Creating fair machine learning models with Generative Adversarial Networks

Hamaad Musharaf Shah, Vice President-Lead Data Scientist at Deutsche Bank

25/06/2019

Disclaimer

This work is not necessarily a representation of any past or current employer of mine.

Introduction

- Ensuring fairness in machine learning is a key aspect especially in the financial services domain.
  - Think of a mortgage underwriting machine learning pipeline that perhaps might be sensitive to attributes such as gender and race without explicitly being trained on such features.
  - This can potentially cause a hidden cost of capital charge from an operational risk perspective if customers are not treated fairly.
- To overcome this potential issue we discuss a method published in the 31st conference on Neural Information Processing Systems, i.e., NeurIPS 2017.
  - The title of that paper: "Learning to Pivot with Adversarial Networks".
- Personally I first came across the aforementioned paper in this blogpost.
  - https://blog.godatadriven.com/fairness-in-ml
- This lecture of mine will discuss the aforementioned paper and Generative Adversarial Networks (GAN) in detail. It will also provide a R version of the code associated in the aforementioned blogpost.

Income Prediction

- We use the “Census Income” dataset to predict whether a person’s income is greater than \$50,000 per year. The link to the dataset is here.
  - https://archive.ics.uci.edu/ml/datasets/Adult
- The features used are denoted as below.
  - $x \sim P_X(x)$
- The sensitive attribute(s) are denoted as below.
  - $z \sim P_Z(z)$
- The target variable is denoted as below.
  - $y \sim P_Y(y)$

Classifier and Adversarial Model

- The classifier is a deep learner denoted as below.
  - $C(x|\theta_C) : x \rightarrow [0,1]$
- Note that we are not explicitly training the model on the sensitive attributes. However it is entirely possible that there might be some hidden bias within our features that perhaps act as proxies for the sensititive attributes.
- The sensitive attributes for this exercise are race and gender. These are binary variables.
- For each of these sensitive attributes we want to ensure the following holds.
  - $P(C(x|\theta_C)|z=1) = P(C(x|\theta_C)|z=0)$
- We introduce the concept of an adversarial deep learner model which takes in the predictions of the classifier in order to predict the sensitive attributes as below.
  - $R(C(x|\theta_C)|\theta_R) : C(x|\theta_C) \rightarrow [0,1]$
- In an ideal scenario the classifier will perform at its optimal level while simultaneously the adversarial model will be completely unable to predict sensitive attributes given the predictions from the optimal classifier. Therefore, $C(x|\theta_C^*)$ and Z will be independent random variables.
- We now introduce the GAN model in which the generator plays the role of the classifier and the discriminator plays the role of the adversarial model.

Generative Adversarial Networks

- The purpose of deep learning is to learn a representation of high dimensional and noisy data using a sequence of differentiable functions, i.e., geometric transformations, that can perhaps be used for supervised learning tasks among others.
- It has had great success in discriminative models while generative models have fared worse due to the limitations of explicit maximum likelihood estimation (MLE).
- Adversarial learning as presented in the GAN model aims to overcome these problems by using implicit MLE.

Generative Adversarial Networks

- There are 2 main components to a GAN, the generator and the discriminator, that play an adversarial game against each other.
- In doing so the generator learns how to create realistic synthetic samples from noise, i.e., the latent space z, while the discriminator learns how to distinguish between a real sample and a synthetic sample.
- The representation learnt by the discriminator can later on be used for other supervised learning tasks, i.e., automatic feature engineering or representation learning.

Generative Adversarial Networks

Generator

- Assume that we have a prior belief on where the latent space z lies: $p(z)$.
- Given a draw from this latent space the generator G, a deep learner parameterized by θ_G, outputs a synthetic sample.
  - $G(z|\theta_G) : z \rightarrow x_{synthetic}$

Generative Adversarial Networks

Discriminator

- The discriminator D is another deep learner parameterized by θ_D and it aims to classify if a sample is real or synthetic, i.e., if a sample is from the real data distribution: $p_{\text{data}}(x)$
- Or the synthetic data distribution: $p_G(x)$
- Let us denote the discriminator D as follows.
  - $D(x|\theta_D) : x \rightarrow [0, 1]$
- Here we assume that the positive examples are from the real data distribution while the negative examples are from the synthetic data distribution.

