AMATH 584A: Applied Linear Algebra

Marlin Figgins

Oct. 1, 2020

Introduction

Table of Contents

1 Lecture 1: Overview **1**
1.1 Matrix Decompositions 1
1.1.1 LU decomposiiton 1
1.1.2 QR decomposition 1
1.1.3 Eigenvalue Decomposition 2
1.1.4 Singular Value Decomposition 2

2 Lecture 2 **2**
2.1 Norms 3
2.2 Nullspaces and zero eigenvalues 4
2.3 Linear Operators 4

1. Lecture 1: Overview

This course will be entirely about the problem of $Ax = b$. That is, we're concerning with linear systems. In fact, many problems are of this form. In the age of data science, these variables A and x can get huge quickly. In your typical linear algebra classes, you learn to solve this with Gaussian elimination, but the reality is that this is one of the slowest ways you can solve this problem.

1.1 Matrix Decompositions

Matrix decompositions allow us to solve the problem $Ax = b$ much faster. Let's start with the case of complex square matrices $A \in \mathbb{C}^{n \times n}$.

To solve this problem with Gaussian Elimination, the cost would be on the order of $O(n^3)$. This is fine for small matrices, but immagine you're dealing with large matrices and this begins to blow up in computation time rather quickly.

1.1.1 LU decomposiiton

The LU decomposition allows us to represent our matrix A as

$$A = LU \tag{1.1}$$

where L is lower triangular and U is upper triangular. Our problem becomes

$$Ax = b \tag{1.2}$$
$$LUx = b \tag{1.3}$$
$$Ux = y \tag{1.4}$$
$$Ly = b \tag{1.5}$$

This allows us to use forward and back substituion individually which are of order $O(n^2)$ to solve this probelm. This LU decompoisition already gives a saving of order of n. This is all well and good, but what does it take to get an LU decomposition?

1.1.2 QR decomposition

We want to express our matrix A in the form

$$A = QR \tag{1.6}$$

where Q is a unitary matrix and R is upper triangular. Solving $Ax = b$ with this decomposiiton gives us,

$$QRx = b \tag{1.7}$$
$$Rx = y \tag{1.8}$$
$$Qy = b \tag{1.9}$$
$$Q^T[Qy = b] \tag{1.10}$$
$$y = Q^T b \tag{1.11}$$

1.1.3 Eigenvalue Decomposition

We can write the eigenvale decomposition as

$$A = V\Lambda V^{-1} \tag{1.12}$$

Using this to solve $Ax = b$, we get that

$$V^{-1}[V\Lambda V^{-1}x = b] \tag{1.13}$$
$$\Lambda y = V^{-1}b \tag{1.14}$$

Since Λ is diagonal, the answer is very clear here.

1.1.4 Singular Value Decomposition

The singular value decomposition is one of the most important decomposition algorithms. We decompose A as

$$A = U\Sigma V^* \tag{1.15}$$

Solving $Ax = b$,

$$U\Sigma V^* x = b \tag{1.16}$$
$$\Sigma V^* x = U^* b \tag{1.17}$$
$$\Sigma\hat{x} = \hat{b} \tag{1.18}$$

2. Lecture 2

In reality, we're often dealing with systems and matrices which are not perfectly square. Many problems are not perfectly square. In reality, very few are. The rest of these problems fall into two general categories of systems which we call underdetermined and overdetermined systems. When it comes to solving $Ax = b$ for these problems, the question is ill-posed. Though these systems may have no solutions or infinitely many solutions, most software will still be able to solve the problem $Ax = b$, how is this done?

Underdetermined systems $(m < n)$**.** These systems fundamentally have infinitely many solutions, so $Ax = b$ is instead posed as an optimization problem

$$\min_x ||x||_2 \text{ such that } Ax = b. \tag{2.1}$$

In this case, the minimization of the L^2 norm acts as a regularizer for our desired solution x.

Overdetermined systsems $(m > n)$**.** Due to the abundance of constraints, satisfying $Ax = b$ is technically impossible. In this case, we attempt to find the closest possible solution i.e.

$$\min_x ||Ax + b|| + \lambda ||x||_2 . \tag{2.2}$$

Here the L^2 norm acts as a regularizer for our solution x. We use the parameter λ as a hyperparameter which determines the relative importance of the regularizer λ. Indeed, there are several different ways to do this regularization such as using the L^1 norm.

$$\min_x ||Ax + b|| + \lambda_1 ||x||_1 + \lambda_2 ||x||_2 . \tag{2.3}$$

2.1 Norms

In short, a norm is just a way of quantifying distance. In particular, the two most interesting norms that we'll cover are the L^2 and L^1 norms.

L^2 **norm** We can define the L^2 norm of a vector x as

$$||x||_2 = \sqrt{|x_1|^2 + |x_2|^2} \tag{2.4}$$

Notice, this is the distance that we're used to in most geometric contexts.

L^1 **norm** There are also various other norms such as the L^1 norm which we denote as $||\cdot||_1$

$$||x||_1 = |x_1| + |x_2| \tag{2.5}$$

In applications, the L^1 norm tends to promote sparsity in solutions.

Other norms In general, we can compute the L^p norm of a vector x as

$$||x||_p = (|x_1|^p + |x_2|^p)^{1/p}. \tag{2.6}$$

There are also the L^∞ and L^0 norms.

2.2 Nullspaces and zero eigenvalues

Consider the problem $A^*y = 0$ where A^* is the adjoint of the matrix A. When does $Ax = b$ have a solution?

$$Ax \cdot y = b \cdot y \tag{2.7}$$
$$x \cdot A^*y = b \cdot y \tag{2.8}$$
$$b \cdot y = 0 \tag{2.9}$$

The Fredholm alternative is the statement that b is not orthogonal to y, then $Ax = b$ has no solutions.

Suppose we have the problem $Ax = b$ and a vector x_0 with 0 eigenvalue $Ax_0 = 0$. Then we can generate solutions as any vector of the form

$$x = \xi + \alpha x_0, \tag{2.10}$$

where ξ is a solution. The regularization process is an attempt to avoid this by minimizing across the vectors x_0 in the nullspace.

2.3 Linear Operators

Linear operators are communtative and associative under addition.

(i) *Commutative. (+)* $A + B = B + A$

(ii) *Associative. (+)* $A + (B + C) = (A + B) + C$

(iii) *Distributive.* $A(B + C) = AB + BC$

(iv) *Associative (·).* $(AB)C = A(BC)$

Though we have all these algebraic properties, it is important to know that multiplication is not commutative for matrices in general i.e.

$$BA \neq AB \tag{2.11}$$