{ file w11svm.tex, 09.01.2010 }

11 Kernele, sieci SVM i sieci GDA

W rozdziale tym rozważamy nastepujące zagadnienia:
(a) Co to są funkcje jądra (kernel functions) $K(\mathbf{x}, \mathbf{y})$ i do czego mogą służyć.
(b) Metoda SVM. Jak dla danych separabilnych (2 grupy danych) skontruować liniową funkcję dyskryminacyjną określoną w jednoznaczny sposób wg zasady wyznaczania najszerszego pasa granicznego między obu grupami danych.
(c) Jak uogólnić metodę SVM opracowaną dla danych separabilnych na przypadek danych nieseparabilnych.
(d) Jak za pomocą specyficznych przekształceń $\phi(\mathbf{x})$ przejść z przestrzeni obserwowanych danych R^d do przestrzeni danych przekształconych $\mathcal{F}$ (feature space) i tam wyznaczać liniową funkcję dyskryminacyjną ('the kernel trick').
(e) Metoda GDA. Jak wizualizować dane dla kilku grup budując nowe zmienne oparte o kryterium Fishera - które prowadzi do $G-1$ kanonicznych funkcji dyskryminacyjnych (G - liczba klas), i w jakim względzie są tu pomocne iloczyny skalarne przekształceń $\phi(\mathbf{x})$ spełniających warunek Mercera.

Bardzo często będziemy posługiwać się iloczynem skalarnym dwóch wektorów (dot product) oznaczanym jak niżej:

Niech $\mathbf{a}, \mathbf{b} \in R^n$ oznacza dwa wektory danych. Iloczynem skalarnym $(\mathbf{a} \cdot \mathbf{b})$ tych wektorów nazywamy wyrażenie

$$(\mathbf{a} \cdot \mathbf{b}) = \mathbf{a}^T \mathbf{b} = \sum_{i=1}^{n} a_i b_i. \tag{11.1}$$

Iloczyn skalarny zdefiniowany w powyższy sposób jest w Matlabie obliczany za pomocą funkcji `dot` znajdującej się w podstawowym oprogramowaniu Matlabowskim.

11.1 Przekształcenia kernelowe K(x,y) - kernels

Trochę historii

Już od dawna zauważono w zagadnieniach regresyjnych i klasyfikacyjnych, że można otrzymać bardziej efektywne rozwiązanie, jeżeli oprócz (zamiast) originalnego wektora danych $\mathbf{x} \in R^d$ będzie rozpatrywany rozszerony wektor $\mathbf{z} = \phi(\mathbf{x})$ zawierający m składowych, gdzie $m \geq d$. Przykładem jest tu przekształcenie wielomianowe stopnia p, dołączające do obserwowanych zmiennych $x_1, x_2, \ldots, x_d$ ich potęgi aż do stopnia $p1$ lub iloczyny mieszane postaci $x_i^{q1} x_j^{q2}$, $q1 + q2 \leq p1$. Innym przykładem jest zastosowanie radialnych funkcji bazowych omawiane przy okazji sieci typu RBF. Przez zastosowanie przekształcenia $\phi(\mathbf{x})$ przechodzimy z przestrzeni obserwowanych zmiennych (R^d) do (na ogół) rozszerzonej przestrzeni $\mathcal{F}$ o wymiarze m nazywanej *Feature space*, czyli przestrzenią zmiennych przekształconych.

Zauważono również, że bardzo często rozwiązania zagadnień regresji i korelacji zależą od macierzy kowariancji rozpatrywanych zmiennych, zebranych w postaci tablicy danych

$\mathbf{X} = (x_{ij})$. Jak wiadomo, macierze kowariancji oblicza się z iloczynów skalarnych otrzymywanych jako $\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$, przy czym $\tilde{\mathbf{X}}$ jest wystandaryzowaną (na średnią równą zero) macierzą danych; a wtedy obliczenia mogą być wykonane za pomocą funkcji dot zdefiniowanej wzodem 11.1, wykonanych na wektorach kolumnowych macierzy $\tilde{\mathbf{X}}$.

Podstawowe definicje i twierdzenia

Przechodzimy teraz do bardzo ważnego **pojęcia kernela**, ang. kernel function. Pojęcie to było używane więcej niż 50 lat temu w kontekscie czysto teoretycznych badań liniowych przestrzeni wektorowych, przestrzeni Banacha i przestrzeni Hilberta [2]. W dalszym ciągu przytaczamy definicje z tego źródła. Polecamy również slajdy Lamperta i Maschko z tutorialu prowadzonego na konferencji z Computer Vision w czerwcu 2009 r. [8].

Niech X oznacza niepusty zbiór wektorów danych.
• Funkcję $\psi : \; X \times X \mapsto R$ nazywamy dodatnio określonym kernelem (p.d. Mercel kernel), wtedy i tylko wtedy gdy

$$\sum_{i=1}^{n}\sum_{k=1}^{n} c_j c_k \psi(\mathbf{x}_j, \mathbf{x}_k) \; \geq 0$$

dla wszystkich $n \in N$, $\mathbf{x_1}, \ldots, \mathbf{x_n} \subseteq X$, oraz $c_1, \ldots, c_n \subseteq R$.

- Iloczyn skalarny jest przykładem kernela Mercerowskiego.
- Dla każdego dodatnio określonego kernela $\psi(\mathbf{x}, \mathbf{y})$ zachodzi następująca nierówność:

$$|\psi(\mathbf{x}, \mathbf{y})|^2 \leq \psi(\mathbf{x}, \mathbf{x})\psi(\mathbf{y}, \mathbf{y}).$$

Na bazie zdefiniowanego kernela ψ możemy zdefiniować pseudonormę: $||\mathbf{x}||_\psi \doteq \sqrt{\psi(\mathbf{x}, \mathbf{x})}$.

Następne twierdzenie pokazuje, jak z kernela Mercera możemy utworzyć iloczyn skalarny.

• Twierdzenie. Niech $K(\mathbf{x}, \mathbf{y})$ oznacza funkcję symetryczną dwóch wektorów będącą kernelem, taką że $\forall \mathbf{x}, \mathbf{y} \in X, \; X \subseteq R$. Wtedy możemy określić przekształcenie $\phi : \; X \mapsto \mathcal{F}$, takie że

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y}).$$

Przestrzeń $\mathcal{F}$, do której następuje mapowanie, jest nazywana przestrzenią zmiennych przekształconych (**Feature space**).

W ten sposób kernel K indukuje mapowanie bezpośrednie (*explicit mapping*) – gdy mapowanie ϕ jest znane, oraz mapowanie pośrednie (implicit mapping), gdy ϕ nie jest znane.
Przytoczone twierdzenie mówi, że przekształcenie Φ istnieje. W dowodzie wykorzystuje się rozkład *SVD* oraz własność dodatniej określoności macierzy $K = (k_{ij})_{i,j=1}^{n}$.

Tak więc Φ jest funkcją mapującą

$$\phi : \quad R^n \mapsto \mathcal{F}.$$

Własność ta jest podstawową tzw. triku kernelowego (*kernel trick*) używanego w metodzie SVM, GDA i in. Oddajemy głos Camastrze [1]:

> ... map the data into some other scalar product space (feature space) $\mathcal{F}$ by means of a nonlinear mapping like the above, and perform the linear algorithm (like decision boundary for 2 classes) in the feature space $\mathcal{F}$. In many cases the

> mapping ϕ cannot be explicitly computed, due to the high-dimensionality of $\mathcal{F}$. But this is not an obstacle, when the decision function requires the evaluation of scalar products $\phi(\mathbf{x}) \cdot \phi(\mathbf{y})$, and not the pattern $\phi(\mathbf{x})$ in explicit form. ...

Zastosowanie kerneli w analizie dyskryminacyjnej

Mamy dwie grupy (klasy) danych i chcemy wyznaczyć funkcję dyskryminacyjną która pozwoliłaby nam wyznaczyć granicę miedzy obu klasami. Najprostszą funkcją jest funkcja liniowa $g(x) = w \cdot x + b$, ale okazuje się, że taka funkcja może być nieskuteczna, gdy granica między obu grupami – rozpatrwanymi w przestrzeni R^d (obserwowanych zmiennach) jest nieliniowa.

Jeśli spodziewamy się nieliniowej granicy między grupami danych, możemy postąpić następująco: Mapujemy dane za pomocą przekształcenia ϕ do przestrzeni Hilberta $\mathcal{F}$ z iloczynem skalarnym zdefiniowanym jako

$$K(\mathbf{x}, \mathbf{y}) \doteq \phi(\mathbf{x}) \cdot \phi(\mathbf{y}). \tag{11.2}$$

Rzważane zagadnienie rozwiązujemy w przestrzeni $\mathcal{F}$ stosując tam proste algorytmy liniowe. Jest wiele doniesień z literatury, że taki sposób postępowania jest bardzo skuteczny, a autorzy tych doniesień otrzymywali dobre rozwiązania problemów w takich przypadkach, gdy inne tradycyjne metody – liniowe lub nieliniowe – po prostu zawiodły.

Przy stosowaniu kerneli ważny jest wybór kernela odpowiedniego typu. Wybieramy takie mapowanie, dla którego nie trzeba obliczać bezpośrednio mapowanych danych $\phi(\mathbf{x})$, ale dla którego w łatwy sposób można obliczyć w $\mathcal{F}$ tzw. kernele (czyli wyrażenia postaci 11.2) na podstawie posiadanych obserwacji z przestrzeni R^d. Szeroko stosowanymi przekształceniami o tej własności są: *kernel Gaussowski* i *kernel wielomianowy*, omawiane poniżej.

Kernel Gaussowski ('Gaussian') ma postać:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left\{ -\frac{(\mathbf{x} - \mathbf{y})^2}{2\sigma^2} \right\}. \tag{11.3}$$

Kernel ten ma jeden parametr `sigma`.

Kernel wielomianowy (polynomial, 'poly') ma postać

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^{p1}. \tag{11.4}$$

Kernel wielomianowy ma dwa parametry: `p1` (stopień wielomianu) i stałą `c>0` (wyraz wolny wielomianu). Jeżeli parametr c nie jest określony, to przyjmuje się domyślnie $c = 0$.

Przykłady innych kerneli można znaleźć np. w helpie do funkcji `svkernel` pakietu SVM Matlab Toolbox [3].

Przykład przekształcenia wielomianowego zaczerpnięty z [6].

Mamy 2 wektory danych: $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$.

Konstruujemy wektory zmiennych rozszerzonych za pomocą przekształcenia wielomianowego stopnia drugiego ($p1 = 2$):

$\Phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$

$\Phi(\mathbf{y}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$.

Wtedy okazuje się że iloczyn skalarny w przestrzeni zmiennych przekształconych można wyrazić jako funkcję iloczynu skalarnego zmiennych obserwowanych w R^d:

$(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) = (1 + \mathbf{x} \cdot \mathbf{y})^{\mathbf{2}}$.

11.2 SVM, Support Vector Machines – Wprowadzenie

Rozważana przestrzeń danych (measurement space) jest oznaczana symbolem Ω.

W przestrzeni tej znajdują się wektory danych $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T$, stanowiące tzw. próbkę uczącą $\mathcal{L}$ (learning sample), należące do dwóch klas oznaczanych jako ω_k, $k = 1, 2$. Dla wszystkich elementów próbki znamy ich przynależności klasowe $y_1, ..., y_l$, , gdzie $y_i \in \{1, -1\}$, $i = 1, \ldots, l$.

Szukamy reguł decyzyjnych pozwalających podzielić całą przestrzeń na dwa rozłączne obszary: $\Omega = \Omega_1 + \Omega_2$. Każdy z obszarów Ω_1, Ω_2 może składać się z kilku części, niekoniecznie przylegających do siebie.

Obszary Ω_1 i Ω_2 nie są znane dokładnie, ale uzyskaliśmy z nich próbkę uczącą $\mathcal{L}$ w postaci par $\{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, l$. Pierwszy składnik każdej pary zawiera zaobserwowany wektor danych $\mathbf{x}_i$, natomiast drugi składnik $\{y_i\}$ wykazuje wartości $+1$ lub -1 w zależności od tego, czy punkt $\mathbf{x}_i$ należy do klasy pierwszej czy też drugiej. Podział próbki na dwie grupy jest ostry (crisp); nie ma przynależności rozmytych. Na podstawie uzyskanej próbki danych chcemy zbudować granicę decyzyjną (decision boundary) między obu klasami. Granica ta będzie określana za pomocą funkcji decyzyjnej $g(\mathbf{x})$. Generalnie, funkcja $g(\mathbf{x})$ może być funkcją nieliniową swojego argumentu; jednak, w pierwszym podejściu, będzie nas interesować **liniowa** funkcja decyzyjna.

11.3 Koncepcja SVM dla danych liniowo separabilnych

Wprowadzamy pojęcie danych liniowo separabilnych.

Dane $\{\mathbf{x}_i\}$, $i = 1, \ldots, l$, $\mathbf{x}_i \in R^d$ należące do dwóch klas określonych zmiennymi $\{\mathbf{y}_i\}$, $i = 1, \ldots, l$ są **liniowo separabilne**, jeśli istnieje hiperpłaszczyzna $\mathcal{H}$ postaci $g(\mathbf{x})$

$$\boxed{\mathcal{H} : g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b} \tag{11.5}$$

przyjmująca wartości ($i = 1, \ldots, n$):

$$\begin{cases} g(\mathbf{x}_i) > 0, & \text{dla } \mathbf{x}_i \in \omega_1, \\ g(\mathbf{x}_i) < 0, & \text{dla } \mathbf{x}_i \in \omega_2, \end{cases} \tag{11.6}$$

Podkreślmy, że nierówności we wzorze (11.6) są ostre.

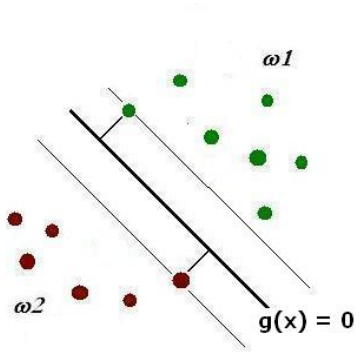
Zapytajmy się teraz, jak dla danych liniowo separabilnych (z założenia) zbudować liniową funkcję decyzyjną postaci (11.5). W przypadku danych separabilnych istnieje nieskończenie wiele takich funkcji które rozdzielają obie klasy. Jak wybrać tę jedyną, najlepszą, dającą szanse na najwyższy stopień generalizacji?

Pomysł na taką funkcję pochodzi of Vapnika [5], który zaproponował metodę nazwaną metodą SVM (wektorów nośnych). Idea SVM jest zilustrowana poniżej rysunkiem 11.1. Na rysunku tym widzimy pogrubioną linię prostą rozdzielającą dwie grupy punktów: zielonych (u góry) i czerwonych (u dołu). Linia ta jest miejscem geometrycznym punktów $\mathbf{x}$ spełniających warunek – porównaj wzory (11.5) i (11.6)

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0.$$

Z własności separabilności rozważanych punktów możemy taką prostą (hiperpłaszczyznę) pogrubić otaczając ją pewnym pasem nie zawierającym żadnego z punktów próbki uczącej. Pas ten jest nazywany marginesem.

Vapnik zaproponował konkretny algorym pozwalający na wyznaczenie wektora $\mathbf{w}$ oraz stałej b ze wzoru 11.6, gwarantujących osiagnięcie marginesu o maksymalnej szerokości. Na



Rysunek 11.1: Koncepcja SVM dla grup danych liniowo separabilnych. Należy znaleźć hiperpłaszczyznę rozdzielającą otoczoną marginesem o największej szerokości.

rysunku 11.1 margines ten jest zaznaczony dwoma cieńszymi liniami. Parametry szukanej hiperpłaszczyzny można wyrazić w takiej skali, żeby maksymalne marginesy były miejscem geometrycznym punktów $\mathbf{x}$ punktów spełniających warunki:

$$g(\mathbf{x}) = +1, \text{ gdy } \mathbf{x} \text{ leży od strony } \omega_1, \quad \text{oraz} \quad g(\mathbf{x}) = -1, \text{ gdy } \mathbf{x} \text{ leży od strony } \omega_2. \quad (11.7)$$

Hiperpłaszczyzny $g(\mathbf{x}) = +1$ lub $g(\mathbf{x}) = -1$ będziemy nazywać hiperpłaszczyznami marginesowymi.

Można stosunkowo łatwo wykazać (korzystając ze wzoru na odległość punktu $\mathbf{x}$ od hiperpłaszczyzny $\mathcal{H}$), że przy powyższych oznaczeniach i ograniczeniach maksymalna szerokość marginesu (M), tj. odległość między hiperpłaszczyznami $\mathbf{w}^T\mathbf{x} + b = +1$ oraz $\mathbf{w}^T\mathbf{x} + b = -1$ wynosi

$$M = \frac{2}{||\mathbf{w}||}.$$

Wynika stąd, że im krótszy wektor $\mathbf{w}$, tym większy (szerszy) jest maksymalny margines czyli pas oddzielający obie klasy punktów; wobec czego powinniśmy wyznaczać hiperpłaszczyznę $\mathcal{H}$ w oparciu o wektor $\mathbf{w}$ o możliwie małej normie.

Sformułowanie kryterium do wyznaczenia parametrów $\mathbf{w}$ i b

Nasze postulaty:

- Chcemy minimalizować wektor $\mathbf{w}$, bo wtedy otrzymamy największy margines.
- Dla punktów próbki uczącej wyznaczana wartość funkcji decyzyjnej ma dawać wartości ≥ 1 lub ≤ -1, bo wszystkie punkty mają leżeć na marginesach lub nazewnątrz pasa wyznaczonego przez oba marginesy. Chcemy żeby odchylenia punktów od odpowiadających im hiperpłaszczyzn marginesowych były generalnie większe od jedności (im większe, tym lepiej) jednak potrzebujemy conajmniej jednego punktu każdej klasy do zakotwiczenia marginesu (czyli punktu leżącego na danej hiperpłaszczyźnie).

Mamy więc zagadnienie optymalizacyjne:

$$\textit{minimizuj po } \mathbf{w} \textit{ wyrażenie}: \qquad \tau(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$$

przy warunkach ubocznych

$$y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad \forall\, \mathbf{x}_i \in \mathcal{L}.$$

Wypisane wyżej warunki prowadzą do uogólnionego zagadnienia optymalizacyjnego typu Lagrange'a, a rozpatrywane zagadnienie optymalizacjne możemy zapisać w postaci następującego kryterium [1]:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\,||\mathbf{w}||^2 \;-\; \sum_{i=1}^{l} \alpha_i \left[y_i \left((\mathbf{x}_i \cdot \mathbf{w}) + b \right) \;-\; 1 \right]. \tag{11.8}$$

Nowo wprowadzone zmienne $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)^T$ stanowią tzw. mnożniki Lagrange'a i spełniają nierówności: $\alpha_i \geq 0,\ \forall_i$.

Kryterium 11.8 (*Lagrangian1*) należy *minimalizować* ze względu na pierwszy składnik, a jednocześnie *maksymizować* ze względu na drugi składnik. Wynika stąd, że szukamy punktu siodłowego (saddle point) globalnego rozwiązania. Mówi się, że tego typu zagadnienia są na ogół źle uwarunkowane.

Sposób rozwiązywania takiego zagadnienia jest przedstawiony np. w [3, 1].

Przyrównując pochodne Lagrangianu (11.8) – względem $\mathbf{w}$ i b – do zera, otrzymujemy:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i\, \mathbf{x}_i y_i, \quad \text{oraz} \quad \sum_{i=1}^{l} \alpha_i\, y_i = 0. \tag{11.9}$$

Wstawiając otrzymane wyrażenia do (11.8) otrzymujemy, że szukana funkcja decyzyjna (klasyfikacyjna) $g(\mathbf{x})$ ma postać (patrz np. [1], [3])

$$g(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i \cdot (\mathbf{x}_i \cdot \mathbf{x}) + b \right) \tag{11.10}$$

Wzór ten pokazuje, że funkcja decyzyjna – wypisana *explicite* – zależy bezpośrednio od mnożników Lagrange'a α_i, $i = 1, \ldots, l$. Jednak, mnożniki te – na podstawie tzw. warunku (KKT) określanym mianem *Karush-Kuhn-Tucker complementarity conditions* – powinny spełniać następujący warunek:

$$\alpha_i \cdot [y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1] = 0, \;\; i = 1, \ldots, l. \tag{11.11}$$

Jest to możliwe tylko w dwóch przypadkach:
a) punkty próbkowe $\mathbf{x}_i$ leżą na marginesach – wtedy wartości α_i `mogą` być dowolne, oraz
b) punkty próbkowe $\mathbf{x}_i$ leżą poza marginesami – wtedy wartości α_i `muszą` być równe zeru.

Tak więc okazuje się, o rozwiązaniu decydują tylko te wektory próbki uczącej które leżą na hiperpłaszczyznach marginesowych, bo tylko dla nich wielkości α_i mogą być dodatnie (z dokładnością do błędów zaokrągleń).

Obserwacje $\mathbf{x}_i$ leżące hiperpłaszczyznach marginesowych noszą nazwę **wektorów nośnych** (**support vectors**), w skrócie *SV*.

Tak więc faktycznie wychodzi, że większość mnożników $\alpha_i,\ i = 1, \ldots l$, przyjmuje wartości zerowe (z dokładnoscią do błędu zaokrągleń), a tylko kilka mnożników przyjmuje wartości zdecydowanie dodatnie. Są to mnożniki odpowiadające wektorom nośnym. Oznaczmy zbiór tych wektorów (punktów) symbolem SV. Dla punktów tych mamy

$$g(\mathbf{x}_i) = (\mathbf{w} \cdot \mathbf{x}_i) + b = y_i, \quad \forall i \in SV.$$

Warunek powyższy ma zachodzić dla każdego wektora $\mathbf{x}_i$ stanowiącego wektor nośny i może służyć – przy znanym $\mathbf{w}$ – do wyznaczenia stałej b. W zaistniałej sytuacji stała b nie jest wyznaczona jednoznacznie. Jedną z możliwości, pokazaną w raporcie S. Gunn'a [3]

jest wybranie dwóch punktów $\mathbf{x}_r, \mathbf{x}_s$ należących do przeciwnych klas i obliczenie dla nich średniego wyniku:

$$b = -\frac{1}{2}\mathbf{w} \cdot [\mathbf{x}_r, \mathbf{x}_s]. \tag{11.12}$$

Liczba pozostawionych wektorów nośnych zależy od tego, jaką ustawimy poprzeczkę dla wartości α_i, tzn. kiedy uznamy wartości α_i za zerowe. Np. S. Gunn w swoim pakiecie [3] (funkcja `svtol`) przyjmuje warunek

$$\alpha_i < 1e^{-6}.$$

Tak więc szukana wartość funkcji klasyfikacyjnej wyraża się ostatecznie wzorem:

$$\boxed{g(\mathbf{x}) = sgn\left(\textstyle\sum_{i \in SV} \alpha_i y_i \cdot (\mathbf{x}_i \cdot \mathbf{x}) + b\right),} \tag{11.13}$$

i zależy tylko od kilku wektorów nośnych. Zauważmy jednocześnie, że obliczana wartość funkcji klasyfikacyjnej $g(\mathbf{x})$ zależy faktycznie tylko od iloczynów skalarnego wektora $\mathbf{x}$ z kilkoma wektorami nośnymi o niezerowych współczynnikach α.

Zauważmy dodatkowo, że gdybyśmy chcieli zaklasyfikować jakiś nowy element $\mathbf{x}$, to **wartość funkcji decyzyjnej $g(\mathbf{x})$ zależy tylko od iloczynu skalarnego nowego** (klasyfikowanego) **elementu x z wektorami nośnymi próbki uczącej** użytej do wyznaczenia funkcji decyzyjnej.

11.4 Rozszerzenie koncepcji SVM na dane liniowo nieseparabilne

Jak uogólnić wynik uzyskany w poprzednim punkcie na przypadek gdy zbiory danych (próbki uczącej) wyznaczających obydwie klasy **nie są** liniowo separabilne?

Cortes i Vapnik wykazali [4], że można to zrobić podobnie jak w przypadku liniowo separabilnych wektorów danych, ale kosztem zrezygnownia z własności (11.6) dla niektórych elementów próbki uczącej. W szczególności, jest to możliwe, gdy niektóre elementy próbki uczącej zamiast warunku $y_i\,(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1)$ spełniają tylko następujący warunek

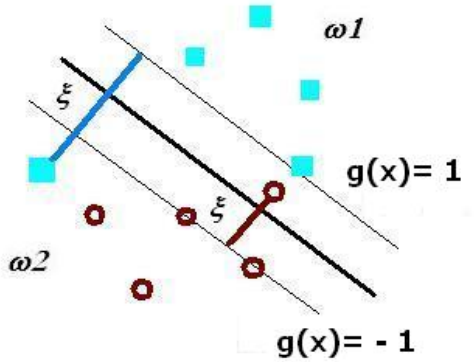
$$y_i\, g(\mathbf{x}_i) \geq 1 - \xi_i, \text{ with } \xi_i \geq 0, \ i = 1, \ldots, l. \tag{11.14}$$

Nowo wprowadzone zmienne ξ_i są nazywane *zmiennymi zwisającymi* (slack variables). Jednak wartości tych zmiennych nie powinny być zbyt duże, i wobec tego wprowadzamy dodatkowy postulat, żeby suma zmiennych ξ_i była 'mała': Jak mała? Reguluje to stała C i w sumie należy rozpatrzyć wyrażenie

$$C \sum_i \xi_i$$

Stała C gra w równaniu optymalizacyjnym rolę parametru regularyzującego (regularization parameter). Stała C jest deklarowana przez użytkownika; w skrajnych przypadkach może ona być równa nieskończoności ($C = inf$).

"The sum $\sum_i \xi_i$ expresses our tolerance in the values of the classifier $g(\mathbf{x})$ for misplaced data vectors (i.e. those that are on the wrong side of the separating hyperplane)."



