

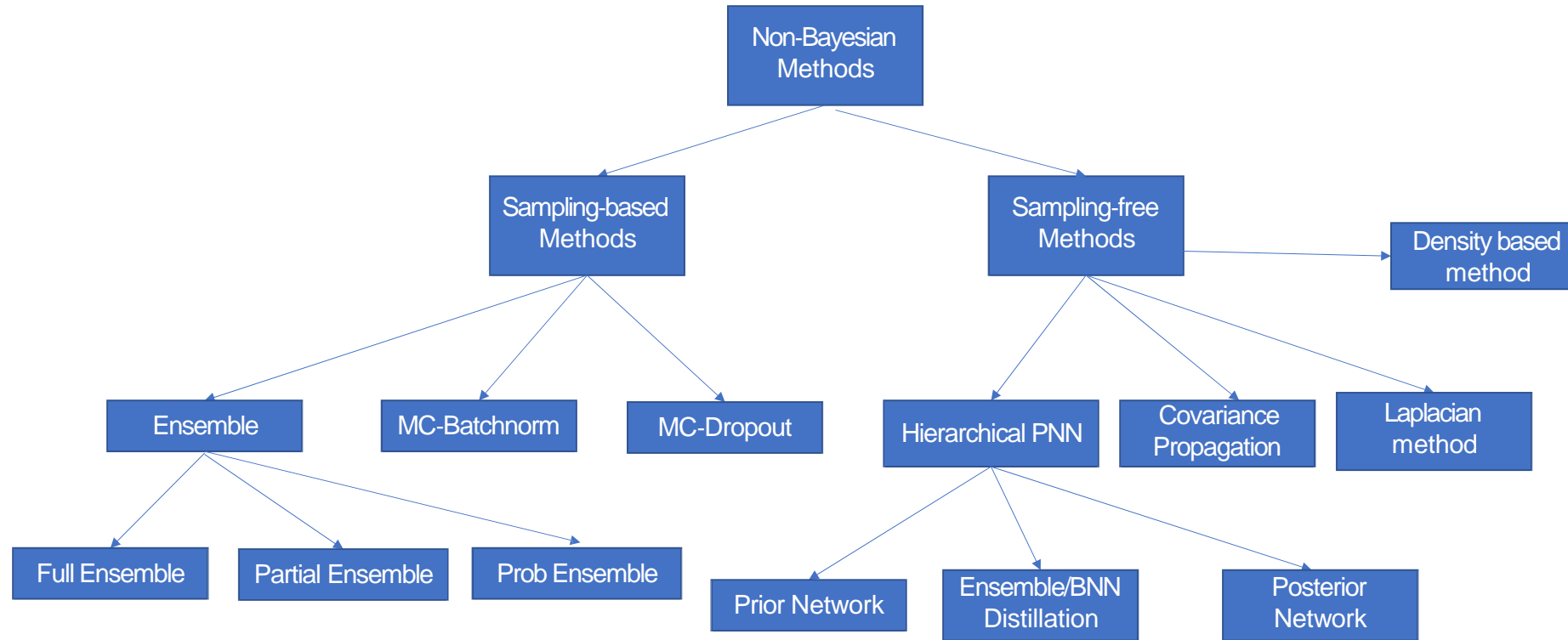
BNN and Deep PGMs

CSE 849 Deep Learning
Spring 2025

Zijun Cui

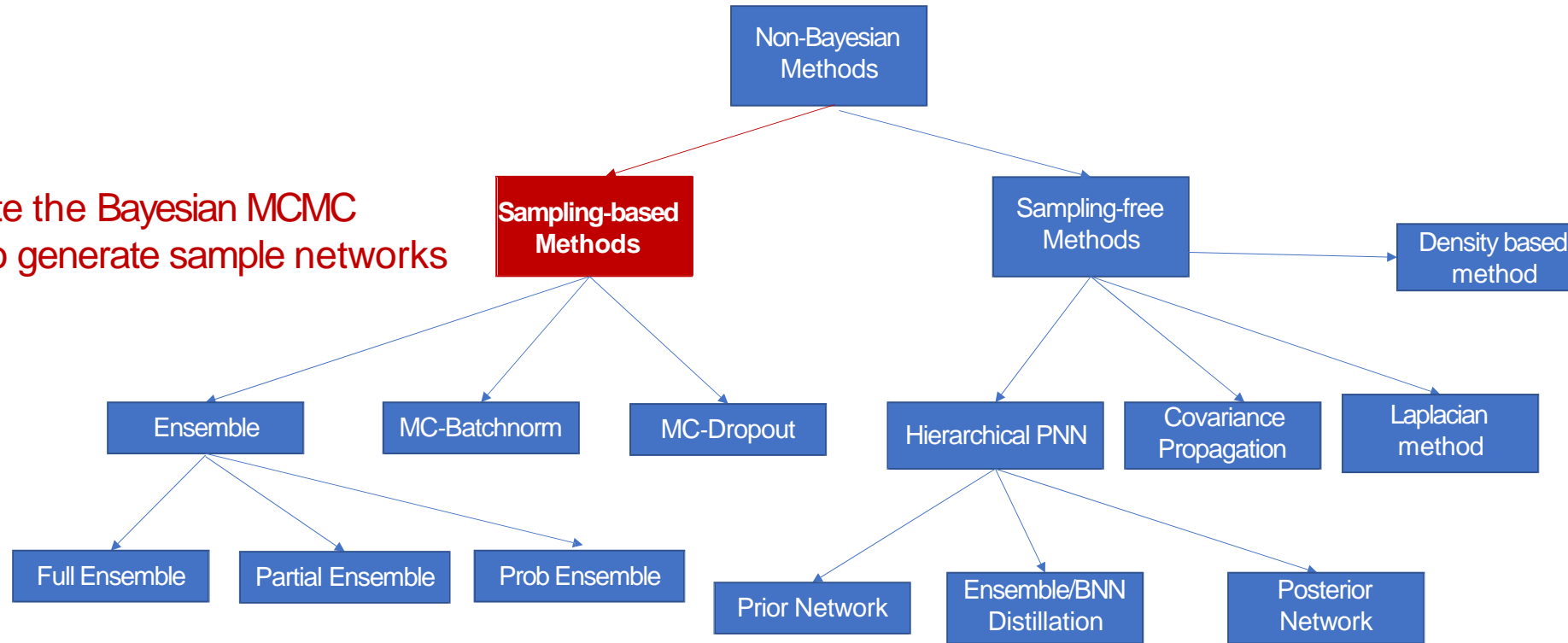
Non-Bayesian Uncertainty Estimation Methods

Types of Non-Bayesian Uncertainty Estimation Methods



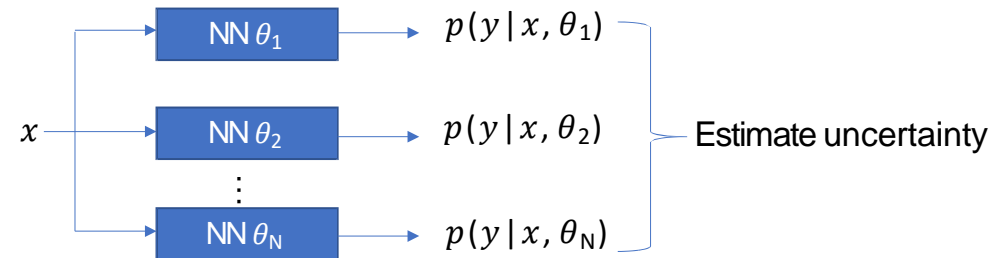
Types of Non-Bayesian Uncertainty Estimation Methods

approximate the Bayesian MCMC sampling to generate sample networks



Ensemble Methods

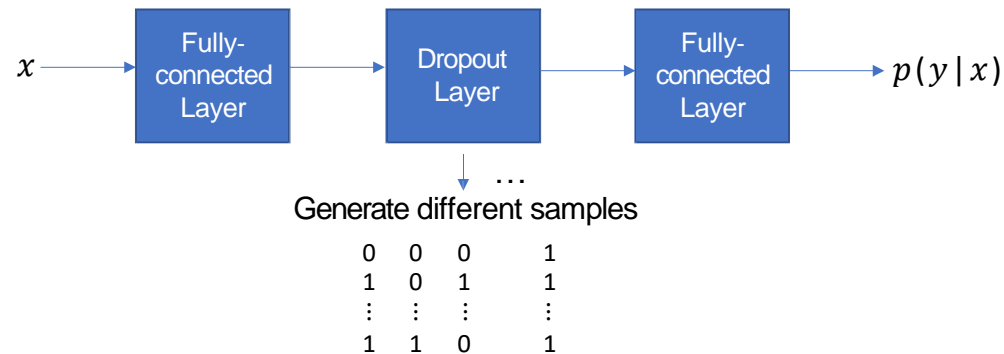
- Full Ensemble
 - Use different initializations to train multiple models (typically 5 to 10) to get multiple parameters of neural networks. Different parameters provide different outputs.
 - We have a bunch of output samples from ensemble models for uncertainty estimation



- Partial Ensemble
 - Partial Ensemble improves the ensemble method in terms of computational and memory costs. It reduces the number of parameters needed for constructing the whole ensemble model.
 - Training Partial Ensemble:
 - Step 1: pre-train the whole parameters $\theta = \{\theta_1, \theta_2\}$
 - Step 2: fix θ_1 , randomly initialize θ_2^i , then train $\theta_2^i, i = 1, 2, \dots, N$

MC-Dropout

- Dropout is a regularization technique for conventional deep neural networks, where it follows the Bernoulli distribution to decide which node to keep or drop.
- Key idea: apply the dropout during testing to produce different networks as BNN samples



- Collect the inference results $p^t(y|x)$ ($t = 1, 2, \dots, T$) from each sample and average these results.
- It has been shown that drop-out samples are mathematically equivalent to BNN samples.

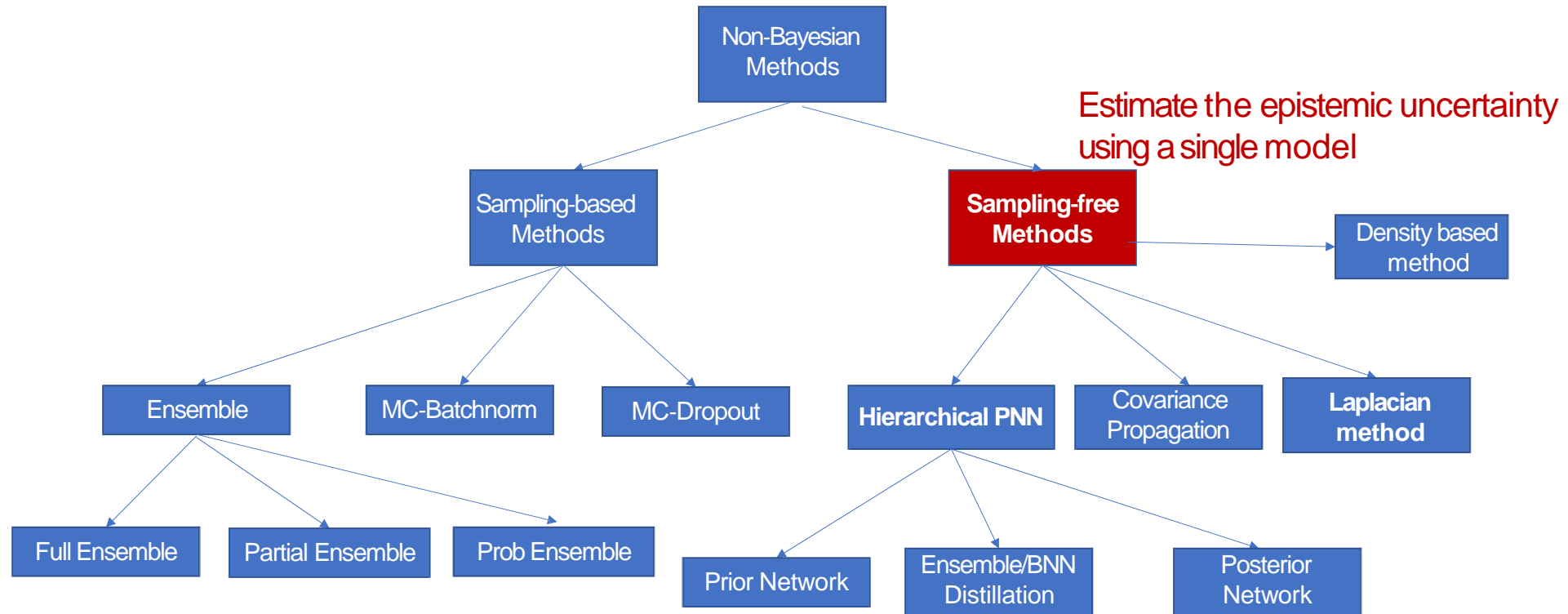
MC-Batch-normalization

- Batch normalization is used during training of DNNs to make the training stable and convergence faster.
- MC batch normalization selects a mini-batch B from training data D and performs batch normalization for each layer
- MC-Batchnorm also applies to testing by collecting the batch-normalization parameters from different mini-batches from training data to generate different outputs.
 - Sample different mini-batches from training data
 - Compute batch statistics (mean and variance) given sampled mini-batches
 - Run the same test input multiple times given batch statistics to get a distribution

Sampling based methods comparison

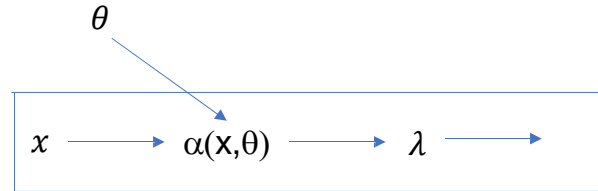
- For ensemble methods:
 - Advantages: better uncertainty estimation than MC-dropout and MC-batchnorm
 - Disadvantages: larger computational and memory cost
- Dropout method
 - It is easy to turn an existing deep net into a Bayesian one., faster than other techniques
 - Sampling at test time might be too expensive for computationally-demanding (eg real time) applications.
 - it has worse uncertainty estimation than Ensemble methods, but the training cost is lower
- MC-Batchnorm:
 - More efficient than MC dropout
 - Accuracy is not good

Types of Non-Bayesian Uncertainty Estimation Methods



Hierarchical Probabilistic Neural Network (HPNN)

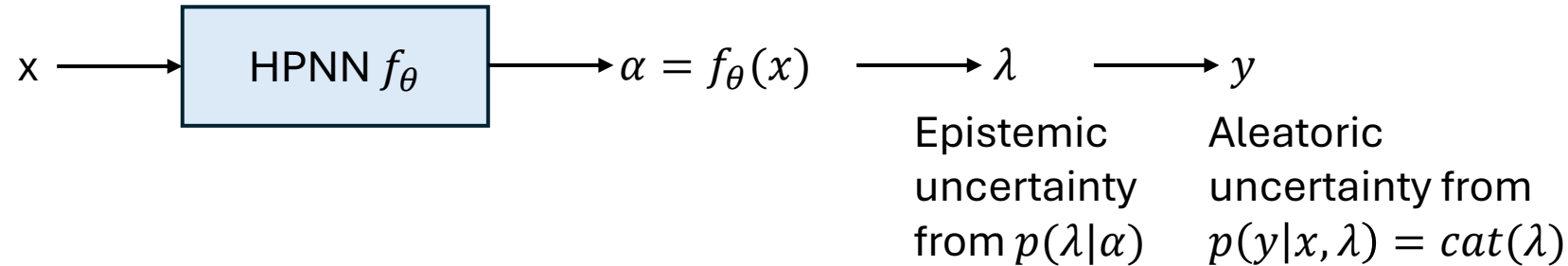
HPNN Architecture



- θ is the parameter of the HPNN model
 - $\alpha(x, \theta)$ are the output of HPNN and are the hyper-parameters that specify the prior distribution of λ , $p(\lambda|\alpha)$
 - λ parameterizes the distribution of $p(y|x, \lambda)$ and are treated as random variables
 - $p(y|x, \lambda)$ (the aleatoric distribution) can be used to estimate aleatoric uncertainty
 - $p(\lambda|\alpha)$ (the epistemic distribution) can estimate the epistemic uncertainty
-
- It is a sampling-free single-network method and hence is efficient during inference.
 - It requires a learning process to learn θ^* , the ML/MAP estimated parameters of HPNN.
 - It can only model the parameter distribution around the estimated parameters θ^* (local uncertainty)

HPNN for Classification and Regression

For classification problem:

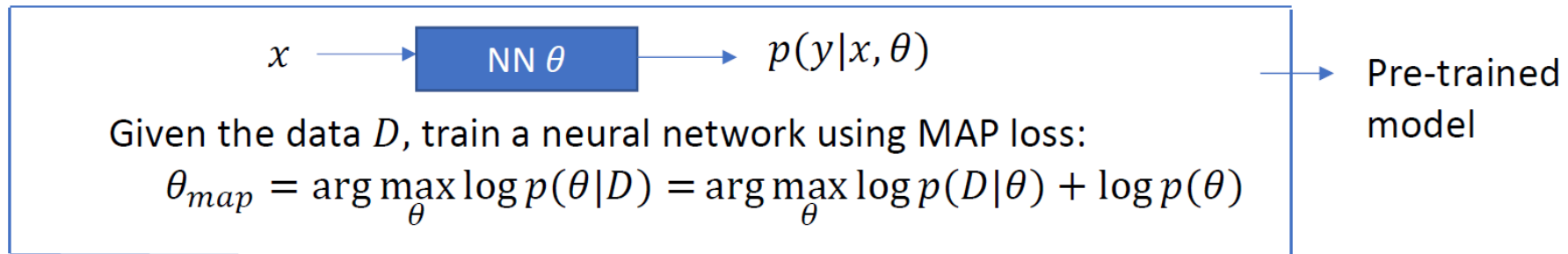


For regression problem:



Laplacian Approximation

- Can we train a single network to obtain different samples of the output distribution $p(y|x, \theta_1), p(y|x, \theta_1) \dots, p(y|x, \theta_N)$? Yes, use **Laplacian approximation**



- Laplacian Approximation: a method to approximate $p(\theta|D)$ by a Gaussian distribution around the mode for the pretrained probabilistic neural networks.

By using the second-order Taylor expansion of $\log p(\theta|D)$ at θ_{map}

$$\log p(\theta|D) \approx \log p(\theta_{map}|D) + J(\theta - \theta_{map}) + \frac{1}{2}(\theta - \theta_{map})^T H(\theta - \theta_{map})$$

$$J = \nabla_{\theta} \log p(\theta|D) \Big|_{\theta=\theta_{map}} = \mathbf{0} \quad H = \nabla_{\theta}^2 \log p(\theta|D) \Big|_{\theta=\theta_{map}}$$

$$\Rightarrow \log p(\theta|D) \approx \log p(\theta_{map}|D) + \frac{1}{2}(\theta - \theta_{map})^T H(\theta - \theta_{map})$$

$$\Rightarrow p(\theta|D) \approx p(\theta_{map}|D) \exp\left(\frac{1}{2}(\theta - \theta_{map})^T H(\theta - \theta_{map})\right)$$

Laplacian Approximation – cont'd

$$\Rightarrow p(\theta|D) \approx p(\theta_{map}|D) \exp\left(\frac{1}{2}(\theta - \theta_{map})^T H(\theta - \theta_{map})\right)$$

Unnormalized multivariate Gaussian distribution

$$\Rightarrow p(\theta|D) \approx N(\theta_{map}, (-H)^{-1}); H = \nabla_{\theta}^2 \log p(\theta|D) \Big|_{\theta=\theta_{map}}$$

- During testing:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \theta)p(\theta|D)d\theta \approx \frac{1}{S} \sum_{s=1}^S p(y^*|x^*, \theta_s); \theta_s \sim p(\theta|D) \approx N(\theta_{map}, (-H)^{-1})$$

- For Laplacian approximation
 - It can be applied to any pre-trained single network. No retraining needed to generate uncertainty.
 - For large models, computing the inverse of Hessian matrix may be difficult. It may need some approximations. For example, only consider $diag(H)$.
 - It only approximates local parameter distribution around θ_{map} .

Bayesian Neural Network Software

The major sampling and variational methods have been implemented in the following software

- TensorFlow probability
 - <https://www.tensorflow.org/probability>
- Edward (A library for probabilistic modeling, inference, and criticism)
 - <http://edwardlib.org/>
- Pyro (Deep Universal Probabilistic Programming, supported by PyTorch)
 - <https://pyro.ai/>
- Gen (An open-source stack for generative modeling and probabilistic inference)
 - <https://www.gen.dev/>

Bayesian Neural Networks: Additional Materials

- A first insight into Bayesian Neural Network

<https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526>

- Introduction to full Bayesian approach

<https://www.youtube.com/watch?v=jN5uYO9qlIc>

- The Bayesian interpretation of weight decay

<https://www.youtube.com/watch?v=vEPQNwxd1Y4>

Deep Probabilistic Graphical Models

- Introduction of Probabilistic Graphical Models
 - Role of probability in AI
 - Graphical models vs Deep nets
- Deep probabilistic Graphical Models
- Combining Deep Models with PGMs

Next step for AI

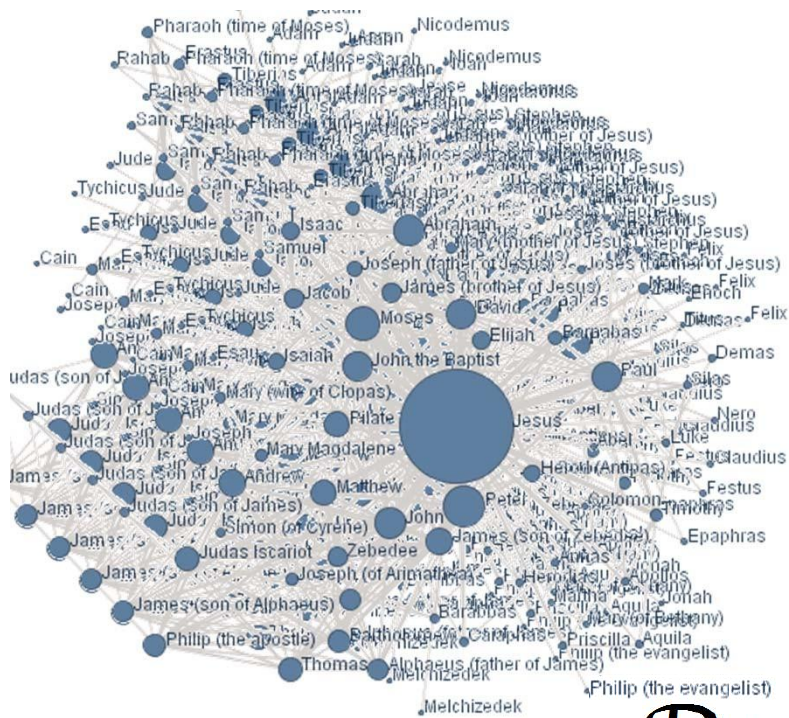
- First AI systems focused on logic:
 - Pre-programmed rules.
- Second wave of AI concerns ability to sense and perceive information
 - Leveraging neural networks to learn over time.
- But, neither solution can do things that human beings do naturally as we navigate the world.
 - They can't think through multiple potential scenarios based on data that you have on-hand while conscious of potential data that you don't have.

Role of Probability in AI

- Probabilistic computing allows us to
 1. Deal with uncertainty in natural data around us
 2. Predict events in the world with an understanding of data and model uncertainty
- Predicting what will happen next in a scenario, as well as effects of our actions, can only be done if we know how to model the world around us with probability distributions
 - Reasoning instead of Recognition!
- Augmenting deep learning with probabilistic methods opens door to understanding why AI systems make the decisions they make.
 - Will help with issues like tackling bias in AI systems.

What Are Graphical Models?

Graph



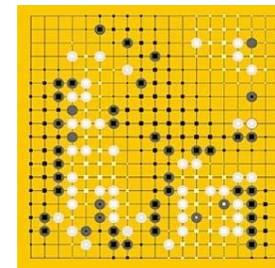
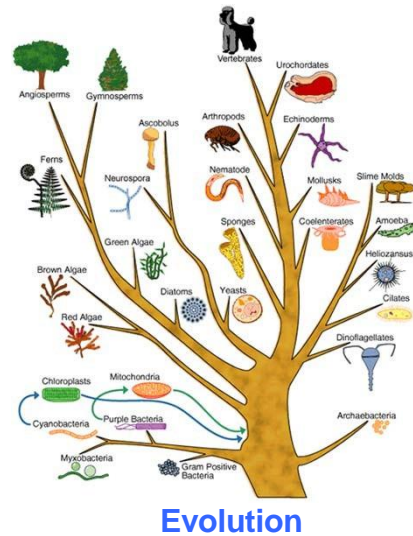
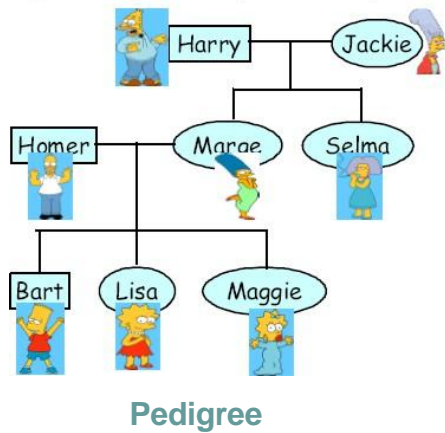
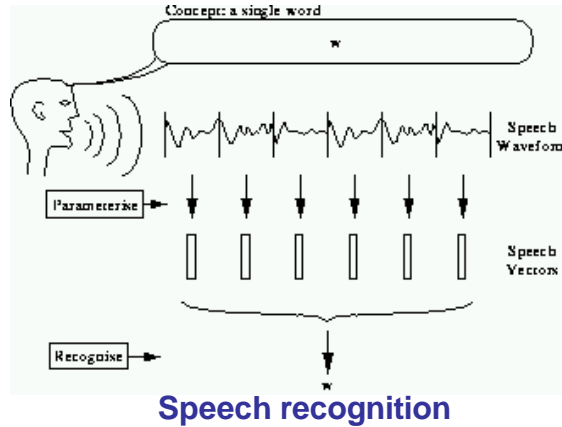
Model

\mathcal{M}_G

Data

$$\mathcal{D} \equiv \{X_1^{(i)}, X_2^{(i)}, \dots, X_m^{(i)}\}_{i=1}^N$$

Reasoning under uncertainty!



The Fundamental Questions

- Representation

- How to capture/model uncertainties in possible worlds?
- How to encode our domain knowledge/assumptions/constraints?

