

**SUPPORT VECTOR MACHINES**

**PAPER 25 MML**

**GHERASIM MIHAI**

**ILICEA ANCA-ȘTEFANIA**

**NEGRU LUIZA**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**2020-2021**

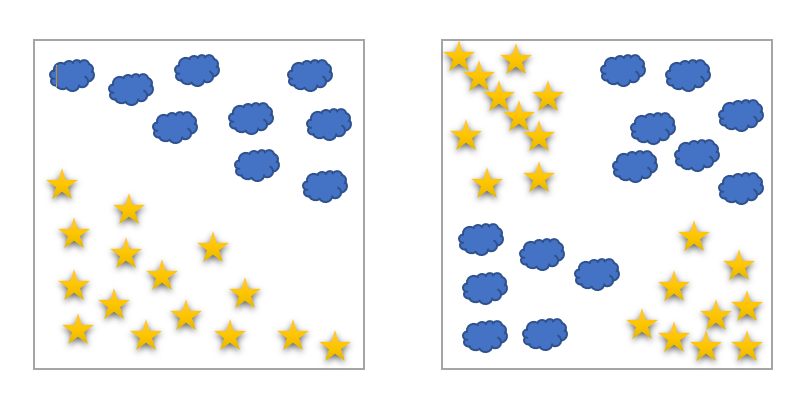
Clasificarea reprezintă, în general, un proces de predicție a claselor din care fac parte un set de date de intrare. Conceptul poate fi implementat utilizând un clasificator. Acesta utilizează date de antrenare pentru a „înțelege” legătura dintre variabilele de intrare și clasa din care fac parte. Câteva exemple de clasificatori sunt k-Nearest Neighbors, SVM, Decision Trees, etc.

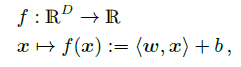
Clasificatorul abordat în lucrarea aleasă este SVM. SVM sau mașinile cu vectori suport reprezintă una dintre cele mai folosite metode în învățarea automată ea fiind bazată în mare parte pe teoria învățării statistice, dar și pe o viziune geometrică. În prima parte a lucrării este subliniată utilitatea și aplicabilitatea acestei metode în cadrul diverselor domenii precum mesajele online(email) și astronomie. Mașinile cu vectori suport reprezintă un model liniar utilizat atât pentru problemele de regresie cât și pentru problemele de clasificare. Acestea pot rezolva atât probleme liniare, cât și ne-liniare, ideea din spatele modelului fiind relativ simplă: algoritmul creează o linie sau un hiperplan care separă datele în clase. Deși SVM a fost conceput pentru clasificarea binară, există câteva soluții care fac posibilă utilizarea modelului și pentru problemele de clasificare multiclase, însă in cadrul lucrării studiate este abordată doar clasificarea binară.

În vederea clasificării este nevoie de un set de date de antrenare etichetate cu clasa din care corespund, etichete care vor devenii și rezultatul clasificatorului, acestea pot fi de orice tip(string, boolean, sau chiar număr). Setul de antrenare va fi de forma {(exemplu1, etichetă1), (exemplu2, etichetă2), ..., (exempluN, etichetăN)}, iar clasificatorul se va antrena pe aceste date în vederea obținerii unei erori cât mai mici, respectiv acurațe cât mai mare la testarea pe o serie de date (neetichetate) complet noi.

Sunt prezentate câteva dintre avantajele mașinilor cu vectori suport printre care se număra oferirea unei viziuni geometrice în învățarea automată bazându-se pe concenpte matematice precum proiecțiile și construirea unei funcții specifice care va fi optimizată pe parcursul procesului de antrenare.

Într-o mare parte a lucrării este analizată metoda prin care se poate trasa cel mai optim un hiperplan pentru a separa 2 clase, daca acestea pot fi separate. În cele ce urmează voi prezenta două exemple pentru cele două cazuri expuse.

Se poate observa ușor faptul că în poza din stânga clasa ce reprezintă steluțele galbele poate fi separată prin intermediul unui hiperplan de clasa norișorilor albaștrii, deci clasele sunt separabile. În cadul imaginii din dreapta, deși avem aceleași doua clase, acestea nu sunt separabile. Ce înseamna de fapt ca două clase să fie separabile? Știm deja că orice hiperplan va separa spațial în două. Prin urmare, clasele sunt separabile dacă “ai loc sa tragi o dreaptă” care le poate separa. În acest context se amintește despre soft margin: cazul în care admiți ca hiperplanul să facă și greșeli, respectiv hard margin: când TODO

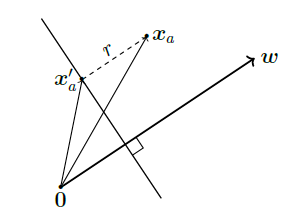
Toată lucrarea se referă, de fapt la modalitatea prin care se poate găsi un hiperplan optim, în cadrul căruia suma erorilor este minimă. În vederea trasării hiperplanului se găseste mai întâi o funcție care ia ca input un vector și scoate un scalar.

Se dorește astfel alegerea hiperparametriilor w și b astfel încât să dea un rezultat pozitiv pentru o clasă și negativ pentru cealaltă ( exemplul din lucrare având etichetele +1 și -1 pentru cele două clase). Altfel spus, având setul inițial {(exemplu1, -1), (exemplu2, +1), ..., (exempluN, +1)} se dorește ca toate exemplele care au asociată eticheta +1 sa rezulte în urma calculului funcției +1, similar pentru cazul cu -1. Sigur că în urma rezolvării ecuației propuse:

Se poate găsi o infinitate de hiperplane, însă există un singur hiperplan care face separarea cea mai bună. Pentru a-l găsi pe cel mai bun și a afla cât de bun este avem nevoie de fapt de mașini cu vector suport.

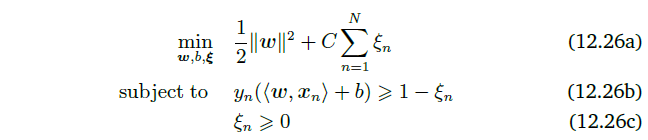
**Conceptul de margine**

În acest sens se vorbeste despre metoda alegerii hiperplanului optim cu ajutorul proiectiilor. Practic, pentru fiecare punct dintr-o clasă se va trasa proiecția sa pe un hiperplan găsit și se caută cea minimă. Acest pas se va repeta pentru fiecare hiperplan generat, iar cel optim va fi, de fapt cel care are proiecția minimă, maximă.

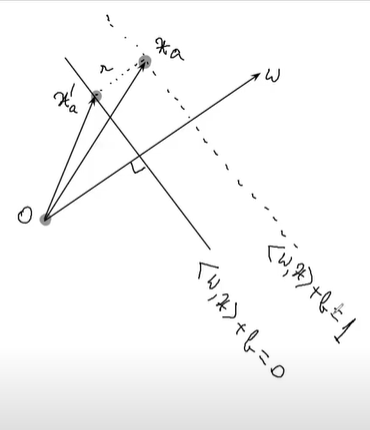
Hiperplanele se găsesc rezolvând ecuatia (1), iar considerând imaginea urmatoare din articol:

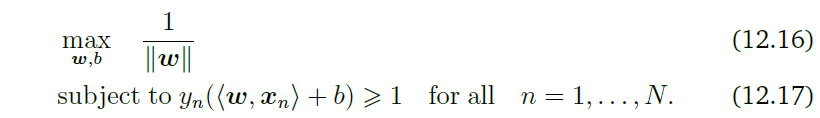
Linia portocalie reprezintă hiperplanul ales în 2D, va fi o dreaptă, iar x\_a un punct exterior hiperplanului în care ecuața (1) nu rezultă 0, pentru exemplul anterior de fapt una din steluțe, se va calcula proiecția ei pe hiperplan care este coliniară cu w, deci se poate scrie ca multiplu de w care ne dorim sa minim din moment ce r trebuie maxim deoarece acestea sunt inversproporționale, fapt demonstrat în lucrare.

