# CSE455/CSE552 – Machine Learning (Spring 2016)

# Homework #3 Report

**Murat ALTUNTAŞ (111044043)**

---

**Part 1:**

Code:

```
l1 <- runif(30, -0.02, 0.02)
biasMax <- 0.02
biasMin <- -0.02
sigmoid <- function(total1){
	total1 <- total1 * (-1)
	return(1/(1+exp(total1)))
}

myNN <- function(data,m,l1) {
	sonuc1 <- c()
	nodes3 <- c()
	nodes2 <- c()
	nodes <- c()
	ilk <- 0
	son <- 0
	for (j in 1:3) {
		end <- 4 * j
		begin <- end-3
		total1 <- 0
		k <- 1
		for (i in begin:end) {
			step <- data[m,k] * l1[i]
			k <- k + 1
			total1 <- step + total1
		}
		nodes[j] <- total1
```

```
        ilk <- begin
        son <- end
        }

        for (j in 1:3) {
                total1 <- 0
                k <- 1
                son <- 3 + son
                ilk <- son-2
                for (i in ilk:son) {
                        step <- sigmoid(nodes[k]+runif(1, biasMin, biasMax)) * l1[i]
                        k <- k + 1
                        total1 <- step + total1
                }
                nodes2[j] <- total1
        }

        for (j in 1:3) {
                total1 <- 0
                k <- 1
                son <- 3 + son
                ilk <- son-2
                for (i in ilk:son) {
                        step <- sigmoid(nodes2[k]+runif(1, biasMin, biasMax)) * l1[i]
                        k <- k + 1
                        total1 <- step + total1
                }
                nodes3[j] <- total1
        }
        return(match(max((sigmoid(nodes3)), na.rm = FALSE),(sigmoid(nodes3))))

}

vect <- c()
for (i in 1:150) {
```

```
            vect[i] <- myNN(iris,i,l1)
}
tb <- table(as.numeric(iris[[5]]) == vect)
cat("%",(as.numeric(tb[names(tb)==TRUE]) / nrow(iris)) *100)
```

Results:

```
> cat("%",(as.numeric(tb[names(tb)==TRUE]) / nrow(iris)) *100)
% 33.33333
```

Comments:

Part 1 de bir train işlemi olmadığı için 3 class olduğundan dolayı sonucun doğru olma ihtimali %33.3 dür.

**Part 2:**

Code:

```
library(neuralnet)
set.seed(101)
size.sample <- 50
iristrain <- iris[sample(1:nrow(iris), size.sample),] # get a training sample from iris
nnet_iristrain <-iristrain
#Binarize the categorical output
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'setosa')
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'versicolor')
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'virginica')
names(nnet_iristrain)[6] <- 'setosa'
names(nnet_iristrain)[7] <- 'versicolor'
names(nnet_iristrain)[8] <- 'virginica'
nn <- neuralnet(setosa+versicolor+virginica ~ Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=nnet_iristrain, hidden=c(3))
plot(nn)
mypredict <- compute(nn, iris[-5])$net.result
# Put multiple binary output to categorical output
maxidx <- function(arr) {
    return(which(arr == max(arr)))
 }
idx <- apply(mypredict, c(1), maxidx)
prediction <- c('setosa', 'versicolor', 'virginica')[idx]
```

```
table(prediction, iris$Species)

rslt <- as.matrix(table(prediction, iris$Species))

cat("%", (sum(diag(rslt)) / sum(rslt) * 100))
```

Results:

```
> table(prediction, iris$Species)

prediction   setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         46         3
  virginica       0          4        47
> rslt <- as.matrix(table(prediction, iris$Species))
> cat("%", (sum(diag(rslt)) / sum(rslt) * 100))
% 95.33333333
> 
```

Comments:

Train işlemlerini yaptıktan sonra datayı predict ediyorum.

**Part 3:**

Code:

```
manh_dist <- function(p,q){ return(sum(abs(p-q))) }

iris<-iris[sample(nrow(iris)),]


allDistance <- function(data){

        s1 <- c()

        s2 <- c()

        s3 <- c()

        s4 <- c()

        alls <- c()


        for (i in 1:3) {

                s1[i] <- runif(1, min(data[1]), max(data[1]))

                s2[i] <- runif(1, min(data[2]), max(data[2]))

                s3[i] <- runif(1, min(data[3]), max(data[3]))

                s4[i] <- runif(1, min(data[4]), max(data[4]))

        }


        alls <- rbind(alls,s1)

        alls <- rbind(alls,s2)
```

```
	alls <- rbind(alls,s3)
	alls <- rbind(alls,s4)
	allDist <- c()

	for (i in 1:150) {
		dist <- c()
		for (k in 1:3) {
			total <- 0
			for (j in 1:4) {
				total <- manh_dist(data[i,j],as.numeric(alls[j,k])) + total
			}
			dist[k] <- total
		}
		allDist <- rbind(allDist,dist)
	}
	return(allDist)
}

allMeans <- function(subClassData1,subClassData2,subClassData3, data){
	s1 <- c()
	s2 <- c()
	s3 <- c()
	s4 <- c()
	alls <- c()

	s1[1] <- mean(subClassData1[[1]])
	s2[1] <- mean(subClassData1[[2]])
	s3[1] <- mean(subClassData1[[3]])
	s4[1] <- mean(subClassData1[[4]])

	s1[2] <- mean(subClassData2[[1]])
	s2[2] <- mean(subClassData2[[2]])
	s3[2] <- mean(subClassData2[[3]])
	s4[2] <- mean(subClassData2[[4]])
```

```
	s1[3] <- mean(subClassData3[[1]])
	s2[3] <- mean(subClassData3[[2]])
	s3[3] <- mean(subClassData3[[3]])
	s4[3] <- mean(subClassData3[[4]])

	alls <- rbind(alls,s1)
	alls <- rbind(alls,s2)
	alls <- rbind(alls,s3)
	alls <- rbind(alls,s4)
	allDist <- c()

	for (i in 1:150) {
		dist <- c()
		for (k in 1:3) {
			total <- 0
			for (j in 1:4) {
				total <- manh_dist(data[i,j],as.numeric(alls[j,k])) + total
			}
			dist[k] <- total
		}
		allDist <- rbind(allDist,dist)
	}
	return(allDist)
}

allClass <- function(allDist, data){
	class1 <- c()
	class2 <- c()
	class3 <- c()

	for (i in 1:150) {
		if(match(1,rank(allDist[i,])) == 1)
		{
			class1 <- rbind(class1, data[i,])
		}
```

```
		else if(match(1,rank(allDist[i,])) == 2)
		{
			class2 <- rbind(class2, data[i,])
		}
		else if(match(1,rank(allDist[i,])) == 3)
		{
			class3 <- rbind(class3, data[i,])
		}
	}
	print("********************************************************************************")

# class1
	tf1 <- (as.character(names(sort(table(class1[[5]]),decreasing=TRUE)[1:1])) == class1[[5]])
	tb1 <- table(as.character(names(sort(table(class1[[5]]),decreasing=TRUE)[1:1])) == class1[[5]])
	print((as.numeric(tb1[names(tb1)==TRUE]) / length(tf1)) *100)
# class2
	tf2 <- (as.character(names(sort(table(class2[[5]]),decreasing=TRUE)[1:1])) == class2[[5]])
	tb2 <- table(as.character(names(sort(table(class2[[5]]),decreasing=TRUE)[1:1])) == class2[[5]])
	print((as.numeric(tb2[names(tb2)==TRUE]) / length(tf2)) *100)
# class3
	tf3 <- (as.character(names(sort(table(class3[[5]]),decreasing=TRUE)[1:1])) == class3[[5]])
	tb3 <- table(as.character(names(sort(table(class3[[5]]),decreasing=TRUE)[1:1])) == class3[[5]])
	print((as.numeric(tb3[names(tb3)==TRUE]) / length(tf3)) *100)

	allDist <- allMeans(class1,class2,class3,iris)
	return(allDist)
}

numOfIteration <- 15
allDist1 <- allDistance(iris)
cl1 <- allClass(allDist1,iris)
for (i in 1:numOfIteration) {
	cl1 <- allClass(cl1,iris)
}
```

Results:

```
> numOfIteration <- 15
> allDist1 <- allDistance(iris)
> cl1 <- allClass(allDist1,iris)
[1]
"***************************************************************************
**********"
[1] 44.04762
[1] 73.58491
[1] 100
> for (i in 1:numOfIteration) {
+     cl1 <- allClass(cl1,iris)
+ }
[1]
"***************************************************************************
**********"
[1] 96.42857
[1] 68.05556
[1] 100
[1]
"***************************************************************************
**********"
[1] 97.05882
[1] 74.24242
[1] 100
[1]
"***************************************************************************
**********"
[1] 94.73684
[1] 77.41935
[1] 100
[1]
"***************************************************************************
**********"
[1] 90
[1] 76.66667
[1] 100
[1]
"***************************************************************************
**********"
[1] 90.90909
[1] 82.14286
[1] 100
[1]
"***************************************************************************
**********"
[1] 87.7551
[1] 86.27451
[1] 100
[1]
"***************************************************************************
**********"
[1] 84.90566
[1] 89.3617
[1] 100
[1]
"***************************************************************************
**********"
[1] 77.9661
[1] 90.2439
[1] 100
[1]
"***************************************************************************
```

```
**********"
[1] 75.80645
[1] 92.10526
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
[1]
"***************************************************************************
**********"
[1] 76.19048
[1] 94.59459
[1] 100
```

Comments:

K-Means algoritmasını implement edip belli bir iterasyon sayısı kadar çalıştırıp en iyi kümelemeyi bulmaya çalışıyorum.