Discrete HMM for DNA Sequence Analysis and Gene Region Prediction

In [12]:
```
import numpy as np
from hmmlearn import hmm
from collections import Counter
from hmmlearn.hmm import CategoricalHMM
```

In [4]:
```
# Mapping nucleotides and gene states to integers
nucleotide_mapping = {'A': 0, 'C': 1, 'G': 2, 'T': 3}
state_mapping = {'N': 0, 'G': 1}
```

In [5]:
```
# Inverse mapping for decoding
inv_state_mapping = {0: 'N', 1: 'G'}
```

In [6]:
```
# Synthetic training data (sequences and labeled gene/non-gene states)
train_sequences = ["ATGCG", "TTCGA", "GGGTT"]
train_states = ["GGGGG", "NNNGG", "GGGNN"]
```

In [7]:
```
# Convert sequences and states to numerical format
observed_sequences = [np.array([nucleotide_mapping[nuc] for nuc in seq]).reshape(-1, 1)
                      for seq in train_sequences]
state_sequences = [np.array([state_mapping[state] for state in states])
                   for states in train_states]
```

In [8]:
```
# Concatenate all sequences for training
X = np.concatenate(observed_sequences)
lengths = [len(seq) for seq in observed_sequences]
```

In [9]:
```
# Initialize a Multinomial HMM with 2 hidden states: Gene (G) and Non-Gene (N)
n_states = 2  # Gene, Non-Gene
n_observations = len(nucleotide_mapping)
```

In [13]:
```
model = hmm.CategoricalHMM(n_components=n_states, n_iter=100, tol=1e-4, verbose=True)
model.n_features = n_observations
```

In [14]:
```
# Train the model using Maximum Likelihood Estimation (MLE)
model.fit(X, lengths)
```

```
        64     -18.08058659      +0.00017862
        65     -18.08044884      +0.00013774
        66     -18.08034266      +0.00010618
        67     -18.08026083      +0.00008183
```

Out[14]:

CategoricalHMM

```
CategoricalHMM(n_components=2, n_features=4, n_iter=100,
               random_state=RandomState(MT19937) at 0x28FCA358E40, tol=0.0001,
               verbose=True)
```

In [15]:
```
print("Initial Probabilities (π):")
print(model.startprob_)
print("\nTransition Probabilities (A):")
print(model.transmat_)
print("\nEmission Probabilities (B):")
print(model.emissionprob_)
```

```
Initial Probabilities (π):
[0.35263532 0.64736468]

Transition Probabilities (A):
[[0.78324568 0.21675432]
 [0.40475911 0.59524089]]

Emission Probabilities (B):
[[6.82116014e-02 2.32920347e-01 6.98761034e-01 1.07017613e-04]
 [2.20522388e-01 1.31530440e-12 9.31494853e-09 7.79477603e-01]]
```

In [16]:
```
# Example test DNA sequence
test_sequence = "ATGGC"
test_observed = np.array([nucleotide_mapping[nuc] for nuc in test_sequence]).reshape(-1, 1)
```

In [17]:
```
# Predict hidden states using Viterbi algorithm
predicted_states = model.predict(test_observed)
predicted_labels = [inv_state_mapping[state] for state in predicted_states]
```

In [18]:
```
# Display results
print("\nTest DNA sequence:       ", test_sequence)
print("Predicted gene regions:  ", "".join(predicted_labels))
```

```
Test DNA sequence:        ATGGC
Predicted gene regions:   GGNNN
```

In [ ]: