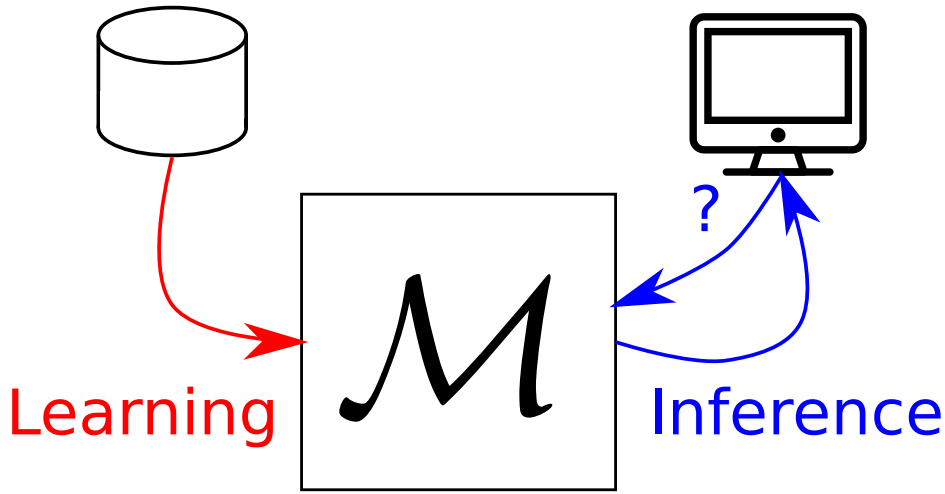
Deep Learning: Past, Present and Future (?)

Kyunghyun Cho

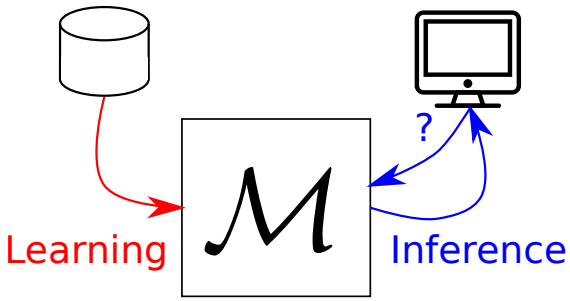
Laboratoire d'Informatique des Systèmes Adaptatifs,
Département d'informatique et de recherche opérationnelle,
Facult des arts et des sciences,
Université de Montréal
cho.k.hyun@gmail.com (chokyun@iro.umontreal.ca)

Machine Learning



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

Machine Learning & Perception

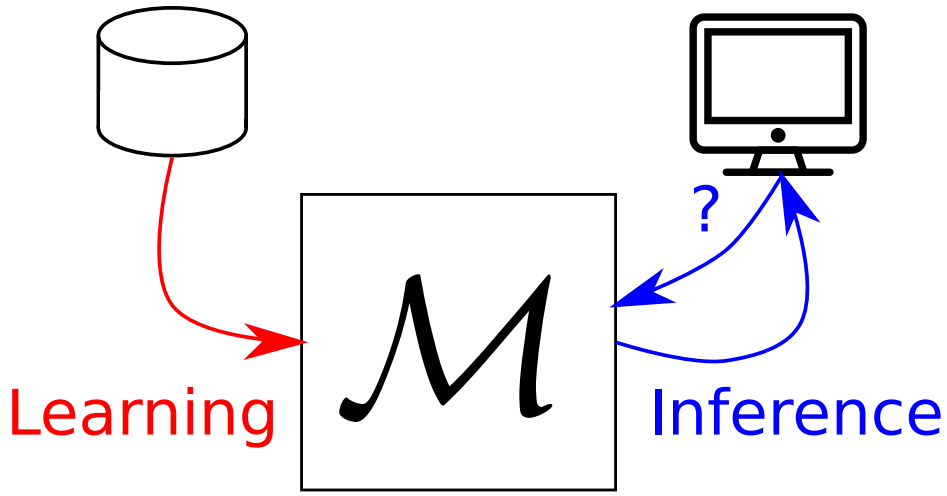


Perception is the organization, identification, and interpretation of *sensory information* in order to represent and *understand the environment*.

–Wikipedia

(Farabet et al., 2013)

Machine Learning & Perception: Examples

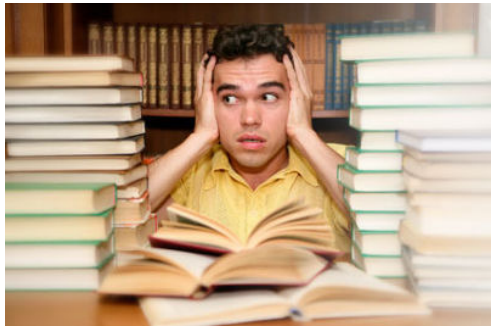


Data	**Sensory Information**	**Query**
Labeled Images	An image	Is a cat in the image?
Transcribed Speech	A speech segment	What is this person saying?
Paraphrases	A pair of sentences	Is this sentence a paraphrase?
Movie Ratings	Ratings of Y and by X	Will a user X like a movie Y?
Parallel Corpora	A Finnish sentence	What is "moi" in English?

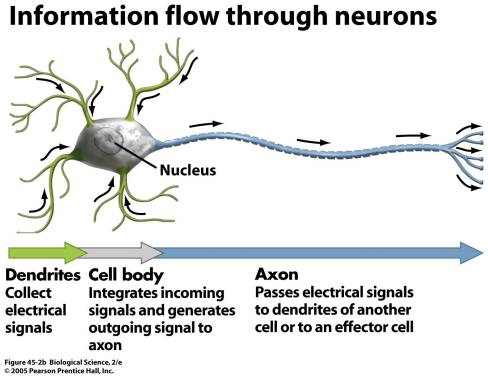
A human possesses *the* best machinery for perception, called a **brain**.

But, how does our brain do it?

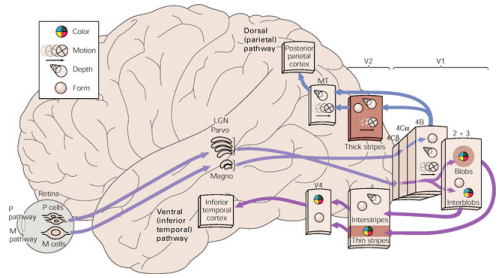
Deep Learning: Motivated from Human Learning



Learn massive data

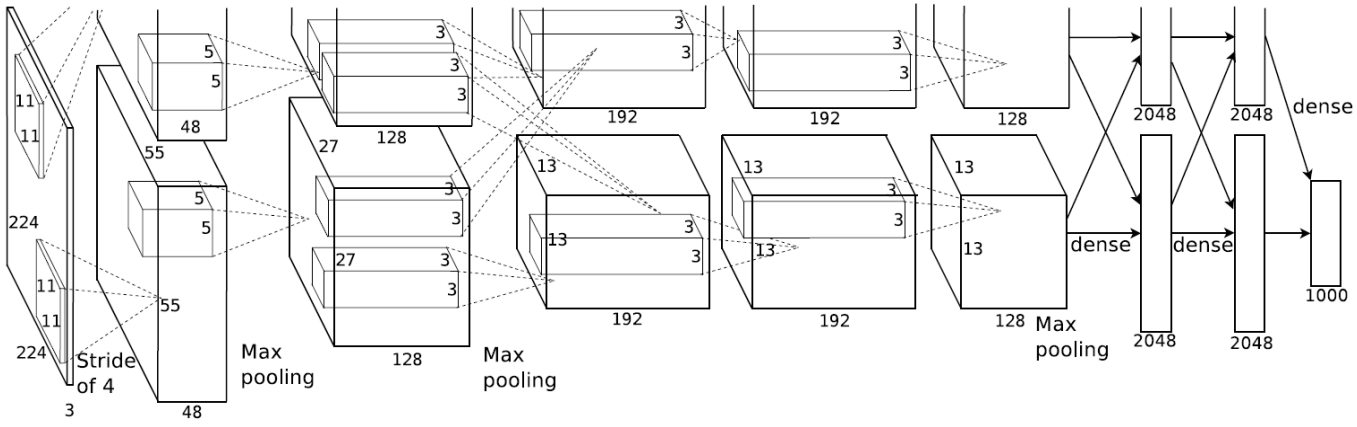


simple functions



(Van Essen&Gallant, 1994)

Multi-layered



(Krizhevsky et al., 2012)

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

– Anonymous Interviewer, 2011
paraphrased

Deep Learning: History

1958 Rosenblatt proposed perceptrons

1980 Neocognitron (Fukushima, 1980)

1982 Hopfield network, SOM (Kohonen, 1982), Neural PCA (Oja, 1982)

1985 Boltzmann machines (Ackley et al., 1985)

1986 Multilayer perceptrons and backpropagation (Rumelhart et al., 1986)

1988 RBF networks (Broomhead&Lowe, 1988)

1989 Autoencoders (Baldi&Hornik, 1989), Convolutional network (LeCun, 1989)

1992 Sigmoid belief network (Neal, 1992)

1993 Sparse coding (Field, 1993)

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad.
(Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

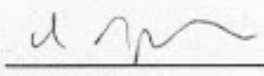
****

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

3/14/95

V. Vapnik



3/14//95

L. Jackel

3/14/95

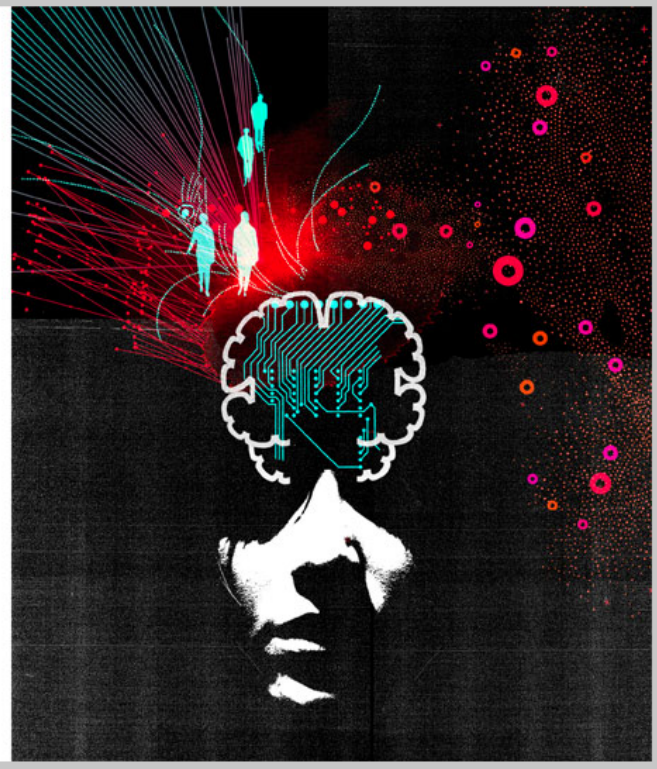
Witnessed by Y. LeCun

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

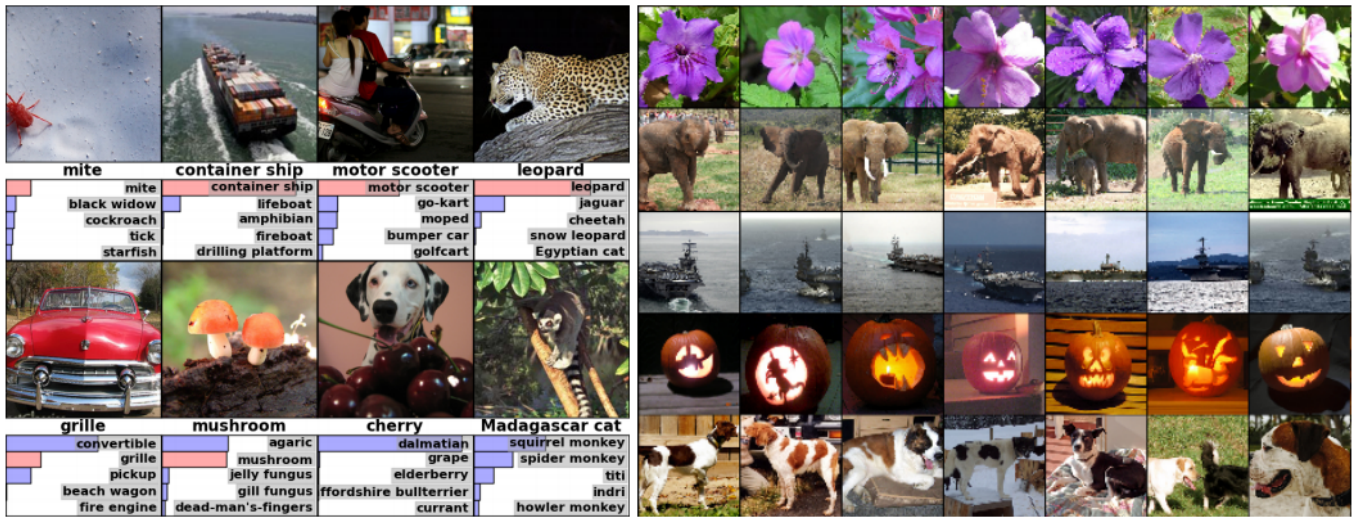


Why all this fuss about deep learning now?

ImageNet: ILSVRC 2012 – Classification Task

Top Rankers

1. SuperVision (**0.153**): Deep Conv. Neural Network (Krizhevsky et al.)
2. ISI (0.262): Features + FV + Linear classifier (Gunji et al.)
3. OXFORD_VGG (0.270): Features + FV + SVM (Simonyan et al.)
4. XRCE/INRIA (0.271): SIFT + FV + PQ + SVM (Perronin et al.)
5. University of Amsterdam (0.300): Color desc. + SVM (van de Sande et al.)



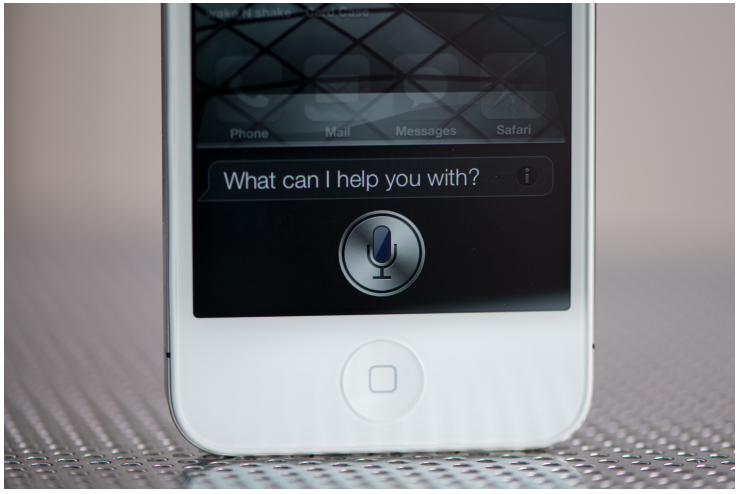
(Krizhevsky et al., 2012)

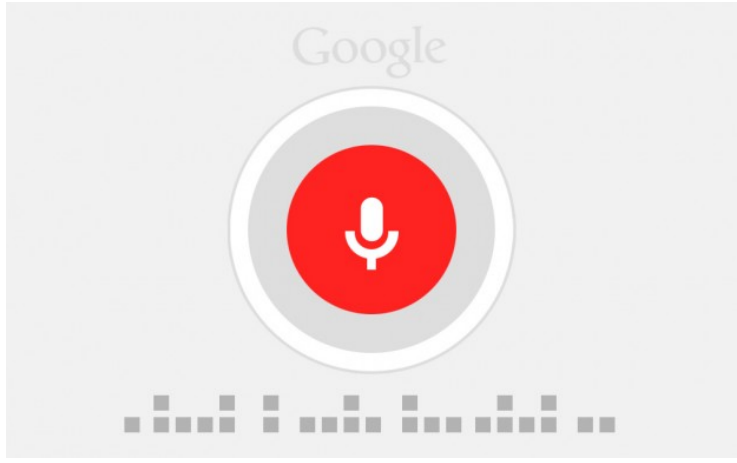
ImageNet: ILSVRC 2013 – Classification Task

Top Rankers

1. Clarifi (**0.117**): **Deep Convolutional Neural Networks (Zeiler)**
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional Neural Networks**
7. Adobe: **Deep Convolutional Neural Networks**
8. VGG: **Deep Convolutional Neural Networks**
9. CognitiveVision: **Deep Convolutional Neural Networks**
10. decaf: **Deep Convolutional Neural Networks**
11. IBM Multimedia Team: **Deep Convolutional Neural Networks**
12. Deep Punx (0.209): **Deep Convolutional Neural Networks**
13. *MIL (0.244): Local image descriptors + FV + linear classifier (Hidaka et al.)*
14. Minerva-MSRA: **Deep Convolutional Neural Networks**
15. Orange: **Deep Convolutional Neural Networks**
16. BUPT-Orange: **Deep Convolutional Neural Networks**
17. Trimps-Soushen1: **Deep Convolutional Neural Networks**
18. *QuantumLeap: 15 features + RVM (Shu&Shu)*

You're already using *deep learning*!

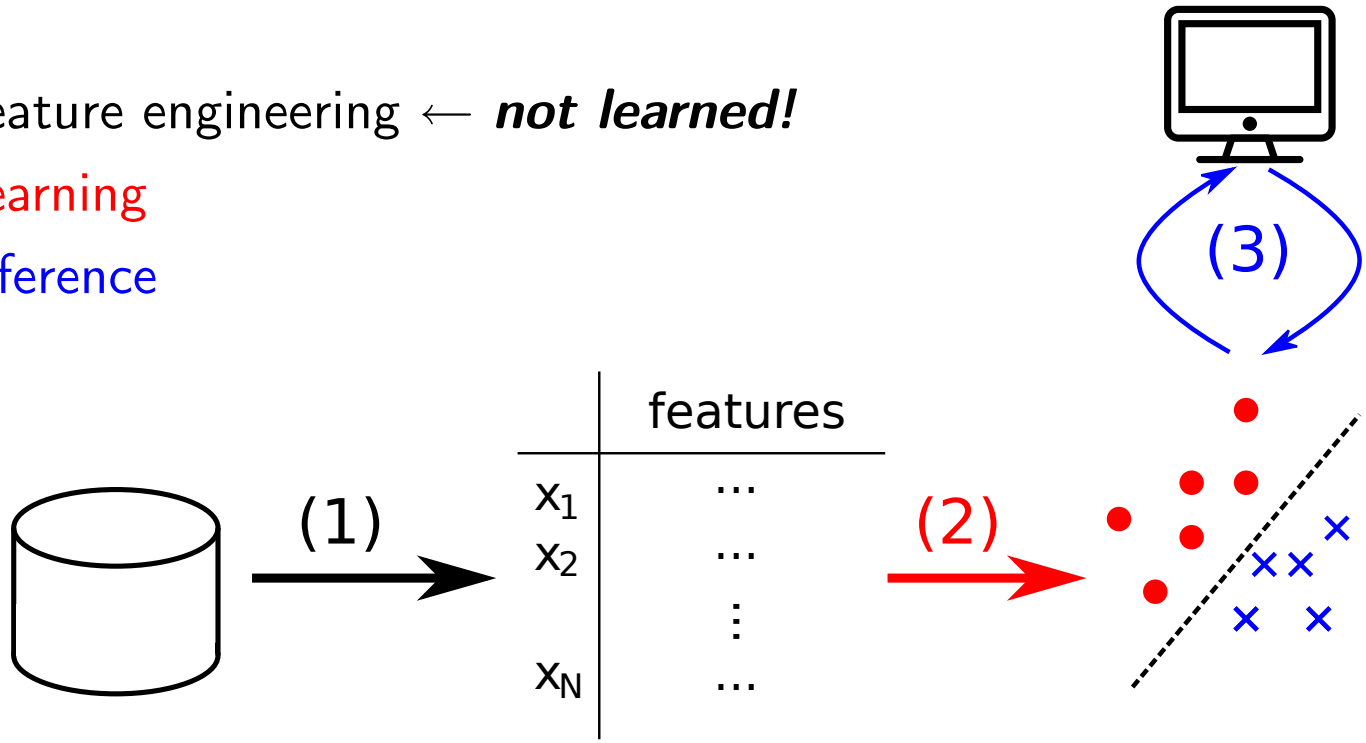




How do you tell deep learning from *not-so-deep* machine learning?

Not-So-Deep Machine Learning

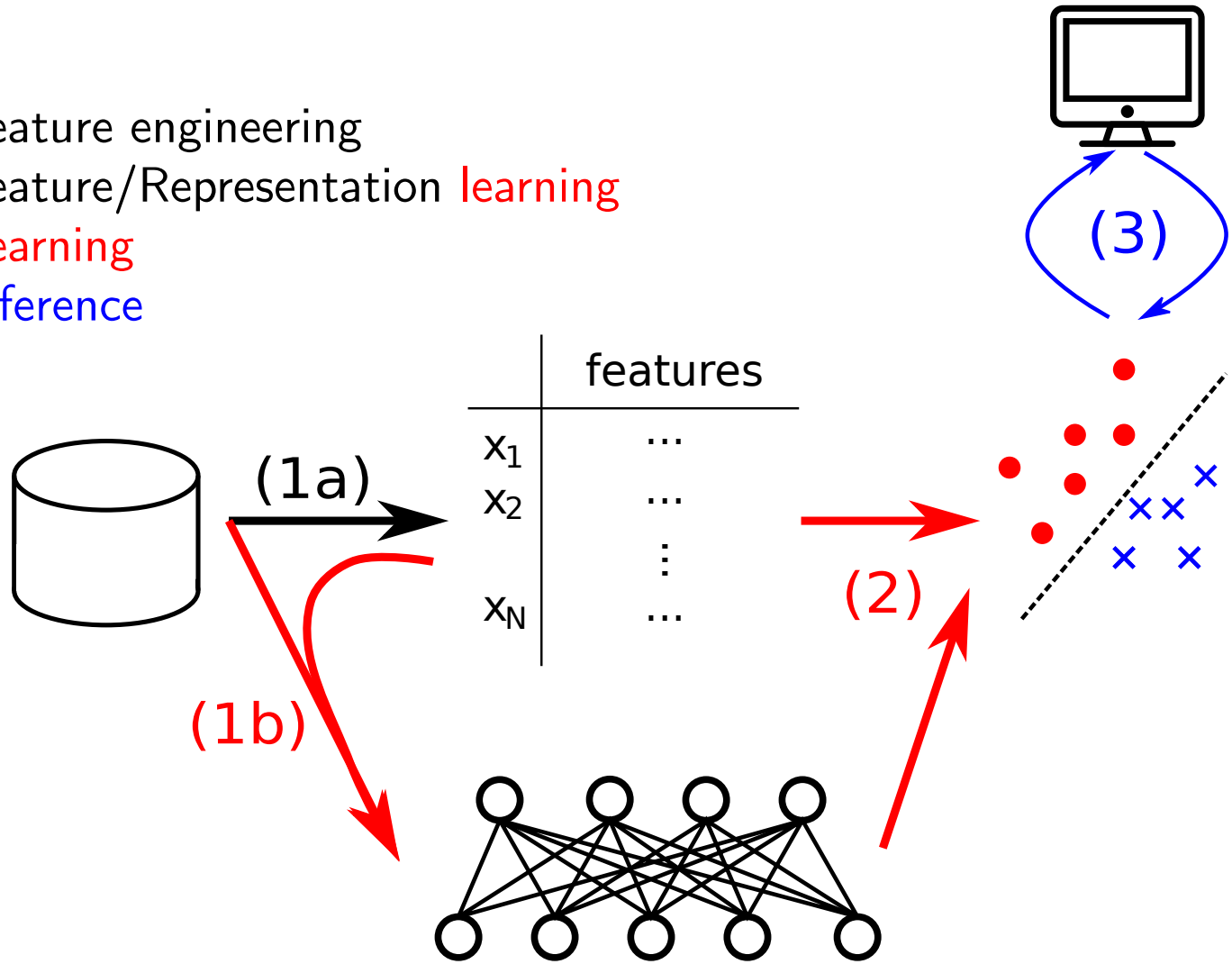
1. Feature engineering ← ***not learned!***
2. Learning
3. Inference



Separation between domain knowledge and general machine learning

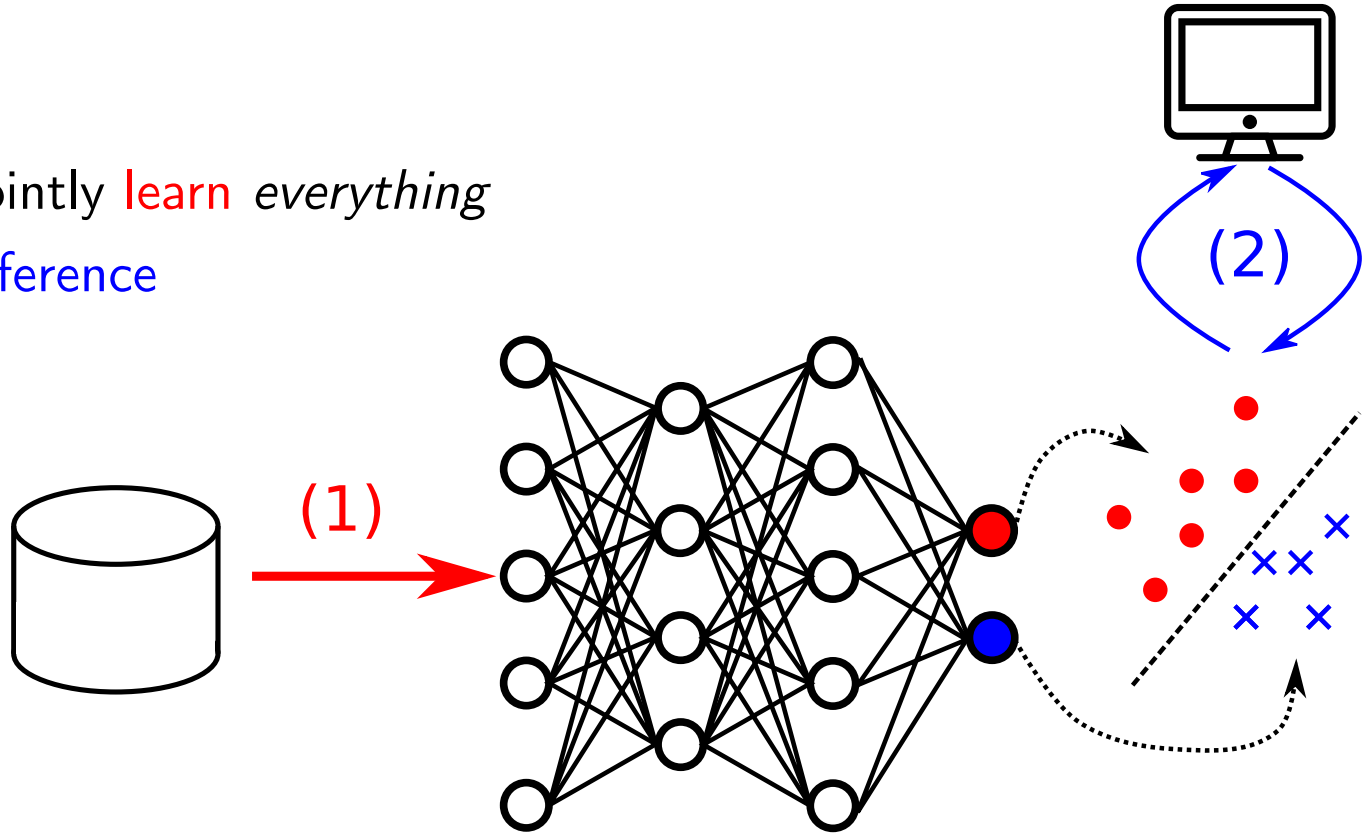
Unsupervised Learning of Representation

1a. Feature engineering
1b. Feature/Representation learning
2. Learning
3. Inference



Deep Learning: toward the *Ultimate* Machine Learning?

1. Jointly learn *everything*
2. Inference



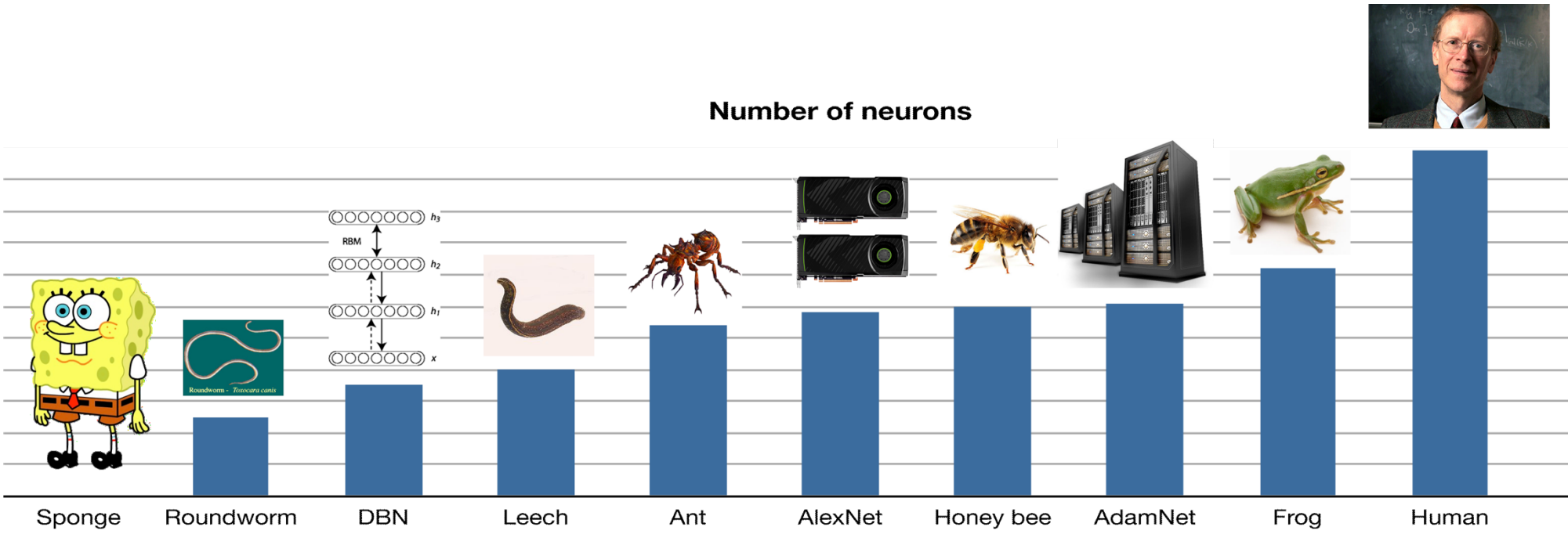
The data decides –Yoshua Bengio

Why now? Why not 20 years ago?

What has happened in last 20+ years?

- We have connected the dots, e.g.,
  - PCA $\Leftrightarrow$ Neural PCA $\Leftrightarrow$ Probabilistic PCA $\Leftrightarrow$ Autoencoder
  - Autoencoder $\Leftrightarrow$ Belief network $\Leftrightarrow$ Restricted Boltzmann machine
- We understand learning better
  - Model structure matters *a lot*
  - Learning *is* but *is not* optimization
  - No need to be scared of non-convex optimization
- We understand how learning and inference interact
- Exponential growth of the amount of data and computational power

And Beyond...



