Neural_Networks_with_R

Odo Luo

March 24, 2021

Homework 3: Neural Networks with R

1 and 2

1. Use the iris data set (data(iris)) with the four features

a. Sepal.Length
b. Sepal.Width
c. Petal.Length
d. Petal.Width and the target Species with the three classes setosa, versicolor, and virginica

2. Create train and test data using a fixed split of 70-30

```
library(knitr)
library(datasets)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
data <- iris %>%
  mutate(id=row_number())
train <- data %>%sample_frac(.70)
test <-  anti_join(data, train,'id')
train <- train %>% select(-id)
test <- test %>% select(-id)
```

3

3. Use the training set to find a good MLP architecture with:

a. 1 to 2 hidden layers and
b. 2 to 5 neurons in the hidden layers

and a logistic activation function. Find the best of all 8 possible architectures (2x4 = 8 combinations; grid search with 10-times stratified1 cross validation or use a validation set – 80-10-10) and use it in the following tasks.

Creating Folds for 10-Fold Cross Validation:

```
train$Species<-as.factor(train$Species)
train<-train %>% mutate(id=row_number())
folds <- createFolds(train$Species)
#train<-cbind(train,dummy(train$Species,sep = "_"))
#train <- train %>% select(-Species)
```

Getting vector "hidden" attribute in neuralnet:

```
getHiddenLayerVector <- function(hiddenLayerNum,neuronsNum){
  hiddenLayer = c()
  for(i in 1:hiddenLayerNum){
    hiddenLayer[i]= neuronsNum
  }
  return (hiddenLayer)
}
```

Calculating mean(accuracy) using 10fold CV:

```
hiddenLayerNum =  2
hiddenLayerNeurons=5

result <- data.frame(layers=integer(),neurons=integer(),accuracy=double())
#1:2 Layer
for( layerNum in 1:hiddenLayerNum){

  #2:5 Neurons per Layer
  for(neuronsNum in 2:hiddenLayerNeurons){
    out<-NULL

    #CV Iterration
    for(cv in 1:10){
      test_cv= train[folds[[cv]],]
      train_cv= train[-test_cv$id,]
      nn= neuralnet(Species~Sepal.Length+Sepal.Width+Petal.Length,data=train_cv,
                    hidden=getHiddenLayerVector(layerNum,neuronsNum),
                    act.fct = "logistic",linear.output = FALSE,rep=2,threshold = 0.5)
```

```
      prediction <- neuralnet::compute(nn, (test_cv %>% select(-id,-Species)))
      prediction <- max.col(prediction$net.result) %>%  as.factor()
      prediction <- factor(prediction,levels = c(1,2,3))
      real <- test_cv %>%   mutate(Species= ifelse(Species=="setosa",1,ifelse(Species=="versicolor",2,i
      real<- as.factor(real[,"Species"])

      c<-confusionMatrix(real,prediction)
      out[cv]<-c$overall[["Accuracy"]]
    }
    result <- rbind(result,data.frame(layerNum,neuronsNum,mean(out)))
  }
}

result
```

