Invatare Automata: Laboratorul 12 – **Clasificare folosind Support Vector Machines**

Alexandru Sorici

15. 05. 2017

1 Scoptul Laboratorului

Scopul laboratorului il reprezinta intelegerea si implementarea unui clasificator SVM, ce poate folosi mai multe tipuri de kernel-uri. In cadrul laboratorului se vor face operatii de clasificare pe seturi de date sintetice, ce pun in evidenta utilitatea kernel-urilor atunci cand spatiul de intrare nu este liniar separabil. Rezolvarea cerintelor se va face pas cu pas urmand indicatiile si explicatiile primite.

2 Descriere teoretica

SVM-urile sunt clasificatoare decizionale (i.e. **nu** au ca rezultat o probabilitate ca un input sa faca parte dintr-o clasa data).

O proprietate importanta a SVM-urilor este ca determinarea parametrilor modelului se face prin rezolvarea unei probleme de optimizare convexa, ceea ce inseamna ca orice solutie locala este si un optim global.

SVM-ul este un clasificator binar, iar problema pe care incearca sa o rezolve este gasirea unui **hiperplan de separatie** care maximizeaza *marginea* intre hiperplan si cele mai apropiate exemple pozitive si negative (vezi Figura 2).

2.1 Cazul spatiilor de intrare liniar separabile

Pe cazul liniar, forma pe care o ia clasificatorul este data de ecuatia:

$$y(x) = w^T x + b \tag{1}$$

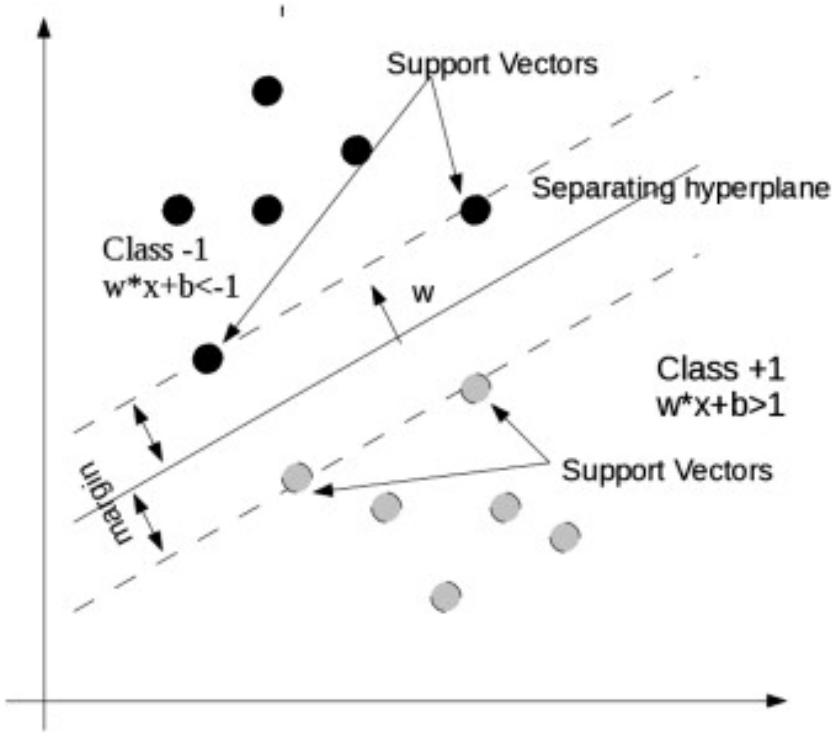

Figure 1: Vizualizare a hiperplanului de separatie si a vectorilor de suport.

unde $y(x)$ este eticheta $\in \{-1, 1\}$ pentru intrarea x, w este un vector de ponderi, iar b este termenul de bias.

In acest caz, problema de optimizare corespunzatoare unui SVM se formuleaza astfel:

$$\begin{aligned} min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ a.i. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, \ldots, n \end{aligned} \tag{2}$$

Ecuatia (2) reprezinta formularea *primara* a problemei de optimizare patratica specifice SVM-ului, care cauta coeficientii w si b a.i. marginea hiperplanului de separatie sa fie maxima.

Forma obtinuta se rezolva insa dificl, astfel incat, in practica se opteaza pentru o formulare *duala*, ce face apel la multiplicatori Lagrange si pentru care se pot utiliza algoritmi de optimizare patratica mai eficienti. Pentru cateva detalii privind dualitatea Lagrange in probleme generale de optimizare parcurgeti sectiunea 5 din aceste notite de curs[1].

[1] http://cs229.stanford.edu/notes/cs229-notes3.pdf

In cazul unui SVM liniar, problema de optimizare duala se scrie precum in ecuatia (3).

$$\begin{aligned} max_a \quad & L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} a_i a_j \langle x^{(i)} x^{(j)} \rangle \\ a.i. \quad & a_i \geq 0, i = 1, \ldots, n \\ & \sum_{i=1}^{n} a_i y^{(i)} = 0 \end{aligned} \tag{3}$$

unde a_i sunt multiplicatorii Lagrange, iar $\langle x^{(i)} x^{(j)} \rangle$ reprezinta produsul scalar al inputurilor $x^{(i)}$ si $x^{(j)}$.
Obtinand valorile multiplicatorilor Lagrange, parametrii initiali ai formularii primare se obtin astfel:

$$w = \sum_{i=1}^{n} a_i y^{(i)} x^{(i)} \tag{4}$$

$$b = \frac{1}{N_S} \sum_{i \in S} \left(y^{(i)} - \sum_{j \in S} a_j y^{(j)} \langle x^{(i)} x^{(j)} \rangle \right) \tag{5}$$

unde S este *multimea vectorilor de suport* ($|S| = N_S$), adica acele intrari $x^{(i)}$ pentru care multiplicatorul Lagrange aferent $a_i = 0$. Cu alte cuvinte, doar *vectorii de suport* vor juca un rol in predictiile pe care le face un SVM antrenat. Forma initiala a predictorului poate fi rescrisa in functie de multiplicatorii Lagrange ca fiind:

$$y(x) = \sum_{i \in S} a_i y^{(i)} \langle x^{(i)} x \rangle + b \tag{6}$$

2.2 Cazul spatiilor de intrare ne-separabile liniar

In multe probleme de clasificare se poate intampla ca spatiul intrarilor sa nu admita o separare liniara a inputurilor pozitive si negative. In acest caz, o tehnica adoptata este cea a unei transformari a vectorilor de intrare intr-un *spatiu al caracteristicilor (input space to feature space mapping)*, avand o dimensionalitate mai mare decat cea a spatiului initial de intrare. Intuitia este ca aceasta mapare poate duce la gasirea unui hiperplan de separatie in spatiul caracteristicilor.
Forma clasificatorului SVM devine atunci:

$$y(x) = w^T \phi(x) + b \tag{7}$$

unde $\phi(x)$ este functia de mapare in spatiul caracteristicilor (feature space).

In practica, un spatiu al caracteristicilor mai mare presupune un efort computational mai mare. Pentru a evita acest lucru, se face uz de asa numitul "kernel trick"[2]. Acesta se refera la capacitatea unor functii $k(x, x^{'})$ definite peste un domeniu χ de a fi exprimate ca produs scalar intr-un alt domeniu ν.
Functia $k : \chi \times \chi \rightarrow R, k(x, x^{'}) = \langle \phi(x), \phi(x^{'}) \rangle_{\nu}$ poarta atunci numele de functie kernel.

In cadrul laboratorului vom folosi doua astfel de kernel-uri: cel liniar (care corespunde produsului scalar exact in spatiul de intrare) si Radial Basis Function[3].

Folosind functiile kernel, reprezentarea duala a problemei de optimizare din cadrul SVM-urilor (ecuatia (3)) se rescrie astfel:

$$\begin{aligned} max_a \quad & L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} a_i a_j k(x^{(i)}, x^{(j)}) \\ a.i. \quad & a_i \geq 0, i = 1, \ldots, n \\ & \sum_{i=1}^{n} a_i y^{(i)} = 0 \end{aligned} \tag{8}$$

Predictia unui SVM antrenat (i.e. pentru care s-au determinat multiplicatorii Lagrange) ajunge sub forma:

$$y(x) = \sum_{i \in S} a_i y^{(i)} k(x^{(i)}, x) + b \tag{9}$$

In practica, in procedura de antrenare a unui SVM, se precalculeaza valorile functiei kernel alese pentru orice pereche $x^{(i)}, x^{(j)}$, stocand valorile intr-o matrice $K_{i,j} = k(x^{(i)}, x^{(j)}$, denumita *matricea Gram* sau *matricea kernel*.

Cazul claselor suprapuse si gestiunea regularizarii

Trecerea intr-un spatiu al caracteristicilor de dimensionalitate mai mare (explicat in sectiunea 2.2) creste de multe ori probabilitatea ca datele sa ajunga separabile liniar. Totusi, acest lucru nu poate fi garantat.
In plus, in unele cazuri nu este clar daca a gasi un hiperplan de separatie exacta este lucrul cel mai de dorit, pentru ca putem ajunge susceptibili la valori extreme (outliers).

[2]https://en.wikipedia.org/wiki/Kernel_method
[3]https://en.wikipedia.org/wiki/Radial_basis_function_kernel

Pentru a aborda problemele acestea, in problema de optimizare a SVM-ului se introduc variabile de regularizare ce au menirea sa reduca "strictetea" cu care se urmareste gasirea hiperplanului cu margine maximala.

Forma problemei de optimizare primara se modifica astfel:

$$\begin{aligned} min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ a.i. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, \ldots, n \\ & \xi_i \geq 0, i = 1, \ldots, n \end{aligned} \tag{10}$$

Astfel, pentru fiecare vector de intrare permitem acum de a avea o margine ≤ 1, iar daca un exemplu din setul de date are marginea $1-\xi_i$, costul functiei obiectiv ar creste cu $C\xi_i$. Parametru C controleaza astfel importanta pe care o acordam clasificarii corecte a fiecarui punct din setul de date. Pentru valori mari (e.g. 1000) se poate obtine un hiperplan cu margine mai mica, daca acel hiperplan va clasifica in mod corect fiecare exemplu. Pentru valori mici (e.g. 0.1), hiperplanul va avea margine mai mare, dar va permite clasificarea incorecta a catorva exemple.

In privinta problemei duale, formularea acesteia devine:

$$\begin{aligned} max_a \quad & L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} y^{(i)}y^{(j)}a_i a_j k(x^{(i)}, x^{(j)}) \\ a.i. \quad & 0 \leq a_i \leq C, i = 1, \ldots, n \\ & \sum_{i=1}^{n} a_i y^{(i)} = 0 \end{aligned} \tag{11}$$

Formularea pentru predictia SVM-ului ramane aceeasi, *dar* setul vectorilor de suport S se modifica, incluzand doar acele intrari $x^{(i)}$ pentru care $0 < a_i < C$

3 Biblioteci necesare si descrierea fisierelor

Biblioteci necesare

In cadrul laboratorului se vor folosi bibliotecile `numpy`, `sklearn` si `cvxopt`. Pentru instalarea tuturor pachetelor necesare rulati makefile-ul atasat.

Nota: pe majoritatea calculatoarelor din laborator, bibliotecile sunt deja instalate. Verificati acest lucru deschizand o consola si ruland:

```
python
import cvxopt
```

Daca nu primit eroare, inseamna ca biblioteca este instalata.

Descriere fisiere

- `svm-skel.py` – contine implementarea clasei de antrenare si de predictie a clasificatorului SVM. Cerinta 4.1 presupune definirea matricei Gram, necesara in calculul parametrilor Lagrange. Cerinta 4.2 presupune formularea problemei de optimizare duale, conform pseudocodului indicat.
- `kernel-skel.py` – contine definitia a doua functii ce implementeaza doua tipuri de kernel diferite: unul liniar si unul bazat pe Radial Basis Functions[4]. Cerinta 4.3 presupune implementarea acestor kernel-uri.
- `svm-test.py` – contine cod ce creeaza seturi de date sintetice (moons[5], circles[6] si un set de date liniar separabil). Scriptul va antrena clasificatorii SVM creati pe baza fiecarui tip de kernel si ii va evalua pe seturile de date generate. Pentru fiecare, scriptul traseaza curba de separatie si afiseaza acuratetea clasificarii. Intrarile care devin *vectori de suport* sunt reprezentate ca cercuri, iar restul ca '+'-uri. Cerinta 4.4 presupune modificarea parametrului C al fiecarui clasificator SVM antrenat folosind puteri ale lui 10 ($C \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$) si analiza modului in care se schimba curba de separatie.

4 Cerinte

4.1 Cerinta 1

Completati in fisierul `kernel-skel.py` cele doua functii corespunzatoare unui kernel liniar si al unuia bazat pe Radial Basis Function.
Kernelul liniar are forma:

$$k(x, y) = \langle x, y \rangle \tag{12}$$

Pentru produsul scalar a doi vectori, uitati-va la functia `numpy.inner` din pachetul `numpy`.

Kernelul bazat pe Radial Basis Function are forma:

$$k(x, y) = exp(-\gamma \|x - y\|^2) \tag{13}$$

[4]https://en.wikipedia.org/wiki/Radial_basis_function_kernel
[5]http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html
[6]http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html

Utilizati functiile `numpy.exp`, `numpy.subtract` si `numpy.linalg.norm` pentru implementarea kernel-ului.

4.2 Cerinta 2

Completati in fisierul `svm-skel.py` functia `_gram_matrix`, astfel incat aceasta sa intoarca matricea Gram (K), dandu-se setul de date X.

4.3 Cerinta 3

Completati in fisierul `svm-skel.py` functia `_compute_multipliers`, astfel incat aceasta sa intoarca lista multiplicatorilor Lagrange ce rezolva formularea duala a problemei de optimizare SVM descrisa de ecuatia (11).
Biblioteca `cvxopt` dispune de o metoda de rezolvare a problemelor de optimizare patratica ce au urmatoarea forma:

$$\begin{aligned} min \quad & \frac{1}{2}x^T Px + q^T x \\ a.i. \quad & Gx \prec h \\ & Ax = b \end{aligned} \tag{14}$$

Problema duala pentru SVM din ecuatia (11) poate fi rescrisa sub forma matriciala astfel:

$$\begin{aligned} min \quad & \frac{1}{2}a^T Qa - 1^T a \\ a.i. \quad & 0 \leq a_i, i = 1, \ldots, n \\ & a_i \leq C, i = 1, \ldots, n \\ & y^T a = 0 \end{aligned} \tag{15}$$

unde $Q = (yy^T) \cdot K$ (i.e. inmultire matriceala intre y si y^T si apoi inmultire element-cu-element cu matricea K).

Urmariti TODO-urile din corpul functiei `_compute_multipliers` pentru a mapa forma ecuatiei (15) in cea a ecuatiei (14).
Hint: folositi functiile `cvxopt.matrix`, `numpy.outer`, `numpy.ones`, `numpy.zeros`, `numpy.diag` si `numpy.vstack`. **Folositi peste tot unde creati matrici sau vectori, apelul `cvxopt.matrix()` ca wrapper peste expresii de numpy. Wrapper-ul este necesar apelarii ulterioare a solver-ului din biblioteca cvxopt.**

4.4 Cerinta 4

Analizati modul in care se modifica scorul de clasificare si forma curbei de separatie atunci cand variati parametrul C in intervalul $[10^{-3}, 10^{3}]$ prin puteri ale lui 10.
Implementati o functie `num_misclassified` in clasa `SVMTrainer` care sa intoarca numarul de vectori de intrare clasificati incorect. Luati drept punct de plecare codul din functia `score` din clasa `SVMPredictor`. Observati influenta parametrului C asupra rezultatului intors de functia `num_misclassified`.