Chapter 4

Linear Algebra

When you work with numerical data, it is naturally organized as a (possibly multidimensional) matrix. Therefore, linear algebra provides the natural context for working with data, and the theory of linear algebra can also give us methods for performing an analysis of that data.

First up is a little review.

It can be argued that all of linear algebra can be understood using the *four fundamental subspaces* associated with a matrix. Because they form the foundation on which we later work, we want an explicit method for analyzing these subspaces- That method will be the *Singular Value Decomposition* (SVD).

Finally, the SVD will give us the tools for performing linear regression analysis, projection analysis, and Principle Components Analysis (PCA).

Whenever we are analyzing new data sets, we can look to the SVD and PCA as the first tools we use to see if we can extract "features" from the data (more on that later).

4.1 Representation, Basis and Dimension

Let us quickly review some notation and basic ideas from linear algebra. First and foremost, we have to define the vector space that we're working with. For example, our "vectors" could be column vectors with n elements in them. Our "vectors" could actually be matrices that are all $m \times n$. The vector space defines the background to which all of our other analysis takes place, so it's an important consideration.

We will typically define our vector space V by constructing a **basis** for it, which is a **linearly independent** and **spanning** set of vectors, and the **dimension** of V is the number of its basis vectors.

Vector Space $\mathbb{R}^n$

Suppose the vector space is $\mathbb{R}^n$, and let H be a subspace of $\mathbb{R}^n$ with basis vectors in set $\mathcal{B}$. Since we're in $\mathbb{R}^n$, we can organize these vectors as columns in a matrix B (so B would be $n \times k$):

$$\mathcal{B} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}, \qquad B = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k].$$

Then every vector in H can be written as a linear combination of the basis vectors. That is, if $\boldsymbol{x} \in H$,

$$\boldsymbol{x} = c_1\boldsymbol{v}_1 + \ldots + c_k\boldsymbol{v}_k$$

An important observation is that the linear combination you see above can be written in matrix-vector form:

$$\boldsymbol{x} = c_1\boldsymbol{v}_1 + \ldots + c_k\boldsymbol{v}_k = B[\boldsymbol{x}]_{\mathcal{B}} \tag{4.1}$$

where $(c_1, c_2, \ldots, c_k)^T = [\boldsymbol{x}]_{\mathcal{B}}$ is called the **coordinates of $\boldsymbol{x}$** with respect to the basis $\mathcal{B}$.

Now, consider that every vector in H can be identified with a point in $\mathbb{R}^k$, and that defines a function that we'll call the **coordinate mapping**:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n \longleftrightarrow [\boldsymbol{x}]_{\mathcal{B}} = \begin{bmatrix} c_1 \\ \vdots \\ c_k \end{bmatrix} \in \mathbb{R}^k$$

If k is small (with respect to n) we think of $\boldsymbol{c}$ is the **low dimensional representation** of the vector $\boldsymbol{x}$, and that H is *isomorphic* to $\mathbb{R}^k$ (Isormorphic meaning one to one and onto linear map is the isormorphism)

Example 4.1.1. If $H = \text{span}\,(\boldsymbol{v}_1, \boldsymbol{v}_2)$ where $\boldsymbol{v}_{1,2} \in \mathbb{R}^5$, then H is *isomorphic* to the plane $\mathbb{R}^2$ - but is not *equal* to the plane. The isomorphism is the coordinate mapping (call it F). For example, if $\mathbf{x} \in H$, then

$$\mathbf{x} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$$

and $F(\mathbf{x}) = (c_1, c_2) \in \mathbb{R}^2$.

Computing the Coordinates

Generally, finding the coordinates of $\boldsymbol{x}$ with respect to an arbitrary basis (as columns of a matrix V) means that we have to solve the matrix equation for $[\boldsymbol{x}]_{\mathcal{B}}$:

$$\mathbf{x} = B[\boldsymbol{x}]_{\mathcal{B}}.$$

However, if the columns of B are actually orthogonal, then it is very simple to compute the coordinates. We'll start with our vector equation: $\boldsymbol{x} = c_1\boldsymbol{v}_1 + \cdots + c_k\boldsymbol{v}_k$. Now take the inner product of both sides with $\boldsymbol{v}_j$:

$$\boldsymbol{x} \cdot \boldsymbol{v}_j = c_1\boldsymbol{v}_1 \cdot \boldsymbol{v}_j + \cdots + c_j\boldsymbol{v}_j \cdot \boldsymbol{v}_j + \cdots + c_k\boldsymbol{v}_k \cdot \boldsymbol{v}_j$$

All the dot products are 0 (due to orthogonality) except for the dot product with $\boldsymbol{x}$ and $\boldsymbol{v}_j$ leading to the formula:

$$\boldsymbol{x} \cdot \boldsymbol{v}_j = 0 + 0 + \ldots + c_j\boldsymbol{v}_j \cdot \boldsymbol{v}_j + \cdots + 0 \quad \Rightarrow \quad c_j = \frac{\boldsymbol{x} \cdot \boldsymbol{v}_j}{\boldsymbol{v}_j \cdot \boldsymbol{v}_j}$$

And we remember that this is the scalar projection of $\boldsymbol{x}$ onto $\boldsymbol{v}_j$, which we've seen in Calc III and linear algebra:

$$\text{Proj}_v(\mathbf{u}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}}\mathbf{v}$$

Therefore, we can think of the linear combination as the following, which simplifies if we use **orthonormal basis vectors**:

$$\begin{aligned} \mathbf{x} &= \text{Proj}_{v_1}(\mathbf{x}) + \text{Proj}_{v_2}(\mathbf{x}) + \cdots + \text{Proj}_{v_k}(\mathbf{x}) \qquad (4.2) \\ &= (\mathbf{x} \cdot \mathbf{v}_1)\mathbf{v}_1 + (\mathbf{x} \cdot \mathbf{v}_2)\mathbf{v}_2 + \cdots + (\mathbf{x} \cdot \mathbf{v}_k)\mathbf{v}_k \end{aligned}$$

IMPORTANT NOTE: In the event that $\mathbf{x}$ is NOT in H, then Equation 4.2 gives the (orthogonal) **projection of $\mathbf{x}$** into H.

The important conclusion we want to make is the following: Out of all possible bases we might choose for a subspace of a vector space, the "best" one is one that is orthonormal, since the coordinates of a vector are easy to compute using the inner product. In fact, if the space is a subspace of $\mathbb{R}^n$ and the (orthonormal) basis is stored as the columns of a matrix B ($n \times k$ in our previous example), then the coordinates of any $\mathbf{x}$ can be simply computed as:

$$[\mathbf{x}]_{\mathcal{B}} = B^T\mathbf{x}$$

Of course, you might be noting that in that case,

$$\mathbf{x} = B[\mathbf{x}]_{\mathcal{B}} = BB^T\mathbf{x}$$

This matrix, BB^T, is a very interesting one- One which we examine more closely in the next section.

Projections

Consider the following example. If a matrix $U = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$ has orthonormal columns (so if U is $n \times k$, then that requires $k \le n$), then $U^T U$ is $k \times k$, and can be computed as:

$$U^T U = \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_k \end{bmatrix} [\mathbf{u}_1, \ldots, \mathbf{u}_k] = \begin{bmatrix} \mathbf{u}_1^T\mathbf{u}_1 & \mathbf{u}_1^T\mathbf{u}_2 & \cdots & \mathbf{u}_1^T\mathbf{u}_k \\ \mathbf{u}_2^T\mathbf{u}_1 & \mathbf{u}_2^T\mathbf{u}_2 & \cdots & \mathbf{u}_2^T\mathbf{u}_k \\ \vdots & & & \vdots \\ \mathbf{u}_k^T\mathbf{u}_1 & \mathbf{u}_k^T\mathbf{u}_2 & \cdots & \mathbf{u}_k^T\mathbf{u}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = I_k$$

But UU^T (which is $n \times n$) is NOT the identity if $k \neq n$ (If $k = n$, then the previous computation proves that the inverse is the transpose).

