From Helsinki to Machine Translation

KyungHyun Cho
kyunghyun.cho@aalto.fi

Deep Learning and Bayesian Modeling Group,
Department of Information and Computer Science,
Aalto University School of Science

September 19, 2013

For your information. . .

In English

- Last name: *Cho*
- First name: *Kyunghyun* or *Kyung Hyun*

In Korean,

조경현

Montreal $\Longleftrightarrow$ Helsinki

Aalto University and ICS Department

Aalto University was founded in 2010 by merging

1. Helsinki University of Technology
2. Helsinki School of Art and Design
3. Helsinki School of Economics

Department of *Information and Computer Science (ICS)*

1. Theoretical Computer Science
2. **Information Science**: *where I'm at*

Neural Network Research at Aalto: Earlier Time

Long, long time ago (- 1994)

80's

- (Kohonen, 1982) Self-Organizing Map
- (Oja, 1982) Neural Principal Component Analysis

The Neural Networks Research Centre (1995 - 2005)

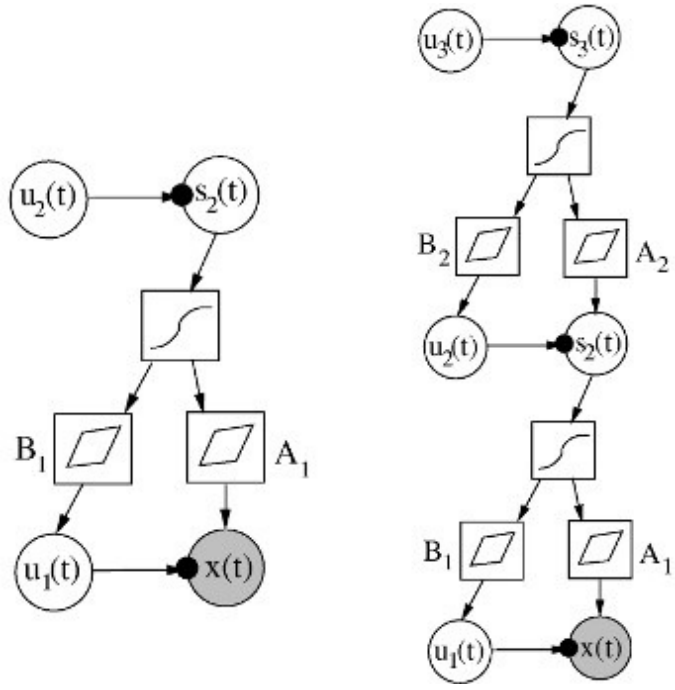
90's – early 2000's

- (Oja, 1991) 5-layer Autoencoder for Nonlinear PCA
- (Valpola, 1996) Sparse coding
- (Hyvärinen et al., 2001) Independent Component Analysis

The Adaptive Informatics Research Centre (2006 - 2011)

Finnish Centre of Excellence in Computational Inference Research (2012 - 2017)

Dark Age started by taking *Bayesian* treatment too seriously on *deep* neural networks. . .



(Raiko, 2001)

- Raiko later said, ”we should’ve used point-estimates for parameters”

Machine Learning Other than Neural Networks

Diverse resarch areas

- Bioinformatics: Prof. Sami Kaski, Prof. Lähdesmäki...
- Image and Information Retrieval: Dr. Jorma Laaksonen...
- Industrial Applications: Dr. Amaury Lendasse...
- Multi-view, Multi-task Learning: Prof. Sami Kaski, Dr. Jaakko Peltonen...
- Speech Recognition: Prof. Mikko Kurimo, Dr. Kalle Palomäki
- Non-negative Matrix Factorization: Prof. Erkki Oja, Dr. Zhirong Yang...
- Bayesian PCA: Dr. Tapani Raiko, Dr. Alexander Ilin...
- and so on...

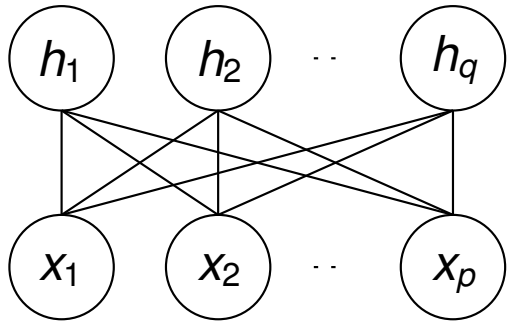
It's only *neural networks* that went through the dark age.

Neural Networks performs Inference

Some alternative views of neural networks:

1. Nonlinear function approximator (supervised)
2. Multi-layered feature extractor (unsupervised)
3. *Fast inference engine*

Example: RBM vs Autoencoder

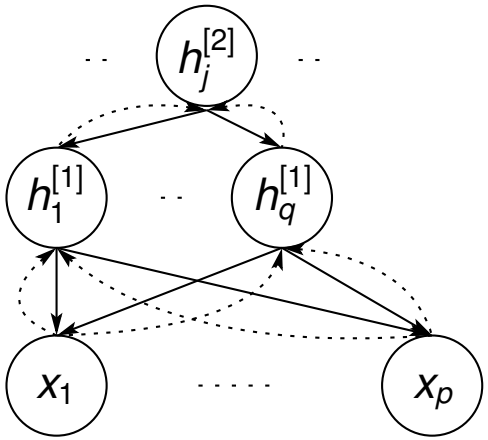


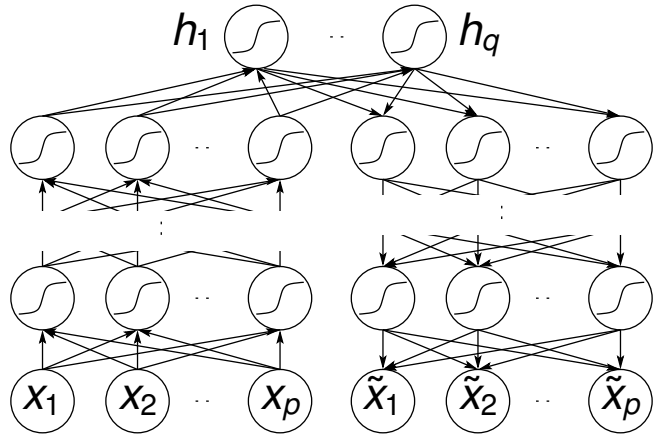
	RBM	Autoencoder
Inference	$\mathbb{E}[\mathbf{h}] = \sigma\left(\mathbf{W}^\top \mathbf{x}\right)$	$\mathbf{h} = \sigma\left(\mathbf{W}^\top \mathbf{x}\right)$
Generation	$\mathbb{E}[\mathbf{x}] = \sigma(\mathbf{W}\mathbf{h})$ $(\mathbf{h} \sim \mathcal{B}(\mathbb{E}[\mathbf{h}]))$	$\mathbf{x} = \sigma(\mathbf{U}\mathbf{h})$ (often, $\mathbf{U} = \mathbf{W}$)
Learning	Maximum Likelihood	Minimum Reconstruction
	Stochastic	Deterministic*

An RBM is a *probabilistic* model of which inference and generation can correspond *exactly* to the feedforward computation of a neural network.

* I acknowledge that Yoshua may disagree.

Example: Sigmoid Belief Net* as a Stochastic Autoencoder





	SBN	Autoencoder
Inference	Feedforward + Sampling	Feedforward
Generation	Feedforward + Sampling	Feedforward
Learning	Variational Lower-bound	Minimum Reconstruction
Difference	Stochastic	Deterministic

* This argument is strictly limited to the case of training a sigmoid belief network with wake-sleep algorithm (Hinton et al., 1995)

However, it all breaks down with Deep Boltzmann Machines...

Let's assume that it's still early 2013, before Goodfellow et al.'s work on DBM and Bengio et al.'s work on GSN.

Inference in DBM requires both bottom-up and top-down signals:

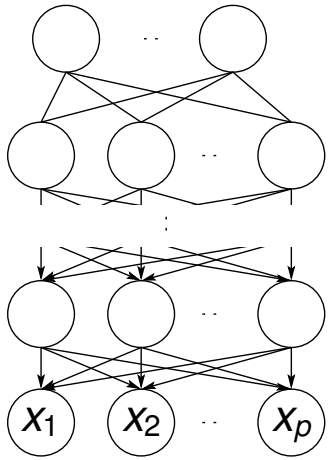
$$p(h_j^{[l]} = 1 \mid \mathbf{h}^{[l-1]}, \mathbf{h}^{[l+1]}, \theta) = \phi\left(\sum_{k=1}^{q_{l-1}} h_k^{[l-1]} w_{kj}^{[l-1]} + \sum_{i=1}^{q_{l+1}} h_i^{[l+1]} w_{ji}^{[l]} + c_j^{[l]}\right),$$

This does not correspond well with any feedforward neural network.

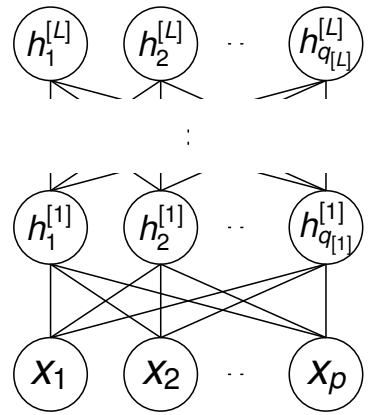
Can we use this mismatch to our advantage?

Let's further assume that no one knows whether the centering trick works well on large DBMs yet and that everyone wants to train a DBM.

Deep Belief Network



Deep Boltzmann Machine

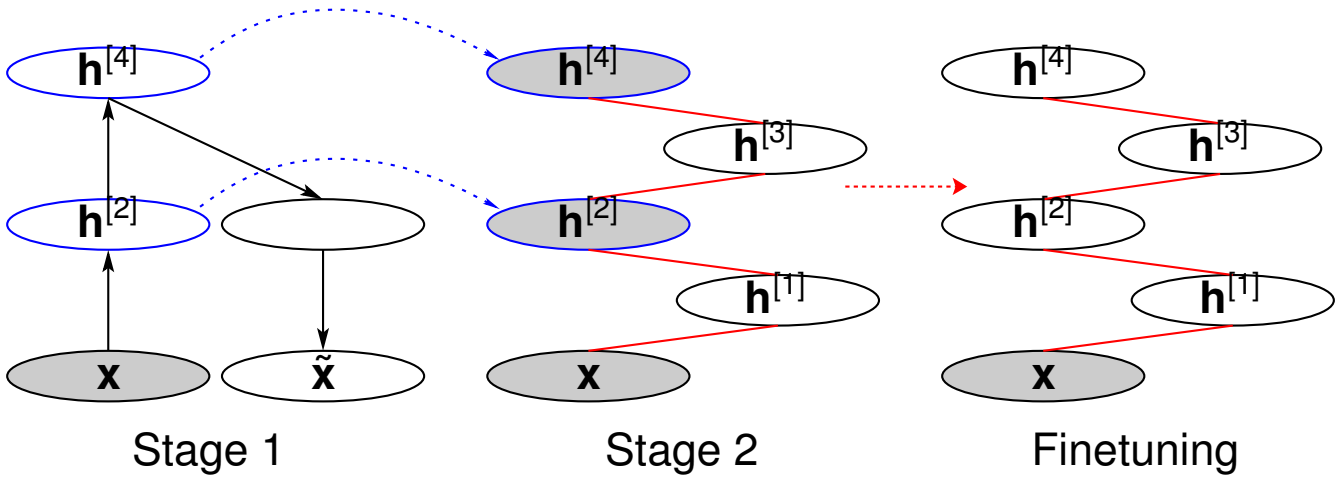


Few motivations:

- We can *train directed models well* either with or without pretraining
- Except for the directedness of arrows, *DBM and DBN look same*
- We believe that DBN extracts a *good abstract concept* of an input

Two-Stage Pretraining Algorithm

1. Train a deep belief net (or a deep autoencoder)
2. Fit the posterior of a DBM to the posterior from the DBN
3. Continue finetuning



- In fact, you can simultaneously perform Stage 1 and 2
- Though, failed to find anyone who likes this idea so far

Disclaimer

I miserably failed the course "Statistical Natural Langauge Processing" in 2010. Other than those dealt during the first two lectures of the course my knowledge about natural language processing has been close to nil. Hence, I do not guarantee that the following materials will eventually turn out to be either novel, meaningful, interesting or worth spending your hours.

Statistical Machine Translation from Scratch

Generative model $\mathcal{M}_t$ for a language $L_t = \{\mathbf{s}_0, \mathbf{s}_1, \dots\}$:

1. Select a component from a cateogorical distribution

$$c \sim \mathcal{C}(\mathbf{p})$$

2. Generate a sequence from the selected stochastic process

$$\mathbf{s} = (s_0, s_1, s_2, \dots) \sim \mathcal{LP}_t(\boldsymbol{\theta}_c)$$

Note that

- $\mathbf{p}$ is independent of a language
- But, $\mathcal{LP}_t$ is dependent on a language

Statistical Machine Translation from Scratch

Machine translation between L_s and L_t in two steps:

1. Find the most likely *concept* of a source sentence

$$\hat{c} = \arg\max_{c} p\left(c \mid \mathbf{s}_s, \mathcal{M}_s\right)$$

2. Find the most likely *sentence* of a target language

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} p\left(\mathbf{s} \mid \theta_{\hat{c}}, \mathcal{M}_t\right)$$

Approaching it by Neural Networks

Machine translation between L_s and L_t in two steps:

1. **Inference** An encoder extracts $\hat{\mathbf{z}}$ from $\mathbf{s}_s$ of length K

$$\mathbf{z}^{k+1} = f(s_s^{k+1}, \mathbf{z}^k), t = 0, \ldots, K-1$$
$$\hat{\mathbf{z}} = \mathbf{z}^K$$

2. **Generation** A decoder biased with $\mathbf{z}$ generates $\mathbf{s}_t$

$$s_t^{k+1}, \mathbf{h}^{k+1} = g(s_t^k, \mathbf{h}^k, \hat{\mathbf{z}}) \text{ until } s_t^{k+1} = \langle \text{stop} \rangle$$

Note that

- $\hat{\mathbf{z}}$ is $\hat{c}$
- The encoder and decoder may be trained jointly

Current Challenges

1. Is this a right approach?
2. Isn't it too hard. . . ?
3. Is the choie of Recurrent NNs right?
4. This is an extremely deep NN with a tiny bottleneck. Is it even possible to train it well?
5. . . .

Far beyond: will machine learning prove **P** $=$ **NP** or **P** $\neq$ **NP**?

Decision problems in **NP***: Binary classification with N-dim binary input

Assuming that trainining time of a classifier is $O(M)$ with M training samples:

- How many samples do we need to train a classifier that classifies **NP**-complete problems with ϵ error?
- Is M polynomially bounded with respect to N?
- Is obtaining M training samples bounded polynomially with respect to N?

The solution must be verifiable in polynomial time.