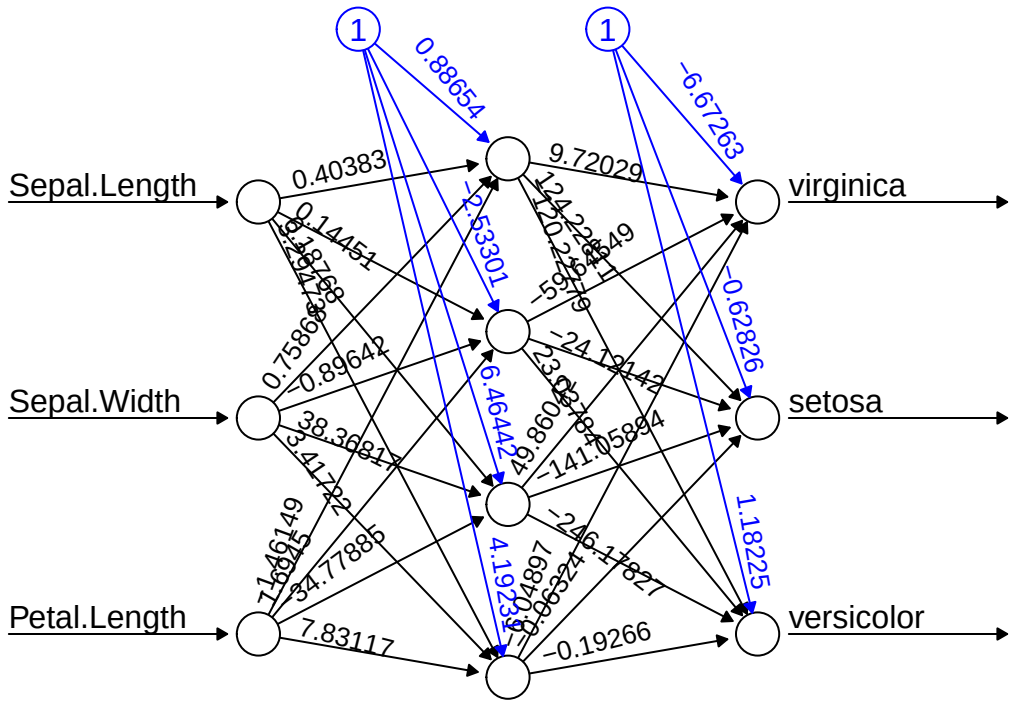
```
##   layerNum neuronsNum mean.out.
## 1        1          2 0.7321212
## 2        1          3 0.9709091
## 3        1          4 0.8609091
## 4        1          5 0.9909091
## 5        2          2 0.6325253
## 6        2          3 0.7493939
## 7        2          4 0.8854545
## 8        2          5 0.9627273
```

4 Visualize the best model using the plot function of neuralnet.

Both, one hidden layer with either 4 or 5 neurons per layer show good results, so both will be visualized.

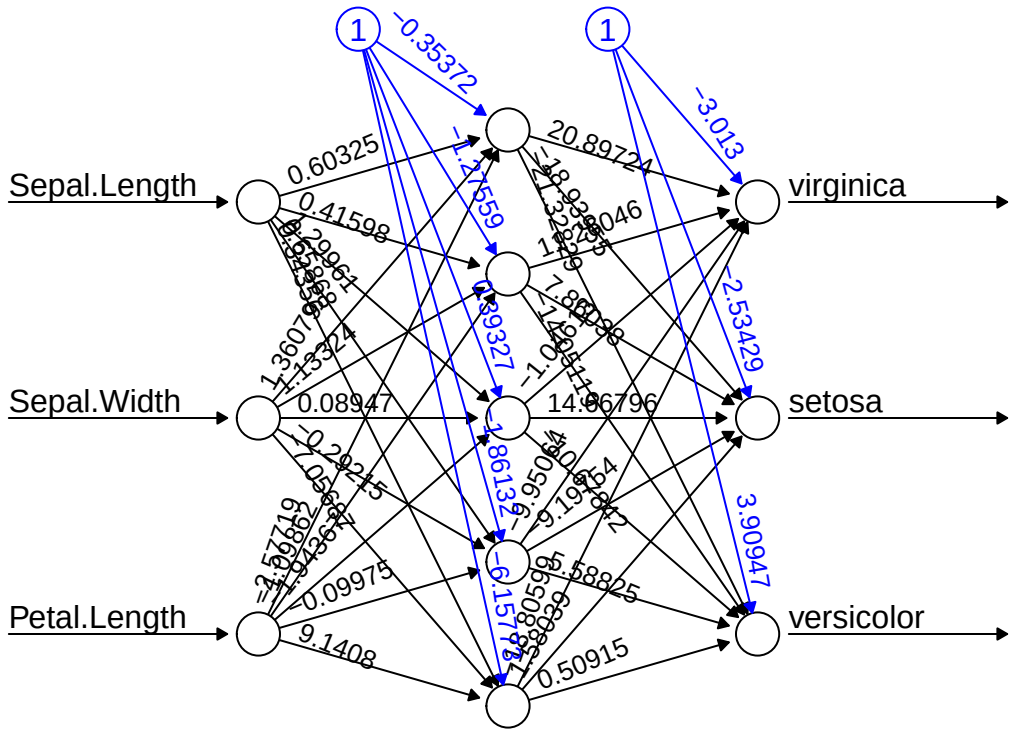
```
nnFour= neuralnet( Species ~Sepal.Length+Sepal.Width+Petal.Length,data=train,
                   hidden=4,
                   act.fct = "logistic",linear.output = FALSE,rep=2,threshold = 0.5)
```

```
plot(nnFour,1)
```



Error: 0.415905 Steps: 2477

```
nnFive= neuralnet( Species ~Sepal.Length+Sepal.Width+Petal.Length,data=train,
                   hidden=5,
                   act.fct = "logistic",linear.output = FALSE,rep=2,threshold = 0.5)
plot(nnFive,1)
```



Error: 1.685796 Steps: 207

5 Use the best model with the winning parameters to evaluate test loss and interpret the confusion matrix.

Both, one hidden layer with either 4 or 5 neurons per layer show good results, so both will be evaluated.

Real Species

```
real <- test %>%
  mutate(Species= ifelse(Species=="setosa",
                         1,ifelse(Species=="versicolor",2,ifelse(Species=="virginica",3,NA))))
real<- as.factor(real[,"Species"])
```

```
prediction1 <- neuralnet::compute(nnFour, (test %>% select(-Species)))
prediction1 <- max.col(prediction1$net.result) %>%  as.factor()
prediction1 <- factor(prediction1,levels = c(1,2,3))

confusionMatrix(real,prediction1)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  1  2  3
##          1 17  0  0
##          2  0 10  4
##          3  0  0 14
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9111
##                  95% CI : (0.7878, 0.9752)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 9.959e-13
##
##                   Kappa : 0.8661
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   1.0000   0.7778
## Specificity            1.0000   0.8857   1.0000
## Pos Pred Value         1.0000   0.7143   1.0000
## Neg Pred Value         1.0000   1.0000   0.8710
## Prevalence             0.3778   0.2222   0.4000
## Detection Rate         0.3778   0.2222   0.3111
## Detection Prevalence   0.3778   0.3111   0.3111
## Balanced Accuracy      1.0000   0.9429   0.8889
```

```
prediction2 <- neuralnet::compute(nnFive, (test %>% select(-Species)))
prediction2 <- max.col(prediction2$net.result) %>%  as.factor()
prediction2 <- factor(prediction2,levels = c(1,2,3))

confusionMatrix(real,prediction2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 17  0  0
##          2  0 11  3
##          3  0  1 13
##
## Overall Statistics
##
##                Accuracy : 0.9111
##                  95% CI : (0.7878, 0.9752)
##     No Information Rate : 0.3778
##     P-Value [Acc > NIR] : 1.099e-13
##
##                   Kappa : 0.8661
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9167   0.8125
## Specificity            1.0000   0.9091   0.9655
## Pos Pred Value         1.0000   0.7857   0.9286
## Neg Pred Value         1.0000   0.9677   0.9032
```

```
## Prevalence             0.3778   0.2667   0.3556
## Detection Rate         0.3778   0.2444   0.2889
## Detection Prevalence   0.3778   0.3111   0.3111
## Balanced Accuracy      1.0000   0.9129   0.8890
```

Interpretatin Confusion Matrix

The NN with 5 Neuros seems to have a overall better Accuracy (True/False Ratio). However, the NN with 4 Neurons seems to have better Senstivity regarding Class 1 and Class 2 but a worse Senstivity towards Class 3. Including senstivity from both matrizes, data suggest that the both NN are good in indentifying class 1 cases, but the NN wiht 4 Neurons in the hidden layer tends to (wrongly) classfies more cases as Class 2.

6 Discuss the problem of overfitting in neural networks and how you can spot it as well as prevent it from happening.

Standard way to avoid overfitting is by applying cross validation.

https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a
https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/