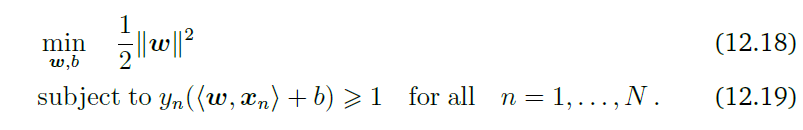
Se prezintă în continuare metoda prin care r este maximizat. Astfel SVM-ul va trasa doua hiperplane imaginare pe care le vor atinge cele doua clase.

Separarea perfectă dată de un hiperplan este, în principiu greu de atins, așadar în continuare este abordată metoda care trebuie aplicată pentru clasele care nu pot fi perfect separate. Nu se va cauta în acest sens un hiperplan care face o separare perfectă, ci unul care va face o separare cu un număr minim de erori. Eroarea este notata cu csi și este adăugată in formulă în acest caz.

**Hard margin SVM**

******

O altă abordare poate fi de a scala datele astfel încât valoarea predictorului f(x) = ‹w, x› + b sa fie 1 pentru cel mai apropiat exemplu de hiperplan. Dacă presupunem că punctul xa este cel mai aproape, rescalăm axele astfel încât xa să se afle exact pe margine, iar puctul xa’ este proiecția lui ortognală pe hiperplan. Prin substituții si simplificări începând de la egalitațile: și yn(‹w, xn› + b) ≥ r , ajungem la , pentru că vrem ca exemplele pozitive și exemplele negative să fie la distanță de cel puțin 1 de hiperplan, rezultă condiția: yn(‹w, xn› + b) ≥ 1. Combinând maximizarea marjei cu faptul că exemplele trebuie să fie pe partea corectă a hiperplanului (pe baza etichetelor lor) ne oferă:

Iar ceea ce rezulta minimizând in loc să maximizăm, si adaugând o constanta de , cunoscută și sub numele de hard margin SVM este:

**Soft margin SVM**

În cazul în care datele nu sunt linear separabile, se dorește permiterea anumite erori în clasificarea datelor. Așadar se va prezenta în continuare SVM-urile cu margini flexibile („soft”).

Noua constrângere permite o marjă funcțională mai mică de 1 și conține o penalizare a costului Cξi pentru orice punct de date care se încadrează în marja de pe partea corectă a hiperplanului de separare (adică, când 0 < ξi ≤ 1), sau pe partea greșită a hiperplanului de separare (adică, atunci când ξ i> 1). Declarăm astfel o preferință pentru marjele care clasifică corect datele de antrenament, dar “înmoaie” constrângerile pentru a permite date care nu pot fi separate cu o penalitate proporțională cu suma cu care exemplul este clasificat greșit. C este parametru de cost care controleaza compromisul dintre maximizarea marginii si minimizarea erorii de clasificare. Prin urmare, am dori să minimizăm suma totală a penalităților ξi pentru i = 0, ..., n (aceasta este o limită superioară a erorilor de clasificare).



Considerăm problema de clasificare binara (rezultatul predictiei este una dintre cele două etichete {+1, −1}), funcția de eroare / pierdere pentru fiecare exemplu de etichete trebuie să fie adecvată pentru clasificarea binară. De exemplu, pierderea pătratică care este utilizată pentru regresie nu este potrivită pentru clasificarea binară.

Funcția ideală de pierdere între etichetele binare este de a număra câte nepotriviri există între predicție și etichetă. Aceasta înseamnă că pentru un predictor f aplicat unui exemplu xn, comparăm ieșirea f (xn) cu eticheta yn, unde f este: f(x) = ‹w, x› + b.

