University of Stavanger

Stavanger, January 8, 2023

Laboratory exercise 1

ELE520 Machine learning

A PDF version of report of the student solution to the exercise including figures and answers to questions shall be submitted on CANVAS.[1]

The laboratory exercise is Python based and is to be carried out at the laboratory. Acquaintance with functions and passing of variables to and from such functions is important. Those unfamiliar with these concepts may refer to
www.ux.uis.no/∼trygve-e/PythonIntro/pythonintro.pdf

Problem 1

The intention of this problem is to visualise probability density functions and feature vectors. You can download the Python file *pdffuns.py* (Code listing 1) and *labsol1.py* (Code listing 2) from the course web site. The py-file, *pdffuns.py*, implements the 1D pdf-function described in equation 1.

In jupyter notebook you can run the code in *labsol1.py* by giving the command `%run labsol1`. This will generate the plot shown in figure 1.

Figure 1 shows a gaussian probability density function,

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{1}$$

where $\mu = 1$ and $\sigma^2 = 5$. In this problem a step by step expanation is given, how the curve is generated for this function. Based on this description you are supposed to generate a corresponding plot for $p(\boldsymbol{x})$ shown in equation 2 where $\boldsymbol{x}$ is 2-dimensional. It is recommended to add a new function, *norm2D*, to *pdffuns.py* and to rewrite *labsol1.py*, adding the code necessary to plot the two-dimensional pdf function.

[1] If it is not possible to export the Jupyter notebook to PDF, the ipynb and py files can be submitted.

Trygve Eftestøl, Professor
Faculty of Science and Technology
Department of Electrical and Computer Engineering
Kjølv Egelands hus
University of Stavanger
N-4036 Stavanger.
Telephone: +47 51 83 10 00.
Telefax: +47 51 83 17 50.
E-mail: trygve.eftestol@uis.no
www.ux.his.no/˜trygve-e

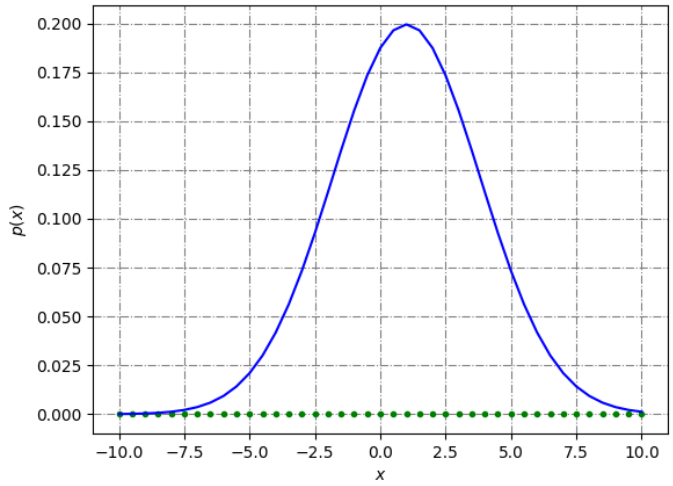

Figure 1: Normally distributed 1-dimensional probability density function.

The probability density function , $p(\boldsymbol{x}),\ \boldsymbol{x} = (x_1\ x_2)^t$, has a gaussian distribution

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{l}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2}$$

around the expected value $\boldsymbol{\mu} = (1\ 1)^t$ with the covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix}. \tag{3}$$

a) The blue curveform is computed in discrete points of computation. These points points of computation are shown as green dots along the x-axis. These points are generated as a vector `x=np.arange(-10,10.5,0.5).reshape(-1,1)` in Python.

   Generate a corresponding 2-dimensional grid of points of computation for the computations of $p(\boldsymbol{x})$. Let the points of computation along the two axes be defined as `x1=np.arange(-10,10.5,0.5).reshape(-1,1)` and `x2=np.arange(-9,10.5,0.5).reshape(-1,1)`. Note that the vectors have different lengths. Use the command `np.meshgrid` to put the coordinates into a `X1`-vector and a `X2`-vector.[2]

b) The function values in the points of computation are computed for every single point of computation and placed into a vector, `p`, by using a `for`-loop. Do the same for $p(\boldsymbol{x})$, but remember to use a double `for`-loop. Place the computed values in the matrix, `p`.

[2] Although not stated in the original vesrion of the problem text, it will be useful to plot the grid points.

c) The function values of $p(x)$ is plotted against x by using the *Matplotlib* function `plot`. Do the same for $p(\boldsymbol{x})$, but by using the plotting function `plot_surface`.

d) Rewrite the function so that you give `mu`, `Sigma` and the points of computation along the axes as input parameters to norm2D and get `p` and the grid of computation points as a result.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import sys
import pdb

def norm1D(my,Sgm,x):

    [n,d]=np.shape(x)
    p = np.zeros(np.shape(x))
    for i in np.arange(0, n):
        p[i] = 1 / (np.sqrt(2 * np.pi) * Sgm) * \
            np.exp(-1 / 2 * np.square((x[i] - my)) / (np.
   square(Sgm)))

    return p
```

Code Listing 1: pdffuns

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from pdffuns import *

x = np.arange(-10, 10.5, 0.5)
x = x.reshape(x.size, 1)
my = 1
Sgm = 2
p = norm1D(my, Sgm, x)

fc = np.array([1, 1, 1]) * 1
pcol = np.array([1, 1, 1]) * 0
fig, ax = plt.subplots(1, 1)
fig.set_facecolor(fc)
ax.plot(x, x * 0, 'g.')
ax.plot(x, p, 'b')
ax.grid(color='gray', linestyle='-.')
```

```
ax.tick_params(colors=pcol)
ax.set(xlabel='$x$', ylabel='$p(x)$')

plt.show()
```

Code Listing 2: labsol1.py