Pattern Recognition und Machine Learning
Hochschule München
Sommersemester 2023
Prof. Dr.-Ing. Claudius Schnörr

Blatt 1

F. Freter, E. Kirchberger, S. Symhoven & J. Wustl

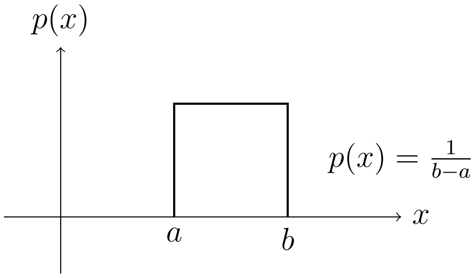
Aufgabe 2a: Statistik

1. Dichte, Erwartungswert, Varianz und Kovarianz

   a) Die Dichte der gleichverteilten 1D-Zufallsvariable x im Intervall $[a, b] \subset \mathbb{R}$ ist definiert als:

   $$p(x) = \begin{cases} \frac{1}{b-a} & \text{für } a \leq x \leq b \\ 0 & \text{sonst} \end{cases}$$

   und kann wie folgt skizziert werden:

   Die Dichte auf dem Intervall $[a, b]$ hat den Wert $\frac{1}{b-a}$.

   b) Erwartungswert:

   $$\begin{aligned} E[X] &= \int_{-\infty}^{\infty} x\, p(x)\, dx \\ &= \int_a^b x \cdot \frac{1}{b-a}\, dx \\ &= \frac{1}{b-a} \cdot \left[\frac{x^2}{2}\right]_a^b \\ &= \frac{b^2 - a^2}{2(b-a)} \\ &= \frac{a+b}{2} \end{aligned}$$

   c) Varianz

$$
\begin{aligned}
Var(X) &= E[(x-\mu_x)^2] \\
&= \int_a^b (x-\mu_x)^2\, p(x)\, dx \\
&= \int_a^b (x - \frac{a+b}{2})^2 \cdot \frac{1}{b-a}\, dx \\
&= \left[\frac{1}{3}(x-\frac{a+b}{2})^3 \cdot \frac{1}{b-a}\right]_a^b \\
&= \frac{1}{3(b-a)} \left((b-\frac{a+b}{2})^3 - (a-\frac{a+b}{2})^3 \right) \\
&= \frac{1}{3(b-a)} \left((\frac{b-a}{2})^3 + (\frac{b-a}{2})^3 \right) \\
&= \frac{1}{3(b-a)} 2(\frac{b-a}{2})^3 \\
&= \frac{2(\frac{b-a}{2})^3}{3(b-a)} \\
&= \frac{2}{3}\frac{(\frac{b-a}{2})^3}{(b-a)} \\
&= \frac{1}{12} \cdot (b-a)^2
\end{aligned}
$$

2. Sample Mean und Kovarianzmatrix

a)

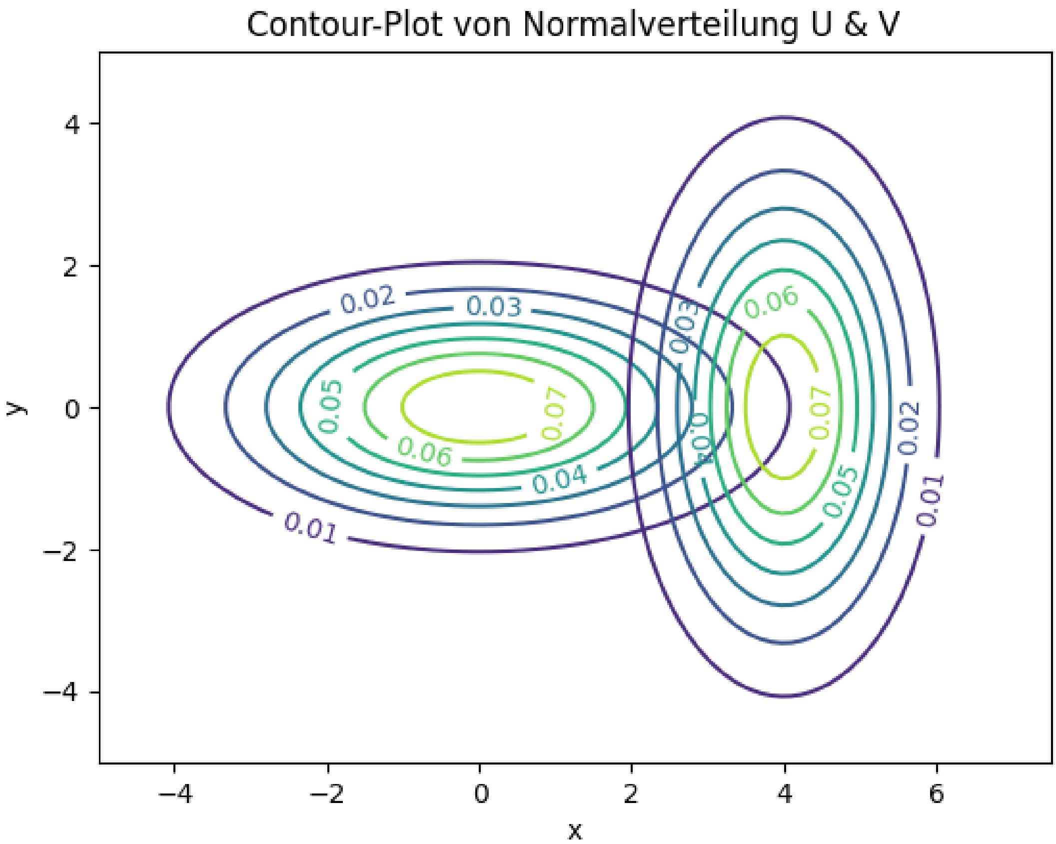
$$
\begin{aligned}
\mu_X &= \frac{1}{n}\sum_i X_i \\
&= \frac{1}{4}\sum_{i=1}^4 X_i \\
&= \frac{1}{4} * \left[\begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \begin{pmatrix} -5 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right] \\
&= \frac{1}{4} \begin{pmatrix} -4 \\ 4 \end{pmatrix} \\
&= \begin{pmatrix} -1 \\ 1 \end{pmatrix}
\end{aligned}
$$

b)

$$\begin{aligned}
\sum_{\mathbf{xx}} &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mu_{\mathbf{x}})(\mathbf{x}_i - \mu_{\mathbf{x}})^T \\
&= \frac{1}{4}\left[\begin{pmatrix}2\\1\end{pmatrix}\begin{pmatrix}2 & 1\end{pmatrix} + \begin{pmatrix}0\\2\end{pmatrix}\begin{pmatrix}0 & 2\end{pmatrix} + \begin{pmatrix}-4\\0\end{pmatrix}\begin{pmatrix}-4 & 0\end{pmatrix} + \begin{pmatrix}2\\1\end{pmatrix}\begin{pmatrix}2 & 1\end{pmatrix}\right] \\
&= \frac{1}{4}\left[\begin{pmatrix}4 & 2\\2 & 1\end{pmatrix} + \begin{pmatrix}0 & 0\\0 & 4\end{pmatrix} + \begin{pmatrix}16 & 0\\0 & 0\end{pmatrix} + \begin{pmatrix}4 & 2\\2 & 1\end{pmatrix}\right] \\
&= \frac{1}{4}\begin{pmatrix}24 & 4\\4 & 6\end{pmatrix} \\
&= \begin{pmatrix}6 & 1\\1 & 1.5\end{pmatrix}
\end{aligned}$$

3. Multivariate Normalverteilung

a) Die beiden Verteilungen der Zufallsvariablen **u** und **v** liegen wie folgt im x/y-Koordinatensystem. Die Verteilung der Zufallsvariable **u** entspricht der horizontalen Ellipse links. Diese hat den Mittelpunkt im Ursprung, eine Varianz von 4 in x-Richtung und eine Varianz von 2 in y-Richtung. Die Verteilung der Zufallsvariable **v** entspricht der vertikalen Ellipse rechts. Diese hat den Mittelpunkt um 4 in x-Richtung verschoben, eine Varianz von 2 in x-Richtung und eine Varianz von 4 in y-Richtung:



b) Die Trennfunktion muss eine Funktion sein, die die Grenze zwischen den beiden Verteilungen darstellt.

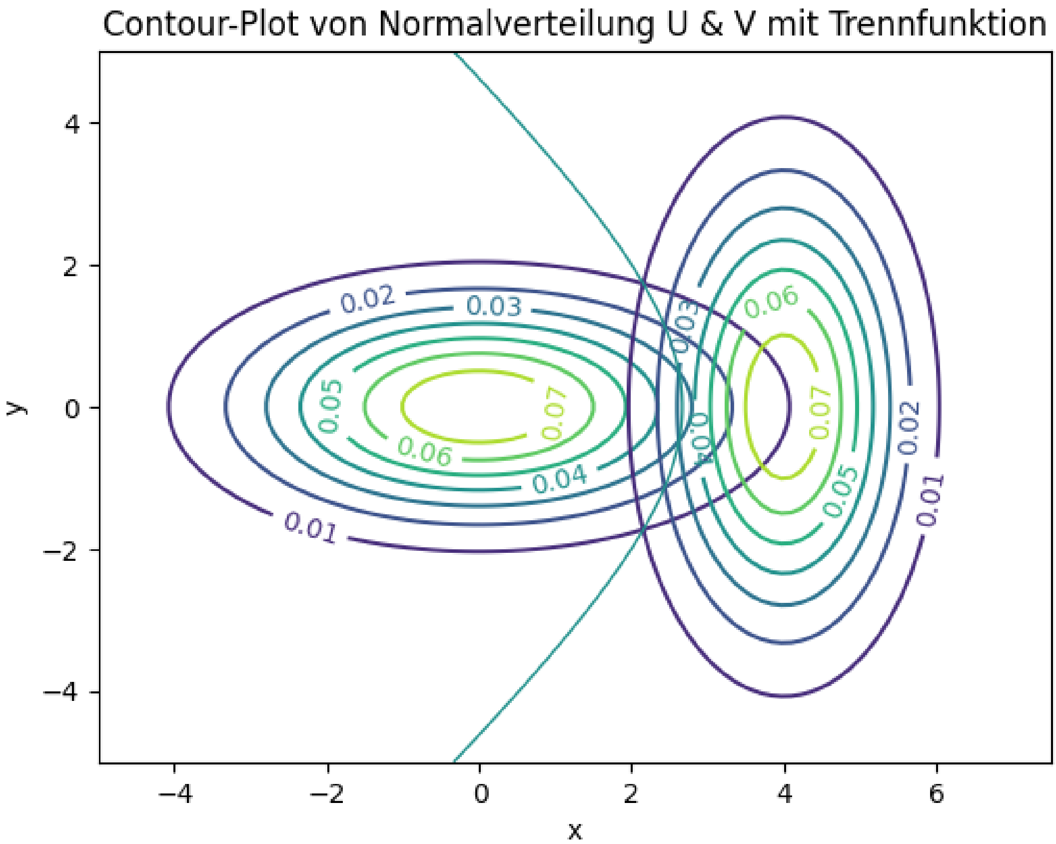
Bei einer Bayes-Entscheidungsregel lautet die Trennfunktion:

$$T(x) = \log\left(\frac{p(x|C_1)}{p(x|C_2)}\right) + \log\left(\frac{p(C_2)}{p(C_1)}\right) \tag{1}$$

mit $p(x|C_i)$ als der Wahrscheinlichkeit, dass x in der Klasse C_i liegt, und $p(C_i)$ als der A-priori-Wahrscheinlichkeit der Klasse C_i.
Da in diesem Fall die A-priori-Wahrscheinlichkeiten der beiden Verteilungen gleich sind, vereinfacht sich die Gleichung zu:

$$T(x) = \log\left(\frac{p(x|U)}{p(x|V)}\right) \tag{2}$$

Die Trennfunktion sieht dann wie folgt aus:



4. Entropie einer diskreten Verteilung

a)

$$I_1 = -ln(P(1)) = -ln(0.2) \approx 1.6094$$

b)

$$I_2 = -ln(P(2)) = -ln(0.8) \approx 0.2231$$

c)

$$\begin{aligned} H(X) &= -\sum P(x) \cdot \ln(P(x)) \\ &= -[P(1) \cdot \ln(P(1)) + P(2) \cdot \ln(P(2))] \\ &= -[0,2 \cdot \ln(0,2) + 0,8 \cdot \ln(0,8)] \\ &\approx 0,5004 \end{aligned}$$

d)

$$\begin{aligned} H(X) &= -\sum P(x) \cdot \ln(P(x)) \\ &= -[P(1) \cdot \ln(P(1)) + P(2) \cdot \ln(P(2))] \\ &= -[0,1 \cdot \ln(0,1) + 0,9 \cdot \ln(0,9)] \\ &\approx 0,4689 \end{aligned}$$

Die Entropie ist etwas niedriger als zuvor, was darauf hindeutet, dass die Verteilung jetzt etwas vorhersehbarer ist. Das macht Sinn, da die Wahrscheinlichkeit für Ereignis P_2 gestiegen und für P_1 gefallen ist. Dass P_2 eintritt, ist nun also sicherer als vorher und die Entropie somit geringer, da das Ergebnis vorhersehbarer wird.

