Learning and Inference in Deep, Unsupervised Neural Networks

Kyunghyun Cho

Department of Information and Computer Science
Aalto University, School of Science
kyunghyun.cho@aalto.fi

21 March 2014

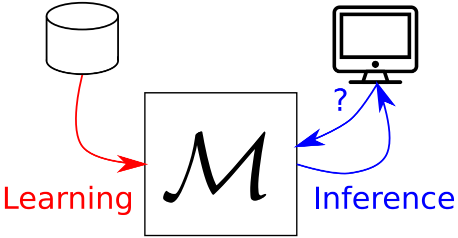
This work was done under the supervision of Prof. Juha Karhunen, Prof. Tapani Raiko and Dr. Alexander Ilin at the Deep Learning and Bayesian Modeling Group, Department of Information and Computer Science, Aalto University School of Science between mid 2009 and early 2014.

Aalto University
School of Science

Boltzmann machines? I remember working on them in 80s and 90s..

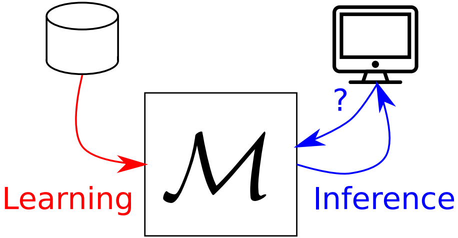
– Anonymous, 2011
paraphrased

Machine Learning in a Single Slide



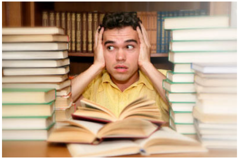
1. Let the model $\mathcal{M}$ *learn* the data D
2. Let the model $\mathcal{M}$ *infer* unknown quantities

Examples

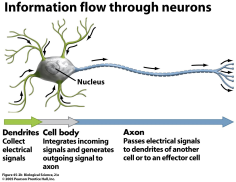


Data	Query
Movie Ratings	Will a user X like a movie Y?
Tagged Images	Is a cat in this image?
Transcribed Speech	What is this person saying?
Parallel Corpora	What is “moi” in English?

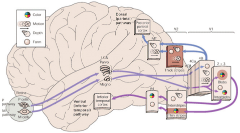
Deep Learning in a Single Slide



Learn massive data

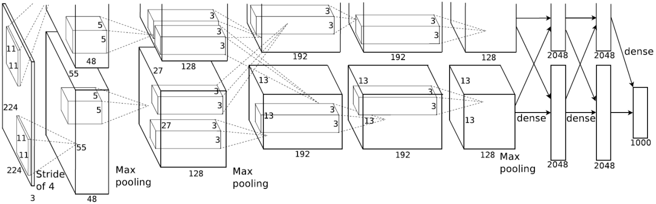


simple functions



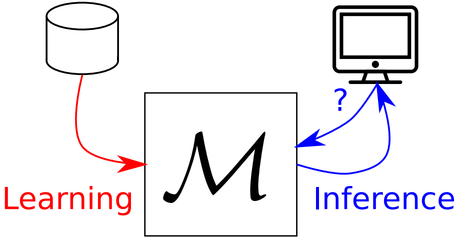
(Van Essen&Gallant, 1994)

Multi-layered



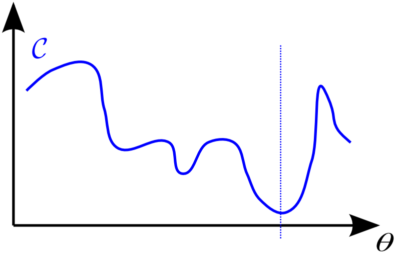
(Krizhevsky et al., 2012)

Challenges in Machine Learning



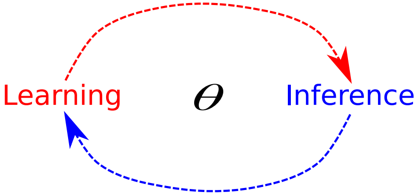
1. Learning is *not* trivial
2. Learning and inference are *not* separate

Learning Difficulties



- True cost $\mathcal{C}$ is *not* available: Only empirical cost $\tilde{\mathcal{C}}$ available
- Often, non-convex optimization with many local/apparent minima
- Impractical to compute either $\mathcal{C}$ or $\tilde{\mathcal{C}}$, as $|D| \to \infty$

Vicious Cycle or Virtuous Cycle?



Example: MLE for feedforward neural networks

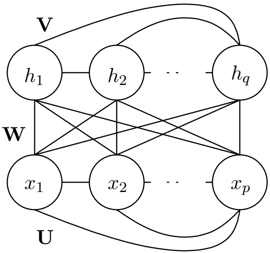
$$\min_{\boldsymbol{\theta}} \mathcal{C}(\boldsymbol{\theta}) \approx \min_{\boldsymbol{\theta}} \tilde{\mathcal{C}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} -\frac{1}{N} \sum_{(\mathbf{x},t)\in D} \log p(y = t \mid \mathbf{x}, \boldsymbol{\theta})$$

What gets worse with *deep* learning?

- Learning
  - No access to $\mathcal{C}$, but only to $\tilde{\mathcal{C}}$
  - Highly entangled inference and learning
  - **High-dimensional**
  - **Non-convex with a lot of local (apparent) minima**
  - **Intractable to compute even** $\tilde{\mathcal{C}}$, because $|D| \to \infty$
- Inference
  - **No analytical expression**, often
  - **Intractable to compute**, often
  - Difficult to analyze and understand

Deep, Unsupervised Neural Networks
Boltzmann Machines and Autoencoders

Boltzmann Machines



Popular variants:

- If $\mathbf{V} = \mathbf{0}$ and $\mathbf{U} = \mathbf{0}$, restricted Boltzmann machine (RBM)
- If $\mathbf{U} = \mathbf{0}$ and layered $\mathbf{h}$, deep Boltzmann machine (DBM)

1. Negative energy over $\mathbf{x}$ and $\mathbf{h}$:

$$-E(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \mathbf{b}^\top \mathbf{x} + \mathbf{c}^\top \mathbf{h} + \mathbf{x}^\top \mathbf{W} \mathbf{h} + \frac{1}{2} \mathbf{x}^\top \mathbf{U} \mathbf{x} + \frac{1}{2} \mathbf{h}^\top \mathbf{V} \mathbf{h}$$

2. Probability over $\mathbf{x}$ and $\mathbf{h}$:

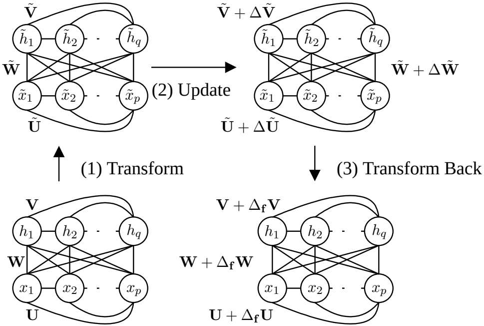
$$p(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{-E\left(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}\right)\right\}$$

3. Learn $p(\mathbf{x})$ by maximizing

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \in D}\left[\log \frac{1}{Z(\boldsymbol{\theta})} \sum_{\mathbf{h}} e^{-E(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta})}\right]$$

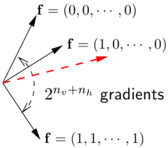
with stochastic gradient descent

Learning: Observations

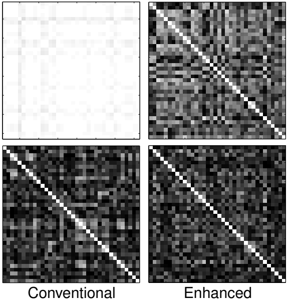


- Invariant to bit-flipping transformation
- 2^{p+q} descent directions
- leading to (potentially all) different solutions
- Which direction?
- How to tractably decide?