Generative Adversarial Networks

Game

- A GAN simultaneously trains the discriminator to correctly classify real and synthetic examples while training the generator to create synthetic examples such that the discriminator incorrectly classifies real and synthetic examples.
- This 2 player minimax game has the following objective function.

$$\min_{G(z|\theta_G)} \max_{D(x|\theta_D)} V(D(x|\theta_D), G(z|\theta_G)) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \log D(x|\theta_D) + \mathbb{E}_{z \sim p(z)} \log(1 - D(G(z|\theta_G)|\theta_D))$$

Generative Adversarial Networks

Game

- Please note that the above expression is basically the objective function of the discriminator.

$$\mathbb{E}_{x\sim p_{\text{data}}(x)} \log D(x|\theta_D) + \mathbb{E}_{x\sim p_G(x)} \log(1 - D(x|\theta_D))$$

- It is clear from how the game has been set up that we are trying to obtain a solution θ_D for D such that it maximizes $V(D,G)$ while simultaneously we are trying to obtain a solution θ_G for G such that it minimizes $V(D,G)$.

Generative Adversarial Networks

Game

- We do not simultaneously train D and G. We train them alternately: Train D and then train G while freezing D. We repeat this for a fixed number of steps.
- If the synthetic samples taken from the generator G are realistic then implicitly we have learnt the distribution $p_G(x)$. In other words, $p_G(x)$ can be seen as a good estimation of $p_{\text{data}}(x)$.
- The optimal solution will be as follows.

$$p_G(x) = p_{\text{data}}(x)$$

Generative Adversarial Networks

Game

- To show this let us find the optimal discriminator D^* given a generator G and sample x.

$$
\begin{aligned}
V(D, G) &= \mathbb{E}_{x\sim p_{\text{data}}(x)} \log D(x|\theta_D) + \mathbb{E}_{x\sim p_G(x)} \log(1 - D(x|\theta_D)) \\
&= \int_x p_{\text{data}}(x) \log D(x|\theta_D) dx + \int_x p_G(x) \log(1 - D(x|\theta_D)) dx \\
&= \int_x \underbrace{p_{\text{data}}(x) \log D(x|\theta_D) + p_G(x) \log(1 - D(x|\theta_D))}_{J(D(x|\theta_D))} dx
\end{aligned}
$$

Generative Adversarial Networks

Game

- Let us take a closer look at the discriminator's objective function for a sample x.

$$
\begin{aligned}
J(D(x|\theta_D)) &= p_{\text{data}}(x) \log D(x|\theta_D) + p_G(x) \log(1 - D(x|\theta_D)) \\
\frac{\partial J(D(x|\theta_D))}{\partial D(x|\theta_D)} &= \frac{p_{\text{data}}(x)}{D(x|\theta_D)} - \frac{p_G(x)}{(1 - D(x|\theta_D))} \\
0 &= \frac{p_{\text{data}}(x)}{D^*(x|\theta_{D*})} - \frac{p_G(x)}{(1 - D^*(x|\theta_{D*}))} \\
p_{\text{data}}(x)(1 - D^*(x|\theta_{D*})) &= p_G(x)D^*(x|\theta_{D*}) \\
p_{\text{data}}(x) - p_{\text{data}}(x)D^*(x|\theta_{D*})) &= p_G(x)D^*(x|\theta_{D*}) \\
p_G(x)D^*(x|\theta_{D*}) + p_{\text{data}}(x)D^*(x|\theta_{D*})) &= p_{\text{data}}(x) \\
D^*(x|\theta_{D*}) &= \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}
\end{aligned}
$$

Generative Adversarial Networks

Game

- We have found the optimal discriminator given a generator. Let us focus now on the generator's objective function which is essentially to minimize the discriminator's objective function.

$$
\begin{aligned}
J(G(x|\theta_G)) &= \mathbb{E}_{x\sim p_{\text{data}}(x)} \log D^*(x|\theta_{D^*}) + \mathbb{E}_{x\sim p_G(x)} \log(1 - D^*(x|\theta_{D^*})) \\
&= \mathbb{E}_{x\sim p_{\text{data}}(x)} \log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) + \mathbb{E}_{x\sim p_G(x)} \log\left(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) \\
&= \mathbb{E}_{x\sim p_{\text{data}}(x)} \log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) + \mathbb{E}_{x\sim p_G(x)} \log\left(\frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right) \\
&= \int_x p_{\text{data}}(x) \log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx + \int_x p_G(x) \log\left(\frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx
\end{aligned}
$$

- We will note the Kullback–Leibler (KL) divergences in the above objective function for the generator:

$$
D_{KL}(P||Q) = \int_x p(x) \log\left(\frac{p(x)}{q(x)}\right) dx
$$

Generative Adversarial Networks

Game

- Recall the definition of a λ divergence.

$$D_{\lambda}(P||Q) = \lambda D_{KL}(P||\lambda P + (1-\lambda)Q) + (1-\lambda)D_{KL}(Q||\lambda P + (1-\lambda)Q)$$

