Feature Selection in the Presence of Monotone Batch Effects

Peng Dai [1] **Sina Baharlouei** [1] **Taojian Tu** [2] **Bangyan L. Stiles** [2] **Meisam Razaviyayn** [1] **Sze-chuan Suen** [1]

Abstract

We study the problem of feature selection in the presence of monotone batch effects when merging datasets from disparate technologies and different environments affects the underlying causal dependence of data features. We propose two novel algorithms for this task: 1) joint feature selection and batch effect correction through transforming the data batches using Generative Adversarial Networks (GANs); 2) transforming data using a batch-invariant characteristic (i.e., feature rank) to append datasets. We assess the performance of the feature selection methods used in conjunction with our batch effect removal methods. Our experiments on synthetic data show that the former method combined with Lasso improves the F_1 score significantly, even with few samples per dataset. This method outperforms popular batch effect removal algorithms, including Combat-Seq, Limma, and PCA. Comparatively, while the ranking method is computationally more efficient, its performance is worse due to the information loss resulting from ignoring the magnitude of data.

1. Introduction

Public-use datasets are becoming increasingly popular as funding requirements, data transparency, and open-source culture push researchers to share gathered data. Aggregation of related datasets allows for greater statistical power during analysis, particularly in the biological and genomics data contexts, where each experiment may only contain a few samples due to high experimental costs.

Merging datasets from disparate environments comes with challenges, as datasets from the same environment may be subject to similar biases. For example, differences in genome sequencing machines (Pareek et al., 2011), hybridization protocols (Young et al., 2020), and transformation methods (Robinson & Oshlack, 2010; Risso et al., 2014) may lead to batch effects (i.e. systematic non-biological differences between batches of samples) in gene expression data. Batch effects can harm the performance of statistical inference algorithms (in particular, feature selection for detecting useful biomarkers) by imposing bias on the predictions, increasing the false discovery rate, and reducing prediction accuracy (Sims et al., 2008). Thus, detection and removal of batch effects are crucial pre-processing stages of the statistical analysis of various bioinformatics tasks such as (single-cell) RNA sequencing (Chen et al., 2011), metabolomics analysis (Liu et al., 2020), and cancer classification (Almugren & Alshamlan, 2019; Leek et al., 2010).

Prior literature has studied the problem of mitigating batch effects. These methods can be categorized into several general categories. Clustering and KNN-based methods remove batch effects by finding the common sources of variations among datasets based on the proximity of data points (Butler et al., 2018; Zhang et al., 2019; Li et al., 2020; Fang et al., 2021; Lakkis et al., 2021). More specifically, each data batch is considered a cluster and the batch effect is viewed as the between-cluster variances (Fang et al., 2021). These methods are computationally expensive (due to computing pairwise distances of data points) and might not perform well if the batch effect changes the original distances of data points drastically and differently among different batches. Another collection of methods is based on reducing the dimension of the original data by removing the batch effects as spurious dimensions in data (Haghverdi et al., 2018; Alter et al., 2000; Butler et al., 2018). A common disadvantage of such methods is that by projecting different batches onto the same low-dimensional space, valuable information contributing to the inference phase may be lost. Yet another class of approaches formulate the batch effect problem as a parametric model and estimate the unknown parameters by classification or regression techniques (Johnson et al., 2007; Lazar et al., 2013; Leek, 2014; Risso et al., 2014; Robinson et al., 2010; Love et al., 2014; Zhang et al., 2020). The most popular method in this category is arguably ComBat-seq (Zhang et al., 2020), which considers a joint negative binomial regression model for gene counts and the batch effects. Their approach adjusts the parameters of the model by matching the cumulative density functions of the data

[1]Industrial and Systems Engineering Department, University of Southern California, Los Angeles, CA. [2]Department of Pharmacology and Pharmaceutical Sciences, University of Southern California, Los Angeles, CA.. Correspondence to: Peng Dai <pengdai@usc.edu>.

Published at the ICML 2023 Workshop on Spurious Correlations, Invariance, and Stability. Honolulu, Hawaii, USA.

