



Навчання з підкріпленням

Лекція 1: Вступ

Кочура Юрій Петрович
iuriy.kochura@gmail.com
[@y_kochura](https://twitter.com/y_kochura)

Сьогодні

- Огляд основ машинного навчання
- Побудова нейронних мереж

Огляд основ машинного навчання

Що таке машинне навчання?

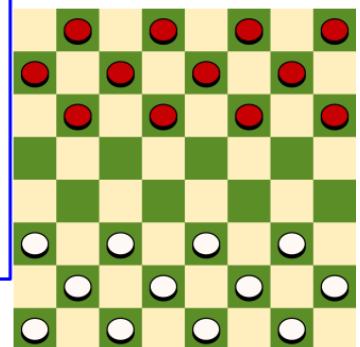
Визначення за Артур Семюель

Артур Семюель (1959): Машинне навчання - це область навчання, яка надає комп'ютеру можливість вчитися не будучи явно запrogramованим.



A. L. Samuel*

**Some Studies in Machine Learning
Using the Game of Checkers. II—Recent Progress**



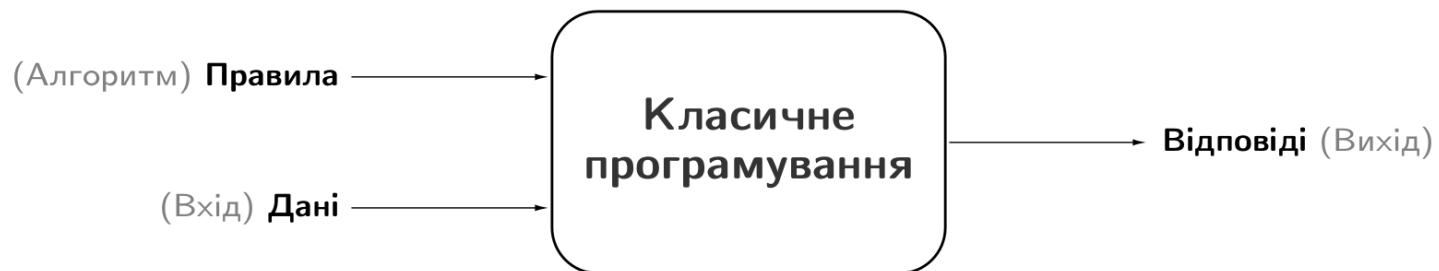
Визначення за Том Мітчелл

Том Мітчелл (1998): Комп'ютерна програма, яка учається з досвіду E по відношенню до деякого класу задач T та міри продуктивності P називається машинним навчанням, якщо її продуктивність у задачах з T , що вимірюється за допомогою P , покращується з досвідом E .

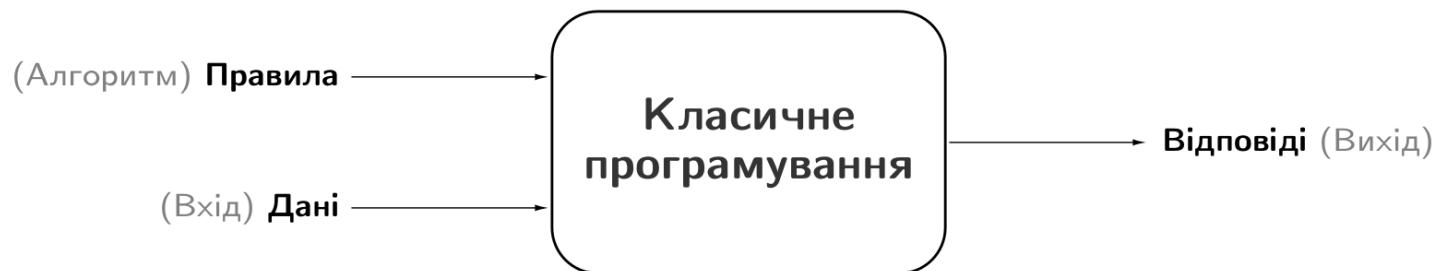


- Досвід (дані): ігри в які грає програма сама з собою
- Вимір продуктивності: коефіцієнт виграшу

Класичне програмуванн **vs** машинне навчання

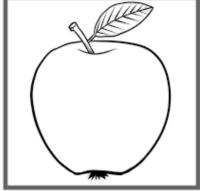


Класичне програмування vs машинне навчання



ТИПИ НАВЧАННЯ

За характером навчальних даних (**досвіду**) машинне навчання поділяють на чотири типи: контролльоване (з учителем), напівконтрольоване, неконтрольоване (без учителя) та з підкріпленням.

Контрольоване навчання Supervised learning	Напівконтрольоване навчання Semi-supervised learning	Неконтрольоване навчання Unsupervised learning	Навчання з підкріплення Reinforcement learning
Дани: (x, y) x – приклад, y – мітка			
Мета – знайти функцію відображення $x \rightarrow y$			
Приклад  Це є яблуко.			

Типи навчання

Контрольоване навчання Supervised learning	Напівконтрольоване навчання Semi-supervised learning	Неконтрольоване навчання Unsupervised learning	Навчання з підкріплення Reinforcement learning
Дани: (x, y) x – приклад, y – мітка	Дани: (x, y) та $x, (x, y) < x $ x – приклад, y – мітка		
Мета – знайти функцію відображення $x \rightarrow y$	Мета – знайти функцію відображення або категорію $x \rightarrow y$		
Приклад  Це є яблуко.	Приклад  Це є яблуко.		

Типи навчання

Контрольоване навчання Supervised learning	Напівконтрольоване навчання Semi-supervised learning	Неконтрольоване навчання Unsupervised learning	Навчання з підкріплення Reinforcement learning
<p>Дани: (x, y) x – приклад, y – мітка</p>	<p>Дани: (x, y) та $x, (x, y) < x$ x – приклад, y – мітка</p>	<p>Дани: x x – приклад, немає міток!</p>	
<p>Мета – знайти функцію відображення $x \rightarrow y$</p>	<p>Мета – знайти функцію відображення або категорію $x \rightarrow y$</p>	<p>Мета – знайти правильну категорію.</p>	
<p>Приклад</p>  <p>Це є яблуко.</p>	<p>Приклад</p>  <p>Це є яблуко.</p>	<p>Приклад</p>  <p>Цей об'єкт схожий на інший.</p>	

Типи навчання

Контрольоване навчання Supervised learning	Напівконтрольоване навчання Semi-supervised learning	Неконтрольоване навчання Unsupervised learning	Навчання з підкріплення Reinforcement learning
<p>Дані: (x, y) x – приклад, y – мітка</p>	<p>Дані: (x, y) та $x, (x, y) < x$ x – приклад, y – мітка</p>	<p>Дані: x x – приклад, немає міток!</p>	<p>Дані: пари стан-дія</p>
<p>Мета – знайти функцію відображення $x \rightarrow y$</p>	<p>Мета – знайти функцію відображення або категорію $x \rightarrow y$</p>	<p>Мета – знайти правильну категорію.</p>	<p>Мета – максимізація загальної винагороди, отриманої агентом при взаємодії з навколошнім середовищем.</p>
<p>Приклад</p>  <p>Це є яблуко.</p>	<p>Приклад</p>  <p>Це є яблуко.</p>	<p>Приклад</p>  <p>Цей об'єкт схожий на інший.</p>	<p>Приклад</p>  <p>Їжте це, бо це зробить вас сильнішим.</p>

Як вчиться людина?

- Ми та інші розумні істоти, вчимось завдяки **взаємодії** із своїм оточенням
- Взаємодії часто бувають **послідовними** - майбутні взаємодії можуть залежати від попередніх
- Ми направлені на **результат**
- Ми можемо вчитися **не маючи прикладів** оптимальної поведінки

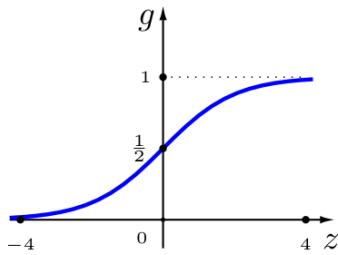
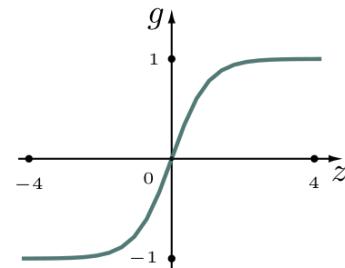
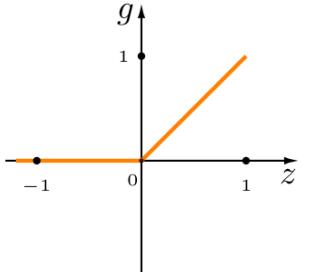
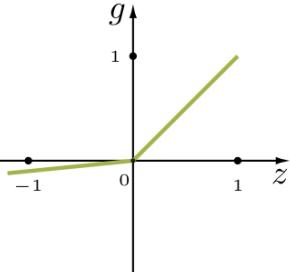
Мозок людини

Базовою обчислювальною одиницею мозку є нейрон. Мозок дорослої людини складається з 86 мільярдів нейронів, які з'єднані між собою приблизно 10^{14} – 10^{15} синапсами.

Біологічний та штучний нейрон

Біологічний нейрон	Штучний нейрон
<p>Дендрити Ядро Тіло клітини Аксон Відростки аксону Термінали аксону Імпульс спрямований до тіла клітини Імпульс спрямований від тіла клітини</p>	<p>Імпульс від аксону (вхід) x_0 Синапс Дендрит ω_0 $b = \omega_0 x_0$ x_0 x_1 x_2 \vdots x_m ω_1 ω_2 ω_m $\omega_1 x_1$ $\omega_2 x_2$ $\omega_m x_m$ $z = \sum_{n=1}^m \omega_n x_n + b$ Тіло клітини Активаційна функція g $g(z)$ Імпульс на аксоні (вихід)</p>

Деякі функції активації

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ $\epsilon \ll 1$
			

Людина добре сприймати
візуальну інформацію



Що Ви бачите?



Собака-вівця чи швабра?

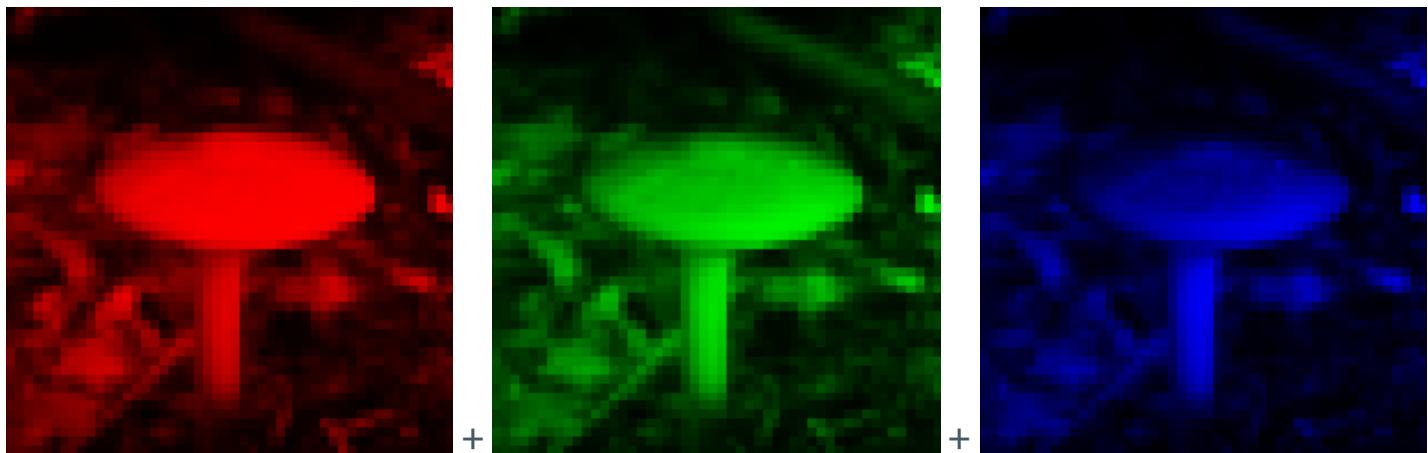
Людський мозок настільки добре інтерпретує візуальну інформацію, що **роздріб** між зображенням та його семантичною інтерпретацією (пікселями) важко оцінити інтуїтивно:



Це мухомор.



Це мухомор.



Це мухомор.

```
array([[[0.03921569, 0.03529412, 0.02352941, 1.          ],
       [0.2509804 , 0.1882353 , 0.20392157, 1.          ],
       [0.4117647 , 0.34117648, 0.37254903, 1.          ],
       ...,
       [0.20392157, 0.23529412, 0.17254902, 1.          ],
       [0.16470589, 0.18039216, 0.12156863, 1.          ],
       [0.18039216, 0.18039216, 0.14117648, 1.          ]],

      [[0.1254902 , 0.11372549, 0.09411765, 1.          ],
       [0.2901961 , 0.2509804 , 0.24705882, 1.          ],
       [0.21176471, 0.2        , 0.20392157, 1.          ],
       ...,
       [0.1764706 , 0.24705882, 0.12156863, 1.          ],
       [0.10980392, 0.15686275, 0.07843138, 1.          ],
       [0.16470589, 0.20784314, 0.11764706, 1.          ]],

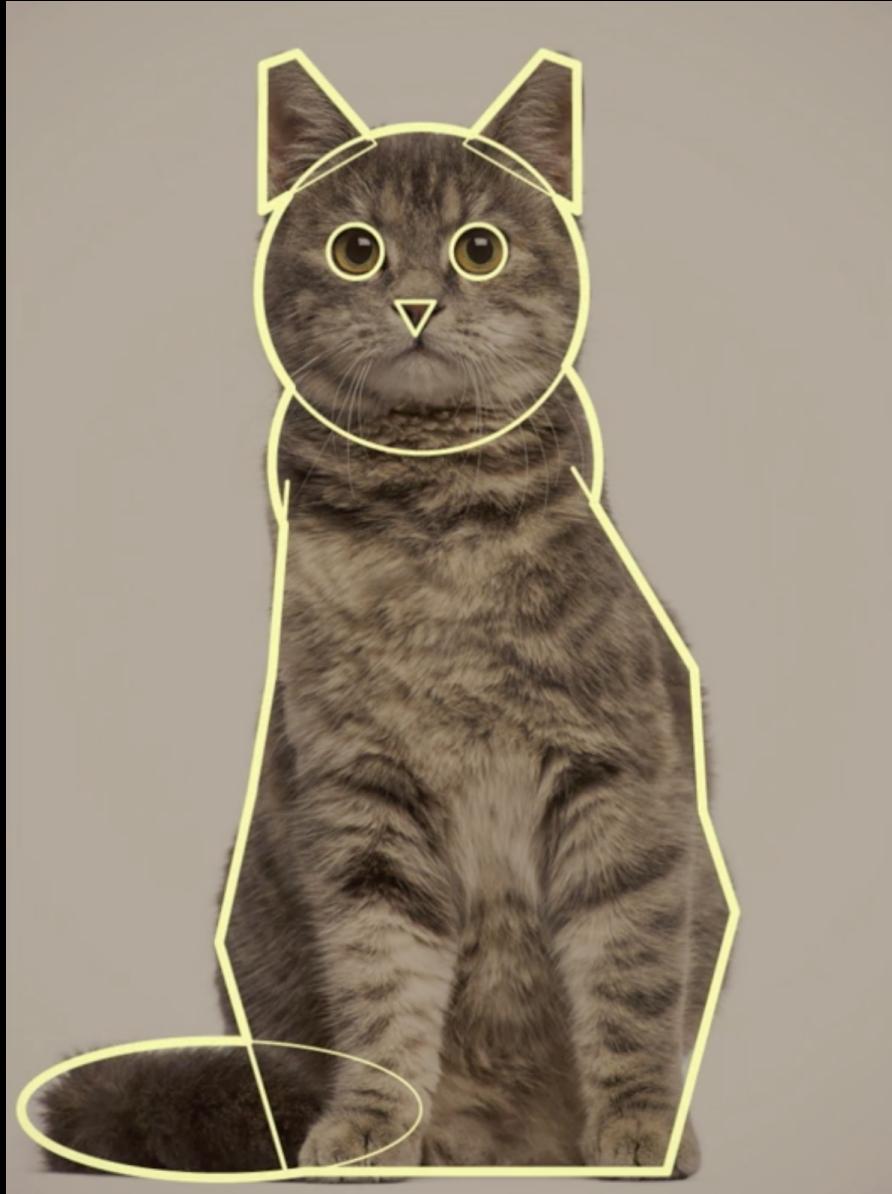
      [[0.14117648, 0.12941177, 0.10980392, 1.          ],
       [0.21176471, 0.1882353 , 0.16862746, 1.          ],
       [0.14117648, 0.13725491, 0.12941177, 1.          ],
       ...,
       [0.10980392, 0.15686275, 0.08627451, 1.          ],
       [0.0627451 , 0.08235294, 0.05098039, 1.          ],
       [0.14117648, 0.2        , 0.09803922, 1.          ]],

      ...,
```

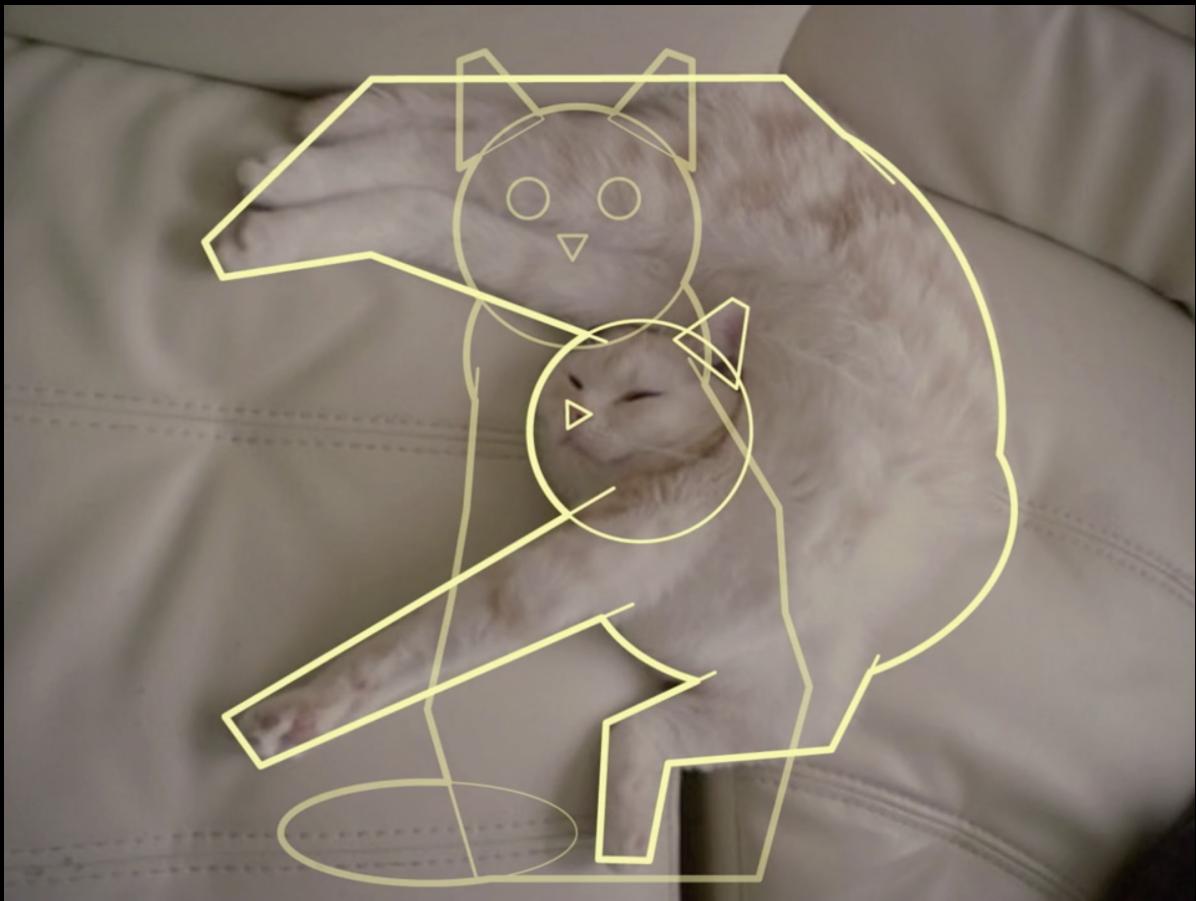
Це мухомор.

Як навчити машин бачити?



