Alex Lemon
EE263 (Introduction to Linear Dynamical Systems)
Homework 8
August 14, 2013

1 Linear system with one-bit quantized output

Consider the autonomous linear dynamical system

$$\dot{x}(t) = Ax(t), \qquad y(t) = \operatorname{sgn}(cx(t)),$$

where

$$A = \begin{bmatrix} -0.1 & 1.0 \\ -1.0 & 0.1 \end{bmatrix}, \qquad \text{and} \qquad c = \begin{bmatrix} 1 & -1 \end{bmatrix}.$$

Intuitively, the output of our system is quantized with one bit of precision. The following outputs are observed:

$$y(0.4) = +1, \qquad y(1.2) = -1, \qquad y(2.3) = -1, \qquad \text{and} \qquad y(3.8) = +1.$$

What, if anything, can you say about

$$y(0.7), \qquad y(1.8), \qquad \text{and} \qquad y(3.7)?$$

2 A search engine using low-rank approximation

In this problem we examine how linear algebra and low-rank approximation can be used to find matches to a search query in a set of documents. Consider a corpus with n documents, and a vocabulary of m terms. We define the term-by-document matrix $A \in \mathbb{R}^{m \times n}$ such that

$$a_{ij} = \text{ the number of times term } i \text{ occurs in document } j.$$

A query is a vector $q \in \mathbb{R}^m$ that expresses a criterion by which to select a document. We will consider query vectors that have 1s in the entries corresponding to the words that we want to search for, and 0s in all the other entries; however, more complex weighting schemes are possible. A simple measure of the relevance of document j to the query is given by

$$a_j^\mathsf{T} q,$$

where a_j denotes the jth column of A. This criterion is biased towards large documents. For this reason, we use a normalized version of the inner product in order to rank the relevance of the documents:

$$\frac{a_j^\mathsf{T} q}{\|a_j\| \|q\|}.$$

Note that this normalized relevance criterion is the cosine of the angle between the document and query vectors. Since all the entries of the query and document vectors are nonnegative, our normalized relevance criterion lies between 0 and 1. Let $\tilde{A}$ denote the normalized term-by-document matrix:

$$\tilde{A} = \begin{bmatrix} \frac{1}{\|a_1\|} a_1 & \cdots & \frac{1}{\|a_n\|} a_n. \end{bmatrix}$$

If $\tilde{q} = \frac{1}{\|q\|} q$ is a normalized query vector, then

$$c = \tilde{A}^{\mathsf{T}} \tilde{q}$$

is a column vector giving an estimate of the relevance of each document to the query. The file `search_engine_data.m` defines the following variables.

- `term`, an $m \times 1$ cell array containing the search terms
- `document`, an $n \times 1$ cell array containing the URLs of the documents
- `A`, the $m \times n$ term-by-document matrix

(a) Compute the normalized term-by-document matrix, $\tilde{A}$. Make a stem plot of the singular values of $\tilde{A}$.

(b) Perform a query for the word "students" ($i = 53$). What are the top 5 results?

(c) Let $\hat{A}_r$ be the best rank-r approximation of $\tilde{A}$. Compute $\hat{A}_{32}$, $\hat{A}_{16}$, $\hat{A}_8$, and $\hat{A}_4$. Perform a query for the word "students" using each of these matrices. Comment on the results.

(d) Are there advantages to using low-rank approximations rather than the full term-by-document matrix? You may assume that a very large number of searches will be performed before the term-by-document matrix is updated.

3 A heuristic for the maximum-cut problem

Consider an undirected graph with n nodes and m edges. Suppose B and C partition the set of nodes: that is, $B \cap C = \emptyset$, and $B \cup C = \{1, \ldots, n\}$. The number of cuts associated with this partition is the number of edges with between a node in B and node in C. The maximum-cut problem is the task of choosing a partition that maximizes the number of cuts for a given graph. For any partition, the number of cuts is no more than m; if the number of cuts is equal to m, then the graph is bipartite.

