Systems of Linear Equations
Duke COMPSCI 309s

Siyang Chen

Spring 2014
A system of linear equations is the following:

\[ Ax = b \]

Where \( A \) is an \( n \times m \) matrix, \( x \) is a \( m \)-dimensional vector, and \( b \) is an \( n \)-dimensional vector.
Brief Linear Algebra Review

The standard way of solving such a system is via a process called Gaussian elimination:

- $r \leftarrow 0$
- For $j \in \{1, \ldots, m\}$:
  - $r' \leftarrow \arg\max_{r'>r} \{A_{r',j}\}$
  - If $A_{r',j} = 0$:
    - Continue
  - Swap $A_{r'}$ and $A_r$
  - $A_r \leftarrow (1/A_{r,j}) \times A_r$
  - For $i \in \{1, \ldots, r - 1, r + 1, \ldots, n\}$:
    - $A_i \leftarrow A_i - (A_{i,j}/A_{r,j}) \times A_r$
- $r \leftarrow r + 1$

What is $r$ when the algorithm terminates?

The rank of the matrix.

Complexity? $O(n^3)$
Brief Linear Algebra Review

The standard way of solving such a system is via a process called Gaussian elimination:

- $r \leftarrow 0$
- For $j \in \{1, \ldots, m\}$:
  - $r' \leftarrow \arg \max_{r' > r} \{A_{r', j}\}$
  - If $A_{r', j} = 0$:
    - Continue
  - Swap $A_{r'}$ and $A_r$
  - $A_r \leftarrow (1/A_{r,j}) \times A_r$
- For $i \in \{1, \ldots, r - 1, r + 1, \ldots, n\}$:
  - $A_i \leftarrow A_i - (A_{i,j}/A_{r,j}) \times A_r$
- $r \leftarrow r + 1$

What is $r$ when the algorithm terminates?
Brief Linear Algebra Review

The standard way of solving such a system is via a process called *Gaussian elimination*:

- \( r \leftarrow 0 \)
- For \( j \in \{1, \ldots, m\} \):
  - \( r' \leftarrow \arg \max_{r', r} \{A_{r', j}\} \)
  - If \( A_{r', j} = 0 \):
    - Continue
  - Swap \( A_{r'} \) and \( A_r \)
  - \( A_r \leftarrow (1/A_{r, j}) \times A_r \)
  - For \( i \in \{1, \ldots, r - 1, r + 1, \ldots, n\} \):
    - \( A_i \leftarrow A_i - (A_{i, j}/A_{r, j}) \times A_r \)
  - \( r \leftarrow r + 1 \)

What is \( r \) when the algorithm terminates? The *rank* of the matrix.
Brief Linear Algebra Review

The standard way of solving such a system is via a process called Gaussian elimination:

- $r \leftarrow 0$
- For $j \in \{1, \ldots, m\}$:
  - $r' \leftarrow \arg\max_{r', > r} \{A_{r', j}\}$
  - If $A_{r', j} = 0$:
    - Continue
  - Swap $A_{r'}$ and $A_r$
  - $A_r \leftarrow (1/A_{r, j}) \times A_r$
  - For $i \in \{1, \ldots, r - 1, r + 1, \ldots, n\}$:
    - $A_i \leftarrow A_i - (A_{i, j}/A_{r, j}) \times A_r$
  - $r \leftarrow r + 1$

What is $r$ when the algorithm terminates? The rank of the matrix. Complexity?
Brief Linear Algebra Review

The standard way of solving such a system is via a process called **Gaussian elimination**:

1. \( r \leftarrow 0 \)
2. For \( j \in \{1, \ldots, m\} \):
   1. \( r' \leftarrow \arg \max_{r' > r} \{ A_{r', j} \} \)
   2. If \( A_{r', j} = 0 \):
      1. Continue
   3. Swap \( A_{r'} \) and \( A_r \)
   4. \( A_r \leftarrow (1/A_{r, j}) \times A_r \)
   5. For \( i \in \{1, \ldots, r - 1, r + 1, \ldots, n\} \):
      1. \( A_i \leftarrow A_i - (A_{i, j}/A_{r, j}) \times A_r \)
   6. \( r \leftarrow r + 1 \)

What is \( r \) when the algorithm terminates? The **rank** of the matrix.

Complexity? \( O(nm^2) \)
In case you forgot:

- The *rank* of a matrix is the dimension of its *row space* (the subspace generated by its rows) or its *column space*.

- The *nullity* of a matrix is the dimension of its *null space*, the set of vectors $x$ such that $Ax = 0$. 

**Theorem (Rank-Nullity)**

Let $A$ be an $n \times m$ matrix. Then 

$$\text{rank}(A) + \text{nullity}(A) = m$$

(For a proof, consult your nearest linear algebra textbook.)
Brief Linear Algebra Review

In case you forgot:

- The *rank* of a matrix is the dimension of its *row space* (the subspace generated by its rows) or its *column space*.

