Modelowanie Matematyczne - Lab 5

Adam Przemysław Chojecki

11 czerwca 2024

Teoria stojąca za sieciami neuronowymi

Sieci neuronowe są bardzo popularnym narzędziem do modelowania danych. Odpowiedzialne są za wiele sukcesów Sztucznej Inteligencji.
Sieci neuronowe są tak zwanymi "uniwersalnymi przybliżaczami" (ang. universal approximator). Jako matematyczną podstawę ich działania uważa się Twierdzenie o uniwersalnym aproksymatorze (ang. Universal Approximation Theorem). Na początek kilka definicji:

Definicja 1 *$I_n = [0,1]^m$. Funkcję σ nazwiemy dyskryminacyjną (ang. discriminatory), jeśli dla każdej miary $\mu \in M(I_m)$ warunek:*

$$\forall_{w\in\mathbb{R}^m,\theta\in\mathbb{R}} \int_{I_m} \sigma(w^\top x + \theta) d\mu(x) = 0$$

implikuje, że $\mu = 0$.

Przykład 1 *Następujące funkcje są dyskryminacyjne:*

- *$ReLU(t) = \max(0,t)$ i jego różniczkowalna wersja $GeLU(t)$*
- *Wszystkie ograniczone mierzalne funkcje sigmoidalne $\sigma(t)$, czyli:*
  $\lim_{t\to-\infty} \sigma(t) = 0$ oraz
  $\lim_{t\to+\infty} \sigma(t) = 1$

Twierdzenie 1 *Twierdzenie o uniwersalnym aproksymatorze (ang. Universal Approximation Theorem)*
Niech σ będzie ciągłą funkcją sigmoidalną. Wtedy zbiór:

$$\left\{ I_n \ni x \mapsto \sum_{j=1}^{N} \alpha_j \sigma(w_j^\top x + \theta_j) : N \in \mathbb{N}; \alpha_j, \theta_j \in \mathbb{R}; w_j \in \mathbb{R}^m \right\}$$

jest gęsty w $C(I_n)$.

Więcej informacji i dowody można znaleźć tutaj:

- "Approximation by Superpositions of a Sigmoidal Function" G. Cybenko
- `https://math.stackexchange.com/questions/4195328/understanding-why-relu-is-discriminatory-proof-understanding`

Zadanie - zbuduj sieć neuronową dla $m = 1$

1. Wybierz funkcję $\sigma(t)$, np. ReLU, GeLU, $sigmoid(t) = \frac{1}{1+e^{-t}}$, $arctan(t)$.

2. Wybierz małą stałą $\epsilon > 0$ zwaną stałą uczenia. Mniej więcej $\epsilon = 0.001$ powinien być dobrym wyborem.

   Wybierz $N \in \mathbb{N}$. Mniej więcej 15 powinno być ok.

3. Napisz funkcję `forward(x, w, alpha, theta)`, która przyjmuje

   - liczbę $x \in \mathbb{R}$;
   - wektory (tak zwane "wagi")
     $w = (w_j)_{j=1}^N, \alpha = (\alpha_j)_{j=1}^N, \theta = (\theta_j)_{j=1}^N \in \mathbb{R}^N$

   i zwraca $\sum_{j=1}^N \alpha_j \sigma(w_j \cdot x + \theta_j)$.

4. Wylosuj początkowe wartości `w, alpha, theta`.

   Nie ma tu dobrych ani złych wyborów, ale w moich eksperymentach dobrze sprawdziły się rozkłady:

   - `w, alpha` $\sim \mathcal{N}_N\left(0, (\frac{200}{N})^2\right)$
   - `theta` $\sim \mathcal{N}_N\left(0, (\frac{2000}{N})^2\right)$

5. Użyj kodu z drugich zajęć, z zadania pierwszego, aby wygenerować zaszumione dane. Wygeneruj $n = 100$ obserwacji $X = (x_i)_{i=1}^n$ z przedziału $[-20, -10] \cup [10, 20]$ oraz odpowiadające im zaszumione $Y = (y_i)_{i=1}^n$.

6. Napisz funkcję `MSE(X, Y, w, alpha, theta)`, która przyjmuje dane X, Y oraz wagi w, α, θ i zwraca błąd średniokwadratowy:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\texttt{forward}(x_i, \texttt{ w, alpha, theta}) - y_i)^2$$

7. Policz pochodną (gradient) MSE po `w, alpha, theta`. To bedzie zależało od X oraz Y.

8. Zaktualizuj wektory `w, alpha, theta` przy pomocy punktu 6 (mnożąc minus gradient przez stałą uczenia ϵ).

9. Policz MSE.

10. Powtarzaj kroki 7, 8 i 9 aż do zbieżności.

    - Jeśli MSE w jednym kroku się zwiększy, to znaczy, że należy zmniejszyć ϵ (np. 10 krotnie).
    - Jeśli MSE w jednym kroku zmniejsza się bardzo powoli, to można spróbować zwiększyć ϵ (np. 10 krotnie).

11. Narysuj przewidywane wartości $f(x)$ tak jak na zajęciach drugich. Jak zachowuje się estymator na zbiorze uczącym? Jak uogólnia się on poza zbiór uczący?

12. Powtórz eksperyment, wielokrotnie losując początkowe wartości wag. Przetestuj różne N.

13. Zobacz jak zachowa się proces uczenia sieci, która zaczyna z całkiem dobrej wartości wag:

    - $N = 3$
    - $w = (1, -1, 0)$
    - $\alpha = (-10, -15, 1)$
    - $\theta = (0, 0, 200)$