HMM-based neural spike analysis

Michael C. Nechyba

BMI Project
Electrical and Computer Engineering
University of Florida

http://mil.ufl.edu/~nechyba
Goal:
Classify arm movement into two classes: (1) stationary and (2) moving

Talk outline:

• Classifier overview
• HMM/VQ discussion
• Generating training data
• Results
• Current/future work
$O^* = \{ O_{t-N+1}, O_{t-N+2}, O_{t-1}, O_t \}$

Classifier block diagram

signal-to-symbol conversion

HMM (stationary)
$P(O^* | \lambda_s)$

HMM (moving)
$P(O^* | \lambda_m)$
Observable Markov models

- Can directly observe state (nothing hidden)
- \( N \) states with probabilistic transitions \( a_{ij} \)
- Three-state example:
Hidden Markov models (HMMs)

- Underlying state no longer directly observable.
- Each state has probability distribution of observables associated with it.

- Two types:
  - Discrete output
  - Continuous output
Discrete-output HMM example

Hidden Markov model:

Sample observation sequence:
Continuous-output HMM

Hidden Markov model:

Sample observation sequence: $X = \{x_t\}, \ t = \{1, \ldots, T\}$.
Why discrete-output HMMs?

- Much more computationally efficient
- Combine with VQ to model/classify real data
Hidden Markov model applications

1. Speech recognition
2. Language modeling
3. Gesture recognition (e.g. sign language)
4. Hand-writing recognition
5. Facial-expression recognition (e.g. sign language)
6. Human skill modeling (e.g. surgical procedures)
7. Human control strategy analysis (e.g. driving)
8. Robot control (e.g. autonomous driving)
9. And others...
Discrete-output HMM parameters

Definitions:

• $S_i = \text{state } i$ ; $q_t = \text{state of system at time } t$

• $v_k = \text{observable } k$ ; $O_t = \text{observable at time } t$

HMM parameters:

• $N \times N$ state-transition matrix $A$ :

$$a_{ij} \equiv P(q_t = S_j | q_{t-1} = S_i) , \quad i, j \in \{1, \ldots, N\}$$

• $N \times L$ output probability distribution matrix $B$ :

$$b_j(k) \equiv P(O_t = v_k | q_t = S_j) , \quad j \in \{1, \ldots, N\} , \quad k \in \{1, \ldots, k\}$$
Discrete-output HMM example

Hidden Markov model:

![Diagram of a hidden Markov model with three states and transition probabilities between them.]

Output probability matrix:

\[ B = \begin{bmatrix}
\frac{1}{4} & \frac{3}{5} & \frac{1}{5} & \text{(red)} \\
\frac{1}{4} & \frac{1}{5} & \frac{3}{5} & \text{(green)} \\
\frac{1}{2} & \frac{1}{5} & \frac{1}{5} & \text{(yellow)}
\end{bmatrix} \]
Hidden Markov models: 3 basic problems

Definitions:

• \( O = \{O_t\}, \ t = \{1, \ldots, T\} \)

• \( \lambda = \{A, B, \pi\} = \text{hidden Markov model} \)

1. Evaluation: Compute \( P(O|\lambda) \).

2. Decoding: Compute most likely state sequence \( Q^* \):

\[
Q^* = \{q_t\}, \ t = \{1, \ldots, T\}.
\]

3. Training: Compute maximum-likelihood estimate \( \lambda^* \):

\[
P(O|\lambda) \leq P(O|\lambda^*), \ \forall \lambda
\]
Classifier block diagram

\[ O^* = \{ O_{t-N+1}, O_{t-N+2}, O_{t-1}, O_t \} \]

signal-to-symbol conversion

HMM (stationary)

\[ \lambda_s \]

\[ P(O^* | \lambda_s) \]

HMM (moving)

\[ \lambda_m \]

\[ P(O^* | \lambda_m) \]
Vector quantization algorithm

1. **Initialization:** Choose some initial setting for the $L$ centroids $\{\mu_i\}$ in the VQ codebook.

2. **Classification:** Classify each $x_j$ into class $\omega_i$ such that,

$$\text{dist}(x_j, \mu_i) \leq \text{dist}(x_j, \mu_l), \forall l.$$  

3. **Codebook update:** Update the centroid for each class $\omega_i$,

$$\mu_i = \frac{1}{n_i} \left( \sum_{x_j \in \omega_i} x_j \right).$$

4. **Termination:** Stop when the distortion $D$ has decreased below some threshold level, or when the algorithm has converged to a constant level of distortion.
LBG VQ algorithm

1. Initialization: one centroid of all data
2. Splitting: split each centroid into two \((\mu_i + \varepsilon, \mu_i - \varepsilon)\).
3. Vector quantization on current number of centroids.
4. Iterate steps 2 and 3 until satisfied...
Examples (synthetic data)

Uniformly distributed data

Example #2

Example #1

Let’s see some action...(qt movies)
Classifier block diagram

\[ O^* = \{ O_{t-N+1}, O_{t-N+2}, O_{t-1}, O_t \} \]

signal-to-symbol conversion

HMM (stationary)

\[ P(O^* | \lambda_s) \]

\[ \lambda_s \]

HMM (moving)

\[ P(O^* | \lambda_m) \]

\[ \lambda_m \]
Generating training data

Velocity of arm (unfiltered)

Time index (100 msec)

Hand segment data into two classes:
• Stationary
• Moving
Resulting training data

104 neural channels

- time
- less neural activity
- more neural activity

- movement
- no movement
Classification results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Stationary Arm (% correct)</th>
<th>Moving Arm (% correct)</th>
</tr>
</thead>
<tbody>
<tr>
<td>single-state HMM</td>
<td>87.7%</td>
<td>84.4%</td>
</tr>
<tr>
<td>four-state HMM</td>
<td>92.9%</td>
<td>90.5%</td>
</tr>
<tr>
<td>six-state HMM</td>
<td>95.1%</td>
<td>90.9%</td>
</tr>
<tr>
<td>six-state HMM (γ = 1.2)</td>
<td>93.6%</td>
<td>96.6%</td>
</tr>
<tr>
<td>six-state HMM (γ = 1.6)</td>
<td>90.7%</td>
<td>98.4%</td>
</tr>
</tbody>
</table>

Classifier details:
- HMMs — left-to-right (Bakis) models, variable number of states
- VQ — 256 prototype vectors (clusters)
- Decision rule:
  \[
  \frac{P(O^* | \lambda_s)}{P(O^* | \lambda_m)} = \gamma \quad (\gamma \text{ nominally equal to } 1)
  \]
Biasing the classifier

Classification performance as a function of bias

% correct vs $\gamma$

- **stationary**
- **moving**
Conclusion & future work

Caveats about results:
• Limited results
• “Leaving-$k$-out” cross validation

Future work:
• Signal-to-symbol conversion — reduce loss of information (distortion)
• More extensive statistical testing
• From dichotomy to polychotomy (motion primitives)
• Generalization beyond single data set (when available)
• Suggestions?