Deep Learning – GAN Assignment

Part 1

The discriminator's input is of the data dimensions (e.g., 8 for diabetes without the label) and its architecture contains 3 dense layers – The first layer with 256 units and an activation function of LeakyReLU, the second layer containing 512 units and an activation function of LeakyReLU and the last layer contains 1 unit with an activation function of sigmoid. After the first layer, we added a dropout of 0.2. This way, the discriminator informs us about whether the sample is real or fake – 1 if real and otherwise 0.

The generator's input is of latent dimensions (of the noise it will get, e.g., 8 in Diabetes data and 20 in German Credit data) and its architecture contains 3 dense layers – The first layer with 512 units and activation function of LeakyReLU, the second layer containing 256 units with an activation function of LeakyReLU and the last layer contains number of units as the number of the data dimensions(e.g. 8 for Diabetes data) with an activation function of tanh. After the first layer, we added a dropout of 0.4.

This architecture and the chosen hyperparameters are based on this paper https://arxiv.org/pdf/1904.09135.pdf which used on the Diabetes data.

Our GAN model's outputs are in the scale of -1 to 1 (the last layer's activation function is tanh), thus for the diabetes data, we first performed min-max scaling in order to be able to produce legitimate examples. For the German credit data, for the categorical features, we created one hot vectors and for the ordinal features we used ordinal encoder. Then in order to scale all the values between -1 to 1 we used min max scaler with appropriate range. After the fake samples' generation, we inverse transform the values back to their classes.

We created 2 GAN models – one for the diabetes data and one for the German credit data.

Questions answering - Analyzing the products of our model:

3.A. We calculated the distance between the original samples and the generated ones and chose the closest ones after transformation. Then we used inverse transformation to reconstruct the samples. (We also cast the samples into integers for better understanding).

Diabetes Data

Samples that fooled the detector:

```
sample number 24:
Fake sample : [  3 115  77  38 212  33   0  29]
Closest real : [  2 104  80  45 191  33   0  29]
Euclidean Distance: 0.3078351318836212
sample number 28:
Fake sample : [  2 152  81  54 318  41   0  24]
Closest real : [  1 136  74  50 204  37   0  24]
Euclidean Distance: 0.4348710775375366
sample number 10:
Fake sample : [  1 110  78  39  91  34   0  22]
Closest real : [  2 107  74  30 100  33   0  23]
Euclidean Distance: 0.2132774293422699
sample number 76:
Fake sample : [  0 171  81  32 108  41   0  21]
Closest real : [  0 162  76  36   0  49   0  26]
Euclidean Distance: 0.47395753860473633
sample number 12:
Fake sample : [ 1 80 77 16 15 23  0 26]
Closest real : [ 1 81 72 18 40 26  0 24]
Euclidean Distance: 0.22155877947807312
```

As we can see the values that fooled the detector are very similar to the real ones. Also, the distances between fake examples and real ones are low.

Samples that did not fooled the detector:

```
sample number 52:
Fake sample : [ 13 150  59  41  20  36   0  27]
Closest real : [ 14 175  62  30   0  33   0  38]
Euclidean Distance: 0.5134834051132202
sample number 82:
Fake sample : [  9 145  62  25  43  29   0  31]
Closest real : [ 10 125  70  26 115  31   0  41]
Euclidean Distance: 0.4571422338485718
sample number 95:
Fake sample : [ 15 143  44  15   0  32   0  25]
Closest real : [ 14 175  62  30   0  33   0  38]
Euclidean Distance: 0.689517080783844
sample number 75:
Fake sample : [ 11 124  58  31  21  32   0  33]
Closest real : [ 10 129  62  36   0  41   0  38]
Euclidean Distance: 0.3906545341014862
sample number 27:
Fake sample : [  9 158  61  42  73  36   0  27]
Closest real : [ 10 129  62  36   0  41   0  38]
Euclidean Distance: 0.5548930168151855
```

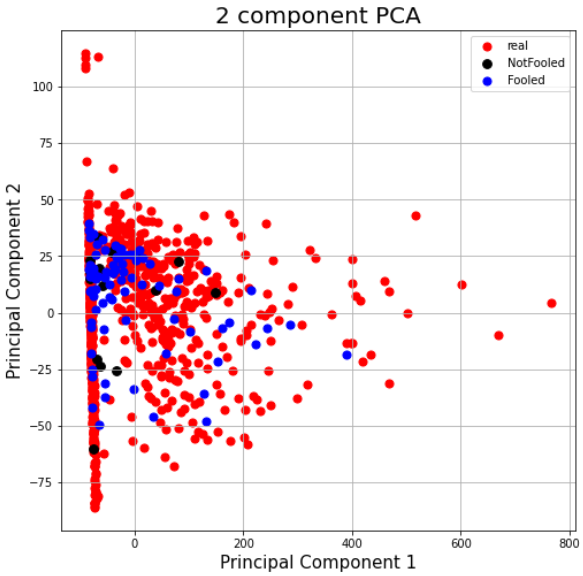
As we can see the values that did not fool the detector are a little bit different from the real ones. Also, the distances between fake examples and real ones are higher than the previous samples that fooled the detector.

In addition, we calculated the mean distance between fake samples which fooled the detector and their closest real samples, and the mean distance between fake samples which **did not** fool the detector and their closest real samples. We found out that the first mean distance ("fooled") is lower than the second one (Did not "fool"):

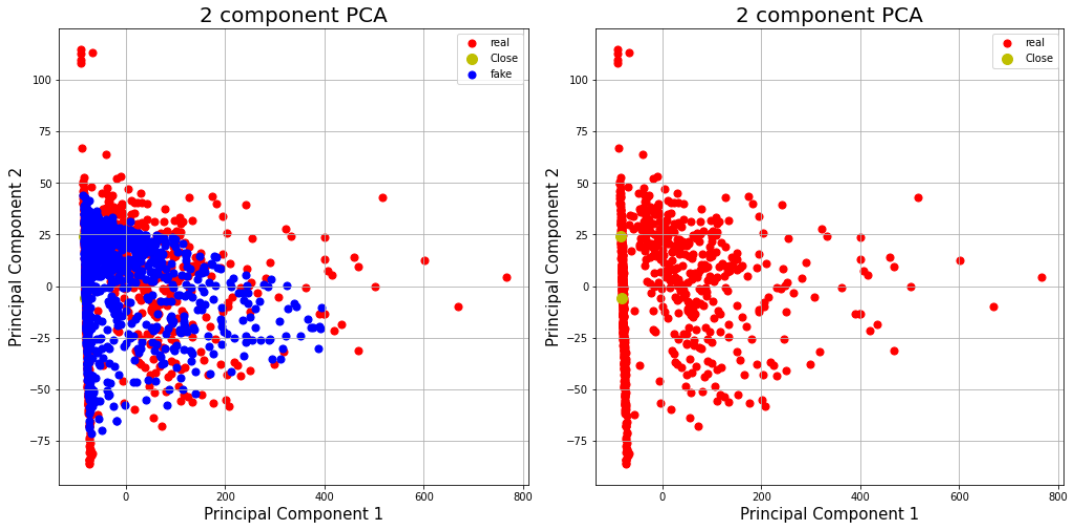
The mean distance of samples which fooled the detector: 0.3738189935684204

The mean distance of samples which did not fool the detector: 0.47054868936538696

The following visualization shows the real samples (red), the samples that fooled the detector (blue) and the samples that did not fool the detector (black). As we can see the samples that fooled the detector (blue) are indeed similar to some of the original data. But the samples that did not fool the detector are very close to samples that did fool, so it is very hard to say why the model decided to classify these samples as fake.

