CE 395 Special Topics in Machine Learning

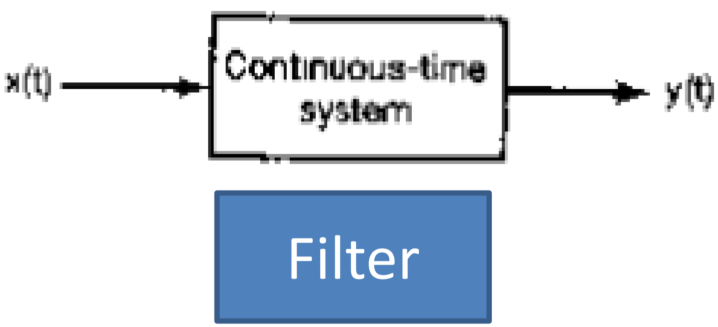
Assoc. Prof. Dr. Yuriy Mishchenko

Fall 2017

FOURIER TRANSFORM (CONTINUED)

LTI filters & FT

LTI systems/filters ⇔ **convolution**, and **convolution** ⇔ **multiplication** in **Fourier Space**, thus LTI filter in Fourier space is simply a multiplication with the Fourier image of the response h(t)



$$\boxed{y(\omega) = h(\omega) \cdot x(\omega)}$$

LTI filters & FT

The Fourier image of the response function $h(t)$, $h^{\sim}(\omega)$, is called **transfer function**, **frequency response,** or **frequency characteristic** of the filter

LTI filters & FT

IMPORTANT:

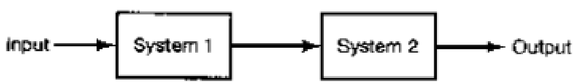
h~(ω) can be also viewed as (**<u>multiplicative</u>**) action of LTI filter on a single frequency sinusoidal input $x(t)=e^{i\omega_0 t}$

$$\boxed{F\left\lfloor x(t)=e^{i\omega_0 t}\right\rfloor \Rightarrow y(t)=h(\omega_0)\cdot e^{i\omega_0 t}}$$

SHOW THIS ACTION FROM THE DEFINITION OF CONVOLUTION AND LTI FILTER

LTI filters & FT

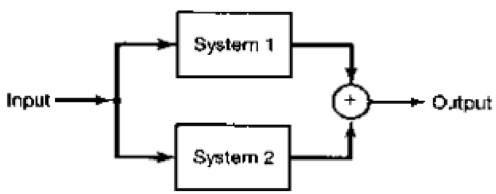
Further examples: show these properties of system interconnections in Fourier space:



(a)

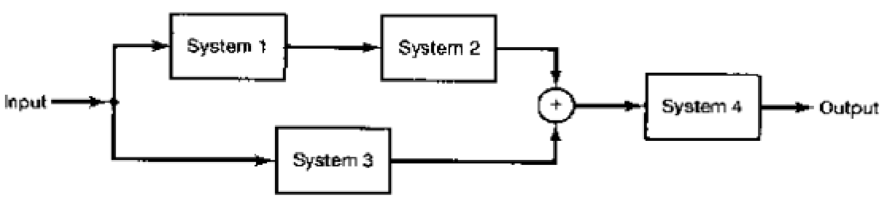
$$\tilde{h}_{2*1}(\omega)=\tilde{h}_2(\omega)\cdot\tilde{h}_1(\omega)\Rightarrow\Rightarrow$$

$$h_{2*1}[\tilde{x}(\omega)]=\left(\tilde{h}_2(\omega)\cdot\tilde{h}_1(\omega)\right)\cdot\tilde{x}(\omega)$$



(b)

$$\tilde{h}_{1+2}[x(\omega)]\Rightarrow(\tilde{h}_1(\omega)+\tilde{h}_2(\omega))\cdot\tilde{x}(\omega)$$



(c)

$$\tilde{H}[\tilde{x}(\omega)]=\tilde{h}_4(\omega)\cdot(\tilde{h}_2(\omega)\cdot\tilde{h}_1(\omega)+\tilde{h}_3(\omega))\cdot\tilde{x}(\omega)$$

LTI filters & FT

For general digital linear filters can find:

$$y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b)$$
$$-a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a).$$

$Z = e^{i\omega}$ **Discrete-Shift operator**

For single-frequency input x(t)=exp(iωt)

$$\left(1 + a(2)Z^{-1} + \ldots + a(n_a+1)Z^{-n_a}\right)y(t) = \left(b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b}\right)x(t)$$

$$\tilde{h}(\omega) = \frac{b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b}}{1 + a(2)Z^{-1} + \ldots + a(n_a+1)Z^{-n_a}} = \frac{P(Z^{-1})}{Q(Z^{-1})}$$

$\leftarrow$IIR filter

LTI filters & FT

Similarly for FIR filters:

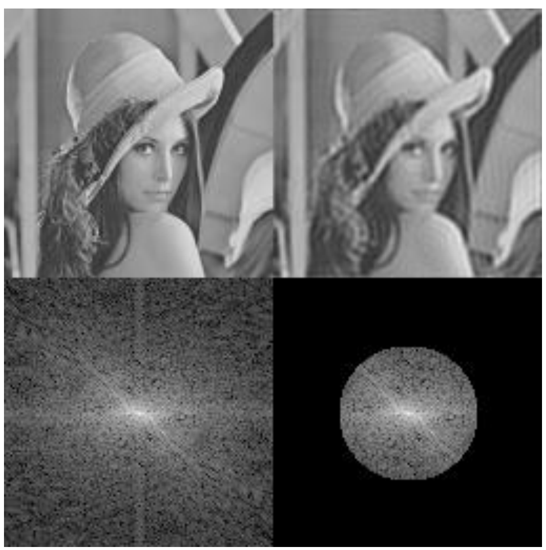
$$y(t) = \left(b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b}\right)x(t)$$

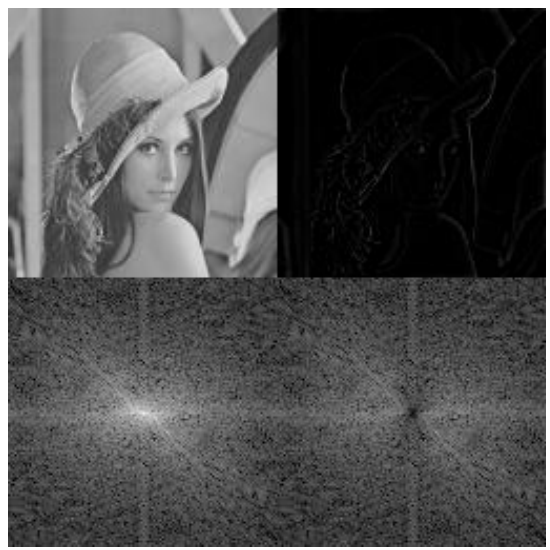
$$\tilde{h}(\omega) = b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b} = P(Z^{-1})$$

