AMO

PROJEKT 2

RAWICKI PAWEŁ

OPIS PROJEKTU

W ramach projektu zostały zaimplementowane skrypty do wyznaczania podziałów na dwa zbiory.

Zadanie zostało wykonane przy następującej deompozycji:

1. W pierwszej części, wyznaczyłem dodatkowo SVM dla płaszczyzny w 3 wymiarach. Zrobiłem to w celu, weryfikacji zaimplementowanego rozwiązania oraz, by zwizualizować problem.
2. Rozwiązanie dla płaszczyzny trzy wymiarowej zostało zostało rozszerzone następnie do przestrzeni pięciowymiarowej i dla danych wdbc.
3. W rozwiązaniu korzystałem z dwóch podejść, do wyznaczania rozwiązania: funkcji solve, która automatycznie wybiera odpowiedni algorytm z zestawu funkcji optymalizyjących Matlaba. Oraz funkcją fmincon dla zadania prymalnego i quadprog dla zadania dualnego
4. W zadaniu prymalnym został wykorzystany twardy margines
5. W zadaniu z własnym danymi, losowo je generowałem, tyle samo punktów uczących co testowych
6. Dane wdbc arbitralnie podzieliłem na zbiór uczący się i testowy
7. Została zbadane dwie następujące płaszczyzny, punkty wybrałem tak, by nie były zbytnio oczywiste:

```
% Do wielu wymiarów
% baseRatios=[10,15,20,25,30,50];

% Do trzech wymiarów.
% A=-10;
% B=10;
% C=20;
% D=30;
```

WNIOSKI

Wnioski załaczam na samym początku, projektu, ponieważ niżej załącze wiele funkcji

Implementacja solvera dla funkcja solve, ma kolosalne znaczenie. Dla niezakomentowanej treści, została wykorzystana funkcja fmincon. Dla zakomentowanej, było wykorzystywany solver quadprog, czyli z programowania kwadratowego. Ponadtwo, w przypadku z pętlą, algorytm działał zdecydowanie wolniej, a dla danych rzędu 200 punktów, nie udało mi się otrzymać wyników, ponieważ dział zbyt długo. Wynika z tego, że mnożenie macierzy działa zdecydowanie szybciej w Matlabie niż iterowanie po pętli

```
type solveOptimizeThis.m
```

```
function [result]=solveOptimizeThis(c,points,y)
    K = points*points';
    H = diag(y)*diag(c)*K*diag(y)*diag(c);
    result =(sum(c)-0.5*sum(sum(H)));
end
```

```
% SLOWER VERSION, SOLVER TAKES QUADPROG TO THIS
% function [result]=solveOptimizeThis(c,points,y)
%     sumIJ=0;
%     for i=1: length(points)
%       for j=1: length(points)
%           sumIJ=sumIJ+y(i)*c(i)*points(i,:)*points(j,:)'*y(j)*c(j);
%       end
%     end
%     result =(sum(c)-0.5*sumIJ);
% end
```

Wykorzystanie funkcji solve jest zdecydowanie wolniejsze niż użycie quadprog lub fmincon

PRZEBIEG PROJEKU

WYKORZYSTANE FUNKCJE

FUNKCJA DO GENEROWANIA DANYCH

```
type generateData.m
```

```
function [pointsAbove,pointsBelow, pointsLearn,pointsAboveTest,pointsBelowTest,pointsTest,y] = generateData(amountOf
    dimensionsAmount=length(baseRatios)-1;

    pointsAbove =zeros(amountOfPointsAbove,dimensionsAmount);
    pointsBelow=zeros(amountOfPointsBelow,dimensionsAmount);

    pointsAboveTest =zeros(amountOfPointsAbove,dimensionsAmount);
    pointsBelowTest=zeros(amountOfPointsBelow,dimensionsAmount);

    for k=1:amountOfPointsAbove
        points=rand(dimensionsAmount-1,1)*200-100;
        z=(baseRatios(1:end-2)*points+baseRatios(end))/(-baseRatios(end-1));
        z=z+rand(1)*100;
        pointsAbove(k,:)=[points',z];

        points=rand(dimensionsAmount-1,1)*200-100;
        z=(baseRatios(1:end-2)*points+baseRatios(end))/(-baseRatios(end-1));
        z=z+rand(1)*100;
        pointsAboveTest(k,:)=[points',z];
    end

    for k=1:amountOfPointsBelow
        points=rand(dimensionsAmount-1,1)*200-100;
        z=(baseRatios(1:end-2)*points+baseRatios(end))/(-baseRatios(end-1));
        z=z-rand(1)*100;
        pointsBelow(k,:)=[points',z];

        points=rand(dimensionsAmount-1,1)*200-100;
        z=(baseRatios(1:end-2)*points+baseRatios(end))/(-baseRatios(end-1));
        z=z-rand(1)*100;
        pointsBelowTest(k,:)=[points',z];
    end

    pointsLearn=[pointsAbove; pointsBelow];
    pointsTest=[pointsAboveTest; pointsBelowTest];

    y=[ones(amountOfPointsAbove,1);-ones(amountOfPointsBelow,1)];
end
```

FUNKCJA DO RYSOWANIA WYNIKÓW

```
type drawSurface.m
```

```
function  drawSurface(chartTitle,functionDuration,ratios,baseRatios,pointsAbove,pointsBelow, badlyClassified)
    amountOfPointsAbove=length(pointsAbove);
    amountOfPointsBelow=length(pointsBelow);
    amountOfPoints=amountOfPointsAbove+amountOfPointsBelow;

    [X,Y] = meshgrid(-100:5:100,-100:5:100);
    Z = (baseRatios(1)*X+baseRatios(2)*Y+baseRatios(4))/(-baseRatios(3));

    Z1=(ratios(1)*X+ratios(2)*Y+ratios(4))/(-ratios(3));
    figure
    CO(:,:,:) = ones(length(X)); % red
    CO(:,:,2) = ones(length(X)).*linspace(0.5,0.6,length(X)); % green
    CO(:,:,3) = ones(length(X)).*linspace(0,1,length(X)); % blue
    surf(X,Y,Z,CO)
    hold on
    surf(X,Y,Z1)
    hold on
    for k=1:amountOfPointsAbove
        scatter3(pointsAbove(k,1),pointsAbove(k,2),pointsAbove(k,3),'r')
    hold on
    end
    for k=1:amountOfPointsBelow
        scatter3(pointsBelow(k,1),pointsBelow(k,2),pointsBelow(k,3),'b')
    hold on
    end
    title(chartTitle +newline +...
        "WSPÓŁCZYNNIKI PŁASZCZYZNY: "+strjoin(string(baseRatios), ', ') + newline+...
        "WYLICZONE WSPÓŁCZYNNIKI PŁASZCZYZNY: "+strjoin(string(ratios), ', ') + newline+...
        "LICZBA PUNKTÓW: "+ amountOfPoints+newline+...
        "CZAS[s] LICZENIA PRZEZ SOLVER: "+functionDuration +newline+  ...
        "ŹLE SKLASYFIKOWANE: "+ badlyClassified)

    legend(["PŁASZCZYZNA ZADANA", "PŁASZCZYZNA WYZNACZONA"])
    hold off
end
```

FUNKJCA DO SPRAWDZANIA WYNIKÓW

```
type validateResults.m
```

```
function [countBadlyClassifiedLearnData,countBadlyClassifiedTestData] = validateResults(pointsLearn,yLearn,pointsTes
    countBadlyClassifiedLearnData=0;
    countBadlyClassifiedTestData=0;
%     badlyClassifiedLearnData=[];
%     badlyClassifiedTestData=[];
    for k=1:length(pointsLearn)
      resultLearn=  [pointsLearn(k, :),1]*ratios';
      if(resultLearn*yLearn(k)<=0)

%           badlyClassifiedLearnData(k,:)=[k, resultLearn];
          countBadlyClassifiedLearnData=countBadlyClassifiedLearnData+1;
      end

    end
    for k=1:length(pointsTest)
      resultTest=  [pointsTest(k, :),1]*ratios';
      if(resultTest*yTest(k) <=0)
%         badlyClassifiedTestData(k,:)=[k, resultTest];
        countBadlyClassifiedTestData=countBadlyClassifiedTestData+1;
      end
    end
```

```
end
```

