Sieci Neuronowe

Dlaczego sieci neuronowe?

Rozpoznawanie mowy

- Do 2009: GMM-HMM

  Odsetek błędnie rozpoznanych wyrazów: **23-27%**

- 2009

  A. Mohamed, G. Dahl, i G. Hinton, "Deep belief networks for phone recognition", NIPS 2009

  DNN-HMM

  Odsetek błędnie rozpoznanych wyrazów: **16-18%**

  Google Voice Input: **12.3%**

Dlaczego sieci neuronowe?

Rozpoznawanie zawartości zdjęć

ImageNet Large-Scale Visual Recognition Challenge

- Do 2012: FV+SVM

  Odsetek błędów: Top-1 = **45.7%,** Top-5 = **25.7%**

- 2012

  A. Krizhevsky, I. Sutskever i G.E. Hinton "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012

  8-mio warstwowa sieć konwolucyjna

  Odsetek błędów: Top-1 = **37.5%,** Top-5 = **17.0%**

- 2015

  Zespół sieci ResNet (152 warstwy ukryte!)

  Odsetek błędów: Top-5 = **3.57%**

Dlaczego sieci neuronowe?

2016/2017:

- Marzec 2016

  AlphaGo wygrywa mecz przeciwko Lee Sedol, 9p.

  Maj 2017

  Kolejna iteracja AlphaGo wykrywa z Ke Jie, obecnie najmocniejszym graczem w Go.

- Przełom 2016/2017
  Neuronowe systemy translacji tekstu wypierają klasyczne systemy statystycznego tłumaczenia maszynowego.

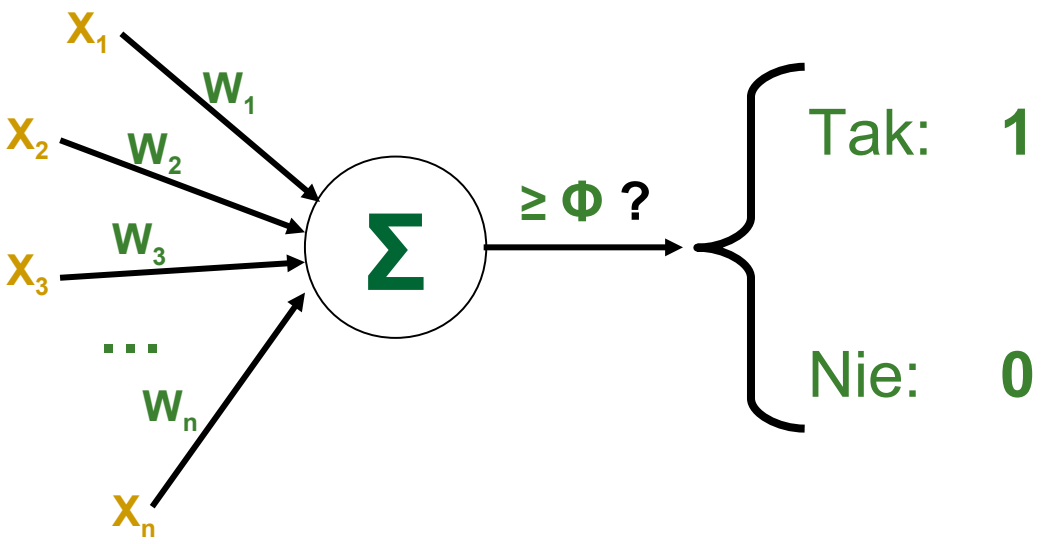
2017/2018:

Alpha Zero: bot uczący się grać w Go/Shogi/Szachy „od zera” (początkowo zna jedynie zasady gry). Jest mocniejszy od wszystkich dotychczasowych wersji AlphaGo.

2019:

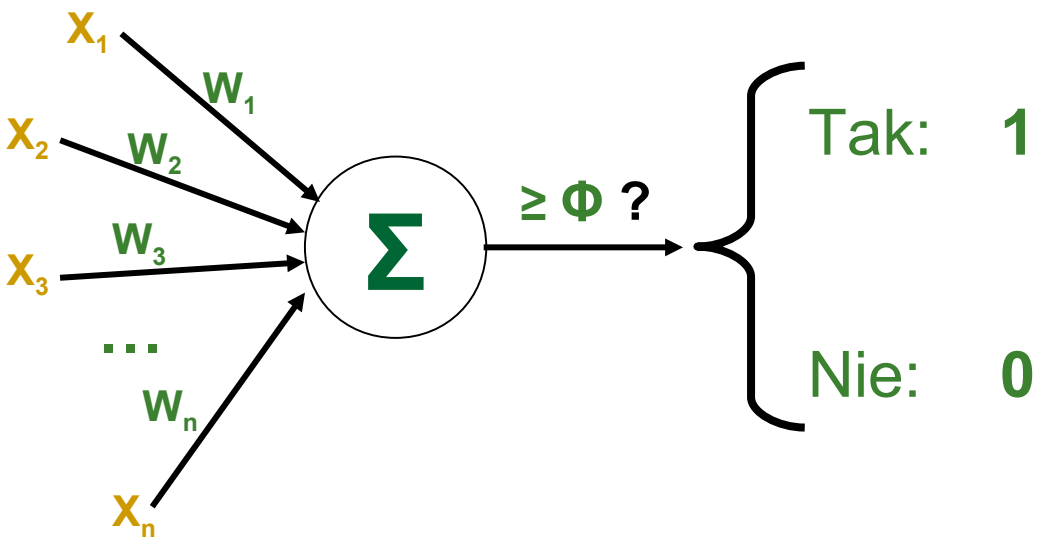
AlphaStar: bot wygrywający z profesjonalnymi graczami w StarCraft II

Perceptron



F. Rosenblatt 1957, The Perceptron – a perceiving and recognizing automaton, Cornell Aeronautical Laboratory Report

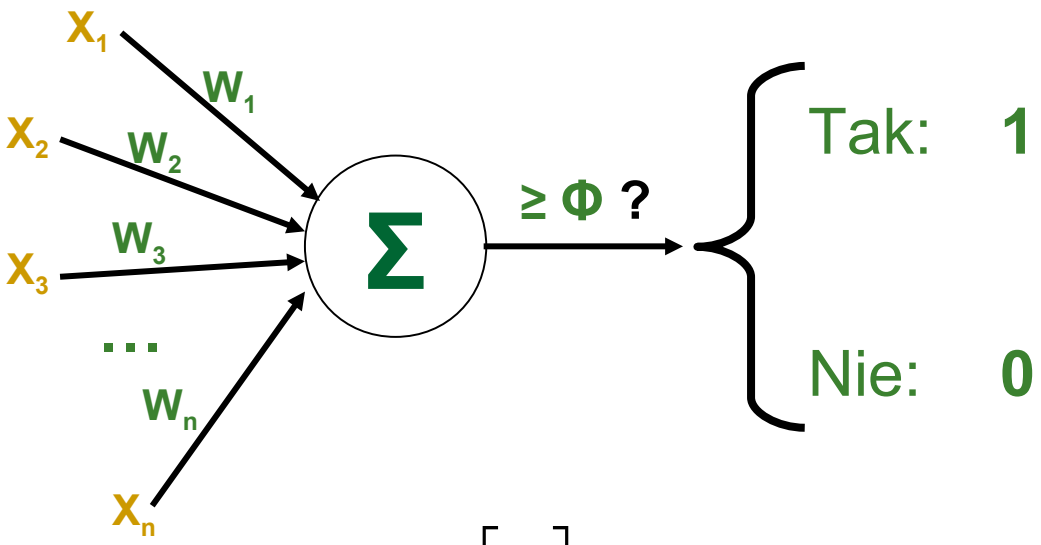
Perceptron



Czy obiekt na zdjęciu przedstawia twarz?

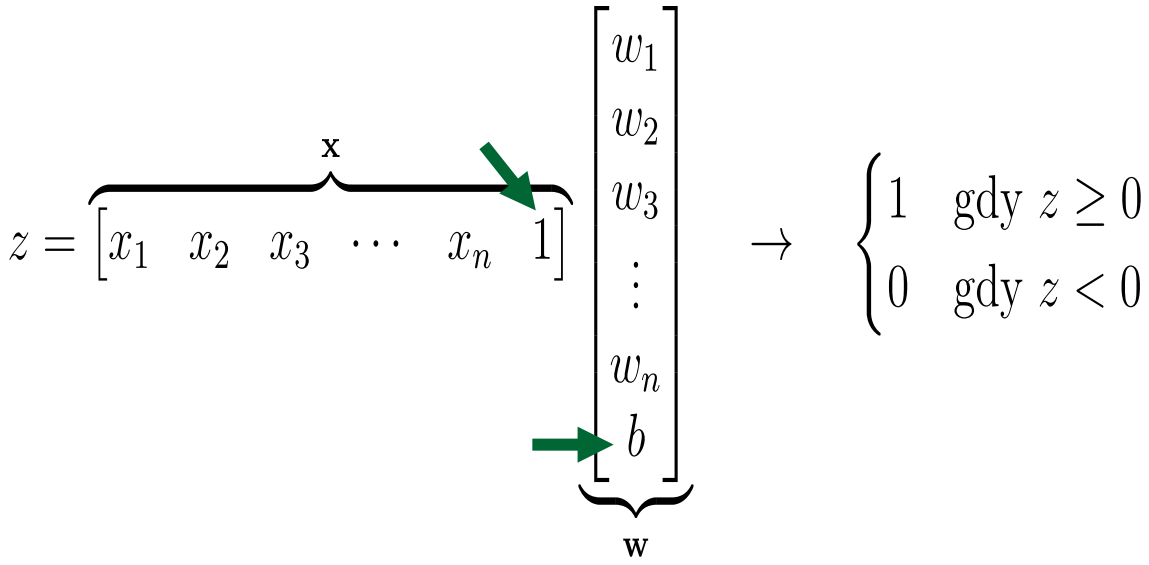
- x_1 – czy jest elipsoidalny?
- x_2 – czy wykazuje symetrię w pionie?
- x_3 – …

Perceptron



$$z = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} + b \quad \rightarrow \quad \begin{cases} 1 & \text{gdy } z \geq 0 \\ 0 & \text{gdy } z < 0 \end{cases}$$

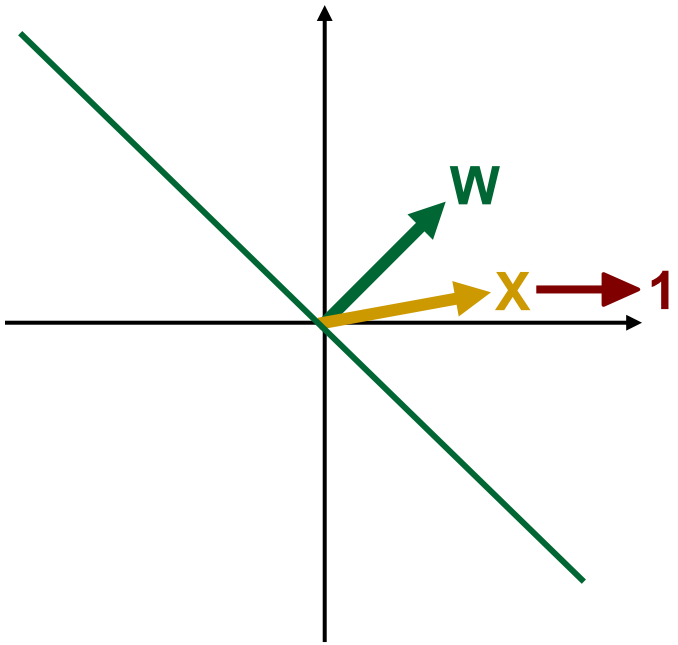
Perceptron

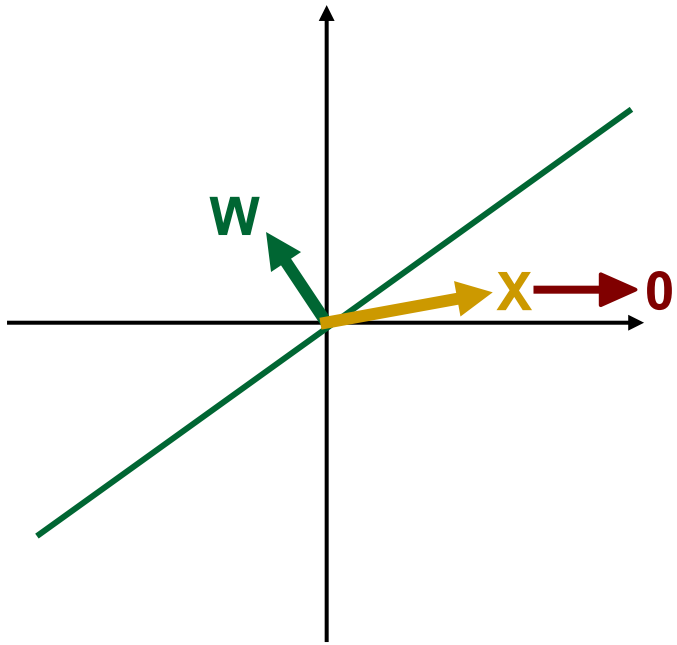


$$z = \mathbf{xw} \quad \rightarrow \quad \begin{cases} 1 & \text{gdy } z \geq 0 \\ 0 & \text{gdy } z < 0 \end{cases}$$

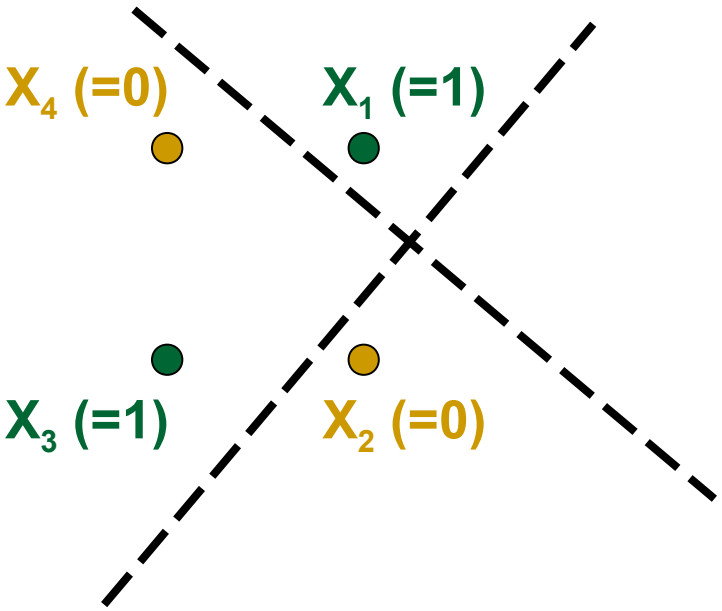
Perceptron

$$z = \mathbf{xw} \quad \rightarrow \quad \begin{cases} 1 & \text{gdy } z \geq 0 \\ 0 & \text{gdy } z < 0 \end{cases}$$





Perceptron



Problem XOR: brak rozwiązania liniowego.
Perceptron go nie rozwiąże.

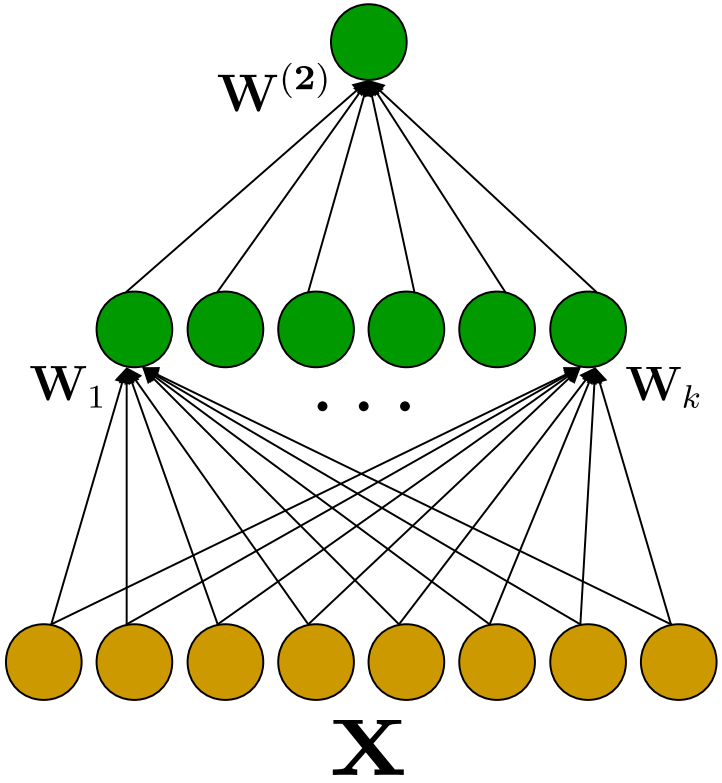
Minsky M. L. and Papert S. A. 1969. Perceptrons, MIT Press.

Dlaczego sieci głębokie

Perceptron wielowarstwowy

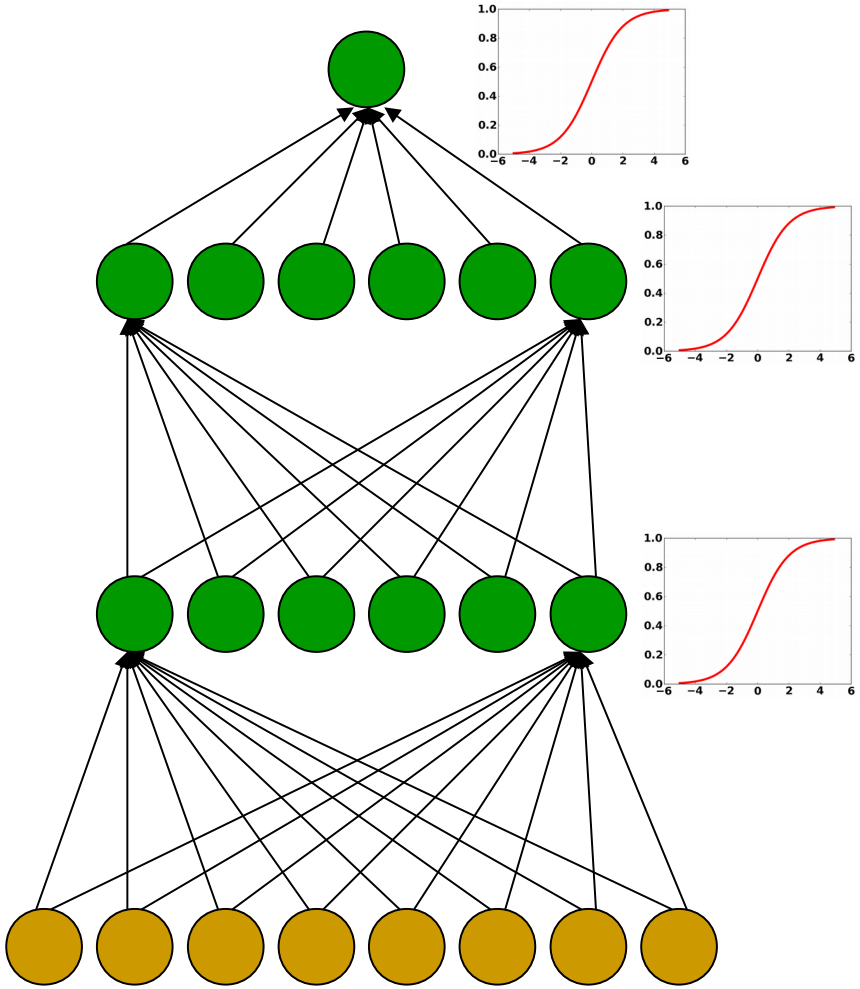
Jeśli odpowiednio dobierzemy cechy (deskryptory) obiektów, perceptron będzie mógł je prawidłowo sklasyfikować.

- Jednak nie chcemy konstruować cech „ręcznie”.
- Może więc wyuczymy zbiór perceptronów, tak by rozpoznawały pewne podstawowe cechy w obiektach.
- Następnie cechy te będą wejściem do końcowego perceptronu.



Dlaczego sieci głębokie

Perceptron wielowarstwowy



„Perceptron” wielowarstwowy (*Multilayer perceptron*).

Sieć neuronowa z wieloma nieliniowymi warstwami.

Chcemy aby warstwy te wykrywały wysoko-poziomowe cechy rozpoznawanych obiektów.

Jak wyuczyć taką głęboką sieć neuronową?

Sieci Neuronowe

Algorytm wstecznej propagacji błędu

Uczenie nadzorowane

Dany jest zbiór obserwacji. Dla każdej obserwacji znamy również wartość pewnej cechy (etykieta, wartość przewidywana):

$$\mathcal{X} = \left\{ \left(\mathbf{v}^1, t^1 \right), \left(\mathbf{v}^2, t^2 \right), \ldots, \left(\mathbf{v}^n, t^n \right) \right\}$$

- Obserwacja: zbiór mutacji. Etykieta: nowotwór vs. zmiana łagodna.
- Obserwacja: treść wiadomości. Etykieta: spam vs. nie spam.
- Obserwacja: raport finansowy spółki. Wartość przewidywana: zmiana kursu akcji.

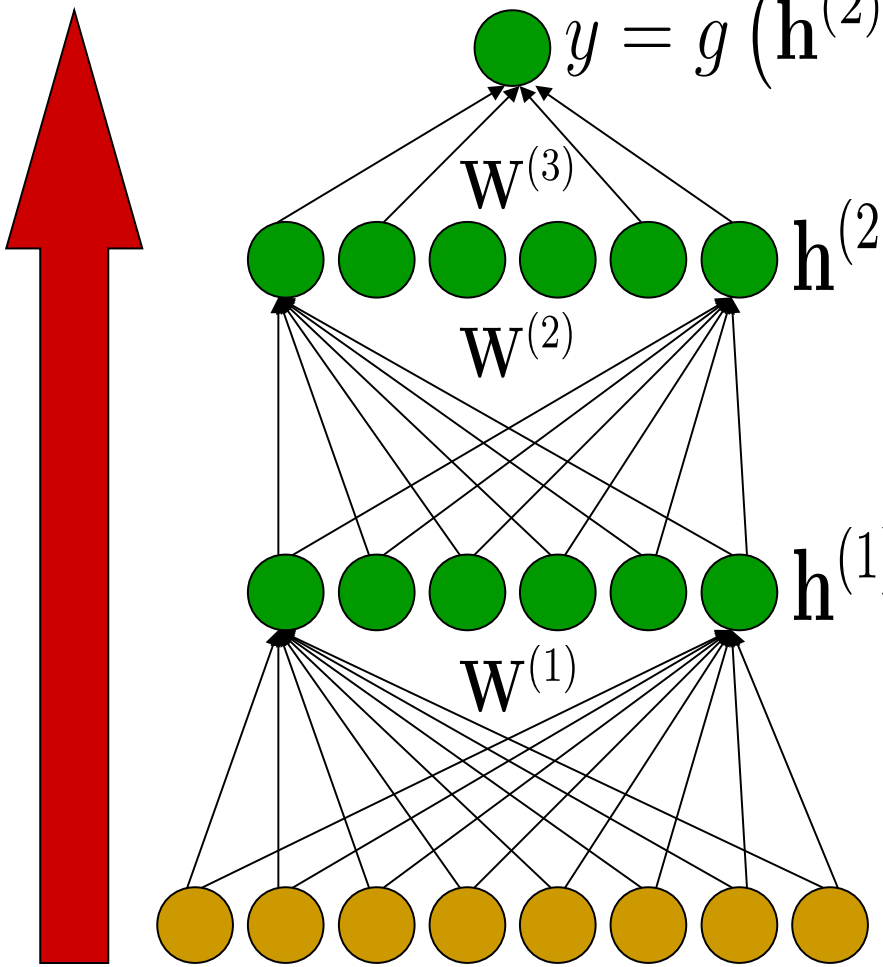
Uczenie nadzorowane

Dany jest zbiór obserwacji. Dla każdej obserwacji znamy również wartość pewnej cechy (etykieta, wartość przewidywana):

$$\mathcal{X} = \left\{ \left(\mathbf{v}^1, t^1 \right), \left(\mathbf{v}^2, t^2 \right), \ldots, \left(\mathbf{v}^n, t^n \right) \right\}$$

Celem uczenia jest zbudowanie modelu, który na podstawie informacji zawartej w zbiorze znanych obserwacji i etykiet będzie przewidywał etykiety dla nowych obserwacji (z tego samego rozkładu).

