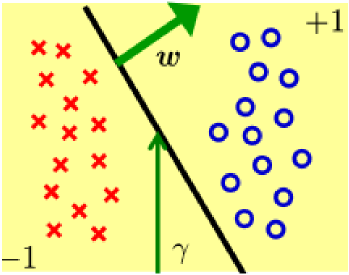
Support Vector Machines

March 28, 2019

Hard margin classification



$$f(x) = w^T x + b \quad (1)$$

- w, b parametri - normala și intercept pentru hiperplanul de separație
- parametri învățați a.î. **marginea** pt. samples este pozitivă:

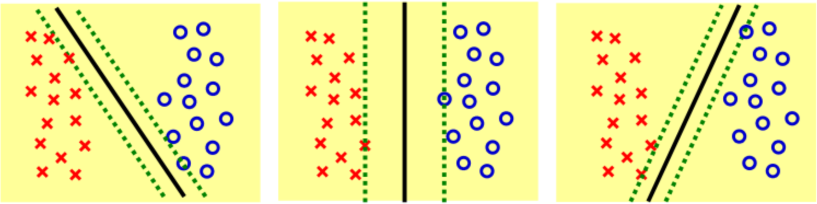
$$f(x_i)y_i = (w^T x_i + b)y_i > 0, \quad i = 1 \dots n \quad (2)$$

$$y_i \in \{-1, +1\}, \; i = 1 \dots n$$

- dificil de rezolvat, constrângerea se rescrie - scalele pentru w și b sunt arbitrare:

$$(w^T x_i + b)y_i \geq 1, \quad i = 1 \ldots n \tag{3}$$

- dacă există $\{w, b\}$ setul $\{(x_i, y_i)\}_{i=1}^n$ este zis **liniar separabil**
- există un număr infinit de hiperplane de separație
- îl selectăm pe cel care are marginea maximă



(a) Small margin (b) Large margin (c) Small margin

Marginea față de hiperplan

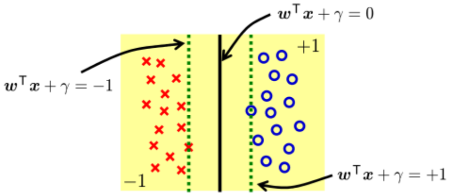
- distanța d de la un punct x la hiperplanul $g(x) = w^T x + b = 0$:

$$x = x_p + d\frac{w}{\|w\|} \tag{4}$$

$$g(x) = g\left(x_p + d\frac{w}{\|w\|}\right) =$$

$$w^T\left(x_p + d\frac{w}{\|w\|}\right) + b = (w^T x_p + b) + \frac{w^T w}{\|w\|}d = \frac{\|w\|^2}{\|w\|}d = d\|w\|$$

$$\Rightarrow d = \frac{g(x)}{\|w\|} \tag{5}$$



- pentru toate punctele x_i, marginile m_i vor fi:

$$m_i = (w^T x_i + b) y_i / \|w\| \quad (6)$$

$$\min_{i=1\ldots n} \frac{(w^T x_i + b) y_i}{\|w\|} = \frac{1}{\|w\|} \quad (7)$$

- geometric, marginea unui set (**x** sau **o**) este jumătate din distanța dintre cele două **hiperplane**
- denumit hard margin SVM:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \; s.t. \, (w^T x_i + b) y_i \geq 1, \; i = 1 \ldots n \quad (8)$$

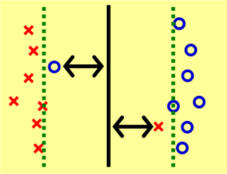
Soft margin SVM

- hard SVM cere linear separability - în practică rar satisfăcut
- soft margin SVM relaxează prin adăugarea erorii ξ_i pentru margini

$$\min_{w,b,\xi} \left[\frac{1}{2} \|w\|^2 + C \sum_{i=1^n} \xi_i \right]$$

$$s.t. \ (w^T x_i + b) y_i \geq 1 - \xi_i, \ \xi_i \geq 0, \ i = 1 \dots n \qquad (9)$$

- C controlează erorile peste margine - C mare determină ξ_i mici (tinde la hard margin SVM)
- ξ_i sunt denumite **slack variables**



Optimizarea duală a SV classification

- ecuația anterioară ia forma unei probleme quadratic programming cu constrângeri liniare
- numărul de parametri $d + n + 1$ nu scalează bine pentru dataseturi mari
- Lagrangianul[1] problemei de optimizare:

$$L(w, b, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i((w^T x_i + b)y_i - 1 + \xi_i) - \sum_{i}\beta_i\xi_i \tag{10}$$

- duality principle: soluția maximizând problema duală este $\leq$ soluția minimizând problema primal

$$\max_{\alpha,\beta}\min_{w,b,\xi} L(w, b, \xi, \alpha, \beta)\; s.t.\; \alpha \geq 0,\; \beta \geq 0 \tag{11}$$

[1] soluția la optimizarea lui f(x, y) cu consrângerea g(x, y) se bazează pe aceeași direcție a gradienților, $\nabla f(x, y) = \lambda \nabla g(x, y)$

- condiția de optimalitate pentru $\min_{w,b,\xi} L(w, b, \xi, \alpha, \beta)$ determină:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{12}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{13}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i + \beta_i = C \tag{14}$$

- w și β pot fi exprimate în funcție de α, iar ξ_i dispar din Lagrangian

- forma Lagrange duală se simplifică:

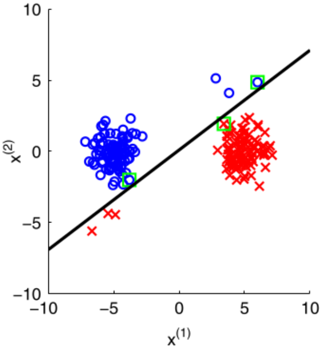
$$\hat{\alpha} = arg\max_{\alpha}\left[\sum_i \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\right]$$

$$s.t. \sum_{i=1}^{n}\alpha_i y_i = 0,\ 0 \le \alpha_i \le C,\ i = 1\dots n \tag{15}$$

- quadratic problem cu doar *n* variabile
- sau rezolvare populară: John Platt, Sequential Minimal Optimization algorithm, construiește soluția pas cu pas (rapidă)
- soluția $\hat{\alpha}$ dă soluția SV classification:

$$\hat{w} = \sum_i \hat{\alpha}_i y_i x_i \quad \hat{b} = y_i - \sum_{j:\hat{\alpha}_j>0}\hat{\alpha}_j y_j x_i^T x_j \tag{16}$$

Sparsitatea soluției duale



- condițiile de optim vin din minimizarea cu constrângeri de tip inegalitate - condițiile Karush-Kuhn-Tucker (KKT)
- variabilele duale α și constrângerile satisfac condițiile **complementary slackness**:

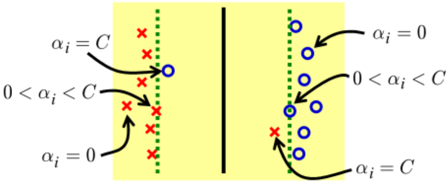
$$\alpha_i(m_i - 1 + \xi_i) = 0 \quad \text{și} \quad \beta_i \xi_i = 0 \quad (17)$$

- unde

$$m_i = (w^T x_i + b) y_i \qquad (18)$$

variabilele duale α_i și marginile m_i satisfac relațiile:

- $\alpha_i = 0$ implică $m_i \geq 1$: sample-ul x_i e pe graniță sau înăuntrul ei, corect clasificat
- $0 < \alpha_i < C$ implică $m_i = 1$: x_i chiar pe graniță, corect clasificat (vector suport)
- $\alpha_i = C$ implică $m_i \leq 1$: x_i pe graniță sau în afara marginii; dacă $\xi_i > 1$, $m_i < 0$ și x_i e clasificat incorect
- $m_i > 1$ implică $\alpha_i = 0$: x_i este în margine, corect clasificat
- $m_i < 1$ implică $\alpha_i = C$: x_i este în afara marginii, incorect



$$\alpha_i((w^T x_i + b)y_i - 1 + \xi_i) = 0 \quad (19)$$

Kernel model

- de regulă în modele precum regresia liniară, sau polinomială, funcțiile bază sunt fixate (polinoame, sinusoide):

$$f_\theta(x) = \sum_{j=1}^{b} \theta_j \phi_j(x) = \theta^T \phi(x) \tag{20}$$

- unde θ_j sunt parametri iar

$$\phi(x) = (1, x, x^2, \dots x^{b-1}) \tag{21}$$

- modelul kernel utilizează sample-urile x_i pentru proiectarea funcțiilor bază

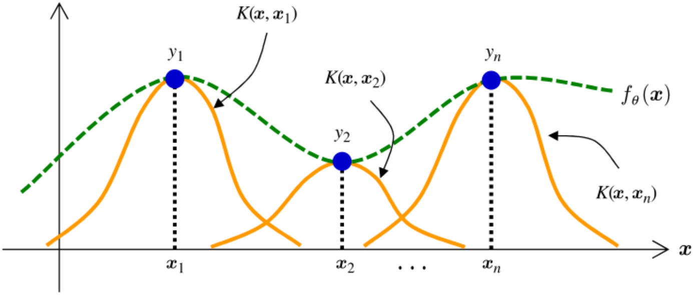
- fie o funcție de două variabile $K(\cdot, \cdot)$
- modelul kernel este definit ca o combinație liniară de funcții $\{K(x, x_j\}_{j=1}^{n})$:

$$f_\theta(x) = \sum_{j=1}^{n} \theta_j K(x, x_j) \tag{22}$$

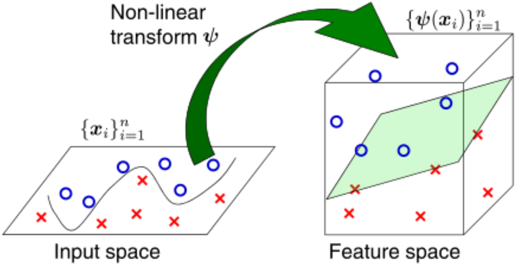
- ca funcție kernel, kernelul Gaussian este cel mai popular:

$$K(x, c) = exp\left(-\frac{\|x - c\|^2}{2h^2}\right) \tag{23}$$

- h - Gaussian bandwidth, c - Gaussian center



- funcțiile Gaussiene localizate la x_i, este învățată înălțimea lor
- modelul aproximează funcția doar în vecinătatea x_i -urilor
- modelul multiplicativ încerca să aproximeze funcția pe tot spațiul
- nr. de parametri dat de n, independent de dimensionalitatea spațiului
- expresia lui x în sine nu contează, el apare doar în funcția kernel $K(x_i, x_j)$ - kernel trick



- sample-urile x_i sunt translatate într-un feature space folosind o funcție nonliniară Ψ
- SVM liniar este antrenat pe spațiul transformat $\{\Psi(x_i)\}_{i=1}^n$
- spațiul Ψ este de multe ori infinit dimensional - mai mari șanse să fie liniar separabile
- kernel trick: în optimizarea liniară SVM apar doar inner-product-urile:

$$x_i^T x_j = < x_i, x_j > \tag{24}$$

- similar:

$$< \Psi(x_i), \Psi(x_j) > \tag{25}$$

- trick-ul este specificarea directă a produsului scalar, chiar dacă nu se cunoaște forma analitică a lui $\Psi(x_i)$:

$$< \Psi(x_i), \Psi(x_j) >= K(x_i, x_j) \tag{26}$$

- kernel trick se aplică și pentru alte probleme: clustering, dimensionality reduction

Bibliografie

- M. Sugiyama, "Introduction to Machine Learning", Elsevier 2016
- https://en.wikipedia.org/wiki/Sequential_minimal_optimization#Algorithm
- https://www.syncfusion.com/ebooks/support_vector_machines_succinctly/solving-the-optimization-problem
- https://www.svm-tutorial.com/2016/09/duality-lagrange-multipliers/
- N. Cristianini, J.S. Taylor, "An Introduction to SVM and Other Kernel-based Learning Methods"
- http://www.csc.kth.se/utbildning/kth/kurser/DD3364/Lectures/KKT.pdf