





Poprawa jakości i stabilizacja słabych klasyfikatorów

Bagging
Boosting



Przesłanki

★ Dla wielu praktycznych przypadków otrzymywane reguły decyzyjne są obarczone dużym błędem. Najczęściej spowodowane jest to małą ilością obiektów uczących (jak to ma miejsce np. w diagnostyce medycznej, gdzie dla wielu problemów decyzyjnych nie jesteśmy w stanie zebrać dostatecznie dużego zbioru uczącego). O tym, że jest to problem istotny świadczą także dedykowane małym zbiorom uczącym metody testowania jakości klasyfikatorów, jak choćby metoda *leave-one-out*.



Wstęp



- ★ Przedstawmy metody poprawiania jakości i stabilizacji tzw. słabych klasyfikatorów (ang. *weak classifiers*), tzn. takich których jakość jest niewiele lepsza od jakości decyzji losowej oraz, dla których niewielka zmiana w materiale uczącym może powodować otrzymanie diametralnie różnych jakościowo klasyfikatorów.



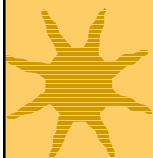
Wstęp



- ★ Wszystkie metody bazują na koncepcji wielokrotnej modyfikacji pierwotnego zbioru uczącego i dla każdej z nich stworzeniu nowego klasyfikatora. Do podjęcia decyzji wykorzystuje się decyzję komitetu klasyfikatorów otrzymanych w kolejnych iteracjach.



Wstęp



- ★ Takie grupowanie klasyfikatorów stabilizuje ich działanie oraz w wielu wypadkach wydatnie poprawia jakość ich klasyfikacji. Spośród metod tworzenia takich klasyfikatorów złożonych najpopularniejszymi koncepcjami są *bagging* i *boosting*, choć i wcześniejsze prace dotyczące tzw. *bootstrappingu* jak najbardziej wpisują się w ten nurt.



Wstęp



- ★ Należy tu jednak wspomnieć, że metoda *bootstrappingu*, będąca uogólnieniem metody *jackknife*, polega na wielokrotnym generowaniu obiektów z rozkładem estymowanym próbą pierwotną. Otrzymane nowe zbiory mogą służyć np. do estymacji błędu rzeczywistego klasyfikatora otrzymanego na podstawie zbioru pierwotnego. Zauważmy także, że mogą być one podstawą do tworzenia nowych zbiorów uczących i tym samym nowych klasyfikatorów. Dla takiego komitetu klasyfikatorów możemy zaproponować, którychś z opisywanych wcześniej, schematów otrzymywania decyzji grupowej.



Wstęp



- ★ Wadą powyższej metody jest jednak jej dość duża złożoność obliczeniowa, dlatego większą popularności zyskały metody mniej złożone, wspomniane wcześniej *boosting* i *bagging*, a co najważniejsze dające zbliżone rezultaty.



Bagging



- ★ *Bagging*, opracowany przez Breimana w 1996 roku, jest algorytmem, którego celem jest uzyskanie, na podstawie słabego klasyfikatora stabilnej hipotezy o lepszej jakości rozpoznawania. Uzyskuje się ją także poprzez wyuczenie grupy klasyfikatorów na modyfikowanych zbiorach uczących wygenerowanych na podstawie losowania ze zwracaniem elementów ze zbioru pierwotnego (prawdopodobieństwo wylosowania każdego elementu jest takie same). W rezultacie, niektóre przykłady wcale nie występują w ciągu uczącym pojedynczy klasyfikator, natomiast niektóre przykłady występują kilkakrotnie. Zbiory te mają taką samą licznosc jak zbiór pierwotny.



Bagging



Oryginalny zbiór uczący:	1, 2, 3, 4, 5, 6, 7, 8
zbiór uczący dla klasyfikatora 1	2, 7, 8, 3, 7, 6, 3, 1
zbiór uczący dla klasyfikatora 2	7, 8, 5, 6, 4, 2, 7, 1
zbiór uczący dla klasyfikatora 3	3, 6, 2, 7, 5, 6, 2, 2
zbiór uczący dla klasyfikatora 4	4, 5, 1, 4, 6, 4, 3, 8



Bagging



★ W poniższej metodzie wspólna decyzja klasyfikatorów (otrzymanych na podstawie niezależnych zbiorów uczących) podejmowana jest poprzez zwykłe głosowanie większościowe. Zwróćmy także uwagę na fakt, że uczenie poszczególnych klasyfikatorów składowych jest niezależne (nie zależy od kroku poprzedniego), zatem zadanie *baggingu* może zostać zdekomponowane i może być wykonywane współbieżnie przez niezależne realizatory, co w przypadku zastosowań praktycznych może mieć duże znaczenie.



Bagging - pseudokod



Wejście: Zbiór uczący LS
 T ilość iteracji

Powtarzaj od $t:=1$ do T



Wylosuj zbiór uczący $LS(t)$ na podstawie losowania z powtórzeniami elementów ze zbioru LS



Naucz, na podstawie $LS(t)$, klasyfikator Ψ_t



Bagging - podjęcie decyzji przez komitet



$$\bar{\Psi}(x) = \arg \max_j \sum_{t=1}^T \delta(\Psi_t(x), j)$$



$$j \in \{1, \dots, M\}$$



$$\delta(i, j) \text{ delta Kroneckera}$$



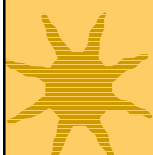
Boosting



- ★ *Boosting* jest także metodą mającą na celu zwiększenie skuteczności działania dowolnego algorytmu uczącego. Wywodzi się ona z teoretycznych prac nad modelem uczenia PAC (ang. *Probably Approximately Correct*) którego idea polega na określeniu tzw. warunków PAC-nauczalności, czyli warunków przy których uczeń znajduje dobry klasyfikator dużym prawdopodobieństwem. Jest to niejako przeformułowane zadanie testowania hipotez statystycznych, które dla określonej próbki określa, czy prawdziwa jest hipoteza, czy też konthipoteza. Odpowiedź zawsze udzielana była na pewnym poziomie ufności (czyli z pewnym prawdopodobieństwem). W teorii PAC pytanie sformułowane jest odwrotnie, mianowicie jakie warunki musi spełniać próbka aby otrzymać hipotezę na określonym poziomie ufności.



Boosting



- ★ Jedną z pierwszych odpowiedzi na pytanie, czy „słaby” klasyfikator może zostać wzmocniony. Udowodniono, że wynikiem takiego wzmocnienia powinien być „silny” klasyfikator.
- ★ W 1995 roku zaprezentowano algorytm AdaBoost, który pozbawiony został wad wcześniejszych algorytmów *boostingowych*. Przedrostek „Ada” jest skrótem od adaptacyjny, ponieważ algorytm ten adaptuje się do błędów swoich klasyfikatorów składowych, czyli dopasowuje swoje działanie tak, aby te błędy minimalizować.



Boosting



- ★ Działanie algorytmu AdaBoost rozpoczyna się od nadania każdemu z otrzymanych obiektów uczących jednakowych wag tak, aby suma wszystkich wag wynosiła 1. Każda waga obrazuje stopień trudności w poprawnym klasyfikowaniu danego przykładu przez klasyfikatory powstające w kolejnych rundach. Główna pętla algorytmu ma na celu utworzenie tylu klasyfikatorów składowych, ile jest przewidzianych rund *boostingu* i dla każdego z tych klasyfikatorów określenie wagi danego klasyfikatora w końcowym głosowaniu. Aby to osiągnąć, każda runda rozpoczyna się od określenia zbioru uczącego na podstawie wag poszczególnych przykładów, po czym, przy użyciu tego zbioru tworzony jest klasyfikator.



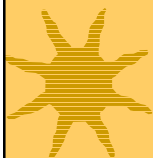
Boosting



- ★ W praktyce są tu dwie możliwości postępowania:
 - jeżeli klasyfikator jest algorytmem, który potrafi bezpośrednio skorzystać z wag poszczególnych przykładów, wówczas można to wykorzystać, przekazując mu ciąg uczący wraz z wagami poszczególnych przykładów, co określane jest jako „*boosting* przez ważenie” (ang. *boosting by reweighting*).
 - w przeciwnym wypadku, jeżeli algorytm klasyfikatora nie potrafi skorzystać bezpośrednio z wag przykładów, wówczas należy na podstawie wartości poszczególnych wag utworzyć zbiór uczący i taki ciąg przekazać do klasyfikatora, co określane jest jako „*boosting* przez próbkowanie” (ang. *boosting by resampling*).



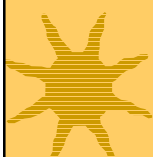
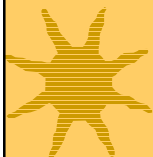
Boosting



Oryginalny zbiór uczący:	1, 2, 3, 4, 5, 6, 7, 8
zbiór uczący po pierwszej iteracji	2, 7, 8, 3, 7, 6, 3, 1
zbiór uczący po drugiej iteracji	1, 4, 5, 4, 1, 5, 6, 4
zbiór uczący po trzeciej iteracji	7, 1, 5, 8, 1, 8, 1, 4
zbiór uczący po czwartej iteracji	1, 1, 6, 1, 1, 3, 1, 5



Boosting



- * Kolejnym krokiem jest określenie błędu klasyfikatora na tym samym zbiorze, na którym klasyfikator był uczony. Na tej podstawie obliczany jest współczynnik, który charakteryzuje „ważność” klasyfikatora danej rundy. Współczynnik ten jest tym większy, im większa część przykładów uczących została poprawnie sklasyfikowana. Następną operacją wykonywaną w danej rundzie jest modyfikacja wag przykładów z ciągu uczącego, w wyniku czego następuje zwiększenie wag przykładów błędnie klasyfikowanych i zmniejszenie wag przykładów klasyfikowanych poprawnie. Dzięki temu, w czasie uczenia klasyfikatora *boostingowego*, w kolejnych rundach następuje koncentrowanie się klasyfikatorów składowych na trudniejszych przykładach, które zwiększają swój udział w ciągu uczącym. Na końcu każdej rundy następuje normalizacja wag. Ostateczna decyzja klasyfikatora *boostingowego* wyznaczana jest za pomocą głosowania ważonego T klasyfikatorów składowych.



Boosting

Wejście: Zbiór uczący LS o liczności n
 T ilość iteracji

1. Powtarzaj od $i:=1$ do n
 $D_1(i) := 1/n$, gdzie $D_1(i)$ jest prawdopodobieństwem wylosowania i tego elementu ze zbioru LS
2. Powtarzaj od $t:=1$ do T
 - a. Naucz klasyfikator Ψ_t zbiorem LS zgodnym z rozkładem D_t ,
 - b. $\varepsilon_t := \sum_{\Psi_t(x_i) \neq y_i} D_t(i)$
 - c. jeżeli $\varepsilon_t > 1/2$
wtedy
 - i) $T := t-1$
 - ii) Wyjdź z pętli
 - d. $\beta_t := \frac{\varepsilon_t}{1-\varepsilon_t}$
 - e. $Z := 0$
 - f. Powtarzaj od $i:=1$ do n
 - i. jeżeli $\Psi_t(x_i) = j_t$
wtedy
 $D_{t+1}(i) := D_t(i) \times \beta_t$
w przeciwnym wypadku
 $D_{t+1}(i) := D_t(i)$
 - ii. $Z := Z + D_{t+1}(i)$
 - g. Powtarzaj od $i:=1$ do n
 $D_{t+1}(i) := \frac{D_{t+1}(i)}{Z}$

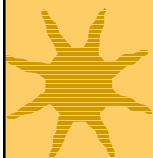


Boosting - decyzja

$$\bar{\Psi}(x) = \arg \max_j \sum_{t=1}^T \delta(\Psi_t(x), j) \log\left(\frac{1}{\beta_t}\right)$$



Boosting



- ★ AdaBoost.M1, ma jednak tę zasadniczą wadę, że nie radzi sobie z przypadkami, gdy słaby klasyfikator ma błąd próbki większy niż $1/2$. Ściślej, jeżeli w określonej rundzie t , błąd klasyfikatora na ciągu testującym wyniesie powyżej $1/2$, wówczas główna pętla algorytmu *boostingu* jest przerywana, a utworzony klasyfikator *boostingowy* składa się z $t-1$ klasyfikatorów składowych. Zasadniczy problem pojawia się jeśli taka sytuacja wystąpi już w pierwszej rundzie - wówczas tak stworzony klasyfikator *boostingowy* nie będzie w stanie przedstawić żadnej odpowiedzi, ponieważ nie będzie zawierał żadnego klasyfikatora składowego.



