Energy-Based and Generative Models: Sigmoid Belief Networks

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

CSCI-736

2/23/2023

Thanks to Geoffrey Hinton & Fei-Fei Li

A spectrum of machine learning tasks

Typical Statistics----------Artificial Intelligence

- Low-dimensional data (e.g. less than 100 dimensions)
- Lots of noise in the data
- There is not much structure in the data, and what structure there is, can be represented by a fairly simple model.
- The main problem is distinguishing true structure from noise.

- High-dimensional data (e.g. more than 100 dimensions)
- The noise is not sufficient to obscure the structure in the data if we process it right.
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.
- The main problem is figuring out a way to represent the complicated structure so that it can be learned.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

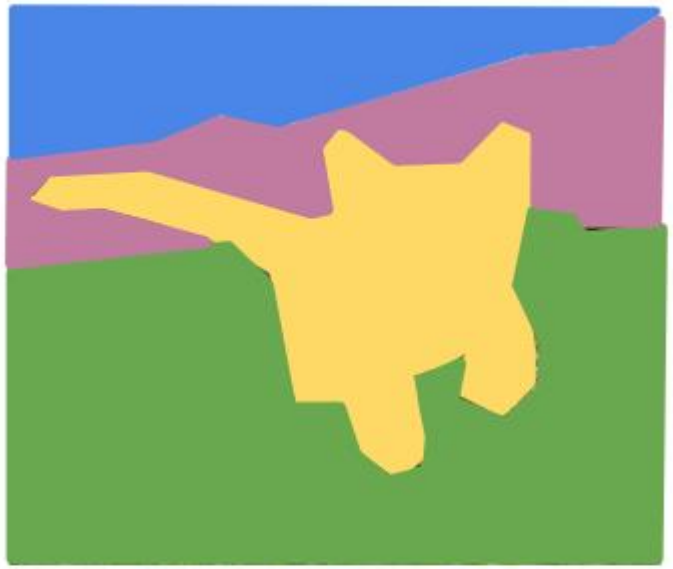
Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised Learning Tasks