Aufgabe 2b: Statistik

1 Imports

[5]:
```
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from scipy.stats import multivariate_normal
```

2 Berechnen des Mittelwertsvektors und der Kovarianzmatrix

[6]:
```
x1 = np.array([[1], [2]])
x2 = np.array([[-1], [-1]])
x3 = np.array([[-5], [1]])
x4 = np.array([[1], [2]])

# Stichprobenmatrix erstellen
X = np.hstack((x1, x2, x3, x4))

X_mean = np.mean(X,  axis=1, keepdims=True)

print(f'Mittelwertsvektor:\n {X_mean}')
```

```
Mittelwertsvektor:
 [[-1.]
 [ 1.]]
```

[7]:
```
C =  (1 / X.shape[1]) * np.dot((X - X_mean), (X - X_mean).T)

print(f'Auto Kovarianzmatrix:\n {C}')
```

```
Auto Kovarianzmatrix:
 [[6.   1. ]
 [1.   1.5]]
```

3 Subplot Funktion für 3D-, Contour- & 2D-Sample-Plot

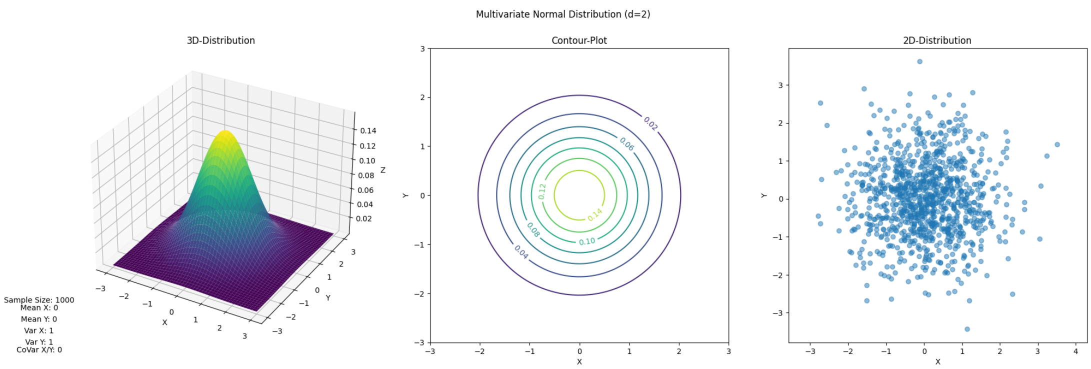
[8]:
```
def generate_plts(mean, cov, sample_size):
    # Gitter erstellen und anhand der Mean/Cov automatisch scalen
    cov_max = max(cov[0][0], cov[1][1])
    mean_max = max(mean[0], mean[1])
    x_min = y_min = mean_max - 3 * np.sqrt(cov_max)
    x_max = y_max = mean_max + 3 * np.sqrt(cov_max)

    x, y = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min,␣
↪y_max, 100))
    pos = np.dstack((x, y))
```

