UCZENIE MASZYNOWE III - SVM

mgr inż. Adam Kupryjanow

Plan wykładu

- Wprowadzenie
- LSVM – dane separowalne liniowo
- SVM – dane nieseparowalne liniowo
- Nieliniowy SVM
- „Kernel trick”
- Przykłady zastosowań

Historia

- 1992 wprowadzony przez Boser, Guyon & Vapnik
- Algorytm z mocnymi podstawami teoretycznymi, wywodzący się ze statystyki
- Teoria uczenia (Vapnik & Chervonenkis) z lat 60tych
- Dobrze przebadany eksperymentalnie i zastosowany w wielu dziedzinach: bioinformatyka, rozpoznawanie tekstu/obrazu, …

Dane liniowo separowalne

- Dane $\{\mathbf{x}_i\}$, i = 1, …, l, $\mathbf{x}_i \in R^d$ należące do dwóch klas określonych zmiennymi $\{y_i\}$, i = 1,…,l są liniowo separowalne, jeśli istnieje hiperpłaszczyzna $\mathcal{H}$ postaci $g(x)$:

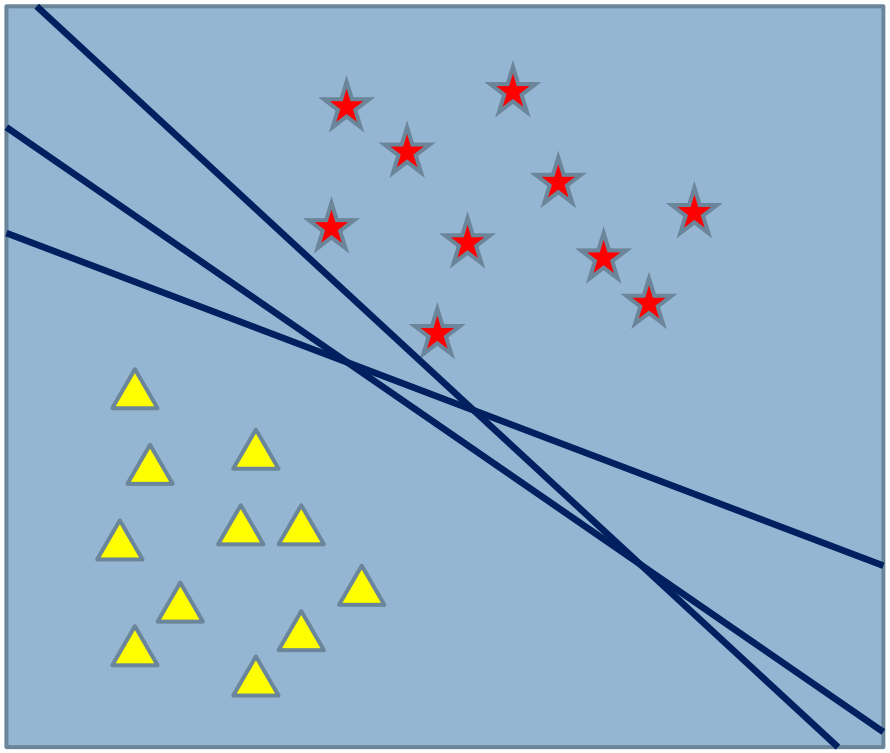
$$\mathcal{H}: g(x) = \mathbf{w}^t\mathbf{x} + b$$

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

$$\begin{cases} g(x_i) > 0 \; dla \; \boldsymbol{x}_i \in w_1 \\ g(x_i) < 0 \; dla \; \boldsymbol{x}_i \in w_2 \end{cases}$$

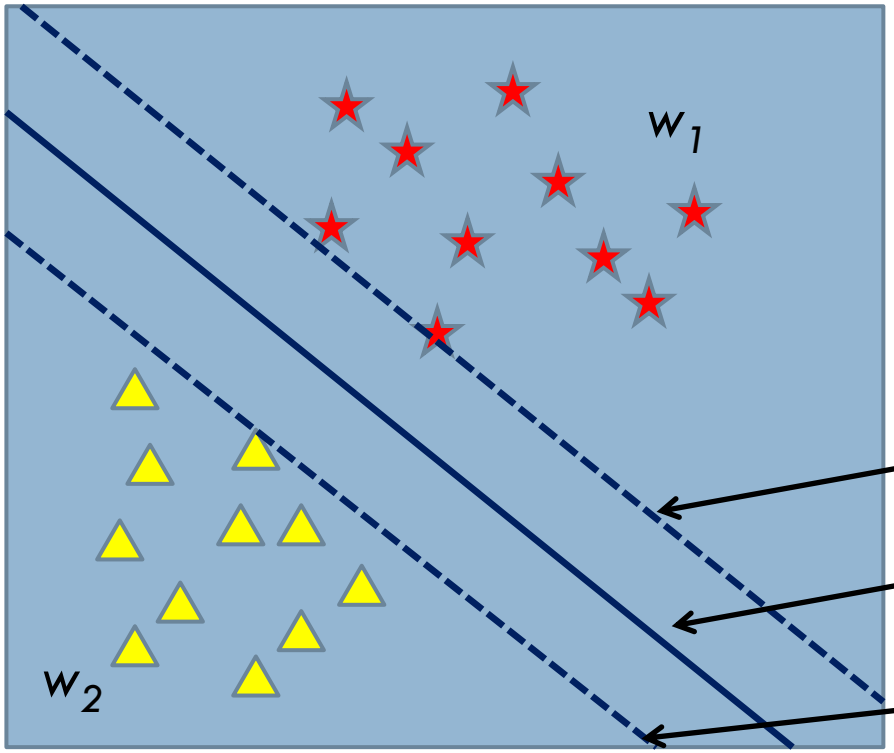
Jak wyznaczyć hiperpłaszczyznę?

Istnieje nieskończenie wiele funkcji rozdzielających dwie klasy



Liniowa maszyna wektorów nośnych LSVM

- Pomysł Vapnika metoda SVM (wektory nośne)
- Algorytm wyznaczanie hiperpłaszczyzny dąży do uzyskania jak największego marginesu



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

$g(x) = 1$ – hiperpłaszczyzna marginesowa

$g(x) = 0$

$g(x) = -1$ - hiperpłaszczyzna marginesowa

Wyznaczanie marginesu

- Można wykazać, że maksymalna odległość pomiędzy marginesami $\mathbf{w}^t\mathbf{x} + b = 1$ i $\mathbf{w}^t\mathbf{x} + b = -1$ wynosi $M = \frac{2}{\|w\|}$, gdzie $\|w\| = \sqrt{w_1^2 + w_2^2 + \cdots + w_i^2}$
- Rozwiązanie powinno dążyć do uzyskania jak najkrótszego wektora **w**, ponieważ wtedy uzyskany margines będzie największy
- Postulaty:
  - minimalizować wektor **w** -> największy margines
  - próbki punktów uczących dla funkcji decyzyjnej mają dawać wartości ≥ 1 lub ≤ -1

LSVM - zagadnienie optymalizacji

- minimalizować po **w** wyrażenie $\frac{1}{2}\|w\|^2$
- przy warunku ograniczającym:

$$y_i \cdot \big((w \cdot x_i) + b\big) \geq 1, \forall_i \epsilon U$$

- Powyższe warunki prowadzą do uogólnionego równania Lagrange'a

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i [y_i\big((x_i \cdot w) + b\big) - 1]$$

gdzie α_i to mnożnik Lagrange'a i $\alpha_i > 0$

LSVM - zagadnienie optymalizacji

- Równanie Lagrange'a powinno różniczkować się po **w** i b
- Porównując pochodne L(w,b,α) względem **w** i b do zera otrzymujemy:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \; oraz \; \sum_{i=1}^{l} \alpha_i y_i = 0$$

- Podstawiając otrzymane wartości do równania otrzymujemy funkcję decyzyjną $g(x)$:

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

LSVM - zagadnienie optymalizacji

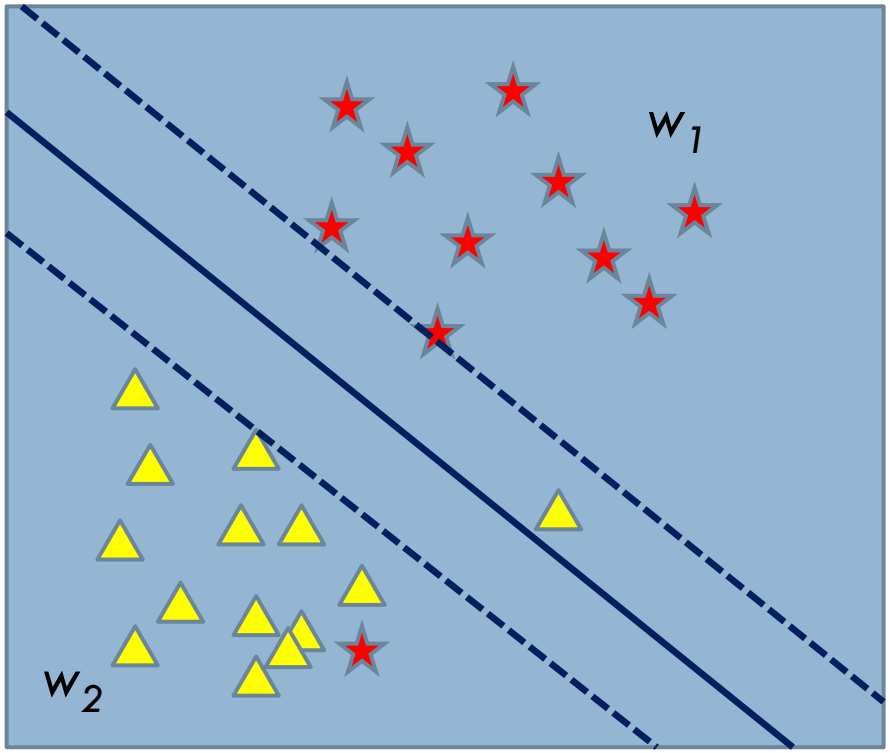
- Funkcja $g(x)$ zależy bezpośrednio od mnożników Lagrange'a (α_i)
- Mnożniki na podstawie twierdzenia KKT (Karush-Kuhn-Tucker) powinny spełniać warunek:

$$\sum_{i=1}^{l} \alpha_i \left[y_i \left((x_i \cdot w) + b \right) - 1 \right] = 0, i = 1, \ldots, l$$

Możliwe gdy:

- x_i leży na marginesie -> α_i dowolne
- x_i leży poza marginesami -> α_i równe zeru

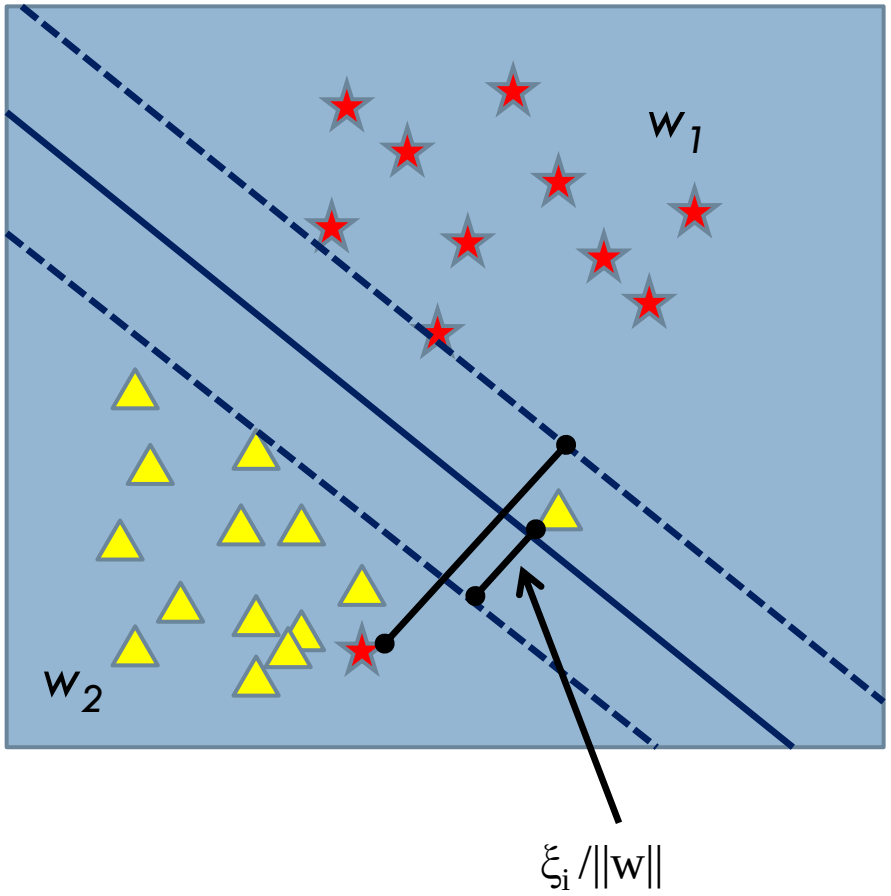
Dane liniowo nieseparowalne



- Cortes i Vapnik wykazali, że możliwe jest zastosowanie wektorów nośnych stosują pewne rozszerzenie założeń

$$y_i \cdot \big((w \cdot x_i) + b\big) \geq 1, \forall_i \epsilon U \longrightarrow y_i \cdot \big((w \cdot x_i) + b\big) \geq 1 - \xi_i, \forall_i \epsilon U$$

Dane liniowo nieseparowalne



- Nowa zmienna ξ_i nazywana „zwisem” (slack variable)
- Wartość ξ_i powinna być „mała”, aby ją określić rozpatrujemy:

$$C \sum_i \xi_i$$

Gdzie C to parametr generalizujący deklarowany przez użytkownika

- Jeżeli $0 \leq \xi_i \leq 1$ to punkt danych leży wewnątrz strefy separujące, po właściwej stronie
- Jeżeli $\xi_i > 1$, punkt po niewłaściwej stronie hiperpłaszczyny =błąd klasyfikacji

Zagadnienie optymalizacji -SVM

- minimalizuj wyrażenie: $\frac{1}{2}\|w\|^2 + C\sum_i^l \xi_i$
- przy warunkach:

$$y_i \cdot \big((w \cdot x_i) + b\big) \geq 1 - \xi_i, \forall_i \epsilon U \qquad \xi_i \geq 0, \forall_i$$

- Otrzymujemy Lagrangian:

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i\{[y_i((x_i \cdot w) + b)] - 1 + \xi_i\} - \sum_{i=1}^{l} \beta_i \xi_i$$

- Wyznaczamy pochodne cząstkowe względem w, b i ξ, i podstawiamy otrzymane wartości do Lagrangianu. Z warunku zerowania pochodnej Lagrangianu względem ξ otrzymujemy:

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

Zagadnienie optymalizacji -SVM

- W tej sytuacji można wyeliminować β i zastąpić przez α
- Do rozwiązania pozostaje problem dualny:

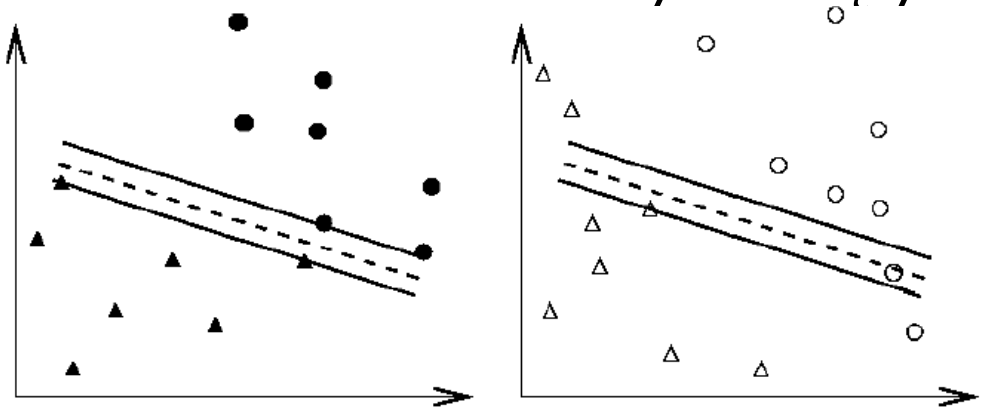
$$\max_{\alpha} W(\alpha) = \max_{\alpha} (\alpha \cdot 1 - \frac{1}{2}\alpha \cdot D\alpha)$$

