Exam 2019

Name : Anton Sakarias Rørbæk Sihm

Student nr: 201504954

Exercise 1

Consider the following optimization problem:

Maximize: $f(x_1, x_2) = 4x_1 + 2x_2 + x_3$

Subject to: $x_1 \leq 1;$ (i)

$4x_1 + 1x_2 \leq 4;$ (ii)

$8x_1 + +4x_2 + 1x_3 \leq 16;$ (iii)

and $x_1 \geq 0; x_2 \geq 0;$

a) Write out the simplex tableaux for the problem and show the first step needed to bring in a new variable into the solution (e.g. argue what column and row to choose, and what elementary operations are needed for the first reductions). Find the maximum for f on the feasible set (you may use MatLab).

```
% Simplex tuable for the problem
% Step change inquality to equality by adding slacks var
%    x1   x2,   x3, s1   s2   s3    M     B
A = [1,    0,   0,   1,   0,   0,   0,   1;
     4,    1,   0,   0,   1,   0,   0,   4;
     8,   4,    1,   0,   0,   1,   0,   16;
     -4,  -2,  -1,  0,   0,   0,    1,   0]
```

```
A =
        1          0          0          1          0          0          0          1
        4          1          0          0          1          0          0          4
        8          4          1          0          0          1          0         16
       -4         -2         -1          0          0          0          1          0
```

```
% Step two, find most negative --> they are equal -- we take x1
% Step three a positive ratio than is smallets
% B/entry  for all entries
% 1/1 = 1  |   4/4 = 1    |    16/8 = 2     --> pivot on R1 or R2 --> we take R1

% R3 - 8R1
A(3,:) = A(3,:) - 8*A(1,:);

% R2 - 4R1
A(2,:) = A(2,:) - 4*A(1,:);

% R4 + 4R1
```

```
A(4,:) = A(4,:) + 4*A(1,:)
```

```
A =
     1     0     0     1     0     0     0     1
     0     1     0    -4     1     0     0     0
     0     4     1    -8     0     1     0     8
     0    -2    -1     4     0     0     1     4
```

```
% This is the first iteration
% ion of the simplix algorithm.
% lets finish the rest computationally
x1 = optimvar('x1');
x2 = optimvar('x2');
x3 = optimvar('x3');

prob = optimproblem('Objective', 4*x1 +2*x2 + x3,'ObjectiveSense','max');
prob.Constraints.c1 = x1  <= 1;
prob.Constraints.c2 = 4*x1 + x2 <= 4;
prob.Constraints.c3 = 8*x1 + 4*x2 + x3 <= 16;
prob.Constraints.c4 = x1 >= 0;
prob.Constraints.c5 = x2 >= 0;

problem = prob2struct(prob);
[sol,fval,exitflag,output] = linprog(problem);
```

```
Optimal solution found.
```

```
max_value = 4*0 + 2*0 + 16;
fprintf("Our maxpoint is [x1, x2, x3]  = [ %.2f,  %.2f, %.2f] with max value : %.2f ", sol, max
```

```
Our maxpoint is [x1, x2, x3]  = [ 0.00,  0.00, 16.00] with max value : 16.00
```

b) Write out the dual problem for the optimization problem above and find the solution of this. Explain how to find the solution to the dual problem, and explain how to interpret the solution.

```
% Step change inquality to equality by adding slacks var
%    x1   x2,   x3, s1   s2   s3    M     B
A = [1,    0,   0,   1,   0,   0,   0,   1;
     4,    1,   0,   0,   1,   0,   0,   4;
     8,   4,    1,   0,   0,   1,   0,   16;
     -4,  -2,  -1,  0,   0,   0,    1,   0];

A=[1 0 0 1 0 0;
   4 1 0 0 1 0;
   8 4 1 0 0 1];
b=[1;4;16];
c=-[4;2;1;0;0;0];
v=[4;5;6];

simplex(c,A,b,v,1)
```

```
Initial tableau:
     1     0     0     1     0     0     1
```

```
      4          1          0          0          1          0          4
      8          4          1          0          0          1         16
     -4         -2         -1          0          0          0          0

Pivot point:
      1          1

New tableau:
      1          0          0          1          0          0          1
      0          1          0         -4          1          0          0
      0          4          1         -8          0          1          8
      0         -2         -1          4          0          0          4

Pivot point:
      2          2

New tableau:
      1          0          0          1          0          0          1
      0          1          0         -4          1          0          0
      0          0          1          8         -4          1          8
      0          0         -1         -4          2          0          4

Pivot point:
      1          4

New tableau:
      1          0          0          1          0          0          1
      4          1          0          0          1          0          4
     -8          0          1          0         -4          1          0
      4          0         -1          0          2          0          8

Pivot point:
      3          3

New tableau:
      1          0          0          1          0          0          1
      4          1          0          0          1          0          4
     -8          0          1          0         -4          1          0
     -4          0          0          0         -2          1          8

Pivot point:
      1          1

New tableau:
      1          0          0          1          0          0          1
      0          1          0         -4          1          0          0
      0          0          1          8         -4          1          8
      0          0          0          4         -2          1         12

Pivot point:
      2          5

New tableau:
      1          0          0          1          0          0          1
      0          1          0         -4          1          0          0
      0          4          1         -8          0          1          8
      0          2          0         -4          0          1         12

Pivot point:
      1          4

New tableau:
      1          0          0          1          0          0          1
      4          1          0          0          1          0          4
      8          4          1          0          0          1         16
```

```
       4              2              0              0              0              1             16
ans =
       0
       0
      16
       1
       4
       0
```

```
chosenClass_x4
```

```
chosenClass_x4 =
       1
```

Exercise 2

Consider the function defined $g(x_1, x_2) = x_1^3 - 3x_1 - x_2^2$

a) Find the critical points for the function g.

