Linear Algebra

- Much of scientific computation involves modeling n-tuples of numbers
- Assumed to live in a linear vector space
  - Even non-linear problems are solved by linearization
- Some interpretations of matrices and vectors
- Matrix vector multiplication and complexity
- Identity, Inverse, Singular Matrices
- Permutation, Lower and Upper Triangular Matrices

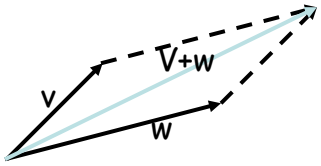
Vectors

- Ordered set of numbers: (1,2,3,4)
- Example: (*x,y,z*) coordinates of a pt in space.
- The 16384 pixels in a 128×128 image of a face
- Vectors usually indicated with bold lower case letters. Scalars with lower case
- Usual operations assumed with vectors:
  - Addition operation $\mathbf{u} + \mathbf{v}$, with:
    - Identity $\mathbf{0}$ $\quad \mathbf{v} + \mathbf{0} = \mathbf{v}$
    - Inverse - $\quad \mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
  - Scalar multiplication:
    - Distributive rule: $\alpha(\mathbf{u} + \mathbf{v}) = \alpha(\mathbf{u}) + \alpha(\mathbf{v})$

      $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

Vector Addition

$$\mathbf{v} + \mathbf{w} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$



Vector Spaces

- A *linear combination* of vectors results in a new vector:

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n$$

- If the only set of scalars such that

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n = \mathbf{0}$$

  is $\alpha_1 = \alpha_2 = \ldots = \alpha_3 = \mathbf{0}$

  then we say the vectors are *linearly independent*
- The *dimension* of a space is the greatest number of linearly independent vectors possible in a vector set
- For a vector space of dimension *n*, any set of *n* linearly independent vectors form a *basis*

Vector Spaces: Basis Vectors

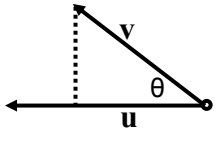
- Given a basis for a vector space:
  - Each vector in the space is a *unique* linear combination of the basis vectors
  - The *coordinates* of a vector are the scalars from this linear combination
  - Best-known example: Cartesian coordinates
    - Example
  - Note that the same vector **v** will have different coordinates in different bases

Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

$$\mathbf{v}_1 \bullet \mathbf{v}_2 = \mathbf{x}_1\mathbf{x}_2 + \mathbf{y}_1\mathbf{y}_2 + \mathbf{z}_1\mathbf{z}_2 \qquad \text{(in 3D)}$$

- Useful for many purposes
  - Computing the length of a vector: *length*(**v**) = sqrt(**v** • **v**)
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors:
    $\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}|\,|\mathbf{v}| \cos(\theta)$
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another



Vector norms

$v = (x_1, x_2, \ldots$ n

Two norm (Euclidean norm)

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

If $\|v\|_2 = 1$, v is a unit vector

Infinity norm

$\|v\|_\infty = \max(|x|_1, |x|_2, \ldots$ n

One norm ("Manhattan distance")

$$\|v\|_2 = \sum_{i=1}^{n} \left| x_i \right|$$

For a 2 dimensional vector, write down the set of vectors with two, one and infinity norm equal to unity

Orthonormal Basis

- Let S = { $\mathbf{v}_1$, $\mathbf{v}_2$, K, $\mathbf{v}_n$ } be a basis for an inner product space V. Then S is an orthonormal basis for V if

  a) ($\mathbf{v}_i$, $\mathbf{v}_j$) = 0 for $i \neq j$

  b) ($\mathbf{v}_i$, $\mathbf{v}_i$) = 1 for all i

Theorem

Let S = { $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ } be an orthonormal basis for an inner product space V and let **v** be any vector in V.

Then $\mathbf{v} = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \ldots + c_n \mathbf{v}_n$

where $c_i = (\mathbf{v}, \mathbf{v}_i)$ for all i

Proof -

$$\begin{aligned}(\mathbf{v},\mathbf{v}_i) &= (c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots \quad \cdots \\ &= (c_1\mathbf{v}_1,\mathbf{v}_i) + (c_2\mathbf{v}_2,\mathbf{v}_i) + \cdots \quad \cdots \\ &= c_1(\mathbf{v}_1,\mathbf{v}_i) + c_2(\mathbf{v}_2,\mathbf{v}_i) + \cdots \quad \cdots \\ &= c_1 \bullet 0 + c_2 \bullet 0 + \cdots \quad \cdots \\ &= c_i\end{aligned}$$

Gram-Schmidt Process

- Orthogonal bases can make computations more numerically stable, and sparse.
- If S = { $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ } is a basis (not orthonormal) for an inner product space V, is there a way to convert it to an orthonormal basis?

Gram-Schmidt Process

- Replace the basis with an orthonormal basis $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$

$$S = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \right\}$$

$$\mathbf{v}_1 = \mathbf{u}_1 \Rightarrow \mathbf{w}_1 = \mathbf{v}_1 / \|\mathbf{v}_1\| = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$

$$\mathbf{v}_2 = \mathbf{u}_2 - (\mathbf{w}_1, \mathbf{u}_2)\mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{\sqrt{2}} \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ -1/2 \end{bmatrix}$$

$$\mathbf{w}_2 = \mathbf{v}_2 / \|\mathbf{v}_2\| = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{bmatrix}$$

Gram-Schmidt Process

Example (continued)

$$\mathbf{v}_3 = \mathbf{u}_3 - (\mathbf{u}_3, \mathbf{w}_1)\mathbf{w}_1 - (\mathbf{u}_3, \mathbf{w}_2)\mathbf{w}_2$$

$$= \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} - \sqrt{2}\begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix} - \sqrt{2}\begin{bmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{w}_3 = \mathbf{v}_3 / \|\mathbf{v}_3\| = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Orthonormal set is $\left\{ \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{bmatrix}, \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$

Comments

- Key idea in Gram-Schmidt is to subtract from every new vector, $\mathbf{u}_k$, its components in the directions already determined, $\{ \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k-1} \}$
- Remove any intermediate zero vectors from process
- When doing Gram-Schmidt by hand, it simplifies the calculation to multiply the newly computed $\mathbf{v}_k$ by an appropriate scalar to clear fractions in its components. The resulting vectors are normalized at the end of the computation
- On a computer it is better to just normalize at the end … otherwise errors due to divisions by small numbers are

Modified Gram Schmidt

- Normalize at the end

Input: Suppose we have linearly independent vectors $x_1, x_2, \ldots, x_k$.

Output: An orthonormal set of vectors $q_1, \ldots, q_k$ with the property that $\text{span}\{q_1, \ldots, q_j\} = \text{span}\{x_1, \ldots, x_j\}$ for all $j \leq k$.

Algorithm:

STEP 1 Compute orthogonal vectors $\{z_1, \ldots, z_k\}$:

$$z_1 = x_1$$

$$z_2 = x_2 - \text{proj}_{z_1}(x_2) = x_2 - \frac{\langle x_2, z_1 \rangle}{\langle z_1, z_1 \rangle} z_1$$

$$\vdots$$

$$z_k = x_k - \text{proj}_{z_1}(x_k) - \cdots - \text{proj}_{z_{k-1}}(x_k) = x_k - \frac{\langle x_k, z_1 \rangle}{\langle z_1, z_1 \rangle} z_1 - \cdots - \frac{\langle x_k, z_{k\ 1} \rangle}{\langle z_{k\ 1}, z_{k\ 1} \rangle} z_{k\ 1}$$

STEP 2 Normalize vectors:

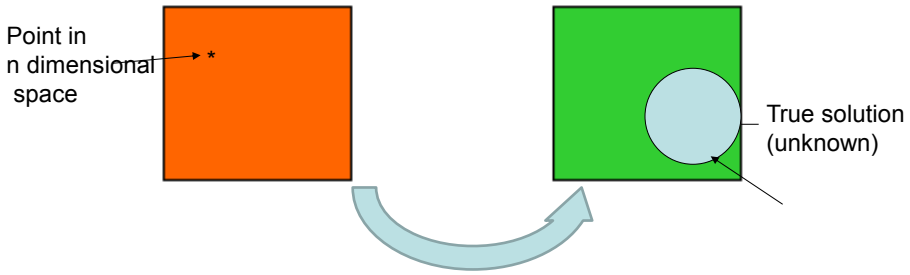
$$q_1 = \frac{z_1}{\|z_1\|}, \quad q_2 = \frac{z_2}{\|z_2\|}, \quad \ldots, \quad q_k = \frac{z_k}{\|z_k\|}$$

Transformations

- So far we viewed a matrix as a collection of vectors
- Saw how to orthognalize this collection
- Instead we now look at Matrices as linear functions that map vector spaces onto other vector spaces

Transform view

- Matrix takes a point in a *n* dimensional space to a point in a *m* dimensional space
- *n* dimensional space is the domain *on* which matrix acts



  - Output values assumed in the *m* dimensional space may not span entire space but just live in a small region of that space
  - That region is the "range" of the matrix

Linear Transformations: Matrices

- A *linear transformation*:
  - Maps one vector to another
  - Preserves linear combinations
- Thus behavior of linear transformation is completely determined by what it does to a basis
- Linear transform from a finite vector space to another can be represented by a *matrix*
- Choose the Kronecker basis
- Allows us to express the "abstract" matrix object as a concrete object composed of $m \times n$ numbers

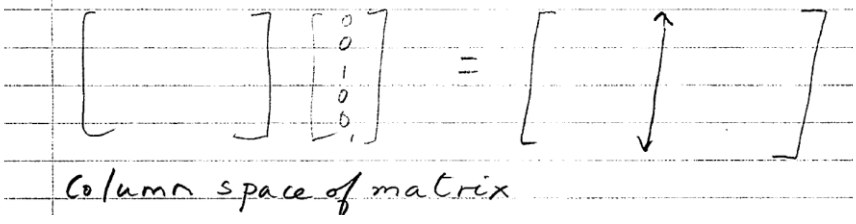
Matrix Transformations

- A *sequence* or *composition* of linear transformations corresponds to the product of the corresponding matrices
  - Note: the matrices to the *right* affect vector first
  - Note: order of matrices matters!
- The *identity matrix* **I** has no effect in multiplication
- Some (not all) matrices have an inverse:

$$\mathbf{M}^{-1}(\mathbf{M}(\mathbf{v})) = \mathbf{v}$$

Matrices

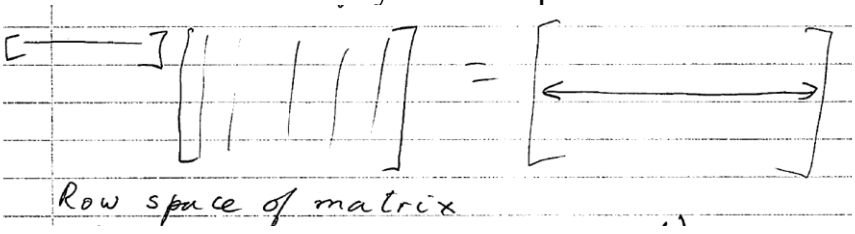
- Matrix operations usually defined via action on column vectors
- Suppose we check actions on a set of simple basis vectors



- This multiplication outputs a column of the matrix
- All outputs of the matrix-vector product (transform) will be a set of scaled sums of columns

Left and right multiplication

- Usual multiplication is on the right
- Can also define multiplication on the left … takes *m* vectors to *n* vectors
- Can be characterized by the row space of matrix



Algorithms to be studied - 1

- Linear system solution via LU decomposition
  - For a transform from *n* space to *n* space, given the image of the transform and the matrix computer the original image
  - Stability, etc.
  - Uses to compute matrix inverse, determinant etc.
  - **Never use the inverse to solve a linear system**
- Least Squares
  - Given *m* outputs for a *n* dimensional linear model, compute the best (least-squares) estimate of the parameters of the model
  - Normal equations
  - QR decomposition
    - Givens Rotations and Householder transformations
  - **Never use the normal equations!**

Algorithms to be studied - 2

- Eigenvalues and Eigenvectors
  - For a given matrix, compute the special directions ("natural basis vectors") which are not transformed
  - QR algorithm
- Singular Value Decomposition
  - Complete view of the row and column spaces of a matrix
  - Matrix conditioning
  - Regularization
  - Pseudoinverse
  - Principal Components Analysis

Ways to define matrices

- Perhaps all entries are random …
- More often, they are somehow functions of *i* and *j*

Example: A **Hilbert** matrix A of size 5×5, with element (i, j)

$$a_{ij} = \frac{1}{i + j - 1}.$$

- Matlab code to generate matrix
- If matrix is of size $N \times N$ how many operations are needed to enter values?
- Sometimes each column or row is given as a formula:
  - Example Vandermonde matrix in polynomial interpolation

```
n=5;
A = zeros(n,n);
for i=1:n,
   for j=1:n,
       A(i,j) = 1/(i+j-1);
   end
end
```

Some special matrices

Example: A **Vandermonde** matrix A is defined by a vector of elements $x_1, \ldots, x_n$. Its first column is all ones. Each later column is the preceding one times this vector.

- Matlab code
- How many operations and memory does this take?
- Vectorized operations

```
n = length(x);
V(:,1) = ones(n,1);
for j=2:n,
   V(:,j) = x.*V(:,j-1);
end
```

- Matrix may be sparse, i.e. most elements are zero.
- How many operations/memory?
- Answer still N^2 unless we avoid referring to the zero elements altogether

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$$

```
D = diag([1 2 4 6]);
```

Some special matrices

Example: A **tridiagonal** matrix

$$T = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 5 & 2 & 7 & 0 \\ 0 & 9 & 4 & 8 \\ 0 & 0 & 6 & 6 \end{bmatrix}$$

can be defined by

```
T = diag([1 2 4 6]) + diag([5 9 6],-1) + diag([3 7 8], 1);
```

- Matrices may be built up from "blocks" of smaller matrices

Matrix norms

- Can be defined using corresponding vector norms
  - Two norm
  - One norm
  - Infinity norm
- Two norm is hard to define … need to find maximum singular value
  - related to idea that matrix acting on unit sphere converts it in to an ellipsoid
- Frobenius norm is defined just using matrix elements

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$$

$$\|A\|_1 = \max_{\|x\|_1=1} \|Ax\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|$$

$$\|A\|_\infty = \max_{\|x\|_1=1} \|Ax\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|$$

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^2 \right)^{1/2}$$