Boosting



- ★ Ograniczenie błędu próbki klasyfikatora składowego na poziomie $1/2$ jest tym trudniejsze do spełnienia, im większa jest ilość klas do jakich dany przykład może należeć. W przypadku, gdy przykłady z danej dziedziny mogą należeć do M -klas i klasyfikator ma skuteczność porównywalną z losowym zgadywaniem, wówczas prawdopodobieństwo błędu wyniesie. Jak widać, już w przypadku pięciu klas, oczekiwany błąd takiego klasyfikatora wyniósłby $0,8$. Zauważyli to także twórcy AdaBoost i uwzględnili to w kolejnej modyfikacji AdaBoost.M2. Modyfikacja ta opiera się na innym oszacowaniu błędu popełnianym przez klasyfikator w kolejnych iteracjach poprzez wprowadzenie tzw. pseudostraty, którą można interpretować jako funkcje strat. Ciekawym jest to, że funkcja ta jest modyfikowana w kolejnych iteracjach zgodnie z błędem popełnianym przez klasyfikator.



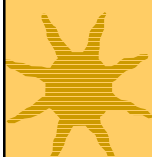
Boosting



- ★ Kolejnym problemem jest ustalenie liczby iteracji algorytmu. Nie ma uniwersalnej odpowiedzi na pytanie ile klasyfikatorów składowych powinno wchodzić w skład grupy. Eksperymenty przeprowadzone przez różnych badaczy pokazują, że ilość ta silnie zależy od klasyfikatora, który podlega poprawie. Zwykle największy efekt uzyskany przez zastosowanie klasyfikatorów grupowych występuje w pierwszych kilku-kilkunastu rundach. Dla niektórych klasyfikatorów, np. C4.5 dalsze zwiększanie ilości klasyfikatorów składowych nie przynosi zwykle zasadniczej poprawy. W innych pracach, gdzie jako klasyfikatory otrzymywano na podstawie innych algorytmów uczenia, ustalono liczbę iteracji na 100. W tej samej pracy do rzeczywistego problemu rozpoznawania ręcznie pisanych kodów pocztowych przyjęto 30 iteracji. W innych rzeczywistych problemach decyzyjnych zaobserwowano wręcz, przy zbyt dużej liczbie iteracji, skłonności takich klasyfikatorów do przeuczania.



Boosting vs bagging dla C4.5



	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 dla C4.5

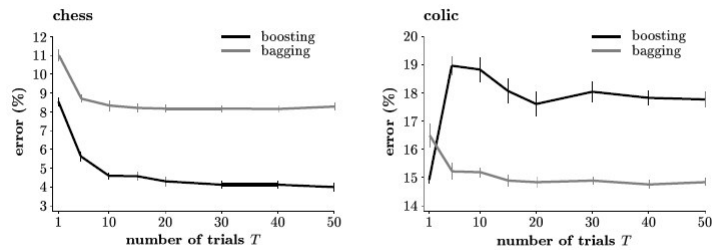


Figure 1: Comparison of bagging and boosting on two datasets



Boosting

- ★ Stabilizacja działania po początkowych iteracjach zaobserwowano także dla opisywanych w poprzednim punkcie metod *baggingu*.
- ★ Reasumując, trzeba podkreślić, że uzyskany rezultat zależy od kilku czynników, między innymi:
 - metody tworzenia klasyfikatora złożonego,
 - użytego klasyfikatora składowego,
 - bazy przykładów, na której się zjawisko bada.



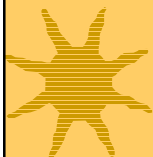
Boosting



- ★ Iteracyjny charakter algorytmów *boostingowych* oraz zależność kroków kolejnych od przednich powoduje, że są one znacznie obciążone obliczeniowo. W odróżnieniu od metod *baggingowych* proces *boostowania* klasyfikatora nie może zostać zdekomponowany i wykonywany przez niezależne realizatory, gdyż do wygenerowania wartości częstości, z jaką mogą zostać wylosowane poszczególne elementy uczące w kolejnej iteracji, wykorzystywane są ich wartości z iteracji poprzedniej.



Boosting



- ★ W literaturze można znaleźć szereg modyfikacji algorytmu *boostingowego*, jak choćby *FloatBoost*, w której proponuje się usunięcie otrzymywanych w kolejnych iteracjach klasyfikatorów, jeżeli ich jakość jest mniejsza niż założona. Inna modyfikacja polega na tym, że klasyfikatory przechodzą poprzez procedurę *boostingu* dla różnych obszarów decyzyjnych, co poprawia ich jakość, niestety silnie wpływa na obciążenie obliczeniowe takiej metody. Istnieje także szereg modyfikacji związanych z przeznaczeniem *boostingu* dla konkretnych klasyfikatorów, czy też łączenie powyżej zaprezentowanych metod ze sobą.