深度学习-神经网络基础知识

史春奇

2017 年

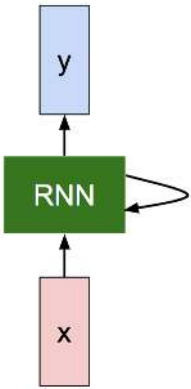
目录

第二部分（下）

- DBN 和 DBM 的特性
  - RBM 如何训练？
  - DBN 如何 Tune ？
  - DBM 和 DBN 相比？
- 自动编码器的理解
  - AE 和 PCA 如何联系？
  - 为什么要有 Sparse AE？
  - VAE 的优点？
- 生成对抗网络的理解
  - GAN 和 VAE 的不同？
  - GAN ZOO 的关系？

1 递归神经网络 RNN

- 基本的 RNN 结构



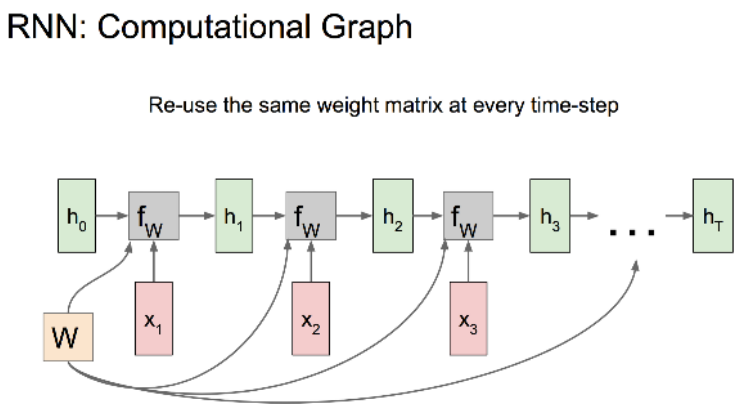
$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

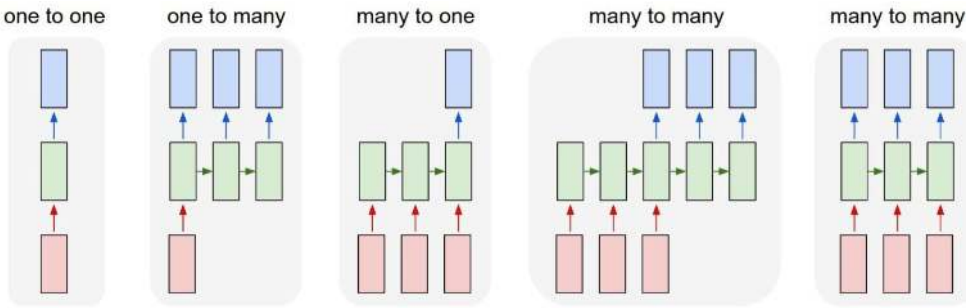
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

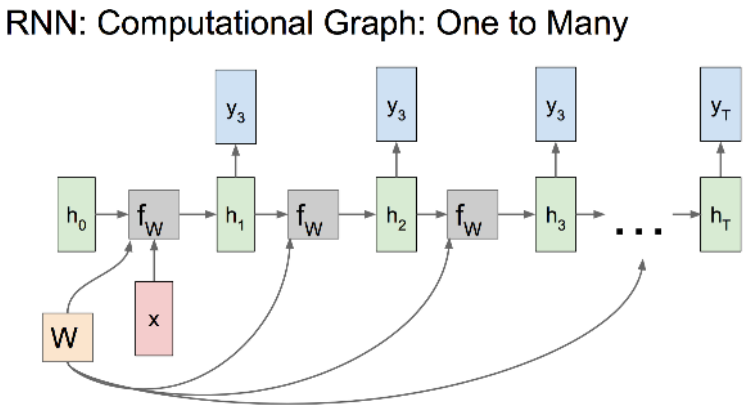
- 基本的 RNN 结构，按时间展开



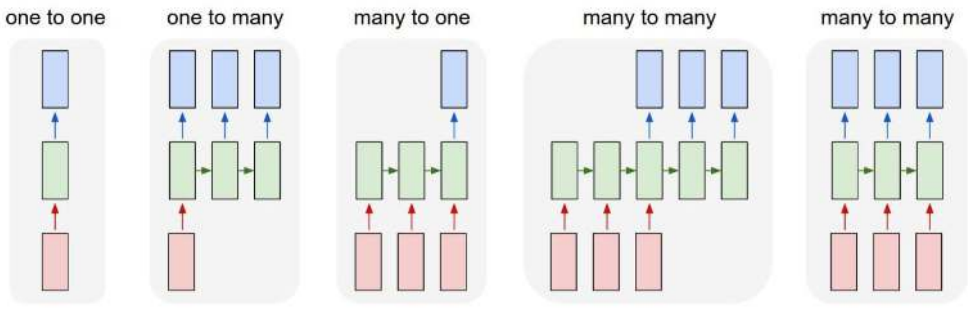
- 1 到多：图像到句子


one to one
one to many
many to one
many to many
many to many
e.g. **Image Captioning**
image -> sequence of words

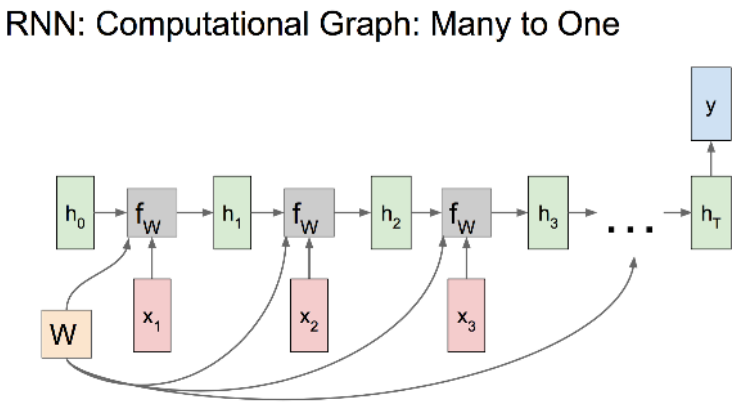
- 1 到多，按时间展开


RNN: Computational Graph: One to Many
y3
y3
y3
yT
h0
fW
h1
fW
h2
fW
h3
hT
W
x

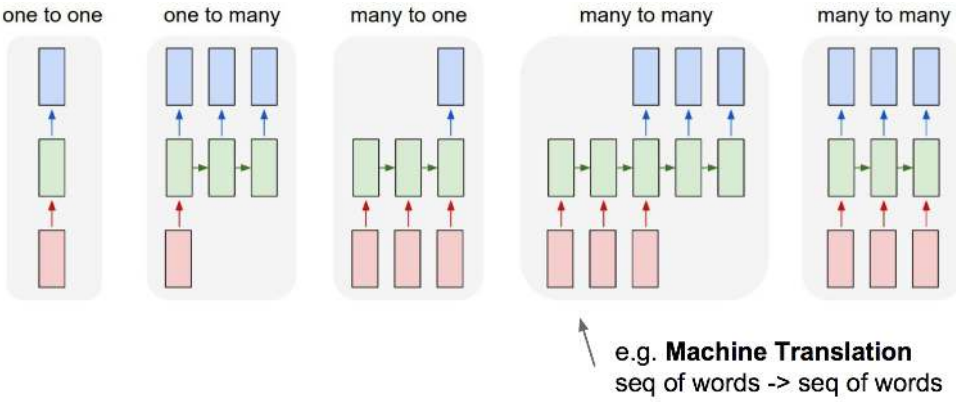
- 多到 1：句子到情感


one to one
one to many
many to one
many to many
many to many
e.g. Sentiment Classification
sequence of words -> sentiment

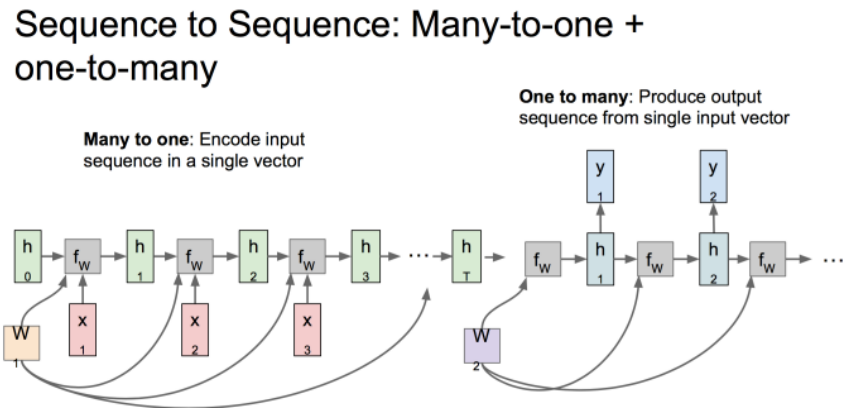
- 多到 1，按时间展开


RNN: Computational Graph: Many to One
y
h0
fW
h1
fW
h2
fW
h3
hT
W
x1
x2
x3

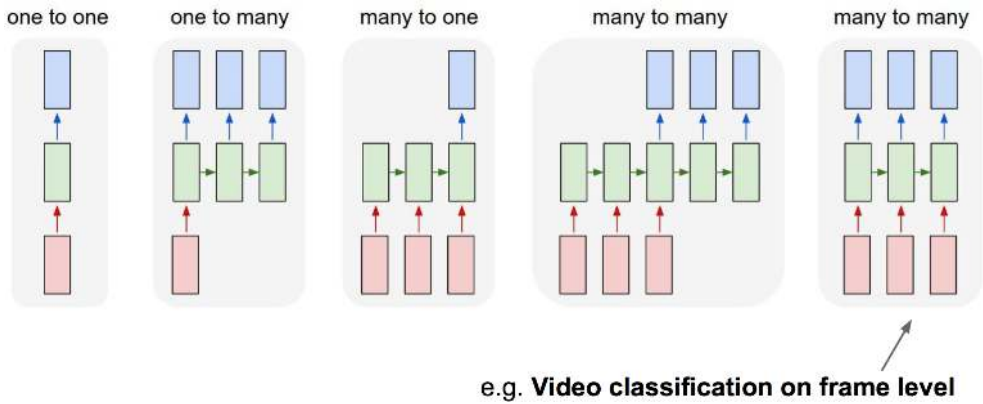
- 多到多：seq2seq 机器翻译



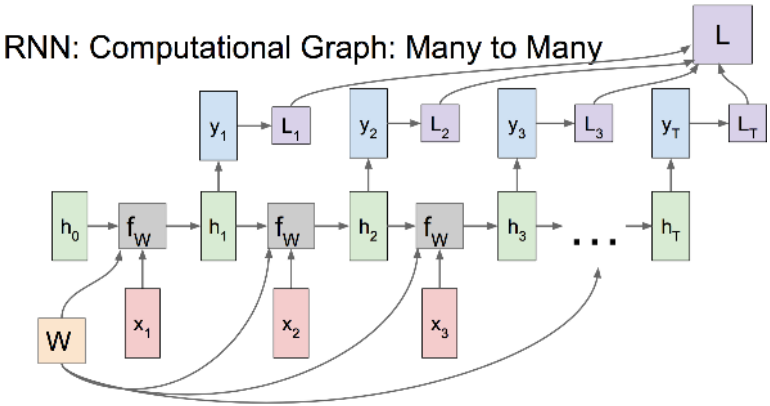
- 多到多 (seq2seq)，按时间展开



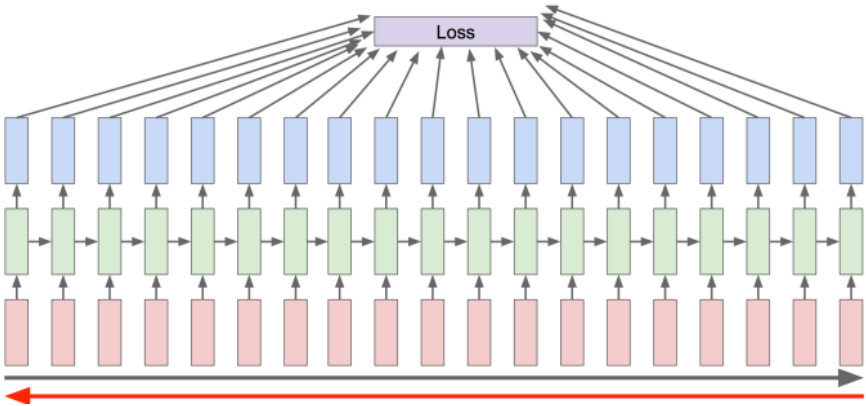
- 多到多：视频基于帧的分类



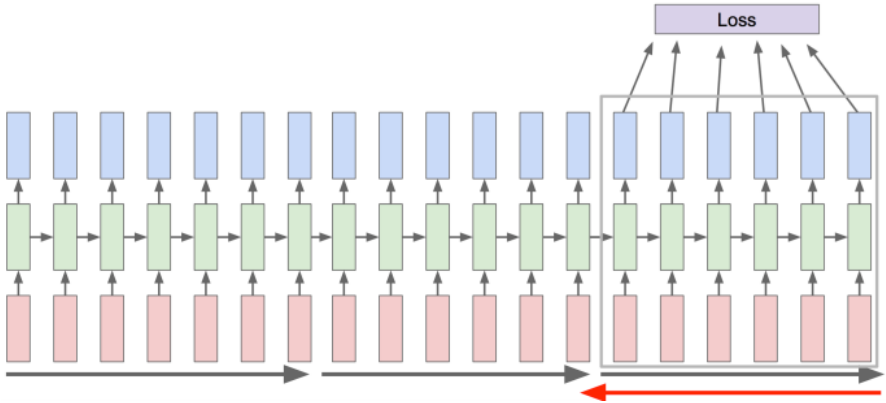
- 多到多，按时间展开


RNN: Computational Graph: Many to Many
L
y1
L1
y2
L2
y3
L3
yT
LT
h0
fW
h1
fW
h2
fW
h3
hT
W
x1
x2
x3

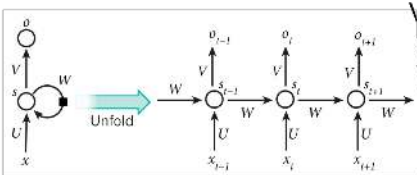
- BPTT：backpropagation through time



- Truncated BPTT: 划分成少量步骤



- Truncated BPTT



V的梯度：

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} = (\hat{y}_3 - y_3) \otimes s_3$$

W的梯度：

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

W的梯度：

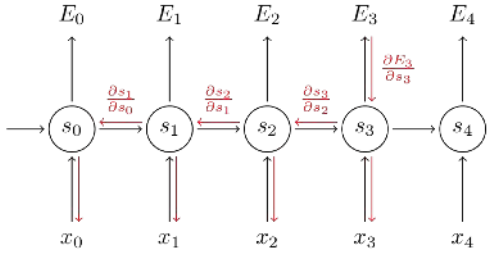
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

参数：U、W、V

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

Log损失：

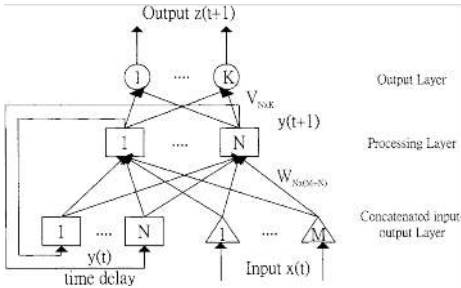
$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$
$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t$$



- Truncated BPTT

```
def bptt(self, x, y):
    T = len(y)
    # Perform forward propagation
    o, s = self.forward_propagation(x)
    # We accumulate the gradients in these variables
    dLdU = np.zeros(self.U.shape)
    dLdV = np.zeros(self.V.shape)
    dLdW = np.zeros(self.W.shape)
    delta_o = o
    delta_o[np.arange(len(y)), y] -= 1.
    # For each output backwards...
    for t in np.arange(T)[::-1]:
        dLdV += np.outer(delta_o[t], s[t].T)
        # Initial delta calculation: dL/dz
        delta_t = self.V.T.dot(delta_o[t]) * (1 - (s[t] ** 2))
        # Backpropagation through time (for at most self.bptt_truncate steps)
        for bptt_step in np.arange(max(0, t-self.bptt_truncate), t+1)[::-1]:
            # print "Backpropagation step t=%d bptt step=%d " % (t, bptt_step)
            # Add to gradients at each previous step
            dLdW += np.outer(delta_t, s[bptt_step-1])
            dLdU[:,x[bptt_step]] += delta_t
            # Update delta for next step dL/dz at t-1
            delta_t = self.W.T.dot(delta_t) * (1 - s[bptt_step-1] ** 2)
    return [dLdU, dLdV, dLdW]
```

- RTRL real time recurrent learning



$$p_{ij}^k(t) = \frac{\partial y_k(t)}{\partial w_{ij}}$$

$$\frac{\partial J(t)}{\partial w_{ij}} = \sum_{k \in U} e_k(t) p_{ij}^k(t)$$

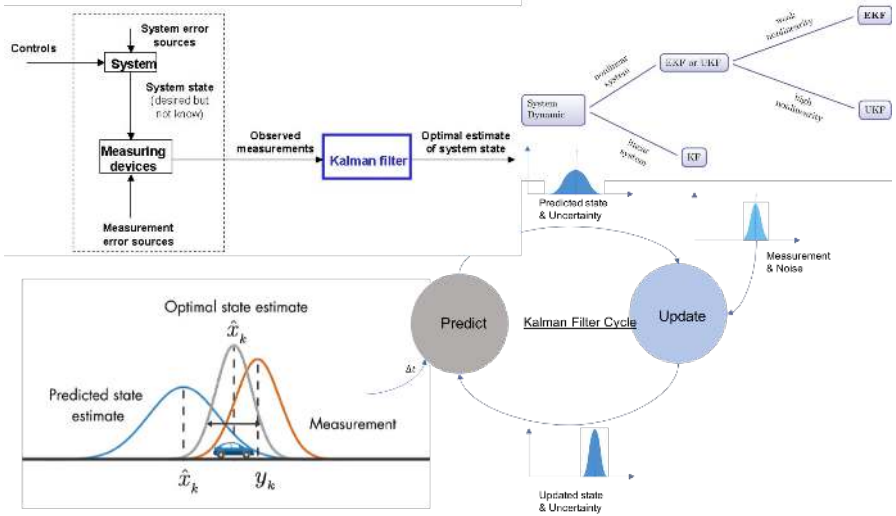
$$p_{ij}^k(t+1) = f_k'(s_k(t+1)) \left[\sum_{l \in U} w_{kl} p_{ij}^l(t) + \delta_{ik} x_j(t) \right]$$

$$p_{ij}^k(t_0) = \frac{\partial y_k(t_0)}{\partial w_{ij}} = 0$$

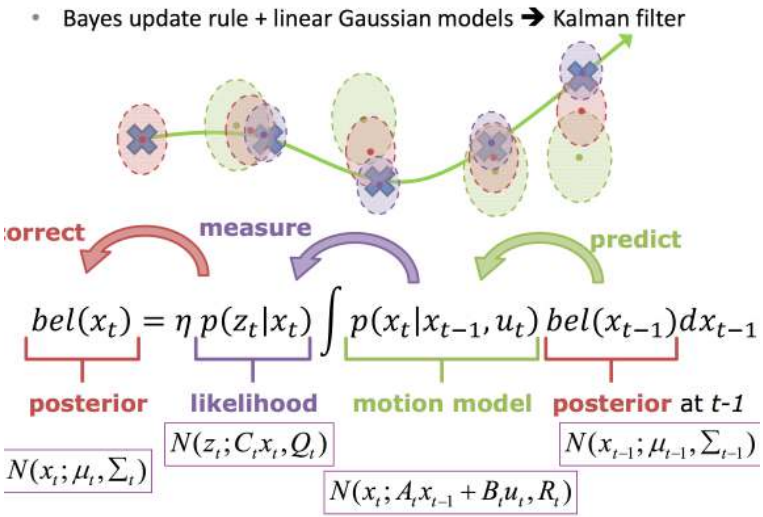
$\delta_{ik} = 1$ *if* $i = k$ *and* 0 *otherwise*

和 BPTT 的等价，反向和前向的区别！

- ELF extended Kalman filter



- Kalman filter



- Kalman filter

1. **Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
9. Return μ_t, Σ_t

Requirements:

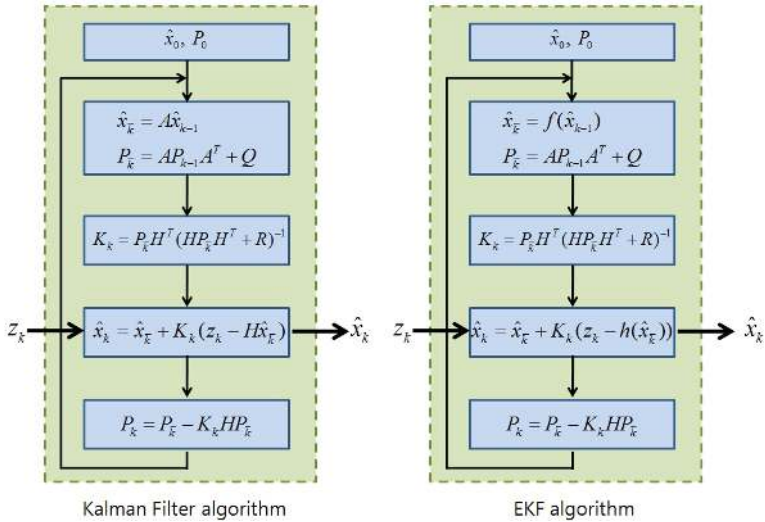
- Initial belief is Gaussian
- Linear Gaussian state-space model

- Extended Kalman filter

1. **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction:
3. $\bar{\mu}_t = g(u_t, \mu_{t-1})$ ⟵ $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$ ⟵ $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$ ⟵ $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$ ⟵ $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$ ⟵ $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
9. Return μ_t, Σ_t

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

- Extended Kalman filter



- Extended Kalman filter with Noise

1. **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction:
3. $\bar{\mu}_t = g(u_t, \mu_{t-1})$
4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + W_t R_t W_t^T$
5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + V_t Q_t V_t^T)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$
8. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
9. Return μ_t, Σ_t

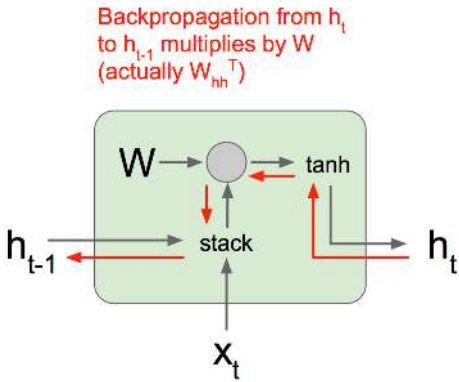
$$G_t = \frac{\partial g(u_t, \mu_{t-1}, 0)}{\partial x_{t-1}}$$

$$W_t = \frac{\partial g(u_t, \mu_{t-1}, \varepsilon_t)}{\partial \varepsilon_t}$$

$$H_t = \frac{\partial h(\bar{\mu}_t, 0)}{\partial x_t}$$

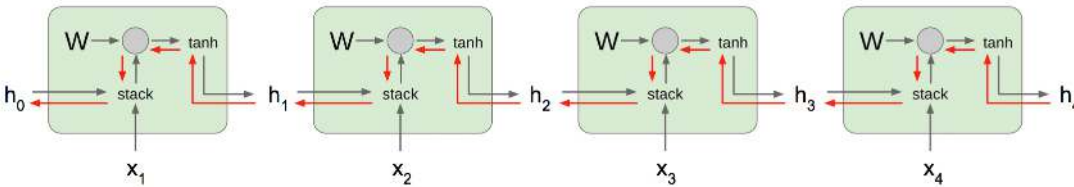
$$V_t = \frac{\partial h(\bar{\mu}_t, \delta)}{\partial \delta_t}$$

- Vanilla RNN 矩阵表示



$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

- Vanilla RNN 的梯度消失、爆炸

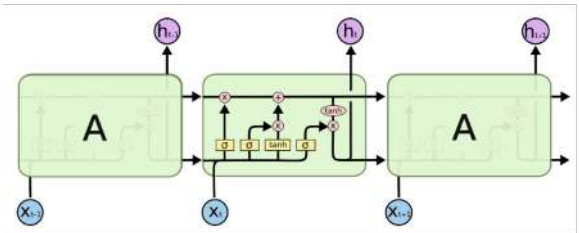


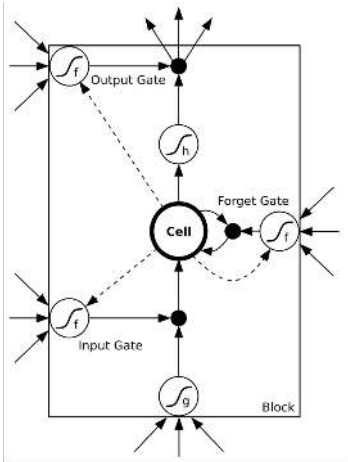
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

- Long Short Term Memory





$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

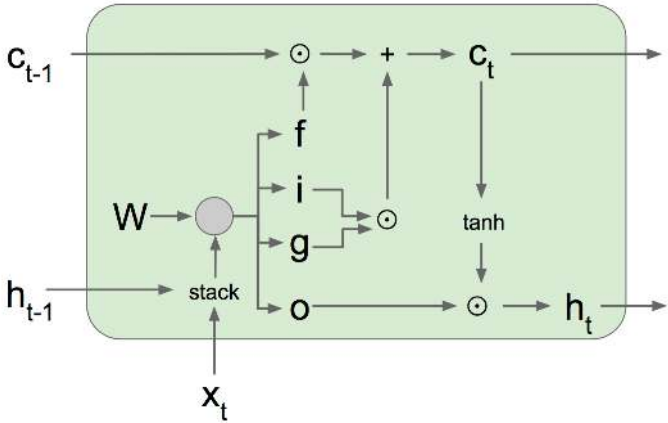
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- LSTM 矩阵表示

[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- LSTM 矩阵表示

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

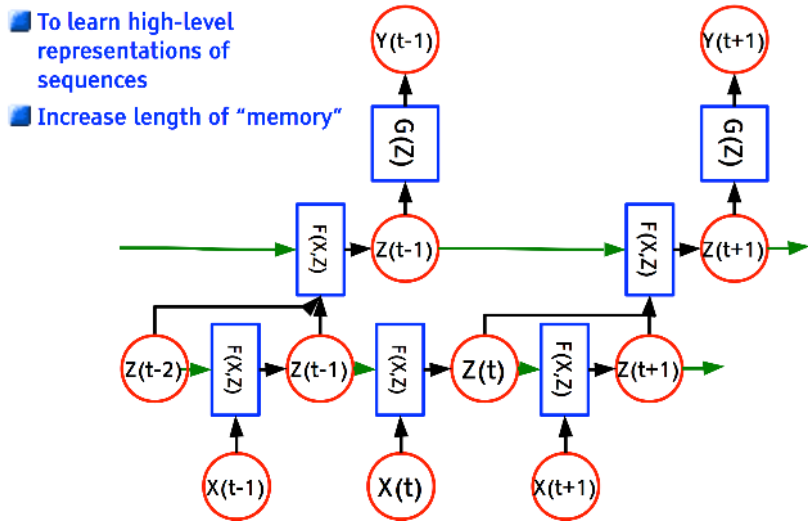
LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- LSTM 记忆单元


To learn high-level representations of sequences
Increase length of "memory"
Y(t-1)
G(Z)
F(X,Z)
Z(t-1)
Z(t-2)
X(t-1)
Z(t)
X(t)
Z(t+1)
X(t+1)
Y(t+1)

- LSTM 记忆单元

Some state variable are constrained to evolve slowly (matrix near unity)
- Slow state used as context for faster state.

"Learning Longer Memory in Recurrent Neural Networks",
- Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, Marc'Aurelio Ranzato [arxiv:1412.7753]

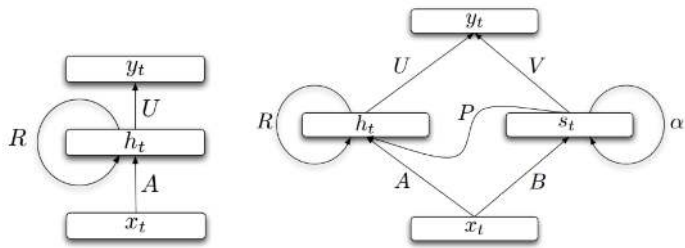
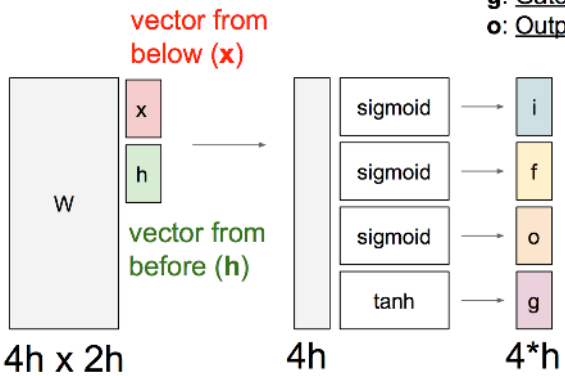


Figure 1: (a) Simple recurrent network. (b) Recurrent network with context features.

- LSTM 如何学习

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



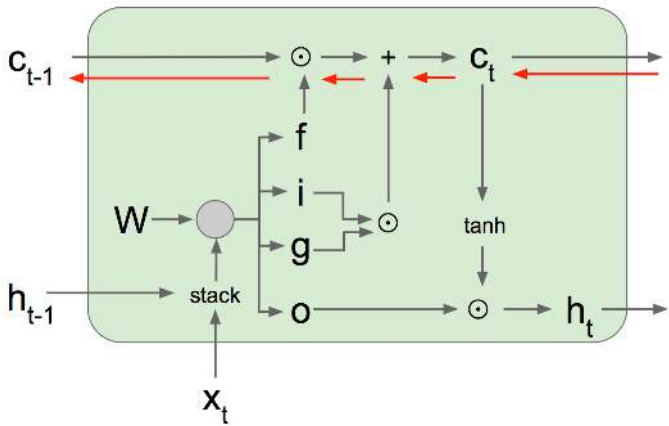
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- LSTM 梯度反传

[Hochreiter et al., 1997]



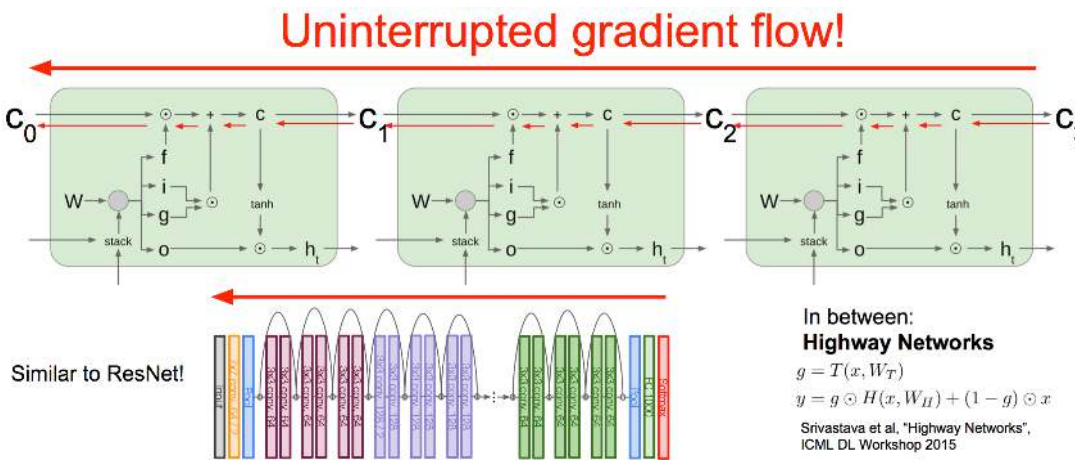
Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

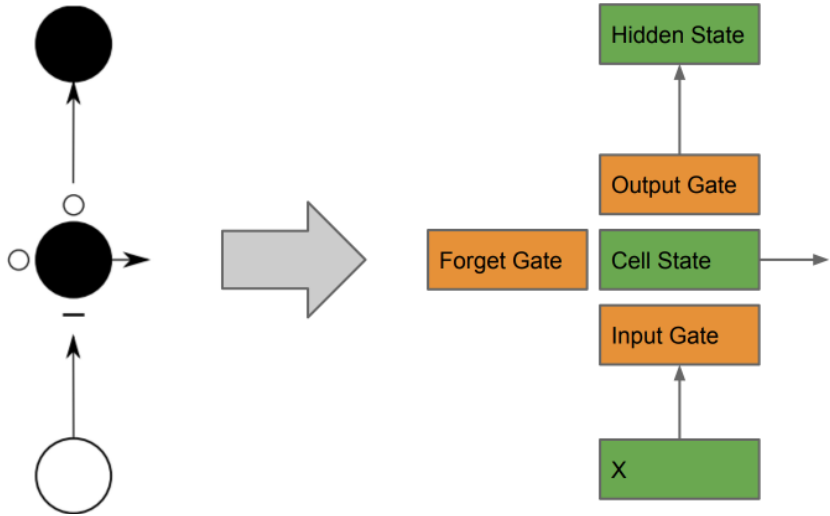
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

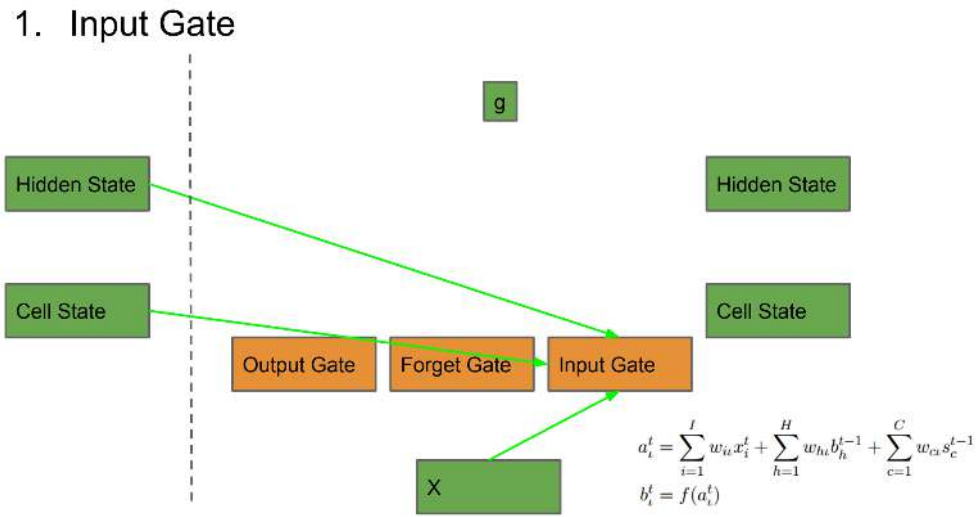
- LSTM 梯度反传


Uninterrupted gradient flow!
C0
C1
C2
C3
W
stack
f
i
g
o
tanh
c
ht
Similar to ResNet!
In between:
Highway Networks
$g = T(x, W_T)$
$y = g \odot H(x, W_H) + (1 - g) \odot x$
Srivastava et al. "Highway Networks", ICML DL Workshop 2015

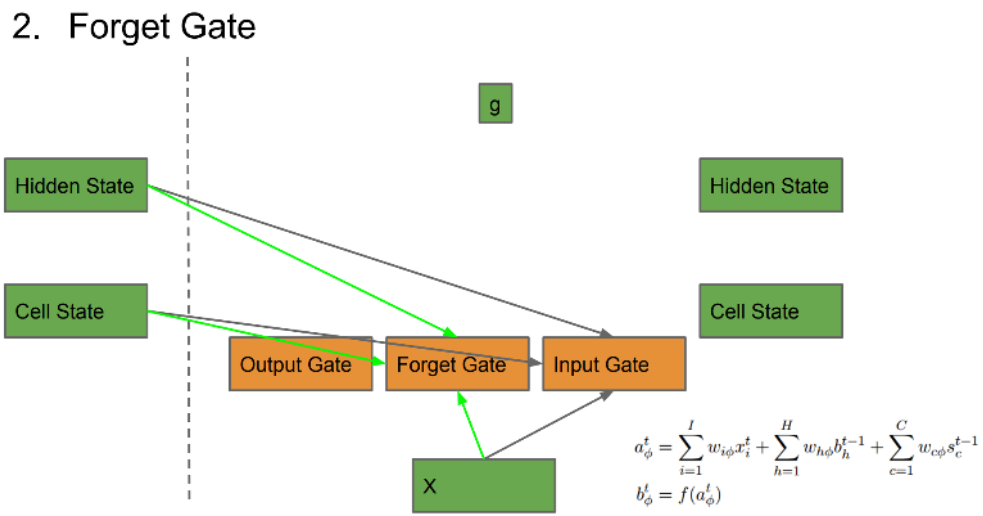
- LSTM 正向传播


Hidden State
Output Gate
Forget Gate
Cell State
Input Gate
X

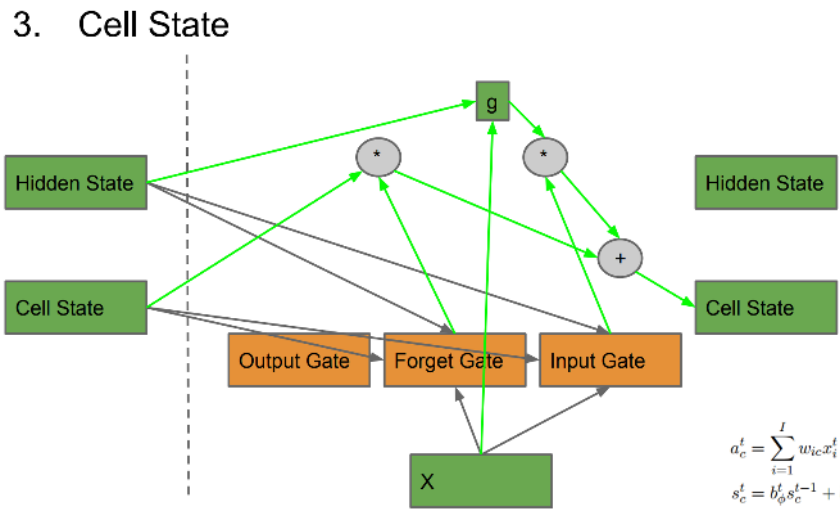
- LSTM 正向传播：输入门



- LSTM 正向传播：忘记门

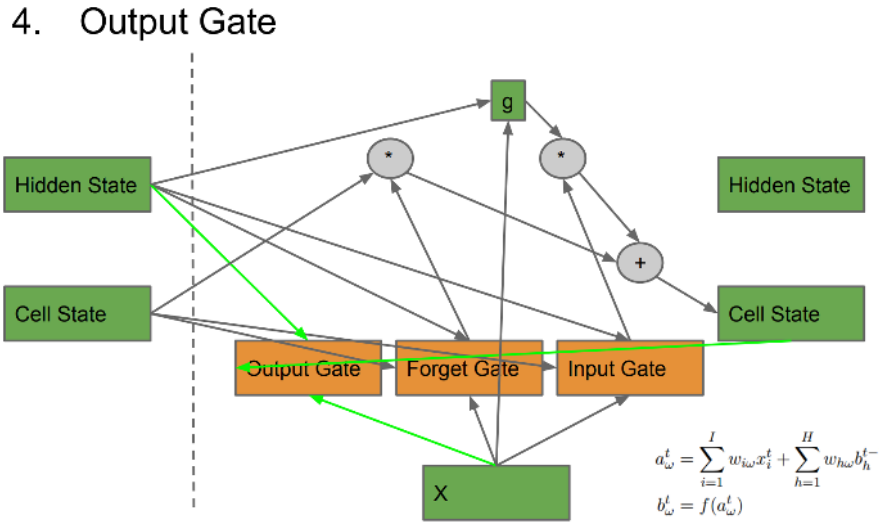


- LSTM 正向传播：记忆单元 Cell 状态

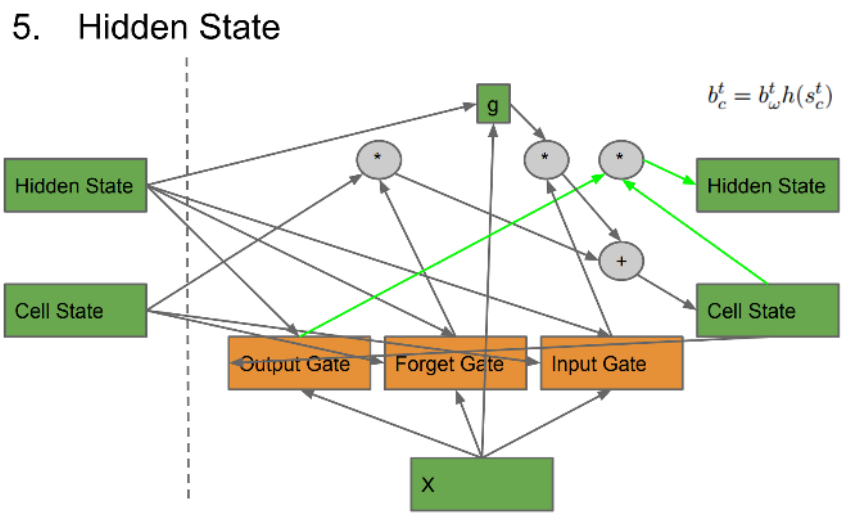


$$a_c^t = \sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} b_h^{t-1}$$
$$s_c^t = b_\phi^t s_c^{t-1} + b_\iota^t g(a_c^t)$$

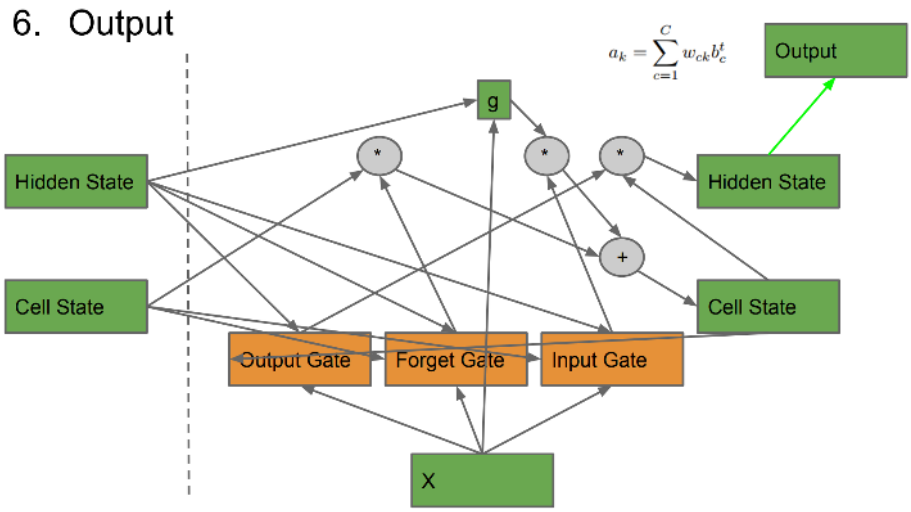
- LSTM 正向传播：输出门



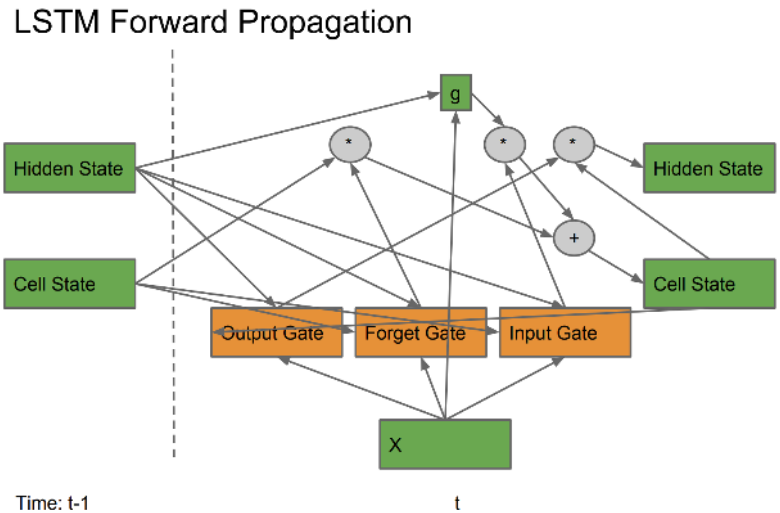
- LSTM 正向传播：隐藏状态



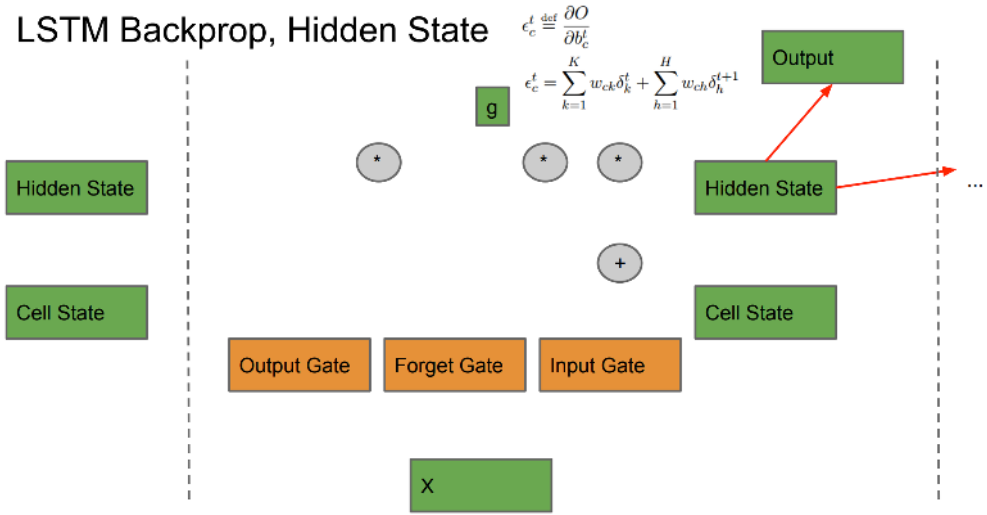
- LSTM 正向传播：最后输出



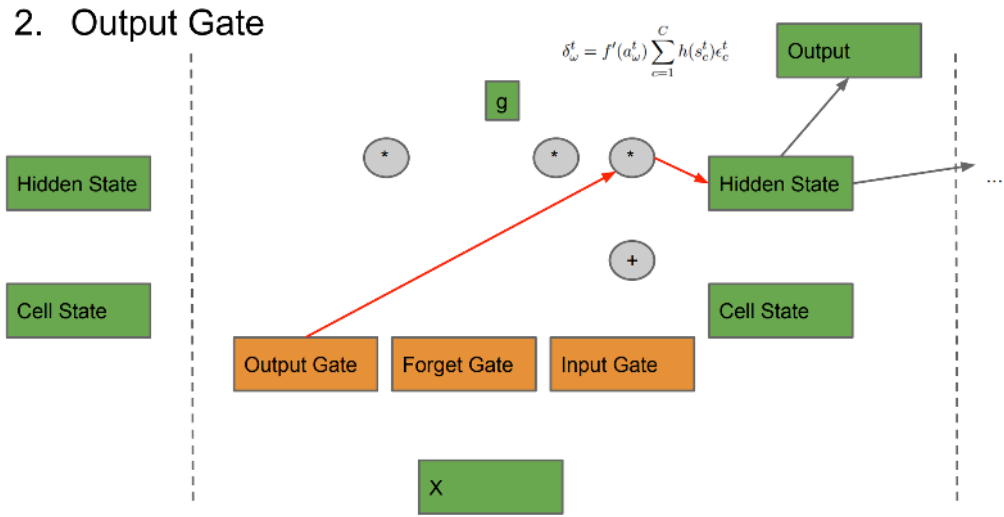
- LSTM 正向传播


LSTM Forward Propagation
g
*
*
*
+
Hidden State
Cell State
Hidden State
Cell State
Output Gate
Forget Gate
Input Gate
X
Time: t-1
t

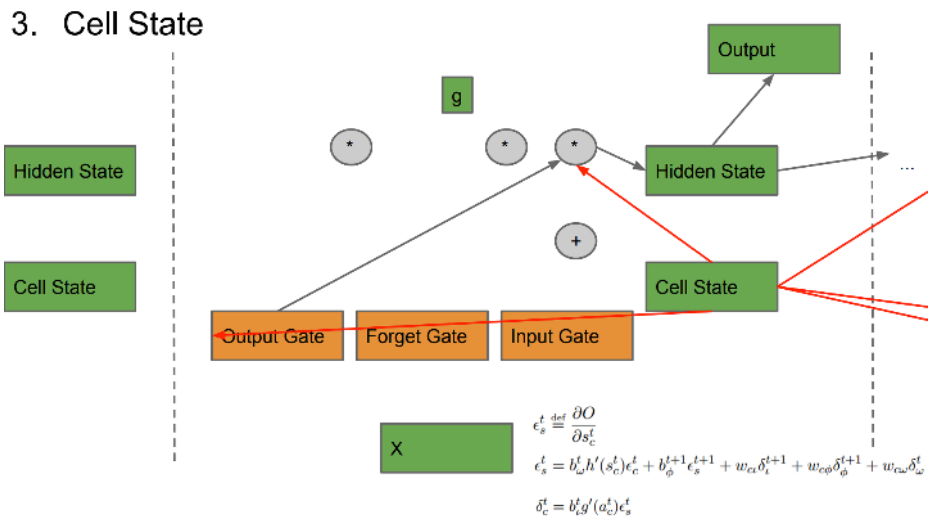
- LSTM 反向传播：输出和隐藏状态



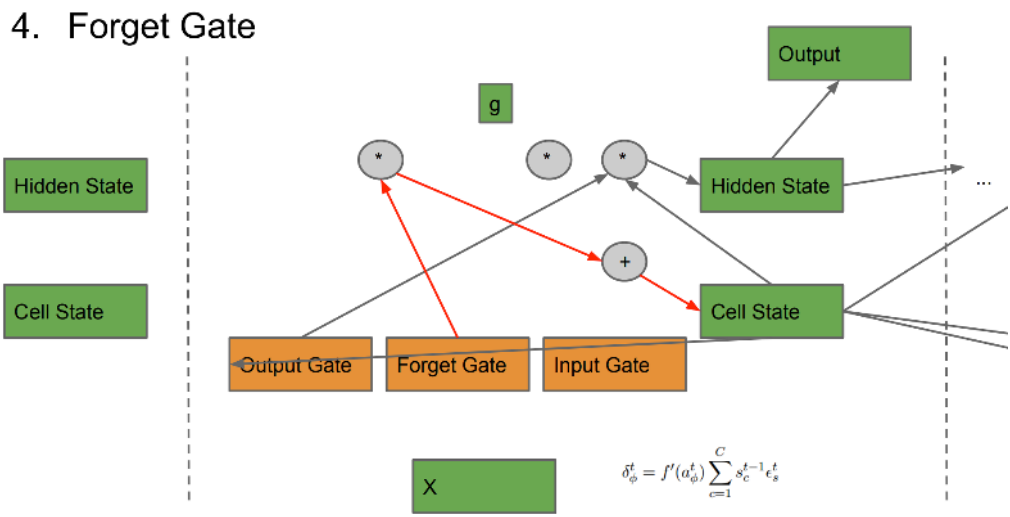
- LSTM 反向传播：输出门



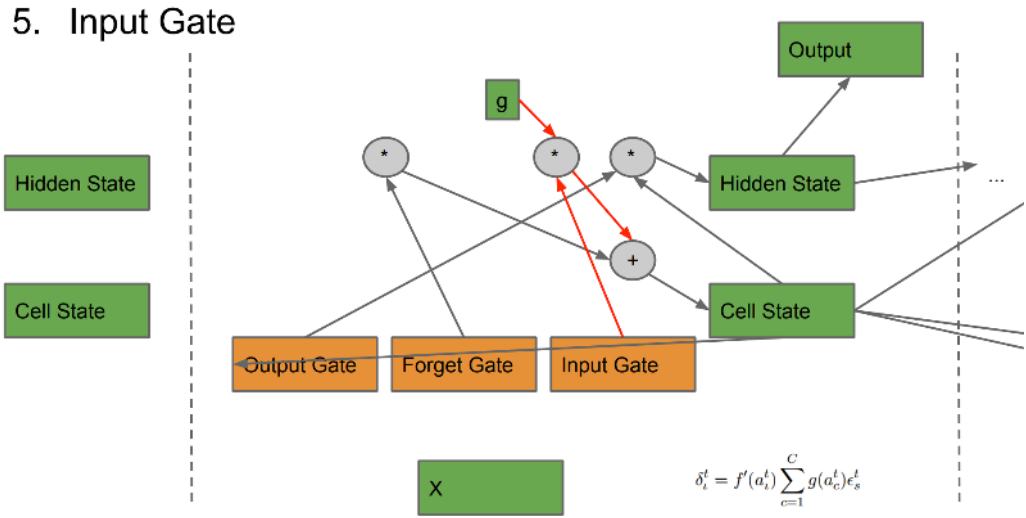
- LSTM 反向传播：记忆 Cell 状态



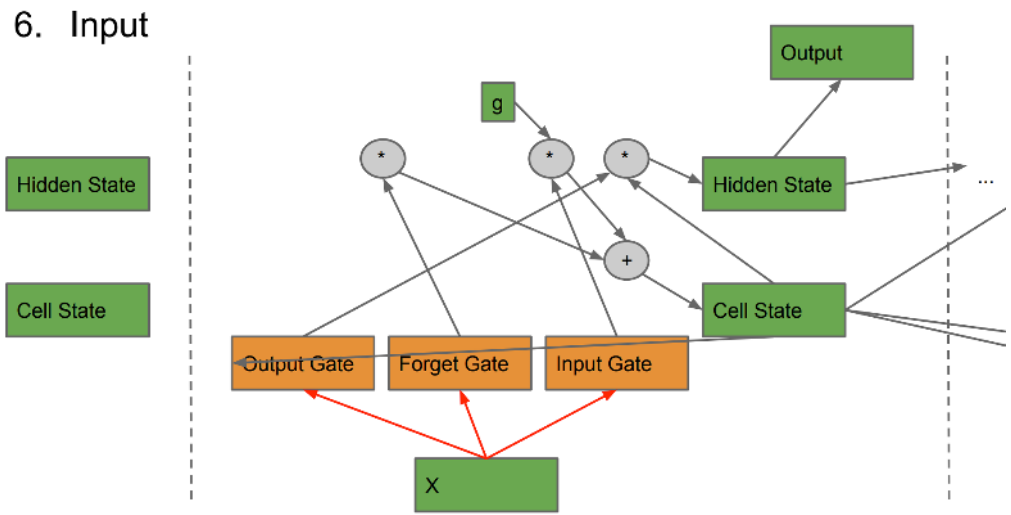
- LSTM 反向传播：忘记门



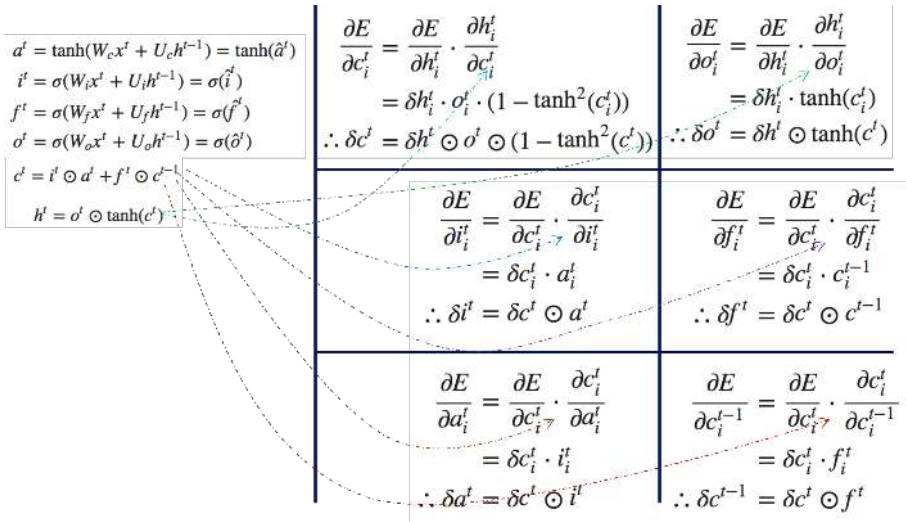
- LSTM 反向传播：输入门



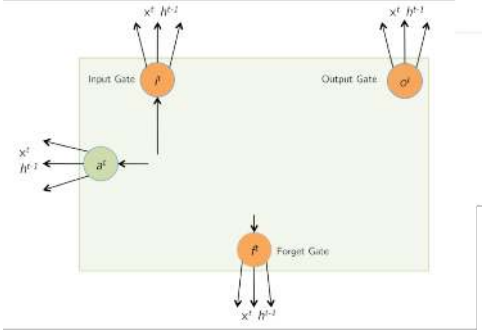
- LSTM 反向传播：输入


6. Input
Output
g
Hidden State
Hidden State
...
Cell State
Cell State
Output Gate
Forget Gate
Input Gate
X

- LSTM 反向传播详细


$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t)$
$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t)$
$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t)$
$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t)$
$c^t = i^t \odot a^t + f^t \odot c^{t-1}$
$h^t = o^t \odot \tanh(c^t)$
$\frac{\partial E}{\partial c_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial c_i^t}$
$= \delta h_i^t \cdot o_i^t \cdot (1 - \tanh^2(c_i^t))$
$\therefore \delta c^t = \delta h^t \odot o^t \odot (1 - \tanh^2(c^t))$
$\frac{\partial E}{\partial o_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial o_i^t}$
$= \delta h_i^t \cdot \tanh(c_i^t)$
$\therefore \delta o^t = \delta h^t \odot \tanh(c^t)$
$\frac{\partial E}{\partial i_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial i_i^t}$
$= \delta c_i^t \cdot a_i^t$
$\therefore \delta i^t = \delta c^t \odot a^t$
$\frac{\partial E}{\partial f_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial f_i^t}$
$= \delta c_i^t \cdot c_i^{t-1}$
$\therefore \delta f^t = \delta c^t \odot c^{t-1}$
$\frac{\partial E}{\partial a_i^t} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial a_i^t}$
$= \delta c_i^t \cdot i_i^t$
$\therefore \delta a^t = \delta c^t \odot i^t$
$\frac{\partial E}{\partial c_i^{t-1}} = \frac{\partial E}{\partial c_i^t} \cdot \frac{\partial c_i^t}{\partial c_i^{t-1}}$
$= \delta c_i^t \cdot f_i^t$
$\therefore \delta c^{t-1} = \delta c^t \odot f^t$

- LSTM 的 BPTT



Forward Pass: $z^t = \begin{bmatrix} \hat{a}^t \\ \hat{i}^t \\ \hat{f}^t \\ \hat{o}^t \end{bmatrix} = W \times I^t$

Given $\delta a^t, \delta i^t, \delta f^t, \delta o^t$, find δz^t

$$\delta \hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t))$$
$$\delta \hat{i}^t = \delta i^t \odot i^t \odot (1 - i^t)$$
$$\delta \hat{f}^t = \delta f^t \odot f^t \odot (1 - f^t)$$
$$\delta \hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$
$$\delta z^t = \left[\delta \hat{a}^t, \delta \hat{i}^t, \delta \hat{f}^t, \delta \hat{o}^t\right]^T$$

Forward Pass: $z^t = W \times I^t$

Given δz^t, find $\delta W^t, \delta h^{t-1}$

$$\delta I^t = W^T \times \delta z^t$$

As $I^t = \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix}$,

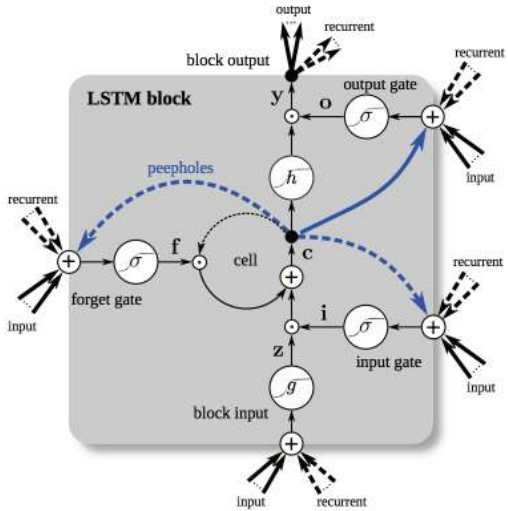
δh^{t-1} can be retrieved from δI^t

$$\delta W^t = \delta z^t \times (I^t)^T$$

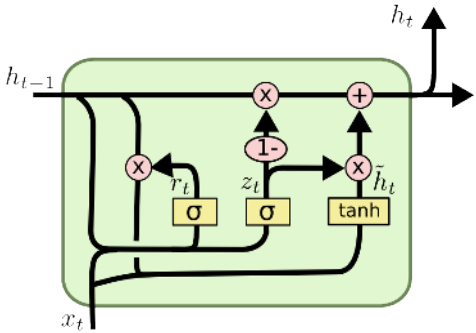
$$x = [x^1, x^2, \cdots, x^T],$$

$$\delta W = \sum_{t=1}^{T} \delta W^t$$

- LSTM Peepholes



- GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

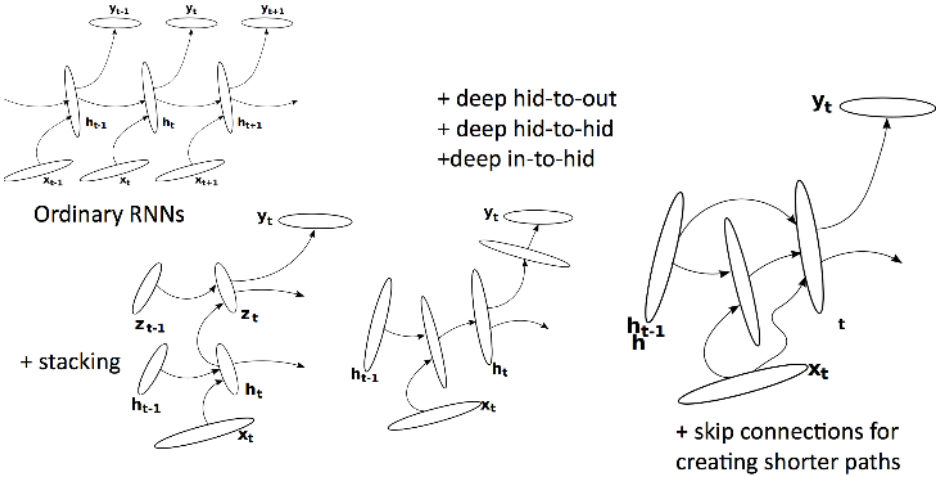
- GRU

1. 复位门 Reset Gate：如何结合以前隐藏状态和当前输入
2. 更新门 Update Gate：如何保留内部状态

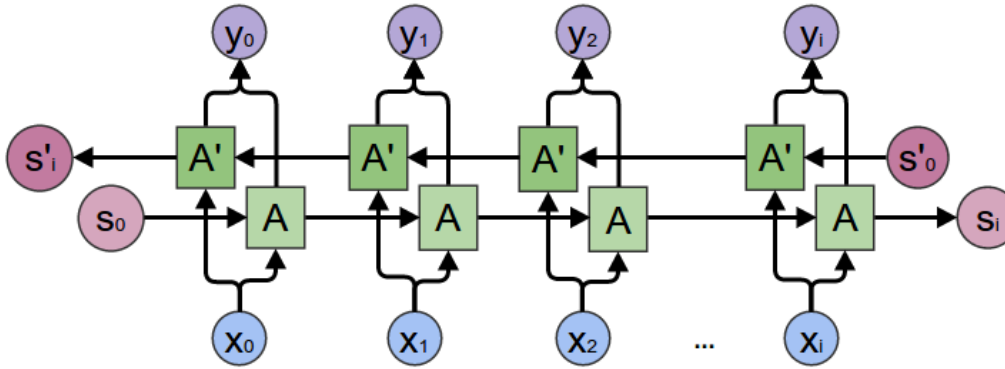
- LSTM vs GRU

1. 效果差不多
2. GRU 参数更少，需要的数据量少
3. LSTM 表达能力强，需要的数据量大

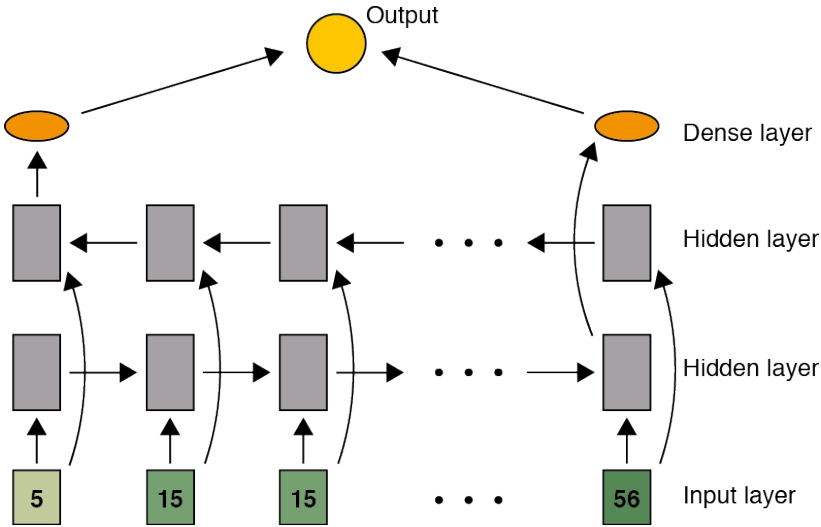
- 深度 RNN
  - ICLR 2014, *How to construct deep recurrent neural networks*



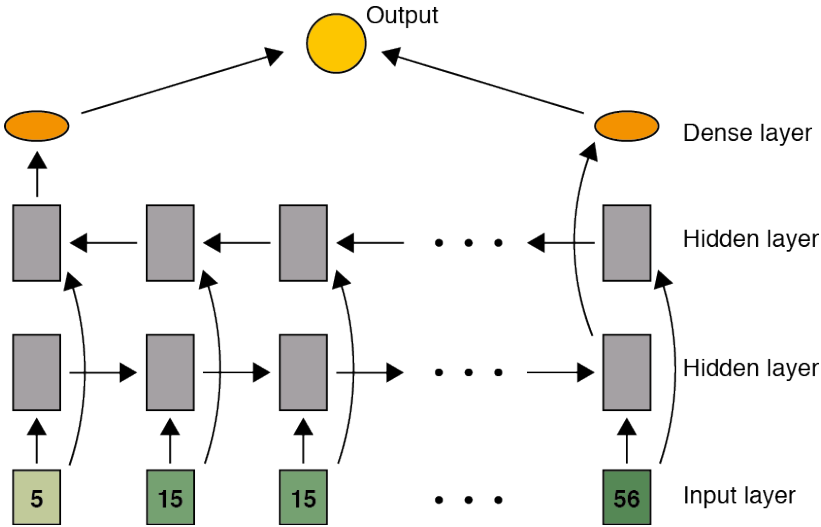
- Bidirectional RNN



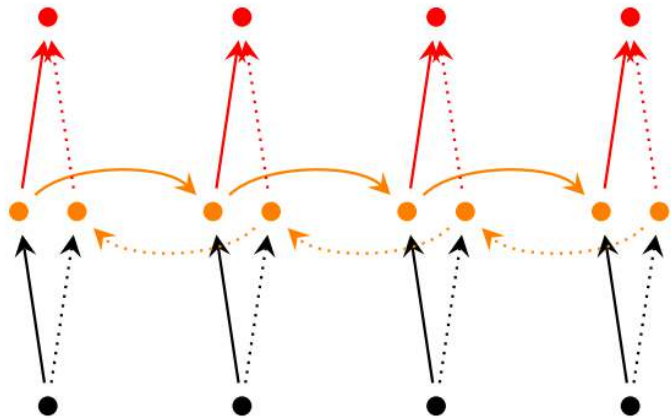
- Bidirectional RNN



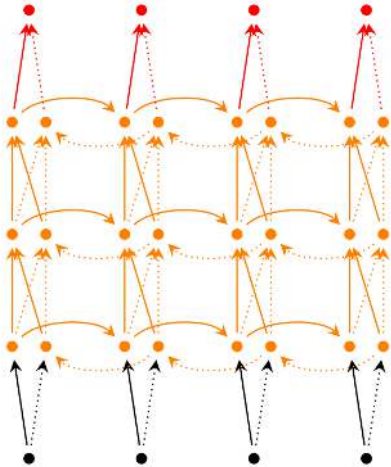
- Bidirectional RNN



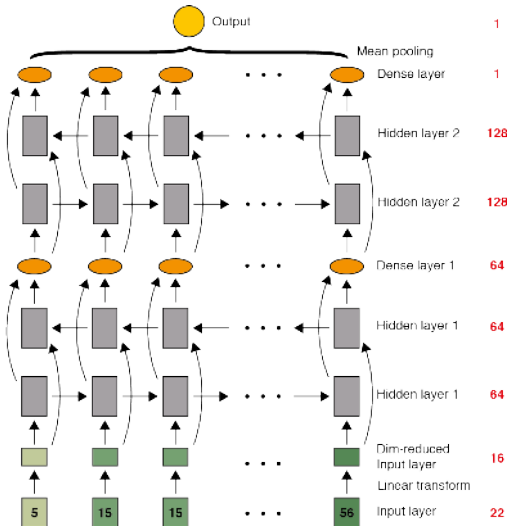
- Bidirectional RNN



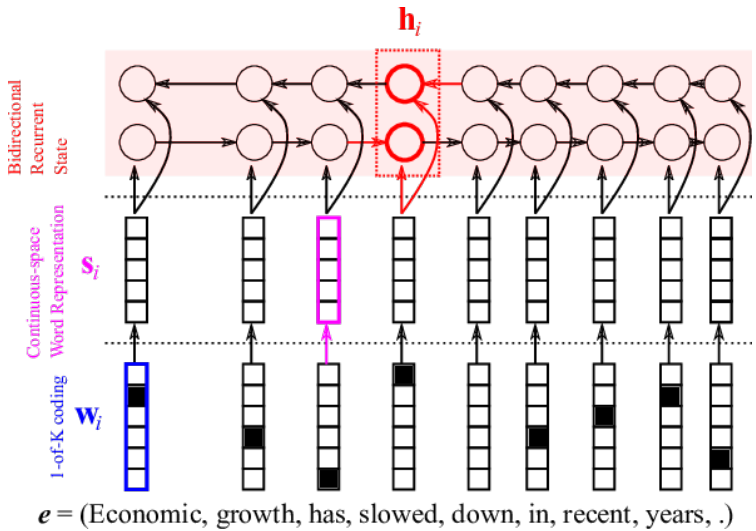
- Deep (Bidirectional) RNNs



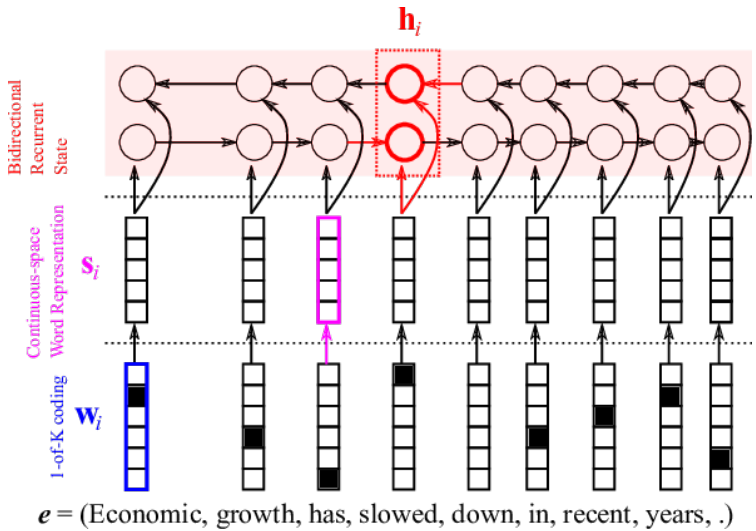
- Deep (Bidirectional) RNNs



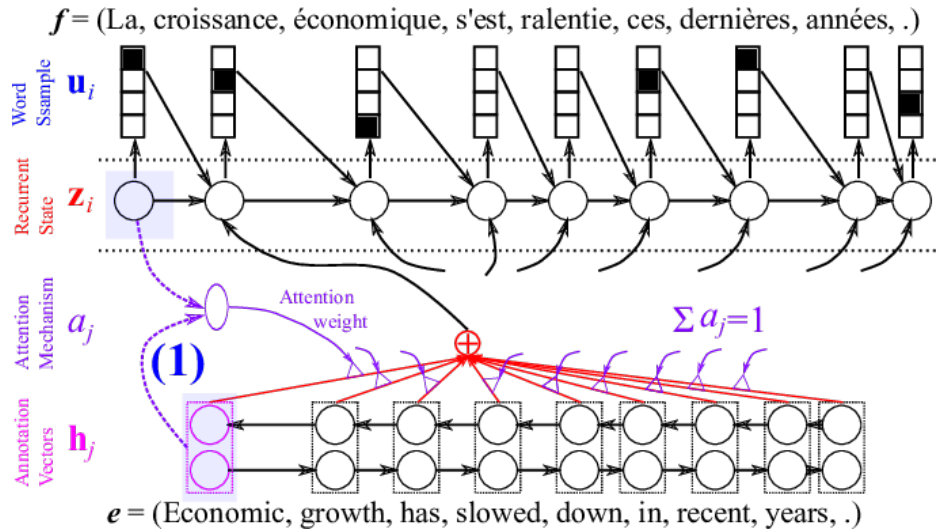
- Bidirectional RNNs 应用



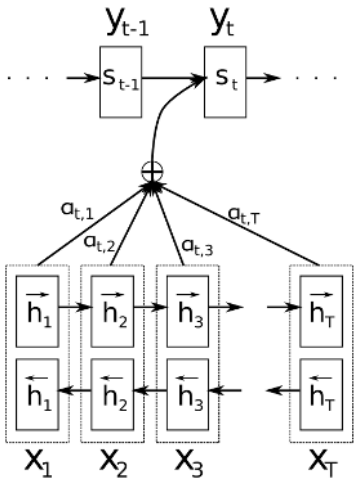
- Bidirectional RNNs 应用



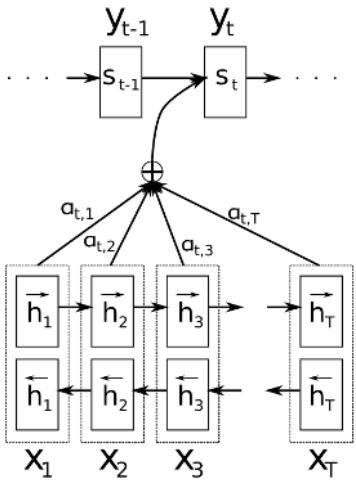
- Bidirectional RNN 和 Attention 机制



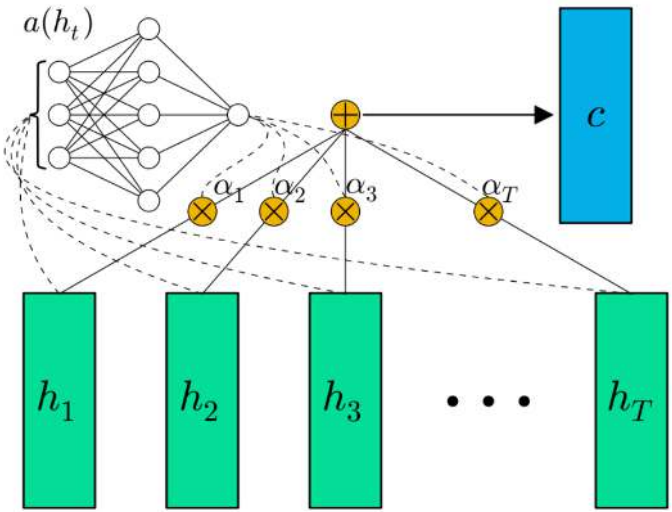
- Attention 机制



- Attention 机制



- Attention 机制：权重学习

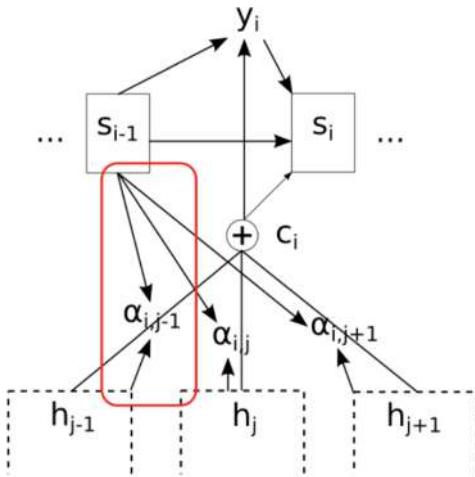


- Attention 机制

$$e_{ij} = v^T \tanh(W s_{i-1} + V h_j) \quad (1)$$

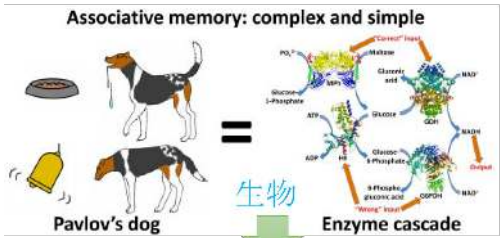
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L} \exp(e_{ik})} \quad (2)$$

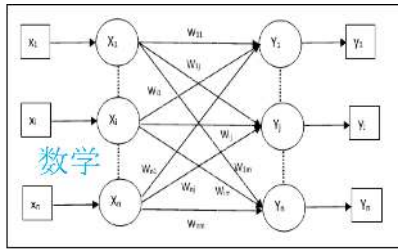
- nonlinearity (tanh) is crucial!
- simplest model possible
- Vh_j is precomputed => quadratic complexity with low constant

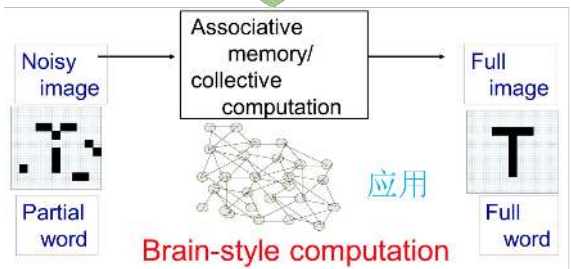


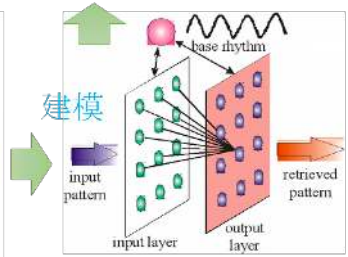
2 玻尔兹曼机 BM

- 联想记忆 Associative Memory

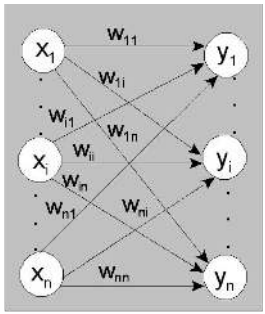








- 能量函数 Energy Memory



$(\mathbf{x}^1, \mathbf{y}^1)$
$(\mathbf{x}^2, \mathbf{y}^2)$
$\vdots$
$(\mathbf{x}^p, \mathbf{y}^p)$

$\mathbf{x}^i \in R^n$
$\mathbf{y}^i \in R^m$

$\mathbf{x}^i \equiv \mathbf{y}^i$ Autoassociative

$\mathbf{x}^i \neq \mathbf{y}^i$ Heteroassociative

Hebb学习：

$$y = \sum_j w_j x_j$$

$$\Delta w_i = \eta x_i y$$

梯度下降：

$$w := w - \eta \nabla E(w)$$

能量函数：

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} x_i y_j$$

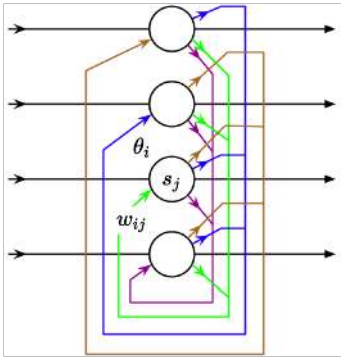
- Hopfield 网络能量函数

能量函数：

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i y_j$$

神经单元设置：

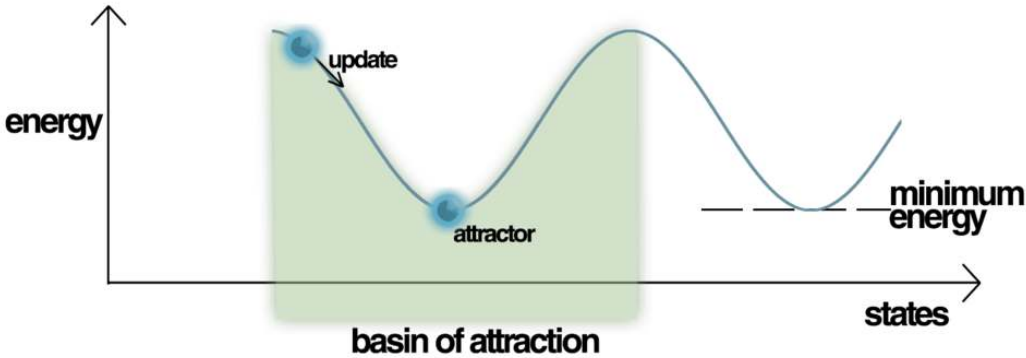
$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij}s_j \geq \theta_i \\ -1 & \text{otherwise.} \end{cases}$$



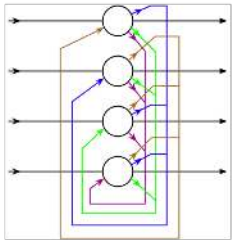
Hopfield网络能量函数：

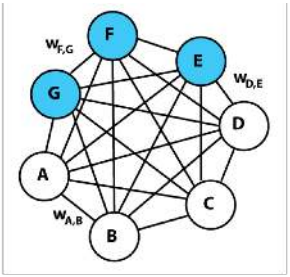
$$E = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i$$

- Hopfield 网络能量



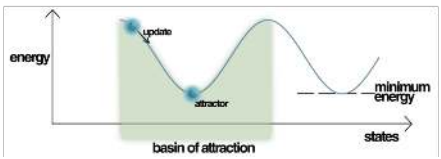
- 玻尔兹曼机 Boltzmann Machine

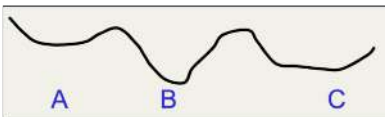




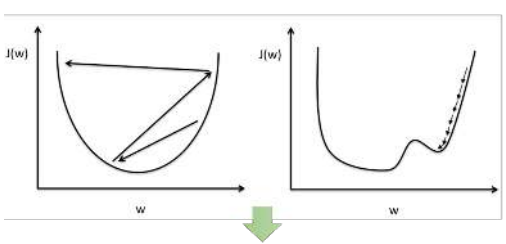
$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

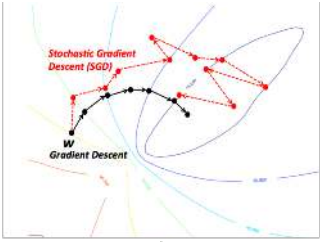
$$E = -\left(\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i\right)$$

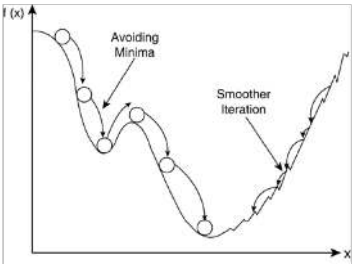


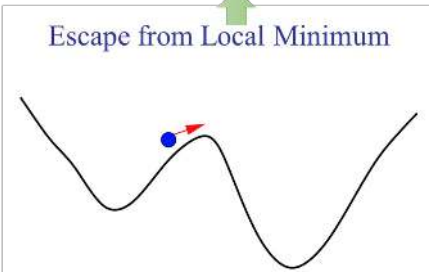


- 引入随机、跳出局部最小值









- 玻尔兹曼分布

能级状态：

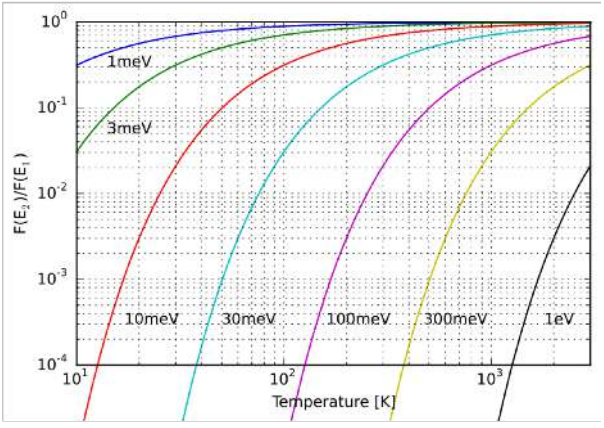
$$F(\text{state}) \propto e^{-\frac{E}{kT}}$$

粒子数量占比：

$$\frac{N_i}{N} = \frac{e^{-\varepsilon_i/kT}}{\sum_{j=1}^{M} e^{-\varepsilon_j/kT}}$$

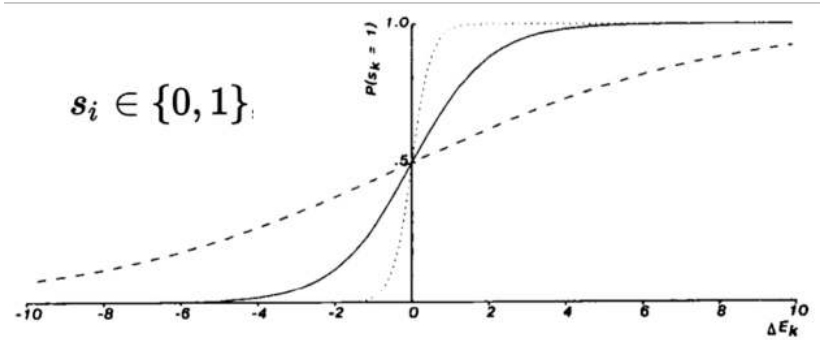
能级概率：

$$p_i = \frac{e^{-\varepsilon_i/kT}}{\sum_{j=1}^{M} e^{-\varepsilon_j/kT}}$$

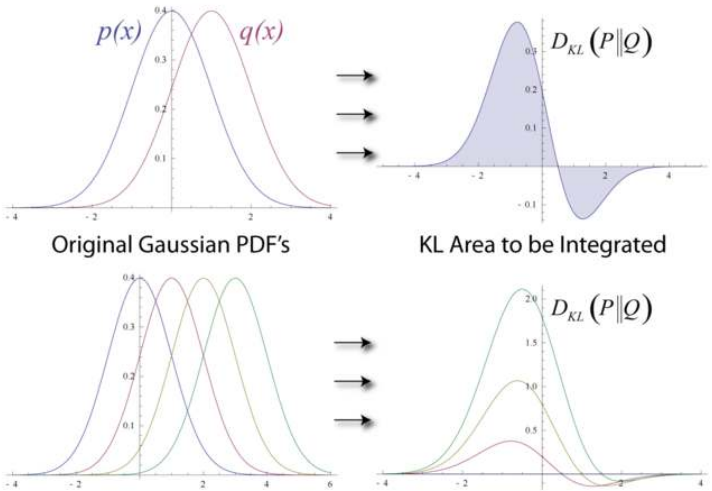


- 玻尔兹曼机的分布

$$p(s_i{=}1) = \frac{1}{1+e^{-\Delta E_i/T}}$$



- 分布间距离：KL 散度


p(x)
q(x)
$D_{KL}(P\|Q)$
Original Gaussian PDF's
KL Area to be Integrated
$D_{KL}(P\|Q)$

- KL 散度

  连续定义：

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\, \log \frac{p(x)}{q(x)}\, dx,$$

  离散定义：

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\, \log \frac{P(i)}{Q(i)}.$$

熵的解释：交互熵 - 熵

$$\begin{array}{rcccc} D_{\mathrm{KL}}(P\|Q) & = & -\sum_x p(x)\log q(x) & + & \sum_x p(x)\log p(x) \\ & = & H(P,Q) & - & H(P) \end{array}$$

- KL 散度称为 KL 距离：度量大于 0

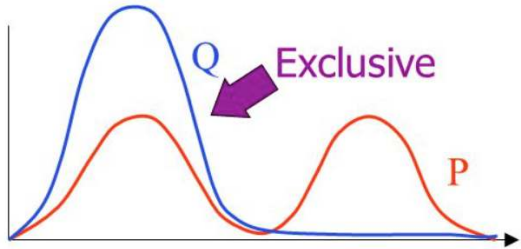
$$
\begin{aligned}
D(p||q) &= \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log\frac{q(x)}{p(x)} \\
&= -\mathbb{E}\left[\log\frac{q}{p}\right] \\
&\geq -\log\left(\mathbb{E}\left[\frac{q}{p}\right]\right) \\
&= -\log\left(\sum_{x\in\mathcal{X}} p(x)\frac{q(x)}{p(x)}\right) \\
&= 0,
\end{aligned}
$$

- KL 散度性质 1：

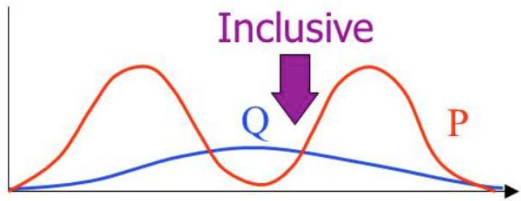
$$
\begin{aligned}
D(p||q) &= \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log\frac{q(x)}{p(x)} \\
&= -\mathbb{E}\left[\log\frac{q}{p}\right] \\
&\geq -\log\left(\mathbb{E}\left[\frac{q}{p}\right]\right) \\
&= -\log\left(\sum_{x\in\mathcal{X}} p(x)\frac{q(x)}{p(x)}\right) \\
&= 0,
\end{aligned}
$$

- KL 距离性质 2：非对称度量

Minimising
$KL(Q||P)$
$= \sum_H Q(H) \ln \frac{Q(H)}{P(H|V)}$



Minimising
$KL(P||Q)$
$= \sum_H P(H|V) \ln \frac{P(H|V)}{Q(H)}$



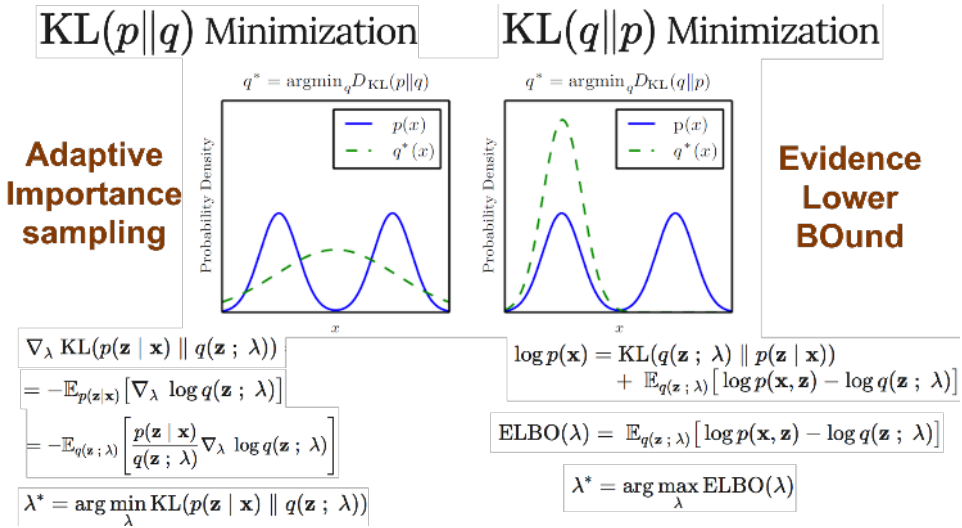
- KL 距离性质 3：最大似然函数等价最小 KL 距离

带参数分布　最小KL距离

$$
\begin{aligned}
D(\hat{p}||p_\theta) &= \sum_{x\in\mathcal{X}} \hat{p}(x)\log\frac{\hat{p}(x)}{p_\theta(x)} \\
&= -H(\hat{p}) - \sum_{x\in\mathcal{X}} \hat{p}(x)\ \log\ p_\theta(x) \\
&= -H(\hat{p}) - \frac{1}{N}\sum_{x\in\mathcal{X}}\sum_{n=1}^{N} \delta(x-x_n)\ \log\ p_\theta(x) \\
&= -H(\hat{p}) - \frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n)
\end{aligned}
$$

最大Log似然度　带参数分布

- KL 距离性质 4：ELBO vs Importance Sampling



- BM 的最小 KL 距离 (最大似然估计)

$$\mathrm{KL}(\mathbb{P}_{\text{target}} || \mathbb{P}_\theta) \equiv \sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \log \frac{\mathbb{P}_{\text{target}}(\tilde{\mathbf{x}})}{\mathbb{P}_\theta(\tilde{\mathbf{x}})} \qquad f(\theta)$$

$$= \sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \log \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) - \sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \log \mathbb{P}_\theta(\tilde{\mathbf{x}})$$

$$\nabla \log \mathbb{P}_\theta(\mathbf{x}) = \nabla \log \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{\sum_{\hat{\mathbf{x}}} \exp\left(-E_\theta(\hat{\mathbf{x}})\right)}$$

$$= -\nabla E_\theta(\mathbf{x}) - \nabla \log \sum_{\hat{\mathbf{x}}} \exp\left(-E_\theta(\hat{\mathbf{x}})\right)$$

$$= -\nabla E_\theta(\mathbf{x}) + \frac{\sum_{\hat{\mathbf{x}}} \exp\left(-E_\theta(\hat{\mathbf{x}})\right) \nabla E_\theta(\hat{\mathbf{x}})}{\sum_{\tilde{\mathbf{x}}} \exp\left(-E_\theta(\tilde{\mathbf{x}})\right)}$$

$$= -\nabla E_\theta(\mathbf{x}) + \sum_{\hat{\mathbf{x}}} \mathbb{P}_\theta(\hat{\mathbf{x}}) \nabla E_\theta(\hat{\mathbf{x}}),$$

$$\nabla f(\theta) = \sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \nabla \log \mathbb{P}_\theta(\tilde{\mathbf{x}})$$

$$= -\sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \nabla E_\theta(\tilde{\mathbf{x}}) + \sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \sum_{\hat{\mathbf{x}}} \mathbb{P}_\theta(\hat{\mathbf{x}}) \nabla E_\theta(\hat{\mathbf{x}})$$

$$= -\sum_{\tilde{\mathbf{x}}} \mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) \nabla E_\theta(\tilde{\mathbf{x}}) + \sum_{\hat{\mathbf{x}}} \mathbb{P}_\theta(\hat{\mathbf{x}}) \nabla E_\theta(\hat{\mathbf{x}})$$

$$= -\sum_{\tilde{\mathbf{x}}} (\mathbb{P}_{\text{target}}(\tilde{\mathbf{x}}) - \mathbb{P}_\theta(\tilde{\mathbf{x}})) \nabla E_\theta(\tilde{\mathbf{x}}).$$

$$\nabla f(\theta) = -\mathbb{E}_{\text{target}}\left[\nabla E_\theta(\boldsymbol{X})\right] + \mathbb{E}_\theta\left[\nabla E_\theta(\boldsymbol{X})\right]$$

- BM 的参数更新

$$\nabla f(\theta) = -\mathbb{E}_{\text{target}}\left[\nabla E_\theta(\boldsymbol{X})\right] + \mathbb{E}_\theta\left[\nabla E_\theta(\boldsymbol{X})\right]$$

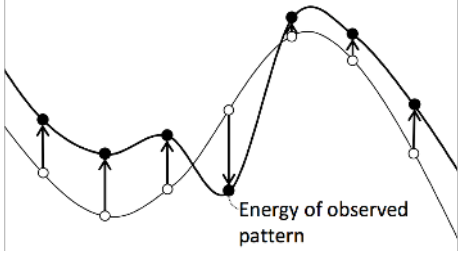
$$E_\theta(\mathbf{x}) = -\sum_{i=1}^{N} b_i\, x_i - \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} w_{i,j}\, x_i\, x_j = -\mathbf{b}^\top \mathbf{x} - \mathbf{x}^\top \mathbf{W}\mathbf{x}.$$

$$\frac{\partial}{\partial b_i} f(\theta) = \mathbb{E}_{\text{target}}[X_i] - \mathbb{E}_\theta[X_i]$$

$$\frac{\partial}{\partial w_{i,j}} f(\theta) = \mathbb{E}_{\text{target}}[X_i\, X_j] - \mathbb{E}_\theta[X_i\, X_j]$$

$$\frac{\partial}{\partial b_i} E_\theta(\mathbf{x}) = -x_i$$

$$\frac{\partial}{\partial w_{i,j}} E_\theta(\mathbf{x}) = -x_i\, x_j$$



- BM 的参数更新的直观理解

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle s_i s_j \rangle_{\mathbf{v}} - \langle s_i s_j \rangle_{model}$$

Derivative of log probability of one training vector, v under the model.

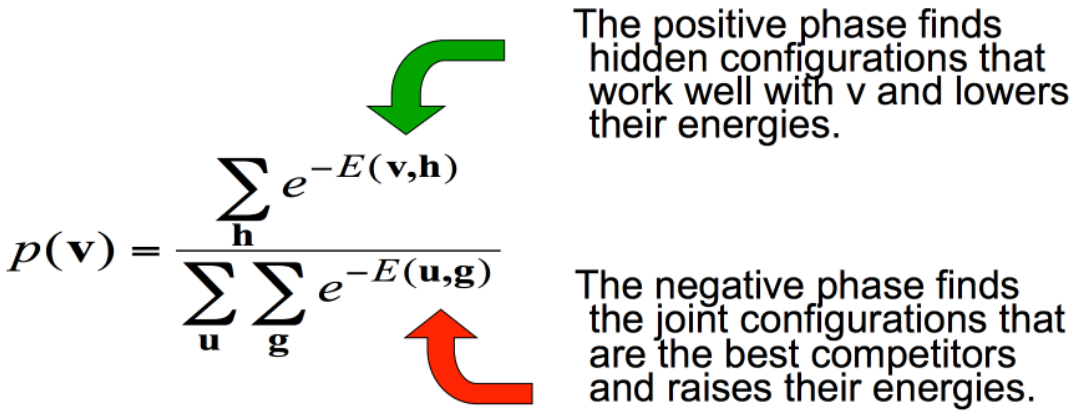
Expected value of product of states at thermal equilibrium when v is clamped on the visible units

Expected value of product of states at thermal equilibrium with no clamping

$$\Delta w_{ij} \propto \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

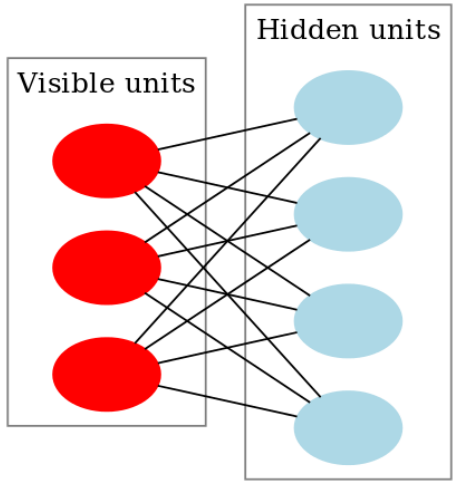
1. 正向：在可见端固定 x，等稳定后，计算期望值
2. 反向：松开可见端 x，等稳定后，计算期望值

- BM 的最大似然的直观理解



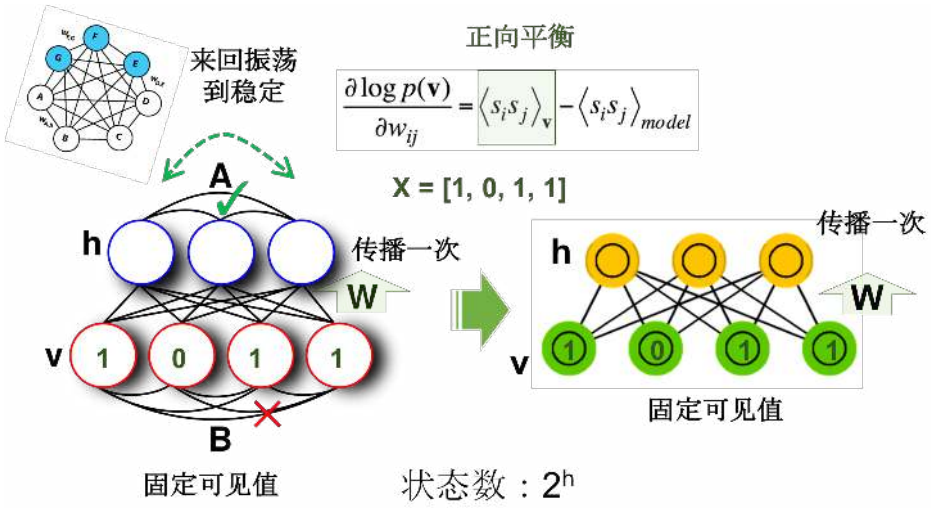
1. 正向：找到隐藏层参数，修正降低能量
2. 反向：找到联合参数，修正提高能量

- 受限玻尔兹曼机 Restricted Boltzmann Machines

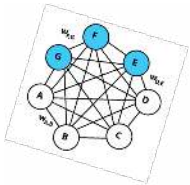


BM 过于复杂，难以训练！

- RBM 的好处：正向平衡计算一次

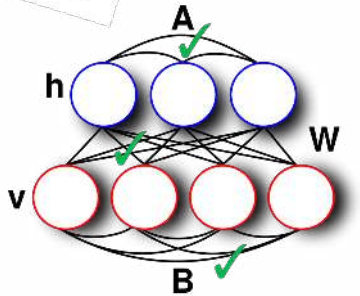


- RBM 的好处：反向平衡下上来回振荡

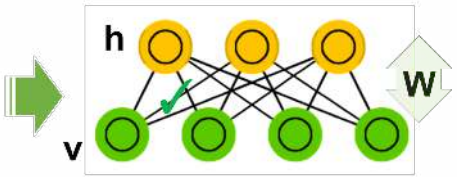


反向平衡

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle s_i s_j \rangle_{\mathbf{v}} - \langle s_i s_j \rangle_{model}$$



全部来回振荡到稳定　状态数：2^{h+v}



上下来回振荡到稳定

- RBM 的好处，条件独立性

RMB 能量函数 (分离了 V、H 内部乘法)：

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

矩阵形式：

$$E(v,h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h$$

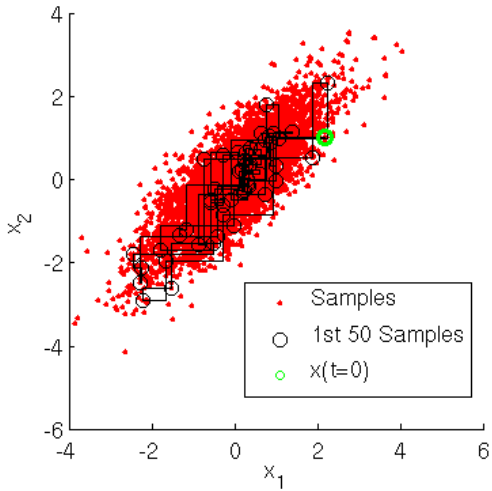
可见层条件独立性：

$$P(v|h) = \prod_{i=1}^{m} P(v_i|h)$$

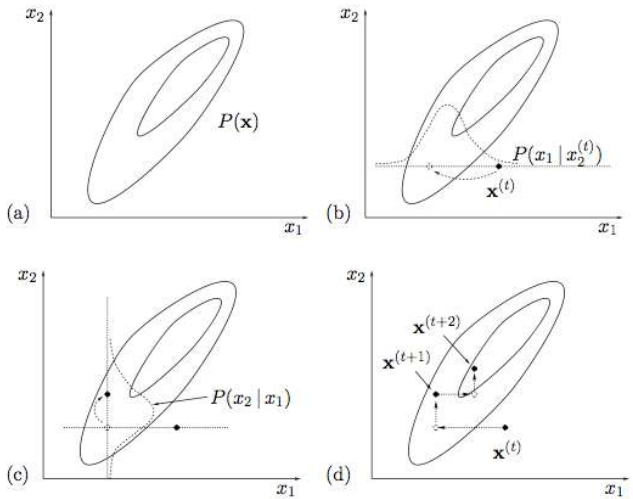
隐藏层条件独立性：

$$P(h|v) = \prod_{j=1}^{n} P(h_j|v)$$

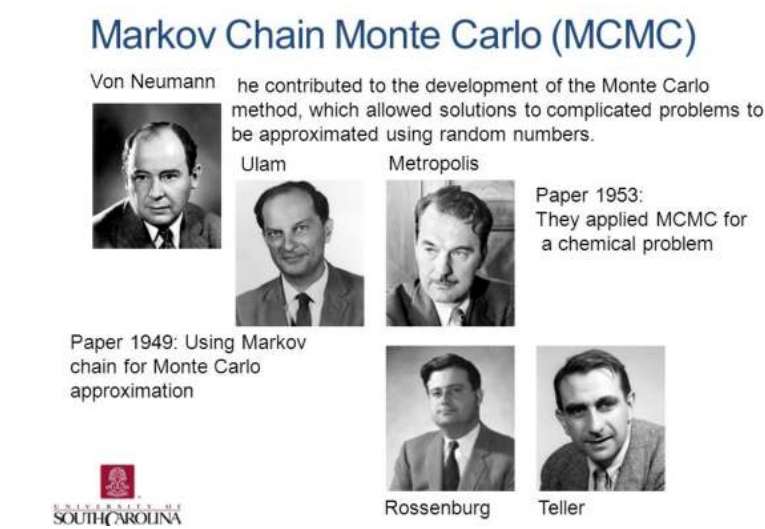
- 吉布斯采样求解：Gibbs Sampling



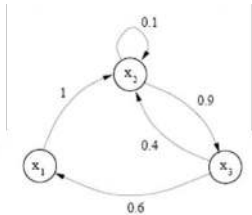
- 吉布斯采样求解：Gibbs Sampling



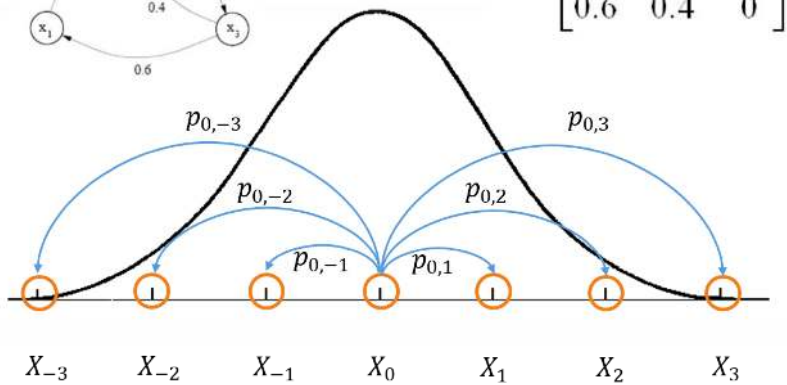
- 蒙特卡洛马尔科夫链：MCMC



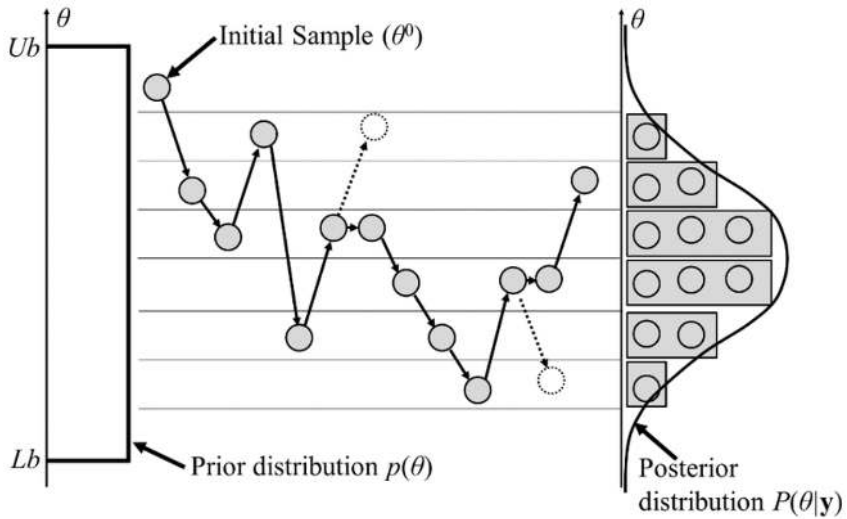
- MCMC 迁移



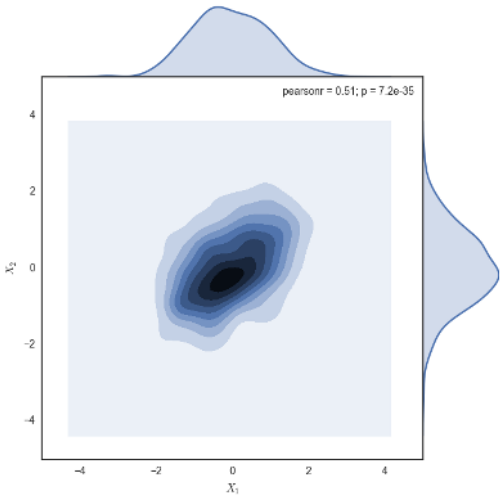
$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$



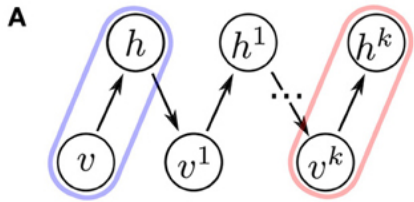
- MCMC 分布抽样


θ
Ub
Lb
Initial Sample (θ^0)
Prior distribution $p(\theta)$
θ
Posterior
distribution $P(\theta|\mathbf{y})$

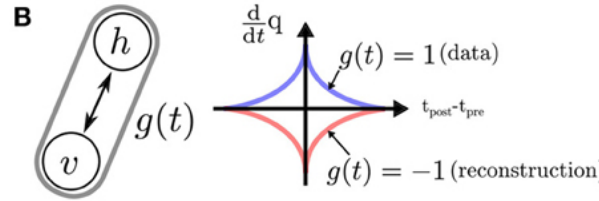
- Gibbs 抽样收敛：两个独立变量核心分布



- RBM Gibbs 抽样



$$\Delta w \propto \langle vh \rangle_{\text{data}} - \langle v^k h^k \rangle_{\text{recon}}$$



$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{q} = g(t)\mathrm{STDP}(v(t), h(t))$$

- 正向反向平衡的重新解读

$$\frac{\partial E(\mathbf{X};\Theta)}{\partial \Theta} = \frac{\partial \log Z(\Theta)}{\partial \Theta} - \frac{1}{K}\sum_{i=1}^{K}\frac{\partial \log f(x_i;\Theta)}{\partial \Theta}$$

$$= \frac{\partial \log Z(\Theta)}{\partial \Theta} - \left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}}$$

反向平衡

$\mathbf{X}^0 \equiv \mathbf{X}$ 输入

$$\frac{\partial E(\mathbf{X};\Theta)}{\partial \Theta} = \left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^\infty} - \left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^0}$$

重建

$$\Theta_{t+1} = \Theta_t + \eta\left(\left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^0} - \left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{\mathbf{X}^1}\right)$$

$$\frac{\partial \log Z(\Theta)}{\partial \Theta} = \frac{1}{Z(\Theta)}\frac{\partial Z(\Theta)}{\partial \Theta}$$

$$= \frac{1}{Z(\Theta)}\frac{\partial}{\partial \Theta}\int f(x;\Theta)\,dx$$

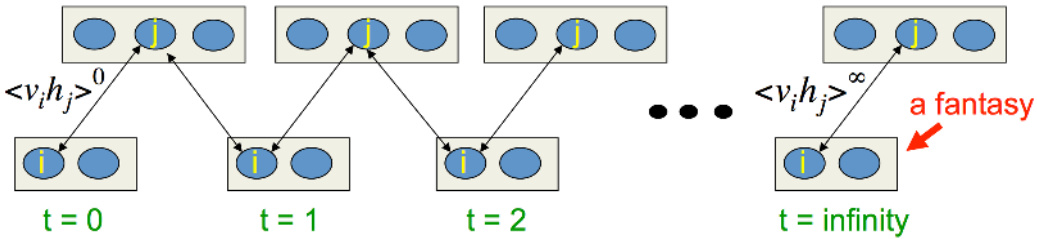
$$= \frac{1}{Z(\Theta)}\int \frac{\partial f(x;\Theta)}{\partial \Theta}\,dx$$

$$= \frac{1}{Z(\Theta)}\int f(x;\Theta)\frac{\partial \log f(x;\Theta)}{\partial \Theta}\,dx$$

$$= \int p(x;\Theta)\frac{\partial \log f(x;\Theta)}{\partial \Theta}\,dx$$

$$= \left\langle \frac{\partial \log f(x;\Theta)}{\partial \Theta} \right\rangle_{p(x;\Theta)}$$

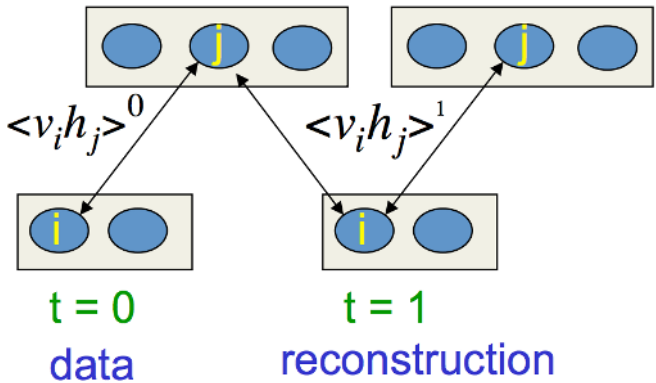
- RBM Gibbs 抽样：可见层一端视角



Start with a training vector on the visible units. Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

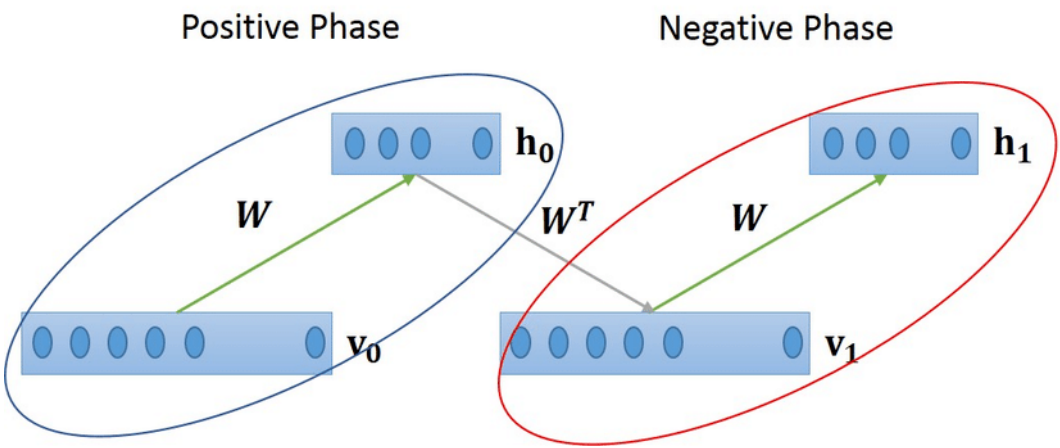
$$\Delta w_{ij} = \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^\infty \right)$$

- RBM Gibbs 1 步抽样：重建视角

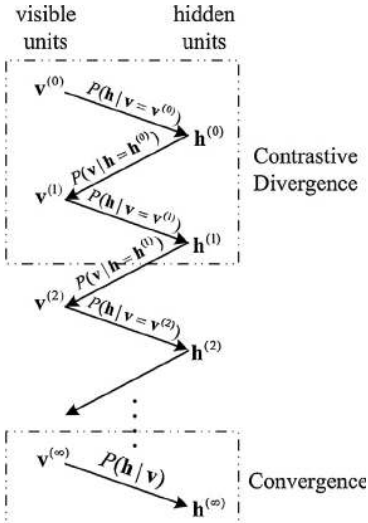


$$\Delta w_{ij} = \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^1 \right)$$

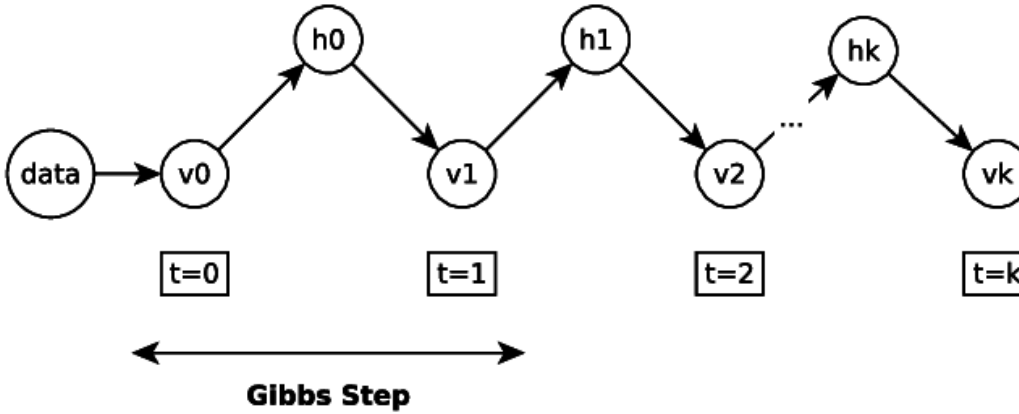
- Contrastive Divergence: Gibbs 1 步抽样



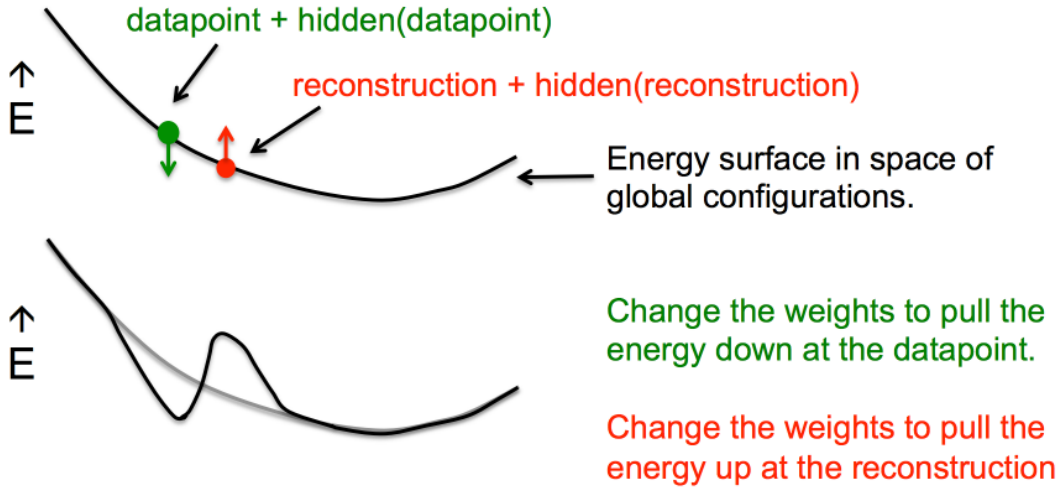
- Contrastive Divergence: Gibbs 1 步抽样



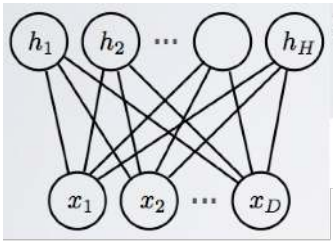
- CD-K: Gibbs K 步抽样



- CD-K: 全局优化观


datapoint + hidden(datapoint)
reconstruction + hidden(reconstruction)
E
Energy surface in space of global configurations.
E
Change the weights to pull the energy down at the datapoint.
Change the weights to pull the energy up at the reconstruction.

- CD 算法具体更新

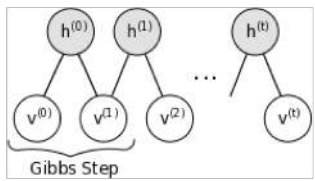


$$\frac{\partial - \log p(\mathbf{x}^{(t)})}{\partial \theta} = \underbrace{\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)}, \mathbf{h})}{\partial \theta} \middle| \mathbf{x}^{(t)}\right]}_{\text{positive phase}} - \underbrace{\mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right]}_{\text{negative phase}}$$

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$



网络权重 $\mathbf{W} \Longleftarrow \mathbf{W} + \alpha\left(\mathbf{h}(\mathbf{x}^{(t)})\,\mathbf{x}^{(t)\top} - \mathbf{h}(\tilde{\mathbf{x}})\,\tilde{\mathbf{x}}^{\top}\right)$

隐藏层偏差 $\mathbf{b} \Longleftarrow \mathbf{b} + \alpha\left(\mathbf{h}(\mathbf{x}^{(t)}) - \mathbf{h}(\tilde{\mathbf{x}})\right)$

输入层偏差 $\mathbf{c} \Longleftarrow \mathbf{c} + \alpha\left(\mathbf{x}^{(t)} - \tilde{\mathbf{x}}\right)$

Sigmoid 函数

- Product of Experts (PoE) 框架

似然函数

$$P(x|\{\theta_j\}) = \frac{1}{Z}\prod_{j=1}^{M} f_j(x|\theta_j)$$

$$Z = \int \mathrm{d}x \prod_{j=1}^{M} f_j(x|\theta_j)$$

Log似然函数

$$L(\{\theta_j\}|\{x_n\}) = \sum_{n=1}^{N}\sum_{j=1}^{M} \log f_j(x_n|\theta_j) - N\log Z$$

梯度

$$\nabla_j L = \sum_{n=1}^{N} \nabla_j \log f_j(x_n) - N\langle \nabla_j \log f_j(x)\rangle_{P(x)}$$

梯度下降

$$\theta_j \to \theta_j + \eta\nabla_j L$$

Restricted Boltzmann Machines (RBM)

$$P(x,h) = \frac{1}{Z}\exp\left(\sum_i \alpha_i x_i + \sum_j \beta_j h_j + \sum_{ij} W_{ij}x_i h_j\right)$$

$$P(x) = \frac{1}{\tilde{Z}}\prod_i \exp(\alpha_i x_i)\prod_j \left(1 + \exp(\beta_j + \sum_i W_{ij}x_i)\right)$$

Independent Components Analysis (ICA)

$$P(x|\{w_j\}) = |\det(W)|\prod_{j=1}^{M} p_j\left(\sum_i w_{ji}x_i\right)$$

- PoE vs MoE (Mixture of Experts)

Mixture Model

$$p\left(\vec{d} \mid \theta_1, \ldots, \theta_n\right) = \sum_m \alpha_m f_m\left(\vec{d} \mid \theta_m\right)$$

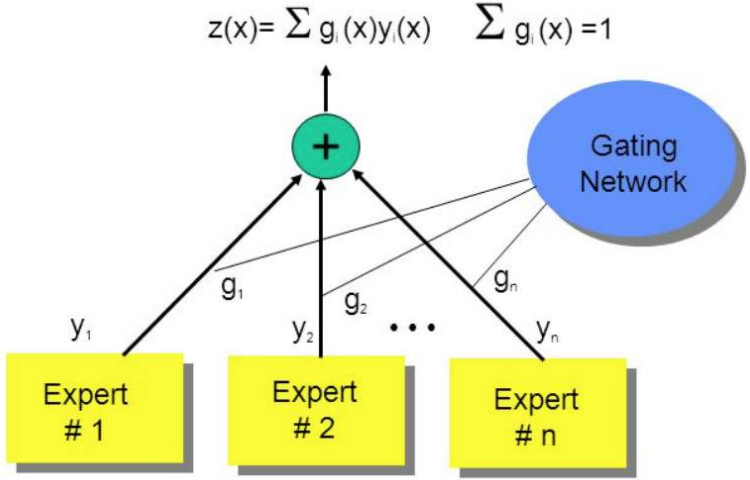
Use EM to learn parameters

Product of Experts

$$p\left(\vec{d} \mid \theta_1, \ldots, \theta_n\right) = \frac{\prod_m f_m\left(\vec{d} \mid \theta_m\right)}{\sum_{\vec{c}} \prod_m f_m\left(\vec{c} \mid \theta_m\right)}$$

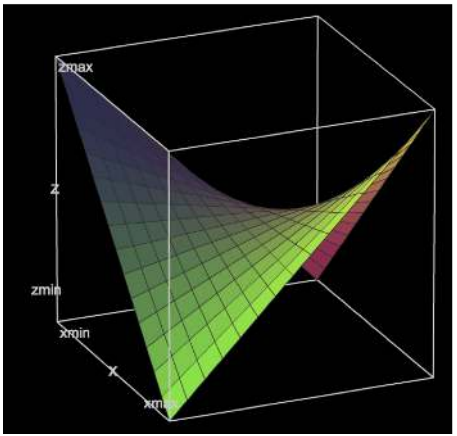
Use Contrastive Divergence to learn parameters.

- MoE (Mixture of Experts)

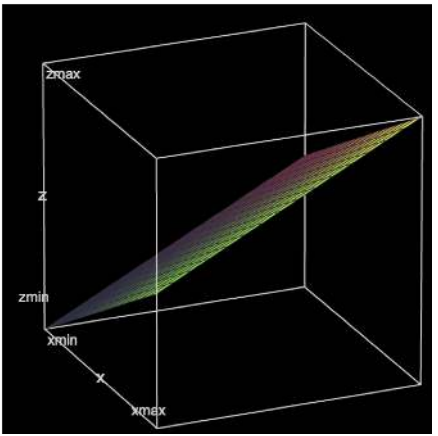


- PoE vs MoE

Z = X * Y

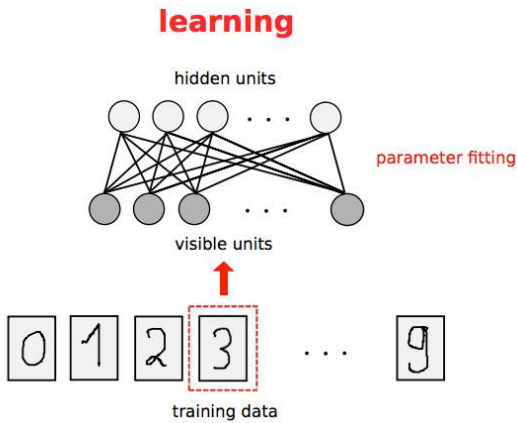


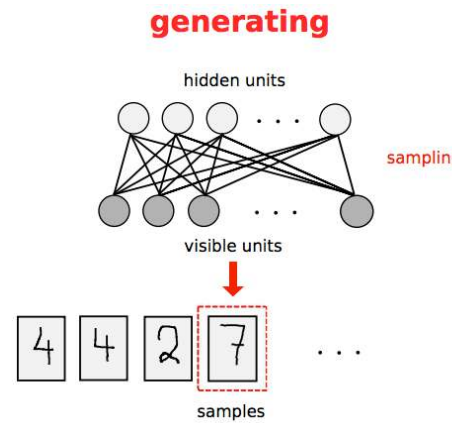
Z = X + Y



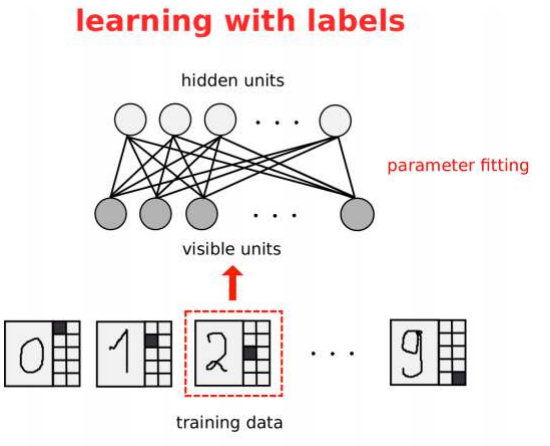
乘法比加法更为尖锐，加法类似投票

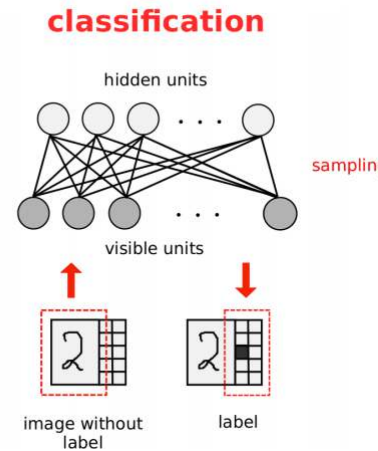
- RMB 做无监督学习


learning
hidden units
parameter fitting
visible units
training data

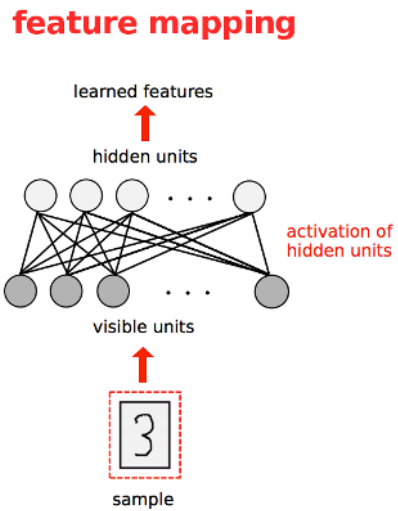

generating
hidden units
sampling
visible units
samples

- RMB 做有监督学习

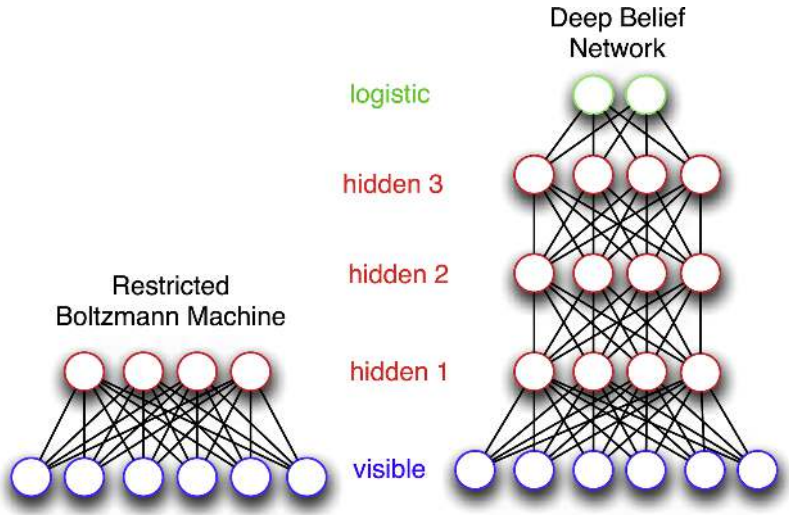

learning with labels
hidden units
parameter fitting
visible units
training data


classification
hidden units
sampling
visible units
image without label
label

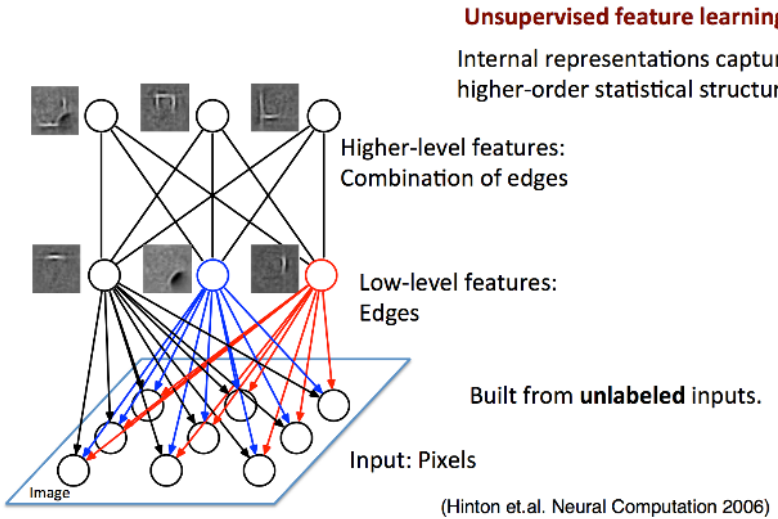
- RMB 做特征提取


feature mapping
learned features
hidden units
activation of
hidden units
visible units
3
sample

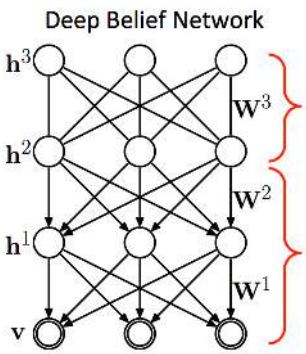
- Deep Belief Network (DBN)



- DBN 分层特征表示


Unsupervised feature learning.
Internal representations capture higher-order statistical structure
Higher-level features: Combination of edges
Low-level features: Edges
Built from unlabeled inputs.
Input: Pixels
Image
(Hinton et.al. Neural Computation 2006)

- DBN = Sigmoid Belief Network + RBM



The joint probability distribution factorizes:

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

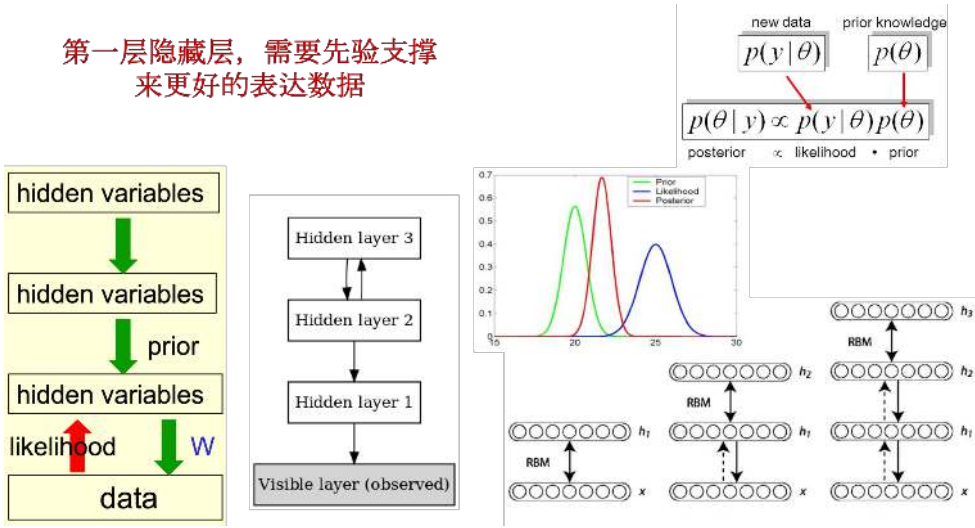
Sigmoid Belief Network　　RBM

$$P(\mathbf{h}^2, \mathbf{h}^3) = \frac{1}{\mathcal{Z}(W^3)} \exp\left[\mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$

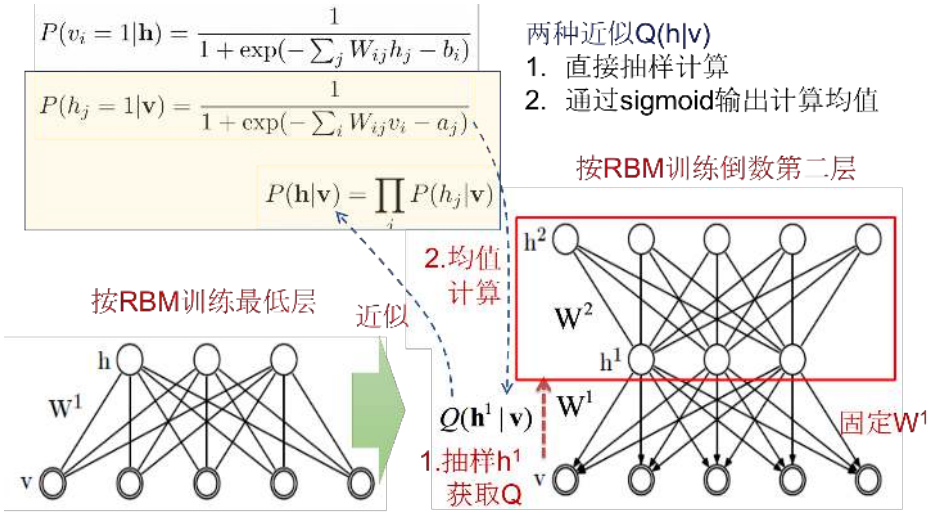
$$P(\mathbf{h}^1|\mathbf{h}^2) = \prod_j P(h_j^1|\mathbf{h}^2) \qquad P(h_j^1 = 1|\mathbf{h}^2) = \frac{1}{1 + \exp\left(-\sum_k W_{jk}^2 h_k^2\right)}$$

$$P(\mathbf{v}|\mathbf{h}^1) = \prod_i P(v_i|\mathbf{h}^1) \qquad P(v_i = 1|\mathbf{h}^1) = \frac{1}{1 + \exp\left(-\sum_j W_{ij}^1 h_j^1\right)}$$

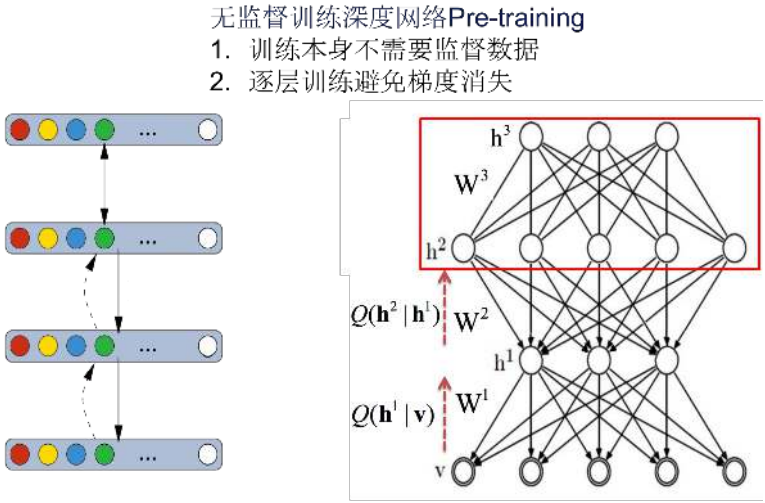
- DBN 增强第一层隐藏层表达能力



- DBN Layer-Wise Training



- Greedy-Layer Wise Pre-Training



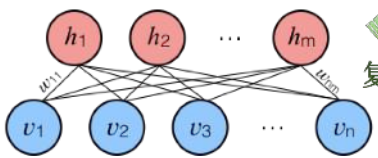
- RBM 等价无限 Sigmoid Belief Network

$$P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

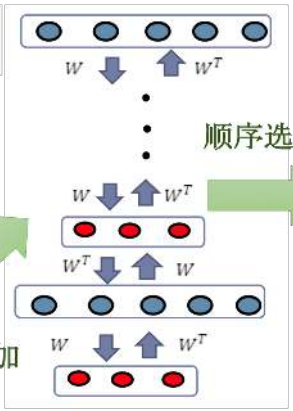
$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

$$P(v_i|\boldsymbol{h}) = sigm(\sum_j w_{ij} h_j + b_i)$$

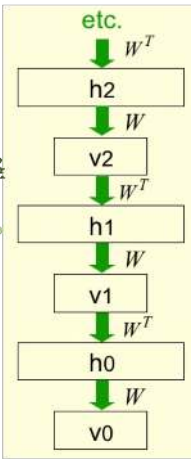
$$P(h_i|\boldsymbol{v}) = sigm(\sum_i w_{ij} v_i + c_j)$$



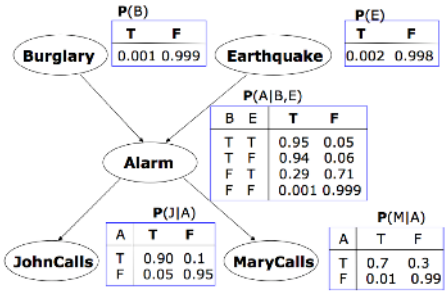
复制叠加



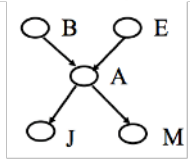
顺序选择



- 贝叶斯网络 (Bayes Belief Network)



$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$



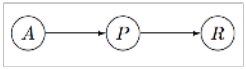
$$P(B=T,E=T,A=T,J=T,M=F)=$$

按信念Belief传播的方式计算条件概率的乘积

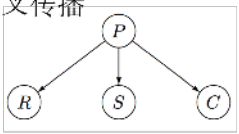
$$=P(J=T \mid B=T,E=T,A=T,M=F)P(B=T,E=T,A=T,M=F)$$
$$=P(J=T \mid A=T)P(B=T,E=T,A=T,M=F)$$
$$P(M=F \mid B=T,E=T,A=T)P(B=T,E=T,A=T)$$
$$P(M=F \mid A=T)P(B=T,E=T,A=T)$$
$$P(A=T \mid B=T,E=T)P(B=T,E=T)$$
$$P(B=T)P(E=T)$$
$$=P(J=T \mid A=T)P(M=F \mid A=T)P(A=T \mid B=T,E=T)P(B=T)P(E=T)$$

- Sigmoid Belief Network

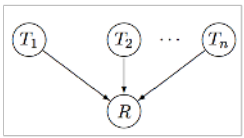
顺序传播



分叉传播



聚合传播



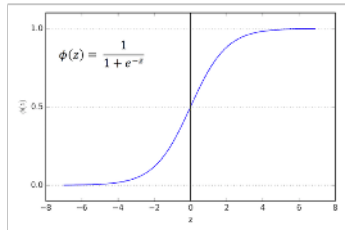
$$\mathrm{p}(\mathbf{U}) = \mathrm{p}(A_1 = a_1, A_2 = a_2, \ldots, A_n = a_n) = \prod_i \mathrm{p}(a_i \mid pa(A_i))$$

Binomial 二项分布

$$L(\boldsymbol{\mu} \mid Y) = \prod_{i=1}^{n} \left(1_{y_i=1}(\mu_i) + 1_{y_i=0}(1-\mu_i)\right)$$

$$\ln\left(\frac{P}{1-P}\right) = \alpha + \beta x \qquad \frac{P}{1-P} = e^{\alpha+\beta x}$$

Sigmoid 激活函数

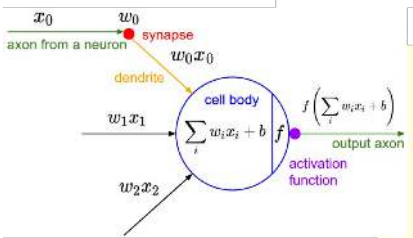


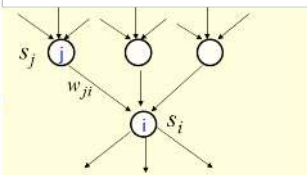
Sigmoid 信念网络

- Sigmoid Belief Network 计算

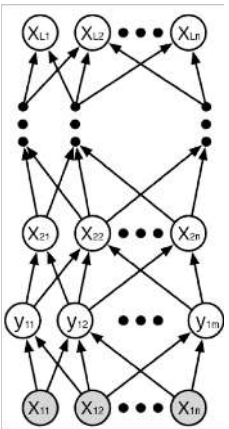
$$\mathbf{P}(X_1, X_2, .., X_n) = \prod_{i=1,..n} \mathbf{P}(X_i \mid pa(X_i))$$

$$p(y_{k:} = 1 | x_{k+1:}) = \sigma\left(W^\top x_{k+1:} + c\right)$$
$$p(x_{k:} = 1 | y_{k:}) = \sigma\left(W y_{k:} + b\right)$$

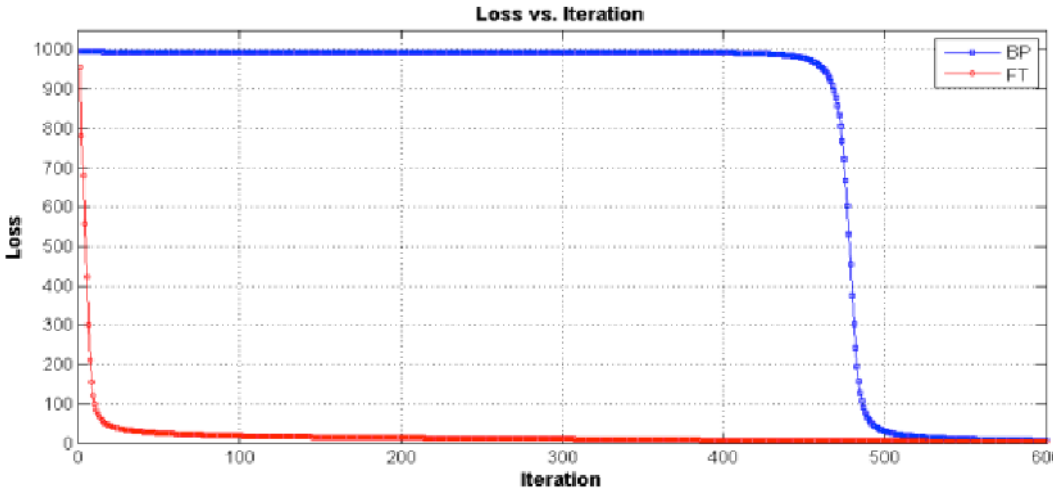




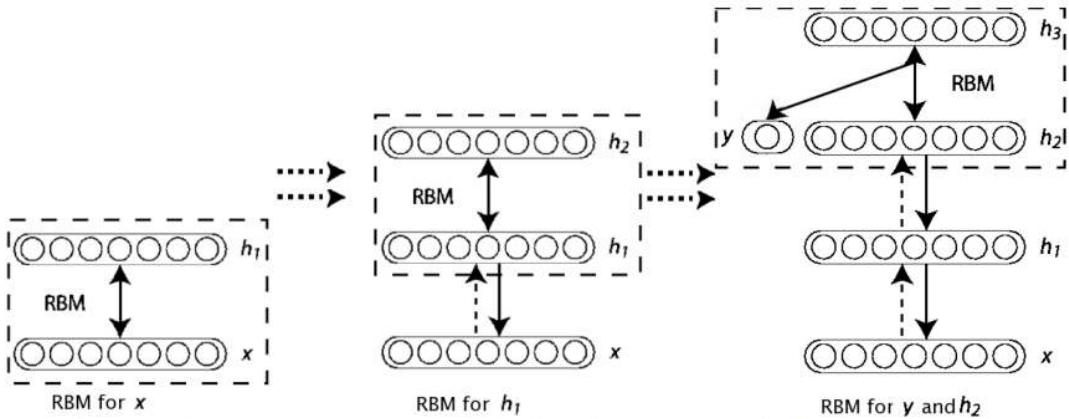
$$p_i \equiv p(s_i = 1) = \frac{1}{1 + \exp(-\sum_j s_j w_{ji})}$$



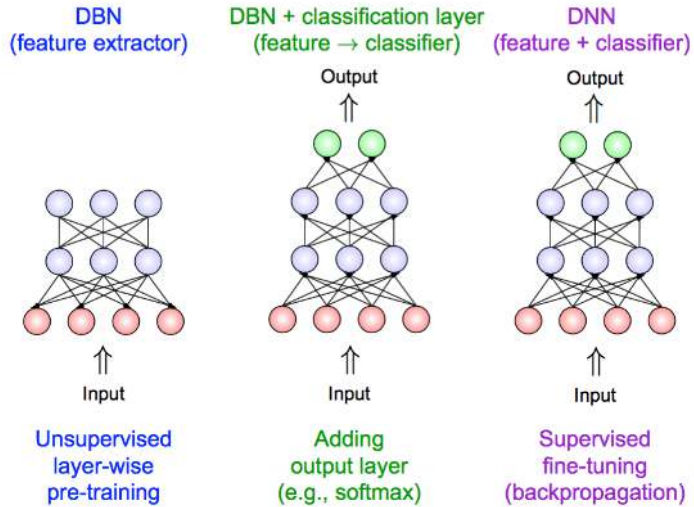
- Pre-Training 加速了收敛



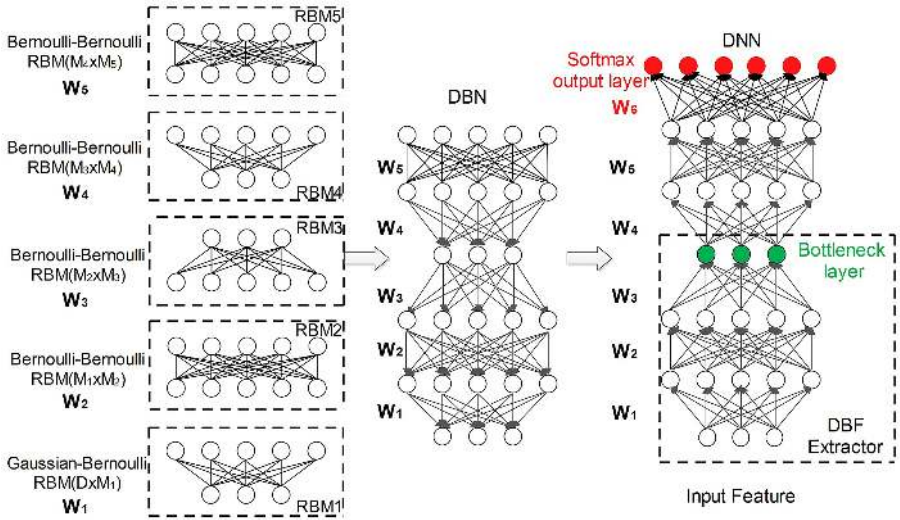
- DBN 做分类学习


RBM
RBM
RBM
h_3
h_2
h_1
x
y
RBM for x
RBM for h_1
RBM for y and h_2
Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)

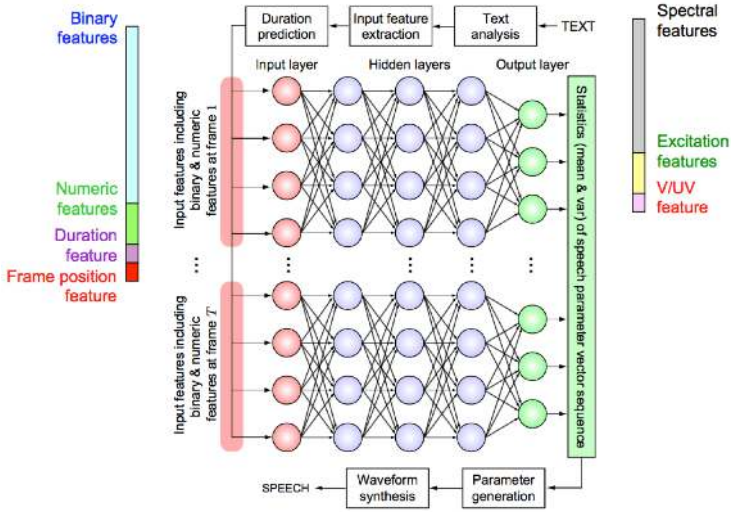
- DBN 做特征学习


DBN
(feature extractor)
DBN + classification layer
(feature → classifier)
DNN
(feature + classifier)
Output
Output
Input
Input
Input
Unsupervised
layer-wise
pre-training
Adding
output layer
(e.g., softmax)
Supervised
fine-tuning
(backpropagation)

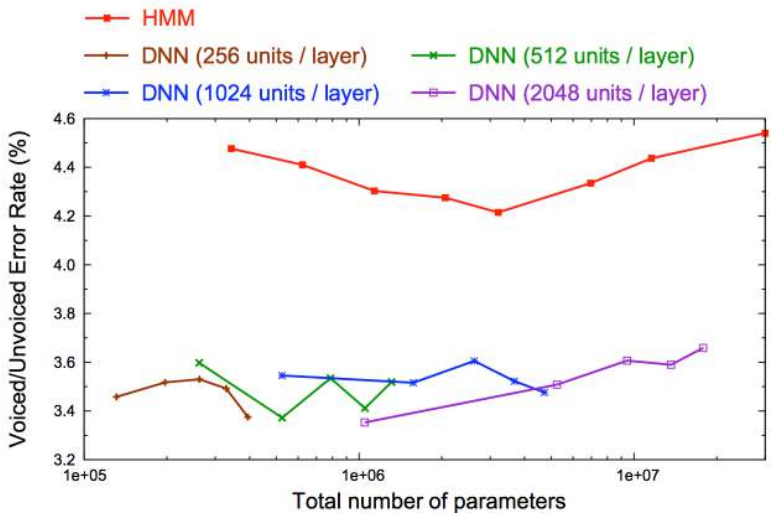
- DNN Deep Neural Network



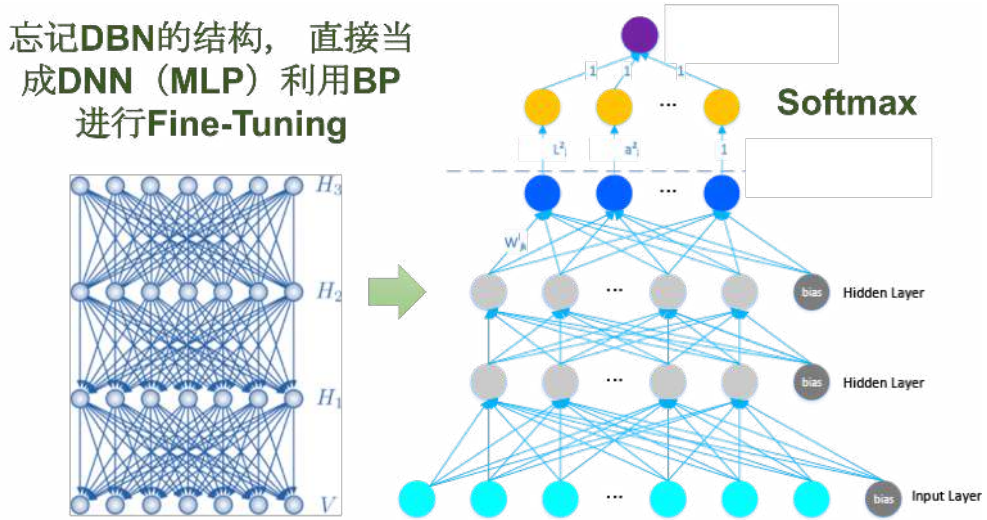
- DNN 在语言识别


Binary features
Numeric features
Duration feature
Frame position feature
Duration prediction
Input feature extraction
Text analysis
TEXT
Input layer
Hidden layers
Output layer
Input features including binary & numeric features at frame 1
Input features including binary & numeric features at frame T
Statistics (mean & var) of speech parameter vector sequence
Spectral features
Excitation features
V/UV feature
SPEECH
Waveform synthesis
Parameter generation

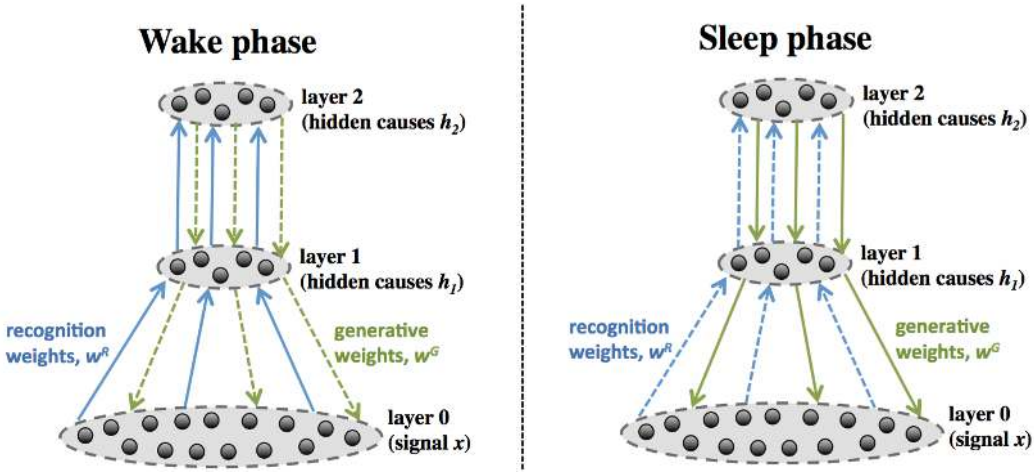
- DNN 在语言识别大获成功



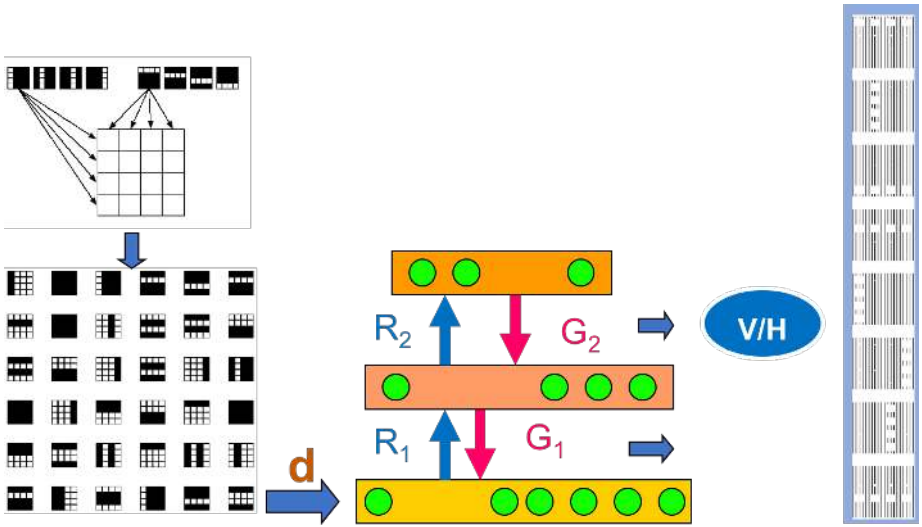
- Fine-Tuning 有监督学习：BP 算法



- Fine-Tuning 无监督学习：wake-sleep



- wake-sleep 算法



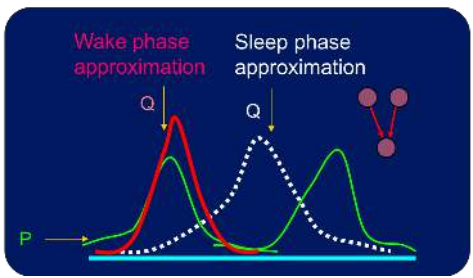
- wake-sleep 算法：KL(P, Q) vs KL(Q, P)

$$\log P[d;\mathcal{G}] \geq \log P[d;\mathcal{G}] - \mathrm{KL}(\mathrm{Q}[\mathrm{h}|\mathrm{d};\mathcal{R}],\mathrm{P}[\mathrm{h}|\mathrm{d};\mathcal{G}])$$

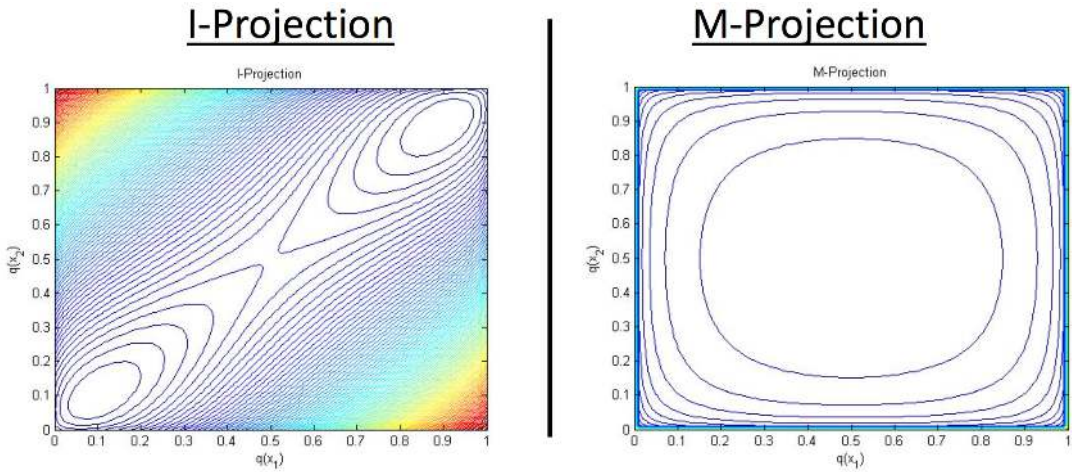
$$\equiv -\mathcal{F}(d;\mathcal{R},\mathcal{G})$$

Free energy

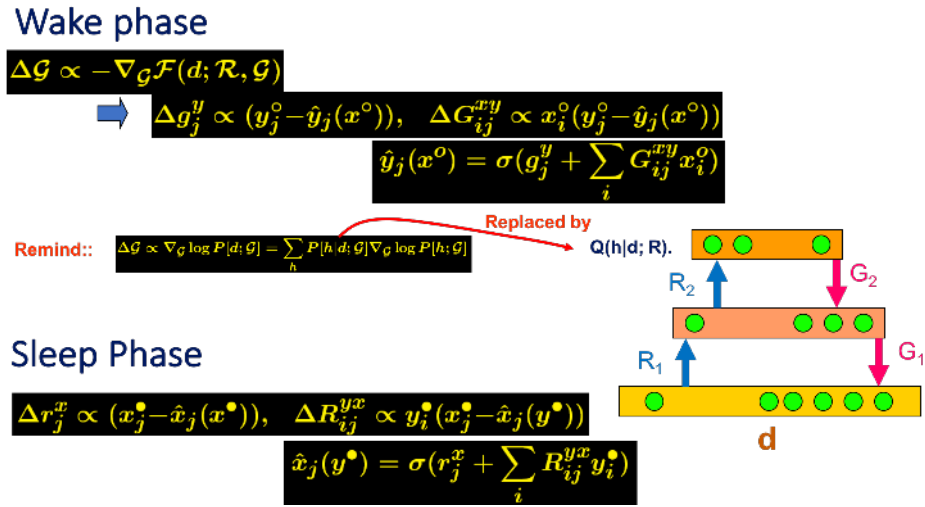
$$\mathcal{F}(d;\mathcal{R},\mathcal{G}) = -\log P[d;\mathcal{G}] + \mathrm{KL}(\mathrm{Q}[\mathrm{h}|\mathrm{d};\mathcal{R}],\mathrm{P}[\mathrm{h}|\mathrm{d};\mathcal{G}])$$



- KL(P, Q): I-Projection vs KL(Q, P): M-Projection



- wake-sleep 算法：认知和生成

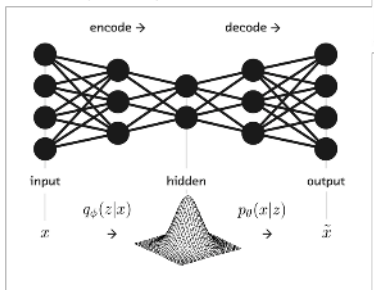


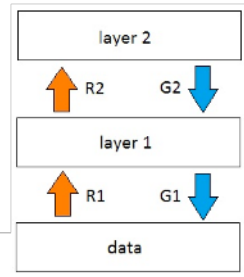
- wake-sleep 最新衍生：VAE(认知）和 GAN(生成)

Wake 阶段

自下而上从数据学习
认知模型

最大KL(Q, P)， 衍生到VAE

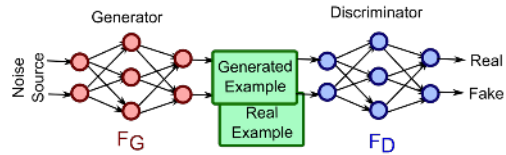




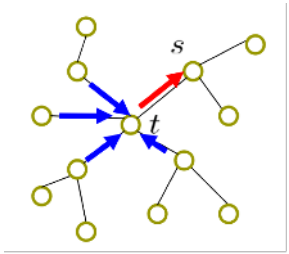
Sleep 阶段

自上而下从模型
再生数据

最大KL(P, Q)， 衍生到GAN

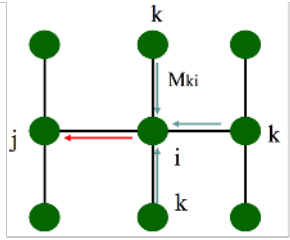


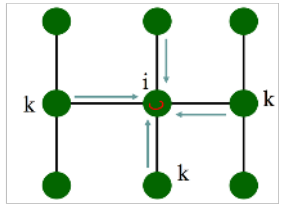
- Fine-Tuning 无监督学习：mean-field approximation



单向网络

$$p(x_1, \cdots, x_m) = \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t)$$





多源单向网络

$$M_{i \to j}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_k M_{k \to i}(x_i)$$

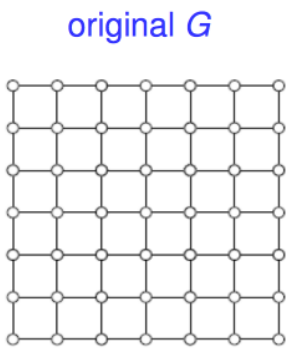
external evidence

Compatibilities (interactions)

双向网络

$$b_i(x_i) \propto \psi_i(x_i) \prod_k M_k(x_k)$$

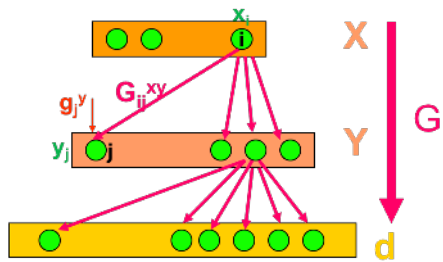
- 什么是 mean-field approximation？近似 p(x)



找个可以分解的 q(x) 使得 KL(p(x)||q(x)) 最小。

- 为什么可以使用 mean-field approximation?

$$p(y_j = 1) = \frac{1}{1 + \exp(-\sum_i x_i G_{ij}^{xy} - g_j^y)}$$

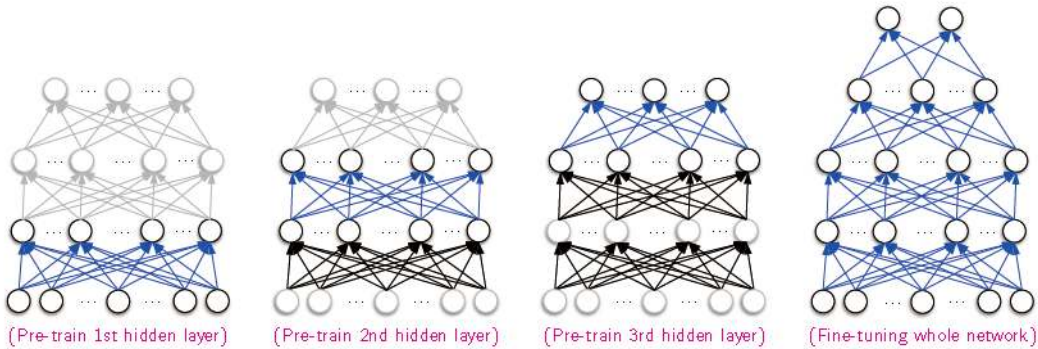


条件概率都是可以
由独立子单元乘积

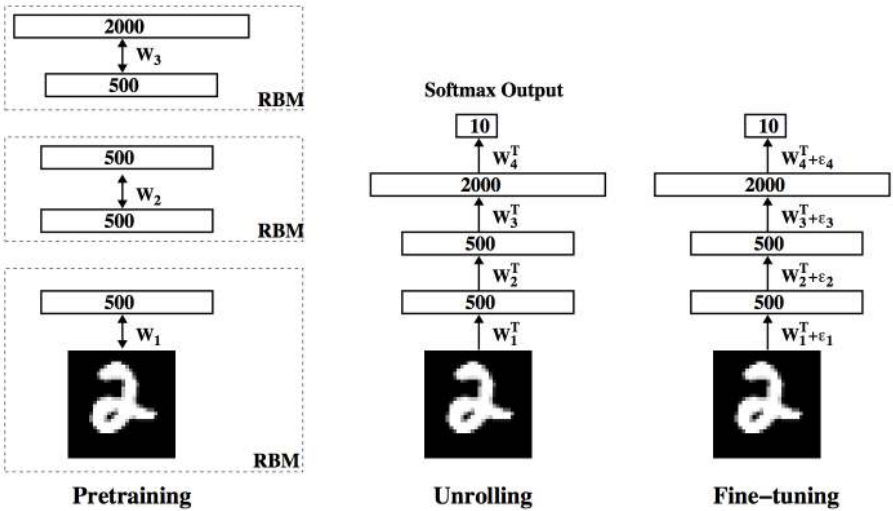
$$P[d|y;\mathcal{G}] = \prod_k P[d_k|y;\mathcal{G}]$$

$$P[x;\mathcal{G}] = \prod_i P[x_i;\mathcal{G}], \quad P[y|x;\mathcal{G}] = \prod_j P[y_j|x;\mathcal{G}]$$

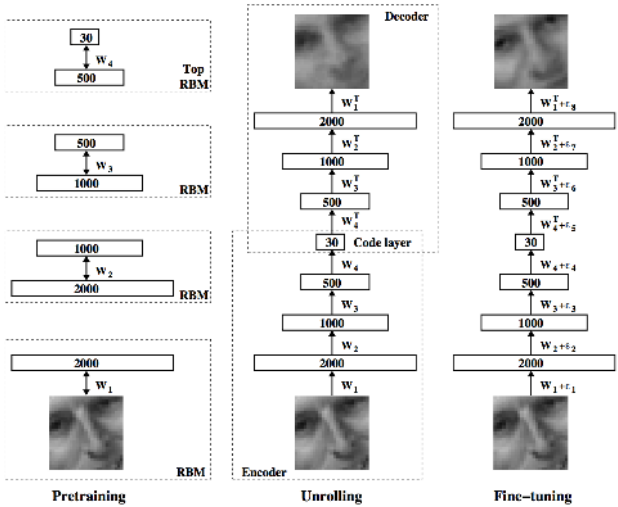
- DNN 有监督学习整个过程



- DNN 图像识别


2000
W3
500
RBM
500
W2
500
RBM
500
W1
RBM
Pretraining
Softmax Output
10
W4T
2000
W3T
500
W2T
500
W1T
Unrolling
10
W4T+ε4
2000
W3T+ε3
500
W2T+ε2
500
W1T+ε1
Fine−tuning

- Deep Auto-Encoder



- DNN 研究带来了 ReLU 的诞生

Sigmoid → 权重共享

不同偏差：

$$-0.5, -1.5, -2.5, ... -(N-0.5)$$

概率求和：

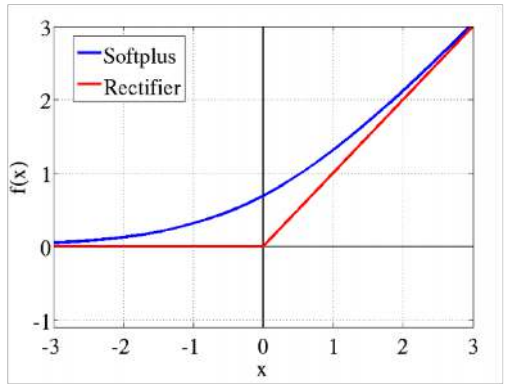
$$\sum_{i=1}^{\infty} \sigma(x-i+0.5) \approx \log(1+e^x)$$

softplus 函数和导数

$$f(x) = \ln[1+\exp(x)]$$

$$f'(x) = \exp(x)/[\exp(x)+1] = 1/[1+\exp(-x)]$$

ReLU Rectified linear units

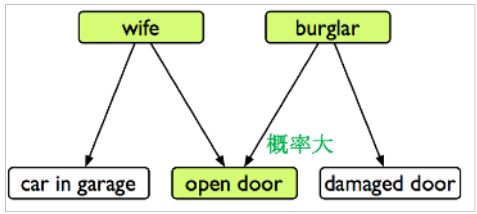


- DBN 潜在问题

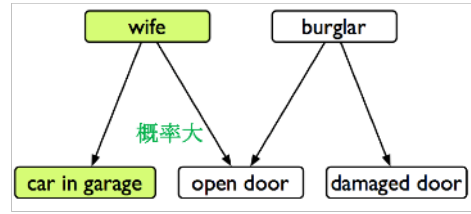
1. 由于是单向网络，存在 “explaining away” 的问题
   - 对于不确定性数据或者模糊数据潜在问题
   - 稳定性不强
2. 贪心逐层优化，很难全局最优
3. 只有前向推理，没有反向推理

- explaining away 现象

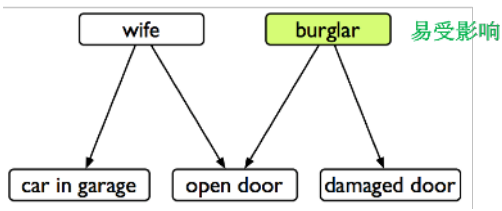
开门的两种可能性



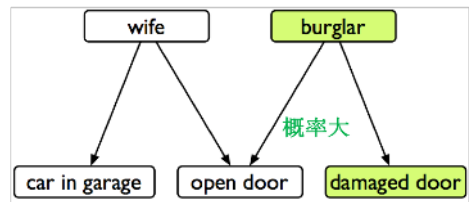
如果一个相邻条件独立现象成立



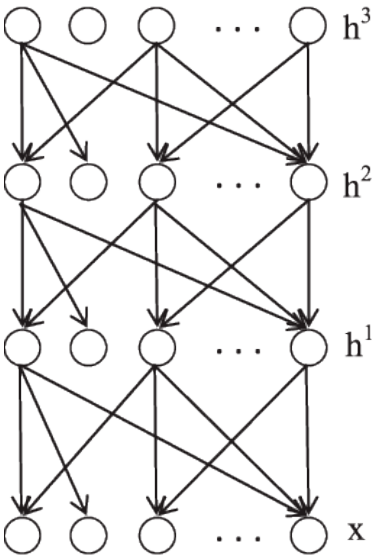
会对最初的理解改动



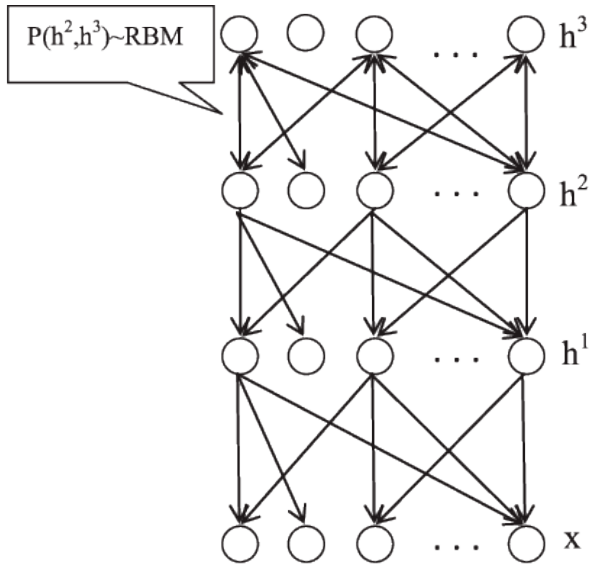
如果一个相邻条件独立现象成立



- Deep Boltzmann Machine (DBM)



Sigmoid Belief Networks

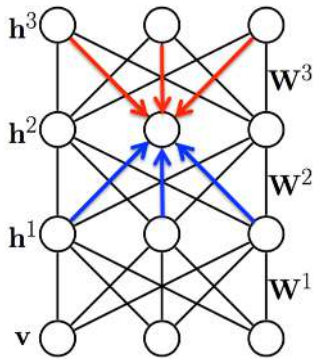


Deep Belief Networks

- DBM：一种随机场 Random Field 无向图

$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$

Deep Boltzmann Machine



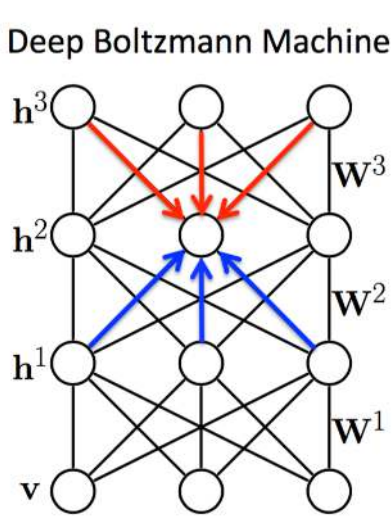
$\theta = \{W^1, W^2, W^3\}$ model parameters

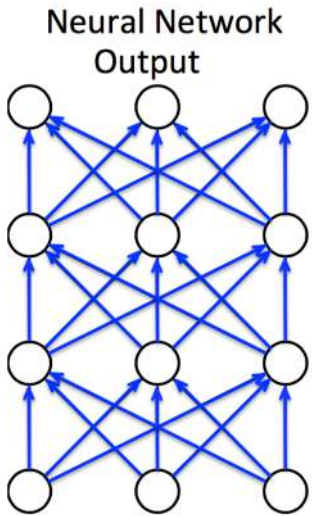
- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

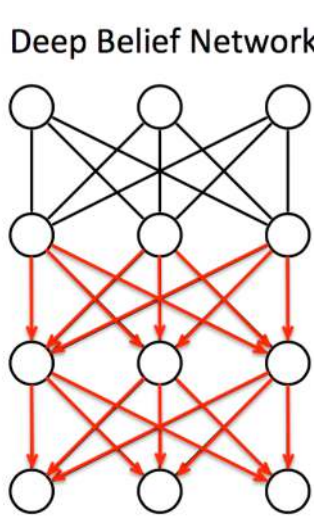
$$P(h_j^2 = 1|\mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W_{kj}^3 h_k^3 + \sum_m W_{mj}^2 h_m^1\right)$$

Top-down　　Bottom-up

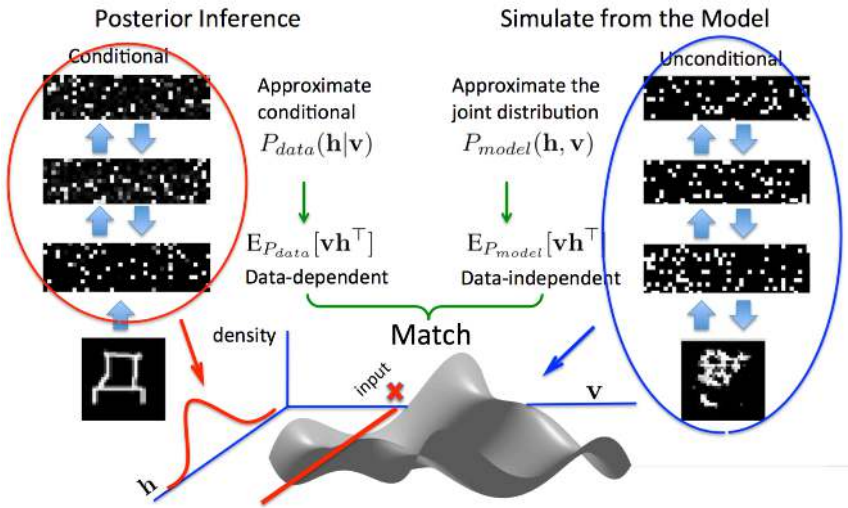
- 神经网络、概率图模型 (贝叶斯网络、随机场)



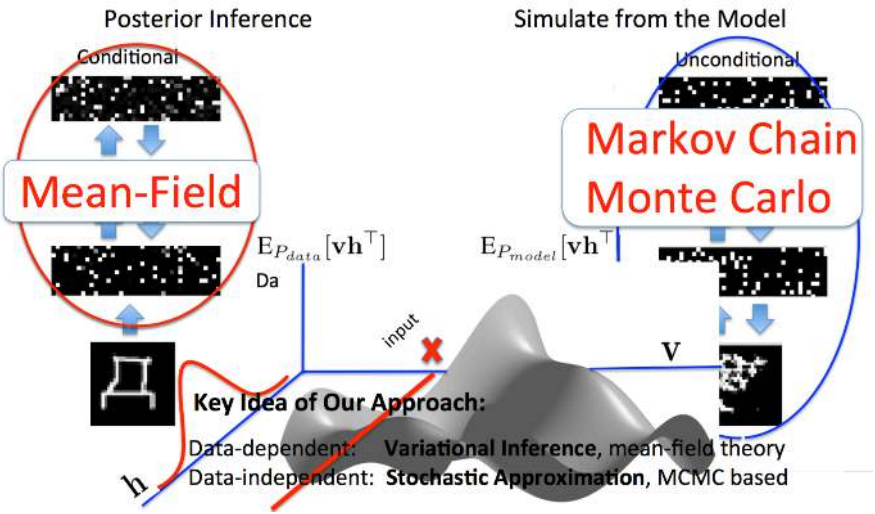




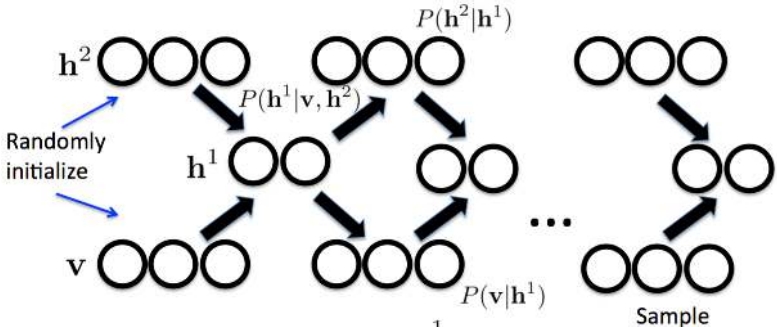
- 推理和数据重建都有


Posterior Inference
Simulate from the Model
Conditional
Unconditional
Approximate conditional
$P_{data}(\mathbf{h}|\mathbf{v})$
Approximate the joint distribution
$P_{model}(\mathbf{h},\mathbf{v})$
$\mathrm{E}_{P_{data}}[\mathbf{v}\mathbf{h}^\top]$
Data-dependent
$\mathrm{E}_{P_{model}}[\mathbf{v}\mathbf{h}^\top]$
Data-independent
Match
density
input
h
v

- 变分推理和 MCMC 方法



- 随机采样计算 CD 算法

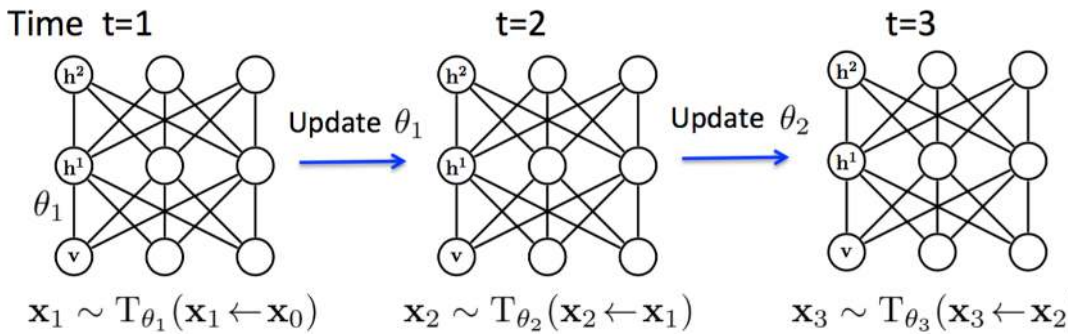


$$P(h_m^1 = 1|\mathbf{v},\mathbf{h}^2) = \frac{1}{1+\exp(-\sum_i W_{im}^1 v_i - \sum_j W_{mj}^2 h_j^2)}$$

$$P(h_j^2 = 1|\mathbf{h}^1) = \frac{1}{1+\exp(-\sum_m W_{mj}^2 h_m^1)}$$

$$P(v_i = 1|\mathbf{h}^1) = \frac{1}{1+\exp(-\sum_m W_{im}^1 h_m^1)}$$

- 顺序逐层更新



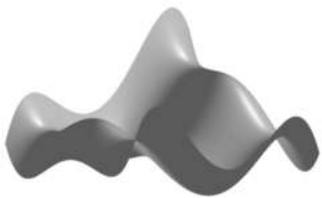
Update θ_t and $\mathbf{x}_t$ sequentially, where $\mathbf{x} = \{\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2\}$

- CD 近似算法更新公式

Update rule decomposes:

$$\theta_{t+1} = \theta_t + \alpha_t \underbrace{\left(\mathrm{E}_{P_{data}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]\right)}_{\text{True gradient}} + \alpha_t \underbrace{\left(\mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top] - \frac{1}{M}\sum_{m=1}^{M} \mathbf{v}_t^{(m)} \mathbf{h}_t^{(m)\top}\right)}_{\text{Noise term } \epsilon_t}$$

Almost sure convergence guarantees as learning rate $\alpha_t \to 0$



Problem: High-dimensional data: the energy landscape is highly multimodal

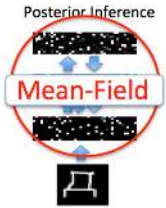
Key insight: The transition operator can be any valid transition operator – Tempered Transitions, Parallel/Simulated Tempering.

- 变分推理公式

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

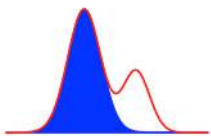
$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$



Mean-Field: Choose a fully factorized distribution:

$$Q_\mu(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} q(h_j|\mathbf{v}) \text{ with } q(h_j = 1|\mathbf{v}) = \mu_j$$

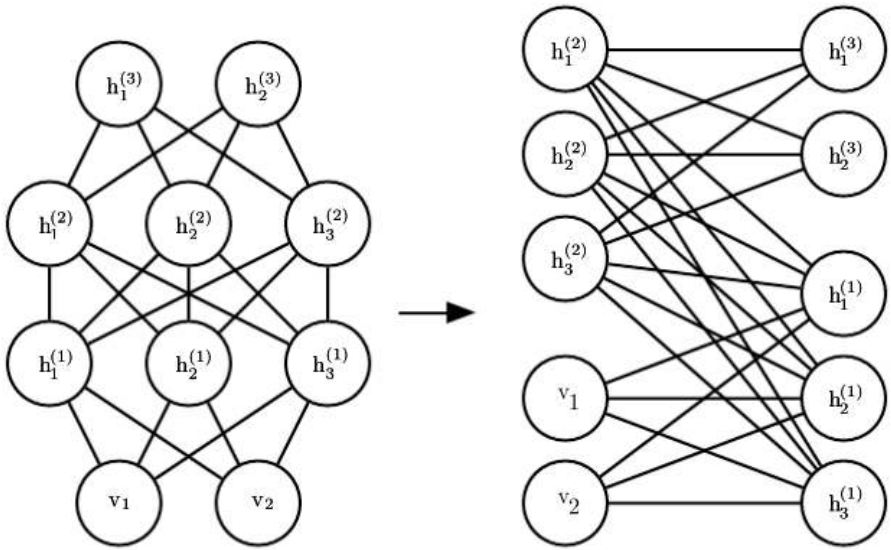
Variational Inference: Maximize the lower bound w.r.t. Variational parameters μ.



Nonlinear fixed-point equations:

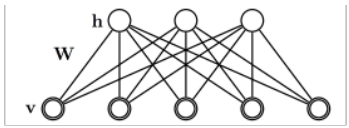
$$\mu_j^{(1)} = \sigma\Big(\sum_i W_{ij}^1 v_i + \sum_k W_{jk}^2 \mu_k^{(2)}\Big)$$
$$\mu_k^{(2)} = \sigma\Big(\sum_j W_{jk}^2 \mu_j^{(1)} + \sum_m W_{km}^3 \mu_m^{(3)}\Big)$$
$$\mu_m^{(3)} = \sigma\Big(\sum_k W_{km}^3 \mu_k^{(2)}\Big)$$

- DBM 的 RBM 视角

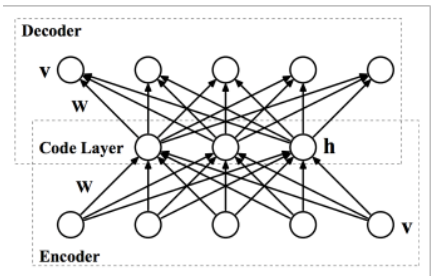

$h_1^{(3)}$
$h_2^{(3)}$
$h_1^{(2)}$
$h_2^{(2)}$
$h_3^{(2)}$
$h_1^{(1)}$
$h_2^{(1)}$
$h_3^{(1)}$
v_1
v_2
$h_1^{(2)}$
$h_2^{(2)}$
$h_3^{(2)}$
v_1
v_2
$h_1^{(3)}$
$h_2^{(3)}$
$h_1^{(1)}$
$h_2^{(1)}$
$h_3^{(1)}$

3 自动编码器 AE

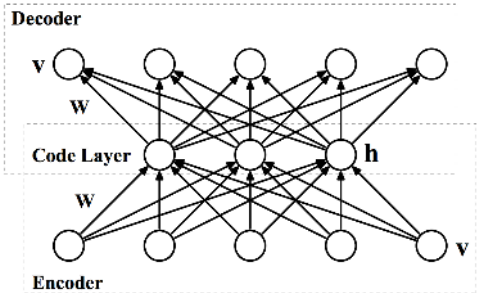
- 从 RBM 到 AutoEncoder：一次重建



有向展开[CD-1]



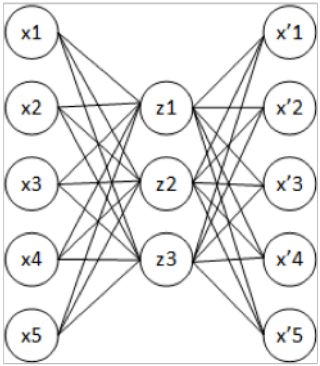
- 自动编码器 AutoEncoder



$$\text{Encoder: } h_j = \frac{1}{1+\exp(-\sum_i v_i W_{ij})},\ j = 1, ..., K.$$

$$\text{Decoder: } \hat{v}_i = \frac{1}{1+\exp(-\sum_j h_j W_{ij})},\ i = 1, ..., D.$$

- 重建角度下的 AE



$$z = f(Wx) \qquad y = g(Vz)$$

理想重建：

$$J = \sum_{n=1}^{N} |x(n) - \hat{x}(n)|^2$$

$$\hat{x}(n) = Q^{-1}Qx(n)$$

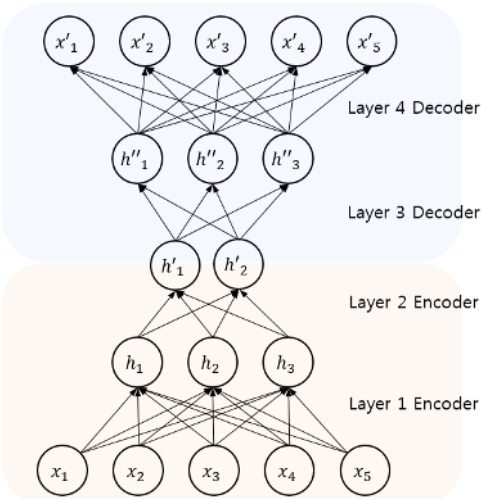
$$J = \sum_{n=1}^{N} |x(n) - Q^{-1}Qx(n)|^2$$

现实重建：（PCA 重建）

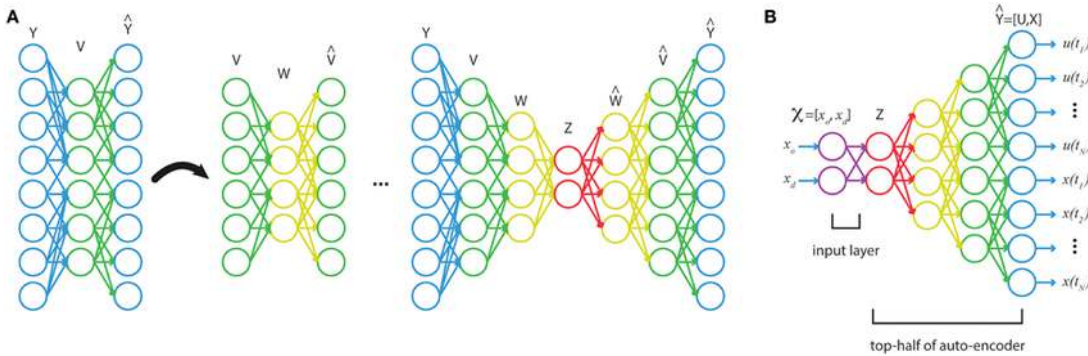
$$y = g(Vf(Wx)) = VWx$$

$$J = \sum_{n=1}^{N} |x(n) - VWx(n)|^2$$

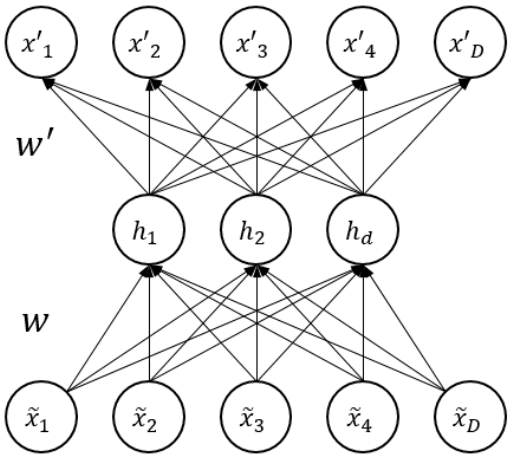
- 堆栈 (深度)AE: DBM 的有向展开



- Stacked AE 的对称逐层训练：也可以看成两层 DBN

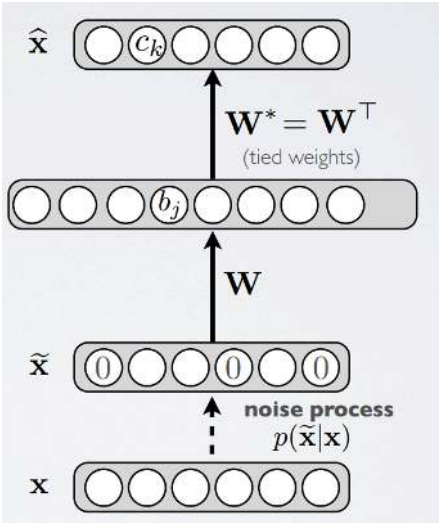


- 抗噪声 AE (Denoising AE)：如何满足带噪声情况？

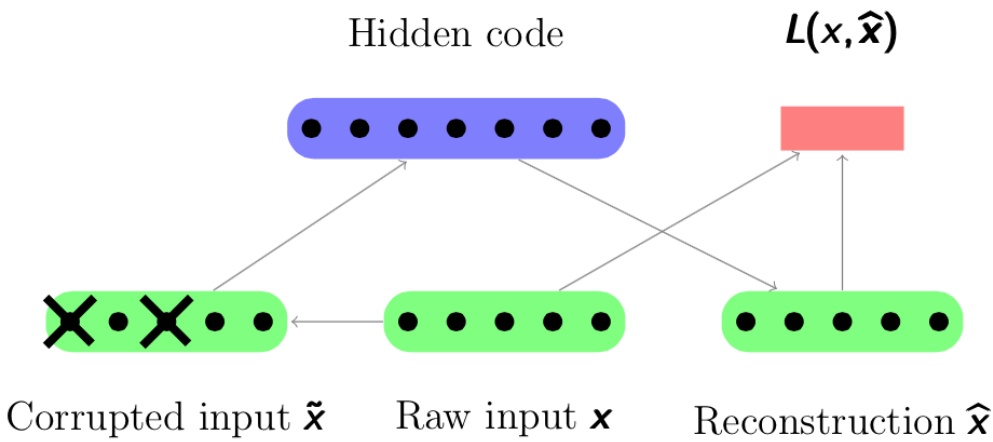


$\tilde{x}$ = x + noise

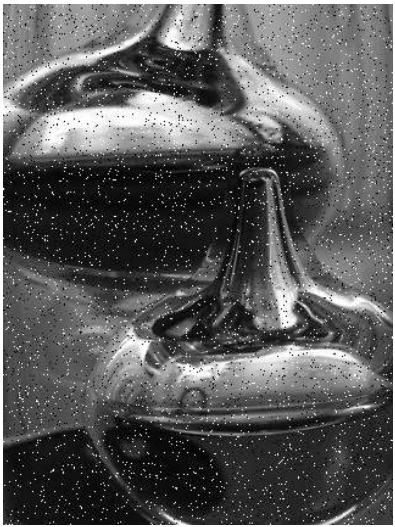
- DAE：X 数据主动加噪声



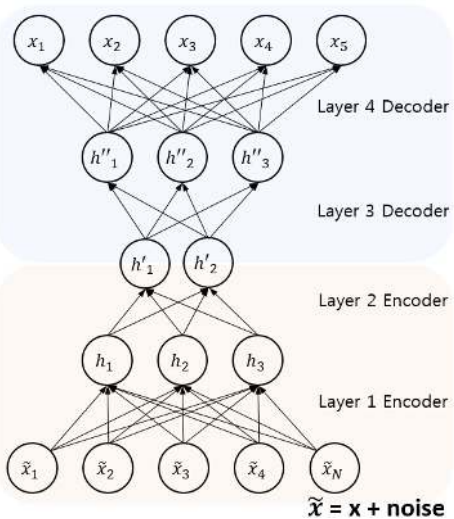
- DAE：X 数据破坏 Corrupted Data



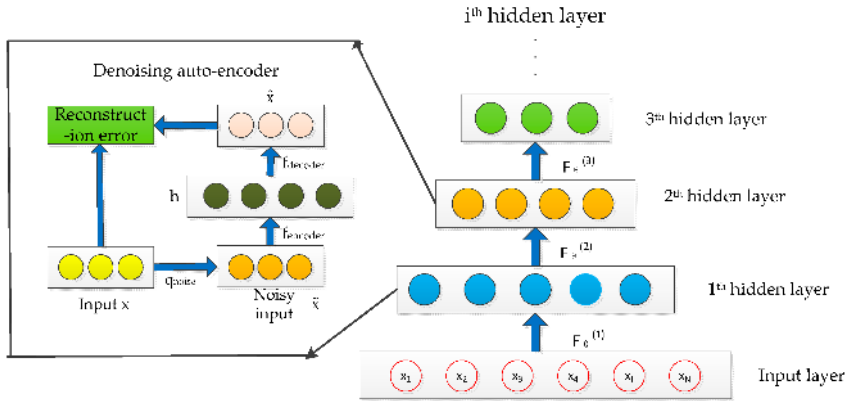
- 椒盐噪声 Salt-and-pepper noise



- 堆栈抗噪 AE (Stacked-Denoising AE)

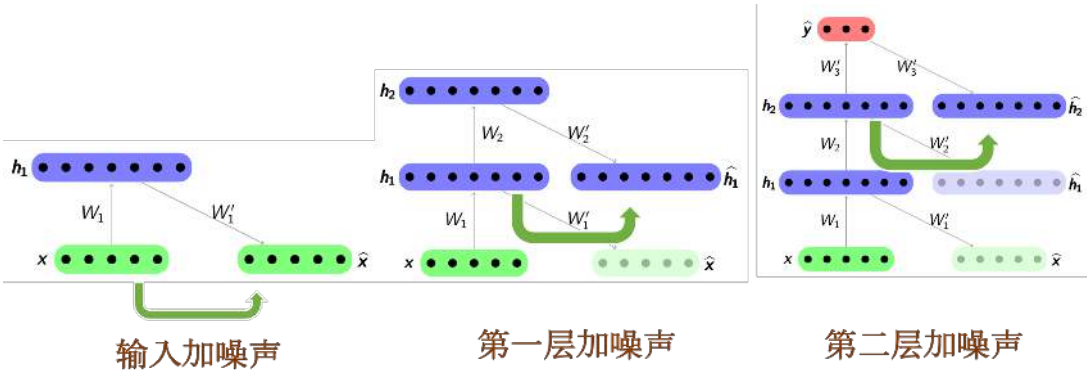


- 堆栈抗噪 AE (Stacked-Denoising AE)



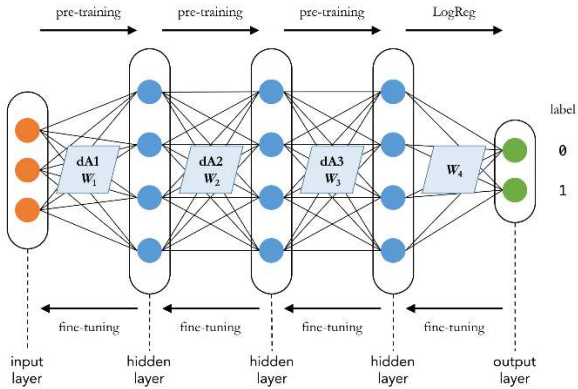
1）逐层训练的时候每层加噪声

- 堆栈抗噪 AE (Stacked-Denoising AE)：部分类似加 Dropout



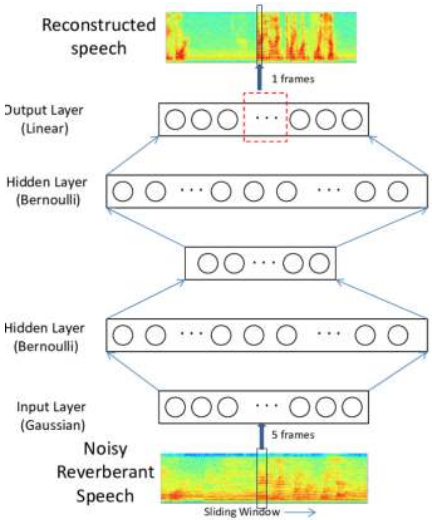
1）逐层训练的时候每层加噪声

- 堆栈抗噪 AE (Stacked-Denoising AE)：Fine-Tune

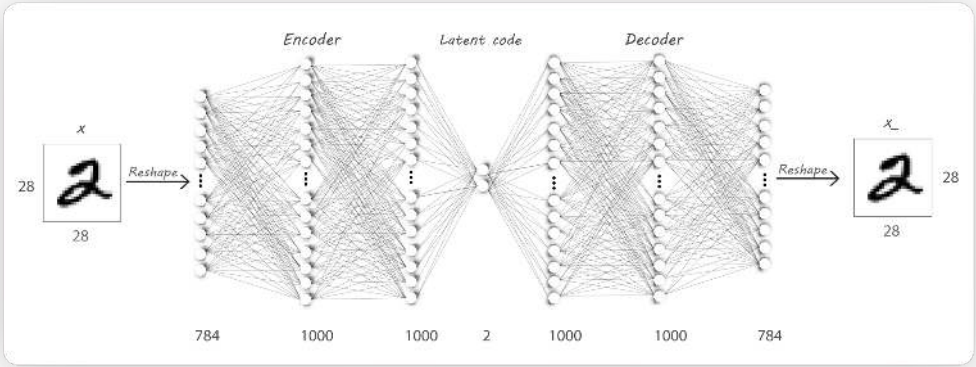


2）Fine-Tuning 的时候不加噪声

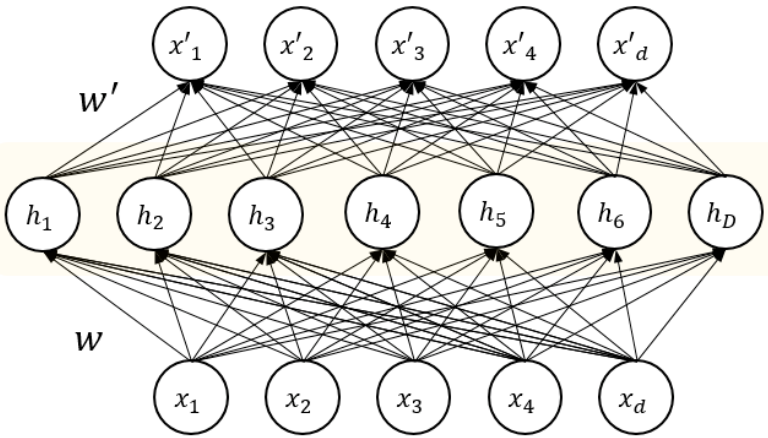
- 语音识别的噪声过滤


Reconstructed speech
1 frames
Output Layer (Linear)
Hidden Layer (Bernoulli)
Hidden Layer (Bernoulli)
Input Layer (Gaussian)
5 frames
Noisy Reverberant Speech
Sliding Window

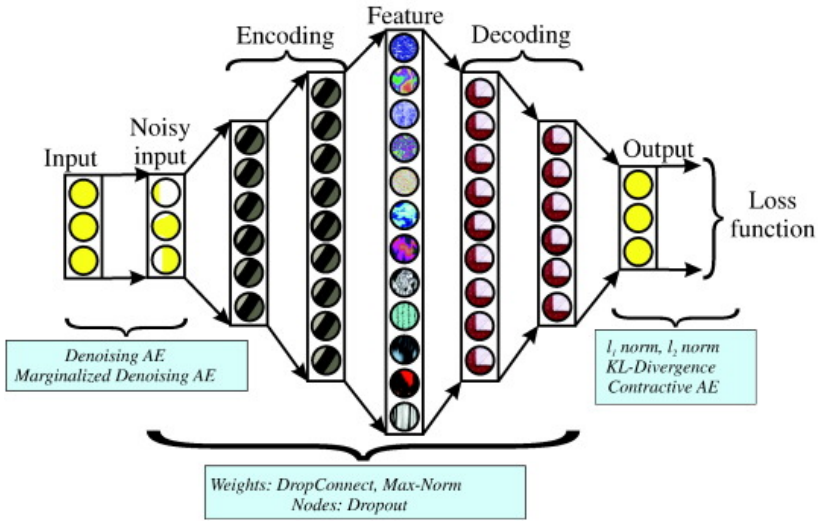
- 图像识别的噪声过滤


Encoder
Latent code
Decoder
x
28
28
Reshape
Reshape
x_
28
28
784
1000
1000
2
1000
1000
784

- 稀疏 AE（Sparse AE）



- 堆栈稀疏 AE（Stacked Sparse AE）



- Sparse AE 的目标：MSE+Reg(Weight)+Cost(Sparsity)

平方距离：

$$J(W,b;x,y)=\frac{1}{2}\left\|h_{W,b}(x)-y\right\|^2$$

权重正则化项：

$$\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

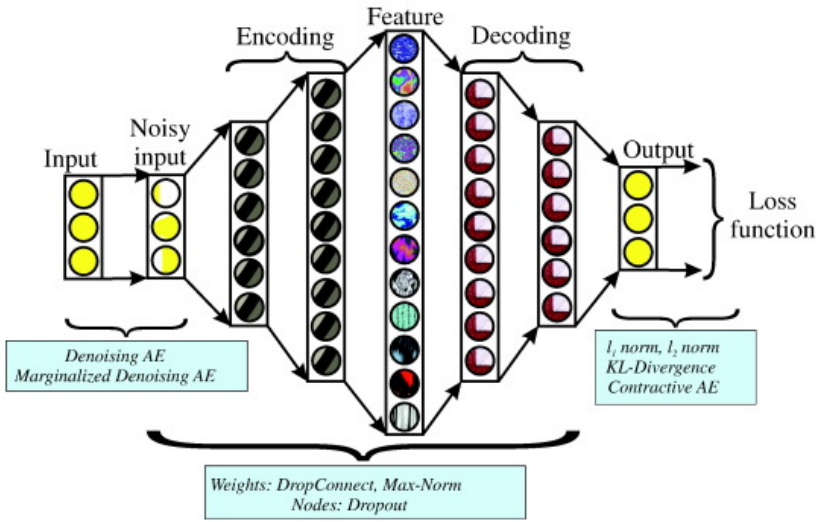
稀疏代价：

$$\sum_{j=1}^{s_2}\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}$$

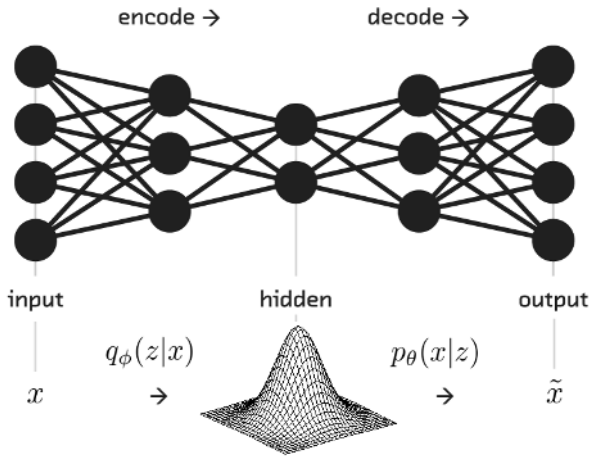
其中概率估算：

$$\hat{\rho}_j=\frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right]$$

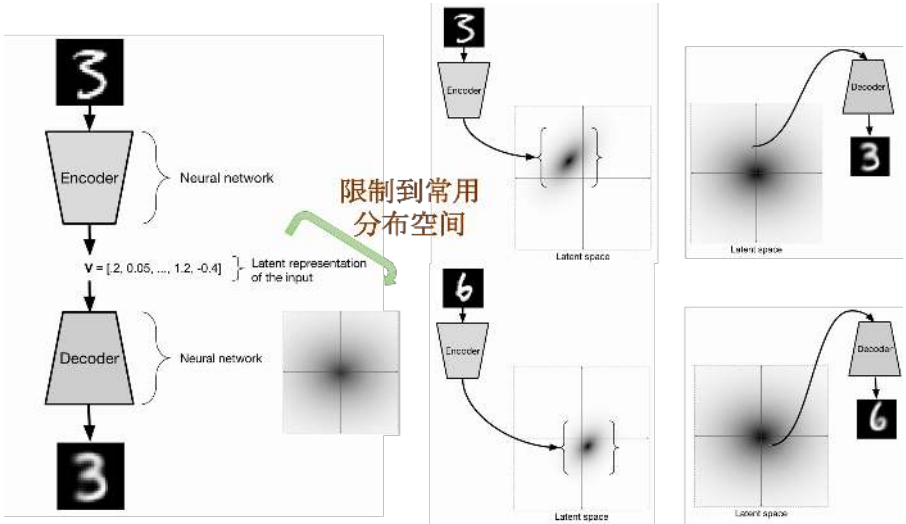
- Sparse AE 可以构建特征词典



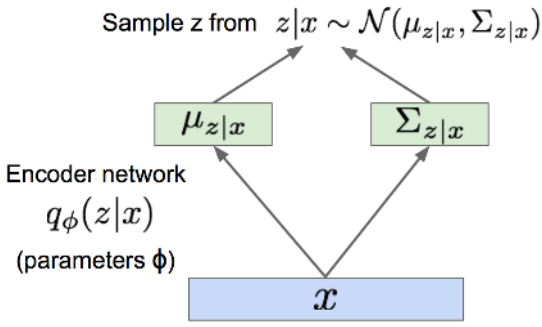
- 变分 AE（Variational AE）

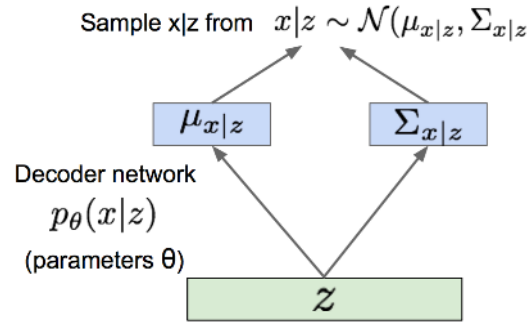


- VAE：想法来源

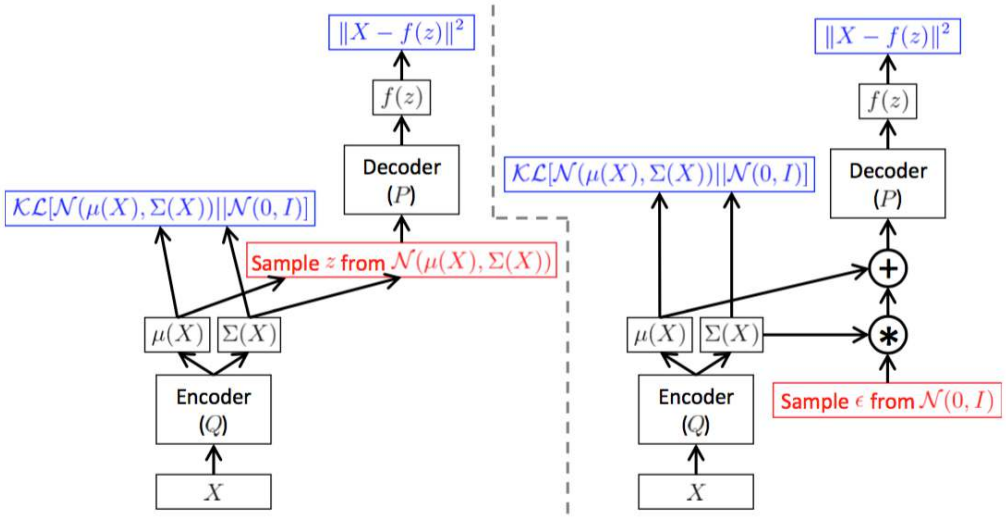

Encoder
Neural network
V = [.2, 0.05, ..., 1.2, -0.4]
Latent representation of the input
Decoder
Neural network
限制到常用分布空间
Latent space
Latent space
Latent space
Latent space

- VAE 要求

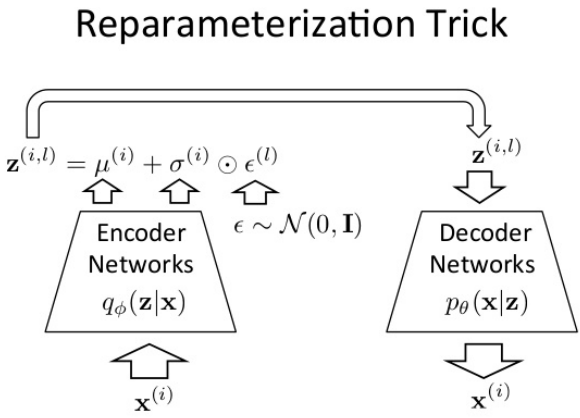

Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x}$
$\Sigma_{z|x}$
Encoder network
$q_\phi(z|x)$
(parameters ϕ)
x


Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
(parameters θ)
z

- VAE 构建需要的分布：基于标准正态分布



- 基于标准正态分布构建分布技巧：Reparameterization Trick



- ELBO 下限推理

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}\\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}
$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

- ELBO 重建的含义

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

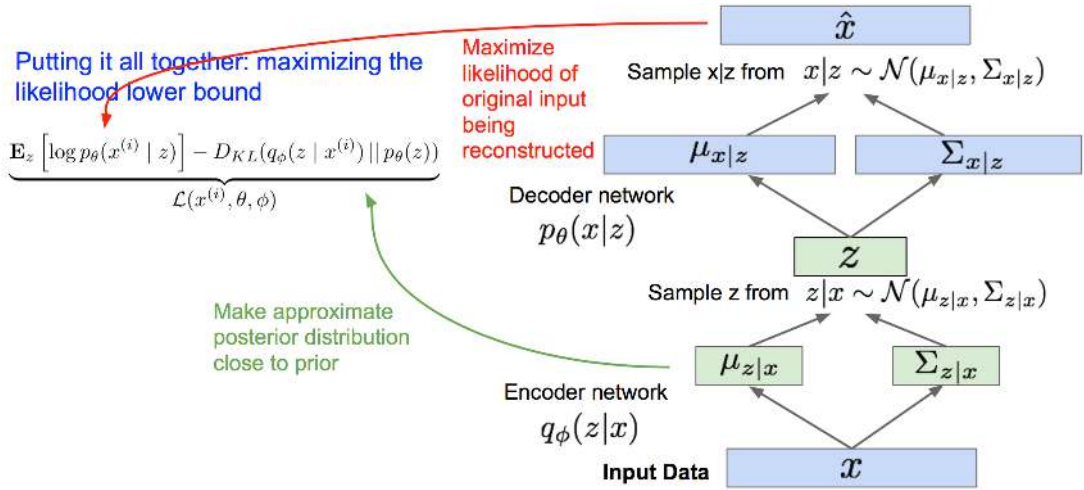
$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

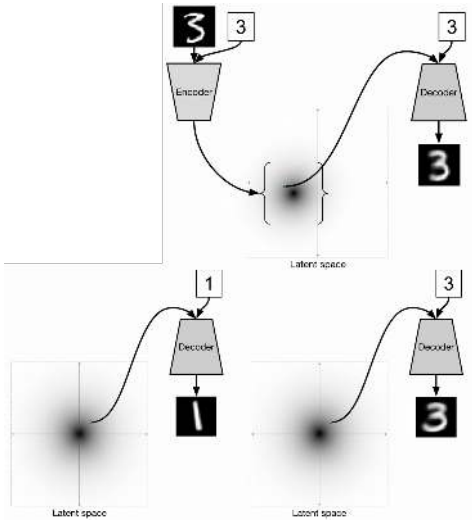
$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

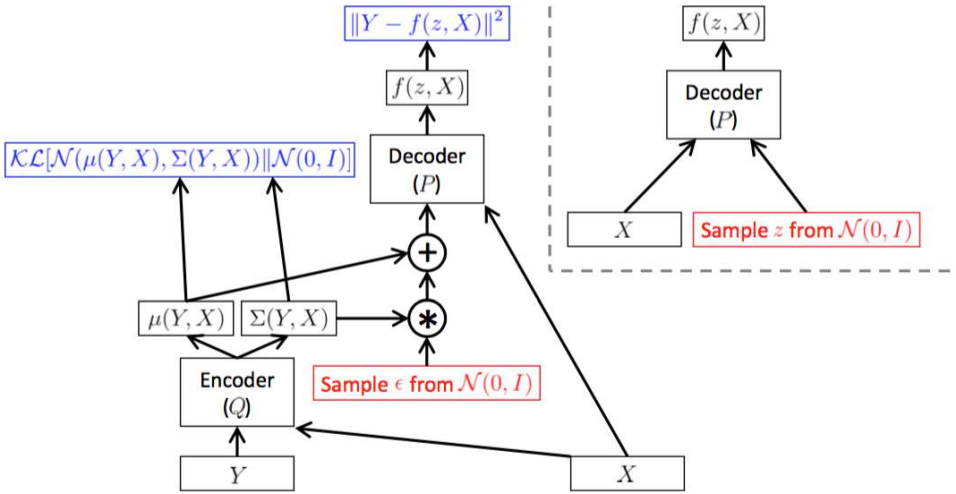
- ELBO：KL 分布距离作为正则化限制



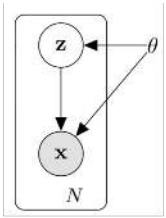
- 条件 VAE （Conditional VAE）



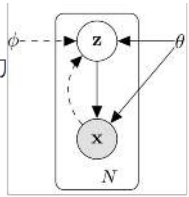
- 条件 VAE （Conditional VAE）



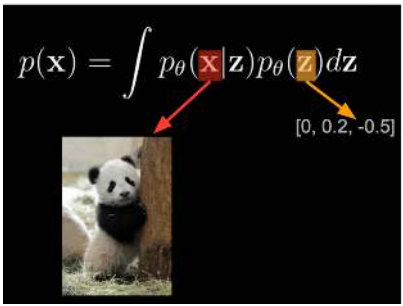
- VAE 的变分思想回顾



不是直接利用Z分布，
而是求解一个接近Z的
带参数的任意分布



变分思想

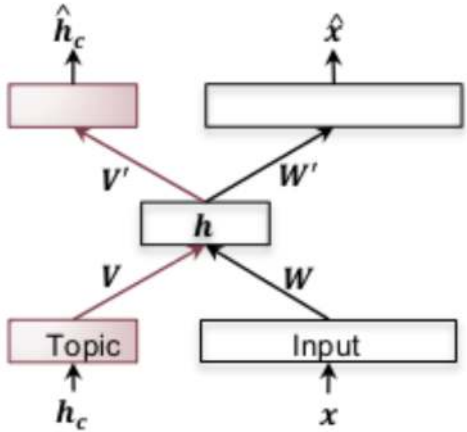


$$\log p_\theta(\mathbf{x}_i) \geq$$

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})]$$

$$KL[q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})]$$

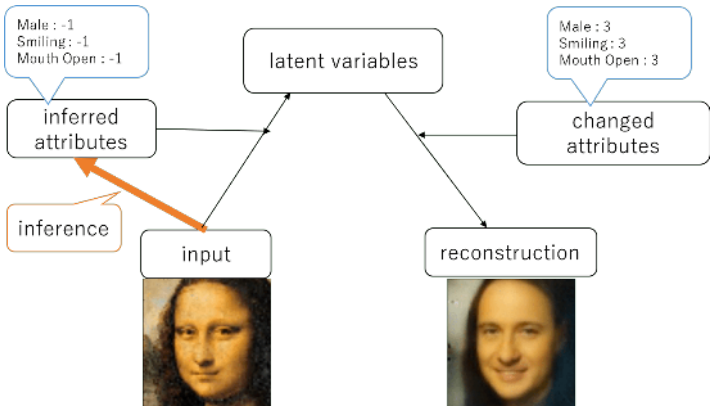
- 上下文 AE（Context-Sensitive AE）



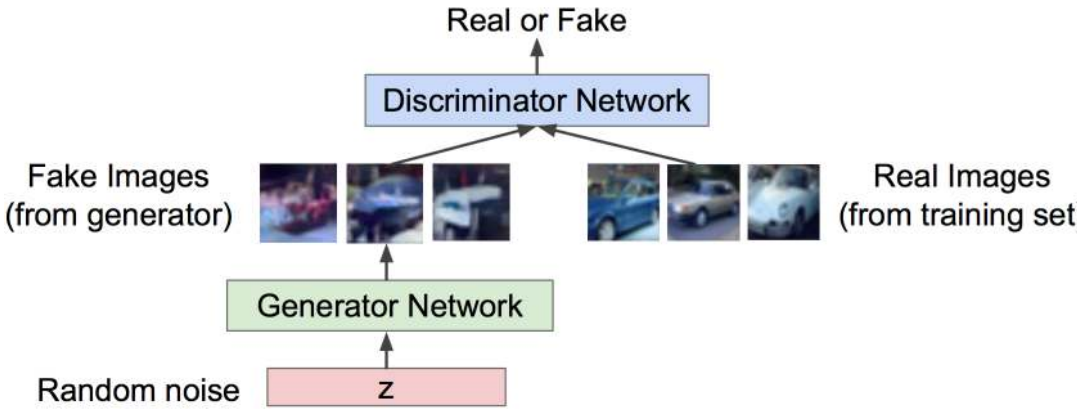
$$l(\mathbf{x}, \mathbf{h}_c) = ||\mathbf{x} - \hat{\mathbf{x}}||^2 + \lambda||\mathbf{h}_c - \hat{\mathbf{h}}_c||^2$$

4 生成对抗网络 GAN

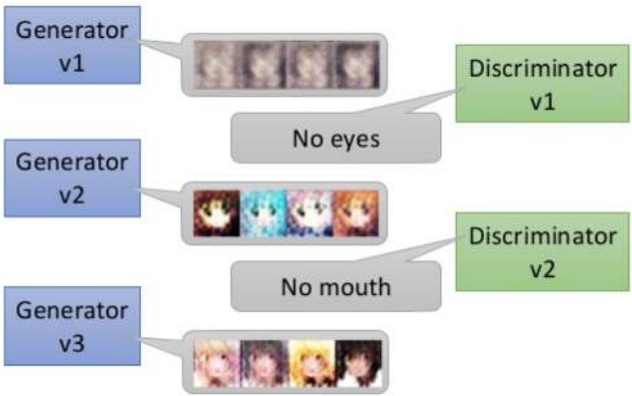
- VAE 更好的评价函数



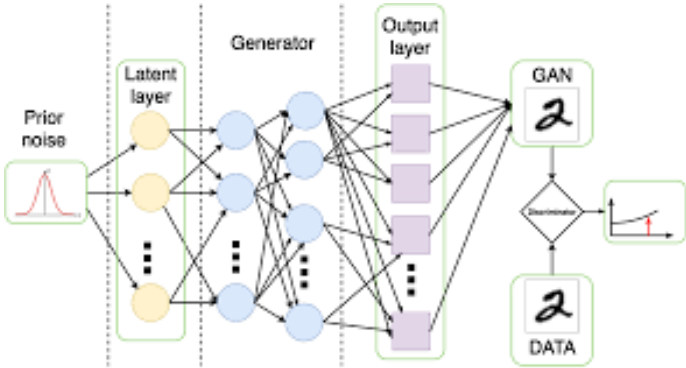
- 生成对抗网络 Generative Adversarial Network：神经网络评价



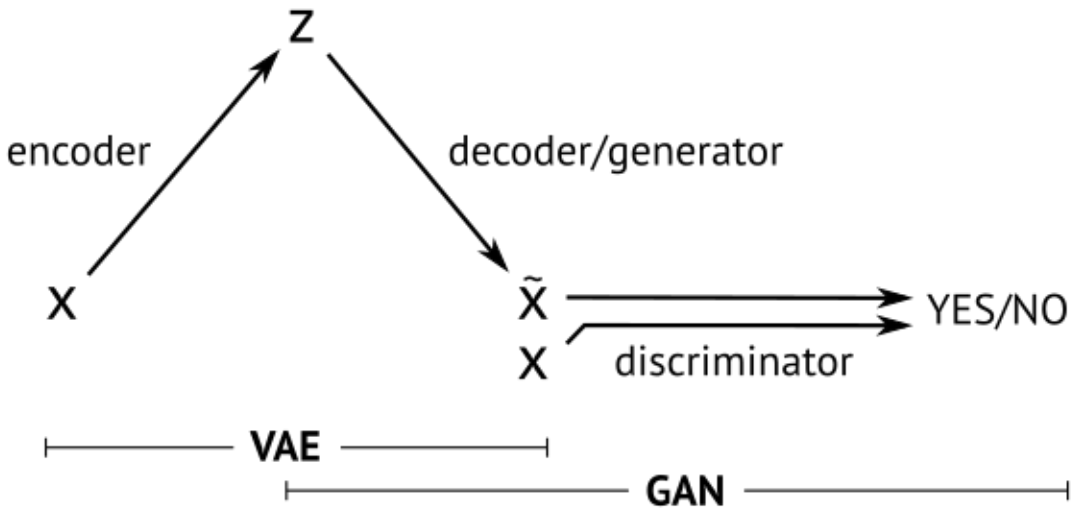
- 生成 + 判别的不停尝试


Generator
v1
Discriminator
v1
No eyes
Generator
v2
Discriminator
v2
No mouth
Generator
v3

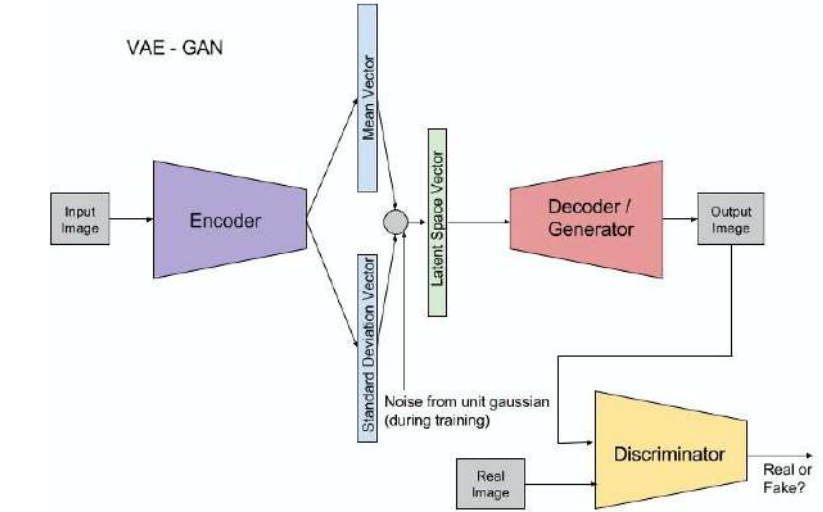
- 生成 + 判别的网络



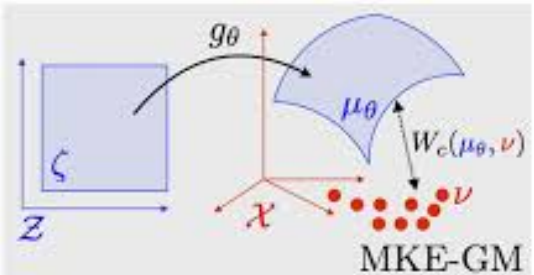
- GAN 和 VAE 的对比

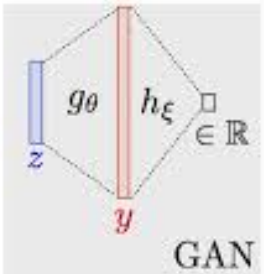

Z
encoder
decoder/generator
X
X̃
X
YES/NO
discriminator
VAE
GAN

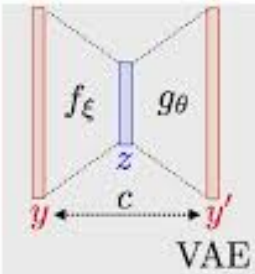
- VAE 和 GAN


VAE - GAN
Input Image
Encoder
Mean Vector
Standard Deviation Vector
Latent Space Vector
Noise from unit gaussian (during training)
Decoder / Generator
Output Image
Discriminator
Real Image
Real or Fake?

- VAE 和 GAN


g_θ
μ_θ
$W_c(\mu_\theta, \nu)$
ζ
$\mathcal{Z}$
$\mathcal{X}$
ν
MKE-GM


g_θ
h_ξ
$\in \mathbb{R}$
z
y
GAN


f_ξ
g_θ
z
c
y
y'
VAE

- GAN 的优化目标

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

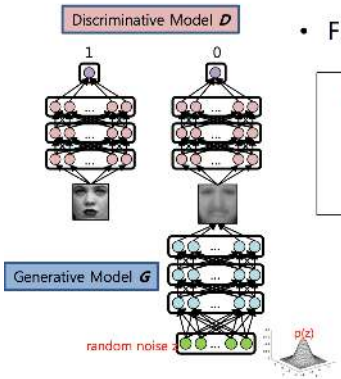
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- GAN 的优化目标

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z\sim p_{z(z)}}[\log(1-D(G(z)))]$$



- Fixed D, minimize V(G):

$$\min_G V_D(G) = \min\left[\mathbb{E}_{z\sim p_{z(z)}}[\log(1-D(G(z)))]\right]$$

Try to make $D(G(z)) = 1$

Stochastic Gradient

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right).$$

- GAN 的优化目标：求导

目标

$$J(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1-D(G(\boldsymbol{z})))]$$
$$= \int_{(\boldsymbol{x})} p_{data}(\boldsymbol{x})\log(D(\boldsymbol{x}))dx + \int_{\boldsymbol{z}} p_z(\boldsymbol{z})\log(1-D(G(\boldsymbol{z})))dz$$
$$= \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x})\log(D(\boldsymbol{x})) + p_G(\boldsymbol{x})\log(1-D(\boldsymbol{x}))dx$$

最优求导

$$D^* = \underset{D}{\operatorname{argmax}}\, J(G,D)$$
$$= \underset{D}{\operatorname{argmax}} \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x})\log(D(\boldsymbol{x})) + p_G(\boldsymbol{x})\log(1-D(\boldsymbol{x}))dx$$

$$f(D) = a\log(D) + b\log(1-D).$$
$$\frac{\partial f}{\partial D} \triangleq 0$$
$$D^* = \frac{a}{a+b}$$

优化公式

最优

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_G(\boldsymbol{x})}$$

- GAN 的优化目标: 最优情况

For G fixed, the optimal discriminator D is

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Plug in the minimax objective

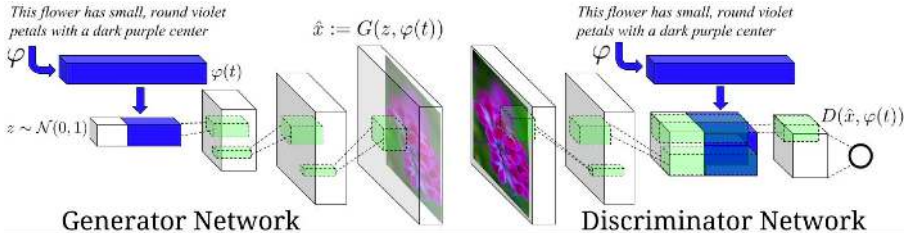
$$\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}$$

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} \mid p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} \mid p_g}{2}\right)$$

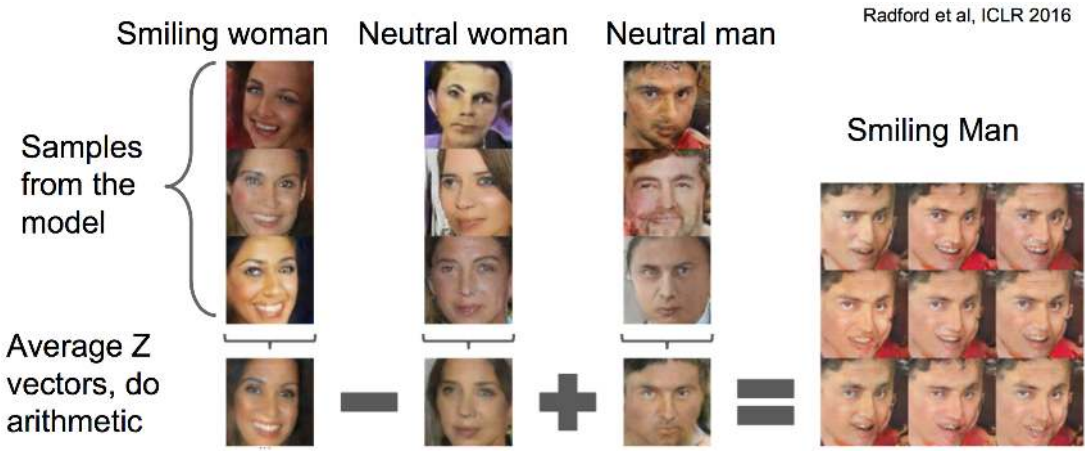
$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

$C^* = -\log 4$ is the global minimum and the only solution is $p_g = p_{data}$

- DCGAN

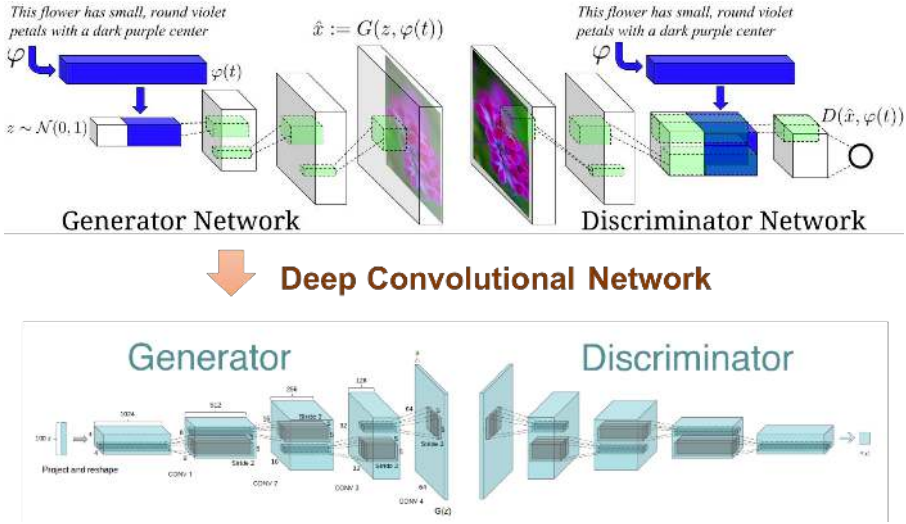

This flower has small, round violet petals with a dark purple center
$\hat{x} := G(z, \varphi(t))$
This flower has small, round violet petals with a dark purple center
φ
$\varphi(t)$
$z \sim \mathcal{N}(0,1)$
$D(\hat{x}, \varphi(t))$
Generator Network
Discriminator Network
Deep Convolutional Network
Generator
Discriminator
Project and reshape
CONV 1
CONV 2
CONV 3
CONV 4
G(z)

- DCGAN 嵌入空间：发型

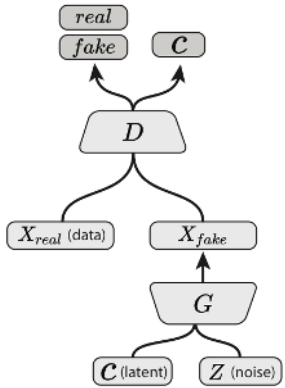


- DCGAN 嵌入空间：眼镜

- DCGAN


This flower has small, round violet petals with a dark purple center
$\hat{x} := G(z, \varphi(t))$
This flower has small, round violet petals with a dark purple center
φ
$\varphi(t)$
$z \sim \mathcal{N}(0,1)$
$D(\hat{x}, \varphi(t))$
Generator Network
Discriminator Network
Deep Convolutional Network
Generator
Discriminator
Project and reshape
CONV 1
CONV 2
CONV 3
CONV 4
G(z)

- InfoGAN：Mutual information 衡量主要特征和内部变化

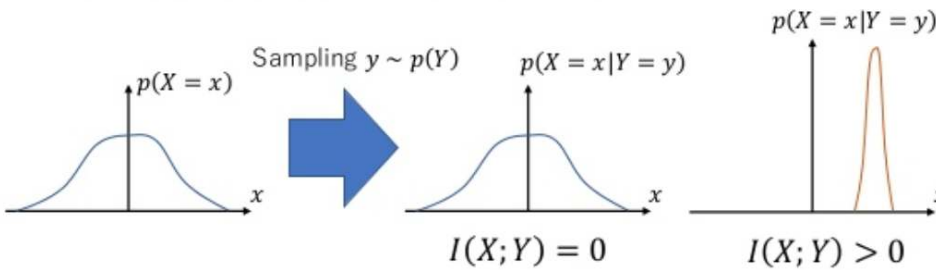


InfoGAN
(Chen, et al., 2016)

- Mutual Information

$I(X;Y) = H(X) - H(X|Y)$, where

- $H(X) = E_{x\sim p(X)}[-\ln p(X = x)]$:
  Entropy of the prior distribution
- $H(X|Y) = E_{y\sim p(Y),x\sim p(X|Y=y)}[-\ln p(X = x|Y = y)]$:
  Entropy of the posterior distribution



- Mutual Information to GAN

$$
\begin{aligned}
I[X,Y] &= H[Y] - \mathbb{E}_x H[Y|X=x] \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log p(y|x) \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log \frac{p(y|x)q(y|x)}{q(y|x)} \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x) + \mathbb{E}_x \mathbb{E}_{y|x} \log \frac{p(y|x)}{q(y|x)} \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x) + \mathbb{E}_x KL[p(y|x)\|q(y|x)] \\
&\geq H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x)
\end{aligned}
$$

$$
I[X,Y] = \max_q \left\{ H[Y] + \mathbb{E}_{x,y} \log q(y|x) \right\}
$$

$$
I[X,Y] \geq H[Y] + \max_\psi \mathbb{E}_{x,y} \log q(y|x;\psi)
$$

$$
\begin{aligned}
\ell_{GAN}(\theta) &= I[x,y] \\
&\geq h(0.5) + \max_\psi \mathbb{E}_{x,y} \log q(y|x;\psi) \\
&= h(0.5) + \max_\psi \left\{ \mathbb{E}_{x_{real}} \log q(1|x;\psi) + \mathbb{E}_{x_{fake}} \log q(0|x;\psi) \right\}
\end{aligned}
$$

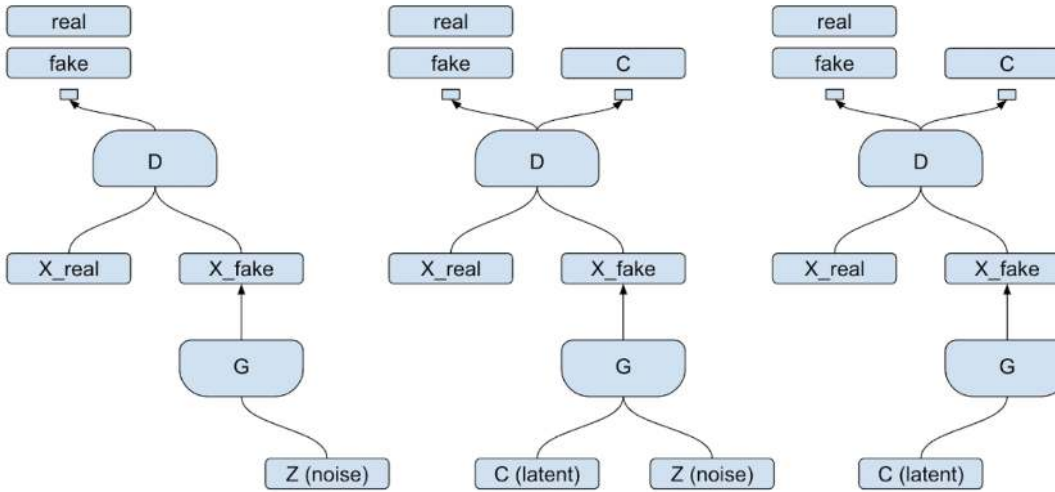
$$
\ell_{GAN}(\theta) + h(0.5) \geq \max_\psi \left\{ \mathbb{E}_{x \sim P_{data}} \log D(x,\psi) + \mathbb{E}_{c,z} \log(1 - D(G(c,z,\theta),\psi)) \right\}
$$

- InfoGAN: 内部变化

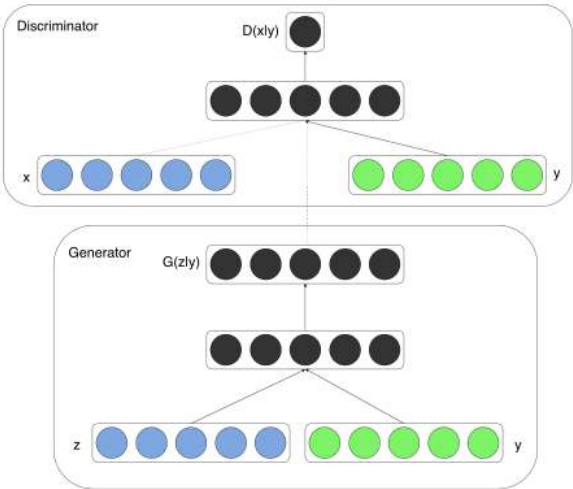
(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

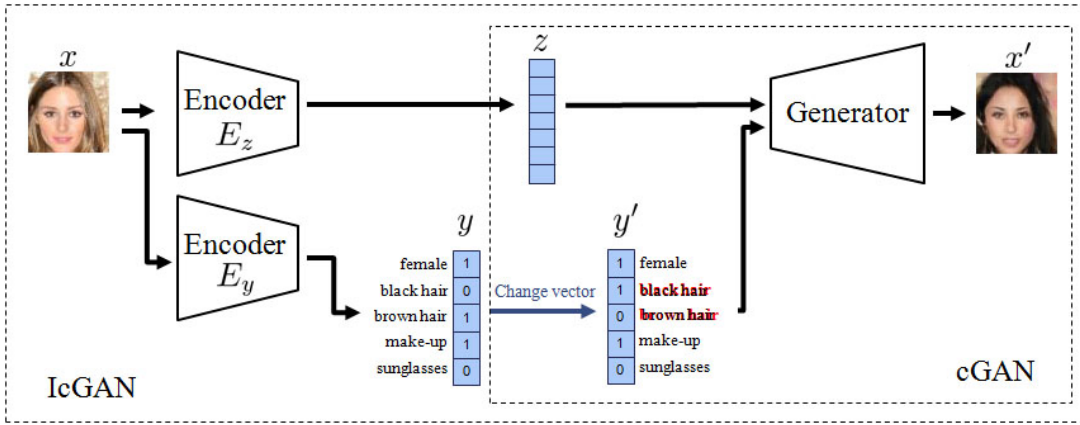
- InfoGAN：Vanilla GAN -> InfoGAN -> InfoGAN w



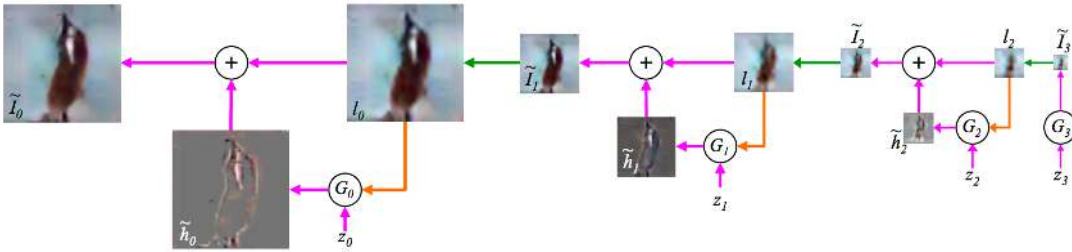
- Conditional GAN



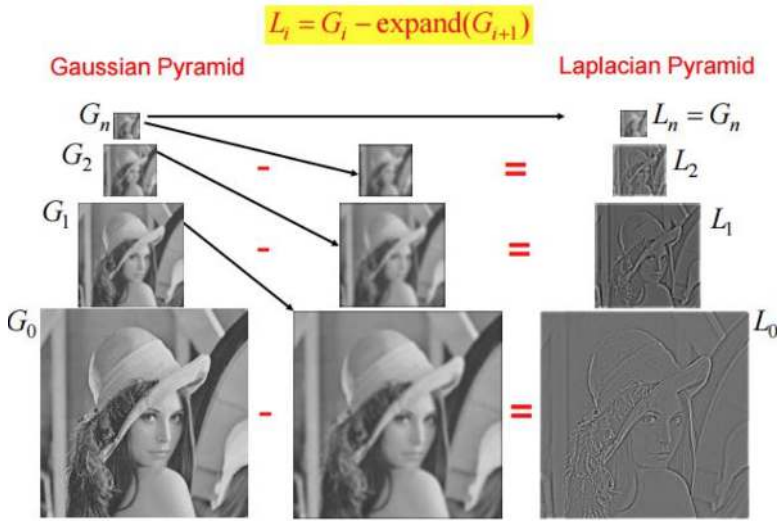
- Invertible Conditional GAN



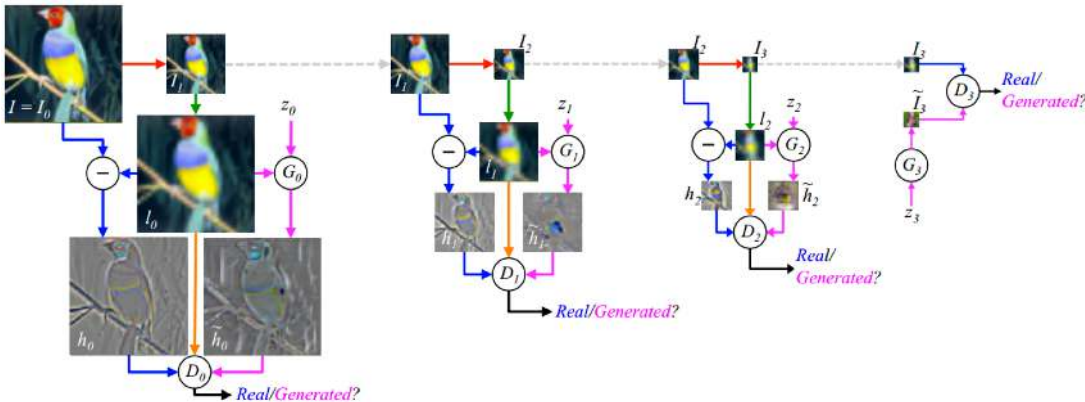
- LapGAN


$\tilde{I}_0$
l_0
$\tilde{I}_1$
l_1
$\tilde{I}_2$
l_2
$\tilde{I}_3$
$\tilde{h}_0$
$\tilde{h}_1$
$\tilde{h}_2$
G_0
G_1
G_2
G_3
z_0
z_1
z_2
z_3

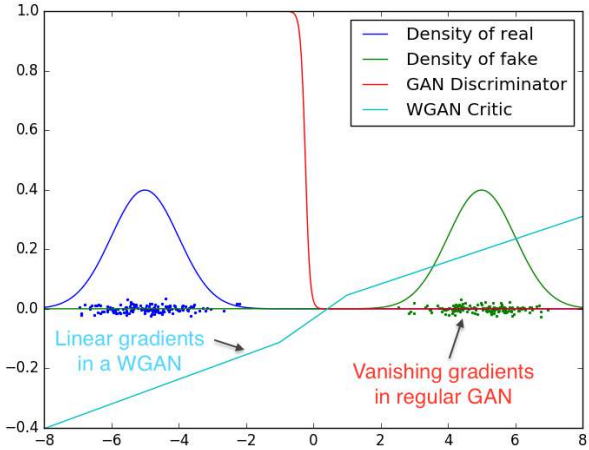
- Laplacian Pyramid



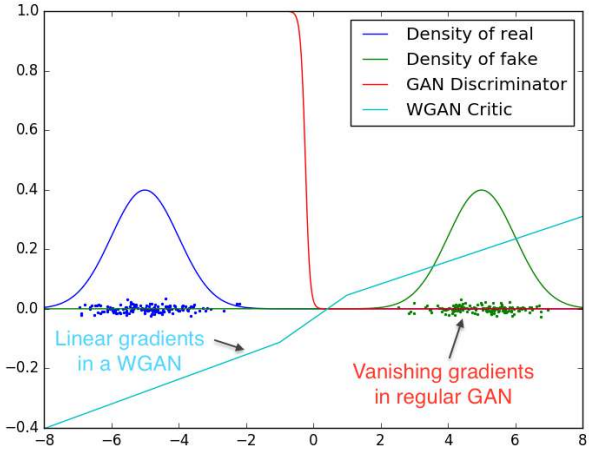
- LapGAN Train



- Wasserstein GAN



- Wasserstein GAN



- Wasserstein Distance

Jensen-Shannon divergence

$$JS(P_r, P_g) = \frac{1}{2} KL(P_r \| P_m) + \frac{1}{2} KL(P_g \| P_m)$$

Wasserstein distance

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$$

1-Lipschitz functions :

Kantorovich-Rubinstein duality

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_\theta}[f(x)]$$

感谢 Stanford, CMU, MIT 等网上公开课程和资料！

AI2ML