Echivalenta cu loss function este e hinge loss, l(t) = max{0, 1 − t}, unde t = y f(x) = y(‹w, x› + b).

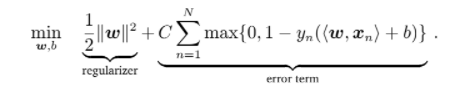
În funcție de unde se regăsește puctul față de hiperplan avem 3 cazuri:

* Dacă f (x) este pe partea corectă a hiperplanului, și mai departe decat 1 ca si distanta, atunci înseamnă că t ≥ 1, hinge loss returnează o valoare zero
* Daca f (x) este pe partea corecta, dar prea aproape de hiperplan atunci 0 < t < 1,

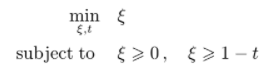
hinge loss-ul returneaza o valoare pozitiva

* Când exemplul este pe partea greșită a hiperplanului, t < 0, hinge loss-ul returnează o valoare si mai mare, care crește liniar

În cazul în care predicția este corectă, penalitatea crește linear dacă ne apropiem de marginea hiperplanului, acest fapt nu este favorabil noua, așa că putem folosi următoarea alternativă:

Căutăm să minimizăm pierderea totală, folosind hinge loss pe un set de date {(x1, y1), . . . ,(xn , yn )}.

Putem înlocui în mod echivalent minimizarea pierderii articulației peste t cu a

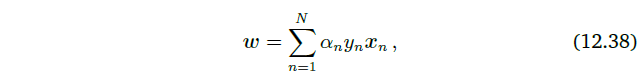
minimizarea unei variabile slack ξ cu două constrângeri. În formă de ecuație,

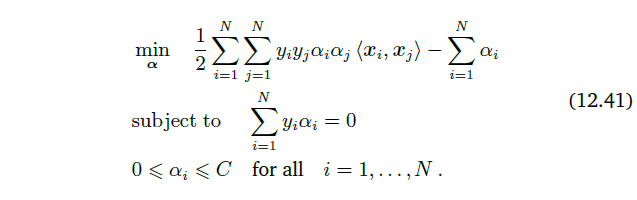
care este exchivalenta cu

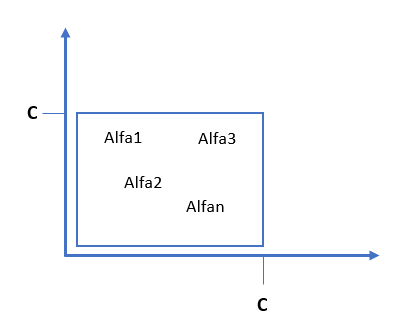
Ajungand astfel la soft margin SVM.

**Mașini cu vector suport duale**

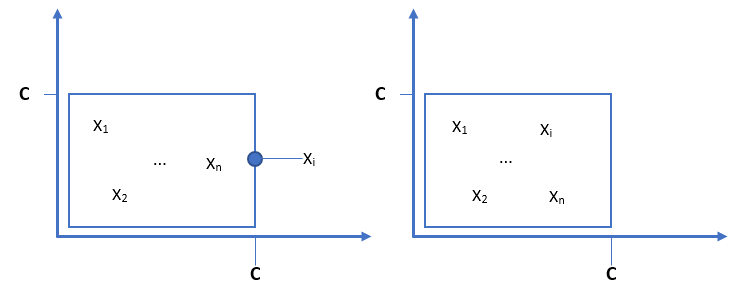
Se trece în continuare la rezolvarea problemei de SVM folosind dualitatea. Practic, pentru rezolvarea problemei primale SVM se va reprezenta problema în formă duală și se va încerca rezolvarea ei. În comparație cu problema primală, abordarea folosind forma duală nu mai este dependentă de numărul de caracteristici din datele de antrenare, ci numărul de parametri apăruți în cadrul problemei duale depinde de numărul de date din setul de antrenare. De aici concludem că această metodă de rezolvare se întamplă de cele mai multe ori în momentul în care numărul de caracteristici extrase este mai mare decât numărul de date din setul de antrenare.