- If λ takes the value of 0.5 this is then called the Jensen-Shannon (JS) divergence. This divergence is symmetric and non-negative.

$$D_{JS}(P||Q) = 0.5D_{KL}\left(P\middle|\middle|\frac{P+Q}{2}\right) + 0.5D_{KL}\left(Q\middle|\middle|\frac{P+Q}{2}\right)$$

Generative Adversarial Networks

Game

- Keeping this in mind let us take a look again at the objective function of the generator.

$$
\begin{aligned}
J(G(x|\theta_G)) &= \int_x p_{\text{data}}(x) \log\left(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx + \int_x p_G(x) \log\left(\frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx \\
&= \int_x p_{\text{data}}(x) \log\left(\frac{2}{2}\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx + \int_x p_G(x) \log\left(\frac{2}{2}\frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx \\
&= \int_x p_{\text{data}}(x) \log\left(\frac{1}{2}\frac{1}{0.5}\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx + \int_x p_G(x) \log\left(\frac{1}{2}\frac{1}{0.5}\frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)}\right) dx \\
&= \int_x p_{\text{data}}(x) \left[\log(0.5) + \log\left(\frac{p_{\text{data}}(x)}{0.5(p_{\text{data}}(x) + p_G(x))}\right)\right] dx \\
&+ \int_x p_G(x) \left[\log(0.5) + \log\left(\frac{p_G(x)}{0.5(p_{\text{data}}(x) + p_G(x))}\right)\right] dx
\end{aligned}
$$

Generative Adversarial Networks

Game

$$
\begin{aligned}
&= \log\left(\frac{1}{4}\right) + \int_x p_{\text{data}}(x)\left[\log\left(\frac{p_{\text{data}}(x)}{0.5(p_{\text{data}}(x) + p_G(x))}\right)\right] dx \\
&+ \int_x p_G(x)\left[\log\left(\frac{p_G(x)}{0.5(p_{\text{data}}(x) + p_G(x))}\right)\right] dx \\
&= -\log(4) + D_{KL}\left(p_{\text{data}}(x)\middle|\middle|\frac{p_{\text{data}}(x) + p_G(x)}{2}\right) + D_{KL}\left(p_G(x)\middle|\middle|\frac{p_{\text{data}}(x) + p_G(x)}{2}\right) \\
&= -\log(4) + 2\left(0.5D_{KL}\left(p_{\text{data}}(x)\middle|\middle|\frac{p_{\text{data}}(x) + p_G(x)}{2}\right) + 0.5D_{KL}\left(p_G(x)\middle|\middle|\frac{p_{\text{data}}(x) + p_G(x)}{2}\right)\right) \\
&= -\log(4) + 2D_{JS}(p_{\text{data}}(x)||p_G(x))
\end{aligned}
$$

Generative Adversarial Networks

Game

- It is clear from the objective function of the generator above that the global minimum value attained is $-\log(4)$ which occurs when the following holds.

$$p_G(x) = p_{\text{data}}(x)$$

- When the above holds the Jensen-Shannon divergence, i.e., $D_{JS}(p_{\text{data}}(x)||p_G(x))$, will be zero. Hence we have shown that the optimal solution is as follows.

$$p_G(x) = p_{\text{data}}(x)$$

Generative Adversarial Networks

Game

- Note that the above implies that the optimal value for the discriminator is as follows.

$$
\begin{aligned}
D^*(x|\theta_{D^*}) &= \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} \\
&= \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{data}}(x)} \\
&= \frac{1}{2}
\end{aligned}
$$

- The optimal discriminator cannot distinguish between real and synthetic data and this is precisely what we need for the adversarial model to do as well: To not be able to distinguish between sensitive attributes.

Classifier and Adversarial Model

- Returning back to our problem at hand, we use adversarial training to find a classifier such that its predictions cannot be used to predict for sensitive attributes by the adversarial model.
- This 2 player minimax game has the following objective function. Note that $\tilde{y}$ are predictions from the classifier.

$$\max_{C(x|\theta_C)} \min_{R(C(x|\theta_C)|\theta_R)} V(C(x|\theta_C), R(C(x|\theta_C)|\theta_R))$$

$$= \underbrace{\mathbb{E}_{x \sim P_X(x)} \mathbb{E}_{y \sim P_{Y|X}(y|x)} \log C(x|\theta_C)}_{\text{Classifier maximizes this term}} - \underbrace{\mathbb{E}_{x \sim P_X(x)} \mathbb{E}_{\tilde{y} \sim P_{\tilde{Y}|X}(\tilde{y}|x)} \mathbb{E}_{z \sim P_{Z|\tilde{Y}}(z|\tilde{y})} \log R(C(x|\theta_C)|\theta_R)}_{\text{Classifier minimizes this term and the adversarial maximizes this term}}$$

