



AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Detekcja sylwetek ludzkich

Tomasz Kryjak

Wydział EAlIB
Katedra Automatyki i Robotyki

21.05.2018

Wprowadzenie

Co to jest detekcja sylwetek ludzkich?



Do czego można to zastosować?

- ✖ systemy wspomagające kierowcę (zapobieganie kolizjom),
- ✖ systemy automatycznego monitoringu wizyjnego,
- ✖ analiza materiału wideo,
- ✖ monitoring pacjentów,
- ✖ i wszystkie zastosowania w których niezbędne jest określenie czy dany obiekt jest człowiekiem.

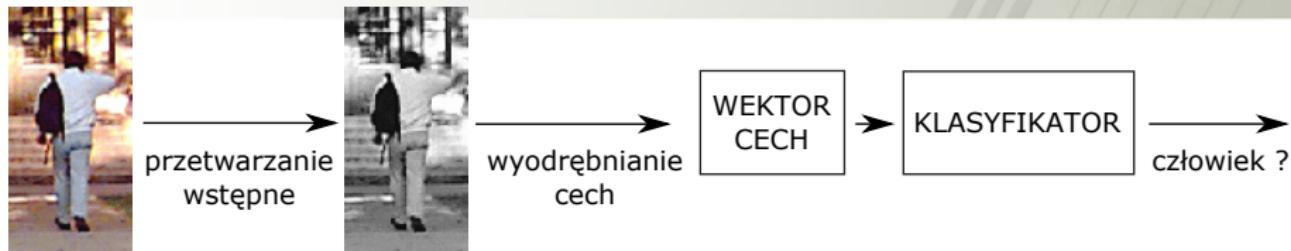
Czy zadanie jest łatwe?

Niestety, jak większość algorytmów wizyjnych, które mają działać w warunkach rzeczywistych detekcja sylwetek ludzkich jest trudna. Wpływa na to szereg różnych czynników:

- ✖ sylwetka może przyjmować różne pozy i ułożenia (np. człowiek stojący, idący),
- ✖ człowiek może być różnie ubrany (kolory, rodzaje),
- ✖ człowiek występuje na różnym tle – może być to zarówno jednorodna ściana, jak i scena o skomplikowanej strukturze,
- ✖ w rzeczywistych scenariuszach zwykle występują przesłonięcia, a także zmiany oświetlenia,
- ✖ zdjęcie sylwetki może zostać zrobione z wielu różnych ujęć (z przodu, z boku, pod kątem).

Detekcja

Zagadnienie detekcji obiektów



De facto najważniejszy jest etap wyodrębniania cech. Klasyfikacji prawie nigdy nie dokonuje się na obrazie wejściowym (na podstawie pikseli), gdyż w takim przypadku wektor cech miałby bardzo duży rozmiar.

Kluczem jest wybór takich cech, które dobrze opiszą dany obiekt i to niezależnie od warunków akwizycji, pozy itp. Oczywiście dobre przetwarzanie wstępne również poprawia skuteczność detekcji.

Detekcja, a klasyfikacja

Detekcja – wykrywanie np. obiektów pierwszoplanowych, ale też ludzi.

Klasyfikacja – przypisywanie wykrytych obiektów do konkretnych klas np. człowiek, samochód, rower. Może być binarna (tylko dwie klasy: ludzi / reszta)

W omawianym przypadku oba zagadnienia są ze sobą połączone.

Uczenie maszynowe - klasyfikacja

Mając dany wektor cech prawie nigdy (oprócz bardzo prostych przypadków) nie buduje się “ręcznie” klasyfikatora tj. nie tworzy się szeregu instrukcji if:

```
if cecha1 > 0.5 && cecha2 < 0.2 then  
    obiekt = człowiek;  
end
```

Stosuje się za to metody automatyczne, które często określa się mianem uczenia maszynowego. Samo zagadnienie jest tematem na osobny kurs akademicki. W tym momencie omówione zostaną tylko podstawowe informacje, niezbędne do posługiwania się tego typu metodami.

Uczenie maszynowe - definicja

Dziedzina, która umożliwia komputerom uczenie się bez właściwego programowania (Arthur Samuel – 1959).

Program **uczy się** z doświadczenia E jakiegoś zagadnienia T, które można ocenić miarą P, jeśli w miarę działania jego zdolność do rozwiązywania zagadnienia T, wg. miary P poprawia się wraz z doświadczeniem E (Tom Mitchell – 1998).

Przykład – filtr antyspamowy:

- ✖ T – klasyfikacja wiadomości e-mail jako spam/nie spam,
- ✖ P – liczba poprawnie sklasyfikowanych e-maili,
- ✖ E – obserwacja jak użytkownik oznacza ręcznie e-maile.

Uczenie maszynowe - przykłady

Analiza wielkich baz danych (ang. *big data*). Przykłady to informacje o ruchu w sieci, dane medyczne, biologiczne itp.

Aplikacje, których nie można zaprogramować *explicitie* (lub jest to trudne/niewygodne). Przykłady: sterowanie autonomicznym helikopterem, rozpoznawanie pisma odręcznego, przetwarzanie języka naturalnego, sztuczne systemy wizyjne.

Jest to troszkę działanie w stylu “nie wiemy jak to opisać” więc używamy metody automatycznej. Przy czym lata doświadczeń pokazują bardzo dobrą skuteczność podejścia. A ostatnim „hitem” są tzw. głębokie sieci neuronowe (Deep Neural Networks).

