Stochastic processes and Data mining

Meher Krishna Patel

Created on : Octorber, 2017

Last updated : May, 2020

Table of contents

Chapter 1

The concept of Random Variable

1.1 Introduction

In this chapter, concepts of random variables, distribution and density functions are discussed. Also, some of the widely used distribution functions are shown in this chapter.

Note: Good knowledge of probability theory and python is required for this tutorial. Click here to see the basic tutorials on various topics.

1.2 Random variable (r.v.)

A random variable $\mathbf{x}(\zeta)$ is a single-valued function which assigns a real number (i.e. value of $\mathbf{x}(\zeta)$) to each outcome (i.e. ζ) of an experiment. Since random variable converts the outcomes into the numbers, therefore it allows us to perform various mathematical operations on the outcomes, such as calculation of probabilities, distribution and density functions etc.

Note: $X(\zeta)$ is usually written as X.

1.2.1 Example: Defining random variable

In the experiment of tossing a coin three times, let X is the r.v. which gives the number of heads in the experiment. Then all the possible outcomes of the experiment can be represented as below,

ζ (outcomes)	$X(\zeta)$ (counts of H)
TTT	0
TTH	1
THT	1
THH	2
HTT	1
HTH	2
HHT	2
HHH	3

- Find the probability of $X = 2$ from above table i.e. number of heads = 2.

We can find the probability by looking at either side of the table i.e. by looking at the pattern for two 'HH' (from left side) or numbers (from right side). Since there are three rows (out of eight rows) with two 'HH', therefore the probability is 3/8.

- Similarly, the probability of $X < 2$ (i.e. 1 or 0 H) is 4/8.

Note:

- In above two calculations, r.v. (on the right side of table) has no advantage over the outcomes (on left side of the table). But, assigning the numbers to outcomes can be extremely useful, if we use some programming language to calculate the probability as shown below,

```
>>> # store outcomes in a list
>>> X = [0, 1, 1, 2, 1, 2, 2, 3]

>>> # Pr(X=2)
>>> x_eq_2 = [x for x in X if x==2]  # Find 2 in X
>>> x_eq_2
[2, 2, 2]

>>> pr_x_eq_2 = len(x_eq_2)/len(X)
>>> pr_x_eq_2
0.375

>>> # Pr(X<2)
>>> # find number less than 2 in X
>>> x_lt_2 = [x for x in X if x<2]
>>> x_lt_2
[0, 1, 1, 1]

>>> pr_x_lt_2 = len(x_lt_2)/len(X)
>>> pr_x_lt_2
0.5
```

- Hence, r.v. can be quite useful, when we want to calculate the probability of outcomes of some experiment, which has very large number of possible outcomes.

1.3 Cumulative Distribution function (CDF)

Cumulative distribution function ({F_X}(x)) of a r.v. 'X' is define as,

$$F_X(x) = P(X \leq x), -\infty < x < \infty$$

- We can calculate the distribution function for example *Example: Defining random variable* as follows,

x	$X \leq x$	$F_X(x)$
-1	ϕ	0
0	{TTT}	1/8
1	{TTT, TTH, THT, HTT}	4/8
2	{TTT, TTH, THT, HTT, HHT, HTH, THH }	7/8
3	S (complete set)	1
4	S (complete set)	1

- Again, same can be achieved using Python as follows,

```
>>> # list of outcomes
>>> X = [0, 1, 1, 2, 1, 2, 2, 3]
```

(continues on next page)

(continued from previous page)

```
>>> F = []  # list to store distribution values
>>> for i in range(-1, 5):    # -1 to 4
...     t = [x for x in X if x<=i] # calculate distribution
...     F.append(len(t)/len(X))    # append distribution in F
...
>>> F   # print(F)
[0.0, 0.125, 0.5, 0.875, 1.0, 1.0]
```

1.3.1 Properties of CDF

1. $$0 \leq F_X(x) \leq 1$$

2. $$F_X(x_1) \leq F_X(x_2), \quad if \ x_1 < x_2$$

3. $$\lim_{x \to \infty} F_X(x) = F_X(\infty) = 1$$

4. $$\lim_{x \to -\infty} F_X(x) = F_X(-\infty) = 0$$

5. $$\lim_{x \to a^+} F_X(x) = F_X(a^+) = F_X(a), \quad a^+ = \lim_{\varepsilon \to 0} a + \varepsilon$$

