AdaBoost: Mądrość tłumu

*Artur JEŻ**

* Wydział Matematyki i Informatyki, Uniwersytet Wrocławski

W 1906 roku statystyk Francis Galton był świadkiem konkursu, w którym uczestnicy mieli określić wagę mięsa uzyskanego z prezentowanego wołu. Galton przeanalizował 787 oddanych głosów i odkrył, że ich mediana była odległa od prawdziwego wyniku o niecałe 0,8%. Podobne zjawiska, w których wysokiej jakości odpowiedź wypracowana jest na podstawie wielu odpowiedzi słabej jakości, zaobserwowano wielokrotnie i są one określane mianem *mądrości tłumu* (*wisdom of the crowds*); są one różnorako interpretowane z punktu widzenia psychologii, teorii ewolucji, teorii gier... W tym artykule spojrzymy na nie z formalnego punktu widzenia i przełożymy na praktykę uczenia maszynowego.

Swoje wyniki Galton opublikował w artykule „Vox Populi", *Nature*, 7 marca 1907 roku. *Średnia* z oddanych głosów była dokładną wagą. Wydaje się, że Galton wybrał medianę ze względu na prostsze rachunki. Więcej o analizie Galtona: „Revisiting Francis Galton's forecasting competition", *Statistical Science* 29(3), 420–424, 2014.

Uczenie

Sformalizujmy nasz problem: *Uczeń* to algorytm, który stara się skonstruować pewną ustaloną funkcję $f : \mathcal{X} \to \{-1, 1\}$ na podstawie próbek $(x_1, f(x_1)), \ldots, (x_t, f(x_t))$, gdzie $x_1, \ldots, x_t$ są wybrane losowo według pewnego rozkładu prawdopodobieństwa. Oznaczmy przez $H[p, t]$ funkcję zwróconą przez ucznia H na podstawie t próbek, w których $x_1, \ldots, x_t$ są wylosowane niezależnie rozkładem p.

Wybór wartości $\{-1, 1\}$ okaże się przydatny już za chwilę. Często używa się też $\{0, 1\}$ lub {TAK, NIE}. Zauważmy, że przybliża to wiele naturalnych problemów: Sprzedać czy kupić akcje? Będzie padać czy nie? Wybory wygrają demokraci czy republikanie?

Błąd funkcji h zwróconej przez ucznia dla ustalonego rozkładu p definiujemy jako

$$\mathrm{err}_p(h) = \sum_{x \in \mathcal{X} : h(x) \neq f(x)} p(x),$$

czyli frakcję złych odpowiedzi funkcji h ważoną rozkładem p. Rozkład p może odpowiadać prawdziwej częstotliwości występowania elementów z $\mathcal{X}$, ale może być inny – wówczas błąd będzie większy, jeżeli złe odpowiedzi wystąpią przy odpowiedziach, którym przypisaliśmy wyższe prawdopodobieństwa.

Dla przykładu niech $\mathcal{X} = \{1, \ldots, 7\}$, a funkcja f zwraca 1 dla liczb pierwszych i -1 dla pozostałych. Dla $t = 3$ i rozkładu jednostajnego $p(i) = 1/7$ wylosowane zostały liczby $2, 3, 7$. Przykładowy uczeń na podstawie próbek $(2, 1), (3, 1), (7, 1)$ zwrócił funkcję stale równą 1: $h(i) = 1$ dla $i \in \mathcal{X}$. Błąd tej funkcji wynosi $\mathrm{err}_p(h) = 3/7$.

Dobry uczeń, dla każdego rozkładu p, z prawdopodobieństwem dowolnie bliskim 1 zwróci funkcję o bardzo małym błędzie, o ile ma dostęp do wielu próbek, formalnie:

$$\forall \epsilon > 0 \, \forall p \, \forall \delta > 0 \, \exists t_0 \, \forall t > t_0 \ \mathbb{P}[\mathrm{err}_p(H[p, t]) < \epsilon] \geqslant 1 - \delta,$$

przy czym $\mathbb{P}$ jest prawdopodobieństwem liczonym po $x_1, \ldots, x_t$ niezależnie wylosowanych rozkładem p. Uczeń jest słaby, gdy jedyne, co o nim potrafimy powiedzieć, to to, że jego błąd jest mniejszy niż błąd losowej odpowiedzi, tj. $1/2$:

$$\exists \epsilon > 0 \, \forall p \, \forall \delta > 0 \, \exists t_0 \, \forall t > t_0 \ \mathbb{P}[\mathrm{err}_p(H[p, t]) < 1/2 - \epsilon] \geqslant 1 - \delta.$$

Używamy uproszczonej definicji *PAC learning*, w ogólności trzeba jeszcze wziąć pod uwagę efektywną konstruowalność funkcji, zależność t_0 od δ, ograniczyć klasę funkcji, do której f należy. Jednak z punktu widzenia tego artykułu to uproszczenie całkowicie wystarcza.

Zwróćmy uwagę na różną kwantyfikację po ϵ: uczeń dobry jest „dowolnie dobry", uczeń słaby jest „trochę lepszy" od losowego.

Co prawda nie pozwalamy, by zwrócona przez ucznia funkcja losowała, ale przyjęcie za punkt odniesienia błędu $1/2$ jest dość naturalne.

Chcemy wiedzieć, czy jeśli potrafimy coś „trochę", to możemy nauczyć się tego „dobrze"? Formalnie: czy z funkcji zwróconych przez słabego ucznia potrafimy skonstruować odpowiedź ucznia dobrego? Na przykład czy zamiast konstruować skomplikowane i mało efektywne drzewo decyzyjne możemy użyć drzew prostych i szybkich?

Pytanie to postawili oryginalnie M. Kearns, L. Valiant: „Cryptographic Limitations on Learning Boolean Formulae and Finite Automata", *STOC* 1989: 433–444.

Dość szybko pokazano, że odpowiedź jest twierdząca, ale konstrukcje były skomplikowane i nie miały praktycznego zastosowania. Zmienił to algorytm AdaBoost autorstwa Y. Freunda i R. Schapire'a, który miał prosty dowód poprawności i dobrze działał w praktyce. W 2003 roku za AdaBoost przyznano jego autorom nagrodę Gödla – jedną z najważniejszych nagród w informatyce teoretycznej.

Y. Freund, R. Schapire: „Game Theory, On-Line Prediction and Boosting", *COLT* 1996: 325–332.
Nazwa AdaBoost to akronim *Adaptive Boosting*: *boosting* od wzmacniania słabych uczniów, a *adaptive*, bo uczeń dopasowuje się do kolejnych wyników.

Algorytm AdaBoost

Funkcja skonstruowana przez AdaBoost będzie postaci:

$$\mathrm{sgn}\Big(\sum_{i=1}^{m} \alpha_i H[p_i, t]\Big) \tag{1}$$

dla z góry ustalonego t i zdefiniowanych w kolejnych krokach algorytmu współczynników $\alpha_1, \ldots, \alpha_m$ i rozkładów prawdopodobieństwa $p_1, \ldots, p_m$. Rozkład p_1 jest dowolny (zwykle: jednostajny). Z kolei sgn jest funkcją znaku zwracającą -1 dla liczb ujemnych, a 1 dla liczb dodatnich oraz zera (nie chcemy zwracać 0, żeby funkcja zgadzała się ze specyfikacją).

for $i \leftarrow 1$ to m **do**
  $h_i \leftarrow H[p_i, t]$
  $e_i \leftarrow \mathrm{err}_{p_i}(h_i)$
  $\alpha_i \leftarrow \frac{1}{2}\log\left(\frac{1-e_i}{e_i}\right)$
  $Z_i \leftarrow e_i \cdot e^{\alpha_i} + (1-e_i)\cdot e^{-\alpha_i}$
  for $x \in \mathcal{X}$ **do**

$$p_{i+1}(x) \leftarrow \begin{cases} p_i(x)\cdot e^{\alpha_i}/Z_i & \text{jeśli } h_i(x) \neq f(x) \\ p_i(x)/(e^{\alpha_i}\cdot Z_i) & \text{jeśli } h_i(x) = f(x) \end{cases}$$

return $\operatorname{sgn}\left(\sum_{i=1}^m \alpha_i h_i\right)$

Pseudokod algorytmu znajduje się obok. W i-tym kroku AdaBoost oblicza (w praktyce: przybliża) błąd e_i funkcji $h_i = H[p_i, t]$. Następnie na podstawie e_i definiuje wagę α_i oraz tworzy rozkład p_{i+1}: punkty x, dla których h_i daje dobry wynik (tj. $h_i(x) = f(x)$), mają prawdopodobieństwo dzielone przez e^{α_i}, zaś te, dla których wynik jest niepoprawny, mnożone przez e^{α_i}. Tak określone prawdopodobieństwa p_{i+1} mogą nie sumować się do jedynki, dlatego dzielone są przez ich sumę Z_i:

$$Z_i = \sum_{x:h_i(x)\neq f(x)} p_i(x)\cdot e^{\alpha_i} + \sum_{x:h_i(x)=f(x)} p_i(x)\cdot e^{-\alpha_i} = (e_i\cdot e^{\alpha_i} + (1-e_i)\cdot e^{-\alpha_i}).$$

Intuicyjnie, zależy nam tu na tym, aby funkcja h_{i+1} dała dobry wynik dla tych punktów, dla których h_i dała wynik zły. Co okaże się kluczowe w analizie błędu, w wykładniku e^{α_i} pojawia się to samo α_i, które jest współczynnikiem w zwracanej przez algorytm kombinacji liniowej (1).

Dla niektórych wartości nasz pseudokod „nie działa” – jeśli $e_i = 0$, to α_i wychodzi $+\infty$. Ma to jednak intuicyjne wyjaśnienie: skoro $e_i = 0$, to $h_i = f$ i chcemy zwrócić h_i, i w pewnym sensie to się dzieje: skoro $\alpha_i = +\infty$, to znak zwracanej przez AdaBoost sumy jest równy znakowi h_i.

Dla zilustrowania, spróbujemy nauczyć się przy użyciu AdaBoost rozpoznawać liczby parzyste ze zbioru $1, 2, 3, 4$. Mamy więc $\mathcal{X} = \{1, 2, 3, 4\}$, a funkcja $f : \mathcal{X} \to \{-1, 1\}$ jest zdefiniowana tak: $f(i) = 1$ dla $i \in \{2, 4\}$ oraz $f(i) = -1$ dla $i \in \{1, 3\}$. Wykonamy cztery iteracje algorytmu, więc funkcję f skonstruujemy na podstawie czterech odpowiedzi słabego ucznia.

Początkowo wszystkim liczbom z $\mathcal{X}$ przypisujemy prawdopodobieństwo $p_1(i) = 1/4$. Załóżmy, że funkcja h_1 zwrócona przez ucznia dla rozkładu p_1 myli się tylko na 1. Jej błąd to zatem $e_1 = 1/4$, czyli $\alpha_1 = \log(\sqrt{3})$, $e^{\alpha_1} = \sqrt{3}$ i $Z_1 = \sqrt{3}/2$. Definiując p_2, prawdopodobieństwo dla 1 mnożymy więc przez $\sqrt{3}/(\sqrt{3}/2) = 2$, zaś dla $2, 3, 4$ przez $(1/\sqrt{3})/(\sqrt{3}/2) = 2/3$. W ten sposób mamy większą szansę, że funkcja zwrócona przez ucznia będzie poprawna dla 1.

Niech teraz funkcja h_2 zwrócona dla rozkładu p_2 myli się tylko na 2. Jej błąd wynosi $e_2 = 1/6$, co daje $\alpha_2 = \log(\sqrt{5})$, $e^{\alpha_2} = \sqrt{5}$ i $Z_2 = \sqrt{5}/3$, a zatem definiując p_3, prawdopodobieństwo dla 2 mnożymy przez 3, a dla pozostałych przez $3/5$ itd:

i	$h_i(1)$	$h_i(2)$	$h_i(3)$	$h_i(4)$	$p_i(1)$	$p_i(2)$	$p_i(3)$	$p_i(4)$	e_i	e^{α_i}	Z_i	$\widehat{\mathrm{err}}$
1	$\underline{1}$	1	-1	1	1/4	1/4	1/4	1/4	1/4	$\sqrt{3}$	$\sqrt{3}/2$	0,866
2	-1	$\underline{-1}$	-1	1	1/2	1/6	1/6	1/6	1/6	$\sqrt{5}$	$\sqrt{5}/3$	0,645
3	-1	1	$\underline{1}$	1	3/10	1/2	1/10	1/10	1/10	3	3/5	0,387
4	-1	1	-1	$\underline{-1}$	1/6	5/18	1/2	1/18	1/18	$\sqrt{17}$	$\sqrt{17}/9$	0,177

Błędne wartości zwrócone przez ucznia są podkreślone: funkcja h_i myli się jedynie na i. Zwrócone funkcje są oczywiście przykładowe i mocno wpływają na działanie algorytmu. Na przykład mogłoby się zdarzyć, że funkcja h_2 też zwraca błędną wartość dla 1. Szansę na to staramy się jednak minimalizować, dostosowując prawdopodobieństwa.

Po czterech iteracjach zwracamy funkcję

$$\operatorname{sgn}\big(\log(\sqrt{3})h_1 + \log(\sqrt{5})h_2 + \log(3)h_3 + \log(\sqrt{17})h_4\big).$$

Łatwo sprawdzić, że funkcja ta jest równa funkcji f, której staraliśmy się nauczyć, np. dla $i = 1$ mamy: $\operatorname{sgn}(\log(\sqrt{3}) - \log(\sqrt{5}) - \log(3) - \log(\sqrt{17})) = -1$.

Ostatnia kolumna zawiera górne oszacowanie błędu dotychczas skonstruowanej funkcji dla jednostajnego rozkładu prawdopodobieństwa, które zaraz zdefiniujemy. Zauważmy, że po czwartej rundzie to oszacowanie jest mniejsze niż $1/4$, co pokazuje, że zwracana funkcja musi być poprawna (funkcja zwrócona po trzeciej iteracji też jest już poprawna).

Analiza błędu

Chcemy oszacować z góry błąd $\mathrm{err}_p\left(\operatorname{sgn}\sum_{i=1}^m \alpha_i h_i\right)$. Takie funkcje trudno szacować, trudno też konstruować algorytm, który stara się ją minimalizować. Zamiast tego często używa się ograniczeń górnych i konstruuje algorytmy tak, by minimalizowały takie ograniczenie górne, zwyczajowo nazywane wówczas funkcją kosztu. My skorzystamy z $\widehat{\mathrm{err}}_p$, zadanego jako:

$$\widehat{\mathrm{err}}_p(g) = \sum_{x\in\mathcal{X}} p(x)\exp(-f(x)g(x)) \geqslant \sum_{x\in\mathcal{X}: f(x)\neq \operatorname{sgn} g(x)} p(x) = \mathrm{err}_p(\operatorname{sgn} g),$$

Rozwiązanie zadania F 1067.
Suma napięć na kondensatorach C_1 i C_2 równa jest napięciu baterii $\mathcal{E}$, tak samo jak suma napięć na kondensatorach C_3 i C_4. Ładunek zgromadzony na kondensatorze C_1 jest równy ładunkowi na kondensatorze C_2. W związku z tym potencjał U_A w punkcie A wynosi:

$$U_A = \frac{1/C_1}{1/C_1 + 1/C_2} = \frac{C_2}{C_1 + C_2}.$$

Analogiczne rozumowanie prowadzi do wniosku, że potencjał U_B w punkcie B wynosi:

$$U_B = \frac{1/C_3}{1/C_3 + 1/C_4} = \frac{C_4}{C_3 + C_4}.$$

Napięcie $U_{AB} = 0$, gdy $U_A = U_B$. Po prostych przekształceniach otrzymujemy, że poszukiwanym warunkiem jest $C_2/C_1 = C_4/C_3$.

Rozwiązanie zadania F 1068.
Wewnątrz przewodnika (pod powierzchnią ziemi) natężenie pola elektrycznego wynosi zero. Po zewnętrznej stronie powierzchni przewodnika pole elektryczne ma tylko składową prostopadłą do tej powierzchni. Takie cechy na płaskiej powierzchni (zakładamy oczywiście, że h jest znacznie mniejsze od promienia Ziemi) ma superpozycja pól pochodzących od naładowanego przewodu i jego „obrazu", tj. równoległego doń przewodu znajdującego się na głębokości h pod powierzchnią i równomiernie naładowanego ładunkiem λ na jednostkę powierzchni. Pole elektryczne wytwarzane przez jednorodny, liniowy rozkład ładunku jest prostopadłe do przewodu, a jego wartość zależy wyłącznie od odległości od naładowanego przewodu. Tę wartość wyznaczymy, posługując się prawem Gaussa. Zamykamy odcinek l przewodu w walcu o promieniu r i osi symetrii pokrywającej się z przewodem. Strumień pola przez powierzchnię takiego walca wynosi $2\pi Erl$ i jest równy ładunkowi zawartemu w walcu dzielonemu przez przenikalność elektryczną próżni ε_0: $2\pi Erl = \lambda l/\varepsilon_0$, a więc:

$$E = \frac{\lambda}{2\pi\varepsilon_0 r}.$$

Na powierzchni ziemi, w odległości x od pionowego rzutu przewodu, składowa E_n pola prostopadła do tej powierzchni jest sumą składowych od przewodu i jego „obrazu" i wynosi:

$$E_n = \frac{\lambda h}{\pi\varepsilon_0(h^2+x^2)}.$$

Powierzchniową gęstość ładunku, σ, otrzymujemy, znowu stosując prawo Gaussa – zamykamy niewielki element powierzchni ziemi w walcu i obliczamy strumień pola. Tym razem strumień przez powierzchnię boczną (pole jest prostopadłe do powierzchni) i „denko" walca znika (zerowe pole wewnątrz ziemi-przewodnika). Wynik:

$$\sigma = \frac{\lambda h}{\pi(h^2+x^2)}.$$

gdzie $\exp(\cdot)$ oznacza funkcję wykładniczą. Istotnie: jeśli $f(x) = \operatorname{sgn} g(x)$, to x wnosi 0 do $\operatorname{err}_p(\operatorname{sgn} g)$, zaś do $\widehat{\operatorname{err}}(g)$: liczbę nieujemną; jeśli $f(x) \neq \operatorname{sgn} g(x)$, to x daje $p(x)$ w $\operatorname{err}_p(\operatorname{sgn} g)$, zaś w $\widehat{\operatorname{err}}(g)$: $p(x)\exp(-f(x)g(x)) \geqslant p(x)$.

Postać funkcji zwracanej przez AdaBoost jest tak dobrana, by łatwo było obliczyć dla niej funkcję kosztu $\widehat{\operatorname{err}}_p$. Aby to pokazać, zauważmy najpierw, że mnożenie/dzielenie przez e^{α_i} w zależności od tego, czy zachodzi $h_i(x) = f(x)$, można zwięźle wyrazić, korzystając z tego, że $f(x), h_i(x) \in \{-1, 1\}$:

$$\exp(-\alpha_i f(x) h_i(x)) = \begin{cases} e^{\alpha_i} & \text{jeśli } h_i(x) \neq f(x) \\ e^{-\alpha_i} & \text{jeśli } h_i(x) = f(x). \end{cases}$$

Zależność na p_{i+1} można więc zapisać jako $p_{i+1}(x) = p_i(x)\exp(-\alpha_i f(x)h_i(x))/Z_i$. Teraz przyjmując $p = p_1$, otrzymujemy

$$\widehat{\operatorname{err}}_p\Big(\sum_{i=1}^m \alpha_i h_i\Big) = \sum_{x\in\mathcal{X}} p(x)\exp\Big(-f(x)\sum_{i=1}^m \alpha_i h_i(x)\Big) = \sum_{x\in\mathcal{X}} p_1(x)\prod_{i=1}^m \exp(-\alpha_i f(x)h_i(x))$$
$$= \sum_{x\in\mathcal{X}} p_1(x)\prod_{i=1}^m \frac{p_{i+1}(x)}{p_i(x)}Z_i = \sum_{x\in\mathcal{X}} p_{m+1}(x)\prod_{i=1}^m Z_i = \prod_{i=1}^m Z_i \cdot \sum_{x\in\mathcal{X}} p_{m+1}(x) = \prod_{i=1}^m Z_i.$$

Zauważmy też, że zależność ta jest prawdziwa dla *dowolnych* wartości $\alpha_1, \ldots, \alpha_m$ i zdefiniowanych dla nich $Z_1, \ldots, Z_m$. Teraz wiadomo, dlaczego $\alpha_i = \frac{1}{2}\log\Big(\frac{1-e_i}{e_i}\Big)$: jest to wartość, dla której $Z_i(\alpha) = e_i e^{\alpha} + (1-e_i)\cdot e^{-\alpha}$ jest najmniejsze, o czym można się przekonać, używając rachunku różniczkowego.

Pozostają nam proste rachunki: podstawiając $\alpha_i = \frac{1}{2}\log\Big(\frac{1-e_i}{e_i}\Big)$ do Z_i, dostajemy

$$Z_i = 2\sqrt{e_i(1-e_i)} = \sqrt{1-4\Big(e_i-\frac{1}{2}\Big)^2} \leqslant \exp\Big(-2\Big(e_i-\frac{1}{2}\Big)^2\Big),$$

przy czym nierówność wynika z nierówności $1-x \leqslant e^{-x}$, prawdziwej dla każdego rzeczywistego x. Pozwala nam to oszacować błąd AdaBoost:

$$\operatorname{err}_p\Big(\operatorname{sgn}\sum_{i=1}^m \alpha_i h_i\Big) \leqslant \widehat{\operatorname{err}}_p\Big(\sum_{i=1}^m \alpha_i h_i\Big) = \prod_{i=1}^m Z_i \leqslant \prod_{i=1}^m \exp\Big(-2\Big(e_i-\frac{1}{2}\Big)^2\Big).$$

Przeanalizujmy dokładniej oszacowanie $\exp\big(-2\big(e_i-\frac{1}{2}\big)^2\big)$. Jeśli $e_i = 1/2$, to wynosi ono 1, co ma naturalną interpretację: jeśli h_i myli się w 1/2 przypadków, to analiza daje takie samo ograniczenie błędu, jak bez h_i. Jeśli błąd jest duży: $e_i > 1/2$, to wciąż zyskujemy: $(e_i - 1/2)^2 > 0$. Co też ma interpretację: możemy h_i zamienić na $-h_i$, którego błąd jest mniejszy niż 1/2; łatwo sprawdzić, że AdaBoost poprawnie zmienia prawdopodobieństwa, gdy $e_i > 1/2$.

Z definicji słabego ucznia mamy $\frac{1}{2} - e_i > \epsilon$ z prawdopodobieństwem przynajmniej $1-\delta$ dla pewnego stałego $\delta > 0$. Czyli „oczekujemy" błędu wykładniczo malejącego wraz z m. Używając standardowych metod rachunku prawdopodobieństwa, można uzasadnić, że implikuje to warunek silnego ucznia (dla odpowiedniej liczby t_0 zależnej od δ).

Praktyka: uczenie maszynowe

W praktyce dokładne określenie błędu e_i jest trudne (musimy mieć informację o wszystkich wartościach f oraz h_i). Ale nawet przybliżona wartość, którą można obliczyć przez losowe próbkowanie, prowadzi do przybliżonej wartości α_i i sensownej wartości Z_i, która też gwarantuje zmniejszenie błędu.

W implementacji AdaBoost za h_i przyjęto warunki biorące pod uwagę tylko jedną cechę – nie zawsze spełniają one warunek bycia słabym uczniem, ale działają dobrze w praktyce. Taka implementacja jest bardzo szybka i bardzo dobrze sobie radzi w praktyce.

AdaBoost zapoczątkował nowe podejście w uczeniu maszynowym, znane jako *boosting*: zamiast próbować skonstruować bardziej skomplikowany, dokładny model, używamy wielu prostszych modeli, których odpowiedzi (jakoś) uśredniamy (być może z wagami). Warianty boostingu zaimplementowane są praktycznie w każdym narzędziu do uczenia maszynowego.

Dane są zwykle wielowymiarowe i $\mathcal{X}$ jest zbiorem nie liczb, a krotek, w których cechy odpowiadają kolejnym pozycjom. Przykład można było zobaczyć w poprzednim numerze *Delty*, w artykule o czarnej skrzynce, która groźność zwierzęcia oceniała na podstawie trzech cech (Δ^2_{23}). AdaBoost użyłby tu trzech funkcji h_i, po jednej dla każdej cechy.

Liniowe kombinacje drzew decyzyjnych są między innymi stosowane w lasach losowych (*random forest*).