Machine Learning

Homework 1

RAPPAPORT Gabrielle

September 14, 2019

1 Question 1

We are given the following constraints :

$$\forall i \in \{1, ..., n\}, a_i^T x \geq b_i$$

We assume that :

$$\exists M : a_i^T x \geq M \text{ and } \forall i \in \{1, ..., n\}, b_i \geq M$$

We want to model the requirement that at least k of the constraints are satisfied.
Let's introduce the binary variable :

$$\forall i \in \{1, ..., n\}, y_i \in \{0, 1\} \text{ such as } \sum_{i=1}^{n} y_i \leq k$$

We can reformulate our constraints as :

$$\forall i \in \{1, ..., n\}, a_i^T x \geq b_i(1 - y_i) + M.y_i$$

This means that :

$$y_i = 0 \Rightarrow a_i^T x \geq b_i, \text{the } i^e \text{ constraint is satisfied}$$
$$y_i = 1 \Rightarrow a_i^T x \geq M, \text{the } i^e \text{ constraint is not satisfied}$$

Finally, we obtain the following model :

$$a_i^T x \geq b_i(1 - y_i) + M.y_i, \forall i \in \{1, ..., n\}$$

$$\sum_{i=1}^{n} \leq k$$

$$y_i \in \{0, 1\}$$

2 Question 2

2.1 Question a

We want to solve the classical least square problem :

$$\min \sum_{i=1}^{n}(y_i - \beta_0 + \beta^T.x_i)^2$$

We name β_0^* and β^* the optimal solutions.
To solve this problem, we will use the matrix representation. We name :

$$\overline{\beta} = \begin{pmatrix} \beta_0 * \\ \beta^* \end{pmatrix} \text{ and } \overline{X} = \begin{pmatrix} 1 & x_1 & \cdots & x_d \end{pmatrix}$$

Which give us the equivalent problem :

$$\begin{aligned} \sum_{i=1}^{n}\left(y_i - \beta_0 - \beta^\top x_i\right)^2 &= \left\|y - \overline{X}.\overline{\beta}\right\|_2^2 \\ &= (y - \overline{X\beta})^\top (y - \overline{X\beta}) \end{aligned}$$

To find the minimum we find B which cancels the derivative.

$$\begin{aligned} &\frac{\partial}{\partial \overline{\beta}}\left\{(y - \overline{X\beta})^\top (y - \overline{X\beta})\right\} \\ =&\frac{\partial}{\partial \overline{\beta}}\left\{\left(y^\top - (\overline{X\beta})^\top\right)(y - \overline{X\beta})\right\} \\ =&\frac{\partial}{\partial \overline{\beta}}\left\{y^\top y - y^\top \overline{X}_{\overline{\beta}} - (\overline{X\beta})^\top y + \overline{\beta}^\top \overline{X}^\top \overline{X\beta}\right\} \\ =& -2\overline{X}^\top y + 2\overline{X}^\top \overline{X\beta} \\ =& -2\overline{X}^\top (y - \overline{X\beta}) \end{aligned}$$

Then :

$$-2\overline{X}^\top (y - \overline{X\beta}) = 0 \Leftrightarrow \beta = \left(\overline{X}^\top \overline{X}\right)^{-1} \overline{X}^T y$$

2.2 Question b

We want to solve the following problem effectively :

$$\min \sum_{n=1}^{m}\left|y_i - \beta_0 - \beta^\top x_i\right|$$

Since we are adding positive elements, we know that :

$$\min \sum_{n=1}^{m}\left|y_i - \beta_0 - \beta^\top x_i\right| = \sum_{i=1}^{m} \min\left|y_i - \beta_0 - \beta^\top x_i\right|$$

We know that :

$$\min \; \left| y_i - \beta_0 - \beta^\top x_i \right| \Leftrightarrow$$

$$\begin{aligned} \min \; & z_i \\ \text{s.t.} \; & \left| \begin{array}{l} z_i \geq y_i - \beta_0 - \beta^\top x_i \\ z_i \geq -y_i + \beta_0 + \beta^\top x_i \end{array} \right. \end{aligned}$$

