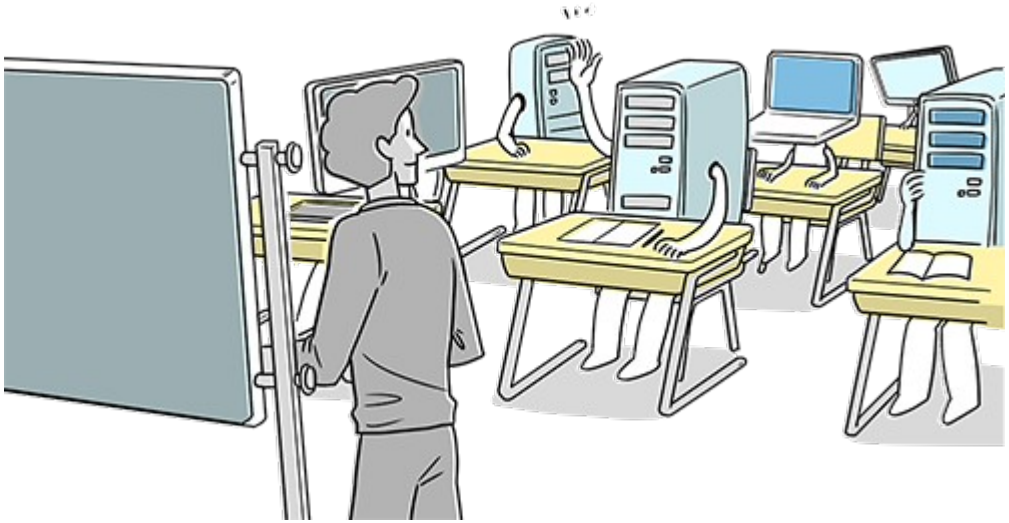
Deep Learning
wykład 1



Marcin Wolter

IFJ PAN

18 listopada 2020

- Mój adres e-mail: *marcin.wolter@ifj.edu.pl*
  marcin.wolter@ujk.edu.pl

- Slajdy, programy zamieszczam na Microsoft Teams oraz w serwisie *github.com*
- Pierwszy raz prowadzę zajęcia na platformie Teams, proszę o wyrozumiałość.

Program wykładu

- Płytkie sieci neuronowe (przypomnienie) - shallow neural networks

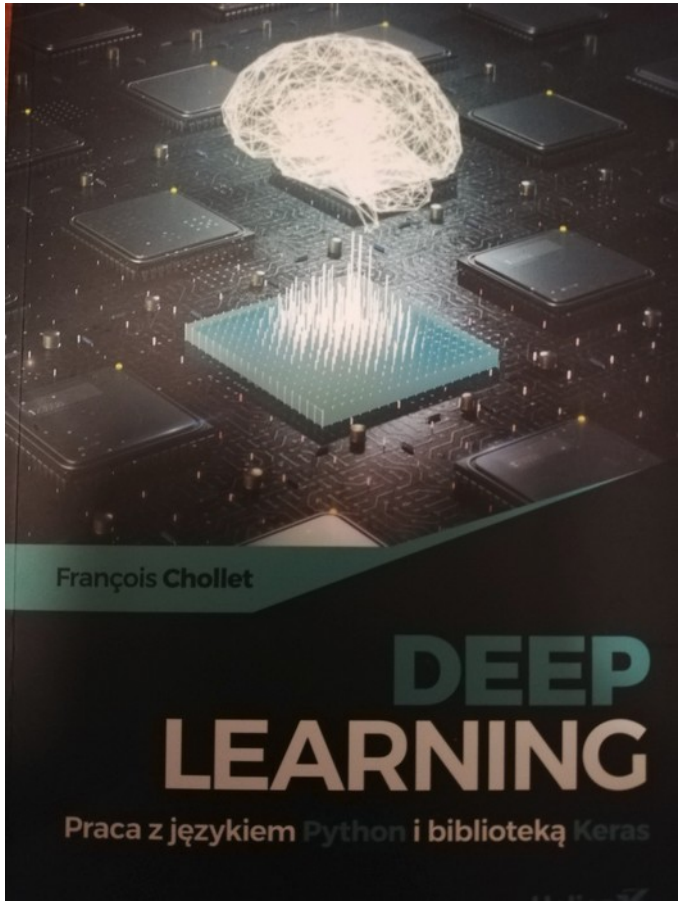
  Pakiet Scikits Learn - tutorial

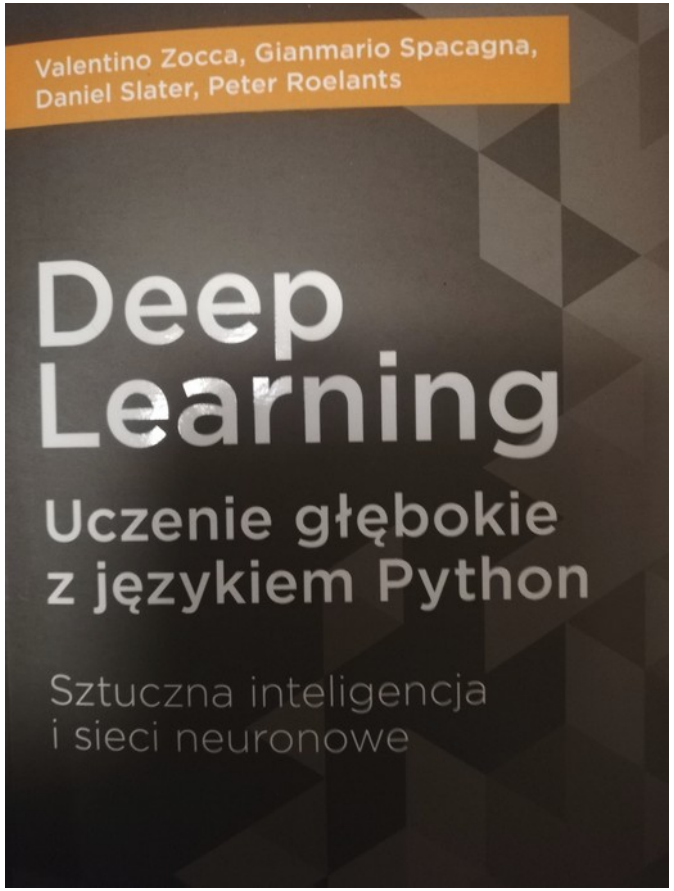
- Głębokie sieci neuronowe - deep neural networks

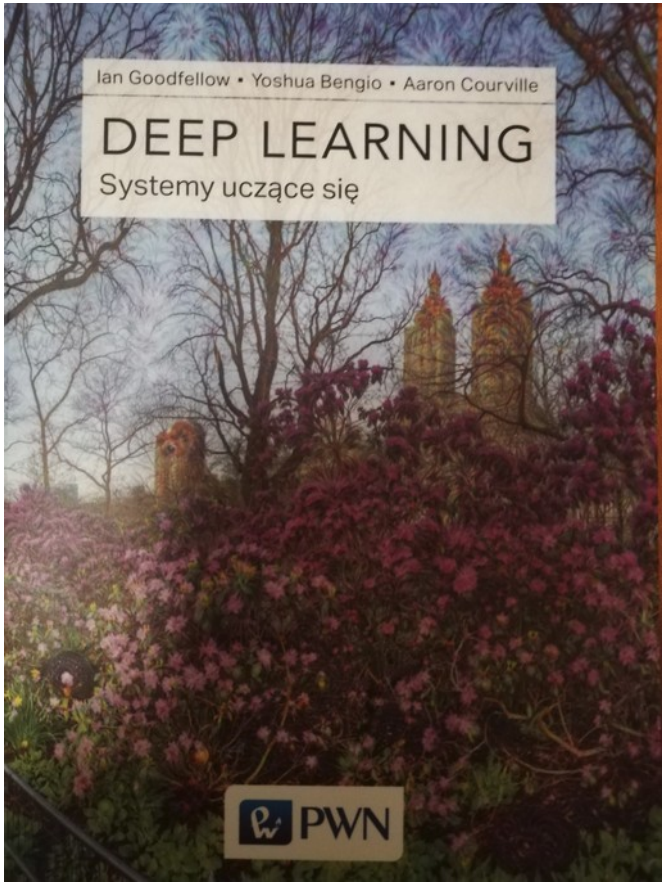
  Pakiety Keras + Tensorflow – tutorial

- Trening sieci neuronowej: próbki treningowa, walidacyjna i testowa. Przeuczenie, optymalizacja hyperparametrów
- Konwolucyjne sieci neuronowe - Convolutional Deep Neural Network
- Cross-validation
- Bayesowskie sieci neuronowe oraz tzw. Mixed Density Networks
- Sieci generacyjne - Generative Adversarial Networks (GANs)
- Ewentualnie jeszcze coś... np. sieci rekurencyjne LSTM (Long Short Term Memory) I analiza/generowanie tekstu.

Materiały







A także, a nawet przede wszystkim, blogi, tutoriale, artykuły i przykłady z internetu

Zaliczenie i praca na zajęciach

- Zadania z ćwiczeń 33%

  – proszę umieszczać je na github.com i wysyłać link

- Projekt na zakończenie kursu 66%

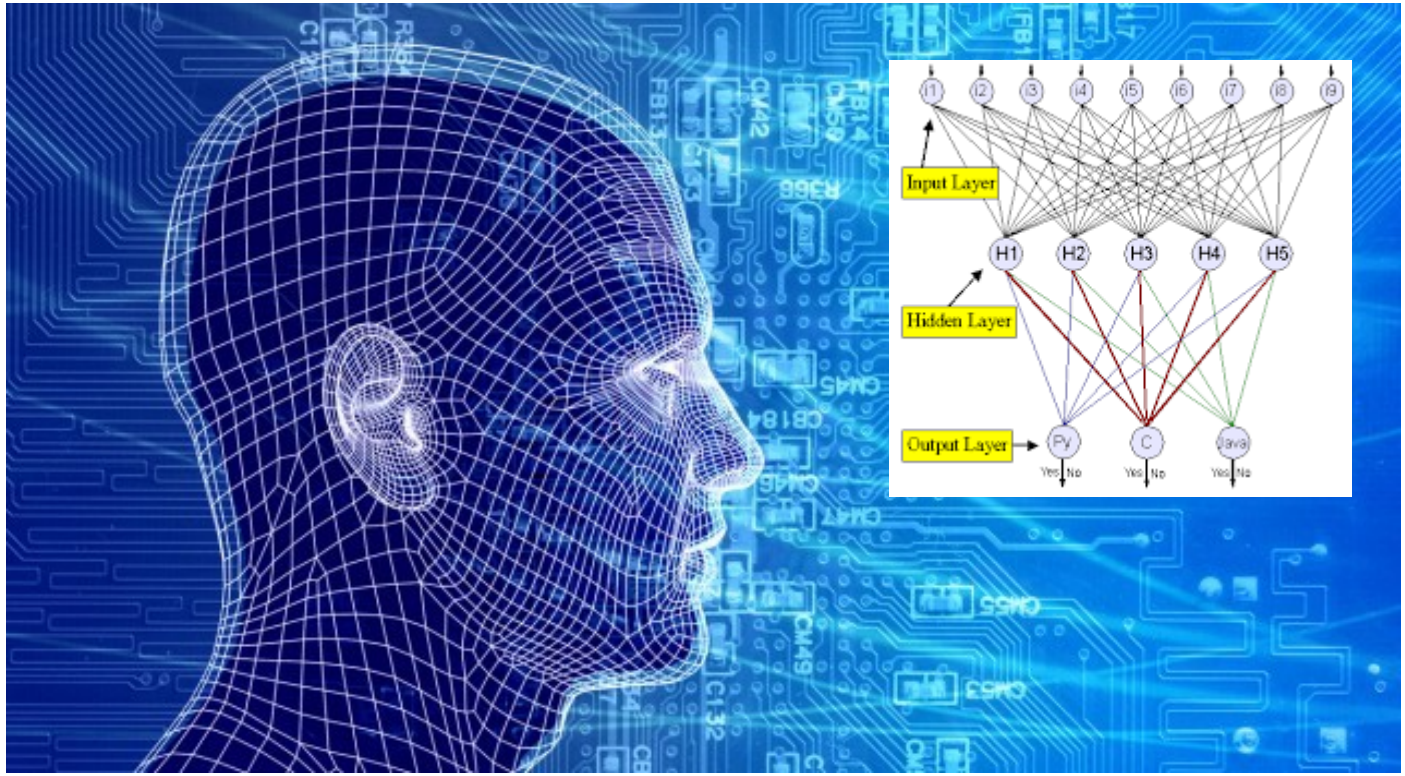
  - prosty projekt związany z głębokim uczeniem i prezentacja na forum grupy

Praca na zajęciach:

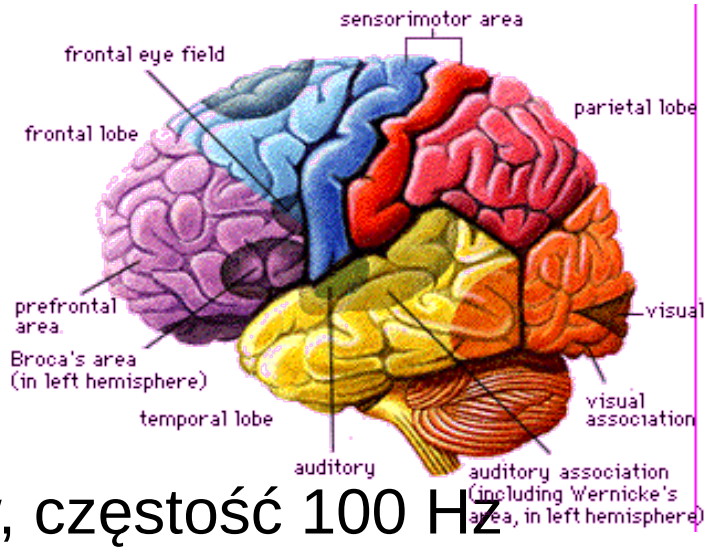
Oprócz części teoretycznej chciałbym, abyśmy sporą część czasu podczas wykładu przeznaczyli na wspólne pisanie programów. W czasie ćwiczeń spróbują Państwo samodzielnie napisać podobne.

Płytkie sieci neuronowe
Shallow neural networks

Przypomnienie wykładu prof.. Broniowskiego



Inspiracja działaniem mózgu



- **Mózg człowieka:**
  - 10^{14} neuronów, częstość 100 Hz
  - równoległe procesowanie danych (kompleksowe rozpoznanie obrazu w 100 ms – tylko 10 kroków!!!)
  - uczy się na przykładach
  - odporny na błędy i częściowe uszkodzenia
- **Sieć neuronowa:**
  - tylko algorytm, niekoniecznie odzwierciedlający działanie mózgu.

Historia

1938 N. Rashevsky, neurodynamika - sieci neuronowe jako układy dynamiczne, sieci rekurencyjne.

1943 W. McCulloch, W. Pitts, sieci neuronowe=układy logiczne

1958 F. Rosenblatt, perceptron, sieć jako funkcja;

1973 Chr. von der Malsburg, samoorganizacja w mózgu;

1982 Kohonen, Self-Organizing Maps

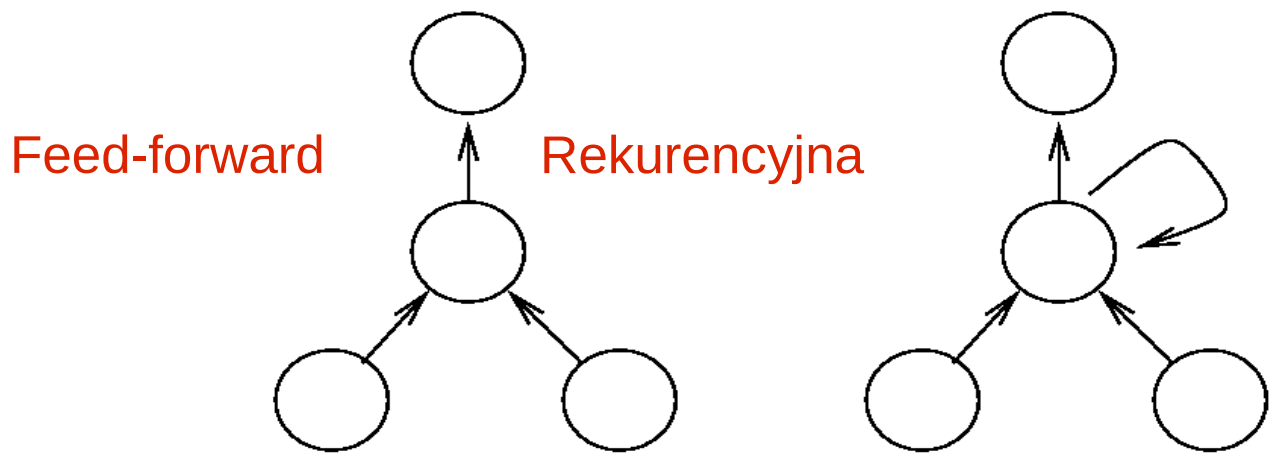
...

1986 **wsteczna propagacja błędów; liczne zastosowania.**

2010 (około) głębokie sieci neuronowe

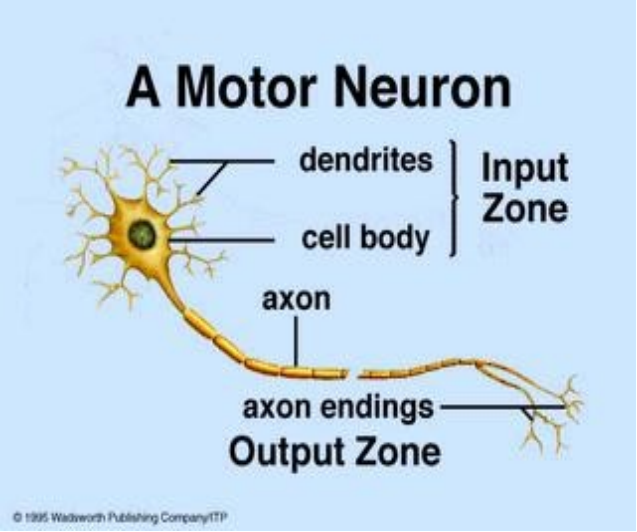
Co to jest sieć neuronowa?