Here is a computation one might make for UU^T (these are OUTER products):

$$UU^T = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_k \end{bmatrix} = \mathbf{u}_1\mathbf{u}_1^T + \mathbf{u}_2\mathbf{u}_2^T + \cdots + \mathbf{u}_k\mathbf{u}_k^T$$

Example 4.1.2. Consider the following computations:

$$U = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad U^T U = 1 \qquad UU^T = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

If UU^T is not the identity, what is it? Consider the following computation:

$$UU^T\mathbf{x} = \mathbf{u}_1\mathbf{u}_1^T\mathbf{x} + \mathbf{u}_2\mathbf{u}_2^T\mathbf{x} + \cdots + \mathbf{u}_k\mathbf{u}_k^T\mathbf{x}$$

$$= \mathbf{u}_1(\mathbf{u}_1^T\mathbf{x}) + \mathbf{u}_2(\mathbf{u}_2^T\mathbf{x}) + \cdots + \mathbf{u}_k(\mathbf{u}_k^T\mathbf{x})$$

which we recognize as the projection of $\mathbf{x}$ into the space spanned by the orthonormal vectors of U. In the previous section, we called this matrix B, but it is common notation that a matrix with orthonormal vectors is typically denoted by U.

Just to repeat then: If we have a set of orthonormal vectors in a matrix U forming a basis for some subspace H, then if $\mathbf{x} \in H$,

$$\mathbf{x} = U[\mathbf{x}]_U = U(U^T\mathbf{x}) = (UU^T)\mathbf{x}$$

However, if $\mathbf{x}$ is not completely contained in H, then

$$\text{Proj}_H(\mathbf{x}) = UU^T\mathbf{x}$$

And of course, if $n > k$ and U is $n \times k$, then $U^T U = I$.

Notes about Programming

Programming in Matlab

- Find a random matrix with orthonormal columns

  SOLUTION: Use the "QR" decomposition of a matrix A. That is, decompose matrix A as a product of matrix Q (with orthonormal columns) times matrix R.

```
X=randn(9,6);
[Q,R]=qr(X,0);
Q'*Q          % Should be 6 x 6 identity matrix.
```

- Take a matrix X and "normalize" it by making each column have norm 1.

  SOLUTION: Find a row representing the norm of each column, then divide each row of the matrix by that row.

```
X=[1,0;0,1;1,0]
RowNorms=sqrt(sum(X.*X));
C=X./RowNorms
```

- Let $\boldsymbol{x}$ be a random vector in $\mathbb{R}^9$. We're going to project this into the subspace spanned by the first three columns of the matrix Q that we constructed previously.

  - First, what are the coordinates? In linear algebra, the coordinates (a vector in $\mathbb{R}^3$) are: $Q^T\boldsymbol{x}$, where this Q is 9×3.
  - Next, project this vector into the subspace: In linear algebra notation, $QQ^T\boldsymbol{x}$

  Here it is in Matlab, using the Q from the QR decomposition.

```
x=rand(9,1);
Coords=Q(:,1:3)'*x;
Projx=Q(:,1:3)*Coords;
Check=Q*(Q'*Projx);  % Don't compute QQ^T first!
norm(Check-Projx);  %Should be close to zero
```

Programming in Python

- Find a random matrix with orthonormal columns.

  SOLUTION: Use the "QR" decomposition of a matrix A. That is, decompose matrix A as a product of matrix Q (with orthonormal columns) times matrix R.

```
A=np.random.randn(9,6)
q,r=np.linalg.qr(A)
q.shape              #Answer is (9,6)
D=np.matmul(q.T,q)   #Should be 6 x 6 identity
```

- Take a matrix X and "normalize" it by making each column have norm 1.

  SOLUTION: Find a row representing the norm of each column, then divide each row of the matrix by that row.

```
import numpy as np
import numpy.linalg

X=np.array([[1, 0],[0,1],[1,0]])
RowNorms=np.linalg.norm(X,axis=0)
C=X/RowNorms[np.newaxis,:]
Output:
array([[0.70710678, 0.        ],
       [0.        , 1.        ],
       [0.70710678, 0.        ]])
```

- Let $\boldsymbol{x}$ be a random vector in $\mathbb{R}^9$. We're going to project this into the subspace spanned by the first three columns of the matrix Q that we constructed previously.

- First, what are the coordinates? In linear algebra, the coordinates (a vector in $\mathbb{R}^3$) are: $Q^T\boldsymbol{x}$, where this Q is 9×3.
- Next, project this vector into the subspace: In linear algebra notation, $QQ^T\boldsymbol{x}$

```
import numpy as np
import numpy.linalg

x=np.random.rand(9,1)  #Random vector in R^9
A=np.random.rand(9,3)  #We'll go with a 3-d subspace in R^9
Q,R=np.linalg.qr(A)

Q.shape   #This checks the dimensions- This is 9 x 3

Coords=np.matmul(Q.T,x)  #This is a 3 x 1 vector, Q^Tx
xProjected=np.matmul(Q,Coords)  #xProjected is 9 x 1

#Optional: Check by projecting the projection (shouldn't change)
Check=np.matmul(Q,np.matmul(Q.T,xProjected))
np.linalg.norm(xProjected-Check)  #Returns something like 2 x 10^(-16)
```

Programming in R

- Find a random matrix with orthonormal columns

  SOLUTION: Use the "QR" decomposition of a matrix A. That is, decompose matrix A as a product of matrix Q (with orthonormal columns) times matrix R.

```
A<-matrix(rnorm(54),nrow=9)   # rnorm are random, normal dist.
X=qr(A)                       # X is a QR-object
Q<-qr.Q(X)                    # Extract the matrix Q
dim(Q)                        # Check dimensions on Q (returns (9,6))
H<- t(Q) %*% Q                # Compute Q^TQ to get 6 x 6 identity
```

- Take a matrix X and "normalize" it by making each column have norm 1.

  SOLUTION: Find a row representing the norm of each column, then divide each row of the matrix by that row.

```
X<-matrix(c(1,1,0,0,1,0),3,2)
N<-sqrt(colSums(X^2)
C<-sweep(X,2,N,FUN='/')
```

- Let $\boldsymbol{x}$ be a random vector in $\mathbb{R}^9$. We're going to project this into the subspace spanned by the first three columns of the matrix Q that we constructed previously.
  - First, what are the coordinates? In linear algebra, the coordinates (a vector in $\mathbb{R}^3$) are: $Q^T\boldsymbol{x}$, where this Q is 9×3.
  - Next, project this vector into the subspace: In linear algebra notation, $QQ^T\boldsymbol{x}$

```
> x<-matrix(runif(9),ncol=1)
> A<-matrix(runif(27),ncol=3)
> X<-qr(A)
> Q<-qr.Q(X)            #This is a bit awkward
> Coords<-t(Q)%*%x
> xProjected<-Q %*% Coords
> Check=Q %*% (t(Q) %*% xProjected)
> y=xProjected-Check
> sqrt(sum(y^2))
[1] 5.360492e-16
```

Exercises

1. Let the subspace H be formed by the span of the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ given below. Given the point $\boldsymbol{x}_1, \boldsymbol{x}_2$ below, find which one belongs to H, and if it does, give its coordinates. (NOTE: The basis vectors are NOT orthonormal)

$$\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix} \quad \boldsymbol{v}_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \quad \boldsymbol{x}_1 = \begin{bmatrix} 7 \\ 4 \\ 0 \end{bmatrix} \quad \boldsymbol{x}_2 = \begin{bmatrix} 4 \\ 3 \\ -1 \end{bmatrix}$$

2. Show that the plane H defined by:

$$H = \left\{ \alpha_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \text{ such that } \alpha_1, \alpha_2 \in \mathbb{R} \right\}$$

is isormorphic to $\mathbb{R}^2$.

3. Let the subspace G be the plane defined below, and consider the vector $\boldsymbol{x}$, where:

$$G = \left\{ \alpha_1 \begin{bmatrix} 1 \\ 3 \\ -2 \end{bmatrix} + \alpha_2 \begin{bmatrix} 3 \\ -1 \\ 0 \end{bmatrix} \text{ such that } \alpha_1, \alpha_2 \in \mathbb{R} \right\} \qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

   (a) Find the matrix (UU^T in our notes) that takes an arbitrary vector and projects it (orthogonally) to the plane G.
   (b) Find the orthogonal projection of the given $\boldsymbol{x}$ onto the plane G.
   (c) Find the distance from the plane G to the vector $\boldsymbol{x}$.