- Inference

- How do I answers questions/queries according to my model and/or based given data?

e.g.: $P(X_i | \mathcal{D})$

- Learning

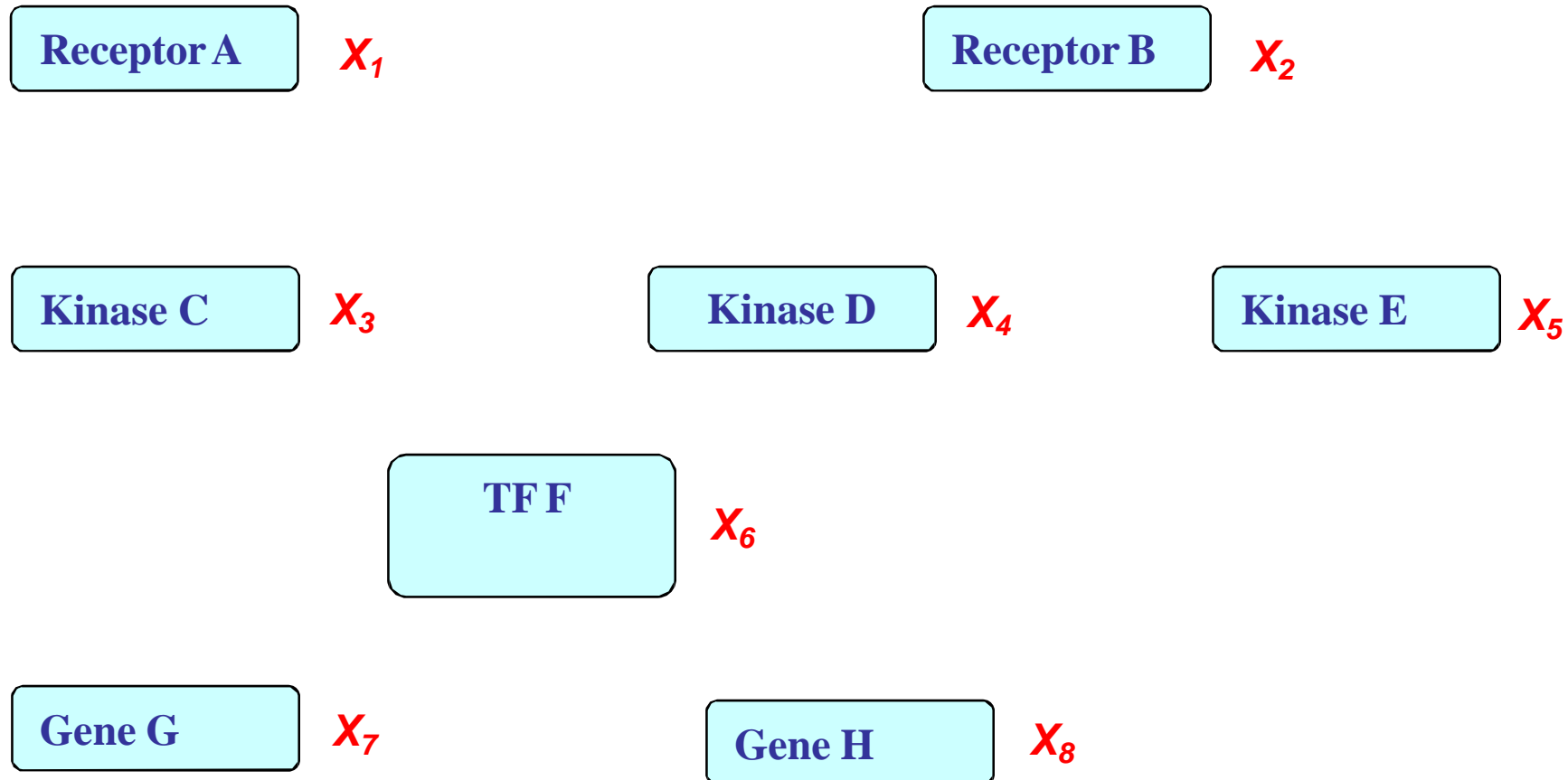
- What model is "right" for my data?

e.g.: $\mathcal{M} = \arg \max_{\mathcal{M} \in \mathcal{M}} F(\mathcal{D}; \mathcal{M})$

What is a Graphical Model?

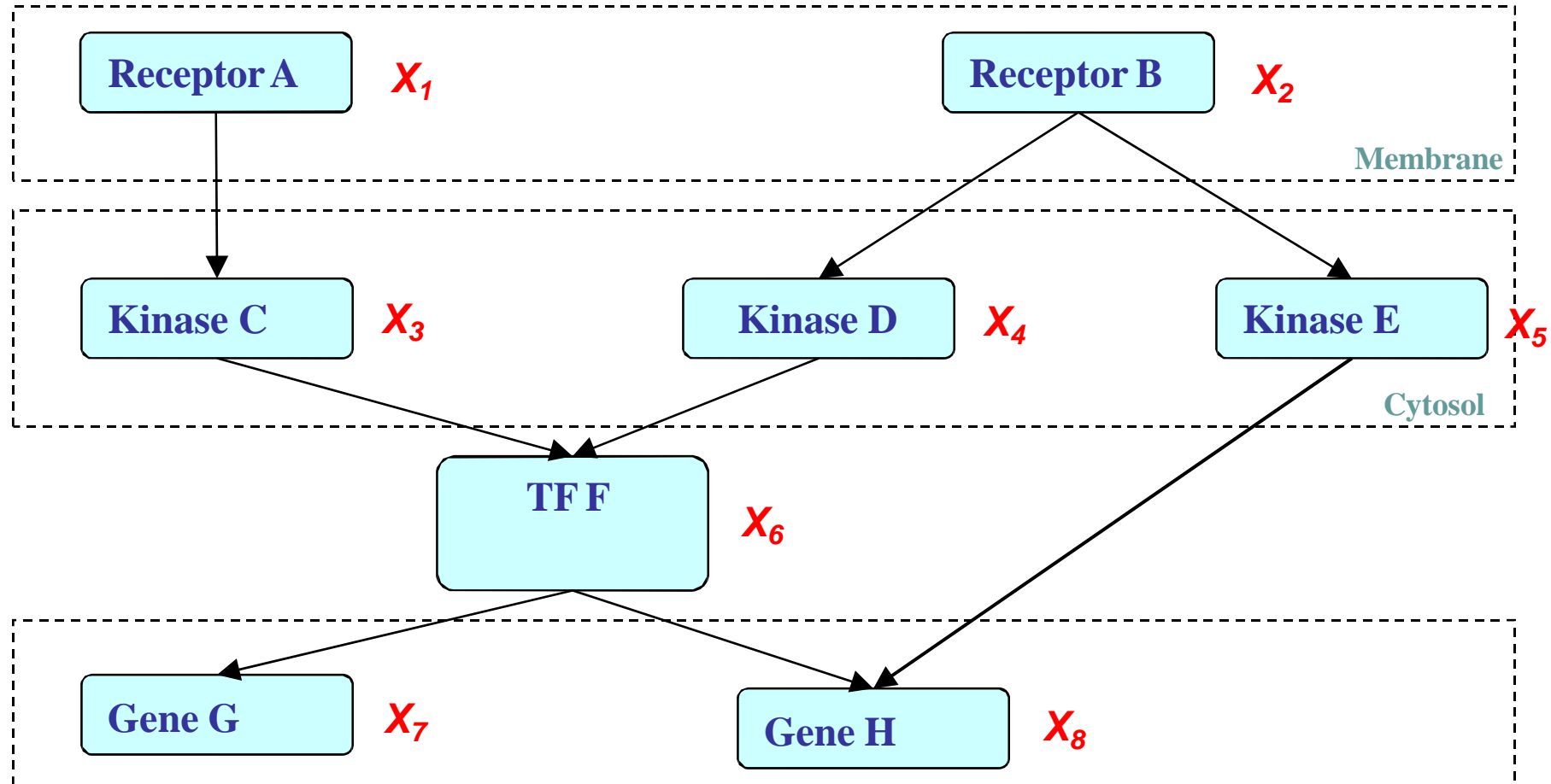
--- Multivariate Distribution in High-D Space

- A possible world for cellular signal transduction:



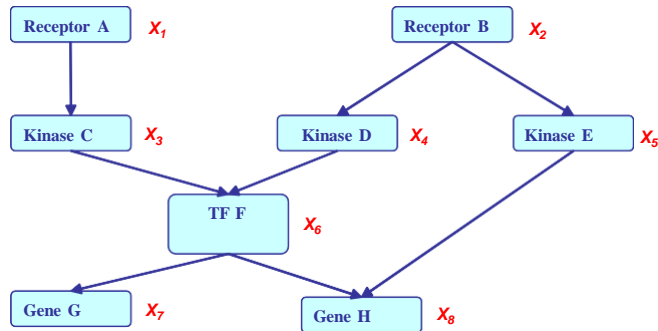
GM: Structure Simplifies Representation

- Dependencies among variables



Probabilistic Graphical Models

- If X_i 's are **conditionally independent** (as described by a **PGM**), the joint can be factored to a product of simpler terms, e.g.,

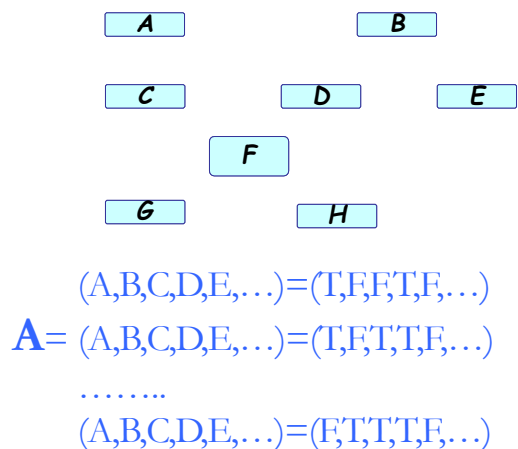


$$\begin{aligned} &P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) \\ &= P(X_1) P(X_2) P(X_3/X_1) P(X_4/X_2) P(X_5/X_2) \\ &\quad P(X_6/X_3, X_4) P(X_7/X_6) P(X_8/X_5, X_6) \end{aligned}$$

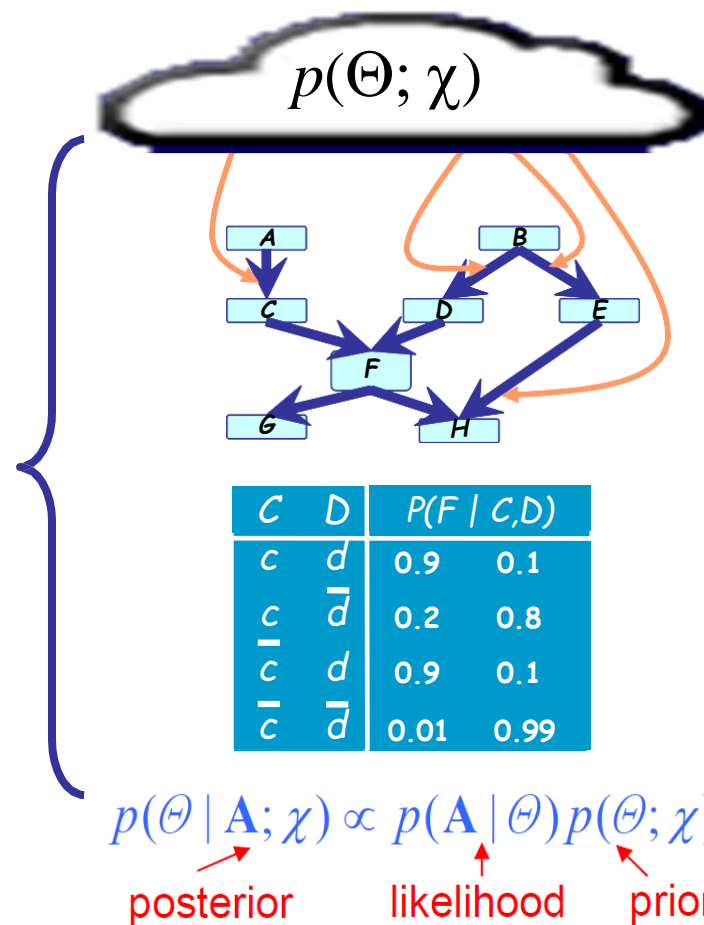
- Why we may favor a PGM?
 - Incorporation of domain knowledge and causal (logical) structures
1+1+2+2+2+4+2+4=18, a 16-fold reduction from 2^8 in representation cost !

GM: MLE and Bayesian Learning

- Probabilistic statements of Θ is conditioned on the values of the observed variables \mathbf{A}_{obs} and prior $p(\Theta | \chi)$

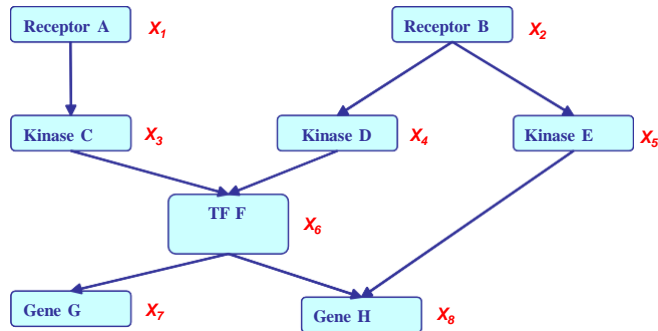


$$\Theta_{\text{Bayes}} = \int \Theta p(\Theta | \mathbf{A}, \chi) d\Theta$$



Probabilistic Graphical Models

- If X_i 's are **conditionally independent** (as described by a **PGM**), the joint can be factored to a product of simpler terms, e.g.,



$$\begin{aligned} &P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8) \\ &= P(X_1) P(X_2) P(X_3/X_1) P(X_4/X_2) P(X_5/X_2) \\ &\quad P(X_6/X_3, X_4) P(X_7/X_6) P(X_8/X_5, X_6) \end{aligned}$$

- Why we may favor a PGM?

- Incorporation of domain knowledge and causal (logical) structures

$1+1+2+2+2+4+2+4=18$, a 16-fold reduction from 2^8 in representation cost !

