Why, How and When to Scale your Features

Feature scaling can vary your results a lot while using certain algorithms and have a minimal or no effect in others. To understand this, let's look at why features need to be scaled, varieties of scaling methods and when we should scale our features.

Why Scaling

Most of the time, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use **Euclidean distance** between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg and 5000 gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.

To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by **scaling**.

How to Scale Features

There are four common methods to perform Feature Scaling.

1. **Standardization:**

Standardization replaces the values by their Z scores.

$$x' = \frac{x - \bar{x}}{\sigma}$$

This redistributes the features with their mean **μ = 0** and standard deviation **σ =1** . `sklearn.preprocessing.scale` helps us implement standardisation in python.

2. Mean Normalization:

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

This distribution will have values between **-1 and 1**with **μ=0**.

Standardisation and **Mean Normalization** can be used for algorithms that assume zero centric data like **Principal Component Analysis(PCA).**

3. Min-Max Scaling:

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

This scaling brings the value between 0 and 1.

4. Unit Vector:

$$x' = \frac{x}{||x||}$$

Scaling is done considering the whole feature vecture to be of unit length.

Min-Max Scaling and **Unit Vector** techniques produce values of range [0,1]. When dealing with features with hard boundaries this is quite useful. For example, when dealing with image data, the colors can range from only 0 to 255.

When to Scale

Rule of thumb I follow here is any algorithm that computes distance or assumes normality, **and scales your features!!!**

Some examples of algorithms where feature scaling matters are:

- **k-nearest neighbors** with an Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
- Scaling is critical, while performing **Principal Component Analysis(PCA)**. PCA tries to get the features with maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features.
- We can speed up **gradient descent** by scaling. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.
- **Tree based models** are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modelling trees.
- Algorithms like **Linear Discriminant Analysis(LDA), Naive Bayes** are by design equipped to handle this and give weights to the features accordingly. Performing a features scaling in these algorithms may not have much effect.

feature scaling is necessary while using ridge and lasso regression; basically using regularization parameters(L2 and L1 norm). Standard least squares doesn't often require them but while using regularization with Standard least squares, scaling is necessary