Homework #5 (4 points)

Submission requirements

Upload a **single PDF or HTML file** of your IJulia notebook for this entire assigment. Do not submit an .ipynb file. Clearly denote which question each section of your file corresponds to.

Problem 1 - Nonlinear Manufacturing

A manufacturing company produces n individual products. The production process for each product involves m stages with limited resource availability for each stage. The company wants to minimize the total production cost while satisfying demand and staying within resource restrictions. The cost function can be represented as the product of the production level of each product raised to a given power (a_i for product i), multiplied by cost coefficients (c_i for product i). In other words, the cost can be written as $\Pi_{i=1}^{n}(c_i x_i^{a_i})$.

There is a minimum demand level that must be met for each product i (d_i). Each product i requires r_{ij} of resource j for the jth stage of production, and no more than R_j of resource j can be used in total. (In other words, $\sum_{i=1}^{n} r_{ij}x_i \leq R_j$ for all stages $j = 1, \cdots, m$.

(a) Express this problem as a nonlinear program, and convert it into a convex optimization problem. *Hint*: Use the log function.

(b) Consider a simple instance of this problem where $n = 3$ and $m = 3$. The cost coefficients are $c = [1; 2; 3]$ and the exponents are $a = [0.5; 0.4; 0.3]$. Demand for products 1, 2, and 3 is 100, 150, and 200, resepctively. The resource constraints can be represented as follows:

$$0.1x_1 + 0.2x_2 + 0.3x_3 \leq 500$$
$$0.2x_1 + 0.1x_2 + 0.4x_3 \leq 600$$
$$0.3x_1 + 0.3x_2 + 0.2x_3 \leq 700$$

Solve this problem using JuMP. Use the Ipopt solver (or any other solver that can solve convex optimization problems) and the "@NLobjective(...)" command to let the solver know it needs to deal appropriately with a nonlinear objective function. What are the optimal production levels for each product?

Problem 2 - Minimum Enclosing Circles

Formulate a quadratic program to solve this variation of the minimum enclosing ball (MEB) problem. You need to determine the centers z_j and radii d_j of a set of balls (circles) $j \in J$ so that each of a set of points $i \in I$ located at c_i are enclosed in your generated circles. The catch: each point is assigned a "color." The set of points of each color should be enclosed in a circle of the "same color" (note that it doesn't actually

matter what color your circle is; you will just need a set of constraints for all the points belonging to a given color).

Your task is to find the optimal set of circles such that each circle encloses the appropriate set of points and the radius (defined below) is minimized. All points must belong to at least one circle. It is okay for a point to belong to multiple circles.

Solve the problem twice, defining the minimum radius in each of these two ways:

(a) Define minimum radius as the sum of all the radii of your k circles (or radii squared). Try solving this with at least two values of (k) representing the number of colors (e.g., (k = 2) and (k > 2)) to test your model.

(b) Define minimum radius as the minimum of the largest of the k radii (or radii squared) -- this is a variation on the minimax problem. Try solving this with at least two values of (k) representing the number of colors (e.g., (k = 2) and (k > 2)) to test your model.

(c) What do you observe (visually) about the circles produced by (a) and (b)? Is there a noticable difference in circle size? (Does this match what you expect from our Regularization lectures?)

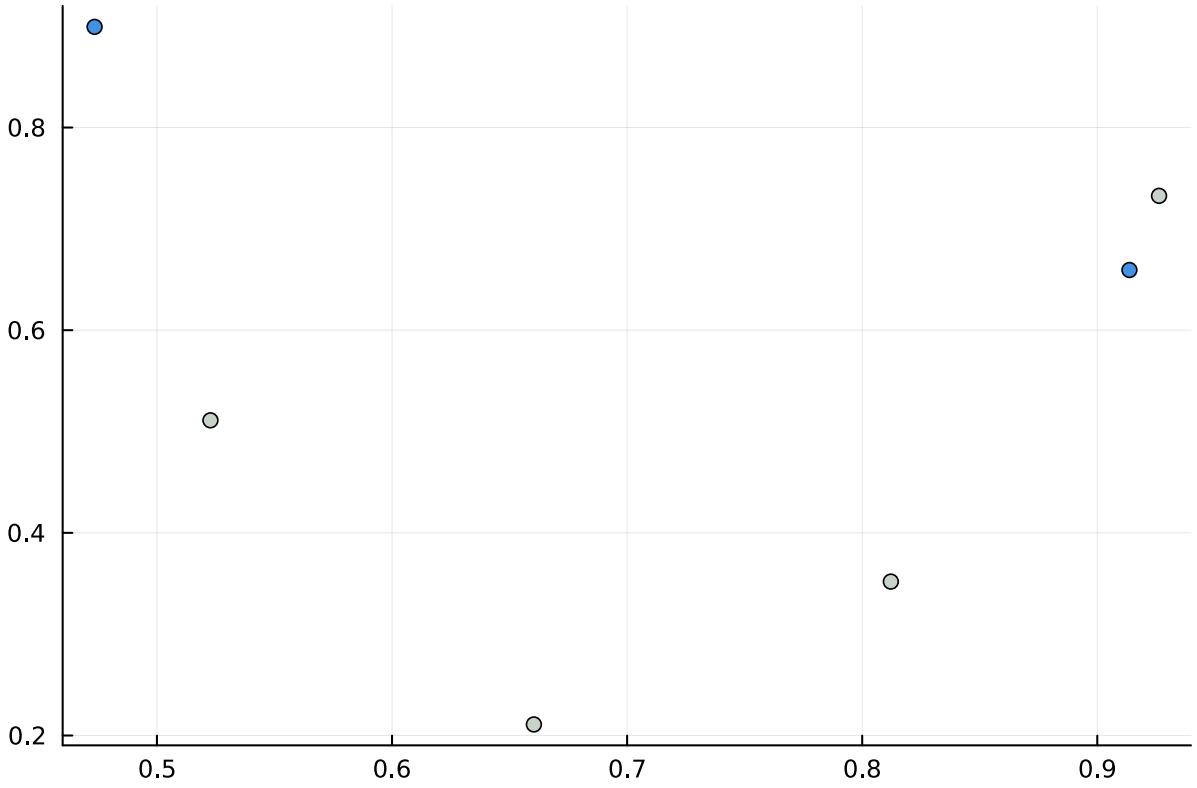
You can use the following code to start building your model. Start with $n = 6$ points to help with debugging, but display the output of your model for both $n = 6$ and a very large value of n, such as $n = 200$.

Hints:

- You can use Gurobi for this problem if you formulate the model with a variable r representing the *radius squared* of each colored circle
- Some additional code is posted below to graph the solution of your problem against the original set of points. It won't run until you build and solve your model, but it might help you get on the right track.
- There's a problem on Week 5 Practice Problems that is very similar. You might want to use that code as a starting point.

In [4]:
```
n = 6 # number of points (small value to start)
k = 2 # number of colors (circles) to use (small value to start)

using Random, Plots

# randomly generate n points with (x,y) coords between 0 and 1
points = [(rand(),rand()) for i in 1:n]

# randomly assigned points to colors
point_color = [[] for i in 1:k]
pc = rand(1:k,n)

for i in 1:n
    for j in 1:k
        if pc[i] == j
            point_color[j] = vcat(point_color[j],points[i])
        end
    end
end

# randomly generate color hex code
color = ["" for i in 1:k]
for i in 1:k
    color[i] = randstring(['0':'9'; 'a':'f'])
end

fig = scatter()
#for each of the n points, plot it on a scatterplot
for j in 1:k
    for i in 1:length(point_color[j])
        colstring = string("#",color[j])
        scatter!([point_color[j][i][1]],[point_color[j][i][2]],
            color=colstring,legend=false)
    end
end

# show the figure
fig
```

Out[4]:



In [ ]:
```
### FOR PLOTTING AFTER SOLVING PROBLEM ###

fig = plot(aspect_ratio=:equal)
# replot n points
# for each of the colors
for j in 1:k
    # for each point belonging to that color
    for i in 1:length(point_color[j])
        # create a string of the color hex code
        colstring = string("#",color[j])
        # plot the ith point belonging to color j (x_coord, y_coord)
        scatter!( [point_color[j][i][1]],[point_color[j][i][2]],
            color=colstring,
            legend=false) # color of point is randomly generated hex code
    end
end

t = range(0,stop=2*pi,length=100) # parameter that traverses the perimeter c

# for each color
for i in 1:k
    # plot circle centers
    plot!([value(x[i,1])],[value(x[i,2])])
    # plot perimeter of circle by tracing with t parameter. radius is square
    plot!([value(x[i,1]) .+ sqrt(max(0,value(r[i])))*cos.(t)],
        [value(x[i,2]) .+ sqrt(max(0,value(r[i])))*sin.(t)],legend=false)
end

# display figure
fig
```

Problem 3 - Puppies!

Let's suppose the CS524 class has decided collectivley to adopt some puppies. There are 6 different breeds of dogs we can adopt. Each breed has a minimum and maximum number of puppies if we adopt any of that breed (we have to adopt entire litters so no puppies are left alone!). These restrictions, along with the expected happiness we get from each breed, are in the following table:

Breed	Min adopted	Max adopted	Happiness
Golden retriever	5	7	8
Shiba Inu	2	10	9
Great Dane	3	5	4
Pomeranian	6	13	5
Water Spaniel (WI state dog!)	3	15	10
Husky	7	10	3

Because of how insane Huskies are, the total number of Husky puppies adopted should be no more than the combined number of Golden Retriver, Water Spaniel, and Shiba Inu puppies. In addition, if any Pomeranians are adopted, we require that at least the minimum number of Great Dane puppies are adopted. We will adopt exactly 35 total dogs and obviously we want to maximize our total happiness. Which breeds of puppies should we adopt, and how many of each? Formulate an integer *linear* program (i.e., all constraints \& objective must be linear functions, even though variables can now be restricted to integer values) that can answer this question, then implement and solve it using Julia/JuMP. (Cbc, HiGHS, and Gurobi can all solve integer linear programming problems.)

In [ ]: