

# Linear Algebra for Machine Learning

Sargur N. Srihari

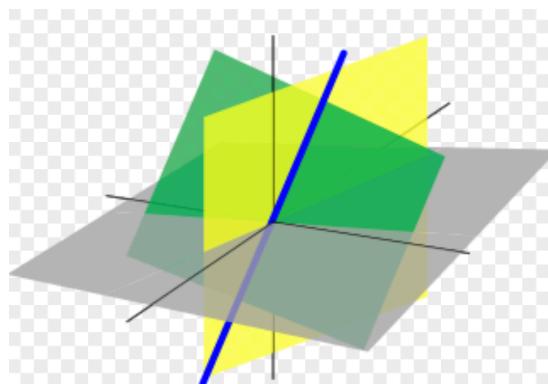
[srihari@cedar.buffalo.edu](mailto:srihari@cedar.buffalo.edu)

# What is linear algebra?

- Linear algebra is the branch of mathematics concerning linear equations such as

$$a_1x_1 + \dots + a_nx_n = b$$

- In vector notation we say  $a^T x = b$
  - Called a linear transformation of  $x$
- Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations



Linear equation  $a_1x_1 + \dots + a_nx_n = b$   
defines a plane in  $(x_1, \dots, x_n)$  space  
Straight lines define common solutions  
to equations

# Why do we need to know it?

- Linear Algebra is used throughout engineering
  - Because it is based on continuous math rather than discrete math
    - Computer scientists have little experience with it
- Essential for understanding ML algorithms
  - E.g., We convert input vectors  $(x_1, \dots, x_n)$  into outputs by a series of linear transformations
- Here we discuss:
  - Concepts of linear algebra needed for ML
  - Omit other aspects of linear algebra

# Linear Algebra Topics

- Scalars, Vectors, Matrices and Tensors
- Multiplying Matrices and Vectors
- Identity and Inverse Matrices
- Linear Dependence and Span
- Norms
- Special kinds of matrices and vectors
- Eigendecomposition
- Singular value decomposition
- The Moore Penrose pseudoinverse
- The trace operator
- The determinant
- Ex: principal components analysis

# Scalar

- Single number
  - In contrast to other objects in linear algebra, which are usually arrays of numbers
- Represented in lower-case italic  $x$ 
  - They can be real-valued or be integers
    - 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
- Written in lower-case bold such as  $\mathbf{x}$ 
  - its elements are in italics lower case, subscripted

