67800 - Probabilistic Methods in Artificial Intelligence **(Due: 06/06/24)**

Project 1 - Representation

Instructor: Prof. Gal Elidan *Name:* Hadar Tal

1 Warmup

2 Sampling

1. Use the sampling procedure described in the recitation to implement in the HMM class the method `sample`.

   Implemented the `sample` method in the `HMM` class.

2. Implement in the HMM class the method `log_joint`, that calculates the joint probability for each sample $p(X_{1:T}, O_{1:T})$.

   Implemented the `log_joint` method in the `HMM` class.

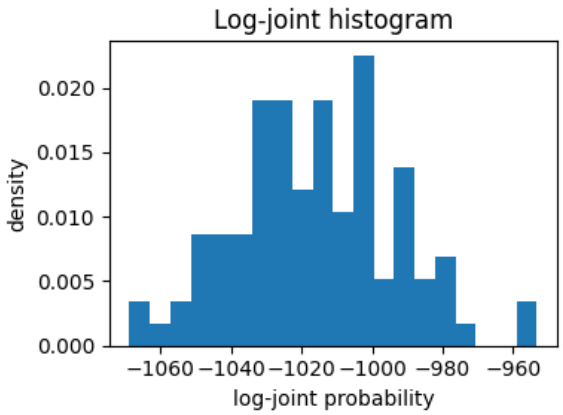
3. Sample $N = 100$ samples $\xi^{[i]} = (X_{1:T}, O_{1:T})$ from the HMM1 model with $T = 1000$. Calculate the joint probability of each sample $p(\xi^{[i]})$.

   Loaded the HMM1 model and sampled $N = 100$ samples from the model with $T = 1000$.

   ```
   Load HMM1. CPDs:
   prior
   ['prior(0)=0.849' 'prior(1)=0.151']
   transition_mat
   [['tau(0->0)=0.750' 'tau(1->0)=0.250']
   ['tau(0->1)=0.250' 'tau(1->1)=0.750']]
   emission_mat
   [['e(0->0)=0.830' 'e(0->1)=0.170']
   ['e(1->0)=0.170' 'e(1->1)=0.830']]
   ```

   The log joint probabilities histogram is shown below.

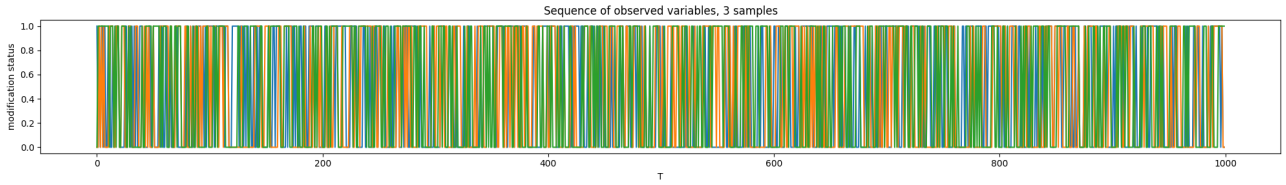
4. Plot the histogram of the log joint probabilities [*]. Why are the joint probabilities so small, even though the samples were sampled from the "correct" model?

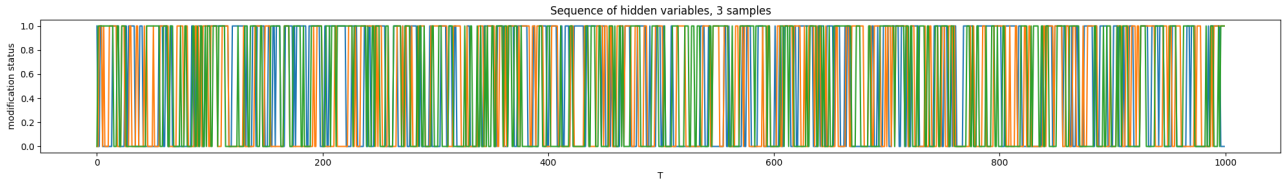


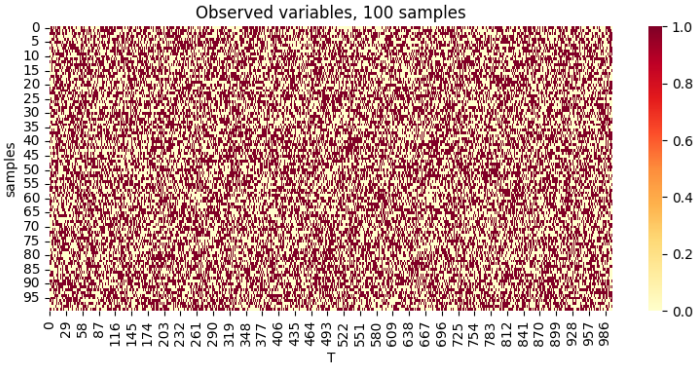
The joint probabilities are so small because they are the product of the prior, transition, and emission probabilities. The transition matrix and the emission matrix are relatively far from the identity matrix. This implies that the randomness (or entropy) in the model is large, meaning that the system transitions

between states frequently and the observations can vary significantly. As a result, each individual sequence of observations and hidden states becomes rare, leading to very small joint probabilities. In essence, the higher the entropy, the more unique each sample is, which decreases the probability of any specific sample occurring.

5. Plot 3 examples of the sampled observations on top of the genome [*]. Plot a heatmap of all samples [*].

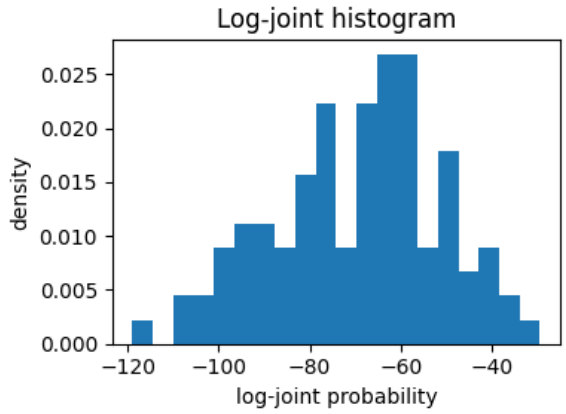


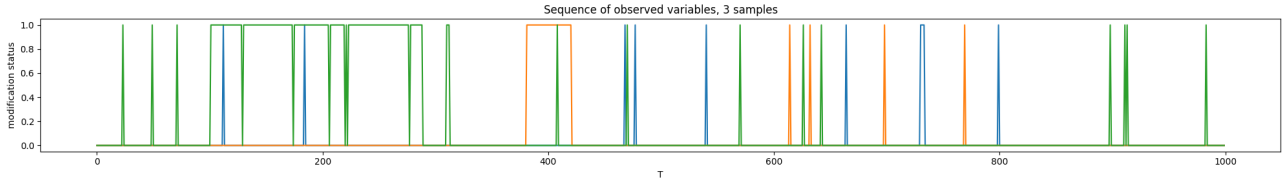


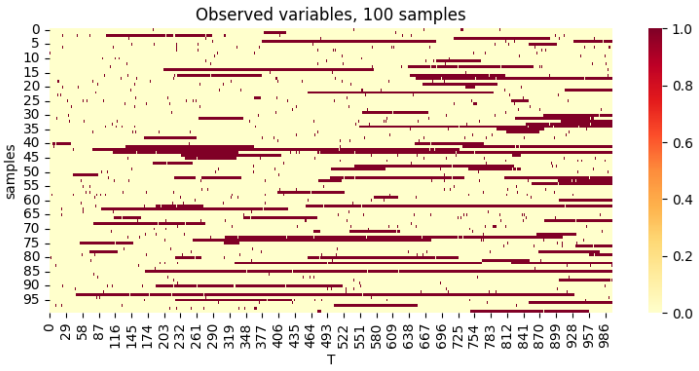


6. Do the same (3-5) for HMM2 with $T = 1000$. What are the differences? How can they be explained?

```
Load HMM2. CPDs:
prior
['prior(0)=1.000' 'prior(1)=0.000']
transition_mat
[['tau(0->0)=0.999' 'tau(0->1)=0.001']
['tau(1->0)=0.005' 'tau(1->1)=0.995']]
emission_mat
[['e(0->0)=0.990' 'e(0->1)=0.010']
['e(1->0)=0.010' 'e(1->1)=0.990']]
```







The differences between the results of HMM1 and HMM2 are evident in the log joint probabilities and the behavior of hidden and observed sequences:

(a) **Log Joint Probabilities:**

- **HMM1:** Log joint probabilities are much smaller (more negative).
- **HMM2:** Log joint probabilities are higher (less negative).

(b) **Hidden and Observed Sequences:**

- **HMM1:** Frequent transitions between states, indicating high randomness.
- **HMM2:** Fewer transitions, indicating more stability and persistence in states.

Explanation

(a) **Transition and Emission Matrices:**

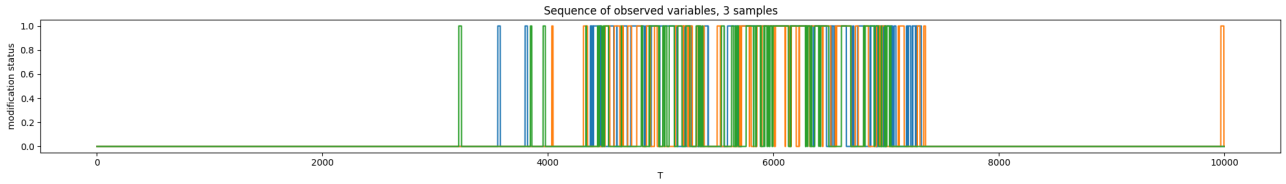
- **HMM1:** Matrices far from the identity matrix, implying high entropy (more randomness). This results in lower log joint probabilities due to the rarity of each sequence.
- **HMM2:** Matrices closer to the identity matrix, implying lower entropy (less randomness). This results in higher log joint probabilities because sequences are more predictable.

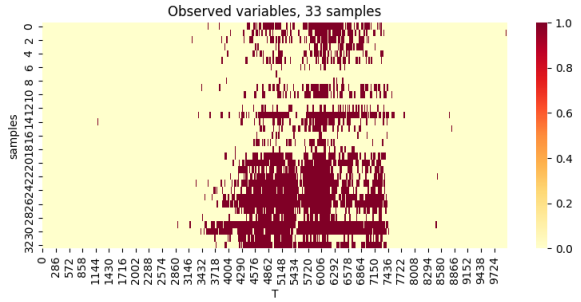
(b) **Entropy and Probability Distribution:**

- **High Entropy (HMM1):** Greater randomness means lower probability for specific sequences, resulting in very negative log joint probabilities.
- **Low Entropy (HMM2):** Less randomness increases the probability for specific sequences, resulting in higher log joint probabilities.

In summary, HMM1's higher entropy leads to more random and rare sequences with lower log joint probabilities, while HMM2's lower entropy leads to more stable and common sequences with higher log joint probabilities.

7. Load real observations from the `small_binary_data.csv` file [*]. Plot these observations same as the sampled data (3) [*]. What is the pattern in the real data? Explain the differences. Does this data fit the HMM model assumptions?





The real data from `small_binary_data.csv` shows a distinct pattern: many 1's cluster in the middle of the sequence, while 0's appear predominantly at the beginning and end. This indicates the sequence does not follow the typical Markov properties assumed in HMMs.

Differences and Explanation

(a) **Pattern Differences:**

- **Real Data:** Clear structure with 1's clustering in the middle and 0's at the edges.
- **HMM Sampled Data:** More randomness and frequent state transitions, as expected from a Markov process.

(b) **Explanation:**

- Real data exhibits non-Markovian behavior where state occurrences are not solely dependent on the previous state, indicating a structured or periodic phenomenon.
- This clustering pattern suggests that transitions between states are not memoryless, which is a fundamental assumption of HMMs.

The real data shows a clear deviation from HMM assumptions, evidenced by the structured clustering of states. This suggests the real-world process generating this data has a different underlying mechanism than what an HMM can capture.

3 Calculating Prior, Likelihood, and Posterior

As we discussed above, calculating the priors, likelihoods, and posteriors is difficult. However, there are efficient ways to calculate these probabilities using Dynamic-Programming. These are examples of inference problems, which we'll explore more in the next chapter of the course. For now, we'll focus on calculating the prior $p(X_t)$ for some t:

1. We define a subproblem for each $t = 1, \ldots, T$ and $k \in \text{Val}(X)$:

   $$P[t, k] = p(X_t = k)$$

   How can we calculate $P[1, k]$ directly from the network CPDs?

2. Using the multiplicative rule, we get a recursive formula for P:

   $$P[t, k] = p(X_t = k) = \sum_{l \in \text{Val}(X)} p(X_t = k | X_{t-1} = l) \cdot p(X_{t-1} = l) = \sum_{l} p(X_t = k | X_{t-1} = l) \cdot P[t-1, l]$$

   Using this formula, we can iteratively fill a table $P_{|\text{Val}(X)| \times T}$ by first filling the first column with the start conditions $P[1, k]$, and continuing by columns: At each stage t, we use the t-th column to compute the $(t+1)$-th. Implement in the HMM class the method `log_p_Xt`, that calculates the table $\log P[t, k] = \log p(X_t = k)$.

3. Implement in the HMM class the method `log_p_Xt_given_Ot`, that calculates $\log p(X_t | o_t)$. Hint: How can you use $\log p(X_t = k)$ to calculate the point-wise log posterior $\log p(X_t | o_t)$?

4. Next, we supply you with a function `_log_forward` that calculates the log of the table $\log F[t, k] = \log p(X_t = k, o_{1:t})$ in a similar way to the log prior (we'll talk about this algorithm later in the course) [*]. Implement in the HMM class a method `log_likelihood`, that calculates the log-likelihood of the observations $\log p(o_{1:m})$.

4 Identifying Corrupt Data

You are given a validation and a test dataset, each including a set of observations. The validation dataset was sampled directly from the HMM2 model. In the test dataset, the data is noisy, and some modification statuses were flipped (corrupted). We want to differentiate between samples with noisy observations and the rest. Intuitively, our trained Bayesian network should assign lower probability to corrupted observations and higher probability to the real ones. We can use this intuition to filter out corrupted data:

1. Calculate the log-likelihood of each observation in the validation dataset (which includes only real observations) under the HMM2 model:

   $$\log p(o_{1:m})$$

2. Compute the average μ and the standard deviation σ of the log-likelihood on the validation dataset (average and std of $\log p(o_{1:m})$ over the validation set samples).

3. Define a prediction rule: A sample with observations that have log-likelihood $\log p(o_{1:m})$ below three standard deviations of the average marginal log-likelihood, are classified as corrupted. That is,

   $$\text{if } \log p(o_{1:m}) < \mu - 3 \cdot \sigma \Rightarrow o_{1:m} \text{ is corrupted}$$

   This rule states that if a sample's observations are very unlikely to originate from the same distribution as the validation data, then it is corrupted. In our case, "very unlikely" is translated to 3 standard deviations away from the mean. This practice follows a rule called The Empirical Rule.

4. For each sample in the test dataset, calculate its log-likelihood and classify it as corrupted or real using the prediction rule.

5. Plot the histogram of the log likelihood $\log p(o_{1:m})$ for the validation data [*]. On top of this histogram, plot two more histograms: One of the marginal log probability for the observations classified as "real", and another for the observations classified as "corrupted" [*]. Explain the results.

5 Predict Active Promoters

Say we want to predict the most likely segmentation to active promoters given a sequence of modifications. We'll try to use the following rule:

$$\forall t, \hat{X}_t = \arg\max_x p(X_t|O_t)$$