R Programming - Challenge B

Thorunn Helgadóttir and Jérôme Poulain

22/11/2017

Task 1B - Predicting house prices in Iowa

Step 1: Feedforward neural network

A feedforward neural network defines a mapping $y = f(x; \theta)$ where the objective is to learn the value of the parameters θ that result in the best function approximation. Information travels first through the input nodes, then through the hidden nodes and finally through the output nodes. Information travels only in one direction in the network, which is forward, so there are no loops nor feedback connections.

Step 2: Train technique

```r
library(nnet)

training<-nnet(SalePrice~.,train,size=3,linout=TRUE,skip=TRUE)
```

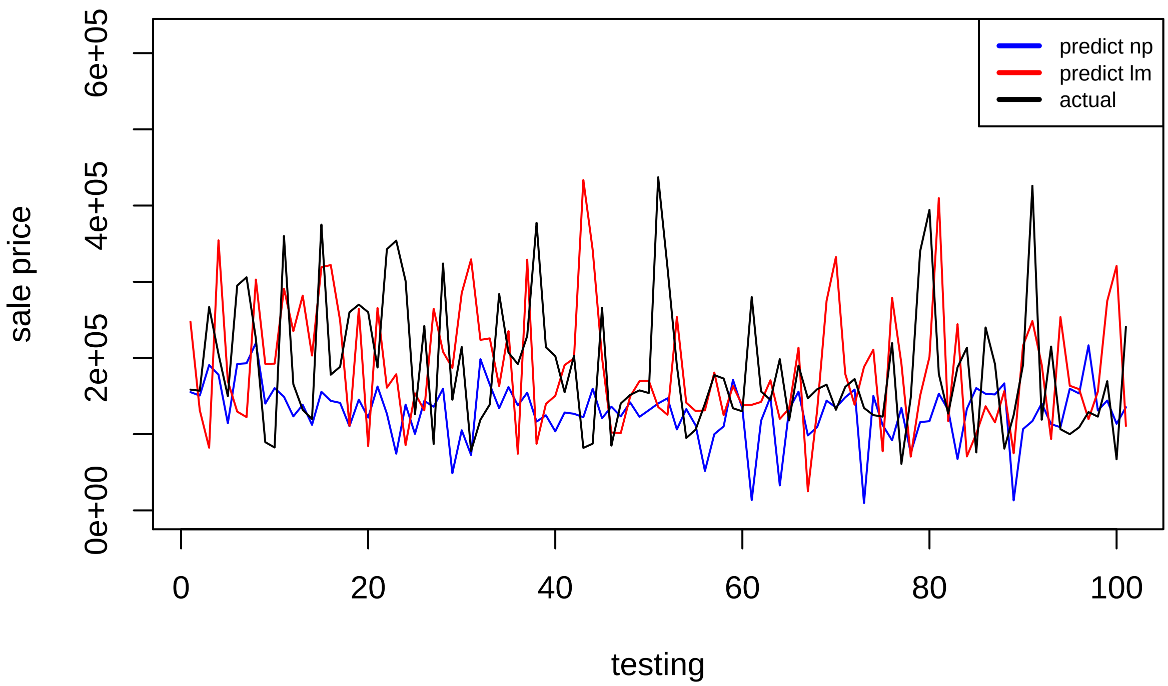
```
## # weights:  931
## initial  value 54000127089806.335937
## iter  10 value 3978334847833.084961
## iter  20 value 3464442486184.060059
## iter  30 value 1549351826891.888672
## iter  40 value 1374536167175.844727
## iter  50 value 1126031016560.729980
## iter  60 value 960794783465.447754
## iter  70 value 831281230542.217407
## iter  80 value 764320744759.812988
## iter  90 value 730066246732.057251
## iter 100 value 691761881548.019531
## final  value 691761881548.019531
## stopped after 100 iterations
```

Step 3: Predictions

```r
predict<-predict(training,test)

predictions<-data.frame(predict=predict,actual=train[1:1459,74])
head(predictions)
```

```
##    predict actual
## 1 111937.6 208500
## 2 147690.4 181500
## 3 184482.1 223500
## 4 191305.5 140000
## 5 209070.2 250000
## 6 172809.8 143000
```



predict np
predict lm
actual
sale price
0e+00
2e+05
4e+05
6e+05
0
20
40
60
80
100
testing

TASK 2B - OVERFITTING IN MACHINE LEARNING (continued)

Step 1: Estimating a low flexibility local linear model

```r
# model_low
ll.fit.lowflex <-npreg(y~x,bws=0.5, training_data, regtype="ll")
summary(ll.fit.lowflex)
```

```
##
## Regression Data: 122 training points, in 1 variable(s)
##                  x
## Bandwidth(s): 0.5
##
## Kernel Regression Estimator: Local-Linear
## Bandwidth Type: Fixed
## Residual standard error: 1.085438
## R-squared: 0.8540956
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

```r
y1fit<-fitted.values(ll.fit.lowflex)

df_y1fit<-data.frame(y1fit)
```

Step 2: Estimating a high flexibility local linear model

```r
# model_high
ll.fit.highflex <-npreg(y~x,bws=0.01, training_data, regtype="ll")
summary(ll.fit.highflex)
```

```
##
## Regression Data: 122 training points, in 1 variable(s)
##                   x
## Bandwidth(s): 0.01
##
## Kernel Regression Estimator: Local-Linear
## Bandwidth Type: Fixed
## Residual standard error: 0.5070779
## R-squared: 0.9680171
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 1
```

```r
y2fit<-fitted.values(ll.fit.highflex)
df_y2fit<-data.frame(y2fit)
```

Step 3: Plot scatterplot

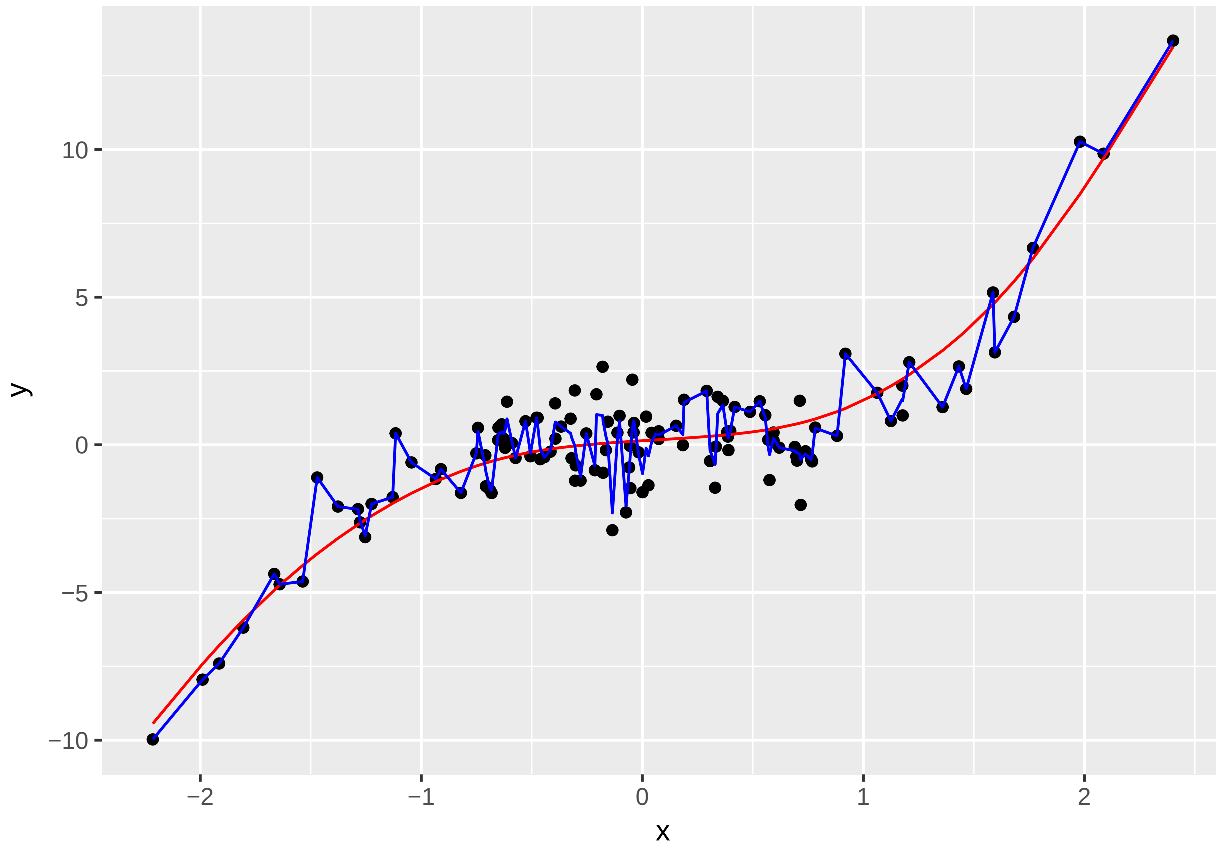


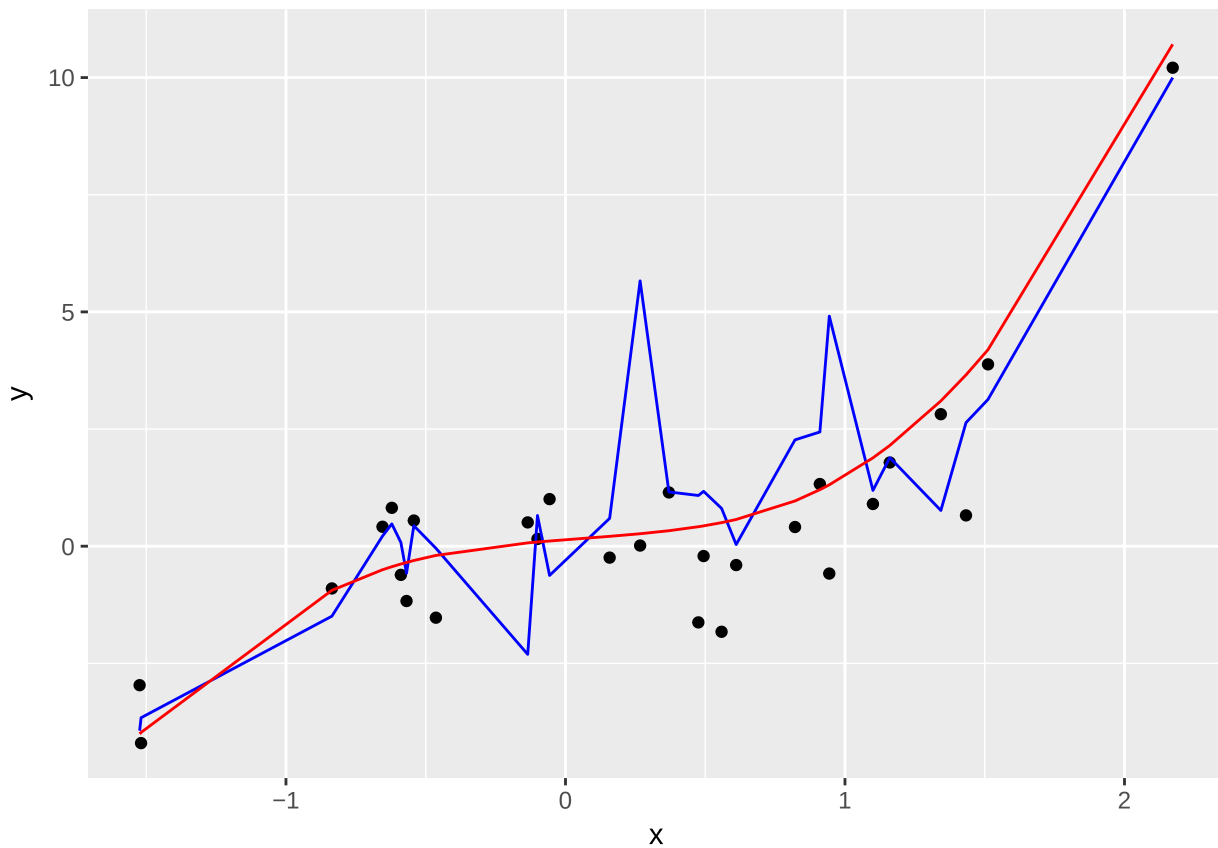
Figure 1: Predictions of ll.fit.lowflex and ll.fit.highflex on training data

Step 4: Model comparison

The predictions from the high flexibility local linear model are more variable since they fluctuate more around the scatter points. However, since they're closer to the actual values, represented by the scatter points, the high flexibility model is less biased.

Step 5: Plot predictions using test data