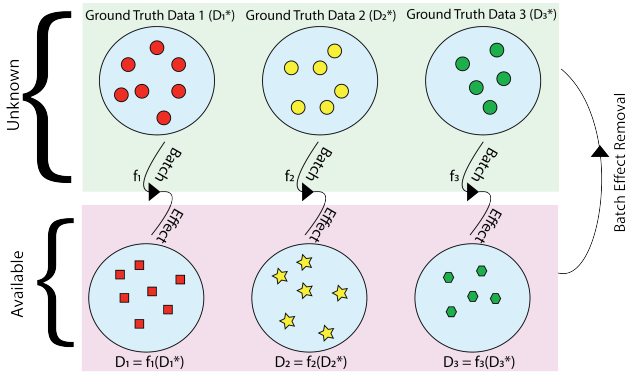


Figure 1: The batch effect can be conceptualized as transformations on each of the ground-truth datasets which changes the distribution of data in each in a potentially different way.

generated by the model and the original data. The common problem of these approaches is the strict parametric assumptions on the data distribution (such as negative binomial) and the model (linearity) that limit their applicability. A more recent class of methods remove the batch effect by matching the empirical distribution of different batches by minimizing a distribution distance measure such as Kullback-Leibler (KL) Divergence (Lakkis et al., 2021) or maximum mean discrepancy (MMD) (Shaham et al., 2017; Niu et al., 2022). Shaham et al. (2017) mitigates batch effect by minimizing the Maximum Mean Discrepancy (MMD) between the two distributions of samples. They consider one dataset as the reference, and the other dataset is transformed to have a similar distribution as the reference dataset. While their algorithm works for two input datasets, our methodology utilizes all input datasets to reduce the batch effect on all of them simultaneously. Another significant difference between our framework and theirs is that we perform joint batch removal and feature selection together instead of doing these steps independently. As discussed, removing batch effects jointly with feature selection can significantly enhance the feature selection accuracy. The reason is many feature selection strategies (such as the popular LASSO algorithm) make certain assumptions on the dataset (e.g., generalized linear models). Such assumptions may be invalidated when the batch effect removal procedure does not account for the downstream feature selection procedure.

Feature Selection Methods: Feature selection when no batch effect is involved is extensively studied in the literature (Saeys et al., 2007; Jović et al., 2015). A popular approach selects features with the highest correlation (e.g., Pearson or Spearman (He & Yu, 2010)) with the target variable. To choose the most relevant features, one can then compute the p-value for each computed correlation and choose the ones with the highest correlation after adjusting with false discovery rate control mechanisms such as the Benjamini-Hochberg procedure (Benjamini & Hochberg, 1995). Notice that one can rely on exact statistical approaches such as permutation tests to avoid distributional assumptions. A more popular approach for finding the most relevant features with the highest prediction power of the target variable is to formulate the problem as a Lasso regression task. The output of the method is the features corresponding to the non-zero elements of the regression parameter vector (Vinga, 2021).

In this work, we handle the feature selection task and batch effect removal through transformation of the data jointly. In particular, we remove the batch effect by finding the optimal transformations that minimize the maximum mean discrepancy (MMD) of different data batches.

2. Problem Formulation and Methodology

Let us first rigorously define the problem of distribution-free feature selection in the presence of batch effects: let $\mathcal{D}_1, \ldots, \mathcal{D}_m$ be a collection of datasets from m unique laboratories studying the interactions of d input features (e.g. gene expressions, proteomics data, etc) and a target variable y. Ideally, all datasets in the collection follow the identical distribution P^* describing the joint distribution of input features and the target variable. However, due to the aforementioned factors in the previous section known as the batch effect, the datasets in the collection do not follow the ground-truth distribution P^*. Formally, the batch effect can be described as m different transformations $f_1, \ldots, f_m$ applied to the ground-truth datasets $D_1^*, \ldots, D_m^* \sim P^*$ (see Figure 1). **We assume that all functions $f_i, i = 1, \ldots, m$, are monotonically increasing.** Thus each observed dataset can be viewed as a transformation of the ground-truth data:

$$\mathcal{D}_i = f_i(\mathcal{D}_i^*) \quad i = 1, \ldots, m. \tag{1}$$

The goal of batch effect removal is to learn the underlying transformations $f_1, \ldots, f_m$ on the data batches such that the ground-truth datasets $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are recovered up to bijective mappings (see Figure 1). In other words, the optimal transformations on the datasets make the distributions of them as *close* as possible to each other. Thus, one can quantify the quality of batch effect removal based on how close the resulting distributions of datasets are after the transformations. Note that it is crucial for transformations to be bijective. Otherwise, one can transform all datasets to zero, and the difference between the transformed datasets will be zero. However, such transformations are not desirable. Unfortunately, finding the optimal transformations over the set of bijective functions is challenging from the optimization point of view. We propose a novel formulation in the next section to avoid such spurious solutions.

2.1. MMD-based Approach

We utilize the maximum mean discrepancy (Gretton et al., 2012) to measure the proximity of dataset distributions (Arjovsky et al., 2017). Maximum Mean Discrepancy (MMD) is a statistical measure that quantifies the dissimilarity between probability distributions. Given two datasets or distributions, MMD aims to determine the dissimilarity by calculating the discrepancy between their respective means in the Reproducing Kernel Hilbert Space (RKHS). Mathematically, let z and z' be independent random variables following distribution p, and w as well as w' independent random variables following distribution q. Then, the MMD between distributions p and q is

$$\mathrm{MMD}^2[p,q] = \mathbf{E}_{z,z'}\left[k\left(z,z'\right)\right] - 2\mathbf{E}_{z,w}[k(z,w)] + \mathbf{E}_{w,w'}\left[k\left(w,w'\right)\right]$$

An empirical estimate of MMD is given by (Gretton et al., 2006)

$$\mathrm{MMD}_u^2[x,y] = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k\left(x_i,x_j\right) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k\left(y_i,y_j\right) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k\left(x_i,y_j\right),$$

where $k(\cdot,\cdot)$ is a non linear kernel function. In our work, we apply the Gaussian kernel, defined as

$$k(z,w) = \exp\left(\frac{-|z-w|^2}{2\sigma^2}\right).$$

Here, σ is a hyper-parameter to control the width of the kernel. To accurately estimate MMD and to avoid the vanishing gradient phenomenon, σ takes a list of values. We set $\sigma = [10^{-2}, 10^{-1}, \ldots, 10^{11}]$, which is proper for even large distributional differences.

We assume the predictor x^* without batch effect follows a multi-variate Gaussian distribution $\mathcal{N}(\mu,\Sigma)$ with the zero mean and an unknown covariance matrix. Therefore, we can formulate the batch effect removal problem as finding the optimal transformation and covariance matrix Σ such that the total difference of transformed distributions and the underlying normal distribution is minimized:

$$\min_{\mathbf{\Phi},\Sigma} \Sigma_{i\in E}\mathrm{MMD}\Big(\mathbf{\Phi}_i(\mathcal{D}_i), \mathcal{N}(0,\Sigma)\Big), \tag{2}$$

where $\mathrm{MMD}(\cdot,\cdot)$ measures the maximum mean discrepancy between the two input distributions/datasets. Each $\mathbf{\Phi}_i$ transformation is modeled by a two-layer neural network. We add a non-negativity constraint on the weights of the neural networks to ensure the corresponding transformations are monotone. To avoid spurious solutions, we unify the feature selection (Lasso regression) and the batch effect removal task into the following optimization problem:

$$\min_{\mathbf{\Phi},\boldsymbol{\theta}} \quad \frac{1}{n}\sum_{i=1}^{n}\sum_{(\mathbf{x},y)\in\mathcal{D}_i} (\boldsymbol{\theta}^T\Phi_i(\mathbf{x}) - y)^2 + \lambda\|\boldsymbol{\theta}\|_1 + \mu\sum_{i=1}^{m}\mathrm{MMD}\Big(\Phi_i(\mathcal{D}_i), \mathcal{N}(0,\Sigma)\Big). \tag{3}$$

We simply the model by generating Σ randomly and do not incorporate it into the training process. Here, the first term in the objective function minimizes the regression loss; the second term aims to remove the batch effect, and the last term imposes sparsity on the weights of the linear model. Problem (3) can be optimized using a first-order iterative algorithm where at each iteration, a step of projected gradient descent is applied on the parameters of the transformations and then $\boldsymbol{\theta}$ is updated by the proximal graident method. The details of the algorithm is relegated to Appendix A. In Appendix B, we propose a simple approach based on the order-statistics transformation ("Ranking method"). This approach is less computationally intensive compared to the MMD-based approach; however, as we observe in our numerical experiments, the performance of MMD-based approach is drastically superior. We additionally consider a Low-Rank MMD method, described in the next section, for enhanced performance.

2.2. Low-Rank MMD Method

Problem (3) consists of many optimization parameters in the high dimensional setting when the number of data points is very limited compared to the dimension of the data ($n \ll d$), In particular, the unknown covariance matrix Σ has $\mathcal{O}(d^2)$ parameters to optimize. For instance, when $d = 100$ (or even larger for genomic datasets), A contains 10,000 variables, making it difficult to train. A practical approach is considering a multi-Gaussian distribution with the randomly generated covariance matrix as the reference distribution.

