Lista 2 - Mineração de dados

Luben, Luiz Piccin, Vinicius Hideki

10/4/2021

Exercício 1:

Podemos implementar a regressão linear local da seguinte maneira:

```
# kernel usado
# escrevendo 3 funcoes de kernel
# kernel gaussiano
gauss_kernel = function(x, x_i, h){
  return(1/sqrt(2*pi*(h^2))*exp(-(x - x_i)^2/(2*h^2)))
}

# epanechnikov
epanech_kernel = function(x, x_i, h){
  return(1 - ((x - x_i)^2/(h^2))*((x - x_i)^2 <= h))
}

# unifrome
unif_kernel = function(x, x_i, h){
  return((x - x_i)^2 <= h)
}


loc_lm = function(x_train, y_train, x_test, h, kernel){
  # calculando os pesos para cada x_test através do kernel e normalizando o vetor
  w = lapply(x_test, eval(kernel), x_i = x_train, h = h) %>% unlist() %>% matrix(
    nrow = length(x_train), ncol = length(x_test))
  w = sweep(w, 2, colSums(w), FUN = '/')

  # matriz de covariaveis com relacao a y, levando em conta o intercepto
  covs = model.matrix(y_train ~ x_train)
  estims = matrix(ncol = 2, nrow = length(x_test))
  # computando os estimadores de beta0 e beta1 localmente para cada x do teste
  for(i in 1:length(x_test)){
    omega = diag(w[,i])
    estims[i, ] = solve(t(covs) %*% omega %*% covs) %*% (t(covs) %*% omega %*% y_train)
  }
  # retornando as estimativas
  colnames(estims) = c("beta0", "beta1")
  return(estims)
```

```
}

fit_loc_lm = function(x, x_train, y_train, h, kernel){
  estims = loc_lm(x_train, y_train, x, h, kernel)
  vars = cbind(rep(1, length(x)),
               x)
  preds = estims * vars
  return(rowSums(preds))
}
```

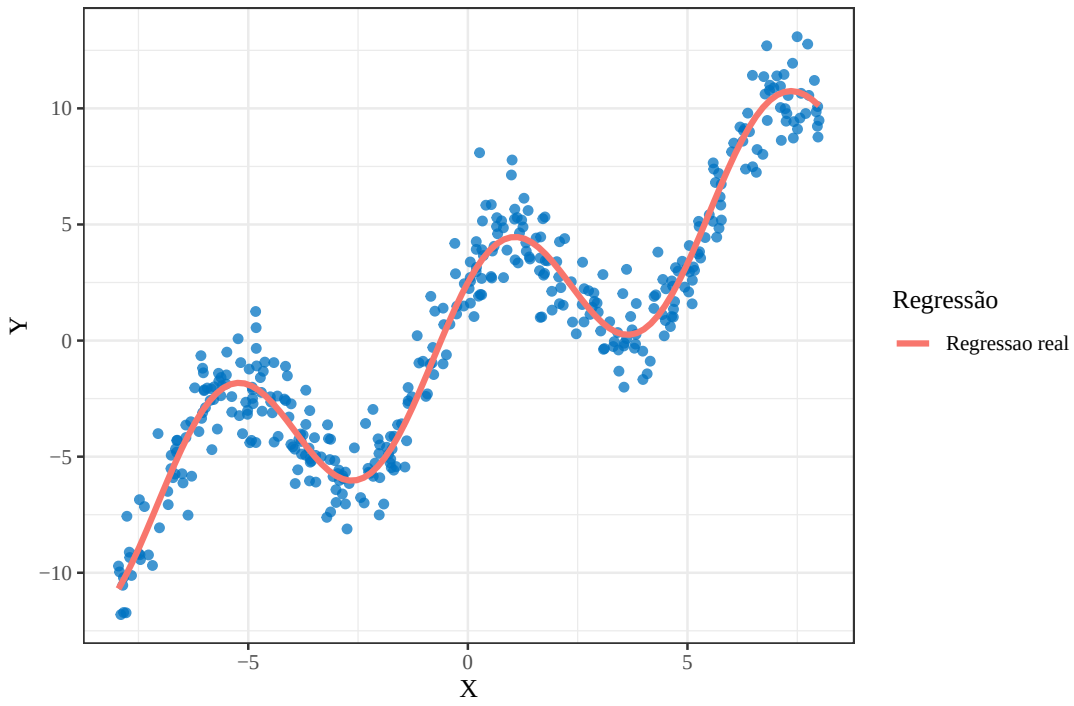
Simulando alguns dados utilizando senos e cossenos para formar um comportamento de onda:

```
# conjunto inteiro
n = 400
x_all = runif(n, -8, 8)
y_all = 2.5*cos(x_all) + 2.5*sin(x_all) + x_all + rnorm(n, sd = 1.25)

sim_data = data.frame(X = x_all, Y = y_all)
```

Formato do gráfico de dispersão:

```
# regressao real
reg_real = function(x){
  return(2.5*cos(x) + 2.5*sin(x) + x)
}

sim_data %>%
  ggplot(aes(x = X, y = Y))+
  geom_point(color = "#0073C2FF", alpha = 0.75)+
  labs(x = "X",
       y = "Y",
       title = "Gráfico de dispersão dos dados simulados com regressão real",
       colour = "Regressão")+
  stat_function(fun = reg_real,
                aes(colour = "Regressao real"), size = 1.25)+
  theme_bw()+
  theme(text = element_text(size = 11,
                            family ="serif"),
        plot.title = element_text(hjust = 0.5))
```

Gráfico de dispersão dos dados simulados com regressão real



Separando o conjunto de treinamento e teste e testando inicialmente para apenas $h = 0.2$, mostrando as 10 primeiras observações de teste:

```
set.seed(1250, sample.kind="Rounding")
```

```
## Warning in set.seed(1250, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
n_train = 300
id_train = sample(1:n, size = n_train, replace = F)

x_train = x_all[id_train]
x_test = x_all[-id_train]
y_train = y_all[id_train]


coefs = loc_lm(x_train, y_train, x_test, h = 0.2, kernel = "gauss_kernel")
coefs %>% as.data.frame() %>% head(10)
```

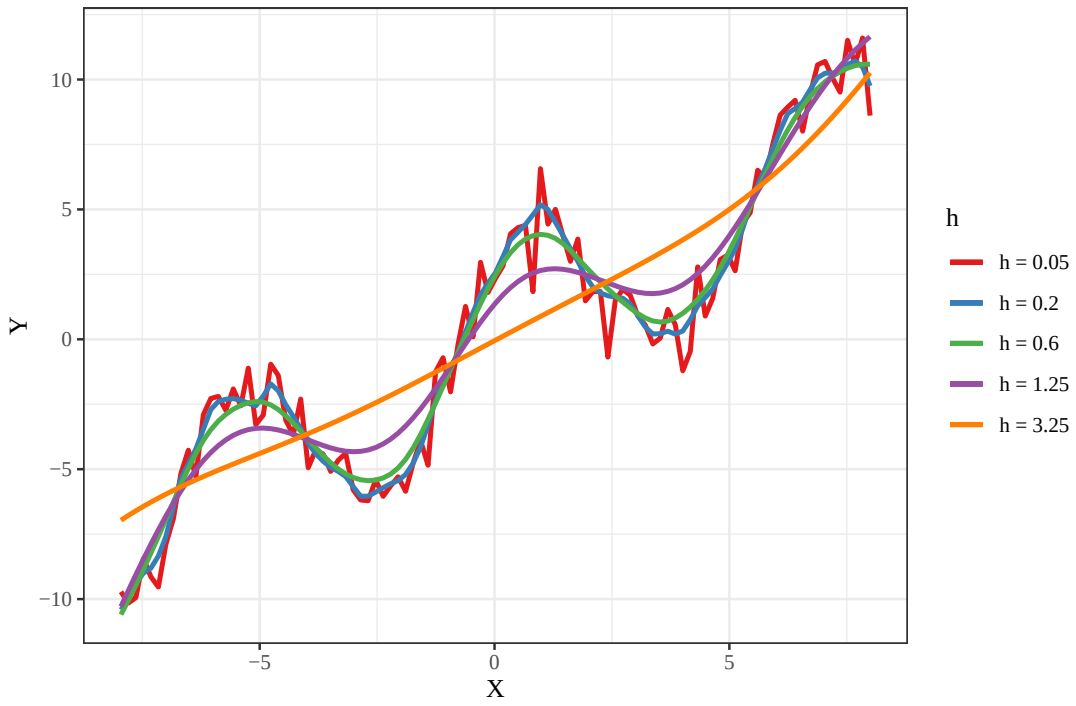
beta0	beta1
16.0928881	3.2032229
5.2523858	-1.5489638
0.2425503	1.3719024
-12.4953789	3.1825520
-14.6364856	-2.9807853
-15.9261305	-3.0168966

beta0	beta1
2.9824889	2.2116348
-11.3065297	-1.8128688
10.2265828	0.0033342
2.3611532	4.5627060

Podemos analisar graficamente para diferentes valores de h com o kernel gaussiano fixado, utilizando os dados de treinamento no range dado:

```
# testando para diferentes h's
h = c(0.05, 0.2, 0.6, 1.25, 3.25)

p1 = sim_data %>%
  ggplot(aes(x = X, y = Y))+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = h[1], kernel = gauss_kernel),
                aes(colour = "h = 0.05"), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = h[2], kernel = gauss_kernel),
                aes(colour = paste0("h = ", h[2])), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = h[3], kernel = gauss_kernel),
                aes(colour = paste0("h = ", h[3])), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = h[4], kernel = gauss_kernel),
                aes(colour = paste0("h = ", h[4])), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = h[5], kernel = gauss_kernel),
                aes(colour = paste0("h = ", h[5])), size = 1)+
  labs(x = "X",
       y = "Y",
       title = "Gráfico de dispersão dos dados simulados com as regressões estimadas",
       colour = "h")+
  theme_bw()+
  theme(text = element_text(size = 11,
                            family ="serif"),
        plot.title = element_text(hjust = 0.5)) +
  scale_colour_brewer(palette = "Set1")


p1
```

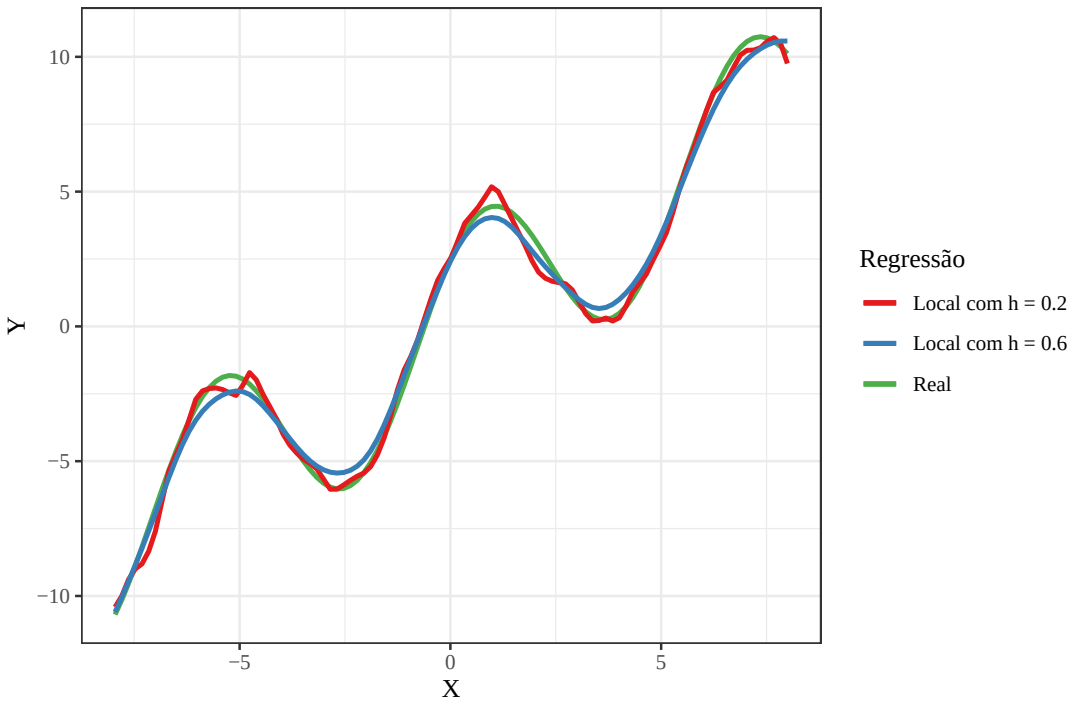
Gráfico de dispersão dos dados simulados com as regressões estimadas



Se compararmos com a regressão verdadeira apenas usando $h = 0.2$ e $h = 0.6$, teremos:

```
sim_data %>%
  ggplot(aes(x = X, y = Y))+
  labs(x = "X",
       y = "Y",
       title = "Gráfico de dispersão dos dados simulados com regressão real e local",
       colour = "Regressão")+
  stat_function(fun = reg_real,
                aes(colour = "Real"), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = 0.2, kernel = gauss_kernel),
                aes(colour = paste0("Local com h = 0.2")), size = 1)+
  stat_function(fun = fit_loc_lm, args = list(x_train = x_train,
                                              y_train = y_train,
                                              h = 0.6, kernel = gauss_kernel),
                aes(colour = paste0("Local com h = 0.6")), size = 1)+
  theme_bw()+
  theme(text = element_text(size = 11,
                            family ="serif"),
        plot.title = element_text(hjust = 0.5))+
  scale_colour_brewer(palette = "Set1")
```

Gráfico de dispersão dos dados simulados com regressão real e local



Exercício 2:

Lendo os dados e transformando os textos em matriz documento-texto:

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(SnowballC)
av_texto = read.csv("/home/kuben/estatistica_UFSCAR/Mineracao de dados/listas/lista_2/TMDb_updated.CSV"
```