```
#### Classifcation neural net
### Libraries
library(neuralnet)

### Setup
data.crabs = read.csv("australian-crabs.csv")
data.crabs$sex = ifelse(data.crabs$sex=="Male", 1, 0)
#data.RWCL = data.frame(RW=data.crabs$RW, CL=data.crabs$CL) # for
compute function

# General way to create formula
#p = dim(data.crabs)[2]
#names = names(data.crabs[4:p])
#formula = paste("sex ~ ", paste(names, collapse=" + "))
formula = "sex~RW+CL" # since the formula is so simple in this case

### Functions
## Returns missclassifcation rate for given y and yfit
mcr = function(y, yfit){
  return(mean(abs(y-yfit)))
}

### Implementation
## Train NN

# nn1 is better but harder to interpret
#nn1 = neuralnet(formula=formula, data=data.crabs, hidden=10,
threshold = 0.1, act.fct = "logistic")

# nn is basically a logistic regression model
nn = neuralnet(formula=formula, data=data.crabs, hidden=0,
err.fct="ce", linear.output = FALSE)

## Make predictions
#pred = compute(nn, data.RWCL)$net.result>0.5
pred = unlist(nn$net.result)>0.5

## Plot predictions
plot(data.crabs$CL, data.crabs$RW, col=rgb(1-
data.crabs$sex,0,data.crabs$sex, 0.5), pch=19)
points(data.crabs$CL, data.crabs$RW, col=rgb(1-pred,0,pred, 0.5),
pch=19)

## Calculate MCR
#mcr(pred, data.crabs$sex) # 0.025 for nn1
mcr(pred, data.crabs$sex) # 0.035

## Boundry
#plot(nn) # to extract which weight correspond to which for
calculation of boundry

# Boundry through 'lines'
#nn.weights = unlist(nn$weights)
#boundry = (nn.weights[3]*data.crabs$CL+nn.weights[1])/-nn.weights[2]
```

```
#lines(data.crabs$CL, boundry)

# Boundry through 'abline'
intercept = -nn.weights[1]/nn.weights[2]
slope = -nn.weights[3]/nn.weights[2]
abline(intercept, slope, col=1)
```