If we calculate the gradiant and set it equal to zero, we can solve for critical points : $\nabla f(x) = 0$

```
clearvars;
syms x1 x2
g = x1^3 - 3*x1-x2^2;

gradiant_g = gradient(g, [x1, x2]);
gradiant_g_at_0 = gradiant_g == 0
```

gradiant_g_at_0 =

$$\begin{pmatrix} 3\,{x_1}^2 - 3 = 0 \\ -2\,x_2 = 0 \end{pmatrix}$$

```
[x1,x2] = solve([gradiant_g_at_0], [x1, x2])
```

x1 =

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

x2 =

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
point1 = [x1(1), x2(1)], point2 = [x1(2), x2(2)]
```

point1 = $\begin{pmatrix} -1 & 0 \end{pmatrix}$

point2 = $\begin{pmatrix}1 & 0\end{pmatrix}$

b) Classify each critical point as a local maximum, local minimum, or saddl

$$D = D(a,b) = f_{xx}(a,b)\, f_{yy}(a,b) - \left[f_{xy}(a,b)\right]^2$$

We then have the following classifications of the critical point.

1. If $D > 0$ and $f_{xx}(a,b) > 0$ then there is a relative minimum at (a,b).
2. If $D > 0$ and $f_{xx}(a,b) < 0$ then there is a relative maximum at (a,b).
3. If $D < 0$ then the point (a,b) is a saddle point.
4. If $D = 0$ then the point (a,b) may be a relative minimum, relative maximum or a saddle point. Other techniques would need to be used to classify the critic

```
clear x1 x2
syms x1 x2

hessian_g = hessian(g,[x1,x2])
```

hessian_g =

$$\begin{pmatrix}6\,x_1 & 0\\ 0 & -2\end{pmatrix}$$

```
D = hessian_g(1,1)*hessian_g(2,2)-(hessian_g(2,1))^2
```

D = $-12\,x_1$

```
fxx = hessian_g(1,1)
```

fxx = $6\,x_1$

```
% point1
x1 = point1(1);
x2 = point1(2);

d_p1 = subs(D)
```

d_p1 = 12

```
fxx_p1 = subs(fxx)
```

fxx_p1 = -6

```
fprintf("Since D > 0 : %.3f > 0 and fxx < 0 : fxx < %.3f then point (%.3f, %.3f) is a local MA
```

```
Since D > 0 : 12.000 > 0 and fxx < 0 : fxx < -6.000 then point (-1.000, 0.000) is a local MAX
```

```
% point2
x1 = point2(1);
x2 = point2(2);
d_p2 = subs(D)
```

```
d_p2 = 
```
-12

```
fxx_p2 = subs(fxx)
```

```
fxx_p2 = 
```
6

```
fprintf("Since D < 0 : %.3f < 0 then point (%.3f, %.3f) is a Saddle Point",d_p2, point2(1), poi
```

```
Since D < 0 : -12.000 < 0 then point (1.000, 0.000) is a Saddle Point
```

Now let $f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 1)^2$ and $h(x_1, x_2) = (x_1 - 1)^2 + (x_2 + 2)^2 - 1.$

c) Find the minimum and maximum of f subject to the constraint $h(x_1, x_2) = 0$ (argue for your calculations).

We will use the Lagrange multiplier for the problem

$\nabla f(x) + \lambda * \nabla h(x) = 0$

We can solve the follwing problem by solving 3 linequeations with 3 uknowns

```
clearvars
syms x1 x2
f = (x1-2)^2 + (x2-1)^2;
gradiant_f = gradient(f, [x1,x2]);

clear x1 x2
syms x1 x2 lambda
h = (x1 - 1)^2 + (x2 + 2)^2 - 1
```

```
h = 
```
$(x_1 - 1)^2 + (x_2 + 2)^2 - 1$

```
gradiant_h = gradient(h, [x1,x2])
```

```
gradiant_h = 
```
$$\begin{pmatrix} 2x_1 - 2 \\ 2x_2 + 4 \end{pmatrix}$$

```
% Lagrange multiplier
Lagrange_multiplier = gradiant_f + lambda*gradiant_h == 0
```

```
Lagrange_multiplier = 
```
$$\begin{pmatrix} 2x_1 + \lambda\,(2x_1 - 2) - 4 = 0 \\ 2x_2 + \lambda\,(2x_2 + 4) - 2 = 0 \end{pmatrix}$$

```
% we now have 2 equation and 3 unknowns
% we can use the fuction h = 0 as the last one:
equation1 = Lagrange_multiplier(1)
```

equation1 = $2\,x_1 + \lambda\,(2\,x_1 - 2) - 4 = 0$

```
equation2 = Lagrange_multiplier(2)
```

equation2 = $2\,x_2 + \lambda\,(2\,x_2 + 4) - 2 = 0$

```
equation3 = (x1 - 1)^2 + (x2 + 2)^2 - 1 == 0
```

equation3 = $(x_1 - 1)^2 + (x_2 + 2)^2 - 1 = 0$

```
[x1,x2,lambda] = solve([equation1, equation2, equation3], [x1,x2, lambda]);
x1 = double(x1)
```

```
x1 =
     949/721
     493/721
```

```
x2 = double(x2)
```

```
x2 =
    -758/721
   -2126/721
```

```
lambda = double(lambda)
```

```
lambda =
    3038/1405
    -949/228
```

```
% candidate coordinates
s1 = [x1(1), x2(1)];
s2 = [x1(2), x2(2)];
% Solutions
%(x1-2)^2 + (x2-1)^2;
f1 = (s1(1)-2)^2 + (s1(2)-1)^2
```

```
f1 =
    6569/1405
```

```
f2 = (s2(1)-2)^2 + (s2(2)-1)^2
```

```
f2 =
    1975/114
```

```
fprintf("maximum: %.2f at point : (%.2f, %.2f) and minimum: %.2f at point (%.2f, %.2f)", f2, s2
```

```
maximum: 17.32 at point : (0.68, -2.95) and minimum: 4.68 at point (1.32, -1.05)
```

Exercise 3

Answer the following with **ONE sentence** per question.

a) Are particle swarm optimization methods guaranteed to find the global optimum if the search is run for enough iterations (Yes/No)?

b) In the context of optimization, is the global optimum necessarily unique (Yes/No)?

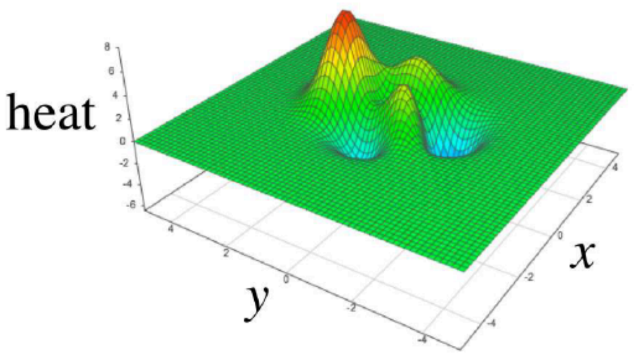
c) In simulated annealing, what are the two conditions that make the search algorithm more likely to accept a **worse** candidate solution?

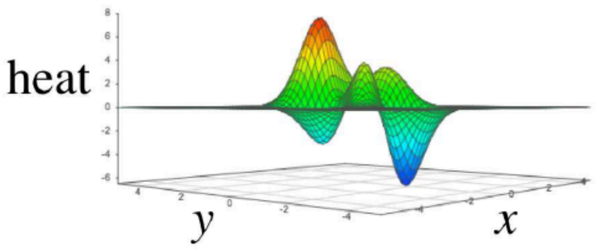
d) What is discrete optimization? Name one example of a well-known discrete optimization problem.

e) What is the main difference between Newton and Quasi-Newton methods?

```
% a)
% No, even tho we run for a long times the algoritm is not 100% sure to converge for a global o

% b)
% No, there can in principible be a global opimum there can be achiceved
% multiple places with the exact same value. Fx max tempurature = 28, can
% maybe be found at diffent places if the temp is exact = 28.

% c)
% Iterations + temparature. If we have been running for at long time, and
% the temp is high, the algorithm is more likly for accept at worse
% candidate, to try and explore new candidate solutions

% d) Discrete optimazations only works on discrete variables. One example could be path finding

% e) The main difference is that the newton method uses the Gradiant, which
% is recalculated each interation. The Quasi-newton uses at aproximation
% which is much faster for higher dimension problems.
```

f) We are helping a cat find the warmest location in the room to have a sleep. Each location in the room is represented by two integer coordinates (*x*,*y*) such that *x* and *y* are each in the interval [-7, 7]. Every location gives a real-value heat score in the range [-8, 8] described by a function *heat*. The distribution of heat across the room according to function *heat* is partially illustrated in the graphs below, e.g. *heat*(2,2) = 8.





We want to use a **genetic algorithm** to find the warmest location in the room, i.e. the (x,y) coordinates that have the highest "heat" score.

i) Develop a representation of a candidate solution location as a chromosome. Explain your representation, and give one example of a chromosome with the corresponding (x,y) coordinate.

```
% Example could be f(1,1) = 3;
%  9-bit chromosome       x   y   t
% convertiong to bits:   001 001 010
```

ii) Define your **objective** function *f* for population size *N* so that you can evaluate each candidate solution. Give an example of *f* using a population of two candidate solutions (x1,y1), (x2,y2) where:
heat(x1,y1) = 4
heat(x2,y2) = -4

```
%TODO
```

Exercise 4

A classification problem is formed by two classes. We are given a set of 2-dimensional data **X** = [$\mathbf{x}_1$ $\mathbf{x}_2$ … $\mathbf{x}_5$]:

$$\boldsymbol{X} = \begin{bmatrix} 0 & 1 & 3 & 3 & 4 \\ 1 & 1 & 1 & 2 & 2 \end{bmatrix},$$

each belonging to one of the two classes, as indicated in the class label vector **l** = [l_1 l_2 … l_5]:

$$l = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 \end{bmatrix}.$$

Using the above (training) vectors and the corresponding class labels, classify the following vectors:

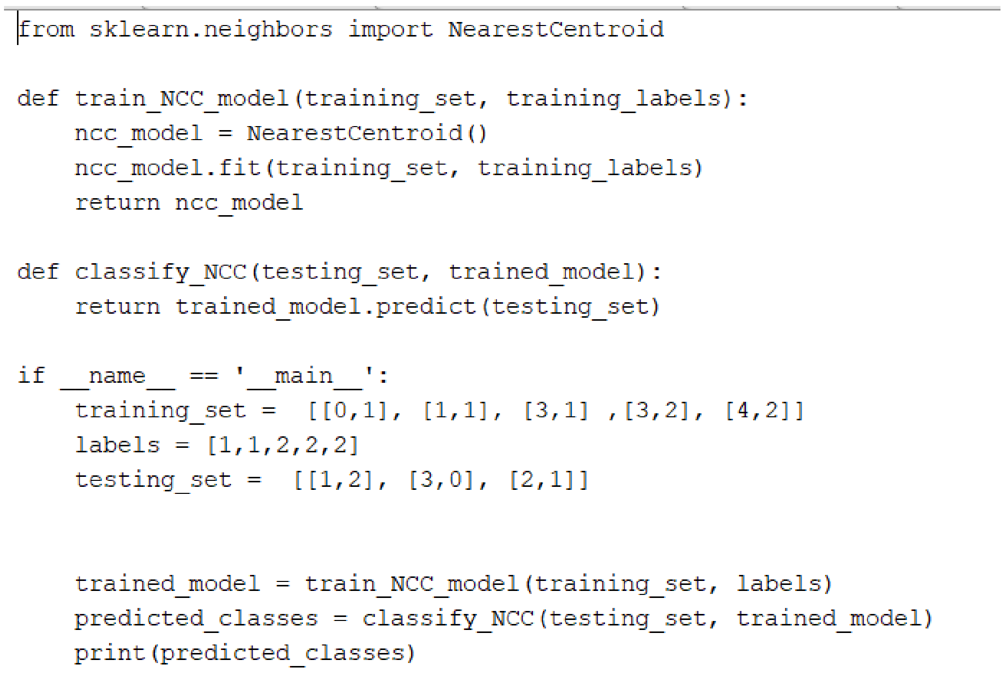
$$\mathbf{X}_6 = [1 \quad 2]^T, \quad \mathbf{x}_7 = [3 \quad 0]^T, \quad \mathbf{x}_8 = [2 \quad 1]^T$$

a) The Nearest Class Centroid (NCC) classifier

