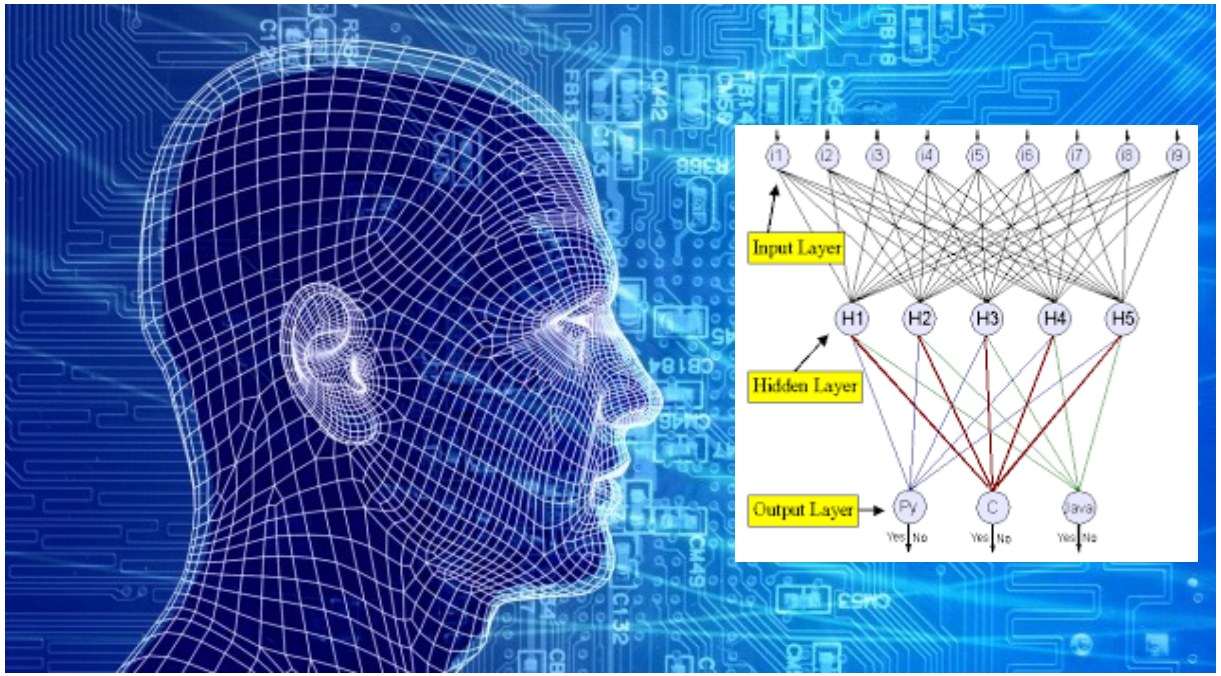
Machine learning

Lecture 3



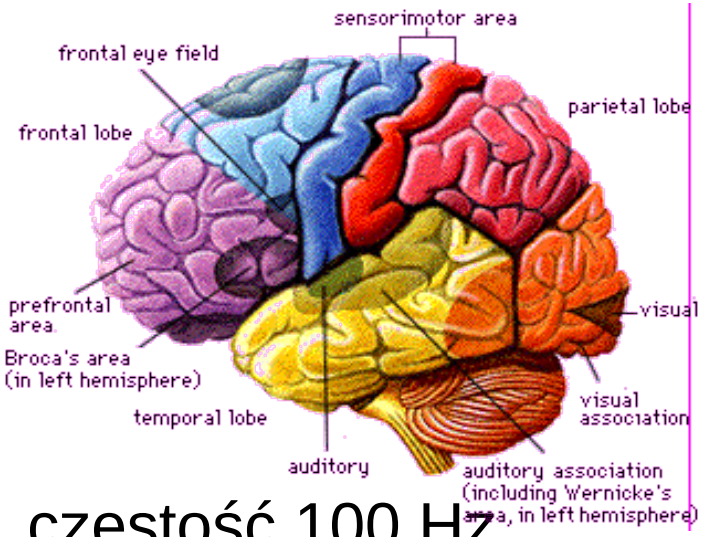
Marcin Wolter

IFJ PAN

16 March 2017

- Sieci neuronowe i ich zastosowanie w fizyce.
- Bayesowskie sieci neuronowe.

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;

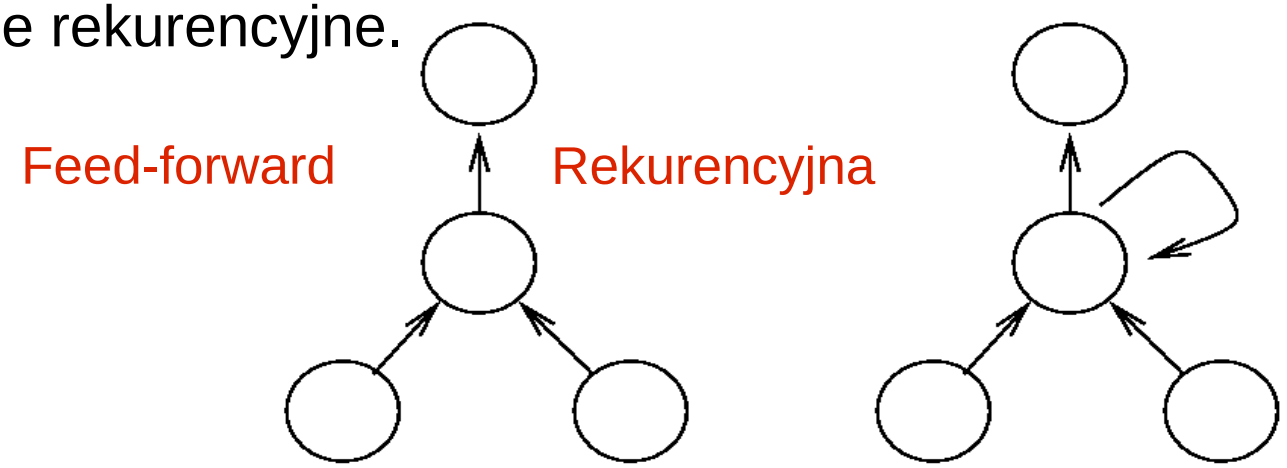
1982 Kohonen, Self-Organizing Maps

...

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

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

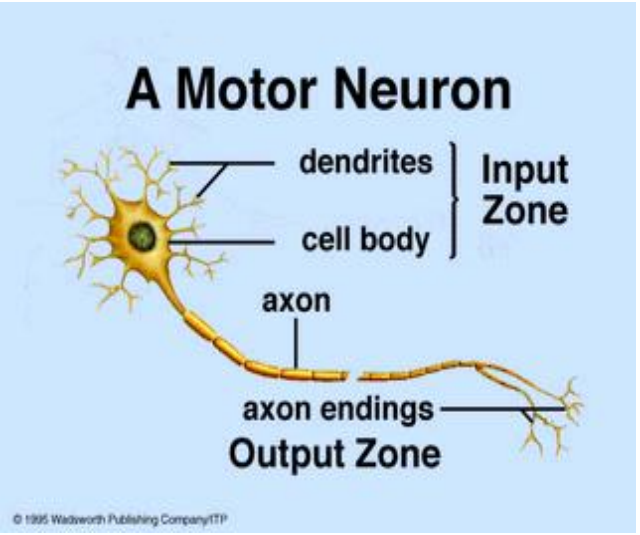


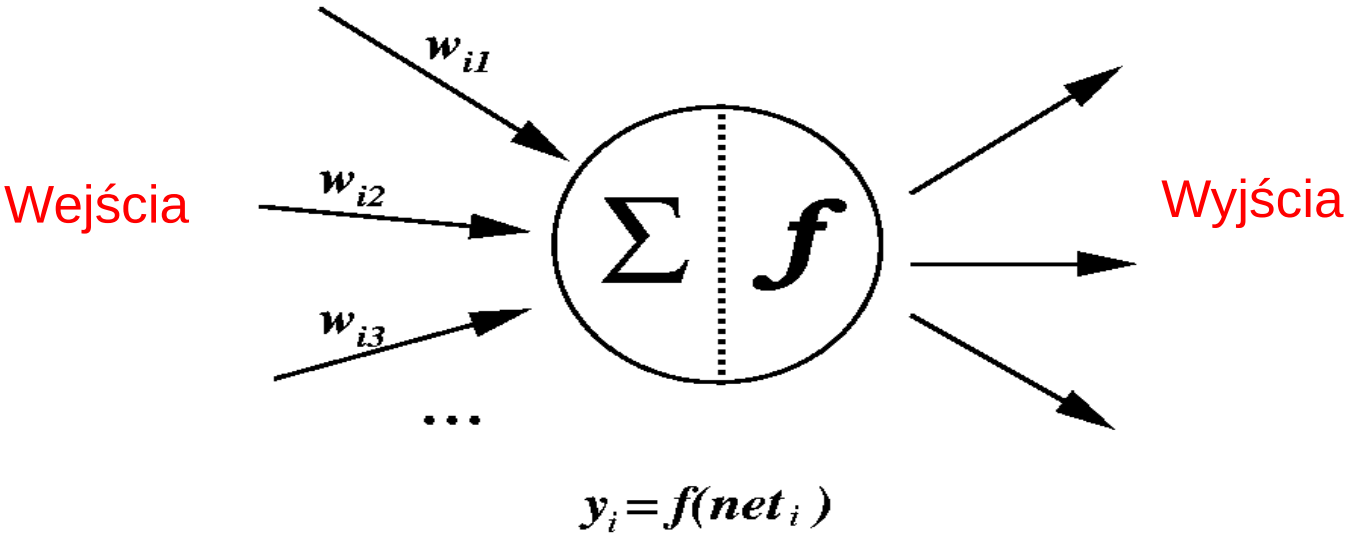
Do czego są używane?

- Systemy ekspertowe
- Rozpoznawanie obrazu
- Przewidywania (meteorologia, giełda...)
- W fizyce cząstek elementarnych:
  - Analiza danych (głównie selekcja przypadków)
  - Wstępna selekcja przypadków podczas zbierania danych – czasy ~ms (dedykowana elektronika, układy scalone)

Neuron – podstawowa cegiełka

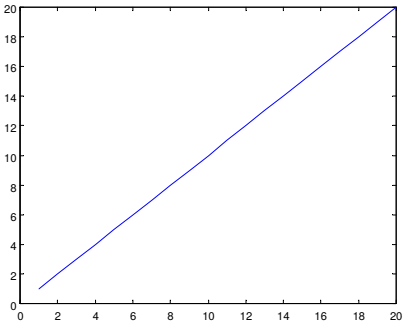
- Funkcja ważonej sumy wejść $$y_i = f\left(\sum_j w_{ij}\, y_j\right)$$





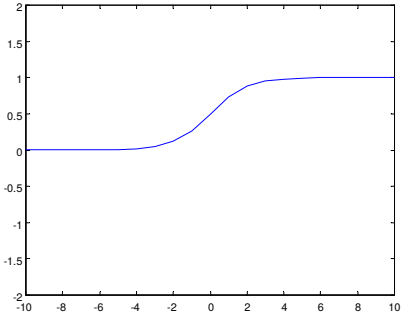
- Funkcję *f* nazywamy funkcją aktywacji

Typowe funkcje aktywacji



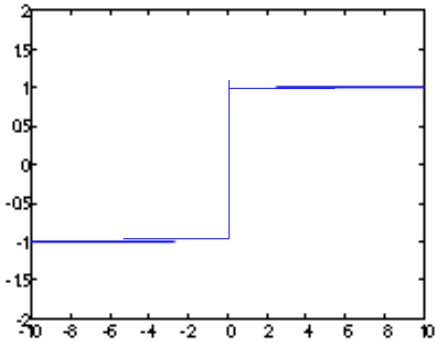
Liniowa (wtedy całą sieć nazywamy liniową)

$$y = x$$

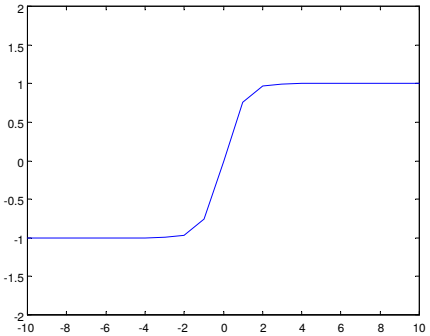


Logistic

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



Progowa

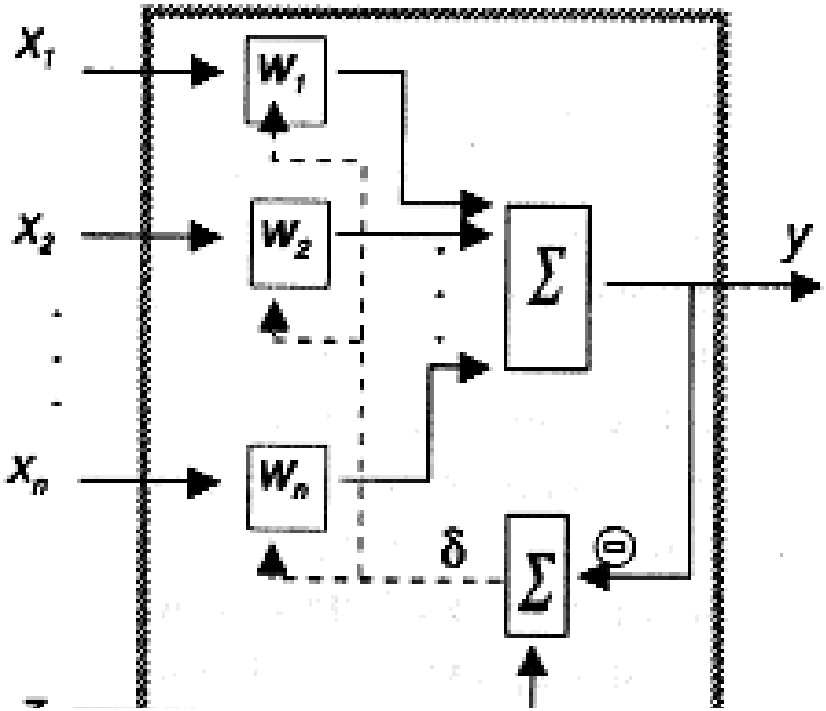


Tangens hiperboliczny

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

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!)

