Analysis and Result Replication of the paper: "Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure"

Mohamad Lakkis

August, 2024

Contents

1 Introduction

In this Analysis, we will be explaining concepts of this paper in depth and adding to it my own Analysis and interpretation, to make it easier for anyone to understand it. I will also be adding examples. And finally we will be trying to replicate some of the results, since not all results can be replicated due to hardware limitations. Once done all of the my replicated results will be added to the paper, and the code will be available on my github.
We will first start by explaining and analyzing the paper, and then we will move towards implementing the code and replicating results.

2 Understanding the Problem

What the referred paper explains how we can actually de-bias the training data to make it more representative of the true data. In many cases, the data that we have can be biased, especially towards points that are not that highly represented in the data. So for example, say you have a dataset of faces (example from the paper) and you want to classify whether a given image is a face or not. Maybe in your own dataset you have many pictures of asian people, and a very small portion of european people, this will lead the model to be more accurate on new object that comes from asian backgrounds, and less accurate towards other faces, we want our model to actually learn this bias, and to adapt to it, even when we don't have a lot of data from that specific characteristic. So couple of questions arises, one of them is how can we actually detect these rarer points in the data. But first we need to define what rare means in the data. What we mean by rare is having low numbers of samples when we filter based on particular features. So based on the last example we filtered based if the person is asian or not. But following this analysis another follow-up question arrises, how do we choose these features? To showcase the innovational approach that the paper suggest, we need first to look back and see what the past methods where doing and what is the downside of each one, for then to combine all of these positive features into this new tunable algorithm.

2.1 Past Methodologies and their Downside for learning the latent structure of the data

- This field of learning latent structure of the data is very popular especially for re-sampling methods, and generating new data based on the latent structure of the data. (Common examples: GANs, VAE, etc)
- In this analysis we will go in depth on how VAE works, the GANs architecture are for a different analysis.
- What we will see that these old models focused on class imbalances, they do NOT account for the variability in each class. So for example, we have three classes dogs, cats, none. What these old models thrive to do is to

make the frequency of cats as equal to frequency to dogs, to remove the bias.

- So to start we will giving couple of these methods and their downside.

2.1.1 Resampling for class imbalance

- This re-sampling approach address the class imbalance problem. Meaning we can have more samples of one class than the other, this is what this algorithm solves.
  Downside:

- **Ignores the variability in each class.** Within a single class, there might be different subgroups or features that are underrepresented. For instance, in a dataset of images of cats, there could be many more images of certain breeds of cats than others, leading to a within-class imbalance, which is a problem that this algorithm doesn't address.

- Maybe you could say that in additon to using this algorithm we can **manually annotate the data** based on specific features. But this approach is not always feasible since maybe some of these filters might require a combination of features, say eyes and nose for example. And even if this was simple we require experts in the field to manually annotate each point in the data. Which is not feasible, and not efficient.

2.1.2 Generating debiased data

- These generative transformations actually help us to generate new data(i.e. artificial data) that is more representative of the true data. And there are many popular algorithms to do so, such as: GANs, VAEs, etc.
  Downside:

- Although these algorithms account for the variabilty in each class and they do learn these latent variables, they are **static methods** meaning we apply these on time at the beginning of the training, by generating this new "unbiased" dataset and using as our new dataset for our training (whether image classification, text classification, or ...).

- In addition to that these algorithms use artificial data, and **Not true data** (as we will explain in the section of how "VAEs" work)

2.1.3 Supervised Learning Approaches

- They work similar to the following: For instance, consistently high residuals for a particular subgroup could suggest that the model is underperforming for that group, potentially due to bias in the training data. For

example, if a model performs significantly better on one class compared to another, this might indicate a bias in the data or the model.
Downside:

- **Manual Annotation Required:** Supervised learning requires labeled data, which means that biases within the data must often be identified and labeled manually. This can be a labor-intensive process, especially for large or complex datasets.
- **Correlation with Specific Class:** For example, if a model predicting job hiring decisions relies heavily on a feature like "name," it might indicate bias based on name-related cultural or ethnic assumptions.
- **Bias coming from feature importance**

2.1.4 Clustering to identiy bias

- Clustering was used as a per-processing technique to inform the resampled training data smaller sets of cluster that represents smaller subgroups.
- K-means clustering has been used to identify biases in data by grouping similar data points into clusters, which can inform resampling strategies.
Downside:
- K-means clustering struggles to scale effectively with high-dimensional data like images, where each data point might have thousands of features (pixels). In such cases, the **concept of a "cluster" becomes less meaningful**, and k-means may fail to capture the true underlying structure of the data.
- K-means clustering requires significant **pre-processing** to extract relevant features from the data before clustering can be applied. This preprocessing can be complex, especially for high-dimensional data, and may require **domain expertise** to ensure meaningful feature extraction.

2.2 The Proposed Method of Uncovering and Mitigating Algorithmic Bias

- The prposed method takes all of the downsides from the previous section and tries to address them all.
- So simply said: this solution does NOT require any data pre-processing, nor any annotation, nor prior training, nor prior testing, nnor artificail data, rather it focuses on adaptive learning (learning along the way) the latent features, and to re-sample from the original data, it learns these latent featues in an unsupervised manner.
- The method uses a novel approach (a modified one) of VAEs. Which we will refer to as (as in the orignal paper) DB-VAE.

- To understand this algorithm we will provide the pseudo-code as in the paper, and then explain how it works.

Algorithm 1 Adaptive re-sampling for automated debiasing of the DB-VAE architecture

Require: Training data $\{X, Y\}$, batch size b

1: Initialize weights $\{\phi, \theta\}$
2: **for** each epoch E_t **do**
3: Sample $z \sim q_\phi(z|X)$
4: Update $\hat{Q}_i(z_i(x)|X) \leftarrow \prod_i \frac{1}{\hat{Q}_i(z_i(x)|X)+\alpha}$
5: $\mathcal{W}(z(x)|X) \leftarrow \prod_i \frac{1}{\hat{Q}_i(z_i(x)|X)+\alpha}$
6: **while** iter $< \frac{n}{b}$ **do**
7: Sample $x^{\text{batch}} \sim \mathcal{W}(z(x)|X)$
8: $\mathcal{L}(\phi, \theta) \leftarrow \frac{1}{b} \sum_{i \in x^{\text{batch}}} \mathcal{L}_i(\phi, \theta)$
9: Update: $[w \leftarrow w - \eta \nabla_{\phi,\theta} \mathcal{L}(\phi, \theta)]_{w \in \{\phi,\theta\}}$
10: **end while**
11: **end for**

- So by ϕ we mean the weights of the encoder, and by θ we mean the weights of the decoder.
- Line 3: Encoding the input data (X) into the latent space using the params ϕ, in other words for each x, we have $[z_1(x), \ldots, z_k(x)]$
- Line 3: Trivial(update each $\hat{Q}_i(z_i(x)|X)$ based on this x that is being processed)(i.e. updating the histograms)
- Line 5: this distribution is used to sample for this whole epoch, i.e. all of the mini batches of size b will be sampled from this distribution, until our next epoch. Well you might ask why: first it is more efficient because updtating for each batch can be computationally expensinve. It also allows the model to focus on a balanced subset of data points
- Line 6: To mimic the feeling of passing through the whole n points
- Line 7: this batch is of size b
- Line 8: How well the model is performing given the current x_{batch}, based on the weights ϕ and θ. Note that each $\mathcal{L}_i$ can take two forms: only $\mathcal{L}_y(y, \hat{y})$ or the whole expression depending on the class at hand
- Line 9: Similar to SGD, adam, GD,etc. But notice here that we also need to take special care of the case of negative classes we need to update only ϕ in that case(gradient of decoder and latent space being excluded), and in the case of positive classes we need to update both ϕ and θ.
- Once these weights are updated we call it one time step, meaning in theory for ach epoch we will have $\frac{n}{b}$ time steps.