Zaawansowane Metody Uczenia Maszynowego
Praca Domowa 3

Adam Przemysław Chojecki

12 maja 2025

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 *$I_p = [0,1]^p$. Funkcję $\sigma\colon \mathbb{R} \to \mathbb{R}$ nazwiemy dyskryminacyjną (ang. discriminatory), jeśli dla każdej miary $\mu \in M(I_p)$ warunek:*

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

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:

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

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 z jedną warstwą ukrytą dla $p = 1$

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 $\epsilon = 0.0001$ powinien być dobrym wyborem.

   Wybierz liczbę neuronów w warstwie ukrytej $H \in \mathbb{N}$. 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") $w = (w_j)_{j=1}^H, \alpha = (\alpha_j)_{j=1}^H, \theta = (\theta_j)_{j=1}^H \in \mathbb{R}^H$

   i zwraca $\sum_{j=1}^H \alpha_j \sigma(w_j \cdot x + \theta_j)$.
4. Napisz funkcję `function_factory(w, alpha, theta)`, która przyjmuje te same parametry co `forward()` (poza `x`) i zwraca funkcję, która przyjmuje `x` i zwraca `forward()` zastosowany na tym `x` na wcześniej ustalonych wagach.
5. Wylosuj początkowe wartości `w, alpha, theta`.

   Nie ma tu dobrych ani złych wyborów. Wylosuj np. z rozkładu normalnego z wariancją $(\frac{2000}{H})^2$
6. Wygeneruj $N = 100$ obserwacji treningowych $(x_i, y_i)_{i=1}^N$ z modelu $Y_i = f(X_i) + \varepsilon_i$, $f(X) = \frac{\sin(12(X+0.2))}{X+0.2}$, $X \sim U([-0.8, 0.2] \cup [1, 2])$, $\varepsilon \sim N(0, 1)$.
7. Napisz funkcję `IMSE(g)`, która na wejściu przyjmie funkcję `g` i zwróci jej wycałkowany kwadratowy błąd względem funkcji $f(X)$ na dziedzinie $[-1, 2]$. Zwróć uwagę, że dziedzina do przecałkowania jest większa niż dziedzina danych uczących. To przecałkowanie może odbywać się poprzez klasyczne numeryczne całkowania biorąc punkty jednostajnie rozłożone (wzięte w równych odstępach) na dziedzinie całkowania $[-1, 2]$.
8. Napisz funkcję `MSE(X, Y, w, alpha, theta)`, która przyjmuje dane X, Y oraz wagi w, α, θ i zwraca błąd średniokwadratowy:

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

9. Policz (na kartce) pochodną (gradient) MSE po `w, alpha, theta`. Będzie ona zależała od X oraz Y.
10. Napisz funkcje `grad_w()`, `grad_alpha()`, `grad_theta()`, które będą przyjmować te same parametry co `MSE()` i zwracać gradient po odpowiednich parametrach.
11. Zaktualizuj wektory `w, alpha, theta` o wartość minus gradient razy stałą uczenia ϵ.
12. Policz MSE na zbiorze treningowym.
13. Powtarzaj kroki 11 i 12 aż do zbieżności MSE.
    - Pierwsze 5 iteracji mogą pogarszać MSE, ale później powinien się on już pomniejszać.
    - 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. 2 krotnie).

Porównaj wartość funkcji `IMSE()`:

- Na swojej nauczonej sieci.
- Na wyniku obliczeń smoothing splines (lab 10) dla różnych wartości parametru `s`.
- Na wyniku estymacji estymatorem gęstościowym (lab 10) dla różnych wartości parametru `s`.
- Dla sieci neuronowej w której przy aktualizacji wag dodatkowo odejmujemy $\lambda\cdot$`w`, $\lambda\cdot$`alpha`, $\lambda\cdot$`theta`. Jest to regularyzacja Ridge.
- Dla sieci neuronowej z regularyzacją LASSO. Jaka musi być zastosowana modyfikacja przy aktualizacji wag? (Przypomnij sobie funkcję miękkiego progowania)

Zinterpretuj wyniki.

Sugestie i uwagi

- Zastosuj wektoryzację dla przyśpieszenia obliczeń (wektory dla `R`, albo macierze typu `np.array` dla `Python`).
- Monitoruj MSE w kolejnych iteracjach i rysuj wykres MSE vs iteracja – pomoże to ocenić szybkość zbieżności i ewentualne zatrzymanie.
- Eksperymentuj z różną liczbą neuronów H, różnymi wartościami współczynnika uczenia ϵ i parametrem regularyzacji λ. Im większe H, tym (raczej) dłużej będzie trwać uczenie.