Post02-kaiqian-zhu

kaiqianzhu

11/29/2017

K-Nearest Neighbors

Motivation/Introduction

Nowadays, machine learning is so popular that almost everyone hears about it, but most people don't know what it really is, even computer scientists in other fields know only a little about machine learning knowledge. In this post, as a computer science major student, I hope I can give a clear and insightful introduction to machine learning by exploring an interesting machine learning algorithm: **K Nearest Neighbors(KNN)** with you.

Machine learning is a broad topic. It is a branch in computer science that studies the design of algorithms that can learn. For simplicity, we can understand ML as a two steps operation. **First**, as the name implied, **learning** from the **existing data**. **Second**, using what we have learned to **predicted or classified** the **new input data** that we have not seen before. **In the case of KNN**, we assume that we have several groups of labeled samples (existing data), and each labeled sample has certain features which identify it as a member of the group it is labeled. Now, suppose we have an unlabeled input sample which needs to be classified as one of the several groups, how can we do that? The short answer is by applying KNN algorithm. But how to perform the KNN algorithm? You may ask. The detail you want to know is explained in below sections :).

Application

Step 1: Learn!
To be able learn something, we first need to prepare the data that we wanted to learn from, namely, the **exiting data**. In order to achieve reproducibility, we make use of the R build-in datase: **Iris** or we can import it from the UC Irvine machine learning repository.

```r
# iris is a R buid-in dataset
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
# To learn how to import dataset from well-known repository, we do the below import.

iris <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"),
                 header = FALSE)
head(iris)
```

```
##    V1  V2  V3  V4          V5
## 1 5.1 3.5 1.4 0.2 Iris-setosa
## 2 4.9 3.0 1.4 0.2 Iris-setosa
## 3 4.7 3.2 1.3 0.2 Iris-setosa
## 4 4.6 3.1 1.5 0.2 Iris-setosa
## 5 5.0 3.6 1.4 0.2 Iris-setosa
## 6 5.4 3.9 1.7 0.4 Iris-setosa
```

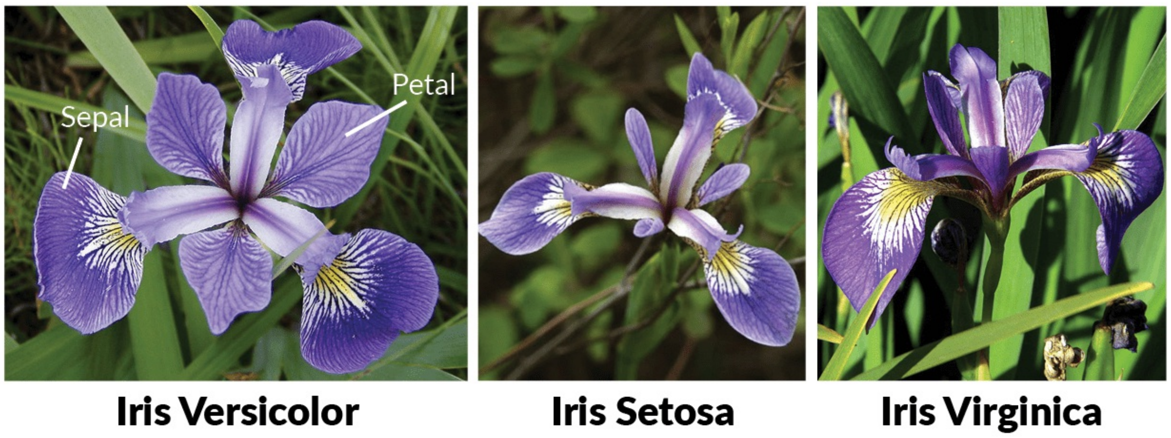
Notice the difference between the buid-in iris and the uci version iris dataset, we have to make additional change to the column name to make the dataset more clear!

```r
# modify the column name. Although the species name is different, it does not affect the meaning of the dataset.
names(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")

head(iris)
```

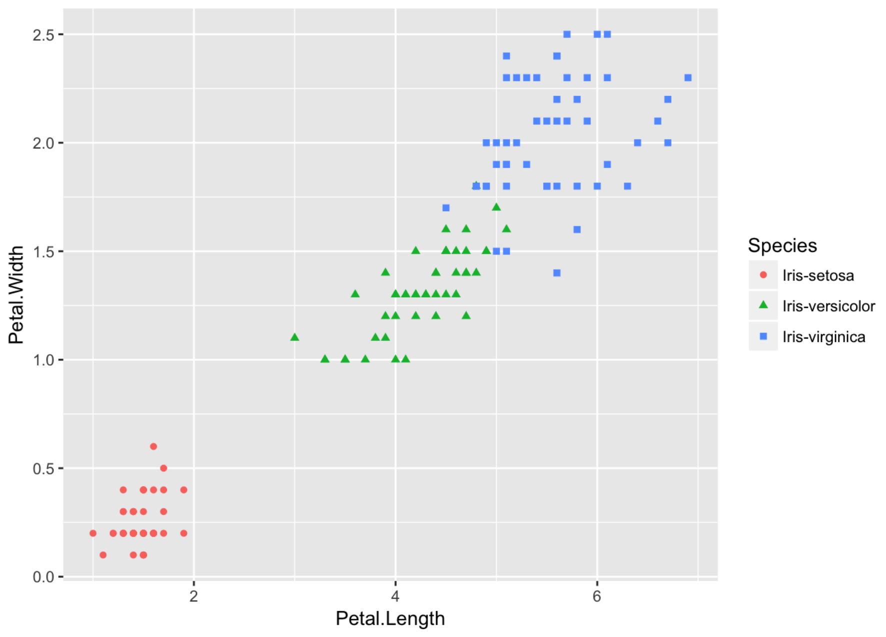
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width     Species
## 1          5.1         3.5          1.4         0.2 Iris-setosa
## 2          4.9         3.0          1.4         0.2 Iris-setosa
## 3          4.7         3.2          1.3         0.2 Iris-setosa
## 4          4.6         3.1          1.5         0.2 Iris-setosa
## 5          5.0         3.6          1.4         0.2 Iris-setosa
## 6          5.4         3.9          1.7         0.4 Iris-setosa
```

You can now see the look the same! But you may say, wait a second, what the * is iris, I never heard of it? Before we move on to more interesting stuff, let me show you some picture of the irirs.

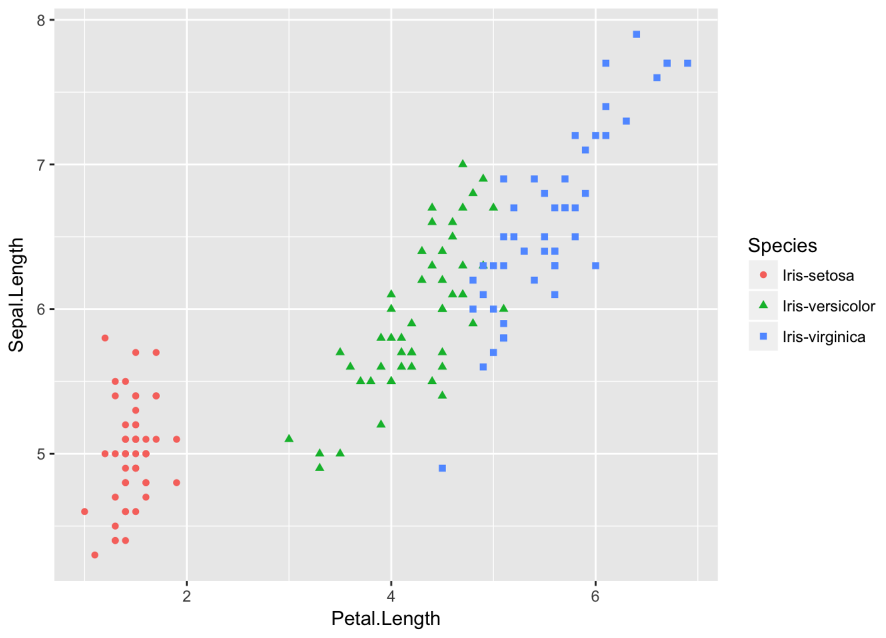


Above is the pictures of different groups of Iris that we are going the classify. We may already notice that the existing dataset we are working with comes with features that help us to classify the sample. They are spal length, spal width, petal length and petal width. Say, we only interested in 2d workplace, which 2 features should we pick in order to better classify the 3 groups? In other words, which 2 features make the 3 groups lineary separable? Let's observe from the below plots.

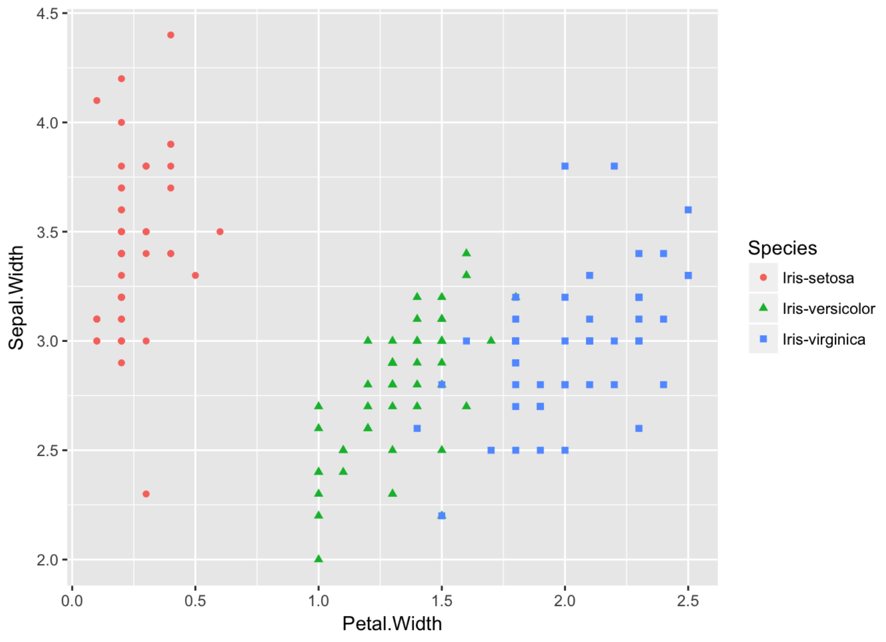
```
ggplot(iris, aes(Petal.Length, Petal.Width, group=Species)) +
  geom_point(aes(shape=Species, color=Species))
```



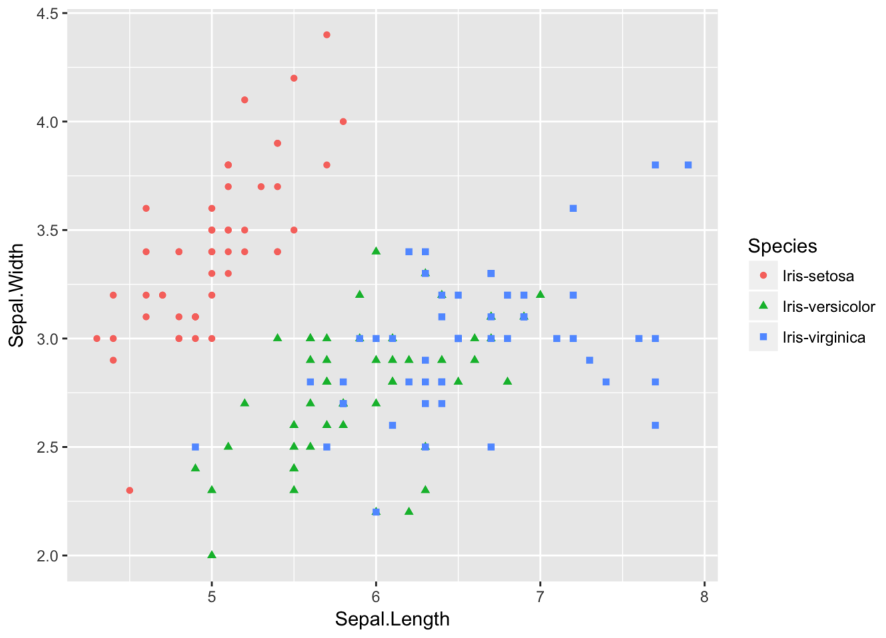
```
ggplot(iris, aes(Petal.Length, Sepal.Length, group=Species)) +
  geom_point(aes(shape=Species, color=Species))
```



