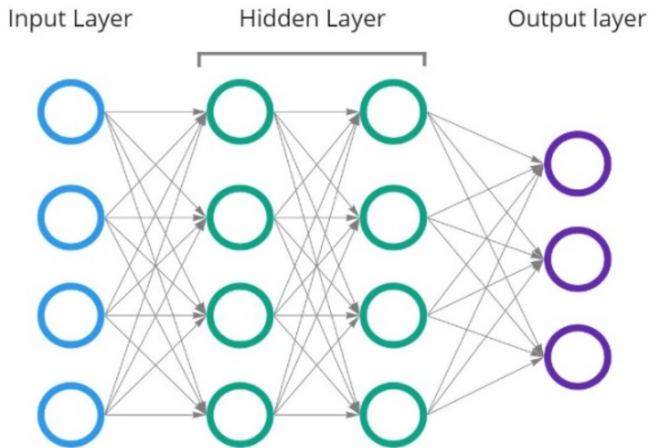
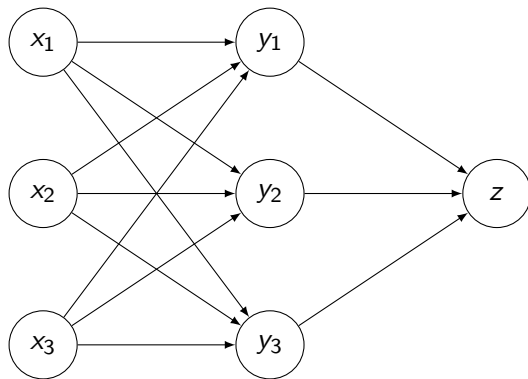


Typowa reprezentacja sieci neuronowej



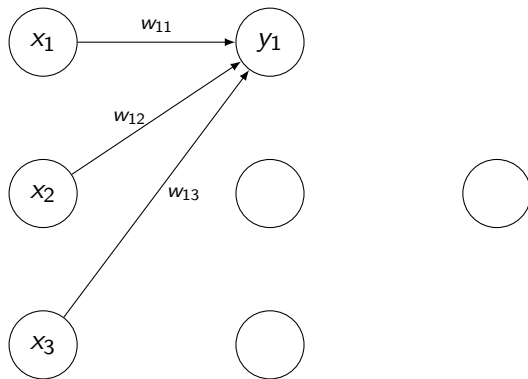
<https://www.researchgate.net/figure/Neural-Network-Representation'fig2'370269724>

Typowa reprezentacja sieci neuronowej



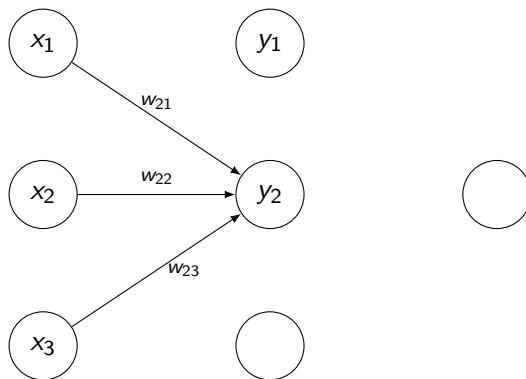
Niech $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ – funkcja aktywacji (np. sigmoid, ReLU, tgh).

Typowa reprezentacja sieci neuronowej



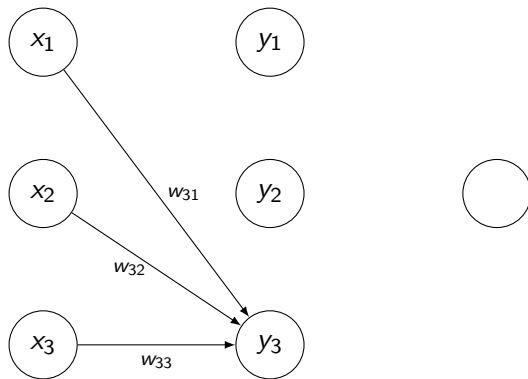
$$y_1 := \sigma(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1).$$

Typowa reprezentacja sieci neuronowej



$$y_2 := \sigma(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2).$$

Typowa reprezentacja sieci neuronowej



$$y_3 := \sigma(w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3).$$

Razem:

$$y_1 = \sigma(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1),$$

$$y_2 = \sigma(w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2),$$

$$y_3 = \sigma(w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3).$$

Oznaczenie:

$$y'_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1,$$

$$y'_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2,$$

$$y'_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3.$$

Wtedy:

$$y_1 = \sigma(y'_1),$$

$$y_2 = \sigma(y'_2),$$

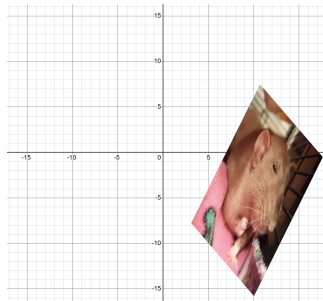
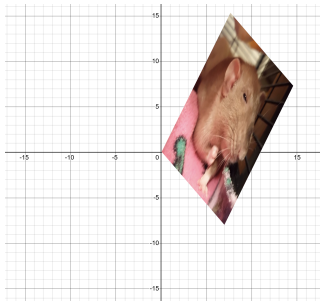
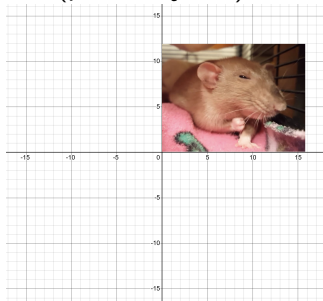
$$y_3 = \sigma(y'_3).$$

Macierzowo:

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \sigma(y_1') \\ \sigma(y_2') \\ \sigma(y_3') \end{pmatrix} \quad \left[= \sigma \left(\begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} \right) \right].$$

Przekształcenie afiniczne – złożenie przekształcenia liniowego z translacją (przesunięciem).



$$F(X) = AX + B.$$

$$X \in \mathbb{R}^n, A \in M_{m \times n}(\mathbb{R}), B \in \mathbb{R}^m.$$

Sieć neuronowa F (właściwie: wielowarstwowy perceptron) to zatem złożenie przekształceń afinicznych „przeplatanych” z funkcjami aktywacji, to znaczy:

$$F = \sigma_n \circ F_n \circ \dots \circ \sigma_1 \circ F_1 \circ \sigma_0 \circ F_0,$$

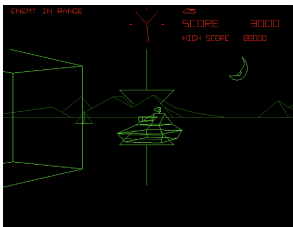
gdzie $\sigma_i: \mathbb{R}^* \rightarrow \mathbb{R}^*$ to zwektoryzowane funkcje aktywacji, a F_i to kompatybilne przekształcenia afiniczne.

Dla konkretnego wejścia (wektora X), obliczenia związane z wyznaczeniem wyjścia ($= F(X)$) to głównie „czysta” algebra liniowa. Te obliczenia łatwo się paralelizują.

Dygresja w dygresji: grafika 3D

Grafika 3D: w znacznej części algebra liniowa.

- Elementy sceny 3D to wielościany.
- Główny problem: narysować dużo wielościanów uwzględniając perspektywę i ruch kamery.
- Typowe rozwiązanie: kamera stoi w miejscu, przekształcamy resztę świata (obrot światła wokół kamery zamiast obrotu kamery w świecie).
- Te transformacje (wielościanów) to przekształcenia afiniczne.



1980: mało ścian



2015: więcej ścian

Dygresja w dygresji: grafika 3D

Pierwotnie obliczenia związane z grafiką 3D były wykonywane przez CPU.

Lata '90: upowszechnienie GPU – wyspecjalizowane układy do (głównie) obliczeń macierzowych.

- CPU: mało rdzeni ($\sim$ kilkanaście) ogólnego zastosowania.
- GPU: dużo rdzeni (tysiące); rdzenie są wyspecjalizowane, indywidualnie mało wydajne, wykorzystujące wysoką paralelizację obliczeń macierzowych.
- Surowa moc obliczeniowa GPU i CPU (z podobnej „półki”): $\sim 10:1$ na korzyść GPU.

GPU jest zoptymalizowane pod „te same” obliczenia, co grafika 3D – stąd zastosowanie GPU w uczeniu maszynowym.

Druga połowa lat '10: TPU (*tensor processing unit*).

Wytrenowany model ML (w ujęciu matematycznym i nieco uproszczonym) to po prostu funkcja.

„Surowe” dane opisujące wytrenowany model (zakładamy tu sieć neuronową):

- Opis architektury modelu (rozmiary macierzy, definicje funkcji przejścia).
- Faktyczne wartości współczynników macierzy i biasów (parametry).
- Dla np. modeli językowych: zanurzenia tokenów (por. Algebra liniowa 1 (AI), lista 4 laboratoriów).
- Inne metadane.

Różne standardowe formaty przechowywane danych (jako binarne bloby).

Silnik inferencyjny (w rozumieniu ML): software wykonujący wytrenowany model.

Ładuje model (może obsługiwać wiele formatów) i *wydajnie* wylicza wykonanie modelu dla podanych danych wejściowych; np. [Torch, Tensorflow,] ONNX, Llama,

Typowy schemat aplikacji ML (i nie tylko ML):

- Silnik inferencyjny z załadowanym modelem.
- Interfejs (np. webowy).
- Kod „sklejający” silnik z interfejsem.

Dzisiejszy ekosystem: duży wybór dostawców inferencji (hostujących silniki i modele) oraz interfejsów.

Przykład na resztę wykładu: Hugging Face. <https://huggingface.co/>