gdzie, $D_{ij} = y_i y_j \cdot (x_i \cdot x_j)$

z ograniczeniami: $\alpha \cdot \beta = 0, oraz\ 0 \leq \alpha_i \leq C$

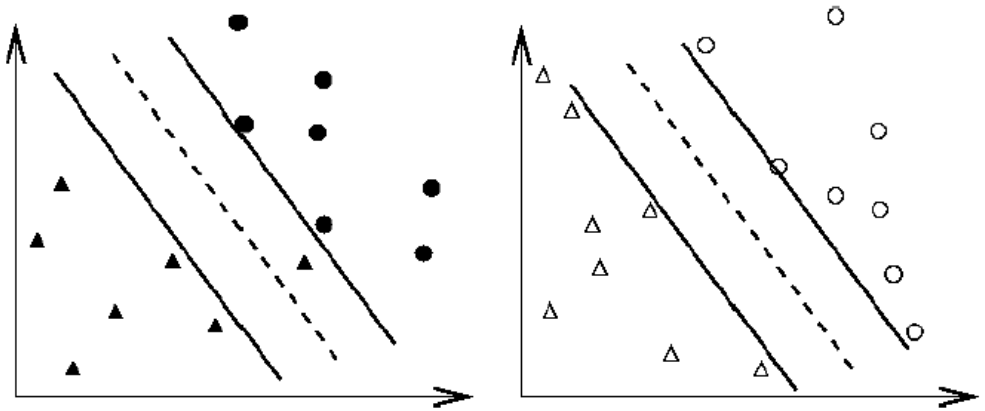
Jakiego C używać?

- Blanz i Vapnik zalecili stosowania C = 5, ale tak naprawdę C należy dobierać w zależności od danych uczących.



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data



(c) Training data and a better classifier

(d) Applying a better classifier on testing data

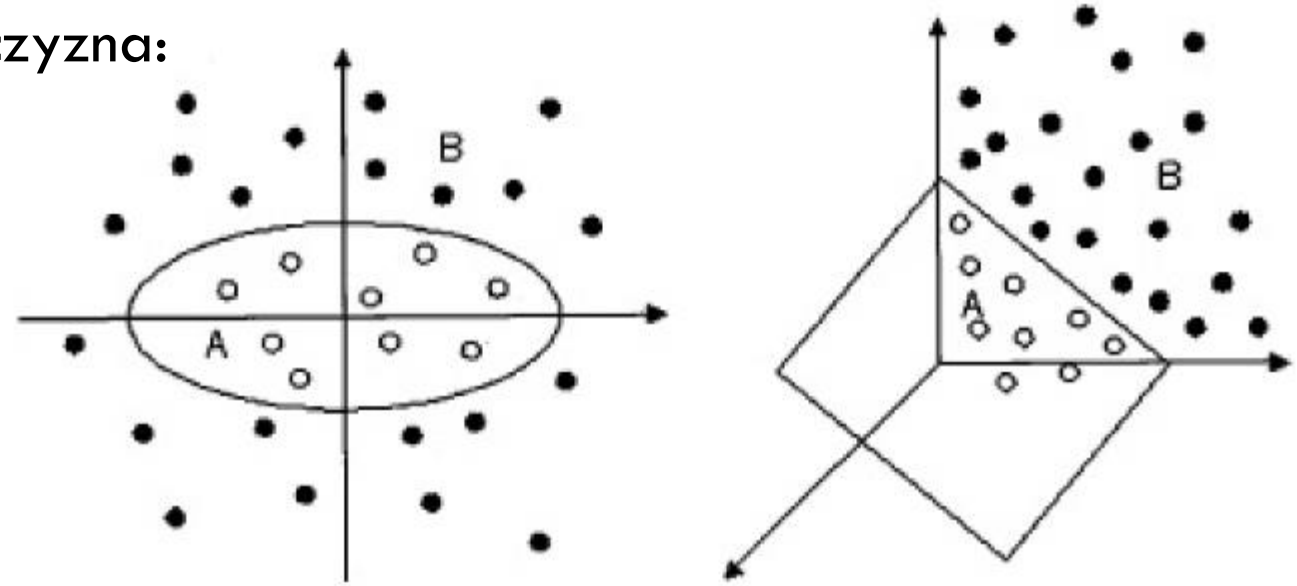
Nieliniowy SVM

- Transformacja do przestrzeni o wyższym wymiarze
- Projekcja danych oryginalnych $x \in R^d$ do przestrzeni wymiarowej $n>d$ w której dane z dużym prawdopodobieństwem będą separowane liniowo

