Learning generative models with side information

Vladislav Skripnyuk

April, 2017

Outline

Generative models recap

- Discriminative models
  - learn conditional distribution $p(y|x)$
- Generative models
  - learn joint distribution $p(x, y)$ or marginal $p(x)$

too vague, let's consider it more closely

How they learn $p(x)$

- Some parametric model $p(x|\theta)$ is defined
- Observations are random variables whose distribution depends on model parameters
- optimize objective function w.r.t. model parameters
  - $\hat{\theta}_{MLE} = \arg\max_{\theta} p(x|\theta)$
  - $\hat{\theta}_{MAP} = \arg\max_{\theta} p(\theta|x) =$

    $= \arg\max_{\theta} \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta} = \arg\max_{\theta} p(x|\theta)p(\theta)$

Examples

1-d Gaussian

- Gaussian model is simple and well-known
- in 1-d case $\theta = (\mu, \sigma^2)$

$$p(x|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

- maximum likelihood estimators

$$\mu_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{MLE})^2$$

Examples

n-d Gaussian

- Gaussian model for n dimensional variables
- in n-d case $\theta = (\mu, \Sigma)$

$$p(x|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{(x_i - \mu)\Sigma^{-1}(x_i - \mu)}{2}\right)$$

- maximum likelihood estimators

$$\mu_{MLE} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\sigma_{MLE} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_{MLE})(x_i - \mu_{MLE})^T$$

Lemma

conditionals and marginals of multivariate normal

Assume an n-dimensional random vector

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

where x_1 is d-dimensional and x_2 is n-d - dimensional, has multivariate normal distribution with mean μ and covariance matrix Σ

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

Then

- The marginal distributions of x_1 and x_2 are also normal with mean vector μ_i and covariance matrix Σ_{ii} $(i = 1, 2)$, respectively.
- The conditional distribution of x_i given x_j is also normal with

$$\mu_{i|j} = \mu_i + \Sigma_{ij}\Sigma_{jj}^{-1}(x_j - \mu_j)$$

$$\Sigma_{i|j} = \Sigma_{jj} + \Sigma_{ij}^T\Sigma_{ii}^{-1}\Sigma_{ij}$$

Examples

Gaussian mixture models

- Single Gaussians cannot model
  - distributions with multiple modes
  - distributions with nonlinear correlations
- Gaussian mixture models

$$p(x) = \sum_{i=1}^{k} p(x|z = i)p(z = i)$$

  - can fit anything given enough components
  - $p(z)$ and $p(x|z)$ remain simple
  - maximum likelihood estimator cannot be derived in closed form, techniques like EM - algorithm are used

