Progetto: Neural Networks

Davide Zicca

27/6/2021

Motivazione e Data Visualization

Da uno dei repository online di dataset per allenarsi con il Machine Learning, ho scelto il dataset: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data. Esso contiene 392 osservazioni e 9 variabili. Si tratta di un dataset contenente info su autoveicoli. La prima operazione eseguita è stata quindi quella di caricare il dataset su R e procedere con la data visualization:

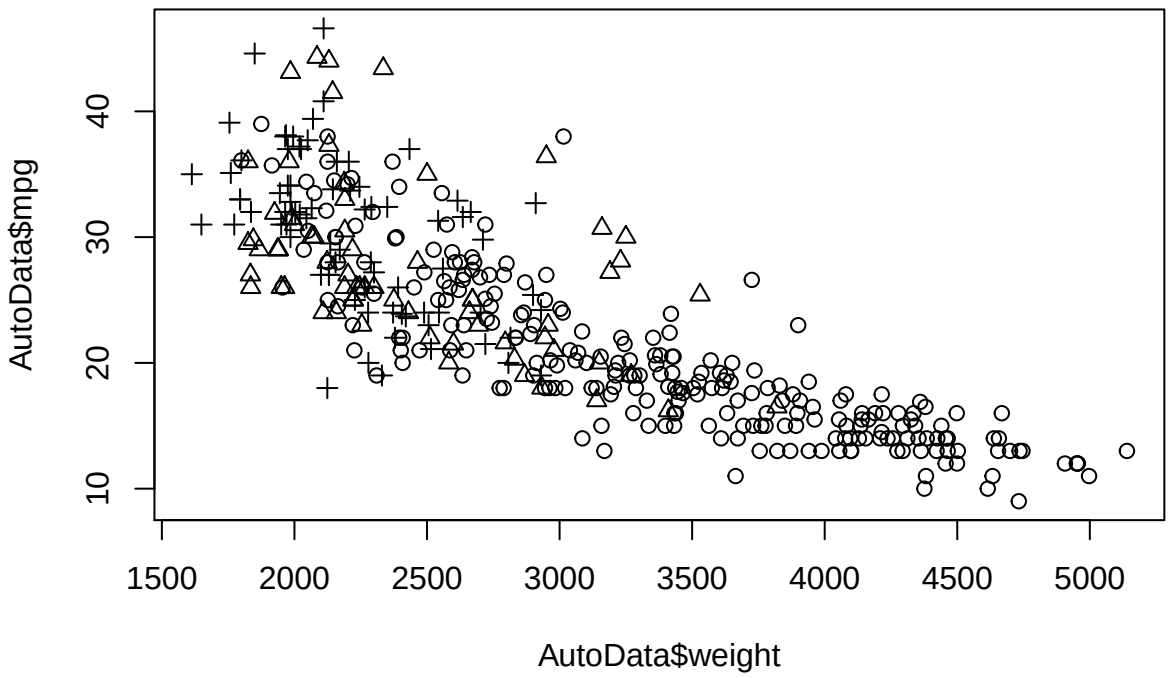
```
AutoData <- read.table(url("https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg
names(AutoData)<- c("mpg","cylinders","displacement","horsepower","weight","acceleration",
                    "year","origin","name")
str(AutoData)
```

```
## 'data.frame':    398 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : chr  "130.0" "165.0" "150.0" "150.0" ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel
```

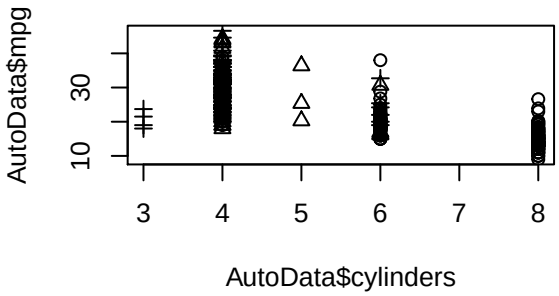
```
AutoData<-AutoData[!(AutoData$horsepower=="?"),]
AutoData$horsepower<-as.integer(AutoData$horsepower)
str(AutoData)
```