FUNKCJA DO WYLICZANIA QUADPROG W ZADANIU DUALNYM

```
type dualQuadprog.m
```

```
function [ ratios, functionDuration ] = dualQuadprog( points, y )
    C = 1;
    N = length(y);
    K = points*points';
    H = 2*diag(y)*K*diag(y);
    f = -ones(N,1);
    Aeq = y';
    beq = [0];
    LB = zeros(N,1);
    UB = C*ones(N,1);

    tic
    ratios = quadprog(H, f, [], [], Aeq, beq, LB, UB, []);
    functionDuration=toc;
end
```

FUNKCJA DO WYLICZANIA SOLVE W ZADANIU DUALNYM

```
type dualSolve.m
```

```
function [cRatios,functionDuration] = dualSolve(points,y)
    amountOfPoints=length(points);

    flow = optimvar('c',amountOfPoints);

    obj=solveOptimizeThis(flow,points,y);
    p = optimproblem('Objective', obj,'ObjectiveSense','maximize');

    p.Constraints.constr1=optimconstr(amountOfPoints);
    p.Constraints.constr2=optimconstr(amountOfPoints);

    for k=1:amountOfPoints
        p.Constraints.constr1(k)=flow(k)>=0;
        p.Constraints.constr2(k)=flow(k)<=1/(2*amountOfPoints);
    end
    p.Constraints.equality=y'*flow==0;

    x0.c=zeros(amountOfPoints,1);

    tic
    sol= solve(p,x0);
    functionDuration =toc;
    cRatios=sol.c;

end
```

FUNKCJA DO WYLICZANIA FMICON W ZADANIU PRYMALNYM

```
type primalFmincon.m
```

```
function [ratios,functionDuration] = primalFmincon(pointsAbove,pointsBelow,planeDimension)
    amountOfPointsAbove=length(pointsAbove);
    amountOfPointsBelow=length(pointsBelow);
    amountOfPoints=amountOfPointsAbove+amountOfPointsBelow;

    A=zeros(amountOfPoints,planeDimension+1);
    b=zeros(amountOfPoints,1);
    for k=1:amountOfPointsAbove
```

```
        A(k,:)=[pointsAbove(k,:),1];
        b(k)=-1;
    end
    for k=1:(amountOfPointsBelow)
        A(k+amountOfPointsAbove,:)=-[pointsBelow(k,:),1];
        b(k+amountOfPointsAbove)=-1;
    end

    % HARD MARGIN
    fun= @(x)sqrt(sum(x(1:end-1).^2));

    x0=zeros(1,planeDimension+1);
    tic
    % I DO NOT KNOW WHY BUT FOR VALIDATION SIGN MATTERS
    ratios = -fmincon(fun,x0,A,b);
    functionDuration=toc;
end
```

FUNKCJA DO WYLICZANIA SOLVE W ZADANIU PRYMALNYM

```
type primalSolve.m
```

```
function [ratios,functionDuration] = primalSolve(pointsAbove,pointsBelow,planeDimension)
    amountOfPointsAbove=length(pointsAbove);
    amountOfPointsBelow=length(pointsBelow);

    flow = optimvar('W',planeDimension+1);

    % HARD MARGIN
    fun= @(x)sqrt(sum(x(1:end-1).^2));

    obj =fun(flow);

    problem = optimproblem('Objective', obj,'ObjectiveSense','minimize');

    problem.Constraints=optimconstr(amountOfPointsAbove+amountOfPointsBelow);
    for k=1:amountOfPointsAbove
        problem.Constraints(k) = (pointsAbove(k,:)*flow(1:end-1)+flow(end))>=1;
    end
    for k=1:amountOfPointsBelow
        problem.Constraints(k+amountOfPointsAbove) = -(pointsBelow(k,:)*flow(1:end-1)+flow(end))>=1;
    end

    x0.W=ones(1,planeDimension+1);

    tic;
    sol= solve(problem,x0);
    functionDuration=toc;

    ratios=sol.W';
end
```

FUNKCJA DO WYLICZANIA WSPÓŁCZYNNIKÓW W ZADANIU DUALNYM

```
type getDualRatios.m
```

```
function [ratios] = getDualRatios(pointsAbove,pointsBelow,cRatios)
    amountOfPointsAbove=length(pointsAbove);
    amountOfPointsBelow=length(pointsBelow);

    adding=cRatios(1:amountOfPointsAbove);
    substracting =cRatios(amountOfPointsAbove+1:end);

    W=adding'*pointsAbove-substracting'*pointsBelow;
```

```
    bUp=zeros(amountOfPointsAbove,1);
    for k=1: amountOfPointsAbove
        bUp(k)=-(W*pointsAbove(k,:)')+1;
    end

    bDown=zeros(amountOfPointsBelow,1);
    for k=1: amountOfPointsBelow
        bDown(k)=-(W*pointsBelow(k,:)')-1;
    end

    b=(min(bUp)+max(bDown))/2;
    ratios=[W,b];
end
```

PŁASZCZYZNA W 3D

ZADANIE PRYMALNE

```
type myPrimal3D.m
```

```
clear all;

% DECLARE PLANE
%  Ax+By+Cz+D=0
A=-10;
B=10;
C=20;
D=30;
baseRatios=[A,B,C,D];
planeDimension=length(baseRatios)-1;

amountOfPointsAbove =100;
amountOfPointsBelow =100;

%  GENERATE DATA
[pointsAbove,pointsBelow, pointsLearn,pointsAboveTest,pointsBelowTest,pointsTest,y] = generateData(amountOfPointsAbo

% FIND PLANE FACTORS
[ratiosFmincon,functionDurationFmincon] = primalFmincon(pointsAbove,pointsBelow,planeDimension);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM FMINCON
[ratiosSolve,functionDurationSolve] = primalSolve(pointsAbove,pointsBelow,planeDimension);

% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataFmincon,countBadlyClassifiedTestDataFmincon] = validateResults(pointsLearn,y,pointsTes
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,y,pointsTest,y,

% DRAW DATA
chartTitle="ZADANIE PRYMALNE, DANE UCZĄCE" +newline + "FMINCON";
drawSurface(chartTitle,functionDurationFmincon,ratiosFmincon,baseRatios,pointsAbove,pointsBelow,countBadlyClassified

chartTitle="ZADANIE PRYMALNE, DANE TESTOWE" +newline + "FMINCON";
drawSurface(chartTitle,functionDurationFmincon,ratiosFmincon,baseRatios,pointsAboveTest,pointsBelowTest,countBadlyCl


chartTitle="ZADANIE PRYMALNE, DANE UCZĄCE" +newline + "SOLVE";
drawSurface(chartTitle,functionDurationSolve,ratiosSolve,baseRatios,pointsAbove,pointsBelow,countBadlyClassifiedLear

chartTitle="ZADANIE PRYMALNE, DANE TESTOWE" +newline + "SOLVE";
drawSurface(chartTitle,functionDurationSolve,ratiosSolve,baseRatios,pointsAboveTest,pointsBelowTest,countBadlyClassi
```

```
myPrimal3D;
```

```
Local minimum found that satisfies the constraints.
```

```
Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Solving problem using fmincon.

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

ZADANIE DUALNE

```
type myDual3D.m
```

```
clear all;

% DECLARE PLANE
A=-10;
B=10;
C=20;
D=30;

baseRatios=[A,B,C,D];

amountOfPointsAbove =100;
amountOfPointsBelow =100;

amountOfPoints=amountOfPointsAbove+amountOfPointsBelow;

% GENERATE DATA
[pointsAbove,pointsBelow, pointsLearn,pointsAboveTest,pointsBelowTest,pointsTest,y] = generateData(amountOfPointsAbo