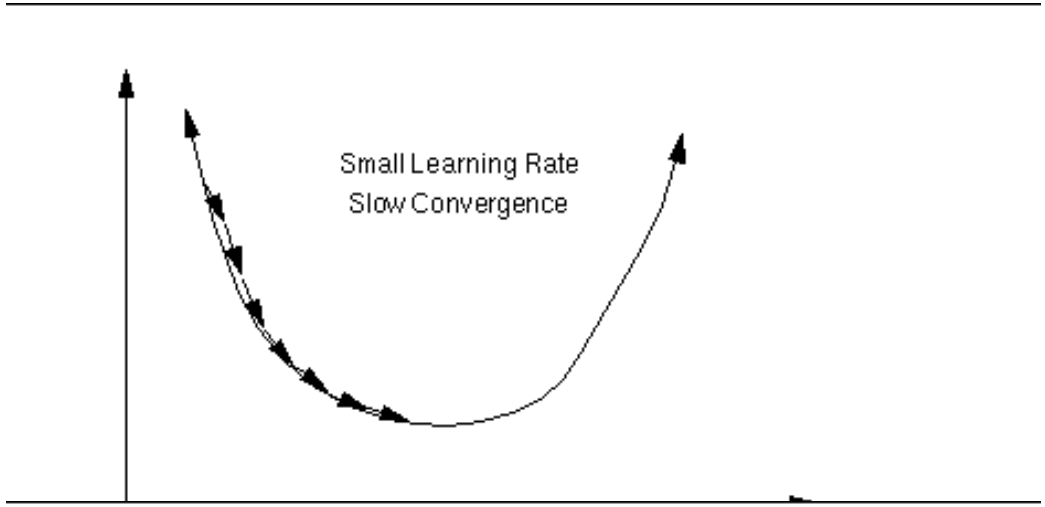
- ZADANIE – minimalizacja funkcji kary:
- Znajdź: $\chi^2 = \sum (z^{(j)} - y^{(j)})^2$
- Nowy zestaw wag:

$$\delta = z - y$$

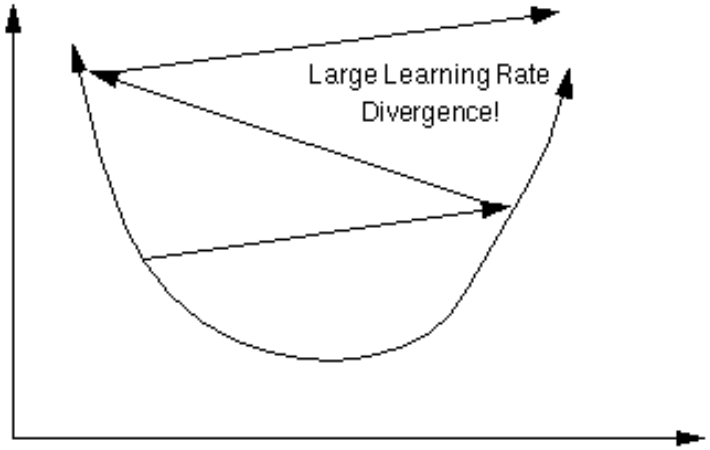
- η - szybkość uczenia

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

Szybkość uczenia



- Za mała

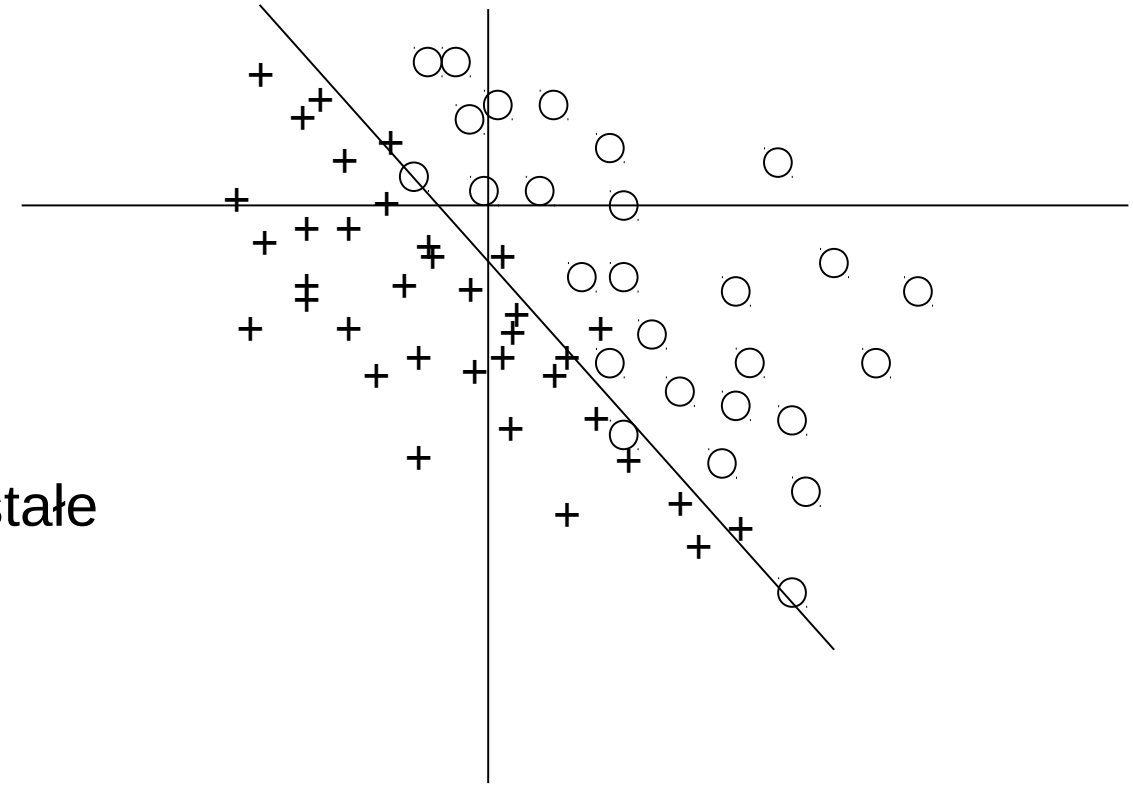


- Za duża

Co potrafi pojedynczy neuron (perceptron)

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

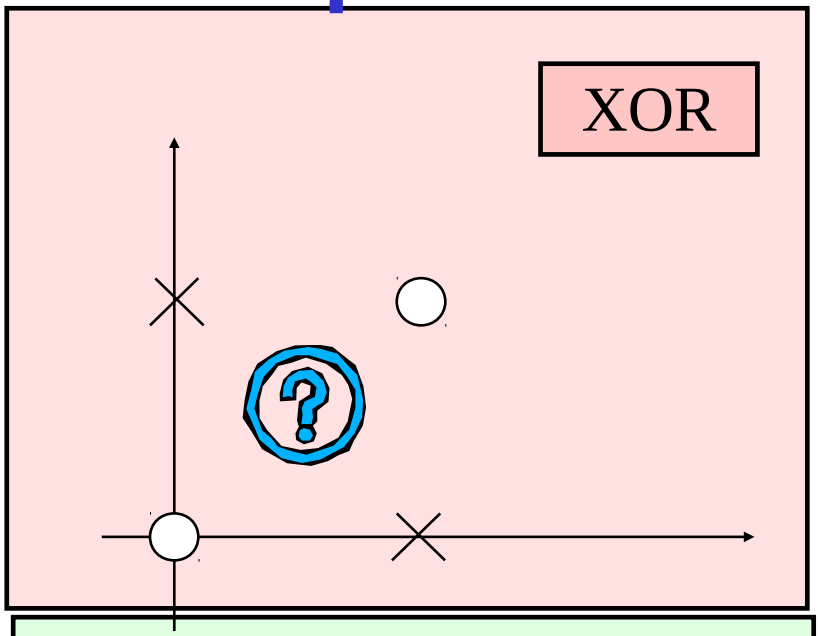
- Punkty leżące nad ową prostą klasyfikujemy jako 1, zaś pozostałe jako 0.

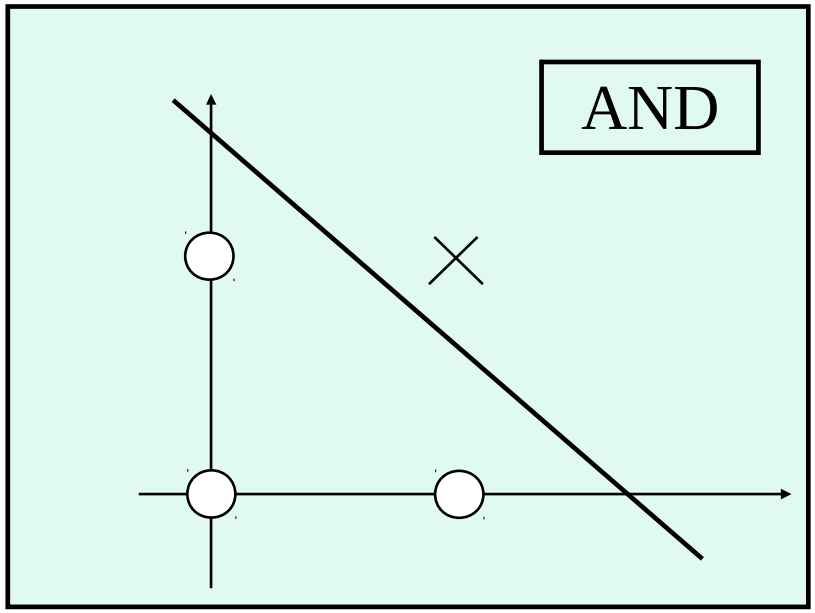


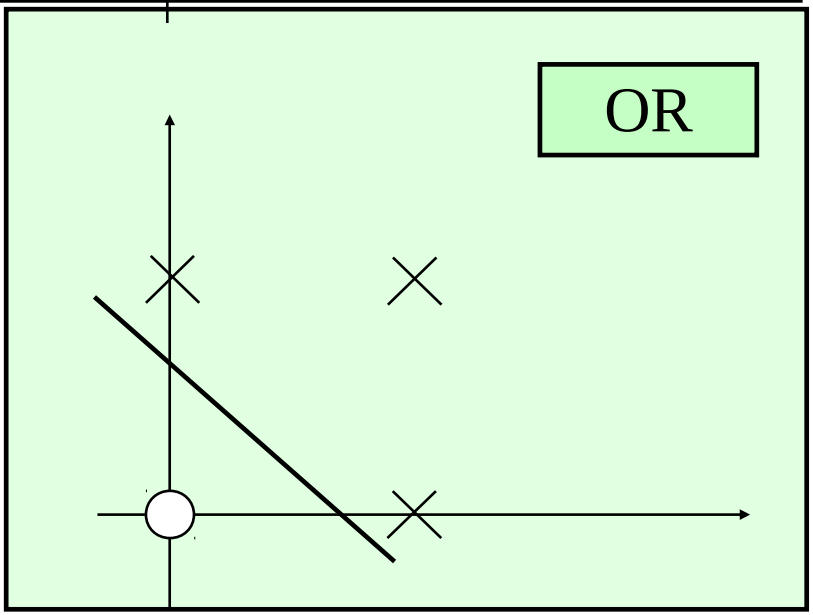
Czego perceptron nie potrafi

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

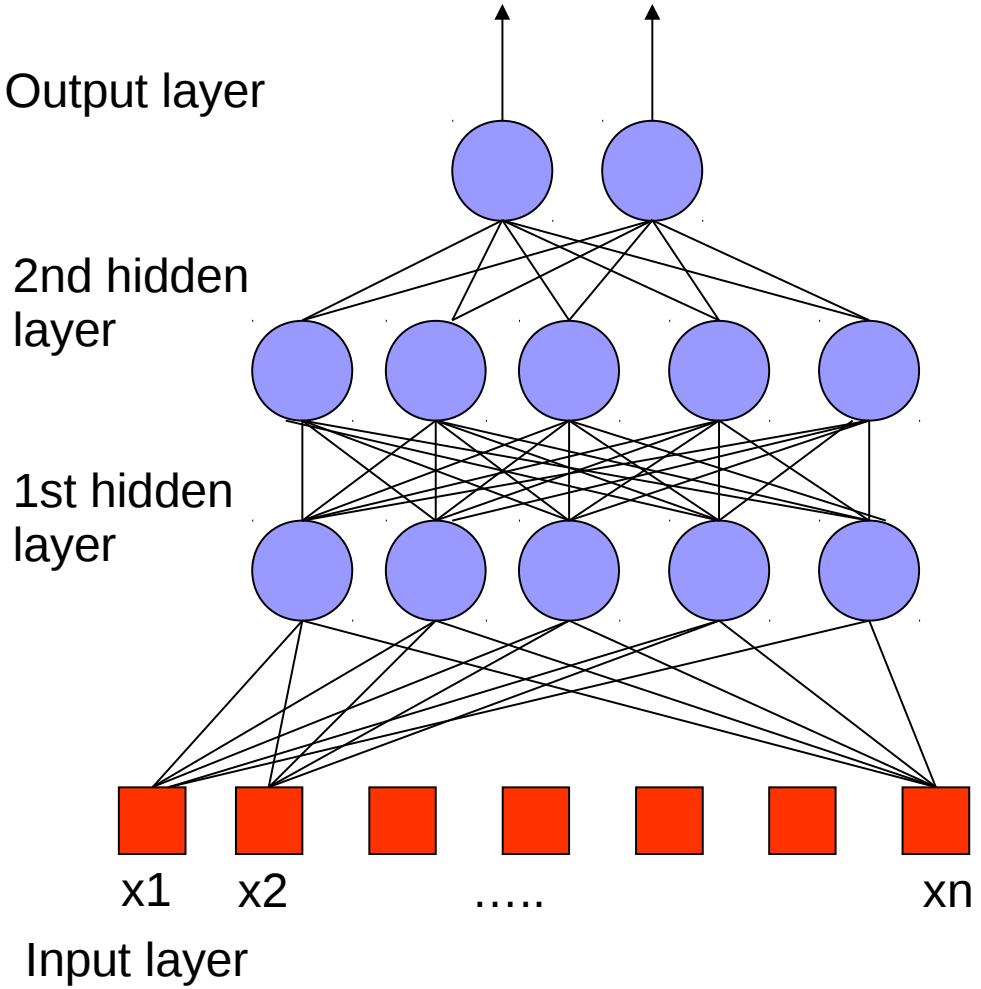
Applet perceptron







Może więc sieć neuronów?