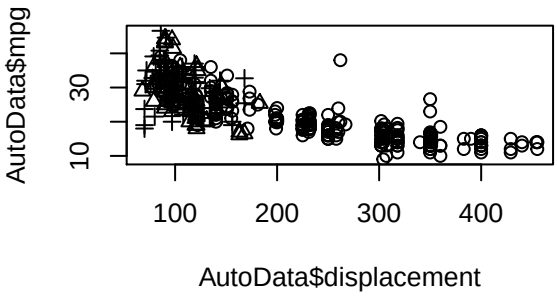
```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : int  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel
```

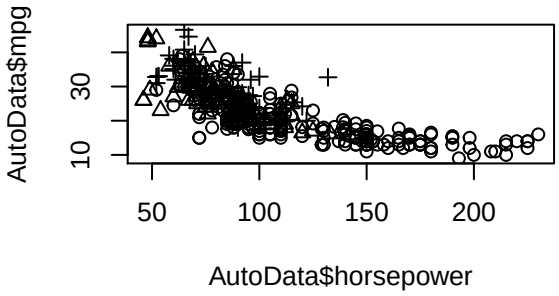
```
plot(AutoData$weight, AutoData$mpg, pch=AutoData$origin)
```

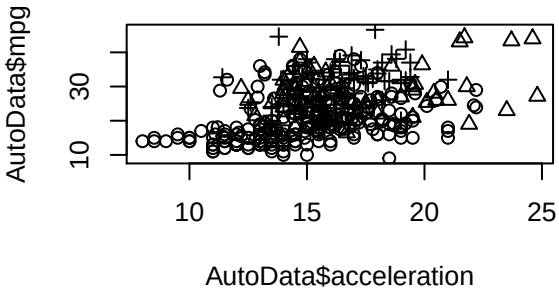



```
par(mfrow=c(2,2))
plot(AutoData$cylinders, AutoData$mpg, pch=AutoData$origin)
plot(AutoData$displacement, AutoData$mpg, pch=AutoData$origin)
plot(AutoData$horsepower, AutoData$mpg, pch=AutoData$origin)
plot(AutoData$acceleration, AutoData$mpg, pch=AutoData$origin)
```










```
dev.off()
```

```
## null device
##           1
```

Fit del modello

Prima di procedere alla modellazione, sono state calcolate:

1. media
2. varianza
3. *scale* dei dati
4. divisione del dataset in train e test

```
mean_data <- apply(AutoData[1:6], 2, mean)
sd_data <- apply(AutoData[1:6], 2, sd)

AutoDataScaled <- as.data.frame(scale(AutoData[,1:6],center =
                                      mean_data, scale = sd_data))
head(AutoDataScaled, n=20)
```

```
##            mpg  cylinders displacement horsepower     weight acceleration
## 1  -0.69774672  1.4820530   1.07591459  0.6632851  0.6197483  -1.28361760
## 2  -1.08211534  1.4820530   1.48683159  1.5725848  0.8422577  -1.46485160
## 3  -0.69774672  1.4820530   1.18103289  1.1828849  0.5396921  -1.64608561
## 4  -0.95399247  1.4820530   1.04724596  1.1828849  0.5361602  -1.28361760
```

```
## 5  -0.82586959  1.4820530   1.02813354  0.9230850  0.5549969  -1.82731962
## 6  -1.08211534  1.4820530   2.24177212  2.4299245  1.6051468  -2.00855363
## 7  -1.21023822  1.4820530   2.48067735  3.0014843  1.6204517  -2.37102164
## 8  -1.21023822  1.4820530   2.34689042  2.8715843  1.5710052  -2.55225565
## 9  -1.21023822  1.4820530   2.49023356  3.1313843  1.7040399  -2.00855363
## 10 -1.08211534  1.4820530   1.86907996  2.2220846  1.0270935  -2.55225565
## 11 -1.08211534  1.4820530   1.80218649  1.7024847  0.6892089  -2.00855363
## 12 -1.21023822  1.4820530   1.39126949  1.4426848  0.7433646  -2.73348966
## 13 -1.08211534  1.4820530   1.96464205  1.1828849  0.9223139  -2.18978763
## 14 -1.21023822  1.4820530   2.49023356  3.1313843  0.1276377  -2.00855363
## 15  0.07099053 -0.8629108  -0.77799001 -0.2460146 -0.7129531  -0.19621355
## 16 -0.18525522  0.3095711   0.03428778 -0.2460146 -0.1702187  -0.01497955
## 17 -0.69774672  0.3095711   0.04384399 -0.1940546 -0.2396793  -0.01497955
## 18 -0.31337809  0.3095711   0.05340019 -0.5058145 -0.4598340   0.16625446
## 19  0.45535916 -0.8629108  -0.93088936 -0.4278746 -0.9978592  -0.37744756
## 20  0.32723628 -0.8629108  -0.93088936 -1.5190342 -1.3451622   1.79736053
```

```
index = sample(1:nrow(AutoData),round(0.70*nrow(AutoData)))
train_data <- as.data.frame(AutoDataScaled[index,])
test_data <- as.data.frame(AutoDataScaled[-index,])

n = names(AutoDataScaled)
f = as.formula(paste("mpg ~", paste(n[!n %in% "mpg"],
                                    collapse = " + ")))
library(neuralnet)
NNRModel<-neuralnet(f,data=train_data,hidden=3,linear.output=TRUE)
```

