Constrained Optimization & KKT Conditions

CS164: OPTIMIZATION METHODS

Minerva Schools at KGI

Albion Krasniqi

November 2020

The assignment instructions are available here.

Part 1: KKT Conditions for Linear Programming

A linear program can be written in the form

$$\min_{x} c^T x \quad \text{subject to} \quad Ax \leq b,$$

for matrices $A \in \mathrm{R}^{mxn}, b \in \mathrm{R}^{mx1}$ *and* $c \in \mathrm{R}^{nx1}$. Write down the KKT conditions for this LP and explain why the optimal solution will always be on a vertex or facet.

The Langragian equation of the given LP is: $L(x, \lambda) = c^T x + \lambda^T (Ax - b)$, where λ is a vector composed of m multipliers ($\lambda \in \mathrm{R}^{1xm}$). The KKT conditions for this problem are:

- *The Stationarity constraint for minimization*: this constraint guarantees that the solution is optimal by ensuring that the sum of the gradients of the objective function and the weighted constraints equals zero

$$\nabla_x L(x, \lambda) = \nabla_x c^T x + \lambda^T \nabla_x (Ax - b) = 0$$

- *The Complementary slackness*: makes sure that only active constraints have non-zero multipliers

$$\lambda^T * g(x) = \lambda^T * (Ax - b) = 0$$

- *Primal Feasibility*: $g(x) = (Ax - b) \leq 0$
- *Dual Feasibility*: $\lambda \geq 0$

The complementary slackness condition indicates that we can have two possible cases, which are:

- The first case, $Ax - b = 0 \rightarrow Ax = b$. From that, we can say that when the constraints are active, and the solution lies on the facets or vertices.
- The second case, $\lambda^T = 0$, all the multipliers are zero ($\lambda^T = 0$), it means that all the constraints are inactive (equivalent to saying that we are dealing with an unconstrained optimization). But if that is the case, then from the stationarity constraint condition, we infer that $\nabla_x c^T x \rightarrow c^T = 0$. Then the solution would be only the trivial case.

Part 2: Expressing l_1 and l_∞ as Linear Program Problems

Show how the l_1 and l_∞ regression problems can be expressed as linear programs, by defining slack variables and inequality constraints as needed.

L_1 norm (Manhattan distance): is the sum of the absolute values of each component in a vector.

$$\min_{\theta} \left|\left|Y - X\theta\right|\right|_1 \rightarrow \min_{\theta} \sum_{i=1} |Y_i - X_i\theta|$$

We can transform the minimization problem to a linear program by introducing a slack variable s for each component of the original vector. These slack variables will provide bounds to the absolute values.

$$-s_i \leq |Y_i - X_i\theta| \leq s_i \quad \text{where } s_i \geq 0 \quad \text{for } i = 1,2,\dots,n$$

For the expression above we can derive 2 different inequalities:

$$Y_i - X_i\theta \leq s_i \rightarrow Y_i - X_i\theta - s_i \leq 0$$

$$-s_i \leq Y_i - X_i\theta \rightarrow X_i\theta - Y_i - s_i \leq 0$$

By minimizing over the slack variable s, simultaneously we will minimize the objective function as well. Now, we can rewrite the minimization problem in LP form as follows:

$$\min_{\theta,s} \sum_{i=1} s_i$$

Subject to: $Y_i - X_i\theta - s_i \leq 0$

$$X_i\theta - Y_i - s_i \leq 0$$

To express l_1 norm as a linear program, we had to introduce N slack variables and 2n constraints.

L_∞ norm takes only the largest absolute value of all the components of the vector. We will need to minimize only the value of the largest component. Thus, only one slack variable will be enough to provide a bound on this value.

$$\min_{\theta} \left|\left|Y - X\theta\right|\right|_\infty \rightarrow \min_{\theta} \sum_{i=1} |Y_i - X_i\theta|$$

To transform this to an LP problem let us introduce a slack variable t, which bounds the absolute value of the largest component of the vector (by doing this we bound all other components as well).

$$-t \leq |Y_i - X_i\theta| \leq t \quad \text{where } t \geq 0 \quad \text{for } i = 1,2,\dots,n$$

Again, from the expression above we can derive 2 different inequalities:

$$Y_i - X_i\theta \leq t \rightarrow Y_i - X_i\theta - t \leq 0$$

$$t \leq Y_i - X_i\theta \rightarrow X_i\theta - Y_i - t \leq 0$$

By minimizing over the slack variable t, simultaneously we will minimize the objective function as well. Now, we can rewrite the minimization problem in LP form as follows:

$$\min_{\theta,t} t$$

$$\text{Subject to: } Y_i - X_i\theta - t \leq 0$$

$$X_i\theta - Y_i - t \leq 0$$

To express l_∞ norm as a LP, we had to introduce only one slack variables and 2n constraints.

Part 3: Solving l_1 and l_∞ regression problems using CVXPY

This code generates a synthetic dataset for testing regression algorithms. Compute the parameters that define the line of best fit for l_1 and l_∞. Plot both lines in different colors over the scatter plot.

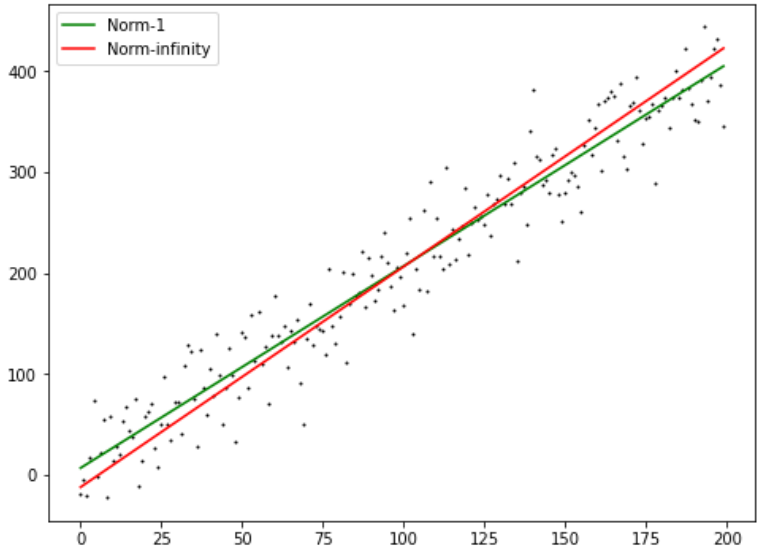


Figure 1. Line of best fit for l_1 and l_∞ regression

In this part, I used CVXPY to find the parameters that define the line of best fit for both l_1 and l_∞ regression (See Appendix for more information).

Table 1. Parameters of l_1 and l_∞ regression

Regression Type	l_1 regression	l_∞ regression
Slope	2.004	2.189
Intercept	6.346	-12.665

Appendix

November 22, 2020

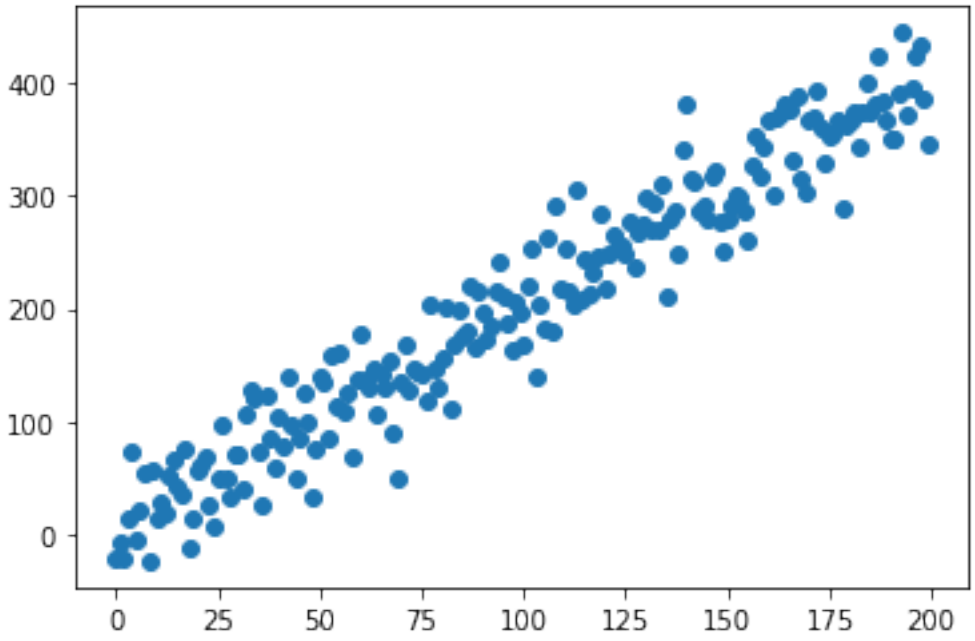
Link to .ipynb version of notebook: https://gist.github.com/AlbionKransiqi/484204ff4ef859241fa1a0e795963d95

1 l_1 and l_{inf} Regression

Some code is given below to generate a synthetic dataset. Using CVX, solve two linear programs for computing the regression line for l_1 and l_inf regression. Plot the lines over the data to evaluate the fit.