4. If the low dimensional representation of a vector $\boldsymbol{x}$ is $[9, -1]^T$ and the basis vectors are $[1, 0, 1]^T$ and $[3, 1, 1]^T$, then what was the original vector $\boldsymbol{x}$? (HINT: it is easy to compute it directly)

5. If the vector $\boldsymbol{x} = [10, 4, 2]^T$ and the basis vectors are $[1, 0, 1]^T$ and $[3, 1, 1]^T$, then what is the low dimensional representation for $\boldsymbol{x}$?

6. Let $\boldsymbol{a} = [-1, 3]^T$. Find a square matrix P so that $P\boldsymbol{x}$ is the orthogonal projection of $\boldsymbol{x}$ onto the span of $\boldsymbol{a}$.

7. Refer to one of the programming languages (Matlab/Python/R), and reproduce finding an arbitrary 10×4 matrix with orthonormal columns. Use a random $\boldsymbol{x} \in \mathbb{R}^{10}$, and first find the coordinates of $\boldsymbol{x}$ with respect to the four columns in Q, then compute the orthogonal projection of $\boldsymbol{x}$ into the subspace spanned by the first four columns of Q.

8. To prove that we have an *orthogonal* projection, the vector $\text{Proj}_u(\boldsymbol{x}) - \boldsymbol{x}$ should be orthogonal to $\boldsymbol{u}$. Use this definition to show that our earlier formula was correct- that is,

$$\text{Proj}_u(\boldsymbol{x}) = \frac{\boldsymbol{x} \cdot \boldsymbol{u}}{\boldsymbol{u} \cdot \boldsymbol{u}} \boldsymbol{u}$$

is the orthogonal projection of $\mathbf{x}$ onto $\mathbf{u}$.

9. Continuing with the last exercise, show that $UU^T\mathbf{x}$ is the *orthogonal* projection of $\mathbf{x}$ into the space spanned by the columns of U by showing that $(UU^T\boldsymbol{x} - \boldsymbol{x})$ is orthogonal to $\mathbf{u}_i$ for any $i = 1, 2, \cdots, k$.

4.2 The Four Fundamental Subspaces

Given any $m \times n$ matrix A, we consider the mapping $A : \mathbb{R}^n \to \mathbb{R}^m$ by:

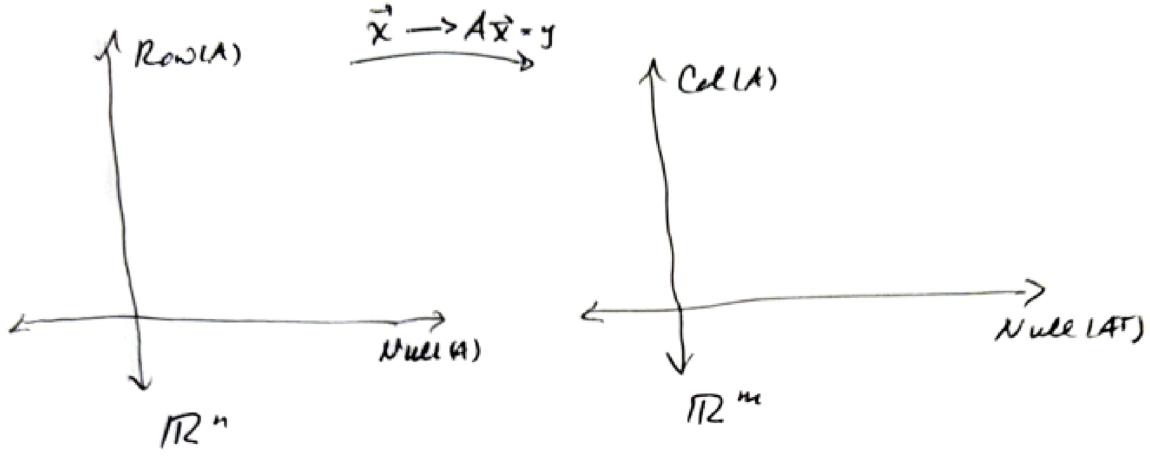
$$\boldsymbol{x} \to A\boldsymbol{x} = \boldsymbol{y}$$

The four subspaces allow us to completely understand the domain and range of the mapping. We will first define them, then look at some examples.

Definition 4.2.1. The Four Fundamental Subspaces

- The **row space** of A is a subspace of $\mathbb{R}^n$ formed by taking all possible linear combinations of the rows of A. Formally,
$$\text{Row}(A) = \left\{\boldsymbol{x} \in \mathbb{R}^n \,|\, x = A^T\boldsymbol{y} \;\; y \in \mathbb{R}^m\right\}$$
- The **null space** of A is a subspace of $\mathbb{R}^n$ formed by
$$\text{Null}(A) = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, A\boldsymbol{x} = \boldsymbol{0}\}$$
- The **column space** of A is a subspace of $\mathbb{R}^m$ formed by taking all possible linear combinations of the columns of A.
$$\text{Col}(A) = \{\boldsymbol{y} \in \mathbb{R}^m \,|\, y = A\boldsymbol{x} \quad \in \mathbb{R}^n\}$$
The column space is also the image of the mapping. Notice that $A\boldsymbol{x}$ is simply a linear combination of the columns of A:
$$A\boldsymbol{x} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + \cdots + x_n\boldsymbol{a}_n$$
- Finally, we define the **null space** of A^T can be defined in the obvious way (see the Exercises).

The fundamental subspaces subdivide the domain and range of the mapping in a particularly nice way. Below we give a "cartoon" of the relationship between the four subspaces. On the left is the domain of the matrix mapping, and it represents $\mathbb{R}^n$. On the right is the codomain, and it represents $\mathbb{R}^m$. The spaces can apparently be split into two subspaces each. In the domain, these are the Row and Null spaces. In the codomain, these are the Column space and the null space of A^T (we don't use this one much).



In the diagram, notice that the axes are representing subspaces. The picture suggests that the two spaces that split our domain and range are actually orthogonal subspaces- and that is true.

Theorem 4.2.1. *Let A be an $m \times n$ matrix. Then*

- *The nullspace of A is orthogonal to the row space of A*
- *The nullspace of A^T is orthogonal to the columnspace of A*

Proof: We'll prove the first statement, the second statement is almost identical to the first. To prove the first statement, we have to show that if we take any vector $\mathbf{x}$ from nullspace of A and any vector $\mathbf{y}$ from the row space of A, then $\mathbf{x} \cdot \mathbf{y} = 0$.

Alternatively, if we can show that $\mathbf{x}$ is orthogonal to each and every row of A, then we're done as well (since $\mathbf{y}$ is a linear combination of the rows of A).

In fact, now we see a strategy: Write out what it means for $\mathbf{x}$ to be in the nullspace using the rows of A. For ease of notation, let $\mathbf{a}_j$ denote the j^{th} **row** of A, which will have size $1 \times n$. Then:

$$A\mathbf{x} = \mathbf{0} \quad \Rightarrow \quad \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{a}_1\mathbf{x} \\ \mathbf{a}_2\mathbf{x} \\ \vdots \\ \mathbf{a}_m\mathbf{x} \end{bmatrix} = \mathbf{0}$$

Therefore, the dot product between any row of A and $\mathbf{x}$ is zero, so that $\mathbf{x}$ is orthogonal to every row of A. Therefore, $\mathbf{x}$ must be orthogonal to any linear combination of the rows of A, so that $\mathbf{x}$ is orthogonal to the row space of A. $\square$

Before going further, let us recall how to construct a basis for the column space, row space and nullspace of a matrix A. We'll do it with a particular matrix:

Example 4.2.1. Construct a basis for the column space, row space and nullspace of the matrix A below that is row equivalent to the matrix beside it, RREF(A):

$$A = \begin{bmatrix} 2 & 0 & -2 & 2 \\ -2 & 5 & 7 & 3 \\ 3 & -5 & -8 & -2 \end{bmatrix} \qquad \text{RREF}(A) = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The first two columns of the original matrix form a basis for the columnspace (which is a subspace of $\mathbb{R}^3$):

$$\text{Col}(A) = \text{span}\left\{ \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix} \right\}$$

A basis for the row space is found by using the row reduced rows corresponding to the pivots (and is a subspace of $\mathbb{R}^4$). You should also verify that you can find a basis for the null space of A, given below (also a subspace of $\mathbb{R}^4$). If you're having any difficulties here, be sure to look it up in a linear algebra text:

$$\text{Row}(A) = \text{span}\left\{ \begin{bmatrix} 1 \\ 0 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} \qquad \text{Null}(A) = \text{span}\left\{ \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

We will often refer to the dimensions of the four subspaces. We recall that there is a term for the dimension of the column space- That is, the rank.

Definition 4.2.2. The *rank* of a matrix A is the number of independent columns of A.

In our previous example, the rank of A is 2. Also from our example, we see that the rank is the dimension of the column space, and that this is the same as the dimension of the row space (all three numbers correspond to the number of pivots in the row reduced form of A). Finally, a handy theorem for counting is the following.

The Rank Theorem. Let the $m \times n$ matrix A have rank r. Then

$$r + \dim(\text{Null}(A)) = n$$

This theorem says that the number of pivot columns plus the other columns (which correspond to free variables) is equal to the total number of columns.

Example 4.2.2. The Dimensions of the Subspaces.

Given a matrix A that is $m \times n$ with rank k, then the dimensions of the four subspaces are shown below.

- $\dim(\text{Row}(A)) = k$
- $\dim(\text{Col}(A)) = k$
- $\dim(\text{Null}(A)) = n - k$
- $\dim\left(\text{Null}(A^T)\right) = m - k$

There are some interesting implications of these theorems to matrices of data- For example, suppose A is $m \times n$. With no other information, we do not know whether we should consider this matrix as n points in $\mathbb{R}^m$, or m points in $\mathbb{R}^n$. In one sense, it doesn't matter! The theorems we've discussed shows that the dimension of the columnspace is equal to the dimension of the rowspace. Later on, we'll find out that if we can find a basis for the columnspace, it is easy to find a basis for the rowspace. We'll need some more machinery first.

4.3 Exercises

In the exercises below, recall that the usual norm for a vector is the Euclidean norm, or the $2-norm$, which is defined as:

$$\|\mathbf{x}\| = \sqrt{x_1^1 + x_2^2 + \cdots x_n^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$

1. Show that $\text{Null}(A^T) \perp \text{Col}(A)$. Hint: You may use what we already proved.

2. If A is $m \times n$, how big can the rank of A possibly be?

3. Show that multiplication by an orthogonal matrix preserves lengths: $\|\mathbb{Q}\boldsymbol{x}\|_2 = \|\boldsymbol{x}\|_2$ (Hint: Use properties of inner products). Conclude that multiplication by $\mathbb{Q}$ represents a rigid rotation.

4. Prove the Pythagorean Theorem by induction: Given a set of n orthogonal vectors $\{\boldsymbol{x}_i\}$

$$\|\sum_{i=1}^{n} \boldsymbol{x}_i\|^2 = \sum_{i=1}^{n} \|\boldsymbol{x}_i\|^2$$

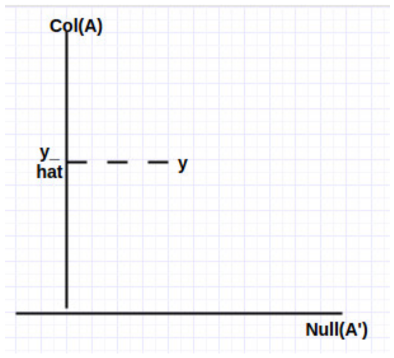
   The case where $n = 1$ is trivial, so you might look at $n = 2$ first. Try starting with

$$\|\mathbf{x} + \mathbf{y}\|^2 = (\mathbf{x} + \mathbf{y})^T(\mathbf{x} + \mathbf{y}) = \cdots$$

   and then simplify to get $\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2$. Now try the induction step on your own.

5. Let A be an $m \times n$ matrix where $m > n$, and let A have rank n. Let $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^m$, such that $\hat{\boldsymbol{y}}$ is the orthogonal projection of $\boldsymbol{y}$ onto the column space of A. We want a formula for the matrix $\mathbb{P} : \mathbb{R}^m \to \mathbb{R}^m$ so that $\mathbb{P}\boldsymbol{y} = \hat{\boldsymbol{y}}$.

   The following image shows the relevant subspaces:



(a) Why is the projector not $\mathbb{P} = AA^T$?

(b) Since $\hat{\boldsymbol{y}} - \boldsymbol{y}$ is orthogonal to the column space of A, show that

$$A^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) = \mathbf{0} \tag{4.3}$$

(c) Show that there exists $\boldsymbol{x} \in \mathbb{R}^n$ so that Equation (4.3) can be written as:

$$A^T(A\boldsymbol{x} - \boldsymbol{y}) = 0 \tag{4.4}$$

(d) Argue that $A^T A$ (which is $n \times n$) is invertible, so that Equation (4.4) implies that

$$\boldsymbol{x} = \left(A^T A\right)^{-1} A^T \boldsymbol{y}$$

(e) Finally, show that this implies that

$$\mathbb{P} = A\left(A^T A\right)^{-1} A^T$$

Note: If A has rank $k \neq n$, then we will need something different, since $A^T A$ will not be full rank. The missing piece is the singular value decomposition, to be discussed later.

6. The Orthogonal Decomposition Theorem: if $\boldsymbol{x} \in \mathbb{R}^n$ and W is a (non-zero) subspace of $\mathbb{R}^n$, then $\boldsymbol{x}$ can be written *uniquely* as

$$\boldsymbol{x} = \boldsymbol{w} + \boldsymbol{z}$$

where $\boldsymbol{w} \in W$ and $\boldsymbol{z} \in W^\perp$.

To prove this, let $\{\boldsymbol{u}_i\}_{i=1}^p$ be an orthonormal basis for W, define $\boldsymbol{w} = (\boldsymbol{x} \cdot \boldsymbol{u}_1)\boldsymbol{u}_1 + \ldots + (\boldsymbol{x} \cdot \boldsymbol{u}_p)\boldsymbol{u}_p$, and define $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{w}$. Then:

(a) Show that $\boldsymbol{z} \in W^\perp$ by showing that it is orthogonal to every $\boldsymbol{u}_i$.

(b) To show that the decomposition is unique, suppose it is not. That is, there are two decompositions:

$$\boldsymbol{x} = \boldsymbol{w}_1 + \boldsymbol{z}_1, \quad \boldsymbol{x} = \boldsymbol{w}_2 + \boldsymbol{z}_2$$

Show this implies that $\boldsymbol{w}_1 - \boldsymbol{w}_2 = \boldsymbol{z}_2 - \boldsymbol{z}_1$, and that this vector is in both W and $W^\perp$. What can we conclude from this?

7. Use the previous exercises to prove the **The Best Approximation Theorem** If W is a subspace of $\mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{R}^n$, then the point closest to $\boldsymbol{x}$ in W is the orthogonal projection of $\boldsymbol{x}$ into W.

4.4 The Decomposition Theorems

The matrix factorization that arises from an eigenvector/eigenvalue decomposition is useful in many applications, so we'll briefly review it here and build from it until we get to our main goal, the Singular Value Decomposition.

4.4.1 The Eigenvector/Eigenvalue Decomposition

First we have a basic definition:

Let A be an $n \times n$ matrix. If there exists a scalar λ and non-zero vector $\boldsymbol{v}$ so that

$$A\boldsymbol{v} = \lambda \boldsymbol{v}$$

then we say that λ is an eigenvalue and $\boldsymbol{v}$ is an associated eigenvector.

An equivalent formulation of the problem is to solve $A\mathbf{v} - \mathbf{v} = \mathbf{0}$, or, factoring $\mathbf{v}$ out,

$$(A - \lambda I)\mathbf{v} = \mathbf{0}$$

This equation always has the zero solution (letting $\mathbf{v} = \mathbf{0}$), however, we need to have non-trivial solutions, and the only way that will happen is if $A - \lambda I$ is non-invertible, or equivalently,

$$\det(A - \lambda I) = 0$$

which, when multiplied out, is a polynomial equation in λ that is called the **characteristic equation**.

Therefore, we find the eigenvalues first, then for each λ, there is an associated subspace- The null space of $A - \lambda I$, or the **eigenspace** associated with λ, denoted by E_λ.

The way to finish the problem is to give a "nice" basis for the eigenspace- If you're working by hand, try one with integer values. If you're on the computer, it is often convenient to make them unit vectors.

Some vocabulary associated with eigenvalues: Solving the characteristic equation will mean that we can have repeated solutions. The number of repetitions is the *algebraic multiplicity* of λ. On the other hand, for each λ, we find the eigenspace which will have a certain dimension- The dimension of the eigenspace is the *geometric multiplicity* of λ.

Examples:

1. Compute the eigenvalues and eigenvectors for the 2×2 identity matrix.

   SOLUTION: The eigenvalue is 1 (a double root), so the algebraic multiplicity of 1 is 2.

   On the other hand, if we take $A - \lambda I$, we simply get the zero matrix, which implies that every vector in $\mathbb{R}^2$ is an eigenvector. Therefore, we can take any basis of $\mathbb{R}^2$ is a basis for E_1, and the geometric multiplicity is 2.

2. Consider the matrix
$$\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

   Again, the eigenvalue 1 is a double eigenvalue (so the algebraic multiplicity is 2), but solving $(A - \lambda I)\boldsymbol{v} = 0$ gives us:
$$2v_2 = 0 \quad \Rightarrow \quad v_2 = 0$$

   That means v_1 is free, and the basis for E_1 is $[1, 0]^T$. Therefore, the algebraic multiplicity is 1.

Definition: A matrix for which the algebraic and geometric multiplicities are not equal is called *defective*.

There is a nice theorem relating eigenvalues:
Theorem: If X is square and invertible, then A and $X^{-1}AX$ have the same eigenvalues.
Sometimes this method of characterizing eigenvalues in terms of the determinant and trace of a matrix:

$$\det(A) = \Pi_{i=1}^{n}\lambda_i \quad \operatorname{trace}(A) = \sum_{i=1}^{\infty}\lambda_i$$

Symmetric Matrices and the Spectral Theorem

There are some difficulties working with eigenvalues and eigenvectors of a general matrix. For one thing, they are only defined for square matrices, and even when they are defined, we may get real or complex eigenvalues.

If a matrix is symmetric, beautiful things happen with the eigenvalues and eigenvectors, and it is summarized below in the Spectral Theorem.

The Spectral Theorem: If A is an $n \times n$ symmetric matrix, then:

1. A has n real eigenvalues (counting multiplicity).
2. For each distinct λ, the algebraic and geometric multiplicities are the same.
3. The eigenspaces are mutually orthogonal- both for distinct eigenvalues, and we'll take each E_λ to have an orthonormal basis.
4. A is orthogonally diagonalizeable, with $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. That is, if V is the matrix whose columns are the (othornormal) eigenvectors of A, then

$$A = VDV^T$$

Some remarks about the Spectral Theorem:

- If a matrix is real and symmetric, the Spectral Theorem says that its eigenvectors form an orthonormal basis for $\mathbb{R}^n$.
- The first part is somewhat difficult to prove in that we would have to bring in more machinery than we would like. If you would like to see a proof, it comes from the *Schur Decomposition*, which is given, for example, in "Matrix Computations" by Golub and Van Loan.

The following is a proof of the third part. Supply justification for each step: Let $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ be eigenvectors from distinct eigenvalues, λ_1, λ_2. We show that $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0$:

$$\lambda_1\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = (A\boldsymbol{v}_1)^T\boldsymbol{v}_2 = \boldsymbol{v}_1^T A^T \boldsymbol{v}_2 = \boldsymbol{v}_1^T A \boldsymbol{v}_2 = \lambda_2 \boldsymbol{v}_1 \cdot \boldsymbol{v}_2$$

Now, $(\lambda_1 - \lambda_2)\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0$.

The Spectral Decomposition: Since A is orthogonally diagonalizable, then

$$A = (\boldsymbol{q}_1\ \boldsymbol{q}_2\ \ldots\ \boldsymbol{q}_n) \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_1^T \\ \boldsymbol{q}_2^T \\ \vdots \\ \boldsymbol{q}_n^T \end{pmatrix}$$

so that:

$$A = (\lambda_1 \boldsymbol{q}_1 \ \lambda_2 \boldsymbol{q}_2 \ \ldots \ \lambda_n \boldsymbol{q}_n) \begin{pmatrix} \boldsymbol{q}_1^T \\ \boldsymbol{q}_2^T \\ \vdots \\ \boldsymbol{q}_n^T \end{pmatrix}$$

so finally:

$$A = \lambda_1 \boldsymbol{q}_1 \boldsymbol{q}_1^T + \lambda_2 \boldsymbol{q}_2 \boldsymbol{q}_2^T + \ldots + \lambda_n \boldsymbol{q}_n \boldsymbol{q}_n^T$$

That is, A is a sum of n rank one matrices, each of which is a projection matrix.

Exercises

1. Prove that if X is invertible and matrix A is square, then A and $X^{-1}AX$ have the same eigenvalues.

2. **Programming Exercise:** Verify the spectral decomposition for a symmetric matrix. Here are examples in Matlab, Python and R. First, we'll construct a random symmetric 6×6 matrix, then we'll compute the eigenvalues and eigenvectors.

   - Matlab:

     ```
     %Construct a random, symmetric, 6 x 6 matrix:
     N=6;
     Atemp=rand(N,N);
     A=(Atemp+Atemp')/2;  %A is symmetric

     %Compute the eigenvalues of A:
     [Q,L]=eig(A);   %NOTE:  A = Q L Q'
                     %L is a diagonal matrix

     %Now form the "spectral sum"
     S=zeros(6,6);
     for i=1:6
       S=S+L(i,i)*Q(:,i)*Q(:,i)';
     end

     max(max(S-A))
     ```

     Note that the maximum of $S - A$ should be a very small number! (By the spectral decomposition theorem).

   - Python:

     ```
     import numpy as np
     import numpy.linalg

     N=6;
     Atemp=np.random.rand(N,N)
     A=(Atemp+Atemp.T)/2       #Makes A symmetric

     D,V=numpy.linalg.eig(A)
     S=np.zeros((6,6))

     for i in range(0,6):
         S=S+D[i]*np.outer(V[:,i],V[:,i])
     ```

```
print(numpy.linalg.norm(A-S,'fro'))
```

- R:

```
A<-matrix(runif(36),ncol=6)
A<-(A+t(A))/2 #Makes A symmetric
P<-eigen(A)  #Creates a structure holding info
D<-P$values
V<-P$vectors
S=matrix(0,nrow=6,ncol=6)
for (i in 1:6){
  S=S+D[i]*( V[,i] %*% t(V[,i]) )
}
print(norm(A-S,"F"))
```

4.4.2 The Singular Value Decomposition

There is a special matrix factorization that is extremely useful, both in applications and in proving theorems. This is mainly due to two facts, which we shall investigate in this section: (1) We can use this factorization on *any* matrix, (2) The factorization defines explicitly the rank of the matrix, and gives **orthonormal bases** for all **four fundamental subspaces** of A.

In what follows, assume that A is an $m \times n$ matrix (so A maps $\mathbb{R}^n$ to $\mathbb{R}^m$ and is not necessarily square). We'll build the factorization by using the Spectral Theorem twice.

Step 1. Although A itself is not symmetric, A^TA is $n \times n$ and symmetric, so the Spectral Theorem applies, and we define an $n \times n$ orthonormal matrix V and diagonal matrix D_1 (with eigenvalues λ along the diagonal): using that theorem:

$$A^TA = VDV^T$$

We will assume that the eigenvalues are ordered from largest to smallest.

Step 2. Similarly, the $m \times m$ matrix AA^T is also symmetric, and so applying the Spectral Theorem again defines an $m \times m$ orthonormal matrix U and diagonal matrix D_2 such that

$$AA^T = UD_2U^T$$

We will assume that the eigenvalues are ordered from largest to smallest.

Step 3. Later, we will show that the non-zero diagonal elements of D_1 and D_2 are identical. Assuming the rank of A to be k, we'll define **the singular values** of A as:

$$\sigma_i = \begin{cases} \sqrt{\lambda_i} & \text{if } 1 \le i \le k \\ 0 & \text{for the remaining values} \end{cases}$$

where λ_i is the i^{th} diagonal element of D_1 or D_2. The $m \times n$ matrix Σ is defined to be the $m \times n$ diagonal matrix with singular values along the diagonal (ordered from largest to smallest).

Theorem **The Singular Value Decomposition (SVD)** Let A be any $m \times n$ matrix of rank k. Then we can factor the matrix A as the following product:

$$A = U\Sigma V^T$$

where U is an orthogonal $m \times m$ matrix, Σ is a diagonal $m \times n$ matrix, and V is an orthogonal $n \times n$ matrix. The columns of U are called the *left singular vectors* and the columns of V are called the *right singular vectors*. Further, there are exactly k non-zero singular values of A.

Before we get to the exercises, there is a different way of expressing the SVD that is similar to our spectral decomposition. Recall that the rank is k:

$$A = U\Sigma V^T = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \cdots + \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T$$

Please note that this is a sum of k matrices, not scalars- That is, we're computing outer products, $\sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ is an $m \times n$ matrix.

4.5 Exercises

In the exercises below, we'll look more closely at the SVD and the relationships between the vectors in U and V. As you're working through these, be sure to keep in mind the dimensions of the objects you're working with.

For all of the exercises, assume that A is $m \times n$ with rank k, and the SVD of A is $U\Sigma V^T$.

1. Show that if λ is an eigenvalue of $A^T A$, then λ is also an eigenvalue of AA^T.

   Hint: If λ is an eigenvalue of $A^T A$ with eigenvector $\mathbf{v}$, then $A^T A\mathbf{v} = \lambda\mathbf{v}$. Now what equation must be true if λ is an eigenvalue of AA^T?

2. There is a beautiful relationship between the column vectors of U and V. We will assume that A is $m \times n$ with rank k, and $A = U\Sigma V^T$ is the SVD of A. Then:

   $$A\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i \qquad \text{and} \qquad A^T \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i$$

   NOTE: The importance of this is that, if you know the left singular vectors, then you can compute the right singular vectors.

   HINT: You might start with $A\boldsymbol{v}_i$, then recall that $A = U\Sigma V^T$, which can be written as a sum of k matrices. The proof of the second equation is similar.

3. We said that $\sigma_i = \sqrt{\lambda_i}$. Here we prove that $\lambda_i \geq 0$, so that the singular values are always real:

   Show that $\lambda_i \geq 0$ for $i = 1..n$ by showing that $\|A\boldsymbol{v}_i\|^2 = \lambda_i$. HINT: As a starting point, you might rewrite

   $$\|A\boldsymbol{v}_i\|^2 = (A\boldsymbol{v})^T A\boldsymbol{v}$$

4. Prove that, if $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are distinct eigenvectors of $A^T A$, then their corresponding images, $A\boldsymbol{v}_i$ and $A\boldsymbol{v}_j$, are orthogonal.

5. Prove that, if $\boldsymbol{x} = \alpha_1 \boldsymbol{v}_1 + \ldots \alpha_n \boldsymbol{v}_n$, then $\|A\boldsymbol{x}\|^2 = \alpha_1^2 \lambda_1 + \ldots + \alpha_n^2 \lambda_n$. (You might check the hint in #3).

6. Let W be the subspace generated by the basis $\{\boldsymbol{v}_j\}_{j=k+1}^n$, where $\boldsymbol{v}_j$ are the eigenvectors associated with the *zero* eigenvalues of $A^T A$ (therefore, we are assuming that the first k eigenvalues are NOT zero). Show that $W = \text{Null}(A)$.

   Hint: To show this, take an arbitrary vector $\boldsymbol{x}$ from W. Rather than showing directly that $A\boldsymbol{x} = \mathbf{0}$, instead show that the magnitude of $A\boldsymbol{x}$ is zero. We also need to show that if we take any vector from the nullspace of A, then it is also in W.

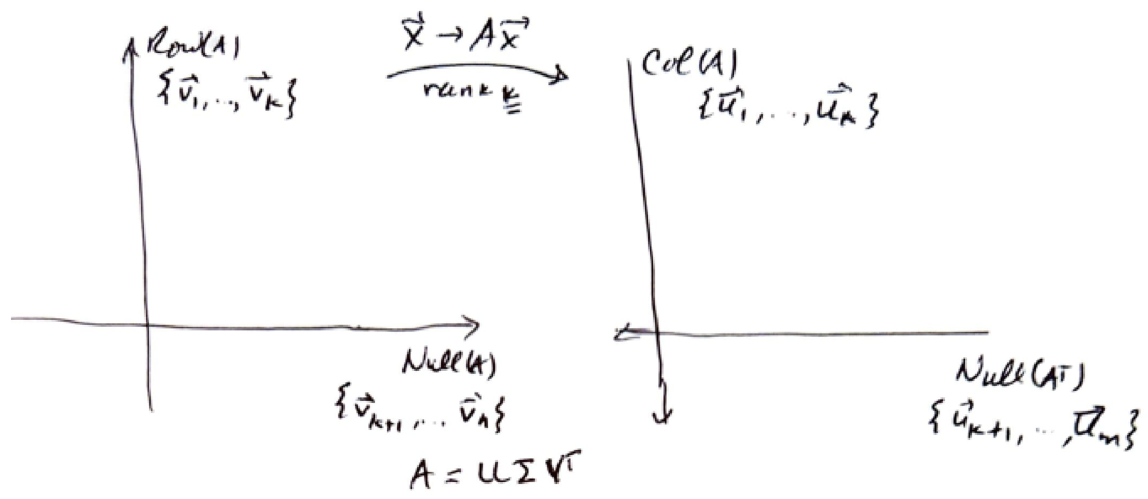
7. Prove that if the rank of $A^T A$ is r, then so is the rank of A.

   Hint: How does the previous exercise help?

8. Compute the SVD by hand of the following matrices:

   $$A_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

   (Hint: $A^T A$ and AA^T are symmetric matrices.)



Row(A)
$\{\vec{v}_1, \ldots, \vec{v}_k\}$
$\vec{x} \to A\vec{x}$
rank k
Col(A)
$\{\vec{u}_1, \ldots, \vec{u}_k\}$
Null(A)
$\{\vec{v}_{k+1}, \ldots, \vec{v}_n\}$
Null(A^T)
$\{\vec{u}_{k+1}, \ldots, \vec{u}_m\}$
$A = U\Sigma V^T$

Figure 4.1: The SVD of A (`[U,S,V]=svd(A)`) completely and explicitly describes the 4 fundamental subspaces associated with the matrix, as shown. We have a one to one correspondence between the rowspace and columnspace of A, the remaining $\boldsymbol{v}$'s map to zero, and the remaining $\boldsymbol{u}$'s map to zero (under A^T).

4.6 Notes about the SVD

The SVD computes a basis for all 4 fundamental subspaces for matrix A. To be more specific, let $A = U\Sigma V^T$ be the SVD which has rank k. Be sure that the singular values are ordered from highest to lowest. Then:

1. A basis for the columnspace of A, Col(A) is $\{\boldsymbol{u}_i\}_{i=1}^k$
2. A basis for nullspace of A, Null(A) is $\{\boldsymbol{v}_i\}_{i=k+1}^n$
3. A basis for the rowspace of A, Row(A) is $\{\boldsymbol{v}_i\}_{i=1}^k$
4. A basis for the nullspace of A^T, Null(A^T) is $\{\boldsymbol{u}_i\}_{i=k+1}^m$

Those are the basis vectors, but does σ_i also have some geometric interpretation? Yes- Recall that $A\mathbf{v}_i = \sigma_i \boldsymbol{u}_i$. Therefore, we can think of each σ_i as either a stretching or contraction factor along each $\boldsymbol{v}_i$ (which turns into $\boldsymbol{u}_i$.

The SVD is one of the premier tools of linear algebra primarily because it allows us to completely reveal everything we need to know about a matrix mapping: The rank, the basis of the nullspace, a basis for the column space, the basis for the nullspace of A^T, and of the row space. See Figure 4.1.

Lastly remembering that matrix multiplication is function composition, and multiplication by an orthogonal matrix represents a "rotation", the SVD provides a means of decomposing **any** linear mapping into two "rotations" and a scaling. This will become important later when we try to deduce a mapping matrix from data.

4.6.1 The Reduced (or "economy-size") SVD

If you have a matrix A that is 50000×3, you should NOT ask for the complete SVD decomposition unless you really, really mean it.

Why? Think about the sizes of your matrices- The matrix U would be 50000×50000, which is very large, and most of which is not needed. Indeed, the rank of your matrix is at most 3, so there are only at most three columns of your matrix that are really needed (unless again, you specifically require the basis for the nullspace of A^T). Instead, what you want to compute is called the Reduced SVD (or the economy-size SVD).

Definition: The Reduced SVD

Let A be $m \times n$ with rank k. Then we can write:

$$A = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$$

where $\tilde{U}$ is an $m \times k$ matrix with orthogonal columns, $\tilde{\Sigma}$ is an $k \times k$ *square* matrix, and $\tilde{V}$ is an $n \times k$ matrix.

Most of the time we'll only need the reduced SVD- The only difference is that we're stripping away the two nullspaces from the set of four fundamental subspaces. We'll see an example comparing the full with the reduced SVD below.

Remarks on some language: The reader will see several ways of referring to the "non-full" SVD. For example, there is reduced SVD, truncated SVD, and thin SVD to name three. I typically refer to the reduced SVD by using the rank of the matrix k to define matrix sizes (as above). If $m > n$, and we use n to define the sizes, then the decomposition is referred to as the "thin SVD" (Golub and Van Loan). However, computer software rarely will compute the rank of the matrix unless requested, so they actually compute the thin SVD, but will refer to it as the reduced SVD. The moral of the story is to be sure you know which version it is that you're using!

4.7 Programming with the SVD

Here, we'll perform some computations with a small, 5×3 matrix to illustrate some of the formulas we've been working with. We'll make the matrix have rank 2, then we'll compare the full and reduced SVD formulas. For this example, we'll take

$$A = \begin{bmatrix} -1 & 2 & 4 \\ 1 & 0 & -2 \\ 0 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 0 \end{bmatrix}$$

Now, the full SVD would look like $U\Sigma V^T$, where U, Σ, and V are below (resp):

$$\begin{bmatrix} 0.78 & 0.35 & -0.38 & -0.35 & -0.06 \\ -0.28 & -0.45 & -0.82 & -0.19 & -0.03 \\ 0.25 & -0.05 & -0.23 & 0.63 & 0.69 \\ 0.46 & -0.60 & 0.11 & 0.38 & -0.52 \\ 0.21 & -0.55 & 0.34 & -0.54 & 0.49 \end{bmatrix} \begin{bmatrix} 5.68 & 0 & 0 \\ 0 & 3.42 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.07 & -0.57 & -0.82 \\ 0.63 & -0.66 & 0.41 \\ 0.77 & 0.49 & -0.41 \end{bmatrix}$$

The reduced SVD would strip away the nullspaces, leaving us with $\tilde{U}, \tilde{\Sigma}, \tilde{V}$ below (resp):

$$\begin{bmatrix} 0.78 & 0.35 \\ -0.28 & -0.45 \\ 0.25 & -0.05 \\ 0.46 & -0.60 \\ 0.21 & -0.55 \end{bmatrix} \begin{bmatrix} 5.68 & 0 \\ 0 & 3.42 \end{bmatrix} \begin{bmatrix} -0.07 & -0.57 \\ 0.63 & -0.66 \\ 0.77 & 0.49 \end{bmatrix}$$

Secondly, let's look at the decomposition we discussed a few pages ago. That was:

$$A = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \cdots + \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T$$

In our case, $k = 2$, and we'll compute this using our three computer languages as a check.

Matlab and the SVD

The full command is `[U,S,V]=svd(A)`, where $A = USV^T$. For the reduced SVD, add a zero in the list of arguments, `[U,S,V]=svd(A,0)`. Then here's the script using our example matrix A:

```
A=[-1 2 4;1 0 -2;0 1 1;1 3 1;1 2 0];
[U,S,V]=svd(A);
Temp=S(1,1)*U(:,1)*V(:,1)'+S(2,2)*U(:,2)*V(:,2)';
norm(A-Temp,'fro')
% We could compute "Temp" as:
Temp2=U(:,1:2)*S(1:2,1:2)*V(:,1:2)';
norm(A-Temp2,'fro')
```

Side note: The `'fro'` part of the norm is short for "Frobenius". The Frobenius norm of a matrix is like treating the whole matrix as one big vector, then applying the Euclidean norm. That is, take every element, square it, then sum those and take the square root of the result.

Python and the SVD

Quick note: After working with Python and matrices for a bit, I see that in Python 3.5 and later, the `@` symbol is used for matrix multiplication, so I'll use that below rather than `np.matmul`.

As a side note, if we want columns indexed as i to j from array A, use the notation `A[:,i:j+1]`. So the first two columns would be `A[:,0:2]`, where columns 0 and 1 are extracted.

```
import numpy as np

A=np.array([[-1,2,4],[1,0,-2],[0,1,1],[1,3,1],[1,2,0]])
U,S,VT=np.linalg.svd(A,full_matrices=0)
Temp=S[0]*U[:,:1] @ VT[:1,:] + S[1]*U[:,1:2] @ VT[1:2,:]
# We could compute Temp as:
Temp2=U[:,0:2] @ np.diag(S[:2]) @ VT[0:2,:]
```

R and the SVD

```
A<-cbind(c(-1,1,0,1,1),c(2,0,1,3,2),c(4,-2,1,1,0))

A.svd<-svd(A)  #Creates a structure holding info
U<-A.svd$u
D<-A.svd$d #Note that R uses UDV^T instead of USV^T
V<-A.svd$v
Temp<-D[1] * U[,1] %*% t(V[,1]) + D[2]* U[,2] %*% t(V[,2])
norm(A-Temp)
Temp2<-U[,1:2] %*% diag(D[1:2]) %*% t(V[,1:2])
norm(A-Temp2)
```

Next, we'll be looking at two applications- One in image processing, and one in computing a generalized inverse.

4.8 Generalized Inverses

Let a matrix A be $m \times n$ with rank k. In this general case, A does not have an inverse (A is not even square). Is there a way of restricting the domain and range of the mapping $\boldsymbol{y} = A\boldsymbol{x}$ so that the **restricted map** is invertible? And if so, how do we compute that inverse?

We'll find that the "inverse function" will be called the **Moore-Penrose Pseudo-Inverse**, and is relatively easy to compute using the SVD of matrix A. Indeed, let's consider the matrix equation:

$$A\boldsymbol{x} = \boldsymbol{b}$$

Replacing A by the reduced SVD (reduced to $m \times k, k \times k, n \times k$ matrices), we have:

$$U\Sigma V^T\boldsymbol{x} = \boldsymbol{b}$$

Since U is $m \times k$ with orthonormal columns, U^TU is the $k \times k$ identity, so we multiply both sides by U^T:

$$U^TU\Sigma V^T\boldsymbol{x} = U^T\boldsymbol{b} \quad \Rightarrow \quad \Sigma V^T = U^T\boldsymbol{b}$$

The matrix Σ is $k \times k$ with singular values along the diagonal. Because the rank is k, we have k non-zero singular values, so Σ is invertible. In the exercises, we'll show that the inverse is found by taking the reciprocal of each diagonal element- We'll see how that's computed in the programming section. Multiply both sides by the inverse:

$$\Sigma^{-1}\Sigma V^T\boldsymbol{x} = \Sigma^{-1}U^T\boldsymbol{b} \quad \Rightarrow \quad V^T\boldsymbol{c} = \Sigma^{-1}U^T\boldsymbol{x}$$

Finally, multiply both sides by V. Remember, $VV^T \neq I$, however, $VV^T\boldsymbol{x}$ is the projection of $\boldsymbol{x}$ into the rowspace of A. We'll call that $\tilde{\boldsymbol{x}}$.

$$\tilde{\boldsymbol{x}} = V\Sigma^{-1}U^T\boldsymbol{b}$$

This matrix product is the pseudo-inverse, denoted by dagger notation

$$A^\dagger = V\Sigma^{-1}U^T$$

You might recall that A is $m \times n$, and the pseudo-inverse $A^\dagger$ is $n \times m$. Just to be clear, this "inverse" will take a general $\boldsymbol{b} \in \mathbb{R}^m$ and map it back to an $\boldsymbol{x}$ in the rowspace of A. If you check, A will map $\boldsymbol{x}$ to the the projection of $\boldsymbol{b}$ to the columnspace of A (unless it is in the columnspace already). That's what we mean when we say that this mapping has been restricted to be between the $k-$dimensional rowspace and columnspace. **Restricted to these two subspaces**, the mapping $\boldsymbol{x} \to A\boldsymbol{x}$ is 1-1 and onto, and invertible!

Finding the rank of A

What we haven't discussed yet is the determination of the rank k. Theoretically, the value of k is the number of non-zero singular values (or non-zero eigenvalues of AA^T or A^TA).

The problem is one that is not theoretical, but computational. We need to determine if a number is zero, or if it is not- Seems simple, but we seldom see *exactly zero* in our computations- In the computations, what typically happens is that the singular values *approach* zero, and then we need to decide on a cut-off: how close to zero is zero?

The cut-off is typically arbitrary and problem-dependent. For example, if there is a clear drop between numbers (like $\sigma_2 = 8$ and $\sigma_3 = 0.01$), then we might go ahead and take the rank to be 2.

Often we look at the eigenvalues of AA^T or A^TA rather than the singular values- The reason is that later we'll see that the eigenvalues actually represent some statistical properties of the data (related to variance). In order to be consistent, rather than looking at the eigenvalues themselves, we'll look at the normalized eigenvalues:

$$\frac{\lambda_1}{\sum_i \lambda_i}, \frac{\lambda_2}{\sum_i \lambda_i}, \ldots, \frac{\lambda_n}{\sum_i \lambda_i}$$

These sum to 1, like probabilities. One idea is to take the rank k to be that scaled eigenvalue such that the sum of the first k values is at least, say, 0.99.

There are other ways as well. In fact, for large matrices, Gavish and Donoho[1] published some work on finding the optimal rank.

[1]Gavish and Donoho, "*The Optimal Hard Threshold for Singular Values is* $4/\sqrt{3}$", IEEE Transactions on Information Theory, v 60, Issue 8, 2014, pg 5040 - 5053

Programming Examples in Matlab, Python and R

The problem we'll be solving on the computer is to find the solution to the following matrix equation by explicitly computing the pseudo-inverse.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 4 \\ -1 \\ 2 \\ 1 \end{bmatrix}$$

In the code below, we will first construct the SVD of A, and then inspect the diagonal values of S to determine the rank (in this case, the rank is 2). We then compute the pseudo-inverse, `Ap`, and find the (least squares) solution to the equation above.

Finally, we verify our theory by comparing $A\mathbf{x}$ and the projection of $\boldsymbol{b}$ into the columnspace of A (they should be the same). Similarly, we compare our solution $\boldsymbol{x}$ with the projection of $\boldsymbol{x}$ into the rowspace (the projection of the projection does not change, so these should be the same vector).

We should note that each of these languages has the pseudo-inverse built in (probably `pinv`), but before we use it, we want to be sure we understand how it is constructed- The (possibly) non-trivial part of the construction is the determination of the rank of A.

- Matlab:

```
A=[1 2 3;4 5 6;7 8  9;10 11 12];
b=[4;-1;2;1];
[U,S,V]=svd(A);
S
Ap=V(:,1:2)*diag(1./diag(S(1:2,1:2)))*U(:,1:2)';  %Pseudo-inverse
xsoln=Ap*b                                          %The "solution" to our system.
A*xsoln                                             %This is what Ax actually is.
U(:,1:2)*U(:,1:2)'*b                                %See if Ax is the proj of b into
                                                    %   col(A).
V(:,1:2)*V(:,1:2)'*xsoln                            %See if x is the same as proj of x
                                                    %   into row(A).
```

- Python:

```
import numpy as np

A=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
b=np.array([[4],[-1],[2],[1]])
U,S,VT=np.linalg.svd(A,full_matrices=0)
print(S)      # In Python, S is not a diagonal matrix

Ap= VT[0:2,:].T @ np.diag(1/S[0:2]) @ U[:,0:2].T
xsoln=Ap @ b
print(A @ xsoln)                    #This vector and the next should be the same.
print(U[:,0:2] @ U[:,0:2].T @ b)
print(xsoln)                        #This vector and the next should be the same.
print(VT[0:2,:].T @ VT[0:2,:] @ xsoln)
```

- R:

```
A<-cbind(c(1,4,7,10),c(2,5,8,11),c(3,6,9,12))
```

```
b=cbind(c(4,-1,2,1))

A.svd<-svd(A)  #Creates a structure holding info
U<-A.svd$u
S<-A.svd$d #Note that R uses UDV^T instead of USV^T
V<-A.svd$v

Ap=V[,1:2] %*% diag(1/S[1:2]) %*% t(U[,1:2])
xsoln=Ap %*% b

A%*%xsoln                          #This vector and the next should be equal
U[,1:2] %*% t(U[,1:2]) %*% b
V[,1:2] %*% t(V[,1:2]) %*% xsoln  #This vector and the next should be equal
xsoln
```

4.9 Exercises

Before working through the exercises, be sure you've tried out the SVD and pseudo-inverse code examples so you have the template files ready for the homework.

1. In each of the programming languages, we built Σ^{-1} by replacing each diagonal element with its reciprocal. Because we had rank 2, we only used the first two elements.

   What happens if you forget that, and compute (in each language, respectively):

   ```
   Matlab:  U*diag(1./diag(S))*V'        Python:  VT.T @ np.diag(1/S) @ U.T
   In R:  V%*% diag(1/S)%*% t(U)
   ```

   Compare these to your previous pseudo-inverse. Notice anything? What happened?

2. Some sources say that, if A is full rank (let's assume that $m > n$ with rank n), then the pseudo-inverse can be computed as the following. We want to verify this using the SVD.

   $$A^{\dagger} = (A^T A)^{-1} A^T$$

   Hint: Start with $A = U\Sigma V^T$, and note that $\Sigma^{-1}\Sigma^{-1} = \Sigma^{-2}$, and $\Sigma^{-2}\Sigma = \Sigma^{-1}$, since Σ is a diagonal matrix.

3. Let A be $m \times n$ with rank k, so that $A^{\dagger} = U\Sigma^{-1}V^T$ is from the rank k SVD of A.

   (a) Show that the pseudo-inverse of the pseudo-inverse is the matrix A: $(A^{\dagger})^{\dagger} = A$

   (b) Simplify the expression $AA^{\dagger}$ using the SVD.

   (c) Simplify the expression $A^{\dagger}A$ using the SVD.

   (d) Given $A\boldsymbol{x} = \boldsymbol{b}$, solve for $\boldsymbol{x}$ by first multiplying both sides by $A^{\dagger}$, and use your previous simplification to simplify the equation.

4. Consider

   $$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

   (a) Before solving this problem, what are the dimensions of the four fundamental subspaces?

   (b) Use Matlab, Python or R to compute the SVD of the matrix A, and solve the problem by computing the pseudoinverse of A explicitly.

(c) Check your answer explicitly and verify that $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are in the rowspace and columnspace, similar to the example.

5. Consider

$$\begin{bmatrix} 2 & 1 & -1 & 3 \\ -1 & 0 & 1 & -2 \\ 7 & 2 & -5 & 12 \\ -3 & -2 & 0 & -4 \\ 4 & 1 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x4 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 0 \\ -2 \\ 6 \end{bmatrix}$$

(a) Find the dimensions of the four fundamental subspaces by using the SVD of A (in Matlab, Python or R).

(b) Solve the problem.

(c) Check your answer explicitly and verify that $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are in the rowspace and columnspace.