Deep Learning의 이해

2016.3.30
Kmobile 딥러닝 1-day 워크샵

Eun-Sol Kim

Biointelligence Laboratory
Department of Computer Science and Engineering
Seoul National Univertisy

http://bi.snu.ac.kr

목차

History of Neural Network Research

The New York Times

Google Hong Kong

Neural network
Back propagation

Deep belief net
Science

Microsoft

Speech

IMAGENET

1986 2006 2011 2012

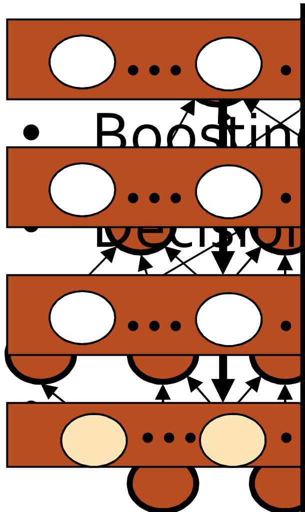
deep learning results

- Unsupervised & Layer-wised pre-training

ing (norm
blems

P, HOG)

ectures



- Boosting
- Decision

Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep Conv Net
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

Deep Networks Advance State of Art in Speech

Object recognition over 1,000,000 images and 1,000 categories
(2 GPU)

Deep Learning leads to breakthrough in speech recognition at MSR.

Microsoft

Slides from Wanli Ouyang wlouyang@ee.cuhk.edu.hk

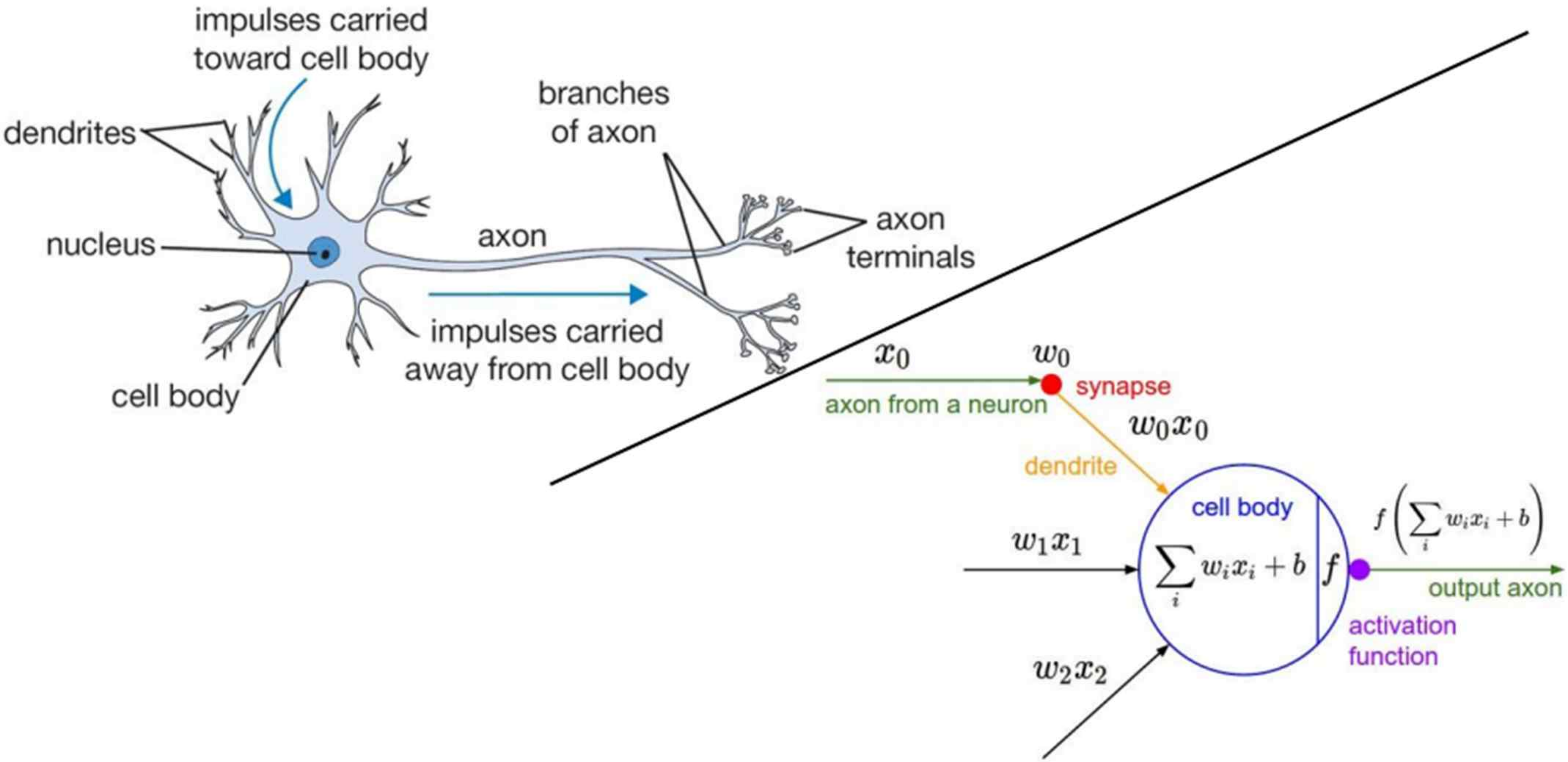
Classification Problem

- 데이터 x가 주어졌을 때 해당되는 레이블 y를 찾는 문제
  - ex1) x: 사람의 얼굴 이미지,   y: 사람의 이름
  - ex2) x: 혈당 수치, 혈압 수치, 심박수,   y: 당뇨병 여부
  - ex3) x: 사람의 목소리,   y: 목소리에 해당하는 문장

- x: D차원 벡터,   y: 정수 (Discrete)

- 대표적인 패턴 인식 알고리즘
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbor
  - Multi-Layer Perceptron (Artificial Neural Network; 인공신경망)

Perceptron (1/3)

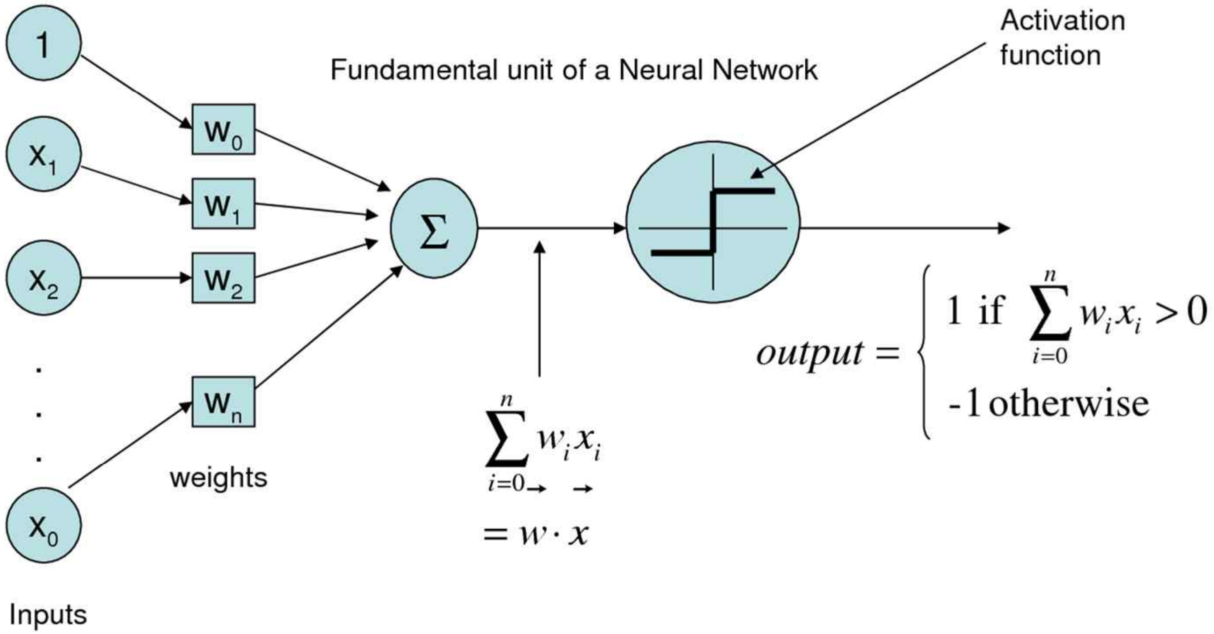


http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

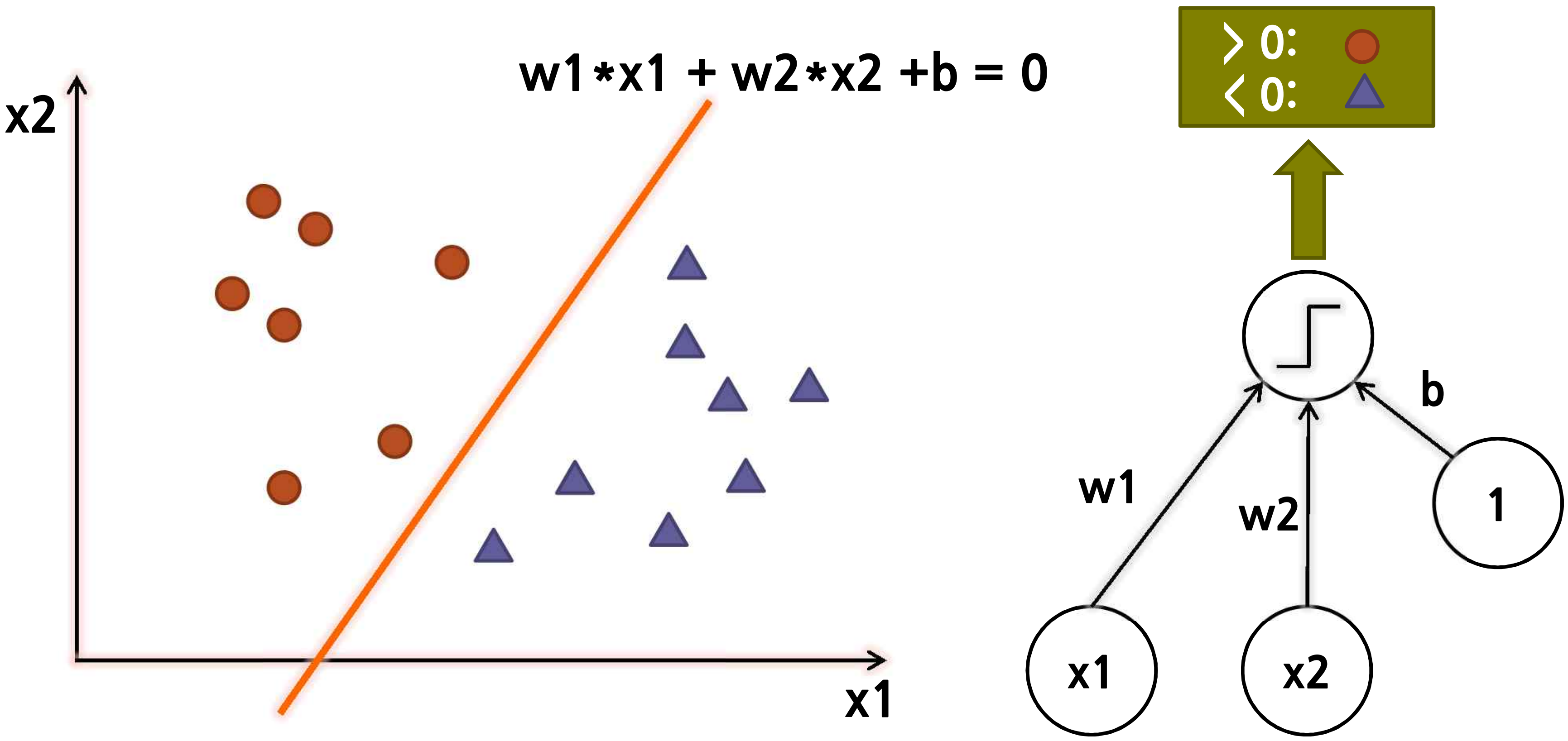
Perceptron (2/3)

Artificial Neural Networks

The Perceptron

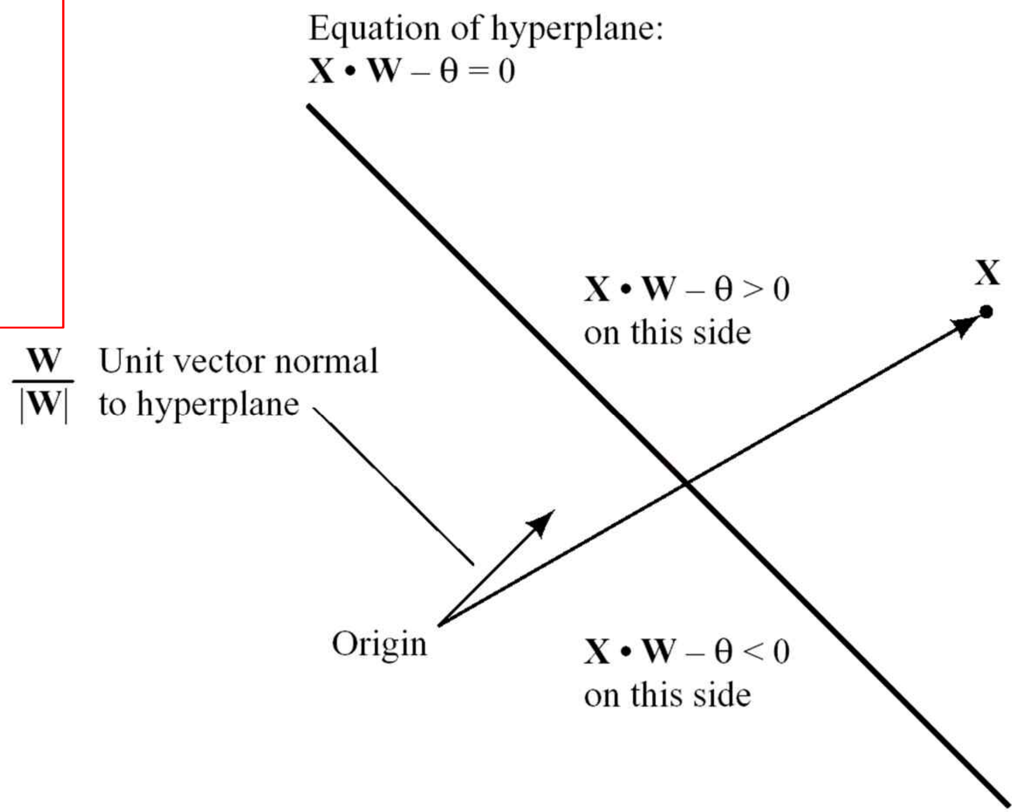


Perceptron (3/3)

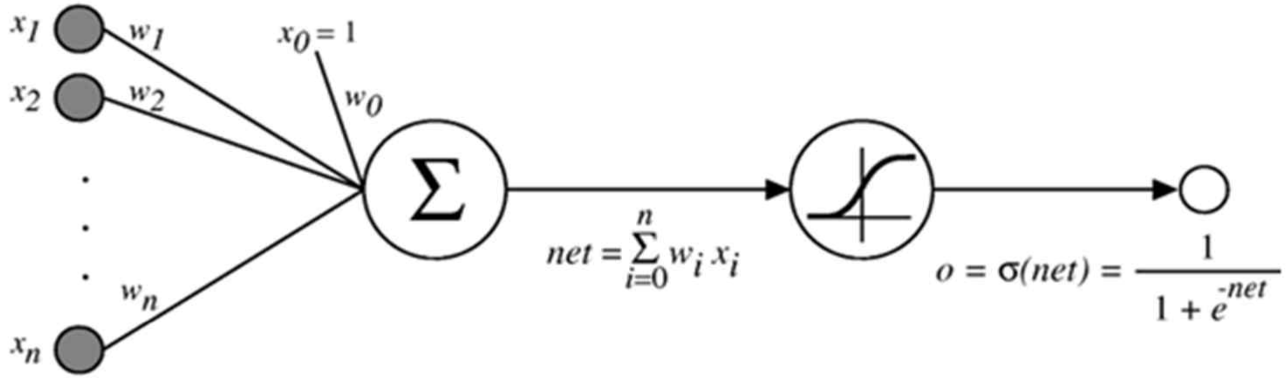


Parameter Learning in Perceptron

start:
The weight vector w is generated randomly
test:
A vector $x \in P \cup N$ is selected randomly,
If $x \in P$ and $w \cdot x > 0$ goto test,
If $x \in P$ and $w \cdot x \leq 0$ goto add,
If $x \in N$ and $w \cdot x < 0$ go to test,
If $x \in N$ and $w \cdot x \geq 0$ go to subtract.
add:
Set $w = w + x$, goto test
subtract:
Set $w = w - x$, goto test

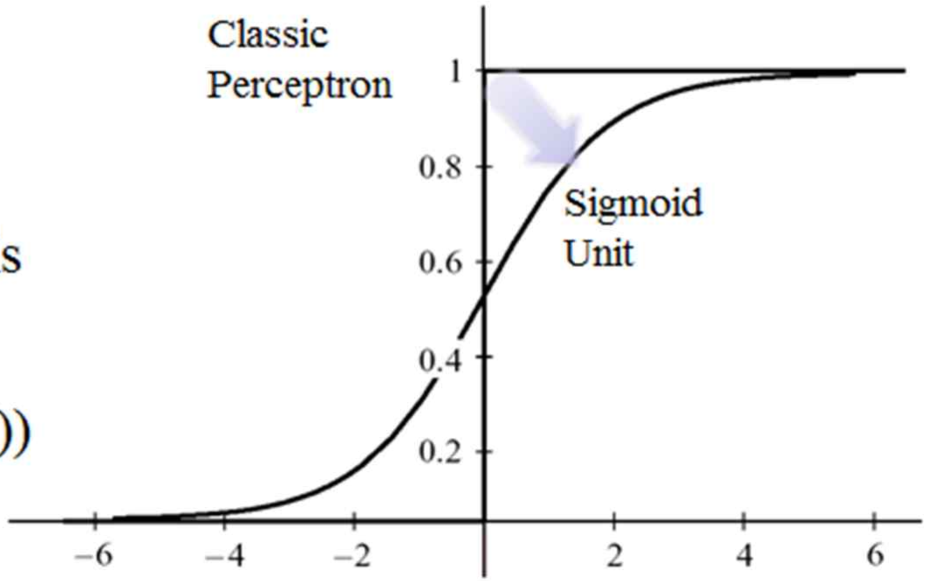


Sigmoid Unit



Sigmoid function is
Differentiable

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



Learning Algorithm of Sigmoid Unit

- Loss Function

Target, Unit Output

$$\varepsilon = (d - f)^2$$

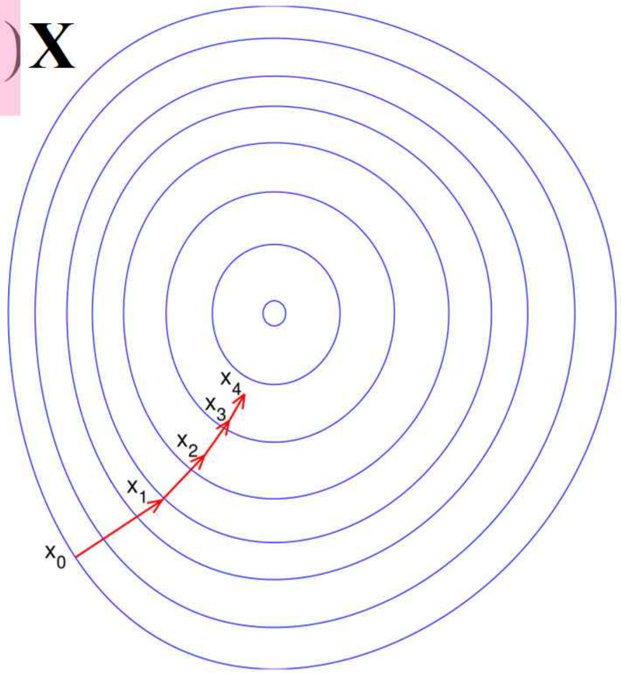
- Gradient Descent Update

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d-f)\frac{\partial f}{\partial s}\mathbf{X} = -2(d-f)f(1-f)\mathbf{X}$$

$$f(s) = 1/(1+e^{-s})$$

$$f'(s) = f(s)(1-f(s))$$

$$\mathbf{W} \leftarrow \mathbf{W} + c(d-f)f(1-f)\mathbf{X}$$



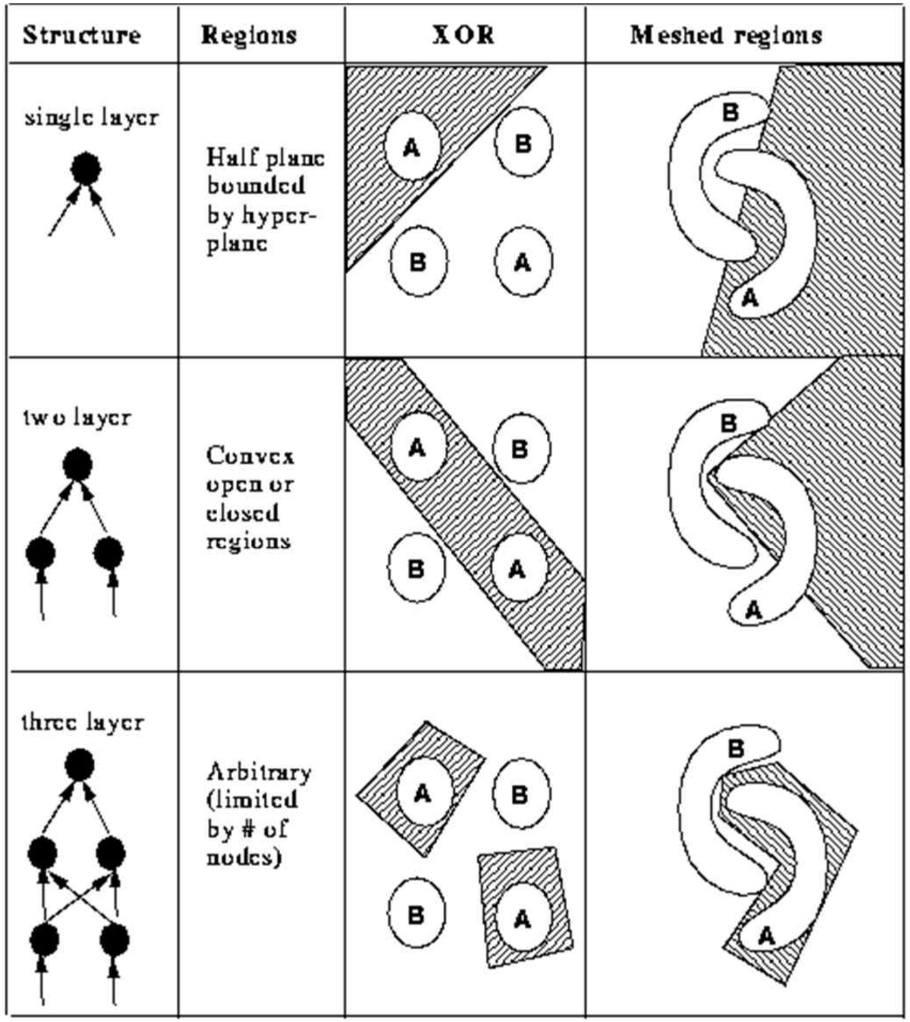
Need for Multiple Units and Multiple Layers

- Multiple boundaries are needed (e.g. XOR problem)

→ Multiple Units

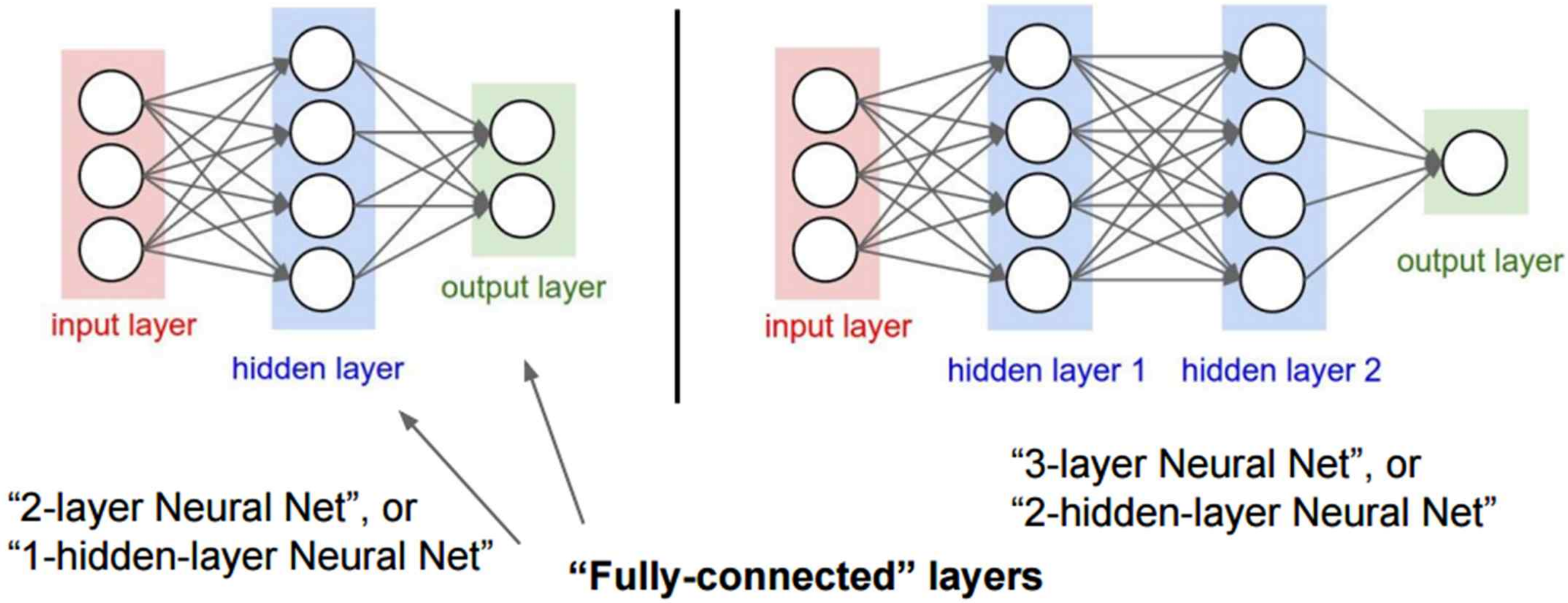
- More complex regions are needed (e.g. Polygons)

→ Multiple Layers



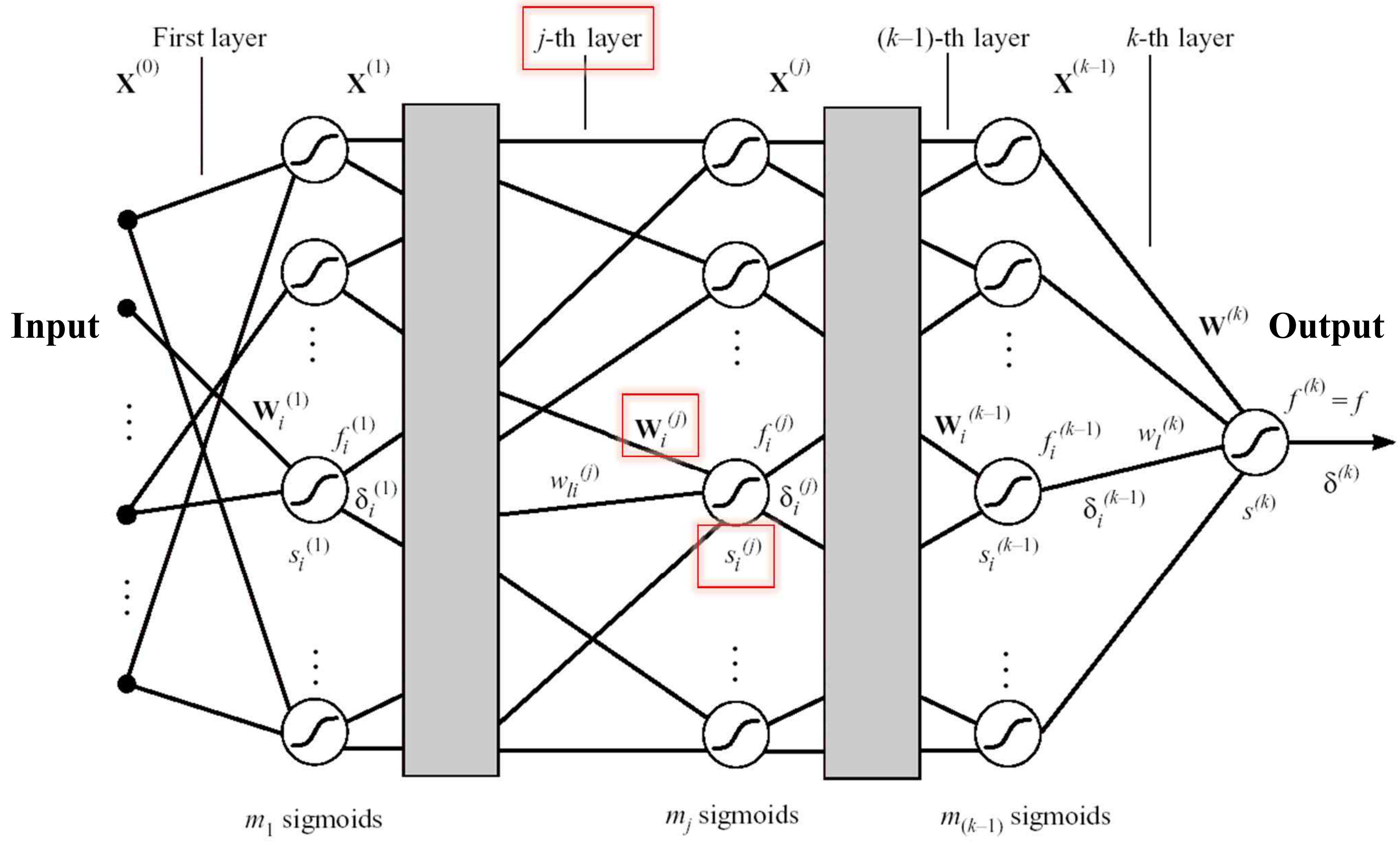
http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Structure of Multilayer Perceptron



