CSE 519: Data Science
Steven Skiena
Stony Brook University

Lecture 15: Linear Algebra Review

Get the Matrix!

The most critical part of your data science project is reducing all the information you can find one or more data matrices, ideally as large as possible.

Rows are examples.

Columns are distinct features/attributes.

You need to be building your matrix now!

Linear Algebra

Linear algebra is the mathematics of matrices.

This makes it the language of data science.

Many machine learning algorithms are best understood through linear algebra.

You presumably had an undergraduate course in linear algebra, but here I will review what you need to know.

What Can n*m Matrices Represent?

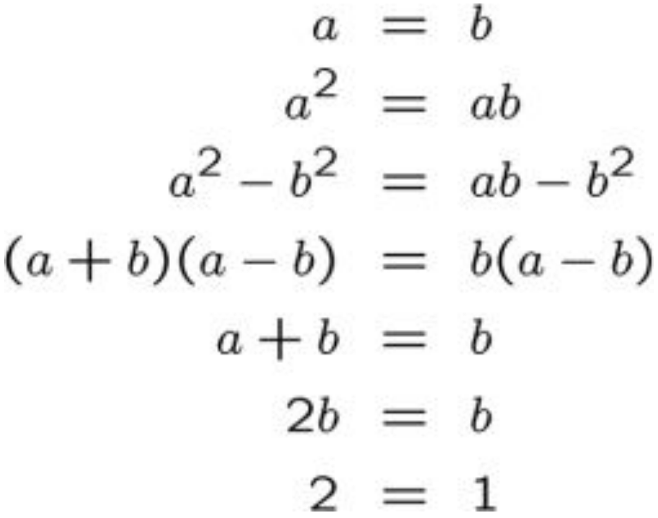
- Data: rows are objects, columns features.
- Geometric point sets: rows are points, columns are dimensions
- Systems of Equations: rows are equations, columns are coefficients for each variable.
- Graphs/Networks: M[i,j] denotes the number of edges from vertex i to vertex j.
- Vectors: any row, column or d*1 matrix

Linear Algebra Formulae

- Concise formulas written as products of matrices provides great power.
- Algebraic substitution coupled with a rich set of identities yields elegant, mechanical ways to manipulate such formulae.
- But such strings of operations can, I find, be difficult to interpret and understand.

$$w = (A^T A)^{-1} A^T b$$

Algebraic Proof: 2=1



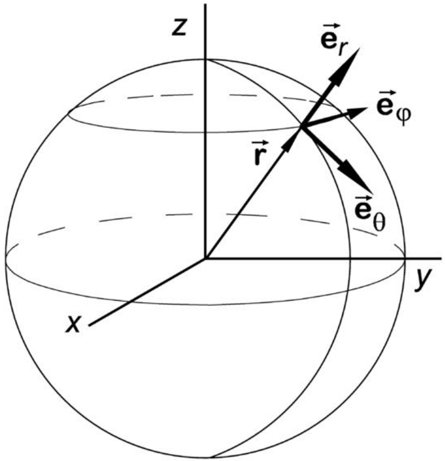
$$\begin{aligned} a &= b \\ a^2 &= ab \\ a^2 - b^2 &= ab - b^2 \\ (a+b)(a-b) &= b(a-b) \\ a+b &= b \\ 2b &= b \\ 2 &= 1 \end{aligned}$$

Lessons from the Proof

- Algebraic proofs do not, to me, generally carry intuition about *why* things are work.
- They are easier to verify than to create.
- Even so, there are special cases / singularities to watch for, like division by 0.
- In linear algebra, such cases include singular / non-invertible matrices.

Points vs. Vectors

Points in *d* dimensions can be represented as unit vectors (points on the sphere), plus their magnitudes.

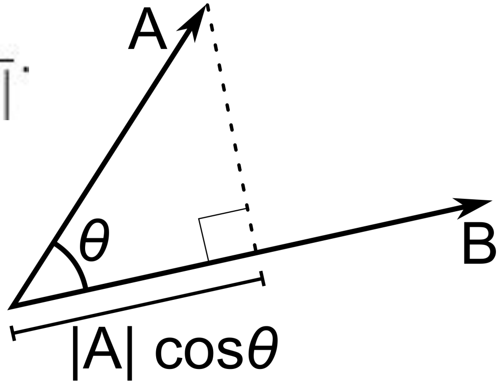


Distances between points become angles between vectors, for purposes of comparison.
Ignoring magnitudes is a form of scaling, making all points directly comparable.

Angles between Vectors

To compute angle AB: $\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}.$

Cos(0)=1, Cos(Pi/2)=0, Cos(Pi)=-1



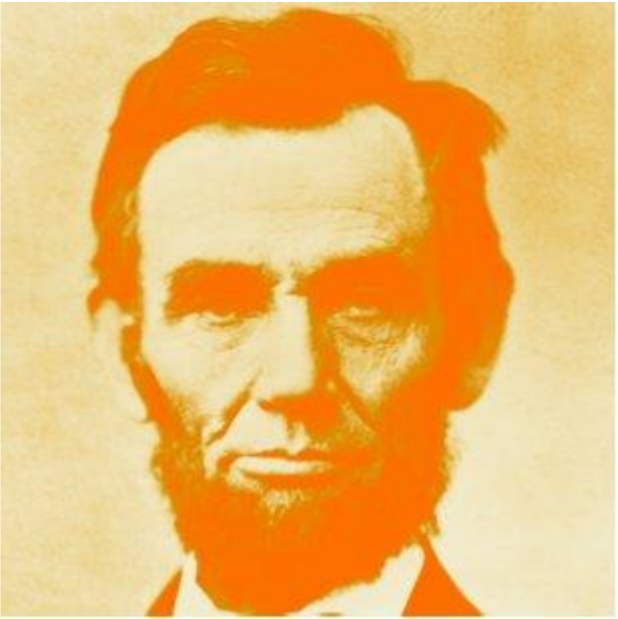
Scores like correlation coefficients:

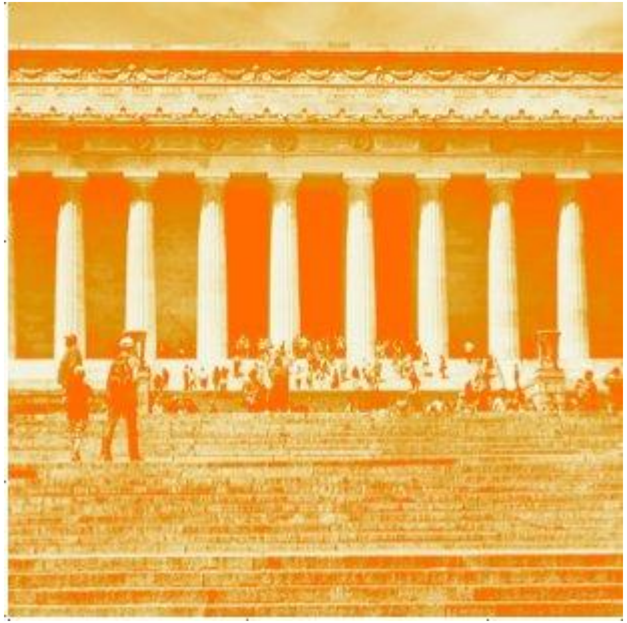
Cos = correlation of mean zero variables!

For unit vectors, ||A||=||B||=1, so the angle between A and B is defined by the dot product.

Visualizing Matrix Operations

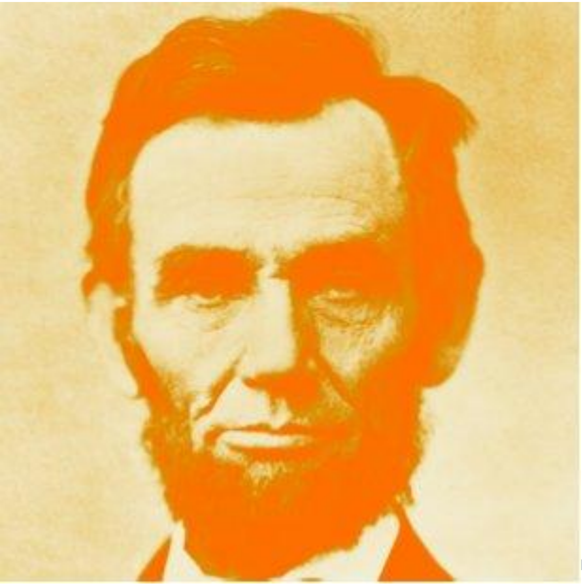
We use images to represent matrices

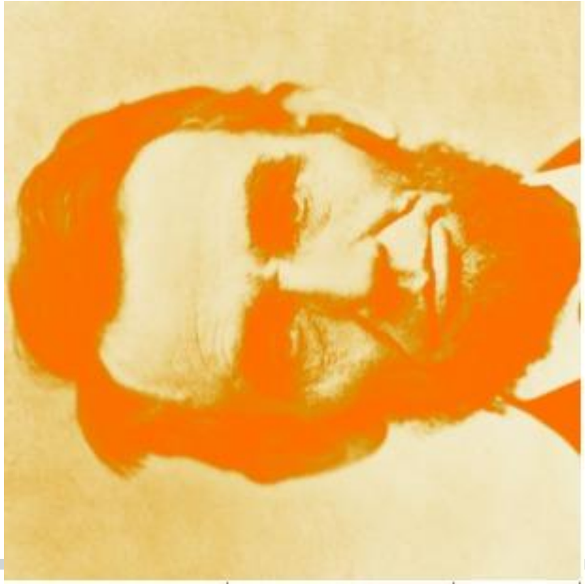




A Matrix and its Transpose

The transpose of a matrix M interchanges rows and columns, turning an a*b matrix to a b*a matrix.



