School of Electronics and Computer Science
University of Southampton

COMP6245(2019/20): Foundations of Machine Learning Lab 1(of 6)

Issue	01/10 2019
Deadline	10/10 (10:00 AM)
Feedback by	14/10

Objective

Two uses (sampling and projections) of the property of multivariate Gaussian densities

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

Preliminaries

For laboratory exercises, we will use `Python` programming language in a `Jupyter Notebook` environment. Snippets of code will be provided along with tasks you are being asked to do. These are provided to help you get started. These should not be assumed to be complete working pieces of software.
Familiarise yourselves with the desktop computing environment in the laboratory. Start a `cmd` window, navigate to a directory where you can save your files and activate the notebook by typing `jupyter notebook`. Here is how to get started with some commands that we will often find on top of the notebook in this course:

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

1 Linear Algebra

Here is a quick refresher on some simple manipulations on vectors and matrices we will need in this module:

1. Scalar product between two vectors: $a = \boldsymbol{x}^T \boldsymbol{y}$
2. Norm of a vector: $b = \sqrt{x(1)^2 + x(2)^2}$
3. Angle between two vectors
4. Matrix vector multiplication: $\boldsymbol{v} = B\,\boldsymbol{z}$
5. Quadratic form: $\boldsymbol{z}^T B\,\boldsymbol{z}$
6. Trace of a matrix: $\sum b_{ii}$
7. Determinant of a matrix: $\det(B)$ or $|B|$
8. Matrix inverse: $\text{inv}(B)$ or B^{-1}
9. Eignevalues and Eigenvectors: $B\,\boldsymbol{u} = \lambda\,\boldsymbol{u}$

10. Advanced Topic: Singular Value Decomposition (SVD)

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

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

theta = np.argcos(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(v.shape)
print(v)

print(z.T @ B @ z)

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

print(np.linalg.inv(B) @ 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 very last command in the above exercise (*i.e.* `print(U @ U.T)`? Can you formally prove this is what you would expect for the specific structure in matrix B? Advanced material: Search for a document with title `The Matrix Cookbook` which is a beautiful collection of matrix identities and results.

2 Random Numbers and Univariate Distributions

Generate 1000 uniform random numbers and plot a histogram.

```
x = np.random.rand(10000,1)
plt.figure(figsize=(3,3))
n, bins, patches = plt.hist(x, bins=20,
                       color='m', alpha=0.8, rwidth=0.8)
print(n)
print(bins)
```

Note how the choice of the number of bins and the number of samples affects the histogram you observe. Here is a bit of code to help you study this.

```
MaxTrials = 10
NumSamples = 200
NumBins = 20
for trial in range(MaxTrials):
    x = np.random.rand(NumSamples,1)
    counts, bins, patches = plt.hist(x, NumBins)
    plt.clf()
    print('Variation within bin counts: ', np.var(counts/NumSamples))
```

Repeat the above with 1000 random numbers drawn from a Gaussian distribution of mean 0 and standard deviation 1 using `x = np.random.randn(1000,1);`.

Now try the following where we add and subtract uniform random numbers:

```
N = 1000;
x1 = np.zeros(N)
for n in range(N):
    x1[n] = sum(np.random.rand(12,1))-np.sum(np.random.rand(12,1));
plt.hist(x1,40,
    color='b', alpha=0.8, rwidth=0.8)
plt.xlabel('Bin', FontSize=16)
plt.ylabel('Counts', FontSize=16)
plt.grid(True)
plt.title('Histogram',FontSize=16)
```

What do you observe? How does the resulting histogram change when you change the number of numbers you add/subtract? Is there a theorem that explains it?