The common approach for generating a zero-mean multi-Gaussian distribution ($X \sim \mathcal{N}^{n\times d}(0,\Sigma)$) is by first generating a standard normal Gaussian distribution $Z \sim \mathcal{N}^{n\times d}(0, I^{d\times d})$ and then generating a random matrix $A^{d\times d}$. The resulting multi-Gaussian distribution follows a covariance matrix of A^TA. In the problem context, the matrix A is the variable matrix that needs to be trained.

Although our simulation results show that this method beats the state-of-the-art approaches in the literature, the randomly generated covariance matrix can be arbitrarily far from the covariance matrix of the true data distribution. Thus, to overcome the mentioned limitation of the random generation of the covariance matrix, we assume that the ground-truth covariance matrix is low-rank. This assumption reduces the number of optimization parameters significantly. Further, in most high-dimensional biological datasets, the majority of

feature pairs are almost independent. Thus, the ground-truth matrix is sparse in practice making the low-rank assumption on the covariance matrix of the data practically reasonable.

A can be generated with dimension $s \times d$ where $s \ll d$. Correspondingly, $Z \sim \mathcal{N}^{n\times s}(0, I^{s\times s})$ and then $X = AZ$. Consequently, the optimization problem is modified to include the low-rank matrix as follows:

$$\min_{\boldsymbol{\Phi},\boldsymbol{\theta},\boldsymbol{A}} \quad \frac{1}{n}\sum_{i=1}^{n}\sum_{(\mathbf{x},y)\in\mathcal{D}_i} \ell\Big(h_{\boldsymbol{\theta}}\big(\boldsymbol{\Phi}_i(\mathbf{x})\big), \boldsymbol{\Phi}_i(y)\Big) + \mu\sum_{i=1}^{m}\mathrm{MMD}\Big(\Phi_i(\mathcal{D}_i), AZ\Big) + \lambda\|\boldsymbol{\theta}\|_1. \quad (4)$$

This modified approach is referred to as the "Low-Rank MMD" method, which is implemented and evaluated on synthetic datasets. The results, shown in Table 1, indicate that both the low-rank MMD method and original MMD methods perform well in two of the four scenarios. However, the low-rank method offers additional advantages and potential for further exploration. One of the key benefits of the low-rank approach is its increased explainability. By incorporating a low-rank matrix, the model becomes more interpretable, allowing a better understanding of the underlying factors influencing the data. Furthermore, the low-rank method demonstrates greater adaptability as the number of samples changes.

In particular, as we vary the low-rank parameter s according to the number of samples, the low-rank model exhibits enhanced performance. This flexibility allows the model to effectively capture the underlying patterns and dependencies in the data, resulting in improved predictive power. By adjusting the low-rank parameter dynamically, the low-rank method can leverage the available information and adapt to different dataset sizes.

We train the neural networks ($\boldsymbol{\Phi}$) and linear coefficients ($\boldsymbol{\theta}$) via Adam optimizer with full batch to minimize the loss function described as 3.

3. Numerical Experiments

In this section, we evaluate the performance of MMD-based and ranking-based methods on simulated datasets. To do this, we first consider the feature selection tasks in the presence of distributional shifts. For this task, we measure performance using the $\mathcal{F}1$ score which is defined as the harmonic mean of recall and precision. Moreover, to evaluate the effectiveness of the methods in removing batch effects, we visualize the data batches before and after the transformation.

3.1. Simulated Data

To evaluate the performance of our proposed methods against state-of-the-art baselines in the literature, we generate datasets with different number of batches m and the number of data points n in each batch. To this end, we generate mn data points with dimension $d = 100$. Each batch follows a normal distribution with a randomly assigned chosen mean and covariance. The target variable y is a linear function of the input data plus a mean zero normal noise ($y = \mathbf{x}^T\beta^* + \epsilon$). To induce sparsity, each entry of β^* is set to zero with the probability of 90%. To add batch effects to batch k, we transform the dimension j in data point i in batch k as follows:

$$x'_{ij} = a_k x_{ij}^{\frac{5}{3}} + b_k x_{ij} + c_k + \epsilon_k$$

where a_k, b_k and c_k are randomly generated positive numbers for batch k and ϵ_k is Gaussian noise. We vary the number of data points and data batches to evaluate the effectiveness of each approach in different scenarios. Besides MMD-based and ranking-based methods proposed in this paper, we evaluate several state-of-the-art approaches including CombatSeq (Zhang et al., 2020), Limma (Smyth, 2005), zero-mean-unit-variance, and PCA (Leek et al., 2010). We also compare outcomes when we apply Lasso on the datasets without moving batch effects. Table 1 reports the $\mathcal{F}1$ score for the aforementioned approaches in four different scenarios.