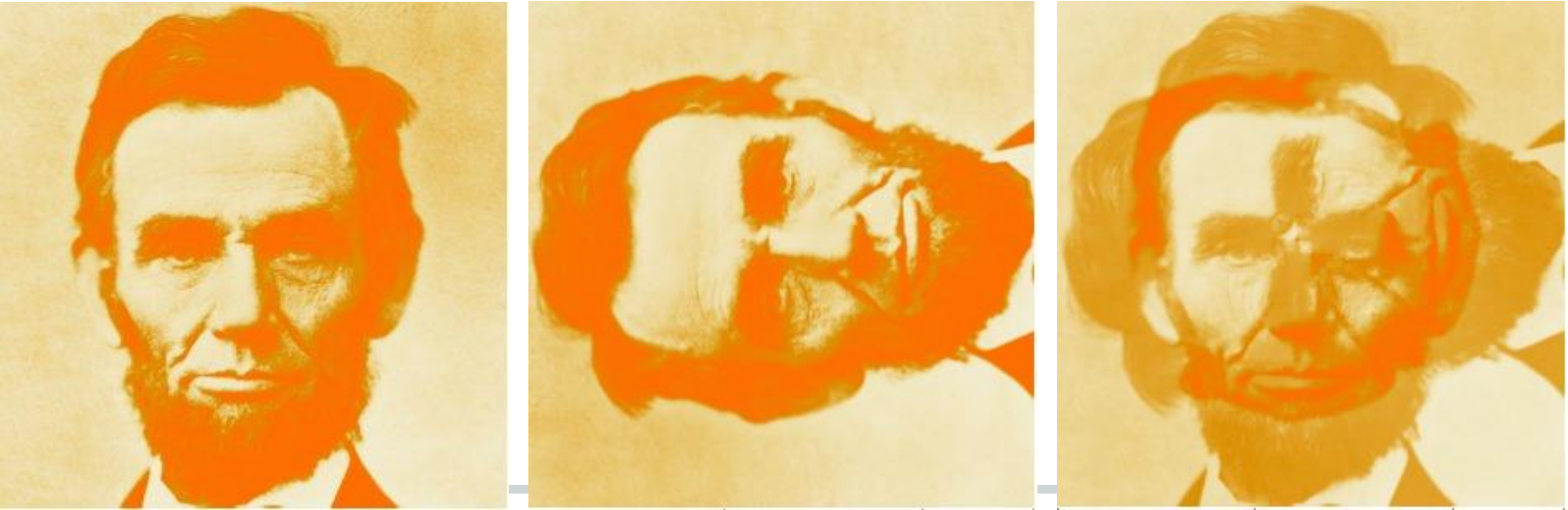


$$M_{ij}^T = M_{ji}$$

Note that colors get rescaled when magnitudes change.

Addition and Transposition

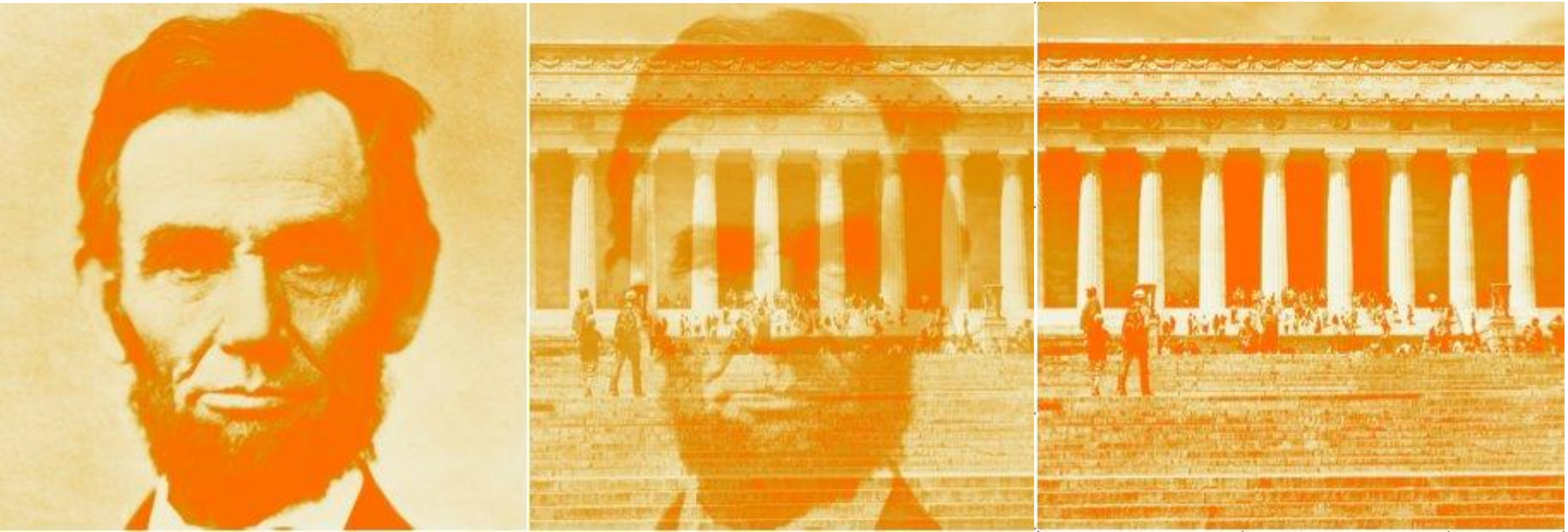
A mix of scalar multiplication and addition.



Linear Combination: B=(A+C)/2

Mix of scalar multiplication/ addition.

$$\alpha \cdot A + (1 - \alpha) \cdot B$$



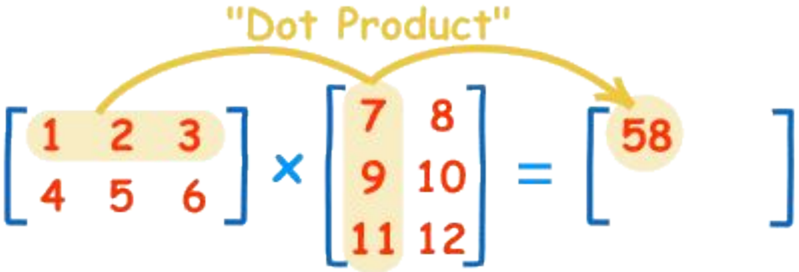
Matrix Multiplication / Dot Products

The product A*B is defined by: $C_{i,j} = \sum_{i=1}^{k} A_{i,k} \cdot B_{k,j}$

A*B must share inner dimensions to multiply.

Each element of the product matrix is a dot product of row/column vectors.

Dot products measure how “in sync” the two vectors are, as in computing covariance or correlation.



Properties of Matrix Multiplication

It is associative but not commutative: $\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}=\begin{bmatrix}2 & 1\\1 & 0\end{bmatrix}$

$$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}\cdot\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}=\begin{bmatrix}1 & 2\\1 & 1\end{bmatrix}$$

$$\begin{aligned}&\left(\begin{bmatrix}3 & 2\\-1 & 0\end{bmatrix}\begin{bmatrix}-2 & 3\\4 & 2\end{bmatrix}\right)\begin{bmatrix}-1 & 5\\1 & 2\end{bmatrix}\\&=\begin{bmatrix}2 & 13\\2 & -3\end{bmatrix}\begin{bmatrix}-1 & 5\\1 & 2\end{bmatrix}\\&=\begin{bmatrix}11 & 36\\-5 & 4\end{bmatrix}\end{aligned}$$

$$\begin{aligned}&\begin{bmatrix}3 & 2\\-1 & 0\end{bmatrix}\left(\begin{bmatrix}-2 & 3\\4 & 2\end{bmatrix}\begin{bmatrix}-1 & 5\\1 & 2\end{bmatrix}\right)\\&=\begin{bmatrix}3 & 2\\-1 & 0\end{bmatrix}\begin{bmatrix}5 & -4\\-2 & 24\end{bmatrix}\\&=\begin{bmatrix}11 & 36\\-5 & 4\end{bmatrix}\end{aligned}$$

Multiplication by the identity commutes: $IA = AI = A$

Although the *O(xyz)* algorithm is simple to program, faster, more numerically stable algorithms exist in highly optimized libraries.

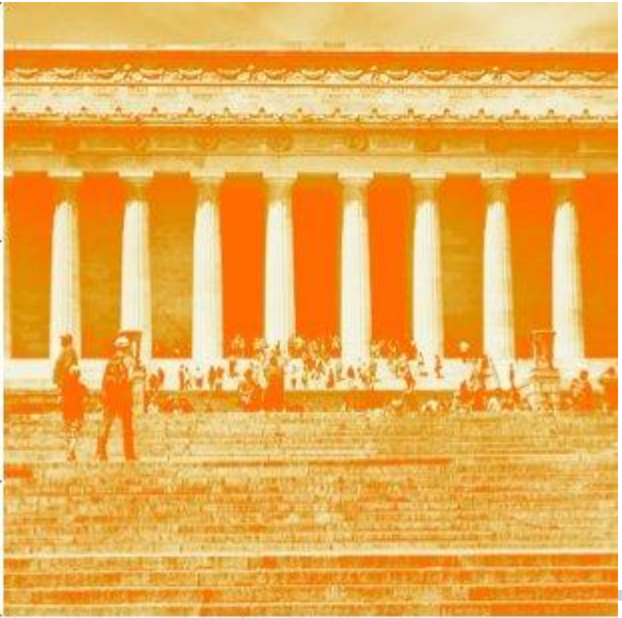
Multiplying Feature Matrices

Suppose A is an n*d data matrix. What is A times its transpose?:

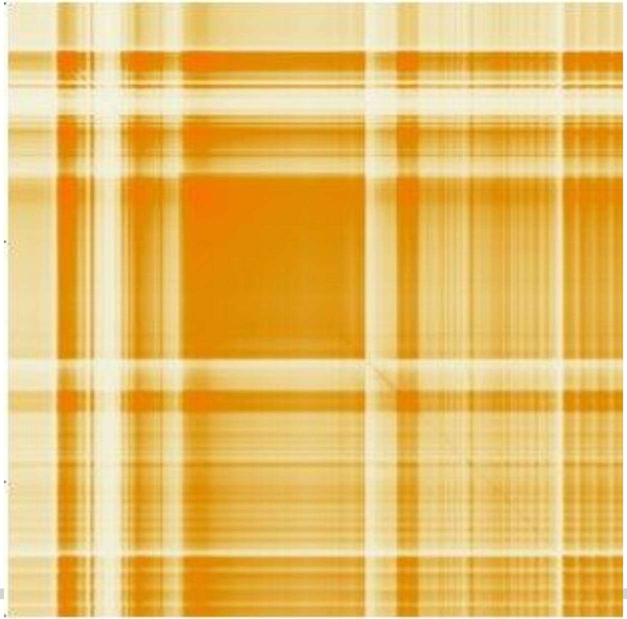
- $A \cdot A^T$ is an n*n matrix of dot products, measuring “in sync-ness” among points.
- $A^T \cdot A$ is a d*d matrix of dot products, measuring “in sync-ness” among features.

These are called covariance matrices.

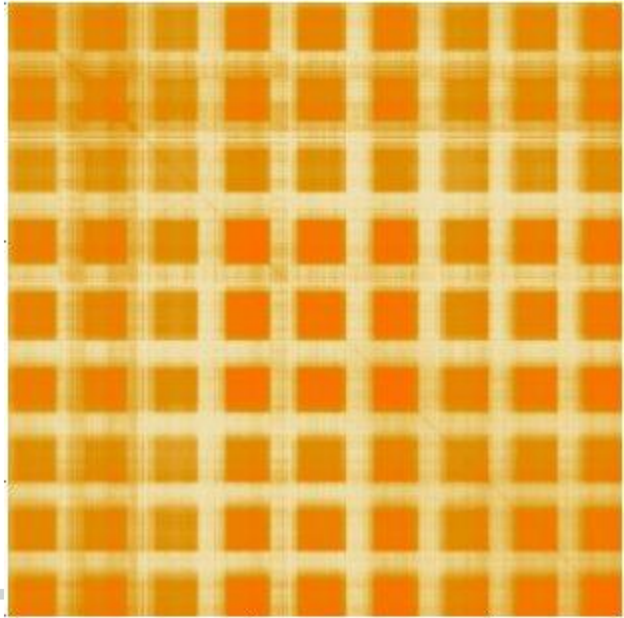
Row or Column Covariance Matrix?



$A \cdot A^T$



$A^T \cdot A$



Interpreting Matrix Multiplication

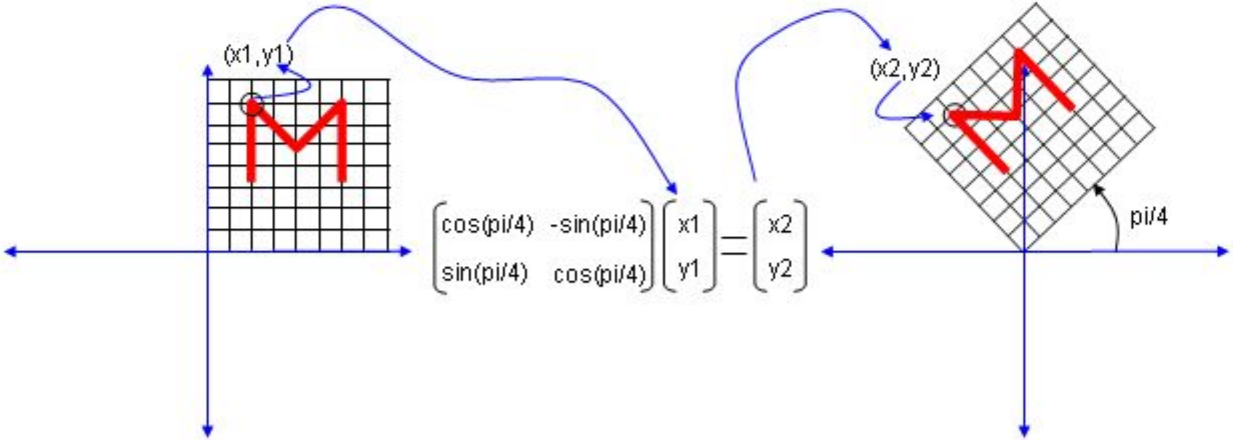
- Multiplying 0/1 adjacency matrices yield paths of length two: *a[i,k]=a[i,j]*a[j,k]*
- Multiplication by permutation matrices rearrange rows/columns:

$$P=\begin{pmatrix}0&0&1&0\\1&0&0&0\\0&0&0&1\\0&1&0&0\end{pmatrix}\quad M=\begin{pmatrix}m_{11}&m_{12}&m_{13}&m_{14}\\m_{21}&m_{22}&m_{23}&m_{24}\\m_{31}&m_{32}&m_{33}&m_{34}\\m_{41}&m_{42}&m_{43}&m_{44}\end{pmatrix}$$

$$PM=\begin{pmatrix}m_{31}&m_{32}&m_{33}&m_{34}\\m_{11}&m_{12}&m_{13}&m_{14}\\m_{41}&m_{42}&m_{43}&m_{44}\\m_{21}&m_{22}&m_{23}&m_{24}\end{pmatrix}$$

Interpreting Matrix Multiplication

- Rotating points in space:



$$\begin{bmatrix} \cos(pi/4) & -\sin(pi/4) \\ \sin(pi/4) & \cos(pi/4) \end{bmatrix} \begin{bmatrix} x1 \\ y1 \end{bmatrix} = \begin{bmatrix} x2 \\ y2 \end{bmatrix}$$

Multiplying something by the right matrix can have magic properties, in arbitrary dimensions.

Dividing Matrices

The inverse operation to multiplication is division.

An important special case of division is inversion: A*A^{-1} = I implies A^{-1}=I/A

In fact it is equivalent, because A/B = A*B^{-1}

Matrix Inversion

A^{-1} is the multiplicative inverse of A if A * A^{-1} = I, where I is the identity matrix.

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

If matrix A has an inverse, it can be computed by solving a linear system using Gaussian elimination.

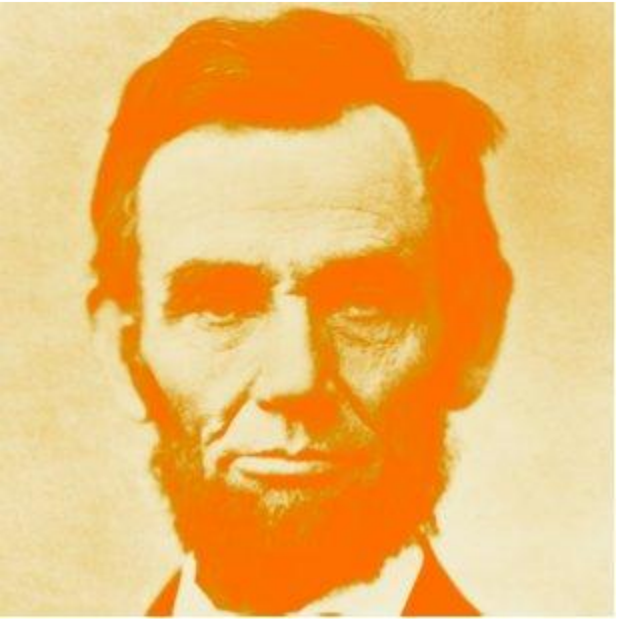
$$[\mathbf{A} \quad \mathbf{I}] = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 3 & 0 & 1 & 0 \\ 4 & -3 & 8 & 0 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 3 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & -3 & -4 & 0 & -4 & 1 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 0 & 3 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 3 & -4 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & -9/2 & 7 & -3/2 \\ 0 & 1 & 0 & -2 & 4 & -1 \\ 0 & 0 & 1 & 3/2 & -2 & 1/2 \end{bmatrix}$$

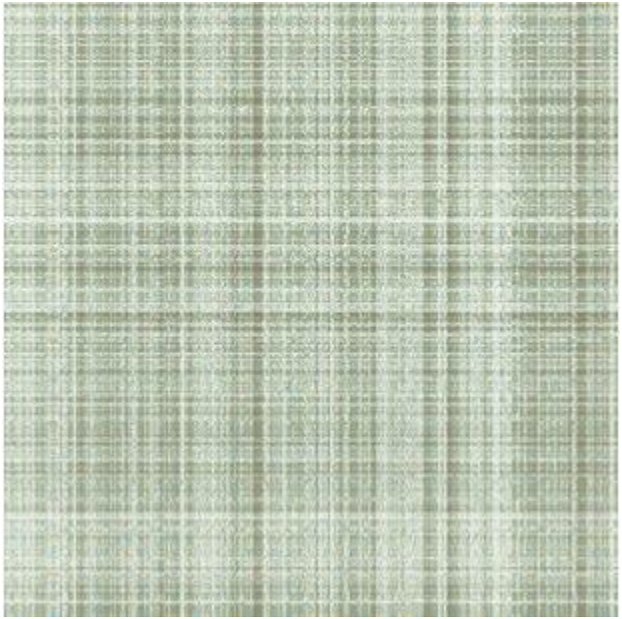
$$\mathbf{A}^{-1} = \begin{bmatrix} -9/2 & 7 & -3/2 \\ -2 & 4 & -1 \\ 3/2 & -2 & 1/2 \end{bmatrix}$$

Inverse of Lincoln

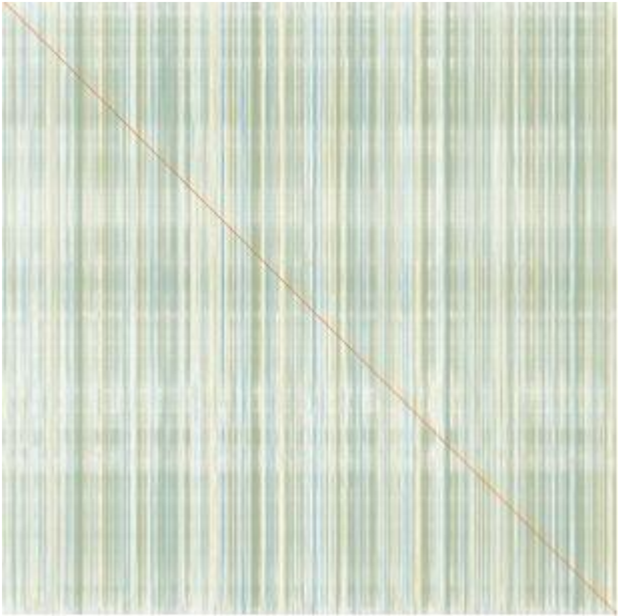
L



L^{-1}



LL^{-1}



Matrix Inversion and Linear Systems

Multiplying both sides of Ax = b by the inverse of A yields: *(A^{-1}A)x = A^{-1}b* or *x = A^{-1}b*

Thus solving linear equations is equivalent to matrix inversion.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 8 \\ 3 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} -232 \\ 129 \\ 19 \end{bmatrix} = \begin{bmatrix} -9.28 \\ 5.16 \\ 0.76 \end{bmatrix}.$$

The inverse makes it cheap to evaluate many b vectors. However, Gaussian elimination is more numerically stable than inversion.

Matrix Rank

Systems of equations are underdetermined if rows can be expressed as linear combinations of other rows.

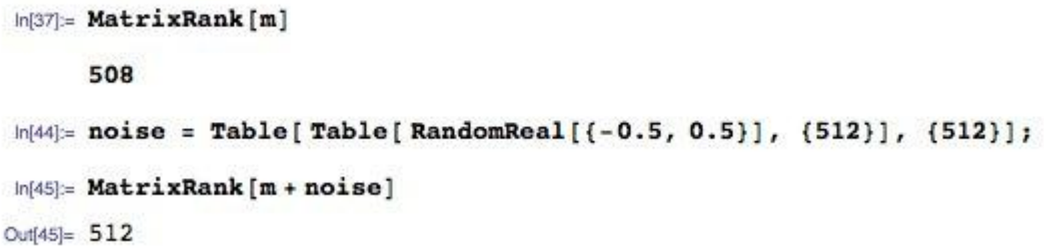
The rank of a matrix is a measure of the number of linearly independent rows.

An n*n matrix should be rank n for all operations to be properly defined on it.

Increasing Lincoln Memorial’s Rank

Some rows of the Lincoln Memorial are not linearly independent, so it is not full rank.

Adding small amounts of random noise increases rank without serious image distortion.



```
In[37]:= MatrixRank[m]

         508

In[44]:= noise = Table[Table[RandomReal[{-0.5, 0.5}], {512}], {512}];

In[45]:= MatrixRank[m + noise]

Out[45]= 512
```

Factoring Matrices

Many important machine learning algorithms can be viewed as factoring a matrix.

Suppose *n*m* matrix *A* can be expressed as the product *B*C*, i.e an *n*k* matrix times a *k*m* matrix.

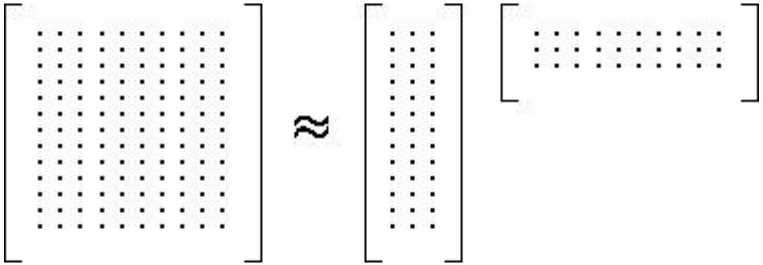
If k<min(n,m), B and C compress matrix A.

Further, B is a small feature matrix replacing A.

Factoring Word-Document Matrices

If A is a document/word co-occurrence matrix, and A=BC, where B is d*k and C is k*w:

- B is a compressed feature vector for docs
- C is a compressed feature vector for words



LU Decomposition

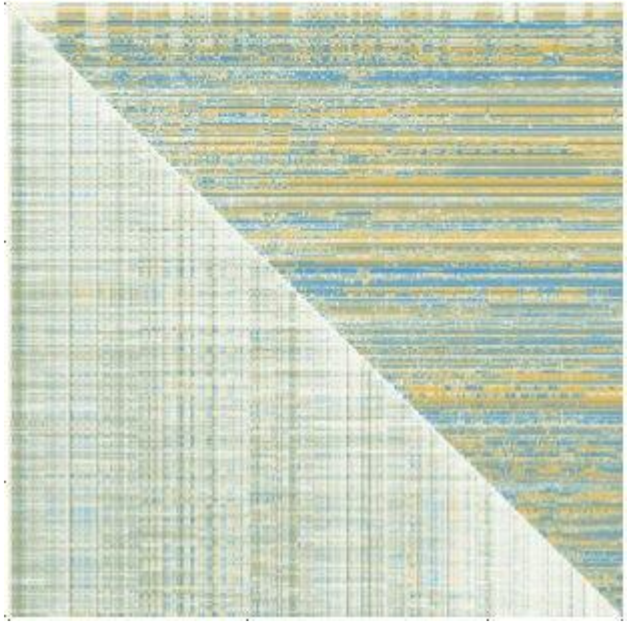
Factoring a matrix M representing lower and upper triangular matrices L and U prove useful in solving linear systems.

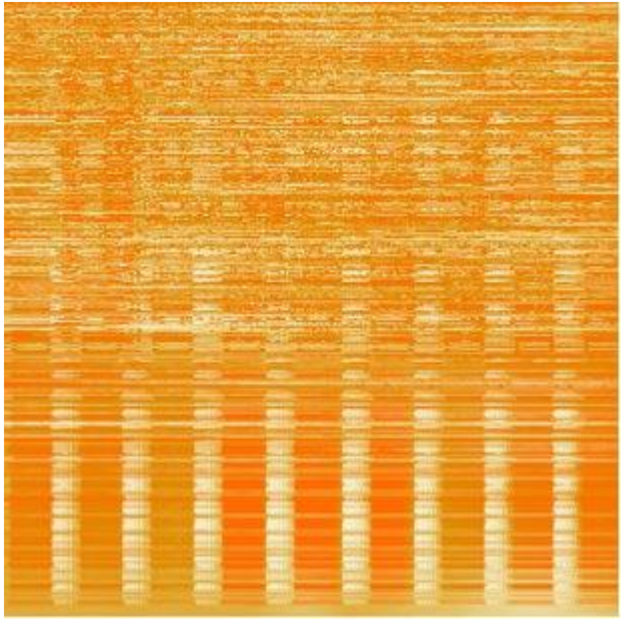
The determinant of M is the product of the main diagonal elements of U.

A determinant of 0 means the matrix is not full rank.

LU Decomposition of Memorial

Rows were permuted by this solver.





Lessons from Lincoln

- Multiplying the factors of the matrix did not reconstruct it exactly, due to numerical instability.
- The high matrix condition number should have tipped us off that we had trouble.
- Still, the gross features of the data are largely preserved.

Eigenvalues and Eigenvectors

Multiplying a vector U by a matrix A can have the same effect as multiplying it by a scalar l.

$$\begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ -1 \end{bmatrix} = -6 \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

Thus the eigenvalue-eigenvector pair (U,l) must encode a lot of information about matrix A!

Computing Eigenvalues

The n distinct eigenvalues of a rank n matrix can be found by factoring its characteristic equation:

$$\det(A-\lambda I)=\begin{vmatrix}-\lambda & 1 & 1\\ 1 & -\lambda & 1\\ 1 & 1 & -\lambda\end{vmatrix}$$

$$=-\lambda^3+3\lambda+2$$

$$=(\lambda-2)(\lambda+1)^2 \qquad \lambda_1=2,\ \lambda_{2,3}=-1$$

Faster algorithms exist to find the largest eigenvalues, which are the most important.

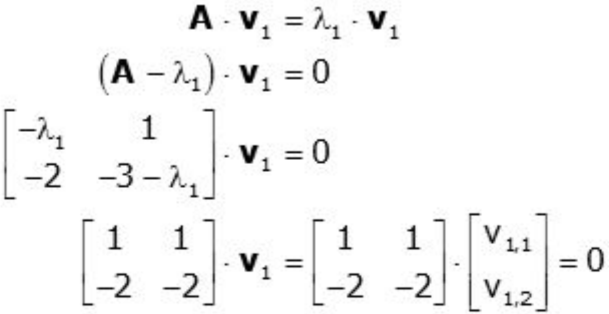
Properties of Eigenvalues/vectors

- A full-rank matrix has n vector-value pairs.
- Each pair of vectors from a symmetric matrix are mutually orthogonal, like x-y axes. E.g. the dot product is of (2,-1) and (1,2) is zero.
- Thus eigenvectors can play the role of dimensions or basis in n-dimensional space.

Vectors/values are found by solving linear systems.

Computing Eigenvectors

The vector associated with a given eigenvalue can be computed by solving a linear system:



$$\mathbf{A}\cdot\mathbf{v}_1 = \lambda_1\cdot\mathbf{v}_1$$

$$(\mathbf{A}-\lambda_1)\cdot\mathbf{v}_1 = 0$$

$$\begin{bmatrix} -\lambda_1 & 1 \\ -2 & -3-\lambda_1 \end{bmatrix}\cdot\mathbf{v}_1 = 0$$

$$\begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix}\cdot\mathbf{v}_1 = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix}\cdot\begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} = 0$$

Another approach uses v’ = (A*v)/l to compute approximations to v until it converges.

Eigenvalue Decomposition

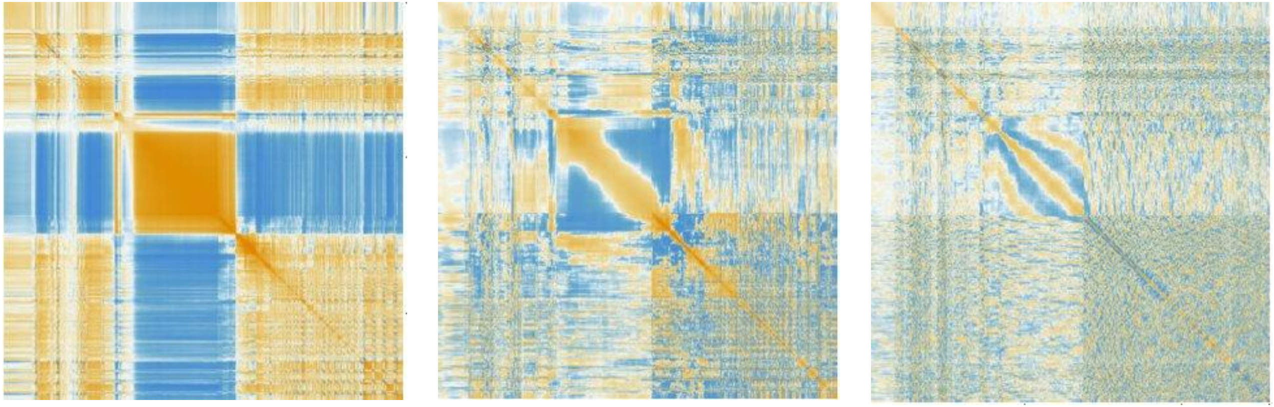
Any *n*n* symmetric matrix *M* can be decomposed into its n eigenvector products:

$$M = \sum_{i=1}^{n} \lambda_i U_i {U_i}^T$$

Larger eigenvalues correspond to more important vector products.

Reconstructing a Covariance Matrix

Summing only the largest vectors performs dimension reduction, identifying the most important features of the matrix.



Covariance matrix error for the Lincoln memorial reduces when summing the 1, 5, and 50 largest eigenvectors for n=512

Singular Value Decomposition

The SVD of an n*m matrix M factors it $M = UDV^T$ where D is diagonal (weighted identity matrix)

Thus UD weights each column of U by D, as does DV^T.

Retaining only the rows/column with large weights permits us to compress m features with relatively little loss.

Reconstruction from SVD

The outer product of vectors yields a matrix

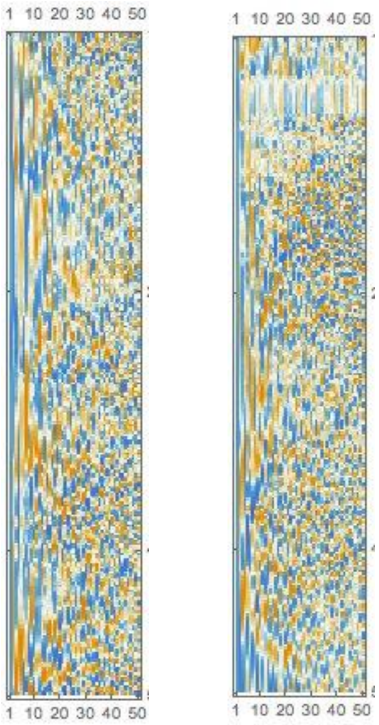
$$P = X \otimes Y \qquad\qquad P[j,k] = X[j]Y[k]$$

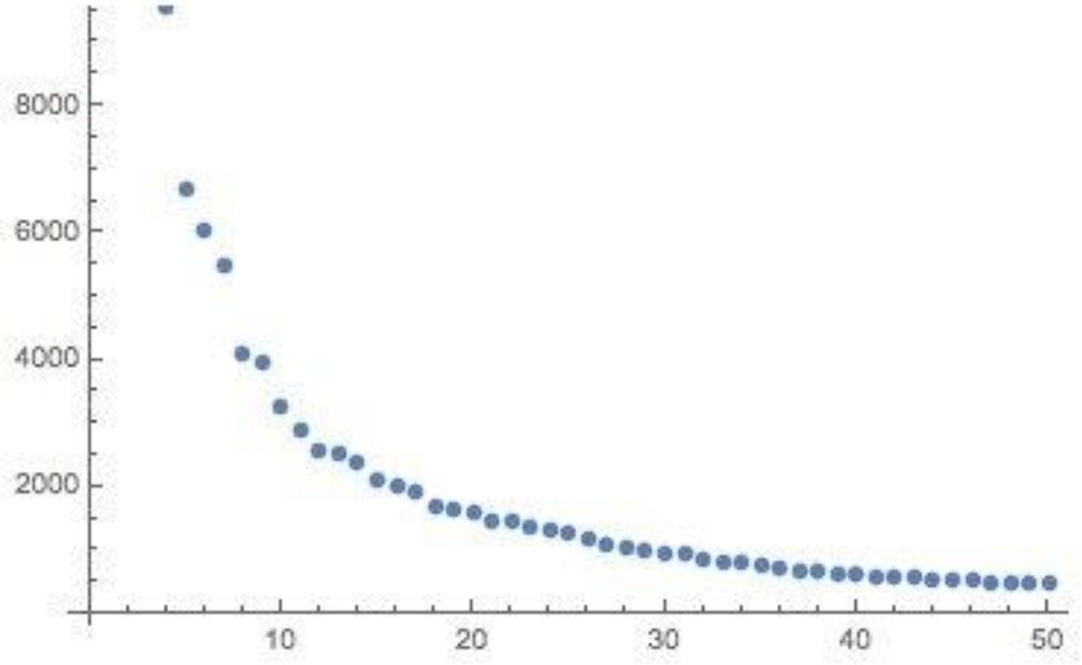
Matrix M can be expressed a sum of outer products from SVD: (UD) k and (V^T)_k.

$$C = A \cdot B = \sum_k A_k \otimes B_k^T$$

Summing only the largest matrix products produces an approximation of M

Error Declines with Dimensionality





Reconstructing Lincoln

Lincoln’s face from 5 and 50 singular values, a substantial compression of the original matrix.