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

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

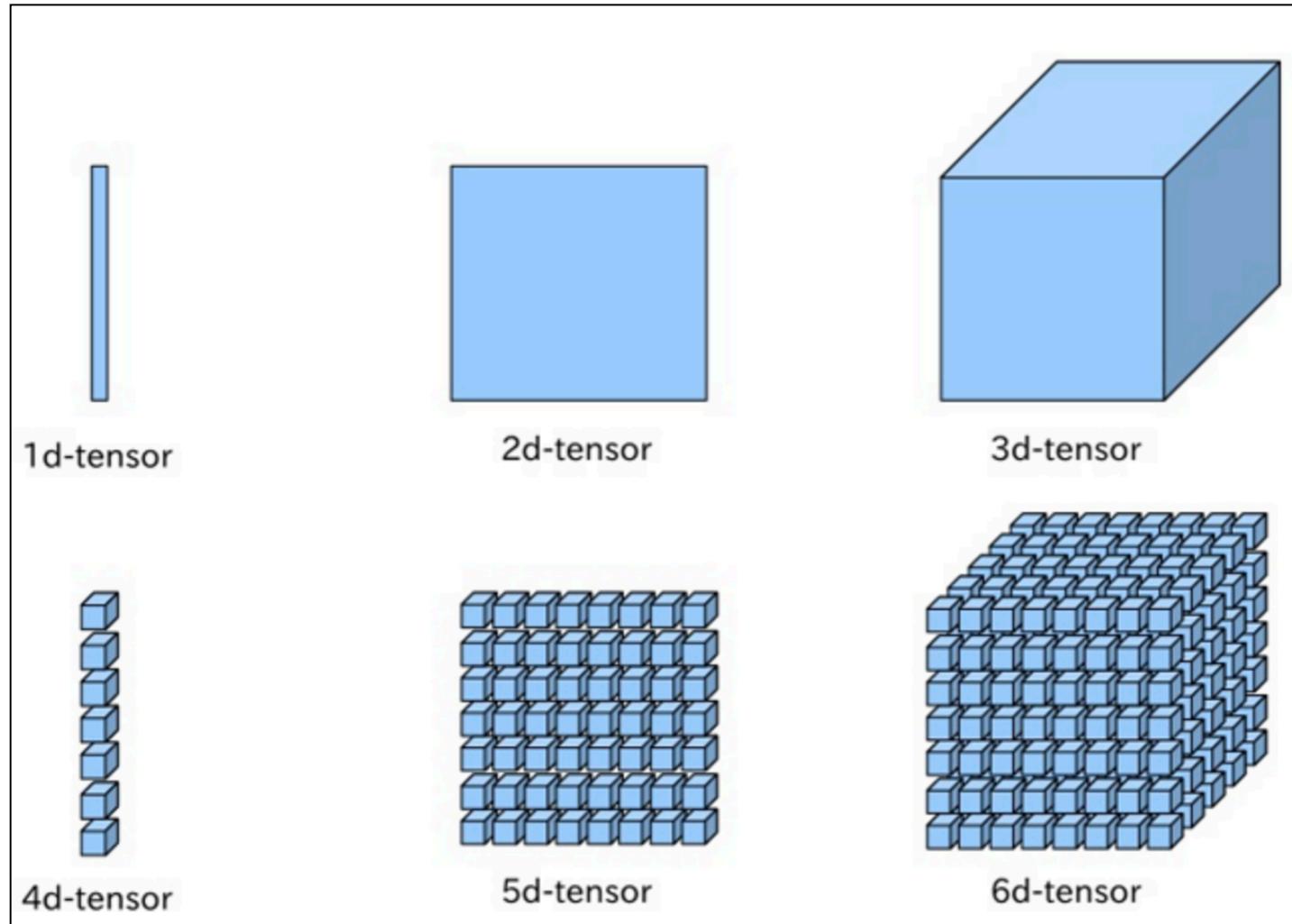
# Matrices

- 2-D array of numbers
  - So each element identified by two indices
- Denoted by bold typeface  $A$ 
  - Elements indicated by name in italic but not bold
    - $A_{1,1}$  is the top left entry and  $A_{m,n}$  is the bottom right entry
    - We can identify nos in vertical column  $j$  by writing  $:$  for the horizontal coordinate
    - E.g., 
$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$
    - $A_{i,:}$  is  $i^{\text{th}}$  row of  $A$ ,  $A_{:,j}$  is  $j^{\text{th}}$  column of  $A$
  - If  $A$  has shape of height  $m$  and width  $n$  with real-values then  $A \in \mathbb{R}^{m \times n}$

# Tensor

- Sometimes need an array with more than two axes
  - E.g., an RGB color image has three axes
- A tensor is an array of numbers arranged on a regular grid with variable number of axes
  - See figure next
- Denote a tensor with this bold typeface:  $\mathbf{A}$
- Element  $(i,j,k)$  of tensor denoted by  $\mathbf{A}_{i,j,k}$

# Shapes of Tensors



# Transpose of a Matrix

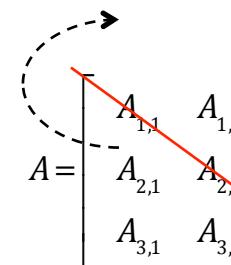
- An important operation on matrices
- The transpose of a matrix  $A$  is denoted as  $A^T$
- Defined as

$$(A^T)_{i,j} = A_{j,i}$$

– The mirror image across a diagonal line

- Called the main diagonal , running down to the right starting from upper left corner

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



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

# Vectors as special case of matrix

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

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

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

- A scalar is a matrix with one element

$$a = a^T$$

# Matrix Addition

- We can add matrices to each other if they have the same shape, by adding corresponding elements
  - 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
- Less conventional notation used in ML:
  - Vector added to matrix  $C = A + b \Rightarrow C_{i,j} = A_{i,j} + b_j$ 
    - Called broadcasting since vector  $b$  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  $m \times n$  and  $B$  is of shape  $n \times p$  then *matrix product*  $C$  is of shape  $m \times p$

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

- Note that the standard product of two matrices is not just the product of two individual elements
  - Such a product does exist and is called the element-wise product or the Hadamard product  $A \odot B$

# Multiplying Vectors

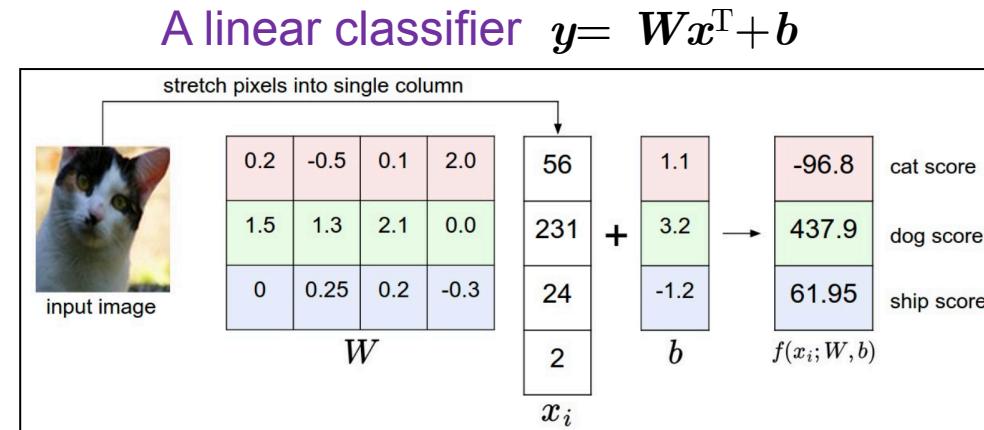
- Dot product between two vectors  $x$  and  $y$  of same dimensionality is the matrix product  $x^T y$
- We can think of matrix product  $C=AB$  as computing  $C_{ij}$  the dot product of row  $i$  of  $A$  and column  $j$  of  $B$

# Matrix Product Properties

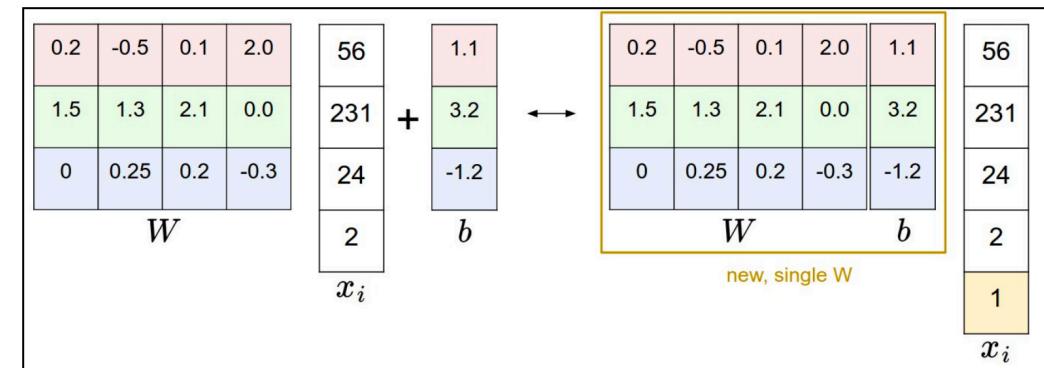
- Distributivity over addition:  $A(B+C)=AB+AC$
- Associativity:  $A(BC)=(AB)C$
- Not commutative:  $AB=BA$  is not always true
- Dot product between vectors is commutative:  
 $x^T y = y^T x$
- Transpose of a matrix product has a simple form:  $(AB)^T=B^T A^T$

# Example flow of tensors in ML

Vector  $x$  is converted into vector  $y$  by multiplying  $x$  by a matrix  $W$



A linear classifier with bias eliminated  $y = Wx^T$



# Linear Transformation

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

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n &= b_2 \\ &\vdots \\ A_{n1}x_1 + A_{n2}x_2 + \dots + A_{nn}x_n &= b_n \end{aligned}$$

$n$  equations in  
 $n$  unknowns

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

$n \times n$        $n \times 1$        $n \times 1$

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

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

# Identity and Inverse Matrices

- Matrix inversion is a powerful tool to analytically solve  $Ax=b$
- 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  $Ax = b$  as follows:

$$Ax = b$$

$$A^{-1}Ax = A^{-1}b$$

$$I_n x = A^{-1}b$$

$$x = A^{-1}b$$

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

# Solving Simultaneous equations

- $Ax = b$

where  $A$  is  $(M+1) \times (M+1)$

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

$b$  is  $N \times 1$

# Example: System of Linear Equations in Linear Regression

- Instead of  $Ax=b$
- We have  $\Phi w = t$ 
  - where  $\Phi$  is  $m \times n$  design matrix of  $m$  features for  $n$  samples  $x_j$ ,  $j=1,..n$
  - $w$  is weight vector of  $m$  values
  - $t$  is target values of sample,  $t=[t_1,..t_n]$
  - We need weight  $w$  to be used with  $m$  features to determine output

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

# Closed-form solutions

- Two closed-form solutions
  1. Matrix inversion  $x = A^{-1}b$
  2. Gaussian elimination

# Linear Equations: Closed-Form Solutions

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

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

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

2. Gaussian Elimination  
 followed by back-substitution

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

$$\begin{array}{c} L_2 - 3L_1 \rightarrow L_2 \quad L_3 - 2L_1 \rightarrow L_3 \quad -L_2/4 \rightarrow L_2 \\ \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 3 & 5 & 6 & 7 \\ 2 & 4 & 3 & 8 \end{array} \right] \sim \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 2 & 4 & 3 & 8 \end{array} \right] \sim \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 0 & -2 & 7 & -2 \end{array} \right] \sim \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & -2 & 7 & -2 \end{array} \right] \\ \sim \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 1 & 2 \end{array} \right] \sim \left[ \begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array} \right] \sim \left[ \begin{array}{ccc|c} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array} \right] \end{array}$$

# Disadvantage of closed-form solutions

- If  $A^{-1}$  exists, the same  $A^{-1}$  can be used for any given  $b$ 
  - 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  $n \times n$  matrix
- Software solutions use value of  $b$  in finding  $x$ 
  - E.g., difference (derivative) between  $b$  and output is used iteratively

# How many solutions for $Ax=b$ exist?

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

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

# 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  $\{v^{(1)}, \dots, v^{(n)}\}$  with coefficients  $c_i$  is  $\sum_i c_i v^{(i)}$
  - System of equations is  $Ax=b$ 
    - 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
  - Thus determining whether  $Ax=b$  has a solution is equivalent to determining whether  $b$  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 $Ax=b$

- 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  $A \in \mathbb{R}^{m \times n}$  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