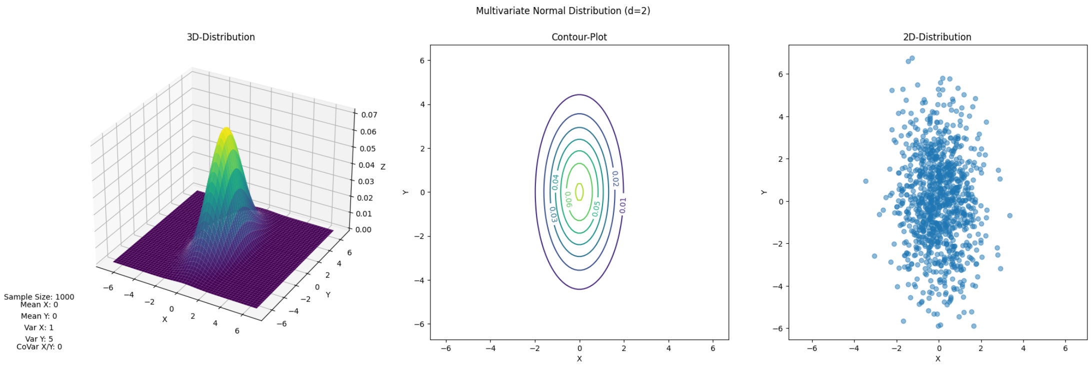
```python
    # Normalverteilungswerte berechnen
    rv = multivariate_normal(mean, cov)
    z = rv.pdf(pos)

    # Zufällige Samples generieren
    samples = rv.rvs(sample_size)

    fig = plt.figure(figsize=(24,7))

    # 3D Plot
    ax1 = fig.add_subplot(1, 3, 1, projection='3d')
    ax1.set_title('3D-Distribution')
    ax1.plot_surface(x, y, z, cmap='viridis', linewidth=0)
    ax1.set_xlabel('X')
    ax1.set_ylabel('Y')
    ax1.set_zlabel('Z')

    # Contour Plot
    ax2 = fig.add_subplot(1, 3, 2)
    ax2.set_title('Contour-Plot')
    ax2.set_xlabel('X')
    ax2.set_ylabel('Y')
    cp = ax2.contour(x, y, z)
    ax2.clabel(cp, inline=True, fontsize=10)

    # 2D-Distribution of random sample
    ax3 = fig.add_subplot(1, 3, 3)
    ax3.set_title('2D-Distribution')
    ax3.set_xlabel('X')
    ax3.set_ylabel('Y')
    ax3.scatter(samples[:, 0], samples[:, 1], alpha=0.5)

    plt.axis('equal')

    fig.suptitle('Multivariate Normal Distribution (d=2)')

    fig.text(0.1, 0.22, f'Sample Size: {sample_size}', horizontalalignment='center', verticalalignment='center')
    fig.text(0.1, 0.2, f'Mean X: {mean[0]}', horizontalalignment='center', verticalalignment='center')
    fig.text(0.1, 0.17, f'Mean Y: {mean[1]}', horizontalalignment='center', verticalalignment='center')
    fig.text(0.1, 0.14, f'Var X: {cov[0][0]}', horizontalalignment='center', verticalalignment='center')
    fig.text(0.1, 0.11, f'Var Y: {cov[1][1]}', horizontalalignment='center', verticalalignment='center')
```

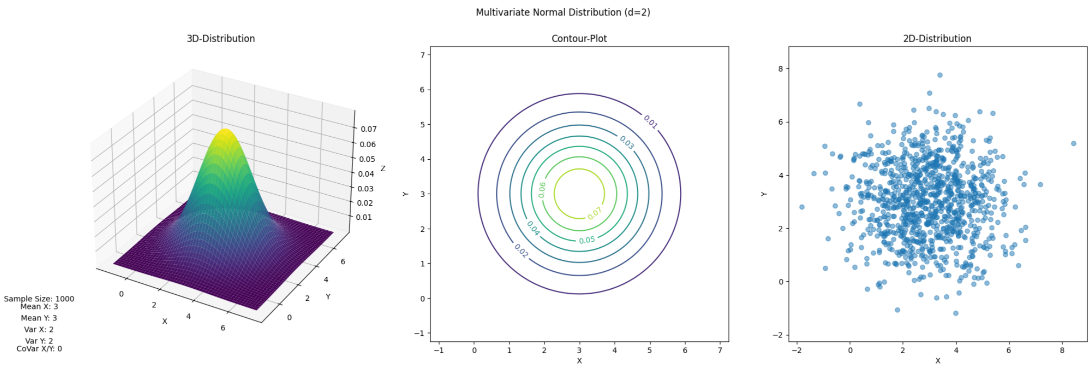
```
    fig.text(0.1, 0.09, f'CoVar X/Y: {cov[0][1]}', horizontalalignment='center', verticalalignment='center')
    plt.show()
```

