Applying Histogram Matching as a Data Transformation for ML

Spitzer, Nadav
nspitzern@gmail.com

Cohen, Shimon
shimonchoen655@gmail.com

February 2022

Abstract

There are many ways and examples to Transform Data - from a simple function like log *to a more complex one such as* coxbox. *We would like to introduce a new method for Data-Transformation, with usage of* Histogram-Matching *from the* Image-Process *world. The method tries to match a histogram of a given source distribution to another histogram, representing a target distribution, based on the properties of the two.*

1 Introduction

The assumption of normality suggests that every data used for statistical test or regression, should fit roughly to a bell curve. Many methods were suggested in order to achieve this property: applying a *log* function, raise by an *exponent* etc. Each one of these methods preforms very well on particular data, while performs sufficiently, sometimes even bad, on other type of data. For example, a *log* function performs very well on data that is skewed to the left, while performing bad on data that is highly skewed to the right.

2 Solution overview

2.1 Histogram-Matching

The *Histogram-Matching* algorithm is defined as follows: Given two distributions, the source and the target (denote as D_{source}, D_{target}), we calculate their histograms (H_{source}, H_{target}). Following, we calculate the *cumulative distribution functions* (*CDF*) of the two histograms - $F_s()$ for the source distribution and $F_t()$ for the target. For every value $v_s \in D_{source}$, we find the interpolated value so that there are two values $v_{t1}, v_{t2} \in D_{target}$ such that:

$$F_t(v_{t1}) \leq F_s(v_s) \leq F_t(v_{t2})$$

This mapping, $M()$ is the result of the *Histogram-Matching*. Finally, we apply this mapping for every value in D_s.

2.2 Our Idea

Our idea was to match a given distribution to a normal distribution with the *Histogram Matching* algorithm.

Algorithm 1: Normal Matching Algorithm

Given distribution D_s:

$\mu_s = \frac{1}{|D_s|} \sum_{d \in D_s} d$

$\sigma_s = \sqrt{\frac{1}{|D_s|} \sum_{d \in D_s} (d - \mu_s)^2}$

$D_t \sim \mathcal{N}(\mu_s, \sigma_s)$

$D_{matched} \leftarrow HistogramMatching(D_s, D_t)$

We have also introduced a normalization of the matched distribution to the values of the original distribution in order to maintain the same value range.

2.3 Research points

2.3.1 Transform different distributions

In order to test the correctness of our algorithm, we tested its results on different distributions such as *uniform, poisson*, etc. We measured the results by calculating the $pValue$ given by the *KS-test* and *Shapiro-Test*, between the transformed distribution and a normal sample.

2.3.2 Histogram matching on selected data sets

After testing the correctness of the algorithm, we applied the transformations on different data sets in order to see the effect it will have on the model's learning process.

2.4 Dataset types

In our research we used two types of data sets, Regression and Classification, three data sets each. Each data set containing data from different fields such as Housing prices and the Possibility of rain in Australia.

2.5 Insights from data

Various parameters of the features may give us some insights on the data. Our attempt to discover such information used the $mean, std, entropy, skewness$ etc., didn't provide anything new about the features. Features that made the scores betters could have had the same values as features that made the scores worse.

2.6 Dataset manipulation

Our method, by nature, applies to numerical features and not to categorical ones. The obvious state is that numerical features are sampled from different distributions, while categorical ones tend to have only several *bins* such that each category falls exactly in one of them. Furthermore, categorical features don't create a varied histogram that can be interpreted as a distribution, like numerical features usually do. To combat with this issue, we split each dataset into *numerical* and *categorical* features, and applied our algorithm only on the numerical ones. In order to have a well defined split, we decided that a feature is numerical by checking how many unique values are in that feature, and keep only those that are above a given threshold (usually 5%). All the other features are considered *categorical*. Another issue was receiving samples that were missing their target value (N/A). Those elements were removed in the case of a classification task, and replaced with the mean of the train-set.

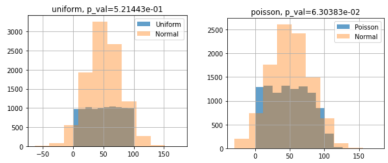


Figure 1: Distributions are matched to a normal distribution based on the same mean and std. In figure (a) match from *Uniform* distribution and in figure (b) match from *Poisson* distribution. The p_val indicates the *KS test* of the matched distribution to a normal distribution with the same mean and std.