This finally gives us the linear program, which is the most efficient way to solve the problem :

$$\begin{aligned} \min \sum_{i=1}^{n} & z_i \\ \text{s.t.} \; & \left| \begin{array}{l} z_i \geq y_i - \beta_0 - \beta^\top x_i, \, \forall i \\ z_i \geq -y_i + \beta_0 + \beta^\top x_i, \, \forall i \end{array} \right. \end{aligned}$$

2.3 Question c

In this question, we want to model that at most k of the d coefficient $\beta_1, ..., \beta_d$ have non zero values.
To do so, we will use the *big-M method*. We introduce, a constant M big enough and new **binary** variables a_i such as :

$$\beta_i \neq 0 \Rightarrow a_i = 0$$

We obtain the following linear problem :

$$\begin{aligned} \min \sum_{i=1}^{n} & z_i \\ \text{s.t.} \; & \left| \begin{array}{rl} z_i & \geq y_i - \beta_0 - \beta^\top x_i, \, \forall i \\ z_i & \geq -y_i + \beta_0 + \beta^\top x_i, \, \forall i \\ \beta_i & \leq M.a_i \\ \beta_i & \geq -M.a_i \\ \sum_{i=1}^{d} a_i & \leq k \\ a_i & \in \{0,1\}, \, \forall i \end{array} \right. \end{aligned}$$

2.4 Question d

We want to formulate the *Robust Linear Regression Problem* as an integer programming problem:

$$\min \text{ Median}(|y_1 - \beta_0 - \beta^T x_1|, ..., |y_n - \beta_0 - \beta^T x_n|)$$
$$\text{where } n = 2k + 1$$

Since the number of elements $|y_i - \beta_0 - \beta^T x_i|$ is odd, the median is $|y_{k+1} - \beta_0 - \beta^T x_{k+1}|$, therefore we want to minimize $c = |y_{k+1} - \beta_0 - \beta^T x_{k+1}|$.
It means that we want to minimize c such as c is bigger than the first $k + 1$ elements.
This gives us the following linear program :

$$\begin{aligned} &\min c \\ &\text{s.t.} \left| \begin{array}{ll} c & \geq h_i - M(1 - z_i), \forall i \\ h_i & \geq y_i - \beta_0 - \beta^T x_i, \forall i \\ h_i & \geq -y_i + \beta_0 + \beta^T x_i, \forall i \\ \sum_{i=1}^n z_i & = k \\ z_i & \in \{0,1\}, \forall i \end{array} \right. \end{aligned}$$

ATTENTION : k+1 —> Les k+1 contraintes sont vérifiées.

3 Question 3

September 14, 2019

1 Exercice 3

[2]:
```
#import Pkg
#Pkg.add("JuMP")
#Pkg.add("Gurobi")

using JuMP, Gurobi, DataFrames, CSV, Random, LinearAlgebra
```

[3]:
```
#using Pkg
#Pkg.add("PyPlot")
using PyPlot
```

2 Question a

- Model for the l0-regularization

[4]:
```
function regularized_regression_l0(y, X, ρ)

    M = 10000

    n,m = size(X)
    model = Model(solver = GurobiSolver(TimeLimit=45))

    @variable(model, β[1:m])
    @variable(model, z[1:m], Bin)

    @objective(model, Min, sum((y - X*β).^2) + ρ*sum(z))

    for i in 1:m
        @constraint(model, β[i] <= M * z[i])
        @constraint(model, (-β[i]) <= M * z[i])
    end

    sol = solve(model)

    return (getobjectivevalue(model), getvalue(β))

end
```

[4]: regularized_regression_l0 (generic function with 1 method)

- Model for the l1-regularization

[6]:
```
function regularized_regression_l1(y, X, ρ)

    M = 10000

    n,m = size(X)
    model = Model(solver = GurobiSolver(TimeLimit=45))

    @variable(model, β[1:m])
    @variable(model, z[1:m])

    @objective(model, Min, sum((y - X*β).^2) + ρ*sum(z))

    for i in 1:m
        @constraint(model, (z[i]) >= β[i])
        @constraint(model, (z[i]) >= - β[i])
    end

    sol = solve(model)

    return (getobjectivevalue(model), getvalue(β))

end
```

[6]: regularized_regression_l1 (generic function with 1 method)

- Model for the l2-regularization

[7]:
```
function regularized_regression_l2(y, X, ρ)

    M = 10000

    n,m = size(X)
    model = Model(solver = GurobiSolver(TimeLimit=45))

    @variable(model, β[1:m])

    @objective(model, Min, sum((y - X*β).^2) + ρ*sum(β.^2))

    sol = solve(model)

    return (getobjectivevalue(model), getvalue(β))

end
```

[7]: regularized_regression_l2 (generic function with 1 method)

2.1 Question b

[ ]:
```
sparseX2 = CSV.read("sparseX2.csv")
sparseY2 = CSV.read("sparseY2.csv")
```

Creation of the validation, training and testing sets.

[9]:
```
function split_data(X, y, val, test)
    n = size(X, 1)
    index = shuffle([i for i in 1:n])

    size_validation = floor(Int,val*n)
    size_test = floor(Int,(val+test)*n)

    ind_validation = index[1:size_validation]
    ind_test = index[size_validation+1:size_test]
    ind_train = index[size_test+1:n]

    X_validation = X[ind_validation, :]
    X_test = X[ind_test, :]
    X_train = X[ind_train, :]

    y_validation = y[ind_validation, :]
    y_test = y[ind_test, :]
    y_train = y[ind_train, :]

    return (
        convert(Matrix,X_validation),
        convert(Matrix,X_test),
        convert(Matrix,X_train),
        convert(Matrix, y_validation),
        convert(Matrix,y_test),
        convert(Matrix,y_train)
        )

end
```

[9]: split_data (generic function with 1 method)

[ ]:
```
(
    X_validation,
    X_test,
    X_train,
    y_validation,
    y_test,
    y_train
) = split_data(sparseX2, sparseY2, 0.25, 0.25)
```

We choose ρ as the value that gives the best mean squared prediction error on the validation set.

[12]:
```
function mean_square_error(y_actual, y_predicted)
    return sum((y_actual - y_predicted).^2)
end
```

[12]: mean_square_error (generic function with 1 method)

[13]:
```
function compute_error_l0(ρ, X, y)
    objective, β = regularized_regression_l0(y_train, X_train, ρ)
    error = mean_square_error(y, X * β)
    return error
end
```

[13]: compute_error_l0 (generic function with 1 method)

[14]:
```
function compute_error_l1(ρ, X, y)
    objective, β = regularized_regression_l1(y_train, X_train, ρ)
    error = mean_square_error(y, X * β)
    return error
end
```

[14]: compute_error_l1 (generic function with 1 method)

