



# Czym jest AI?

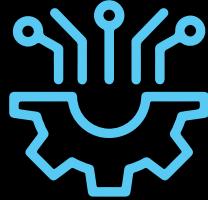
ARTIFICIAL INTELLIGENCE (AI)

W JĘZYKU POLSKIM: SZTUCZNA INTELIGENCJA (SI)

to programy komputerowe oparte na uczeniu maszynowym potrafiące **przyjmować dane od użytkowników i uczyć się na nich**. Nauczone programy potrafią transformować lub generować od nowa dane na podstawie zapamiętanych parametrów zależnie od potrzeb wykorzystania.



Supervised Learning



Machine Learning

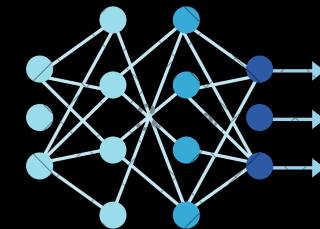


Unsupervised Learning

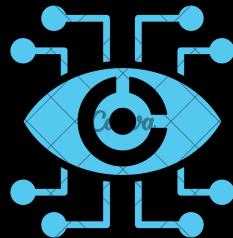


Reinforcement Learning

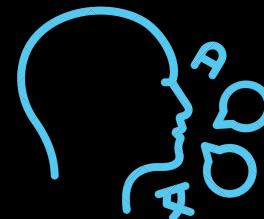
Typy AI



Deep Learning

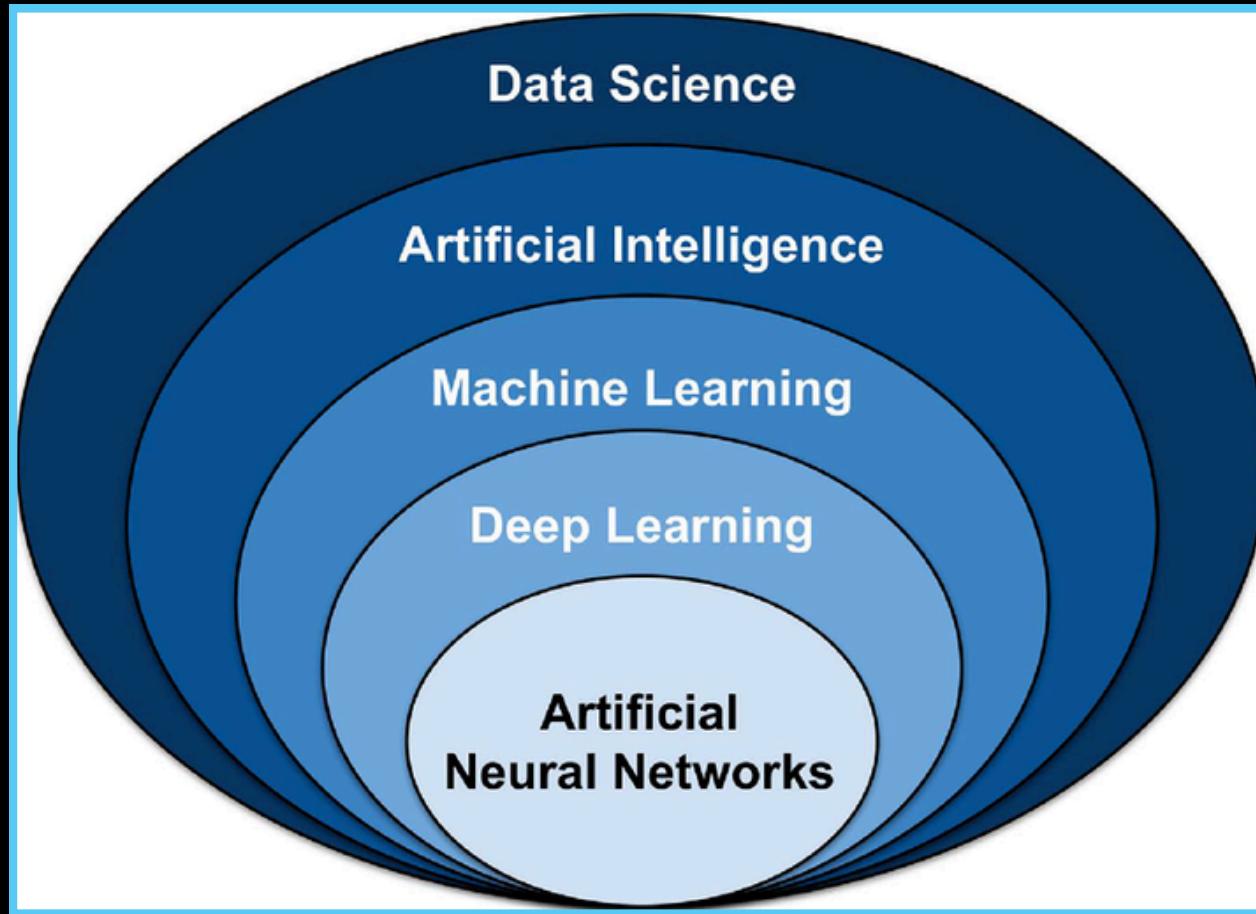


Computer Vision



Natural Language Processing

# Typy AI



<https://viso.ai/deep-learning/ml-ai-models/>

# Klasyfikacja



# Klasyfikacja

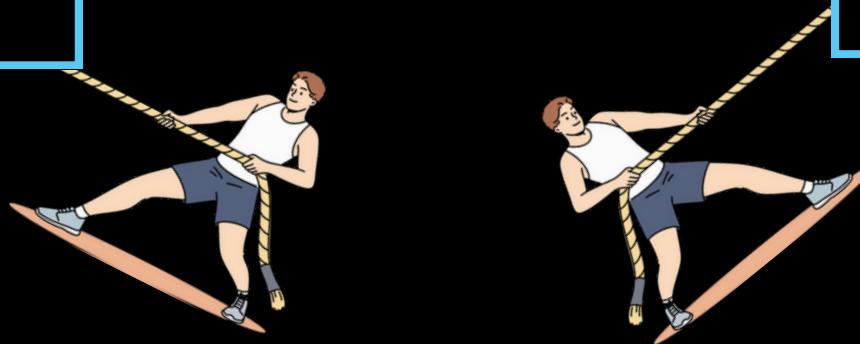


<https://viso.ai/deep-learning/ml-ai-models/>

# Regresja

zmienna zależna  
(docelowa)

zmienne niezależne  
(predykcyjne)



funkcja



znacznie szybszy, bo bazuje na  
działaniach bezpośrednio na  
tensorach

problem z niektórymi danymi ->  
konieczność dodatkowej obróbki  
przed ich konwersją na tensorы

składnia typowa dla Pythona

proste tworzenie przez  
dziedziczenie klas

wspiera typy danych z biblioteki  
NumPy

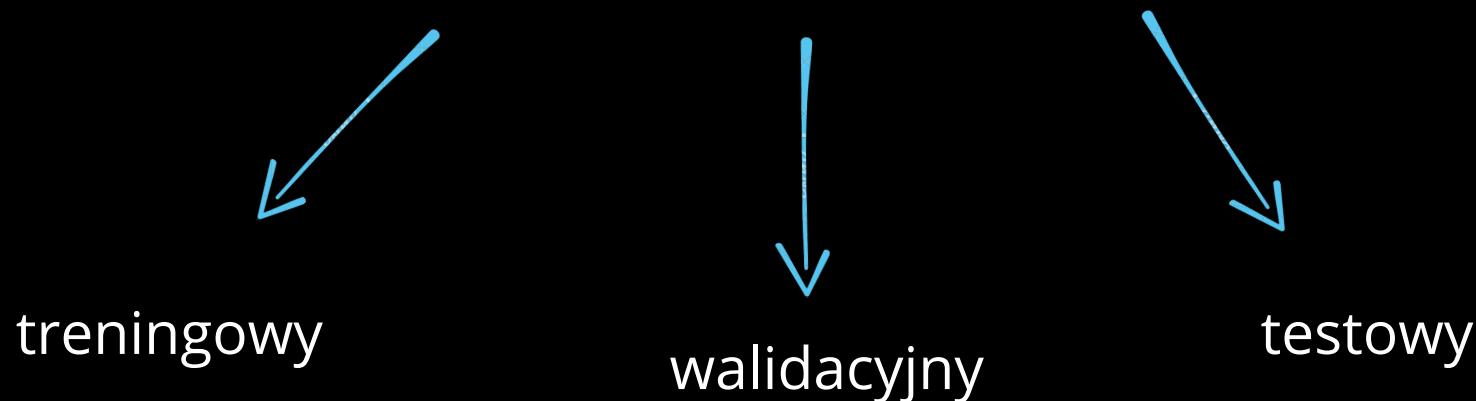
również używa tensorów, ale  
działania na nich są  
abstrahowane przez moduł

# A gdzie liczyć?



CPU                          vs                          GPU

# Podział zbiorów



# Podział zbiorów

treningowy

walidacyjny

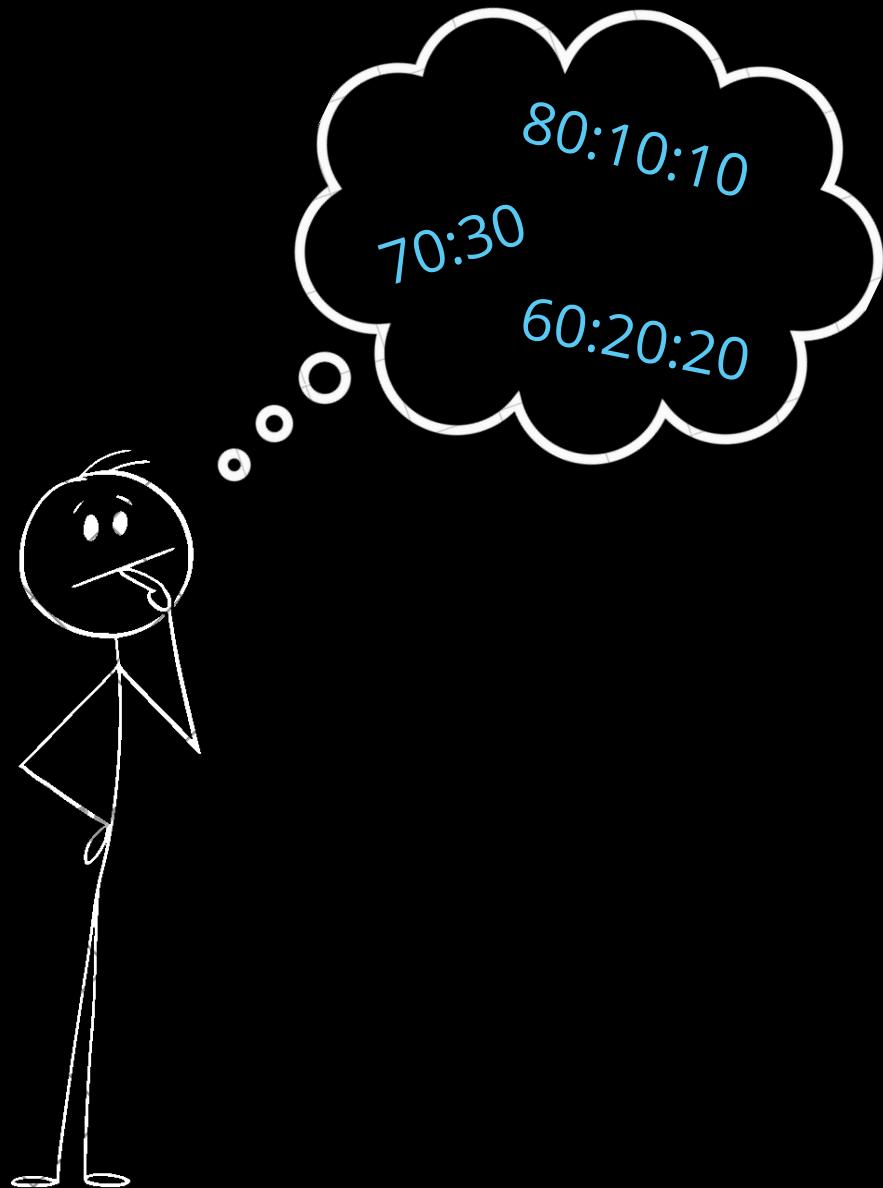
testowy

Random Split

Cross-validation

Stratified split

# Podział zbiorów



# Podział zbiorów

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)  
  