```
%For this I will use python.
```

I did the following in python. Th steps are as follows:

```
% Calculate the centroid for each class --> mean each x_i of each class
% Calculate the dist from test sample to each centroid
% Classify by taking the smallest dist to centroid.
% The python code can be seen as follows:
```



```python
from sklearn.neighbors import NearestCentroid

def train_NCC_model(training_set, training_labels):
    ncc_model = NearestCentroid()
    ncc_model.fit(training_set, training_labels)
    return ncc_model

def classify_NCC(testing_set, trained_model):
    return trained_model.predict(testing_set)

if __name__ == '__main__':
    training_set =  [[0,1], [1,1], [3,1] ,[3,2], [4,2]]
    labels = [1,1,2,2,2]
    testing_set =  [[1,2], [3,0], [2,1]]


    trained_model = train_NCC_model(training_set, labels)
    predicted_classes = classify_NCC(testing_set, trained_model)
    print(predicted_classes)
```

```
%  .\Nearest_Centroid.py  yields the prediction for x_i for i = 6, 7, 8.
%  prediction =  [1 2 2]
```

b) The Nearest Neighbor Classifier (using only one neighbor)

```
%Agian for this I will use python.
```

I did the following in python. Th steps are as follows:

```
% Calculate the dist to each class from the testing sample
% Classify by taking asking the k =1 neighbors for the class.
% vote from the set og neigbors, which is this case is only one.
% The python code can be seen as follows:
```

```python
from sklearn import neighbors

def train_NCC_model(training_set, training_labels, number_of_neighbors):
    nnc_model = neighbors.KNeighborsClassifier(number_of_neighbors, weights="uniform")
    nnc_model.fit(training_set, training_labels)
    return nnc_model

def classify_NNC(testing_set, trained_model):
    return trained_model.predict(testing_set)

if __name__ == '__main__':
    training_set =  [[0,1], [1,1], [3,1] ,[3,2], [4,2]]
    labels = [1,1,2,2,2]
    testing_set =  [[1,2], [3,0], [2,1]]
    number_of_neighbors = 1

    trained_model = train_NCC_model(training_set, labels, number_of_neighbors)
    predicted_classes = classify_NNC(testing_set, trained_model)
    print(predicted_classes)
```

```
% .\Nearest_Neighbor.py yields the following prediction for x_i for i = % 6, 7, 8.
% prediction = [1 2 1]
```

c) The Bayes-based classification scheme, where:

$$p(\boldsymbol{x}_i|c_k) = \frac{\|\boldsymbol{x}_i - \boldsymbol{m}_k\|_2^{-2}}{\sum_{m=1}^{K}\|\boldsymbol{x}_i - \boldsymbol{m}_l\|_2^{-2}},$$

where $\boldsymbol{m}_k$ is the class mean vector of class c_k, k=1,2 and $\|v\|_2^{-2}$ = 1 / ($v^\top$ v).

d) Compare (qualitatively) the decision functions obtained by using the NCC classifier and the above Bayes-based classification scheme.

```
% TODO
% TODO
% TODO
```

Exercise 5

The two classes of a binary classification problem are formed by the blue and red samples plotted in Figure 3.

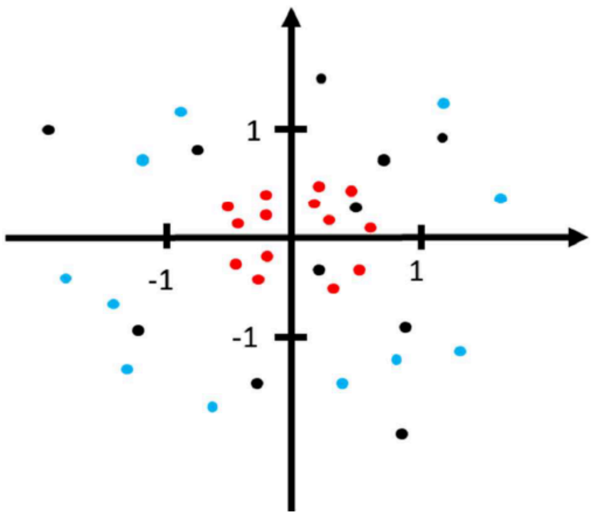
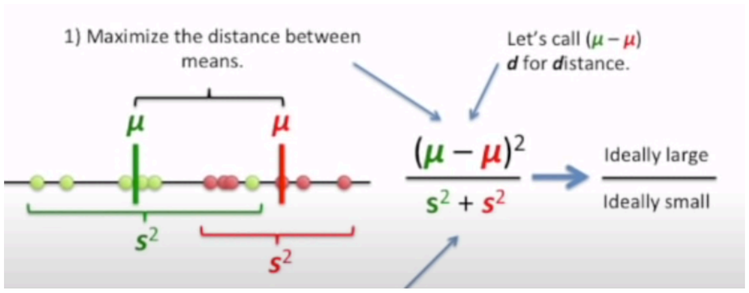


Fig. 3: Training samples of a two-class classification problem

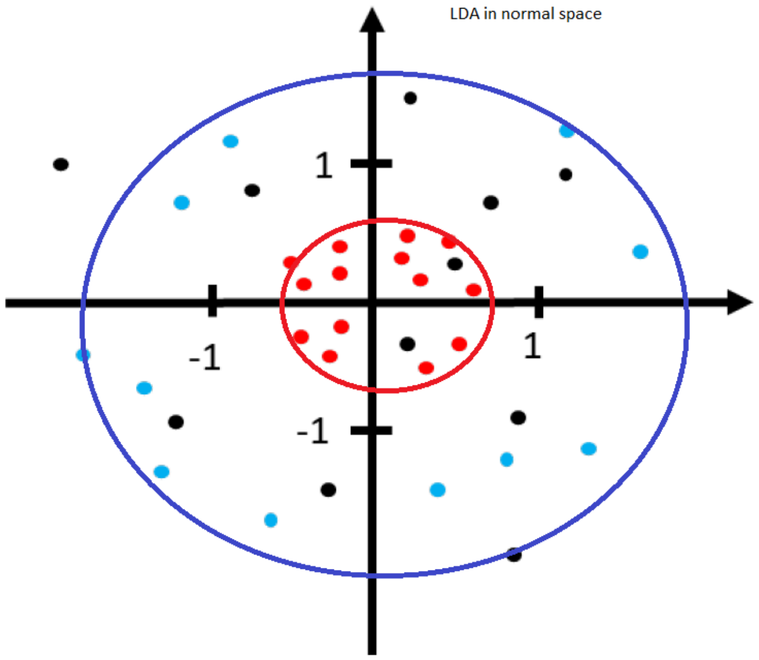
a) Would you use Linear Discriminant Analysis in order to discriminate the two classes in a one-dimensional space? If yes, why. If no, why not?

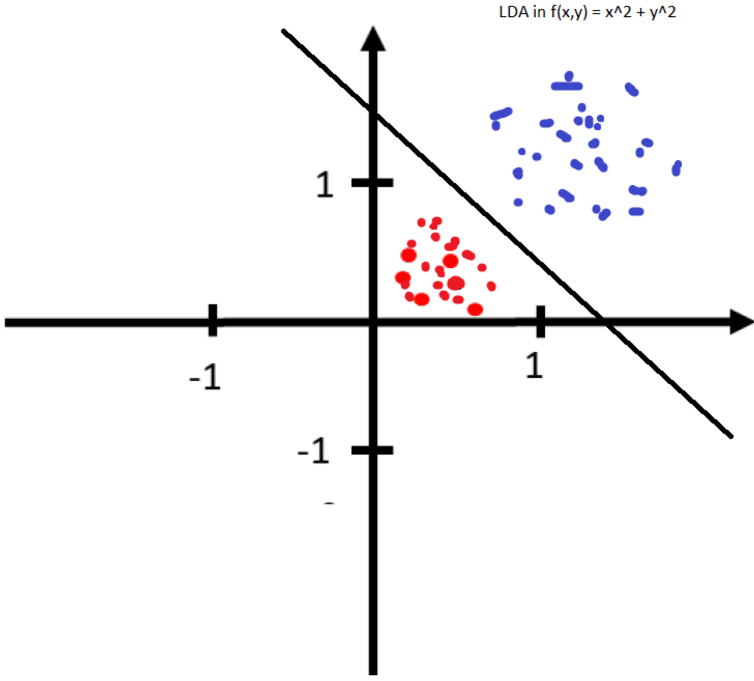

1) Maximize the distance between means.
Let's call $(\mu-\mu)$ *d* for *d*istance.
μ
μ
$\frac{(\mu-\mu)^2}{s^2+s^2}$
Ideally large
Ideally small
s^2
s^2

```
% Since the means for both classes seems very close, I whould not use LDA
% for this, since the distance between the two mean are very small swicth
% generates a bad classification between the classes. We usually want a big
% different between the means and a small scatter for both classes.
% So NO.
```

b) Describe how we can use a Generalized Linear Discriminant Function in order to classify the test samples (plotted as black dots) using a linear classifier. Draw the linear classification function in the original feature space (the one shown in Figure 3) and the space where linear classification is performed.

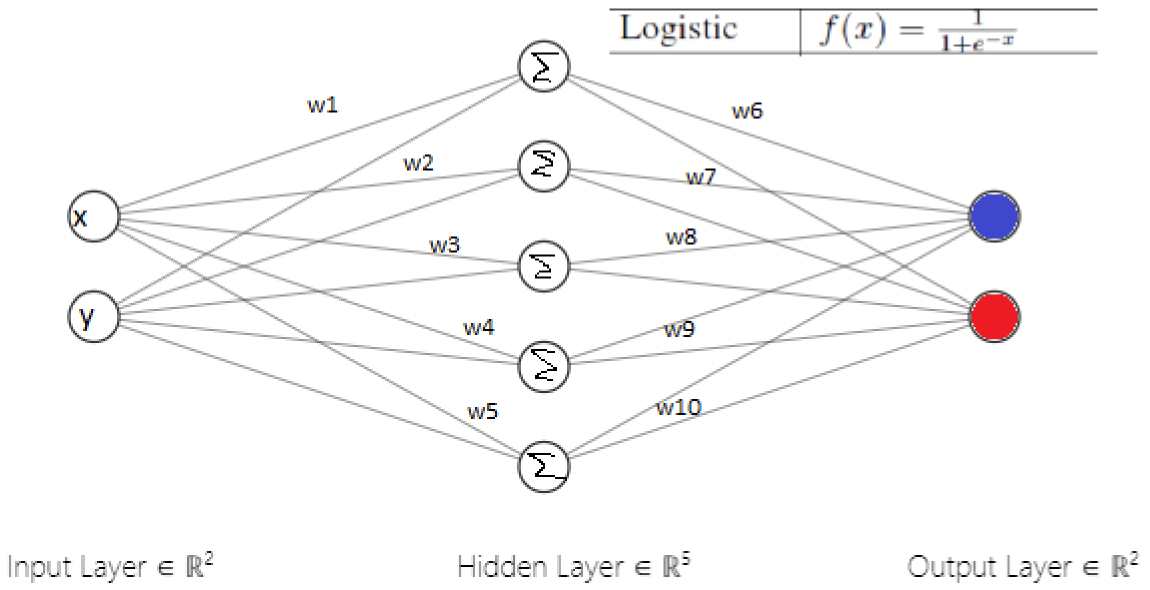
```
% Since the current domain is very bad for LDA, we can transform the data
% into antother domain where the mean is much better:
% f(x,y) --> f(x^2, y^2) makes a good space for the LDA
% f(x,y) --> to polar form  = R < Theta also makes a good space for LDA
% with the R values.7

% Only one circle to seperate the classes !!!!!!!
```





c) Draw a neural network with one hidden layer that can be used for the above classifica
In your drawing include all the parameters and activation functions. Explain (with equ
we cannot use linear activation functions for its neurons in that case.

