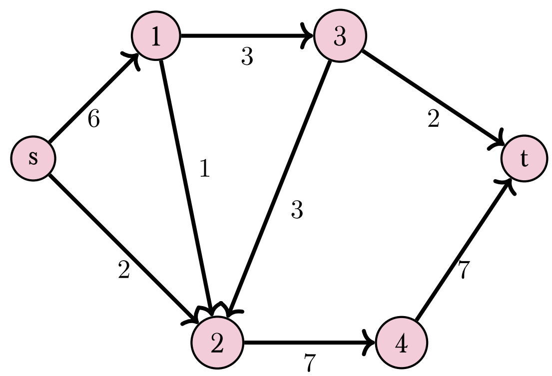
ISyE524: Introduction to Optimization
Problem Set #3
Due: March 17 at 11:59PM

Notes and Deliverables:

- Submit solutions to all problems in the form of a single PDF file of the IJulia notebook to the course dropbox. (If you do not submit as a single PDF you will lose points.)
- Be sure that your answers (including any pictures you include in the notebook) are in the correct order. (If you do not submit the problems in order, you will lose points.)

1 MaxFlow

Consider the network below, in which the label on each arc represents the capacity of the arc.



1-1 Problem Write a linear program formulation to maximize the flow from source s to sink t. Please use a "circulation arc" from t to s in your formulation.

1-2 Problem Write the dual of the linear programming formulation from Problem 1-1.

1-3 Problem Implement your model from Problem 1-1 in Julia and JuMP to find the maximum flow from s to t in the network.

1-4 Problem Use the complementary slackness conditions to find an optimal *dual* solution to your formulation in Problem 1-2. You should note how the optimal dual solution identifies a ***minimum cut*** in the network.

1-5 Problem Find a cut in the network whose capacity equals the value of the maximum flow. Recall a cut can be specified as a set of nodes S that contains the source node. The capacity of the cut is the sum of the capacity of edges that start in S and end outside of S.

2 Lasso

In this problem, we will investigate different approaches for performing polynomial regression on the data in the file `lasso_data.csv`.

You can read and plot the data with the following

Julia Code

```
using PyPlot, CSV, DataFrames

data = CSV.read("lasso-data.csv", DataFrame)
x = data[:,1]
y = data[:,2]

cla()
figure(figsize=(8,4))
plot(x,y,"r.", markersize=10)
grid("True")
# Only need this in vscode?
display(gcf())
```

2-1 Problem Create a Julia/JuMP model to solve a convex optimization problem that finds the best polynomial fit for a polynomial of degree 6. Print the value of the error (total squared residuals) as well as the values of the coefficients in the polynomial.

2-2 Problem Create a Julia/JuMP model to solve a convex optimization problem that finds the best polynomial fit for a polynomial of degree 18. Print the value of the error (total squared residuals) as well as the values of the coefficients in the polynomial.

2-3 Problem Make a plot that shows the data as well as the best fit to the data for polynomials of degree $d = 6$ and $d = 18$ that you created in Problems 2-1 and 2-2

2-4 Problem Our model is too complicated because it has too many parameters. One way to simplify our model is to look for a sparse model (where many of the parameters are zero). Solve the $d = 18$ problem once more, but this time use the Lasso (L_1 regularization). Start with a small λ and progressively make λ larger until you obtain a model with at most $K = 6$ coefficients positive. Print the resulting (squared) error. And plot the resulting fit.

Note: due to numerical inaccuracy in the solver, you may need to round very small coefficients (say less than 10^{-5}) down to zero.

3 Beam Me Up

In treating a cancerous tumor with radiotherapy, physicians want to bombard the tumor with radiation while avoiding the surrounding normal tissue as much as possible. They must therefore select which of a number of possible radiation beams to use, and figure the "weight" (actually the exposure time) to allot for each beam.

Abstractly, in a mathematical model of this decision problem, we are given a set of "beams" B, and a set of regions R, where $R = N \cup T$, where N is the set of normal regions, and T is the set of cancerous (tumor) regions.

If the weight/exposure of beam $b \in B$ is x_b (a continuous decision variable), then beam $b \in B$ delivers an amount of radiation of $a_{br}x_b$ to region $r \in R$. The total amount of radiation delivered to a particular region $r \in R$ is obtained by adding the contributions of each beam:

$$\sum_{b \in B} a_{br}x_b.$$

The maximum weight that can be applied for any beam $b \in B$ is W_b. Ideally, each tumor region $r \in T$ should receive as large a dose as possible, while each non-tumor region $r \in N$ should receive a dose of at most p_r.

The ***tumor score*** of any tumor region is the amount of radiation received:

$$\tau_r := \sum_{b \in B} a_{br}x_b \quad \forall r \in T.$$

The ***tissue damage*** of any normal region is the total amount delivered to that region over the prescribed amount

$$\Delta_r := \max\left(\sum_{b \in B} a_{br}x_b - p_r, 0\right) \quad \forall r \in N$$

Doctors would like to simultaneously maximize $\sum_{r \in T} \tau_r$ while minimizing $\sum_{r \in N} \Delta_r$.

3-1 Problem Write a weighted multi-objective optimization model (a linear program) that shows the tradeoff between the amount of good-dose ($\sum_{r \in T} \tau_r$) delivered and the amount of damaging dose delivered ($\sum_{r \in N} \Delta_r$)

Hint: You will need to write the problem using linear inequalities. Do not use the $\max$ operator in your constraints. Recall how to model (some) piecewise linear functions using linear inequalities.

3-2 Problem Suppose there are a collection of 6 beams, each with a maximum weight/intensity of $W_b = 3 \; \forall b \in B$. There are also 6 regions, regions 1-3 are normal and regions 4-6 are cancerous. The amount of radiation delivered by beam b to region r is given in the matrix below:

	normal1	normal2	normal3	tumor1	tumor2	tumor3
beam1	15	7	8	12	12	6
beam2	13	4	12	19	15	14
beam3	9	8	13	13	10	17
beam4	4	12	12	6	18	16
beam5	9	4	11	13	6	14
beam6	8	7	7	10	10	10

All normal regions have a desired upper bound of $p_r := 65 \; \forall r \in N$. Implement your model from Problem 3-1 in JuMP and plot the Pareto frontier/tradeoff curve

If you make a vector called `tumor_dose` and a vector called `normal_penalty` that contain the tumor dose and normal tissue penalty amounts for different values of tradeoff parameter, then this code will make a pareto plot for you.

Julia Code

```
using PyPlot

function paretoPlot(x,y)
    figure(figsize=(10,10))
    plot( x, y, "b.-", markersize=4 )
    xlabel("Tumor Dose")
    ylabel("Normal Penalty")
    # Only need this in vscode?
    display(gcf())
end
;

paretoPlot(tumor_dose, normal_penalty)
```