```
ggplot(iris, aes(Petal.Width, Sepal.Width, group=Species)) +
  geom_point(aes(shape=Species, color=Species))
```



```
ggplot(iris, aes(Sepal.Length, Sepal.Width, group=Species)) +
  geom_point(aes(shape=Species, color=Species))
```

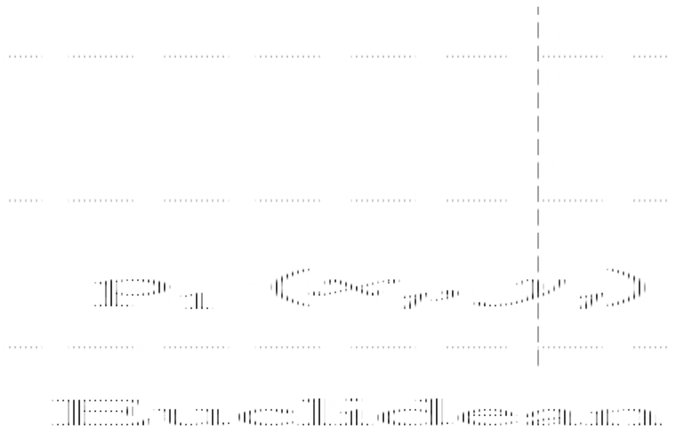


Maybe we would like to pick Petal.Length, Petal.Width if we can only operate in 2d, since points from a group in our existing dataset are not overlapping with other groups' points.
what if we can do this in 3d?

```
plot_ly(iris, x = ~Petal.Length, y = ~Petal.Width, z = ~Sepal.Width, color = ~Species) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Petal Length'),
                      yaxis = list(title = 'Petal Width'),
                      zaxis = list(title = 'Sepal Width')))
```

We will learn how to implement this and how to test our model accuracy in below coding section.

Step 2: Classification!
Now what should we do when a new data sample comes in, how do we find the k-nearest neighbors and how to we classify after we find the knn? There are many ways of doing so. The most popular way is to determine the knn by eucildean distance:



And after that classify the input new data point as the group of the majority of the k nearest neighbors labeled group. Let say we set k to be 7 and if 5 of the knn are in Setosa, our new data point can be classify as Setosa.

Coding
Now let see how can we implement the KNN algorithm. There is one more question we have not discussed above. That is how do we measure the accuracy of the model we trained in the **Learn** part. The answer is we reserve a portion of the existing labeled dataset to test our model(usally 1/4). More advanced method to test like k-fold Cross-Validation can be learned in CS189.
Since most of us are not CS major, we can make use of some existing machine learning package to help us achieve the same purpose by writing more concise and clean code. In R, we can use **caret**.
As mentioned above, we separate our existing dataset into train dataset and test dataset by:

```r
# partition the dataset based on Species label
index <- createDataPartition(iris$Species, p=0.75, list=FALSE)
# subset train dataset with index we get above
iris.train_dataset <- iris[index,]
# subset test dataset with index we get above
iris.test_dataset <- iris[-index,]
```

We have imported the caret package at the beginning and it can perform a lot of ml algorithm and I strongly encourage you to try some of them later. You can also try more advanced library like tensorflow in python if you find machine learning very interesting.

```r
# Train the model, as we discussed in the Learn step section.
# notice we utilized all 4 features here. iris.train_dataset[,1:4]
# We discussed 2 features and 3 features case
# by seting only the object to be knn, we can apply the desired machine learning alg.
model_knn <- train(iris.train_dataset[, 1:4], iris.train_dataset[, 5], method='knn')

# doing the classification step we mentioned above with only one line of code!
classifications<-predict(object=model_knn,iris.test_dataset[,1:4])

# Evaluate the classifications
table(classifications)
```

```
## classifications
##     Iris-setosa Iris-versicolor  Iris-virginica
##              12              13              11
```

```r
# Confusion matrix that show the classification we made against the real label
confusionMatrix(classifications,iris.test_dataset[,5])
```

```
## Confusion Matrix and Statistics
## 
##                  Reference
## Prediction        Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              12               0              0
##   Iris-versicolor           0              12              1
##   Iris-virginica            0               0             11
## 
## Overall Statistics
## 
##                Accuracy : 0.9722          
##                  95% CI : (0.8547, 0.9993)
##     No Information Rate : 0.3333          
##     P-Value [Acc > NIR] : 4.864e-16       
## 
##                   Kappa : 0.9583          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: Iris-setosa Class: Iris-versicolor
## Sensitivity                      1.0000                 1.0000
## Specificity                      1.0000                 0.9583
## Pos Pred Value                   1.0000                 0.9231
## Neg Pred Value                   1.0000                 1.0000
## Prevalence                       0.3333                 0.3333
## Detection Rate                   0.3333                 0.3333
## Detection Prevalence             0.3333                 0.3611
## Balanced Accuracy                1.0000                 0.9792
##                      Class: Iris-virginica
## Sensitivity                         0.9167
## Specificity                         1.0000
## Pos Pred Value                      1.0000
## Neg Pred Value                      0.9600
## Prevalence                          0.3333
## Detection Rate                      0.3056
## Detection Prevalence                0.3056
## Balanced Accuracy                   0.9583
```

Wow we achieve an accuracy of around 95% with just few lines of code! Try it yourselves with different dataset you can find online!

Conclusions

Above, we discussed a interesting machine learning algorthm KNN that help us classify discrete sample data points. Hopefully, you have a deeper understanding of what machine learning is like. Machine Learning, indeed, can be complex and difficult to learn but as a genereal person, we can easily utilize the benefit that machine learning algorithm by just calling the api(or functions) defined in useful package like caret, scikit-learn and tensorflow which abstracted the intimidating machine learning knowledge from you. Now what we need to do is learn what algorithm is, know our dataset well and integrate the easy api! From above example, we see that we can easily do so with few lines of code! Impressive!
In fact, KNN is more powerful than what I just introduced to you. It can also classify continuous dataset as well. You can explore more about this by clicking the link below:
Nearest Neighbor Classification From CS189

Take-home message
1. Machine Learning can be easy. It is just learn and predict!
2. Machine Learning methods can be easily implement by using packages like caret!
3. Only 3 steps for us to master machine learning: know you exitsting dataset well, find mechine learning algorithm that good for your case, and finally learn the function call of the package of your choice!

References

Nearest Neighbor Classification From CS189
Best way to learn kNN Algorithm using R Programming
Machine Learning in R for beginners
Iris Data Source
Caret
Caret Doc
display image
3d Plot