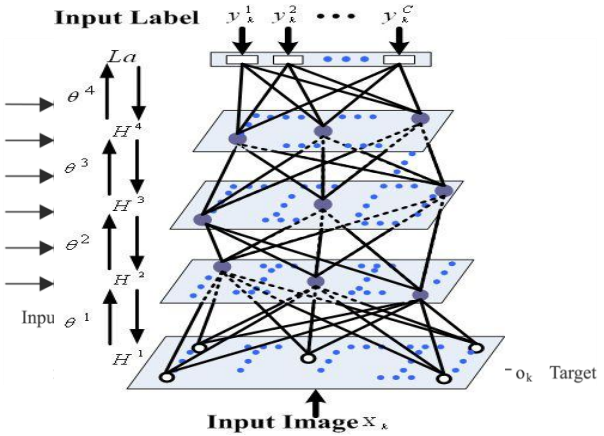
Deep Belief Network

Tiger

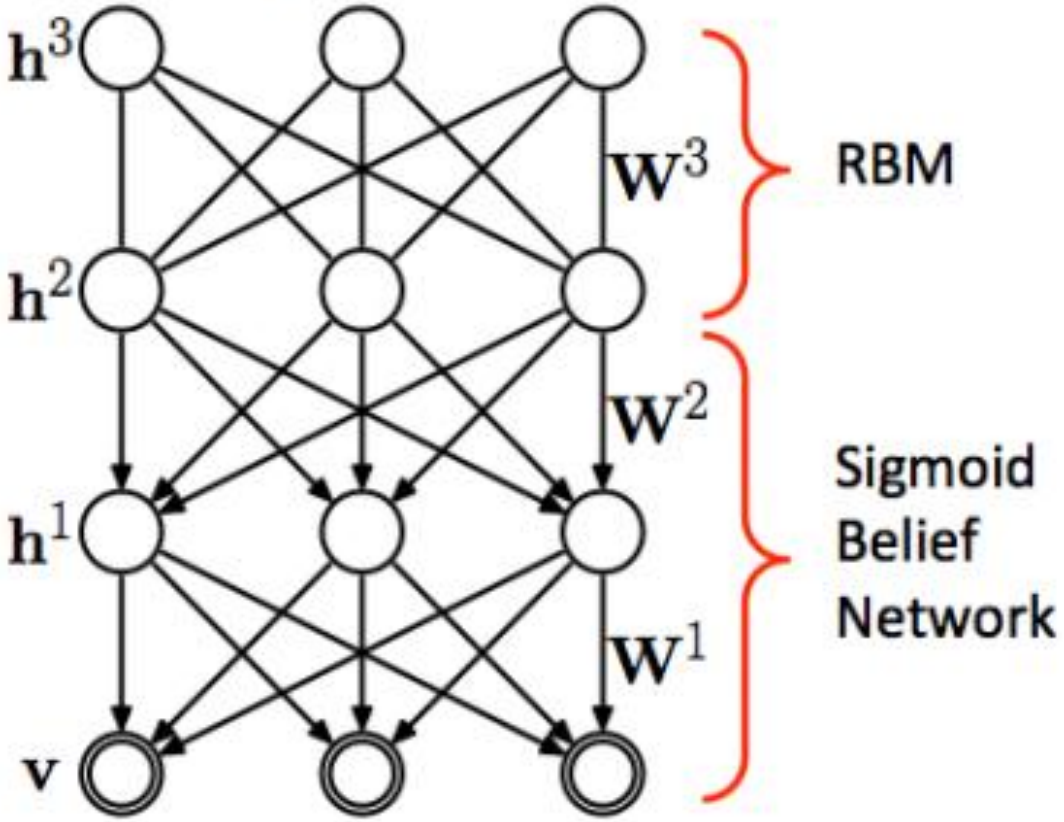
2016

Brief History

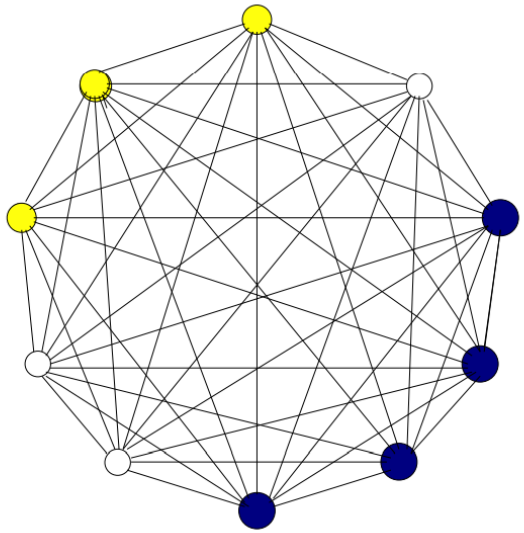


- Perceptron (Classification)
  - Single perception (McCulloch W. & Pitts W., 1943)
  - Multi-layer Perceptron (Rosenblatt F., 1958)
- Artificial Neural Network (State Machine)
  - Backpropagation (Linnainmaa, 1970; Werbos P., 1974; Rumelhard D., et al, 1986)
- Deep Learning (Reconstruction)
  - Layered learning (Ivakhnenko and Lapa, 1965)
  - (Restricted) Boltzmann Machine (Hinton, 1986)
  - Sigmoid belief network (Neal, 1991)

