości neuronów w każdej kolejnej warstwie ukrytej sieci neuronowej. Domyślnie wynosi [64, 32, 16] co tworzy sieć o 3 warstwach ukrytych, gdzie ilość neuronów w warstwie wynosi kolejno 64, 32 oraz 16.
--plot <store_true> - wyświetla wykres średniego błędu dla każdej kolejnej
--seed <int> - ziarno generatora liczb losowych.
--test_ratio <float> - informacja o tym jak duża część zbioru danych będzie podzbiorem testowym. Domyślnie wynosi 0.25, czyli 25% danych będzie stanowić podzbiór testowy.
--test_params <store_true> - włączenie trybu testowego, pozwalającego na wielokrotne wywołanie programu dla różnych parametrów, oraz porównanie jakości jego wyników. Tryb ten ignoruje wartości innych flag poza „—plot”.

Eksperymenty

Wpływ parametrów na optymalizacje działania perceptronu:

1: Wykres średniego błędu na prawdziwą ocenę wina dla parametrów:

- *Liczba iteracji: 10000,*
- *Współczynnik uczenia: 0,05*
- *Warstwy ukryte: [16, 8, 4, 2]*



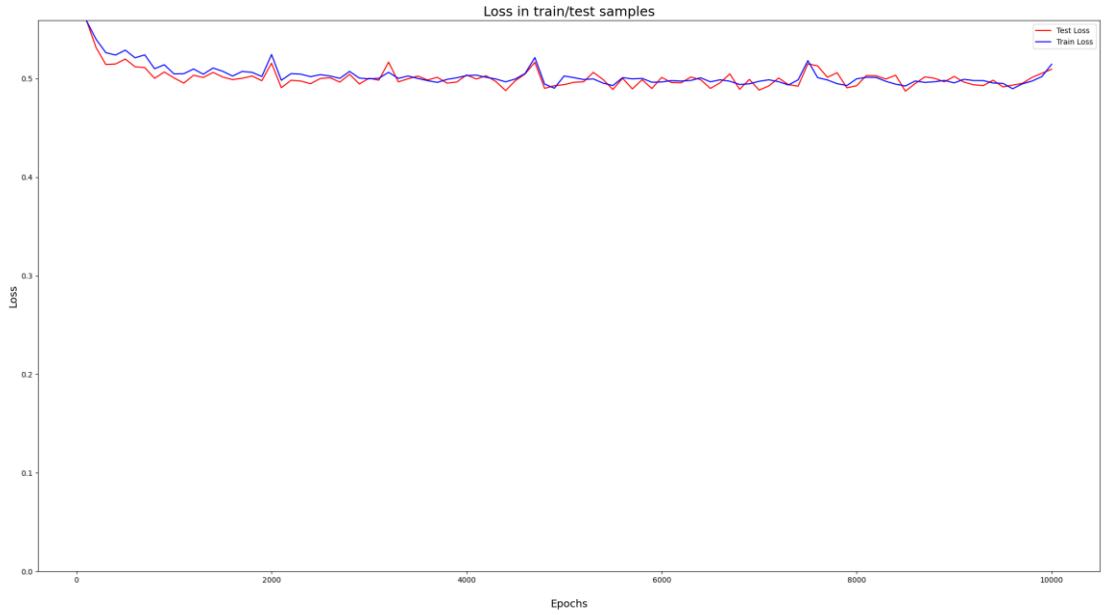
2: Pomiary oceny wyników perceptronu w zależności od jego parametrów początkowych:

Liczba epok	Współczynnik uczenia	Rozkład neuronów warstw ukrytych	Strata (MSE) przy próbce treningowej	Strata (MSE) przy próbce testowej	Średni błąd przy próbce testowej
100	0,1	[4, 2]	0,5535	0,5523	0,5861
		[4, 4, 4]	0,5588	0,5591	0,5929
		[8, 4, 2]	0,5715	0,5731	0,6033
		[16, 8, 4, 2]	0,6057	0,6257	0,6267
	0,05	[4, 2]	0,5733	0,5796	0,6023
		[4, 4, 4]	0,5858	0,5972	0,6140
		[8, 4, 2]	0,6037	0,6215	0,6251
		[16, 8, 4, 2]	0,6301	0,6548	0,6369
	0,01	[4, 2]	1,2136	1,2405	0,8611
		[4, 4, 4]	0,9203	0,9493	0,7692
		[8, 4, 2]	1,0234	1,0648	0,7973
		[16, 8, 4, 2]	1,1401	1,1980	0,8281
	0,001	[4, 2]	27,2422	27,3769	5,1545
		[4, 4, 4]	25,8207	25,9629	5,0121
		[8, 4, 2]	26,2352	26,3696	5,0500
		[16, 8, 4, 2]	22,8154	22,9694	4,6357
1000	0,1	[4, 2]	0,5047	0,5001	0,5422
		[4, 4, 4]	0,5330	0,5156	0,5581
		[8, 4, 2]	0,5043	0,4990	0,5431
		[16, 8, 4, 2]	0,5451	0,5447	0,5641
	0,05	[4, 2]	0,5278	0,5166	0,5500
		[4, 4, 4]	0,5146	0,5041	0,5479
		[8, 4, 2]	0,5207	0,5101	0,5514
		[16, 8, 4, 2]	0,5262	0,5132	0,5549
	0,01	[4, 2]	0,5523	0,5504	0,5836
		[4, 4, 4]	0,5480	0,5439	0,5822
		[8, 4, 2]	0,5686	0,5687	0,6006
		[16, 8, 4, 2]	0,5793	0,5848	0,6087
	0,001	[4, 2]	1,1268	1,1745	0,8243
		[4, 4, 4]	0,8402	0,8801	0,7318
		[8, 4, 2]	0,9699	1,0165	0,7773