  
from torch.utils.data import random_split  
  
train_dataset, val_dataset, test_dataset = random_split(dataset, [train_size, val_size, test_size])
```

# Jak wielki powinien być zbiór danych?



Krótką odpowiedź brzmi: Tak.

# Overfitting w akcji, a.k.a halucynacje GPT

A screenshot of a ChatGPT conversation window. The background is dark grey, and the messages are white text on a light grey background. The conversation starts with a user message from 'AG' asking about the number of letters in the word 'nineteen'. ChatGPT responds correctly with 'There are eight letters in the word "nineteen".' The user then asks if ChatGPT is sure, and ChatGPT apologizes for a mistake, stating there are actually nine letters. The URL <https://www.decipherzone.com/blog-detail/chat-gpt-memes> is visible at the bottom of the screenshot.

AG How many letters are there in the word "nineteen"?

There are eight letters in the word "nineteen".

AG are you sure?

I apologize for my mistake. There are actually nine letters in the word "nineteen". Thank you for pointing that out.

<https://www.decipherzone.com/blog-detail/chat-gpt-memes>

RNN

CNN

GAN

## Typy sieci neuronowych

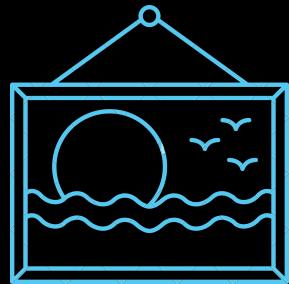
autoekondery

MLP

transfomery

# Pre-trenowane modele

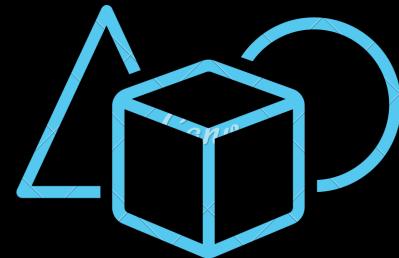
Analiza  
obrazów



ResNet

MobileNetV2

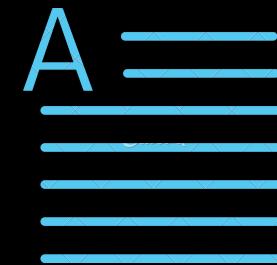
Detekcja  
obiektów



YOLO

Mask R-CNN

Przetwarzanie  
tekstu



BERT

GPT

# Pre-trenowane modele

TorchVision

<https://pytorch.org/vision/stable/models.html>

# Jak zapisać model?



# Jak zapisać model?

```
from joblib import dump  
  
dump(scaler, 'scaler.joblib')  
dump(svd, 'svd.joblib')  
dump(vectorizer, 'vectorizer.joblib')  
dump(classifier, 'sign_language_classifier.joblib')
```

torch.save()

