Homework #6

CS5402 — Intro To Data Mining

Illya Starikov

Due Date: July 27th, 2018

1 Support Vectors

```
library(kernlab)

xy = matrix(c(1, 2, 4, 1, 1, 5, 2, 1, 2, 0, 1, 2), nrow=6, ncol=2)
z =  matrix(c(1, 1, -1, 1, 1, -1), nrow=6, ncol=1)

svp = ksvm(xy, z, type="C-svc", kernel='vanilladot', C=100, scaled=c())

print(xmatrix(svp))
print(predict(svp, matrix(c(0, 1), nrow=1, ncol=2)))
print(predict(svp, matrix(c(4, 1), nrow=1, ncol=2)))
```

The output is as follows:

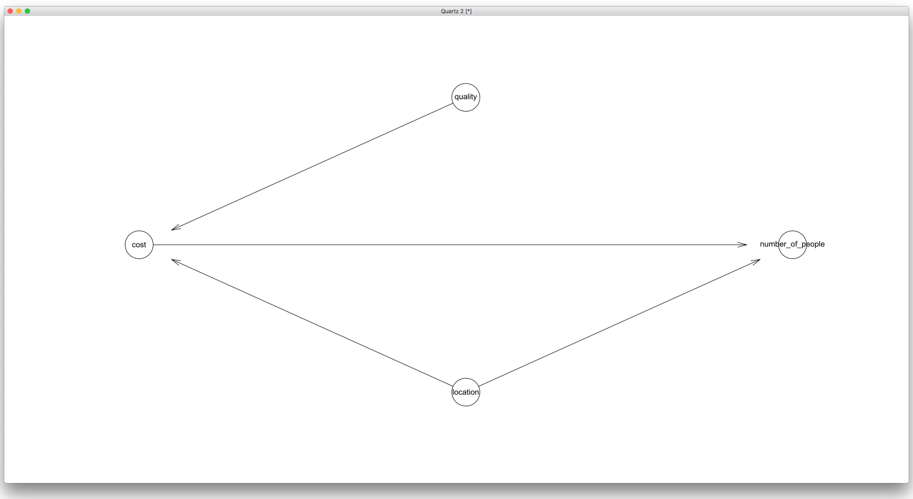
```
 Setting default kernel parameters
[[1]]
  X1 X2
2  2  1
3  4  2

[1] 1
[1] -1
```

2 Bayes Network

```
library(bnlearn)
dag = model2network("[location][cost|location:quality][quality][number_of_
    people|location:cost]")

location.values         = factor(c("Good", "Bad"))
quality.values          = factor(c("Good", "Normal", "Bad"))
```

```
cost.values              = factor(c("High", "Low"))
number_of_people.values = factor(c("High", "Low"))

plot(dag)

location.prob          = array(c(0.6, 0.4), dim=2, dimnames=list(location=
    location.values))
quality.prob           = array(c(0.3, 0.5,0.2), dim=3, dimnames=list(
    quality=quality.values))
cost.prob              = array(c(0.8, 0.2, 0.6, 0.4, 0.1, 0.9, 0.6, 0.4,
    0.6, 0.4, 0.05, 0.95), dim=c(2,3,2), dimnames=list(cost=cost.values,
    quality=quality.values, location=location.values))
number_of_people.prob = array(c(0.6, 0.4, 0.8, 0.2, 0.1, 0.9, 0.6, 0.4),
    dim=c(2, 2, 2), dimnames=list(number_of_people=number_of_people.values,
     cost=cost.values, location=location.values))

conditional_probability_table = list(location=location.prob, quality=
    quality.prob, cost=cost.prob, number_of_people=number_of_people.prob)
print(conditional_probability_table)

bayes_network = custom.fit(dag, conditional_probability_table)

location          = factor(c("Good", "Bad"))
quality           = factor(c("Normal", "Good"))
ccost             = factor(c("High", "Low"))
number_of_people  = factor(c("Low", "High"))

d_test        = data.frame(location, quality, ccost, number_of_people)
names(d_test) = c("location", "quality", "cost", "number_of_people")
prediction    = predict(bayes_network, "quality", d_test, debug=FALSE)

table(prediction, d_test[,"quality"])
print(prediction)
```

The network looks as follows:



The output looks as follows:

```
$location
location
Good  Bad
 0.6  0.4

$quality
quality
  Good Normal    Bad
   0.3    0.5    0.2

$cost
, , location = Good

      quality
cost   Good Normal Bad
  High  0.8    0.6 0.1
  Low   0.2    0.4 0.9

, , location = Bad

      quality
cost   Good Normal  Bad
  High  0.6    0.6 0.05
  Low   0.4    0.4 0.95


$number_of_people
, , location = Good

                cost
number_of_people High Low
            High  0.6 0.8
            Low   0.4 0.2

, , location = Bad

                cost
number_of_people High Low
            High  0.1 0.6
            Low   0.9 0.4
```

```
prediction Good Normal
    Good      0      0
    Normal    1      1
    Bad       0      0
[1] Normal Normal
Levels: Good Normal Bad
```

3 DBScan

```
using RDatasets, Clustering, Distances, Gadfly

cars = dataset("datasets", "mtcars")
x = convert(Array, cars[:,3])'
y = convert(Array, cars[:,11])'
distances = pairwise(Euclidean(), x, y)

cluster = dbscan(distances, 2, 5)

assignments = assignments(cluster)
cluster_plot = plot(x=x, y=y, color=assignments, Geom.point);

print(cluster)
```

```
Clustering.DbscanResult(
    [1, 5, 7, 12, 13, 14, 15, 16, 17, 22, 23, 24, 25, 29, 31],
    [1, 1, 1, 1, 2, 1, 3, 1, 1, 1, 1, 4, 5, 6, 7, 8, 9, 1, 1, 1,
        1, 10, 11, 12, 13, 1, 1, 1, 14, 1, 15, 1],
    [18, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Gadfly could not produce a graph.

4 Decision Tree

```
using RDatasets, DecisionTree, DataFrames

cars = dataset("datasets", "mtcars")

features = convert(Array, cars[:, [12, 3, 5, 11]])
classification = convert(Array, cars[:, 2])
model = build_tree(classification, features)

print_tree(model)
```

```
Feature 3, Threshold 118.0
L-> Feature 3, Threshold 96.0
    L-> Feature 3, Threshold 65.5
        L-> Feature 1, Threshold 1.5
            L-> 33.9 : 1/1
            R-> Feature 3, Threshold 57.0
                L-> 30.4 : 1/1
                R-> 24.4 : 1/1
        R-> Feature 3, Threshold 92.0
            L-> Feature 1, Threshold 1.5
                L-> 27.3 : 1/2
                R-> 26.0 : 1/1
            R-> 22.8 : 2/2
    R-> Feature 1, Threshold 3.0
        L-> Feature 3, Threshold 107.0
            L-> Feature 2, Threshold 5.0
                L-> 21.5 : 1/1
                R-> 18.1 : 1/1
            R-> Feature 4, Threshold 4.5
                L-> 21.4 : 2/2
                R-> 30.4 : 1/1
        R-> 21.0 : 2/2
R-> Feature 3, Threshold 192.5
    L-> Feature 2, Threshold 7.0
        L-> Feature 4, Threshold 4.5
            L-> 17.8 : 1/2
            R-> 19.7 : 1/1
        R-> Feature 1, Threshold 2.5
            L-> Feature 3, Threshold 162.5
                L-> 15.2 : 1/2
                R-> 19.2 : 1/2
            R-> 16.4 : 1/3
    R-> Feature 3, Threshold 237.5
        L-> Feature 3, Threshold 222.5
            L-> 10.4 : 2/2
            R-> 14.7 : 1/1
        R-> Feature 3, Threshold 254.5
            L-> 14.3 : 1/2
            R-> Feature 3, Threshold 299.5
                L-> 15.8 : 1/1
                R-> 15.0 : 1/1
```

5 KMeans

```
using CSV, DataFrames
using Clustering
using Gadfly

data_frame = CSV.read("./problem-5.csv"; types=[Float64, Float64, Float64])
data = convert(Array, data_frame[:, [1, 2]])'

k_means = kmeans(data, 3; maxiter=200, display=:iter)

a = assignments(k_means)
c = counts(k_means)

print("Assignments: ")
println(a')

print("Counts: ")
println(c')

plot(x=data[1, :], y = data=[2, :], color=a, Geom.point)
```

```
  Iters               objv        objv-change | affected
-------------------------------------------------------------
      0       1.000000e+02
      1       4.880000e+01       -5.120000e+01 |        0
      2       4.880000e+01        0.000000e+00 |        0
K-means converged with 2 iterations (objv = 48.80000000000001)
Assignments: [1 1 1 1 2 1 3]
Counts: [5 1 1]
```