Zespoły modeli predykcyjnych
boosting i inne wykład 9

Jerzy Stefanowski

Instytut Informatyki PP

2021

Akademia Innowacyjnych Zastosowań Technologii Cyfrowych (AI-TECH)
projekt finansowany z środków Programu Operacyjnego Polska Cyfrowa
POPC.03.02.00-00-0001/20

Fundusze Europejskie
Polska Cyfrowa

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Rozwoju Regionalnego

Plan wykładu 9 i częściowo 10tego

- Metody Boosting
  - AdaBoost
- Porównania Bagging vs. Boosting
- Boosted trees ensembles (funkcyjny boosting)

---------- kolejny wykład

- Zróżnicowanie klasyfikatorów składowych
- Generalizacja stosowa (stacking) i tzw. mixture of experts
- Podejścia zespołowe do danych silnie wieloklasowych
- Podsumowanie

Boosting

- Schapire [1990] rozważanie teoretyczne, jak można tzw. słabe modele (ang. weak learner) rozbudować do predyktorów o lepszej trafności (ang. strong learner)
  - **Weak learner** − algorytm tworzenia klasyfikatora o trafności trochę lepszej niż głosowanie większościowe (>0.5 dla klasyfikacji binarnej)
- Wprowadził ideę tzw. **wzmacniania klasyfikatora** (ang. boosting) poprzez odpowiednio ukierunkowane losowanie przykładów do zbiorów uczących dodatkowych klasyfikatorów
- Później [Freund & Shapire, 1996] rozwinięte do praktycznego i efektywnego algorytmu AdaBoost
  - Wagowanie przykładów – może być realizowane przez zmianę prawdopodobieństwa wylosowania
- Liczne rozszerzenia i zastosowania

Boosting - inspiracje

- Schapire [1990] weak learners – proponuje użyć/uczyć trzy klasyfikatory C1,C2 lub C3 na podzbiorach z oryginalnych danych uczących *S*
  - Pierwszy C1 uczony na wylosowanym podzbiorze S1 z *S*
  - Stwórz kolejny podzbiór S2 z *S* złożony w połowie z przykładów poprawnie sklasyfikowanych przez C1, a w połowie źle sklasyfikowanych (te przykłady odpowiednio losowane z S)
  - Naucz nowy klasyfikator C2 na S2
  - Trzeci klasyfikator C3 uczony na tych przykładach z S, na których predykcje C1 i C2 są niezgodne
- Faza klasyfikacji nowego **x**
  - Użyj klasyfikatorów C1 i C2, jeśli predykcje zgodne, to wynik
  - Jeśli ich predykcje są niezgodne, to użyj C3

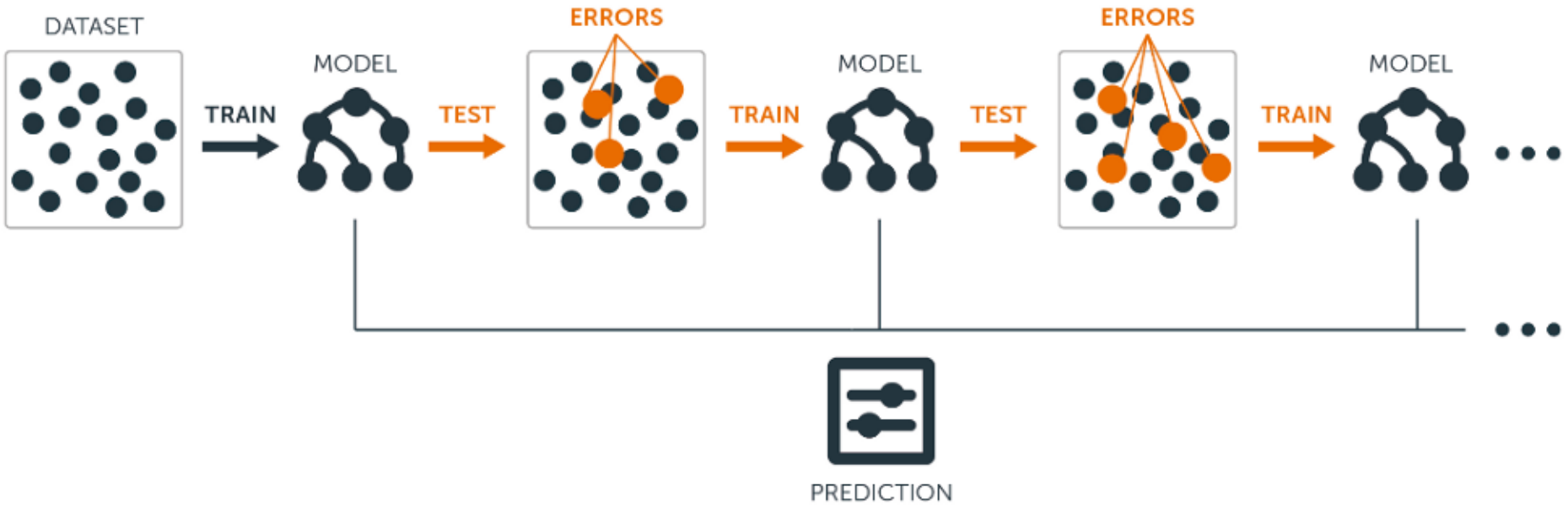
Pierwszy boosting

- Schapire – podał dowód, że dla binarnej klasyfikacji, błąd takiego złożonego klasyfikatora jest z góry ograniczony przez połowę błędu najlepszego z klasyfikatorów składowych C1, C2, C3
- Należy oczekiwać poprawy trafności całego klasyfikatora złożonego
- Możliwe tworzenie innych sekwencyjnych / iteracyjnych rozwiązań -> …

Boosting - AdaBoost

- (Freund & Shapire, 1996) wprowadzili algorytm AdaBoost jako uogólnienie wzmacnia poprzez **iteracyjne wagowanie przykładów**
- Podejście iteracyjne – **sekwencyjne** dodawanie kolejnego klasyfikatora do zespołu.
- W każdej iteracji zmiana rozkładu wag przykładów w S
- Kolejne klasyfikatory w sekwencji, Ct, Ct+1 uczone ze zmodyfikowanego zbioru tak, aby skupiać "zainteresowanie" na przykładach poprzednio źle klasyfikowanych (podniesienie wag źle sklasyfikowanym przykładom, obniżenie wag dobrze sklasyfikowanym)
  - Kolejne składowe klasyfikatory to eksperci od trudnych przykładów
- Predykcja całego systemu – głosowanie ważone / predykcja składowego klasyfikatora z nieliniową funkcją zależną od błędu w fazie uczenia

Ilustracja graficzna



Schemat uczenia AdaBoost

Input: dane oryginalne *S* (*n* przykładów), Algorytm uczący *L*, liczba iteracji *T* (klasyfikatorów składowych C_i)

Inicjalizacja zbioru D_i = przypisz wagi przykładom $w_p=1/n$

For *i* = 1,…, *T* do

1. Naucz klasyfikator C_i na zbiorze D_i
2. Oblicz błąd e_i na zbiorze D_i
3. Jeśli e_i > próg (0.5 dla binarnej klas), to przerwij
4. Oblicz $\beta_i=e_i/(1-e_i)$
5. Zmodyfikuj wagi każdego przykładu dobrze sklasyfikowanemu $w_p=w_p * \beta_i$, a źle sklasyfikowanemu przemnóż przez 1
6. Znormalizuj nowe wagi aby ich suma była 1 ($w_p/\Sigma w$) -> utwórz nowy zbiór D_{i+1}
7. Przejdź do punktu 1

Proces powtarzany do wyczerpania liczby iteracji lub warunku stop e_i

AdaBoost klasyfikowanie

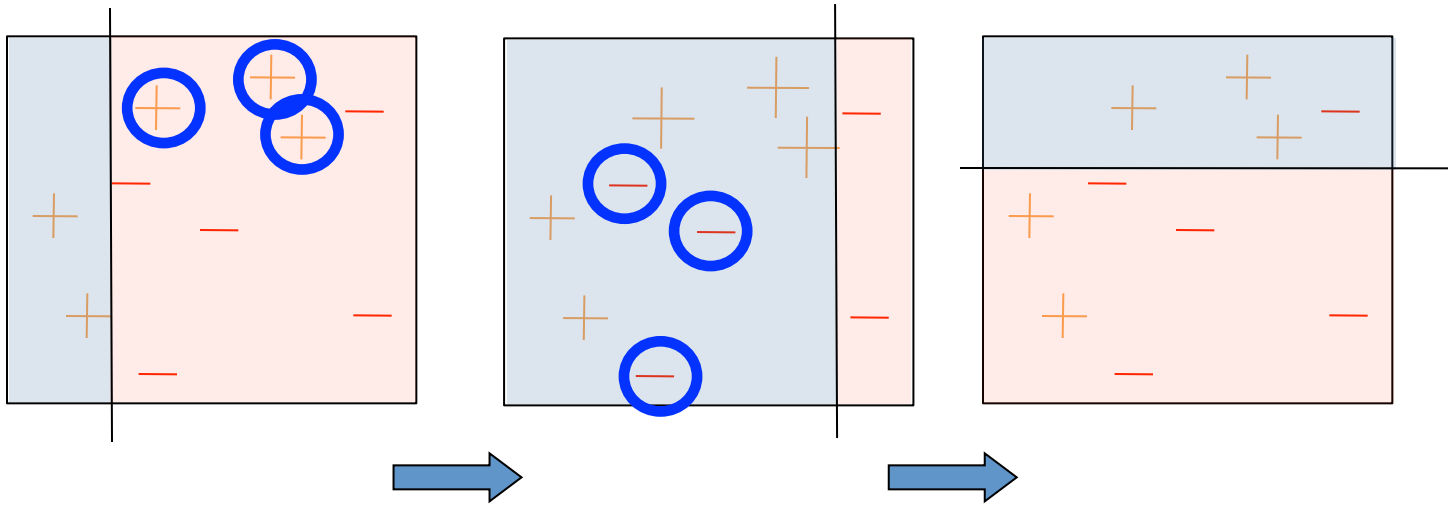
Z każdym klasyfikatorem składowym C_i jest związany współczynnik $\beta_i = e_i/(1-e_i)$

Klasyfikowanie nowego przykładu x

1. Oblicz predykcje klasy x wg każdego klasyfikatora C_i
2. Oblicz sumę wskazań dla każdej z klas K_j

   $V_j = \text{sum}_{\{i:Ci=kj\}} \log(1/\beta_i)$
3. Wybierz klasę z maksymalną wartością V_j

Przykład ilustracyjny wzmacniania



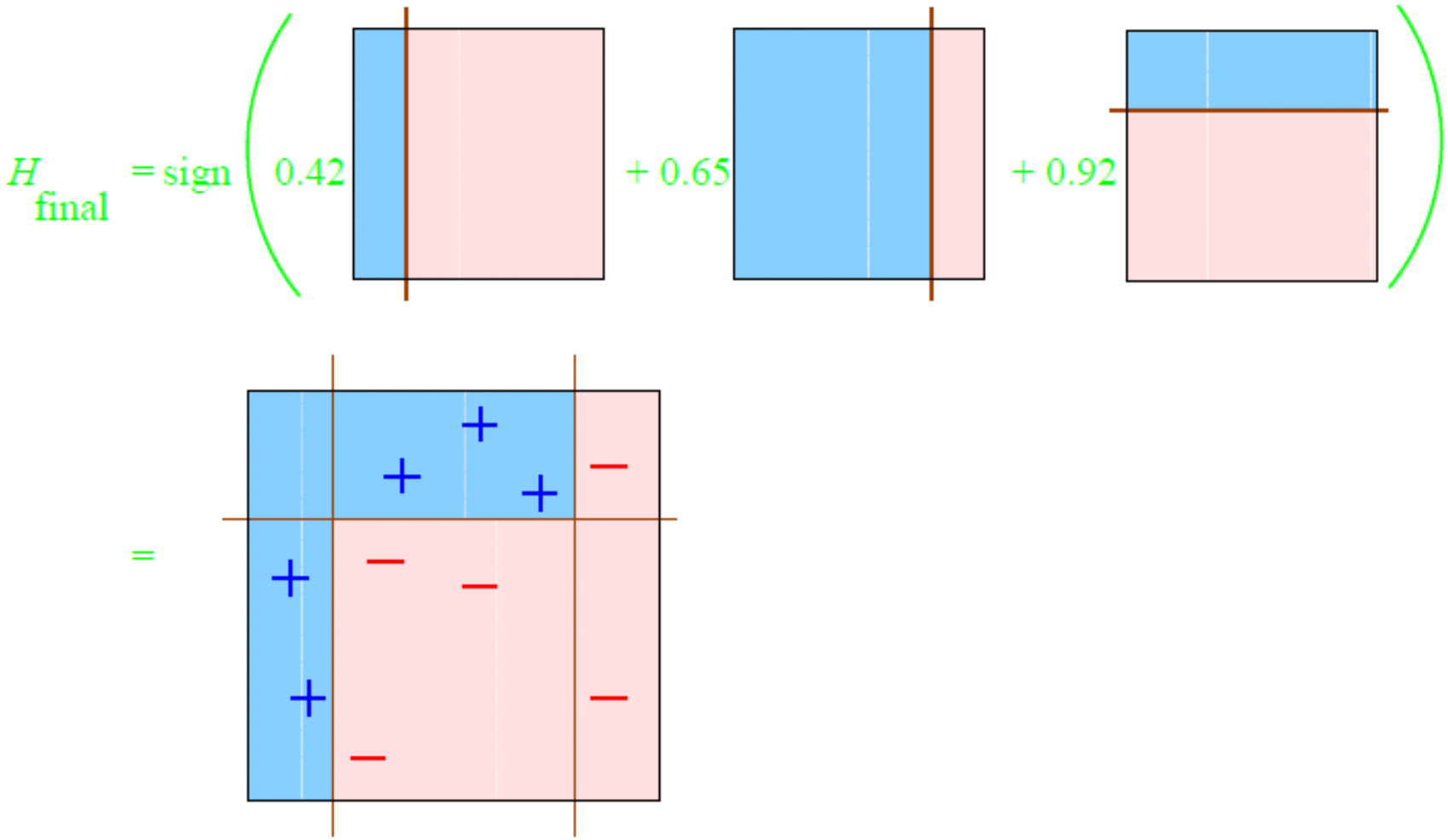
e1=0.3
β=0.42

e2=0.21
β=0.65

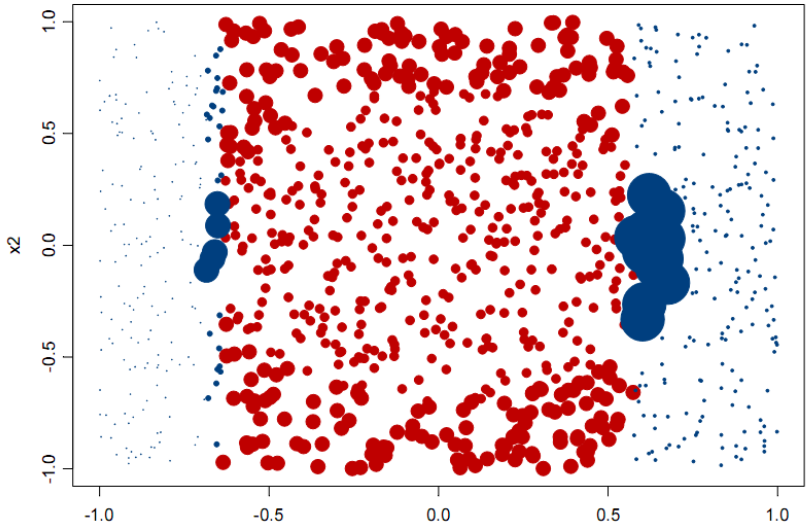
Kolejne iteracje uczenia klasyfikatorów składowych w AdaBoost