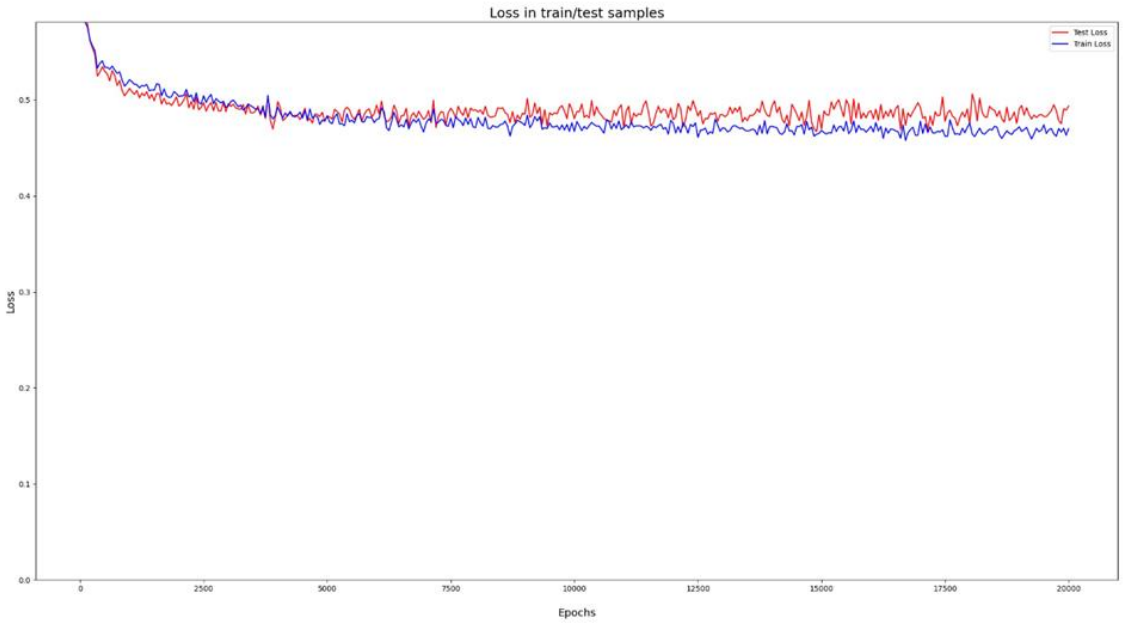
		[16, 8, 4, 2]	1,2071	1,2843	0,8430
2500	0,1	[4, 2]	0,5045	0,4975	0,5419
		[4, 4, 4]	0,5081	0,5168	0,5491
		[8, 4, 2]	0,5014	0,4865	0,5396
		[16, 8, 4, 2]	0,5043	0,4974	0,5418
	0,05	[4, 2]	0,5112	0,5064	0,5485
		[4, 4, 4]	0,5101	0,5031	0,5500
		[8, 4, 2]	0,5017	0,4970	0,5434
		[16, 8, 4, 2]	0,5075	0,4996	0,5441
	0,01	[4, 2]	0,5305	0,5225	0,5600
		[4, 4, 4]	0,5317	0,5199	0,5604
		[8, 4, 2]	0,5303	0,5190	0,5611
		[16, 8, 4, 2]	0,5554	0,5539	0,5899
	0,001	[4, 2]	0,6041	0,6200	0,6193
		[4, 4, 4]	0,5992	0,6188	0,6204
		[8, 4, 2]	0,6514	0,6797	0,6482
		[16, 8, 4, 2]	0,6854	0,7175	0,6634
10000	0,1	[4, 2]	0,4980	0,4920	0,5389
		[4, 4, 4]	0,5068	0,5078	0,5465
		[8, 4, 2]	0,4695	0,4956	0,5304
		[16, 8, 4, 2]	0,4513	0,4823	0,5230
	0,05	[4, 2]	0,4992	0,4953	0,5395
		[4, 4, 4]	0,5004	0,4938	0,5420
		[8, 4, 2]	0,4735	0,4924	0,5366
		[16, 8, 4, 2]	0,4594	0,4793	0,5340
	0,01	[4, 2]	0,5056	0,4947	0,5438
		[4, 4, 4]	0,5110	0,5079	0,5506
		[8, 4, 2]	0,5024	0,4950	0,5397
		[16, 8, 4, 2]	0,5116	0,4985	0,5431
	0,001	[4, 2]	0,5484	0,5451	0,5804
		[4, 4, 4]	0,5484	0,5442	0,5795
		[8, 4, 2]	0,5555	0,5518	0,5887
		[16, 8, 4, 2]	0,5967	0,6122	0,6191

3: Wykres przedstawiający stratę - błąd średniokwadratowy - wyniku na podzbiorze uczącym i testowym w zależności od liczby epok perceptronów dla parametrów:

- *współczynnik nauki: 0,1*
- *rozkład neuronów na warstwach ukrytych: [4, 2]*
- *ilości iteracji: 100-10000, interwał: 50*



- *współczynnik nauki: 0,05*
- *rozkład neuronów na warstwach ukrytych: [8, 4, 2]*
- *ilości iteracji: 100-20000, interwał: 50*



(*linia czerwona – strata dla podzbioru testowego, linia niebieska – strata dla podzbioru uczącego)*

Wnioski

Wpływ liczby iteracji na optymalizację perceptronu

Podobnie jak w innych programach implementujących wariację algorytmu „gradient descent”, jednym z najważniejszych hiper-parametrów, dla osiągnięcia zadowalających rezultatów predykcji, jest liczba iteracji („epochs”). Dobranie zbyt małych wartości tego parametru, szczególnie w połączeniu z niskim współczynnikiem uczenia, doprowadzać może do wysokiej wartości funkcji kosztu oraz średniego błędu, przez brak możliwości wystarczająco dokładnej analizy próbki uczącej. Innym potencjalnym problemem takiego obrotu zdarzeń, jest również „predykcja” tych samych wartości (np. 6). Taktyka ta, mimo tego, że w naszym przypadku doprowadza do stosunkowo niskich strat oraz średniego błędu przewidywań, nie będzie charakteryzować się wysoką celnością, szczególnie dla innych przypadków o bardziej równomiernym rozkładzie wartości.

Zwiększenie liczby iteracji natomiast, dla większości przypadków wydaje się zwiększać dokładność wyników. Mimo wszystko jednak, należy pamiętać że wyższe ich wartości zwiększają także wymaganą moc obliczeniową programu oraz mogą doprowadzić do tzw. „przeuczenia”. Na potencjał wystąpienia tego zjawiska może wskazywać przedstawiony powyżej wykres nr 3, porównujący stratę perceptronu dla zbioru uczącego oraz testowego. Widzimy na nim subtelną lecz wyraźną rozbieżność, rosnącą wraz ze wzrostem liczby epok, wywołaną przez wzrost straty w przypadku zbioru testowego lecz spadek w przypadku uczącego. Można więc stwierdzić, iż zbyt długi proces uczenia może zbytnio wyspecjalizować sieć pod zbiór uczący, która dopasowując do niego wagi oraz „biasy” w bardzo dokładny sposób, nie będzie odpowiednio przygotowana na próbkę testową. Z tego też względu, optymalnymi wydają się wartości w okolicach 10 tysięcy iteracji.

Wpływ współczynnika uczenia na optymalizację perceptronu

Mimo tego, że mniejsze wartości współczynnika uczenia mogą doprowadzić do większej dokładności programu, jak widzimy w wynikach pomiarów, wymaga to często niezwykle wysokich liczb iteracji. Wartości straty będące znacząco większe przy połączeniu niskich liczb epok (np. 100) z niskimi współczynnikami uczenia (np. 0,001) , mogą świadczyć o tym, że sieć nie zdąża zoptymalizować odpowiednio wartości wag, zanim program zakończy fazę uczenia. Dlatego też, mimo faktu, iż poprzez zwiększenie współczynnika uczenia w pewnym sensie zmniejszamy jego potencjał na najdokładniejsze wyniki, większe wartości rzędu 0,1 lub 0,05 wydają się optymalne dla większości konfiguracji.

Wpływ rozkładu neuronów na warstwach ukrytych na optymalizację perceptronu

Zmiana liczb warstw ukrytych oraz rozkładu neuronów na nich się znajdujących ma zdecydowanie dużo bardziej złożony i mniej przewidywalny wpływ na optymalizację od innych parametrów. Zwiększając liczbę wag, czyli także połączeń pomiędzy neuronami pogłębiamy znacząco możliwości analizy danych sieci, lecz także zwiększamy szanse na potencjalne pomyłki w postaci błędnie przypisanych wag oraz dostrzegania negatywnych dla wyników korelacji. W większości przypadków, można wysunąć wniosek, że ze względu na dość prostą strukturę naszego perceptronu oraz małą ilość neuronów wejścia i wyjścia, korzystniejsze dla optymalnego działania są mniejsze liczby warstw z mniejszą liczbą neuronów (takie jak [4, 2]). Bardziej złożone układy warstw ukrytych zdają się czasem działać lepiej dla wyższych liczby iteracji oraz mniejszych wartości współczynnika uczenia, choć bardzo ciężko dostrzec w tym przypadku wyraźne tendencje zmian wartości błędu.