2 Sprawozdanie z laboratorium Programowania Sieciowego

Grzegorz Podlaski

218738

data oddania spr: 24.05.2019

termin zajęć: pt TN 19

Perceptron wielowarstwowy. Użycie warstwy „ukrytej”.

Perceptron jako jeden element nie jest używany obecnie na większą skalę. Jego najpoważniejszą wadą, wynikają z budowy i sposobu implementacji jest dwupoziomowa decyzja. Jest to za mało przy analizie bardziej skomplikowanych zbiorów danych czy obiektów. Potęga leży jednak w ilości. Dlatego warstwy z których składają się sieci neuronowe często charakteryzują się wieloma setkami poszczególnych neuronów, które reagują na odrębny impuls wejściowy. Dla przykładu można tu powołać dane mnist.

Mnist to usystemowane dane, dla których każda kolumna zawiera pewną informację o obrazku. Często nie są to nawet obrazy, a jakieś zdarzenia, sytuacje lub nawet akcje.

Zadanie do wykonania

Zadanie polegało na użyciu metody tworzącej sieć wielowarstwowego sieci perceptronów. Znajduję się ona w bibliotece sklearn. Tworzy ona 1 warstwę neuronów w liczbie równej ilości danych wejściowych. Dla naszego przykładu cyfr będzie to liczba pomnożonej liczby kolumn oraz wierszy (a dokładniej wiersz z pliku CSV) reprezentująca liczbę jak plik png. Należy tu przeprowadzić proces standaryzacji danych. Następnie znaleźć takie parametry, dla których sieć będzie najskuteczniejsza. Tymi parametrami są: algorytm(adam, sgd, lbfgs); współczynnik uczenia się sieci(0.001,0.0001,0.00001) oraz liczba neuronów w warstwach ukrytych(10-100). Warstwy ukryte służą temu, aby wykrywać pewne zależności występowania w danych wejściowych, np. proste odcinki, łuki, okręgi. Następnie przypisać występowanie tych elementów do konkretnej liczby. Największy procent dobrze trafionych próbek będzie tym ekstremum. Dla poprawności badań powinno się wykonać wiele razy testowanie na jednakowym zestawie parametrów, lecz z przyczyn ograniczonych zasobów sprzętowych, przeprowadzone zostały jedynie podwójne badania na danych parametrach.

	ada			sgd			lgfgs		
	0.001	0.0001	1E-05	0.001	0.0001	1E-05	0.001	0.0001	1E-05
10	0.934	0.933	0.903	0.930	0.903	0.732	0.916	0.907	0.918
20	0.950	0.953	0.923	0.949	0.910	0.820	0.950	0.948	0.947
30	0.964	0.962	0.930	0.957	0.917	0.813	0.959	0.962	0.960
40	0.970	0.970	0.940	0.964	0.921	0.846	0.962	0.963	0.962
50	0.970	0.971	0.941	0.965	0.919	0.844	0.967	0.965	0.968
60	0.973	0.973	0.948	0.966	0.919	0.848	0.970	0.970	0.970
70	0.975	0.974	0.946	0.967	0.919	0.848	0.971	0.971	0.972
80	0.976	0.972	0.951	0.965	0.918	0.851	0.972	0.971	0.969
90	0.977	0.975	0.951	0.969	0.917	0.854	0.974	0.976	0.975
100	0.977	0.974	0.949	0.968	0.921	0.855	0.971	0.975	0.975

czas trwania algorytmu: 27390.85960793495 7H 36 MIN
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@: naj_eta: 0.001, naj_alg: adam, naj_licz_N: 90

wspolczynnik uczenia:	0.001	0.0001	1E-05	0.001	0.0001	1E-05	0.001	0.0001	1E-05
liczba neuronow	adam			sgq			lbfgs		
10	0.830	0.616	0.264	0.541	0.256	0.247	0.768	0.758	0.829
20	0.890	0.775	0.331	0.569	0.304	0.249	0.872	0.873	0.884
30	***0.925***	0.805	0.384	0.635	0.301	0.219	0.890	0.887	0.891
40	***0.929***	0.858	0.415	0.671	0.289	0.239	0.896	0.902	0.904
50	***0.928***	0.835	0.454	0.637	0.290	0.256	0.904	0.905	0.915
60	***0.939***	0.870	0.454	0.640	0.305	0.256	0.909	0.916	0.904
70	***0.940***	0.892	0.511	0.672	0.337	0.241	***0.925***	0.915	0.909
80	***0.940***	0.890	0.496	0.663	0.324	0.249	0.915	0.909	0.918
90	***0.941***	0.905	0.514	0.671	0.341	0.262	***0.924***	0.915	0.917
100	***0.946***	0.893	0.549	0.668	0.347	0.267	0.913	0.917	***0.924***

Wnioski

najlepiej sprawdził się algorytm adam. Możliwe że przy większej ilości neuronów lub przy lepszej korekcie variancji, inny algorytm mógł wyjść lepiej.