Alternating LS for Matrix Factorization (cont'd)

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

Now look at (1). Let $\tilde{a}_j^T$ be the jth row of A.

$$\min_{A\in\mathbb{R}^{m\times k}} \|Y - AB^{(i)}\|_F^2 = \min_{\tilde{a}_j,\ \forall j} \sum_{j=1}^{m} \|\tilde{y}_j^T - \tilde{a}_j^T B^{(i)}\|_F^2$$

$$\Updownarrow$$

$$\min_{\tilde{a}_j} \|\tilde{y}_j^T - \tilde{a}_j^T B^{(i)}\|_F^2 \quad \forall j$$

$$\Updownarrow^{T}$$

$$\min_{\tilde{a}_j} \|\tilde{y}_j - (B^{(i)})^T \tilde{a}_j\|_2^2 \quad \forall j$$

$$\tilde{a}_j^{(i+1)} = [B^{(i)}(B^{(i)})^T]^{-1} B^{(i)} \tilde{y}_j$$

$$[\tilde{a}_j^{(i+1)}]^T = \tilde{y}_j^T (B^{(i)})^T \cdot [B^{(i)}(B^{(i)})^T]^{-1}$$

LS

$(B^{(i)})^T$ full column rank

$$A^{(i+1)} = Y(B^{(i)})^T [B^{(i)}(B^{(i)})^T]^{-1}$$

$B^{(i)}$ full row rank

Alternating LS for Matrix Factorization (cont'd)

The updates of ALS can be written as

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

- ALS is guaranteed to converge an optimal solution to $\min_{\mathbf{A},\mathbf{B}} \|\mathbf{Y} - \mathbf{AB}\|_F^2$ under some mild assumptions [2]

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

Low-Rank Matrix Completion

Aim: Given $\mathbf{Y} \in \mathbb{R}^{m \times n}$ 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, recover the missing entries of $\mathbf{Y}$

Applications: recommender system, data science, etc.

Example: Movie recommendation [3]

- $\mathbf{Y}$ records how user i likes movie j
- $\mathbf{Y}$ has lots of missing entries; A user doesn't watch all movies

$$\mathbf{Y} = \begin{array}{c} \text{movies} \\ \begin{bmatrix} 2 & 3 & 1 & ? & ? & 5 & 5 \\ 1 & ? & 4 & 2 & ? & ? & ? \\ ? & 3 & 1 & ? & 2 & 2 & 2 \\ ? & ? & ? & 3 & ? & 1 & 5 \end{bmatrix} \end{array} \ \text{users}$$

- $\mathbf{Y}$ may be assumed to have low rank;
  Research shows that only a few factors affect users' preferences

---

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

ALS alternative for Low-Rank Matrix Completion

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

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

An ALS alternative for matrix completion:[4]

- Consider an equivalent reformulation of the problem

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

[4] 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.

ALS alternative for Low-Rank Matrix Completion (cont'd)

- Do alternating optimization according to the equivalent problem

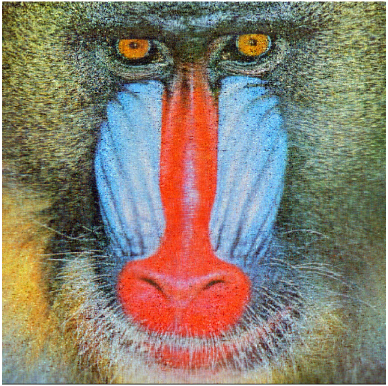
$$
\begin{aligned}
\mathbf{A}^{(i+1)} &= \arg \min_{\mathbf{A} \in \mathbb{R}^{m \times k}} \|\mathbf{Y} - \mathbf{A}\mathbf{B}^{(i)} - \mathbf{R}^{(i)}\|_F^2 \\
\mathbf{B}^{(i+1)} &= \arg \min_{\mathbf{B} \in \mathbb{R}^{k \times n}} \|\mathbf{Y} - \mathbf{A}^{(i+1)}\mathbf{B} - \mathbf{R}^{(i)}\|_F^2 \\
\mathbf{R}^{(i+1)} &= \arg \min_{\substack{\mathbf{R} \in \mathbb{R}^{m \times n} \\ r_{ij}=0,\ \forall (i,j) \in \Omega}} \|\mathbf{Y} - \mathbf{A}^{(i+1)}\mathbf{B}^{(i+1)} - \mathbf{R}\|_F^2
\end{aligned}
$$

- The first two equations can be solved via LS as before

- The third equation has the closed-form solution

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

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. $k = 120$.

Beyond LS

LS $\min_x \|y - Ax\|_2^2$

$\min_x \sum_{i=1}^{m} (y_i - \tilde{a}_i^T x)^2$

$y_i - \tilde{a}_i^T x$ is the ith element of $y - Ax$

- let $\tilde{\mathbf{a}}_i^T \in \mathbb{R}^{1 \times n}$ denote the ith row of $\mathbf{A}$

  The LS problem can be rewritten as

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

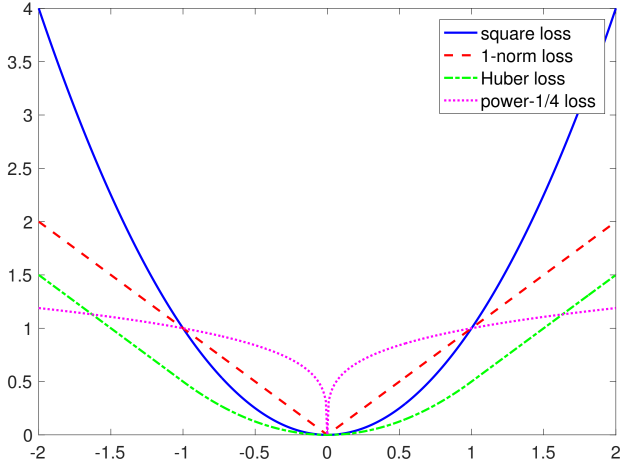
  where $\ell(z) = |z|^2$ is a loss function for measuring the badness of fit

- We can indeed use other loss functions such as
  - 1-norm loss: $\ell(z) = |z|$
  - Huber loss: $\ell(z) = \begin{cases} \frac{1}{2}|z|^2, & |z| \leq 1 \\ |z| - \frac{1}{2}, & |z| > 1 \end{cases}$
  - power-p loss: $\ell(z) = |z|^p$, with $p < 1$

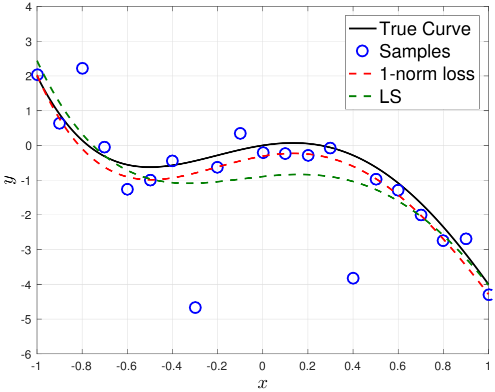
- The above loss functions are more robust against outliers

- However, they require optimization and don't result in a clean closed-form solution as LS

Illustration of Loss Functions



Example of Curve Fitting



"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.

Cheaper LS Solution

Recall that LS requires to solve the normal equation

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

Complexity: $\mathcal{O}(n^3)$

- We also need to compute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{y}$, whose complexities are $\mathcal{O}(mn^2)$ and $\mathcal{O}(mn)$, respectively

$\mathcal{O}(n^3)$ is expensive for very large n

We may acquire computationally less expensive LS solutions, with compromise of solution accuracy

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

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

where $\mu > 0$ is a step size

Convergence results:

- For convex f and with proper μ, gradient descent converges to an optimal solution
- For non-convex f and with proper μ, gradient descent converges to a stationary point

Gradient Descent (cont'd)

$f(x) = \|y - Ax\|_2^2$

Gradient descent for LS:

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \underbrace{2\mu}_{w}\underbrace{(\mathbf{A}^T\mathbf{A}\mathbf{x}^{(k-1)} - \mathbf{A}^T\mathbf{y})}_{\nabla f(x^{(k-1)})}, \quad k = 0, 1, \ldots$$

Complexity for dense **A**:

- Computing $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{y}$: $\mathcal{O}(mn^2)$ and $\mathcal{O}(mn)$ (same as before)
  - $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{y}$ are cached for subsequent use
- Each iteration: $\mathcal{O}(n^2)$

Complexity for sparse **A**:

- Computing $\mathbf{A}^T\mathbf{y}$: $\mathcal{O}(nnz(\mathbf{A}))$
- Each iteration: $\mathcal{O}(n + nnz(\mathbf{A}))$
  - $\mathbf{A}^T\mathbf{A}$ is not necessarily sparse, so we do $\mathbf{A}\mathbf{x}^{(k-1)}$ and then $\mathbf{A}^T(\mathbf{A}\mathbf{x}^{(k-1)})$

More advanced optimization methods can be applied (e.g., conjugate gradient method)

Online LS

Recall the LS formulation

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

Originally, the solving of LS is a batch process, i.e., solve one $\mathbf{x}$ given the whole $(\mathbf{A}, \mathbf{y})$

In many applications, each $(\tilde{\mathbf{a}}_t, y_t)$ comes as time t goes
We want the solving process to be adaptive/in real time

Incremental Gradient Descent for Online LS

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:

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

- Also called stochastic gradient descent, least mean squares (LMS) (in 70's)

Incremental gradient descent for LS:

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

- At each time t, only need the last iterate $\mathbf{x}_{t-1}$ and the current data $(\tilde{\mathbf{a}}_t, y_t)$

Recursive LS

Recursive LS (RLS) formulation:

$$\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{t} \lambda^{t-i} |\tilde{\mathbf{a}}_i^T \mathbf{x} - y_i|^2$$

where $0 < \lambda \leq 1$ is prescribed, called the forgetting factor

- Weigh the importance of $|\tilde{\mathbf{a}}_i^T \mathbf{x} - y_i|^2$ w.r.t. time t: The present is most important while distant pasts are insignificant
- How much we remember the past depends on λ

At first look, the RLS solution is $\mathbf{x}_t = \mathbf{R}_t^{-1}\mathbf{q}_t$ (assume $\mathbf{R}_t$ nonsingular), where

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

$\mathbf{x}_t$ can be derived recursively by using the Woodbury matrix identity and exploiting the problem structures

Woodbury Matrix Identity

For $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with proper sizes,

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

assuming that the inverses above exist

For the RLS problem, it is sufficient to consider the special case

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

$B = -b$, b column vector

$D = 1$

$C = b^T$

Recursive LS

It can be verified that

$$\mathbf{R}_t = \lambda \mathbf{R}_{t-1} + \tilde{\mathbf{a}}_t \tilde{\mathbf{a}}_t^T, \quad \mathbf{q}_t = \lambda \mathbf{q}_{t-1} + y_t \tilde{\mathbf{a}}_t$$

Using the Woodbury matrix identity,

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

Let $\mathbf{P}_t = \mathbf{R}_t^{-1}$ and $\mathbf{g}_t = \dfrac{1}{1 + \frac{1}{\lambda}\tilde{\mathbf{a}}_t^T \mathbf{R}_{t-1}^{-1} \tilde{\mathbf{a}}_t}(\frac{1}{\lambda}\mathbf{R}_{t-1}^{-1}\tilde{\mathbf{a}}_t)$. Then,

$$\mathbf{g}_t = \frac{1}{1 + \frac{1}{\lambda}\tilde{\mathbf{a}}_t^T \mathbf{P}_{t-1} \tilde{\mathbf{a}}_t}(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t), \; (*) \quad \mathbf{P}_t = \tfrac{1}{\lambda}\mathbf{P}_{t-1} - \mathbf{g}_t(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t)^T$$