Learning: Enhanced Gradient



Covariance of Weight Updates



- Importance/weight of each direction $\nabla_{\mathbf{f}} \mathcal{L}$

$$\prod_{k=1}^{n_v+n_h} \langle x_k \rangle_{\mathrm{dm}}^{f_k} \left(1 - \langle x_k \rangle_{\mathrm{dm}}\right)^{1-f_k}$$

- Weighted sum of all possible updates

$$\nabla_e w_{ij} = \mathrm{Cov}_{\mathrm{d}}\left(x_i, h_j\right) - \mathrm{Cov}_{\mathrm{m}}\left(x_i, h_j\right)$$

$$\nabla_e b_i = \langle x_i \rangle_{\mathrm{d}} - \langle x_i \rangle_{\mathrm{m}} - \sum_j \langle h_j \rangle_{\mathrm{dm}} \nabla_e w_{ij}$$

$$\nabla_e c_j = \langle h_j \rangle_{\mathrm{d}} - \langle h_j \rangle_{\mathrm{m}} - \sum_i \langle x_i \rangle_{\mathrm{dm}} \nabla_e w_{ij}$$

Learning vs. Inference

- Boltzmann machine learning requires inference.

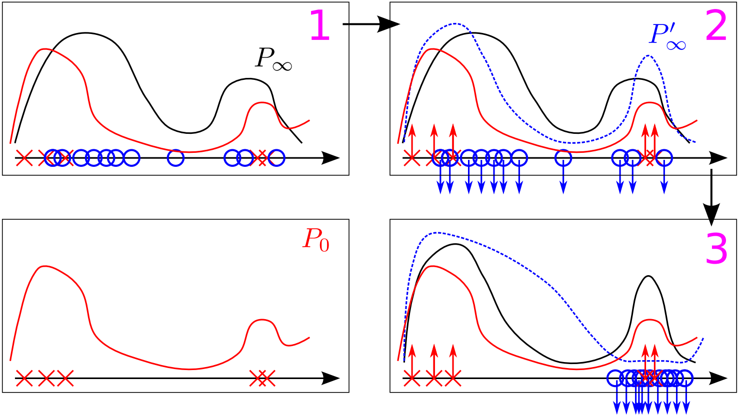
$$\nabla_e w_{ij} = \mathrm{Cov_d}\left(x_i, h_j\right) - \mathrm{Cov_m}\left(x_i, h_j\right)$$

- *What is the covariance between x_i and h_j with the current $\boldsymbol{\theta}$?*
  - NP-Hard problem even in the case of RBMs (Long&Servedio, 2010)
  - Monte Carlo approximation with persistent MCMC

$$\mathrm{Cov_m}\left(x_i, h_j\right) \approx \left(\frac{1}{N}\sum_{n=1}^{N} x_i^{(n)} h_j^{(n)}\right) - \left(\frac{1}{N}\sum_{n=1}^{N} x_i^{(n)}\right)\left(\frac{1}{N}\sum_{n=1}^{N} h_j^{(n)}\right)$$

  - Gibbs sampling

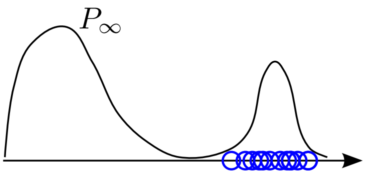
Learning vs. Inference: Vicious Cycle

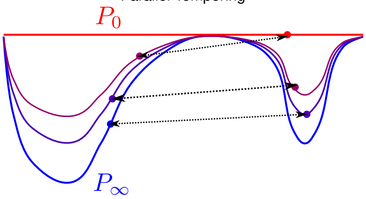


Failed inference (sampling) breaks learning

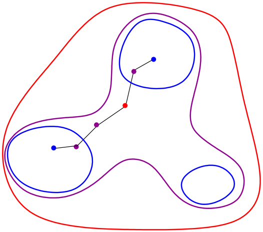
$\rightarrow$ *Good MCMC sampler is needed!*

Inference: Better MCMC Sampler





- MCMC with a *local* jump cannot easily escape an isolated mode

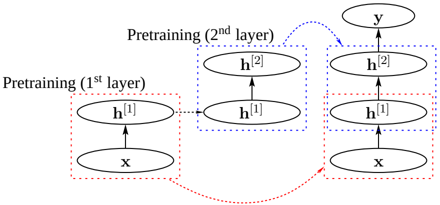


Parallel Tempering

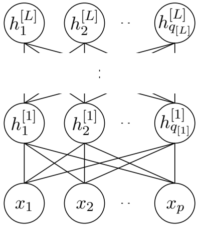
- Parallel chains between P_0 and P_∞
- Jump via tempered chains
- Better exploration of the state space

From an RBM to a *deeper* neural network..

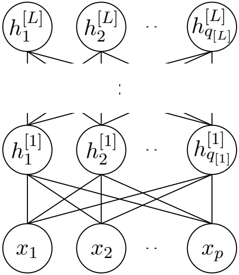
Deep Belief Network
(Pretrained MLP)



Deep Boltzmann Machine



Deep Boltzmann Machines



- *Undirected* Hierarchical Model
- Negative Energy

$$
\begin{aligned}
-E(\mathbf{x}, \mathbf{h} \mid \boldsymbol{\theta}) = \\
&\mathbf{b}^\top \mathbf{x} + \mathbf{c}_{[1]}^\top \mathbf{h}_{[1]} + \mathbf{x}^\top \mathbf{W} \mathbf{h}_{[1]} \\
&+ \sum_{l=2}^{L} \left(\mathbf{c}_{[l]}^\top \mathbf{h}_{[l]} + \mathbf{h}_{[l-1]}^\top \mathbf{U}_{[l-1]} \mathbf{h}_{[l]} \right)
\end{aligned}
$$

- The further away a layer from **x**, the more abstract concept the layer learns
- Hierarchical representation with both bottom-up and top-down signals

Learning: Depressing Observation

Observation: Lack of Structures in Deeper Hidden Layers

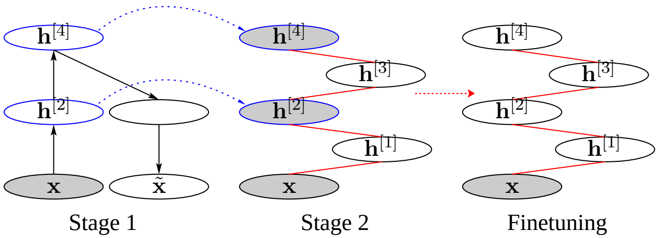
Which direction does learning move toward? – matching data and model statistics

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial u_{ij}^{[l]}} \propto \left\langle h_i^{[l]} h_j^{[l+1]} \right\rangle_{p(\mathbf{h}|\mathbf{v},\boldsymbol{\theta})p_D(\mathbf{v})} - \left\langle h_i^{[l]} h_j^{[l+1]} \right\rangle_{p(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})}$$

What happens if $p(\mathbf{h} \mid \mathbf{v}, \boldsymbol{\theta})$ does not have *any* structure?

- Learning will *not* utilize deeper layers easily
- Especially severe at the intial stage of learning

Learning: Hierarchical structure *borrowed* from DBN



Stage 1 Recursively train a stack of RBMs to get $Q(\mathbf{h}_- \mid \mathbf{x})$

Stage 2 Train a large RBM $\iff$ Maximize the variational lower-bound of DBM

$$\mathbb{E}_{D(\mathbf{v})}\left[\log p(\mathbf{v}^{(n)} \mid \boldsymbol{\theta})\right] \geq \mathbb{E}_{D(\mathbf{v})Q(\mathbf{h}_-)}\left[\log \sum_{\mathbf{h}_+} e^{\left\{-E(\mathbf{v}^{(n)},\mathbf{h}_-,\mathbf{h}_+)\right\}}\right] + \mathcal{H}(Q) - \log Z(\boldsymbol{\theta})$$

Q: Have I worked on anything other than Boltzmann machines?

