Removal of Batch Effects using Generative Adversarial Networks

Uddeshya Upadhyay, Arjun Jain
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

Abstract
Many biological data analysis processes like Cytometry or Next Generation Sequencing (NGS) produce massive amounts of data which needs to be processed in batches for down-stream analysis. Such datasets are prone to technical variations due to difference in handling the batches possibly at different times, by different experimenters or under other different conditions. This adds variation to the batches coming from the same source sample. These variations are known as Batch Effects. It is possible that these variations and natural variations due to biology confound but such situations can be avoided by performing experiments in a carefully planned manner. Batch effects can hamper down-stream analysis and may also cause results to be inconclusive. Thus, it is essential to correct for these effects. Some recent methods propose deep learning based solution to solve this problem. We demonstrate that this can be solved using a novel Generative Adversarial Networks (GANs) based framework. The advantage of using this framework over other prior approaches is that here we do not require to choose a reproducing kernel and define its parameters. We demonstrate results of our framework on a Mass Cytometry dataset.

Introduction
Batch effects are technical sources of variation that have been added to the samples during handling. They are common in many biological data analysis pipelines as many such experiments require the sample to be divided in different batches and such effects might get introduced at the time of creation of these batches due to variation in environmental conditions such as temperature, instruments or other experimenter related conditions. Although these are not the only sources of variation which may cause the batches to differ but not correcting for them will lead to different outputs from different batches of identical samples and the experiments will be rendered inconclusive.

Currently intuition and domain knowledge from the expert side is used to identify the candidate underlying parameters which might have caused such batch effects. One of the preliminary tests to check if a particular underlying variable is the cause for variation is to plot the projection of the batches on few principle components and mark them differently on the basis of different value of concerned variable, if the points separate out then clearly the given variable is responsible for the effect (Reese et al. 2013).

Related Work
Some of the recent deep learning based methods to solve this problem utilize residual networks to learn a near identity mapping from source to target by optimizing the Maximum Mean Discrepancy (MMD) between the transformed source and original target (Shaham et al. 2017, Dziugaite, Roy, and Ghahramani 2015, Li, Swersky, and Zemel 2015). MMD is one the several methods used to quantify the distance between two continuous distributions. It uses the distance measure between the means of distribution in some transformed space as a proxy to distance measure between distributions in original space. Let $P$ and $Q$ be two distributions over set $A$, let $\phi : A \rightarrow H$ be a transformation from $A$ to reproducing kernel Hilbert space $H$ then MMD is defined as

$$MMD(P, Q) = \| E_{X_1 \sim P}(\phi(X_1)) - E_{X_2 \sim Q}(\phi(X_2)) \|$$ (1)

MMD depends upon the choice of reproducing kernel and hence one needs to devise a method to find the optimum kernel parameters. We propose a solution based on a Generative Adversarial Network (GAN) and report our results on a mass cytometry dataset which was used for a similar study before. Since our method does not involve reproducing kernel, we don’t have to discover and specify kernel parameters explicitly.

Methods

Visualizing Batch Effects
Figure 1 shows an example where a dataset consisting various biological groups has been plotted in 2-dimensional space by taking the projection on first two major principle components. Figure 1(a) shows that while plotting the population without any markers does not reveal a discernible pattern while plotting the same points marked with colour representing their group (in this case population consists of four different groups) shows that each group is well mixed in the three different clusters distinctly visible. The underlying cause of such clustering is sampling date and this is clear if we plot the points marking each point according to sampling dates. The three different dates correspond to three different
clusters\footnote{Visualization taken from \url{http://www.molmine.com/magma/global_analysis/batch_effect.html}}. In general plotting projections along a few principle components and marking the points according to different underlying variables can help detect the cause of batch effects. However, it requires the domain knowledge and intuition of an analyst to hypothesize which variable might be causing such effects.

**Generative Models**

Generative models take training data i.e. samples from $P_{data}$ (which is unknown) and learn to represent the unknown distribution. Some models do this explicitly by estimating the unknown distribution using $P_{model}$. This can be done in many ways such as modelling $P_{data}$ by a family of known distributions and then estimating the parameters of the family by optimization techniques such as Maximum Likelihood Estimate (MLE). Other such explicit methods are Variational Autoencoders, Boltzmann Machines, etc.

Often it suffices to not have an explicit representation of $P_{data}$ in the form of $P_{model}$ but to be able to produce samples from $P_{model}$ which approximate $P_{data}$. Generative Adversarial Networks (GANs) are one such approach where $P_{model}$ does not estimate $P_{data}$ directly, but the trained network can generate new samples from $P_{model}$ which approximates $P_{data}$ as explained in (Goodfellow 2016). We provide details of how this indirect method can be useful for our problem in the following sections.

**Generative Adversarial Networks**

Generative Adversarial Network (GAN) (Goodfellow et al. 2014) is a framework where two artificial neural networks compete against each other. One of them is a Generator ($G$) with parameters $\theta_G$ and the other is a Discriminator ($D$) with parameters $\theta_D$. Typically generator takes an input $z \sim P_{prior}$ where $P_{prior}$ is a simple prior distribution like Gaussian and outputs a value $x \sim P_{model}$ where $P_{model}$ tries to approximate $P_{data}$. Generator minimizes the cost function $C_G(:, \theta_G; \theta_D)$ which depends upon the parameters of $G$ as well as $D$.

The Discriminator takes input $x \sim P_{data} \cup P_{model}$, i.e. inputs are sampled from both training data and the output produced by generator. Let $x_{real} \sim P_{data}$ and $x_{fake} \sim P_{model}$, then the task of discriminator is to distinguish between the input coming from $P_{data}$ ($x_{real}$) and generated by generator $x_{fake}$. This is achieved by maximizing the cost function $C_D(:, \theta_G; \theta_D)$. Equation (2) and (3) describes $C_G$ and $C_D$. This arrangement sets up a min-max game between $G$ and $D$.

\begin{align}
C_G(z; \theta) &= -E_{z \sim P_{prior}} \log D(G(z)) \tag{2} \\
C_D(x; \theta) &= -E_{x \sim P_{data}} \log D(x) - E_{z \sim P_{prior}} \log (1-D(G(z))) \tag{3}
\end{align}

Here $\Theta = \{\theta_G; \theta_D\}$. Equation (2) here uses the non-saturating heuristic in order to train network efficiently (Goodfellow 2016).

**Batch effect correction using GANs**

We consider two data batches, source ($S$) and target ($T$) coming from the identical initial samples but prepared under different conditions. We assume that the experiments were performed in a careful manner, this can minimize confounding and leave batch effect as the prime reason for variations across batches. Removal of these batch effects can be posed as problem of finding a map $\Psi$ such that

\begin{equation}
\Psi(x) \in T \ \forall x \in S \tag{4}
\end{equation}

To use a GAN for finding such a mapping $\Psi$ we frame the problem in the following manner.

Let source ($S$) and target ($T$) have an underlying distribution $P_S$ and $P_T$ (both of which are unknown). We set up a

![Figure 1: PCA plots showing samples from a dataset consisting of four biological groups. (a) shows the projections without any markers, (b) shows projections marked according to their biological group, it is clear that there are 3 major clusters and each cluster has good mix of every biological groups. (c) shows projections marked with sampling dates and it is evident that date is the factor introducing batch effects.](http://www.molmine.com/magma/global_analysis/batch_effect.html)
GAN such that the generator takes input $z \sim P_S$ and produces output $x_{fake} \sim P_{model}$. Discriminator learns to discriminate between $x_{real} \sim P_T$ and $x_{fake} \sim P_{model}$. Then $\Psi$ can be obtained by training the GAN and using the generator $G$ to produce $x_{fake}$ from $z$, networks are trained to get the parameters $(\theta^*_G, \theta^*_D)$

$$
\theta^*_D = \underset{\theta_D}{\arg\min} \left[ -\left( E_{x \sim P_T} \log D(x) + E_{z \sim P_S} \log (1-D(G(z))) \right) \right] 
$$

$$
\theta^*_G = \underset{\theta_G}{\arg\min} \left[ -E_{z \sim P_S} \log D(G(z)) \right] 
$$

The network is trained in such a manner that while updating the weights of discriminator we fix the weights of the generator and vice-versa as discussed in next section.

**Experiments and Results**

**Dataset**

Mass Cytometry is a process to analyze the properties of the cells by labelling different proteins of the cell with antibodies conjugated with isotopically pure metals. These labelled cells are then nebulized and metal conjugated antibodies are ionised, such metal signals are analyzed to determine the properties of the cells.

The dataset consists of Peripheral Blood Mononuclear Cells (PBMC) samples from two Multiple Sclerosis (MS) patients. Samples were collected on two different days, 90 days apart (baseline and after Gilenya treatment). Samples were cryopreserved. Each sample was divided in two batches on two different days and one of them was stimulated with PMA/ionomycin. Batches prepared on first and second day were treated as target and source respectively. Pre-processing of the dataset was done in identical manner as explained in (Shaham et al. 2017; Finck et al. 2013).

**Architecture and Adversarial Training**

Our framework consists of two networks, a generator $G$ and a discriminator $D$. Generator consists of batch-norm layers (Sergey Ioffe 2015), linear layers and residual skip (He et al. 2016a) connections as these will be essential to learn a mapping which is close to an identity mapping (Shaham et al. 2017; He et al. 2016b). Our generator is required to map a collection of points from target to source batches, both of which are 25-dimensional therefore $G$ is designed such that input-output dimensions match. $G$ is shown in Figure 4. Discriminator $D$ is designed such that it takes the input form higher dimensional space (25-dimensional in this dataset) and outputs a scalar quantity. It consists of batch-norm layers and linear layers and applies sigmoid activation before outputting the scalar. $D$ is shown in Figure 5.

To train both the networks, at every iteration 256 points were sampled randomly from target and source batches. The group of points from source batch is passed through the generator which outputs “fake samples”, this along with “real samples” (from target batch) is passed through the discriminator with labels of “real samples” and “fake samples” set to 1 and 0 respectively. Discriminator optimizes for equation 3. Generator uses the “fake samples” with labels set to 1 and optimizes for Equation 2. This is done in sequence hence while training $D$ parameters of $G$ are fixed and vice-versa. We used adam (Kingma and Ba 2015) optimizer with (learning rate, $\beta_1, \beta_2$) set to $(1e-3, 0.9, 0.999)$ respectively.

**Results**

Figure 2 shows the PCA plots of the dataset (using first two major principle components). In total there were eight batches (2 patients $\times$ 2 conditions $\times$ 2 days) and four

![Figure 2: PCA plots of four source-target pairs. Blue and Maroon represents source and target respectively. Top-row [a1-d1] represents data before calibration, each plot has data from the same patient under same conditions but batches were created on two different days. Note the variations in batches. Bottom-row [a2-d2] represents data after calibration using a trained GAN.](image-url)
Figure 3: Box plots comparing the MMD Loss between the source and target pairs, pre-calibration and post-calibration using GAN. 256 points were sampled from source and targets to calculate the loss and this was repeated 100 times to report the final plots.

Conclusions
We present a novel solution to correct for batch effects which are very frequent in biological data analysis using Generative Adversarial Networks. Results of our experiments conclude that GANs can be used for this task without performing any explicit kernel based computation and thus reducing the need for domain knowledge of an expert or the intuition on an analyst to define the kernel or other hyper-parameters.

References
[Dziugaite, Roy, and Ghahramani 2015] Dziugaite, G. K.; Roy, D. M.; and Ghahramani, Z. 2015. Training generative neural networks via maximum mean discrepancy optimization. In Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI’15, 258–267.

[Finck et al. 2013] Finck, R.; Simonds, E. F.; Jager, A.; Krishnaswamy, S.; Sachs, K.; Fantl, W.; Pe’er, D.; Nolan, G. P.; and Bendall, S. C. 2013. Normalization of mass cytometry data with bead standards. Cytometry Part A 83(5):483–494.

[Goodfellow et al. 2014] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In Advances in neural information processing systems (NIPS), 2672–2680.

[Goodfellow 2016] Goodfellow, I. 2016. Generative adversarial networks. In Advances in Neural Information Processing Systems (NIPS) Tutorial.

[He et al. 2016a] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.

[He et al. 2016b] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In European conference on computer vision, 630–645. Springer.

[Kingma and Ba 2015] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR).

[Li, Swersky, and Zemel 2015] Li, Y.; Swersky, K.; and Zemel, R. 2015. Generative moment matching networks. In International Conference on Machine Learning (ICML), 1718–1727.

[Reese et al. 2013] Reese, S. E.; Archer, K. J.; Therneau, T. M.; Atkinson, E. J.; Vachon, C. M.; de Andrade, M.; Kocher, J.-P. A.; and Eckel-Passow, J. E. 2013. A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal component analysis. Bioinformatics 29(22):2877–2883.

[Sergey Ioffe 2015] Sergey Ioffe, C. S. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning (ICML), 448–456.

[Shaham et al. 2017] Shaham, U.; Stanton, K. P.; Zhao, J.; Li, H.; Raddassi, K.; Montgomery, R.; and Kluger, Y. 2017. Removal of batch effects using distribution-matching residual networks. Bioinformatics 33(16):2539–2546.