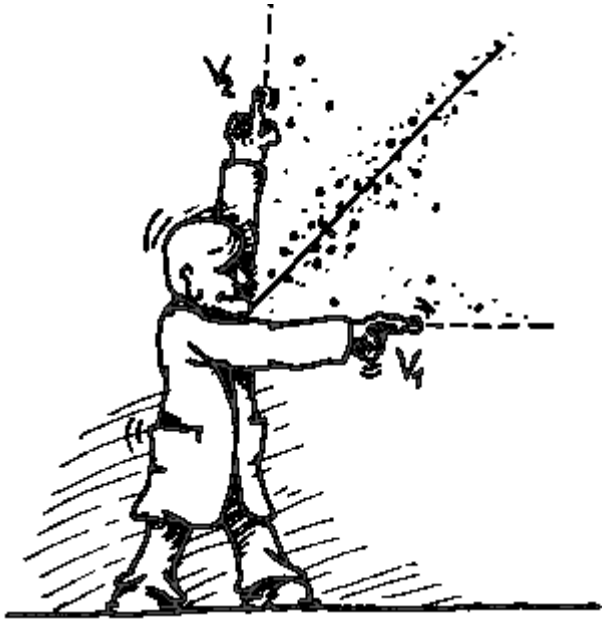
Algorytmy uczące się

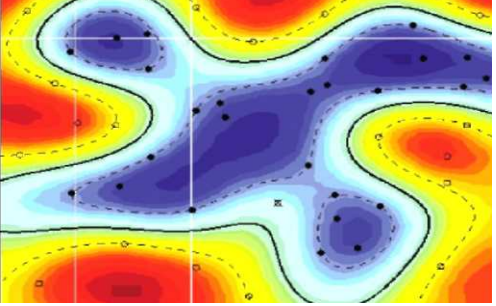
w fizyce cząstek i nie tylko

Marcin Wolter
Instytut Fizyki Jądrowej PAN, Kraków



Seminarium
7 grudnia 2012, Warszawa

Jest to temat mojej rozprawy habilitacyjnej

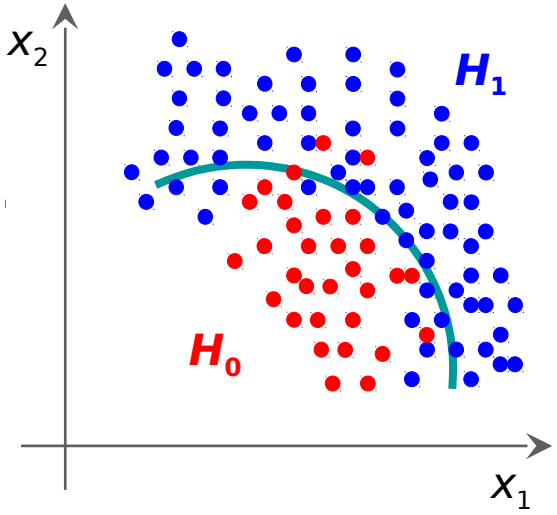


Algorytmy uczące się, algorytmy wielu zmiennych

- **Dlaczego uczące się?**
  - Bo chcemy, aby algorytm sam nauczył się na przykładach.
  - Bo nie znamy zależności funkcyjnych pomiędzy zmiennymi – ale dodana informacja pod postacią zależności funkcyjnych poprawia skuteczność algorytmu.
- **Dlaczego wielu zmiennych?**
  - Bo chcemy optymalnie wykorzystać całą dostępną informację. Rozkłady prawdopodobieństwa wielu zmiennych zawierają więcej informacji niż rozkłady dla każdej zmiennej z osobna.
- **Do czego można stosować?**
  - Klasyfikacja (np. sygnał i tło),
  - Regresja – aproksymacja funkcji ciągłej,
  - Inne, jak selekcja zmiennych, aproksymacja prawdopodobieństwa...
- **W jakich dziedzinach można stosować?**
  - W wielu: fizyka, chemia, meteorologia, finanse, rozpoznawanie obrazów, analiza tekstów...

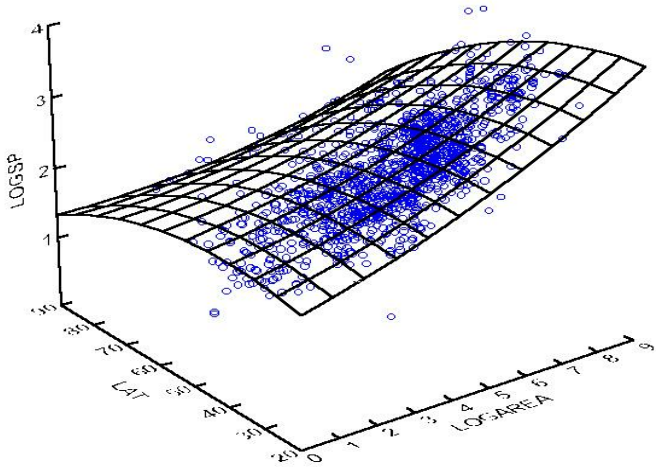
Rodzaje algorytmów

czyli jak spożytkować dostępną informację?



- **Klasyfikacja:** znalezienie funkcji ***f(x1,x2)*** zwracającej prawdopodobieństwo, że dany punkt należy do danej klasy (np. klasy ,,sygnał'').

- **Regresja:** dopasowanie ciągłej funkcji (np. rekonstrukcja energii cząstki na podstawie wskazań kilku detektorów).



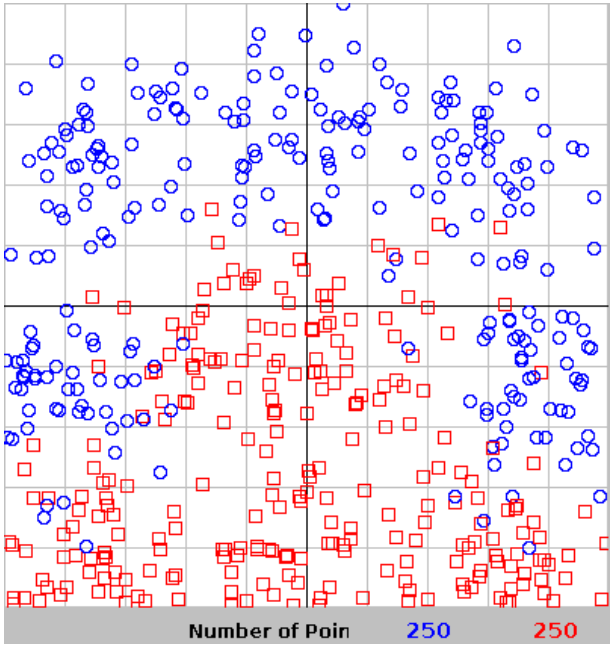
Kilka popularnych metod

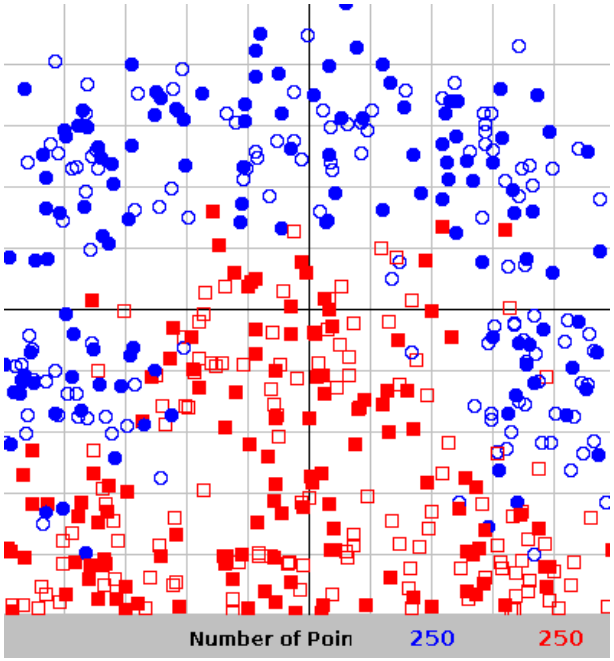
- **Metody liniowe**
  - Cięcia
  - Dyskryminanty Fishera
- **Metody nieliniowe**
  - Naiwny klasyfikator bayesowski,
  - Estymatory gęstości prawdopodobieństwa, (Probability Density Estimator),
  - Metoda najbliższych sąsiadów,
  - PDE_RS (Probability Density Estimator with Range Searches),
  - Sieci neuronowe,
  - Bayesowskie sieci neuronowe,
  - Wzmocnione drzewa decyzyjne (Boosted Decision Trees),
  - Maszyny wektorów wspierających (Support Vector Machines).
- **I wiele innych...**
  - analiza składowych, niezależnych (Independant Component Analysis)
  - analiza składowych głównych.

- **Dlaczego jest ich tak dużo???**
- **Czym się różnią i do czego służą?**
- **Czy naprawdę potrzebujemy ich wszystkich?**

Jak wytrenować algorytm?

Uczenie z nauczycielem



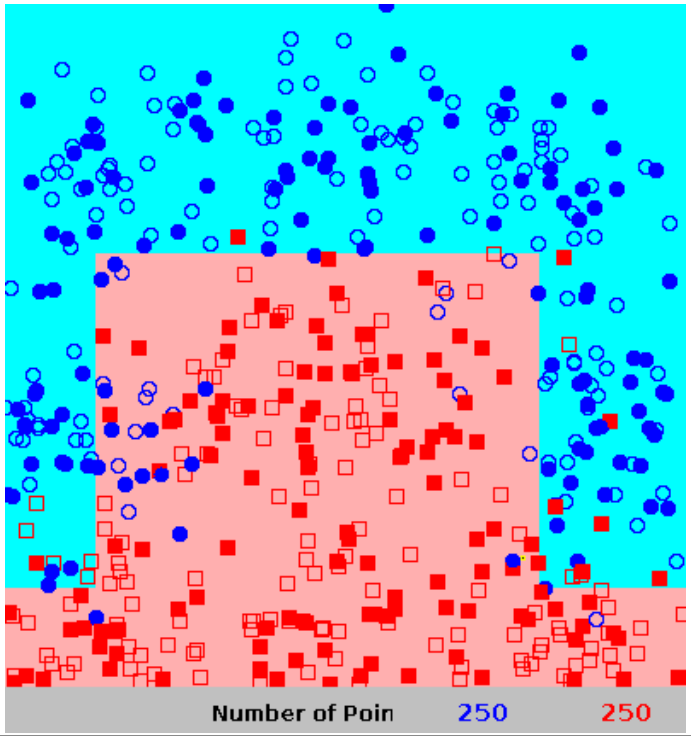


- Potrzebne są **dane treningowe**, dla których znamy poprawną odpowiedź, np. czy jest to sygnał czy tło (np. z symulacji komputerowej).
- Dzielimy ten zbiór danych na **dwie części** – jedną używamy do **treningu**, drugą do **sprawdzenia** jego wyników.
  - **Dlaczego zbiór treningowy i testowy?**

    Chcemy sprawdzić,czy wynik uczenia jest zbliżony dla zbioru użytego do treningu i drugiego, niezależnego statystycznie. Istotne w przypadku przetrenowania (o tym za chwilę).

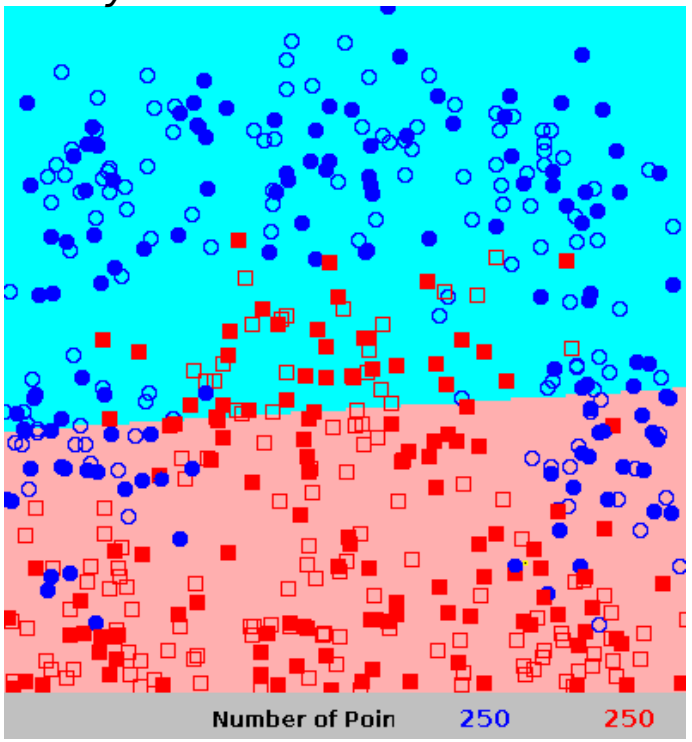
- Klasyfikacja – znalezienie funkcji ***f(x)*** $\in$ ***F*** (***F***-klasa funkcji) najlepiej opisującej prawdopodobieństwo, że dany przypadek jest sygnałem.
- Problem minimalizacji funkcji straty (np. $\chi^2 = \sum (f(x) - y_{true})^2$, gdzie $y_{true} = \pm 1$).
- W granicy N → ∞ funkcja ***f(x)*** będzie prawdziwym rozkładem prawdopodobieństwa.
- Poszczególne algorytmy różnią się: klasą funkcji ***F*** (np. liniowe, nieliniowe), funkcją straty, sposobem jej minimalizacji.

Cięcia



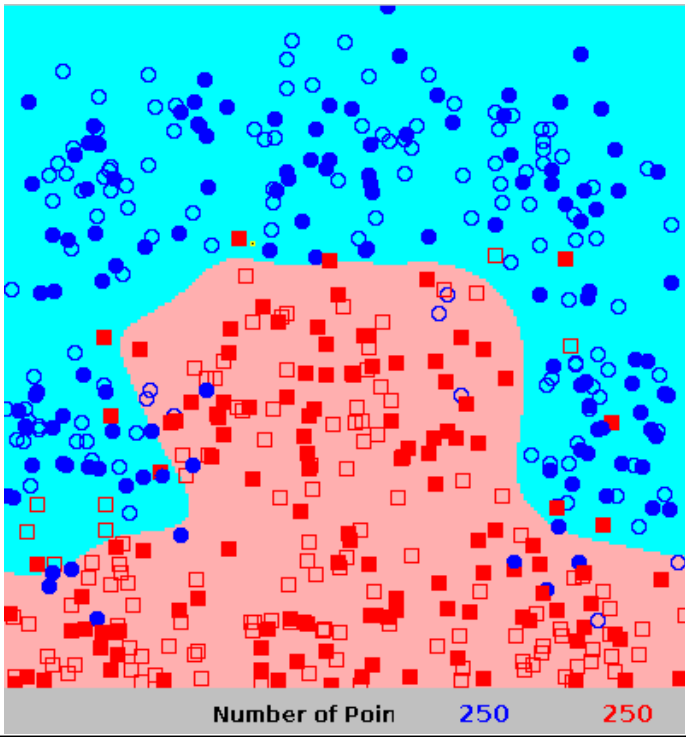
Klasyfikator liniowy

Klasyfikator Fishera

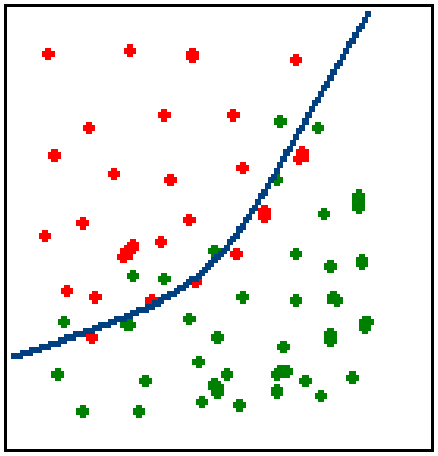


Klasyfikator nieliniowy

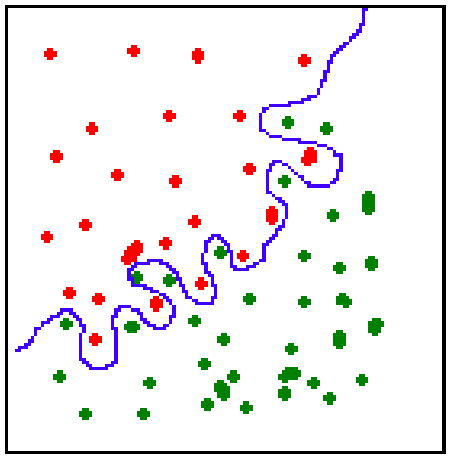
Sieć neuronowa



Przetrenowanie

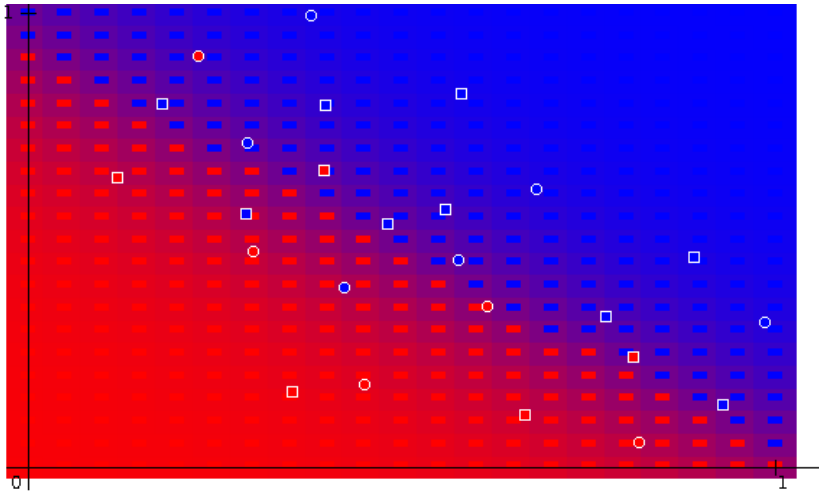


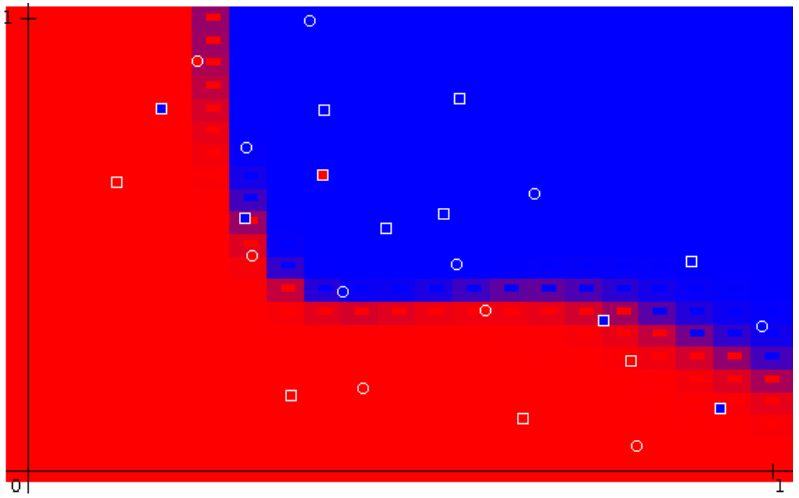
Poprawnie

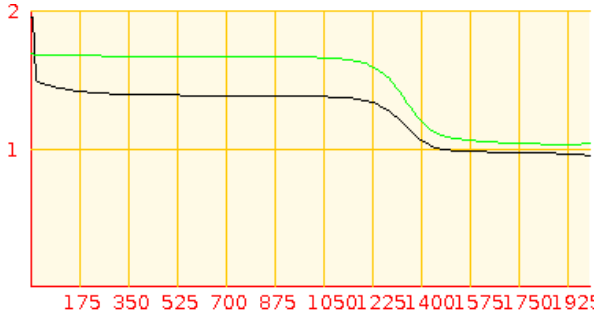


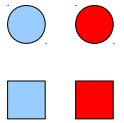
Przetrenowanie

- **Przetrenowanie** – algorytm “uczy się” poszczególnych przypadków, a nie ogólnych zasad.
- Efekt występuje we wszystkich metodach uczących się.
- Remedium – kontrola z użyciem dodatkowego zbioru danych.



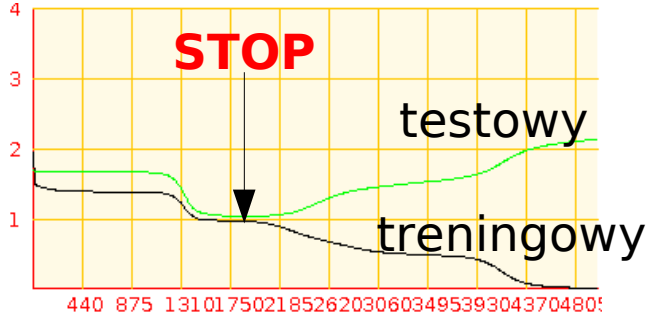






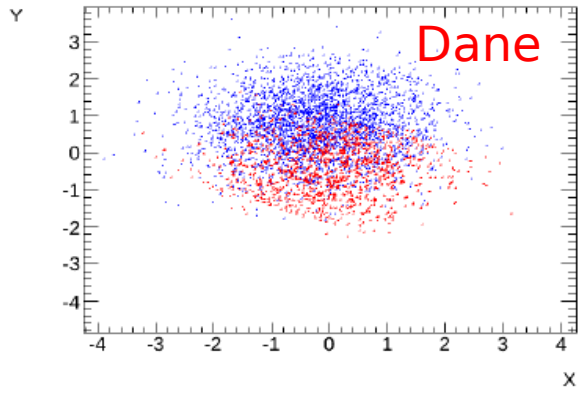
zbiór treningowy

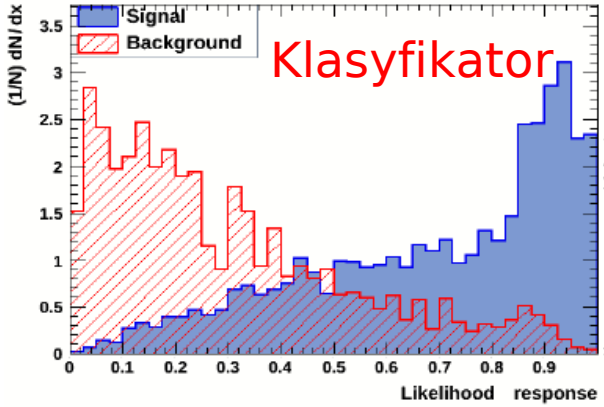
zbiór testowy

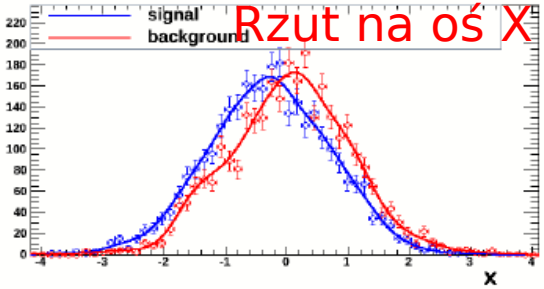


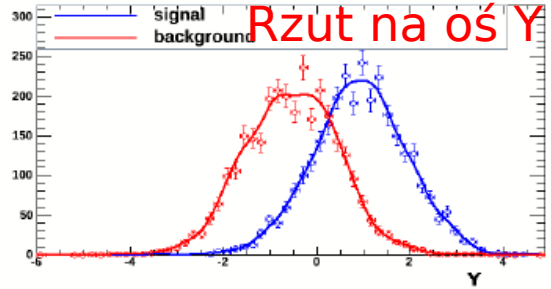
Przykład z użyciem sieci neuronowej.

Naiwny klasyfikator bayesowski









Nazywany także metodą rzutowanych prawdopodobieństw, ,,projected likelihood".

- Oparty na założeniu o wzajemnej <u>niezależności</u> zmiennych (dlatego ,,naiwny"):

$$P_i(S) = \frac{\mathcal{L}_i(S)}{\mathcal{L}_i(S) + \mathcal{L}_i(B)}$$

$$\mathcal{L}_i(S) = \prod_{k=1}^{N} p_k(S)(x_i)$$

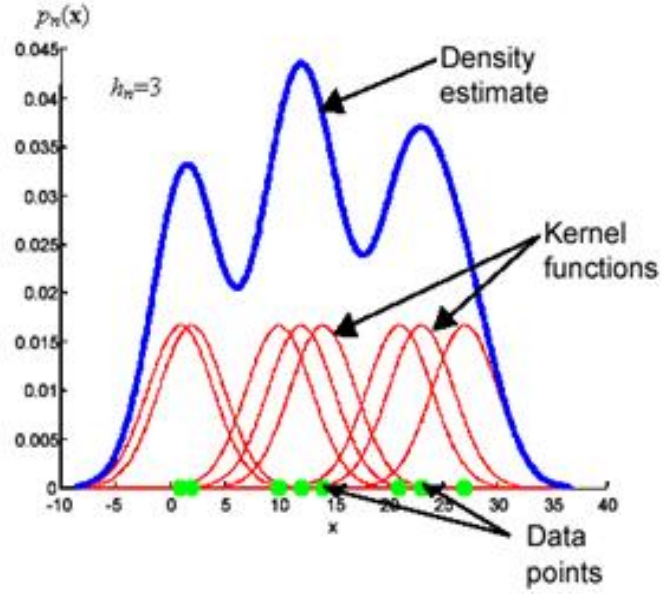
$$\mathcal{L}_i(B) = \prod_{k=1}^{N} p_k(B)(x_i)$$

- Wynikowe prawdopodobieństwo, że sygnał (tło) jest iloczynem prawdopodobieństw dla poszczególnych zmiennych.
- Szybki i stabilny, w wielu przypadkach dobrze sprawdza się przy klasyfikacji danych.

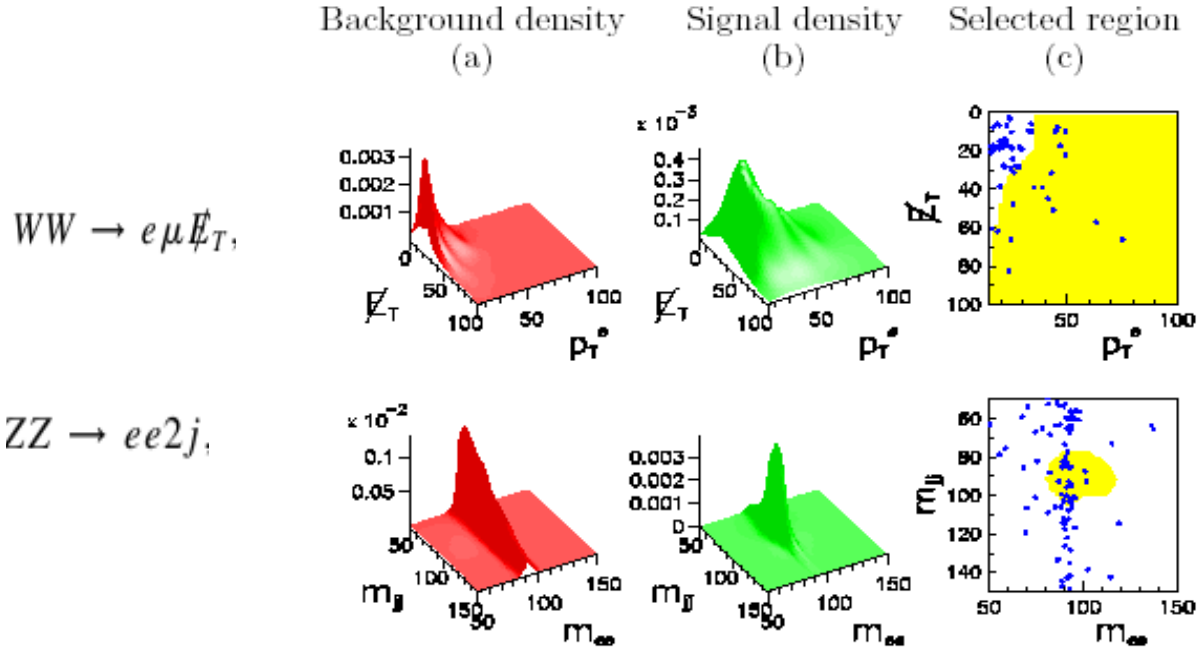
Jądrowe estymatory gęstości

Aproksymacja nieznanego rozkładu prawdopodobieństwa jako **sumy funkcji jądrowych** (kernel) umieszczonych w punktach x_n zbioru treningowego (Parzen, lata 1960-te).

$$D(x) = \frac{P(S)}{P(S) + P(B)}$$



- Typowe funkcje jądrowe: Gauss, $1/x^n$ itp.
- Proste pojęciowo, ale użycie tej metody wymaga dużo czasu procesora i pamięci.



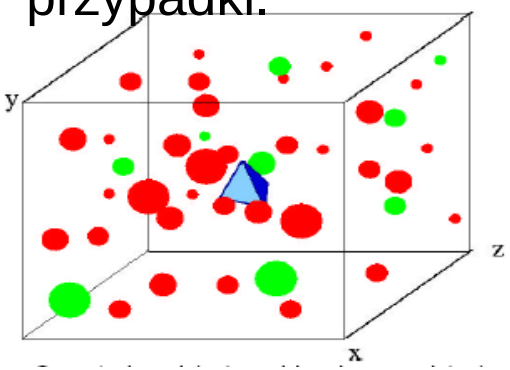
Pakiet QUAERO, eksperyment D0.

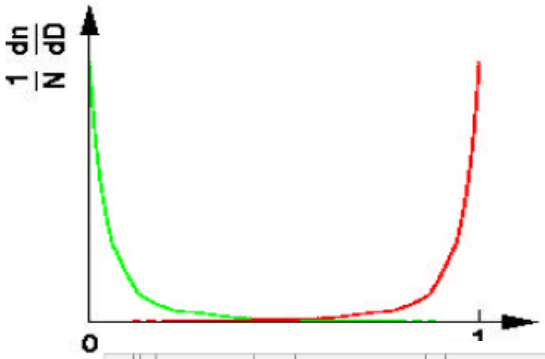
D0 Collaboration, V. M. Abazov i inni, Search for new physics using QUAERO: A general interface to DØ event data, Phys. Rev. Lett. 87 (2001) 231801,

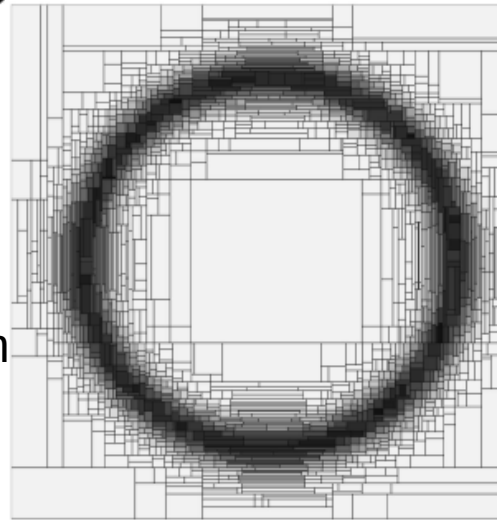
Jak przyspieszyć? Metoda PDE_RS

***PDE_RS** - Probability Density Estimation with Range Searches.*

- Zlicz przypadki sygnału (ns) i tła (nb) w N-wymiarowej kostce wokół klasyfikowanego, nowego przypadku – potrzeba tylko kilku przypadków z całego zbioru treningowego.
- Rozmiar hiperkostki podlega optymalizacji (parametr metody).
- Dyskryminator D(x) dany jest jako: $D(x) = \frac{n_S}{n_S + n_b}$
- Przypadki zapisane w drzewie binarnym – szybciej znajdywane są sąsiednie przypadki.

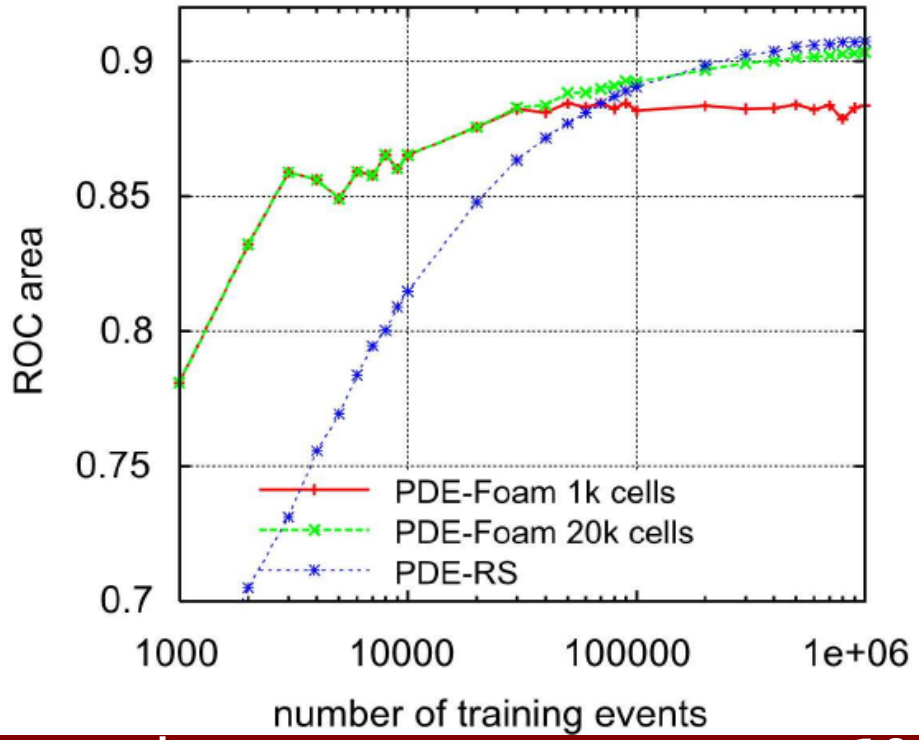




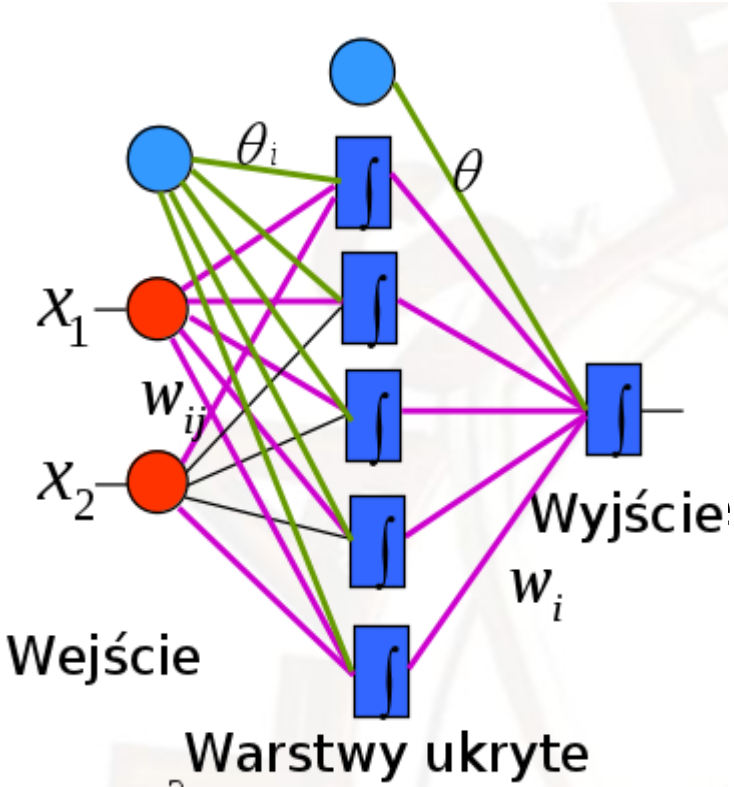


Dalsze usprawnienie:
podzielić przestrzeń tak, aby rozmiar poszczególnych obszarów odzwierciedlał gęstość przypadków - algorytm **FOAM** prof. Stanisława Jadacha.

Opublikowana przez T. Carli, B. Koblitz, NIM A 501 (2003) 576-588, używana np. przez eksperyment HERA



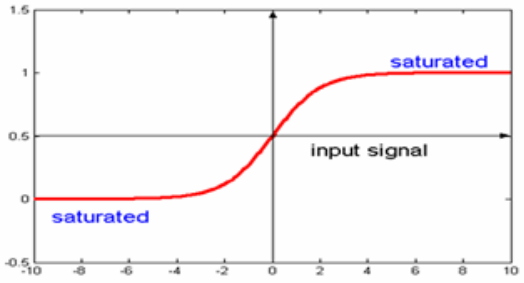
Sieci neuronowe



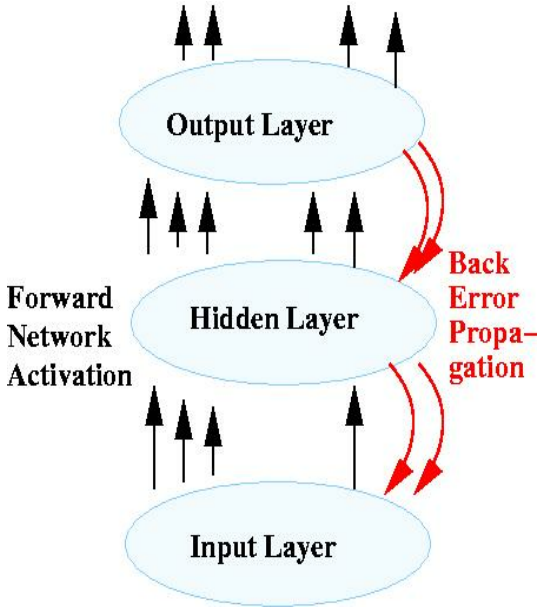
$$a_i = \sum_{j=1}^{2} w_{ij} x_j + \theta_i \quad \rightarrow f(a_i)$$

$$n(x,w) = f\left(\sum_{i=1}^{5} w_i f(a_i) + \theta\right)$$

Funkcja aktywacji

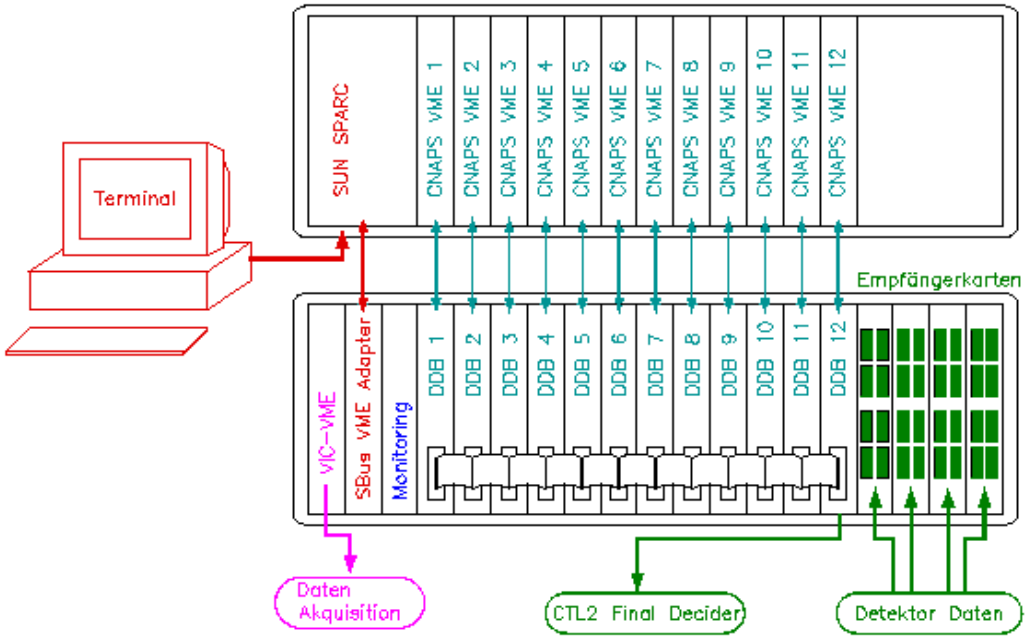


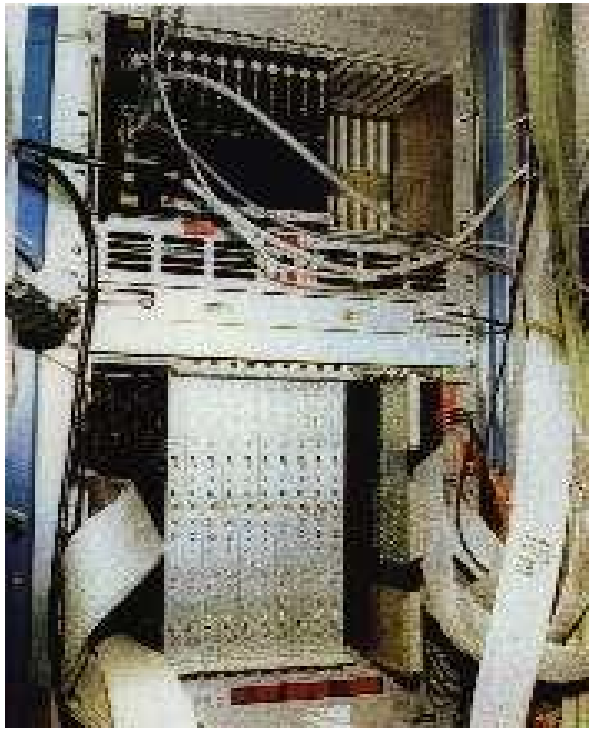
- Każdy węzeł: na wyjściu uzyskujemy sumę wejść z wagami transformowaną przez funkcję aktywacji.
- Jak wytrenować sieć wielowarstwową? Jak dopasować wagi w warstwach ukrytych? Ten problem wstrzymywał rozwój algorytmów sieci neuronowych przez prawie 30 lat.
- Rozwiązanie – propagacja do tyłu (**backpropagation**). Różnica między wartością oczekiwaną a otrzymaną na wyjściu y-*f(x,w)* jest propagowana wstecznie przez sieć używając aktualnych wag („rewolucja” lat '80).



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).



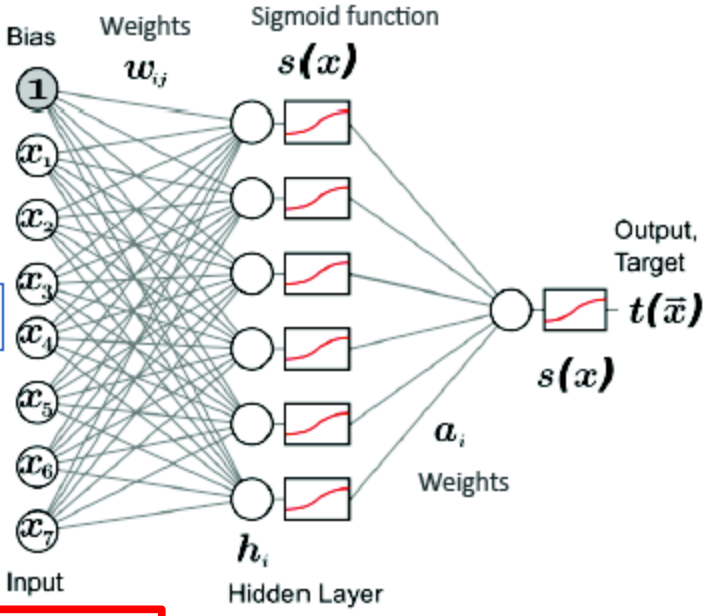


J. K. Kohne i inni,
Realization of a second level neural network trigger for the H1 experiment at HERA,
Nucl. Instrum. Meth. A389 (1997)

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.

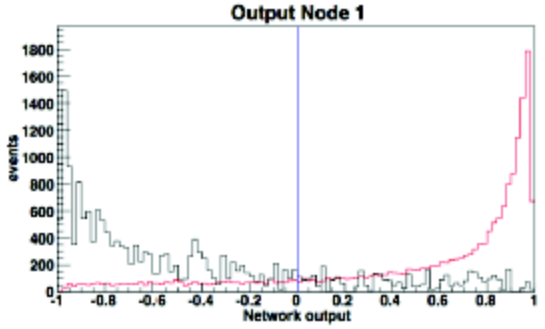


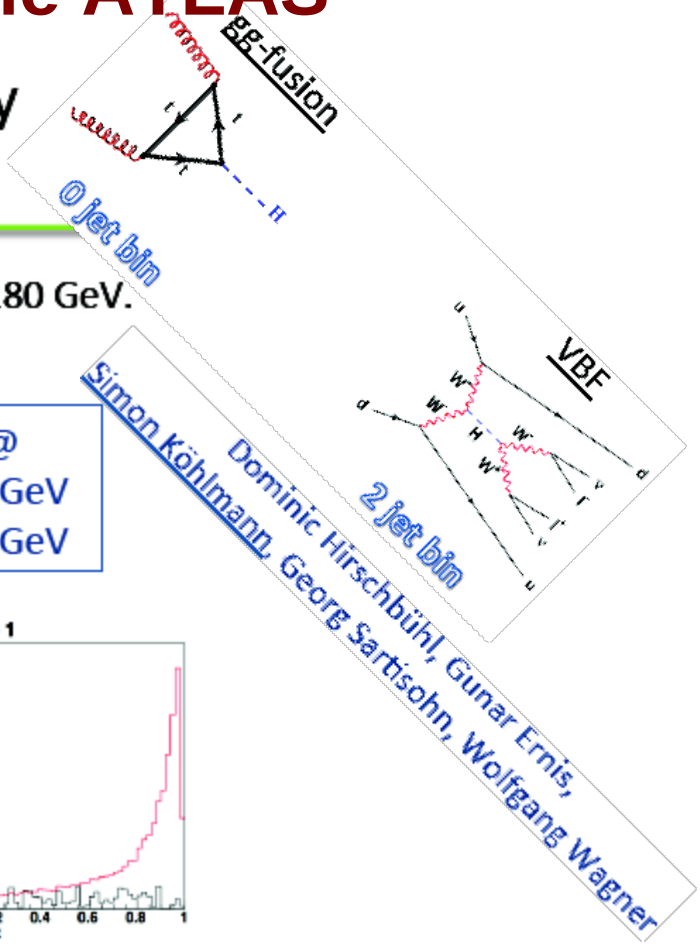
input variables

6 hidden nodes

Implementation with NeuroBayes

Training @
m_H = 150 GeV
m_H = 180 GeV





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

06 Dec 2012

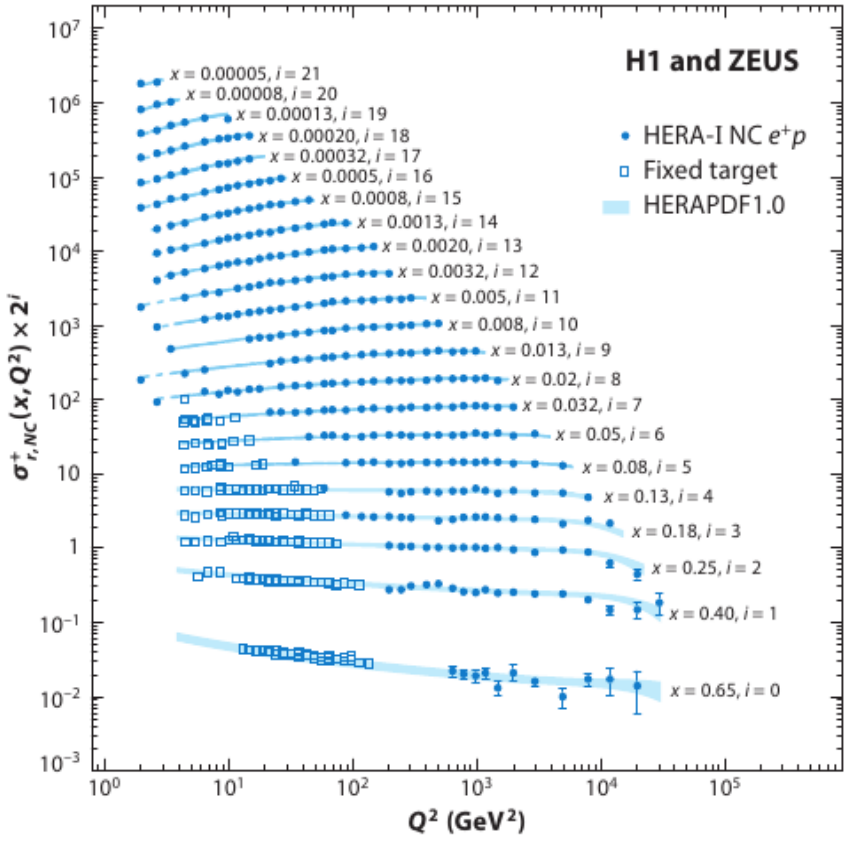
06 Dec 2012

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.

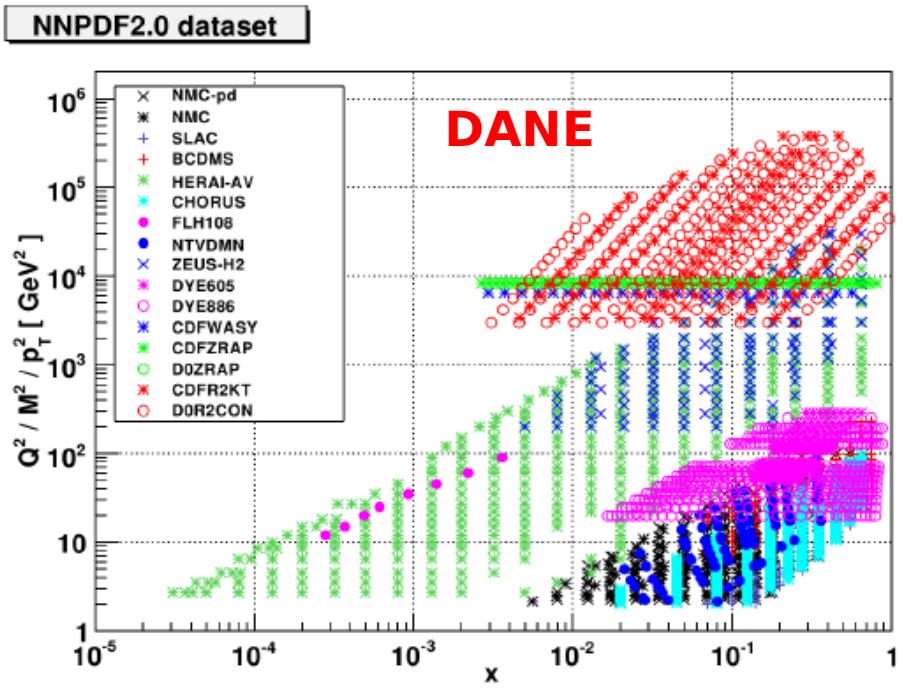
- **Stworzenie rozkładów gęstości prawdopodobieństwa partonów**

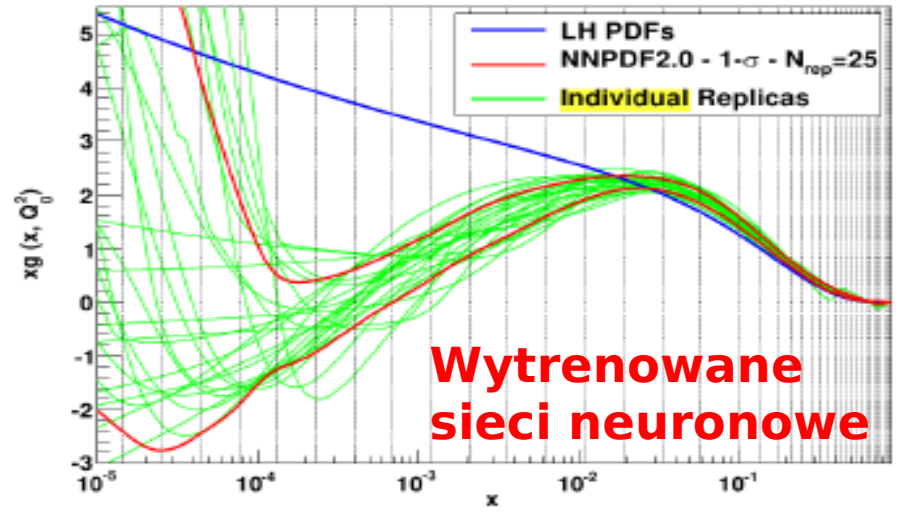
  Dopasowanie sieci neuronowych, po jednej dla każdej repliki.

- **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

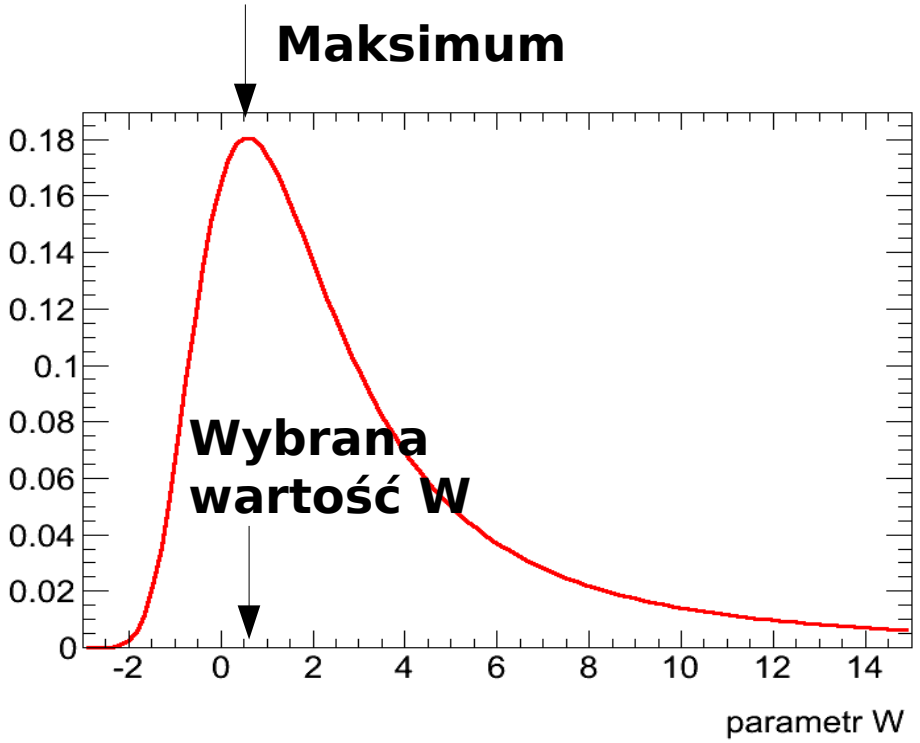
Dla każdej funkcji f(x) z przestrzeni funkcji F znajdujemy prawdopodobieństwo *a posteriori* $\boldsymbol{p(f \mid T)}$ używając zbioru treningowego $\boldsymbol{T= (x, y)}$.

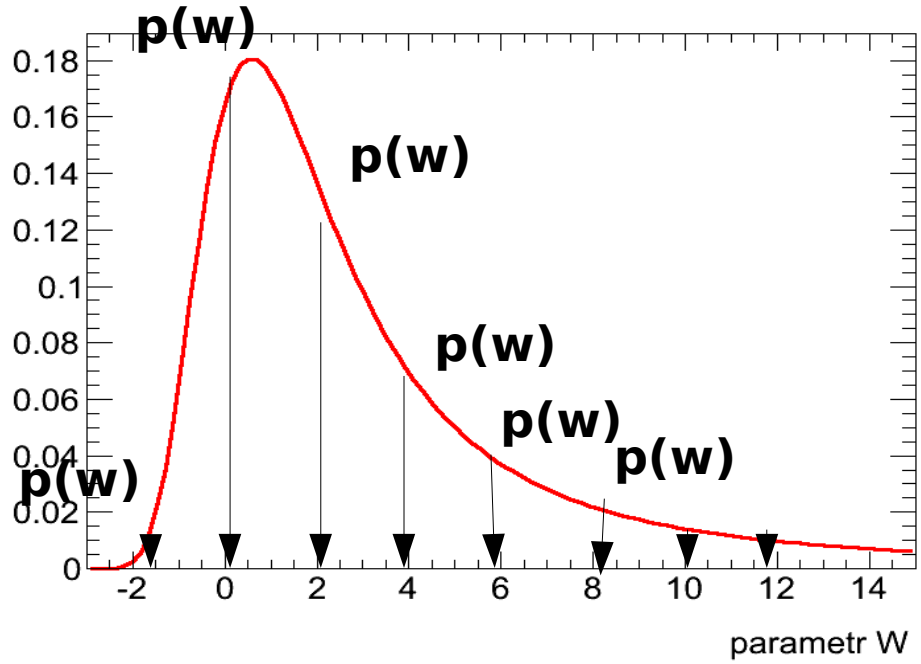
W uczeniu bayesowskim NIE ZNAJDUJEMY jednej, najlepszej funkcji, ale używamy wielu funkcji ważonych ich prawdopodobieństwem.

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

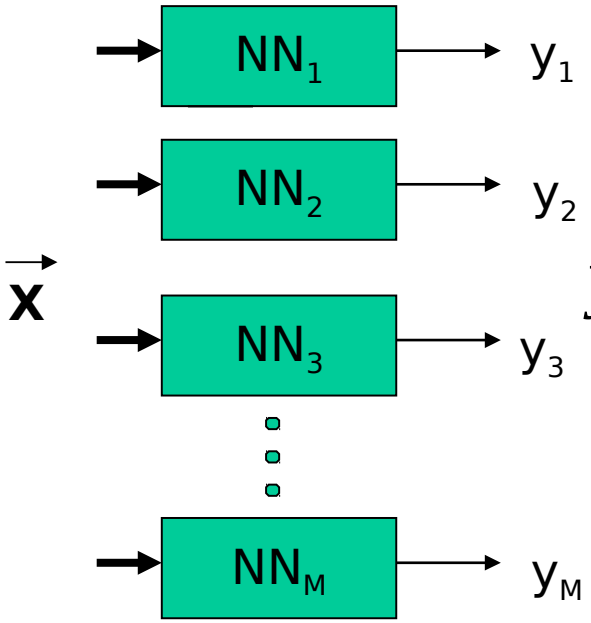
Każda funkcja (lub wartość parametru) ma przypisane prawdopodobieństwo (wagę).

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.



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

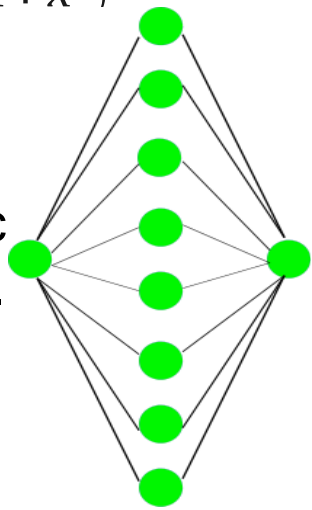
C.M. Bishop
"Neural Networks for Pattern Recognition",
Oxford 1995

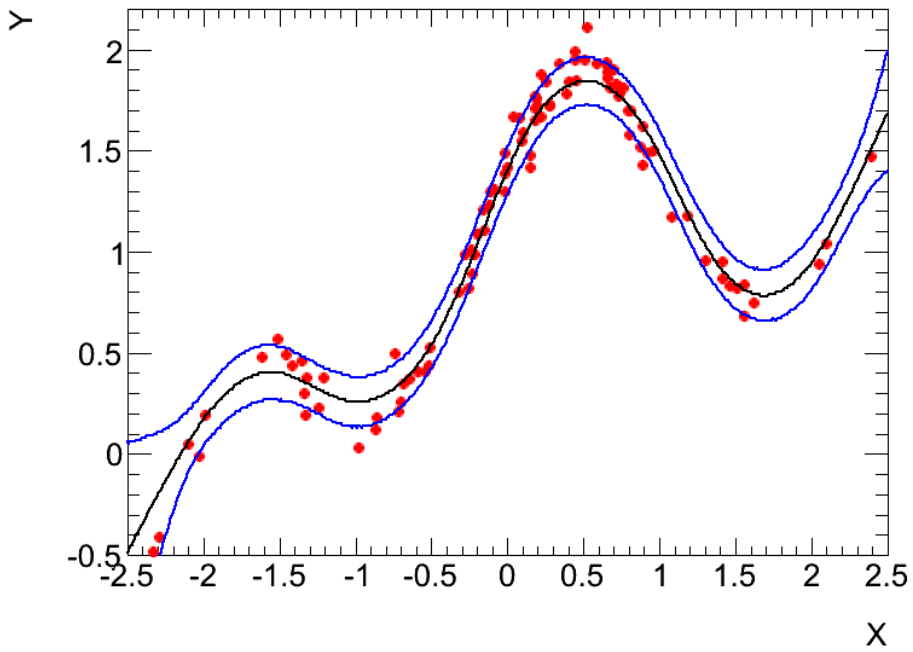
Darmowe oprogramowanie (używane np. przez kolaborację D0):
Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html

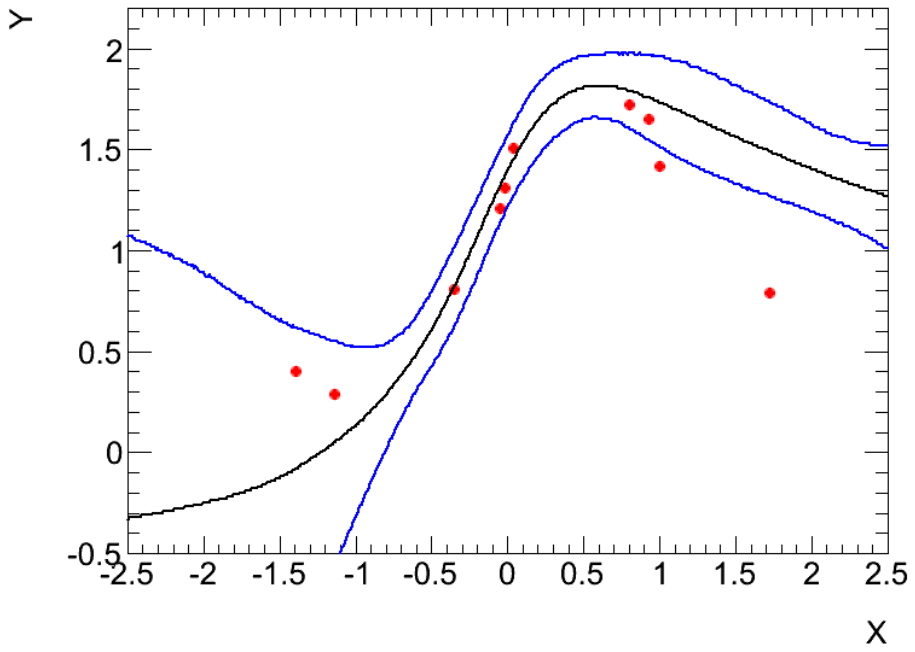
Przykład BNN

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

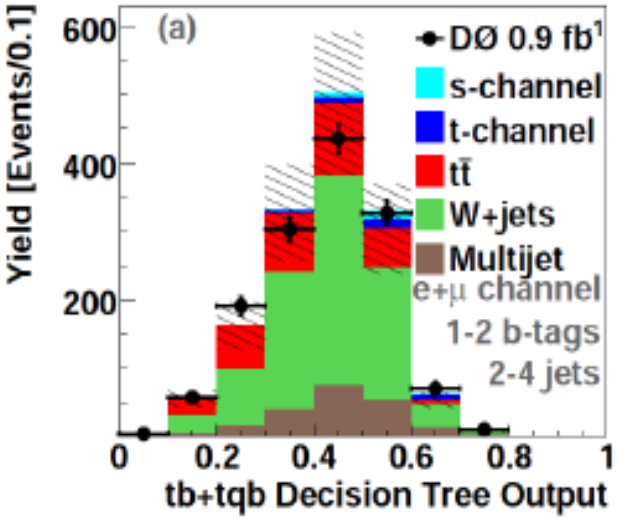
- Dane generowane za pomocą funkcji: $y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2)$ z gaussowskim szumem o odchyleniu standardowym 0.1
- 400 sieci neuronowych, z rozkładu odpowiedzi sieci wyznaczamy medianę oraz 10% Qnt i 90% Qnt (10% sieci dało odpowiedź o wartośc poniżej dolnej niebieskiej linii oraz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

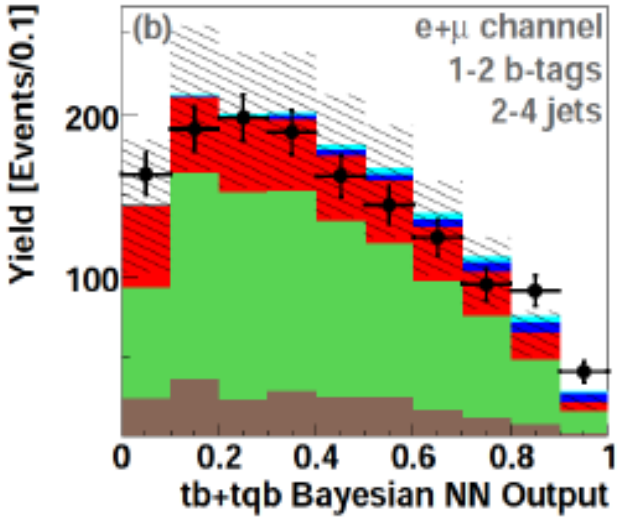






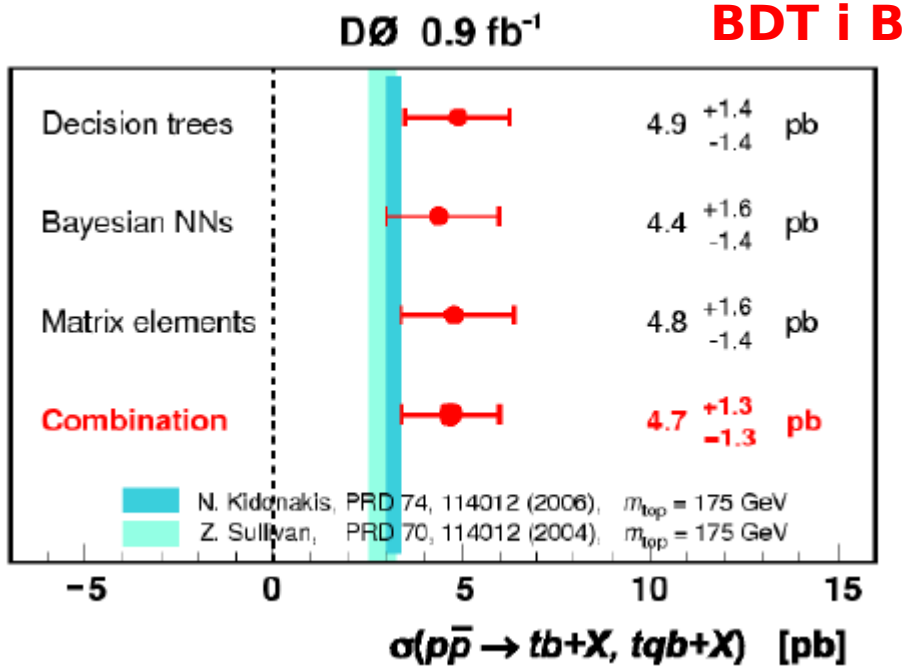
Przykład – poszukiwanie pojedynczego kwarku top w eksperymencie D0





Zbliżona dokładność BDT i BNN

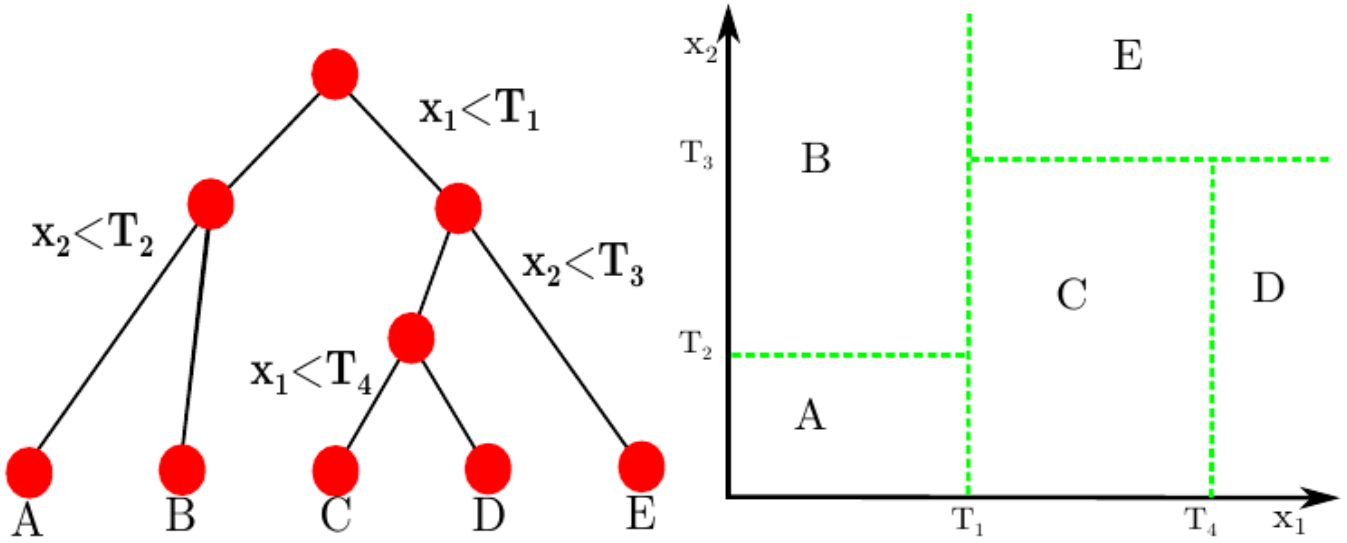
Analiza z użyciem:
wzmocnionych drzew decyzyjnych (BDT) oraz bayesowskich sieci neuronowych



D0 Collaboration,
PRD 78 012005, 2008

Drzewa decyzyjne

- Drzewo decyzyjne – sekwencja cięć, do każdego liścia przypisujemy klasę (np. ,,sygnał” lub ,,tło”)



- Bardzo proste i szybkie, ale niestabilne i czułe na małe zmiany w danych.

Rozwiązanie:

- Uczenie zespołowe (wzmacnianie) – boosting, bagging, random forests...
- ***Te procedury można stosować do różnych klasyfikatorów, nie tylko drzew!***

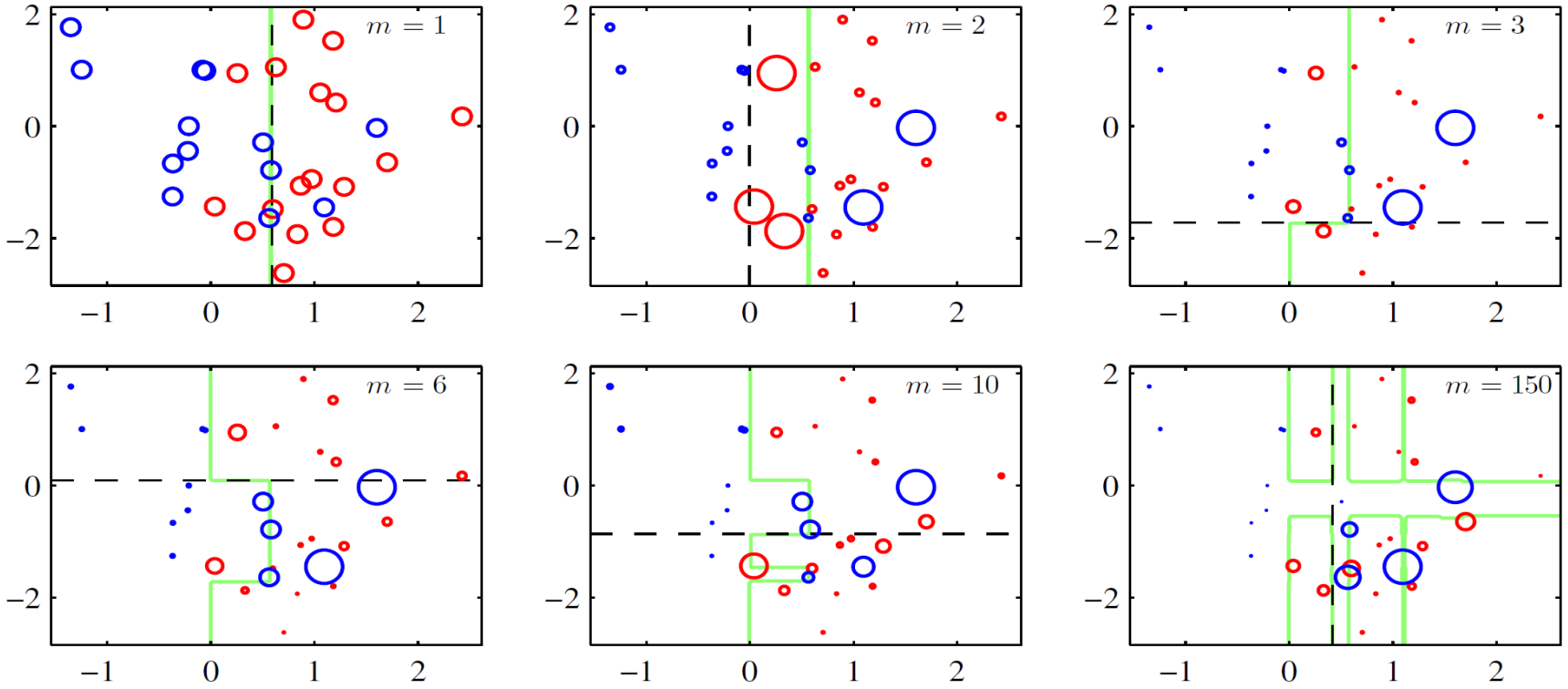
Wzmocnione drzewa decyzyjne - AdaBoost

<u>Problem</u>: Czy potrafimy wzmocnić słaby klasyfikator?

<u>Odpowiedź</u>: Tak, stosując go wiele razy.

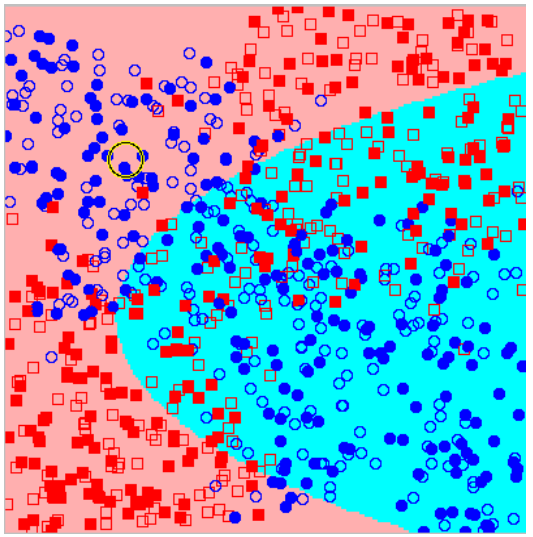
Najpopularniejszy algorytm:
AdaBoost (Freund & Schapire 1996 – nagroda Gödel'a)

- *Zbuduj drzewo decyzyjne.*
- *Zwiększ wagi źle sklasyfikowanych przypadków.*
- *Powtarzaj wiele razy tworząc wiele drzew.*
- *Klasyfikuj przypadki drogą głosowania przez wszystkie drzewa.*

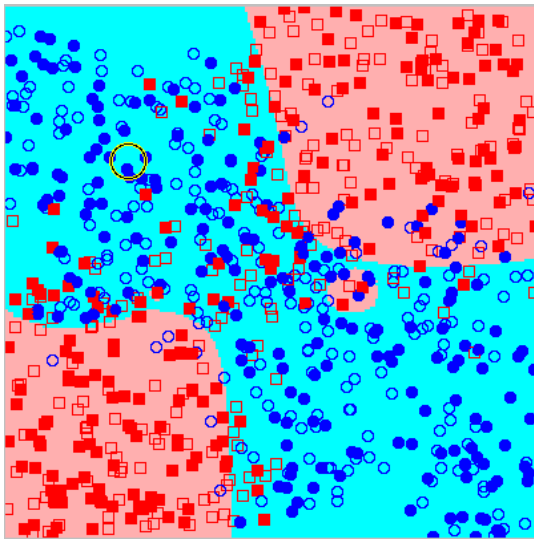


Wzmacnianie klasyfikatora

- 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!



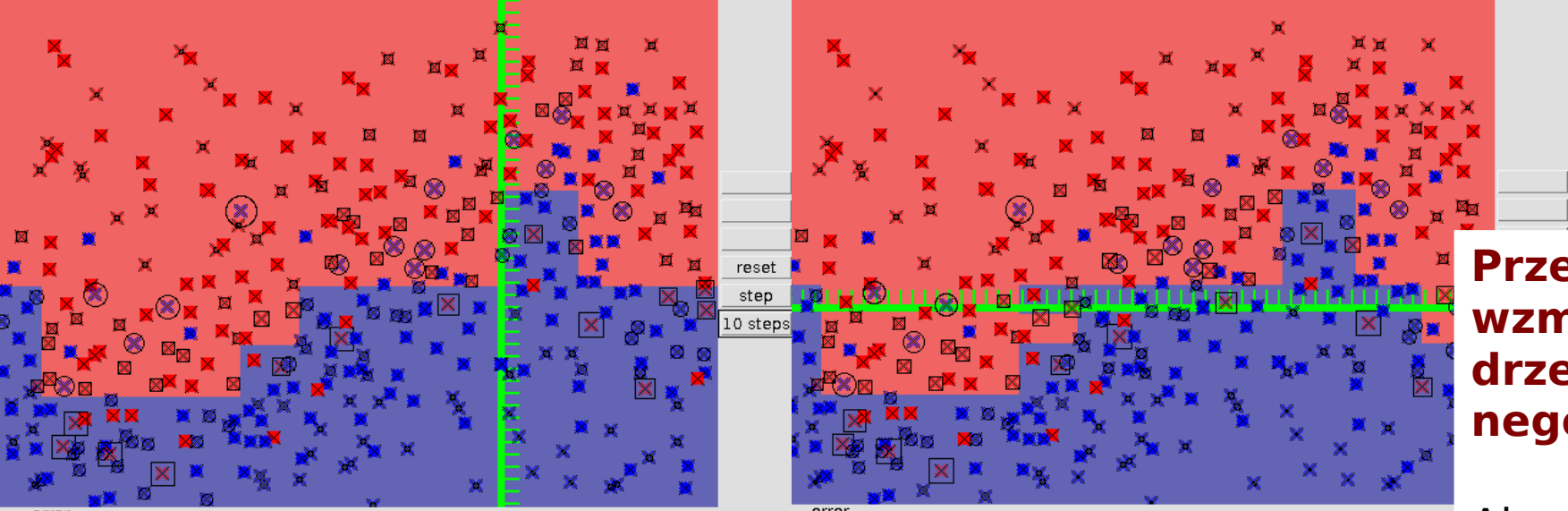
Naiwny klasyfikator bayesowski

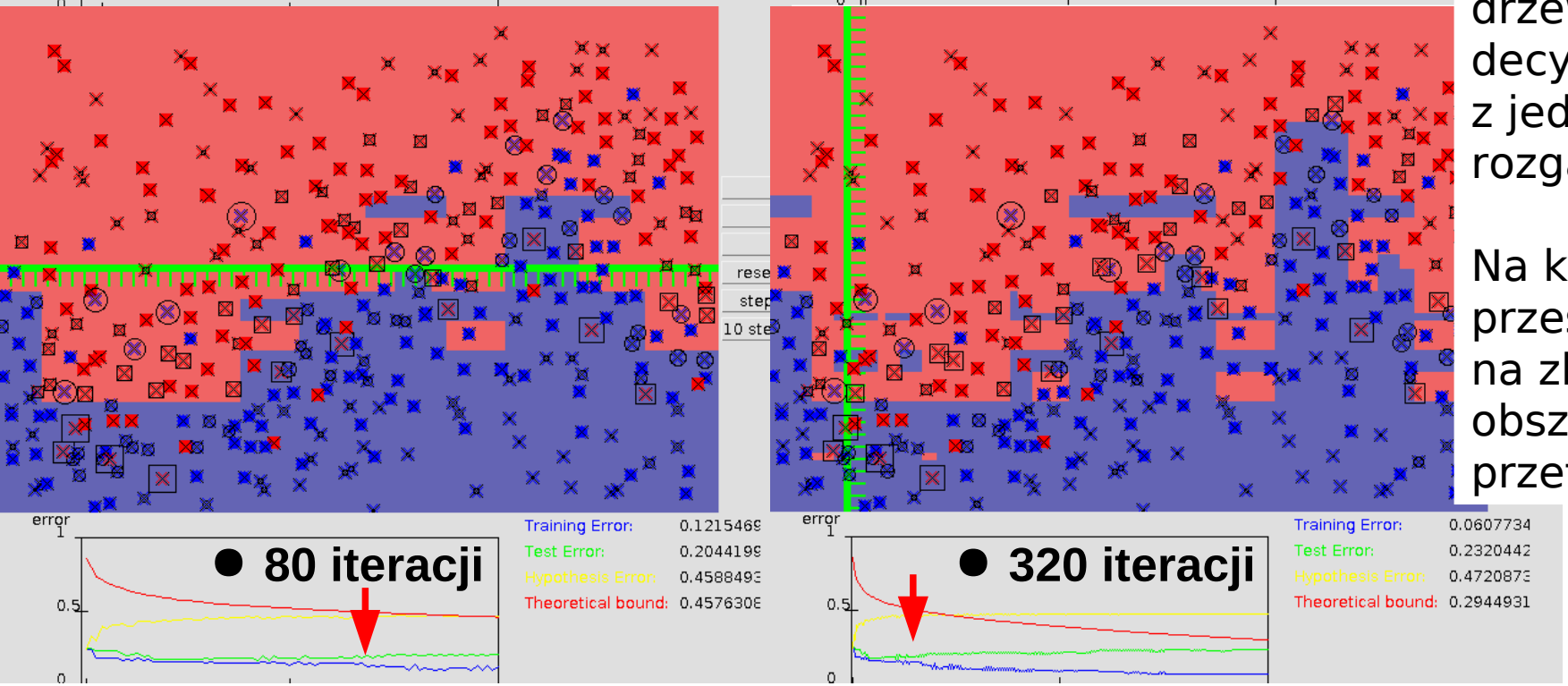


Wzmocniony klasyfikator bayesowski

Zastosowanie algorytmu wzmacniania (5 iteracji) do naiwnego klasyfikatora bayesowskiego.

Przetrenowanie wzmocnionego drzewa decyzyjnego (BDT)

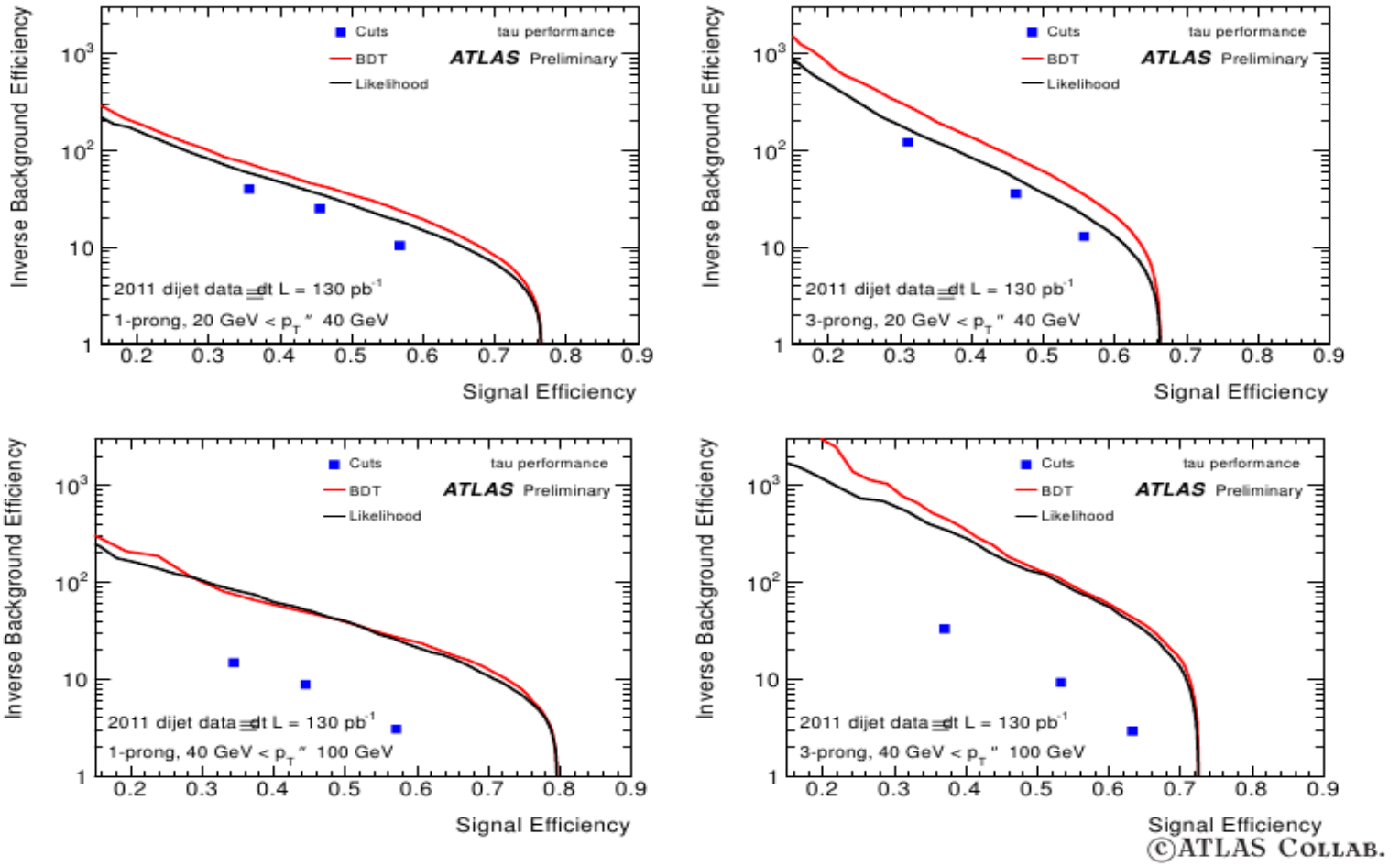




Algorytm wzmacniania zastosowany do drzew decyzyjnych z jednym rozgałęzieniem.

Na koniec dzieli przestrzeń na zbyt drobne obszary - przetrenowanie.

Identyfikacja hadronowych rozpadów leptonów tau w eksperymencie ATLAS



- Szereg zmiennych identyfikujących, żadna z nich pojedynczo nie daje dobrej identyfikacji.
- **Użycie metod wielu zmiennych zwiększa skuteczność identyfikacji.**

Uczenie bez nauczyciela

- Dotychczasowe przykłady – uczenie z nauczycielem.
- Uczenia bez nauczyciela - odpowiedź nie jest znana. Algorytm musi sam znaleźć zależności pomiędzy danymi i odpowiednio je pogrupować.

  Wytrenowany odzwierciedla statystyczną strukturę danych.