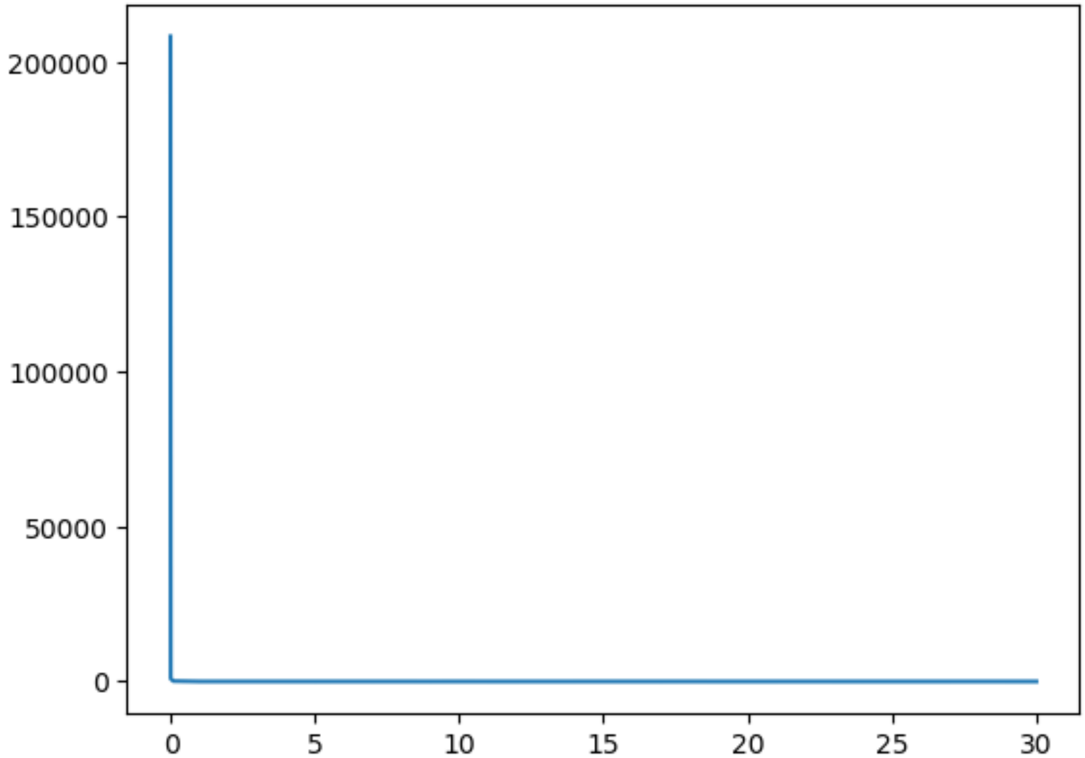
[15]:
```
function compute_error_l2(ρ, X, y)
    objective, β = regularized_regression_l2(y_train, X_train, ρ)
    error = mean_square_error(y, X * β)
    return error
end
```

[15]: compute_error_l2 (generic function with 1 method)

[ ]:
```
ρ_l0 = vcat([0.001, 0.01, 0.1], [i for i=1:30])
errors_l0 = zeros(length(ρ_l0))

for i in 1:length(errors_l0)
    errors_l0[i] = compute_error_l0(ρ_l0[i], X_validation, y_validation)
end
```

[17]:
```
plot(ρ_l0, errors_l0)
```



[17]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x14212a748>
```

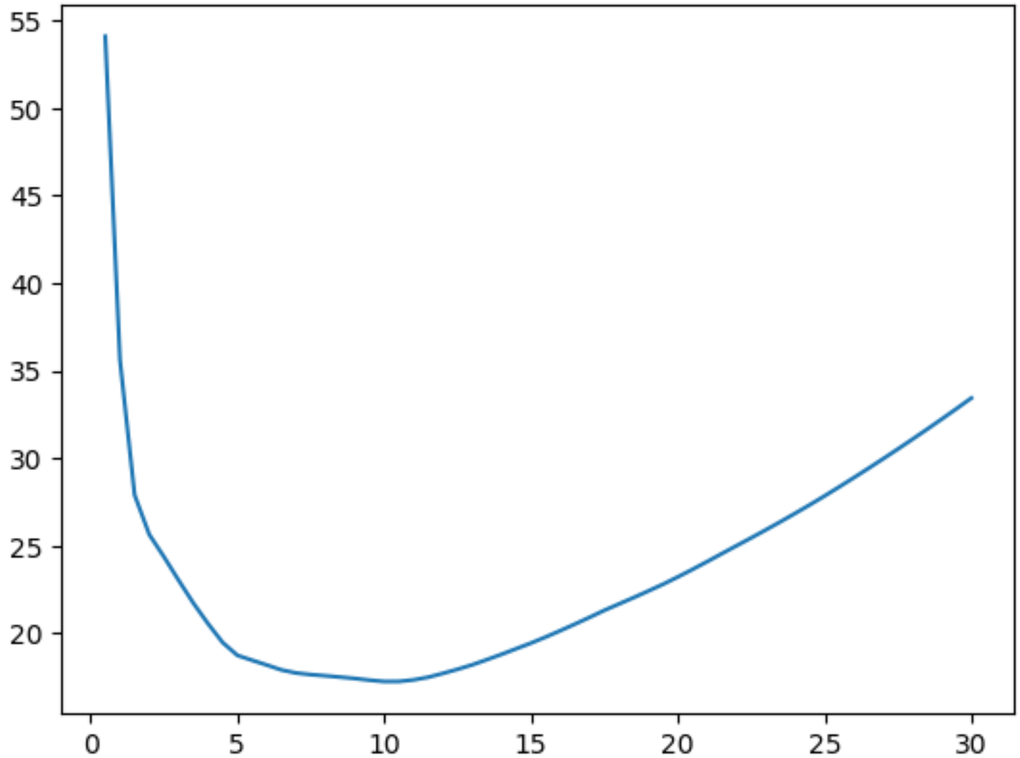
[41]:
```
ρ_optimal_l0 = ρ_l0[argmin(errors_l0)]
```

[41]:
```
30.0
```

[ ]:
```
ρ_l1 = [i/2 for i=1:60]
errors_l1 = zeros(length(ρ_l1))

