Viterbi Algorithm

Renhe W.

Contents

List of Figures

List of Tables

1 Viterbi 算法

维特比算法，也称为维特比译码算法，是一种动态规划技术，用于确定生成给定观察事件序列的最可能的隐藏状态序列，即维特比路径。它在马尔可夫信息源和隐马尔可夫模型（HMM）等领域广泛应用。

“维特比路径”和“维特比算法”这两个术语还用于各种动态规划算法，旨在识别观察结果的最可能解释。例如，在统计句法分析中，动态规划算法可用于发现最可能的上下文无关派生（句法树）的字符串，有时称为“维特比分析”。

维特比算法最初由安德鲁·维特比（Andrew Viterbi）于 1967 年提出，用于解码数字通信系统中的卷积码，以减小噪音的影响。此算法在许多领域得到广泛应用，包括 CDMA 和 GSM 蜂窝网络、拨号调制解调器、卫星通信、深空通信以及 802.11 无线网络，用于解码卷积码。此外，它经常用于语音识别、关键词检测、计算语言学和生物信息学等任务。例如，在语音识别领域，其中音频信号被视为观察到的事件序列，而文本字符串被视为生成观察到的音频信号的潜在原因，因此维特比算法用于识别与音频信号相关的最可能的文本字符串。

1.1 算法步骤

Viterbi 算法

目的: 给定一个观测序列和一个隐马尔可夫模型 (HMM)，找到最有可能的隐状态序列。

输入:

- 观测序列: $O = \{o_1, o_2, \ldots, o_T\}$
- 状态转移概率矩阵: P
- 观测概率矩阵: B
- 初始状态概率: π

输出:

- 最有可能的隐状态序列: $S^* = \{s_1^*, s_2^*, \ldots, s_T^*\}$

1.2 伪代码

首先是一些问题必要的设置。设观察值空间为 $O = \{o_1, o_2, \ldots, o_N\}$ 、状态空间为 $S = \{s_1, s_2, \ldots, s_K\}$ 、观察值序列为 $Y = \{y_1, y_2, \ldots, y_T\}$, A 为 $K \times K$ 转移矩阵，其中 A_{ij} 为从状态 s_i 转移到 s_j 的转移概率、B 为 $K \times N$ 放射矩阵 (emission matrix)，其中 B_{ij} 为在状态 s_i 观察到 o_j 的概率、大小为 K 的初始概率数组 π, π_i 为 $x_1 == s_i$

的概率。我们称路径 $X = \{x_1, x_2, \ldots, x_T\}$ 为生成观察值 $Y = \{y_1, y_2, \ldots, y_T\}$ 的状态序列。

在这个动态规划问题中,我们构造了两个大小为 $K \times T$ 的二维表 T_1, T_2 。T_1 的每个元素,$T_1[i,j]$,保存生成 $Y = \{y_1, y_2, \ldots, y_j\}$ 时最有可能的路径 $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_j\}$ $\hat{x}_j = s_i$ 的概率。T_2 的每个元素, $T_2[i,j]$, 保存最有可能路径 $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{j-1}, \hat{x}_j\}$, $\forall j, 2 \leq j \leq T$ 的 $\hat{x}_{j-1}$ 。

我们按 $K \cdot j + i$ 增序填充两个表 $T_1[i,j], T_2[i,j]$ 。

$$T_1[i,j] = \max_k \left(T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j}\right),$$
$$T_2[i,j] = \arg\max_k \left(T_1[k, j-1] \cdot A_{ki} \cdot B_{iy_j}\right)$$

输入

- 观察空间 $O = \{o_1, o_2, \ldots, o_N\}$,
- 状态 $S = \{s_1, s_2, \ldots, s_K\}$,
- 观察序列 $Y = \{y_1, y_2, \ldots, y_T\}$ 若在 t 时间观察值为 o_i , 则 $y_t == i$,
- 大小为 $K \cdot K$ 的转移矩阵 A, A_{ij} 为从状态 s_i 到 s_j 的转移概率,
- 大小为 $K \cdot N$ 的放射矩阵 B, B_{ij} 为状态 s_i 观察到 o_j 的概率,
- 大小为 K 的初始概率数组 π, π_i 为 $x_1 == s_i$ 的概率,