<u>Characteristic h~(ω) is a polynomial in Z^{-1}</u>

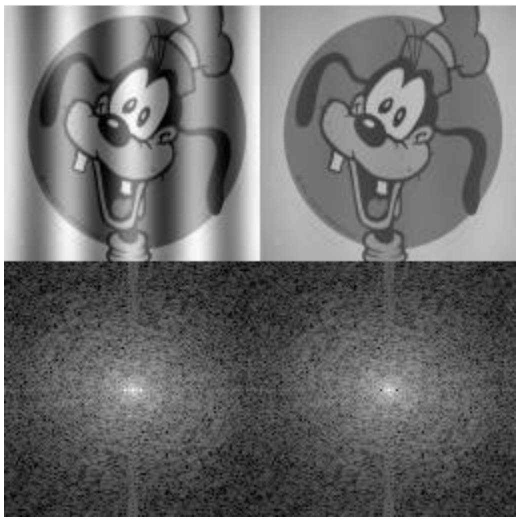
Second look at denoising and sharpening

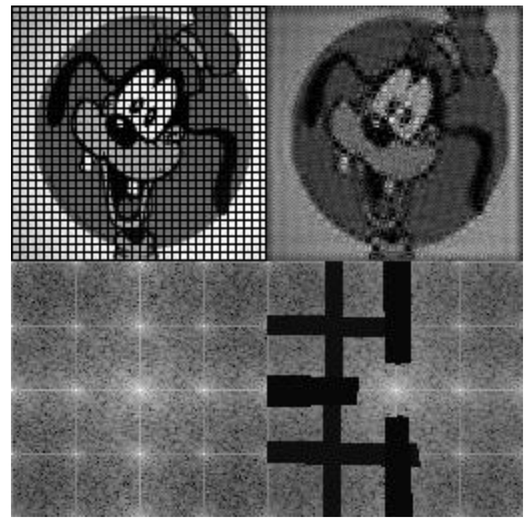
Action of LTI filter in Fourier space is multiplication of $\tilde{x}(\omega)$ with $\tilde{h}(\omega)$, $\tilde{x}(\omega)\cdot\tilde{h}(\omega)$





Filtering out specific frequencies

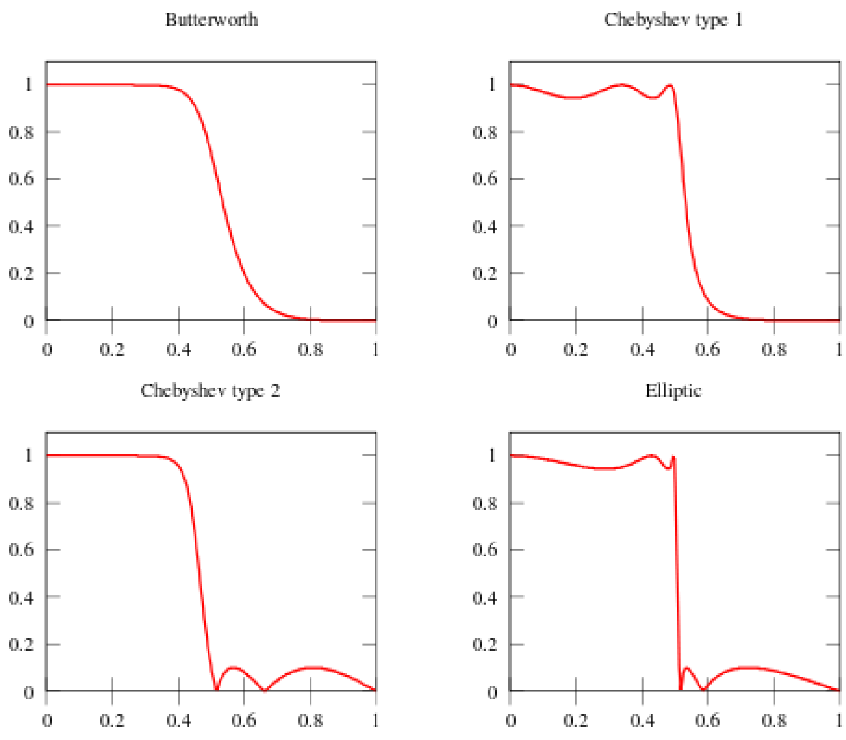




Low/high pass filters

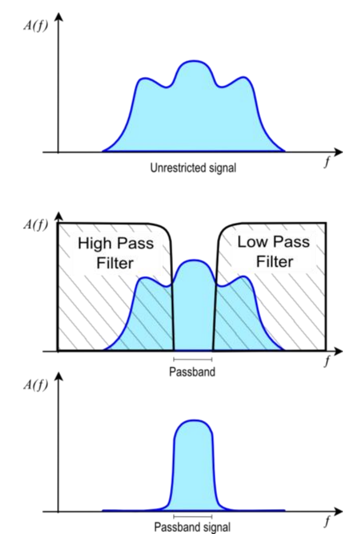
Low/high pass filters are characterized by near-zero transfer function at high/low frequencies ω

$$h(\omega)$$



Action:

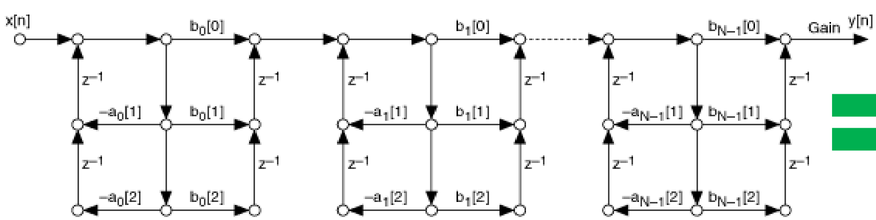
$$y(\omega) = h(\omega) \cdot x(\omega)$$



SOS filters, revisit

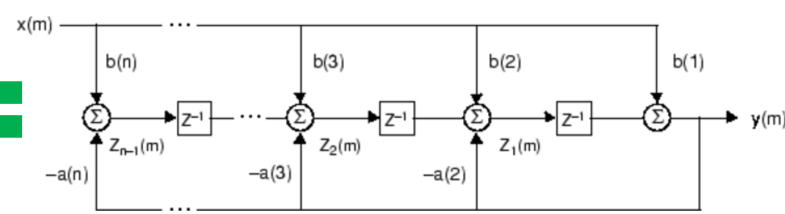
$$\tilde{h}(\omega) = \overbrace{\prod_k \frac{\beta_1^{(k)} + \beta_2^{(k)} Z^{-1} + \beta_3^{(k)} Z^{-2}}{\alpha_1^{(k)} + \alpha_2^{(k)} Z^{-1} + \alpha_3^{(k)} Z^{-2}}}^{\text{SOS}} = \frac{P(Z^{-1})}{Q(Z^{-1})} = \frac{b(1) + b(2)Z^{-1} + \ldots + b(n_b+1)Z^{-n_b}}{1 + a(2)Z^{-1} + \ldots + a(n_a+1)Z^{-n_a}}$$

(By Fundamental Theorem of Algebra)



SOS

=



Direct form 1

MATRICES AND VECTORS

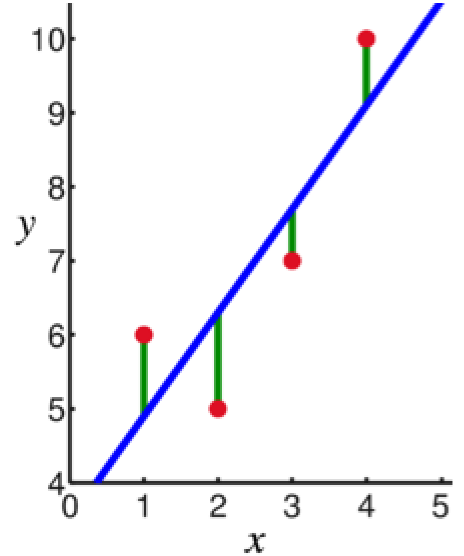
Why matrices and vectors?

- ML learning algorithms are typically applied to problems with very large numbers of inputs, x as well as parameters θ. Computationally, these are represented by ***numerical vectors*** $[x_1,x_2,\ldots,x_n]$
- Operations on such vectors is a large part of ML data manipulations
- Optimization of ML algorithms also is always carried out over ***vector spaces*** – *the concepts of linear algebra are important here !*

Example – linear regression

$$y_i = \theta_1 \cdot \phi_1(x_i) + \theta_2 \phi_2(x_i) + \ldots + \theta_p \phi_p(x_i)$$

Represent the output variable y as a linear combination of (nonlinear) features ϕ of inputs *x*



Example – linear regression

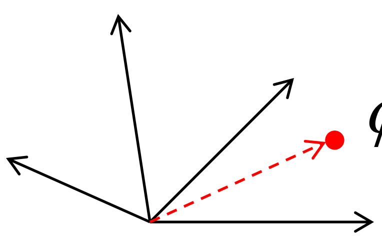
Prediction – situations are described by numerical (feature) vectors

- $\phi \rightarrow y$

Optimization – best-fit model is described by numerical (coefficient) vector

- $\{\text{data}\} \rightarrow \theta$

Feature space $\boldsymbol{\Phi}$

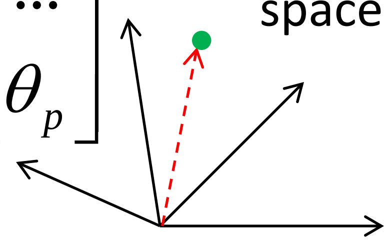


$$\phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_p \end{bmatrix}$$

$$\mathrm{y} = \phi^{\mathrm{T}} \cdot \theta$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_p \end{bmatrix} = \theta$$

Model parameter space $\boldsymbol{\Theta}$



Vectors

Numerical vector is a 1D array of numbers

$$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

column vector

$$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}_{n\times 1}$$

row vector

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}_{1\times n}$$

Vectors

Examples of vectors:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \qquad \begin{bmatrix} 3.1 \\ 2.7 \\ 5.5 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} +1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} -5 & 2.77 & 0 & 33 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 5 \end{bmatrix}$$

Matrices

Matrix is a numerical table or 2D array

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & & \\ \dots & & \dots & \\ a_{n1} & & & a_{nm} \end{bmatrix} \Rightarrow \left[a_{ij}\right]_{n\times m}$$

Matrices

Examples of matrices:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & -5 & 1 \\ -1 & 2 & 7 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1.15 & -2.2 & 3 \\ 3.50 & 0 & -7 \\ 1.33 & 2.3 & 2.5 \end{bmatrix}$$

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Some special matrices

- Square matrix
- Identity matrix
- Diagonal matrix
- Rectangular matrix
- Banded-diagonal matrix
- Symmetric matrix
- Asymmetric matrix
- Upper/Lower-triangular matrix

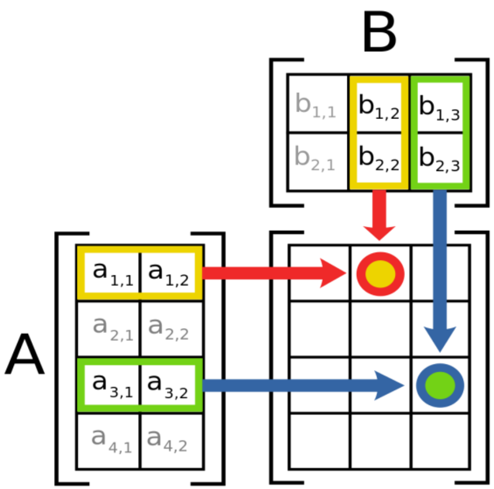
Some special matrices

- Orthogonal matrix ($Q^T Q = I$)
- Rank-deficient matrix (later)
- Singular matrix (later)

Key operations on vectors

- Addition and subtraction
- Multiplication by a number
- Transpose
- Element-wise functions $f(x)$
- Range-functions $f(x_{i..j})$

Matrix multiplication

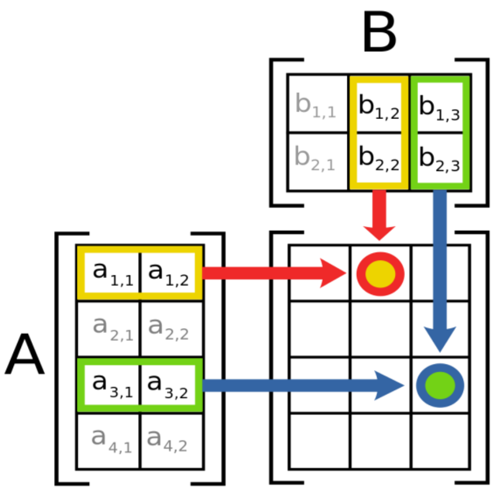


$$\overset{\text{Matrix } \mathbf{A} \text{ is } 3x4}{\begin{bmatrix} 8 & 3 & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}} \overset{\text{Matrix } \mathbf{B} \text{ is } 4x4}{\begin{bmatrix} 5 & \cdot & \cdot & \cdot \\ 4 & \cdot & \cdot & \cdot \\ 3 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot \end{bmatrix}} = \overset{\text{Matrix } \mathbf{C} \text{ is } 3x4}{\begin{bmatrix} 53 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}$$

$$\text{because } c_{11} = \sum_{k=1}^{4} a_{1k} b_{k1} = 8 \cdot 5 + 3 \cdot 4 + 0 \cdot 3 + 1 \cdot 1 = 53$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Matrix multiplication



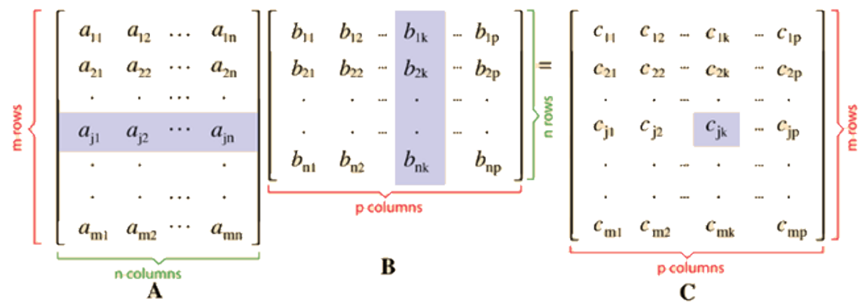
$$C_{ij} = (A * B)_{ij} = \sum_{q=1}^{m} A_{iq} B_{qj}$$

$$\underset{\text{Matrix } \mathbf{A} \text{ is } 3x4}{\begin{bmatrix} 8 & 3 & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}} \underset{\text{Matrix } \mathbf{B} \text{ is } 4x4}{\begin{bmatrix} 5 & \cdot & \cdot & \cdot \\ 4 & \cdot & \cdot & \cdot \\ 3 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot \end{bmatrix}} = \underset{\text{Matrix } \mathbf{C} \text{ is } 3x4}{\begin{bmatrix} 53 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}$$

$$\text{because } c_{11} = \sum_{k=1}^{4} a_{1k} b_{k1} = 8 \cdot 5 + 3 \cdot 4 + 0 \cdot 3 + 1 \cdot 1 = 53$$