```
# model_low
predh<-predict(ll.fit.highflex, newdata=testing_data)
predl<-predict(ll.fit.lowflex, newdata=testing_data)
```



The predictions from the high flexibility model are more variable since they fluctuate more around the scatter points.The bias in the high flexibility model has increased since the predictions lie further away from the scatter.

Step 6: creating a vector of bandwidth

Step 7: variation of our estimations according to different flexibility

We summarized all our fitted values from the regressions for each bandwidth in one matrix called df_finalyfit, whose each column is the vector of the fitted values of the corresponding brandwidth

Step 8: computing the MSE for training data

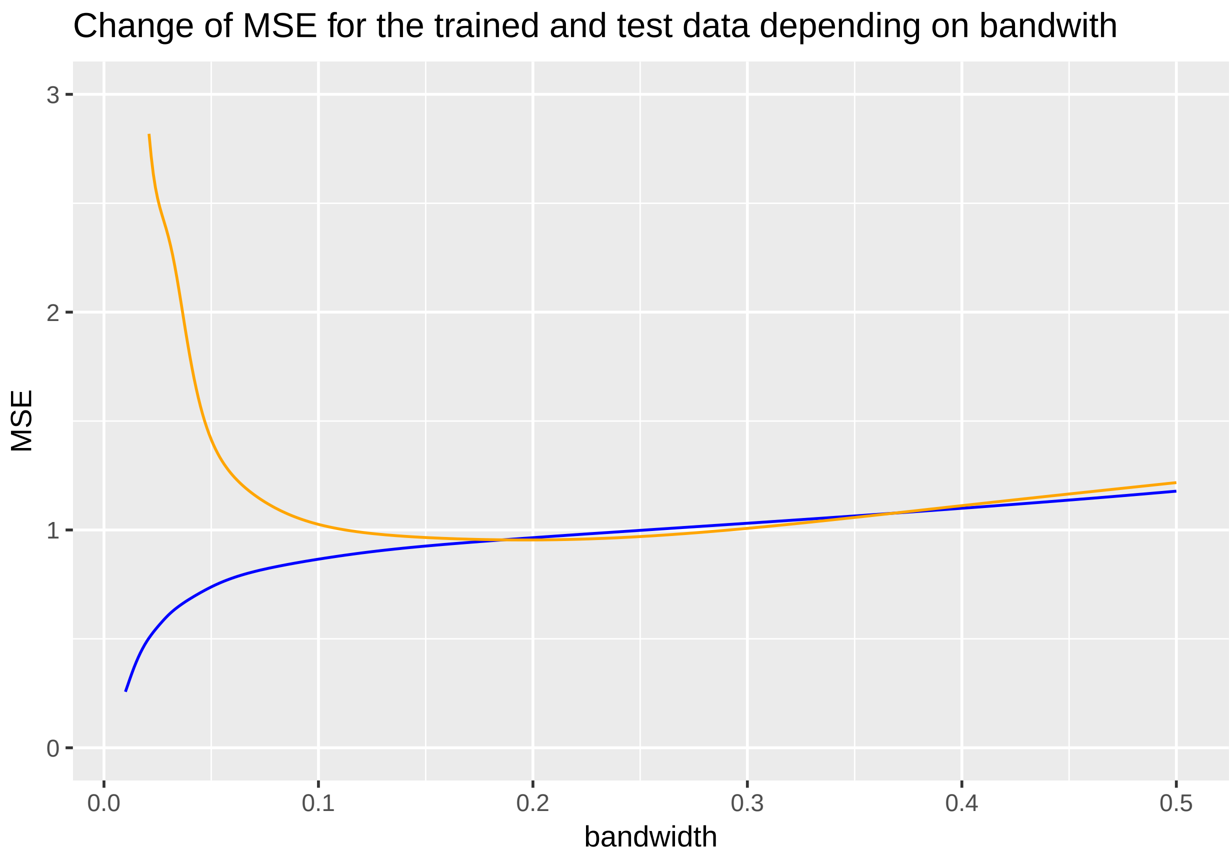
We summarized the MSE of our models in the vector MSEtrain

Step 9: computing the MSE for the test data

Simmilarly, we summarized our MSE of our models using the "test data" in thhe vector MSEtest

Because some MSE (those of the 10 highest flexibility models) are irrelevant , I will not take them into account into my plot (actually, they only accentuate the idea that predictions from models with high felxibility are very biased).

Step 10: plot MSE



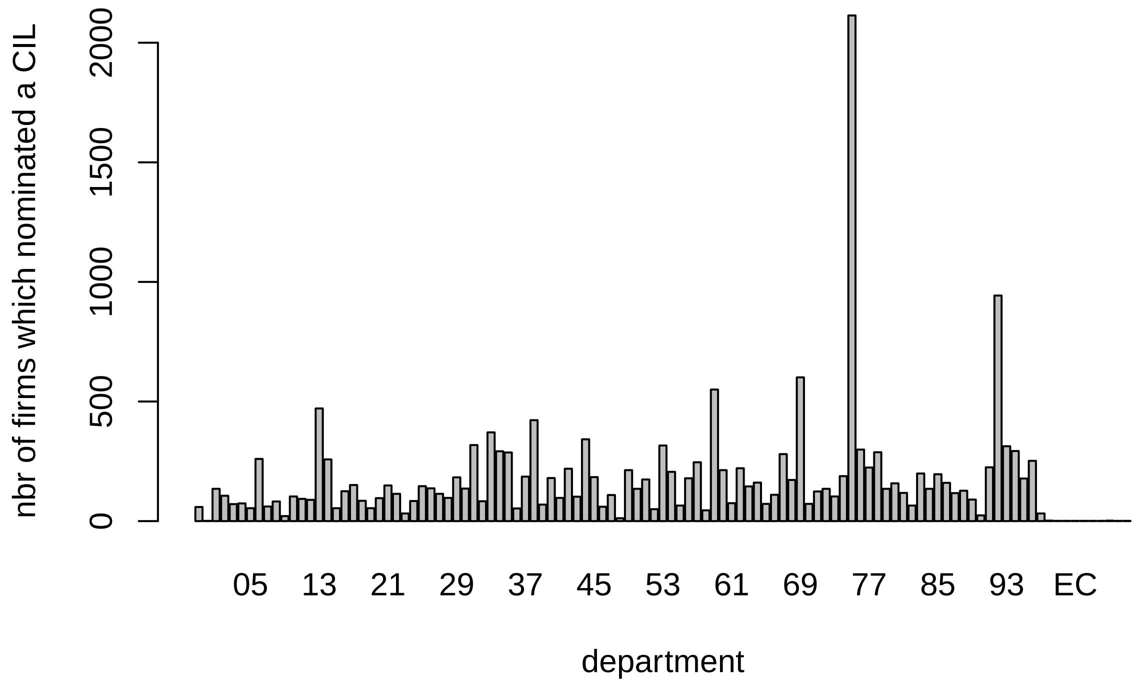