$$\begin{aligned}\mathbf{x}_t = \mathbf{P}_t \mathbf{q}_t &= \underbrace{\mathbf{P}_{t-1}\mathbf{q}_{t-1}}_{x_{t-1}} - \lambda \mathbf{g}_t(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t)^T \mathbf{q}_{t-1} + \tfrac{1}{\lambda} y_t \mathbf{P}_{t-1}\tilde{\mathbf{a}}_t - y_t \boxed{\mathbf{g}_t(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t)^T \tilde{\mathbf{a}}_t} \\ &= \mathbf{x}_{t-1} - (\tilde{\mathbf{a}}_t^T \mathbf{x}_{t-1})\mathbf{g}_t + y_t \mathbf{g}_t\end{aligned}$$

$$(*) \Leftrightarrow g_t \cdot \frac{1}{\lambda} \tilde{a}_t^T P_{t-1} \tilde{a}_t = \frac{1}{\lambda} P_{t-1} \tilde{a}_t - g_t$$

Recursive LS

RLS recursion:

$$
\begin{aligned}
\mathbf{g}_t &= \frac{1}{1 + \frac{1}{\lambda}\tilde{\mathbf{a}}_t^T \mathbf{P}_{t-1}\tilde{\mathbf{a}}_t}\left(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t\right) \\
\mathbf{P}_t &= \tfrac{1}{\lambda}\mathbf{P}_{t-1} - \mathbf{g}_t\left(\tfrac{1}{\lambda}\mathbf{P}_{t-1}\tilde{\mathbf{a}}_t\right)^T \\
\mathbf{x}_t &= \mathbf{x}_{t-1} + (y_t - \tilde{\mathbf{a}}_t^T \mathbf{x}_{t-1})\mathbf{g}_t
\end{aligned}
$$

Remarks:

- It replaces the term $2\mu\tilde{\mathbf{a}}_t$ in incremental gradient descent with $\mathbf{g}_t$
- The RLS recursion may be numerically unstable as empirical results suggested. Modified RLS schemes were developed to mend this issue

Matrix Computations
Chapter 4: Eigenvalues, Eigenvectors, and Eigendecomposition
Section 4.1 Eigendecomposition

Jie Lu
ShanghaiTech University

Eigenvalues and Eigenvectors

Definition: Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$). If there exists $\mathbf{v} \in \mathbb{C}^n, \mathbf{v} \neq 0$ s.t.

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v} \qquad \text{for some } \lambda \in \mathbb{C}, \tag{$*$}$$

then we say $\mathbf{v}$ is a (right) eigenvector associated with eigenvalue λ of $\mathbf{A}$.

- In general, $\mathbf{Ax}$ differs from $\mathbf{x}$ in magnitude and direction. However, if $\mathbf{x}$ is an eigenvector of $\mathbf{A}$ and $\mathbf{A}, \mathbf{x}$ are real, then $\mathbf{Ax}$ and $\mathbf{x}$ are parallel
- $(*)$ is called an eigenvalue problem or eigen-equation
- Any solution $(\mathbf{v}, \lambda)$ to $(*)$ is called an eigen-pair of $\mathbf{A}$
- If $(\mathbf{v}, \lambda)$ is an eigen-pair of $\mathbf{A}$, $(\alpha\mathbf{v}, \lambda)$ for any $\alpha \in \mathbb{C}, \alpha \neq 0$ is also an eigen-pair of $\mathbf{A}$ $\quad A(\alpha v) = \alpha(Av) = \alpha(\lambda v) = \lambda(\alpha v)$
- If there exists a **row** vector $\mathbf{w}$, $\mathbf{w} \neq 0$ s.t. $\mathbf{wA} = \lambda\mathbf{w}$ for some $\lambda \in \mathbb{C}$, we say $\mathbf{w}$ is a left eigenvector associated with eigenvalue λ of $\mathbf{A}$

