Assign5-1\_Support Vector Machines

Jonathan De Los Santos

2/24/2021

# Data Preparation

This example uses German credit customer data. To apply a Support Vector Machine model, we need to identify features that need to be converted to factors or normalized then perform the necessary transformations.

credit <- read.csv("Data Sets/5.0-GermanCredit.csv")  
str(credit)

## 'data.frame': 1000 obs. of 21 variables:  
## $ credit.rating : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ account.balance : int 1 1 2 1 1 1 1 1 3 2 ...  
## $ credit.duration.months : int 18 9 12 12 12 10 8 6 18 24 ...  
## $ previous.credit.payment.status: int 3 3 2 3 3 3 3 3 3 2 ...  
## $ credit.purpose : int 2 4 4 4 4 4 4 4 3 3 ...  
## $ credit.amount : int 1049 2799 841 2122 2171 2241 3398 1361 1098 3758 ...  
## $ savings : int 1 1 2 1 1 1 1 1 1 3 ...  
## $ employment.duration : int 1 2 3 2 2 1 3 1 1 1 ...  
## $ installment.rate : int 4 2 2 3 4 1 1 2 4 1 ...  
## $ marital.status : int 1 3 1 3 3 3 3 3 1 1 ...  
## $ guarantor : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ residence.duration : int 4 2 4 2 4 3 4 4 4 4 ...  
## $ current.assets : int 2 1 1 1 2 1 1 1 3 4 ...  
## $ age : int 21 36 23 39 38 48 39 40 65 23 ...  
## $ other.credits : int 2 2 2 2 1 2 2 2 2 2 ...  
## $ apartment.type : int 1 1 1 1 2 1 2 2 2 1 ...  
## $ bank.credits : int 1 2 1 2 2 2 2 1 2 1 ...  
## $ occupation : int 3 3 2 2 2 2 2 2 1 1 ...  
## $ dependents : int 1 2 1 2 1 2 1 2 1 1 ...  
## $ telephone : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ foreign.worker : int 1 1 1 2 2 2 2 2 1 1 ...

#head(credit, 20)

## Data Prep: Factor Transformations

The following variables need to be converted to factors:

factors <- credit[,c(1,2,4,5,7:13,15:21)]  
str(factors)

## 'data.frame': 1000 obs. of 18 variables:  
## $ credit.rating : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ account.balance : int 1 1 2 1 1 1 1 1 3 2 ...  
## $ previous.credit.payment.status: int 3 3 2 3 3 3 3 3 3 2 ...  
## $ credit.purpose : int 2 4 4 4 4 4 4 4 3 3 ...  
## $ savings : int 1 1 2 1 1 1 1 1 1 3 ...  
## $ employment.duration : int 1 2 3 2 2 1 3 1 1 1 ...  
## $ installment.rate : int 4 2 2 3 4 1 1 2 4 1 ...  
## $ marital.status : int 1 3 1 3 3 3 3 3 1 1 ...  
## $ guarantor : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ residence.duration : int 4 2 4 2 4 3 4 4 4 4 ...  
## $ current.assets : int 2 1 1 1 2 1 1 1 3 4 ...  
## $ other.credits : int 2 2 2 2 1 2 2 2 2 2 ...  
## $ apartment.type : int 1 1 1 1 2 1 2 2 2 1 ...  
## $ bank.credits : int 1 2 1 2 2 2 2 1 2 1 ...  
## $ occupation : int 3 3 2 2 2 2 2 2 1 1 ...  
## $ dependents : int 1 2 1 2 1 2 1 2 1 1 ...  
## $ telephone : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ foreign.worker : int 1 1 1 2 2 2 2 2 1 1 ...

### Factor Transformation Function

Let’s create a function that we can apply to the variables above:

to.Factor <- function (df, variables){  
 for (i in variables){  
 df[[i]] <- as.factor(df[[i]])  
 }  
 return(df)  
}

Now we will duplicate our original data set, pass our factors dataframe into the above function, and store it in the new dataframe

# Duplicate data for transformation  
t.credit <- credit  
  
var.factors <- c("credit.rating",  
 "account.balance",  
 "previous.credit.payment.status",  
 "credit.purpose",  
 "savings",  
 "employment.duration",  
 "installment.rate",  
 "marital.status",  
 "guarantor",  
 "residence.duration",  
 "current.assets",  
 "other.credits",  
 "apartment.type",  
 "bank.credits",  
 "occupation",  
 "dependents",  
 "telephone",  
 "foreign.worker"  
 )  
  
t.credit <- to.Factor(t.credit, var.factors)  
  
#str(t.credit)

# A

### Normalization

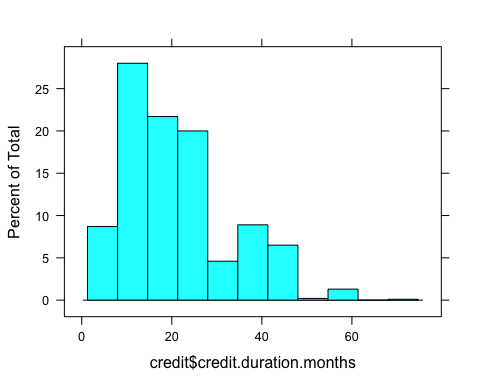
First we will test the three remaining integer variables for normalization, then apply a normalization function to skewed distributions. In this case, we see that all three variables are skewed and require normalization.

#install.packages("lattice")  
library(lattice)

## Warning: package 'lattice' was built under R version 4.0.2

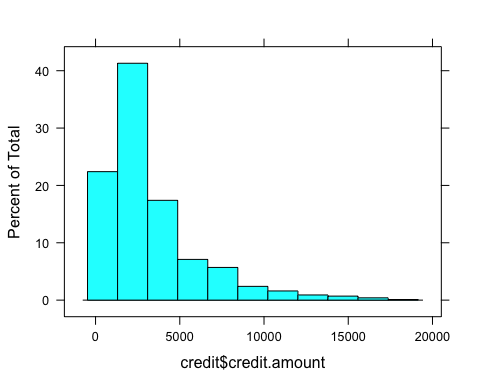
histogram(credit$credit.duration.months, credit)

## Warning in histogram.numeric(credit$credit.duration.months, credit): explicit  
## 'data' specification ignored



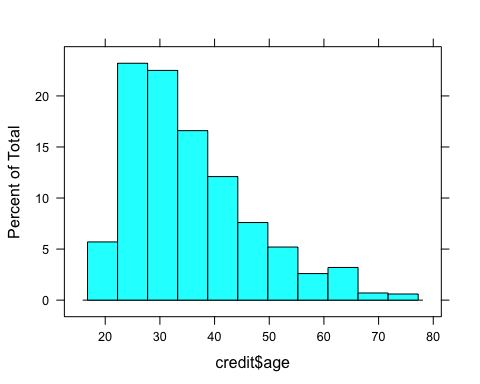
histogram(credit$credit.amount, credit)

## Warning in histogram.numeric(credit$credit.amount, credit): explicit 'data'  
## specification ignored



histogram(credit$age, credit)

## Warning in histogram.numeric(credit$age, credit): explicit 'data' specification  
## ignored

 #### Normalization Function

scale.variable <- function(df, variables){  
 for (i in variables){  
 df[[i]] <- scale(df[[i]], center = T, scale = T)  
 }  
 return(df)  
}  
  
# Apply normalization function to integers  
var.normalize <- c("credit.duration.months",  
 "credit.amount",  
 "age")  
  
t.credit <- scale.variable(t.credit, var.normalize)  
#str(t.credit)

# B

## Confirm Transformations

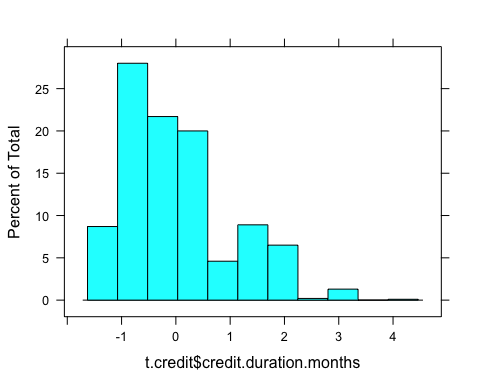
Let’s double check that factors have been converted and the remaining integers have been normalized. Unfortunately in this case our curves have only shifted slightly.

str(t.credit)

