Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 14, 2018
Marco Chiarandini

DM559 – Linear and Integer Programming

Sheet 2, Spring 2018 [pdf format]

Exercise 1*

True-False exercises on page 42 of [AR].

(a) Two $n \times n$ matrices, A and B, are inverses of one another if and only if $AB = BA = 0$.

(b) For all square matrices A and B of the same size, it is true that $(A + B)^2 = A^2 + 2AB + B^2$.

(c) For all square matrices A and B of the same size, it is true that $A^2 - B^2 = (A - B)(A + B)$.

(d) If A and B are invertible matrices of the same size, then AB is invertible and $(AB)^{-1} = A^{-1}B^{-1}$.

(e) If A and B are matrices such that AB is defined, then it is true that $(AB)^T = A^T B^T$.

(f) The matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is invertible if and only if $ad - bc \neq 0$.

(g) If A and B are matrices of the same size and k is a constant, then $(kA + B)^T = kA^T + B^T$.

(h) If A is an invertible matrix, then so is A^T.

(i) If $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$ and I is an identity matrix, then $p(I) = a_0 + a_1 + a_2 + \cdots + a_m$.

(j) A square matrix containing a row or column of zeros cannot be invertible.

(k) The sum of two invertible matrices of the same size must be invertible.

Exercise 2*

Given the matrices:

$$A\begin{bmatrix} 1 & 3 & 5 \\ -1 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 1 \\ 3 & 2 \\ -1 & 4 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

which of the following matrix epxressions are defined? Compute those which are defined.

a) $A\mathbf{d}$

b) $AB + C$

c) $A + C^T$

d) $C^T C$

e) BC

f) $\mathbf{d}^T B$

g) $C\mathbf{d}$

h) $\mathbf{d}^T\mathbf{d}$

i) $\mathbf{d}\mathbf{d}^T$

j) A/C

k) C/A

Exercise 3*

a. If A and B are invertible $n \times n$ matrices, then using the definition of the inverse, prove that

$$(AB)^{-1} = B^{-1}A^{-1}$$

b. Prove that

$$(AB)^T = B^T A^T$$

c. Using the previous result, show that if A is non singular then A^T is nonsingular and

$$(A^T)^{-1} = (A^{-1})^T.$$

Exercise 4* Given

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

show that R is nonsingular and $R^{-1} = R^T$.

Exercise 5

Let A be an $n \times n$ matrix and let

$$B = A + A^T \quad \text{and} \quad C = A - A^T$$

1. Show that B is symmetric and C is skew symmetric (a matrix is *skew symmetric* if $A^T = -A$)
2. Show that every $n \times n$ matrix can be represented as a sum of a symmetric matrix and a skew-symmetric matrix.[1]

Exercise 6 Information Retrieval

The following is Application 2 of Section 1.3 from [Le]

[1] Exercise 30 of Section 1.4 of [Le] ed 8th p 57

The growth of digital libraries on the Internet has led to dramatic improvements in the storage and retrieval of information. Modern retrieval methods are based on matrix theory and linear algebra.

In a typical situation, a database consists of a collection of documents and we wish to search the collection and find the documents that best match some particular search conditions. Depending on the type of database, we could search for such items as research articles in journals, Web pages on the Internet, books in a library, or movies in a film collection.

To see how the searches are done, let us assume that our database consists of m documents and that there are n dictionary words that can be used as keywords for searches. Not all words are allowable, since it would not be practical to search for common words such as articles or prepositions. If the key dictionary words are ordered alphabetically, then we can represent the database by an $m \times n$ matrix A. Each document is represented by a column of the matrix. The first entry in the jth column of A would be a number representing the relative frequency of the first key dictionary word in the jth document. The entry a_{2j} represents the relative frequency of the second word in the jth document, and so on. The list of keywords to be used in the search is represented by a vector $\mathbf{x}$ in $\mathbb{R}^m$. The ith entry of $\mathbf{x}$ is taken to be 1 if the ith word in the list of keywords is on our search list; otherwise, we set $x_i = 0$. To carry out the search, we simply multiply A^T times $\mathbf{x}$.

Simple Matching Searches

The simplest type of search determines how many of the key search words are in each document; it does not take into account the relative frequencies of the words. Suppose, for example, that our database consists of these book titles:

B1. *Applied Linear Algebra*
B2. *Elementary Linear Algebra*
B3. *Elementary Linear Algebra with Applications*
B4. *Linear Algebra and Its Applications*
B5. *Linear Algebra with Applications*

B6. *Matrix Algebra with Applications*
B7. *Matrix Theory*

The collection of keywords is given by the following alphabetical list:

algebra, *application*, *elementary*, *linear*, *matrix*, *theory*

For a simple matching search, we just use 0's and 1's, rather than relative frequencies for the entries of the database matrix. Thus, the (i, j) entry of the matrix will be 1 if the ith word appears in the title of the jth book and 0 if it does not. We will assume that our search engine is sophisticated enough to equate various forms of a word. So, for example, in our list of titles the words *applied* and *applications* are both counted as forms of the word *application*. The database matrix for our list of books is the array defined by Table 4.