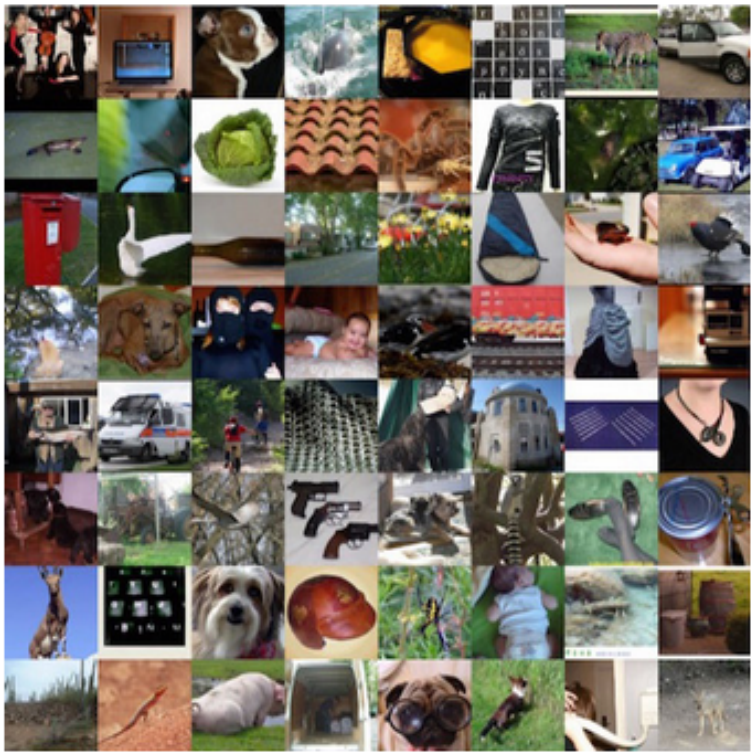
Applications

Sampling

Since we learned distribution of data p(x), we can sample from this distribution

- inversed cumulative distribution function
- ancestral sampling
- MCMC

Generated images



Real images (ImageNet)

Generated images

Generative models are still applicable for regression and classification
Bayes rule

$$p(y|x, \mathbf{X}) = \frac{p(x, y|\mathbf{X})}{\int p(x, y|\mathbf{X})dy}$$

where

$(x, y) = \tilde{x}$ - object from test set, with features x and target variable y

$\mathbf{X} = \{(x_i, y_i)\}_{i=1}^{n}$ - training set

Performing inference for objects from test set involves computing $p(\tilde{x}|\mathbf{X})$

- Bayesian treatment

$$p(\tilde{x}|\mathbf{X}) = \int p(\tilde{x}, \theta|\mathbf{X}) d\theta = \int p(\tilde{x}|\theta) p(\theta|\mathbf{X}) d\theta$$

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} \quad \text{or} \quad \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

  Integration can be intractable in general case

- Frequentist reasoning (integration is approximated with $p(\tilde{x}|\mathbf{X}, \hat{\theta})$)
  - Maximum likelihood:

$$\hat{\theta} = \arg\max_{\theta} p(\mathbf{X}|\theta)$$

  - Maximum a posteriori

$$\hat{\theta} = \arg\max_{\theta} p(\theta|\mathbf{X}) = \arg\max_{\theta} p(\mathbf{X}|\theta)p(\theta)$$

Applications

Feature extraction

Right features are critical

- representations make important aspects explicit
- remove irrelevant information
- smaller feature space

Examples: hidden variables for GMM and RBM

Applications

Data compression

ImageNet provides information about how our world looks like.

- It contains 14.000.000 images = 200GB of pixel data.
- GANs have approximately 100 million parameters so they can compress 200 GB of pixel data into 100 MB of weights

Boltzmann machines

Definition

Distribution over set of binary variables $x = \{x_i\}_{i=1}^n \quad x_i \in \{0, 1\}$

$$p(x|\theta) = \frac{1}{Z(\theta)} \exp(-E(x|\theta))$$

$$\theta = (W, b) \qquad E(x) = -x^T W x - b^T x \qquad Z(\theta) = \sum_x \exp(-E(x|\theta))$$

- W - symmetric matrix with zeros along the diagonal, w_{ij} - connection strength between unit i and unit j
- b - bias, b_i - bias of unit i in the global energy function

They are named after the Boltzmann distribution in statistical mechanics

Boltzmann machines

Motivation

Why are Boltzmann machines so appealing?

- some terminology borrowed from physics
- biological plausibility
  - Hebb's rule: fire together, wire together
  - wakefulness and dream sleep during learning
- Hinton, Simon Osindero, Yee-Whye Teh "A fast learning algorithm for deep belief nets" in 2006 put an end to "Neural Net Winter"

Confronting the partition function of unnormalized probability distributions

$$p(x|\theta) = \frac{1}{Z(\theta)}\tilde{p}(x|\theta)$$

$$Z(\theta) = \int \tilde{p}(x|\theta)dx \quad \text{or} \quad Z(\theta) = \sum_x \tilde{p}(x|\theta)$$

This operation is intractable for interesting models, straight forward maximum likelihood approach is not applicable.

Gradient ascent for log-likelihood

$$\nabla_\theta \log p(x|\theta) = \nabla_\theta \log \tilde{p}(x|\theta) - \nabla_\theta \log Z(\theta)$$

This is a well-known decomposition into the **positive phase** and **negative phase** of learning

$$\nabla_\theta \log Z(\theta) \approx \frac{\log Z(\theta + \epsilon) - \log Z(\theta)}{\|\epsilon\|}$$

approximate $Z(\theta) = \int \tilde{p}(x|\theta)dx$ with Monte Carlo sampling

Gradient ascent for log-likelihood

$$\nabla_\theta \log p(x|\theta) = \nabla_\theta \log \tilde{p}(x|\theta) - \nabla_\theta \log Z(\theta)$$

This is a well-known decomposition into the **positive phase** and **negative phase** of learning

$$\nabla_\theta \log Z(\theta) \approx \frac{\log Z(\theta + \epsilon) - \log Z(\theta)}{\|\epsilon\|}$$

approximate $Z(\theta) = \int \tilde{p}(x|\theta)dx$ with Monte Carlo sampling

What's wrong?

Negative phase

$$\nabla_\theta \log Z(\theta) = \frac{\nabla_\theta Z(\theta)}{Z(\theta)} = \frac{\nabla_\theta \sum_x \tilde{p}(x|\theta)}{Z(\theta)} = \frac{\sum_x \nabla_\theta \tilde{p}(x|\theta)}{Z(\theta)}$$

For models that guarantee $p(x) > 0$ for all x, we can substitute $\exp(\log \tilde{p}(x))$ for $\tilde{p}(x)$

$$\frac{\sum_x \nabla_\theta \exp(\log(\tilde{p}(x|\theta)))}{Z(\theta)} = \frac{\sum_x \exp(\log(\tilde{p}(x|\theta)))\nabla_\theta \log(\tilde{p}(x|\theta))}{Z(\theta)} =$$

$$= \frac{\sum_x \tilde{p}(x|\theta)\nabla_\theta \log(\tilde{p}(x|\theta))}{Z(\theta)} = \sum_x p(x|\theta)\nabla_\theta \log(\tilde{p}(x|\theta)) =$$

$$= \mathbb{E}_{x\sim p(x)} \nabla_\theta \log(\tilde{p}(x|\theta))$$

Negative phase

$$\nabla_\theta \log Z(\theta) = \mathbb{E}_{x \sim p(x)} \nabla_\theta \log(\tilde{p}(x|\theta))$$

Now we can approximate it using Monte Carlo sampling

This derivation made use of summation over discrete x, but a similar result applies using integration over continuous x:

$$\nabla_\theta \sum_x \tilde{p}(x|\theta) = \sum_x \nabla_\theta \tilde{p}(x|\theta) \iff \nabla_\theta \int_x \tilde{p}(x|\theta) dx = \int_x \nabla_\theta \tilde{p}(x|\theta) dx$$

Leibniz's rule (under some regularity conditions)

$$\frac{\mathrm{d}}{\mathrm{d}y} I(y) = \frac{\mathrm{d}}{\mathrm{d}y} \int f(x, y) dx = \int \frac{\partial}{\partial y} f(x, y) dx$$

Boltzmann machines

LLH gradient

Under the assumption of independence gradient of mean log-likelihood becomes

$$\nabla_\theta \frac{1}{n} \log \mathcal{L}(x_1, .., x_n|\theta) = \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} \log p(x_i|\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log p(x_i|\theta) =$$

$$= \sum_{i=1}^{n} \frac{1}{n} \nabla_\theta \log \tilde{p}(x_i|\theta) - \nabla_\theta \log Z(\theta) =$$