Perceptron wielowarstwowy



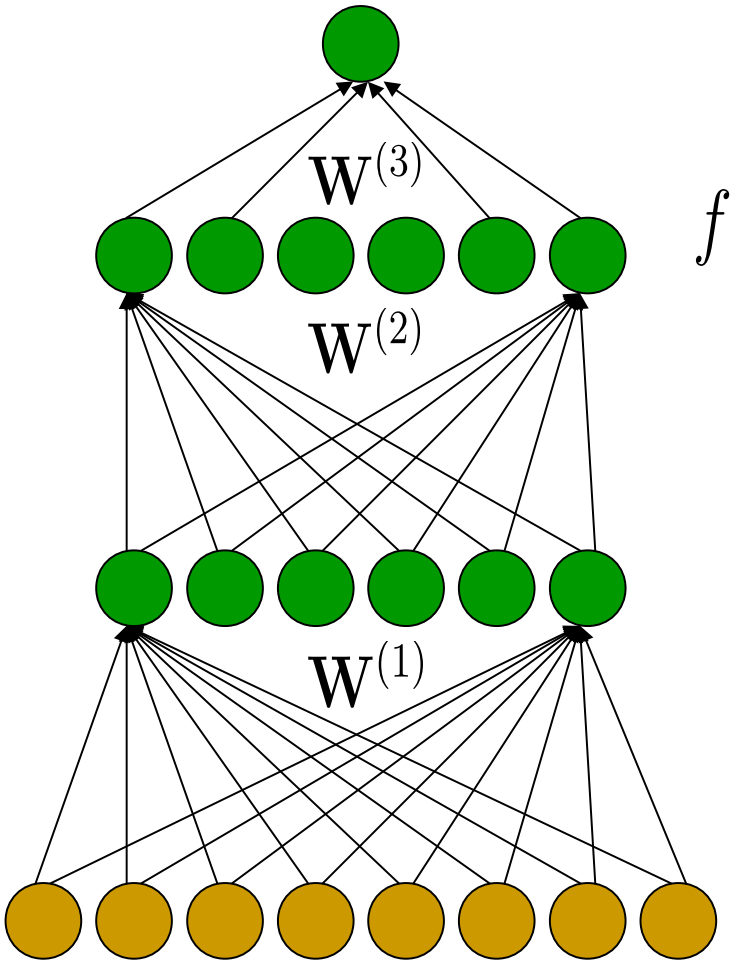
$y = g\left(\mathbf{h}^{(2)}\mathbf{W}^{(3)}\right)$ Wyjście (odpowiedź) sieci

$\mathbf{h}^{(2)} = f\left(\mathbf{h}^{(1)}\mathbf{W}^{(2)}\right)$ Aktywacje ukryte

$\mathbf{h}^{(1)} = f\left(\mathbf{v}\mathbf{W}^{(1)}\right)$ Aktywacje ukryte

$\mathbf{v}$ Obserwacja

Perceptron wielowarstwowy

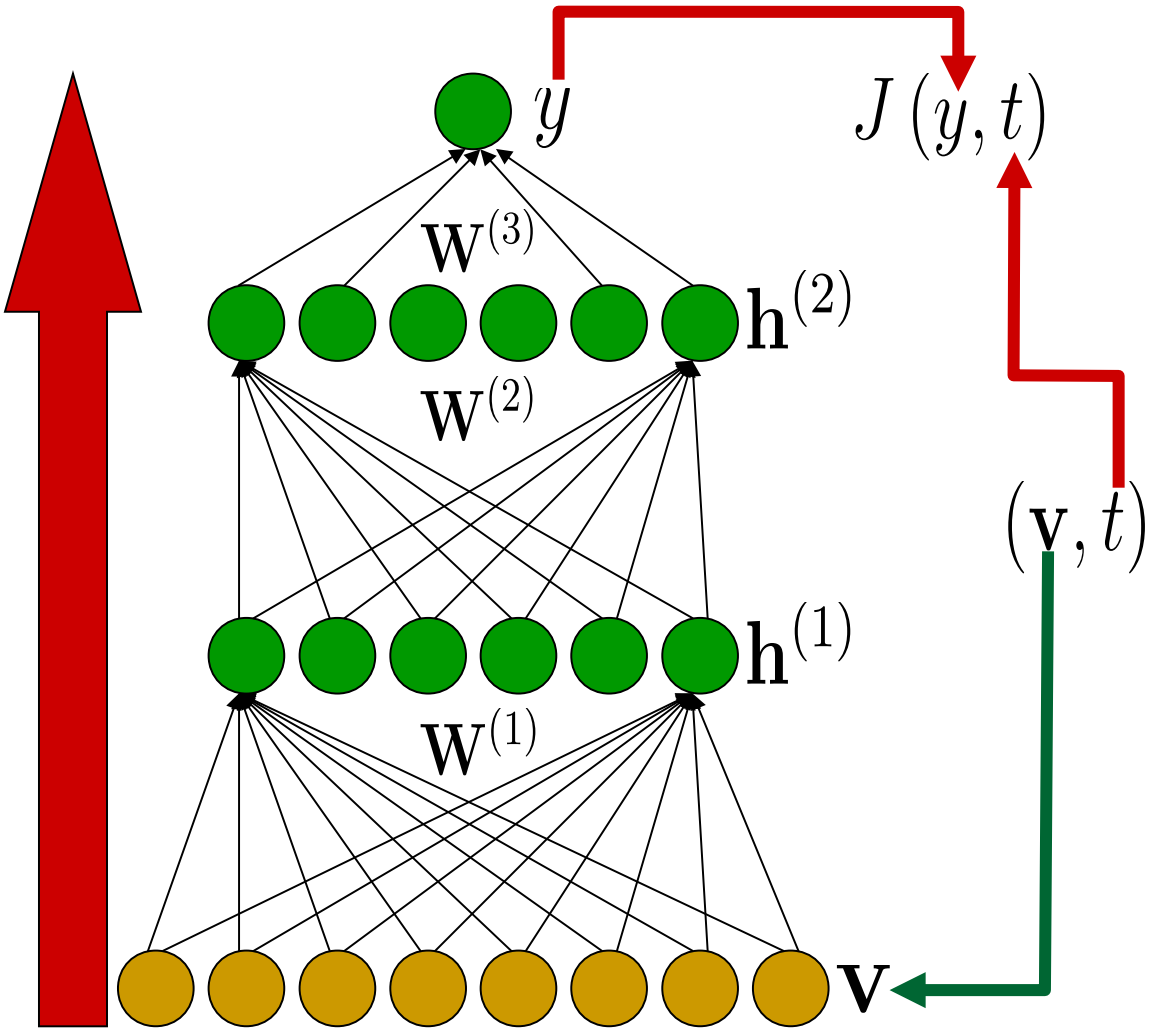


$f(\cdot),\ g(\cdot)$ – funkcje aktywacji.

- Z reguły nieliniowe.
- Na przykład funkcja sigmoidalna (logistyczna):

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

Algorytm wstecznej propagacji błędu

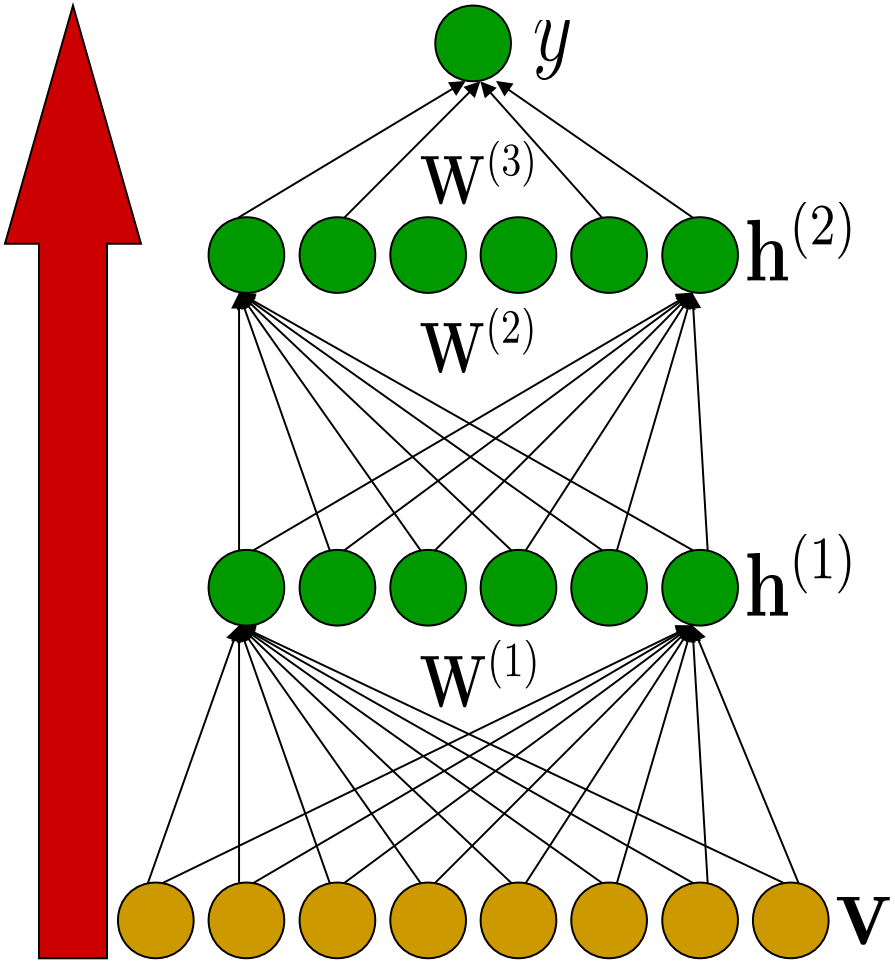


Wprowadzamy **funkcję kosztu:**

$$J(y,t)$$

Funkcja ta opisuje koszt (stratę) jaką ponosimy ze względu na różnicę pomiędzy odpowiedzią sieci a wartością przewidywaną.

Algorytm wstecznej propagacji błędu



Wprowadzamy **funkcję kosztu:**

$$J(y,t)$$

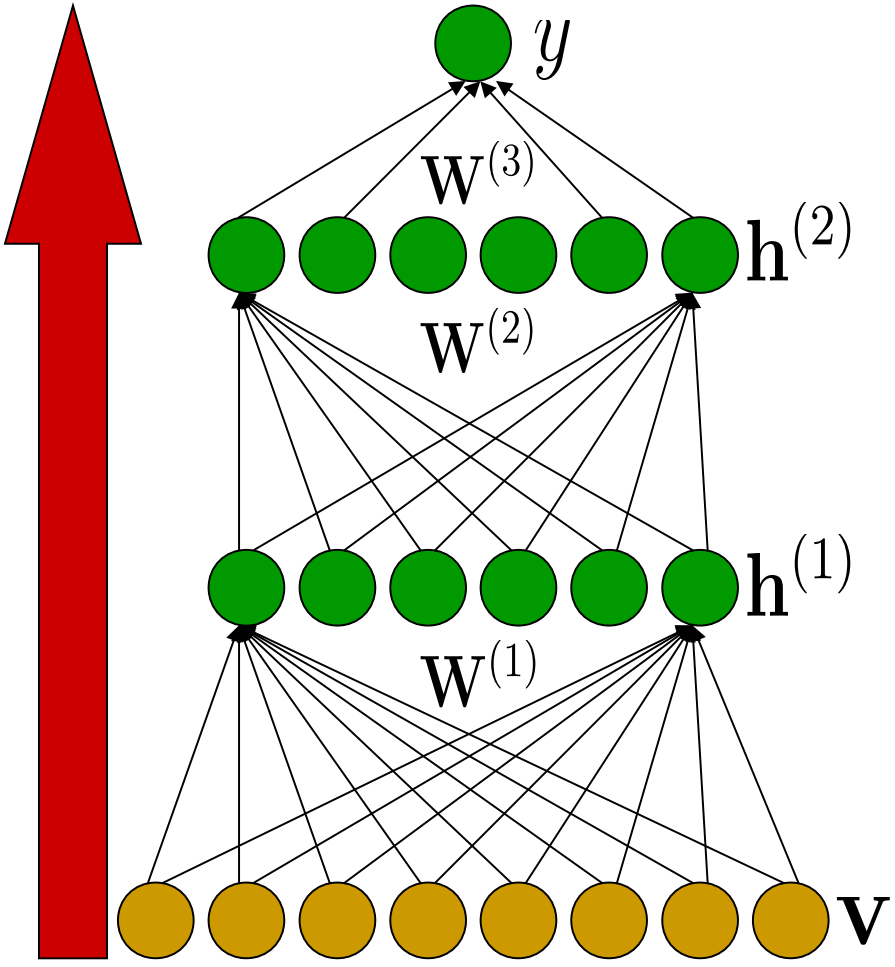
Jaką postać może mieć funkcja kosztu?

Szukamy funkcji różniczkowalnej, rosnącej gdy rośnie różnica pomiędzy odpowiedzią sieci a wartością przewidywaną.

Na przykład:

$$J(y,t) = \frac{1}{2}(t-y)^2$$

Algorytm wstecznej propagacji błędu



Wprowadzamy **funkcję kosztu:**

$$J(y, t)$$

Celem uczenia będzie minimalizacja kosztu (na zbiorze uczącym) ze względu na parametry modelu (wagi sieci).