3 Uncertainty in Estimation

Much of what we do in Machine Learning is estimating parameters of a model from a finite set of data. Here is why one might think having more data is a good thing: Consider estimating the variance of a uni-variate Gaussian distribution from samples drawn from it. If we did this from different realiations of the samplings, we would expect slightly different answers (every time we do it). We would, of course, like this variation to be small. Fig. 1 shows how the variance of the estimation varies with sample size. Here is a snippet of code to study this.

```
sampleSizeRange = np.linspace(100,200,40)
plotVar = np.zeros(len(sampleSizeRange))
for sSize in range(len(sampleSizeRange)):
    numSamples = np.int(sampleSizeRange[sSize])
    MaxTrial=2000
    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)

plt.plot(sampleSizeRange, plotVar)
```

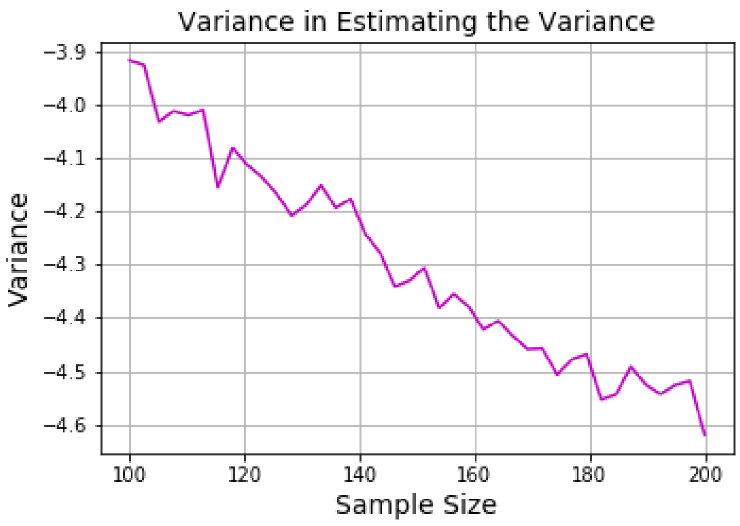

Figure 1: Uncertainty in Estimation

4 Bivariate Gaussian Distribution

Here are two functions to help us calculate and plot contours on a multivariate Gaussian density.

```
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
```

Let us now plot these for three different values of means and covariance matrices.

```
nx, ny = 50, 40
plt.figure(figsize=(6,6))

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, 3)
plt.grid(True)
```

On the same plot draw contours on the two bivariate Gaussian densities:

$$\boldsymbol{m2} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \ \boldsymbol{C2} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{m3} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \ \boldsymbol{C2} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

5 Sampling from a Multivariate Gaussian Distribution

Consider the covariance matrix $\boldsymbol{C} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.
Factorize this into a lower trinagular matrix and its transpose (Cholosky decomposition), $\boldsymbol{A}^t \boldsymbol{A} = \boldsymbol{C}$, and confirm the factorization is correct by multiplying.

```
C=[[2, 1], [1, 2]]
print(C)
A = np.linalg.cholesky(C)
print(A)
print(A @ A.T)
```

Generate 10000 bivariate Gaussian random numbers by `X = np.random.randn(1000,2);` Transform each of the two dimensional vectors (rows of `X`) by `Y = X @ A`.

```
X = np.random.randn(10000,2)
Y = X @ A
print(X.shape)
print(Y.shape)
```

Now draw a scatter plot of `X` and `Y`.

```
plt.scatter(Y[:,0], Y[:,1], s=3, c='m')
plt.scatter(X[:,0], X[:,1], s=3, c='c')
plt.grid(True)
plt.title('Scatter of Isotropic and Correlated Gaussian Densities')
```

What do you observe?

6 Distribution of Projections

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

```
theta = np.pi/3
u = [np.sin(theta), np.cos(theta)]
print('The vector: ', u)
print('Sum of squares: ', u[0]**2+u[1]**2)
print('Degrees: ', theta*180/np.pi)
```

Compute the variance of projections of the data in $\boldsymbol{X}$ along this direction:

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

Plot how this projected variance changes as a function of θ:

```
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 )

plt.plot(pVars)
plt.grid(True)
plt.xlabel('Direction', fontsize=14)
plt.ylabel('Variance of Projections', fontsize=14)
```

Explain what you observe by calculating the eigenvectors of the covariance matrix. How does what you have done above differ for $\boldsymbol{C} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$.

Export the figures for inclusion in a report. Try `plt.savefig`.

Report

Describe the work you have done as a short report. Upload a *pdf* file **no longer than four pages** using the ECS handin system: `http://handin.ecs.soton.ac.uk`. Please use LaTeX to typeset your report. If you are not familiar with LaTeX, now is the best time to learn. Please make sure your name and email are included in the report you upload.