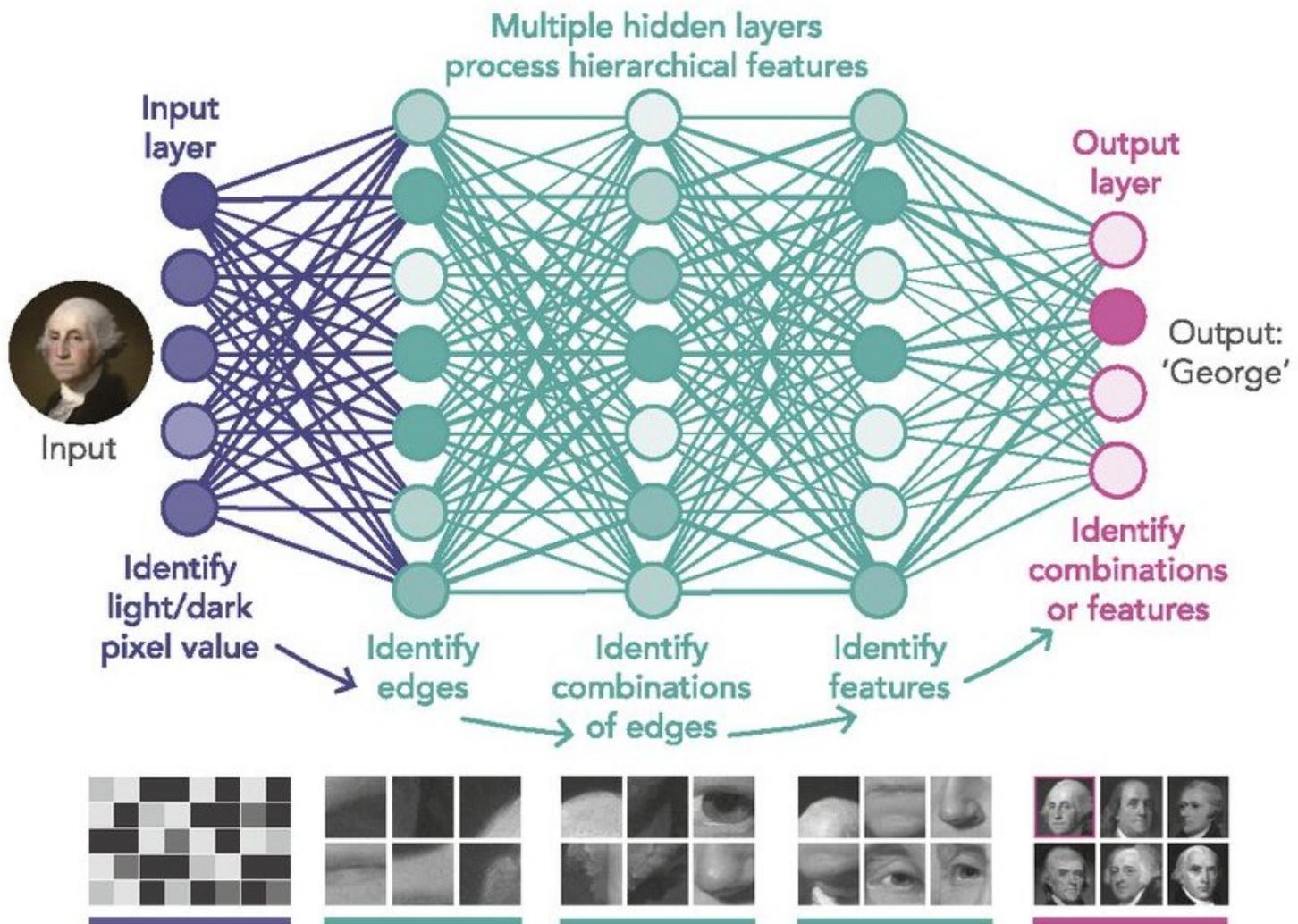






Для пошуку шаблону в даних (витягування семантичної інформації, ознак) потрібна побудова **складних моделей**, які б отримати вручну було б дуже складно.

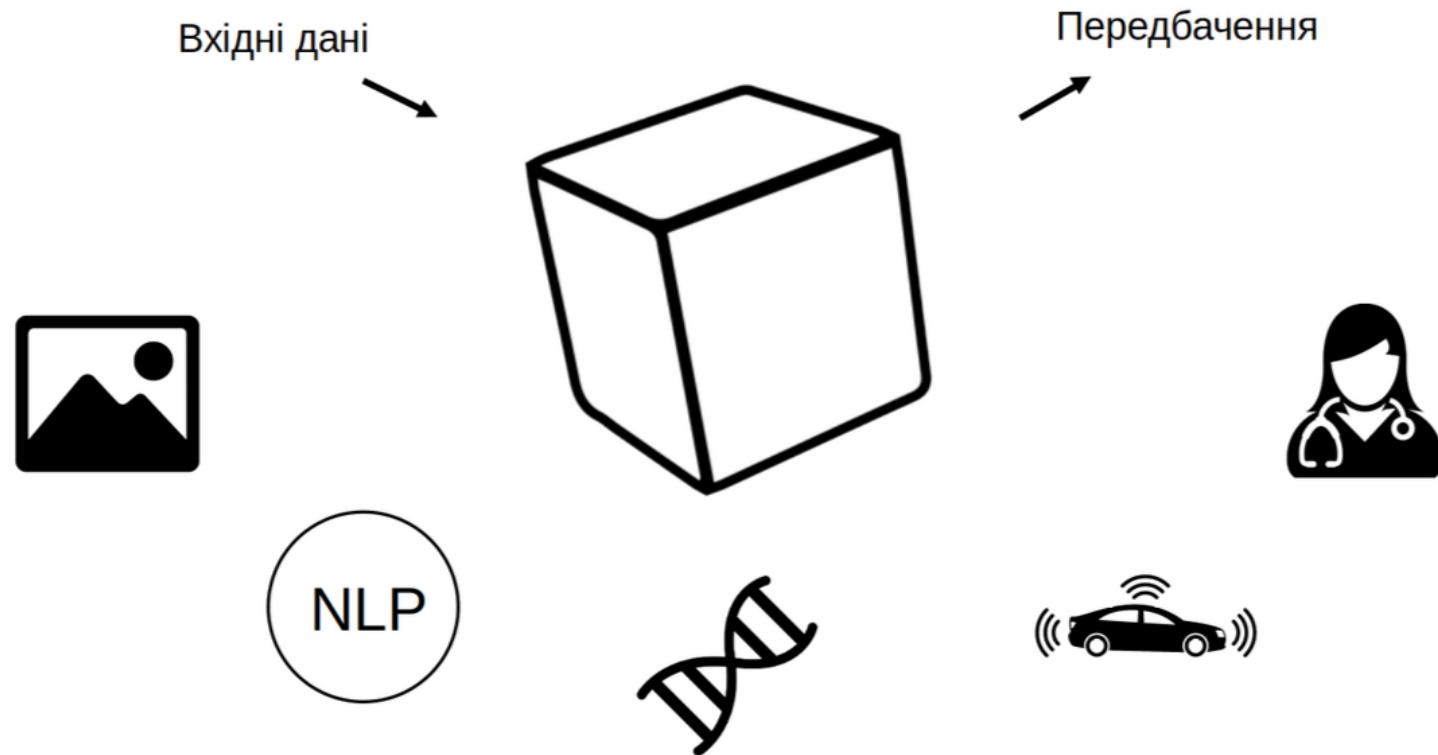
Однак, можна написати програму, яка буде **вчитись** знаходити шаблон в даних самостійно.



Що входить до задачі машинного навчання?

- Постановка проблеми + дані
- Навчання моделі
- Визначення функції втрат
- Вибір алгоритму оптимізації

Які дані використовуються?



Ознаки у машинному навчанні

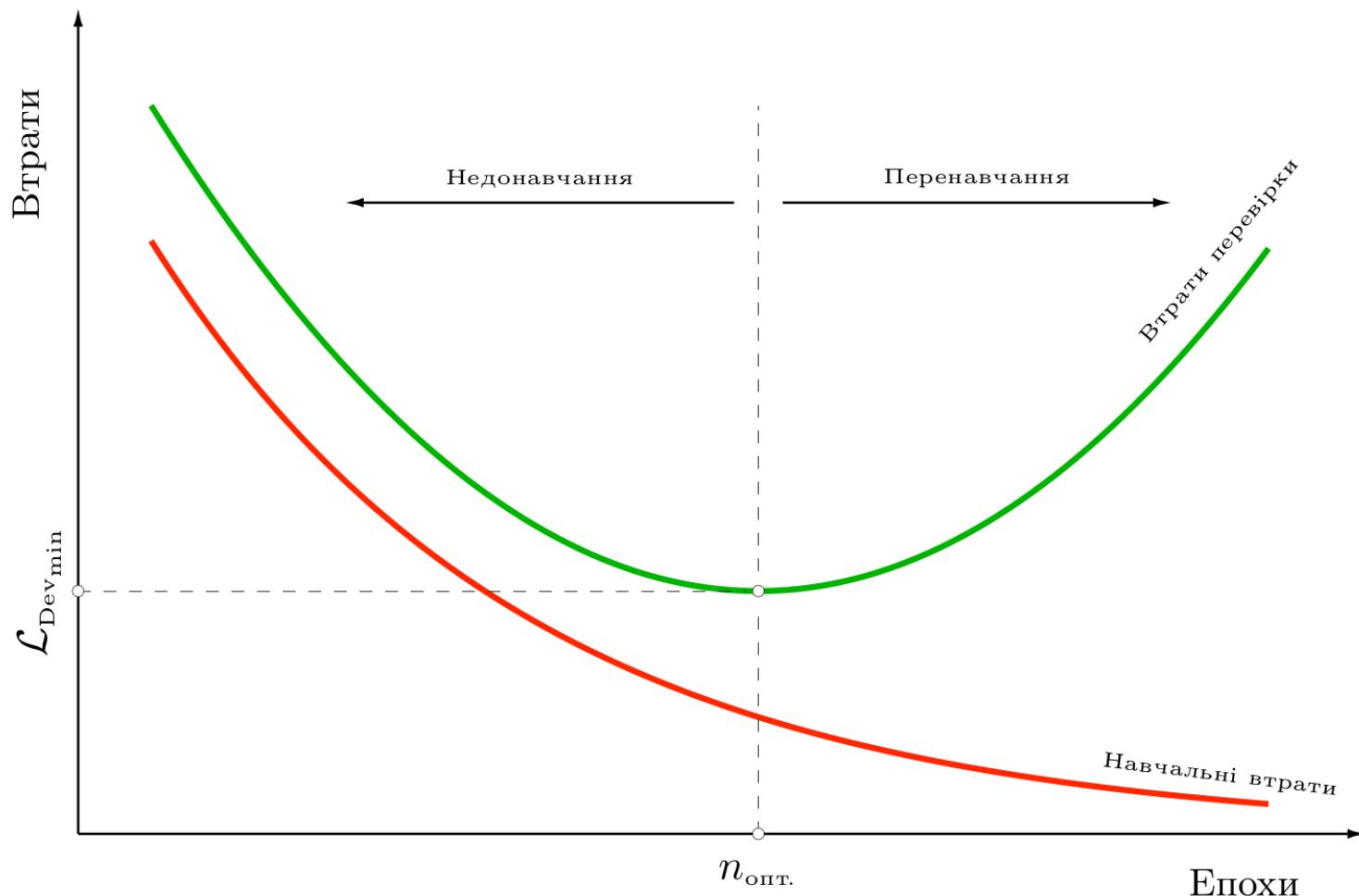
Ознаки - це спостереження, які використовуються для прийняття рішень моделлю.

- Для класифікації зображень **кожен** піксель є ознакою
- Для розпізнавання голосу, **частота** та **гучність** є ознаками
- Для безпілотних автомобілів дані з **камер, радарів і GPS** є ознаками

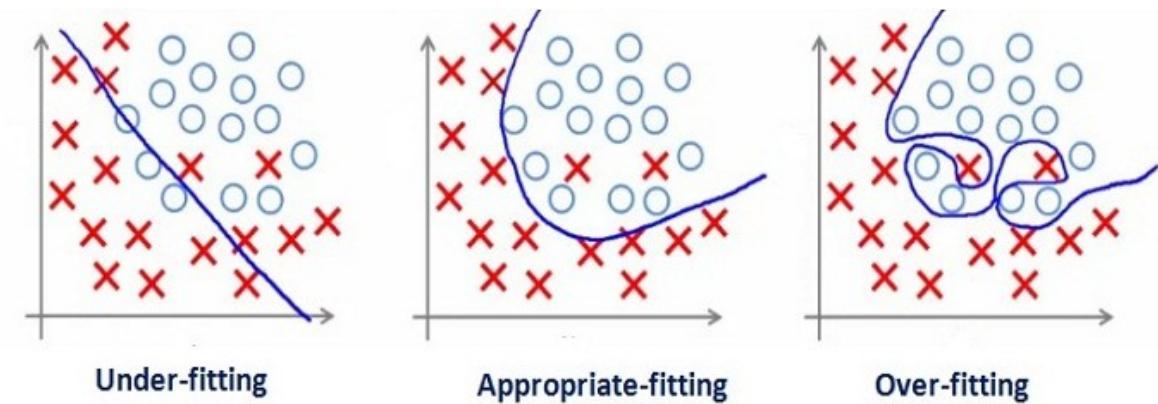
Типи ознак у робототехніці

- Пікселі (RGB дані)
- Глибина (сонар, лазерні далекоміри)
- Орієнтація або прискорення (гіроскоп, акселерометр, компас)

Недонавчання vs перенавчання

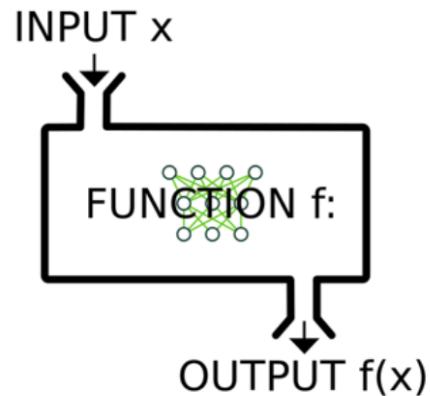


Недонавчання vs перенавчання



Що таке модель?

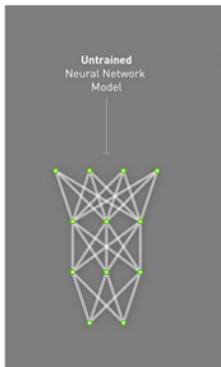
Хоча те, що знаходиться всередині глибинної нейронної мережі, може бути складним, за своєю суттю це просто функції. Вони беруть певні вхідні дані: **INPUT x** і генерують деякі вихідні дані: **OUTPUT f(x)**



З чого складається модель?

Компоненти моделі

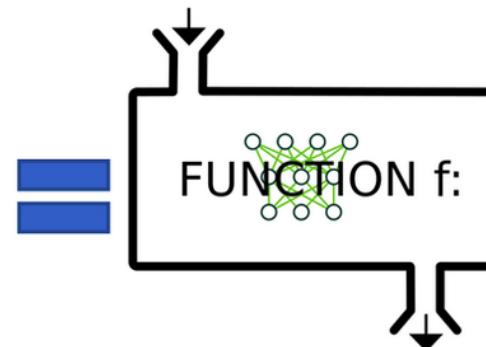
Архітектура мережі = [deploy.prototxt](#)



Навченні ваги = ***.caffemodel



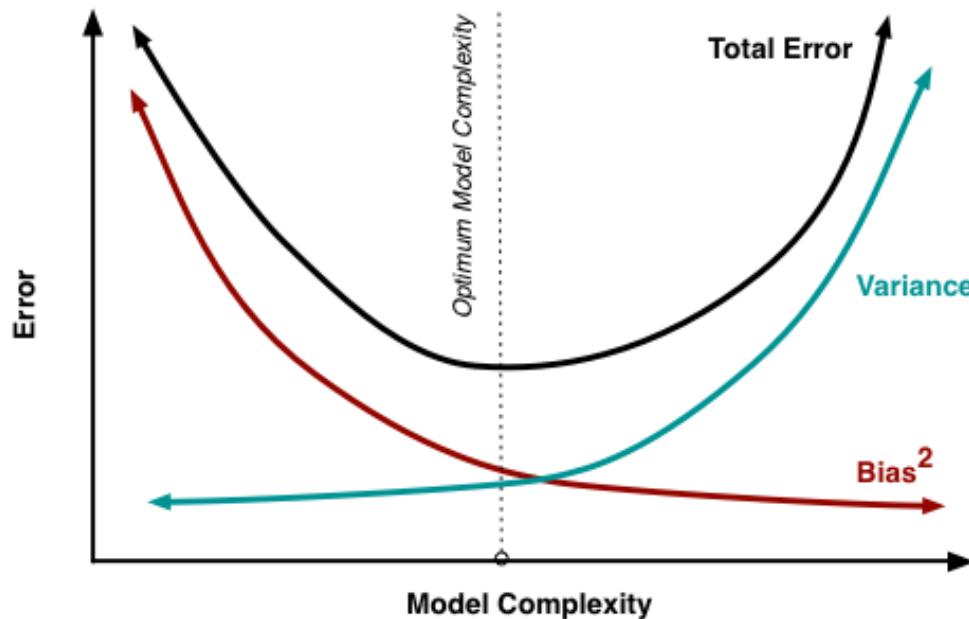
Модель



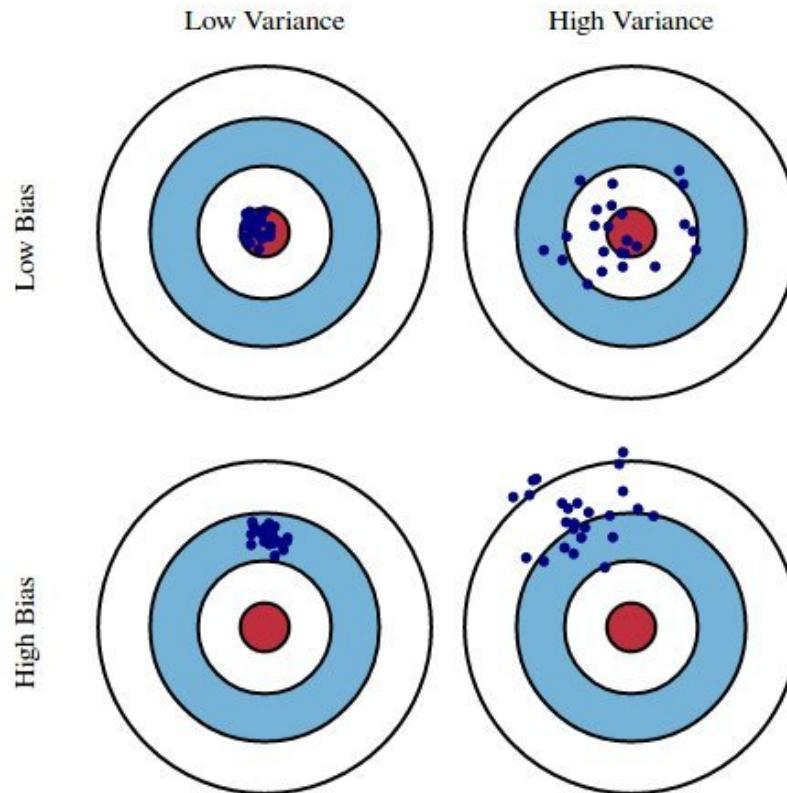
Джерела помилок моделі

- Зсув (Bias)
- Розкид (Variance)
- Шум (Irreducible error)

$$Err = Bias^2 + Variance + Irreducible\ error$$



Інтуїція



Області застосування та успіхи ШІ



Detectron2: A PyTorch-based modular object detection ...



Copy link



Object detection, pose estimation, segmentation (2019)



Google DeepMind's Deep Q-learning playing Atari Break...



Share



Reinforcement learning (Mnih et al, 2014)



AlphaStar Agent Visualisation



Share



Strategy games (Deepmind, 2016-2018)



NVIDIA Autonomous Car



Share



Autonomous cars (NVIDIA, 2016)



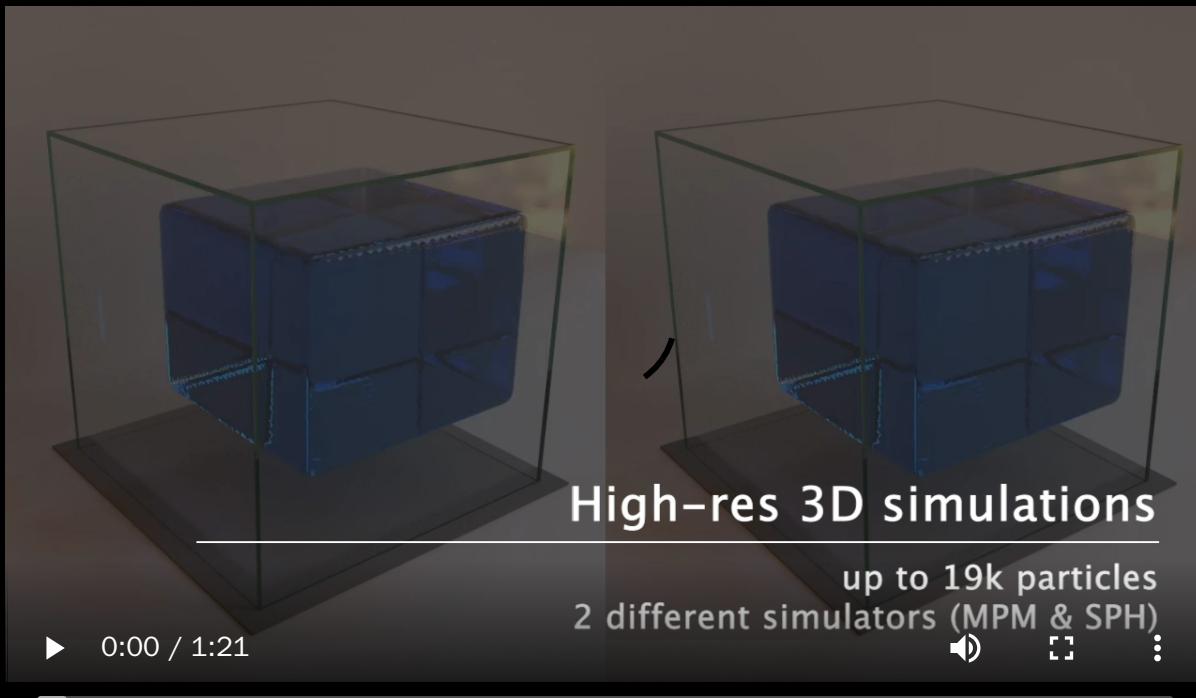
Full Self-Driving



Share



Autopilot (Tesla, 2019)



Physics simulation (Sanchez-Gonzalez et al, 2020)



AlphaFold: The making of a scientific breakthrough



Share



AI for Science (Deepmind, AlphaFold, 2020)



Google Assistant will soon be able to call restaurants an...



Share



Speech synthesis and question answering (Google, 2018)



Artistic style transfer for videos



Share

Sintel movie, III



Artistic style transfer (Ruder et al, 2016)

T

A Style-Based Generator Architecture for Generative Ad...



Share



Image generation (Karras et al, 2018)



GTC Japan 2017 Part 9: AI Creates Original Music



Share



Music composition (NVIDIA, 2017)



Behind the Scenes: Dalí Lives



Share



Dali Lives (2019)



Reface 2жyViV V0д2м0киЙсі к0мур. ли д2 D нI в ипV.



Share

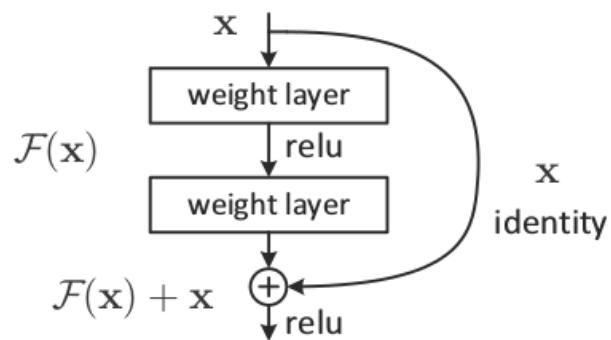
Reface оживив відомі київські мурали до Дня Києва (2021)



*Асоціацією обчислювальної техніки (ACM) нагороджено в 2018 році премією Тюрінга таких науковців: **Yann LeCun, Geoffrey Hinton, Yoshua Bengio** за концептуальні та інженерні прориви, які зробили в глибинних нейронних мережах.*

Чому DL працює?

Алгоритми (старі та нові)



Зростає кількість даних



Програмне забезпечення



Більш швидкі обчислювальні машини





"For the last forty years we have programmed computers; for the next forty years we will train them."

Chris Bishop, 2020.

Виклики ШІ

Основним викликом штучного інтелекту та машинного навчання є прийняття правильних рішень в умовах **невизначеності**

Перцептрон

Одношарова нейронна мережа

Перцептрон vs Логістична регресія

Перцептрон

Перцептрон (Rosenblatt, 1958)

$$g(z) = \begin{cases} 1 & \text{if } z = \sum_i w_i x_i + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Ця модель спочатку була мотивована біологією, де w_i — це синаптичні ваги для вхідних сигналів x_i та g активації.

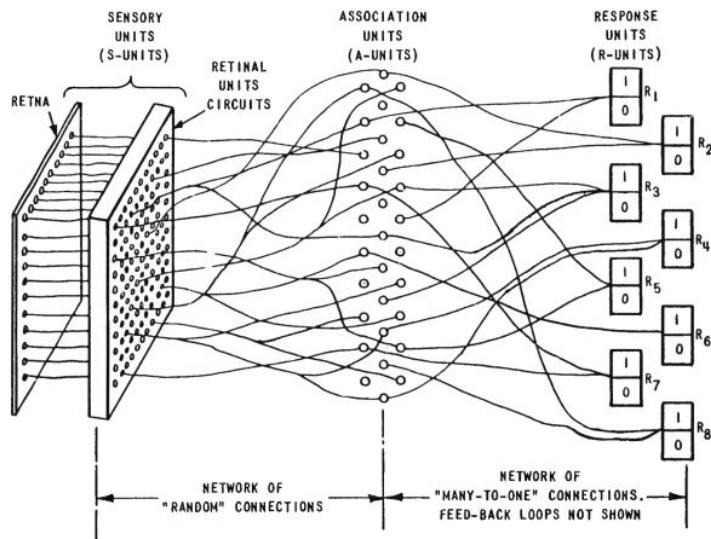
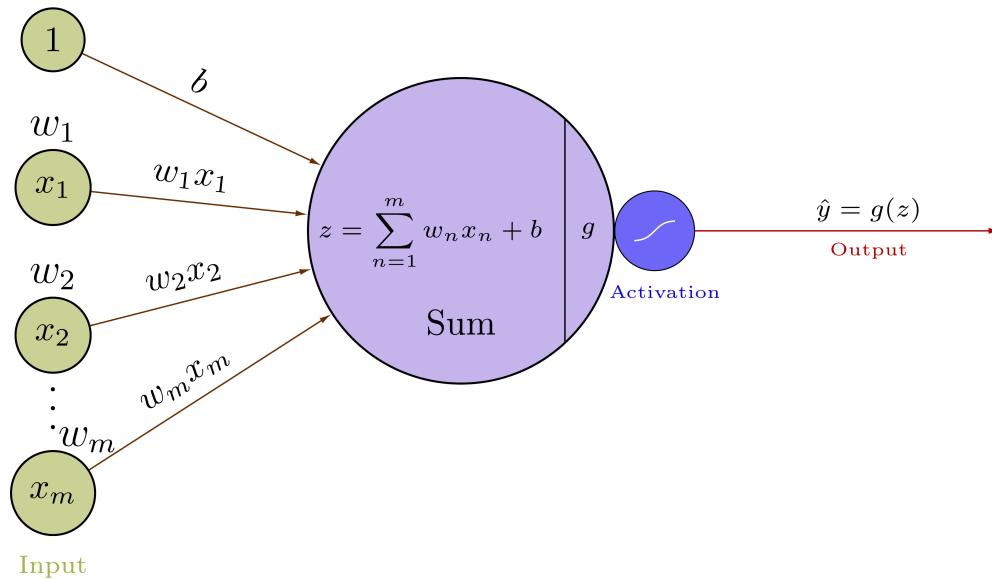


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

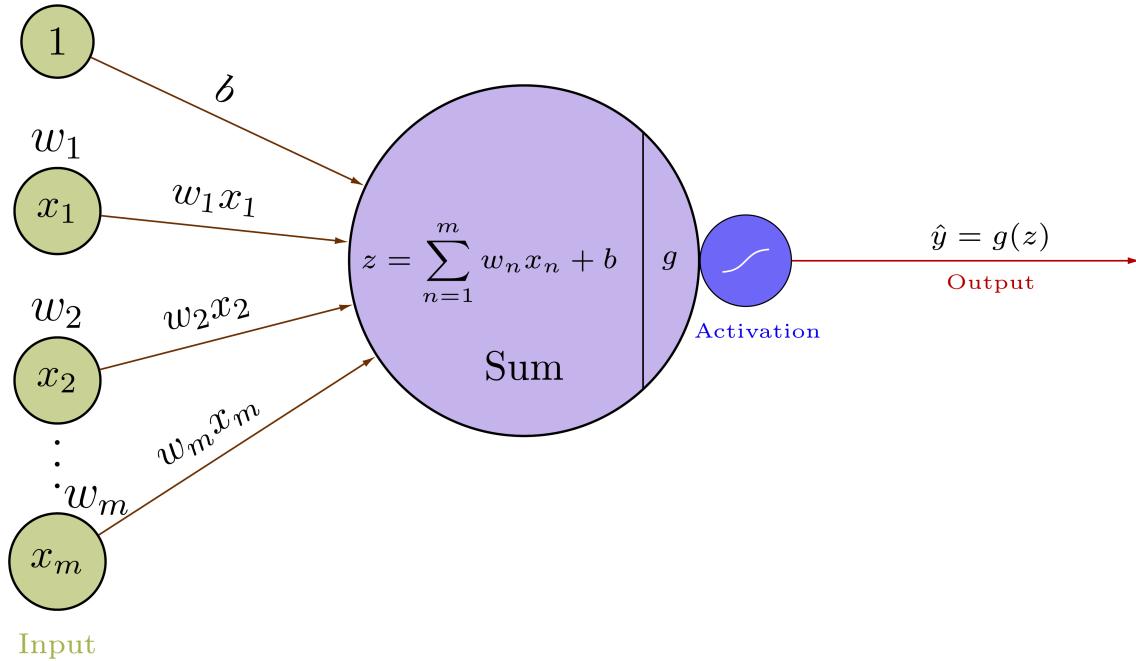


$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \quad \mathbf{X}^T = [x_1 \quad x_2 \quad \cdots \quad x_m]$$

$$z = \sum_{n=1}^m w_n x_n + b = \mathbf{X}^T \cdot \mathbf{W} + b = \mathbf{W}^T \cdot \mathbf{X} + b$$

$$\hat{y} = g(z)$$

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$



Пряме поширення

$$\begin{aligned}
 z &= \sum_{n=1}^m w_n x_n + b = \mathbf{X}^T \cdot \mathbf{W} + b = \mathbf{W}^T \cdot \mathbf{X} + b \\
 \hat{y} &= g(z) \\
 \mathcal{L}(\hat{y}, y) &= -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))
 \end{aligned}$$

Приклад

Припустимо $m = 3$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.7 \\ 0.5 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} \quad b = 0.8$$

$$\begin{aligned} z &= \sum_{n=1}^3 w_n x_n + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + b = \\ &= 1 \cdot -0.1 + -2 \cdot 0.7 + 2 \cdot 0.5 + 0.8 = 0.3 \end{aligned}$$

$$\begin{aligned} z &= \mathbf{X}^T \cdot \mathbf{W} + b = [x_1 \quad x_2 \quad x_3] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + b = \\ &= w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0.3 \end{aligned}$$

$$\hat{y} = g(z) = g(\mathbf{X}^T \cdot \mathbf{W} + b) = \frac{1}{1 + \exp(-z)} = \frac{1}{1 + \exp(-0.3)} \approx 0.57$$

Одновимірний градієнтний спуск

Одновимірний градієнтний спуск

Розглянемо деяку неперервну, диференційовану функцію $f : \mathbb{R} \rightarrow \mathbb{R}$. Розкладши в ряд Тейлора, ми отримуємо:

$$f(x + \varepsilon) = f(x) + \varepsilon f'(x) + \mathcal{O}(\varepsilon^2)$$

Для простоти давайте виберемо фіксований розмір кроку $\alpha > 0$ та оберемо $\varepsilon = -\alpha f'(x)$. Підставивши це у попередній вираз:

$$f(x - \alpha f'(x)) = f(x) - \alpha f'^2(x) + \mathcal{O}(\alpha^2 f'^2(x))$$

Якщо похідна $f'(x) \neq 0$ не зникає, ми робимо прогрес, оскільки $\alpha f'^2(x) > 0$. Крім того, ми завжди можемо вибрати α досить малим, щоб вирази вищого порядку занулити. Тому ми приходимо до

$$f(x - \alpha f'(x)) \lesssim f(x)$$

Це означає, що якщо ми використовуємо:

$$x \leftarrow x - \alpha f'(x)$$

для ітерації по x , значення функції $f(x)$ може зменшитись.

```

import numpy as np

def f(x):  # Objective function
    return x**2

def f_grad(x):  # Gradient (derivative) of the objective function
    return 2 * x

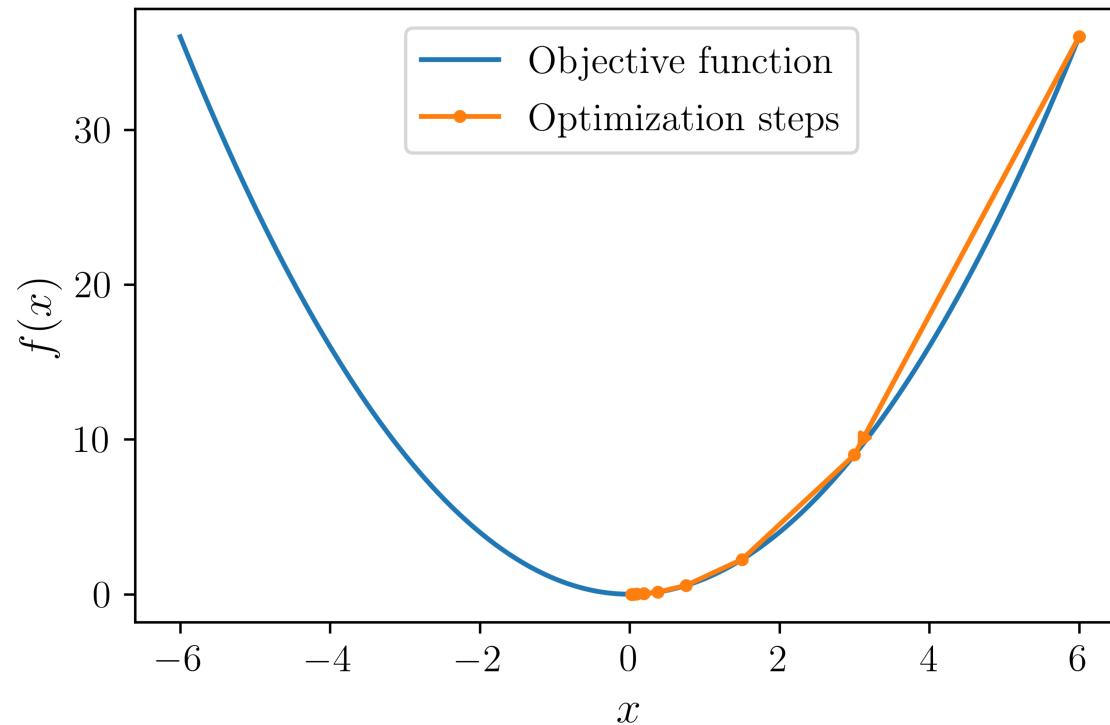
def bgd(alpha, f_grad):
    x = 6.0          # Initial value of x
    results = [x]
    epoch = 8         # Number of iterations
    for i in range(epoch):
        x -= alpha * f_grad(x)
        results.append(float("%.6f" % x))
    print(f'epoch {epoch}, x: {x:.6f}')
    return results

results = bgd(0.25, f_grad)
print(results)

```

epoch 8, x: 0.023438
[6.0, 3.0, 1.5, 0.75, 0.375, 0.1875, 0.09375, 0.046875, 0.023438]

Хід оптимізації за значеннями x

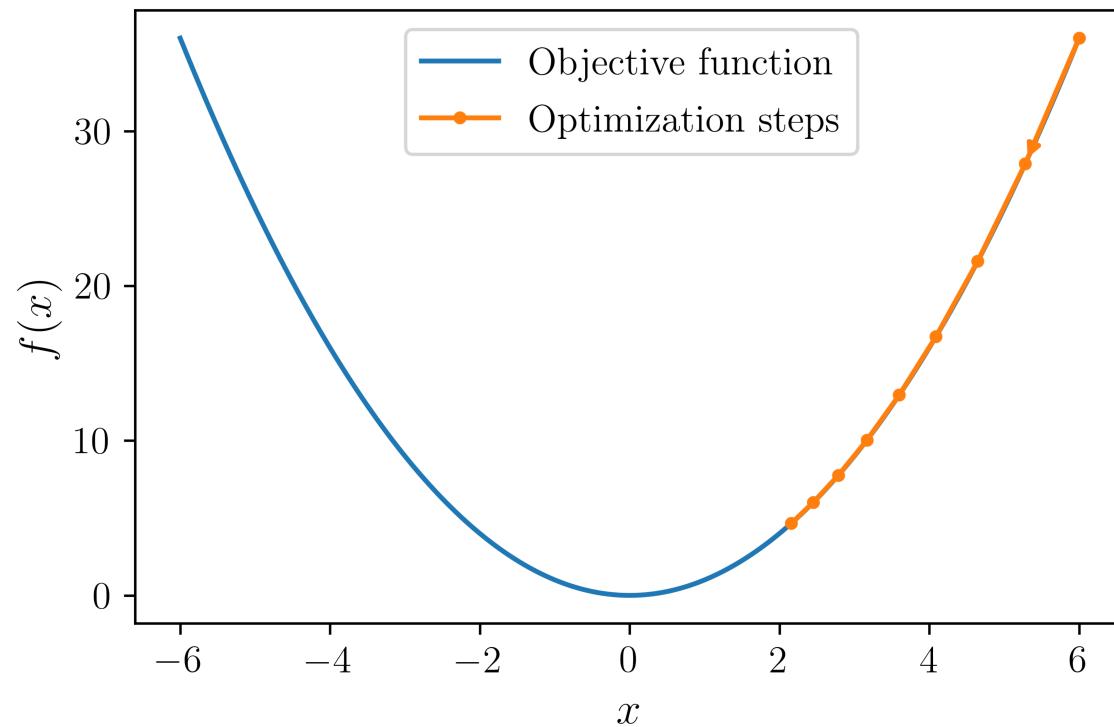


One-Dimensional Gradient Descent ($\alpha = 0.25$)

Хід оптимізації за значеннями x

```
results = bgd(0.06, f_grad)
```

```
epoch 8, x: 2.157807
```

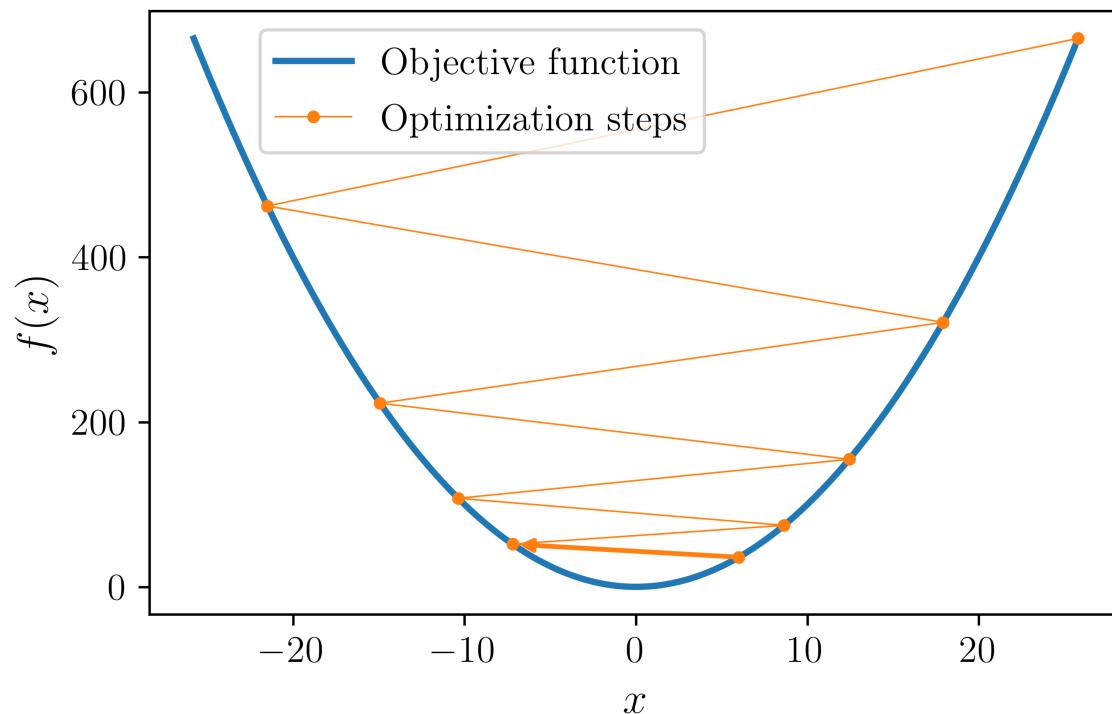


One-Dimensional Gradient Descent ($\alpha = 0.06$)

Хід оптимізації за значеннями x

```
results = bgd(1.1, f_grad)
```

epoch 8, x: 25.798902



One-Dimensional Gradient Descent ($\alpha = 1.1$)

Перцептрон: Зворотне поширення

У позначеннях Лейбніца **правило ланцюжка** стверджує, що

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{k \in \text{parents}(\ell)} \frac{\partial \ell}{\partial u_k} \underbrace{\frac{\partial u_k}{\partial \theta_i}}_{\text{recursive case}}$$

Зворотне поширення

- Оскільки нейронна мережа є **композицією диференційованих функцій**, загальні похідні втрат можна оцінити зворотно, застосовуючи рекурсивно правило ланцюжка до її обчислювального графу.
- Реалізація цієї процедури називається зворотним **автоматичним диференціюванням** або **зворотним поширенням**.

Пряме поширення

$$z = \sum_{n=1}^m w_n x_n + b = \mathbf{X}^T \cdot \mathbf{W} + b = \mathbf{W}^T \cdot \mathbf{X} + b$$

$$\hat{y} = g(z) = \sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{n} \sum_{i=1}^n (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

Зворотне поширення

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial z} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} = \hat{y} - y$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \mathbf{W}} = \mathbf{X}^T \cdot (\hat{y} - y)$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial b} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b} = \hat{y} - y$$

Оновлення параметрів

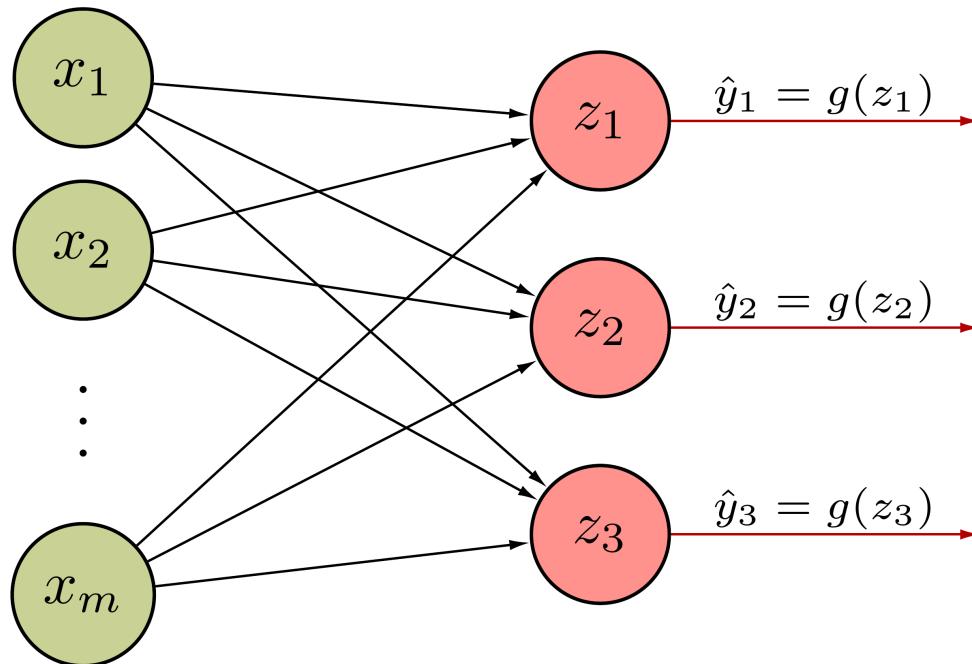
$$\mathbf{W} = \mathbf{W} - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \mathbf{W}}$$

$$b = b - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial b}$$

Персептрон з багатьма виходами

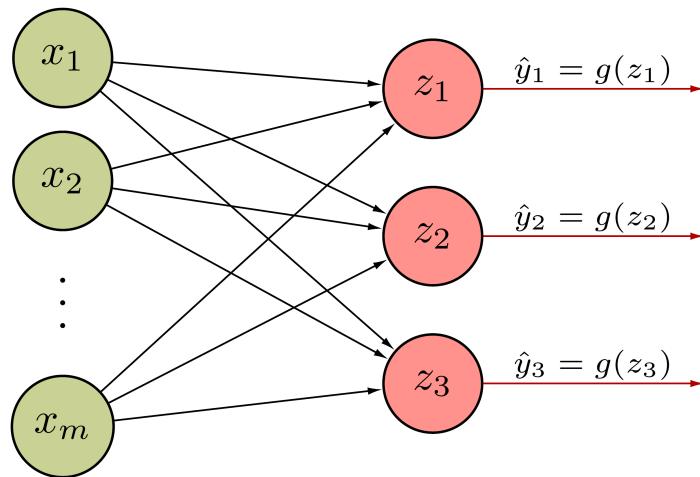
Multi Output Perceptron

Оскільки всі входи щільно з'єднані з усіма виходами, ці шари називаються Dense



$$z_j = \sum_{n=1}^m w_{j,n} x_n + b_j$$

Example



$$\mathbf{X}^{m \times 1} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad \mathbf{W}^{3 \times m} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ w_{31} & w_{32} & \cdots & w_{3m} \end{bmatrix} \quad \mathbf{b}^{3 \times 1} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\boxed{\begin{aligned} \mathbf{z} = \mathbf{W} \cdot \mathbf{X} + \mathbf{b} &= \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ w_{31} & w_{32} & \cdots & w_{3m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \\ &= \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + \cdots + w_{1m}x_m + b_1 \\ w_{21}x_1 + w_{22}x_2 + \cdots + w_{2m}x_m + b_2 \\ w_{31}x_1 + w_{32}x_2 + \cdots + w_{3m}x_m + b_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \end{aligned}}$$



Dense layer from scratch

```
class MyDenseLayer(tf.keras.layers.Layer):
    def __init__(self, input_dim, output_dim):
        super(MyDenseLayer, self).__init__()

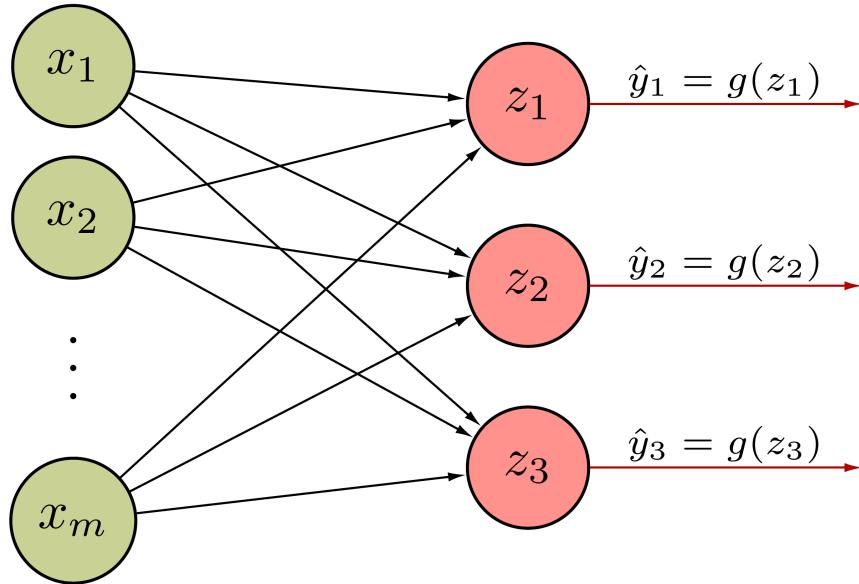
        # Initialize weights and bias
        self.W = self.add_weight([input_dim, output_dim])
        self.b = self.add_weight([1, output_dim])

    def call(self, inputs):
        # Forward propagate the inputs
        z = tf.matmul(inputs, self.W) + self.b

        # Feed through a non-linear activation
        output = tf.math.sigmoid(z)

    return output
```

Оскільки всі входи щільно з'єднані з усіма виходами, ці шари називаються **Dense**

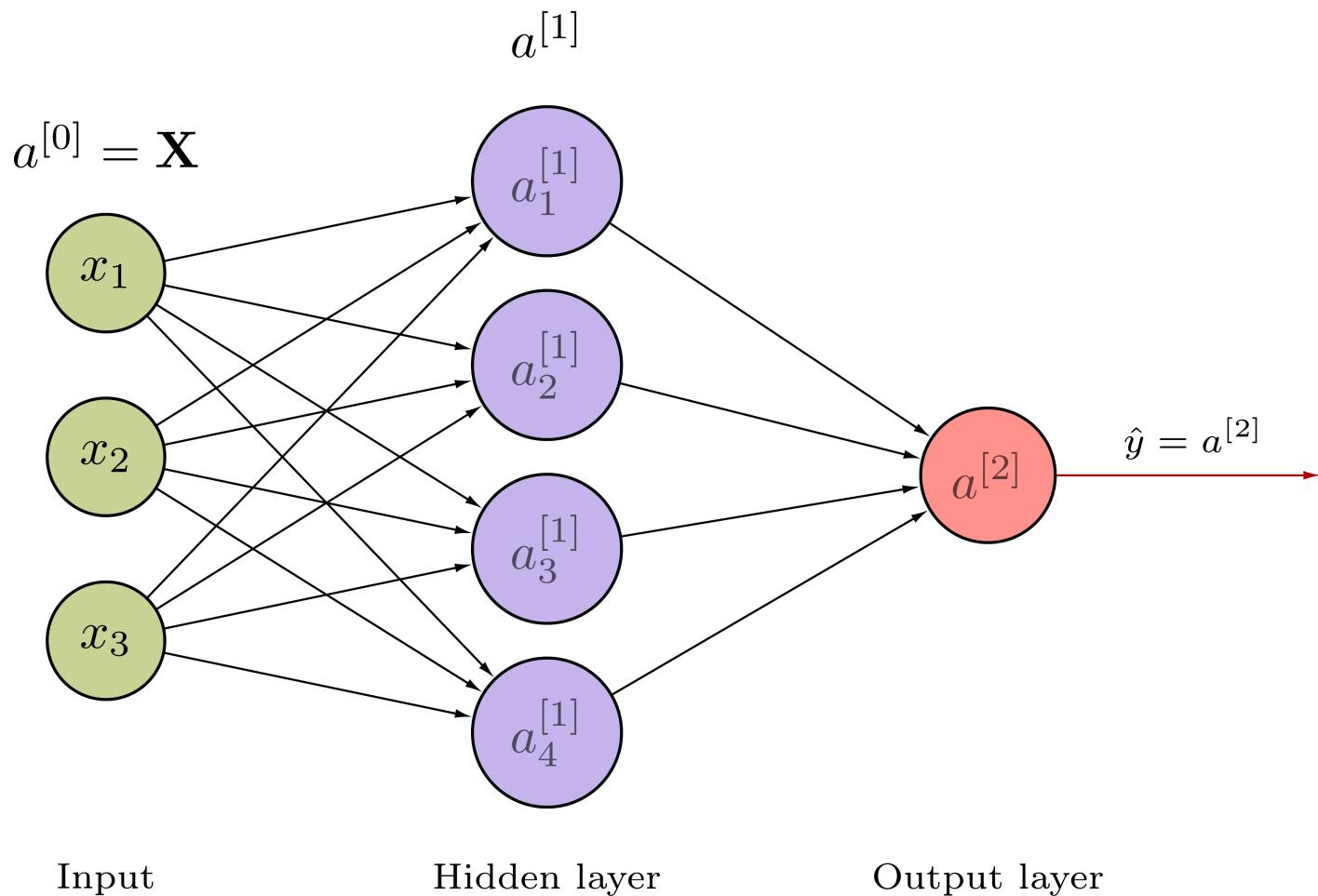


```
import tensorflow as tf  
  
layer = tf.keras.layers.Dense(  
    units=3)
```

