Probability and Random Process Assignment

Normal and Laplace Distribution

Normal Distribution and Laplace Distribution are two of the most commonly used statistical distributions. Normal Distribution is a continuous probability distribution which is symmetrical around the mean and has a bell-shaped curve. Laplace Distribution is a continuous probability distribution which is asymmetric, with two modes that are separated by a dip in the center. Both Normal and Laplace Distributions are used to model real-world data and can be implemented in Python using the built-in scipy.stats and numpy packages. These packages provide classes and functions to generate random data based on the Normal and Laplace Distributions, as well as to calculate the probability density functions, cumulative distribution functions and other related statistical measures. We used python and required packages for showing how random variable are used in probability density functions

Normal Distribution

In a normal distribution, data is symmetrically distributed with no skew. When plotted on a graph, the data follows a bell shape, with most values clustering around a central region and tapering off as they go further away from the center. Normal distributions are also called Gaussian distributions or bell curves because of their shape.

The normal distribution is a continuous probability distribution. It is also called the Gaussian distribution because it was first described by Carl Friedrich Gauss. The normal distribution is a very important distribution in statistics. It is used to model many random phenomena. For example, the heights of people in a population are approximately normally distributed. The normal distribution is also used to model the distribution of errors in measurements.

We can use programming to generate random numbers from a normal distribution. The probability density function for norm is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

where μ is the mean and σ is the standard deviation. The square of the standard deviation, σ^2, is called the variance.

In [ ]:
```
import numpy as np
from scipy.stats import norm
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1)
```

Calculate the first four moments:

In [ ]:
```
mean, var, skew, kurt = norm.stats(moments='mvsk')
```

Display the probability density function (pdf):

In [ ]:
```
x = np.linspace(norm.ppf(0.01),norm.ppf(0.99), 100)
ax.plot(x, norm.pdf(x),
       'r-', lw=5, alpha=0.6, label='norm pdf')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1ed740ef640>]
```

Freeze the distribution and display the frozen pdf:

In [ ]:
```
rv = norm()
ax.plot(x, rv.pdf(x), 'k-', lw=2, label='frozen pdf')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1ed74289f40>]
```

Check accuracy of cdf and ppf:

In [ ]:
```
vals = norm.ppf([0.001, 0.5, 0.999])
np.allclose([0.001, 0.5, 0.999], norm.cdf(vals))
```

Out[ ]:
```
True
```

Generate random numbers:

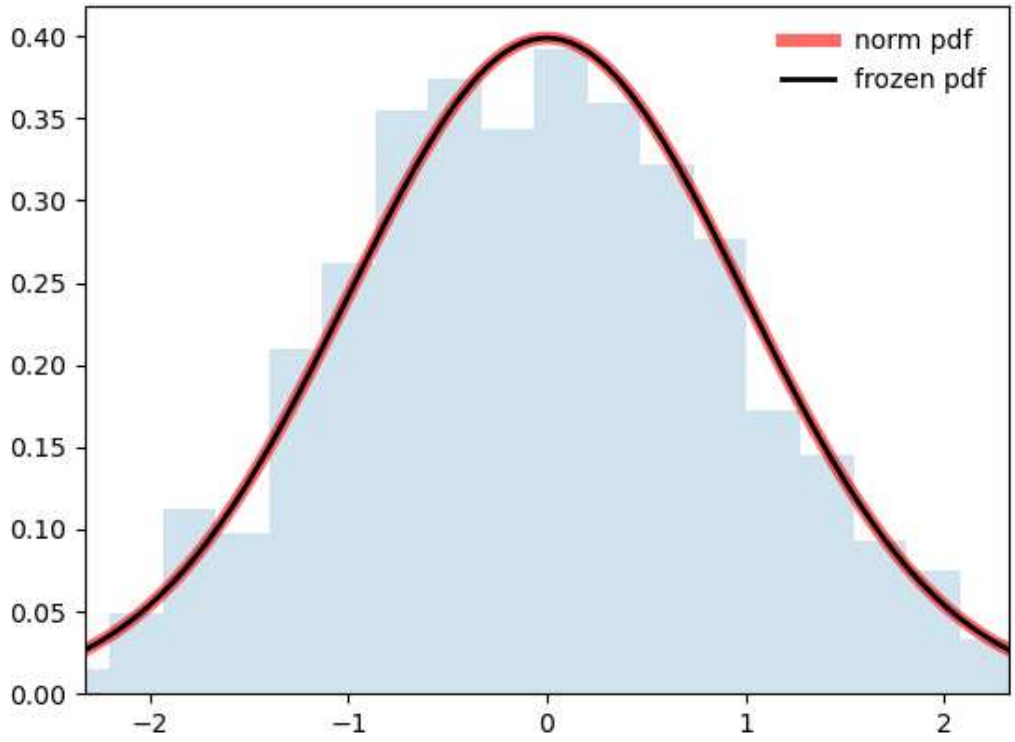
In [ ]:
```
r = norm.rvs(size=1000)
```

And compare the histogram:

In [ ]:
```
ax.hist(r, density=True, bins='auto', histtype='stepfilled', alpha=0.2)
ax.set_xlim([x[0], x[-1]])
ax.legend(loc='best', frameon=False)
plt.show()
```

In [ ]:
```
from IPython import display
display.Image("Figure_2.png")
```

Out[ ]:



Laplace Distribution

In probability theory and statistics, the Laplace distribution is a continuous probability distribution named after Pierre-Simon Laplace. It is also sometimes called the double exponential distribution, because it can be thought of as two exponential distributions (with an additional location parameter) spliced together along the abscissa, although the term is also sometimes used to refer to the Gumbel distribution. The difference between two independent identically distributed exponential random variables is governed by a Laplace distribution, as is a Brownian motion evaluated at an exponentially distributed random time. Increments of Laplace motion or a variance gamma process evaluated over the time scale also have a Laplace distribution. A random variable has a Laplace distribution if its probability density function is

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

Here, μ is a location parameter and $b > 0$, which is sometimes referred to as the "diversity", is a scale parameter. If $\mu = 0$ and b=1, the positive half-line is exactly an exponential distribution scaled by 1/2.

The probability density function of the Laplace distribution is also reminiscent of the normal distribution; however, whereas the normal distribution is expressed in terms of the squared difference from the mean μ , the Laplace density is expressed in terms of the absolute difference from the mean. Consequently, the Laplace distribution has fatter tails than the normal distribution.

In [ ]:
```
import numpy as np
```

```
from scipy.stats import laplace
import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1)
```

Calculate the first four moments:

In [ ]:
```
mean, var, skew, kurt = laplace.stats(moments='mvsk')
```

Display the probability density function (pdf):

In [ ]:
```
x = np.linspace(laplace.ppf(0.01),
                laplace.ppf(0.99), 100)
ax.plot(x, laplace.pdf(x),
       'r-', lw=5, alpha=0.6, label='laplace pdf')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1ed01866970>]
```

Freeze the distribution and display the frozen pdf:

In [ ]:
```
rv = laplace()
ax.plot(x, rv.pdf(x), 'k-', lw=2, label='frozen pdf')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1ed01871400>]
```

Check accuracy of cdf and ppf:

In [ ]:
```
vals = laplace.ppf([0.001, 0.5, 0.999])
np.allclose([0.001, 0.5, 0.999], laplace.cdf(vals))
```

Out[ ]:
```
True
```

Generate random numbers:

In [ ]:
```
r = laplace.rvs(size=1000)
```

And compare the histogram:

In [ ]:
```
ax.hist(r, density=True, bins='auto', histtype='stepfilled', alpha=0.2)
ax.set_xlim([x[0], x[-1]])
ax.legend(loc='best', frameon=False)
plt.show()
```

In [ ]:
```
from IPython import display
display.Image("Figure_3.png")
```

Out[ ]: