SamplingMethod

April 10, 2022

[1]:

```
import jax.numpy as jnp
import jax.random as random
from scipy.spatial.distance import cdist                                # cdist is␣
 ↪used for generating random covariance matrix
```

[2]:

```
from sklearn.preprocessing import PowerTransformer                      # For␣
 ↪Converting Uniform Distribution to Normal Distribution
```

[3]:

```
# Setting key for randomization in JAX
key = random.PRNGKey(3)
```

1 Sampling Function For MVN

[16]:

```
def mvn_samples(mu, cov, n_samples=10):
    """
    Arguments :
        mu     -- mean of shape (dimension)
        cov    -- covariance matrix of shaoe( dimension,dimension)
        n_sample -- default to 10
    output :
        samples = mvn_sample of shape (dimensions,n_samples)

    """
    pt = PowerTransformer()
    key = random.PRNGKey(3)
    d = cov.shape[0]   # number of dimensions

    U = random.uniform(key,[d,n_samples])

    U = pt.fit_transform(U.T)


    L = jnp.linalg.cholesky(cov)
    samples = L.dot(U.T) + mu

    return samples
```

2 Evaluating Sampling Function

[17]:

```
dimension = 3          # Change it if you want
n_samples = 100         # Change it if you want

# Mean Matrix
mu = random.uniform(key,(1,dimension))

# Covariance matrix
var_matrix = random.uniform(key,[dimension,1])
cov = jnp.exp(-cdist(var_matrix , var_matrix, "euclidean")) + 1e-7*jnp.
 ↪eye(dimension)
```

[18]:

```
# Generating Samples

X = mvn_samples(mu.T, cov, n_samples = n_samples)
```

[19]:

```
X.shape
```

[19]:

```
(3, 100)
```

[20]:

```
# print(X)
```

[20]:

```

```

3 Checking Means of MVN and Generated Samples

[21]:

```
print("Random Uniform Mean     : ",mu)
print("Samples Generated Mean   : ",X.mean(axis=1))
```

```
Random Uniform Mean     :  [[0.23853767 0.02100694 0.60624325]]
Samples Generated Mean  :  [0.23853767 0.02100694 0.6062432 ]
```

[30]:

```
print("Random  Uniform   Cov      : ")
print(cov)
print()
print("Samples Generated Cov   : ")
print(jnp.cov(X))
```

```
Random  Uniform   Cov      :
[[1.0000001  0.8045029  0.692321  ]
 [0.8045029  1.0000001  0.55697423]
 [0.692321   0.55697423 1.0000001 ]]
```

```
Samples Generated Cov   :
[[1.0101012  0.8934811  0.55802715]
 [0.8934811  1.1401922  0.50697994]
 [0.55802715 0.50697994 0.81446916]]
```

Minor difference is Negligible.

Built in Sampling Distribution also has minor difference for means and covariance.

[ ]:

```
```

3.1 Generating Multiple Samples and comparing Means and Covariance

[34]:

```
dims = [
        [2,10],
        [3,50],
        [4,100]
      ]

for i in range(3):
    dimension = dims[i][0]
    n_samples = dims[i][1]
    mu = random.uniform(key,(1,dimension))

    # Covariance matrix
    var_matrix = random.uniform(key,[dimension,1])
    cov = jnp.exp(-cdist(var_matrix , var_matrix, "euclidean")) + 1e-7*jnp.
 ↪eye(dimension)


    X = mvn_samples(mu.T, cov, n_samples = n_samples)

    print("Sample : {}".format(i+1))
    print(" Random Uniform Mean     : ",mu)
    print(" Samples Generated Mean  : {}\n".format(X.mean(axis=1)))

    print(" Random  Uniform   Cov      : ")
    print(cov)
    print()
    print(" Samples Generated Cov   : ")
    print(jnp.cov(X))
    print()
    print()
```

```
Sample : 1
 Random Uniform Mean     :  [[0.4532044 0.516343 ]]
 Samples Generated Mean  : [0.4532044  0.51634306]
 Random  Uniform   Cov      :
[[1.0000001 0.9388134]
 [0.9388134 1.0000001]]

 Samples Generated Cov   :
[[1.1111113  0.794049  ]
 [0.794049   0.64343786]]


Sample : 2
 Random Uniform Mean     :  [[0.23853767 0.02100694 0.60624325]]
 Samples Generated Mean  : [0.23853767 0.02100695 0.60624325]

 Random  Uniform   Cov      :
[[1.0000001  0.8045029  0.692321  ]
 [0.8045029  1.0000001  0.55697423]
 [0.692321   0.55697423 1.0000001 ]]

 Samples Generated Cov   :
[[1.0204083  0.9253422  0.60540146]
 [0.9253422  1.1884226  0.60163885]
 [0.60540146 0.60163885 0.88049227]]


Sample : 3
 Random Uniform Mean     :  [[0.23853767 0.01111937 0.60624325 0.33480167]]
 Samples Generated Mean  : [0.23853767 0.01111938 0.60624325 0.3348017 ]

 Random  Uniform   Cov      :
[[1.0000001  0.79658747 0.692321   0.9082242 ]
 [0.79658747 1.0000001  0.55149424 0.7234801 ]
 [0.692321   0.55149424 1.0000001  0.76227987]
 [0.9082242  0.7234801  0.76227987 1.0000001 ]]

 Samples Generated Cov   :
[[1.0101008  0.78176194 0.72696954 1.019525  ]
 [0.78176194 0.9736625  0.6183117  0.7566353 ]
 [0.72696954 0.6183117  1.048394   0.83298415]
 [1.019525   0.7566353  0.83298415 1.1787816 ]]
```

[ ]: