Projekt z przedmiotu "Podstawy Sztucznej Inteligencji"

Piotr Chachuła, Ilya Kuzmich, Valera Burau

January 24, 2019

1 Algorytm uczenia

Jako sposób uczenia sieci przyjety został algorytm najszybszego spadku. Jest to metoda gradientowa, która w każdej swojej iteracji zmienia wagi sieci w kierunku $-\nabla J(\theta_t)$ z predkościa β_t. Jeśli zdefiniujemy funkcje celu jako

$$J(\theta_t) = \frac{1}{2}\|Y - f(X;\theta_t)\|^2 \tag{1}$$

to proces aktualizacja wag bedzie przebiegał nastepujaco

$$\theta_t + 1 := \theta_t - \beta_t \frac{d}{d\theta_t^T}\frac{1}{2}\|f(x_t;\theta_t) - y_t\|^2 \tag{2}$$

Używajac propagacji wstecznej pochodna funkcji błedy po poszczególnych wagach bedzie miała wartość

$$\frac{dJ(\theta)}{d\theta} = \frac{1}{2}\frac{d(f(x_t;\theta_t)-y_t)^2}{d\theta} = \frac{1}{2}\frac{\delta(f(x_t;\theta_t)-y_t)^2}{\delta f(x_t;\theta_t)}\frac{\delta f(x_t;\theta_t)}{\delta\theta_t} = (f(x_t;\theta_t) - y_t)\frac{\delta f(x_t;\theta_t)}{\delta\theta_t} \tag{3}$$

Z uwagi na to, że równanie opisujace sieć jest dość złożone z powodu wystepowania hiperbolicznych funkcji aktywacji, przyjeliśmy stały wartość parametru β w każdej iteracji.
W naszej sieci postanowiliśmy aktualizować wagi po każdej próbce uczacej.

2 Środowisko pracy

Jako jezyk tworzenia naszej sieci zdecydowaliśmy sie na Python.
Uznaliśmy, że jako jezyk wyższego poziomu dysponujacy dobrymi bibliotekami matematycznymi, a jednocześnie zwracajac uwage na jego popularność bedzie on idealnym narzedziem do tworzenia, testowania i ogólnego poszerzania wiedzy z zakresu sieci neuronowych.

3 Ogólny model sieci neuronowej

Jako model sieci neuronowej postanowiliśmy przyjać sieć o 4 wejściach, jednej werstwie neuronów ukrytych(po wstepnym przejrzeniu danych uznaliśmy, że

bedzie to zupełnie wystarczajace, poza tym dodawanie warstw znaczaco zwieksza złożoność sieci) oraz jednym neuronie wyjsciowym. Neuronem wyjsciowym jest zwykła suma wyjść z warstwy ukrytej. Funkcja aktywacji neuronów ukrytych jest tangens hiperboliczny. Dodatkowo aby zwiekszyć szanse na to, że siec bedzie sie szybciej uczyła i żadziej wpadała w minima lokalne w procesie uczenia, postanowiliśmy normalizować wejscia sieci w przedziale $[-1; 1]$ (wartości minimalne oraz maksymalne brane ze zbioru uczacego).

4 Dobór parametrów

4.1 Liczba neuronów ukrytych

Na samym poczatku postanowikiśmy dobrać odpowiednia ilość neuronów ukrytych. Aby to zrobić przyjeliśmy stała wartość parametru $\beta = 0.01$ oraz podział danych uczacych do weryfikujacych odpowiednio $3 : 1$. Z uwagi na to że sieć służy jako model procesu statycznego, nie dynamicznego, zdecydowaliśmy sie przedstawiać wyniki działania sieci w postaci macierzy błedów klasyfikacji. Wyniki tych testów przedstawione sa ponieżej (w opisie tabel wartość N oznacza liczbe neuronów ukrytych). Dodatkowo uczenie było przeprowadzane dla 1000 iteracji uczacych (chyba że wystapiło kryterium stopu).
W tabelach przedstawione sa najlepsze wyniki z 10 prób uczenia.

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	38/12	0/0	0/0
	Versicolor	0/0	35/10	0/0
	Virginica	0/0	3/2	38/12

Table 1: Uczenie/Weryfikacja dla $N = 1$

Jak widać w tabeli 1, już dla jednego neuronu sieć radzi sobie całkiem nieźle. Warto jednak sprawdziź co sie stanie przy zwiekszaniu liczby neuronów.

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	38/12	0/0	0/0
	Versicolor	0/0	34/11	0
	Virginica	0	4/1	38/12

Table 2: Uczenie/Weryfikacja dla $N = 2$

Dla 2 neuronów (tab. 2) zmiany sa niezauważalne. Zwiekszyliśmy zatem liczbe neuronów ukrytych do 5.