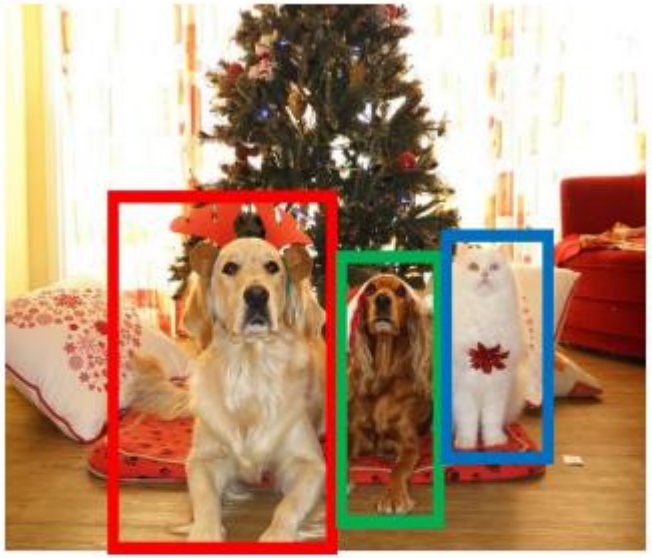
Cat

Classification



GRASS, CAT,
TREE, SKY

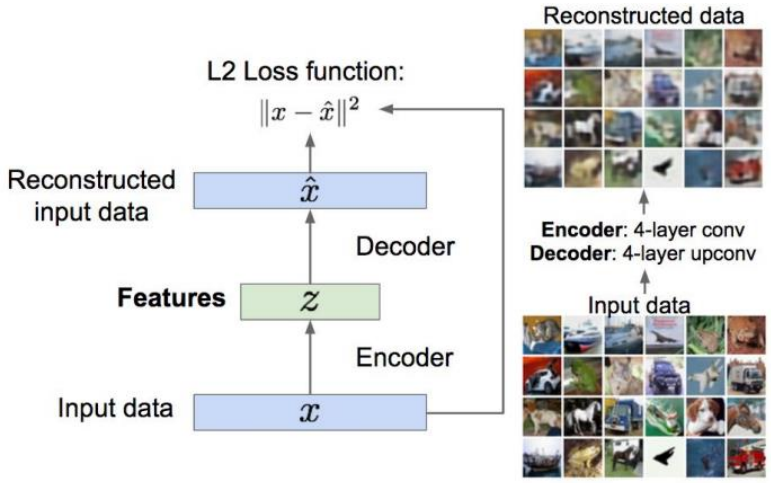
Semantic Segmentation



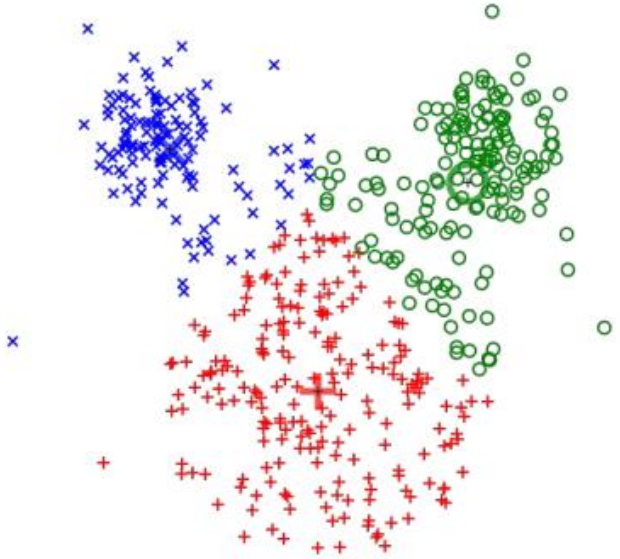
DOG, DOG, CAT

Object Detection

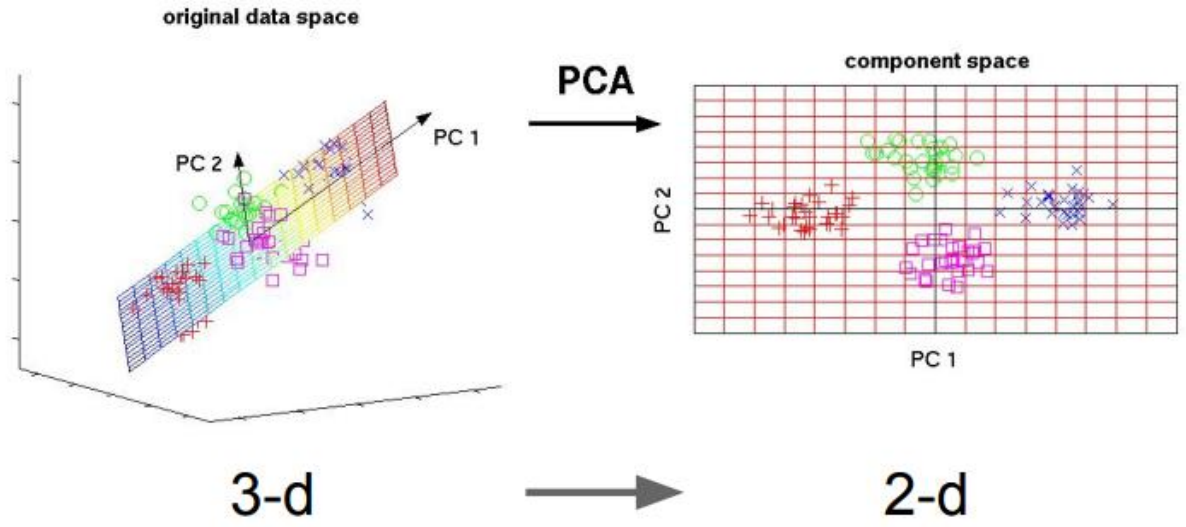
Unsupervised Learning Tasks



Autoencoders
(Feature learning)

