10417-617
Deep Learning: Fall 2020

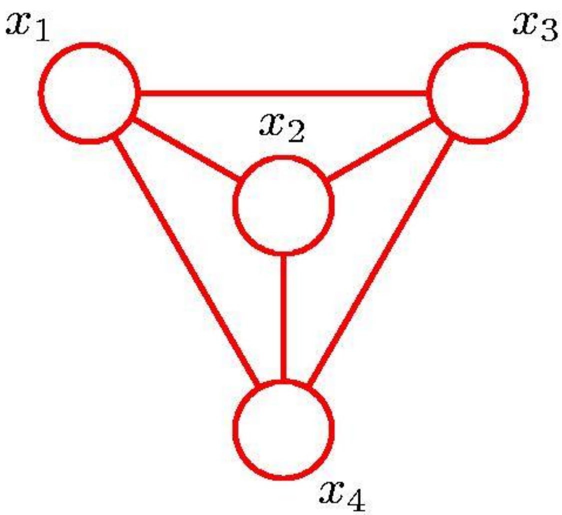
Andrej Risteski

Machine Learning Department

Lecture 13:
Variational methods, applications to learning latent-variable directed models

Graphical Models

Recall: graph contains a set of nodes connected by edges.



In a probabilistic graphical model, each node represents a random variable, links represent "probabilistic dependencies" between random variables.

Graph specifies how joint distribution over all random variables decomposes into a **product** of factors, each factor depending on a subset of the variables.

Two types of graphical models:

- **Bayesian networks**, also known as Directed Graphical Models (the links have a particular directionality indicated by the arrows)
- **Markov Random Fields**, also known as Undirected Graphical Models (the links do not carry arrows and have no directional significance).

Algorithmic pros/cons of latent-variable models (so far)

RBM's

- Hard to draw samples ✗

(In fact, #P-hard provably, even in Ising models)

- Easy to sample posterior distribution over latents ✓

Directed models

- Easy to draw samples ✓

- Hard to sample posterior distribution over latents ✗

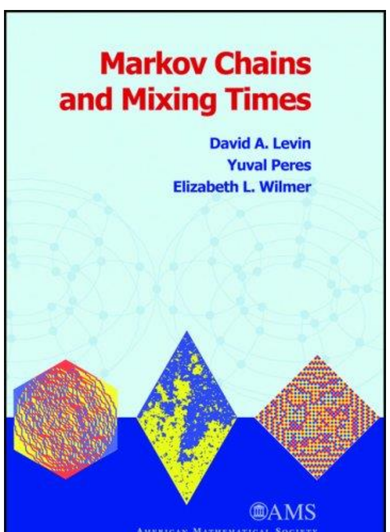
(In fact, #P-hard even in mixtures)

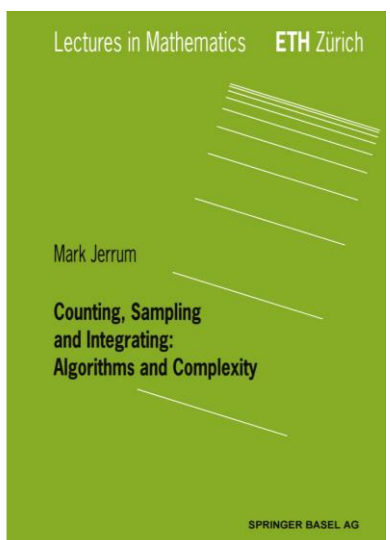
Algorithmic approaches

When faced with a difficult to calculate probabilistic quantity (partition function, difficult posterior), there are two families of approaches:

MARKOV CHAIN MONTE CARLO

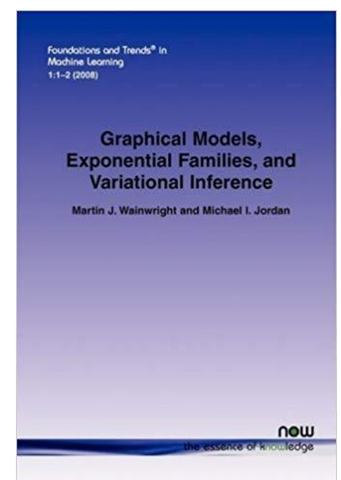
- Random walk w/ equilibrium distribution the one we are trying to sample from.





VARIATIONAL METHODS

- Based on solving an optimization problem.



Part I: approximating posteriors via variational methods

Sampling posteriors in latent-variable directed models

Recall, sampling from the posterior distribution P(z|x) is **hard**:

$$P(\text{Diseases}, \text{Symptoms}) = P(\text{Diseases}) \;\; P(\text{Symptoms}|\text{Diseases})$$

Latent

Data

Simple, explicit

By Bayes rule, $P(\text{Diseases}|\text{Symptoms}) \propto P(\text{Diseases}, \text{Symptoms})$

Up to normalizing const, simple…

Complicated partition function:

$$\sum_{\text{Diseases}} P(\text{Diseases}, \text{Symptoms})$$

Again, can be #P-hard to sample from!!

Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x):\text{distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

$$-H\big(q\,(z|x)\big) \quad - \quad \mathbb{E}_{z\sim q}[\log\ p(z,x)]$$

In fact, for every $q(z|x)$, we have

$$\log p(x) \ = -\big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[\log\ p(z,x)]\big) + KL(q\,(z|x)||\,p(z|x)\big)$$

Variational methods for partition functions

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x):\text{distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

In fact, for every $q(z|x)$, we have

$$\log p(x) = KL(q\,(z|x) || \, p(z|x)) - \left(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[\log \; p(z,x)]\right)$$

Why: $$0 \leq KL(q\,(z|x) || \, p(z|x)) = \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} p(z|x)$$

$$= -H\big(q\,(z|x)\big) - \mathbb{E}_{q\,(z|x)} \log \frac{p(z,x)}{p(x)}$$

$$= -H\big(q\,(z|x)\big) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x) + \log p(x)$$

Equality is attained if and only if $KL(q\,(z|x) || \, p(z|x))$=0 i. e. $q\,(z|x) = p(z|x)$

Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x):\text{distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

$$\log p(x) \;= -\big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[\log\; p(z,x)]\big) + KL(q\,(z|x)||\,p(z|x))$$

Why is this useful?

(1) Instead of finding the argmax over **all** distributions over Z, we can maximize over some **simpler** parametric family Q, i.e. we can solve

$$\max_{q(z|x)\in Q} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)$$

The argmax of the above distribution solves $\min_{q(Z|X)\in Q} KL(q\,(z|x)||\,p(z|x))$.

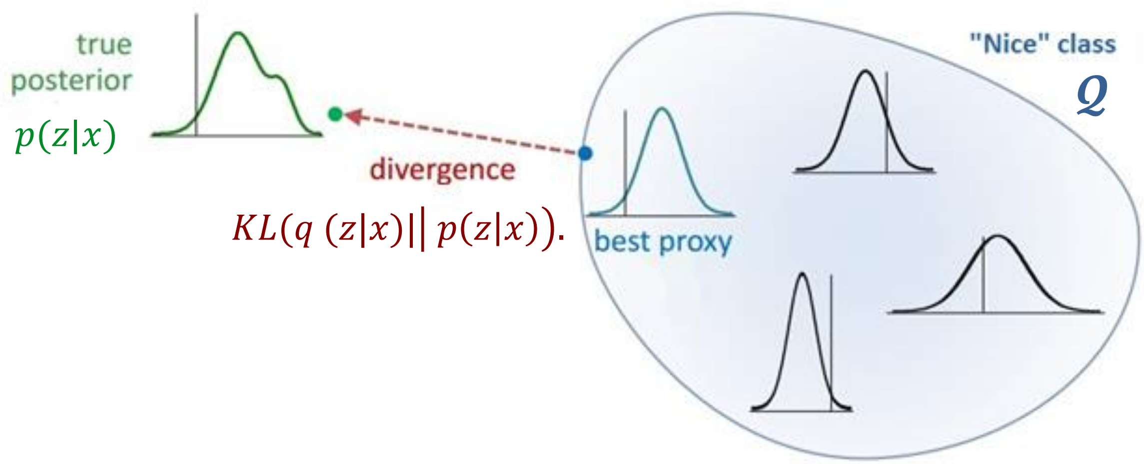
In other words, we are finding the **projection** of $p(z|x)$ onto Q.

Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x)\text{:distribution over } Z}{\operatorname{argmax}} \; \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

$$\log p(x) \quad = -\Big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[\log\; p(z,x)]\Big) + KL\big(q\,(z|x)\big|\big|\, p(z|x)\big)$$



Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z, x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x):\text{distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z, x)\}$$

$$\log p(x) \;\; = -\big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[p(z,x)]\big) + KL\big(q\,(z|x)\big|\big|\, p(z|x)\big)$$

Why is this useful?

(1) Instead of finding the argmax over *all* distributions over Z, we can maximize over some **simpler** parametric family Q, i.e. we can solve

$$\max_{q(z|x)\in Q} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z, x)\}$$

There are several common families Q that are used for which the above optimization is solveable – we will see **mean-field** family today, **neural-net** parametrized families when we study variational autoencoders.

Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x)\text{:distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

$$\log p(x) \;\; = -\big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[p(z,x)]\big) + KL\big(q\,(z|x) \big\|\, p(z|x)\big)$$

Why is this useful?

(2) Provides a lower bound on $\log p(x)$ -- sometimes called the **ELBO (evidence lower bound)**, since

$$\log p(x) \geq \max_{q(z|x)\in Q} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)$$

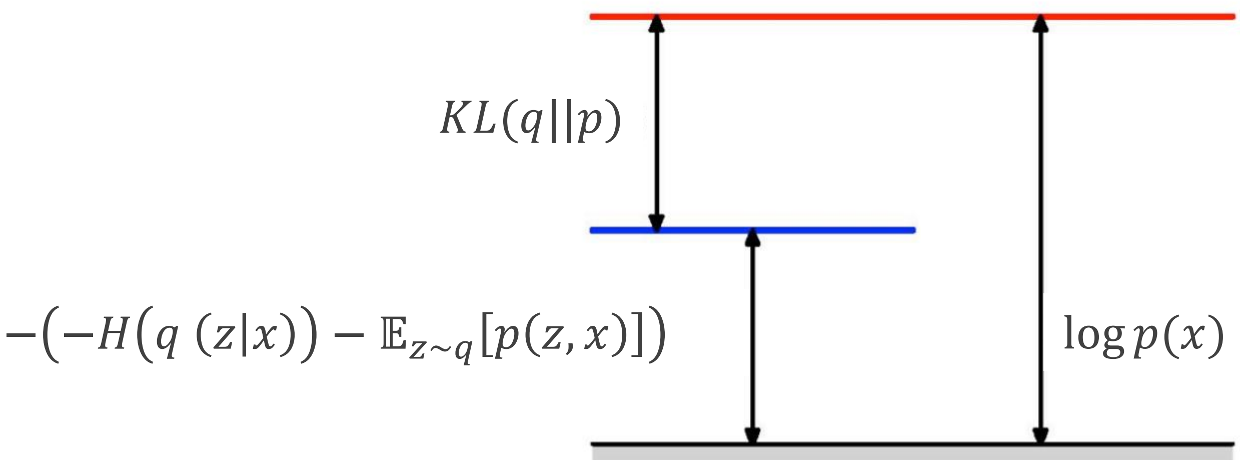
This will be useful when learning latent-variable directed models (stay tuned !).

Variational methods for approximating posteriors

Gibbs variational principle: Let $p(z,x)$ be a joint distribution over latent variables and observables. Then,

$$p(z|x) = \underset{q(z|x)\text{:distribution over } Z}{\operatorname{argmax}} \mathbb{E}_{q\,(z|x)} \log q\,(z|x) - \mathbb{E}_{q\,(z|x)} \log p\,(z,x)\}$$

$$\log p(x) \;\; = -\big(-H\big(q\,(z|x)\big) - \mathbb{E}_{z\sim q}[p(z,x)]\big) + KL\big(q\,(z|x)\big|\big|\, p(z|x)\big)$$



Solving the mean-field relaxation: coordinate ascent

Inspiration from physics: consider the case where Q contains product distributions, that is, for every $q(\cdot\,|x) \in Q$:

$$q(z|x) = \Pi_{i=1}^{d} q_i(z_i|x).$$

Consider updating a **single** coordinate of the mean-field distribution, that is keep $q_{-i}\ (z_i|x)$ fixed, and optimize for $q_i\ (z_i|x)$. We have:

$$KL(q\ (z|x) || \, p(z|x)) \ = \mathbb{E}_{q\ (z|x)} \log q\ (z|x) - \ \mathbb{E}_{q\ (z|x)} \log p\ (z, x)$$

$$= \sum_i \ \mathbb{E}_{q_i\ (z_i|x)} \log q_i\ (z_i|x) - \mathbb{E}_{q_i\ (z_i|x)} \left[\mathbb{E}_{q_{-i}(z_{-i}|x)} \log p\ (z_i, z_{-i}, x)\right]$$

$$= \mathbb{E}_{q_i\ (z_i|x)} \log q_i\ (z_i|x) - \ \mathbb{E}_{q_i\ (z_i|x)}[\log \tilde{p}\ (z_i, x)] + C$$

Renormalize to make it a distribution

Solving the mean-field relaxation: coordinate ascent

Inspiration from physics: consider the case where Q contains product distributions, that is, for every $q(\cdot\,|x) \in Q$:

$$q(z|x) = \Pi_{i=1}^{d} q_i(z_i|x).$$

Consider updating a **single** coordinate of the mean-field distribution, that is keep $q_{-i}\,(z_i|x)$ fixed, and optimize for $q_i\,(z_i|x)$. We have:

$$KL(q\,(z|x)||\,p(z|x)) \;= \mathbb{E}_{q_i\,(z_i|x)} \log q_i\,(z_i|x) - \;\mathbb{E}_{q_i\,(z_i|x)}[\log \tilde{p}\,(z_i, x)] + C$$

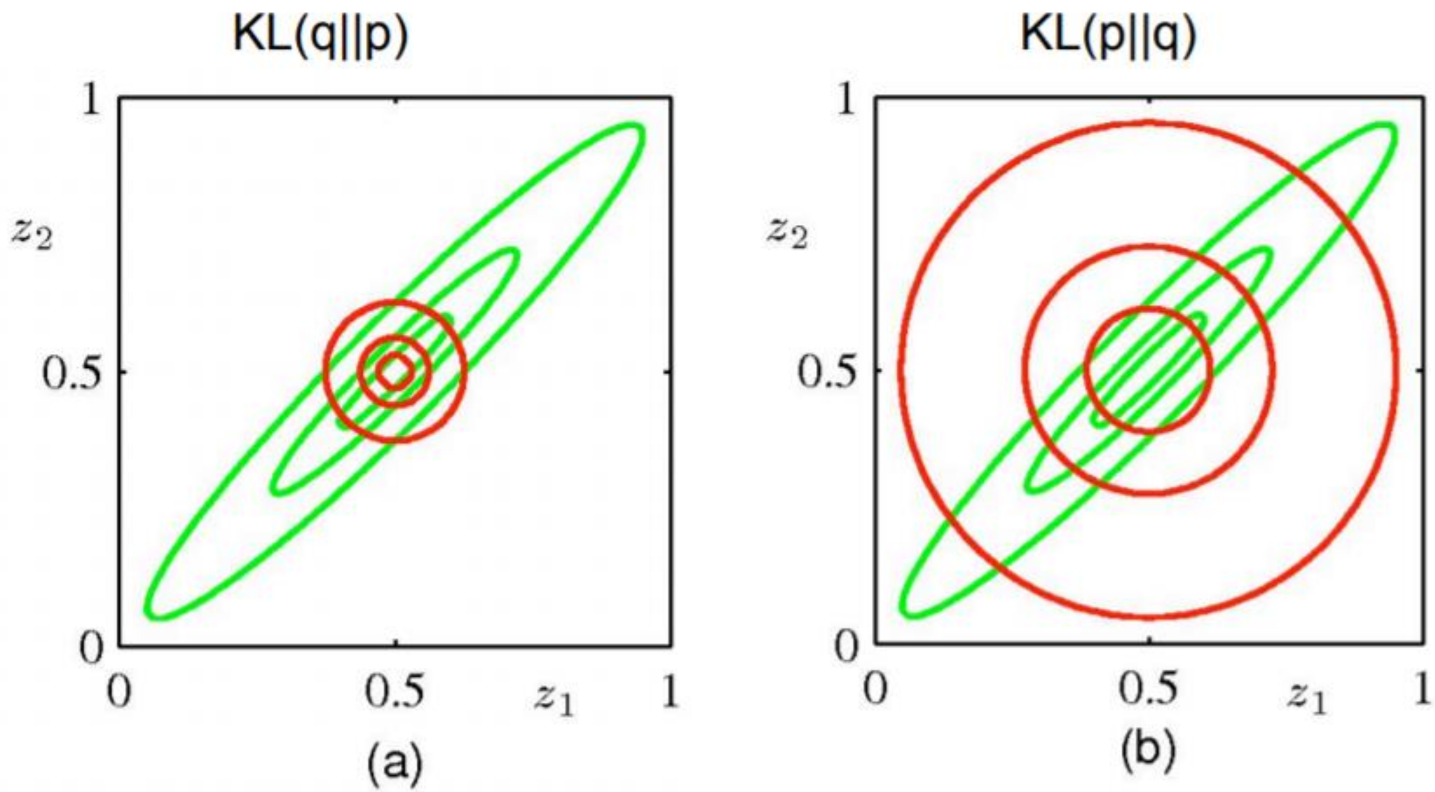
$$= KL(q_i\,(z_i|x)||\,\tilde{p}(z_i, x)) + C$$

Optimum is $q_i\,(z_i|x) = \tilde{p}(z_i, x)$

$$= \frac{\mathbb{E}_{q_{-i}(z_{-i}|x)} \log p\,(z_i, z_{-i}, x)}{\int_{z_i} \mathbb{E}_{q_{-i}(z_{-i}|x)} \log p\,(z_i, z_{-i}, x)}$$

Coordinate ascent: iterate above updates!

What if we changed the order of p, q in KL divergence?



Approximation is too compact.

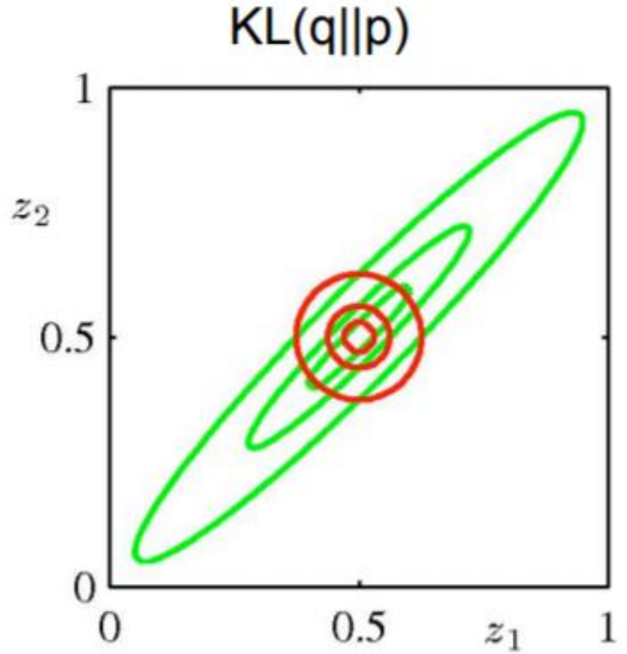
Approximation is too spread.

What if we changed the order of p, q in KL divergence?

$$\mathrm{KL}(q||p) = -\int q(\mathbf{Z}) \ln \frac{p(\mathbf{Z})}{q(\mathbf{Z})} \mathrm{d}\mathbf{Z}.$$

There is a large positive contribution to the KL divergence from regions of Z space in which:
- p(Z) is near zero
- unless q(Z) is also close to zero.

Minimizing KL(q||p) leads to distributions q(Z) that avoid regions in which p(Z) is small.



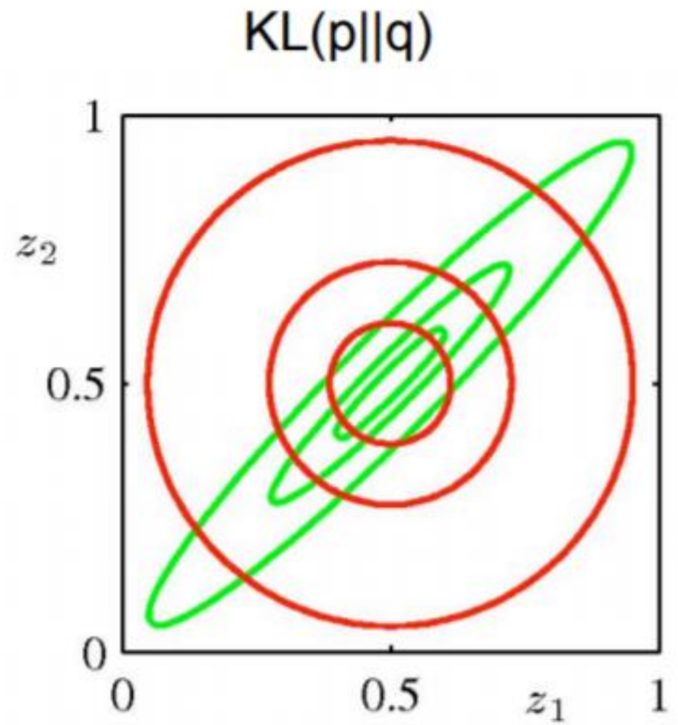
(a)

What if we changed the order of p, q in KL divergence?

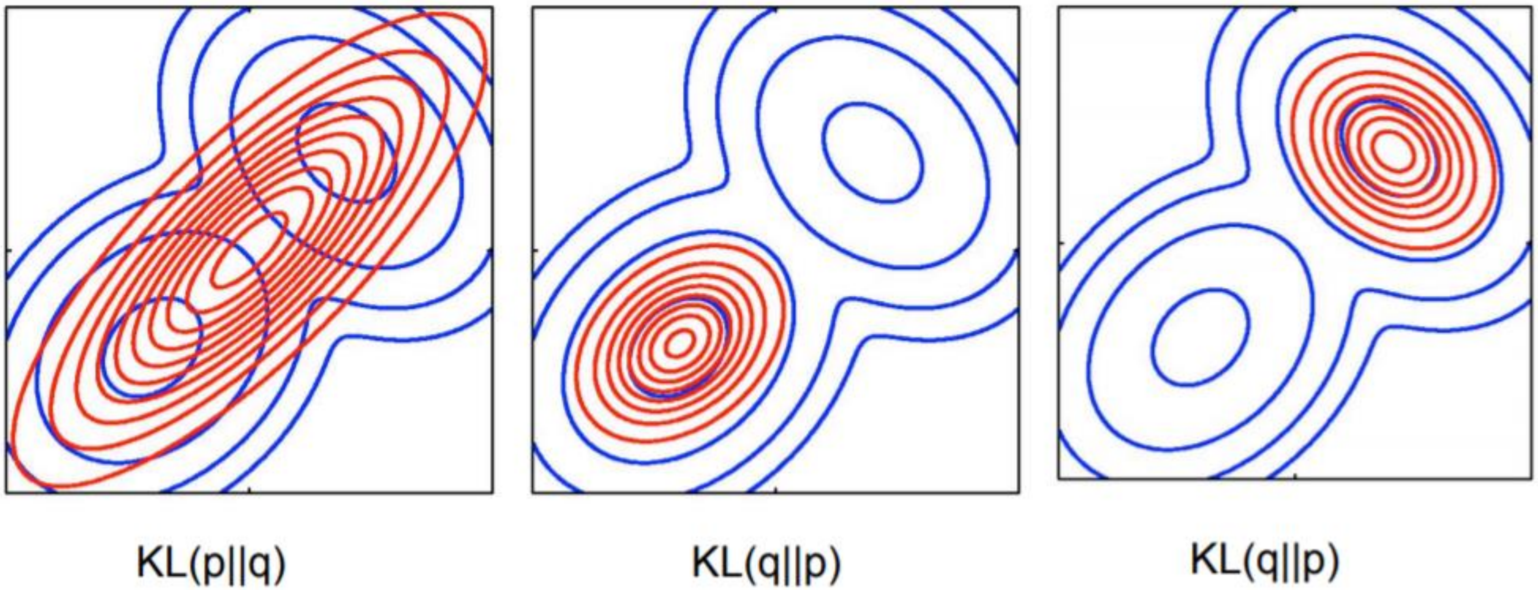
$$\mathrm{KL}(p||q) = -\int p(\mathbf{Z}) \ln \frac{q(\mathbf{Z})}{p(\mathbf{Z})} \mathrm{d}\mathbf{Z}.$$

There is a large positive contribution to the KL divergence from regions of Z space in which:
- q(Z) is near zero,
- unless p(Z) is also close to zero.

Minimizing KL(p||q) leads to distributions q(Z) that are nonzero in regions where p(Z) is nonzero.



What happens when posterior class is not rich enough?



Blue contours show bimodal distribution, red contours single Gaussian distribution that best approximates it.

KL(q||p) will tend to find a single mode, whereas KL(p||q) will average across all of the modes.

Part II: learning latent-variable directed models

Learning latent-variable directed graphical models

How should we try to learn the parameters of a graphical model?

The most obvious strategy: maximum likelihood estimation

Given data $x_1, x_2, \dots, x_n$, solve the optimization problem

$$\max_{\theta \in \Theta} \sum_{i=1}^{n} \log p(x_i)$$

Latent variables: we will use the Gibbs variational principle again!

$$\log_\theta \; p(x) = \max_{q(z|x):\text{ distribution over } \mathcal{Z}} H(q(z|x)) + \mathbb{E}_{q(z|x)}[\log p_\theta(x,z)]$$

Hence, MLE objective can be written as double maximization:

$$\max_{\theta \in \Theta} \max_{\{q_i(z|x_i)\}} \sum_{i=1}^{n} H(q_i(z|x_i)) + \mathbb{E}_{q_i(z|x_i)}[\log p_\theta(x_i,z)]$$

Expectation-maximization/ variational inference

The canonical algorithm for learning a single-layer latent-variable Bayesian network is an iterative algorithm as follows.

Consider the max-likelihood objective, rewritten as in the previous slide:

$$\max_{\theta\in\Theta} \max_{\{q_i(z|x_i)\in\mathcal{Q}\}} \sum_{i=1}^{n} H(q_i(z|x_i)) + \mathbb{E}_{q_i(z|x_i)}[\log p_\theta(x_i, z)]$$

Algorithm maintains iterates $\theta^t, \{q_i^t(z|x_i)\}$, and updates them iteratively

(1) **Expectation (E)-step**:

Keep θ^t fixed, set $\{q_i^{t+1}(z|x_i) \in \mathcal{Q}\}$, s.t. they maximize the objective above.

(2) **Maximization (M)-step**:

Keep $\{q_i^t(z|x_i)\}$ fixed, set θ^{t+1} s.t. it maximizes the objective above.

Clearly, every step cannot make the objective worse!

Does *not* mean it converges to global optimum – could, e.g. get stuck in a local minimum.

Expectation-maximization/ variational inference

The canonical algorithm for learning a single-layer latent-variable Bayesian network is an iterative algorithm as follows.

Consider the max-likelihood objective, rewritten as in the previous slide:
$$\max_{\theta\in\Theta} \max_{q_i(z|x_i)} \sum_{i=1}^{n} H(q_i(z|x_i)) + \mathbb{E}_{q_i(z|x_i)}[\log p_\theta(x_i, z)]$$

Algorithm maintains iterates $\theta^t, q_i^t(z|x_i)$, and updates them iteratively

(1) **Expectation step**:

Keep θ^t and set $q_i^{t+1}(z|x_i)$, s.t. they maximize the objective above.

If the class is infinitely rich, the optimum is $q_i^{t+1}(z|x_i) = p_{\theta^t}(z|x_i)$

This is called **expectation-maximization (EM)**.
If class is not infinitely rich, it's called **variational inference**.

Examples

1: Mixtures of spherical Gaussians

Consider a mixture of K Gaussians with unknown means $p = \sum_{i=1}^{K} \frac{1}{K} \mathcal{N}(\mu_i, I_d)$

Let's try to calculate the E and M steps.

E-step: the optimal $q_i^{t+1}(z|x_i)$ is $p_{\theta^t}(z|x_i)$. Can we calculate this?

By Bayes rule, $p_{\theta^t}(z = k|x_i) \propto p(x_i|z = k) \propto e^{-\left\|x_i - \mu_k^t\right\|^2}$

Writing out the normalizing constant, we have

$$p_{\theta^t}(z = k|x_i) = \frac{e^{-\left\|x_i - \mu_k^t\right\|^2}}{\sum_{k'} e^{-\left\|x_i - \mu_{k'}^t\right\|^2}}$$

"Soft" version of assigning point to nearest cluster

Examples

<u>1: Mixtures of spherical Gaussians</u>

Consider a mixture of K Gaussians with unknown means $p = \sum_{i=1}^{K} \frac{1}{K} \mathcal{N}(\mu_i, I_d)$

Let's try to calculate the E and M steps.

M-step: given a quess $q_i^t(z|x_i)$, we can rewrite the maximization for θ as:

$$\max_{\theta \in \Theta} \sum_{i=1}^{n} H\big(q_i^t(z|x_i)\big) + \mathbb{E}_{q_i^t(z|x_i)}[\log p_\theta(x_i, z)]$$

$$= \mathbb{E}_{q_i^t(z|x_i)} [\cancel{\log p(z)} + \log p_\theta(x|z)]$$

Doesn't depend on θ

$$\mathbb{E}_{q_i^t(z|x_i)}[\ \log p_\theta(x|z)]$$

Examples

1: Mixtures of spherical Gaussians

Consider a mixture of K Gaussians with unknown means $p = \sum_{i=1}^{K} \frac{1}{K} \mathcal{N}(\mu_i, I_d)$

Let's try to calculate the E and M steps.

M-step: given a quess $q_i^t(z|x_i)$, we can rewrite the maximization for θ as:

$$\max_{\theta} \; \mathbb{E}_{q_i^t(z|x_i)}[\, \log p_\theta(x|z)] = \max_{\theta} \; -\sum_{i=1}^{n} \sum_{k=1}^{K} q_i^t(z=k|x_i)||x_i - \mu_k\,||^2$$

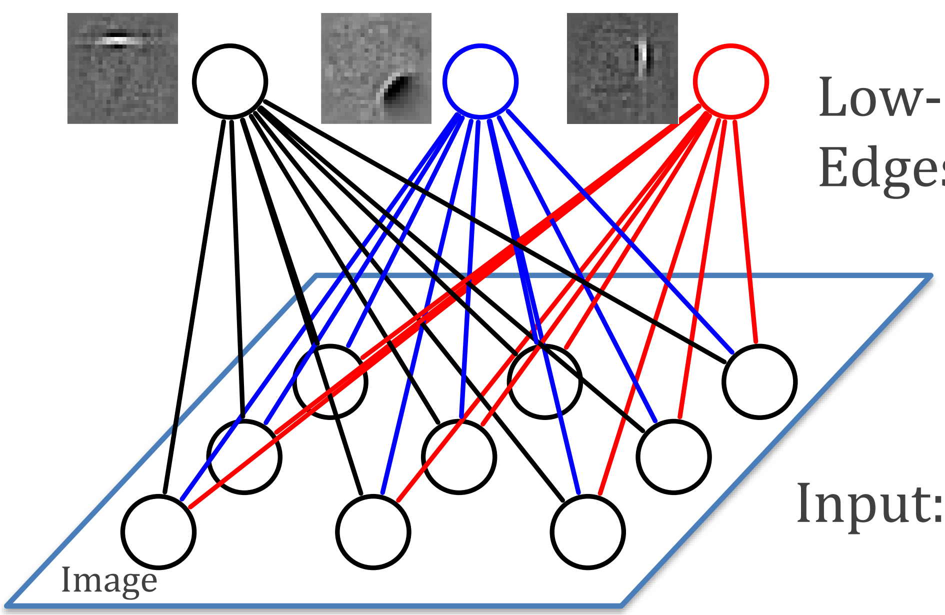
Setting the derivative wrt to μ_k to 0, we have:

$$\mu_k^{t+1} = \sum_{i=1}^{n} \frac{e^{-\left||x_i - \mu_k^t\right||^2}}{\sum_{k'} \; e^{-\left||x_i - \mu_{k'}^t\right||^2}} x_i$$

Average points, weighing nearby points more

Part III: Deep Belief Networks (DBNs)

Deep Belief Network



Low-level features:
Edges

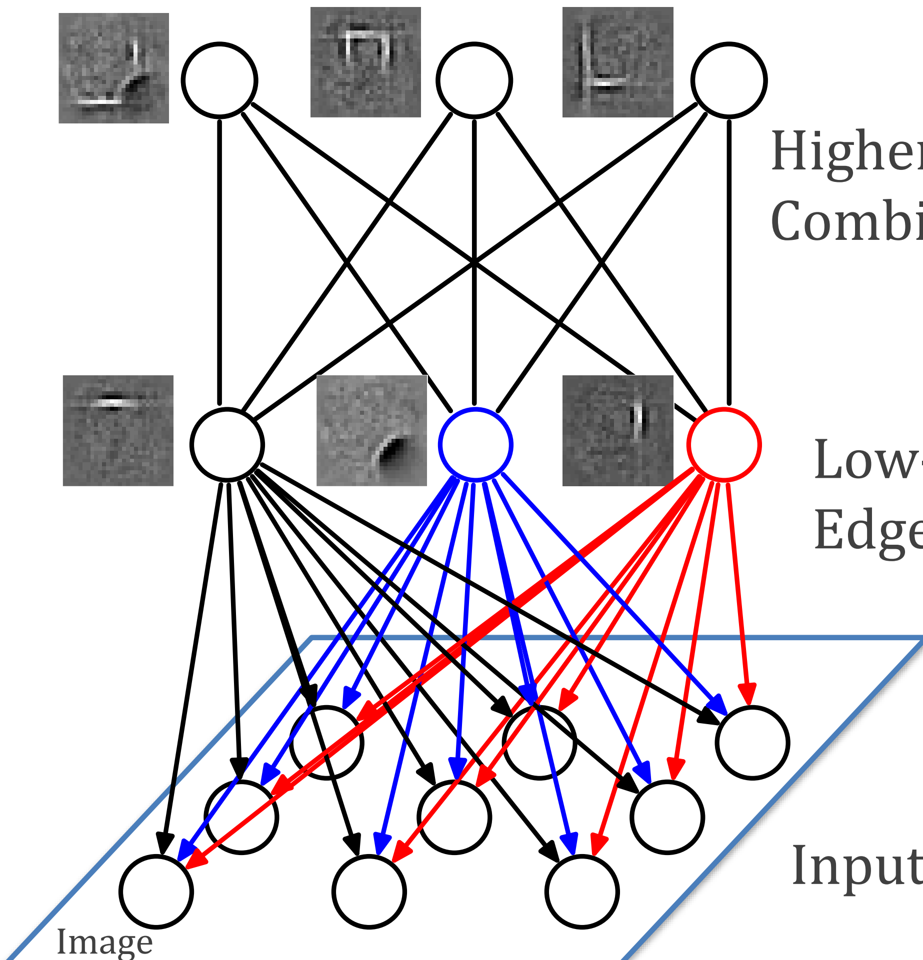
Built from **unlabeled** inputs.

Input: Pixels

(Hinton et.al. Neural Computation 2006)

Deep Belief Network

Internal representations capture higher-order statistical structure



Higher-level features:
Combination of edges

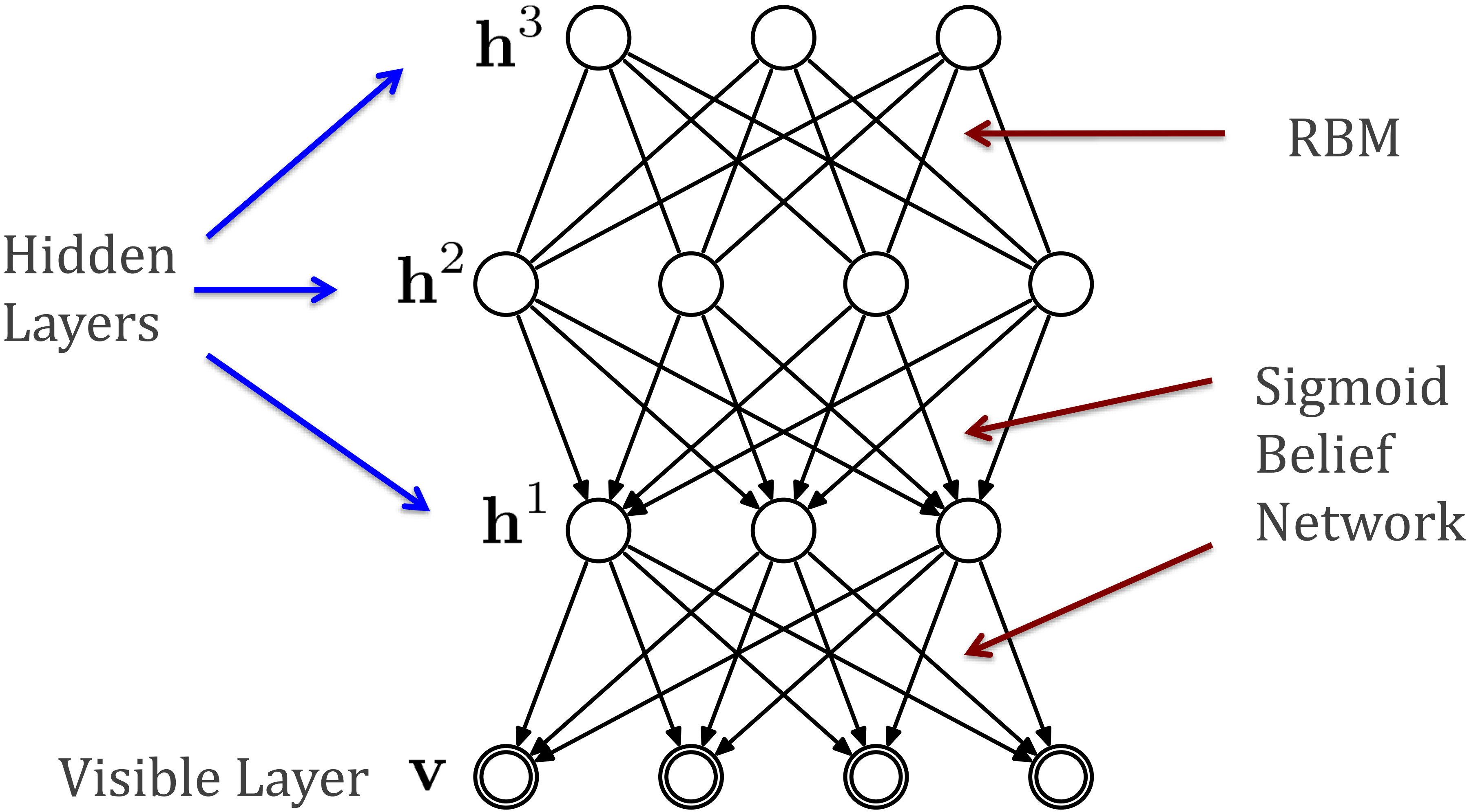
Low-level features:
Edges

Built from **unlabeled** inputs.

Input: Pixels

(Hinton et.al. Neural Computation 2006)

Deep Belief Network



Deep Belief Network

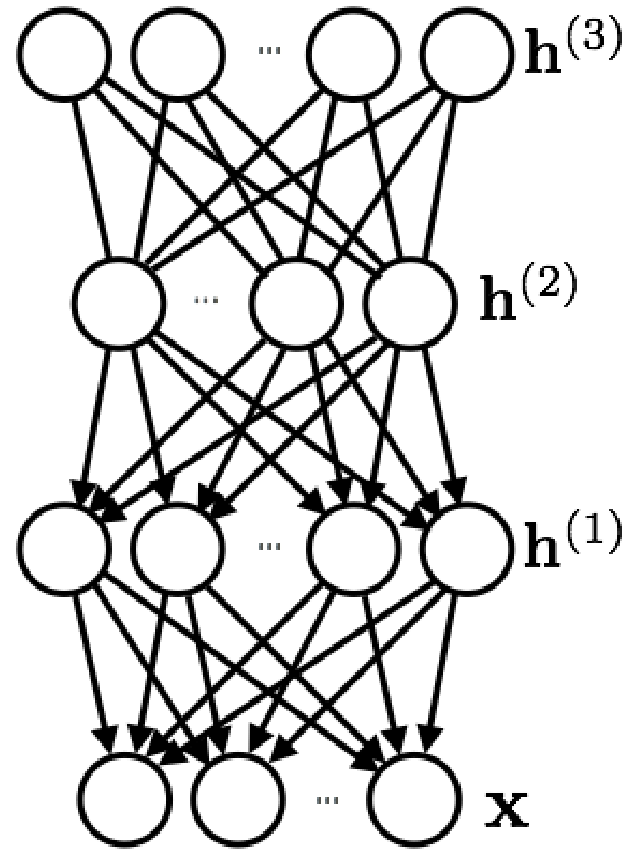
- it is a generative model that mixes undirected and directed connections between variables

- top 2 layers' distribution $p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$ is an RBM!

- other layers form a Bayesian network with conditional distributions:

$$p(h_j^{(1)} = 1|\mathbf{h}^{(2)}) = \text{sigm}(\mathbf{b}^{(1)} + \mathbf{W}^{(2)^\top}\mathbf{h}^{(2)})$$

$$p(x_i = 1|\mathbf{h}^{(1)}) = \text{sigm}(\mathbf{b}^{(0)} + \mathbf{W}^{(1)^\top}\mathbf{h}^{(1)})$$



Deep Belief Network

The joint distribution of a DBN is as follows

$$p(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})\, p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)})\, p(\mathbf{x}|\mathbf{h}^{(1)})$$

where

$$p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \exp\left(\mathbf{h}^{(2)\top}\mathbf{W}^{(3)}\mathbf{h}^{(3)} + \mathbf{b}^{(2)\top}\mathbf{h}^{(2)} + \mathbf{b}^{(3)\top}\mathbf{h}^{(3)}\right)/Z$$

$$p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}) = \prod_j p(h_j^{(1)}|\mathbf{h}^{(2)})$$

$$p(\mathbf{x}|\mathbf{h}^{(1)}) = \prod_i p(x_i|\mathbf{h}^{(1)})$$

(I realize this looks odd.)

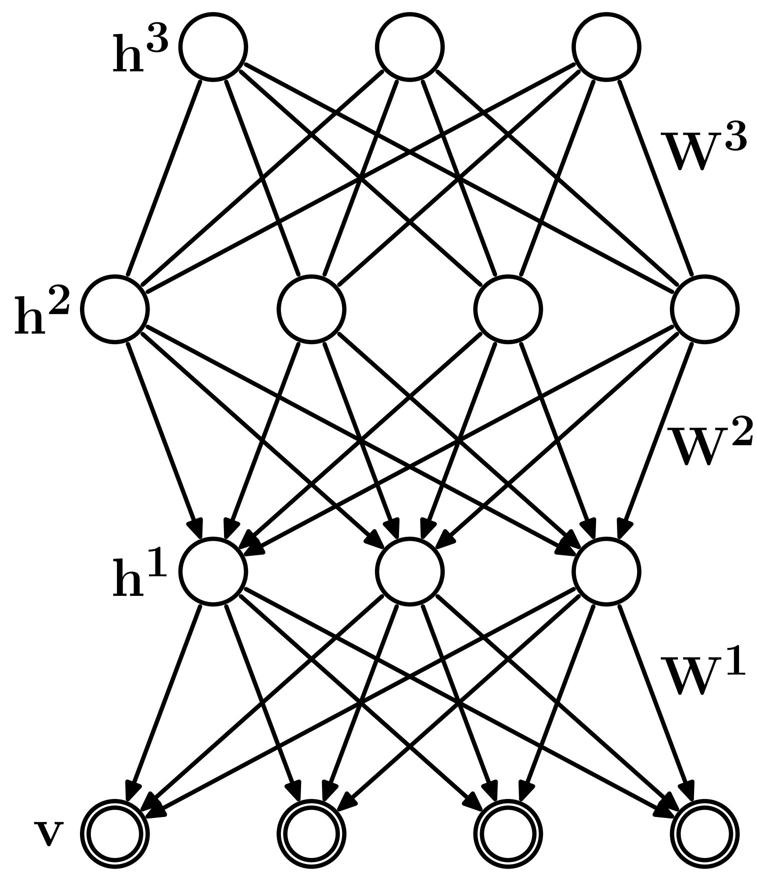
DBN Layer-wise Training

Approximate Inference

$Q(\mathbf{h}^3|\mathbf{h}^2)$

$Q(\mathbf{h}^2|\mathbf{h}^1)$

$Q(\mathbf{h}^1|\mathbf{v})$



Generative Process

$P(\mathbf{h}^2, \mathbf{h}^3)$

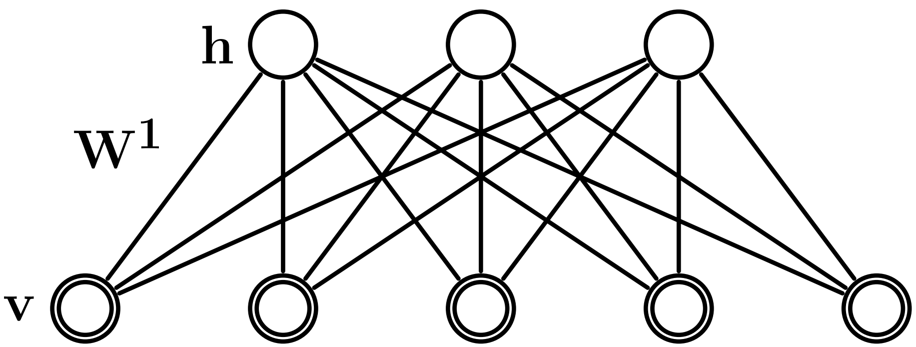
$P(\mathbf{h}^1|\mathbf{h}^2)$

$P(\mathbf{v}|\mathbf{h}^1)$

$$Q(\mathbf{h}^t|\mathbf{h}^{t-1}) = \prod_j \sigma\left(\sum_i W^t h_i^{t-1}\right) \qquad P(\mathbf{h}^{t-1}|\mathbf{h}^t) = \prod_j \sigma\left(\sum_i W^t h_i^t\right)$$

DBN Layer-wise Training

- Learn an RBM with an input layer v=x and a hidden layer h.

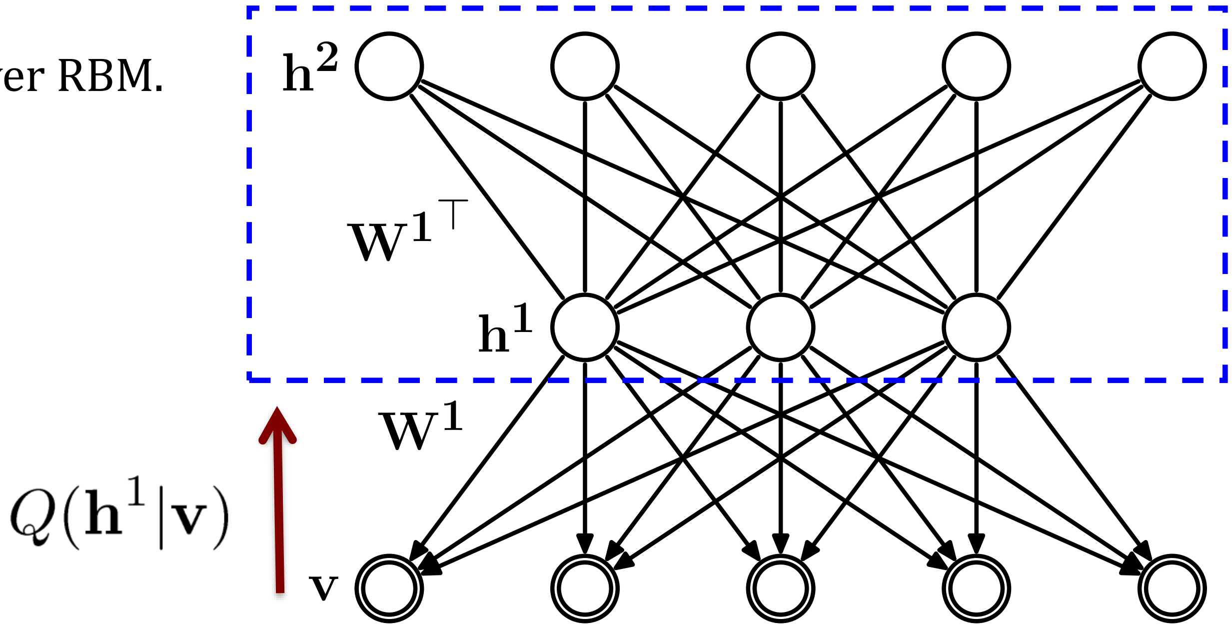


DBN Layer-wise Training

- Learn an RBM with an input layer v=x and a hidden layer h.

- Treat inferred values $Q(\mathbf{h}^1|\mathbf{v}) = P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-layer RBM.

- Learn and freeze 2nd layer RBM.



DBN Layer-wise Training

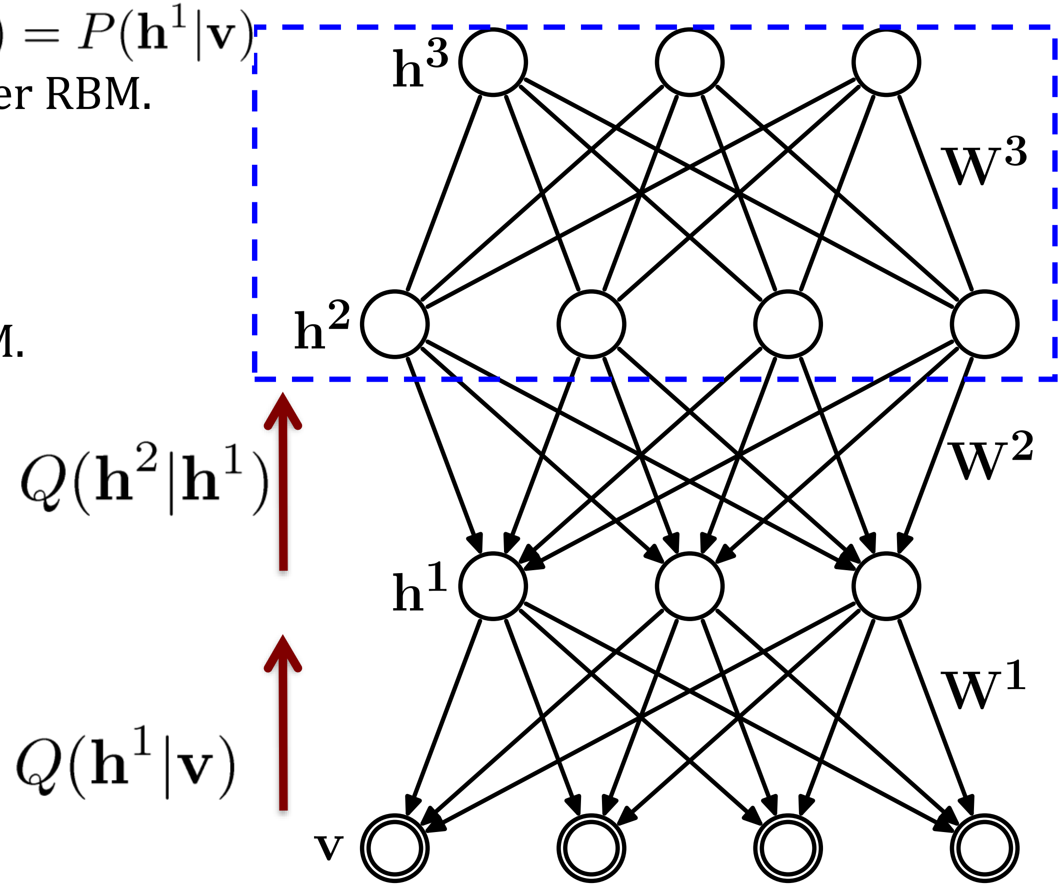
- Learn an RBM with an input layer v=x and a hidden layer h.

Unsupervised Feature Learning.

- Treat inferred values $Q(\mathbf{h}^1|\mathbf{v}) = P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-layer RBM.

- Learn and freeze 2nd layer RBM.

- Proceed to the next layer.

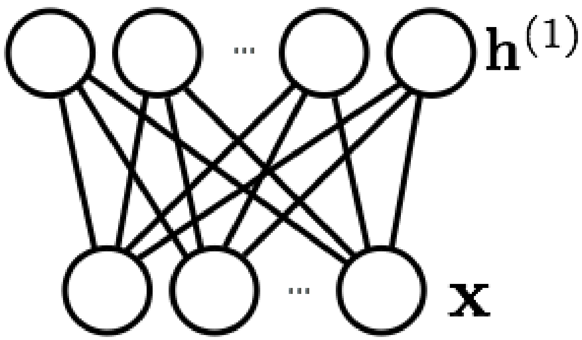


Where does this training come from??

Variational intuitions

Let's write the marginal p(x) in terms of the Gibbs variational principle.

As $p(x) = \sum_{h^{(1)}} p(x, h^{(1)})$ (i.e. the normalizing constant for $p\left(h^{(1)}\right) \propto p(x, h^{(1)})$), we have:



For every distribution $q\left(\mathbf{h}^{(1)} \middle| \mathbf{x}\right)$:

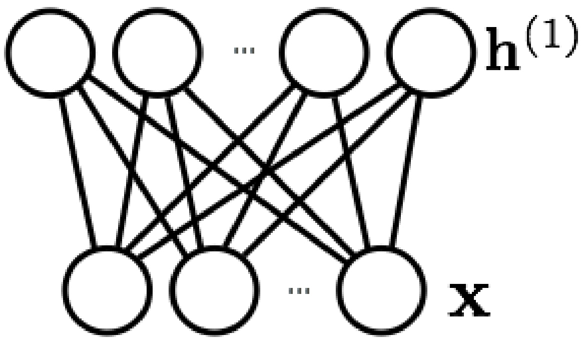
$$\log p(\mathbf{x}) \geq \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{h}^{(1)}) - \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log q(\mathbf{h}^{(1)}|\mathbf{x})$$

Equality is attained if $q\left(\mathbf{h}^{(1)} \middle| \mathbf{x}\right) = p\left(\mathbf{h}^{(1)} \middle| \mathbf{x}\right)$.

Variational intuitions

Let's write the marginal p(x) in terms of the Gibbs variational principle.

As $p(x) = \sum_{h^{(1)}} p(x, h^{(1)})$ (i.e. the normalizing constant for $p(h^{(1)}) \propto p(x, h^{(1)})$), we have:



For every distribution $q(\mathbf{h}^{(1)}|\mathbf{x})$:

$$\log p(\mathbf{x}) \geq \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{h}^{(1)}) - \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log q(\mathbf{h}^{(1)}|\mathbf{x})$$

*The idea will be to add layers, s.t. we improve the **variational bound on the RHS**.*

Variational intuitions

adding 2nd layer means
untying the parameters

$$\log p(\mathbf{x}) \quad \geq \quad \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \left(\log p(\mathbf{x}|\mathbf{h}^{(1)}) + \log p(\mathbf{h}^{(1)}) \right)$$

$$- \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log q(\mathbf{h}^{(1)}|\mathbf{x})$$

- When adding a second layer, we model $p(\mathbf{h}^{(1)})$ using a separate set of parameters
    - ➢ they are the parameters of the RBM involving $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$
    - ➢ $p(\mathbf{h}^{(1)})$ is now the marginalization of the second hidden layer

$$p(\mathbf{h}^{(1)}) = \sum_{\mathbf{h}^{(2)}} p(\mathbf{h}^{(1)}, \mathbf{h}^{(2)})$$

Variational intuitions

adding 2nd layer means
untying the parameters

$$\log p(\mathbf{x}) \geq \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \left(\log p(\mathbf{x}|\mathbf{h}^{(1)}) + \log p(\mathbf{h}^{(1)}) \right)$$

$$- \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log q(\mathbf{h}^{(1)}|\mathbf{x})$$

- we can train the parameters of … the bound. This is equivalent to m… . terms are constant:

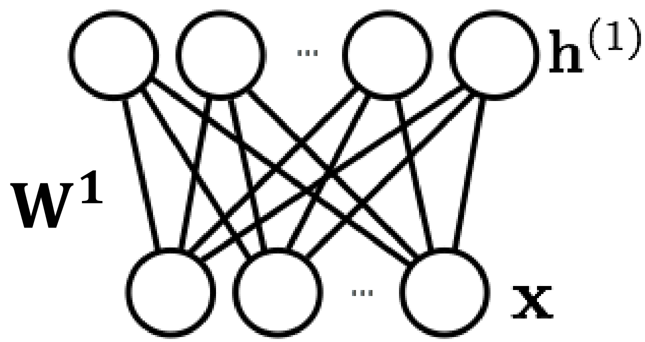
$$- \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log p(\mathbf{h}^{(1)})$$

- this is like training an RBM on data generated from $q(\mathbf{h}^{(1)}|\mathbf{x})$!

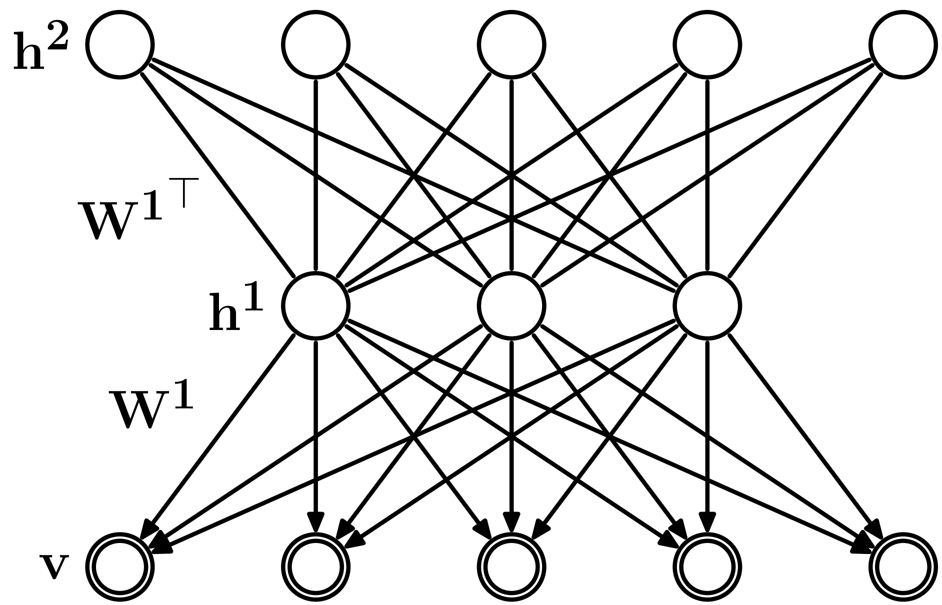
Layerwise training improves variational lower bound

Does the lower bound improve?

Observation: a two-layer DBN with appropriately tied weights is equivalent to an RBM:



$\Leftrightarrow$



Formal proof is a little annoying. Intuition:

- Gibbs sampling converges to model distribution in first case.
- Gibbs sampling on top two layers, plus one last sample of x given $\mathrm{h}^{(1)}$ converges to model distribution in second.
- The steps in these two random walks are *exactly* the same.

Does the lower bound improve?

adding 2nd layer means
untying the parameters

$$\log p(\mathbf{x}) \geq \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \left(\log p(\mathbf{x}|\mathbf{h}^{(1)}) + \log p(\mathbf{h}^{(1)}) \right)$$
$$- \sum_{\mathbf{h}^{(1)}} q(\mathbf{h}^{(1)}|\mathbf{x}) \log q(\mathbf{h}^{(1)}|\mathbf{x})$$

- for $q(\mathbf{h}^{(1)}|\mathbf{x})$ we use **the posterior of the first layer RBM**. This is equivalent to a feed-forward (sigmoidal) layer, followed by sampling
- by initializing the weights of the second layer RBM as the transpose of the first layer weights, **the bound is initially tight!** (As we showed, a 2-layer DBN with tied weights is equivalent to a 1-layer RBM)
- Need not keep being tight:

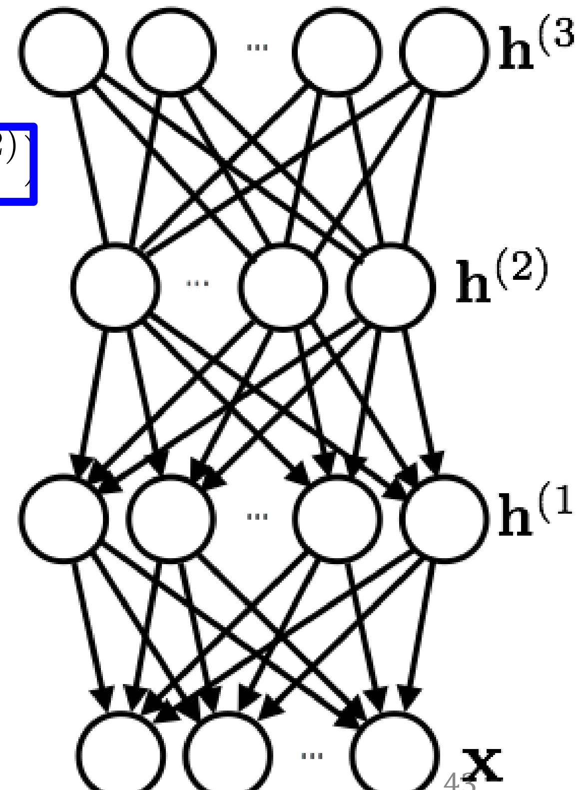
  as $p(\mathbf{h}^{(1)})$ changes, so does $p(\mathbf{h}^{(1)}|\mathbf{x})$, and so does the KL to $q(\mathbf{h}^{(1)}|\mathbf{x})$

Does the lower bound improve?

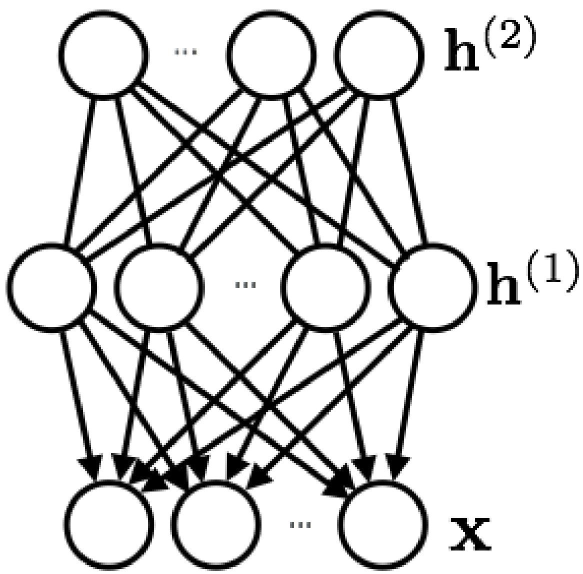
This is where the RBM stacking procedure comes from:

- **idea**: improve prior on last layer by adding another hidden layer

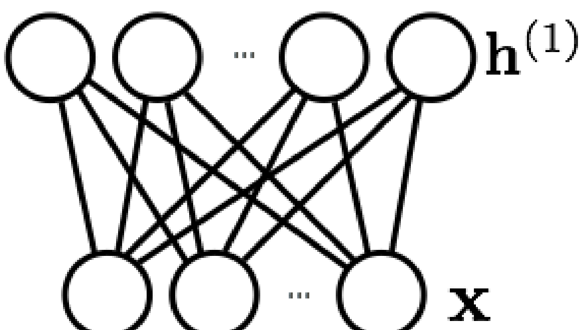
$$p(\mathbf{h}^{(1)},\mathbf{h}^{(2)}) = p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}) \sum_{\mathbf{h}^{(3)}} p(\mathbf{h}^{(2)},\mathbf{h}^{(3)})$$



$$p(\mathbf{x},\mathbf{h}^{(1)}) = p(\mathbf{x}|\mathbf{h}^{(1)}) \sum_{\mathbf{h}^{(2)}} \boxed{p(\mathbf{h}^{(1)},\mathbf{h}^{(2)})}$$



$$p(\mathbf{x}) = \sum_{\mathbf{h}^{(1)}} \boxed{p(\mathbf{x},\mathbf{h}^{(1)})}$$



Deep Belief Networks

This process of adding layers can be repeated recursively

- we obtain the greedy layer-wise pre-training procedure for neural networks

We now see that this procedure corresponds to maximizing a bound on the likelihood of the data in a DBN

- in theory, if our approximation $q(\mathbf{h}^{(1)}|\mathbf{x})$ is very far from the true posterior, the bound might be very loose
- this only means we might not be improving the true likelihood
- we might still be extracting better features!

Fine-tuning is done by the Up-Down algorithm

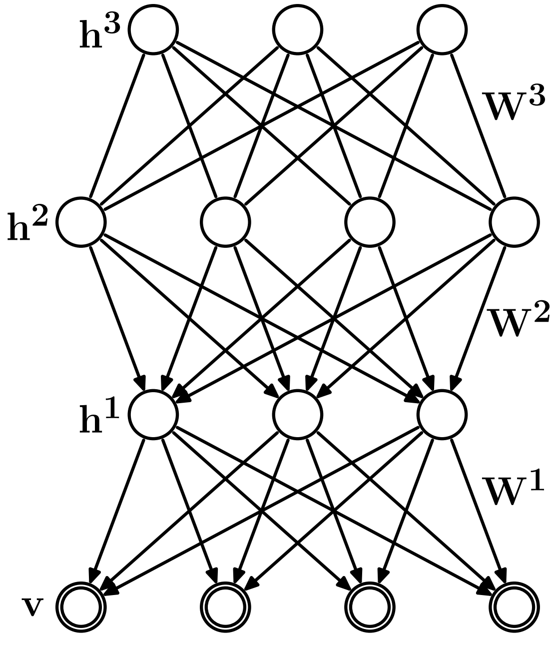
- A fast learning algorithm for deep belief nets. Hinton, Teh, Osindero, 2006.

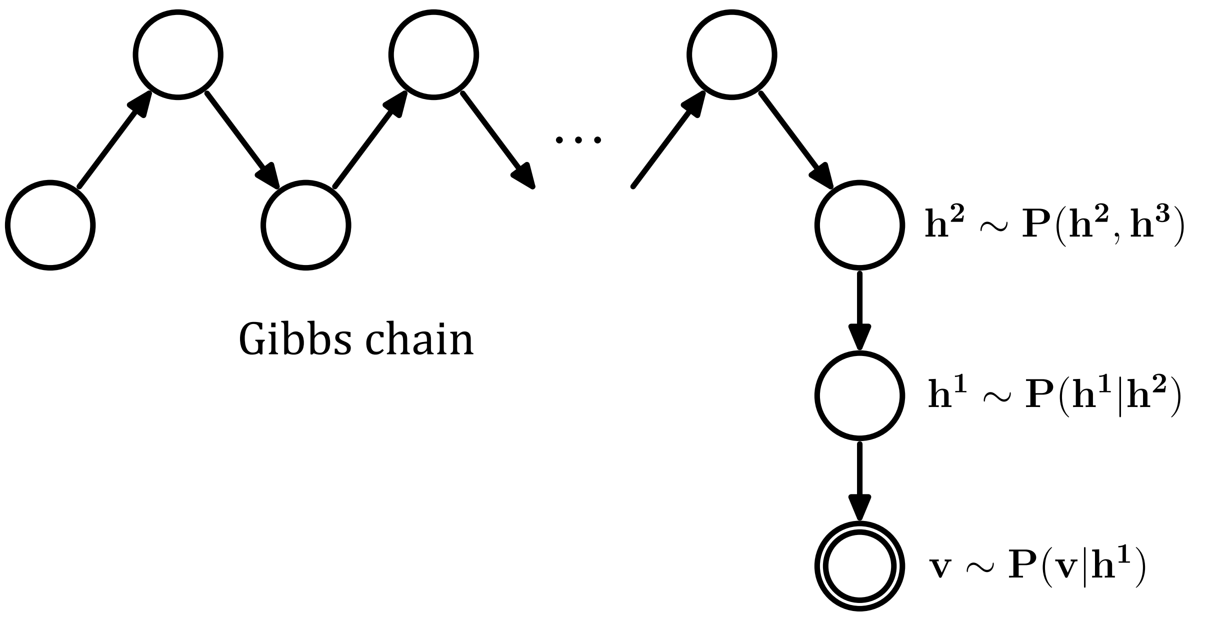
Sampling from DBNs

- To sample from the DBN model:

$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3) = P(\mathbf{v}|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2)P(\mathbf{h}^2, \mathbf{h}^3)$$

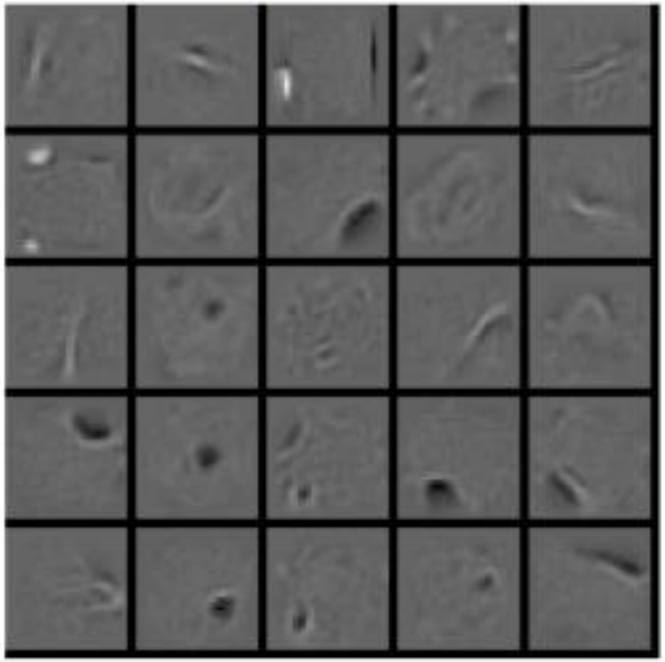
- Sample h^2 using alternating Gibbs sampling from RBM.
- Sample lower layers using sigmoid belief network.



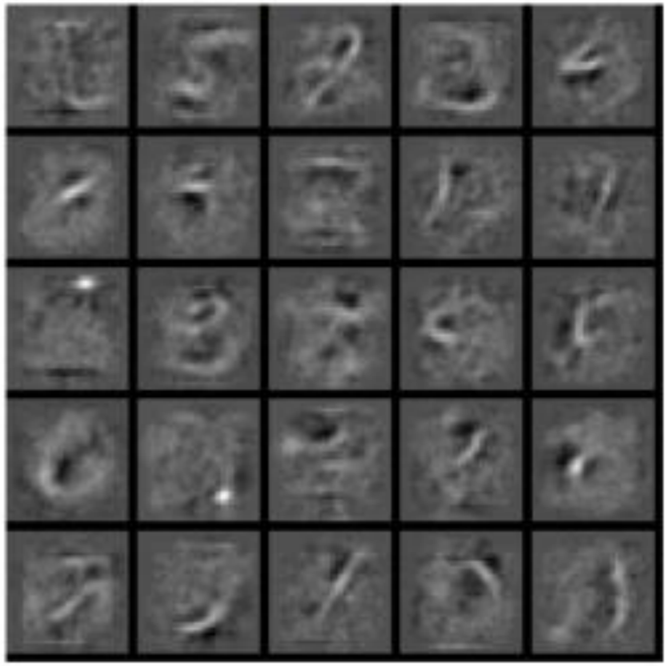


Learned Features

1^{st}-layer features

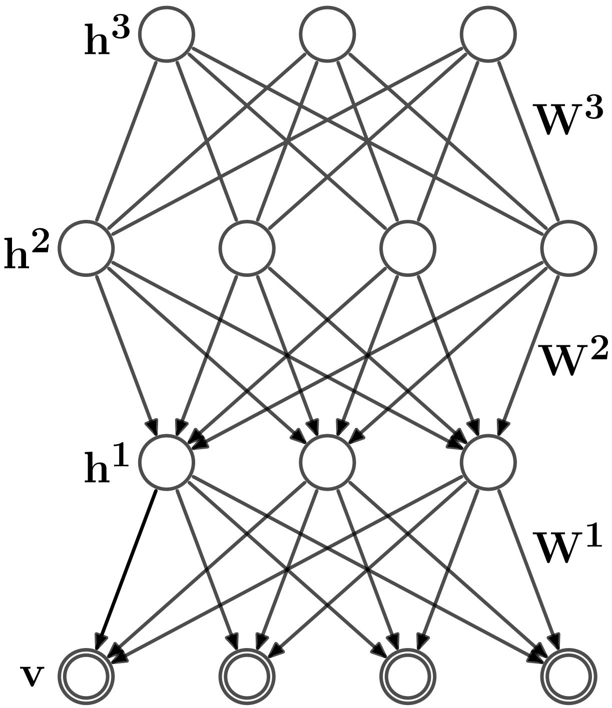


2^{nd}-layer features

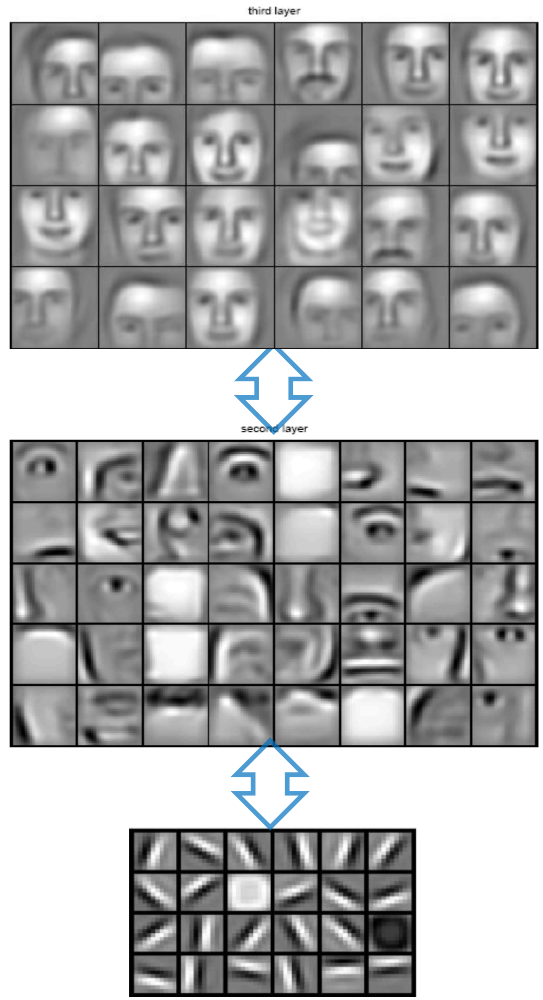


Learning Part-based Representation

Convolutional DBN



Faces



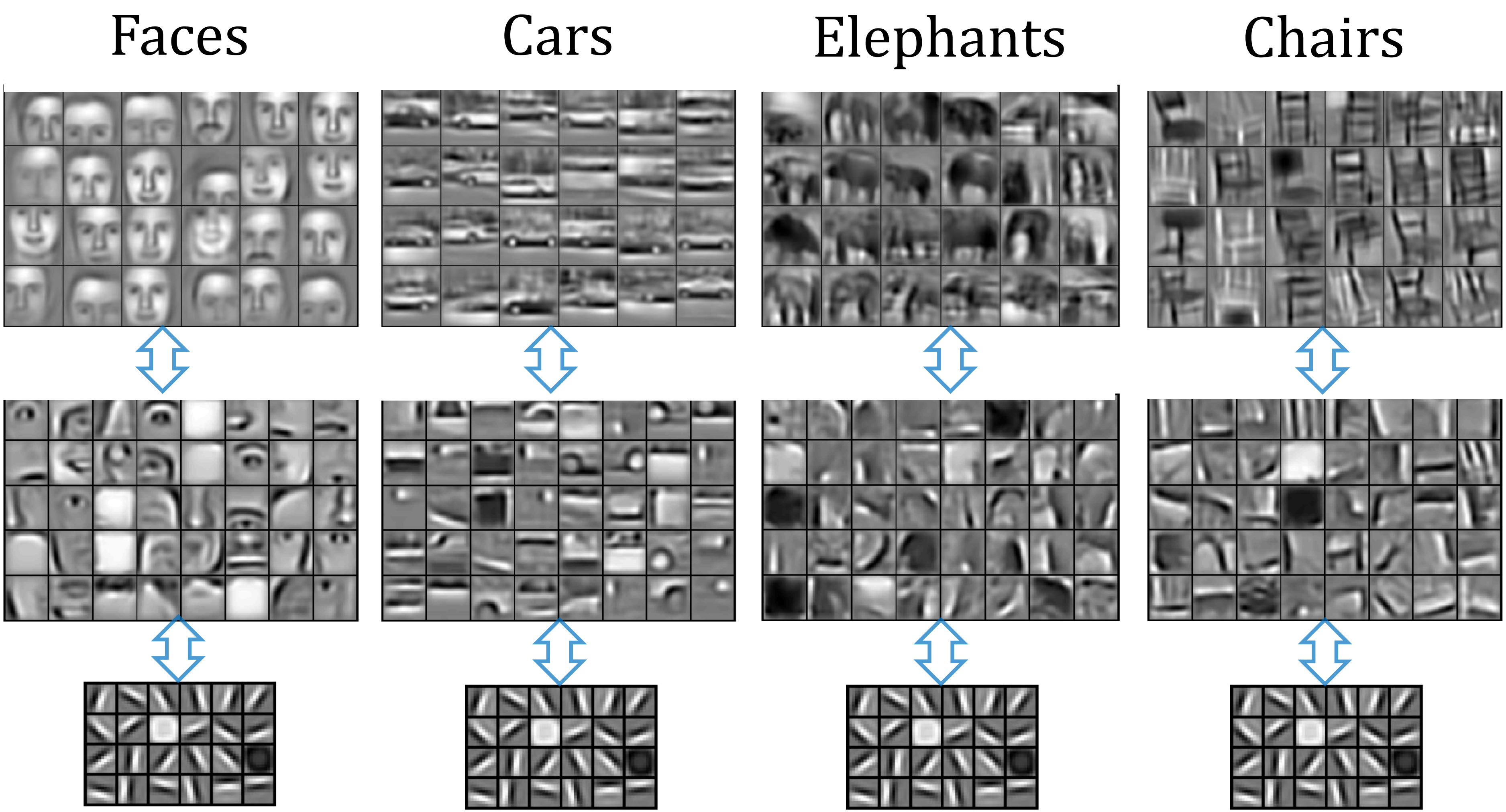
Groups of parts.

Object Parts

Trained on face images.

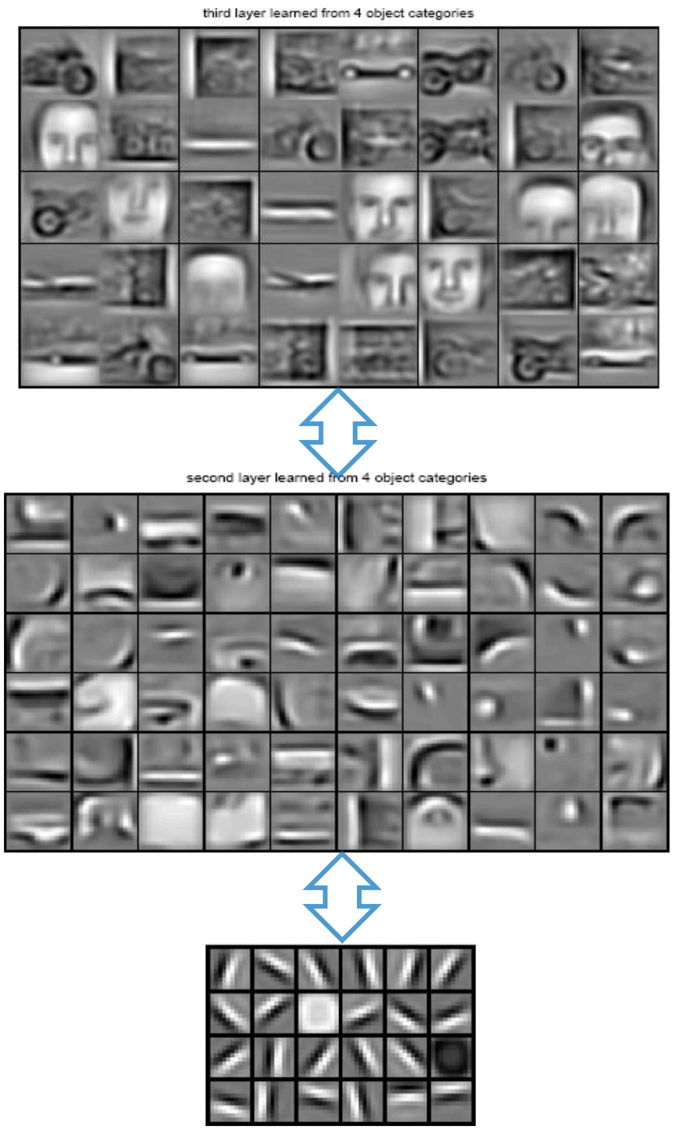
Lee et.al., ICML 2009

Learning Part-based Representation



Lee et.al., ICML 2009

Learning Part-based Representation



Groups of parts.

Class-specific object parts

Trained from multiple classes (cars, faces, motorbikes, airplanes).

Lee et.al., ICML 2009