# Use of a Vector in Regression

- A design matrix
  - N samples, D features



- Feature vector has three dimensions
- This is a regression problem

# Norms

- Used for measuring the size of a vector
- Norms map vectors to non-negative values
- Norm of vector  $x = [x_1, \dots, x_n]^T$  is distance from origin to  $x$ 
  - It is any function  $f$  that satisfies:

$$f(x) = 0 \Rightarrow x = 0$$

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

$$\forall \alpha \in R \quad f(\alpha x) = |\alpha| f(x)$$

# $L^P$ Norm

- Definition:

$$\|x\|_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- $L^2$  Norm

- Called Euclidean norm

- Simply the Euclidean distance between the origin and the point  $x$
- written simply as  $\|x\|$
- Squared Euclidean norm is same as  $x^T x$

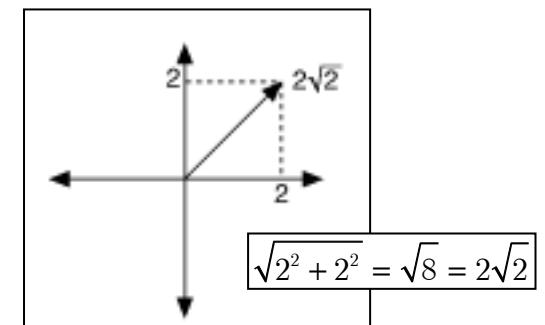
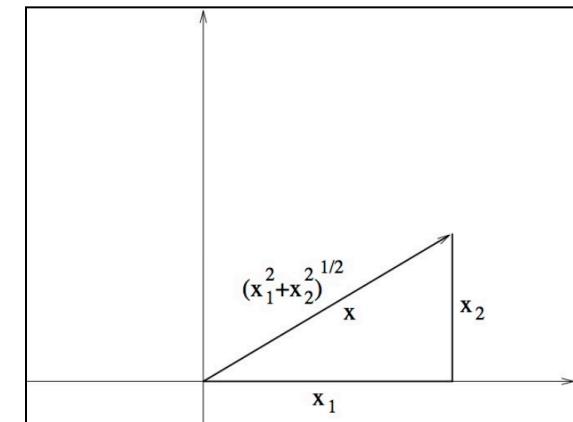
- $L^1$  Norm

- Useful when 0 and non-zero have to be distinguished
  - Note that  $L^2$  increases slowly near origin, e.g.,  $0.1^2=0.01$ )

- $L^\infty$  Norm

$$\|x\|_\infty = \max_i |x_i|$$

- Called max norm

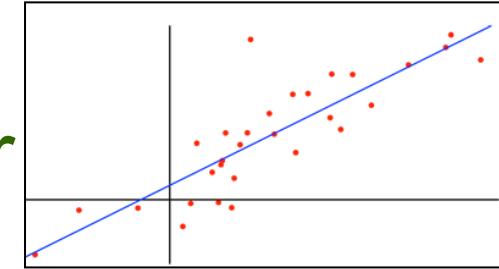


# Use of norm in Regression

- Linear Regression

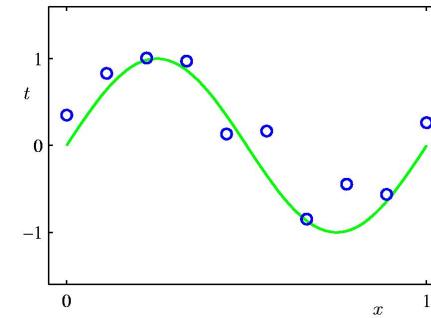
$\mathbf{x}$ : a vector,  $\mathbf{w}$ : weight vector

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_d x_d = \mathbf{w}^T \mathbf{x}$$



With nonlinear basis functions  $\phi_j$

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$



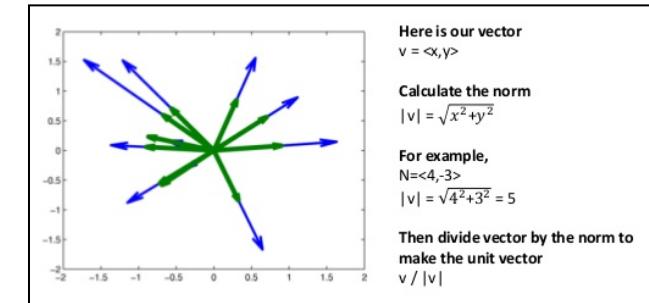
- Loss Function

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Second term is a weighted norm  
called a regularizer (to prevent overfitting)

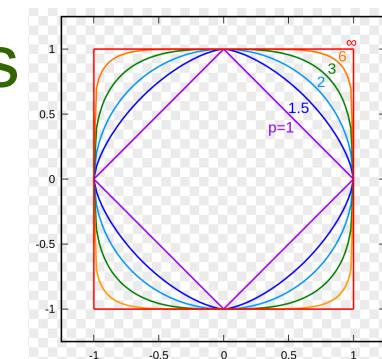
# $L^P$ Norm and Distance

- Norm is the length of a vector



- We can use it to draw a unit circle from origin

- Different  $P$  values yield different shapes
  - Euclidean norm yields a circle



- Distance between two vectors ( $v, w$ )

- $$\text{dist}(v, w) = \|v - w\|$$

$$= \sqrt{(v_1 - w_1)^2 + \dots + (v_n - w_n)^2}$$

Distance to origin would just be sqrt of sum of squares

# Size of a Matrix: Frobenius Norm

- Similar to  $L^2$  norm

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

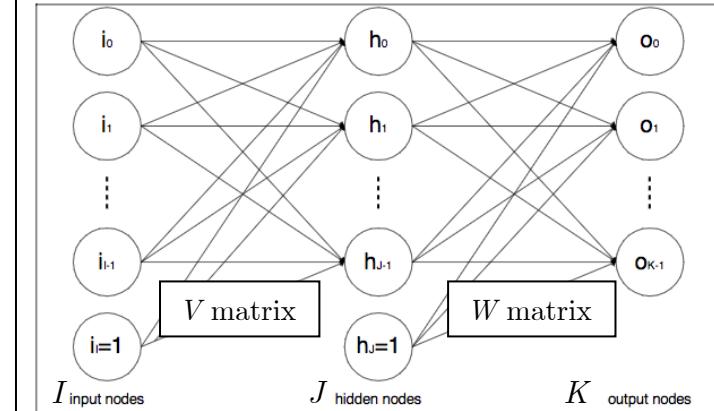
$$A = \begin{bmatrix} 2 & -1 & 5 \\ 0 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix} \quad \|A\| = \sqrt{4+1+25+\dots+1} = \sqrt{46}$$

- Frobenius in ML

- Layers of neural network involve matrix multiplication
- Regularization:

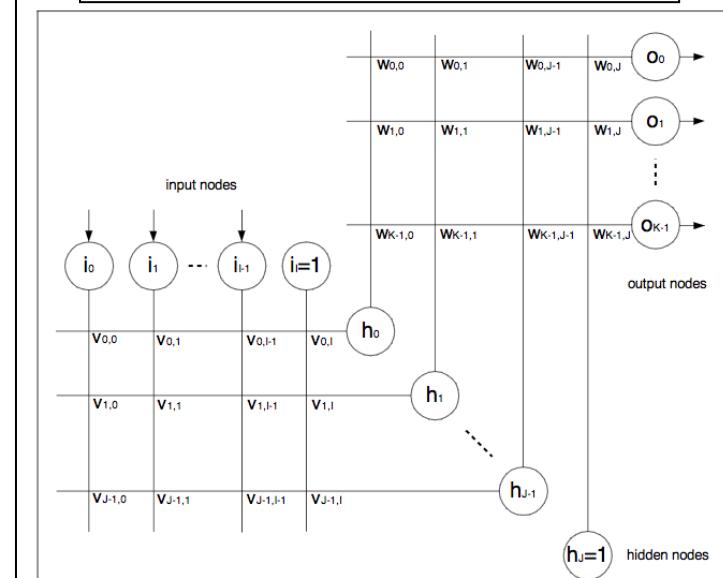
- minimize Frobenius of weight matrices  $\|W(i)\|$  over  $L$  layers

$$J_R = J + \lambda \sum_{i=1}^L \|W^{(i)}\|_F$$



$$I_{1 \times (I+1)} \times V_{(I+1) \times J} = \text{net}_J$$

$$h_j = f(\text{net}_j) \quad f(x) = 1/(1+e^{-x})$$



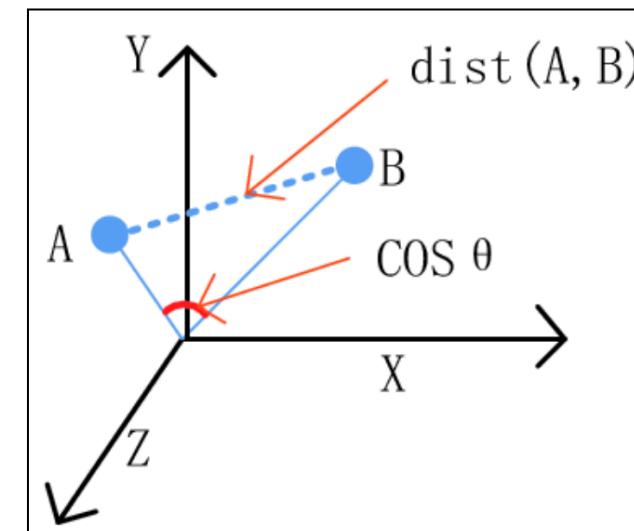
# Angle between Vectors

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

$$\mathbf{x}^T \mathbf{y} \Rightarrow \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos \theta$$

- Cosine between two vectors is a measure of their similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$



# Special kind of Matrix: Diagonal

- Diagonal Matrix has mostly zeros, with non-zero entries only in diagonal
  - E.g., identity matrix. where all diagonal entries are 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- E.g., covariance matrix with independent features

