Lecture 18

Least squares estimation
(revisited & generalized)

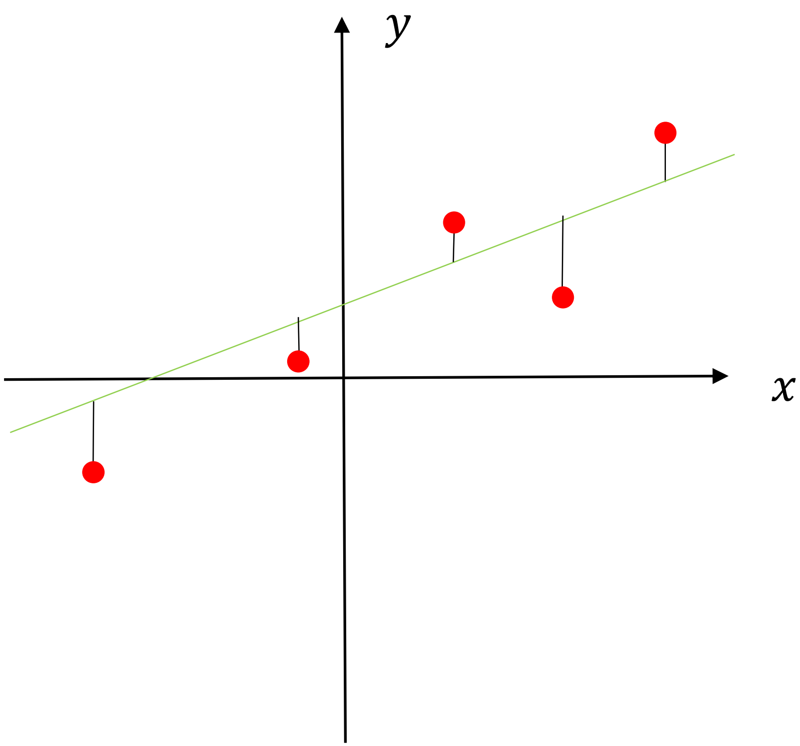
Singular value decomposition (SVD)

Tues. Nov. 6, 2018

Recall from lecture 5.

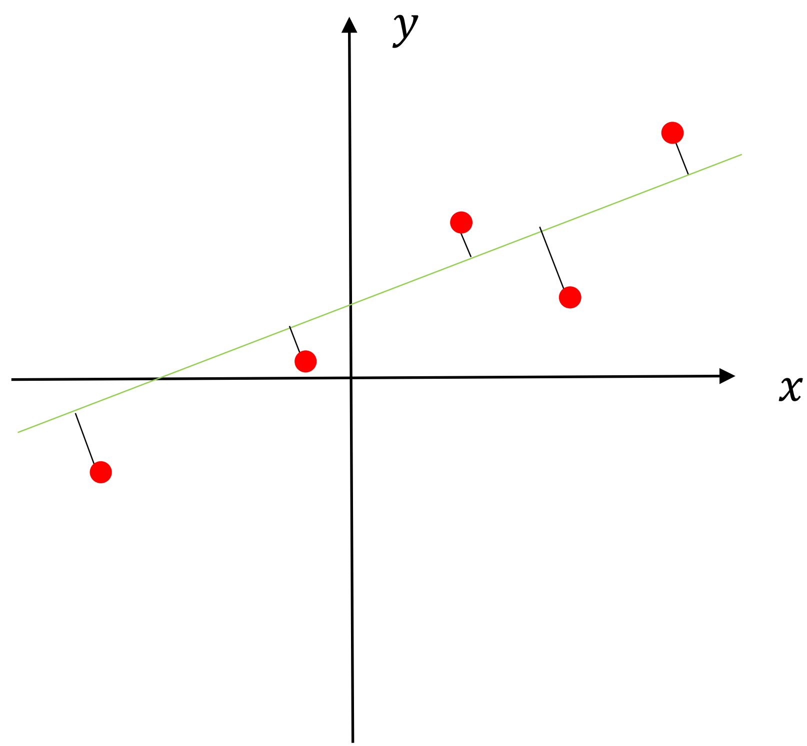
Version 1: linear regression

Error is distance to line in y direction only.



Version 2: "total least squares"

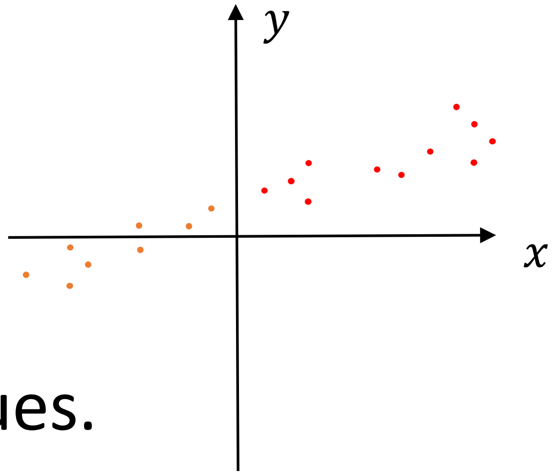
Error is distance perpendicular to line.



