Metody eksploracji danych
6. Klasyfikacja (kontynuacja)

Piotr Szwed
Katedra Informatyki Stosowanej AGH
2016

Support Vector Machines
k-Nearest Neighbors

Support Vector Machines
Maszyny Wektorów Wspierających

Liniowa separowalność danych

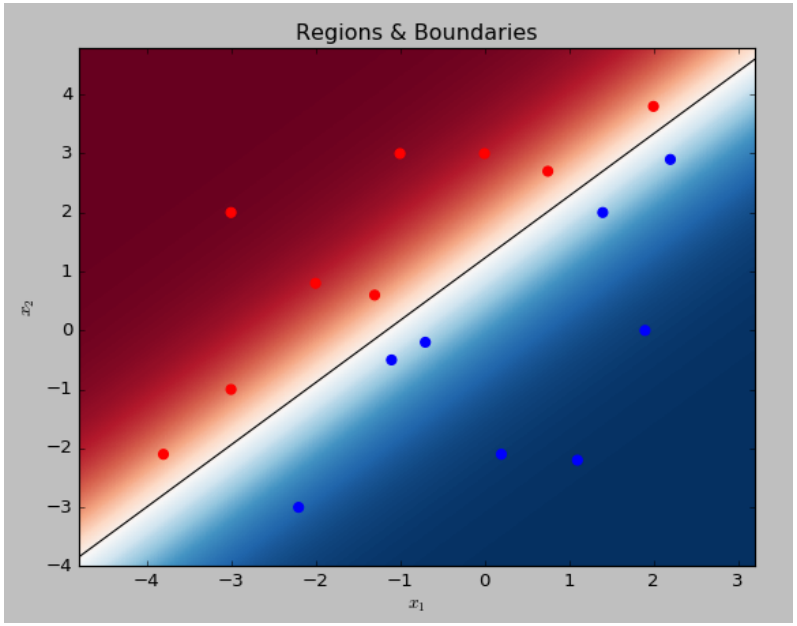
- Rozważmy zagadnienie binarnej klasyfikacji, w którym zbiorem etykiet jest $\{-1,1\}$
- Zbiór danych $D = \{(x_i, y_i): i = 1, \dots, m\}$, gdzie $x_i \in R^n$ jest **linowo separowalny**, jeżeli istnieje hiperpłaszczyzna $w^T x + b = 0$, taka że:

$$\forall i: y_i = +1. w^T x_i + b > 0$$
$$\forall i: y_i = -1. w^T x_i + b < 0$$

- Powyższe dwa warunki można zapisać jako jeden:

$$\forall i: y_i = sign(w^T x_i + b)$$

- W tym przypadku: $w \in \mathrm{R}^n$, b to element dotychczas oznaczny jako w_0.

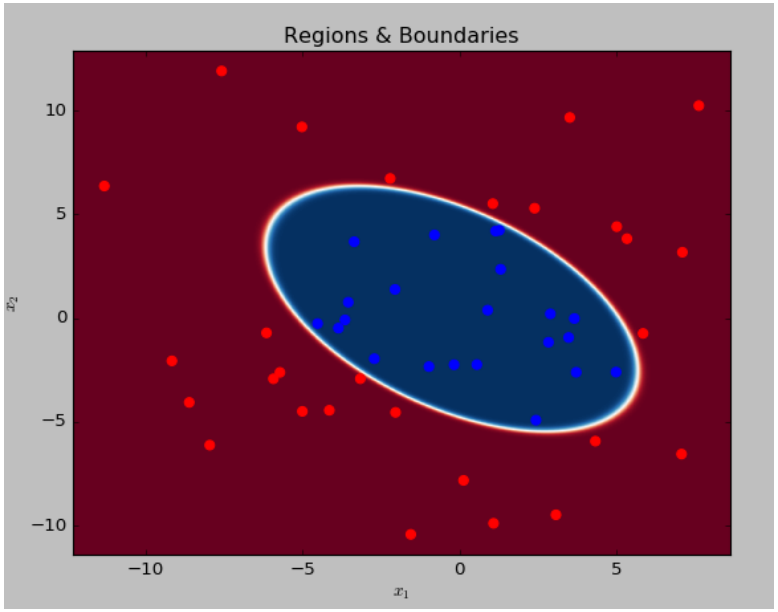


Liniowa separowalność danych

- W przypadku danych, które nie są bezpośrednio separowalne przez hiperpłaszczynę, możliwe jest wprowadzenie nieliniowych cech $h(x)$, np. wielomianów.

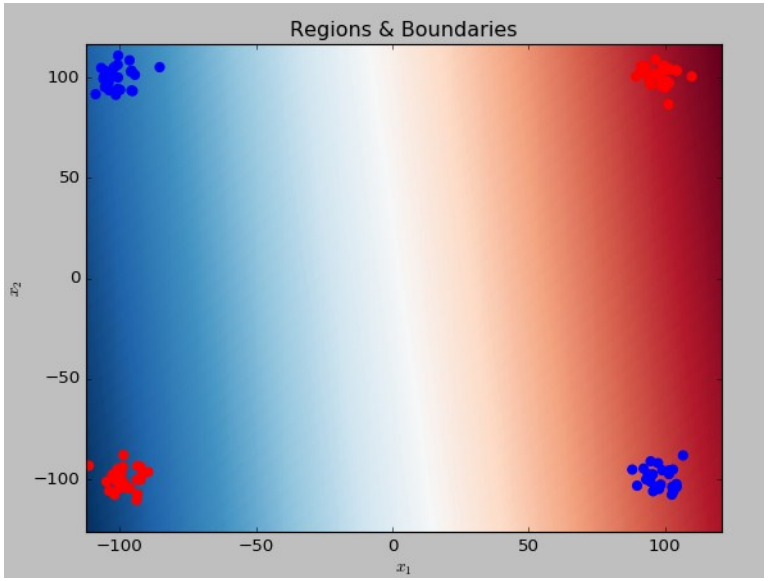
- W przestrzeni cech model dalej jest **liniowy** i równanie hiperpłaszczyzny ma postać:

$$w^T h(x) + b = 0$$

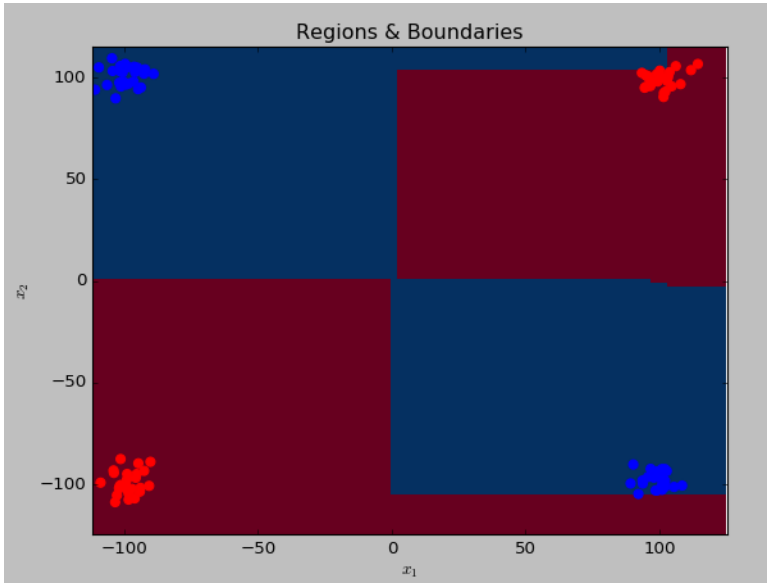


Liniowa separowalność danych

- Nieliniowe modele:
  - Drzewa decyzyjne
  - Sieci neuronowe
  - Najbliżsi sąsiedzi (kNN)
- Nieliniowe modele mogą zapewnić kształty regionów decyzyjnych zgodne z etykietami klas
  - trudniejsze do uczenia
  - ryzyko nadmiernego dopasowania



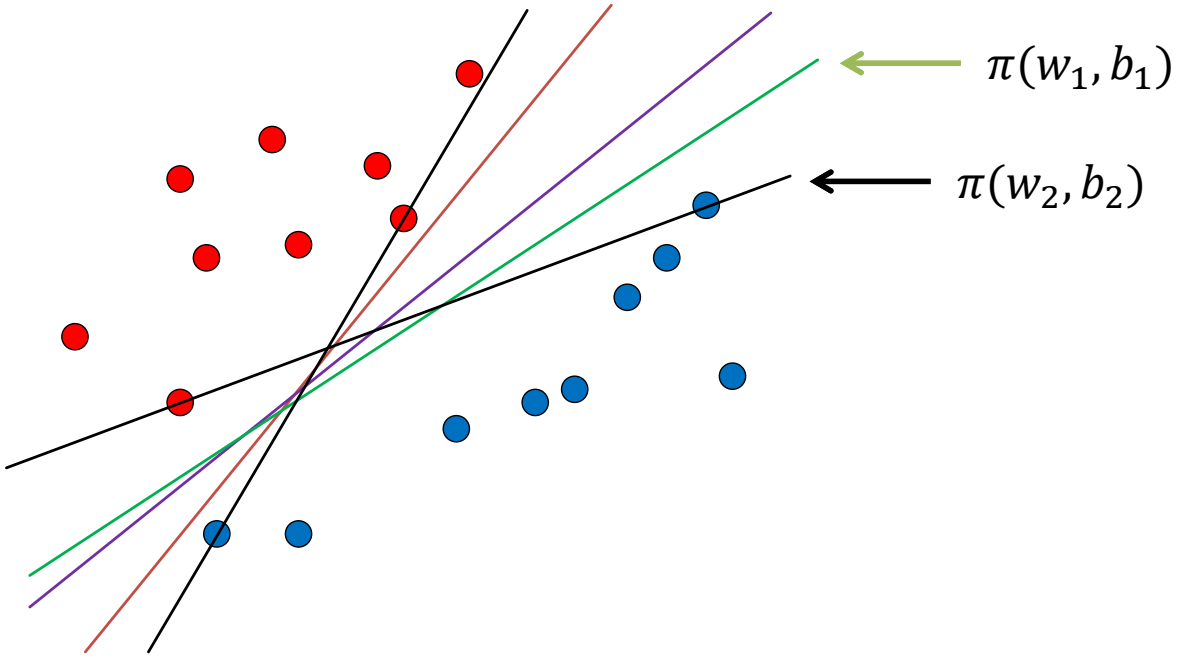
Regresja logistyczna



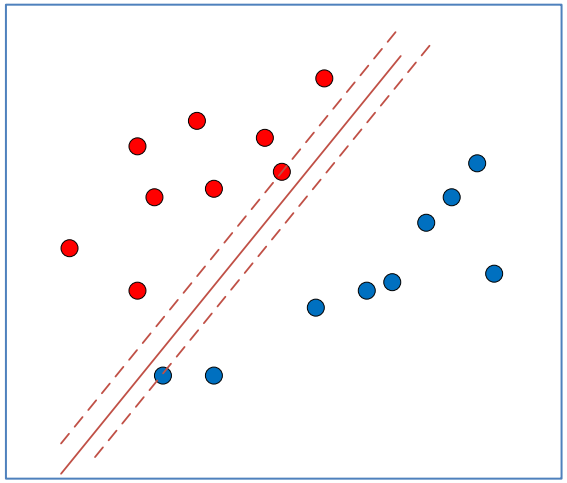
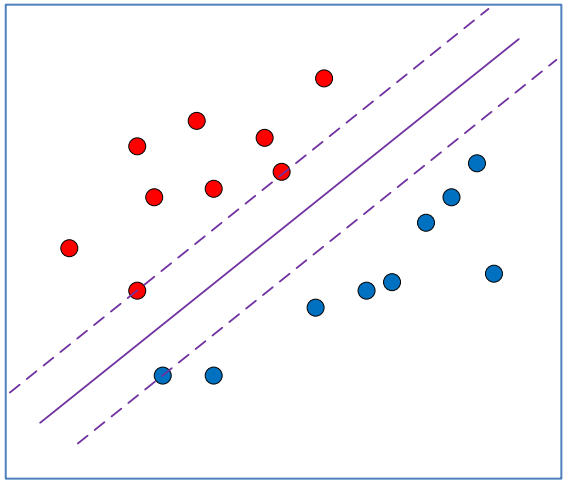
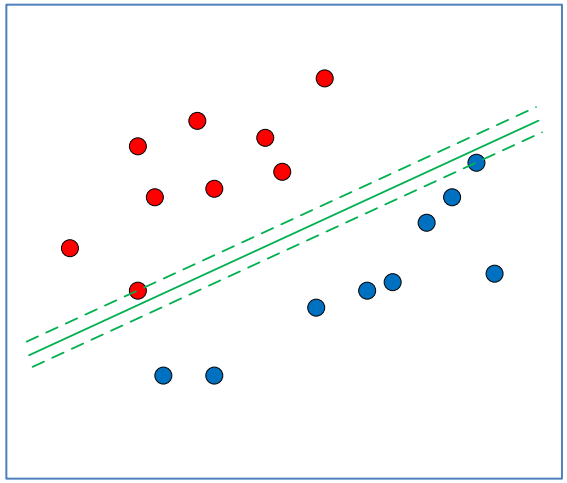
Drzewo decyzyjne

Koncepcja SVM

- Zakładając, że dane są separowalne, zazwyczaj istnieje (nieskończenie) wiele hiperpłaszczyzn $\pi(w, b)$ rozdzielających dane.
- Kryterium wyboru: $\pi(w_1, b_1)$ jest lepszą granicą klas niż $\pi(w_2, b_2)$ jeżeli obserwacje są od niej bardziej odległe. (Mniejsza podatność na szum i nadmierne dopasowanie)



Margines

- Hiperpłaszczyznę separującą klasy można rozsuwać w obie strony (linie przerywane), aż do momentu, w którym pojawią się na nich obserwacje.
- Margines to odległość pomiędzy hiperpłaszczyznami oznaczonymi liniami przerywanymi
- Wielkość marginesu zależy od ułożenia hiperpłaszczyzny

Model matematyczny

- Załóżmy, że zbiór uczący $D = \{(x_i, y_i): i = 1, \dots, m\}$ jest linowo separowalny przez hiperpłaszczyznę o marginesie ρ.
- Wówczas:
  1. $w^T x_i + b \leq -\frac{\rho}{2}$ jeżeli $y_i = -1$
  2. $w^T x_i + b \geq +\frac{\rho}{2}$ jeżeli $y_i = +1$
- Oba warunki można zapisać jako:

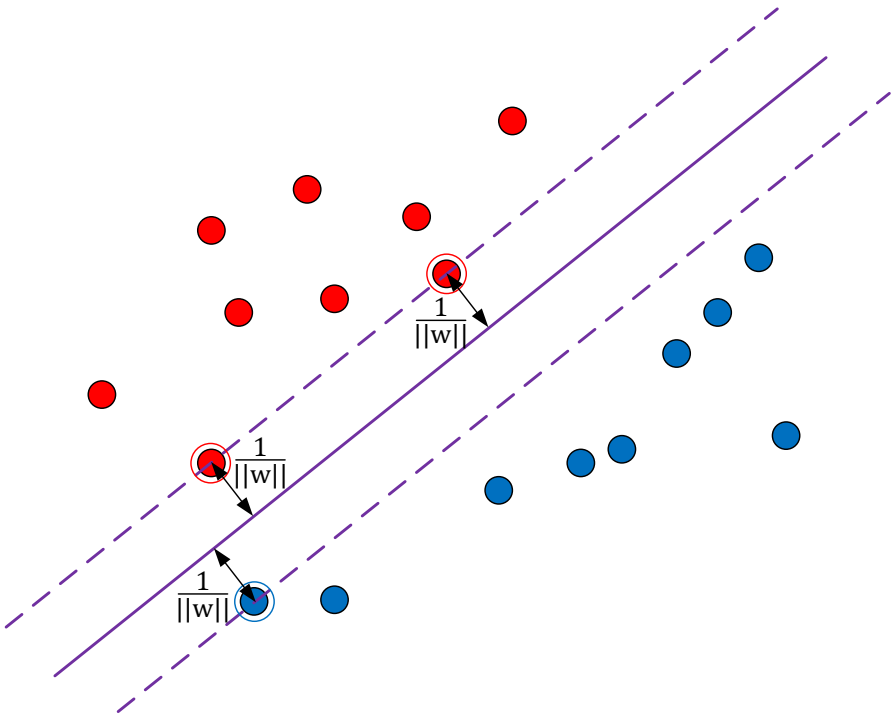
$$y_i(w^T x_i + b) \geq \frac{\rho}{2}$$

- Przeskalujmy w i b dzieląc obie strony przez $\frac{\rho}{2}$. Równanie (po przeskalowaniu) ma postać:

$$y_i(w^T x_i + b) \geq 1$$

Wektory wspierające

Wektory wspierające (nośne, ang. **support vectors**) to obserwacje, dla których nierówność $\mathrm{y_i}(w^T x_i + b) \geq 1$ jest równością, czyli zachodzi:
$\forall x_s\colon \mathrm{y_s}(w^T x_s + b) = 1$



Obliczmy odległość wektora wspierającego od hiperpłaszczyny:

$$r = \frac{|w^T x_s + b|}{\|w\|} = \frac{\mathrm{y_s}(w^T x_s + b)}{\|w\|} = \frac{1}{\|w\|}$$

Czyli margines dla przeskalowanych w i b może być wyrażony jako:

$$\rho = 2r = \frac{2}{\|w\|}$$

Optymalna hiperpłaszczyzna

Zagadnienie optymalizacji

Znajdź w i b takie, że:

$\rho = 2r = \frac{2}{\|w\|}$ jest maksymalizowane (funkcja celu)

przy ograniczeniach: $\forall i = 1, m$: $\mathrm{y_i}(w^T x_i + b) \geq 1$

przeformułowane jako:

Zagadnienie optymalizacji **kwadratowej**

Znajdź w i b takie, że:

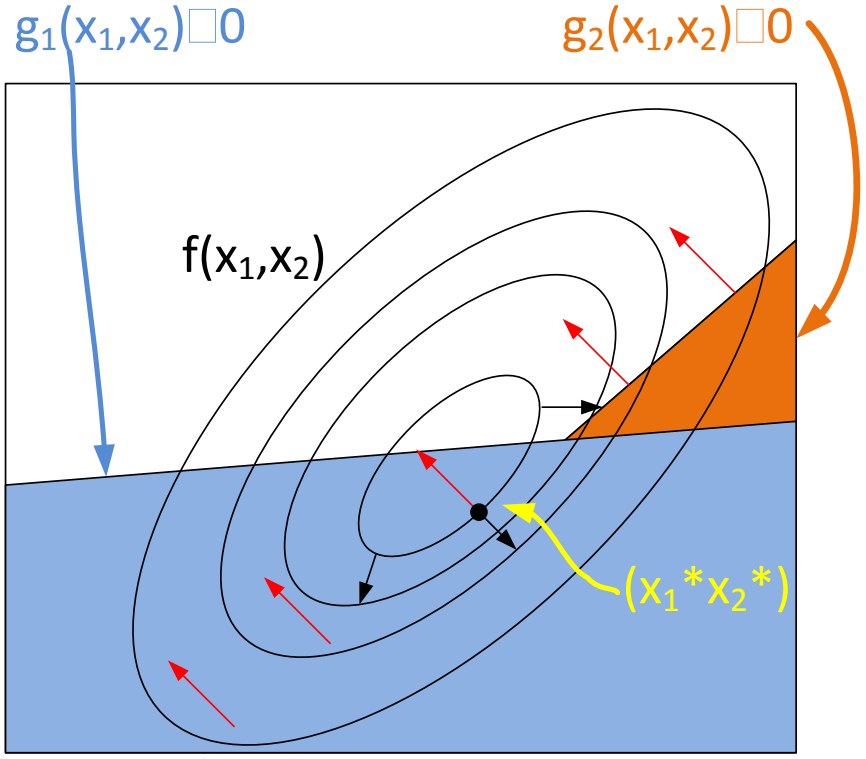
$\Phi(w) = \frac{1}{2} w^T w$ jest minimalizowane (funkcja celu)

przy ograniczeniach: $\forall i = 1, m$: $\mathrm{y_i}(w^T x_i + b) \geq 1$

- Kwadratowa funkcja celu: zmienne to w i b
- Z liniowymi ograniczeniami (y_i i x_i odgrywają rolę współczynników)

Problem optymalizacji

- Problem optymalizacji z ograniczeniami rozwiązuje się przez zastosowanie tzw. mnożników Lagrange'a
- [Także https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions].
- W punkcie optymalnym spełniającym ograniczenia wektory gradientów funkcji i aktywnych ograniczeń muszą być równoległe i przeciwnie skierowane



$$L(x_1, x_1, \alpha) = f(x_1, x_2) - \sum_i \alpha_i g_i(x_1, x_2)$$

- α_i - mnożniki Lagrange'a
- Ograniczenie g_1 nie jest aktywne w punkcie optymalnym . Stąd: $g_1(x_1^*, x_2^*) < 0$ oraz $\alpha_1 = 0$
- Ograniczenie g_2 jest aktywne w punkcie optymalnym, czyli spełnione będzie $g_2(x_1^*, x_2^*) = 0$ oraz $\alpha_2 \neq 0$

Mnożniki Lagrange'a

- Funkcja Lagrange'a dla zgadnienia SVM:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i(w^T x_i + b) - 1\right)$$

  - $\frac{1}{2}\|w\|^2$ to oryginalna funkcja celu
  - $\alpha_i y_i((w^T x_i + b) - 1)$ to ograniczenia wynikające z obserwacji w zbiorze uczącym.
  - Zauważmy, że $\alpha_i = 0$, dla obserwacji nie będących wektorami wspierającymi oraz $\alpha_i \neq 0$ dla wektorów wspierających.
- Rozwiązanie polega na:
  - obliczeniu pochodnych $L(w, b, \alpha)$ względem zmiennych w i b
  - przyrównaniu pochodnych do 0 (warunek konieczny)
  - wstawieniu otrzymanych zależności na w^* i b^* do $L(w, b, \alpha)$
  - rozwiązaniu **problemu dualnego maksymalizacji** $L(w^*, b^*, \alpha)$ w przestrzeni współczynników α.
  - $[w^*, b^*, \alpha^*]$ jest punktem siodłowym, będącym miejscem minimum $L(w, b, \alpha^*)$ względem w i b oraz maksimum $L(w^*, b^*, \alpha)$ względem α.

Mnożniki Lagrange'a

1. Obliczamy pochodne i przyrównujemy do 0:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i(w^T x_i + b) - 1\right)$$

- $\frac{\partial L(w,b,\alpha)}{\partial w_j} = w_j - \sum_{i=1}^{m} \alpha_i y_i (x_{ij})$, stąd

  $\frac{\partial L(w,b,\alpha)}{\partial w_j} = 0 \;\Rightarrow\; w_j = \sum_{i=1}^{m} \alpha_i y_i (x_{ij}) \Rightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i$

- $\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$

2. Podstawiamy do wzoru $L(w,b,\alpha)$:

- $L(\alpha) = \frac{1}{2}\|\sum_{i=1}^{m} \alpha_i y_i x_i\|^2 - \sum_{i=1}^{m} \alpha_i y_i \left((\sum_{j=1}^{m} \alpha_j y_j x_j)^T x_i\right) - \sum_{i=1}^{m} \alpha_i y_i b + \sum_{i=1}^{m} \alpha_i \cdot 1$
- $L(\alpha) = \frac{1}{2}\|\sum_{i=1}^{m} \alpha_i y_i x_i\|^2 - (\sum_{j=1}^{m} \alpha_j y_j x_j)^T \sum_{i=1}^{m} \alpha_i y_i x_i + \sum_{i=1}^{m} \alpha_i \cdot 1$
- $L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\|\sum_{i=1}^{m} \alpha_i y_i x_i\|^2$

Zagadnienie dualne

Zagadnienie dualne maksymalizacji

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i x_i \right\|^2$$

przy ograniczeniach: $\alpha_i \geq 0$ oraz $\sum_{i=1}^{m} \alpha_i y_i = 0$

Postać alternatywna

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \, \alpha_j y_i y_j (x_i^T x_j)$$

przy ograniczeniach: $\alpha_i \geq 0$ oraz $\sum_{i=1}^{m} \alpha_i y_i = 0$

- Składnik $\sum_{i=1}^{m} \alpha_i$ odpowiada maksymalizacji marginesu, który jest ograniczany przez odległości pomiędzy wektorami wspierającymi (drugi składnik).
- Pierwsza postać wydaje się bardziej efektywna (ale druga jest preferowana):
  - sumuje się m wektorów i oblicza ich iloczyn skalarny.
  - w drugiej wersji oblicza się m^2 iloczynów skalarnych

Model SVM

- Hiperpłaszczyzna $wx + b = 0$ wyznacza granicę klas. Stąd funkcja decyzyjna ma postać: $f(x) = sign(wx + b)$.
- Podstawiając $w = \sum_{i=1}^{m} \alpha_i y_i x_i$, otrzymujemy:

$$f(x) = sign(\sum_{i=1}^{m} \alpha_i y_i (x^T x_i) + b)$$

- Oznaczmy przez S zbiór indeksów wektorów wspierających $S = \{i: \alpha_i \neq 0\}$. Postać funkcji decyzyjnej można uprościć

$$f(x) = sign\left(\sum_{i \in S} \alpha_i y_i (x^T x_i) + b\right)$$

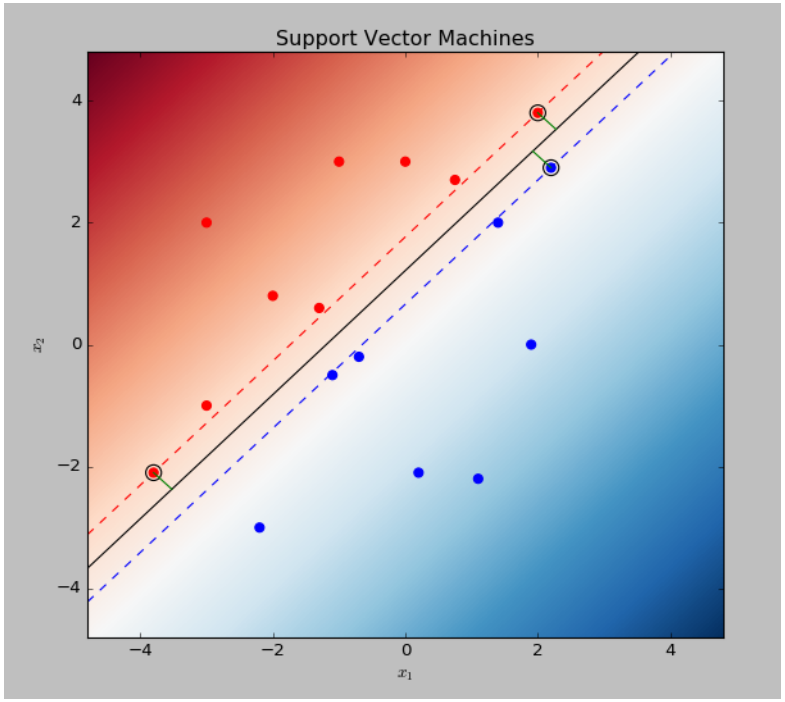
- Składnik $b = \frac{1}{|S|} \sum_{i \in S} (y_i - \sum_{j=1}^{m} \alpha_j \, y_j (x_i^T x_j))$

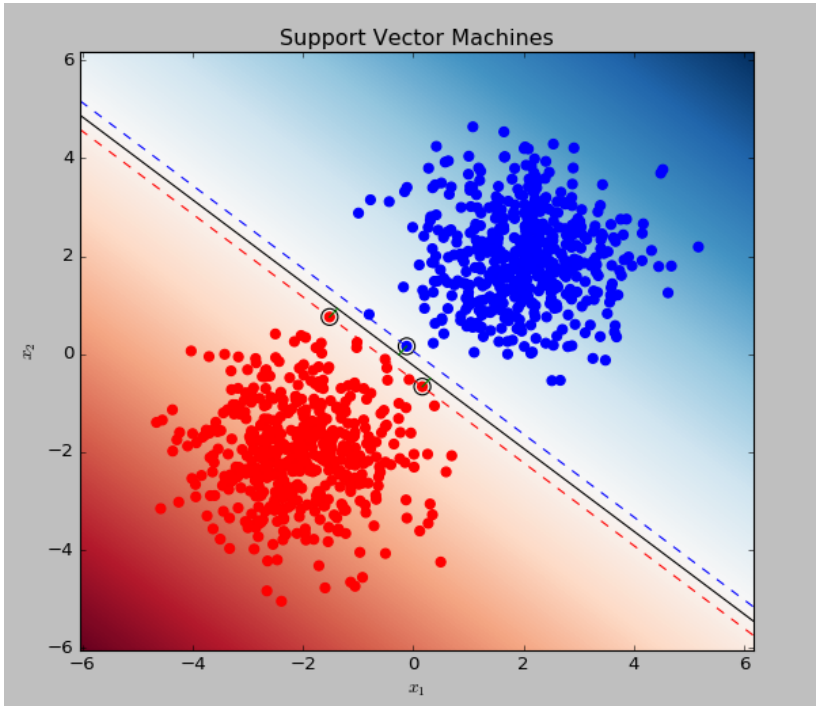
Model klasyfikatora SVM jest reprezentowany przez:

- zbiór wektorów wspierających: $\{x_j: j \in S\}$
- zbiór współczynników problemu dualnego $\{\alpha_j: j \in S\}$
- współczynnik b (intercept)

Nie jest konieczne jawne wyznaczanie wektora w

Przykład





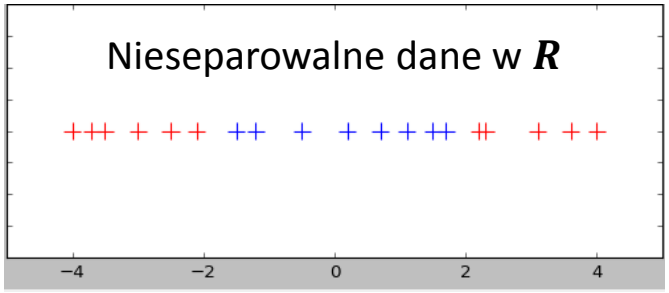
Niezależnie od liczby obserwacji, w obu przypadkach model składa się z 3 wektorów wspierających wykorzystywanych przez funkcję decyzyjną.

Ogólnie: liczba wektorów wspierających $|S| \leq n + 1$, czyli zależy od wymiarów problemu (dla liniowo separowalnych obserwacji).

Obserwacje na ogół nie są liniowo separowalne

- Dwie możliwości:
  - Transformacja zagadnienia, w którym obserwacje $x_i \in R^n$ do postaci w przestrzeni cech $h(x_i) \in R^N$, gdzie N będzie wystarczająco duże, aby zapewnić separowalność $h(x_i)$. Na przykład transformacja może polegać na wyznaczeniu czynników wielomianowych wybranego stopnia
  - Wprowadzenie *miękkiego marginesu*: dopuszczenie obserwacji naruszających separowalność i uwzględnienie ich w funkcji celu jako kary za przekroczenie ograniczeń.
- W praktyce stosuje się kombinację obu rozwiązań

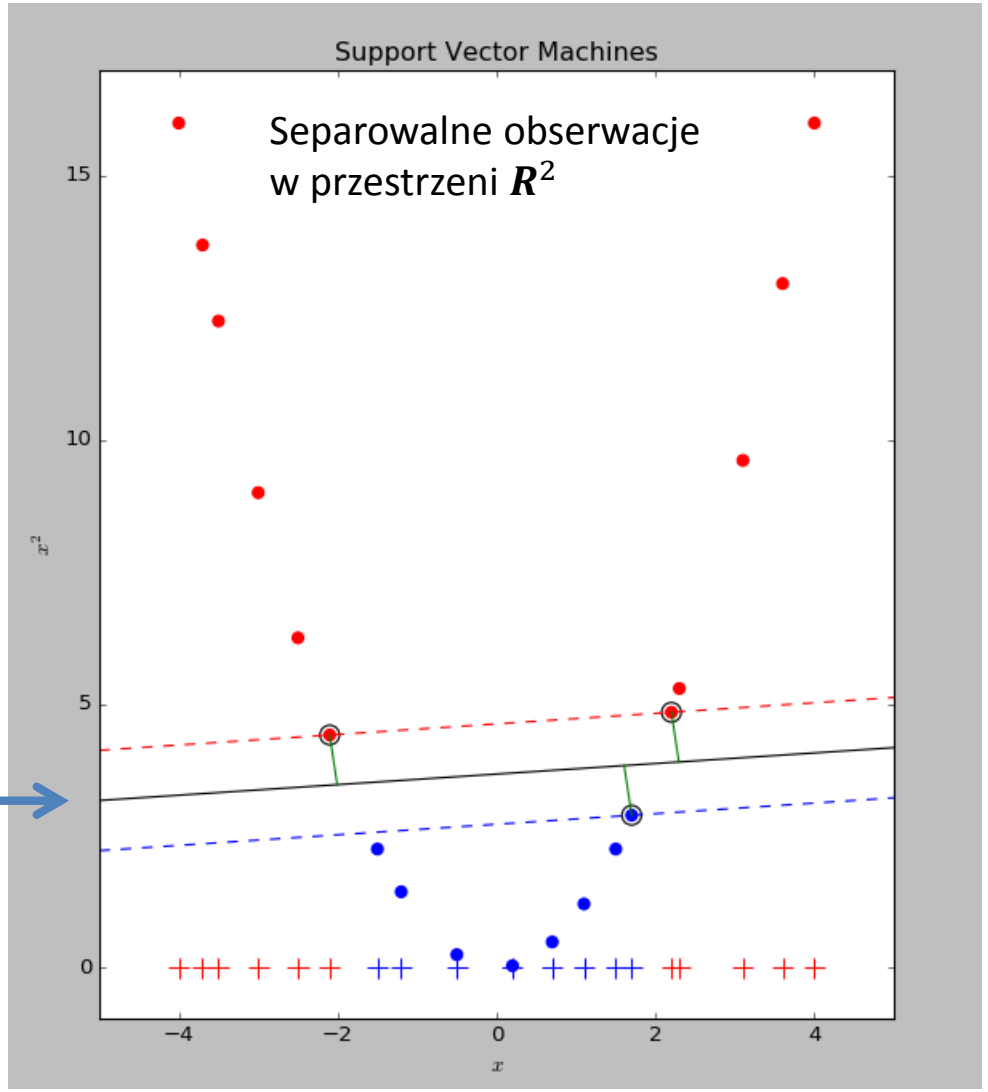
Transformacja do przestrzeni cech - przykład



Niech $h(x) \in \boldsymbol{R}^2$, gdzie

- $h_1 x) = x$
- $h_2(x) = x^2$

Transformacja pozwala na odseparowanie obserwacji w $\boldsymbol{R}^2$



Funkcja jądra (kernel)

- Po transformacji do przestrzeni cech funkcja decyzyjna ma postać:

$$f(x) = sign(\sum_{i \in S} \alpha_i y_i (h(x)^T \, h(x_i)) + b)$$

- Oznaczmy $K(x_i, x_j) = h(x)^T \, h(x_i)$. Po podstawieniu:

$$f(x) = sign\left(\sum_{i \in S} \alpha_i y_i K(x, x_i) + b\right)$$

- Czynnik $K(x_i, x_j)$ to tzw. jądro (**kernel)**, czyli funkcja odpowiedzialna za obliczanie iloczynu skalarnego w przestrzeni cech.

Przykład:

- Dwuwymiarowy wektor $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- Kernel $K(x, z) = (1 + x^T z)^2$

$$K(x,z) = 1 + 2x_1z_1 + 2x_2z_2 + 2x_1z_1x_2x_2 + x_1^2z_1^2 + x_2^2z_2^2$$
$$= \begin{bmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1x_2 & x_1^2 & x_2^2 \end{bmatrix}^T \begin{bmatrix} 1 & \sqrt{2}z_1 & \sqrt{2}z_2 & \sqrt{2}z_1z_2 & z_1^2 & z_2^2 \end{bmatrix}$$

- Czyli $h(x) = \begin{bmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1x_2 & x_1^2 & x_2^2 \end{bmatrix}$

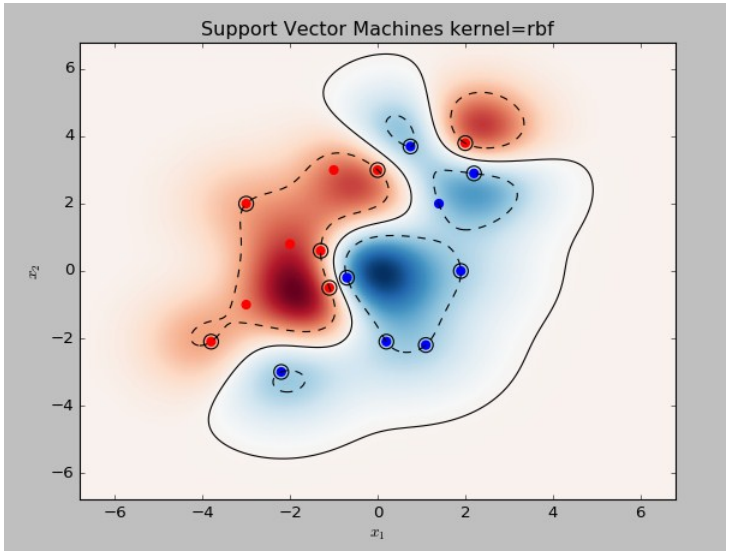
Kernel w niejawny sposób odwzorowuje przestrzeń X do przestrzeni cech $h(X)$ bez konieczności ich wyznaczania. Jest to tzw. **kernel trick**

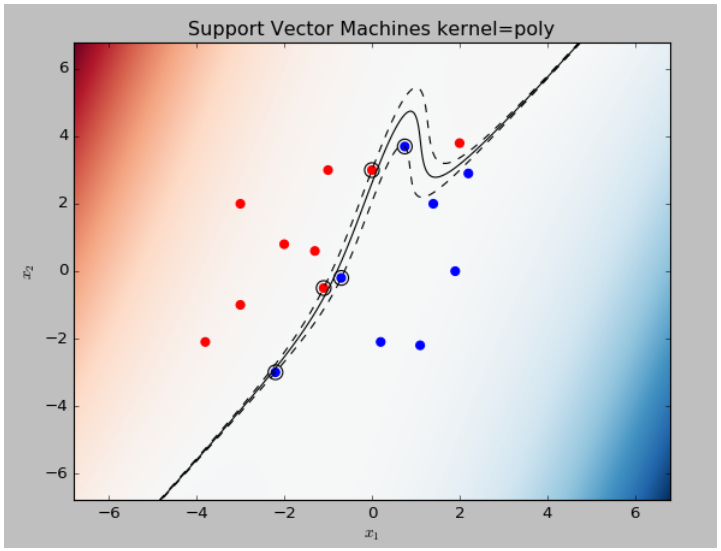
Stosowane funkcje jądra

Kernel	
Liniowy	$K(x_i, x_j) = x_i^T x_j$ Odwzorowuje $h(x) = x$
Wielomian stopnia p	$K(x_i, x_j) = \left(1 + x_i^T x_j\right)^p$ Wymiar $h(x)$ to $\binom{n+p-1}{p}$. Jest to liczba rzędu n^p. Np. dla $n = 100$ i $p = 6$ $N = 1\,609\,344\,100$
Funkcja Gaussa (RBF, radial basis function)	$K(x_i, x_j) = \mathrm{e}^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}}$ Wektor cech jest nieskończeniewymiarowy. Granicą klas jest kombinacja funkcji obliczana dla wektorów wspierających
Sigmoidalny	$K(x_i, x_j) = \tanh\left(\gamma\, \mathrm{x_i^T x_j} + \mathrm{r}\right)$

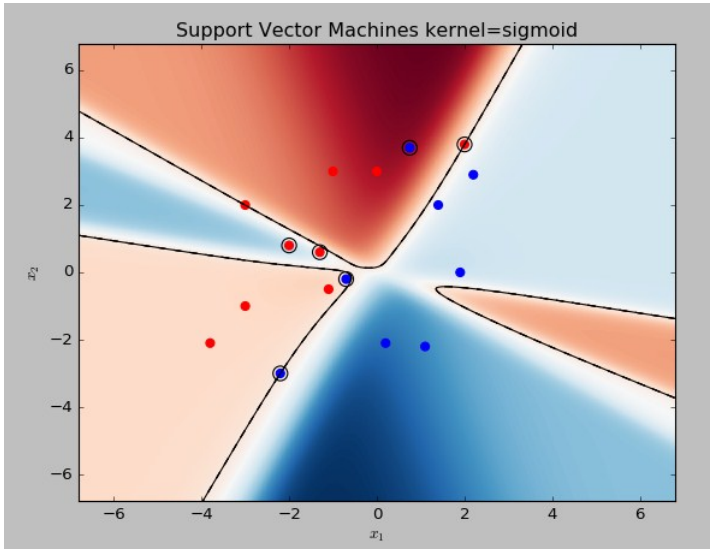
Uczenie SVM wymaga obliczenia $K(x_i, x_j)$ dla wszystkich $m \times m$ par obserwacji. Aby funkcja mogła być użyta jako kernel wymagane jest, aby macierz [K(x_i,x_j)] była nieujemnie określona dla dowolnych wektorów (x_i,i=1,…,m). Dalsze informacje: macierz Grama i twierdzenie Mercera.

Przykłady



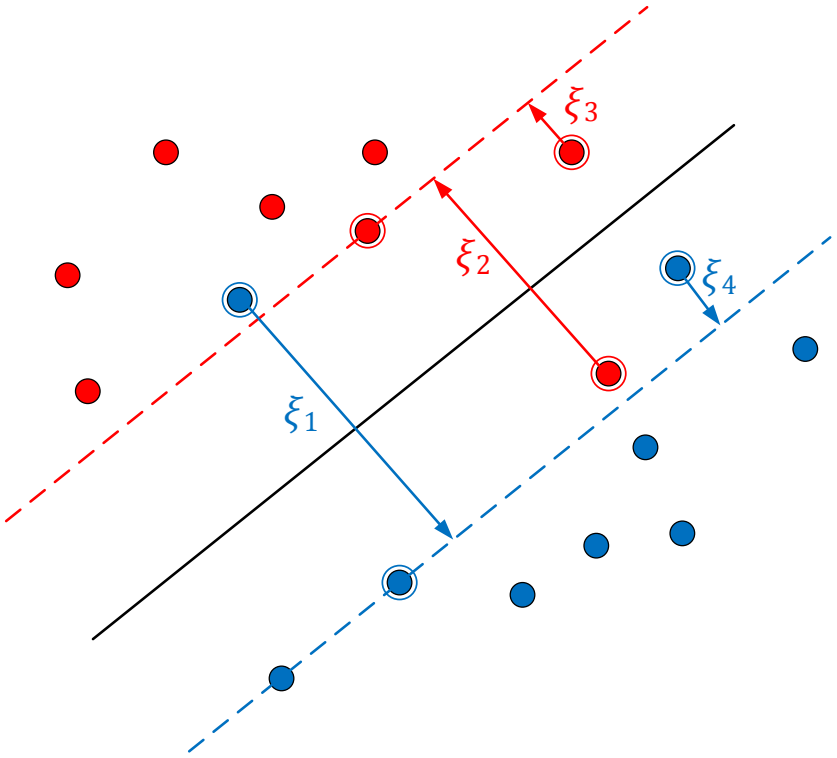


- Granice klas i marginesy dla różnych funkcji jądra (kerneli)
- Najlepsze rezultaty dla kernela gaussowskiego (RBF) ale równocześnie duża liczba wektorów wspierających
- W przypadku kernela wielomianowego, stopień wynosił 3
- W rzeczywistości uczenie modelu z użyciem miękkich marginesów (patrz dalej). Widoczne dla kernela sigmoidalnego.



Miękkie marginesy

- Jeżeli w zbiorze uczącym występują obserwacje (x_i, y_i), oraz (x_j, y_j) takie, że $x_i = x_j$ i $y_i \neq y_j$, dane nigdy nie będą liniowo separowalne, nawet po transformacji do przestrzeni cech o dowolnej złożoności.
- Miękkie marginesy są rozwiązaniem, które pozwala w takim przypadku zbudować klasyfikator SVM (ale ma ono charakter bardziej ogólny).



- W modelu dopuszcza się obserwacje leżące po niewłaściwej stronie hiperpłaszczyzn wyznaczających marginesy. W funkcji celu wprowadza się współczynniki kary ξ_i za przekroczenie ograniczeń.

Miękkie marginesy

Zagadnienie optymalizacji

Znajdź w, b, ξ takie, że:

$\Phi(w, b, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$ jest minimalizowane

przy ograniczeniach:

$$\forall i = 1, m: \; y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\forall i = 1, m: \xi_i \geq 0$$

Parametr C pozwala na kontrolowanie kompromisu pomiędzy celem maksymalizacji marginesu oraz dopasowaniem do danych uczących

- Jeżeli stała C jest mała – wybierany jest szeroki margines kosztem dopasowania do danych uczących
- Jeżeli stała C jest duża, margines staje się mały i następuje dobre dopasowanie do danych uczących

Hinge loss

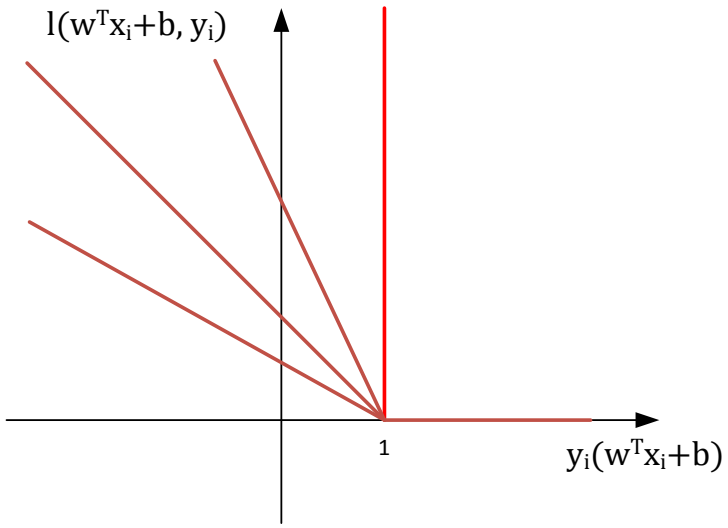
Funkcja celu:

$$\Phi = \frac{1}{2} w^T w + C \sum_{i=1}^{m} l(w^T x + b, y_i)$$

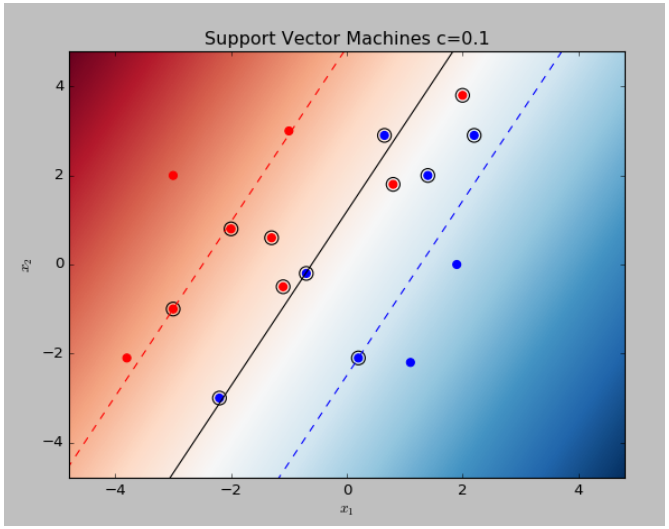
Złożoność modelu

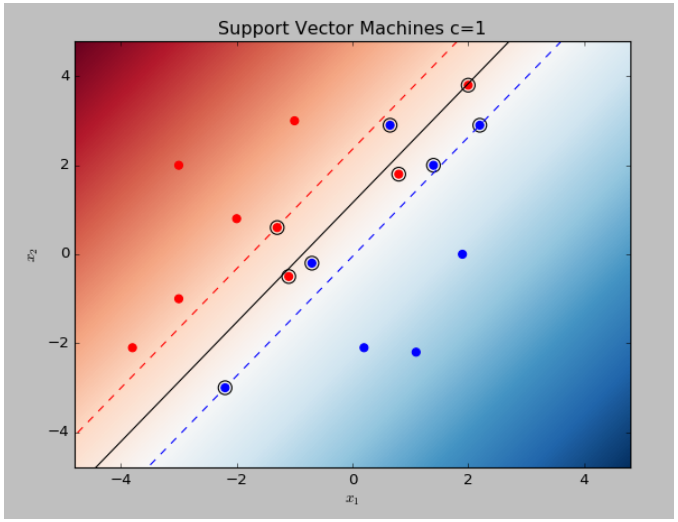
Błąd uczenia

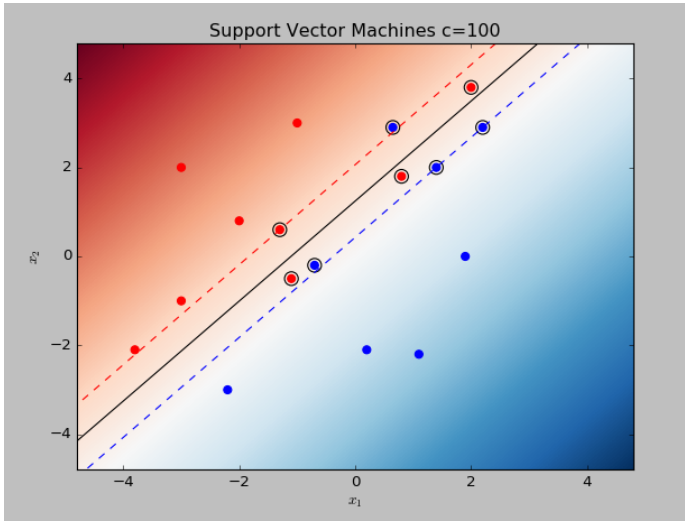
Dla modelu z miękkimi marginesami drugi składnik ma postać funkcji składanej z dwóch półprostych (z „zawiasem”). Nachylenie półprostej zależy od stałej C

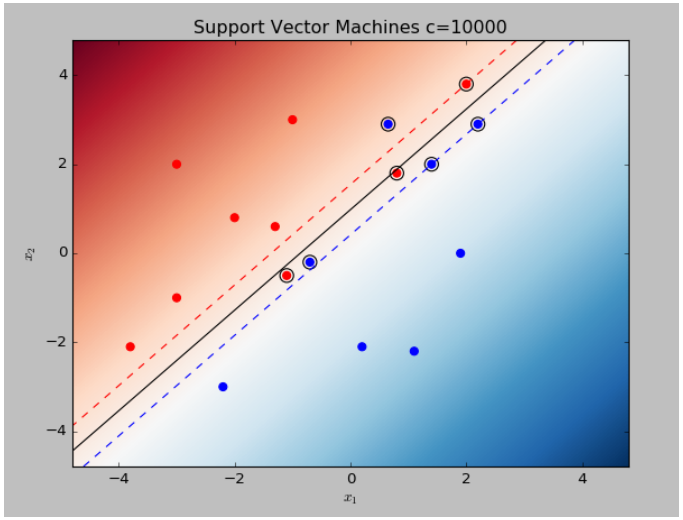


Przykłady


Support Vector Machines c=0.1
x_1
x_2


Support Vector Machines c=1
x_1
x_2


Support Vector Machines c=100
x_1
x_2


Support Vector Machines c=10000
x_1
x_2

Optymalizacja

- Wprowadzenie stałej C w funkcji celu w niewielkim stopniu zmienia dualną postać problemu (dodatkowe ograniczenie)

$$\max_{\alpha} J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

przy ograniczeniach: $0 \leq \alpha_i \leq C$ oraz $\sum_{i=1}^{m} \alpha_i y_i = 0$

- Optymalizacja
  - Ze względu na ograniczenie $\sum_{i=1}^{m} \alpha_i y_i = 0$ nie jest możliwa optymalizacja wzdłuż współrzędnych, jeśli $m - 1$ współczynników jest ustalonych, nie można optymalizować m-tego [Patrz dla porównania metoda Gaussa Seidla/coordinate descent]
  - Muszą być optymalizowane co najmniej dwa współczynniki na raz

Najbardziej popularnym algorytmem jest SMO (Sequential minimal optimization)
while (nie stop):

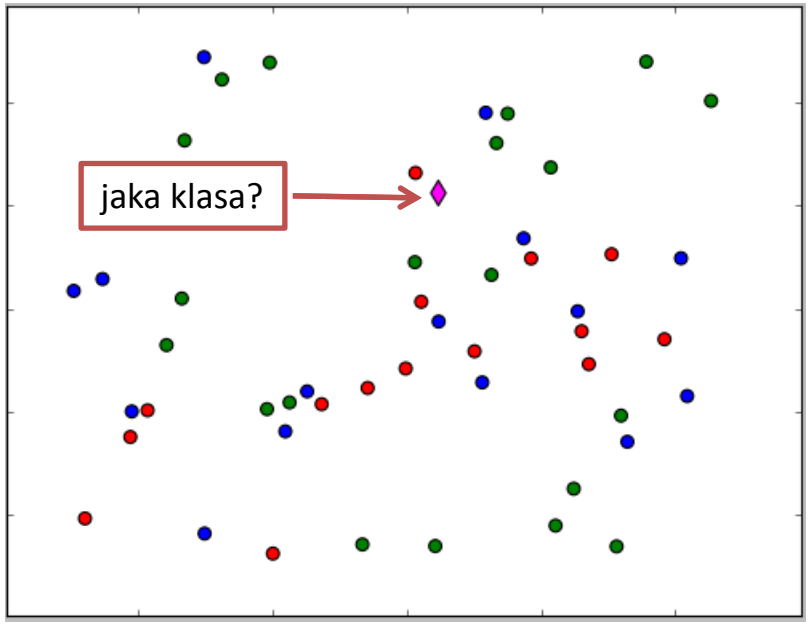
1. Wybierz parę α_i i α_j naruszającą ograniczenia (metoda wyboru heurystyczna).
2. Popraw wartość $J(\alpha)$, dla zmiennych α_i i α_j, **zachowując pozostałe współczynniki stałe**. W takim przypadku $J(\alpha)$ jest zwykłą funkcją kwadratową jednego ze współczynników α_i lub α_j

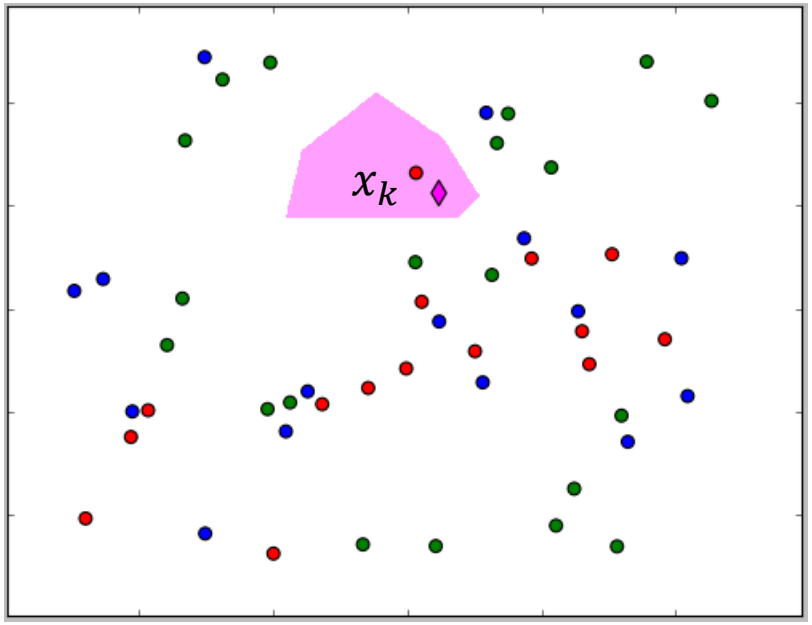
Patrz np: http://brain.fuw.edu.pl/edu/index.php/Uczenie_maszynowe_i_sztuczne_sieci_neuronowe/Wyk%C5%82ad_9#Algorytm_SMO_-_sekwencyjnej_minimalnej_optymalizacji

kNN
Algorytm najbliższych sąsiadów

Algorytm najbliższych sąsiadów

Algorytm najbliższych sąsiadów opiera się na prostym pomyśle przewidywane wartości wyjściowe mogą być ustalane na podstawie podobieństw do obserwacji w zbiorze uczącym.

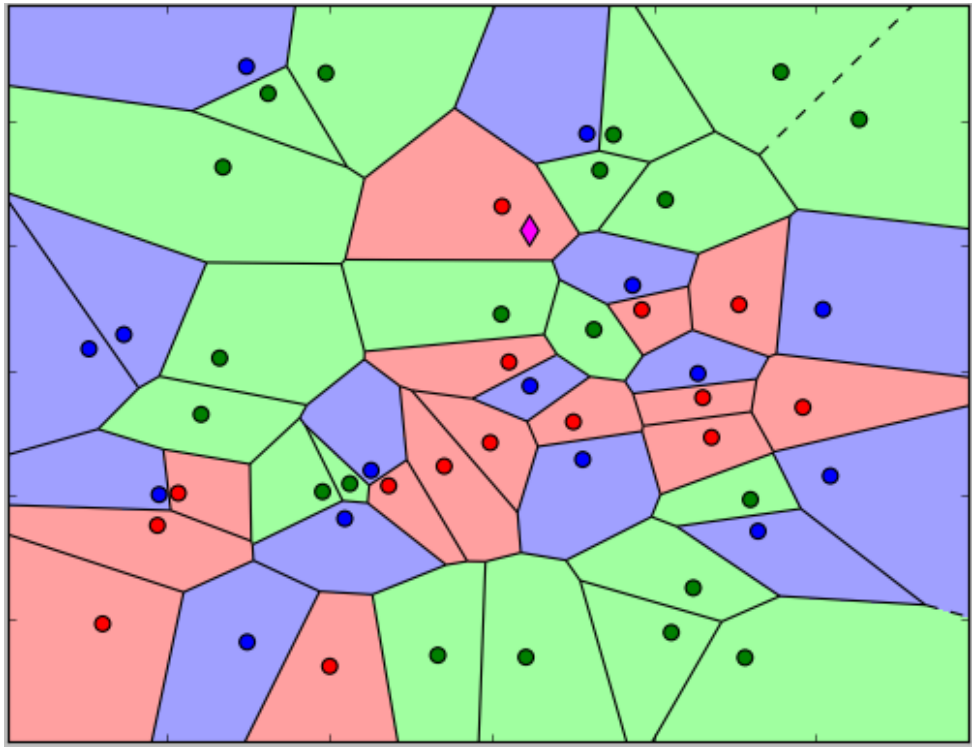




- Na przykład etykieta klasy dla nieznanej obserwacji (na rys. oznaczonej rombem) może być ustalona na podstawie etykiety najbliższego sąsiada.
- Po prawej zbiór punktów $V(x_k)$ których odległość od pozostałych obserwacji x_i jest większa od odległości od x_k.
- Obszar $V(x_k) = \{x \in \mathrm{R}^2 : \forall i \,.\, d(x, x_k) \leq d(x, x_i)\}$ nazywany jest komórką Woronoja (ang. Voronoi cell)

Diagram Woronoja

- Podział przestrzeni na komórki Woronoja nazywany jest diagramem Woronoja (ang. Voronoi tessalation)
  - Granice pomiędzy dwoma obszarami wyznaczają symetralne odcinków łączących dwa punkty (w 2D)
  - W przestrzeni wielowymiarowej komórki są wielobokami wypukłymi, a ich ściany elementami hiperpłaszczyzn



Odległość

Odległość (metryka): to funkcja $d: X \times \mathrm{X} \to R_0^+$ spełniająca 3 warunki

1. Identyczność $d(x, y) = 0 \Leftrightarrow x = y$
2. Symetria $d(x, y) = d(y, x)$
3. Nierówność trójkąta: $d(x, z) \leq d(x, y) + d(y, z)$

Przykłady:

- Metryka Minkowskiego ($X = R^n$):

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- Metryka Manhattan ($p = 1$): $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- Metryka Euklidesa ($p = 2$): $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
- Metryka Czebyszewa ($p = \infty$): $d(x, y) = \max_{\mathrm{i}} |x_i - y_i|$

Podobieństwo

- Podobieństwo sim: $X \times X \rightarrow R$ jest funkcją o charakterze przeciwnym do odległości (dla blisko położonych obiektów, $d(x,y) = 0$, $sim(x,y)$ przyjmuje duże wartości.
- Często podobieństwo jest skalowane do przedziału [0,1].
- Podobieństwo mające własność **rozróżnialności** obiektów spełnia:
  1. $sim(x,y) = 1 \Leftrightarrow x = y$
  2. $sim(x,z) \geq sim(x,y) + sim(y,z) - 1$

Przykład: dla $x, y \in \mathrm{R}^n$ podobieństwo cosinusowe:

$$sim(x,y) = \frac{x^T y}{\|x\|\|y\|}$$

Nie zapewnia rozróżnialności obiektów

Podobieństwo może zostać przekształcone w (pół)metrykę :

$$d(x,y) = 1 - sim(x,y)$$

Warunek trójkąta nie musi być spełniony (stąd półmetryka, ang. semimetrc).

Podobieństwo może być zdefiniowane dla różnego typu obiektów bez konieczności wyznaczania ich cech w R^n: ciągów symboli, zbiorów, grafów, itd.

Funkcje jądra (kernele) $K(x,y)$ mogą być postrzegane jako rodzaj podobieństwa.

Różne miary odległości/podobieństwa

- Dla zmiennych binarnych lub nominalnych

$$d(x_i, x_j) = 1 - I(x_i, x_j),$$

gdzie $I(a, b)$ to tzw. funkcja wskaźnikowa: $I(a, b) = \begin{cases} 0, \text{gdy } a \neq b \\ 1, \text{gdy } a = b \end{cases}$

- Dla zmiennych będących zbiorami:
  - odległość to tzw. odległość Hamminga

$$d(x_i, x_j) = |(x_i \backslash \mathrm{x}_j) \cup (x_j \backslash \mathrm{x}_i)|$$

  - podobieństwo definiuje się jako indeks Jaccarda:

$$sim(x_i, x_j) = \frac{|x_i \cap x_j|}{|x_i \cup x_j|}$$

- Dla ciągów symboli (np. tekstów) $x_j = (x_{11}, \ldots, x_{1k})$, $x_2 = (x_{21}, \ldots, x_{2m})$, gdzie $x_{1j}, x_{2i} \in A$ odległość Levenshteina to najmniejsza liczba edycji: usunięć, zamian, przestawień przekształcających ciąg x_1 w x_2.

Brakujące dane

- Przy obliczaniu odległości pomiędzy obserwacjami x_i i x_j może zdarzyć się, że dla pewnego atrybutu k dane nie są określone (reprezentowane przez symbol '?').
  Czyli na przykład $x_{ik} =?$
- Strategia zachowawcza:
  - Dla zmiennych nominalnych zawsze $I(?,\cdot) = I(\cdot,?) = 1$
  - Dla zmiennych numerycznych, gdy $x_{ik} =?$ wybierana jest najbardziej odległa w stosunku do x_{jk} wartość ze zbioru uczącego

    $$x_{ik} \leftarrow \arg \max_{x_k \in \mathrm{X}(k)} \{|x_k - x_{jk}|\}$$

    gdzie $X(k) = \{x_{ik} : j = 1, \ldots, m\}$
  - Jeśli obie wartości numeryczne są nieokreślone, składnik odległości to:

    $$\sup X(k) - \inf X(k)$$
- Strategia optymistyczna
  - Dla zmiennych nominalnych zawsze $I(?,\cdot) = I(\cdot,?) = 1$, ale $I(?,?) = 0$ (równoważne dodaniu dodatkowej zmiennej nominalnej ?)
  - Zastąpienie brakujących wartości średnimi:

    $$x_{ik} \leftarrow mean(X[k])$$
  - Jeśli obie wartości nieokreślone, ich różnica wynosi 0

K najbliższych sąsiadów

- Niech $D = \{(x_i, y_i): i = 1, \dots, m\}$ będzie zbiorem danych (uczącym)
- Dla danej nieznanej obserwacji x uporządkujmy D w postaci ciągu $D(x) = ((x_j, y_j): j = 1, \dots, m)$, takiego, że

$$\forall\, j = 1, m-1.\, d\big(x, x_j\big) \leq d(x, x_{j+1})$$

- Pierwszych k wyrazów ciągu $D(x)$ definiuje $N_k(x)$, czyli k-elementowe sąsiedztwo x:

$$N_k(x) = \{(x_k, y_k), \dots, (x_k, y_k)\}$$

- Dla zagadnienia regresji przewidywana wartość wyjściowa jest wartością średnią y w sąsiedztwie:

$$\hat{y}(x) = \frac{1}{k} \sum_{(x,y) \in N_k(x)} y$$

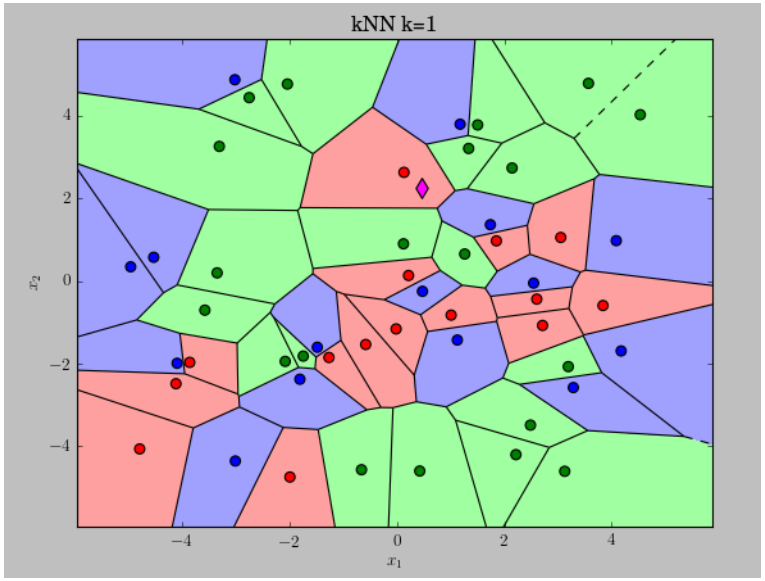
- Dla zagadnienia klasyfikacji:

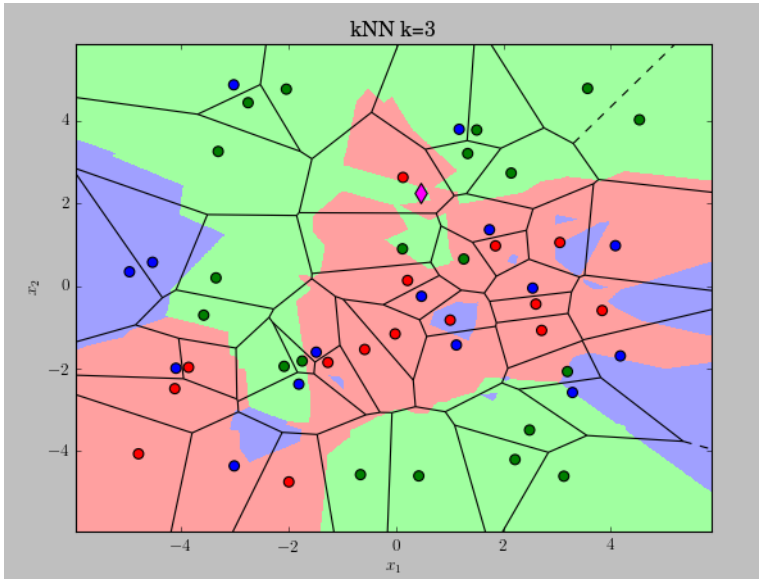
$$p(Y = y'|x) = \frac{1}{k} \sum_{(x\prime, y\prime) \in N_k(x)} I(Y = y')$$

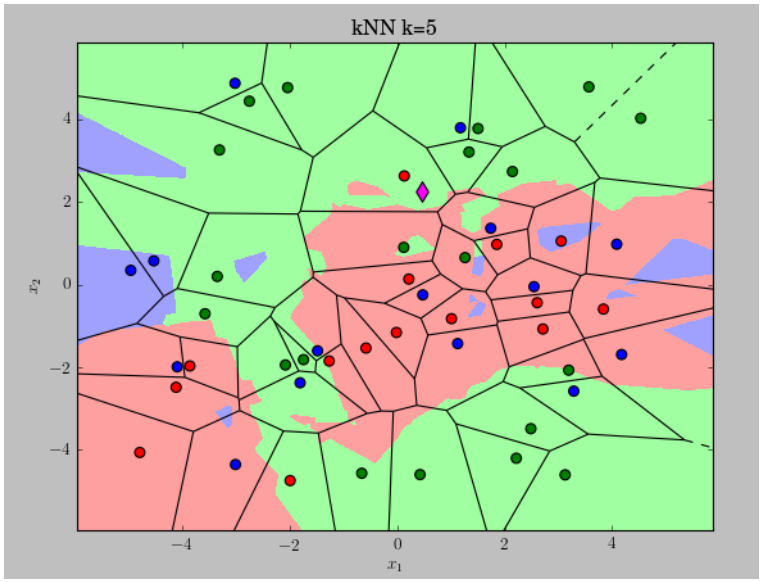
$$\hat{y}(x) = \arg\max_{y'} p(Y = y'|x)$$

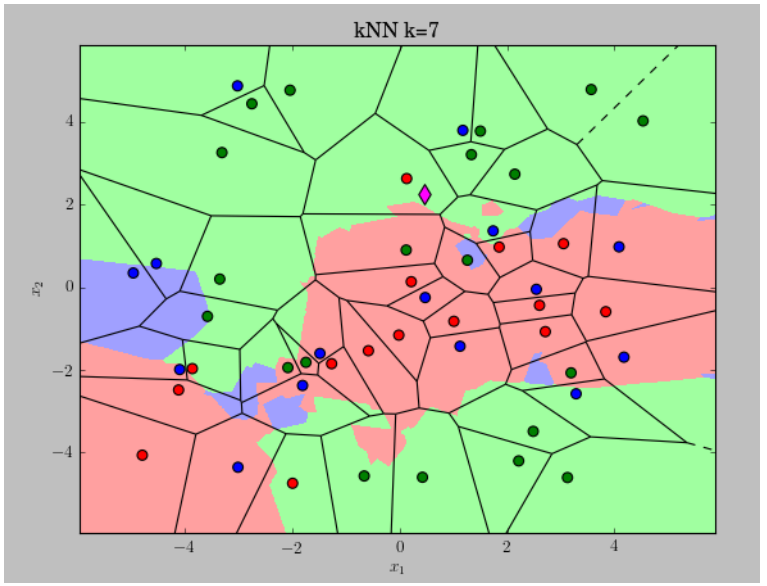
Czyli przewidywaną klasą jest etykieta przeważająca wśród sąsiadów w $N_k(x)$.