- Bayesian Philosophy

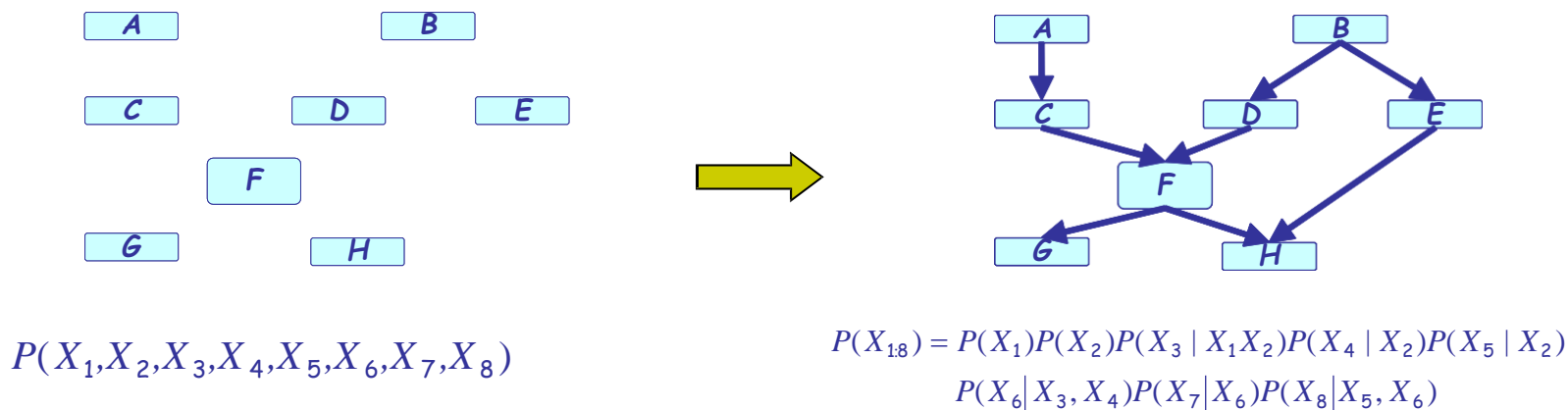
- Knowledge meets data



So What Is a PGM After All?

- The informal blurb:

- It is a smart way to write/specify/compose/design exponentially-large probability distributions without paying an exponential cost, and at the same time endow the distributions with *structured semantics*



- A more formal description:

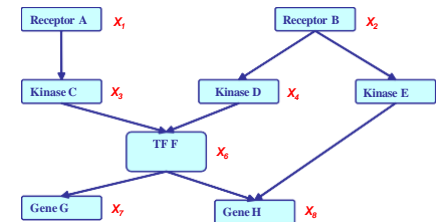
- It refers to a family of distributions on a set of random variables that are compatible with all the probabilistic independence propositions encoded by a graph that connects these variables

Two types of GMs

- **Directed edges** give **causality** relationships (**Bayesian Network** or **Directed Graphical Model**):

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

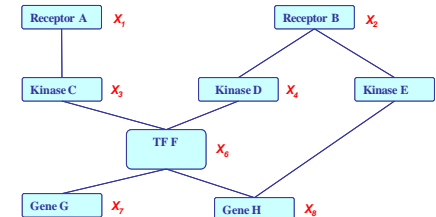
$$= P(X_1) P(X_2) P(X_3/X_1) P(X_4/X_2) P(X_5/X_2) P(X_6/X_3, X_4) P(X_7/X_6) P(X_8/X_5, X_6)$$



- **Undirected edges** simply give **correlations** between variables (**Markov Random Field** or **Undirected Graphical model**):

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

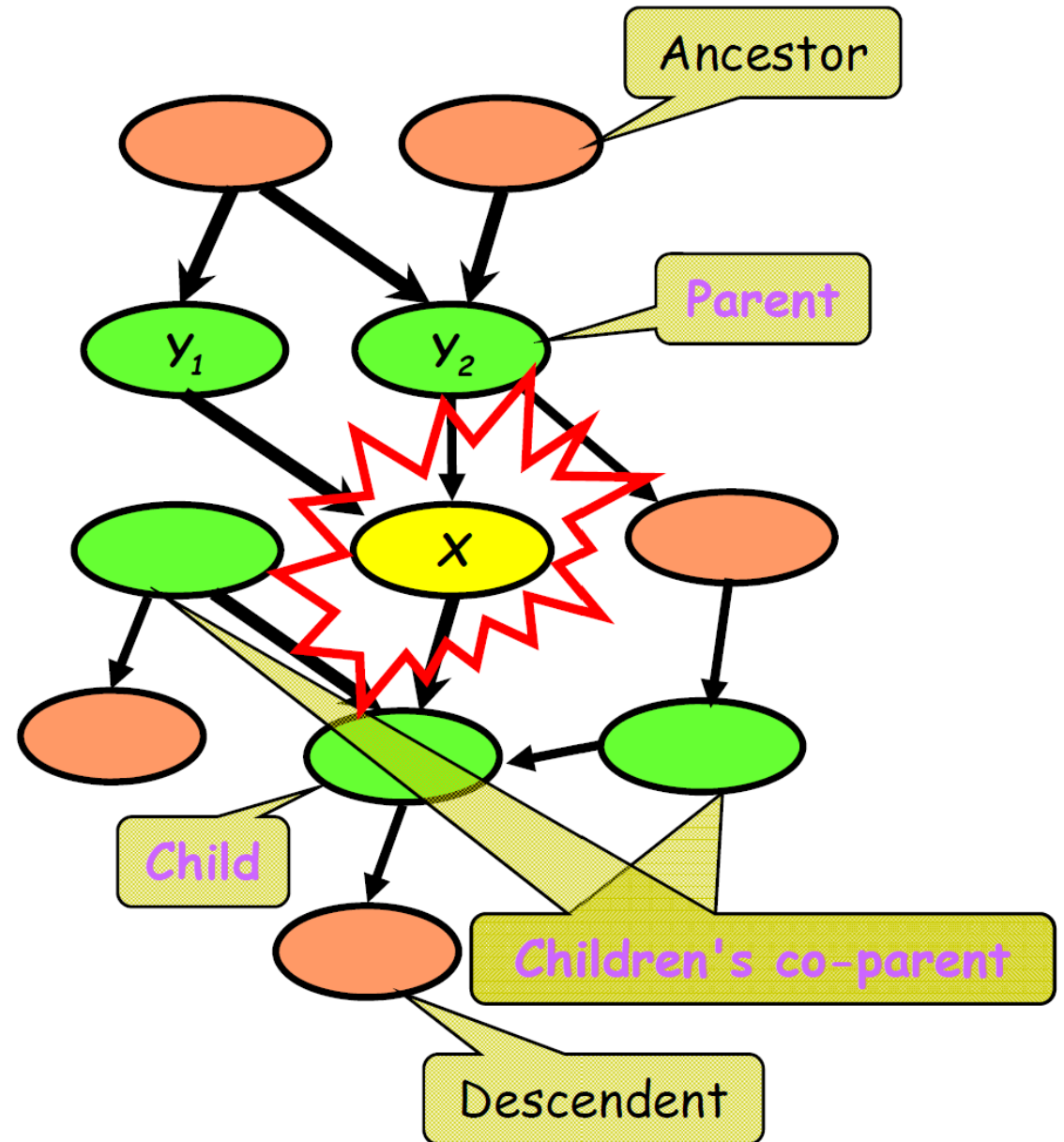
$$= \frac{1}{Z} \exp\{E(X_1) + E(X_2) + E(X_3, X_1) + E(X_4, X_2) + E(X_5, X_2) + E(X_6, X_3, X_4) + E(X_7, X_6) + E(X_8, X_5, X_6)\}$$



Bayesian Networks

Structure: Directed Acyclic Graph (**DAG**)

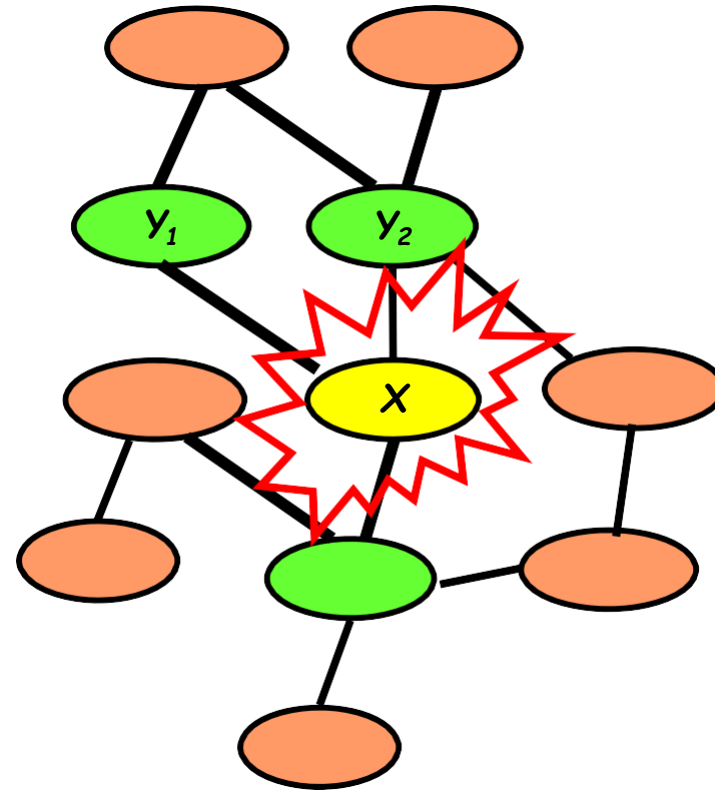
- Meaning: a node is **conditionally independent** of every other node in the network outside its **Markov blanket**
- Local conditional distributions (**CPD**) and the **DAG** completely determine the **joint** dist.
- Give **causality** relationships, and facilitate a **generative** process



Markov Random Fields

Structure: *undirected graph*

- Meaning: a node is **conditionally independent** of every other node in the network given its **Directed neighbors**
- Local contingency functions (**potentials**) and the **cliques** in the graph completely determine the **joint** dist.
- Give **correlations** between variables, but no explicit way to generate samples



GMs are your old friends

Density estimation

Parametric and nonparametric methods

Regression

Linear, conditional mixture, nonparametric

Classification

Generative and discriminative approach

Fancier applications:

Reinforcement learning

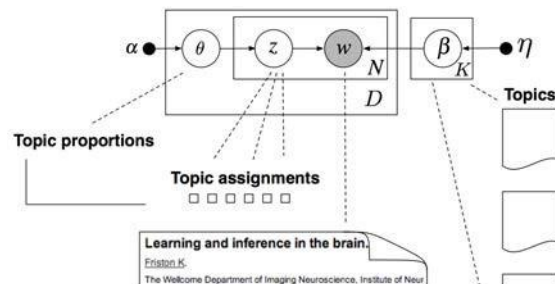
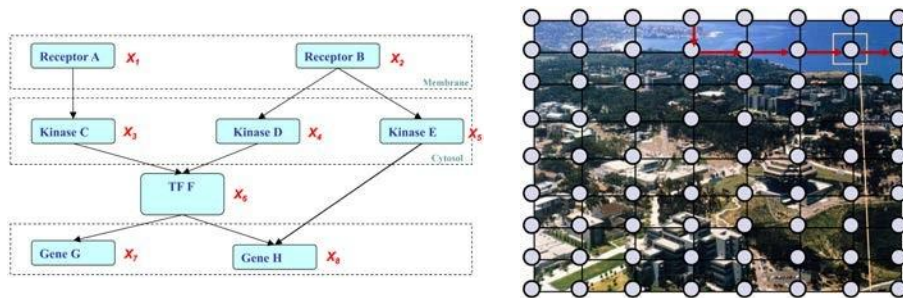
Machine translation

...

Graphical models vs. Deep nets

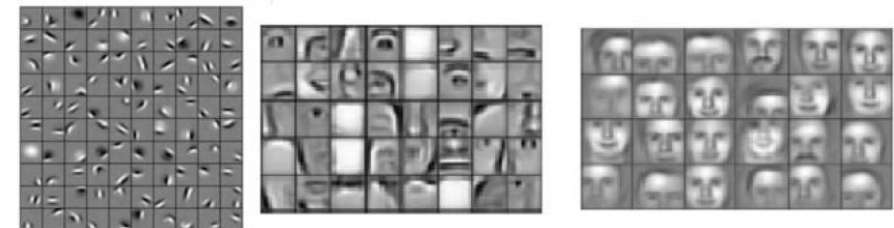
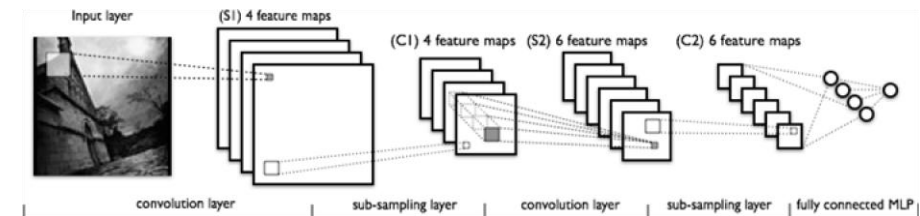
Graphical models

- Representation for encoding meaningful knowledge and the associated uncertainty in a graphical form



Deep neural networks

- Learn representations that facilitate computation and performance on the end-metric (intermediate representations are not guaranteed to be meaningful)



Graphical models vs. Deep nets

Graphical models

- Representation for encoding meaningful knowledge and the associated uncertainty in a graphical form
- Learning and inference are based on a rich toolbox of well-studied (structure-dependent) techniques (e.g., EM, message passing, VI, MCMC, etc.)
- Graphs represent models

Deep neural networks

- Learn representations that facilitate computation and performance on the end-metric (intermediate representations are not guaranteed to be meaningful)
- Learning is predominantly based on the gradient descent method (aka backpropagation); Inference is often trivial and done via a “forward pass”
- Graphs represent computation

Graphical models vs. Deep nets

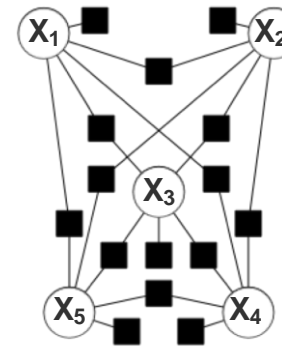
Graphical models

Utility of the graph

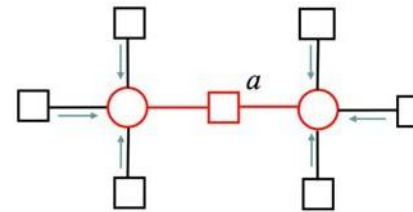
- synthesizing a global loss function from local structure
 - potential function, feature function, etc.
- designing sound and efficient inference algorithms
 - Sum-product, mean-field, etc.
- inspire approximation and penalization
 - Structured MF, Tree-approximation, etc.
- monitoring theoretical and empirical behavior and accuracy of inference

Utility of the loss function

- A major measure of quality of the learning algorithm and the model



$$\log P(X) = \sum_i \log \phi(x_i) + \sum_{i,j} \log \psi(x_i, x_j)$$

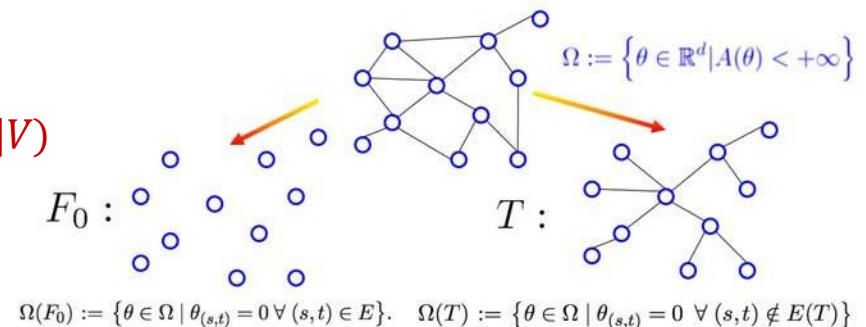


$$m_{i \rightarrow a}(x_i) = \prod_{c \in N(i) \setminus a} m_{c \rightarrow i}(x_i)$$

$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i \rightarrow a}(x_i)$$

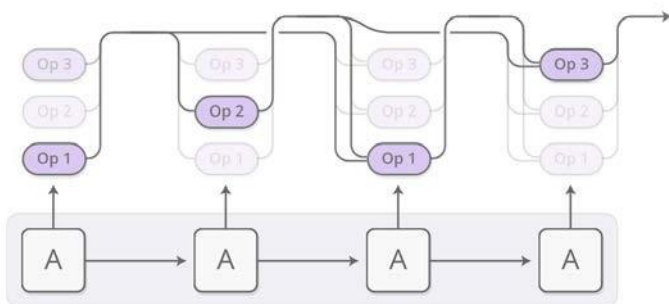
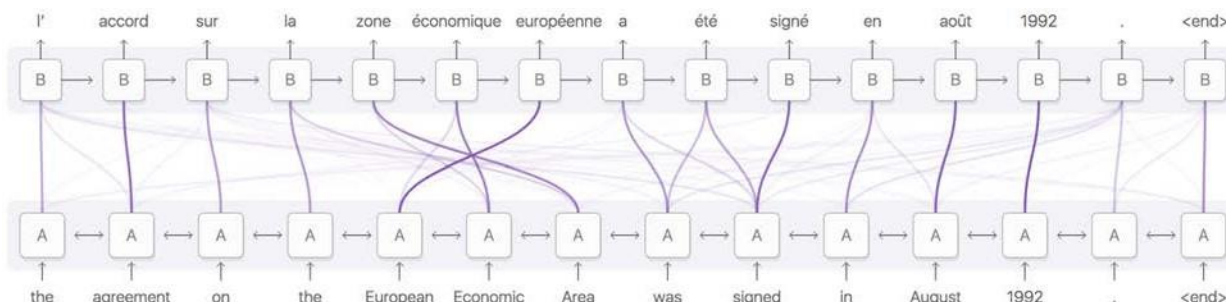
$$m_{a \rightarrow i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} m_{j \rightarrow a}(x_j)$$

$$Q(H) \sim P(H|V)$$



$$\theta = \operatorname{argmax}_{\theta} P_{\theta}(V)$$

Graphical models vs. Deep nets




Deep neural networks

Utility of the network

- conceptually synthesize complex decision hypothesis
 - stage-wise projection and aggregation
- organizing computational operations
 - stage-wise update of latent states
- designing processing steps and computing modules
 - Layer-wise parallelization
- No obvious utility in evaluating DL inference algorithms

Utility of the Loss Function

- Global loss? Well it is complex and non-convex...

	DL	 ? ML (e.g., GM)
Empirical goal:	e.g., classification, feature learning	e.g., latent variable inference, transfer learning
Structure:	Graphical	Graphical
Objective:	Something aggregated from local functions	Something aggregated from local functions
Vocabulary:	Neuron, activation function, ...	Variable, potential function, ...
Algorithm:	A single, unchallenged, inference algorithm – Backpropagation (BP)	A major focus of open research, many algorithms, and more to come
Evaluation:	On a black-box score – end performance	On almost every intermediate quantity
Implementation:	Many tricks	More or less standardized
Experiments:	Massive, real data (GT unknown)	Modest, often simulated data (GT known)

Deep Probabilistic Graphical Models

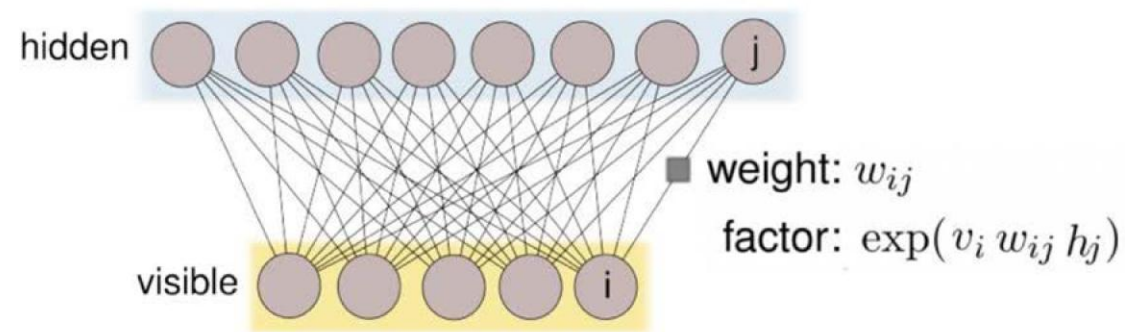
- Introduction of Probabilistic Graphical Models
 - Role of probability in AI
 - Graphical models vs Deep nets
- **Deep probabilistic Graphical Models**
- Combining Deep Models with PGMs

Graphical Models vs. Deep Nets

- ❑ So far:
 - ❑ Graphical models are representations of probability distributions
 - ❑ Neural networks are function approximators (with no probabilistic meaning)
- ❑ Some of the neural nets are in fact proper graphical models (i.e., ***units/neurons represent random variables***):
 - ❑ Boltzmann machines (Hinton & Sejnowsky, 1983)
 - ❑ Restricted Boltzmann machines (Smolensky, 1986)
 - ❑ Learning and Inference in sigmoid belief networks (Neal, 1992)
 - ❑ Fast learning in deep belief networks (Hinton, Osindero, Teh, 2006)
 - ❑ Deep Boltzmann machines (Salakhutdinov and Hinton, 2009)

I: Restricted Boltzmann Machines

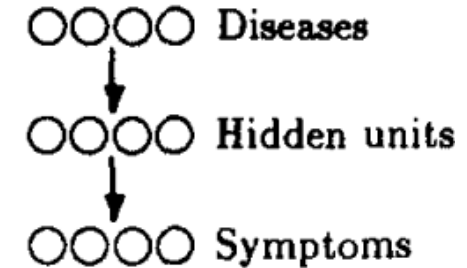
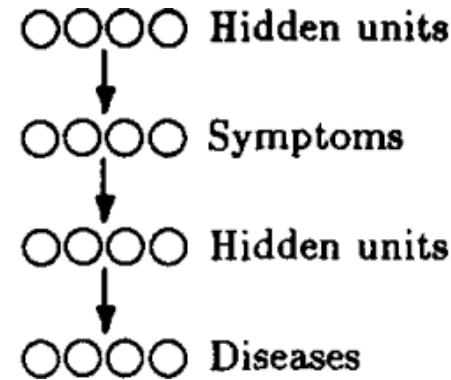
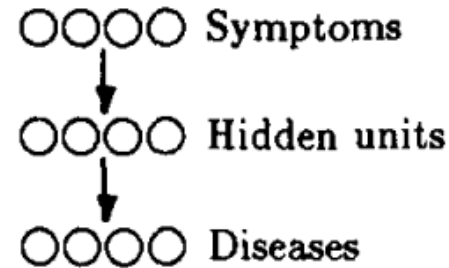
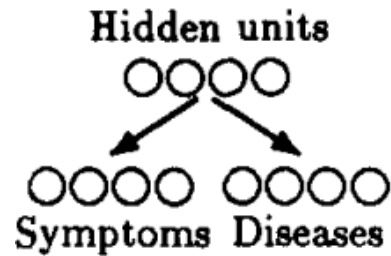
- RBM is a Markov random field represented with a bi-partite graph
- All nodes in one layer/part of the graph are connected to all in the other; no inter-layer connections



- Joint distribution:

$$P(v, h) = \frac{1}{Z} \exp \left\{ \sum_{i,j} w_{ij} v_i h_j + \sum_i b_i v_i + \sum_j c_j h_j \right\}$$

II: Sigmoid Belief Networks

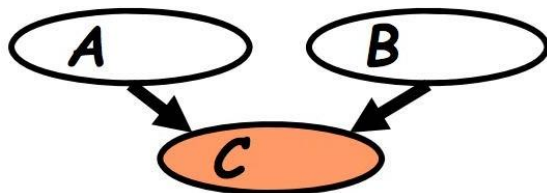


from Neal,
1992

- Sigmoid belief nets are simply Bayesian networks over binary variables with conditional probabilities represented by sigmoid functions (also called regression BN):

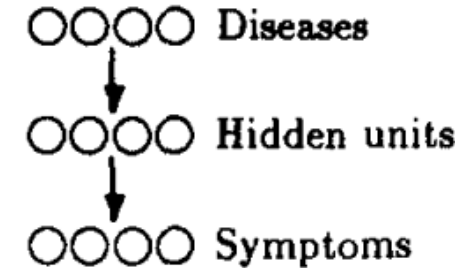
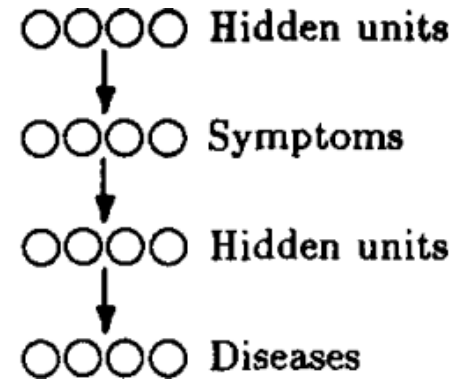
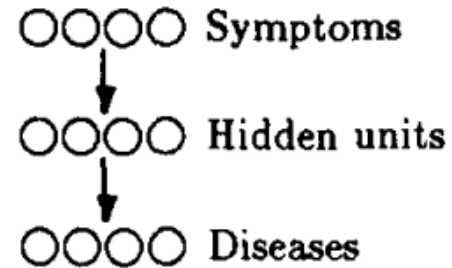
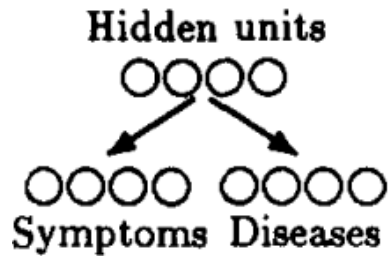
$$P(x_i | \pi(x_i)) = \sigma \left(x_i \sum_{x_j \in \pi(x_i)} w_{ij} x_j \right)$$

- Bayesian networks exhibit a phenomenon called “explain away effect”



If A correlates with C, then the chance of B correlating with C decreases. \Rightarrow A and B become correlated given C.

II: Sigmoid Belief Networks

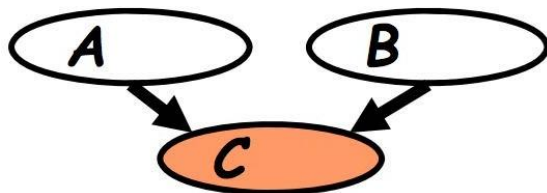


from Neal,
1992

- Sigmoid belief nets are simply Bayesian networks over binary variables with conditional probabilities represented by sigmoid functions:

$$P(x_i | \pi(x_i)) = \sigma \left(x_i \sum_{x_j \in \pi(x_i)} w_{ij} x_j \right)$$

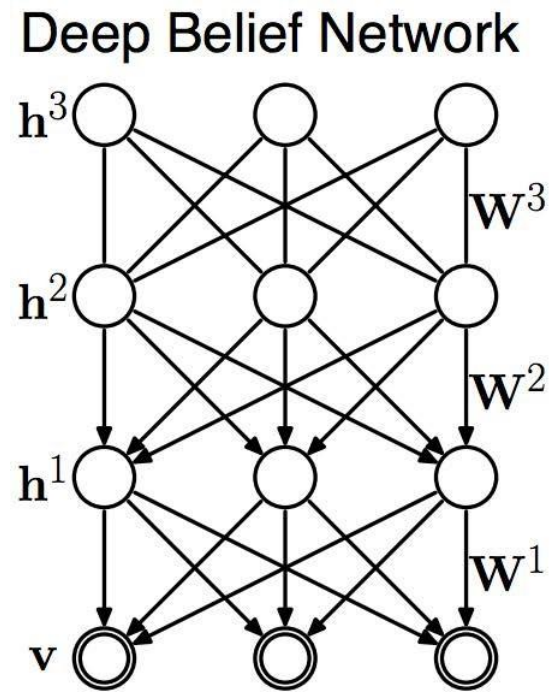
- Bayesian networks exhibit a phenomenon called “explain away effect”



Note:

Due to the “explain away effect,” when we condition on the visible layer in belief networks, hidden variables all become dependent.

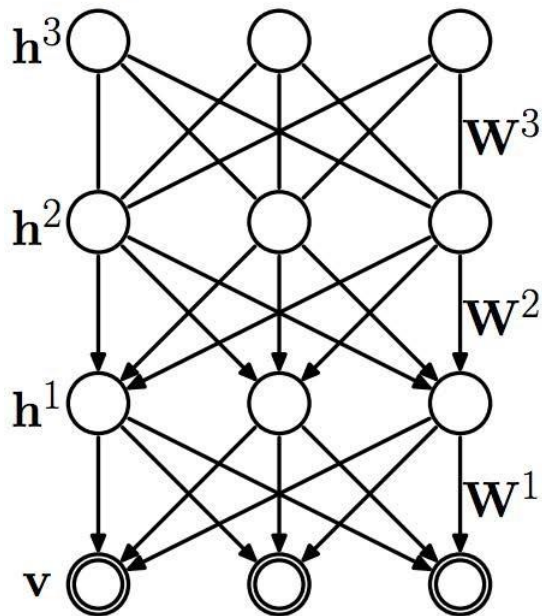
III: Deep Belief Nets



- DBNs are hybrid graphical models (chain graphs)

Deep Belief Networks

Deep Belief Network



- DBNs represent a joint probability distribution

$$P(v, h^1, h^2, h^3) = P(h^2, h^3)P(h^1|h^2)P(v|h^1)$$

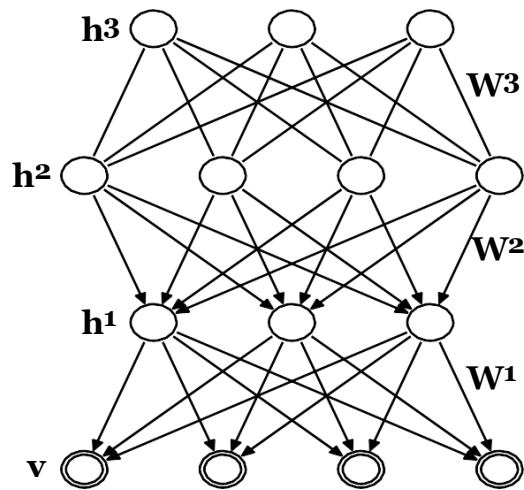
- Note that $P(h^2, h^3)$ is an RBM and the conditionals $P(h^1|h^2)$ and $P(v|h^1)$ are represented in the sigmoid form
- The model is trained by optimizing the log likelihood for a given data $\log P(v)$

Challenges:

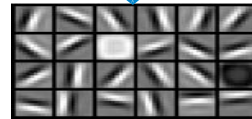
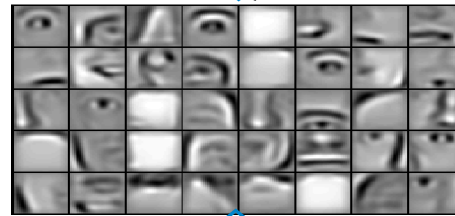
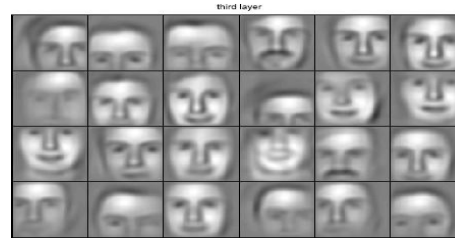
- Exact inference in DBNs is problematic due to explain away effect
- Training is done in two stages:
 - greedy pre-training + ad-hoc fine-tuning; no proper joint training
- Approximate inference is feed-forward (bottom-up)

DBN Representation Learning

Convolutional DBN



Faces



Groups of parts.

$$\mathbf{h}_3^* = \operatorname{argmax}_{\mathbf{h}_3} p(\mathbf{h}_3 | \mathbf{v})$$

Object Parts

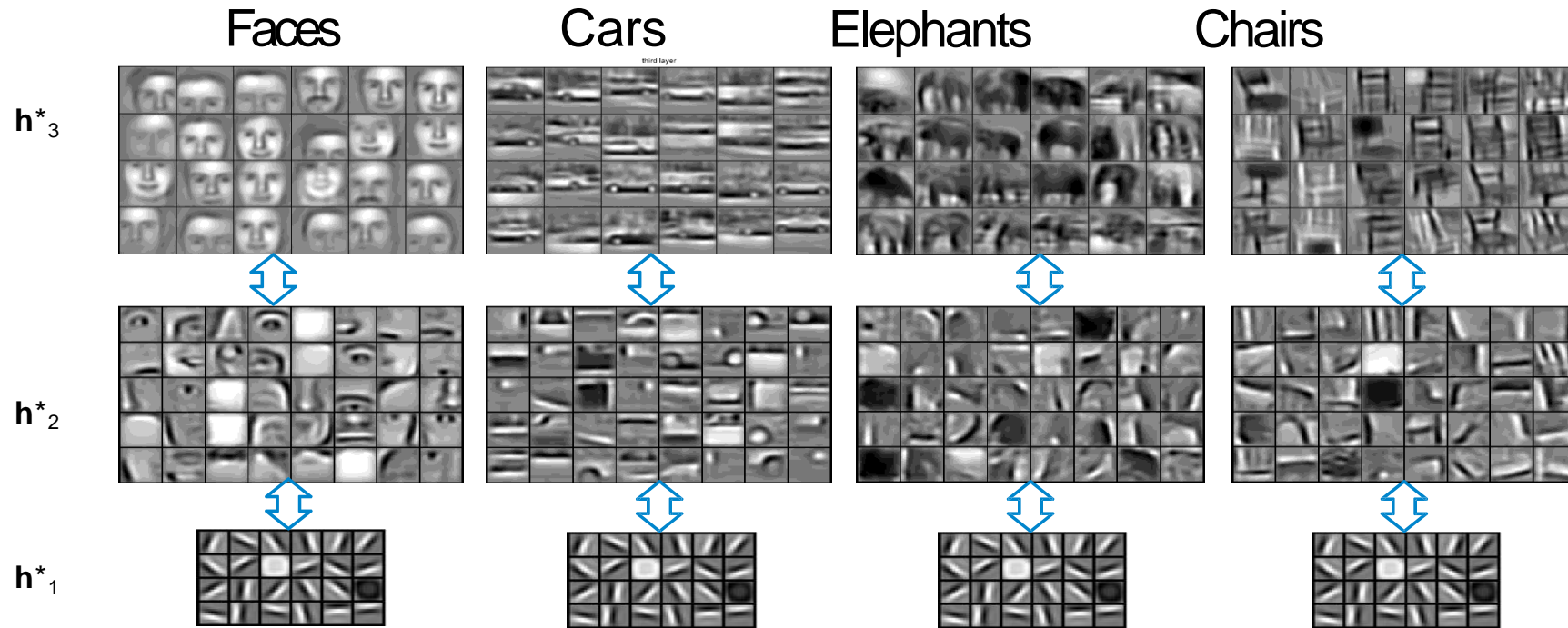
$$\mathbf{h}_2^* = \operatorname{argmax}_{\mathbf{h}_2} p(\mathbf{h}_2 | \mathbf{v})$$

Edges

$$\mathbf{h}_1^* = \operatorname{argmax}_{\mathbf{h}_1} p(\mathbf{h}_1 | \mathbf{v})$$

Lee et.al., ICML 2009

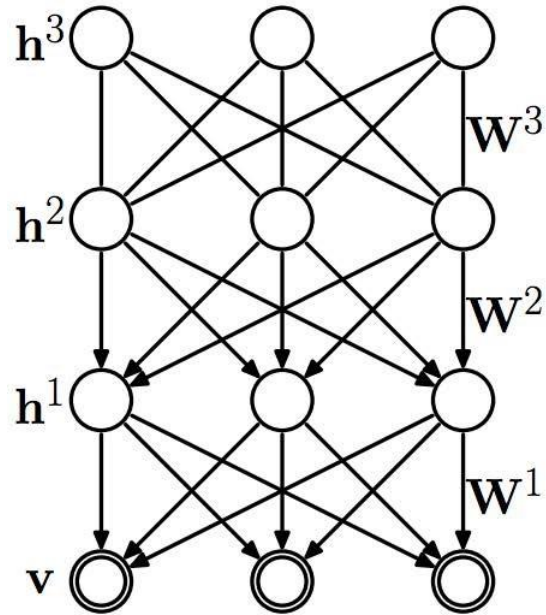
DBN Representation Learning



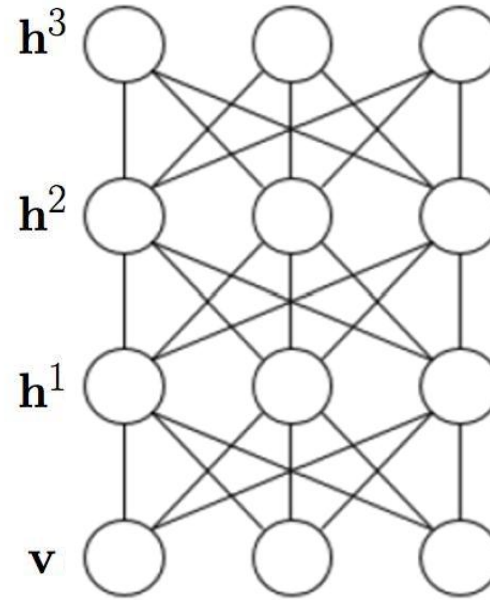
Lee et.al., ICML 2009

Deep Belief Nets and Boltzmann Machines

Deep Belief Network



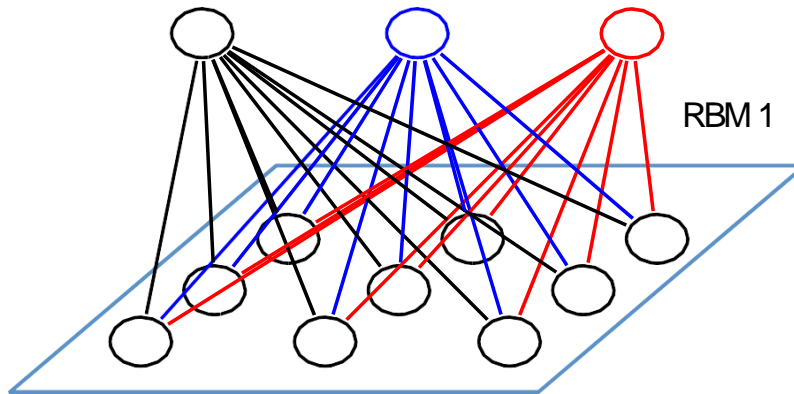
Deep Boltzmann Machine



- DBMs are fully un-directed models (Markov random fields):
 - Can be trained similarly as RBMs via MCMC (Hinton & Sejnowski, 1983)
 - Use a variational approximation of the data distribution for faster training (Salakhutdinov & Hinton, 2009)
 - Similarly, can be used to initialize other networks for downstream tasks

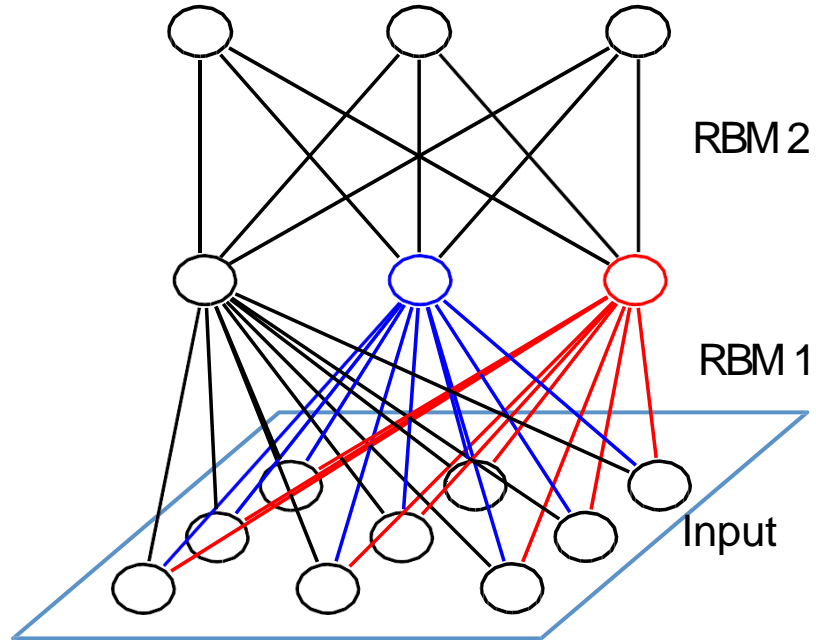
Deep Boltzmann Machines

RBMs can be used as building blocks to construct the Deep Boltzmann Machines (DBMs) by stacking RBMs on top of each other .



(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Deep Boltzmann Machines



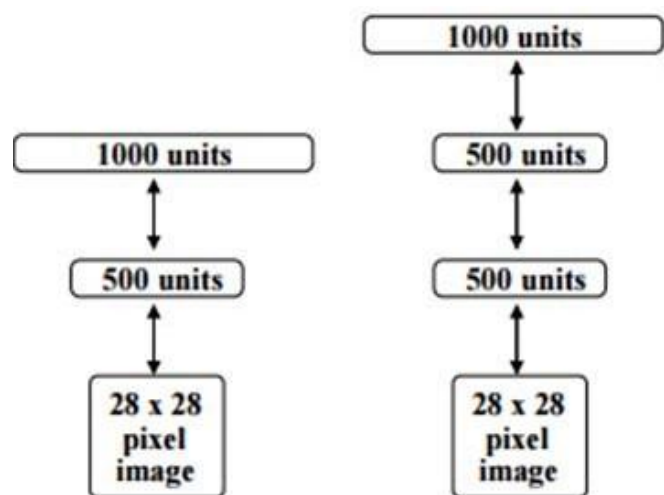
(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

DBM Applications

- Generative data modeling $\mathbf{x} \sim p(\mathbf{x})$
 - Single modality data modeling
 - Multi-modal data modeling
- Feature/representation learning $\mathbf{h} \sim \operatorname{argmax}_{\mathbf{h}} p(\mathbf{h}|\mathbf{x})$
- Classification $y^* = \operatorname{argmax}_y p(\mathbf{y}|\mathbf{x})$

Generative Model of Handwritten Digits

Generate \mathbf{x} by sampling $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$



Two DBMs



Training data



**Generated data
by 2-layer DBM**



**Generated data
by 3-layer DBM**

Generative Model of 3-D Objects

R. Salakhutdinov and G. Hinton

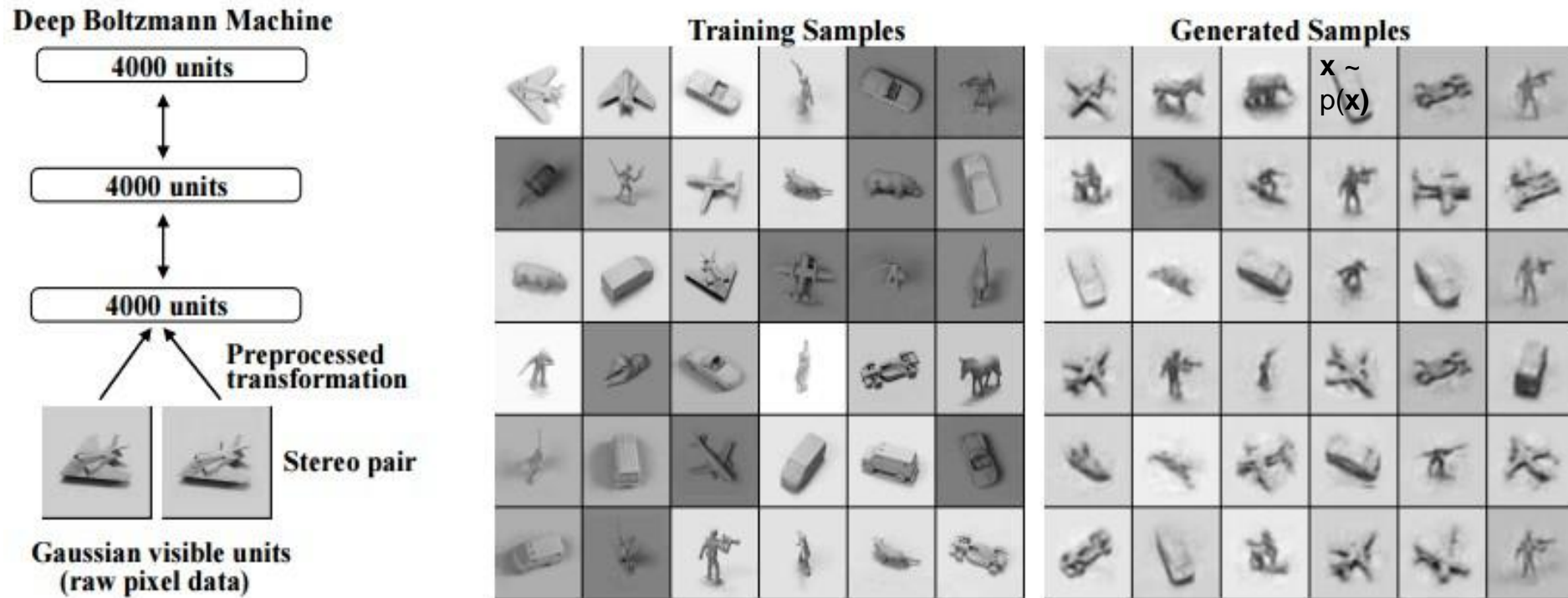


Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

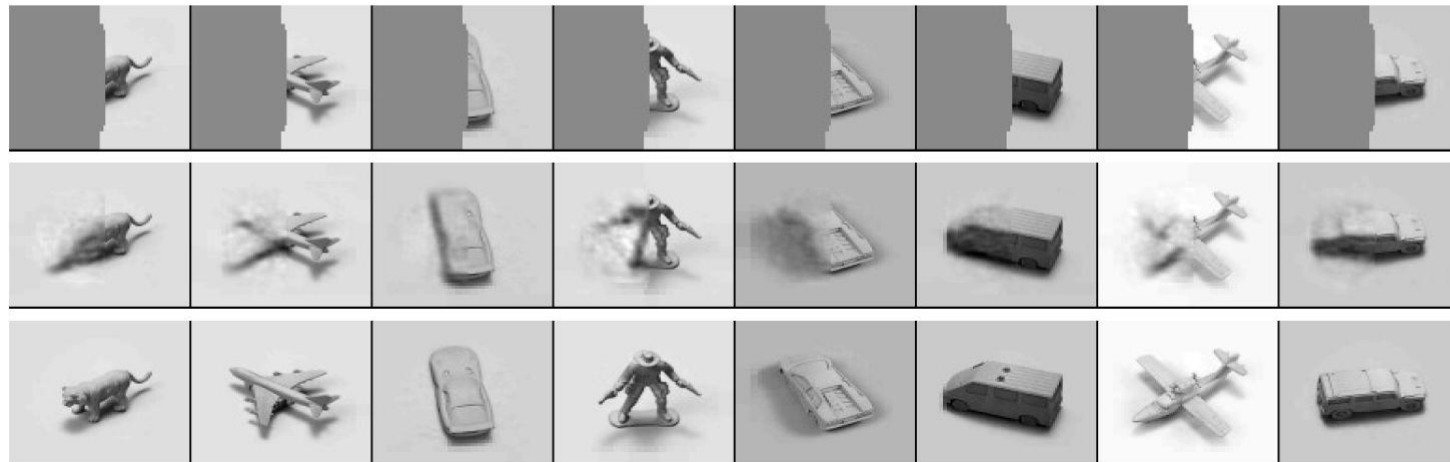
Generate \mathbf{x} by Gibbs sampling $p(\mathbf{x})$

Pattern Completion

Given an occluded input $\hat{\mathbf{x}}$, infer the original pattern \mathbf{x} by

$$\mathbf{h}_L^* = \operatorname{argmax} p(\mathbf{h}_L | \hat{\mathbf{x}})$$

$$\mathbf{x}^* = \operatorname{argmax} p(\mathbf{x} | \mathbf{h}_L^*)$$



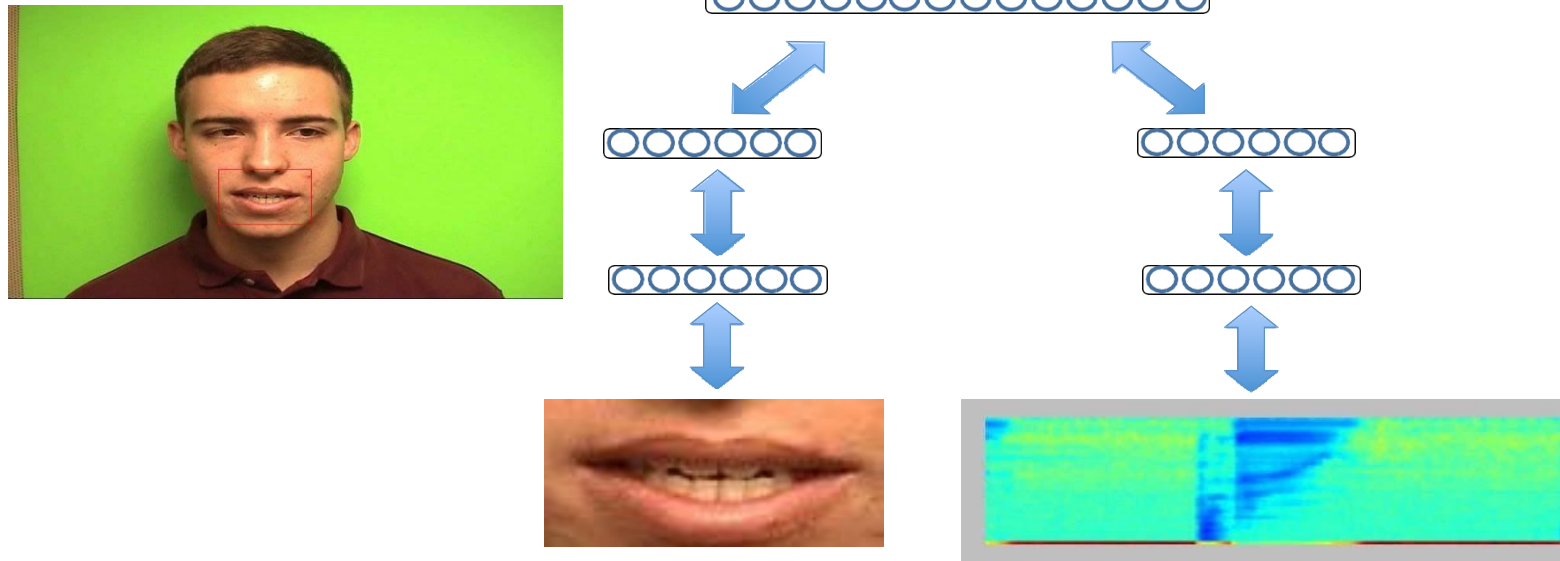
Occluded pattern ($\hat{\mathbf{x}}$)

Recovered pattern (\mathbf{x}^*)

Original pattern (\mathbf{x})

Multimodal DBM for Video and Audio

Cuave Dataset



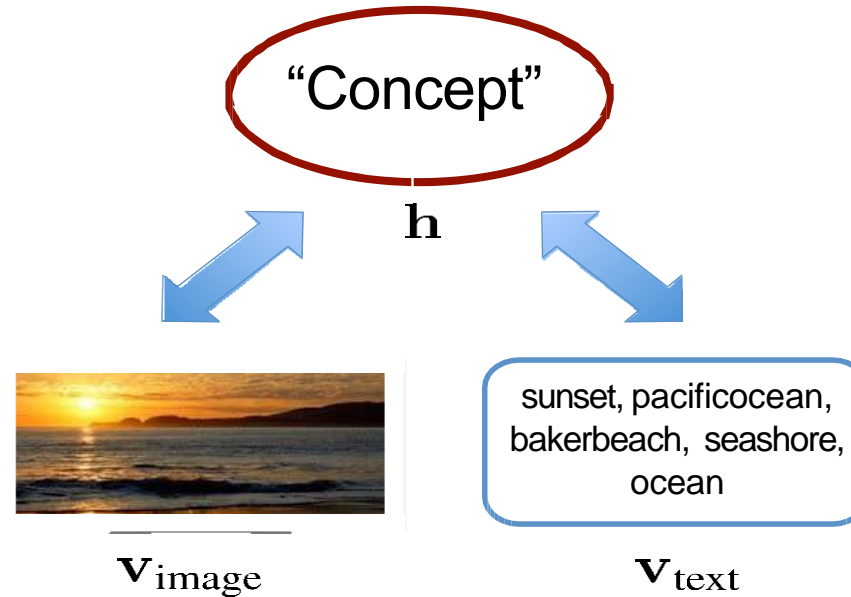
Multimodal DBM for Classification

- Learn a joint density model:

$$P(\mathbf{h}, \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}}).$$

- \mathbf{h} : “fused” representation for classification, retrieval.

$$P(\mathbf{h} | \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$$

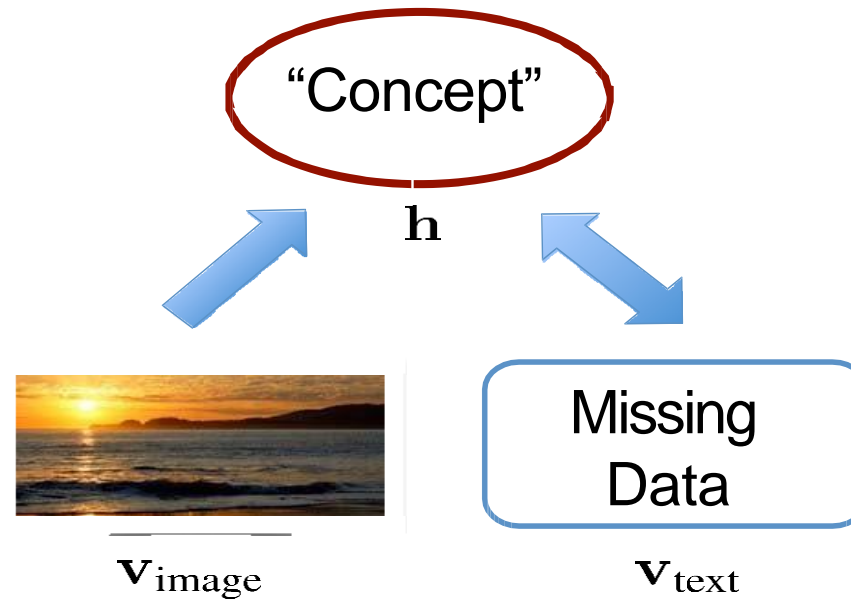


Multimodal DBM for Text Generation

Given the multimodal DBM,
infer the missing modality
given one modality, i.e.,

$$\mathbf{v}_{text}^* = \underset{\mathbf{v}_{text}}{\operatorname{argmax}} p(\mathbf{v}_{text} \mid \mathbf{v}_{image})$$

for image annotation



Text Generated from Images

Given



Generated

dog, cat, pet, kitten,
puppy, ginger, tongue,
kitty, dogs, furry



sea, france, boat, mer,
beach, river, bretagne,
plage, brittany



portrait, child, kid,
ritratto, kids, children,
boy, cute, boys, italy

Given



Generated

insect, butterfly, insects,
bug, butterflies,
lepidoptera



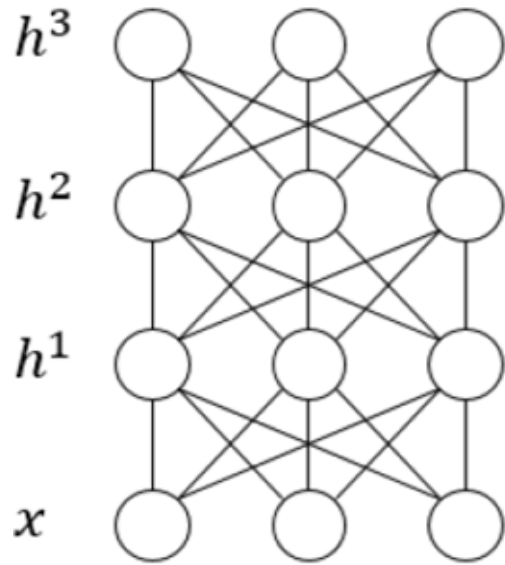
graffiti, streetart, stencil,
sticker, urbanart, graff,
sanfrancisco



canada, nature,
sunrise, ontario, fog,
mist, bc, morning

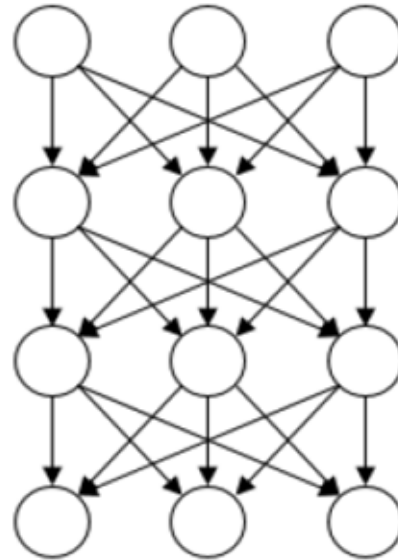
Different Types of Deep PGMs

Depending on the types of the building block, they can be divided into undirected deep PGMs, directed deep PGMs, and hybrid deep PGMs.