http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

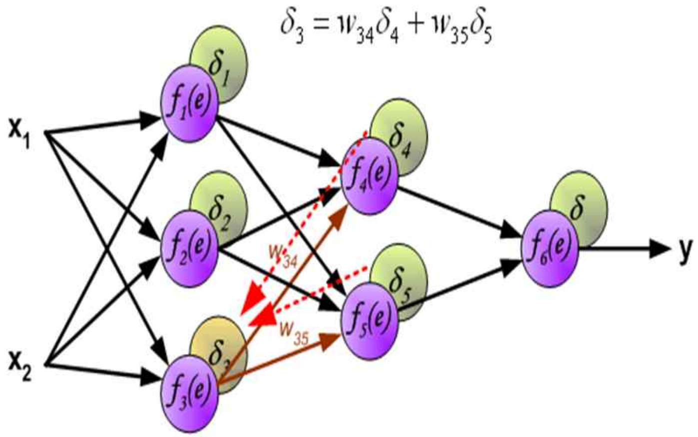
Structure of Multilayer Perceptron (MLP; Artificial Neural Network)



Learning Parameters of MLP

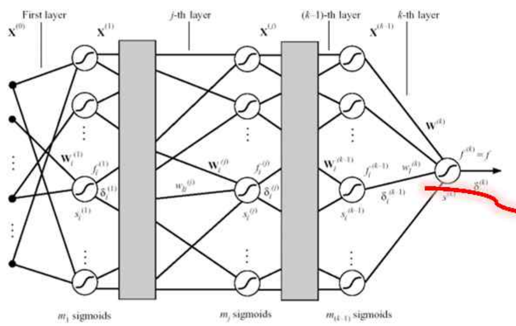
- Loss Function
  - We have the same Loss Function
  - But the # of parameters are now much more (Weight for **each layer** and **each unit**)
  - To use Gradient Descent, we need to calculate the **gradient for all the parameters**
- Recursive Computation of Gradients
  - Computation of loss-gradient of **the top-layer** weights is **the same** as before
  - Using the **chain rule**, we can compute the loss-gradient of lower-layer weights **recursively (Back Propagation)**

$$\varepsilon = (\underbrace{d}_{\text{Target}} - \underbrace{f}_{\text{Unit Output}})^2$$



Back Propagation Learning Algorithm (1/3)

- Gradients of <u>top-layer</u> weights and update rule



$$\varepsilon = (d - f)^2$$

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(d - f)\frac{\partial f}{\partial s}\mathbf{X} = -2(d - f)f(1 - f)\mathbf{X}$$

Gradient Descent update rule

$$\mathbf{W} \leftarrow \mathbf{W} + c(d - f)f(1 - f)\mathbf{X}$$

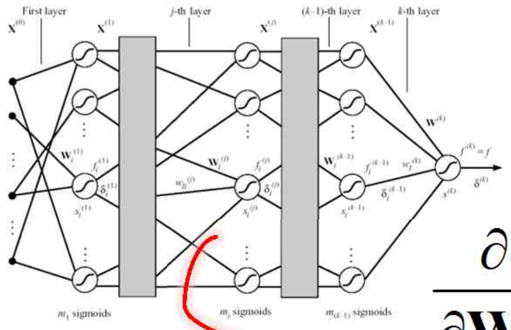
- Store intermediate value delta for later use of chain rule

$$\delta^{(k)} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} = (d - f)\frac{\partial f}{\partial s_i^{(j)}}$$

$$= (d - f)f(1 - f)$$

Back Propagation Learning Algorithm (2/3)

- Gradients of lower-layer weights



Weighted sum

$$s_i^{(j)} = \mathbf{X}^{(j-1)} \cdot \mathbf{W}_i^{(j)}$$

$$\frac{\partial \varepsilon}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \frac{\partial s_i^{(j)}}{\partial \mathbf{W}_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)}$$

$$= -2(d-f)\frac{\partial f}{\partial s_i^{(j)}} \mathbf{X}^{(j-1)} = -2\delta_i^{(j)} \mathbf{X}^{(j-1)}$$

Local gradient

$$\frac{\partial \varepsilon}{\partial s_i^{(j)}} = \frac{\partial (d-f)^2}{\partial s_i^{(j)}} = -2(d-f)\frac{\partial f}{\partial s_i^{(j)}}$$

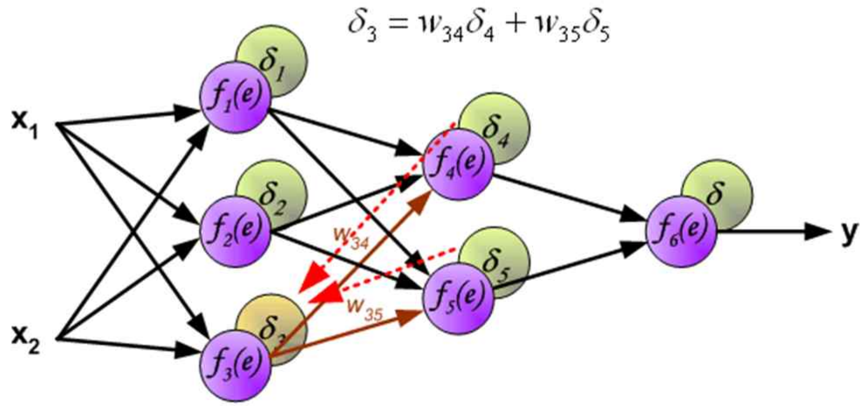
Gradient Descent Update rule for lower-layer weights

$$\mathbf{W}_i^{(j)} \leftarrow \mathbf{W}_i^{(j)} + c_i^{(j)} \delta_i^{(j)} \mathbf{X}^{(j-1)}$$

Back Propagation Learning Algorithm (3/3)

- Applying chain rule, recursive relation between delta's

$$\delta_i^{(j)} = f_i^{(j)}(1 - f_i^{(j)}) \sum_{l=1}^{m_{j+1}} \delta_i^{(j+1)} w_{il}^{(j+1)}$$

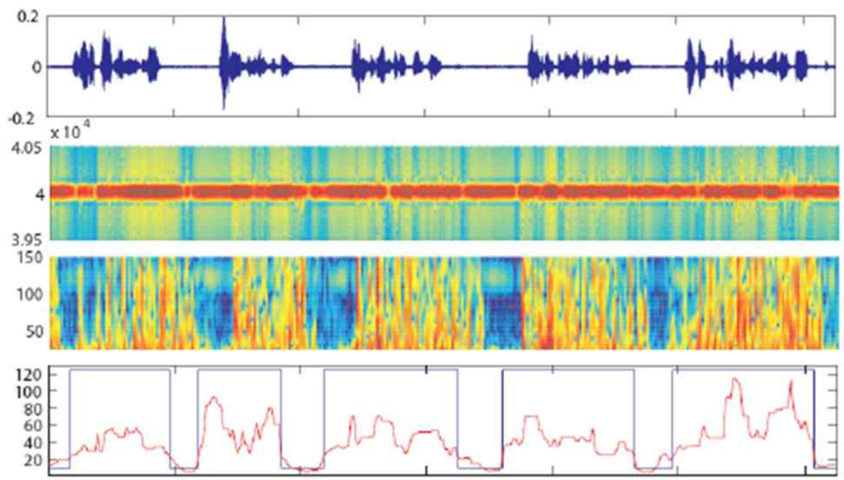


Algorithm: Back Propagation

1. **Randomly Initialize weight parameters**
2. **Calculate the activations of all units (with input data)**
3. **Calculate top-layer delta**
4. **Back-propagate delta from top to the bottom**
5. **Calculate actual gradient of all units using delta's**
6. **Update weights using Gradient Descent rule**
7. **Repeat 2~6 until converge**

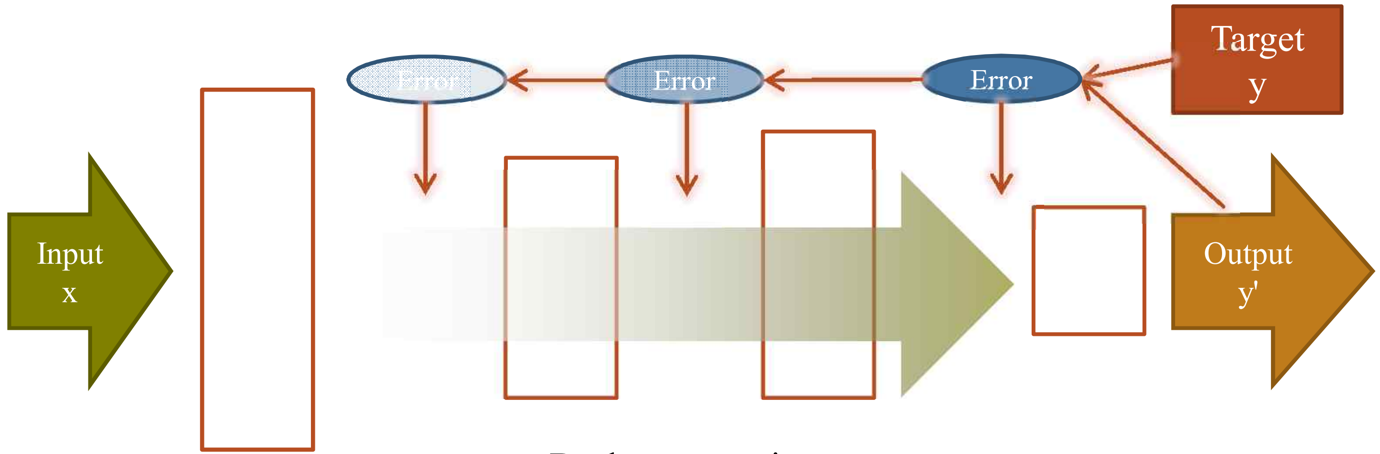
Applications

- Almost All Classification Problems
  - Face Recognition
  - Object Recognition
  - Voice Recognition
  - Spam mail Detection
  - Disease Detection
  - etc.