输出

- 最有可能的隐含状态序列 $X = \{x_1, x_2, \ldots, x_T\}$

具体的算法伪代码如下:

Algorithm 1 VITERBI 算法

function VITERBI(O, S, π, Y, A, B)
  for each state s_i **do**
    $T_1[i, 1] \leftarrow \pi_i \cdot B_{i,y_1}$
    $T_2[i, 1] \leftarrow 0$
  end for
  for $i \leftarrow 2, 3, \ldots, T$ **do**
    for each state s_j **do**
      $T_1[j, i] \leftarrow \max_k \left(T_1[k, i-1] \cdot A_{k,j} \cdot B_{j,y_i}\right)$
      $T_2[j, i] \leftarrow \arg\max_k \left(T_1[k, i-1] \cdot A_{k,j} \cdot B_{j,y_i}\right)$
    end for
  end for
  $z_T \leftarrow \arg\max_k \left(T_1[k, T]\right)$
  $x_T \leftarrow s_{z_T}$
  for $i \leftarrow T, T-1, \ldots, 2$ **do**
    $z_{i-1} \leftarrow T_2[z_i, i]$
    $x_{i-1} \leftarrow s_{z_{i-1}}$
  end for
  return X
end function

2 Examples

2.1 Markov Switching 模型示例

假设我们有两个状态：一个低均值状态和一个高均值状态。- 隐状态集合: $S = \{1, 2\}$ - 观测值: $O = \{o_1, o_2, \ldots, o_T\}$ 是实数值时间序列数据

状态转移概率矩阵 P 为:

$$P = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

我们假设在状态 1 (低均值状态) 时，观测值的均值为 2 ，标准差为 1 ; 而在状态 2 (高均值状态) 时，观测值的均值为 5 ，标准差为 1。

现在，假设我们有以下观测序列: $O = \{2.1, 5.2, 5.0, 2.3, 2.0, 5.1\}$ 。我们的目标

是使用 Viterbi 算法确定最有可能的状态链。

使用 Viterbi 算法：

1. 初始化:

使用初始状态概率和第一个观测值的概率密度函数来初始化每个状态的概率。

2. 递归计算:

对于每个时间点 t，对于每个状态 j，计算:

$$V[t,j] = \max_i \left(V[t-1,i] \times P[i,j] \times \text{PDF}\left(o_t \mid \text{ state } j\right)\right)$$

其中，PDF 表示概率密度函数，它是在给定状态时观测值的概率。

3. 终止:

在最后一个时间点 T，找到具有最大概率的状态。

4. 回溯路径:

从最后一个时间点开始，使用回溯指针回溯到第一个时间点，确定整个状态链。

```
% Define the state transition probability matrix and observation
matrices
P = [0.8, 0.2; 0.3, 0.7];
mean_vals = [2, 5];  % means for states 1 and 2 respectively
std_devs = [1, 1];  % standard deviations for states 1 and 2

% Define the observed sequence
O = [2.1, 5.2, 5.0, 2.3, 2.0, 5.1];
T = length(O);
num_states = 2;

% Initialization
V = zeros(T, num_states);
ptr = zeros(T, num_states);

for i = 1:num_states
V(1, i) = normpdf(O(1), mean_vals(i), std_devs(i));
end

% Recursive computation
for t = 2:T
  for j = 1:num_states
    % Calculate the maximum probability and the state that resulted
in this max probability
    [max_prob, argmax_state] = max(V(t-1, :) .* P(:, j)' .* normpdf(
```

```
    O(t), mean_vals(j), std_devs(j)));
24
25          V(t, j) = max_prob;
26          ptr(t, j) = argmax_state;
27        end
28      end
29
30      % Backtracking the most probable path
31      states = backtrack(V, ptr);
32
33      % Backtrack function
34      function states = backtrack(V, ptr)
35          T = size(V, 1);
36          [~, states(T)] = max(V(T, :));
37
38          for t = T-1:-1:1
39          states(t) = ptr(t+1, states(t+1));
40        end
41      end
42
43
```

Listing 1: The Matlab code of Markov