CSCI 270 Homework 6

Qilin Ye

October 31, 2022

1. Rain or Drought

Solution. The algorithm goes as follows, and it should be quite explanatory.

```python
def hw6_1(n, k, p):
    # n: number of days in a year
    # k: number of drought days
    # p: array (size n+1) of probability of rain on each day, where p[0] is not used
   if k > n: return 0
    dp = [[0 for _ in range(n+1)] for _ in range(n+1)]
    dp[0][0] = 1

    for i in range(1, n+1):
        dp[i][0] = dp[i-1][0] * p[i]
        for j in range(1, min(i, n)+1):
            dp[i][j] = dp[i-1][j] * p[i] + dp[i-1][j-1] * (1-p[i])

    # at least k drought days = sum over dp[n][j], j in [k, n]
    return sum(dp[n][j] for j in range(k, n+1))
```

Main idea: maintain a $n \times n$ (1-indexed) array with `dp[i][j]` being

$$\mathbb{P}(i,j) \coloneqq \text{probability of (exactly) } j \text{ droughts in first } i \text{ days.}$$

Intuitively, the base case $\mathbb{P}(0,0) = 1$ is trivially true, and for $1 \leqslant i \leqslant n$, $\mathbb{P}(i,0)$, the probability of first i days all being rainy, by independence, is $\prod_{j=1}^{i} p_i$, which can be re-written as $p_i \cdot \mathbb{P}(i-1,0)$.
Having ruled out the second component being 0, we now look at $\mathbb{P}(i,j)$, where $j \neq 0$. In order to have precisely j drought days among the first i days, either

- There are j drought days in the first $i-1$ days, followed by a rainy day i, or
- There are $j-1$ drought days in the first $i-1$ days, followed by a drought day i.

These two events partition the event $\{j \text{ drought days in first } i \text{ days}\}$, so

$$\mathbb{P}(i,j) = p_i \cdot \mathbb{P}(i-1,j) + (1-p_i)\mathbb{P}(i-1,j-1).$$

Note that the reasoning holds even if $i = j$, which involves $\mathbb{P}(i-1,j) = 0$, since it is impossible to have $j > i-1$ drought days in $i-1$ days. And of course, more generally, $\mathbb{P}(i,j) = 0$ for $i < j$, for otherwise having j drought days in such i days is clearly impossible.

Proof of correctness. We claim `dp[i][j]` $= \mathbb{P}(i, j)$. The $(0, 0)$ entry is set to $0 = \mathbb{P}(0, 0)$ by line 7. Every time we increment i by 1, we first set `dp[i][0]` to be p_i times `dp[i-1][0]` which, using an induction on i, guarantees

$$\texttt{dp[i][0]} = p_i \cdot \texttt{dp[i-1][0]} = p_i \cdot \mathbb{P}(i-1, 0) = \mathbb{P}(i, 0). \tag{*}$$

Finally, for general (i, j) entries, if $j > i$ we skip, as shown in line 14, since in such case $\mathbb{P}(i, j) = 0$ and `dp` also initialized that entry to 0. Otherwise, we update `dp[i][j]` using (*), which is implemented on line 15.

Runtime. $\mathcal{O}(1)$ work for each (i, j) entry, so $\mathcal{O}(n^2)$ work overall, since the algorithm has at most n^2 loops. Finally, everything else — initialization, `dp[0][0]`, and number to return — can be finished in $\mathcal{O}(n^2)$, constant, and $\mathcal{O}(n)$ time, respectively. Thus the total time complexity is $\mathcal{O}(n^2)$.