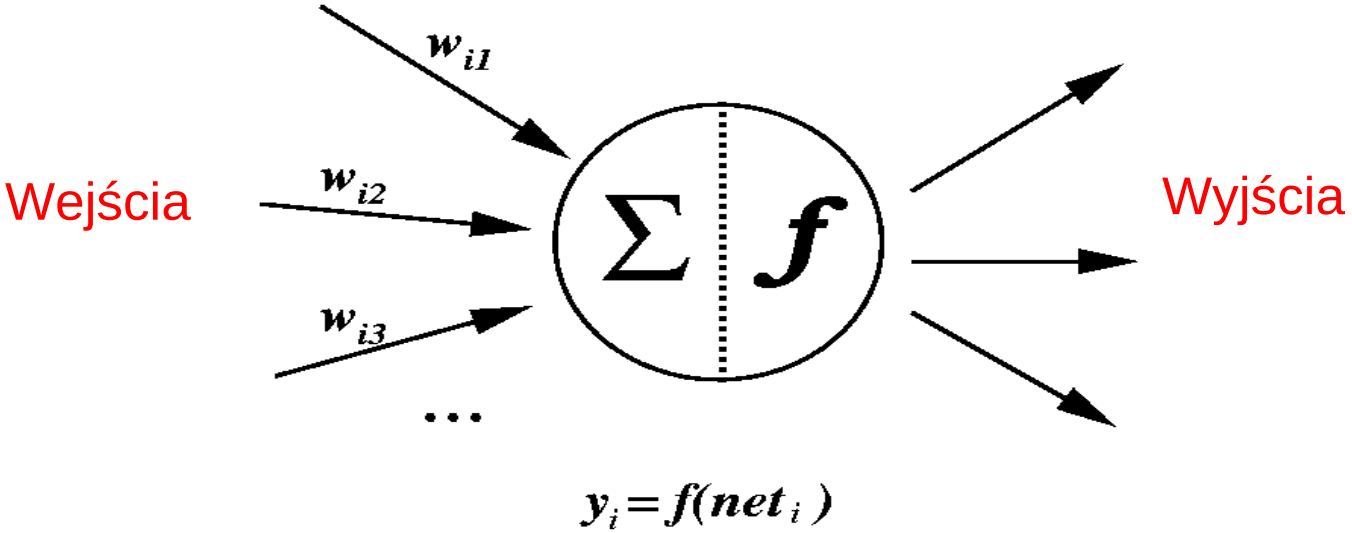
- Sieć neuronowa - model matematyczny będący złożeniem wielu funkcji (przeważnie nieliniowych)
- **Zadania:**
  - **Klasyfikacja przypadków** – np. odróżnienie sygnału od tła
  - **Regresja** – aproksymuje funkcję rzeczywistą
- **Dwa rodzaje sieci:**
  - **Feed forward** – informacja przesyłana od wejścia do wyjścia bez sprzężeń zwrotnych
  - Rekurencyjne – pętle rekurencyjne.
- **Uczenie:**
  - z nauczycielem
  - bez nauczyciela



Neuron – podstawowa cegiełka

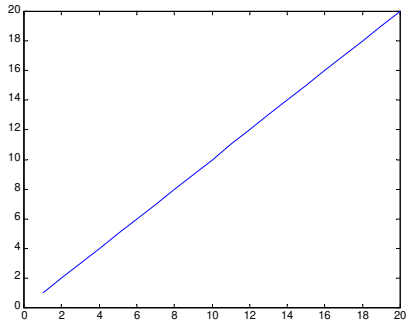
- Funkcja ważonej sumy wejść $$y_i = f\left(\sum_j w_{ij}\, y_j\right)$$





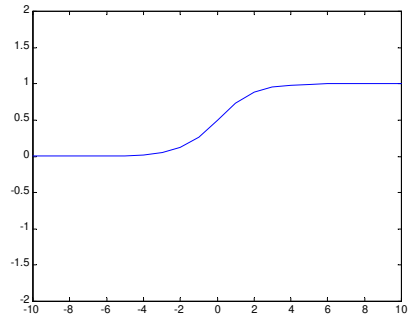
- Funkcję *f* nazywamy funkcją aktywacji

Typowe funkcje aktywacji



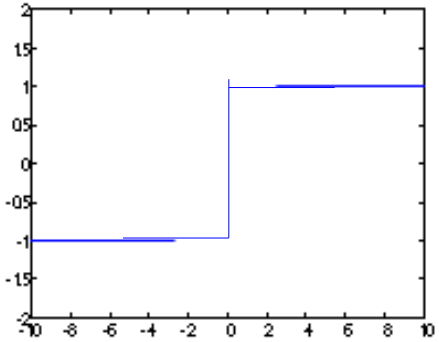
Liniowa (wtedy całą sieć nazywamy liniową)

$$y = x$$

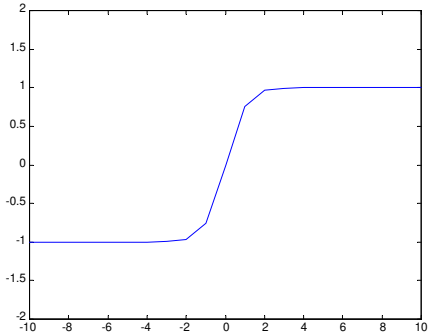


Logistic

$$y = \frac{1}{1 + \exp(-x)}$$



Progowa

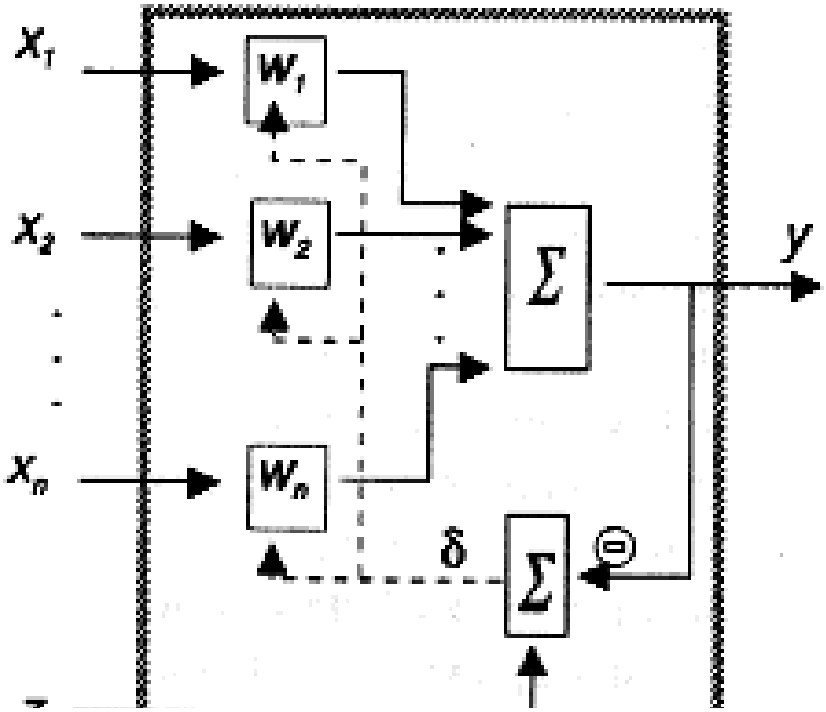


Tangens hiperboliczny

$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Uczenie pojedynczego neuronu

Neuron uczy się na przykładach
Nauka z nauczycielem – znamy poprawne odpowiedzi



ADALINE

(Adaptive Linear Network)

- X_i – dane wejściowe
- Y - wartość na wyjściu
- Z - poprawna wartość wyjściowa (uczenie z nauczycielem!)

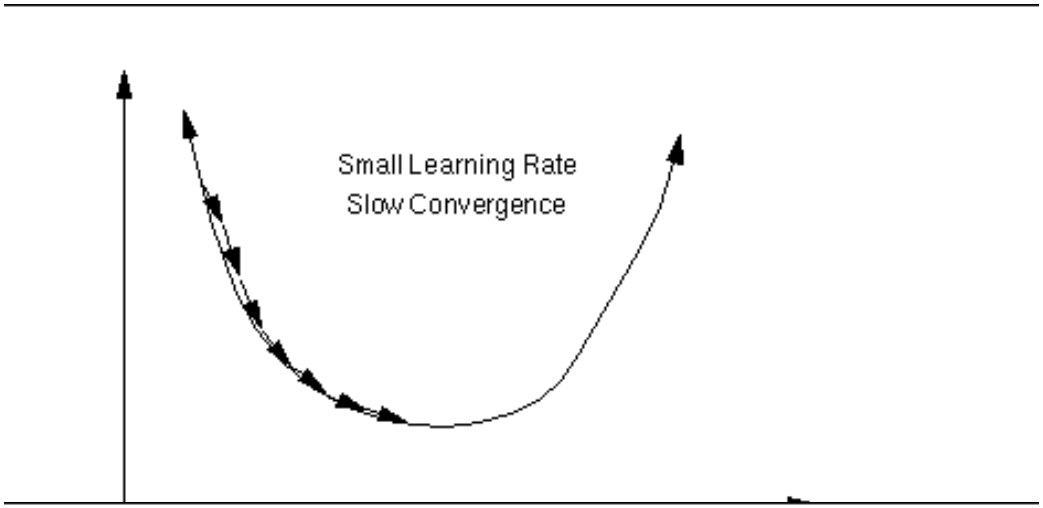
- ZADANIE – minimalizacja funkcji kary:
- Znajdź: $\chi^2 = \sum (z^{(j)} - y^{(j)})^2$
- Nowy zestaw wag:

$$\delta = z - y$$

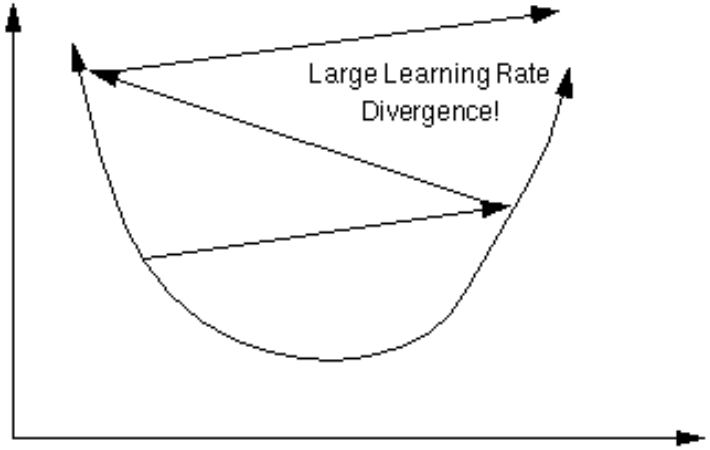
- η - szybkość uczenia

$$W' = W + \eta \cdot \delta \cdot X$$

Szybkość uczenia



- Za mała

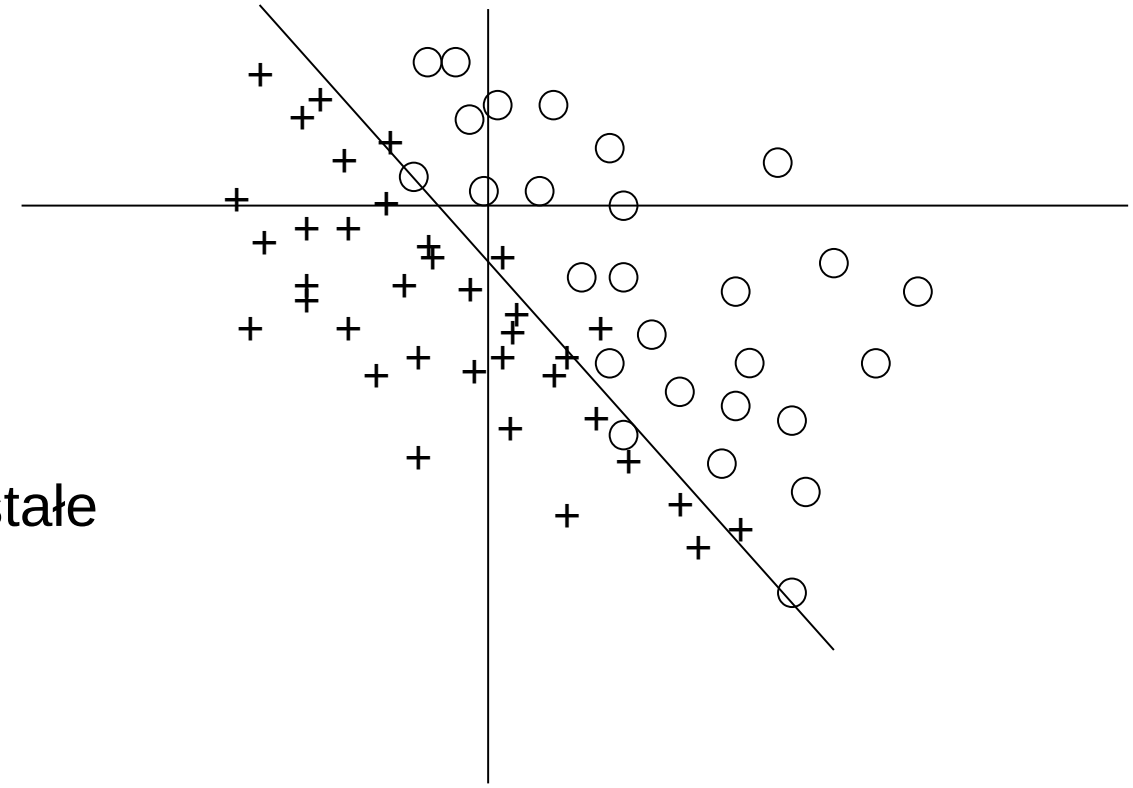


- Za duża

Co potrafi pojedynczy neuron (perceptron)

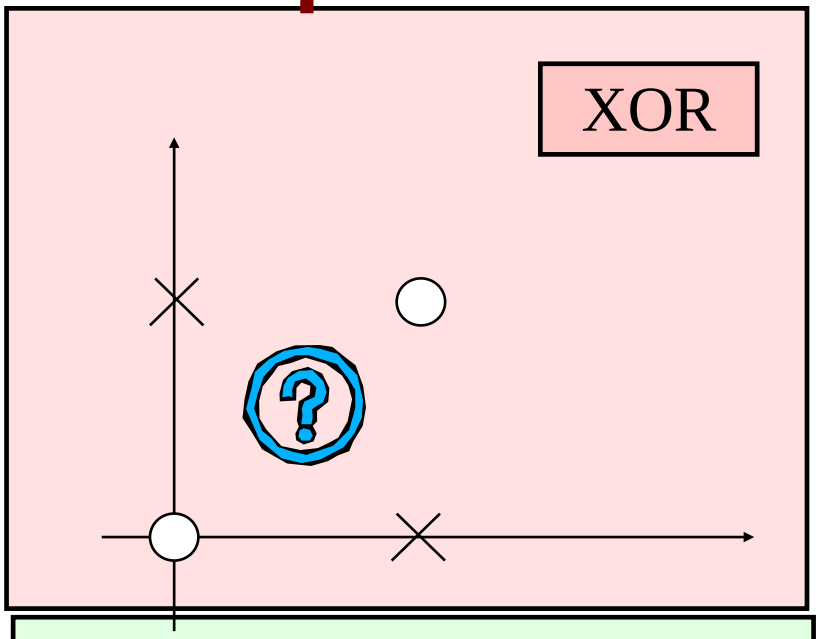
- Perceptron (z progową funkcją aktywacji) może dzielić płaszczyznę za pomocą linii (ogólnie: hiperpłaszczyzny w przestrzeni n-wymiarowej).

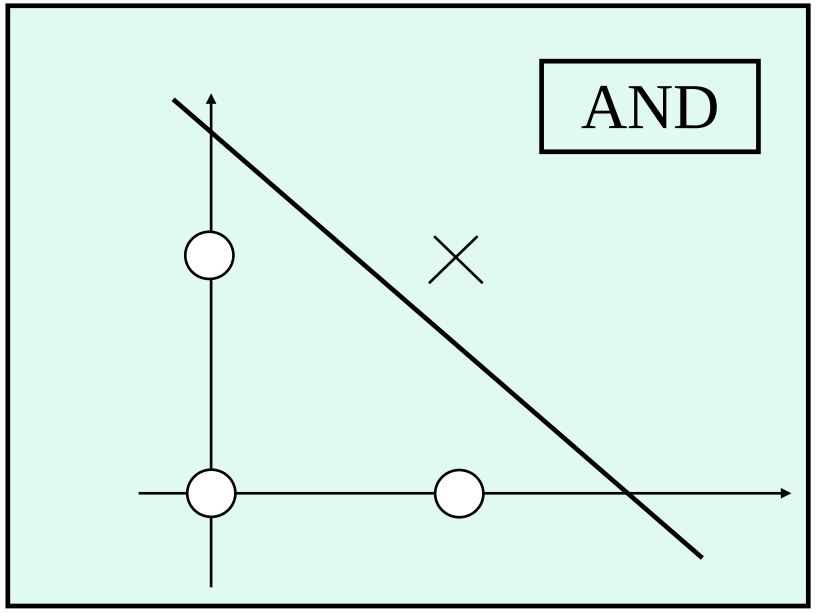
- Punkty leżące nad ową prostą klasyfikujemy jako 1, zaś pozostałe jako 0.

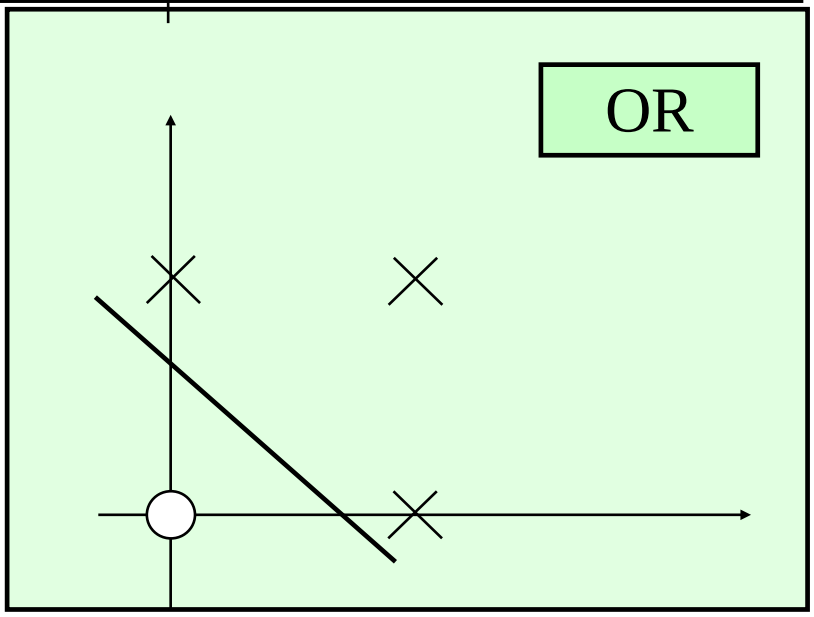


Czego perceptron nie potrafi?

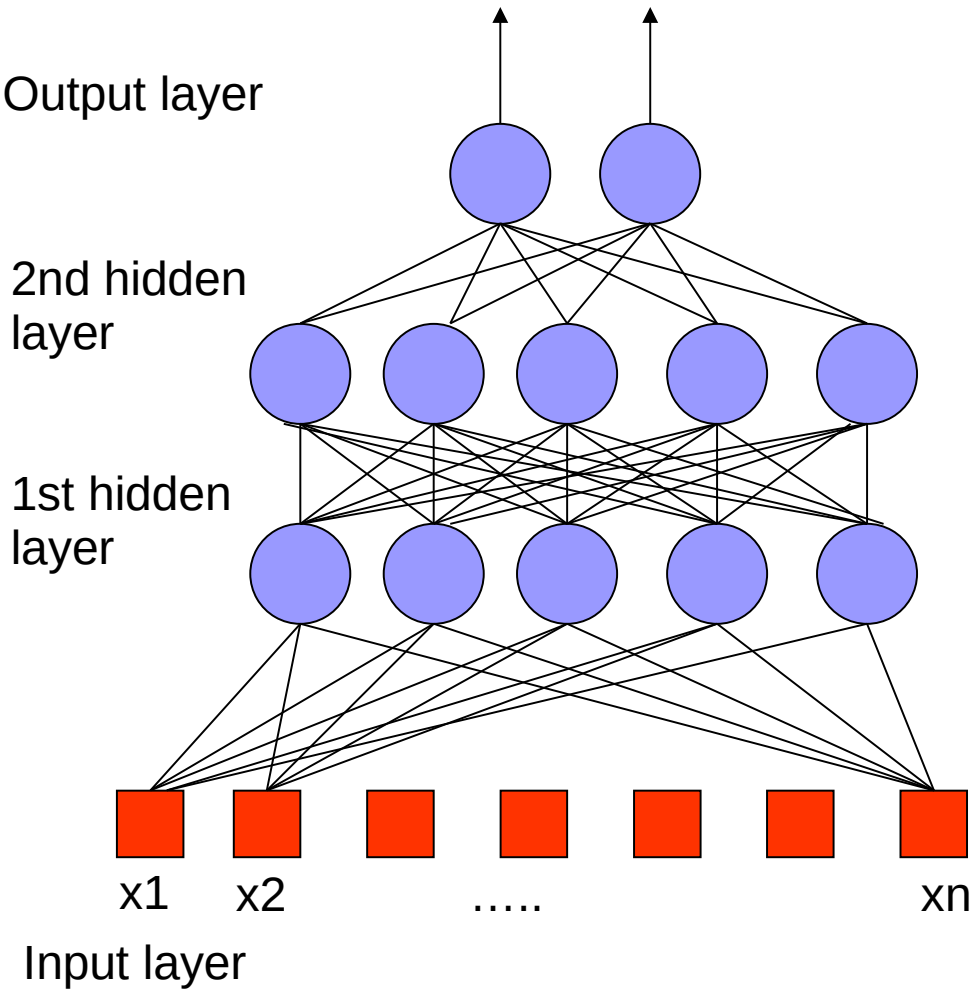
- Pojedynczy perceptron nie potrafi odróżniać zbiorów nieseparowalnych liniowo, np. funkcji XOR.
- Odkrycie tych ograniczeń (1969) na wiele lat zahamowało rozwój sieci neuronowych.







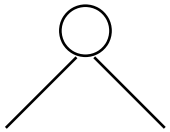
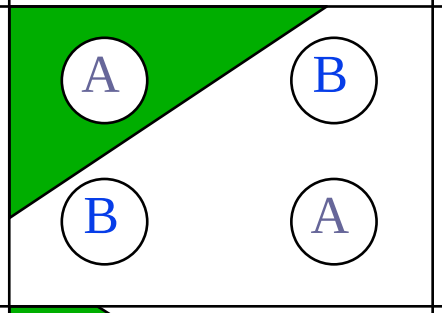
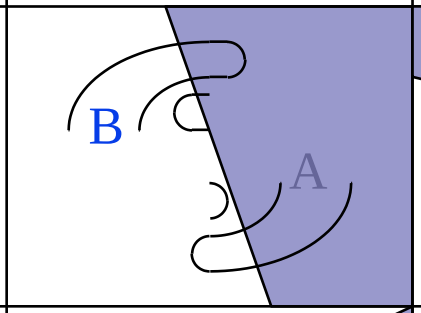
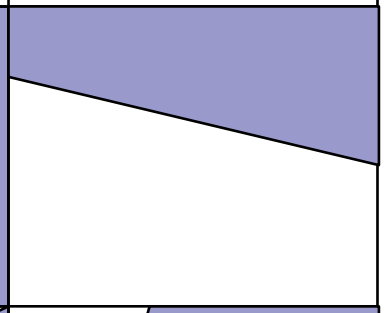
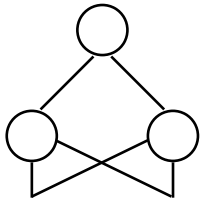
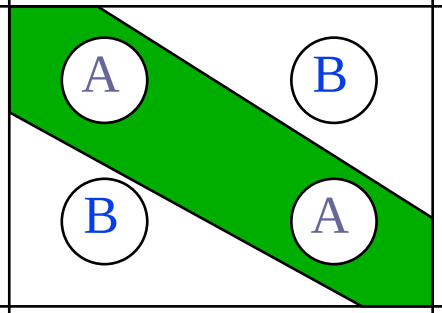
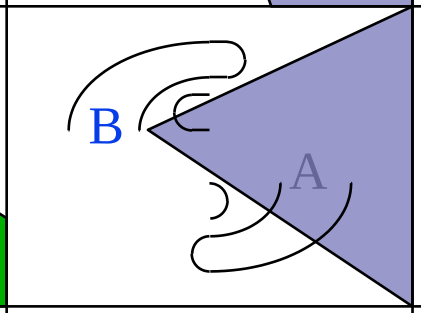
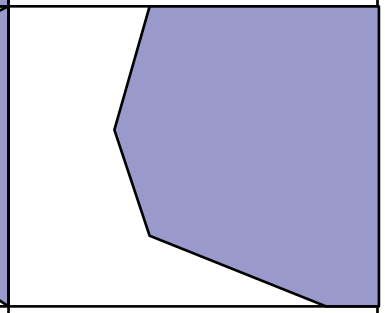
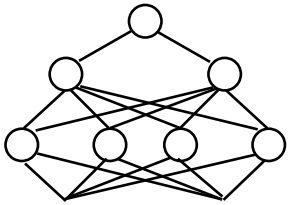
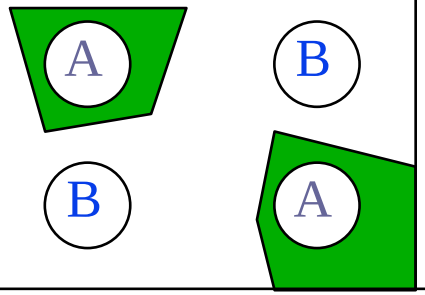
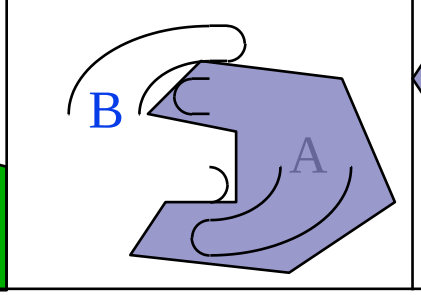
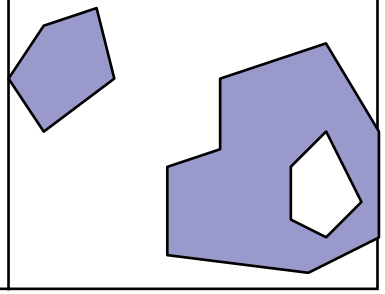
Może więc sieć neuronów?



- Informacja propaguje się od wejścia do wyjścia
- Sieć jest złożeniem wielu funkcji aktywacji (w ogólności nieliniowych)
- Odpowiednio złożona sieć może odtworzyć dowolną funkcję.

Co potrafi sieć neuronów

(progowa funkcja aktywacji)

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Abitrary (Complexity Limited by No. of Nodes)			

Jak uczyć sieć wielowarstwową?

- Minimalizacja *funkcji ryzyka* ze względu na zestaw wag ω:

$$R(\omega) = \frac{1}{N} \sum_i [t_i - n(x_i, \omega)]^2$$

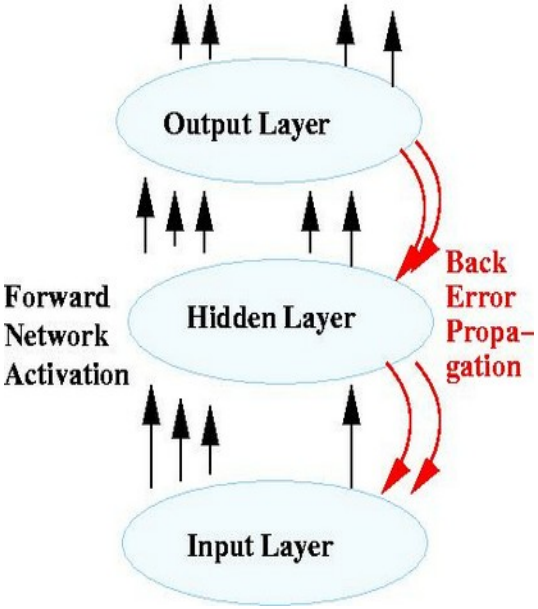
- Problem – jak korygować wagi w głębszych warstwach sieci porównując tylko wartości na płaszczyźnie wyjściowej.
- To pytanie wstrzymało rozwój sieci neuronowych na 30 lat, aż do lat 80-tych.
- Rozwiązanie - metoda propagacji do tyłu **(backpropagation)**. Błąd $\delta = t - n(x, \omega)$ jest propagowany wstecz poprzez sieć z użyciem aktualnych wag.

Typowa procedura uczenia

- Dwa zbiory danych: do nauki i do sprawdzenia.
- $\chi^2=\Sigma(z-y)^2$ obliczane jest dla obydwu zbiorów i porównywane, aby zapobiegać ***przeuczeniu.***
- ***propagacja do tyłu (backpropagation).*** Różnica między wartością oczekiwaną a otrzymaną na wyjściu y-*f(x,w)* jest propagowana wstecznie przez sieć używając aktualnych wag. Zmiana wagi:

$$dw_{ij} = \rho x_i (t_j - y_j),$$

- gdzie ρ szybkość uczenia, t_j prawdziwa wartość wyjściowa na węźle j, y_j obliczoną przez sieć, a x_i jest aktualną wartością na węźle *i* w płaszczyźnie poprzedzającej warstwę wyjściową.

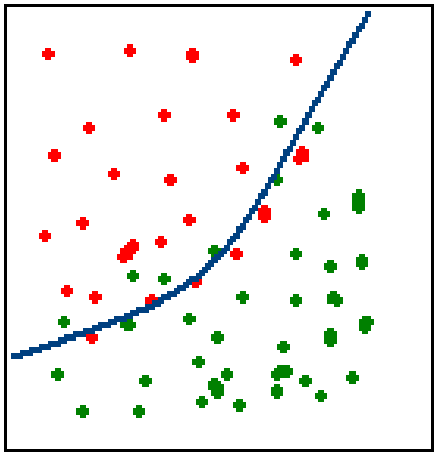


Znajdowanie minimum

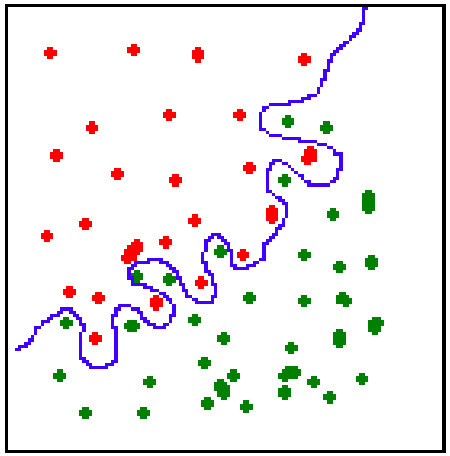
- Nie ma pewności, czy znaleźliśmy lokalne czy globalne minimum funkcji straty $\chi^2=\Sigma(z-y)^2$

- Mechanizmy zapobiegające ugrzęźnięciu w lokalnym minimum:
  - Wybór przypadkowych wag początkowych, powtarzanie uczenia
  - Dodawanie szumu, aby algorytm opuścił lokalne minimum (jittering).

Przeuczenie (Overtraining)

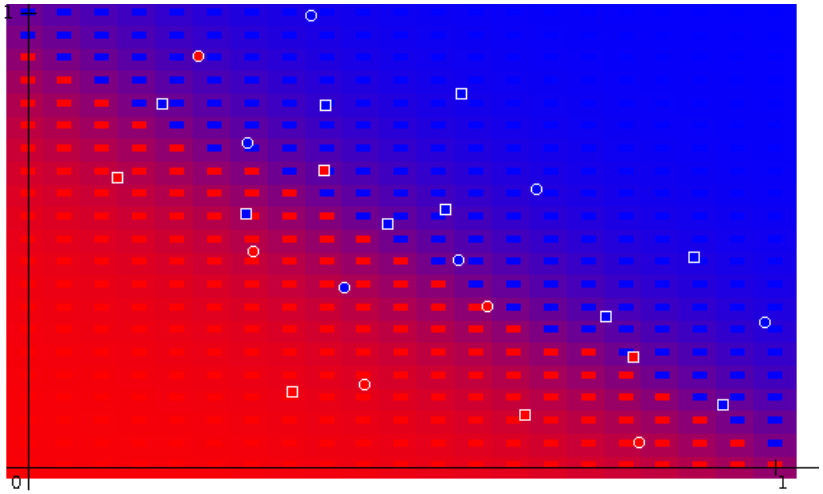


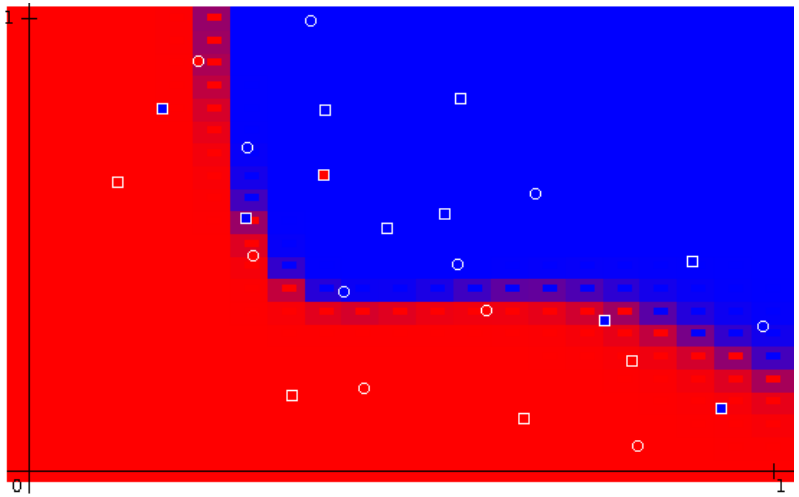
Dobrze

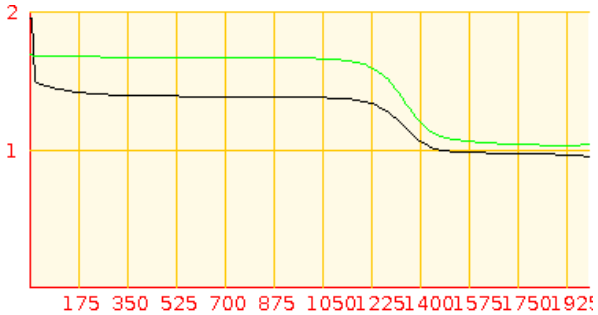


Przetrenowanie

- **Przeuczenie** – Algorytmuczy się poszczególnych przypadków, a nie ogólnych zasad.
- Efekt pojawia się we wszystkich algorytmach uczących
- Remedium – sprawdzenie na oddzielnym zbiorze danych.

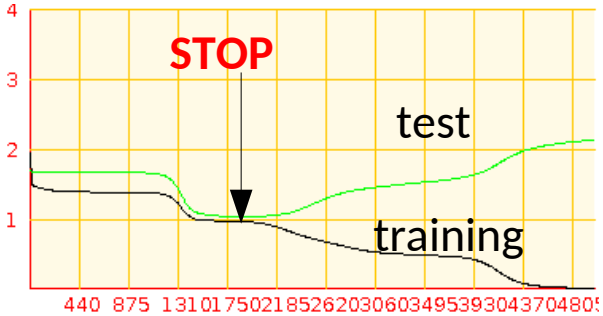






Training sample

Test sample



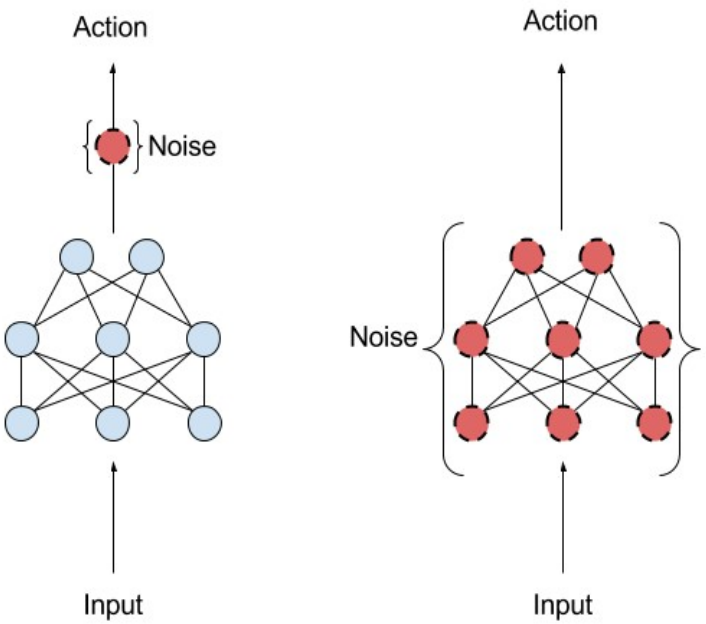
Przykład z siecią neuronową.

Znajdowanie minimum

- Nie ma pewności, czy znaleźliśmy lokalne czy globalne minimum funkcji straty $\chi^2=\Sigma(z-y)^2$

- Mechanizmy zapobiegające ugrzęźnięciu w lokalnym minimum:
  - Wybór przypadkowych wag początkowych, powtarzanie uczenia
  - Dodawanie szumu, aby algorytm opuścił lokalne minimum (jittering).

