Învăţare automată

1. **(Exemple de funcţii care măsoară similaritatea dintre două imagini oarecare; demonstrarea unor proprietăţi)**

În învăţarea automată, *funcţiile de similaritate* joacă un rol foarte important atât în clasificarea automată (vedeţi de exemplu algoritmul k-NN) cât şi în clusterizare.

Considerăm X o mulţime formată din imagini dreptunghiulare de dimensiuni arbitrare, în care fiecare pixel este reprezentat sub forma unui număr întreg din mulţimea $\{0, \ldots, 255\}$. Fie $k_1 : X \times X \to \mathbb{R}$ o *funcţie de similaritate* a imaginilor, definită astfel: $k_1(x, x')$ = numărul de zone pătratice (engl., pixel patches), de dimensiune 16×16 pixeli, care apar atât în imaginea x cât şi în imaginea x'.[1]

a. Demonstraţi că funcţia k_1 are proprietatea următoare: există $d \in \mathbb{N}^*$ şi o funcţie („mapare“) $\phi : X \to \mathbb{R}^d$ astfel încât $k_1(x, x') = \phi(x) \cdot \phi(x')$ pentru orice x şi $x' \in X$.[2] **(Am notat cu $\cdot$ produsul scalar al vectorilor.)**

b. Fie acum o altă *funcţie de similaritate*:

$$k_2(x, x') = \begin{cases} 1 & \text{dacă } k_1(x, x') \geq 1, \\ & \\ 0 & \text{în caz contrar.} \end{cases} \quad \begin{array}{l} \text{adică, există cel puţin un “patch”} \\ \text{(zonă pătratică) comun(ă) pentru } x \text{ şi } x' \end{array}$$

Demonstraţi că funcţia k_2 are proprietatea următoare: există o mulţime de imagini $x_1, \ldots, x_n$ şi un vector-coloană $f \in \mathbb{R}^n$ astfel încât $f^\top G f < 0$, unde $\top$ reprezintă operaţia de transpunere, iar G este matricea de tip $n \times n$ având elementele $G(i, j) \stackrel{def.}{=} k_2(x_i, x_j)$, pentru $i, j = 1, \ldots, n$.[3]

***Sugestie*: Puteţi face demonstraţia în manieră constructivă, adică alegând în mod convenabil numărul n şi „construind“ / definind imaginile $x_1, \ldots, x_n$ astfel încât să existe $f \in \mathbb{R}^n$ cu propietatea $f^\top G f < 0$.**

[1] *Precizare*: Nu se va lua în considerare poziţia în care apare o astfel de *zonă pătratică*, ci doar *imaginea* / coloritul ei ca atare. O astfel de *zonă* poate să apară în mai multe poziţii într-o aceeaşi imagine, însă funcţia $k_1(x, x')$ va „contoriza“ o dată o anumită *zonă pătratică* dacă ea apare [nu are importanţă de câte ori anume] atât în x cât şi în x'.

[2] Altfel spus, funcţia k_1 este *funcţie-nucleu*.

[3] Altfel spus, funcţia k_2 *nu* este pozitiv semidefinită, deci *nu* este funcţie-nucleu (conform teoremei lui Mercer).

2. **(Funcţii de cost: o proprietate foarte utilă pentru mai mulţi algoritmi de învăţare automată care produc separatori liniari)**

Fie o funcţie $L : \mathbb{R}^2 \to \mathbb{R}$. Vom spune că L este *funcţie de cost* sau *funcţie de pierdere* (engl., loss function) dacă ea este $i.$ nenegativă şi $ii.$ convexă în raport cu primul argument, pentru orice valoare fixată a celui de-al doilea argument.

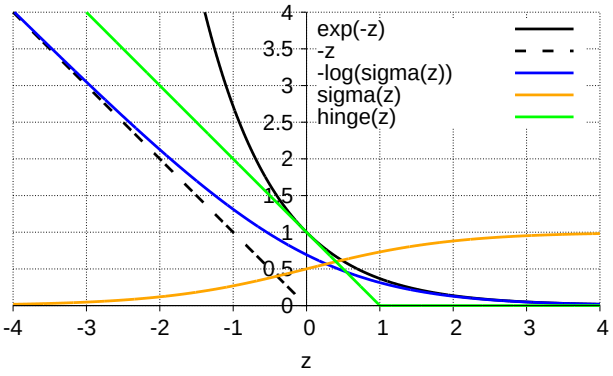
Spre *exemplu*,

– pentru regresia liniară şi pentru perceptronul liniar se foloseşte – cu rol de funcţie de cost – pătratul erorii, adică $L(r, y) = (y - r)^2$,

– pentru regresia logistică se foloseşte *funcţia de cost logistică* $L(r, y) = \ln(1 + e^{-yr})$,

– pentru maşini cu vectori-suport (SVM) se foloseşte *funcţia de cost hinge* $L(r, y) = \max\{0, 1 - yr\}$, iar

– pentru AdaBoost, se foloseşte *funcţia de cost* [*negativ*] *exponenţială* $L(r, y) = e^{-yr}$.



(În figura alăturată, am folosit notaţia $z = yr$.)

Fie de asemenea numerele d şi $n \in \mathbb{N}^*$ şi elementele $x_1, \ldots, x_n \in \mathbb{R}^d$ (numite în învăţarea automată *instanţe neetichetate*), precum şi $y_1, \ldots, y_n \in \mathbb{R}$ (desemnând fie clase, fie — ca în cazul regresiei liniare şi al perceptronului liniar — valori reale oarecare pentru funcţia de învăţat).

Este de remarcat faptul următor (foarte util din punct de vedere teoretic, dar şi din punct de vedere practic): unii algoritmi clasici de învăţare automată — precum cei menţionaţi mai sus — pot fi văzuţi ca rezolvând [câte] o *problemă de optimizare* (cu termen de regularizare de normă L_2) de forma următoare:

$$\min_{w \in \mathbb{R}^d} \left(\frac{1}{n} \sum_{i=1}^{n} L(w \cdot x_i, y_i) + \frac{\lambda}{2} \|w\|^2 \right), \tag{1}$$

unde $\lambda \geq 0$, operatorul $\cdot$ desemnează produsul scalar al vectorilor, operatorul $\| \ \|$ desemnează norma euclidiană ($\|w\| \stackrel{def.}{=} \sqrt{w^2}$), iar L este o anumită funcţie de cost.

Demonstraţi că orice soluţie w^* a *problemei de optimizare* (1) poate fi scrisă sub forma următoare:

$$w^* = \sum_{i=1}^{n} \alpha_i x_i \text{ pentru anumite valori } \alpha_i \in \mathbb{R}. \tag{2}$$

***Observaţie*: Conform relaţiei (2), vectorul-soluţie $w*$ pentru problema de optimizare (1) se „reprezintă“ ca o combinaţie liniară de instanţele $x_1, \ldots, x_n$.**

***Atenţie*: Veţi rezolva problema în *cazul particular* când $\lambda > 0$, iar funcţia de cost $L : \mathbb{R}^2 \to \mathbb{R}$ este derivabilă în raport cu primul argument, pentru orice valoare fixată a celui de-al doilea argument. Puteţi, eventual, să folosiţi următoarele formule de calcul cu *derivate vectoriale* care generalizează bine-cunoscutele reguli de derivare pentru funcţii de o variabilă reală: $\frac{\partial}{\partial w}(a \cdot w) = a$ şi $\frac{\partial}{\partial w} w^2 = 2w$, pentru orice a şi w din $\mathbb{R}^d$.**