Output del modello e plot:

```
summary(NNRModel)
```

```
##                     Length Class      Mode
## call                   5   -none-     call
## response             274   -none-     numeric
## covariate           1370   -none-     numeric
## model.list             2   -none-     list
## err.fct                1   -none-     function
## act.fct                1   -none-     function
## linear.output          1   -none-     logical
## data                   6   data.frame list
## exclude                0   -none-     NULL
## net.result             1   -none-     list
## weights                1   -none-     list
## generalized.weights    1   -none-     list
## startweights           1   -none-     list
## result.matrix         25   -none-     numeric
```

```
plot(NNRModel,cex=0.6,cex.axis=0.6,cex.lab=0.6)
```

Calcolo delle predizione e del MSE:

```
NNRModel$result.matrix
```

```
##                                [,1]
## error                 2.892353e+01
## reached.threshold     9.047677e-03
## steps                 3.474000e+03
```

```
## Intercept.to.1layhid1    -6.815453e+00
## cylinders.to.1layhid1     1.437443e+01
## displacement.to.1layhid1  1.331045e+01
## horsepower.to.1layhid1   -6.392218e+00
## weight.to.1layhid1       -3.642454e+00
## acceleration.to.1layhid1  1.777997e+00
## Intercept.to.1layhid2    -3.913971e+00
## cylinders.to.1layhid2     1.025031e+00
## displacement.to.1layhid2  1.716987e+00
## horsepower.to.1layhid2   -7.281453e+00
## weight.to.1layhid2       -9.191111e-01
## acceleration.to.1layhid2 -9.262792e-01
## Intercept.to.1layhid3     2.735325e+00
## cylinders.to.1layhid3    -3.497342e-01
## displacement.to.1layhid3  5.721799e+00
## horsepower.to.1layhid3   -3.841771e+00
## weight.to.1layhid3       -3.121711e+00
## acceleration.to.1layhid3 -1.064958e-01
## Intercept.to.mpg         -4.786843e-01
## 1layhid1.to.mpg          -1.141293e+00
## 1layhid2.to.mpg           1.220946e+00
## 1layhid3.to.mpg           7.967548e-01
```

```
PredNetTest <- compute(NNRModel,test_data[,2:6])

MSE.net <- sum((test_data$mpg - PredNetTest$net.result)^2)/nrow(test_data)
```

Confronto del modello neural network con una regressione lineare

Fit della regressione lineare e calcolo MSE:

```
LModel <- lm(mpg~., data=train_data)
summary(LModel)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4839 -0.3459 -0.0578  0.3069  2.0883
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.01978    0.03360   0.589  0.55649
## cylinders    -0.17145    0.10870  -1.577  0.11589
## displacement  0.14557    0.15327   0.950  0.34308
## horsepower   -0.33892    0.10824  -3.131  0.00193 **
## weight       -0.53376    0.10769  -4.957 1.27e-06 ***
## acceleration -0.03671    0.05607  -0.655  0.51312
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5519 on 268 degrees of freedom
```

```
## Multiple R-squared:  0.7135, Adjusted R-squared:  0.7082
## F-statistic: 133.5 on 5 and 268 DF,  p-value: < 2.2e-16
```

```
PredLModel <- predict(LModel,test_data)

MSE.lm <- sum((PredLModel - test_data$mpg)^2)/nrow(test_data)
```