To sum up, we see in our plot that very flexible models (with the lowest bandwidth) produce baised predictions as soons as you change your sample, whereas models with less flexibility are less biased regardless of the data. We may add that the MSE curve has a minimum (here, for the bandwith of value 0.196)

Task 3B - Privacy regulation compliance in France

Step 1: import the cnil dataset

```
##      Siren           Responsable         Adresse
##  Min.   :        0   Length:18629      Length:18629
##  1st Qu.:328922067   Class :character   Class :character
##  Median :413893454   Mode  :character   Mode  :character
##  Mean   :457377665
##  3rd Qu.:520301044
##  Max.   :999999999
##  NA's   :301
##  Code_Postal           Ville              NAF
##  Length:18629       Length:18629       Length:18629
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    TypeCIL            Portee
##  Length:18629       Length:18629
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

Step 2: nice table



Step 3: the SIREN file

My computer is not powerful enough to solve this question, and unfortunately We're running out of time to try on another device. Nevertheless, Here are some unsuccessful attempts to solve it. - increasing my memory with memory.limit - dealing with big data thanks to fread, read.csv.ffdf, read.csv2.ffdf - I also tried to divide my SIREN datadset in smaller portions, then creating a loop to fill so that every row of the CNIL file look into each small portion until finding its equivalence (but my code was not working)