Linear Algebra for Machine Learning

Sargur N. Srihari
srihari@cedar.buffalo.edu

Overview

- Linear Algebra is based on continuous math rather than discrete math
  - Computer scientists have little experience with it
- Essential for understanding ML algorithms
- Here we discuss:
  - Discuss scalars, vectors, matrices, tensors
  - Multiplying matrices/vectors
  - Inverse, Span, Linear Independence
  - SVD, PCA

Scalar

- Single number
- Represented in lower-case italic x
  - E.g., let $x \in \mathbb{R}$ be the slope of the line
    - Defining a real-valued scalar
  - E.g., let $n \in \mathbb{N}$ be the number of units
    - Defining a natural number scalar

Vector

- An array of numbers
- Arranged in order
- Each no. identified by an index
- Vectors are shown in lower-case bold

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} \Rightarrow \mathbf{x}^T = [x_1, x_2, .. x_n]$$

- If each element is in R then x is in R^n
- We think of vectors as points in space
  - Each element gives coordinate along an axis

Matrix

- 2-D array of numbers
- Each element identified by two indices
- Denoted by bold typeface $\boldsymbol{A}$
- Elements indicated as $A_{m,n}$
  - E.g., $\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$
- $A_{i:}$ is ith row of A, $A_{:j}$ is jth column of $\boldsymbol{A}$
- If A has shape of height m and width n with real-values then $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Tensor

- Sometimes need an array with more than two axes
- An array arranged on a regular grid with variable number of axes is referred to as a tensor
- Denote a tensor with bold typeface: $\mathbf{A}$
- Element (i,j,k) of tensor denoted by $\mathrm{A}_{i,j,k}$

Transpose of a Matrix

- Mirror image across principal diagonal

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \\ A_{1,3} & A_{2,3} & A_{3,3} \end{bmatrix}$$

- Vectors are matrices with a single column
  - Often written in-line using transpose

$$\mathbf{x} = [x_1,..,x_n]^{\mathrm{T}}$$

- Since a scalar is a matrix with one element

$$a = a^{\mathrm{T}}$$

Matrix Addition

- If A and B have same shape (height m, width n)

$$C = A + B \Rightarrow C_{i,j} = A_{i,j} + B_{i,j}$$

- A scalar can be added to a matrix or multiplied by a scalar

$$D = aB + c \Rightarrow D_{i,j} = aB_{i,j} + c$$

- Vector added to matrix (non-standard matrix algebra)

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{b} \Rightarrow \boldsymbol{C}_{i,j} = A_{i,j} + b_j$$

  – Called broadcasting since vector b is added to each row of A

Multiplying Matrices

- For product $C=AB$ to be defined, A has to have the same no. of columns as the no. of rows of B
- If A is of shape mxn and B is of shape nxp then *matrix product* C is of shape mxp

$$C = AB \Rightarrow C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

- Note that the standard product of two matrices is not just the product of two individual elements

Multiplying Vectors

- Dot product of two vectors x and y of same dimensionality is the matrix product $\mathrm{x}^{\mathrm{T}}\mathrm{y}$
- Conversely, matrix product $C=AB$ can be viewed as computing C_{ij} the dot product of row i of A and column j of B

Matrix Product Properties

- Distributivity over addition: $A(B+C)=AB+AC$
- Associativity: $A(BC)=(AB)C$
- Not commutative: $AB=BA$ is not always true
- Dot product between vectors is commutative: $x^Ty=y^Tx$
- Transpose of a matrix product has a simple form: $(AB)^T=B^TA^T$

Linear Transformation

- $\boldsymbol{Ax}=\boldsymbol{b}$
  - where $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ and $\boldsymbol{b}\in\mathbb{R}^{n}$
  - More explicitly

$$\boxed{\begin{array}{c} A_{11}x_1 + A_{12}x_2 + \ldots. + A_{1n}x_n = b_1 \\ A_{21}x_1 + A_{22}x_2 + \ldots. + A_{2n}x_n = b_2 \\ \\ A_{n1}x_1 + A_{m2}x_2 + \ldots. + A_{n,n}x_n = b_n \end{array}}$$

n equations in
n unknowns

$$A=\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{nn} \end{bmatrix} \quad \boldsymbol{x}=\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{b}=\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

n x n $\qquad$ n x 1 $\qquad$ n x 1

Can view A as a *linear transformation* of vector $\boldsymbol{x}$ to vector $\boldsymbol{b}$

