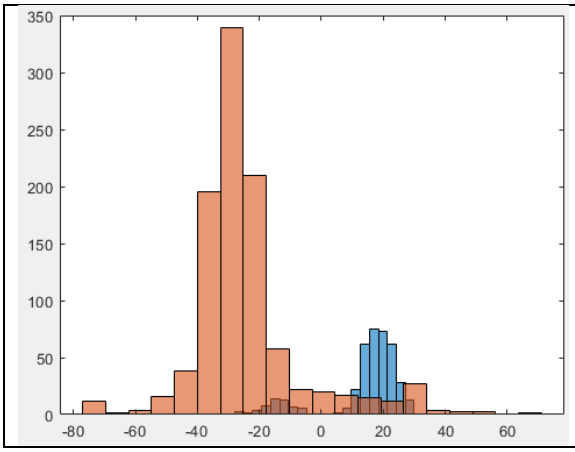
Module-Lab 3: Probabilistic Machine Learning

Exercise 1 (Histograms and Normal distribution)

Similarly as in the Lab#1 and 2, make use of the following files and respective variables *X1.txt, Labels.txt*

Remember that the class/category designated by positives has label "1" (ie, digit = 1 in the variable **Labels**) while label = 0 represents the negative class.

a) We are interested in creating two Histograms, in the same graph, of the data in **X1**. So, plot the histogram for the data corresponding to the positives and another histogram for the negatives, as shown below:



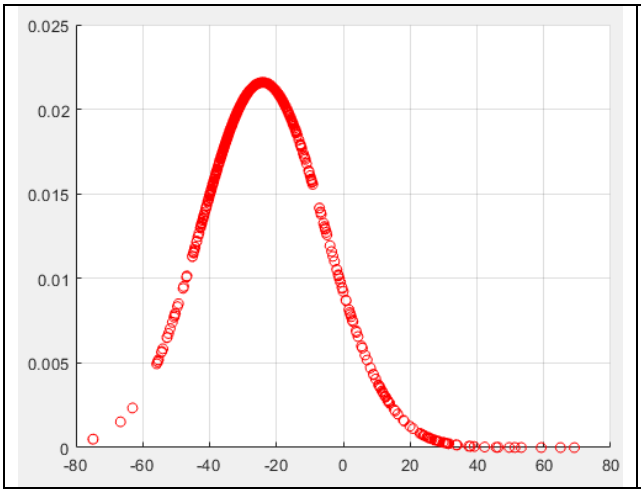

```
%
% Example in Matlab

load("X1.txt")
load("Labels.txt")
npos = 400;
nneg = 1000;

figure(1); cla
nbins = 20;
histogram(X1(1:npos),nbins);
hold on
h = histogram(X1(npos+1:1400),nbins);
```

The histogram of the positives is shown in blue. Notice that these histograms are not normalized.

b) To make it easier to visualize and understand probability densities, we will use only the 'negatives' examples ie, the left-hand side histogram (in orange). From now on, Xneg denotes the negative examples of X1 (ie, Xneg = X1(401:1400). Using Xneg, extract a Normal distribution that fits the data and plot the resulting distribution.

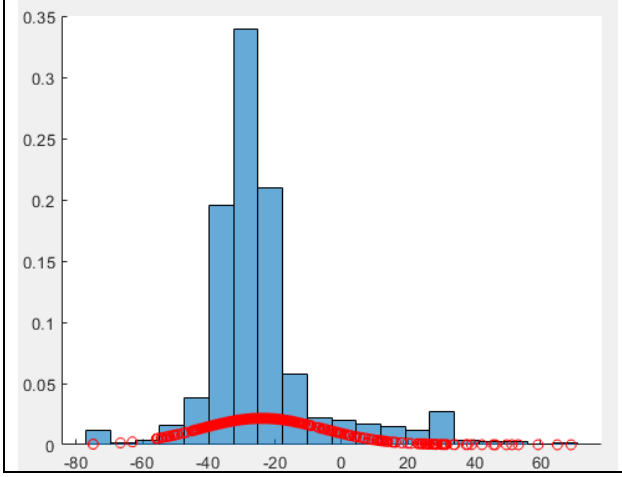



```
%
% Example in Matlab

Xneg = X1(npos+1:1400);
Npdf = fitdist(Xneg,'Normal');
y = pdf(Npdf,Xneg);
plot(Xneg,y,'or');
grid
```

NB: the mean and the standard-deviation (Npdf.mean, Npdf.sigma) can be calculated, in Matlab, by the functions: **mean()** and **std()**.

