НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
Факультет прикладної математики
Кафедра прикладної математики

ЛАБОРАТОРНА РОБОТА

з дисципліни “Системи глибинного навчання”

на тему: “Розробка програмного забезпечення для реалізації двошарового персептрону з сигмоїдальною функцією активації”

Керівник:
Терейковський І. А.

Студентки IV курсу, групи КМ-03
Пюстонен С.Р.,

Київ – 2023

ЗМІСТ

ВСТУП

Постановка задачі

Частина 1

Завдання: розробити програмне забезпечення для реалізації класичного нейрону. Передбачити режим навчання на одному навчальному прикладі та режим розпізнавання.

Частина 2

Завдання: розробити програмне забезпечення для реалізації елементарного двошарового персептрону із структурою 1-1-1. Передбачити режим навчання на одному навчальному прикладі та режим розпізнавання.

Частина 3

Завдання: розробити програмне забезпечення для реалізації двошарового персептрону із структурою 2-3-1. Передбачити режим навчання «ON-LINE» та режим розпізнавання.

Піддослідна функція $x_1 + x_2 = y$

2Теоретична частина

2.1Структура штучного нейрона

Штучні нейрони, що також називаються нейронними клітинами, вузлами, модулями, моделюють структуру й функції біологічних нейронів.

Вхідними сигналами штучного нейрона $x_i(i = \overline{1,N})$ є вихідні сигнали інших нейронів, кожний з яких узятий зі своєю вагою $w_i(i = \overline{1,N})$, аналогічною синаптичній силі.

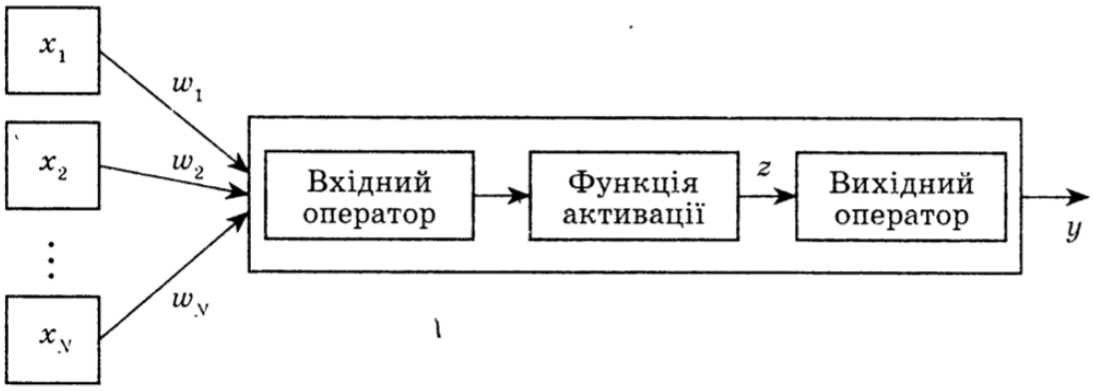


Рис. 2.1.1. Структура штучного нейрона

Нелінійний оператор перетворення вектора вхідних сигналів x у вихідний сигнал у може бути записаний у такий спосіб: $y = f_{\text{вих}}(f_a(f_{\text{вх}}(x, w)))$

Вхідний оператор нейрона - це математична функція, яка обчислює зважену суму (max/min/добуток) вхідних сигналів разом зі зваженим зсувом (bias), і результат цієї функції використовується як вхід для активаційної функції нейрона. Зважена сума та зсув – це параметр, які можна налаштовувати.

Функція активації описує правило переходу нейрона, що перебуває а момент часу k у стані $z(k)$, у новий стан $z(k + 1)$ при надходженні сигналів x.

$$z(k+1) = f_a(f_{\text{вх}}(x, w))$$

Активаційні функції:

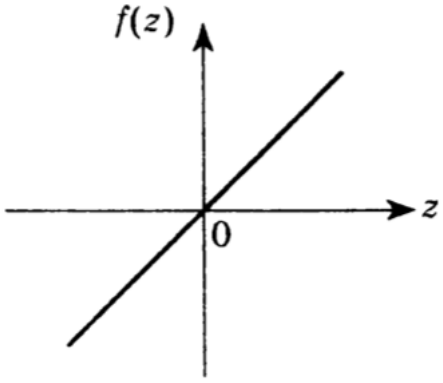


Рис. 2.1.2. Лінійна функція

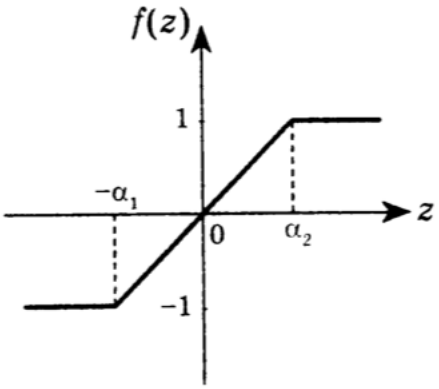


Рис. 2.1.3. Лінійна біполярна функція з насиченням

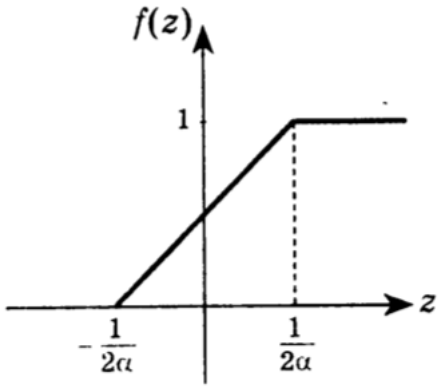


Рис. 2.1.4. Лінійна уніполярна функція з насиченням

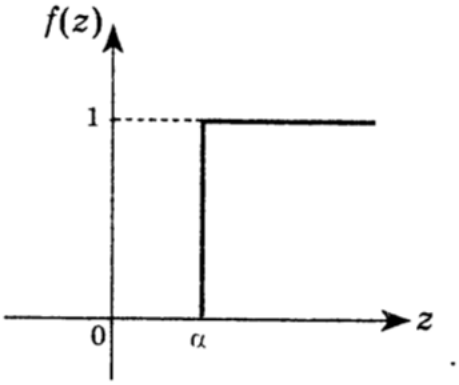


Рис. 2.1.5. Уніполярна порогова функція

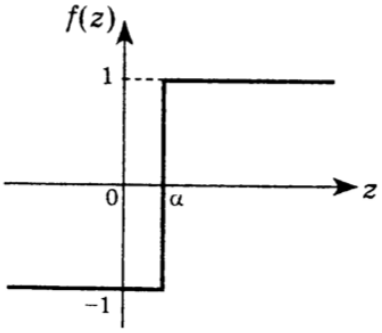


Рис. 2.1.6. Біполярна порогова функція

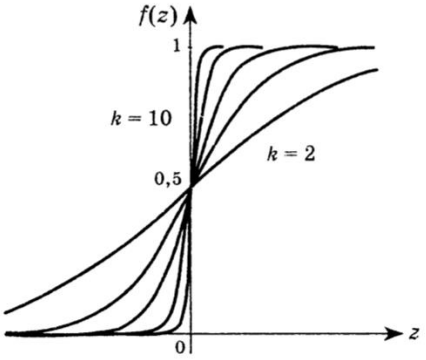


Рис. 2.1.7. Логістична функція

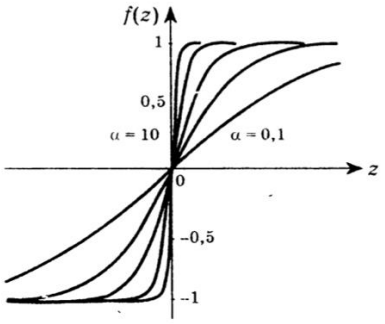


Рис. 2.1.8. Функція гіперболічного тангенса функція

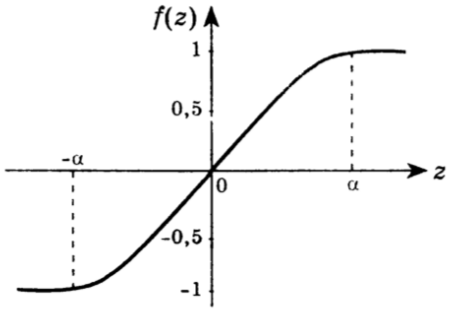


Рис. 2.1.9. Функція синусоїдальна з насиченням

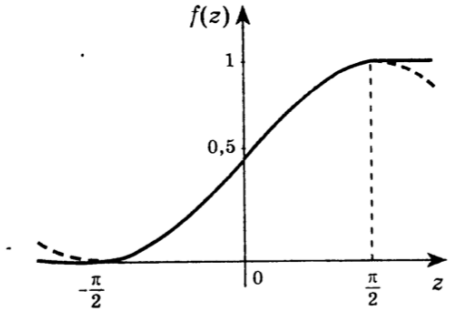


Рис. 2.1.10. Функція косинусоїдальна з насиченням

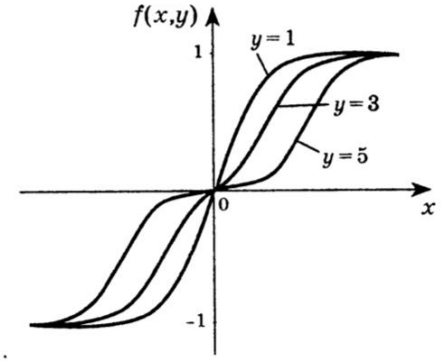


Рис. 2.1.11. Функція “модульована сигмоїда”

Вихідний оператор служить для представлення стану нейрона в бажаній області значень. Зазвичай у більшості робіт цей оператор не виділяють, а під вхідним сигналом нейрона розуміють сигнал після оператора активації. Однак під час аналізу й синтезу ШНМ, що містять різні активаційні функції, які мають різні області значень й області визначення, виникає необхідність використання такого оператора $f_{\text{вих}}$.

2.2 Формальна модель нейрона Маккаллоха-Піттса

Формальний штучний нейрон (нейрон Маккалоха-Піттса) може поданий як багатовхідний перетворювач із ваговими коефіцієнтами w_{ij}, які також називаються синаптичними вагами або підсилювачами.

Клітина тіла (сома) описується нелінійною обмежувальною або пороговою функцією. Найпростіша модель штучного нейрона підсумує єн ваг входів і здійснює нелінійне перетворення.

$$y_i = f\left(\sum_{i=1}^{N} w_{ij} x_i + \theta_j\right)$$

, де y_i – вихідний сигнал j-го нейрона

f – обмежувальна або порогова функція (активаційна)

N – кількість входів

w_{ij} – синапсичні ваги

x_i – вхідні сигнали

θ_j – пороговий сигнал (зсув)

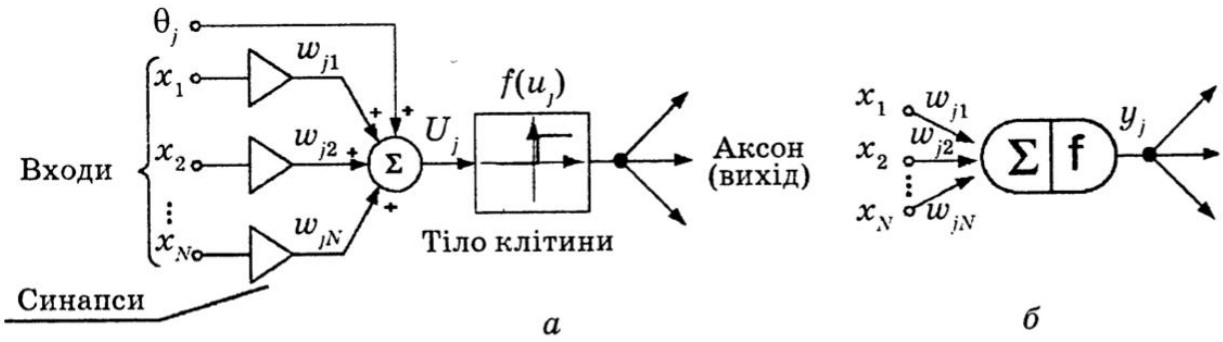


Рис. 2.2.1 Модель штучного нейрона Маккалоха-Піттса

Модель сигмоїдальної функції може бути побудована на основі традиційних електронних компонент. В електричному ланцюзі напруга імітує ***тіло клітини (сому),*** провідності замінюють вхідну структуру (***дендрити***) і вихідну (***аксон***) та різні резистори моделюють синапсичні ваги (***синапси***). Вихідна напруга імітує вихідну реакцію біологічного нейрона.

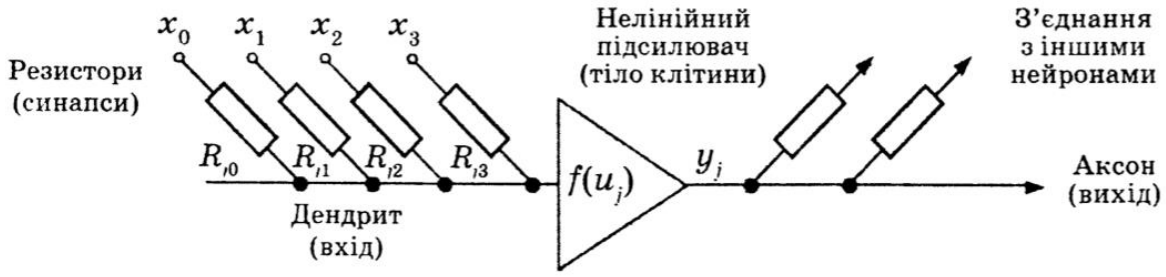


Рис. 2.2.2 Електронна аналогова модель нейронної клітини

2.3 Одношаровий перспептрон

Персептрон - це простий вид штучного нейрону або модель машинного навчання, яка була розроблена Френком Розенблаттом в 1957 році. Основна ідея персептрона полягає в тому, щоб мати бінарний класифікатор, який може приймати рішення на основі вагованих вхідних сигналів і порогової функції активації.

Одношаровий персептрон - це специфічний вид нейронної мережі, яка має лише один шар нейронів, і цей шар пов'язаний з вхідними даними.

Існують різні шляхи реалізації процесу навчання персептрона, однак у їхній основі лежить таке правило: вагові коефіцієнти персептрона змінюються тільки тоді, коли виникає розбіжність між його фактичною й бажаною реакціями.

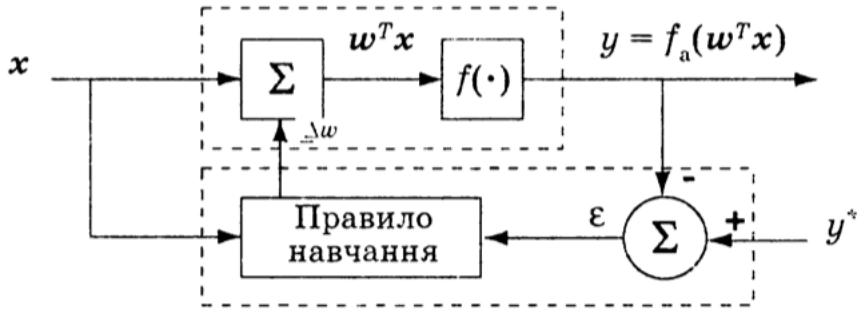


Рис. 2.3.1 Схема навчання перспетрона

Теорема збіжності стверджує, що якщо задача має розвязок, то він може бути отриманий за кінцеве число ітерацій. Це означає, що персептрон може

навчитися правильно класифікувати подані йому образи на кінцевому числі пар, що навчають.

2.4 Багатошаровий перспетрон

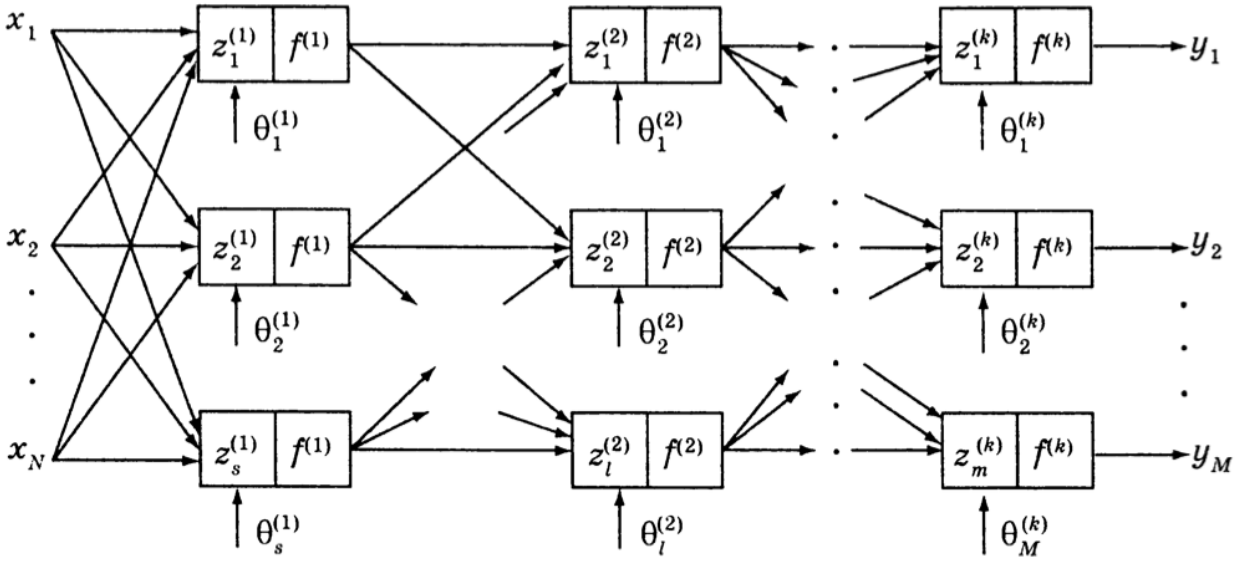


Рис. 2.4.1 Багатошаровий перспетрон

Багатошарові персептрони краще одношарових через їхню здатність моделювати складні нелінійні залежності в даних.

Алгоритм зворотнього поширення помилки (Backpropagation) - це метод навчання нейронних мереж, який використовується для оновлення ваг і зменшення помилок прогнозів. Він розповсюджує помилки від виходу мережі до її входу, коригуючи ваги за допомогою градієнтного спуску.
Алгоритм навчання:

1. Вибрати із заданої навчальної множини чергову пару $\left(x(i), y*(i)\right)\ i = \overline{1,P}$, що навчає, і подати на вхід мережі вхідний сигнал $x(i)$.
2. Обчислити реакцію мережі $y(i)$.
3. Порівняти отриману реакцію $y(i)$ з необхідною $y*(i)$ і визначити помилку $y*(i) - \ y(i)$.

4. Скорегувати ваги так, щоб помилка була мінімальною.
5. Кроки 1-4 повторювати для всієї множини пар, зо навчають, доти, поки на заданій множині помилка не досягне необхідної величини.

2.5 Резюмуємо, додаємо деталі. Реалізація нейронної мережі та навчання

4-рівнева нейронна мережа складається з 4 нейронів для вхідного рівня , 4 нейронів для прихованих шарів і 1 нейрона для вихідного рівня .

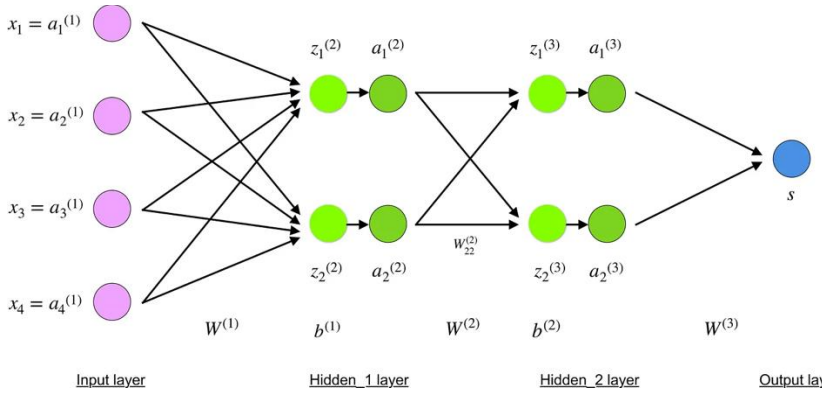


Рис. 2.5.1 Проста ілюстрація 4-шарової нейронної мережі

Вхідний шар. Нейрони, пофарбовані у фіолетовий колір , представляють вхідні дані. Вони можуть бути простими, як скаляри, або більш складними, як вектори чи багатовимірні матриці $x_i = a_i^1, i \in 1,2,3,4$.

Приховані шари. Остаточні значення на прихованих нейронах, пофарбовані в зелений колір , обчислюються за допомогою *z^l* — зважених вхідних даних у шарі l та *a^l* — активацій у шарі l . Для шарів 2 і 3 рівняння такі:

Для $l = 2$:

$$z^2 = W^1 x + b^1$$

$$a^2 = f(z^{(2)})$$

Для $l = 3$:

$$z^3 = W^2 a^2 + b^2$$

$$a^2 = f(z^{(3)})$$

$W^1, W^3 -$ це ваги в шараз 2 і $3, b^2$ і $b^3 -$ зміщення в цих шарах.

Активації a^2 і a^3 обчислюються за допомогою функції активації f . Як правило, ця функція f є нелінійною (наприклад, sigmoid , ReLU , tanh) і дозволяє мережі вивчати складні шаблони в даних.

Уважно подивившись, ви можете побачити, що

всі $x, x^2, a^2, z^3, a^3, W^1, W^2, b^1$ і b^2 не мають індексів, представлених на ілюстрації 4-рівневої мережі вище. Причина в тому, що ми об'єднали всі значення параметрів у матриці, згруповані за шарами. Це стандартний спосіб роботи з нейронними мережами, і обчислення повинні бути зручними.

$\boldsymbol{W^1}$ - вагова матриця форми (n, m) , де n — кількість вихідних нейронів (нейронів наступного шару), а m — кількість вхідних нейронів (нейронів попереднього шару). Для нас $n = 2$ і $m = 4$.

$$w^1 = \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 & w_{14}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 & w_{24}^1 \end{bmatrix}$$

Примітка: перше число в індексі будь-якої ваги відповідає індексу нейрона в наступному шарі (у нашому випадку це *шар Hidden_2*) , а друге число відповідає індексу нейрона в попередньому шарі (у нашому випадку це *вхід шар*).

$\boldsymbol{x}$ – вхідний вектор форми $(m, 1)$, де m — кількість вхідних нейронів. Для нас $m = 4$.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$\boldsymbol{b^1}$- вектор зсуву форми $(n, 1)$, де n — кількість нейронів у поточному шарі. Для нас $n = 2$.

$$b^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \end{bmatrix}$$

Дотримуючись рівняння z^2ми можемо використати наведені вище визначення W^1, x і b^1 для отримання « *рівняння для* z^2» :

$$z^{(2)} = \begin{bmatrix} W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + W_{14}^{(1)}x_4 \\ W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + W_{24}^{(1)}x_4 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

Вихідний шар.

Останньою частиною нейронної мережі є вихідний рівень, який створює прогнозоване значення. У нашому простому прикладі він представлений як один нейрон, пофарбований у **синій колір** і оцінений таким чином: $s = W^3a^3$ Знову ж таки, ми використовуємо матричне представлення, щоб спростити рівняння. Щоб зрозуміти логіку, що лежить в основі, можна використовувати наведені вище прийоми.

Forward propagation.

$$x_i = a^1 \; Input \; layer$$
$$z^2 = W^1x + b^1 \; neuron \; value \; at \; Hidden_1 \; layer$$
$$a^2 = f(z^{(2)}) \; activation \; value \; at \; Hidden_1 \; layer$$
$$z^3 = W^2a^2 + b^2 \; neuron \; value \; at \; Hidden_2 \; layer$$
$$a^3 = f(z^{(3)}) \; activation \; value \; at \; Hidden_2 \; layer$$
$$s = W^3a^3 \; Output \; layer$$

Рис. 2.5.2 Огляд рівнянь прямого поширення

Останнім кроком у прямому проході є оцінка прогнозованого виходу s порівняно з очікуваним виходом y . Вихідні дані y є частиною навчального набору даних (x, y) , де x є вхідними даними (як ми бачили в попередньому

розділі). Оцінка між s та y відбувається через функцію вартості . Це може бути таким простим, як MSE (середня квадратична помилка), або більш складним, як крос-ентропія .

Ми назвемо цю функцію витрат C і позначимо її так: $C = cost(s, y)$,

де вартість може дорівнювати MSE, крос-ентропії або будь-якій іншій функції витрат .

Базуючись на значенні C , модель «знає», наскільки потрібно відкоригувати свої параметри, щоб наблизитися до очікуваного результату y . Це відбувається за допомогою алгоритму зворотного поширення.

Backward propagation.

Зворотне розповсюдження спрямоване на мінімізацію функції витрат шляхом коригування вагових коефіцієнтів мережі та зміщень. Рівень коригування визначається градієнтами функції витрат відносно цих параметрів.

Градієнт функції $C(x_1, x_2, \dots, x_m)$ у точці x є вектором частинних похідних C за x: $\frac{\partial C}{\partial x} = [\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \dots, \frac{\partial C}{\partial x_m}]$.

Похідна функції C вимірює чутливість до зміни значення функції (вихідне значення) відносно зміни її аргументу x (вхідне значення) . Іншими словами, похідна вказує нам напрямок руху C.

Градієнт показує, наскільки потрібно змінити параметр x (у позитивному чи негативному напрямку), щоб мінімізувати C.

Обчислення цих градієнтів відбувається за допомогою техніки, яка називається ланцюговим правилом .

Для однієї ваги $(w_{jk})^l$ градієнт дорівнює:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \qquad chain\ rule$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \qquad by\ definition$$

$$m\ -\ number\ of\ neurons\ in\ l-1\ layer$$

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \qquad by\ differentiation\ (calculating\ derivative)$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \qquad final\ value$$

Подібний набір рівнянь можна застосувати до $\left(b_j\right)^l$:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \qquad chain\ rule$$

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \qquad by\ differentiation\ (calculating\ derivative)$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} 1 \qquad final\ value$$

Спільну частину в обох рівняннях часто називають *«локальним градієнтом»* і виражають таким чином:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \qquad local\ gradient$$

Градієнти дозволяють оптимізувати параметри моделі:

$$while\ (termination\ condition\ not\ met)$$

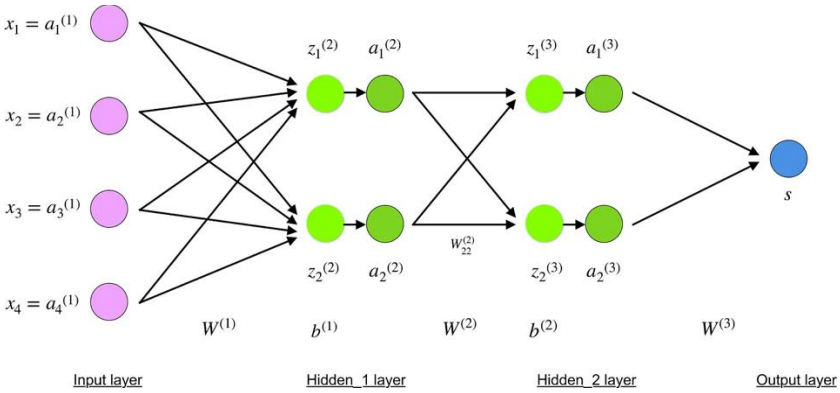
$$w := w - \epsilon \frac{\partial C}{\partial w}$$

$$b := b - \epsilon \frac{\partial C}{\partial b}$$

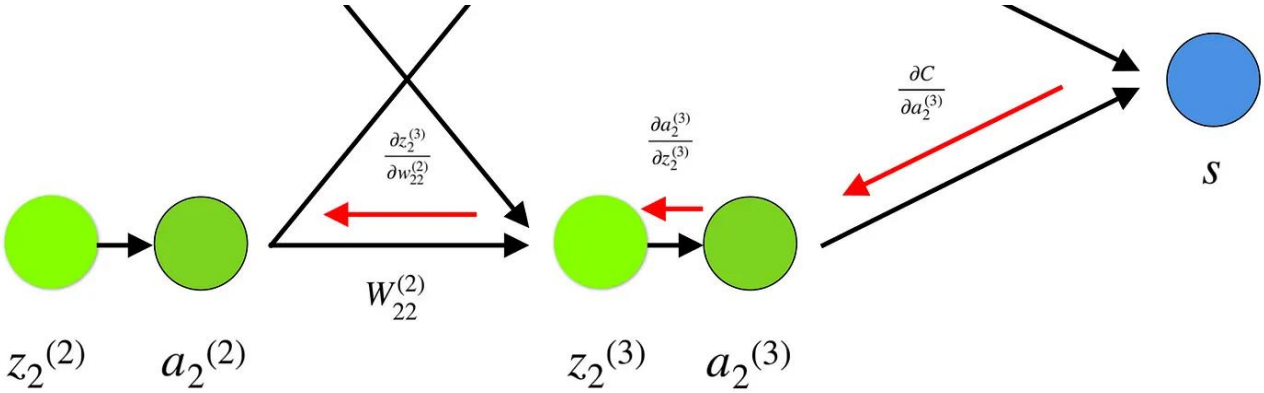
$$end$$

Приклад.

Для прикладу нижче розрахуємо градієнт C відносно однієї ваги - w_{22}^2.



Збільшимо масштаб нижньої частини наведеної вище нейронної мережі:



Вага $(w_{22})^2$ з'єднує $(a_2)^2$ і $(z_2)^2$, тому обчислення градієнта вимагає застосування правила ланцюга через $(a_2)^3$ і $(z_2)^3$:

$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \cdot a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} \cdot f'(z_2^{(3)}) \cdot a_2^{(2)}$$

ЗПрактична частина

3.1. Частина 1

Завдання: розробити програмне забезпечення для реалізації класичного нейрону. Передбачити режим навчання на одному навчальному прикладі та режим розпізнавання.

Код програми знаходиться у ДОДАТКах.

Код протестовано на лекційному прикладі:

$x_0 = 1\ x_1 = 0.2\ x_2 = 0.3\ w_0 = 0.1\ w_1 = 0.5\ w_2 = 0.4\ y = 0.5$

i	x0	x1	x2	y	w0	w1	w2	xs	ym	d	dw0	dw1	dw2
0	1	0,2	0,3	0,5	0,1	0,5	0,4	0,32	0,5793	-0,01933	-0,0193	-0,004	-0,006
1	1	0,2	0,3	0,5	0,08067	0,4961	0,3942	0,2982	0,574	-0,01809	-0,0181	-0,004	-0,005
…	…	…	…	…	…	…	…	…	…	…	…	…	…
25	1	0,2	0,3	0,5	-0,13695	0,4526	0,3289	0,0522	0,5131	-0,00326	-0,0033	-7E-04	-1E-03

Лекційний приклад 3.1.

```
Iteration: 0
Σ 0.32
Output 0.5793242521487495
Const: x0=1.0, x1=0.2, x2=0.3, y(target)=0.5
δ: -0.01933192811206358
dw0 -0.01933192811206358, dw1 -0.003866385622412716, dw2 -0.005795784336190735
upd_w0 0.08066807188793643, upd_w1 0.4961336143775873, upd_w2 0.39420042156638097
==================
Iteration: 1
Σ 0.2981549212333682
Output 0.5739914101388652
Const: x0=1.0, x1=0.2, x2=0.3, y(target)=0.5
δ: -0.018092769632575233
dw0 -0.018092769632575233, dw1 -0.003618553926515047, dw2 -0.00542783088977257
upd_w0 0.06257530225536119, upd_w1 0.49251506045107224, upd_w2 0.3887725906766084
==================
```

Скріншот роботи програми (Ітерації 0, 1) 3.1.

```
=================
Iteration: 25
Σ 0.052241449425411535
Output 0.5130573928387344
Const: x0=1.0, x1=0.2, x2=0.3, y(target)=0.5
δ: -0.0032621219828617375
dw0 -0.0032621219828617375, dw1 -0.0006524243965723475, dw2 -0.0009786365948585211
upd_w0 -0.1402165915178958, upd_w1 0.4519566816964208, upd_w2 0.32793502254463136
=================
Target Output: 0.5
Neuron Output after training: 0.5130573928387344
```

Скріншот роботи програми (Ітерація 25 (остання)) 3.2.

