Dynamic Bayes Nets (DBNs)

- We want to track multiple variables over time, using multiple sources of evidence
- Idea: Repeat a fixed Bayes net structure at each time
- Variables from time $t$ can condition on those from $t-1$

- DBNs with evidence at leaves are (in principle) HMMs
Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Procedure: “unroll” the network for $T$ time steps, then eliminate variables until $P(X_T|e_{1:T})$ is computed

Online belief updates: Eliminate all variables from the previous time step; store factors for current time only
DBN Particle Filters

- A particle is a **complete** sample for a time step

- **Initialize**: Generate prior samples for the $t=1$ Bayes net
  - Example particle: $G_1^a = (3,3)$ $G_1^b = (5,3)$

- **Elapse time**: Sample a successor for each particle
  - Example successor: $G_2^a = (2,3)$ $G_2^b = (6,3)$

- **Observe**: Weight each entire sample by the likelihood of the evidence conditioned on the sample
  - Likelihood: $P(E_1^a | G_1^a) \times P(E_1^b | G_1^b)$

- **Resample**: Select prior samples (tuples of values) in proportion to their likelihood
SLAM

- SLAM = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

http://www.youtube.com/watch?v=7iIDdvCXIFM
Hello,

Do you want free printr cartridge? Why pay more when you can get them ABSOLUTELY FREE! Just

\[ x \rightarrow f(x) \rightarrow y \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># free</td>
<td>2</td>
</tr>
<tr>
<td>YOUR_NAME</td>
<td>0</td>
</tr>
<tr>
<td>MISSPELLED</td>
<td>2</td>
</tr>
<tr>
<td>FROM_FRIEND</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

SPAM or +

\[ j \rightarrow f(j) \rightarrow “2” \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXEL-7,12</td>
<td>1</td>
</tr>
<tr>
<td>PIXEL-7,13</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>NUM_LOOPS</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Mistake-Driven Classification

- Start with initial classifier parameters
- See where your current classifier makes mistakes
- Adjust parameters to fix mistakes
- Stop when held-out accuracy levels off
- Overfitting
- Cross-validation
Some (Simplified) Biology

- Very loose inspiration: human neurons
- Inhibitory / excitatory
Linear Classifiers

• Inputs are feature values
• Each feature has a weight
• Sum is the activation

\[
activation_{w}(x) = \sum_{i} w_{i} \cdot f_{i}(x) = w \cdot f(x)
\]

• If the activation is:
  – Positive, output +1
  – Negative, output -1
Modeling Choices

• Activation function
  – Step
  – Ramp
  – Sigmoid
  – Piecewise linear
  – Gaussians (RBFs)

• Layout
  – Fully connected
  – Layered
  – **Feedforward** (hidden nodes)
  – Cyclic
Used for

- **Classification**
- Clustering
- Pattern association (associative memory tasks)
- Function approximation
- Forecasting (timeseries)
- Control (e.g., RL)
- Optimization
- Search

- Fast to **use**
In Contrast to NEAT

• Using error signals, not evolution
• Often faster to train
• Harder to achieve as high performance
Classification: Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

\[
\begin{pmatrix}
\# \text{ free} & : 4 \\
\text{YOUR\_NAME} & : -1 \\
\text{MISSPELLED} & : 1 \\
\text{FROM\_FRIEND} & : -3 \\
\cdots
\end{pmatrix}
\]

\[
\begin{pmatrix}
\# \text{ free} & : 2 \\
\text{YOUR\_NAME} & : 0 \\
\text{MISSPELLED} & : 2 \\
\text{FROM\_FRIEND} & : 0 \\
\cdots
\end{pmatrix}
\]

\[
\begin{pmatrix}
\# \text{ free} & : 0 \\
\text{YOUR\_NAME} & : 1 \\
\text{MISSPELLED} & : 1 \\
\text{FROM\_FRIEND} & : 1 \\
\cdots
\end{pmatrix}
\]

\[
dotprod{w}{f} \text{ positive means the positive class}
\]
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to $Y=+1$
  - Other corresponds to $Y=-1$
  - (Step function)

$w$

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS</td>
<td>-3</td>
</tr>
<tr>
<td>free</td>
<td>4</td>
</tr>
<tr>
<td>money</td>
<td>2</td>
</tr>
</tbody>
</table>

$-1 = \text{HAM}$

$+1 = \text{SPAM}$

$f \cdot w = 0$
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights
    \[ y = \begin{cases} 
      +1 & \text{if } w \cdot f(x) \geq 0 \\
      -1 & \text{if } w \cdot f(x) < 0 
    \end{cases} \]
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

\[ w = w + y^* \cdot f \]