Matrix multiplication

$$C_{ij} = (A * B)_{ij} = \sum_{q=1}^{m} A_{iq} B_{qj}$$



illegal

$$\overset{\text{Matrix A}}{\begin{bmatrix} 1 & 4 & 6 & 10 \\ 2 & 7 & 5 & 3 \end{bmatrix}} \cdot \overset{\text{Matrix B}}{\begin{bmatrix} 1 & 4 & 6 & 10 \\ 2 & 7 & 5 & 3 \\ 9 & 0 & 11 & 8 \end{bmatrix}}$$

$$\overset{\text{Matrix A}}{\begin{bmatrix} 1 & 4 & 6 & 10 \\ 2 & 7 & 5 & 3 \end{bmatrix}} \cdot \overset{\text{Matrix B}}{\begin{bmatrix} 1 & 4 & 6 \\ 2 & 7 & 5 \\ 9 & 0 & 11 \\ 3 & 1 & 0 \end{bmatrix}} = \overset{\text{Product}}{\begin{bmatrix} 93 & 42 & 92 \\ 70 & 60 & 102 \end{bmatrix}}$$

Matrix multiplication

Explicitly 2x2 and 3x3 matrix products:

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} e & f \\ g & h \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}}_{C}$$

$$\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix} \begin{pmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{pmatrix} =$$

$$\begin{pmatrix} a11.b11 + a12.b21 + a13.b31 & a11.b12 + a12.b22 + a13.b32 & a11.b13 + a12.b23 + a13.b33 \\ a21.b11 + a22.b21 + a23.b31 & a21.b12 + a22.b22 + a23.b32 & a21.b13 + a22.b23 + a23.b33 \\ a31.b11 + a32.b21 + a33.b31 & a31.b12 + a32.b22 + a33.b32 & a31.b13 + a32.b23 + a33.b33 \end{pmatrix}$$

Matrix multiplication

Exercise:

$$\begin{bmatrix} \$3 & \$4 & \$2 \end{bmatrix} \times \begin{bmatrix} 13 & 9 & 7 & 15 \\ 8 & 7 & 4 & 6 \\ 6 & 4 & 0 & 3 \end{bmatrix} = \begin{bmatrix} \$83 & \$63 & \$37 & \$75 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 4 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 9 & -3 \\ 6 & 1 \end{bmatrix} = \begin{bmatrix} (-1)(9)+4(6) & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

Matrix A

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

Matrix B

$$\begin{bmatrix} 7 & 4 & 9 \\ 8 & 1 & 5 \end{bmatrix}$$

Matrix multiplication

- Some important properties of matrix multiplication:
  - A*B≠B*A (non commutative)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

  - A*(B*C)=(A*B)*C (associative)
  - A*(B+C)=A*B+A*C (distributive)

Matrix multiplication

Some special matrix products:

- $A*I=A$
- $A*A^{-1}=A^{-1}*A=I$
- $A*B^{-1}=A/B$ (right division)
- $B^{-1}*A=A\backslash B$ (left division)
- $O^{T}*O=I$ (orthogonal matrix)
- A*c (matrix-column vector multiply => c-vector)
- r*A (matrix-row vector multiply => r-vector)
- r*A*c (quadratic form => number)

Dot/inner product

- The matrix product of a **row** and a **column** vector is called **dot** or **scalar** product
  - The multiplied vectors must have the same length!
- The dot product's **result is a number** – [1xn] * [nx1] = [1x1]

Dot/inner product

$$a^T b = s = \sum_{i=1}^{n} a_i b_i$$

$$r * c = \begin{bmatrix} r_1 & \dots & r_n \end{bmatrix} * \begin{bmatrix} c_1 \\ \dots \\ c_n \end{bmatrix} = r_1 c_1 + r_2 c_2 + \dots r_n c_n = \sum_{q=1}^{n} r_q c_q$$

Dot/inner product

The dot product of vector with itself is called **square-norm**

$$x^T x = x_1{}^2 + x_2{}^2 + \ldots + x_n{}^2 = | x |^2$$

Example

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = 14$$

Tensor/outer product

- The matrix product of a **column** and a **row** vector is called **tensor** or **outer** product
- The multiplied row and column vectors **need not** have the same length!
- The dot product's **result is a matrix** – [nx1] * [1xm] = [nxm]

Tensor/outer product

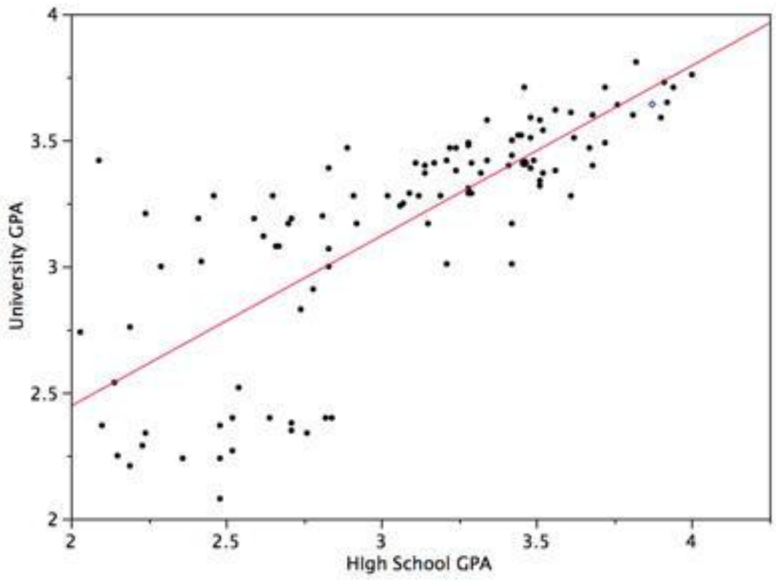
$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^{\mathrm{T}} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}.$$

Example:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^T$$

Example – linear regression

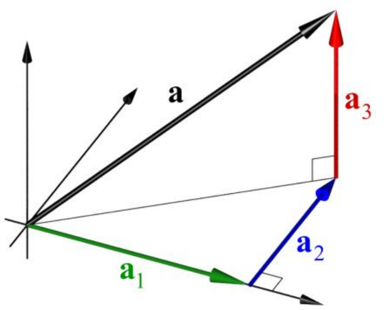
$$y = \theta^T x + \theta_0 = [whiteboard]$$



Norms

Norm is a numerical measure of how big a vector is (typically some sort of average of its values):

$$\|x\|_p = \left(\sum |x|_i^p \right)^{1/p}$$



L$_2$ $$\|x\|_2 = \left(\sum x_i^2 \right)^{1/2}$$

L$_1$ $$\|x\|_1 = \sum |x_i|$$

L$_\infty$ $$\|x\|_\infty = \max |x_i|$$

Norms

Matrix norms:

$$\|A\|_2 = \left(\sum A_{ij}^2\right)^{1/2}$$

$$\|A\|_1 = \sum \left|A_{ij}\right|$$

$$\|A\|_\infty = \max \left|A_{ij}\right|$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