Characteristic Polynomial

Every $\mathbf{A} \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$) has n (possibly repeated) eigenvalues

- From the eigenvalue problem,

$$\begin{aligned} \mathbf{Av} = \lambda \mathbf{v} \text{ for some } \mathbf{v} \neq \mathbf{0} \quad &\Longleftrightarrow \quad (\lambda \mathbf{I} - \mathbf{A})\mathbf{v} = \mathbf{0} \text{ for some } \mathbf{v} \neq \mathbf{0} \\ &\Longleftrightarrow \quad \mathbf{v} \in \text{Null}(\lambda \mathbf{I} - \mathbf{A}) \text{ for some } \mathbf{v} \neq \mathbf{0} \\ &\Longleftrightarrow \quad \det(\lambda \mathbf{I} - \mathbf{A}) = 0 \end{aligned}$$

- $p(\lambda) := \det(\lambda \mathbf{I} - \mathbf{A})$ is called the characteristic polynomial of $\mathbf{A}$ (The characteristic polynomial can also be defined to be $\det(\mathbf{A} - \lambda \mathbf{I})$, which differs from $p(\lambda)$ by a sign $(-1)^n$)
- $p(\lambda) = 0 \Longleftrightarrow \lambda$ is an eigenvalue of $\mathbf{A}$

  If A is real, $\alpha_0, \cdots, \alpha_{n-1}$ are real
- It can be shown that $p(\lambda)$ is a polynomial of degree n, i.e., $p(\lambda) = \alpha_0 + \alpha_1 \lambda + \alpha_2 \lambda^2 + \ldots + \alpha_n \lambda^n$ where α_i's depend on $\mathbf{A}$ and in fact, $\alpha_n = 1$
- Therefore, $p(\lambda)$ has n possibly repeated roots, which are the n eigenvalues of $\mathbf{A}$
- $p(\lambda)$ can be factored as $p(\lambda) = \prod_{i=1}^{n} (\lambda - \lambda_i)$, where $\lambda_1, \ldots, \lambda_n$ are the roots of $p(\lambda)$

  Any vector in $\text{Null}(\lambda I - A)$ except 0 is
- Given an eigenvalue λ of $\mathbf{A}$, $\text{Null}(\lambda \mathbf{I} - \mathbf{A})$ is called the eigenspace of $\mathbf{A}$ associated with λ

  an eigenvector associated with λ

Complex Eigenvalues and Eigenvectors

An eigenvalue can be complex even if **A** is real

- A polynomial $p(\lambda) = \alpha_0 + \alpha_1\lambda + \alpha_2\lambda^2 + \ldots + \alpha_n\lambda^n$ with real coefficients α_i's can have complex roots
- **Example**: Consider

$$\mathbf{A} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

  - $p(\lambda) = \lambda^2 + 1$, so $\lambda_1 = \boldsymbol{j}$, $\lambda_2 = -\boldsymbol{j}$

$$AV_R + jAV_I$$
$$\| $$
$$A(V_R + jV_I)$$
$$= \lambda V_R + j\lambda V_I$$
$$\Leftrightarrow AV_R = \lambda V_R$$
$$AV_I = \lambda V_I$$

If **A** is real and there exists a real eigenvalue λ of **A**, the associated eigenvector **v** can be taken as real

- When $\lambda\mathbf{I} - \mathbf{A}$ is real, we can define $\mathcal{N}(\lambda\mathbf{I} - \mathbf{A})$ on $\mathbb{R}^n$
- If **v** is a complex eigenvector of a real **A** associated with a real λ, we can write $\mathbf{v} = \mathbf{v}_\mathrm{R} + \boldsymbol{j}\mathbf{v}_\mathrm{I}$, where $\mathbf{v}_\mathrm{R}, \mathbf{v}_\mathrm{I} \in \mathbb{R}^n$. We can verify that both of $\mathbf{v}_\mathrm{R}$ and $\mathbf{v}_\mathrm{I}$ are eigenvectors associated with λ