Discussion 2

ECE 236A, Fall 22

Course Project

TA: Xinlin Li & Merve Karakas

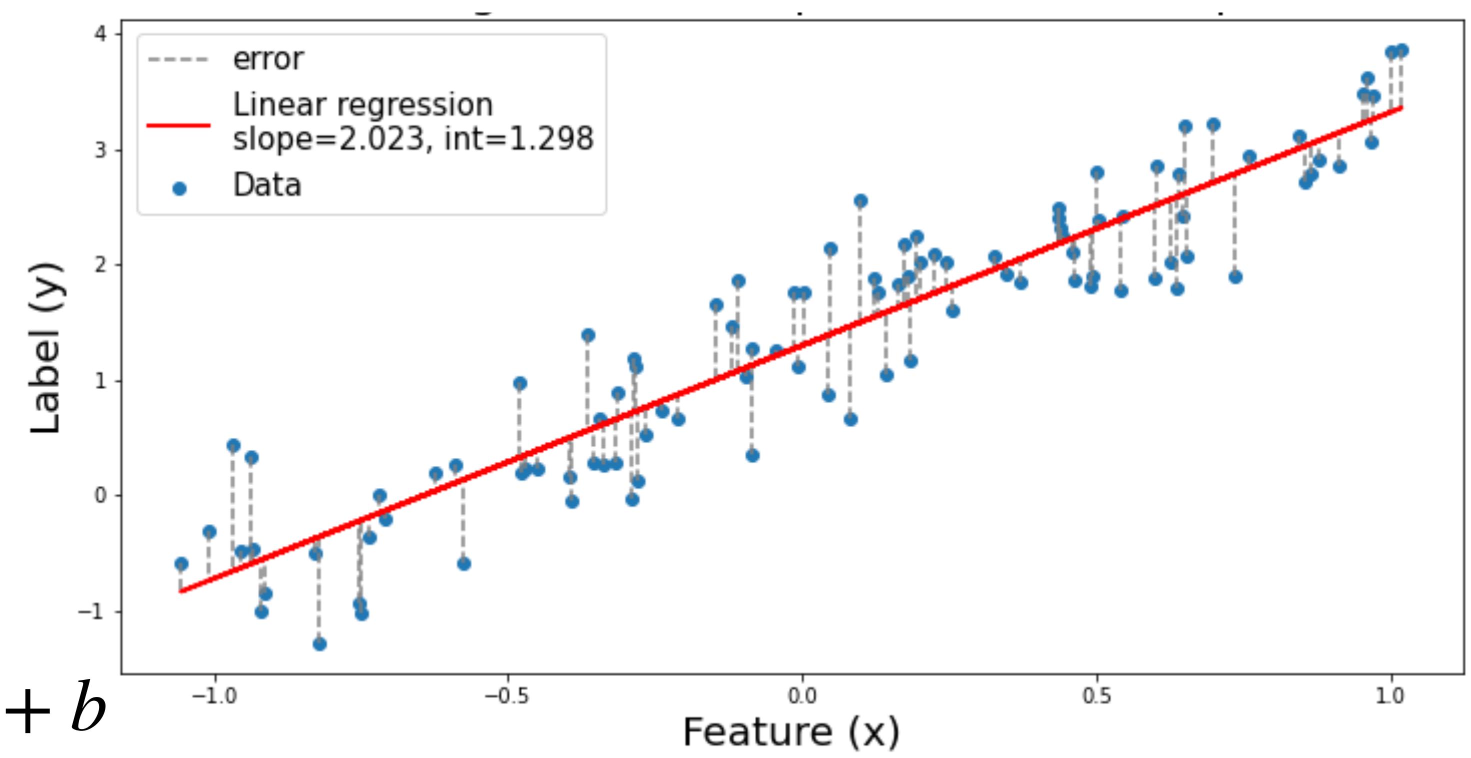
Background

Sparse Designs for Linear Regression

- Linear Regression
- Distributed Settings
- Sparsity
- Online vs Offline Training

Linear Regression

- **Data**: $(X_{train}, Y_{train}) = (x^i, y^i)_{i=1}^{N_{train}}$
  - features: $x^i \in \mathbb{R}^M$
  - labels: $y^i \in \mathbb{R}$
- **Task**: predict $\hat{y}^i = g(x^i)$
  - linear regressor: $g(x^i) = \theta^T x^i + b$
  - parameters: weight θ & bias b



- Evaluation: **mean absolute error (MAE)**

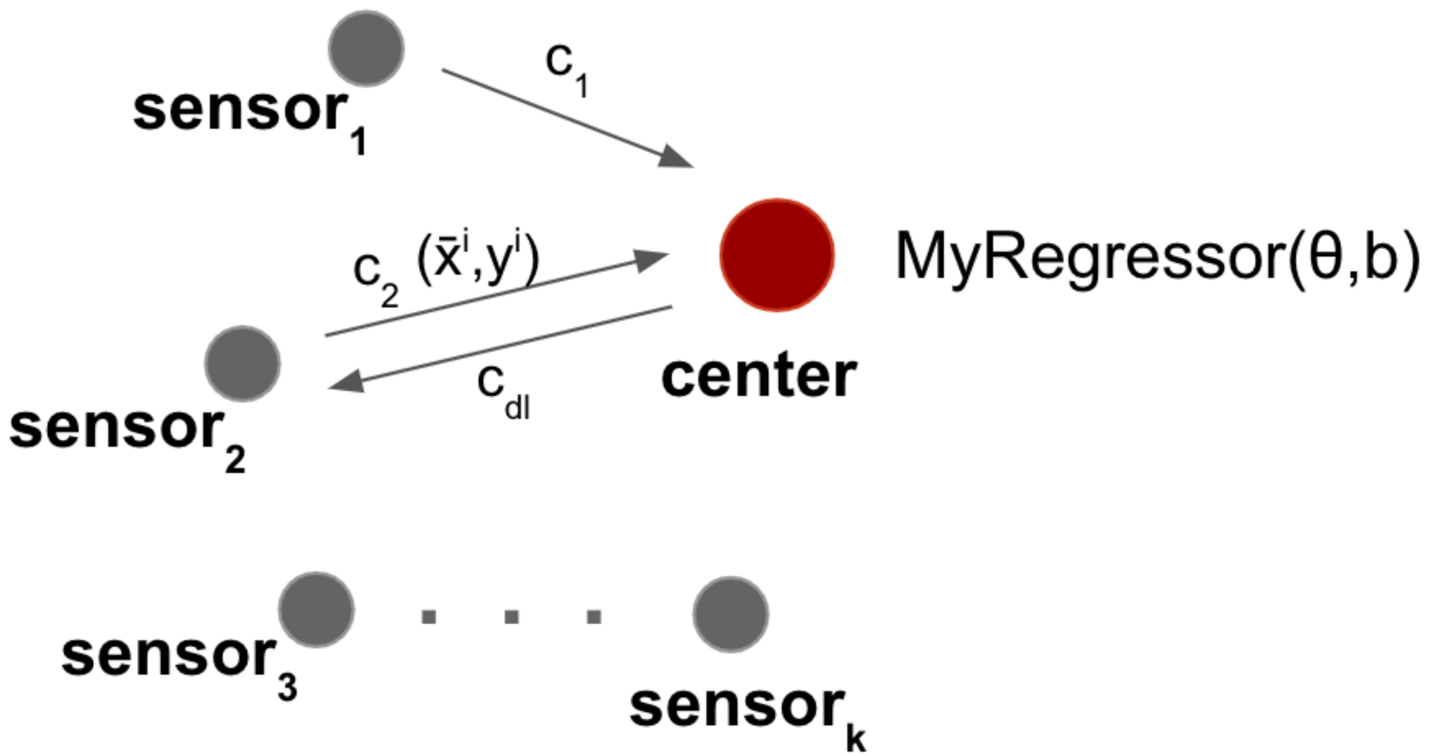
$$error = \frac{1}{N} ||Y - (X\theta + b1)||_1$$

- **Train a regression:**

$$\min_{\theta,b} \; loss = training\ error$$

Distributed Setting

- **Sensors**: observe (x^i, y^i) and send $(\bar{x}^i = f(x^i), y^i)$ to the center
- **Center**: receive data from sensors and train a linear regressor



- **Communication (sensor to center) is constrained**

$$Communication\ Cost = N_{train} * M$$

Sparsity

- **Observations**:
  - Large number of features —> a subset might be enough
  - Small number of features —> less training samples is needed

- **Sparse Regressor:**
  - weight vector θ contains a **small** number of **nonzero** entries

- Enforce sparsity during training:
  - Add $l_1 - norm$ **penalty** to weight vector θ

$$\min_{\theta,b} \; loss_{l1} = \frac{1}{N} || Y_{train} - (X_{train}\theta + b1) ||_1 + \alpha || \theta ||_1$$

Offline Training vs Online Training

- All training **data** is available
- Train the regressor **once**
- **Centralized**

- **Data samples** are received one by one
- **Continuously** update the regressor
- **Distributed**
  - communication constraints

