hw1

April 5, 2022

[1]:
```
import numpy as np
import pandas as pd
pd.options.display.float_format = "{:.2g}".format
from scipy.stats import norm, multivariate_normal
from scipy.stats.mvn import mvnun
from scipy.optimize import bisect
```

```
/var/folders/rl/jll8zb7n49d7ns3jcsyf8g4h0000gn/T/ipykernel_87857/1895691198.py:5
: DeprecationWarning: Please use `mvnun` from the `scipy.stats` namespace, the
`scipy.stats.mvn` namespace is deprecated.
  from scipy.stats.mvn import mvnun
```

[2]:
```
def disp_arr(arr, index_1=True):
    s = arr.shape
    display(pd.DataFrame(arr, index=range(s[0]), columns=range(s[1]))
            .iloc[int(index_1):, int(index_1):])

PD = np.zeros((4,4))
for i in range(1,4):
    for j in range(1,4):
        if i==j: PD[i,j] = 0.1*i
        else: PD[i,j] = 0.06

print("PD Matrix")
disp_arr(PD)
```

```
PD Matrix
```

```
     1    2    3
1  0.1 0.06 0.06
2 0.06  0.2 0.06
3 0.06 0.06  0.3
```

1 Question 1

Find the three values of correlation…

$$z_n = \Phi^{-1}(PD_n)$$
$$PDJ_{i,j} = \int_{-\infty}^{z_i}\int_{-\infty}^{z_j} \phi(Z_i, Z_j, \rho_{i,j})dZ_i dZ_j$$
$$\hat{\rho}_{i,j} = \underset{\rho_{i,j}}{\arg\min}\left\| \int_{-\infty}^{z_i}\int_{-\infty}^{z_j} \phi(Z_i, Z_j, \rho_{i,j})dZ_i dZ_j - PDJ_{i,j}\right\|_2$$

[3]:
```
def joint_from_copula(z_i, z_j, rho_ij):
    """Uses Z-scores and Corr to determine the joint probability of default in␣
 ↪a 2-Copula"""
    copula = multivariate_normal(mean=np.array([0,0]), cov=np.array([[1,␣
 ↪rho_ij],[rho_ij,1]]))
    return copula.cdf(np.array([z_i, z_j]))

def corr_from_joint(pd_i, pd_j, joint):
    """Computes pairwise Corr from pairwise PDs"""
    z_i = norm.ppf(pd_i)
    z_j = norm.ppf(pd_j)

    if np.allclose(pd_i*pd_j, joint): return 0
    elif pd_i*pd_j > joint: lo, hi = -0.999, 0
    else: lo, hi = 0, 0.999

    return bisect(lambda x: (joint_from_copula(z_i, z_j, x)-joint), lo, hi)

Corr = np.ones(PD.shape)

for i in range(1,4):
    for j in range(1,4):
        if i==j: continue
        else: Corr[i,j] = corr_from_joint(PD[i,i], PD[j,j], PD[i,j])

print("Correlation Matrix")
disp_arr(np.round(Corr,4))
```

```
Correlation Matrix

     1   2    3
1    1 0.6 0.43
2  0.6   1    0
3 0.43   0    1
```

…find the three values of default correlation…

$$\begin{aligned} Dcorr[D_1, D_2] &= \frac{Cov[D_1, D_2]}{\sqrt{Var[D_1]Var[D_2]}} \\ &= \frac{PDJ - PD_1PD_2}{\sqrt{PD_1(1-PD_1)PD_2(1-PD_2)}} \end{aligned}$$

[4]:
```
def dcorr(pd_i, pd_j, joint):
    return (joint - pd_i*pd_j) / (pd_i*(1-pd_i)*pd_j*(1-pd_j))**0.5

DCorr = np.ones((4,4))

for i in range(1,4):
    for j in range(1,4):
        if i==j: continue
        else: DCorr[i,j] = dcorr(PD[i,i], PD[j,j], PD[i,j])

print("Default Correlation")
disp_arr(DCorr)
```

```
Default Correlation

     1        2        3
1    1     0.33     0.22
2 0.33        1 -7.6e-17
3 0.22 -7.6e-17        1
```

2 Question 2

Suppose that three firms each have PD = 0.10...

State the three values of PDJ. State the range of possible values for the probability that all three of the firms default. State the probability that all three default under the Gauss copula.

[5]:
```
zs = norm.ppf([0.1]*4)
Corr = np.array([[1,1  ,1  ,1  ],
                 [1,1  ,0.4,0.5],
                 [1,0.4,1  ,0.6],
                 [1,0.5,0.6,1  ]])

PD = np.ones(Corr.shape)*0.1

for i in range(1,4):
    for j in range(1,4):
        if i==j: continue
        else: PD[i,j] = joint_from_copula(zs[i], zs[j], Corr[i,j])

print("PD Matrix")
disp_arr(np.round(PD,4))
```

```
PD Matrix

      1     2     3
1   0.1 0.027 0.032
2 0.027   0.1 0.039
3 0.032 0.039   0.1
```

...bounds on probability that all three of the firms default.

$$PDJ_{1,2,3} \in [0, 0.029]$$

[6]:
```python
def all_from_copula(zs, cov):
    """Uses Z-scores and Corr to determine the joint probability of default in␣
 ↪an n-Copula"""
    n = zs.shape[0]
    copula = multivariate_normal(mean=np.array([0]*n), cov=cov)
    return copula.cdf(zs)

zs = zs[1:]

pd_123 = all_from_copula(zs, Corr[1:, 1:])
print(f"Probability that all 3 default: {pd_123:.2g}")
```

```
Probability that all 3 default: 0.016
```

3 Question 3

Suppose a firm rated A has correlation 0.4 with a firm rated B. In the following period, a Firm A remains rated A with prob = 0.5, and so forth...

In your answer file, create a three-by-three grid such as here...

[7]:
```python
TP = np.array([[0.5,0.4,0.1],
               [0.3,0.5,0.2]])

TP_c = np.array([[1.0,0.5,0.1,0],
                 [1.0,0.7,0.2,0]])

zTP = norm.ppf(TP_c)

rho_AB = 0.4
Cor = np.array([[1,rho_AB],[rho_AB,1]])

zTPA = zTP[0][::-1]
zTPB = zTP[1][::-1]

zA = {state: (zTPA[i],zTPA[i+1]) for state, i in zip(["D","B","A"],[0,1,2])}
zB = {state: (zTPB[i],zTPB[i+1]) for state, i in zip(["D","B","A"],[0,1,2])}
```

[8]:
```python
Trans = pd.DataFrame(index=['A','B','D'], columns=['D','B','A'])

DEBUG = False
for i, r in enumerate(Trans.index):
    for j, c in enumerate(Trans.columns):
        # integration bounds
        bounds = np.array([zA[c], zB[r]])
        if DEBUG: print(f"B={r},A={c}\n",bounds)

        Trans.loc[r,c] = mvnun(bounds[:,0], bounds[:,1], np.zeros(2), Cor)[0]

assert np.allclose(np.sum(Trans.sum()),1.0) # Transition Probabilities sum to 1
print("The transition probabilities sum to 1.")
Trans
```

```
The transition probabilities sum to 1.
```

[8]:

```
        D     B     A
A  0.0094 0.084  0.21
B   0.047  0.21  0.24
D   0.044   0.1 0.055
```

4 Question 4

Suppose that four firms have PDs equal to 1%, 2%, 3%, and 4% and the probability that any given pair defaults equals 0.1%. What is the matrix of correlations? Explain why the defaults of the four firms can or cannot be connected by a Gauss copula.

[9]:

```
PD = np.ones((5,5))

for i in range(1,5):
    for j in range(1,5):
        if i==j: PD[i,j] = i*0.01
        else: PD[i,j] = 0.001

print("PD Matrix")
disp_arr(PD)
```

```
PD Matrix

      1     2     3     4
1  0.01 0.001 0.001 0.001
2 0.001  0.02 0.001 0.001
3 0.001 0.001  0.03 0.001
4 0.001 0.001 0.001  0.04
```

[10]:

```
Corr = np.ones(PD.shape)

for i in range(1,5):
    for j in range(1,5):
        if i==j: continue
        else: Corr[i,j] = corr_from_joint(PD[i,i], PD[j,j], PD[i,j])
```

```
print("Correlation Matrix")
disp_arr(np.round(Corr,4))

eig_vals, eig_vecs = np.linalg.eig(Corr[1:,1:])

assert (eig_vals >= np.zeros(eig_vals.shape)).all()
print("All the Correlation Matrix's Eigenvalues are >= 0. Therefore, the matrix␣
 ↪is Positive Semidefinite.")

assert np.linalg.det(Corr[1:,1:]) >= 0
print("The Correlation Matrix's determinant is non-negative.")
```

```
Correlation Matrix

     1     2      3      4
1    1  0.31   0.24   0.18
2 0.31     1    0.1  0.044
3 0.24   0.1      1 -0.036
4 0.18 0.044 -0.036      1

All the Correlation Matrix's Eigenvalues are >= 0. Therefore, the matrix is
Positive Semidefinite.
The Correlation Matrix's determinant is non-negative.
```

Analysis: Correlation Matrix The correlation matrix using the default probabilities appears valid. It is symmetric and positive semidefinite. Furthermore, the determinant is non-negative.

```
[11]: zs = norm.ppf(np.diag(PD))
PD = np.ones(Corr.shape)
for i in range(PD.shape[0]): PD[i,i] = 0.01*i

for i in range(1,5):
    for j in range(1,5):
        if i==j: continue
        else: PD[i,j] = joint_from_copula(zs[i], zs[j], Corr[i,j]-0.001)

print("PD Matrix")
disp_arr(np.round(PD,4))
```

```
PD Matrix

      1     2     3     4
1  0.01 0.001 0.001 0.001
2 0.001  0.02 0.001 0.001
3 0.001 0.001  0.03 0.001
4 0.001 0.001 0.001  0.04
```

Analysis: Reversion to Default Probabilities The correlation matrix successfully reverts back to the Probability of Default Matrix. Because:

$$PD_{i,i} \in [\underbrace{0}_{\rho_{i,j}=-1}, \underbrace{\min(PD_i, PD_j)}_{\rho_{i,j}=1}]$$

A naive check of the Probability of Default Matrix appears to indicate that it may be valid.

Analysis: Gaussian Copula The Gaussian Copula can adaquately address most distributions with correlated default probabilities. Changing correlation between each marginal distribution can be visualized as expansion while rotating of the joint distribution ($\rho \in [1 \to 0]$) then contraction while rotating ($\rho \in [0 \to -1]$).

[ ]:

```

```