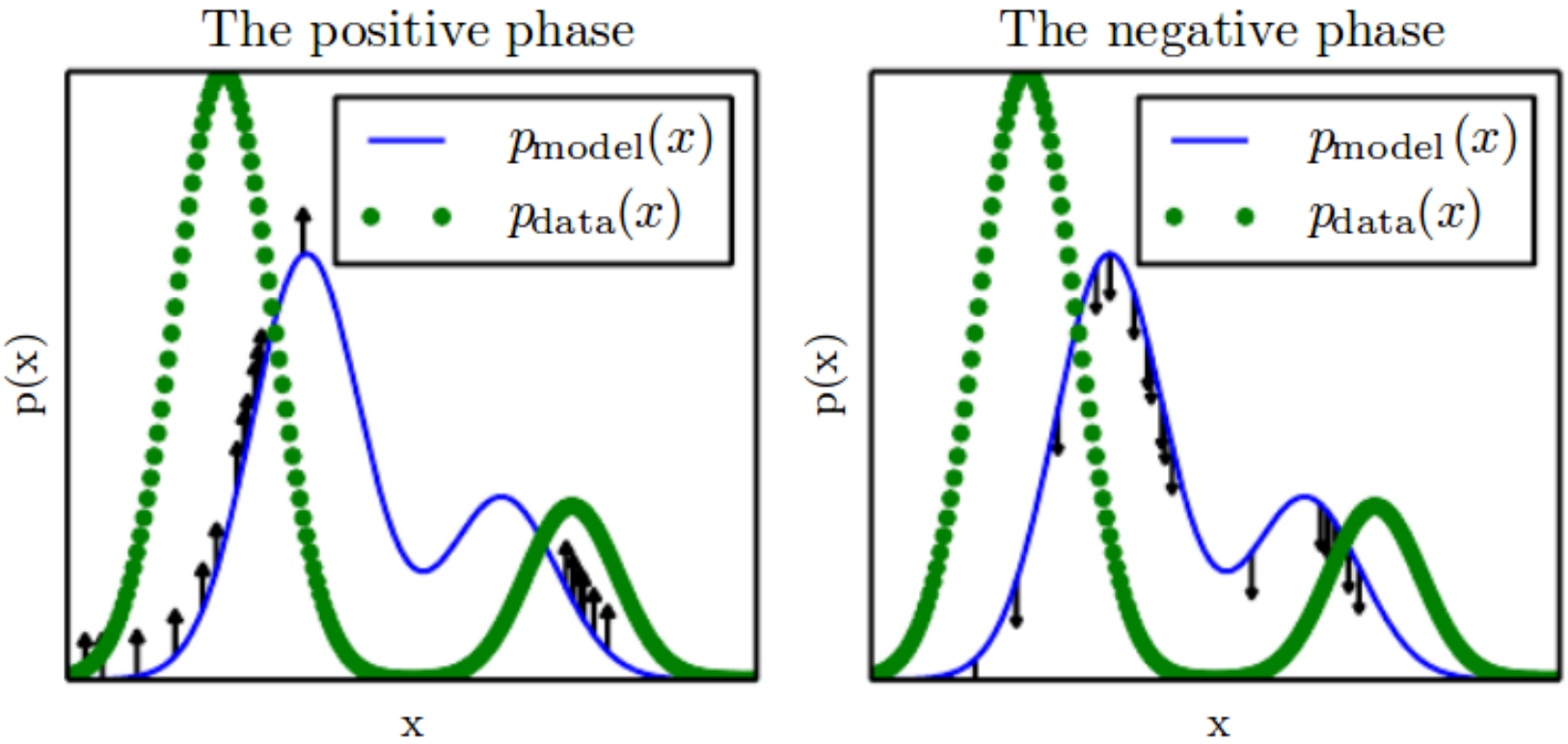
$$= \mathbb{E}_{x \sim p_{data}(x)} \nabla_\theta \log(\tilde{p}(x|\theta)) - \mathbb{E}_{x \sim p_{model}(x)} \nabla_\theta \log(\tilde{p}(x|\theta))$$

- To approximate first expectation we can sample mini-batches from train set
- for second expectation we need more sophisticated techniques for sampling

Boltzmann machines

wake sleep algorithm

- in the positive phase we push up on unnormalized probabilty of points sampled from data distribution
- in the negative phase we push down on unnormalized probabilty of points sampled from current model distribution



Boltzmann machines

wake sleep algorithm

This process resembles the change between sleep and wakefulness

- during wakefulness (positive phase) model experiences real world data and increases it's probability
- during sleep (negative phase) model dreams or experiences samples from current distribution and decreases their probability

Hidden variables

- For GMM we introduced hidden variables to increase it's expressive power.
- Although learning is impractical in general Boltzmann machines, it can be made quite efficient when we keep only connections between hidden and visible variables.