Norms

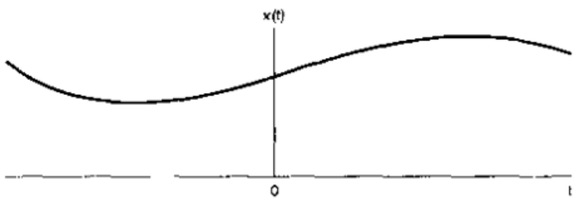
Properties of norms

- $|x| \geq 0$
- $|x|=0 \Leftrightarrow x=0$
- $|kx|=|k|\,|x|$
- $|x+y| \leq |x|+|y|$ (***the triangle inequality***)

Verify these for L_2, L_1 and L_∞

Signals as matrices

When ML is applied to signals, signals in ML algorithms can be represented as vectors



$$x(t) \rightarrow [x_n]$$

$$I(x,y) \rightarrow [I_{nm}]$$

Signals as matrices

Signal convolution using matrices:

$$x'(n) = \sum_{n'=-\infty}^{\infty} x(n')h(n-n') = x(n) * h(n)$$

$$x' = Ax, A_{nm} = h(n-m)$$

<u>The convolution is a matrix multiplication</u> **SHOW HOW**

For computational reasons, never do this in practice !!! **WHY ?**

Signals as matrices

A general linear filter using matrices:

$$x'(n) = \sum_{n'} h_{n'}(n) \cdot x(n')$$

$$x' = Ax, A_{nm} = h_m(n)$$

Signals as matrices

DFT using matrices:

$$x(t) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-ik\omega_0 t} \qquad \tilde{x}(k) = \sum_{t=0}^{N-1} x(t)e^{ik\omega_0 t}$$

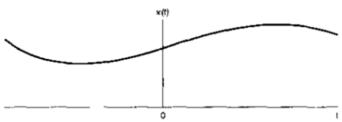
$$x = A\tilde{x},\, A_{tk} = \frac{1}{N}e^{-ik\omega_0 t} \qquad \tilde{x} = A^{-1}x,\, A_{kt} = e^{ik\omega_0 t}$$

Note – because of the special form of the FT matrix A the inverse is very simple
(verify that inverse is correct)

Signals as matrices

Bottom line: Many digital signal processing operations can be implemented as a matrix operations

$$x * h = Ax, A_{nm} = h(n-m)$$



$$x(t) \rightarrow [x_n]$$

$$h[x] = Ax, A_{nm} = h_m(n)$$

$$I(x,y) \rightarrow [I_{nm}]$$

$$x = A\tilde{x},\ \tilde{x} = A^{-1}x,\ A_{tk} = \frac{1}{N}e^{-ik\omega_0 t}$$

Systems of linear equations

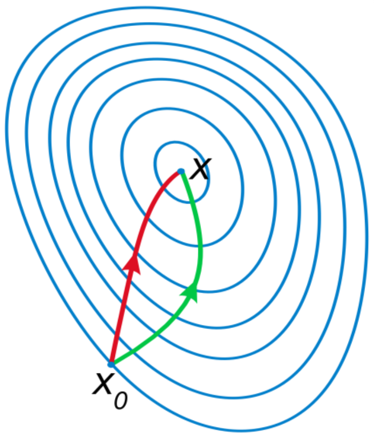
Motivation in ML: Optimizing data models using Newton method

$$[H_f^{(2)}(x_0)]*(x-x_0)=-g_f^{(1)}(x_0)$$

$$H_{11}\delta x_1 + H_{12}\delta x_2 + \ldots + H_{1n}\delta x_n = -g_1$$

$$\ldots$$

$$H_{n1}\delta x_1 + H_{n2}\delta x_2 + \ldots + H_{nn}\delta x_n = -g_n$$



Systems of linear equations

Gaussian elimination:

$$\begin{cases} A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n = b_1 \\ \ldots \\ A_{n1}x_1 + A_{n2}x_2 + \ldots + A_{nn}x_n = b_n \end{cases}$$

EXAMPLE 1 Solve the system by Gaussian elimination.

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 1 \\ 4x_1 + x_2 - x_3 &= 5 \\ x_1 + x_2 + x_3 &= 0. \end{aligned}$$

$$\left[\begin{array}{rrr|r} 2 & -1 & 1 & 1 \\ 4 & 1 & -1 & 5 \\ 1 & 1 & 1 & 0 \end{array}\right] \begin{array}{l} \\ \text{row } 2 - \tfrac{4}{2} \text{ row } 1 \\ \text{row } 3 - \tfrac{1}{2} \text{ row } 1 \end{array}$$

$$\left[\begin{array}{rrr|r} 2 & -1 & 1 & 1 \\ 0 & 3 & -3 & 3 \\ 0 & \tfrac{3}{2} & \tfrac{1}{2} & -\tfrac{1}{2} \end{array}\right] \begin{array}{l} \\ \\ \text{row } 3 - \tfrac{1}{2} \text{ row } 2 \end{array}$$

$$\left[\begin{array}{rrr|r} 2 & -1 & 1 & 1 \\ 0 & 3 & -3 & 3 \\ 0 & 0 & 2 & -2 \end{array}\right]$$

Systems of linear equations

Systems of linear equations as matrix equations:

$$Ax = b$$

$$\begin{bmatrix} & & \\ & \boldsymbol{A} & \\ & & \end{bmatrix} \begin{bmatrix} \\ \boldsymbol{x} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \boldsymbol{b} \\ \\ \end{bmatrix}$$

$$\begin{cases} A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n = b_1 \\ \ldots \\ A_{n1}x_1 + A_{n2}x_2 + \ldots + A_{nn}x_n = b_n \end{cases}$$

Solution is:

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

Systems of linear equations

Gauss-Jordan elimination for $A \to A^{-1}$ $O(n^3)$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 5 & 3 & 0 & 1 & 0 \\ 1 & 0 & 8 & 0 & 0 & 1 \end{array}\right] \xrightarrow[R3 = R3 - \cdot R1]{R2 = R2 - 2 \cdot R1} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2-2\cdot 1 & 5-2\cdot 2 & 3-2\cdot 3 & 0-2\cdot 1 & 1-2\cdot 0 & 0-2\cdot 0 \\ 1-1 & 0-2 & 8-3 & 0-1 & 0-0 & 1-0 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & -2 & 5 & -1 & 0 & 1 \end{array}\right] \xrightarrow[R3 = R3 + 2R2]{} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0+2\cdot 0 & -2+2\cdot 1 & 5+2\cdot(-3) & -1+2\cdot(-2) & 0+2\cdot 1 & 1+2\cdot 0 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & -1 & -5 & 2 & 1 \end{array}\right] \xrightarrow[R3 = -1 \cdot R3]{} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right] \xrightarrow[R2 = R2 + 3 \cdot R3]{R1 = R1 - 3 \cdot R3} \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -14 & 6 & 3 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -14 & 6 & 3 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right] \xrightarrow{R1 = R1 - 2 \cdot R2} \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & -40 & 16 & 9 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