Rysunek 11.2: SVM dla dwóch klas (grup) liniowo nieseparabilnych. Widoczne hiperpłaszczyzny marginesowe o równaniach $g(x) = +1$ i $g(x) = -1$ wspierające się na jednym i dwóch punktach. Jednak funkcja $g(x)$ klasyfikuje błędnie dwa punkty; zwisają one z marginesów swojej grupy w kierunku grupy przeciwnej. Wielkość 'zwisu' tych punków oznaczono na rysunku symbolem ξ.

Przykład dla danych nie posiadających własności 'liniowa separabilność' jest pokazany na rysunku 11.2. Są tam pokazane dwa punkty, które wykraczają poza hiperpłaszczyzny marginalne swojej grupy i 'zwisają' z tej hiperpłaszczyzny w kierunku grupy przeciwnej.

Formułowane kryterium optymalizacyjne zawiera teraz znacznie więcej warunków. Znajdywanie szukanego rozwiązania przebiega podobnie, jak w przypadku danych separabilnych. Generalnie, nasz postulat jest następujący:

$$minimizuj \quad \tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i$$

przy warunkach ubocznych

$$y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \ \geq \ 1 - \xi_i, \forall\, \mathbf{x}_i \in \mathcal{L}, \ \text{oraz} \ \ \xi_i \geq 0 \ \forall_i \,.$$

Odpowiedni Lagrangian (*Lagrangian2*) jest następujący (por. [3], wzór (31)):

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i \ - \sum_{i=1}^{l} \alpha_i \left\{ [((\mathbf{x}_i \cdot \mathbf{w}) + b)\, y_i] \ - \ 1 + \xi_i \right\} - \sum_{i=1}^{l} \beta_i \xi_i. \tag{11.15}$$

Zmienne $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_l)^T$ stanowią tzw. mnożniki Lagrange'a wynikające z ograniczeń na wartości zmiennych zwisających ξ. Mamy: $\beta_i \geq 0, \ \forall_i$.

Obliczmy pochodne cząstkowe względem parametrów $\mathbf{w}, b$ i wielkości $\boldsymbol{\xi}$. Obliczone pochodne muszą się zerować parametrów (zmiennych) przy osiągnięciu minimum minimizowanego kryterium. Wstawiając otrzymane stąd wyrażenia spowrotem do Lagrangianu (11.15), otrzymamy tzw. problem dualny, w którym niewiadomymi są już tylko wektory $\boldsymbol{\alpha}$ i $\boldsymbol{\beta}$. Otrzymujemy również z warunku zerowania się pochodnej Lagrangianu2 względem ξ, że

$$\alpha_i + \beta_i = C, \ \forall_i.$$

W tej sytuacji możemy wyeliminować $\boldsymbol{\beta}$ (tzn. wyrazić je przez $\boldsymbol{\alpha}$) i pozostaje nam do rozwiązania następujący `problem dualny`:

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \left(\boldsymbol{\alpha} \cdot \mathbf{1} \ - \ \frac{1}{2} \boldsymbol{\alpha} \cdot \mathbf{D} \boldsymbol{\alpha} \right), \quad \text{gdzie} \ \ D_{ij} = y_i y_j \cdot (\mathbf{x}_i \cdot \mathbf{x}_i) \tag{11.16}$$

z restrykcjami: $\boldsymbol{\alpha} \cdot \mathbf{y} = 0$, oraz $0 < \alpha_i \leq C$.

Rozwiązanie tego problemu jest podobne jak w przypadku danych separabilnych, jednak mnożniki Lagrage'a zyskały teraz dodatkowe ograniczenie w formie nierówności

$$0 \leq \alpha_i \leq C$$

Jakie C ma być używane? Blanz i Vapnik (patrz dokumentacja S. Gunn'a [3]) zalecali przyjmowanie wartości $C = 5$. Jednak .. "ultimately C must be chosen to reflect the knowledge of the noise on the data".

Korzystanie z przekształceń kernelowych

Stwierdziliśmy, że funkcja klasyfikująca $g(\mathbf{x})$ wymaga jedynie wyznaczenia iloczynów skalarnych $(\mathbf{x}_j \cdot \mathbf{x}) = \mathbf{x}_j^T \mathbf{x}$, $j \in \mathcal{SV}$ natomiast nigdy nie są potrzebne bezpośrednio wartości przekształconych danych $\varphi(\mathbf{x})$.

Korzystamy z wybranych przekształceń przeprowadzajćych obserwowane dane do rozszerzonej przestrzeni $\mathcal{F}$ 'feature space', i takich że iloczyn skalarny przekształconych zmiennych daje się wyrazi jako prosta funkcja iloczynu skalarnego wektorów danych z obserwowanej przestrzeni R^d.

Dane, które w przestrzeni oryginalnych obserwacji R^d były liniowo nieseparabilne, mogą stać się separabilnymi w przestrzeni $\mathcal{F}$. Wtedy dane klasyfikujemy w przestrzeni $\mathcal{F}$.

11.5 Software do obliczania sieci SVM - S. Gunn'a

Steve Gunn [3] zaprogramował pakiet The Matlab SVM Toolbox przeznaczony do obliczania zagadnień regresyjnych i dyskryminacyjnch dla dwóch grup danych. Poniżej omawiamy tylko **zagadnienia związane z klasyfikacją** i wyznaczaniem granicy decyzyjnej między dwoma grupami danych. Dokumentacja pakietu (52 strony) znajduje się w Raporcie (plik pdf) dołączonym do pakietu [3].
Fragmenty obliczeń zostały zaprogramowane w postaci skryptów postaci M-files (rozszerzenie `.m`). czyli w postaci plików tekstowych wykonywalnych bezpośrednio pod Matlabem. Wyjątek stanowi funkcja do optymalizacji kryterium do wyznaczania granicy decyzyjnej (metoda 'quadratic programming'). Funkcja ta, o nazwie `qp`, została skompilowana z tekstu źródłowego w języku `C` i jest dostarczona jako plik binarny o nazwie `qp.dll`. Oprogramowanie działa również pod Matlabem 7. Toolbox ten znajduje się chwilowo w IIn w folderze 'aba' na Posejdonie, podkartoteka 'svm'.

Podstawowe funkcje do klasyfikacji

```
[nsv, alpha, b0] = svc(X,Y,ker,C);
```

Dane:
Tablica zmiennych objaśniających $X_{n\times d}$,
$Y_{n\times 1}$, wartości docelowe, target, przyjmuje tylko wartości $+1$ i -1,
typ kernela, może być:

'linear' - przekształcenie tożsamościowe
'poly' - wielomianowe, z parametrem p1 równym stopniowi wielomianu
'rbf' - za pomocą radialnej funkcji bazowej z parametrem sigma
'sigmoid' - sigmoidale, p1 is scale, p2 is offset
'spline' - patrz [3]
'bspline' - p1 is degree of bspline
'fourier' - patrz [3]
'erfb' - p1 is width of gaussian rbfs (sigma)
'anova' - p1 is max order of terms

Przekształcające funkcje jądrowe mogą korzystać z jednego, dwóch lub żadnego parametru. Pierwszy parametr nazywa się $p1$, a drugi $p2$. Parametry te są deklarowane jako zmienne globalne. Jeśli obliczenia mają dostarczyć liniowej funkcji dyskryminacyjnej w przestrzeni R^d, to należy zadeklarować *linear* lub *poly* z parametrem $p1 = 1$.

