Basic Method

The basic data generator learns from a table T and generates a table T_{syn}. all columns of the table are considered to be random variables. We assume these random variables must follow a joint distribution. One row is one observation from the joint distribution. We face a lot of difficulties like

1. Real-world tabular data consists of mixed types. To simultaneously generate a mix of discrete and continuous columns, GANs must apply both softmax and tanh on the output.
2. In images, pixels' values follow a Gaussian-like distribution, which can be normalized to [−1,1] using a min-max transformation. A tanh function is usually employed in the last layer of a network to output a value in this range. Continuous values in tabular data are usually non-Gaussian where min-max transformation will lead to vanishing gradient problems.
3. We use kernel density estimation to estimate the number of modes in a column. Vanilla GAN couldn't model all modes on a simple 2Ddataset; thus it would also struggle in modeling the multimodal distribution of continuous columns.

To avoid this we can apply spatial normalization.

Spatial Normalization

In this method, each column is processed independently. Each value is represented as a one-hot vector indicating the mode, and a scalar indicating the value within the mode. Steps

1. For each column C_i use a gaussian mixture model to estimate the mode m_i and fit the GM model. where η is the mode, φ is the standard deviation and μ is the weights.

$$\mathbb{P}_{C_i}\left(C_{i,j}\right) - \sum_{k=1}^{N} \mu_k \mathbb{N}\left(C_{i,j}; \eta_k, \phi_k\right)$$

2. For each value of $C_{i,j}$ compute the probability coming from each mode. This can be computed as

$$\rho_k = \mu_k \mathbb{N}\left(C_{i,j}; \eta_k, \phi_k\right)$$

3. Sample one mode from given the probability density, and use the sampled mode to normalize the value
4. This can be done by let $\beta_{i,j}$ is one hot representation of $C_{i,j}$ indicating the k^{th} mode,
   $\beta_{i,j}$ = [0, 0, 0 …..1, 0, 0….]. 1 at k^{th} index.
   $$\alpha_{i,j} = \frac{C_{i,j} - \eta_k}{4\phi_k}$$
5. Then the row can be transformed as
   $$r_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus ... \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus d_{1,j} \oplus ... \oplus d_{N_d,j}$$

Conditional Generator

Traditionally, the generator in a GAN is fed with a vector sampled from a standard multivariate normal distribution (MVN). By training together with a Discriminator network, one eventually obtains a deterministic transformation that maps the standard MVN into the distribution of the data. This method of training a generator does not account for the imbalance in the categorical columns. If the training data are randomly sampled during training, the rows that fall into the minor category will not be sufficiently represented, thus the generator may not be trained correctly. If the training data are resampled, the generator learns the resampled distribution which is different from the real data distribution. This problem is reminiscent of the “class imbalance” problem in discriminatory modeling - the challenge however is exacerbated since there is not a single column to balance and the real data distribution should be kept intact.

Specifically, the goal is to resample efficiently in a way that all the categories from discrete attributes are sampled evenly (but not necessarily uniformly) during the training process, and to recover the(not-resampled) real data distribution during the test.

The conditional vector is the way of indicating the condition. Let D is the discret column and d is the one hot vector of D and m is the mask then

$$m_i^{(k)} = \begin{cases} 1 & if\ i = i^* and\ k = k^* \\ 0 & otherwise \end{cases}$$

Then the conditional vector is the concat of all masks.

Training and parameter optimization

1. Construct fakeZ which is a random distribution of same mean and variance as the real data
2. Calculate conditional vector C and mask M
3. Concat C with fakeZ
4. fake = Net_G(fakeZ) with activation
5. Construct real data from training data
6. y_fake = Net_D(fake)
7. y_real = Net_D(real)
8. Calculate the gradient penalty from WGAN
9. Calculate discriminator loss by loss_d = -mean(y_fake) - mean(y_real)
10. Backprop both loss_d and gradient penalty
11. Optimize the parameters of Net_D by adam optimizer
12. Repeat step 1 to 6
13. Calculate the conditional loss between fake, C and M by cross entropy formula
14. Calculate generator loss by loss_g = -mean(y_fake) + conditional loss
15. Backprop loss_g
16. Optimize the parameters of Net_G by adam optimizer

Evaluation

a)Evaluation through a Regressor:

For this evaluation method , the motive is to find the similarity between the distribution of each feature column of the real and generated data.Hence for this we follow an iterative approach where each single feature is taken as to be the target variable in different iterations and the rest as the feature data. And each iteration consists of the following steps.

1.As we have done for our Generative Models , similarly here too we apply the Spatial Normalisation on both real and Generated data before passing them through the Regressor.

2.Now a linear Regressor is trained on the real data features in such a way that in each time each data feature column is labeled as the prediction/target(Y) column and the rest of the features are treated as the feature columns for this prediction.The motivation for training such a regressor is to find dependency of each feature with respect to all other feature and then to evaluate how close is the distribution of the generated feature column corresponding to this feature.

3. Once the regressor is trained then the other feature columns of both real and generated features are fed into regresor separately to get a real and generated feature column(the one for which the evaluation is done) respectively.

4.Once we get the prediction for the feature column of both the real and generated data is done we find their respective standard deviation and mean and compare them as a method of evaluating the similarity the closer the values are the more similar they are.
Mean-

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Mean is used to get an estimation of the point where the data is centred upon or get a measure of most common terms , so comparing the mean we get the similarity of the frequent points in the data.
Standard Deviation-

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2},$$

This standard deviation is used to check the spread of the distribution hence comparing this would give us the similarity of spread.

Hence comparing these 2 quantities would give the overall similarity of the real and generated distribution.