Table 4 Array Representation for Database of Linear Algebra Books

	Books						
Key Words	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**
algebra	1	1	1	1	1	1	0
application	1	0	1	1	1	1	0
elementary	0	1	1	0	0	0	0
linear	1	1	1	1	1	0	0
matrix	0	0	0	0	0	1	1
theory	0	0	0	0	0	0	1

If the words we are searching for are *applied, linear*, and *algebra*, then the database matrix and search vector are respectively given by

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

If we set $\mathbf{y} = A^T\mathbf{x}$, then

$$\mathbf{y} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 3 \\ 2 \\ 0 \end{bmatrix}$$

The value of y_1 is the number of search word matches in the title of the first book, the value of y_2 is the number of matches in the second book title, and so on. Since

$y_1 = y_3 = y_4 = y_5 = 3$, the titles of books B1, B3, B4, and B5 must contain all three search words. If the search is set up to find titles matching all search words, then the search engine will report the titles of the first, third, fourth, and fifth books.

Relative-Frequency Searches

Searches of noncommercial databases generally find all documents containing the key search words and then order the documents based on the relative frequencies of the keywords. In this case, the entries of the database matrix should represent the relative frequencies of the keywords in the documents. For example, suppose that in the dictionary of all key words of the database the 6th word is *algebra* and the 8th word is *applied*, where all words are listed alphabetically. If, say, document 9 in the database contains a total of 200 occurrences of keywords from the dictionary, and if the word *algebra* occurred 10 times in the document and the word *applied* occurred 6 times, then the relative frequencies for these words would be $\frac{10}{200}$ and $\frac{6}{200}$, and the corresponding entries in the database matrix would be

$$a_{69} = 0.05 \quad \text{and} \quad a_{89} = 0.03$$

To search for these two words, we take our search vector $\mathbf{x}$ to be the vector whose entries x_6 and x_8 are both equal to 1 and whose remaining entries are all 0. We then compute

$$\mathbf{y} = A^T\mathbf{x}$$

The entry of $\mathbf{y}$ corresponding to document 9 is

$$y_9 = a_{69} \cdot 1 + a_{89} \cdot 1 = 0.08$$

Note that 16 of the 200 words (8% of the words) in document 9 match the key search words. If y_j is the largest entry of $\mathbf{y}$, this would indicate that the jth document in the database is the one that contains the keywords with the greatest relative frequencies.

Advanced Search Methods

A search for the keywords *linear* and *algebra* could easily turn up hundreds of documents, some of which may not even be about linear algebra. If we were to increase the number of search words and require that all search words be matched, then we would run the risk of excluding some crucial linear algebra documents. Rather than match all words of the expanded search list, our database search should give priority to those documents which match most of the keywords with high relative frequencies. To accomplish this, we need to find the columns of the database matrix A that are "closest" to the search vector $\mathbf{x}$. One way to measure how close two vectors are is to define *the angle between the vectors*. We will do this in Section 1 of Chapter 5.

We will also revisit the information retrieval application after we have learned about the *singular value decomposition* (Chapter 6, Section 5). This decomposition can be used to find a simpler approximation to the database matrix, which will speed up the searches dramatically. Often it has the added advantage of filtering out *noise*; that is, using the approximate version of the database matrix may automatically have the effect of eliminating documents that use keywords in unwanted contexts. For example, a dental student and a mathematics student could both use *calculus* as one of their

search words. Since the list of mathematics search words does not contain any other dental terms, a mathematics search using an approximate database matrix is likely to eliminate all documents relating to dentistry. Similarly, the mathematics documents would be filtered out in the dental student's search.

Web Searches and Page Ranking

Modern Web searches could easily involve billions of documents with hundreds of thousands of keywords. Indeed, as of July 2008, there were more than 1 trillion Web pages on the Internet, and it is not uncommon for search engines to acquire or update as many as 10 million Web pages in a single day. Although the database matrix for pages on the Internet is extremely large, searches can be simplified dramatically, since the matrices and search vectors are *sparse*; that is, most of the entries in any column are 0's.

For Internet searches, the better search engines will do simple matching searches to find all pages matching the keywords, but they will not order them on the basis of the relative frequencies of the keywords. Because of the commercial nature of the Internet, people who want to sell products may deliberately make repeated use of keywords to ensure that their Web site is highly ranked in any relative-frequency search. In fact, it is easy to surreptitiously list a keyword hundreds of times. If the font color of the word matches the background color of the page, then the viewer will not be aware that the word is listed repeatedly.

For Web searches, a more sophisticated algorithm is necessary for ranking the pages that contain all of the key search words. In Chapter 6, we will study a special type of matrix model for assigning probabilities in certain random processes. This type of model is referred to as a *Markov process* or a *Markov chain*. In Section 3 of Chapter 6, we will see how to use Markov chains to model Web surfing and obtain rankings of Web pages.

References

1. Berry, Michael W., and Murray Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, SIAM, Philadelphia, 1999.

Exercise 7

Let

$$A = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

a) Calculate $A\mathbf{b}_1$ and $A\mathbf{b}_2$

b) Calculate $\vec{\mathbf{a}}_1 B$ and $\vec{\mathbf{a}}_2 B$.

c) Multiply AB, and verify that its column vectors are the vectors in part (a) and its row vectors are the vectors in part (b).

Exercise 8 Block multiplication

In the first exercise session, we have seen that a matrix A of size $m \times n$ can be represented also by its column vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ and by its row vectors $\vec{\mathbf{a}_1}, \vec{\mathbf{a}_2}, \ldots, \vec{\mathbf{a}_m}$, (or for an alternative notation: $\mathbf{a}_{\cdot 1}, \mathbf{a}_{\cdot 2}, \ldots, \mathbf{a}_{\cdot n}$ and by its row vectors $\mathbf{a}_{1\cdot}, \mathbf{a}_{2\cdot}, \ldots, \mathbf{a}_{m\cdot}$), that is:

$$A = \left[\begin{array}{c|c|c|c} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{array}\right] = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix} = \left[\begin{array}{c|c|c|c} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2n} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline a_{m1} & a_{m2} & \cdots & a_{mn} \end{array}\right] = \begin{bmatrix} \vec{\mathbf{a}_1} \\ \vec{\mathbf{a}_2} \\ \vdots \\ \vec{\mathbf{a}_m} \end{bmatrix}$$

and we have shown that matrix multiplication can be rewritten using row and column vectors and carried out in different steps. That is, if A is a matrix $m \times n$ and B is a mtrix $n \times r$, $AB = [A\mathbf{b}_1, A\mathbf{b}_2, \ldots, A\mathbf{b}_r] = [\vec{\mathbf{a}}_1 B, \vec{\mathbf{a}}_2 B, \ldots, \vec{\mathbf{a}}_n B]^T$.
This idea of *partitioning matrices* in submatrices can be generalized. For example, let:

$$C = \begin{bmatrix} 1 & 1 & 9 & 4 & 5 \\ 9 & 1 & 1 & 6 & 3 \\ 0 & 1 & 8 & 1 & 5 \\ 3 & 2 & 2 & 2 & 2 \end{bmatrix}$$

then C can be subdivided into four submatrices $C_{11}, C_{12}, C_{21}, C_{22}$:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \left[\begin{array}{ccc|cc} 1 & 1 & 9 & 4 & 5 \\ 9 & 1 & 1 & 6 & 3 \\ \hline 0 & 1 & 8 & 1 & 5 \\ 3 & 2 & 2 & 2 & 2 \end{array}\right]$$

Let A be a matrix $m \times n$ and B be a mtrix $n \times r$. Show that:

a) if $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$ where B_1 is an $n \times t$ matrix and B_2 is an $n \times (r - t)$ matrix, then

$$AB = A\begin{bmatrix} B_1 & B_2 \end{bmatrix} = \begin{bmatrix} AB_1 & AB_2 \end{bmatrix}$$

b) if $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, where A_1 is an $k \times n$ matrix and A_2 is an $(m - k) \times n$ matrix, then

$$AB = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} B = \begin{bmatrix} A_1 B \\ A_2 B \end{bmatrix}$$

c) if $A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$ and $B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$, where A_1 is an $m \times s$ matrix and A_2 is an $m \times (n - s)$ matrix, B_1 is an $s \times r$ matrix and B_2 is an $(n - s) \times r$ matrix, then

$$AB = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = A_1 B_1 + A_2 B_2$$

d) If A and B be are both partitioned as follows:

$$\mathrm{A}=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{matrix} k \\ m-k \end{matrix}, \qquad \mathrm{B}=\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}\begin{matrix} s \\ n-s \end{matrix},$$
$$\begin{matrix} s & n-s \end{matrix} \qquad \begin{matrix} t & r-t \end{matrix}$$

then

$$AB=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}=\begin{bmatrix} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{bmatrix}$$

Exercise 9 Matrix Multiplication

a Design an algorithm for efficient matrix multiplication.

b In the previous exercise we showed that if A, B and C are n by n (square) matrices, and A_{11}, B_{11}, A_{12}, B_{12} etc. are $n/2$ by $n/2$ submatrices the following decomposition holds:

$$C=AB=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}=\begin{bmatrix} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{bmatrix}$$

How could you use this fact in your algorithm for matrix multiplication? How would you design your algorithm for a parallel computing environment?

What design would minimize the amount of data transferred between RAM and cache on a single machine or the amount transferred between nodes on a distributed memory multi-node machine?