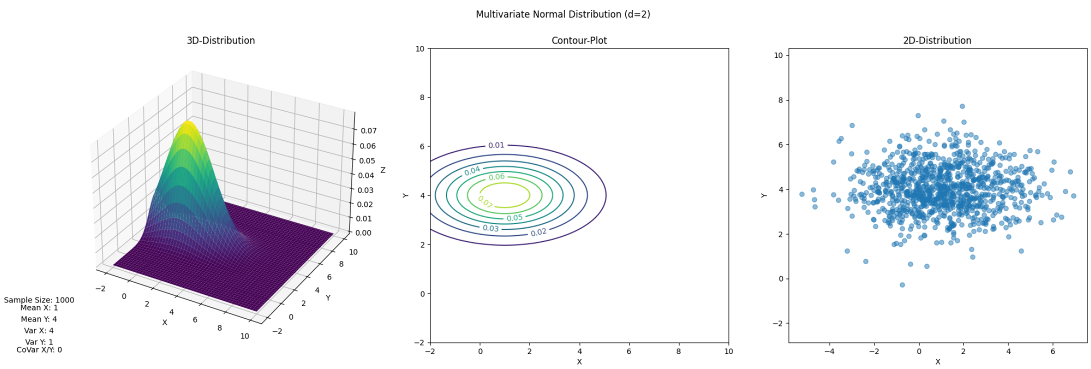
[11]:

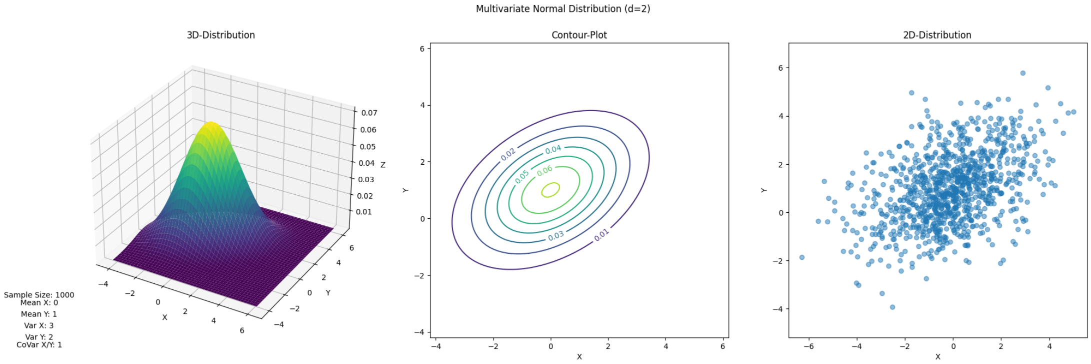
```
sample_size = 1000

# Mittelwert, Kovarianzmatrix und Sample Size definieren
mean = [0, 0]
cov = [[1, 0], [0, 1]]
generate_plts(mean, cov, sample_size)

mean = [0, 0]
cov = [[1, 0], [0, 5]]
generate_plts(mean, cov, sample_size)

mean = [3, 3]
cov = [[2, 0], [0, 2]]
generate_plts(mean, cov, sample_size)

mean = [1, 4]
cov = [[4, 0], [0, 1]]
generate_plts(mean, cov, sample_size)

mean = [0, 1]
cov = [[3, 1], [1, 2]]
generate_plts(mean, cov, sample_size)
```




Multivariate Normal Distribution (d=2)
3D-Distribution
Contour-Plot
2D-Distribution
Sample Size: 1000
Mean X: 0
Mean Y: 0
Var X: 1
Var Y: 5
CoVar X/Y: 0


Multivariate Normal Distribution (d=2)
3D-Distribution
Contour-Plot
2D-Distribution
Sample Size: 1000
Mean X: 3
Mean Y: 3
Var X: 2
Var Y: 2
CoVar X/Y: 0

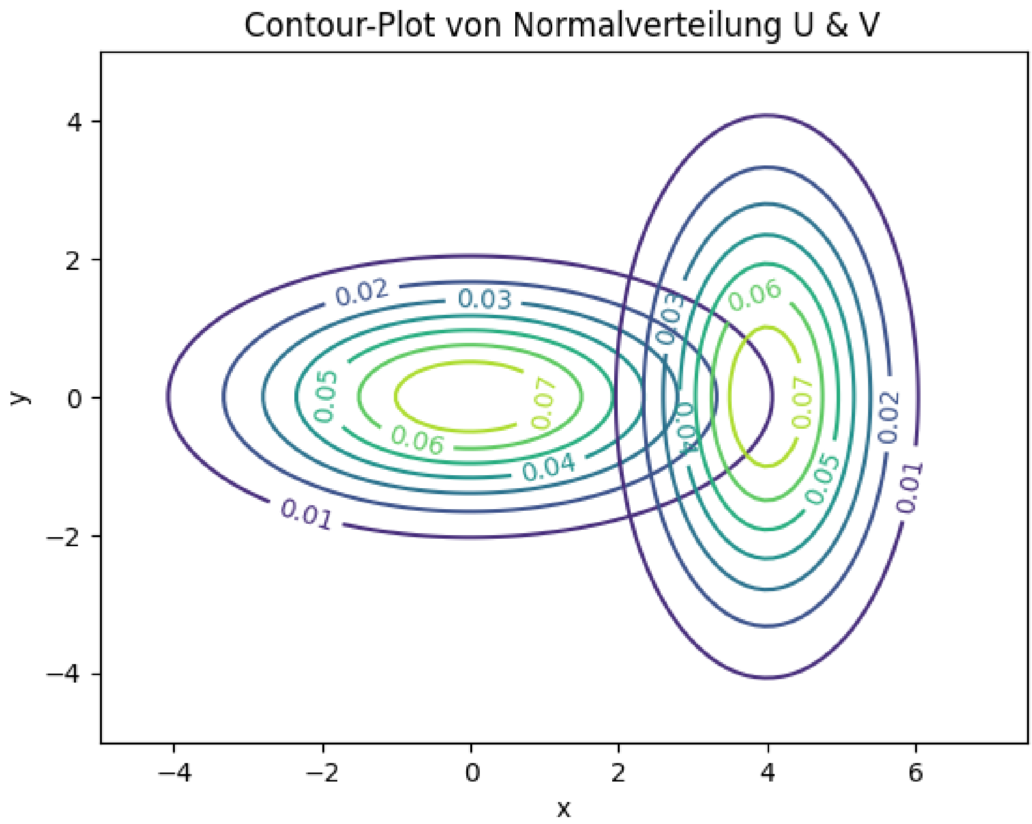

Multivariate Normal Distribution (d=2)
3D-Distribution
Contour-Plot
2D-Distribution
Sample Size: 1000
Mean X: 1
Mean Y: 4
Var X: 4
Var Y: 1
CoVar X/Y: 0



4 Lage von U & V

[105]:
```python
# Parameter für die erste Normalverteilung
mean_u = [0, 0]
cov_u = [[4, 0], [0, 1]]

# Parameter für die zweite Normalverteilung
mean_v = [4, 0]
cov_v = [[1, 0], [0, 4]]

# Gitter erstellen
x, y = np.meshgrid(np.linspace(-5, 7.5, 100), np.linspace(-5, 5, 100))
pos = np.dstack((x, y))

# PDFs der Normalverteilungen berechnen
rv_u = multivariate_normal(mean_u, cov_u)
z_u = rv_u.pdf(pos)

rv_v = multivariate_normal(mean_v, cov_v)
z_v = rv_v.pdf(pos)

# Contour-Plots erstellen
fig, ax = plt.subplots()
cp_u = ax.contour(x, y, z_u)
ax.clabel(cp_u, inline=True, fontsize=10)
cp_v = ax.contour(x, y, z_v)
ax.clabel(cp_v, inline=True, fontsize=10)

# Titel und Achsenbeschriftungen hinzufügen
ax.set_title('Contour-Plot von Normalverteilung U & V')
ax.set_xlabel('x')
ax.set_ylabel('y')
```

```
# Schaubild anzeigen
plt.show()
```



5 Trennfunktion für U & V

[106]:
```
# Trennfunktion berechnen
z_trenn = np.log(rv_u.pdf(pos) / rv_v.pdf(pos))


# Contour-Plots erstellen
fig, ax = plt.subplots()
cp_u = ax.contour(x, y, z_u)
ax.clabel(cp_u, inline=True, fontsize=10)
cp_v = ax.contour(x, y, z_v)
ax.clabel(cp_v, inline=True, fontsize=10)
cp_trenn = ax.contour(x, y, z_trenn, levels=0, linewidths=1)

# Titel und Achsenbeschriftungen hinzufügen
```

```python
ax.set_title('Contour-Plot von Normalverteilung U & V mit Trennfunktion')
ax.set_xlabel('x')
ax.set_ylabel('y')


# Schaubild anzeigen
plt.show()
```