

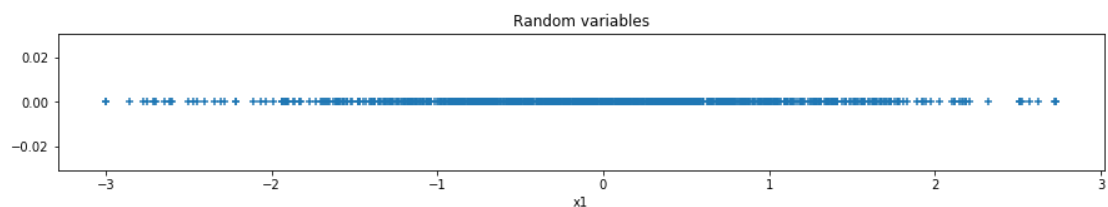
Gaussian distribution or normal distribution

Describing random variables

In probability, we have variables drawn randomly. The first way to get to know these variables is to draw a histogram, to see how these variables are distributed.

```
In [2]: import matplotlib.mlab as mlab
import numpy as np
from matplotlib import pyplot as plt

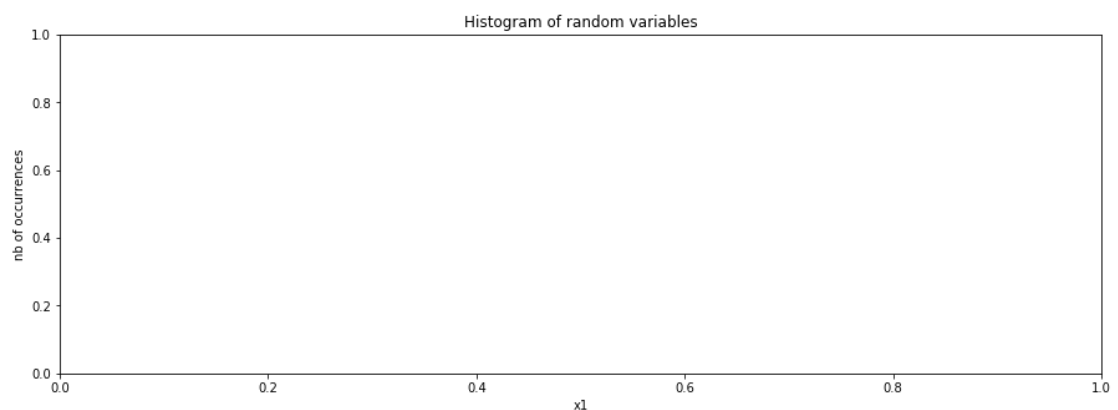
arr = np.random.randn(1000)
fig = plt.figure(figsize=(15,2))
plt.scatter(arr,np.zeros(len(arr)),marker = '+')
plt.title('Random variables')
plt.xlabel('x1')
plt.show()
```



Q1. Please plot the histogram of the random variable x1.

```
In [16]: plt.figure(figsize=(15,5))
#Q1. plot the histogram
#...

plt.title('Histogram of random variables')
plt.xlabel('x1')
plt.ylabel('nb of occurrences')
plt.show()
```



Gaussian function with 1 variable (1-D)

The Gaussian function is defined as $\mathcal{N}(x) = \frac{1}{\sigma\sqrt{2\pi}} * \exp(-\frac{(x-\mu)^2}{2\sigma^2})$

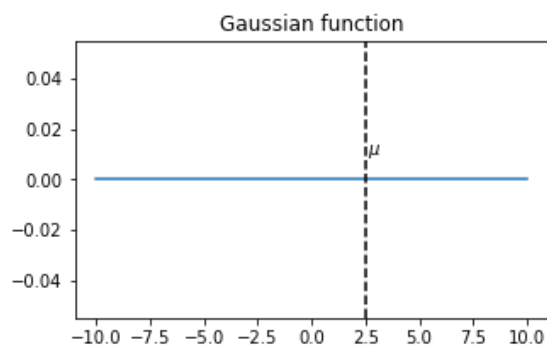
Q2. Please plot a Gaussian with mean 2.5 and variance 3.

```
In [15]: from matplotlib import pyplot as plt
import numpy as np

def gaussian(x, mu, sig):
    #Q2. Please define the Gaussian function
    #...
    return 0*x

mu= 2.5;
sig=3;
fig = plt.figure(figsize=(5,3))
x= np.linspace(-10, 10, 120);
plt.plot(x,gaussian(x, mu, sig))

plt.axvline(mu,color='k', linestyle='--')
plt.title('Gaussian function')
plt.text(mu+0.1, .01, r'$\mu$')
plt.show()
```



Gaussian distribution

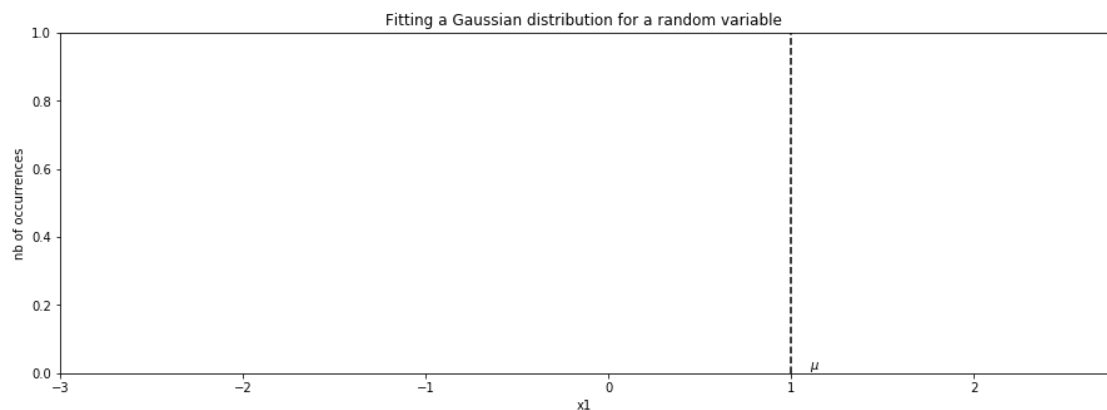
We can describe a random variable distribution with a Gaussian function.

Q3. Please compute the parameters of the Gaussian function to describe the distribution of x_1 .

```
In [20]: fig = plt.figure(figsize=(15,5))
plt.xlim((min(arr), max(arr)))

mu = 1
sig = 1
#Q3. Please compute the parameters of the Gaussian function to describe the distribution of x1,
#then plot the histogram and the gaussian function
#...

plt.title('Fitting a Gaussian distribution for a random variable')
plt.xlabel('x1')
plt.ylabel('nb of occurrences')
plt.axvline(mu,color='k', linestyle='--')
plt.text(mu+0.1, .01, r'$\mu$')
plt.show()
```



Multivariate Normal/Gaussian distribution

The multivariate normal distribution of a k -dimensional random vector $X = [X_1, X_2, \dots, X_k]^T$ can be written in the following notation:

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

with k -dimensional mean vector: $\boldsymbol{\mu} = E[\mathbf{X}] = [E[X_1], E[X_2], \dots, E[X_k]]^T$,

and $k \times k$ covariance matrix: $\boldsymbol{\Sigma} =: E[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T] = [\text{Cov}[X_i, X_j]; 1 \leq i, j \leq k]$. is supposed positive definite.

Concretely, the density function is defined by:

$$f(\mathbf{x}) = f(x_1, \dots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

Illustration with 2D Gaussians

Q4. Please plot a 2D Gaussian with $\mu=[0.1, 0.5]$ and variance=0.5 for both dimensions

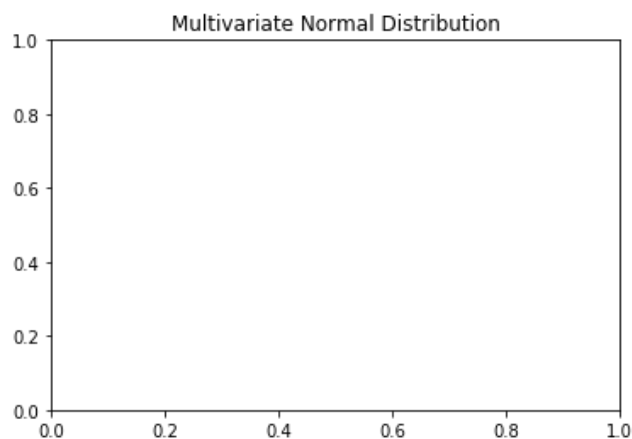
```
In [25]: import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import proj3d
import pylab
import numpy as np
from scipy.stats import multivariate_normal

x, y = np.mgrid[-1.0:1.0:30j, -1.0:1.0:30j]

# Need an (N, 2) array of (x, y) pairs.
xy = np.column_stack([x.flat, y.flat])
z = 0*x;

#Q4. Please plot a 2D Gaussian with mu=[0.1,0.5] and variance=0.5 for both dimensions
#...

plt.title('Multivariate Normal Distribution')
ax.set_zlim(-0.5,0.6)
ax.set_zticks(np.linspace(0,0.6,6))
ax.view_init(27, -21)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('Probability')
plt.show()
```



Gaussian Mixture Model

A Bayesian Gaussian mixture model is commonly extended to fit a vector of unknown parameters (denoted in bold), or multivariate normal distributions. In a multivariate distribution (i.e. one modelling a vector \mathbf{x} with N random variables) one may model a vector of parameters (such as several observations of a signal or patches within an image) using a Gaussian mixture model prior distribution on the vector of estimates given by

$$p(\mathbf{X}, \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^K \pi_i \mathcal{N}(\mathbf{X}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

where the i^{th} component is characterized by normal distributions with weights π_i , means $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{\Sigma}_i$.

Q5. Please plot a GMM with $K=2$ Gaussians with :

- $\pi_1 = 0.3$, $\mu_1 = [-0.6, 0.0]$, $\text{covariance}_1 = \begin{bmatrix} .2 & 0 \\ 0 & .2 \end{bmatrix}$
- $\pi_2 = 0.7$, $\mu_2 = [0.5, 0.0]$, $\text{covariance}_2 = \begin{bmatrix} .4 & 0 \\ 0 & .4 \end{bmatrix}$

```
In [27]: #Q5. Please plot a GMM with K=2 Gaussians
#...

plt.title('Gaussian Mixture')
ax.set_zlim(-1.5,1.5)
ax.set_zticks(np.linspace(0,1.5,5))
x2, y2, _ = proj3d.proj_transform(mu1[0],mu1[1],1.6, ax.get_proj())
pylab.annotate(
    "$gaussian(\mu_1)$",
    xy = (x2,y2), xytext = (-20, 20),
    textcoords = 'offset points', ha = 'right', va = 'bottom',
    bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha = 0.5),
    arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=0'))
x3, y3, _ = proj3d.proj_transform(mu2[0],mu2[1],1, ax.get_proj())
pylab.annotate(
    "$gaussian(\mu_2)$",
    xy = (x3,y3), xytext = (20, 20),
    textcoords = 'offset points', ha = 'right', va = 'bottom',
    bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha = 0.5),
    arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=0'))
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('Probability')
plt.show()

-----
NameError                                Traceback (most recent call last)
<ipython-input-27-93cc46aed9cf> in <module>()
      6 ax.set_zlim(-1.5,1.5)
      7 ax.set_zticks(np.linspace(0,1.5,5))
----> 8 x2, y2, _ = proj3d.proj_transform(mu1[0],mu1[1],1.6, ax.get_proj())
      9 pylab.annotate(
     10     "$gaussian(\mu_1)$",

NameError: name 'mu1' is not defined
```

Gaussian Mixture Models

Modeling the problem

We consider that we have M demonstrations of the task. Each task $m \in 1, \dots, M$ contains datapoints $\mathbf{X}_t = (x, y)$. Aggregating these datapoints from all demonstrations, we have N datapoints $\mathbf{X}_{t=1}^N$.

Probabilistic model

In Gaussian Mixture Model (GMM), we model the data \mathbf{X} with a probability density function composed of K Gaussians :

$$p(\mathbf{X}, \pi, \mu, \Sigma) = \sum_{i=1}^K (\pi_i \mathcal{N}(\mathbf{X}, \mu_i, \Sigma_i))$$

where the i^{th} component is characterised by a normal distribution with weight π_i , mean μ_i and a covariance matrix Σ_i .

Learning the probability function

We first model our demonstrations data with a GMM. We must first find parameters π_i, μ_i, Σ_i so that function p represents best our data, ie. so as to maximise $p(\pi, \mu, \Sigma | \mathbf{X})$. The most common method is the Expectation-Maximization (E-M) algorithm.

- Initialise $\pi_i, \mu_i, \Sigma_i \leftarrow$ random values}
- LOOP
 - Expectation
 - $h_{t,i} \leftarrow \pi_i \mathcal{N}(\mathbf{X}_t | \mu_i, \Sigma_i)$
 - $h_{t,i} \leftarrow \frac{h_{t,i}}{\sum_{n=1}^K h_{t,n}}$
 - Maximisation
 - $\pi_i \leftarrow \frac{\sum_{t=1}^N h_{t,i}}{N}$
 - $\mu_i \leftarrow \frac{\sum_{t=1}^N h_{t,i} \mathbf{X}_t}{\sum_{t=1}^N h_{t,i}}$
 - $\Sigma_i \leftarrow \frac{\sum_{t=1}^N h_{t,i} (\mathbf{X}_t - \mu_i)(\mathbf{X}_t - \mu_i)^T}{\sum_{t=1}^N h_{t,i}}$

We will model data with a GMM with $K=3$ Gaussians

Q6. Import the data, plot them and implement your variable \mathbf{X}_t . The data is in file data.npy and contains a numpy ndarray of size (120,10). The first line corresponds to the position x and the others correspond to recordings of y .

```
In [28]: import numpy as np
from matplotlib import pyplot as plt

readdata = np.load("data.npy")

#print readdata
#print type(readdata)
#print readdata.shape

#Q6. Plot the data and implement your variable  $\mathbf{X}_t$ 
# ....
```

```
-----
IOError                                Traceback (most recent call last)
<ipython-input-28-fce53c5579af> in <module>()
      2 from matplotlib import pyplot as plt
      3
----> 4 readdata = np.load("data.npy")
      5
      6 #print readdata

/anaconda2/lib/python2.7/site-packages/numpy/lib/npio.pyc in load(file, mmap_
mode, allow_pickle, fix_imports, encoding)
    368     own_fid = False
    369     if isinstance(file, basestring):
--> 370         fid = open(file, "rb")
    371         own_fid = True
    372     elif is_pathlib_path(file):

IOError: [Errno 2] No such file or directory: 'data.npy'
```

Q7. Implement E-M algorithm and plot the Gaussians