Лабораторна робота №8

Тема: «Градієнт. Градієнтний спуск».

Мета: Дослідження алгоритму градієнтного спуску.

Час виконання: 4 години.

Навчальні питання:

1). Етапи побудови алгоритму;

3). Підбір параметрів нейрона для найпростішого випадку з використанням регресійної моделі.

Теоретичні відомості

Алгоритм підбору ваг для мережі з заданою архітектурою.

Суть даного методу полягає в тому, що вводиться деякий критерій у вигляді функції, який необхідно мінімізувати.

Нехай:

L - кількість нейронів в мережі;

k_r - кількість нейронів в шарі *r*, де *r* = 1, 2, ……*L*;

k_L - кількість вихідних нейронів;

$k_0 = l$ - розмір входу;

$x(i) = \left(x_1(i), x_2(i), ...x_{k0}(i)\right)$ - вхідний вектор ознак;

$y(i) = \left(y_1(i), y_2(i), ...y_{kL}(i)\right)$ - вихідний вектор, який повинен бути вірно класифікований.

В поточному стані, мережа при навчанні дає результат $\hat{y}(i)$ який відмінний від $y(i)$

Позначимо:

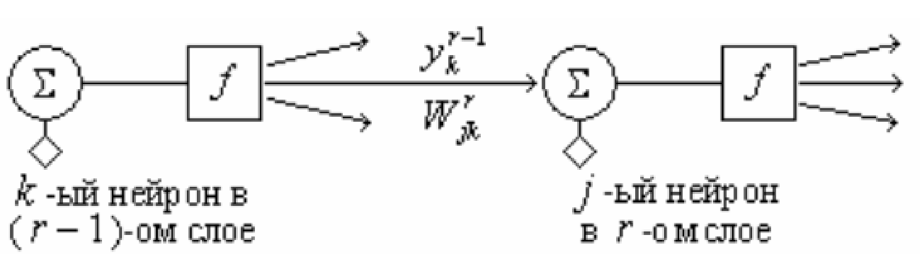
$$J = \sum_{i=1}^{N} \varepsilon(i),$$

де *N* – кількість прецендентів; $\varepsilon(i)$ - помилка на *i* – прецендентi;

$$\varepsilon(i) = \frac{1}{2}\sum_{m=1}^{kL} e_m^2(i) = \frac{1}{2}\sum_{m=1}^{kL}(y_m(i) - \hat{y}_m(i))^2,$$

де *i = 1,2……N*; *J* – функція всіх синаптичних ваг і порогів. Таким чином, приходимо до задачі оптимізації $J(W) \to \min$, яка є метою навчання мереж.

W – множина синаптичних ваг.



Нехай y_k^{r-1} - вихід *k*- го нейрона *(r-1)* – го шару; W_j^r - ваговий вектор (включаючи порогове значення) *j* – го нейрона в *r* – му шарі, тобто $W_j^r = \left(W_{j0}^r,\ W_{j1}^r, ..., W_{jk_{r-1}}^r\right)$, де

k_{r-1} - кількість нейронів в *(r-1)* – му шарі. Таким чином, *J* – розривна функція *M* змінних, де:

$$M = \sum_{i=1}^{N} k_{r-1} k_r \ .$$

Функція *J* розривна, оскільки розривна функція активації f :

$$f(x) = \begin{cases} 1, \ x > 0 \\ 0, \ x < 0 \end{cases} .$$

Метод градієнтного спуску для задачі мінімізації.

Нехай $W = \{W_j^r; \ j = 1,2.....,k; \ r = 1,2,......L\}$. Тоді метод градієнтного спуску виглядає так:

$\Delta W = -\mu \dfrac{dJ}{dW}$, де μ - крок градієнтного спуску. Для реалізації даного алгоритму, необхідно знаходити градієнт $\dfrac{dJ}{dW_j^r}$.

Обчислення градієнта.

Аргумент функції активації *j* – го нейрона *r* – го шару

$$V_j^r = \sum_{k=1}^{kr-1} W_{jk}^r y_k^{r-1}(i) + W_{j0}^r = \sum_{k=0}^{kr-1} W_{jk}^r y_k^{r-1}(i)$$

приймає різні значення в залежності від індекса прецендента. В даному випадку $y_0^{r-1}(i) = 1$.

У вхідному шарі, при *r* = 1, $y_k^{r-1}(i) = x_k(i), \ k = 1,2......k_0$. У вихідному шарі, при *r* = *L*, $y_k^r(i) = \hat{y}_k(i), \ k = 1,2,....k_L$. Розглянемо вихідний шар *r* = *L*.

$$\varepsilon(i) = \frac{1}{2} \sum_{m=1}^{k_L} (e_m(i))^2 = \frac{1}{2} \sum_{m=1}^{k_L} \left(f\left(V_m^L(i)\right) - y_m(i) \right)^2 = \varepsilon\left(V_m^L(i)\right) = \varepsilon\left(V_m^L\left(W_m^L\right), i\right)$$

$$\frac{\partial \varepsilon(i)}{\partial W_j^L} = \frac{\partial \varepsilon(i)}{\partial V_j^L} \cdot \frac{\partial V_j^L}{\partial W_j^L}$$

$\dfrac{\partial V_j^L}{\partial W_j^L} = y^{r-1}(i)$ - не залежить від *j* – го номера нейрона в шарі, тобто маємо однаковий вектор похідних для всіх нейронів *(r-1)* – го шару.

$$\frac{\partial \varepsilon(i)}{\partial V_j^L} = \left(f\left(V_j^L(i)\right) - y_j(i) \right) \cdot f'\left(V_j^L(i)\right) = e_j(i) \cdot f'\left(V_j^L(i)\right)$$

Значить для останнього шару $\dfrac{\partial \varepsilon(i)}{\partial W_j^L} = y^{r-1}(i) e_j(i) f'(V_j^L(i))$.

Розглянемо прихований шар *r* < *L*. Маємо залежність:

$$V_k^r = V_k^r\left(V_j^{r-1}\right)$$

$$\frac{\partial\varepsilon(i)}{\partial V_j^{r-1}(i)} = \sum_{k=1}^{k_r}\frac{\partial\varepsilon(i)}{\partial V_k^r(i)}\cdot\frac{\partial V_k^r(i)}{\partial V_j^{r-1}(i)}$$

$$\frac{\partial V_k^r(i)}{\partial V_j^{r-1}(i)} = \frac{\partial}{\partial V_j^{r-1}(i)}\left[\sum_{m=0}^{k_{r-1}} W_{km}^r y_m^{r-1}(i)\right],$$

але $y_m^{r-1}(i) = f(V_m^{r-1}(i))$, значить:

$$\frac{\partial V_k^r(i)}{\partial V_j^{r-1}(i)} = W_{kj}^r\frac{\partial y_j^{r-1}(i)}{\partial V_j^{r-1}(i)} = W_{kj}^r f'\left(V_j^{r-1}(i)\right)$$

$$\frac{\partial\varepsilon(i)}{\partial V_j^{r-1}(i)} = \left[\sum_{k=1}^{k_r}\frac{\partial\varepsilon(i)}{\partial V_k^r(i)}W_{kj}^r\right]\cdot f'\left(V_j^{r-1}(i)\right)$$

Сума в квадратних дужках, відома з попереднього кроку.

Опис алгоритму.

0. **Початкове наближення.** Випадково обираються ваги невеликих значень W_{jk}^r, *r= 1,2…..L, j=1,2……k*$_r$*, k=0,1,2,………k*$_{r-1}$.

1. **Прямий прохід.** Для кожного вектора прецендента $x(i)$, обчислюються всі $V_j^r(i)$ $y_j^r(i) = f(V_j^r(i))$, *j=1,2……k*$_r$*, r=1,2,……L.* Обчислюється також поточне значення функції $J(W)$:

Цикл по $i = 1,2,...,N$ (по прецедентам):

Вычислить:

$$y_k^0(i) = x_k(i), \quad k = 1,2,...,k_0 .$$

$y_0^0(i) = 1$.

