Model Fitting

CS6240 Multimedia Analysis

Leow Wee Kheng

Department of Computer Science
School of Computing
National University of Singapore
Model fitting fits a model to input data.

- It is equivalent to registration.
- A model can be parametric or non-parametric.
- Input data can be data points or images.

Let’s start with fitting of lines, planes, and curves to data points.
Fitting of line in 2-D space can be stated as follows:

*Given data points \((x_i, z_i), i = 1, \ldots, n,\) determine the line (i.e., linear function) \(f(x)\) that minimizes the error \(z - f(x)\).*

This is also called **linear regression**.
Equation of a line in 2-D space is

\[ z = a_1 x + a_2. \]  

(1)

The difference \( e_i = z_i - (a_1 x_i + a_2) \) is the fitting error.

So, the problem is to find the line, parameterized by \( a_1 \) and \( a_2 \), that minimizes the sum-squared error \( E \)

\[ E = \sum_i \| z_i - (a_1 x_i + a_2) \|^2. \]  

(2)

To find the least-squared fitting line, combine Eq. 1 for all \( i \):

\[
\begin{bmatrix}
  z_1 \\
  \vdots \\
  z_n
\end{bmatrix} =
\begin{bmatrix}
  x_1 & 1 \\
  \vdots & \vdots \\
  x_n & 1
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2
\end{bmatrix}
\]  

(3)
which can be written as

\[ Z = MA \]  \hspace{1cm} (4)

where

\[
Z = \begin{bmatrix}
z_1 \\ 
\vdots \\ 
z_n 
\end{bmatrix}, \quad M = \begin{bmatrix}
x_1 & 1 \\ 
\vdots & \vdots \\ 
x_n & 1 
\end{bmatrix}, \quad A = \begin{bmatrix}
a_1 \\ 
a_2 
\end{bmatrix}.
\]

Then, the least-square solution for \( A \) is given by

\[ A = (M^\top M)^{-1} M^\top Z. \]  \hspace{1cm} (5)
Plane Fitting

The same method can be applied to the fitting of planes in 3-D space.

Equation of planes in 3-D space:

\[ z = a_1 x + a_2 y + a_3. \]  \hspace{1cm} (6)

Then, we obtain

\[
\begin{bmatrix}
  z_1 \\
  \vdots \\
  z_n 
\end{bmatrix} =
\begin{bmatrix}
  x_1 & y_1 & 1 \\
  \vdots & \vdots & \vdots \\
  x_n & y_n & 1 
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 
\end{bmatrix}
\]  \hspace{1cm} (7)

which can also be written as

\[ Z = MA. \]  \hspace{1cm} (8)
Curved Surface Fitting

Consider data points \((x_i, y_i, z_i)\) that define a curved surface in 3-D space.

A simple way to represent the curved surface is by means of an order-\(m\) polynomial function:

\[
z = \sum_{k} \sum_{l} a_{kl} x^k y^l, \quad 0 \leq k + l \leq m.
\]
For example, a 2nd-order polynomial function is

\[ z = a_{20}x^2 + a_{02}y^2 + a_{11}xy + a_{10}x + a_{01}y + a_{00}. \]  

(10)

Combining the equation for all \( i \) gives

\[
\begin{bmatrix}
z_1 \\
n \\
z_n
\end{bmatrix} = 
\begin{bmatrix}
x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\
n \\
x_n^2 & y_n^2 & x_ny_n & x_n & y_n & 1
\end{bmatrix}
\begin{bmatrix}
a_{20} \\
\vdots \\
a_{00}
\end{bmatrix}
\]  

(11)

which can also be written as

\[ Z = MA. \]  

(12)

So, fitting of polynomial curved surface in 3-D space is easy!

Exercise: Fit a polynomial curve in 2-D space.
Another general way to represent a curved surface is by means of basis functions $h_k(x, y)$, $k = 1, \ldots, K$:

$$z = \sum_{k=1}^{K} a_k h_k(x, y).$$  \hspace{1cm} (13)

- Usually use radially symmetric functions called radial basis functions.
- Usually want basis functions to have finite support, i.e., have non-zero value over small domain.
- Examples: Gaussian, splines, wavelets, etc.
Previous sections illustrate fitting of models to data points.

What about fitting models to image content?

This is more tedious because

- digital image is a discrete object, and
- image content can be very complex.

Here, we look at some simple examples as illustrations.
Point Fitting

The simplest example is to fit a model to a point in the image.

How does a point look like in an image?

(a) A point.  (b) The enlarged image of a point.

- A point usually occupies more than one pixel.
- A point does not have sharp edges. The edges are smooth or blurred.
An appropriate model of a point is 2D Gaussian.

\[ g(x, y) = \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]  

(14)
So, a point can be modeled by the 2D Gaussian $G$ as follows:

$$G(x; \theta) = G(x; A, B, \sigma, u) = A + B \exp \left(-\frac{\|x - u\|^2}{2\sigma^2}\right)$$  \hspace{1cm} (15)

- $x = (x, y)$: any location in the image.
- $A$: intensity of background (dark region).
- $B$: peak intensity of point (brightest region).
- $u = (u, v)$: peak location, i.e., center of point.
- $\sigma$: amount of spread of the Gaussian.
- $\theta = (A, B, \sigma, u, v)^\top$: parameters of the point model.
If the model $G$ matches a point in image $I$ perfectly, then

$$G(x; \theta) = I(x)$$

(16)

for all locations $x$ within the model $G$.

So, the point fitting problem is to find the parameters $\theta$ that minimize the error $E$:

$$E(\theta) = \frac{1}{2} \sum_{x \in W} (G(x; \theta) - I(x))^2$$

(17)

where $W$ is the extent of $G$ (like a small window or template).
Since $E$ is differentiable, we can compute the derivatives of $E$ with respect to the parameters, $\partial E / \partial \theta = (\partial E / \partial A, \ldots, \partial E / \partial v)^\top$.

\[
\begin{align*}
\frac{\partial E}{\partial A} &= \sum_{x \in W} (G(x; \theta) - I(x)) \\
\frac{\partial E}{\partial B} &= \sum_{x \in W} (G(x; \theta) - I(x)) \exp \left( -\frac{\|x - u\|^2}{2\sigma^2} \right) \\
\frac{\partial E}{\partial u} &= \sum_{x \in W} (G(x; \theta) - I(x)) \exp \left( -\frac{\|x - u\|^2}{2\sigma^2} \right) \frac{x - u}{\sigma^2} \\
\frac{\partial E}{\partial \sigma} &= \sum_{x \in W} (G(x; \theta) - I(x)) \exp \left( -\frac{\|x - u\|^2}{2\sigma^2} \right) \frac{\|x - u\|^2}{\sigma^3}
\end{align*}
\]
It is difficult to solve $\partial E/\partial \theta = 0$ analytically for the minimum. In this case, we can apply \textbf{gradient descent} method as follows:

Initialize $\theta$.
Repeat until convergence

update $\theta$ in the direction of negative gradient of $E$:

$$\Delta \theta = -\eta \frac{\partial E}{\partial \theta}$$  \hspace{1cm} (19)

i.e.,

$$\theta(t + 1) = \theta(t) - \eta \frac{\partial E}{\partial \theta}.$$  \hspace{1cm} (20)

- $\eta$ is a constant update rate.
- $\eta$ should be small to ensure stability and convergence.
- There are many possible convergence criteria, e.g., $E(t + 1) - E(t)$ is small enough.
Intuitive Idea of Gradient Descent

Consider $\frac{\partial E}{\partial A}$. Assume $B = 0$ for now.

- When $A > I$, $\Delta A < 0$, $A$ is decreased towards $I$.
- When $A < I$, $\Delta A > 0$, $A$ is increased towards $I$.

Notes:

- Gradient descent changes the parameters towards the desired values.
- Can use this to check whether the signs of the equations are correct.
- In theory, when the desired values are attained, the error $E$ is minimized.
- In practice, gradient descent can get trapped in local minimum.
Example: Fit a Gaussian model to a point in the image.

Actual parameter values: \((A, B, u, v, \sigma) = (10, 170, 4.4, 3.7, 1.8)\)
Initial estimate \(\theta(0): (A, B, u, v, \sigma) = (0, 100, 5, 3, 1)\)

Optimization error \(E(\theta)\) over iteration \(t\):
Error of position \((u, v)\) over iteration \(t\): 

- Need a good initial estimate. Otherwise, algorithm can get trapped in a **local minimum**.
- Since \((u, v)\) can be real-valued, model fitting produces a **sub-pixel localization** of the point in the image.
Edge Fitting

Edge fitting can be performed in a similar manner. An edge is defined by a change of intensity, which is a blurred step function.
Edge Model

- \((O, x, y)\) is the global coordinate system of the image.
- \((O', x', y')\) is the local coordinate system in which the edge is defined.
A unit step function is defined as

\[ U(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \]

An ideal 2-D step edge \( S \) located at \( O' \) in the coordinate system \((O', x', y')\) along the \( y' \)-axis is given by

\[ S(x', y') = U(x') . \]
A 2-D blurred edge $F$ can be modeled by convolving the 2-D step edge $S$ with a 1-D Gaussian $G$ across the edge:

$$F(x', y'; \sigma) = \int G(w; \sigma) S(x' - w, y') \, dw$$  \hspace{1cm} (22)

where

$$G(w; \sigma) = \exp \left( -\frac{w^2}{2\sigma^2} \right)$$  \hspace{1cm} (23)
$O'$ is located at $(u, v)$ of the global coordinate system.

