2024_final_project

May 7, 2024

1 2024 Spring ORF307 Final Project

Instructor: B. Stellato

AIs: I. Wang, V.Ranjan, J. Smith, P. Beneventano

1.1 Instructions

Exam files: the exam files are available to download at THIS LINK.

If you cannot click on the link, then try copy + pasting the following:

https://www.dropbox.com/scl/fo/b6an98pypucpvdhb81osz/AHSPIHMuLOtnttr-CDG6ZZY?rlkey=6wph0csyr9gldp2lo7akkx4ln&st=ckxp1xua&dl=0

Date and time: From May 8, 2024 at 00:01 am to May 10, 2024 at 11:59pm.

- Total time after download: 24 hours. No late submissions allowed. Note that the deadline is either 24 hours after download, or May 10, 2024 at 11:59pm, **whichever comes first**.

1.2 Exam Rules

- You are allowed to use all course materials on the midterm (lecture notes, books, precept materials, code, and homeworks). But you cannot use internet to search for answers.
- You have to justify all your answers. If you use code from the course materials, you have to explain what each step means.
- You cannot communicate with anyone during the exam.
- No late submissions allowed. Make sure your submission goes through on time. You can resubmit as many times as you like until your time expires.
- The exam is to be submitted electronically on Gradescope before 11:59pm on the final day.

2 Final Exam

2.1 House Price Modeling

You are given data about houses and are tasked with building a model to predict the sell price of these houses. For the data, you have 545 total samples with 11 features:

Feature	Description
area	the area in 1000s of square feet
bedrooms	the number of bedrooms
bathrooms	the number of bathrooms
stories	the number of stores
mainroad	1 if the house is connected to the main road and 0 otherwise
guestroom	the number of guestrooms
basement	1 if the house has a basement and 0 otherwise
airconditioning	1 if the house has centralized AC and 0 otherwise
parking	the number of extra parking spaces
prefarea	1 if the house is located in a 'preferred area' and 0 otherwise
furnished	1 if the house comes prefurnished and 0 otherwise

The data also contains a price column, which is the sell price of the house in millions of USD. For modeling purposes, we split the data into a training and testing set with an 80/20 split.

To model prices, you decide to build a linear model. That is, your goal is to determine a vector of weights θ to approximately fit: $y \approx X\theta$, where X is the matrix of features and y is the vector of prices. However, your cousin is a realtor and told you that all these features contribute positively to the price of the house. You trust your cousin and you are including this information by enforcing that **all weights in the vector θ are nonnegative**.

For this project, we will fit the linear model and handle the nonnegativity in a few different ways.

Run the following cells to import the data. Note that after importing the data and separating out X and y, we prepend a column of ones to the feature matrices in order to learn an offset.

[1]:
```
import numpy as np
np.set_printoptions(precision=4)  # Print few decimal places
np.set_printoptions(suppress=True)  # Suppress scientific notation
import cvxpy as cp
import pandas as pd
from numpy.linalg import cholesky as llt
import matplotlib.pyplot as plt
```

[2]:
```
dftrain = pd.read_csv('train.csv')
dftest = pd.read_csv('test.csv')
dftrain  # to look at the training dataset
```

[2]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	\
0	1.750	3.62	2.0	1.0	1.0	1.0	0.0	
1	2.695	4.00	2.0	1.0	1.0	1.0	0.0	
2	2.870	3.04	2.0	1.0	1.0	0.0	0.0	

```
3    2.590   3.60       2.0        1.0      1.0       1.0        0.0
4    4.515   9.86       3.0        1.0      1.0       1.0        0.0
..     …     …          …          …        …         …          …
431  6.790   4.00       3.0        2.0      2.0       1.0        0.0
432  4.305  10.36       2.0        1.0      1.0       1.0        0.0
433  9.800   5.75       3.0        2.0      4.0       1.0        1.0
434  3.710   3.60       3.0        1.0      1.0       1.0        0.0
435  5.040   6.60       3.0        1.0      1.0       1.0        1.0

     basement  airconditioning  parking  prefarea  furnished
0         0.0              0.0      0.0       0.0        0.0
1         0.0              0.0      0.0       0.0        0.0
2         0.0              0.0      0.0       0.0        0.0
3         0.0              0.0      0.0       0.0        0.0
4         0.0              0.0      0.0       0.0        0.0
..        …                …        …         …          …
431       1.0              1.0      0.0       1.0        0.0
432       0.0              0.0      1.0       1.0        0.0
433       0.0              1.0      1.0       1.0        0.0
434       0.0              0.0      1.0       0.0        0.0
435       1.0              0.0      0.0       1.0        1.0

[436 rows x 12 columns]
```

[3]:
```
Xy_train = dftrain.to_numpy()
Xy_test = dftest.to_numpy()

# prepending a column of ones to both train and test data
X_train = np.column_stack([np.ones(Xy_train.shape[0]), Xy_train[:, 1:]])
y_train = Xy_train[:, 0]
X_test = np.column_stack([np.ones(Xy_test.shape[0]), Xy_test[:, 1:]])
y_test = Xy_test[:, 0]
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

[3]:
```
((436, 12), (436,), (109, 12), (109,))
```

For this project, whenever you fit a model in any of the parts, make sure that you train the model using the training data (X_train and y_train). However, when *evaluating* a model, use the normalized train and test errors, which are computed by the following functions:

[4]:
```
def normalized_test_error(theta):
  # Test error calculation
    return np.linalg.norm(X_test @ theta - y_test) / np.linalg.norm(y_test)

def normalized_train_error(theta):
    # Training error calculation
    return np.linalg.norm(X_train @ theta - y_train) / np.linalg.norm(y_train)
```

2.2 Part A: Least Squares (30 Points)

For this part, we will use least squares. After solving any least squares problem, we will post-process the output by clipping the negative values to zero. This can be done with `theta_clip = np.maximum(theta, 0)`.

1. (10 points) First, we will use least squares to solve the problem. Let the *training residual* be $r = X\theta - y$. That is, we aim to solve the following problem:

$$\text{minimize} \quad \|X\theta - y\|_2^2.$$

Write the equation that you need to solve to compute the optimal solution $\theta^\star$. Solve the least squares problem over training data `X_train` and `y_train`. Then, clip the negative values of $\theta^\star$ and report the normalized train and test errors.

2. (10 points) Now, to help with the generalization to the test set, add a secondary objective, with positive weight λ, that penalizes the squared 2-norm of θ, i.e., $\lambda\|\theta\|_2^2$. Write the equation that you need to solve to compute the optimal $\theta^\star$.

3. (10 points) Form the optimization problem from part A.2 with `X_train` and `y_train`. For each $\lambda \in \{0.1, 1, 10, 100, 1000\}$, solve the problem and clip the negative values from $\theta^\star$. Report the normalized train and test error for each value of λ. What do you notice?

2.3 Part B: Linear Programming (35 Points)

Your other cousin, is a consultant and took ORF307 a few years ago. He told you that solving a linear program (LP) works much better for this problem.

1. (5 points) Replace the least squares objective with an ℓ_1-norm objective on the training residual $r = X\theta - y$, subject to nonnegativity constraints on θ, and add an ℓ_1-norm penalty on θ scaled by λ. Write down the problem you obtain. Next, formulate and solve the problem in CVXPY over training data `X_train` and `y_train`, for $\lambda = 2$, and report the objective and θ values. Also report the normalized train and test errors.

2. (5 points) Write the above problem as an LP.

3. (10 points) Take the dual of the LP.

4. (5 points) For $\lambda = 2$, solve the primal and dual LPs with cvxpy, and verify the zero duality gap (up to numerical precision). Verify, for the primal, that you obtain the same solution and normalized train and test errors as in B.1. Compare the train and test errors to the ones obtained with least squares in part A.

5. (6 points) Let's look at the sensitivity of the solution with respect to λ. Let the ℓ_1 regularizer be scaled instead by $\lambda + \epsilon$, where $\lambda = 2$. Give an expression for the primal optimal solution $p^\star(\epsilon)$ around $\epsilon = 0$, in terms of the variables in B.2 and/or B.3. In terms of global sensitivity, is this expression an upper or lower bound of the true primal solution after perturbing λ? Explain why. Verify this by plotting the true objective values as well as the estimated objective values, for ϵ taking 100 values in the range $[-2, 40]$ (you can use `np.linspace(-2,40,100)`).

6. (4 points) For the set-up in problem B.5, plot the normalized train and test errors for the true optimal solutions, as a function of ϵ in the range $[-2, 40]$. In a separate graph, plot also

the 1-norm of the θ values obtained. Comment on both trends. Do they corroborate your findings from part A?

2.4 Part C: Integer Programming (35 Points)

You are probably smarter than both your cousins and you think that only a few of these features actually matter. For this part, you will impose the constraint that at most k of the values in θ are strictly positive.

1. (5 points) Formulate the new constraint in terms of the *cardinality* of θ. We define cardinality of a vector θ the number of non-zero components of the vector θ. We denote it as $\mathbf{card}(\theta)$. Write the new optimization problem where we minimize the ℓ_1-norm of the training residual $r = X\theta - y$, subject to θ being nonnegative and cardinality of θ being at most k.
2. (5 points) Formulate the new problem with the cardinality constraints as a mixed-integer optimization problem (MIP) using the big-M method. You do not need to compute M explicitly; you may just assume that it is a sufficiently large constant.
3. (10 points) Solve the MIP, over training data `X_train` and `y_train`, for all possible $k \in \{1, 2, \ldots, 11\}$ using the SCIPY (HIGHS) solver (by calling `problem.solve(solver=cp.SCIPY)`) and plot the normalized test and train errors vs k. Use the same solver and the value $M = 100$ to solve all the remaining questions in this part.
4. (10 points) Consider now the ℓ_1-regression model with a penalty on the number of components, i.e.

$$\begin{array}{ll} \text{minimize} & \|X\theta - y\|_1 + \lambda \cdot \mathbf{card}(\theta) \\ \text{subject to} & \theta \geq 0. \end{array}$$

Formulate the problem with the big-M formulation to represent the cardinality. Find the solution of this problem with CVXPY for 32 evenly spaced $\lambda \in [0, 8]$ (you can define the λ values with `np.linspace(0, 8, 32)`). Then plot the normalized train and test errors vs λ and $\mathbf{card}(\theta)$ of the solution found vs λ. This should take at most a few minutes to run.

5. (5 points) Answer the following questions and explain why.

- What is the trend of the errors, what does it imply?
- There is a certain value of k after which the test error decreases only by decrements smaller than 0.01. Which one? What does this mean? Which are the nonzeros of θ in that case?
- How does the solution at point C.4 relate to the one at point C.3?