Feature Scaling

What is feature scaling? Feature scaling is a way of transforming your data into a common range of values. There are two common scalings:

1. Standardizing
2. Normalizing

Standardizing

Standardizing is completed by taking each value of your column, subtracting the mean of the column, and then dividing by the standard deviation of the column. In Python, let's say you have a column in `df` called `height`. You could create a standardized height as:

```
df["height_standard"] = (df["height"] - df["height"].mean()) / df["height"].std()
```

This will create a new "standardized" column where each value is a comparison to the mean of the column, and a new, standardized value can be interpreted as the number of standard deviations the original height was from the mean. This type of feature scaling is by far the most common of all techniques (for the reasons discussed here, but also likely because of precedent).

Normalizing

A second type of feature scaling that is very popular is known as **normalizing**. With normalizing, data are scaled between 0 and 1. Using the same example as above, we could perform normalizing in Python in the following way:

```
df["height_normal"] = (df["height"] - df["height"].min()) /     \
                      (df["height"].max() - df['height'].min())
```

When Should I Use Feature Scaling?

In many machine learning algorithms, the result will change depending on the units of your data. This is especially true in two specific cases:

1. When your algorithm uses a distance-based metric to predict.
2. When you incorporate regularization.

Distance Based Metrics

In future lessons, you will see one common supervised learning technique that is based on the distance points are from one another called Support Vector Machines (or SVMs). Another technique that involves distance based methods to determine a prediction is k-nearest neighbors (or k-nn). With either of these techniques, choosing not to scale your data may lead to drastically different (and likely misleading) ending predictions.

For this reason, choosing some sort of feature scaling is necessary with these distance based techniques.

Regularization

When you start introducing regularization, you will again want to scale the features of your model. The penalty on particular coefficients in regularized linear regression techniques depends largely on the scale associated with the features. When one feature is on a small range, say from 0 to 10, and another is on a large range, say from 0 to 1 000 000, applying regularization is going to unfairly punish the feature with the small range. Features with small ranges need to have larger coefficients compared to features with large ranges in order to have the same effect on the outcome of the data. (Think about how $ab = ba$ for two numbers a and b.) Therefore, if regularization could remove one of those two features with the same net increase in error, it would rather remove the small-ranged feature with the large coefficient, since that would reduce the regularization term the most.

Again, this means you will want to scale features any time you are applying regularization.

- A useful Quora post on the importance of feature scaling when using regularization.

A point raised in the article above is that feature scaling can speed up convergence of your machine

A point raised in the article above is that feature scaling can speed up convergence of your machine learning algorithms, which is an important consideration when you scale machine learning applications.

Use the quiz below to get some practice with feature scaling.

Feature Scaling Exercise

Previously, you saw how regularization will remove features from a model (by setting their coefficients to zero) if the penalty for removing them is small. In this exercise, you'll revisit the same dataset as before and see how scaling the features changes which features are favored in a regularization step. See the "Quiz: Regularization" page for more details. The only thing different for this quiz compared to the previous one is the addition of a new step after loading the data, where you will use sklearn's `StandardScaler` to standardize the data before you fit a linear regression model to the data with L1 (Lasso) regularization.

Perform the following steps:

1. Load in the data

- The data is in the file called 'data.csv'. Note that there's no header row on this file.
- Split the data so that the six predictor features (first six columns) are stored in `X`, and the outcome feature (last column) is stored in `y`.

2. (NEW) Perform feature scaling on data via standardization

- Create an instance of sklearn's `StandardScaler` and assign it to the variable `scaler`.
- Compute the scaling parameters by using the `.fit_transform()` method on the predictor feature array, which also returns the predictor variables in their standardized values. Store those standardized values in `X_scaled`.

3. Fit data using linear regression with Lasso regularization

- Create an instance of sklearn's `Lasso` class and assign it to the variable `lasso_reg`. You don't need to set any parameter values: use the default values for the quiz.
- Use the `Lasso` object's `.fit()` method to fit the regression model onto the data. Make sure that you apply the fit to the *standardized* data from the previous step (`X_scaled`), not the original data.

4. Inspect the coefficients of the regression model

- Obtain the coefficients of the fit regression model using the `.coef_` attribute of the `Lasso` object. Store this in the `reg_coef` variable: the coefficients will be printed out, and you will use your observations to answer the question at the bottom of the page.

feature_scaling.py | data.csv | solution.py

```
# TODO: Add import statements
import numpy as np
import pandas as pd
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler

# Assign the data to predictor and outcome variables
# TODO: Load the data
train_data = pd.read_csv('data.csv', header = None)
X = train_data.iloc[:,:-1]
y = train_data.iloc[:,-1]

# TODO: Create the standardization scaling object.
scaler = StandardScaler()

# TODO: Fit the standardization parameters and scale the data.
X_scaled = scaler.fit_transform(X)

# TODO: Create the linear regression model with lasso regularization.
lasso_reg = Lasso()

# TODO: Fit the model.
lasso_reg.fit(X_scaled, y)

# TODO: Retrieve and print out the coefficients from the regression model.
reg_coef = lasso_reg.coef_
```

RESET QUIZ TEST RUN SUBMIT ANSWER

QUESTION 2 OF 2

For which of the predictor features has the lasso regularization step zeroed the corresponding coefficient, on the standardized data?

- [x] Column X1
- [ ] Column X2
- [ ] Column X3
- [x] Column X4
- [ ] Column X5
- [ ] Column X6

SUBMIT

NEXT