FUNDAMENTALS OF LEARNING AND INFORMATION PROCESSING

SESSION 03: LINEAR ALGEBRA (III)

Dr Gang Li

Deakin University, Geelong, Australia

2018-11-06

Matrix Factorization (IV): SVD for *arbitrary* Matrix **3**
The Geometry of Linear Transformation 4
SVD: *Singular Value Decomposition* 7
SVD Application (I): *Data Compression* 10
SVD Application (II): *Principle Component Analysis* 13
SVD Application (II-b): *Fisher Linear Discriminant* (Binary) 18
SVD Application (II-c): *Fisher Linear Discriminant* (Multiple) 23
SVD Application (III): *Least Squares Approximations* 28

Matrix Factorization (V): Others **31**
Matrix Factorization: Revisit 32
NMF: *Nonnegative Matrix Factorization* 33

References **34**

Table of Content

Matrix Factorization (IV): SVD for *arbitrary* Matrix
The Geometry of Linear Transformation
SVD: *Singular Value Decomposition*
SVD Application (I): *Data Compression*
SVD Application (II): *Principle Component Analysis*
SVD Application (II-b): *Fisher Linear Discriminant* (Binary)
SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)
SVD Application (III): *Least Squares Approximations*

Matrix Factorization (V): Others
Matrix Factorization: Revisit
NMF: *Nonnegative Matrix Factorization*

References

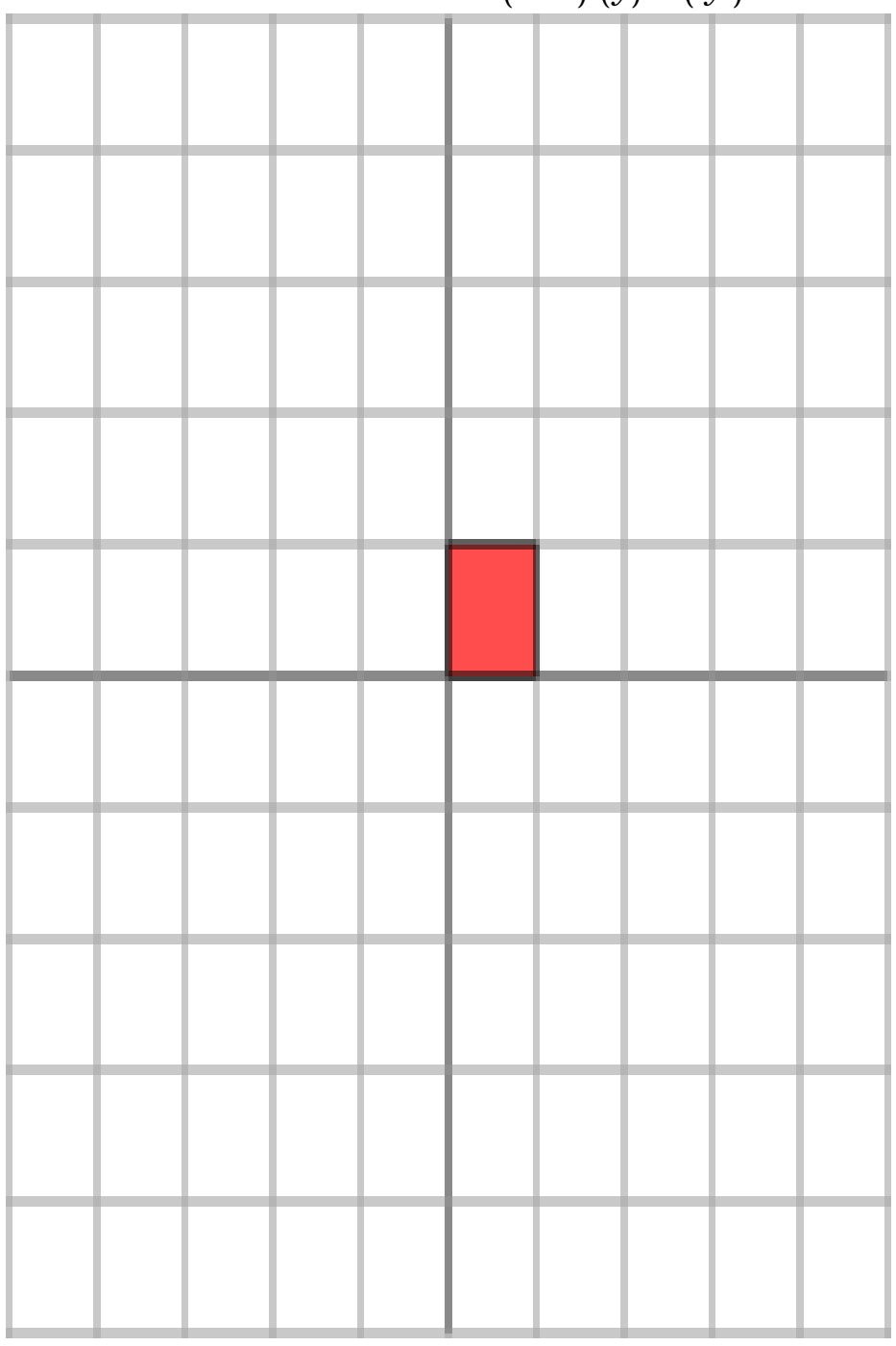
Matrix Factorization (IV): SVD for *arbitrary* Matrix

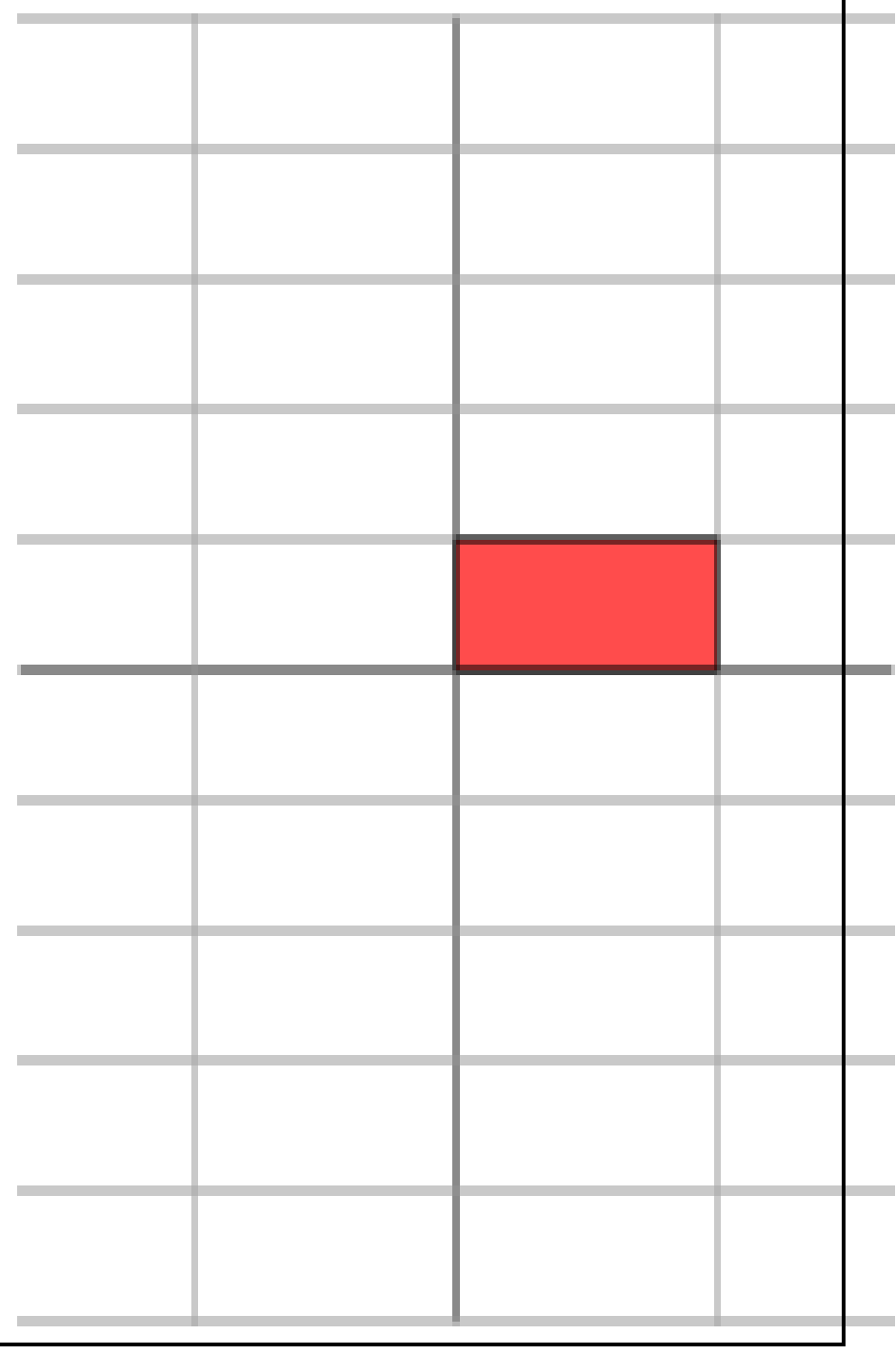
The Geometry of Linear Transformation

Defn

Any *matrix hit* operation can be considered as a linear transformation that takes a point (x, y) in the plane and transforms it into another point using matrix multiplication:

- Consider one *diagonal* matrix: $\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3x \\ y \end{pmatrix}$



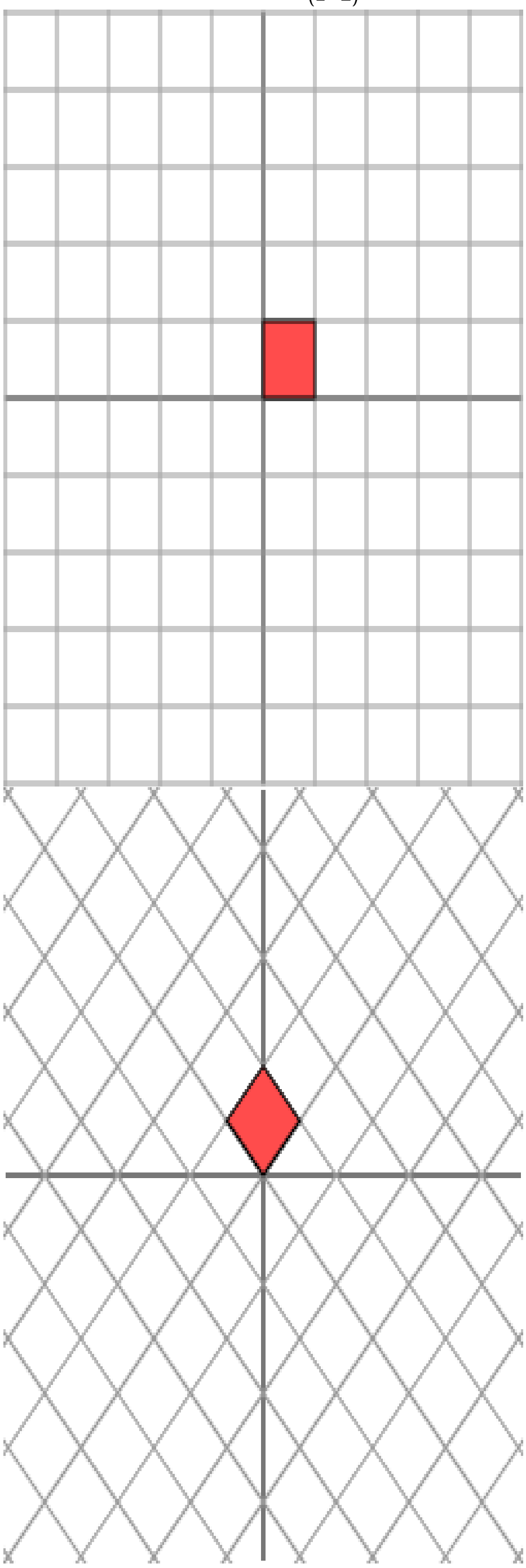


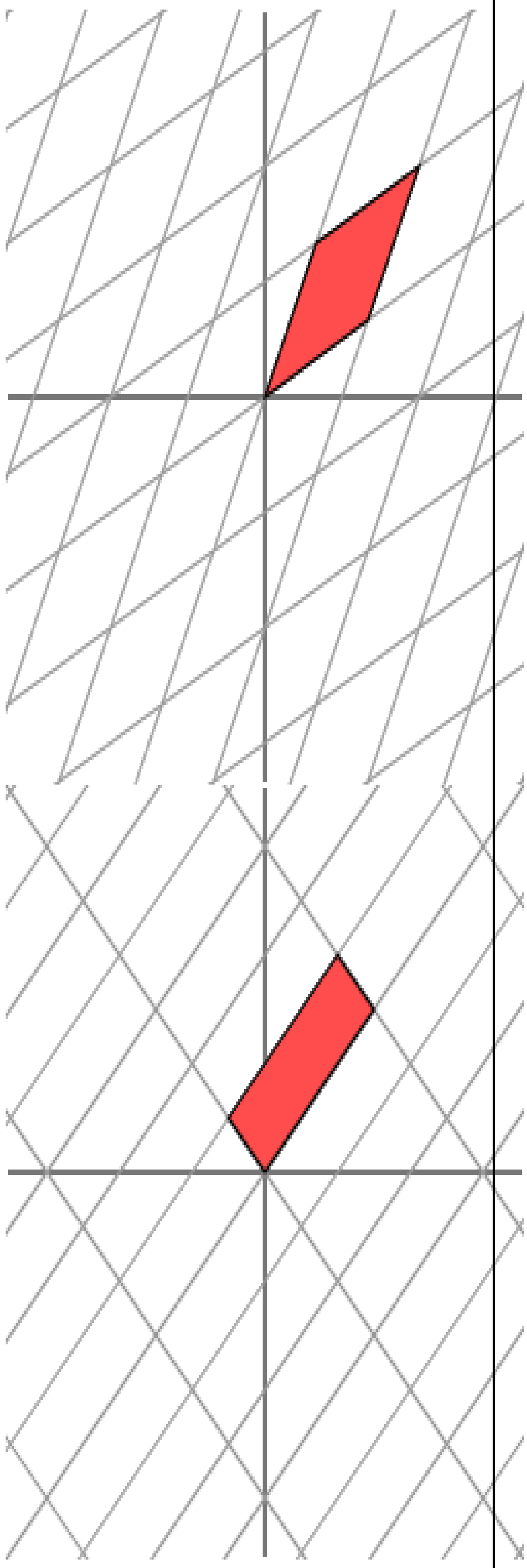
The Geometry of Linear Transformation

Defn

Any *matrix hit* operation can be considered as a linear transformation that takes a point (x, y) in the plane and transforms it into another point using matrix multiplication:

- Consider one *symmetric* matrix: $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$



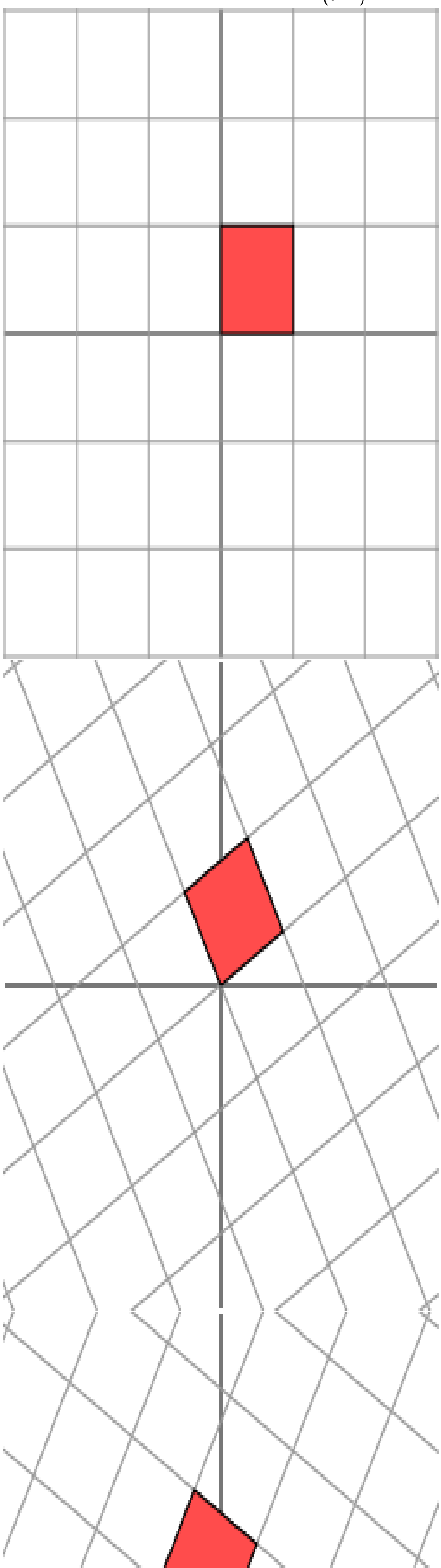


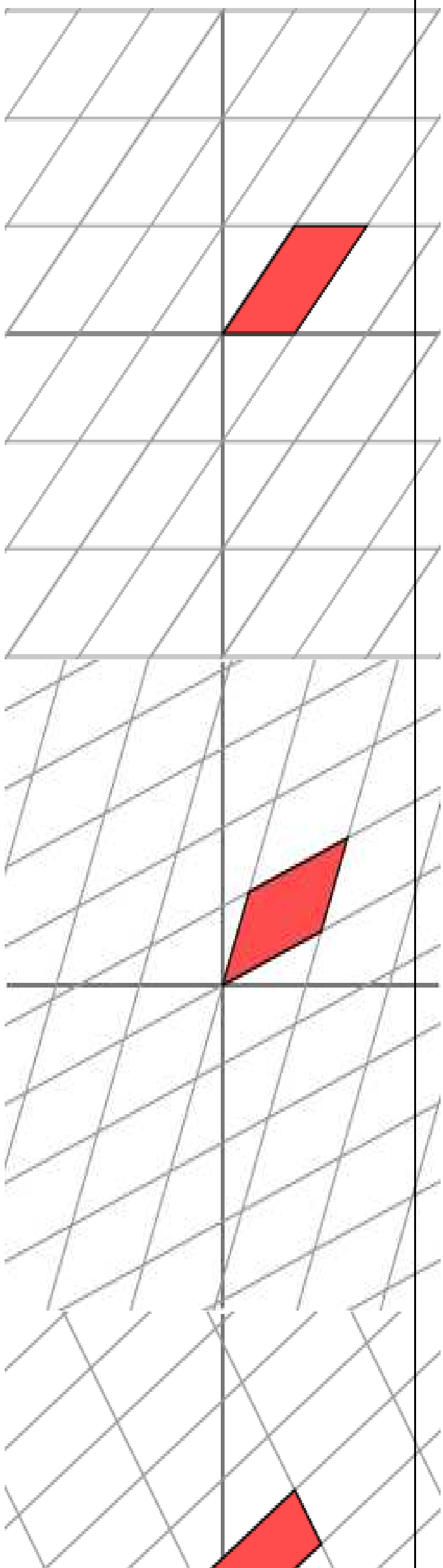
- Let's rotate our grid through a 45 degree angle into a new basis v_1, v_2.
- This new grid is transformed by the diagonal matrix: the grid is stretched by a factor of 3 in one direction: $Av_i = \lambda_i v_i$

The Geometry of Linear Transformation

Defn Any *matrix hit* operation can be considered as a linear transformation that takes a point (x, y) in the plane and transforms it into another point using matrix multiplication:

- Consider one *arbitrary* matrix for *shear*: $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$





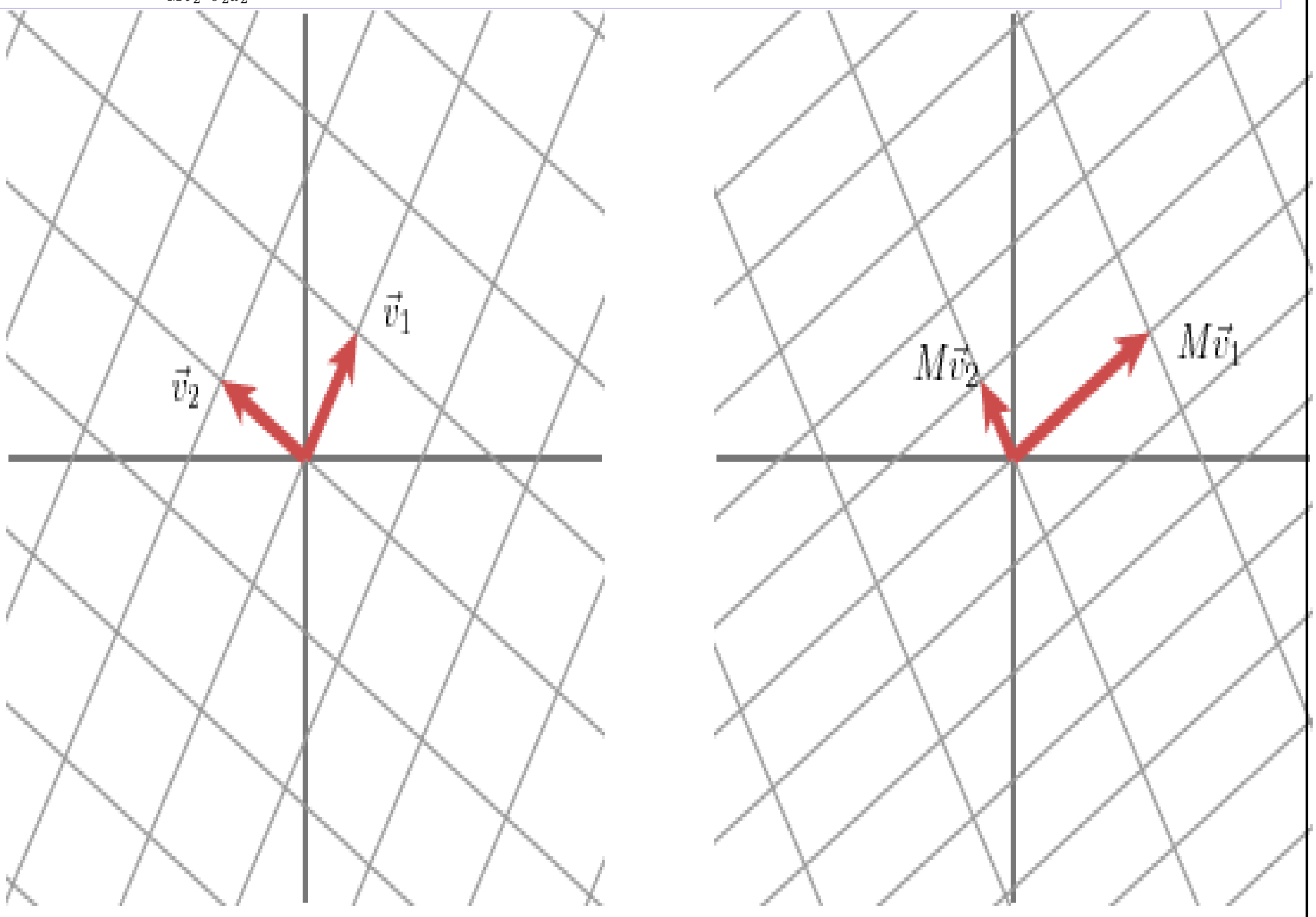
SVD: *Singular Value Decomposition*

2D

Find an appropriate orthogonal unit vectors v_1 and v_2, so that after the linear transformation by matrix M, vectors Mv_1 and Mv_2 are orthogonal. Let

- u_1 and u_2 denote unit vectors in the direction of Mv_1 and Mv_2
- The lengths of Mv_1 and Mv_2 – denoted by σ_1 and σ_2 – describe the amount that the grid is stretched in those particular directions.
- σ_1 and σ_2 are called the *singular values* of M.

Hence we have $\begin{matrix} Mv_1=\sigma_1 u_1 \\ Mv_2=\sigma_2 u_2 \end{matrix}$



- For a general vector x, based on the orthogonal unit vectors v_1 and v_2, we have $x = (v_1 \cdot x)v_1 + (v_2 \cdot x)v_2$.
  - $Mx = (v_1 \cdot x)Mv_1 + (v_2 \cdot x)Mv_2 = (v_1 \cdot x)\sigma_1 u_1 + (v_2 \cdot x)\sigma_2 u_2$
  - $Mx = (v_1^T x)\sigma_1 u_1 + (v_2^T x)\sigma_2 u_2 = \sigma_1 u_1 v_1^T x + \sigma_2 u_2 v_2^T x$
  - $M = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$
- This is also referred to as *Singular Vector Decomposition*: $M = U\Sigma V^T$, where U is an orthogonal matrix whose columns are the vectors u_1 and u_2, Σ is a *diagonal* matrix with entries σ_1 and σ_2, and V is an orthogonal matrix whose columns are v_1 and v_2.

SVD: *Singular Value Decomposition*

Defn

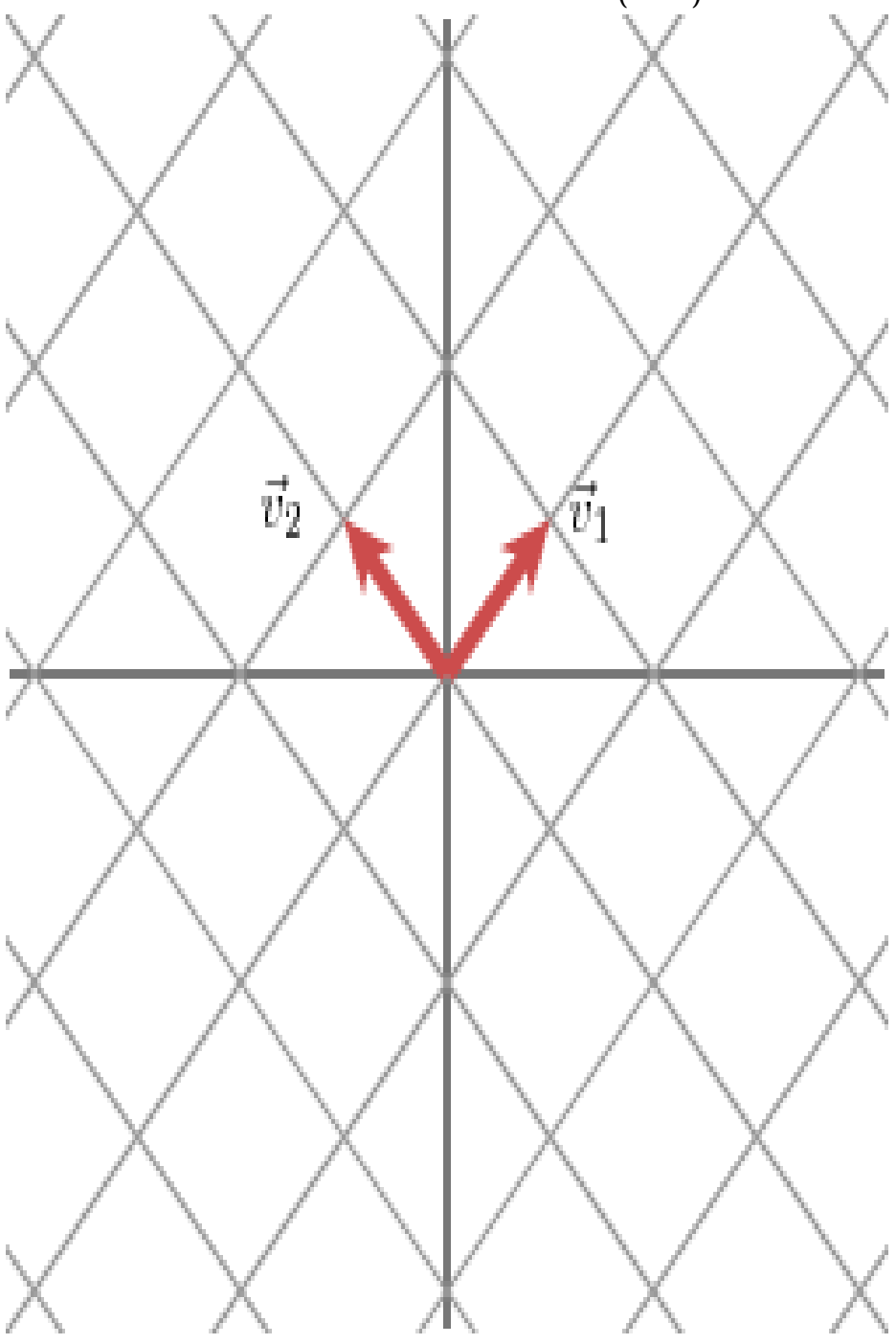
For an arbitrary matrix M, suppose its SVD is given as $M = U\Sigma V^T$, we have

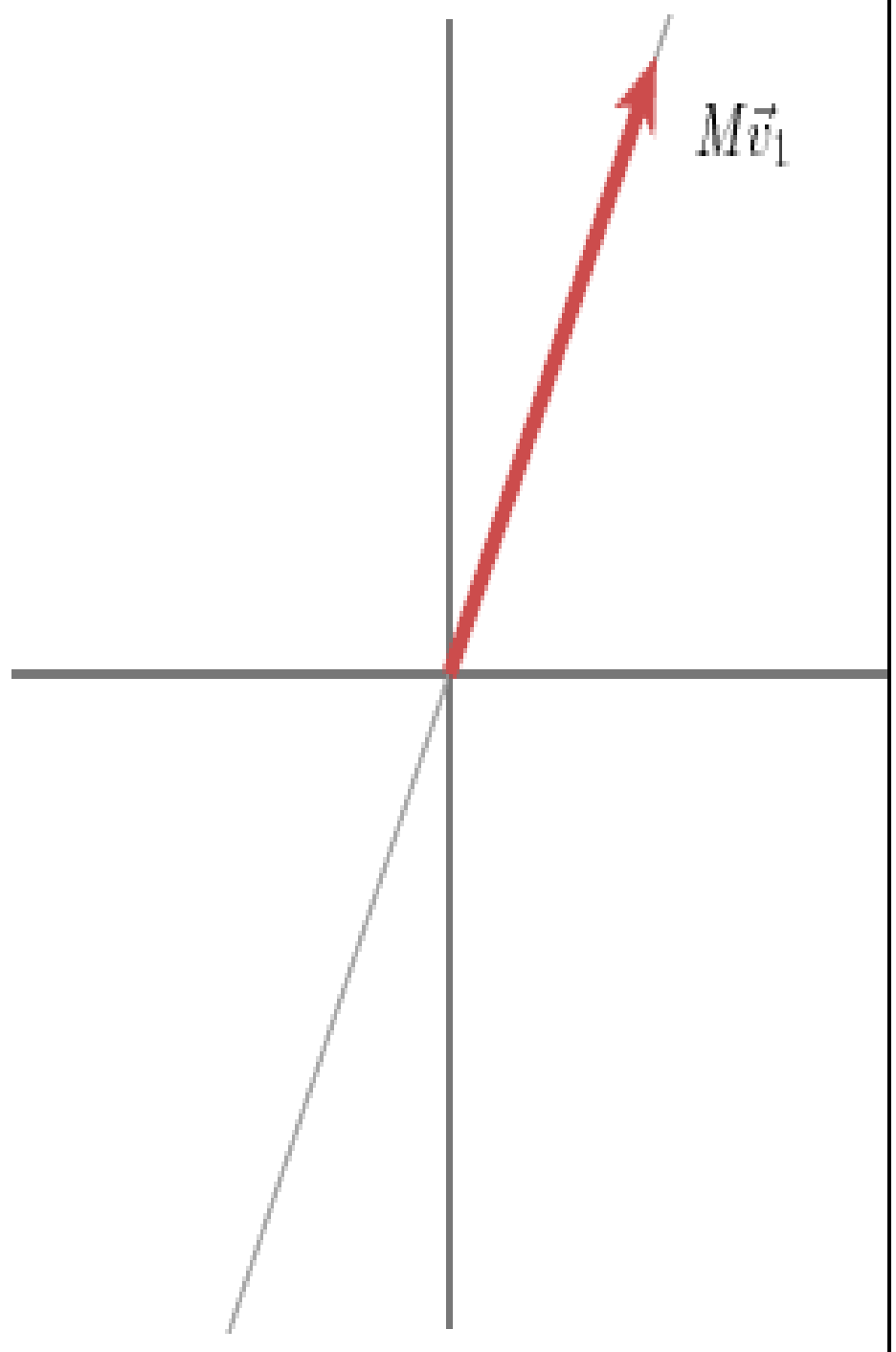
$M^T M = V\Sigma^T \Sigma V^T$ the right singular vectors (columns of V) must be the eigenvectors of the symmetric matrix $M^T M$;
$MM^T = U\Sigma\Sigma^T U^T$ the left singular vectors (columns of U) must be the eigenvectors of the symmetric matrix MM^T;

- Let v_i be an unit eigenvector of $M^T M$, the vector u_i are obtained as unit vectors in the direction of Mv_i, the singular values are then given by $\sigma_i = |Mv_i|$.
- We can prove that u_i and u_j are orthogonal:
  - $Mv_i = \sigma_i u_i$, and $Mv_j = \sigma_j u_j$
  - $Mv_i \cdot Mv_j = v_i^T M^T M v_j = v_i \cdot M^T M v_j = v_i \cdot \lambda_j v_j = 0$
  - $Mv_i \cdot Mv_j = \sigma_i \sigma_j u_i \cdot u_j = 0$.
- Therefore, u_i and u_j are orthogonal so we have found an *orthogonal* set of vectors v_i that is transformed into another *orthogonal* set u_i.

SVD: *Singular Value Decomposition*

Let's now look at the singular matrix $M = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$ with the following geometric effect:





- The second singular value $\sigma_2 = 0$ so $M = u_1 \sigma_1 v_1^T$.
- The *rank* of M is the number of non-zero singular values.

SVD Application (I): *Data Compression*

Defn For two matrix $X \in \mathscr{R}^{m \times k}$ with columns x_i, and $Y \in \mathscr{R}^{k \times n}$ with rows y_i^T, the *outer product expansion* is $XY = \sum_{i=1}^{k} x_i y_i^T$, with $x_i y_i^T$ is an *outer product* of vectors x_i and y_i.

- Let $X = \begin{pmatrix} u_1 \cdots u_k \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{pmatrix} = \begin{pmatrix} \sigma_1 u_1 & \cdots & \sigma_k u_k \end{pmatrix}$, and $Y = \begin{pmatrix} v_1^T \\ \vdots \\ v_k^T \end{pmatrix}$
- Then $M = XY$ can be expressed as an *outer product expansion*:

$$M = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

By using less k, the M can be approximated with a higher compression ratio.

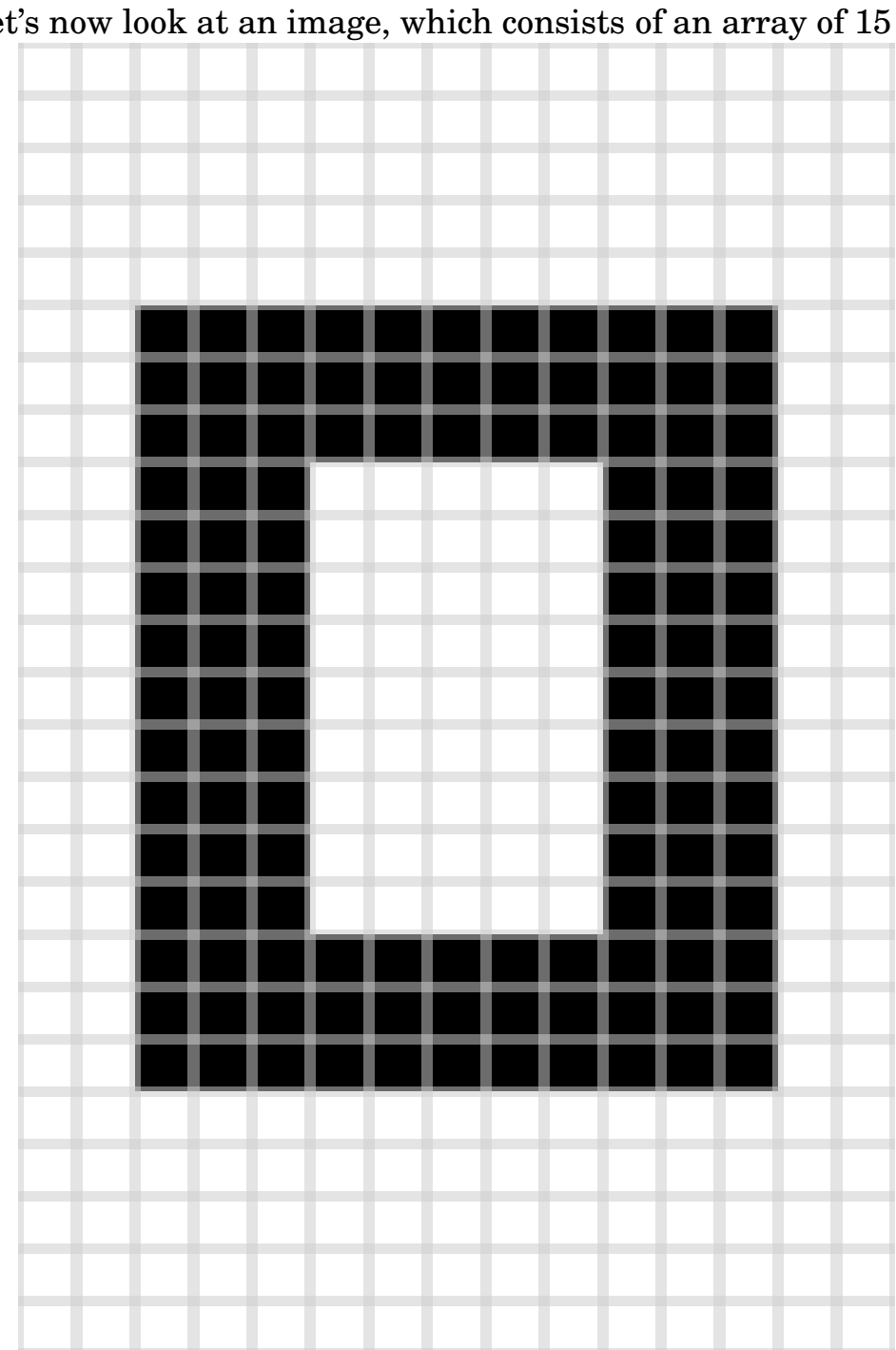
- It provides a different explanation on how M transforms an arbitrary vector x:

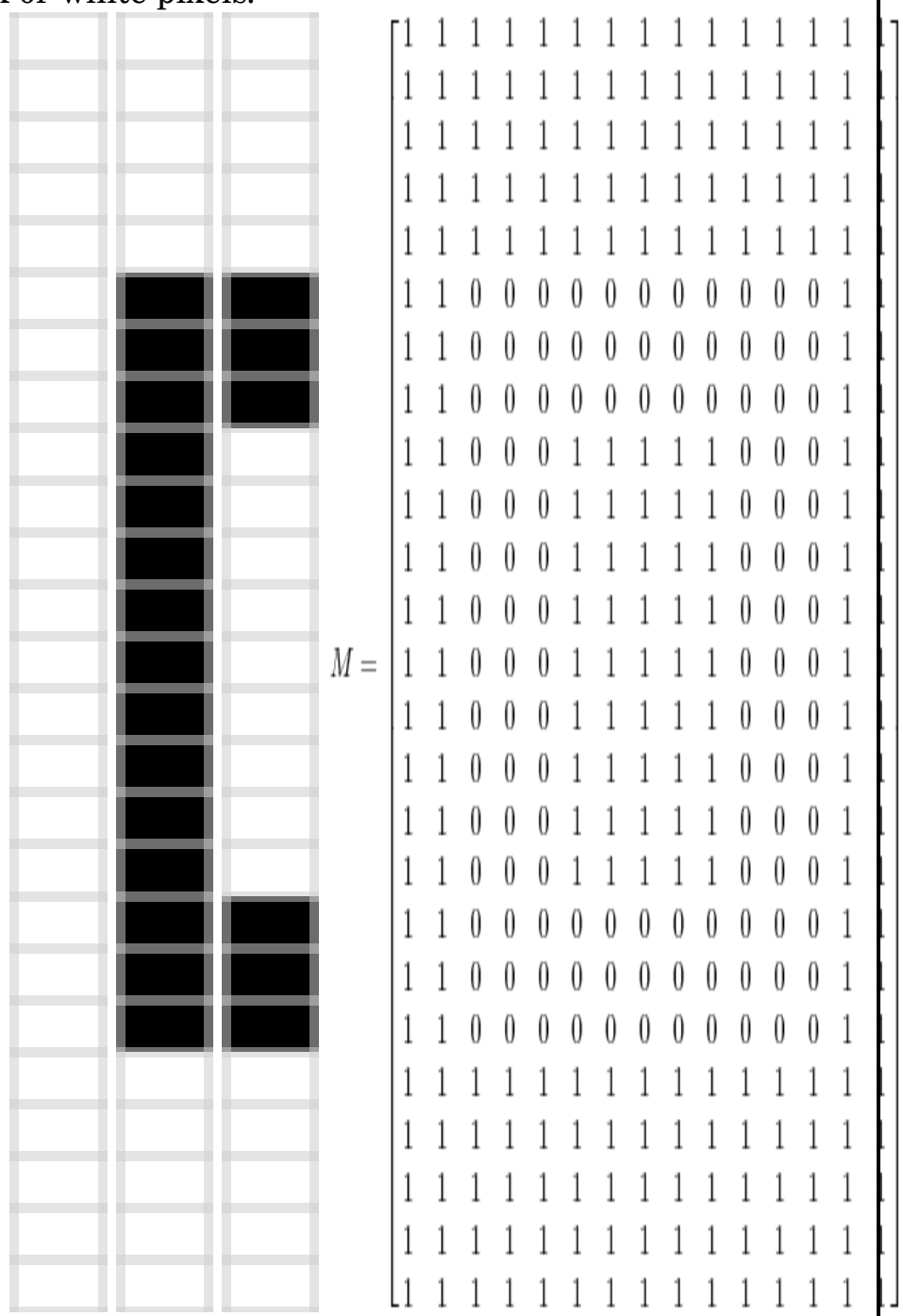
$$Mx = \sum_{i=1}^{k} \sigma_i u_i v_i^T x = \sum_{i=1}^{k} (v_i^T x) \sigma_i u_i$$

Under the action of M, each v component of x becomes a u component after scaling by the appropriate σ.

SVD Application (I): *Data Compression*

Let's now look at an image, which consists of an array of 15×25 black or white pixels.



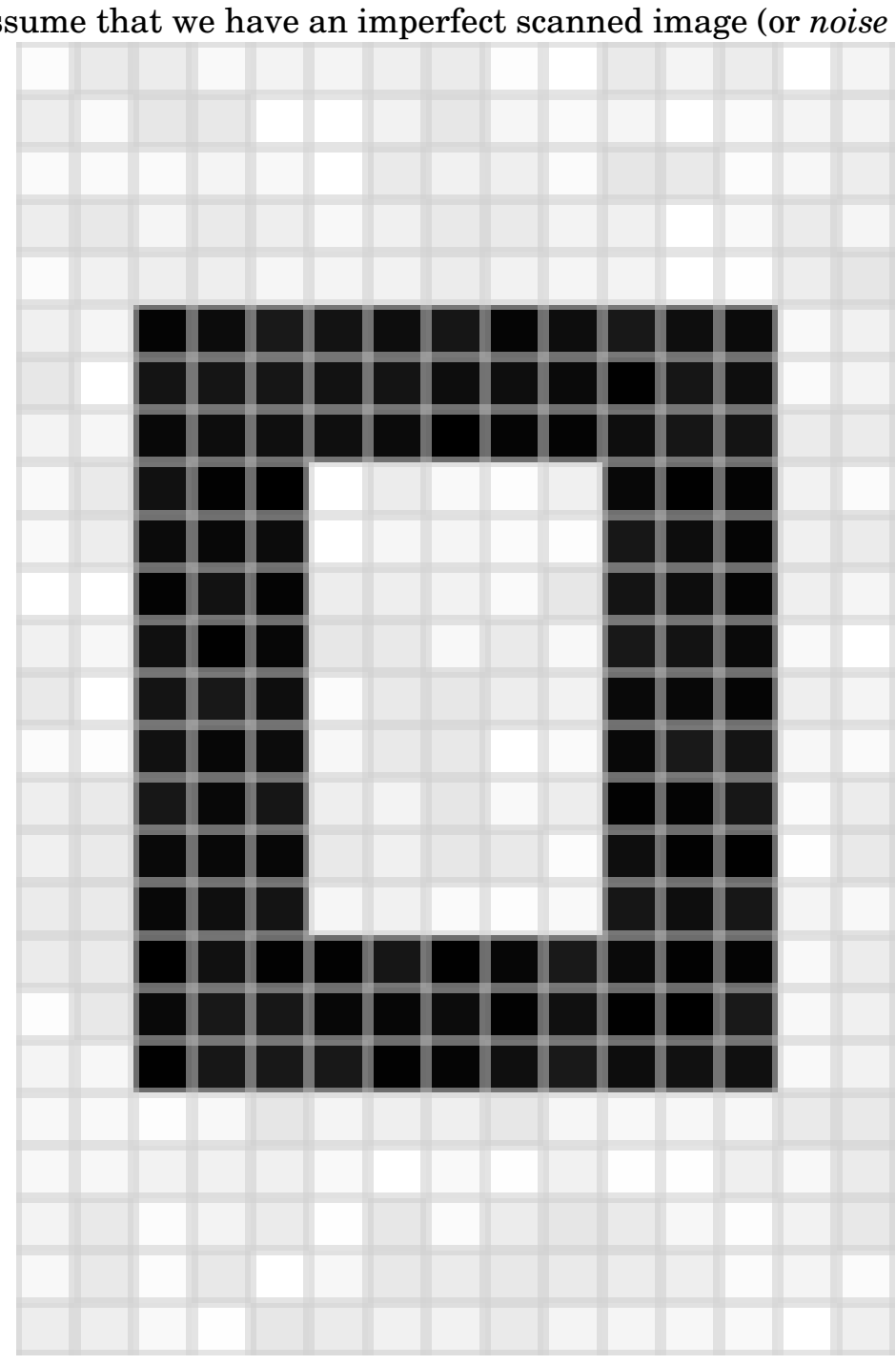


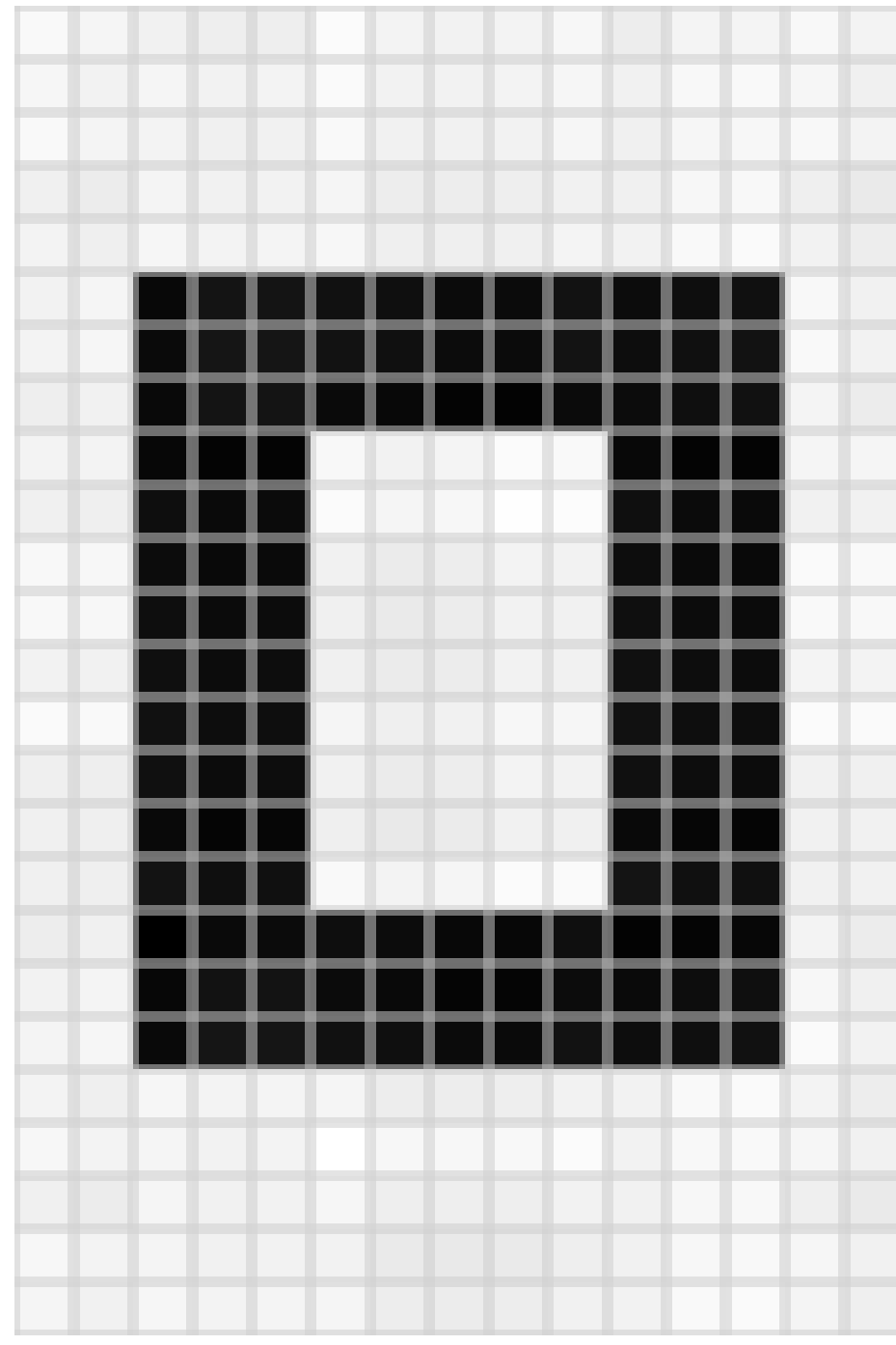
$$M = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0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

- The original matrix M has 375 entries. Performing SVD on M, results in only three non-zero singular values: $\sigma_1 = 14.72$, $\sigma_2 = 5.22$ and $\sigma_3 = 3.31$ Therefore, we have $M = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + u_3\sigma_3 v_3^T$, which can be represented using only 123 entries:
- SVD can discover and eliminate the redundancy in the matrix.

SVD Application (I): *Data Compression*

Assume that we have an imperfect scanned image (or *noise* image), we want to de-noise it.





- We apply SVD on the 15×25 matrix and get the following singular values: 14.15, 4.67, 3.00, 0.20, $\cdots$, 0.05.
- As only 3 singular values are important (bigger), we can approximate M by

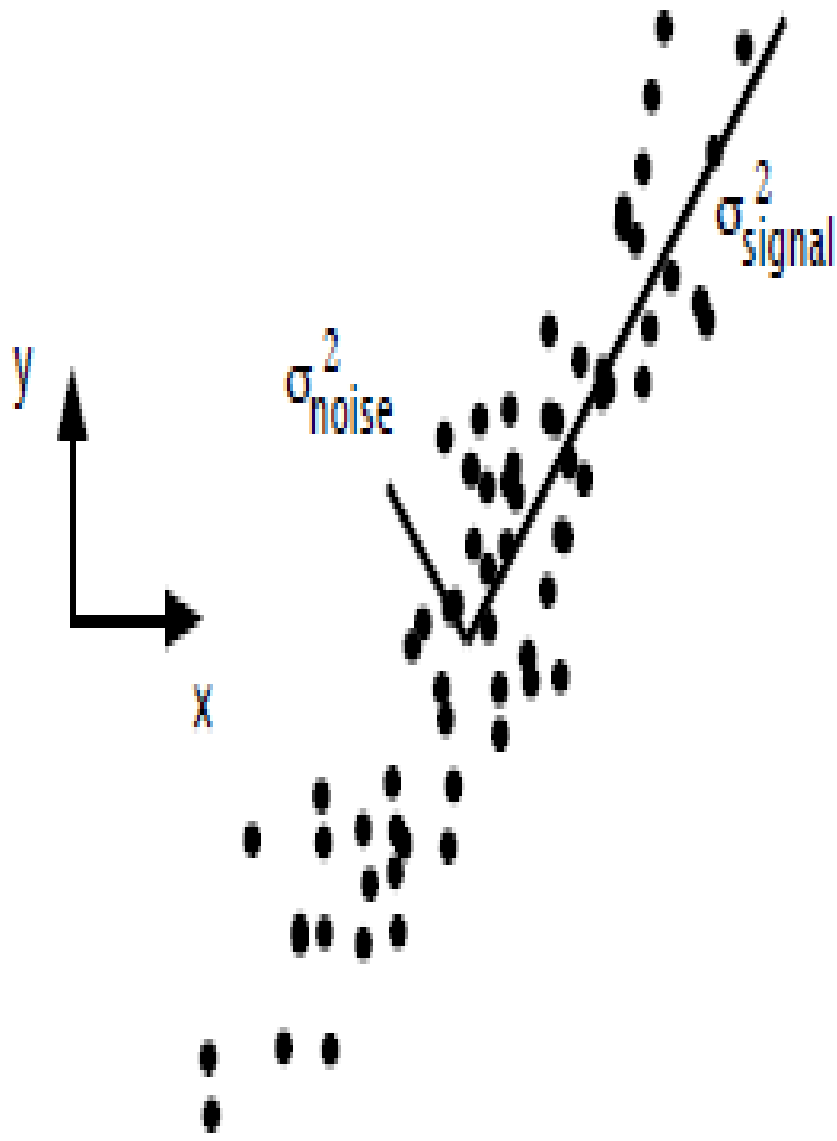
$$M = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + u_3\sigma_3 v_3^T$$

SVD Application (II): *Principle Component Analysis*

Let M be a $m \times n$ data matrix in which the *rows* represent the instances, the *columns* represent the variables. M is *centred* so that each column has zero mean.

Assume that w is a $n \times 1$ unit column vector of projection weights that result in the *largest variance* when the data M is projected along w.

- The projection of a vector x onto w is: $w^T x = \sum_{i=1}^{n} x_i w_i$
- The projection of data M along w is: Mw



As the *mean* of Mw is 0, the *variance* of Mw is:

$$\sigma_w^2 = (Mw)^T(Mw) = w^T M^T M w = w^T C w$$

PCA tries to find the unit vector w to maximize variance, subject to $w^T w = 1$.

SVD Application (II): *Principle Component Analysis*

PCA

Finding the unit vector w to maximize variance, subject to $w^T w = 1$. This is equivalent to maximizing

$$f = w^T C w - \lambda(w^T w - 1)$$

- Differentiating with respect to w yields: $\frac{\partial f}{\partial w} = 2Cw - 2\lambda w = 0$
- Hence $Cw = M^T M w = \lambda w$.
  - The best w is $w = v_1$, the the first right singular vector of M
  - and $\lambda = \sigma_1^2$
- In this case, the variance is given by

$$w^T C w = w^T \lambda w = \sigma_1^2 w^T w = \sigma_1^2$$

so the *singular value* is *the variance in the direction of the principal component*.

- Once the first principal component is found, we continue in the same fashion to look for the next one, which is orthogonal to all the principal components already found.
- The solutions are: *the right singular vectors v_k of M*, and *the variance in each direction is given by the corresponding singular values σ_k*.

SVD Application (II): *Principle Component Analysis*

PCA

For a data matrix M, *Principle Component Analysis* can be done via:

1. Center the data by subtracting the mean of each column
2. Computer SVD of the centered matrix M: $M = U\Sigma V^T$
3. The principal components are the columns of V, the coordinates of the data in the basis defined by the principal components are $U\Sigma$.

- If we check the formula for SVD: $M_{m\times n} = U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}$
  - $M_{m\times n}V_{n\times r} = U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}V_{n\times r} = U_{m\times r}\Sigma_{r\times r}$

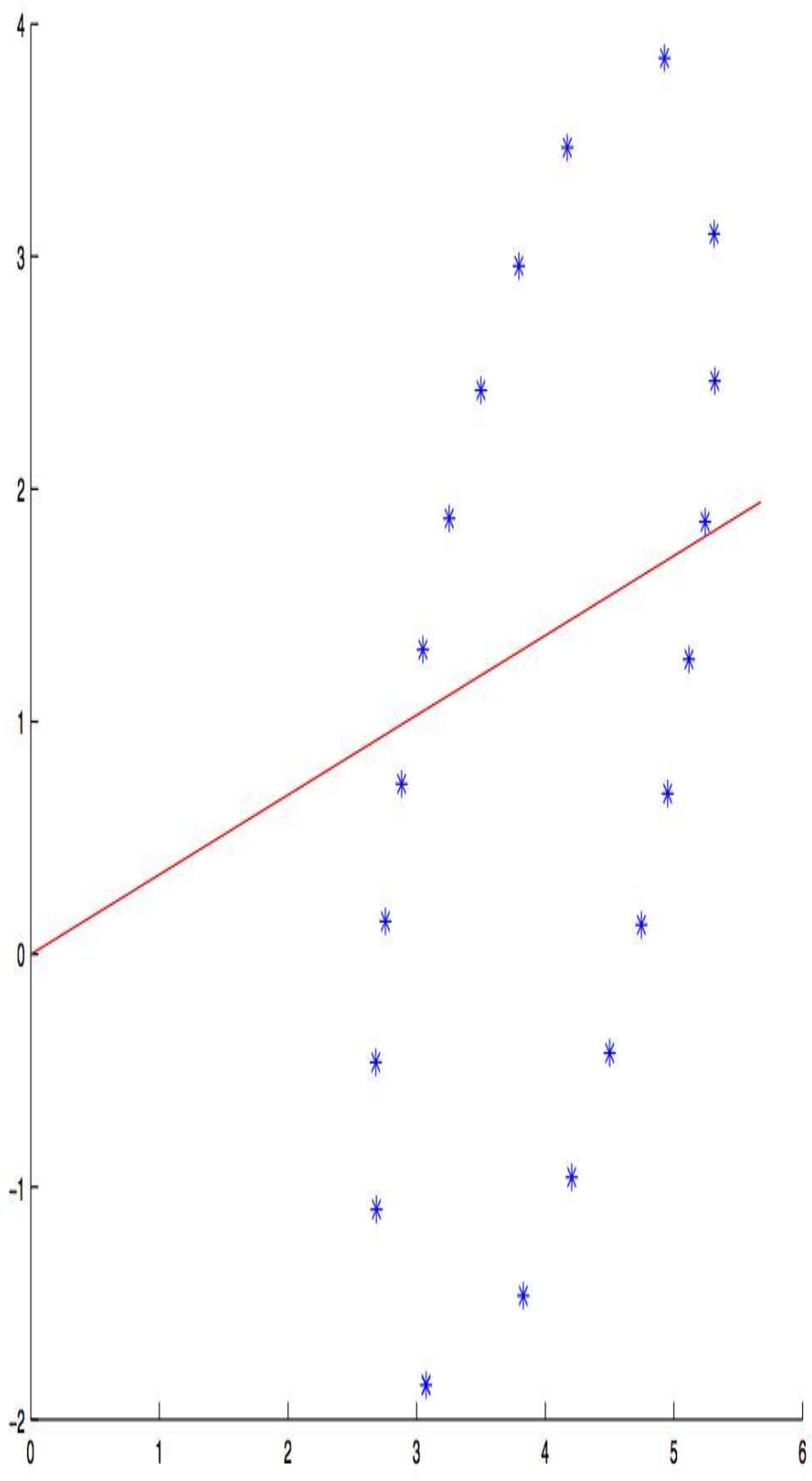
  Here $V_{n\times r}$ is a matrix, which can compress the columns of the $m \times n$ matrix M, into a $m \times r$ matrix: this is actually PCA for *feature extraction / reduction*.
- Similarly,
  - $U^T_{r\times m}M_{m\times n} = U^T_{r\times m}U_{m\times r}\Sigma_{r\times r}V^T_{r\times n} = \Sigma_{r\times r}V^T_{r\times n}$

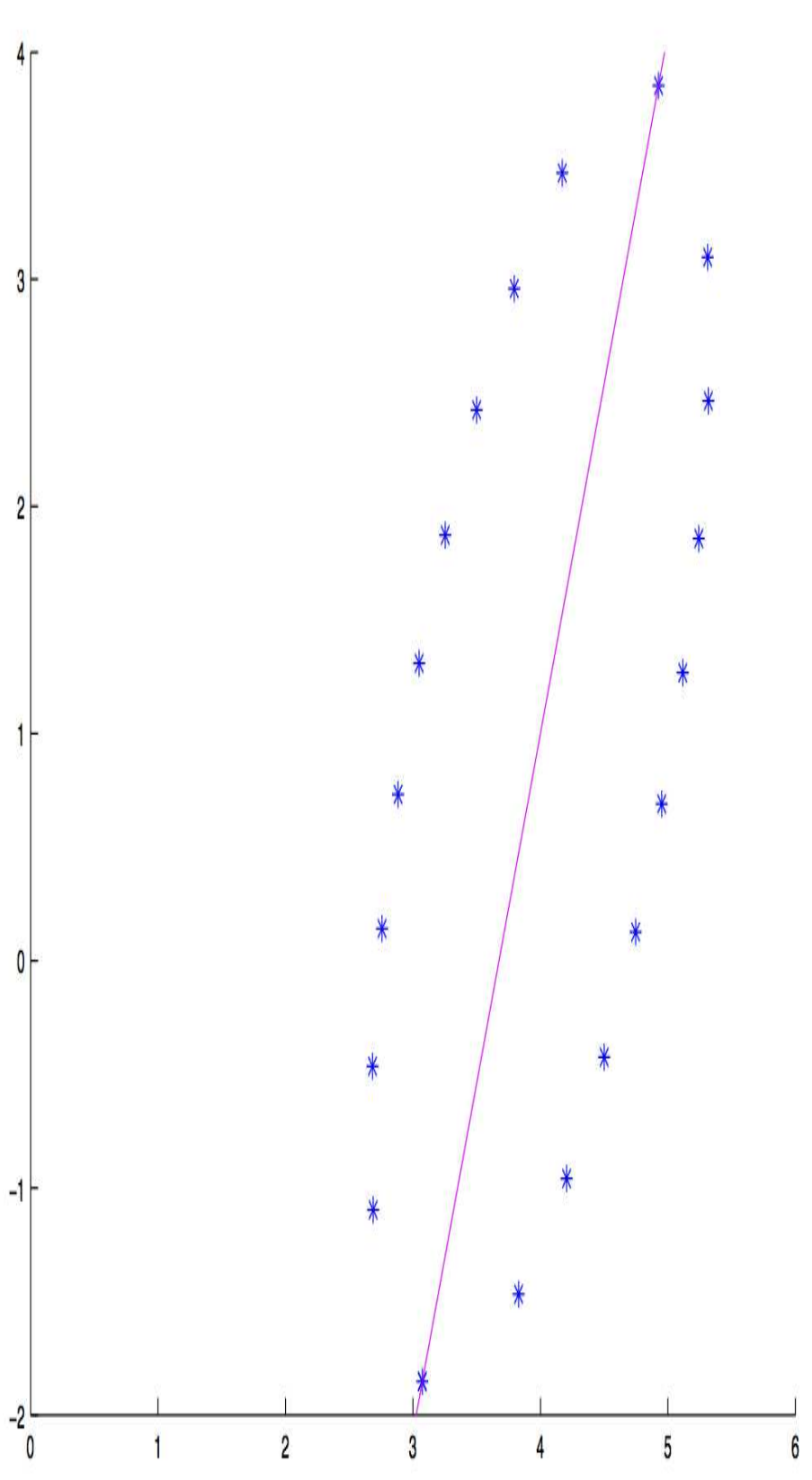
  Here $U^T_{r\times m}$ is a matrix, which can compress the rows of the $m \times n$ matrix M, into a $r \times n$ matrix: this is actually *sample reduction*.
- Singular values can be used to estimate how many principal components to keep.

  Rule of thumb: keep enough to explain 85% of the variation: $\frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{n}\sigma_i^2} \approx 85\%$

SVD Application (II): *Principle Component Analysis*

SVD will give vectors that go through the origin.
Centering makes sure that the origin is in the middle of the data set.





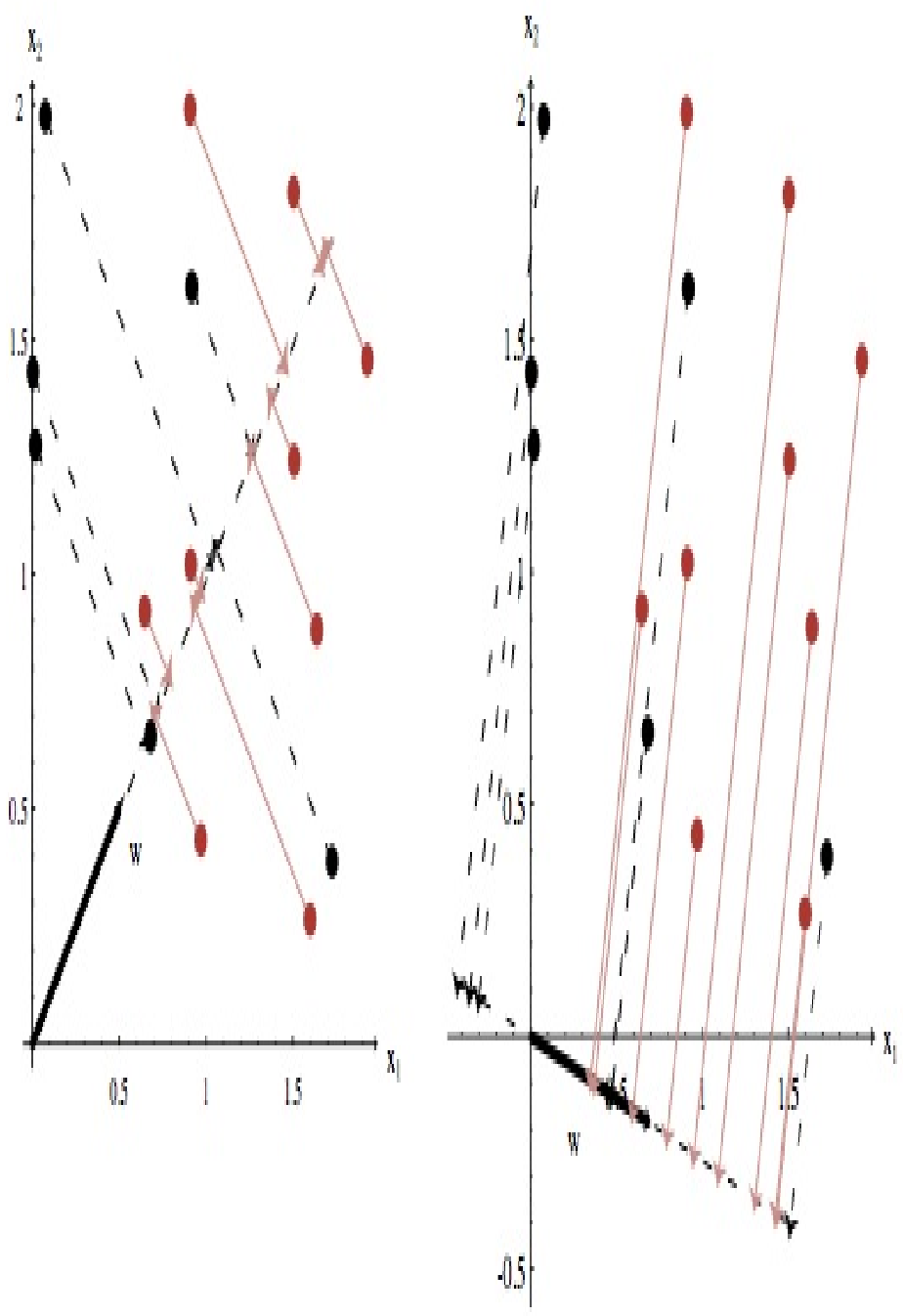
SVD Application (II): *Principle Component Analysis*

Exercise:

- Let M be a centered data matrix, and $M = U\Sigma V^T$.
  - the principal components of M are V,
  - the coordinates in the basis defined by the principal components are $U\Sigma$.
- As $M^T = V\Sigma^T U^T$, aren't the principal components of M^T given by U and the coordinates $V\Sigma^T$? so
  - the principal components of M are the new coordinates of M^T and vice versa,
  - modulo a multiplication by the diagonal matrix Σ (or its inverse)?

SVD Application (II-b): *Fisher Linear Discriminant* (Binary)

From training data we want to: find out a direction where *the separation between the class means is high* and *the overlap between the classes is small*.



Assume that w is a $n \times 1$ unit column vector of projection weights when the data M is projected along w.

- The projection of a vector x onto w is: $w^T x = \sum_{i=1}^{n} x_i w_i$
- The projection of data M along w is: Mw

LDA tries to find the unit vector w so that the separation between the projections of class means is high and the projections of the class overlap is small, subject to $w^T w = 1$.

SVD Application (II-b): *Fisher Linear Discriminant* (Binary)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_1 is in subset $\mathscr{D}_1$ labelled as ω_1, and n_2 is in subset $\mathscr{D}_2$ labelled as ω_2. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ which is divided into the subset $\mathscr{Y}_1$ and $\mathscr{Y}_2$ will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- If m_i is the d-dimensional sample mean: $m_i = \frac{1}{n_i}\sum_{x\in\mathscr{D}_i} x$. the *sample mean* of the projected points is then:

$$\tilde{m}_i = \frac{1}{n_i}\sum_{y\in\mathscr{Y}_i} y = \frac{1}{n_i}\sum_{x\in\mathscr{D}_i} w^t x = w^t m_i$$

- Hence the square of distance between the projected means is:

$$|\tilde{m}_1 - \tilde{m}_2|^2 = \left|w^t(m_1 - m_2)\right|^2 = (w^t m_1 - w^t m_2)^2 = w^t(m_1 - m_2)(m_1 - m_2)^t w = w^t S_B w$$

where the *between-class scatter matrix* $S_B = (m_1 - m_2)(m_1 - m_2)^t$

SVD Application (II-b): *Fisher Linear Discriminant* (Binary)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_1 is in subset $\mathscr{D}_1$ labelled as ω_1, and n_2 is in subset $\mathscr{D}_2$ labelled as ω_2. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ which is divided into the subset $\mathscr{Y}_1$ and $\mathscr{Y}_2$ will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- the *scatter* for projected samples labelled ω_i is

$$\tilde{s}_i^2 = \sum_{y\in\mathscr{Y}_i}(y - \tilde{m}_i)^2 = \sum_{x\in\mathscr{D}_i}(w^t x - w^t m_i)^2 = \sum_{x\in\mathscr{D}_i} w^t(x - m_i)(x - m_i)^t w = w^t S_i w$$

where the scatter matrix $S_i = \sum_{x\in\mathscr{D}_i}(x - m_i)(x - m_i)^t$

- the total *within-class scatter* of the projected samples is then

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^t S_W w$$

where $S_W = S_1 + S_2$.

SVD Application (II-b): *Fisher Linear Discriminant* (Binary)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_1 is in subset $\mathscr{D}_1$ labelled as ω_1, and n_2 is in subset $\mathscr{D}_2$ labelled as ω_2. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ which is divided into the subset $\mathscr{Y}_1$ and $\mathscr{Y}_2$ will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- The *Fisher linear criterion* is then

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{w^t S_B w}{w^t S_W w}$$

which is also known as the generalized *Rayleigh quotient*.

SVD Application (II-b): *Fisher Linear Discriminant* (Binary)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_1 is in subset $\mathscr{D}_1$ labelled as ω_1, and n_2 is in subset $\mathscr{D}_2$ labelled as ω_2. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ which is divided into the subset $\mathscr{Y}_1$ and $\mathscr{Y}_2$ will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- a vector w that maximizes $J(\cdot)$ must satisfy

$$S_B w = \lambda S_W w$$

when S_w is *nonsingular*, we have a conventional eigenvalue problem:

$$S_w^{-1} S_B w = \lambda w$$

in special case, we have a solution: the *canonical variate* $w = S_W^{-1}(m_1 - m_2)$.

SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_i is in subset $\mathscr{D}_i$ labelled as ω_i. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ divided into c subsets $\mathscr{Y}_i$ ($i = 1, \cdots, c$) will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- the *within-class scatter matrix* is

$$S_W = \sum_{i=1}^{c} S_i$$

with $S_i = \sum_{x \in \mathscr{D}_i} (x - m_i)(x - m_i)^t$, and $m_i = \frac{1}{n_i} \sum_{x \in \mathscr{D}_i} x$.

SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_i is in subset $\mathscr{D}_i$ labelled as ω_i. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ divided into c subsets $\mathscr{Y}_i$ ($i = 1, \cdots, c$) will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- the *total mean vector* is $m = \frac{1}{n} \sum_x x = \frac{1}{n} \sum_{i=1}^{c} n_i m_i$.
- the *total scatter matrix* is

$$S_T = \sum_x (x - m)(x - m)^t = \sum_{i=1}^{c} \sum_{x \in \mathscr{D}_i} (x - m_i + m_i - m)(x - m_i + m_i - m)^t = S_W + \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^t$$

- let the *between-class scatter matrix* be

$$S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^t$$

Hence $S_T = S_W + S_B$.

SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_i is in subset $\mathscr{D}_i$ labelled as ω_i. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ divided into c subsets $\mathscr{Y}_i$ ($i = 1, \cdots, c$) will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- let $y = W^t x$ be the projected samples. Then we have $\tilde{m}_i = \frac{1}{n_i} \sum_{y \in \mathscr{Y}_i} y$, $\tilde{m} = \frac{1}{n_i} \sum_{i=1}^{c} n_i \tilde{m}_i$, $\tilde{S}_W = \sum_{i=1}^{c} \sum_{y \in \mathscr{Y}_i} (y - \tilde{m}_i)(y - \tilde{m}_i)^t$, and $\tilde{S}_B = \sum_{i=1}^{c} n_i (\tilde{m}_i - \tilde{m})(\tilde{m}_i - \tilde{m})^t$
- the *within-class scatter matrix* on the projected space is then $\tilde{S}_W = W^t S_W W$
- the *between-class scatter matrix* on the projected space is then $\tilde{S}_B = W^t S_B W$

SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_i is in subset $\mathscr{D}_i$ labelled as ω_i. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ divided into c subsets $\mathscr{Y}_i$ ($i = 1, \cdots, c$) will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

- The *Fisher linear criterion* is then

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{|W^t \tilde{S}_B W|}{|W^t \tilde{S}_W W|}$$

- The columns of an optimal W are the generalized eigenvectors that correspond to the largest eigenvalues in

$$S_B w_i = \lambda_i S_W w_i$$

- Because S_B is the sum of c matrices of rank one or less, and because only $c - 1$ of them are independent, S_B is of rank $c - 1$ or less. Hence no more than $c - 1$ of geigenvalues are nonzero, and W has *no more than* $c - 1$ columns.

SVD Application (II-c): *Fisher Linear Discriminant* (Multiple)

LDA

Suppose a set of n d-dimensional samples $x_1, \cdots, x_n$, n_i is in subset $\mathscr{D}_i$ labelled as ω_i. If projection on a unit vector w, a corresponding set of n samples $y_1, \cdots, y_n$ divided into c subsets $\mathscr{Y}_i$ ($i = 1, \cdots, c$) will be generated. LDA tries to find the best w such that

1. the *between-class scatter* is big
2. the *within-class scatter* is small

Closing Thoughts on LDA

- The way we introduced Fisher's linear discriminant analysis, no assumptions are needed. It is simply a sensible rule to try to classify observations.
- But if we have two groups from two multivariate normal distributions with the same covariance matrix, then Fisher's linear discriminant analysis corresponds exactly to the maximum likelihood rule for classification. (For more than two groups this is generally not the case.)

SVD Application (III): *Least Squares Approximations*

LSA

For the least squares form: $A^T A\bar{x} = A^T b$, if $A^T A$ is *invertible*, we have the least squares approximation $\bar{x} = (A^T A)^{-1} A^T b$.

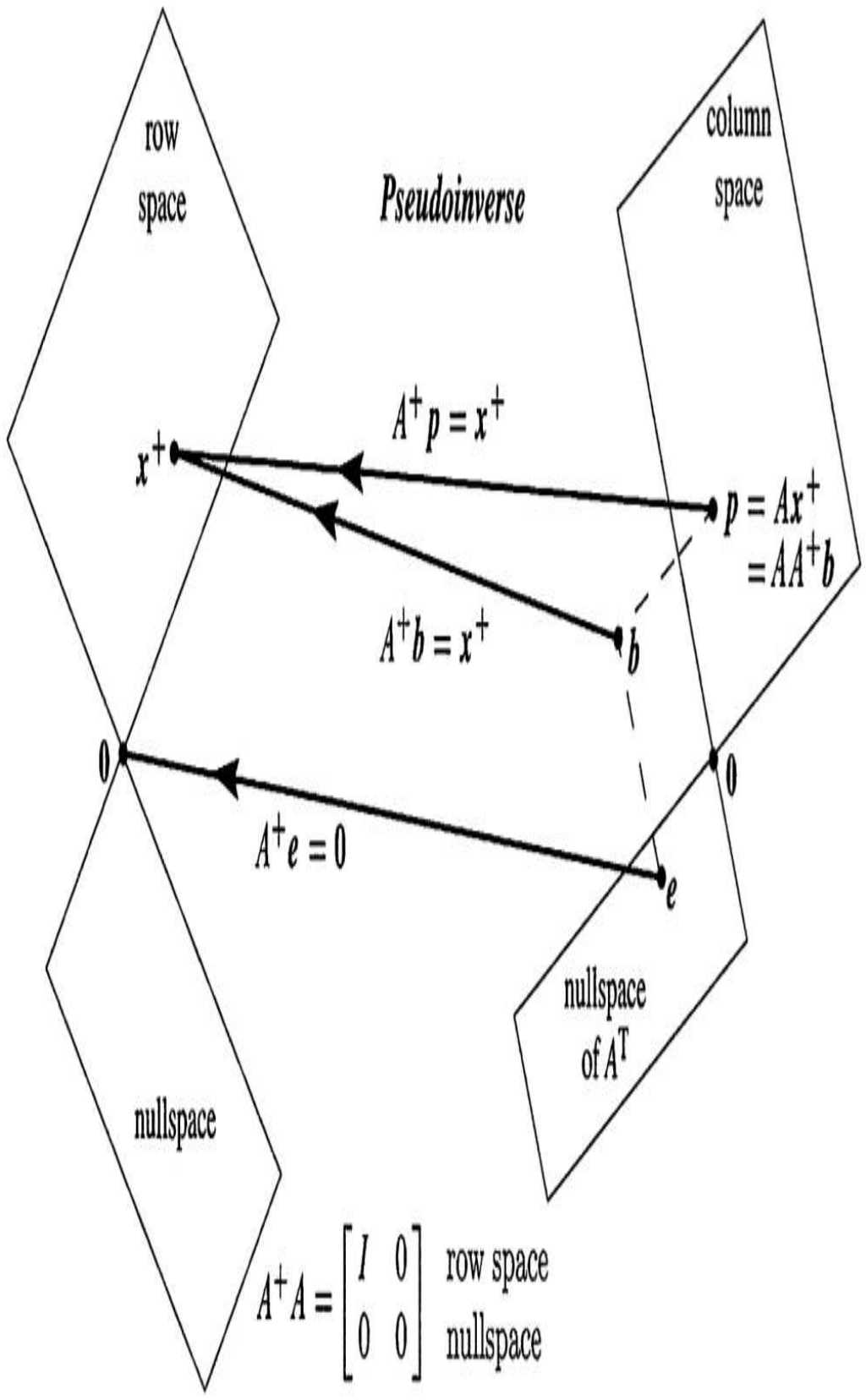
- Now what if A has dependent columns ($rank < n$)?
  - There can be many solutions to $A^T A\bar{x} = A^T b$.
  - Then find the solution with the *minimal length*.
- Let us back to SVD, in which a multiplies v_i in the row space to give $\sigma_i u_i$ in the column space. A^{-1} must to the opposite!
  - If $Av = \sigma u$, then $A^{-1}u = \frac{v}{\sigma}$.
  - The matrix that multiplies u_i to produce v_i/σ exists, and is the *pseudoinverse* A^+:

$$A^+ = V\Sigma^+ U^T = \begin{pmatrix} & & \\ v_1 & \cdots v_r \cdots & v_n \\ & & \end{pmatrix} \begin{pmatrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_r^{-1} \end{pmatrix} \begin{pmatrix} & & \\ u_1 & \cdots u_r \cdots & u_m \\ & & \end{pmatrix}^T$$

SVD Application (III): *Least Squares Approximations*

LSA

The *pseudoinverse* $A^+ = V\Sigma^+U^T = \begin{pmatrix} v_1 \cdots v_r \cdots v_n \end{pmatrix} \begin{pmatrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_r^{-1} \end{pmatrix} \begin{pmatrix} u_1 \cdots u_r \cdots u_m \end{pmatrix}^T$



- When $m = n = r$ we have $A^{-1} = A^+$, as $U\Sigma V^T$ can be inverted into $V\Sigma^{-1}U^T$.
- When $r < m$ or $r < n$, *pseudoinverse* A^+ is with the same rank r:
  - $A^+u_i = \frac{1}{\sigma_i}v_i$ for $i \le r$
  - $A^+u_i = 0$ for $i > r$
- Ax^+ in the column space goes back to $A^+Ax^+ = x^+$ in the row space.
  - AA^+: projection matrix onto the *column* space of A
  - A^+A: projection matrix onto the *row* space of A

SVD Application (III): *Least Squares Approximations*

LSA

For the least squares form: $A^TA\bar{x} = A^Tb$, if A^TA is *invertible*, we have the least squares approximation $\bar{x} = (A^TA)^{-1}A^Tb$. Now we try a different perspective.

- We are to choose the vector x so as to minimize $|Ax - b|$:

  $$Ax - b = U\Sigma V^Tx - b = U\Sigma V^Tx - U(U^Tb) = U(\Sigma y - c)$$

  with $y = V^Tx$ and $c = U^Tb$. As U is an orthogonal, we have $|U(\Sigma y - c)| = |\Sigma y - c|$.
- Let the entries of Σ be σ_i for $1 \le i \le k$, then $\Sigma y - c = \begin{pmatrix} \sigma_1 y_1 - c_1 \\ \vdots \\ \sigma_k y_k - c_k \\ -c_{k+1} \\ \vdots \\ -c_m \end{pmatrix}^T$
  - When $y_i = c_i/\sigma_i$ for $1 \le i \le k$, $\Sigma y - c$ assumes its minimal length.
  - Let Σ^+ be te matrix that transposes Σ and inverting the nonzero diagonal entries. Then $y = \Sigma^+c$ and the general solution is $\bar{x} = Vy = V\Sigma^+U^Tb$.

Matrix Factorization: Revisit

Defn

We wish to decompose the matrix A by writing it as a product of two or more matrices:

$$A_{m\times n} = B_{m\times k} C_{k\times n}, \quad A_{m\times n} = B_{m\times k} C_{k\times r} D_{r\times n},$$

- This is done in such a way that the right side of the equation yields some useful information or insight to the nature of the data matrix A, or is in other ways useful for solving the problem at hand.
- There are numerous useful *matrix decompositions*, also known as *matrix factorization*.

In addition to those methods in this course, many other methods exist (which needs to be included in this course by future TULIP Lab members):

- Nonnegative Matrix Factorization
- Factorial Analysis
- ICA
- PMF
- ...

NMF: *Nonnegative Matrix Factorization*

Defn

Given a nonnegative matrix $M \in \mathscr{R}^{m\times n}$, we wish to express the matrix as a product of two nonnegative matrices $W \in \mathscr{R}^{m\times k}$ and $H \in \mathscr{R}^{k\times n}$:

$$M \approx WH$$

- NMF is not unique: Let D be a diagonal matrix with positive diagonal entries. Then

$$M \approx (WD)(D^{-1}H)$$

- NMF minimizes $\|A - WH\|$.
- Algorithms exist, both basic and more advanced.

Related Resources

G. Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 2006
G. Strang. MIT linear algebra 18.06, 2005
T. Tao. UCLA Linear Algebra Math115A, 2002
D. Austin. We recommend a singular value decomposition, 2009
D. Kalman. A *Singularly Valuable Decomposition*: The SVD of a matrix. *The College Mathematics Journal*, 27:2–23, 1996

Questions?

Contact Information

Associate Professor **Gang Li**
School of Information Technology
Deakin University, Australia

GANGLI@TULIP.ORG.AU

TEAM FOR UNIVERSAL LEARNING AND INTELLIGENT PROCESSING