School of Electronics and Computer Science
University of Southampton

COMP6245(2021/2222): Foundations of Machine Learning (MSc) Lab 1

Issue	08 October 2021
Deadline	15 October 2021

Objective

To study two uses of properties of multi-variate Gaussian densities (sampling and projection):

$$\boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{m}, C\right), \ \boldsymbol{y} = A\,\boldsymbol{x} \implies \boldsymbol{y} \sim \mathcal{N}(A\boldsymbol{m}, A\,C\,A^T)$$

Preliminaries

For exercises in this module, we will use `Python` programming language in a `Jupyter` notebook environment. We will assume basic competence in `Python`. Snippets of code will be provided, along with tasks you are required to carry out. What is provided is purely guides to help you get started and should not be taken to be complete working software. As the labs in this module will be online, as a first step, please set up `Anaconda` environment in your own computer.
Here is a piece of code to get started with a `Jupyter` notebook.

```
%matplotlib online
import matplotlib.pyplot as plt
import numpy as np
print("Hello World!")
```

Here is a quick refresher on some manipulations on vectors and matrices we will need in this module along with some simple commands in Python. We will use lower case letters for scalars, lower case bold letters for vectors and upper case letters for matrices.

1. Scalar product of two vectors: $a = \boldsymbol{x}^T\,\boldsymbol{y}$
2. Vector norm: $b = \sqrt{\sum_{i=1}^{d} x^2(i)}$
3. Angle between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$: (look up the expression)
4. Symmetric matrix: $B^T = B$
5. Rank of a matrix as the number of linearly independent rows/columns
6. Matrix vector multiplication: $A\,\boldsymbol{x}$
7. Quadratic form: $b = \boldsymbol{x}^T\,A\,\boldsymbol{x}$
8. Trace of a matrix B: $\sum_i B_{ii}$
9. Determinant of a matrix, denoted $\texttt{det}B$ or $|B|$
10. Eigenvalues and eigenvectors: $B\,\boldsymbol{u} = \lambda\boldsymbol{u}$
11. Advanced topic: Singular Value Decomposition (SVD)

12. Please try the following. At each step, you should ask yourself if you notice any specific property of matrices, vector etc. you recall from previous phases of your education.

```
x = np.array([1, 2])
y = np.array([-2, 1])
a = np.dot(x, y)
print(a)
print(x.T @ y)

b = np.linalg.norm(x)
c = np.sqrt(x[0]**2 + x[1]**2)
print(b, c)

theta = np.arccos(np.dot(x,y) / (np.linalg.norm(x) * np.linalg.norm(x)))
print(theta*180/np.pi)

B = np.array([[3,2,1], [2,6,5], [1,5,9]], dtype=float)
print(B)
print(B - B.T)

z = np.random.rand(3)
v = B @ z
print(B.shape, z.shape, v.shape)

print(z.T @ B @ z)

print(np.trace(B))
print(np.linalg.det(B))

D, U = np.linalg.eig(B)
print(D)
print(U)

print(np.dot(U[:,0], U[:,1]))
print(U @ U.T)
```

What do you observe for the last command above (i.e. print(np.dot(U[:,0], U[:,1])))? Can you formally prove that this is the result you would expect for the specific structure in the matrix B?

13. Now, for some advanced material and fun, find the following two items:

    - A document with title `The Matrix Cookbook` written by K.B.Petersen and M.S.Pedersen. This is a neat resource with all the basics we need and much more. A very useful reference material to have around.
    - "*It had to be U - the SVD song*" on `youtube`, which is a nice piece of art that tells you what Singular Value Decomposition is and gives a good example of where it is used.

1 Random Numbers and Uni-variate Densities

Generate 1000 uniform random numbers and plot a histogram.

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

x = np.random.rand(1000,1)
x = np.random.rand(1000,1)

fig, ax = plt.subplots(nrows=1, ncols=2,figsize=(10,4))
n1bins, n2bins = 4, 40
ax[0].hist(x, bins=n1bins)
ax[0].set_ylim(0,250)
ax[0].set_xlabel("Bins", fontsize=16)
ax[0].set_ylabel("Count", fontsize=16)
ax[0].tick_params(axis='both', which='major', labelsize=14)
ax[0].set_title("Histogram: bins=%4d"%(n1bins), fontsize=16)

ax[1].hist(x, bins=n2bins)
ax[1].set_ylim(0,250)
ax[1].set_xlabel("Bins", fontsize=16)
ax[1].set_ylabel("Count", fontsize=16)
ax[1].tick_params(axis='both', which='major', labelsize=14)
ax[1].set_title("Histogram: bins=%4d"%(n2bins), fontsize=16)

plt.savefig("histograms_uniform.png")
plt.tight_layout()
```

Think through the following:

- Though the data is from a uniform distribution, the histogram does not appear flat. Why?
- Every time you run it, the histogram looks slightly different? Why?
- Do the above observations change (if so how) if you had started with more data (*i.e.* $100,000$ instead of 1000?

Let us now add and subtract some uniform random numbers:

```
N = 1000
x1 = np.zeros(N)
for n in range(N):
    x1[n] = np.sum(np.random.rand(12,1)) - np.sum(np.random.rand(12,1))
fig, ax = plt.subplots(figsize=(5,5))
ax.hist(x1, 20)
```

What do you observe? How does the resulting histogram change when you change the number of uniform random numbers you add and subtract (*i.e.* fewer numbers than 12)?

2 Uncertainty in Estimation

Much of what we study in machine learning has to do with estimating parameters of models from a finite set of data. Consider estimating the variance of a uni-variate Gaussian density using samples drawn from it. When we estimate the variance from different sets of samples ("*different realizations of a process*"), the answer we get each time will be slightly different. But if we had more data, we would expect the variation in the answer to be small. Let's see if this is true:

```
MaxTrial = 2000
sampleSizeRange = np.linspace(100, 200, 40)
plotVar = np.zeros(len(sampleSizeRange))
for sSize in range(len(sampleSizeRange)):
    numSamples = int(sampleSizeRange[sSize])
vStrial = np.zeros(MaxTrial)
for trial in range(MaxTrial):
    xx = np.random.randn(numSamples,1)
vStrial[trial] = np.var(xx)
plotVar[sSize] = np.var(vStrial)
fig, ax = plt.subplots(figsize=(4,4))
ax.plot((plotVar))
```

3 Bi-variate Gaussian Distribution

We will come across the multi-variate Gaussian distribution, defined in some $d-$ dimensional space quite a lot in this module. We can study some properties of this with $d = 2$, bi-variate, because in two dimensions, it is convenient to visualize some of these.

```
def gauss2D(x, m, C):
    Ci = np.linalg.inv(C)
    dC = np.linalg.det(C1)
    num = np.exp(-0.5 * np.dot((x-m).T, np.dot(Ci, (x-m))))
    den = 2 * np.pi * dC

    return num/den

def twoDGaussianPlot (nx, ny, m, C):
    x = np.linspace(-5, 5, nx)
    y = np.linspace(-5, 5, ny)
    X, Y = np.meshgrid(x, y, indexing='ij')

    Z = np.zeros([nx, ny])
    for i in range(nx):
        for j in range(ny):
            xvec = np.array([X[i,j], Y[i,j]])
            Z[i,j] = gauss2D(xvec, m, C)

    return X, Y, Z
```

This is a function of two variables. You can plot contours on this function or visualize it as a three dimensional surface plot.

```
# Plot contours
#
nx, ny = 50, 40
m1 = np.array([0,2])
C1 = np.array([[2,1], [1,2]], np.float32)
Xp, Yp, Zp = twoDGaussianPlot (nx, ny, m1, C1)

plt.contour(Xp, Yp, Zp, 5)
```

Draw contours of the following distributions: $\mathcal{N}\left(\begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\right)$, $\mathcal{N}\left(\begin{bmatrix} 1.2 \\ 0.2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\right)$

and $\mathcal{N}\left(\begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\right)$

4 Sampling from a multi-variate Gaussian

Suppose we are tasked with drawing several samples from a multi-variate Gaussian density with mean $\boldsymbol{m} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and covariance matrix $C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$. Here is a way to do this using the properties of multi-variate Guassians we have learnt:

- Factorize the covariance matrix into a lower triangular matrix and its transpose: $C = A\,A^T$:

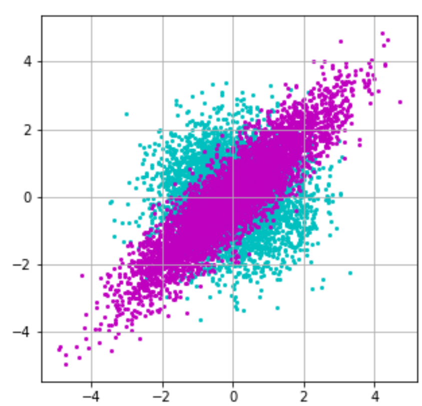
```
C = np.array([[2.0,1.0], [1.0,2]])
A = np.linalg.cholesky(C)
print(A @ A.T)
```

- Generate 5000 bivariate Gaussian random data by `X = np.random.randn(5000,2)` and transform each of the data (rows of X) by $Y = X\,A$.

```
X = np.random.randn(1000,2)
Y = X @ A
print(X.shape)
print(Y.shape)
```

- Draw scatter plots of X and Y

```
fig, ax = plt.subplots(figsize=(5,5))
ax.scatter(X[:,0], X[:,1], c="c", s=4)
ax.scatter(Y[:,0], Y[:,1], c="m", s=4)
ax.set_xlim(-6, 6)
ax.set_ylim(-6, 6)
```



5 Distribution of Projections

- Construct a vector $\boldsymbol{u} = [\sin\theta \quad \cos\theta]^T$, parameterized by the variable θ.

```
theta = np.pi/3
u = [np.sin(theta), np.cos(theta)]
print("The vector: ", u)
print("Magnitude : ", np.sqrt(u[0]**2 + u[1]**2))
print("Angle     : ", theta*180/np.pi)
```

- Compute the variance of projected data along this direction

```
yp = Y @ u
print(yp.shape)
print("Projected variance: ", np.var(yp))
```

- Now, using the above write a program that plots the variance of the projected data as you change θ over the range 0 to 2π.

```
# Store projected variances in pVars & plot
#
nPoints = 50
pVars = np.zeros(nPoints)
thRange = np.linspace(0, 2*np.pi, nPoints)
for n in range(nPoints):
    theta = thRange[n]
    u = [np.sin(theta), np.cos(theta)]
    pVars[n] = np.var(Y @ u)

fig, ax = plt.subplots(figsize=(5,3))
ax.plot(pVars)
```

  What are the maxima and minima of the resulting plot?

- Compute the eigenvalues an eigenvectors of the covariance matrix C
- Can you see a relationship between the eignevalues and eigenvectors and the maxima and minima of the way the projected variance changes?
- The shape of the graph might have looked sinusoidal for this two dimensional problem. Can you analytically confirm if this might be true?

Report

Upload a report of **no more than four pages** describing your work to ECS Handin page set up for this exercise. Make sure you write your name and email on the work you upload. If possible, use LaTeXto typeset your report. Do not include screen dumps; export your figures as high quality images; make sure any figures uncluded are numbered and referred to by their numbers in the text; write succinct captions to figures.

Mahesan Niranjan October 2021