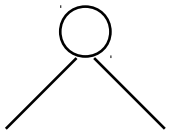
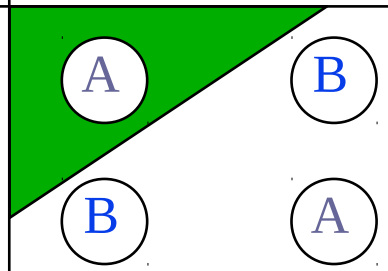
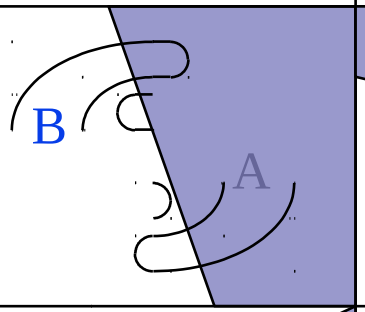
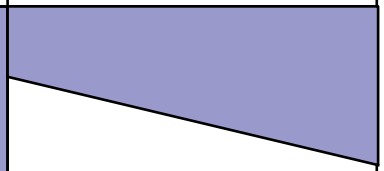
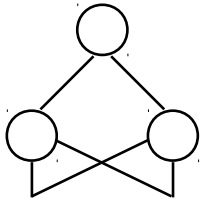
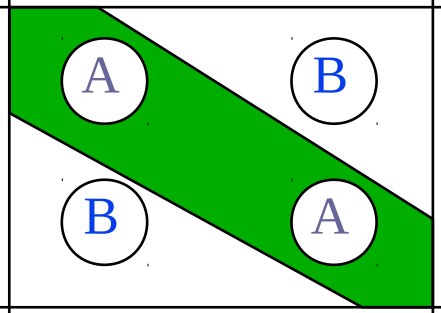
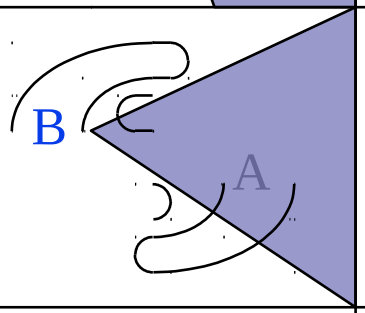
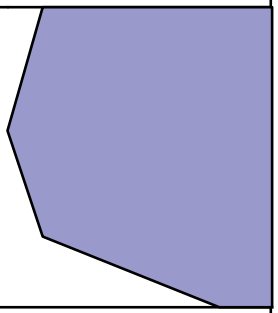
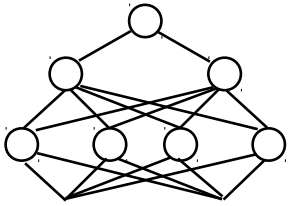
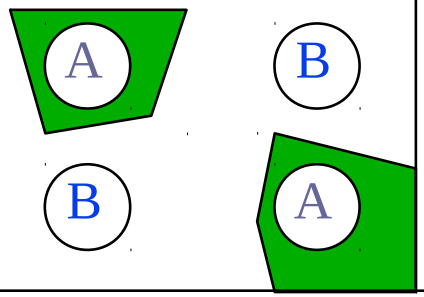
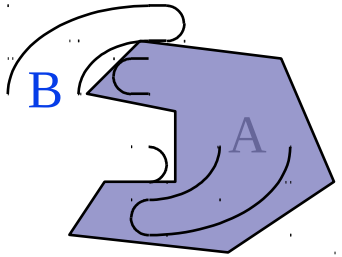
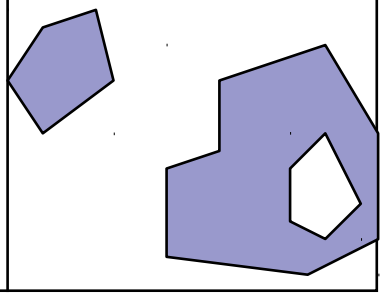


- Informacja propaguje się od wejścia do wyjścia
- Sieć jest złożeniem wielu funkcji aktywacji (w ogólności nieliniowych)
- Odpowiednio złożona sieć może odtworzyć dowolną funkcję.

Co potrafi sieć neuronów

(progowa funkcja aktywacji)

applet general1

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.

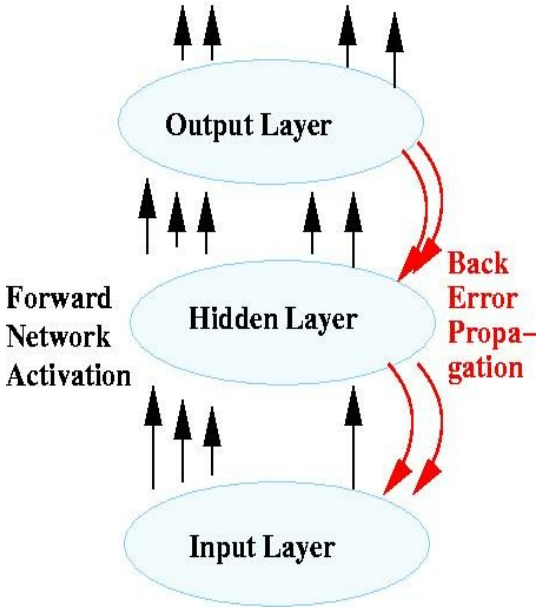
Typowa procedura uczenia

- Dwa zbiory danych: do nauki i do sprawdzenia.
- $\chi^2=\sum(z-y)^2$ 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.



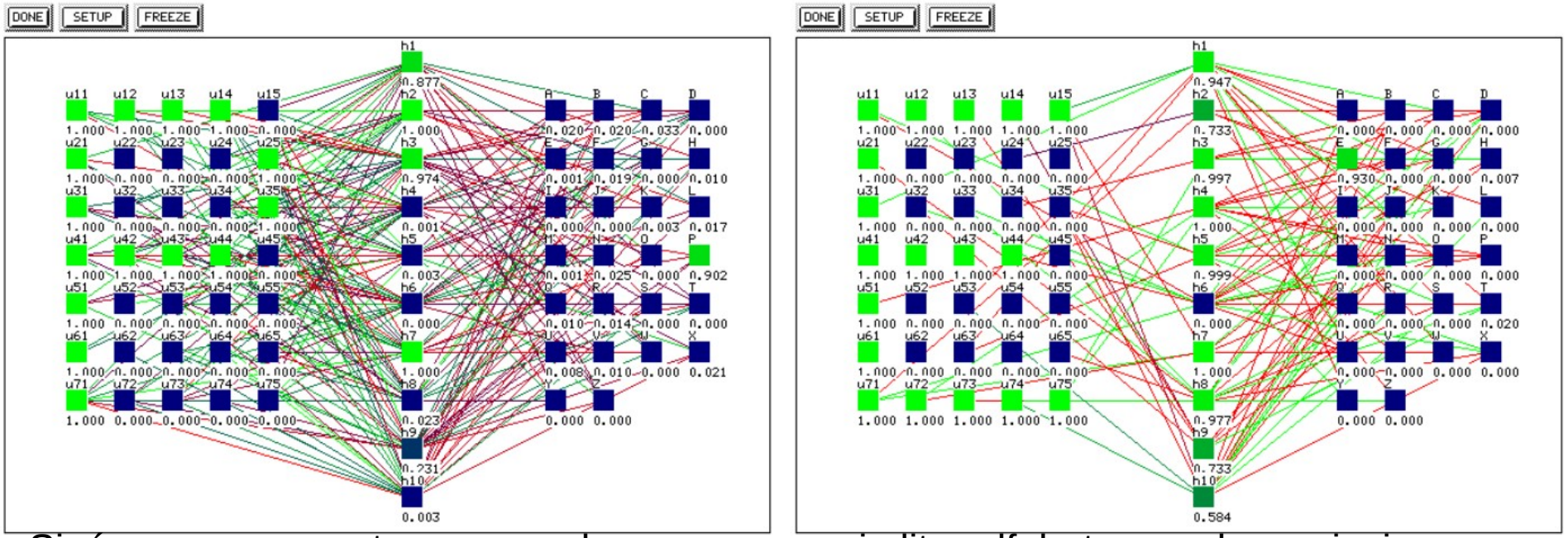
Znajdowanie minimum

- Nie ma pewności, czy znaleźliśmy lokalne czy globalne minimum funkcji straty $\chi^2=\sum(z-y)^2$
- Mechanizmy zapobiegające ugrzęźnięciu w lokalnym minimum:
  - Wybór przypadkowych wag początkowych, powtarzanie uczenia
  - Dodawanie szumu, aby algorytm opuścił lokalne minimum (jittering).

Przycinanie sieci

Algorytmy usuwające mało znaczące połączenia sieci lub całe węzły
- Upraszczają siec, zwiększają szybkość
- Zapobiegają przetrenowaniu

Alternatywa – stopniowo rozbudowujemy sieć dodając nowe neurony/warstwy, aż uzyska optymalną wielkość.



Sieć neuronowa wytrenowana do rozpoznawania liter alfabetu przed przycięciem za pomocą algorytmu optymalny chirurg mózgu (po lewej) oraz po przycięciu (po prawej).

A Strategy for Discovering a Low-Mass Higgs Boson at the Tevatron

Pushpalatha Bhat, Russell Gilmartin and Harrison B. Prosper

FERMILAB-Pub-00/006

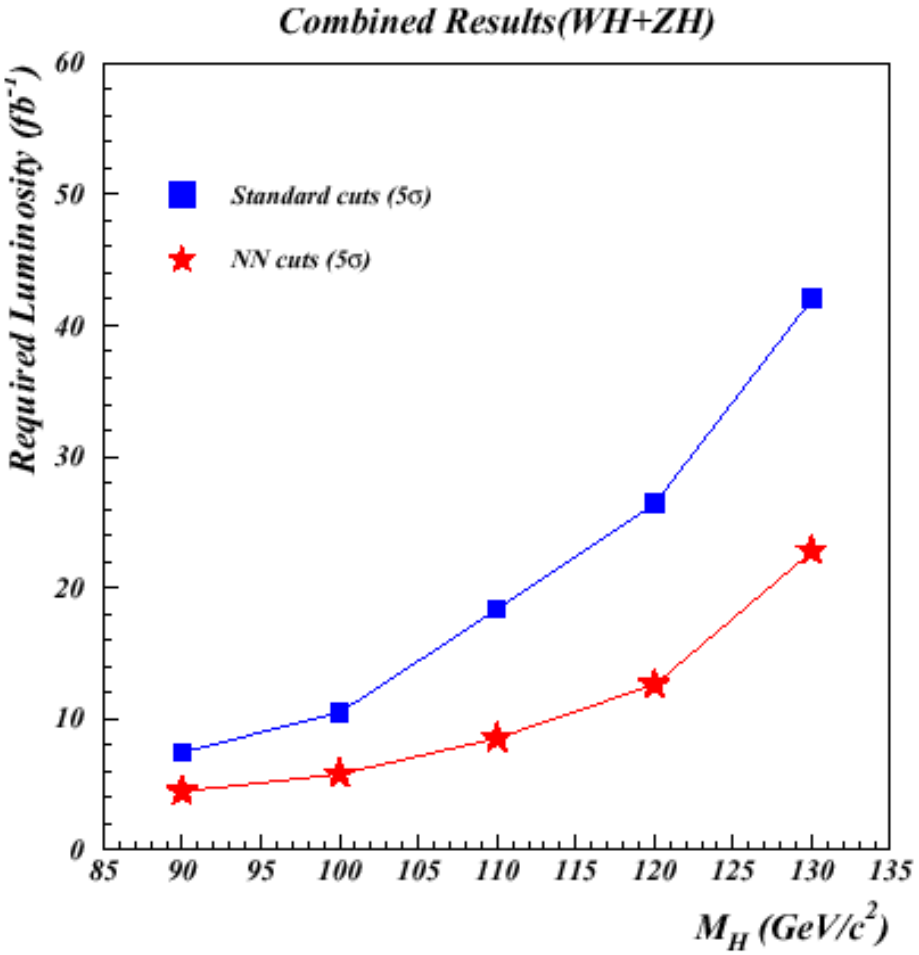
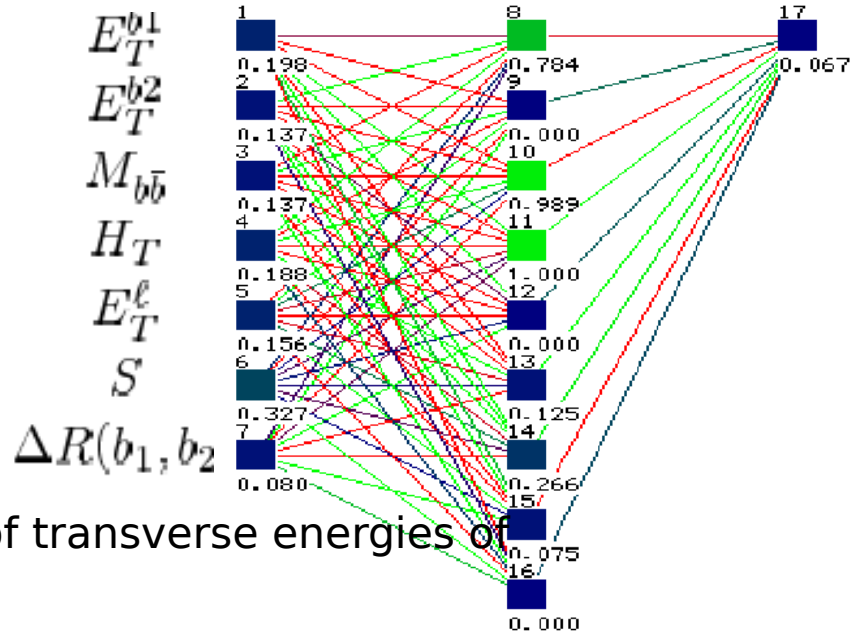


FIG. 8. Required integrated luminosity for a 5σ observation with all channels combined. The luminosities given are for a *single* Tevatron experiment, as in the previous plots.