Boltzmann machines

RBMs

Two previous points give rise to a new type of model called **Restricted Boltzmann Machine** (RBM).

Definition

Distribution over set of visible and hidden binary variables

$$v = \{v_i\}_{i=1}^n \text{ and } h = \{h_i\}_{i=1}^m \quad v_i \in \{0,1\}, h_i \in \{0,1\}$$

$$p(v,h|\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h|\theta)) \quad \theta = (W,b,c)$$

$$E(x) = -h^T W v - b^T v - c^T h \qquad Z(\theta) = \sum_{v,h} \exp(-E(v,h|\theta))$$

- W - $m \times n$ matrix, w_{ij} - connection strength between h_i and v_j
- b, c - bias

Gradient ascent for RBMs

$$p(v|\theta) = \sum_h p(v, h|\theta) = \sum_h \frac{e^{-E(v,h|\theta)}}{Z(\theta)}$$

To map this formulation to one similar to case of EBM without hidden variables, we introduce the notation (inspired from physics) of free energy, defined as follows:

$$\mathcal{F}(v|\theta) = -\log \sum_h e^{-E(v,h|\theta)}$$

which allows us to write,

$$p(v|\theta) = \frac{e^{-\mathcal{F}(v|\theta)}}{Z(\theta)}$$

For models with partition function we derived:

$$\nabla_\theta \log p(x) = \nabla_\theta \log(\tilde{p}(x|\theta)) - \mathbb{E}_{x \sim p(x|\theta)} \nabla_\theta \log(\tilde{p}(x|\theta))$$

For EBM with hidden variables $\tilde{p}(x|\theta) = e^{-\mathcal{F}(x|\theta)}$, so

$$\nabla_\theta \log p(x) = -\nabla_\theta \mathcal{F}(x|\theta) + \mathbb{E}_{x \sim p(x|\theta)} \nabla_\theta \mathcal{F}(x|\theta)$$

For RBMs with binary units free energy takes notably simple form

$$\mathcal{F}(v|\theta) = -\log \sum_h e^{-E(v,h|\theta)} = -\log \sum_h e^{h^T W v + b^T v + c^T h} =$$

$$= -\log \sum_{h_1} .. \sum_{h_m} e^{h^T (Wv+c)} e^{b^T v} = -b^T v - \log \sum_{h_1} .. \sum_{h_m} \prod_{i=1}^{n} e^{h_i (Wv+c)_i} =$$

$$= -b^T v - \log \prod_{i=1}^{n} \sum_{h_i} e^{h_i (Wv+c)_i} = -b^T v - \sum_{i=1}^{n} \log \sum_{h_i} e^{h_i (Wv+c)_i} =$$

$$= -b^T v - \sum_{i=1}^{n} \log(1 + e^{(Wv+c)_i})$$

Finally we can compute partial derivatives of $\mathcal{F}(v)$ w.r.t. parameters

$$\frac{\partial \mathcal{F}(v)}{\partial W_{ij}} = -\frac{e^{(Wv+c)_i} v_j}{1 + e^{(Wv+c)_i}} = -\sigma((Wv + c)_i) v_i$$

$$\frac{\partial \mathcal{F}(v)}{\partial c_i} = -\frac{e^{(Wv+c)_i}}{1 + e^{(Wv+c)_i}} = -\sigma((Wv + c)_i)$$

$$\frac{\partial \mathcal{F}(v)}{\partial b_i} = -v_i$$

Due to the specific structure of RBMs, visible and hidden units are conditionally independent given one-another

$$p(h|v) = \frac{1}{p(v)}\frac{1}{Z}\exp(b^T v + c^T h + h^T W v) =$$

$$= \frac{1}{Z'}\prod_{i=1}^{n}\exp(h_i(c + Wv)_i) = \prod_{i=1}^{n} p(h_i|v)$$

Similarly

$$p(v|h) = \prod_{i=1}^{n} p(v_i|h)$$

Univariate conditionals

$$p(h_i = 1|v) = \frac{\tilde{p}(h_i = 1|v)}{\tilde{p}(h_i = 0|v) + \tilde{p}(h_i = 1|v)} = \frac{e^{(Wv+c)_i}}{e^0 + e^{(Wv+c)_i}} = \sigma((Wv+c)_i)$$