```
% I ) draw with http://alexlenail.me/NN-SVG/index.html
% We will have 1 hidden layer, and 2 outputs reasoned that we have 2 output
% classes. The neuron size for the hidden layer can be of any set, but it
% heavyly depends on the input problem.
% The input layer is the size of then dimension

% Input should be number of input values   (example Image of 10*10 gives 100 input)
% weigts are wrong.
```



```
% II ) All layers of the neural network collapse into one, because a linear
% combination of linear functions is still a linear function. Not matter
% how many hidden layers we have, the final activaition function wil just
% be a linear function of the first layer.
```

Exercise 6

In a two-class classification problem, the distribution of the class-conditional probabilities $p(x|c_k)$, k=1,2 is given in Figure 1.

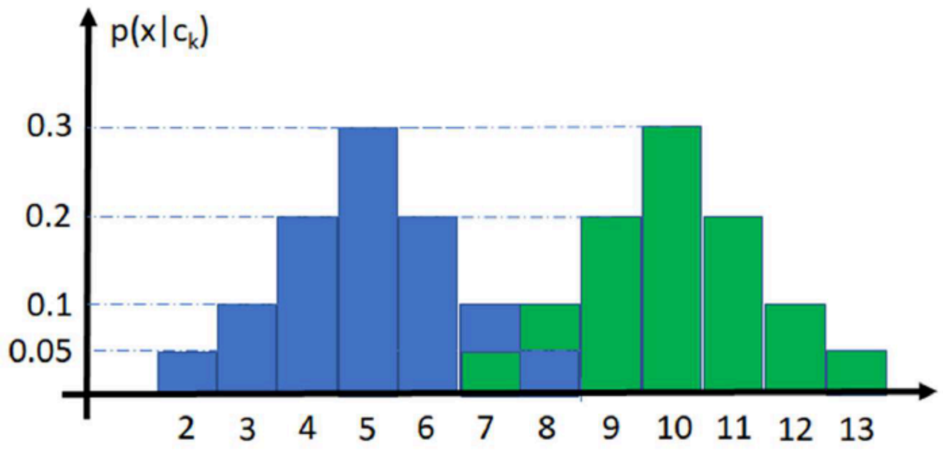


Fig. 1: Class-conditional probabilities of class 1 (blue) and class 2 (green)

The number of samples in class 1 and class 2 is equal to 100 and 200, respectively.

a) Classify (using the trained classifier) the following vectors (test) samples:

$$x_1 = 3, \quad x_2 = 7, \quad x_3 = 8 \text{ and } x_4 = 9$$

We will use the bayes formula to calculate the $P(c_k| x)$ since we alreaddy have $p(x \mid c_k)$ given for every x in the trained model we can use bayers rule, to get $P(c_k| x)$. Note that ine the decision function $P(x)$ gets canceled out:

Decide c_1 if $P(c_1|x) > P(c_2|x)$, otherwise decide c_2

$$P(c_k|x) = \frac{P(x|c_k) * P(c_k)}{P(x)}$$

We will use this decision function to decide the class:

$$P(c_1|x) > P(c_2|x)$$

inserting from bayers we get:

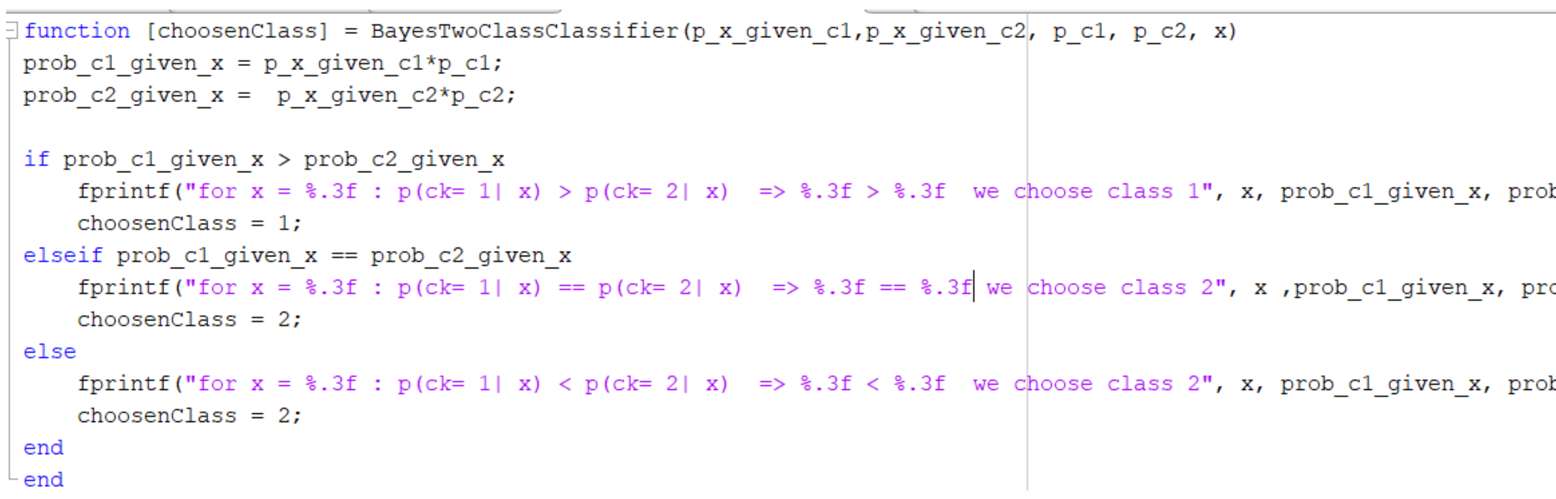
$$\frac{P(x|c_1) * P(c_1)}{P(x)} > \frac{P(x|c_2) * P(c_2)}{P(x)}$$

Multiply by P(X), it cancels out.

$$P(x|c_1) * P(c_1) > P(x|c_2) * P(c_2)$$

```
% For 3=4, x=7, x=8, x=9
p_x_3_given_c1 = 0.1;
p_x_3_given_c2 = 0;

p_x_7_given_c1 = 0.1;
p_x_7_given_c2 = 0.05;

p_x_8_given_c1 = 0.05;
p_x_8_given_c2 = 0.1;

p_x_9_given_c1 = 0;
p_x_9_given_c2 = 0.2;

p_c1 = 100/300;
p_c2 = 200/300;
```

I will use the BayesTwoClassClassifer.m which i implmented for the classification.



```
function [choosenClass] = BayesTwoClassClassifier(p_x_given_c1,p_x_given_c2, p_c1, p_c2, x)
prob_c1_given_x = p_x_given_c1*p_c1;
prob_c2_given_x =  p_x_given_c2*p_c2;

if prob_c1_given_x > prob_c2_given_x
    fprintf("for x = %.3f : p(ck= 1| x) > p(ck= 2| x)  => %.3f > %.3f  we choose class 1", x, prob_c1_given_x, prob
    choosenClass = 1;
elseif prob_c1_given_x == prob_c2_given_x
    fprintf("for x = %.3f : p(ck= 1| x) == p(ck= 2| x)  => %.3f == %.3f we choose class 2", x ,prob_c1_given_x, prc
    choosenClass = 2;
else
    fprintf("for x = %.3f : p(ck= 1| x) < p(ck= 2| x)  => %.3f < %.3f  we choose class 2", x, prob_c1_given_x, prob
    choosenClass = 2;
end
end
```

```
chossenClass_4 = BayesTwoClassClassifier(p_x_3_given_c1,p_x_3_given_c2,p_c1,p_c2,4);
```

```
for x = 4.000 : p(ck= 1| x) > p(ck= 2| x)  => 0.033 > 0.000  we choose class 1
```

```
chossenClass_7 = BayesTwoClassClassifier(p_x_7_given_c1,p_x_7_given_c2,p_c1,p_c2,7);
```

```
for x = 7.000 : p(ck= 1| x) == p(ck= 2| x)  => 0.033 == 0.033 we choose class 2
```

```
chossenClass_8 = BayesTwoClassClassifier(p_x_8_given_c1,p_x_8_given_c2,p_c1,p_c2,8);
```

```
for x = 8.000 : p(ck= 1| x) < p(ck= 2| x)  => 0.017 < 0.067  we choose class 2
```

```
chossenClass_9 = BayesTwoClassClassifier(p_x_9_given_c1,p_x_9_given_c2,p_c1,p_c2,9);
```

```
for x = 9.000 : p(ck= 1| x) < p(ck= 2| x)  => 0.000 < 0.133  we choose class 2
```

b) Consider that the classification risk for the two classes is given by the matrix:

$$\Lambda = \begin{bmatrix} 0 & 0.3 \\ 0.2 & 0 \end{bmatrix}$$

where $\Lambda_{ij} = \lambda((\alpha_i | c_k)$ is the risk of taking action α_i while the correct class is c_k. What is the classification result for the (test) samples x_i, i=1,…,4?

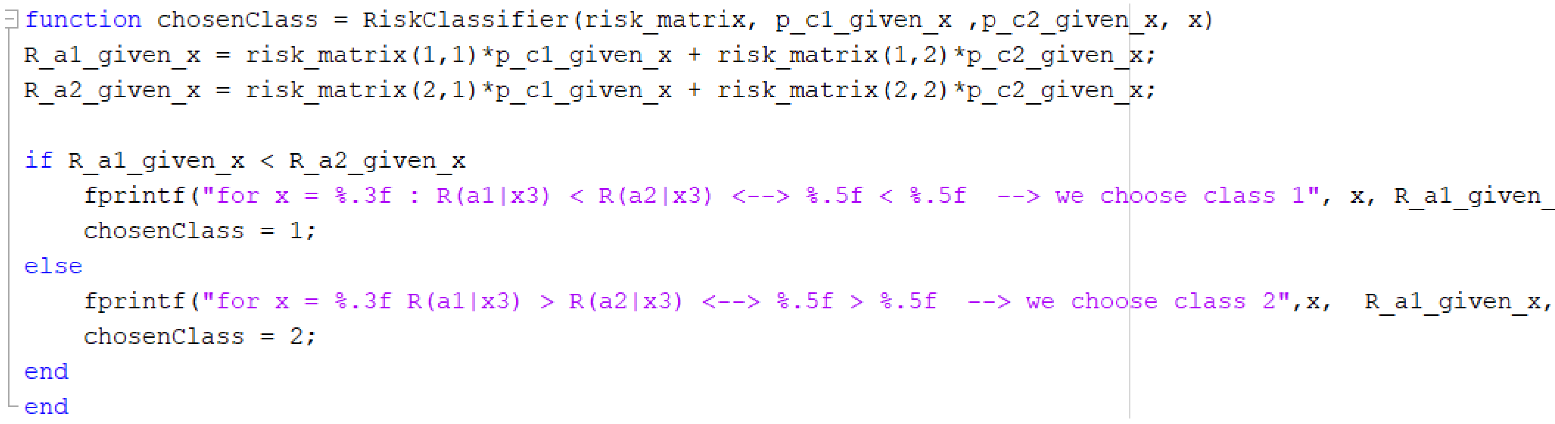
For the two-class case, we have

$$\begin{aligned} R(\alpha_1|\mathbf{x}) &= \lambda(\alpha_1|c_1)P(c_1|\mathbf{x}) + \lambda(\alpha_1|c_2)P(c_2|\mathbf{x}) \\ R(\alpha_2|\mathbf{x}) &= \lambda(\alpha_2|c_1)P(c_1|\mathbf{x}) + \lambda(\alpha_2|c_2)P(c_2|\mathbf{x}) \end{aligned}$$

We classify $\mathbf{x}$ to c_1 if $R(\alpha_1|\mathbf{x}) < R(\alpha_2|\mathbf{x})$, or:

$$(\lambda(\alpha_2|c_1) - \lambda(\alpha_1|c_1))\, P(c_1|\mathbf{x}) > (\lambda(\alpha_1|c_2) - \lambda(\alpha_2|c_2))\, P(c_2|\mathbf{x})$$

We will use the RiskClassifer.m which I implemented for the classification in matlab.



```
function chosenClass = RiskClassifier(risk_matrix, p_c1_given_x ,p_c2_given_x, x)
R_a1_given_x = risk_matrix(1,1)*p_c1_given_x + risk_matrix(1,2)*p_c2_given_x;
R_a2_given_x = risk_matrix(2,1)*p_c1_given_x + risk_matrix(2,2)*p_c2_given_x;

if R_a1_given_x < R_a2_given_x
    fprintf("for x = %.3f : R(a1|x3) < R(a2|x3) <--> %.5f < %.5f  --> we choose class 1", x, R_a1_given_
    chosenClass = 1;
else
    fprintf("for x = %.3f R(a1|x3) > R(a2|x3) <--> %.5f > %.5f  --> we choose class 2",x,  R_a1_given_x,
    chosenClass = 2;
end
end
```

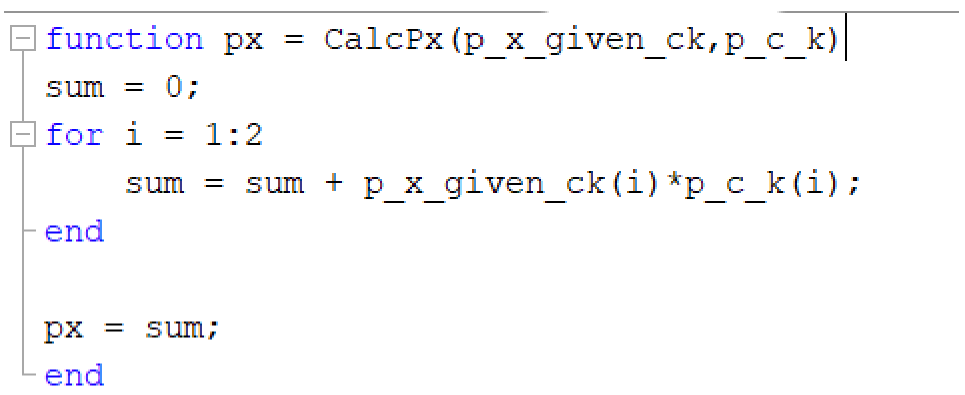
```
% I will assume that i = the i'th risk and for the j -> k-class
risk_matrix = [  0,  0.3;
                0.2,  0  ]
```

```
risk_matrix =
       0             3/10
       1/5           0
```

```
% We need to calculate p(x_i) for i = 1,2,3,4 for the input value of the
% risk classifier
```

$$p(x) = \sum_{k=1}^{K} p(x|c_k) P(c_k)$$

```
% The function for p(x) is implemented in CalcPx.m
```



```
function px = CalcPx(p_x_given_ck,p_c_k)
sum = 0;
for i = 1:2
    sum = sum + p_x_given_ck(i)*p_c_k(i);
end

px = sum;
end
```

```
% for all
p_c_k = [p_c1, p_c2];

% x = 3
p_x3_given_ck = [p_x_3_given_c1, p_x_3_given_c2];
p_x3 = CalcPx(p_x3_given_ck,p_c_k );

% bayes
p_c1_given_x3 = (p_x_3_given_c1*p_c1)/p_x3;
p_c2_given_x3 = (p_x_3_given_c2*p_c2)/p_x3;
%classify
chosenClass_x4 = RiskClassifier(risk_matrix,p_c1_given_x3, p_c2_given_x3, 3);
```

```
for x = 3.000 : R(a1|x3) < R(a2|x3) <--> 0.00000 < 0.20000  --> we choose class 1
```

```
% x = 7
p_x7_given_ck = [p_x_7_given_c1, p_x_7_given_c2];
p_x7 = CalcPx(p_x7_given_ck,p_c_k );

% bayes
p_c1_given_x7 = (p_x_7_given_c1*p_c1)/p_x7;
p_c2_given_x7 = (p_x_7_given_c2*p_c2)/p_x7;
%classify
chosenClass_x7 = RiskClassifier(risk_matrix,p_c1_given_x7,p_c2_given_x7, 7);
```

```
for x = 7.000 R(a1|x3) > R(a2|x3) <--> 0.15000 > 0.10000  --> we choose class 2
```

```
% x = 8
p_x8_given_ck = [p_x_8_given_c1, p_x_8_given_c2];
p_x8 = CalcPx(p_x8_given_ck, p_c_k);

% bayes
p_c1_given_x8 = (p_x_8_given_c1*p_c1)/p_x8;
p_c2_given_x8 = (p_x_8_given_c2*p_c2)/p_x8;
%classify
chosenClass_x8 = RiskClassifier(risk_matrix,p_c1_given_x8,p_c2_given_x8, 8);
```

```
for x = 8.000 R(a1|x3) > R(a2|x3) <--> 0.24000 > 0.04000  --> we choose class 2
```

```
% x = 9
p_x9_given_ck = [p_x_9_given_c1, p_x_9_given_c2];
p_x9 = CalcPx(p_x9_given_ck, p_c_k);
```

```
% bayes
p_c1_given_x9 = (p_x_9_given_c1*p_c1)/p_x9;
p_c2_given_x9 = (p_x_9_given_c2*p_c2)/p_x9;
%classify
chosenClass_x9 = RiskClassifier(risk_matrix,p_c1_given_x9,p_c2_given_x9, 9);
```

```
for x = 9.000 R(a1|x3) > R(a2|x3) <--> 0.30000 > 0.00000  --> we choose class 2
```