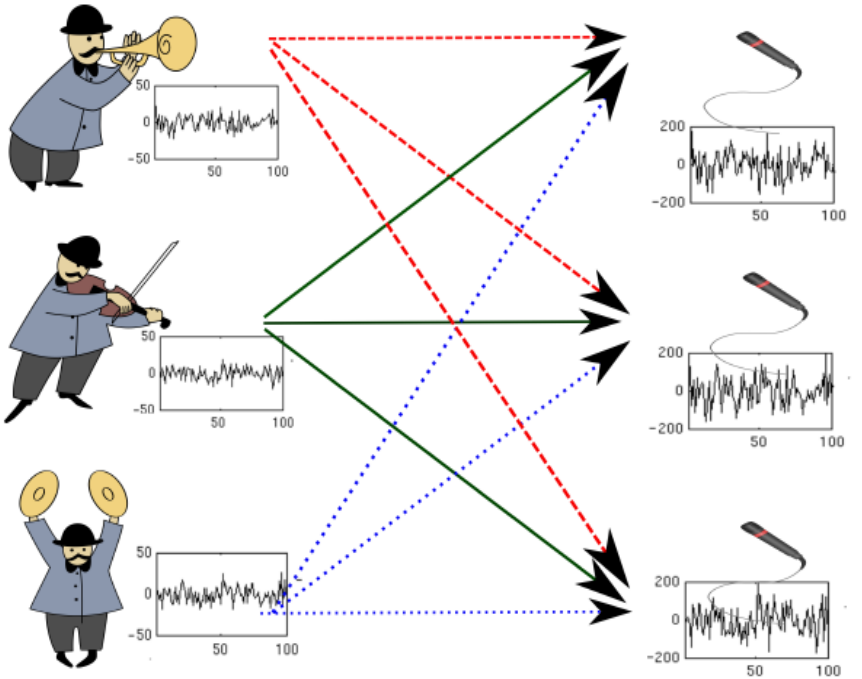
- Przykładowe algorytmy:
  - Sieć Kohonena – Self Organizing Maps
  - Analiza składowych niezależnych - Independent Component Analysis (ICA)

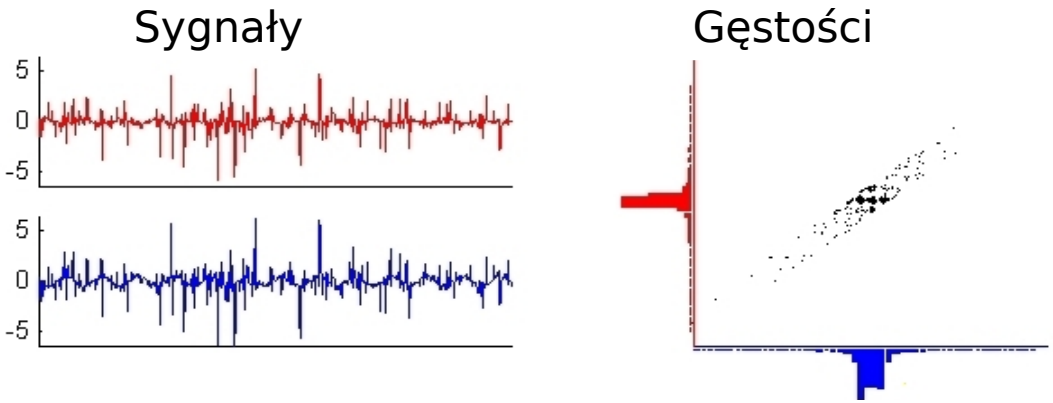
Analiza składowych niezależnych

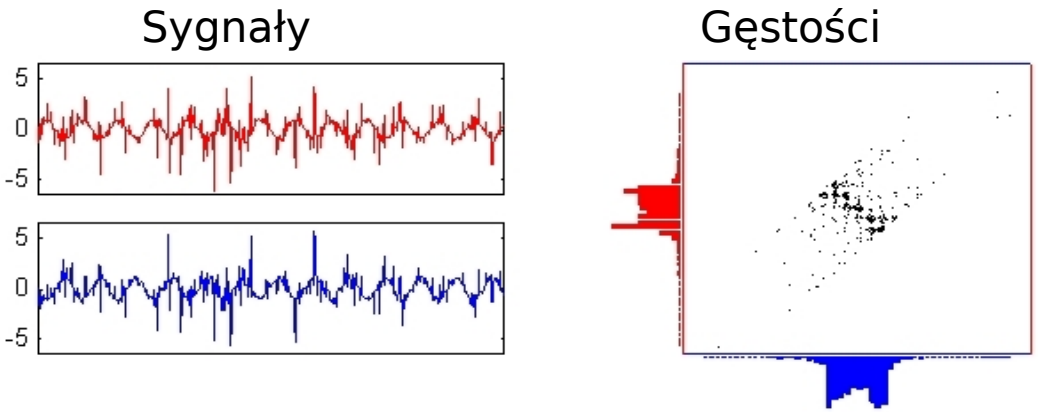
Independent Component Analysis ICA

Opracowana na Helsinki University of Technology *http://www.cis.hut.fi/projects/ica/*

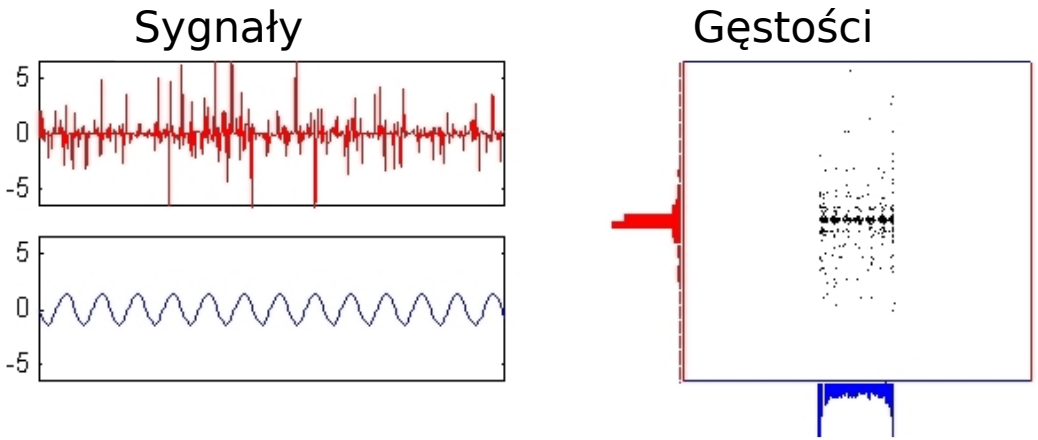
- **Problem:**
  - Załóżmy, że zarejestrowany sygnał **X** jest liniową kombinacją **X = AS** niezależnych źródeł **S**. Zarówno macierz mieszania **A** oraz **S** są nieznane.
  - **Zadanie:** znaleźć macierz **T** odwrotną do **A**, taką że elementy wektora **U = TX** są statystycznie niezależne. **T** jest macierzą odzyskującą pierwotne sygnały.
- **Zastosowania:**
  - Filtrowanie jednego źródła dźwięku z szeregu innych,
  - Separacja sygnałów w telekomunikacji,
  - Separacja sygnałów z mózgu od szumu, separacja sygnałów z różnych obszarów mózgu,
  - Separacja różnych źródeł sygnału w astrofizyce,
  - Dekompozycja sygnałów w monitoringu wiązki akceleratora w FERMILAB.







Sygnały po dekorelacji



Rozseparowane sygnały

Jak działa ICA?

- Mamy dwa zmierzone sygnały (chcemy je rozseparować na dwa niezależne źródła).

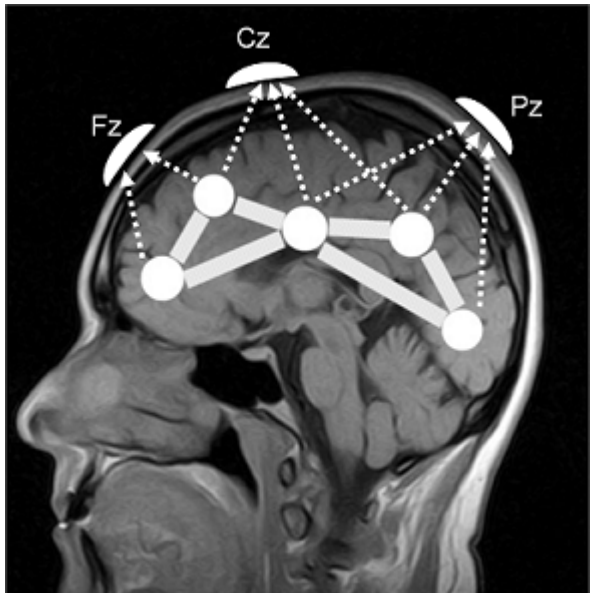
Przygotowanie danych – dekorelacja (współczynniki korelacji równe zero, σ=1).

Złożenie wielu niezależnych rozkładów daje w granicy rozkład Gaussa.

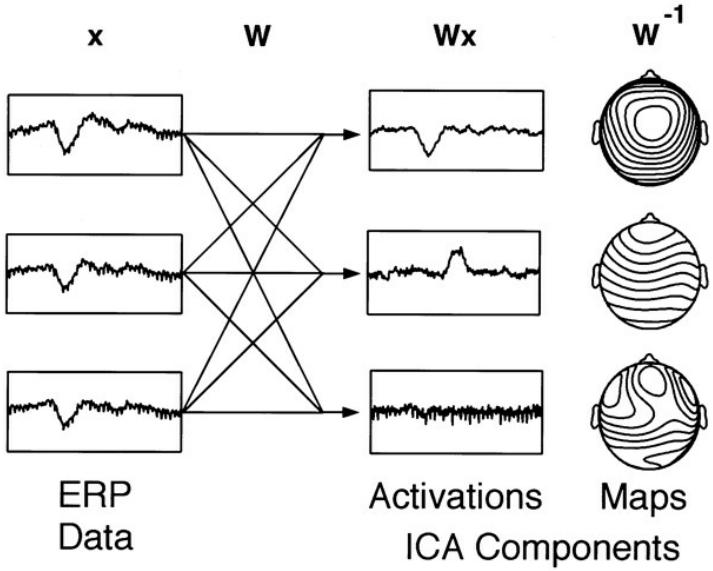
- ICA – obrót, sygnały powinny mieć rozkłady jak najbardziej nie-Gaussowskie (miara nie-Gaussowskiego kształtu - kurtoza).
- *Kurtoza:* $\mathrm{Kurt} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^4}{\sigma^4} - 3$

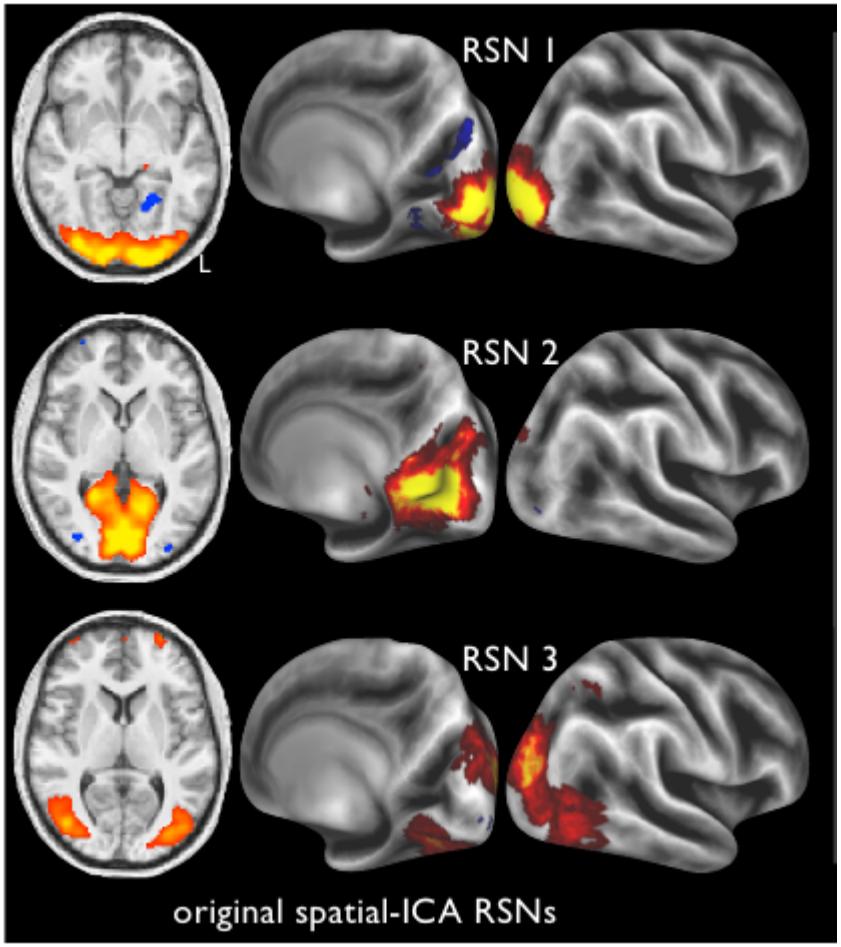
gdzie μ jest średnią rozkładu, a σ odchyleniem standardowym.

ICA – badania mózgu, separacja sygnałów



Dekompozycja za pomocą ICA

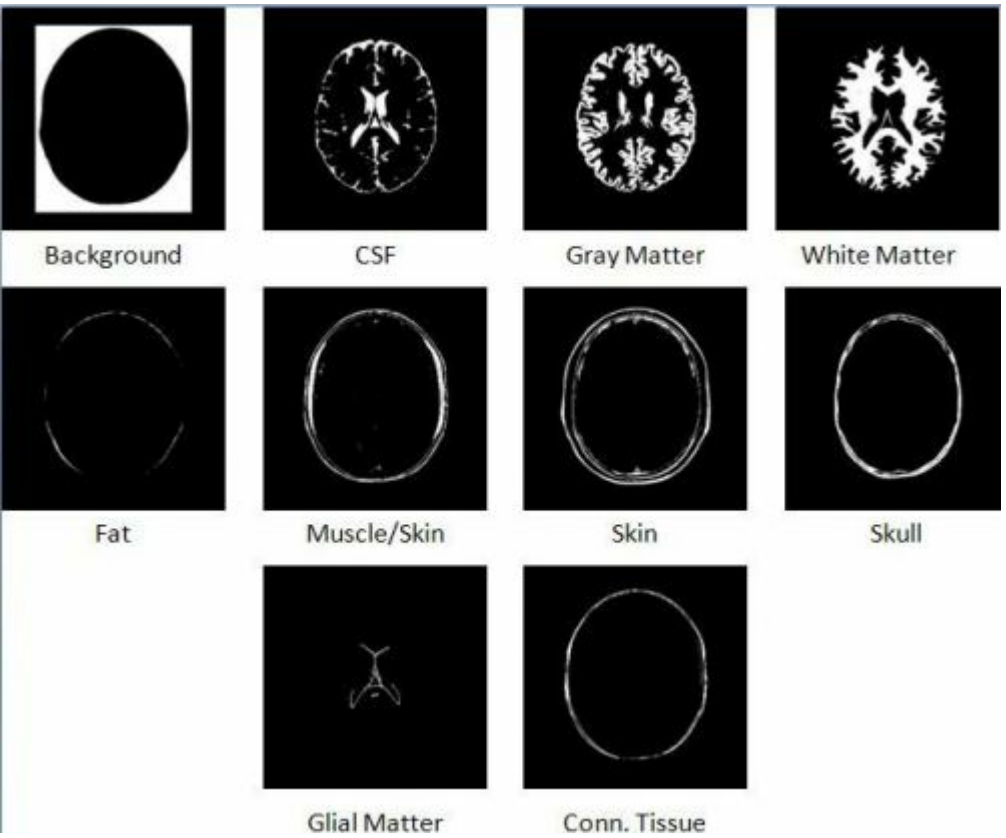




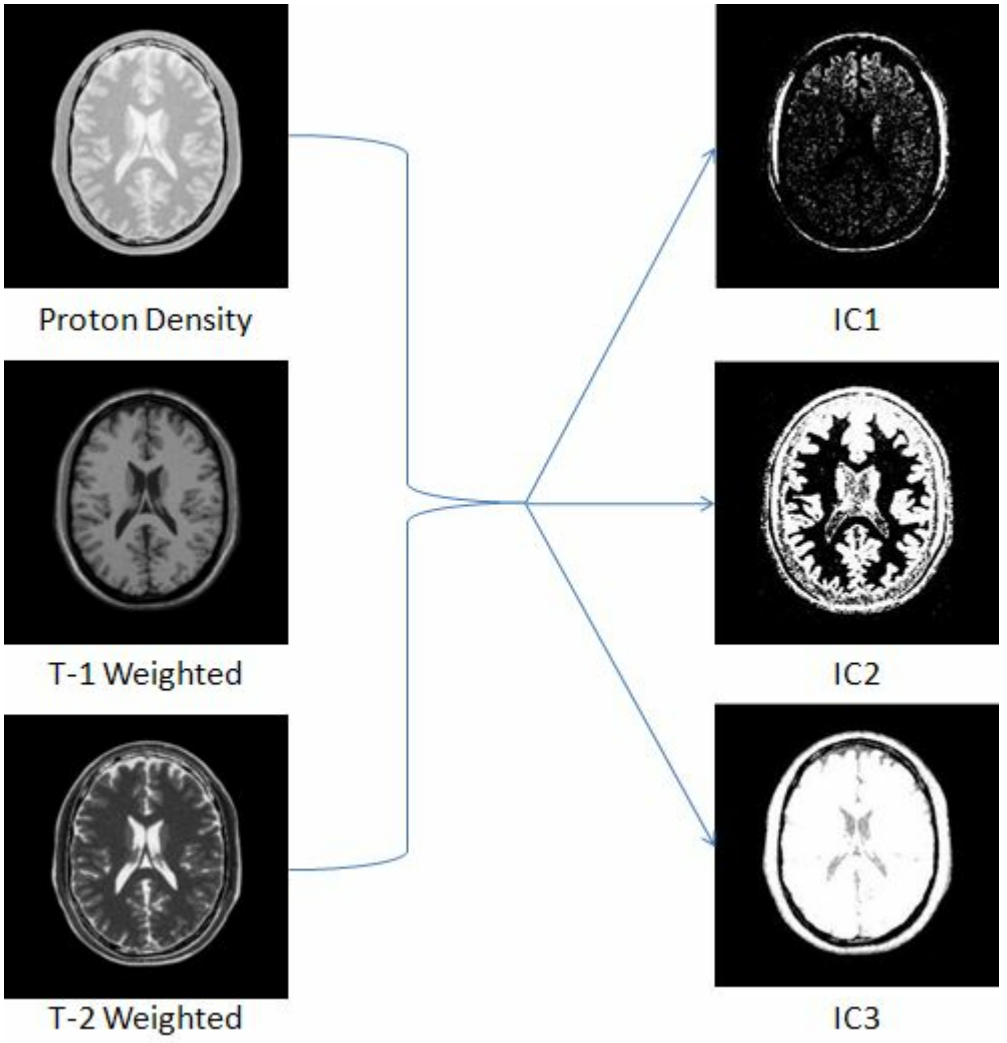
3 składowe z 21-wymiarowej dekompozycji za pomocą algorytmu ,,spatial-ICA''.

PNAS February 21, 2012 vol. 109no. 8 3131-3136

ICA a rezonans magnetyczny



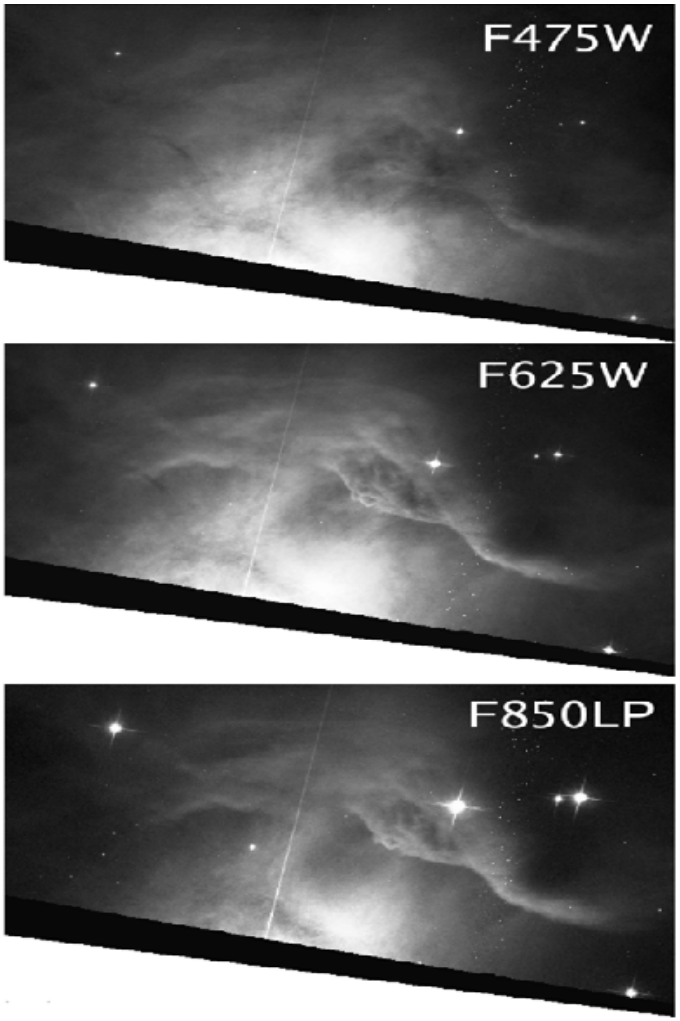
Źródła sygnału



Pomiary **Rozseparowane składowe**

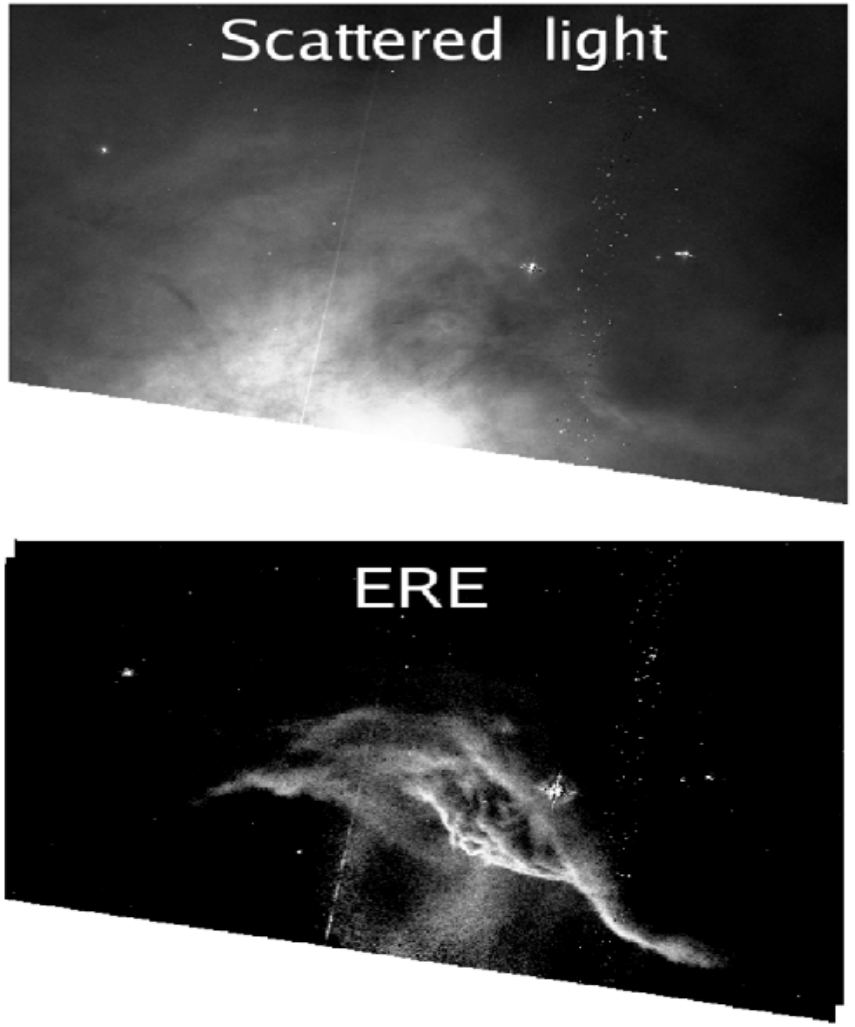
Blind Source Separation in Magnetic Resonance Images
January 30, 2010 by Shubhendu Trivedi

ICA – astronomia



Trzy obrazy mgławicy NGC 7023 uzyskane z trzema różnymi filtrami.

A&A 479, L41-L44 (2008)
DOI: 10.1051/0004-6361:20079158



Rozproszone światło (u góry) oraz obraz ERE (Extended Red Emission – u dołu) uzyskane za pomocą algorytmu FastICA z trzech obrazów źródłowych.

Rev. Thomas Bayes
(1702 - 1761)

Uwagi końcowe

Analiza wielowymiarowa jest pomocna, jeśli chcemy uzyskać z danych jak najwięcej informacji.

W każdym przypadku klasyfikacji staramy się rozwiązać ***ten sam*** problem:

Znaleźć ***dobrą*** i ***praktyczną (!)***, aproksymację b*ayesowskiej funkcji decyzyjnej (Bayes Discriminant Function – BDF)* – idealnej funkcji klasyfikującej, danej przez (nieznane) gęstości prawdopodobieństwa sygnału i tła.
Wszystkie omówione metody są różnymi algorytmami aproksymującymi tę funkcję.

Oznacza to, że jeżeli metoda daje wynik zbliżony do limitu Bayesa, to nie jest możliwe poprawienie wyniku, nawet poprzez stosowanie najbardziej wymyślnej metody klasyfikacji.

Po latach prób widać, że nie istnieje jeden, uniwersalny i najlepszy algorytm analizy wielowymiarowej. Należy być sceptycznym wobec tych, którzy twierdzą inaczej!
Należy starać się znaleźć metodę najlepiej pasującą do naszego problemu.

REKLAMA: Pakiet *T*MVA

- *T*MVA (Toolkit for MultiVariate Analysis) - zintegrowany z pakietem *ROOT.*

Andreas Höcker, Helge Voss, Kai Voss, Joerg Stelzer et. al. http://tmva.sourceforge.net

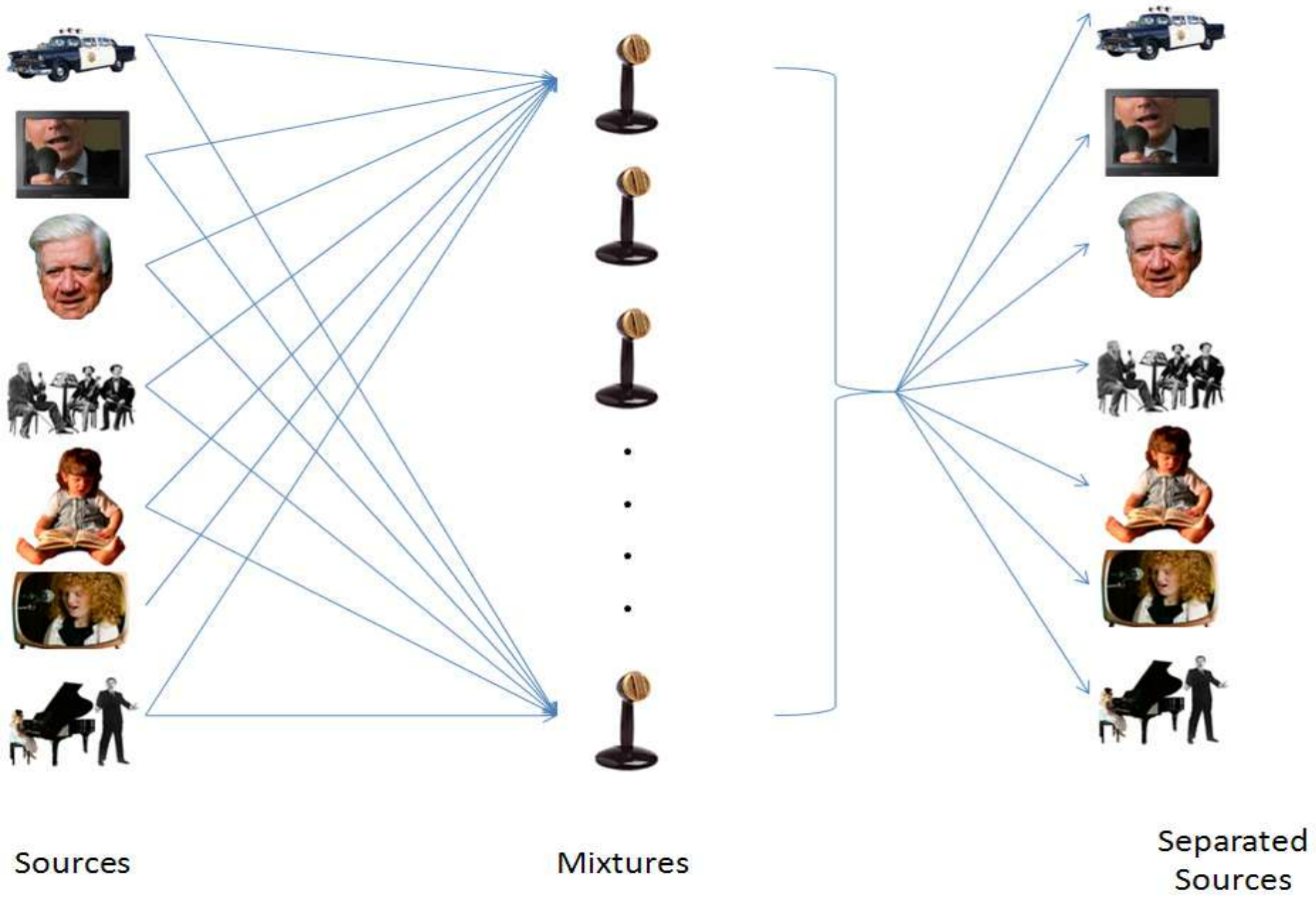
Marcin Wolter, Andrzej Zemla *(IFJ PAN Krakow)*

- Zawiera implementacje wielu metod wykorzystywanych do separacji danych.

- Optymalizacja cięć (rectangular cut optimisation)
- Metoda największej wiarygodności (Likelihood estimator)
- Wielowymiarowa metoda największej wiarygodności (PDE-RS)
- Dyskryminanty Fishera
- Sieci neuronowe
- Drzewa decyzyjne (Boosted/Bagged Decision Trees)

Deser

- Cocktail Party Demo - applet pokazujący działanie algorytmu analizy składowych niezależnych na przykładzie problemu ,,przyjęcia cocktailowego".
  http://research.ics.aalto.fi/ica/cocktail/cocktail_en.cgi



Ciekawe applety demonstracyjne

- Cocktail Party Demo
- Support Vector Machine
- LocBoost Applet
- AdaBoost applet
- Neural Network applets