ENGG 5781 Matrix Analysis and Computations Lecture 2: Linear Representations and Least Squares

Wing-Kin (Ken) Ma

2024–25 First Term

Department of Electronic Engineering The Chinese University of Hong Kong

Lecture 2: Least Representations and Least Squares

- Part I: linear representations
 - time-series modeling, Vandemonde matrix
 - basis representation
 - discrete-time linear time-invariant systems, Toeplitz matrix, circulant matrix
 - OFDM, localization
- Part II: least squares (LS)
 - projection theorem, orthogonal projection, pseudo-inverse
 - LS by optimization
- Part III: extensions
 - matrix factorization, PCA, matrix completion
 - gradient descent, online algorithms

Main Result

• Problem: given $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, solve

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \tag{LS}$$

- find an ${\bf x}$ whose residual ${\bf r}={\bf y}-{\bf A}{\bf x}$ is the smallest in the Euclidean sense
- **Solution:** suppose that **A** has full column rank. The solution to (LS) is unique and is given by

$$\mathbf{x}_{\mathsf{LS}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$$

- if A is semi-orthogonal, the solution is simplified to $\mathbf{x}_{LS} = \mathbf{A}^T \mathbf{y}$
- unless specified, in this lecture we will assume ${\bf A}$ to have full column rank without further mentioning

Part I: Linear Representations

Linear Representation

There are numerous applications in which we deal with a representation

$$\mathbf{y} = \mathbf{A}\mathbf{x},$$

or

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v},$$

where y is given; A is given or stipulated; x is to be determined; v is noise or error.

Time Series

- let y_t , $t = 0, 1, \ldots$, be a real-valued time series.
- examples: speech signal, music, stock market index, real-time seismic waveforms, air quality index (AQI), sunspot counts, ...



Sunspot time series. Source: http://sunspotwatch.com

Time Series

- one can analyze a time series using model-free techniques such as Fourier transform
 - by model-free, we mean that we make little assumptions on the time series
- we can also apply a model
- model-based approaches exploit problem natures and can work very well assuming that you choose a right model for your data

Harmonic Model for Time Series

• Harmonic model:

$$y_t = \sum_{i=1}^k A_i r_i^t \cos(2\pi f_i t + \phi_i) + v_t, \quad t = 0, 1, \dots$$

for some positive integer k and for some $A_i > 0$, $r_i > 0$, $f_i \in \left[-\frac{1}{2}, \frac{1}{2}\right)$, $\phi_i \in [0, 2\pi)$, $i = 1, \ldots, k$; v_t is noise or modeling error.

- (A_i, r_i, f_i, ϕ_i) 's are model parameters and unknown
- -k is called the model order; also unknown but we can plug a guess number
- we can use the *Hilbert transform*¹ to convert y_t to a complex time series

$$\tilde{y}_{t} = \sum_{i=1}^{k} A_{i} r_{i}^{t} e^{j(2\pi f_{i}t + \phi_{i})} + \tilde{v}_{t} = \sum_{i=1}^{k} \alpha_{i} z_{i}^{t} + \tilde{v}_{t},$$

where $\alpha_i = A_i e^{j\phi_i}$, $z_i = r_i e^{j2\pi f_i}$.

¹call hilbert on MATLAB

Harmonic Model for Time Series

• suppose z_i 's are known, and the observation time window is T. Then,



- we can estimate the amplitude-phase coefficients α_i 's from $\{\tilde{y}_t\}$ via LS, given information of the frequencies f_i 's and the damping coefficients r_i 's

Vandemonde Matrix

A matrix $\mathbf{A} \in \mathbb{C}^{m imes n}$ is said to be Vandemonde if it takes the form

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_n \\ z_1^2 & z_2^2 & \cdots & z_n^2 \\ \vdots & & & \vdots \\ z_1^{m-1} & z_2^{m-1} & \cdots & z_n^{m-1} \end{bmatrix},$$

where $z_i \in \mathbb{C}$, i = 1, ..., n, are called the roots of the Vandemonde matrix.

- Fact: a Vandemonde A has full rank if its roots are distinct; i.e., $z_i \neq z_j$ for all i, j with $i \neq j$
 - Vandemonde matrices possess a stronger linear independence property: if we pick any k columns of A, with $k \leq m$, they are always linearly independent.

Autoregessive Model for Time Series

• Autoregressive (AR) model:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_q y_{t-q} + v_t, \quad t = 0, 1, \ldots$$

for some coefficient $\mathbf{a} \in \mathbb{R}^q$ and for some positive integer (or model order) q.



- model y_t as being related to its past values in a linear manner
- also called the all-pole model in signals and systems

Autoregessive Model for Time Series

• Prediction: suppose a is known and we have the time series up to time t-1.

- we may predict the present from the past via

$$\hat{y}_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_q y_{t-q}$$

- we may also try to predict the future by recursively running

$$\hat{y}_{t+d} = a_1 \hat{y}_{t+d-1} + a_2 \hat{y}_{t+d-2} + \ldots + a_q \hat{y}_{t+d-q}, \quad d = 1, 2, \ldots$$

where we denote $\hat{y}_{t-i} = y_{t-i}$ for $i = 1, \ldots, q$.



blue: Hang Seng Index during a certain time period. red: training phase; $\hat{y}_t = \sum_{i=1}^q a_i y_{t-i}$; **a** is obtained by LS; q = 10. green: prediction phase; $\hat{y}_{t+d} = \sum_{i=1}^q a_i \hat{y}_{t+d-i}$.

Autoregessive Model for Time Series

• let T + 1 be the observation time window. We have



- we can estimate the AR coefficients a_i 's from $\{y_t\}_{t=0}^T$ via LS

Moving Average Model for Time Series

• Moving Average (MA) model:

$$y_t = b_1 v_t + b_2 v_{t-1} + \ldots + b_p v_{t-p+1}, \quad t = 0, 1, \ldots$$

for some coefficient $\mathbf{b} \in \mathbb{R}^p$; p is the model order; v_t is unknown but assumed to be "white."

• not as simple as the AR case; *roughly* speaking we can do this trick:

$$Y(z) = B(z)V(z) \Longrightarrow \underbrace{\frac{1}{B(z)}}_{=A(z)} Y(z) = V(z) \Longrightarrow \quad \begin{array}{c} \text{convert back in time as AR} \\ \text{with many } a_i\text{'s} \end{array}$$

here X(z) denotes the z-transform of x_t .

- one can also do ARMA
- further reading: [Stoica-Moses'97]

Polynomial Model for Time Series

• Polynomial model:

$$y_t = a_0 + a_1 t + a_2 t^2 + \ldots + a_p t^p + v_t, \quad t = 0, 1, \ldots,$$

where $\mathbf{a} \in \mathbb{R}^{p+1}$.

– p=1: a line, p=2: quadratic, \ldots

- Interpolation: use $a_0 + a_1t + a_2t^2 + \ldots + a_pt^p$ to predict y_t for any $t \in \mathbb{R}$
- we have

$$\underbrace{ \begin{bmatrix} y_0 \\ \vdots \\ y_t \\ \vdots \\ y_{T-1} \end{bmatrix}}_{=\mathbf{y}} = \underbrace{ \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \vdots & & \vdots \\ 1 & t & \cdots & t^p \\ \vdots & & \vdots \\ 1 & T-1 & \cdots & (T-1)^p \end{bmatrix}}_{=\mathbf{A}} \underbrace{ \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{bmatrix}}_{=\mathbf{x}} + \underbrace{ \begin{bmatrix} v_0 \\ \vdots \\ v_t \\ \vdots \\ v_{T-1} \end{bmatrix}}_{=\mathbf{v}}$$

- \mathbf{A}^T is Vandemonde with distinct roots; thus \mathbf{A} has full rank

Curve Fitting

Aim: given a set of input-output data pairs $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$, i = 1, ..., m, find a function f(x) that fits the data well



Curve Fitting





"True" curve: the true f(x); p = 5. Fitted curve: estimated f(x); a obtained by LS; p = 5.

Basis Representation

• Aim: represent a given vector \mathbf{y} using a basis $\{\phi_1, \dots, \phi_n\} \subseteq \mathbb{R}^n$:

$$\mathbf{y} = \sum_{i=1}^{n} x_i \boldsymbol{\phi}_i = \mathbf{\Phi} \mathbf{x},$$

where \mathbf{x} is the coefficient

- we will call $\mathbf{\Phi} \in \mathbb{R}^{n imes n}$ a basis matrix or a dictionary
- in particular, we wish x would be sparse, or approximately sparse in the sense that $\|\mathbf{x}\|_2^2$ is dominated by a few x_i 's
- \bullet having a sparse ${\bf x}$ is good as it enables compact representation and compression
- Φ is specifically designed; many designs lead to orthogonal Φ

Basis Representation

• example: orthonormal Fourier basis

$$\phi_{i} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 \\ e^{j2\pi(i-1)/n} \\ \vdots \\ e^{j2\pi(n-1)(i-1)/n} \end{bmatrix}, \quad i = 1, \dots, n.$$

- Φ^H is a discrete Fourier transform (DFT) matrix; it can be verified that if we let $\Psi = \Phi^H$ then $\psi_i = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 \ e^{-j2\pi(i-1)/n} \ \dots \ e^{-j2\pi(n-1)(i-1)/n} \end{bmatrix}^T$

- Φ is an inverse DFT (IDFT) matrix
- we don't store Φ physically; we use fast Fourier transform (FFT) and inverse FFT (IFFT) to implement $\mathbf{x} = \Phi^H \mathbf{y}$ and $\mathbf{y} = \Phi \mathbf{x}$, resp.
- FFT or IFFT complexity: $\mathcal{O}(n \log(n))$
- other basis examples: discrete cosine transform (DCT), Haar, wavelets, ...

Basis Example: DFT Basis



n = 8; circles: values of the basis elements; lines: interpolated values for better visualization; blue: real part of the basis elements; red: imaginary part of the basis elements.

Basis Example: Haar Wavelet



n = 8; circles: values of the basis elements; lines: interpolated values for better visualization.

Basis Representation Example for Images



Image representation using a 2D-DCT basis. Left: an image is first cropped into patches, each with a size of 8×8 . Right: each patch is represented by a linear combination of basis elements.

Basis Representation Example for Images



(a) 2D DCT dictionary.



(b) 2D Haar wavelet dictionary.

Illustration of the 2D DCT and Haar wavelet dictionaries. Source: [Aharon-Elad-Bruckstein'06]. Note that the dictionaries shown are overcomplete.

• consider linear time-invariant system models in discrete-time signal processing:

$$y_t = \sum_{i=0}^p h_i x_{t-i} + v_t, \quad t = 0, 1, \dots$$

where x_t is the input signal; y_t is the output signal; v_t is noise; $\{h_t\}$ is the system impulse response.

- some mild assumptions: $\{h_t\}$ is finite in length; $x_t = 0$ for $t = -1, -2, \ldots$
- applications: communications, acoustics, image processing...



(a) multipath propagation in wireless communications.



- System identification: given an input signal block $\{x_t\}_{t=0}^{T-1}$ and an output signal block $\{y_t\}_{t=0}^{T-1}$, find the system impulse response $\{h_t\}_{t=0}^p$.
 - applications: channel estimation in communications, identification of acoustic impulse responses,...

• we have



- Deconvolution: given an output signal block $\{y_t\}_{t=0}^{T-1}$ and the system impulse response $\{h_t\}_{t=0}^p$, estimate the input signal block $\{x_t\}_{t=0}^{T-1}$
 - applications: equalization in communications, de-reverberation in room acoustics, image deblurring,...
- we have



– ${\bf A}$ is band diagonal and Toeplitz

Toeplitz Matrix

A matrix $\mathbf{A} \in \mathbb{R}^{n imes n}$ is said to be Toeplitz if it takes the form

$$\mathbf{A} = \begin{bmatrix} h_0 & h_{-1} & \dots & h_{-n+1} \\ h_1 & h_0 & h_{-1} & & \vdots \\ \vdots & h_1 & h_0 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & h_{-1} \\ h_{n-1} & \dots & \dots & h_1 & h_0 \end{bmatrix},$$

or $a_{ij} = h_{i-j}$ for all i, j.

- for a general $\mathbf{A} \in \mathbb{R}^{n \times n}$, solving $\mathbf{A}\mathbf{x} = \mathbf{y}$ requires $\mathcal{O}(n^3)$
- for a Teoplitz A, $\mathbf{A}\mathbf{x} = \mathbf{y}$ may be solved in $\mathcal{O}(n^2)$
 - done by exploiting structures; see [Golub-Van Loan'12] for details

A matrix $\mathbf{A} \in \mathbb{R}^{n imes n}$ is said to be circulant if it takes the form

$$\mathbf{A} = \begin{bmatrix} h_0 & h_{n-1} & \dots & h_1 \\ h_1 & h_0 & h_{n-1} & \dots & h_2 \\ h_2 & h_1 & h_0 & \dots & h_3 \\ \vdots & & & & \vdots \\ \vdots & & & & & \vdots \\ h_{n-1} & \dots & \dots & h_1 & h_0 \end{bmatrix}.$$

• for a circulant A, Ax = y may be solved in $\mathcal{O}(n \log(n))$

• let $\{ oldsymbol{\phi}_1, \dots, oldsymbol{\phi}_n \}$ be the DFT basis, and observe that

$$\begin{split} \mathbf{A}\phi_{i} &= \frac{1}{\sqrt{n}} \begin{bmatrix} h_{0} & h_{n-1} & \dots & h_{1} \\ h_{1} & h_{0} & h_{n-1} & \dots & h_{2} \\ h_{2} & h_{1} & h_{0} & \dots & h_{3} \\ \vdots & & & \vdots \\ h_{n-1} & \dots & & h_{1} & h_{0} \end{bmatrix} \begin{bmatrix} 1 \\ e^{j2\pi(i-1)/n} \\ \vdots \\ e^{j2\pi(n-1)(i-1)/n} \end{bmatrix} \\ &= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} h_{k} e^{-j2\pi k(i-1)/n} \\ &= d_{i} \end{bmatrix} \begin{bmatrix} 1 \\ e^{j2\pi(i-1)/n} \\ e^{j4\pi(i-1)/n} \\ \vdots \\ e^{j2\pi(n-1)(i-1)/n} \end{bmatrix} = d_{i}\phi_{i}. \end{split}$$

- note
$$e^{j2\pi k(i-1)/n} = e^{-j2\pi (n-k)(i-1)/n}$$
 for any $k \in \{0, 1, \dots, n-1\}$

• let $\mathbf{D} = \text{Diag}(d_1, \ldots, d_n)$. We have

$$\begin{aligned} \mathbf{A}\boldsymbol{\phi}_{i} &= d_{i}\boldsymbol{\phi}_{i}, \ i = 1, \dots, n &\iff \mathbf{A}[\ \boldsymbol{\phi}_{1}, \dots, \boldsymbol{\phi}_{n}\] = [\ \boldsymbol{\phi}_{1}, \dots, \boldsymbol{\phi}_{n}\] \mathbf{D} \\ &\iff \mathbf{A}\boldsymbol{\Phi} = \boldsymbol{\Phi}\mathbf{D} \\ &\iff \mathbf{A} = \boldsymbol{\Phi}\mathbf{D}\boldsymbol{\Phi}^{H} \end{aligned}$$

• Fact (as a summary): a circulant matrix $\mathbf{A} \in \mathbb{R}^{n imes n}$ can be decomposed as

$$\mathbf{A} = \mathbf{\Phi} \mathbf{D} \mathbf{\Phi}^H,$$

where Φ is the IDFT matrix; $\mathbf{D} = \text{Diag}(d_1, \ldots, d_n)$; $d_i = \sum_{k=0}^{n-1} h_k e^{-j2\pi k(i-1)/n}$.

- as will be studied, the above decomposition is an eigendecomposition

- Question: how does a circulant A help us solve y = Ax?
 - suppose $d_i \neq 0$ for all i
 - we have $\mathbf{A}^{-1} = \mathbf{\Phi} \mathbf{D}^{-1} \mathbf{\Phi}^H$ and

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} = \mathbf{\Phi}\underbrace{(\mathbf{D}^{-1}\underbrace{(\mathbf{\Phi}^{H}\mathbf{y})}_{n \text{ multiplies}})}_{\text{IFFT}}$$

- complexity: one FFT + n multiplies + one IFFT = $O(n \log(n))$
 - * the above complexity assumes that d_1, \ldots, d_n have been pre-computed; computing d_1, \ldots, d_n requires FFT and the complexity is $\mathcal{O}(n \log(n))$

Circulant Approximation of Linear Time-Invariant Systems

• back to deconvolution, we may approximate the system matrix as being circulant

$$\underbrace{ \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_p \\ \vdots \\ \vdots \\ y_{T-1} \end{bmatrix} }_{=\mathbf{y}} \approx \underbrace{ \begin{bmatrix} h_0 & & h_p & \dots & h_1 \\ h_1 & h_0 & & & \ddots & \vdots \\ h_p & \dots & h_1 & h_0 & & & \\ h_p & \dots & h_1 & h_0 & & & \\ & \ddots & & \ddots & & & \\ & & h_p & \dots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \\ \vdots \\ \vdots \\ x_{T-1} \end{bmatrix} + \underbrace{ \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_p \\ \vdots \\ \vdots \\ v_{T-1} \end{bmatrix} }_{=\mathbf{x}}$$

- appears to be a reasonable approximation if $p \ll T$
 - * a common trick in image processing problems such as deblurring (2D)
- in communications we can even make circulant systems happen

OFDM in Communications

- let $\{\bar{x}_t\}_{t=0}^{T-1}$ be the input signal block we want to send
- physically transmit the input signal block $\{x_t\}_{t=0}^{T+p-1}$ this way:

 $x_t = \bar{x}_{t+T-p}, \quad t = 0, 1, \dots, p-1; \qquad x_{t+p} = \bar{x}_t, \quad t = 0, 1, \dots, T-1$



OFDM in Communications

• ignore $\{y_t\}_{t=0}^{p-1}$ and consider $\{y_t\}_{t=p}^{T+p-1}$ only. It can be verified that



- transceiver scheme 1:
 - transmitter side: put info. in $\bar{\mathbf{x}}$; e.g., $\bar{\mathbf{x}} \in \{-1,1\}^T$ for binary signaling
 - receiver side: estimate $\bar{\mathbf{x}}$ by solving $\mathbf{y} = \mathbf{A}\bar{\mathbf{x}}$ for circulant \mathbf{A} ; 1 FFT+ 1 IFFT
 - such a transceiver scheme is called single-carrier modulation (SCM)
OFDM in Communications

• recall

$$\mathbf{y} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{v} = \mathbf{\Phi}\mathbf{D}\mathbf{\Phi}^H\bar{\mathbf{x}} + \mathbf{v}$$

- transceiver scheme 2:
 - transmitter side: $\bar{\mathbf{x}} = \mathbf{\Phi}\tilde{\mathbf{x}}$ where $\tilde{\mathbf{x}}$ is the info. signal block (say, $\tilde{\mathbf{x}} \in \{-1, 1\}^T$ for binary signaling); 1 IFFT
 - receiver side: $\mathbf{y} = \mathbf{\Phi} \mathbf{D} \tilde{\mathbf{x}} + \mathbf{v}$, so estimate $\tilde{\mathbf{x}}$ via $\mathbf{D}^{-1} \mathbf{\Phi}^{H} \mathbf{y}$; 1 FFT
 - such a transceiver scheme is called orthogonal frequency division multiplexing (OFDM)
- further reading: OFDM details such as cyclic prefix insertion and removal, noise amplification effects, comparison of OFDM and SCM, MMSE receiver; they have been widely described in the literature, so find literature by yourself

Localization

- Aim: locate the Cartesian coordinate of a sensor or device using distance info.
 - applications: localization in a wireless sensor network, GPS
- let $\mathbf{x} \in \mathbb{R}^2$ be the coordinate of the sensor
- the sensor communicates with anchors, which are sensors or devices that know their locations
- let $\mathbf{a}_i \in \mathbb{R}^2$, $i = 1, \dots, m$, be the anchors' locations
- the sensor measures the distances

 $d_i = \|\mathbf{x} - \mathbf{a}_i\|_2, \ i = 1, \dots, m,$

which can be done by time-of-arrival measurements, received signal strength measurements, ping-pong,...



Localization

• observe that

$$d_i^2 = \|\mathbf{x} - \mathbf{a}_i\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{a}_i^T\mathbf{x} + \|\mathbf{a}_i\|_2^2, \quad i = 1, \dots, m,$$

and re-organize the equations as a matrix equation

$$\begin{bmatrix} \|\mathbf{a}_1\|_2^2 - d_1^2 \\ \vdots \\ \|\mathbf{a}_m\|_2^2 - d_m^2 \end{bmatrix} = \begin{bmatrix} 2\mathbf{a}_1^T & -1 \\ \vdots & \vdots \\ 2\mathbf{a}_m^T & -1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \|\mathbf{x}\|_2^2 \end{bmatrix}.$$

Note that the above matrix equation is nonlinear.

• Idea: solve the linear matrix equation

$$\underbrace{\begin{bmatrix} \|\mathbf{a}_1\|_2^2 - d_1^2 \\ \vdots \\ \|\mathbf{a}_m\|_2^2 - d_m^2 \end{bmatrix}}_{=\mathbf{y}} = \underbrace{\begin{bmatrix} 2\mathbf{a}_1^T & -1 \\ \vdots & \vdots \\ 2\mathbf{a}_m^T & -1 \end{bmatrix}}_{=\mathbf{A}} \begin{bmatrix} \mathbf{x} \\ z \end{bmatrix}$$

where (\mathbf{x}, z) is a *free variable* on \mathbb{R}^3 ; or, no constraint $z = \|\mathbf{x}\|_2^2$

Localization

- in practice, the sensor obtains noisy measurements $\hat{d}_i = d_i + v_i$, i = 1, ..., m, where v_t is noise
- we do the engineers' way:
 - replace d_i 's by \hat{d}_i 's, and compute the LS solution $\mathbf{u} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$;
 - use $\hat{\mathbf{x}} = [u_1, u_2]^T$ as the location estimate
- further reading: [Sayed-Tarighat-Khajehnouri'05]



Number of anchors: m = 4. Noise standard deviation: 0.1581km. Number of trials: 200.

Part II: Least Squares

LS Solution

Theorem 2.1. A vector \mathbf{x}_{LS} is an optimal solution to the LS problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2$$

if and only if it satisfies

$$\mathbf{A}^T \mathbf{A} \mathbf{x}_{\mathsf{LS}} = \mathbf{A}^T \mathbf{y}. \tag{*}$$

- the optimality condition in (*) is true for any A, not just full-column rank A
- $\bullet\,$ suppose that ${\bf A}$ has full-column rank
 - $\mathbf{A}^T \mathbf{A}$ is nonsingular (verify as a mini-exercise)
 - the solution to (*) is uniquely given by $\mathbf{x}_{\mathsf{LS}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y}$
- (*) is called the normal equations
- the same result holds for the complex case, viz., $\mathbf{A}^{H}\mathbf{A}\mathbf{x}_{\mathsf{LS}} = \mathbf{A}^{H}\mathbf{y}$

LS and the Projection Theorem

- Theorem 2.1 can be shown using the projection theorem
- $\bullet~$ let \mathbf{x}_{LS} be an LS solution, and observe that

$$\Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \arg\min_{\mathbf{z}\in\mathcal{R}(\mathbf{A})} \|\mathbf{z}-\mathbf{y}\|_{2}^{2} = \mathbf{A}\mathbf{x}_{\mathsf{LS}}$$

• by the projection theorem (Theorem 1.2 in Lecture 1), we have

$$\Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \mathbf{A}\mathbf{x}_{\mathsf{LS}} \quad \iff \quad \mathbf{z}^{T}(\mathbf{A}\mathbf{x}_{\mathsf{LS}} - \mathbf{y}) = 0 \text{ for all } \mathbf{z} \in \mathcal{R}(\mathbf{A})$$
$$\iff \quad \mathbf{x}^{T}\mathbf{A}^{T}(\mathbf{A}\mathbf{x}_{\mathsf{LS}} - \mathbf{y}) = 0 \text{ for all } \mathbf{x} \in \mathbb{R}^{n}$$
$$\iff \quad \mathbf{A}^{T}(\mathbf{A}\mathbf{x}_{\mathsf{LS}} - \mathbf{y}) = \mathbf{0}$$

Orthogonal Projections

- the projections of ${\bf y}$ onto ${\cal R}({\bf A})$ and ${\cal R}({\bf A})^\perp$ are, resp.,

$$\Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \mathbf{A}\mathbf{x}_{\mathsf{LS}} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$
$$\Pi_{\mathcal{R}(\mathbf{A})^{\perp}}(\mathbf{y}) = \mathbf{y} - \Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = (\mathbf{I} - \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T)\mathbf{y}$$

 \bullet the orthogonal projector of ${\bf A}$ is defined as

$$\mathbf{P}_{\mathbf{A}} = \mathbf{A} (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$$

the orthogonal complement projector of \mathbf{A} is defined as

$$\mathbf{P}_{\mathbf{A}}^{\perp} = \mathbf{I} - \mathbf{A} (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T.$$

- obviously, we want to write $\Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \mathbf{P}_{\mathbf{A}}\mathbf{y}$, $\Pi_{\mathcal{R}(\mathbf{A})^{\perp}}(\mathbf{y}) = \mathbf{P}_{\mathbf{A}}^{\perp}\mathbf{y}$
- note: a more general definition for orthogonal projectors will be studied later