= za tutorialem “A Tutorial on Boosting” by Yoav Freund and Rob Schapire

Końcowy zespół klasyfikatorów

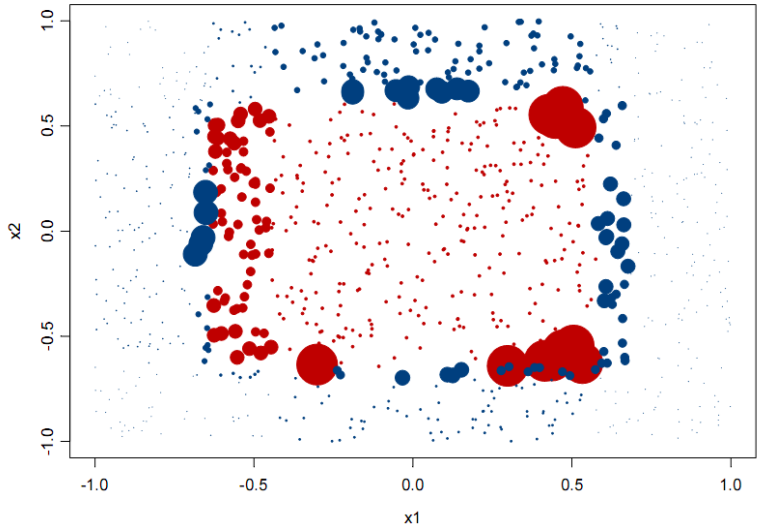


za tutorialem “A Tutorial on Boosting” by Yoav Freund and Rob Schapire

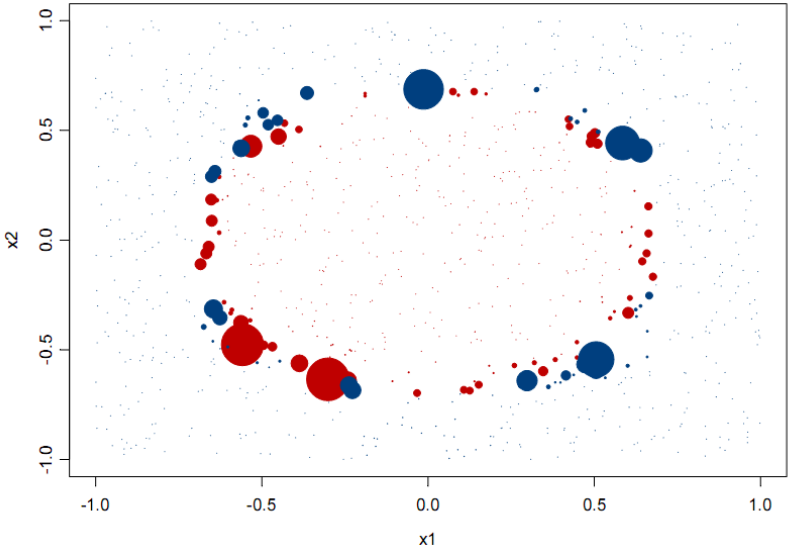


Klasy (kolor) i waga (wielkość) przykładów po 1 iteacji AdaBoost

3 iteracje



20 iteracje



Z wykładu Elder, John. From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods. 2007.

Analiza teoretyczna

Freund i Schapire – oszacowanie błędu AdaBoost

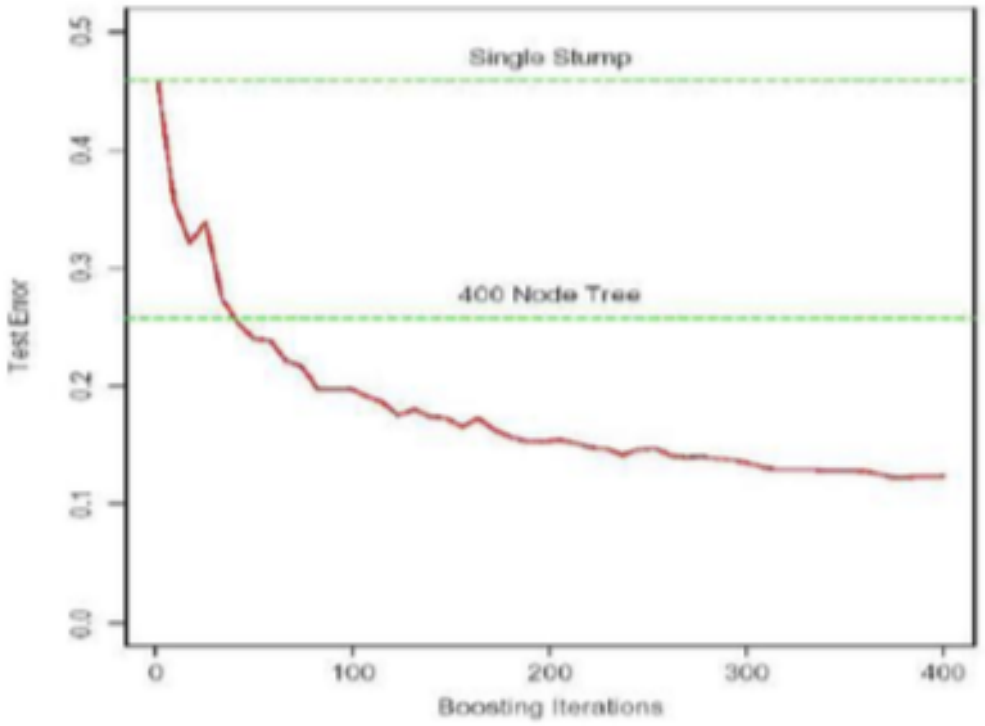
$$E < 2^T \prod_{t=1}^{T} \sqrt{e_t(1-e_t)}$$

gdzie oczekuję się, że błąd klasyfikatora $e_t < 0.5$

Przy pewnych założeniach, błąd powinien maleć wraz ze zwiększaniem liczby składowych klasyfikatorów *T*, lecz nie zawsze! − istnieje w praktyce możliwość przeuczenia

Przykład budowy dużego zespołu

boosting of decision stumps on simulated data



from Hastie, Tibshirani, Friedman: The Elements of Statistical Learning, Springer Verlag 2001

Interpretacje działania boosting

- Spojrzenie z punktu widzenia statystycznej teorii uczenia się – sprawdź Tibishirani, Hastie, Friedman: The Elements of Statistical Learning (pdf wersji autorskiej dostępny, np. https://web.stanford.edu/~hastie/Papers/ESLII.pdf)
- Prace samych autorów = patrz np. R.Schapire Theoretical Views of Boosting.
- Ciekawa własność wzmacniania i sposobów wypracowania decyzji – tzw. **minimalizacja marginesu**, patrz np. https://www.cc.gatech.edu/~isbell/tutorials/boostingmargins.pdf

Uwagi

- Adaptacja wag odbywa się w ten sposób, aby uczynić problem możliwie trudnym dla następnego klasyfikatora. Jednocześnie dla takich wag poprzedni klasyfikator stanie się słabym klasyfikatorem / działa inaczej niż kolejne
- Klasyfikator z mniejszym błędem e_i, a co za tym idzie, z większym $\log(1/\beta_i)$ ma większy wpływ na ostateczną decyzję zespołu
- Boosting może często dać poprawę trafności, lecz nie zawsze zwiększanie iteracji jest właściwe / przeuczenie
- Algorytm uczący *L* musi być zdolny do uwzględnienia wag przykładów
- Jeśli nie, alternatywne podejścia − zmiana prawdopodobieństwa wylosowania przykładu proporcjonalna do wag

Charakterystyka Adaboost

Pro

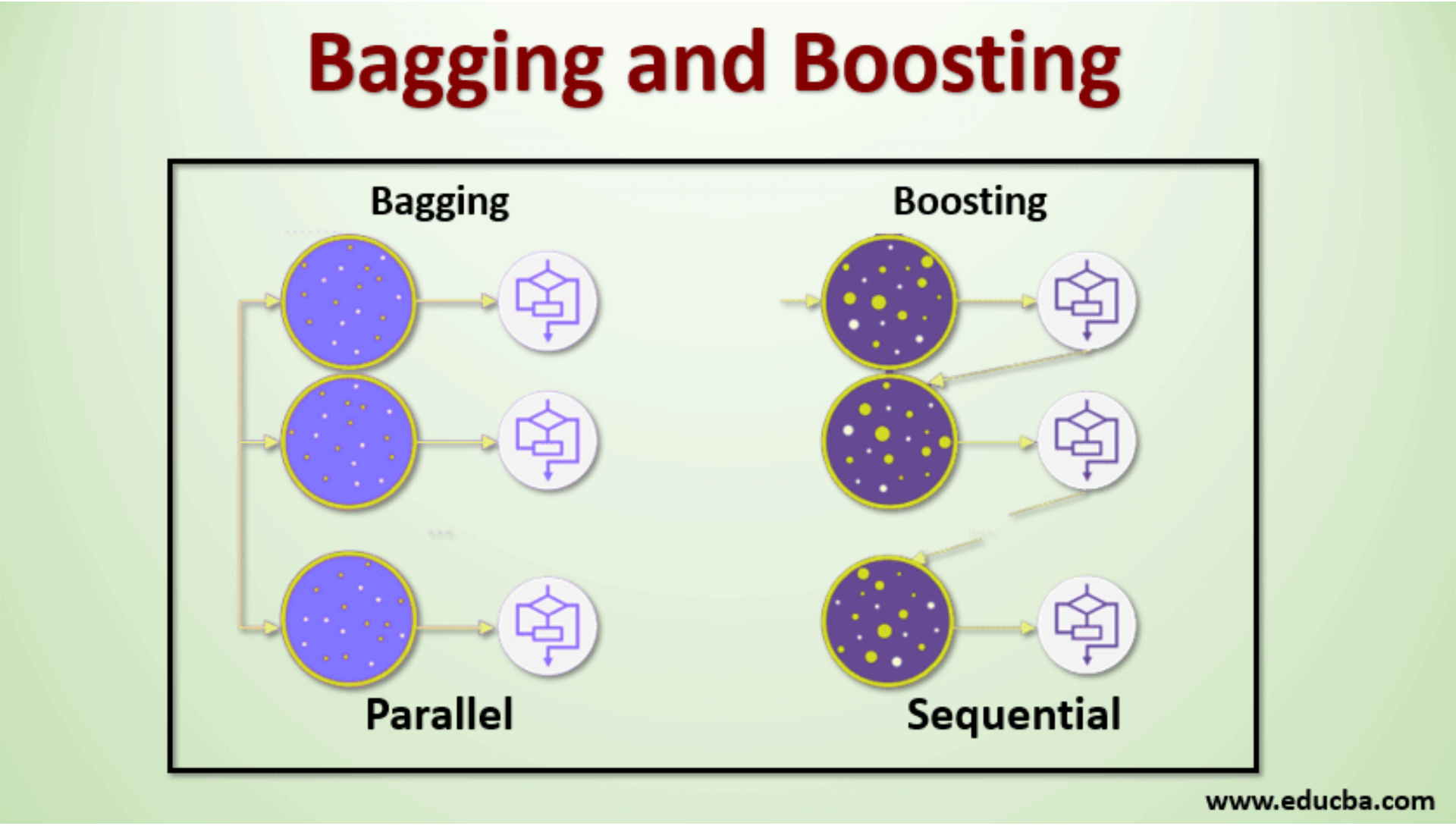
- Relatywnie prosty, łatwy w implementacji i szybki
- Poza liczbą modeli, ew. warunkiem stopu nie wymaga wielu parametrów do strojenia (w odróżnieniu od XGBoost)
- Nie potrzeba dodatkowej wiedzy nt. słabszego uczenia klasyfikatora, może być użyty z wieloma algorytmami
- Może być uogólniany – później

Cons

- Faktyczna poprawa predykcji zależy od danych oraz właściwości algorytmu uczącego podstawowe klasyfikatory
- Może się przeuczyć
- Większa trudność skalowalnej implementacji dla większych danych (Big data) niż bagging lub RF

Boosting vs. Bagging

- Bagging
  - modyfikacja danych poprzez losowanie przykładów,
  - klasyfikatory uczone niezależnie (możliwość zrównoleglenie obliczeń)
  - Redukcja wariancji błędu
- Boosting
  - sekwencyjne rozbudowywanie zespołu – kolejny składnik zależny od działania wcześniejszych
  - modyfikacja poprzez zmianę rozkładu wag przypisywanych przykładom
  - Może redukować wariancje, lecz także bias



Różnice w przetwarzaniu danych

Boosting vs. Bagging with C4.5 [Quinlan 96]

	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
average	*15.66*	*14.11*		*.905*	*13.36*		*.847*		*.930*

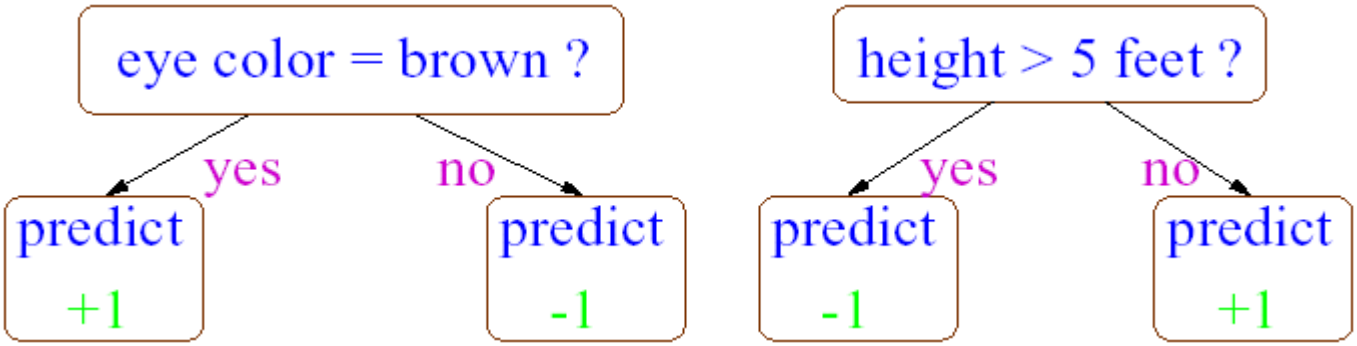
Table 1: Comparison of C4.5 and its bagged and boosted versions.

Boosting vs. Bagging

- Bagging nie działa ze stabilnymi algorytmami, boosting może działać
- Boosting podatny na przeuczenie dla zaszumianych i trudnych danych. Bagging mniej, zwłaszcza odporny jest RF
- Boosting na ogół może prowadzić do wyższych przyrostów trafności, lecz może też prowadzić do pogorszenia na niektórych danych
- Bagging na ogół zawsze polepsza trafność chodź średnio mniej
- Bagging potencjalnie łatwiejszy do uogólnienia

Boosting z drzewami

- Klasyfikatory o zbyt małym bias (np. k-NN) słabo wzmacniane [Rayens], często stosowany z drzewami
- W odróżnieniu od Random Forest (gdzie pozwala się budować głębsze, mniej zredukowane, drzewa) w koncepcji wzmacniania boosting na ogół wykorzystuje się płytsze drzewa (nawet tzw. decision stumps)
- Zbyt duża liczba drzew może prowadzić do przeuczenia dla niedoskonałych danych
- Dla mniejszych drzewa – zwłaszcza dla function gradient boosting – głębokość i np. min. loss dla warunków podziału są globalnymi parametrami wymagającymi specjalnej optymalizacji



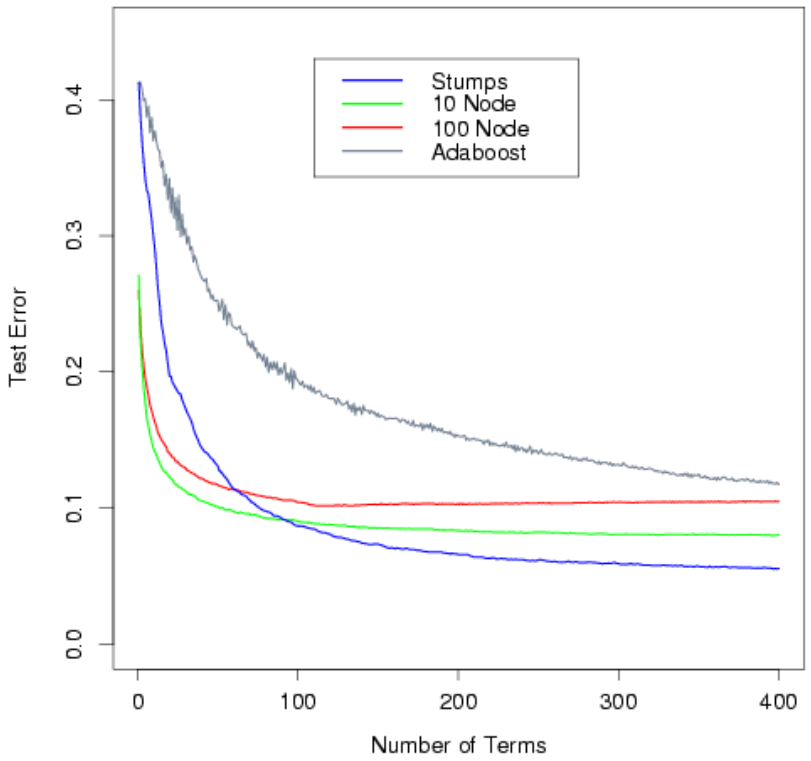


Figure 10.9: *Boosting with different sized trees, applied to the example (10.2) used in Figure 10.2. Since the generative model is additive, stumps perform the best. The boosting algorithm used the binomial deviance loss in Algorithm 10.3; shown for comparison is the AdaBoost algorithm 10.1.*

Rysunek za książka EST

DECORATE (Melville & Mooney, 2003)

- Przykład uogólnienia boostingu poprzez wprowadzenie sztucznych przykładów do zbioru D_t
- Zwiększa zróżnicowanie prób i klasyfikacji
- Skuteczne dla przetwarzanie mniejszych zbiorów danych, gdzie re-weighting i re-sampling ma mniejszy potencjał dywersyfikacji danych uczących

Wzmacnienie – boosting - ogólniej

- Ogólna metoda służąca polepszeniu predykcji dowolnych klasyfikatorów uczonych dowolnymi algorytmami.
- Pierwsze inspiracje L.Valiant,M. Kearns; Rozwinięte przez Yoav Freund i Robert Schapire
- Pierwsze zastosowania (drzewa, ANN): OCR, rozpoznanie obrazów
- Liczne realizacji pomysłu wzmacniania słabych klasyfikatorów, nie tylko Adaboost

Rozwój pomysłu wzmacniania klasyfikatorów

- Uogólnienia Adaboost, np.
  - Wprowadzenie uczenia z kosztami (MetaCost)
  - Specjalizowane przelosowanie obecności przykładów dla danych niezbalansowanych SmoteBoost
  - Cascade classifiers, np. dla rozpoznawania obrazów
  - W organiczonym stopniu w uczeniu przyrostowych (Learn++)
  - …
- Wzmacnianie gradientowe − ang. gradient boosting (obecnie b. efektywne biblioteki, np. XGBoost, CatBoost i LightGBM)

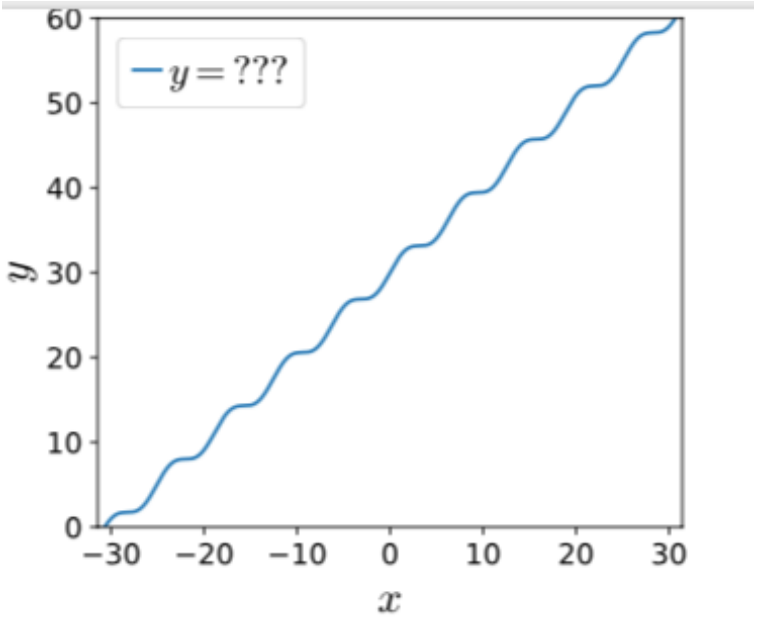
Wzmacnianie gradientowe - krótko

Gradient boosting

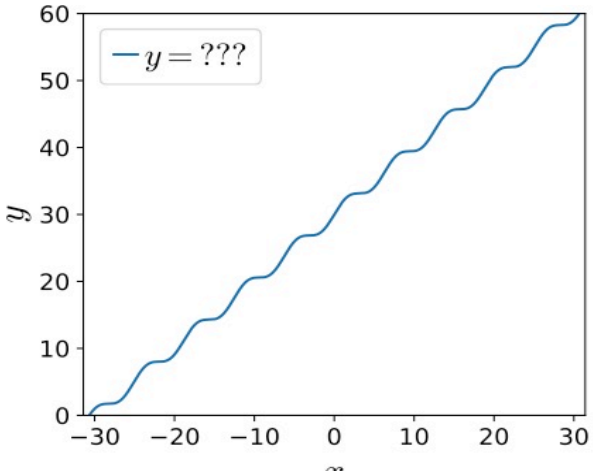
- Zasada dopasowywania **addytywnego modelu** do danych krok po kroku
- W każdym kroku wprowadza się kolejny “słaby” model w celu poradzenia sobie z błędami poprzedników
- W gradient boosting − są one identyfikowane poprzez **ujemne gradienty** wybranej funkcji straty
- W Adaboost − stosuje się wagowanie błędnie sklasyfikowanych przykładów
- W obu wersjach wskazują one kierunki zmiany

Modele addytywne

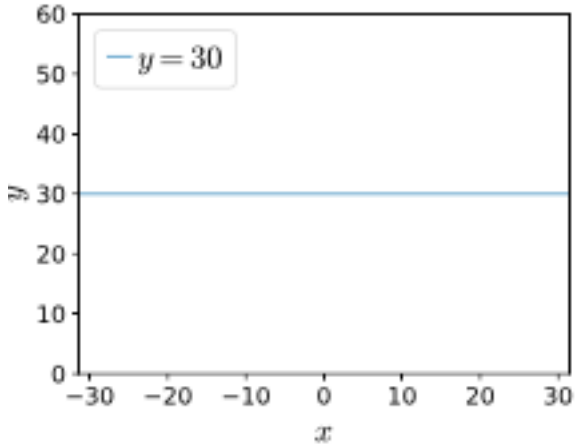
- Połączenie prostszych składników, po to aby zamodelować dopasowanie do złożonej sytuacji $F(x)=f_1(x)+f_2(x)+f_3(x)+\ldots$
- Przykład ilustracyjny − poszukiwanie funkcji regresji dopasowanej do złożonej krzywej *y* vs. *x*

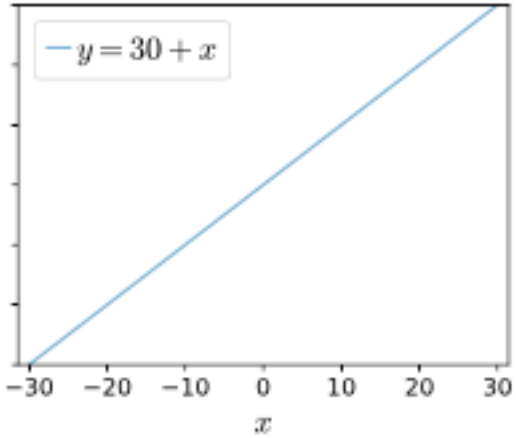


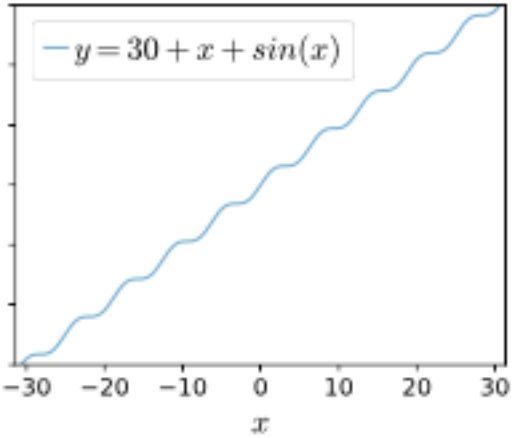
Modele adytywne 2



$$\hat{y} = \sum_{m=1}^{M} f_m(x)$$

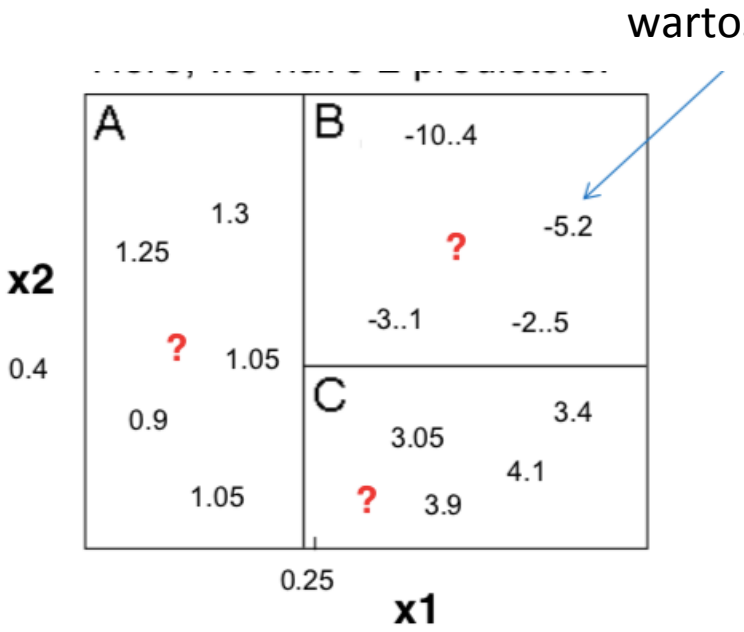


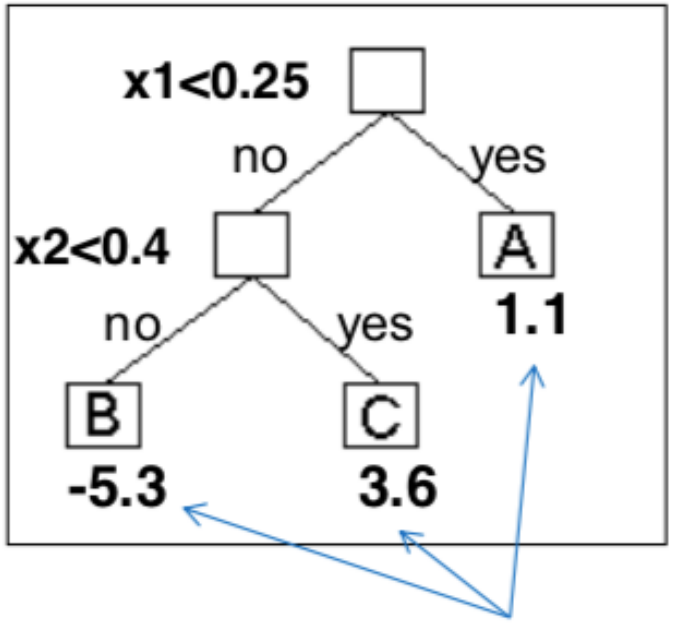




Zbudowano zespół *F*(x)=30+x+sin(x)

Przykład z drzewami regresji

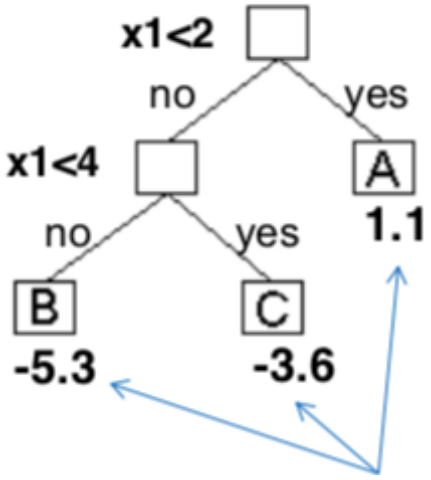




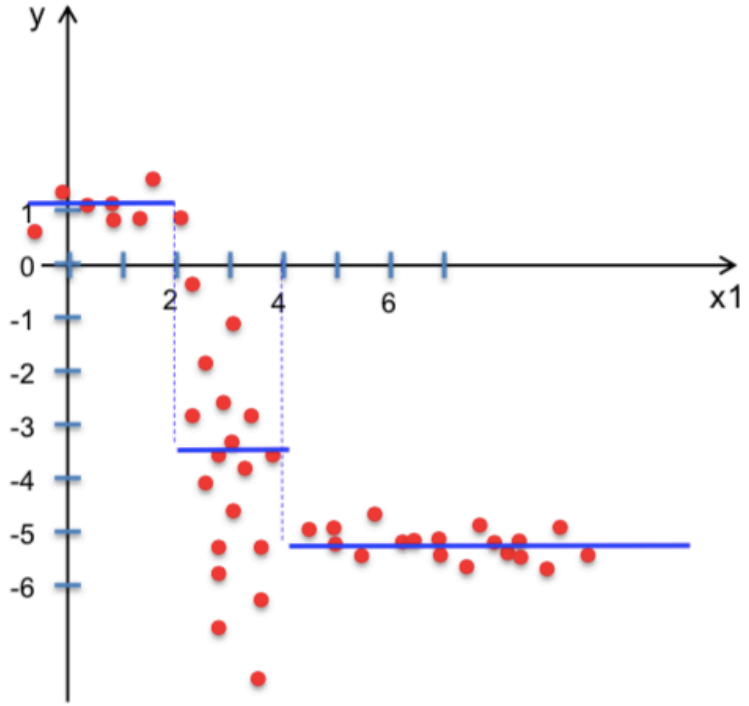
średnie wartości y
w obszarach A,B,C

Cel: minimalizacja błędu średniokwadratowego MSE

Drzewo regresji dla jednej zmiennej x1

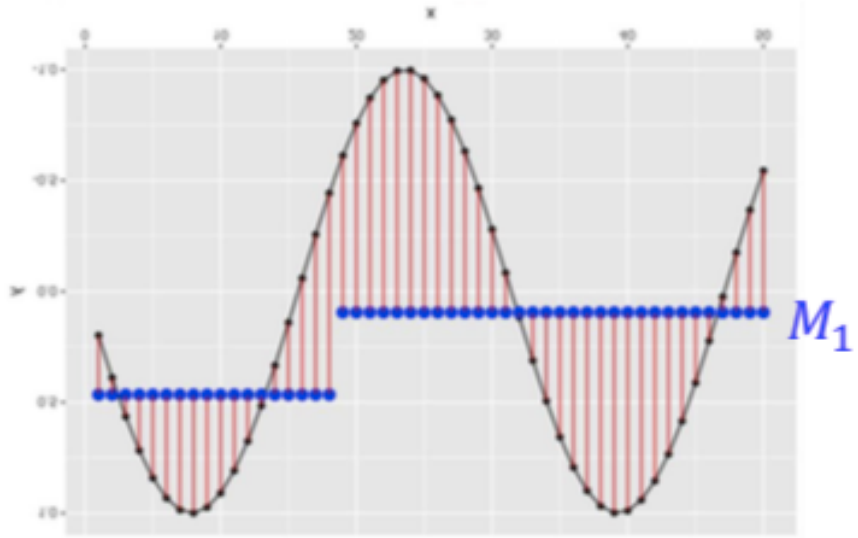


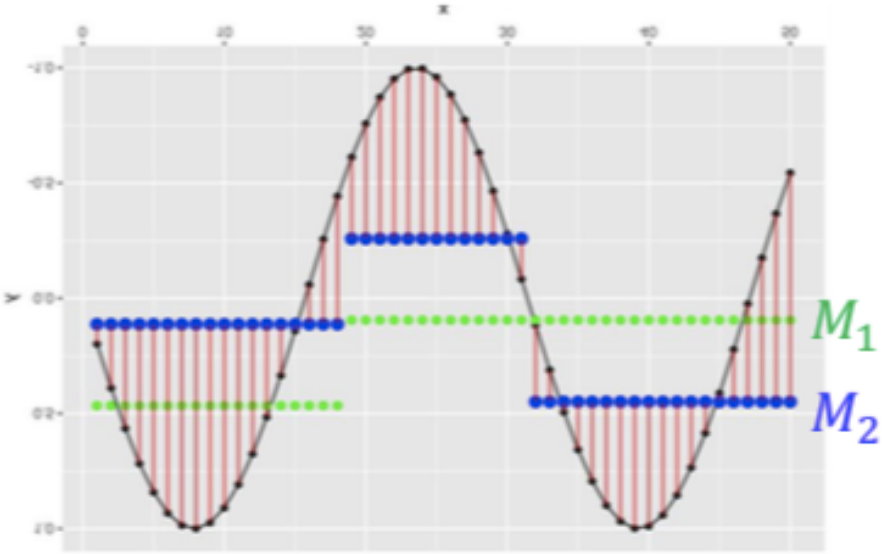
Kolejne podziały x1
średnie wartości y
w obszarach A,B,C



Rozważmy b. złożoną funkcję wymagającą modelu addytywnego F(x)

Funkcja sin(x)

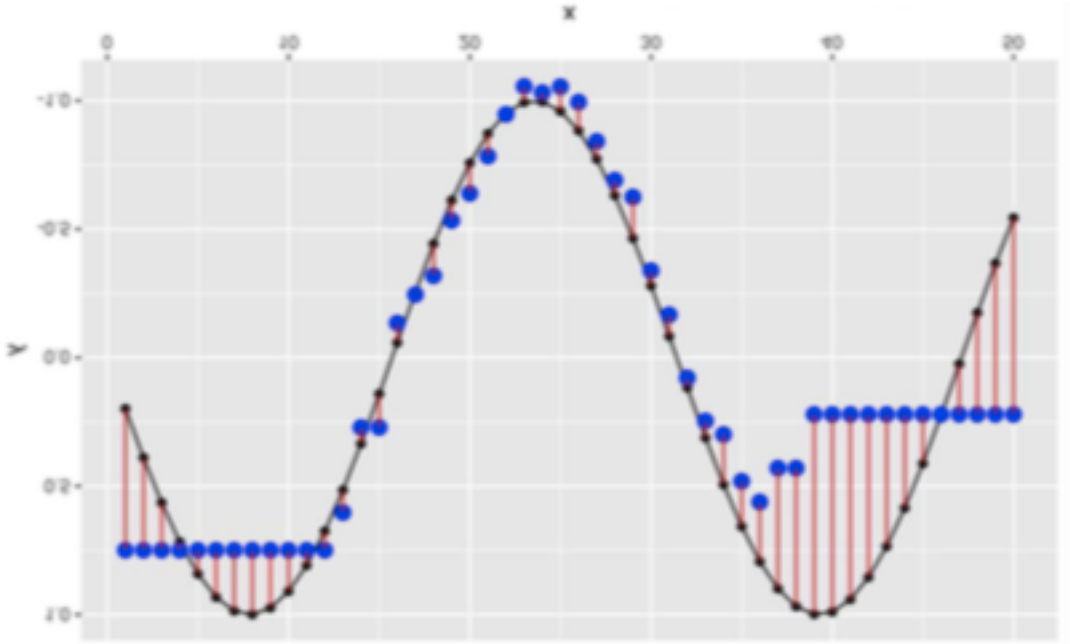




1. Zbuduj płytkie drzewo regresji T1 – pierwszy model dopasowany do danych M_1=T1. Niedopasowanie T1 do danych opisują residua funkcji $r_i=y_i-\hat{y}_i$
2. Zbuduj kolejne drzewo T2 na danych z wyjściami – residua r_i. Model zostaje rozszerzony $M_2=M_1+\theta T2$ gdzie θ jest optymalizowany, w celu lepszego dopasowania do danych. Oblicz ponownie residua dla M2
3. Buduj kolejne drzew dla dopasowania się do residuów z 2, I postępują tak do warunku stopu

Boosted regression trees – model końcowy

Model końcowy jest ważonym uśrednieniem krokowo tworzonych modeli T ; $M=T_1+\eta\Sigma\theta_i T_i$ gdzie η jest prędkością uczenia (zapobieganie przeuczeniu)



Dokładniej dla regresji

Ogólny schemat (dla regresji z MSE)

Dane uczące (x_1,y_1), (x_2,y_2),…,(x_n,y_n) => cel znaleźć model F(x) który minimalizuje błąd średniokwadratowy

Rozpoczynamy od pierwszego prostego modelu $F_1(x)$, który dla kolejnych xi popełnia błędy niedopasowania (residua F), np. $F(x_1)=0.8$ gdy $y_1=0.9$; $F(x_2)=1.4$ gdy $y_2=1.3$

Poszukujemy nowego modelu regresji h, który może być dodany do F, tak aby osiągnąć poprawę:

$$F(x_1)+h(x_1) = y_1$$

$$F(x_2)+h(x_2) = y_2$$

….

$$F(x_n)+h(x_n) = y_n$$

Gradient boosting dla regresji

Alternatywnie poszukujemy nowego modelu regresji h np. drzewa regresji, który powinien

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\ldots.$$
$$h(x_n) = y_n - F(x_n)$$

gdzie y − F(x) to residua r (wskazujące gdzie dotychczasowy model M źle działa). Utwórz nowy zbiór uczący x z wyjściem r (x, y-F(x)) i naucz model h1 = F1

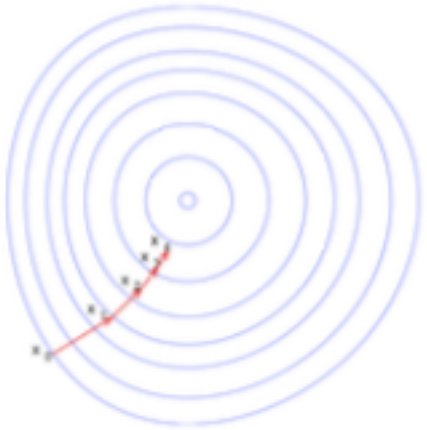
Model addytywny F(x)+h1(x) może nadal nie dopasowywać się dość dobrze do danych, powtórz postępowanie w kolejnych iteracjach

Ogólny schemat boostingu - kolejny model poprawia ograniczenia wcześniejszych

Spadek gradientu

Podejście spadku gradientu - uniwersalna metoda minimalizacja funkcji poprzez przesuwanie się w kierunku przeciwnym do gradientu:

$$\theta_{i+1} = \theta_i - \delta \frac{\partial J}{\partial \theta_i}$$



Gradient boosting dla regresji

Jak podejście wzmacniania regresji ma się do spadku gradientu?

W regresji funkcja straty $L(y,F(x))=(y-F(x))^2/2$

Celem jest minimalizacja sumy kwadratów w danych

$$J = \sum_i L(y_i, F(x_i))$$

W przypadku regresji

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

Czyli residua są ujemnymi gradientami

$$r_i = y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Gradient boosting dla regresji

Residua f. regresji ⇔ ujemny gradient f. straty

Dopasowanie h do residuów ⇔ dopasowanie h do ujemnych gradientów f. straty

Rozbudowana modeli F wg. residuów ⇔ krokowa rozbudowane wg. ujemnych gradientów

Prowadzi to do ogólnego sformułowanie podejścia funkcyjnego gradientowego wzmacnianie (gradient boosted ensemble)

Przebieg wzmacniania gradientowego dla regresji

Zaincjuj pierwszy model $F(x)=\Sigma_i y_i/n$ oraz j = 1

Postępuj do warunku stopu

1. Oblicz ujemne gradienty funkcji straty *L* i utwórz nowy zbiór uczący D_j
2. Naucz model F_j z D_j (dopasowujący się do ujemnych gradientów) $F=F_1+\eta\Sigma\theta_j F_j$
3. Optymalizuj θ_j w celu dobrego dopasowania do danych
4. j = j +1

Ogólny schemat − może być użyty także dla innych postaci funkcji straty

Wersja dla drzew regresji [za JM]

Gradient Tree Boosting Algorithm

1. Initialize $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$

Źródło – wykład IPI PAN dla doktorantów

Inne funkcje straty *L*

Interpretacja statystyczna [FHT – EST 2000]

Odpowiedź złożonego klasyfikatora jako model addytywny

$$F(x) = \sum_{i=1}^{T} \theta_i f_i(x)$$

Adaboost jako metoda krokowego poszukiwania minimum funkcji straty

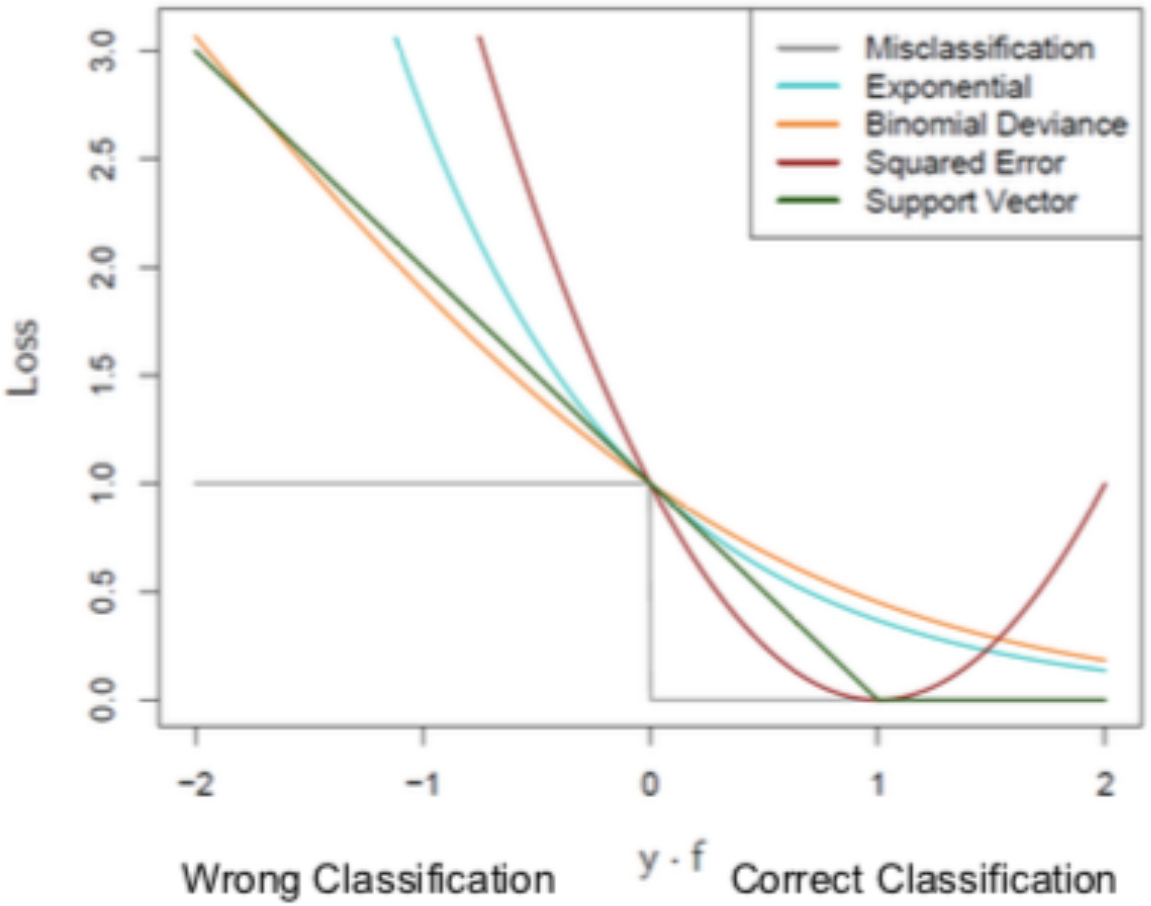
$$\frac{1}{n}\sum_{j=1}^{n} L(y_i, f(x_i)) \quad gdzie \quad L(y, f) = e^{-yf}$$

FHT pokazali, że taka funkcja ma własność, że powyższa optymalizacja prowadzi do przybliżenia klasyfikatora Bayesowskiego

Istnieją alternatywne ciągłe funkcje straty – co prowadzi do tzw.

funkcyjnego wzmacniania gradientowego i szerszej klasy metod

Inne funkcje straty



Misclassification

$$L(y,F) = I[y \neq sign(F)]$$

Exponential / AdaBoost

$$L(y,F) = \exp(-yF)]$$

Binomial Deviance

$$L(y,F) = \log(1+\exp(-2yF)$$

Quadratic / L2-Boost

$$L(y,F) = (y-F)^2$$

SVM

$$L(y,\mathrm{F}) = y \cdot (1 - y \cdot F)$$

Inne wersje zadania

- Rozwiązanie wzmacniania gradientowego przekształca się dla wersji klasyfikacyjnej oraz uczenia się rankingów

- Bardzo ciekawy przykład rozpoznawania liter dostępny w Cheng Li: A gentle introduction to gradient boosting

Wersja klasyfikacyjna za wykład IPI PAN

Szukamy funkcji klasyfikacyjnej minimalizujacej ryzyko empiryczne:
$\mathrm{argmin}_f\, n^{-1} \sum_{i=1}^{n} L(Y_i, f(X_i))$

- Inicjalizacja: $\hat{f}^{[0]}(\cdot) \equiv \mathrm{argmin}_c n^{-1} \sum_{i=1}^{n} L(Y_i, c)$.
  Dla $m = 1, \ldots, m_{stop}$:
- (i) Oblicz

$$U_i = -\frac{\partial}{\partial f} L(Y_i, f)_{|f=\hat{f}^{[m-1]}(X_i)} \quad i = 1, 2, \ldots, n.$$

(ii) Zastosuj wybraną metodę oszacowania funkcji regresji do próby (X_i, U_i):

$$(X_i, U_i) \longrightarrow \hat{g}^{[m]}(\cdot)$$

(szacowanie gradientu).
(iii) $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \nu \times \hat{g}^{[m]}(\cdot)$.
Albo $\nu_m = \mathrm{argmin}_\nu L(f^{[m-1]}(\cdot) + \nu \times \hat{g}^{[m]}(\cdot))$

Uwagi o Funkcyjnego Wzmacniania Gradientowego

Wzmacnianie gradientowe (gradient boosting)

- Inspiracja Breiman (1999) o iteracyjnej minimalizacji funkcji straty w boosting
- Friedman,Hastie, Tibushirani (2000) addytywne modele w Adaboost oraz uogólnienia na różne funkcje straty

Extreme Gradient Boosting

T.Chen (2014/2016) = dodatkowo regularyzacja w celu monitorowania przeuczenia / wymaga specjalnej optymalizacji parametrów

Implementacja XGBoost (T.Chen w DMLC)

Późniejsza implementacja LightGBM

Ekstremalne Wzmacnianie Gradientowe

Extreme Gradient Boosting - EXBoost

Tiangi Chen (2016) wprowadzenie składnika regularyzacji do funkcji straty -> minimalizacja liczby modeli i monitorowanie przeuczenia − lecz trudniejsza do obliczenia

Ponadto wymaga specjalnej optymalizacji parametrów (zwłaszcza dla drzew)

- Opis różnic do GB np. w https://towardsdatascience.com/boosting-algorithm-xgboost-4d9ec0207d

Efektywna implementacja biblioteka XGBoost

Z powodzeniem zastosowana w wielu konkursach (np. patrz platforma Kaggle)

Obecnie bardzo popularny

Odnośniki do literatury

- Intensywny rozwój od lat 90 poprzedniego wieku
- Wiele różnych propozycji:
  - R.Polikar,Ensemble based systems in decision making, IEEE Circuits and Systems Magazine, vol. 6, no. 3, pp. 21–45, 2006.
  - Schapire, Robert E. (1990). The Strength of Weak Learnability . Machine Learning. 5 (2): 197–227.,
  - Yoav Freund and Robert E. Schapire (1997); A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, Journal of Computer and System Sciences, 55(1):119-139
  - Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36(1/2):105–139, 1999.
  - Leo Breiman (1998). "Arcing classifier (with discussion and a rejoinder by the author)". Ann. Stat. 26 (3): 801–84
  - Tianqi Chen i Carlos Guestrin, 2016. XGBoost: A Scalable Tree Boosting System ,https://arxiv.org/pdf/1603.02754.pdf

Pytanie i komentarze?

Dalszy kontakt:

jerzy.stefanowski@cs.put.poznan.pl

http://www.cs.put.poznan.pl/jstefanowski/

Fundusze Europejskie Polska Cyfrowa

Rzeczpospolita Polska

Unia Europejska Europejski Fundusz Rozwoju Regionalnego