Limitations and Breakthrough

- Limitations
  - Back Propagation barely changes lower-layer parameters (Vanishing Gradient)
  - Therefore, Deep Networks cannot be fully (effectively) trained with Back Propagation

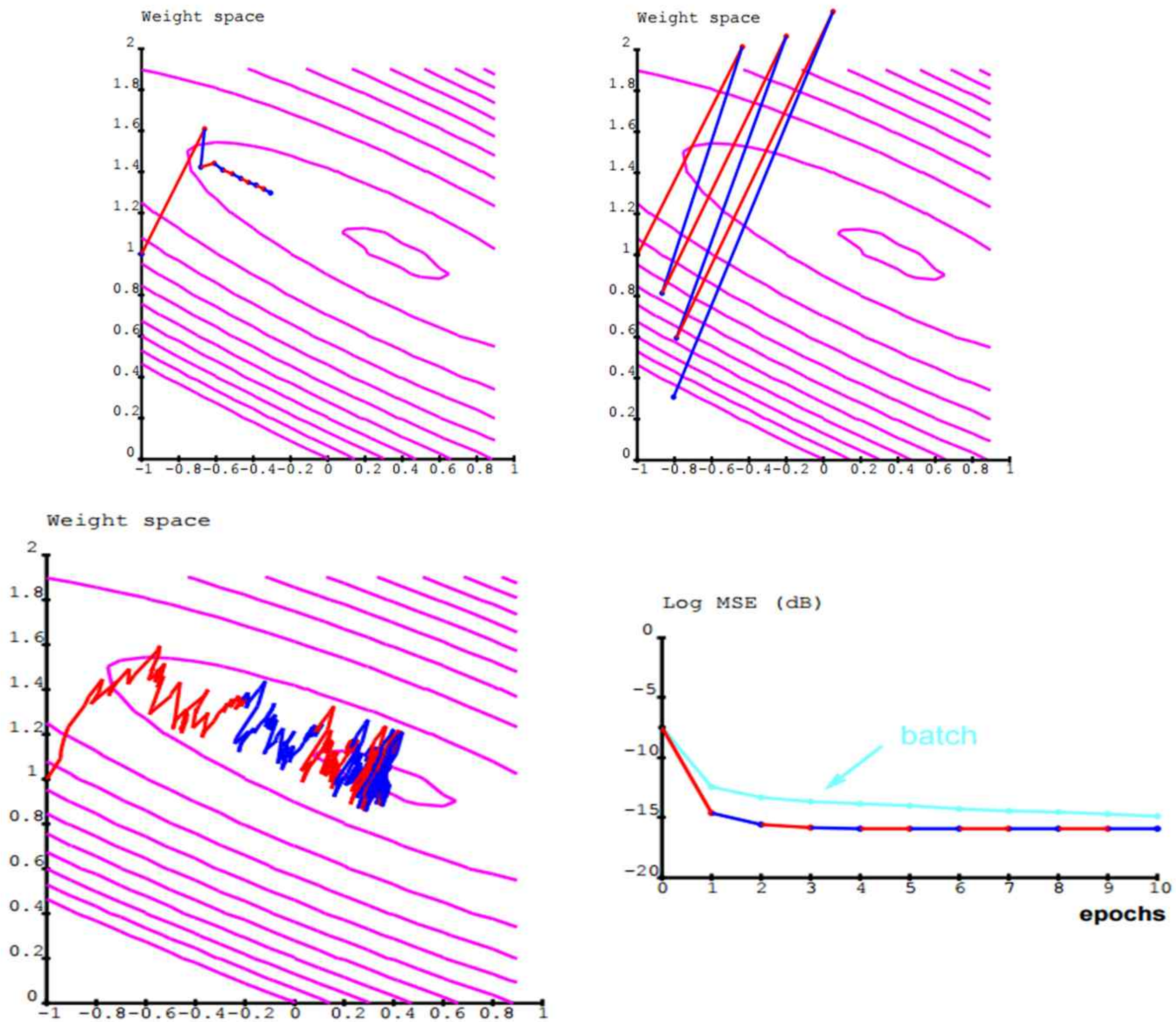


Back-propagation

- Breakthrough
  - Deep Belief Networks (Unsupervised Pre-training)
  - Convolutional Neural Networks (Reducing Redundant Parameters)
  - Rectified Linear Unit (Constant Gradient Propagation)

Some Issues (1/3)

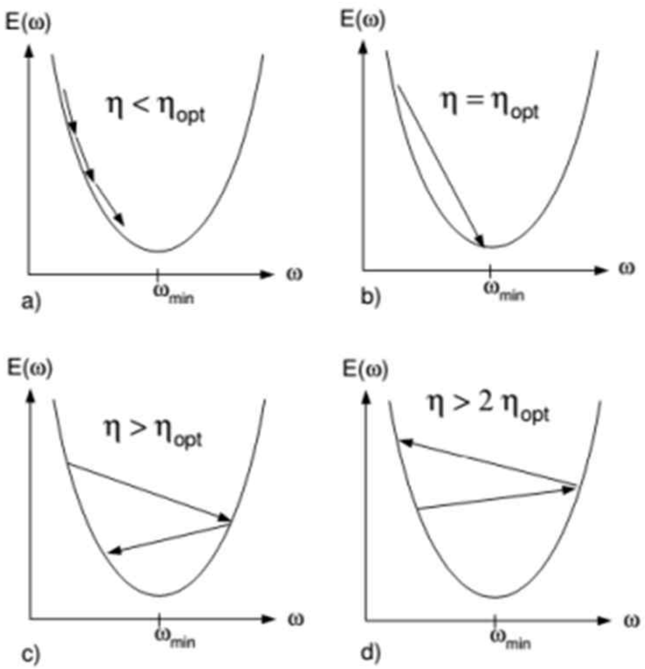
- Stochastic Gradient Descent

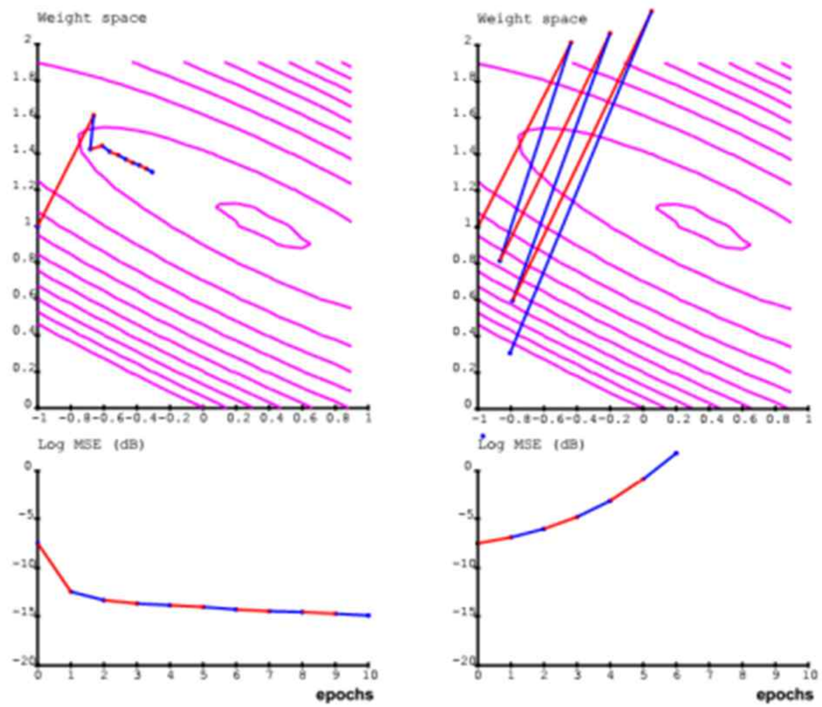


http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

Some Issues (2/3)

- Learning Rate Adaptation
- Momentum
- Weight Decay

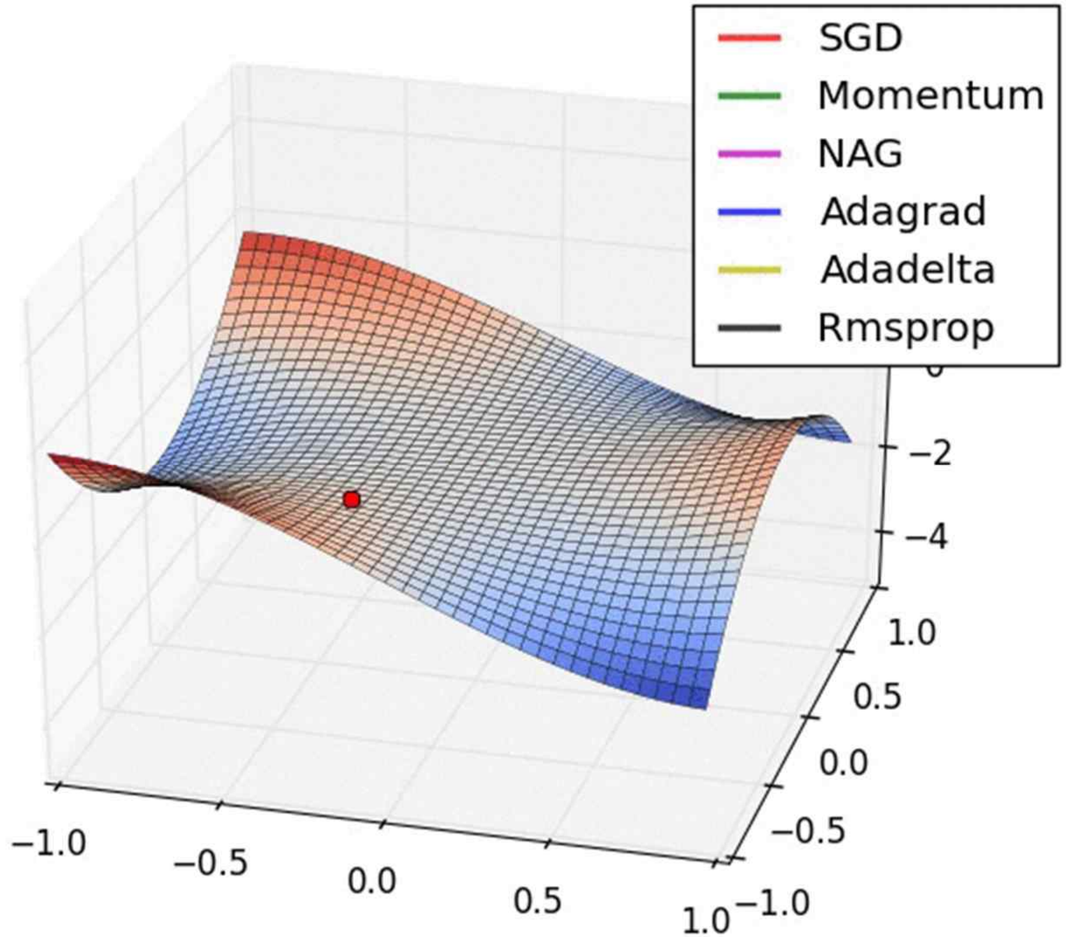




http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
http://courses.cs.tau.ac.il/Caffe_workshop/Bootcamp/pdf_lectures/Lecture%204%20CNN%20-%20optimization.pdf

