MATH 168 Midterm Exam

Exam Structure

This is a 72-hour take-home midterm exam. It contains 8 problems. Many of these problems can be done independently of each other. **You do not need to do all the problems**. There is a total of 23 available points spread among the problems. Your score on the midterm can be at most 15. So, you can choose-your-own-adventure through the problems in order to aim for a total of 15 points.

There are two main paths that I expect most of you will take. Both of them total 15 points.

- **Path 1: "I like math more than coding."** Problems 1, 2, 3, 4, 5, and 6.
- **Path 2: "I like coding more than math."** Problems 1, 3, 5, 7, and 8.

You're welcome to mix and match between these paths. However, it is likely a better use of your time to focus on doing one path really well than to put partial effort into problems from both paths.

Grading

This exam will be graded in a somewhat more traditional fashion: no re-attempts, but we will grant partial credit. To receive full credit on problems, you should:

- Carefully read all instructions.
- Clearly explain your arguments and ideas, often using more words than mathematical symbols.
- When coding, write concise code with helpful comments.

"Extra Credit"

For every 4 points you earn **beyond** 15, we'll give you a homework point. So, if you earn 19 points, you'll earn full credit on the midterm and 1 homework point. If you earn all 23 points, you'll get full credit on the midterm and 2 homework points.

Permissible Resources

You are welcome to consult any pre-existing resources, including Newman, lecture notes and videos from this course, and anything you can find on the internet. Please cite your sources when you use them, including a URL when applicable. You **may not** seek help from other human beings by discussing the exam or soliciting help on online forums such as Chegg or StackExchange.

You are welcome to ask me questions of clarification. You can also ask me hints, although these will usually come with a "point cost" attached to them.

Mathematical Setup

So far in MATH 168, we have seen several methods for *ranking* nodes in graphs. These include betweenness centrality, PageRank, and Katz centrality. As you might expect, there are many others, and in this problem we will study one of them.

The famous *Cats Centrality* (CC, not to be confused with Katz centrality) is named for its primary historical application area of ranking famous cats on the internet. Like many other algorithms, CC ranks cats based on a series of pairwise comparisons. In standard deployments, users are shown pairs of cats, and asked to pick which one they like more. This process is repeated many times to form a *directed network* of pairwise preferences. Figure 1 shows an example of such a directed network. Here, an arrow $i \to j$ means that a user preferred cat j to cat i. Popular cats are cats with lots of incoming arrows and few outgoing arrows.

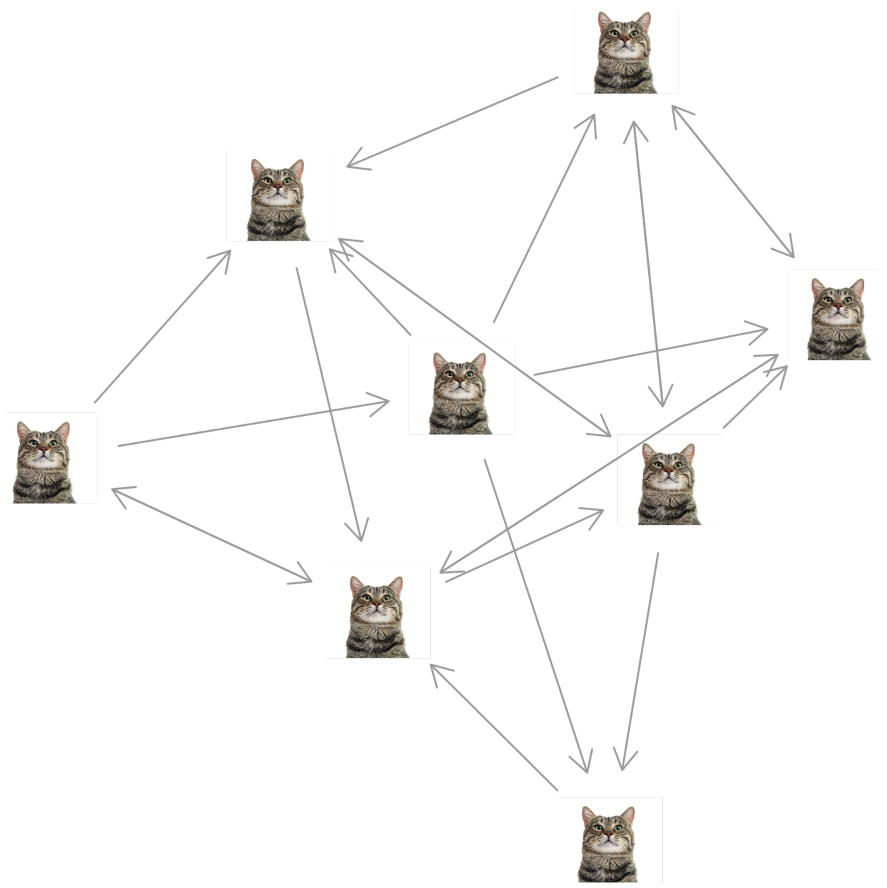

Figure 1: A directed network of pairwise preferences. An arrow $i \to j$ means that a user preferred j to i. Links with arrows on both sides mean that one user preferred i to j and one user preferred j to i.

Consider a set of n cats. Let a_{ij} be the number of times that a user preferred cat j to cat i. We can collect all these rankings in a *directed adjacency matrix* $\mathbf{A} \in \mathbb{Z}_+^{n \times n}$. Note that $\mathbf{A}$ is not symmetric: in general, $a_{ij} \neq a_{ji}$. Additionally, we can have $a_{ij} > 1$ if multiple users preferred cat j to cat i. We assume that there are no self-loops, i.e. $a_{ii} = 0$ for all i.

Unlike methods we've seen based on random walks, counting paths, or eigenvector calculations, Cats Centrality is an *optimization* model. The aim is to assign to each cat i a rank $s_i \in \mathbb{R}$ such that a higher rank means an overall more preferable cat. The CC objective is a function that takes a vector of rankings $\mathbf{s} \in \mathbb{R}^n$ and returns a real number reflecting the quality of the ranking. Its mathematical structure is

$$\mathrm{CC}(\mathbf{s}) = \sum_{i \in N} \sum_{j \in N} a_{ij}(s_j - s_i - 1)^2 + \lambda \sum_{i \in N} s_i^2 \ , \tag{1}$$

where $\lambda \geq 0$ is a scalar *regularization parameter* that is usually chosen to be very small. The rankings are computed by solving the optimization problem

$$\hat{\mathbf{s}} = \mathrm{argmin}_{\mathbf{s} \in \mathbb{R}^n} \mathrm{CC}(\mathbf{s}) \ . \tag{2}$$

That is, the ranking vector $\hat{\mathbf{s}}$ is a vector that makes the objective function as small as possible.

Problem 1. (2 points, Path 1 and Path 2)

Assume in this problem that $\lambda = 0$ (this is called *unregularized Cats Centrality*). Consider the simple network of pairwise comparisons shown in Figure 2. Each arrow corresponds to a single pairwise comparison. Find a ranking vector $\mathbf{s} \in \mathbb{R}^6$ that achieves $\mathrm{CC}(\mathbf{s}) = 0$, and explain why this is the smallest possible value that the CC objective function can achieve. Then, find a **second** ranking vector $\mathbf{s}'$ that also achieves $\mathrm{CC}(\mathbf{s}') = 0$.

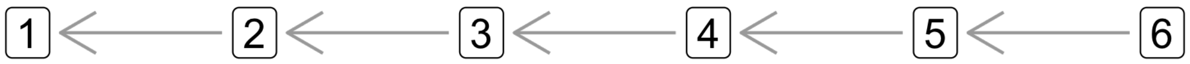
$$\begin{tikzcd} \boxed{1} & \boxed{2} \arrow[l] & \boxed{3} \arrow[l] & \boxed{4} \arrow[l] & \boxed{5} \arrow[l] & \boxed{6} \arrow[l] \end{tikzcd}$$

Figure 2: A path network of pairwise comparisons.

Hint. Remember, an arrow $i \to j$ means that j is preferred to i. So, in your solution, the node with the highest score should be node 1.

Problem 2. (5 points, Path 1)

Show by direct calculation that the equation $\nabla \mathrm{CC}(\mathbf{s}) = 0$, with the CC objective function given by (1), is equivalent to the following linear system:

$$\left[\mathbf{K}^{\text{out}} + \mathbf{K}^{\text{in}} - (\mathbf{A} + \mathbf{A}^T) + \lambda \mathbb{I}\right] \mathbf{s} = [\mathbf{K}^{\text{in}} - \mathbf{K}^{\text{out}}]\mathbf{1}\ . \tag{3}$$

Here,

- $\mathbf{K}^{\text{in}}$ is a diagonal matrix. Its diagonal entries k_i^{in} count the number of times node i was preferred in a pairwise comparison, also called the *in-degree*. The in-degree can be computed as $k_i^{\text{in}} = \sum_{j \in N} a_{ji}$.
- Similarly, $\mathbf{K}^{\text{out}}$ is a diagonal matrix. Its diagonal entries k_i^{out} count the number of times node i was **not** preferred in a pairwise comparison, also called the *out-degree*. The out-degree be computed as $k_i^{\text{out}} = \sum_{j \in N} a_{ij}$.
- $\mathbb{I}$ is the matrix identity.
- $\mathbf{1} = (1, 1, \dots, 1)^T$ is the vector of n copies of the number 1.

Note. This calculation shows that solving (3) for $\mathbf{s}$ will yield a critical point of the Cats Centrality objective function (1). It's true that, if $\lambda > 0$, this critical point is in fact the *global minimum*. Demonstrating this is the subject of Problem 6.

Hint. When performing this calculation, you might find it useful to remember that $a_{ii} = 0$ for all $i \in N$.

Problem 3. (3 points, Path 1 and Path 2)

Although this problem involves an equation shown in Problem 2, you don't need to complete Problem 2 to complete this problem.

Assume that $\lambda > 0$. Suppose that $\mathbf{A} = \mathbf{A}^T$. This scenario corresponds to the case that, for every pair (i, j), cat j is preferred to cat i exactly as often as cat i is preferred to cat j. Using (3), prove that $\mathbf{s} = \mathbf{0}$. That is, all cats have the same Cats Centrality in this case.

Hint. Recall that, in order to prove that $\mathbf{x} = \mathbf{0}$ is the unique solution of the linear system $\mathbf{M}\mathbf{x} = \mathbf{b}$, it is necessary to prove something about *both* $\mathbf{M}$ and $\mathbf{b}$.

Problem 4. (2 points, Path 1)

Let $\mathbf{s}(\lambda)$ be the solution of (3) for fixed λ. Consider the vector

$$\mathbf{t} \triangleq \lim_{\lambda \to \infty} \lambda \mathbf{s}(\lambda) \ . \tag{4}$$

Give a formula for $\mathbf{t}$ in terms of the vectors and matrices appearing in (3). You should explain how you came to your answer, but you do not need to give a formal proof.

Note. Intuitively, this result shows that choosing λ too large makes the Cats Centrality kind of boring. This is the reason that we usually choose small values of λ.

Problem 5. (2 points, Path 1 and Path 2)

Consider the unregularized Cats Centrality, which corresponds to $\lambda = 0$. Prove that, if $\mathbf{s}$ is a solution of (3) in this case, then so is $\mathbf{s} + \alpha\mathbf{1}$ for any α. Here, $\mathbf{1}$ is the vector containing all ones.