for i in 1:length(errors_l1)
    errors_l1[i] = compute_error_l1(ρ_l1[i], X_validation, y_validation)
end
```

[20]:
```
plot(ρ_l1, errors_l1)
```



[20]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject <matplotlib.lines.Line2D object at 0x1428417b8>
```

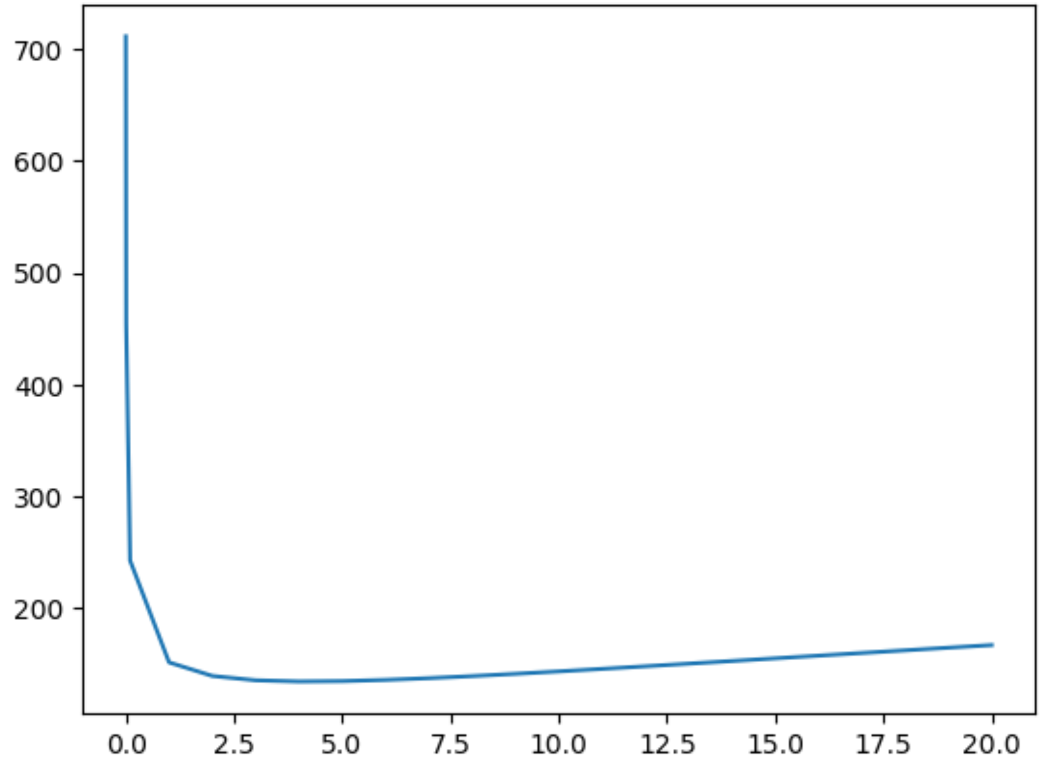
[42]:
```
ρ_optimal_l1 = ρ_l1[argmin(errors_l1)]
```

[42]:
```
10.0
```

[ ]:
```
ρ_l2 = vcat([0.001, 0.01, 0.1], [i for i=1:20])
errors_l2 = zeros(length(ρ_l2))

for i in 1:length(errors_l2)
    errors_l2[i] = compute_error_l2(ρ_l2[i], X_validation, y_validation)
end
```

[28]:
```
plot(ρ_l2, errors_l2)
```



[28]:
```
1-element Array{PyCall.PyObject,1}:
 PyObject &lt;matplotlib.lines.Line2D object at 0x141a41f60&gt;
```

[43]:
```
ρ_optimal_l2 = ρ_l2[argmin(errors_l2)]
```

[43]:
```
4.0
```

Compute $||y - X\beta||2$ on the testing set using the β's from l0-regularized, l1-regularized linear regression, l2- regularized linear regression, and standard linear regression ($\rho = 0$)

[50]:
```
l0_error = compute_error_l0(ρ_optimal_l0, X_test, y_test)
```

```
Academic license - for non-commercial use only
Optimize a model with 200 rows, 200 columns and 400 nonzeros
Model has 5050 quadratic objective terms
Variable types: 100 continuous, 100 integer (100 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+04]
  Objective range  [3e+00, 3e+02]
  QObjective range [2e-02, 3e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [0e+00, 0e+00]
Found heuristic solution: objective 0.0000000
Presolve time: 0.00s
Presolved: 200 rows, 200 columns, 400 nonzeros
```

```
Presolved model has 5050 quadratic objective terms
Variable types: 100 continuous, 100 integer (100 binary)

Root relaxation: objective -9.524350e+02, 981 iterations, 0.03 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 -952.43503    0   78    0.00000 -952.43503      -     -    0s
H    0     0                       -309.6583805 -952.43503   208%     -    0s
H    0     0                       -413.9725003 -952.43503   130%     -    0s
     0     0 -952.43503    0   78 -413.97250 -952.43503   130%     -    0s
H    0     0                       -631.5297179 -952.43503  50.8%     -    0s
H    0     0                       -631.8148568 -952.43503  50.7%     -    0s
     0     2 -952.43503    0   78 -631.81486 -952.43503  50.7%     -    0s
H  744   552                       -632.3423572 -884.06346  39.8%  27.2    1s
H 2792  1173                       -632.3758008 -847.67096  34.0%  28.1    3s
H 2810  1186                       -632.4516075 -846.68895  33.9%  28.1    3s
H 2816  1190                       -633.1807952 -846.68895  33.7%  28.1    3s
  5002  2258 -725.62254   28   75 -633.18080 -828.40078  30.8%  28.1    5s
H 5971  2722                       -633.3912979 -822.42464  29.8%  28.0    5s
 12034  5283 -670.01060   32   72 -633.39130 -799.96555  26.3%  27.8   10s
H14090  6176                       -633.5924643 -794.95228  25.5%  28.0   12s
 17025  7198 -706.92109   28   77 -633.59246 -788.03072  24.4%  27.9   15s
 24809  9817 -741.81739   26   79 -633.59246 -774.31503  22.2%  27.5   20s
H27145 10533                       -633.6506843 -771.17807  21.7%  27.4   21s
 31627 11967 -714.26739   27   74 -633.65068 -766.26198  20.9%  27.3   25s
 39822 14334 -682.52802   28   69 -633.65068 -758.31175  19.7%  27.3   30s
H40251 14434                       -633.9829945 -757.89283  19.5%  27.3   30s
H40311 14393                       -634.6981061 -757.78877  19.4%  27.2   30s
 47551 16181 -706.87827   30   75 -634.69811 -751.93922  18.5%  27.1   35s
 54518 17681 -746.29602   27   78 -634.69811 -746.33075  17.6%  27.0   40s
 61880 19284 -650.50638   34   68 -634.69811 -741.44961  16.8%  26.9   45s

Explored 62113 nodes (1674274 simplex iterations) in 45.00 seconds
Thread count was 4 (of 4 available processors)

Solution count 10: -634.698 -633.983 -633.651 ... -631.815

Time limit reached
Best objective -6.346981061123e+02, best bound -7.413865505370e+02, gap 16.8093%

[U+250C] Warning: Not solved to optimality, status: UserLimit
[U+2514] @ JuMP /Users/gabriellerappaport/.julia/packages/JuMP/I7whV/src/solvers.
 ↪jl:212
```

