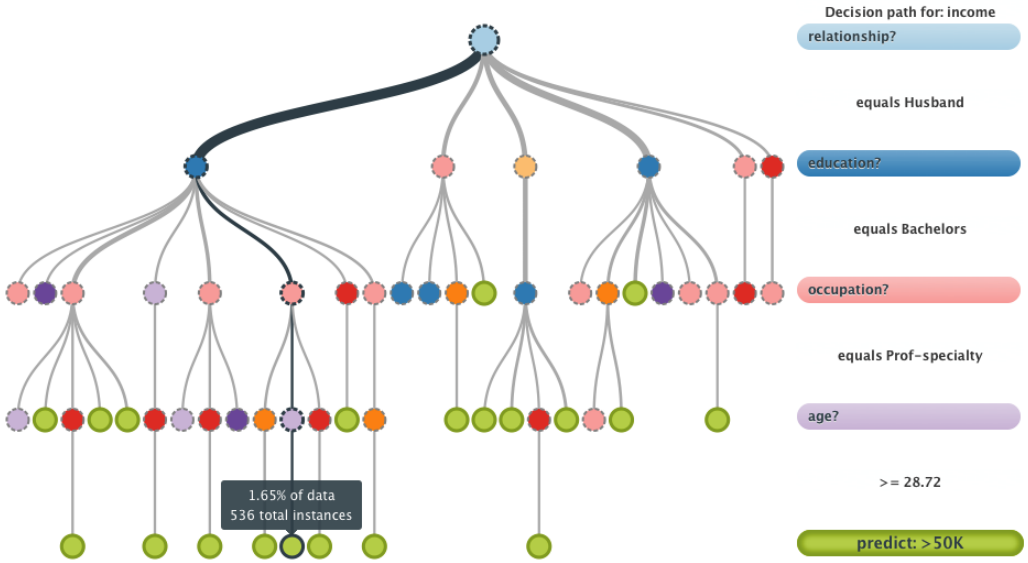
Machine learning

Lecture 2



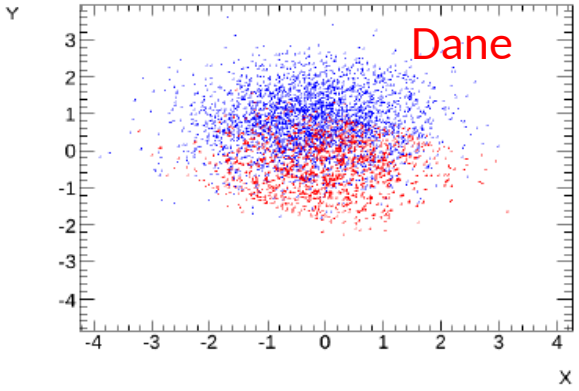
Marcin Wolter

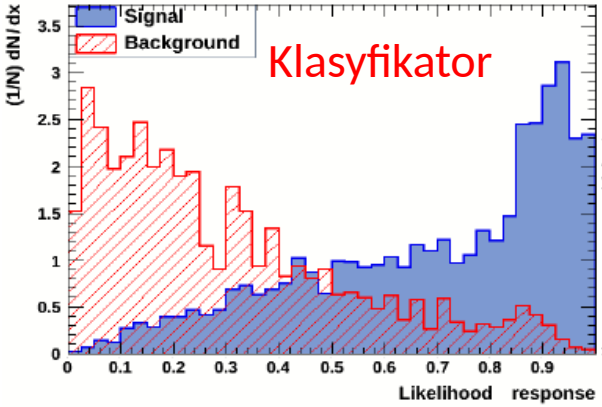
IFJ PAN

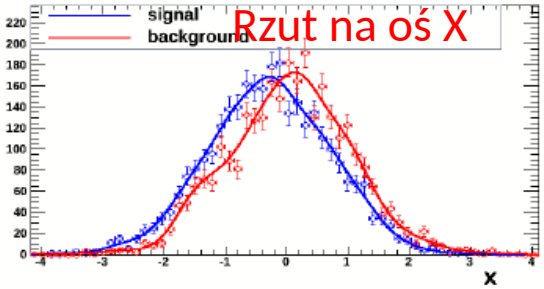
9 March 2017

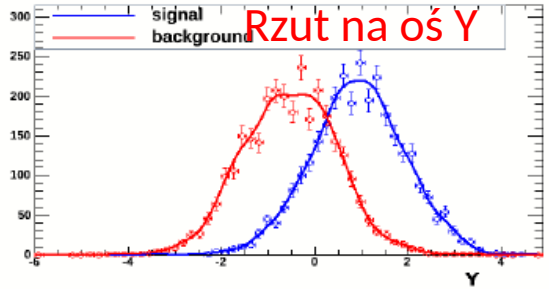
- Proste metody nieliniowe jak naiwny klasyfikator bayesowski, metoda k-najbliższych sąsiadów, metody jądrowe Parzena.
- Wzmocnone drzewa decyzyjne – Boosted Decision Trees (BDT)

Naiwny klasyfikator bayesowski









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

- Oparty na założeniu o wzajemnej niezależności zmiennych (dlatego ,,naiwny"):

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

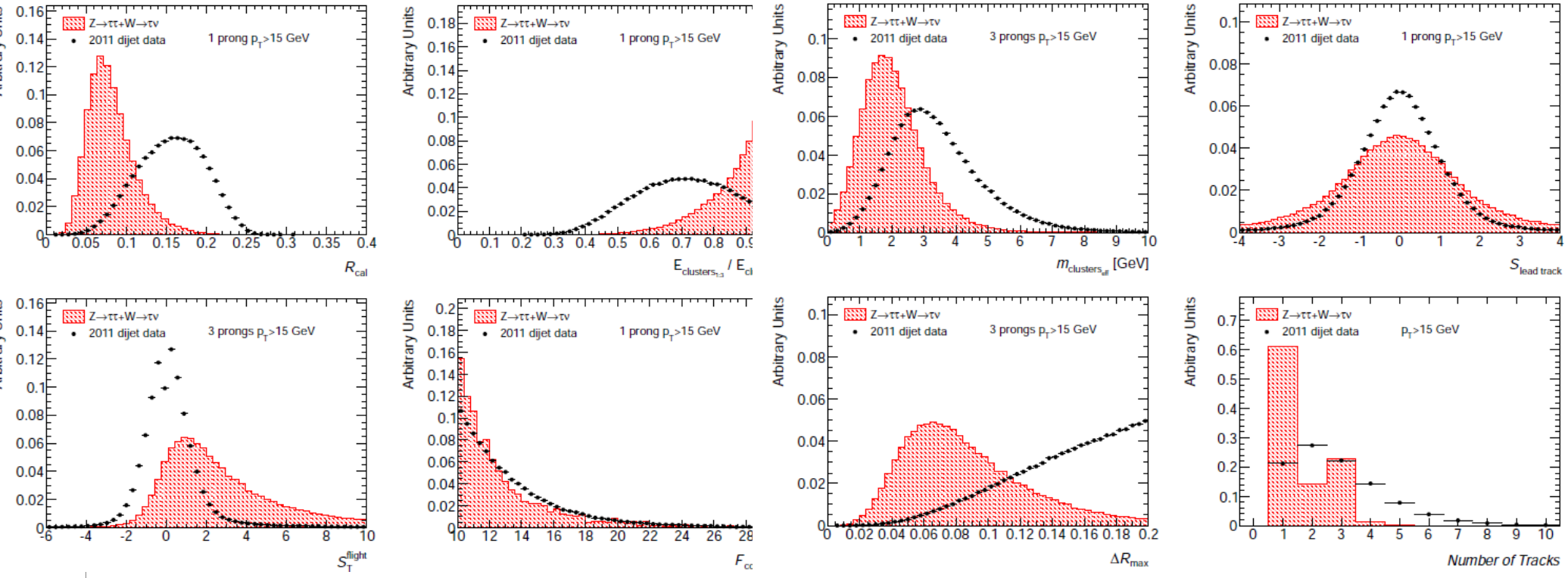
$$\mathcal{L}_i(S) = \prod_{k=1}^{N} p_k(S)(x_i)$$

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

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

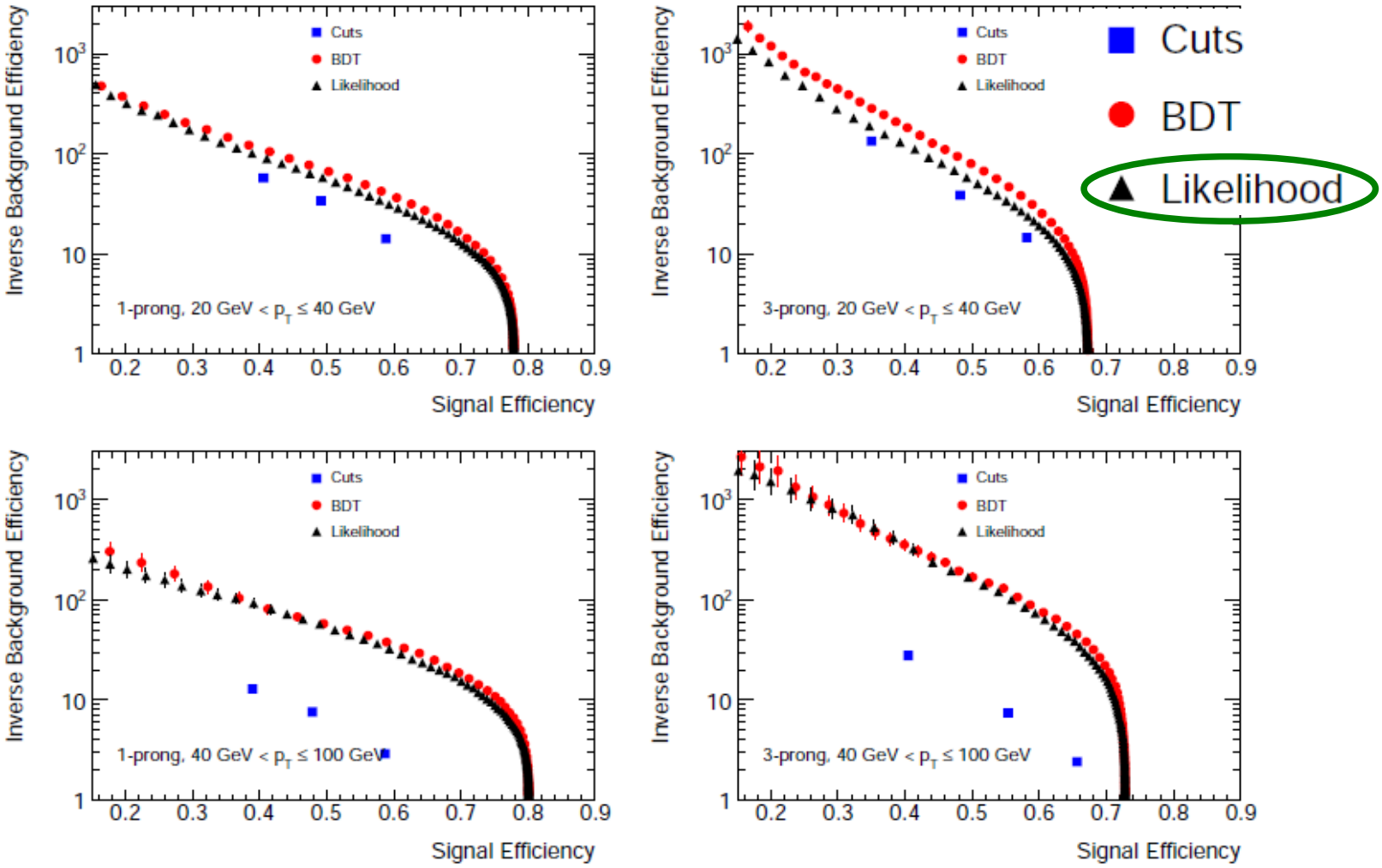
Identyfikacja tau w eksperymencie ATLAS

- Szereg zmiennych identyfikujących, żadna z nich nie daje „dobrej” identyfikacji => metody wielu zmiennych, informacja ze wszystkich



Przykładowe zmienne identyfikujące.

Identyfikacja tau – zastosowane metody

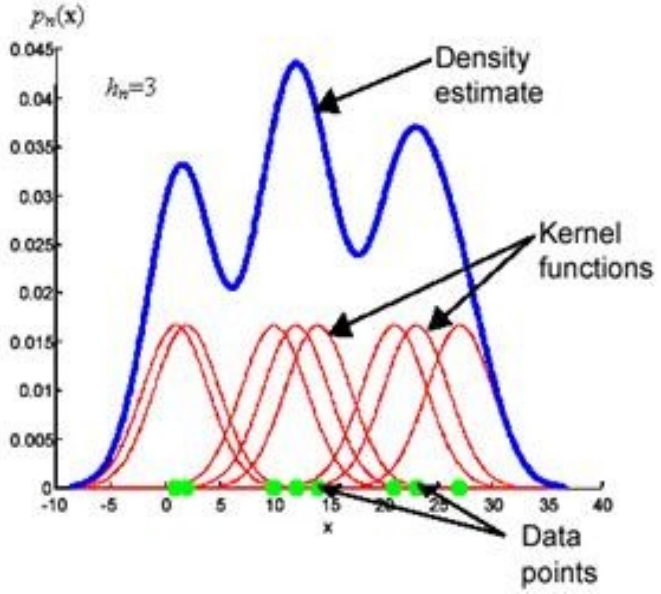


- **Cięcia – powszechnie znane.**
- **Likelihood (Naive Bayes) – właśnie omówione.**
- **Boosted Decision Trees (BDT) –omówimy niebawem**

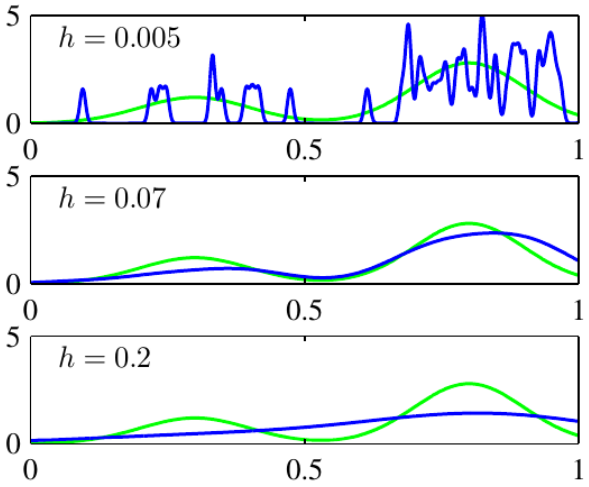
Jądrowe estymatory gęstości

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.



Aproksymowana gęstość prawdopodobieństwa (niebieska) porównana z prawdziwą gęstością prawdopodobieństwa (zielona) w zależności od szerokości funkcji Gaussa użytej jako jądro. Widzimy, że parametr określający jej szerokość, pełni rolę parametru wygładzającego.

Pakiet QUAERO z eksperymentu D0

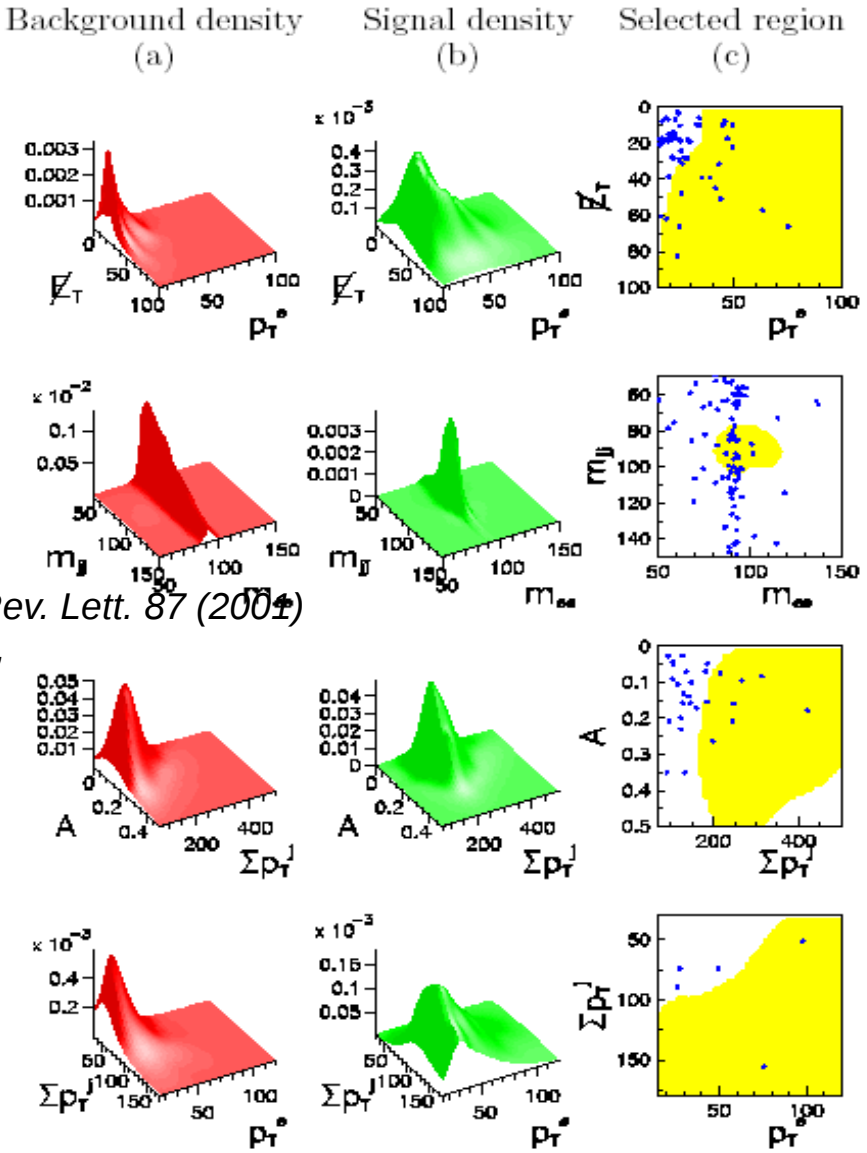


FIG. 1. The background density (a), signal density (b), and selected region (shaded) (c) determined by QUAERO for the standard model processes discussed in the text. From top to bottom the signals are: $WW \to e\mu\not{E}_T$, $ZZ \to ee\,2j$, $t\bar{t} \to e\not{E}_T\,4j$, and $t\bar{t} \to e\mu\not{E}_T\,2j$. The dots in the plots in the rightmost column represent events observed in the data.

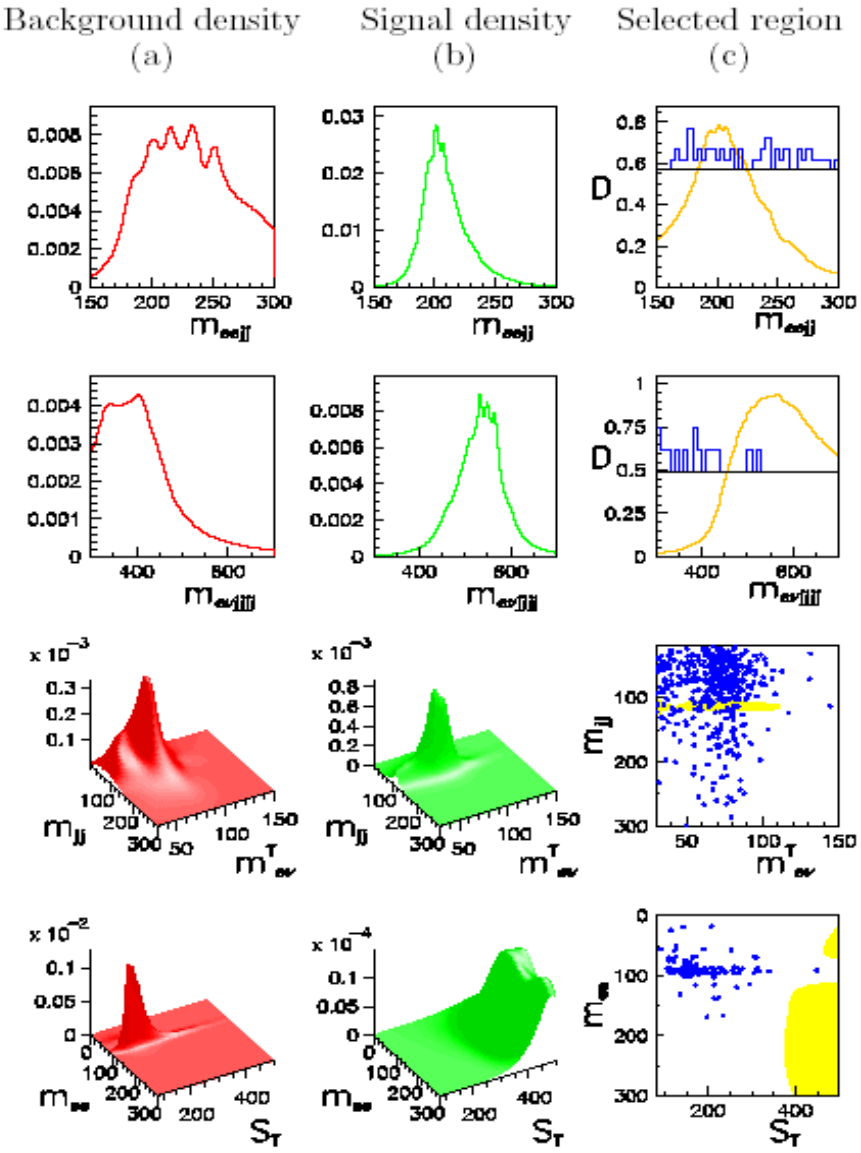


FIG. 2. QUAERO's analysis of signatures involving undiscovered particles. From top to bottom the hypothetical signals are: $h_{200} \to ZZ \to ee\,2j$, $Z'_{550} \to t\bar{t} \to e\not{E}_T\,4j$, $Wh_{115} \to e\not{E}_T\,2j$, and $LQ_{225}\overline{LQ}_{225} \to ee\,2j$. Plots (c) of the first two rows show the discriminant D (curve), the threshold $D_{\rm cut}$ (horizontal line), and the data (histogram); the region with $D > D_{\rm cut}$ is selected.

Phys. Rev. Lett. 87 (2001) 231801,

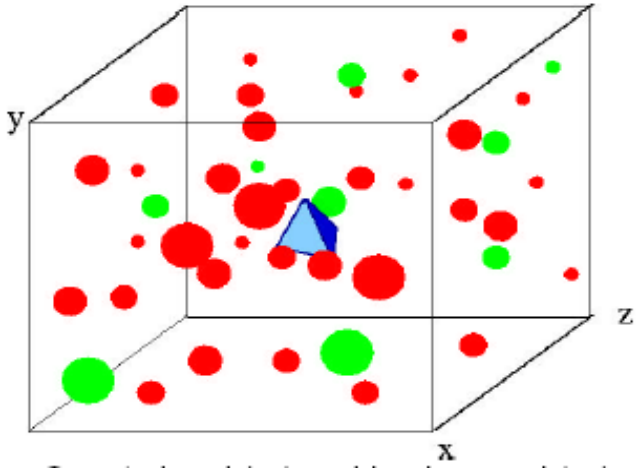
Metoda PDE_RS – rozwinięcie metod Parzena

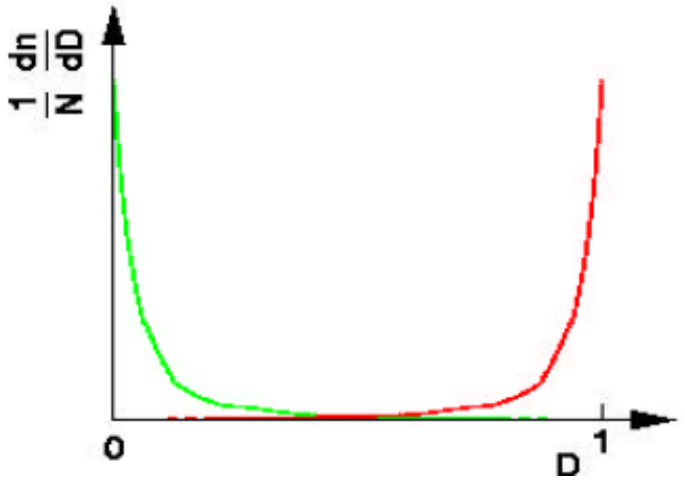
*metoda opublikowana przez T. Carli, B. Koblitz, NIM A 501 (2003) 576-588

Zlicza przypadki sygnału (n_s) i tła (n_b) w N-wymiarowej kostce dookoła klasyfikowanego przypadku – potrzeba tylko kilku przypadków ze zbioru treningowego.

Rozmiary kostki – swobodne parametry metody.

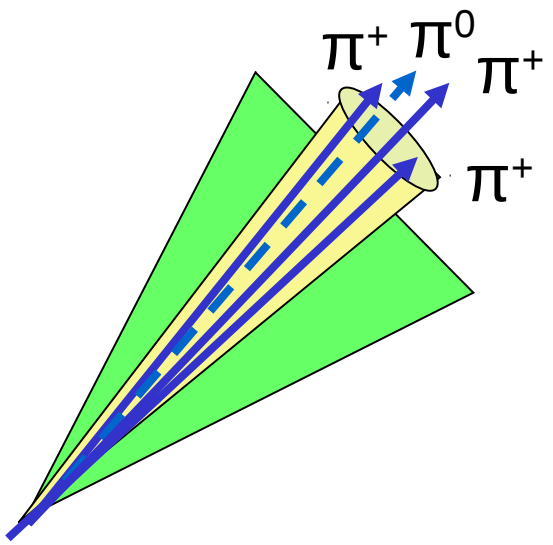
Dyskryminator *D(x)* : $D(x)=\frac{n_S}{n_S+n_b}$





- Prosta analiza.
- Przypadki zapisane w drzewie binarnym – szybciej znajdywane są sąsiednie przypadki.
- Szczególny przypadek metody Parzena – funkcja jądrowa: $f(x)=\begin{cases}1 & (|x|\le d)\\ 0 & (|x|>d)\end{cases}$

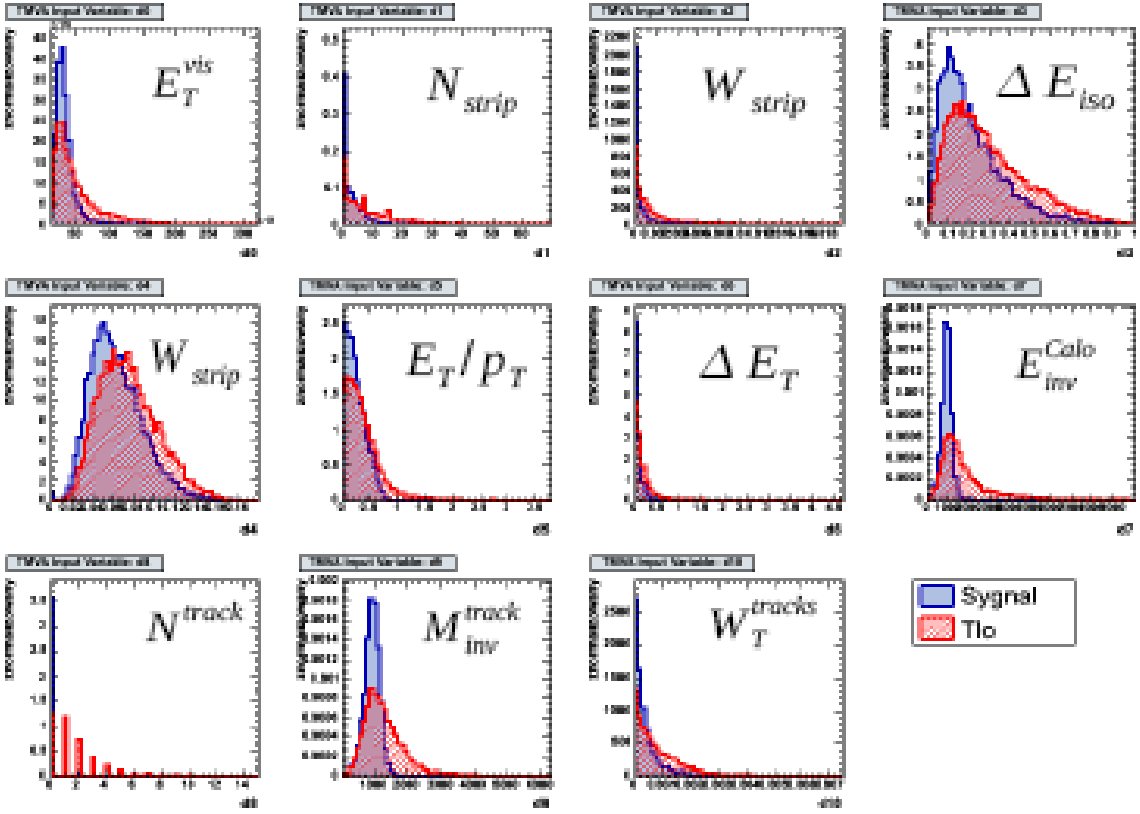
Identyfikacja leptonów τ w eksperymencie ATLAS



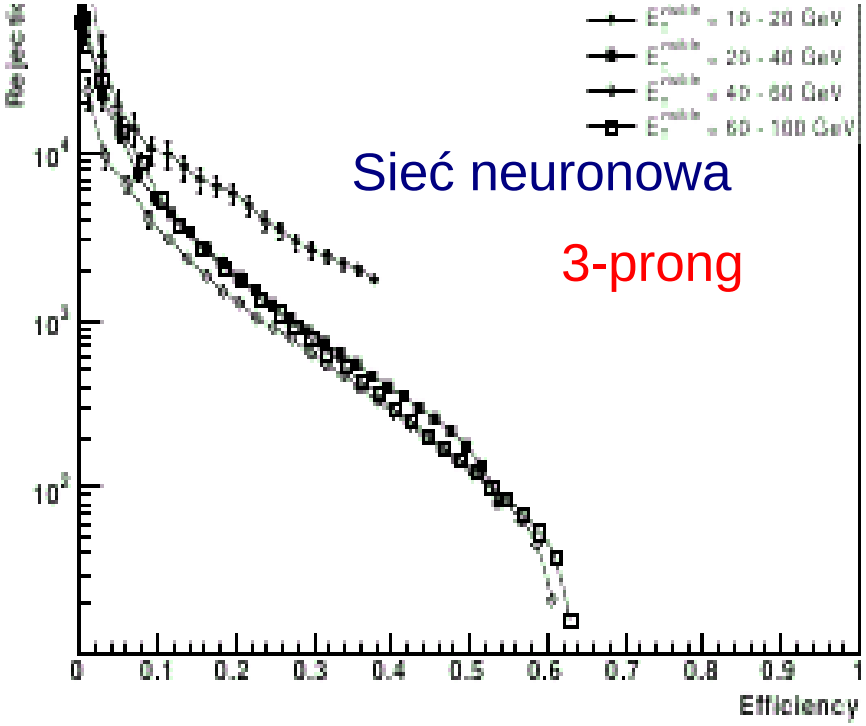
Dane do analizy: hadronowe rozpady τ zrekonstruowane za pomocą algorytmu tau1p3p (z danych Monte-Carlo).

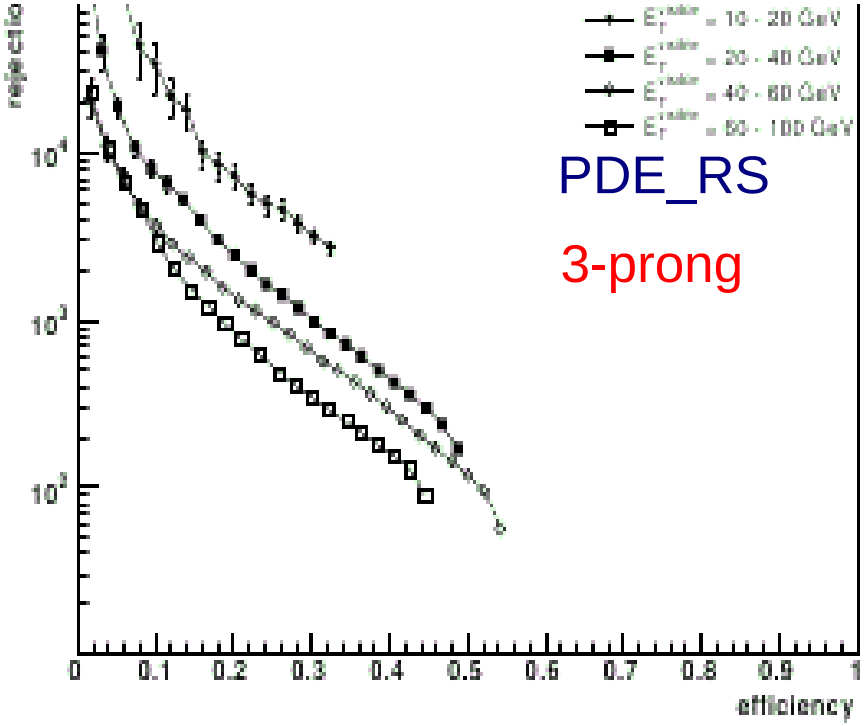
Żadna ze zmiennych samodzielnie nie zapewnia dobrej separacji sygnału i tła – konieczność użycia metody wielu zmiennych.

Przypadki 3-prong



Identyfikacja z użyciem Sieci Neuronowej oraz PDE_RS





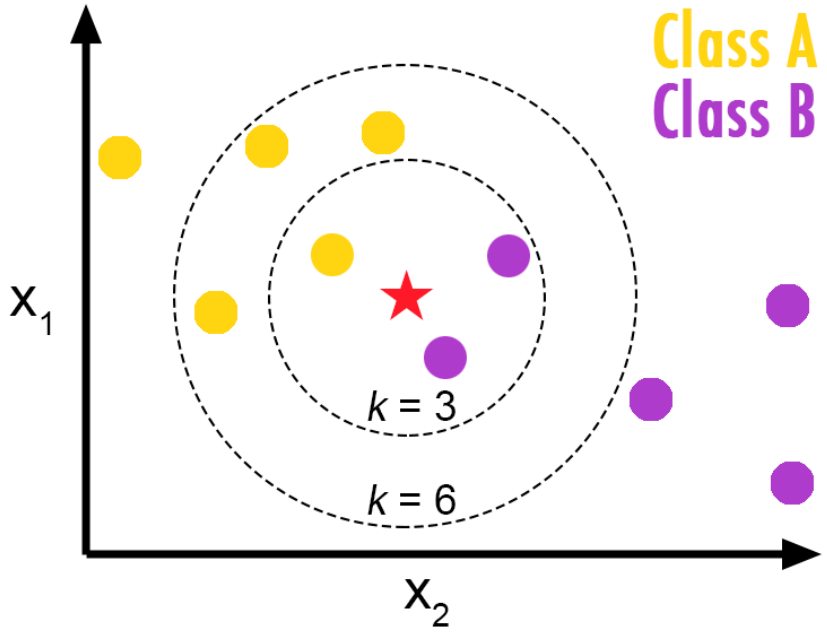
Selection	Efficiency	Rejection cuts	Rejection NN	Rejection PDRS
Standard cuts				
one-prong	0.33	225 ± 10	510 ± 40	460 ± 40
three-prong	0.28	225 ± 10	510 ± 40	460 ± 40

Znaczna poprawa odrzucania tła względem analizy z użyciem cięć.

Metoda kNN
k najbliższych sąsiadów

- zaproponowana już w 1951 roku
- zaliczamy obiekt do tej klasy, do której należy większość z jego k najbliższych sąsiadów,
- lub też określamy prawdopodobieństwo zaliczenia do danej klasy jako:

$$p(S|x) = \frac{k_S}{k}$$



Boosting, bagging, BDT...
czyli o łączeniu klasyfikatorów

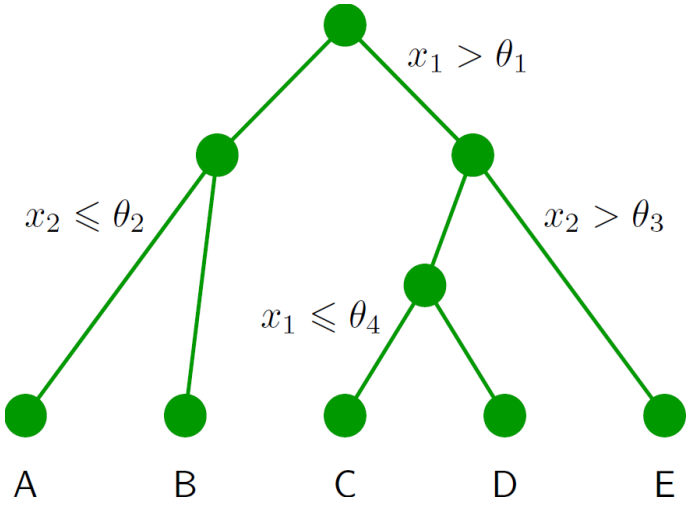
Co oznacza skrót BDT ???

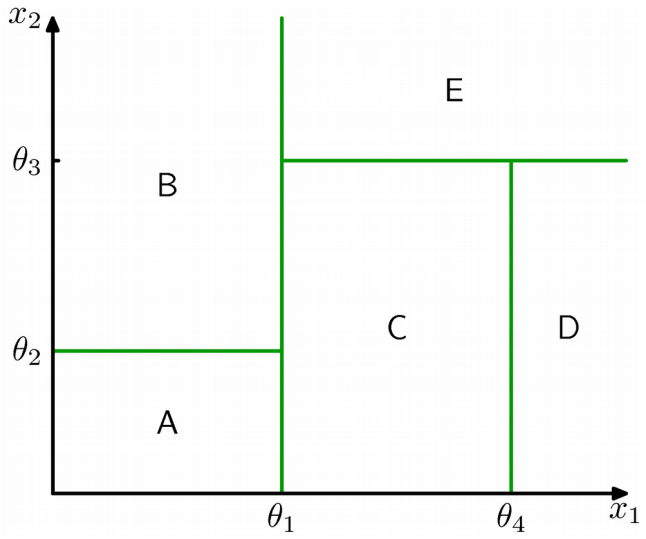
- **BDT – Boosted Decision Tree:**
  - **Decision Tree** – algorytm znany od dawna, powszechnie stosowany we wszelakich systemach eksperckich. Jako drzewo decyzyjne formułuje się np. schematy działania przy udzielaniu pierwszej pomocy: jeśli coś to zrób to i to, sprawdź dalej inny warunek itd.
  - **Boosted**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

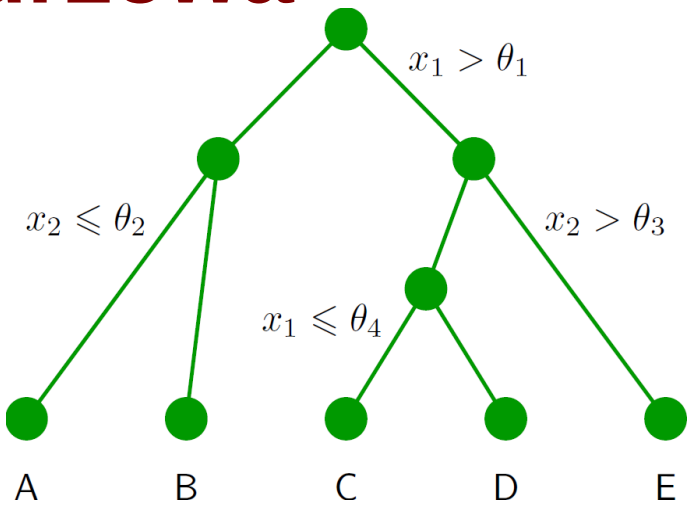




- Proste w interpretacji i wizualizacji
- Odporne na przypadki odstające od innych (*outliers*).
- Słabe zmienne są ignorowane.
- Bardzo szybki trening oraz klasyfikacja.
- Niestety: **czułe na fluktuacje, niestabilne.**

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)

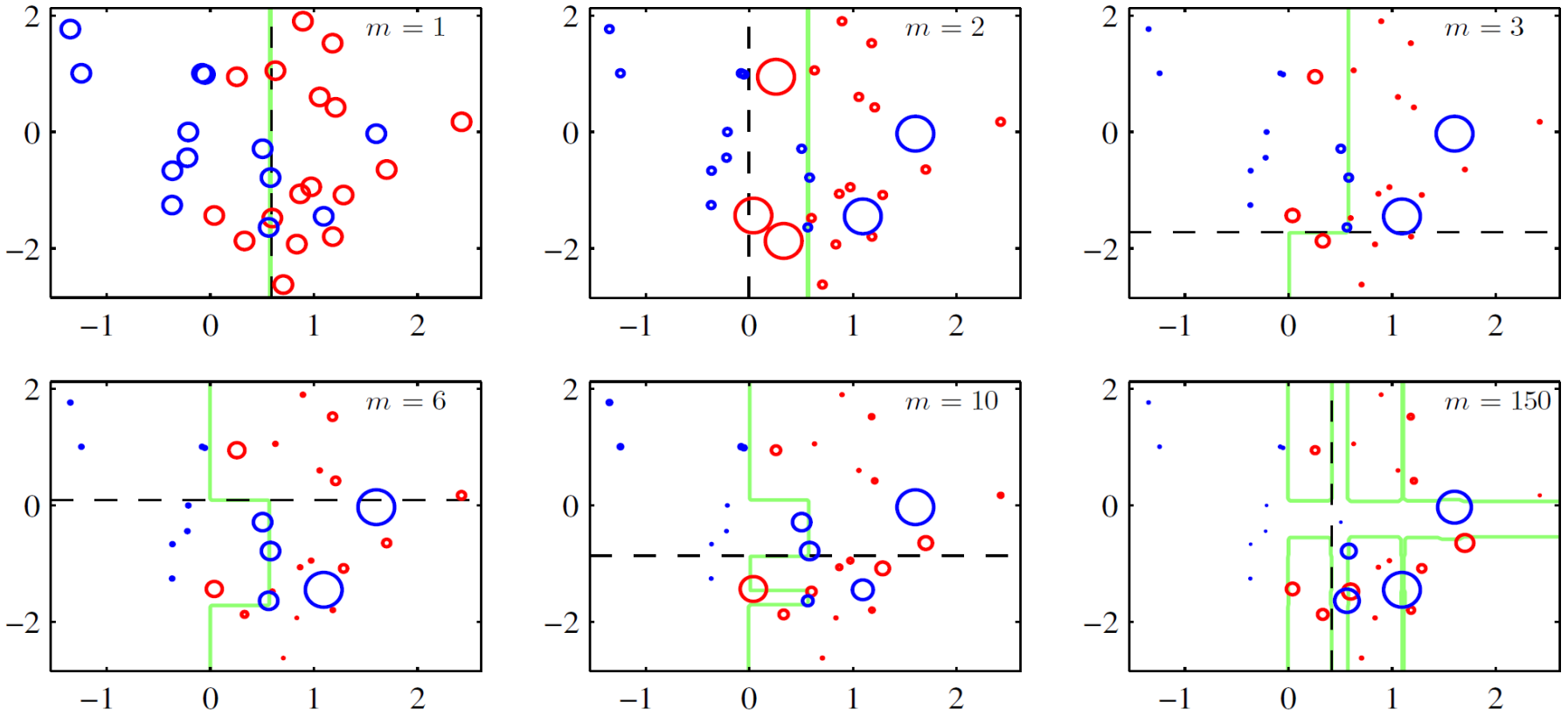
AdaBoost - łączenie klasyfikatorów

Problem: czy mając słaby klasyfikator można go poprawić?

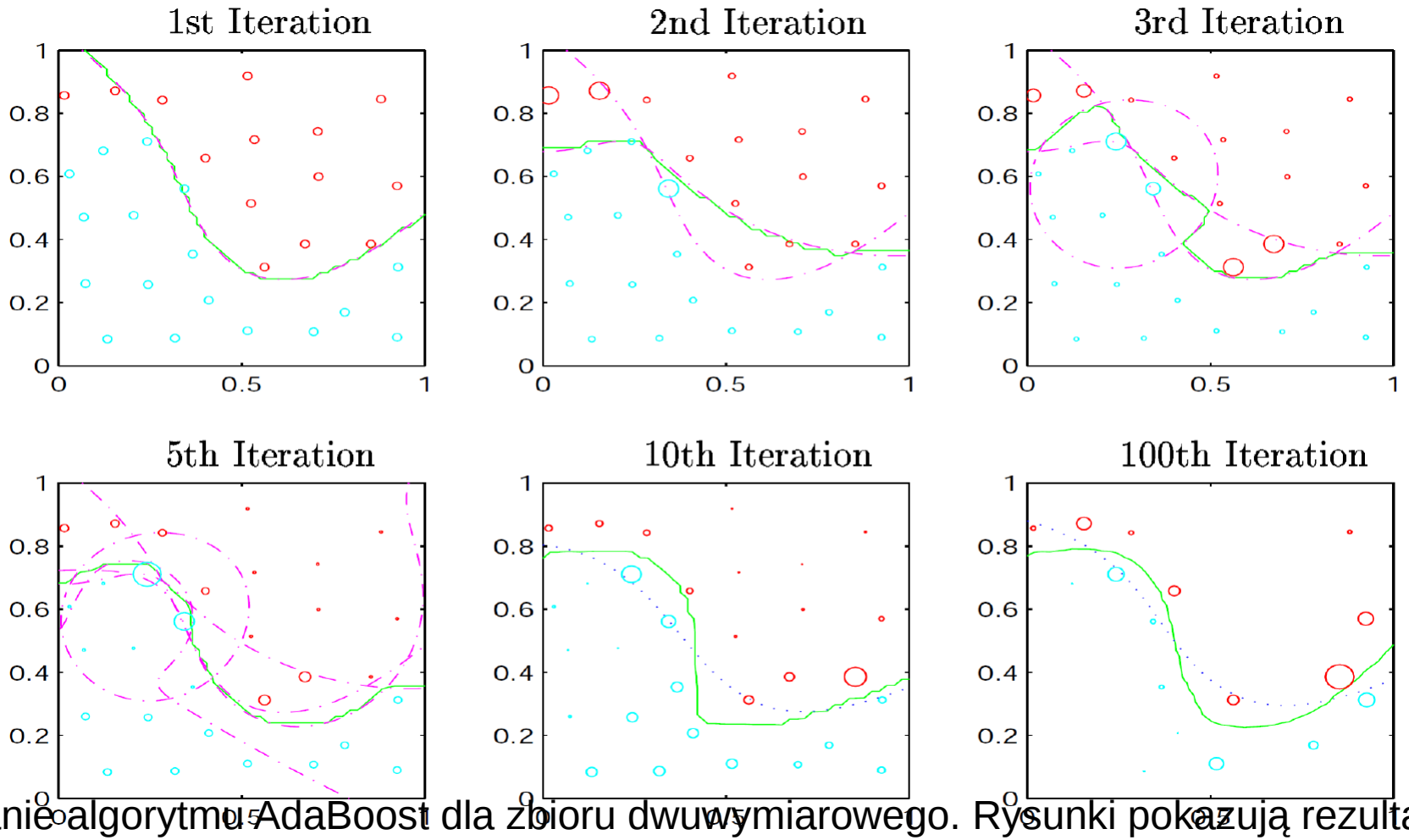
Odpowiedź: tak, stosując go wiele razy.

- 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).

AdaBoost

- Pięć lat po publikacji AdaBoost Friedman pokazał, że algorytm minimalizuje wykładniczą funkcję straty:

$$E = \sum_{n=1}^{N} \exp\left(-t_n f_m(x_n)\right)$$

gdzie *f(x)* jest odpowiedzią algorytmu

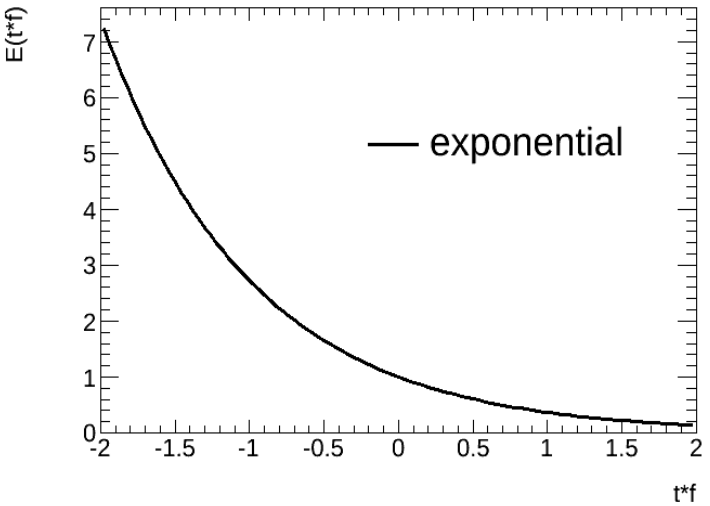
t = 1 sygnał, *t = -1* tło

$t_n \cdot f_m(x_n) > 0$ – poprawnie sklasyfikowany

$t_n \cdot f_m(x_n) < 0$ – niepoprawnie sklasyfikowany

- Funkcja wykładnicza szybko rośnie => duża kara za źle sklasyfikowane punkty, algorytm czuły na pojedyncze odstające od reszty dane. Klasyfikacja staje się gorsza, jeśli dane są słabo separowalne.

  - Może użyć innej funkcji?
  - Friedman w 2000 roku zaproponował kilka innych funkcji straty.



AdaBoost in action

Algorytm AdaBoost

1. Przypisz wszystkim wektorom ze zbioru treningowego wagę $w_i=1/N$.

2. Dla m = 1,...,M:

a)Wytrenuj klasyfikator $y_m(x)$ na zbiorze treningowym minimalizując:

$$J_m = \sum_{n=1}^{N} w_n^m I\left(y_m(x_n) \neq t_n\right)$$

b)Oblicz wielkości:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^m I\left(y_m(x_n) \neq t_n\right)}{\sum_{n=1}^{N} w_n^m}$$

$$\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$$

c)Zmień wartości wag poszczególnych wektorów w zbiorze treningowym:

$$w_n^{m+1} = \begin{cases} \frac{w_n^m}{Z_m} e^{\alpha_m} & \text{dla} \quad y_m(x_n) \neq t_n \\ \frac{w_n^m}{Z_m} e^{-\alpha_m} & \text{dla} \quad y_m(x_n) = t_n \end{cases}$$

3.Wynik głosowania dany jest jako:

$$Y_M(\mathbf{x}) = sign\left[\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right]$$

Wszystkie klasyfikatory trenujemy na tym samym zbiorze danych, ale z różnym zestawem wag. Wagi te zależą od wyników poprzedniego treningu, a więc trudno jest trenować równolegle wiele klasyfikatorów.

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \tag{14.17}$$

   (c) Update the data weighting coefficients

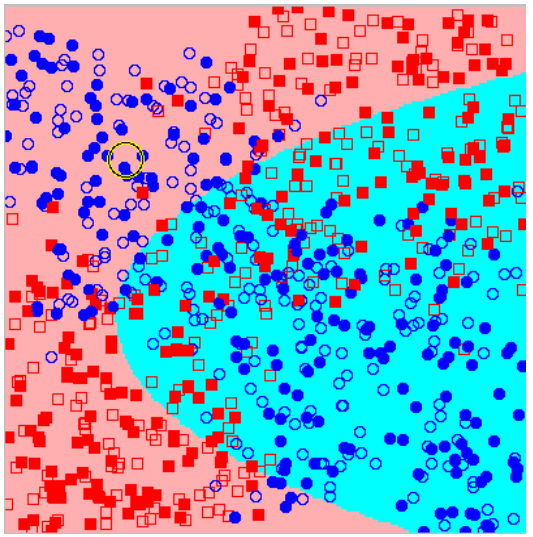
$$w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{14.18}$$

3. Make predictions using the final model, which is given by

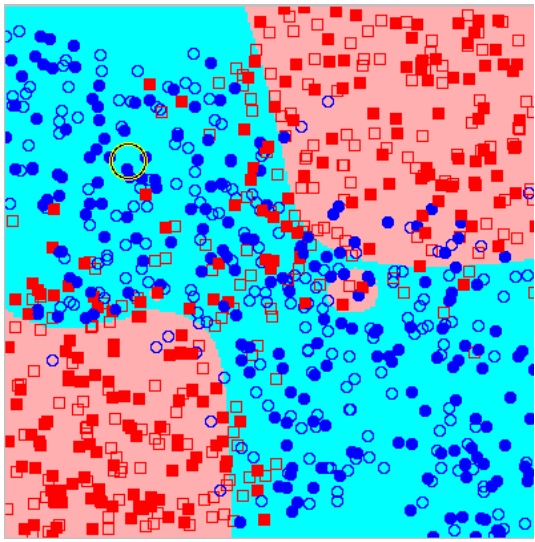
$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \right). \tag{14.19}$$

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



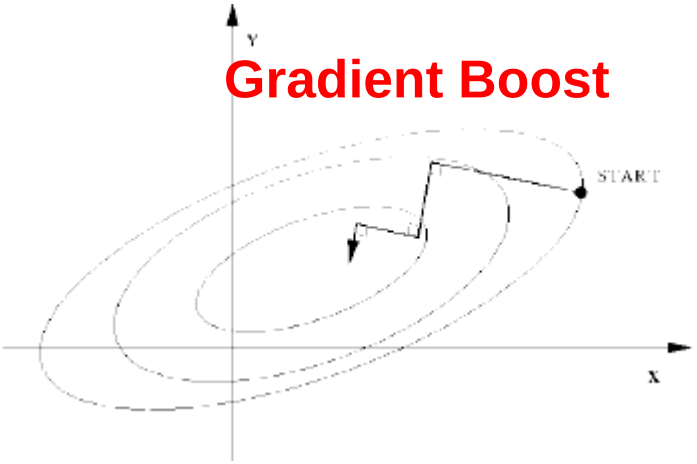
Wzmocniony klasyfikator bayesowski

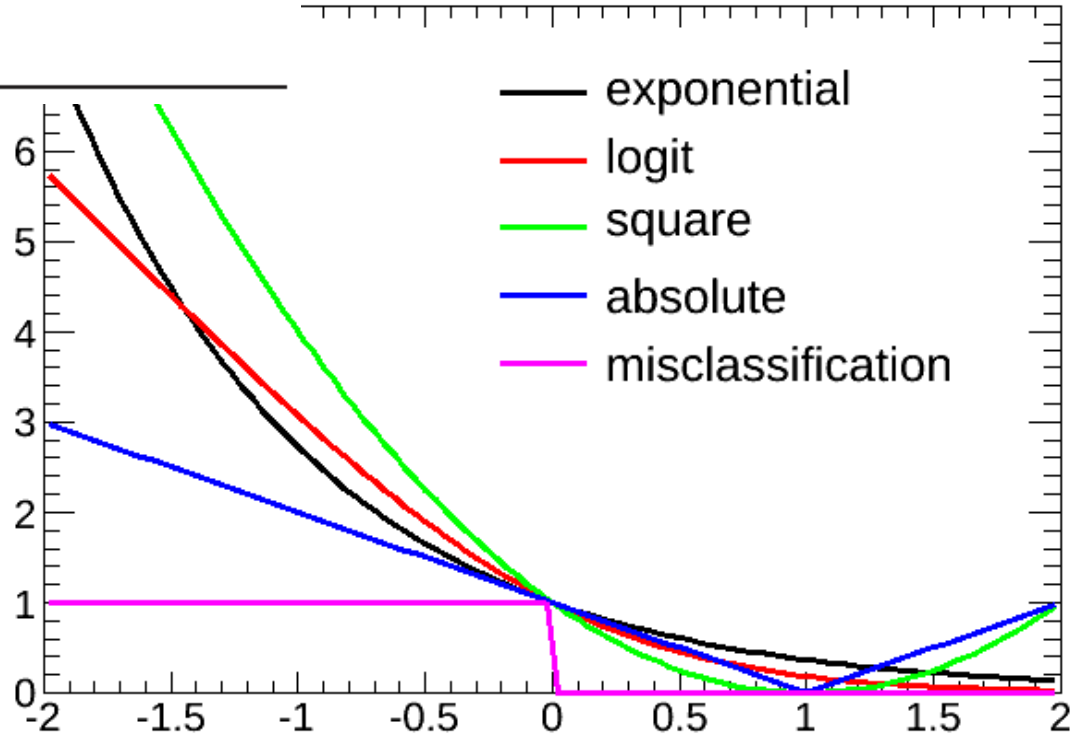
Zastosowanie algorytmu wzmacniania (5 iteracji) do naiwnego klasyfikatora bayesowskiego.

Boosting – inne funkcje straty

- Przykłady innych funkcji straty używanych w algorytmach typu boosting:

Nazwa funkcji	Równanie	Nazwa algorytmu
wykładnicza	$\exp(-t_n f_m(x_n))$	*AdaBoost*
prawdopodobieństwa	$\log(1+\exp(-2t_n f_m(x_n))$	*LogitBoost*
błąd kwadratowy	$(t_n - f_m(x_n))^2$	*SquareBoost*
błąd absolutny	$\lvert t_n - f_m(x_n)\rvert$	
zero-jedynkowa	$\begin{cases} 0 & sign(f_m(x_n)) \neq t_n \\ 1 & sign(f_m(x_n)) = t_n \end{cases}$	



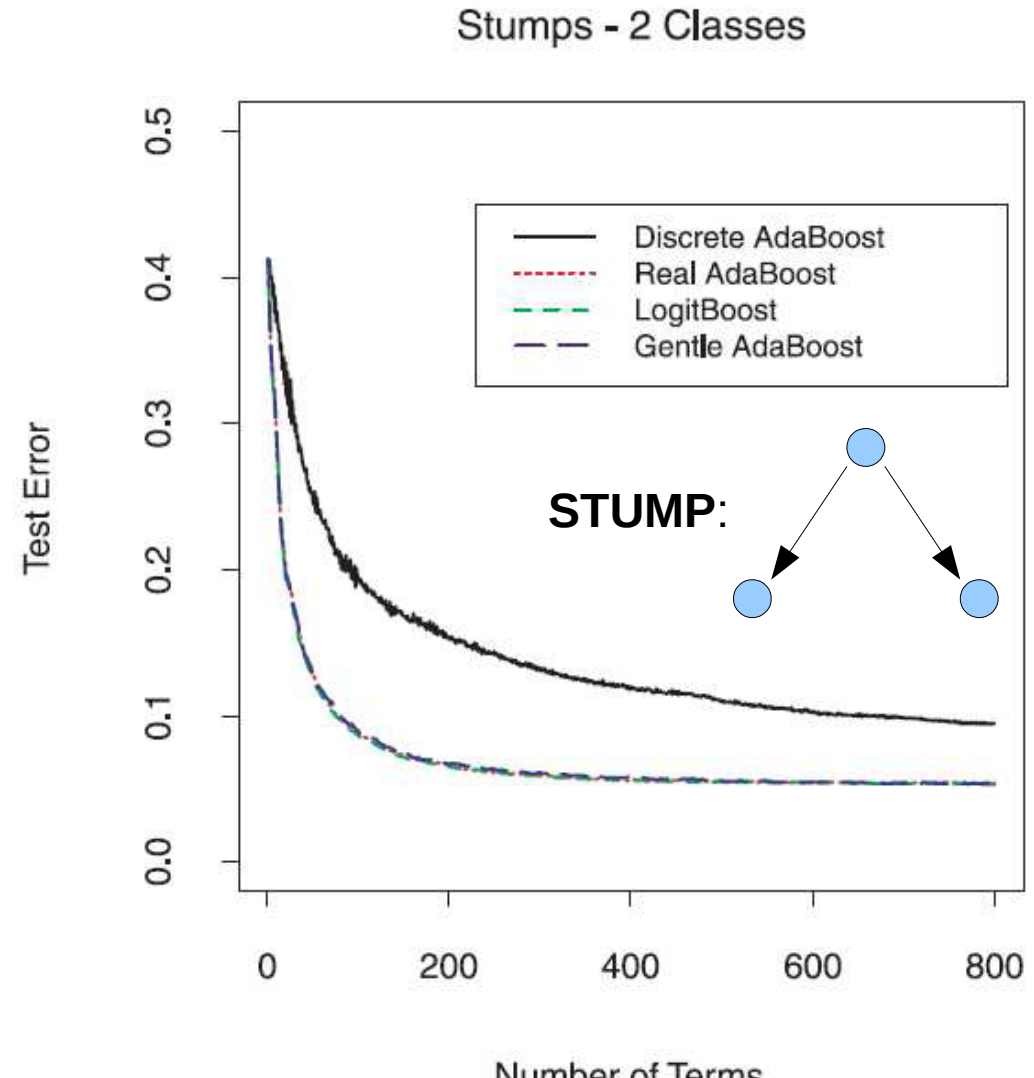


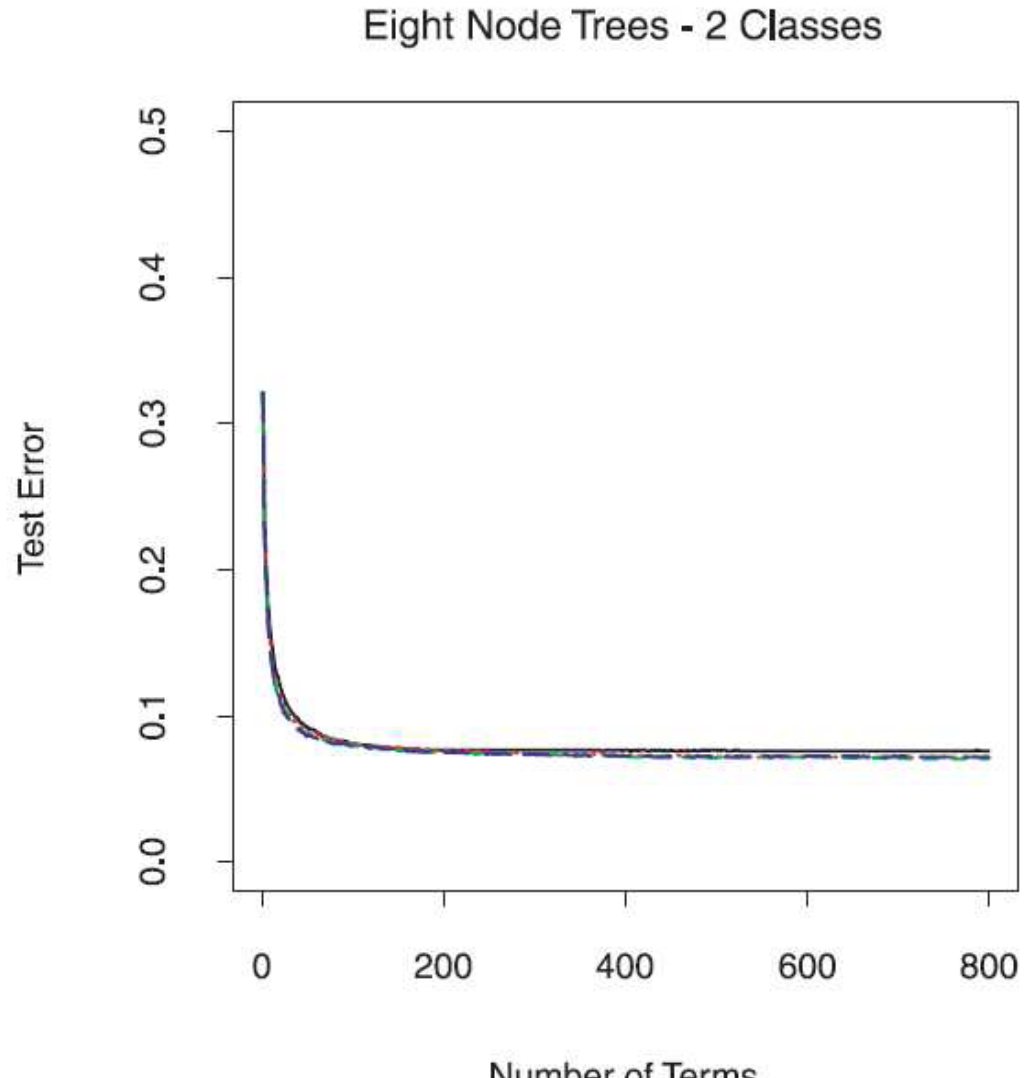
Zalety:
Nie są tak czułe na „odstające” punkty
Odpowiednio dobrana „czułość” jest ważna, gdy w danych są obszary gdzie miesza się sygnał i tło (błąd Bayesowski jest duży).

Wady:
Nie mamy szybkiego i prostego algorytmu minimalizacji jak w AdaBoost => minimalizacja jak w ogólnych przypadkach, np. dopasowywania funkcji.

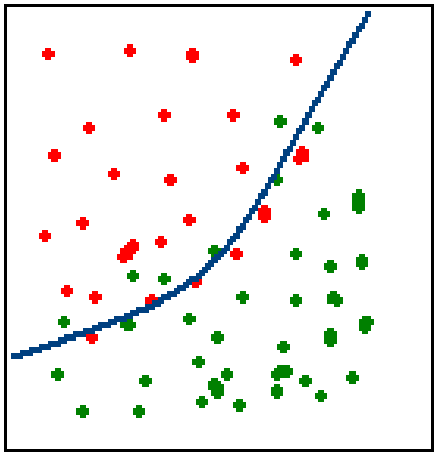
Boosting – różne funkcje straty

- Discrete AdaBoost – opisany poprzednio podstawowy AdaBoost
- Real AdaBoost, LogitBoost, Gentle AdaBoost- modyfikacje (różne funkcje straty oraz algorytmy minimalizacji) zaproponowane przez Friedmana.
- Przy większych drzewach wyniki identyczne.

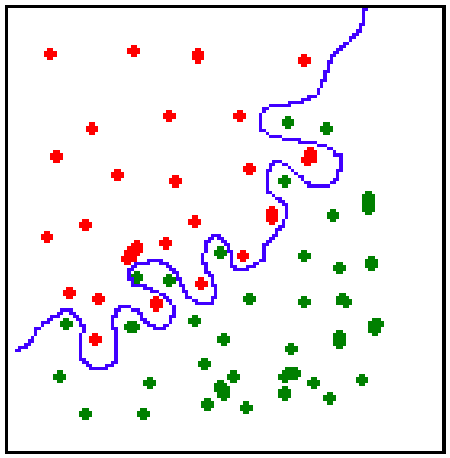




Przetrenowanie

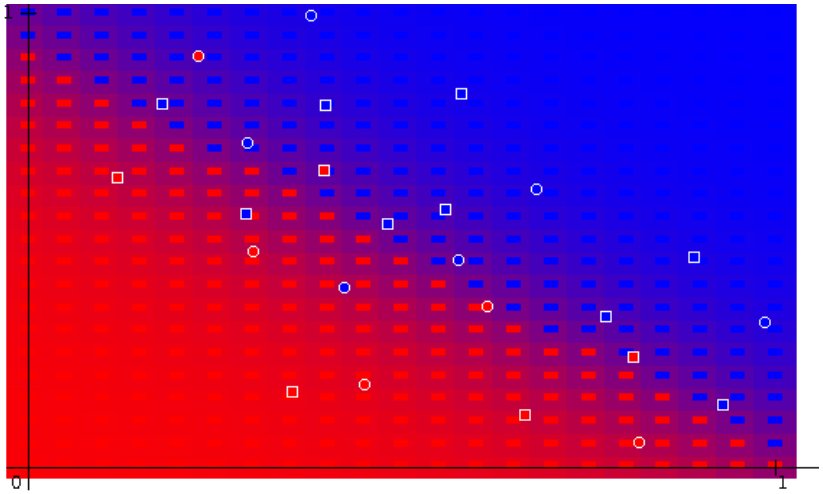


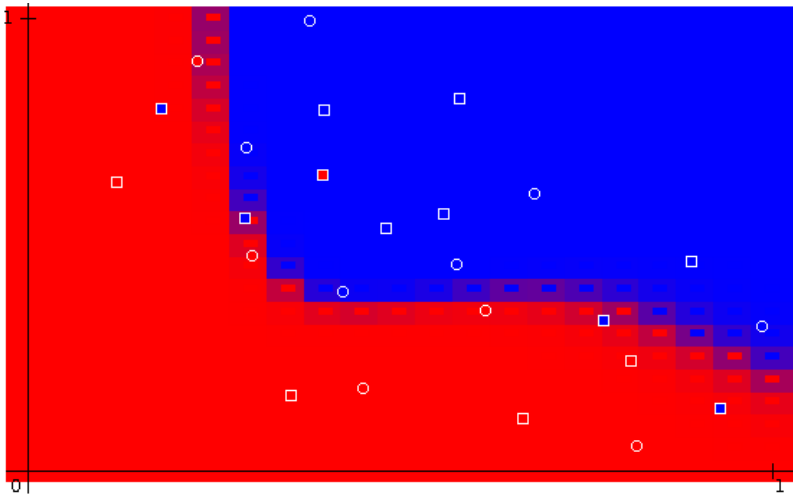
Poprawnie

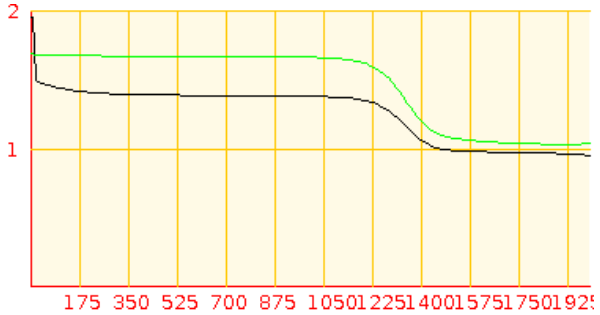


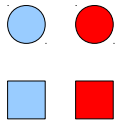
Przetrenowanie

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



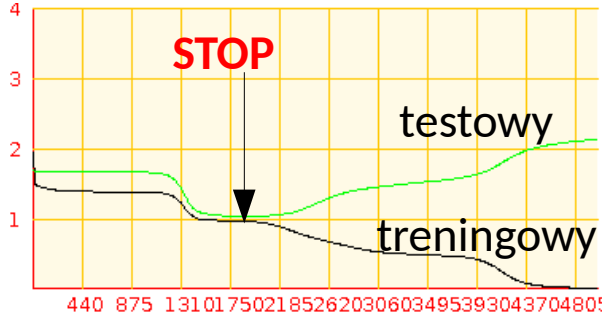






zbiór treningowy

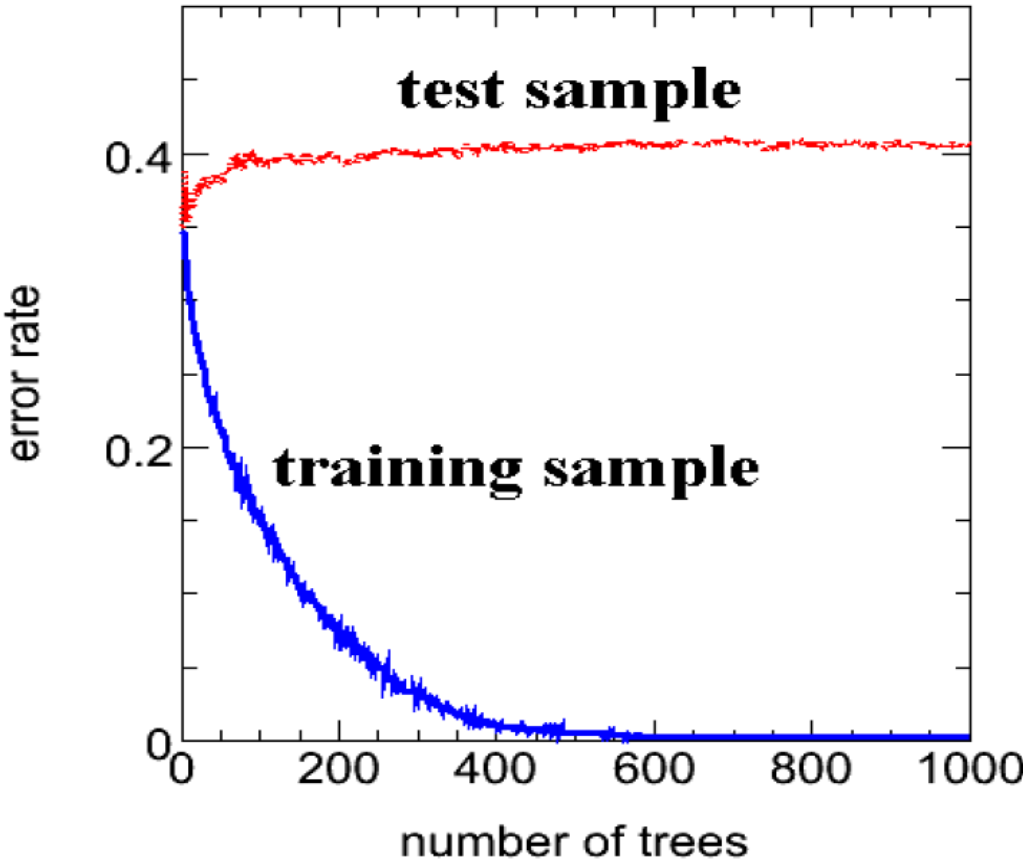
zbiór testowy



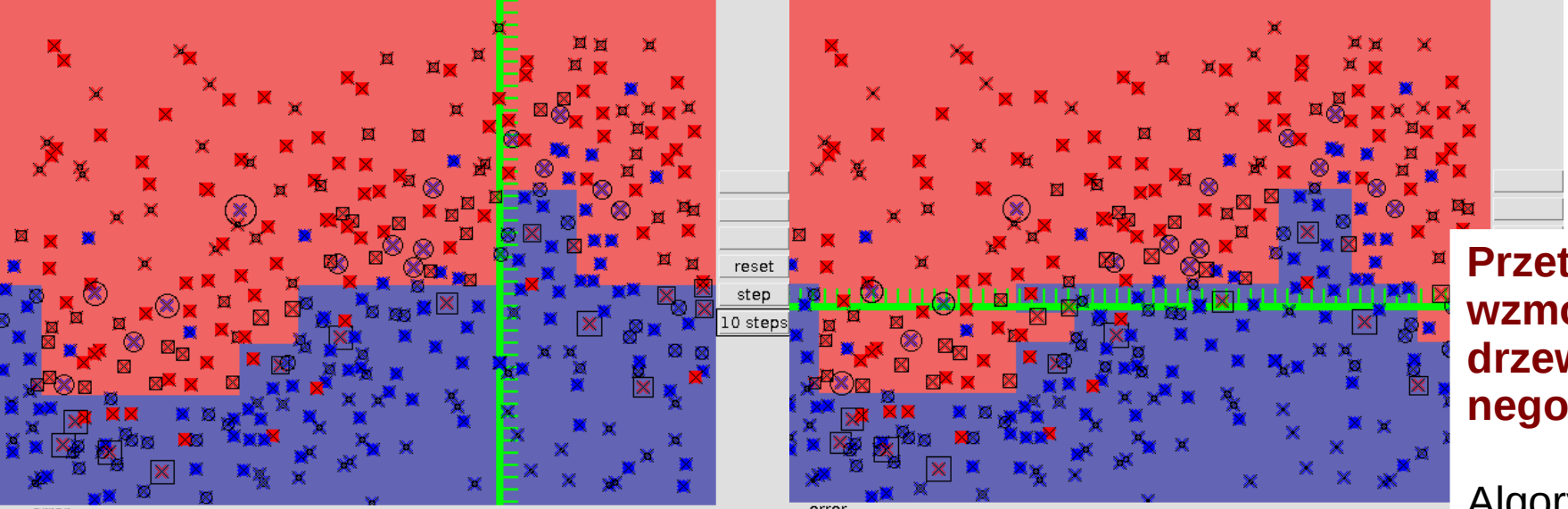
Przykład z użyciem sieci neuronowej.

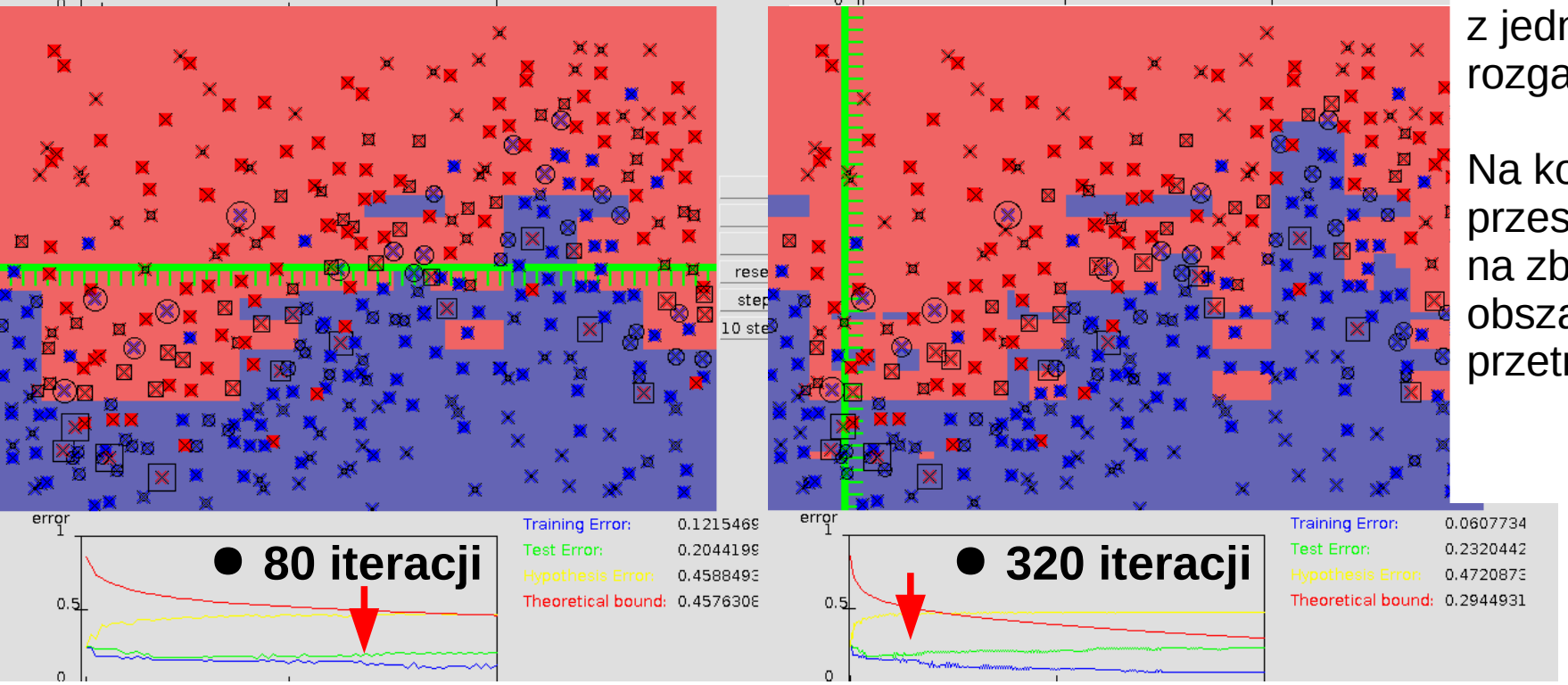
Odporność na przeuczenie

- Po zastosowaniu klasyfikatora do zbioru danych treningowych widać, że błąd klasyfikacji spada wykładniczo do zera wraz ze wzrostem liczby użytych drzew decyzyjnych.
- Tym samym błąd klasyfikacji dla niezależnego zbioru testowego powinien rosnąc (przeuczenie).
- Jednak nic takiego nie widzimy, dla zbioru testowego błąd pozostaje w zasadzie stały!
- Efekt nie do końca zrozumiały...



Rezultat użycia algorytmu AdaBoost do separacji przypadków mSUGRA od przypadków $t\bar{t}$ przy energiach LHC





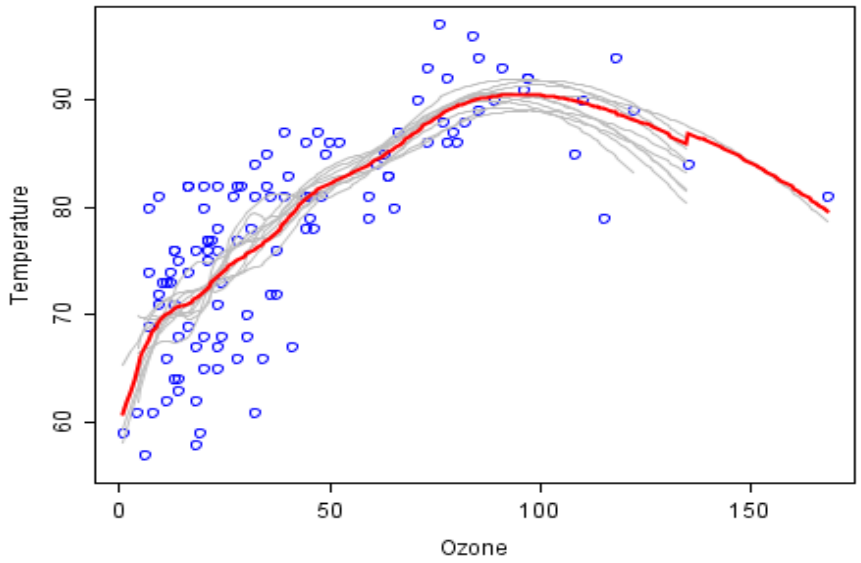
Przetrenowanie wzmocnionego drzewa decyzyjnego (BDT)

Algorytm wzmacniania zastosowany do drzew decyzyjnych z jednym rozgałęzieniem.

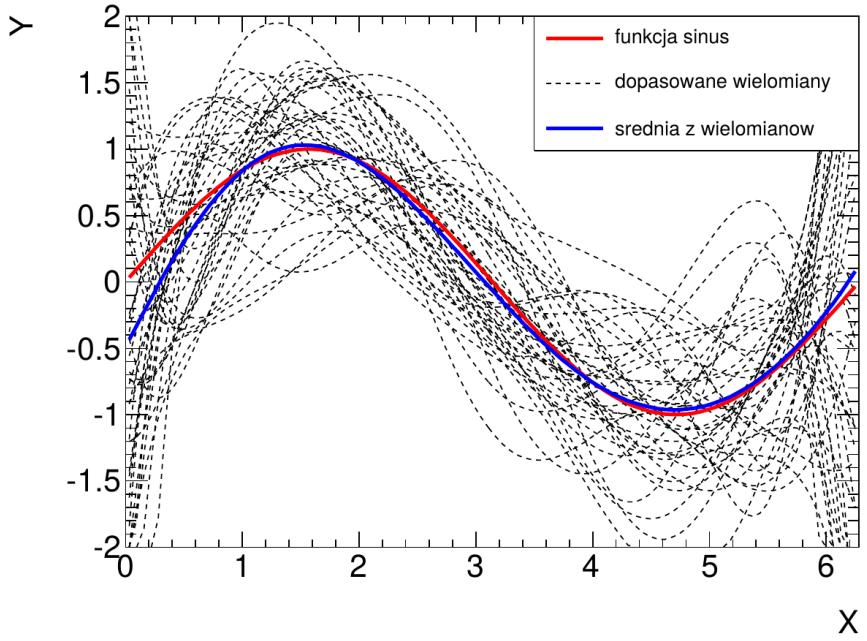
Na koniec dzieli przestrzeń na zbyt drobne obszary - przetrenowanie.

Bagging (Bootstrap AGGregatING)

- **Algorytm zaproponowany przez Leo Breimana w 1994 roku:**
  - Losuj N przypadków z N-elementowego zbioru treningowego z powtórzeniami
  - Trenuj na tym zbiorze klasyfikator.
  - Klasyfikuj nowe przypadki poprzez głosowanie wszystkich drzew decyzyjnych.



Dla tych danych czerwona linia (średnia ze 100 klasyfikatorów) jest gładsza, stabilniejsza, mniej podatna na przetrenowanie niż każdy z nich.



Analogia: uśrednienie wielu kiepsko pasujących funkcji daje dobre dopasowanie

Bagging

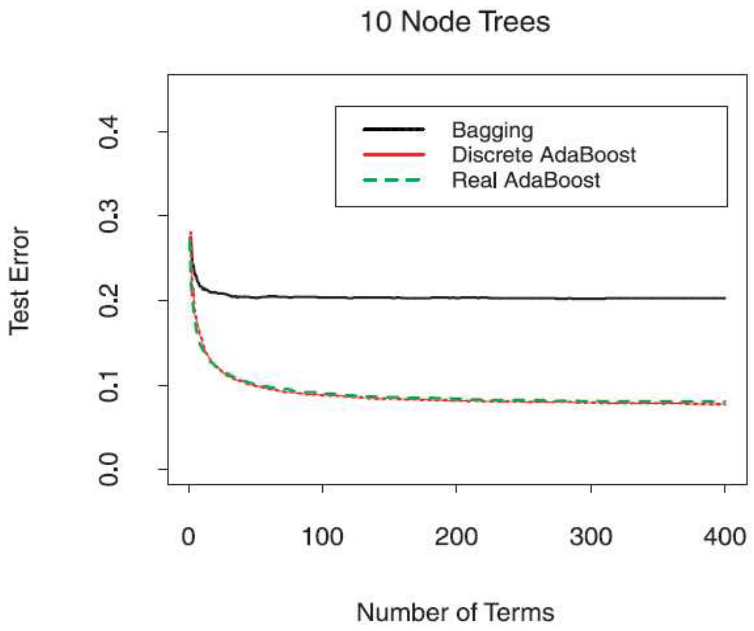
- **Random Forest** - dalsze rozwinięcie metody Bagging:
  - K razy losujemy N elementów ze zbioru N elementowego
  - Dla każdego zbioru trenujemy drzewo decyzyjne
  - Budując drzewo w każdym rozgałęzieniu używamy tylko m losowo wybranych zmiennych (dodatkowa randomizacja).

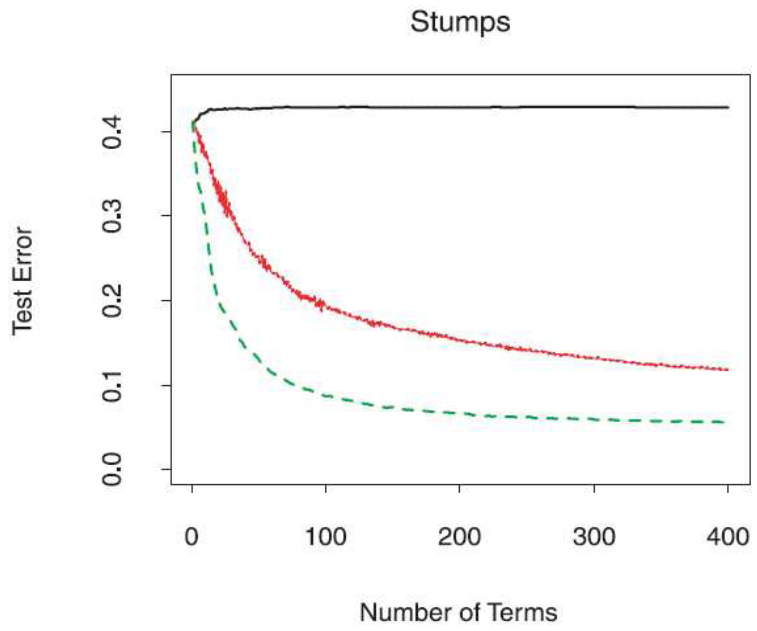
- Boosting przeważnie daje lepsze przewidywania, ale bagging ma tendencję do zachowywania się lepiej w obecności szumu albo przypadków bardzo odstających od reszty (“outliers”).

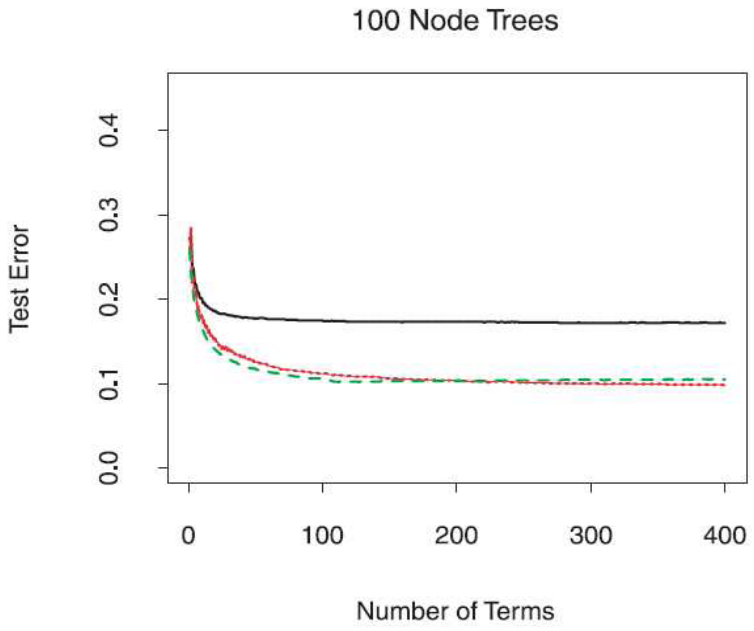
  Bauer and Kohavi, “An empirical comparison of voting classification algorithms”, Machine Learning 36 (1999)

- Bagging jest algorytmem równoległym, boosting sekwencyjnym.

Bagging vs. boosting





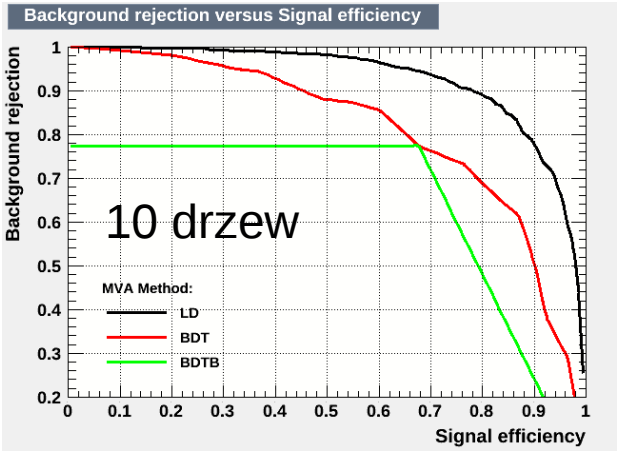


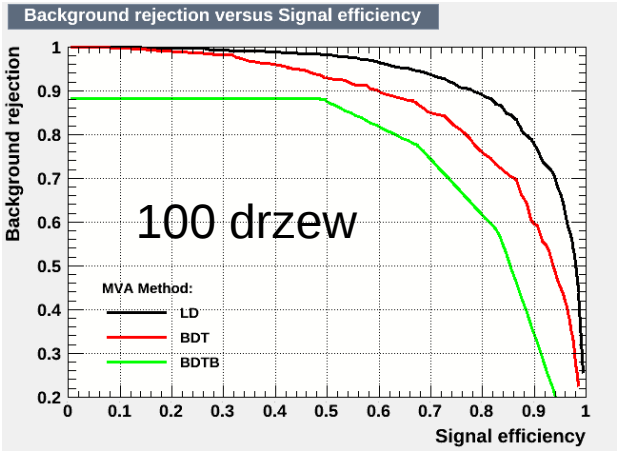
Porównanie zależności błędu dla różnych algorytmów Boosting i Bagging dla klasyfikacji na dwie klasy w funkcji liczby iteracji.
Widać, że dla "stumps" (bardzo małych drzew decyzyjnych) zbieżność Discrete AdaBoost jest gorsza, niż Real AdaBoost. Bagging spisuje się gorzej niż boosting [Friedman 2000].

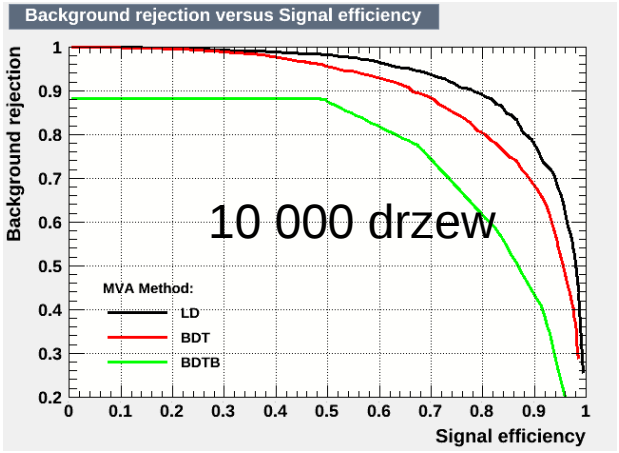
LogitBoost minimalizuje: $\sum_i \log\left(1 + e^{-y_i f(x_i)}\right)$

Małe ćwiczenie z TMVA

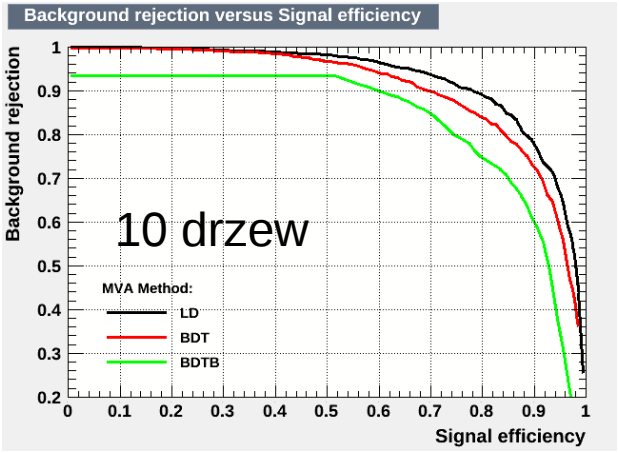
Separacja dwóch rozkładów Gaussa w 4 wymiarach.

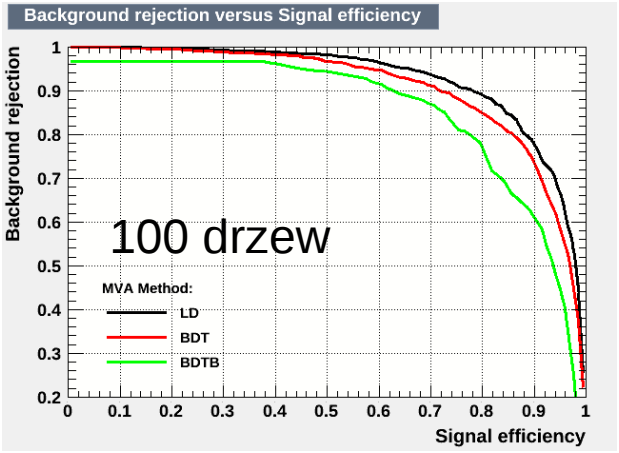


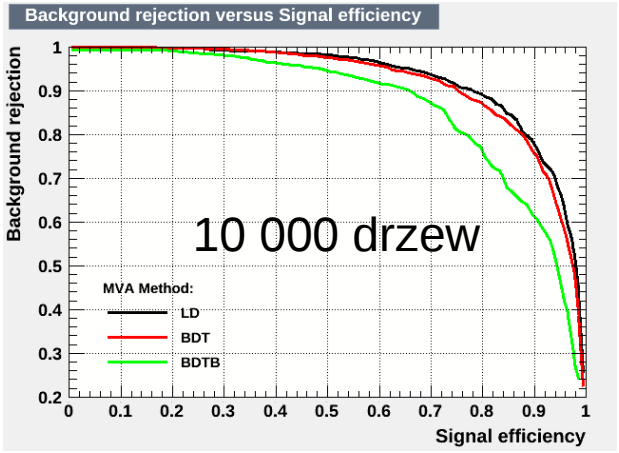




STUMPS (drzewa o głębokości 1)

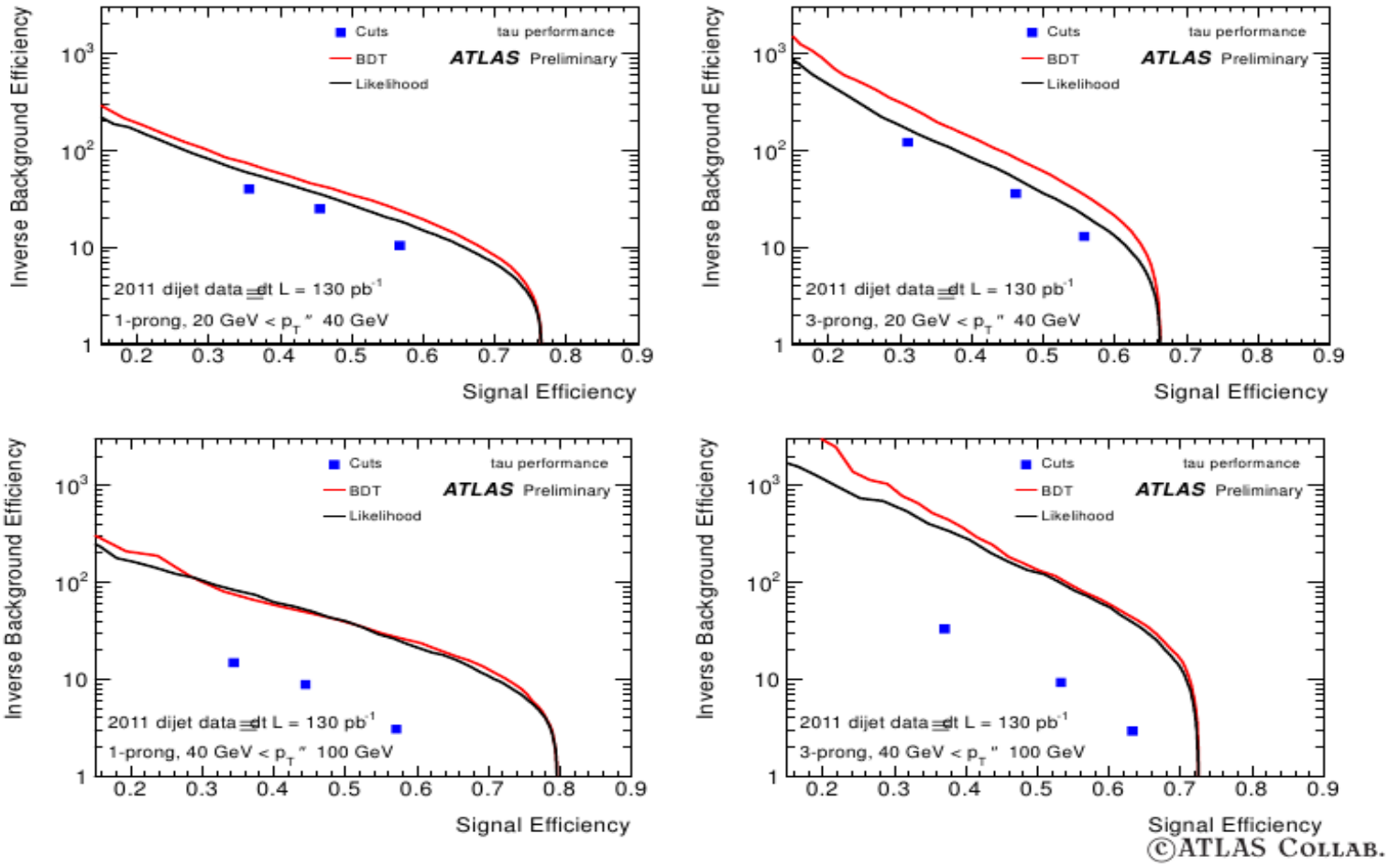






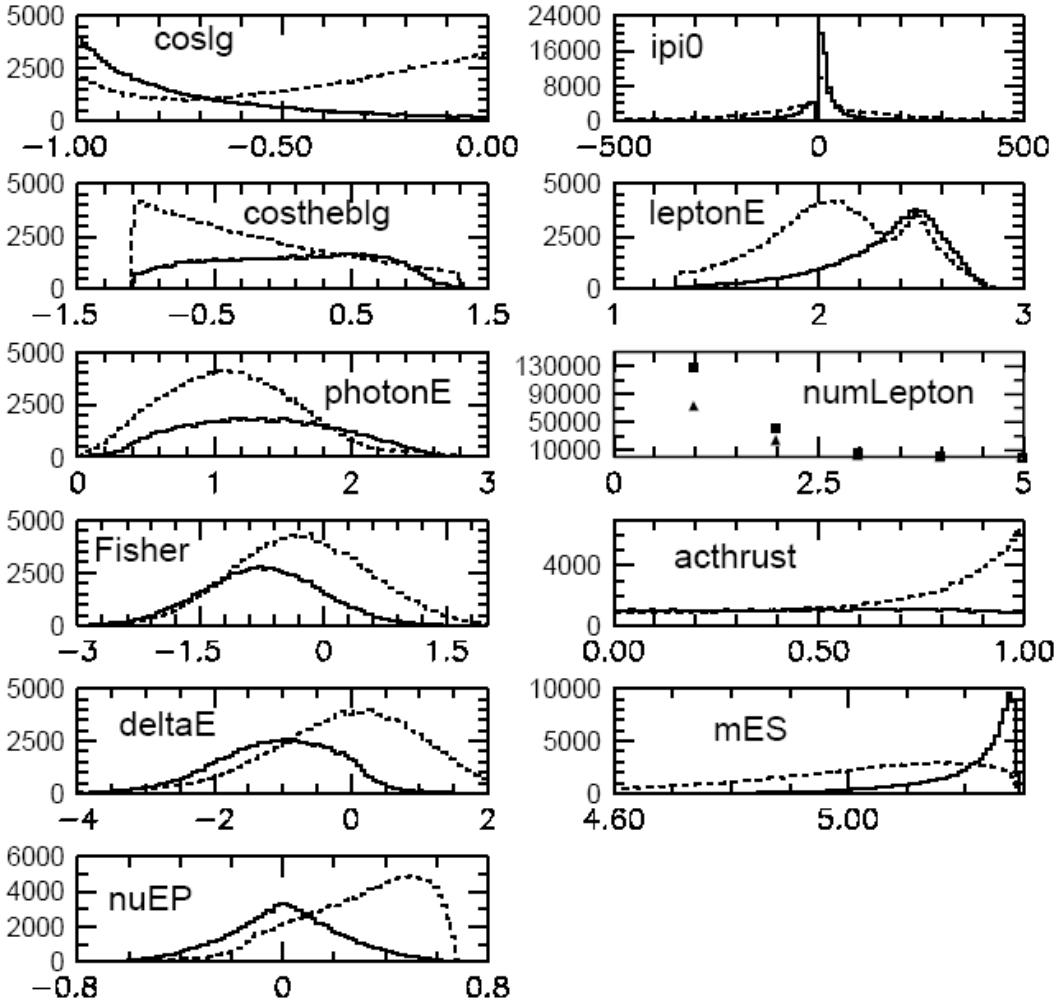
Drzewa o głębokości 10

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.**

Zastosowanie do poszukiwań $B \to \gamma l \nu$ w eksperymencie BaBar



11-D data: 10 ciągłych i 1 dyskretna zmienna.

Niektóre zmienne są silnie skorelowane ρ(Fisher,acthrust)=0.8 oraz ρ(costheblg,nuEP)=-0.87.

Dane Monte Carlo:

- trening = 500K
- walidacja = 250K
- test = 250K

Wyniki

Znaczenie sygnału (signal significance) uzyskane za pomocą różnych metod klasyfikacji:

Method	$B \to \gamma e \nu$					$B \to \gamma \mu \nu$				
	$\mathcal{S}_{\text{train}}$	$\mathcal{S}_{\text{valid}}$	$\mathcal{S}_{\text{test}}$	W_1	W_0	$\mathcal{S}_{\text{train}}$	$\mathcal{S}_{\text{valid}}$	$\mathcal{S}_{\text{test}}$	W_1	W_0
Original method	2.66	-	2.42	37.5	202.2	1.75	-	1.62	25.8	227.4
Decision tree	3.28	2.72	2.16	20.3	68.1	1.74	1.63	1.54	29.0	325.9
Bump hunter with one bump	2.72	2.54	2.31	47.5	376.6	1.76	1.54	1.54	31.7	393.8
AdaBoost with binary splits	2.54	2.65	2.27	84.2	1288.5	1.68	1.74	1.47	49.7	1087.7
AdaBoost with decision trees	13.63	2.99	2.62	58.0	432.8	11.87	1.97	1.75	41.6	523.0
Combiner of background subclassifiers	3.03	2.88	2.49	83.2	1037.2	1.84	1.90	1.66	55.2	1057.1
Bagging with decision trees	9.20	3.25	**2.99**	69.1	465.8	8.09	2.07	**1.98**	49.4	571.1

Trening 50 „boosted decision trees” lub 100 „bagged decision trees” zajął kilka godzin na komputerach SLACu.

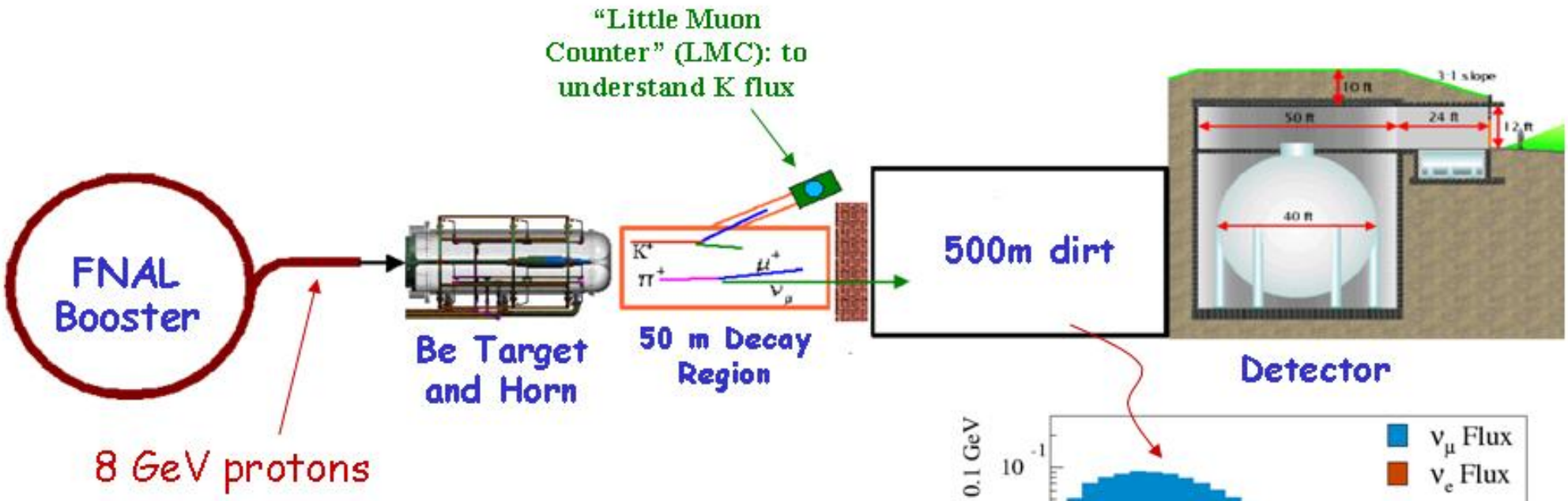
W_1 i W_0 są oczekiwanymi liczbami przypadków sygnału i tła przy świetlności 210 fb^{-1} .

Najczystszy sygnał z użyciem „bagging decision trees”.

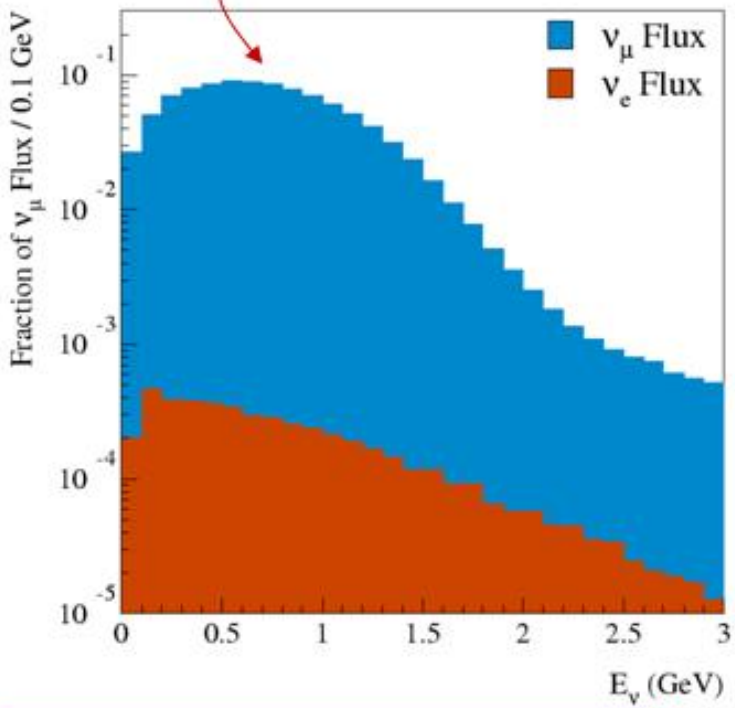
„Bagged decision trees” poprawiają o 14% czystość sygnału w porównaniu do „boosted decision trees”:

- **8% dzięki użyciu „bagging” zamiast „boosting”.**
- **9% dzięki optymalizacji S/√(S+B) zamiast indeksu Gini.**

Eksperyment Mini-BooNE – wiązka neutrin



- Proton flux ~ 6E16 p/hr (goal 9E16 p/hr)
  - ~ 1 detected neutrino/minute
  - L/E ~ 1

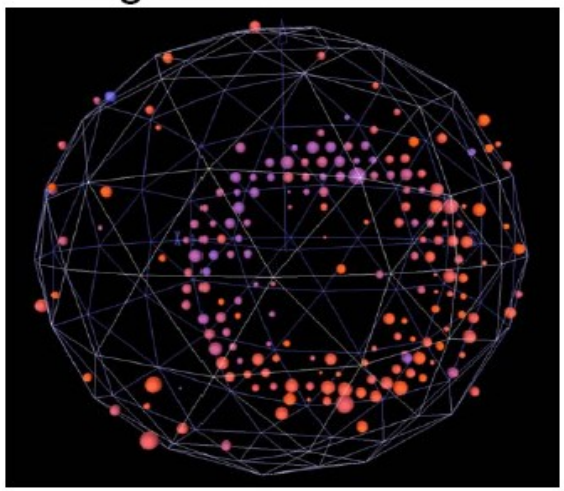


Przykładowe przypadki

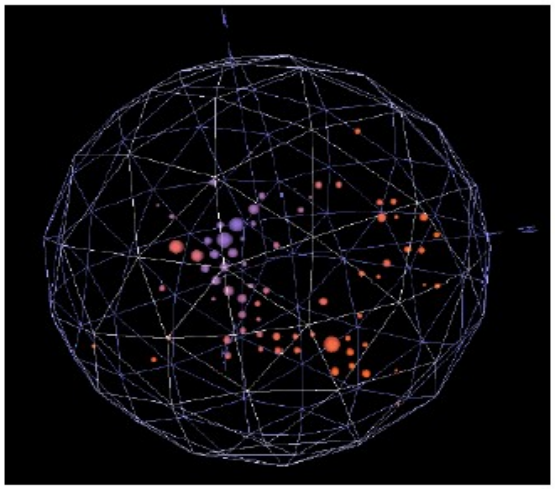
Pattern of hit tubes (with charge and time information)
allows reconstruction of track location and direction
and separation of different event types.

e.g. candidate events:

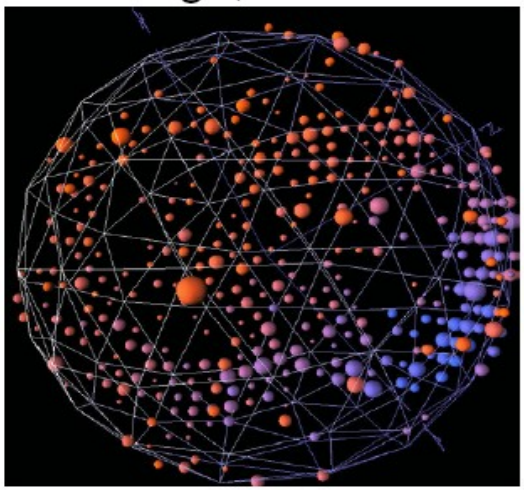
size = charge, color = time



muon
from ν_μ interaction



Michel electron
from stopped μ decay
after ν_μ interaction



$\pi^0 \rightarrow$ two photons
from ν_μ interaction

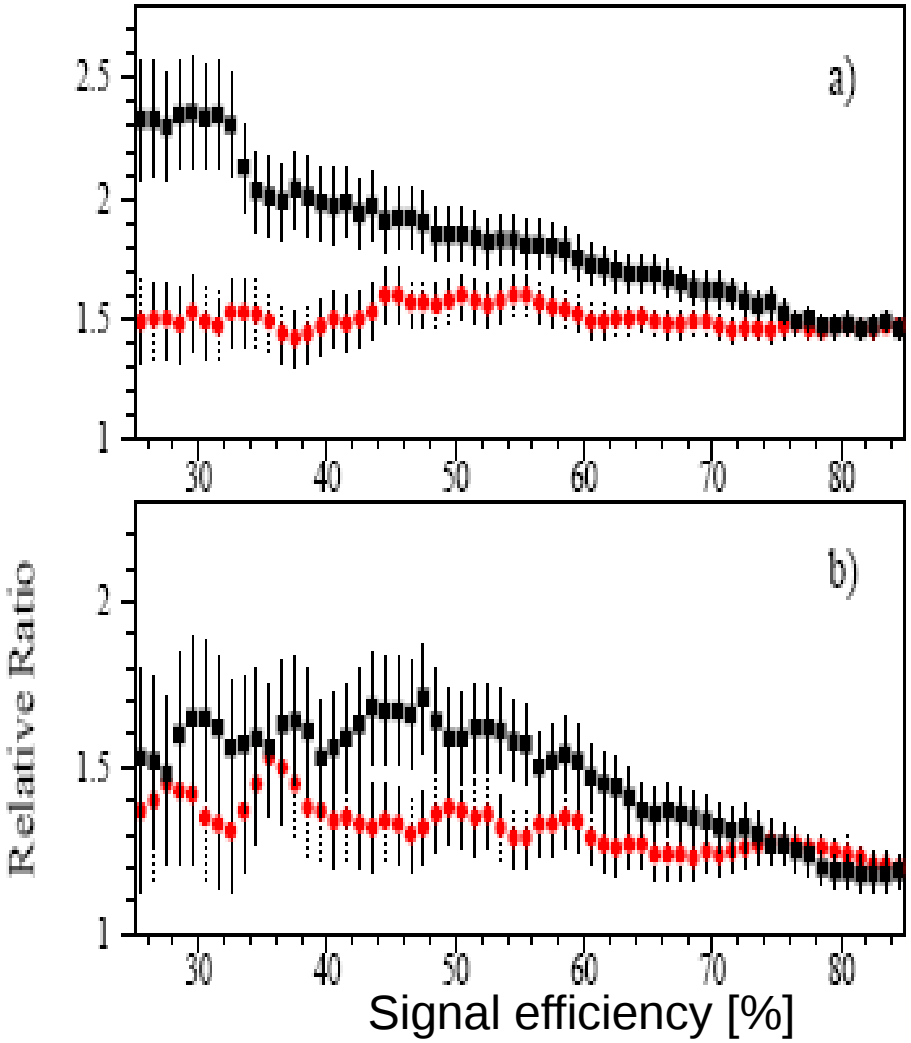
Porównanie drzew decyzyjnych i sieci neuronowej

- A. Tło: “cocktail events”. Czerwony - 21 zmiennych, czarny - 52 zmienne.
- B. Tło: przypadki z Pi0. Czerwony - 22 zmienne, czarny - 52 zmienne.

- Pokazany stosunek:

$$R = \frac{N_{tło}(Sieć\ Neuronowa)}{N_{tło}(Drzewo\ Decyzyjne)}$$

- Użycie drzew decyzyjnych dało lepsze wyniki oraz szybszy trening.



Podsumowanie

- Boosting, bagging – w „cudowny” sposób można uzyskać silny klasyfikator ze słabego
- Stosowany przeważnie dla drzew decyzyjnych – bo proste, szybkie w nauce i klasyfikacji.
- Daje dobrze wyniki, często bez pracochłonnego tuningu parametrów

  - „the best out-of-box classification algorithm”.
- Coraz popularniejszy w fizyce wysokich energii
- Może po prostu modny...

Czy należy teraz już tylko stosować BDT:

„Jeżeli metoda daje wynik zbliżony do limitu Bayesa, to nie jest możliwe poprawienie wyniku, nawet poprzez stosowanie najbardziej wymyślne metody klasyfikacji.”

Harrison B. Prosper