Lista 6 (Lab) Termin wysłania na SVN do 24.05.2020

Na tej liście nie korzystamy z bibliotek uczenia maszynowego np. keras. Jej celem jest poznanie algorytmów uczenia sieci neuronowych.

1. (10pt) Zaimplementuj algorytm propagacji wstecznej przedstawiony na wykładzie 9 dla sieci neuronowej 3-4-1 (3-wejścia, 4-warstwa ukryta, 1-wyjście). Rozwiąż pokazany problem XOR. Następnie
   - zamiast funkcji sigmoidalnej

     $$\sigma(x) = \frac{1}{1 - e^{-x}}$$

     wykorzystaj funkcje aktywacji ReLU

     $$\mathrm{ReLU}(x) = \max(0, x).$$

   - wykorzystaj kombinacje funkcji sigmoidalnej σ z ReLU np. warstwa ukryta ma σ a warstwa wyjściowa jest ReLU lub inne kombinacje. Poeksperymentuj!

   Które rozwiązanie daje lepszą dokładność? Odpowiedź na to pytanie pisząc program testowy, który to pokazuje (np. po uruchomieniu w terminalu dostajemy wyniki dla różnych kombinacji z opisem). Rozwiąż ten problem również dla innych funkcji logicznych AND i OR. Dlaczego w danych wejściowych z przykładu z wykładu mamy ostatnią kolumnę z samymi jedynkami. Na to pytanie odpowiedź w komentarzu kodu źródłowego programu.

2. (15pt) Zaprojektuj sieć neuronową dwuwarstwową do aproksymacji funkcji np. sieć 1-5-1 lub 1-10-1 itp. i naucz ją funkcji (jedna funkcja dla jednej sieci)
   - Weźmy dane wejściowe, jako próbkowana funkcja paraboliczna

     ```
     >>> x=np.linspace(-50,50,26)
     >>> y=x**2
     >>> plt.scatter(x,y)
     ```

     Sieć testuj dla wektora wejściowego

     ```
     >>> x=np.linspace(-50,50,101)
     >>> y - wynik sieci dla wejścia x
     >>> plt.scatter(x,y)
     ```

   - Weźmy dane wejściowe, jako próbkowana funkcja sinus

     ```
     >>> x = np.linspace(0,2,21)
     >>> y = np.sin((3*np.pi/2) * x)
     >>> plt.scatter(x,y)
     ```

     Sieć testuj dla wektora wejściowego

     ```
     >>> x = np.linspace(0,2,161)
     >>> y = wynik sieci dla wejścia x
     >>> plt.scatter(x,y)
     ```

   Dobierz też odpowiednie dla problemu funkcje aktywacji! Dokładnie σ, ReLU lub $\tanh$. W czasie uczenia pokazuj aproksymacje funkcji co np. 100 krok, czyli co 100 krok np. dla funkcji parabolicznej dla wektora x = np.linspace(-50,50,101) pokaż wynik działania sieci wykorzystując matplotlib. Dostaniemy animacje procesu uczenia sieci. Wyświetlaj jeszcze aktualny krok uczenia oraz błąd średnio-kwadratowy. Animacje procesu uczenia wykonaj dla dwóch powyższych zbiorów danych (parabola i sinus).

3. (20pt)* Wykonaj zadanie drugie dla sieci o trzech warstwach np. 1-10-10-1. Sam napisz algorytm propagacji wstecznej dla sieci trójwarstwowej. Jakie wyniki aproksymacji dostaniemy dla tej ,,głębszej'' sieci?