The maximum-cut problem has many applications. We describe one such application here, although you do not need to know about the application in order to solve the problem that follows. Suppose we have a communication system that operates with a two-phase clock. During the even periods $t = 0, 2, 4, \ldots$, each node in B transmits data to its neighbors in C; during the odd periods $t = 1, 3, 5, \ldots$, each node in C transmits data to its neighbors in B. The number of cuts is the number of successful transmissions that can occur in a two-period cycle. The maximum-cut problem is to assign nodes to the two groups in order to maximize the overall efficiency of communication.

The maximum-cut problem is hard to solve exactly, at least if we do not want to try all, or even most, of the 2^n possible partitions. In this problem, we explore an approximation algorithm for the maximum-cut problem (that is, a method of finding a partition that has a large number of cuts, although the corresponding partition may not have the maximum number of cuts.)

We can encode a partition using a vector $x \in \{-1, +1\}^n$: the associated partition has $i \in B$ if $x_i = +1$, and $i \in C$ if $x_i = -1$. We can describe the graph using its adjacency matrix $A \in \mathbb{R}^{n \times n}$, where

$$A_{ij} = \begin{cases} 1 & \text{there is an edge between nodes } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases}$$

Note that A is symmetric, and $A_{ii} = 0$ for $i = 1, \ldots, n$.

(a) Find a symmetric matrix $P \in \mathbb{S}^n$ such that $P_{ii} = 0$ for $i = 1, \ldots, n$, and a constant $d \in \mathbb{R}$ such that $x^\mathsf{T} P x + d$ is the number of cuts corresponding to the partition encoded by the vector x. Then, the maximum-cut problem can be written as

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{maximize}} & : x^\mathsf{T} P x + d \\ \text{subject to} & : x_i^2 = 1 \qquad i = 1, \ldots, n. \end{array}$$

(b) A famous heuristic for the maximum-cut problem is to replace the n constraints $x_i^2 = 1$ with the single constraint $\sum_{i=1}^n x_i^2 = n$. This gives us the relaxed problem

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{maximize}} & : x^\mathsf{T} P x + d \\ \text{subject to} & : \sum_{i=1}^n x_i^2 = n. \end{array}$$

Explain how to solve the relaxed problem. Let $x^\star$ be a solution of the relaxed problem. We can round $x^\star$ to a vector $x \in \{-1, +1\}^n$ describing a partition by taking

$$x_i = \begin{cases} +1 & x_i^\star \geq 0, \\ -1 & x_i^\star < 0. \end{cases}$$

(c) Apply your method to the data in `maximum_cut_heuristic_data.m`. What is the number of cuts of the partition that you find?

Another heuristic for the maximum-cut problem is to generate a large number of random partitions, and take the one that yields the largest number of cuts. In `MATLAB`, we can generate a random partition using the following code.

```
x = sign(rand(n,1) - 0.5);
```

Generate $N = 1000$ such random partitions, and report the largest number of cuts achieved by one of the random partitions.

4 Simultaneously estimating student ability and exercise difficulty.

Each of n students takes an exam that contains m questions. Student j receives a grade $G_{ij} \geq 0$ on question i. One simple model for predicting grades is to estimate

$$G_{ij} \approx \hat{G}_{ij} = \frac{a_j}{d_i},$$

where $a_j \geq 0$ is the ability of student j, and $d_i > 0$ is the difficulty of question i. To ensure a unique model, we normalize the exam question difficulties so that the mean exam question difficulty across the m questions is 1; otherwise, a_j and d_i would not be uniquely determined because we can scale them both by any positive constant without changing our estimates.

In this problem you are given the matrix $G \in \mathbb{R}^{m \times n}$ of grades. Your task is to find a set of nonnegative student abilities, and a set of positive, normalized question difficulties such that $G_{ij} \approx \hat{G}_{ij}$. In particular, choose your model in order to minimize the root mean squared error:

$$J = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (G_{ij} - \hat{G}_{ij})^2}.$$

(a) Explain how to solve this problem. You can ignore the constraints that the a_j be nonnegative, and the d_i be positive.

(b) Carry out your method on the data in `ability_difficulty_data.m`. Report the optimal value of J, the ratio of the optimal value of J and the root mean squared value of G_{ij}; give the difficulties of the exam questions.