Definind functia Lagrange pe baza ecuatiei 12.26a si obtinand 12.34 construim derivatele partiale ale functiei in raport cu cele trei variabile primale w, b si csi. În urma rezolvării ecuatiilor cu derivate partiale prin egalarea acestora cu 0, obținem 12.38 care ne spune că soluția pentru problema de optimizare se află chiar în datele de antrenare i.e. vectorul w din problema primală este o combinație liniară a exemplelor din setul de antrenare.

Înlocuind in ecuația initială 12.34 w cu 12.38 și egalând derivatele parțiale pentru a le inlocui și pe acestea. Astfel se opține problema duală de SVM care trebuie minimizată.

Practic rezolvarea problemei s-a redus aflarea parametrului alpha cu condițiile impuse de Lagrangian se va desfășura astfel căutarea într-un box de tipul:

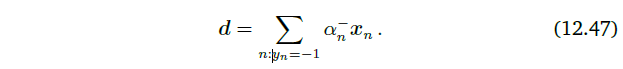
O remarcă interesantă subliniată în capitolul referitor la dualitate este explicarea provenienței numelui de Support Vector Machines. Astfel încât, pentru unii alfa care vor fi 0, conform formulei 12.23 valorile x si y asociate cu acestea (exemplul din setul de antrenare împreună cu eticheta sa vor devenii și ele 0 spre urmare acestea nu vor contribui la rezolvarea problemei, cele rămase, în schimb, vor reprezenta un “suport” în rezolvarea problemei, de aici fiind posibilă extragerea numelui metodei.

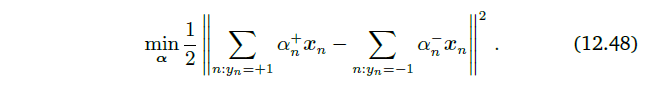
O dată ce parametrii alfa au fost aflați în urma rezolvării problemei de optimizare, putem calcula w deoarece nu mai avem alte necunoscute, acesta fiind numit în secțiune w\* pentru a se putea recunoaște că el este cel optim. Sigura necunoscută rămasă este b\* corespunzător lui w\* care se va calcula prin intermediul formulei 12.42 dacă macar unul din exemplele xn se află pe conturul cutiei sau se calculează pentru toate exemplele din antrenare cu ajutorul formulei dacă niciunul dintre exemple nu se află pe conturul cutiei și se află mediana acestora dacă se află în interior.

**Dualitatea văzulă prin intermediul acoperirii convexe**

 O altă modalitate prin care poate fi văzută problema duală este din punct de vedere geometric prin intermediul acoperirilor convexe. Se demonstrează în continuare faptul că orice punct de pe acoperirea convexă poate fi scris ca o reprezentare a celorlalte puncte. Se va considera o înfășurătoare convexă pentru fiecare dintre cele două clase și două puncte c și d presupunând că c este cel mai apropiat punct de mulțimea opusă făcând parte din mulțimea pozitivă, iar d este cel mai apropiat punct de mulțimea opusă făcând parte din mulțimea negativă. Pentru ca c și d să fie cât mai apropiate unul de celălalt se încearcă minimizarea diferenței c – d rezultând problema:

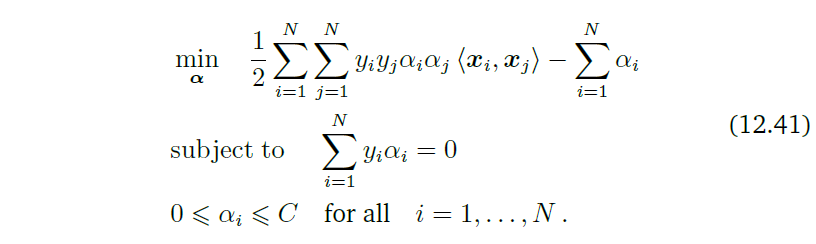
Din moment ce variabila c este pe partea pozitivă, conform afirmațiilor precedente aceasta se poate nota cu:

Analog d se poate exprima ca și următoarea sumă

Prin înlocuiri, aducând la un loc (12.46), (12.47) în cadrul (12.45) se obține problema de optimizare (12.48) constânsă de faptul că în cadrul acoperirilor convexe suma coerficienților trebuie să fie 1.

**Experimente și rezultate**

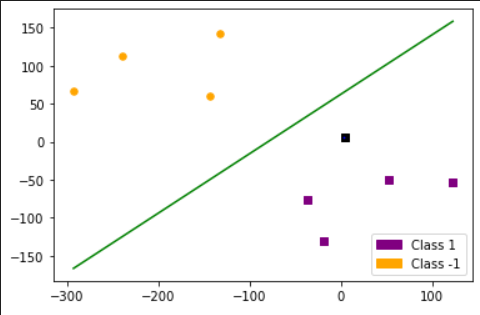
În vederea efectuării experimentelor au fost folosite seturi de date compuse din 2 clase cu etichetele yi = { -1, 1 } ce reprezintă coordonate de puncte în 2D. Acestea au fost generate prin intermediul [1]. Algoritmii utilizați în vederea analizei performanței sunt Sequential Minimal Optimization [2] și Projected Gradient Descent.

Problema de optimizare rezolvată de ambii algoritmi este:

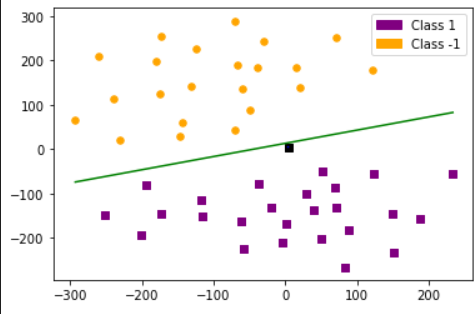
|  |  |
| --- | --- |
| **Gradient proiectat** | **Sequential Minimal Optimization** |
| Rezultate pentru un set compus din 8 de exemple, **C = 0.5**  Timp de rulare: **7.28s** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Gradient proiectat**

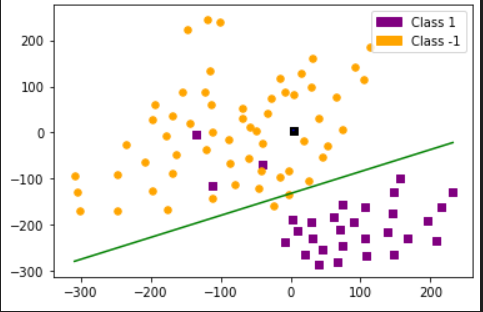
Rezultate pentru un set compus din 8 de exemple, C = 0.5

Timp de rulare: 7.28s

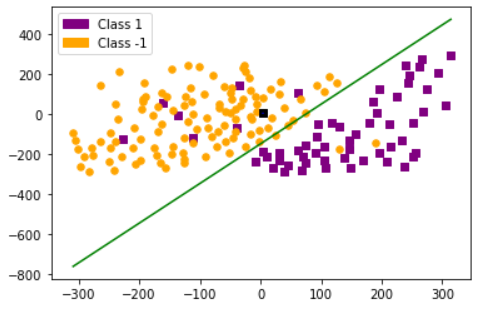
Rezultate pentru un set compus din 47 de exemple, C = 10-4