2. HMM Maximum Likelihood

Solution. Since the parameters are significantly more complicated than usual DP problems previously encountered, we will instead provide a pseudocode, shown below.

```
1  # HMM param: observables 'a' to 'z'
2  # HMM param: transition probabilities p_{v,(v,w)}, where v and w are Markov states; if (v,w) does not exist, fill
       p_{v,(v,w)} to 0 to simply calculations later on
3  # HMM param: emission probabilities q_{v,c}, where v is a state and c an observable
4  # HMM param: staring state start_state
5
6  # param: observed string x[1...T]
7  # define: P[_v][_num] to be the most likelihood of the most likely vertex sequence ending at time _num on
       state/vertex _v.
8  Define P[][], over {states/vertices} × {1, ..., T}, initialized to zeros
9
10 # optimal: define a backtracking array
11 # Define b[][] with same dimension as P, initialized to null pointers
12
13 # initialization
14 Set P[start_state][1] = q_{start_state, x[1]}
15
16 # DP recurrence
17 For each t in 2, ..., T:
18    For each vertex v:
19       P[v][t] = max_{state s} P[s][t-1] · p_{s,(s,v)} · q_{v,x[t]}
20       # optimal backtracking
21       # b[v][t] = argmax_{state s} P[s][t-1] · p_{s,(s,v)} · q_{v,x[t]}
22
23 # maximum likelihood:
24 max_likelihood = max_{state s} P[s][T].
25 # backtrack:
26 # traverse through b[][], with replacement b[s][t] -> b[b[s][t]][t-1], until first argument is null
```

Main idea. We first consider the general case when $t \geqslant 2$. By definition

$$\begin{aligned}\text{lik}(v_1, ..., v_t) &= \prod_{j=1}^{t} q_{v_j,x_j} \prod_{j=1}^{t-1} p_{v_j,(v_j,v_{j+1})} \\ &= \left[\prod_{j=1}^{t-1} q_{v_j,x_j} \prod_{j=1}^{t-2} p_{v_j,(v_j,v_{j+1})}\right] q_{v_t,x_t} \cdot p_{v_{t-1},(v_{t-1},v_t)} \\ &= \text{lik}(v_1, ..., v_{t-1}) \cdot p_{v_{t-1},(v_{t-1},v_t)} \cdot q_{v_t,x_t}.\end{aligned}$$

Therefore, if we keep v_t fixed, treating the likelihood as a function of $t-1$ variables,

$$\max_{v_1,...,v_{t-1}} \text{lik}(v_1, ..., v_t) = \max_{v_1,...,v_{t-1}} \text{lik}(v_1, ..., v_{t-1}) \cdot p_{v_{t-1},(v_{t-1},v_t)} \cdot q_{v_t,x_t}. \tag{**}$$

From (**), we have mathematically shown the general recurrence relation needed for this problem. In particular, if we already know the optimal length $t-1$ vertex sequence ending on any state, then the maximum likelihood of a length t sequence ending with v_t depend only on:

- Maximal likelihoods of length $t-1$ sequences ending on any state,
- The emission probability q_{v_t,x_t}, and
- The transition probabilities $p_{v_{t-1},(v_{t-1},v_t)}$ from any vertex to v_t.

In the code, we have padded all non-existent transition probabilities to probability 0, so the above argument is safe to assume all transition probabilities are defined, although some are zero.
For general cases (i.e., when $t \geqslant 2$), since any length t vertex sequence $(v_1, ..., v_t)$ is *some* length $t-1$ sequence $(v_1, ..., v_{t-1})$ with v_t appended, the optimal one is no exception. Therefore, assuming `dp[v][t-1]` stores the optimal vertex sequence of length $t-1$ for all states / vertices `v`, the code at line 19 ensures that we store the maximum likelihood length t vertex sequence ending at v in `dp[v][t]`, according to (**).
For base cases (when $t = 1$), we note that the question dictates the start state, so `P[start_state][1]` is simply the emission probability $q_{\text{start\_state},s[1]}$. The system cannot start from any other state, so `P[s][1] = 0` otherwise.

Proof of correctness. Base case(s) is(are) set up according to the remarks made above, and the recurrence relation is inductively implemented on line 19, whose correctness has been addressed above.

Runtime. Each time line 19 is called, it will make at most $|V|$ comparisons before outputting the maximal one. The loop is repeated $\mathcal{O}(T|V|)$ times, so the loop (lines 17 to 21) takes $\mathcal{O}(T|V|^2)$. The complexity of all other works are also bounded by this quantity: padding nonexistent $p_{v,(v,w)}$ takes at most $|E| \leqslant |V^2|$ repeats, initializing `P[][]` takes $\mathcal{O}(T|V|)$, and returning the maximum likelihood takes at most $\mathcal{O}(|V|)$ time. Hence the total runtime is $\mathcal{O}(T|V|^2)$.