## 'data.frame': 1000 obs. of 21 variables:  
## $ credit.rating : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ account.balance : Factor w/ 3 levels "1","2","3": 1 1 2 1 1 1 1 1 3 2 ...  
## $ credit.duration.months : num [1:1000, 1] -0.241 -0.987 -0.738 -0.738 -0.738 ...  
## ..- attr(\*, "scaled:center")= num 20.9  
## ..- attr(\*, "scaled:scale")= num 12.1  
## $ previous.credit.payment.status: Factor w/ 3 levels "1","2","3": 3 3 2 3 3 3 3 3 3 2 ...  
## $ credit.purpose : Factor w/ 4 levels "1","2","3","4": 2 4 4 4 4 4 4 4 3 3 ...  
## $ credit.amount : num [1:1000, 1] -0.787 -0.167 -0.861 -0.407 -0.39 ...  
## ..- attr(\*, "scaled:center")= num 3271  
## ..- attr(\*, "scaled:scale")= num 2823  
## $ savings : Factor w/ 4 levels "1","2","3","4": 1 1 2 1 1 1 1 1 1 3 ...  
## $ employment.duration : Factor w/ 4 levels "1","2","3","4": 1 2 3 2 2 1 3 1 1 1 ...  
## $ installment.rate : Factor w/ 4 levels "1","2","3","4": 4 2 2 3 4 1 1 2 4 1 ...  
## $ marital.status : Factor w/ 3 levels "1","3","4": 1 2 1 2 2 2 2 2 1 1 ...  
## $ guarantor : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ residence.duration : Factor w/ 4 levels "1","2","3","4": 4 2 4 2 4 3 4 4 4 4 ...  
## $ current.assets : Factor w/ 4 levels "1","2","3","4": 2 1 1 1 2 1 1 1 3 4 ...  
## $ age : num [1:1000, 1] -1.2809 0.0403 -1.1048 0.3046 0.2165 ...  
## ..- attr(\*, "scaled:center")= num 35.5  
## ..- attr(\*, "scaled:scale")= num 11.4  
## $ other.credits : Factor w/ 2 levels "1","2": 2 2 2 2 1 2 2 2 2 2 ...  
## $ apartment.type : Factor w/ 3 levels "1","2","3": 1 1 1 1 2 1 2 2 2 1 ...  
## $ bank.credits : Factor w/ 2 levels "1","2": 1 2 1 2 2 2 2 1 2 1 ...  
## $ occupation : Factor w/ 4 levels "1","2","3","4": 3 3 2 2 2 2 2 2 1 1 ...  
## $ dependents : Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 1 ...  
## $ telephone : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ foreign.worker : Factor w/ 2 levels "1","2": 1 1 1 2 2 2 2 2 1 1 ...

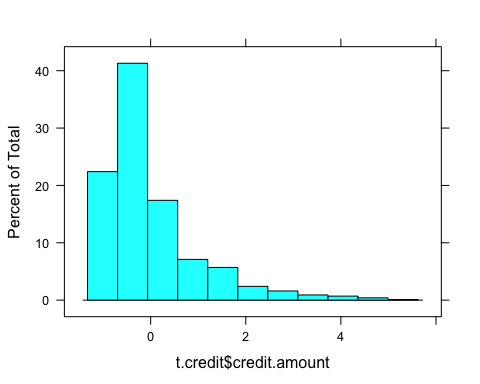
histogram(t.credit$credit.duration.months, t.credit)

## Warning in histogram.numeric(t.credit$credit.duration.months, t.credit):  
## explicit 'data' specification ignored



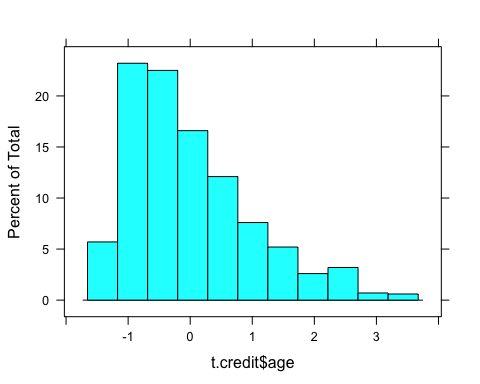
histogram(t.credit$credit.amount, t.credit)

## Warning in histogram.numeric(t.credit$credit.amount, t.credit): explicit 'data'  
## specification ignored



histogram(t.credit$age, t.credit)

## Warning in histogram.numeric(t.credit$age, t.credit): explicit 'data'  
## specification ignored



# C

## Data Partition

library(caret)

## Warning: package 'caret' was built under R version 4.0.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.2

cpartition <- createDataPartition(y = t.credit$credit.rating, p = 0.6, list = FALSE)  
credit.train <- t.credit[cpartition,]  
credit.test <- t.credit[-cpartition,]   
  
str(credit.train)

## 'data.frame': 600 obs. of 21 variables:  
## $ credit.rating : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ account.balance : Factor w/ 3 levels "1","2","3": 1 2 1 1 3 1 1 1 1 2 ...  
## $ credit.duration.months : num [1:600, 1] -0.241 -0.738 -0.738 -1.236 -0.241 ...  
## $ previous.credit.payment.status: Factor w/ 3 levels "1","2","3": 3 2 3 3 3 3 2 2 3 3 ...  
## $ credit.purpose : Factor w/ 4 levels "1","2","3","4": 2 4 4 4 3 1 3 3 4 3 ...  
## $ credit.amount : num [1:600, 1] -0.787 -0.861 -0.407 -0.677 -0.77 ...  
## $ savings : Factor w/ 4 levels "1","2","3","4": 1 2 1 1 1 2 4 3 1 1 ...  
## $ employment.duration : Factor w/ 4 levels "1","2","3","4": 1 3 2 1 1 3 3 2 2 4 ...  
## $ installment.rate : Factor w/ 4 levels "1","2","3","4": 4 2 3 2 4 1 2 2 1 4 ...  
## $ marital.status : Factor w/ 3 levels "1","3","4": 1 1 2 2 1 3 3 2 2 2 ...  
## $ guarantor : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ residence.duration : Factor w/ 4 levels "1","2","3","4": 4 4 2 4 4 4 4 3 2 4 ...  
## $ current.assets : Factor w/ 4 levels "1","2","3","4": 2 1 1 1 3 3 3 1 1 1 ...  
## $ age : num [1:600, 1] -1.281 -1.105 0.305 0.393 2.595 ...  
## $ other.credits : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...  
## $ apartment.type : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 1 1 1 2 2 ...  
## $ bank.credits : Factor w/ 2 levels "1","2": 1 1 2 1 2 2 2 1 2 1 ...  
## $ occupation : Factor w/ 4 levels "1","2","3","4": 3 2 2 2 1 3 2 3 2 3 ...  
## $ dependents : Factor w/ 2 levels "1","2": 1 1 2 2 1 1 1 2 2 1 ...  
## $ telephone : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...  
## $ foreign.worker : Factor w/ 2 levels "1","2": 1 1 2 2 1 1 1 1 1 1 ...

## Build svm() Model

library(e1071)  
  
svm1 <- svm(formula = credit.rating~., data = credit.train)  
  
summary(svm1)

##   
## Call:  
## svm(formula = credit.rating ~ ., data = credit.train)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
##   
## Number of Support Vectors: 380  
##   
## ( 200 180 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

## e1071 SVM Evaluation

The predicted model evaluation of svm() with the default costs and gamma is displayed below. It has a success rate of **70.5%,** and performed particularly poorly on frequency of false positives.

pred <- predict (svm1, credit.test)  
table(pred, credit.test$credit.rating)

##   
## pred 0 1  
## 0 29 17  
## 1 91 263

agreement <- pred == credit.test$credit.rating  
table(agreement)

## agreement  
## FALSE TRUE   
## 108 292

prop.table(table(agreement))

## agreement  
## FALSE TRUE   
## 0.27 0.73

# D

## Linear Kernel

### Build Linear Kernel

library(kernlab)

## Warning: package 'kernlab' was built under R version 4.0.2

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':  
##   
## alpha

vSVM <- ksvm (credit.rating~., data = credit.train,  
 kernel = "vanilladot")

## Setting default kernel parameters

# Print output  
vSVM

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 314   
##   
## Objective Function Value : -289.8048   
## Training error : 0.206667

### Evaluate Linear Kernel

The predicted model evaluation of ksvm() with a linear kernel (vanilladot)is displayed below. It has a success rate of **73.75%.**

myPredictions <- predict(vSVM, credit.test)  
head(myPredictions)

## [1] 0 1 1 1 1 0  
## Levels: 0 1

table(myPredictions, credit.test$credit.rating)

##   
## myPredictions 0 1  
## 0 55 38  
## 1 65 242

agreement <- myPredictions == credit.test$credit.rating  
table(agreement)

## agreement  
## FALSE TRUE   
## 103 297

prop.table(table(agreement))

## agreement  
## FALSE TRUE   
## 0.2575 0.7425

## Non-Linear Kernel

### Build Gaussian Kernel

# RBFdot SVM  
rbfSVM <- ksvm(credit.rating~., data = credit.train, kernel = "rbfdot")  
rbfSVM

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.0648794733388866   
##   
## Number of Support Vectors : 393   
##   
## Objective Function Value : -287.4632   
## Training error : 0.168333

### Evaluate Gaussian Kernel

The predicted model evaluation of ksvm() with the Gaussian RBF kernel (rbfdot)is displayed below. It has a success rate of **73%** and also erred towards false positives.

rbfPredictions <- predict(rbfSVM, credit.test)  
head(rbfPredictions)

## [1] 1 1 1 1 1 1  
## Levels: 0 1

table(rbfPredictions, credit.test$credit.rating)

##   
## rbfPredictions 0 1  
## 0 42 27  
## 1 78 253

rbf\_agreement <- rbfPredictions == credit.test$credit.rating  
table(rbf\_agreement)

## rbf\_agreement  
## FALSE TRUE   
## 105 295

prop.table(table(rbf\_agreement))

## rbf\_agreement  
## FALSE TRUE   
## 0.2625 0.7375

# Conclusion

The linear kernel SVM performed the best out of the models performed with a success rate of 73.75%, only slightly better than the non-linear kernel at 73%. Based on the higher success rate and lower computational demands, I would recommend usage of this model.