Generative modeling of the visual scene

$$p(S|I, C) \propto p(I|S, C)p(S|C)$$

posterior likelihood prior

image = f(shape, material, illumination, ..., unknowns)

What's the problem?
Difficulties with identifying the causal structure

- Space of $S, C$ is huge
- If we define $P(I|S,C)$, must be able to invert it or search it
- Need efficient algorithms
- Many unknowns: identity of objects, types of scene elements, illuminations
- Might never have encountered some structures
- Is it even the right approach?
- Can we solve a simple case?
Toy problems - Handwritten digits

$I$ - Data: pixels (black or white)

$S$ - Complex “random” patterns with simple underlying structure
Toy problems - Handwritten digits

$I$ - Data: pixels (black or white)

$S$ - Complex “random” patterns with simple underlying structure

Must learn the model from the data.
Motivation: learn hierarchical, context dependent representations

- many real-world patterns are hierarchical in structure
- interpretation of patterns depends on context
- essential for complex recognition tasks
Motivation: learn hierarchical, context dependent representations

- many real-world patterns are hierarchical in structure
- interpretation of patterns depends on context
- essential for complex recognition tasks
Motivation: learn hierarchical, context dependent representations

- many real-world patterns are hierarchical in structure
- interpretation of patterns depends on context
- essential for complex recognition tasks
Motivation: learn hierarchical, context dependent representations

- many real-world patterns are hierarchical in structure
- interpretation of patterns depends on context
- essential for complex recognition tasks

The toy problem:
is it a pattern or a collection of features?

- objects, scene properties and structure, ...
- surface properties, contours, ...
- features
- pixels
Motivation: learn hierarchical, context dependent representations

- many real-world patterns are hierarchical in structure
- interpretation of patterns depends on context
- essential for complex recognition tasks

The toy problem:
*Is it a pattern or a collection of features?*
Approach

• Derive learning algorithm from a hierarchical statistical model
• model has the form of a Bayesian belief network (generative model)
• higher order structure is encoded with a *hierarchical* prior
• units states are *context dependent*: feedback from higher levels influences states in lower levels
• *learning* adapts the weights to find the most probable model of the ensemble of input patterns

*A Bayesian Belief network*

Key function: \( p(S_i \mid \text{pa}(S_i)) \) - the probability of state \( S_i \) given its parents, i.e. it’s causes.
A simple causal model: Noisy-OR

- We assume each cause $C_j$ can produce effect $E_i$ with probability $f_{ij}$.
- The noisy-OR model assumes the parent causes of effect $E_i$ contribute independently.
- The probability that none of them caused effect $E_i$ is simply the product of the probabilities that each one did not cause $E_i$.
- The probability that any of them caused $E_i$ is just one minus the above, i.e.

$$P(E_i | \text{par}(E_i)) = P(E_i | C_1, \ldots, C_n)$$

$$= 1 - \prod_{i} (1 - P(E_i | C_j))$$

$$= 1 - \prod_{i} (1 - f_{ij})$$

$$P(C | D, O, F) = 1 - (1 - f_{CD})(1 - f_{CO})(1 - f_{CF})$$
A general one-layer causal network

- Could either model causes and effects
- Or equivalently stochastic binary features.
- Each input $x_i$ encodes the probability that the $i$th binary input feature is present.
- The set of features represented by $\phi_j$ is defined by weights $f_{ij}$ which encode the probability that feature $i$ is an instance of $\phi_j$. 
The data: a set of stochastic binary patterns

Each column is a distinct eight-dimensional binary feature.

There are five underlying causal feature patterns.

*What are they?*
The data: a set of stochastic binary patterns

Each column is a distinct eight-dimensional binary feature.

true hidden causes of the data

inferred causes of the data
Hierarchical Statistical Models

A Bayesian belief network:

The joint probability of binary states is

\[ P(S|W) = \prod_i P(S_i|\text{pa}(S_i), W) \]

The probability \( S_i \) depends only on its parents:

\[ P(S_i|\text{pa}(S_i), W) = \begin{cases} h(\sum_j S_j w_{ji}) & \text{if } S_i = 1 \\ 1 - h(\sum_j S_j w_{ji}) & \text{if } S_i = 0 \end{cases} \]

The function \( h \) specifies how causes are combined, \( h(u) = 1 - \exp(-u), u > 0 \).

Main points:
- hierarchical structure allows model to form high order representations
- upper states are priors for lower states
- weights encode higher order features
Learning Objective

Adapt $W$ to find the most probable explanation of the input patterns. The probability of the data is

$$P(D_{1:N}|W) = \prod_{n} P(D_n|W)$$

$P(D_n|W)$ is computed by marginalizing

$$P(D_n|W) = \sum_{k} P(D_n|S_k, W)P(S_k|W)$$

Computing this sum exactly is intractable, but we can still make accurate approximations.
Approximating $P(D_n|W)$

Good representations should have just one or a few possible explanations for most patterns.

- in this case, most $P(D_n|S_k, W)$ will be zero

- the following approximation will be very accurate once weights have adapted

$$P(D_n|W) \approx P(D_n|\hat{S}, W)P(\hat{S}|W)$$

- approximation becomes increasingly accurate as learning proceeds
Using EM to adapt the network parameters

The complexity of the model is controlled by placing a prior on the weights.

- assume the prior to be the product of gamma distributions
- objective function becomes

\[
\mathcal{L} = P(D_{1:N} | W) P(W | \alpha, \beta)
\]

A simple and efficient EM formula for adapting the weights can be derived using the transformations \( f_{ij} = 1 - \exp(-w_{ij}) \) and \( g_i = 1 - \exp(-u_i) \).

\[
f_{ij} = \frac{\alpha - 1 + 2f_{ij} + \sum_n S_i^{(n)} S_j^{(n)} f_{ij} / g_j^{(n)}}{\alpha + \beta + \sum_n S_i^{(n)}}
\]

- \( f_{ij} \) can be interpreted as the frequency of state \( S_j \) given cause \( S_i \)
- \( f_{ij} \) is a weighted average of the number of times \( S_j \) was active given \( S_i \)
- the ratio \( f_{ij} / g_j \) inversely weights each term by number causes for \( S_j \)
Inferring the best representation of the observed variables

• Given on the input $D$, there is no simple way to determine which states are the input’s most likely causes.
  - Computing the most probable network state is an inference process
  - We want to find the explanation of the data with highest probability
  - This can be done efficiently with Gibbs sampling
• Gibbs sampling is another example of an MCMC method
• Key idea:

  *The samples are guaranteed to converge to the true posterior probability distribution*
Gibbs Sampling

Gibbs sampling is a way to select an ensemble of states that are representative of the posterior distribution $P(S|D, W)$.

- Each state of the network is updated iteratively according to the probability of $S_i$ given the remaining states.

- this conditional probability can be computed using (Neal, 1992)

$$P(S_i = a|S_j : j \neq i, W) \propto P(S_i = a|\text{pa}(S_i), W) \prod_{j \in \text{ch}(S_i)} P(S_j|\text{pa}(S_j), S_i = a, W)$$

- limiting ensemble of states will be typical samples from $P(S|D, W)$

- also works if any subset of states are fixed and the rest are sampled
The Gibbs sampling equations (derivation omitted)

The probability of $S_i$ changing state given the remaining states is

$$P(S_i = 1 - S_i | S_j : j \neq i, \mathbf{W}) = \frac{1}{1 + \exp(-\Delta x_i)}$$

$\Delta x_i$ indicates how much changing the state $S_i$ changes the probability of the whole network state

$$\Delta x_i = \log h(u_i; 1 - S_i) - \log h(u_i; S_i)$$
$$+ \sum_{j \in \text{ch}(S_i)} \log h(u_j + \delta_{ij}; S_j) - \log h(u_j; S_j)$$

• $u_i$ is the causal input to $S_i$, $u_i = \sum_k S_k w_{ki}$

• $\delta_{ij}$ specifies the change in $u_j$ for a change in $S_i$, $\delta_{ij} = +S_j w_{ij}$ if $S_i = 0$, or $-S_j w_{ij}$ if $S_i = 1$
Interpretation of the Gibbs sampling equation

- The Gibbs equation can be interpreted as: $\text{feedback} + \sum \text{feedforward}$

- $\text{feed-back}$: how consistent is $S_i$ with current causes?

- $\sum \text{feedforward}$: how likely is $S_i$ a cause of its children

- Feedback allows the lower-level units to use information only computable at higher levels

- Feedback determines (disambiguates) the state when the feedforward input is ambiguous
The higher-order lines problem

Can we infer the structure of the network given only the patterns?
Weights in a 25-10-5 belief network after learning

The first layer of weights learn that patterns are combinations of lines.

The second layer learns combinations of the first layer features.

The first layer of weights learn that patterns are combinations of lines.
The Shifter Problem

Shift patterns

weights of a 32-20-2 network after learning
Gibbs sampling: feedback disambiguates lower-level states

One the structure learned, the Gibbs updating convergences in two sweeps.
Another toy problem – Handwritten digits

Weights learned in a 64-32-32-16 network
Logistic belief nets

- generative model:

\[ p(v_i = 1) = \sigma(b_i + \sum_j h_j w_{ij}) \]

- \( \sigma(x) \) is the logistic function:

\[ \sigma(x) = \frac{1}{1 + \exp(-x)} \]

- with deep networks, it is possible to learn complex joint probability distributions
Wake-sleep learning

- For probabilistic models
  - top-down weights generate patterns from model distribution
  - bottom-up weights convey distribution of data-vectors
  - ideally the two distributions should match
- The “wake-sleep” algorithm adjusts the weights so that the distribution from recognition (wake) matches the distribution from generation (sleep)
Wake-sleep learning

- For each digit in training set
  - bottom-up pass: use recognition weights to stochastically set hidden states
    \[ p(h_j = 1) = \sigma(b_j + \sum_i v_i w_{ij}) \]
  - adjust generative weights to improve how model generates training data:
    \[ \Delta w_{ji} \propto h_j (h_i - \hat{h}_i) \]
  - \( \hat{h}_i \) is the probability of activating state \( i \) given inferred states \( h_j \)
Generative model for hand-written digits

- Generation:
  - use alternating Gibbs sampling from top-level assoc. memory
  - use directed weights to stochastically generate pixel probs. from sampled binary of 500 hidden units

- Recognition:
  - Use bottom-up weights to produce binary activities in two lower layers
  - use alternating Gibbs sampling in the top two layers
Demo: deep-belief network model (Hinton)