[50]: 16.229156793039543

[53]:
```
l1_error = compute_error_l1(ρ_optimal_l1, X_test, y_test)
```

```
Academic license - for non-commercial use only
Optimize a model with 200 rows, 200 columns and 400 nonzeros
Model has 5050 quadratic objective terms
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [3e+00, 3e+02]
  QObjective range [2e-02, 3e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 100 rows and 0 columns
Presolve time: 0.00s
Presolved: 100 rows, 200 columns, 200 nonzeros
Presolved model has 5050 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 199
 AA' NZ     : 9.900e+03
 Factor NZ  : 1.219e+04
 Factor Ops : 1.092e+06 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   2.14000000e+06  0.00000000e+00  0.00e+00 2.12e+03  1.01e+06     0s
   1   2.07540565e+06 -9.51734680e+02  2.95e-07 2.12e-03  1.04e+04     0s
   2   2.20304958e+03 -9.51573018e+02  1.57e-10 1.10e-06  1.58e+01     0s
   3  -7.31306591e+02 -8.85413674e+02  2.99e-09 1.22e-07  7.71e-01     0s
   4  -8.25442948e+02 -8.40833167e+02  4.43e-10 2.05e-08  7.70e-02     0s
   5  -8.35758966e+02 -8.37264584e+02  1.84e-10 2.22e-09  7.53e-03     0s
   6  -8.36690413e+02 -8.36776719e+02  2.40e-10 2.70e-09  4.32e-04     0s
   7  -8.36734721e+02 -8.36739711e+02  2.97e-10 4.46e-09  2.49e-05     0s
   8  -8.36736224e+02 -8.36736451e+02  1.49e-10 1.39e-09  1.13e-06     0s
   9  -8.36736252e+02 -8.36736254e+02  1.53e-11 2.18e-10  5.73e-09     0s

Barrier solved model in 9 iterations and 0.02 seconds
Optimal objective -8.36736252e+02
```

[53]: 20.376884563437116

[47]:
```
l2_error = compute_error_l2(ρ_optimal_l2, X_test, y_test)
```

```
Academic license - for non-commercial use only
Optimize a model with 0 rows, 100 columns and 0 nonzeros
```

```
Model has 5050 quadratic objective terms
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [3e+00, 3e+02]
  QObjective range [2e-02, 3e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve time: 0.00s
Presolved: 0 rows, 100 columns, 0 nonzeros
Presolved model has 5050 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 199
 AA' NZ     : 4.851e+03
 Factor NZ  : 4.950e+03
 Factor Ops : 3.284e+05 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   0.00000000e+00  0.00000000e+00  0.00e+00 2.40e+02  0.00e+00     0s
   1  -2.79969317e+02 -2.54327433e+01  3.26e-08 2.00e+02  0.00e+00     0s
   2  -5.91843984e+02 -1.50106285e+02  4.64e-08 1.43e+02  0.00e+00     0s
   3  -8.22895664e+02 -4.23828169e+02  6.84e-08 7.68e+01  0.00e+00     0s
   4  -9.16833998e+02 -9.16833396e+02  9.32e-08 7.70e-05  0.00e+00     0s
   5  -9.16833985e+02 -9.16833985e+02  6.72e-14 7.72e-11  0.00e+00     0s

Barrier solved model in 5 iterations and 0.01 seconds
Optimal objective -9.16833985e+02
```

[47]: 180.90945992777353

[64]:
```
println(" l0 error: ", l0_error)
println(" l1 error: ", l1_error)
println(" l2 error: ", l2_error)
```

```
 l0 error: 16.229156793039543
 l1 error: 20.376884563437116
 l2 error: 180.90945992777353
```

rho = 3, error 42
ordinary least square : rho = 0
baseline :
avg(concat (train validation)