First cell is directly taken from assignment instructions

[2]:
```
## l_1 and l_infinity regression using cvxpy
import numpy as np
import cvxpy as cvx
import matplotlib.pyplot as plt

## generate a synthetic dataset

## actual parameter values
theta1_act = 2
theta2_act = 5

## Number of points in dataset
N = 200

## Noise magnitude
mag = 30

## datapoints
x = np.arange(0,N)
y = theta1_act * x + theta2_act *np.ones([1,N]) + np.random.normal(0,mag,N)

plt.figure()
## Scatter plot of data
plt.scatter(x,y)
plt.show()
```



1.1 l_1 Regression

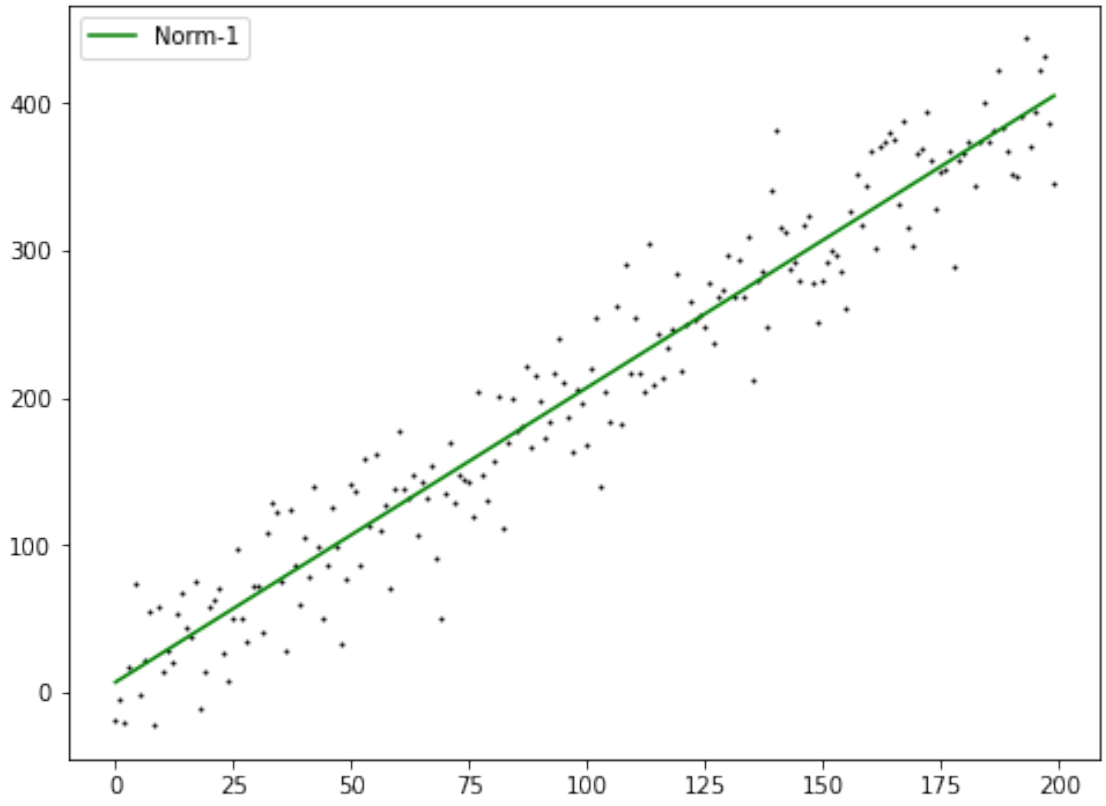
[3]:
```
## reshaping the data according to the minimzation problem
X = np.vstack((x, np.ones(N))).T
Y = y.flatten()

## defining the variables
theta_1 = cvx.Variable(2)
## slack variables
s = cvx.Variable(N)


## the objective function
obj_1 = cvx.Minimize(np.ones(N).T@s)

## constrains
constraints = [Y-X@theta_1<=s,Y-X@theta_1>=-s]

## solving the optimization problem
prob_1 = cvx.Problem(obj_1,constraints)
prob_1.solve();
```

