STANDARDIZATION IN MACHINE LEARNING

Article · March 2021

CITATIONS
5

READS
2,976

1 author:

Sachin Vinay
Delhi Technological University
2 PUBLICATIONS **5** CITATIONS

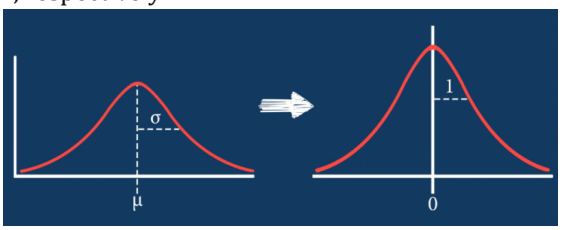
SEE PROFILE

STANDARDIZATION

1. **What does Feature Scaling mean?**
   In practice, we often encounter different types of variables in the same dataset. A significant issue is that the range of the variables may differ a lot. Using the original scale may put more weights on the variables with a large range. In order to deal with this problem, we need to apply the technique of features rescaling to independent variables or features of data in the step of data pre-processing. The terms normalisation and standardisation are sometimes used interchangeably, but they usually refer to different things.
   The goal of applying Feature Scaling is to make sure features are on almost the same scale so that each feature is equally important and make it easier to process by most ML algorithms.

2. What is standardization?

The result of **standardization** (or **Z-score normalization**) is that the features will be rescaled to ensure the mean and the standard deviation to be 0 and 1, respectively.



$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

Ans.

- The concept of standardization comes into picture when continuous independent variables are measured at different scales.
- This technique is to re-scale features value with the distribution value between 0 and 1 is useful for the optimization algorithms, such as gradient descent, that are used within machine learning algorithms that weight inputs (e.g., regression and neural networks). Rescaling is also used for algorithms that use distance measurements, for example, K-Nearest-Neighbours (KNN).
- It means these variables do not give equal contribution to the analysis. For example, we are performing customer segmentation analysis in which we are trying to group customers based on their homogenous (similar) attributes.
- A variable called 'transaction amount' that ranges between $100 and $10000 carries more weightage as compared to a variable i.e. number of transactions that in general ranges between 0 and 30. Hence, it is required to transform the data to comparable scales. **The idea is to rescale an original variable to have equal range and/or variance.**

3. **Why Should We Use Feature Scaling?**

Some machine learning algorithms are sensitive to feature scaling while others are virtually invariant to it.

Gradient Descent Based Algorithms

Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled. Take a look at the formula for gradient descent below:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

The presence of feature value X in the formula will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature. To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

Having features on a similar scale can help the gradient descent converge more quickly towards the minima.

Distance-Based Algorithms

Distance algorithms like KNN, K-means, and SVM are most affected by the range of features. This is because behind the scenes **they are using distances between data points to determine their similarity.**

For example, let's say we have data containing high school CGPA scores of students (ranging from 0 to 5) and their future incomes (in thousands Rupees):

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40
3	4	4.5	50
4	5	4.2	52

Since both the features have different scales, there is a chance that higher weightage is given to features with higher magnitude. This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biased towards one feature.

Therefore, we scale our data before employing a distance based algorithm so that all the features contribute equally to the result.

	Student	CGPA	Salary '000
0	1	-1.184341	1.520013
1	2	-1.184341	-1.100699
2	3	0.416120	-1.100699
3	4	1.216350	0.209657
4	5	0.736212	0.471728

Scaling has brought both the features into the picture and the distances are now more comparable than they were before we applied scaling.

Tree-Based Algorithms

Tree-based algorithms, on the other hand, are fairly insensitive to the scale of the features. Think about it, a decision tree is only splitting a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. This split on a feature is not influenced by other features.

So, there is virtually no effect of the remaining features on the split. This is what makes them invariant to the scale of the features!

4. What are the Methods of Standardizations?

Standardisation (Z-score Normalization)	Max-Min Normalization
$x_{stand} = \frac{x - \text{mean}(x)}{\text{standard deviation}(x)}$	$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$

1. Z score

Z score standardization is one of the most popular method to normalize data. In this case, we rescale an original variable to have a **mean of zero** and **standard deviation of one**.

$$z = \frac{x - mean}{std.dev}$$

Mathematically, scaled variable would be calculated by subtracting mean of the original variable from raw value and then divide it by standard deviation of the original variable.

Code

```
#Import library
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_X = sc_X.fit_transform(df)
#Convert to table format - StandardScaler
sc_X = pd.DataFrame(data=sc_X, columns=["Age",
"Salary","Purchased","Country_France","Country_Germany",
"Country_spain"])
sc_X
```

Interpretation

A value of 1 implies that the value for that case is one standard deviation above the mean, while a value of -1 indicates that a case has a value one standard deviations lower than the mean.

Important Point

The standardized values do not lie in a particular interval. It can be any real number.

2. Min-Max Scaling

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

The equation of Max-Min Normalization.

Code

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(df)
scaled_features = scaler.transform(df)
#Convert to table format - MinMaxScaler
df_MinMax = pd.DataFrame(data=scaled_features,
columns=["Age",
"Salary","Purchased","Country_France","Country_Germany",
"Country_spain"])
```

3. Standard Deviation Method

In this method, we divide each value by the standard deviation. The idea is to have **equal variance**, but different means and ranges.
Formula : x/stdev(x)

X.scaled = data.frame(scale(X, center= FALSE , scale=apply(X, 2, sd, na.rm = TRUE)))

var_x2 = 0.08833861

5. WHEN TO STANDARDIZE DATA AND WHY?

For distance based models, standardization is performed to prevent features with wider ranges from dominating the distance metric. But the reason we standardize data is not the same for all machine learning models, and differs from one model to another.

So before which ML models and methods you have to standardize your data and why?

1- BEFORE PCA:

In Principal Component Analysis, features with high variances/wide ranges, get more weight than those with low variance, and consequently, they end up illegitimately dominating the First Principal Components (Components with maximum variance). I used the word "Illegitimately" here, because the reason these features have high variances compared to the other ones is just because they were measured in different scales.

Standardization can prevent this, by giving same weightage to all features.

2- BEFORE CLUSTERING:

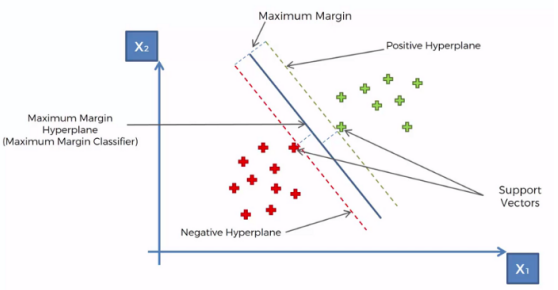
Clustering models are distance based algorithms, in order to measure similarities between observations and form clusters they use a distance metric. So, features with high ranges will have a bigger influence on the clustering. Therefore, standardization is required before building a clustering model.

3- BEFORE KNN:

k-nearest neighbors is a distance based classifier that classifies new observations based on similarity measures (e.g., distance metrics) with labeled observations of the training set. Standardization makes all variables to contribute equally to the similarity measures .

4- BEFORE SVM

Support Vector Machine tries to maximize the distance between the separating plane and the support vectors. If one feature has very large values, it will dominate over other features when calculating the distance. So Standardization gives all features the same influence on the distance metric.



5- BEFORE MEASURING VARIABLE IMPORTANCE IN REGRESSION MODELS

You can measure variable importance in regression analysis, by fitting a regression model using the **standardized** independent variables and comparing the absolute value of their standardized coefficients. But, if the independent variables are not standardized, comparing their coefficients becomes meaningless.

6- BEFORE LASSO AND RIDGE REGRESSION

LASSO and Ridge regressions place a penalty on the magnitude of the coefficients associated to each variable. And the scale of variables will affect how much penalty will be applied on their coefficients. Because coefficients of variables with large variance are small and thus less penalized. Therefore, standardization is required before fitting both regressions.

6. What are the cases when we can't apply standardization

- Logistic Regression
- Tree based algorithms

Tree based algorithms such as Decision Tree, Random forest and gradient boosting, are fairly insensitive to the scale of the features. Think about it, a decision tree is only splitting a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. This split on a feature is not influenced by other features.

7. **What is Normalization?**

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

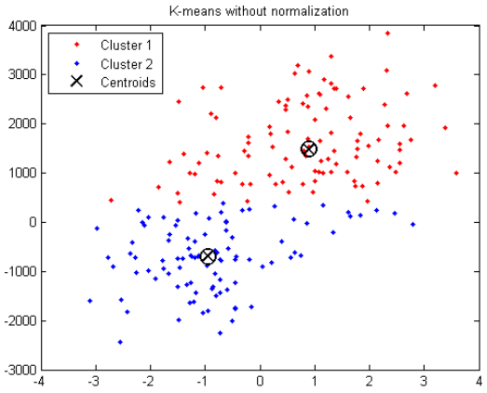
Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

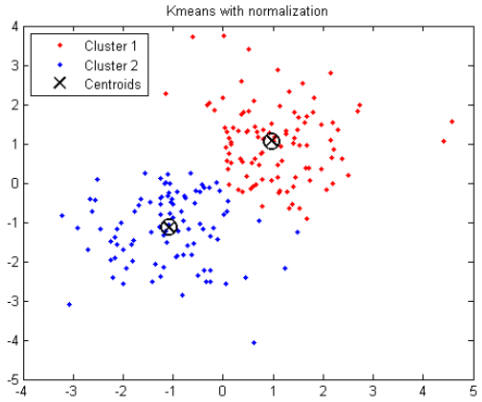
- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

8. **What are the Effects of normalisation on clusters**

Lets take an example

```
X = [randn(100,2)+ones(100,2);...
randn(100,2)-ones(100,2)];

% Introduce denormalization
% X(:, 2) = X(:, 2) * 1000 + 500;

opts = statset('Display','final');

[idx,ctrs] = kmeans(X,2,...
'Distance','city',...
'Replicates',5,...
'Options',opts);

plot(X(idx==1,1),X(idx==1,2),'r.','MarkerSize',12)
hold on
plot(X(idx==2,1),X(idx==2,2),'b.','MarkerSize',12)
plot(ctrs(:,1),ctrs(:,2),'kx',...
'MarkerSize',12,'LineWidth',2)
plot(ctrs(:,1),ctrs(:,2),'ko',...
'MarkerSize',12,'LineWidth',2)
legend('Cluster 1','Cluster 2','Centroids',...
'Location','NW')
title('K-means with normalization')
```





9. When it is important to standardize variables?

- It is important to standardize variables before running **Cluster Analysis**. It is because cluster analysis techniques depend on the concept of measuring the distance between the different observations we're trying to cluster. If a variable is measured at a higher scale than the other variables, then whatever measure we use will be overly influenced by that variable.

- Prior to **Principal Component Analysis**, it is critical to standardize variables. It is because PCA gives more weightage to those variables that have higher variances than to those variables that have very low variances. In effect the results of the analysis will depend on what units of measurement are used to measure each variable. Standardizing raw values makes equal variance so high weight is not assigned to variables having higher variances.

- It is required to standardize variable before using **k-nearest neighbors** with an Euclidean distance measure. Standardization makes all variables to contribute equally.

- All SVM kernel methods are based on distance so it is required to scale variables prior to running final **Support Vector Machine (SVM)** model.
- It is necessary to standardize variables before using **Lasso and Ridge Regression.** Lasso regression puts constraints on the size of the coefficients associated to each variable. However, this value will depend on the magnitude of each variable. The result of centering the variables means that there is no longer an intercept. This applies equally to ridge regression.
- In regression analysis, we can calculate **importance of variables** by ranking independent variables based on the descending order of absolute value of standardized coefficient.
- In regression analysis, when an interaction is created from two variables that are not centered on 0, some amount of collinearity will be induced. **Centering first addresses this potential problem**. In simple terms, having non-standardized variables interact simply means that when X1 is big, then X1X2 is also going to be bigger on an absolute scale irrespective of X2, and so X1 and X1X2 will end up correlated.
- In regression analysis, it is also helpful to standardize a variable when you include power terms X^2. Standardization removes collinearity.

10. What is the difference between normalization and standardization

- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbours and Neural Networks.
- Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.
- However, at the end of the day, the choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using.
- There is no hard and fast rule to tell you when to normalize or standardize your data. You can always start by fitting your model to raw, normalized and standardized data and compare the performance for best results.
- It is a good practice to fit the scaler on the training data and then use it to transform the testing data. This would avoid any data leakage during the model testing process. Also, the scaling of target values is generally not required.

11. **What is the difference between sklearn.preprocessing import MinMaxScaler Normalization and sklearn.preprocessing.Normalizer? When to use MinMaxScaler and when to Normalize?**

Normalizer is also a normalization technique. The only difference is the way it computes the normalized values. By default, it is calculating the l2 norm of the row values i.e. each element of a row is normalized by the square root of the sum of squared values of all elements in that row.
As mentioned in the documentation, it is useful in text classification where the dot product of two Tf-IDF vectors gives a cosine similarity between the different sentences/documents in the dataset. Other than that, as I mentioned in the article, there is no sure way to know which scaling technique should be used when. The best way is to create multiple scaled copies of the data and then try them out and see which one gives the best result.

12. Write a code for implementation of standardization in python or R programming?

Normalization using sklearn

To normalize your data, you need to import the *MinMaxScalar* from the sklearn library and apply it to our dataset.

```
# data normalization with skilearn

From sklearn.preprocessing import MinMaxScaler

#fit scaler on training data

Norm= MinMaxScaler().fit(X_train)

#transform training data

X_train_norm = norm.transform(X_train)

#transform testing databs

X_test_norm= norm.transform(X_test)
```

Standardization using sklearn

To standardize your data, you need to import the *StandardScalar* from the sklearn library and apply it to our dataset.