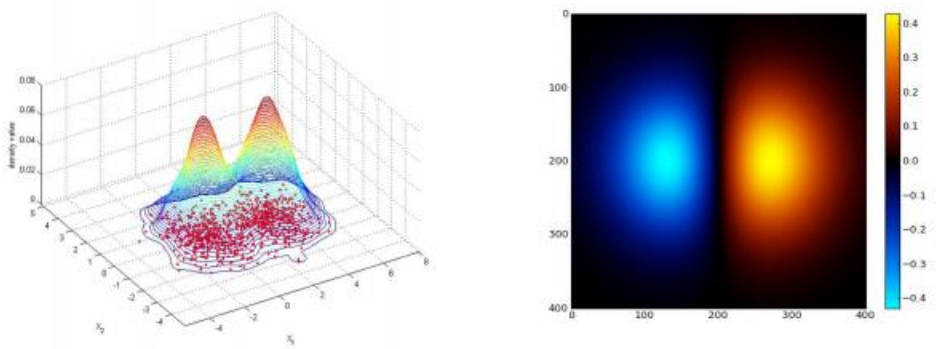


K-means clustering



3-d → 2-d

Principal Component Analysis
(Dimensionality reduction)



2-d density estimation

Generative Models

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.

- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
- Training generative models can also enable inference of latent representations that can be useful as general features

Taxonomy of Generative Models

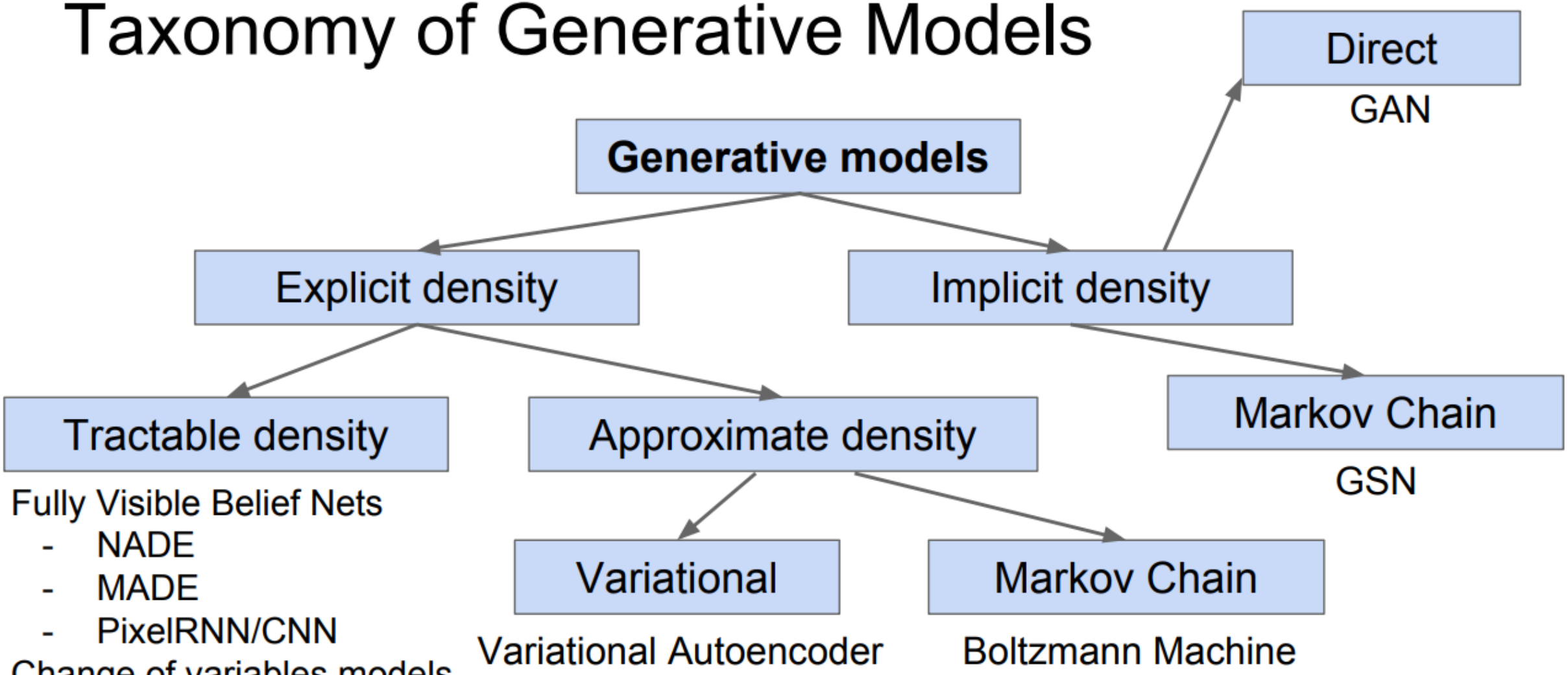
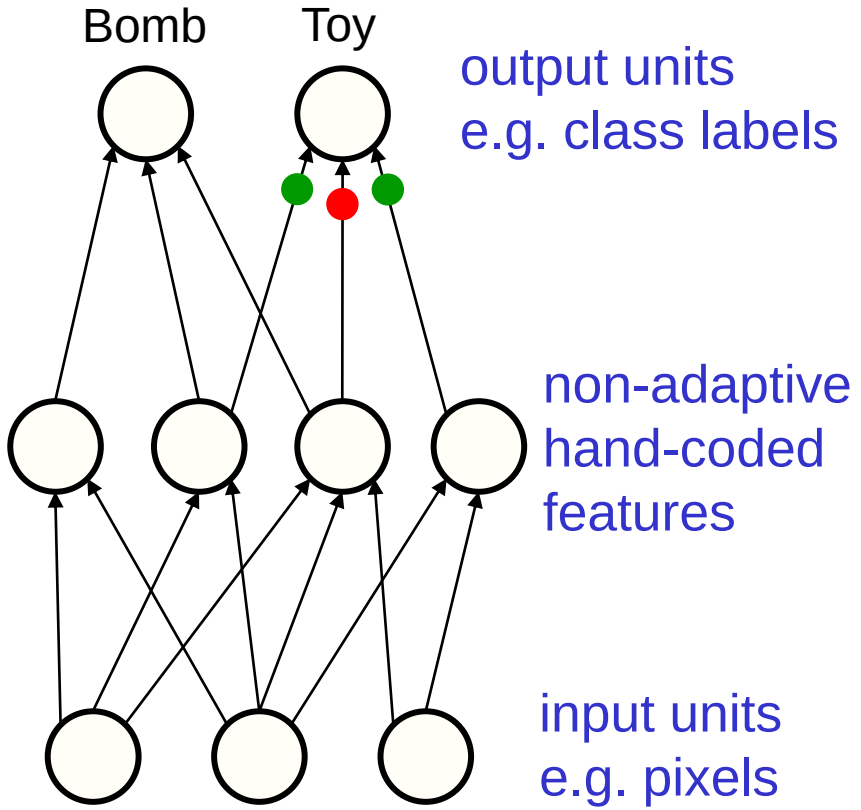


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Historical background:
First generation neural networks

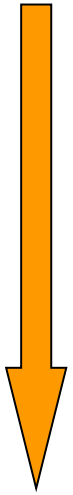
- Perceptrons (~1960) used a layer of hand-coded features and tried to recognize objects by learning how to weight these features.
  - There was a neat learning algorithm for adjusting the weights.
  - But perceptrons are fundamentally limited in what they can learn to do.



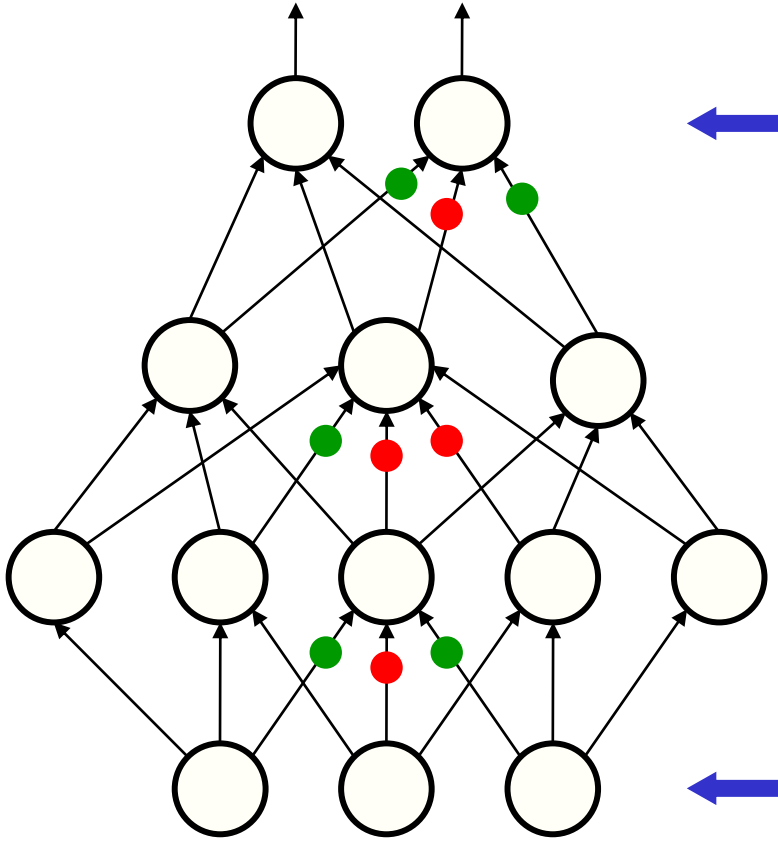
Sketch of a typical perceptron from the 1960's

Second generation neural networks (~1985)

Back-propagate error signal to get derivatives for learning



Compare outputs with correct answer to get error signal



outputs

hidden layers

input vector

What is wrong with back-propagation?

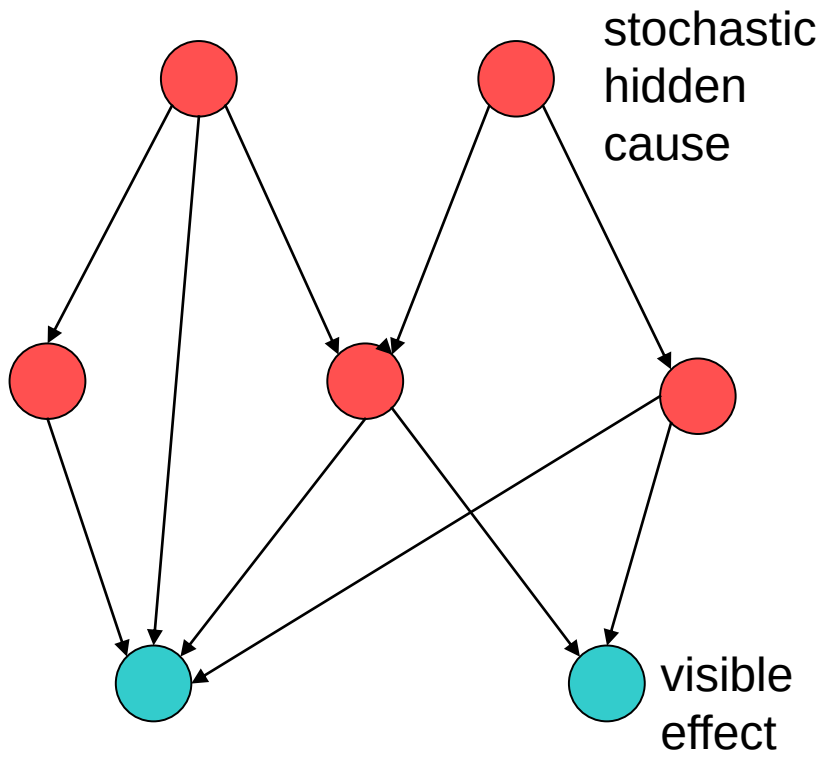
- It requires labeled training data.
  - Almost all data is unlabeled.
- The learning time does not scale well
  - It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.

Overcoming the limitations of back-propagation

- Keep the efficiency and simplicity of using a gradient method for adjusting the weights, but use it for modeling the structure of the sensory input.
  - Adjust the weights to maximize the probability that a generative model would have produced the sensory input.
  - Learn p(image) not p(label | image)
    - If you want to do computer vision, first learn computer graphics
- What kind of generative model should we learn?

Belief Nets

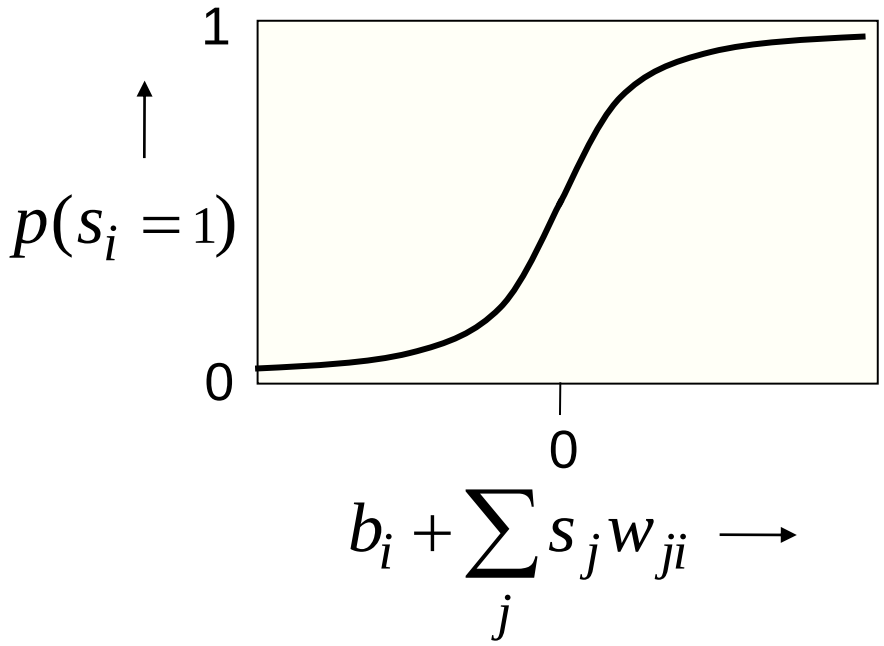
- A belief net is a directed acyclic graph composed of stochastic variables.
- We get to observe some of the variables and we would like to solve two problems:
- The inference problem: Infer the states of the unobserved variables.
- The learning problem: Adjust the interactions between variables to make the network more likely to generate the observed data.



We will use nets composed of layers of stochastic binary variables with weighted connections. Later, we will generalize to other types of variable.

Stochastic binary units
(Bernoulli variables)

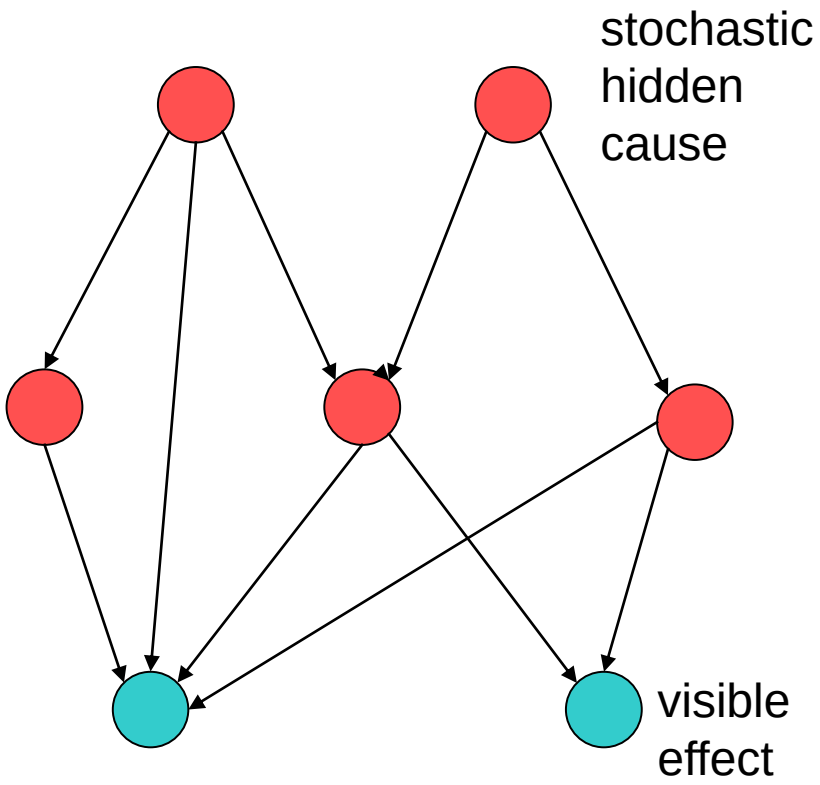
- These have a state of 1 or 0.
- The probability of turning on is determined by the weighted input from other units (plus a bias)



$$p(s_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ji})}$$

Learning Deep Belief Nets

- It is easy to generate an unbiased example at the leaf nodes, so we can see what kinds of data the network believes in.
- It is hard to infer the posterior distribution over all possible configurations of hidden causes.
- It is hard to even get a sample from the posterior.
- So how can we learn deep belief nets that have millions of parameters?



Questions??