Orthogonal Projections

- properties of $\mathbf{P}_{\mathbf{A}}$ (same properties apply to $\mathbf{P}_{\mathbf{A}}^{\perp}$):
 - $\mathbf{P}_{\mathbf{A}}$ is idempotent; i.e., $\mathbf{P}_{\mathbf{A}}\mathbf{P}_{\mathbf{A}}=\mathbf{P}_{\mathbf{A}}$
 - $\mathbf{P}_{\mathbf{A}} = \mathbf{P}_{\mathbf{A}}^T$
- additional properties that will be revealed in later lectures:
 - the eigenvalues of $\mathbf{P}_{\mathbf{A}}$ are either zero or one
 - $\mathbf{P}_{\mathbf{A}}$ can be written as $\mathbf{P}_{\mathbf{A}} = \mathbf{U}_1 \mathbf{U}_1^T$ for some semi-orthogonal \mathbf{U}_1
 - * we can also prove it here:
 - · there always exists a semi-orthogonal U_1 such that $\mathcal{R}(\mathbf{A}) = \mathcal{R}(\mathbf{U}_1)$
 - $\cdot \Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \Pi_{\mathcal{R}(\mathbf{U}_1)}(\mathbf{y}) = \mathbf{U}_1 \mathbf{U}_1^T \mathbf{y}$
 - as $\Pi_{\mathcal{R}(\mathbf{A})}(\mathbf{y}) = \Pi_{\mathcal{R}(\mathbf{U}_1)}(\mathbf{y})$ holds for any \mathbf{y} , or $(\mathbf{P}_{\mathbf{A}} \mathbf{U}_1\mathbf{U}_1^T)\mathbf{y} = \mathbf{0}$ for any \mathbf{y} , we must have $\mathbf{P}_{\mathbf{A}} = \mathbf{U}_1\mathbf{U}_1^T$

Pseudo-Inverse

 \bullet the pseudo-inverse of a full-column-rank ${\bf A}$ is defined as

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

- A^{\dagger} satisfies $A^{\dagger}A = I$, but not necessarily $AA^{\dagger} = I$
- $\mathbf{A}^{\dagger}\mathbf{y}$ is the LS solution
- note: a more general definition for the pseudo-inverse will be studied later

LS by Convex Optimization

- we can also prove the LS optimality condition by optimization
- the gradient of a continuously differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

• Fact: consider an unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable

- suppose f is convex (we skip the def. here). A point \mathbf{x}^* is an optimal solution if and only if $\nabla f(\mathbf{x}^*) = \mathbf{0}$
- for non-convex f, any point $\hat{\mathbf{x}}$ satisfying $\nabla f(\hat{\mathbf{x}}) = \mathbf{0}$ is a stationary point

LS by Convex Optimization

• Fact: consider a quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{R}\mathbf{x} + \mathbf{q}^T \mathbf{x} + c,$$

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is symmetric; i.e., $r_{ij} = r_{ji}$ for all i, j.

- $\nabla f(\mathbf{x}) = \mathbf{R}\mathbf{x} + \mathbf{q}$
- f is convex if \mathbf{R} is positive semidefinite (PSD); for now it suffices to know that if \mathbf{R} takes the form $\mathbf{R} = \mathbf{A}^T \mathbf{A}$ for some \mathbf{A} , it is PSD
- the LS objective function is

$$f(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 = \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} - 2(\mathbf{A}^T \mathbf{y})^T \mathbf{x} + \|\mathbf{y}\|_2^2.$$

Using the above optimization facts, \mathbf{x}_{LS} is an LS optimal solution if and only if $\mathbf{A}^T \mathbf{A} \mathbf{x}_{LS} - \mathbf{A}^T \mathbf{y} = \mathbf{0}$.

LS by Convex Optimization

- using optimization results is handy in some (actually, many) cases
- example: consider a regularized LS problem

 $\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2, \quad \text{for some constant } \lambda > 0.$

- solution by optimization: $\nabla f(\mathbf{x}) = 2\mathbf{A}^T \mathbf{A} \mathbf{x} - 2\mathbf{A}^T \mathbf{y} + 2\lambda \mathbf{x}$. Thus the optimal solution is

$$\mathbf{x}_{\mathsf{RLS}} = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{y}$$

- solution by the projection thm., in contrast: have to rewrite the problem as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix} \mathbf{x} \right\|_2^2,$$

and use the projection theorem to get the same result.

Part III-A: Matrix Factorization

Matrix Factorization

There are also many applications in which we deal with a representation of multiple given y_i 's via

$$\mathbf{y}_i = \mathbf{A}\mathbf{b}_i + \mathbf{v}_i, \quad i = 1, \dots, n,$$

where $\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{b}_i \in \mathbb{R}^k$, i = 1, ..., n; \mathbf{v}_i 's are noise. In particular, both \mathbf{b}_i 's and \mathbf{A} are to be determined.

• for example, in basis representation, we want to learn the dictionary from data

Matrix Factorization

Problem: given $\mathbf{Y} \in \mathbb{R}^{m \times n}$ and a positive integer $k < \min\{m, n\}$, solve

$$\min_{\mathbf{A} \in \mathbb{R}^{m \times k}, \mathbf{B} \in \mathbb{R}^{k \times n}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}\|_F^2$$



• also called low-rank matrix approximation: let $\mathbf{Z} = \mathbf{AB}$. It has $rank(\mathbf{Z}) \leq k$.

Principal Component Analysis

Aim: given a collection of data points $y_1, \ldots, y_n \in \mathbb{R}^m$, perform a low-dimensional representation

$$\mathbf{y}_i = \mathbf{A}\mathbf{b}_i + \mathbf{c} + \mathbf{v}_i, \quad i = 1, \dots, n,$$

where $\mathbf{A} \in \mathbb{R}^{m \times k}$ is a basis matrix; $\mathbf{b}_i \in \mathbb{R}^k$ is the coefficient for \mathbf{y}_i ; $\mathbf{c} \in \mathbb{R}^m$ is the base or mean in statistics terms; \mathbf{v}_i is noise or modeling error.