Least squares: version 1 (linear regression)

Model was:

$$y_i = m\, x_i + b + \; n_i$$



where n_i is additive noise in the y_i values.

We solved for:

$$\underset{m,\, b}{\operatorname{argmin}} \sum_i (y_i - (m\, x_i + b))^2$$

Heads up!

In this problem formulation,

- (x_i, y_i) are data points i.e. constants

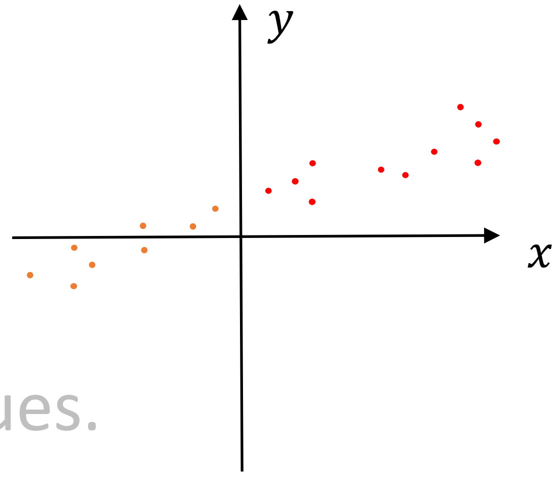
- $(\ m, b)$ are variables we are solving for.

Usually when we write $y = mx + b$, the (x, y) are the variables and (m, b) are the constants.

Changing the notation to avoid confusion...

Model is:

$$y_i = u_1\, x_i + u_2 + n_i$$

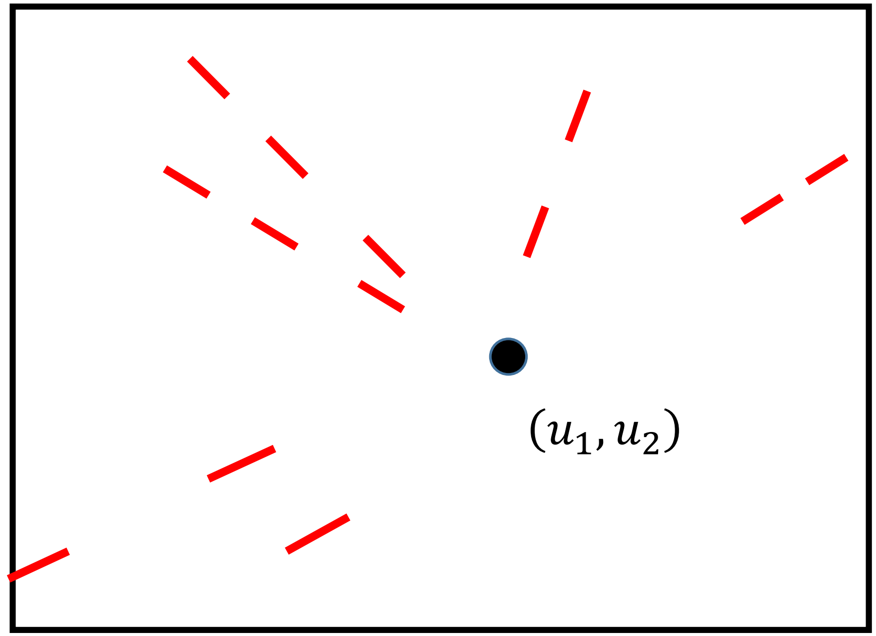


where n_i is additive noise in the y_i values.

We are solving:

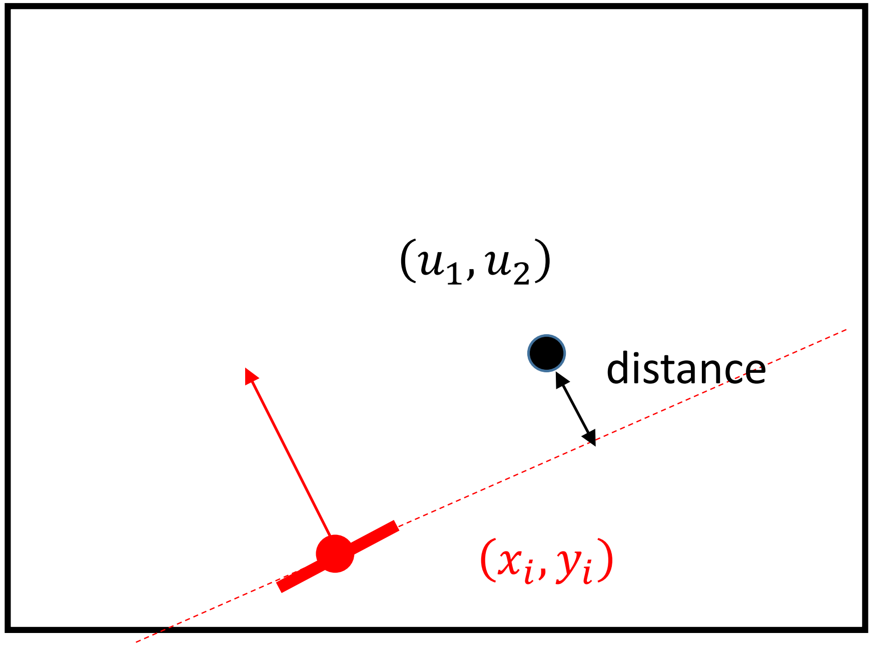
$$\operatorname*{argmin}_{u_1, u_2} \sum_i (y_i - (u_1\, x_i + u_2))^2$$

Vanishing point detection (lecture 6)



Suppose we have a set of edge estimates $(x_i, y_i,\ \theta_i)$.

We would like to estimate a vanishing point (u_1, u_2). It is the point that is closest to the lines that are defined by the edges.



For each edge estimate $(x_i, y_i,\ \theta_i)$ and for any hypothetic vanishing point (u_1, u_2), the distance from that vanishing point to the line defined by the edge is defined by distance shown above. See formula on next slide.

Vanishing point detection (lecture 6)

To find the vanishing point, we minimized the sum of squared distances from each point (x_i, y_i) to line defined by θ_i and (u_1, u_2):

$$\underset{(u_1, u_2)}{argmin} \sum_{i=1}^{N} \left((x_i - u_1,\ y_i - u_2) \cdot (\cos\theta_i, \sin\theta_i) \right)^2$$

For both of previous problems, we minimized a sum of squares of the form:

$$\| \boldsymbol{A}\,\boldsymbol{u} - \boldsymbol{b} \|^2$$

where

$\boldsymbol{A}$ is an N x 2 data matrix

$\boldsymbol{u}$ is a 2 x 1 vector of variables (u_1, u_2)

$\boldsymbol{b}$ is an N x 1 data vector

NOTE: The $\boldsymbol{A}$ and $\boldsymbol{b}$ were given. We solved for $\boldsymbol{u}$.

More generally…… Many least squares problems can be written as:

Find the $\boldsymbol{u}$ that minimizes:

$$\| \boldsymbol{A}\,\boldsymbol{u} - \boldsymbol{b} \|^2$$

where

$\boldsymbol{A}$ is an $m \times n$ data matrix

$\boldsymbol{u}$ is a $n \times 1$ vector of variables

$\boldsymbol{b}$ is a $m \times 1$ data vector

$$
\begin{aligned}
\| \mathbf{Au} - \mathbf{b} \|^2 &= (\mathbf{Au} - \mathbf{b})^T (\mathbf{Au} - \mathbf{b}) \\
&= \mathbf{u}^T \mathbf{A}^T \mathbf{Au} - 2\mathbf{b}^T \mathbf{Au} + \mathbf{b}^T \mathbf{b}
\end{aligned}
$$

$$\| \mathbf{Au} - \mathbf{b} \|^2 = (\mathbf{Au} - \mathbf{b})^T(\mathbf{Au} - \mathbf{b})$$

$$= \mathbf{u}^T\mathbf{A}^T\mathbf{Au} - 2\mathbf{b}^T\mathbf{Au} + \mathbf{b}^T\mathbf{b}$$

Take the derivative with respect to each of the ***u*** variables and set to 0.

$$2\mathbf{A}^T\mathbf{Au} - 2\mathbf{A}^T\mathbf{b} = 0$$

$$\mathbf{A}^T\mathbf{Au} = \mathbf{A}^T\mathbf{b}$$

If the columns are of A are linearly independent, then $\mathbf{A}^T\mathbf{A}$ is invertible.

$$\| \mathbf{Au} - \mathbf{b} \|^2 \;=\; (\mathbf{Au} - \mathbf{b})^T(\mathbf{Au} - \mathbf{b})$$

$$= \mathbf{u}^T\mathbf{A}^T\mathbf{A}\mathbf{u} - 2\mathbf{b}^T\mathbf{A}\mathbf{u} + \mathbf{b}^T\mathbf{b}$$

Take the derivative with respect to each of the ***u*** variables and set to 0.

$$2\mathbf{A}^T\mathbf{A}\mathbf{u} - 2\mathbf{A}^T\mathbf{b} = 0$$

$$\mathbf{A}^T\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{b}$$

If the columns are of A are linearly independent, then:

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

So, to find the u that minimizes:

$$\| \mathbf{Au} - \mathbf{b} \|^2 \;=\; (\mathbf{Au} - \mathbf{b})^T(\mathbf{Au} - \mathbf{b})$$

we compute:

$$\mathbf{u} = \underbrace{(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T}\mathbf{b}$$

This is called the pseudoinverse of **A**.

(Draw geometric interpretation on blackboard.)

Example: Image Registration (lectures 11 & 12)

- Measure spatial and temporal derivatives of image intensities (data)
- Estimate the image translation component between frames

$$\boldsymbol{u} = \left(h_x, h_y\right)$$

- This was basic Lucas-Kanade (Assignment 3).

- Estimate the image translation and more general deformations (rotation, shear, scale)

$$\boldsymbol{u} = \left(h_x, h_y,\ D_{11}, D_{12}, D_{21}, D_{22}\right)$$

Least squares: version 2

Many least squares problems can be written as:

Find the $\boldsymbol{u}$ that minimizes L2 norm

$$\| \boldsymbol{A}\, \boldsymbol{u} \|^2$$

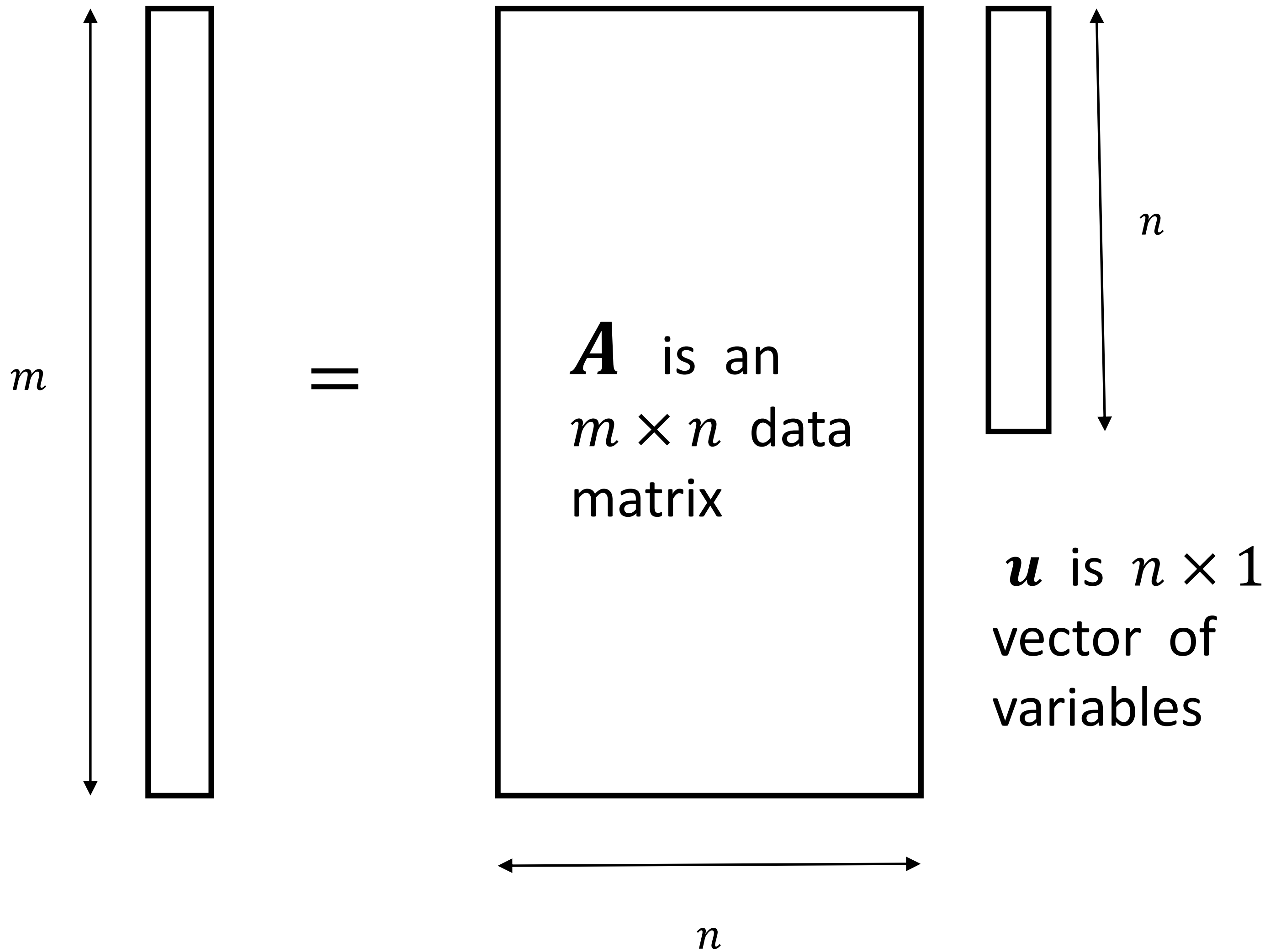
subject to $\| \boldsymbol{u} \| = 1.$

where

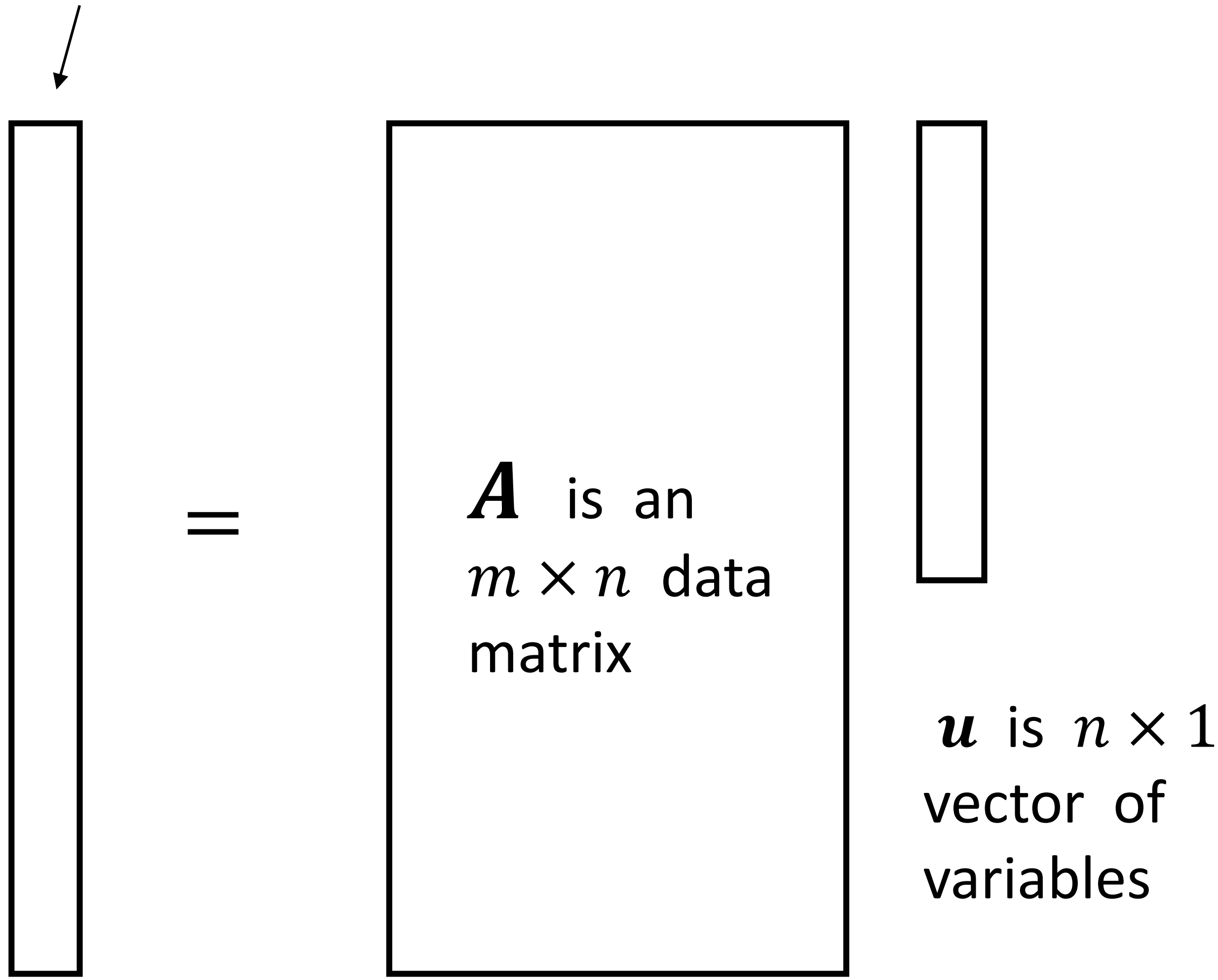
$\boldsymbol{A}$ is an $m \times n$ data matrix

$\boldsymbol{u}$ is a $n \times 1$ vector of variables

Find unit length $\boldsymbol{u}$ that minimizes the length of this vector.



Find a linear combination of the columns of $\boldsymbol{A}$ that minimizes the length of this vector, where $\boldsymbol{u}$ has unit length.



Examples of Version 2 Problems (coming soon)

- Camera calibration

  (if you have a 3D object with known points and you can locate the points in the image, then you can solve for the camera intrinsin and extrinsic parameters)

- Image Stitching for panoramas

  (if you take a sequence of pictures from same camera at same position but different orientations, then you can stitch the images together. Solve for the perspective deformation between images -- called a homography)

- Binocular Stereo

  (If you have two images of a scene from two cameras and you can identify *some* corresponding points between the two images, then you can compute a constraint -- called the Fundamental Matrix -- on *all* corresponding pairs of points. This narrows the search space for correspondences.)

Examples of Version 2 Problems (coming soon)

- Camera calibration

  (if you have a 3D object with known points and you can locate the points in the image, then you can solve for the camera intrinsin and extrinsic parameters)

- Image Stitching for panoramas

  (if you take a sequence of pictures from same camera at same position but different orientations, then you can stitch the images together. Solve for the perspective deformation between images -- called a homography)

- Binocular Stereo

  (If you have two images of a scene from two cameras and you can identify *some* corresponding points between the two images, then you can compute a constraint -- called the Fundamental Matrix -- on *all* corresponding pairs of points. This narrows the search space for correspondences.)

Examples of Version 2 Problems (coming soon)

- Camera calibration

  (if you have a 3D object with known points and you can locate the points in the image, then you can solve for the camera intrinsin and extrinsic parameters)

- Image Stitching for panoramas

  (if you take a sequence of pictures from same camera at same position but different orientations, then you can stitch the images together. Solve for the perspective deformation between images -- called a homography)

- Binocular Stereo

  (If you have two images of a scene from two cameras and you can identify *some* corresponding points between the two images, then you can compute a constraint -- called the Fundamental Matrix -- on *all* corresponding pairs of points. This narrows the search space for correspondences.)

Least squares: version 2

Many least squares problems can be written as:

Find the $\boldsymbol{u}$ that minimizes L2 norm

$$\| \boldsymbol{A} \, \boldsymbol{u} \|^2$$

subject to $\| \boldsymbol{u} \| = 1.$

Solution (claimed back in lecture 5):

Compute the eigenvectors of $\boldsymbol{n} \times \boldsymbol{n}$ matrix $\boldsymbol{A}^{\boldsymbol{T}} \boldsymbol{A}$.

Take the unit eigenvector which has the smallest eigenvalue.

Why does eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ within minimum eigenvalue solve the problem ?

$$\| \boldsymbol{A}\,\boldsymbol{u} \|^2 = \quad \boldsymbol{u}^T\boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{u}$$

Why does eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ within minimum eigenvalue solve the problem ?

$$\| \boldsymbol{A}\,\boldsymbol{u} \|^2 = \quad \boldsymbol{u}^T \boldsymbol{A}^T \boldsymbol{A}\,\boldsymbol{u}$$

$$= \quad \lambda\, \boldsymbol{u}^T \boldsymbol{u}$$

when $\boldsymbol{u}$ is an eigenvector (and λ is its eigenvalue)

$$= \quad \lambda$$

when $\boldsymbol{u}$ has unit length

$$\geq \quad 0$$

because L2 norm is non-negative.

Why does eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ within minimum eigenvalue give us the solution ?

Also, linear algebra tells us that the $n \times n$ matrix $\boldsymbol{A}^T\boldsymbol{A}$ has

- non-negative eigenvalues
- n perpendicular eigenvectors (if eigenvalues are unique)

Therefore we can write any vector $\boldsymbol{u}$ as a sum of these eigenvectors.

Why does eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ within minimum eigenvalue give us the solution ?

Also, linear algebra tells us that the $n \times n$ matrix $\boldsymbol{A}^T\boldsymbol{A}$ has

- non-negative eigenvalues
- n perpendicular eigenvectors (if eigenvalues are unique)

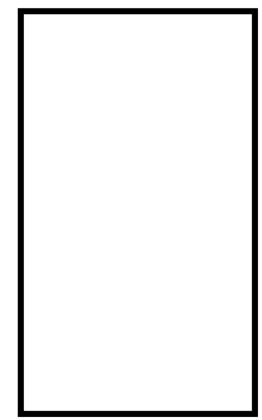
Therefore we can write any vector $\boldsymbol{u}$ as a sum of these eigenvectors.

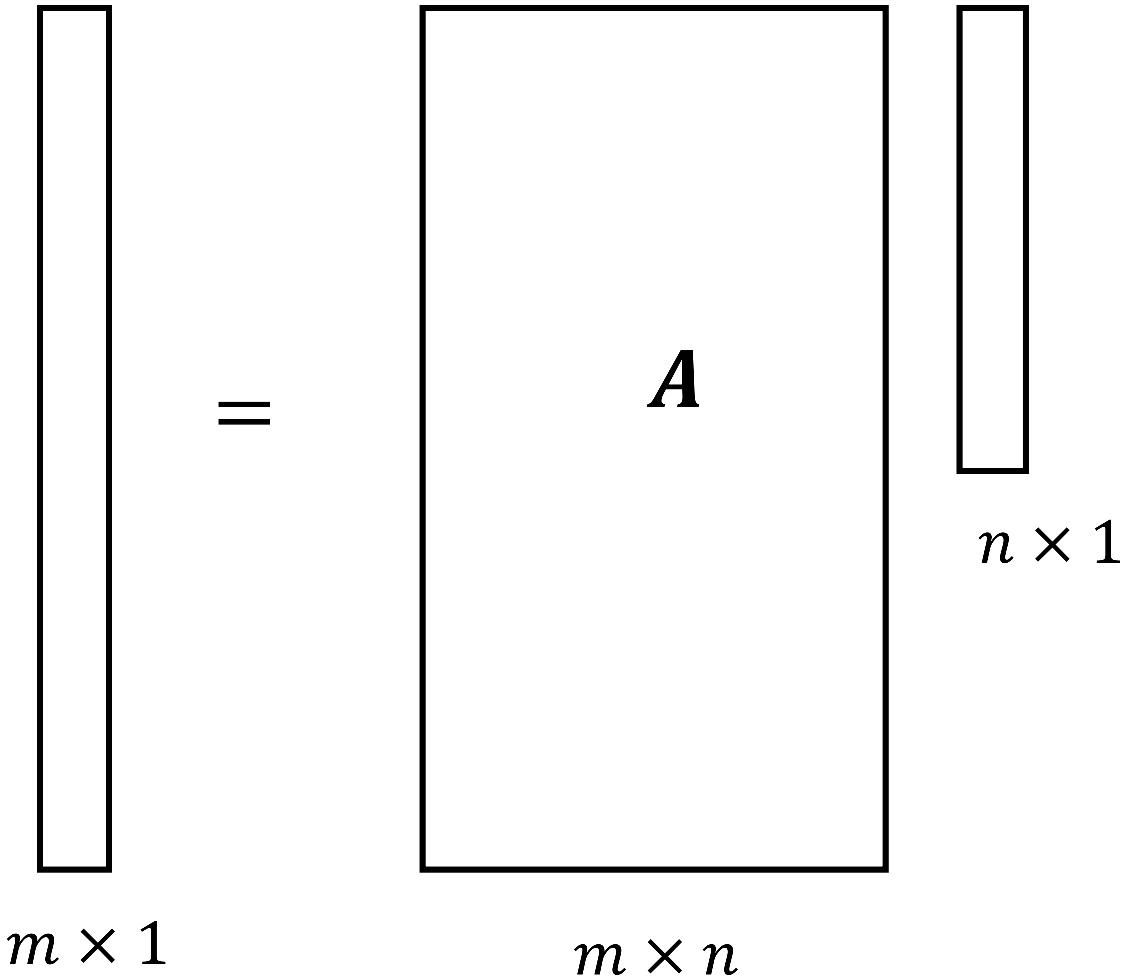
By inspection, if $\boldsymbol{u}$ is any vector of unit length then $\boldsymbol{u}^T\boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{u}$ will be smallest when $\boldsymbol{u}$ is the eigenvector with smallest eigenvalue.

Singular Value Decomposition (SVD)

$\boldsymbol{A}$ is any $m \times n$ data matrix.

In our examples, $m \geq n$.




$\boldsymbol{A}$
$n \times 1$
$m \times 1$
$m \times n$

A few slides ago, we mentioned that the eigenvectors of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ had :

- non-negative eigenvalues
- n orthogonal eigenvectors (if eigenvalues are unique)

Therefore we can write:

$$\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}\ \boldsymbol{V} = \boldsymbol{V}\ \boldsymbol{\Lambda}$$

where

- the columns of $\boldsymbol{V}$ are orthonormal eigenvectors of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ (unit length and orthogonal)
- $\boldsymbol{\Lambda}$ is a diagonal matrix (of eigenvalues, which are non-negative).

A few slides ago, we mentioned that the eigenvectors of $\boldsymbol{A^T A}$ had :

- non-negative eigenvalues
- n orthogonal eigenvectors (if eigenvalues are unique)

Therefore we can write:

$$\boldsymbol{A^T A \; V \; = \; V\Sigma^2}$$

where

- the columns of $\boldsymbol{V}$ are orthonormal eigenvectors of $\boldsymbol{A^T A}$ (unit length and orthogonal)
- $\boldsymbol{\Sigma}$ is a diagonal matrix of "singular values" (the square roots of the $\boldsymbol{\Lambda}$).

$$\boldsymbol{A^T A \; V \;=\; V \, \Sigma^2}$$

Multiplying on the left by $\boldsymbol{V^T}$ gives:

$$\boldsymbol{V^T A^T A \; V \;=\; \Sigma^2}$$

$$\boldsymbol{A^T A \ V} \ = \ \boldsymbol{V} \, \color{red}{\boldsymbol{\Sigma}^2}$$

Multiplying on the left by $\boldsymbol{V^T}$ gives:

$$\boldsymbol{V^T A^T A \ V} \ = \color{red}{\boldsymbol{\Sigma}^2}$$

By inspection, the columns of $\boldsymbol{A \ V}$ are orthogonal and

$$\boldsymbol{A \ V} \ = \boldsymbol{U} \ \color{red}{\boldsymbol{\Sigma}}$$

$$A^T A\ V = V\,\Sigma^2$$

Multiplying on the left by V^T gives:

$$V^T A^T A\ V = \Sigma^2$$

By inspection, the columns of $A\ V$ are orthogonal and

$$A\ V = U\ \Sigma$$

Since the columns of V are orthonormal, right multiplying by V^T gives:

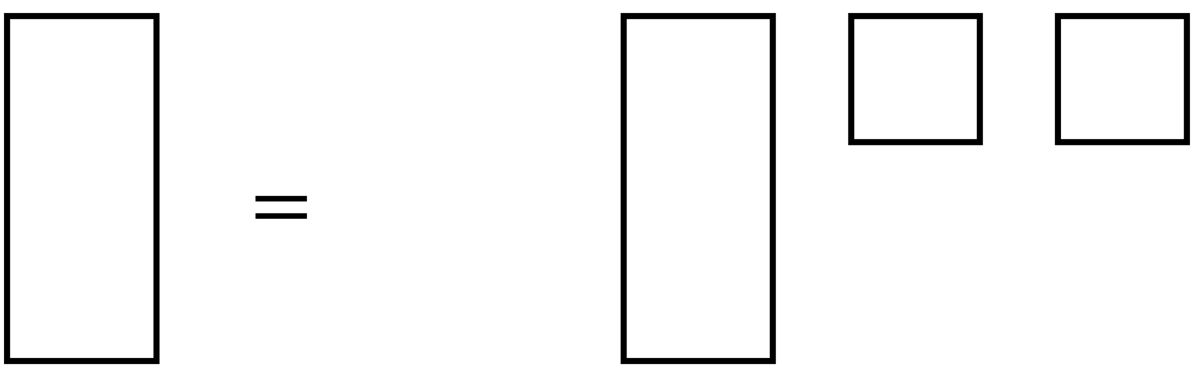
$$\boxed{A = U\ \Sigma\ V^T}$$

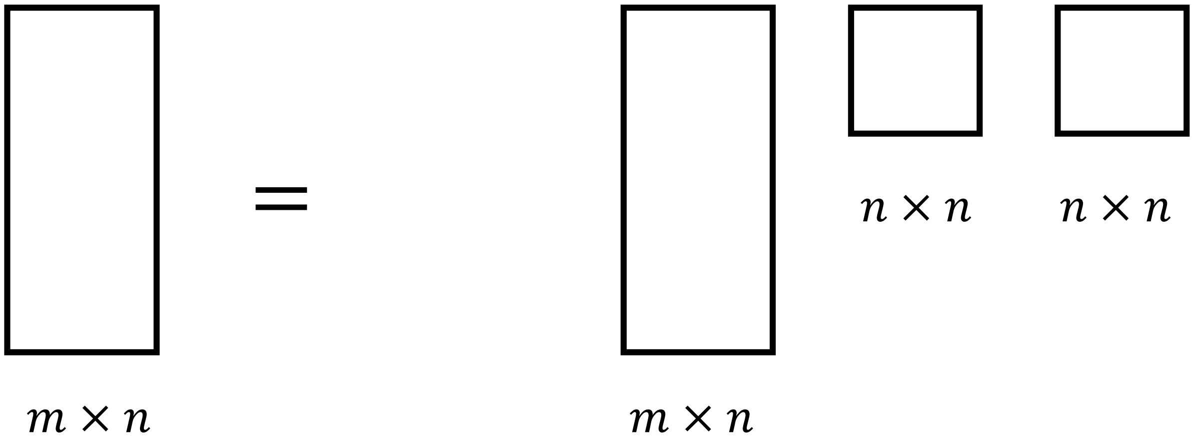
Singular Value Decomposition (SVD)

$\boldsymbol{A}$ is any $m \times n$ matrix.

In our examples, $m \geq n$.

Then we can write $\boldsymbol{A} = \boldsymbol{U} \, \color{red}{\boldsymbol{\Sigma}} \, \boldsymbol{V}^{\boldsymbol{T}}$.





$$\boldsymbol{A} = \boldsymbol{U}\ \boldsymbol{\Sigma}\ \boldsymbol{V}^{\boldsymbol{T}}$$

Map each column of $\boldsymbol{V}$ in $\boldsymbol{R}^{\boldsymbol{n}}$ to a corresponding column of $\boldsymbol{U}$ in $\boldsymbol{R}^{\boldsymbol{m}}$.

scale length independently in directions of the columns of $\boldsymbol{V}$

project onto the columns of $\boldsymbol{V}$

(which are orthonormal)

Matlab

[U, S, V] = svd(A)

Columns of U are ordered to correspond to singular values

Singular values returned as a vector in decreasing order

Columns of V are ordered to correspond to singular values

Examples of Version 2 Problems (coming soon)

- Camera calibration
- Image Stitching for panoramas
- Binocular Stereo

For each of these problems, we will set up a data matrix A and solve the problem by taking the SVD. The solution will be the column of V that corresponds to the smallest singular value.