Chapter 8. Template Matching

Chang-Su Kim

Many slides are modified from Serigos Theodoridis’s own notes.
Template Matching

Given a set of reference patterns, known as templates, determine which one matches a test pattern best.

Note: each class is represented by a single typical pattern

Ex) aplke (test pattern)
Distance between Sequences

- Reference pattern: \( r(i), i = 1, ..., I \)
- Test pattern: \( t(j), j = 1, ..., J \)
- Path:
  \[
  (0,0) = (i_0, j_0), (i_1, j_1), ..., (i_f, j_f) = (I, J)
  \]
- Path cost
  \[
  D = \sum_{k=1}^{f} \frac{d(i_k, j_k | i_{k-1}, j_{k-1})}{
  
  \]
- The distance is defined as \( D_{\text{min}} \) over all path
Bellman’s Optimality Principle

• Optimal Path

\[(i_0, j_0) \overset{\text{opt}}{\rightarrow} (i_f, j_f)\]

• Optimal path constrained to pass through \((i, j)\)

\[(i_0, j_0) \overset{\text{opt s.t (i,j)}}{\rightarrow} (i_f, j_f)\]

• Bellman’s principle

\[ (i_0, j_0) \overset{\text{opt s.t (i,j)}}{\rightarrow} (i_f, j_f) = (i_0, j_0) \overset{\text{opt}}{\rightarrow} (i, j) \oplus (i, j) \overset{\text{opt}}{\rightarrow} (i_f, j_f) \]
Dynamic Programming

\[ D_{\text{min}}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} D_{\text{min}}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1}) \]

- Local constraints: there is a set of allowed predecessors.
Example 8.1
Figure 8.3 shows the optimal paths (black lines) to reach the nodes at step $k = 3$ starting from the nodes at step $k = 0$. The grid contains three nodes per step. Only the optimal paths, up to step $k = 3$, have been drawn. The goal of this example is to extend the previous paths to the next step and compute the optimal paths terminating at the three nodes at step $k = 4$. Bellman's principle will be employed. Assume that the accumulated costs of the optimal paths $D_{\text{min}}(3,j_3), j_3 = 0, 1, 2$ at the respective nodes are:

$$D_{\text{min}}(3, 0) = 0.8, D_{\text{min}}(3, 1) = 1.2, D_{\text{min}}(3, 2) = 1.0$$ (8.2)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>(4, 0)</th>
<th>(4, 1)</th>
<th>(4, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 0)</td>
<td>0.8</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>(3, 1)</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>(3, 2)</td>
<td>0.7</td>
<td>0.2</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Edit Distance

• Typing errors for “beauty”
  – Wrongly identified symbol (e.g. “befuty”)
  – Insertion error (e.g. “bearuty”)
  – Deletion error (e.g. “beuty”)

• Edit distance between two strings $A$ and $B$
  
  $D(A, B) = \min_j [C(j) + I(j) + R(j)]$

  – Minimum total number of changes, insertions, and deletions
  – e.g.
    
    • beuty $\rightarrow$ beauty
    • beuty $\rightarrow$ beaty $\rightarrow$ beauty
    • beuty $\rightarrow$ bety $\rightarrow$ beuty $\rightarrow$ bety $\rightarrow$ beuty $\rightarrow$ beauty
Edit Distance

- Delete
  \[ d(i, j|i, j - 1) = 1 \]
- Insertion
  \[ d(i, j|i - 1, j) = 1 \]
- Diagonal transition
  \[ d(i, j|i - 1, j - 1) = \begin{cases} 0 & r(i) = t(j) \\ 1 & r(i) \neq t(j) \end{cases} \]
Edit Distance (Algorithm)

- Initialization
  - $D(0,0) = D(0,*) = D(*,0) = 0$
- For $i = 1$ to $I$
  - For $j = 1$ to $J$
    - $D(i,j) = \min \begin{cases} 
    D(i-1,j-1) + d(i, j|i-1,j-1) \\
    D(i-1,j) + 1 \\
    D(i,j-1) + 1
  \end{cases}$
  - End For
- End For
- Output $D(I,J)$