Some Issues (3/3)

- State-of-the-art optimization techniques on NN

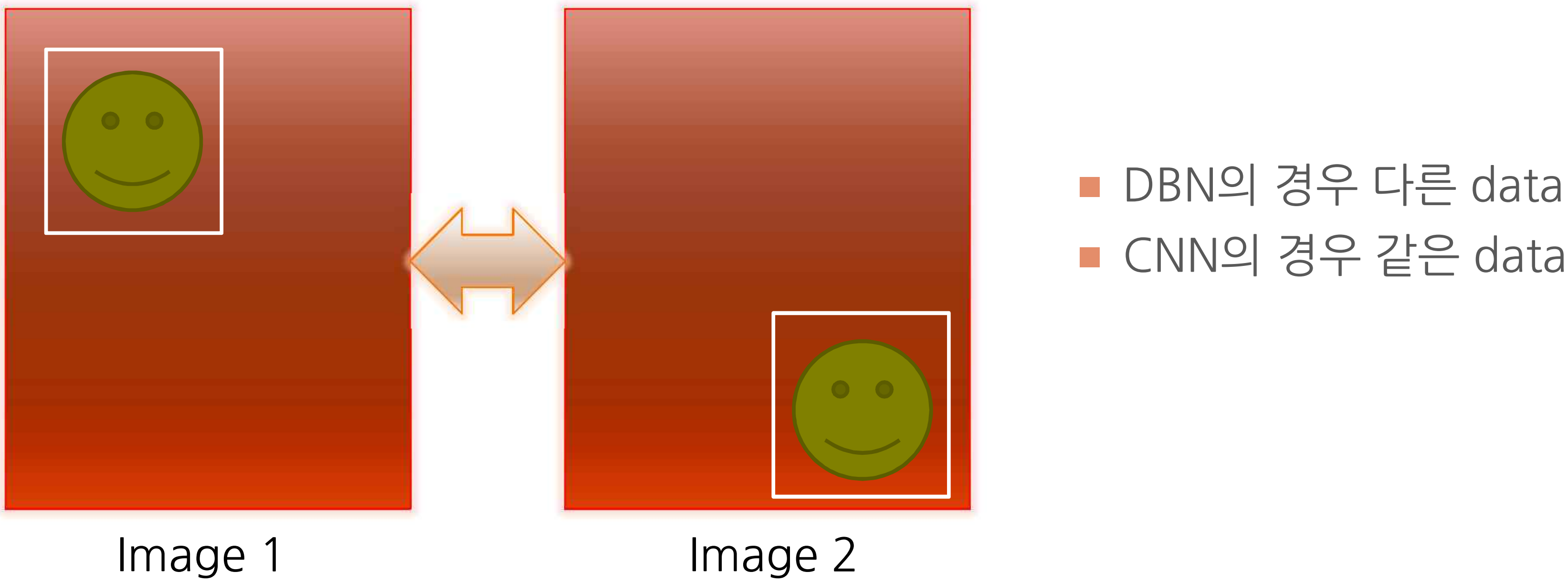


Convolutional Neural Networks

Slides by Jiseob Kim

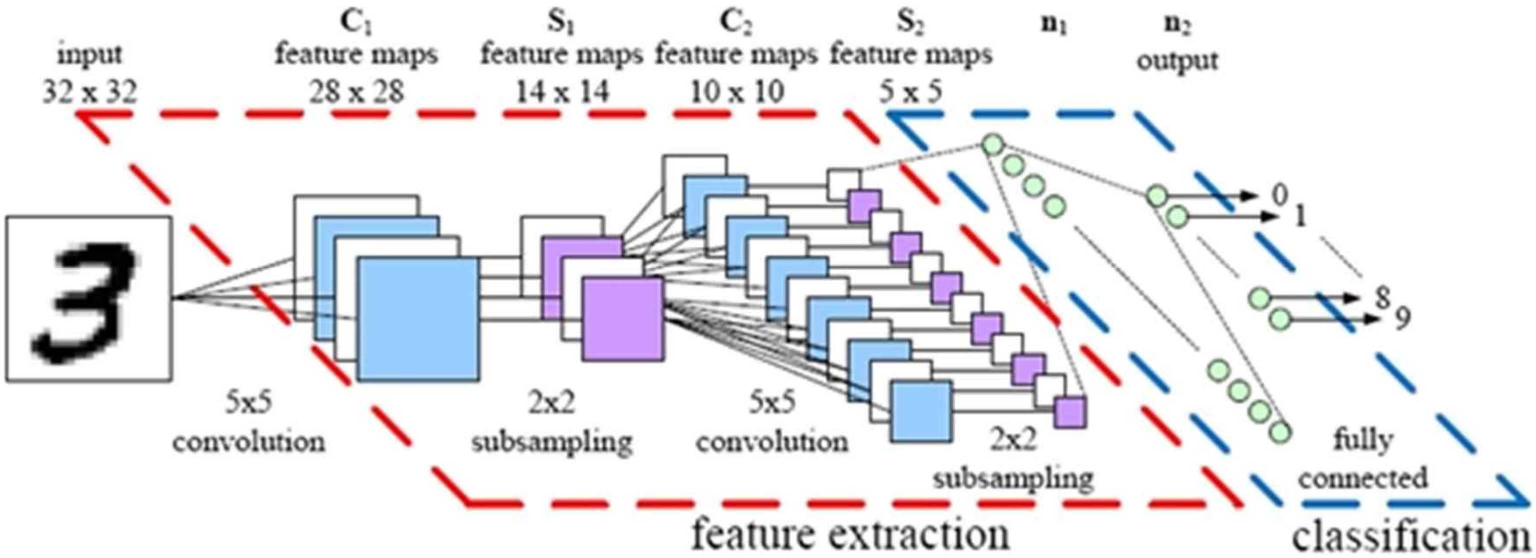
Motivation

- Idea:
  - Fully connected 네트워크 구조는 학습해야할 파라미터 수가 너무 많음
  - 이미지 데이터, 음성 데이터 (spectrogram)과 같이 각 feature들 간의 위상적, 기하적 구조가 있는 경우 Local한 패턴을 학습하는 것이 효과적



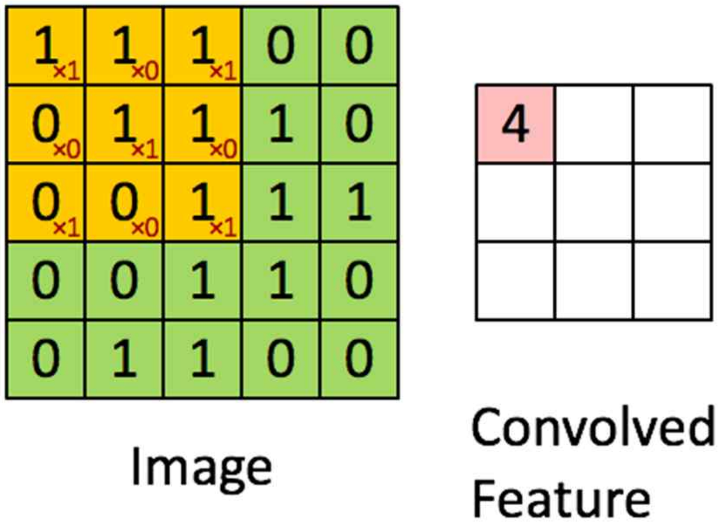
Structure of Convolutional Neural Network (CNN)

- Convolution과 Pooling (Subsampling)을 반복하여 상위 Feature를 구성
- Convolution은 Local영역에서의 특정 Feature를 얻는 과정
- Pooling은 Dimension을 줄이면서도, Translation-invariant한 Feature를 얻는 과정

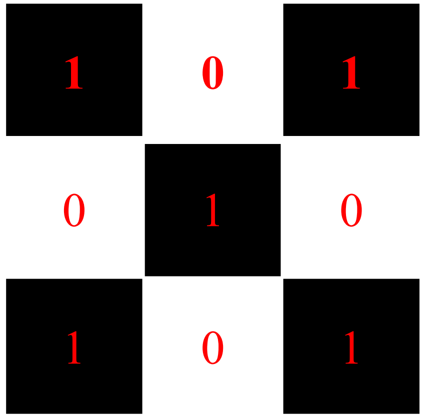


http://parse.ele.tue.nl/education/cluster2

Convolution Layer

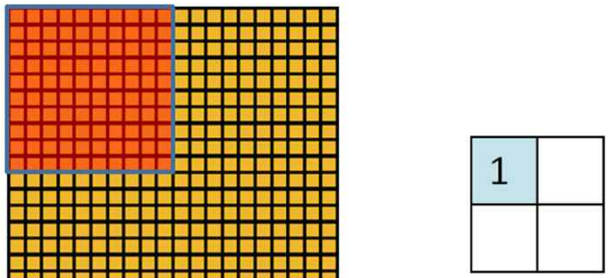


- The Kernel Detects pattern:



- The Resulting value Indicates:
  - How much the pattern matches at each region

Max-Pooling Layer

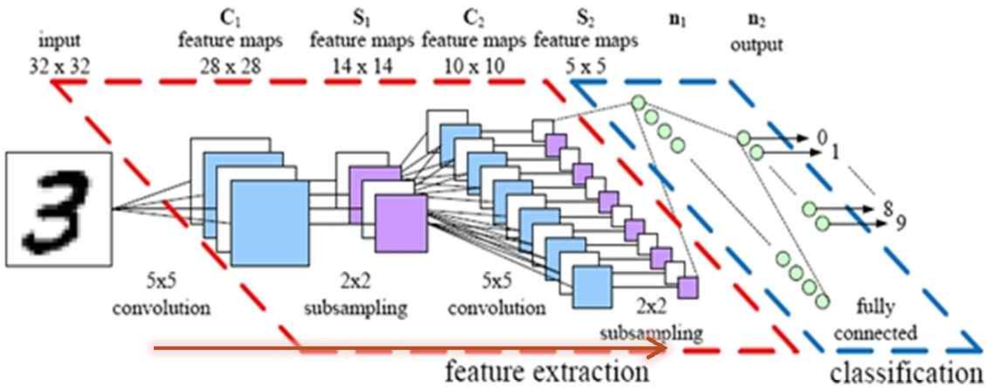


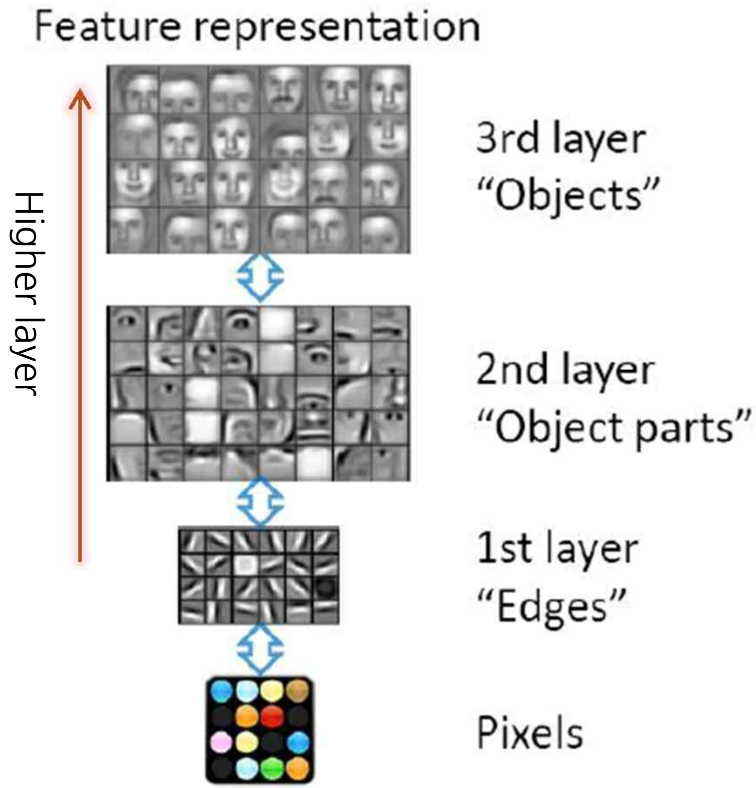
Convolved feature

Pooled feature

- The Pooling Layer summarizes the results of Convolution Layer
  - e.g.) 10x10 result is summarized into 1 cell
- The Result of Pooling Layer is Translation-invariant

Remarks



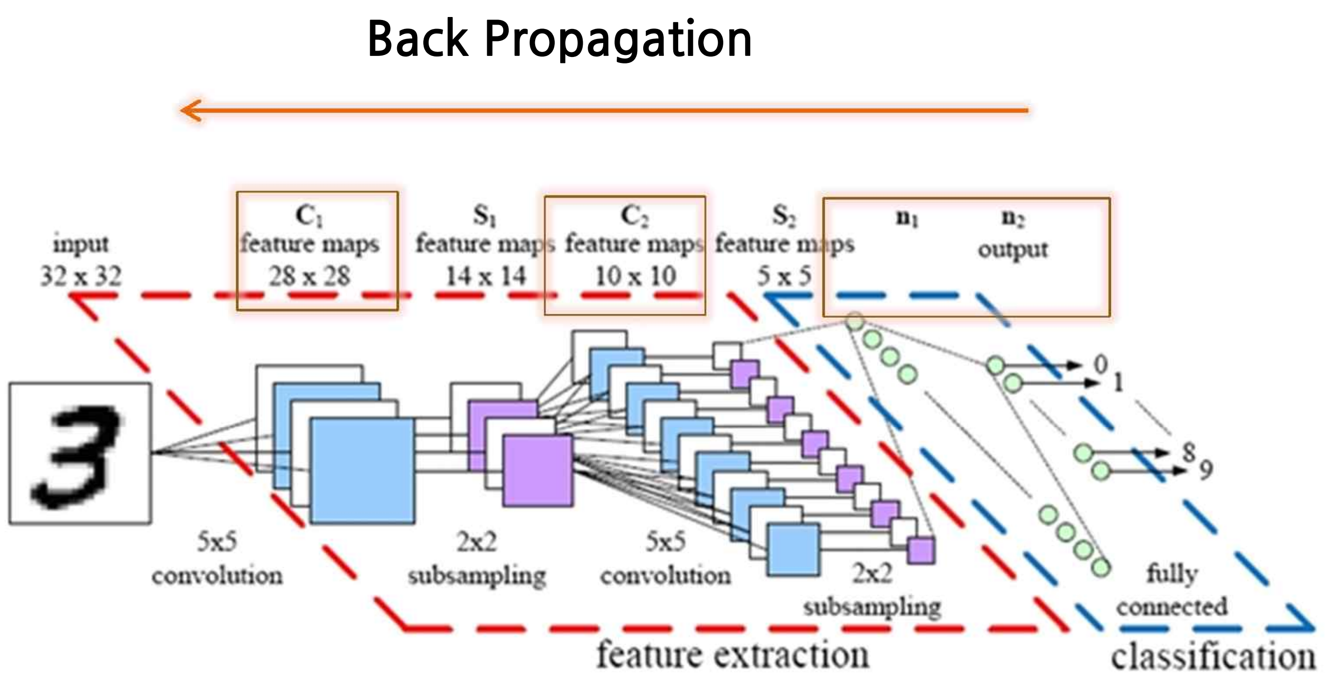


- Higher layer catches more specific, abstract patterns

- Lower layer catches more general patterns

Parameter Learning of CNN

- CNN is just another Neural Network with sparse connections
- Learning Algorithm:
  - Back Propagation on Convolution Layers and Fully-Connected Layers



Applications (Image Classification) (1/4)

Image Net Competition Ranking
(1000-class, 1 million images)

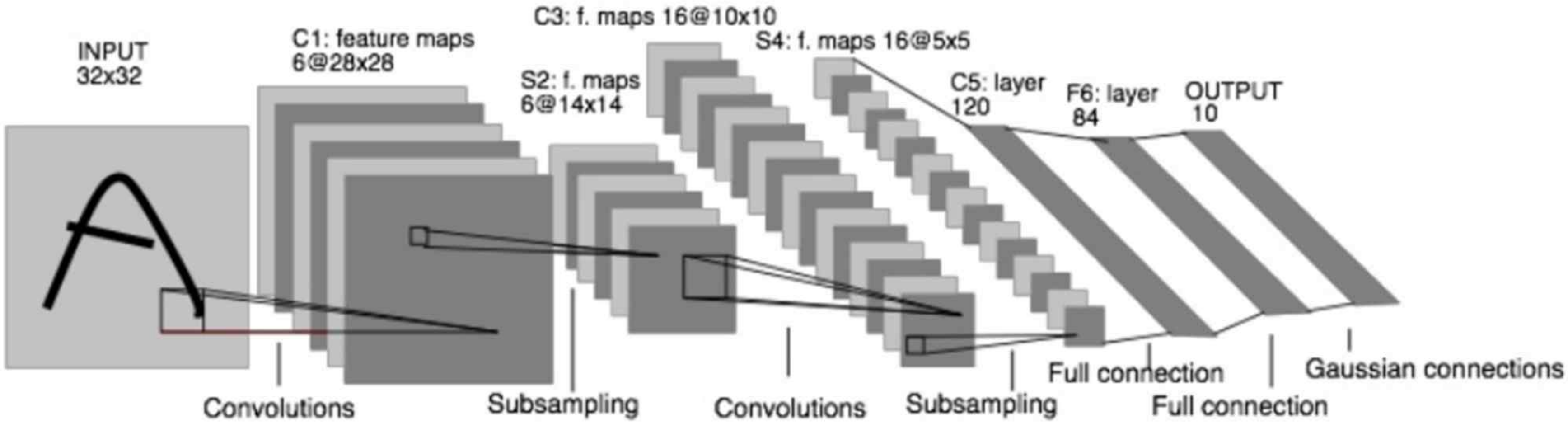
1. Clarifi (**0.117**): **Deep Convolutional Neural Networks (Zeiler)**
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional 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**

ALL CNN!!