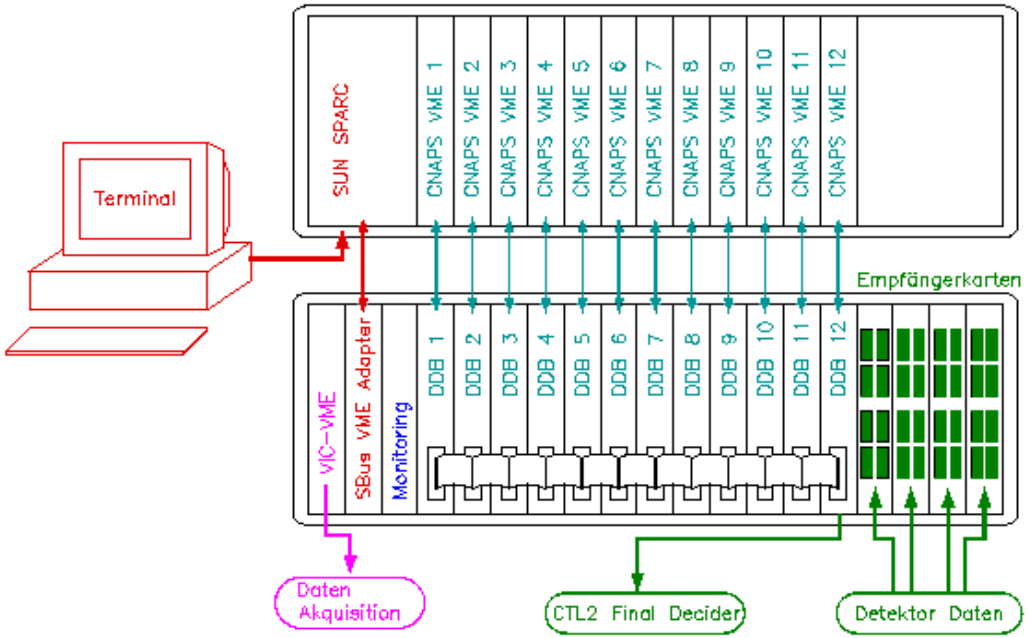


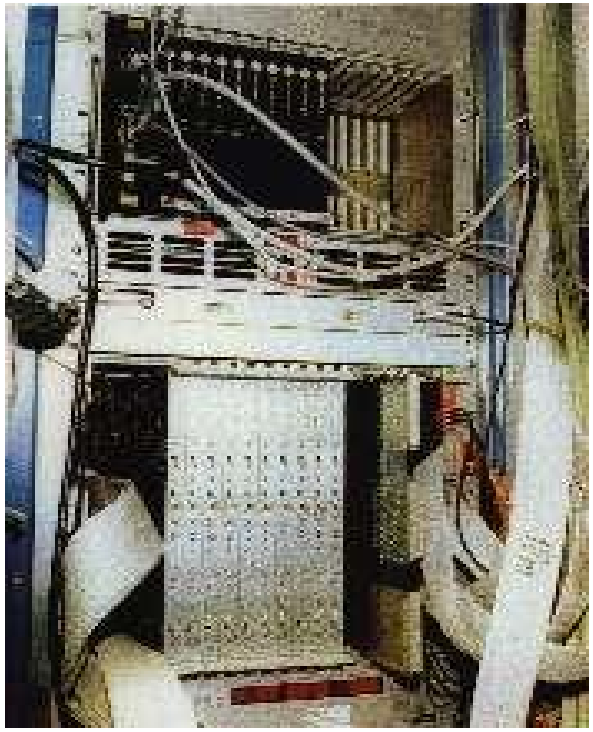
H_T – sum of transverse energies of all tracks.

Przykład:
przewidywania ile potrzeba przypadków do odkrycia hipotetycznej cząstki Higgsa w laboratorium Fermilab.

System wstępnej selekcji przypadków (*trigger*) w eksperymencie H1 (1996 r)

- Wytrenowana sieć neuronowa jest szybka – nadaje się do sytemu *triggera*.
- W drugim stopniu selekcji (L2) – sprzętowa implementacja sieci neuronowej.
- Implementacja na równoległych procesorach (CNAPS z Adaptive Solutions).
- Czas decyzji – (L1 hardware – 2.3 µs, L2 sieć neuronowa - 20 µs, L3 mikroprocesory - 800 µs).



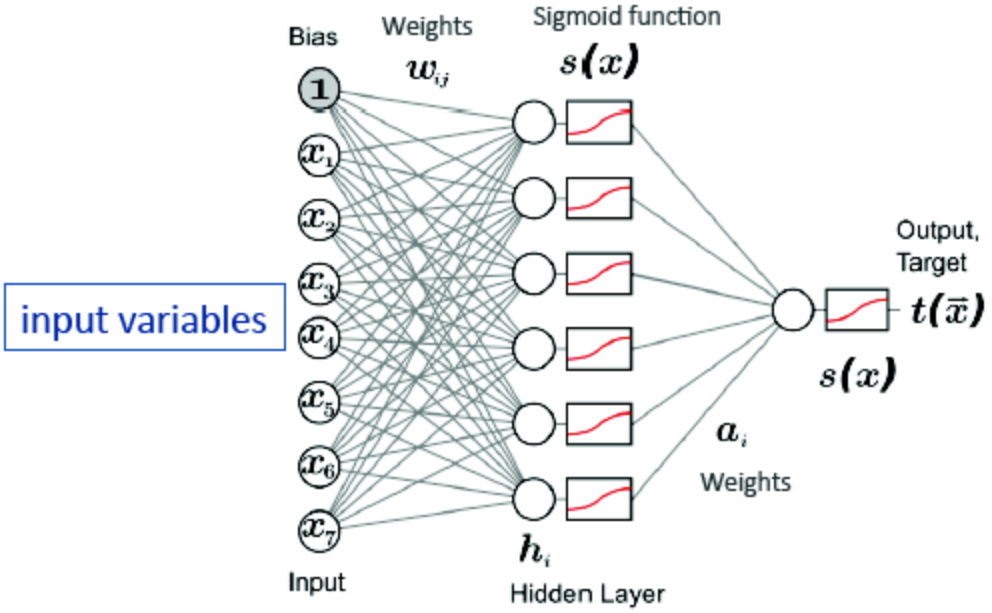


J. K. Kohne i inni,
Realization of a second level neural network trigger for the H1 experiment at HERA,
Nucl. Instrum. Meth. A389 (1997)

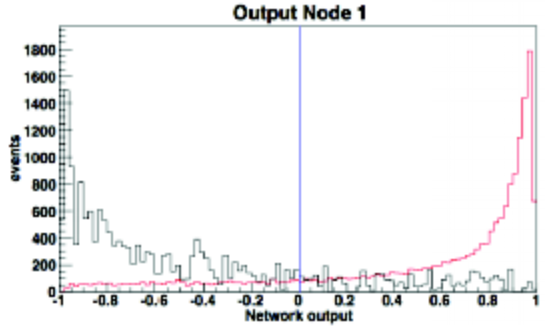
Zastosowanie sieci neuronowych do selekcji sygnału w poszukiwaniach naładowanego bozonu Higgsa w eksperymencie ATLAS

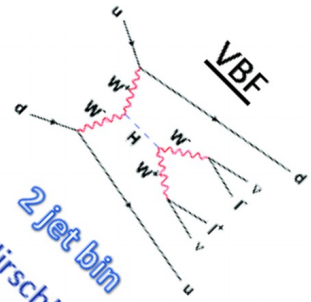
Using Neural Networks to improve sensitivity
Input variables for the training of NN

Neural Networks are trained in the 0-jet bin and 2-jet bin for m_H = 150 GeV and 180 GeV.



Training @
m_H = 150 GeV
m_H = 180 GeV





Simon Köhlmann, Georg Sartisohn, Wolfgang Wagner, Dominic Hirschbühl, Gunar Ernis,

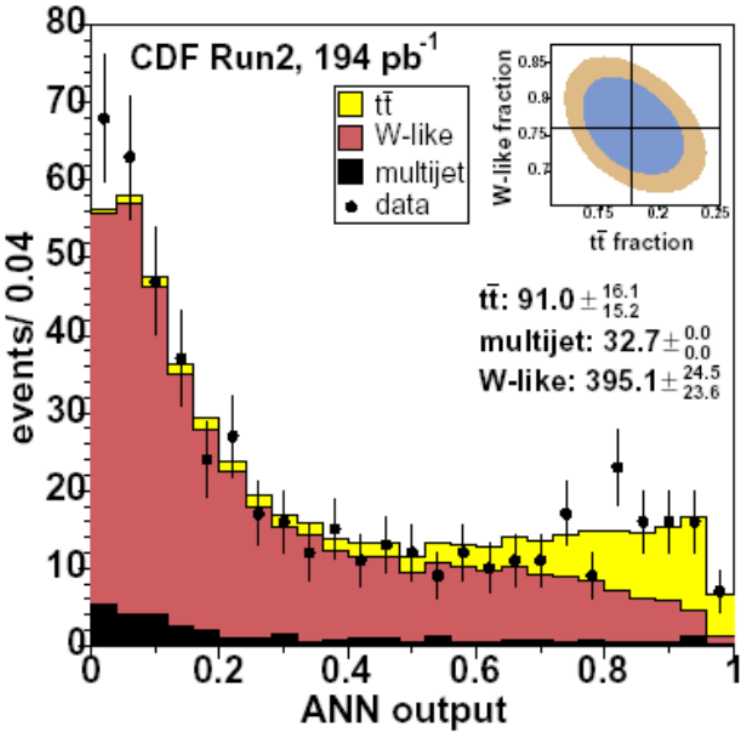
<phi-t> *Implementation with NeuroBayes*

06 Dec 2012

19

Search for Higgs bosons predicted in Two-Higgs-Doublet Models in the WW→lvlv decay channel

Identyfikacja kwarków *t* w eksperymencie CDF

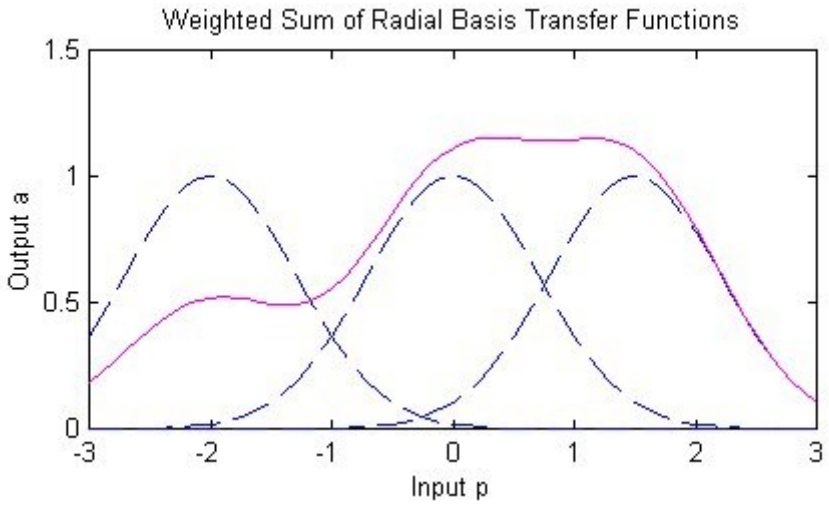


Rysunek 4.5: Rozkład odpowiedzi sieci neuronowej dla stanu końcowego $W+ \geq 3$ dżety (bozon W oraz co najmniej 3 dżety) w eksperymencie CDF, porównany z wynikami dopasowania funkcji [69].

Radial Base Functions (RBF)

- Odwzorowanie zbioru wejściowego w wyjściowy przez dopasowanie wielu pojedynczych funkcji aproksymujących do wartości zadanych.
- Ważne jedynie w wąskim obszarze przestrzeni wielowymiarowej.
- Neuron w warstwie ukrytej - funkcja zmieniająca się radialnie wokół centrum *c* i przyjmującą wartości niezerowe tylko w otoczeniu tego centrum:

  $f_i(x) = f_i(||x - c||)$ nazywamy radialną funkcją bazową.