- Principal component analysis (PCA):
 - choose $\mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}_i$
 - let $ar{\mathbf{y}}_i = \mathbf{y}_i \mathbf{c}$, and solve

$$\min_{\mathbf{A},\mathbf{B}} \|\bar{\mathbf{Y}} - \mathbf{A}\mathbf{B}\|_F^2$$

– we may also want a semi-orthogonal ${\bf A}$



Principal Component Analysis

- applications: dimensionality reduction, visualization of high-dimensional data, compression, extraction of meaningful features from data,...
- an example:
 - senate voting: http://livebooklabs.com/keeppies/c5a5868ce26b8125

Aim: discover thematic information, or topics, from a (often large) collection of documents, such as books, articles, news, blogs,...

• bag-of-words representation: represent each document as a vector of word counts



- $\bullet~$ let n~ be the number of documents
- let $\mathbf{y}_i \in \mathbb{R}^m$ be the bag-of-words representation of the ith document, $i=1,\ldots,n$
- let $\mathbf{Y} = [\mathbf{y}_1, \dots \mathbf{y}_n] \in \mathbb{R}^{m imes n}$, called the term-document matrix
- hypotheses: [Turney-Pantel'10]
 - if documents have similar columns vectors in ${f Y}$, or similar usage of words, they tend to have similar meanings
 - the topic of a document will probabilistically influence the author's choice of words when writing the document



Source: [Blei'12].

 \bullet Problem: apply matrix factorization to a term-document matrix ${\bf Y}$



- ${\bf A}$ is called a term-topic matrix, ${\bf B}$ is called a topic-document matrix
- Interpretation:
 - each column a_i of A should represent a theme topic, e.g., local affairs, foreign affairs, politics, sports... in a collection of newspapers
 - as $\mathbf{y}_i pprox \mathbf{A} \mathbf{b}_i$, each document is postulated as a linear combination of topics
 - matrix factorization aims at discovering topics from the documents



Topics found in a real set of documents. Source: [Blei'12]. The document set consists of 17,000 articles from the journal *Science*. The topics are discovered using a technique called *latent Dirichlet allocation*, which is not the same as, but has strong connections to, matrix factorization.

- topic modeling via matrix factorization has been used in, or is tightly connected to
 - information retrieval, natural language processing, machine learning
 - document clustering, classification and retrieval
 - latent semantic analysis, latent semantic indexing: finding similarities of documents, finding similarities of terms (are "cars," "Lamborghini," and "Ferrari" related?)
- though not considered in this course, it seems better to also model A, B as element-wise non-negative—this will lead to *non-negative matrix factorization*
- further reading: [Turney-Pantel'10]
 - as an aside, it mentions a related application where computers can achieve a score of 92.5% on multiple-choice synonym questions from TOEFL, whereas the average human score is 64.5%

Matrix Factorization

The matrix factorization problem

$$\min_{\mathbf{A} \in \mathbb{R}^{m \times k}, \mathbf{B} \in \mathbb{R}^{k \times n}} \|\mathbf{Y} - \mathbf{AB}\|_F^2$$

- has non-unique factors
 - suppose $(\mathbf{A}^{\star}, \mathbf{B}^{\star})$ is an optimal solution to the problem, and let $\mathbf{Q} \in \mathbb{R}^{k \times k}$ be any nonsingular matrix. Then $(\mathbf{A}^{\star}\mathbf{Q}^{-1}, \mathbf{Q}\mathbf{B}^{\star})$ is also an optimal solution.
 - the non-uniqueness of $({\bf A},{\bf B})$ makes the above matrix factorization formulation a bad formulation for problems such as topic modeling
- is non-convex, but can be solved by singular value decomposition (beautifully)
- can also be handled by LS

Matrix Factorization

• Alternating LS (ALS): given a starting point $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)})$, do

$$\mathbf{A}^{(i+1)} = \arg\min_{\mathbf{A}\in\mathbb{R}^{m\times k}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}^{(i)}\|_{F}^{2}$$
$$\mathbf{B}^{(i+1)} = \arg\min_{\mathbf{B}\in\mathbb{R}^{k\times n}} \|\mathbf{Y} - \mathbf{A}^{(i+1)}\mathbf{B}\|_{F}^{2}$$

for i = 0, 1, 2, ..., and stop when a stopping rule is satisfied.

• let's make a mild assumption that $\mathbf{A}^{(i)}, \mathbf{B}^{(i)}$ have full rank at every i. Then,

 $\mathbf{A}^{(i+1)} = \mathbf{Y}(\mathbf{B}^{(i)})^T (\mathbf{B}^{(i)}(\mathbf{B}^{(i)})^T)^{-1}, \quad \mathbf{B}^{(i+1)} = ((\mathbf{A}^{(i+1)})^T \mathbf{A}^{(i+1)})^{-1} (\mathbf{A}^{(i+1)})^T \mathbf{Y}$

- ALS is guaranteed to converge an optimal solution to $\min_{A,B} ||Y AB||_F^2$ under some mild assumptions [Udell-Horn-Zadeh-Boyd'16]
 - note: this result is specific and does not directly carry forward to other related problems such as low-rank matrix completion

Low-Rank Matrix Completion

- let $\mathbf{Y} \in \mathbb{R}^{m \times n}$ be a matrix with missing entries, i.e., the values y_{ij} 's are known only for $(i, j) \in \Omega$ where Ω is an index set that indicates the available entries
- \bullet Aim: recover the missing entries of ${\bf Y}$
- application: recommender system, data science
- example: movie recommendation (further reading: [Koren-Bell-Volinsky'09])
 - ${f Y}$ records how user i likes movie j
 - Y has lots of missing entries; a user doesn't watch all movies
- $\mathbf{Y} = \begin{bmatrix} 2 & 3 & 1 & ? & ? & 5 & 5 \\ 1 & ? & 4 & 2 & ? & ? & ? \\ ? & 3 & 1 & ? & 2 & 2 & 2 \\ ? & ? & 3 & ? & 1 & 5 \end{bmatrix} \text{ users}$

movies

Y may be assumed to have low rank;
research shows that only a few factors affect users' preferences.