(Goodfellow, 2013)

Today's Tutorial: Introduction to Deep Learning

1. Deep Learning: Past, Present and Future
2. Supervised Neural Networks
   - Multilayer Perceptron and Learning
   - Regularization
   - Practical Recipe
   - Task-specific Neural Network
3. Unsupervised Neural Networks
   - Unsupervised Learning
   - Generative Modeling with Neural Networks
     - Density/Distribution Learning
     - Learning to Infer
   - Semi-Supervised Learning and Pretraining
4. Advanced Topics
   - Beyond Computer Vision
   - Advanced Topics

Supervised Neural Networks

Kyunghyun Cho

Laboratoire d'Informatique des Systèmes Adaptatifs,
Département d'informatique et de recherche opérationnelle,
Facult des arts et des sciences,
Université de Montréal
cho.k.hyun@gmail.com, kyunghyun.cho@umontreal.ca

Warning!!!

The next 9–10 slides may be extremely boring.

Supervised Learning: Rough Picture

Data:

$$D = \{(x_1, y_1), (x_2, y_2) \ldots, (x_N, y_N)\}$$

Assumption:

$$y_n = f^*(x_n)$$

Find a function f, using D, that emulates f^* as well as possible on samples potentially *not* included in D.

Supervised Learning: Probabilistic Picture

Underlying distributions:

$$X \text{ and } Y \mid X$$

Data:

$$D = \{(x, y) \mid x \sim X \ \& \ y \sim Y \mid X = x\}$$

Find a distribution $p(y \mid x)$, using D, that emulates $Y \mid X$ as well as possible on new samples from $p(x)$ potentially *not* included in D.

Supervised Learning: Evaluation and Generalization

Evaluation:

$$\mathbb{E}_{x\sim p(x)}\left[\|f(x)-f^*(x)\|^p\right] \approx \sum_{x\in D_{\textbf{test}}} \left[\|f(x)-f^*(x)\|^p\right]$$

or

$$\mathbb{E}_{x\sim p(x)}\left[\mathrm{KL}(p(y\mid x)\|p^*(y\mid x))\right] \approx \sum_{x\in D_{\textbf{test}}} \mathrm{KL}(p\|p^*),$$

where $D \neq D_{\text{test}}$

Why do we test the found solution f^* or p^* on D_{test}, not on D?

Linear/Ridge Regression

Underlying assumption:

$$y = W^* x + b^* + \epsilon,$$

where ϵ is a white Gaussian noise.

Data:

$$D = \{(x_1, y_1), (x_2, y_2) \ldots, (x_N, y_N)\},$$

where $y_n = W^* x_n + b^* + \epsilon$.

Learning:

$$W, b = \arg\min_{W,b} \frac{1}{N} \sum_{n=1}^{N} \|Wx_n + b - y_n\|_2^2 + \lambda \left(\|W\|_F^2 + \|b\|_2^2 \right)$$

Multilayer Perceptron: (Binary) Classification

Underlying assumption:

$$y \sim \mathcal{B}\left(p = f_{\boldsymbol{\theta}^*}(x)\right),$$

where $f_{\boldsymbol{\theta}^*}$ is a nonlinear function parameterized with $\boldsymbol{\theta}^*$ and $\mathcal{B}(p)$ is a Bernoulli distribution of mean p.

Data:

$$D = \{(x_1, y_1), (x_2, y_2) \ldots, (x_N, y_N)\},$$

where $y_n \sim \mathcal{B}\left(p = f_{\boldsymbol{\theta}^*}(x_n)\right)$.

Learning:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} y_n \log f_{\boldsymbol{\theta}}(x_n) + (1 - y_n) \log\left(1 - f_{\boldsymbol{\theta}}(x_n)\right) + \lambda \Omega(\boldsymbol{\theta}, D)$$

Learning as an Optimization

Ultimately, learning is (*mostly*)

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} c\left((x_n, y_n) \mid \boldsymbol{\theta}\right) + \lambda \Omega\left(\boldsymbol{\theta}, D\right),$$

where $c\left((x, y) \mid \boldsymbol{\theta}\right)$ is a per-sample cost function.

Gradient Descent

Gradient-descent Algorithm:

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \eta \nabla L(\boldsymbol{\theta}^{t-1})$$

where, in our case,

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} l\left((x_n, y_n) \mid \boldsymbol{\theta}\right).$$

Let us assume that $\Omega\left(\boldsymbol{\theta}, D\right) = 0$.

Stochastic Gradient Descent

Often, it is too costly to compute $C(\boldsymbol{\theta})$ due to a large training set.

Stochastic gradient descent algorithm:

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \eta^t \nabla l\left((x', y') \mid \boldsymbol{\theta}^{t-1}\right),$$

where (x', y') is a randomly chosen sample from D, and

$$\sum_{t=1}^{\infty} \eta^t \to \infty \text{ and } \sum_{t=1}^{\infty} \left(\eta^t\right)^2 < \infty.$$

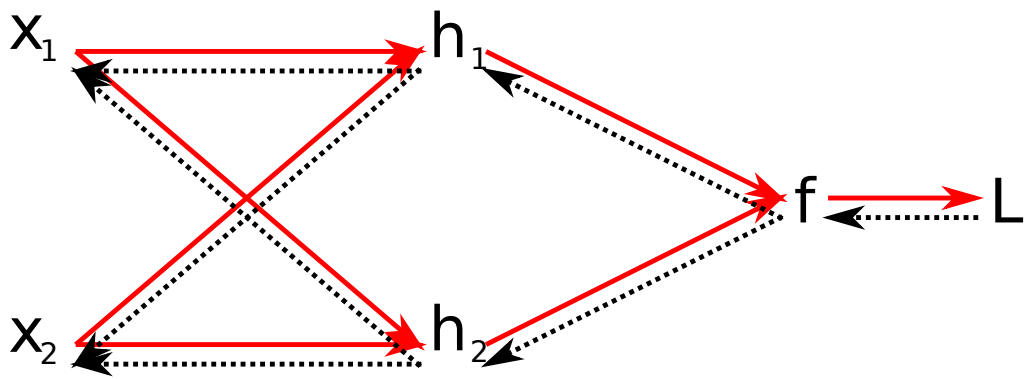
Let us assume that $\Omega\left(\boldsymbol{\theta}, D\right) = 0$.

Any question so far?

Almost there...

How do we compute the gradient efficiently for *deep* neural networks?

Backpropagation Algorithm – (1) Forward Pass



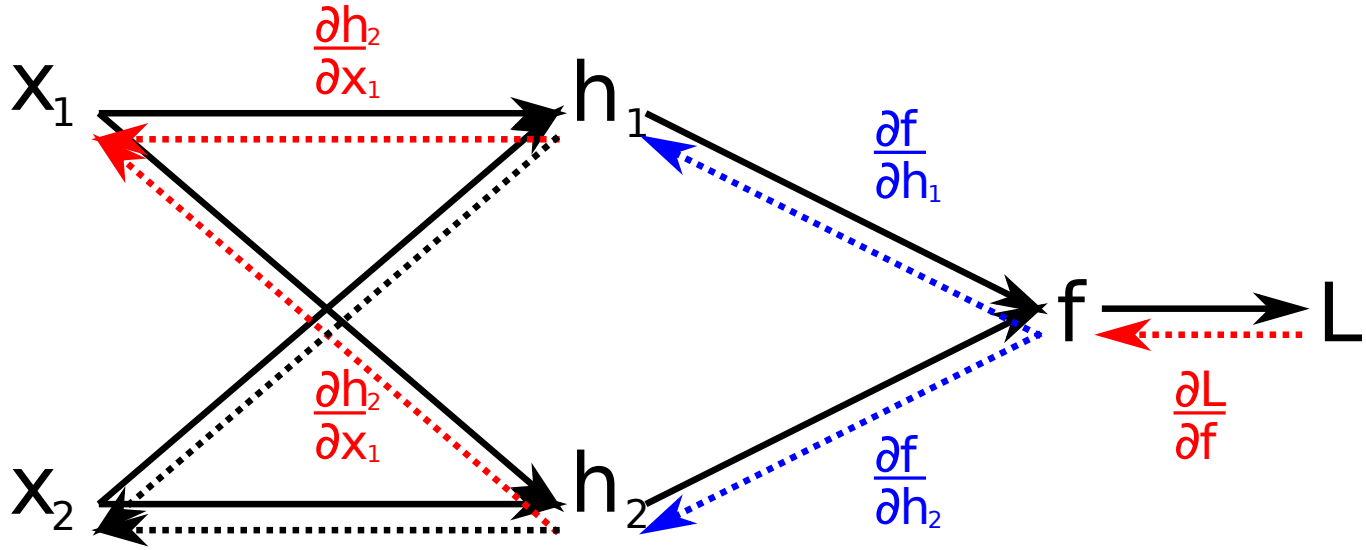
Forward Computation:

$$L(f(h_1(x_1, x_2, \boldsymbol{\theta}_{h_1}), h_2(x_1, x_2, \boldsymbol{\theta}_{h_2}), \boldsymbol{\theta}_f), y)$$

Multilayer Perceptron with a single hidden layer:

$$L(\mathbf{x}, y, \boldsymbol{\theta}) = \frac{1}{2}\left(y - \mathbf{U}^\top \phi\left(\mathbf{W}^\top \mathbf{x}\right)\right)^2$$

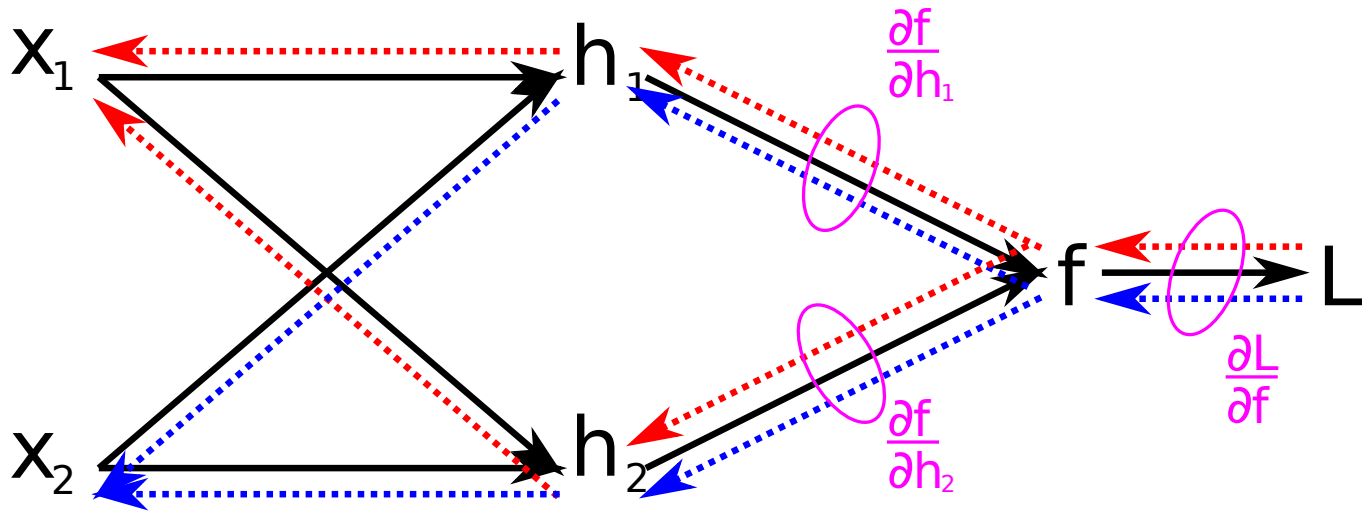
Backpropagation Algorithm – (2) Chain Rule



Chain rule of derivatives:

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial x_1} = \frac{\partial L}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial x_1} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial x_1}\right)$$

Backpropagation Algorithm – (3) Shared Derivatives

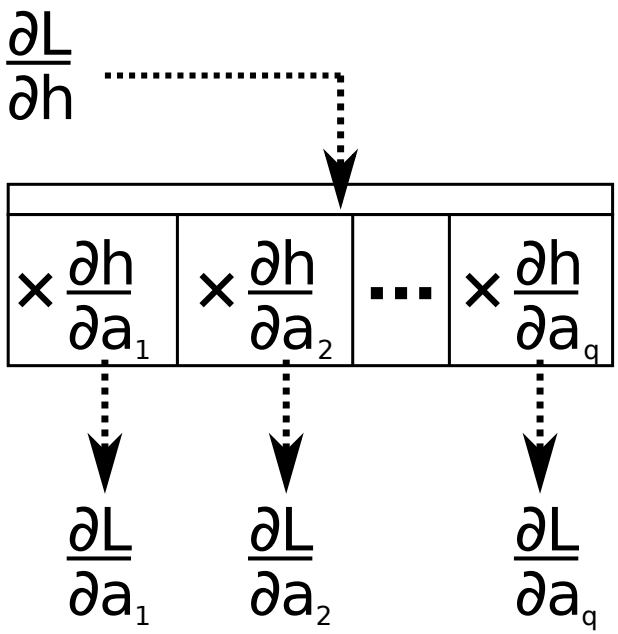


Local derivatives are *shared*:

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial x_1} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial x_1}\right)$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial x_2} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial x_2}\right)$$

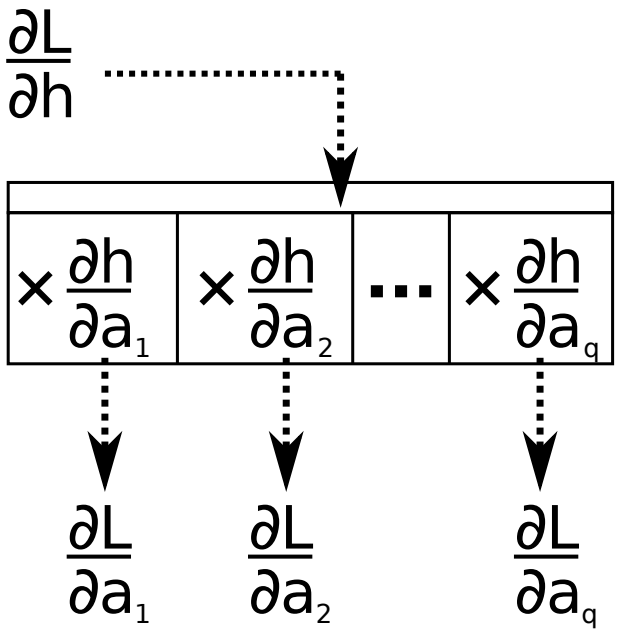
Backpropagation Algorithm – (4) Local Computation



Each node computes

- Forward: $h(a_1, a_2, \ldots, a_q)$
- Backward: $\frac{\partial h}{\partial a_1}, \frac{\partial h}{\partial a_2}, \ldots, \frac{\partial h}{\partial a_q}$

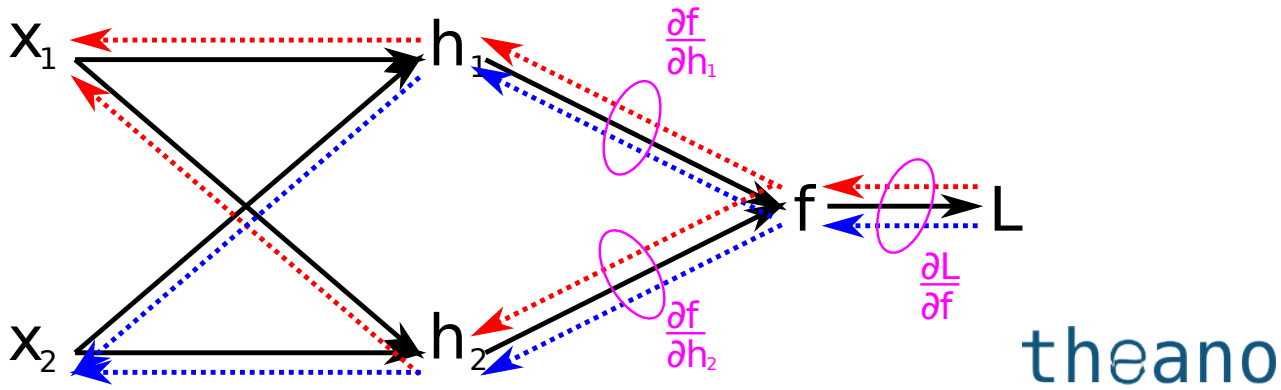
Backpropagation Algorithm – Requirements



- Each node computes a *differentiable* function[1]
- Directed Acyclic Graph[2]

[1] Well. . . ?
[2] Well. . . ?

Backpropagation Algorithm – Automatic Differentiation



- Generalized approach to computing partial derivatives
- As long as your neural network fits the requirements, you do *not* need to derive the derivatives yourself!
  - Theano, Torch, ...

Any question on backpropagation and automatic differentiation?

Regularization – (1) Maximum a Posteriori

Probabilistic Perspective: Find a model $\mathcal{M}$ by...

- Maximum Likelihood (ML): $\arg\max_{\boldsymbol{\theta}} p(D \mid \mathcal{M})$
- Maximum a Posteriori (MAP): $\arg\max_{\boldsymbol{\theta}} p(\mathcal{M} \mid D)$

What is the probability of $\boldsymbol{\theta}$ given a current data D?

$$p(\mathcal{M} \mid D) = \frac{p(D \mid \mathcal{M})p(\mathcal{M})}{\sum_{\mathcal{M}} p(D, \mathcal{M})} \propto p(D \mid \mathcal{M})p(\mathcal{M})$$

$P(\mathcal{M})$: *What do we think a good model should be?*

Regularization – (2) Weight Decay

$P(\mathcal{M})$: *What do we think a good model should be?*

Weight-Decay Regularization

- Prior Distribution: $\theta_j \sim \mathcal{N}\left(0, (M\lambda)^{-1}\right)$

Maximum a Posteriori Estimation

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} p(y^n \mid x^n, \boldsymbol{\theta}) + \lambda \sum_{j=1}^{M} \theta_j^2.$$

Regularization – (3) Smoothness and Noise Injection

Prior on a Model: Smoothness

- $f(x) \approx f(x+\epsilon)$: the model should be insensitive to small change/noise $\iff$ Minimize $\sum_{n=1}^{N} \left| \frac{\partial f(x^n)}{\partial x} \right|^2$

Regularizing $\sum_{n=1}^{N} \left| \frac{\partial f(x^n)}{\partial x} \right|^2$ is *equivalent* to adding random Gaussian noise in the input (Bishop, 1995)

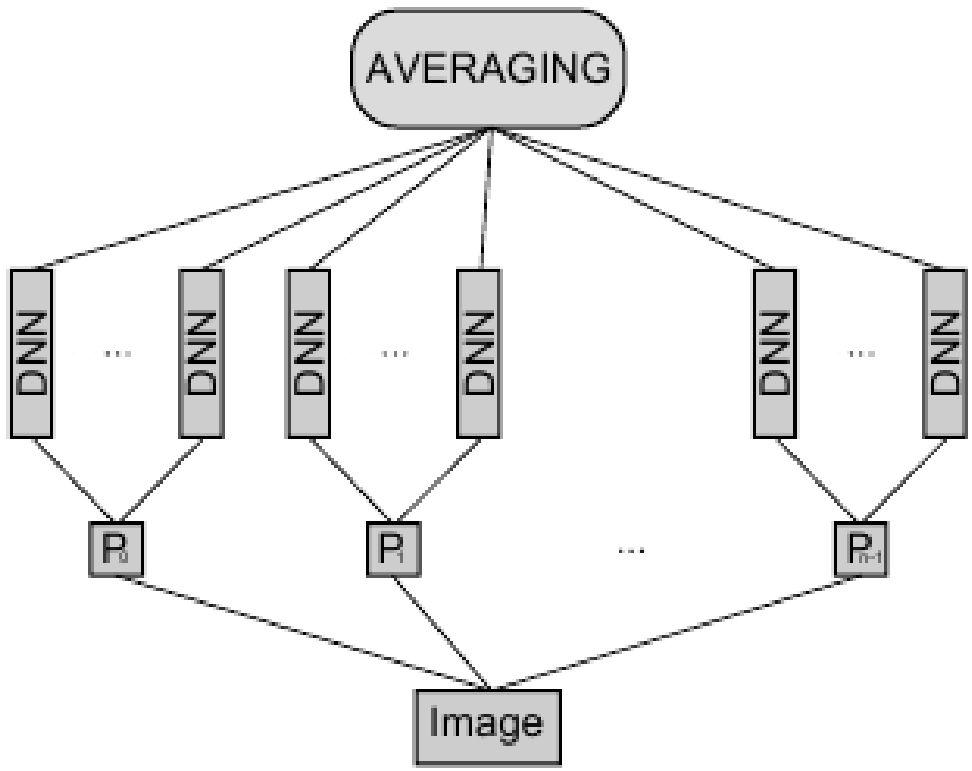
$$\arg\min_{\theta} \sum_{n=1}^{N} \| f_\theta(x^n) - y^n \|^2 + \lambda \left| \frac{\partial f(x^n)}{\partial x} \right|^2$$

$$\approx \sum_{\epsilon} p(\epsilon) \left(\arg\min_{\theta} \sum_{n=1}^{N} \| f_\theta(x^n + \epsilon) - y^n \|^2 \right)$$

Regularization – (4a) Ensemble Learning and Dropout

Wisdom of the Crowd: *Train M classifiers and let them vote*

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_{\mathcal{M}_m}(x)$$

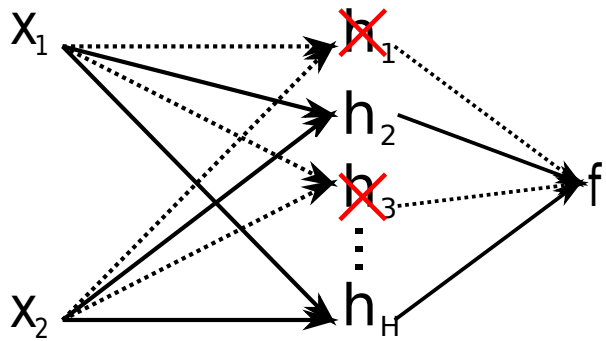


(Ciresan et al., 2012)

Regularization – (4b) Ensemble Learning and Dropout

Dropout: Train one, but exponentially many classifiers (Hinton et al., 2012)

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \log \mathbb{E}_{\mathbf{m}}\left[p(y^n, \mathbf{m} \mid x^n, \theta)\right] \geq \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\mathbf{m}}\left[\log p(y^n, \mathbf{m} \mid x^n, \theta)\right]$$



Each update samples one network out of exponentially many classifiers.

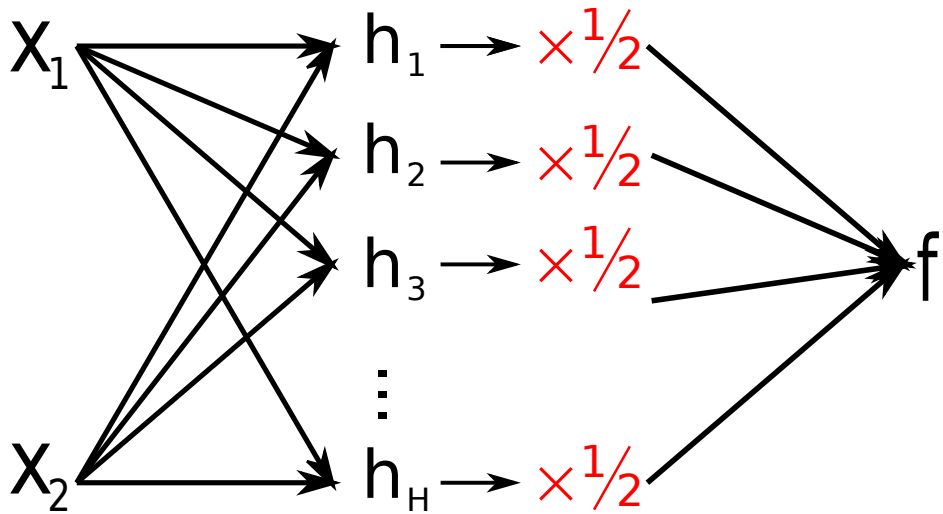
$$\theta^t = \theta^{t-1} - \eta^t \nabla l\left((x', y'), \mathbf{m} \mid \theta^{t-1}\right),$$

where $m_{i,l} \sim \mathcal{B}(0.5)$.

Regularization – (4b) Ensemble Learning and Dropout

Dropout: When testing, halve the activations

$$\mathcal{L}(\theta) \geq \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\mathbf{m}}\left[\log p(y^n, \mathbf{m} \mid x^n, \theta)\right] \approx \frac{1}{N}\sum_{n=1}^{N} p\left(y^n, \mathbf{m} = \frac{1}{2} \mid x^n, \theta\right)$$



Do you see why I spent so much time on regularization?

Common Recipe for Deep Neural Networks

1. Use a piecewise linear hidden unit
   - Rectifier: $h(x) = \max\{0, x\}$ (Glorot&Bengio, 2011)
   - Maxout: $h(\{x_1, \ldots, x_p\}) = \max\{x_1, \ldots, x_p\}$ (Goodfellow et al., 2013)
2. Preprocess data and choose features carefully
   - Images: Whitening? Local contrast normalization? Raw? SIFT? HoG?
   - Speech: Raw? Spectrum?
   - Text: Characters? Words? Tree?
   - General: z-Normalization?
3. Use Dropout and other regularization methods
4. Unsupervised Pretraining (Hinton&Salakhutdinov, 2006)
   - Few labeled samples, but a lot of unlabeled samples
5. Carefully search for hyperparameters
   - Random search, Bayesian optimization
     (Bergstra&Bengio, 2013;Bergstra et al., 2011;Snoek et al.,2012)
6. Often, deeper the better
7. Build an ensemble of neural networks

But, nobody seems to use a vanilla multilayer perceptron, right?

How to Encode Prior/Domain Knowledge?

Data Preprocessing and Feature Extraction:

- Object recognition from images
  - Lighting condition shouldn't matter $\longrightarrow$ Contrast normalization
- Gesture recognition from skeleton
  - Relative positions of joints are important $\longrightarrow$ Relative coordinate system w.r.t. the body center
- Language Processing
  - Word counts are important $\longrightarrow$ Bag-of-Words representation

Model Architecture Design

Convolutional Neural Networks – (1)

Suitable for Images, Videos and Speech

Prior/Domain Knowledge

- Translation invariance
- Rotation invariance: Images and Videos
- Temporal invariance: Videos and Speech
- Frequency invariance: Speech

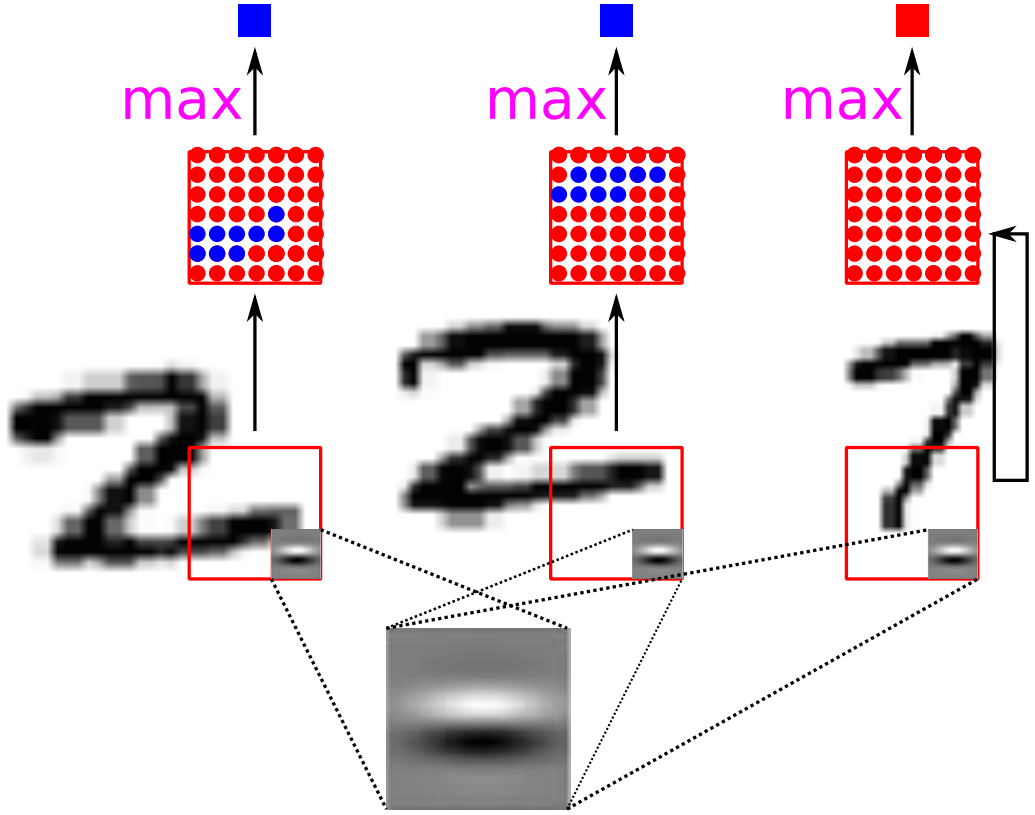
Convolutional Neural Networks – (2) Convolution and Pooling

Convolution

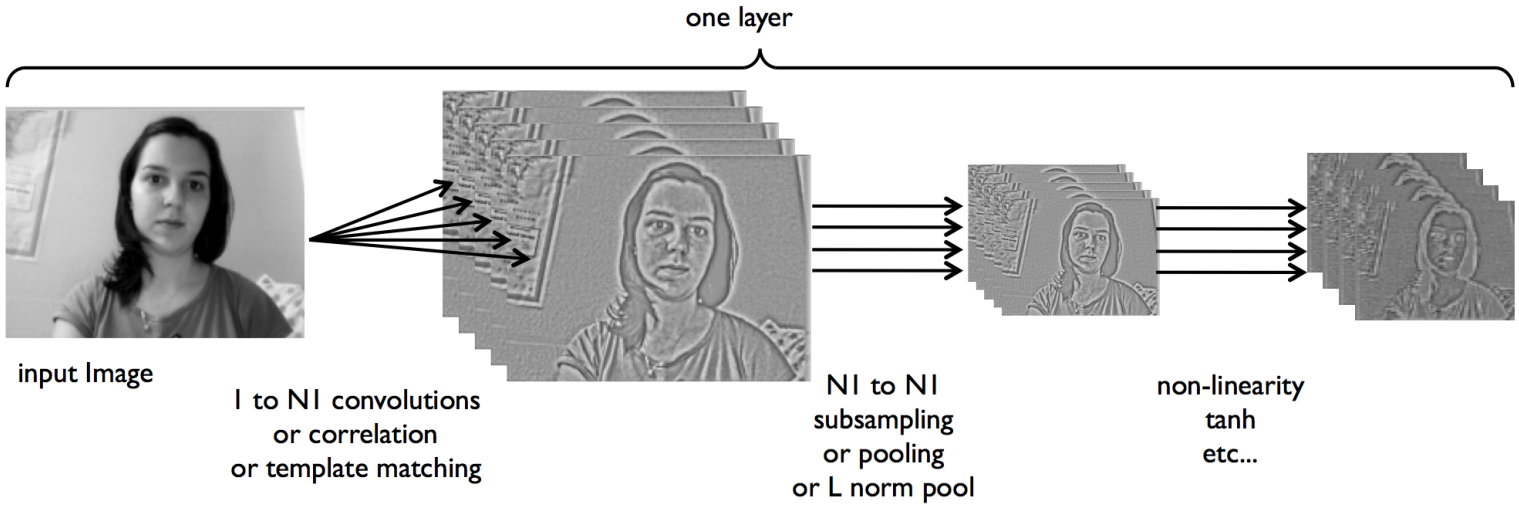
- Global

Pooling

- Local



Convolutional Neural Networks – (3)
Convolutional Neural Network



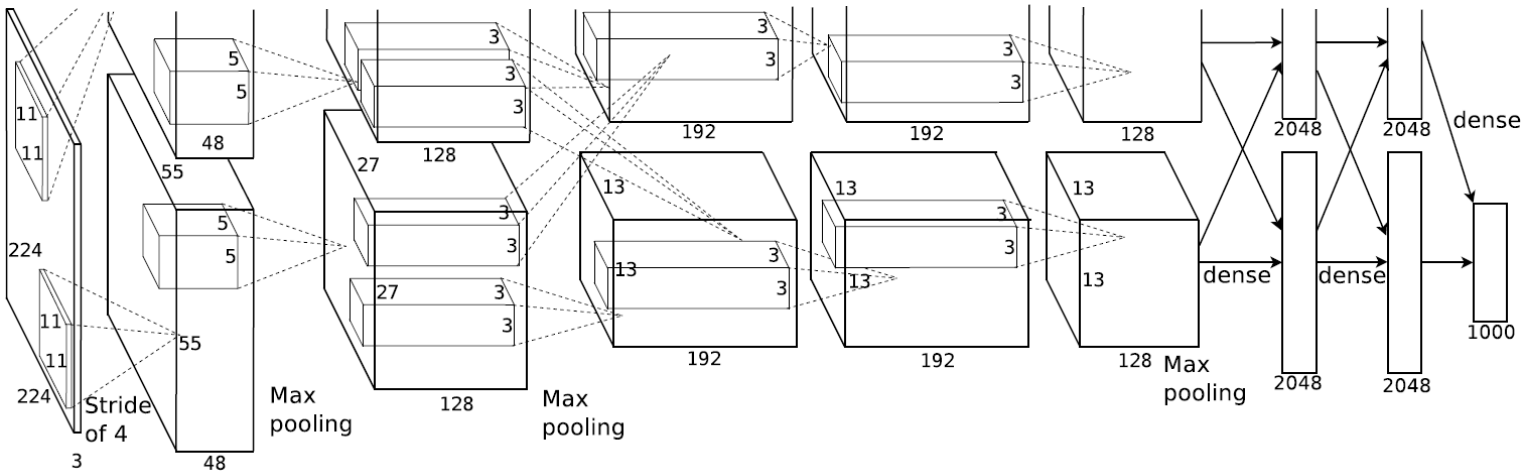
(TeraDeep, 2013)