Uczenie maszynowe - analogia



My ludzie znakomitą większość umiejętności nabywamy poprzez tak rozumiane „uczenie się”. Ktoś nam coś pokazuje (np. jak wygląda bocian) i następnie już jesteśmy w stanie samodzielnie to rozpoznać (nawet w innych okolicznościach). Przy czym nie zastanawiamy się zbytnio nad tym jak to się odbywa. Tj. świadomie nie analizujemy poszczególnych cech anatomicznych tego ptaka.

Uczenie maszynowe - rodzaje

Dla porządku trzeba jeszcze dodać, że uczenie maszynowe występuje w dwóch odmianach:

- ✖ z nauczycielem (ang. *supervised learning*) – dane są poprawne odpowiedzi tj. etykiety danych.
- ✖ bez nauczyciela (ang. *unsupervised learning*) – nie ma poprawnych danych, algorytm sam analizuje dane.

Regresja – przewidywanie ciągłej wartości (np. cen mieszkań w zależności od powierzchni na podstawie kilku przykładów).

Klasyfikacja – przewidywanie czy obiekt należy do danej klasy czy nie (wynik binarny).

Przykłady do analizy (uczenie z nauczycielem czy bez ?):

- ✖ mając daną bazę e-mail oznaczonych jako spam/nie-spam naucz filtr,
- ✖ mając bazę artykułów pogrupuj je tematycznie,
- ✖ mając bazę danych o klientach i ich zakupach dokonaj ich grupowania (np. na segmenty),
- ✖ mając bazę danych o wyników pacjentów z cukrzycą lub nie naucz algorytm wykrywać cukrzycę.

Uczenie maszynowe - jako narzędzie

Uczenie maszynowe to dziedzina informatyki i związane jest z tzw. sztuczną inteligencją.

Natomiast jest to również bardzo użyteczne narzędzie w wielu zastosowaniach praktycznych np. *computer vision*.

I aby go używać nie trzeba dokładnie znać działania poszczególnych metod, a jedynie pewne ogólne wytyczne – por. jeździć samochodem, a wiedzieć jak jest zbudowany.

Najczęściej stosowane metody:

- ✚ regresja (jedna i wiele zmiennych),
- ✚ sztuczne sieci neuronowe (także głębokie),
- ✚ SVM – maszyna wektorów nośnych (ang. *Support Vector Machine*),
- ✚ klasteryzacja (np. K-średnich),
- ✚ drzewa decyzyjne.



Uczenie maszynowe - metodologia (1)

AGH

Na wejściu mamy dane (wektory cech) oznaczone etykietami: człowiek, brak człowieka. W tym momencie etap od obrazu do wektora cech jest mniej istotny.

Aby algorytm poprawnie się nauczył rozpoznawania danych powinno być sporo – my uczymy się przez całe życie. Inaczej mówiąc na podstawie 5 próbek trudno o generalizację.

Zwykle stosuje się podział zbioru wejściowego na uczący (70 %) i testowy (30 %). **Uwaga.** Błądem metodologicznym jest testowanie algorytmu na zbiorze uczącym. Pomija się wtedy bardzo istotny aspekt uczenia maszynowego, jakim jest zdolność do “generalizacji” wiedzy. W rzeczywistości nie można zgromadzić zdjęć wszystkich możliwych ludzi. Poza tym, zwykle wychodzi ok. 100% skuteczności, co jest wartością niemiarodajną.

W bardziej zaawansowanym scenariuszu stosuje się podział na trzy zbiory:

- ✖ uczący (60 %),
- ✖ walidacyjny (20 %),
- ✖ testowy (20 %),

Dodatkowy zbiór walidacyjny wprowadza się celem analizy wpływu różnych modyfikacji algorytmu na działanie klasyfikacji. Zbiór testowy zawsze „imituje” warunki rzeczywiste i jest używany „na końcu”.

Uczenie maszynowe - metodologia (3)

Warto jeszcze pamiętać o możliwości „douczania” klasyfikatora. Przykładowo, jeśli dokonamy analizy wyników to błędnie wykryte przykłady (z działającej on-line aplikacji) możemy dołączyć do zbioru uczącego i jeszcze raz uruchomić procedurę uczenia. Zwykle takie postępowanie przynosi dobre rezultaty. Jest to również sposób na dostosowanie „ogólnego” klasyfikatora do konkretnych warunków (sceny).

Histogram zorientowanych gradientów

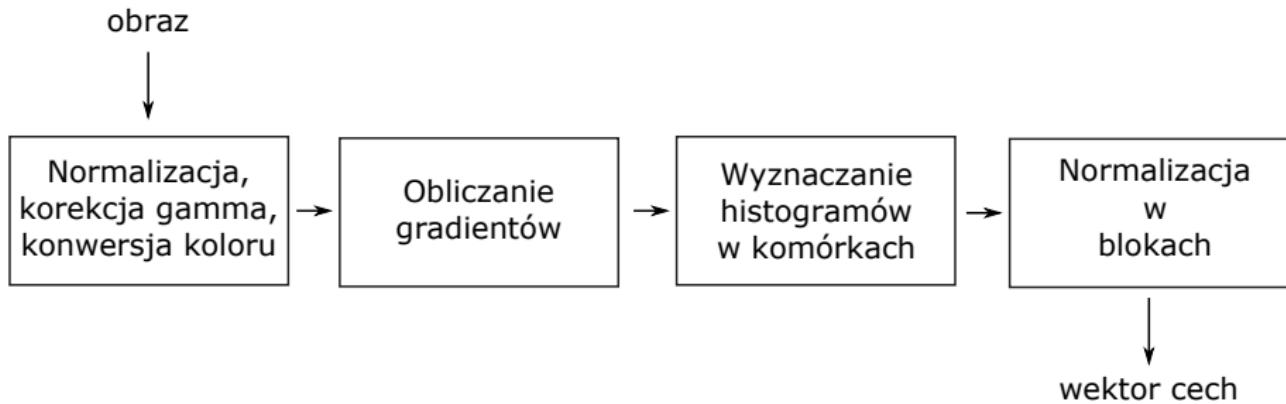
HOG – metoda

Algorytm został zaproponowany w 2005 roku przez Navneet'a Dalal'a i Bill'a Triggs'a. Jest on opisany np. w artykule *Histograms of Oriented Gradients for Human Detection* oraz pracy doktorskiej *Finding People in Images and Videos*.

Metoda ta jest obecnie najczęściej stosowana do detekcji sylwetek ludzkich, przy czym jako deskryptor cech może być wykorzystywana w innych aplikacjach (detekcja rowerów, głów itp.).

Jak sama nazwa wskazuje podstawą są histogramy zorientowanych gradientów. Dlaczego? Autorzy założyli i eksperymentalnie wykazali, że dobrze opisują one kształt obiektów, będąc przy tym niezależnym od czynników takich jak oświetlenie i kolor.

HOG – schemat blokowy



HOG – przetwarzanie wstępne

Autorzy metody sprawdzili kilka przestrzeni barw (odcienie szarości, RGB, CIE Lab) oraz wykorzystywali korekcję gamma. Okazało się, że parametry te nie mają istotnego wypływu na sposób działania algorytmu. Wynika z użytej na późniejszym etapie normalizacji.

Jednakże użycie koloru poprawia nieco wyniki detekcji.

W rozważaniach przyjmiemy przetwarzanie w przestrzeni RGB (tj. wejściowej)

HOG – obliczanie gradientu (1)

Autorzy metody przetestowali szereg metod wyznaczania gradientu – maski [-1,1], [-1, 0, 1], [1, -8, 0, 8, -1] oraz Sobela i Robertsa.

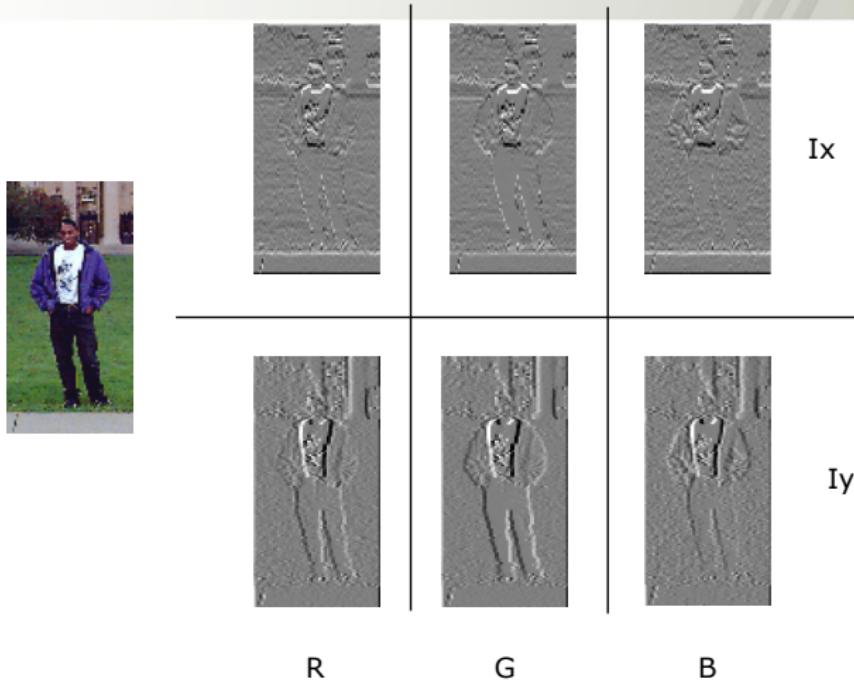
Okazało się, że najlepsze wyniki daje prosta jednowymiarowa maska [-1, 0, 1].

Uwaga. Dla obrazów kolorowych do dalszych obliczeń wybierany jest gradient tej składowej, którego moduł jest największy. Moduł gradientu:

$$G = \sqrt{I_x^2 + I_y^2} \quad (1)$$

gdzie: I_x i I_y to gradient poziomy i pionowy.

HOG – obliczanie gradientu (2)



HOG – obliczanie gradientu (3)



R



G

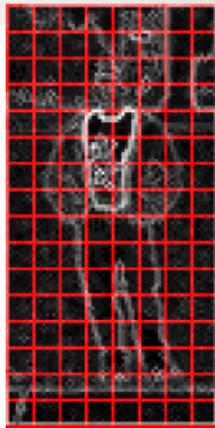


B



$\max(R, G, B)$

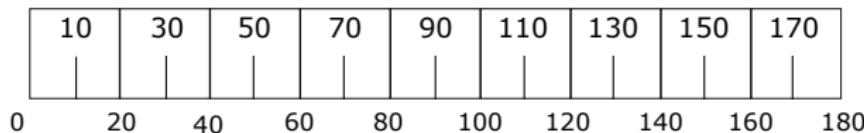
HOG – obliczanie histogramu



Zakładamy, że przetwarzane są obrazy o rozmiarach 64×128 . Dzielone są one na komórki (ang. *cell*) o rozmiarze 8×8 pikseli. Autorzy metody wprawdzie zaproponowali rozmiar 6×6 , ale ew. pogorszenie detekcji jest bardzo nieznaczne, a ułatwienia wynikające ze stosowania potęgi liczby 2 bardzo duże.

HOG – obliczanie histogramu

Dla każdej komórki wyliczany jest histogram kierunków gradientów. Zakłada się, że kąty mogą być w zakresie $[0;180]$, przy czym zakres ten dzieli się (zwykle) na 9 obszarów.



HOG – obliczanie histogramu

Autorzy sprawdzali czy uwzględnienie pełnego zakresu [0;360] poprawia wyniki. Okazało się, że w zadaniu detekcji ludzi nie, ale w przypadku innych np. rowerów, samochodów już tak.

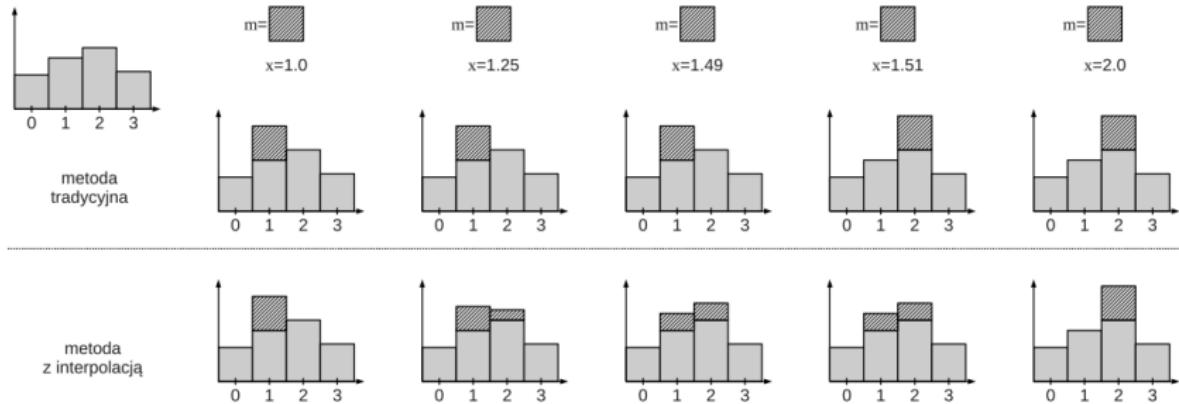
Wymagany kąt oblicza się jako:

$$\Theta = \arctan\left(\frac{I_y}{I_x}\right) \quad (2)$$

Trzeba przewidzieć “specjalne” traktowanie skrajnych przedziałów – zawijanie.

W obliczaniu histogramu stosuje się interpolację liniową. Dlaczego ?

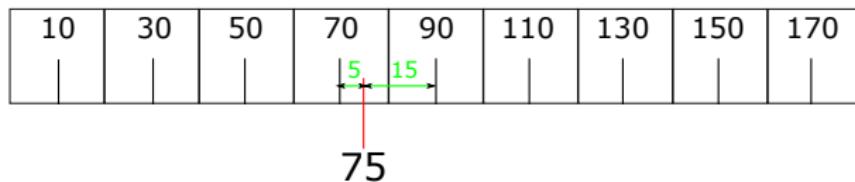
HOG – interpolacja



Rysunek autorstwa Mateusza Komorkiewicza.

HOG – interpolacja

Przykład: Zakładamy, że mamy kąt 75, amplituda wynosi 100. Dany kąt leży pomiędzy przedziałami 70, a 90 (trzeba to określić). Odległość do środka przedziału 70 wynosi 5, a do 90 – 15. Jeśli znormalizujemy to do przedziału [0;1] otrzymamy: 5 – 0.25, 15 – 0.75.



„Podział” amplitudy powinien być następujący:

✗ przedział $70 - 0.75 \cdot 100 = 75$

✗ przedział $90 - 0.25 \cdot 100 = 25$

Można to oczywiście opisać wzorami:

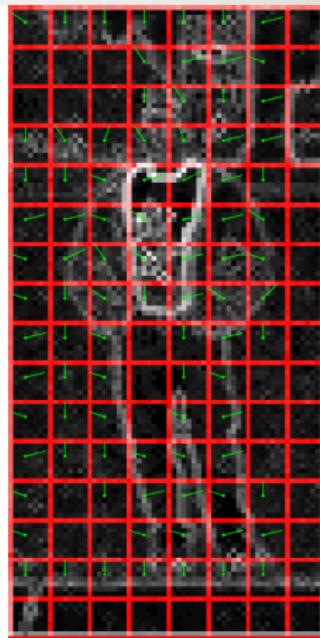
$$h(t_0) = h(t_0) + m\left(1 - \frac{t - t_0}{b}\right) \quad (3)$$

$$h(t_1) = h(t_1) + m\left(\frac{t - t_0}{b}\right) \quad (4)$$

gdzie: t_0 i t_1 to “środki” sąsiednich przedziałów, t – aktualny kąt, h – histogram, b – odległość pomiędzy środkami przedziałów (tu 20).

Finalnie, dla każdej komórki otrzymujemy histogram kierunków gradientów.

HOG – interpolacja



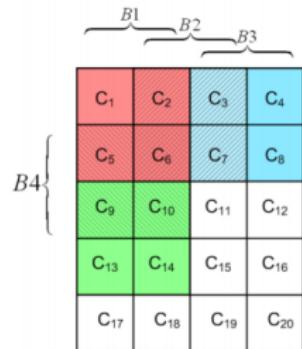
HOG – normalizacja w blokach

Moduły gradientów mają bardzo różne wartości w różnych częściach obrazu. Wynika to z nierównomiernego oświetlenia, czy różnicy w kontraste pomiędzy pierwszym planem, a tłem.

Użycie jako deskryptora otrzymanych histogramów nie prowadziłoby do dobrych rezultatów detekcji. Konieczne jest dodanie etapu normalizacji.

Autorzy algorytmu zaproponowali normalizację w blokach (ang. *blocks*) tj. połączonych sąsiednich komórkach. Najlepsze wyniki daje normalizacja w zachodzących na siebie blokach.

HOG – normalizacja w blokach



Użycie nachodzących bloków skutkuje wielokrotną normalizacją tych samych danych. Pozornie jest to nadmiarowość. Jednak uzyskane wyniki potwierdzają zasadność takiego postępowania (poprawa wyników o 4%).

HOG – normalizacja w blokach

Autorzy rozważali dwa rodzaje bloków:

- ✚ prostokątne (R-HOG) – rozmiary 1×2 , 2×2 .
- ✚ okrągłe (C-HOG).

My będziemy używać bloków R-HOG o następujących parametrach:

- ✚ kwadratowy o rozmiarze 2×2 .
- ✚ bloki zachodzą na siebie o 1 (o połowę długości),
- ✚ nakładanie się bloków realizowane jest zarówno w pionie, jak i poziomie.



HOG – możliwe schematy normalizacji

AGH

Niech: v – wektor nieznormalizowany (tj. histogram), $\|v\|_k$ norma ($k = 1, 2$), ϵ – niewielka stała.

✚ L2 –

$$v = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \quad (5)$$

✚ L2-Hys – jak L2 tylko z limitem na v równym 0.2

✚ L1 –

$$v = \frac{v}{\|v\|_1 + \epsilon} \quad (6)$$

✚ L1-sqrt –

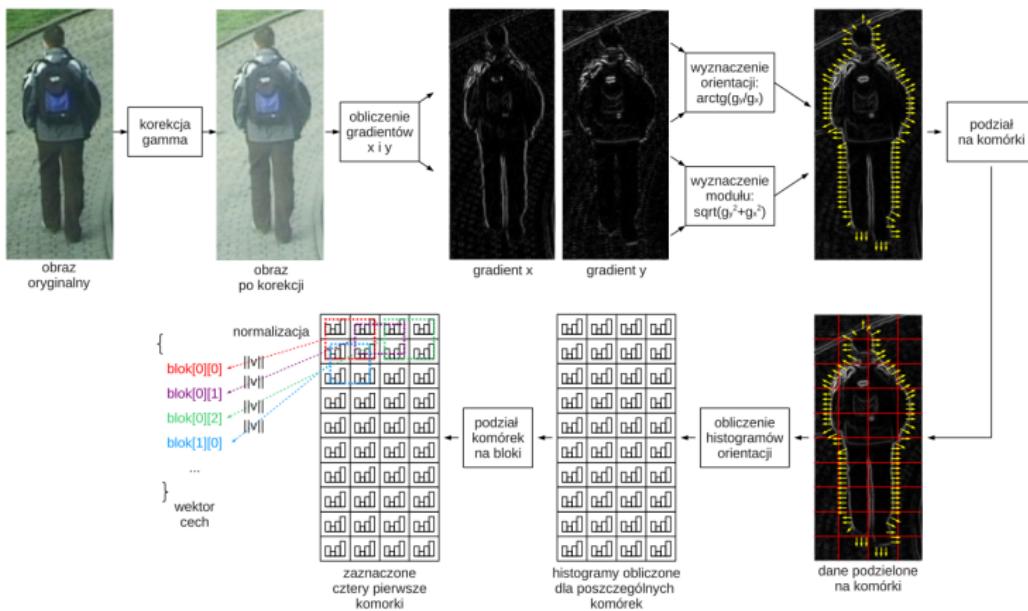
$$v = \frac{v}{\sqrt{\|v\|_1 + \epsilon}} \quad (7)$$

Badania pokazały, że normy L2, L2-Hys i L1-sqrt prowadzą do zbliżonych wyników. Dla L1 skuteczność spada o 5%, a brak o 27%.

Podsumowując:

- ✖ bierzemy cztery sąsiednie komórki, tworzymy z nich jeden wektor ($9 \times 4 = 36$) i normalizujemy,
- ✖ wynik stanowi element wektora cech,
- ✖ przesuwamy się do sąsiedniego bloku (o jedną komórkę) i powtarzamy procedurę.
- ✖ finalnie otrzymamy dla przyjętych założeń wektor cech o długości **3780**.

HOG – podsumowanie



Rysunek autorstwa Mateusza Komorkiewicza.

Klasyfikator SVM

SVM – ang. *Support Vector Machine* – maszyna wektorów nośnych (wsparcia).

Metoda zaproponowana przez Vladimira Vapnika w 1992. W podstawowej wersji jest to liniowy klasyfikator binarny, ale metodę można rozszerzyć na wiele klas oraz wprowadzić nieliniowość.

Co to znaczy **binarny** ?

Co to znaczy **liniowy** ?

Uczenie klasyfikatora odbywa się z nauczycielem (tj. po podstawie etykietowanych próbek).

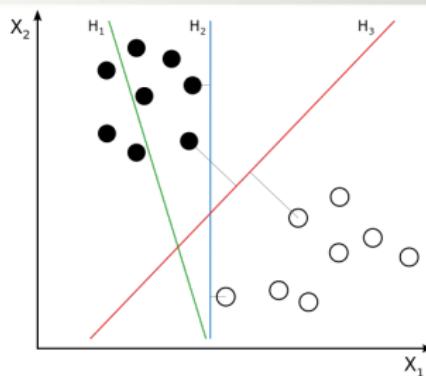
SVM – wstęp cd.