```
# data standardization with  sklearn
from sklearn.preprocessing import StandardScaler

# copy of datasets
X_train_stand = X_train.copy()
```

```
X_test_stand = X_test.copy()

# numerical features
num_cols = ['Item_Weight','Item_Visibility','Item_MRP','Outlet_Establishment_Year']

# apply standardization on numerical features
for i in num_cols:

# fit on training data column
scale = StandardScaler().fit(X_train_stand[[i]])

# transform the training data column
X_train_stand[i] = scale.transform(X_train_stand[[i]])

# transform the testing data column
X_test_stand[i] = scale.transform(X_test_stand[[i]])
```

It is always great to visualize your data to understand the distribution present. We can see the comparison between our unscaled and scaled data using boxplots.What algorithms need feature scaling

Algorithm(s)	Reason of applying feature scaling
1. K-Means	Use the Euclidean distance measure.
2. K-Nearest-Neighbours	Measure the distances between pairs of samples and these distances are influenced by the measurement units
3. Principal Component Analysis (PCA)	Try to get the feature with maximum variance
4. Artificial Neural Network	Apply Gradient Descent
5. Gradient Descent	Theta calculation becomes faster after feature scaling and the learning rate in the update equation of Stochastic Gradient Descent is the same for every parameter

13. **Let's see the effect of perform a PCA on the standardized and the non-standardized datasets**

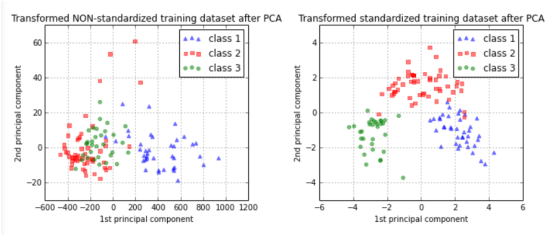
```
from sklearn.decomposition import PCA


# on non-standardized data

pca = PCA(n_components=2).fit(X_train)

X_train = pca.transform(X_train)

X_test = pca.transform(X_test)
```

```
# om standardized data

pca_std = PCA(n_components=2).fit(X_train_std)

X_train_std = pca_std.transform(X_train_std)

X_test_std = pca_std.transform(X_test_std)
```

Let us quickly visualize how our new feature subspace looks like (note that class labels are not considered in a PCA - in contrast to a Linear Discriminant Analysis - but I will add them in the plot for clarity).

```
from matplotlib import pyplot as plt


fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10,4))



for l,c,m in zip(range(1,4), ('blue', 'red', 'green'), ('^', 's', 'o')):

ax1.scatter(X_train[y_train==l, 0], X_train[y_train==l, 1],

color=c,

label='class %s' %l,

alpha=0.5,

marker=m

)


for l,c,m in zip(range(1,4), ('blue', 'red', 'green'), ('^', 's', 'o')):

ax2.scatter(X_train_std[y_train==l, 0], X_train_std[y_train==l, 1],

color=c,

label='class %s' %l,

alpha=0.5,

marker=m

)
```

```
ax1.set_title('Transformed NON-standardized training dataset after PCA')

ax2.set_title('Transformed standardized training dataset after PCA')


for ax in (ax1, ax2):


ax.set_xlabel('1st principal component')

ax.set_ylabel('2nd principal component')

ax.legend(loc='upper right')

ax.grid()

plt.tight_layout()


plt.show()
```



Lets check the classification accuracy with and without standardization

```
from sklearn import metrics


pred_train = gnb.predict(X_train)


print('\nPrediction accuracy for the training dataset')

print('{:.2%}'.format(metrics.accuracy_score(y_train, pred_train)))


pred_test = gnb.predict(X_test)


print('\nPrediction accuracy for the test dataset')

print('{:.2%}\n'.format(metrics.accuracy_score(y_test, pred_test)))
```

```
Prediction accuracy for the training dataset

81.45%


Prediction accuracy for the test dataset

64.81%
```

```
pred_train_std = gnb_std.predict(X_train_std)


print('\nPrediction accuracy for the training dataset')

print('{:.2%}'.format(metrics.accuracy_score(y_train, pred_train_std)))


pred_test_std = gnb_std.predict(X_test_std)


print('\nPrediction accuracy for the test dataset')

print('{:.2%}\n'.format(metrics.accuracy_score(y_test, pred_test_std)))
```

```
Prediction accuracy for the training dataset

96.77%


Prediction accuracy for the test dataset

98.15%
```

As we can see, the standardization prior to the PCA definitely led to an decrease in the empirical error rate on classifying samples from test dataset.