c) Now, extract a normalized histogram from Xneg and then plot both, the histogram you have got and the Normal distribution in the same graph.




```
%
% Example in Matlab

nbins = 20;
figure(2); cla
H = histogram(Xneg,nbins,
'Normalization','probability');

hold on
plot(Xneg,y,'or'); % from (b)
```

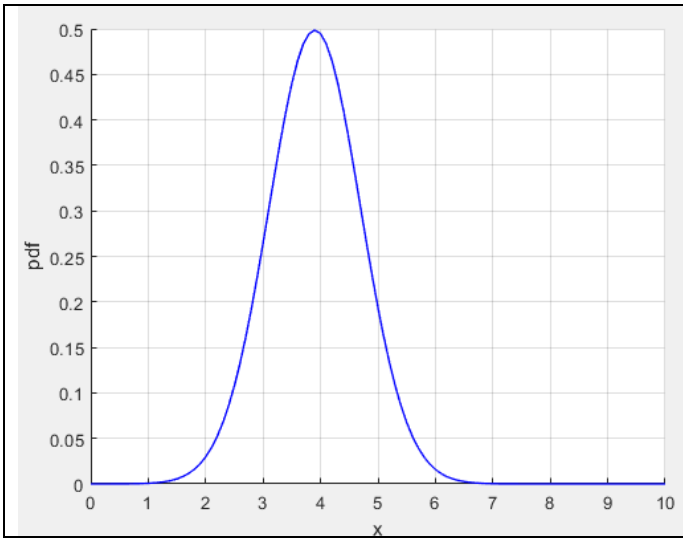
d) Take a look at the histogram's values eg, ***H.Values***. Now, compare it to the values of the standard histogram eg, **h.values**, where `h = histogram(Xneg,nbins);` (exe1.a)
Now, check the h.values when they are normalized

`h.Values/nneg` (where *nneg* = 1000)
they should be the same as in ***H.Values***

e) Repeat exe1.d above but this time using nbins=50 and then 100.

Exercise 2 (simple densities and Moments)

a) Now, let's study some fundamentals on **moments** using Normal and Uniform distributions. We begin by creating a variable x = {0.0, 0.1, 0.2, 0.3, …, 10.0} (in Matlab x=0:0.1:10). Consider a Normal distribution with $\mu = 3.9$ and $\sigma = 0.8$, and plot the distribution as




```
%
% some piece of code in Matlab

mu = 3.9; sig = 0.8;
y = normpdf(x,mu,sig);
```

In Matlab, the function **normpdf** is equivalent to (1/sqrt(2*pi*sig^2))*exp(-0.5/sig^2*(x - mu).^2)

b) Calculate the following moments:
1) first moment =
2) second moment = 16.01
3) 2nd central moment =

Hints (in Matlab):
```
sig1 = sqrt(sig); %because of the "normpdf" convention
myfun_m1 = @(x,mu,sig) x .* exp(-((x-mu).^2)/(2*sig.^2)) / (sig*sqrt(2*pi));
M1 = integral(@(x) myfun_m1(x, mu, sig1), 0, 10);
```

Check whether the following expression is true:
Central_moment = 2ndMoment – (1stMoment)^2

c) Let's study the Uniform distribution by running the following code:

```
dx = 0.01;
x = 0:dx:10;
n = length(x); y = zeros(1,n);
% % Let's define an interval
x_interval = [2,5];
PDF = 1/(5 - 2);
ix1 = find(x == a); ix2 = find(x == b);
y(ix1:ix2) = PDF;
figure(1); cla; hold on
ylabel('pdf'); xlabel('x');
plot(x,y,'-b','LineWidth',1,'MarkerSize',2)
```

Calculate the following moments:
1) first moment =
2) second moment =
3) 2nd central moment = 0.75

Exercise 3 (Central Limit Theorem - CLT)
The CLT states that as the sample size (**n**) gets large, the distribution of the sample mean approaches the Normal regardless of the distribution the r.v. has been drawn.
Implement in Matlab/Python a code to verify the CLT. Below thre is a piece of code in Matlab that may be helpful.

a) Run the code for **n = 1, 2, 10, 50**

```
n = 1; ns = 500;
x = unifrnd(0,10,[n,ns]); % interval [0,10]
% Alternatively:  x = 0 + (10-0).*rand(ns,1);

% Calculate the mean of each sample
Xbar = mean(x);
if n<2, Xbar = x; end
histfit(xbar); % Plot histogram with superimposed Normal
```