Exam 1

STAT 950

Emily Robinson

November 5, 2019

Problem 1

Using the baseball data and keeping the total number of candidates examined close to 2,500, change the steepest ascent local search algorithm to employ 2-neighborhoods.

```
baseball <- read.csv("data/baseball.dat", sep="")
```

(a) Include the function in your printed version of the exam.

```
calcAIC <- function(candidate, X, y){
  n   = dim(X)[1]
  Xc  = as.matrix(cbind(rep(1,n), X[,candidate]))
  pc  = dim(Xc)[2]
  fit = lm.fit(Xc,y)
  return(n*log(crossprod(fit$residuals)/n)+2*(pc+1))
}

localSearch <- function(objectiveFn, candidate, nSteps = 20, reStarts = 5, minimize = FA
  p = length(candidate)
  objectiveValues = rep(NA, (nSteps+1)*reStarts)
  finalCandidates = matrix(0,reStarts,p)
  finalObjective  = rep(NA, reStarts)
  it = 0

  for(s in 1:reStarts){
    candidate = (runif(p)<0.5)
    currentObjective = objectiveFn(candidate,...)
    it = it+1
    objectiveValues[it] = NA
    it = it+1
    objectiveValues[it] = currentObjective

    for(i in 1:nSteps){
      bestCandidate = candidate
      bestObjective = currentObjective
      for(j in 1:p){
        for(k in j:p){
```

```
          newCandidate = candidate
          newCandidate[j] = !newCandidate[j]
          if(k == j){newCandidate[k] = newCandidate[k]
        }else{newCandidate[k] = !newCandidate[k]}
          newObjective = objectiveFn(newCandidate,...)

          if(minimize){
            if(newObjective < bestObjective){
              bestObjective = newObjective
              bestCandidate = newCandidate
            }
          } else{ #maximize
            if(newObjective > bestObjective){
              bestObjective = newObjective
              bestCandidate = newCandidate
            }
          }
        }
      }
      currentObjective = bestObjective
      candidate = bestCandidate
      it = it+1
      objectiveValues[it] = currentObjective
    }
    finalCandidates[s,candidate] = 1
    finalObjective[s] = bestObjective
  }
  return(list(it = it,
              finalObjective = finalObjective,
              finalCandidates = finalCandidates,
              objectiveValues = objectiveValues))
}

X <- baseball[,-1]
y <- log(baseball$salary)
candidate <- rep(1,dim(X)[2])
results   <- localSearch(calcAIC, candidate, nSteps = 20, reStarts = 5, minimize = TRUE,
```

(b) If you had R random starts each of S steps searching a neighborhood of size k of the p parameters, then what is the total number candidates examined?

The total number of candidates examined is $R \cdot S \cdot \sum_{i=1}^{k} \binom{p}{i} + R$.

(c) How many restarts and steps did you usesIn total how many candidates did you examine?

With 5 restarts, 20 steps per restart, and 27 parameter selection, using a 2-neighborhood local search, I examined 37805 candidates out of $2^{27} = 134,217,728$ possible candidates. This is far more than 2500, but that would require only 7 restarts and 1 steps per restart, which does not run long enough to stabalize the local search algorithm

```
numCand <- function(R, S, p, k){
  c <- matrix(NA, k, 1)
  for(i in 1:k){
    c[i] <- choose(p,i)
  }
  R*S*sum(c)+R
}
numCand(R = 5, S = 20, p = 27, k = 2)
```

```
## [1] 37805
```

```
numCand(R = 7, S = 1, p = 27, k = 2)
```

```
## [1] 2653
```

(d) What was the smallest AIC you obtained?

The best AIC I obtained was -416.94.

```
# Smallest AIC for each of the five restarts
minAIC <- data.frame("SmallestAIC" = results$finalObjective)
kable(minAIC)
```

SmallestAIC
-415.2714
-414.6119
-414.6030
-414.6030
-414.6030

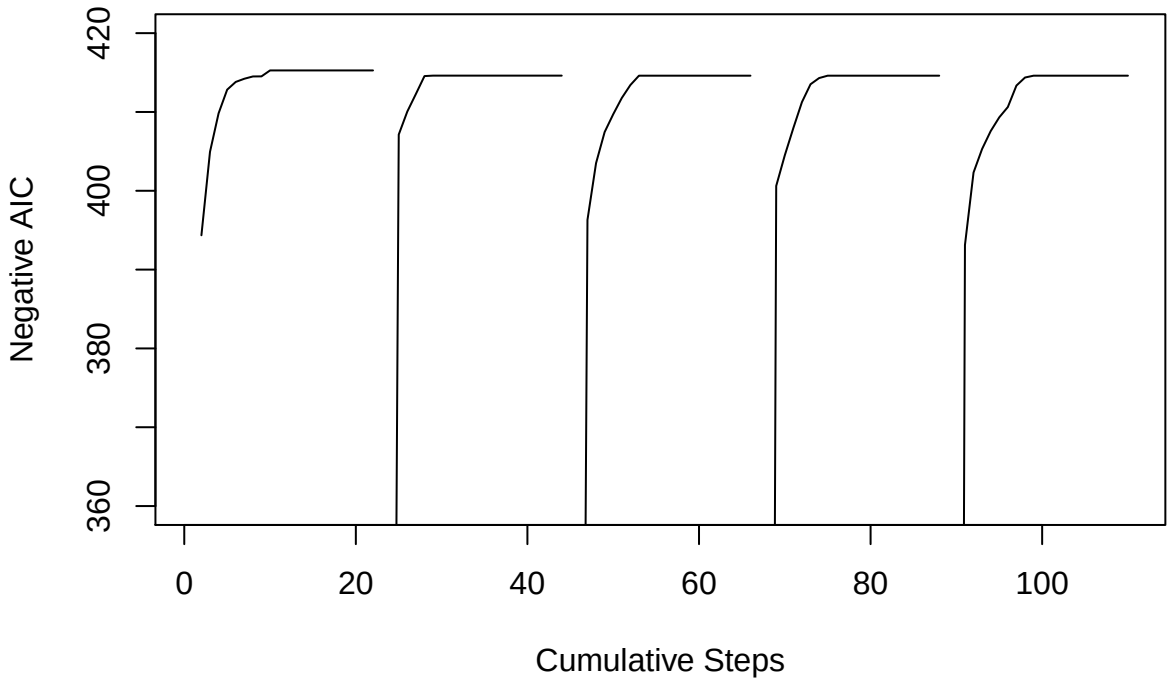
```
# Overall best AIC
bestAIC <- data.frame("BestAIC" = min(minAIC))
kable(bestAIC)
```

BestAIC
-415.2714

(e) Include a pllot similar to the one on page 10 of the Combinatorial Optimization notes.

```
results_data = data.frame(cum_step = seq(1,results$it,1), negAIC = -results$objectiveVal
plot(results_data$cum_step, results_data$negAIC, type = "l",
```

```
    xlab = "Cumulative Steps", ylab = "Negative AIC", ylim = c(360, 420))
```



Problem 2

Modify the genetic search algorithm to search for K clusters which minimize the total within group sum of squares of p characteristics. The total within-group sum of squares is

$$TWGSS = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

where $C_k = \{i : \text{ individual } i \text{ is in cluster } k\}$, x_{ij} is value of characteristic j measure in individual i, $\bar{x}_{kj}$ is the average value of characteristic j in cluster k.

```
wine <- read.csv("data/wine.dat", sep="")
```

(a) Include the function in your printed version of the exam.

Categorizing the wines by the given regions provides a total within group sum of squares of 1292.847.

```
calcTWGSS <- function(candidate, X){
    K = length(unique(candidate))
    p = dim(X)[2]
```

```
    SS <- matrix(NA, K, p)

    for(k in 1:K){
      for(j in 1:p){
        x_ij <- X[candidate == k,j]
        xbar_kj <- mean(x_ij)
        SS[k,j] <- sum((x_ij - xbar_kj)^2)
      }
    }
  return(sum(SS))
}

X <- wine[,-1]
y <- wine[,1]
kable(calcTWGSS(y, X))
```

x
1292.847

```
geneticAlgo <- function(objectiveFn, candidate, G = 99, P = 25, muRate = 0.01, maximum =
  p = length(candidate)
  multiplier = ifelse(maximum, 1, -1)
  candidates = matrix(sample(1:K, P*p, replace = T),P, p)
  offspring = matrix(NA, P, p)
  objectiveValuesMat = matrix(NA, P, G+1)
  bestObjectiveValue = -Inf
  objectiveValuesMat[,1] = apply(candidates, 1, function(c){objectiveFn(c,...)})
  for(i in 1:P){
    if(multiplier*objectiveValuesMat[i,1] > bestObjectiveValue){
      bestCandidate = (1:p)[candidates[i,]]
      bestObjectiveValue = multiplier*objectiveValuesMat[i,1]
    }
  }
  for(g in 1:G){
    fitness = 2*rank(multiplier*objectiveValuesMat[,g])/(P*(P+1))
    for(i in 1:P){
      parents = rbind(candidates[sample(P,1,prob = fitness),],
                      candidates[sample(P,1),])
      crossover = runif(p)<0.5
      offspring[i,] = parents[1,]
      offspring[i,crossover] = parents[2,crossover]
      mutations = runif(p) < muRate
      offspring[i,mutations] = sample(1:K, length(mutations[mutations == T]), replace =
```

```
    }
    candidates = offspring
    objectiveValuesMat[,g+1] = apply(candidates,1,function(c){objectiveFn(c,...)})
    for(i in 1:P){
      if(multiplier*objectiveValuesMat[i,g+1] > bestObjectiveValue){
        bestCandidate = (1:p)[candidates[i,]]
        bestObjectiveValue = multiplier*objectiveValuesMat[i,g+1]
      }
    }
  }
  return(list(bestValue = multiplier*bestObjectiveValue, bestCandidate = bestCandidate,
}
```

(b) Use your function to group the wines from problem 3.8 into 2, 3, and 4 clusters based on the 13 chemical characteristics. What were the total within-group sum of squares you found for 2, 3, and 4 clusters?

```
X <- wine[,-1]
candidate <- rep(1,dim(X)[1])
geneticResults <- matrix(NA, 3,2)
for(k in 2:4){
results  <- geneticAlgo(calcTWGSS, candidate, G = 500, P = 200, muRate = 0.005, maximum
geneticResults[k-1,1] <- k
geneticResults[k-1,2] <- results$bestValue
}
colnames(geneticResults) <- c("K", "TWGSS")
kable(geneticResults)
```

K	TWGSS
2	1649.521
3	1287.443
4	1192.885

Problem 3

p be an interpolating polynomial of order m of the function f at nodes $a = x_0 < ... < x_m = b$. That is

$$p(x) = \sum_{i=0}^{m} f(x_i) \prod_{j=0, j\neq i}^{m} \frac{(x - x_j)}{(x_i - x_j)}.$$

(a) Show that

$$f(x) - p(x) = \frac{f^{m+1}(\xi_x)}{(m+1)!} \prod_{i=0}^{m} (x - x_i)$$

for some $\xi_x \in [a, b]$.

Note:

$$F(y) = f(y) - p(y) - \frac{f(x) - p(x)}{\prod_{i=0}^{m}(x - x_i)} \prod_{i=0}^{m}(y - x_i)$$

will be useful.

Recognizing where $F(y) = 0$, we obtain $m + 1$ roots from the nodes (i.e. $y = x_i$) since $f(y) - p(y) = 0$ and $\prod_{i=0}^{m}(y - x_i) = 0$, thus implying $F(y) = 0$. In addition, we obtain 1 root when $y = x$ since

$$F(x) = f(x) - p(x) - \frac{f(x) - p(x)}{\prod_{i=0}^{m}(x - x_i)} \prod_{i=0}^{m}(x - x_i) = [f(x) - p(x)] - [f(x) - p(x)] = 0.$$

Therefore, F has $m + 2$ roots. Taking the first derviative of F, we get

$$F^{(1)}(y) = f^{(1)}(y) - p^{(1)}(y) - \frac{f(x) - p(x)}{\prod_{i=0}^{m}(x - x_j)} \sum_{i=0}^{m} 1 \prod_{j \neq i}(y - x_j).$$

Since $\sum_{i=0}^{m} 1 \prod_{j \neq i}(y - x_j)$ is a polynomial of order m, similar reasoning to above, $F^{(1)}$ has $m + 1$ roots. The table below continues the pattern.

Function	**Number of roots**
F	$m + 2$
$F^{(1)}$	$m + 1$
$F^{(2)}$	m
$F^{(k)}$	$m + 2 - k$
$F^{(m+1)}$	1 which we call ξ

Therefore, taking the $(m+1)^{st}$ derivative of F, since $p(y)$ is of order m, we know $p^{(m+1)}(y) = 0$ and $\prod_{i=0}^{m}(y - x_i)$ is a polynomial of order $m + 1$, we know

$$F^{(m+1)}(\xi) = f^{(m+1)} - 0 - \frac{f(x) - p(x)}{\prod_{i=0}^{m}(x - x_i)}(m + 1)!$$

$$\implies \quad \frac{f(x) - p(x)}{\prod_{i=0}^{m}(x - x_i)}(m + 1)! = f^{(m+1)}(\xi)$$

$$\implies \quad f(x) - p(x) = \frac{f^{(m+1)}(\xi)}{(m + 1)!} \prod_{i=0}^{m}(x - x_i).$$

(b) Use 3a) to show that for $m = 1$

$$\int_a^b f(x) - p(x)dx = \frac{-(b - a)^3 f^{(2)}(\xi)}{12}$$

for some $\xi \in [a, b]$.

Let $m = 1$. Then $a = x_0 < x_1 = b$. Therefore,

$$f(x) - p(x) = \frac{f^{(2)}(\xi_x)}{2}(x - a)(x - b)$$

$$
\begin{aligned}
\int_a^b f(x) - p(x)dx &= \frac{f^{(2)}}{2}\int_a^b (x-a)(x-b)dx \\
&= \frac{f^{(2)}}{2}\int_a^b (x^2 - ax - bx + ab)dx \\
&= \frac{f^{(2)}}{2}\left[\frac{x^3}{3} - \frac{ax^2}{2} - \frac{bx^2}{2} + abx\Big|_a^b\right] \\
&= \frac{f^{(2)}}{2}\left[\frac{2x^3 - 3ax^2 - 3bx^2 + 6abx}{6}\Big|_a^b\right] \\
&= \frac{f^{(2)}}{12}\left[-b^3 + 3ab^2 + a^3 - 3a^2b\right] \\
&= \frac{f^{(2)}}{12}\left[-(b^3 - 3ab^2 - a^3 + 3a^2b)\right] \\
&= \frac{-(b-a)^3 f^{(2)}}{12}.
\end{aligned}
$$