Laboratory report of Datamining

K-means

Clustering is a kind of unsupervised learning which consists in a dataset partitioning with the goal of maximizing intra cluster similarity and minimizing inter cluster similarity.

K-means is a clustering algorithm and consists in the following steps [1]:

1. Decide on a value for k (k is the number of clusters).
2. Initialize the k cluster centers (randomly, if necessary).
3. Decide the class membership of the N object by assigning them to the nearest cluster center.
4. Re-estimate the k cluster centers, by assuming the memberships found above are correct.
5. If none of the N object changed membership in the last iteration, exit. Otherwise, goto 3.

Here is the code of the kmeans:

```
#k-means on the first 4 columns (without the classes)
#we choose an high number of max iterations
#and we reinitialize the algorithms 20 times, to try different
#random center sets to avoid local minima
#we only use "cluster", which is a vector containing the cluster of each point
cluster <- kmeans(iris[, 1:4], 3, 100, 20)$cluster
classes <- iris[, 5]
#we compare the target classes to the clusters found by the algorithm
t <- table(cluster, t(classes))
```

The table is the following

```
cluster setosa versicolor virginica
      1     50          0         0
      2      0         48        14
      3      0          2        36
```

Then we evaluate the clustering using a measure called *purity*.

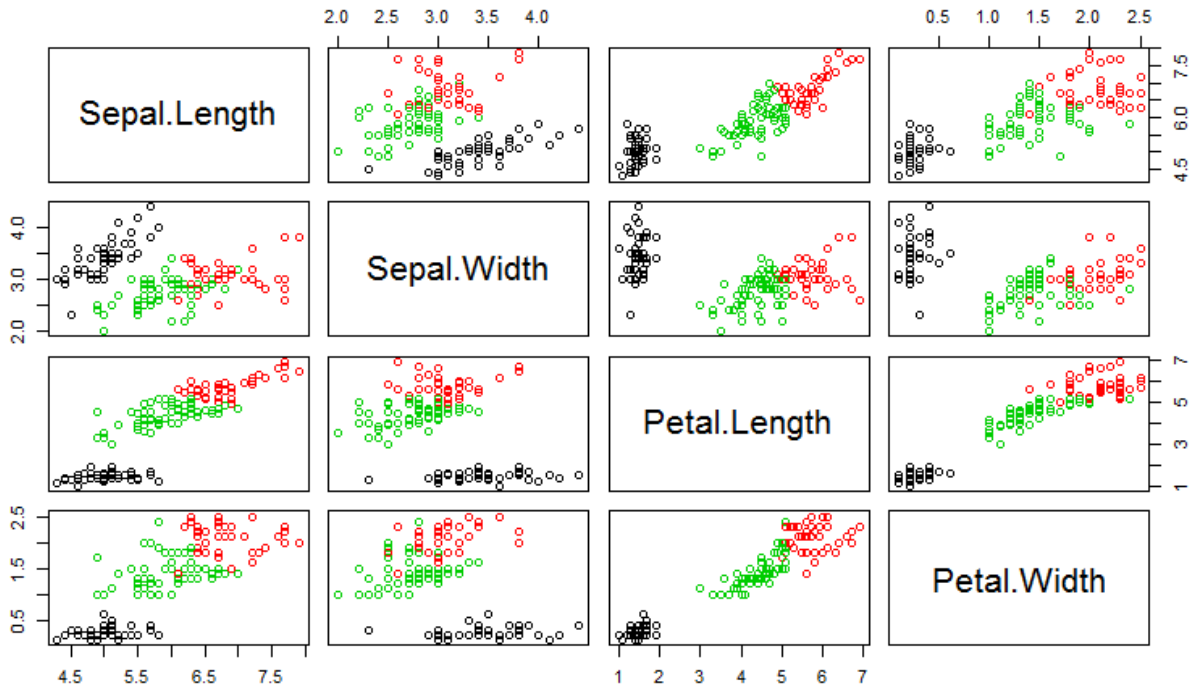
To compute purity, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned examples and dividing by N [2].

```
purity <- 0
for (nr in 1:nrow(t)) {
  purity <- purity + max(t[nr,])
}
purity <- purity / nrow(iris)
```

The resulting purity of the clustering is 0.893.

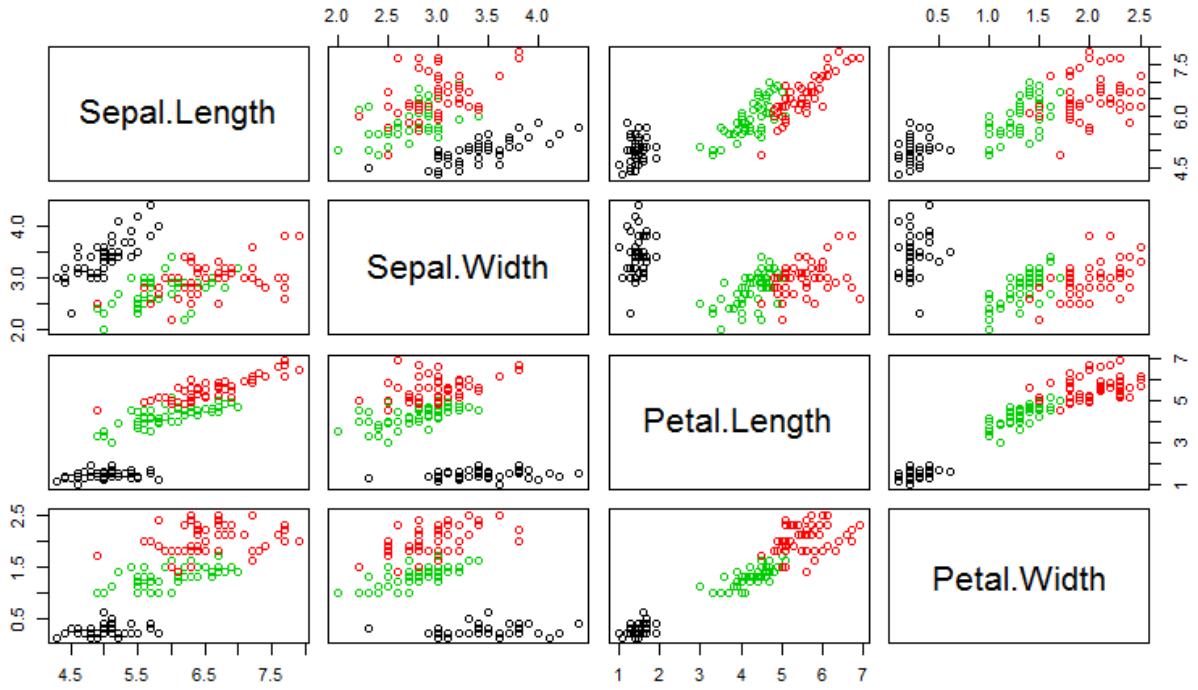
Now we plot the result with the command

```
plot(iris[, 1:4], col = cluster)
```



We compare this graph with the dataset target classes:

```
plot(iris[, 1:4], col = classes)
```



The result of the clustering is acceptable, but virginica and versicolor are difficult to separate, as we can see in the graphs.

We notice in the above graphs that some attributes seem to be more useful than others in dividing the dataset. So, now we perform the clustering only on subsets of the attributes, trying to maximize the purity.

```
#combinations represents the strings of the attribute combination
combinations <- character(0)
purities <- numeric(0)
for (num_attr in 1:4) {
  #comb is a vector containing all the n-attributes combinations
  comb <- combn(1:4, num_attr)
  for (i in 1:ncol(comb)) {

    cluster <- kmeans(iris[,comb[, i]], 3, 20, 20)$cluster
    classes <- iris[, 5]
    t <- table(cluster, t(classes))

    purity <- 0
    for (nr in 1:nrow(t)) {
      purity <- purity + max(t[nr,])
    }
    purity <- purity / nrow(iris);

    purities <- append(purities, purity)
    combinations <- append(combinations, paste(comb[, i], collapse=" "))
  }
}
results <- cbind(combinations, purities)
```

These are the results:

	combinations	purities
4	4	0.96
10	3 4	0.96
14	2 3 4	0.953333333333333
3	3	0.946666666666667
8	2 3	0.926666666666667
9	2 4	0.926666666666667
13	1 3 4	0.893333333333333
15	1 2 3 4	0.893333333333333
6	1 3	0.88
11	1 2 3	0.88
12	1 2 4	0.826666666666667
5	1 2	0.82
7	1 4	0.813333333333333
1	1	0.666666666666667
2	2	0.56

As we can see, we can have the highest purities using only the 4th attribute, the Petal Witdh.

K-Nearest-Neighbour

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

We will use Euclidean distance. [3]

Here is the code. We tried different values for k; for little values the accuracy didn't change much, while it decreased for big values (after 10). So we chose 3.

```
#we will try different sizes of the training set, from 75 to 145
train_sizes <- rep(c(75), 15) + (0:14)*5
accuracies <- numeric(0)

for (train_size in train_sizes) {
  average_acc <- 0
  #we execute the algorithms many times to get an average accuracy
  for (iteration in 1:10000) {
    #we choose 100 random rows from the dataset
    #we will use these rows as training set
    #and the other 50 as test set
    train_indexes <- sample(150, train_size)
    train <- iris[train_indexes, ]
    test <- iris[-train_indexes, ]
    cl <- factor(t(train[, 5]))
    prediction <- knn(train[, 1:4], test[, 1:4], cl, k = 3)
    corrects <- prediction == t(test[, 5])
    #we count the number of correct results and divide it by N
    accuracy <- length(which(corrects))/nrow(test)
    average_acc <- average_acc + accuracy
  }
  average_acc <- average_acc/10000
  accuracies <- append(accuracies, average_acc)
}
results <- cbind(train_sizes, accuracies)
```

The results are the following:

	train_sizes	accuracies
15	145	0.9610200
12	130	0.9609800
14	140	0.9606300
11	125	0.9605320
9	115	0.9604429
8	110	0.9604150
10	120	0.9601567
7	105	0.9599467
13	135	0.9598333
6	100	0.9591380
5	95	0.9587727
4	90	0.9586250
3	85	0.9574492
2	80	0.9570986
1	75	0.9567520

It works better with a high number of examples in the training set, as we can see in the table.

The algorithms has always a high accuracy, and it fits the problem better than k-means.

Bibliography

[1] S. Calderara, *Unsupervised Learning,* Simone Calderara.

[2] P. R. a. H. S. Christopher D. Manning, «Evaluation of clustering,» 07 04 2009. [Online]. Available: http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html.

[3] S. Sayad, «K Nearest Neighbors - Classification,» [Online]. Available: http://www.saedsayad.com/k_nearest_neighbors.htm.