Confronto dell'errore quadratico medio del modello neural network e della regressione lineare:

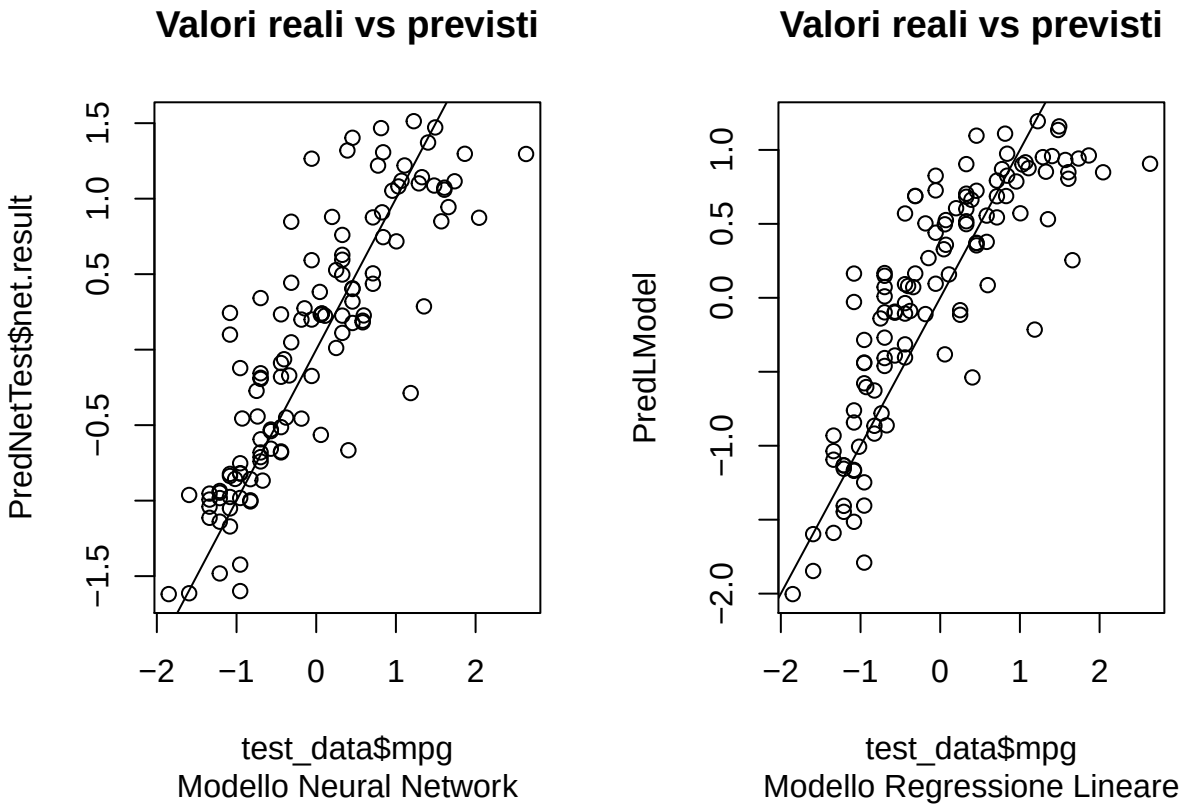
```
MSE_netVSlm= as.data.frame(table(MSE.net, MSE.lm))
MSE_netVSlm= subset(MSE_netVSlm, select = -Freq)
MSE_netVSlm
```

```
##              MSE.net             MSE.lm
## 1 0.248609797885515 0.285289920687686
```

Confronto grafico:

1. Modello Neural Network
2. Modello Regressione Lineare

```
par(mfrow=c(1,2))
plot(test_data$mpg,PredNetTest$net.result,col='black',main="Valori reali vs previsti",
     sub = "Modello Neural Network")
abline(0,1,lwd=1)
plot(test_data$mpg,PredLModel,col='black',main="Valori reali vs previsti",
     sub = "Modello Regressione Lineare")
abline(0,1,lwd=1)
```



Conclusione

Dal confronto del MSE e dal confronto grafico si evince come il modello neural network abbia un MSE inferiore e, quindi, è da preferire rispetto alla regressione lineare Nel modello neuralnet i valori dei dati sono dispersi vicino al suo momento centrale (media), mentre nella regressione lineare i valori sono più dispersi.