Systems of linear equations

- If inverse of a matrix A^{-1} does not exist, such matrix is called **singular**
- Singular matrices have specific form after Gauss-Jordan procedure, called **rank-deficient**, shown below
- Linear systems with rank-deficient form cannot have a solution in general

Rank deficient matrices:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}$$

Systems of linear equations

Over-determined (toll) and under-determined (long) systems:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

well formed

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

over-determined or toll - no solutions

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

under-determined or long - infinitely many solutions

Systems of linear equations

- For large problems (n>100,1000) Gauss-Jordan elimination becomes not doable, $O(n^3)$ time!
- In that situation, iterative methods can be used to obtain approximate solutions such as Conjugate Gradients (see Advanced) – essentially an approximate solution to the min-square problem shown below:

$$Ax = b \Rightarrow \min |Ax - b|^2$$

Systems of linear equations

In that type of approaches, matrix inverse is not explicitly constructed; instead $\boldsymbol{A}^{-1}\boldsymbol{b}$ is computed for each new $\boldsymbol{b}$ separately as an approximate solution of the below optimization problem. (Not that the two are not equivalent.)

$$Ax = b \Rightarrow \min |Ax - b|^2$$

QUESTIONS FOR SELF-CONTROL

- Explain what frequency response or frequency characteristic of a filter or a linear system is?
- What is the output of a linear system/filter if input is a harmonic $e^{i\omega t}$?
- Solve the differential equation $y''-2y'+2y=x(t)$ using the method of Fourier transform.
- Find the frequency characteristic of the filter $y(n)=x(n)-2x(n-1)+3y(n-1)-y(n-2)$ using the method of Fourier transform.
- What are the advantages and disadvantages of Butterworth, Chebyshev and Elliptical low-pass filters for applications?
- What does $\theta^T x$ mean in the linear equation formula $y= \theta^T x$?
- Calculate the matrix product $[3,2,3][1,2,1]^T$.
- Calculate the matrix product $[3,2,3]^T[1,2,1]$.
- Calculate the matrix product $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \\ -2 & 0 & 1 \end{bmatrix}\begin{bmatrix} -2 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$

- Prove associative law of matrix product (AB)C=A(BC) from matrix product's definition.
- Prove distributive law of matrix product A(B+C)=AB+AC from matrix product's definition.
- Define each of the following: square matrix, identity matrix, diagonal matrix, rectangular matrix, banded-diagonal matrix, symmetric matrix, asymmetric matrix, upper/lower-triangular matrix, orthogonal matrix, singular matrix, rank-deficient matrix.
- Define matrix transposition; find the transpose of the matrix $[a_{ij}]=i-2j$, $i,j=1..10$.
- Assume x is 15x1 column vector and A is 15x15 square matrix. What is the shape of the matrix products x^TAx, Ax, x^TA?
- Define the square norm of a vector by using matrix multiplication.
- Define triangle inequality.

- Represent Fourier and inverse Fourier transform via matrix product. Let's say that the Fourier transform's matrix is A and the inverse Fourier transform's matrix is B. Prove by direct calculation that A*B=I in general.
- Solve the following system of linear equations via Gaussian elimination

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 1 \\ 4x_1 + x_2 - x_3 &= 5 \\ x_1 + x_2 + x_3 &= 0. \end{aligned}$$

- Calculate the inverse of the coefficients matrix in the previous system via Gauss-Jordan method. Verify that indeed the solution is $x=A^{-1}b$.
- Perform Gauss-Jordan elimination on the following matrix and prove that it is rank-deficient and singular (that is the inverse cannot exist).

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \\ 2 & 5 & 2.5 \end{bmatrix}$$

- Define well-formed, under-determined and over-determined systems of linear equations. Define the main properties of the solutions of each class of systems.

ADVANCED

Differential equations & FT

Linear differential and integral equations become very simple in Fourier space:

$$y'(t) + Ay(t) = Bx(t) \quad \Rightarrow \quad i\omega\tilde{y}(\omega) + A\tilde{y}(\omega) = B\tilde{x}(\omega)$$

Solution: $\tilde{y}(\omega) = B/(i\omega + A) \cdot \tilde{x}(\omega)$

Is in fact a filter with: $h(\omega) = B/(i\omega + A)$

Differential equations & FT

Linear differential and integral equations become very simple in Fourier space:

$$A\int y(t)dt + y(t) = Bx(t) \Rightarrow (i\omega)^{-1}A\tilde{y}(\omega) + \tilde{y}(\omega) = B\tilde{x}(\omega)$$

Solution: $\tilde{y}(\omega) = i\omega B/(i\omega + A)\cdot\tilde{x}(\omega)$

$$h(\omega) = ?$$

Differential equations & FT

In general:

$$\sum_{k=0}^{n_a} a_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{n_b} b_k \frac{d^k x(t)}{dt^k}$$

In Fourier space the solution is:

$$\tilde{y}(\omega) = \frac{b_0 + (i\omega)^1 b_1 + \ldots + (i\omega)^{n_b} b_{n_b}}{a_0 + (i\omega)^1 a_1 + \ldots + (i\omega)^{n_a} a_{n_a}} \tilde{x}(\omega)$$

Differential equations & FT

Question: Can any LTI system be described by a differential equation?

The answer is No, only the following:

$$h(\omega) = \frac{b_0 + (i\omega)^1 b_1 + \ldots + (i\omega)^{n_b} b_{n_b}}{a_0 + (i\omega)^1 a_1 + \ldots + (i\omega)^{n_a} a_{n_a}} = \frac{P(i\omega)}{Q(i\omega)}$$

(this is called rational function)

Conjugate Gradients Method

- **Conjugate Gradients Method** (CGM) is a method for approximately solving systems of linear equations Ax=b when A is symmetric
- Assuming the size of system Ax=b (that is the number of equations and variables) is p, CGM is exact if continued for p steps
- A special property of GCM is that a good approximation to the solution x* is obtained already after few first steps of the algorithm
- If terminated early, thus, CGM becomes an approximate linear solver that can be used for large scale linear problems

Conjugate Gradients Method

CGM constructs a series of conjugate directions forming a new vector basis for x, in which basis the solution x* takes on a simple form.

Conjugate directions are such vectors $\boldsymbol{P_i}$ that $\boldsymbol{P_i^T A P_j = 0}$ for any $i \neq j$, given the problem's matrix A

Conjugate Gradients Method

In conjugate basis, considering $P_i^T A P_j = 0$, $i \neq j$, the solution x* is trivially found:

$$\boxed{x^* = \sum_{i=1}^{p} a_i P_i,}$$

$$Ax^* = b \Rightarrow P_i A x^* = P_i b \Rightarrow \boxed{a_i = \frac{P_i^T b}{P_i^T A P_i}}$$

Of course the challenge then is to find the conjugate basis.

Conjugate Gradients Method

GCM constructs such basis starting with initial guess point x_0, residual r_0=b-Ax_0 and the 1st conjugate direction p_0=r_0. Then, GCM repeats for each next step k:

$$a_k = \frac{r_k^{\;T} r_k}{p_k^{\;T} A p_k}$$

$$x_{k+1} = x_k + a_k p_k$$

$$r_{k+1} = r_k - a_k A p_k$$

$$b_k = \frac{r_{k+1}^{\;T} r_{k+1}}{r_k^{\;T} r_k}$$

$$p_{k+1} = r_{k+1} + b_k p_k$$

Conjugate Gradients Method

The motivation for these choices is to maintain the following two conditions, which therefore ensure that the directions p_k are conjugate:

$$r_{k+1}{}^T r_k = 0$$

$$p_{k+1}{}^T A p_k = 0$$

It is fun to see how these work.
I advise you to check by yourself that these two conditions are indeed maintained in GCM algorithm $\rightarrow$

GCM Algorithm:

$$a_k = \frac{r_k{}^T r_k}{p_k{}^T A p_k}$$

$$x_{k+1} = x_k + a_k p_k$$

$$r_{k+1} = r_k - a_k A p_k$$

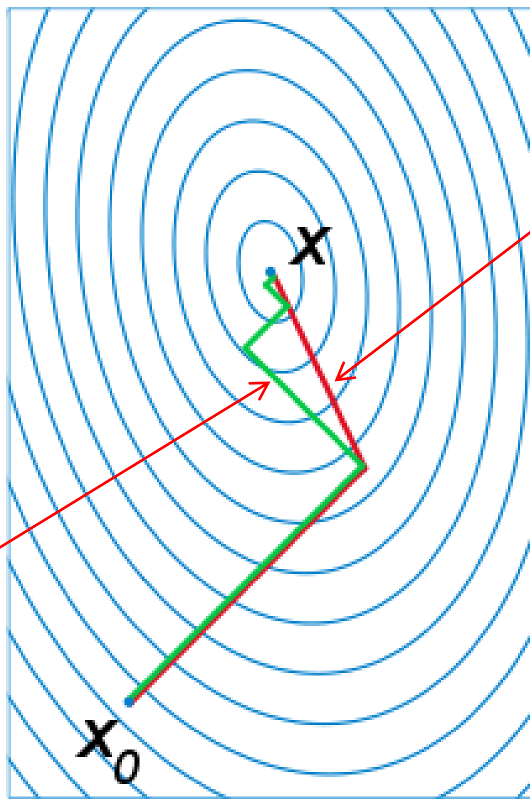
$$b_k = \frac{r_{k+1}{}^T r_{k+1}}{r_k{}^T r_k}$$

$$p_{k+1} = r_{k+1} + b_k p_k$$

Conjugate Gradients Method

The conjugate basis constructed by CGM usually allows a very good approximation to the solution x* to be made already after the first few steps of the algorithm – in that sense CGM is a very efficient approximate algorithm for solving large-scale symmetric linear problems.

Conjugate Gradients Method

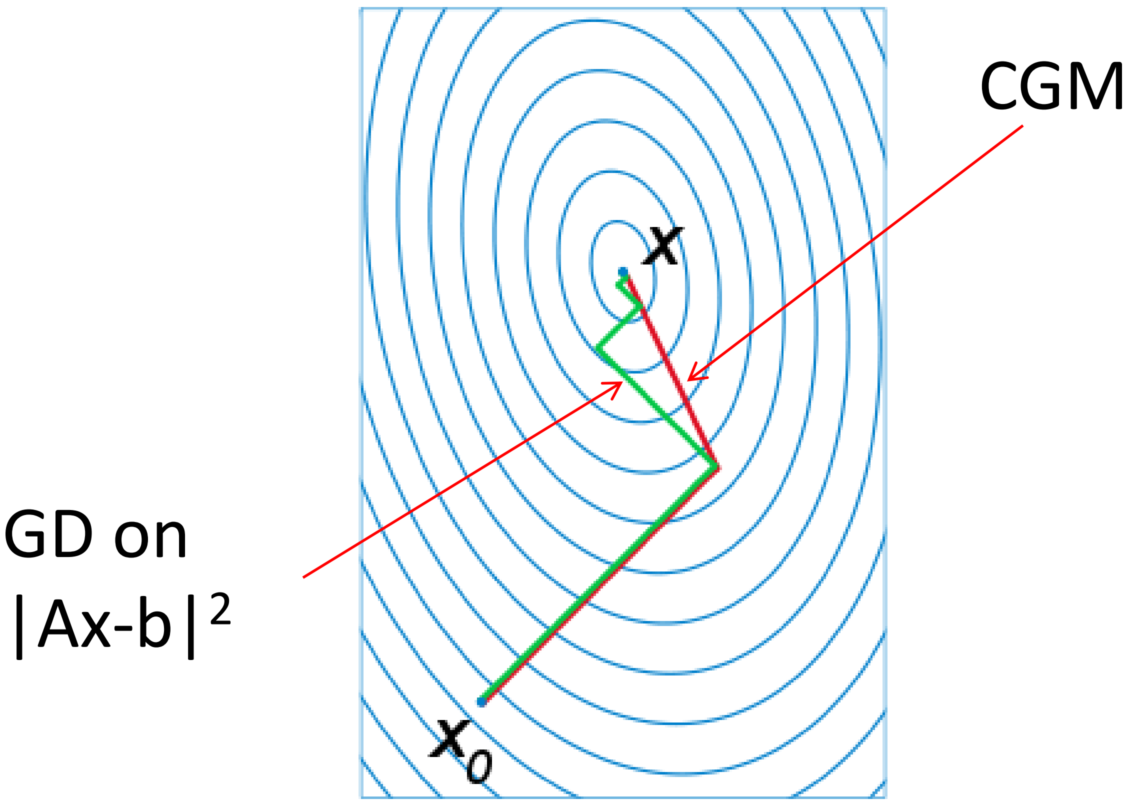


CGM

GD on $|Ax-b|^2$

CGM's time complexity is typically $O(\kappa(A)p^2)$, where $\kappa(A)$ is the condition number of the problem's matrix A

Conjugate Gradients Method



Note that also one straightforward option for large linear problems is to use the Gradient Descent algorithm (next lecture) on the large quadratic optimization problem **min |Ax-b|²**, which will also work to yield solution Ax*=b, but typically will be much slower than GCM

Geometrical interpretation of dot product

Geometrically dot product is related to cosine of the angle between two vectors

$$a^T b = \sum a_i b_i = |a||b|\cos\phi \le |a||b|$$

This follows from **Schwarz inequality**, and allows **cos** to be defined much more generally in any number of dimensions:

$$\cos\phi \stackrel{def}{=} \frac{a^T b}{|a||b|}$$

Schwarz inequality

$$a^T b \le |a||b|$$

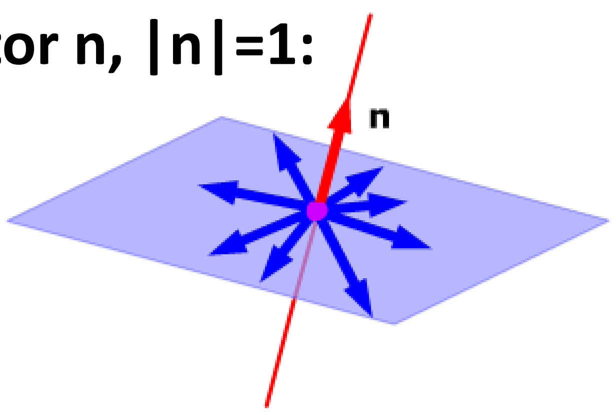
$$\textit{proof follows from } (a-\lambda b)^2 = (a-\lambda b)^T (a-\lambda b) \ge 0, \textit{ then choose } \lambda = a^T b / |b|$$

High-dimensional planes

Definition of a plane: family of vectors orthogonal to a common direction n

$$n^T \cdot (x - x_0) = 0 (because\ \cos\frac{\pi}{2} = 0)$$

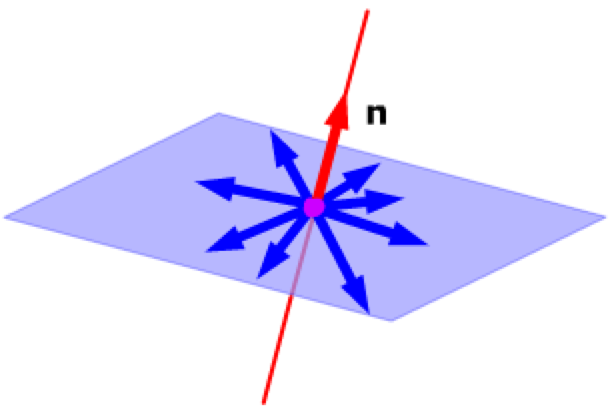
The normal vector n, |n|=1:



High-dimensional planes

Equation of plane ($c = n^T x_0$ is up/down shift):

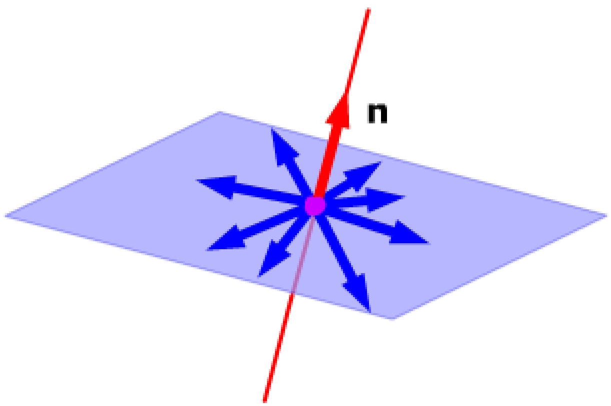
$$\boxed{n^T x - c = 0}$$



$$n^T x - c = d$$ is (also) the (signed) distance from the plane.

High-dimensional planes

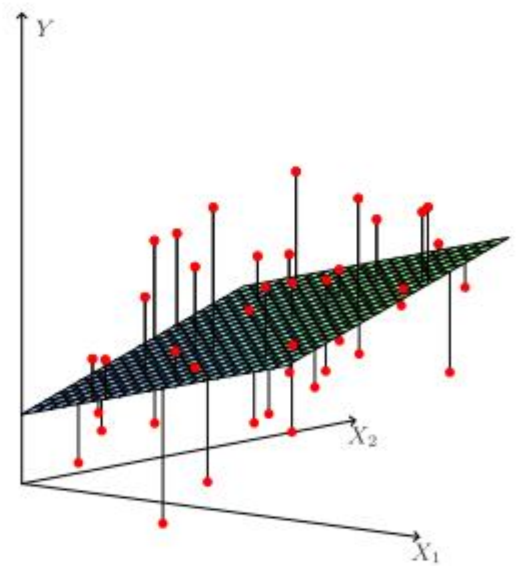
Another view on plane is as the all linear combinations of the vectors comprising the plane



$$\upsilon = \sum \alpha_i \upsilon_i = span(\upsilon_1, \ldots, \upsilon_m)$$

This is the same definition with **vector space** – indeed every plane is a vector space !

Geometrical view of linear regression



Minimize Residual Sum-of-Squares:

$$\min_{\theta} \sum (y_i - x_i^T \theta)^2$$

Alternative view of this problem:

$$\min \left\| \vec{y} - \hat{\vec{y}} \right\|^2$$

$$s.t.$$

$$\hat{y} = X\theta$$

Geometrical view of linear regression

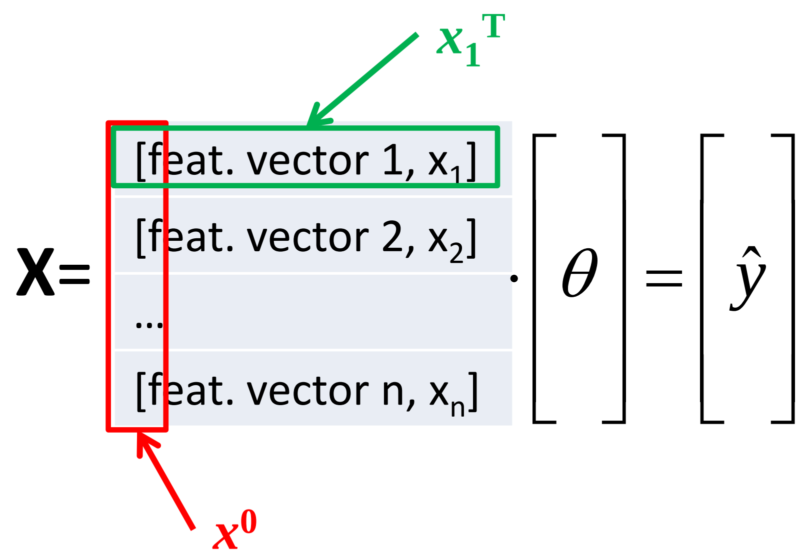
Here:

$\boldsymbol{y}=[y_i]$ is a column vector nx1 of n outputs y_i

$$\hat{y} = X\theta$$

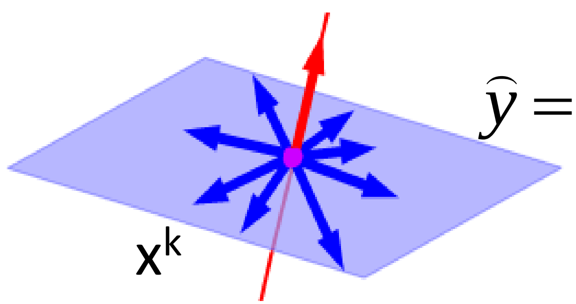
$$that\,is\;\; y_i = \vec{x}_i^T\theta,\; i = 1..n$$

$\boldsymbol{X}=[x^0x^1\ldots x^p]$ is a nxp matrix made of p columns each describing one feature x_k for n inputs, or n rows each representing one input vector x_i^T – either way is the same matrix



Geometrical view of linear regression

$\widehat{y} = X \cdot \theta$ can be viewed as a linear combination of p column-vectors x^k, each of size nx1, and therefore a vector in a p-dimensional plane inside nx1-dimensional column vector space. The plane is built on the p column-vectors x^k



$\widehat{y} = X \cdot \theta,\ size\ n \times 1$

Geometrical view of linear regression

Therefore, linear regression can be treated as a projection of nx1 vectors y onto the p-plane in nx1 space spanned on the column-vectors x^k forming the feature matrix **X**. The minimization of the residual distance $|y-y\hat{}|^2$ can be viewed geometrically as finding such a point in that plain that has the least distance from the full y!

$$\min\left\|\vec{y}-\hat{\vec{y}}\right\|^2$$