The problem

- Create a generative model and a learning algorithm that can learn high-dimensional complex sequence data
- There are many applications where the HMM and the LDS are not nearly powerful enough
Tractable sequence models

- HMM – has $N^2$ parameters and when generating data the hidden state only carries log $N$ bits of information
- LDS – assumptions too unrealistic, data can rarely be explained linearly, seldom gaussian
Less tractable sequence models

• More powerful models with distributed hidden states need approximate inference for learning:
  – Particle filtering
  – Assumed density filtering
  – MCMC
Our new model

• Built from the Restricted Boltzmann Machine (RBM)
• A casual sequence of RBMs
• Highly intractable, lost of approximations
The Restricted Boltzmann Machine

- Powerful model of binary data
- Has an easy exact inference algorithm
- Has an efficient learning algorithm
- Has a natural hierarchical multilayered extension
  - This is a big win over other models
The Restricted Boltzmann Machine

- An undirected model
- Given $V$, $H$ is factorial
Goals:

- Construct a sequence model using RBM's
- The model should inherit some good qualities of the RBM
The Temporal Restricted Boltzmann Machine

- Use an RBM to model each item in the sequence
- Connect the RBM's with directed connections
- The result: a hybrid directed/undirected model
The Temporal Restricted Boltzmann Machine

Dynamic biases from previous timesteps capture sequential structure
The Temporal Restricted Boltzmann Machine

- Each RBM has the same parameters
The Temporal Restricted Boltzmann Machine

- With the past fixed, the undirected observation model makes exact inference easy
Approximate filtering

- The distribution \( P(H_T|V_1,\ldots,V_T,H_1,\ldots,H_{T-1}) \) is factorial by definition.
- This suggests: compute a factorial approximation to the filtering distribution.
- Like a very simple form of Assumed Density Filtering.
Approximate filtering

- Variables are binary in \( \{0,1\}^N \)
- A factorial distribution is in \( [0,1]^N \)
- \( P(H_T|V_1,...,V_T,H_1,...,H_{T-1}) \) is factorial, thus in \( [0,1]^N \)
- Use approx (\( \mu_t \) is \( H_t \)'s distribution):
  \[ \mu_T = P(H_T|V_1,...,V_T,\mu_1,...,\mu_{T-1}) \]
  using mean-field equations
Learning

- Select a random training vector
- Sample from the approximate *filtering* distribution (no smoothing)
- Slightly increase the log likelihood of each RBM given the fully visible data (this is easy for the RBM's)
- Not variational because learning ignores change in approximate posterior
Multilayered Models

• We can introduce additional hidden layers to get a better representation

• The result: a slightly better generative model
Multilayered TRBM's

- Using the idea of a multilayered RBM it is possible to add hidden layers to the TRBM model one layer at a time
- Has natural approximate inference
- Improves generative model
Multilayered TRBM's

- Use another TRBM Q to learn the aggregated approximate posterior:
  \[ P_{\text{agg}}(H_{1:T}) = \sum V P_{\text{approx}}(H_{1:T}|V_{1:T})D(V_{1:T}) \]
  - D is the data distribution

- To generate, sample Q(H_{1:T}), then sample \( P(V_{1:T}|H_{1:T}) \), approximately

- Can recurse for Q to make very deep models

- Has some variational justification
Multilayered TRBM's

Learning the Q model

Sampling from the hierarchical model

\[ P_{\text{approx}}(H_{1:T}|V_{1:T}) \]

\[ Q(H_{1:T},R_{1:T}) \]

\[ P_{\text{approx}}(V_{1:T}|H_{1:T}) \]

\[ Q(H_{1:T},R_{1:T}) \]

The data distribution

The model's distribution
Finetuning

• The learned weights can be further finetuned by the Wake-Sleep algorithm (and doesn't hurt performance)
Visible-Hidden Connections
Visible-Visible Connections
Questions?
Denoising using the TRBM

• The TRBM can be used to denoise: Reconstruct the noisy data from the hidden variables

• The TRBM denoises well, even though it was not “meant” for this task
Denoising Results

- Data
- Noise
- 1 hid
- 2 hid