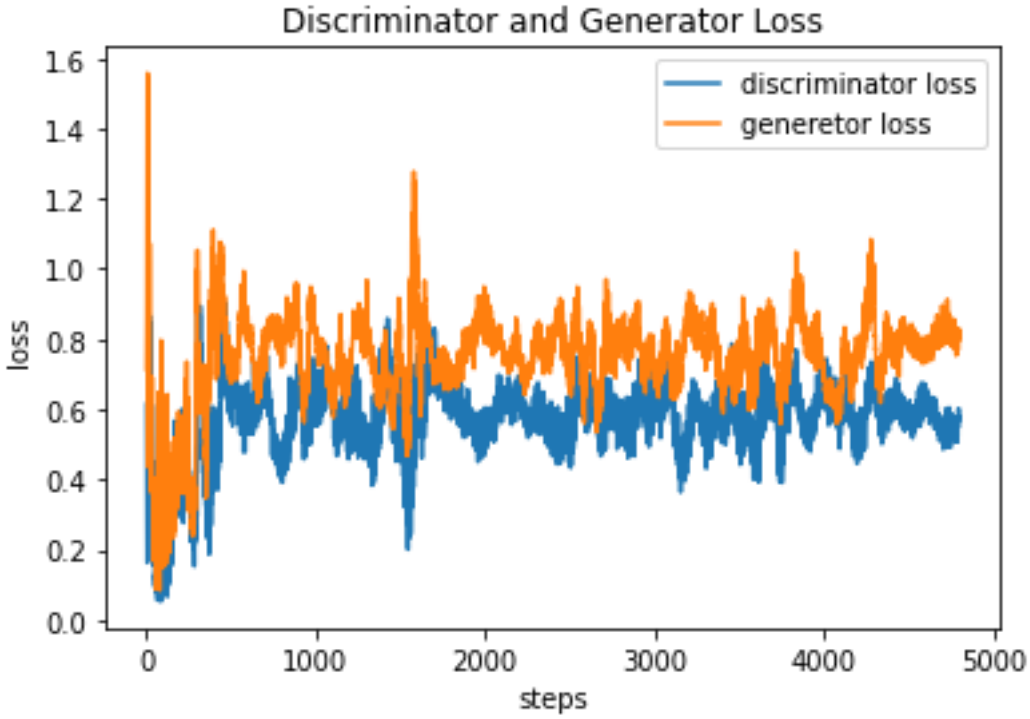


3.B. We generated 100 samples at random after our model was converged. The result showed 82 of 100 generated samples were able to pass as real samples. In the following graphs we used a PCA dimensionality reduction technique to visualize the real and fake data. As we can see, our model was able to generate samples very similar to the real data. The red dots represent real data, the blue ones represent fake data (1000 samples), and the yellow ones represent the closest real and fake samples. The left graph shows the real and fake data, while the right one shows only the real data.



The generated samples are presented on the notebook.

3.C. Graph describing the loss of the generator and the discriminator:



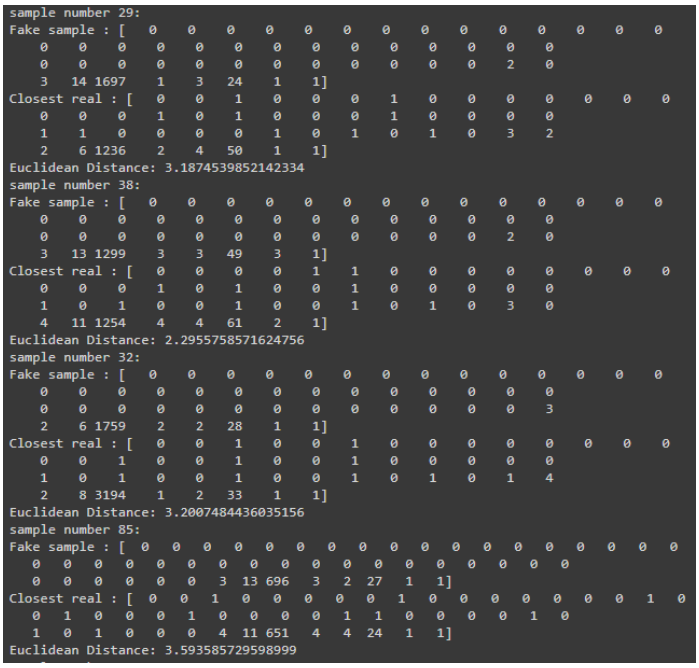
As we can see from the graph, in the beginning of the training both models go "back and forth" while the discriminator was far better than the generator (the loss of the generator is very high).

The discriminator was a consistent leader while its loss is between 0.4-0.6 and the loss of the generator is a little bit higher between 0.7-1.

We can understand the result because the classification task of the discriminator is easier than the generator task.

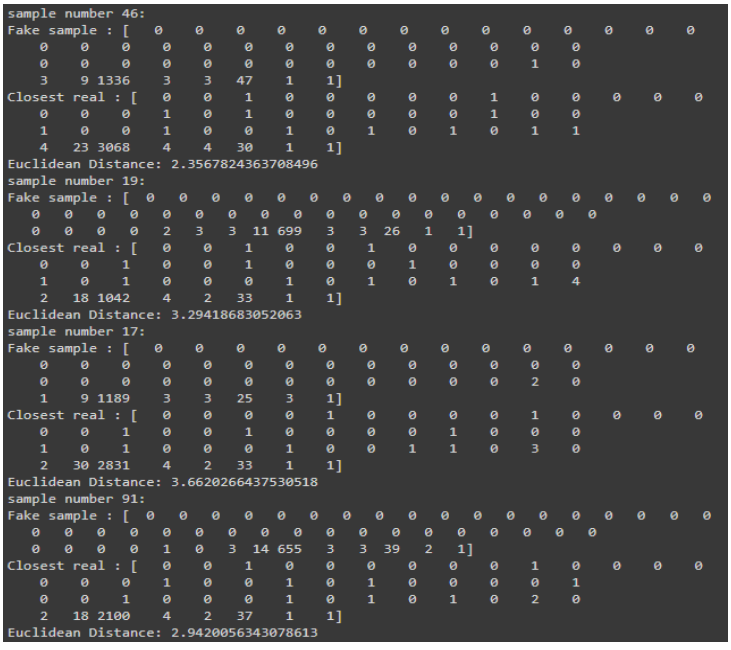
German Credit Data

Samples that fooled the detector:



As we can see the values that fooled the detector are very similar to the real ones. Also, the distances between fake examples and real ones are bigger than the previous distances in diabetes data, and that's because we have more features here and thus we have more arguments in the distance function what it makes it bigger.

Samples that did not fool the detector:



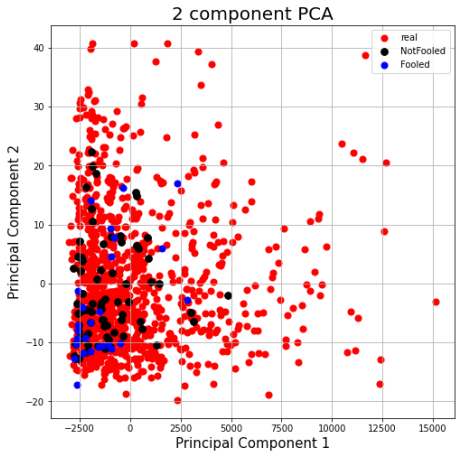
As we can see the values that did not fool the detector are different from the real ones.

In addition, we calculated the mean distance between fake samples which fooled the detector and their closest real samples, and the mean distance between fake samples which **did not** fool the detector and their closest real samples. We found out that both distances are very close

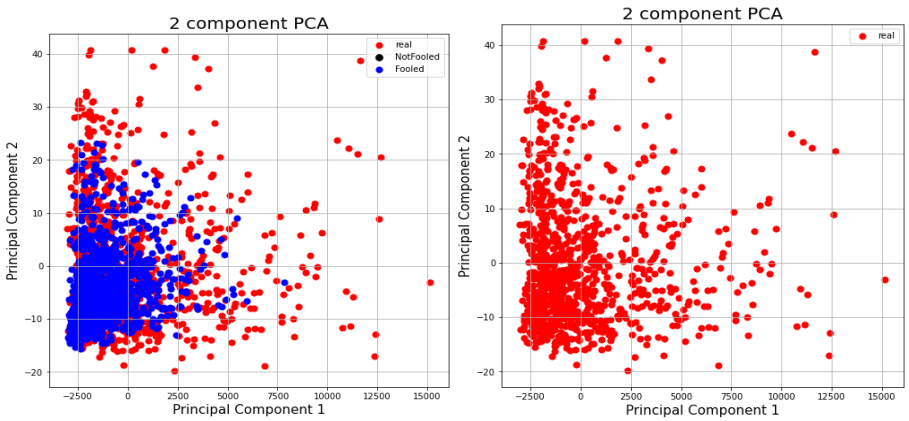
The mean distance of samples which fooled the detector: 3.4326775074005127

The mean distance of samples which did not fool the detector: 3.432727098464966

The following visualization shows the real samples (red), the samples that fooled the detector (blue) and the samples that did not fool the detector (black). As we can see the samples that fooled the detector (blue) are indeed similar to some of the original data. But the samples that did not fool the detector are very close to samples that did fool and to real samples, so it is very hard to say why the model decided to classify these samples as fake.

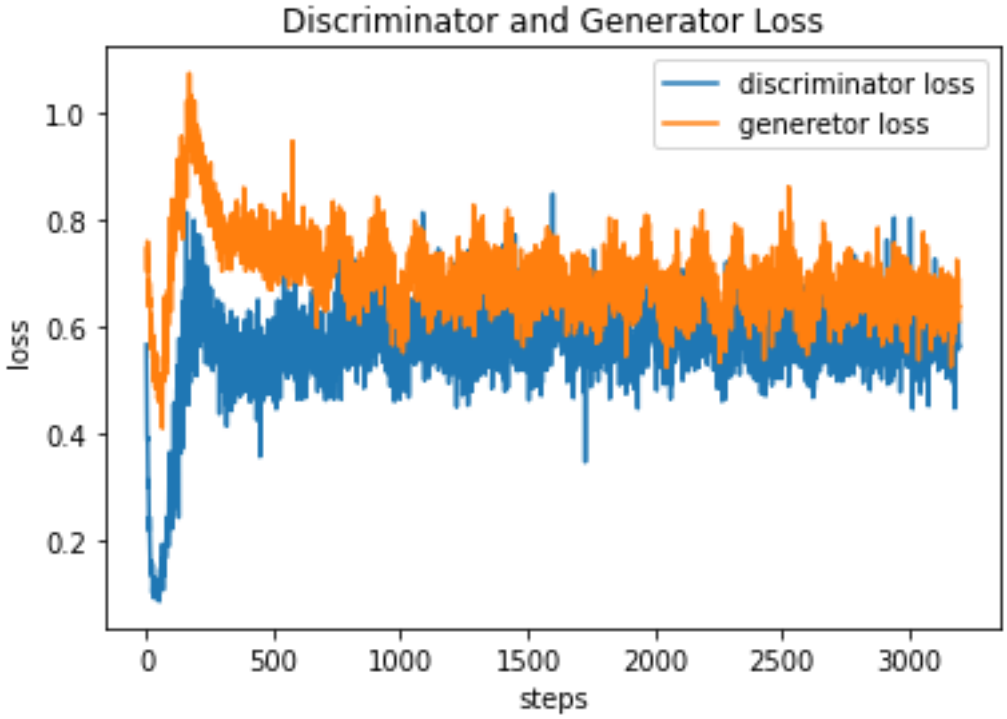


3.B. We generated 100 samples at random after our model was converged. The result showed 29 of 100 generated samples were able to pass as real samples. In the following graphs we used a PCA dimensionality reduction technique to visualize the real and fake data. It seems that our model was able to generate samples very similar to the real data, but the model was able to detect them. The red dots represent real data, the blue ones represent fake data (1000 samples). The left graph shows the real and fake data, while the right one shows only the real data.



The generated samples are presented on the notebook.

3.C. Graph describing the loss of the generator and the discriminator:



As we can see from the graph, in the beginning of the training both models go "back and forth" while the discriminator was far better than the generator (the loss of the generator is higher).

The discriminator was a consistent leader while its loss is between 0.4-0.7 and the loss of the generator is a little bit higher between 0.6-1.

We can understand the result because the classification task of the discriminator is easier than the generator task.