Unsupervised Learning: Encoder-Decoder Perspective

- Sparse coding:

Encoder $\mathbf{h} = \arg\min_{\mathbf{h}} \|\mathbf{x} - \mathbf{W}^\top \mathbf{h}\| + \lambda \Omega(\mathbf{h})$

Decoder $\mathbf{x} = \mathbf{W}^\top \mathbf{h}$

- Probabilistic PCA:

Encoder $\mathbb{E}[\mathbf{h}] = (\mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I})^{-1} \mathbf{W}^\top \mathbf{x}$

Decoder $\mathbb{E}[\mathbf{x}] = \mathbf{W}^\top \mathbf{h}$

- RBM:

Encoder $\mathbb{E}[\mathbf{h}] = \sigma(\mathbf{W}\mathbf{x} + \mathbf{c})$

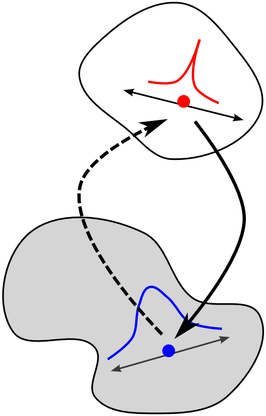
Decoder $\mathbb{E}[\mathbf{x}] = \sigma(\mathbf{W}^\top \mathbf{h} + \mathbf{b})$

- DBM:

Encoder $\boldsymbol{\mu}^{[l]} \leftarrow \sigma(\mathbf{U}_{[l]}^\top \boldsymbol{\mu}^{[l+1]} + \mathbf{U}_{[l-1]} \boldsymbol{\mu}^{[l-1]} + \mathbf{c}^{[l]}),$

$\mathbb{E}\left[\mathbf{h}^{[l]}\right] \approx \boldsymbol{\mu}^{[l]}$

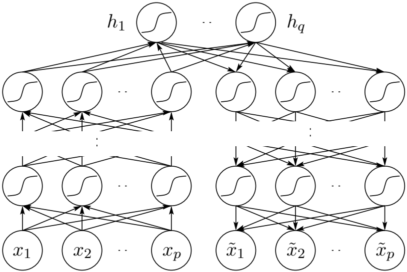
Decoder $\mathbb{E}[\mathbf{x}] = \sigma(\mathbf{W}^\top \mathbf{h}^{[1]} + \mathbf{b})$



Denoising Autoencoder: Explicit Sparsification

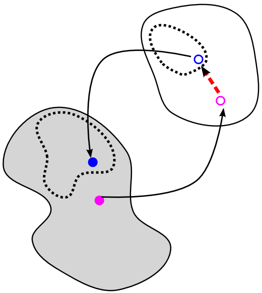
Sparse Denoising Autoencoder

- Encoder $\mathbf{h} = f(\mathbf{x}) : \mathbb{P} \to \mathbb{Q}$
- Decoder $\tilde{\mathbf{x}} = g(\mathbf{h}) : \mathbb{Q} \to \mathbb{P}$



$$\mathbb{P} = \left\{ \mathbf{x} \in \mathbb{R}^p \,\middle|\, \exists \mathbf{x}^{(n)} \in D, \|\mathbf{x} - \mathbf{x}^{(n)}\|_2^2 \leq \epsilon \right\}$$
$$\mathbb{Q} \approx \left\{ \mathbf{h} = f(\mathbf{x}) \,\middle|\, \mathbf{x} \in \mathbb{P}, \|\mathbb{E}_{\mathbf{x} \in \mathbb{P}}[h_j] - \rho\|_2^2 = 0 \right\}$$

What do we do with $\mathbf{x} \notin \mathbb{P}$?



1. Encode: $\mathbf{h} = f(\mathbf{x})$
2. Sparsify: $\tilde{\mathbf{h}} = R(\mathbf{h})$
3. Decode: $\tilde{\mathbf{x}} = g(\tilde{\mathbf{h}})$

And beyond..

- Theoretical understanding beyond universal approximator properties
- Deep learning for long sequences
- Deep learning for $p \gg n$ and $n \to 1$
- New models that tackles learning and inference directly