% FIND PLANE FACTORS WITHOUT b
[cRatiosSolve,functionDurationSolve] = dualSolve(pointsLearn,y);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM FMINCON
[cRatiosQuadprog, functionDurationQuadprog]=dualQuadprog(pointsLearn,y);

% ADD b TO PLANE FACTORS
ratiosQuadprog=getDualRatios(pointsAbove,pointsBelow,cRatiosQuadprog);
ratiosSolve=getDualRatios(pointsAbove,pointsBelow,cRatiosSolve);

% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataQuadprog,countBadlyClassifiedTestDataQuadprog] = validateResults(pointsLearn,y,pointsT
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,y,pointsTest,y,

% DRAW DATA
chartTitle="ZADANIE DUALNE, DANE UCZĄCE" +newline + "QUADPROG";
drawSurface(chartTitle,functionDurationQuadprog,ratiosQuadprog,baseRatios,pointsAbove,pointsBelow,countBadlyClassifi

chartTitle="ZADANIE DUALNE, DANE TESTOWE" +newline + "QUADPROG";
drawSurface(chartTitle,functionDurationQuadprog,ratiosQuadprog,baseRatios,pointsAboveTest,pointsBelowTest,countBadly
```

```
chartTitle="ZADANIE DUALNE, DANE UCZĄCE" +newline + "SOLVE";
drawSurface(chartTitle,functionDurationSolve,ratiosSolve,baseRatios,pointsAbove,pointsBelow,countBadlyClassifiedLear

chartTitle="ZADANIE DUALNE, DANE TESTOWE" +newline + "SOLVE";
drawSurface(chartTitle,functionDurationSolve,ratiosSolve,baseRatios,pointsAboveTest,pointsBelowTest,countBadlyClassi
```

```
myDual3D;
```

```
Solving problem using quadprog.
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

PRZESTRZEŃ PIĘCIO WYMIAROWA

ZADANIE PRYMALNE

```
type myPrimal.m
```

```
clear all;

% DECLARE PLANE
baseRatios=[10,15,20,25,30,50];
planeDimension=length(baseRatios)-1;

amountOfPointsAbove =100;
amountOfPointsBelow =100;
amountOfPoints=amountOfPointsAbove+amountOfPointsBelow;

% GENERATE DATA
[pointsAbove,pointsBelow, pointsLearn,pointsAboveTest,pointsBelowTest,pointsTest,y] = generateData(amountOfPointsAbo

% FIND PLANE FACTORS
[ratiosFmincon,functionDurationFmincon] = primalFmincon(pointsAbove,pointsBelow,planeDimension);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM FMINCON
[ratiosSolve,functionDurationSolve] = primalSolve(pointsAbove,pointsBelow,planeDimension);


% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataFmincon,countBadlyClassifiedTestDataFmincon] = validateResults(pointsLearn,y,pointsTes
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,y,pointsTest,y,

% DISPLAY RESULTS
```

```
disp("ZADANIE PRYMALNE DLA PRZESTRZENI 5 WYMIAROWEJ"+newline)
disp("ZADANA PŁASZCZYZNA: "+strjoin(string(baseRatios), ', '))
disp("LICZBA PUNKTÓW (UCZĄCYCH/TESTOWYCH): "+ amountOfPoints);

disp(newline)

disp("ZADANIE PRYMALNE 5 WYMIARY, FMINCON");
disp("CZAS: " + functionDurationFmincon);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosFmincon), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataFmincon);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataFmincon);

disp(newline)

disp("ZADANIE PRYMALNE 4 WYMIARY, SOLVE");
disp("CZAS: " + functionDurationSolve);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosSolve), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataSolve);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataSolve);
```

```
myPrimal;
```

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Solving problem using fmincon.

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
ZADANIE PRYMALNE DLA PRZESTRZENI 5 WYMIAROWEJ

ZADANA PŁASZCZYZNA: 10, 15, 20, 25, 30, 50
LICZBA PUNKTÓW (UCZĄCYCH/TESTOWYCH): 200


ZADANIE PRYMALNE 5 WYMIARY, FMINCON
CZAS: 0.43207
WYZNACZONA PŁASZCZYZNA: 0.13294, 0.2126, 0.27925, 0.33923, 0.42002, 0.35661
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 0
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 3


ZADANIE PRYMALNE 4 WYMIARY, SOLVE
CZAS: 0.65268
WYZNACZONA PŁASZCZYZNA: 0.13294, 0.2126, 0.27925, 0.33923, 0.42002, 0.35661
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 0
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 3
```

ZADANIE DUALNE

```
type myDual.m
```

```
clear all;
```

```
% DECLARE PLANE
baseRatios=[10,15,20,25,30,50];
planeDimension=length(baseRatios)-1;

amountOfPointsAbove =100;
amountOfPointsBelow =100;
amountOfPoints=amountOfPointsAbove+amountOfPointsBelow;

% GENERATE DATA
[pointsAbove,pointsBelow, pointsLearn,pointsAboveTest,pointsBelowTest,pointsTest,y] = generateData(amountOfPointsAbo

% FIND PLANE FACTORS WITHOUT b
[cRatiosSolve,functionDurationSolve] = dualSolve(pointsLearn,y);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM FMINCON
[cRatiosQuadprog, functionDurationQuadprog]=dualQuadprog(pointsLearn,y);

% ADD b TO PLANE FACTORS
ratiosQuadprog=getDualRatios(pointsAbove,pointsBelow,cRatiosQuadprog);
ratiosSolve=getDualRatios(pointsAbove,pointsBelow,cRatiosSolve);

% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataQuadprog,countBadlyClassifiedTestDataQuadprog] = validateResults(pointsLearn,y,pointsT
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,y,pointsTest,y,



% DISPLAY RESULTS
disp("ZADANIE DUALNE DLA PRZESTRZENI 5 WYMIAROWEJ")
disp("ZADANA PŁASZCZYZNA: "+strjoin(string(baseRatios), ', '))
disp("LICZBA PUNKTÓW (UCZĄCYCH/TESTOWYCH): "+ amountOfPoints);

disp(newline);

disp("ZADANIE DUALNE 5 WYMIARÓW, QUADPROG");
disp("CZAS: " + functionDurationQuadprog);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosQuadprog), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataQuadprog);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataQuadprog);

disp(newline)

disp("ZADANIE DUALNE 5 WYMIARÓW, SOLVE");
disp("CZAS: " + functionDurationSolve);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosSolve), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataSolve);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataSolve);
```

```
myDual;
```

```
Solving problem using quadprog.
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
```

```
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
ZADANIE DUALNE DLA PRZESTRZENI 5 WYMIAROWEJ
ZADANA PŁASZCZYZNA: 10, 15, 20, 25, 30, 50
LICZBA PUNKTÓW (UCZĄCYCH/TESTOWYCH): 200


ZADANIE DUALNE 5 WYMIARÓW, QUADPROG
CZAS: 0.038172
WYZNACZONA PŁASZCZYZNA: 0.069315, 0.10408, 0.14262, 0.17397, 0.20203, 0.072714
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 6
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 4


ZADANIE DUALNE 5 WYMIARÓW, SOLVE
CZAS: 0.89307
WYZNACZONA PŁASZCZYZNA: 0.030662, 0.053672, 0.067592, 0.086316, 0.10236, 0.15045
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 2
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 1
```

DANE WBDC

ZADANIE PRYMALNE

type wdbcPrimal.m

```
clear all;

planeDimension=30;

