HMM (Hidden Markov Model)

Example: $n = 2$:

$$P(x_0, y_0, x_1, y_1) = P(x_0) \cdot P(y_0|x_0) \cdot P(x_1|x_0, y_0) \cdot P(y_1|x_1, y_0).$$

MLSE:

$$P(x_0, x_1, \ldots, x_n, y_0, y_1, \ldots, y_n) = \prod_{0}^{n} P(x_0) Q(x_0, y_0) p(x_0|x_0) q(x_1|y_1) \cdots p(x_{n-1}|x_n) q(x_n|y_n).$$

Goal: Find $\text{MAP}[X^n|Y^n = y^n] \Rightarrow \text{Find the best seq of states (hidden that best explains obs. seq.}.$

$$x^{n*} = \arg \max_{x^n} P[X^n = x^n|Y^n = y^n] = \arg \max_{x^n} \prod_{0}^{n} P(x_0) Q(x_0, y_0) / P(x_0, x_1) Q(x_1, y_1) \cdots P(x_{n-1}, x_n) Q(x_n, y_n)$$

$$= \arg \max_{x^n} \left[ \log P(x_0) Q(x_0, y_0) + \sum_{m=1}^{n} \log P(x_{m-1}, x_m) Q(x_m, y_m) \right] \leq -d_{0}(x_0) \leq -d_{m}(x_{m-1}, x_m)$$

$$x^{n*} = \arg \min_{x^n} \left[ d_{0}(x_0) + \sum_{m=1}^{n} d_{m}(x_{m-1}, x_m) \right]$$
MANY FORMS OF INFERENCE

1. FILTERING:
   \[ Y_0, Y_1, \ldots, Y_T \rightarrow \text{Filter} \rightarrow \hat{X}_T \]
   Ex.: Tracking position in real-time
   Ex.: Monitor current health of patient given symptoms \( Y_i, i = 0 \)

2. PREDICTION:
   \[ Y_0, Y_1, \ldots, Y_T \rightarrow \text{Predict} \rightarrow \hat{Y}_{T+1} \]
   Ex.: Radar tracking, stock price prediction

3. SMOOTHING:
   \[ Y_0, Y_1, \ldots, Y_T \rightarrow \text{Smooth} \rightarrow \hat{X}_t \quad (t \leq T) \]
   Ex.: Infer cause of car crash, "post-mortem" analysis

4. Max. Likelihood State Est. (MLSE)
   \[ Y_0, Y_1, \ldots, Y_T \rightarrow \text{MLSE} \rightarrow \{ \hat{X}_0, \hat{X}_1, \ldots, \hat{X}_T \} \]
   Ex.: Conv. coding (Viterbi alg.), auto-spell
Ex: "Nearly Honest" Casino: uses a fair die most of the time but switches to "loaded" die occasionally.

Given an observed seq. of die rolls (e.g. 6, 6, 1, 6), infer the most likely seq. of "hidden" states (e.g. F, F, L, F, L...).
Casino Ex.: “Trellis” diagram

```
FAIR
 do (F)  6  1  2  6  6  6  1
     do (L) do (LFL) do (LFL) do (LFL)
```

**LOADED**

Goal: \[
\min_{x^n} \left[ d_0(x_0) + \sum_{m=1}^{n} d_m(x_{m-1}, x_m) \right]
\]

distance \( (x^n) \)

i.e. find min. length path from stage 0 to stage \( n \)
(Shortest Path problem — can use Bellman-Ford)

**Key idea:**

```
SF ---------- Chicago ---------- NY
        Atlanta
```

Principle of dynamic programming (DP): If shortest path from SF-NY goes through Chicago, then the Chi-NY leg of the journey = shortest path of trip starting at Chi. & ending in N.Y.

:: can recursively compute shortest path
For \( m = 0, 1, \ldots, n \) and \( x \in \mathcal{X} \), let \( V_m(x) \) be the length of the shortest path from state \( X(m) \) to the \( n^{th} \) stage.

Also, let \( V_n(x) = 0 \) for all \( x \in \mathcal{X} \).

\[
V_m(x) = \min_{x' \in \mathcal{X}} \left\{ d_{m+1}(x, x') + V_{m+1}(x') \right\}; \quad m = 0, 1, \ldots, n-1
\]

Finally, let \( V = \min_{x \in \mathcal{X}} \left\{ d_0(x) + V_0(x) \right\} \).

Then, \( V \) is the length of the shortest path.

\[
\begin{array}{ccccccc}
X(0) & X(1) & \ldots & X(k-1) & X(k) & \ldots & X(n) \\
1 & 0 & \ldots & 0 & 0 & \ldots & 1 \\
\end{array}
\]

\( \text{Shortest Path} \)
Alg.
1. Calculate \( \{V_m(x)\} \) recursively for \( m = 0, \ldots, 0 \)
   using (+). At each step record & store "arc" out of each "x" that achieves minimum, i.e. "next-step" info. Say SP from \( x(m) = x \)
   goes to \( x(m+1) = x' \), then store \( S_m(x) = x' \)
   Do this for all \( x \in \mathcal{X} \).
2. Find the value \( x_0^* \) that achieves min. \( \min(\mathcal{A}) \)
3. "Unpack" the MAP shortest-path seq.
   \[
   x_0^*, S_0(x_0^*), S_1(x_1^*), \ldots, S_{n-1}(x_{n-1}^*)
   \]
   \[
   x_0^* \quad x_1^* \quad x_2^* \quad \ldots \quad x_n^*
   \]
Naive comp. cost = \( O(n^2) \)

Cost to find shortest path = \( O(n^2) \)

Cost of a tree = \( O(n^2) \)

\[
\begin{align*}
dm(L) &= \log_{2} p(L, L, \cdot) (1, k_{1}) \quad \text{Initial state} = F_{1} \text{ or } G_{1} \\
dm(F, F) &= \log_{2} p(F, F, \cdot) (F, y_{1}) \\
dm(F, L) &= \log_{2} p(F, L, \cdot) (F, y_{1}) \\
dm(L, F) &= \log_{2} p(L, F, \cdot) (L, y_{1}) \\
\end{align*}
\]