(m, n)	S1 (5,10)	S2 (50,10)	S3 (5,100)	S4 (50,100)
Combat-Seq	0.424	0.313	0.444	0.759
Limma	0.077	0.109	0.217	0.238
PCA	0.143	0,089	0.228	0.238
Zero-Mean Unit-Variance	0.061	0.204	0.231	0.16
Original Data	0.381	0.145	0.289	0.289
Ranking	0.444	0.095	0.214	0.214
Shaham	0.326	0.143	0.289	0.297
MMD	**0.410**	**0.727**	**0.880**	**0.857**
Low-Rank MMD	**0.537**	**0.400**	**0.857**	**0.909**

Table 1: $\mathcal{F}1$ Scores for Different Methods and Scenarios

3.2. Interpreting the Results

From Table 1, it is evident that the MMD-based approach performs significantly better than other methods by a large margin in scenario 2 to 4. In Scenario 1, Combat-Seq works slightly better. This can be attributed to the requirement in the MMD method that there be an adequate number of samples to obtain the optimal transformations (modeled by two-layer neural networks) effectively. Conversely, other benchmarks perform even worse than applying Lasso on the original datasets. This suggests that these approaches may alter the underlying information within the datasets.

Additionally, the ranking method does not effectively select the correct features, potentially leading to the loss of crucial information.

3.3. Convergence of the Training Approach

Based on Equation (3), the supervised learning loss, MMD loss and $\mathcal{L}1$ norm are expected to decrease during the training process as we minimize the objective. Figure 2 plots the training process for scenario 3, the upper left is the logarithmic prediction loss or supervised training loss, the upper right is the $\mathcal{L}1$ norm of θ, the lower left is the sum of MMD between transformed data and reference data, the lower right is the objective value in (3). We can see overall the loss shows a declining trend. Small MMD loss indicates the batch effects are corrected, and the transformed data's distributions are close to that of the reference data. Figure 3 shows the MMD between each pair of datasets before and after transformation in scenario 3. The diagonals are black (MMD=0) because the statistical distance between a dataset and itself is 0.

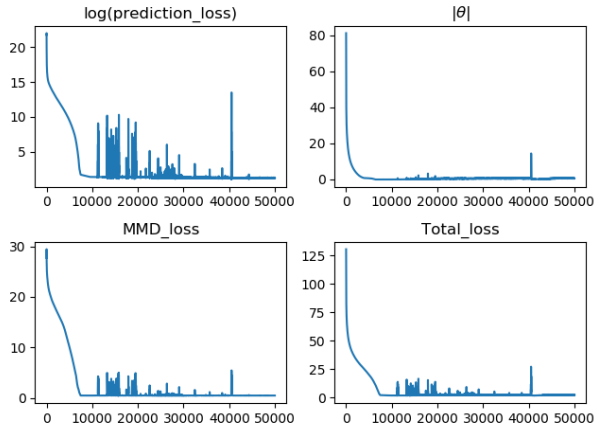
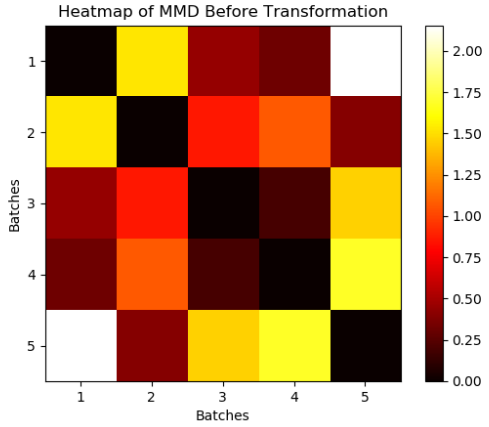
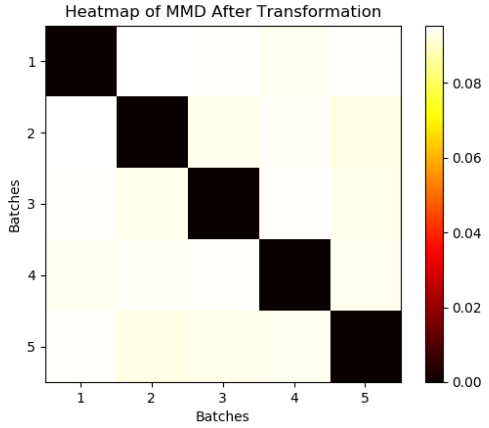


