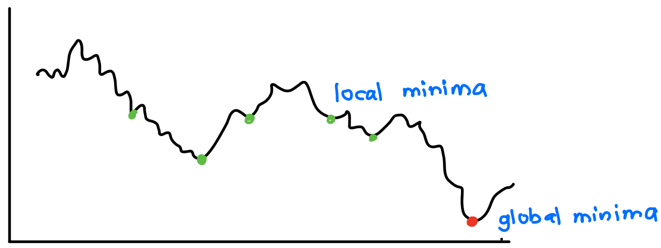
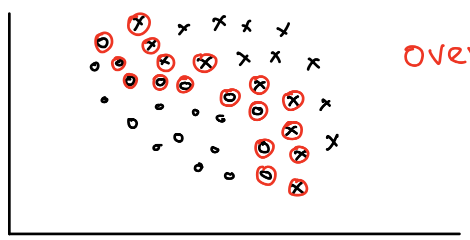
Problems so far,

→ Discriminative methods like SVM and backprop need a long time to train.

→ Backprop would get caught in local minima.



SVM - We get large # of support vectors for hard problems.



overfitting ?

→ Diminishing gradients prevent deep topologies.

More problems :

→ Need more training data : P(lablel | data)

→ Hard to learn a non over fitted model just by relying on labels. Why not look at inter data patterns

Some rough ideas :

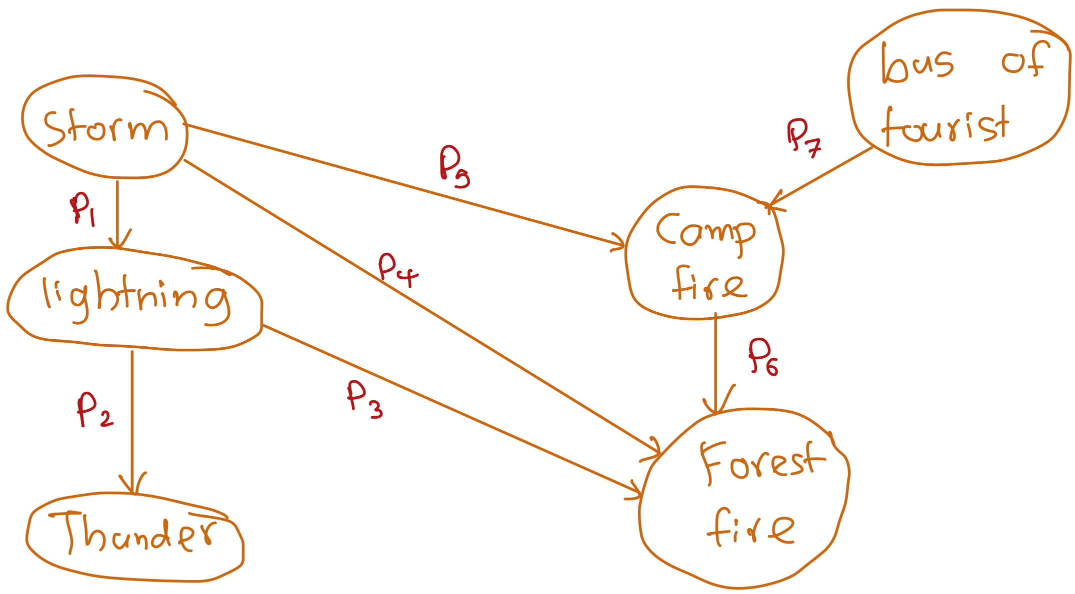
* Discriminate VS Generate

* Do not calculate P(label | data) but P(data)

  Not $\underbrace{\text{Van Gough vs Picaso}}_{\text{discriminative}}$ but just $\underbrace{\text{van Gogh!}}_{\text{generative}}$

* Intelligence is understanding the data.
* If you undersond it, you can regenerate it.
* To use just the data, we need to use its energy.

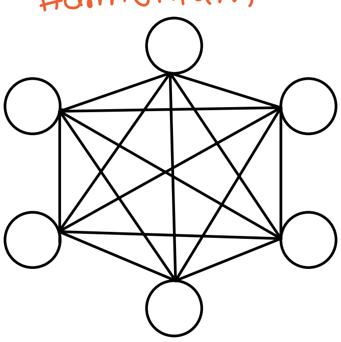
When is there a forest fire ?



(Bayesian) Belief Network

Potential Model

"Harmonium" $\longrightarrow$ "RBM"



Boltzmann Machines

eg:- Recurrent networks

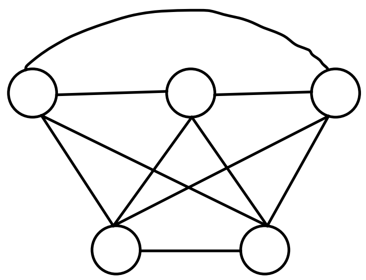
Hopfield networks with hidden neurons

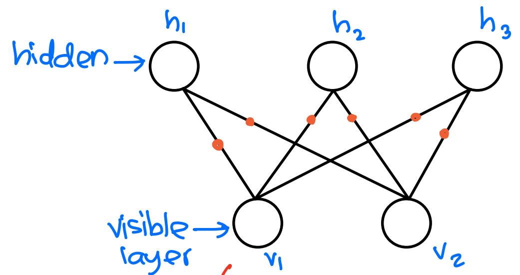
stochastic

Special form of Markov random fields

They use energy functions

Restriction = Simplification





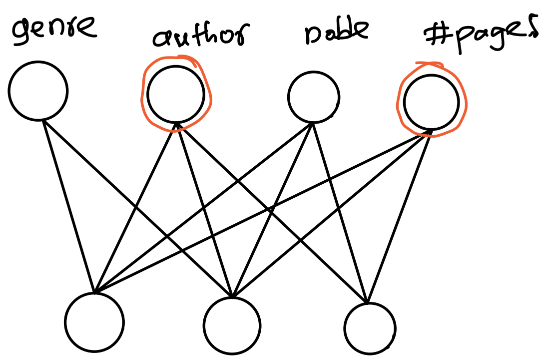
Boltzman machine with interconnection within each layer

Restricted BM

Often use binary neurons

$v, h \rightarrow$ binary $\quad E(v,h) = \underbrace{-b^T v}_{\text{bias assign to hidden neuron}} - c^T h - h^T \underbrace{W}_{\text{weights}} v$

energy that takes to work on this belief network



book1	book2	book3
1	0	0
0	1	1
1	1	1

1 - Likes

0 - Don't Like

* There is no dependency between book1 and book2

Ignoring the bias, $E(v,h) = -\sum_{i,j} \underbrace{v_i h_j}_{\text{binary state}} \underbrace{W_{ij}}_{\text{weight}}$

Restriction = no connection between hidden/visible units.

* Conditional Independence

Hence we can write, $p(h|v) = \prod_i p(h_i|v)$

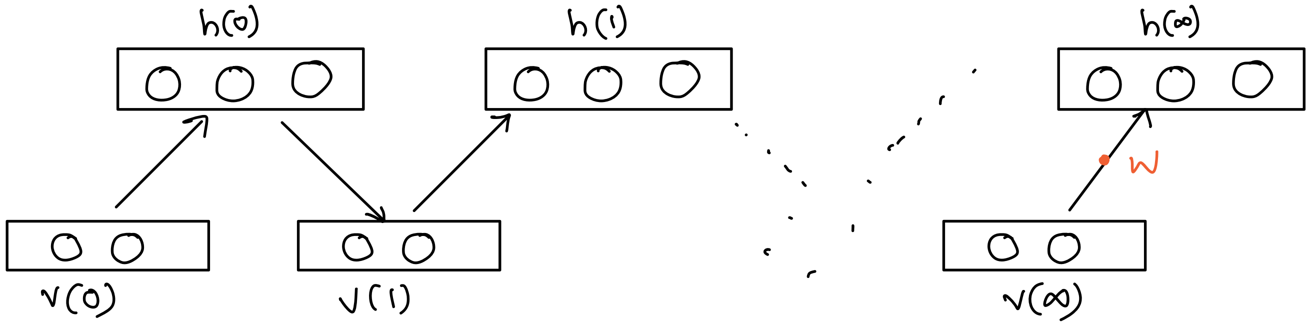
$p(v|h) = \prod_i p(v_i|h)$

Learning means to assign a probability to every possible pair of visible/hidden vectors via the energy function.

$$P(\underline{v},\underline{h}) = \frac{e^{-E(v,h)}}{\underbrace{\sum_{v,h} e^{-E(v,h)}}_{\text{Partition function}}}$$

Probability assigned to a visible vector :

$$P(\underline{v}) = \frac{1}{Z} \sum_h e^{-E(v,h)}$$



$$\frac{\partial \log P(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{train} - \langle v_i h_j \rangle_{model}$$

$\langle \cdots \rangle$ means expectation under distribution by training model

$$E(v,h) = -\sum v_i h_j w_{ij} \longrightarrow \frac{\partial E}{\partial w} = v_i h_j$$

$$\Delta w_{ij} = \eta\left[\langle v_i h_j\rangle^{(0)} - \langle v_i h_j\rangle^{(\infty)}\right]$$

$$h^{(n+1)} \approx g(w^T \cdot v^{(n)} + c)$$

$$v^{(n+1)} \approx g(w^T h^{(n)} + b)$$

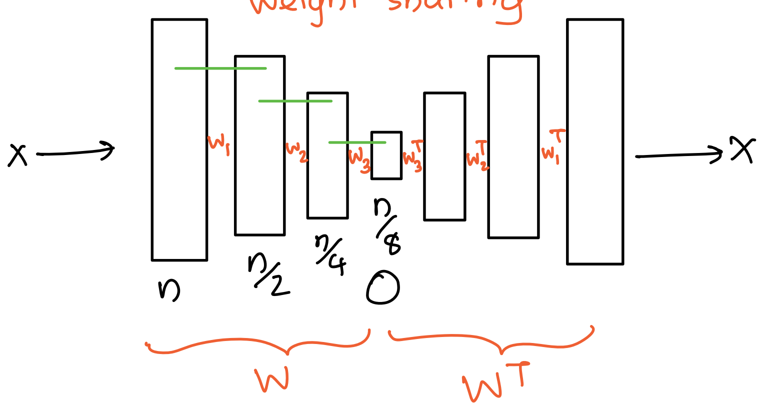
g ↓ Sigmoidal

Learning method : Contrastive Divergence

Back to the AutoEncoders

Deep AEs cannot be trained with backprop ← problem

weight sharing



① Train RBMs (CD)

② fine tune with backprop