Note. This property is often called *translation invariance*. In the context of data analysis, it's usually considered to be a nuisance, since it means that there isn't a single "right" answer. The purpose of setting $\lambda > 0$ in practice is to remove translation invariance.

Problem 6. (1 point, Path 1)

Prove using any method that the Hessian (matrix of second derivatives) of the Cats Centrality objective function CC with respect to the entries of $\mathbf{s}$ is positive-definite provided that $\lambda > 0$.

Note. It follows from this fact that the solution to (3) is in fact the unique global minimizer of the the the Cats Centrality objective (1). So, if $\lambda > 0$, there is a *unique* set of rankings returned by the Cats Centrality algorithm.

Hint. Can you relate the matrix $\mathbf{K}^{\text{out}} + \mathbf{K}^{\text{in}} - (\mathbf{A} + \mathbf{A}^T)$ to a matrix we've seen before?

Problem 7. (5 points, Path 2)

In this problem you need to read and understand the linear system in (3). However, you can complete this problem without completing Problem 2.

In Python or your favorite programming language, write a function called `cats_centrality` that implements the regularized Cats Centrality of a directed network. Your function should accept two arguments: a directed graph of pairwise comparisons and the regularization parameter $\lambda > 0$. Here's what to do:

- First, write a helper function called `compute_matrices` that computes and returns the matrices that appear in (3).
- In your `cats_centrality` function, first call `compute_matrices` to get all the matrices you need to define the linear system (3). Then, use a standard function for solving linear systems to find the solution $\mathbf{s}$.
- Return the solution $\mathbf{s}$.

Include comments throughout your code to help us understand what you're doing. Additionally, please spend some effort to make your code as simple and brief as reasonably possible. For full credit, please ensure that `compute_matrices` is **under 15 lines of code** (excluding comments), and that `cats_centrality` is **under 5 lines**. **Each line should be under 80 characters long**. In fact, it's possible to implement both solutions in many fewer lines than that, especially if you are comfortable with Numpy array programming or its equivalents. My own solutions were 5 and 2 lines, respectively.

For this problem, all you need to do is submit carefully documented code – no outputs necessary.

Hint. In Python with NetworkX and Numpy, the following code will compute the adjacency matrix of a directed graph `G` as a 2D Numpy array:

```
A = nx.to_numpy_array()
```

Hint. If you are less comfortable constructing matrices, vectors, and linear systems in Python, you might benefit from these lecture notes. If you are less comfortable with creating and manipulating arrays in general, you might benefit from this book chapter.

Problem 8. (3 points, Path 2)

The data at the URL below contains a list of edges for a network of directed comparisons.

```
https://raw.githubusercontent.com/PhilChodrow/
intro-networks/main/data/sample-data.csv
```

It has a `source` column and a `target` column. In each case, the `target` column is preferred to the `source` column.

Read this data into your software of choice. Then, compute the Cats Centrality for each node in the data, using your implementation from the previous problem. You can use the regularization parameter $\lambda = 0.1$. Then,

- Create a network visualization in which the size or color of each node corresponds to its Cats Centrality. Please make sure that edges in your visualization are appropriately labelled with arrows to indicate the direction of comparison.
- Submit the image in the accompanying data directory corresponding to the node with the highest Cats Centrality. For example, if node 0 is the highest-ranked, please show the image `cats/0-maru.jpeg` in your submission.

Hint. If you are working in Python with NetworkX, the following code will correctly construct a directed network from the data.

```
url = "https://raw.githubusercontent.com/PhilChodrow/
intro-networks/main/data/sample-data.csv"
df = pd.read_csv(url)
G = nx.from_pandas_edgelist(df, create_using = nx.DiGraph)
```

If you run into problems related to accessing the data directly from online, you can instead download the data and read it in from your local filesystem.

Figure 3: The current #BestCat on the entire internet, as determined by Cats Centrality. Image source.