Deep Generative Models
Capítulo 20

Capítulo 20
Gabriel Buginga
PESC

Sumário

Capítulo 20

Gabriel Buginga

PESC

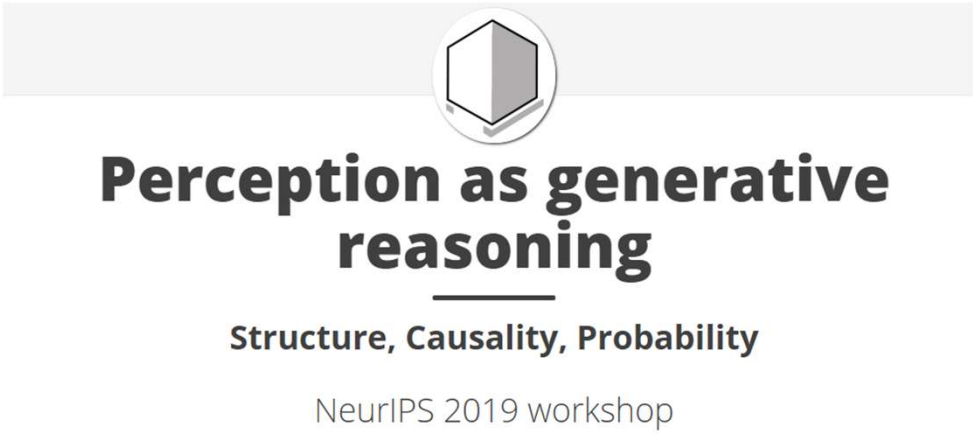
O que são?

- Essencialmente descobrir a função de distribuição dos dados, de tal forma que entre outras objetivos, se consiga gerar amostras $\mathrm{x} \sim p(x)$.
  - *Naive Bayes*
  - Histograma
    - O problema de dimensionalidade muito alta cria dificuldades intransponíveis para abordagens dessa natureza de contagem básica.
- O "*Deep*" informa que esses modelos usam ligações hierárquicas de variáveis latentes ou simplemente mais camadas em modelos de redes neurais de diversas arquiteturas. Tema de todo o curso.
- Note que modelos discriminatórios não são treinados para isso. Eles modelam probabilidades condicionais $p(y|x)$. Lembre-se do softmax.

Por que estudar modelos generativos?

- Identificar anomalias: pergunte à p(x): se muito pequena então esse x provavelmente não veio da distribuição.
- Gerar amostras de distribuições complexas, como images de cenas naturais, faces e dados médicos para diversos fins. Como aprendizado não-supervisionado, ou semi-supervisionado.
- Compressão.

- *Free Energy Principle:* teoria de funcionamento do cérebro que aplica modelo generativos para explicar a diminuição de uma quantidade chamada “free energy” que calcula um limite para a surpresa do agente estar em determinado estado. Menor surpresa maior chance de o organismo estar num estado homeostático; “peixe fora d’água possui enorme supresa sensorial”.



20.1 Boltzmann Machines

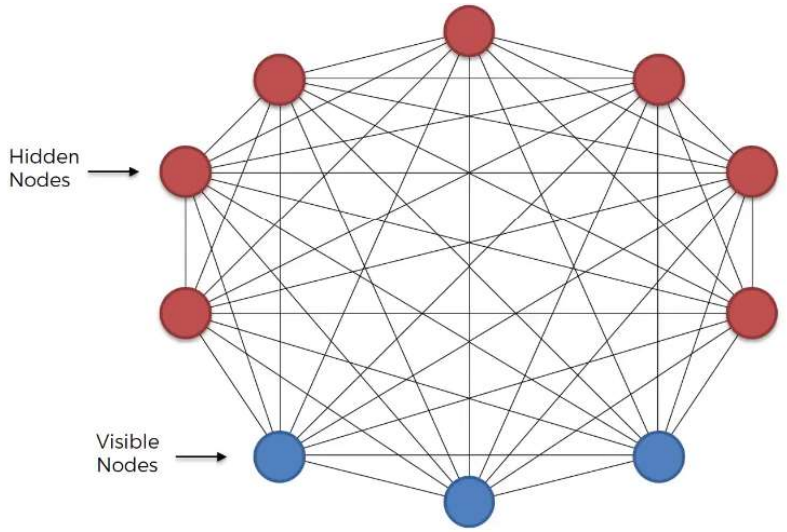
- A máquina de Boltzmann é um modelo probabilístico baseado em energia que modela uma distribuição sobre um vetor binário d-dimensional.
- Podemos aumentar o poder de expressão do modelo adicionando **variáveis latentes** (similar ao MLP), com isso a Máquina de Boltzmann torna-se um aproximador universal de PMF's sobre variáveis discretas.

$$\mathbf{x} \in \{0,1\}^d \qquad E(\boldsymbol{x}) = -\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x},$$

$$P(\boldsymbol{x}) = \frac{\exp\left(-E(\boldsymbol{x})\right)}{Z};$$

Função de partição

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \boldsymbol{R}\boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{S}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}.$$



20.1 Boltzmann Machines

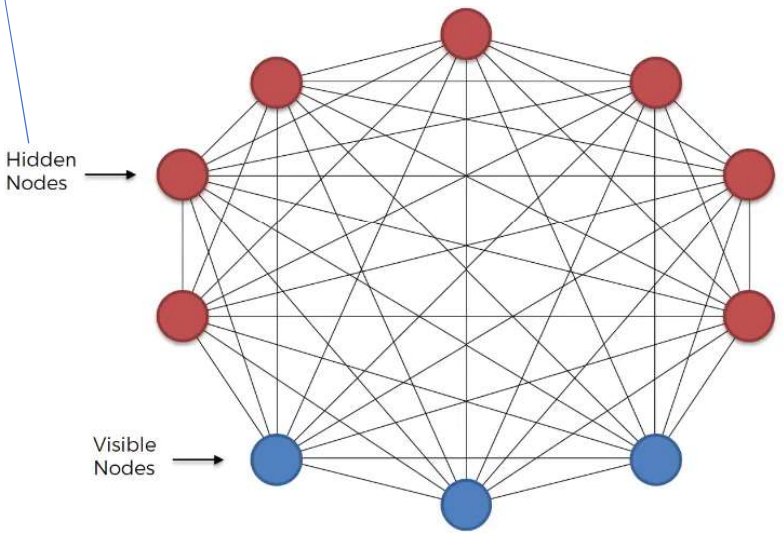
- Aprendizado: o modelo possui função de partição intratável, portanto devem ser usados os métodos do Capítulo 18. Especialmente os métodos de Maximum Likelihood.
  1. Inicialize os pesos
  2. Deixe a rede “rodar” para obter amostras do estado simuladas
  3. Compute o gradiente e atualize os pesos
  4. Itere mais uma vez
- Aplica uma forma de aprendizado local: Hebbian; o novo valor de um peso depende apenas das estatísticas das unidades envolvidas.

$$\mathbf{x} \in \{0,1\}^d \qquad E(\boldsymbol{x}) = -\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x},$$

$$P(\boldsymbol{x}) = \frac{\exp\left(-E(\boldsymbol{x})\right)}{Z},$$

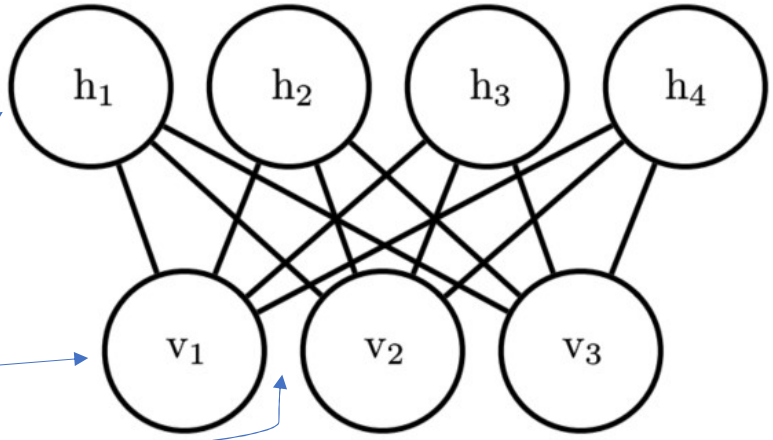
Função de partição

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \boldsymbol{R}\boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{S}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}.$$



20.2 Restricted Boltzmann Machines

- As RBMs são modelos probabilísticos estruturados com ligações não direcionadas, possuindo
  - uma camada de variáveis latentes
  - uma camada de variáveis vísiveis
  - não possui ligação dentro das camadas



RBM

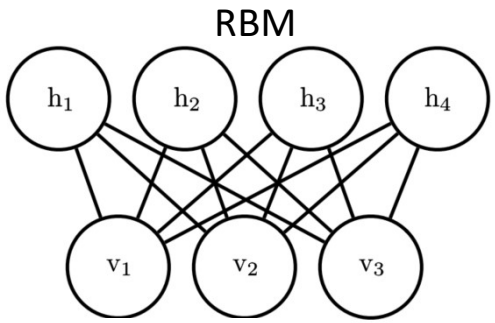
$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h},$$

$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)$$

Função partição **intratável**

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}, \boldsymbol{h})\right\}$$

20.2 Restricted Boltzmann Machines

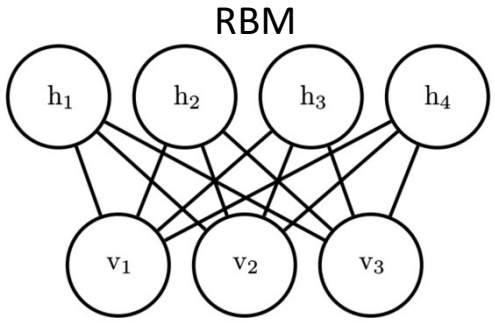


- Considere que v e h são variáveis binárias multidimensionais (RBMs podem ser extendidas para outra distribuições).
- As condicionais são fatorizáveis dada a falta de conexões dentro das camadas.
- Perceba que os v's são constantes em relação a essa distribuição condicional.

$$
\begin{aligned}
P(\boldsymbol{h} \mid \boldsymbol{v}) &= \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})} \\
&= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\left\{ \boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} \right\} \\
&= \frac{1}{Z'} \exp\left\{ \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} \right\} \\
&= \frac{1}{Z'} \exp\left\{ \sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j \right\} \\
&= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left\{ c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j \right\}.
\end{aligned}
$$

$$P(h_j = 1 \mid \boldsymbol{v})$$

20.2 Restricted Boltzmann Machines



- Já se sabe que a condicional é fatorial em relação aos h então:

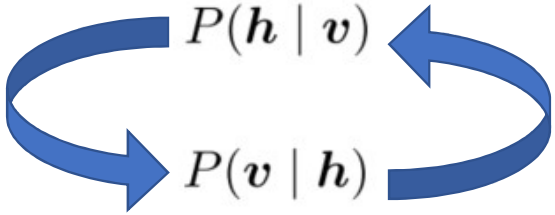
$$
\begin{aligned}
P(h_j = 1 \mid \boldsymbol{v}) &= \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \\
&= \frac{\exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}}{\exp\{0\} + \exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}} \\
&= \sigma\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right).
\end{aligned}
$$

$$
P(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{n_h} \sigma\left((2\boldsymbol{h} - 1) \odot (\boldsymbol{c} + \boldsymbol{W}^\top \boldsymbol{v})\right)_j .
$$

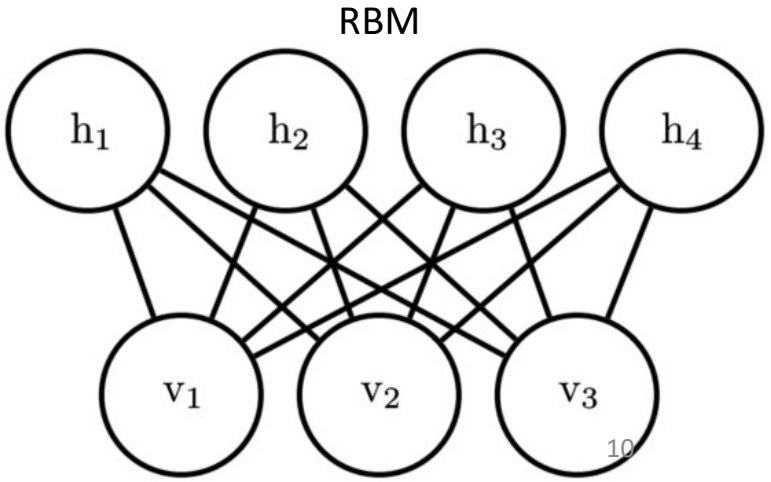
- Similarmente pode-se obter a condicional em de v em relação à h:

$$
P(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{n_v} \sigma\left((2\boldsymbol{v} - 1) \odot (\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h})\right)_i
$$

20.2 Restricted Boltzmann Machines

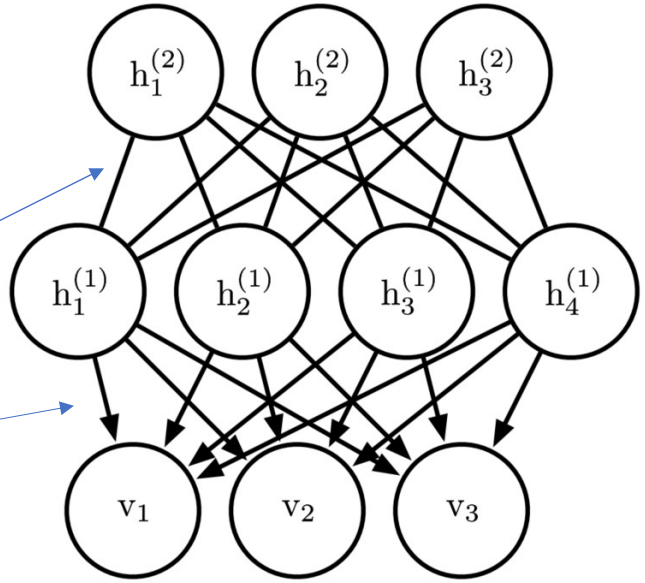


- **Treinamento:** a RBM permite cálculo e diferenciação eficiente de $\tilde{P}(\boldsymbol{v})$ e amostras MCMC eficientes na forma de amostragem de **Gibbs em bloco**.
- Portanto qualquer técnica do Capítulo 18 para modelos com função partição intratáveis pode ser usado:
  - *Constrastive Divergence*
  - *Stochastic Maximum Likelihood*
  - *Ratio Matching*
  - Etc…



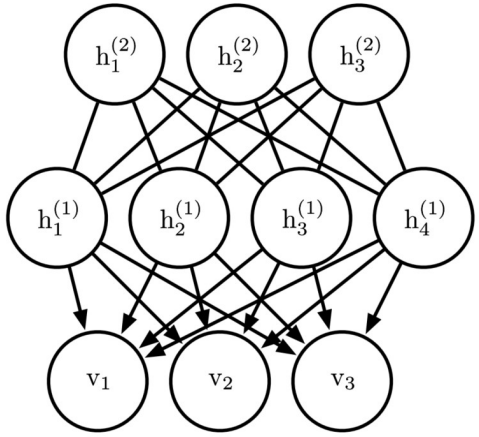
20.3 Deep Belief Networks

- Em 2006, iniciou a renascença do aprendizado profundo mostrando que modelos profundos podem ser otimizados e obter resultados melhores que modelos SVM+kernel. Porém hoje não são tão utilizados mesmo fora do paradigma generativo.
- Possuem várias camadas latentes com variáveis tipicamente binárias e uma camada visível binária ou real.
- Conexões :
  - camadas superiores são não-direcionadas
  - todas as outras apontam para a visível.
- Note que uma DBN com apenas uma camada é uma RBM (*Restricted Boltzmann Machine*).



DBN

20.3 Deep Belief Networks



DBN

- Seja uma DBN com l camadas possuindo matrizes de peso $W^{(1)}$,..., $W^{(l)}$, e bias $b^{(0)}$,..., $b^{(l)}$, a distribuição representada por essa DBN é:

Relação não-direcionada

$$P(\boldsymbol{h}^{(l)}, \boldsymbol{h}^{(l-1)}) \propto \exp\left(\boldsymbol{b}^{(l)\top}\boldsymbol{h}^{(l)} + \boldsymbol{b}^{(l-1)\top}\boldsymbol{h}^{(l-1)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l)}\right)$$

Note que a função de partição é implícita e não depende de h

Relação direcionada das demais camadas

$$P(h_i^{(k)} = 1 \mid \boldsymbol{h}^{(k+1)}) = \sigma\left(b_i^{(k)} + \boldsymbol{W}_{:,i}^{(k+1)\top}\boldsymbol{h}^{(k+1)}\right) \forall i, \forall k \in 1, \ldots, l-2,$$

Camada visível binária

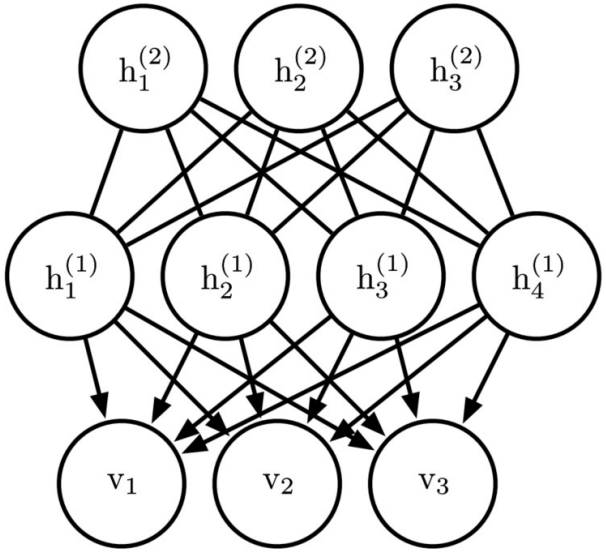
$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(b_i^{(0)} + \boldsymbol{W}_{:,i}^{(1)\top}\boldsymbol{h}^{(1)}\right) \forall i.$$

Camada visível contínua

$$\mathbf{v} \sim \mathcal{N}\left(\boldsymbol{v}; \boldsymbol{b}^{(0)} + \boldsymbol{W}^{(1)\top}\boldsymbol{h}^{(1)}, \boldsymbol{\beta}^{-1}\right)$$

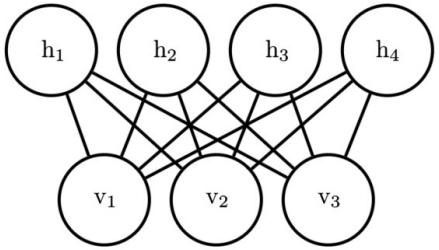
20.3 Deep Belief Networks

- Gerando amostras: amotragem de Gibbs nas duas camadas superiores e amostragem ancestral até a camada visível.
- Inferência: intratável pelas ligações direcionais (“*explaining away effect*”) e não-direcionais nas duas superiores. O ELBO também é intratável pois se necessita calcular valores esperados de cliques do tamanho da profundidade da rede.
- *Log-likelihood*: função de partição intratável e inferência intratável. **Possui ambos os problemas**.
- Então como podemos treinar uma DBN?



DBN

20.3 Deep Belief Networks



1

Começe treinando uma RBM para maximizar: $\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \log p(\boldsymbol{v})$

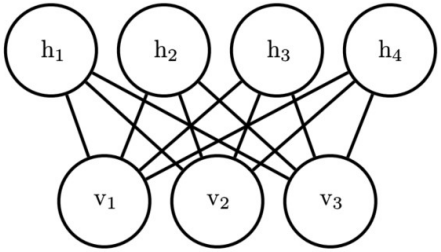
utilizando os métodos *contrastive divergence* ou *stochastic maximum likelihood.*

Os parâmetros treinados dessa RBM definirão os parametros da primeira camada da DBN.

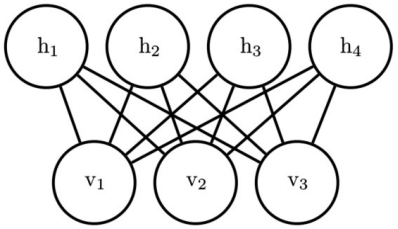
2

Depois uma segunda RBM é treinada para aproximar: $\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \mathbb{E}_{\mathbf{h}^{(1)} \sim p^{(1)}(\boldsymbol{h}^{(1)} | \boldsymbol{v})} \log p^{(2)}(\boldsymbol{h}^{(1)})$

Note que $p^{(1)}$ é a probabilidade representada pela **primeira** RBM, e que $p^{(2)}$ é a probabilidade representada pela **segunda** RBM.



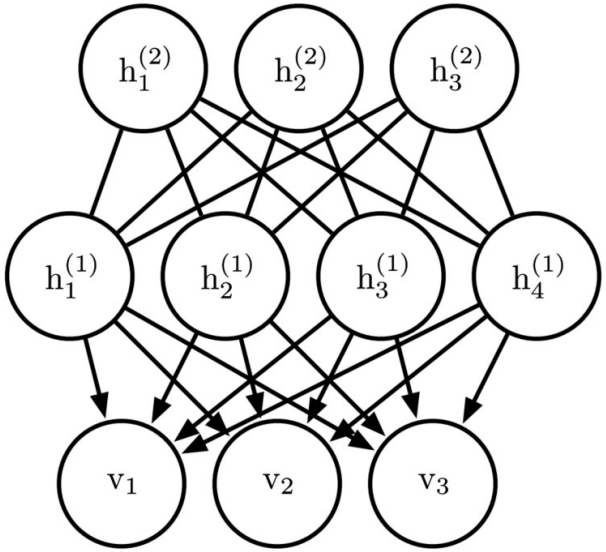
xN



Pode ser justificada como aumentando uma limite inferior variacional da *log-likehood* dos dados sob o DBN.

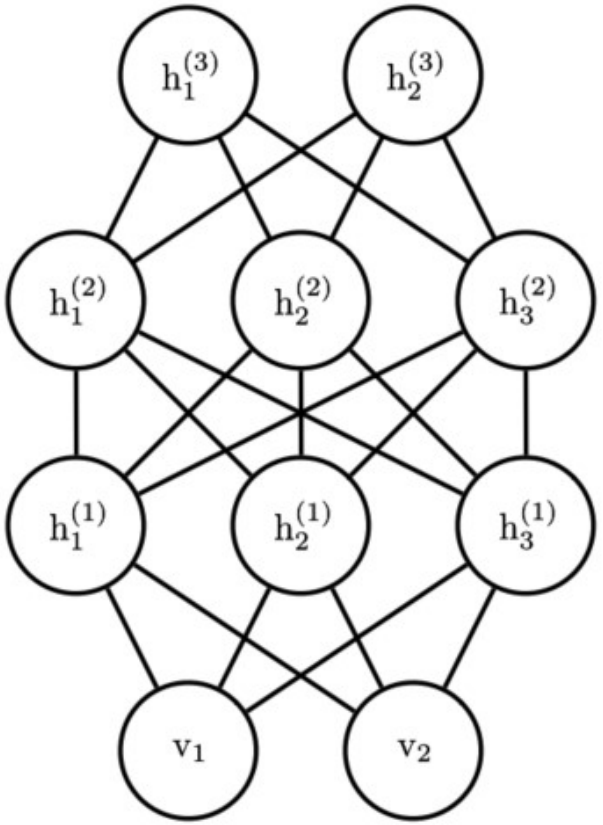
20.3 Deep Belief Networks

- Depois desse treinamento ainda se pode utilizar o wake-sleep para melhorar o modelo (não tão usual) e utilizá-lo como modelo generativo.
- Todavia o interesse principal é em usar os pesos aprendidos para inicializar os pesos de uma MLP. Depois realizando o “fine-tuning” dessa MLP numa tarefa classificatória.
  - É método heurístico, onde não necessariamente gerará um bom limite inferior vide a MLP só possuir fluxo ascendente de informações (das variáveis visíveis para as latentes).



DBN

20.4 Deep Boltzmann Machines



- Modelo inteiramente com ligações não-direcionadas.
- Porém, diferente da RBM, possui diversas camadas de variáveis latentes.
- Dentro de cada camada as variáveis são independentes.
- Considerando doravante que as variáveis são binárias.

- É um modelo baseado em energia, que se possuir 3 camadas obtêm a seguinte distribuição de probabilidade conjunta:

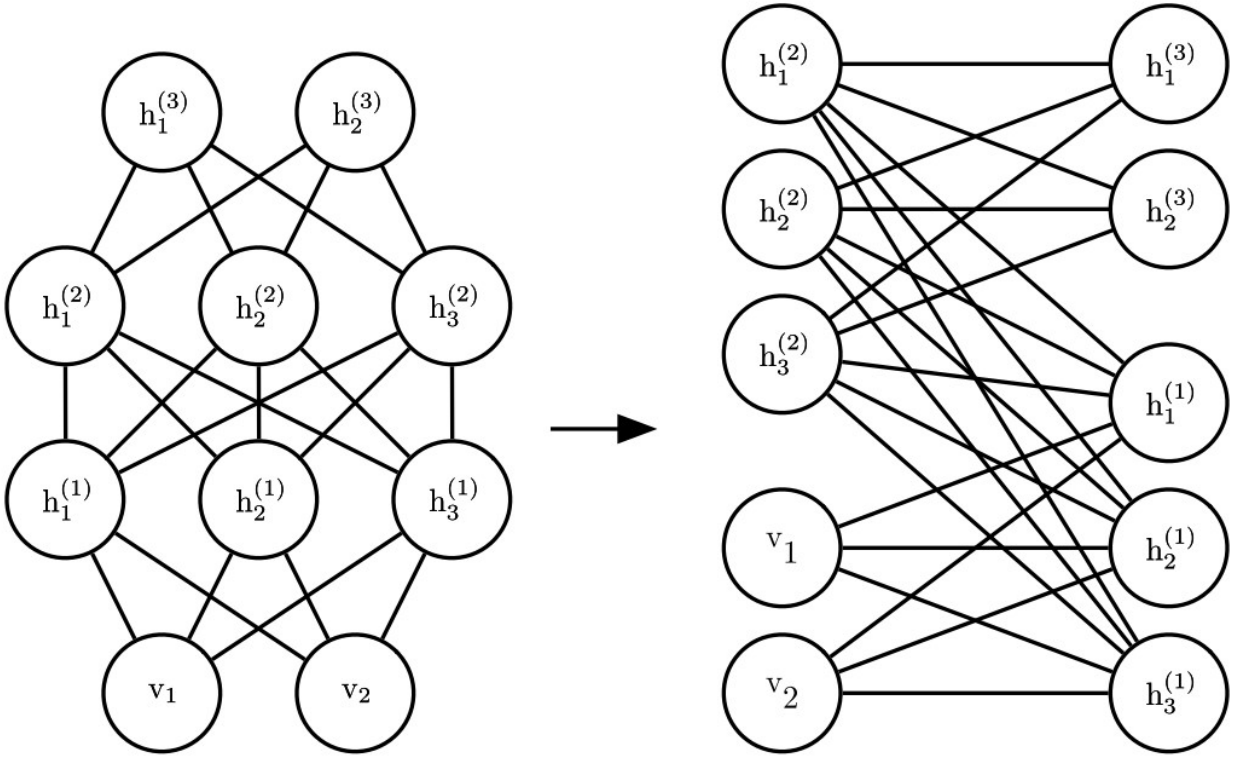
$$P\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\right) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta})\right)$$

$$E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta}) = -\boldsymbol{v}^{\top}\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)} - \boldsymbol{h}^{(1)\top}\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)} - \boldsymbol{h}^{(2)\top}\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)}.$$

20.4 Deep Boltzmann Machines

1. As camadas da DBM podem ser orgazinadas num grafo bipartido.

2. Com isso, quando condiciona-se nas variáveis de uma camada par, as camadas na camada ímpar tornam-se condicionalmente independentes. E vice-versa.



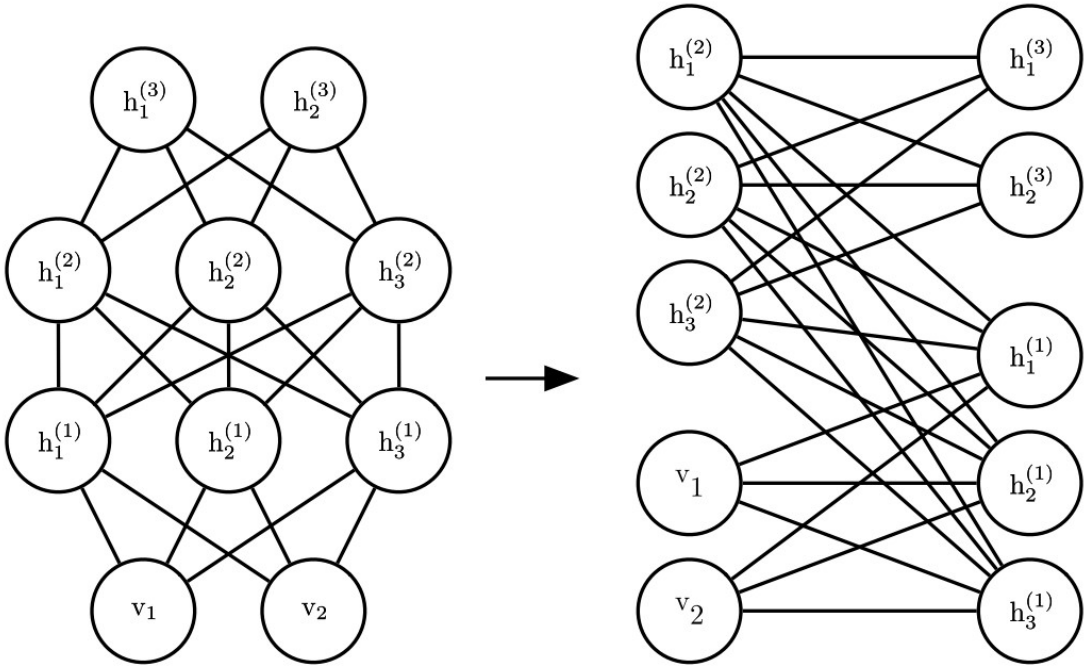
20.4 DBM

3. Logo, como dentro de cada camada cada variável é independente por definição da DBM, tem-se:

$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{W}^{(1)}_{i,:}\boldsymbol{h}^{(1)}\right)$$

$$P(h^{(1)}_i = 1 \mid \boldsymbol{v}, \boldsymbol{h}^{(2)}) = \sigma\left(\boldsymbol{v}^\top \boldsymbol{W}^{(1)}_{:,i} + \boldsymbol{W}^{(2)}_{i,:}\boldsymbol{h}^{(2)}\right)$$

$$P(h^{(2)}_k = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{h}^{(1)\top}\boldsymbol{W}^{(2)}_{:,k}\right)$$

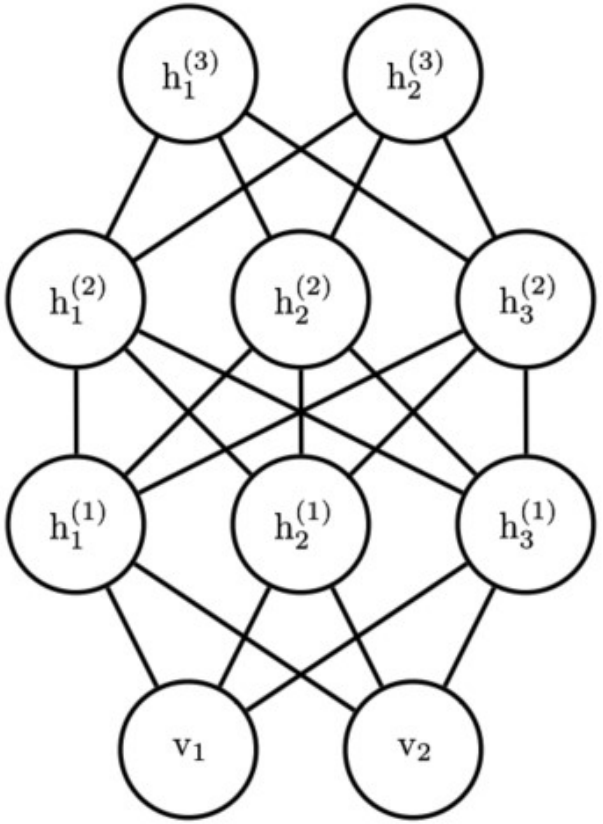


Agora se pode realizar amostagrem de Gibbs com uma iteração nas camadas pares e outras nas ímpares. Em vez de l+1 iterações.

20.4 Deep Boltzmann Machines

- Embora uma camada possua uma distruibuição fatorial quando condicionada com as outras camadas vizinhas, a distribuição sobre todas as camadas latentes condicionadas sobre as vísiveis **não fatora.**
- Portanto precisa-se de outros métodos para achar a distribuição posterior da DBM, i.e. $p(h^{(1)}, h^{(2)}, h^{(3)}|v)$.

- **Resolução:**
  - Aplicar na DBM o método de “mean field inference”, lembrem do capítulo 19 “Approximate Inference”



- **Resolução:**
  - Aplicar na DBM o método de “mean field inference”.

Para achar os melhores parâmetros dessas Bernollis aplica-se as “mean field equations” da seção 19.56:

Essas equações foram obtidas resolvendo aonde as derivadas do limite inferior variacional são zero.

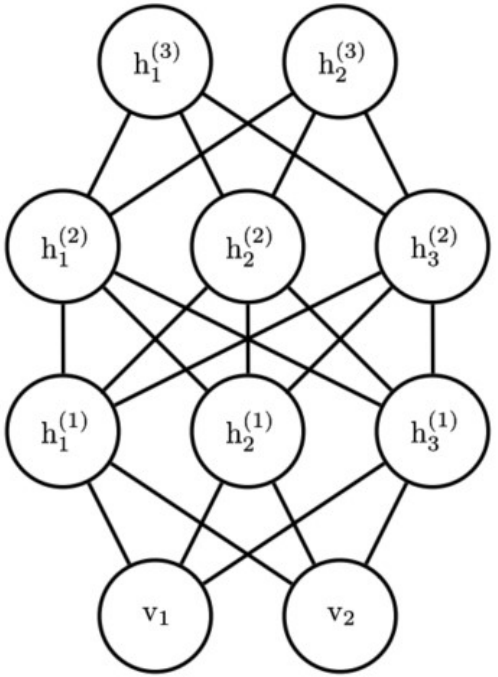
$$\tilde{q}(h_i \mid \boldsymbol{v}) = \exp\left(\mathbb{E}_{\mathbf{h}_{-i} \sim q(\mathbf{h}_{-i} \mid \boldsymbol{v})} \log \tilde{p}(\boldsymbol{v}, \boldsymbol{h})\right)$$

Aplicando essas equações obtêm-se regras de iteração dos parâmetros de Bernolli do Q que se está tentando encontrar:

$$\begin{cases} \hat{h}_j^{(1)} = \sigma\left(\sum_i v_i W_{i,j}^{(1)} + \sum_{k'} W_{j,k'}^{(2)} \hat{h}_{k'}^{(2)}\right), \quad \forall j, \\ \hat{h}_k^{(2)} = \sigma\left(\sum_{j'} W_{j',k}^{(2)} \hat{h}_{j'}^{(1)}\right), \quad \forall k. \end{cases}$$

Num ponto fixo dessas equações tem-se um máximo local do limite inferior variacional. **Obtendo-se Q.**

20.4 Deep Boltzmann Machines



Treinamento da DBM: deve-se enfrentar tanto o problema da função de partição quanto o problema do distribuição posterior.

- Condicional: usa-se o Q achado com o método de inferência aproximada mostrado.
- O aprendizado portanto segue maximizando o limite inferior variacional da *log-likelihood* dos dados.

$$\mathcal{L}(Q, \boldsymbol{\theta}) = \sum_i \sum_{j'} v_i W^{(1)}_{i,j'} \hat{h}^{(1)}_{j'} + \sum_{j'} \sum_{k'} \hat{h}^{(1)}_{j'} W^{(2)}_{j',k'} \hat{h}^{(2)}_{k'} - \log Z(\boldsymbol{\theta}) + \mathcal{H}(Q) < \log P(\boldsymbol{v}; \boldsymbol{\theta})$$

A função de partição é calculada com ajuda dos métodos do capítulo 18, nesse caso o *stochastic maximum likelihood* é um dos únicos que podem ser utilizados com bons resultados na prática.

Algoritmo 20.1: SML variacional para treinamento de uma DBM com duas camadas latentes

Set ϵ, the step size, to a small positive number
Set k, the number of Gibbs steps, high enough to allow a Markov chain of $p(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \boldsymbol{\theta} + \epsilon \Delta_{\boldsymbol{\theta}})$ to burn in, starting from samples from $p(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}; \boldsymbol{\theta})$.

Initialize three matrices, $\tilde{\boldsymbol{V}}$, $\tilde{\boldsymbol{H}}^{(1)}$, and $\tilde{\boldsymbol{H}}^{(2)}$ each with m rows set to random values (e.g., from Bernoulli distributions, possibly with marginals matched to the model's marginals).

while not converged (learning loop) **do**

Sample a minibatch of m examples from the training data and arrange them as the rows of a design matrix $\boldsymbol{V}$.

Initialize matrices $\hat{\boldsymbol{H}}^{(1)}$ and $\hat{\boldsymbol{H}}^{(2)}$, possibly to the model's marginals.

while not converged (mean field inference loop) **do**

$$\hat{\boldsymbol{H}}^{(1)} \leftarrow \sigma\left(\boldsymbol{V}\boldsymbol{W}^{(1)} + \hat{\boldsymbol{H}}^{(2)}\boldsymbol{W}^{(2)\top}\right).$$

$$\hat{\boldsymbol{H}}^{(2)} \leftarrow \sigma\left(\hat{\boldsymbol{H}}^{(1)}\boldsymbol{W}^{(2)}\right).$$

end while

Equações de ponto fixo para achar o Q da inferência aproximada mostrada acima.

$$\Delta_{\boldsymbol{W}^{(1)}} \leftarrow \frac{1}{m}\boldsymbol{V}^\top \hat{\boldsymbol{H}}^{(1)}$$

$$\Delta_{\boldsymbol{W}^{(2)}} \leftarrow \frac{1}{m}\hat{\boldsymbol{H}}^{(1)\top}\hat{\boldsymbol{H}}^{(2)}$$

for $l = 1$ to k (Gibbs sampling) **do**

Gibbs block 1:

$$\forall i, j, \tilde{V}_{i,j} \text{ sampled from } P(\tilde{V}_{i,j} = 1) = \sigma\left(\boldsymbol{W}^{(1)}_{j,:}\left(\tilde{\boldsymbol{H}}^{(1)}_{i,:}\right)^\top\right).$$

$$\forall i, j, \tilde{H}^{(2)}_{i,j} \text{ sampled from } P(\tilde{H}^{(2)}_{i,j} = 1) = \sigma\left(\tilde{\boldsymbol{H}}^{(1)}_{i,:}\boldsymbol{W}^{(2)}_{:,j}\right).$$

Gibbs block 2:

$$\forall i, j, \tilde{H}^{(1)}_{i,j} \text{ sampled from } P(\tilde{H}^{(1)}_{i,j} = 1) = \sigma\left(\tilde{\boldsymbol{V}}_{i,:}\boldsymbol{W}^{(1)}_{:,j} + \tilde{\boldsymbol{H}}^{(2)}_{i,:}\boldsymbol{W}^{(2)\top}_{j,:}\right).$$

end for

Gera-se amostras do **modelo atual** utilizando a amostragem de Gibbs em bloco.

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}; \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}).$$

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

Lembrete: Não se tem p diretamente mas sim o limite inferior variacional.

$$\Delta_{\boldsymbol{W}^{(1)}} \leftarrow \Delta_{\boldsymbol{W}^{(1)}} - \frac{1}{m}\boldsymbol{V}^\top \tilde{\boldsymbol{H}}^{(1)}$$

$$\Delta_{\boldsymbol{W}^{(2)}} \leftarrow \Delta_{\boldsymbol{W}^{(2)}} - \frac{1}{m}\tilde{\boldsymbol{H}}^{(1)\top}\tilde{\boldsymbol{H}}^{(2)}$$

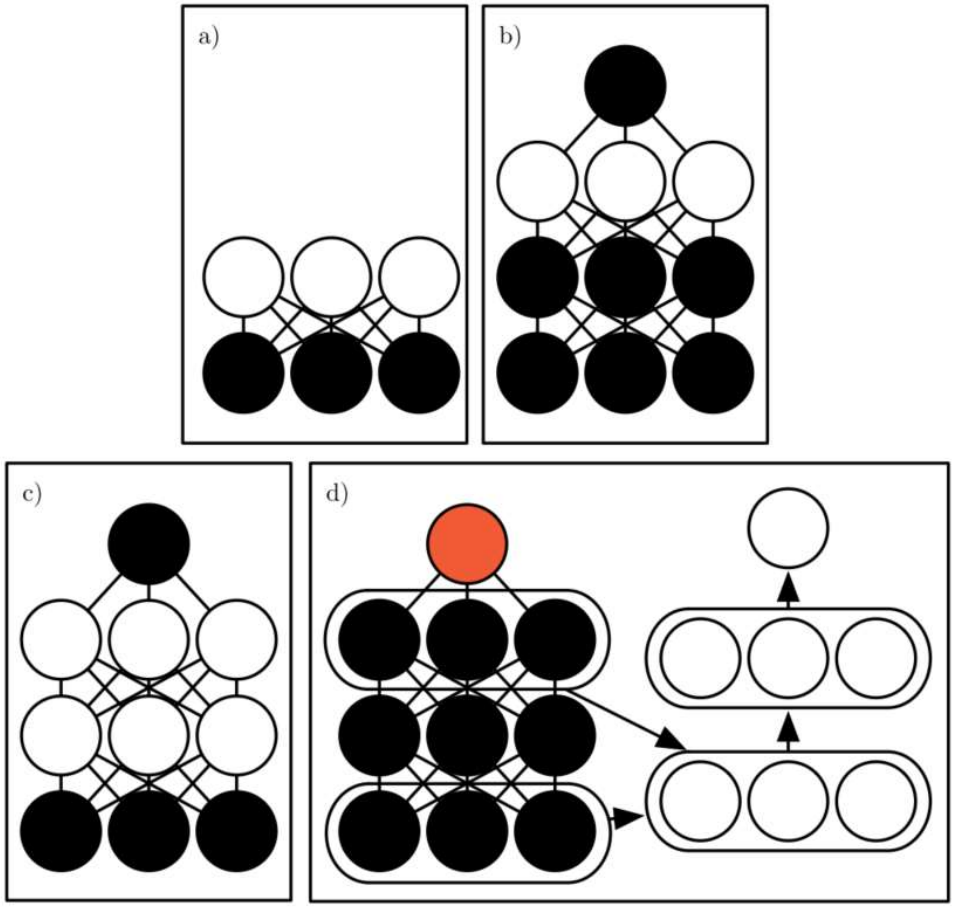
$$\boldsymbol{W}^{(1)} \leftarrow \boldsymbol{W}^{(1)} + \epsilon\Delta_{\boldsymbol{W}^{(1)}}$$ (this is a cartoon illustration, in practice use a more effective algorithm, such as momentum with a decaying learning rate)

$$\boldsymbol{W}^{(2)} \leftarrow \boldsymbol{W}^{(2)} + \epsilon\Delta_{\boldsymbol{W}^{(2)}}$$

end while

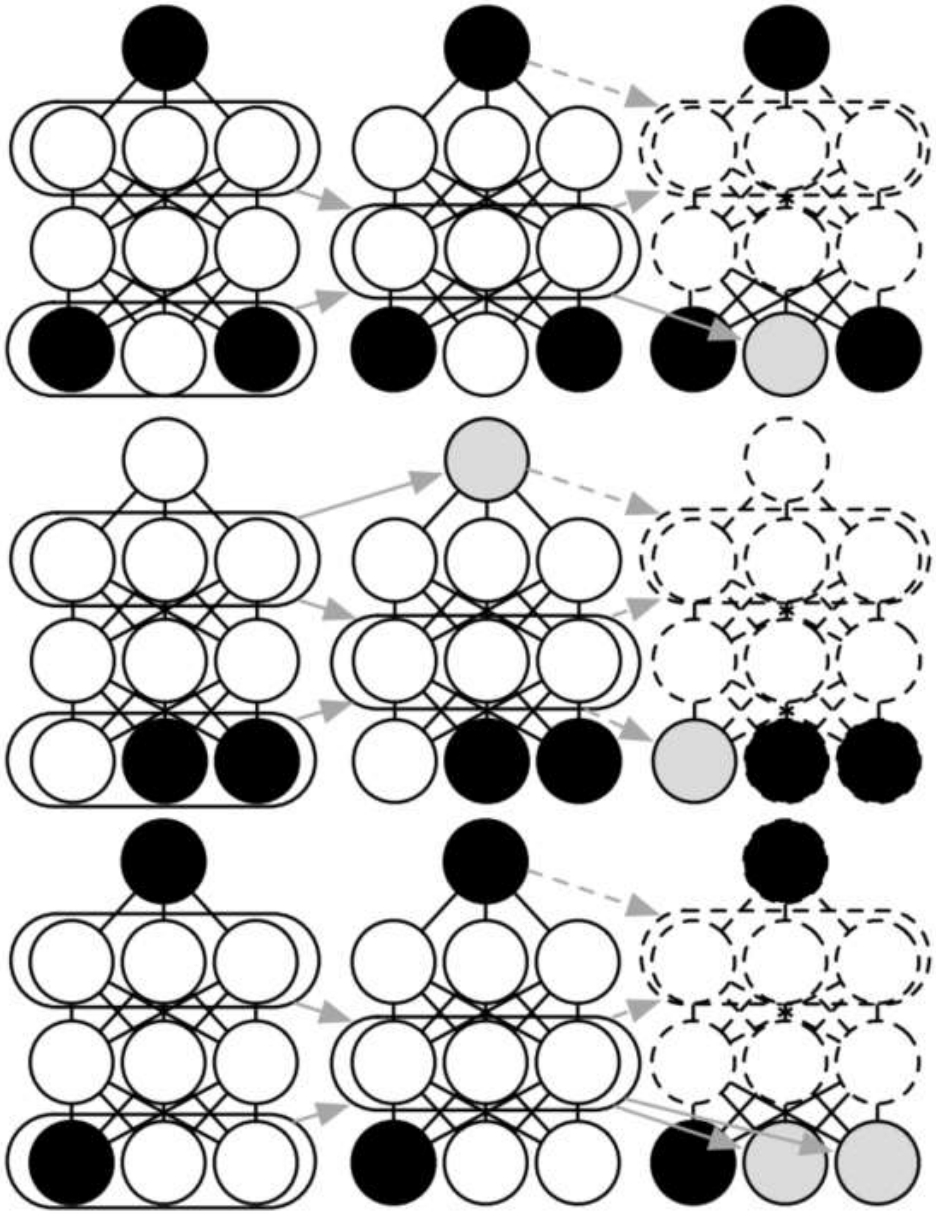
20.4 Deep Boltzmann Machines

- **Pré-treino por camadas:** aplicar o algoritmo 20.1 com inicialização randômica não funciona bem. Para melhorar os resultados treinam-se várias RBMs parecido com o jeito de se treinas as DBN, ou seja, empilha-se várias RBMs (mudando-se alguns parâmetros). Depois aplica-se PCD na DBM inteira:

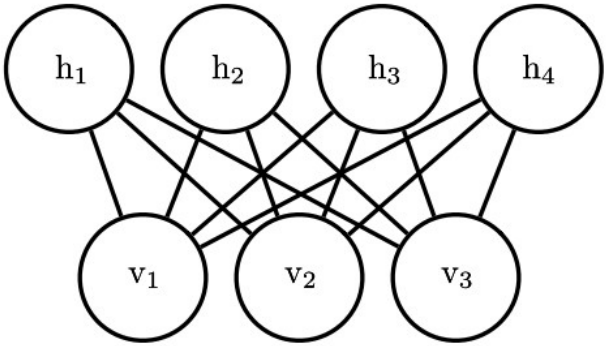


20.4 Deep Boltzmann Machines

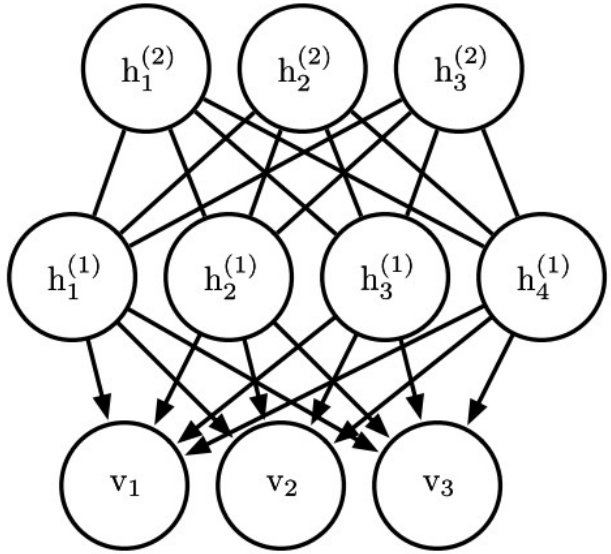
- Realizar classificação com DBMs treinados via empilhamento de RBMs tem certos pontos negativos:
  - Difícil de acompanhar performance
  - Muitos pedaços do algoritmo para hiperparametrizar
  - O MLP não possui as vantagens do DBM
- Duas soluções:
  - *Centered* DBM: repametrização do modelo para construir uma Hessiana melhor condicionada no início do treinamento.
  - *Multi-prediction* DBM: enxerga as "mean field equations" do algoritmo de inferência aproximada como definindo uma família de redes recorrentes para resolução de qualquer problema de inferência possível.



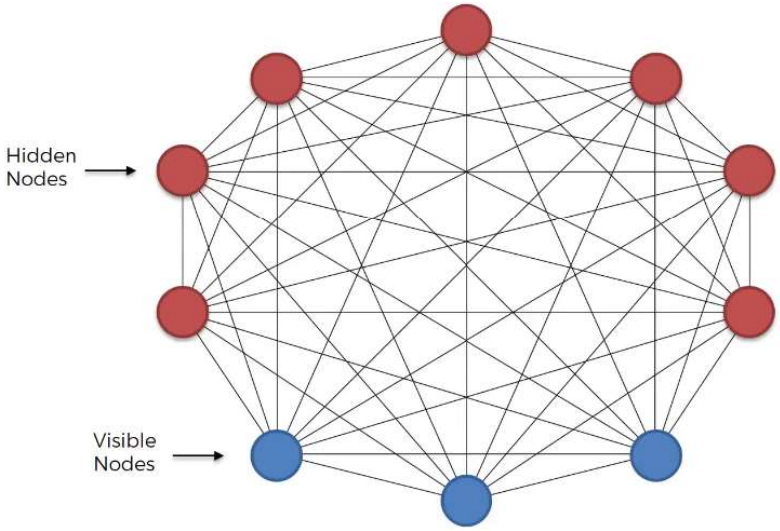
Recapitulando as diferenças entre DBN, RBM e DBM



Restricted Boltzmann Machine

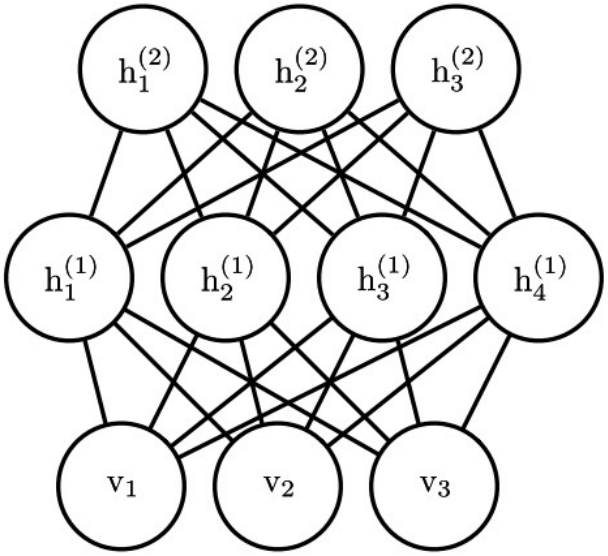


Deep Belief Network



Boltzmann Machine

Diferentes: note a ausência de ligações dentro das camadas da DBM



Deep Boltzmann Machine

20.5 Boltzmann Machines for Real-Valued Data

- Deseja-se usar o modelo RBM ou DBM para **variáveis contínuas**, encaixando melhor em problemas de imagem e áudio.
- **A primeira maneira** é ter as variáveis visíveis contínuas e as latentes binárias. Possuindo condicional sobre as visíveis dada as latentes no formato de uma Gaussiana: $p(\boldsymbol{v} \mid \boldsymbol{h}) = \mathcal{N}(\boldsymbol{v}; \boldsymbol{W}\boldsymbol{h}, \boldsymbol{\beta}^{-1})$
  - Para isso, depois de certas manipulações tem-se que a função de energia do RBM toma a seguinte forma:

$$E(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{2}\boldsymbol{v}^\top(\boldsymbol{\beta} \odot \boldsymbol{v}) - (\boldsymbol{v} \odot \boldsymbol{\beta})^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{h}$$

20.5 Boltzmann Machines for Real-Valued Data

- Porém, a Gaussiana não modela a informação das covariâncias condicionais o que produz um viés indutivo ruim para, por exemplo, imagens naturais pois **a relação entre os pixels tende a ser mais informativa que seus valroes absolutos**.
- Modelos alternativos que tratam esse problema foram propostos, dentre eles:
  - *Mean and Covariance* RBM (mcRBM)
  - *Mean Product of Student t-distributions* (mPoT)
  - *Spike and Slab Restricted Boltzmann Machines* (ssRBMs)

Mean and Covariance RBM (mcRBM)

- Utiliza suas camadas latentes para codificar a média e covariância condicional de maneira independente:

$$E_{\text{mc}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}, \boldsymbol{h}^{(c)}) = E_{\text{m}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}) + E_{\text{c}}(\boldsymbol{x}, \boldsymbol{h}^{(c)})$$

A Gaussiana-Bernolli RBM de antes

$$E_{\text{m}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} - \sum_j \boldsymbol{x}^\top \boldsymbol{W}_{:,j} h_j^{(m)} - \sum_j b_j^{(m)} h_j^{(m)}$$

Novo termo que modela as covariâncias condicionais

$$E_{\text{c}}(\boldsymbol{x}, \boldsymbol{h}^{(c)}) = \frac{1}{2}\sum_j h_j^{(c)} \left(\boldsymbol{x}^\top \boldsymbol{r}^{(j)}\right)^2 - \sum_j b_j^{(c)} h_j^{(c)}$$

Obtêm-se:

$$p_{\text{mc}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}, \boldsymbol{h}^{(c)}) = \frac{1}{Z}\exp\left\{-E_{\text{mc}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}, \boldsymbol{h}^{(c)})\right\}$$ Função de energia combinada

$$p_{\text{mc}}(\boldsymbol{x} \mid \boldsymbol{h}^{(m)}, h^{(c)}) = \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{C}^{\text{mc}}_{\boldsymbol{x}|\boldsymbol{h}}\left(\sum_j \boldsymbol{W}_{:,j} h_j^{(m)}\right), \boldsymbol{C}^{\text{mc}}_{\boldsymbol{x}|\boldsymbol{h}}\right)$$ Condicional das observações dada as variáveis latentes h

Mean Product of Student t-distributions (mPoT)

- Parecido com o mcRBM porém a distribuição condicional complementar sobre as variáveis latentes é dada por distribuições Gamma condicionalmente independente:

$$E_{\mathrm{mPoT}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}, \boldsymbol{h}^{(c)}) = E_{\mathrm{m}}(\boldsymbol{x}, \boldsymbol{h}^{(m)}) + E_{\mathrm{c}}(\boldsymbol{x}, \boldsymbol{h}^{(c)})$$

$$= E_m(\boldsymbol{x}, \boldsymbol{h}^{(m)}) + \sum_j \left(h_j^{(c)} \left(1 + \frac{1}{2} \left(\boldsymbol{r}^{(j)\top} \boldsymbol{x} \right)^2 \right) + (1 - \gamma_j) \log h_j^{(c)} \right)$$

Agora são distribuições Gamma

Spike and Slab Restricted Boltzmann Machines (ssRBMs)

- Outro modo de modelar a covariância estrutural de dados com valor nos reais.
- Não precisa de inversão matricial ou métodos Monte Carlo Hamiltonianos como as supracitadas mcRBMs.
- Possui dois conjuntos de variáveis latentes: binárias *spike* e reais *slab*. A média das variáveis visíveis condicionadas as latentes é dada por $(\boldsymbol{h} \odot \boldsymbol{s})\boldsymbol{W}^\top$. Logo h seleciona uma coluna de W. Portanto, as variáveis *spike* determinam se aquele componente estará presente, já a variável *slab* correspondente determina a intensidade do componente se ele estiver presente.

Função de energia do modelo

$$E_{\text{ss}}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{h}) = -\sum_i \boldsymbol{x}^\top \boldsymbol{W}_{:,i} s_i h_i \; + \frac{1}{2}\boldsymbol{x}^\top \left(\boldsymbol{\Lambda} + \sum_i \boldsymbol{\Phi}_i h_i\right) \boldsymbol{x}$$
$$+ \frac{1}{2}\sum_i \alpha_i s_i^2 \; - \sum_i \alpha_i \mu_i s_i h_i \; - \sum_i b_i h_i + \sum_i \alpha_i \mu_i^2 h_i$$

Variações convolucionais da ssRBM obtêm ótimos resultados na produção de amostras de imagens naturais

20.6 Convolutional Boltzmann Machines

- Dados com alta dimensão como imagens são pesados computationalmente, por isso, criam-se operações como convoluções discretas com kernels pequenos para explorar a estrutura espacial ou temporal que seja invariante a translações.
- Alem desses filtros é necessário operações de pooling para diminuição do tamanho sucessivo das camadas: **como definir pooling para modelos baseados em energia?**
- **Solução: *probabilistic max pooling*** (Lee, 2009).
  - Forçar a condição que as unidades de detecção (vetor binário $d \in \mathbb{R}^4$) possuam no máximo uma unidade atividade num determinado momento.
    - $[0,0,0,1]$, $[0,0,1,0]$, $[0,1,0,0]$, $[1,0,0,0]$, $[0,0,0,0]$
    - Onde o max-pooling p é sempre 1 quando qualquer dessas unidades forem 1, e zero no caso $[0,0,0,0]$
    - Em vez de ter de normalizar para 2^4combinações, tem-se apenas 5.
  - Ainda pior desempenho que CNNs supervisionadas.
  - Possuem certos problemas na adaptabilidade com inputs de tamanhos diversos.

20.7 Boltzmann Machines for Structured or Sequential Outputs

- Quando se deseja prever baseado num x algum y com **estrutura**, onde diferentes pedaços do saída são relacionados e subordinados com outros pedaços.
- Exemplo: síntese de fala: o y possui rica estrutura como forma de onda.
- Para isso pode-se usar uma distribuição de probabilidade condicional: $p(y|x)$.
  - Outro exemplo: modelar sequências, i.e. estimar $p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\tau)})$, para isso RBMs condicionais modelam $p(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)})$
- Outra maneira de modelar sequências como: $p(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(t-m)})$ é modelar apenas $p(\boldsymbol{x}^{(t)})$ numa RBM, mas introduzir a informação dos tempos anteriores nos parâmetros *bias* os quais tornam-se uma função linear das ***m*** últimas variáveis. Quase uma nova RBM para cada $x^{(t)}$ quando se muda o valor das variáveis passadas.
- Ainda outra maneira: usar uma RNN para emitir todos os parâmetros de um RBM para cada passo no tempo.

20.8 Other Boltzmann Machines

- Pode-se treinar BMs para objetivos discriminativos $\log p(y|\boldsymbol{v})$, utilizando um treinamento que possua uma combinação linear de critérios discriminativos e generativos.
  - Porém acabam sendo menos poderosas que MLPs supervisionadas com as tecnologias presentes.
- RBMs são modelos ricos com diversas possibilidades de mudanças de suas suposições como por exemplo:
  - Modelar em sua função de energia relações de ordem superior como ligações envolvendo a multiplição de várias variáveis, pense nos monômios de ordem 3.
  - Um Modelo possuir dois conjuntos de unidades latentes, um interagindo com $\boldsymbol{v}$ e y e outro apenas com $\boldsymbol{v}$.
- Embora precisem de mais estudo para aplicação - diferente de uma simples re-definição de uma camada em um modelo MLP – o campo ainda é aberto para novas metodologias.

20.9 Back-Propagation through Random Operations

- Redes tradicionais produzem uma saída deterministica dada uma entrada x, porém no campo generativo é desejável obter certo nível de **estocasticidade** nessa transformação.
- Um modo de se realizar isso é simplesmente alimentar a rede com um input de ruído Gaussiano a mais, i.e. um z, mudando a rede para uma função $f(x, z)$ que pode sofrer retro-propagação normalmente como antes.
- Exemplo:

$$\mathrm{y} \sim \mathcal{N}(\mu, \sigma^2) \longrightarrow y = \mu + \sigma z \; ; \; \mathrm{z} \sim \mathcal{N}(z; 0, 1)$$

- Depois fica fácil incorporar essa operação de amostragem num grafo computational maior, podendo-se calcular as perdas em y. Além disso, a média e a variância pode ser o output de uma rede, i.e. $\mu = f(\boldsymbol{x}; \boldsymbol{\theta})$, $\sigma = g(\boldsymbol{x}; \boldsymbol{\theta})$.
- **Princípio geral:** *reparametrization trick, stochastic back-propagation, or perturbation analysis*:

$$\mathbf{y} \sim p(\mathbf{y} \mid \boldsymbol{\omega}) \quad \Longleftrightarrow \quad \boldsymbol{y} = f(\boldsymbol{z}; \boldsymbol{\omega})$$

20.9 Back-Propagation through Random Operations

- E se o valor do meu output da minha função já “estocasticizada” for discreta?

$$\boldsymbol{y} = f(\boldsymbol{z}; \boldsymbol{\omega})$$

- O valor da derivada é zero um muitos lugares, não se possui informação suficiente para atualizar os paramêtros via um $J(y)$.
- Solução: algoritmo REINFORCE (**RE**ward Increment = nonnegative **F**actor × **O**ffset **R**einforcement × **C**haracteristic **E**ligibility):
  - Em vez de derivar $J(f(\boldsymbol{z}; \boldsymbol{\omega}))$, usa-se a função bem comportada e que permite gradiente descendente: $\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} J\left(f(\boldsymbol{z}; \boldsymbol{\omega})\right)$.
  - Como?

20.9 Back-Propagation through Random Operations

- Como?

$$\mathbb{E}_z[J(\boldsymbol{y})] = \sum_{\boldsymbol{y}} J(\boldsymbol{y}) p(\boldsymbol{y}),$$

$$\frac{\partial \log p(\boldsymbol{y})}{\partial \boldsymbol{\omega}} = \frac{1}{p(\boldsymbol{y})} \frac{\partial p(\boldsymbol{y})}{\partial \boldsymbol{\omega}}$$

$$\frac{\partial \mathbb{E}[J(\boldsymbol{y})]}{\partial \boldsymbol{\omega}} = \sum_{y} J(\boldsymbol{y}) \frac{\partial p(\boldsymbol{y})}{\partial \boldsymbol{\omega}}$$

$$= \sum_{y} J(\boldsymbol{y}) p(\boldsymbol{y}) \frac{\partial \log p(\boldsymbol{y})}{\partial \boldsymbol{\omega}}$$

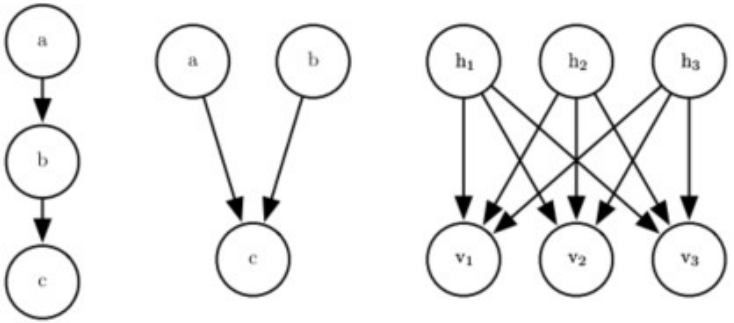
$$\approx \frac{1}{m} \sum_{\mathbf{y}^{(i)} \sim p(\boldsymbol{y}),\, i=1}^{m} J(\boldsymbol{y}^{(i)}) \frac{\partial \log p(\boldsymbol{y}^{(i)})}{\partial \boldsymbol{\omega}}$$

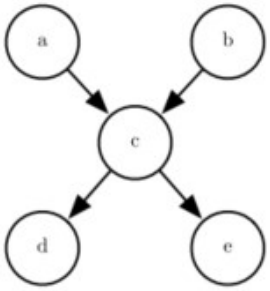
Existem alguns métodos para diminuir a variância dessa estimativa com a utilização de baselines $b(w)$.

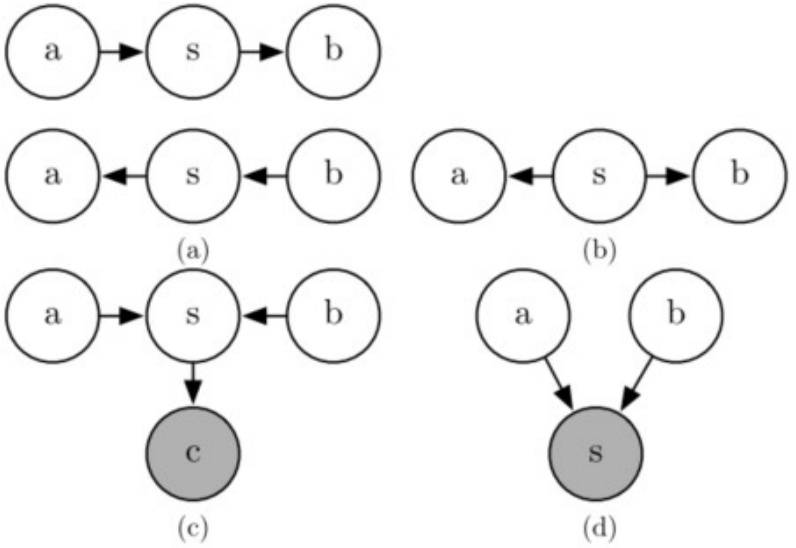
Estimativa Monte Carlo não-enviesada

20.10 Directed Generative Nets

- Até cerca de 2013, modelos gráficos direcionados foram menos populares na comunidade de aprendizado profundo. Os tópicos seguintes apresentam esses modelos pelo ponto de vista dessa comunidade.







20.10.1 Sigmoid Belief Networks

- Estrutura direcionada com uma probabilidade condicional especifica, sendo s um vetor binária numa camada tal, a probabilidade de uma de suas variáveis dados os seus ancestrais toma a seguinte forma:

$$p(s_i) = \sigma\left(\sum_{j<i} W_{j,i} s_j + b_i\right)$$

- Similar à Deep Belief Network, porém sem o teto de RBM.
- Aproximador universal de distribuições de probabilidade sobre a camada visível.
- Amostragem fácil: uma passada de amostragem ancestral.
- Inferência difícil: camada latente dadaas visíveis é intratável, inclusive usando aproximações variacionais, lembre do problema dos cliques.
- Porém metodos recentes como *importance sampling*, *reweighted wake-sleep* (Bornschein and Bengio, 2015) e *bidirectional Helmholtz machines* (Bornschein et al., 2015) conseguem treinar esse modelo de maneira a obter boa performance.

20.10.2 Differentiable Generator Networks

- Modelos desse tipo transformam amostras de uma variável latente z para amostras ou distribuição de x utilizando uma função diferenciável $g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)})$.
- Essas redes geradoras podem ser interpretadas como algoritmos de geração de amostras de acordo com a distribuição que se deseja, onde os parâmetros dessa arquitetura selecionam uma dentre todas as distribuições desse espaço.
- Exemplo, amostrar de uma distribuição normal:

$$\boldsymbol{x} = g(\boldsymbol{z}) = \mu + \boldsymbol{L}\boldsymbol{z}$$

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$$

- De modo mais geral g irá selecionar uma transformação mais complicada:

$$p_z(\boldsymbol{z}) = p_x(g(\boldsymbol{z}))\left|\det\left(\frac{\partial g}{\partial \boldsymbol{z}}\right)\right| \longrightarrow \text{implicitamente} \quad p_x(\boldsymbol{x}) = \frac{p_z(g^{-1}(\boldsymbol{x}))}{\left|\det\left(\frac{\partial g}{\partial \boldsymbol{z}}\right)\right|}$$

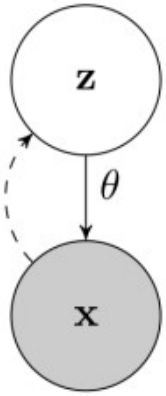
20.10.3 Variational Autoencoders

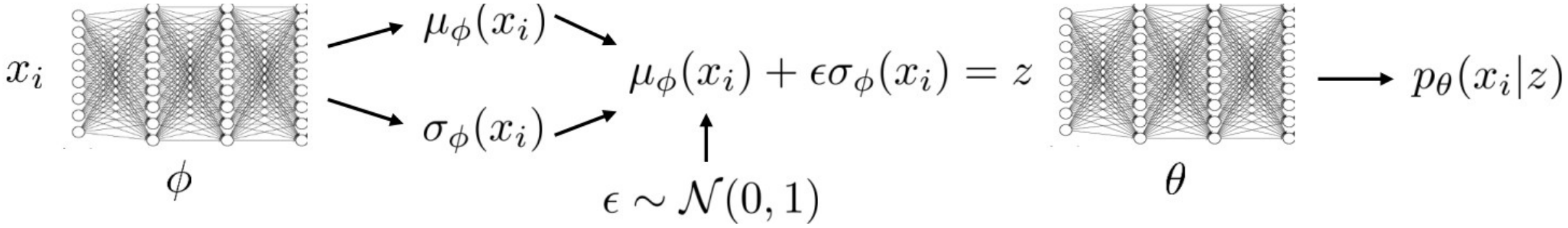
- Modelo direcionado que utiliza os métodos de inferência aproximada (capítulo 19) e pode ser treinado via métodos baseados em gradiente descendente.
- Essencialmente tenta-se aumentar o limite inferior da *log-likelihood* dos dados via uma parametrização eficiente do posterior q, e uma posterior aplicação de gradiente descendente em $L(p, q)$ para os parâmetros de p e q.

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{z}, \boldsymbol{x}) + \mathcal{H}(q(\mathbf{z} \mid \boldsymbol{x})) \\ &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{z}) - D_{\text{KL}}(q(\mathbf{z} \mid \boldsymbol{x})||p_{\text{model}}(\mathbf{z})) \\ &\leq \log p_{\text{model}}(\boldsymbol{x})\end{aligned}$$

20.10.3 Variational Autoencoders

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x)) \qquad p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$$

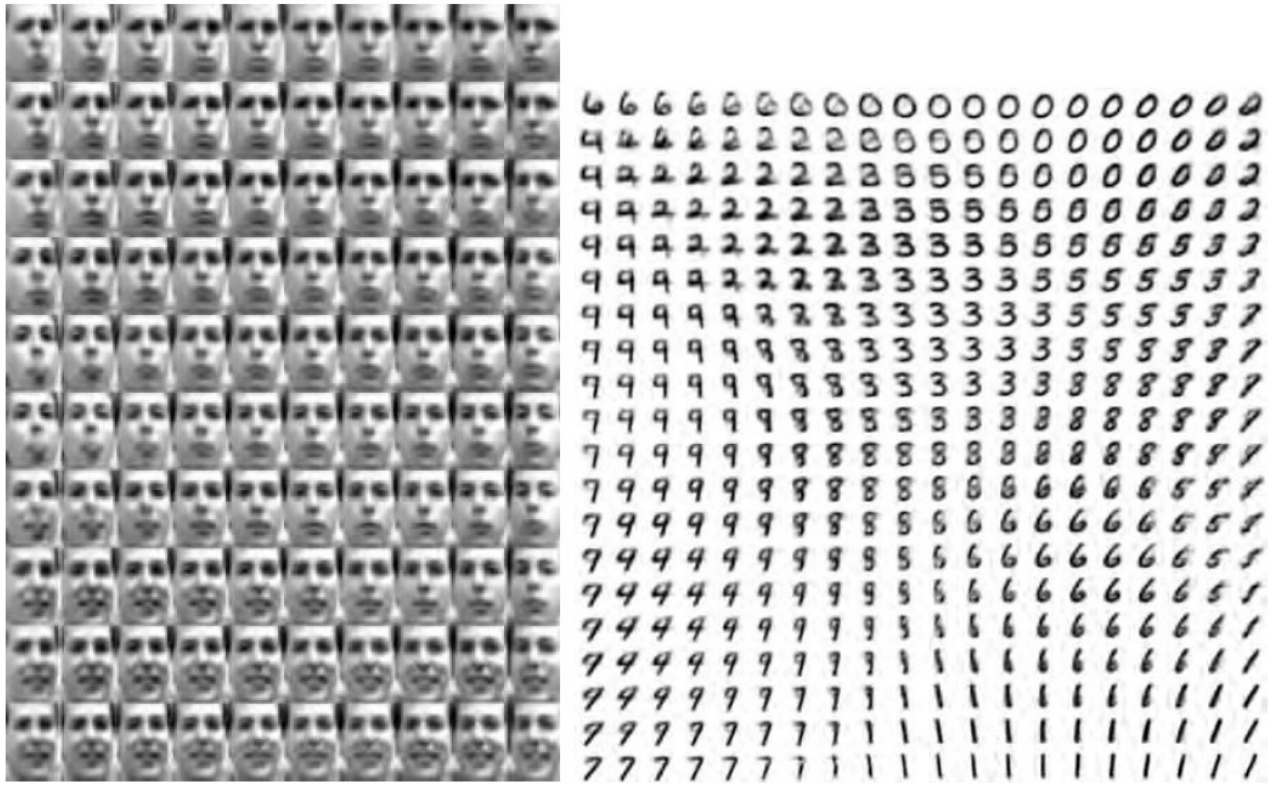




$$\max_{\theta,\phi} \frac{1}{N} \sum_i \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

Figuras retiradas dos slides do curso CS 285-2019

20.10.3 Variational Autoencoders



Imagens geradas pelo código latente z em 2-D via p(x|z) (Kingma and Welling, 2014a). Esquerda: amostradas geradas dos conjunto de dados *Frey faces*. Direita: amostras geradas com o MNIST.

20.10.4 Generative Adversarial Networks

- Baseadas num cenário adversarial, entre uma rede geradora e outra rede discriminatória. Ambas podendo ser redes direcionais, onde a geradora usa o truque da reparametrização para gerar com estocasticidade.

$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x} \sim p_{\text{model}}} \log (1 - d(\boldsymbol{x}))$$

$$g^* = \arg \min_g \max_d v(g, d)$$

20.10.4 Generative Adversarial Networks

DCGAN: amostras de quartos (Radford, 2015). Dados LSUN.

LAPGAN: amostras de igrejas (Denton, 2015). Dados LSUN.

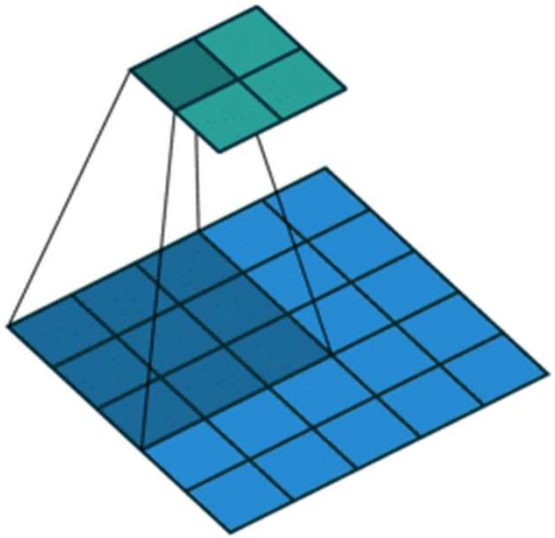
Será melhor explicada na próxima apresentação.

20.10.5 Generative Moment Matching Networks

- Modelo generativo que usa geradores diferenciáveis.
- Diferente das VAEs e GANs não possui um rede pareada.
- Utilizam o método de *moment matching*: treinar o modelo de tal forma que as estatísticas de suas amostras sejam o mais similar possível com as estatísticas das amostras dos dados.
  - Deve-se casar os momentos dos dados: $\mathbb{E}_{\boldsymbol{x}} \Pi_i x_i^{n_i}$; $\boldsymbol{n} = [n_1, n_2, \ldots, n_d]^\top$.
  - Todavia o número de combinações das features em momento cresce muito rápido. Para isso usa-se a função de custo chamada *maximum mean discrepancy*, ou MMD (Schölkopf and Smola, 2002; Gretton et al., 2012): mede o erro nos primeiros momentos em um espaço de dimensão infinita, utilizando um mapa implícito para o espaço das features definido via uma função kernel (parecido com a matemática dos SVMs). MMD é zero se as duas distribuições são iguais.
  - Mesmo com MMD as amostras são de baixa qualidade. Uma melhora pode ser obtida utilizando um autoencoder para gerar um espaço latente com o qual a rede será treinada.

20.10.6 Convolutional Generative Networks

- A rede generativas podem precisar de convoluções “inversas” que aumentem os vetores gradativamente até que no fim da rede em vez de aparecer uma classificação ou outra informação discrimanatória deve-se aparecer uma imagem inteira, possuindo estrutura interna como textura e luminosidade corretas.
- Para isso se define algumas operações específicas como o inverso de uma operação de pooling e a convolução transposta.



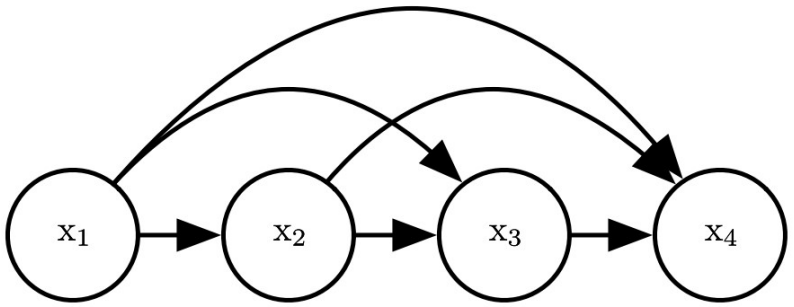
Fonte da imagem:
http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

20.10.7 Auto-Regressive Networks

- São modelos probabilísticos direcionados sem variáveis latentes.
- As distribuições condicionais são representadas por redes neurais.
- A estrutura do grafo é o grafo completo. Ou seja, utilizando a regra da cadeia das probabilidades, tendo V variáveis, pode-se escrever:

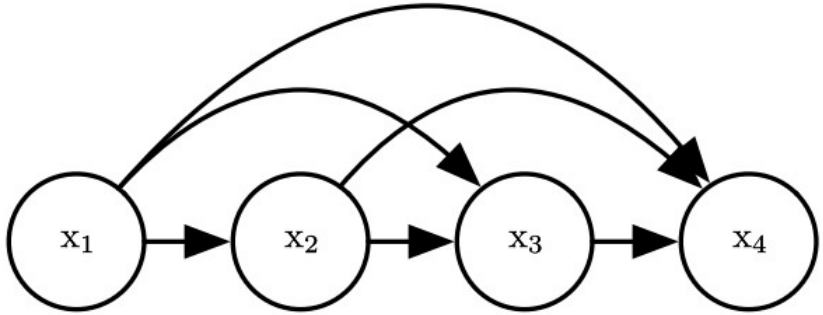
$$p(x_{1:V}) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_1, x_2, x_3)\ldots p(x_V|x_{1:V-1})$$

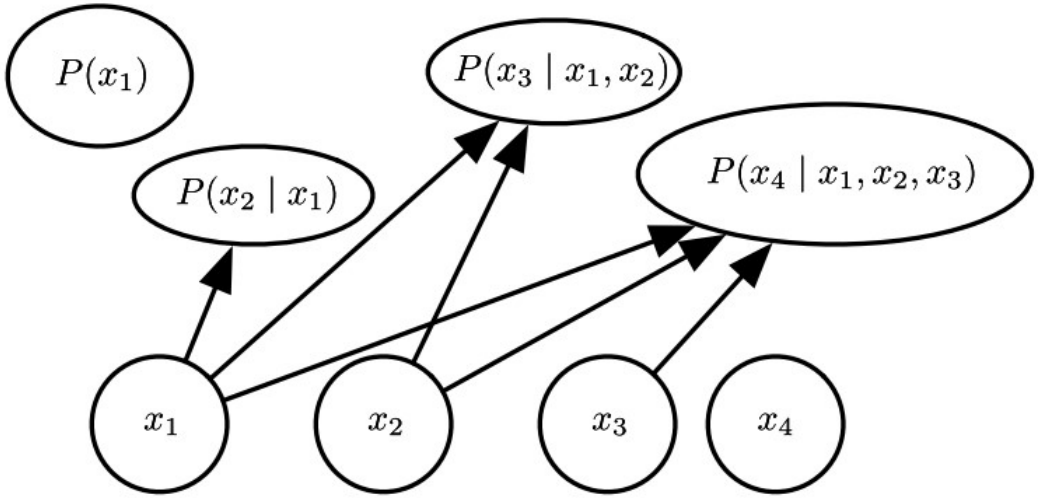
Cada uma dessas condicionais pode ser uma rede neural



20.10.8 Linear Auto-Regressive Networks

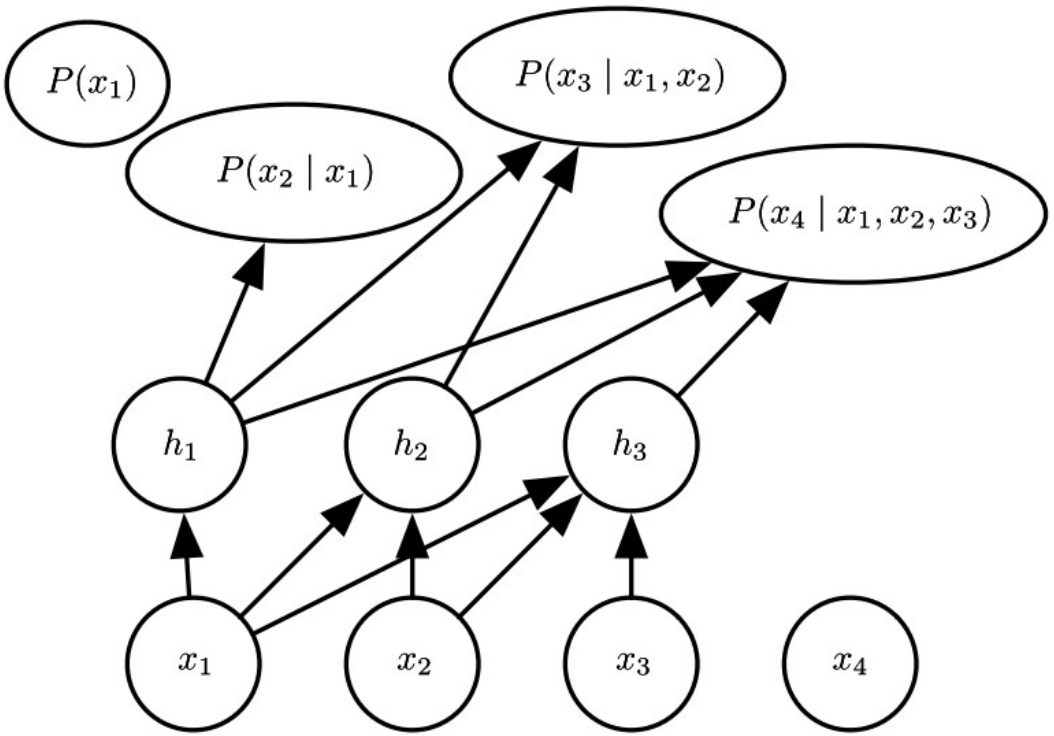
- Modelo auto-regressivo mais simples possível: sem camadas intermediárias e sem compartilhamento de parâmetros.
- Cada condicional é um modelo linear diferente.
- Possui $O(V^2)$ parâmetros, onde V é o número de variáveis.
- São a generalização direta de modelos lineares para o paradigma generativo.





20.10.9 Neural Auto-Regressive Networks

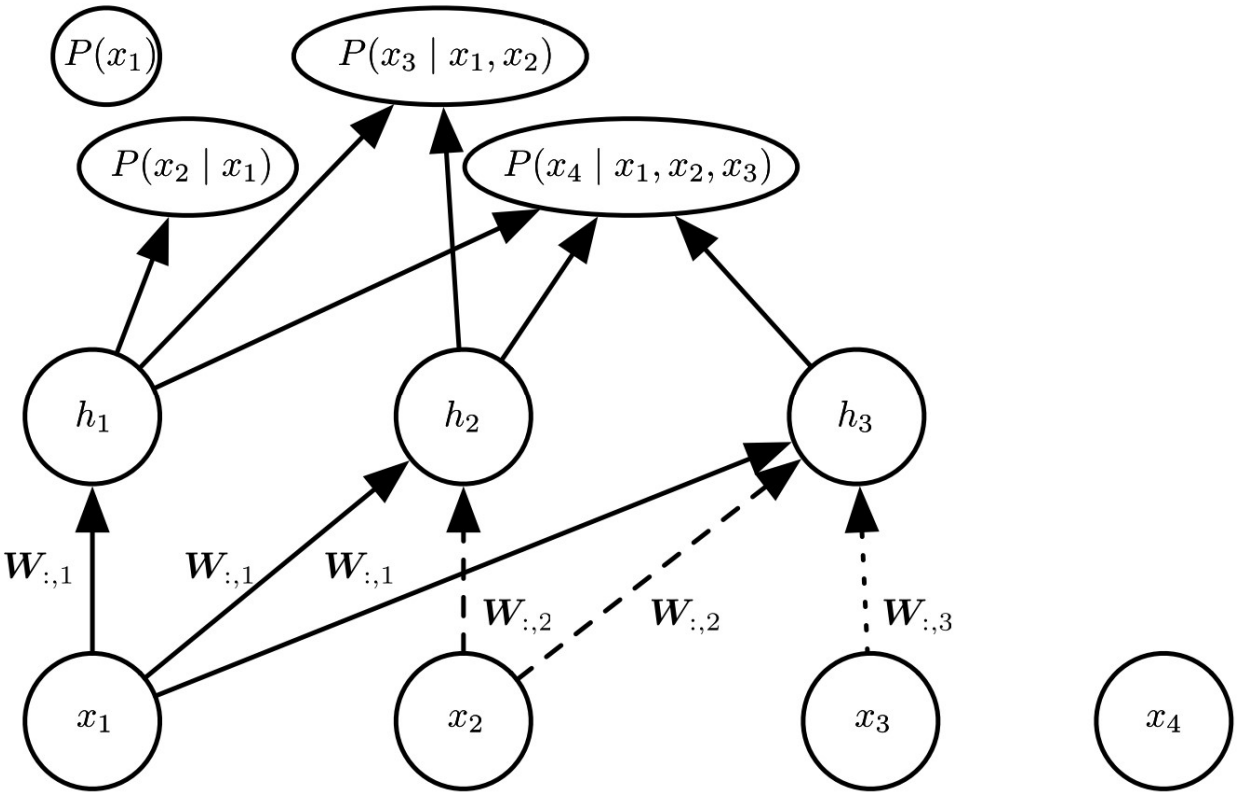
- Modelo auto-regressivo com parametrização de cada condicional mais poderosa, e utiliza compartilhamento de parâmetros.
- Dois pontos positivos:
  1. A Parametrização de cada condicional via uma rede neural diminui o número de paramêtros exponencialmente, ou seja, não precisa enumerar todas combinações das variáveis.
  2. Usa-se a conectividade esquerda-para-direita para juntar todas as redes numa só. As features da camada oculta utilizada para prever x_i podem ser reusadas para prever $x_{i+k}, k > 0$. Aplica o conceito de reuso.



20.10.10 NADE: neural auto-regressive density estimator

- Modelo auto-regressivo com bom desempenho.
- Os parâmetros das unidades ocultas de grupos *j* são compartilhados, criando um maior compartilhamento de recursos que as arquiteturas auto-regressivas anteriores.

$$W'_{j,k,i} = W_{k,i}$$



20.11 Drawing Samples from Autoencoders

- Bengio et al. (2013c) mostra como construir uma cadeia de Markov para o modelo *generalized denoising autoencoders* (lembrem do capítulo 14).

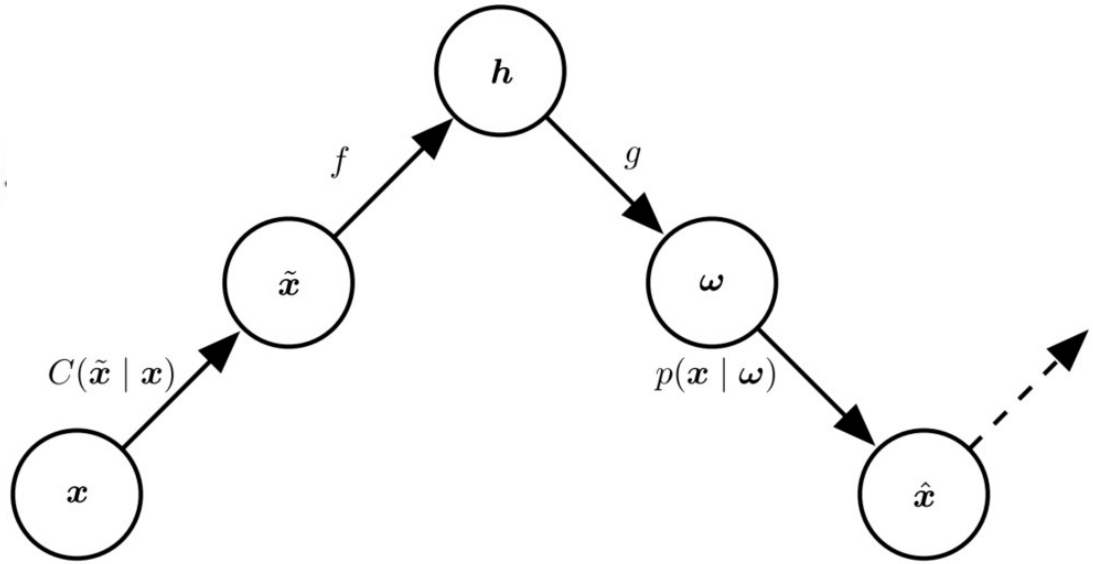
1. Injete ruído corrompedor, amostrando $\tilde{x}$.
2. Codificar $\tilde{x}$ dentro $h = f(\tilde{x})$.
3. Decodifique h para obter os parâmetros:

$$\boldsymbol{\omega} = g(\boldsymbol{h}) \quad p(\mathbf{x} \mid \boldsymbol{\omega} = g(\boldsymbol{h})) = p(\mathbf{x} \mid \tilde{\boldsymbol{x}}).$$

4. Amostre o próximo estado x de

$$p(\mathbf{x} \mid \boldsymbol{\omega} = g(\boldsymbol{h})) = p(\mathbf{x} \mid \tilde{\boldsymbol{x}}).$$

5. Então, a cadeira de Markov forma um estimador implícito consistente.



20.12 Generative Stochastic Networks

- São uma generalização dos *denoising autoencoders* (capítulo 14) que incluem variáveis latentes no processo generativo da cadeia de Markov.
- Parametrizadas por duas distribuições de probabilidade condicionais, cada uma especificando um passo na cadeia de Markov:

$$p(\mathbf{x}^{(k)} \mid \mathbf{h}^{(k)})$$

$$p(\mathbf{h}^{(k)} \mid \mathbf{h}^{(k-1)}, \mathbf{x}^{(k-1)})$$

- Note que o modelo parametriza o próprio processo generativo em vez da descrição matemática da distribuição conjunta das variáveis latentes e visíveis. Em outras palavras o modelo é **implícito** e é definido como distribuição estacionária da cadeia de Markov.
- Pode também ser aplicada em modelos discrimatorios com algumas mudanças.

20.13 Other Generation Schemes

- Processos de amostragem utilizados até o momento foram a amostragem MCMC, amostragem ancestral ou uma combinação dos dois. Todavia, existem outros?
- *Diffusion inversion:*
  - Inspirado na termodinâmica no regime de não-equilíbrio
  - A estrutura da distribuição de probabilidade pode ser aos poucos destruída, i.e. possuir maior entropia (imagine que se aumente a temperatura).
  - O método faz o inverso, ele aprende a reconstruir a estrutura da distribuição com entropia alta. Parecido com amostragens MCMC e a ideia supracitada dos *denoising autoencoders*.
- *Approximate Bayesian Computation (ABC) framework*:
  - Amostras são rejeitadas ou modificadas para forçar os momentos de certas funções das amostras serem iguais a da distribuição dos dados.
  - Parecido com *moment maching*, porém o ABC modifica diretamente as amostras em vez de contruir um modelo inteiro para isso.

20.14 Evaluating Generative Models

- Não são tão diretos quanto as medidas de desempenho de modelos supervisionados.
- Por exemplo, se compararmos dois modelos **A** e **B**:
  - **A** possui um limite inferior para a sua *likelihood* dos dados
  - **B** possui uma estimativa estocástica direta da *likelihood* e dos dados
  - Se Medida de **B** > Medida de **A**, não se pode falar precisamente qual dos dois modelos é o melhor.
  - Nesse caso pode-se testar os modelos diretamente numa tarefa requerida.
- As vezes precisa-se de $\log \tilde{p}(\boldsymbol{x}) - \log Z$, outro problema pois a função de partição pode ser complicada de estimar (lembrem do capítulo 18).
- Mudanças no pré-processamento dos dados mudam radicalmente a *likelihood* dos dados: MNIST com valores contínuos ou com valores binarizados possuem escalas completamente diferentes.

20.14 Evaluating Generative Models

- Inspeção visual pode ser problemática dada a chance do modelo reproduzir o conjunto de treino; um modo de detecção é procurar as vizinhas das amostras geradas no conjunto de treino.
- O modelo pode esquecer alguns modos da distribuição, porém o humano inspecionando não conseguirá lembrar se ele viu todos os sub-tipos de objetos que deveria ver – de 100 classes, o modelo pode ter ignorado 5.
- As vezes a *likelihood* não ser confiável: no MNIST o modelo pode alocar arbitrariamente baixa variancia para pixels que nunca mudam (fundo preto) – constituindo um fonte ilimitada de *likelihood*.
- Theis et al. (2015) argumentam que a métrica deve encaixar no uso, mesmo assim diversos problemas ainda continuam, tornando um sub-campo ativo de pesquisa dentro da comunidade de modelos generativos.

$$D_{\mathrm{KL}}(p_{\mathrm{data}} \| p_{\mathrm{model}}) \quad \Longleftrightarrow \quad D_{\mathrm{KL}}(p_{\mathrm{model}} \| p_{\mathrm{data}})$$

20.15 Conclusion

- Último capítulo extenso e que utiliza muitos conceitos anteriores: capítulo 16, 17, 18 e 19 no mínimo.
- Modelos generativos são uma ótima forma de construir modelos que entendem o mundo como apresentado via dados.
- Responde perguntas sobre inferência e representação.
- Gera dados novos, criando contingência, ou até certa forma de criatividade primitiva.
- A cognição é teorizada em ter profundas conexões com modelos probabilísticos generativos (vide o *free energy principle/predictive processing*). Portanto, avanços nesse campo são os que mais possuem chance de avançar o campo de Inteligência Artificial como um todo.

Referências:

- Livro Deep Learning, Bengio (2016).
  - Todas referências internas ao capítulo 20 acabam por ser relevantes. Incluindo todas as imagens utilizadas. As exceções possuem referências incluídas.
- Bengio et al. (2013c)
- Bengio et al., (2014)
- Sohl-Dickstein et al. (2015)
- Lee, (2009)
- Bornschein et al., (2015)
- Bornschein and Bengio, (2015)
- Theis et al. (2015)
- The free-energy principle: a unified brain theory? (Friston).