OSEMN methodology
Step 4: Model
Feature scaling
Excerpts from Python Machine Learning
Second Edition
By Sebastian Raschka and Vahid Mirjalili[1]
and other sources

Stepan Oskin

September 14, 2019

Abstract

1 Feature scaling

1.1 From Raschka and Mirjalili[1]

Feature scaling is a crucial step in our preprocessing pipeline that can easily be forgotten. Decision trees and random forests are two of the very few machine learning algorithms where we don't need to worry about feature scaling. Those algorithms are scale invariant. However, the majority of machine learning and optimization algorithms behave much better if features are on the same scale.

The importance of feature scaling can be illustrated by a simple example. Let's assume that we have two features where one feature is measured on a scale from 1 to 10 and the second feature is measured on a scale from 1 to 100,000, respectively. When we think of the squared error function in Adaline, it is intuitive to say that the algorithm will mostly be busy optimizing the weights according to the larger errors in the second feature. Another example is the k-nearest neighbors (KNN) algorithm with a Euclidean distance measure; the computed distances between samples will be dominated by the second feature axis.

Now, there are two common approaches to bring different features onto the same scale: normalization and standardization. Those terms are often used quite loosely in different fields, and the meaning has to be derived from the context. Most often, normalization refers to the rescaling of the features to a range of [0, 1], which is a special case of min-max scaling. To normalize our data, we can simply apply the min-max scaling to each feature column, where the new value $x^(i)_{norm}$ of a sample $x^(i)$ can be calculated as follows:

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Here, $x^{(i)}$ is a particular sample, x_{min} is the smallest value in a feature column, and x_{max} the largest value.

Although normalization via min-max scaling is a commonly used technique that is useful when we need values in a bounded interval, standardization can be more practical for many machine learning algorithms, especially for optimization algorithms such as gradient descent. The reason is that many linear models, such as the logistic regression and SVM, initialize the weights to 0 or small random values close to 0. Using standardization, we center the feature columns at mean 0 with standard deviation 1 so that the feature columns have the same parameters as a standard normal distribution (zero mean and unit variance), which makes it easier to learn the weights. Furthermore, standardization maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling, which scales the data to a limited range of values.

The procedure for standardization can be expressed by the following equation:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x} \tag{2}$$

Here, μ_x is the sample mean of a particular feature column and σ_x is the corresponding standard deviation.

Table 1 illustrates the difference between the two commonly used feature scaling techniques, standardization and normalization, on a simple sample dataset consisting of numbers 0 to 5:

Variables can be standardized using the `scikit-learn` objects `StandardScaler` and `MinMaxScaler`. Again, it is also important to highlight that we fit Stan-

Input	Standardized	Min-max normalized
0.0	-1.46385	0.0
1.0	-0.87831	0.2
2.0	-0.29277	0.4
3.0	0.29277	0.6
4.0	0.87831	0.8
5.0	1.46385	1.0

Table 1: Difference between the two commonly used scaling techniques, standardization and normalization, on a simple sample dataset consisting of numbers 0 to 5.

dardScaler class only once—on the training data—and use those parameters to transform the test set or any new data point.

1.2 From Jason Brownlee[2], wikipedia and Andrew Ng's lecture

Jason Brownlee's blog:
https://machinelearningmastery.com/normalize-standardize-time-series-data-python/
wikipedia article on feature scaling:
https://en.wikipedia.org/wiki/Feature_scaling
Andrew Ng's lecture on feature scaling:
http://openclassroom.stanford.edu/MainFolder/VideoPage.php?course=MachineLearning&video=03.1-LinearRegressionII-FeatureScaling&speed=100/

Some machine learning algorithms will achieve better performance if data has a consistent scale or distribution. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization.

For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it. In stochastic gradient descent, feature scaling can sometimes improve the convergence speed of the algorithm. In support vector machines, it can reduce the time to find support vectors. Note that feature scaling changes the SVM result.

Two techniques that can be used to consistently rescale data are :
Normalization

- Also known as feature scaling or unity-based normalization
- Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1.
- Normalization can be useful, and even required in some machine learning algorithms when data has input values with differing scales.
- It may be required for algorithms, like k-Nearest neighbors, which uses distance calculations and Linear Regression and Artificial Neural Networks that weight input values.
- Normalization requires the knowledge or accurate estimation of the minimum and maximum observable values (can be estimated from the available data).
- If needed, the transform can be inverted. This is useful for converting predictions back into their original scale for reporting or plotting.
- If the data presents a time series that is trending up or down, estimating these expected values may be difficult and normalization may not be the best method to use.
- Variables can be normalized using the ‘scikit-learn‘ object ‘MinMaxScaler‘.
- Types of normalization:
  - Rescaling (min-max normalization)

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

  - Rescaling between an arbitrary set of values

$$X' = a + \frac{(X - X_{min})(b - a)}{X_{max} - X_{min}} \tag{4}$$

  - Mean normalization

$$X' = \frac{X - \mu_X}{X_{max} - X_{min}} \tag{5}$$

Standardization (Z-score normalization)

- Standardization is another type of rescaling that is more robust to new values being outside the range of expected values than normalization.
- Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance.
- This can be thought of as subtracting the mean value, or centering the data, and scaling by standard deviation.
- Like normalization, standardization can be useful, and even required in some machine learning algorithms when data has input values with differing scales.
- This method is widely used for normalization in many machine learning algorithms (e.g., support vector machines, logistic regression, and artificial neural networks).
- Standardization assumes that observations fit a Gaussian distribution(bell curve) with a well behaved mean and standard deviation.
- Data can still be standardized if this expectation is not met, but results might not be reliable.
- Standardization requires the knowledge or accurate estimation of the mean and standard deviation of observable values.
- These values can be estimated from training data.
- Types of standardization:
  - General standardization
    $$X' = \frac{X - \mu_X}{\sigma} \tag{6}$$
    , where μ_X is the mean of the feature and σ is its standard deviation
  - Scaling to unit length Another option that is widely used in machine-learning is to scale the components of a feature vector such that the complete vector has length one. This usually means dividing each component by the Euclidean length of the vector:
    $$X' = \frac{X}{||X||_2} \tag{7}$$

- In some applications (e.g. Histogram features) it can be more practical to use the L1 norm (i.e. Manhattan Distance, City-Block Length or Taxicab Geometry) of the feature vector.
- This is especially important if in the following learning steps the Scalar Metric is used as a distance measure.

References

[1] S. Raschka and V. Mirjalili, *Python Machine Learning, 2nd Ed.* . Birmingham, UK: Packt Publishing, 2 ed., 2017.

[2] J. Brownlee, "How to Prepare Data For Machine Learning," 2013.