% READ DATA
dataLearn=readtable('wdbcLearn.dat');
pointsAboveLearn=table2array(dataLearn(strcmpi(dataLearn.Var31,{'B'})>0,1:planeDimension));
pointsBelowLearn=table2array(dataLearn(strcmpi(dataLearn.Var31,{'M'})>0,1:planeDimension));
pointsLearn=[pointsAboveLearn;pointsBelowLearn];
amountPointsAboveLearn=length(pointsAboveLearn);
amountPointsBelowLearn=length(pointsBelowLearn);
amountOfPointsLearn=amountPointsAboveLearn+amountPointsBelowLearn;
yLearn=[ones(amountPointsAboveLearn,1);-ones(amountPointsBelowLearn,1)];

dataTest=readtable('wdbcTest.dat');
pointsAboveTest=table2array(dataTest(strcmpi(dataTest.Var31,{'B'})>0,1:planeDimension));
pointsBelowTest=table2array(dataTest(strcmpi(dataTest.Var31,{'M'})>0,1:planeDimension));
pointsTest=[pointsAboveTest;pointsBelowTest];
amountPointsAboveTest=length(pointsAboveTest);
amountPointsBelowTest=length(pointsBelowTest);
amountOfPointsTest=amountPointsAboveTest+amountPointsBelowTest;
yTest=[ones(amountPointsAboveTest,1);-ones(amountPointsBelowTest,1)];


% FIND PLANE FACTORS
[ratiosFmincon,functionDurationFmincon] = primalFmincon(pointsAboveLearn,pointsBelowLearn,planeDimension);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM FMINCON
[ratiosSolve,functionDurationSolve] = primalSolve(pointsAboveLearn,pointsBelowLearn,planeDimension);


% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataFmincon,countBadlyClassifiedTestDataFmincon] = validateResults(pointsLearn,yLearn,poin
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,yLearn,pointsTe


disp("ZADANIE PRYMALNE, WDBC, FMINCON");
disp("CZAS: " + functionDurationFmincon);
```

```
disp("LICZBA PUNKTÓW UCZĄCYCH: "+ amountOfPointsLearn);
disp("LICZBA PUNKTÓW TESTOWYCH: "+ amountOfPointsTest);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosFmincon), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataFmincon);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataFmincon);

disp(newline)

disp("ZADANIE PRYMALNE, WDBC, SOLVE");
disp("CZAS: " + functionDurationSolve);
disp("LICZBA PUNKTÓW UCZĄCYCH: "+ amountOfPointsLearn);
disp("LICZBA PUNKTÓW TESTOWYCH: "+ amountOfPointsTest);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosSolve), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataSolve);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataSolve);
```

```
wdbcPrimal;
```

```
Solver stopped prematurely.

fmincon stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 3.000000e+03.


Solving problem using fmincon.

Solver stopped prematurely.

fmincon stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 3.000000e+03.

ZADANIE PRYMALNE, WDBC, FMINCON
CZAS: 0.91589
LICZBA PUNKTÓW UCZĄCYCH: 350
LICZBA PUNKTÓW TESTOWYCH: 219
WYZNACZONA PŁASZCZYZNA: 85.93955, -0.5857919, -5.560106, -0.4862278, -393.7781, 167.4099, 161.8143, -249.7381, 222.8
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 0
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 14


ZADANIE PRYMALNE, WDBC, SOLVE
CZAS: 1.1372
LICZBA PUNKTÓW UCZĄCYCH: 350
LICZBA PUNKTÓW TESTOWYCH: 219
WYZNACZONA PŁASZCZYZNA: 85.94734, -0.5861032, -5.561226, -0.4862246, -393.7561, 167.3695, 161.9153, -249.801, 222.91
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 0
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 14
```

ZADANIE DUALNE

```
type wdbcDual.m
```

```
clear all;

planeDimension=30;

% READ DATA
dataLearn=readtable('wdbcLearn.dat');
pointsAboveLearn=table2array(dataLearn(strcmpi(dataLearn.Var31,{'B'})>0,1:planeDimension));
pointsBelowLearn=table2array(dataLearn(strcmpi(dataLearn.Var31,{'M'})>0,1:planeDimension));
pointsLearn=[pointsAboveLearn;pointsBelowLearn];
amountPointsAboveLearn=length(pointsAboveLearn);
amountPointsBelowLearn=length(pointsBelowLearn);
amountOfPointsLearn=amountPointsAboveLearn+amountPointsBelowLearn;
```

