Studying the covariance III: LHC $t\bar{t}$ data generated using IonQ

(Dated: August 19, 2021)

We want to see whether our quantum GAN has truly learnt the distribution it is supposed to. This includes whether it correctly captures the covariance matrix. In a previous set of tests we attempted to quantify how well the covariance matrix is learnt compared to the exactly specified one in the $3D$ Gaussian setup and expanded also towards the LHC dataset. In this next test we compare between the LHC data generated via the quantum simulator and the AWS service (in this case using the IonQ machines).

I. SETUP

The basic setup is exactly as in the previous studies with the only difference that now we compare the previously generated LHC data from the quantum simulator with that generated by executing the circuit on an IonQ machine provided through the AWS cloud service.

Note, for this study we need the samples generated saved. In our git there is currently only one run where the samples are provided in this way ():

1. $n_{samp} = 1000$, $d_{latent} = 5$, $n_{layers} = 2$

it would be great to make more samples available as in
`qgmc/classical-quantum-gan/results/ionQ_1k/out_aws_generator_samples.txt` for further analysis.

II. APPROACHING THE COVARIANCES

A. 3D histograms

As before we convert the samples $N(real)$ and $N(fake)$ into 3D histograms, this time there are only 1k samples so we reduce the number of bins:

```
# there is an n_bin dependence here
nbin=15

# data looks like
datareal=(x,y,z)
datafake=(x,y,z)

D_real=np.histogramdd(datareal.T,bins=nbin,range=[[0,9e5],[-6e5,0],[-4,4]])
D_fake=np.histogramdd(datafake.T,bins=nbin,range=[[0,9e5],[-6e5,0],[-4,4]])
```

In Fig. 1(top row) the densities for the real data, i.e. the target, are shown. They are projected from $(E^2, M_T, R) \to (E^2, R), (E^2, M_T), (M_T, R)$. The middle row shows the differences $D(real) - D(fake)$ for the fake data being simulated. This is the setup we have been using to train as well. As before we do not observe a perfect training.

In the bottom row we next show the difference of the fake data obtained from the simulator and from the quantum machine. We observe some differences. In an ideal scenario these should be equal. Keeping in mind that this time the number of samples is much lower, these deviations could also be due to data augmentation.

Unfortunately, since the normalisation has to be the same for the sets being compared we cannot easily plug in a higher sampled result from the simulator to isolate the effect on the quantum side.

Also the single number measures based on this visualisation will suffer, as there is no comparison to a different setup and the data augmentation error cannot be readily estimated.

B. Density KL divergences

As usual we compute the KL divergences based on the estimated densities. We find:

```
# d_latent=5, n_layers=2, n_samp=1000
simulated:      KL_sum = 0.007 (as in n_samp=10000 case)
quantum:        KL_sum = 0.02
```

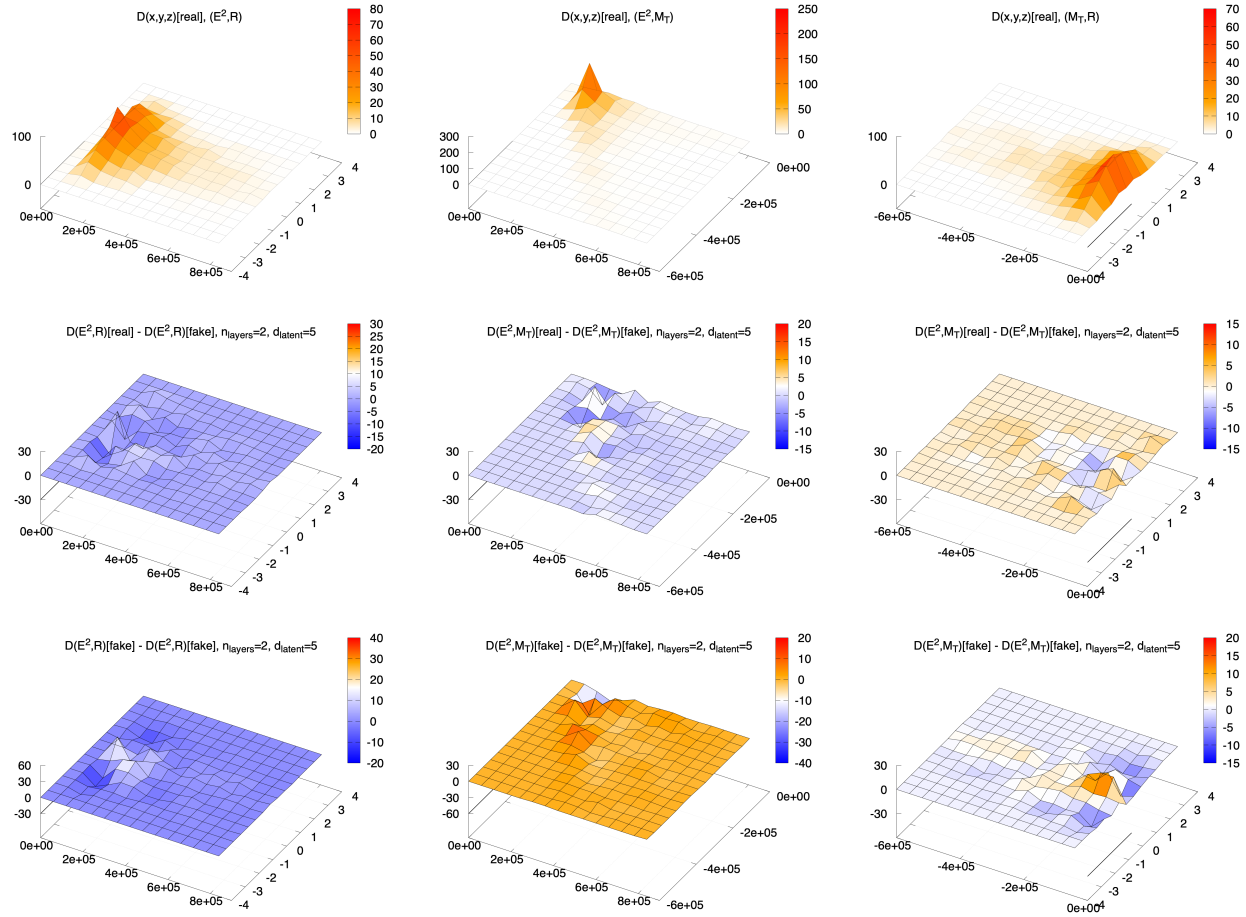

FIG. 1: Estimated densities and differences, please see text for details.

In the ideal case the simulated and quantum cases should be equal. That the simulated result matches that found at higher samples is a sign that the bin or samples dependence might not be that severe for this measure.

C. Covariance matrix

Next we look at the covariance matrices, in particular:

1. target, simulated and quantum cov.mat
2. Eigenvalues and Eigenvectors in all cases

The full results are:

```
# ### Checking Nqubits=3, latent_dim=5, layers=2

 Estimated target covmat 10k:
[[ 7.98751959e+10 -4.49449461e+10 -1.60790274e+03]
 [-4.49449461e+10  4.67255376e+10  2.09009246e+03]
 [-1.60790274e+03  2.09009246e+03  1.21827223e+00]]

  Estimated target covmat:
[[ 4.53255981e+10 -2.53417836e+10 -3.18634627e+03]
 [-2.53417836e+10  2.85224498e+10 -5.81615430e+02]
 [-3.18634627e+03 -5.81615430e+02  1.23848211e+00]]
```

```
  Relative difference 10k and 1k target covmats, dmeas= 0.7476624918451166 and matrix:
[[ 0.56745524  0.56384056  1.98167849]
 [ 0.56384056  0.61042529 -0.27827258]
 [ 1.98167849 -0.27827258  1.01658897]]

  Obtained learned covmat (simulated):
[[ 7.43572673e+10 -1.69438109e+10 -2.13775076e+04]
 [-1.69438109e+10  1.63872691e+10  7.76311975e+03]
 [-2.13775076e+04  7.76311975e+03  1.17718414e+00]]

  Obtained learned covmat (aws):
[[ 3.54792098e+10 -1.33538667e+10  5.06172601e+03]
 [-1.33538667e+10  1.86539929e+10 -4.42784204e+03]
 [ 5.06172601e+03 -4.42784204e+03  8.63830681e-01]]

  Relative difference target and learned aws covmat, dmeas= 1.6930049611659739 and matrix:
[[ 0.78276319  0.52695055 -1.58856746]
 [ 0.52695055  0.65401089  7.61300648]
 [-1.58856746  7.61300648  0.69749145]]

  Relative difference learned simulated and aws covmat , dmeas= 0.256804157714675 and matrix:
[[ 0.47714515  0.78812652 -0.23677812]
 [ 0.78812652  1.13832224 -0.5703689 ]
 [-0.23677812 -0.5703689   0.73381101]]

  Eigenvalues of target 10k, 1k, simulated and aws covmat:
  [1.21817431e+00 1.53965763e+10 1.11204157e+11]
  [1.23786906e+00 1.02258509e+10 6.36221971e+10]
  [1.17037056e+00 1.17981279e+10 7.89464085e+10]
  [8.62641557e-01 1.12837709e+10 4.28494318e+10]

 Averaged ratio of Eigenvalues (target 10k/1k): 1.4125
 Averaged ratio of Eigenvalues (target 10k/simulated): 1.2515
 Averaged ratio of Eigenvalues (target 10k/aws): 1.7906
 Averaged ratio of Eigenvalues (target 1k/simulated): 0.9101
 Averaged ratio of Eigenvalues (target 1k/aws): 1.2753
 Averaged ratio of Eigenvalues (simulated/aws): 1.4149
```

Comparing the relative differecen of cov.mat.s for the 10k/1k targets, i.e. real data, we observe a measure of 0.75, while for the Eigenvalue measures we observe a deviation of $\sim 40\%$ from ideal. The 10k/1k results shows that data augmentation plays a significant role here. At the same time, the results show that the covariance matrices estimated from the samples generated by the simulator or the quantum machines differ, what seems, significantly. The relative difference between them is just 0.26, which is far away from the ideal case of 1, and also the Eigenvalue deviation shows a rather large value of 1.4. Comparing the 10k and 1k targets with the simulated results we once more observe a difference of 0.34 in the Eigenvalue deviation. For the quantum generated results this spread is 0.51.

It would be interesting to figure out whether this difference in spread can be linked to data augmentation or whether it is an effect due to the noise from the quantum machine. In the latter case we would have to find a way to suppress these effects in order to protect the cov.mat. from washing out.

III. OVERVIEW

Generating using the quantum machine we are faced with a noisy computing device. It could be that errors accumulate and become visible in the distributions we are wanting to sample from using the qGAN. In this simple study we found that a conclusive answer to this question is not yet possible as we observe significant effects due to having only 1000 samples when compared to the previous case of 10000. Another question that arises at this point

is whether one quantum machine performs better than another, or whether the same behavior can be observed by including a known noise in a run using a simulator.

Things to do:

- We should perform the same study using a simulator with noise.
- It would be great to increase the statistics to 10k (this is just a random number though, since it's what we had before) ...
- and to make samples available from the other quantum machines for a comparison between the IonQ and IBM setups.