Deep Belief Net



Hopfield Network



- New State, local field

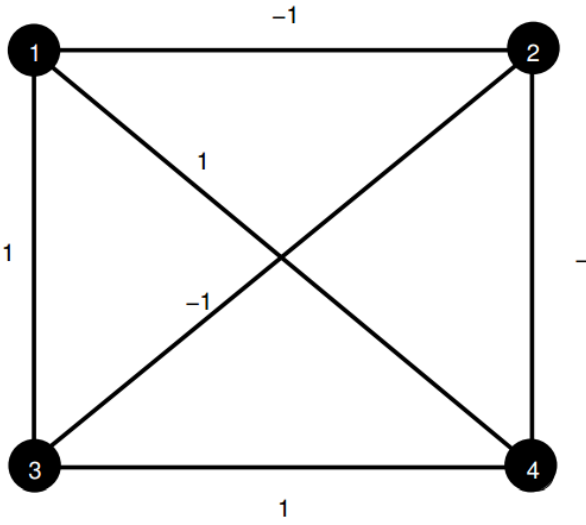
$$h_i = \sum_i w_i x_i$$

- Output formula

$$o = \begin{cases} 1 : \sum_i w_i x_i \geq 0 \\ 0 : \sum_i w_i x_i < 0 \end{cases}$$

- Maintain its output until updated
- Constrains
  - Symmetrical
  - No self connection

Hopfield – Energy



- Local energy

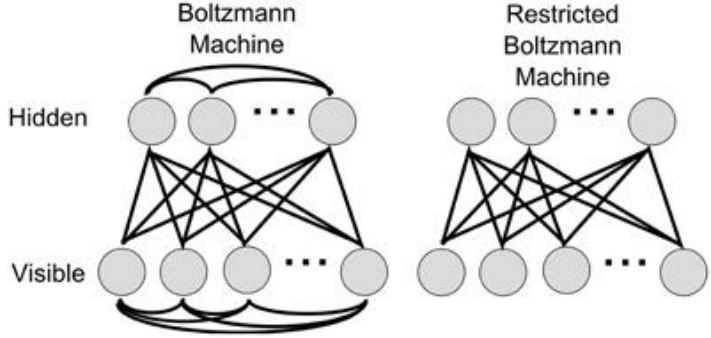
$$E_i = -\frac{1}{2} h_i x_i$$

$$E_{all} = \sum E_i = \sum_i -\frac{1}{2} h_i x_i = -\frac{1}{2} \sum_{i,j} w_{ij} x_i x_j$$

- The energy of the network can only decrease or stay at the same, if an update causes a neuron to change sign, then the energy will decrease
- Hebb's rule for weight updates

$$w_{ij} = \frac{1}{N} x_i x_j$$

Boltzmann Machine



- A stochastic network of symmetrically coupled binary units {0,1}
- Activation energy = hi
- Use labeled data to train the system hidden variables
- Probability of activation pi=f(hi)=1/(1+exp(hi))
- Eij_old=pi*pj
- After the system is stable, recalculate the visible variables
- Eij_new=pi*pj
- Weight update

  Wij=wij+L*(Eij_old-Eij_new)
- Total energy

$$E = -\frac{1}{2}\left(v^{\mathrm{T}}Lv + h^{\mathrm{T}}Jh + v^{\mathrm{T}}Wh\right)$$

- Redo the whole process until the total energy converges

Restricted Boltzmann Machine

- There is no visible-visible and hidden-hidden connections
- Contrastive divergence
  - L*(Eij_old-Eij_new) can get the machine diverge faster
  - Gradient descent in nature

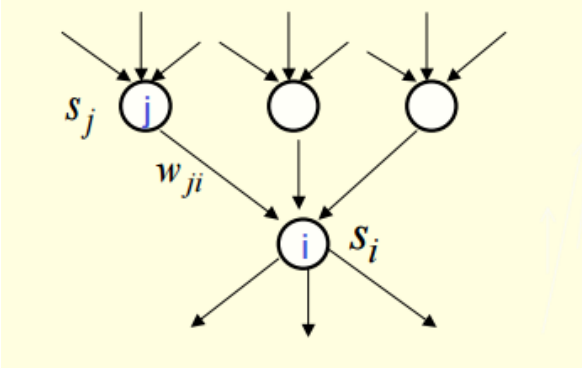
Layer-wise Pre-training and BP

- Each layer of RBM is capable for unsupervised learning
- It can also be conducted in supervised manner
- One (big) assumption is that each layer has functional locality
- After all layers are trained, BP to fine-tune the overall weight
  - Does not change the discovery structure, scaling effect

Layers (why?)

- Assuming input has hierarchical structure, the hierarchical design is more efficient for intermediate re-use
- Computationally feasible for layer computation

Sigmoid Belief Network



- Universal approximator
- Pi=f(W*X)=1/(1+exp(hi+bi))
- Why use this if there is carried away problem?

The process

- Fit W1 of first layer RBM
- Freeze W1 and use sample h1 to feed h2
- Continue until reach top level
- Do a few iteration of sampling in top level
- Carry a top-down stochastic pass

Further Resources

- http://deeplearning4j.org/restrictedboltzmannmachine.html
- http://www.cs.toronto.edu/~hinton/absps/pdp7.pdf
- http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-2/
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-sequence-learning/#recurrent-neural-networks
- http://www.iro.umontreal.ca/~pift6266/H10/notes/gradient.html
- http://deeplearning.net/tutorial/logreg.html
- https://deeplearningworkshopnips2010.files.wordpress.com/2010/09/nips10-workshop-tutorial-final.pdf
- https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf
- And there are many many more