BTP Weekly Report 1
February 2, 2018

---

1 Problem Statement

We consider a scenario where we imagine ourselves as Botanists and we want to classify **Iris** flowers. We chose machine learning as our tool to do the task. For instance, a sophisticated machine learning program could classify flowers based on photographs. Our ambitions are more modest– we're going to classify Iris flowers based solely on the length and width of their sepals and petals.

The Iris genus entails about 300 species, but our program will classify only the following three:

- Iris setosa
- Iris virginica
- Iris versicolor

2 Dataset

Fortunately, this being our starting problem for machine learning we don't need a very huge amount of data. We require training datasets and test datasets. There is a set of 120 entries for this problem which we will use. Refer Dataset.

There are five columns in this dataset. The first four are feature and and the last column is the label. Each label is naturally a string (for example, "setosa"), but machine learning typically relies on numeric values. Therefore, someone mapped each string to a number. Here's the representation scheme:

- 0 for Setosa
- 1 for Versicolor
- 2 for Virginica

3 Models and Training

The Iris classification problem is an example of supervised machine learning in which a model is trained from examples that contain labels.

A **model** is the relationship between features and the label. For the Iris problem, the model defines the relationship between the sepal and petal measurements and the Iris species.

4 Analysis of Dataset

We have a dataset that contains 5 features where we train our model with 120 examples and test them on the rest 30 examples.
The tools used in the plotting of different type of graphs for the analysis purpose are:

- Matplotlib
- Seaborn
- Pandas

We plot a lot of different type of curves and graphs that can show a relation between the quantites like mean, median etc with the features. In the features in our dataset, we have one feature called **ID**. It's nothing but a serial number given to each example.
To verify that our labels are correctly assigned, we'd verify it using **Jupyter Notebook**. It's our choice of playground where we will do our analysis.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print le
print y
```

And as expected it outputs an array of 50 0's, 50 1's and 50 2's.

BTP Weekly Report 2
February 23, 2018

Data Visualization

When comparing variables in a multidimensional or multivariate dataset, some conclusions must be drawn from the patterns in the dataset. In comparing different variables it is good practice to first make an educated assumption on what type of patterns you wish to find.

- Parallel Coordinates
- Andrews Curves
- Pair Plot
- Box Plot

Parallel Coordinates

Parallel coordinates is a plotting technique for plotting multivariate data. It allows one to see clusters in data and to estimate other statistics visually. Using parallel coordinates points are represented as connected line segments. Each vertical line represents one attribute. One set of connected line segments represents one data point. Points that tend to cluster will appear closer together.

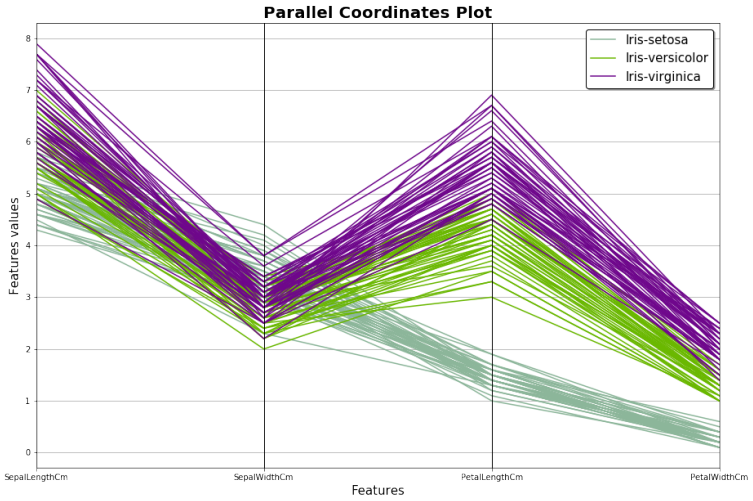

Figure 1: Parallel Coordinates Plot

Andrews Curves

Andrews curves allow one to plot multivariate data as a large number of curves that are created using the attributes of samples as coefficients for Fourier series. By coloring these curves differently for each class it is possible to visualize data clustering. Curves belonging to samples of the same class will usually be closer together and form larger structures.

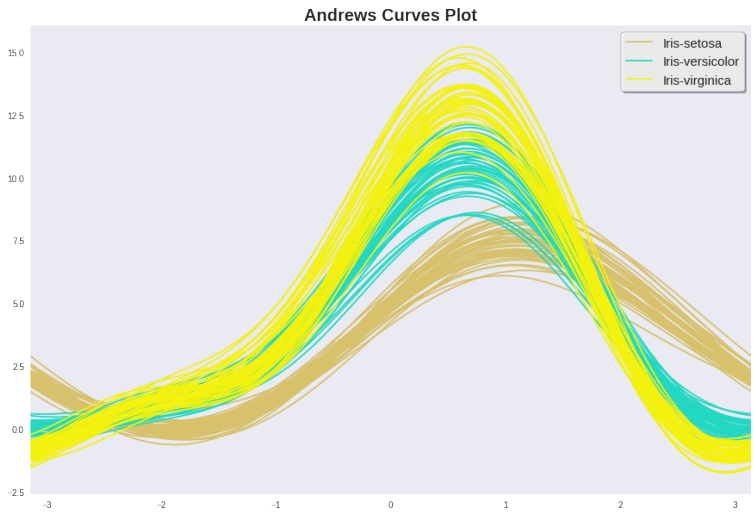

Figure 2: Andrews Curves Plot

Pair Plot

The first scatterplot graph compares the sepal length with the classification of flower. The second scatterplot graph compares the sepal width with the classification of flower. The third scatterplot graph compares the petal length with the classification of flower. The fourth scatterplot graph compares the petal width with the classification of flower. From these four scatterplots we can determine a pattern and therefore create a possible predictor.

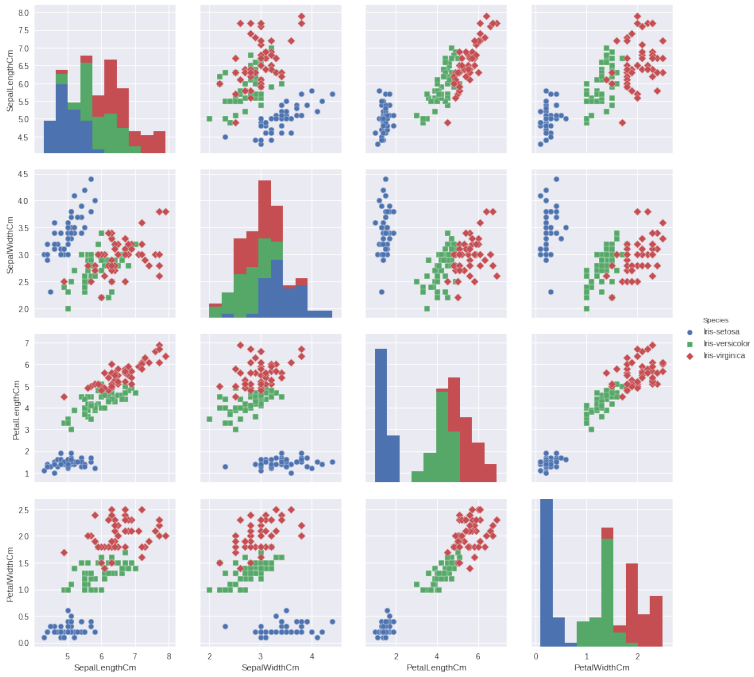

Figure 3: Pair Plot

Box Plot

After you check the distribution of the data by plotting the histogram, the second thing to do is to look for outliers. Identifying the outliers is important because it might happen that an association you find in your analysis can be explained by the presence of outliers.

Through box plots we find the minimum, lower quartile (25th percentile), median (50th percentile), upper quartile (75th percentile), and maximum of an continues variable.Each horizontal line starting from bottom will show the minimum, lower quartile, median, upper quartile and maximum value.

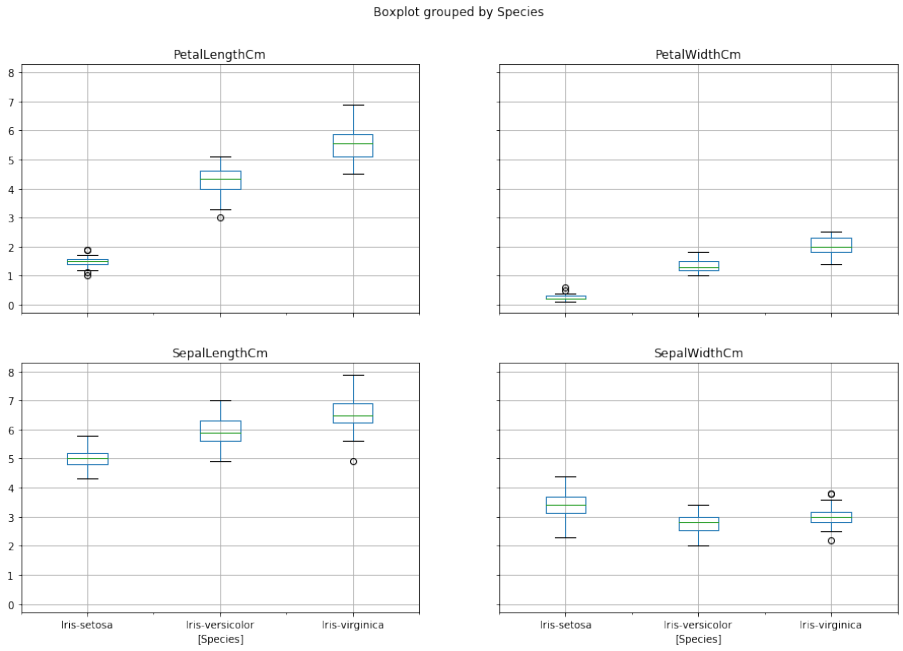

Figure 4: Box Plot

Cost Function

Cost functions are used to estimate how badly models are performing. A cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y. This cost function can be estimated by iteratively running the model to compare estimated predictions against the known values of y.

The objective of a ML model, therefore, is to find parameters, weights or a structure that minimises the cost function.The parameter which minimizes cost function is gradient descent.Gradient descent is an efficient optimization algorithm that attempts to find a local or global minima of a function.

22 lines (16 sloc) | 569 Bytes — Raw | Blame | History

```
function [J,grad] = IrisCost(theta,X,y,lambda)
%number of training example in dataset
m=size(X,1);
 %gradient should have  same size as that of theta
 grad=zeros(size(theta));

 J=0;
 %regularized costfunction
J=(1/m)*sum(-y'*log(sigmoid(X*theta))-(1-y)'*log(1-sigmoid(X*theta))) +(lambda/(2*m))*sum((theta(2:length(theta))).*theta(2:length(theta)))


%grad as sam size as of theta
grad=(1/m)*sum(X.*repmat((sigmoid(X*theta)-(y)),1,columns(X)));
temp=theta;
temp(1)=0;
grad(1)=grad(1);
grad(2:end)=(grad(2:end))' + ((lambda/m)*temp(2:end));

 grad=grad(:);

end
```

Figure 5: Cost Function

'X' is the number of features in the data set and 'y' is the corresponding predictive value.

4.1 Sigmoid Function

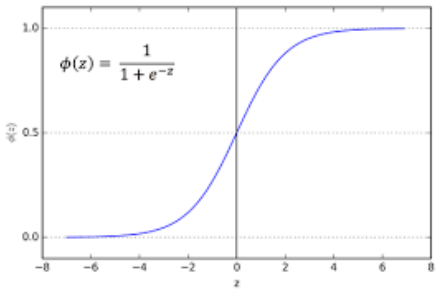

Figure 6: Sigmoid Function

For classification, as in the labeling iris task, linear regression is not the right approach as it will give too much weight to data far from the decision frontier. A linear approach is to fit a sigmoid function or logistic function.

4.2 Multiclass Classification: One-vs-all

Now we will approach the classification of data when we have more than two categories. Instead of y = 0,1 we will expand our definition so that y = 0,1...n. Since y = 0,1...n, we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

13 lines (9 sloc) | 253 Bytes

```
function [all_theta]=onevsall(X,label,y)

lambda=.3
n=size(X,2);
theta_t=zeros(n,1);
all_theta=zeros(label,n);

options=optimset('GradObj','on','MaxIter',50);
for i=1:label

[all_theta(i,:)]=fminunc(@(t)IrisCost(t,X,(y==i),lambda),theta_t,options);
end
```

Figure 7: One-vs-all

Updates Theta for minimizing costfunction after 50 iterations.Updated theta is passed on to predictall function which predicts index of species.

4.3 Predictall

Predictall function predicts the index of species.

7 lines (5 sloc) | 93 Bytes

```
function p=predictall(X,theta);


p=sigmoid(X*theta');
[pr index]=max(p,[],2);
 p=index;
 end
```

Figure 8: Predictall

Now taking the mean gives accuracy of training model.