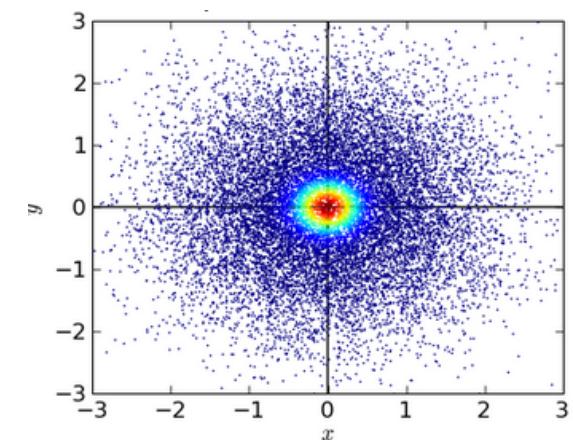
$$Cov(X, Y) = \sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$$

$$\text{Covariance} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

$$\text{Covariance} = \frac{-64.57}{8}$$

$$\text{Covariance} = -8.07$$

$$\begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_K^2 \end{bmatrix}$$



If  $\text{Cov}(X, Y)=0$  then  $E(XY)=E(X)E(Y)$

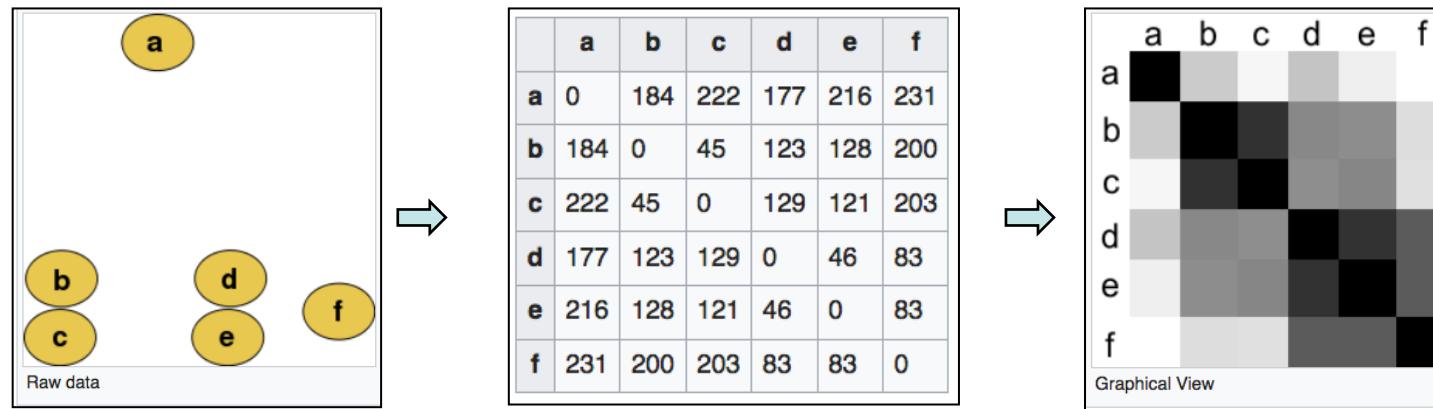
$$N(\mathbf{x} | \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

# Efficiency of Diagonal Matrix

- $\text{diag}(\mathbf{v})$  denotes a square diagonal matrix with diagonal elements given by entries of vector  $\mathbf{v}$
- Multiplying vector  $\mathbf{x}$  by a diagonal matrix is efficient
  - To compute  $\text{diag}(\mathbf{v})\mathbf{x}$  we only need to scale each  $x_i$  by  $v_i$
- Inverting a square diagonal matrix is efficient
  - Inverse exists iff every diagonal entry is nonzero, in which case  $\text{diag}(\mathbf{v})^{-1} = \text{diag}([1/v_1, \dots, 1/v_n]^T)$

# Special kind of Matrix: Symmetric

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



- E.g., covariance matrices are symmetric

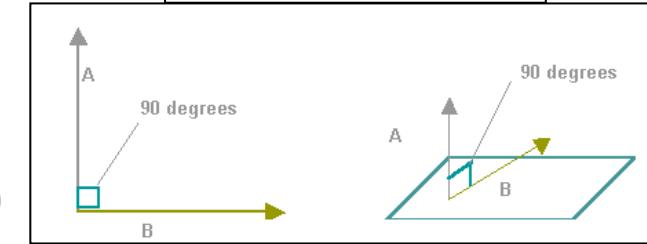
$$\Sigma = \begin{pmatrix} 1 & .5 & .15 & .15 & 0 & 0 \\ .5 & 1 & .15 & .15 & 0 & 0 \\ .15 & .15 & 1 & .25 & 0 & 0 \\ .15 & .15 & .25 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & .10 \\ 0 & 0 & 0 & 0 & .10 & 1 \end{pmatrix}$$

# Special Kinds of Vectors

- Unit Vector
  - A vector with unit norm  $\|x\|_2 = 1$
- Orthogonal Vectors
  - A vector  $x$  and a vector  $y$  are orthogonal to each other if  $x^T y = 0$ 
    - If vectors have nonzero norm, vectors at 90 degrees to each other
  - Orthonormal Vectors
    - Vectors are orthogonal & have unit norm
    - Orthogonal Matrix
      - A square matrix whose rows are mutually orthonormal:  $A^T A = A A^T = I$
      - $A^{-1} = A^T$

$$\|x\|_2 = 1$$

$$\begin{bmatrix} 2 \\ -3 \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ -2 \\ 7 \end{bmatrix}$$