So, transform edge from local system to global system:

$$
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  (x - u) \cos \theta + (y - v) \sin \theta \\
  -(x - u) \sin \theta + (y - v) \cos \theta
\end{bmatrix}
$$

(24)

Let the gray level on the darker side be $A$ and the gray level on the brighter side be $B$. Then, the final edge model $M$ is:

$$
M(x, y; \theta) = A + BF(x', y'; \sigma)
$$

(25)

where $\theta = (u, v, \theta, \sigma, A, B)^T$ is the parameter vector.
Can compute the error of match $E(\theta)$ as

$$E(\theta) = \sum_{x \in W} (M(x; \theta) - I(x))^2$$

(26)

where $W$ is the extent of $M$.

Now, apply algorithm to find the $\theta$ that minimizes $E(\theta)$. In this case, it is not so easy to differentiate $E$ wrt $\theta$.

One method is to apply Powell’s direction set algorithm [PTVF92].

- Powell’s algorithm does not require the user to provide the partial derivatives of $E$.
- It estimates the partial derivative numerically.
Fitting Other Models

- Other parametric models such as corners can be fitted to the image in a similar way. (Exercise, [DB93, DG90])
- Fitting general models is more complex.
- Two kinds of models:
  - Parametric: represented by parametric but non-analytic equations.
  - Non-parametric: represented by a set of discrete data points.
The previous method minimizes error of one of the coordinates ($z$).

What if you want to minimize the perpendicular distances of the points to the line (or plane)?

Need to compute the distance of a point to a line (or plane).
Equation of hyperplane \( \pi \) (line in 2-D, plane in 3-D, etc.):

\[
a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + a_{n+1} = 0.
\]  

(27)

Denote \( \mathbf{a} = (a_1 \cdots a_n)^\top \), \( \mathbf{x} = (x_1 \cdots x_n)^\top \).

Then, equation of \( \pi \) can be written as

\[
\mathbf{a}^\top \mathbf{x} + a_{n+1} = 0
\]

(28)

where \( \mathbf{x} \) is a point on \( \pi \).

Then, the unit vector \( \mathbf{u} \) of \( \pi \) is \( \mathbf{a}/\|\mathbf{a}\| \). (Exercise)
Consider a point $p$ not on $\pi$.

Let $x$ denote the perpendicular projection of $p$ on $\pi$. Perpendicular distance of $p$ to $\pi$, denoted as $d$, is given by

$$d = p^\top u - x^\top u = \frac{p^\top a - x^\top a}{\|a\|} = \frac{a^\top p + a_{n+1}}{\|a\|}.$$  

(29)
So, finding the best-fit hyperplane $\pi$ is to find the $\mathbf{a}$ that minimizes the distance $D$

$$D = \sum_i \left[ \frac{\mathbf{a}^\top \mathbf{p} + a_{n+1}}{\|\mathbf{a}\|} \right]^2.$$  (30)

- This equation is difficult to differentiate.
- Can apply Powell’s direction set algorithm [PTVF92].

Notes:

- So, fitting $n$-D linear or non-linear models to data points in $n + 1$-D space is not too difficult.
- But, fitting other models are not so easy.
- Example: How to fit a curve (1-D object) to data points in 3-D space?
- No convenient analytic form. How to even represent them?
Problems and algorithms discussed:

<table>
<thead>
<tr>
<th>problem</th>
<th>corresp.</th>
<th>model</th>
<th>algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>line, plane</td>
<td>known</td>
<td>linear</td>
<td>linear least square</td>
</tr>
<tr>
<td>curve, curved surface</td>
<td>known</td>
<td>polynomial, RBF</td>
<td>linear least square</td>
</tr>
<tr>
<td>line, plane</td>
<td>unknown</td>
<td>linear</td>
<td>Powell</td>
</tr>
<tr>
<td>point in image</td>
<td>unknown</td>
<td>Gaussian</td>
<td>gradient descent</td>
</tr>
<tr>
<td>edge in image</td>
<td>unknown</td>
<td>Step</td>
<td>Powell</td>
</tr>
</tbody>
</table>

- Fitting a model to input data is equivalent to registering the model to the input data.
- Model fitting is also equivalent to model learning.
  After fitting, you get a model that fits the data.
- Usually need good initialization to get good optimization results.
Exercise

(1) Develop a method for fitting a polynomial curve in 2-D space.

(2) Develop a method for fitting a corner model to a corner in an image.

(3) The equation of a hyperplane \( \pi \) is given by

\[
a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + a_{n+1} = 0. \tag{31}
\]

Denote \( \mathbf{a} = (a_1 \cdots a_n)^\top \), \( \mathbf{x} = (x_1 \cdots x_n)^\top \). Show that the unit vector \( \mathbf{u} \) of \( \pi \) is \( \mathbf{a}/\|\mathbf{a}\| \).
R. Deriche and T. Blaszka.
Recovering and characterizing image features using an efficient model based approach.

http://citeseer.nj.nec.com/deriche94recovering.html.

R. Deriche and G. Giraudon.
Accurate corner detection: An analytical study.

Reference II


*Numerical Recipes in C.*