2.7 Distribution manipulation

Another method we tested was to scale the matched distribution back to the same range as the original distribution, with a standard normalization formula (See appendix A). This method, as far as we could tell, didn't make any difference in the performance of the models.

3 Results

3.1 Compare Algorithm results to other Normalization methods

The results of *Histogram-Matching* algorithm were measured with 3 methods:

1. Kolmogorov–Smirnov test - tests the fitness of one distribution to another (in this case - the matched distribution to a random normal distribution with the same mean and std).
2. Shapiro-Wilk test - tests the normality of a distribution.
3. QQ-plots.

For all measuring methods the results indicated a very good match to normal distribution. Distributions that were extremely far from being normal were matched to a very plausible normal distribution. The *p value* of both the *KS Test* and the *Shapiro Test* (the possibility of matching to

the tested distribution) was improved in many scales (See Fig. 2).

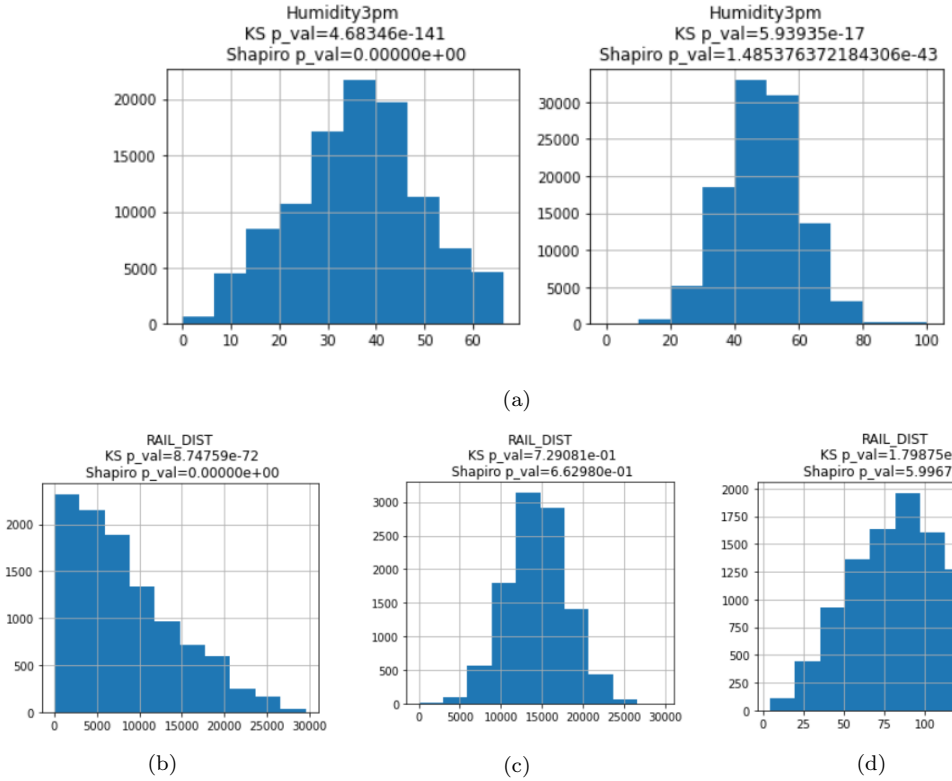
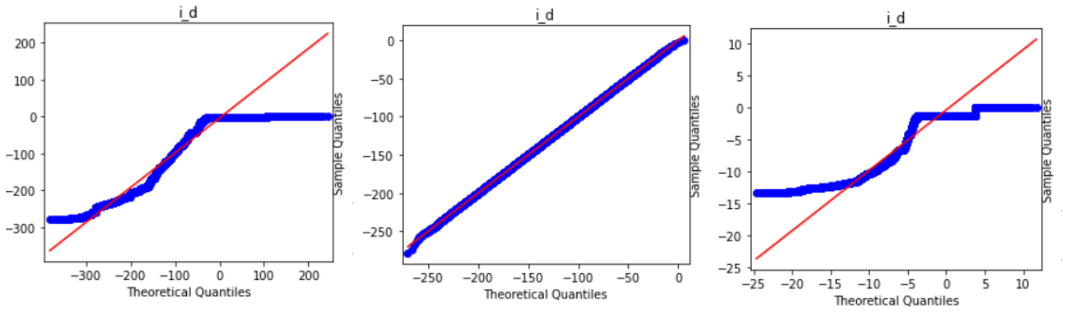


Figure 2: In figure (a), the left histogram is the original distribution and the one the right is the matched distribution. The *p value* above each method indicates the probability of matching to a normal distribution. *Histogram-Matching* improves the probability of null-hypothesis by a very large factor. While *Yeo-Johnson* method normalizes the data as well, its match to normal distribution is not as good. (b) indicates the original distribution, (c) the distribution after *Histogram-Matching* and (d) the distribution after *Yeo-Johnson* transform.

QQ-Plots measure the quantile fit of experimental distribution to a theoretical one. In this case each normalization method was measured as compare to a theoretical Normal Distribution. The *Histogram-Matching* method showed a great fitness results to that distribution in comparison to the original distribution and Yeo-Johnson manipulated distribution (See Fig. 3).

3.2 Compare model performance

After comparing the results of matching the features to Normal-Distribution, our goal was to test the performance of a model on the matched-dataset against other methods. We tested the results of *Linear Regression* and *Logistic Regression* on the tasks of Regression and Classification respectively.



(a) (b) (c)

Figure 3: Comparing the QQ-Plots of one column - Original distribution (a), Matched distribution (b) and after Yeo-Johnson transform (c). The *Matched* distribution has the best fit to the theoritical (=normal) distribution.

We trained 3 models for each method: one for the original dataset (without any transformation), another for the matched dataset (with our *Histogram-Matching Algorithm*) and the last for a transformed dataset with the *Yeo-Johnson* method.

3.2.1 Normalizing all features

Surprisingly our results changed from dataset to dataset. In general, the matching-method performed worse than the other methods (See Table 1).

Dataset	Original	Matched (ours)	Yeo-Johnson
Motor-Measures	0.8157	-3.423	0.627
Miami-Houses	0.696	0.495	0.4934
Crab-Age	0.54	-0.1	0.556
Australia-Weather	77.53%	99.99%	77.53%
Mouse-Classes	100%	100%	100%
Mobile-Price	64.33%	30.33%	67.66%

Table 1: The first 3 datasets are the Regression task and were measured with R^2. The other 3 are Classification task and are measured with *Accuracy*. The *Matched* method, performed very well on the Australia-Weather dataset (predicting if it would rain on the current/next day or not), but in overall it performed worse than the other methods.

3.2.2 Normalizing one feature

Following the previous results, measuring the effect of *Normalizing* only one feature at a time was suggested. The result varied. Some features made the *Matched* model get the same results as the *Regular* model, some made it behave very poorly, and some got very good results (See Table 2). It

was very difficult to determine which feature has caused each of these effects, and unfortunately we still don't have a specific way to predict it.

Best Results - 1 Column Norm.

Dataset	Original	Matched (ours)	Yeo-Johnson
Motor-Measures	0.815	0.826	0.822
Miami-Houses	0.696	0.6941	0.705
Crab-Age	0.54	0.542	0.5477
Australia-Weather	77.53%	77.53%	77.53%
Mouse-Classes	100%	100%	100%
Mobile-Price	64.33%	74%	73.33%

Worst Results - 1 Column Norm.

Dataset	Original	Matched (ours)	Yeo-Johnson
Motor-Measures	0.815	-3.67	0.6018
Miami-Houses	0.696	0.596	0.572
Crab-Age	0.54	-1.28	0.505
Australia-Weather	77.53%	77.53%	77.53%
Mouse-Classes	100%	100%	100%
Mobile-Price	64.33%	33%	61%

Table 2: These tables show the best and worst score each model got when normalizing just one column. For some columns the *Matched* model got even better results then both the *regular* and *Yeo-Johnson* models, but for some it made it much worse.

3.2.3 Distribution comparison

After getting the results, we wanted to check if the data distribution had a substantial affect on the accuracy of the model.

On one Data-set, we could see that when the original data was skewed, the model results for those features was worse than the results on the original data and yeo transform. On the other hand, this behaviour didn't repeat on other Data-sets. This may be because of lack of relevant data.

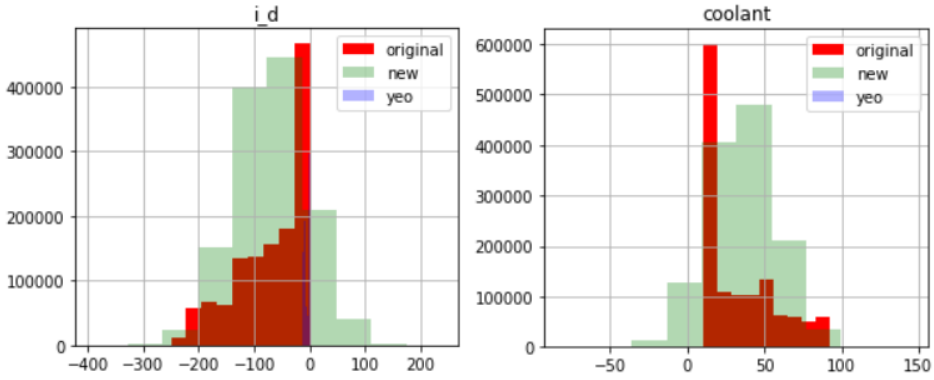
Bad results on model for skewed features

Feature	Matched (ours)	Yeo-Johnson
coolant	-3.675	0.601
i_d	0.308	0.821

Good results on model for skewed features

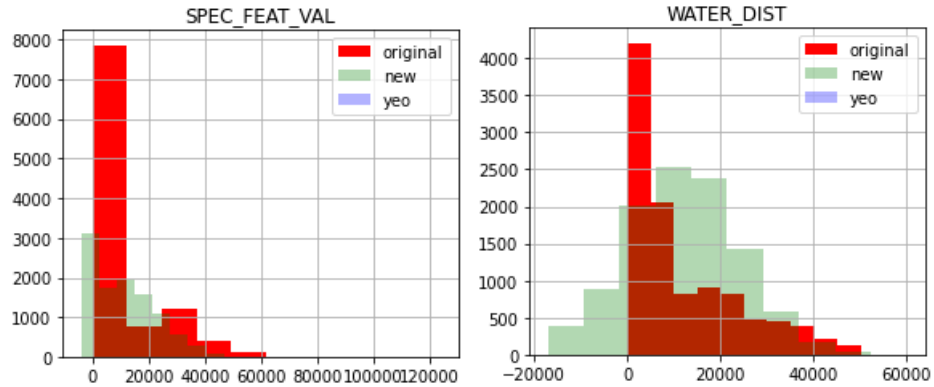
Feature	Matched (ours)	Yeo-Johnson
SPEC_FEAT_VAL	0.6865	0.691
WATER_DIST	0.688	0.701

Table 3: Results of the model on skewed data after transformation



(a) (b)

Figure 4: Visualization of skewed features on which the model yielded bad results



(a) (b)

Figure 5: Visualization of skewed features on which the model yielded good results

As we can see, on a different Data-set, we have some features with skewed data which yield good results. Note that in this case the Data-set had few features.

3.3 Feature correlation

Typically, there is some level of correlations between the features of datasets. Our next guess was to see if any type of correlation could imply the results of the models. It seemed, at a first glance, that features with high correlation to the target features may cause drastic shifts in the results, but it was true only for some datasets and not to all. There were features with the same level of correlation that caused opposite results (for better and worse).

4 Related Work

4.1 Other data transformation techniques

There are many Data-Transformation techniques, all of them try to solve 2 main problems:

1. Match the data to a bell-curve (normal distribution).
2. Prevent skewness of the data.

For example, applying a *Log* function on data that is highly skewed to the left, will cause a spread of the distribution. It does not compel a match to normal distribution, but lowers the skew factor of it. On the other hand, if the data is skewed to the right, this will only make it worse. For this case we will use the opposite function - a *Power (Exponential)* function.
These transformations are quite straight-forward, and there are more complex ones. *Box-Cox* tries to compel a match to normal distribution of the data by performing *Power* and *Log* transformations on the data. This method can only deal with positive values and so *Yeo-Johnson* made a method that handles negative values too.
The solution we suggested differs in the manner that it doesn't try to match the values of the distribution but its histogram. The histogram of a given distribution is matched to the histogram of a normal distribution - creating a mapping function from one distribution to the other.

5 Conclusions

When testing our algorithm's output on models, normalizing one feature at a time yields better results than normalizing all of the numerical features. In which case, we could clearly identify the features which heavily affect the model's accuracy for the worse, but it is still not clear to us as to why this happens. We could not pin-point the differences. One assumption is that skewed data highly affects our algorithm, but this statement needs to be further tested to be validated.

Appendices

A Min-Max Normalization

The normalization is given as:

$$D^{norm}_{matched} = (D_{matched} - min(D_{matched}))\frac{max(D_{matched}) - min(D_{matched})}{max(D_s) - min(D_s)} + min(D_s)$$