Similarly

$$p(v_i = 1|h) = \frac{\tilde{p}(v_i = 1|h)}{\tilde{p}(v_i = 0|h) + \tilde{p}(v_i = 1|h)} = \frac{e^{(hW+b)_i}}{e^0 + e^{(hW+b)_i}} = \sigma((hW+b)_i)$$

MCMC - Markov Chain Monte Carlo methods help to draw samples from complicated probability distributions

Intuition

- We want to walk randomly (construct **Markov Chain**) in state space of random variables, so that the number of times we visit each region is in proportion to measure concentrated in that region
- Distribution over state space at each step becomes closer to desired distribution when we make more steps (**Monte Carlo** algorithm)

Definition

$$X = \{X^{(k)} | k \in \mathbb{N}_0\}$$

X_k take values in a finite set Ω, and $\forall k > 0$ and $\forall j, i, i_0, .., i_{k-1} \in \Omega$

$$p_{ij}^k = P(X^{(k+1)} = j | X^{(k)} = i, .., X^{(0)} = i_0) = P(X^{(k+1)} = j | X^{(k)} = i)$$

If p_{ij}^k does not depend on k, the chain is called homogeneous and the matrix $P = \{p_{ij}\}_{i,j \in \Omega}$ is called the **transition matrix**.

If the starting distribution is given by the probability vector $\mu^{(0)}$, the distribution $\mu^{(k)}$ over $X^{(k)}$ is given by

$$\mu^{(k)T} = \mu^{(0)T} P^k$$

A distribution π is called a stationary distribution if

$$\pi^T = \pi^T P$$

A sufficient (but not necessary) condition for a distribution π to be stationary w.r.t. a Markov chain described by the transition probabilities $\{p_{ij}\}_{i,j \in \Omega}$ is that

$$\forall i, j \in \Omega \quad \pi(i) p_{ij} = \pi(j) p_{ji}$$

This is called the **detailed balance condition**

Markov Chain is irreducible if

$$\forall i, j \in \Omega \exists k > 0 \quad P(X^{(k)} = j | X^{(0)} = i) > 0$$

Theorem

Irreducible Markov Chain over finite state space has unique stationary distribution.

Markov chain is aperiodic if

$$\forall i \in \Omega \quad \gcd(\{k \in \mathbb{N}_0 | P(X^{(k)} = i | X^{(0)} = i)\}) = 1$$

Let for two distributions α and β on a finite state space Ω the **distance of variation** be defined as

$$d_V(\alpha, \beta) = \frac{1}{2} \sum_{x \in \Omega} |\alpha(x) - \beta(x)|$$

Theorem

Let π be the stationary distribution of an irreducible and aperiodic Markov chain on a finite state space with transition matrix P. For an arbitrary starting distribution μ,

$$\lim_{k \to \infty} d_V(\mu^T P^k, \pi^T) = 0$$

With that in mind, in order to sample from a distribution q with a finite state space, we can now construct a Markov Chain, which

- is irreducible and aperiodic
- has stationary distribution $\pi = q$

If k is large enough, the state $x^{(k)}$ of $X^{(k)}$ from the constructed chain is then approximately a sample from π and therefore from q.
To construct such a chain we can use Gibbs sampling

We consider an MRF $\mathcal{X} = (X_1, .., X_n)$ X_i take values in finite set Λ

$$\pi(X) = \frac{1}{Z} e^{-E(X)}$$

Sampling

We construct Markov Chain $X = \{X^{(k)} | k \in \mathbb{N}_0\}$, so that

$$X^{(k)} = (X_1^{(k)}, .., X_n^{(k)}) \in \Lambda^n$$

- First, a variable $X_i, i \in V$ is randomly picked with a probability $q(i)$ given by a strictly positive probability distribution q on V
- new state for X_i is sampled based on its conditional probability distribution given the state $(X_v)_{v \in V \setminus i}$

The transition probability p_{xy} for two states x, y of the MRF $\mathcal{X}$ with $x \neq y$ is

$$p_{xy} = \begin{cases} q(i)\pi(y_i|(x_v)_{v \in V \setminus i}), & \exists i \in V \text{ so that } \forall v \in V \quad v \neq i : x_v = y_v \\ 0 & \textit{else} \end{cases}$$

$$p_{xx} = \sum_{i \in V} q(i)\pi(x_i|(x_v)_{v \in V \setminus i})$$

Proof of detailed balance condition

$$\forall i, j \in \Omega \quad \pi(i)p_{ij} = \pi(j)p_{ji}$$

- if x = y

$$\pi(x)p_{xx} = \pi(x)p_{xx}$$

- if x differs from y in more than one place

$$\pi(x)p_{xy} = 0 = \pi(y)p_{yx}$$

- if x differs from y in one position

$$\pi(x)p_{xy} = \pi(x)q(i)\pi(y_i|(x_v)_{v\in V\setminus i}) =$$

$$= \pi(x_i, (x_v)_{v\in V\setminus i})q(i)\frac{\pi(y_i, (x_v)_{v\in V\setminus i})}{\pi((x_v)_{v\in V\setminus i})} =$$

$$= \pi(y_i, (x_v)_{v\in V\setminus i})q(i)\frac{\pi(x_i, (x_v)_{v\in V\setminus i})}{\pi((x_v)_{v\in V\setminus i})} =$$

$$= \pi(y)q(i)\pi(x_i|(x_v)_{v\in V\setminus i}) = \pi(y)p_{yx}$$

Since conditional distributions are strictly positive, constructed Markov Chain is irreducible and aperiodic.

Finally

Aperiodicity and irreducibility guarantee that the chain converges to the stationary distribution $\pi(x)$

In practice, the single random variables to be updated are chosen in a fixed predefined order. The corresponding algorithm is called **periodic Gibbs sampler**.

Since conditional distributions $p(v|h)$ and $p(h|v)$ are fully factorized, we can sample h given v simultaneously (and vice versa).

Algorithm 18.1 A naive MCMC algorithm for maximizing the log-likelihood with an intractable partition function using gradient ascent.

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow burn in. Perhaps 100 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set.
  $\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  Initialize a set of m samples $\{\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(m)}\}$ to random values (e.g., from a uniform or normal distribution, or possibly a distribution with marginals matched to the model's marginals).
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

Algorithm 18.2 The contrastive divergence algorithm, using gradient ascent as the optimization procedure.

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow a Markov chain sampling from $p(\mathbf{x}; \boldsymbol{\theta})$ to mix when initialized from p_{data}. Perhaps 1-20 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set.
  $\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  for $i = 1$ to m **do**
    $\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{x}^{(i)}$.
  end for
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

Boltzmann machines

Learning

Algorithm 18.3 The stochastic maximum likelihood / persistent contrastive divergence algorithm using gradient ascent as the optimization procedure.

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow a Markov chain sampling from $p(\mathbf{x}; \boldsymbol{\theta} + \epsilon \mathbf{g})$ to burn in, starting from samples from $p(\mathbf{x}; \boldsymbol{\theta})$. Perhaps 1 for RBM on a small image patch, or 5-50 for a more complicated model like a DBM.
Initialize a set of m samples $\{\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(m)}\}$ to random values (e.g., from a uniform or normal distribution, or possibly a distribution with marginals matched to the model's marginals).
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set.
  $\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

Boltzmann machines

Pseudolikelihood

We can easily compute ratios of probabilities:

$$\frac{p(x)}{p(y)} = \frac{\frac{1}{Z}\tilde{p}(x)}{\frac{1}{Z}\tilde{p}(y)} = \frac{\tilde{p}(x)}{\tilde{p}(y)}$$

Conditional probabilities contain ratios

$$\log p(x) = \log p(x_1) + \log p(x_2|x_1) + .. + \log p(x_n|x_{1:n-1})$$

Pseudolikelihood objective

$$\sum_{i=1}^{n} \log p(x_i|x_{-i})$$

Generalized pseudolikelihood objective

$$\sum_{i=1}^{m} \log p(x_{\mathcal{S}^{(i)}}|x_{-\mathcal{S}^{(i)}})$$

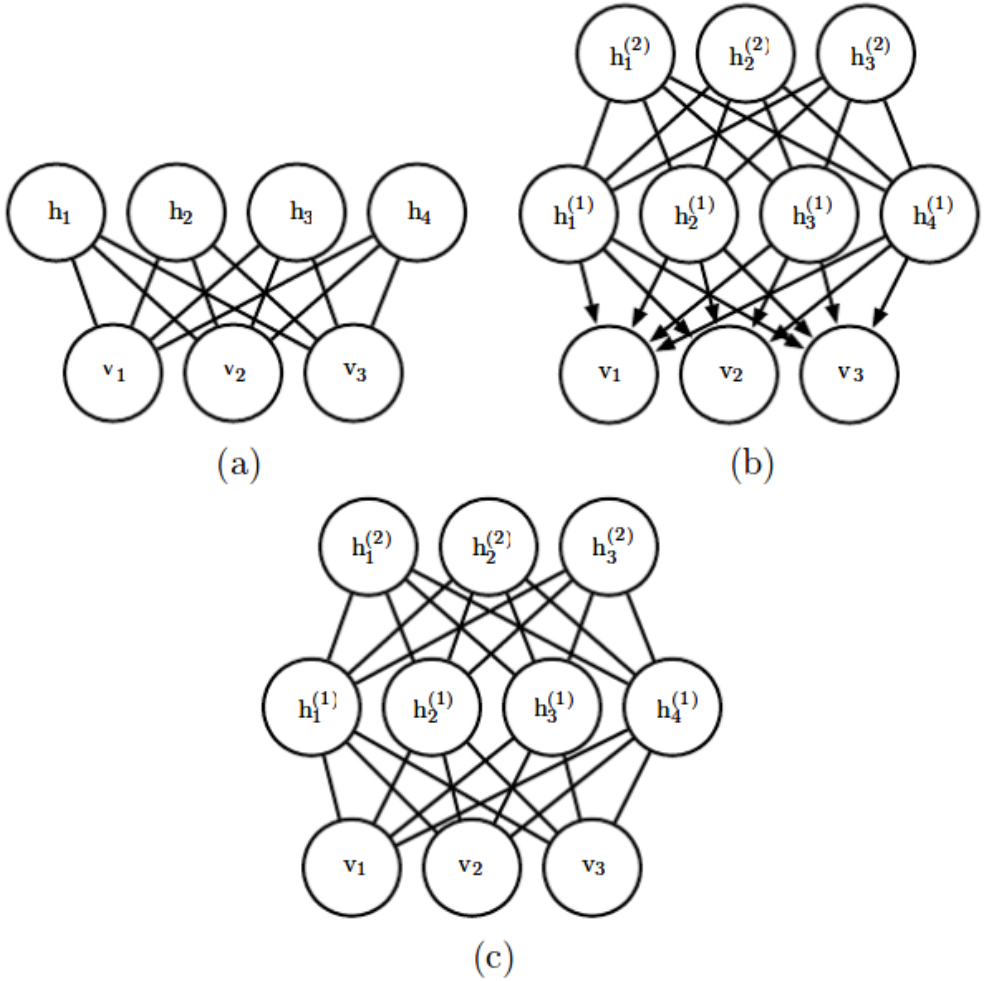
RBM can be treated as one fully connected layer with sigmoidal nonlinearity.
So they can be stacked and form deep models like
Deep Belief Networks *and* **Deep Boltzmann machines**

Possible scenario

RBMs can be stacked and trained layerwise, then fine-tuned in supervised manner with added objective function on the top.

Boltzmann machines

Deep models



Thanks you for attention!