Orthogonal matrices are of interest because their inverse is very cheap to compute

# Matrix decomposition

- Matrices can be decomposed into factors to learn universal properties, just like integers:
  - Properties not discernible from their representation

## 1. Decomposition of integer into prime factors

- From  $12=2 \times 2 \times 3$  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

## 2. Decomposition of Matrix $A$ as $A = V \text{diag}(\lambda) V^{-1}$

- where  $V$  is formed of eigenvectors and  $\lambda$  are eigenvalues, e.g,

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

has eigenvalues  $\lambda=1$  and  $\lambda=3$  and eigenvectors  $V$ :

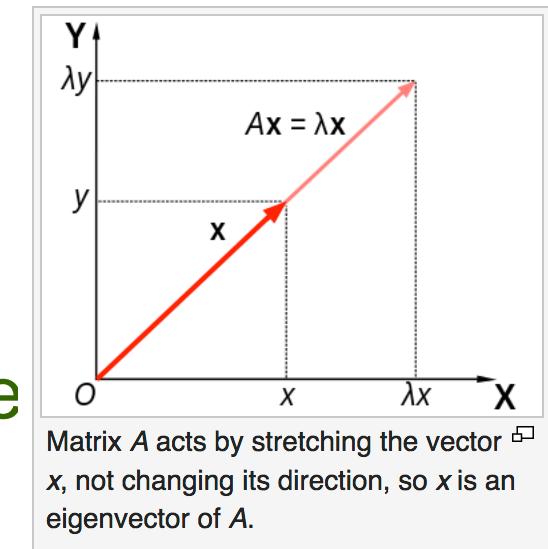
$$v_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, v_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

# Eigenvector

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

$$Av = \lambda v$$

- The scalar  $\lambda$  is known as eigenvalue
- If  $v$  is an eigenvector of  $A$ , so is any rescaled vector  $sv$ . Moreover  $sv$  still has the same eigen value. Thus look for a unit eigenvector



[Wikipedia](#)

# Eigenvalue and Characteristic Polynomial

- Consider  $A\mathbf{v}=\mathbf{w}$

$$A = \begin{bmatrix} A_{1,1} & L & A_{1,n} \\ M & M & M \\ A_{n,1} & L & A_{nn} \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ M \\ v_n \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ M \\ w_n \end{bmatrix}$$

- If  $\mathbf{v}$  and  $\mathbf{w}$  are scalar multiples, i.e., if  $A\mathbf{v}=\lambda\mathbf{v}$

- then  $\mathbf{v}$  is an eigenvector of the linear transformation  $A$  and the scale factor  $\lambda$  is the eigenvalue corresponding to the eigen vector

- This is the *eigenvalue equation* of matrix  $A$

- Stated equivalently as  $(A-\lambda I)\mathbf{v}=0$
- This has a non-zero solution if  $|A-\lambda I|=0$  as
  - The polynomial of degree  $n$  can be factored as

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

- The  $\lambda_1, \lambda_2\dots\lambda_n$  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  $(A - \lambda I)$ , the char poly is

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

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

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

# Eigendecomposition

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

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

# 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  $A$ :  $\{\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(n)}\}$

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

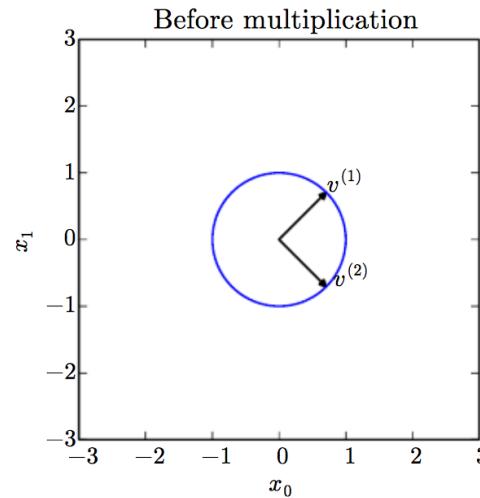
$\Lambda$  is a diagonal matrix of eigenvalues  $\{\lambda_1, \dots, \lambda_n\}$

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

# Effect of Eigenvectors and Eigenvalues

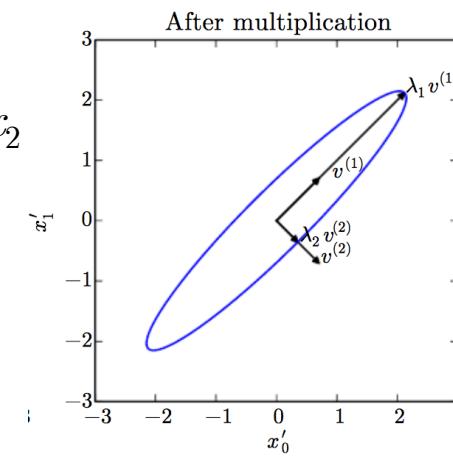
- Example of  $2 \times 2$  matrix
- Matrix  $A$  with two orthonormal eigenvectors
  - $v^{(1)}$  with eigenvalue  $\lambda_1$ ,  $v^{(2)}$  with eigenvalue  $\lambda_2$

