Background on Estimation for Computer Vision
Overview of the Lecture

• Eigenvalues and eigenvectors
• Eigen-Decomposition (applied to Covariance matrix)
• SVD
• Solving a (non-homogeneous) system of linear equations (LS)
• Solving a homogeneous system of linear equations (Orthog. L S)
• Line fitting with Orth. LS
• Line fitting with LS
• Line fitting using LS with Regularization
• Line fitting with RANSAC
Mapping of Vectors

A matrix is a mapping system of vectors!

\[ A_{m \times n} \times x_{n \times 1} = y_{m \times 1} \]

Mapping System  Original Vector  Mapped Vector

How does it work?
Eigenvalues & Eigenvectors

• **Eigenvectors** (for a square $m \times m$ matrix $S$)

  \[ S \mathbf{v} = \lambda \mathbf{v} \]

  (right) eigenvector \quad \text{eigenvalue}

  $\mathbf{v} \in \mathbb{R}^m \neq 0 \quad \lambda \in \mathbb{R}$

• **How many eigenvalues** are there at most?

  \[ S \mathbf{v} = \lambda \mathbf{v} \iff (S - \lambda \mathbf{I}) \mathbf{v} = 0 \]

  only has a non-zero solution if $|S - \lambda \mathbf{I}| = 0$

  This is a $m$-th order equation in $\lambda$ which can have **at most $m$ distinct solutions** (roots of the characteristic polynomial) - can be complex even though $S$ is real.

**Example**

\[
\begin{pmatrix}
6 & -2 \\
4 & 0
\end{pmatrix}
\begin{pmatrix}
1 \\
2
\end{pmatrix}
= \begin{pmatrix}
2 \\
4
\end{pmatrix}
= 2 \begin{pmatrix}
1 \\
2
\end{pmatrix}
\]
Matrix-vector multiplication

\[
S = \begin{bmatrix}
3 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

has eigenvalues 3, 2, 0 with corresponding eigenvectors

\[
\begin{align*}
v_1 &= \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\
v_2 &= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\
v_3 &= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}
\end{align*}
\]

On each eigenvector, \( S \) acts as a multiple of the identity matrix: but as a different multiple on each.

Any vector (say \( x = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \)) can be viewed as a combination of the eigenvectors: \( x = 2v_1 + 4v_2 + 6v_3 \)
Eigenvalues & Eigenvectors

For symmetric matrices, eigenvectors for distinct eigenvalues are orthogonal

\[ S v_{\{1,2\}} = \lambda_{\{1,2\}} v_{\{1,2\}} \], and \( \lambda_1 \neq \lambda_2 \Rightarrow v_1 \cdot v_2 = 0 \)

All eigenvalues of a real symmetric matrix are real.

for complex \( \lambda \), if \( |S - \lambda I| = 0 \) and \( S = S^T \Rightarrow \lambda \in \mathbb{R} \)

All eigenvalues of a positive semidefinite matrix are non-negative

\[ \forall w \in \mathbb{R}^n, w^T S w \geq 0, \text{ then if } S v = \lambda v \Rightarrow \lambda \geq 0 \]
Example

• Let

\[ S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \]  

Real, symmetric.

• Then

\[ S - \lambda I = \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \Rightarrow (2 - \lambda)^2 - 1 = 0. \]

• The eigenvalues are 1 and 3 (nonnegative, real).

• The eigenvectors are orthogonal (and real):

\[ \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \]

Plug in these values and solve for eigenvectors.
Let $S \in \mathbb{R}^{m \times m}$ be a square matrix with $m$ linearly independent eigenvectors (a “non-defective” matrix).

**Theorem:** Exists an eigen decomposition

$$S = U \Lambda U^{-1}$$

- (cf. matrix diagonalization theorem)

- Columns of $U$ are eigenvectors of $S$

- Diagonal elements of $\Lambda$ are eigenvalues of $S$

$$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$
Symmetric Eigen Decomposition

• If $s \in \mathbb{R}^{m \times m}$ is a symmetric matrix:

• **Theorem**: Exists a (unique) **eigen decomposition**
  \[ s = Q \Lambda Q^T \]

• where $Q$ is **orthogonal**:
  
  • $Q^T = Q$
  
  • Columns of $Q$ are normalized eigenvectors
  
  • Columns are orthogonal.
  
  • (everything is real)
If $S$ is co-variance matrix

$$S = \frac{1}{n} \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$$

where $\bar{x}$ is the mean of the set of points $\{x_i\}$

Eigenvalues represent covariances

Eigenvectors represent linearly independent directions of variation in data

Eigenvalues of a covariance matrix of data shown in blue. Data is drawn from a gaussian distribution.
Physical interpretation

• Consider a covariance matrix, $\mathbf{S}$, i.e., $\mathbf{S} = \frac{1}{n} \mathbf{A} \mathbf{A}^T$ for some $\mathbf{A}$

$$\mathbf{S} = \begin{bmatrix} 1 & .75 \\ .75 & 1 \end{bmatrix} \Rightarrow \lambda_1 = 1.75, \lambda_2 = 0.25$$

• Error ellipse with the major axis as the larger eigenvalue and the minor axis as the smaller eigenvalue
Spherical, diagonal, full covariance

\[ \Sigma = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \quad \Sigma = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \quad \Sigma = \begin{pmatrix} \sigma_x^2 & \rho \sigma_x \sigma_y \\ \rho \sigma_x \sigma_y & \sigma_y^2 \end{pmatrix} \]
The concept is used in Principal Component Analysis (PCA)

- Orthogonal directions of greatest variance in data
- Projections along PC1 (Principal Component) discriminate the data most along any one axis
Singular Value Decomposition

For an $m \times n$ matrix $A$ of rank $r$ there exists a factorization (Singular Value Decomposition = SVD) as follows:

\[ A = U \Sigma V^T \]

The columns of $U$ are orthogonal eigenvectors of $A A^T$.

The columns of $V$ are orthogonal eigenvectors of $A^T A$.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of $A A^T$ are the eigenvalues of $A^T A$.

\[ \sigma_i = \sqrt{\lambda_i} \]

\[ \Sigma = \text{diag}(\sigma_1 \ldots \sigma_r) \]
Singular Value Decomposition

• Illustration of SVD dimensions and sparseness
SVD example

Let

\[
A = \begin{bmatrix}
1 & -1 \\
0 & 1 \\
1 & 0
\end{bmatrix}
\]

Thus \( m=3, \ n=2 \). Its SVD is

\[
\begin{bmatrix}
0 & 2/\sqrt{6} & 1/\sqrt{3} \\
1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\
1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3}
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
0 & \sqrt{3} \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
1/\sqrt{2} & 1/\sqrt{2} \\
1/\sqrt{2} & -1/\sqrt{2}
\end{bmatrix}
\]

Typically, the singular values arranged in decreasing order.
Visualization of multiplying with $A$

Figure A.1 The action of a matrix $A$ can be visualized by thinking of the domain as being spanned by a set of orthonormal vectors $v_j$, each of which is transformed to a new orthogonal vector $u_j$ with a length $\sigma_j$. When $A$ is interpreted as a covariance matrix and its eigenvalue decomposition is performed, each of the $u_j$ axes denote a principal direction (component) and each $\sigma_j$ denotes one standard deviation along that direction.
Applications of SVD in Linear Algebra

• Inverse of a $n \times n$ square matrix, $A$
  
  - If $A$ is non-singular, then $A^{-1} = (U\Lambda V^T)^{-1} = V\Lambda^{-1}U^T$ where $\Lambda^{-1} = \text{diag}(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_n)$
  
  - If $A$ is singular, then $A^{-1} = (U\Lambda V^T)^{-1}  \frac{1}{\lambda} V\Lambda_0^{-1}U^T$ where $\Lambda_0^{-1} = \text{diag}(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_p, 0, 0, \ldots, 0)$

• Least squares solutions of a $m \times n$ system

  - $Ax = b$ (A is $m \times n$, $m \geq n$) = $(A^T A)x = A^T b$  
  
  - $x = (A^T A)^{-1} A^T b = A^+ b$

  - If $A^T A$ is singular, $x = A^+ b = (V\Lambda_0^{-1}U^T)b$ where $\Lambda_0^{-1} = \text{diag}(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_p, 0, 0, \ldots, 0)$

• Condition of a matrix

  - Condition number measures the degree of singularity of $A$
    
    - Larger the value of $\lambda_1/\lambda_n$, closer $A$ is to being singular

Line Fitting:

- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares

- What if there are outliers?
  - Robust fitting, RANSAC

- What if there are many lines?
  - Voting methods: RANSAC, Hough transform

- What if we’re not even sure it’s a line?
  - Model selection
Linear equations solving with Standard Least Squares (LS)

\[ Ax = b \]

\[
\begin{bmatrix}
1 & 2 \\
1 & 3 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
4 \\
5 \\
6
\end{bmatrix}
\]

\[ e^2 = ||Ax - b||^2 \]

\[ A^T Ax = A^T b \]

\[ \bar{x} = (A^T A)^{-1} A^T b \]

- If \( A \) has linearly independent columns, \( A^T A \) is square, symmetric and invertible

\[ A^\dagger = (A^T A)^{-1} A^T \]

is so called pseudoinverse of matrix \( A \)
Homogeneous Systems of equations

\[ Ax = 0 \]

There is a unique trivial solution \( x = 0 \)

We need to impose some constraint to avoid trivial Solution, for example

\[ \| x \| = 1 \]

Find such \( x \) that \( \| Ax \|^2 \) is minimized

\[ \| Ax \|^2 = x A^T A x \]

Solution: eigenvector associated with the smallest eigenvalue of \( A^T A \)

or singular vector associated with smallest singular value

\[ A = UΣV^T \quad \text{Since singular values are sorted from large to small, this is the last column of \( V \) (last row of \( V^T \))} \]
Least squares line fitting

Data: \((x_1, y_1), \ldots, (x_n, y_n)\)

Line equation: \(y_i = mx_i + b\)

Find \((m, b)\) to minimize

\[
E = \sum_{i=1}^{n} (y_i - mx_i - b)^2
\]

\[
Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad B = \begin{bmatrix} m \\ b \end{bmatrix}
\]

\[
E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)
\]

\[
\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0
\]

\[
X^T XB = X^T Y
\]

**Normal equations:** Least Squares solution to

\[
XB = Y
\]

\[
B = (X^T X)^{-1} (X^T Y)
\]
Problem with “vertical” least squares

• Not rotation-invariant
• Fails completely for vertical lines
Total least squares  (or Orthogonal Least Squares)

Distance between point \((x_i, y_i)\) and line \(ax + by = d\) (\(a^2 + b^2 = 1\)): \(|ax_i + by_i - d|\)

Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]

**Not stable** to find three values (which could have eigenvalues of very different value; small eigenvalues are critical)

Instead, since the best fit line must pass through the center of mass of the set of points, center the data (move mass center to origin).

This reduces the problem to finding 2 parameters (the unit vector normal to the line). Then solve for \(d\).
Total least squares

Distance between point \((x_i, y_i)\) and line \(ax + by = d\) \((a^2 + b^2 = 1)\): \(|ax_i + by_i - d|\)

Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]

\[
\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0
\]

\[
E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2
\]

\[
\frac{dE}{dN} = 2(U^TU)N = 0
\]

Solution to \((U^TU)N = 0\), subject to \(|N|^2 = 1\): eigenvector of \(U^TU\) associated with the smallest eigenvalue (least squares solution to homogeneous linear system \(UN = 0\))
Total least squares

\[
U = \begin{bmatrix}
  x_1 - \bar{x} & y_1 - \bar{y} \\
  \vdots & \vdots \\
  x_n - \bar{x} & y_n - \bar{y}
\end{bmatrix} \quad U^T U = \begin{bmatrix}
  \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\
  \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2
\end{bmatrix}
\]

second moment matrix
Total least squares

\[ U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix} \]
Least squares: Robustness to noise

Least squares fit to the red points:
Least squares: Robustness to noise

Least squares fit with an outlier:

Problem: squared error heavily penalizes outliers
Least Squares Fitting with Regularization

Noise is amplified by inverse of $\Sigma$ (matrix of eigenvalues)

Sensitivity of a matrix is characterized by condition number (large condition number causes large noise amplification)

$$K = \frac{\sigma_{max}}{\sigma_{min}} = \left\| \frac{\lambda_{max}}{\lambda_{min}} \right\|$$