```
from joblib import load  
  
model = load('sign_language_classifier.joblib')  
vectorizer = load('vectorizer.joblib')  
scaler = load('scaler.joblib')  
svd = load('svd.joblib')
```

torch.load()

# Joblib vs PyTorch

## Joblib

- Potrafi zapisać wszystkie obiekty używane w skryptach, łącznie z modelem i różnymi danymi,
- Prosty w użyciu,
- Nie obsługuje wewnętrznych struktur modeli (np. state-dict),
- Możliwe problemy z kompatybilnością pomiędzy środowiskami Pythona

## PyTorch

- Zapisuje model i jego stan,
- Potrafi zapisać konkretne części modelu (wagi, biasy, itp.),
- Może zapisać TYLKO model PyTorcha, pomija wszystkie inne dane ze skryptu,
- Zapisane modele mogą być kompatybilne tylko z konkretnymi wersjami PyTorch

# Struktura sieci neuronowej

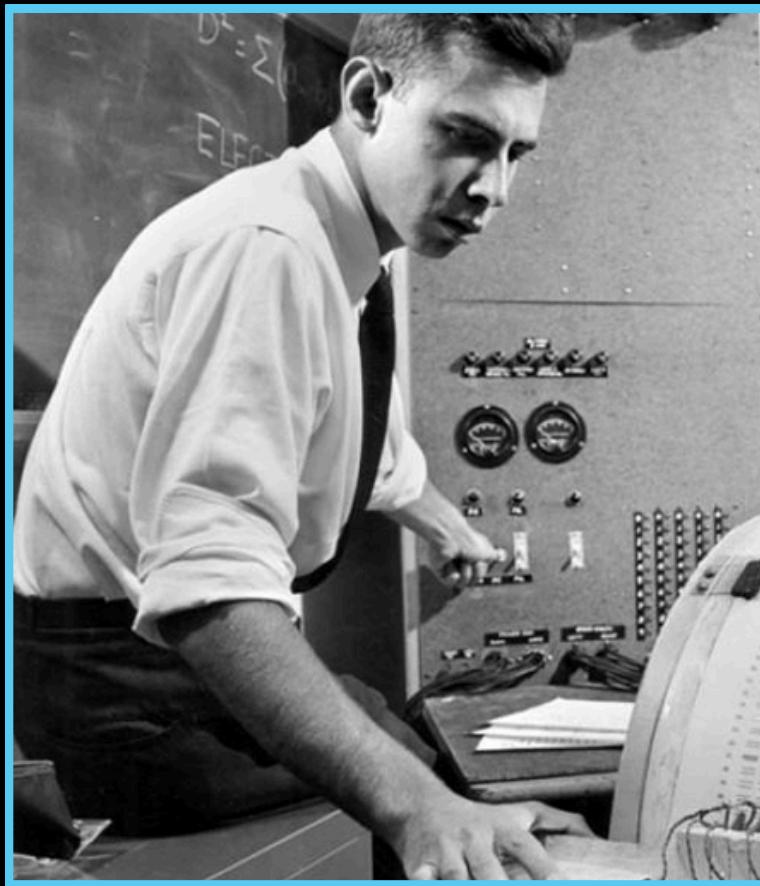
Najprostsza sieć neuronowa jest siecią typu **“Feed Forward”**.

Sieci te cechują się liniarnym, nierekurencyjnym (bez pętli) przepływem danych.

Każda sieć musi mieć warstwę gdzie wprowadzamy dane. Ta warstwa łączy się z warstwami ukrytymi które przetwarzają dane wejściowe za pomocą określonych funkcji, i ostatecznie łączą się z warstwą wyjściową.

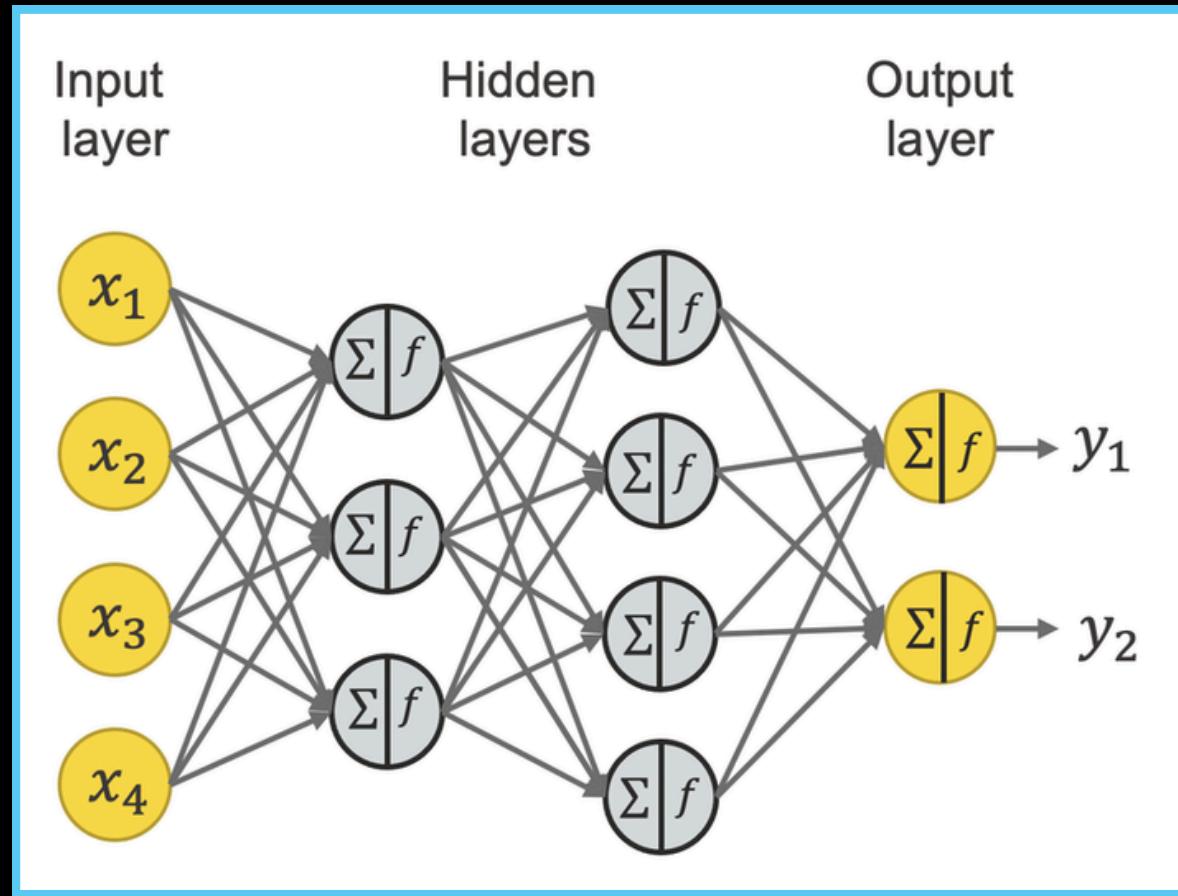
Zależnie od analizowanej ilości parametrów i spodziewanych danych wyjściowych, wymiary tych warstw mogą mieć różny rozmiar.

# Perceptron



Frank Rosenblatt

# Struktura sieci neuronowej



<https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks>

# Funkcje aktywacji

skokowa

$$f(z) = \begin{cases} 1 & \text{jeśli } z \geq 0 \\ 0 & \text{jeśli } z < 0 \end{cases}$$

sigmoidalna

$$f(z) = \frac{1}{1 + e^{-z}}$$

ReLU (Rectified Linear Unit)

$$f(z) = \max(0, z)$$

tangens hiperboliczny

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

softmax

$$f(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

# Funkcje aktywacji

Funkcja	Zastosowanie	Zakres	Wady
Skokowa	Prosta klasyfikacja (np. perceptron)	{0, 1}	Nieliniowa, nieróżniczkowalna
Sigmoid	Klasyfikacja binarna, probabilistyczne	(0, 1)	Zanikanie gradientu, nasycenie
tanh	Warstwy ukryte w sieciach neuronowych	(-1, 1)	Zanikanie gradientu
ReLU	Warstwy ukryte w złożonych sieciach	[0, $\infty$ )	Martwe neurony
Softmax	Klasyfikacja wieloklasowa	(0, 1)	Szybko rośnie złożoność obliczeniowa



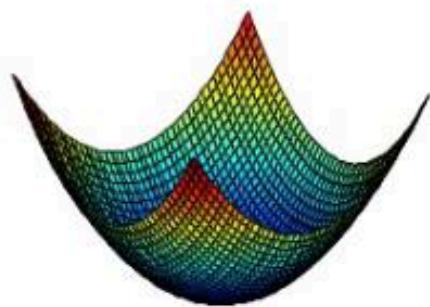
# Zadanie

Próbka	Cecha 1 (x1)	Cecha 2 (x2)	Oczekiwany wynik (y)
1	0.5	1.2	0
2	0.8	0.4	1
3	1.0	1.0	1

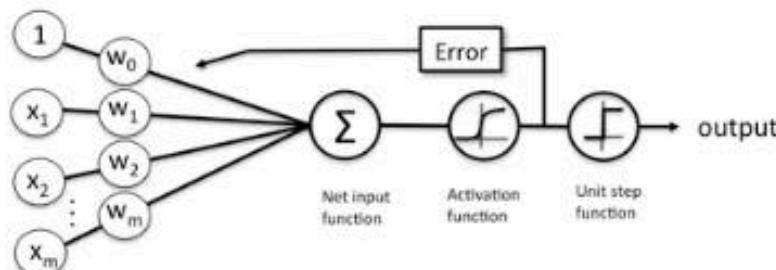
$$z = w_1 \times x_1 + w_2 \times x_2 + b$$

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

You

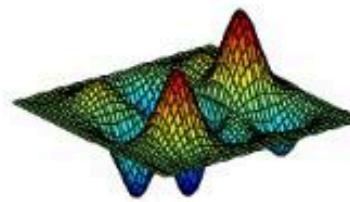


- Unique optimum: global/local.



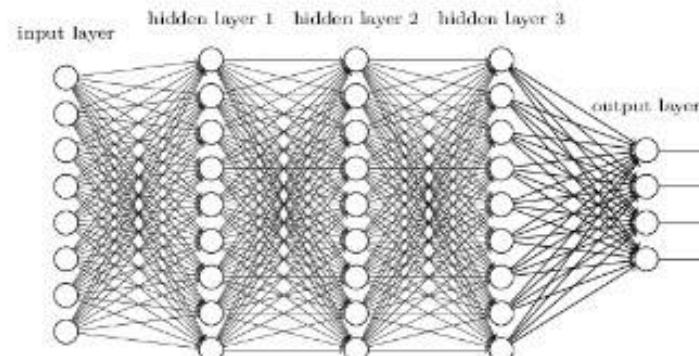
Schematic of a logistic regression classifier.

The guy she tells you  
not to worry about



- Multiple local optima
- In high dimensions possibly

Deep neural network



# Struktura sieci neuronowej

```
class ClassifierCNNNet(nn.Module): 1 usage
    def __init__(self, debug=False):
        super().__init__()
        self.debug = debug
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=20, kernel_size=5)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(in_channels=20, out_channels=40, kernel_size=5)
        self.fc1 = nn.Linear(40 * 61 * 61, out_features=1000)
        self.fc2 = nn.Linear(in_features=1000, out_features=500)
        self.fc3 = nn.Linear(in_features=500, out_features=8)

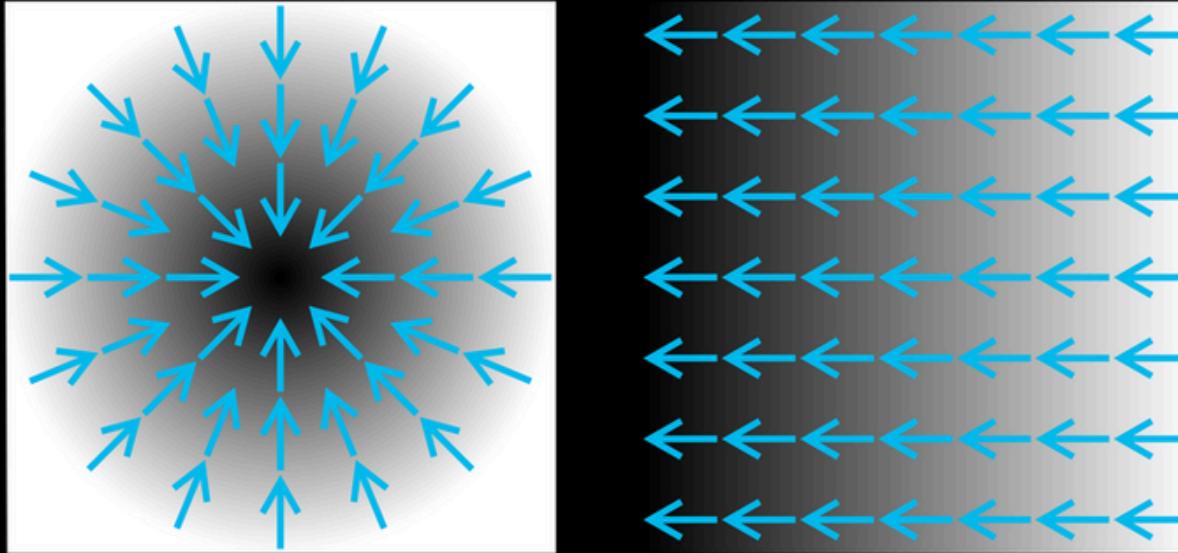
    def forward(self, x):
        x = F.relu(self.conv1(x))
        if self.debug: print(f"After conv1: {x.shape}")
        x = self.pool(x)
        if self.debug: print(f"After pool 1: {x.shape}")
        x = F.relu(self.conv2(x))
        if self.debug: print(f"After conv2: {x.shape}")
        x = self.pool(x)
        if self.debug: print(f"After pool 2: {x.shape}")
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        if self.debug: print(f"After flatten: {x.shape}")
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Przykład sieci konwolucyjnej do klasyfikacji, Jakub Susoł

# Co to gradient?



# Gradienty w AI



Do określenia jak wagи i biasy neuronów powinny być dostosowane żeby polepszyć przewidywania.

