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.

Eksperymenty

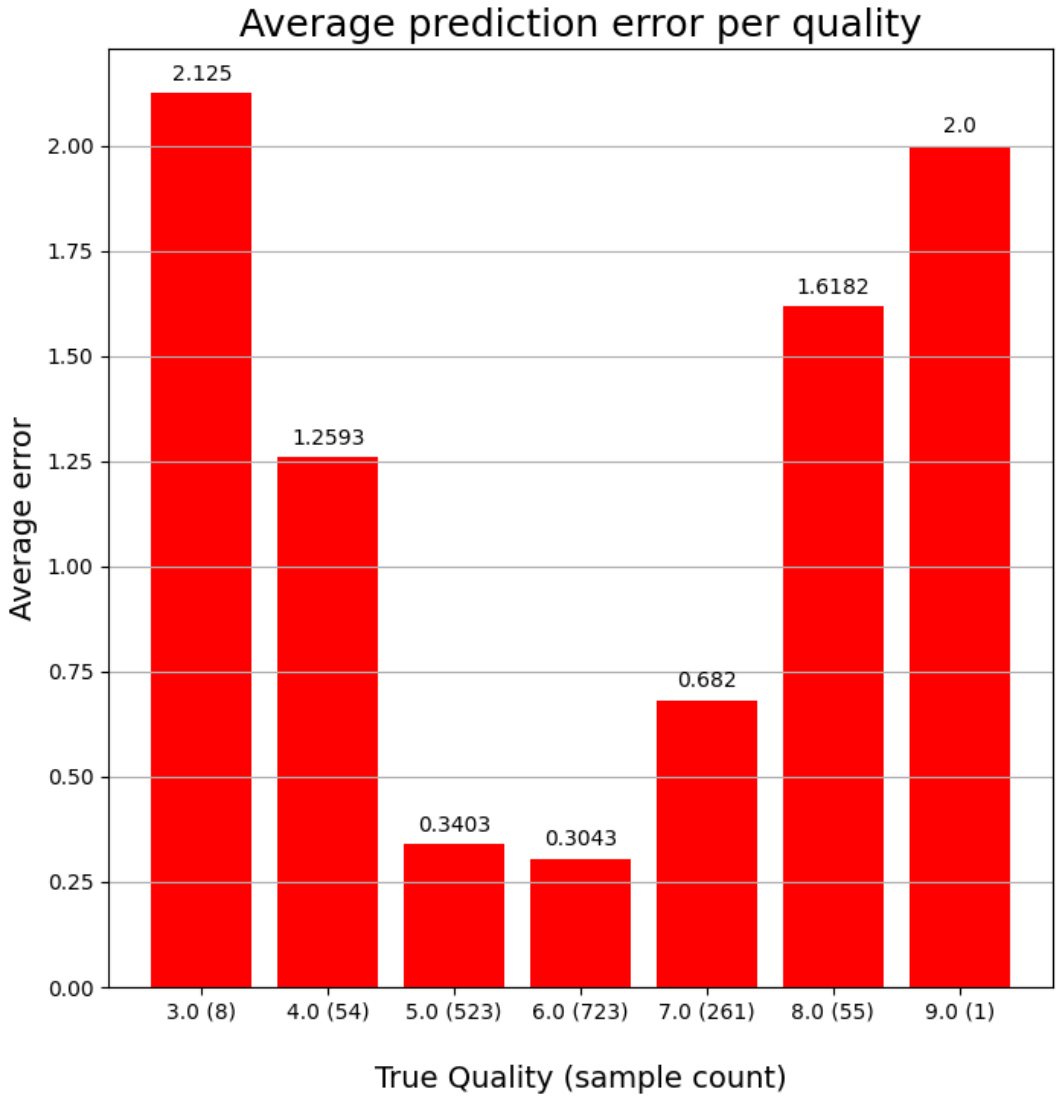
Wpływ parametrów na optymalizacje działania perceptronu:

Liczba iteracji	Współczynnik uczenia	Rozkład neuronów na warstwach ukrytych	Strata (MSE)	Średni błąd predykcji
100	0,1	[4, 2]	0,6825	0,5274
100	0,1	[4, 4, 4]	0,8006	0,6332
100	0,1	[8, 4, 2]	0,8006	0,6332
100	0,1	[16, 8, 4, 2]	0,8006	0,6332
100	0,05	[4, 2]	0,8006	0,6332
100	0,05	[4, 4, 4]	0,8006	0,6332
100	0,05	[8, 4, 2]	0,8006	0,6332
100	0,05	[16, 8, 4, 2]	0,8006	0,6332
100	0,001	[4, 2]	24,0714	4,8271
100	0,001	[4, 4, 4]	24,0714	4,8271
100	0,001	[8, 4, 2]	24,0714	4,8271
100	0,001	[16, 8, 4, 2]	24,0714	4,8271
100	0,0001	[4, 2]	34,7255	5,8271
100	0,0001	[4, 4, 4]	34,7255	5,8271
100	0,0001	[8, 4, 2]	34,7255	5,8271
100	0,0001	[16, 8, 4, 2]	34,7255	5,8271
1000	0,1	[4, 2]	0,5785	0,4862
1000	0,1	[4, 4, 4]	0,5926	0,4929
1000	0,1	[8, 4, 2]	0,5723	0,4812
1000	0,1	[16, 8, 4, 2]	0,5865	0,4892
1000	0,05	[4, 2]	0,5957	0,4948
1000	0,05	[4, 4, 4]	0,5957	0,4948
1000	0,05	[8, 4, 2]	0,5957	0,4972
1000	0,05	[16, 8, 4, 2]	0,8006	0,6332
1000	0,001	[4, 2]	4,1089	1,8369
1000	0,001	[4, 4, 4]	4,1089	1,8369
1000	0,001	[8, 4, 2]	4,1089	1,8369
1000	0,001	[16, 8, 4, 2]	4,1089	1,8369
1000	0,0001	[4, 2]	24,0714	4,8271
1000	0,0001	[4, 4, 4]	24,0714	4,8271

1000	0,0001	[8, 4, 2]	24,0714	4,8271
1000	0,0001	[16, 8, 4, 2]	24,0714	4,8271
2500	0,1	[4, 2]	0,5895	0,4923
2500	0,1	[4, 4, 4]	0,5877	0,4917
2500	0,1	[8, 4, 2]	0,8006	0,6332
2500	0,1	[16, 8, 4, 2]	0,5606	0,4720
2500	0,05	[4, 2]	0,5908	0,4862
2500	0,05	[4, 4, 4]	0,5945	0,4911
2500	0,05	[8, 4, 2]	0,5809	0,4837
2500	0,05	[16, 8, 4, 2]	0,8006	0,6332
2500	0,001	[4, 2]	1,4548	0,9132
2500	0,001	[4, 4, 4]	0,8006	0,6332
2500	0,001	[8, 4, 2]	0,8006	0,6332
2500	0,001	[16, 8, 4, 2]	1,4548	0,9132
2500	0,0001	[4, 2]	24,0714	4,8271
2500	0,0001	[4, 4, 4]	24,0714	4,8271
2500	0,0001	[8, 4, 2]	24,0714	4,8271
2500	0,0001	[16, 8, 4, 2]	24,0714	4,8271
10000	0,1	[4, 2]	0,5889	0,4942
10000	0,1	[4, 4, 4]	0,5748	0,4825
10000	0,1	[8, 4, 2]	0,5465	0,4665
10000	0,1	[16, 8, 4, 2]	0,8006	0,6332
10000	0,05	[4, 2]	0,5840	0,4880
10000	0,05	[4, 4, 4]	0,5815	0,4880
10000	0,05	[8, 4, 2]	0,5729	0,4868
10000	0,05	[16, 8, 4, 2]	0,5735	0,4874
10000	0,001	[4, 2]	0,6609	0,5391
10000	0,001	[4, 4, 4]	0,7735	0,6160
10000	0,001	[8, 4, 2]	0,8006	0,6332
10000	0,001	[16, 8, 4, 2]	0,8006	0,6332
10000	0,0001	[4, 2]	4,1089	1,8369
10000	0,0001	[4, 4, 4]	4,1089	1,8369
10000	0,0001	[8, 4, 2]	4,1089	1,8369
10000	0,0001	[16, 8, 4, 2]	1,4548	0,9132

Wykres średniego błędu na prawdziwą ocenę wina dla parametrów:

- *Liczba iteracji: 10000,*
- *Współczynnik uczenia: 0,1*
- *Warstwy ukryte: [8, 4, 2]*



Wnioski

Wpływ ilości 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, lecz należy pamiętać o tym, że także zwiększa wymaganą moc obliczeniową programu oraz może doprowadzić do tzw. „przeuczenia”.

Wpływ współczynnika uczenia na optymalizację perceptronu

Mimo tego, ze mniejsze wartości współczynnika uczenia mogą doprowadzić do większej dokładności algorytmu, jak widzimy w wynikach pomiarów, wymaga to często niezwykle wysokich liczb iteracji. Wartości średniego błędu, takie jak 4,8271 dla learning_rate=0,0001 przy aż 2500 iteracjach, lub 1,8369 dla 10000 iteracji, 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 jednak działać lepiej dla wyższych liczby iteracji oraz mniejszych wartości współczynnika uczenia, choć ciężko dostrzec w tym przypadku wyraźne tendencje zmian wartości błędu.