Instead of solving

$$\text{arg min}_x \|b - Ax\|$$

We solve

$$\text{arg min}_x \|b - Ax\| + \lambda\|x\|^2$$

Solution

$$x = (A^T A + \lambda I)^{-1} A^T b$$
Covariance od Data in HW1
Fitting with LS, TLS and LS + Regularization
RANSAC

• Robust fitting can deal with a few outliers – what if we have very many?
• Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers

Outline
• Choose a small subset of points uniformly at random
• Fit a model to that subset
• Find all remaining points that are “close” to the model and reject the rest as outliers
• Do this many times and choose the best model

RANSAC for line fitting example

Source: R. Raguram
RANSAC for line fitting example

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize–and–verify loop

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Uncontaminated sample

Source: R. Raguram
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
RANSAC for line fitting

Repeat $N$ times:

• Draw $s$ points uniformly at random
• Fit line to these $s$ points
• Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than $t$)
• If there are $d$ or more inliers, accept the line and refit using all inliers
Choosing the parameters

• **Initial number of points** $s$
  - Typically minimum number needed to fit the model

• **Distance threshold** $t$
  - Choose $t$ so probability for inlier is $p$ (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. $\sigma$: $t^2 = 3.84\sigma^2$

• **Number of samples** $N$
  - Choose $N$ so that, with probability $p$, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: $e$)

Source: M. Pollefeys
Choosing the parameters

- **Initial number of points** $s$
  - Typically minimum number needed to fit the model

- **Distance threshold** $t$
  - Choose $t$ so probability for inlier is $p$ (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. $\sigma$: $t^2 = 3.84\sigma^2$

- **Number of samples** $N$
  - Choose $N$ so that, with probability $p$, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: $e$)

\[
\left(1 - (1-e)^s\right)^N = 1 - p
\]

\[
N = \log(1 - p)/\log\left(1 - (1-e)^s\right)
\]

<table>
<thead>
<tr>
<th>proportion of outliers $e$</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>34</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>17</td>
<td>26</td>
<td>57</td>
<td>146</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>16</td>
<td>24</td>
<td>37</td>
<td>97</td>
<td>293</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>8</td>
<td>20</td>
<td>33</td>
<td>54</td>
<td>163</td>
<td>588</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>26</td>
<td>44</td>
<td>78</td>
<td>272</td>
<td>1177</td>
</tr>
</tbody>
</table>

Source: M. Pollefeys
Choosing the parameters

• Initial number of points $s$
  • Typically minimum number needed to fit the model

• Distance threshold $t$
  • Choose $t$ so probability for inlier is $p$ (e.g. 0.95)
  • Zero-mean Gaussian noise with std. dev. $\sigma$: $t^2 = 3.84\sigma^2$

• Number of samples $N$
  • Choose $N$ so that, with probability $p$, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: $e$)

\[
\left(1 - \left(1 - e\right)^s\right)^N = 1 - p
\]

\[
N = \log(1 - p) / \log\left(1 - \left(1 - e\right)^s\right)
\]

Source: M. Pollefeys
Choosing the parameters

• **Initial number of points** $s$
  • Typically minimum number needed to fit the model

• **Distance threshold** $t$
  • Choose $t$ so probability for inlier is $p$ (e.g. 0.95)
  • Zero-mean Gaussian noise with std. dev. $\sigma$: $t^2 = 3.84\sigma^2$

• **Number of samples** $N$
  • Choose $N$ so that, with probability $p$, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: $e$)

• **Consensus set size** $d$
  • Should match expected inlier ratio

Source: M. Pollefeys
Adaptively determining the number of samples

• Inlier ratio $e$ is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield $e=0.2$

• Adaptive procedure:
  • $N=\infty$, $sample\_count = 0$
  • While $N > sample\_count$
    – Choose a sample and count the number of inliers
    – Set $e = 1 - (\text{number of inliers})/(\text{total number of points})$
    – Recompute $N$ from $e$:
      \[
      N = \log\left(\frac{1 - p}{1 - (1 - e)^r}\right)
      \]
    – Increment the $sample\_count$ by 1
RANSAC pros and cons

• Pros
  • Simple and general
  • Applicable to many different problems
  • Often works well in practice

• Cons
  • Lots of parameters to tune
  • Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  • Can’t always get a good initialization of the model based on the minimum number of samples