```
yLearn=[ones(amountPointsAboveLearn,1);-ones(amountPointsBelowLearn,1)];

dataTest=readtable('wdbcTest.dat');
pointsAboveTest=table2array(dataTest(strcmpi(dataTest.Var31,{'B'})>0,1:planeDimension));
pointsBelowTest=table2array(dataTest(strcmpi(dataTest.Var31,{'M'})>0,1:planeDimension));
pointsTest=[pointsAboveTest;pointsBelowTest];
amountPointsAboveTest=length(pointsAboveTest);
amountPointsBelowTest=length(pointsBelowTest);
amountOfPointsTest=amountPointsAboveTest+amountPointsBelowTest;
yTest=[ones(amountPointsAboveTest,1);-ones(amountPointsBelowTest,1)];


% FIND PLANE FACTORS
[cRatiosQuadprog,functionDurationQuadprog] = dualQuadprog(pointsLearn,yLearn);
clearAllMemoizedCaches; % CLEARING CACHES, BECAUSE SOLVE USES CACHE FROM QUADPROG
[cRatiosSolve,functionDurationSolve] = dualSolve(pointsLearn,yLearn);

% ADD b TO PLANE FACTORS
ratiosQuadprog=getDualRatios(pointsAboveLearn,pointsBelowLearn,cRatiosQuadprog);
ratiosSolve=getDualRatios(pointsAboveLearn,pointsBelowLearn,cRatiosSolve);


% COUNT WRONGLY CLASSIFIED DATA
[countBadlyClassifiedLearnDataQuadprog,countBadlyClassifiedTestDataQuadprog] = validateResults(pointsLearn,yLearn,po
[countBadlyClassifiedLearnDataSolve,countBadlyClassifiedTestDataSolve] = validateResults(pointsLearn,yLearn,pointsTe

% DISPLAY RESULTS
disp("ZADANIE DUALNE, WDBC, QUADPROG");
disp("CZAS: " + functionDurationQuadprog);
disp("LICZBA PUNKTÓW UCZĄCYCH: "+ amountOfPointsLearn);
disp("LICZBA PUNKTÓW TESTOWYCH: "+ amountOfPointsTest);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosQuadprog), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataQuadprog);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataQuadprog);

disp(newline)

disp("ZADANIE DUALNE, WDBC, SOLVE");
disp("CZAS: " + functionDurationSolve);
disp("LICZBA PUNKTÓW UCZĄCYCH: "+ amountOfPointsLearn);
disp("LICZBA PUNKTÓW TESTOWYCH: "+ amountOfPointsTest);
disp("WYZNACZONA PŁASZCZYZNA: "+strjoin(string(ratiosSolve), ', '))
disp("BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: " + countBadlyClassifiedLearnDataSolve);
disp("BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: " + countBadlyClassifiedTestDataSolve);
```

```
wdbcDual;
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

Solving problem using quadprog.
The interior-point-convex algorithm does not accept an initial point.
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.
```

```
<stopping criteria details>
ZADANIE DUALNE, WDBC, QUADPROG
CZAS: 0.19331
LICZBA PUNKTÓW UCZĄCYCH: 350
LICZBA PUNKTÓW TESTOWYCH: 219
WYZNACZONA PŁASZCZYZNA: 0.462704, -0.00985287, 0.0534383, -0.000731345, -0.113212, -0.0639754, -0.231818, -0.124825,
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 138
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 46


ZADANIE DUALNE, WDBC, SOLVE
CZAS: 2.776
LICZBA PUNKTÓW UCZĄCYCH: 350
LICZBA PUNKTÓW TESTOWYCH: 219
WYZNACZONA PŁASZCZYZNA: 0.00259475, -0.0567635, -0.0250635, 0.0147148, -0.000592886, -0.00133505, -0.00187284, -0.00
BŁĘDNIE ZAKWALIKOWANE DANE UCZĄCE: 141
BŁĘDNIE ZAKWALIKOWANE DANE TESTOWE: 47
```