Learning Deep Architectures for AI

Yoshua Bengio

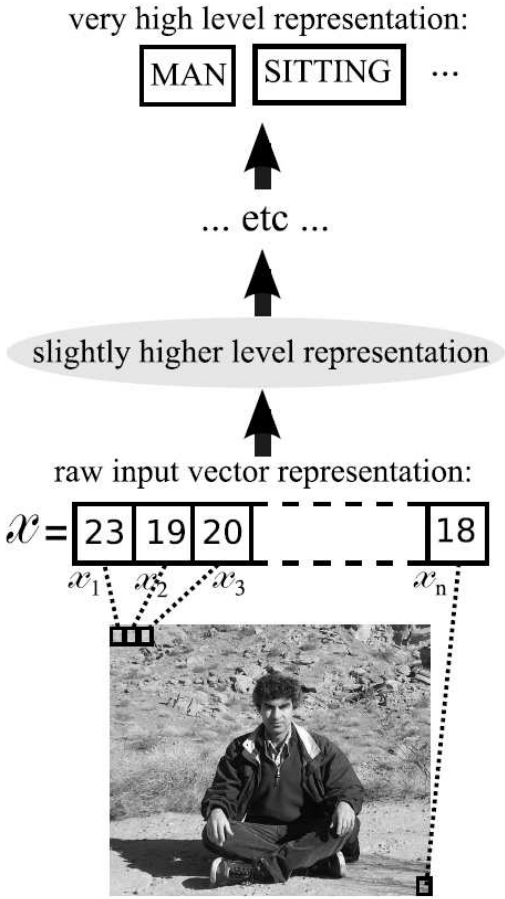
Monograph: Foundations and Trends in Machine Learning (2009)

Discussion by: Piyush Rai

July 25, 2014

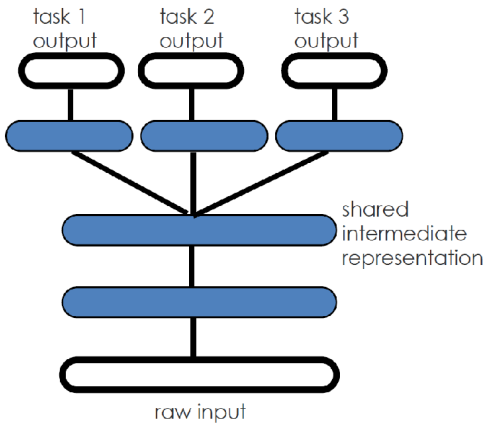
Deep Learning Architectures

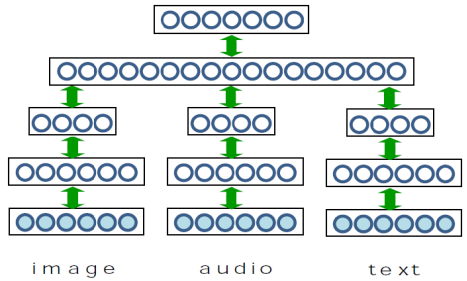
- A multilayer (hierarchical) approach to learn useful feature representations from data
- The hierarchy represents a sequence of progressively more and more abstract feature representations as we move up the hierarchy
  - E.g.: Layer 1: Edges, layer 2: local shapes, layer 3: more abstract categories, ...
- It's an attempt to mimic how information might be represented in the mammal brain



Why Deep Learning?

- Attempt to learn features that
  - are abstract enough to be robust against variations in the data we don't care about (e.g., scale/illumination/pose variations in images)
  - are good across many problems rather than being problem-specific (for transfer/multitask learning, domain adaptation)
  - capture shared aspects present in different modalities of the same objects (for learning from multimodal data)





- Hierarchical (deep) representations can often help accomplish these goals

Early Attempts: Multilayer Neural Networks

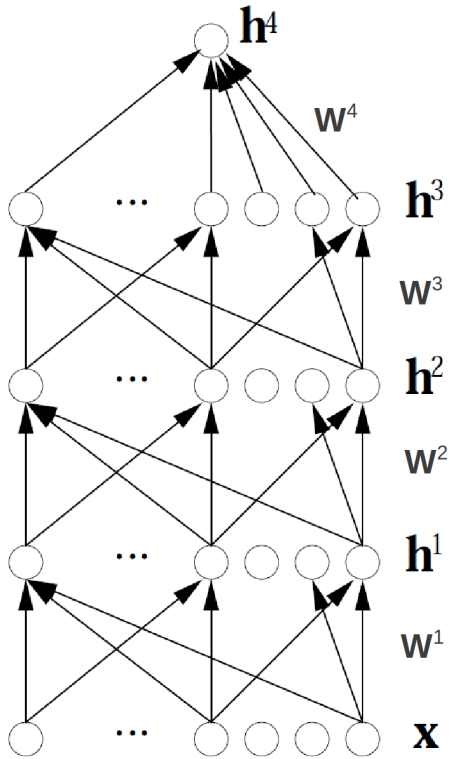
- Multilayer Neural Networks (proposed in the late 80s)

$$\mathbf{h}^k = \text{sigm}(\mathbf{b}^k + W^k \mathbf{h}^{k-1})$$

where $\text{sigm}(u) = 1/(1+\exp(-u))$ and $\mathbf{h}^0 = \mathbf{x}$

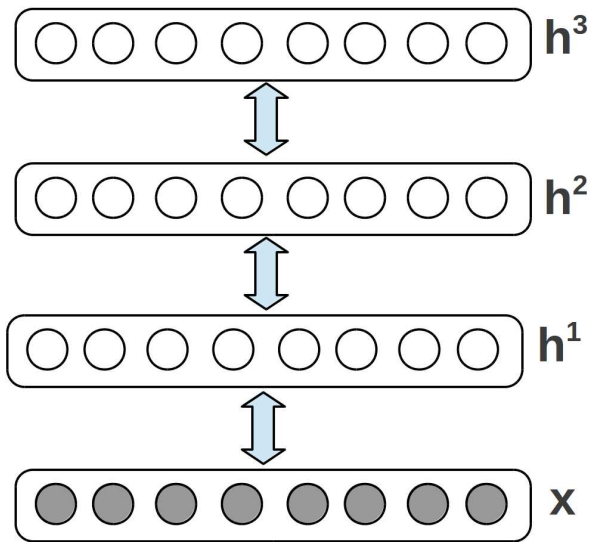
  - Purely discriminative. No generative model for the raw input features $\mathbf{x}$ (connections go upwards)
  - The top-most layer may be used for predicting a label y (e.g., in supervised learning)

- Gradient-based training of such multilayer neural networks with lots of layers is hard when using random initialization
  - Difficult to propagate gradients to the lower layers; too many connections between each pair of layers
  - Note: Some other architectures such as Deep Convolutional Networks don't face these issues



Initial Breakthrough: Layer-wise Training

- Unsupervised pre-training possible in *some* deep **generative** models (Hinton et al, 2006)
  - Greedily train one-layer-at-a-time (in an unsupervised fashion), using simple component models (e.g., Restricted Boltzmann Machine)
  - Use the learned parameters to initialize the full network and fine-tune the parameters for the task of interest (e.g., supervised classification)
- Several explanations as to why unsupervised layer-wise pre-training helps:
  - Captures statistical regularities present in the unlabeled data
  - Acts as a regularizer (or implicit prior)
  - Helps initialize the model parameters into a good region of the parameter space

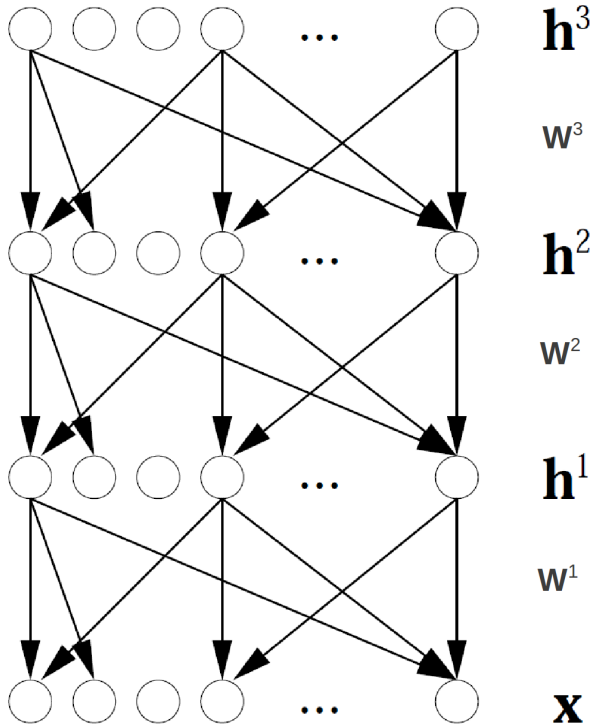


Deep Generative Models

- Not really a new idea. Sigmoid Belief Networks (SBN) studied well before 2006 (Neal, 1992)
  - SBN is essentially a generative multilayer neural network with binary units (also note the arrows go downwards, instead of upwards unlike traditional, discriminative multilayer NN)

$$\mathbf{h}^k = \text{sigm}(\mathbf{b}^k + W^k \mathbf{h}^{k+1})$$

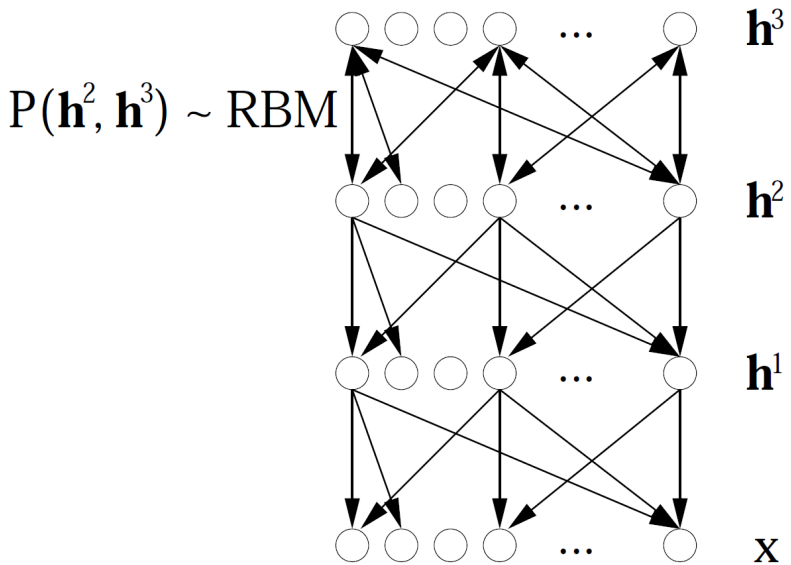
    where $\mathbf{x} = \mathbf{h}^0$
  - The hidden variables in each layer become coupled (due to explaining away) making inference hard
  - Training in SBNs was typically done using variational approximations



$$P(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l) = P(\mathbf{h}^l) \left(\prod_{k=1}^{l-1} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{x} | \mathbf{h}^1)$$

Deep Belief Networks

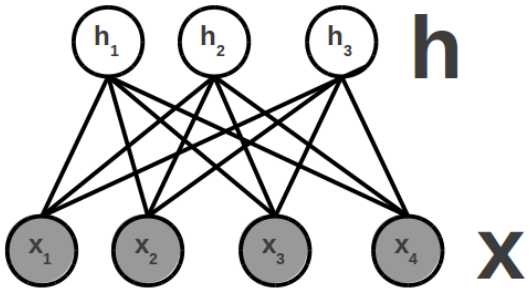
- Proposed by Hinton et al (2006)
- Similar to Sigmoid Belief Network, except the top two layers have undirected connections, modeled using a Restricted Boltzmann Machine (RBM)



$$P(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l) = P(\mathbf{h}^{l-1}, \mathbf{h}^l) \left(\prod_{k=1}^{l-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{x} | \mathbf{h}^1)$$

- This seemingly simple change allows greedy layer-wise training of DBN, using one RBM estimation problem at a time

Restricted Boltzmann Machine (RBM)



- A Boltzmann Machine defines the joint distribution between observed and hidden variables as

$$P(\mathbf{x}, \mathbf{h}) \propto e^{-Energy(\mathbf{x},\mathbf{h})}$$

where $Energy(\mathbf{x}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{h} - \mathbf{c}^\top \mathbf{x} - \mathbf{x}^\top W \mathbf{h} - \mathbf{x}^\top U \mathbf{x} - \mathbf{h}^\top V \mathbf{h}$

- A Restricted Boltzmann Machine (RBM) has connections only between hidden and observed variables (so $U = V = 0$)

$$Energy(\mathbf{x}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{h} - \mathbf{c}^\top \mathbf{x} - \mathbf{x}^\top W \mathbf{h}$$

- For the RBM, $P(\mathbf{h}|\mathbf{x}) = \prod_i P(\mathbf{h}_i|\mathbf{x})$ (therefore inference is exact for the hidden variables: very appealing)

Energy-Based Models

- Consider a set of observed variables $\mathbf{x}$ and hidden variables $\mathbf{h}$

$$P(\mathbf{x}, \mathbf{h}) = \frac{e^{-Energy(\mathbf{x},\mathbf{h})}}{Z} \qquad P(\mathbf{x}) = \sum_{\mathbf{h}} \frac{e^{-Energy(\mathbf{x},\mathbf{h})}}{Z}$$

- Another expression directly in terms of the "free energy"

$$P(\mathbf{x}) = \frac{e^{-FreeEnergy(\mathbf{x})}}{Z}$$

where $Z = \sum_{\mathbf{x}} e^{-FreeEnergy(\mathbf{x})}$, i.e., $FreeEnergy(\mathbf{x}) = -\log \sum_{\mathbf{h}} e^{-Energy(\mathbf{x},\mathbf{h})}$

- Learning in energy-based models like the RBM involves estimating parameters θ of $P(\mathbf{x})$
- This requires evaluating the gradient of (the log of) $P(\mathbf{x})$ w.r.t. θ

Evaluating Gradients in Energy-Based Models

- Gradient of the log-likelihood w.r.t. θ, in terms of the Free Energy

$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = -\frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta} + \sum_{\tilde{\mathbf{x}}} P(\tilde{\mathbf{x}}) \frac{\partial FreeEnergy(\tilde{\mathbf{x}})}{\partial \theta}$$

- Gradient of the log-likelihood w.r.t. θ, in terms of the Energy

$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = -\sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}) \frac{\partial Energy(\mathbf{x}, \mathbf{h})}{\partial \theta} + \sum_{\tilde{\mathbf{x}}, \mathbf{h}} P(\tilde{\mathbf{x}}, \mathbf{h}) \frac{\partial Energy(\tilde{\mathbf{x}}, \mathbf{h})}{\partial \theta}$$

- Need to sample from $P(\mathbf{h}|\mathbf{x})$ (positive phase) and $P(\mathbf{x}, \mathbf{h})$ (negative phase)
- For RBM, things get considerably simplified (e.g., because $P(\mathbf{h}|\mathbf{x})$ factorizes, energy/free-energy derivatives are analytically available, approximate sampling schemes such as Contrastive Divergence can be used)

Contrastive Divergence

- For RBM, positive phase sampling ($P(\mathbf{h}|\mathbf{x})$) is not needed because free-energy derivative is analytically available
- The negative phase sampling is nevertheless needed for $P(\mathbf{x}, \mathbf{h})$
- MCMC (Gibb sampling) is easy for RBM but expensive
- Contrastive Divergence uses two approximations
  - Summation $\sum_{\tilde{\mathbf{x}},\mathbf{h}}$ over all possible inputs replaced by a single sample $\tilde{x}, \mathbf{h}$
  - The MCMC chain to sample $\tilde{\mathbf{x}}, \mathbf{h}$ is run only for k steps (starting with the observed example $\mathbf{x}$): called the k-step Contrastive Divergence (CD-k)
- The updates are given by (upon seeing an example $\mathbf{x}$, and collecting the $k+1$-th step sample $\tilde{\mathbf{x}}$ for the chain initialized with $\mathbf{x}$)

$$\Delta\theta \propto \frac{\partial FreeEnergy(\mathbf{x})}{\partial\theta} - \frac{\partial FreeEnergy(\tilde{\mathbf{x}})}{\partial\theta}$$

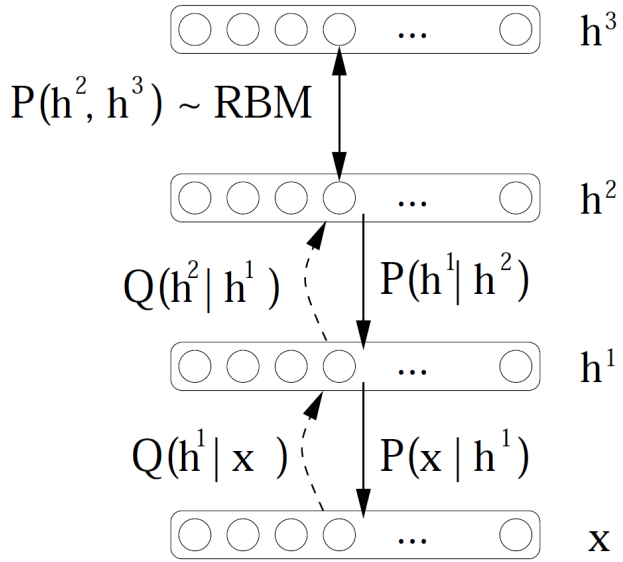
Layer-wise Training of Deep Belief Networks

- Joint distribution of observed and hidden units

$$P(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l) = P(\mathbf{h}^{l-1}, \mathbf{h}^l) \left(\prod_{k=0}^{l-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right)$$

  where $\mathbf{x} = \mathbf{h}^0$

- Suppose $Q(\mathbf{h}^l | \mathbf{h}^{l-1})$ denotes the posterior for hidden variables at layer l. Q is approximate for all layers except the top RBM layer (since RBM inference is exact)
- Each layer is simply treated as an RBM to approximate $Q(\mathbf{h}^l | \mathbf{h}^{l-1})$
- The DBN parameters, thus learned, can be fine-tuned for other supervised tasks
- If desired, samples can be generated using the hidden-to-visible conditionals



Next Week

- Justifications for the layer-wise training
  - Variational justification
  - Annealing and curriculum learning based justifications
- Variants of RBM
- Autoencoder and its variants
- Other deep architectures
  - Deep Convolutional Nets
  - Deep Boltzmann Machines
- Some other recent works..

Thanks! Questions?