

Proiect de laborator

NICULESCU Roberta-Carla, MACOVEI Bogdan

Ianuarie 2020

Scop

Am implementat pachetul "lrfs" (Linear regression from scratch) care contine toti algoritmii prezentati in partea de curs si documentati si in cadrul pachetului. Scopul acestui document este acela de a prezenta modul in care ceea ce am dezvoltat poate fi utilizat in cadrul unui proiect real.

Obiective

Vom utiliza, pentru demonstratie, setul de date "House Sale in King County, USA" de pe Kaggle, referinta <https://www.kaggle.com/harlfoxem/housesalesprediction>. Pentru a trece prin toate etapele unui algoritm de invatare automata, avem in vedere determinarea coeficientilor semnificativi in antrenare, normalizarea setului de date, antrenarea si evaluarea ulterioara.

Proiectul pe care l-am dezvoltat este scris pe pasi foarte mici, pentru a justifica fiecare operatie pe care o aplicam.

1. Preluarea setului de date

Vom utiliza functia **read.csv** din **R** pentru preluarea setului de date, vom inspecta coloanele si vom imparti coloanele in *feature vectori* si *label*.

```
df <- read.csv(file = "C:/Users/bogda/Desktop/kc_house_Data.csv",
                stringsAsFactors = FALSE,
                header = TRUE,
                sep = ",")  
  
head(df)  
  
##          id      date   price bedrooms bathrooms sqft_living  
## 1 7129300520 20141013T000000 221900        3     1.00      1180  
## 2 6414100192 20141209T000000 538000        3     2.25      2570  
## 3 5631500400 20150225T000000 180000        2     1.00       770  
## 4 2487200875 20141209T000000 604000        4     3.00      1960  
## 5 1954400510 20150218T000000 510000        3     2.00      1680  
## 6 7237550310 20140512T000000 1225000       4     4.50      5420  
##    sqft_lot floors waterfront view condition grade sqft_above sqft_basement  
## 1      5650     1          0    0        3    7      1180             0  
## 2      7242     2          0    0        3    7      2170            400  
## 3     10000     1          0    0        3    6       770             0  
## 4      5000     1          0    0        5    7      1050            910  
## 5      8080     1          0    0        3    8      1680             0  
## 6     101930     1          0    0        3   11      3890         1530  
##    yr_built yr_renovated zipcode      lat      long sqft_living15 sqft_lot15  
## 1      1955                 0 98178 47.5112 -122.257        1340      5650  
## 2      1951               1991 98125 47.7210 -122.319        1690      7639
```

```

## 3      1933          0  98028 47.7379 -122.233        2720      8062
## 4      1965          0  98136 47.5208 -122.393        1360      5000
## 5      1987          0  98074 47.6168 -122.045        1800      7503
## 6      2001          0  98053 47.6561 -122.005        4760    101930

```

Vom retine in x toate coloanele, cu exceptia coloanei **price** pe care trebuie sa o prezicem, si care va fi retinuta in y .

```

x_init <- df[c(1, 4:dim(df)[2])]      # toate coloanele, mai putin a treia si data
names(x_init) <- NULL                  # eliminam headerul
y_init <- df[3]$price                  # ce avem de prezis

```

Vom scala datele (fiecare valoare va fi intre 0 si 1) pentru a nu avea erori din cauza ordinului de marime la antrenare.

```

x_init <- scale(x_init)
y_init <- scale(y_init)
X_init <- cbind(1, x_init)

```

2. Analiza corelatiei

Propunem aici doua metode, o metoda bazata pe corelatia statistica si inca una bazata pe Lasso Regression, cu modificarea parametrului λ . Pentru masura statistica, formula utilizata este

$$\text{corr}(x, y) := \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}$$

unde \bar{x} reprezinta media lui x . Prin urmare, definim:

```

corr <- function(a, b) {
  m1 <- sum((a - mean(a)) * (b - mean(b)))
  m2 <- sqrt(sum((a - mean(a))^2) * sum((b - mean(b))^2))
  return(m1 / m2)
}

```

In plus, utila ar fi si existenta unei functii pentru varianta, care se calculeaza ca fiind corelatia la patrat, deci:

```

variance <- function(a, b) {
  return(corr(a, b) ^ 2)
}

```

Aplicam corelatia pe setul nostru de date, i.e. determinam corelatia dintre fiecare coloana din x_init cu y_init .

```

for (i in 1:dim(x_init)[2]) {
  correlation <- corr(x_init[, i], y_init)
  print(paste("Corelatia", i, "este ", correlation))
}

```

```

## [1] "Corelatia 1 este -0.0167621966144451"
## [1] "Corelatia 2 este 0.308349598145638"
## [1] "Corelatia 3 este 0.525137505413962"
## [1] "Corelatia 4 este 0.7020350546118"
## [1] "Corelatia 5 este 0.0896608605871001"
## [1] "Corelatia 6 este 0.256793887550718"
## [1] "Corelatia 7 este 0.266369434030602"
## [1] "Corelatia 8 este 0.397293488294504"
## [1] "Corelatia 9 este 0.0363617891289976"
## [1] "Corelatia 10 este 0.667434256020237"
## [1] "Corelatia 11 este 0.605567298356078"
## [1] "Corelatia 12 este 0.323816020711984"
## [1] "Corelatia 13 este 0.0540115314947927"
## [1] "Corelatia 14 este 0.126433793440893"
## [1] "Corelatia 15 este -0.0532028542983256"
## [1] "Corelatia 16 este 0.307003479995219"
## [1] "Corelatia 17 este 0.0216262410393067"
## [1] "Corelatia 18 este 0.585378903579568"
## [1] "Corelatia 19 este 0.082447152519486"

```

Aplicam, acum, algoritmul Lasso Regression cu diferite valori pentru λ , pentru a vedea ce parametri devin 0 (sau apropiati de 0) pe masura ce valoarea lui λ creste.

```

library(lrfs)

for (lambda_value in c(1e-3)) { # nu sunt incluse toate valorile din cauza duratei lungi de executare
  # sunt insa trecute valorile in tabelul de mai jos
  w_local <- fit_lasso_regression(X_init,
                                    y_init,
                                    learning_rate = 1e-6,
                                    model_train_error = 1e-2,
                                    lambda = lambda_value)

  print(w_local)
}

## [1] 0.000000000 -0.010096694 -0.090603955  0.086327695  0.222812716
## [6] 0.013751731  0.009952637  0.137307425  0.110694260  0.046541309
## [11] 0.307361786  0.207580279  0.073589424 -0.209786614  0.021499161
## [16] -0.084857602  0.227349047 -0.081636375  0.040391101 -0.029489547

```

Pentru a fi mai usor de urmarit, avem urmatorul tabel, unde am retinut doar acele feature-uri cu o corelatie statistica semnificativa:

Cor	bedr	bathr	sqftliv	floor	wtrf	view	grade	sqftabv	sqftbas	sqftliv15
Stat	0.308	0.525	0.702	0.256	0.266	0.397	0.667	0.605	0.323	0.585
L1e-3	-0.090	0.086	0.222	0.009	0.137	0.110	0.307	0.207	0.073	0.040
L1e-2	-0.090	0.086	0.223	0.009	0.137	0.110	0.307	0.206	0.073	0.040
L1e-1	-0.090	0.086	0.232	0.009	0.137	0.110	0.307	0.199	0.069	0.040
L1	-0.090	0.086	0.316	0.009	0.137	0.110	0.307	0.123	0.028	0.040
L10	-0.090	0.086	0.375	0.009	0.137	0.110	0.307	0.069	0.000	0.040

Pentru a putea construi modelul de invatare automata, trebuie sa alegem acei parametri care sunt cei mai semnificativi (si cat mai putini, pentru a nu avea modele complexe).

De exemplu, relevanta cea mai mare o are **sqftliv**, corelatia statistica de 0.702 si o evidenta crestere ca semnificatie in algoritmul *lasso* pe masura ce constanta λ creste. Un alt parametru important este *grade*.

Urmatorul parametru relevant statistic este **sqftabv**, dar observam ca relevanta acestuia scade pe masura ce algoritmul Lasso maresteste constanta λ , astfel ca vom mai lua in seama doar feature-ul **sqftliv15**.

Selectia finala a parametrilor utilizati este:

```
# Regresia cu un parametru
x1 <- scale(df[6]$sqft_living)
y1 <- scale(df[3]$price)
X1 <- cbind(1, x1)

# Regresia cu doi parametri
x2 <- df[c(6, 12)]
names(x2) <- NULL
x2 <- scale(x2)
y2 <- scale(df[3]$price)
X2 <- cbind(1, x2)

# Regresia cu trei parametri
x3 <- df[c(6, 12, 21)]
names(x3) <- NULL
x3 <- scale(x3)
y3 <- scale(df[3]$price)
X3 <- cbind(1, x3)
```

3. Regresia liniara simpla

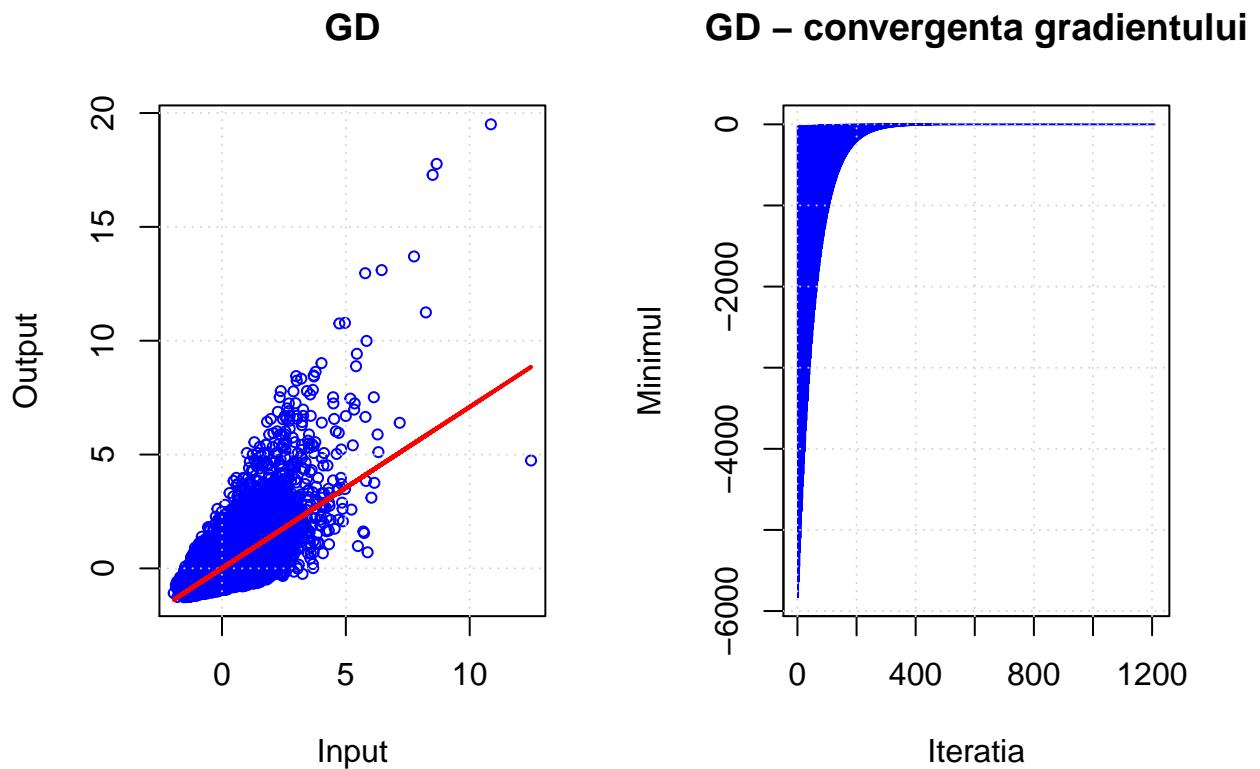
Vom utiliza toti algoritmii implementati, cu setarea *verbose=TRUE* pentru a obtine informatii din timpul antrenarii. Initial, vom imparti datele in date de antrenare si date de testare.

```
train_indices = sample(1:dim(X1)[1], 0.75 * dim(X1)[1])
test_indices = setdiff(1:dim(X1)[1], train_indices)

X1_train = X1[train_indices, ]
y1_train = y1[train_indices]
X1_test = X1[test_indices, ]
y1_test = y1[test_indices]
```

3.1 Antrenare cu Gradient Descent

```
w = fit_linear_regression(X1_train,
                           y1_train,
                           learning_rate = 1e-6,
                           model_train_error = 1e-5,
                           verbose = TRUE)
```



```
y1_pred = w[1] + w[2] * X1_test[, 2]
print(paste("Corelatie", corr(y1_test, y1_pred)))

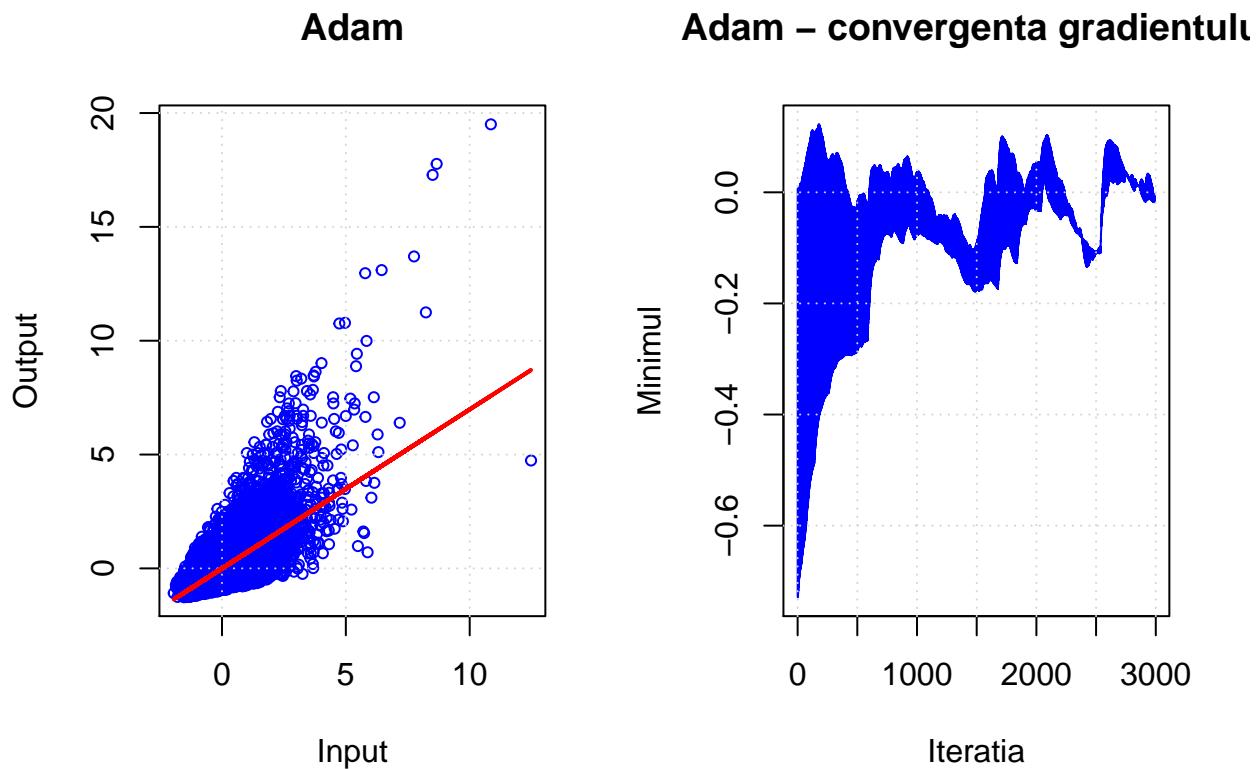
## [1] "Corelatie 0.700010885501168"

print(paste("Varianta", variance(y1_test, y1_pred)))

## [1] "Varianta 0.490015239820129"
```

3.2 Antrenare cu ADAM

```
w = fit_linear_regression(X1_train,
                          y1_train,
                          optimizer="adam",
                          model_train_error = 1e-2,
                          verbose = TRUE)
```



```

y1_pred = w[1] + w[2] * X1_test[, 2]
print(paste("Corelatie", corr(y1_test, y1_pred)))

## [1] "Corelatie 0.700010885501168"

print(paste("Varianta", variance(y1_test, y1_pred)))

## [1] "Varianta 0.490015239820129"

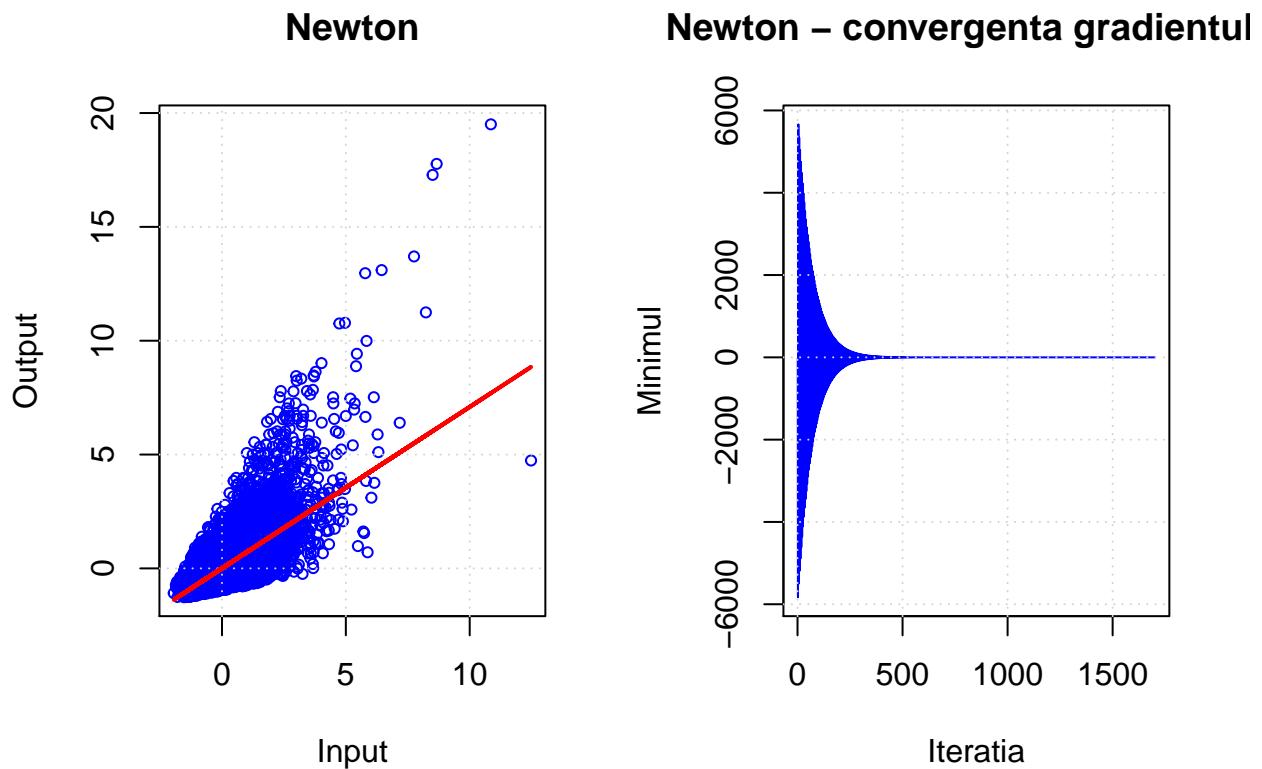
```

3.3 Antrenare cu Newton cu Hessiana fixata

```

w = fit_linear_regression(X1_train,
                          y1_train,
                          optimizer="newton",
                          newton_lambda = 60,
                          model_train_error = 1e-7,
                          verbose = TRUE)

```



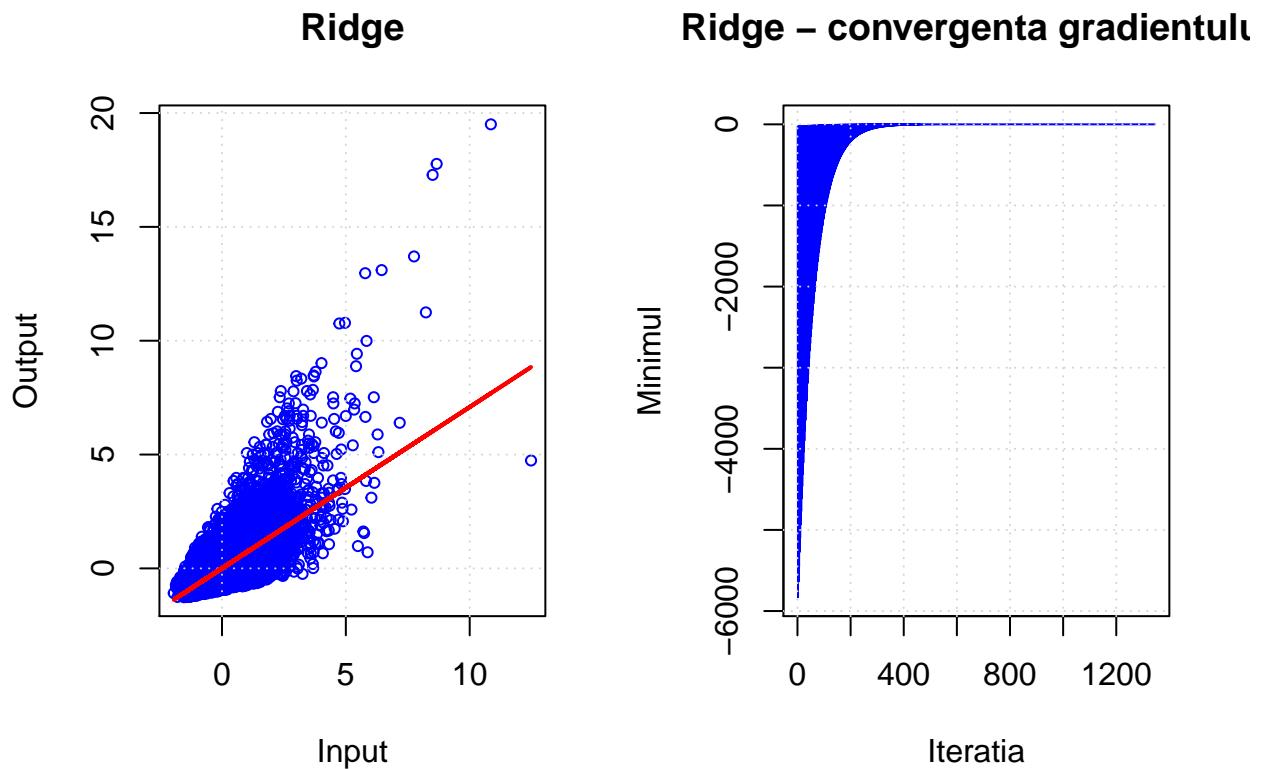
```
y1_pred = w[1] + w[2] * X1_test[, 2]
print(paste("Corelatie", corr(y1_test, y1_pred)))
```

```
## [1] "Corelatie 0.700010885501167"
print(paste("Varianta", variance(y1_test, y1_pred)))
```

```
## [1] "Varianta 0.490015239820129"
```

3.4 Antrenare cu Ridge

```
w = fit_ridge_regression(X1_train,
                          y1_train,
                          learning_rate = 1e-6,
                          model_train_error = 1e-6,
                          verbose = TRUE)
```



```

y1_pred = w[1] + w[2] * X1_test[, 2]

print(paste("Corelatie", corr(y1_test, y1_pred)))

## [1] "Corelatie 0.700010885501168"

print(paste("Varianta", variance(y1_test, y1_pred)))

## [1] "Varianta 0.490015239820129"

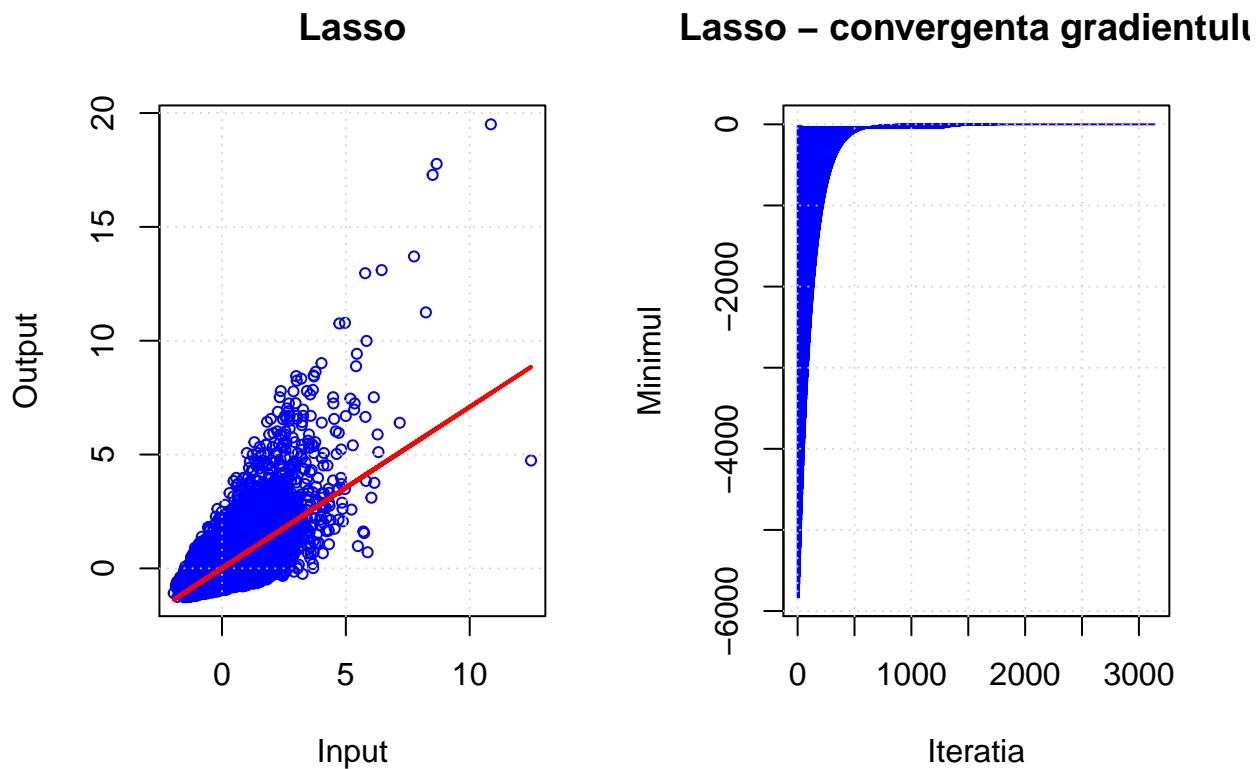
```