Classifier and Adversarial Model

- We note that the classifier is aiming to maximize it's likelihood while minimizing that of the adversarial while the adversarial is aiming to maximize it's own likelihood.
- The solution to this 2 player game theoretically, albeit not empirically as we shall observe, results in a classifier that is both optimal and fair.
- For the optimal classifier $C(x|\theta_C^*)$ to be fair as well it has to be shown to be independent from the sensitive attributes Z.
- Essentially this means that the adversarial model $R(C(x|\theta_C^*)|\theta_R)$ is not able to predict the sensitive attributes, i.e., it essentially becomes an uninformative prior on the sensitive attributes as opposed to an informative posterior since the predictions, i.e., new information, from the classifier do not help in updating this uninformative prior, i.e., $P(Z)$.

Classifier and Adversarial Model

- The objective function of the 2 player game is bounded above as follows.

$$
\begin{aligned}
&\max_{C(x|\theta_C)} \min_{R(C(x|\theta_C)|\theta_R)} V(C(x|\theta_C), R(C(x|\theta_C)|\theta_R)) \\
&= \underbrace{\mathbb{E}_{x \sim P_X(x)} \mathbb{E}_{y \sim P_{Y|X}(y|x)} \log C(x|\theta_C)}_{\text{Classifier maximizes this term}} - \underbrace{\mathbb{E}_{x \sim P_X(x)} \mathbb{E}_{\tilde{y} \sim P_{\tilde{Y}|X}(\tilde{y}|x)} \mathbb{E}_{z \sim P_{Z|\tilde{Y}}(z|\tilde{y})} \log R(C(x|\theta_C)|\theta_R)}_{\text{Classifier minimizes this term and the adversarial maximizes this term}} \\
&\leq \mathbb{E}_{x \sim P_X(x)} \mathbb{E}_{y \sim P_{Y|X}(y|x)} \log C(x|\theta_C^*) - \mathbb{E}_{z \sim P_Z(z)} \log R(C(x|\theta_C^*)|\theta_R)
\end{aligned}
$$

- The optimal classifier $C(x|\theta_C^*)$ maximizes it's own likelihood and minimizes the likelihood of the adversarial such that the upper bound holds, i.e., $R(C(x|\theta_C^*)|\theta_R)$ is forced to move towards the uninformative prior $P(Z)$. Hence the classifier is theoretically both optimal and fair.
- Empirically the upper bound is not strict therefore a trade off has to be made between the classifier being optimal and fair. We shall see this in the following experiment using the "Census Income" data.

Conclusion

- We have shown how to use adversarial training, inspired by the GAN model, in order to provide a trade off between an optimal classifier and a fair classifier, i.e., one that is not sensitive to attributes such as race and gender.
- The results shows that machine learning might actually be used to train fair decision making pipelines such that the raw data's hidden bias is not reflected in this decision making, even though sensitive attributes might have been explicitly removed.
- This is a powerful and interesting way of using the idea of adversarial training derived from the GAN model for a goal that is desirable from an ethics perspective.

References

1. Goodfellow, I., Bengio, Y. and Courville A. (2016). Deep Learning (MIT Press).
2. Geron, A. (2017). Hands-On Machine Learning with Scikit-Learn & Tensorflow (O'Reilly).
3. Radford, A., Luke, M. and Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (https://arxiv.org/abs/1511.06434).
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative Adversarial Networks (https://arxiv.org/abs/1406.2661).
5. Chollet, F. (2018). Deep Learning with Python (Manning).
6. https://blog.godatadriven.com/fairness-in-ml
7. Louppe, G., Kagan, M., Kranmer, K. (2017). Learning to Pivot with Adversarial Networks (https://papers.nips.cc/paper/6699-learning-to-pivot-with-adversarial-networks.pdf)
8. https://towardsdatascience.com/automatic-feature-engineering-using-generative-adversarial-networks-8e24b3c16bf3

Blog on Medium and Code on GitHub

- On GitHub:
  - https://github.com/hamaadshah/autoencoders_keras
  - https://github.com/hamaadshah/gan_keras
  - https://github.com/hamaadshah/market_risk_gan_keras
  - https://github.com/hamaadshah/autoencoders_pytorch
  - https://github.com/hamaadshah/gan_deeplearning4j
- On Medium - search for my article titles below:
  - Automatic feature engineering using deep learning and Bayesian inference.
  - Automatic feature engineering using Generative Adversarial Networks.
  - Using Bidirectional Generative Adversarial Networks to estimate Value-at-Risk for Market Risk Management.