Funkcja `svc` wywołuje następujące funkcje:
`epsilon=svtol(C);` określa, kiedy mnożniki *alpha* można uważać za zerowe.
`H(i,j)=Y(i)*Y(j)*svkernel(ker,X(i,:),X(j,:);` oblicza kernele.
`neqcstr=nobias(ker);` określa jak ma być liczony `bias`.
`[alpha lambda how] = qp(H, c, A, b, vlb, vub, x0, neqcstr);` rozwiązuje zagadnienia sformułowane jako 'programowanie kwadratowe'.

Inne funkcje:
`[X, A, B] = svdatanorm(X,ker,isotropic)` - normalizacja danych, zależy od wybranego kernela. Sam fakt normalizacji danych jest sprawą indywidualnej decyzji użytkownika.
`function y = softmargin(x)`

Przedstawianie wyników:
`function err = svcerror(trnX,trnY,tstX,tstY,ker,alpha,bias)`
`function svcinfo(trn,tst,ker,alpha,bias)`
`function predictedY = svcoutput(trnX,trnY,tstX,ker,alpha,bias,actfunc)`

Przykładowe obliczenia:

Podstawą obliczeń są dane zapamiętane w pamięci peracyjnej jako tablica `X(1:n,1:d)` oraz informacje o przynależności każdego wiersza danych do pierwszej lub drugiej grupy – informacje te powinny być przedstawione w postaci wektora `Y(1:n,1)` o wartościach $+1$ dla klasy pierwszej i wartości -1 dla klasy drugiej.

Przykładowy skrypt– macro obliczeń jest pokazany pon iżej.

```
%% jsvm.m, makro obliczające dyskryminację metodą SVM
load iris1v23
global P1;
P1=2;
ker='poly';
C=inf;
[nsv, alpha, b0] = svc(X,Y,ker,C);
```

```
   iisv=find(alpha¿0.001); size(iisv)  % 16
iisv0=(alpha¿0.001); size(iisv0)

   ii1=find(Y==1); n1=size(ii1) % 83 klasa Pierwsza, target class
ii0=find(Y==-1); n0=size(ii0) % 37 klasa Zerowa, pozostali

   figure(2),
plot(X(ii1,1),X(ii1,2), 'ro', X(ii0,1),X(ii0,2), 'gs')
title('iris3v12')
hold on
plot(X(iisv0,1),X(iisv0,2),'b.', 'markersize', 6)
hold off
```

Graficzny interfejs dla danych o 2 zmiennych

Jest to moduł o nazwie uiclass. Pełni on rolę pewnego rodzaju modułu demonstracyjnego działania pakietu.

Dane dla dwóch grup można wczytać

- manualnie z monitora korzystając z przycisku a (dane grupy pierwszej) lub b (dane grupy drugiej); korzystamy wtedy z Matlabowskiej funkcji `ginput(xi,yi,button)`; lewy przycisk myszy zapamiętuje nowy punkt, prawy przycisk kończy wprowadzanie danej grupy ('a' lub 'b') danych;

- alternatywnie można wczytać dane do obliczeń za pomocą przycisku load. W takim przypadku odpowiedni plik powinien zostać przygotowany wcześniej i umieszczony w odpowiedniej kartotece użytkownika. Plik ten powinien być w formacie mat i zawierać następujące zmienne:

  `X[1:n,1:2]` - analizowane dane, zawierają zarówno elementy z klasy ω_1 jak i ω_2.

  `Y[1:n]` - target, zawiera elementy o wartościach $+1$ i -1 pokazujące, do której klasy należy kolejny, i-ty wiersz tablicy `X[i,1:2], i=1, ... , n`.

  `ker` - zmienna tekstowa, nazwa kernela który ma być zastosowany, np. ker = poly,

  `sep` - zmienna wskazująca, czy chcemy prowadzić obliczenia przy założeniu modelu separabilnego (`sep=1`), czy też nie (`sep=0`).

  `C` - jakaś wartość dla stałej C potrzebnej przy ograniczaniu wpływu zmiennych zwisaących (slack variables), jeśli rozpatruje się model nieseparabilny (stała `sep=0`). Zmienna ta powinna mieć zadaną jakąś wartość początkową, może to być wartość `C=inf`.

  Wszystkie wymienione zmienne należy we wcześniejszej sesji zdefiniować, (tak żeby znalazły się w pamięci operacyjnej), a następnie zapamiętać w postaci pliku o rozszerzeniu `.mat`, np.
  `save iris1v2.mat X Y ker sep C -mat`
  Wtedy plik `iris1v2.mat` zostanie zapamiętany w bieżącym folderze. Na następnych sesjach będzie go można wczytać do pamięci operacyjnej Matlaba za pomocą przycisku load (Matlab po zaktywizowaniu tego przycisku zapyta się, jaki plik o rozszerzeniu .mat ma załadować).

Aby skorzystać z modułu graficznego, należy najpierw wczytać funkcję uiclass() za pomocą instrukcji:
`load uiclass` – wczytuje dane potrzebne do skonstruowania obiektu graficznego - dane te zostają zapamiętane w pamięci operacyjnej ('workspace').

Aby uruchomoć moduł graficzny należy zastartować go pisząc:
`uiclass` – wyświetla okno graficzne na ekranie z odpowiednim menu. Użytkownik - korzystając z odpowiedniego przycisku ('button') wybiera akcję którą chce wykonać.

Najpierw należy określić dane do obliczeń (przyciski load lub a i b).

Gdy mamy już określone dane, wybieramy z menu okna odpowiedni kernel i jego parametry, określamy również parametr sep i C. Mając określone parametry zadania, wciskamy przycisk *classify*. Podczas wykonywania akcji `classify` program najpierw normuje dane (funkcja `svdatanorm(X,ker)`) a następnie wyznacza `[nsv, alpha, bias]` czyli liczbę wektorów nośnych, wektor mnożników Lagrange'a $\boldsymbol{\alpha}$ i bias, czyli stałą `b` za pomocą funkcji `svc(Xnorm,Y,ker,C)`. Wykres klasyfikacyjny jest robiony za pomocą funkcji `svcplot(Xnorm,Y,ker,alpha,bias)`. Otrzymujemy wtedy (liniową lub nieliniową) funkcję decyzyjną (odpowiadającą wartościom $g(x) = 0$), wektory podpierające ze zbioru SV, oraz i odpowiednie marginesy odpowiadające wartościom $g(x) = +1$ i $g(x) = -1$.

Generalnie dążymy do wyboru najprostszego modelu, tj. takiego który wykonuje klasyfikację przy najmniejszej liczbie wektorów podpierających.

11.6 Kernel GDA - Generalized Discriminant Analysis

O stosowaniu różnych nieliniowych funkcji dyskryminacyjnych, pisali [HTB97, DHS01, MikaAL99, RothSt99], w tym również również przy użyciu transformacji kernelowych [MullerAL01, SchollAL99].

Baudat i Anouar [9, 10, 11] zaproponowali bardzo prosty i efektywny model uogólnionej klasycznej Fisherowskiej funkcji dyskryminacyjnej (FLDA) konstruującej kanoniczne zmienne dyskryminacyjne wynikające z kryterium Fishera opartym na maksymizacji zmienności międzygrupowej rozpatrywanej na tle zmienności całkowitej. Podstawowymi pojęciami są tu: macierz zmienności międzygrupowej $\mathbf{B}$, zmienności wewnątrzgrupowej $\mathbf{W}$ oraz całkowitej $\mathbf{T}$. Mamy: $\mathbf{T} = \mathbf{B} + \mathbf{W}$. Metoda działa dla dowolnej liczby klas.

Autorzy pokazali, jak można uzyskać takie zmienne w przestrzeni $\mathcal{F}$ zmiennych przekształconych posługując się tylko kernelami odpowiadającymi tym przekształceniom - bez wykonywania samych przekształceń, tzn. korzystając z własności nazywanej trikiem kernelowym ('kernel trick') opisanym w podsekcji 11.1 tego rozdziału. Metody i przykłady ich stosowania są przedstawione w [9, 10, 11, 12, 13].

Zadanie rozpoczyna się od poszukiwania wektora $\mathbf{v} \in \mathcal{F}$ maksymizującego następujący wskaźnik przedstawiający stosunek zmienności międzygrupowej do zmienności całkowitej (the ratio of the between class to the total inertia)

$$\eta = \frac{\mathbf{v}'\mathbf{B}\mathbf{v}}{\mathbf{v}'\mathbf{T}\mathbf{v}} \tag{11.17}$$

Wskaźnik η przyjmuje wartości z przedziału $0 \leq \eta \leq 1$. Duże (bliskie jedności) wartości wskaźnika η wskazują na dużą separację klas.

Szukane kryterium η jest zdefiniowane w terminach macierzy $\mathbf{B}$ i $\mathbf{T}$ określonych w przestrzeni zmiennych przekształconych $\mathcal{F}$.

Baudat i Anouar zreformułowali kryterium η w ten sposób, aby było ono sformułowane w terminach macierzy kernelowej $K(\mathbf{x}, \mathbf{y})$ która – dzięki trikowi kernelowemu – może być obliczona bezpośrednio z danych obserwowanych w R^d, bez konieczności wykonywania przekształceń $\phi(\mathbf{x})$. Zaproponowane (przez Baudat i Anouar) metody zostały zaprogramowane w postaci kilku funkcji Matlabowskich. Są nimi [1]:

dataGDA=BuildGDA1(Tr,[n1, n2, ... nk], ker);

Dane wejściowe:
Tablica danych Tr o rozmiarze $n \times d$, dane kolejnych grup są podpisane jedna pod drugą; tablica liczebności $[n_1, \ldots, n_K]$ grup zawartych w tablicy Tr,
ker - nazwa kernela, możliwe: 'poly' ze stopniem $P1$ i 'RBF' z parametrem P1=Sigma (parametr p1 jest zmienną globalną).

Funkcja BuildGDA1 oblicza macierz kernelową $\mathbf{K} = (K_{i,j})$, centruje ją, a następnie oblicza jej wektory i wartości własne, służące do konstrukcji wektorów rzutujących. Wyniki obliczeń zostają zapamiętane w strukturze o formacie struct. Zostają wydrukowane informacje, ile obliczono wektorów rzutujących i jaka będzie inercja otrzymanych rzutów.

Do wykreślania zmiennych kanonicznych służy funkcja

plotGDA1(X,Tr, dataGDA, [1,2], 'ro', mz);

Dane wejściowe:
X - dane które chcemy zrzutować, mogą nimi być zarówno dane Tr służące do uczenia sieci, jak i nowe dane – pod warunkiem, że zostały wystandaryzowane tak samo, jak dane uczące Tr,
Tr - dane które posłużyły do utworzenia struktury dataGDA,
dataGDA - wyniki procedury buildGDA, otrzymane dla danych uczących Tr,
$[1, 2]$ - para zmiennych kanonicznych która ma być wykreślona, jeśli jest tylko jedna zmienna kanoniczna, to należy podać parę $[1, 1]$,
'*ro*' - kolor i marker, jakim mają być oznaczana rzutowane punkty,
mz - markersize, wielkość markera, liczona w punktach, np. 12 lub 14.

Dodatkowe funkcje:

`function [eVec,eVal]=EigenSystem(m);` pomocnicza funkcja do obliczenie wektorów i wartości własnych macierzy kernelowej,

`function r=KernelFunction1(ker,x,y);` obliczanie elementów $K(i, j)$ macierzy kernelowej,

`function projectedVectors=SpreadGDA1(T,L,dataGDA);` oblicza wektory rzutów z danych zapamiętanych w strukturze dataGDA

`function XS=stan_dan(X)` standaryzacja statystyczna tablicy $X_{n\times d}$

Przykład zastosowań do danych irisa, rzutowanie 3 odmian irisa. Sieć została wytrenowana próbką uczącą. Rzutujemy zarówno próbkę uczącą, jak i próbkę testową, każda z nich innym markerem. Każda odmiana irysa jest oznaczona innym kolorem.

[1] oryginalne procedury zostały nieznacznie zmodyfikowane przez autorkę tych notatek; aby zaznaczyć modyfikacje, nowe procedury otrzymały przyrostek '1'

```
% ————— makro jGDA, obliczania GDA na danych 'iris' ————————

    iris_ , irisS=stan_dan(iris);

    % training sample and model parameters
irisTr=[irisS(1:25,:); irisS(51:75,:);irisS(101:125,:);] ;
global P1
P1=2;
ker='poly'; % ker='rbf';

irisGDA=BuildGDA1(irisTr,[25,25,25], ker);
% matrix kM m= 75
% Estimated rank of K: 71
% Inertia: 1.000000
% Inertia: 0.999932
mzL=8; mzT=12; % mz - markerSize Learning, Testing
% rzutujemy próbkę uczącą
plotGDA1(irisTr(1:25,:),irisTr, irisGDA, [1,2], 'ro', mzL);
hold on
plotGDA1(irisTr(26:50,:),irisTr, irisGDA, [1,2], 'go', mzL)
plotGDA1(irisTr(51:75,:),irisTr, irisGDA, [1,2], 'bo', mzL)

    % rzutujemy próbkę testową
plotGDA1(irisS(26:50,:),irisTr, irisGDA, [1,2], 'r+',mzT)
plotGDA1(irisS(76:100,:),irisTr, irisGDA, [1,2], 'g+',mzT)
plotGDA1(irisS(125:150,:),irisTr, irisGDA, [1,2], 'b+',mzT)
title(['iris, ', ker, ' p1=', num2str(P1)])
hold off
% ——— koniec makra ————-
```

Literatura

[1] Francesco Camastra: Kernel Methods for Computer Vision, Theory and Applications. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.9315

[2] Francesco Camastra: Kernel Methods for unsupervided Learning. PhD thesis. DISI-TH, Universita di Genova, 2004.

[3] Steve Gunn, Support Vector Machines for Classification and Regression. ISIS Technical Report. 14 May 1998. Image, Speech and Intelligent Systems Group, Univ. of Southampton.

[4] Corinna Cortes, Vladimir Vapnik, Support-vector networks.
Machine Learning (Kluwer) 20: 1–25, 1995.

[5] V. Vapnik, The Nature of Statistical Learning. Springer 1995.

[6] Alessandro Verri, Support Vector Machines for classification. Course 9.520, Class 14, slides 1–32.

[7] Massimiliano Pontil and Alessandro Verri, Support Vector Machines for 3-D Object Reconition. IEEE Trans. on Pattern Analysis and Machine Intelligence, V. 20, Nb. 6, 637–646, 1998. url citeseer.ist.psu.edu/pontil98support.html

[8] Lampert Christoph and Blaschko Matthew, Kernel Methods in Computer Vision. Slides 1-116. http://www.christoph-lampert.de Foundations and trends in Computer Graphics and Vision.

[9] Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural Computation **12**, 2385–2404 (2000).

[10] Baudat, G., Anouar, F.: Kernel-based methods and function approximation, Int. Joint Conference on Neural Networks, pp. 1244–1249, Washington DC, July 15–19, 2001.

[11] Baudat, G., Anouar, F.: Feature vector selection and projection using kernels. Neurocomputing **55**, 21–38 (2003).

[12] Bartkowiak, A., Evelpidou, N.: Visualizing Some Multi-Class Erosion Data Using Kernel Methods. A. Rizzi i M. Vichi (Red), Proceedings in Computational Statistics, 17th Symposium Held in Rome, Contributed paper, pp. 805–812. 2006, Physica Verag, A Springer Company.

[13] Bartkowiak, A., Evelpidou, N.: Visualizing of some multi-class erosion data using GDA and supervised SOM. K. Saeed et al. (Eds), Biometrics, Compuer Security Systems and Artificial Intelligence Applications. Springer 2006, pp. 13–22.

[DHS01] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd Edition. Wiley (2001)

[Elad] David G. Stork and Elad Yom-Tov, Computer Manual in MATLAB to accompany Pattern Classification. Wiley Interscience, 2004, 136 pages. ISBN: 0-471-42977-5.

[MikaAL99] Mika S., Rätsch G., Weston J., Schőlkopf B., Műller K.R.: Fisher Discriminant Analysis with Kernels. Neural Networks for Signal Processing IX, 41-48. IEEE, 1999.

[MullerAL01] Műller K-R., Mika S., Rätsch G., Tsuda K., Schőlkopf B.: An introduction to kernel-based learning algorithms. IEEE Trans. on Neural Networks, **12** 2001, No. 2, 181–2002.

[RothSt99] Roth V., Steinhage V.: Nonlinear discriminant analysis using kernel functions. NIPS, Advances in Neural Information Processing Systems **12**, Proceedings of the 1999 Conference, MIT press Cambridge, MA, London, England 2000, 568-574.

[SchollAL99] Schőlkopf, B., Mika S., Burges Ch.J.C, Knirsch PH., Műller, K.R., Rätsch G., Smola, A.: Input space versus feature space in kernel-based methods. IEEE Trans. on Neural Networks, **10** 1999, No. 5, 1000–1017.

[ShaweChris04] Shawe-Taylor, J., and Christianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press UK. 2004.

[XuAL06] Xu Y., Zhang D., Jin Z., Li M., Yang Y.-J., A fast kernel-based nonlinear discriminant analysis for multi-class problems, Pattern Recognition 39 (2006) 1026 1033.

[YangAL04] Yang J., Jin Z., Yang J-y., Zhang D., Frangi A. F.: Essence of kernel Fisher discriminant: KPCA plus LDA. Pattern Recognition, **37**, 2097-2100 (2004).

[HTB97] Hastie, T., Tibshirani, R., Buja, A.: Flexible discriminant analysis. JASA **89**, 1255–1270 (1994)

Spis treści

Appendix: Additional denotations and Explanations for GDA

Assume that the data vectors belong to c classes with cardinalities $n_1, \ldots, n_c$. Let M denote the sum $M = \sum_{j=1}^{c} n_j$.

The input data vector $\mathbf{x}$ may be indexed either by a general index i ($i = 1, \ldots, M$) or by a double index kl, $k = 1, \ldots, c$; $l = 1, \ldots, n_k$.

Let $\mathbf{x}_{kl}$ denote the lth data vector in the kth class. Assume further that the data vectors – taken as a whole data set – are centered.

Let $\bar{\mathbf{\Phi}}_k$, $k = 1, \ldots, c$ denote the class centers in $\mathcal{F}$ calculated as $\bar{\mathbf{\Phi}}_k = (1/n_k) \sum_{l=1}^{n_c} \mathbf{\Phi}(\mathbf{x}_{kl})$.

Baudat and Anouar formulate their algorithm in terms of the matrices $\mathbf{T}$ and $\mathbf{B}$, defined as

$$T = \frac{1}{M} \sum_{i=1}^{M} \mathbf{\Phi}(\mathbf{x}_i) \mathbf{\Phi}^T(\mathbf{x}_i), \quad B = \frac{1}{M} \sum_{k=1}^{c} n_k \bar{\mathbf{\Phi}}(\mathbf{x}_k) \bar{\mathbf{\Phi}}^T(\mathbf{x}_k). \tag{11.18}$$

The vector $\mathbf{v}$ appearing in (11.17) is the eigenvector solving the two matrix equation

$$\eta \mathbf{T}\mathbf{v} = \mathbf{B}\mathbf{v}. \tag{11.19}$$