- Sometimes we wish to solve for the unknowns $x=\{x_1,..,x_n\}$ when A and $\boldsymbol{b}$ provide constraints

Identity and Inverse Matrices

- Matrix inversion is a powerful tool to analytically solve $\boldsymbol{Ax=b}$
- Needs concept of Identity matrix
- Identity matrix does not change value of vector when we multiply the vector by identity matrix
  - Denote identity matrix that preserves n-dimensional vectors as I_n
  - Formally $I_n \in \mathbb{R}^{n \times n}$ and $\forall \mathbf{x} \in \mathbb{R}^n, I_n \mathbf{x} = \mathbf{x}$
  - Example of I_3 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Matrix Inverse

- Inverse of square matrix A defined as $A^{-1}A = I_n$
- We can now solve $A\boldsymbol{x}=\boldsymbol{b}$ as follows:

$$
\begin{aligned}
&A\boldsymbol{x} = \boldsymbol{b}\\
&A^{-1}A\boldsymbol{x} = A^{-1}\boldsymbol{b}\\
&I_n\boldsymbol{x} = A^{-1}\boldsymbol{b}\\
&\boldsymbol{x} = A^{-1}\boldsymbol{b}
\end{aligned}
$$

- This depends on being able to find A^{-1}
- If A^{-1} exists there are several methods for finding it

Solving Simultaneous equations

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

  where A is $(M+1)$ x $(M+1)$

  $\boldsymbol{x}$ is $(M+1)$ x 1: set of weights to be determined

  $\boldsymbol{b}$ is N x 1

- Two closed-form solutions

  1. Matrix inversion $\boldsymbol{x}=\mathrm{A}^{-1}\boldsymbol{b}$
  2. Gaussian elimination

Linear Equations: Closed-Form Solutions

1. Matrix Formulation: Ax=b
Solution: $x=A^{-1}b$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \quad &\;\; \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

2. Gaussian Elimination followed by back-substitution

$$\begin{aligned} x + 3y - 2z &= 5 \\ 3x + 5y + 6z &= 7 \\ 2x + 4y + 3z &= 8 \end{aligned}$$

$L_2\text{-}3L_1 \rightarrow L_2$ $\quad L_3\text{-}2L_1 \rightarrow L_3$ $\quad -L_2/4 \rightarrow L_2$

$$\left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 3 & 5 & 6 & 7 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 0 & -2 & 7 & -2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & -2 & 7 & -2 \end{array}\right]$$

$$\sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & 0 & 9 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

Example: System of Linear Equations in Linear Regression

- Instead of $\boldsymbol{A}\mathrm{x}=\boldsymbol{b}$
- We have $\boxed{\mathbf{\Phi w}=\mathbf{t}}$
  - where Φ is design matrix of m features (basis functions $\phi_i(x_j)$) for samples x_j, $\mathbf{t}$ is targets of sample
  - We need weight $\mathbf{w}$ to be used with m basis functions to determine output

$$\boxed{y(\mathbf{x},\mathbf{w})=\sum_{i=1}^{m} w_i \phi_i(\mathbf{x})}$$

Disadvantage of closed-form solutions

- If A^{-1} exists, the same A^{-1} can be used for any given $\boldsymbol{b}$
  - But A^{-1} cannot be represented with sufficient precision
  - It is not used in practice
- Gaussian elimination also has disadvantages
  - numerical instability (division by small no.)
  - $O(n^3)$ for n x n matrix
- Software solutions use value of $\boldsymbol{b}$ in finding $\boldsymbol{x}$
  - E.g., difference (derivative) between $\boldsymbol{b}$ and output is used iteratively

How many solutions for $\boldsymbol{Ax=b}$ exist?

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \ldots + A_{2n}x_n &= b_2 \\ \\ A_{m1}x_1 + A_{m2}x_2 + \ldots + A_{mn}x_n &= b_m \end{aligned}$$

- System of equations with
  - n variables and m equations is
- Solution is $\boldsymbol{x}=A^{-1}\boldsymbol{b}$
- In order for A^{-1} to exist $A\boldsymbol{x}=\boldsymbol{b}$ must have *exactly* one solution for every value of $\boldsymbol{b}$
  - It is also possible for the system of equations to have *no solutions* or an *infinite no. of solutions* for some values of $\boldsymbol{b}$
    - It is not possible to have more than one but fewer than infinitely many solutions
  - If $\boldsymbol{x}$ and $\boldsymbol{y}$ are solutions then $\boldsymbol{z}=\alpha\,\boldsymbol{x} + (1-\alpha)\,\boldsymbol{y}$ is a solution for any real α

Span of a set of vectors

- Span of a set of vectors: set of points obtained by a *linear combination* of those vectors
  - A linear combination of vectors $\{\boldsymbol{v}^{(1)},\ldots,\boldsymbol{v}^{(n)}\}$ with coefficients c_i is $\boxed{\sum_i c_i \boldsymbol{v}^{(i)}}$
  - System of equations is $A\boldsymbol{x}=\boldsymbol{b}$
    - A column of A, i.e., $A_{:i}$ specifies travel in direction i
    - How much we need to travel is given by x_i
    - This is a linear combination of vectors $\boxed{A\boldsymbol{x}=\sum_i x_i A_{:,i}}$
  - Thus determining whether $A\boldsymbol{x}=\boldsymbol{b}$ has a solution is equivalent to determining whether $\boldsymbol{b}$ is in the span of columns of A
    - This span is referred to as *column space* or *range* of A

Conditions for a solution to $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$

- Matrix must be square, i.e., $m=n$ and all columns must be *linearly independent*
  - Necessary condition is $n \geq m$
    - For a solution to exist when $\boldsymbol{b} \in \mathbb{R}^m$ we require the column space be all of $\mathbb{R}^m$
  - Sufficient Condition
    - If columns are linear combinations of other columns, column space is less than $\mathbb{R}^m$
      - Columns are linearly dependent or matrix is *singular*
    - For column space to encompass $\mathbb{R}^m$ at least one set of m *linearly independent* columns
- For non-square and singular matrices
  - Methods other than matrix inversion are used

Norms

- Used for measuring the size of a vector
- Norms map vectors to non-negative values
- Norm of vector $\boldsymbol{x}$ is distance from origin to $\boldsymbol{x}$
  - It is any function f that satisfies:

$$f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$$

$$f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) \quad \text{Triangle Inequality}$$

$$\forall \alpha \in \mathbb{R} \quad f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$$

L^P Norm

- Definition $$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$
- L^2 Norm
  - Called Euclidean norm, written simply as $\|\mathrm{x}\|$
  - Squared Euclidean norm is same as $\mathrm{x}^T\mathrm{x}$
- L^1 Norm
  - Useful when 0 and non-zero have to be distinguished (since L^2 increases slowly near origin, e.g., $0.1^2=0.01$)
- L^∞ Norm $$\|\boldsymbol{x}\|_\infty = \max_i |x_i|$$
  - Called max norm

Size of a Matrix

- Frobenius norm

$$\boxed{\|A\|_F = \left(\sum_{i,j} A_{i,j}^2 \right)^{\frac{1}{2}}}$$

- It is analogous to L^2 norm of a vector

Angle between Vectors

- Dot product of two vectors can be written in terms of their L^2 norms and angle θ between them

$$\boxed{\boldsymbol{x}^T \boldsymbol{y} = ||\boldsymbol{x}||_2 ||\boldsymbol{y}||_2 \cos\theta}$$

Special kinds of Matrices

- Diagonal Matrix
  - Mostly zeros, with non-zero entries in diagonal
  - $diag(\boldsymbol{v})$ is a square diagonal matrix with diagonal elements given by entries of vector $\boldsymbol{v}$
  - Multiplying $diag(\boldsymbol{v})$ by vector $\boldsymbol{x}$ only needs to scale each element x_i by v_i

$$\boxed{diag(\boldsymbol{v})\boldsymbol{x} = \boldsymbol{v} \odot \boldsymbol{x}}$$