3.5 Antrenare cu Lasso

```

w = fit_lasso_regression(X1_train,
                         y1_train,
                         learning_rate = 1e-6,
                         lambda = 1e-1,
                         verbose = TRUE)

```



```
y1_pred = w[1] + w[2] * X1_test[, 2]

print(paste("Corelatie", corr(y1_test, y1_pred)))

## [1] "Corelatie 0.700010885501167"

print(paste("Varianta", variance(y1_test, y1_pred)))

## [1] "Varianta 0.490015239820129"
```

4. Regresia liniara multipla

Vom antrena metodele cele mai rapide (Adam si Newton) pentru regresiile cu 2, respectiv 3 parametri. Vom imparti, din nou, setul de date:

```
X2_train = X2[train_indices, ]
y2_train = y2[train_indices]
X2_test = X2[test_indices, ]
y2_test = y2[test_indices, ]

X3_train = X3[train_indices, ]
y3_train = y3[train_indices]
X3_test = X3[test_indices, ]
y3_test = y3[test_indices]
```

4.1 Regresia cu doi parametri

4.1.1 Metoda Adam

```
w = fit_linear_regression(X2_train,
                          y2_train,
                          optimizer = "adam",
                          model_train_error = 1e-1)

y2_pred = w[1] + w[2] * X2_test[, 2]

print(paste("Corelatie", corr(y2_test, y2_pred)))

## [1] "Corelatie 0.700010885501168"

print(paste("Varianta", variance(y2_test, y2_pred)))

## [1] "Varianta 0.490015239820129"
```

4.1.2 Metoda Newton

```
w = fit_linear_regression(X2_train,
                          y2_train,
                          optimizer = "newton",
                          newton_lambda = 60,
                          model_train_error = 1e-6)

y2_pred = w[1] + w[2] * X2_test[, 2]

print(paste("Corelatie", corr(y2_test, y2_pred)))

## [1] "Corelatie 0.700010885501168"

print(paste("Varianta", variance(y2_test, y2_pred)))

## [1] "Varianta 0.490015239820129"
```

4.2 Regresia cu trei parametri in abordarea metodei Newton

```
w = fit_linear_regression(X3_train,
                          y3_train,
                          optimizer = "newton",
                          newton_lambda = 100,
                          model_train_error = 1e-10)

y3_pred = w[1] + w[2] * X3_test[, 2]

print(paste("Corelatie", corr(y3_test, y3_pred)))
```

```
## [1] "Corelatie 0.700010885501168"  
  
print(paste("Varianta", variance(y3_test, y3_pred)))  
  
## [1] "Varianta 0.490015239820129"
```

Concluzii

Observam ca rezultatele obtinute prin implementarea modelului de regresie in trei abordari (fiecare abordare cu numar diferit de parametri) sunt similare, astfel ca, avand in vedere scoruri de complexitate, precum AIC si BIC, preferam modelele mai simple, in acest caz modelele antrenate pentru regresia simpla fiind varianta adevarata.

Ca observatie, rezultatele obtinute sunt comparabile cu kernel-urile disponibile pe Kaggle pentru aceasta competitie, corelatia de 0.7 obtinuta fiind si cea obtinuta de cea mai buna solutie trimisa.