Datasets

1. Synthetic Dataset

- $M = 500$ numerical features
- $N_{train} = 600$ training samples

2. Online News Popularity

- $M = 58$ numerical features
- $N_{train} = 3304$ training samples

- Training data (X_{train}, Y_{train}): used to select data and train the regressor
- Test data (X_{test}, Y_{test}): used to calculate test error ONLY

Project Goals

train a good linear regressor

$$test\ error = \frac{1}{N} || Y_{test} - (X_{test}\theta + b1) ||_1$$

using small amount of training data

$$Communication\ Cost = N_{train} * M$$

Project Tasks

Compression for Sparse Linear Regression

- **Task 1** (7 points) - **Offline** Solution
  - 1-1 (1 point): Reformulate as LP
  - 1-2 (2 points): Train a Regressor
  - 1-3 (1 point): Feature Selection
  - 1-4 (1 point): Samples Selection
  - 1-5 (2 points): Data Selection
- **Task 2** (3 points) - **Online** Solution

Group of 2: **TWO** different algorithms for Task 1-3, 1-4, 1-5 & Task 2

Task 1

Offline Solution

- Assumptions:
  - **centralized**
  - has access to **all** training data

- Sparse Regressor can be obtained by solving

$$\min_{\theta,b} \; loss_{l1} = \frac{1}{N} || Y_{train} - (X_{train}\theta + b1) ||_1 + \alpha || \theta ||_1$$

Task 1-1

Reformulation

- Rewrite the loss function as a **Linear Program**:

$$\min_{\theta,b} \; loss_{l1} = \frac{1}{N} || Y_{train} - (X_{train}\theta + b1) ||_1 + \alpha || \theta ||_1$$

Task 1-2

Regressor Training

- **Implement** an algorithm to train a regressor
  - explore the effect of $l_1 - norm$ penalty
- **Plot** penalty strength (α) vs training error and test error
  - At least 5 values in a reasonably large range

- Feel free to use libraries:
  - e.g. sklearn, cvxpy and any others

Task 1-3

Feature Selection

- Reduce the number of **features** M used for training

- **Plot** number of features vs training error and test error
  - Cover [1%, 10%, 30%, 50%, 100%]

- $l_1 - norm$ penalty is not required
- Does the solution of Task 1-2 help?

Task 1-4

Sample Selection

- Reduce the number of data **samples** N_{train} used for training
- **Plot** number of samples vs training error and test error
  - Cover [1%, 10%, 30%, 50%, 100%]

- $l_1 - norm$ penalty is not required
- Form LP problems
- Heuristic solutions is also acceptable:
  - Look into raw data
  - geometrical properties

Task 1-5

Data Selection

- Reduce **communication cost** $N_{train} * M$ during training
- **Plot** communication cost vs training error and test error
  - Cover [1%, 10%, 30%, 50%, 100%]

- $l_1 - norm$ penalty is not required
- Combine Sample & Feature selection (Task 1-3 & 1-4)
- Is there a way to jointly optimize it?

Task 2

Online Solution

- **Distributed** Setting:
  - center: regressor training   sensor: data selection
- **Assumption**:
  - Consider **ONE** sensor node
  - The sensor keeps a **memory** of all past data
  - **Unlimited** communication from the central node to sensor
    - e.g. model parameters

- $l_1 - norm$ penalty is not required
- Is it possible to update the regressor using one/small sets of data?

Task 2

Online Solution

- **Plot** communication cost vs training error and test error
    - Cover [1%, 10%, 30%, 50%, 100%]

- Feel free to add other plots to justify your algorithms

Given: Skeleton

MyRegressor.py

```python
class MyRegressor:
    def __init__(self, alpha):
        self.weight = None
        self.bias = None
        self.training_cost = 0
        self.alpha = alpha

    def select_features(self):
        ''' Task 1-3
            Todo: '''

        return selected_feat

    def select_sample(self, trainX, trainY):
        ''' Task 1-4
            Todo: '''


        return selected_trainX, selected_trainY

    def select_data(self, trainX, trainY):
        ''' Task 1-5
            Todo: '''

        return selected_trainX, selected_trainY


    def train(self, trainX, trainY):
        ''' Task 1-2
            Todo: '''

```

ToDo:

- Complete each Task **in given method**
  - **Update** class attributes
  - **Output** required results
  - Can add methods input
  - Can call other methods
  - Can define other auxiliary methods

Don’t:

- Change given code & structure
- Change method output

Given: Utilities

utils.py

- Data Preparation
  - No need to do data preprocessing
- Results Visualization
  - No need to format plots

```
import csv
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing, model_selectio

def prepare_data_gaussian(): ...

def prepare_data_news(): ...

def plot_result(result): ...
```

- Do **NOT** change any given code
- **More** utility functions are **allowed** (submit this file if you added any)

