Assigment1

Pavel Linder, Nikita Brancatisano

11/6/2019

Installing packages

```
install.packages("rgl")
install.packages("caret")
install.packages("generics")
install.packages("plot3D")
install.packages("gower")

library(caret)
library("rgl")
library("plot3D")
library("GA")

library("dplyr")
```

What does it mean for a function to be non-convex?

A function is non-convex (or concave) if it is a continuous function whose value at the midpoint of every interval in its domain does not exceed the arithmetic mean of its values at the ends of the interval.

Task 1: Maximization of a non-convex function.

1. Code a method f(x, y) that computes a value z, given an input tuple (x, y).

```
f <- function(x, y)  {
  z <- ((1 - x)^2) + (exp(1) * (y - (x^2))^2)
}
```

2. Code a method that visualizes the Rosenbrock function in 3D

```
x <- y <- seq(-1, 1, length= 20)
z <- outer(x, y, f)
z[is.na(z)] <- 1 # change non-defined elements to 1
persp(x, y, z, theta = 30, phi = 20, expand = 1, col = "lightblue", ticktype = "detailed")
```

3. Code a genetic algorithm, that attempts to find the global maximum of this function.

```
crossovers = c("ga_spCrossover",
              "gabin_spCrossover", "gabin_uCrossover",
              "gareal_spCrossover", "gareal_waCrossover", "gareal_laCrossover", "gareal_blxCrossover",
              "gaperm_cxCrossover", "gaperm_pmxCrossover", "gaperm_pmxCrossover"
        )
results = c()
best <- 0
```

```
name <- 0
for (i in 1:length(crossovers)) {
  GA <- ga(type = "real-valued",
           fitness =  function(x) f(x[1], x[2]),
           lower = c(-1, -1), upper = c(1, 1),
           popSize = 50, maxiter = 1000, run = 100,
           crossover = crossovers[i]
           )
  if (best < GA@fitness) {
    name <- crossovers[i]
    best[1] <- GA@fitness
  }
  results[i] <- GA@fitness
}

results
best
name

plot(GA)
summary(GA)
res <- persp3D(x = x, y = x, z = z,theta = 30, phi = 25, col.palette = bl2gr.colors)

max_x = GA@solution[1]
max_y = GA@solution[2]
max_z = f(GA@solution[1], GA@solution[2])

points3D(max_z, max_y, max_z, col = "black", size = 30, add=T)

# BONUS: Plot the trace of evolution
history_x = c()
history_y = c()

for (i in 1:12) {
    GA2 <- ga(type = "real-valued",
              fitness =  function(x) f(x[1], x[2]),
              lower = c(-1, -1), upper = c(1, 1),
              popSize = 50, maxiter = 25*i, run = 100)
    history_x[i] <- GA2@solution[,1]
    history_y[i] <- GA2@solution[,2]
}

res2 <- persp3D(x = x, y = y, z = z,theta = 30, phi = 25, col.palette = bl2gr.colors, ticktype = "detail
points3D(x = history_x, y = history_y, z = f(history_x, history_y), col = "black", size = 2, transparenc
text3D(x = history_x[1:3], y = history_y[1:3], z = f(history_x[1:3], history_y[1:3]), labels = 25*c(1:3
text3D(x = history_x[10:12], y = history_y[10:12], z = f(history_x[10:12], history_y[10:12]), labels = 2
```

4. Discuss the results.

How does performance vary when you are increasing the number of iterations?

The performance is slowed down because the GA reaches the maximum solution much before we run all of the iterations, thus

increasing the number would only slow down the search without providing any significant benefit.

What about population size?

Increasing the population size actually speeds up the process but there's a diminishing return on how much we can increase the

population. After a certain amount (depending on the problem) increasing it just slows down the GA.

Explain the difference between local and global maxima.

A local maxima is the biggest element for one given subset of the whole function. There can still be other values greater than

this one, outside ofthe range of the subset. The global maximum is the largest overall value of the function.

#Task 2: Genetic feature selection ## Work plan: ## 1) Read in the data ## 2) Split the data into training and test sets ## 3) Build a model using the training set ## 4) Evaluate the model using the test set

1) Read in the dataset

```
genes <- DLBCL
summary(genes)
target <- genes$class
```

2) Split the data into training and test sets

```
splited <- createDataPartition(
  y = target,
  p = .8,
  list = FALSE
)

str(splited)    # The output is a set of integers for the rows of genes that belong in the training set
learn = genes[ splited, ]
test = genes[-splited, ]

nrow(learn)
nrow(test)
table(learn$class)
table(test$class)
learn$X <- NULL
test$X <- NULL
```

3) Build a model using the training set

```
ctrl <- trainControl(number = 3,
                     method = "repeatedcv")

PLSModel <- train(
  class ~ .,
  data = learn,
  method = "pls",
  preProc = c("center", "scale"),
  trControl = ctrl,
  tuneLenght = 1000
)



RFModel <- train(
  class ~ .,
  data = learn,
  method = "rf",
  preProc = c("center", "scale"),
  trControl = ctrl,
  tuneLenght = 1000
)



xgbModel <- train(
  class ~ .,
  data = learn,
  method = "xgbDART",
  preProc = c("center", "scale"),
  trControl = ctrl,
  tuneLenght = 100
)


SVMModel <- train(
  class ~ .,
  data = learn,
  method = "svmRadial",
  preProc = c("center", "scale"),
  trControl = ctrl,
  tuneLenght = 1000
)

models_compare <- resamples(list(RF=RFModel, XGBDART=xgbModel, PLS=PLSModel, SVM=SVMModel))

summary(models_compare)

varimp <- varImp(RFModel)
imp <- varimp$importance
```

```
plot(varimp)
```

4) Evaluate the model using the test set

```
plsClasses <- predict(xgbModel, newdata = test)
str(plsClasses)

plsProbs <- predict(xgbModel, newdata = test, type = "prob")
head(plsProbs)

confusionMatrix(data = plsClasses, test$class)
```