NYU Machine Learning in Economics

Lecture 9: Neural Networks - R Lab Part 2

Dr. George Lentzas

Neural Network Overfitting

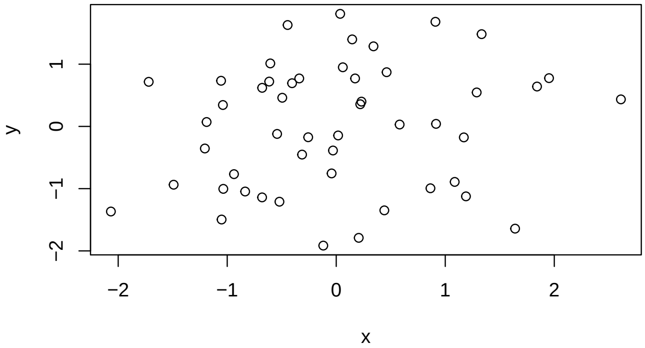
Here we will use the package RSNNS— to fit a number of Neural Networks. First let us see a case of over-fitting.

```r
x <- scale( rnorm(50, 0,1) )
y <- scale(rnorm(50, 0,1) )

net1 <- RSNNS::mlp(x=x,y=y,size = c(10), maxit = 10000,
                   learnFuncParams = 0.0001, linOut = TRUE )
```

Neural Network Overfitting

The data is clear noise.

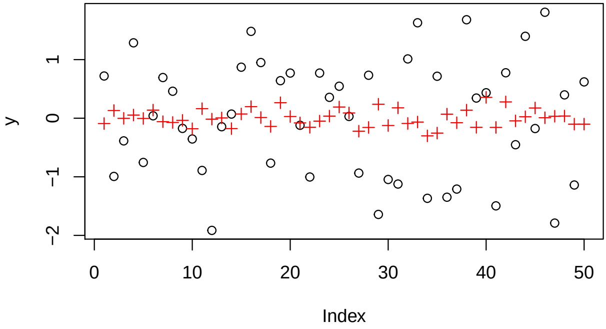
```
plot(x,y)
```



Neural Network Overfitting

The network is trying to uncover some pattern that is clearly not there.

```r
plot(y)
points(net1$fitted.values, col = "red", pch = 3)
```



Now to a Finance Application

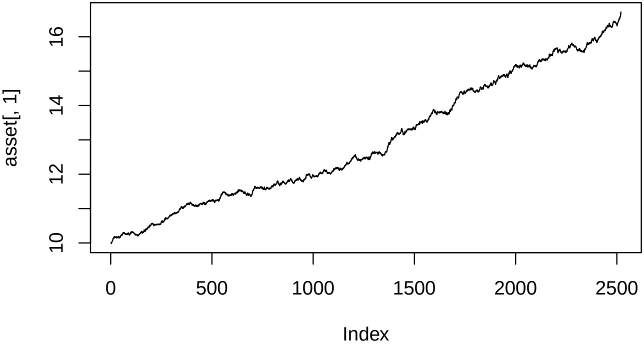
Now let us turn to an application inspired by Hutchinson, Lo and Poggio (1994). First we will simulate an asset with an annual return of 5% and an annual volatility of 60%.

```
asset <- data.frame(matrix(10,2520,1))
mu <- 0.05; sigma <- 0.6

  for (i in 2:2520){

    asset[i,1] <-  asset[i-1,1] * exp( rnorm(1, mu /252, sigma ^2 /252))

  }
```

Neural Network Overfitting

Lets see how this asset did over the last ten years. Not bad!

```
plot( asset[, 1], type="l")
```



Great it looks reasonably realistic!

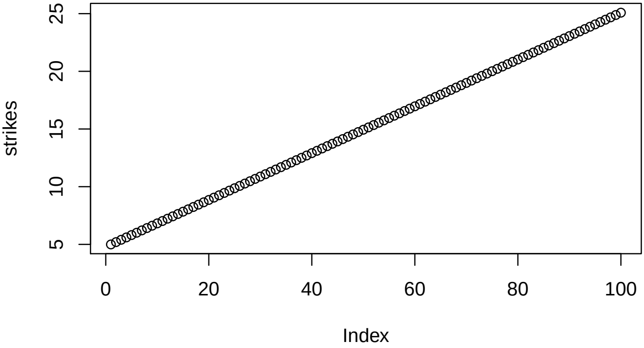
Black Scholes Simulated Prices

Now let us simulate some Black-Scholes vanilla option prices. First we need to come up with some reasonable strikes.

```r
max_asset <- max(asset)
min_asset <- min(asset)

strikes <- seq(min_asset * 0.5, max_asset * 1.5, length.out = 100)
```

Neural Network Overfitting

```
plot(strikes)
```



Black Scholes Simulated Prices

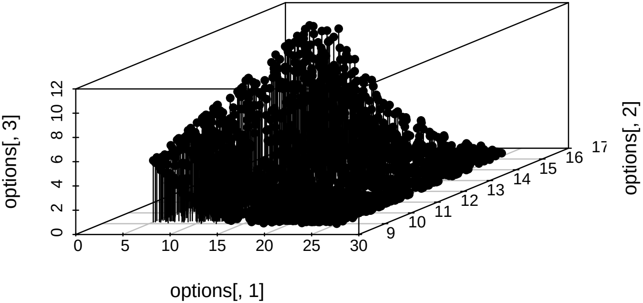
Now to the option prices.

```
options <- data.frame(matrix(NA, dim(asset)[1], 3))
options[, 1] <- sample(strikes, size = dim(asset)[1], replace = TRUE)
options[, 2] <- asset[sample(dim(asset)[1]), ]

rtrn <- na.omit(timeSeries::returns(asset[,1]))
sigma_est <- sqrt(sd(rtrn[,1])  * 252)
mu_est <- mean(rtrn[,1]) * 252
for (i in 1:dim(options)[1]){

 BS <- fOptions::GBSOption(TypeFlag = "c", S = options[i, 2],
                           X = options[i, 1], Time = 1/4,
                           r = 0.08,  b = 0, sigma = sigma_est)
 options[i, 3] <- BS@price
}
```

Let us plot these simulated option prices as functions of strike and price; you can see the obvious non-linearity. Can a Neural Network estimate this relationship?

Black Scholes Simulated Prices

```
scatterplot3d::scatterplot3d(x = options[,1], y =options[,2],   z = options[,3],  pch = 16,   type="h")
```



Neural Network Overfitting

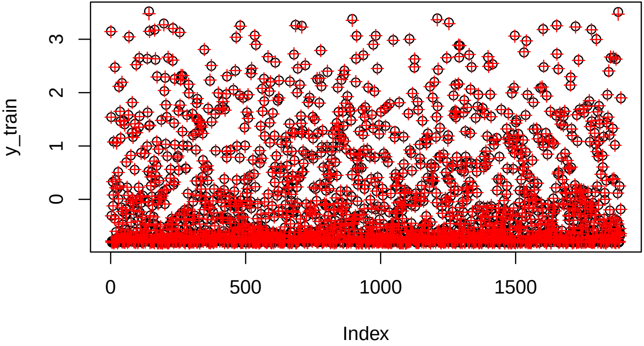
We will use a two layer, ten by ten hidden unit Neural Network.

```r
set.seed(1)
my_sample <- sample.int(nrow(options), round(.75*nrow(options)), replace = F)

train_options <- options[my_sample, ]
test_options <- options[-my_sample, ]
y_train <- scale(train_options[,3])
x_train <- scale(train_options[,1:2])
y_test <- scale(test_options[,3])
x_test <- scale(test_options[,1:2])

net2 <-  RSNNS::mlp(x=x_train,y=y_train,size = c(10,10),
                    maxit = 1000, learnFuncParams = 0.01, linOut = TRUE )
```

First we can visualize in sample performance.

Neural Network Overfitting

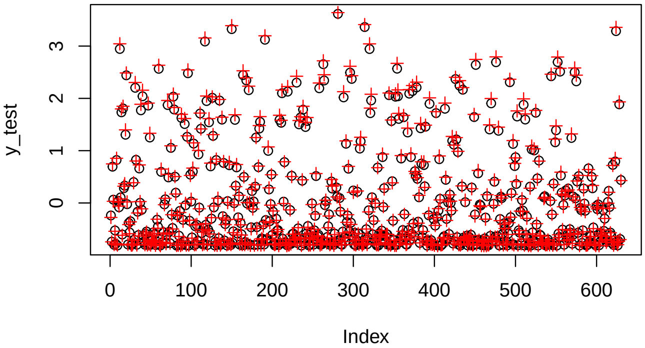
```
plot(y_train)
points(net2$fitted.values, col = "red", pch = 3)
```



Neural Network Overfitting

Let us now see how well the fitted network does out of sample.

```
plot(y_test)
points(predict(net2, x_test), col = "red", pch = 3)
```


y_test
Index