Applet rbf

Funkcje RBF

1. **Gaussian Functions:**

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \text{width parameter } \sigma > 0$$

2. **Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{1/2} \qquad \text{parameter } \sigma > 0$$

3. **Generalized Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{\beta} \qquad \text{parameters } \sigma > 0,\ 1 > \beta > 0$$

4. **Inverse Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{-1/2} \qquad \text{parameter } \sigma > 0$$

5. **Generalized Inverse Multi-Quadric Functions:**

$$\phi(r) = \left(r^2 + \sigma^2\right)^{-\alpha} \qquad \text{parameters } \sigma > 0,\ \alpha > 0$$

6. **Thin Plate Spline Function:**

$$\phi(r) = r^2 \ln(r)$$

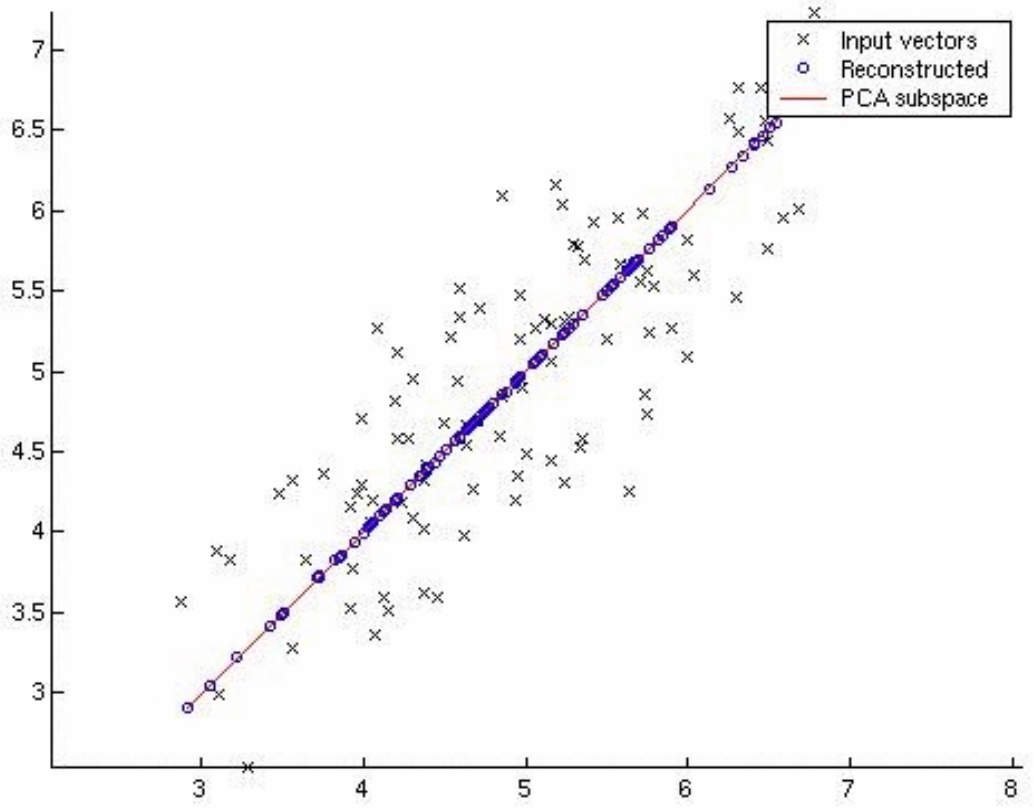
7. **Cubic Function:**

$$\phi(r) = r^3$$

8. **Linear Function:**

$$\phi(r) = r$$

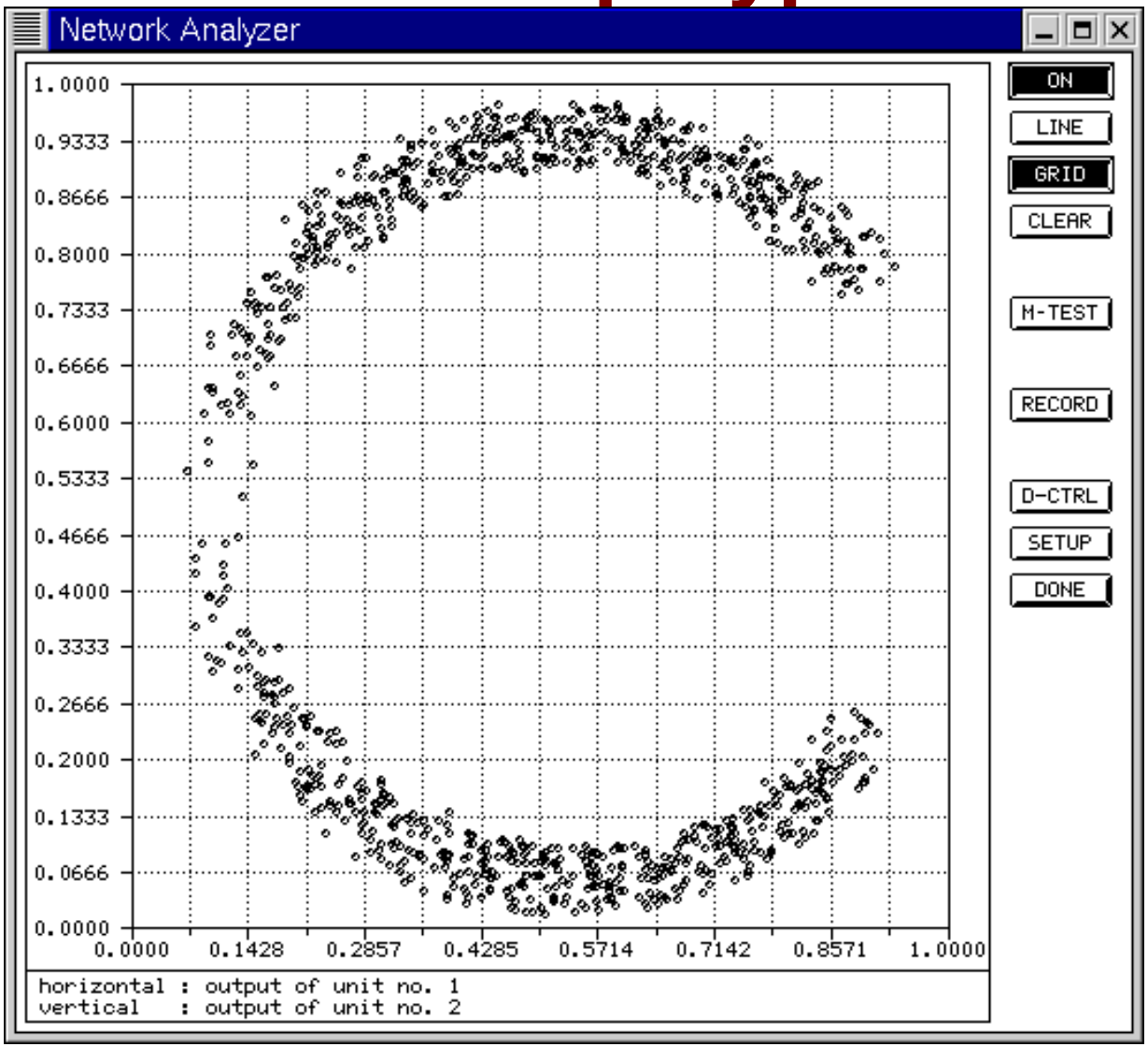
Coś innego...
Nieliniowa PCA
(Principal Component Analysis)



Liniowa PCA:

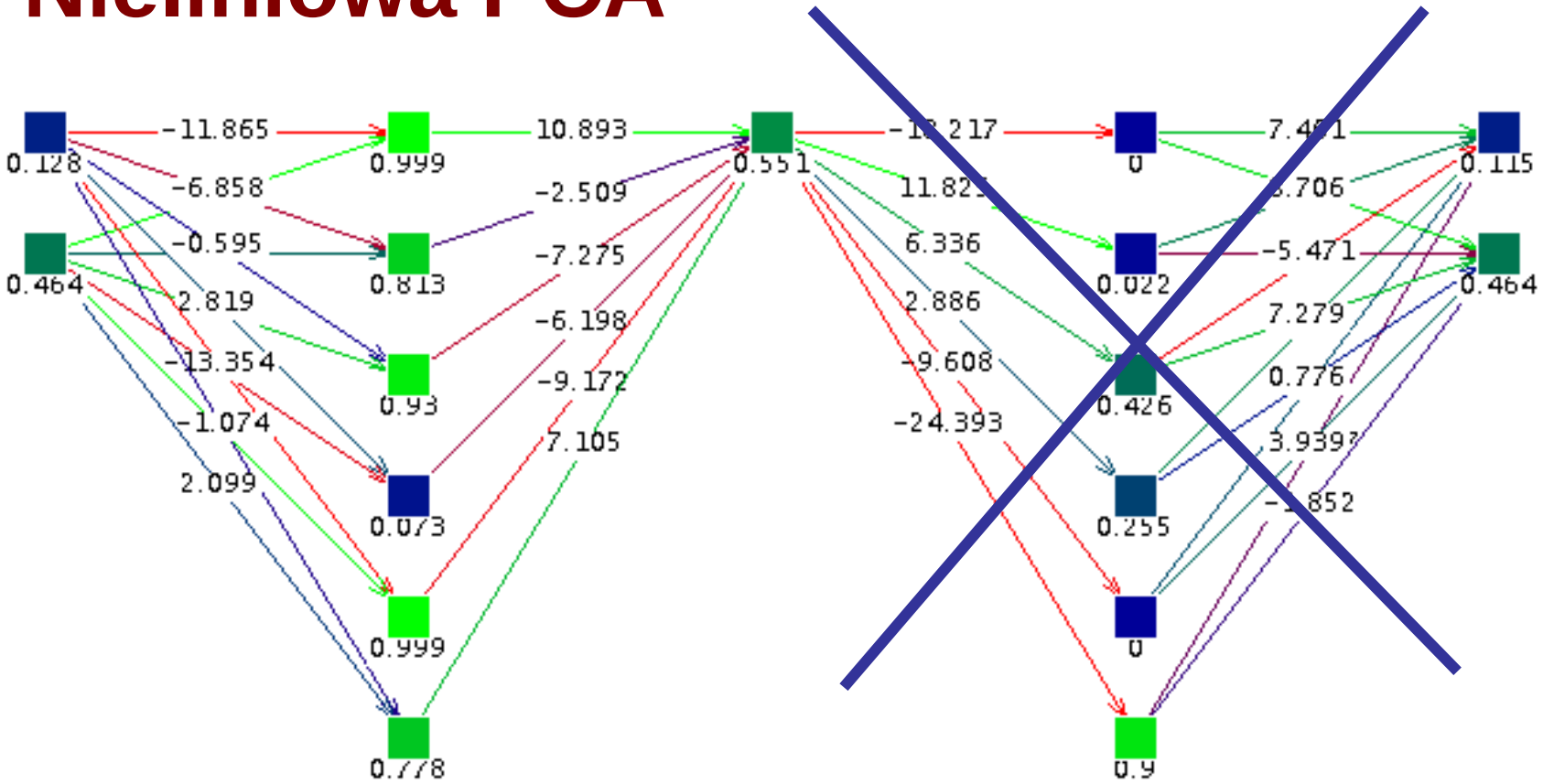
- redukcja wymiarów (tu z dwóch do jednego) tak, aby stracić jak najmniej informacji.
- Znajduje ortogonalną bazę macierzy kowariancji, wektory własne z najmniejszymi wartościami własnymi są pomijane.

A co zrobić z takim nieliniowym przypadkiem?



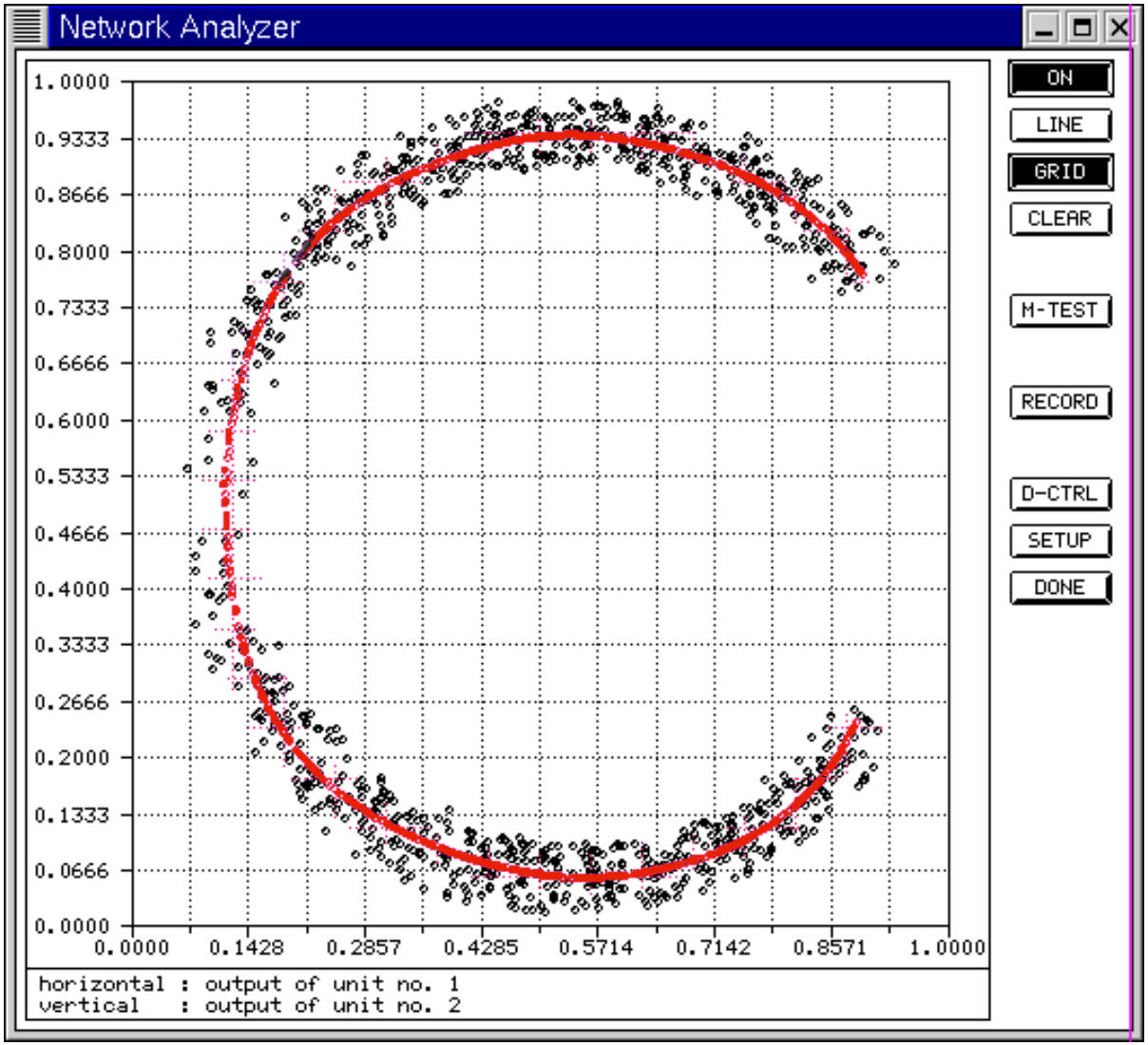
- Jak optymalnie zrobić transformację do jednego wymiaru?

Nieliniowa PCA



Sieć jest trenowana przez podanie takich samych wektorów na wejście i na wyjście. Potem sieć jest rozcinana na pół.

I wynik – transformacja do jednego wymiaru



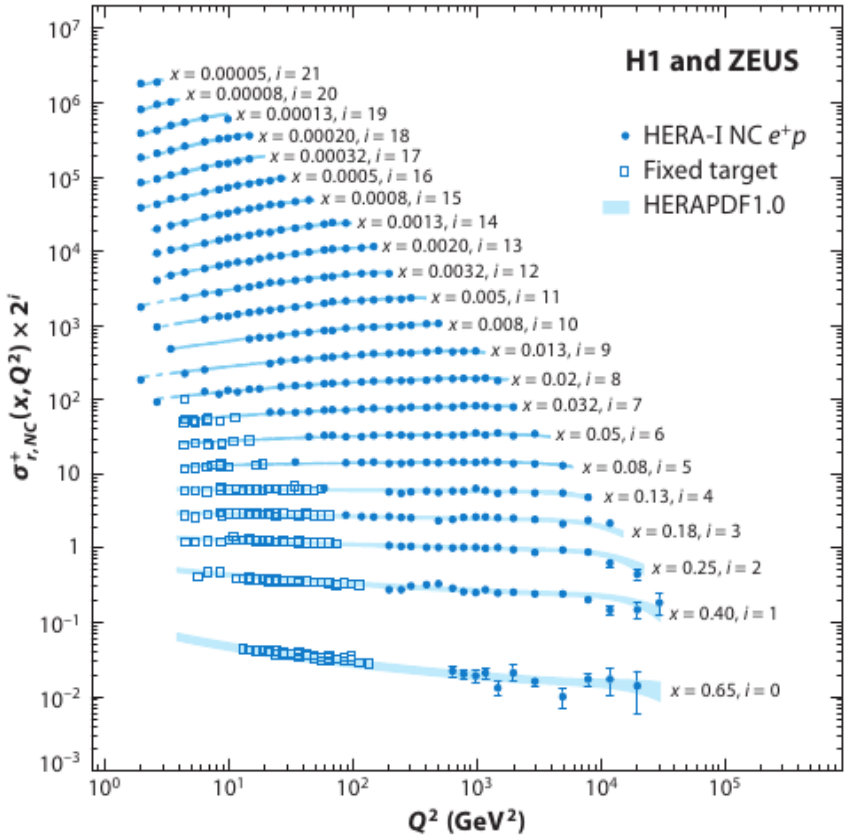
- Transformacja nieliniowa.

Sieci neuronowe aproksymujące funkcje struktury protonu

- Inkluzywny przekrój czynny na rozpraszanie elektronów na protonie można przedstawić za pomocą funkcji struktury $F_1(x,Q^2)$ i $F_2(x,Q^2)$, gdzie x – ułamek pędu niesiony przez parton, Q^2 – przekaz czteropędu.

- Funkcje struktury mierzone są w szeregu eksperymentów, w różnych zakresach kinematycznych. Zestawiając dostępne dane i dopasowując do nich funkcje uzyskamy sparametryzowane funkcje struktury.

- Kolaboracja **NNPDF** używa sieci neuronowych do aproksymacji rozkładu funkcji struktury (lub rozkładu partonów w protonie):
  - nieobciążony estymator (nie wybieramy funkcji aproksymującej),
  - nie histogramujemy przypadków (lepsze wykorzystanie informacji).



NNPDF -aproksymacja funkcji struktury

- **Stworzenie wielu replik danych**

  Wszystkie dane eksperymentalne są użyte do wygenerowania pseudo-danych. Reprodukcja średnich, błędów, korelacji.

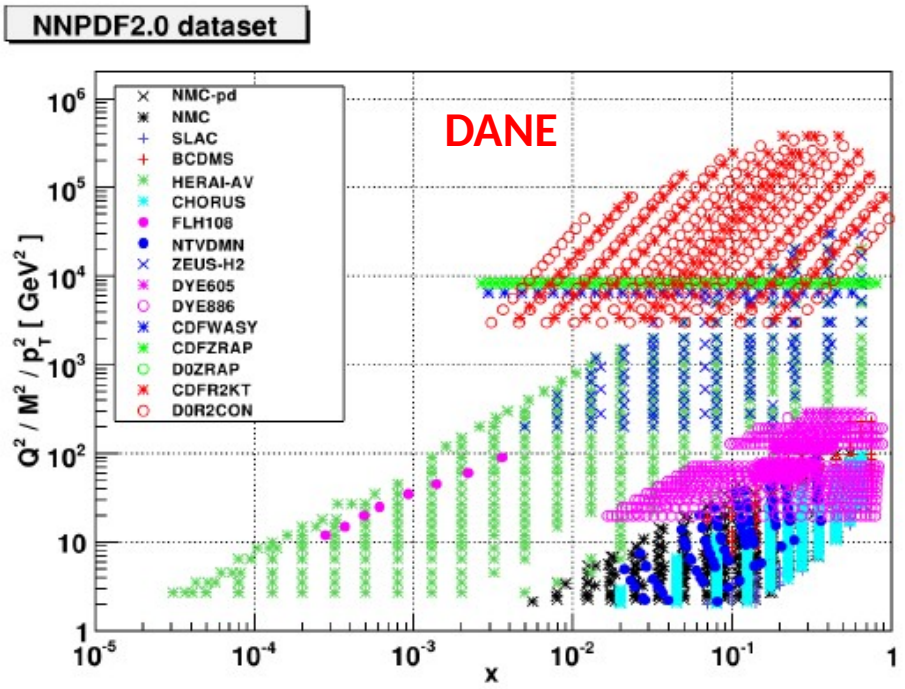
- **Stworzenie rozkładów gęstości prawdopodobieństwa partonów**

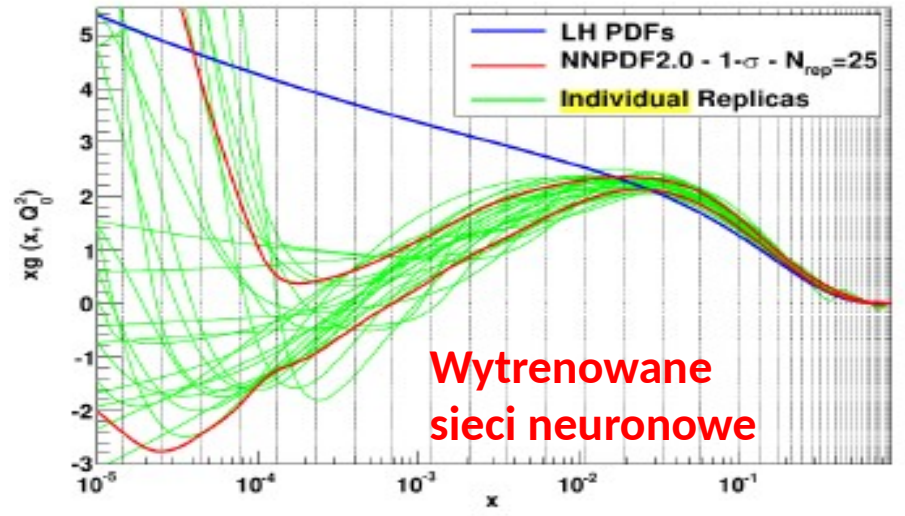
  Dopasowanie sieci neuronowych, po jednej dla każdej repliki.

- **Wiarygodność statystyczna**

  Zestaw wytrenowanych sieci neuronowych jest używany do reprodukowania obserwabli, włączając w to błędy oraz korelacje.

  Uwaga: Wynik uzyskujemy z wielu sieci neuronowych – estymacja błędu.





Uczenie maszynowe a bayesowskie

Uczenie maszynowe

Uczymy algorytm zależności $y = f(x)$ podając **dane treningowe** $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y}) = (x,y)_1, (x,y)_2,\ldots(x,y)_N$ oraz **więzy** dotyczące klasy funkcji.

Uczenie bayesowskie

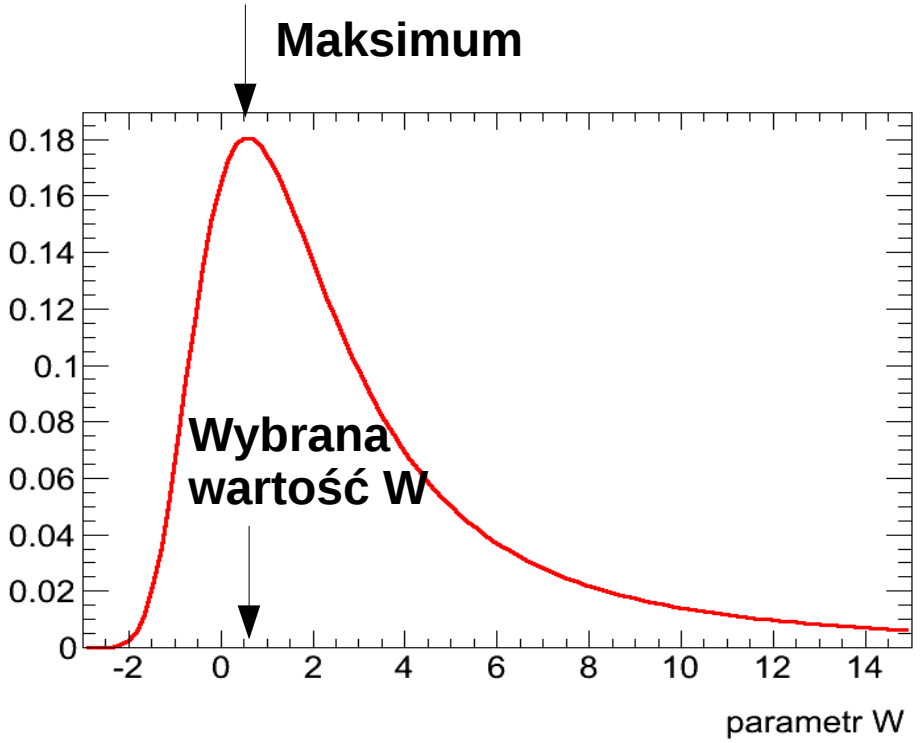
Dla każdej funkcji f(x) z przestrzeni funkcji F znajdujemy prawdopodobieństwo *a posteriori* $\boldsymbol{p(f \mid T)}$ używając zbioru treningowego $\boldsymbol{T= (x, y)}$.

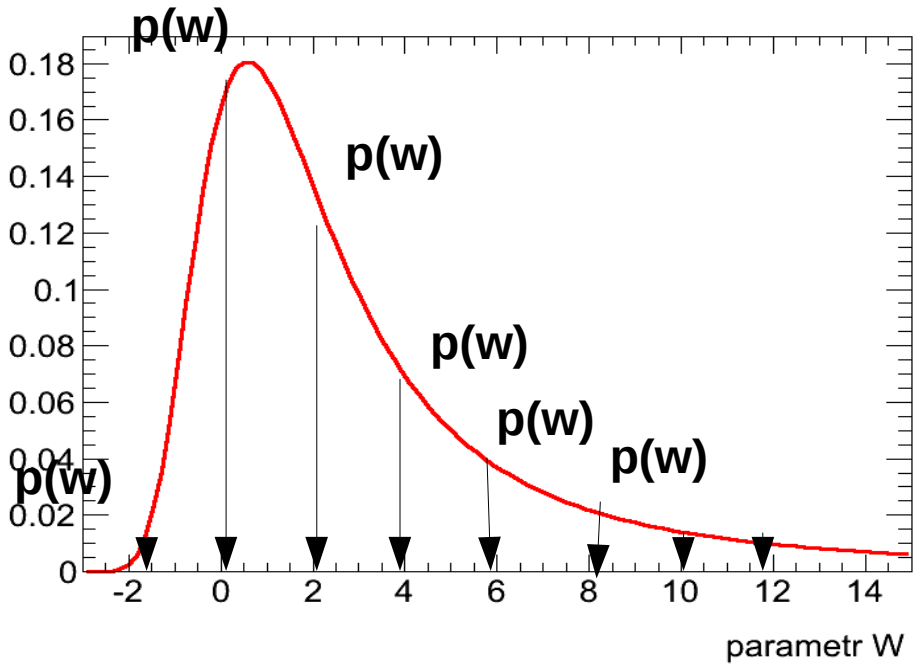
W uczeniu bayesowskim NIE ZNAJDUJEMY jednej, najlepszej funkcji, ale używamy wielu funkcji ważonych ich prawdopodobieństwem.

Prawdopodobieństwo *a posteriori* - prawdopodobieństwo obliczane na podstawie wyników doświadczenia.

Zbiór treningowy $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y})$: zbiór wektorów wejściowych $\boldsymbol{x}$ oraz odpowiedzi $\boldsymbol{y}$.

Uczenie maszynowe a bayesowskie





Uczenie maszynowe

Wybieramy jedną funkcję (lub wartość parametru opisującego funkcję).

Uczenie bayesowskie

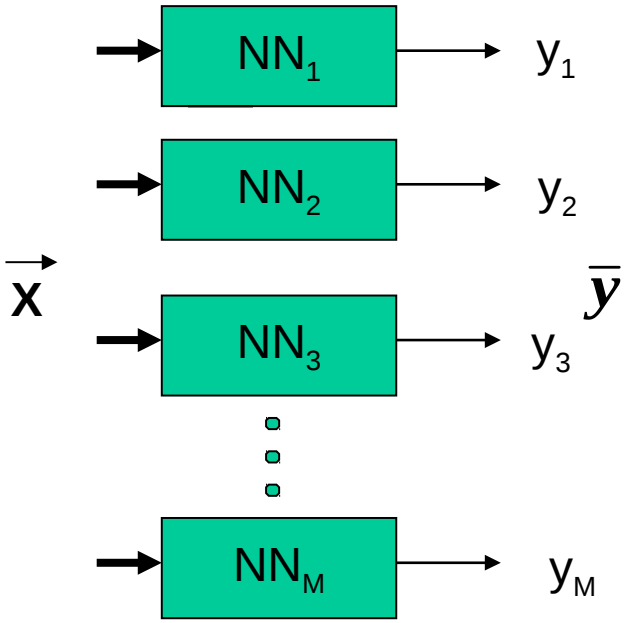
Każda funkcja (lub wartość parametru) ma przypisane prawdopodobieństwo (wagę).

Implementacja: bayesowskie sieci neuronowe

Zamiast wybierać pojedynczy zestaw wag opisujących sieć neuronową znajdźmy gęstość prawdopodobieństwa dla całej przestrzeni wag.

Użyjmy zamiast jednej wiele sieci neuronowych.

Mając wiele sieci możemy uzyskać średnią ważoną lub maksymalnie prawdopodobną sieć, a także błąd estymacji.



$$\bar{y} = \sum_i a_i \, y_i(x)$$

C.M. Bishop
"Neural Networks for Pattern Recognition", Oxford 1995

Darmowe oprogramowanie (używane np. przez kolaborację D0):
Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html

One slide on Bayesian machine learning

Everything follows from two simple rules:

Sum rule: $P(x) = \sum_y P(x, y)$

Product rule: $P(x, y) = P(x)P(y|x)$

Learning:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

$P(\mathcal{D}\|\theta, m)$	likelihood of parameters θ in model m
$P(\theta\|m)$	prior probability of θ
$P(\theta\|\mathcal{D}, m)$	posterior of θ given data $\mathcal{D}$

Prediction:

$$P(x|\mathcal{D}, m) = \int P(x|\theta, \mathcal{D}, m)P(\theta|\mathcal{D}, m)d\theta$$

Model Comparison:

$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}$$

Bayesowskie sieci neuronowe

- Powinniśmy liczyć średnią ze wszystkich możliwych sieci neuronowych...
- Ponieważ sieć neuronowa jest funkcją nieliniową, można posłużyć się rozwinięciem w szereg wokół zestawu parametrów dającym sieć o najmniejszej funkcji straty, czyli takiej, jaką otrzymalibyśmy trenując klasyczną sieć neuronową [Bishop].
- Użycie metod Monte Carlo. Musimy wygenerować zbiór punktów w przestrzeni wag według pewnej gęstości prawdopodobieństwa. Stosowanym rozwiązaniem jest ich generacja z użyciem symulacji Monte Carlo posługującej się łańcuchami Markowa (ang. Markov Chain Monte Carlo, MCMC) [O'Neil].

Bayesowskie sieci neuronowe

Każda sieć opisana przez wektor parametrów **w.**

Dla danych treningowych T= {y,x}, gęstość prawdopodobieństwa w punkcie **w** dana jest przez:

$$p(w,T)=\frac{p(T,w)\,p(w)}{p(T)}$$

równanie Bayesa

p(w) – prawdopodobieństwo a priori, musi być wybrane wcześniej

- Odpowiedzią jest średnia po wszystkich sieciach neuronowych (wartościach **w**)**:**

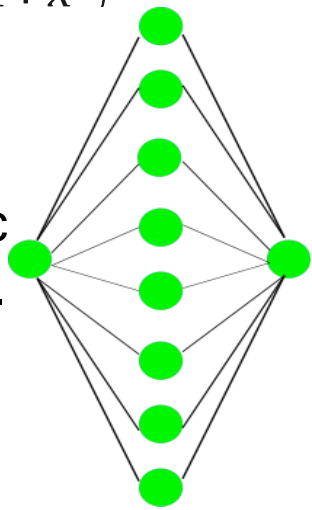
$$\bar{y}(x)=\int f(x,w)\,p(w,T)\,dw \qquad f(x,w)-neural\ network$$

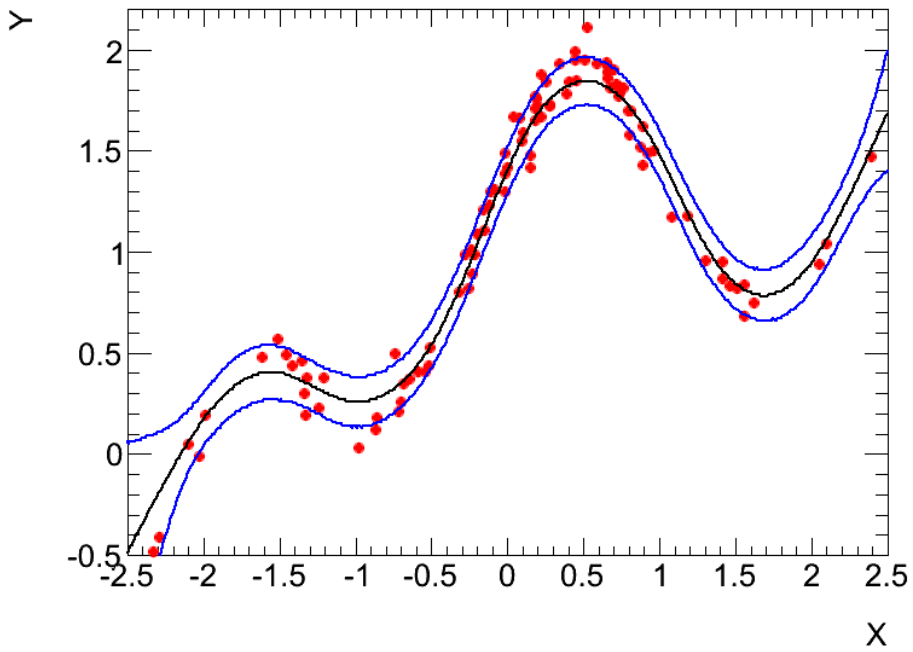
- Obliczanie średniej: próbkowanie z użyciem łańcuchów Markowa.
- **Zalety:**
  - Otrzymujemy błąd estymowanej funkcji,
  - Zwiększona odporność na przetrenowanie i fluktuacje.

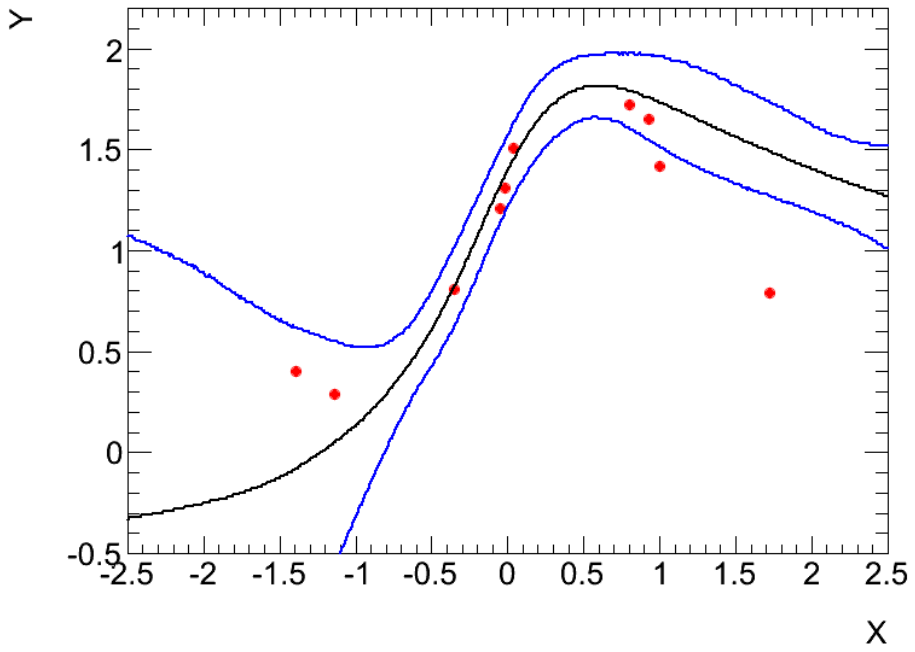
Przykład BNN

Jak bayesowska sieć o 8 węzłach działa przy różnej ilości danych?

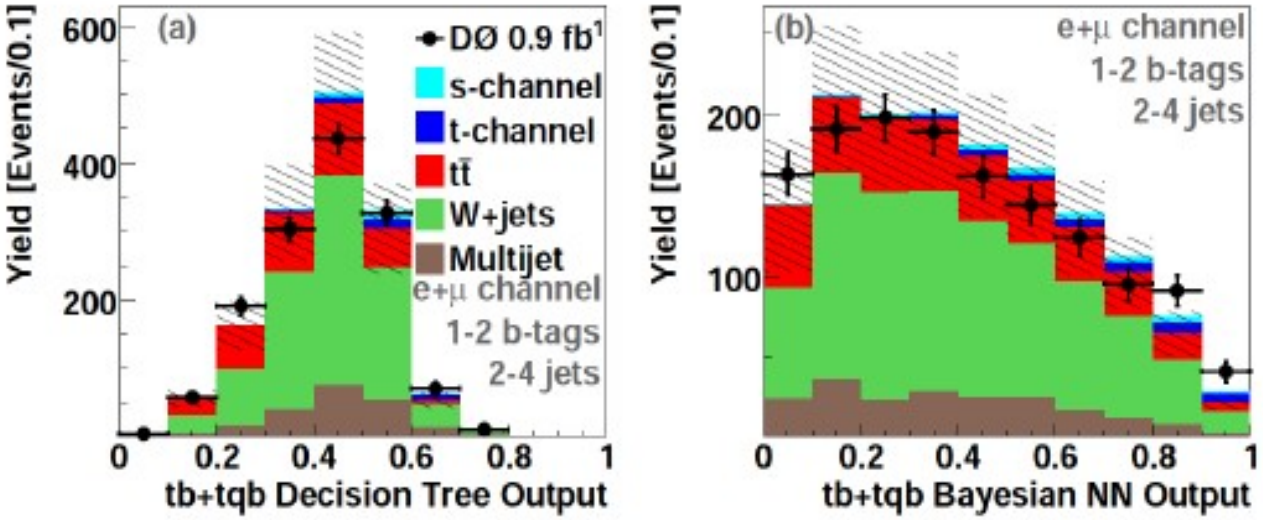
- Dane generowane za pomocą funkcji: $y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2)$ z gaussowskim szumem o odchyleniu standardowym 0.1
- 400 sieci neuronowych, z rozkładu odpowiedzi sieci wyznaczamy medianę oraz 10% Qnt i 90% Qnt (10% sieci dało odpowiedź o wartośc poniżej dolnej niebieskiej linii oraz 10% powyżej górnej niebieskiej linii).
- Gdy użyliśmy zbioru treningowego tylko o 10 punktach błędy bardzo wzrosły (czyli tak jak powinny).





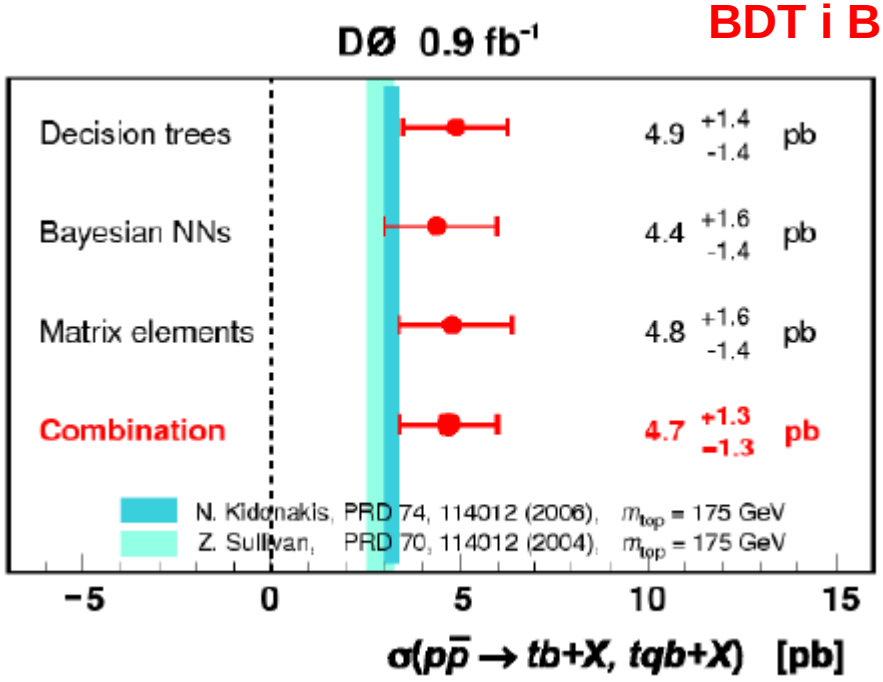


Przykład – poszukiwanie pojedynczego kwarku top w eksperymencie D0



Zbliżona dokładność BDT i BNN

Analiza z użyciem:
wzmocnionych drzew decyzyjnych (BDT) oraz bayesowskich sieci neuronowych



D0 Collaboration,
PRD 78 012005, 2008

Bayesian Neural Network Revival (some recent papers)

- A. Honkela and H. Valpola. Variational learning and bits-back coding: An information-theoretic view to Bayesian learning. IEEE Transactions on Neural Networks, 15:800-810, 2004.
- Alex Graves. Practical variational inference for neural networks. In NIPS 2011.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In ICML, 2015.
- José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In ICML, 2015.
- José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato, Thang Bui, and Richard E Turner. Black-box alpha divergence minimization. In Proceedings of The 33rd International Conference on Machine Learning, pages 1511-1520, 2016.
- Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. ICML, 2016.
- Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. NIPS, 2016.

Zadanie dla ambitnych

https://www.kaggle.com/c/higgs-boson

Open Higgs challenge!!!!!!!!!!!!!!

Machine Learning and HEP

- 90'ies - Neural Nets used by LEP experiments
- BDT (Adaboost) invented in 97
- Machine Learning used extensively at D0/CDF (mostly BDT, also Neural Nets) in the 00'ies
- Last years – mostly BDT built in TMVA ROOT package (popular among physicists). Neural Nets and other techniques treated as obsolete.

- **Not much work within LHC experiments on studying possible better MVA techniques.**
- **Enormous development of Machine Learning in the outside world in the last 10 years ("Big Data", "Data Science", even "Artificial Intelligence" is back).**

- **We have to catch up and learn from computer scientists:**

Make an open Higgs challenge!

- **Task: identify H->tau tau signal out of background in the simulated data.**

How did it work ?

- People register to Kaggle web site hosted https://www.kaggle.com/c/higgs-boson . (additional info on https://higgsml.lal.in2p3.fr).
- …download training dataset (with label) with 250k events
- …train their own algorithm to optimize the significance (à la s/sqrt(b))
- …download test dataset (without labels) with 550k events
- …upload their own classification
- The site automatically calculates significance. Public (100k events) and private (450k events) leader boards update instantly. (Only the public is visible)
- 1785 teams (1942 people) have participated
- most popular challenge on the Kaggle platform (until a few weeks ago)
- 35772 solutions uploaded

Funded by: Paris Saclay Center for Data Science, Google, INRIA

Final leaderboard

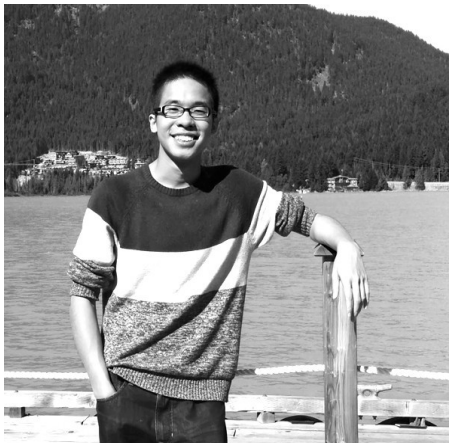
#	Δrank	Team Name ‡ model uploaded * in the money		Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑1	Gábor Melis ‡ *	7000$	3.80581	110	Sun, 14 Sep 2014 09:10:04 (-0h)
2	↑1	Tim Salimans ‡ *	4000$	3.78913	57	Mon, 15 Sep 2014 23:49:02 (-40.6d)
3	↑1	nhlx5haze ‡ *	2000$	3.78682	254	Mon, 15 Sep 2014 16:50:01 (-76.3d)
4	↑38	ChoKo Team		3.77526	216	Mon, 15 Sep 2014 15:21:36 (-42.1h)
5	↑35	cheng chen		3.77384	21	Mon, 15 Sep 2014 23:29:29 (-0h)
6	↑16	quantify		3.77086	8	Mon, 15 Sep 2014 16:12:48 (-7.3h)
7	↑1	Stanislav Semenov & Co (HSE Yandex)		3.76211	68	Mon, 15 Sep 2014 20:19:03
8	↓7	Luboš Motl's team	Best physicist	3.76050	589	Mon, 15 Sep 2014 08:38:49 (-1.6h)
9	↑8	Roberto-UCIIIM		3.75864	292	Mon, 15 Sep 2014 23:44:42 (-44d)
10	↑2	Davut & Josef		3.75838	161	Mon, 15 Sep 2014 23:24:32 (-4.5d)
45	↑5	crowwork ‡	HEP meets ML award XGBoost authors Free trip to CERN	3.71885	94	Mon, 15 Sep 2014 23:45:00 (-5.1d)
782	↓149	Eckhard		3.49945	29	Mon, 15 Sep 2014 07:26:13 (-46.1h)
991	↑4	Rem.		3.20423	2	Mon, 16 Jun 2014 21:53:43 (-30.4h)

The winners

- See http://atlas.ch/news/2014/machine-learning-wins-the-higgs-challenge.html
- 1 : **Gabor Melis** (Hungary) software developer and consultant : wins 7000$.
- 2 : **Tim Salimans** (Neitherland) data science consultant: wins 4000$
- 3 : **Pierre Courtiol** (nhlx5haze) (France) ? : wins 2000$
- HEP meets ML award: (team crowwork), **Tianqi Chen** (U of Washington PhD student in Data Science) and **Tong He** (graduate student Data Science SFU). Provided **XGBoost public software** used by many participants.

https://github.com/dmlc/xgboost

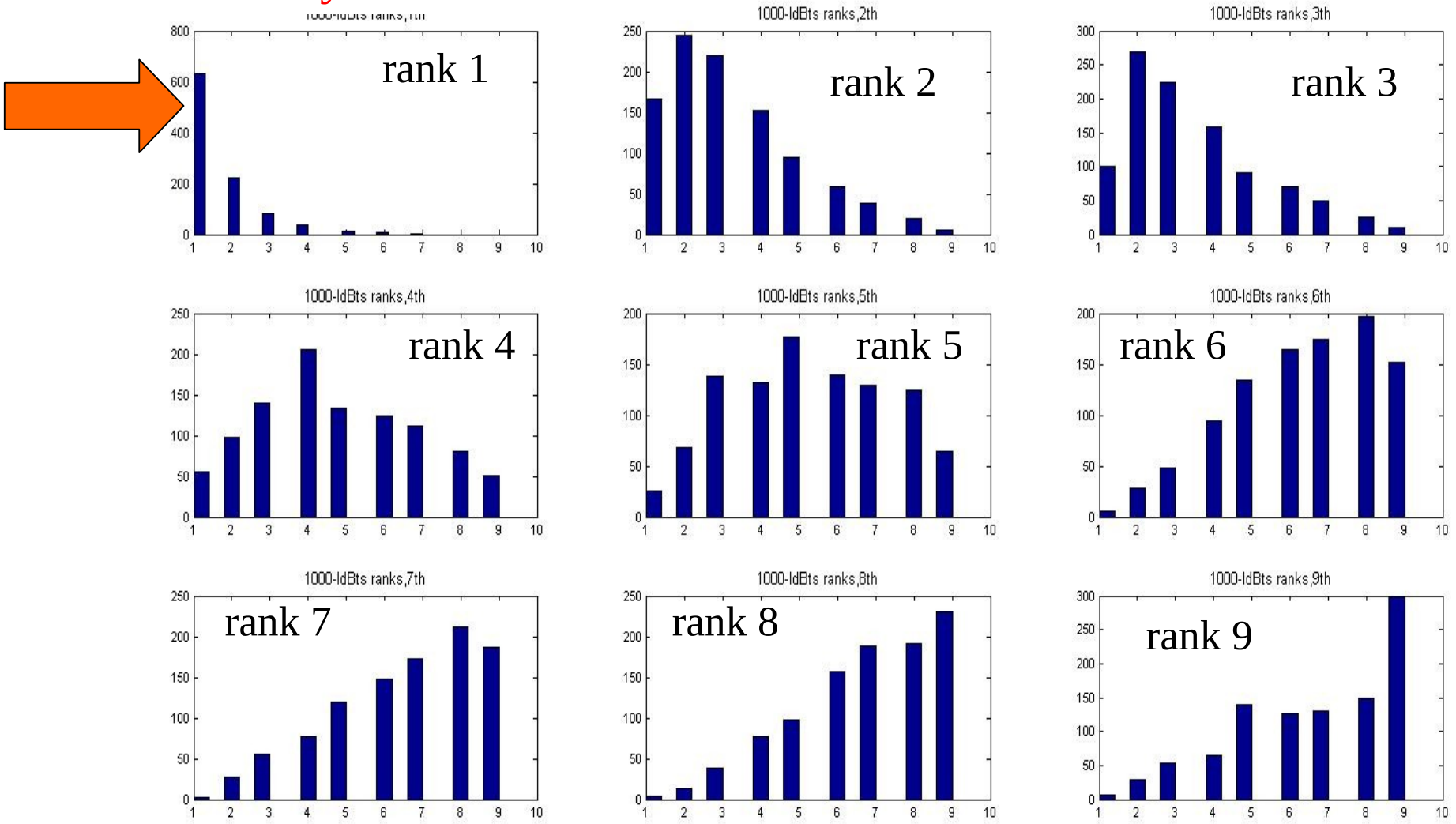
?



Rank distribution after bootstrap

Distribution of rank of participant of rank i after 1000 bootstraps of the test sample.

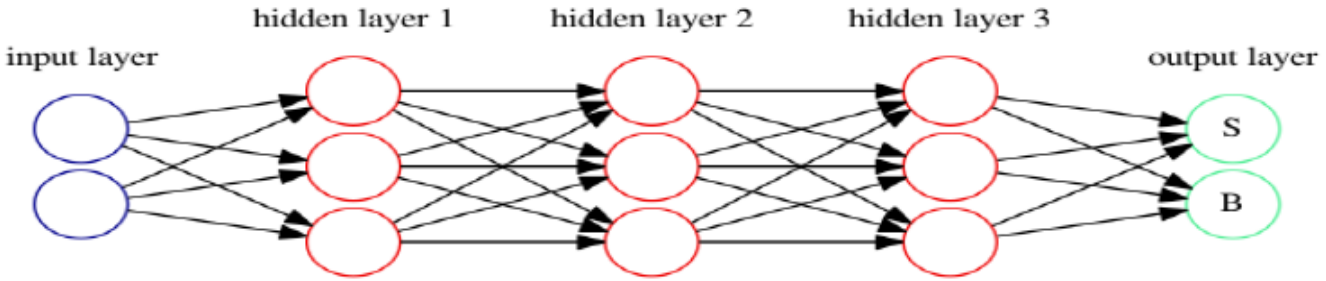
! Gabor clearly better




Who are the
XGBoost)

Deep neural network

- Hierarchical feature extraction – first build abstract objects, than find dependencies between them.
- Deep neural network (DNN)- an artificial neural network with multiple hidden layers of units between the input and output layers.
- Extra layers - composition of features from lower layers, potential of modeling complex data with fewer units than a similarly performing shallow network.



- inputs: normalized features (~30), some log transformed
- 3 hidden layers of 600 neurons each
- output layer: 2 softmax units (one for signal, one for background)
- activation function: “max channel” in groups of 3
- trained to minimize cross entropy
- regularization: dropout on hidden layers, $L_1 + L_2$ penalty and a mild sparsity constraint input weights

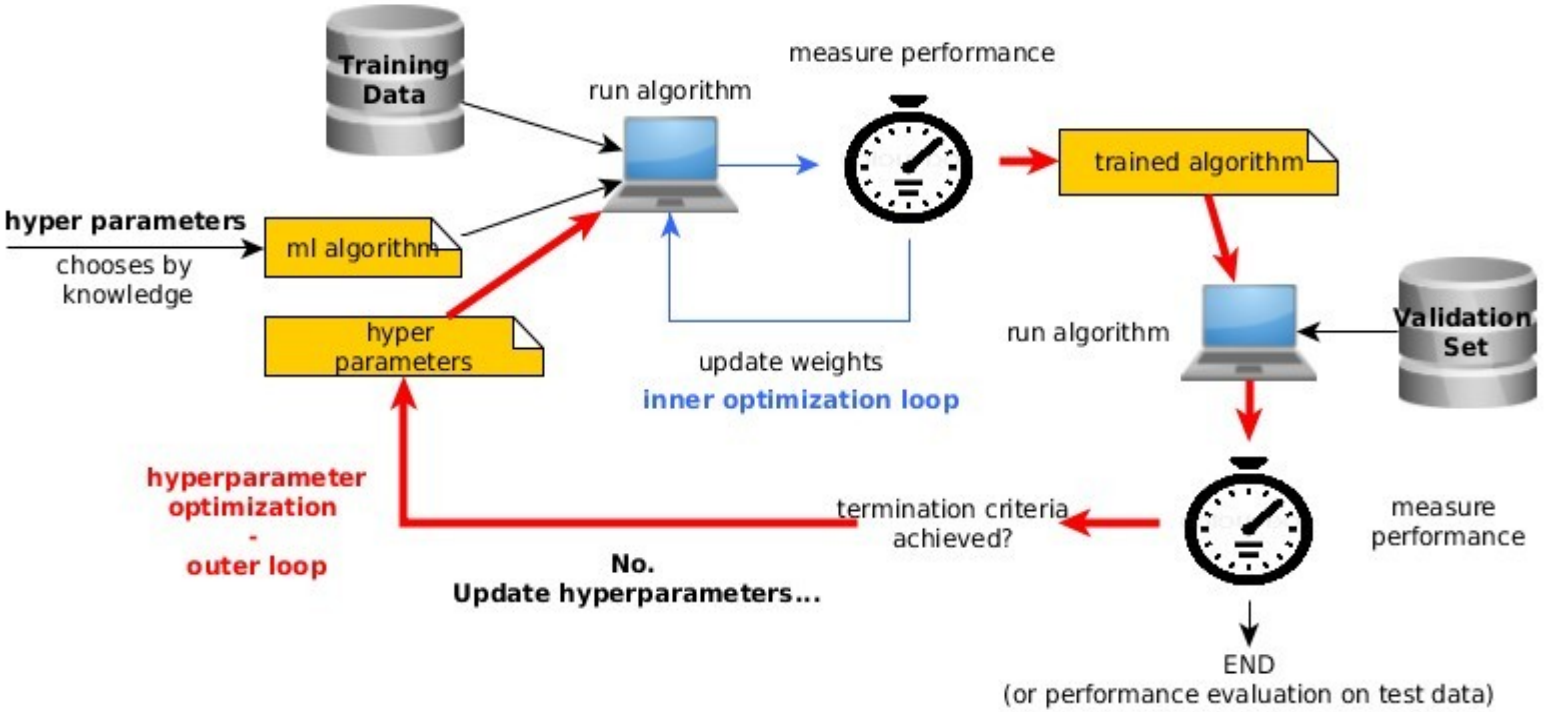
Challenge winning
Gabor's deep neural network
(from Gabor's presentation)

- CV bagged NNs: 3.83
- CV bagged xgboost: 3.79

Remark:
Few years ago some experts claimed neural networks are an obsolete tool :)

Automatic optimization of hyperparameters

- Manual optimization of NN (or any other method) is time consuming.
- Fortunately the Bayesian optimization methods can rival and surpass human domain experts in finding good hyperparameter settings.
- SMAC, SPEARMINT, TPE (and others) are doing that with great success: http://www.cs.ubc.ca/~hutter/papers/13-BayesOpt_EmpiricalFoundation.pdf



Analiza podczas praktyk studenckich

- Próbowaliśmy powtórzyć HiggsChallenge podczas praktyk studenckich.
- Udało się za pomocą TMVA (konwersja danych do formatu root) oraz pakietu XGBoost
- Optymalizacja parametrów XGBoost za pomocą programu hyperopt

A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss (2009)
TMVA 4 Package Documentation
https://tmva.sf.net

Tianqi Chen, Tong He, Bing Xu and Michael Benesty (2014)
XGBoost Package Documentation
https://github.com/dmlc/xgboost

James Bergstra, Dan Yamins, and David D. Cox (2013)
Hyperopt Package Documentation
https://github.com/hyperopt

Rozwiązania Kaggle-Higgs vs Hyperopt

Porównianie wyników uzyskanych przez nas automatycznie z wynikami z najlepszymi znalezionymi parametrami dla XGBoost.

Kto	9. K-H	M. Wolter	Nasze obliczenia
Maks. głębokość	9	10	9
Wsp. uczenia	0.01	0.089	0.059
Liczba drzew	3000	150/250/500	300
Liczba testów	-	300	100
Sub_sample	0.9	1	0.9
Maks. ROC	0.987	0.933/0.934/0.933	0.934

Sub_sample - jaka cześć danych brana jest do procesu uczenia - wprowadza pewną losowość i zapobiega przeuczaniu

Jak widać wyniki przez nas osiągnięte są znacznie słabsze. Prowadziliśmy poszukiwania w innym regionie parametrów.