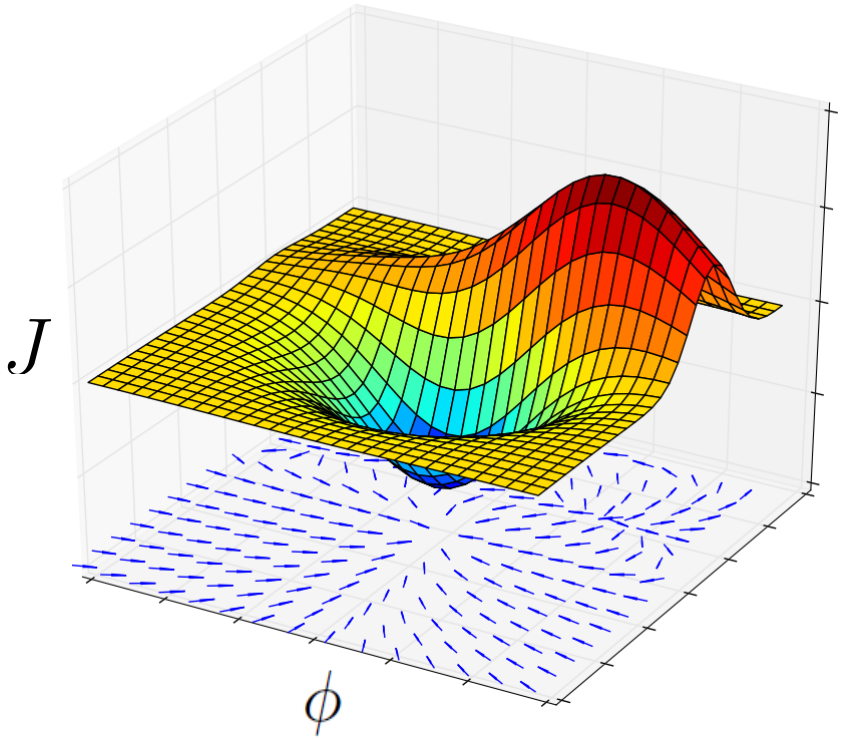
Kosztu nie możemy zminimalizować analitycznie.

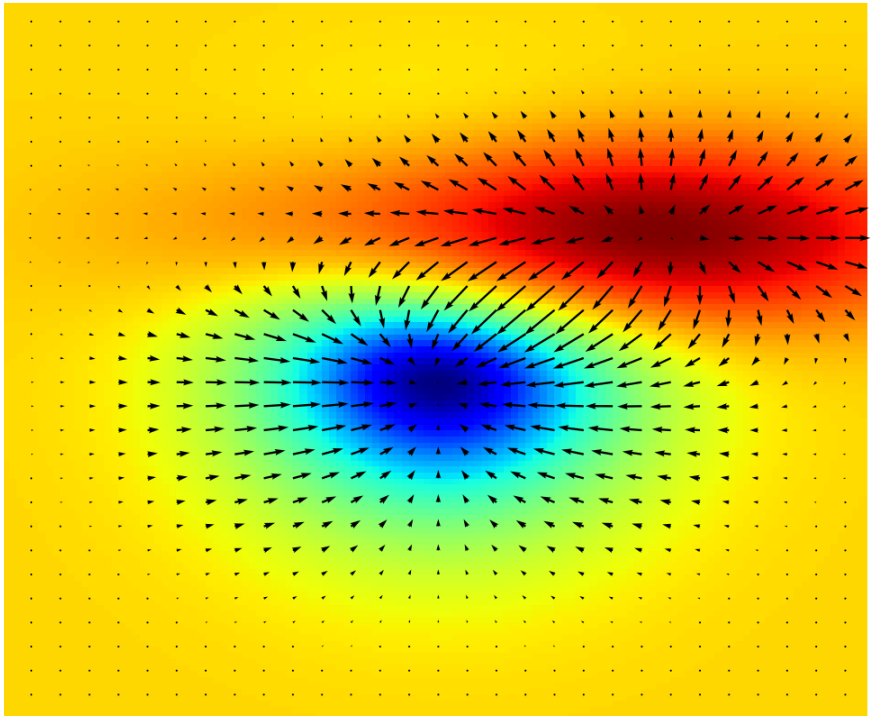
Będziemy więc uczyć stochastycznym spadkiem wzdłuż gradientu.

Metoda spadku wzdłuż gradientu

$$\arg\min_{\phi} J\left(y, t\,;\phi\right)$$

Minimum możemy znaleźć metodą spadku wzdłuż gradientu (*gradient descent*).

