Feature Scaling – Why & When?

The Why?

Features consists of two parts: Magnitude and Unit.

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

If unscaled, these algorithms only take in the magnitude of features neglecting the units. **The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes**. For example, A variable that ranges between 0 and 1000 will outweigh a variable that ranges between 0 and 1. To supress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.

There are 4 ways to scale features:

**1. Standardisation:** Standardisation replaces the values by their Z scores. This redistributes the features with their mean μ = 0 and standard deviation σ =1

**2. Mean Normalisation:** This distribution will have values between -1 and 1 with μ=0.

**Standardisation** and **Mean Normalization** can be used for algorithms that assumes zero centric data like Principal Component Analysis(PCA). So if you are going to use the features for PCA, you can use z-transform.

**3. Min-Max Scaling (Normalization):** This scaling brings the values between 0 and 1.

**4. Unit Vector:** This scaling brings the value between 0 and 1.

The when?

When Should You Use Normalization And Standardization:

***Normalization*** is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve). In this approach, the data is scaled to a fixed range — **usually 0 to 1.**

In contrast to standardization, the cost of having this bounded range is that we will end up with smaller standard deviations, which can suppress the effect of outliers. Thus Min/Max Scalar is sensitive to outliers.

Normalization is useful when your data has varying scales and the algorithm you are using does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.

***Standardization*** assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian.

However, the minimum and maximum values after rescaling, vary according to how spread out the variable was, to begin with, and is highly influenced by the presence of outliers.

Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression, and linear discriminant analysis.

**Min-Max scaling and Unit Vector scaling:**

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.

Some examples of algorithms where feature scaling is required are:

|  |  |
| --- | --- |
| **Required** | **Not required** |
| k-nearest neighbors  Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally. | Decision Trees |
| 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. | Random Forests |
| Linear Regression  We can speed up gradient descent by scaling | XG boost |
|  | Linear Discriminant Analysis(LDA) |
| Naive Bayes |