Low-Rank Matrix Completion

• Problem: given $\{y_{ij}\}_{(i,j)\in\Omega}$, Ω and a positive integer k, solve

$$\min_{\mathbf{A}\in\mathbb{R}^{m\times k},\mathbf{B}\in\mathbb{R}^{k\times n}} \sum_{(i,j)\in\Omega} |y_{ij} - [\mathbf{AB}]_{ij}|^2$$

- ALS can be applied; more tedious to write out the LS solutions than the previous matrix factorization problem but not any harder in principle
- supposingly a very difficult problem, but
- methods like ALS were found to work by means of empirical studies
- recent theoretical research suggests that matrix completion may not be that hard under some assumptions, e.g., ALS can give good results [Sun-Luo'16]

Low-Rank Matrix Completion

- an ALS alternative to matrix completion (easier to program):
 - consider an equivalent reformulation of the matrix completion problem

$$\min_{\mathbf{A}\in\mathbb{R}^{m\times k},\mathbf{B}\in\mathbb{R}^{k\times n},\mathbf{R}\in\mathbb{R}^{m\times n}} \|\mathbf{Y}-\mathbf{AB}-\mathbf{R}\|_{F}^{2} \text{ s.t. } r_{ij}=0, \ (i,j)\in\Omega$$

- do alternating optimization

$$\mathbf{A}^{(i+1)} = \arg\min_{\mathbf{A}\in\mathbb{R}^{m\times k}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}^{(i)} - \mathbf{R}^{(i)}\|_{F}^{2}$$
$$\mathbf{B}^{(i+1)} = \arg\min_{\mathbf{B}\in\mathbb{R}^{k\times n}} \|\mathbf{Y} - \mathbf{A}^{(i+1)}\mathbf{B} - \mathbf{R}^{(i)}\|_{F}^{2}$$
$$\mathbf{R}^{(i+1)} = \arg\min_{\mathbf{R}\in\mathbb{R}^{m\times n}} \|\mathbf{Y} - \mathbf{A}^{(i+1)}\mathbf{B}^{(i+1)} - \mathbf{R}\|_{F}^{2} \text{ s.t. } r_{ij} = 0, \ (i,j) \in \Omega$$

the first two are LS as before; the third has a closed form

$$r_{ij}^{(i+1)} = \begin{cases} 0, & (i,j) \in \Omega\\ [\mathbf{Y} - \mathbf{A}^{(i+1)} \mathbf{B}^{(i+1)}]_{i,j}, & (i,j) \notin \Omega \end{cases}$$

Toy Demonstration of Low-Rank Matrix Completion



Left: An incomplete image with 40% missing pixels. Right: the matrix completion result of the algorithm shown on last page. k = 120.

Part III-B: Other Extensions

Beyond LS

• let $\bar{\mathbf{a}}_i \in \mathbb{R}^n$ denote the *i*th row of **A**. The LS problem can be represented as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^m \ell(\bar{\mathbf{a}}_i^T \mathbf{x} - y_i)$$

where $\ell(z) = |z|^2$ denotes the loss function for measuring the badness of fit

- Question: why don't we use other loss functions?
 - we can indeed use other loss functions, such as
 - * 1-norm loss: $\ell(z) = |z|$
 - * Huber loss: $\ell(z) = \begin{cases} \frac{1}{2}|z|^2, & |z| \le 1\\ |z| \frac{1}{2}, & |z| > 1 \end{cases}$
 - * power-p loss: $\ell(z) = |z|^p$, with p < 1
 - the above loss functions are more robust against outliers, but
 - they require optimization and don't result in a clean closed-form solution as LS

Illustration of Loss Functions



Curve Fitting Example



"True" curve: the true f(x), p = 5. The points at x = -0.3 and x = 0.4 are outliers, and they do not follow the true curve. The 1-norm loss problem is solved by a convex optimization tool.

Gradient Descent

 $\bullet\,$ in LS we need to solve

$$(\mathbf{A}^T \mathbf{A}) \mathbf{x}_{\mathsf{LS}} = \mathbf{A}^T \mathbf{y},$$

and that requires $\mathcal{O}(n^3)$

- we also need to compute $\mathbf{A}^T \mathbf{A}$ and $\mathbf{A}^T \mathbf{y}$; their complexities are $\mathcal{O}(mn^2)$ and $\mathcal{O}(mn)$, resp.
- $\mathcal{O}(n^3)$ is expensive for very large n
- Question: can we have cheaper LS solutions, perhaps with some compromise of the solution accuracies?
Gradient Descent

• consider a general unconstrained optimization problem

$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$

where f is continuously differentiable

• Gradient Descent: given a starting point $\mathbf{x}^{(0)}$, do

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \mu \nabla f(\mathbf{x}^{(k-1)}), \quad k = 1, 2, \dots$$

where $\mu > 0$ is a step size

- take an optimization course to get more details! It is known that
 - for convex f and under some appropriate choice of $\mu,$ gradient descent converges to an optimal solution
 - for non-convex f and under some appropriate choice of μ , gradient descent converges to a stationary point

Gradient Descent

• gradient descent for LS:

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - 2\mu(\mathbf{A}^T \mathbf{A} \mathbf{x}^{(k-1)} - \mathbf{A}^T \mathbf{y}), \quad k = 0, 1, \dots$$

- $\bullet\,$ complexity for dense ${\bf A}\,$
 - computing $\mathbf{A}^T \mathbf{A}$ and $\mathbf{A}^T \mathbf{y}$: $\mathcal{O}(mn^2)$ and $\mathcal{O}(mn)$, resp. (same as before)

* $\mathbf{A}^T \mathbf{A}$ and $\mathbf{A}^T \mathbf{y}$ are cached for subsequent use in gradient descent

- complexity of each iteration: $\mathcal{O}(n^2)$
- ${\ensuremath{\,\bullet\,}}$ complexity for sparse ${\ensuremath{\mathbf{A}\,}}$
 - computing $\mathbf{A}^T \mathbf{y}$: $\mathcal{O}(nnz(\mathbf{A}))$
 - complexity of each iteration: $O(n + nnz(\mathbf{A}))$
 - * $\mathbf{A}^T \mathbf{A}$ is not necessarily sparse, so we do $\mathbf{A} \mathbf{x}^{(k-1)}$ and then $\mathbf{A}^T (\mathbf{A} \mathbf{x}^{(k-1)})$

Gradient Descent

- gradient descent is easy to understand, but there are better algorithms...
- further reading: the conjugate gradient method; see, e.g., https://stanford.edu/class/ee364b/lectures/conj_grad_slides.pdf

Online LS

• recall the LS formulation

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{t=1}^m |\bar{\mathbf{a}}_t^T \mathbf{x} - y_t|^2$$

- \bullet the LS we learnt is a batch process; i.e., solve one ${\bf x}$ given the whole $({\bf A}, {\bf y})$
- there are many applications where new $(\bar{\mathbf{a}}_t, y_t)$ appears as time goes, and we want the process to be adaptive or in real time; i.e., \mathbf{x} is updated with t

Incremental Gradient Descent

• consider an optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{t=1}^m f_t(\mathbf{x})$$

where every f_t is continuously differentiable

• Incremental Gradient Descent:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \mu \nabla f_t(\mathbf{x}_{t-1}), \quad t = 1, 2, \dots$$

- also called stochastic gradient descent, least mean squares (LMS) (in 70's), ...
- incremental gradient descent for LS:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + 2\mu(y_t - \bar{\mathbf{a}}_t^T \mathbf{x}_{t-1})\bar{\mathbf{a}}_t$$

Recursive LS

• Recursive LS (RLS) formulation:

$$\mathbf{x}_{t} = \arg\min_{\mathbf{x}\in\mathbb{R}^{n}} \sum_{i=1}^{t} \lambda^{t-i} |\bar{\mathbf{a}}_{i}^{T}\mathbf{x} - y_{i}|^{2}$$

where $0<\lambda\leq 1$ is a prescribed constant and is called the forgetting factor

- weigh the importance of $|\bar{\mathbf{a}}_i^T \mathbf{x} y_i|^2$ w.r.t. time t; the present is most important; distant pasts are insignificant; how much we remember the pasts depends on λ
- at first look, the RLS solution is $\mathbf{x}_t = \mathbf{R}_t^{-1} \mathbf{q}_t$, where

$$\mathbf{R}_t = \sum_{i=1}^t \lambda^{t-i} \bar{\mathbf{a}}_i \bar{\mathbf{a}}_i^T, \quad \mathbf{q}_t = \sum_{i=1}^t \lambda^{t-i} y_i \bar{\mathbf{a}}_i$$

• a recursive formula for \mathbf{x}_t can be derived by using the Woodbury matrix identity and by using the problem structures carefully

Woodbury Matrix Identity

For $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ of appropriate dimensions, we have

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1},$$

assuming that the inverses above exist.

• for the RLS problem, it is sufficient to know the special case

$$(\mathbf{A} + \mathbf{b}\mathbf{b}^T)^{-1} = \mathbf{A}^{-1} - \frac{1}{1 + \mathbf{b}^T \mathbf{A}^{-1} \mathbf{b}} \mathbf{A}^{-1} \mathbf{b}\mathbf{b}^T \mathbf{A}^{-1}$$

Recursive LS

- it can be verified that $\mathbf{R}_t = \lambda \mathbf{R}_{t-1} + \bar{\mathbf{a}}_t \bar{\mathbf{a}}_t^T$, $\mathbf{q}_t = \lambda \mathbf{q}_{t-1} + y_t \bar{\mathbf{a}}_t$
- by the Woodbury matrix identity,

$$\mathbf{R}_{t}^{-1} = (\lambda \mathbf{R}_{t-1} + \bar{\mathbf{a}}_{t} \bar{\mathbf{a}}_{t}^{T})^{-1} = \frac{1}{\lambda} \mathbf{R}_{t-1}^{-1} - \frac{1}{1 + \frac{1}{\lambda} \bar{\mathbf{a}}_{t}^{T} \mathbf{R}_{t-1}^{-1} \bar{\mathbf{a}}_{t}} (\frac{1}{\lambda} \mathbf{R}_{t-1}^{-1} \bar{\mathbf{a}}_{t}) (\frac{1}{\lambda} \mathbf{R}_{t-1}^{-1} \bar{\mathbf{a}}_{t})^{T}$$

• let
$$\mathbf{P}_t = \mathbf{R}_t^{-1}$$
 and $\mathbf{g}_t = \frac{1}{1 + \frac{1}{\lambda} \bar{\mathbf{a}}_t^T \mathbf{R}_{t-1}^{-1} \bar{\mathbf{a}}_t} (\frac{1}{\lambda} \mathbf{R}_{t-1}^{-1} \bar{\mathbf{a}}_t)$. We have

$$\mathbf{g}_{t} = \frac{1}{1 + \frac{1}{\lambda} \bar{\mathbf{a}}_{t}^{T} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t}} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})$$
$$\mathbf{P}_{t} = \frac{1}{\lambda} \mathbf{P}_{t-1} - \mathbf{g}_{t} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})^{T}$$
$$\mathbf{x}_{t} = \mathbf{P}_{t} \mathbf{q}_{t} = \mathbf{P}_{t-1} \mathbf{q}_{t-1} - \lambda \mathbf{g}_{t} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})^{T} \mathbf{q}_{t-1} + \frac{1}{\lambda} y_{t} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t} - y_{t} \mathbf{g}_{t} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})^{T} \bar{\mathbf{a}}_{t}$$
$$= \mathbf{x}_{t-1} - (\bar{\mathbf{a}}_{t}^{T} \mathbf{x}_{t-1}) \mathbf{g}_{t} + y_{t} \mathbf{g}_{t}$$

Recursive LS

• summary of the RLS recursion:

$$\mathbf{g}_{t} = \frac{1}{1 + \frac{1}{\lambda} \bar{\mathbf{a}}_{t}^{T} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t}} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})$$
$$\mathbf{P}_{t} = \frac{1}{\lambda} \mathbf{P}_{t-1} - \mathbf{g}_{t} (\frac{1}{\lambda} \mathbf{P}_{t-1} \bar{\mathbf{a}}_{t})^{T}$$
$$\mathbf{x}_{t} = \mathbf{x}_{t-1} + (y_{t} - \bar{\mathbf{a}}_{t}^{T} \mathbf{x}_{t-1}) \mathbf{g}_{t}$$

- remarks:
 - comparison with incremental gradient descent: it replaces \mathbf{g}_t with $2\mu \bar{\mathbf{a}}_t$
 - the above RLS recursion may be numerically unstable as empirical results suggested; modified RLS schemes were developed to mend this issue

References

[Stoica-Moses'97] P. Stoica and R. L. Moses, *Introduction to Spectral Analysis*, Prentice Hall, 1997.

[Aharon-Elad-Bruckstein'06] M. Aharon, M.I Elad, and A. Bruckstein, "*K*-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Image Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[Golub-Van Loan'12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd edition, JHU Press, 2012.

[Sayed-Tarighat-Khajehnouri'05] A. H. Sayed, A. Tarighat, and N. Khajehnouri. "Network-based wireless location," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 24–40, 2005.

[Turney-Pantel'10] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, pp. 141–188, 2010.

[Blei'12] D. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[Udell-Horn-Zadeh-Boyd'16] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models", *Foundations and Trends in Machine Learning*, 2016.

[Koren-Bell-Volinsky'09] B. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42 no. 8, pp. 30–37, 2009.

[Sun-Luo'16] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via non-convex factorization." *IEEE Trans. Inform. Theory,* vol. 62, no. 11, pp. 6535–6579, 2016.