- Symmetric Matrix
  - Is equal to its transpose: $A=A^T$
  - E.g., a distance matrix is symmetric with $A_{ij}=A_{ji}$

Special Kinds of Vectors

- Unit Vector
  - A vector with unit norm $\boxed{\|\boldsymbol{x}\|_2 = 1}$
- Orthogonal Vectors
  - A vector $\boldsymbol{x}$ and a vector $\boldsymbol{y}$ are orthogonal to each other if $\boldsymbol{x}^T\boldsymbol{y}=0$
  - Vectors are at 90 degrees to each other
- Orthogonal Matrix
  - A square matrix whose rows are mutually orthonormal

$$A^{-1}=A^T$$

Matrix decomposition

- Matrices can be decomposed into factors to learn universal properties about them not discernible from their representation
  - E.g., from decomposition of integer into prime factors $12=2x2x3$ we can discern that
    - 12 is not divisible by 5 or
    - any multiple of 12 is divisible by 3
    - But representations of 12 in binary or decimal are different
- Analogously, a matrix is decomposed into Eigenvalues and Eigenvectors to discern universal properties

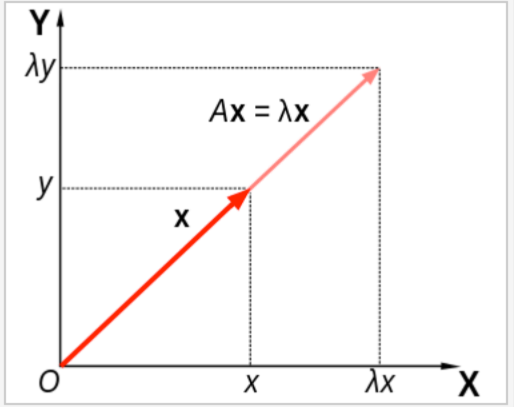
Eigenvector

- An eigenvector of a square matrix A is a non-zero vector v such that multiplication by A only changes the scale of $\boldsymbol{v}$

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

  – The scalar λ is known as eigenvalue

- If v is an eigenvector of A, so is any rescaled vector $s\boldsymbol{v}$. Moreover $s\boldsymbol{v}$ still has the same eigen value. Thus look for a unit eigenvector



Matrix *A* acts by stretching the vector *x*, not changing its direction, so *x* is an eigenvector of *A*.

Wikipedia

Eigenvalue and Characteristic Polynomial

- Consider $Av=w$

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{nn} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \qquad w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

- If v and w are scalar multiples, i.e., if $Av=\lambda v$
  - then v is an eigenvector of the linear transformation A and the scale factor λ is the eigenvalue corresponding to the eigen vector
- This is the *eigenvalue equation* of matrix A
  - Stated equivalently as $(A-\lambda I)v=0$
  - This has a non-zero solution if $|A-\lambda I|=0$ as
    - The polynomial of degree n can be factored as

$$|A-\lambda I| = (\lambda_1-\lambda)(\lambda_2-\lambda)\ldots(\lambda_n-\lambda)$$

    - The $\lambda_1, \lambda_2 \ldots \lambda_n$ are roots of the polynomial and are eigenvalues of A

Example of Eigenvalue/Eigenvector

- Consider the matrix $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
- Taking determinant of $(A\text{-}\lambda I)$, the char poly is

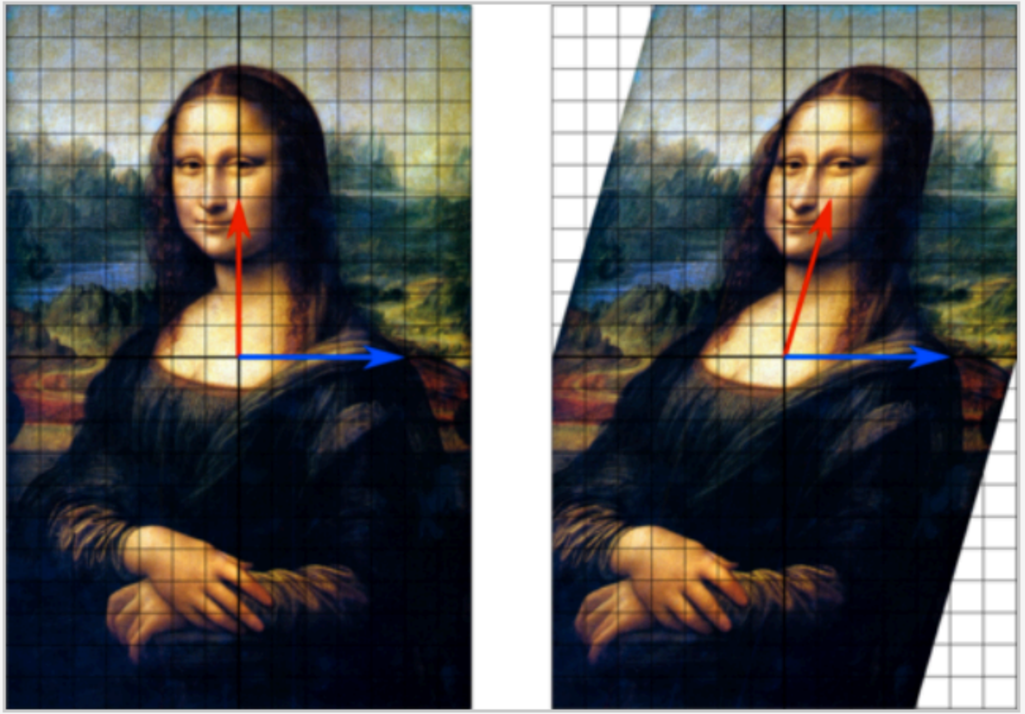
$$\boxed{|A - \lambda I| = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} = 3 - 4\lambda + \lambda^2}$$

- It has roots $\lambda=1$ and $\lambda=3$ which are the two eigenvalues of A
- The eigenvectors are found by solving for $\boldsymbol{v}$ in $A\boldsymbol{v}=\lambda\boldsymbol{v}$, which are

$$\boxed{v_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, v_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}}$$

Example of Eigen Vector

Vectors are grid points



In this shear mapping the red arrow changes direction but the blue arrow does not. The blue arrow is an eigenvector of this shear mapping because it doesn't change direction, and since its length is unchanged, its eigenvalue is 1.

Wikipedia

Eigendecomposition

- Suppose that matrix A has n linearly independent eigenvectors $\{\boldsymbol{v}^{(1)},\ldots,\boldsymbol{v}^{(n)}\}$ with eigenvalues $\{\lambda_1,\ldots,\lambda_n\}$
- Concatenate eigenvectors to form matrix V
- Concatenate eigenvalues to form vector $\lambda=[\lambda_1,\ldots,\lambda_n]$
- Eigendecomposition of A is given by

$$A = V\mathrm{diag}(\lambda)\, V^{-1}$$

Decomposition of Symmetric Matrix

- Every real symmetric matrix A can be decomposed into real-valued eigenvectors and eigenvalues

$$A = Q \Lambda Q^T$$

where Q is an orthogonal matrix composed of eigenvectors of A: $\{\boldsymbol{v}^{(1)},..,\boldsymbol{v}^{(n)}\}$

orthogonal matrix: components are orthogonal or $\boldsymbol{v}^{(i)T}\boldsymbol{v}^{(j)}=0$

Λ is a diagonal matrix of eigenvalues $\{\lambda_1,..,\lambda_n\}$

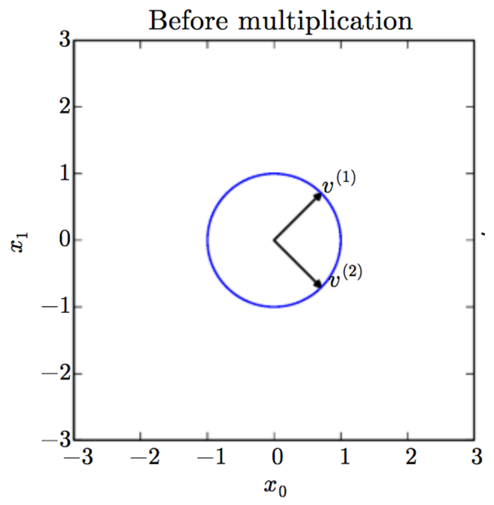
- We can think of A as scaling space by λ_i in direction $v^{(i)}$
  - See figure on next slide

Effect of Eigenvectors and Eigenvalues

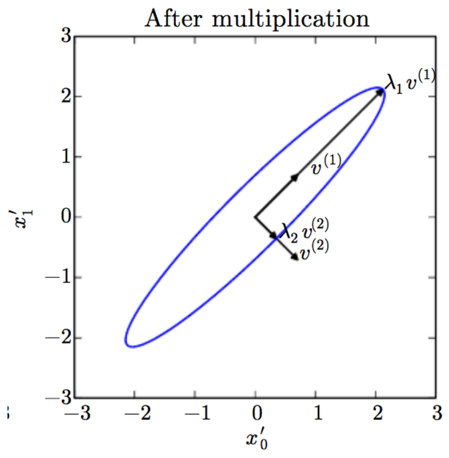
- Example of 2x2 matrix
- Matrix A with two orthonormal eigenvectors
  - $v^{(1)}$ with eigenvalue λ_1, $v^{(2)}$ with eigenvalue λ_2

Plot of unit vectors $\boldsymbol{u} \in \mathbb{R}^2$ (circle)

Plot of vectors $A\boldsymbol{u}$ (ellipse)



with two variables x_1 and x_2



Eigendecomposition is not unique

- Eigendecomposition is $A = Q\Lambda Q^T$
  - where Q is an orthogonal matrix composed of eigenvectors of A
- Decomposition is not unique when two eigenvalues are the same
- By convention order entries of Λ in descending order:
  - Under this convention, eigendecomposition is unique if all eigenvalues are unique

What does eigendecomposition tell us?

- Many useful facts about the matrix obtained

1. Matrix is singular iff any of the eigenvalues are zero
2. Can be used to optimize quadratic expressions of the form

$$f(\boldsymbol{x}) = \boldsymbol{x}^T A \boldsymbol{x} \text{ subject to } ||\boldsymbol{x}||_2 = 1$$

Whenever $\boldsymbol{x}$ is equal to an eigenvector, f is equal to its eigenvalue

Max value of f is max eigen value, min value is min eigen value

Positive Definite Matrix

- A matrix whose eigenvalues are all positive is called *positive definite*
  - Positive or zero is called *positive semidefinite*
- If eigen values are all negative it is *negative definite*
- Positive definite matrices guarantee that $\boldsymbol{x}^T A \boldsymbol{x} \geq \mathbf{0}$

Singular Value Decomposition (SVD)

- Eigendecomposition has form: $A = V\mathrm{diag}(\lambda)\, V^{-1}$
  - If A is not square eigendecomposition undefined
- SVD is a decomposition of the form $A = UDV^T$
- SVD is more general than eigendecomposition
  - Used with any matrix rather than symmetric ones
  - Every real matrix has a SVD (not so of eigen)

SVD Definition

- We write A as the product of three matrices

$$A = UDV^T$$

  - If A is mxn, then U is defined to be mxm, D is mxn and V is nxn
  - D is a diagonal matrix not necessarily square
    - Elements of Diagonal of D are called *singular values*
  - U and V are orthogonal matrices
    - Component vectors of U are called *left singular vectors*
      - Left singular vectors of A are eigenvectors of AA^T
      - Right singular vectors of A are eigenvectors of A^TA

Moore-Penrose Pseudoinverse

- Most useful feature of SVD is that it can be used to generalize matrix inversion to non-square matrices
- Practical algorithms for computing the pseudoinverse of A are based on SVD

$$A^+ = VD^+U^T$$

  - where U,D,V are the SVD of A
    - Psedudoinverse D^+ of D is obtained by taking the reciprocal of its nonzero elements when taking transpose of resulting matrix

Trace of a Matrix

- Trace operator gives the sum of the elements along the diagonal

$$\boxed{Tr(A) = \sum_{i,i} A_{i,i}}$$

- Frobenius norm of a matrix can be represented as

$$\boxed{\|A\|_F = \left(Tr(A)\right)^{\frac{1}{2}}}$$

Determinant of a Matrix

- Determinant of a square matrix $\det(A)$ is a mapping to a scalar
- It is equal to the product of all eigenvalues of the matrix
- Measures how much multiplication by the matrix expands or contracts space

Example: PCA

- A simple ML algorithm is *Principal Components Analysis*
- It can be derived using only knowledge of basic linear algebra

PCA Problem Statement

- Given a collection of m points $\{\boldsymbol{x}^{(1)},...,\boldsymbol{x}^{(m)}\}$ in R^n represent them in a lower dimension.
  - For each point $\boldsymbol{x}^{(i)}$ find a code vector $\boldsymbol{c}^{(i)}$ in R^l
  - If l is smaller than n it will take less memory to store the points
  - This is lossy compression
  - Find encoding function $f(\boldsymbol{x}) = \boldsymbol{c}$ and a decoding function $\boldsymbol{x} \approx g(f(\boldsymbol{x}))$

PCA using Matrix multiplication

- One choice of decoding function is to use matrix multiplication: $g(\boldsymbol{c}) = D\boldsymbol{c}$ where $D \in \mathbb{R}^{n \times l}$
  - D is a matrix with l columns
- To keep encoding easy, we require columns of D to be orthogonal to each other
  - To constrain solutions we require columns of D to have unit norm
- We need to find optimal code $\boldsymbol{c}^*$ given D
- Then we need optimal D

Finding optimal code given D

- To generate optimal code point c^* given input $\boldsymbol{x}$, minimize the distance between input point $\boldsymbol{x}$ and its reconstruction $g(c^*)$

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \left\| \boldsymbol{x} - g(\boldsymbol{c}) \right\|_2$$

  - Using squared L^2 instead of L^2, function being minimized is equivalent to

$$(\boldsymbol{x} - g(\boldsymbol{c}))^T (\boldsymbol{x} - g(\boldsymbol{c}))$$

- Using $g(\boldsymbol{c}) = D\boldsymbol{c}$ optimal code can be shown to be equivalent to $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^T D\boldsymbol{c} + \boldsymbol{c}^T \boldsymbol{c}$

Optimal Encoding for PCA

- Using vector calculus

$$\nabla_c(-2\boldsymbol{x}^T D\boldsymbol{c} + \boldsymbol{c}^T \boldsymbol{c}) = \boldsymbol{0}$$

$$-2D^T\boldsymbol{x} + 2\boldsymbol{c} = \boldsymbol{0}$$

$$c = D^T\boldsymbol{x}$$

- Thus we can encode $\boldsymbol{x}$ using a matrix-vector operation
  - To encode we use $f(\boldsymbol{x}) = D^T\boldsymbol{x}$
  - For PCA reconstruction, since $g(\boldsymbol{c}) = D\boldsymbol{c}$ we use $r(\boldsymbol{x}) = g(f(\boldsymbol{x})) = DD^T\boldsymbol{x}$
  - Next we need to choose the encoding matrix D

Method for finding optimal D

- Revisit idea of minimizing L^2 distance between inputs and reconstructions
  - But cannot consider points in isolation
  - So minimize error over all points: Frobenius norm

$$\boxed{D^* = \arg\min_D \left(\sum_{i,j} \left(x_j^{(i)} - r\left(x^{(i)}\right)_j \right)^2 \right)^{\frac{1}{2}}}$$

    - subject to $D^T D = I_l$
- Use design matrix X, $X \in \mathbb{R}^{m \times n}$
  - Given by stacking all vectors describing the points
- To derive algorithm for finding D^* start by considering the case $l = 1$
  - In this case D is just a single vector $\boldsymbol{d}$

Final Solution to PCA

- For $l = 1$, the optimization problem is solved using eigendecomposition
  - Specifically the optimal $\boldsymbol{d}$ is given by the eigenvector of $X^T X$ corresponding to the largest eigenvalue
- More generally, matrix D is given by the l eigenvectors of X corresponding to the largest eigenvalues (Proof by induction)