Activity Recognition
by Hidden Markov Models
with KDE Emission Probabilities

Massimo Piccardi, Faculty of IT, UTS
massimo@it.uts.edu.au
Agenda

• A 5-minute introduction to HMM
• What we have done
  – Maximum-likelihood HMM estimation with KDE emission probabilities
  – Comparison with conventional HMMs
  – Experiments on state decoding from synthetic data and CAVIAR surveillance datasets
Activity recognition

- Recognising activities: major task in video surveillance

- An activity:
  \[ o(t), t_0 \leq t \leq t_f \]

- Or, given that we sample frames:
  \[ O_1, \ldots, O_t, \ldots, O_T \]
Activity recognition: difficulties

{o: height of hand from table}

Hand Up

$o_1(t)$

$o_2(t)$

$\text{DTW...}$
Activity modelling by hidden Markov models (HMMs)

\[ o_1 \ldots o_T \rightarrow HMM \]
HMM

- $x$: discrete values
- $o$: continuous values

The diagram illustrates the transition probabilities $a_{ij}$ and the emission probabilities $b_j(o_t)$ in a Hidden Markov Model (HMM).
HMM

- $X = x_1 \ldots x_T$ states (hidden)
- $O = o_1 \ldots o_T$ emissions (aka observations)
- $A = \{a_{ij}\}$ state transition probabilities
- $B = \{b_j(o)\}$ emission probabilities
- $\pi = \{\pi_i\}$ initial state probabilities
- $\lambda = \{A, B, \pi\}$ is the HMM

- $p(O \mid \lambda)$ likelihood of sequence in model
Activity recognition by HMMs

$$o_{11} \ldots o_{1T} \rightarrow HMM_1$$

$$o_{21} \ldots o_{2T'} \rightarrow HMM_2$$
Activity recognition by HMMs

\[ o_{11} \cdots o_{1T} \rightarrow HMM_1 \]

\[ o_{21} \cdots o_{2T'} \rightarrow p(O_2 \mid \lambda_1) \]
Learning $\lambda$

- Baum-Welch algorithm: an iterative algorithm in Expectation-Maximization (EM) style finding $\lambda$ that gives a local maximum of the likelihood

- Hybrid models
  - recent modifications to the conventional HMM
  - assume knowledge of state values for supervised training of $B$, the emission probabilities: SVM, ANN etc

- We do not assume knowledge of state values
Conventional $B$: Gaussian Mixtures

- In conventional HMM, each $b_j(o)$ is modelled by a Gaussian Mixture (GM) with a pre-determined number, $M$, of Gaussian components.
- $b_j(o)$ then becomes

\[ b_j(o) = \sum_{l=1}^{M} \alpha_{jl} G(o, \mu_{jl}, \Sigma_{jl}) \]
Learning a GM from $x_i$ data

- Iterate until convergence:

  $$\alpha_l^{new} = \frac{1}{N} \sum_{i=1}^{N} p(l \mid x_i, \Theta^c)$$

  $$\mu_l^{new} = \frac{\sum_{i=1}^{N} x_i p(l \mid x_i, \Theta^c)}{\sum_{i=1}^{N} p(l \mid x_i, \Theta^c)}$$

  $$\Sigma_l^{new} = \frac{\sum_{i=1}^{N} (x_i - \mu_l^{new})(x_i - \mu_l^{new})^T p(l \mid x_i, \Theta^c)}{\sum_{i=1}^{N} p(l \mid x_i, \Theta^c)}$$
Learning $b_j(o)$ from the $o_t$ observations

- Not that different: there's an extra term, $\gamma_j(t)$ that tells which is the probability of being in state $j$ at time $t$
- $\gamma_j(t)$ acts like a weight expressing the "membership" of an observation to a state

\[
\begin{align*}
\alpha_{jl}^{new} &= \frac{\sum_{t=1}^{T} p_j(l \mid o_t, \Theta) \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)} \\
\mu_{jl}^{new} &= \frac{\sum_{t=1}^{T} o_t \ p_j(l \mid o_t, \Theta) \gamma_j(t)}{\sum_{t=1}^{T} p_j(l \mid o_t, \Theta) \gamma_j(t)} \\
\sigma_{jl}^{2new} &= \frac{\sum_{t=1}^{T} (o_t - \mu_{jl}^{new})^2 \ p_j(l \mid o_t, \Theta) \gamma_j(t)}{\sum_{t=1}^{T} p_j(l \mid o_t, \Theta) \gamma_j(t)}
\end{align*}
\]

(NB: 1D case)
Our proposal:
Kernel Density Estimation for $B$
(KDE/HMM)

• Known limitations of GM:
  – modelling a pdf with more modes than the components
  – modelling a pdf with uniform distribution
KDE/HMM

- Kernel Density Estimation can overcome those two problems

- Further advantage: lower number of parameters (aka non-parametric technique)

- Disadvantage: greater computational complexity
Nota Bene

- GM

\[ b_j(o) = \sum_{l=1}^{M} \alpha_{jl} G(o, \mu_{jl}, \Sigma_{jl}) \]

- KDE (Gaussian kernel)

\[ b_j(o) = \frac{1}{T f(\Sigma_j)} \sum_{t=1}^{T} \alpha_{jt} G(o, o_t, \Sigma_j) \]

one component per observation!

only one \( \Sigma \) for all components

for normalisation

we’ll see why the weights

centred in the observation

- Similarity only apparent
Nota Bene

- GM

- KDE (Gaussian kernel)
KDE/HMM: problems to be solved

• How to maximise likelihood for a common $\Sigma$?
• How to maximise likelihood wrt $\Sigma$?
  – KDE has a notorious problem with ML: $\Sigma \rightarrow 0$; but we don’t want to have kernels with null variance
• How to modify Baum-Welch for the KDE case?
• Will KDE, as a non-parametric technique, succeed in finding stable memberships of observations to states?
From [1], taking the derivative of their Equation 7 wrt to a common $\Sigma$:

$$
\sigma_{jt}^{new} = \sum_{l=1}^{M} \left( \sum_{t=1}^{T} \left( o_{lt} - \mu_{jt}^{new} \right)^2 p_j(l \mid o_t, \Theta) \gamma_j(t) \right) \sum_{l=1}^{M} \left( \sum_{t=1}^{T} p_j(l \mid o_t, \Theta) \gamma_j(t) \right)
$$

(NB: 1D case)

KDE from $x_i$ data with maximum pseudo-likelihood

• Leave $x_i$ out when evaluating $p_{KDE}(x_i)$:

$$p_{KDE}(x_i) = \frac{1}{Nf(\Sigma)} \sum_{k=1, k \neq i}^{N} G(o_i, o_k, \Sigma)$$

• Duin: zero crossings of derivative of PL in $\Sigma$

• We propose to modify the EM algorithm so that our result can be directly plugged-in in Baum-Welch

• In experiments performed, learning always converged, and to a maximum of PL
Learning $b_j(o)$ from the $o_t$ observations with MPL

- In Baum-Welch, leave $o_t$ out when evaluating $b_j(o_t)$:

$$b_j(o_t) = \frac{1}{T f(\Sigma)} \sum_{k=1, k \neq t}^{T} \alpha_{jk} G(o_t, o_{jk}, \Sigma_j)$$

- Therefore (common $\Sigma$, MPL):

$$\sigma_{j}^{2}_{\text{new}} = \frac{\sum_{t=1}^{T} \sum_{l=1, o_l \neq o_t}^{T} (o_t - o_l)^2 p_j(l | o_t, \Theta) \gamma_j(t)}{\sum_{t=1}^{T} \sum_{l=1, o_l \neq o_t}^{T} p_j(l | o_t, \Theta) \gamma_j(t)}$$
Modified Baum-Welch for KDE/HMM

\[ \alpha_{jl}^{\text{new}} = \gamma_j(l) \quad \text{or} \quad \alpha_{il}^{\text{new}} = \frac{\sum_{t=1}^{T} p_i(l \mid o_t, \Theta) \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \]

\[ \mu_{jl}^{\text{new}} = o_l \]

\[ \sigma_{jl}^{2\text{new}} = \frac{\sum_{t=1}^{T} \sum_{l=1, o_l \neq o_t}^{T} (o_t - o_l)^2 p_j(l \mid o_t, \Theta) \gamma_j(t)}{\sum_{t=1}^{T} \sum_{l=1, o_l \neq o_t}^{T} p_j(l \mid o_t, \Theta) \gamma_j(t)} \]
Experimental results

- Experiments on state decoding
- Synthetic and real data
- 3 state values (‘classes’) in ground truth
- We create a HMM with 3 state values, hoping that they will come into correspondence with the ground-truth states
- 2 dimensions
- GM with 2 components
With synthetic data

- States:
  - Class 1: A uniform distribution between \( x, y = 0 \) and 18
  - Class 2: Four Gaussian distributions between \( x, y = 21 \) and 26 and \( \sigma = 2 \)
  - Class 3: Another uniform distribution between \( x, y = 30 \) and 60
- Independent training and test sets, 2-fold cross val, 50 iter
- Results [2]:

<table>
<thead>
<tr>
<th>Initial ( \sigma )</th>
<th>( e_{KDE} ) (%)</th>
<th>( e_{GM} ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>67.4</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>46.3</td>
</tr>
<tr>
<td>2.5</td>
<td>0.0</td>
<td>46.3</td>
</tr>
<tr>
<td>10</td>
<td>0.0</td>
<td>46.3</td>
</tr>
</tbody>
</table>

With real data: CAVIAR

- Data sets acquired as part of EU funded CAVIAR project/IST 2001 37540
  http://homepages.inf.ed.ac.uk/rbf/CAVIAR/
- Video clips of activities such as walking, browsing, resting, slumping, fainting, leaving bags, meeting, fighting, with ground truth for each frame (in XML)
With real data: CAVIAR

- 3 state values: Inactive, Walking, Running
- 2 features: speed magnitudes at 5 and 25 frame interval
- Great amount of overlapping in the feature space
- Training: 7 sequences (total 1975 frames); testing: 6 sequences (total 1605 frames)
- Various combination of initial parameters
- Results from the 2 data sets [2]:
  - KDE often (not always) lowest error
  - $e_{KDE} = 14.2\% - 16.4\%$, $e_{GM} = 15.1\% - 23.6\%$
  - KDE substantially insensitive to the choice of its only parameter, $\Sigma$
Running, KDE
Conclusions

• We have proposed a HMM with KDE emission probabilities trained with Maximum (pseudo)Likelihood
• No need to know the states at any stage of training – general application
• Much higher computational cost: $O(T)$ vs $O(M)$ – it may or may not scare you
• Much fewer parameters than GM models – less chances to get trapped in a useless local optimum
• In an experiment on state decoding from real videos, low error rate and very weak (almost no) dependence on the initial parameters