- Regularyzacja – obydwa algorytmy MLPRegressor i MLPClassifier używają parametru regularyzacji “alpha”, który pomaga zredukować przetrenowanie “karząc” wagi o dużej wartości.



Action-Space-Noise (left) and Parameter-Space-Noise (right)

https://matthiasplappert.com/publications/2017_Plappert_Master-thesis.pdf

Algorytmy znajdujące minimum w SKLEARN

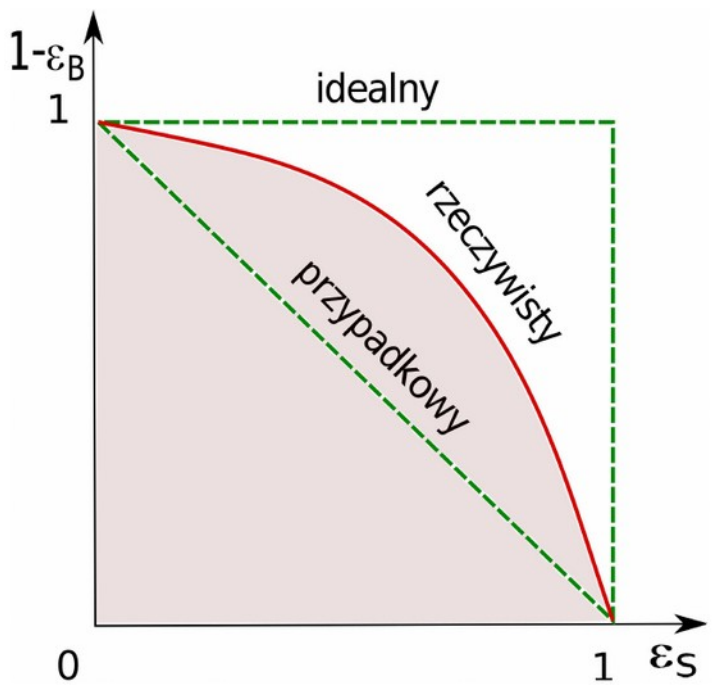
- **Stochastic Gradient Descent (SGD)** - updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation, i.e.

$$w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$$

  where η is the learning rate which controls the step-size in the parameter space search.
- **Adam** - similar to SGD, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments (default).
- **L-BFGS** - approximates the Hessian matrix which represents the second-order partial derivative of a function. Further it approximates the inverse of the Hessian matrix to perform parameter updates. Works well for small datasets.

Krzywa ROC, czyli jak określić jakość klasyfikatora

- ROC (Receiver Operation Characteristic) używana była pierwotnie do kalibracji radarów.
- Pokazuje odrzucanie tła ($1-\varepsilon_B$) vs wydajność identyfikacji sygnału ε_S. Pokazuje jakość klasyfikatora
- AUC (Area Under Curve) – powierzchnia pod krzywą może być używana jako miara jakości klasyfikatora:



AUC = ½ – random

AUC = 1 - ideal

Trochę programowania

- Weźmy przykład z wykładu o regresji z pomocą sieci neuronowych i powtórzmy go z pomocą pakietu Scikit Learn (tak jak zrobiliśmy to z klasyfikacją podczas ćwiczeń:

  https://github.com/marcinwolter/DeepLearning_2020/blob/main/wyk%C5%82ad_1a.ipynb

- Oraz dodajmy krzywą ROC do przykładu z ćwiczeń:

  https://github.com/marcinwolter/DeepLearning_2020/blob/main/cwiczenia_1a.ipynb

https://playground.tensorflow.org

Nice NN demonstrator

Inne klasyfikatory
(oprócz siecineuronowych)

- Cuts
- Linear Discriminants (like Fisher)
- Support Vector Machines
- Naive Bayes (Likelihood Discriminant)
- Kernel Density Estimation
- Decision Trees – Boosted Decision Trees BDT
- Neural Networks
- Bayesian Neural Networks
- Genetic Algorithms

- I wiele, wiele innych…

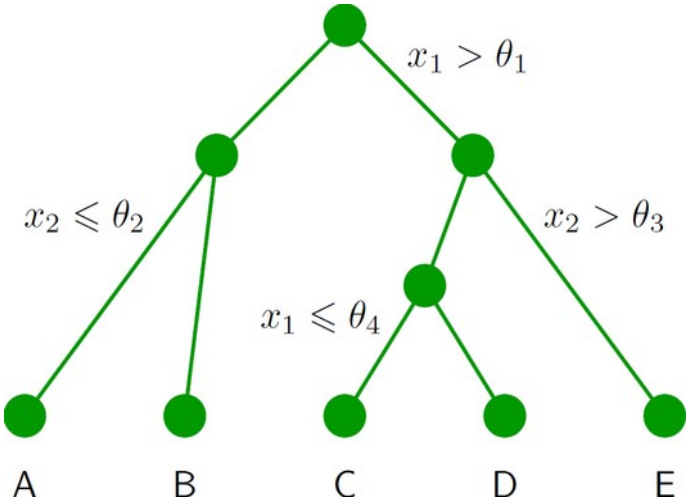
Co oznacza skrót BDT ???

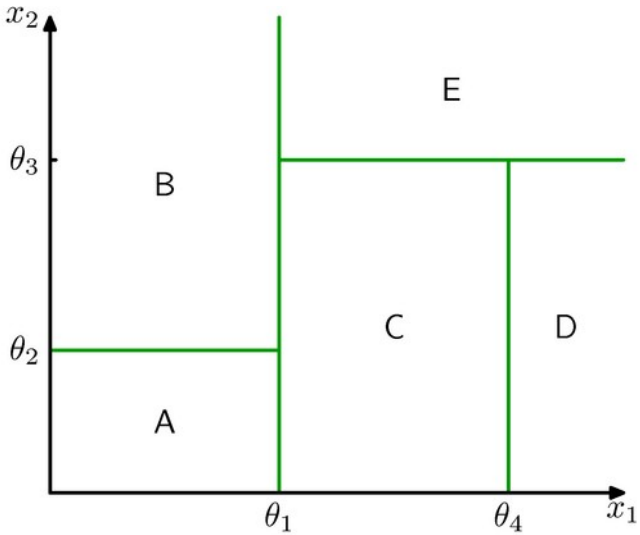
- **BDT – Boosted Decision Tree:**
  - **Decision Tree** – algorytm znany od dawna, powszechnie stosowany we wszelakich systemach eksperckich. Jako drzewo decyzyjne formułuje się np. schematy działania przy udzielaniu pierwszej pomocy: jeśli coś to zrób to i to, sprawdź dalej inny warunek itd.
  - **Boosted** – wzmocniony.

    Metoda łączenia wielu słabych klasyfikatorów w celu uzyskania mocnego klasyfikatora. Nie musi się ograniczać do drzew decyzyjnych! Choć z nimi jest najchętniej używana.

Drzewa decyzyjne

- Drzewo decyzyjne – szereg następujących po sobie cięć, każdy końcowy „liść” (A,B,C,D,E) ma przypisaną klasyfikację, np.”sygnał” i „tło”.

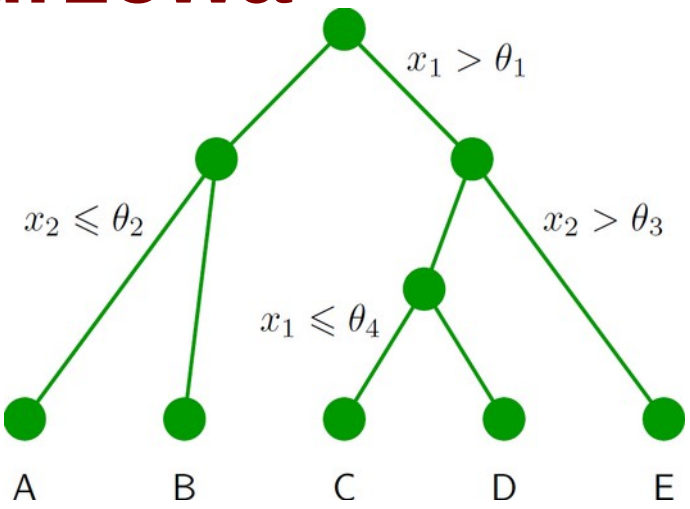




- Proste w interpretacji i wizualizacji
- Odporne na przypadki odstające od innych (*outliers*).
- Słabe zmienne są ignorowane.
- Bardzo szybki trening oraz klasyfikacja.
- Niestety: **czułe na fluktuacje, niestabilne.**

Budowanie drzewa

- Zaczynamy budowę drzewa od korzenia.
- Dzielimy zbiór treningowy na dwa poprzez cięcie „najlepiej separujące” na najlepszej zmiennej.
- Powtarzamy procedurę aż spełnione zostaną warunki końcowe, np. liczba liści, liczba przypadków w liściu itd.
- Stosunek S/B w liściu określa klasyfikację (binarnie sygnał, tło lub liczba rzeczywista określająca prawdopodobieństwo, że jest to sygnał).



Definicje separacji:

- indeks Gini: (*Corrado Gini 1912, używany np. do mierzenia nierównomierności dochodów*)

  p (1-p) : p= P(sygnał), *purity*

- Entropia wzajemna:

  -(p ln p + (1-p)ln(1-p))

- Błędna identyfikacja:

  1-max(p,1-p)

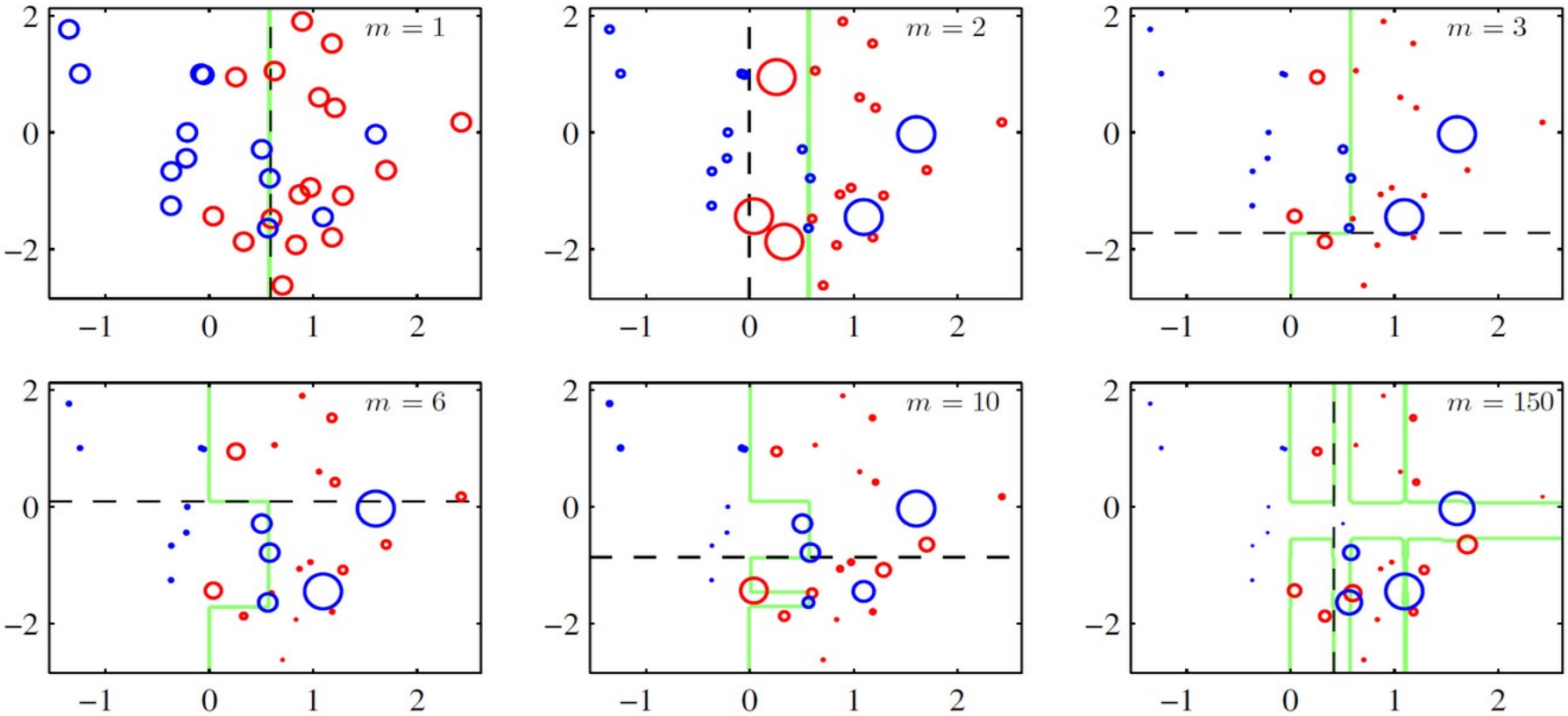
AdaBoost - łączenie klasyfikatorów

Problem: czy mając słaby klasyfikator można go poprawić?

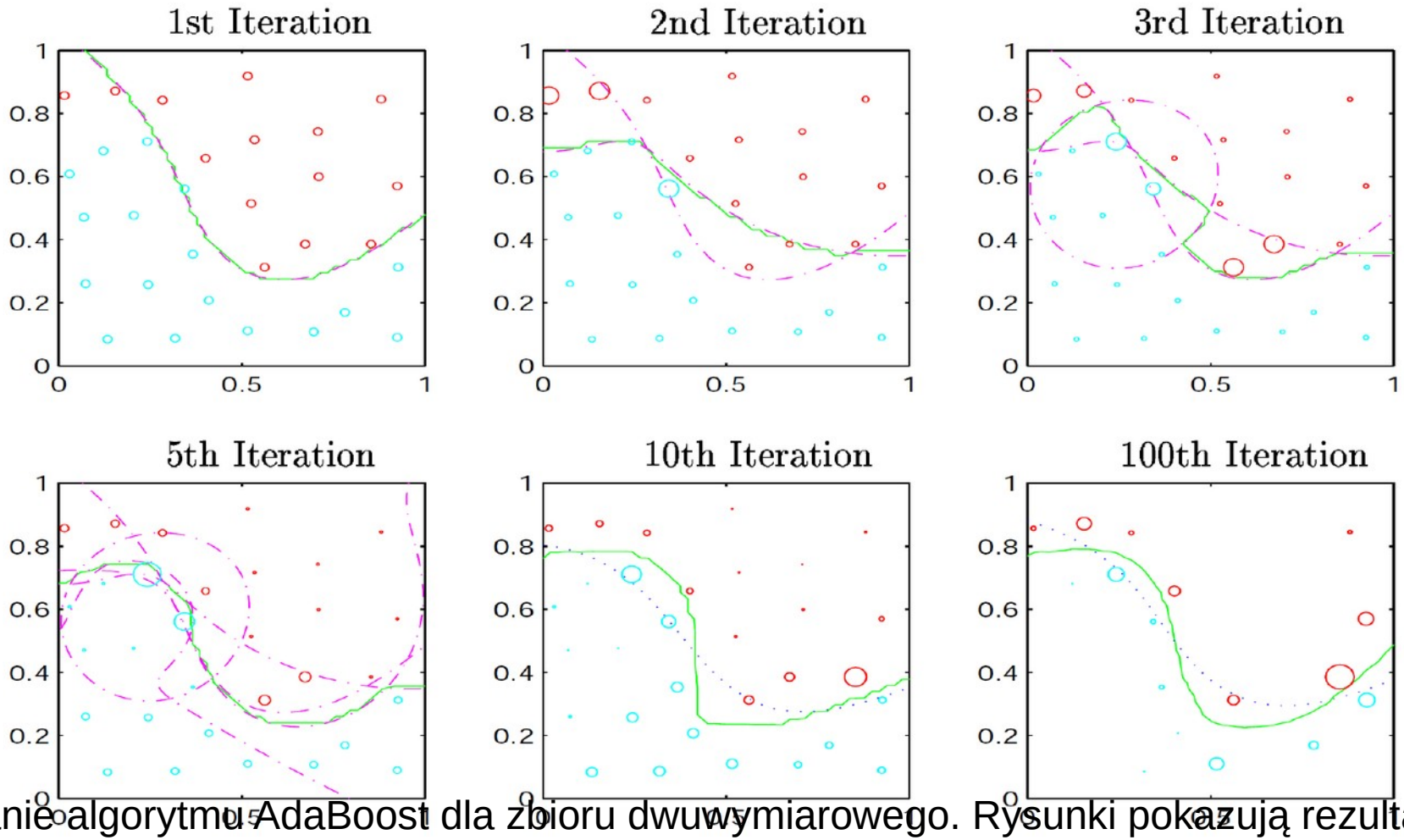
Odpowiedź: tak, stosując go wiele razy.

- Najczęściej używany algorytm: **AdaBoost** (Freund & Schapire 1996 – nagroda Gödla)

- Zbuduj drzewo decyzyjne
- Zwiększ wagi źle sklasyfikowanych przypadków
- Powtarzaj wiele razy (typowo 100-1000)
- Klasyfikuj przypadki na podstawie „głosowania” wszystkich drzew.



AdaBoost



Działanie algorytmu AdaBoost dla zbioru dwuwymiarowego. Rysunki pokazują rezultaty działania klasyfikatora po pierwszej, drugiej, trzeciej, piątej, dziesiątej i setnej iteracji. Linia ciągła obrazuje działanie kombinowanego klasyfikatora, linia przerywana granice klas otrzymanych z poszczególnych klasyfikatorów. Dla dwóch ostatnich rysunków linią przerywaną zaznaczono granice otrzymane za pomocą algorytmu bagging (zaraz o tym będzie).

Wzmacnianie klasyfikatora

- Boosting, bagging – w ,,magiczny" sposób otrzymujemy silny klasyfikator ze słabego.
- **Przeważnie używane do wzmacniania drzew decyzyjnych – Boosted Decision Trees BDT.**
- Dobre wyniki bez pracochłonnego dopasowywania parametrów:

  „*the best out-of-box classification algorithm*”.

- Stosunkowo odporny na przeuczenie.
- Obecnie bardzo modny i często stosowany. I to z dobrymi skutkami!

Trochę programowania

- Dodajmy klasyfikację i regresję za pomocą Boosted Decision Trees do naszych problemów klasyfikacji i regresji:

  https://github.com/marcinwolter/DeepLearning_2020/blob/main/wyk%C5%82ad_1a.ipynb

  https://github.com/marcinwolter/DeepLearning_2020/blob/main/cwiczenia_1a.ipynb

Podsumowanie

- Potrafimy użyć sieci neuronowej oraz Boosted Decision Trees BDT
- Następnym razem zbudujemy pierwszą głęboką sieć neuronową z użyciem pakietu KERAS + TensorFlow