Ensuring Superiority of Stable Regression in the Presence of Covariate Shift

Anna Midgley
annamidg@mit.edu

Nora Hallqvist
norahall@mit.edu

December 2023

Abstract

We investigate the performance of Stable Regression [1] compared to the random assignment of data to training and validation sets in the presence of a covariate shift. In particular, our investigation reveals that without the integration of importance sampling, Stable Regression is not guaranteed to outperform randomization under a covariate shift. However, when importance sampling is applied to both methods, Stable Regression regains it superiority, yielding more accurate and stable results. The code is available at `https://github.com/mogdog18/CovariateStableReg`.

1 Introduction

Stable Regression, where training and validation sets are optimally chosen, has been proven to perform significantly better than randomly constructed counterparts, in terms of predictions error and the standard deviations of resulting predictions [1]. Additionally, stable regression enhances model stability, where there is lower variability in coefficients, predictions and support [1].

Distribution shifts between training and testing are ubiquitous to real-world machine learning (ML) systems. For example, the patients studied in clinical trials may not come from the target population due to sample bias [3]. This facilities the need for ML model to be robust to covariate shifts between the source and target domain, to ensure that the deployed model can predict accurately in the deployed setting. We expect, that robust optimization's performance, like other ML methods, will suffer under covariate shift. Stable regression optimally selects the hardest datapoints to train the model on. If however, these selected datapoints are not representative of the test data, then the deployed model will have weak performance in the target domain. In this paper, we explore the novel approach of combining stable regression optimization with importance sampling to account for covariate shift, aiming to ensure the superiority of stable regression over randomization.

1.1 Objectives

The primary goals are as follows:

- Examine whether stable regression maintains superior performance compared to randomization when confronted with covariate shift between the training and testing datasets, without implementing importance sampling.
- Explore whether stable regression outperforms randomization when dealing with a covariate shift between the training and testing datasets, incorporating the use of importance sampling.

2 Problem Formulation

Throughout this report, we will refer to the full dataset that is used for training, which includes validation data, as training dataset$^{(1)}$. The data that is used only for model training is called the training dataset$^{(2)}$. These superscripts are used for x, y and n too.

2.1 Problem setup

In the context of covariate shift, our scenario involves a labeled training dataset $\{x_j^{\text{train},1}, y_j^{\text{train, }1}\}_{j=1}^{n_{\text{train},1}}$ consisting of $n_{\text{train},1}$ samples, and an unlabeled test dataset $\{x_j^{\text{test}}\}_{j=1}^{n_{\text{test}}}$ with n_{test} samples. The true labels for the test dataset, denoted as $\{y_j^{\text{test}}\}_{j=1}^{n_{\text{test}}}$, are assumed to be unavailable. Importantly, we assume the presence of a covariate shift between the training and test datasets, substantiated by the following assumptions:

- **Assumption 1:** The marginal distributions of the features in the training and test datasets, $\Pr_{\text{train},1}(\mathbf{x})$ and $\Pr_{\text{test}}(\mathbf{x})$, are not equal i.e $\Pr_{\text{train},1}(\mathbf{x}) \neq \Pr_{\text{test}}(\mathbf{x})$.
- **Assumption 2:** The conditional distribution of the response variable given the features, $P_{\text{train},1}(y|\mathbf{x})$ and $\Pr_{\text{test}}(y|\mathbf{x})$, remains invariant across the training and test datasets i.e $\Pr_{\text{train},1}(y|\mathbf{x}) = \Pr_{\text{test}}(y|\mathbf{x})$.

These assumptions collectively define the covariate shift scenario, indicating that while the distributions of the input features differ between training and test datasets, the distribution of the response variable conditioned on the features remains consistent.

2.2 Importance sampling

Before delving into the explanation of two methods to address regression under covariate shift, we'll first elucidate how importance sampling is employed to account for the shift. In a standard supervised problem without covariate shift, the parameters β are determined by minimizing the empirical risk:

$$\mathbb{E}_{(X,Y)\sim \Pr^{\text{train},1}}[L(X,Y;\beta)] \approx \frac{1}{n_{\text{train},1}} \sum_{i=1}^{n_{\text{train},1}} L(x_i, y_i; \beta) \tag{1}$$

However, under a covariate shift, the empirical train error, as defined in Equation 1, is an inaccurate estimate for the test error. Instead, the expected test error is approximated using the weighted empirical errors of the training data [3]:

$$\begin{aligned}\mathbb{E}_{(X,Y)\sim \Pr^{\text{test}}}[L(X,Y;\beta)] &= \int \frac{\Pr^{\text{test}}(x,y)}{\Pr^{\text{train},1}(x,y)} L(x,y;\beta) \Pr^{\text{train},1}(x,y)dxdy \\ &= \mathbb{E}_{(X,Y)\sim \Pr^{\text{train},1}}\left[\frac{\Pr^{\text{test}}(X,Y)}{\Pr^{\text{train},1}(X,Y)} L(X,Y;\beta)\right]\end{aligned}$$

Therefore, given $w(x,y) = \frac{P^{\text{test}}(x,y)}{P^{\text{train},1}(x,y)}$, we can compute the risk with respect to P^{test} using $P^{\text{train},1}$. Notably, the weights $w(x,y)$ are typically unknown and need to be approximated from the data. Utilizing Assumption 2, we find:

$$w(x,y) = \frac{\Pr^{\text{test}}(y|x)\Pr^{\text{test}}(x)}{\Pr^{\text{train},1}(y|x)\Pr^{\text{train},1}(x)} = \frac{\Pr^{\text{test}}(x)}{\Pr^{\text{train},1}(x)}$$

The specific approach for estimating $w(x)$ is detailed in subsection 3.2.

2.3 Traditional Randomization Approach with Importance Sampling

In the traditional approach to address covariate shift in linear regression, given a training dataset $\{x_j^{\text{train},1}, y_j^{\text{train},1}\}_{j=1}^{n_{\text{train},1}}$, the task involves randomly partitioning the points $1, \cdots, n^{\text{train},1}$ into subsets $I_{\text{train},2}$ and I_{val} such that $I_{\text{train},2} \cup I_{\text{val}} = \{1, ..., n_{\text{train},1}\}$ and $I_{\text{train},2} \cap I_{\text{val}} = \varnothing$. The newly partitioned training set is then utilized to minimize the loss function under importance sampling:

$$\frac{1}{n}\sum_{i=1}^{n_{\text{train},2}} w(x_i)|y_i - x_i^T\beta| + \lambda\sum_{i}^{p}\Gamma(\beta_i)$$
$$\text{s.t} \quad w(x_i) = \frac{\Pr^{\text{test}}(x_i)}{\Pr^{\text{train}}(x_i)}, \quad \forall i \in [n_{\text{train},2}]$$

While one could opt for the squared l_2 loss function, in accordance with [1], we proceed with the l_1 loss. Additionally, we employ the l_1 norm as the regularization function.

2.4 Robust Optimization Approach With Importance Sampling

We now propose an approach that combines stable regression [1] with importance sampling. Instead of the conventional method of randomly partitioning the training data into a training and validation set and subsequently re-weighting the loss function, we integrate importance sampling directly into the optimization problem which solves for both the split and the optimal β:

$$\min_{\beta}\max_{z\in\mathbb{Z}} \sum_{i=1}^{n_{\text{train},2}} z_i w(x_i)|y_i - x_i^T\beta| + \lambda\sum_{i=1}^{p}\Gamma(\beta_i)$$
$$\text{s.t } Z = \left\{z : \sum_{i=1}^{n} z_i = K, z_i \in \{0,1\}\right\}$$
$$w(x_i) = \frac{\Pr^{\text{test}}(x_i)}{\Pr^{\text{train}}(x_i)}, \quad \forall i \in [n_{\text{train},2}]$$

Here, K is a hyperparameter determined by the train and validation split proportion. As mentioned previously, we utilize the l_1 norm as the regularization function. Following the methodology outlined in [1], we take the dual of the inner maximization problem to alleviate the multiplication of variables:

$$\begin{aligned}
\min_{\beta,\theta,u_i} \; & K\theta + \sum_{i=1}^{n_{\text{train},2}} u_i + \lambda\sum_{i=1}^{p}\Gamma(\beta_i) \\
\text{s.t} \; & \theta + u_i \geq w(x_i)(y_i - x_i^T\beta), \quad \forall i \in [n_{\text{train},2}] \\
& \theta + u_i \geq -w(x_i)(y_i - x_i^T\beta), \quad \forall i \in [n_{\text{train},2}] \\
& u_i \geq 0, \quad \forall i \in [n_{\text{train},2}] \\
& w(x_i) = \frac{\Pr^{\text{test}}(x_i)}{\Pr^{\text{train}}(x_i)}, \quad \forall i \in [n_{\text{train},2}]
\end{aligned} \tag{2}$$

3 Methodology

3.1 Data

In order to analyze the performance of the methods, we consider both synthetic and real-world datasets.

3.1.1 Synthetic datasets

We construct two synthetic datasets, where the ground truth and characteristics of the problem are known. This allows for straightforward evaluation of the proposed methods for benchmarking, and to validate that the algorithm is working as expected before introducing the complexities of real-world data.

1. **Synthetic dataset 1:**

The first dataset is generated accorded to a polynomial regression example [6], where

$$f(x) = -x + x^3 + \epsilon, \quad \epsilon \sim N(0, 0.3^2)$$

training and testing datapoints are sampled from two normal distributions, respectively $\Pr \sim N(0.5, 0.5^2)$ and $\Pr \sim N(0, 0.3^2)$.

2. **Synthetic dataset 2:**

The second dataset is generated according to the following function [5]:

$$f(x) = \text{sinc}(x) + \epsilon, \quad \epsilon \sim N(0, \frac{1}{4}^2)$$

training and testing datapoints are sampled from two normal distributions, respectively $\Pr \sim N(1, 0.5^2)$ and $\Pr \sim N(2, \frac{1}{4}^2)$.

3.1.2 Real-world datasets

We use three datasets from the UCI Machine Learning Repository, namely, Abalone, Computer Hardware and Concrete [2]. To create a covariate shift, we applied biased sampling strategies using the first principal component obtained through Principal Component Analysis (PCA). The steps used are outlined in Figure 1 [4]:

Input: Entire dataset X
Output: Training set $S_{\text{train},1}$, Test set S_{test}

pca $\leftarrow$ PCA(X) ▷ Perform Principal Component Analysis on X
for $i = 1$ to n **do**
  if pca$[i, 1] \geq$ median **then** ▷ Is sample point i of first principal component above the median
    if $j \sim \mathcal{N}(0, 1) \leq p_1$ **then** ▷ Assign to training set with probability p_1
      Add (X_i, y_i) to $S_{\text{train},1}$
    else
      Add (X_i, y_i) to S_{test}
    end if
  else
    if $j \sim \mathcal{N}(0, 1) \leq p_2$ **then** ▷ Assign to training set with probability p_2
      Add (X_i, y_i) to S_{test}
    else
      Add (X_i, y_i) to $S_{\text{train},1}$
    end if
  end if
end for

Figure 1: Pseudo-code to create covariate shift in real-world datasets using biased sampling

The hyper-parameters p_1 and p_2 are set to 0.1, and 0.9 for all the datasets. It is noted that this method does not create a 90/10 split between train$^{(1)}$ and test. To impose this, we randomly select datapoints from the outputted datasets, such that the final train$^{(1)}$ and test datasets have the desired ratio of points.

3.2 Approach

We explore 4 variations of fitting a model, specifically,

1. Training$^{(2)}$ and validation datasets constructed randomly, and fitting a Linear Lasso Regression model

2. Training$^{(2)}$ and validation datasets constructed randomly, and fitting a Linear Lasso Regression model that uses covariate weights

3. Training$^{(2)}$ and validation datasets constructed and fitting a normal Linear Lasso Regression model, using Stable Regression.

4. Training$^{(2)}$ and validation datasets constructed and fitting a normal Linear Lasso Regression model, using Stable Regression with covariate weights

Across all 4 variations of the problem, we apply this overarching procedure, and do so 1000 times: For a given dataset, we create a testing set by randomly selecting 10% of the data, ensuring that this split introduces a covariate shift within the test set. The remaining 90% of the data, free from covariate shift, is then partitioned into training$^{(2)}$ and validation sets. For the randomized procedure, we randomly select 30% of the (training$^{(1)}$) data for validation, and use the remaining as training$^{(2)}$ data. We find the β value for a sequence of λ values. We then select the optimal β and λ pair, that had the smallest error on the validation set. For the optimization procedure, this splitting was done via solving Equation 2. Specifically, given the β^* from the solving Equation 2 we recover the training$^{(2)}$ and validation split by calculating the residuals on all of the data-points using β^* and then picking the 70% data-points with the largest residuals. The optimal coefficients β are then learned from these training/validation splits over a sequence of λ values. Similarly to randomization, the pair of λ and β that had the smallest error on the validation set was selected. For both methods, the optimal coefficients were then applied to the test set, and the mean squared error (MSE) computed. This procedure was repeated, each time selecting a different test set. We aggregate the results across the iterations taking the mean MSE, standard deviation of the MSE, and the standard deviation of the coefficients.

For variation 2 and 4, we use importance sampling weights to correct for covariate shift. Specifically, we use importance sampling weights for both the model fitted using the randomly constructed data splits, and for Stable Regression as outlined 2. To obtain the weights used in importance sampling, we train a logistic regression model to distinguish between the train$^{(1)}$ and test sets, where the test set contains the covariate shift. The process involved uses the following steps:

1. Concatenate the train$^{(1)}$ and test data, where the response variable is set as 0 for the train set and 1 for the test set.

2. Train a classifier to differentiate between the train and test sets.

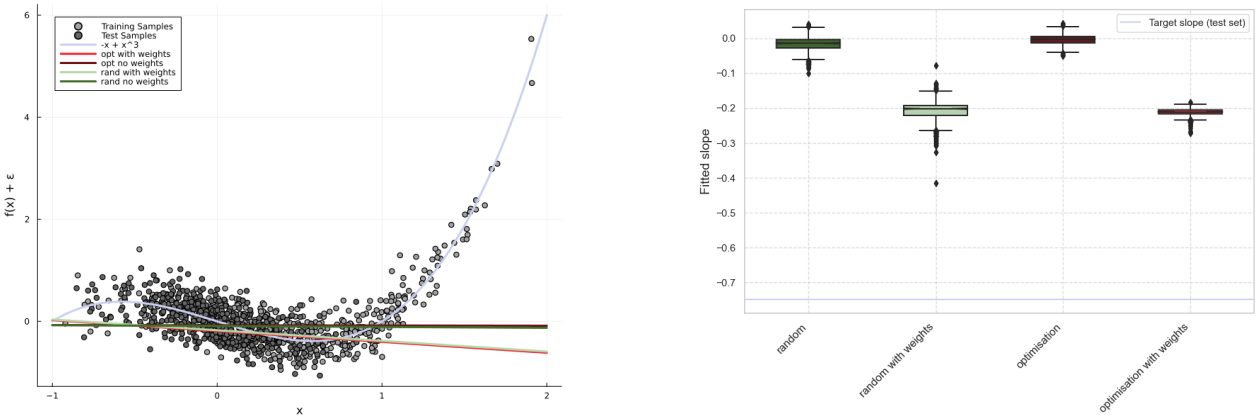
3. Calculate the density ratio $\frac{\Pr(Y=1|X)}{\Pr(Y=0|X)}$ for the train set.

4. Assign weights to the train set using the density ratio, i.e., $w_i = \frac{\Pr(Y=1|x_i)}{\Pr(Y=0|x_i)}$ for $i \in [n_{\text{train},1}]$.

4 Results and Discussion

4.1 Synthetic Data

The performance of the four distinct methods on the synthetic data generated from the function $f(x) = x^3 - x$ is depicted in Figure 2. Of particular interest is the 9.952% improvement in MSE observed when employing the Stable Regression approach compared to randomization with importance sampling. The rationale behind this improvement lies in that Stable Regression selects the most challenging data points, which are also representative of the test set (assisted by weight sampling), thereby enhancing the robustness of the trained regression model compared to the randomization approach. To mitigate the concern that the observed improvement is solely attributable to the Stable Regression approach, we report both the MSE improvements for randomization and Stable Regression independently in Table 1. Notably, both improvements are positive, affirming that both methods benefit from the incorporation of sampling weights. This improvement is further underscored by the slopes depicted in Figure 2a, where weight sampling contributes to a superior fit of the regression model to the distribution of the test set, resulting in a more pronounced negative slope. In Figure 2b, we see that β^* for optimization with weights is not only more stable (shown by the narrower box),

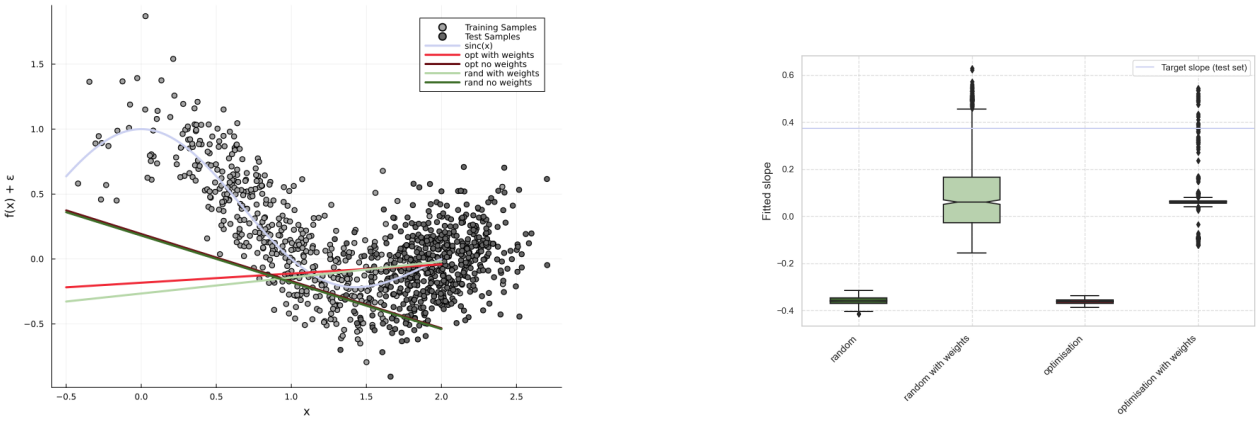


(a) (b)

Figure 2: Comparison of the results for synthetic data with $f(x) = x^3 - x$

but has a median value closer to the true slope of the target domain. The improvement is minimal, and it is noted that the plot only shows the contribution from the slope and not the fitted intercept value.

Next, we analyze the performance of the methods for the synthetic data generated for $f(x) = \text{sinc}(x)$, which is shown in Figure 3. Notably, for this dataset, we observe a negative improvement of -2.2% in the Stable Regression compared to the randomization method when no weights are applied. This result can be attributed to the large discrepancy between the train and test distributions compared to the previous synthetic dataset, as seen comparing Figure 2 and Figure 3. This potentially is causing Stable Regression to select the most challenging instances from the training set which are not representative of the test data. However, similar to the synthetic data generated for $f(x) = x^3 - x$, when importance sampling is used the Stable Regression performs better than randomization. Similarly to the first dataset, we see in Figure 2b that β^* for optimization with weights is not only more stable (shown by the narrower box), but has a median value closer to the true slope of the target domain. There are more outliers in the β^* values for Stable Regression than the first dataset. One possible explanation for this is that the source and target domains in this problem are fairly separate. This impacts the importance sampling weights that are found by the classifier, and can lead to instability when you are dividing by a small probability.



(a) (b)

Figure 3: Comparison of the results for synthetic data with $f(x) = \text{sinc}(x)$

4.2 Real-world Data

The percentage improvements on the prediction error (MSE) for the two real-world datasets are shown in Table 1. Notably, similar to the observation made for synthetic dataset 2, 1 shows that Stable regression is not guaranteed to outperform randomization without weights. Specifically for the computer hardware dataset a -17.70% improvement in MSE is observed. In addition, aligned with synthetic datasets, the real world datasets reveals that Stable Regression consistently outperforms randomization with importance sampling, as evidenced by the positive MSE improvements. Ultimately, this is a novel and exciting finding, showing that stable regression can be enhanced by working importance sampling weights into the method when there is a distribution shift between the source and target domains.

Method / Dataset	Synthetic 1	Synthetic 2	Abalone	Concrete	Computer Hardware
SR vs. Random. with no weights	0.514	-2.185	2.369	2.474	-17.70
SR. vs. Random. with weights	9.952	1.740	0.377	9.890	3.10
Random. with and without weights	79.903	28.563	4.339	46.655	80.29
SR. with and without weights	82.763	31.393	2.440	53.417	85.62

Table 1: MSE Improvement (%) for Different Methods, where SR and Random. are short for Stable Regression and Randomization respectively

5 Conclusion

In this study, we explored the performance of Stable Regression in the context of covariate shift, uncovering a notable increase in prediction errors. Specifically, Stable Regression is not guaranteed to outperform randomization in the presence of a covariate shift. Subsequently, we introduced an innovative framework designed to mitigate this decline in performance. Our investigation revealed that the integration of Stable Regression with importance sampling weights consistently led to a reduction in prediction errors across all four datasets examined. Looking ahead, our future research aims to extend these findings by evaluating the performance of our proposed method on additional real-world datasets, with a preference for those exhibiting inherent covariate shifts.

6 Member Contributions

Throughout this project, we worked closely alongside one another. We split the work as we navigated through the sub-tasks of the project, and did not split large components. Thus it is hard in hindsight to describe the specific contribution of the team members, as it really was a team effort.

References

[1] Dimitris Bertsimas and Ivan Paskov. Stable regression: On the power of optimization over randomization. *Journal of Machine Learning Research*, 21(230):1–25, 2020. URL: http://jmlr.org/papers/v21/19-408.html.

[2] Dheeru Dua and Efi Taniskidou. Uci machine learning repository. 2017. URL: http://archive.ics.uci.edu/ml.

[3] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper_files/paper/2006/file/a2186aa7c086b46ad4e8bf81e2a3a19b-Paper.pdf.

[4] Anqi Liu and Brian D. Ziebart. Robust covariate shift prediction with general losses and feature views, 2017. arXiv:1712.10043.

[5] Klaus-Robert Muller Masashi Sugiyama, Matthias Krauledat. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research 8*, 90:985–1005, 7 2007. URL: https://jmlr.org/papers/volume8/sugiyama07a/sugiyama07a.pdf.

[6] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, 10 2000. doi:10.1016/S0378-3758(00)00115-4.