Najprościej mówiąc algorytm uczenia SVM znajduje hiperpłaszczyznę, która rozdziela dwa zbiory danych (zbiory wektorów cech) z możliwe największym marginesem.

Dane (wektory cech), reprezentowane są jako punkty w N-wymiarowej przestrzeni. Jest to wektor liczb rzeczywistych.

Jeśli każdy punkt ma etykietę tzn. jest powiedziane, że należy do klasy -1 (próbka negatywna) lub 1 (próbka pozytywna), to można wyznaczyć hiperpłaszczyznę rozdzielającą te punkty.

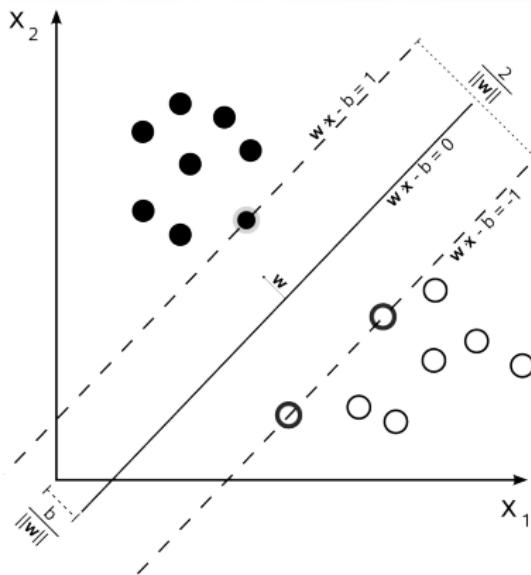
Oczywiście takich hiperpłaszczyzn istnieje wiele. Dlatego wybiera się tą z największym marginesem.



- ✖ H1 – nie oddziela dwóch klas,
- ✖ H2 – oddziela klasy, ale z bardzo niewielkim marginesem,
- ✖ H3 – oddziela klasy z dużym marginesem (maksymalnym).

Źródło: en.wikipedia.org

SVM – intuicja (2)



Źródło: en.wikipedia.org

SVM – trochę matematyki (1)

Równanie hiperpłaszczyzny:

$$w \cdot x + b = 0 \quad (8)$$

gdzie: w – wektor normalny hiperpłaszczyzny, x – punkt na hiperpłaszczyźnie, b – stała.

Ponadto: $\frac{b}{\|w\|}$ (gdzie: $\|w\|$ – norma w) to odległość hiperpłaszczyzny od początku układu współrzędnych.

Optymalna hiperpłaszczyzna separująca to taka, w której odległości pomiędzy nią i najbliższymi próbami pozytywnymi i negatywnymi są maksymalizowane.

SVM – trochę matematyki (2)

Wyznaczamy dwie pomocnicze hiperpłaszczyzny:

$$w \cdot x + b = 1 \quad (9)$$

$$w \cdot x + b = -1 \quad (10)$$

Zadanie: tak dobrać w i b , aby odległości pomiędzy hiperpłaszczyzną optymalną i pomocniczymi były równe i jak największe.

SVM – trochę matematyki (3)

Dane separowalne liniowo – hiperpłaszczyzny można wyznaczyć tak, aby “pośrodku” nie było żadnych punktów.

Zatem

$$w \cdot x_i + b \geq 1 \quad (11)$$

dla próbek pozytywnych.

$$w \cdot x_i + b \leq -1 \quad (12)$$

dla próbek negatywnych.

Łącząc:

$$y_i(w \cdot x_i + b) - 1 \geq 0 \quad (13)$$

gdzie: y_i – próbka pozytywna lub negatywna $y_i \in \{-1, 1\}$, $i = 1 \dots L$, L – liczba próbek.

SVM – trochę matematyki (4)

Próbki, które spełniają nierówność (13) tj. leżą na jednej lub drugiej hiperpłaszczyźnie nazywa się wektorami podpierającymi (nośnymi) – ang. *support vector*.

De facto, tylko na ich podstawie wyznaczane są obie hiperpłaszczyzny. Wszystkie inne próbki można by usunąć, a wynik pozostałby taki sam.

Odległość hiperpłaszczyzn pomocniczych od początku układu współrzędnych dana jest zależnościami: $\frac{b-1}{\|w\|}$ i $\frac{b+1}{\|w\|}$. Zatem odległość pomiędzy hiperpłaszczyznami wynosi:

$$\frac{b+1}{\|w\|} - \frac{b-1}{\|w\|} = \frac{2}{\|w\|} \quad (14)$$

SVM – trochę matematyki (5)

Ponieważ zakłada się, że odległości hiperpłaszczyzn pomocniczych od hiperpłaszczyzny optymalnej muszą być równe, wynoszą one: $\frac{1}{\|w\|}$

Zatem dochodzimy do problemu optymalizacji:

$$\min_{w,b} \|w\|^2 \quad (15)$$

przy ograniczeniu:

$$y_i(w \cdot x_i + b) - 1 \geq 0 \quad (16)$$

W ramach tego kursu nie będziemy rozważać, jak ww. problem rozwiązać.

SVM – do czego sprowadza się klasyfikacja ?

Jeśli mamy nauczoną maszynę (tj. rozwiązaliśmy pokazany problem optymalizacji), to sprawa użycia jest dość prosta. Mamy próbkę Z (wektor cech).

Sprawdzamy, jak jest ona położona w stosunku do wyznaczonej hiperpłaszczyzny.

$$y(Z) = \text{signum}(w \cdot Z + b) \quad (17)$$

Warto zauważyć, że jest to proste mnożenie dwóch wektorów (wag w i cech Z) połączone z dodawaniem b .

SVM – a jak dane nie są separowalne liniowo ?

Jeśli dane wejściowe nie są separowalne liniowo to opisany wyżej algorytm nie zadziała.

W takim przypadku należy wykorzystać SVM z tzw. elastycznym marginesem (ang. *soft margin*). Polega to na osłabieniu nierówności:

$$w \cdot x_i + b \geq 1 \quad (18)$$

$$w \cdot x_i + b \leq -1 \quad (19)$$

tak aby optymalna hiperpłaszczyzna mogła zostać wyznaczona, nawet w przypadku, gdy w przestrzeni pomiędzy hiperpłaszczyznami pomocniczymi znajdują się jakieś pojedyncze próbki.

SVM – a jak dane nie są separowalne liniowo ?

Aby rozwiązać ten problem wprowadza się pojęcie zmiennych osłabiających (ang. *slack variables*) ξ_i , których wartość to odległość punktów niespełniających ww. nierówności od odpowiedniej hiperpłaszczyzny.

$$w \cdot x_i + b \geq 1 - \xi_i \quad (20)$$

$$w \cdot x_i + b \leq -1 + \xi_i \quad (21)$$

Zakłada się: $\xi_i \geq 0$

SVM – a jak dane nie są separowalne liniowo ?

Prowadzi to do modyfikacji problemu optymalizacji:

$$\min_{w,b} \|w\|^2 + C \sum_{i=0}^L \xi_i \quad (22)$$

przy ograniczeniu:

$$y_i(w \cdot x_i + b) - 1 + \xi_i \geq 0 \quad (23)$$

gdzie: $\xi_i \geq 0, i = 0 \dots L$.

Współczynnik C to współczynnik kary, który jest podawany przez użytkownika.

$$\min_{w,b} \|w\|^2 + C \sum_{i=0}^L \xi_i \quad (24)$$

W równaniu mamy dwa człony, a zatem dwa skrajne przypadki:

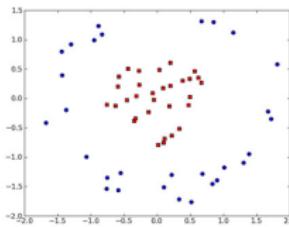
- ✖ małe C – problem sprowadza się do omówionego wcześniej.
Poszukiwany jest duży margines, a ew. błędy są ignorowane.
- ✖ duże C – duża kara za błędy. Margines będzie mniejszy, może nastąpić nadmierne dopasowanie, czyli przetrenowanie (tzw. *overfitting*) – utrata zdolności do generalizacji.

Wybór parametru C jest zatem zawsze kompromisem. Zwykle dobór jest empiryczny tj. dla różnych wartości C sprawdza się wyniki klasyfikacji.

SVM – wersja nieliniowa

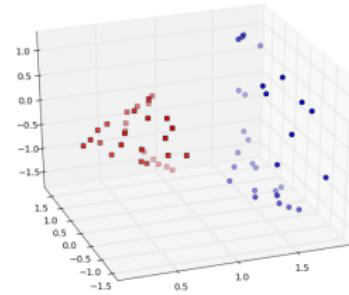
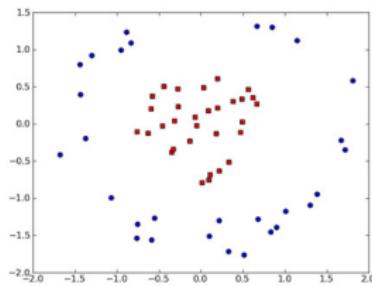
Opisany powyżej mechanizm *soft margin* zdaje egzamin w przypadku, kiedy tylko niewielka część próbek leży pomiędzy hiperpłaszczyznami pomocniczymi. Inaczej mówiąc problem jest liniowo separowalny, ale z pewnymi zakłóceniami.

Może się jednak okazać, że dane nie są separowalne liniowo i wtedy klasyfikator liniowy zupełnie zawodzi.



SVM – wersja nieliniowa

Opisany problem można rozwiązać poprzez odpowiednie przekształcenie danych wejściowych. Przykładowo dane, które nie są separowalne w przestrzeni n -wymiarowej mogą być separowalne w $n+1$ -wymiarowej.



Używamy mapowania:

$$\Phi : \Re^n \mapsto H \quad (25)$$

Wykorzystujemy funkcję jądrową (ang. *kernel function*), która przyjmuje jako argumenty dwa wektory, a zwraca wartość iloczynu skalarnego ich odwzorowań w nowej przestrzeni:

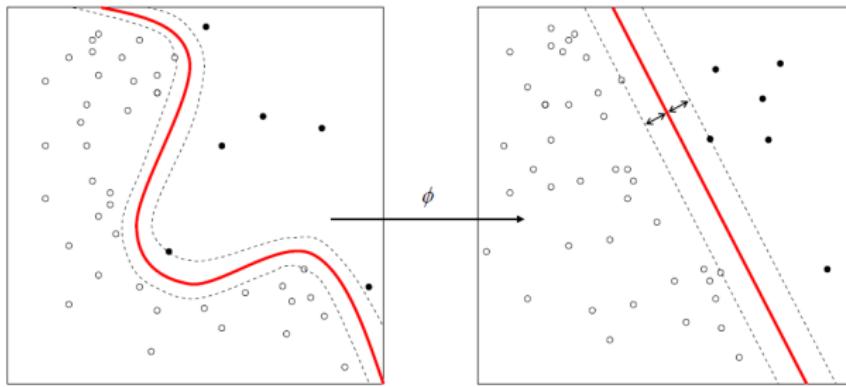
$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \quad (26)$$

Co ciekawe, ani do uczenia, ani klasyfikacji nie potrzebujemy znać wprost funkcji Φ . Interesuje nas tylko zastąpienie iloczynu skalarnego “jakimś” wyrażeniem.

Przykładowe jądra:

- ✚ $K(x_i, x_j) = (x_i \cdot x_j)^d$ – wielomianowe,
- ✚ $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ – RBF (*Radial Basis Function*),
- ✚ $K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j + b)$ – sigmoidalne.

Warto zauważyć, że koncepcję można zastosować do każdego algorytmu klasyfikacyjnego, gdzie wykorzystuje się tylko iloczyn skalarny.

SVM – *kernel trick*

Źródło: en.wikipedia.org

SVM – dobór parametrów

Patrząc czysto technicznie. Mamy narzędzie, które ma kilka parametrów:

- ✖ C
- ✖ jądro + jego parametry.

Do wyboru parametrów wykorzystuje się często tzw. *grid search* (wyszukiwanie w węzłach). Przykładowo sprawdza się następujące wartości parametru $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$.

Wyboru dokonuje się na podstawie wyników klasyfikacji.

SVM – dla wielu klas

Najczęściej wykorzystuje się podejście **jeden** do **reszty** (ang. *one versus all*).

Przykładowo jeśli mamy 4 klasy: A, B, C, D, to będziemy wykorzystywać również cztery SVM:

- ✖ A kontra B,C,D,
- ✖ B kontra A,C,D,
- ✖ C kontra A,B,D,
- ✖ D kontra A,B,C.



SVM – przykład praktyczny (1)

AGH

Przeanalizujmy jak powinniśmy podejść do problemu detekcji pieszych z wykorzystaniem deskryptora HOG oraz klasyfikatora SVM.

- ✖ zbiór danych – MIT CBCL Pedestrian Data (są też inne),
- ✖ analizujemy wycinki o rozmiarze 64×128 pikseli – nie zajmujemy się analiza “całych” obrazków,
- ✖ mamy próbki pozytywne (z człowiekiem) i negatywne (bez człowieka) – [ważny dobór],
- ✖ mamy zbiory pos i neg. Każdy zawiera po 924 obrazy,
- ✖ dzielimy je na 3 części: 60 % uczący, 20 % walidacyjny, 20 % testowy [ręcznie].
- ✖ dla każdego obrazka, z każdego zbioru generujemy deskryptor HOG,
- ✖ najlepiej jest je zapisać w formie plików mat (save)

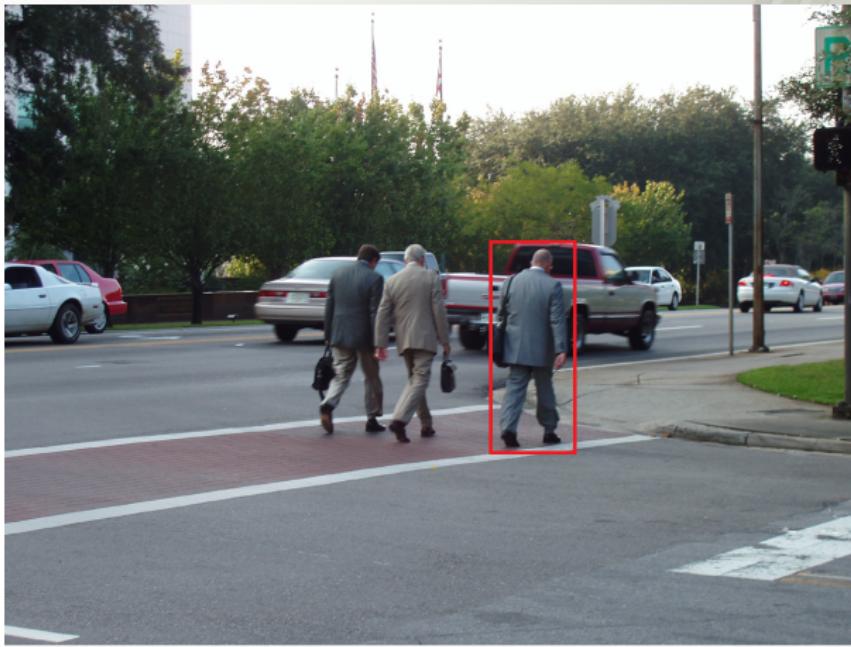
SVM – przykład praktyczny (2)

- ✖ używamy liniowego SVM. Do uczenia funkcja `svmtrain`, do testów `svmclassify`.
- ✖ klasyfikujemy dla zbioru uczącego i dla zbioru walidacyjnego,
- ✖ obliczamy wskaźniki TP, TN, FP, FN i pochodne,
- ✖ analizujemy wyniki, jeśli są poprawne to sprawdzamy zbiór testowy,
- ✖ zadanie dodatkowe – implementacja mechanizmu na pełnych obrazkach:
 - ▶ przesuwne okno,
 - ▶ skale,
 - ▶ agregacja detekcji.

Detkacja na rzeczywistym obrazie (1)



Detkecja na rzeczywistym obrazie (2)



Detkecja w wielu skalach