- The *nullity* of a matrix is the dimension of its *null space*, the set of vectors $x$ such that $Ax = 0$.

**Theorem (Rank-Nullity)**

Let $A$ be an $n \times m$ matrix. Then

$$\text{rank}(A) + \text{nullity}(A) = m$$

(For a proof, consult your nearest linear algebra textbook.)
Lastly, Gaussian elimination, determinant computation, and lots of other linear-algebraic algorithms can be extended from $\mathbb{R}$ to arbitrary fields, most commonly $\mathbb{Z}/p\mathbb{Z}$ for some prime $p$. 
Let's look at this problem first:

▶ http://community.topcoder.com/stat?c=problem_statement&pm=11193
Let’s look at this problem first:


Solution?
Let’s look at this problem first:

▶ http://community.topcoder.com/stat?c=problem_statement&pm=11193

Solution?

▶ Consider a graph of states \((n, m)\) (where \(n\) and \(m\) represent the number of bored and not bored people, respectively) and transitions between them. This graph can be represented acyclically, so a standard dynamic programming approach will work.
TopCoder: MazeWandering

But what if the transitions can form cycles?

▶ http://community.topcoder.com/stat?c=problem_statement&pm=10005
TopCoder: MazeWandering

But what if the transitions can form cycles?

▶ http://community.topcoder.com/stat?c=problem_statement&pm=10005

Here’s where Gaussian elimination comes in handy:
But what if the transitions can form cycles?

Here’s where Gaussian elimination comes in handy:

Let $t_{i,j}$ be the expected time to reach the goal, given that our current state is $(i,j)$. 
TopCoder: MazeWandering

But what if the transitions can form cycles?


Here's where Gaussian elimination comes in handy:

- Let $t_{i,j}$ be the expected time to reach the goal, given that our current state is $(i,j)$.
- By definition, $t_{i^*,j^*} = 0$, where $(i^*,j^*)$ is the location of the goal.
TopCoder: MazeWandering

But what if the transitions can form cycles?


Here’s where Gaussian elimination comes in handy:

- Let $t_{i,j}$ be the expected time to reach the goal, given that our current state is $(i,j)$.
- By definition, $t_{i^*,j^*} = 0$, where $(i^*,j^*)$ is the location of the goal.
- For all other $(i,j)$, $t_{i,j}$ can be expressed as a linear combination of its neighbours.
TopCoder: MazeWandering

But what if the transitions can form cycles?

▶ http://community.topcoder.com/stat?c=problem_statement&pm=10005

Here’s where Gaussian elimination comes in handy:

▶ Let $t_{i,j}$ be the expected time to reach the goal, given that our current state is $(i,j)$.

▶ By definition, $t_{i^*,j^*} = 0$, where $(i^*,j^*)$ is the location of the goal.

▶ For all other $(i,j)$, $t_{i,j}$ can be expressed as a linear combination of its neighbours.

▶ Naively, Gaussian elimination can time out, using $\approx 2500^3$ operations. However, the matrix is sparse, so the runtime is instead $\approx 4 \times 2500^2$. 
Now let’s look at this problem:


But how is this related at all to systems of linear equations?

Observe that we can represent each configuration of `on' lights or `on' switches as a vector in \((\mathbb{Z}/2\mathbb{Z})^n\).

It suffices, then, to find the dimension \(D\) of the space spanned by the rows. The answer is then \(2^D\).

But \(D\) is just the rank of the matrix, which we can compute via Gaussian elimination.
Now let’s look at this problem:

▶ http://community.topcoder.com/stat?c=problem_statement&pm=6407

But how is this related at all to systems of linear equations?
Now let’s look at this problem:


But how is this related at all to systems of linear equations?

- Observe that we can represent each configuration of ‘on’ lights or ‘on’ switches as a vector in $(\mathbb{Z}/2\mathbb{Z})^n$. 
Now let’s look at this problem:


But how is this related at all to systems of linear equations?

- Observe that we can represent each configuration of ‘on’ lights or ‘on’ switches as a vector in $(\mathbb{Z}/2\mathbb{Z})^n$.
- It suffices, then, to find the dimension $D$ of the space spanned by the rows. The answer is then $2^D$. 
TopCoder: LightSwitches

Now let's look at this problem:


But how is this related at all to systems of linear equations?

- Observe that we can represent each configuration of ‘on’ lights or ‘on’ switches as a vector in \((\mathbb{Z}/2\mathbb{Z})^n\).
- It suffices, then, to find the dimension \(D\) of the space spanned by the rows. The answer is then \(2^D\).
- But \(D\) is just the rank of the matrix, which we can compute via Gaussian elimination.