Python stuffs

X. Chen

August 30, 2022

1 Import things

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import math
import pandas as pd
import seaborn as sns
from mpl_toolkits import mplot3d
import numpy as np
import time
import torch as torch
from scipy.stats import norm
```

2 PyTorch

2.1 PyTorch set up

```
torch.set_printoptions(precision=8)
torch.set_default_dtype(torch.float32)
torch.set_default_tensor_type(torch.cuda.FloatTensor)
torch.cuda.is_available()
```

To cpu

```
x.to( "cpu")
```

2.2 PyTorch basic functions

Random tensor

```
torch.randn(N ) # Tensor of 1 x N, dis of N(0,1); "rand" for uniform.
torch.randn(N, M, dtype=torch.double, device='cuda')
```

Zeros tensor

```
torch.zeros(N ) # Tensor of 1 x N.
torch.zeros(2,N,2, dtype=torch.float64)
```

Eye tensor, diagonal matrix

```
torch.eye(dd, dtype=torch.float64,  device='cuda')
```

Copy/ make tensor, **remember to add a dot after 0**.

```
x.detach().clone()
torch.tensor([[a,-a],[1/a,0.]], dtype=torch.float64,  device='cuda'  )
```

Remember to use $torch.$ operation for all things in tensor, it is much faster.

2.3 PyTorch calculations

Elements at same position on last shapes for all four operators $+, -, *, /$, magic tricks

```
a,b  = torch.rand(N,M,2, 2 ),torch.rand(2, 2 )
a+b,a-b,a*b,a/b

# Examples
a=torch.ones(2,2,1,1)
(a*torch.eye(3)).shape #  torch.Size([2, 2, 3, 3]) can matmul with others

a=torch.ones(2,2,1,1,1)
b=torch.rand(3,3, 2)
(a*b).shape #  torch.Size([2, 2, 3, 3, 2])

aa=torch.tensor([1,2,3])
sig1=torch.ones(3,3)*(aa).reshape(1,3) # Gives 3 rows of [1,2,3]
sig2=torch.ones(3,3)*(aa).reshape(3,1) # Gives 3 cols of [1,2,3]^T
```

Useful quick methods 1, have a $N \times d$ tensor, want to have a matrix of $N \times N \times d$ the relationship between different particles,

```
sample_vre = torch.reshape(y[1,:N],(N,1,d))
nominator_tmp = sample_vre-(y[1,:N]*torch.ones(N,N,d)) #slow
nominator_tmp2 = sample_vre-(y[1,:N] ) # quick
```

$N \times 1 \times d - N \times d$ generates a $N \times N \times d$ tensor.
$M \times 1 \times a \times b \times c - N \times a \times b \times c$ generates a $M \times N \times a \times b \times a$ tensor.

Matrix operation A times B, $(NN, MM) \times (MM, NN)$

```
a,b  = torch.rand(N,M,5, 2 ),torch.rand(2, 5 )
a@b==torch.matmul(a,b)
```

Matrix reshape for better calculation

```
x.reshape(nn,nn,dd,1)
```

Matrix operation find $Min, Max, Mean, Sum$ on different dimension

```
x.reshape(n1,n2,n3,n4)
x.min(0),x.max(1),x.mean(2),x.sum(3)
```

Matrix find the inverse matrix, the last two elements must make a square matrix.

```
torch.linalg.inv(c)
```

3 Plots

3.1 Plot of points with regression lines

```
n_of_scheme=[10,20,40,80,160,320,640,1280]
timestep=[10,20,40,80,160,320,640,1280]
plt.rcParams['font.size'] = '16'
plt.rcParams["figure.figsize"] = (7,7)
plt.xlabel('Numbers of Particles')
plt.ylabel('PoC')
#plt.title('My title')
#plt.xlim(0,0.1)
#plt.ylim(0.1,1)
plt.yscale('log')
plt.xscale("log")
##
timestep=np.array(timestep)
m2, b2 = np.polyfit(np.log(timestep),np.log(poc_1),1)
j1,=plt.plot(timestep, (timestep**m2)*np.exp(b2),color='C1' )
k1,=plt.plot(timestep,poc_1,'D',color='C5')

m2, b2 = np.polyfit(np.log(timestep),np.log(poc_2),1)
j2,=plt.plot(timestep, (timestep**m2)*np.exp(b2),color='C2' )
k2,=plt.plot(timestep,poc_2,'s',color='C6')

m2, b2 = np.polyfit(np.log(timestep),np.log(poc_3),1)
j3,=plt.plot(timestep, (timestep**m2)*np.exp(b2),color='C3' )
k3,=plt.plot(timestep,poc_3,'o',color='C7')

m2, b2 = np.polyfit(np.log(timestep),np.log(poc_4),1)
j4,=plt.plot(timestep, (timestep**m2)*np.exp(b2),color='C4' )
k4,=plt.plot(timestep,poc_4,'v',color='C8')

#g,=plt.plot(timestep, (timestep**2),'g-.',label='Order 1',color='black')
z,=plt.plot(timestep, (1/timestep),':',label='Order 1',color='red')
z2,=plt.plot(timestep, (timestep**(-0.5)),'g-.',label='Order 1',color='green')
#q,=plt.plot(timestep, (timestep**1.5),'g-.',label='Order 1.5',color='green')
plt.legend([ (j1,k1),(j2,k2),z,z2],[ r'd=2',r'd=3','Order 1','Order 0.5'])
plt.savefig('pic1.png', dpi=100)
```

3.2 Plot of paths

```
for j in range(ns):
plt.rcParams["figure.figsize"] = (5,5)
plt.rcParams['font.size'] = '10'
plt.xlabel(r'$X_1$')
plt.ylabel(r'$X_2$')
plt.title('SSM')
plt.xlim(-4, 4)
plt.ylim(-4, 4)

a1,=plt.plot( path_ssm[j,:,0], path_ssm[j,:,1],color='black', linewidth=2.0)
a2,=plt.plot( path[j,sp,:,0], path[j,sp,:,1],color='blue', linewidth=0.2)
a3,=plt.plot( pathh[j,0,0], pathh[j,0,1],'D',color='g')
a4,=plt.plot( path[j,sp,0,0], path[j,sp,0,1],'D',color='g')
#   a2,=plt.plot( axiss,path_taming2[1],'-.',color=cc)
#   a3,=plt.plot( axiss,path_taming2[2],'.',color=cc)
plt.legend([(a1),(a2),(a3) ],[r'E path',r'Sp path','Start point'] )
plt.savefig('Path-'+str(j)+'.png', dpi=100)
plt.show()
```

Large amount of paths

```
N=5000
axiss=[]
for i in range(0,M*T+1):
    axiss.append(i*h)
plt.rcParams["figure.figsize"] = (10,10)
plt.rcParams['font.size'] = '20'
plt.xlabel('Time', fontsize=24)
plt.ylabel('Value of paths', fontsize=20)
# plt.title('My title')
plt.xlim(0, T)
plt.ylim(-6.0, 12.0)

for i in range(0,N):
    cc='black'
#     for j in range(0,20):
#         if path[i][j]<-8: cc='red'
    plt.plot( axiss,path_ssm[i],color=cc)

plt.savefig('Paths',dpi=600)
```

3.3 Seaborn multi-plots

Hist plots

```
plt.rcParams['font.size'] = '64'
# Each list is of same length.
data = {'$Min$': mu_list, r'$N_T^{\delta}$': co_list2, '$Max$': sigma_list }
df2 = pd.DataFrame(data)
# set seaborn whitegrid theme
sns.set(style="whitegrid",font_scale=4.25)

g = sns.FacetGrid(df2,col="$Min$",row="$Max$",sharey=False,margin_titles=True,\
sharex=False,height=9.5, aspect=2)
# draw histogram plots
g.map_dataframe(sns.histplot,  x=r'$N_T^{\delta}$') #bins=20,
# control the title of each facet
g = g.set_titles("{col_name}")
co=0
# Add text to each subplot
for (row_val, col_val), ax in g.axes_dict.items():
    ax.text(.1, .6, "Mean: %s  Var: %s "  %\
    (round(n_mean2[co],2),round(n_var2[co],2))  , transform=ax.transAxes)
    ax.text(.1, .3, "Q_le-95: %d  Q_r-95: %d "  %\
    (round(n_ql2[co],2),round(n_qr2[co],2))  , transform=ax.transAxes)
    co+=1
# show the graph
plt.show()
g.tight_layout()
g.savefig("FIG.png")
```

Density plots

```
# Same as above
g = sns.FacetGrid(df2, col="Method",row="Time", hue="Method", margin_titles=True,\
sharex="col",sharey=True, despine=False, height=2.5, aspect=2)

# draw density plots
g = g.map(sns.kdeplot,r'$X_T$', cut=0, fill=True,\
common_norm=False,alpha=1,  legend=False)

# Same as above
```

3.4 3D-plots