Decoding Human Mental States
Representation, learning and inference of Dynamic Bayesian Networks

Rana el Kaliouby
Research Scientist
MIT | Media Lab
Kaliouby@media.mit.edu
http://web.media.mit.edu/~kaliouby

Pattern Recognition
September 2008
What is this lecture about?

- Probabilistic graphical models as a powerful tool for decoding human mental states
- Dynamic Bayesian networks:
  - Representation
  - Learning
  - Inference
- Matlab’s Bayes Net Toolbox (BNT) – Kevin Murphy
- Applications and class projects
Decoding human mental states

- Mindreading
  - Our faculty for attributing mental states to others
  - Nonverbal cues / behaviors / sensors
  - We do that all the time, subconsciously
  - Vital for communication, making sense of people, predicting their behavior
People States

- Emotions (affect)
- Cognitive states
- Attention
- Intentions
- Beliefs
- Desires

Actress Florence Lawrence who was known as “The Biograph Girl”. From A Pictorial History of the Silent Screen.
Channels of People States

Observable:

- Head gestures
- Facial expressions
- Emotional Body language
- Posture / Gestures
- Voice
- Text
- Behavior: pick and manipulate objects

Up-close sensing:

- Temperature
- Respiration
- Pupil dilation
- Skin conductance, ECG, Blood pressure
- Brain imaging
Reading the mind in the face
Autism Research Centre, UK (Baron-Cohen et al., 2003)

Afraid
Angry
Bored
Bothered
Disbelieving
Disgusted
Excited
Fond
Happy
Hurt
Interested
Kind
 Liked
Romantic
Sad
Sneaky
Sorry
Sure
Surprised
Thinking
Touched
Unfriendly
Unsure
Wanting
Reading the mind in Gestures

- Choosing
- Thinking
- Interested evaluation
- Evaluation, skepticism
- Head on palms (boredom)
- Nose touch (deception)
- Mouth cover (deception)

Images from Pease and Pease (2004), The Definitive Book of Body Language

Rana el Kaliouby
kaliouby@media.mit.edu
Reading the mind using EEG

- Feasibility and pragmatics of classifying working memory load with an electroencephalograph Grimes, Tan, Hudson, Shenoy, Rao. CHI08
- Dynamic Bayesian Networks for Brain-Computer Interfaces. Shenoy & Rao. Nips04
- Human-aided Computing: Utilizing Implicit Human Processing to Classify Images Shenoy, Tan. CHI08.
- OPPORTUNITY – AFFECTIVE STATES

Face: Average Response

Nonface: Average Response
Multi-modal

- Combining Brain-computer Interfaces With Vision for Object Categorization Ashish Kapoor, Pradeep Shenoy, Desney Tan. CVPR 2008

Rana el Kaliouby
kaliouby@media.mit.edu
Mindreader

- Facial feature extraction
- Head pose estimation
- Feature point tracking (Nevenvision)
- Head & facial action unit recognition
- Head & facial display recognition
- Mental state inference

Video input (observed)

Hmm ... Let me think about this
Face + Physiology

Rana el Kaliouby
kaliouby@media.mit.edu
Class Project – Pepsi data

Anticipation → Disappointment - Satisfaction

Liking / Disliking

25 consumers, 30 trials, 30 min. videos!
Multi-level Dynamic Bayesian Network
Probabilistic graphical models

Probabilistic models

Graphical models

Directed (Bayesian Belief Nets)

Undirected (Markov Nets)
Representation of Bayes Net

• A graphical representation for the joint distribution of a set of variables

• Structure
  – A set of random variables makes up the nodes in the network. (random variables can be discrete or continuous)
  – A set of directed links or arrows connects pairs of nodes (specifies directionality / causality).

• Parameters
  – Conditional probability table / density
  – quantifies the effects of parents on child nodes
Setting up the DBN

• The graph structure
  – Expert knowledge, make assumptions about the world / problem at hand
  – Learn the structure from data

• The parameters
  – Expert knowledge, intuition
  – Learn the parameters from data
Sprinkler - Structure
Conditional Probability Tables

- Each row contains the conditional probability of each node value for each possible combination of values of its parent nodes.
- Each row must sum to 1.
- A node with no parents has one row (the prior probabilities)
Sprinkler - Parameters

<table>
<thead>
<tr>
<th>C</th>
<th>P(S=F)</th>
<th>P(S=T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>T</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>P(C=F)</th>
<th>P(C=T)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>P(R=F)</th>
<th>P(R=T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>T</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>P(W=F)</th>
<th>P(W=T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>0.01</td>
<td>0.99</td>
</tr>
</tbody>
</table>
Why Bayesian Networks?

- Graph structure supports
  - Modular representation of knowledge
  - Local, distributed algorithms for inference and learning
  - Intuitive (possibly causal) interpretation
Why Bayesian Networks?

- Factored representation may have exponentially fewer parameters than full joint $P(X_1, \ldots, X_n) = \Rightarrow$
  - lower time complexity (less time for inference)
  - lower sample complexity (less data for learning)

$$P(W, S, R, C, \ldots) = P(W \mid R, S)P(R \mid C)P(S \mid C)P(C)$$

Graphical model asserts:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}[X_i])$$
Why Bayesian Networks?

People Patterns
• Uncertainty
• Multiple modalities
• Temporal
• Top-down, bottom-up

Bayesian Networks
• Probabilistic
• Sensor fusion
• Dynamic models
• Hierarchical models
• Top-down, bottom-up
• Graphical -> intuitive representation, efficient inference
Bayes Net ToolBox (BNT)

- Matlab toolbox by Kevin Murphy
- Ported by Intel (Intel’s open PNL)
- Problem set 4

- Representation
  - bnet, DBN, factor graph, influence (decision) diagram
  - CPDs – Gaussian, tabular, softmax, etc

- Learning engines
  - Parameters: EM, (conjugate gradient)
  - Structure: MCMC over graphs, K2

- Inference engines
  - Exact: junction tree, variable elimination
  - Approximate: (loopy) belief propagation, sampling
Represent the mental state agreeing, given two features: head nod and smile. (all are discrete and binary)

- First define the structure
- $N = 3$; % the total number of nodes
- $\text{intra} = \text{zeros}(N)$;
- $\text{intra}(1,2) = 1$;
- $\text{intra}(1,3) = 1$;

- specify the type of node: discrete, binary
- $\text{node_sizes} = [2 \ 2 \ 2]$;
- $\text{onodes} = 2:3$; % observed nodes
- $\text{dnodes} = 1:3$; % all the nodes per time slice
Case study: Mental States
Structure (One classifier or many?)

- Depends on whether the classes are mutually exclusive or not (if yes, we could let hidden node be discrete but say take 6 values)
Case study: Mental States
Structure - Dynamic

- But hang on, what about the temporal aspect of this? (my previous mental state affects my current one)

Rana el Kaliouby
kaliouby@media.mit.edu
Case study: Mental States
Structure - Dynamic

- More compact representation
Case study: Mental States
Structure - Dynamic

- Represent the mental state agreeing, given two features: head nod and smile and make it dynamic

- %intra same as before
- inter = zeros(N);
- inter(1,1) = 1;

- % parameter tying reduces the amount of data needed for learning.
- eclass1 = 1:3; % all the nodes per time slice
- eclass2 = [4 2:3];
- eclass = [eclass1 eclass2];

- % instantiate the DBN
- dynBnet = mk_dbn(intra, inter, node_sizes, 'discrete', dnodes, 'eclass1', eclass1, 'eclass2', eclass2, 'observed', onodes);

Rana el Kaliouby
kaliouby@media.mit.edu
Case study: Mental States
Parameters – (hand-coded)

• How many conditional probability tables do we need to specify?
Case study: Mental States
Parameters – (hand-coded)

% prior P(agreeing)
- dynBnet.CPD{1} = tabular_CPD(dynBnet, 1, [0.5 0.5]);

% P(2|1) head nod given agreeing
- dynBnet.CPD{2} = tabular_CPD(dynBnet, 2, [0.8 0.2 0.2 0.8]);

% P(3|1) smile given agreeing
- dynBnet.CPD{3} = tabular_CPD(dynBnet, 3, [0.5 0.9 0.5 0.1]);

% P(4|1) transition prob
- dynBnet.CPD{4} = tabular_CPD(dynBnet, 4, [0.9 0.2 0.1 0.8]);

<table>
<thead>
<tr>
<th></th>
<th>2 = F</th>
<th>2 = T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = F</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>1 = T</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

High prob of nod if the person is agreeing, v. low prob that we see a nod
if the person is not agreeing
Case study: Mental States

Parameters – (hand-coded)

% prior P(agreeing)
• dynBnet.CPD{1} = tabular_CPD(dynBnet, 1, [0.5 0.5]);

% P(2|1) head nod given agreeing
• dynBnet.CPD{2} = tabular_CPD(dynBnet, 2, [0.8 0.2 0.2 0.8]);

% P(3|1) smile given agreeing
• dynBnet.CPD{3} = tabular_CPD(dynBnet, 3, [0.5 0.9 0.5 0.1]);

% P(4|1) transition prob
• dynBnet.CPD{4} = tabular_CPD(dynBnet, 4, [0.9 0.2 0.1 0.8]);

<table>
<thead>
<tr>
<th></th>
<th>2 = F</th>
<th>2 = T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = F</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>1 = T</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>3 = F</th>
<th>3 = T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = F</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>1 = T</td>
<td>0.9</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Low prob of smile if the person is agreeing, equal prob of smile or not if the person is not agreeing
Case study: Mental States

Parameters – (hand-coded)

% prior P(agreeing)
- \( \text{dynBnet.CPD}\{1\} = \text{tabular\_CPD(dynBnet, 1, [0.5 0.5]);} \)

% \( P(2|1) \) head nod given agreeing
- \( \text{dynBnet.CPD}\{2\} = \text{tabular\_CPD(dynBnet, 2, [0.8 0.2 0.2 0.8]);} \)

% \( P(3|1) \) smile given agreeing
- \( \text{dynBnet.CPD}\{3\} = \text{tabular\_CPD(dynBnet, 3, [0.5 0.9 0.5 0.1]);} \)

% \( P(4|1) \) transition prob
- \( \text{dynBnet.CPD}\{4\} = \text{tabular\_CPD(dynBnet, 4, [0.9 0.2 0.1 0.8]);} \)

<table>
<thead>
<tr>
<th></th>
<th>( 2 = F )</th>
<th>( 2 = T )</th>
<th>( 3 = F )</th>
<th>( 3 = T )</th>
<th>( 4 = F )</th>
<th>( 4 = T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1 = F )</td>
<td>0.8</td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>( 1 = T )</td>
<td>0.2</td>
<td>0.8</td>
<td>0.9</td>
<td>0.1</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

High prob of agreeing now if I was just agreeing, low prob of agreeing now if I wasn’t agreeing
Case study: Mental States
Sampling the DBN

• T = 2;
• ncases = 1000;

• for i=1:ncases
  – ev = sample_dbn(dynBnet, 'length', T);
• end
Case study: Mental States
Parameters – learning

<table>
<thead>
<tr>
<th>Structure</th>
<th>Observability</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known</td>
<td>Full</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>Known</td>
<td>Partial</td>
<td>Expectation Maximization (EM)</td>
</tr>
<tr>
<td>Unknown</td>
<td>Full</td>
<td>Search through model space</td>
</tr>
<tr>
<td>Unknown</td>
<td>Partial</td>
<td>EM + search through model space</td>
</tr>
</tbody>
</table>

- Hierarchical BNs: you can learn the parameters of each level separately
- Learning the parameters:
  - If the data is full observable, then MLE (counting occurrences) (resulting model is applicable to exact inference)
- Learning the structure:
  - Search strategy to explore the possible structures;
  - Scoring metric to select a structure
Case study: Mental States
Parameters – MLE - discriminability

Rana el Kaliouby
kaliouby@media.mit.edu
Learning from data in BNT

- Define DBN structure as before
- Define DBN params as before (random CPTs)
- Also need to define inference/learning engine

- Load the example cases
- Learn the params (specifying the no. of iterations for algorithm to converge)

- \[ \text{dynBnet2, LL, engine2} = \text{learn\_params\_dbn\_em}(\text{engine2, cases, 'max\_iter', 20}); \]
Inference

• Updating your belief state
  – Time propagation
  – Update by measurement

  – Algorithm Bayes filter
    • Givens: \( \text{bel}(x_{t-1}), z_t \)
    • Step 1: \( \text{bel}(x_t) = \sum p(x_t|x_{t-1}) \text{bel}(x_{t-1}) \)
    • Step 2: \( \text{bel}(x_t|z_t) = c \ p(z_t | x_t) \text{bel}(x_t) \)
Inference in DBNs

Inference is belief updating.

Filtering: recursively estimate the belief state
Prediction: predict future state
Smoothing: estimate state of the past given all the evidence up to the current time
Case study: Mental States
Inference in BNT

- Instantiate an inference engine
  
  ```matlab
  engine2 = smoother_engine(jtree_2TBN_inf_engine(dynBnet2));
  ```

- Enter evidence
  
  ```matlab
  engine2 = enter_evidence(engine2, evidence);
  ```

- Compute marginal nodes
  
  ```matlab
  m = marginal_nodes(engine2, 1, 2); % referring to 1st node (hidden class node) in 2nd time slice (t+1)
  ```

- Infer the class
  
  ```matlab
  inferredClass = argmax(m.T);
  ```

Rana el Kaliouby
kaliouby@media.mit.edu
Mental state inference

Sliding Window

Time scale of mental states: 55 frames

Time scale of displays: 30 frames

Time scale of actions: 5 frames
Real time Inference in BNT

**Objective:** Compute the belief state of a hidden mental state \( P(X_i[t] | Y[t-w:t], P(X[t-w:t-1])) \) over time, \( 1 \leq i \leq x \) and \( 1 \leq t \leq T \)

**Given:** \( x \) mental state DBNs as in Figure 7.1, with \( y \) observations nodes \( Y \); evidence length is \( w \) and sliding factor, or lag, is \( dw \)

**Initialization:** Instantiate inference engine such as the unrolled-junction-tree or the forward-backward algorithms

**Initial inference instance**

for all \( t \) in \( w \) time slices do

Get current observations \( Y[t] \);
Enter evidence so far \( Y[1:w] \);
Calculate \( P(X[w]|Y[1:w]) \)

**Inference**

\( t = w + dw \)

for all \( t \) in \( T \) time slices do

Get current observations \( Y[t] \)
Enter evidence so far: \( Y[t-w:t] \) and \( P(X[t-w:t-1]) \)
Calculate \( P(X[t]|Y[t-w:t], P(X[t-w:t-1])) \)
Advance window \( t = t + dw \)
Inference – Naïve Approach

• Unrolling the DBN for a desired number of timesteps and treat as a static BN
• Apply evidence at appropriate times and then run any static algorithm
• Simple, but DBN becomes huge, inference runs out of memory or takes too long.
Inference – Better Approach

- We don’t need the entire unrolled DBN.
- A DBN represents a process that is stationary & Markovian:
  - **Stationary:**
    - the node relationships within timeslice $t$ and the transition function from $t$ to $t+1$ do not depend on $t$.
    - So we need only the initial timeslice and sufficient consecutive timeslices to show the transition function.
  - **Markovian:**
    - the transition function depends only on the immediately preceding timeslice and not on any previous timeslices (e.g., no arrows go from $t$ to $t+2$).
    - Therefore the nodes in a timeslice separate the past from the future.
- Use a 2TDBN that represents the first two timeslices of the process, and we use this structure for inference.
Inference – Better Approach

• Dynamic inference boils down to doing static inference on the 2TDBN and then following some protocol for “advancing” forward one step.

• Interface Algorithm or 1.5 Slice Junction Tree Algorithm (Murphy, 2002) [also in your problem set]

• Exact Inference

• Intuition:
  1. Initialization
     a) Transform DBN into 2 junction trees
        a) Moralize
        b) Triangulate
        c) Build junction tree
     b) Initialize values on the junction trees
        a) Multiply CPTs onto clique potentials
  2. Advance (belief propagation)
     a) Insert evidence into the junction tree
     b) Propagate potentials
Prerequisite concepts

**Moralizing a graph**
Marrying parents of a child

**Clique**
A graph in which every vertex is connected to every other vertex in the graph.

Two cliques:
\[ C_1 = \{a, b, d\}, \quad C_2 = \{b, c, d\} \]

**Junction tree**
A tree of maximal cliques in an undirected graph
1.5 Slice Junction Tree

- Outgoing interface $I_t$
  - Set of nodes in timeslice $t$ with children in timeslice $t+1$
  - $\{A_1, B_1, C_1\}$ is the outgoing interface of timeslice 1

- It d-separates the past from the future (Murphy, 2002)
  - "past" = all nodes in timeslices before $t$ and all non-interface nodes in timeslice $t$
  - "future" = nodes in timeslice $t+1$ and later
  - Therefore the outgoing interface encapsulates all necessary information about previous timeslices to do filtering.
Algorithm Outline

• Initialization:
  - Create two junction trees $J_1$ and $J_t$:
    • $J_1$ is the junction tree for the initial timeslice, created from timeslice 1 of the 2TDBN
    • $J_t$ is the junction tree for each subsequent timeslice and is created from timeslice 2 of the 2TDBN and the outgoing interface of timeslice 1
  - Time is initialized to 0

• Queries:
  - Marginals of nodes at the current timeslice can be queried:
    - If current time = 0, queries are performed on “_1” nodes in $J_1$
    - If current time > 0, queries are performed on “_2” nodes in $J_t$

• Evidence:
  - Evidence can be applied to any node in the current timeslice:
    - If current time = 0, evidence is applied to “_1” nodes in $J_1$
    - If current time > 0, evidence is applied to “_2” nodes in $J_t$
Algorithm Outline

- Advance:
  - Increment the time counter
  - Use outgoing interface from active timeslice to do inference in next timeslice

- Since the outgoing interface d-separates the past from the future, this ensures that when we do inference in the next timeslice we are taking everything that has occurred “so far” into account.

Rana el Kaliouby
kaliouby@media.mit.edu
Initialization of $J_1$

[1] Remove all nodes in timeslice 2 from the 2TDBN. Identify nodes in outgoing interface of timeslice 1, call it $I_1$.


[4] Initialize clique potentials to 1’s, multiply nodes’ CPTs onto cliques.
Initialization of $J_t$

[1] Starting with the whole 2TDBN, identify nodes in outgoing interface of timeslice 1 and 2, call them $I_1$ and $I_2$.

[2] Convert to 2TDBN to 1.5DBN (remove non-interface nodes in timeslice 1).
Initialization of $J_t$

[3] Moralize:
(marry $C_1, C_2$ parents of $D_2$)
(marry $C_1, B_2$ parents of $D_2$)
(marry $B_2, C_2$ parents of $D_2$) ....

then
(marry $A_2, B_2$ parents of $C_2$)
(marry $B_1, C_1$ parents of $B_2$)
(marry $A_1, B_1$ parents of $C_1$) ....
Initialization of $J_t$

[4] Find cliques, form junction tree
Clique:
- $\{A_1B_1C_1B_2\}$
- $\{A_1C_1B_2A_2\}$
- $\{C_1A_2B_2C_2\}$
- $\{C_1B_2C_2D_2\}$

Find clique that contains $I_1$ (in-clique), and $I_2$ out-clique

[5] Initialize clique potentials to 1’s, multiply nodes’ CPTs onto cliques (only of nodes in timeslice 2 because evidence is applied and nodes are queried only in timeslice 2)

then triangulate
(marry $A_1B_2$ parents of $C_1$)
(marry $C_1A_2$ parents of $B_2$)
Initialization Summary

In-clique

out-clique

$A_1B_1C_1$

$B_1C_1$

$B_1C_1D_1$

$A_1C_1B_2$

$C_1A_2B_2$

$C_1B_2C_2$

$C_1B_2C_2D_2$
Advance (Belief propagation)

- **At time t:**
  - Get current junction tree (if time = 0, $J_1$, otherwise $J_t$)
  - Update beliefs in current junction tree
  - Get $a_t$

- **Increment time**

- **After time is incremented**
  - Get current junction tree (always $J_t$)
  - Multiply $a_t$ onto in-clique potential of new junction tree
Advance (Belief propagation)
Approximate Inference

• Why?
  – to avoid exponential complexity of exact inference in discrete loopy graphs
  – Because one cannot compute messages in closed form (even for trees) in the non-linear/non-Gaussian case

• Algorithms:
  – Deterministic approximations: loopy BP, mean field, structured variational, etc
  – Stochastic approximations: MCMC (Gibbs sampling), likelihood weighting, particle filtering, etc
Bayesian Network Classifiers

![Bar chart showing percentages for different responses.]
Project ideas

- Pepsi data (speak to Hyungil / Rana)
- Combining EEG data w/ Face data (trying to get an SDK from Emotiv)
Summary

• Decoding human mental states
• Dynamic Bayesian Networks
  – Representation
  – Learning
  – Inference
• Matlab’s BNT
• Email for project ideas / brainstorming