Цикл по $r = 1,2,...,L$ (по слоям):

Цикл по $j = 1,2,...,k_r$ (по нейронам в слое):

$$V_j^r(i) = \sum_{k=0}^{k_{r-1}} W_{jk}^r y_k^{r-1}(i)$$

$$y_j^r(i) = f\left(V_j^r(i)\right)$$

Конец цикла по j .

Конец цикла по r .

Конец цикла по i .

$$J(W) = \sum_{i=1}^{N}\frac{1}{2}\left(y_j^L(i) - y_j(i)\right)^2$$

2. **Обернений прохід.** Для кожного значення *i=1,2……N, j=1,2,…… k*$_L$ обчислюється $\frac{d\varepsilon(i)}{dV_j^L(i)}$. Потім, послідовно обчислюється $\frac{d\varepsilon(i)}{dV_j^r(i)}$ для всіх *r=(L-1),…,1* та *j=1,2,…… k*:

```
Цикл по i=1,2,...,k_r (по нейронам в слое):
Вычислить:
```

$e_j(i) = y_j^L(i) - y_j(i)$

$\delta_j^L(i) = e_j(i) \cdot f'\left(V_j^{r-1}(i)\right)$

```
Цикл по r=L,L-1,...,2 (по слоям):
      Цикл по j=1,2,...,k_r (по нейронам в слое):
```

$$e_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i) \cdot W_{kj}^r$$

$$\delta_j^{r-1}(i) = e_j^{r-1}(i) \cdot f'\left(V_j^{r-1}(i)\right)$$

```
      Конец цикла по j.
Конец цикла по r.
Конец цикла по i.
```

3. **Перерахунок ваг.** Для всіх *r=1,2,……L* і *j=1,2……k*$_r$ $W_j^r(new) = W_j^r(old) + \Delta W_j^r$, *де* $\Delta W_j^r = -\mu \sum_{i=1}^{N} \frac{\partial \varepsilon(i)}{\partial V_j^r} y^{r-1}(i)$.

- Зупинка алгоритму може реалізуватися по двом критеріям: або $J(W)$ стала менше ніж поріг, або градієнт став досить малим;
- Від вибору μ залежить швидкість сходження алгоритму. Якщо μ не велике, то швидкість сходження теж мала. Якщо μ велике, то і швидкість сходження велика, але можна пропустити мінімум;
- Завдяки багатоекстремальності, існує можливість потрапити в локальний мінімум. Якщо мінімум по якимось причинам не підходить, потрібно починати алгоритм з іншої випадкової точки.

Завдання 1 (hello world для нейронних мереж):

Для **найпростішого** випадку одного нейрону підібрати методом градієнтного спуску параметри θ_1 і θ_0 для $y(\theta_1, \theta_0) = x_1 \cdot \theta_1 + \theta_0$.

Прийняти в першому наближенні $\theta_1 = 5;\ \theta_0 = 7$. Навчальна пара $\{x_1 = 1;\ y_1 = 1\}$. Швидкість навчання $\vartheta = 0.1$; точність $\varepsilon = 0.25$.

Завдання 2

Для умов **завдання 1** прийняти п’ять навчальних пар:

$\{x_1 = 1;\ y_1 = 1\}$;

$\{x_1 = 1.2;\ y_1 = 1.5\}$;

$\{x_1 = 2;\ y_1 = 3.45\}$;

$\{x_1 = 2.8;\ y_1 = 4.0123\}$;

$\{x_1 = 3.1;\ y_1 = 3.09\}$.

Для реалізації обчислювальних алгоритмів рекомендується використання онлайн середовищ тестування (наприклад repl.it, trinket, і.т.д.)

Захист лабораторної роботи передбачає виконання практичних завдань поставлених в роботі, та виконання завдань теоретичного характеру.