Dla 5 neuronów można już zaobserwować drobna poprawe poprawności działania sieci (tab. 3. Można zatem spróbować zwiekszyć krytycznie zwiekszyć ilość neuronów np. do 100.

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	38/12	0/0	0/0
	Versicolor	0/0	36/11	0/0
	Virginica	0/0	2/1	38/12

Table 3: Uczenie/Weryfikacja dla $N = 5$

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	38/12	0	0
	Versicolor	0/0	4/1	0/0
	Virginica	0/0	34/11	38/12

Table 4: Uczenie/Weryfikacja dla $N = 100$

Jak widać w tabeli 4 dla tak dużej liczby neuronów siec całkowiecie przestała działać.
Mimo, że różnica była niewielka, postanowiliśmy zostać przy modelu z 5 neuronami w warstwie ukrytej ponieważ liczba parametrów sieci wciaż pozostawała mała.

4.2 Podział danych

Wiadomo już jak sieć radzi sobie dla podziału danych w stosunku $3 : 1$, warto zatem sprawdzić jaki bedzie efekt gdy odwrócimy ten stosunek.

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	12/38	0/0	0/0
	Versicolor	0/0	12/36	1/1
	Virginica	0/0	0/2	11/37

Table 5: Uczenie/weryfikacja dla podziału danych $1 : 3$

W tabeli 5 widać mały spadek wydajności sieci, co jest całkowicie zrozumiałe, dlatego warto jeszcze sprawdzić skrajna sytuacje podziału $9 : 1$.

		Actual Values		
		Setosa	Versicolor	Virginica
Predicted Values	Setosa	45/5	0/0	0/0
	Versicolor	0/0	43/4	0/0
	Virginica	0/0	2/1	45/5

Table 6: Uczenie/weryfikacja dla podziału danych $9 : 1$

Wyniki sa podobne, aczkolwiek należy pamietać, że dla tak małego zbioru danych weryfikacyjnych trudniej jest określić prawdziwa skuteczność działania sieci.

5 Wizualizacja propagacji wstecznej

Wizualizacja została zrealizowana przy użyciu biblioteki Pillow.
Zaimplementowana przez nas klasa rysuje struktóre sieci, wejścia oraz wyjścia.
Wizualizacja zmiany wag odbywa sie poprzez pokolorowanie odpowiednich synaps na odpowiednio:

- zielono - jeśli wartość wagi jest zwiekszana
- czerwono - jeśli wartość wagi jest zmniejszana

Dodatkowo, w celu poprawienia jakości nasycenia powyższych kolorów zmienia sie proporcionalnie do zmiany wagi.
Przykładowy obraz wygenorowany w taki sposób można zobaczyć na rys 1.

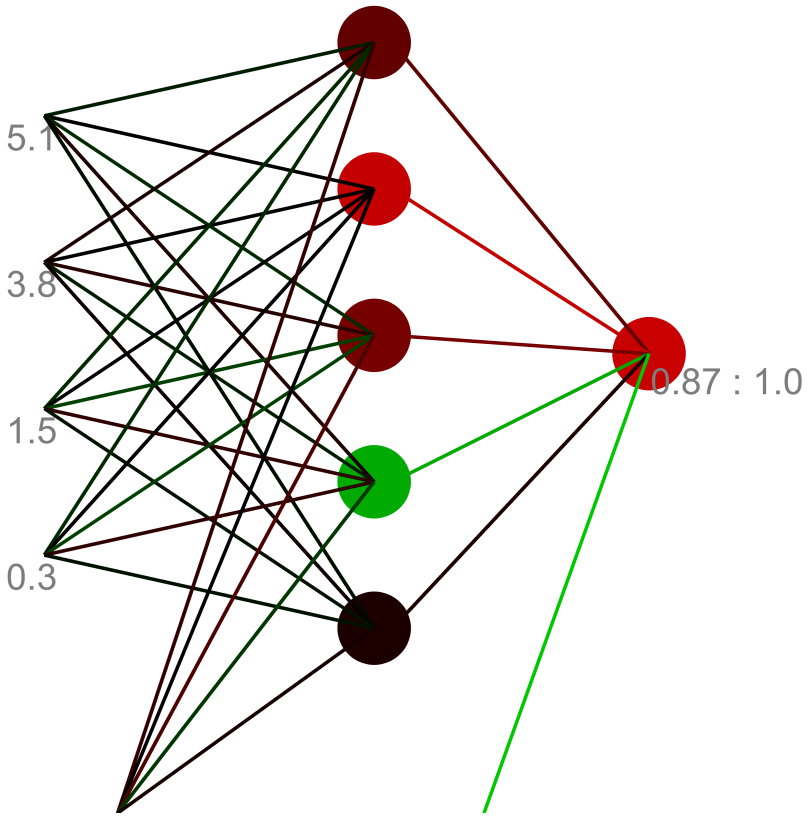

Figure 1: Sieć neuronowa i aktualizacja jej wag