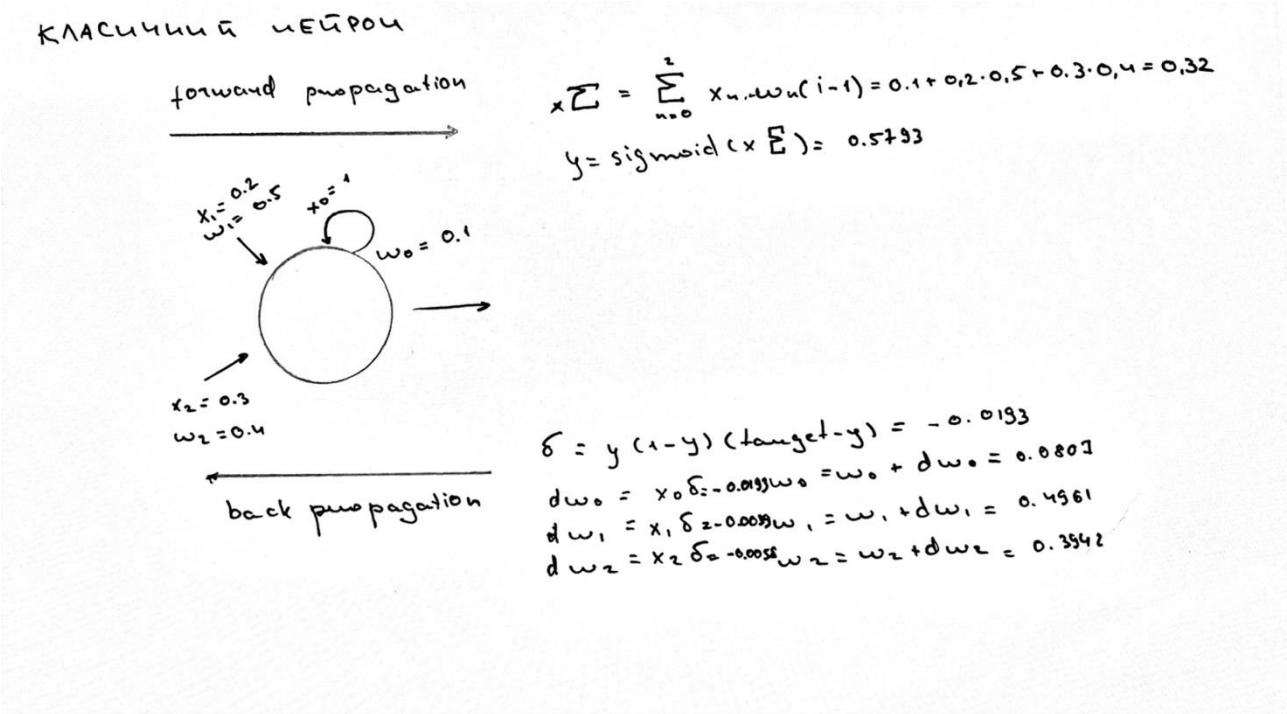


Схема роботи 3.1.

3.2. Частина 2

Завдання: розробити програмне забезпечення для реалізації елементарного двошарового персептрону із структурою 1-1-1. Передбачити режим навчання на одному навчальному прикладі та режим розпізнавання.

Код програми знаходиться у ДОДАТКах.

Код протестовано на лекційному прикладі:

$X = 1\ w_{12} = 0.5\ w_{23} = 0.4\ \ y = 0.5$

i	X	Y	w12	w23	xs2	y2	xs3	y3	d3	dw23	d2	dw12
0	1	0.5	0.5	0.4								
1	1	0.5	0.498209	0.390511	0.5	0.622459	0.248984	0.561926	-0.01524	-0.00949	-0.00179	-0.00179
...	...	...	...	...	...	...	...	...	...	...	...	...
20	1	0.5	0.471838	0.247948	0.472922	0.616075	0.156436	0.539029	-0.0097	-0.00597	-0.00108	-0.00108

Лекційний приклад 3.2.

```
Iteration: 1
X: 1, w1,2: 0.5
Σ1,2: 0.5
y2: 0.6224593312018546
Σ2,3: 0.24898373248074185
Output: 0.5619263470269478
δ3: -0.015244107114216289
dw23: -0.009488836719084504
w23_upd: 0.3905111632809155
δ2: -0.0014329687044158255
dw12: [-0.00143297]
w12_upd: 0.49856703129558416
```

Скріншот роботи програми (Ітерація 1) 3.3.

```
==================
Iteration: 20
X: 1, w1,2: 0.48161983021367294
Σ1,2: 0.48161983021367294
y2: 0.6181303010618353
Σ2,3: 0.15674519981876742
Output: 0.5391062657135878
δ3: -0.009716761215534827
dw23: -0.006006224535504507
w23_upd: 0.2475733193721598
δ2: -0.0005816088133462183
dw12: [-0.00058161]
w12_upd: 0.4810382214003267
==================
```

Скріншот роботи програми (Ітерація 20) 3.4.

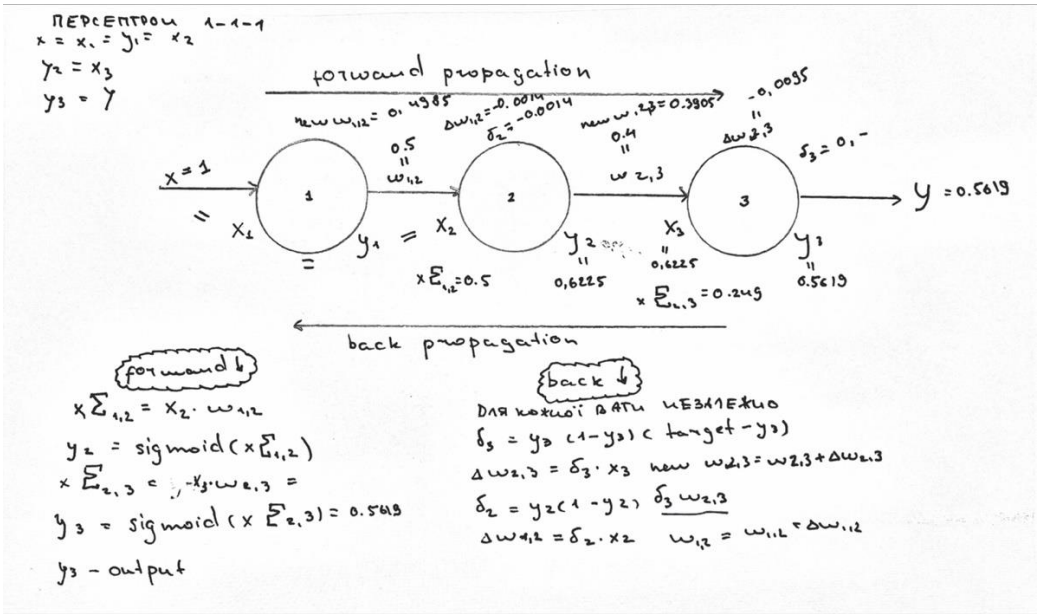


Схема роботи 3.2.

3.3. Частина 3

Завдання: розробити програмне забезпечення для реалізації двошарового персептрону із структурою 2-3-1. Передбачити режим навчання «ON-LINE» та режим розпізнавання.

Піддослідна функція $x_1 + x_2 = y$

Дане завдання було розділено на 2 частини:

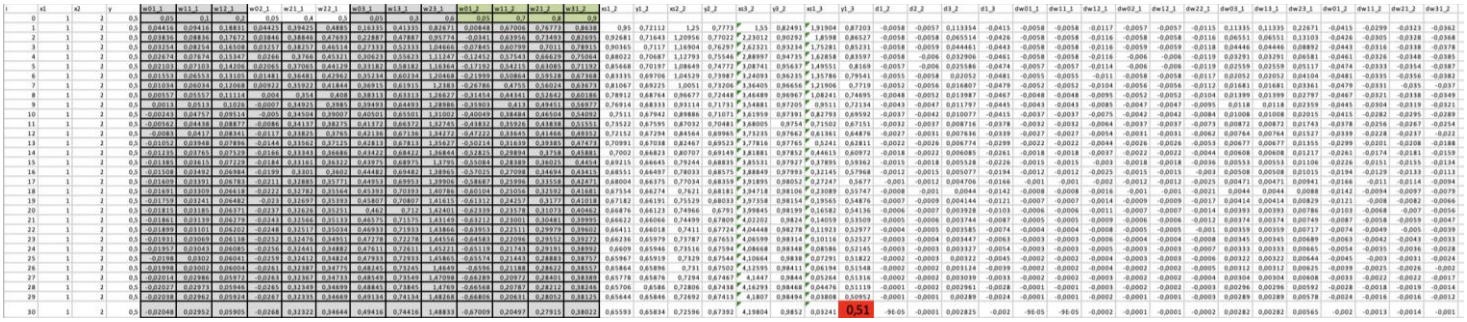
- Розробка програмного забезпечення для реалізації двошарового персептрону із структурою 2-3-1 на одному прикладі;
- Імплементація режиму «ON-LINE» та піддослідної функції.

3.3.1. Розробка програмного забезпечення для реалізації двошарового персептрону із структурою 2-3-1 на одному прикладі.

<u>Код програми знаходиться у ДОДАТКах.</u>

Код протестовано на лекційному прикладі:

$$x_0 = 1\ x_1 = 0.2\ x_2 = 0.3\ w_0 = 0.1\ w_1 = 0.5\ w_2 = 0.4\ y = 0.5$$



Лекційний приклад 3.3.

```
Input data: x = [[1 1 2]], y = [0.5]
w01_1 = 0.05, w11_1 = 0.1, w21_1 = 0.4
w02_1 = 0.05, w12_1 = 0.2, w22_1 = 0.5
w03_1 = 0.05, w13_1 = 0.3, w23_1 = 0.6
w01_2 = 0.05, w11_2 = 0.7, w21_2 = 0.8, w31_2 = 0.9
xs1_2 = 0.9500000000000001, y1_2 = 0.7211151780228631
xs2_2 = 1.25, y2_2 = 0.7772998611746911
xs3_2 = 1.5499999999999998, y3_2 = 0.8249137318359602
xs1_3 = [1.         0.72111518 0.77729986 0.82491373], y1_3 = 0.8720316634810029
...performing backward propagation...
Updated values
w01_1 = 0.044155568959906896, w11_1 = 0.0941555689599069, w21_1 = 0.3883111379198138
w02_1 = 0.04425071618855552, w12_1 = 0.19425071618855552, w22_1 = 0.488501432377111
w03_1 = 0.04460343003710246, w13_1 = 0.29460343003710243, w23_1 = 0.5892068600742049
w01_2 = 0.008484078406121191, w11_2 = 0.6700622388090468, w21_2 = 0.7677296799085387, w31_2 = 0.8657529461873843
```

Скріншот роботи програми (Ітерація 0) 3.5.

```
xs1_2 = 0.5291806029204877, y1_2 = 0.6292919803395746
xs2_2 = 0.8157952523839301, y2_2 = 0.693343060656904
xs3_2 = 1.126729233288876, y3_2 = 0.7552347857071744
xs1_3 = [1.         0.62929198 0.69334306 0.75523479], y1_3 = 0.5050809143493188
...performing backward propagation...
Updated values
w01_1 = -0.02020963834568948, w11_1 = 0.029790361654310534, w21_1 = 0.25958072330862103
w02_1 = -0.022449858354508644, w12_1 = 0.1275501416454913, w22_1 = 0.3551002832909826
w03_1 = -0.020631426358667892, w13_1 = 0.22936857364133217, w23_1 = 0.45873714728266435
w01_2 = -0.6253039031923724, w11_2 = 0.2458221426135721, w21_2 = 0.3042539567778812, w31_2 = 0.3666064136799477
```

Скріншот роботи програми (Ітерація 30) 3.6.

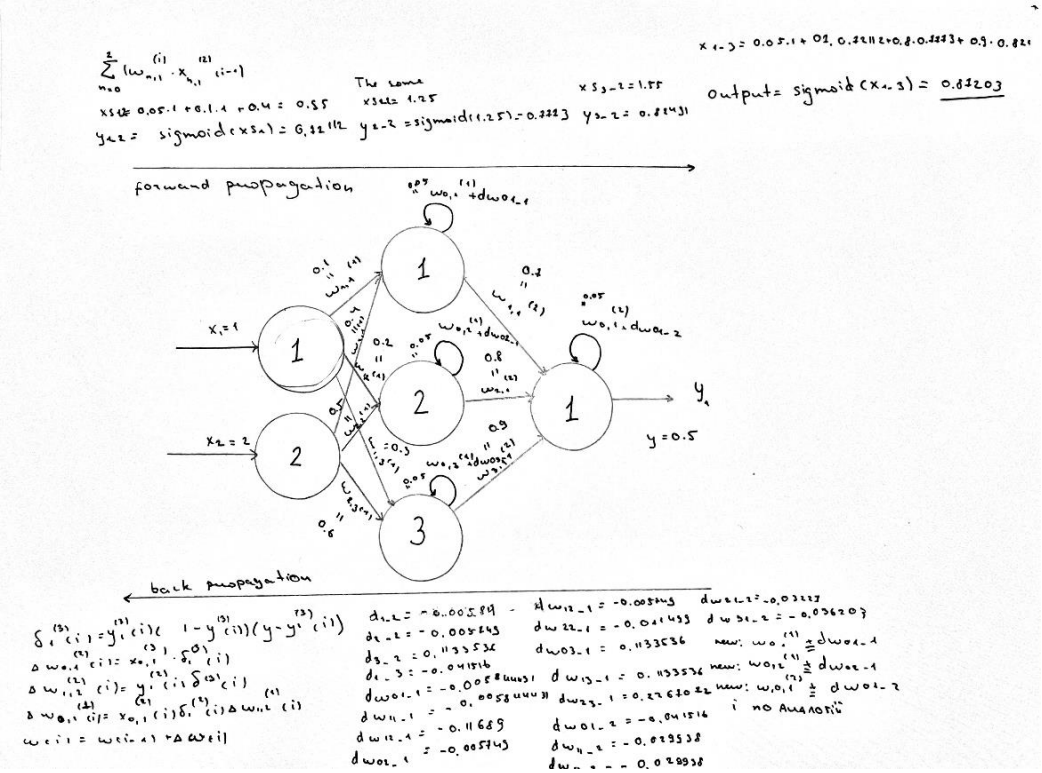
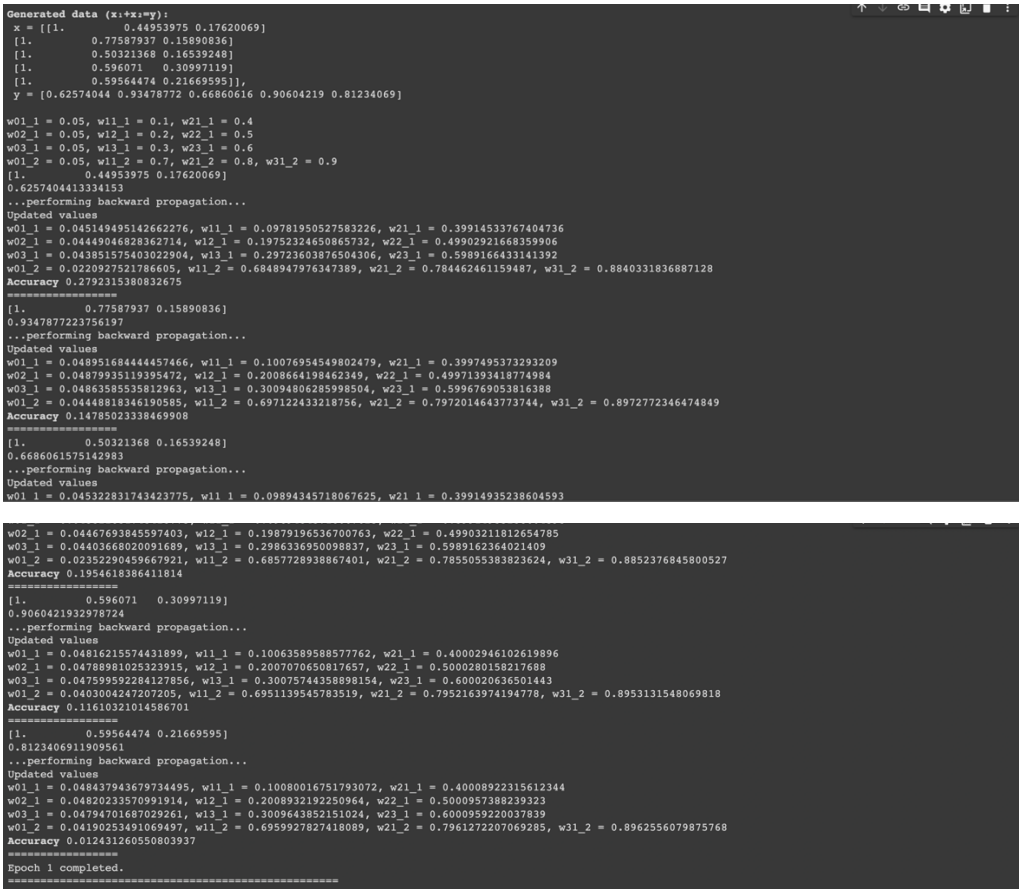


Схема роботи 3.3.

3.3.2. Імплементація режиму «ON-LINE» та піддослідної функції.

```
Generated data (x₁+x₂=y):
 x = [[1.         0.44953975 0.17620069]
 [1.         0.77587937 0.15890836]
 [1.         0.50321368 0.16539248]
 [1.         0.596071   0.30997119]
 [1.         0.59564474 0.21669595]],
 y = [0.62574044 0.93478772 0.66860616 0.90604219 0.81234069]
```

Скріншот роботи програми (Введення піддослідної функції (5 семплів)) 3.7.



Скріншот роботи програми (Виконання однієї епохи) 3.8.

Скріншот роботи програми

(Тестування на прикладі $x_0 = 1\ x_1 = 0.1\ x_2 = 0.3\ y = 0.1 + 0.3 = 0.4$) 3.9.

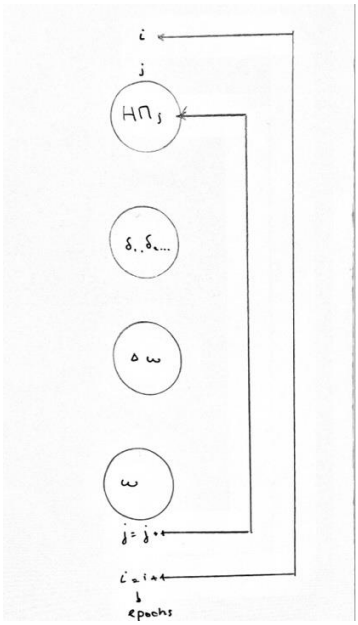


Схема роботи 3.4.

ВИСНОВКИ

У цій лабораторній роботі ми створили та дослідили різні типи нейронних мереж. Починаючи з класичного нейрону, ми розвинулися до двошарових персептронів різної структури.

У першій частині роботи ми розробили класичний нейрон, навчили його на одному прикладі і перевірили режим розпізнавання.

У другій частині створили елементарний двошаровий персептрон та навчили його на одному навчальному прикладі.

У третій частині реалізували більш складний двошаровий персептрон із структурою 2-3-1 та ввели режим навчання "ON-LINE", використовуючи піддослідну функцію.

Ця робота допомогла нам зрозуміти принципи роботи нейронних мереж та їхні можливості у навчанні та розпізнаванні образів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Навч. посібник «Штучні нейронні мережі» Руденко О.Г., Бодянський Є.В. – стор. 25-34, 61-66, 83-100.

ДОДАТКИ

Додаток А

https://colab.research.google.com/drive/1t75jKUQhEdVPLyAI1VUr6-z7jBIztti5?usp=sharing#scrollTo=6lGjUlAI3hv9