Timp de rulare: 9.30

Rezultate pentru un set compus din 89 de exemple, C = 10-4

Timp de rulare: 18.08

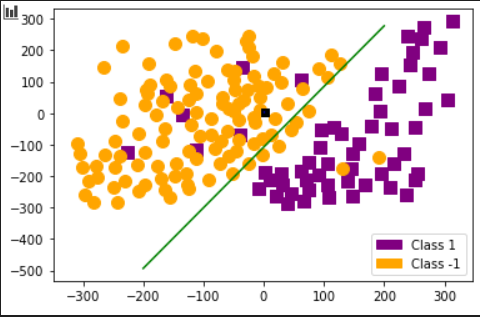
Rezultate pentru un set compus din 172 de exemple, C = 10-4

Timp de rulare: 36.82

1. **Sequential Minimal Optimization**

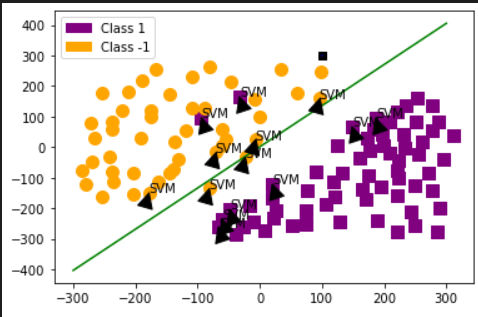
Mențiune: Punctele marcate cu negru sunt cele pentru care parametrul lagrangian alfa este diferit de 0, deci contribuie la aflarea parametrilor hiperplanului w și b. Punctele portocalii fac parte din mulțimea marcată cu -1, iar pătrațelele mov fac parte din clasa cu eticheta pozitivă.

Rezultate pentru un set compus din 172 de exemple, *C* = 0.001:

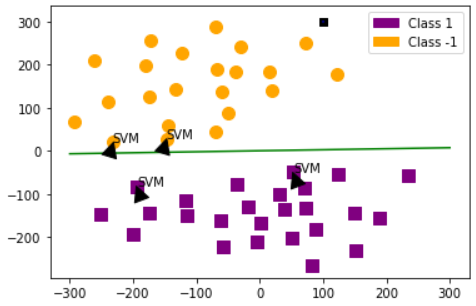
Timp de rulare: 14.03

Rezultate pentru un set compus din 112 de exemple, *C* = 0.001:

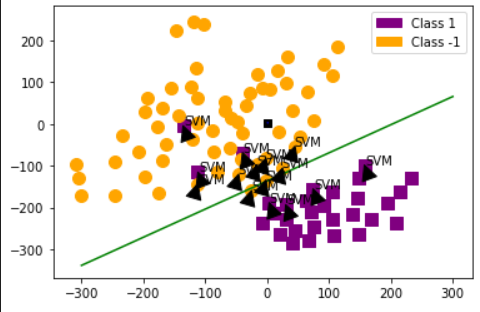
Timp de rulare: 5.03

Pentru acest caz am constatat un timp foarte mare de rulare a algoritmului pentru un C mai mare de 0.001, pentru *C =* 0.6 în urma a 15 minute soluția dată încă nu reușise să genereze variabilele alfa.

Rezultate pentru un set compus din 47 de exemple, *C* = 0.1

 Timp de rulare: 0.75

Pentru 89 de exemple și C = 0.001

Timp de rulare: 9.41s

În cazul unui set de date foarte mic, de 10 exemple, chiar si pentru o variabilă C mare, algoritmul va converge rapid

Timp de rulare:

**Concluzii**

În urma testelor folosind ambii algoritmi se poate concluziona faptul că deși timpul de rulare depinde direct proporțional de performanța dispozitivului pe care sunt rulate, totuși soluția bazată pe gradient proiectat converge mult mai greu decât cea cu SMO.

# Bibliography

|  |  |
| --- | --- |
| [1] | M. Hancock, "2D, Two-class Classification Dataset Maker," [Online]. Available: http://notmatthancock.github.io/software/dataset2d/. [Accessed 23 5 2021]. |
| [2] | Platt and John, "Fast Training of Support Vector Machines using Sequential Minimal Optimization," MIT Press, 1998. |