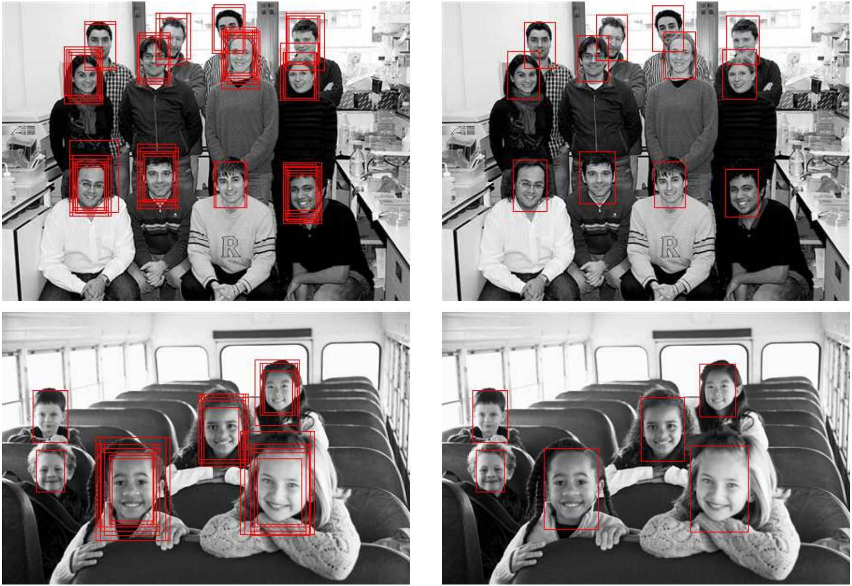
Techniki szybkiej detekcji: ekstrakcja cech poprzez obrazy całkowe, boosting, kaskady klasyfikatorów

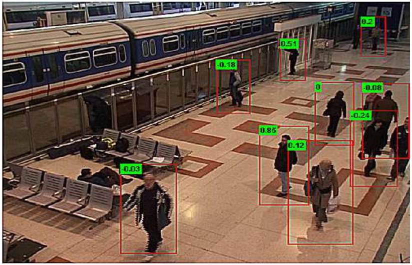
Przemysław Klęsk
pklesk@wi.zut.pl

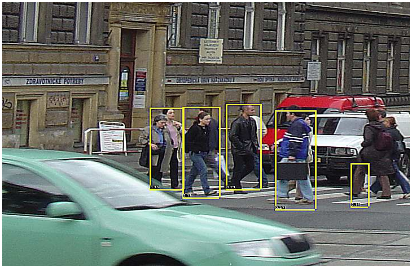
Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej
Wydział Informatyki, ZUT w Szczecinie

Detekcja — przykłady



Detekcja — przykłady





Spis treści

Dwa podejścia do ekstrakcji cech

Cechy reprezentujące (rekonstruujące) wielkości fizyczne

- Częstsze w zadaniach *rozpoznawania* (wiele klas, podobnych) niż *detekcji* (2 klasy: „obiekt vs nie-obiekt", zróżnicowane).
- Np. w rozpoznawaniu twarzy: rozstaw oczu, długość nosa, wysokość czoła, odległość oczu od warg, odległość oczu od brody, kolor skóry, itp.
- Kosztowne obliczeniowo.
- Zwykle nieduże zbiory cech (kilka, kilkanaście, kilkadziesiąt).

Proste cechy graficzne / geometryczne

- Cechy zorientowane na prosty opis elementów kształtu.
- Tanie obliczeniowo.
- Duże zbiory cech do uczenia ($\sim 10^4$, $\sim 10^5$) — „atak brutalny".
- Związek pomiędzy prostymi cechami a klasami bywa niejasny dla projektanta.
- Przykłady: surowe piksele + PCA, cechy Haara, deskryptor HOG, kody teksturowe, torba słów, momenty niskich rzędów (statystyczne, Fouriera, DCT), itp.
- Oczekujemy, że algorytm selekcji cech lub sam algorytm uczący wybierze podzbiór cech istotnych ($\sim 10^2$, $\sim 10^3$) do pracy finalnego detektora.

Spis treści

Złożoność procedury skanującej (detekcyjnej)

- **Szkic procedury**: Wykonujemy pętlę po kilku skalach. Dla ustalonej skali przebiegamy obraz oknem przesuwnym. Dla każdej pozycji okna wykonujemy obliczenia związane z: (1) ekstrakcją cech, (2) wyznaczeniem odpowiedzi detektora (klasyfikatora).
- **Złożoność dla pojedynczej skali (pesymistyczna)**:

$$O\Big((n_x - w_x + 1)(n_y - w_y + 1)/(d_x d_y)\,(n\,c_{fe/px} w_x w_y + n\,c_{d/f})\Big), \tag{1}$$

  gdzie:
  $n_x \times n_y$ — wymiary obrazu,
  $w_x \times w_y$ — wymiary okna przesuwnego,
  d_x, d_y — skoki okna przesuwnego,
  n — liczba cech wybranych ostatecznie do detekcji,
  $c_{fe/px}$ — średni koszt ekstrakcji 1 cechy na piksel,
  $c_{d/f}$ — średni koszt wyznaczenia odpowiedzi klasyfikatora na 1 cechę.
- **Przykład**:
  $n_x = 480$, $n_y = 640$,
  $w_x = w_y = 20$, (np. rozmiar najmniejszych twarzy, które chcemy wykrywać)
  $d_x = d_y = 1$, (wyczerpujące skanowanie obrazu, liczba pozycji okna: 286 281)
  $n = 1000$, (przypuśćmy, że to wystarczająca liczba cech)
  $c_{fe/px} = c_{d/f} = 10^{-9}$s, (optymistycznie)
  → **koszt ≈ 115 s**.

Pomysły na usprawnienie

1. **Obrazy całkowe / kumulanty (ang. *integral images*)**
   - Obrazy całkowe (jeden lub wiele) obliczane jednokrotnie przed całą procedurą (ewentualnie przed pętlą dla danej skali, jeżeli zależne od skali).
   - Cechy wyznaczane w czasie stałym c_{fe}, niezależnym od liczby pikseli w oknie — $O(1)$.
   - Redukcja złożoności do:

$$O\Big((n_x - w_x + 1)(n_y - w_y + 1)/(d_x d_y)\,(n\,c_{fe} + n\,c_{d/f})\Big). \tag{2}$$

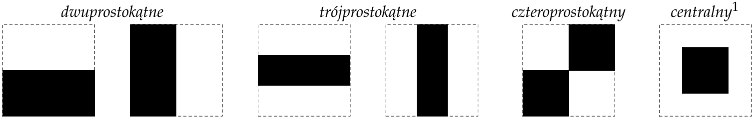
2. **Kaskada klasyfikatorów (ang. *classifiers cascade*)**
   - Pomysł oparty na obserwacji, że okna pozytywne stanowią zwykle bardzo mały ułamek wszystkich okien.
   - Klasyfikator „rozbity” na poziomy (ang. *stages*, *layers*).
   - Poziomy początkowe wykorzystują bardzo mało cech (zwykle kilka). Poziomy dalsze coraz więcej (np. aż do kilkuset).
   - Wskazanie pozytywne wymaga przejścia przez wszystkie poziomy. Wskazanie negatywne po dowolnym poziomie przerywa dalszą analizę.
   - Średnia liczba cech $\bar{n}$ na okno dużo mniejsza od liczby wszystkich cech n ($\bar{n} \ll n$).
   - Redukcja złożoności do:

$$O\Big((n_x - w_x + 1)(n_y - w_y + 1)/(d_x d_y)\,(\bar{n}\,c_{fe/px} w_x w_y + \bar{n}\,c_{d/f})\Big). \tag{3}$$

Spis treści

Cechy Haara (ang. *Haar-like features*)

- Pomysł: **(Viola & Jones, 2001, 2004)**.
- Luźna **analogia do falek Haara** (ang. *Haar wavelets*), w szczególności ortogonalność zaniedbywana w zastosowaniach do detekcji.
- **Szablony**:



- Przykład cech istotnych (niektórych) dla detekcji twarzy:

- **Wartość cechy**: różnica pomiędzy średnią jasnością pikseli w zbiorze „czarnym” i średnią jasnością pikseli w zbiorze „białym” — **zgrubne kontury**.

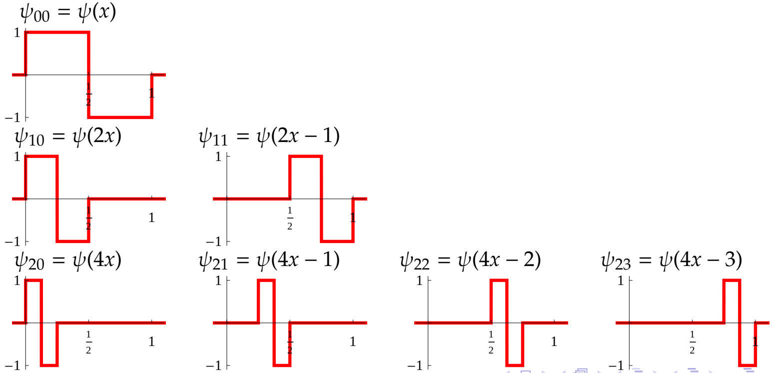
[1] Oryginalnie nie był proponowany w (Viola & Jones, 2001, 2004). Obecnie powszechny.

Falki Haara

- Falka podstawowa

$$\psi(x) = \begin{cases} 1, & 0 \leqslant x < 1/2; \\ -1, & 1/2 \leqslant x < 1; \\ 0, & \text{w.p.r.} \end{cases} \tag{4}$$

- Skalowanie i przesunięcia: $\psi_{j,k} = \psi(2^j x - k)$.



Falki Haara

- Ortogonalność:

$$\forall (j,k) \neq (l,m) \quad \langle \psi_{j,k}, \psi_{l,m} \rangle = \int_0^1 \psi_{j,k}(x)\psi_{l,m}(x)\,dx = 0. \tag{5}$$

- Rozwinięcie (aproksymacja) funkcji:

$$f(x) = c_0 + \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} c_{j,k}\psi_{j,k}(x), \tag{6}$$

gdzie

$$c_0 = \langle f, 1 \rangle = \int_0^1 f(x)\,dx; \tag{7}$$

$$c_{j,k} = \left\langle f, \psi_{j,k}/||\psi_{j,k}||^2 \right\rangle = 1/||\psi_{j,k}||^2 \int_0^1 f(x)\psi_{j,k}(x)\,dx. \tag{8}$$

- Norma funkcji nad $[0,1]$:

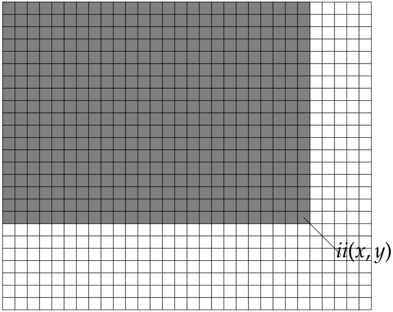
$$||g|| = \sqrt{\int_0^1 g^2(x)\,dx} \qquad \Rightarrow ||g||^2 = \langle g, g \rangle. \tag{9}$$

$$||\psi_{j,k}|| = 2^{-j/2}. \tag{10}$$

Obraz całkowy (ang. *integral image*)

- Dla szybkiego obliczania cech Haara ważne jest użycie **obrazu całkowego** .
- Niech $i(x, y)$ oznacza funkcję obrazu — jasność piksela w punkcie (x, y). Obraz całkowy $ii(x, y)$ określamy jako:

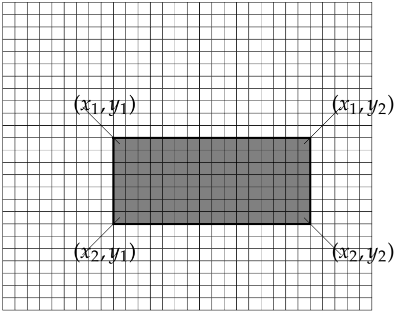
$$ii(x, y) = \sum_{1 \leqslant j \leqslant x} \sum_{1 \leqslant k \leqslant y} i(j, k). \tag{11}$$



Obraz całkowy (ang. *integral image*)

- W jaki sposób, mając obliczony *ii*, można obliczyć szybko sumę jasności pikseli w prostokącie rozpiętym pomiędzy (x_1, y_1) a (x_2, y_2):

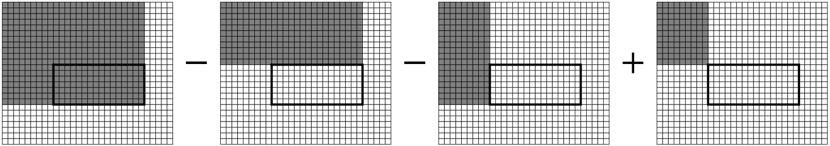
$$\sum_{x_1 \leqslant x \leqslant x_2} \sum_{y_1 \leqslant y \leqslant y_2} i(x,y) = ? \tag{12}$$



Obraz całkowy

- Szybkie obliczanie sumy:

$$\sum_{x_1 \leqslant x \leqslant x_2} \sum_{y_1 \leqslant y \leqslant y_2} i(x,y) = ii(x_2,y_2) - ii(x_1-1,y_2) - ii(x_2,y_1-1) + ii(x_1-1,y_1-1). \tag{13}$$



- **Wystarczają operacje na 4 punktach obrazu całkowego niezależnie od rozmiaru prostokąta — O(1)**.
- Analogia do całki nad prostokątem liczonej jako przyrost funkcji pierwotnej:

$$\int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)\,dx\,dy = F(x_2,y_2) - F(x_1,y_2) - F(x_2,y_1) + F(x_2,y_2), \tag{14}$$

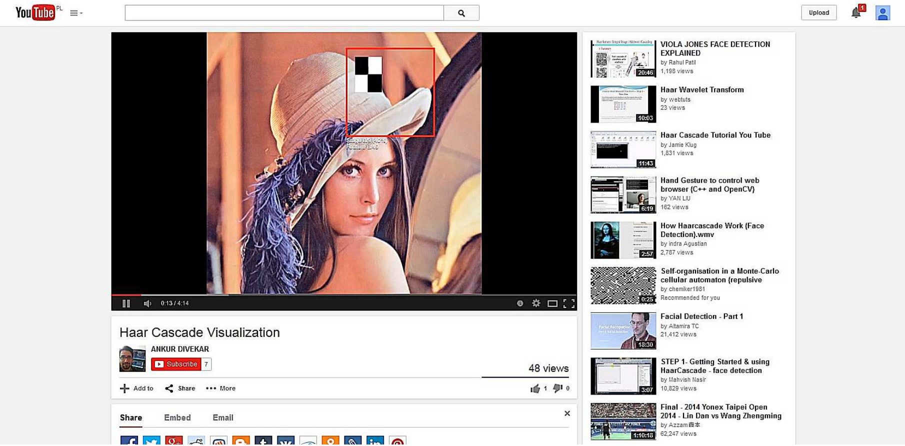
gdzie F jest funkcją pierwotną dla f, tj. $F(x,y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f(u,v)\,du\,dv$.

Wyznaczanie obrazu całkowego

- Wyznaczanie wg definicji (11) na rzecz każdego punktu nieefektywne — $O({n_x}^2 {n_y}^2)$.
- Indukcja wyznacza obraz całkowy w czasie $O(n_x n_y)$.

Algorytm $\textsc{WyznaczObrazCałkowy}(i)$ ▹ obraz i jako argument
    Utwórz dwie tablice: $ii_{n_x \times n_y}$, k_{n_y}. ▹ k posłuży do sumowania w wierszu
    Dla $x := 1, \ldots, n_x$ **powtarzaj**
        Dla $y := 1, \ldots, n_y$ **powtarzaj**
            $k(y) := i(x, y)$.
            Jeżeli $y > 1$ **to** $k(y) := k(y) + k(y-1)$.
            $ii(x, y) := k(y)$.
            Jeżeli $x > 1$ **to** $ii(x, y) := ii(x-1, y) + ii(x, y)$.
    Zwróć ii.

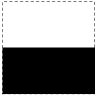
Wizualizacja procedury skanującej



https://www.youtube.com/watch?v=hPCTwxF0qf4

Liczba cech — generowanie wyczerpujące

- **Szkic procedury**: Generujemy wg każdego szablonu cechy nadając im wszystkie możliwe rozmiary i pozycje (z dokładnością 1-pikselową), na które pozwala ustalony rozmiar aktualnego okna ($w_x \times w_y$).
- Sensowne rozmiary minimalne powinny być dobrane z zachowaniem natury wzorca reprezentowanego przez dany szablon.
- Np. dla szablonu:



minimalny rozmiar to 1×2, i liczba wszystkich możliwych cech to:

$$\sum_{1 \leqslant f_x \leqslant w_x} \sum_{2 \leqslant f_y \leqslant w_y} (w_x - f_x + 1)(w_y - f_y + 1) \tag{15}$$

$$= 1/4 w_x (w_x + 1) w_y (w_y - 1) \tag{16}$$

$$= 39\,900 \quad (\text{dla } w_x = 20, w_y = 20). \tag{17}$$

- Sumowanie po 5 szablonach daje łączną liczbę cech: $n \approx 200\,000$.
- **Mankamenty**: (1) bardzo dużo cech w zbiorze uczącym, (2) liczba cech zależy od rozmiaru okna (tj. od skali w ramach, której pracuje w danym momencie procedura skanująca oknem przesuwnym).

Liczba cech — parametryzacja

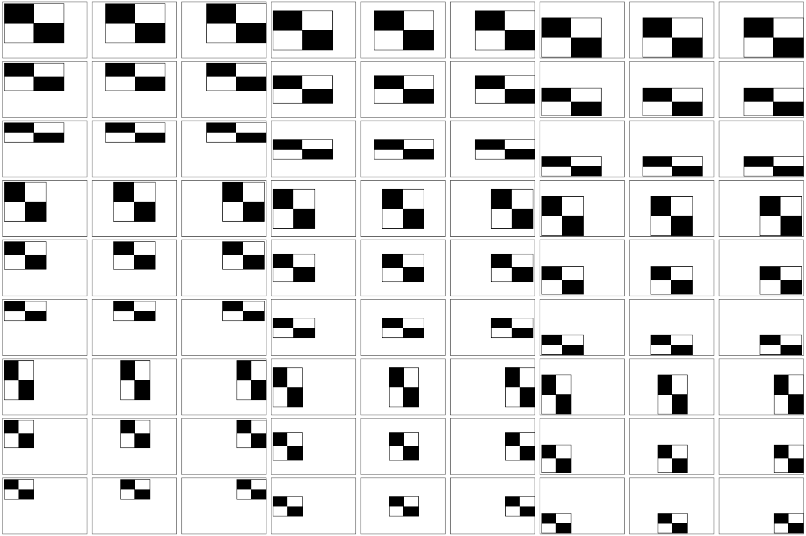
- Zwyczajowo wprowadza się pewną parametryzację poprzez **skalowanie** i **pozycjonowanie** szablonów w ramach okna.
- Niech parametr s oznacza liczbę możliwych skalowanych wersji szablonu wzdłuż każdego z kierunków ($s = 1, 2, \ldots$). Stąd, powstanie s^2 przeskalowanych wersji szablonu.
- Niech parametr p generuje regularną siatkę $(2p-1) \times (2p-1)$ punktów zaczepienia cech ($p = 1, 2, \ldots$).
- **Łączna liczba cech**:

$$n(s,p) = 6s^2(2p-1)^2. \tag{18}$$

- Przykłady:

	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
$s = 1$	5	45	125	245	405
$s = 2$	20	180	500	980	1 620
$s = 3$	45	405	1 125	2 205	3 645
$s = 4$	80	720	2 000	3 920	6 480
$s = 5$	125	1 125	3 125	6 125	10 125

Przykład dla parametryzacji $s = 3, p = 2$



Parametryzacja — przykładowe podejścia

- Arytmetyczne skalowanie szablonów (o stałym przyroście):

$$f_x := \mathrm{round}\left(f_x^{\min} + j(f_x^{\max} - f_x^{\min})/(s-1)\right), \qquad s > 1; \tag{19}$$

$$f_y := \mathrm{round}\left(f_y^{\min} + k(f_y^{\max} - f_y^{\min})/(s-1)\right), \qquad s > 1; \tag{20}$$

  dla $(j,k) \in \{(0,0),(0,1),\ldots(s-1,s-1)\}$, gdzie $f_x \times f_y$ to rozmiar cechy w pikselach, a $f_{\cdot}^{\min}$, $f_{\cdot}^{\max}$ to odpowiednio minimalny i maksymalny ustalony rozmiar cechy wzdłuż danego kierunku. Np. $f_{\cdot}^{\max} = w_{\cdot}$, $f_{\cdot}^{\min} = \mathrm{round}(0.1 w_{\cdot})$.

- Potęgowe (wykładnicze) skalowanie szablonów:

$$f_x := \mathrm{round}\left(\left(\sqrt{2}/2\right)^j f_x^{\max}\right); \tag{21}$$

$$f_y := \mathrm{round}\left(\left(\sqrt{2}/2\right)^k f_y^{\max}\right); \tag{22}$$

  dla $(j,k) \in \{(0,0),(0,1),\ldots(s-1,s-1)\}$. Zwiększenie obu indeksów (j,k) o 1 powoduje dwukrotne pomniejszenie cechy co do jej powierzchni.

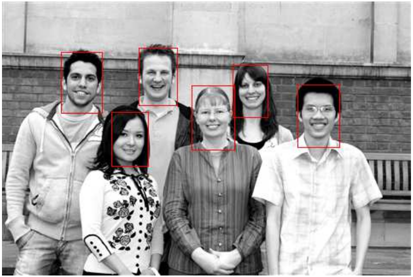
- Pozycje i skoki punktów siatki $(2p-1)\times(2p-1)$ do zaczepiania cech powinny być ustalane na podstawie wolnego marginesu pozostałego pomiędzy szerokością okna i cechy: $w_{\cdot} - f_{\cdot}$.

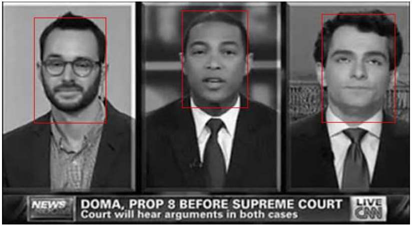
Przykładowy detektor twarzy

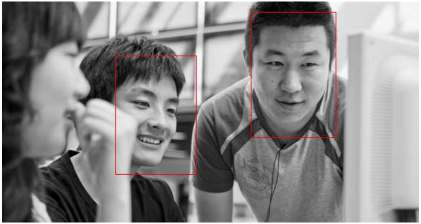
- Materiał uczący: 1 500 zdjęć z "Google Images". Zapytania: „face", „people", „person", „group of people", „family photo", „two person photo", itp.
- Pozycje twarzy na zdjęciach markowane ręcznie.
- Przed generowaniem zbioru uczącego zdjęcia wstępnie unormowane do wysokości 320 pikseli (szerokość proporcjonalna do oryginalnej) oraz przetworzone do skali szarości.
- Obraz przebiegany w 8 skalach. Minimalne rozmiary okna: 50×34 (twarze mniejsze nie będą wykrywane).
- Podczas generowania zbioru uczącego wszystkie okna pozytywne zapamiętywane. Okna negatywne próbkowane losowo na poziomie 0.1%.
- Liczba przykładów w zbiorze uczącym $\approx 80\,000$, liczba cech $n(s, p) = 10\,584$ (parametryzacja: $s = 6$, $p = 4$). Rozmiar zbioru ≈ 5 GB.
- Uczenie przeprowadzone algorytmem *Binning Real Boost*.
- Finalny zbiorowy klasyfikator zawiera 250 słabych klasyfikatorów, każdy oparty na 1 cesze (32 kosze histogramowe na cechę).
- Brak implementacji kaskady — podczas testowania dla każdego okna wyznaczane 250 cech.
- Zrównoleglenie na poziomie pętli po skalach.
- Średni czas analizy jednego okna obrazu: ≈ 0.030 ms; średni czas analizy dla obrazu 320×320: ≈ 1 s (Intel Core i7 2×4 CPU, 1.6 GHz).

Przykładowy detektor twarzy

- Przykłady detekcji bez błędów:

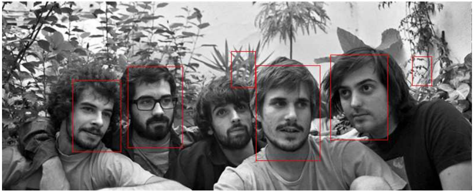




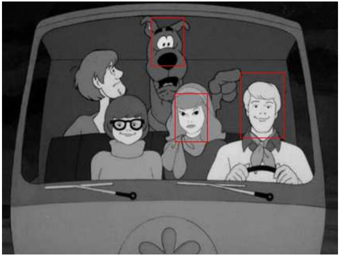


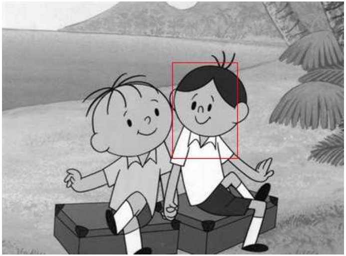
Przykładowy detektor twarzy

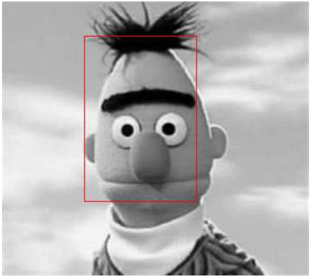
- Przykłady z błędami:

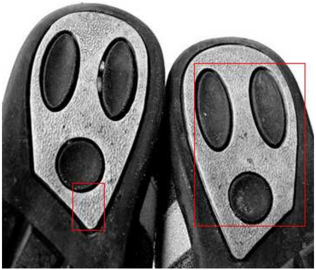


Twarze?







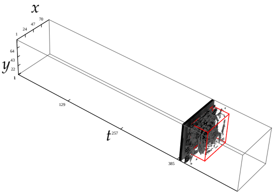


Detektor min (cechy Haara 3D)

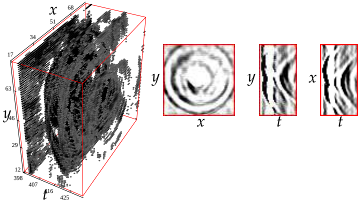
- Obrazy trójwymiarowe — C-skany — z georadaru (*Ground Penetrating Radar*) w układzie: *across track* × *along track* × *time*. Funkcja obrazu: $i(x, y, t)$.
- Oś czasu może być kojarzona intuicyjnie z osią głębokości. Radar częstotliwościowy próbki czasowe otrzymywane otrzymywane z sygnałów zespolonych przez IFFT.
- Obiekty nieprzezroczyste dla radaru generują w obrazie *hiperboloidy*.
- Projekt: (Olech, Kapruziak, Godziuk, Klęsk, 2011–2014). W projekcie badania m.in. nad różnymi cechami 3D obliczanymi poprzez obrazy całkowe: *cechy Haara*, *momenty statystyczne*, *momenty Fouriera*, *deskryptor HOG*.

scena przed zakopaniem

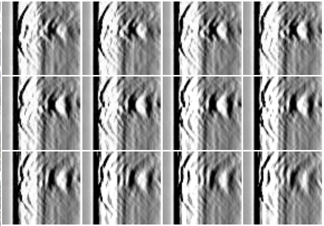
C-scan (po progowaniu)
okno skanujące w punkcie
$(x, y, t) = (17, 12, 398)$



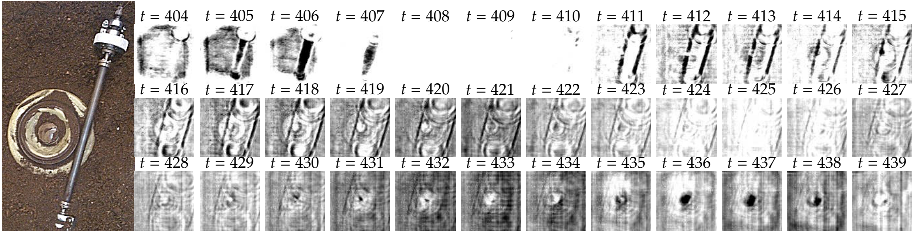
zbliżenie okna skanującego i przekroje



Detektor min (cechy Haara 3D)

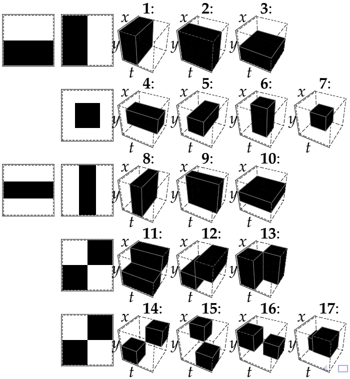


Detektor min (cechy Haara 3D)



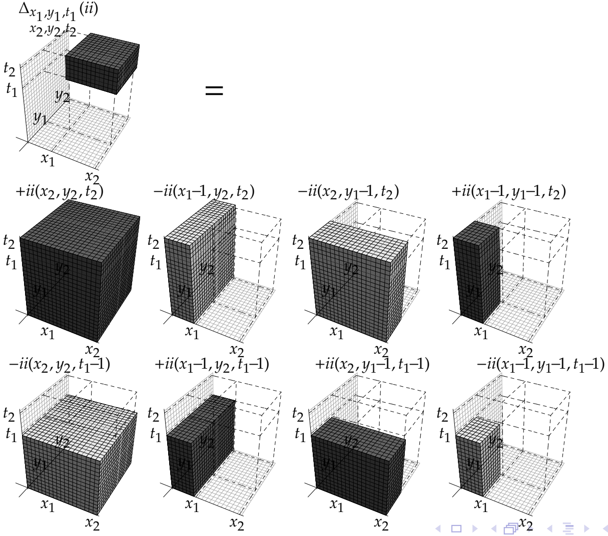
Detektor min (cechy Haara 3D)

- Cechy Haara 3D — propozycja 17 szablonów.
- Liczba cech użytych do uczenia: 17 000.
- Zbiory uczące na podstawie 210 C-skanów: ≈ 7 GB (≈ 100 000 przykładów okien 3D).
- Uczenie za pomocą *boostowanych drzew decyzyjnych* (płytkie drzewa — 4 lub 8 terminali).
- Finalny klasyfikator zawiera 600 drzewek i tym samym wybiera podzbiór maksymalnie 1 800 lub 4 200 cech (odpowiednio dla 4 i 8 terminali).



Detektor min (cechy Haara 3D)

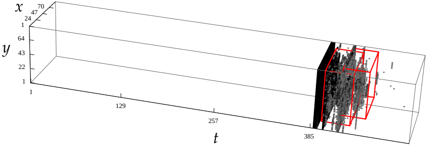
- Przyrost obrazu całkowego $ii(x,y,t)$ obliczany na podstawie 8 punktów:



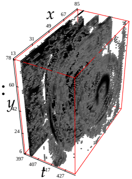
Detektor min (cechy Haara 3D)

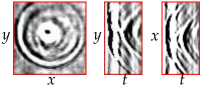
- Przykład detekcji metalowej miny AT z dotatkowym fałszywym alarmem:

1:

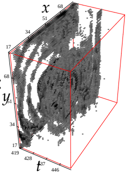


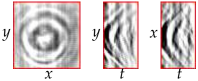
1.1:





1.2:

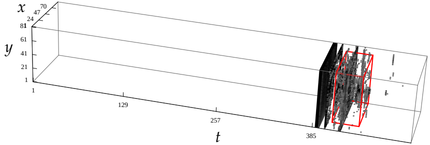




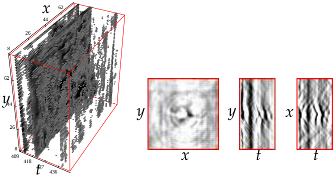
Detektor min (cechy Haara 3D)

- Przykład detekcji plastikowej miny AT:

2:

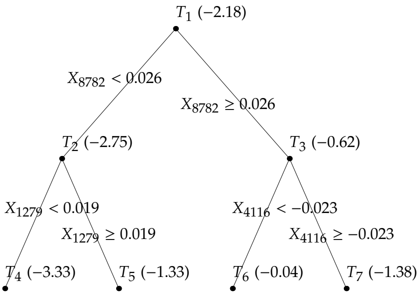


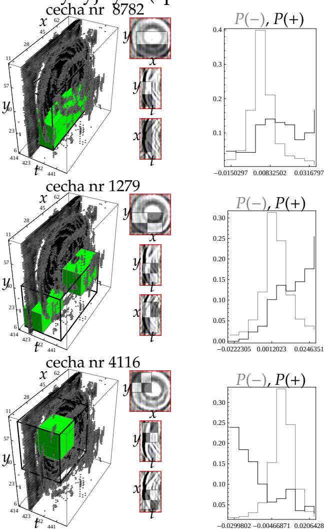
2.1:



Detektor min (cechy Haara 3D)

- Przykłdowe istotne cechy w pierwszym drzewie decyzyjnym (spośród 600 drzew):





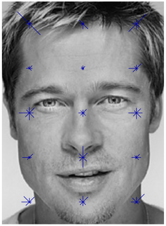
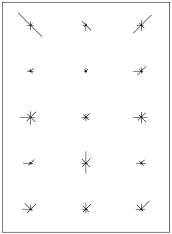
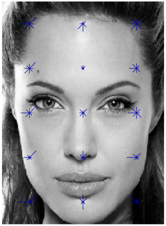
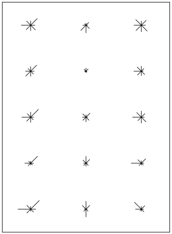
Spis treści

HOG (ang. *Histogram of Oriented Gradients*)

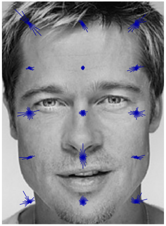
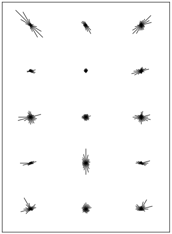
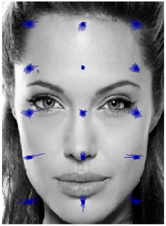
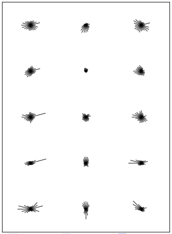
- Pomysł opisany po raz pierwszy w pracy: (Dalal & Triggs, 2005).
- Technika **zlicza wielkości i orientacje lokalnych gradientów** występujących w ramach komórek okna obrazu.
- Orientacje (kierunki) gradientów są dyskretyzowane do skończonego zbioru (dyskretyzacja przedziału $[-\pi/2, \pi/2]$ lub $[0, 2\pi]$).
- Okno dzielone jest na **regularną siatkę komórek** (ang. *cells*).
- **Każdy piksel „głosuje”** w ramach swojej komórki na pewną orientację (kierunek) gradientu z siłą proporcjonalną do długości gradientu zaczepionego w tym pikselu.
- Poprzez **grupowanie komórek w większe bloki** (ang. *blocks*) wprowadza się **normalizację** gradientów, aby zniwelować wpływ lokalnych kontrastów obrazu.
- Wektor cech to konkatenacja rozkładów gradientów (nad dyskretnym zbiorem kierunków) po wszystkich komórkach okna.

HOG — przykłady dla twarzy

- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 8$ przedziałów. Siatka komórek: 5×3. Cech: 120.

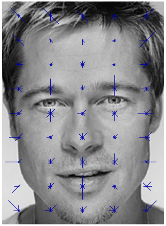
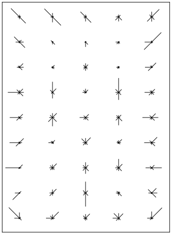
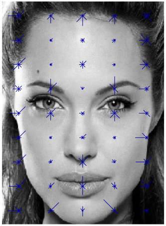
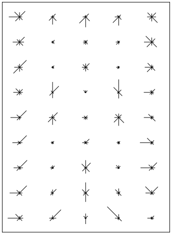





- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 5×3. Cech: 360.

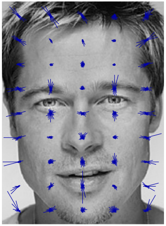
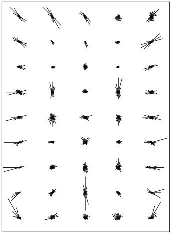
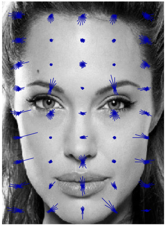





HOG — przykłady dla twarzy

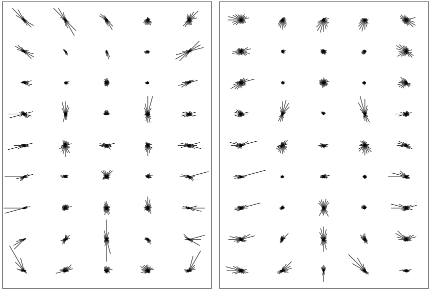
- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 8$ przedziałów. Siatka komórek: 9×5. Cech: 360.

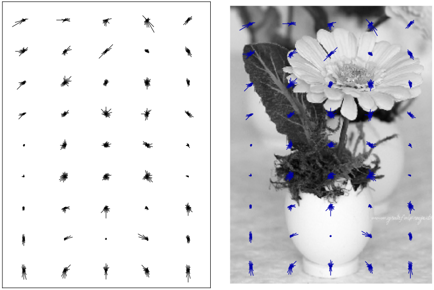
- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 9×5. Cech: 1 080.


HOG — twarz vs nie-twarz

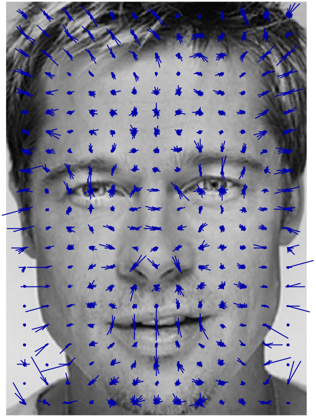


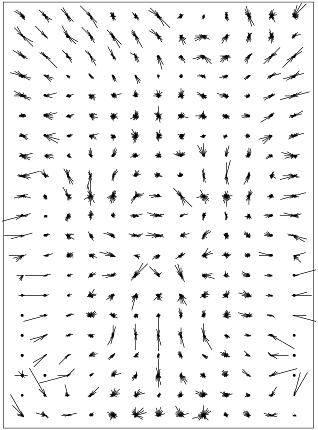
vs



HOG — przykłady dla twarzy

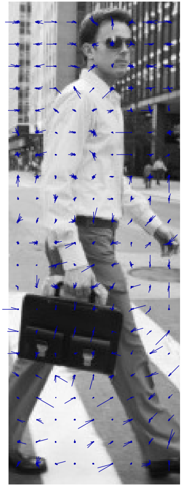
- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 21×13. Cech: 6 552.
- Wizualizacja gradientów na gęstej siatce coraz bardziej przypomina twarz.

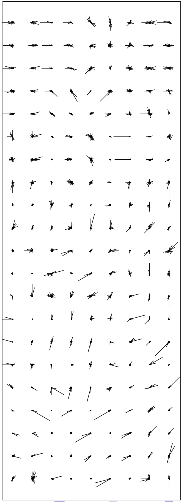




HOG — przykłady dla sylwetek ludzkich

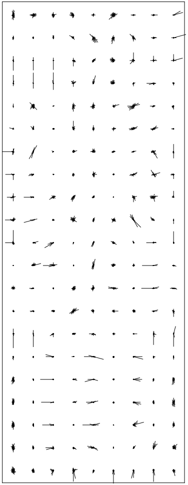
- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 21×9. Cech: $4\,536$.





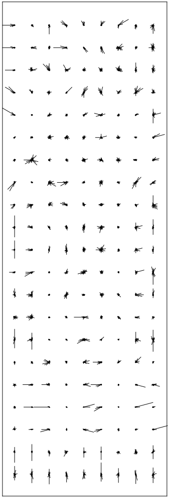
HOG — przykłady dla sylwetek ludzkich

- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 21×9. Cech: 4 536.

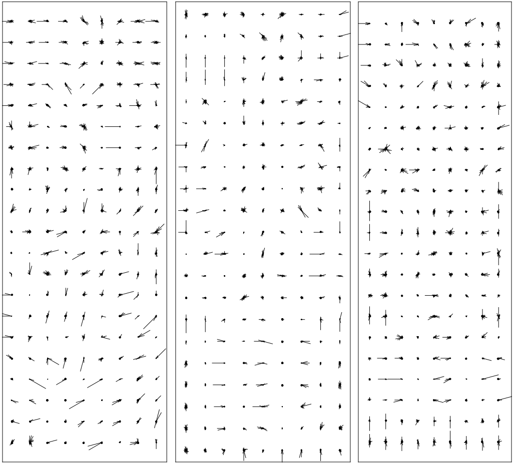


HOG — przykłady dla sylwetek ludzkich

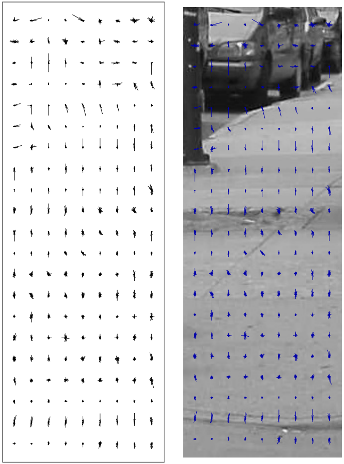
- Dyskretyzacja $[0, 2\pi]$ na $n_\theta = 24$ przedziały. Siatka komórek: 21×9. Cech: 4 536.



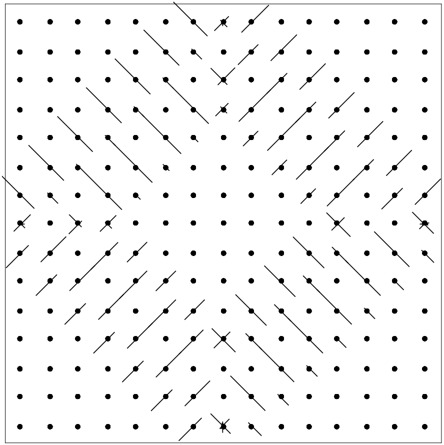
HOG — sylwetka vs nie-sylwetka



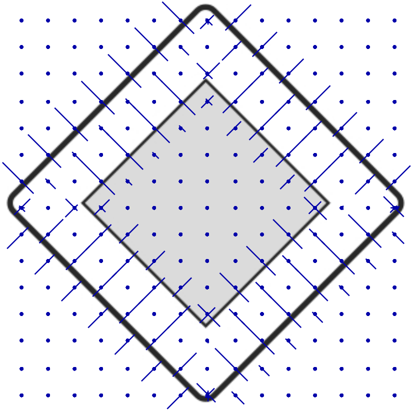
vs

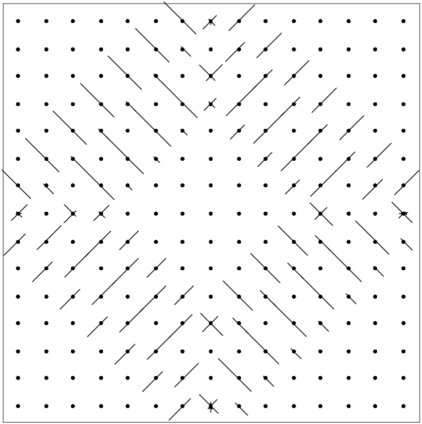


HOG — co to jest?

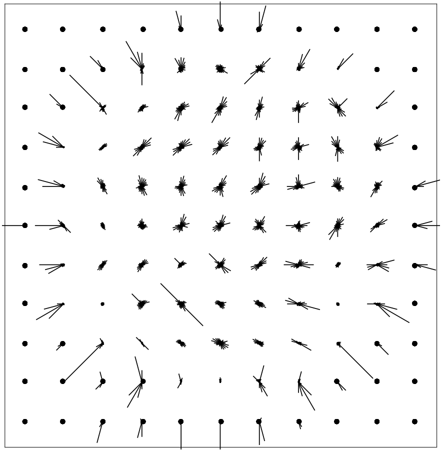


HOG — co to jest?

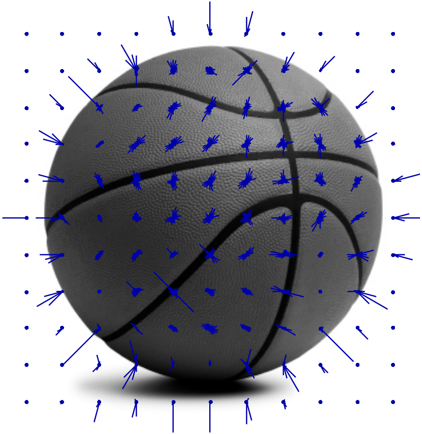


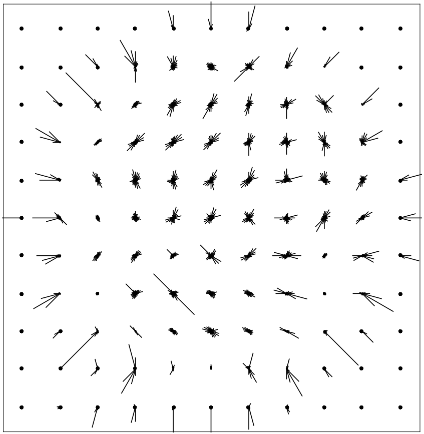


HOG — co to jest?



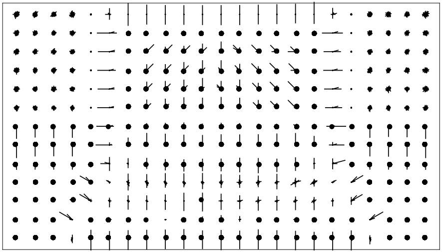
HOG — co to jest?



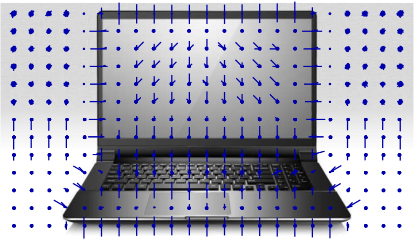
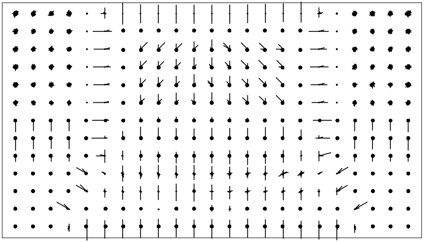


HOG — co to jest?

?



HOG — co to jest?

HOG — opis działania (1)

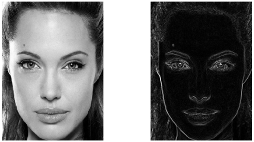
- W pierwszym kroku należy przetworzyć obraz do **poziomów szarości**.
- Następnie **obraz jest splatany z prostymi filtrami estymującymi gradienty** (kontury) wzdłuż każdej z osi: $h_x = (-1, 0, 1)$, $h_y = (-1, 0, 1)^T$, tj.:

$$g_x = i * h_x \tag{23}$$

$$g_y = i * h_y. \tag{24}$$

- **Długość gradientu** w każdym pikselu (j, k) obliczamy jako:

$$G(j,k) = \sqrt{{g_x}^2(j,k) + {g_y}^2(j,k)}. \tag{25}$$



HOG — opis działania (2)

- Dla każdego piksela należy wyznaczyć **dominujący kąt** $\theta(j,k)$.
- Istnieją **dwie możliwości zakresu kąta**, który chcemy rozpatrywać: $[-\pi/2, \pi/2]$ lub $[0, 2\pi)$. Wybór zależy od tego, czy chcemy **uwzględnić czy zaniedbać zwrot gradientu**. Wybór $[-\pi/2, \pi/2]$ niesie też drobne uproszczenie obliczeniowe.
- Prawdziwy (nieuproszczony) gradient jest zwrócony od obszaru ciemniejszego do obszaru jaśniejszego.
- Funkcja $\tan^{-1}$ zwraca wynik z przedziału $(-\pi/2, \pi/2)$. Dlatego w przypadku prawdziwego gradientu trzeba rozpatrzeć kilka przypadków, aby wynik odpowiednio przekształcić do $[0, 2\pi]$.

HOG — opis działania (3)

- Dla zakresu kąta $[-\pi/2, \pi/2]$ (przypadek uproszczony):

$$\theta(j,k) := \begin{cases} 0, & \text{dla } g_y(j,k)=0; \\ \pi/2, & \text{dla } g_y(j,k)>0,\ g_x(j,k)=0; \\ -\pi/2, & \text{dla } g_y(j,k)<0,\ g_x(j,k)=0; \\ \tan^{-1}\dfrac{g_y(j,k)}{g_x(j,k)}, & \text{w.p.r..} \end{cases} \tag{26}$$

- Dla zakresu kąta $[0, 2\pi)$ (prawdziwy gradient):

$$\theta(j,k) := \begin{cases} 0, & \text{dla } g_y(j,k)=0,\ g_x(j,k)\geqslant 0; \\ \pi, & \text{dla } g_y(j,k)=0,\ g_x(j,k)<0; \\ \pi/2, & \text{dla } g_y(j,k)>0,\ g_x(j,k)=0; \\ 3/2\pi, & \text{dla } g_y(j,k)<0,\ g_x(j,k)=0; \\ \tan^{-1}\dfrac{g_y(j,k)}{g_x(j,k)}, & \text{dla } g_y(j,k)\neq 0,\ g_x(j,k)>0; \\ \tan^{-1}\dfrac{g_y(j,k)}{g_x(j,k)}+\pi, & \text{dla } g_y(j,k)\neq 0,\ g_x(j,k)<0. \end{cases} \tag{27}$$

$$\theta(j,k) := \theta(j,k) + 2\pi, \quad \text{jeżeli } \theta(j,k) < 0. \tag{28}$$

HOG — opis działania (4)

- Indywidualne wartości kąta $\theta(j,k)$ i długości $G(j,k)$ gradientu (dla pojedynczych pikseli) mogą być mocno zmiennicze (nawet dla obrazów podobnych).
- Wprowadza się zatem **agregacje** $\theta(j,k)$ i $G(j,k)$ w ramach pewnych prostokątnych otoczeń — **komórek** (ang. *cells*), które stanowią bardziej stabilne statystycznie deskryptory (odporne na nieduże zmiany dla obrazów podobnych).
- Rozmiary komórek (ile na ile pikseli) w ramach okna detekcyjnego wynikają z zadanej przez nas siatki regularnej. Gęstsze siatki przekładają się na większą liczbę cech, ale też na coraz mniejsze komórki (wrażliwe na zmiany).
- Wprowadza się **dyskretyzację zakresu kąta** ($[-\pi/2, \pi/2]$ lub $[0, 2\pi)$) na zadaną liczbę n_θ równoszerokich przedziałów — **koszyków** (ang. *bins*).
- **Każdy piksel głosuje** na koszyk, do którego należy jego kąt $\theta(j,k)$ z siłą głosu $G(j,k)$.
- **Znormalizowane sumy głosów** dla poszczególnych komórek utworzą wspomniane agregacje (stabilne statystki) i dalej cechy.

HOG — opis działania (5)

- Niech kąty brzegowe koszyków będą określone jako:

$$\phi_l = -\pi/2 + l\pi/n_\theta, \qquad l = 0, 1, \ldots, n_\theta; \tag{29}$$

$$\phi_l = -\pi/n_\theta + l2\pi/n_\theta, \qquad l = 0, 1, \ldots, n_\theta; \tag{30}$$

  odpowiednio dla zakresów $[-\pi/2, \pi/2]$ i $[0, 2\pi)$.
- Zatem, kąty środkowe (reprezentatywne) poszczególnych koszyków to:

$$(\phi_l + \phi_{l-1})/2, \qquad l = 1, \ldots, n_\theta. \tag{31}$$

- W przypadku zakresu $[0, 2\pi)$ kąt środkowy dla pierwszego koszyka ($l = 1$) pokrywa się z osią poziomą. Dodatkowo należy uwzględnić „zawienięcie” osi kątów (np. w szczególności, że kąt $-\pi/n_\theta$ odpowiada kątowi $2\pi - \pi/n_\theta$).

HOG — opis działania (6)

- Określamy **macierz głosów** V o wymiarach $n_x \times n_y \times n_\theta$:

$$V(j,k,l) = \begin{cases} G(j,k), & \text{jeżeli } \phi_{l-1} \leq \theta(j,k) < \phi_l; \\ 0, & \text{w.p.r..} \end{cases} \tag{32}$$

- Głosy sumujemy w ramach każdej komórki c. Sumy przechowywane są odrębnie dla poszczególnych koszyków $l = 1, \ldots, n_\theta$:

$$H_1(c,l) = \sum\nolimits_{(j,k)\in c} V(j,k,l). \tag{33}$$

- Ostatecznie, **cechy** H deskryptora HOG obliczane są na podstawie wartości H_1, **normalizując je w ramach bloków** (ang. *blocks*) komórek będących bezpośrednimi sąsiadami:

$$H(c,l) = H_1(c,l) \Big/ \sum\nolimits_{c_q \in N(c)} \sqrt{\|H_1(c_q)\|_2^2 + \epsilon^2}, \tag{34}$$

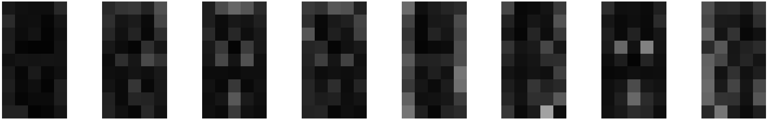
  gdzie: $N(c)$ oznacza zbiór komórek-sąsiadów komórki c, $H_1(c) = \big(H_1(c,1), \ldots, H_1(c,n_\theta)\big)$, $\epsilon > 0$ to dobieralna stała, a notacja $\|\cdot\|$ oznacza normę euklidesową.

HOG — zobrazowanie działania

- Oryginalny obraz, obraz długości gradientów $G(j, k)$, obraz kątów $\theta(j, k)$:

- Obraz macierzy głosów z rozbiciem na kolejne koszyki (przykład dla $n_\theta = 8$)[2]:

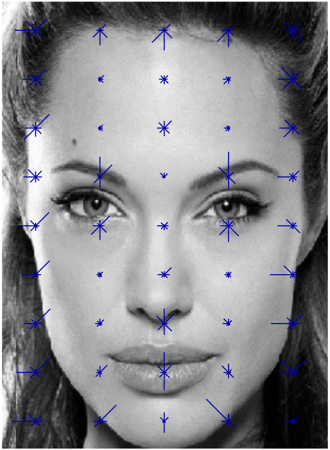
- Wartości cech H nad przykładową siatką 9×5 z rozbiciem na kolejne koszyki ($n_\theta = 8$):



[2] Dla czytelności obrazowanie z negacją odcieni szarości i wyostrzeniem.

HOG — zobrazowanie działania

- Wizualizacja końcowa — w centrum każdej komórki gradienty o długościach $H(c, l)$ narysowane wzdłuż kątów reprezentatywnych poszczególnych koszyków:



HOG — obrazy całkowe

- Pytanie: *Które miejsce można przyspieszyć za pomocą obrazów całkowych (podczas procedury skanującej oknem przesuwnym)?*

HOG — obrazy całkowe

- **Odpowiedź: obliczanie sum głosów w komórkach**, tj.:

$$H_1(c,l) = \sum_{(j,k)\in c} V(j,k,l).$$

- Dla n_θ koszyków należy wprowadzić n_θ **obrazów całkowych kumulujących głosy**:

$$ii_l(x,y) = \sum_{1\leq j\leq x}\sum_{1\leq k\leq y} V(j,k,l), \quad l = 1,\ldots,n_\theta. \tag{35}$$

- Wartość $H_1(c,l)$ dla komórki c rozpiętej pomiędzy punktami $(x_1(c), y_1(c))$ i $(x_2(c), y_2(c))$ można teraz obliczyć jako:

$$H_1(c,l) = ii_l\left(x_2(c), y_2(c)\right) - ii_l\left(x_1(c){-}1, y_2(c)\right) - ii_l\left(x_2(c), y_1(c){-}1\right) + ii_l\left(x_1(c){-}1, y_2(c){-}1\right). \tag{36}$$

- **Dzięki obrazom całkowym (tak samo jak w przypadku cech Haara) wyznaczenie jednej cechy deskryptora HOG odbywa się w czasie stałym O(1)**.

Spis treści

Boosting jako meta-metoda

- **Szkic pomysłu** po raz pierwszy pojawił się w pracy: *"The strength of weak learnability"* **(Schapire, 1990)**.
- Kolejne ważne prace precyzujące współczesną postać boostingu to: (Freund, 1995; Freund & Schapire, 1996, 1997; Schapire & Singer, 1999; Friedman, Hastie, & Tibshirani, 2000).
- Boosting pracuje **stosując sekwencyjnie pewien prosty algorytm uczący do reważonego zbioru danych** (każdy przykład uczący ma swoją wagę, zmieniającą się w kolejnych rundach boostingu).
- W wyniku otrzymujemy **zbiór cząstkowych klasyfikatorów** nazywanych także słabymi klasyfikatorami (ang. *partial/weak classifiers*) — „cokolwiek ciut lepszego od rzutu monetą".
- Ostateczna **odpowiedź zbiorowego (silnego) klasyfikatora** (ang. *ensemble/comittee*) na rzecz pewnego obiektu jest **głosem większościowym** lub **sumą ważoną** odpowiedzi słabych klasyfikatorów.
- Algorytmy boostingu okazują się być dobrym narzędziem do **dużych zbiorów danych**.
- Ważne własności statystyczne obserwowane w praktyce: (1) **zdolność do automatycznej selekcji istotnych cech**, (2) **duża odporność na przeuczenie** — praktyczne zastosowania pokazują, że wraz z dodawaniem nowych słabych klasyfikatorów błąd testowy stabilizuje się (raczej niż rośnie).
- Można pokazać matematycznie, że boosting może być postrzegany jako **addytywny model** dla **regresji logistycznej**.

Notacja

- Niech $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$ oznacza zbiór przykładów uczących, gdzie $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ to wektor cech opisujący i-ty przykład (obiekt), a $y_i \in \{-1, 1\}$ to etykieta klasy skojarzona z obiektem.
- Rundy (iteracje) procedury boostingu będą numerowane jako $t = 1, 2, \ldots, T$.
- Niech w_i oznacza wagę i-tego przykładu uczącego w aktualnej rundzie.
- W razie jawnej potrzeby wskazania numeru rundu t przy wadze będziemy pisali $w_{i,t}$.
- Na wagi możemy patrzeć jak na rozkład prawdopodobieństwa określony nad przykładami, tj. w każdej rundzie mamy: $w_i \geqslant 0$ i $\sum_{i=1}^{m} w_i = 1$.
- Niech f_t oznacza cząstkowy (słaby) klasyfikator powstały w rundzie t.
- Niech F oznacza zbiorowy klasyfikator — zwykła lub ważona suma słabych klasyfikatorów.
- Podczas obserwacji przebiegu algorytmu F_t (z indeksem) będzie oznaczało aktualny stan zbiorowego klasyfikatora w rundzie t, tj. zwykłą lub ważoną sumę $f_1, f_2, \ldots, f_t$. W tym sensie oznaczenie F_T jest tożsame z F.
- Niech $[s]$ oznacza funkcję wskaźnikową zwracającą 1, jeżeli zdanie s jest prawdziwe i 0 w.p.r.

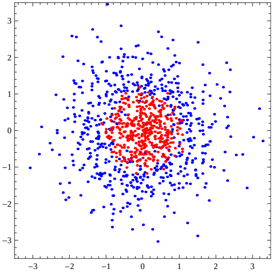
Spis treści

Algorytm *Discrete AdaBoost*

Algorytm DiscreteAdaBoost($\mathcal{D}$)
Rozpocznij od jednostajnego rozkładu wag: $w_i := 1/m$, $i = 1, \ldots, m$.
Dla $t := 1, \ldots, T$ **powtarzaj**
Naucz klasyfikator $f_t(\mathbf{x}) \in \{-1, 1\}$ na danych uczących używając wag w_i.
Oblicz błąd uczący:
$\epsilon_t := \sum_{i=1}^{m} w_i [f(\mathbf{x}_i) \neq y_i]$.
Oblicz wagę klasyfikatora
$\alpha_t := \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.
Aktualizuj wagi przykładów wg:
$Z_t := \sum_{i=1}^{m} w_i e^{-\alpha_t f_t(\mathbf{x}_i) y_i}$.
$w_i := w_i e^{-\alpha_t f_t(\mathbf{x}_i) y_i} / Z_t, \qquad i = 1, \ldots, m.$
Zwróć zbiorowy klasyfikator $F(\mathbf{x}) := \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})$ z decyzją obliczaną jako $\operatorname{sgn} F(\mathbf{x})$.

AdaBoost — zbiór danych do testów

- Dane generowane wg rozkładu łącznego $P(\mathbf{x}, y) = p(\mathbf{x})P(y|\mathbf{x})$.
- Funkcja gęstości $p(\mathbf{x})$ dla dwuwymiarowego rozkładu normalnego $N^2(0, 1)$.
- Rozkład warunkowy: $P(y|\mathbf{x}) = 1/\left(1 + e^{y\beta(x_1{}^2+x_2{}^2-r^2)}\right)$, gdzie $r = 1$, $\beta = 5$.
- Zbiór uczący ($m = 1000$ przykładów):



- **Błąd prawdziwy** dla pewnego klasyfikatora $c(\mathbf{x}) \in \{-1, 1\}$:

$$\mathrm{err}_P(c) = \int_{-\infty}^{\infty} \sum_{y\in\{-1,1\}} [c(\mathbf{x}) \neq y]\, P(y|x)p(\mathbf{x})\, \mathbf{dx}. \qquad (37)$$

- Błąd prawdziwy dla klasyfikatora $c(\mathbf{x}) = 2[x_1{}^2 + x_2{}^2 - r^2 < 0] - 1$ wynosi ≈ 0.084442.

AdaBoost + decision stump (1)

- Słabe klasyfikatory działają w oparciu o jeden wybrany atrybut (cechę) i wykonują progową decyzję:

$$f_t(\mathbf{x}; j, v, d) = \begin{cases} 1, & \text{dla } d(x_j - v) > 0; \\ -1, & \text{w.p.r.}; \end{cases} \tag{38}$$

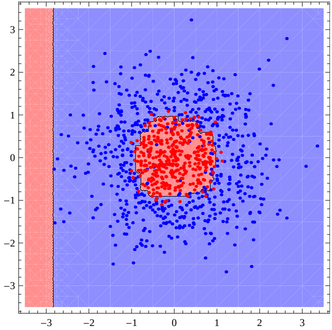
  gdzie $j \in \{1, \ldots, n\}$ to numer cechy, $v \in \mathbb{R}$ wartość progu, a $d \in \{-1, 1\}$ kierunek decyzji.
- Wybór trójki (atrybut, próg, kierunek) zwykle odbywa się wg reguły minimalnego błędu uczącego powstałego w wyniku podziału[3]:

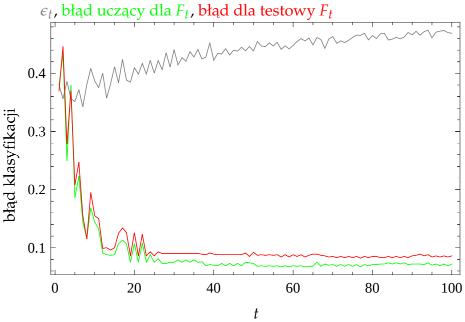
$$(j^*, v^*, d^*) = \arg\min_{(j,v,d)} \sum_{i=1}^{m} w_i [f_t(\mathbf{x}_i; j, v, d) \neq y_i]. \tag{39}$$

[3] Możliwe są inne podejścia, np.: minimalna entropia, minimalny indeks Gini'ego, maksymalny przyrost informacji.

AdaBoost + decision stump (2)

- Wynikowa granica decyzyjna zbiorowego klasyfikatora ($T = 100$) i wykres błędów:

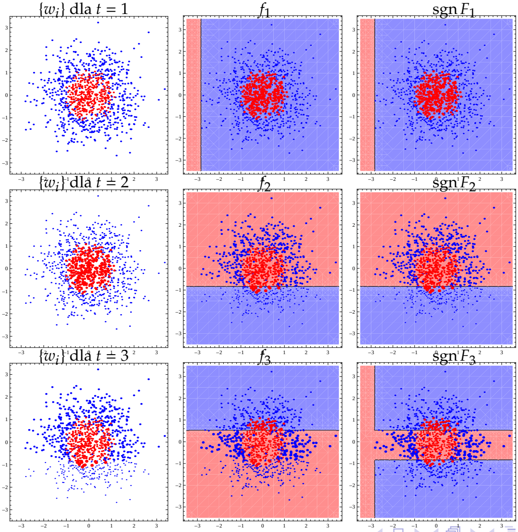




- Błąd prawdziwy: $\text{err}_P(F) \approx 0.092805$.
- Błąd F na próbie testowej (także 1000-elementowej): 0.080.

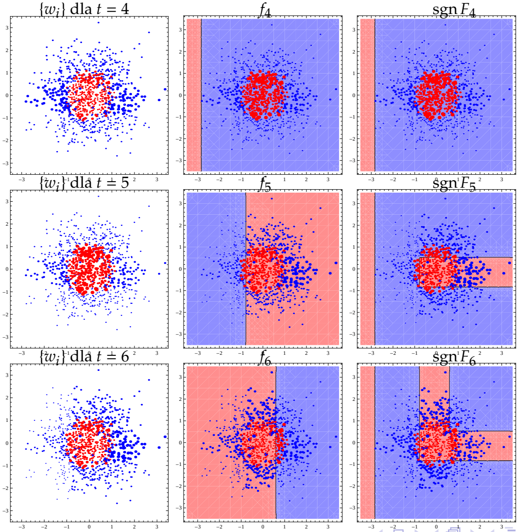
AdaBoost + decision stump (3)

- Przebieg uczenia:



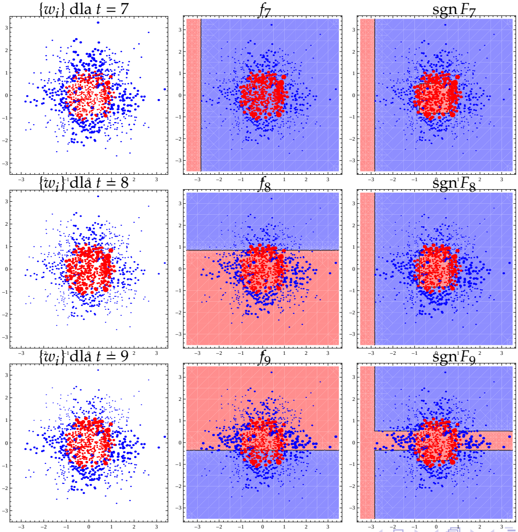
AdaBoost + decision stump (4)

- Przebieg uczenia:



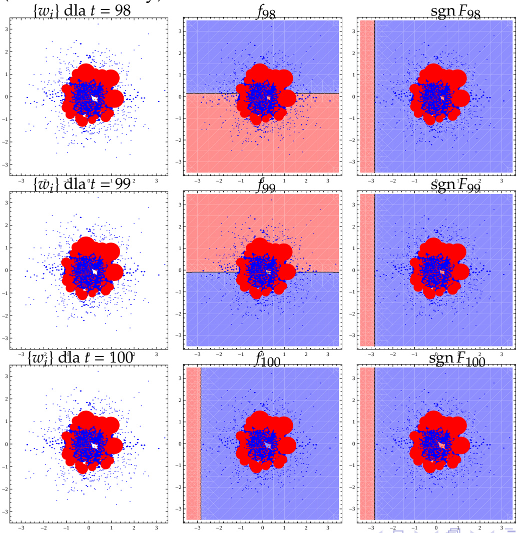
AdaBoost + decision stump (5)

- Przebieg uczenia:



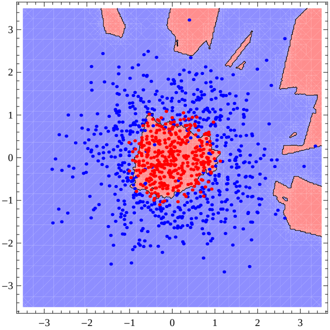
AdaBoost + decision stump (6)

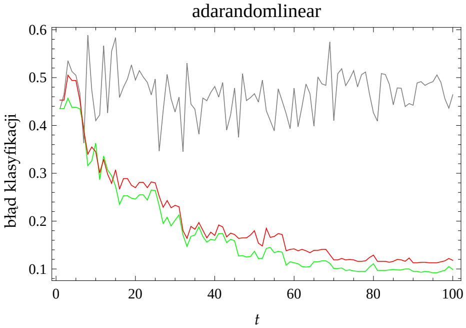
- Przebieg uczenia (końcowe rundy):



AdaBoost + losowe linie (1)

- Słabe klasyfikatory: $f_t(\mathbf{x};\mathbf{c}) = 2[c_0 + c_1x_1 + c_2x_2 > 0] - 1$ o losowych współczynnikach $\mathbf{c} \in [-1, 1]$.
- Wynikowa granica decyzyjna zbiorowego klasyfikatora ($T = 100$) i wykres błędów:

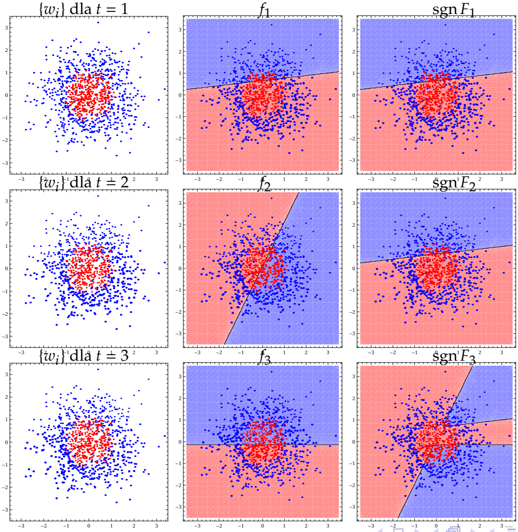




- Błąd prawdziwy: $\mathrm{err}_P(F) \approx 0.118755$.
- Błąd F na próbie testowej (także 1000-elementowej): 0.118.

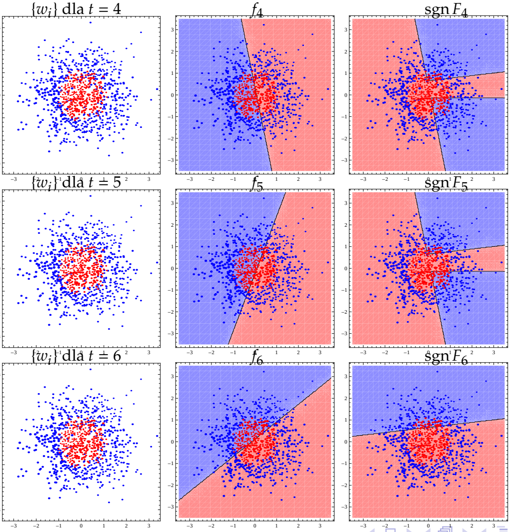
AdaBoost + losowe linie (2)

- Przebieg uczenia:



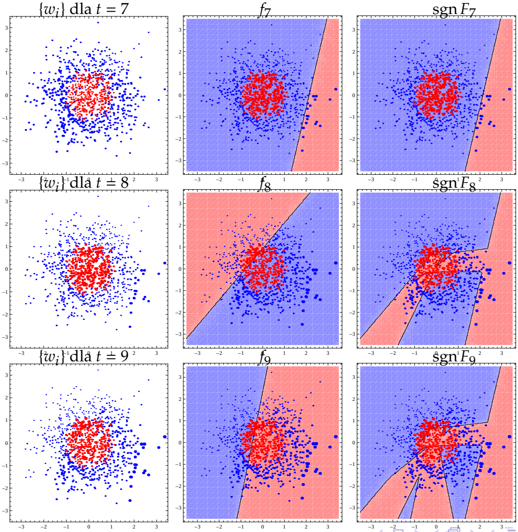
AdaBoost + losowe linie (3)

- Przebieg uczenia:



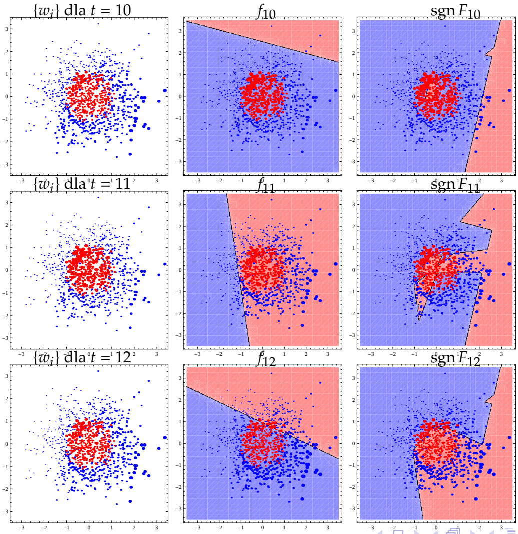
AdaBoost + losowe linie (4)

- Przebieg uczenia:



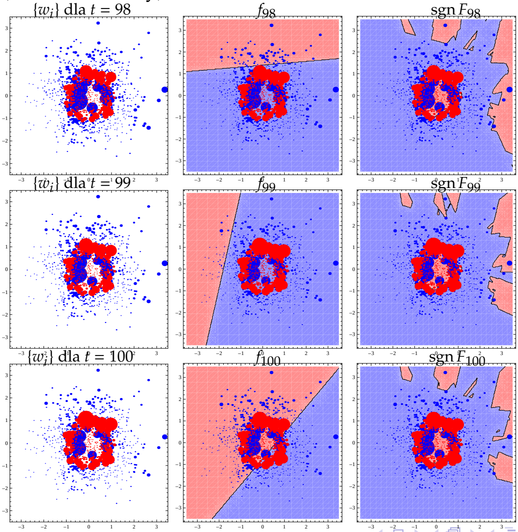
AdaBoost + losowe linie (5)

- Przebieg uczenia:



AdaBoost + losowe linie (6)

- Przebieg uczenia (końcowe rundy):



AdaBoost — uwagi końcowe

- AdaBoost w połączeniu z popularnymi wariantami słabych klasyfikatorów:
  - AdaBoost + decision stump,
  - AdaBoost + drzewka decyzyjne,
  - AdaBoost + klasyfikator liniowy (np. SVM),
  - AdaBoost + naiwny Bayes.
- Wariant "AdaBoost + decision stump" często utożsamiany z ViolaJonesAdaBoost.
- W pojedynczej rundzie **wybór słabego klasyfikatora** (krok 4) może odbywać się teoretycznie wg dowolnego obranego kryterium błędu.
- Zwyczajowo jednak rozważa dwie możliwości — **minimalizację błędu klasyfikacji**: $\arg\min_{f_t} \sum_{i=1}^{m} w_i [f_t(\mathbf{x}_i) \neq y_i]$ lub **minimalizację kryterium wykładniczego**: $\arg\min_{f_t} \sum_{i=1}^{m} w_i e^{-\alpha_t f_t(\mathbf{x}_i) y_i}$.
- Wybór wagi klasyfikatora α_t jest motywowany kryterium wykładniczym Z_t.
- Jeżeli dla pewnego słabego klasyfikatora mamy $\epsilon_t > 1/2$, to waga α_t "zaneguje" jego odpowiedź na przeciwną.

AdaBoost — własności (ćwiczenia)

Pokazać że:

1. wybór $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ minimalizuje kryterium wykładnicze Z_t;
2. Z_t jest równe ilorazowi sum wykładniczych z dwóch kolejnych rund:

$$\sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{t} \alpha_j f_j(\mathbf{x}_i)} \Big/ \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{t-1} \alpha_j f_j(\mathbf{x}_i)}; \tag{40}$$

3. błąd uczący zbiorowego klasyfikatora F jest ograniczony z góry przez iloczyn Z_t, tj.:

$$\frac{1}{m} \sum_{i=1}^{m} [\operatorname{sgn} F(\mathbf{x}_i) \neq y_i] \leqslant \prod_{t=1}^{T} Z_t; \tag{41}$$

4. . . . i tym samym jest niewiększy niż:

$$2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}. \tag{42}$$

Spis treści

RealBoost — uwagi wstępne

- **Pomysł** w pracy: *"Improved boosting using confidence-rated predictions"* **(Schapire & Singer, 1999)**.
- Pełna nazwa: *Real AdaBoost* zwyczajowo skracana do *RealBoost*.
- Sedno: **słabe klasyfikatory** są **rzeczywistoliczbowe** (a nie binarne), tj. $f_t(\mathbf{x}) \in \mathbb{R}$.
- Odpowiedź słabego klasyfikatora jest zwykle ustalana jako przybliżenie połowy przekształcenia **logit**:

$$f_t(\mathbf{x}) = \frac{1}{2} \ln \frac{\widehat{P}_w(y = 1|\mathbf{x})}{\widehat{P}_w(y = -1|\mathbf{x})}, \tag{43}$$

  gdzie $\widehat{P}_w(y = \pm 1|\mathbf{x})$ stanowi oszacowanie rozkładu klas warunkowanego na $\mathbf{x}$ z wykorzystaniem aktualnych wag w_i.
- Na przykład dla "decision stump" rozważając klasyfikator $f(\mathbf{x}; j, v, d)$ mamy:

$$\widehat{P}_w(y = \pm 1|\mathbf{x}; j, v, d) = \begin{cases} \sum\limits_{\{i\colon d(x_{ij}-v)\leqslant 0,\ y_i=\pm 1\}} w_i, & \text{dla } d(x_{ij} - v) \leqslant 0; \\ \sum\limits_{\{i\colon d(x_{ij}-v)>0,\ y_i=\pm 1\}} w_i, & \text{dla } d(x_{ij} - v) > 0. \end{cases} \tag{44}$$

- W przypadku zastosowania drzew decyzyjnych (jako słabe klasyfikatory), każdy terminal wyznacza swoje oszacowanie $\widehat{P}_w(y = \pm 1|\mathbf{x})$.

RealBoost — uwagi wstępne

- **Zbiorowy klasyfikator** ma postać $F(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x})$ z decyzją obliczaną wg $\operatorname{sgn} F(\mathbf{x})$.
- **Rezygnuje się ze współczynników ważących słabe klasyfikatory** — α_t (które miały miejsce w *Discrete AdaBoost*).
- Mechanizm ważenia słabych klasyfikatorów jest niejako wpleciony w same odpowiedzi rzeczywistoliczbowe.
- Można pokazać, że wyrażenie $1/2 \ln \left(\widehat{P}_w(y=1|\mathbf{x}) \middle/ \widehat{P}_w(y=-1|\mathbf{x})\right)$ jest rozwiązaniem zadania **minimalizacji kryterium wykładniczego** określonego poprzez rozkład $\{w_i\}$ na zbiorze danych (tj. na konkretnej próbie).
- Analogicznie, można pokazać, że wyrażenie $1/2 \ln \left(P(y=1|\mathbf{x}) \middle/ P(y=-1|\mathbf{x})\right)$ jest rozwiązaniem zadania minimalizacji kryterium wykładniczego określonego poprzez prawdziwy ale nieznany rozkład łączny generujący dane tj. $P(\mathbf{x}, y) = p(\mathbf{x})P(y|\mathbf{x})$.
- Można zauważyć silne podobieństwa pomiędzy algorytmem *RealBoost* a techniką **regresji logistycznej**.

Algorytm *RealBoost*

Algorytm $\text{RealBoost}(\mathcal{D})$

Rozpocznij od jednostajnego rozkładu wag: $w_i := 1/m,\, i = 1, \ldots, m$.

Dla $t := 1, \ldots, T$ **powtarzaj**

Naucz klasyfikator $f_t(\mathbf{x}) \in \mathbb{R}$ na danych uczących używając wag w_i, tak aby f_t minimalizowało *kryterium wykładnicze* $\sum_{i=1}^{m} w_i e^{-f_t(\mathbf{x}_i) y_i}$ lub równoważnie aby f_t było przybliżeniem połowy przekształcenia logit:

$$f_t(\mathbf{x}) := 1/2 \ln \left(\widehat{P}_w(y = 1|\mathbf{x}) \middle/ \widehat{P}_w(y = -1|\mathbf{x}) \right).$$

Aktualizuj wagi przykładów wg:

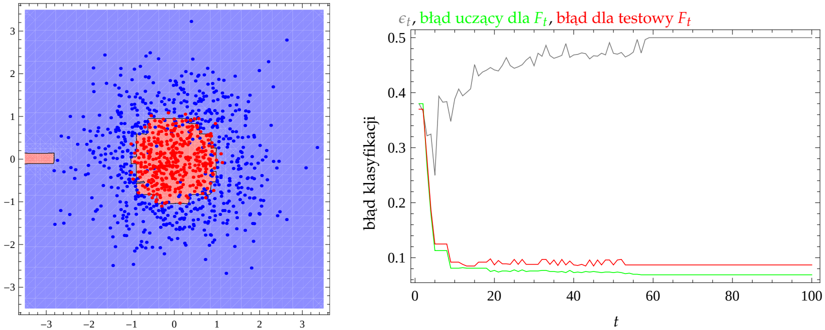
$$Z_t := \sum_{i=1}^{m} w_i e^{-f_t(\mathbf{x}_i) y_i}.$$

$$w_i := w_i e^{-f_t(\mathbf{x}_i) y_i} / Z_t, \qquad i = 1, \ldots, m.$$

Zwróć zbiorowy klasyfikator $F(\mathbf{x}) := \sum_{t=1}^{T} f_t(\mathbf{x})$ z decyzją obliczaną jako $\operatorname{sgn} F(\mathbf{x})$.

RealBoost + decision stump

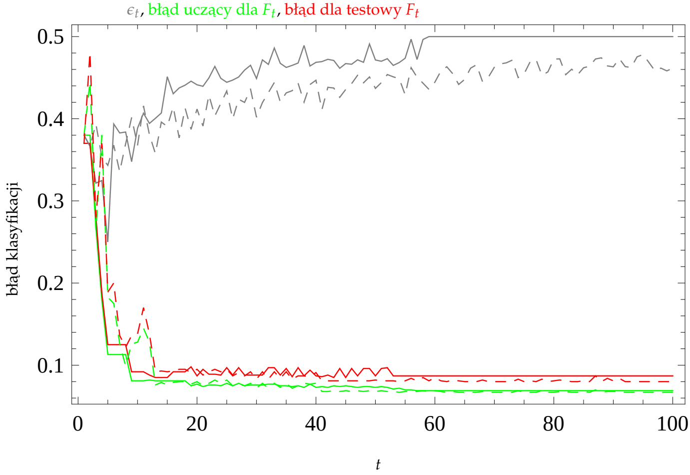
- Wynikowa granica decyzyjna zbiorowego klasyfikatora ($T = 100$) i wykres błędów:



- Błąd prawdziwy: $\mathrm{err}_P(F) \approx 0.092465$.
- Błąd F na próbie testowej (także 1000-elementowej): 0.087.

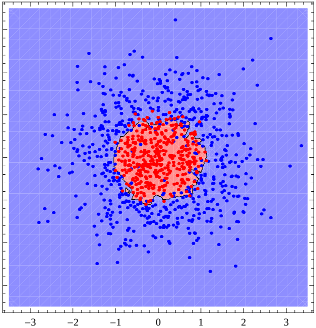
(RealBoost vs AdaBoost) + decision stump

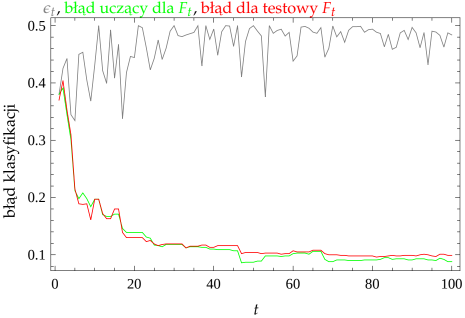
- Cały przebieg uczenia (*krzywe przerywane dot. AdaBoost*):



RealBoost + losowe linie

- Wynikowa granica decyzyjna zbiorowego klasyfikatora ($T = 100$) i wykres błędów:

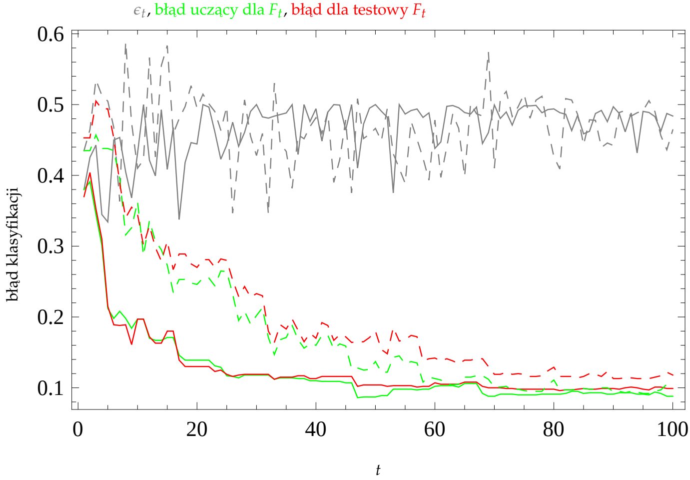




- Błąd prawdziwy: $\mathrm{err}_P(F) \approx 0.0975283$.
- Błąd F na próbie testowej (także 1000-elementowej): 0.099.

(RealBoost vs AdaBoost) + losowe linie

- Cały przebieg uczenia (*krzywe przerywane dot. AdaBoost*):



Spis treści

RealBoost + normals

- Słabe klasyfikatory działają w oparciu o jeden wybrany atrybut (cechę).
- Wykonywane są **aproksymacje rozkładów atrybutów pod warunkiem klas** $p(x_j|y=\pm 1)$ **przez rozkłady normalne**: $\widehat{p}_w(x_j|y=\pm 1) = 1/\sqrt{2\pi\sigma^2_{j\pm}}\, e^{-(x_j-\mu_{j\pm})^2/(2\sigma^2_{j\pm})}$.
- Średnie i wariancje oblicza się jako:

$$\mu_{j-} = \sum_{\{i:\, y_i=-1\}}^{m} w_i x_{ij} \Big/ \sum_{\{i:\, y_i=-1\}}^{m} w_i, \qquad \mu_{j+} = \sum_{\{i:\, y_i=1\}}^{m} w_i x_{ij} \Big/ \sum_{\{i:\, y_i=1\}}^{m} w_i, \tag{45}$$

$$\sigma^2_{j-} = \sum_{\{i:\, y_i=-1\}}^{m} w_i x^2_{ij} \Big/ \sum_{\{i:\, y_i=-1\}}^{m} w_i - \mu^2_{j-}, \qquad \sigma^2_{j+} = \sum_{\{i:\, y_i=-1\}}^{m} w_i x^2_{ij} \Big/ \sum_{\{i:\, y_i=1\}}^{m} w_i - \mu^2_{j+}. \tag{46}$$

- Poprzez twierdzenie Bayesa odpowiedź słabego klasyfikatora jest obliczana jako:

$$f_t(\mathbf{x}; j^*) = \frac{1}{2} \ln \frac{\widehat{p}_w(x_{j^*}|y=1)\widehat{P}_w(y=1)}{\widehat{p}_w(x_{j^*}|y=-1)\widehat{P}_w(y=-1)} \tag{47}$$

$$= \frac{1}{2}\left(\frac{(x_{j^*}-\mu_{j^*-})^2}{2\sigma^2_{j^*-}} - \frac{(x_{j^*}-\mu_{j^*+})^2}{2\sigma^2_{j^*+}} + \ln\frac{\sigma_{j^*-}}{\sigma_{j^*+}} + \ln\frac{\widehat{P}_w(y{=}1)}{\widehat{P}_w(y{=}-1)} \right), \tag{48}$$

gdzie $\widehat{P}_w(y=\pm 1) = \sum_{\{i:\, y_i=\pm 1\}} w_i$ są aktualnymi oszacowaniami prawdopodobieństw klas (a priori) a j^* indeksem cechy, dla której kryterium wykładnicze jest najmniejsze.

RealBoost + bins

- Pomysł (Rasolzadeh et al., 2006) podobny do RealBoost + normals, ale **rozkłady warunkowe przybliżane przez funkcje kawałkami stałe** (implementowane za pomocą koszyków).
- Niech $[a_1, a_2]$ reprezentuje przedział pewnej cechy, a B żądaną liczbę koszyków o równej szerokości.
- Indeks koszyka $\beta(x) \in \{1, \ldots, B\}$, do którego należy x obliczamy jako:

$$\beta(x) = \begin{cases} \lceil B(x - a_1)/(a_2 - a_1) \rceil & \text{dla } a_1 < x \leq a_2; \\ 1 & \text{dla } x \leq a_1; \\ B & \text{dla } a_2 < x. \end{cases} \tag{49}$$

- Niech $\widehat{P}_w(y{=}-1, j \text{ jest w } b) = \sum_{\{i:\ y_i=-1,\ \beta(x_{ij})=b\}} w_i$ oznacza szacowane prawdopodobieństwo zdarzenia, że przykład jest negatywny a jego j-ta cecha należy do kosza b.
- Odpowiedź słabego klasyfikatora (używającego j^*-tej cechy) obliczamy jako:

$$f_t(\mathbf{x}; j^*) = \frac{1}{2} \ln \frac{\widehat{P}_w\left(y = 1, j^* \text{ jest w } \beta(x_{j^*})\right)}{\widehat{P}_w\left(y = -1, j^* \text{ jest w } \beta(x_{j^*})\right)}. \tag{50}$$

RealBoost + drzewa decyzyjne

- Pomysł oparty na znanym **algorytmie CART** (Breiman, Friedman, Olshen, & Stone, 1984).
- Praktyczne eksperymenty pokazują, że grupa płytkich drzew decyzyjnych (otrzymana poprzez boosting) pracuje zwykle lepiej niż jedno głębokie drzewo.
- Algorytm buduje **drzewo binarne rekurencyjnie**, w każdym kroku dzieląc fragment dziedziny **cięciem prostopadłym do pewnej osi** (cechy).
- Wybór najlepszego podziału (j, v) — pary: (numer cechy, wartość progowa) — odbywa się poprzez **minimalizację oczekiwanej nieczystości potomków**.
- Popularne funkcje nieczystości to: *indeks Gini'ego*, *entropia*, *ujemny przyrost informacji*.
- Terminale drzewa zwracają odpowiedzi rzeczywistoliczbowe — połowa przekształcenia logit.
- W powyższym sensie są także **przybliżeniami kawałkami stałymi**, tyle że **rozkładów łącznych wielu zmiennych** (a nie jednej).

RealBoost + drzewa decyzyjne

- Na pewnym etapie rekurencji podziałowej, tj. dla pewnego węzła w drzewie, niech $\{i\}$ oznacza zbiór tylko tych indeksów przykładów uczących, które wpadają w dany węzeł.
- Dla rozważanej na rozcięcie pary (j, v) interesują nas wielkości:

$$
\begin{aligned}
W(L) &= \sum_{\{i\colon x_{ij}<v\}} w_i, & W(y{=}{-1}, L) &= \sum_{\{i\colon x_{ij}<v,\ y_i=-1\}} w_i, & W(y{=}1, L) &= \sum_{\{i\colon x_{ij}<v,\ y_i=1\}} w_i,\\
W(R) &= \sum_{\{i\colon x_{ij}\geq v\}} w_i, & W(y{=}{-1}, R) &= \sum_{\{i\colon x_{ij}\geq v,\ y_i=-1\}} w_i, & W(y{=}1, R) &= \sum_{\{i\colon x_{ij}\geq v,\ y_i=1\}} w_i,
\end{aligned}
\tag{51}
$$

  gdzie L i R oznaczają odpowiednio lewą i prawą część powstałe w wyniku podziału.
- Prawdopodobieństwa wynikające z powyższych to:

$$
\begin{aligned}
\widehat{P}_w(L) &= W(L)/\left(W(L)+W(R)\right), & \widehat{P}_w(R) &= W(R)/\left(W(L)+W(R)\right),\\
\widehat{P}_w(y{=}{-1}|L) &= W(y{=}{-1}, L)/W(L), & \widehat{P}_w(y{=}1|L) &= W(y{=}1, L)/W(L),\\
\widehat{P}_w(y{=}{-1}|R) &= W(y{=}{-1}, R)/W(R), & \widehat{P}_w(y{=}1|R) &= W(y{=}1, R)/W(R).
\end{aligned}
\tag{52}
$$

- Oczekiwaną nieczystość potomków np. wg indeksu Gini'ego oblicza się wtedy jako:

$$
\widehat{P}_w(L)\left(1-\widehat{P}^2_w(y{=}{-1}|L)-\widehat{P}^2_w(y{=}1|L)\right)+\widehat{P}_w(R)\left(1-\widehat{P}^2_w(y{=}{-1}|R)-\widehat{P}^2_w(y{=}1|R)\right).
\tag{53}
$$

- Każdy terminal zwraca odpowiedź: $1/2\ln(\sum_{\{i\colon y_i=1\}} w_i / \sum_{\{i\colon y_i=-1\}} w_i)$.

RealBoost — własności (ćwiczenia)

Pokazać że:

1. wartość oczekiwana kryterium wykładniczego:

$$\mathbb{E}_P\left(e^{-F(\mathbf{x})y}\right) = \int_{\mathbf{x}} \sum\nolimits_{y\in\{-1,1\}} e^{-F(\mathbf{x})y} P(y|\mathbf{x})p(\mathbf{x})\,\mathbf{dx} \tag{54}$$

   (oczekiwana wzięta względem prawdziwego rozkładu łącznego P) osiąga minimum dla:

$$F(\mathbf{x}) = \frac{1}{2}\ln\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}; \tag{55}$$

2. minimalizując kryterium wykładnicze

$$Z_t = \sum_{i=1}^{m} w_{i,t} e^{-f_t(\mathbf{x}_i)y_i} \tag{56}$$

   w sposób zachłanny w każdej rundzie boostingu, jednocześnie minimalizujemy to kryterium dla zbiorowego klasyfikatora tj.:

$$\frac{1}{m}\sum_{i=1}^{m} e^{-F(\mathbf{x}_i)y_i}. \tag{57}$$

Spis treści

Regresja logistyczna

- **Metoda rozwiązywania zadania klasyfikacji za pomocą podejścia znanego z regresji liniowej.**
- Chcemy zamodelować prawdopodobieństwo warunkowe $P(y = 1|\mathbf{x})$ wykorzystując w jakiś sposób formę liniową $a_0 + a_1x_1 + \cdots + a_nx_n$.
- **Problem**: prawdopodobieństwa są ograniczone do $[0, 1]$, a wyrażenie $a_0 + a_1x_1 + \cdots + a_nx_n$ nie jest ograniczone.
- **Sztuczka**: zamiast przybliżać prawdopodobieństwo, można przybliżać **logarytm z ilorazu prawdopodobieństw** (ang. *logarithmic odds ratio*):

$$\ln \frac{P(y = 1|\mathbf{x})}{1 - P(y = 1|\mathbf{x})}. \tag{58}$$

- Rozwiązując równanie

$$a_0 + a_1x_1 + \cdots + a_nx_n = \ln \frac{P(y = 1|\mathbf{x})}{1 - P(y = 1|\mathbf{x})} \tag{59}$$

ze względu na prawdopodobieństwo $P(y = 1|\mathbf{x})$ otrzymujemy

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(a_0 + a_1x_1 + \cdots + a_nx_n)}} \tag{60}$$

(co przypomina **sigmoidę**).

Regresja logistyczna

- Dla rozwiązywania regresji logistycznej (poszukiwania $a_0, \ldots, a_n$) wygodne jest przyjęcie $y_i \in \{0, 1\}$, zamiast $y_i \in \{-1, 1\}$.
- Dla uproszczenia notacji oznaczmy $P(y = 1|\mathbf{x})$ jako $p_i(\mathbf{x}_i)$.
- Tworzymy **funkcję wiarygodności** (ang. *likelihood function*):

$$L = \prod_{i=1}^{m} p(\mathbf{x}_i)^{y_i} \left(1 - p(\mathbf{x}_i)\right)^{1-y_i}. \tag{61}$$

- Maksimum tej funkcji ze względu na $a_0, \ldots, a_n$ jest w tym samym miejscu co maksimum **logarytmu funkcji wiarygodności** (ang. *log-likelihood*):

$$\begin{aligned}\ln L &= \sum_{i=1}^{m} \Big(y_i \ln p(\mathbf{x}_i) + (1 - y_i) \ln\left(1 - p(\mathbf{x}_i)\right) \Big) \\ &\vdots \\ &= \sum_{i=1}^{m} \Big(y_i (a_0 + a_1 x_{i1} + \cdots a_n x_{in}) - \ln\left(1 + e^{a_0 + a_1 x_{i1} + \cdots a_n x_{in}}\right) \Big).\end{aligned} \tag{62}$$

Związek z RealBoost

- Rozważmy **wartość oczekiwaną kryterium wykładniczego** wziętą ze względu na prawdziwy rozkład P, z którego czerpane są pary $(\mathbf{x}, y)$:

$$Q_P(F) = \mathbb{E}_P\left(e^{-yF(\mathbf{x})}\right) = \int_{\mathbf{x}} \sum_{y\in\{-1,1\}} e^{-yF(\mathbf{x})} p(\mathbf{x},y)\,\mathbf{dx}$$
$$= \int_{\mathbf{x}} \left(P(y{=}-1|\mathbf{x})e^{F(\mathbf{x})} + P(y{=}1|\mathbf{x})e^{-F(\mathbf{x})}\right) p(\mathbf{x})\,\mathbf{dx}. \qquad (63)$$

- Wiemy, że zażądanie $\partial Q_P(F)/\partial F = 0$ prowadzi do optymalnego rozwiązania postaci:

$$F^*(\mathbf{x}) = 1/2 \ln\left(P(y{=}1|\mathbf{x})/P(y{=}-1|\mathbf{x})\right) \qquad (64)$$

  (właściwie wystarczy zminimalizować wewnętrzną oczekiwaną w (63) względem rozkładu warunkowego $P(y = \pm 1|\mathbf{x})$).
- Widać, że F^* to **połowa przekształcenia logit** typowego dla regresji logistycznej.
- Jeżeli algorytm uczący byłby w stanie w jakiś sposób znaleźć natychmiast optymalną funkcję F^*, to **procedura boostingu mogłaby zostać zatrzymana już po jednej rundzie**.
- W praktyce, **słabe klasyfikatory są bardzo zgrubnymi przybliżeniami** F^*, stąd też potrzeba wielu rund.

Związek z RealBoost

- Rozwiązując (64) ze względu na prawdopodobieństwo $P(y=1|\mathbf{x})$ otrzymujemy formę **sigmoidy**:

$$P(y=1|\mathbf{x}) = e^{2F^*(\mathbf{x})}\Big/\left(1+e^{2F^*(\mathbf{x})}\right) = 1\Big/\left(1+e^{-2F^*(\mathbf{x})}\right), \tag{65}$$

  — podobieństwo do regresji logistycznej z dokładnością do współczynnika 2 w wykładniku.

- **Regresja logistyczna przybliża** F^* **modelem liniowym**:

$$F^*(\mathbf{x}) \approx a_0 + a_1 x_1 + \cdots + a_n x_n. \tag{66}$$

- **RealBoost przybliża** F^* **liniową kombinacją słabych klasyfikatorów**:

$$F^*(\mathbf{x}) \approx f_1(\mathbf{x}) + \cdots + f_T(\mathbf{x}), \tag{67}$$

  a więc dowolnych funkcji (zwykle prostych), potencjalnie funkcji wielu zmiennych.

Związek z RealBoost

- Rozważmy technikę **rezyduów błędów** (ang. *error residuals*) znaną ogólnie z regresji.
- Używając jej, budujemy sekwencyjnie model addytywny, gdzie każdy kolejny fragment aproksymacji *„objaśnia"* pewną część wielkości docelowej i jest od niej odejmowany, tak aby kolejne fragmenty konentrowały się na rezyduach błędu.
- **Schemat reważenia w boostingu pracuje sposób pokrewny do rezyduów błędów**.

Związek z RealBoost

- Przypuśćmy, że mamy częściowy model F i chcielibyśmy go uaktualnić do $F := F + f$.
- Dla wzorów dokonujących reważenia opartych na danych: $Z = \frac{1}{m}\sum_{i=1}^{m} e^{-F(\mathbf{x}_i)y_i}$, $w_i = e^{-F(\mathbf{x}_i)y_i}/Z$, określamy ich populacyjne odpowiedniki (związane z rozkładem P):

$$Z = \int_{\mathbf{x}} \sum_{y\in\{-1,1\}} e^{-F(\mathbf{x})y} p(\mathbf{x},y)\,\mathbf{dx}; \qquad w(\mathbf{x},y) = e^{-F(\mathbf{x})y}/Z. \tag{68}$$

- Z pracuje jako stała normalizująca, ale jednocześnie $Z = Q_P(F)$ — wartość kryterium dotychczasowego modelu.
- Rozważmy wartość kryterium dla $F + f$:

$$\begin{aligned} Q_P(F{+}f) &= \int_{\mathbf{x}} \sum_{y\in\{-1,1\}} e^{-y\left(F(\mathbf{x})+f(\mathbf{x})\right)} p(\mathbf{x},y)\,\mathbf{dx} \\ &= \int_{\mathbf{x}} \sum_{y\in\{-1,1\}} e^{-yf(\mathbf{x})} \underbrace{e^{-yF(\mathbf{x})} p(\mathbf{x},y)/Z}_{w(\mathbf{x},y)}\,\mathbf{dx} \cdot Z{=}Q_w(f)\cdot Q_P(F). \end{aligned} \tag{69}$$

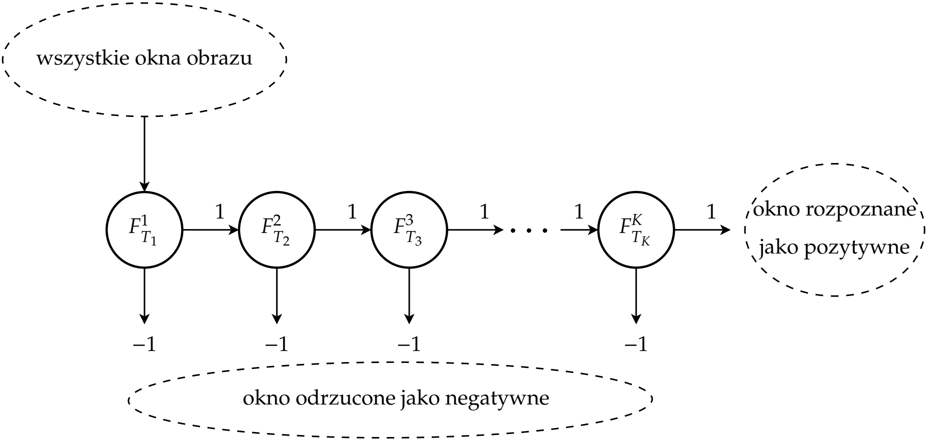
- **Wniosek:** aby zminimalizować $Q_P(F + f)$ wystarcza zachłannie minimalizować $Q_w(f)$; aktualny stan rozkładu w wskazuje, które części wielkości docelowej są już dobrze przybliżone („objaśnione”), a które jeszcze wymagają przybliżenia.

Spis treści

Zarys pomysłu

- Pomysł wykorzystuje obserwację, że **okna negatywne stanowią dominującą większość** wszystkich badanych okien obrazu (np. dla twarzy zwykle $\geqslant 99\%$).
- Warto stworzyć **prostsze klasyfikatory obliczające mniej cech** (i tym samym wydajniesze czasowo) służące **do odrzucania okien negatywnych**.
- Okna, które w trakcie analizy rokują na bycie oknami pozytywnymi, mogą być badane dłużej na podstawie większej liczby cech.
- Uczony jest ciąg zbiorowych klasyfikatorów $(F^1_{T_1}, F^2_{T_2}, \ldots)$, który tworzy się **kaskadę — binarne drzewo decyzyjne zdegenerowane do listy**.
- Rozmiary kolejnych klasyfikatorów w kaskadzie tworzą ciąg niemalejący $T_1 \leqslant T_2 \leqslant \ldots$.
- Kolejne elementy kaskady nazywane są etapami (lub poziomami lub warstwami) (ang. *stages*, *levels*, *layers*).
- **Wskazanie negatywne zwrócone przez dowolny z etapów kaskady przerywa dalsze obliczenia** wzdłuż kaskady i okno jest ostatecznie sklasyfikowane jako negatywne.
- Ostateczne wskazanie pozytywne wymaga przejścia przez wszystkie etapy kaskady (każdy etap musi zwrócić odpowiedź pozytywną).

Schemat kaskady



- Np. w (Viola & Jones, 2004): $K = 32$ poziomy kaskady, $T_1 = 2$, $T_2 = 5$, $T_3 = \cdots = T_5 = 20$, $T_6 = T_7 = 50$, $T_8 = \cdots = T_{12} = 100$, $T_{13} = \cdots = T_{32} = 200$.
- Razem: 4 297 cech, **średnio badanych ≈ 8 cech**.

Wymagania dla całej kaskady i etapów

- W trakcie uczenia należy dobrać **progi decyzyjne** θ_k poszczególnych etapów, gdzie decyzje wg: $\text{sgn}\left(F_{T_k}^k(\mathbf{x}) - \theta_k\right)$, tak aby **każdy etap** miał **bardzo wysoką czułość** (ang. *sensitivity* lub *detection rate*) np. $\geqslant 99.9\%$, a umiarkowanie nieduży **odsetek fałszywych alarmów** (FAR, ang. *false alarm rate*) np. na poziomie $< 50\%$.
- Niech $d_1, d_2, \ldots, d_K$ oznacza ciąg czułości poszczególnych etapów pewnej nauczonej kaskady, zaś $a_1, a_2, \ldots, a_K$ odpowiadający ciąg wartości FAR. Wówczas **wynikowa czułość i wynikowy FAR całej kaskady** wynoszą odpowiednio:

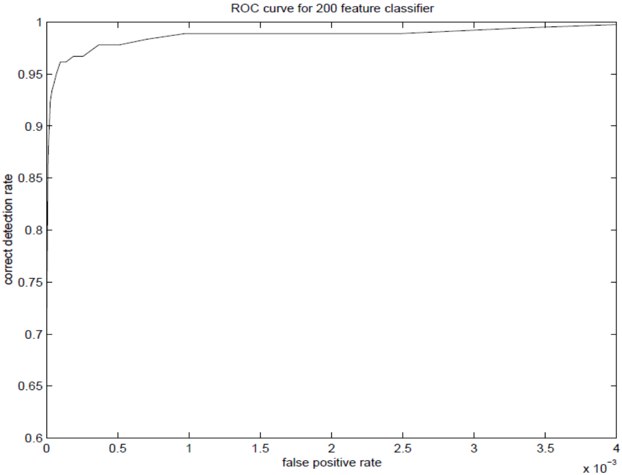
$$D = \prod_{k=1}^{K} d_k, \tag{70}$$

$$A = \prod_{k=1}^{K} a_k. \tag{71}$$

- Mając zadane do osiągnięcia pewne **wymagania dla całej kaskady** (tj. D, A) można wyznaczyć wymagania cząstkowe $d_{\min}$ i $a_{\max}$ dla wszystkich etapów.
- Np. dla $D = 0.98$, $A = 10^{-5}$ i $K = 10$ wystarcza, aby każdy etap osiągnął $d_i \geqslant d_{\min} = 0.998$ (bo $0.998^{10} \geqslant 0.98$) oraz $a_i \leqslant a_{\max} = 0.316$ (bo $0.316^{10} \leqslant 10^{-5}$).

Krzywa ROC

- ROC (ang. *Receiver Operating Characteristic*) — krzywa w układzie (FAR, czułość) opisująca działanie klasyfikatora na danych testowych.
- Każdy punkt na wykresie odpowiada pewnemu progowi decyzyjnemu θ i decyzji obliczanej wg $\mathrm{sgn}\,(F(\mathbf{x}) - \theta)$.
- Przykład ROC z (Viola & Jones, 2004) dla pojedynczego zbiorowego klasyfikatora używającego 200 cech:



Algorytm uczenia kaskady

- Użytkownik nastawia akceptowalne wskaźniki $d_{\min}$ i $a_{\max}$ dla wszystkich etapów.
- Każdy etap kaskady jest uczony w ramach boostingu (np. poprzez AdaBoost lub RealBoost).
- Liczba słabych klasyfikatorów danego etapu jest podnoszona o jeden, aż do momentu gdy etap osiągnie zadane wskaźniki.
- Wskaźniki (czułość, FAR) osiągnięte przez dany etap są mierzone na wydzielonym **zbiorze walidacyjnym**.
- Po dodaniu jednego słabego klasyfikatora korygowany jest próg decyzyjny θ_k danego etapu (zwykle pomniejszany), tak aby osiągnąć zadaną czułość $d_k \geqslant d_{\min}$. W konsekwencji obniża to obserwowany FAR tj. a_k.
- Jeżeli wskazania całościowe dla aktualnej kaskady nie osiągnęły wymaganych D, A, to kaskada jest rozszerzana o kolejny etap.

Uwaga 1: nie jest jasne, czy korygowanie progów decyzyjnych nie osłabia własności uczących i generalizujących kaskady.
Uwaga 2: powyższy algorytm może nie spełniać warunku stopu (zwykle konieczne jest wprowadzenia ograniczenia na liczbę etapów lub sumaryczną liczbę słabych klasyfikatorów).

Algorytm uczenia kaskady

Algorytm TrainCascade($\mathcal{D}, D, A, d_{\min}, a_{\max}, \mathcal{V}$) ▷ $\mathcal{V}$ — zbiór walidacyjny
  Ustal podzbiór przykładów pozytywnych $\mathcal{P}$ oraz negatywnych $\mathcal{N}$ w ramach $\mathcal{D}$.
  $D_0 := 1$, $A_0 := 1$, $k := 0$.
  Dopóki $A_k > A$ **powtarzaj** ▷ A_k — FAR dla pierwszych k etapów kaskady
    $k := k + 1$, $T_k := 0$, $A_k := A_{k-1}$, $F^k := 0$. ▷ dolne indeksy $F^k_{T_k}$ pominięte
    Dopóki $A_k > a_{\max} \cdot A_{k-1}$ **powtarzaj**
      $T_k := T_k + 1$.
      Użyj $\mathcal{P}$ i $\mathcal{N}$ aby nauczyć nowy słaby klasyfikator f, otrzymując: $F^k := F^k + f$.
      Skoryguj próg decyzjny θ_k klasyfikatora F^k, tak aby cała kaskada miała czułość
        $D_k \geqslant d_{\min} \cdot D_{k-1}$ w następujący sposób: $\theta_k := F^k(\mathcal{V}_+)_{\lfloor(1-d_{\min})\cdot\#\mathcal{V}_+\rfloor}$,
        gdzie $F^k(\mathcal{V}_+)$ to posortowany ciąg rzeczywistoliczbowych odpowiedzi
        klasyfikatora F^k na przykładach pozytywnych wśród $\mathcal{V}$. ▷ podnosimy też A_k
      Wykonaj aktualną kaskadę $(F^1, F^2, \ldots, F^k)$ na $\mathcal{V}$ mierząc jej czułość D_k i FAR A_k.
    Jeżeli $A_k > A$ **to**
      $\mathcal{N} := \emptyset$.
      Wykonaj aktualną kaskadę $(F^1, F^2, \ldots, F^k)$ na nowo spróbkowanych oknach
        obrazów negatywnych i włóż do zbioru $\mathcal{N}$ fałszywe alarmy.
    W.p.r.
      Przerwij pętlę.
  Zwróć kaskadę $(F^1, F^2, \ldots, F^k)$.

Bibliografia

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Monterey, CA, USA: Wadsworth & Brooks.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Conference on computer vision and pattern recognition (cvpr'2005)* (Vol. 1, pp. 886–893). San Diego, CA, USA: IEEE.

Freund, Y. (1995). Boosting a Weak Learning Algorithm by Majority. *Informationa and Computation*, *121*(2), 256–285.

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Machine learning: Proceedings of the thirteenth international conference* (pp. 148–156). Morgan Kaufman.

Freund, Y., & Schapire, R. (1997). A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. *Journal of Computer Science and System Sciences*, *55*, 119–139.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *28*(2), 337–407.

Rasolzadeh, B., et al. (2006). Response binning: Improved weak classifiers for boosting. In *Ieee intelligent vehicles symposium* (pp. 344–349).

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*, 1997–227.

Schapire, R., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. In *Conference on computer vision and pattern recognition (cvpr'2001)* (pp. 511–518). IEEE.

Viola, P., & Jones, M. (2004). Robust Real-time Face Detection. *International Journal of Computer Vision*, *57*(2), 137–154.