Figure 2: Training Loss

Non-diagonal values in Figure 3a are roughly between 1 and 2 whereas the post-transformation values in Figure 3b are smaller than 0.1, showing that the batch effects have been corrected. In Appendix C, we further discuss the convergence behaviour and quality of the transformed datasets obtained by our MMD-based approach. We additionally describe the critical process of tuning hyper-parameters λ and θ in Appendix D. All codes and outcomes are publicly available at `https://rb.gy/jehdu`.

Conclusion: We proposed a joint optimization framework for feature selection using Lasso and removing batch effects by matching the distributions of datasets using MMD. Aside from feature selection, the method can be used as an effective tool to remove batch effects in a pre-processing task. The numerical experiments on different scenarios demonstrate the significant improvement in the performance of this approach compared to other state-of-the-art methods.





(b) Heat-map of MMD After Transformation

References

Almugren, N. and Alshamlan, H. A survey on hybrid feature selection methods in microarray gene expression data for cancer classification. *IEEE access*, 7:78533–78548, 2019.

Alter, O., Brown, P. O., and Botstein, D. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

Butler, A., Hoffman, P., Smibert, P., Papalexi, E., and Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature biotechnology*, 36(5):411–420, 2018.

Chen, C., Grennan, K., Badner, J., Zhang, D., Gershon, E., Jin, L., and Liu, C. Removing batch effects in analysis of expression microarray data: an evaluation of six batch adjustment methods. *PloS one*, 6(2):e17238, 2011.

Fang, Z.-Y., Lin, C.-X., Xu, Y.-P., Li, H.-D., and Xu, Q.-S. Rebet: a method to determine the number of cell clusters based on batch effect removal. *Briefings in Bioinformatics*, 22(6):bbab204, 2021.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Haghverdi, L., Lun, A. T., Morgan, M. D., and Marioni, J. C. Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 36(5):421–427, 2018.

He, Z. and Yu, W. Stable feature selection for biomarker discovery. *Computational biology and chemistry*, 34(4): 215–225, 2010.

Johnson, W. E., Li, C., and Rabinovic, A. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

Jović, A., Brkić, K., and Bogunović, N. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 1200–1205. Ieee, 2015.

Lakkis, J., Wang, D., Zhang, Y., Hu, G., Wang, K., Pan, H., Ungar, L., Reilly, M. P., Li, X., and Li, M. A joint deep learning model enables simultaneous batch effect correction, denoising, and clustering in single-cell transcriptomics. *Genome research*, 31(10):1753–1766, 2021.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lazar, C., Meganck, S., Taminau, J., Steenhoff, D., Coletta, A., Molter, C., Weiss-Solís, D. Y., Duque, R., Bersini, H., and Nowé, A. Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in bioinformatics*, 14(4):469–490, 2013.

Leek, J. T. Svaseq: removing batch effects and other unwanted noise from sequencing data. *Nucleic acids research*, 42(21):e161–e161, 2014.

Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., Geman, D., Baggerly, K., and Irizarry, R. A. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10):733–739, 2010.

Li, X., Wang, K., Lyu, Y., Pan, H., Zhang, J., Stambolian, D., Susztak, K., Reilly, M. P., Hu, G., and Li, M. Deep learning enables accurate clustering with batch effect removal in single-cell rna-seq analysis. *Nature communications*, 11(1):1–14, 2020.

Liu, Q., Walker, D., Uppal, K., Liu, Z., Ma, C., Tran, V., Li, S., Jones, D. P., and Yu, T. Addressing the batch effect issue for lc/ms metabolomics data in data preprocessing. *Scientific reports*, 10(1):1–13, 2020.

Love, M. I., Huber, W., and Anders, S. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):1–21, 2014.

Niu, J., Yang, J., Guo, Y., Qian, K., and Wang, Q. Joint deep learning for batch effect removal and classification toward maldi ms based metabolomics. *BMC bioinformatics*, 23 (1):1–19, 2022.

Ouyang, L. and Key, A. Maximum mean discrepancy for generalization in the presence of distribution and missingness shift. *arXiv preprint arXiv:2111.10344*, 2021.

Pareek, C. S., Smoczynski, R., and Tretyn, A. Sequencing technologies and genome sequencing. *Journal of applied genetics*, 52(4):413–435, 2011.

Risso, D., Ngai, J., Speed, T. P., and Dudoit, S. Normalization of rna-seq data using factor analysis of control genes or samples. *Nature biotechnology*, 32(9):896–902, 2014.

Robinson, M. D. and Oshlack, A. A scaling normalization method for differential expression analysis of rna-seq data. *Genome biology*, 11(3):1–9, 2010.

Robinson, M. D., McCarthy, D. J., and Smyth, G. K. edger: a bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics*, 26(1): 139–140, 2010.

Saeys, Y., Inza, I., and Larranaga, P. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

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

Sims, A. H., Smethurst, G. J., Hey, Y., Okoniewski, M. J., Pepper, S. D., Howell, A., Miller, C. J., and Clarke, R. B. The removal of multiplicative, systematic bias allows integration of breast cancer gene expression datasets–improving meta-analysis and prediction of prognosis. *BMC medical genomics*, 1(1):1–14, 2008.

Smyth, G. K. Limma: linear models for microarray data. *Bioinformatics and computational biology solutions using R and Bioconductor*, pp. 397–420, 2005.

Vinga, S. Structured sparsity regularization for analyzing high-dimensional omics data. *Briefings in Bioinformatics*, 22(1):77–87, 2021.

Young, A. P., Jackson, D. J., and Wyeth, R. C. A technical review and guide to rna fluorescence in situ hybridization. *PeerJ*, 8:e8806, 2020.

Zhang, F., Wu, Y., and Tian, W. A novel approach to remove the batch effect of single-cell data. *Cell discovery*, 5(1): 1–4, 2019.

Zhang, Y., Parmigiani, G., and Johnson, W. E. Combat-seq: batch effect adjustment for rna-seq count data. *NAR genomics and bioinformatics*, 2(3):lqaa078, 2020.

A. Algorithm for MMD Method

Algorithm 1 MMD-based feature selection and batch effect removal

1: Initialize $\boldsymbol{\theta}$ with normal distribution and $\boldsymbol{\Phi}_i$ randomly for all $1 \leq i \leq m$.
2: **for** $t = 1, \ldots, T$ **do**
3: Update the parameters in $\boldsymbol{\Phi}_i$ via Adam optimizer and set all negative weights to 0 for all $2 \leq i \leq m$.
4: Update $\boldsymbol{\theta}$ by applying one step of ISTA (Beck & Teboulle, 2009) on Problem (3).
5: Return features corresponding to non-zero elements in $\boldsymbol{\theta}$.
6: **end for**

B. Ranking Method

An alternative, less computationally intense approach to the MMD-based method is to do prediction by relying on features that are invariant under batch effect transformations $f_1, \ldots, f_m$. In particular, since these transformations are monotonically increasing, **they do not change the order of entries in any row of the data matrix.** Thus, the order statistic in each row is unchanged, which means the row-wise order statistics is invariant under the batch effects. After applying the order statistics, one can perform Lasso regression or other feature selection algorithms on the transformed dataset. Despite its simplicity, using the order statistics instead of the original data can lead to information loss, as it only considers the orders not the magnitude of data.

Algorithm 2 Feature Selection with Ranking-based Method

1: Convert original data (X, Y) to ranking data $\mathbf{R}(X, Y) = (X', Y')$ per row;
2: Apply Lasso on $\mathbf{R}(X, Y)$ and select corresponding features.

C. Validate Multi-variate Gaussian Distribution

In Figure 3b, we can observe that the Maximum Mean Discrepancy (MMD) of the transformed datasets are not equal to 0. We consider this to be a reasonable outcome because the reference dataset is generated randomly and is expected to differ from the true underlying distribution. The goal is for the transformed data to be close to a multivariate Gaussian distribution rather than an exact match to the reference data. If the MMD in Figure 3b were 0, it would indicate overfitting. Figure 4 displays the histograms and overlays the corresponding Gaussian curves for the original data and the transformed data achieved through linear mapping with the random vector $\mathbf{a}$. Notably, after the transformation, the data exhibits a clear Gaussian distribution, unlike the data prior to the transformation.