[4]:
```
## printing some information about the L_1 Regression
print("L_1 Regression")
```

```
print("\nstatus:", prob_1.status)
print("optimal value:", prob_1.value)
print(f"Slope θ: {theta_1.value[0]:.3f} and Intercept θ: {theta_1.value[1]:.
↪3f}")
```

```
L_1 Regression

status: optimal
optimal value: 4855.782825761589
Slope θ: 2.004 and Intercept θ: 6.346
```

[5]:
```
## regression equation of l_1 norm
L_1 = x*theta_1.value[0] + theta_1.value[1]

## plotting the l_1 regression
plt.figure(figsize = (8,6))
plt.scatter(x, y, color="black", s=1)
plt.plot(x, L_1, color="g",label="Norm-1")
plt.legend()
plt.show()
```



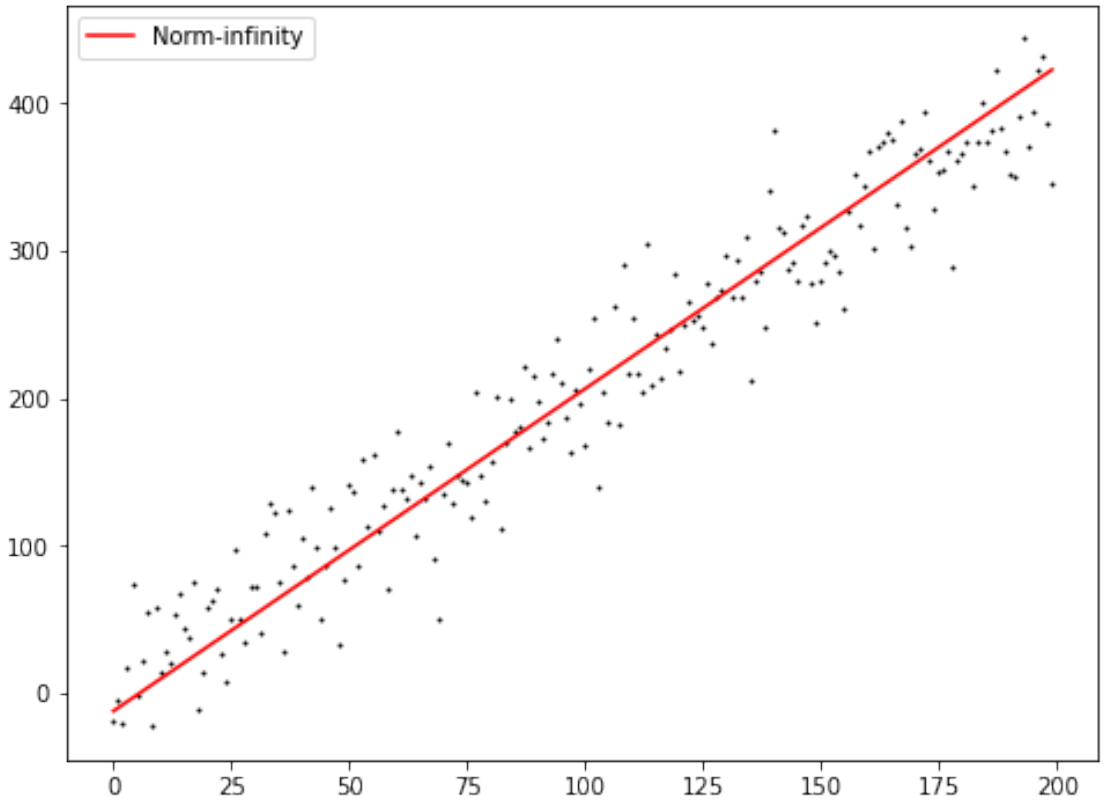
1.2 l_{inf} Regression

[6]:
```
## defining the variables
theta_inf = cvx.Variable(2)
## slack variable
t = cvx.Variable()

## the objective function
obj_inf = cvx.Minimize(t)
## constrains
constraints_inf = [Y-X@theta_inf<=t,Y-X@theta_inf>=-t]

## solving the optimization problem
prob_infinity = cvx.Problem(obj_inf,constraints_inf)
prob_infinity.solve();
```

[7]:
```
## printing some information about the L_infinity regression
print("L_infinity Regression")

print("\nstatus:", prob_infinity.status)
print("optimal value:", prob_infinity.value)
print(f"Slope θ: {theta_inf.value[0]:.3f} and Intercept θ: {theta_inf.value[1]:.
 ↪3f}")
```

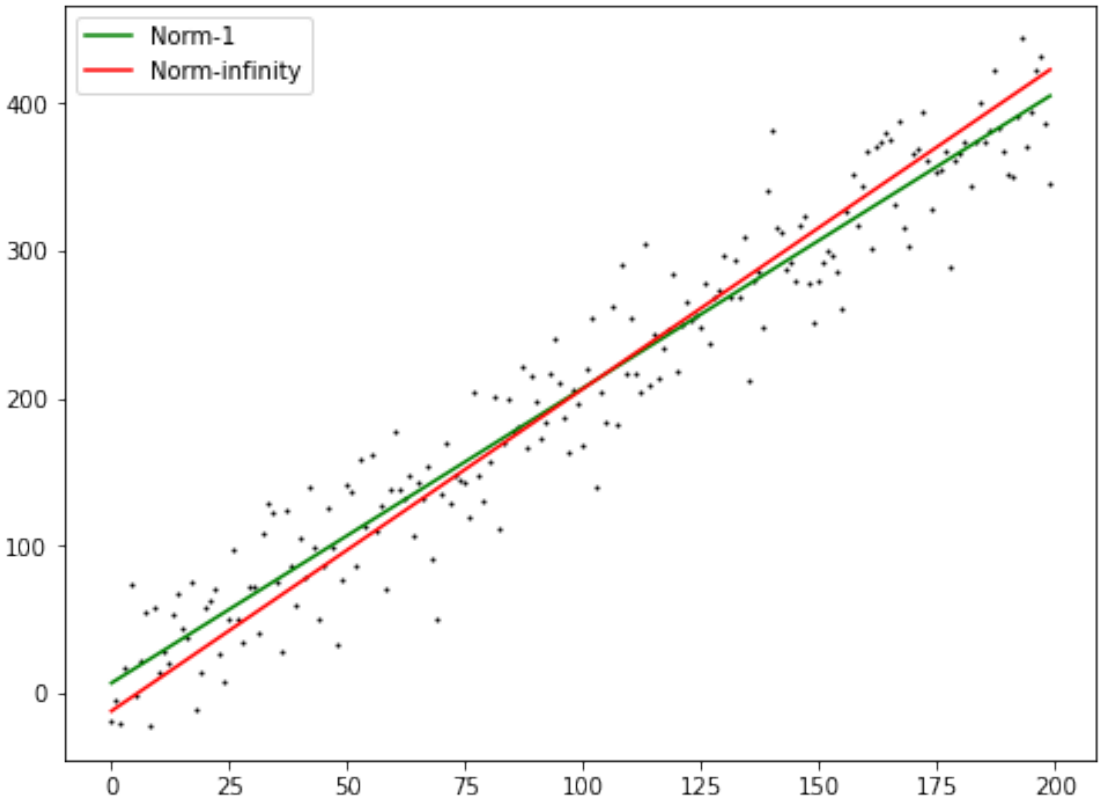
```
L_infinity Regression

status: optimal
optimal value: 87.92144033457558
Slope θ: 2.189 and Intercept θ: -12.665
```

[8]:
```
## regression equation of l_inf norm
L_inf = x*theta_inf.value[0] + theta_inf.value[1]

## plotting the l_inf regression
plt.figure(figsize = (8,6))
plt.scatter(x, y, color="black", s=1)
plt.plot(x, L_inf, color="r",label="Norm-infinity")
plt.legend()
plt.show()
```



1.3 Plotting both lines together

[9]:
```
## combining both plots: l_1 and l_inf regression
plt.figure(figsize = (8,6))
plt.scatter(x, y, color="black", s=1)
plt.plot(x, L_1, color="g",label="Norm-1")
plt.plot(x, L_inf, color="r",label="Norm-infinity")
plt.legend()
plt.show()
```


Norm-1
Norm-infinity
400
300
200
100
0
0
25
50
75
100
125
150
175
200