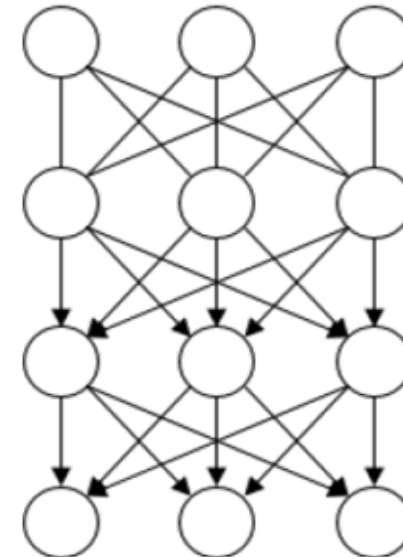
(a)

Deep Boltzmann
Machine (undirected)



(b)

Deep Regression Bayesian
Network (directed)



(c)

Deep Belief Network (hybrid)

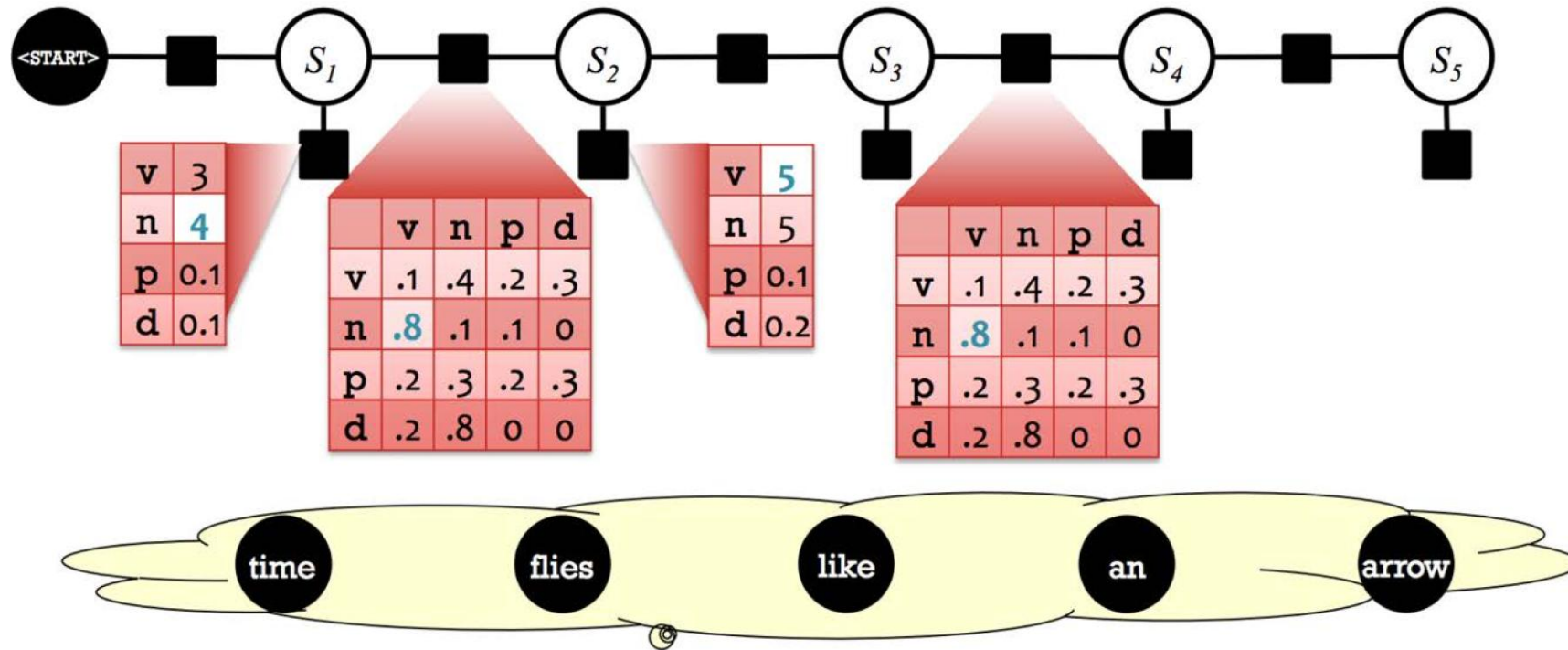
Graphical models vs. Deep networks

- ❑ DL & GM: the fields are similar in the beginning (structure, energy, etc.), and then diverge to their own signature pipelines
- ❑ DL: most effort is directed to comparing different architectures and their components (models are driven by evaluating empirical performance on a downstream tasks)
 - ❑ DL models are good at learning robust hierarchical representations from the data and suitable for simple reasoning (call it “low-level cognition”)
- ❑ GM: the effort is directed towards improving inference accuracy and convergence speed
 - ❑ GMs are best for provably correct inference and suitable for high-level complex reasoning tasks (call it “high-level cognition”)
- ❑ Convergence of both fields is very promising!

Deep Probabilistic Graphical Models

- Introduction of Probabilistic Graphical Models
 - Role of probability in AI
 - Graphical models vs Deep nets
- Deep probabilistic Graphical Models
- **Combining Deep Models with PGMs**

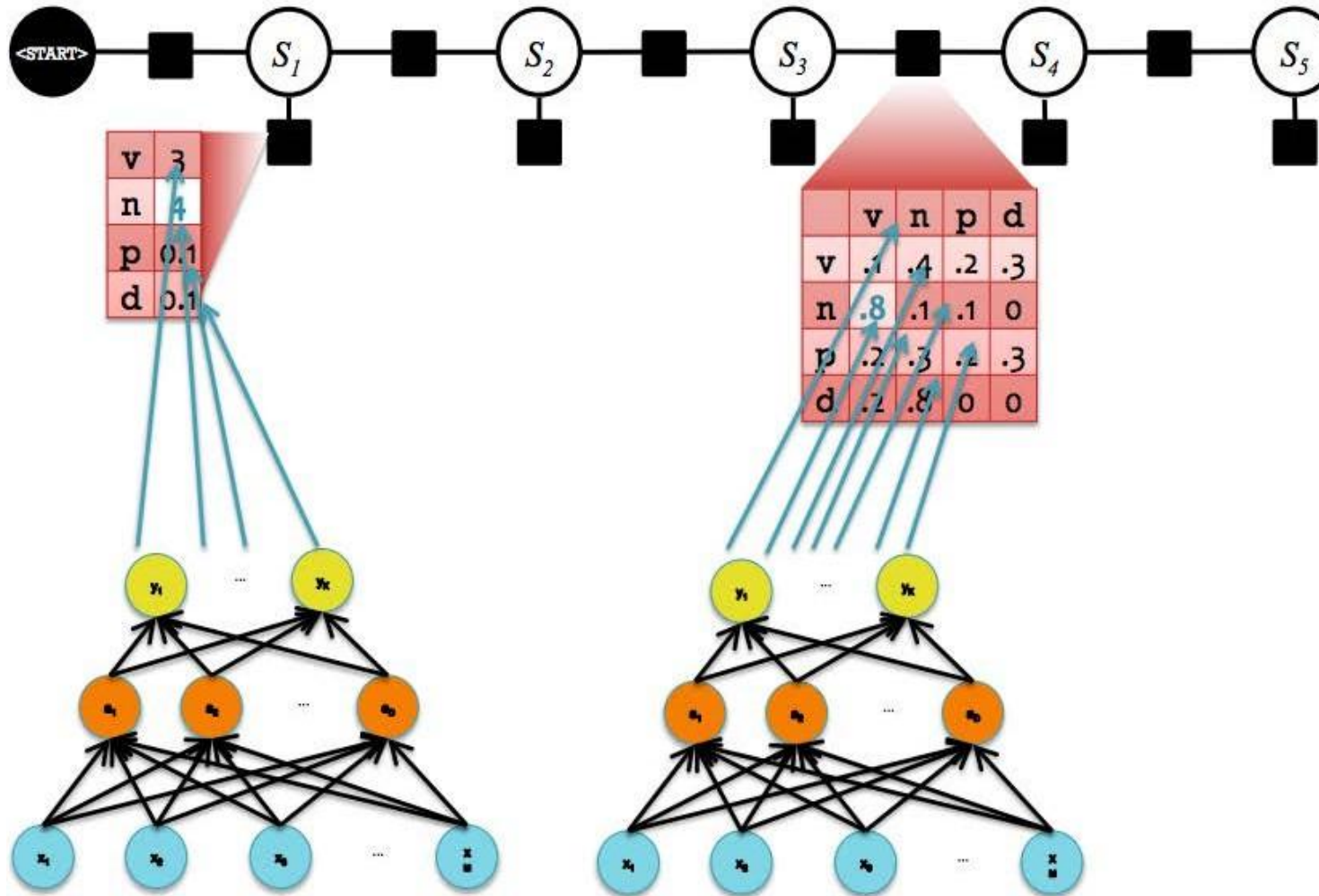
Hybrid NNs + conditional GMs



- In a standard CRF, each of the factor cells is a parameter.
- In a hybrid model, these values are computed by a neural network.

Hybrid: Neural Net + CRF

Forward computation



Using GMs as Prediction Explanations

Satellite imagery



Meaningful attributes

- | | |
|-------------------------|-------------------------|
| 01 Nightlight intensity | 09 Avg. vegetation dec. |
| 02 Is urban | 10 Avg. dist. to market |
| 03 Has electricity | 11 Avg. dist. to road |
| 04 Has generator | 12 Num. of rooms |
| 05 Avg. temperature | 13 Dist. to water src. |
| 06 Avg. percipitation | 14 Water usage p/ day |
| 07 Vegetation | 15 Is water payed |
| 08 Avg. vegetation inc. | 16 HH type: BQ |

Satellite imagery

Area attributes

Using GMs as Predictic

anations

How do we build a powerful predictive model whose predictions we can interpret in terms of semantically meaningful features?

01 Nightlight intensity

09 Avg. vegetation dec.

02 Is urban

10 Avg. dist. to market

03 Has electricity

11 Avg. dist. to road

04 Has generator

12 Num. of rooms

05 Avg. temperature

13 Dist. to water src.

06 Avg. percipitation

14 Water usage p/ day

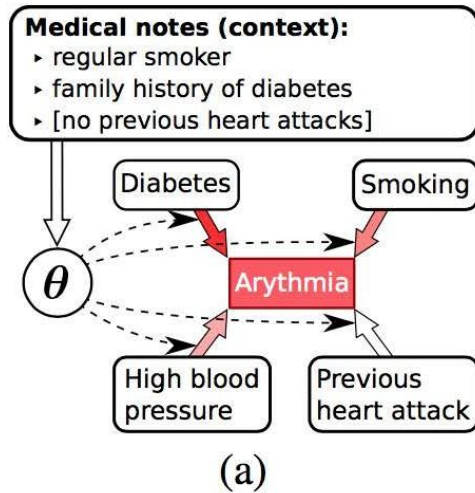
07 Vegetation

15 Is water payed

08 Avg. vegetation inc.

16 HH type: BQ

Contextual Explanation Networks (CENs)



- General idea: Use deep neural nets to generate parameters for graphical models applicable in a given context (e.g., for a given patient).
- Produced GMs are used to make the final prediction \Rightarrow 100% fidelity and consistency.
- GMs are built on top of semantically meaningful variables (not deep embeddings!) and can be used as explanations for each prediction.

Summary and Conclusions

- Probabilistic deep learning models include
 - Probabilistic deep neural networks
 - Generative PDNNs
 - Bayesian deep neural networks
 - Deep probabilistic graphical model

Summary and Conclusions

- **Advantages:**
 - Probabilistic deep learning has the abilities to capture data, model, and output uncertainties.
 - Probabilistic deep learning can quantify both aleatoric and epistemic prediction uncertainty
 - Bayesian deep learning can avoid the learning process and performs prediction using all parameters, hence avoiding overfitting problem.
 - Deep probabilistic graphical models fully capture the input, model, and output uncertainties.
- **Challenges:**
 - Probabilistic deep learning suffers from intractable inference problems. It hence cannot scale up well.
 - MCMC sampling and variational methods are currently two most popular methods to address the inference complexities. These techniques require further developments.
- **Future**
 - Probabilistic deep learning still has many theoretical challenges to overcome before it can be applied in large scale as deep neural networks.
 - It has great potential for many applications, it represents a frontier in AI research, and its success could lead to another AI revolution.