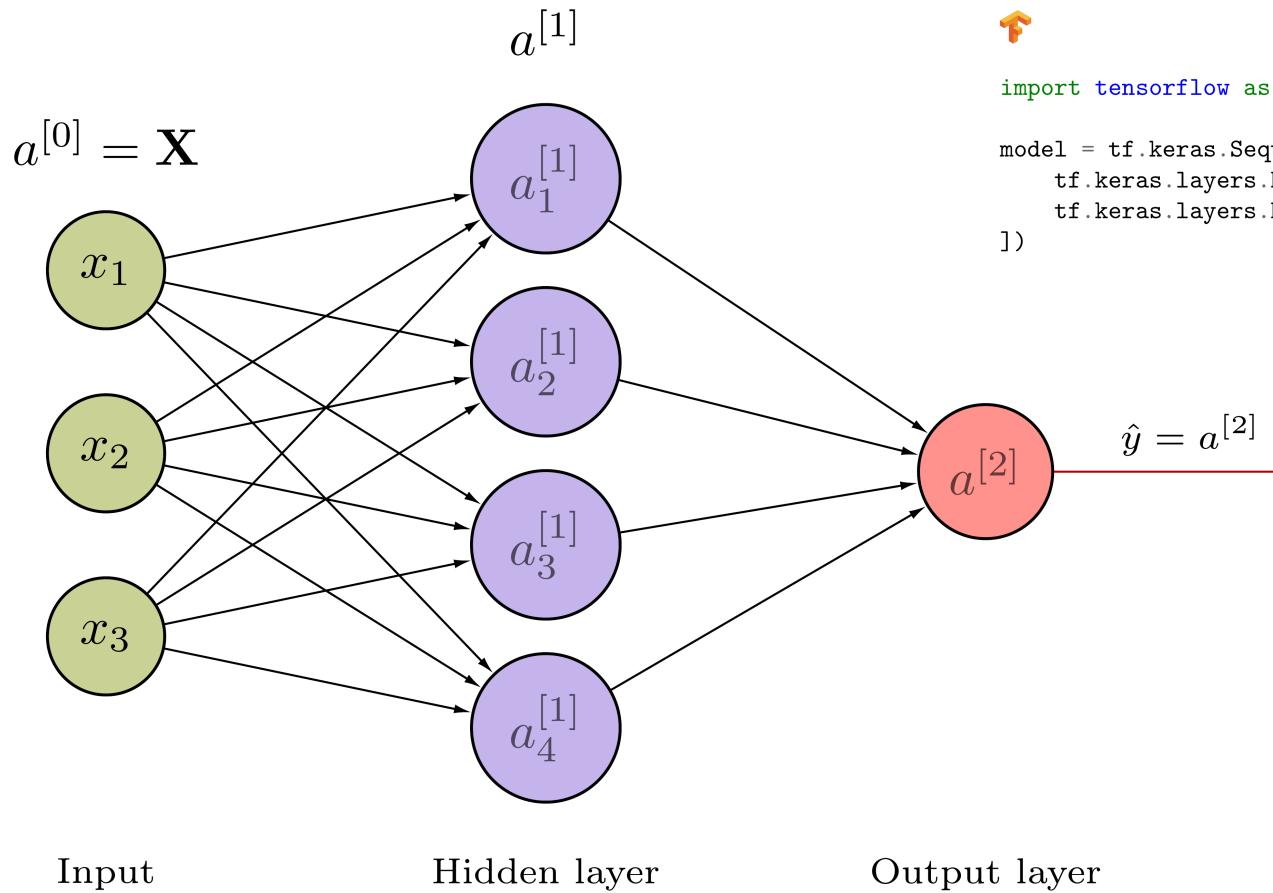
$$z_j = \sum_{n=1}^m w_{j,n} x_n + b_j$$

Багатошаровий перцептрон

Багатошаровий перцептрон



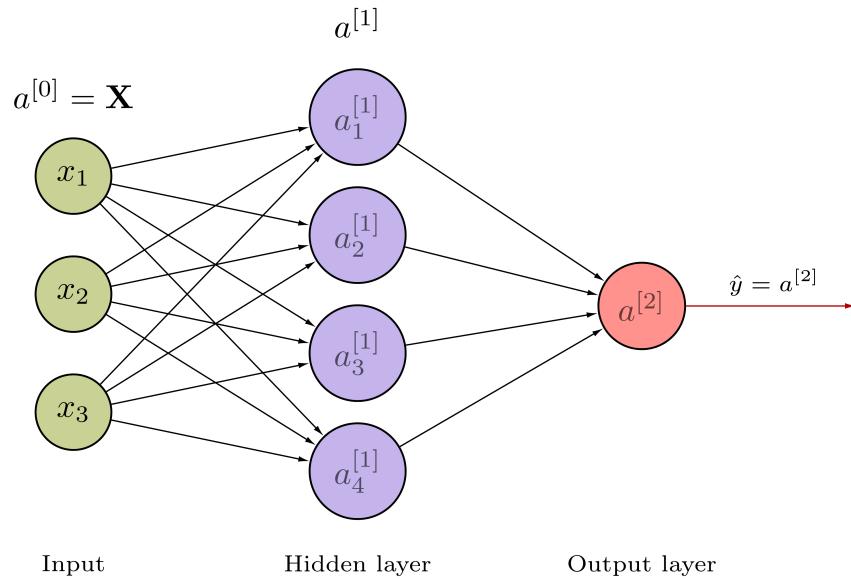
Мережа з одним прихованим шаром



```
import tensorflow as tf
```

```
model = tf.keras.Sequential([  
    tf.keras.layers.Dense(4),  
    tf.keras.layers.Dense(1)  
])
```

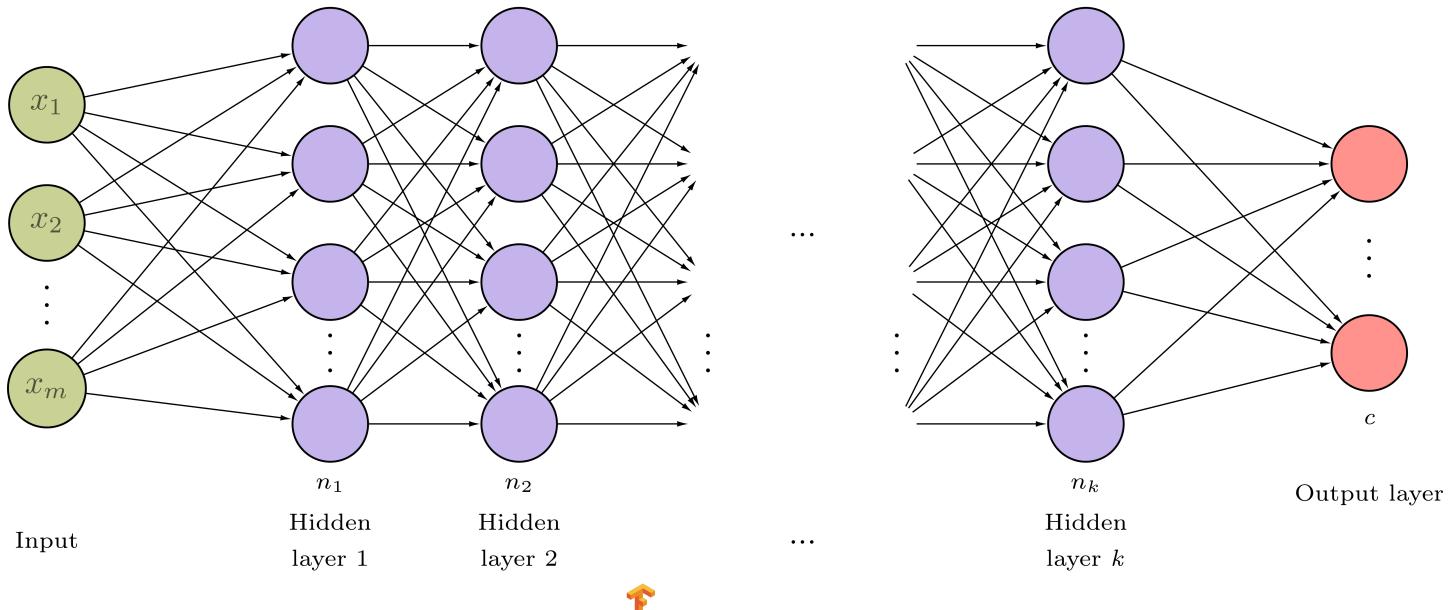
Мережа з одним прихованним шаром



$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{W}^{[1]} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \quad \mathbf{b}^{[1]} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \quad \mathbf{W}^{[2]} = [w_1 \quad w_2 \quad w_3 \quad w_4] \quad b^{[2]} = b$$

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]} \cdot \mathbf{X} + \mathbf{b}^{[1]} \\ \mathbf{a}^{[1]} &= g^{[1]}(\mathbf{z}^{[1]}) \\ z^{[2]} &= \mathbf{W}^{[2]} \cdot \mathbf{a}^{[1]} + b^{[2]} \\ \hat{y} &= a^{[2]} = g^{[2]}(z^{[2]}) \end{aligned}$$

Глибинна нейронна мережа



```
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Dense(n1),
    tf.keras.layers.Dense(n2),
    .
    .
    .
    tf.keras.layers.Dense(nk),
    tf.keras.layers.Dense(c)
])
```

Кінець

Література

- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.