1 Hidden Markov Model (HMM)

Problem 1. We want to estimate the hidden sequence $X^n := (X_0, X_1, \cdots, X_n)$ based on the observation sequence $Y^n := (Y_0, Y_1, \cdots, Y_n)$.

a) What is the probability distribution of $Y^n$ when the hidden markov chain is $X^n = (x_0, \cdots, x_n)$? i.e. what is $P(Y^n | X^n = x)$?

b) What is the distribution of $X^n$? i.e. what is the prior distribution $P(X^n = x)$?

Recall that the maximum a posterior (MAP) estimate is given by

$$\text{MAP}[X^n|Y^n] = \arg \max_{X^n} P(X^n|Y^n) = \arg \max_{X^n} P(Y^n|X^n)P(X^n)$$

c) Using the log function, express the MAP estimate as a shortest path problem. Draw the corresponding graph for the small example above for $n = 2$.

Solution. a) The probability distribution is $P(Y^n | X^n = x) = Q(x_0, Y_0)Q(x_1, Y_1)\cdots Q(x_n, Y_n)$

b) The prior distribution is $P(X^n = x) = \pi_0(x_0)P(x_0, x_1)P(x_1, x_2)\cdots P(x_{n-1}, x_n)$

c) We note that

$$\arg \max_{X^n} P(Y^n|X^n)P(X^n) = \arg \min_{X^n} \{-\log P(Y^n|X^n)P(X^n)\}$$

We have

$$P(Y^n|X^n)P(X^n) = \pi_0(x_0)Q(x_0, Y_0)P(x_0, X_1)Q(X_1, Y_1)\cdots P(x_{n-1}, X_n)Q(X_n, Y_n)$$

hence

$$-\log P(Y^n|X^n)P(X^n) = -\log \pi_0(x_0)Q(x_0, Y_0) + \sum_{k=1}^{n} -\log P(x_{k-1}, X_k)Q(X_k, Y_k)$$

So we want to minimize the length of the path $(X_0, X_1, \cdots, X_n)$, where the length is given by

$$L(X^n) = d(X_0) + \sum_{k=1}^{n} d(X_{k-1}, X_k)$$

For the small example above with $n = 3$, the corresponding graph is:
2 Viterbi algorithm

Problem 2. a) In the small example above, find the MAP estimate of $X^2$ by hand given the observation $Y^2 = \{'normal', 'cold', 'dizzy'\}$. Implement the algorithm on this small example using your favorite language.

b) Express the Viterbi algorithm as a shortest path algorithm.

Solution. a) The multiplicative length of each edge is When we run the viterbi algorithm, we obtain the successive longest paths and values $V_k(u), k = 0, 1, 2, u = \text{'healthy'}, \text{'fever'}$

\[
\begin{array}{c|cccccccccc}
\text{edge} & (S, H_0) & (S, F_0) & (H_0, H_1) & (H_0, F_1) & (F_0, H_1) & (F_0, F_1) & (H_1, H_2) & (H_1, F_2) & (F_1, H_2) & (F_1, F_2) \\
\text{weight} & 0.6*0.5 & 0.4*0.1 & 0.7*0.4 & 0.3*0.3 & 0.4*0.4 & 0.6*0.3 & 0.7*0.1 & 0.3*0.6 & 0.4*0.1 & 0.6*0.6 \\
\text{value} & 0.3 & 0.04 & 0.28 & 0.09 & 0.16 & 0.18 & 0.07 & 0.18 & 0.04 & 0.36 \\
\end{array}
\]

Table 1: Number of modes for a homogeneous road.

A Python implementation of the Viterbi algorithm on the toy example:
HMM.py

states = ('Healthy', 'Fever')

observations = ('normal', 'cold', 'dizzy')

start_probability = {'Healthy': 0.6, 'Fever': 0.4}

transition_probability = {
    'Healthy': {'Healthy': 0.7, 'Fever': 0.3},
    'Fever': {'Healthy': 0.4, 'Fever': 0.6}
}

emission_probability = {
    'Healthy': {'normal': 0.5, 'cold': 0.4, 'dizzy': 0.1},
    'Fever': {'normal': 0.1, 'cold': 0.3, 'dizzy': 0.6}
}

algorithm.py

# Helps visualize the steps of Viterbi.
def print_dptable(V):
    s = " " + " ".join(("%7d" % i) for i in range(len(V))) + "\n"
    for y in V[0]:
        s += "%.5s: " % y
        s += " ".join("%7.5f" % v[y]) for v in V) + "\n"
    print(s)

def viterbi(obs, states, start_p, trans_p, emit_p):
    # Initialize base cases (t == 0)
    V = [y:(start_p[y] * emit_p[y][obs[0]]) for y in states]
    path = {y:[y] for y in states}

    # Run Viterbi for t > 0
    for t in range(1, len(obs)):
        V.append({})
        newpath = {}

        for y in states:
            (prob, state) = max((V[t-1][y0] * trans_p[y0][y] * emit_p[y][obs[t]], y0) for y0 in states)
            V[t][y] = prob
            newpath[y] = path[state] + [y]

        # Don’t need to remember the old paths
        path = newpath
print_dptable(V)
(prob, state) = max((V[t][y], y) for y in states)
return (prob, path[state])

main.py

import HMM as h
import algorithm as a

def example():
    return a.viterbi(h.observations,
                      h.states,
                      h.start_probability,
                      h.transition_probability,
                      h.emission_probability)

print(example())