6. $$P(a < X \leq b) = F_X(b) - F_X(a)$$

7. $$P(X > a) = 1 - F_X(a)$$

1.4 Probability mass function (PMF)

- If X is a discrete r.v. (i.e. X has finite number of values), then probability mass function ({p_X}(x)) is defined as follows,

$$p_X(x_i) = P(X = x_i) = P(X \leq x_i) - P(X \leq x_{i-1}) = F_X(x_i) - F_X(x_{i-1})$$

1.4.1 Properties of PMF

1. $$0 \leq p_X(x_i) \leq 1$$

2. $$p_X(x) = 0, \ if \ x \neq x_i, i = 1, 2, ...$$

3.
$$\sum_i p_X(x_i) = 1$$

4.
$$F_X(x) = P(X \leq x) = \sum_{x_i < x} p_X(x_i)$$

1.5 Probability density function (PDF)

PDF is similar to PMF, but defined for continuous r.v.

$$f_X(x) = \frac{d}{dx} F_X(x)$$

1.5.1 Properties of PDF

1.
$$f_X(x) \geq 0$$

2.
$$\int_{-\infty}^{\infty} f_X(x)dx = 1$$

3.
$$P(a < X \leq b) = \int_a^b f_X(x)dx = F_X(b) - F_X(a)$$

4.
$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(x)dx$$

1.6 Descriptive statistics

In this section, some very useful descriptive statistics terms are defined, which are used to describe the random variables.

1.6.1 Mean or Expected value

The mean or expected value of a r.v. is defined as follows,

$$\mu_X = E[X] = \begin{cases} \sum_i x_i p_X(x_i) \\ \int_{-\infty}^{\infty} x f_X(x)dx \end{cases}$$

1.6.2 Moment

The n^{th} moment of a r.v. is defined as follows,

$$E[X^n] = \begin{cases} \sum\limits_i x_i^n p_X(x_i) \\ \int\limits_{-\infty}^{\infty} x^n f_X(x) dx \end{cases}$$

Note: Mean of X can be defined as 'first moment of X '.

1.6.3 Variance

- The variance of a r.v. is defined as follows,

$$\sigma_x^2 = Var(X) = E\left\{(X - E[x])^2\right\} \tag{1.1}$$

- We will get following equations after solving above equation,

$$\sigma_x^2 = \begin{cases} \sum\limits_i (x_i - \mu_X)^2 p_X(x_i) \\ \int\limits_{-\infty}^{\infty} (x - \mu_X)^2 f_X(x) dx \end{cases}$$

- Also, from equation (1.1), we have

$$Var(X) \geq 0$$

- Further, we get following equation after expanding equation (1.1),

$$Var(X) = E[X^2] - (E[X])^2$$

- The positive square root of variance is known as 'standard deviation'.

1.7 Special random variables

In this section, various important r.v. are shown. Further, there random variables are generated using python as well.

1.7.1 Uniform random variable

A r.v. is said to be uniformly distributed if it's *pdf* is given by,

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & otherwise \end{cases}$$

- Mean and variance of uniform r.v. are given as follows,

$$\mu_X = E[X] = \frac{a+b}{2}$$

$$\sigma_x^2 = Var(X) = \frac{(b-a)^2}{12}$$

- Random variable with uniform density can be created using Scipy and Numpy libraries as follows,

1.7.1.1 Uniform r.v. using Numpy

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np

>>> # generate 10000 samples between -1 and 2
>>> u = np.random.uniform(-1, 2, 10000)

>>> # plot histogram
>>> plt.hist(u)
(array([  988.,  1029.,  1018.,  1007.,   974.,  1015.,  1009.,  1004.,
         970.,   986.]), array([-0.99972473, -0.6998352 , -0.39994568, -0.10005616,  0.19983336,
        0.49972289,  0.79961241,  1.09950193,  1.39939145,  1.69928097,
        1.9991705 ]), <a list of 10 Patch objects>)

>>> # x and y lables
>>> plt.xlabel('x')
<matplotlib.text.Text object at 0xac6d1fac>
>>> plt.ylabel('number of samples')
<matplotlib.text.Text object at 0xac6e0ccc>

>>> # display plot
>>> plt.show()
```

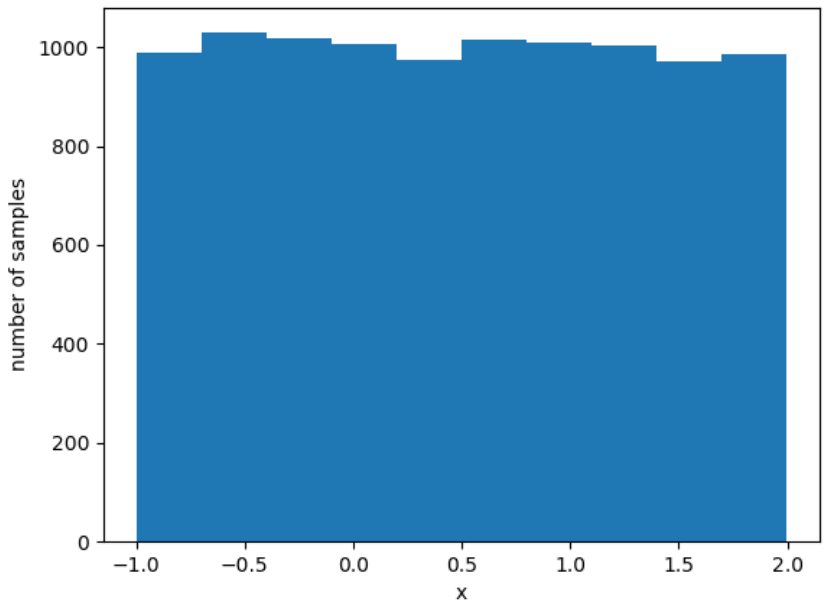

Fig. 1.1: Uniform density function using numpy

```
>>> # mean and variance of r.v. 'u'
>>> u.mean()
0.50215848652388761
>>> u.var()
0.7536921679002635
```

1.7.1.2 Uniform r.v. using scipy

- Scipy provides some additional operations as compare to numpy, as shown in this section,

```
>>> import matplotlib.pyplot as plt
>>> from scipy.stats import uniform
>>> udist = uniform(-1, 4) # start from -1 and move 4 points ahead i.e. -1 to 3
>>> u=udist.rvs(10000)     # generate 10000 samples
>>> plt.hist(u)
(array([  947.,   940.,  1005.,  1015.,   965.,  1029.,  1008.,  1042.,
        1041.,  1008.]), array([-0.99999935, -0.60007695, -0.20015456,
        0.19976783,  0.59969023, 0.99961262,  1.39953501,  1.79945741,
        2.1993798 ,  2.59930219, 2.99922458]),
        <a list of 10 Patch objects>)
>>> plt.xlabel('x')
>>> plt.ylabel('number of samples')
<matplotlib.text.Text object at 0xab81ab6c>
>>> plt.show()
>>>
```

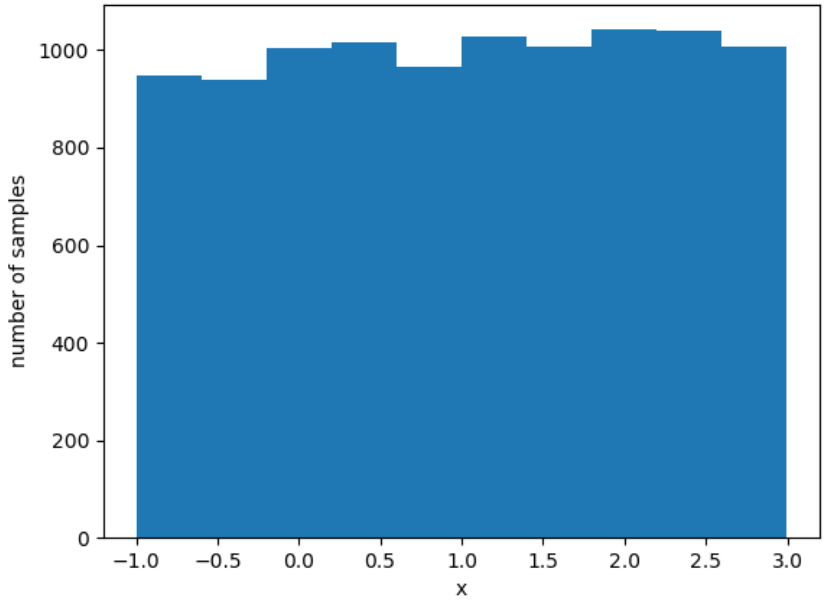

Fig. 1.2: Uniform density function using scipy

Probability density function using scipy and numpy

```
>>> import numpy as np
>>> x = np.linspace(-5, 5, 20000)
>>> plt.plot(x, udist.pdf(x))
[<matplotlib.lines.Line2D object at 0xaac8fc4c>]
>>> plt.xlabel('x')
<matplotlib.text.Text object at 0xab604a6c>
>>> plt.ylabel('pdf p(x)')
<matplotlib.text.Text object at 0xab636d0c>
>>> plt.show()
```

Statistical description using scipy

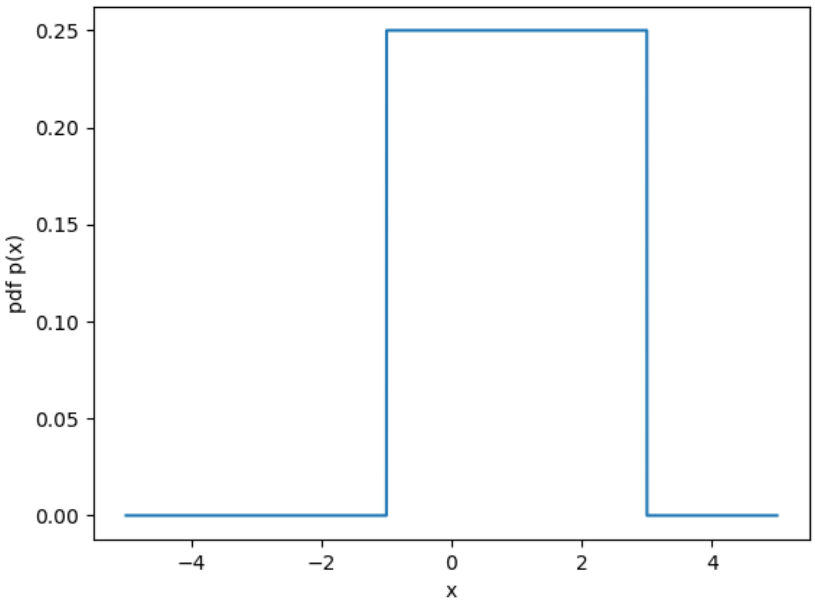

Fig. 1.3: PDF of uniform r.v. using scipy

```
>>> # statistical description
>>> mean, var, skew, kurt = udist.stats(moments='mvsk')
>>> mean
array(1.0)
>>> var
array(1.3333333333333333)
>>> skew
array(0.0)
>>> kurt
array(-1.2)


>>> # value of pdf for different x i.e. p(x)
>>> udist.pdf(0.5)
0.25
>>> udist.pdf(1.5)
0.25
>>> udist.pdf(3.5)
0.0
>>> udist.pdf(-2)
0.0

>>> # CDF values at different points
>>> udist.cdf(1)
0.5
>>> udist.cdf(5)
1.0
>>> udist.cdf(-2)
0.0
```

- Let's plot some more uniformly distributed r.v. as follows,

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
```

(continues on next page)

(continued from previous page)

```
>>> from scipy.stats import uniform

>>> start = [-3, -2, -1]
>>> start = np.array([-3, -2, -1])
>>> width = -2*start
>>> linestyles = ['-', '--', ':']
>>> x = np.linspace(-4, 4, 10000)

>>> for s, w, ls in zip(start, width, linestyles):
...     uniform_dist = uniform(s, w)
...     plt.plot(x, uniform_dist.pdf(x), ls=ls,
...              label=r'$\mu=%i , W=%i$' % (s, w))
...
[<matplotlib.lines.Line2D object at 0xb3731bac>]
[<matplotlib.lines.Line2D object at 0xab84f42c>]
[<matplotlib.lines.Line2D object at 0xab81bb0c>]
>>> plt.xlabel('$x$')
<matplotlib.text.Text object at 0xae86ec0c>
>>> plt.ylabel(r'$p(x|\mu, W)$')
<matplotlib.text.Text object at 0xab847c6c>
>>> plt.show()
>>>
```

- Above code will generate following graph,

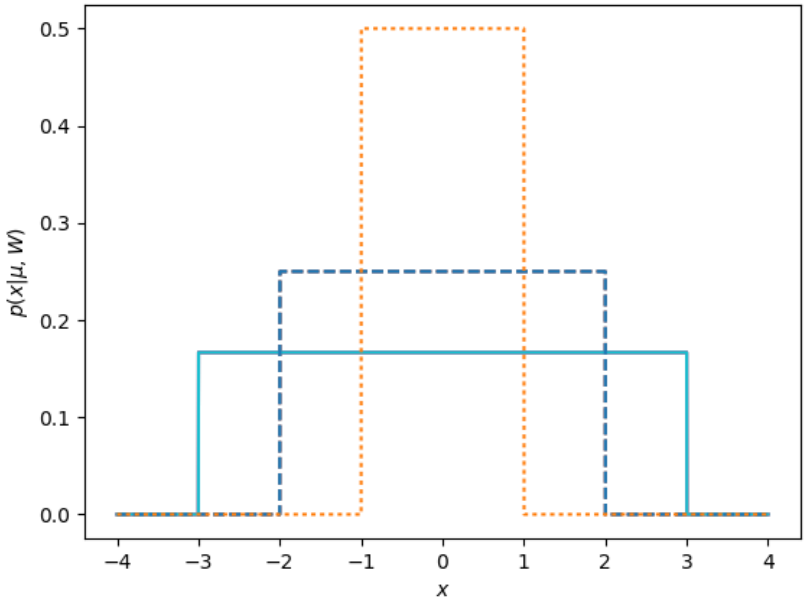

Fig. 1.4: Uniform density function for 3 different values

1.7.2 Gaussian random variable

A r.v. X is called Gaussian distributed, if it's distribution is given by,

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

- Gaussian distributed r.v. using numpy

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>>
>>> # generate 10000 samples with mean = 0, std_deviation = 1
... n = np.random.normal(0, 1, 10000)
>>>
>>> # plot histogram
... plt.hist(n)
(array([   18.,   131.,   648.,  1856.,  2853.,  2664.,  1340.,   422.,
          63.,     5.]), array([-3.71185901, -2.94333302, -2.17480703, -1.40628104, -0.63775505,
        0.13077094,  0.89929692,  1.66782291,  2.4363489 ,  3.20487489,
        3.97340088]), <a list of 10 Patch objects>)
>>>
>>> # x and y lables
... plt.xlabel('x')
<matplotlib.text.Text object at 0xaf99bfec>
>>>
>>> plt.ylabel('number of samples')
<matplotlib.text.Text object at 0xac98dd2c>
>>>
>>> # display plot
... plt.show()
```

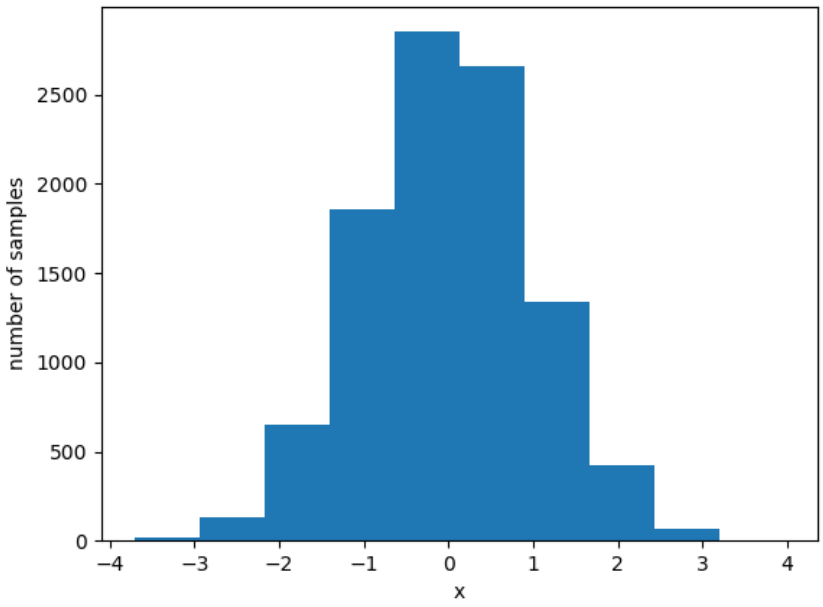

Fig. 1.5: Normal density function using Numpy

- Gaussian distributed r.v. using scipy

```
>>> import matplotlib.pyplot as plt
>>> from scipy.stats import norm
>>> ndist = norm(0, 1) # mean = 0,  std_deviation =  1
>>> n=ndist.rvs(10000)      # generate 10000 samples
>>> plt.hist(n)
(array([    6.,    36.,   293.,  1103.,  2458.,  3001.,  2067.,   827.,
         185.,    24.]), array([-4.15111973, -3.3743863 , -2.59765287, -1.82091944, -1.04418601,
       -0.26745258,  0.50928085,  1.28601427,  2.0627477 ,  2.83948113,
        3.61621456]), <a list of 10 Patch objects>)
>>> plt.show()
```

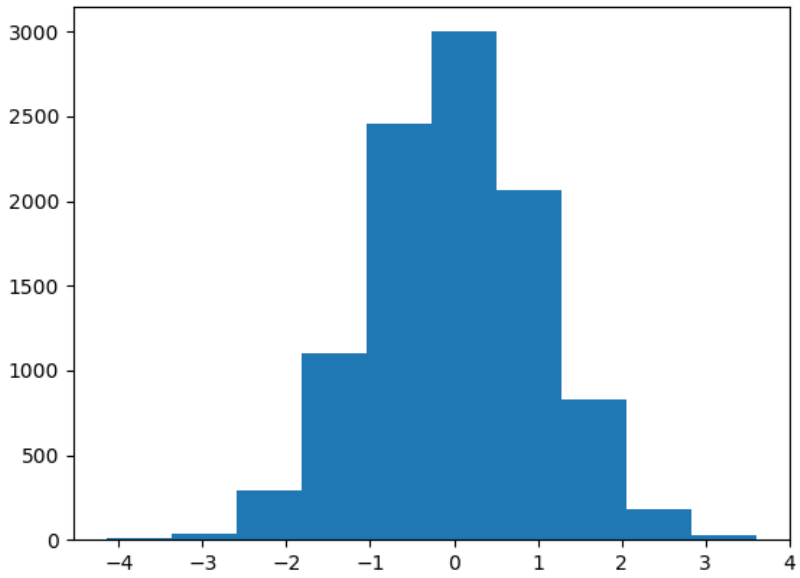

Fig. 1.6: Normal density function using Scipy

- PDF of Gaussian distributed r.v. for three different values of standard deviations,

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> from scipy.stats import norm
>>>
>>> mu = 0  # mean = 0
>>> sigma_values = [0.5, 1.0, 2.0]  # diff std_deviation values
>>>
>>> linestyles = ['-', '--', ':']
>>> x = np.linspace(-5, 5, 10000)
>>>
>>> for sigma, ls in zip(sigma_values, linestyles):
...     # create a gaussian / normal distribution
...     norm_dist = norm(mu, sigma)
...     plt.plot(x, norm_dist.pdf(x), ls=ls, c='black',
...              label=r'$\mu=%i ,\ \sigma=%.1f$' % (mu, sigma))
...
[<matplotlib.lines.Line2D object at 0xab80c26c>]
[<matplotlib.lines.Line2D object at 0xab80cf2c>]
[<matplotlib.lines.Line2D object at 0xab8178cc>]
>>>
>>> plt.xlabel('$x$')
<matplotlib.text.Text object at 0xae881aec>
>>>
>>> plt.ylabel(r'$p(x|\mu, W)$')
<matplotlib.text.Text object at 0xab854b6c>
>>>
>>> plt.show()
```

- Above code will generate following graph,

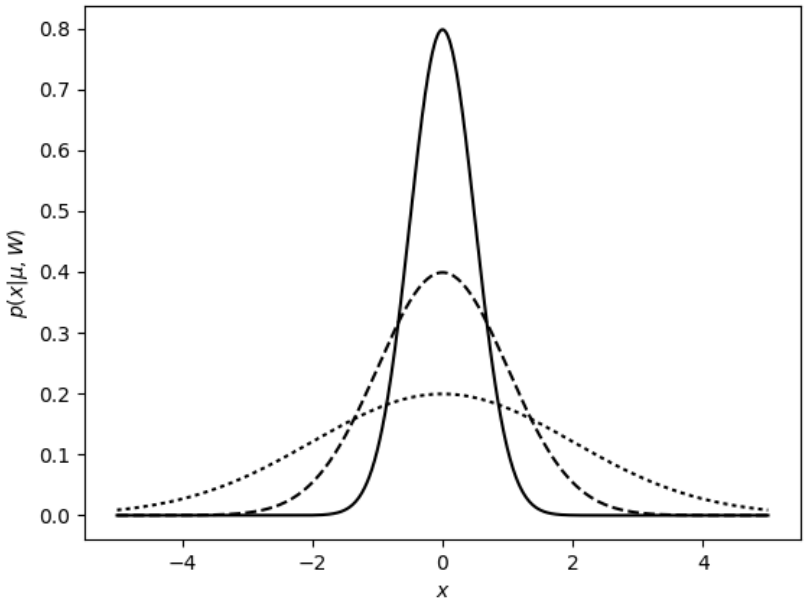

Fig. 1.7: Norm density function for 3 different values

Chapter 2

Indices and tables

- genindex

Index