Przykład

- Mamy nieliniową funkcję mapującą φ: $I=R^2 \rightarrow F=R^3$ 2-wymiarową przestrzeń wejściową (input space) do 3-wymiarowej przestrzeni zmiennych przekształconych (feature space)
- $(x_1, x_2) \rightarrow (z_1, z_2, z_3) := (x_1^2, 2^{0.5}\, x_1\, x_2, x_2)$
- hiperłaszczyzna:

$\langle w \cdot x \rangle = 0$



Model nieliniowy SVM

- funkcja decyzyjna $g(\boldsymbol{x}) = \boldsymbol{w}\phi(\boldsymbol{x})+b$
- problem optymalizacji

minimalizuj wyrażenie: $\frac{1}{2}\|w\|^2$

Przy warunkach ograniczających:

$$y_i \cdot \left((w \cdot \Phi(x_i)) + b\right) \geq 1, \forall_i \epsilon U$$

- Funkcja z mnożnikiem Lagrange'a:

$$\max_{\alpha} W(\alpha) = \max_{\alpha} (\alpha \cdot 1 - \frac{1}{2}\alpha \cdot D\alpha)$$

gdzie $D_{ij} = y_i y_j \cdot (\Phi(x_i) \cdot \Phi(x_j))$

- Funkcja klasyfikująca: $sgn\left(\sum_{i=1}^{l} \alpha_i y_i \cdot (\Phi(x_i) \cdot \Phi(x_j)) + b\right)$

Kernel trick

- Jak obliczyć $\Phi(x_i) \cdot \Phi(x_j)$
- $K(x,z) = (x \cdot z)^2$, $x=(x_1, x_2)$, $z= (z_1, z_2)$
- $K(x,z) = (x \cdot z)^2 = (x_1 z_1 + x_2 z_2)^2 = (x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2) =$
  $(x_1^2, \sqrt{2} x_1 x_2, x_2^2) \cdot (z_1^2, \sqrt{2} z_1 z_2, z_2^2) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$
- Więc D_{ij} można zapisać: $D_{ij} = y_i y_j \cdot K(x_i, x_j)$

$\longrightarrow$ nie trzeba znać funkcji $\phi(x)$, do operacji w wyższej przestrzeni wystarczy znajomość jądra (kernel)

Funkcje jądra (kernel functions)

- wielomianowe (polinomial):

$$K(x, z) = (\mathbf{x} \cdot \mathbf{z} + d)^p \text{ ,} p>0$$

- gaussowskie (radial basis function):

$$K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|}{2\sigma^2}}$$

- sigmoidalne:

$$K(\boldsymbol{x}, \boldsymbol{z}) = tgh(\eta \boldsymbol{x} \cdot \boldsymbol{z} - \delta)$$

Kilka uwag praktycznych

- Normalizuj danej wejściowe
- Rozpoczynaj od zastosowania jądra RBF
- Znajdź optymalne wartości C i σ. Jak? np. grid search
- W klasyfikatorze wykorzystaj parametry znalezione w grid-search

Grid-search – znalezienie maksimum

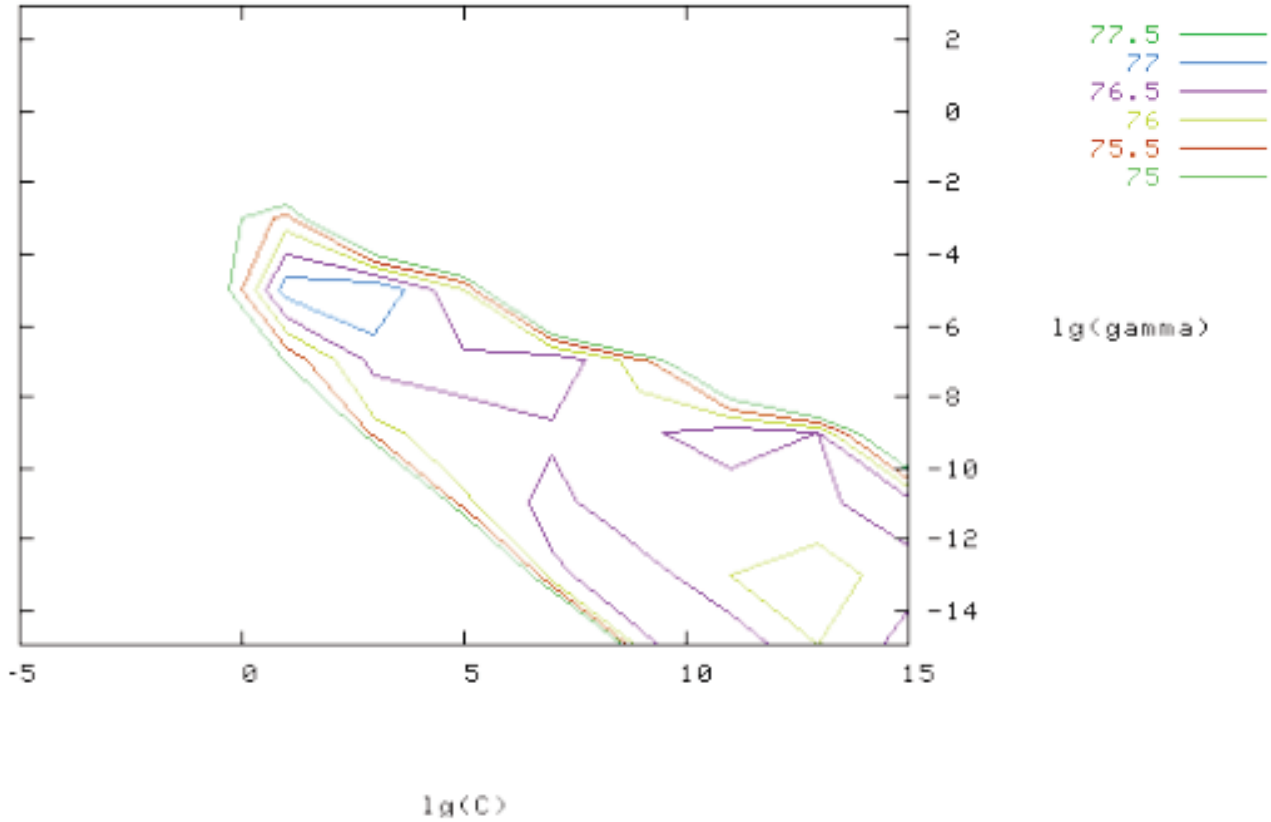


Figure 2: Loose grid search on $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$.

Grid-search – szukanie w okolicy maksimum

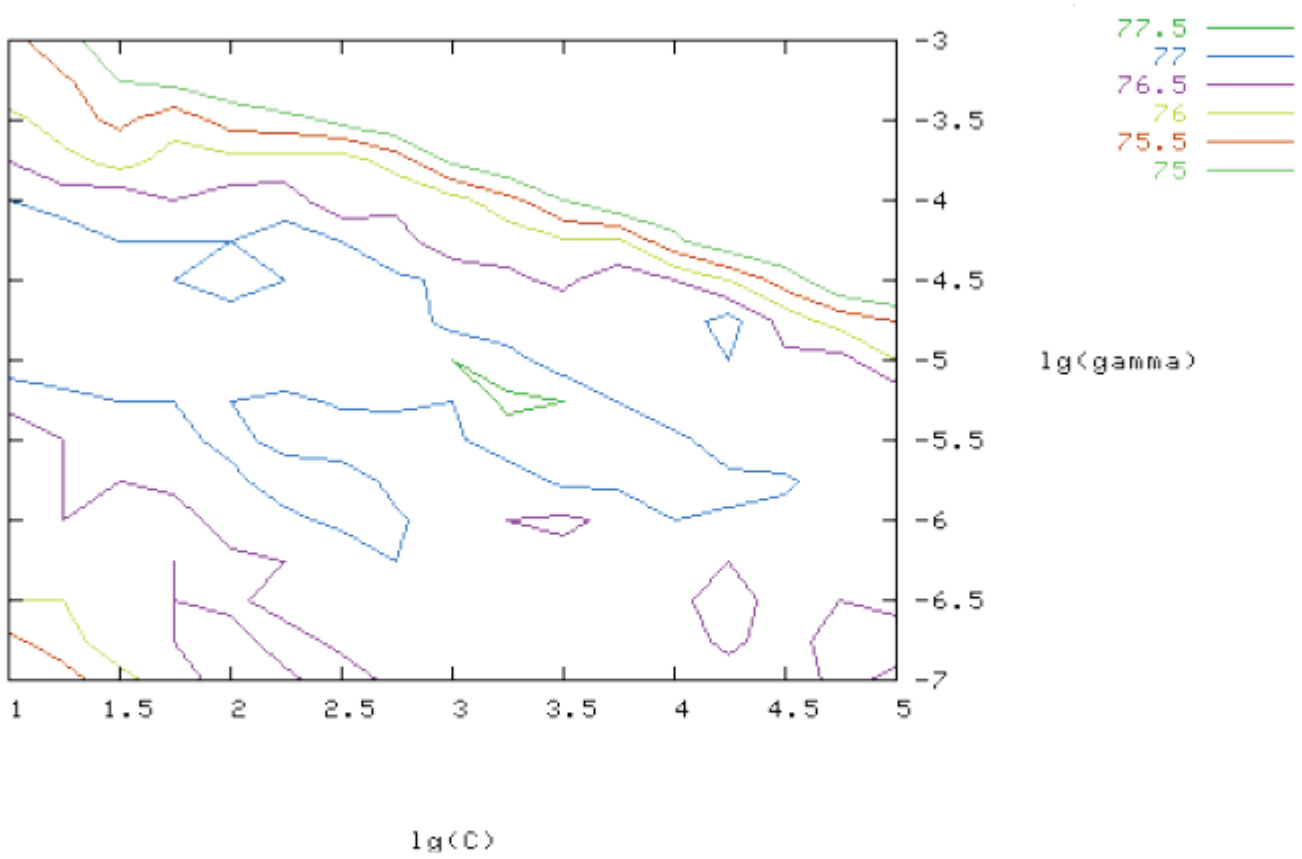


Figure 3: Fine grid-search on $C = 2^1, 2^{1.25}, \ldots, 2^5$ and $\gamma = 2^{-7}, 2^{-6.75}, \ldots, 2^{-3}$.

Klasyfikacja wieloklasowa- One-versus-all

- One-versus-all – wiele klasyfikatorów dwuklasowych. Każdy klasyfikator dzieli dane wejściowe na klasę zainteresowania i na „resztę”.
- Decyzja o przynależności do klasy podejmowana może być w różny sposób np. głosowanie większościowe, pewność decyzji …
- Konieczność wytrenowania tylu klasyfikatorów ile klas.

Klasyfikacja wieloklasowa- One-against-one

- One-against-one – wiele klasyfikatorów dwuklasowych. Klasyfikatory dla każdej pary klas
- Decyzja podejmowana podobnie jak dla One-versus-all
- Konieczność wytrenowania k(k-1)/2 klasyfikatorów, gdzie k to liczba klas

Implementacje

- C++
  - libSVM
  - SVM light
- Java
  - Weka
- Matlab:
  - libSVM
  - Spider

Bibilografia

- A.Bartkowiak: Wykłady nt. Sieci Neuronowych: w11 Kernele, siecie SVM i sieci GDA. http://www.ii.uni.wroc.pl/~aba/
- J. STEFANOWSKI, SVM – Support Vector Machines Metoda wektorów nośnych, http://www.cs.put.poznan.pl/jstefanowski/ml/SVM.pdf
- J. Weston: Support Vector Machine (and Statistical Learning Theory) Tutorial, http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf
- http://www.kernel-machines.org/

Bibliografia

- C. Hsu, C. Chang, C. Lin, “A practical guide to Support Vector Classification”, April 2010, Taiwan. (http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf)
- M. Hoffman, Support Vector Machines — Kernels and the Kernel Trick, http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf

Dziękuję za uwagę !!!