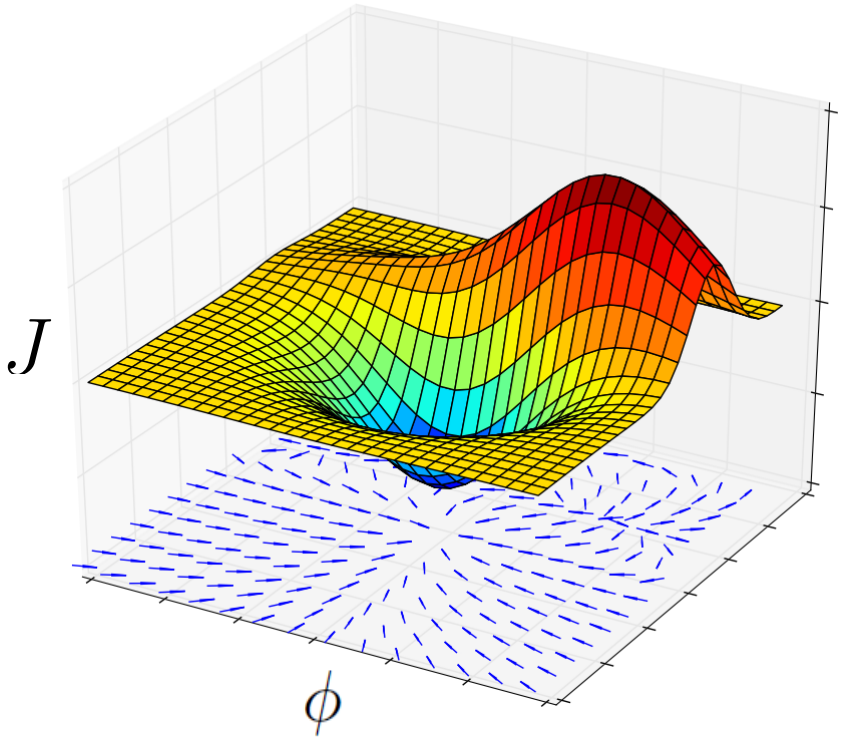




Metoda spadku wzdłuż gradientu

$$\arg\min_{\phi} J\left(y, t\,;\phi\right)$$

Minimum możemy znaleźć metodą spadku wzdłuż gradientu (*gradient descent).*



1. Wybierz parametry początkowe: ϕ_0
2. Powtarzaj:

$$\phi_{t+1} \leftarrow \phi_t - \epsilon \nabla_J \left(\phi_t\right)$$

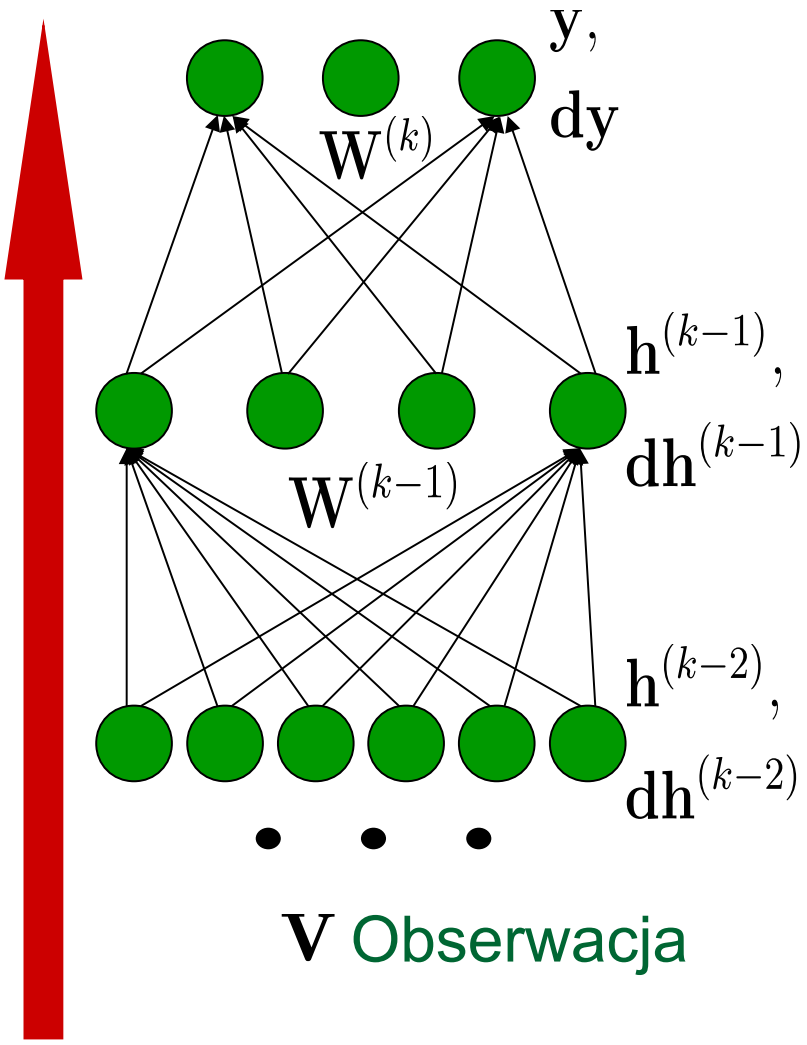
aż model zostanie wyuczony.

ϵ – stała ucząca (*learning rate*).

Algorytm wstecznej propagacji błędu

Faza propagacji sygnału
(*forward pass*)

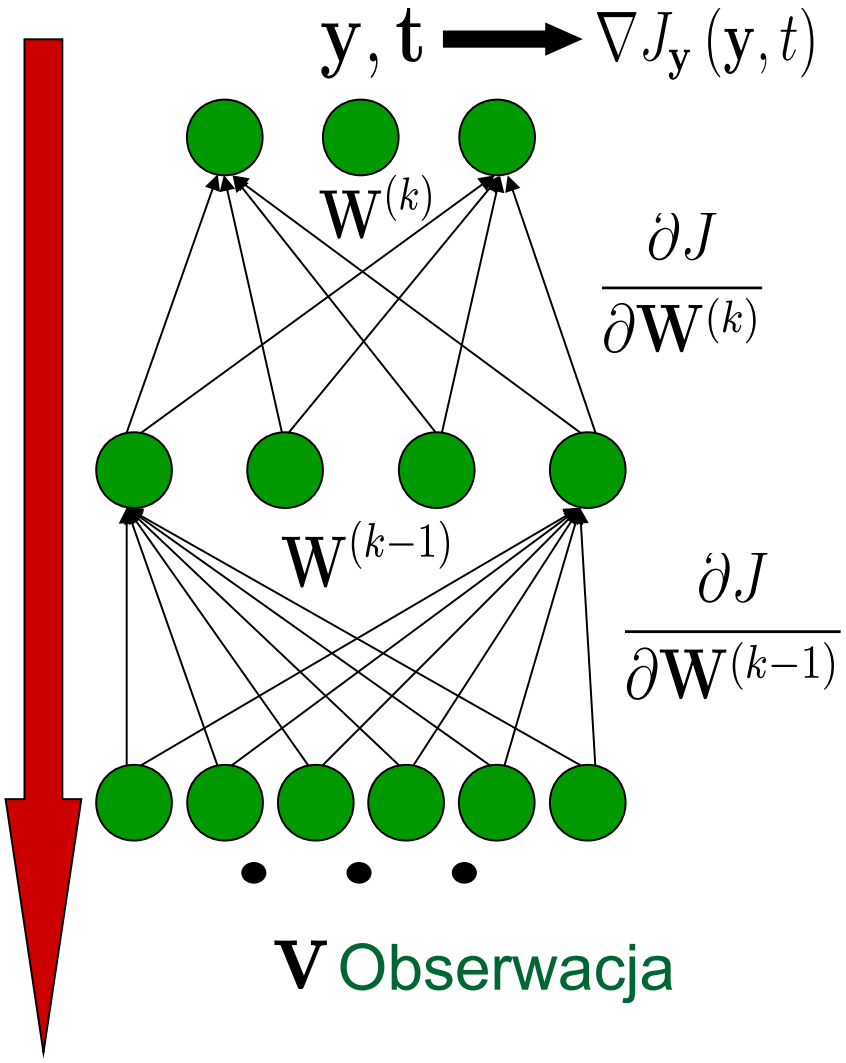
1. Wybierz obserwację ze zbioru uczącego.
2. Wyznacz aktywacje i pochodne aktywacji w sieci.



Algorytm wstecznej propagacji błędu

Wsteczna propagacja błędu
(*error backpropagation*)

1. Wyznacz gradient funkcji kosztu po odpowiedziach sieci.
2. Wyznacz pochodne kosztu po aktywacjach neuronów i pochodne kosztu po wagach.



Algorytm wstecznej propagacji błędu

Wsteczna propagacja błędu (*error backpropagation*)

1. Wyznacz gradient funkcji kosztu po odpowiedziach sieci.
2. Wyznacz pochodne kosztu po aktywacjach neuronów i pochodne kosztu po wagach.

Pochodne:

$$\frac{\partial J}{\partial \mathbf{W}^{(k)}}, \frac{\partial J}{\partial \mathbf{W}^{(k-1)}}, \ldots, \frac{\partial J}{\partial \mathbf{W}^{(1)}}$$

użyj do minimalizacji funkcji kosztu (na przykład stochastycznym spadkiem gradientu).