Convolutional Layer

1. Contrast Normalization
2. Convolution
3. Pooling
4. Nonlinearity

Convolutional Neural Networks – (3)

Deep Convolutional Neural Network



(Krizhevsky et al., 2012)

Recursive Neural Networks – (1)

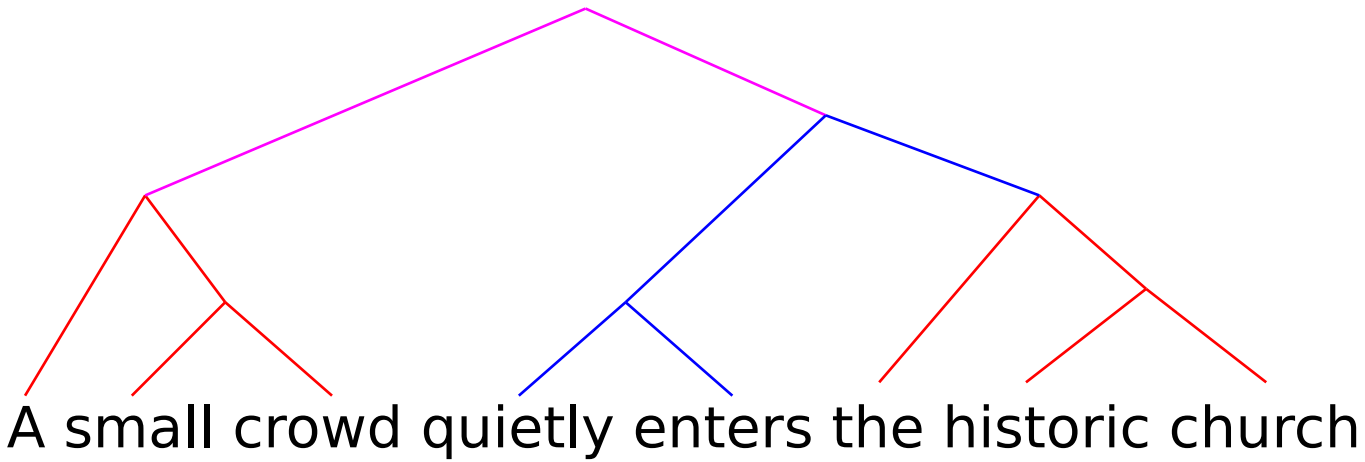
Suitable for Text and Variable-Length Sequences

Prior/Domain Knowledge

- Compositionality $\approx$ Tree-based Grammar (?)
- Location invariance
- Variable Length

Recursive Neural Networks – (2)

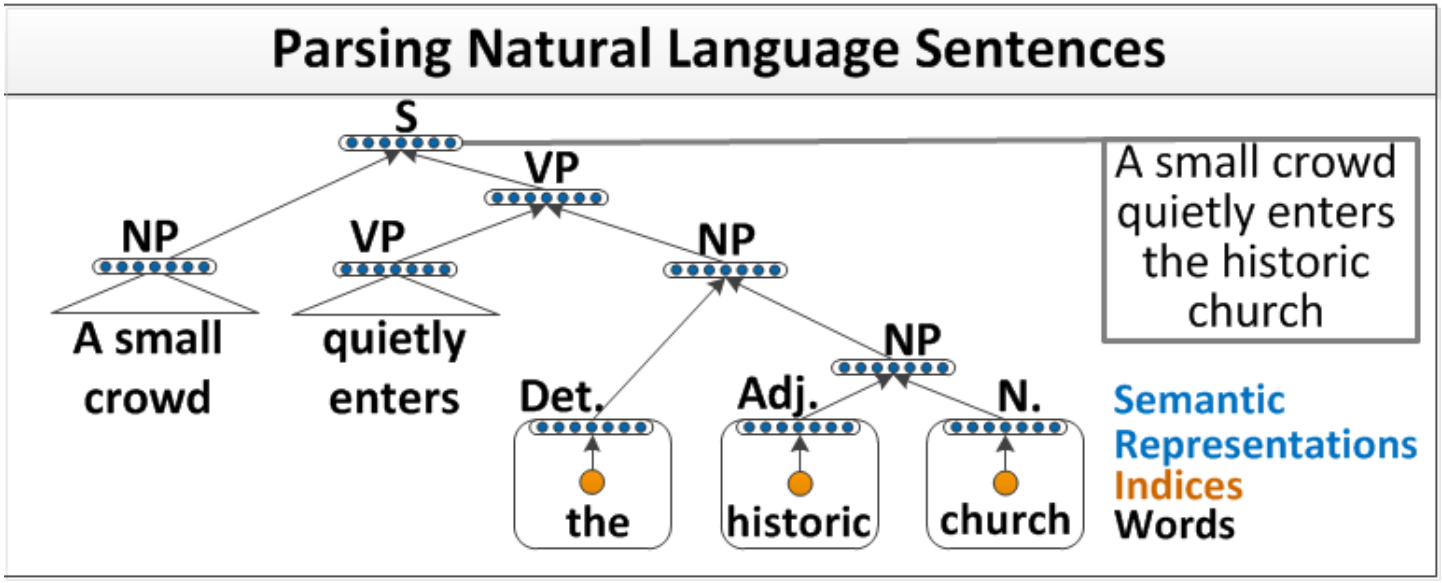
Compositional Structure



Small (local) pieces are glued together to form a global structure

Recursive Neural Networks – (3)

Finding a good, compact representation of variable-length sequence



(Socher et al., 2011)

What other architectures can you think of?

Further Topics

- Is learning solved for supervised neural networks?
- Recurrent neural networks: cope with variable-length inputs/outputs
- Beyond sigmoid and rectifier functions

Unsupervised Neural Networks

Kyunghyun Cho

Laboratoire d'Informatique des Systèmes Adaptatifs,
Département d'informatique et de recherche opérationnelle,
Facult des arts et des sciences,
Université de Montréal
cho.k.hyun@gmail.com (chokyun@iro.umontreal.ca)

Warning!!!

The first half of this session can be boring.

Unsupervised Learning

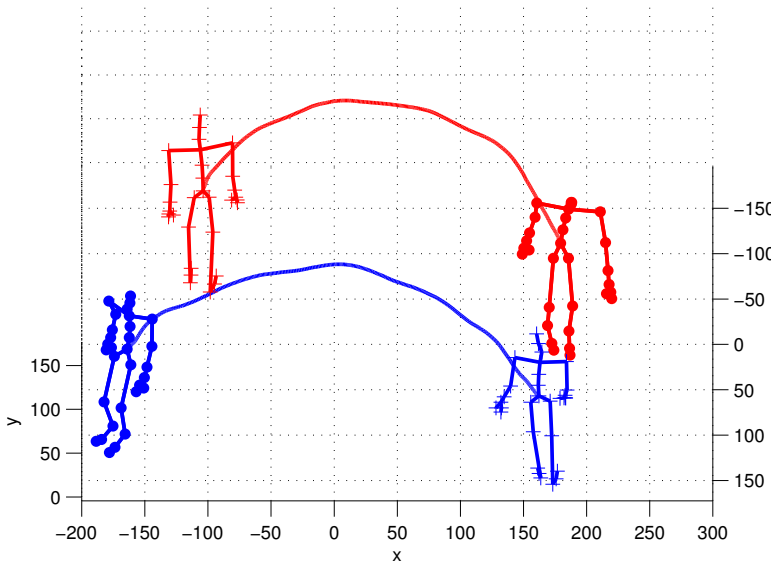
No more label!

$$D = \{x_1, x_2, \ldots, x_N\}$$

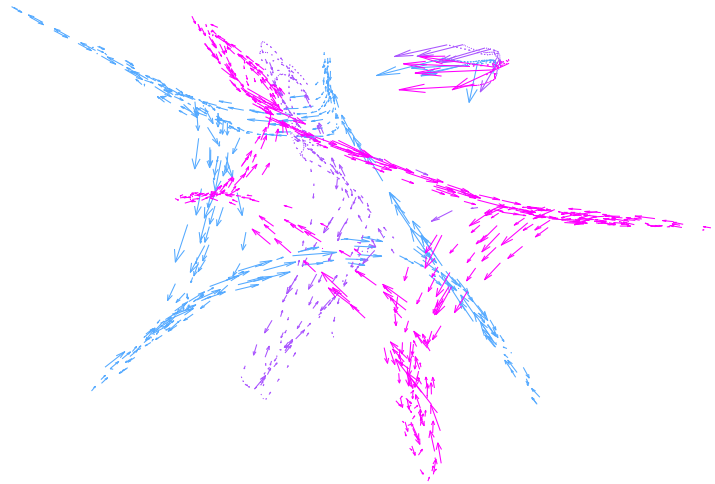
What can we do?

(Exploratory) Data Analysis

The most important step in machine learning



Human Gestures

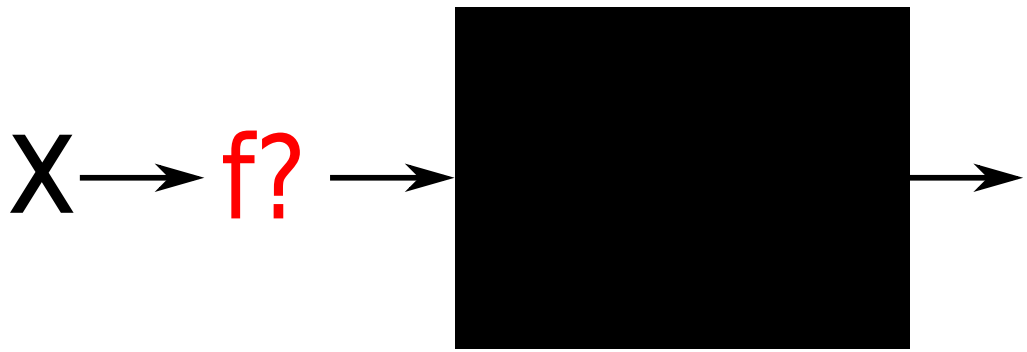


Visualization by DNN

(Cho&Chen, 2014)

Feature Extraction

With domain knowledge $\longrightarrow$ Engineered Features
Without domain knowledge $\longrightarrow$ *Learned* Features



Generative Model: Probabilistic Picture

Underlying distribution:

$$X \sim P_D$$

Data:

$$D = \{x \mid x \sim P_D\}$$

Find a distribution $p(x)$, using D, that emulates P_D as well as possible on new samples from $p(x)$ potentially *not* included in D.

What should we do with data $D = \{\mathbf{x}_n\}_{n=1}^{N}$

Example target tasks

- Classification $p(\mathbf{x}_{\text{class}} \mid \mathbf{x}_{\text{ins}})$
- Missing value reconstruction $p(\mathbf{x}_m \mid \mathbf{x}_o)$
- Denoising $p(\mathbf{x} \mid \tilde{\mathbf{x}})$
- Structured Output Prediction $p(\mathbf{x}_{\text{out}} \mid \mathbf{x}_{\text{in}})$
- Outlier Detection $p(\mathbf{x})? > \tau$

Ultimately, it comes down to learning a distribution of $\mathbf{x}$.

Density/Distribution Estimation – (1)

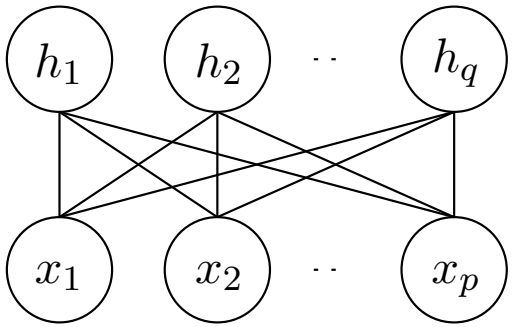
Latent Variable Models $p_{\theta}(x)$

$$\theta^* = \arg\max_{\theta} \frac{1}{N} \sum_{n=1} \log \sum_{h} p(x^n \mid h) p(h)$$

1. Define a parametric form of joint distribution $p_{\theta}(x, h)$
2. Derive a learning rule for θ

Density/Distribution Estimation – (2) Restricted Boltzmann Machines

1. Joint distribution



$$p_\theta(x,h) = \frac{1}{Z(\theta)} \exp\left(\sum_{i=1}^{p}\sum_{j=1}^{q} x_i h_j w_{ij}\right)$$

2. Marginal distribution

$$\sum_h p_\theta(x,h) = \frac{1}{Z(\theta)} \prod_{j=1}^{q}\left(1+\exp\left(\sum_{i=1}^{p} w_{i,j}x_i\right)\right)$$

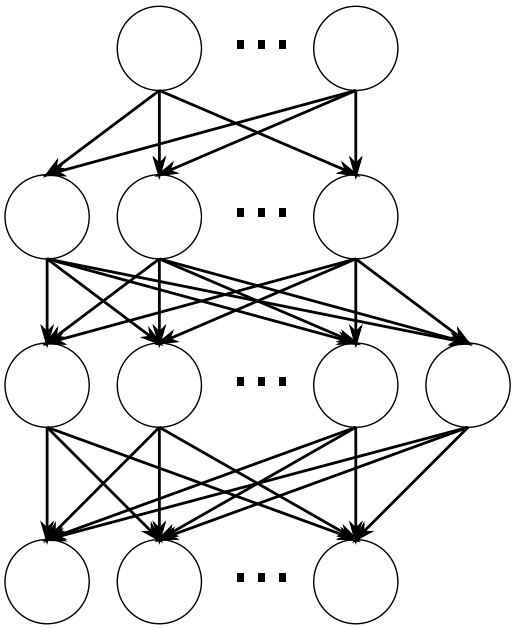
3. Learning rule: Maximum Likelihood

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim P_d}\left[\log\prod_{j=1}^{q}\left(1+\exp\left(\sum_{i=1}^{p} w_{i,j}x_i\right)\right) - \log Z(\theta)\right]$$

$$\nabla_{w_{ij}} = \langle x_i h_j\rangle_d - \langle x_i h_j\rangle_m$$

(Smolensky, 1986)

Density/Distribution Estimation – (3) Belief Networks



1. Joint distribution

$$p_\theta(x, h) = p(x \mid h_1)p(h_1 \mid h_2) \cdots p(h)$$

2. Marginal distribution

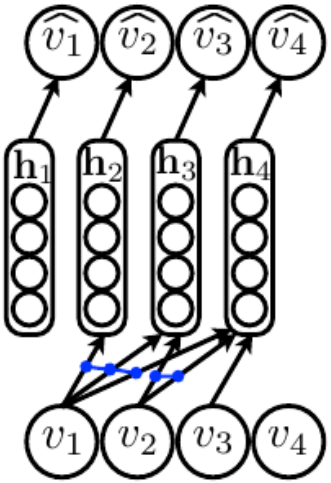
$$\sum_h p_\theta(x, h) = ?$$

3. Learning rule: Maximum Likelihood

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim P_d} \left[\sum_h p_\theta(x, h) \right]$$

(Neal, 1996)

Density/Distribution Estimation – (4) NADE



1. Joint distribution

$$p_\theta(x) = p_\theta(x_1)p_\theta(x_2 \mid x_1) \cdot p_\theta(x_d \mid x_1, \ldots, x_{d-1})$$

2. ~~Marginal distribution~~: *no* latent variable h

3. Learning rule:
   Maximum Likelihood (*fixed order*), Order-agnostic (*all orders*)

(Larochelle&Murray, 2011)

Density/Distribution Estimation – (4) Issues

Intractability! Intractability! Intractability!

(General) Boltzmann Machines

- Normalization Constant $Z(\theta)$
- Marginal Probability $\sum_{\boldsymbol{h}} p(x, h)$
- Posterior Probability $p(h \mid x)$
- Conditional Probability $p(x_{\mathbf{mis}} \mid x_{\mathbf{obs}})$

Belief Networks

- ~~Normalization Constant $Z(\theta)$~~
- Marginal Probability $\sum_{\boldsymbol{h}} p(x, h)$
- Posterior Probability $p(h \mid x)$
- Conditional Probability $p(x_{\mathbf{mis}} \mid x_{\mathbf{obs}})$

Restricted Boltzmann Machines

- Normalization Constant $Z(\theta)$
- ~~Marginal Probability $\sum_{\boldsymbol{h}} p(x, h)$~~
- ~~Posterior Probability $p(h \mid x)$~~
- Conditional Probability $p(x_{\mathbf{mis}} \mid x_{\mathbf{obs}})$

NADE

- ~~Normalization Constant $Z(\theta)$~~
- ~~Marginal Probability $\sum_{\boldsymbol{h}} p(x, h)$~~
- ~~Posterior Probability $p(h \mid x)$~~
- Conditional Probability $p(x_{\mathbf{mis}} \mid x_{\mathbf{obs}})$
- Somewhat unsatisfactory performance

Do we want to learn the distribution?

Generative Model – (1) Learn to Infer

Example target tasks

- Classification $p(\mathbf{x}_{\text{class}} \mid \mathbf{x}_{\text{ins}})$
- Missing value reconstruction $p(\mathbf{x}_m \mid \mathbf{x}_o)$
- Denoising $p(\mathbf{x} \mid \tilde{\mathbf{x}})$
- Structured Output Prediction $p(\mathbf{x}_{\text{out}} \mid \mathbf{x}_{\text{in}})$
- ~~Outlier Detection $p(\mathbf{x})? > \tau$~~

All we want is to infer the conditional distribution of unknown variables

(Goodfellow et al., 2013; Brakel et al., 2013; Stoyanov et al., 2011, Raiko et al., 2014)

Generative Model – (2) Learn to Infer

Approximate Inference in a Graphical Model

$$p(x_{\text{mis}} \mid x_{\text{obs}}) \approx Q(x_{\text{mis}} \mid x_{\text{obs}})$$

Methods:

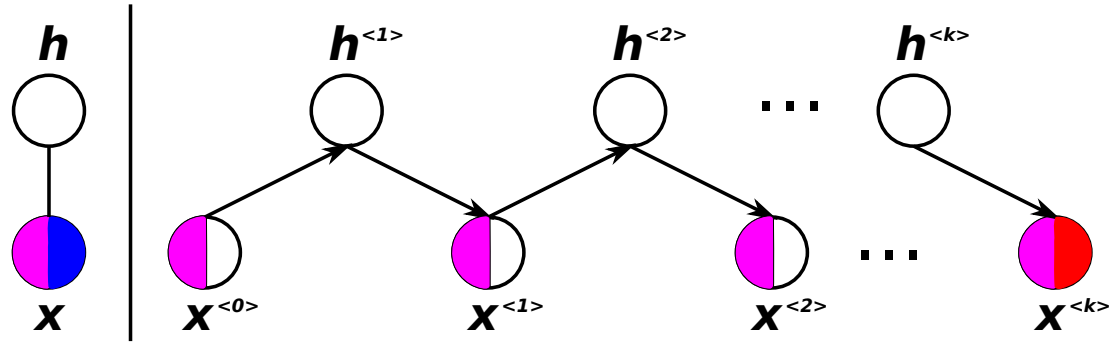
- Loopy belief propagation
- Variational inference/message-passing

At the end of the day...

$$Q^k(x_{\text{mis}} \mid x_{\text{obs}}) = f\left(Q^{k-1}(x_{\text{mis}} \mid x_{\text{obs}})\right)$$

until convergence

Generative Model – (3) Learn to Infer



Approximate Inference in Restricted Boltzmann Machine

$$p(x_{\text{mis}} \mid x_{\text{obs}}) \approx Q(x_{\text{mis}} \mid x_{\text{obs}})$$

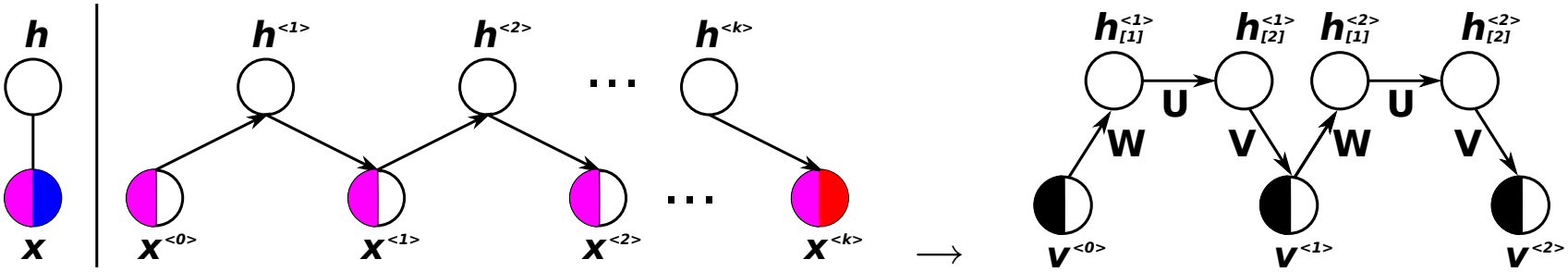
Mean-field Fixed-point Iteration

$$\mu_x^k = \sigma\left(\mathbf{W}\sigma\left(\mathbf{W}^\top \mu_x^{k-1} + \mathbf{c}\right) + \mathbf{b}\right)$$

At the end of the day, a multilayer perceptron with $k-1$ hidden layers.

$\longrightarrow$ *Use backpropagation and stochastic gradient descent!*

Generative Model – (4) Learn to Infer – NADE-k



Further Generalization with *Deep* Neural Networks

$$p(x_{\text{mis}} \mid x_{\text{obs}}) \approx Q(x_{\text{mis}} \mid x_{\text{obs}}) = f_\theta(x_{\text{obs}})$$

Interpret the model as a mixture of NADE's with different orders of variables

- Exact computation of $p(x)$ possible
- Fast inference $p(x_{\text{mis}} \mid x_{\text{obs}})$
- Flexible

(Raiko et al., 2014)

Generative Model – (5) Learn to Infer

Lesson:

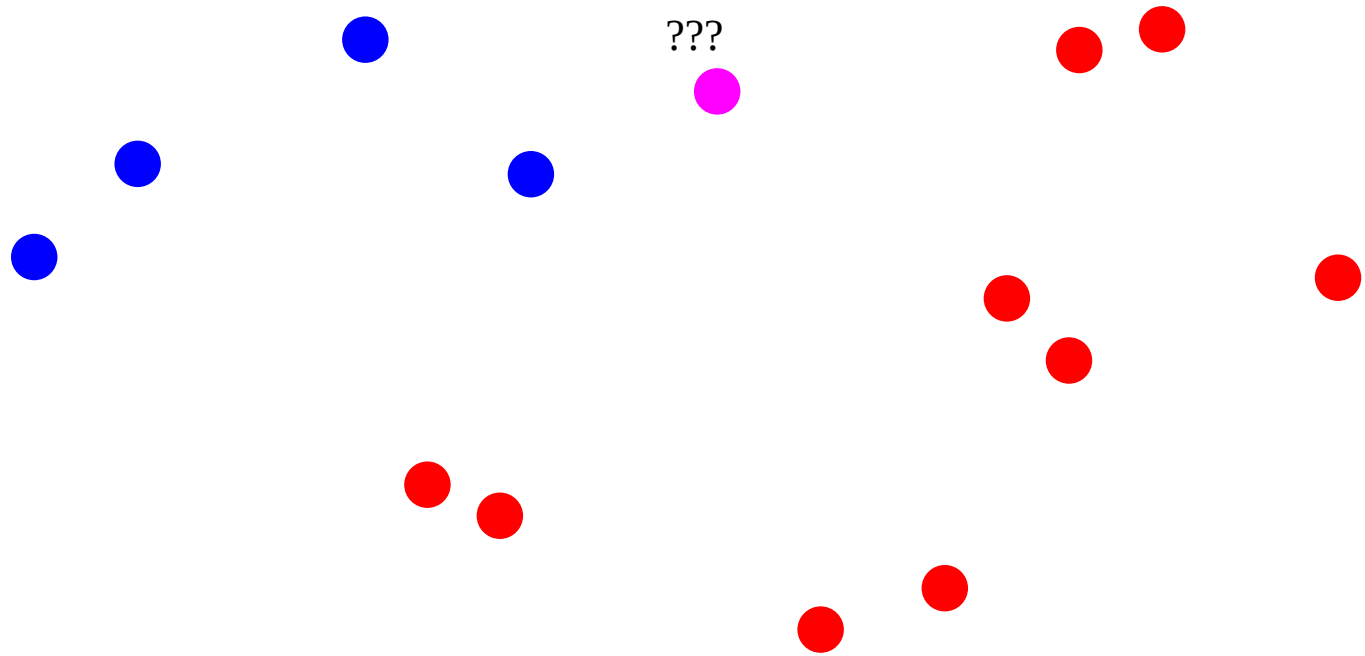
- Do *not* maximize log-likelihood, but *minimize* the actual cost!

$\Longleftrightarrow$

- Don't do what a model tells you to do, but do what you aim to do.

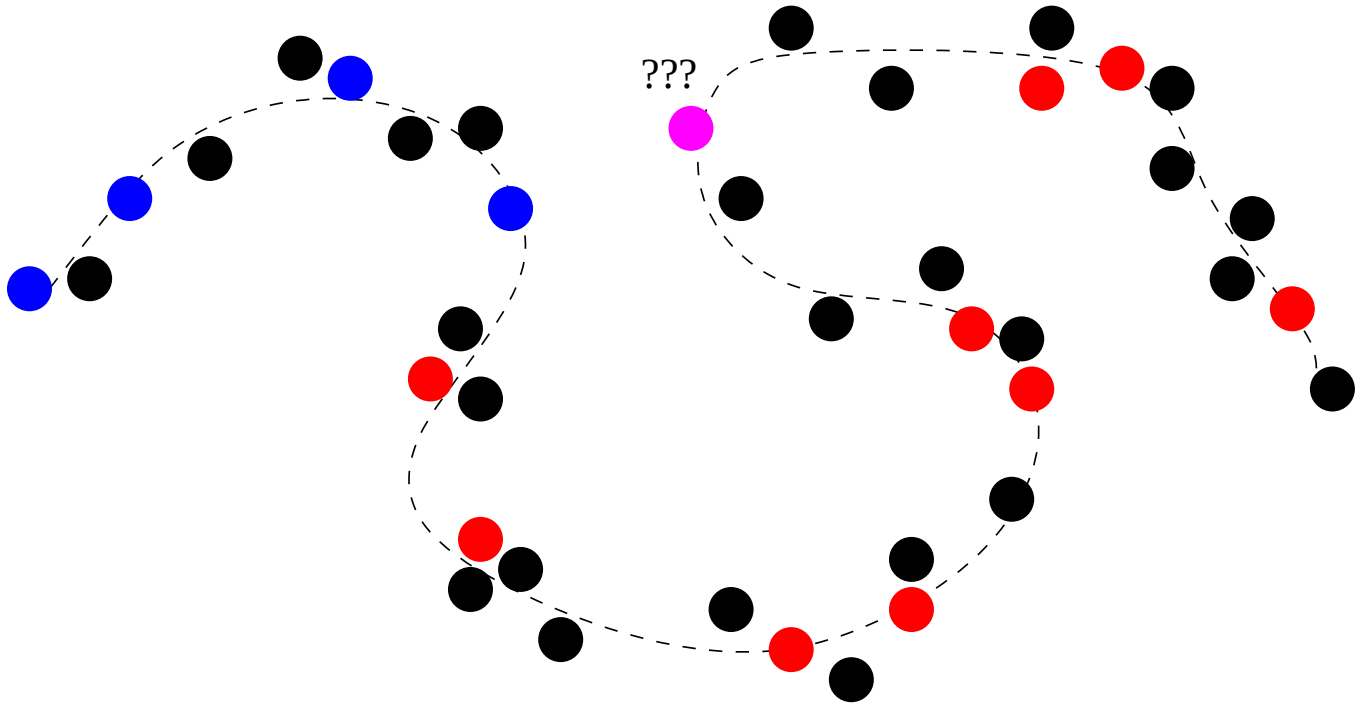
But, popular science journalists don't care about generative models..

Manifold Learning – Semi-Supervised Learning (1)



- Which class does the dot belong to, red or blue?

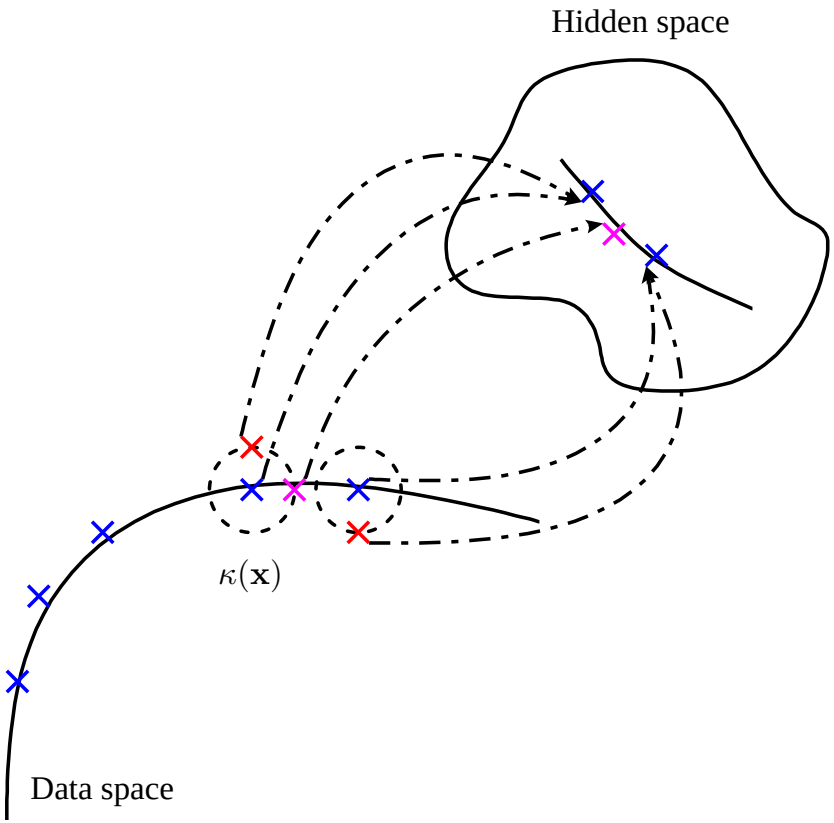
Manifold Learning – Semi-Supervised Learning (2)



- Now, which class does the dot belong to, red or blue?
- The black dots are *unlabeled*

Manifold Learning – New Representation (1)

Representation ϕ *on* the data manifold?



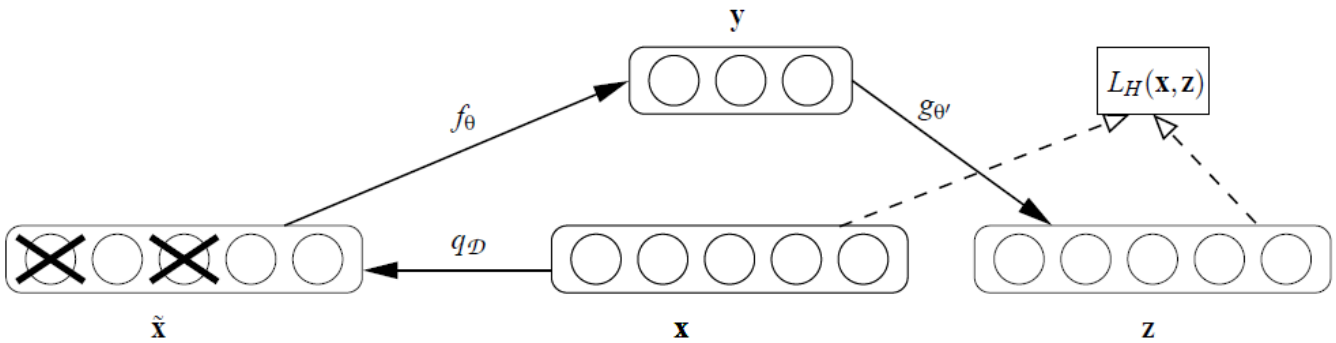
1. ϕ should reflect changes along the manifold

   $$\phi(x_i) \neq \phi(x_j), \text{ for all } x_i, x_j \in D$$

2. ϕ should *not* reflect any change *orthogonal* to the manifold

   $$\phi(x_i + \epsilon) = \phi(x_i)$$

Manifold Learning – New Representation (2)
Denoising Autoencoder



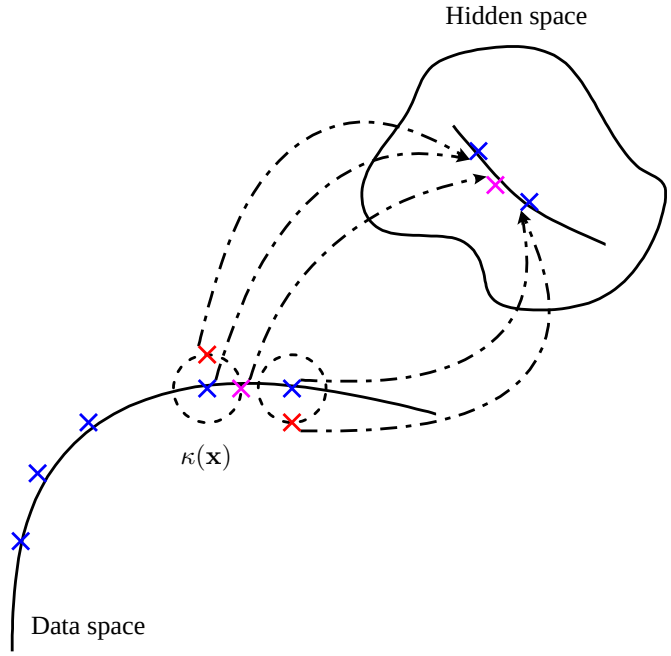
(Vincent et al., 2011)

Representation that capture manifold

1. $\phi(x_i) \neq \phi(x_j)$, for all $x_i, x_j \in D$
2. $\phi(x_i + \epsilon) = \phi(x_i)$

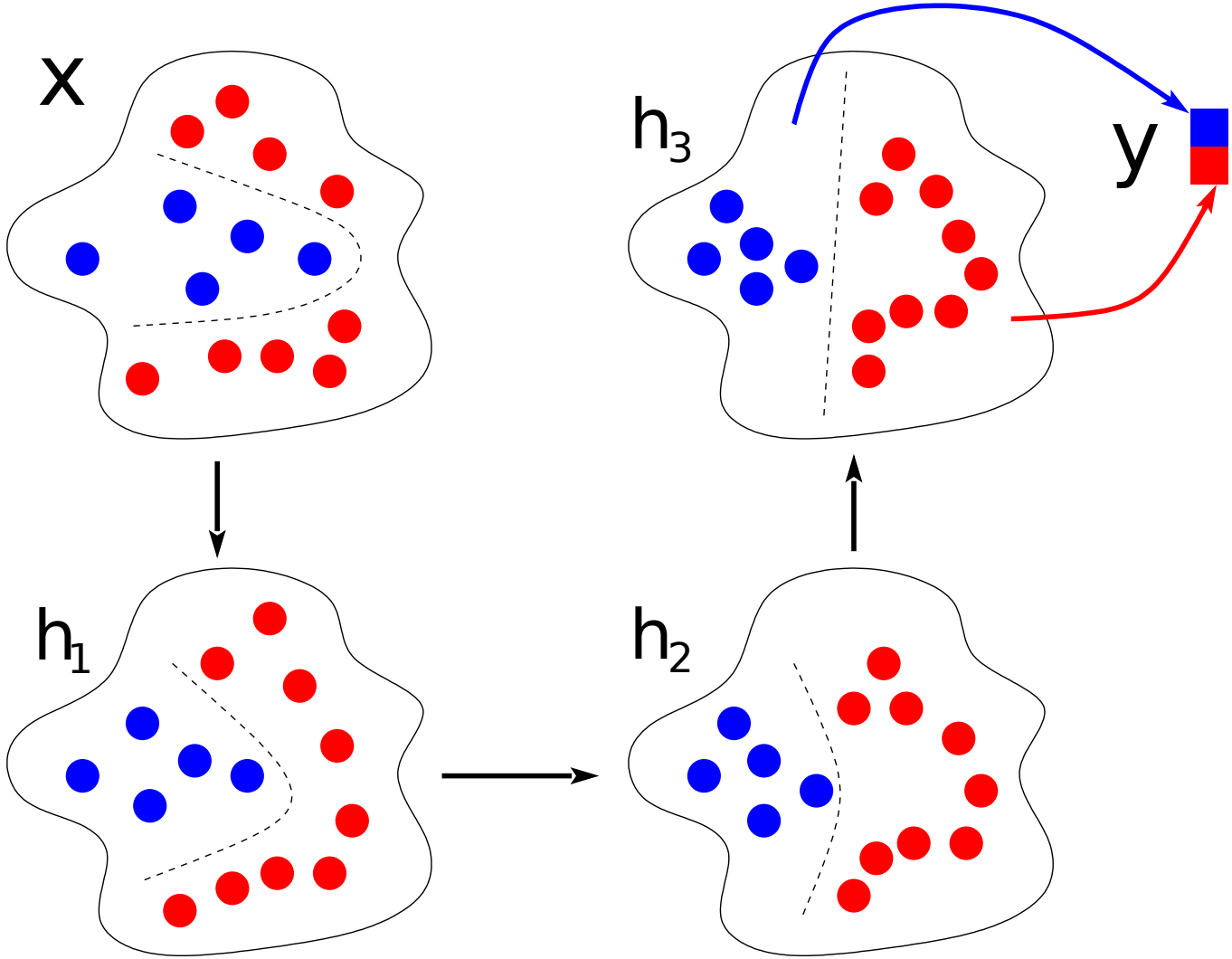
Denoising autoencoder achieves it by

$$\min_{\boldsymbol{\theta},\boldsymbol{\theta}'} \left\| \mathbf{x} - g_{\boldsymbol{\theta}'}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x} + \epsilon\right)\right)\right\|^2$$

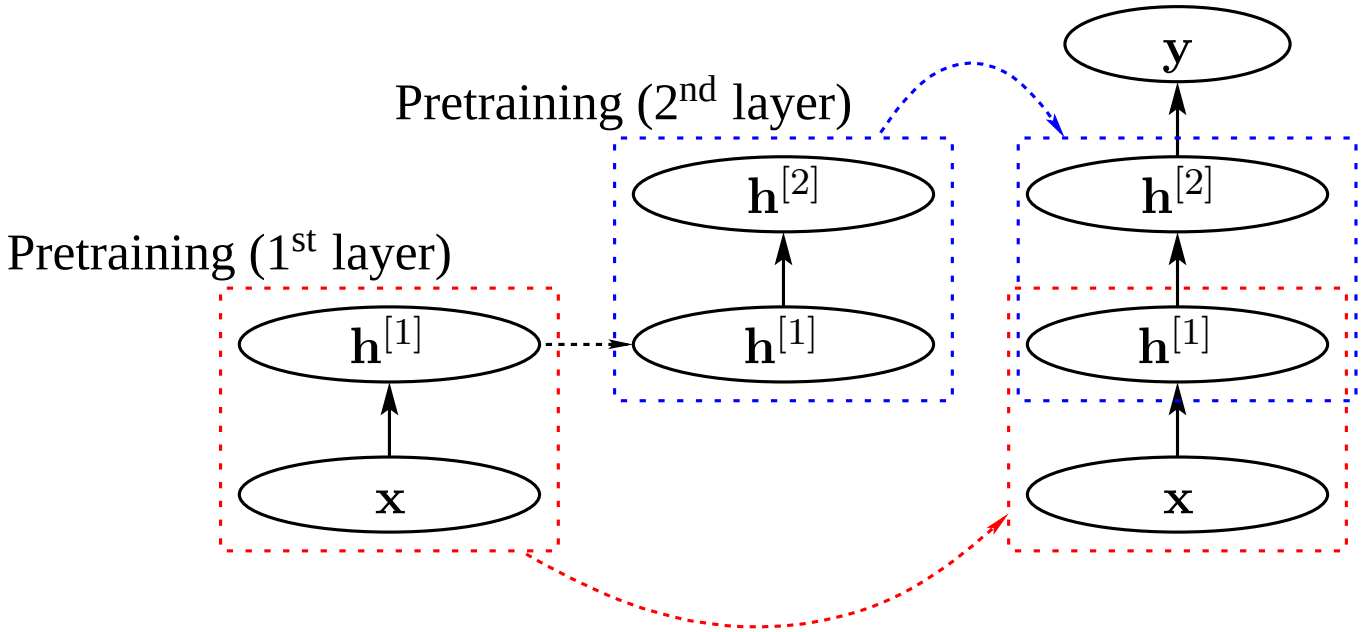


Semi-Supervised Learning in Action (1)

Layer-wise Pretraining



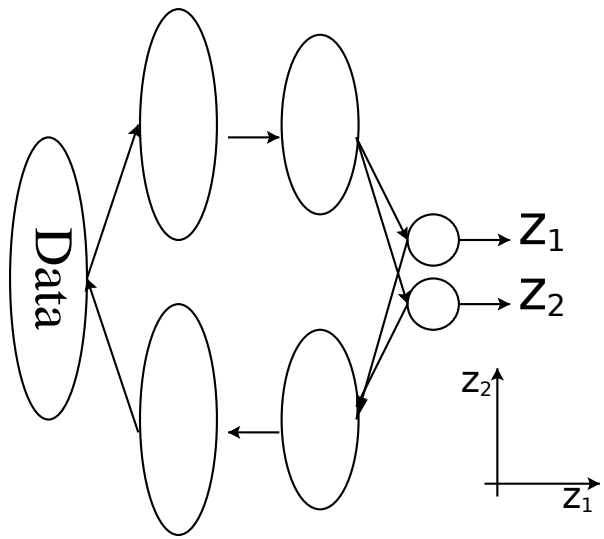
Semi-Supervised Learning in Action (2)
Layer-wise Pretraining



(Hinton&Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007)

Manifold Embedding and Visualization – (1)

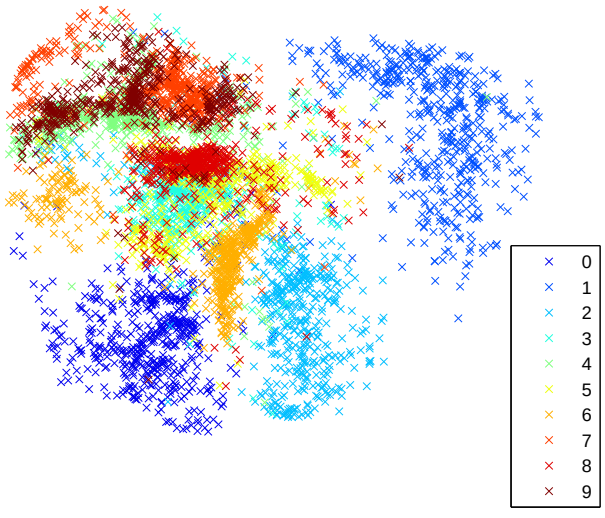
- Manifold Embedding: $\mathcal{M} \subset \mathbb{R}^d \rightarrow \mathbb{R}^q$, $q \ll d$
- If $q = 2$ or 3, data visualization



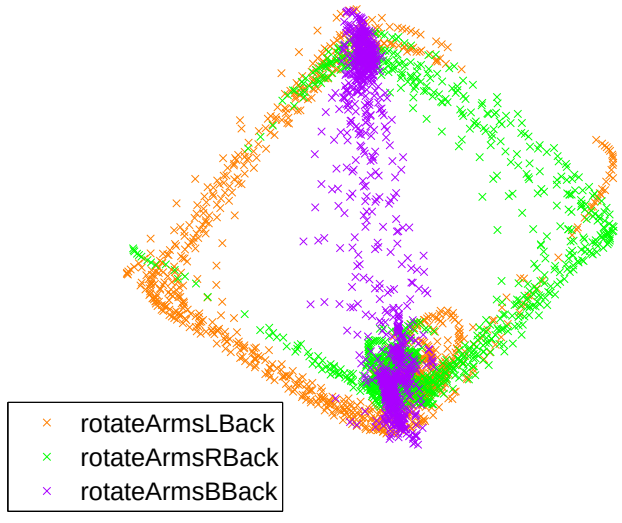
(Oja, 1991; Kramer, 1991; Hinton & Salakhutdinov, 2006)

Manifold Embedding and Visualization – (2)

Handwritten Digits $[0,1]^{196} \rightarrow \mathbb{R}^2$



Pose Frame Data $\mathbb{R}^{30} \rightarrow \mathbb{R}^2$



What other applications can you think of?

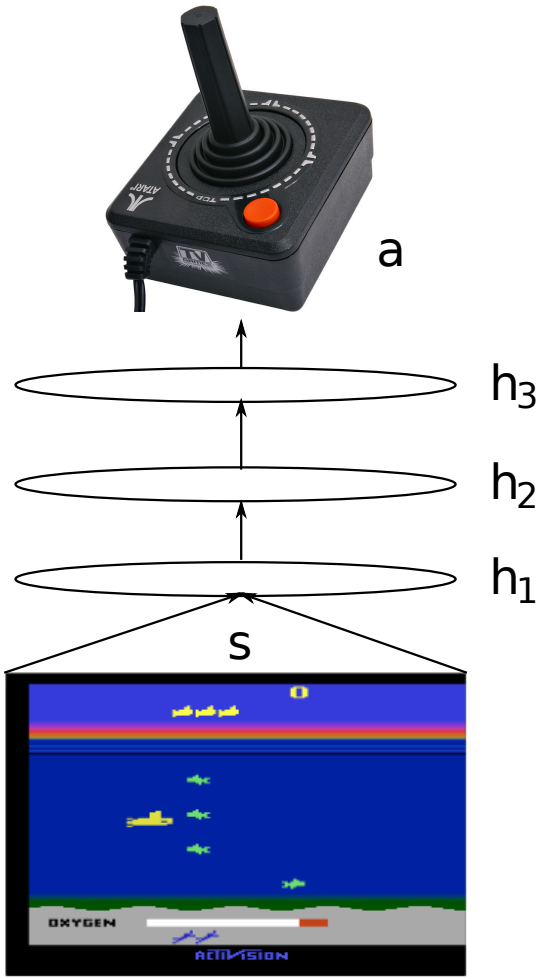
Advanced Topics

Kyunghyun Cho

Laboratoire d'Informatique des Systèmes Adaptatifs,
Département d'informatique et de recherche opérationnelle,
Facult des arts et des sciences,
Université de Montréal
cho.k.hyun@gmail.com (chokyun@iro.umontreal.ca)

Is deep learning all about computer vision and speech recognition?

Deep Reinforcement Learning



Q Learning

- $Q(s, a)$: state-action function
- Action at time t $= \arg\max_{a \in [1,j]} Q(s, a)$
- Update Q on-the-fly

Deep Q Learning

- Model Q with a deep neural network
- Predict $Q(a_j, \cdot)$ for all j at once
- State s: visual perception *not internal states!*

(Mnih et al., 2013)

Natural Language Processing

In neuropsychology, linguistics and the philosophy of language, a natural language or ordinary language is any language which arises, unpremeditated, in the brains of human beings.

–Wikipedia

Natural Language Processing

To machine learning researchers:

Natural Language is a huge set of variable-length sequences of high-dimensional vectors.

Natural Language Processing – (1)
How should we represent a linguistic symbol?

Say, we have four symbols (words):

$$[\textbf{EU}],\ [\textbf{3}],\ [\textbf{France}],\ [\textbf{three}]$$

Most naïve, uninformative coding:

$$\begin{aligned}[\textbf{EU}] &= [1,0,0,0]\\ [\textbf{3}] &= [0,1,0,0]\\ [\textbf{France}] &= [0,0,1,0]\\ [\textbf{three}] &= [0,0,0,1]\end{aligned}$$

Not satisfying..

Natural Language Processing – (2)
How should we represent a linguistic symbol?

Say, we have four symbols (words):

$$[\textbf{EU}],\ [\textbf{3}],\ [\textbf{France}],\ [\textbf{three}]$$

Is there a representation that preserves the similarities of *meaning*s of symbols?

$$\begin{aligned} D([\textbf{EU}], [\textbf{France}]) &< D([\textbf{EU}], [\textbf{3}], \\ D([\textbf{3}], [\textbf{three}]) &< D([\textbf{France}], [\textbf{three}], \\ D([\textbf{3}], [\textbf{three}]) &< \epsilon \\ &\vdots \end{aligned}$$

Natural Language Processing – (3)
Continuous-Space Representation

Sample sentences:

1. There are three teams left for the qualification.
2. 3 teams have passed the first round.

Task: Predict a following word given a current work [**three**]

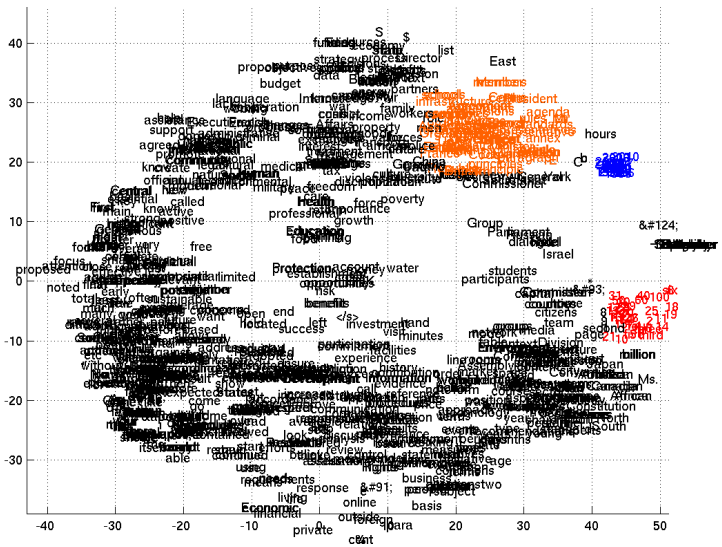
Naïve approach: build a table (so called *n*-gram)

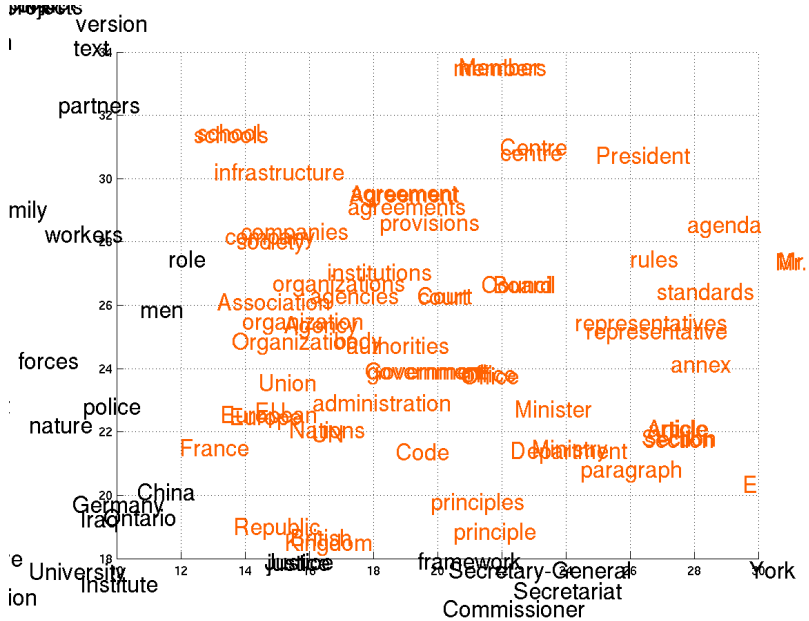
- (three, teams), (3, teams)
- The table can grow arbitrarily.

Machine learning: *compress* the table into a continuous function

- Map three and 3 to nearby points x in a continuous space
- From x, map to [**teams**].

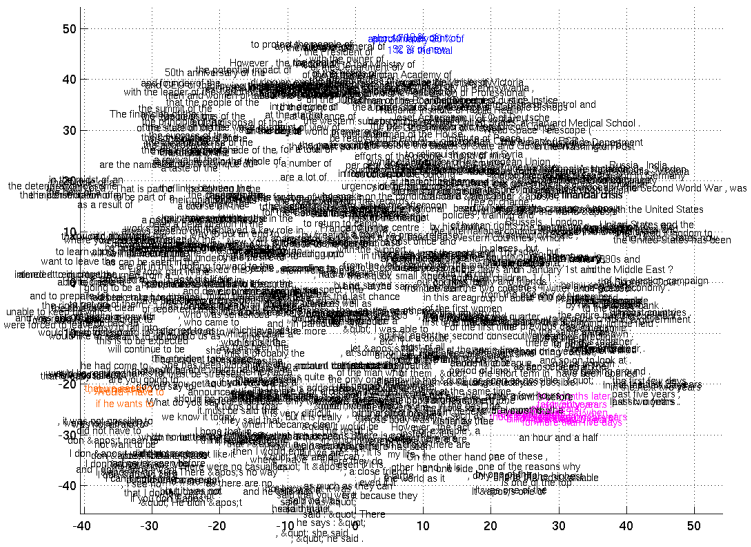
Natural Language Processing – (4)
Continuous-Space Representation

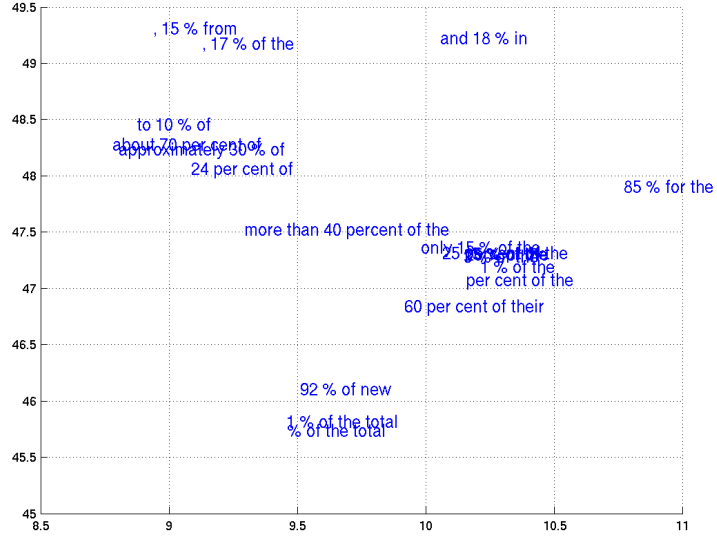




(Cho et al., 2014)

Natural Language Processing – (5)
Beyond Word Representation





(Cho et al., 2014)

NN: *I am very powerful and can model anything as long as I'm fed enough computational resource.*

SVM: *But, you have to optimize a high-dimensional, non-convex function which has many, many local minima!*

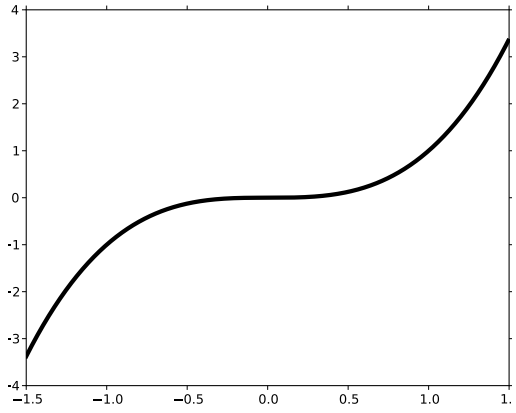
NN: *Really?*

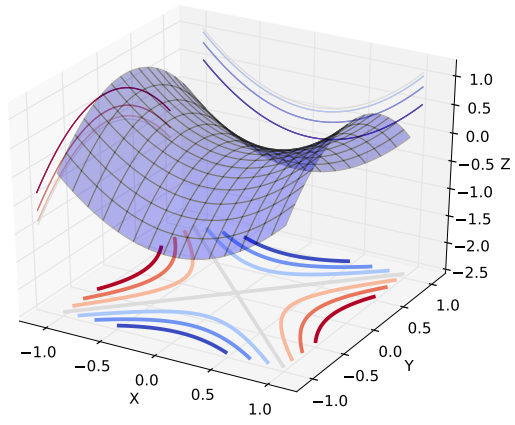
Advanced Optimization – (1) Statistical Physics says. . .

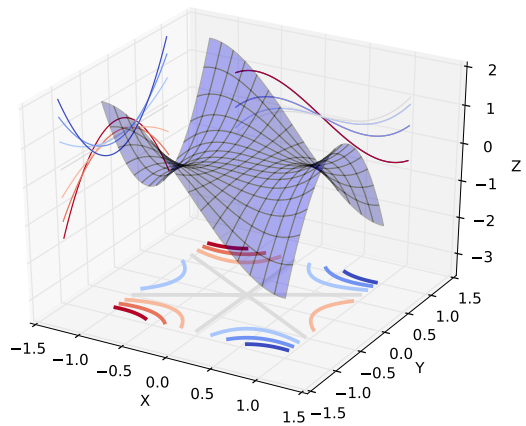
Not really

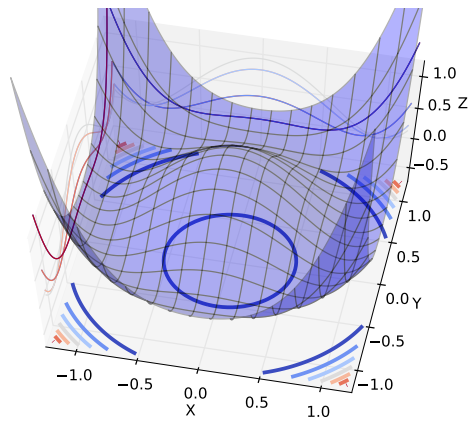
Advanced Optimization – (2)

Local Minima? Saddle Points?









(Dauphin et al., 2014; Pascanu et al., 2014)

Advanced Optimization – (3)
Beyond the 2nd-order Method

(Quasi-)Newton Method

$$\theta \leftarrow \theta - H^{-1} \nabla L(\theta)$$

*H*ow well does the **quadratic** approximation hold when training neural networks?

Saddle-Free Newton Method (*very new!!*) (Dauphin et al., 2014)

$$\theta \leftarrow \theta - |H|^{-1} \nabla L(\theta),$$

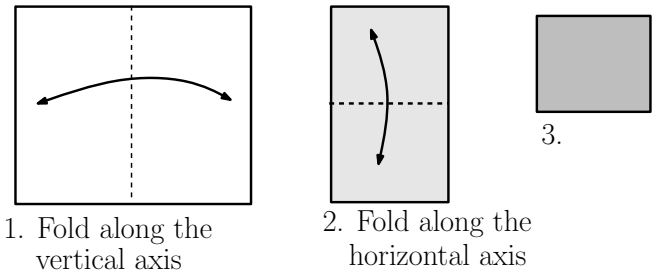
where $|H|$ is constructed by

$$|H| = U |\Sigma| V$$

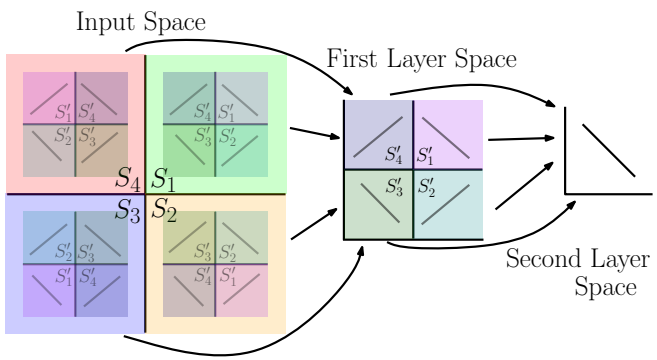
when $H = U \Sigma V$.

Lastly but not at all least,
is there any theoretical ground for using deep neural networks?

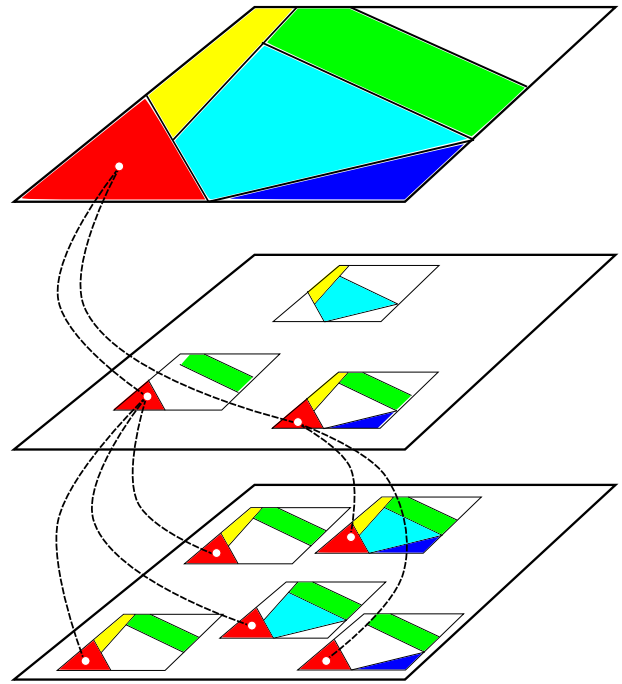
Theoretical Analysis –
Deep Rectifier Networks *Fold* the Space



(a)



(b)



(c)

(Montufar et al., 2014; Pascanu et al., 2014)

Is it the beginning of deep learning or the end of deep learning?