Przykłady







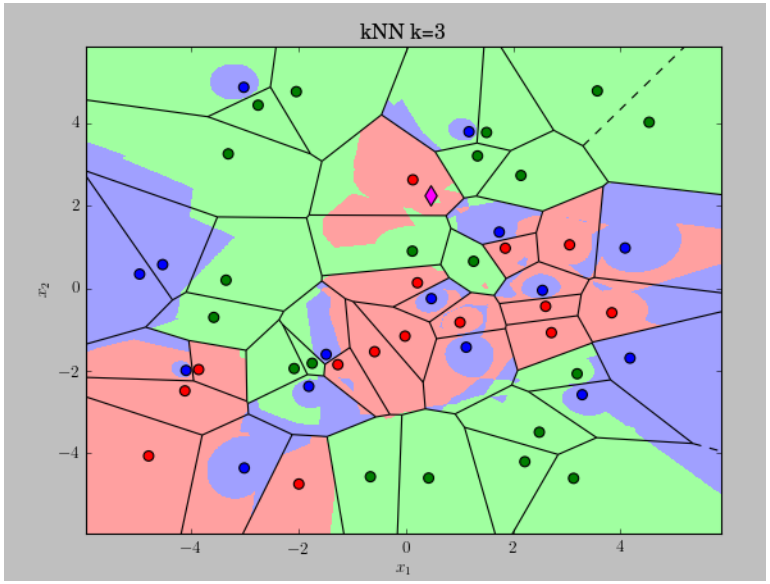


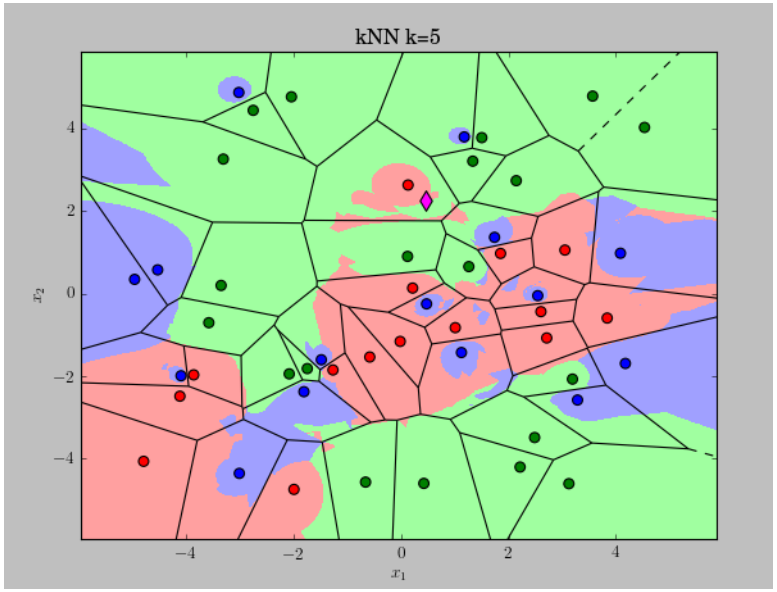
Wagi

Idea wprowadzenia wag polega na uwzględnieniu odległości klasyfikowanej obserwacji x od obserwacji w zbiorze najbliższych sąsiadów $N_k(x)$

$$p(Y = y'|x) = \frac{\sum_{(x\prime,y\prime)\in N_k(x)} w(x,x')I(Y = y')}{\sum_{(x\prime,y\prime)\in N_k(x)} w(x,x')}$$

Waga $w(x,x')$ jest najczęściej odwrotnością odległości $w(x,x') = \frac{1}{d(x,x')}$ lub funkcją $w(d(x,x'))$ zdefiniowaną przez użytkownika.





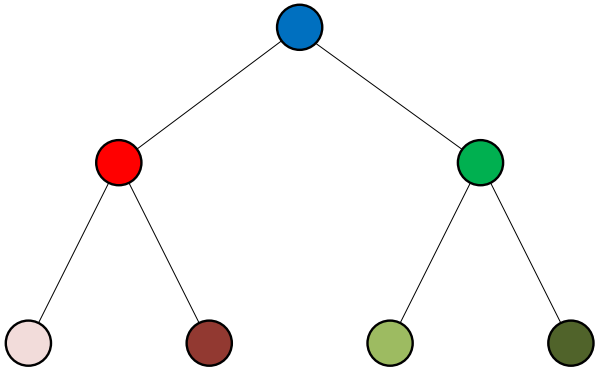
Zmiana regionów decyzyjnych po zastosowaniu wag będących odwrotnością odległości

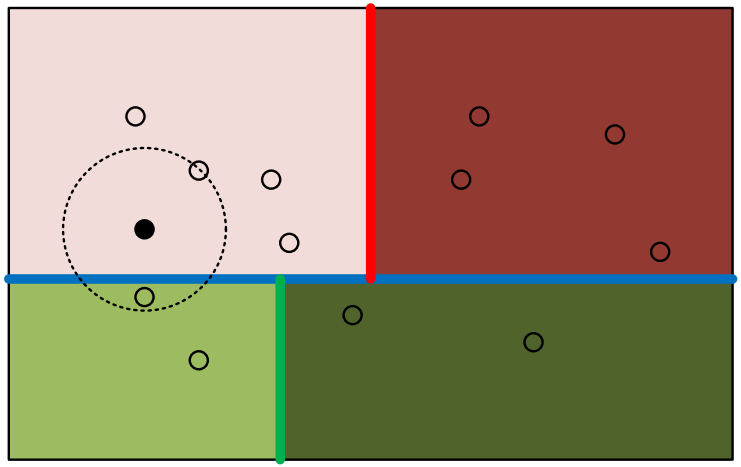
Wyszukiwanie najbliższych sąsiadów

- Czas uczenia algorytmu kNN jest zerowy (model jest po prostu zbiorem instancji)
- Czas wyszukiwania najbliższych sąsiadów jest proporcjonalny do wielkości zbioru uczącego $O(k \cdot n \cdot \boldsymbol{m})$. Działanie algorytmu może być wolne, jeżeli m jest duże, np. 16^6, obserwacje przechowywane są w bazie danych itd.
- Metody przyspieszania wyszukiwania:
  - Reprezentacja zbioru uczącego w postaci drzew: kD-tree lub ball tree
  - Heurystyki ograniczające obliczanie odległości
  - Ograniczanie zbioru analizowanych obserwacji

kD-tree (k-wymiarowe drzewo)

- kD-tree jest drzewem binarnym, w którym kolejne węzły reprezentują podziały przestrzeni wzdłuż jednego z wymiarów

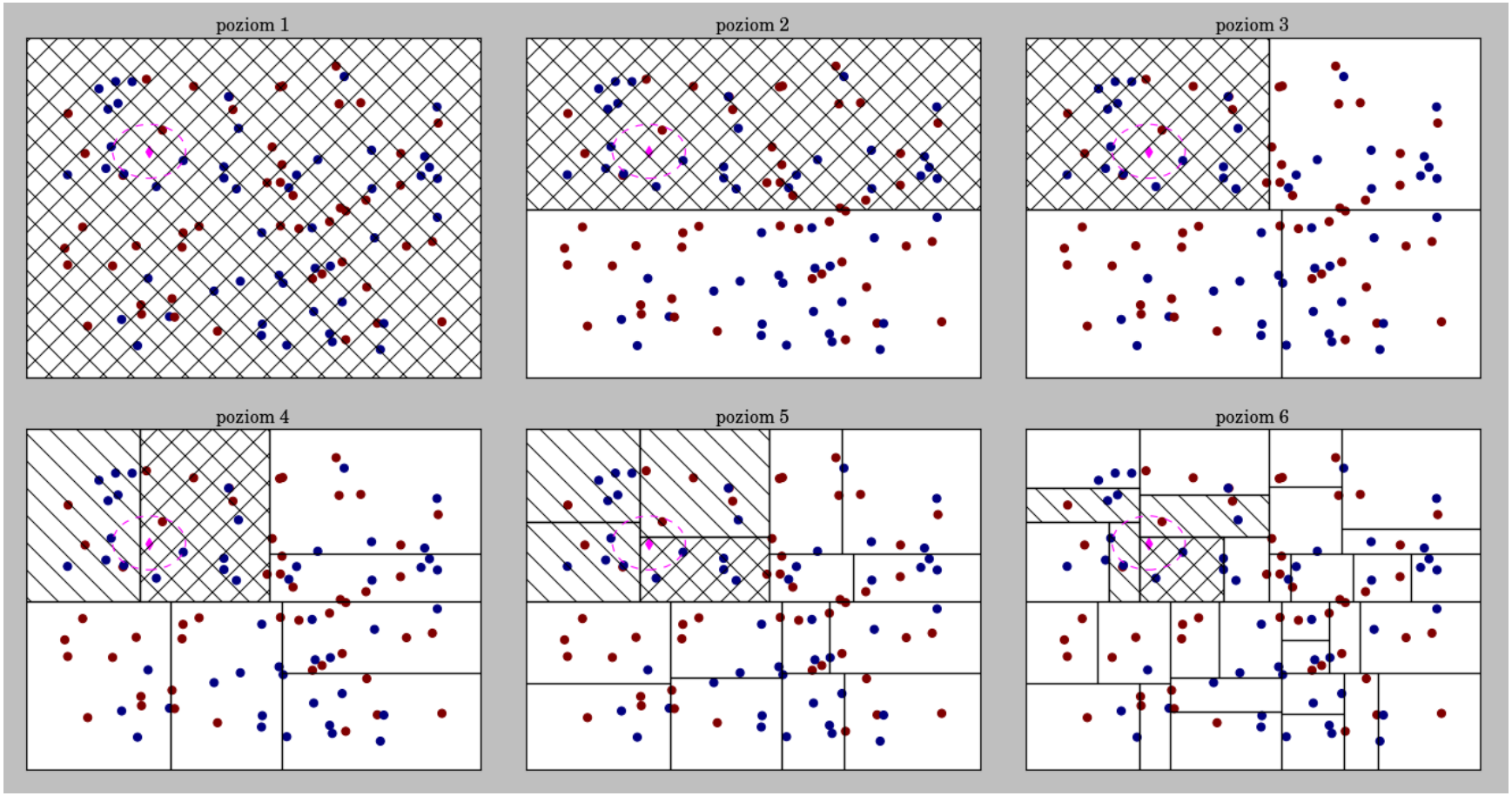




Algorytm wyszukiwania najbliższych sąsiadów polega na:

- przejściu przez drzewo od korzenia do liścia (bloku, który zawiera badany punkt)
- znalezieniu najbliższego sąsiada
- sprawdzeniu, czy kula S o promieniu równym odległości od najbliższego sąsiada przecina węzeł bliźniaczy do liścia
- sprawdzenie przecięć dla węzłów bliźniaczych względem węzłów nadrzędnych i ich potomków (różowy $\rightarrow$ czerwony $\rightarrow$ zielony $\rightarrow$szarozielony)

Przykład



Ostatecznie przeszukane zostaną kreskowane obszary poziomu 6
(Rysunek na podstawie kodu zaczerpniętego z http://www.astroml.org/)

Konstrukcja kD-tree

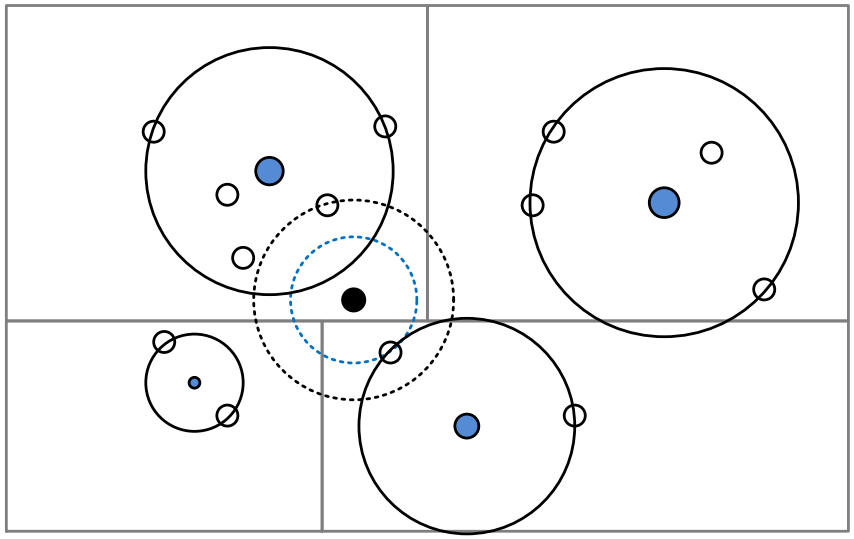
Algorytm rekurencyjny $kdtree(D\colon dane, N\colon node)$:

1. Jeżeli liczba punktów w węźle $|D| \leq \min$ stop
2. Wybór kierunku podziału j: cyklicznie dla kolejnych wymiarów lub w kierunku największej zmienności wartości
3. Podziel dane węzła na dwie części $D_1 = \{x \in D\colon x_j < v_j\}$ i $D_2 = \{x \in D\colon x_j \geq v_j\}$.

   Wartość podziału v_j ustala się poprzez
   - posortowanie danych i wyznaczenie mediany wartości
   - jako wartość średnią
4. Utwórz dwa węzły potomne N_1 i N_2 i wywołaj $kdtree(D_1, N_1)$ i $kdtree(D_2, N_2)$

Ball tree

- W drzewie Ball tree węzły reprezentują kule otaczające punkty w przestrzeni n-wymiarowej
- Ze względu na kształt bloków przypisanych węzłom – łatwiej jest wykluczyć przecięcia i przez to w rzeczywistych zastosowaniach (zwłaszcza dla większych wymiarów przestrzeni) złożoność wyszukiwania spada



Porównanie liczby przecięć dla kD tree (czarna linia przerywana, 4 obszary) i Ball tree (niebieska linia przerywana dwie kule)

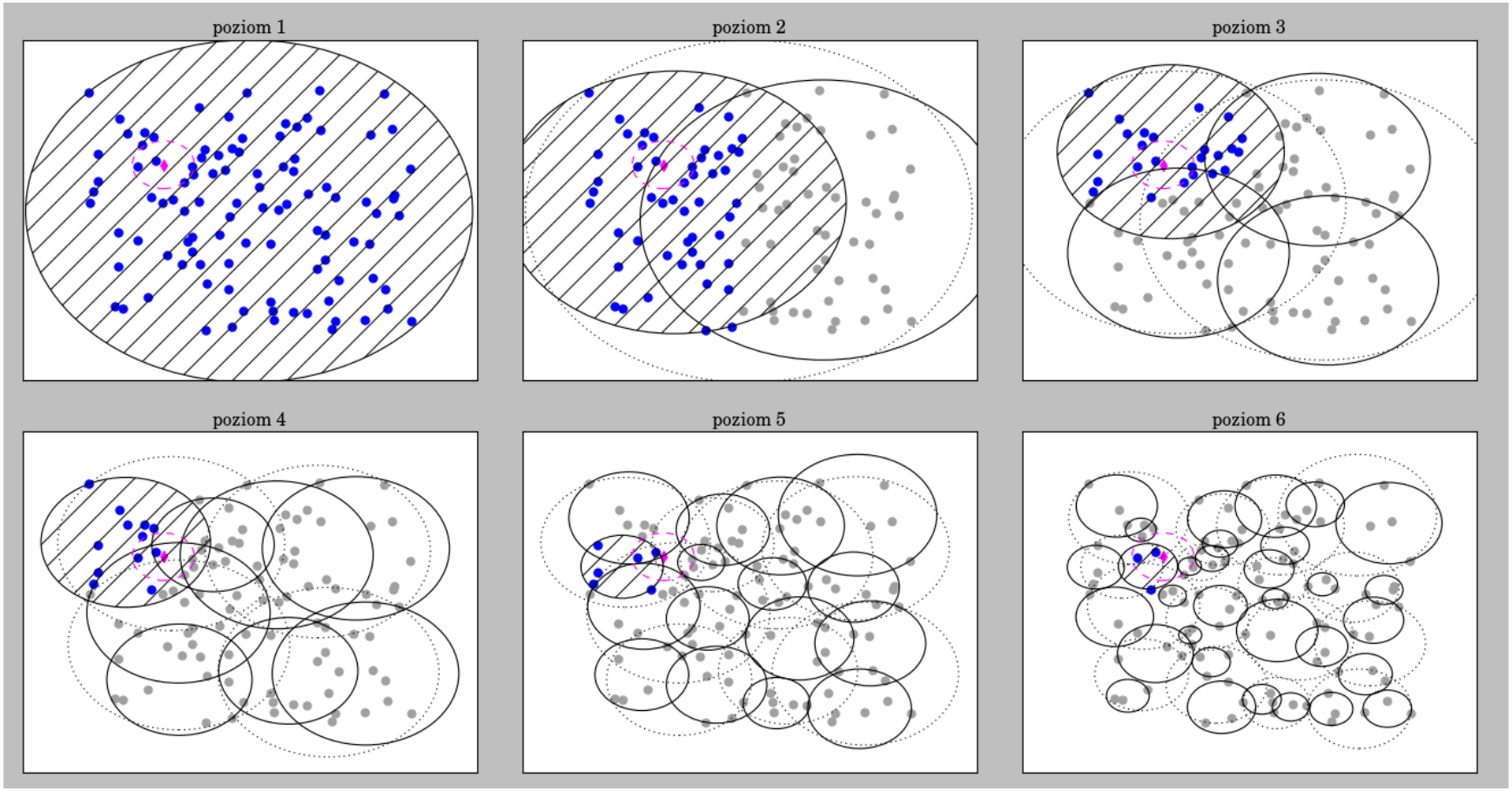
Ball tree: algorytm wyszukiwania

Algorytm przeszukuje rekurencyjnie drzewo składując znalezione punkty w kolejce sortowanej malejąco według odległości.

Odcinane są kule, których środek jest bardziej odległy niż najdalszy punkt w kolejce.

$search(x, q: queue, n: node)$:
  if $d(x, n.center) \geq d(x, q.first())$: return
  if($leaf(n)$):
    przeszukaj punkty x' w n i dodaj do q, jeśli $d(x, x') < d(x, q.first())$
  else:
    $search(x, q, node.left)$
    $search(x, q, node.right)$

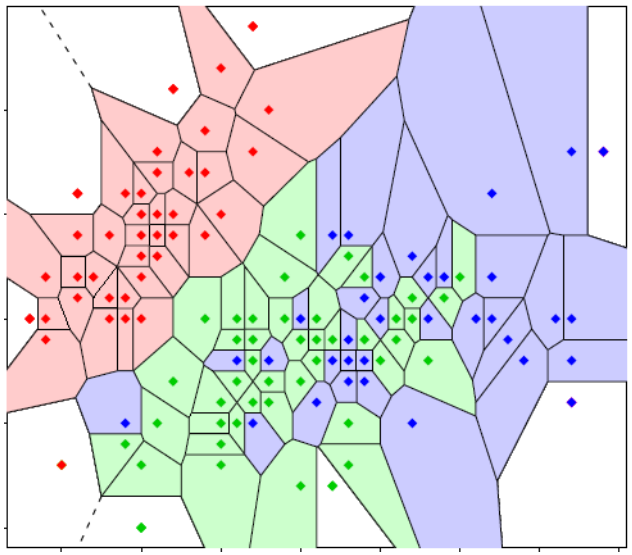
Ball tree - przykład

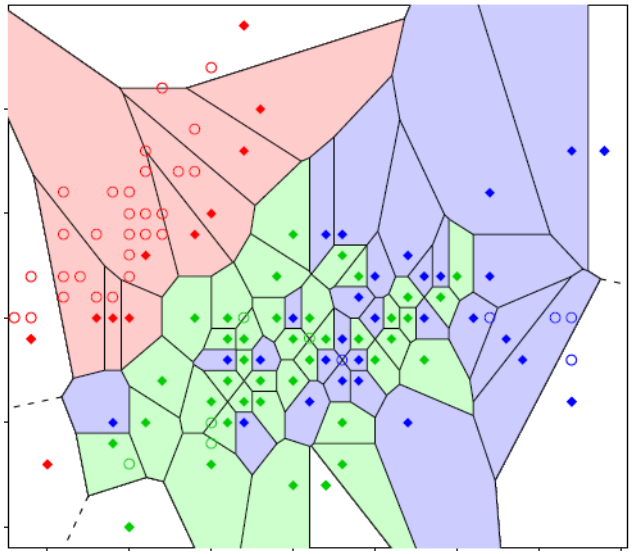


Przeszukane zostaną kule przecinające się z różową elpsą…

Inne opcje przyspieszenia obliczeń

- Częściowe wyznaczanie odległości
  - Dla $r < n$ oblicza się: $d_r(x, x') = \sum_{i=1}^{r}(x_i - x'_i)^2$.
  - Jeżeli $d_r(x, x') > \sup\{d(x, x''): x'' \in N_k(x)\}$ (czyli częściowo wyznaczona odległość jest większa, niż do najdalszego sąsiada w bieżącym otoczeniu) – nie ma sensu dalej prowadzić obliczeń.
- Redukcja zbioru obserwacji bez zmiany granic regionów decyzyjnych (**editing**)





Źródło: https://www.ismll.uni-hildesheim.de/lehre/ml-07w/skript/ml-2up-03-nearest-neighbor.pdf

Typowe opcje kNN

Biblioteki implementujące kNN zazwyczaj oferują następujące opcje:

- Przeszukanie wszystkich elementów (*brute force*).
  Złożoność czasowa $O(n \cdot m)$
- Ball tree
  Złożoność czasowa $O(n \cdot \log m)$
- kD-tree
  - Dla małych $\mathrm{n} \approx 20$, złożoność rzędu $O(n \cdot \log m)$
  - Dla dużych n wzrasta do $O(n \cdot m)$ i ze względu na czas/zużycie pamięci przy budowie drzewa może być mniej efektywne niż *brute force*

Dodatkowe informacje i dyskusja zależności od:

- parametru k
- progu określającego minimalną liczbę obserwacji w węźle
- potencjalnej liczby zapytań

http://scikit-learn.org/stable/modules/neighbors.html