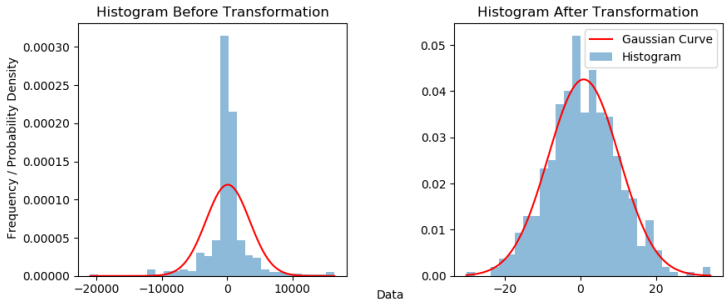


Figure 4: Histogram for Validating Gaussian Distribution

The transformed data does not fully converge to the reference data due to the design of the optimization problem 3. In Problem 3, we conceptualize minimizing the MMD as the main objective and the Lasso as a regularization term. Since the underlying distribution follows a sparse linear relationship, forcing the transformed data to be exactly the same as the reference data would lead to large supervised loss and $\mathcal{L}_1$ norm. Hence, appropriate tuning of hyper-parameters λ and μ becomes crucial in achieving good performance.

D. Hyper-Parameter Tuning

The selection of suitable hyper-parameters in problem 3 significantly impacts the overall performance. However, determining optimal values for λ and θ poses a considerable challenge. In realistic datasets, the true coefficient vector β is unknown, making it difficult to assess the final results accurately. Consequently, an alternative metric is needed to replace the $\mathcal{F}1$ score for conducting cross-validation.

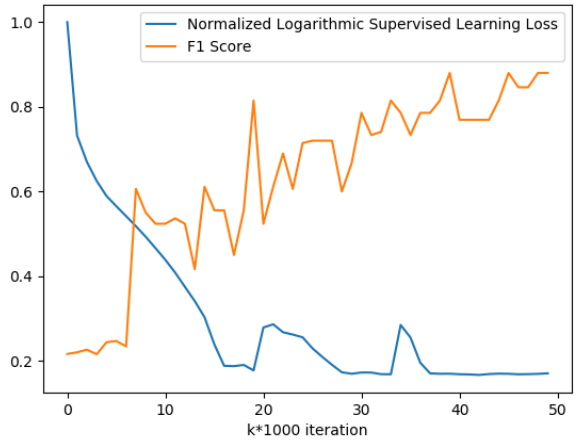


Figure 5: $\mathcal{F}1$ Score Versus Normalized Logarithmic Supervised Learning Loss

One potential indicator is the supervised learning loss. We calculate the $\mathcal{F}1$ score at regular intervals, typically every 1000 iterations, while simultaneously recording the logarithmic supervised learning loss. To facilitate comparison, we normalize the loss by dividing it by the maximum logarithmic supervised learning loss. Figure 5 demonstrates that as the supervised learning loss decreases, the $\mathcal{F}1$ score tends to increase.

Drawing inspiration from this observation, we can utilize cross-validation to identify the optimal values of θ and λ that minimize the converged supervised learning loss for test datasets.

In addition, we have observed that when each term in equation 3, along with its corresponding coefficients, possesses similar magnitudes, the overall performance improves. This phenomenon has also been mentioned in the work by Ouyang (Ouyang & Key, 2021).

The idea behind this approach involves training the model with random values for λ and θ. Once the training loss converges, the values of λ and θ are reset such that the terms $\frac{1}{n}\sum_{i=1}^{n}\sum_{(\mathbf{x},y)\in\mathcal{D}i}\ell\Big(h\boldsymbol{\theta}\big(\boldsymbol{\Phi}_i(\mathbf{x})\big),\boldsymbol{\Phi}i(y)\Big)$ (supervised training loss), $\mu\sum_{i=1}^{m}\text{MMD}\Big(\Phi_i(\mathcal{D}_i),\mathcal{N}(0,\Sigma)\Big)$ (MMD loss), and $\lambda\|\boldsymbol{\theta}\|_1$ ($\mathcal{L}1$ norm) are approximately equal. For instance, if the supervised training loss is 1000, the MMD loss is 0.1, and the L_1 norm is 10, we can set $\mu = 10^4$ and $\theta = 100$. This process is repeated until no further updates are needed for λ and θ. This method assists in determining suitable hyper-parameters and adjusting step sizes accordingly.

Subsequently, cross-validation is employed to search for the values of θ and λ that minimize the supervised learning loss on the test dataset. The empirical settings for λ and θ can also help narrow down the search range during cross-validation implementation.