Lecture 3 : The normal distribution

The Normal distribution (also called a Gaussian distribution) is one of the most popular descriptions of distribution mainly because of the *central limit* effect. The central limit effect represents the case of experimental errors driven purely by randomness that tend to be roughly symmetric around a central value. The normality of the experimental errors can be observed when there is a multitude of components that can result in an experimental error. The normal distribution is parameterized by the mean μ and variance σ^2 and can be expressed using the following analytical expression:

$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

A useful notion when working with normal distribution is to reformulate our sample value using standardization which means the samples have a mean of zero and a standard deviation of 1:

$$z = \frac{y-\mu}{\sigma}$$

Using z, we can also describe a sample value using a *score* that represents the probability of it being between the mean and one standard deviation. This notion is particularly effective in removing magnitude-based effects when we analyze the data. The resulting distribution is given by the following:

$$p(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)$$

We denote a Normal distribution using $\mathcal{N}(\mu, \sigma)$ to highlight that it is parameterized by mean and standard deviation. Having analytical expressions allows us to write computer programs to perform random sampling from a distribution to understand its behavior. The following python code defines a standardized normal distribution and random samples of 5 points.

```
>>> from scipy.stats import norm
>>> r = norm.rvs(size=5)
>>> print(r)
[-1.60551724  1.29864839 -0.98856248 -1.83191687 -1.96522119]
```

We can sample much more points and plot a histogram and see how well it matches the corresponding analytic expression for the pdf:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

z = np.linspace(-3.5,3.5, 100)
r = norm.rvs(size=1000)
rv = norm()
fig, ax = plt.subplots()
ax.plot(z, rv.pdf(z), 'k-', lw=2, label=r'$\mathcal{N}(0,1)$')
ax.hist(r, density=True, histtype='stepfilled', alpha=0.2, label='Samples')
ax.set_xlabel(r'$z$')
ax.set_ylabel(r'$p(z)$')
fig.legend()
plt.show()
```

Scientific experiments are seldom run with just a single variable that impacts the outcome. The Normal distribution equivalent for the multi-variable case is called a multi-variate normal distribution or *mvn* in short. Suppose we are interested in expressing a mvn of a k variables which we

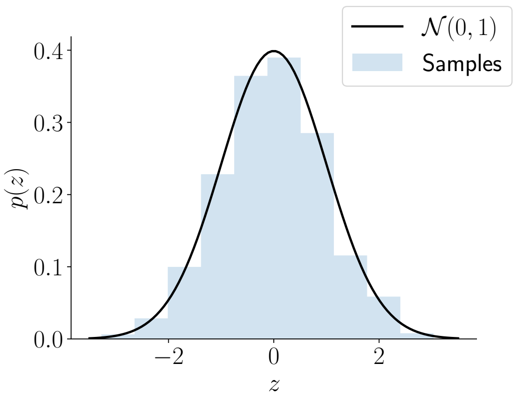

Figure 1: Standardized normal distribution: sampled for 1000 random points

denote using a random vector of size k using $X = [x_1, x_2, \ldots, x_k]$. Much like the single variable case, we need a mean value μ and a covariance Σ to parameterize an mvn using $\mathcal{N}(\mu, \Sigma)$. The two parameters required are obtained as follows from the population:

$$\begin{aligned}\mu &= \mathbb{E}(X) = [\mathbb{E}(x_1), \mathbb{E}(x_2), \ldots, \mathbb{E}(x_k)] \\ \Sigma_{ij} &= \mathbb{E}((x_i - \mu)(x_j - \mu))\end{aligned}$$

We could also talk about what it means for an mvn to be standardized by considering that each component of our random vector is z-normalized i.e. have a mean zero and variance 1. When the $k \times k$ matrix Σ is *positive definite* (i.e. eigenvalues are real positive numbers), we can obtain a probability density function for the random variables using the following formula:

$$p(X_k) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1} (x-\mu)\right)$$

In Figure 2, we have used the following python code to generate samples of different sizes and visualized them in the two-dimensional plane they are drawn from for a given mvn.

```python
from scipy.stats import multivariate_normal as mvn

dist = mvn([0.5, -0.2], [[2.0, 0.3], [0.3, 0.5]])
fig, axs = plt.subplots(1,3, figsize=(3*5, 5))
fig.subplots_adjust(wspace=0.3)
for i, n in enumerate([10,200,1000]):
    s = dist.rvs(size=n)
    axs[i].scatter(s[:,0], s[:,1], color='k', alpha=0.5)
    axs[i].set_xlim([-3,3])
    axs[i].set_ylim([-3,3])
    axs[i].set_title(r'$N = %d$'%n)
plt.show()
```

Observe that as we increase the number of samples drawn, we see a clear pattern of the density emerge around the mean location of the underlying distribution at $\mu = [0.5, -0.2]$. The evaluation of the analytic probability distribution function is plotted on a two-dimensional grid using the contour functions in python as follows shown in Figure 3:

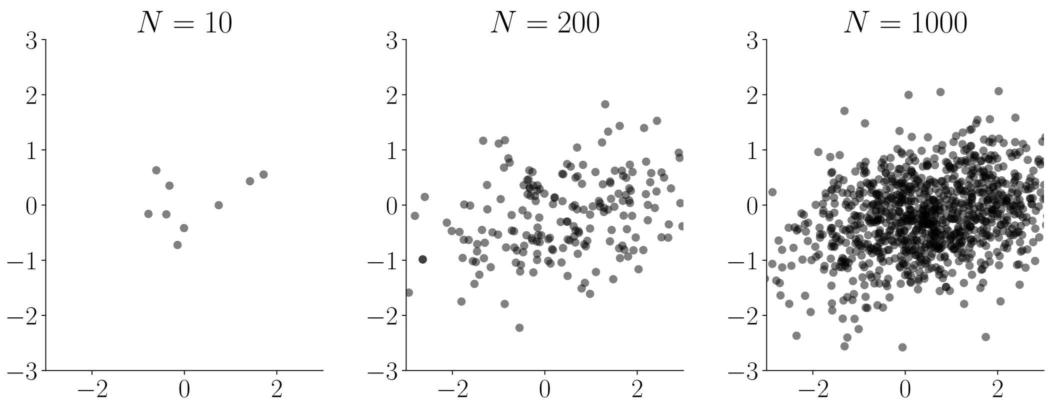

Figure 2: Sampling in a two-dimensional mvn for different sizes

```python
x, y = np.mgrid[-3:3:.01, -3:3:.01]
pos = np.dstack((x, y))
p = dist.pdf(pos)
fig, ax = plt.subplots(figsize=(3,3))
c = ax.contourf(x, y, p, cmap='Blues')
ax.set_xlabel(r'$x_{1}$')
ax.set_ylabel(r'$x_{2}$')
cax = ax.inset_axes([1.05, 0.05, 0.05, 0.9], transform=ax.transAxes)
fig.colorbar(c, cax=cax)
plt.show()
```

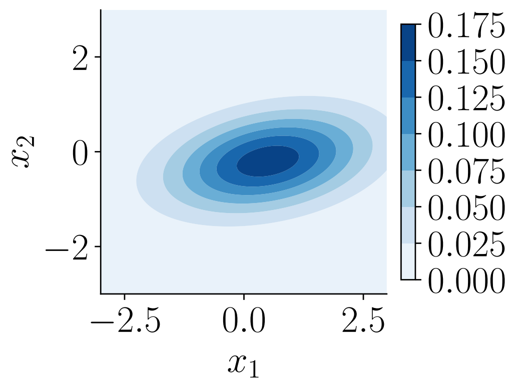

Figure 3: Probability density function of an mvn