Report

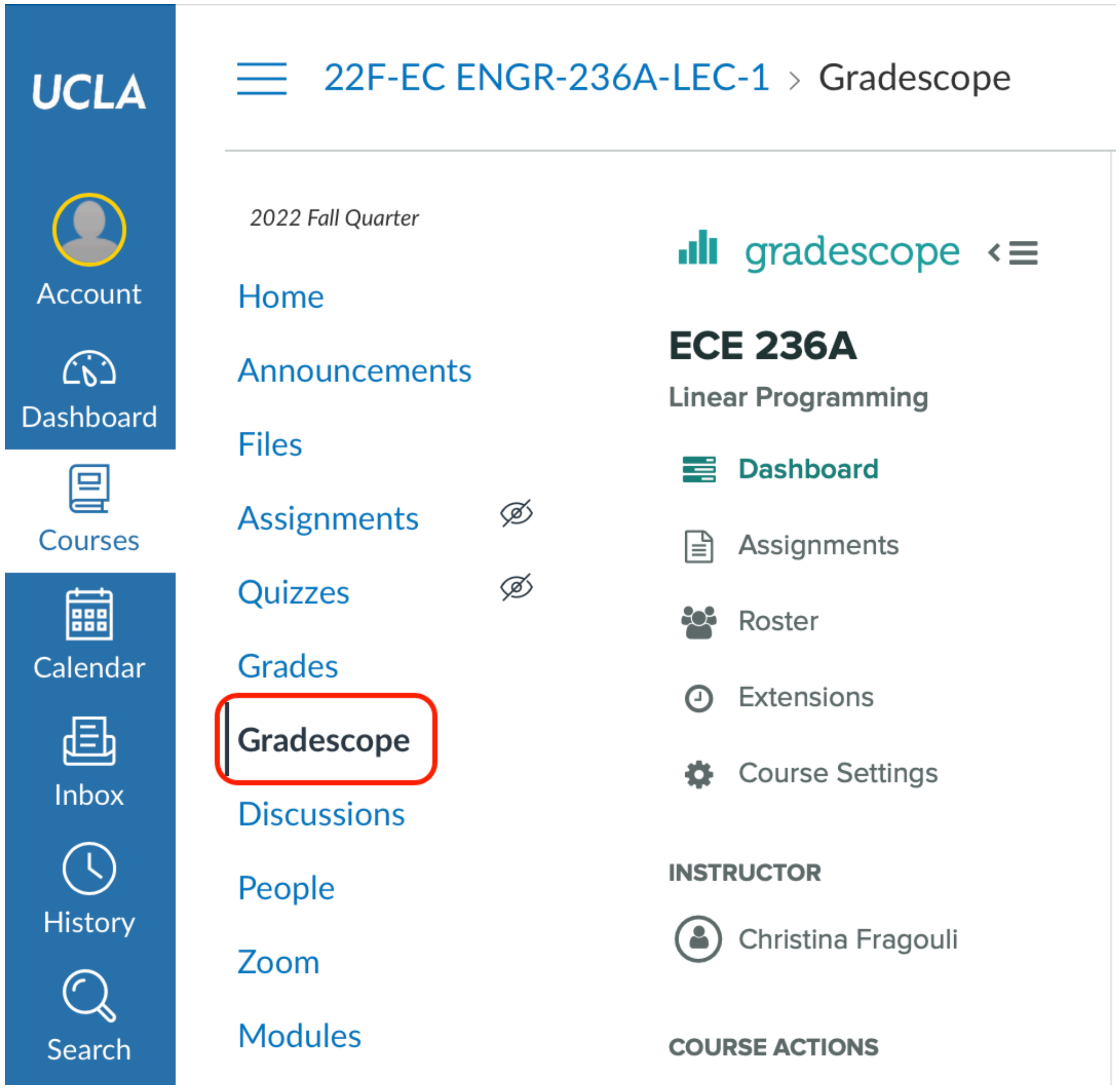
- Up to 2 pages (exc. appendix)
  - group of two: 3 pages
- Formulated LP for Task 1-1
- Intuitions & Detailed description of your algorithms
- Observations & Discussions
- 10 / 18 Plots of your experiment results (in appendix)

Submission

- One **Zip** File
  - 2/3 pages report
  - MyRegressor.py
  - utils.py (if added functions)
  - Optional: experiment code

- **Deadline:** Nov. 3rd

- Submit to **Gradescope**



Grading

- Base (10 points)
  - **Correctness & Completeness**

- Bonus (5 points)
  - **Creativity & Accuracy**
  - 5 points for top 10 groups
  - Presentation on Nov. 17th