Hidden Markov Model (HMM)

- In a Hidden Markov Model, the state is not directly visible. Each state has a probability distribution over the possible output tokens (i.e. observations associated with that state).
- Hence, the sequence of tokens/observations generated by an HMM gives some information about the sequence of hidden states.
- E.g. speech – words are states and utterances are the observed tokens:

States and observations

- Let \( X_t \) denotes the set of state variables at time step \( t \), and \( E_t \) denotes the set of (observations) evidence variables at time step \( t \).
  - The concrete output/evidence at time \( t \) is \( E_t = e_t \).
  - The concrete state at time \( t \) is \( X_t = x_t \).
- Assumption: the same set of variables are state variables \( X_t \) and evidence variables \( E_t \), respectively, at each step \( t \).
- We will assume that the state sequence starts at \( t = 0 \).
- We will assume that the evidence sequence starts at \( t = 1 \).

Transmission & observation (emission) model

- We assume each state depends only on a previous state (1st order Markov process), hence the transition probability \( P(X_t \mid X_{t-1}) \) for \( \forall t \).
- We assume the evidence variables at time \( t \) depend only on the current state, hence the emission probability \( P(E_t \mid X_t) \) for \( \forall t \).
- Stationary process: causal laws that govern the process of change of the world do not change over time.

Example of HMM for weather: Bayesian net

- We have one Boolean state variable that can have two values: \textit{Atmospheric Pressure} = \{Low, High\}.
- We cannot observe it directly, but we can observe whether it’s raining or not – so evidence is also a Boolean variable: \textit{Weather} = \{Rain, Dry\}

Example of HMM for weather: probabilities

- Two hidden states: ‘Low’ and ‘High’ atmospheric pressure.
- Two observations: ‘Rain’ and ‘Dry’ (which is negation of Rain).
- Transition probabilities: \( P(\text{‘Low’} \mid \text{‘Low’}) = 0.3 \), \( P(\text{‘High’} \mid \text{‘Low’}) = 0.7 \), \( P(\text{‘Low’} \mid \text{‘High’}) = 0.2 \) and \( P(\text{‘High’} \mid \text{‘High’}) = 0.8 \).
- Observation probabilities: \( P(\text{‘Rain’} \mid \text{‘Low’}) = 0.6 \), \( P(\text{‘Dry’} \mid \text{‘Low’}) = 0.4 \), \( P(\text{‘Rain’} \mid \text{‘High’}) = 0.4 \), \( P(\text{‘Dry’} \mid \text{‘High’}) = 0.3 \).
- In order to perform any kind of probabilistic inference, we need to specify initial (i.e. prior) probabilities of states at time \( t = 0 \):
  - E.g.: \( P(\text{‘Low’}) = 0.4 \), \( P(\text{‘High’}) = 0.6 \).
Example: a 2-state HMM for DNA

- Top: HMM architecture and parameters. (Note: States are abstract!)
  - green: state transition probabilities, red: emission probabilities.
- Bottom: sequence generation. Sequence score = \( \prod P(\text{letter}_t | \text{state}_t) \)

C-style pseudocode for generating output tokens

```c
// Generate a random number between 0 and 1 according to uniform distribution
double unifRand() {
    return rand() / double(RAND_MAX);
}
int main(){
    seed();
    while (state[t] = W){
        rnd = unifRand();
        if (0.0 < rnd <= 0.35) letter[t] = A;
        if (0.35 < rnd <= 0.5) letter[t] = C;
        if (0.5 < rnd <= 0.65) letter[t] = G;
        if (0.65 < rnd <= 1.0) letter[t] = T;
    }
    return(0);
}
```

C-style pseudocode for generating state transitions

```c
// Generate random number (0, 1] according to uniform distrib
double unifRand() {
    return rand() / double(RAND_MAX);
}
int main(){
    seed();  // new random seed
    state[0] = B;
    rnd = unifRand();
    if (0.0 < rnd <= 0.5) state[1] = W;
    else state[1] = S;
    until (state[t] = E){
        rnd = unifRand();
        .. // transition to a new state
        if(state[t] = W){
            rnd = unifRand(); .. // new output token
            t = t + 1;
        }
    }
    return(0);
}
```

Pseudocode for generating state transitions & observations

```c
double unifRand() {
    return rand() / double(RAND_MAX);
}
int main(){
    seed(); // new random seed
    state[0] = B;
    rnd = unifRand();
    if (0.0 < rnd <= 0.5) state[1] = W;
    else state[1] = S;
    until (state[t] = E){
        rnd = unifRand();
        .. // transition to a new state
        if(state[t] = W){
            rnd = unifRand(); .. // new output token
            t = t + 1;
        }
    }
    return(0);
}
```

Example of trained HMM for DNA (lab 5)

- Left: state transition model; Right: emission model for M and I states.
- We have 4 insertion states, 3 match states and 3 deletion states (# of D and I states depends on # of M states).