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



with two variables  $x_1$  and  $x_2$

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



# 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  $\Lambda$  in descending order:
  - Under this convention, eigendecomposition is unique if all eigenvalues are unique

# What does eigendecomposition tell us?

- Tells us useful facts about the matrix:
  1. Matrix is *singular* if & only if any eigenvalue is zero
  2. Useful to optimize quadratic expressions of form

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

Whenever  $\mathbf{x}$  is equal to an eigenvector,  $f$  is equal to the corresponding eigenvalue

Maximum value of  $f$  is max eigen value, minimum value is min eigen value

Example of such a quadratic form appears in multivariate Gaussian

$$N(\mathbf{x} | \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

# 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  $x^T A x \geq 0$

# Singular Value Decomposition (SVD)

- Eigendecomposition has form:  $A = V \text{diag}(\lambda) V^{-1}$ 
  - If  $A$  is not square, eigendecomposition is 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
    - Same is not true of eigen decomposition

# SVD Definition

- Write  $A$  as a product of 3 matrices:  $A=UDV^T$ 
  - If  $A$  is  $m \times n$ , then  $U$  is  $m \times m$ ,  $D$  is  $m \times n$ ,  $V$  is  $n \times n$
- Each of these matrices have a special structure
  - $U$  and  $V$  are orthogonal matrices
  - $D$  is a diagonal matrix not necessarily square
    - Elements of Diagonal of  $D$  are called *singular values of  $A$*
    - Columns of  $U$  are called *left singular vectors*
    - Columns of  $V$  are called *right singular vectors*
- SVD interpreted in terms of *eigendecomposition*
  - Left singular vectors of  $A$  are eigenvectors of  $AA^T$
  - Right singular vectors of  $A$  are eigenvectors of  $A^TA$
  - Nonzero singular values of  $A$  are square roots of eigen values of  $A^TA$ . Same is true of  $AA^T$

# Use of SVD in ML

1. SVD is used in generalizing matrix inversion
  - Moore-Penrose inverse (discussed next)
2. Used in Recommendation systems
  - Collaborative filtering (CF)
    - Method to predict a rating for a *user-item* pair based on the history of ratings given by the user and given to the item
    - Most CF algorithms are based on *user-item* rating matrix where each row represents a user, each column an item
      - Entries of this matrix are ratings given by users to items
    - SVD reduces no.of features of a data set by reducing space dimensions from  $N$  to  $K$  where  $K < N$

# SVD in Collaborative Filtering

$$\hat{X} \approx U S V^T$$

$$\left( \begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{array} \right)_{m \times n} \approx \left( \begin{array}{ccc} u_{11} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{array} \right)_{m \times r} \left( \begin{array}{ccc} s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{array} \right)_{r \times r} \left( \begin{array}{ccc} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{array} \right)_{r \times n}$$

- $X$  is the utility matrix
  - $X_{ij}$  denotes how user  $i$  likes item  $j$
  - CF fills blank (cell) in utility matrix that has no entry
- Scalability and sparsity is handled using SVD
  - SVD decreases dimension of utility matrix by extracting its latent factors
    - Map each user and item into latent space of dimension  $r$

# 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

$$A^+ = V D^+ U^T$$

– where  $U, D, V$  are the SVD of  $A$

- Pseudoinverse  $D^+$  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

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

- Frobenius norm of a matrix can be represented as

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

# 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  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  in  $R^n$  represent them in a lower dimension.
  - For each point  $\mathbf{x}^{(i)}$  find a code vector  $\mathbf{c}^{(i)}$  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  $f(\mathbf{x}) = \mathbf{c}$  and a decoding function  $\mathbf{x} \approx g(f(\mathbf{x}))$

# PCA using Matrix multiplication

- One choice of decoding function is to use matrix multiplication:  $g(\mathbf{c}) = D\mathbf{c}$  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  $\mathbf{c}^*$  given  $D$
- Then we need optimal  $D$

# Finding optimal code given $D$

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

$$c^* = \arg \min_c \|x - g(c)\|_2$$

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

$$(x - g(c))^T (x - g(c))$$

- Using  $g(c) = Dc$  optimal code can be shown to be equivalent to

$$c^* = \arg \min_c -2x^T Dc + c^T c$$

# Optimal Encoding for PCA

- Using vector calculus

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

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

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

- Thus we can encode  $\mathbf{x}$  using a matrix-vector operation

- To encode we use  $f(\mathbf{x}) = D^T \mathbf{x}$

- For PCA reconstruction, since  $g(\mathbf{c}) = D\mathbf{c}$  we use  $r(\mathbf{x}) = g(f(\mathbf{x})) = DD^T \mathbf{x}$

- 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
 

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

    - 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  $d$

# Final Solution to PCA

- For  $l = 1$ , the optimization problem is solved using eigendecomposition
  - Specifically the optimal  $d$  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)