From Kyunghyun Cho's dnn tutorial

Applications (Image Classification) (2/4)

- 1989, CNN for hand digit recognition, Yann LeCun

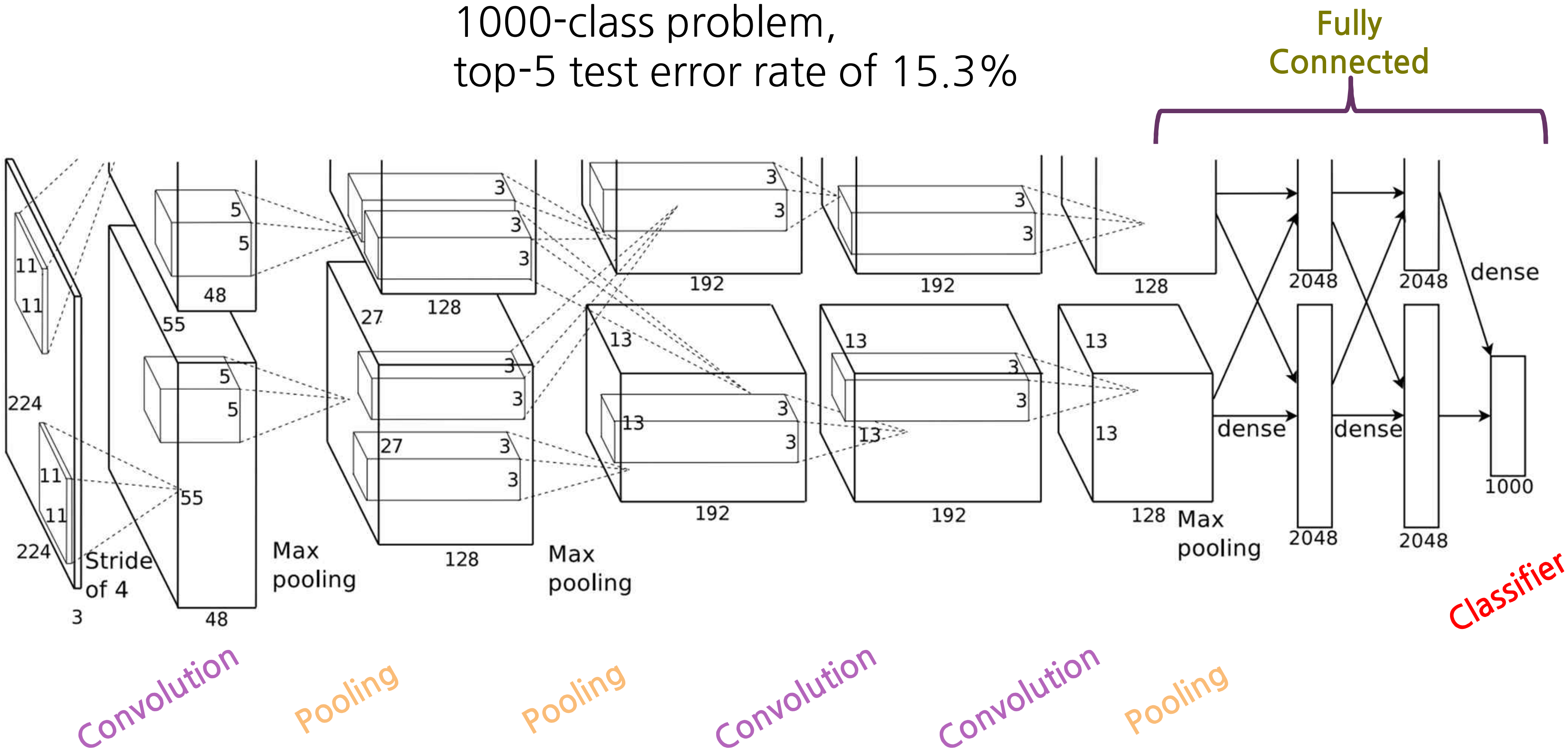


LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [Yann LeCun; LeNet]

Applications (Image Classification) (3/4)

- Krizhevsky et al.: the winner of ImageNet 2012 Competition

1000-class problem,
top-5 test error rate of 15.3%

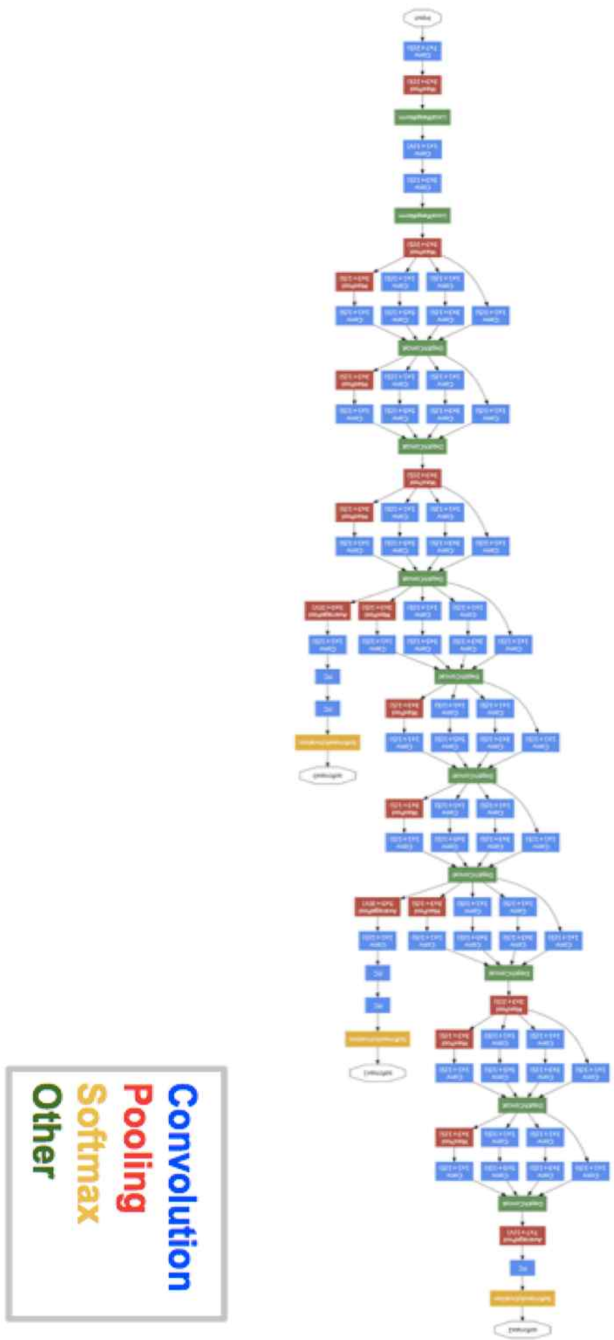


Applications
(Image Classification) (4/4)

- 2014 ILSVRC winner, ~6.6% Top 5 error

Example: VGG

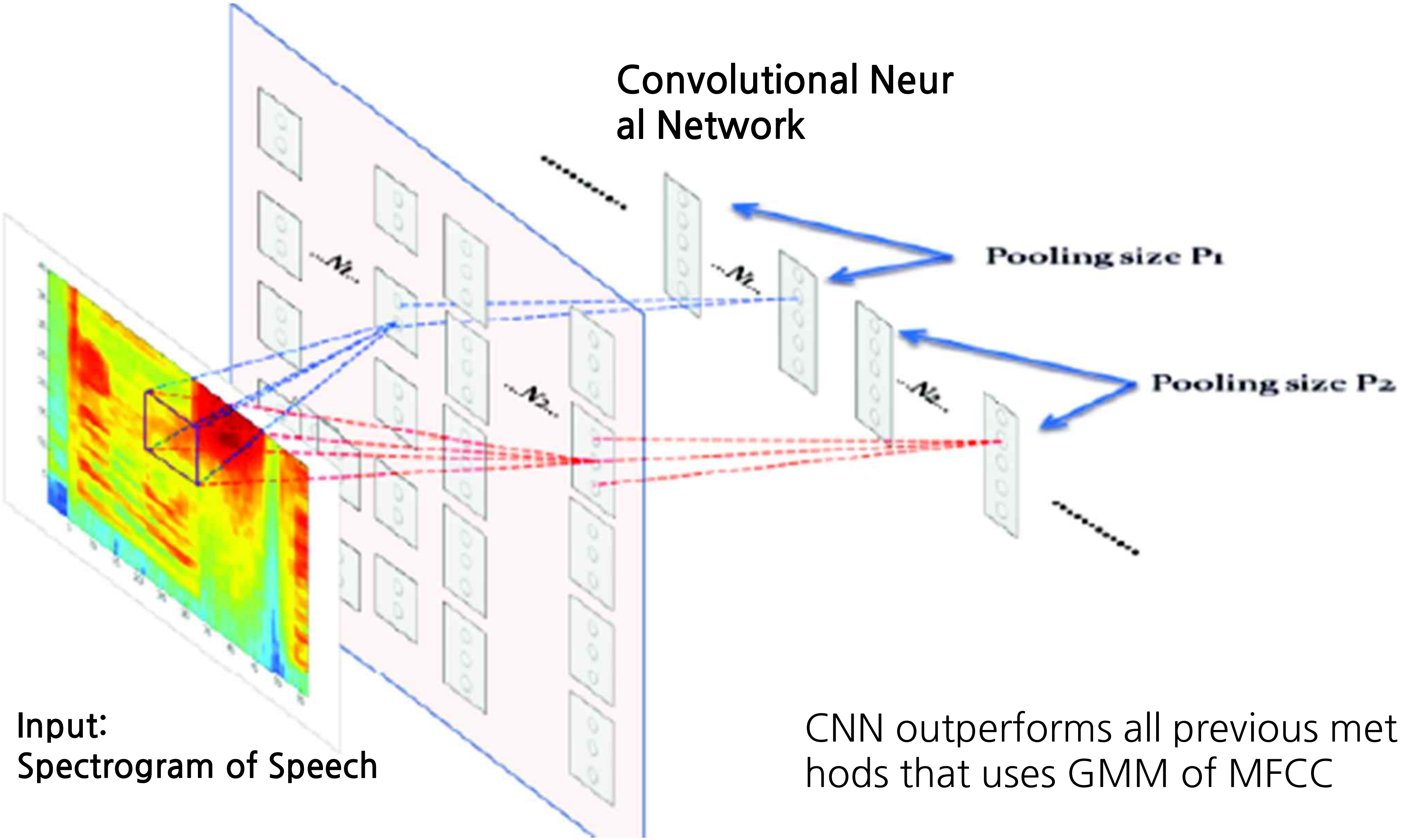
19 layers
3x3 convolution
pad 1
stride 1



http://image-net.org/challenges/LSVRC/2014/results

Application (Speech Recognition)

Convolutional Neur
al Network



Input:
Spectrogram of Speech

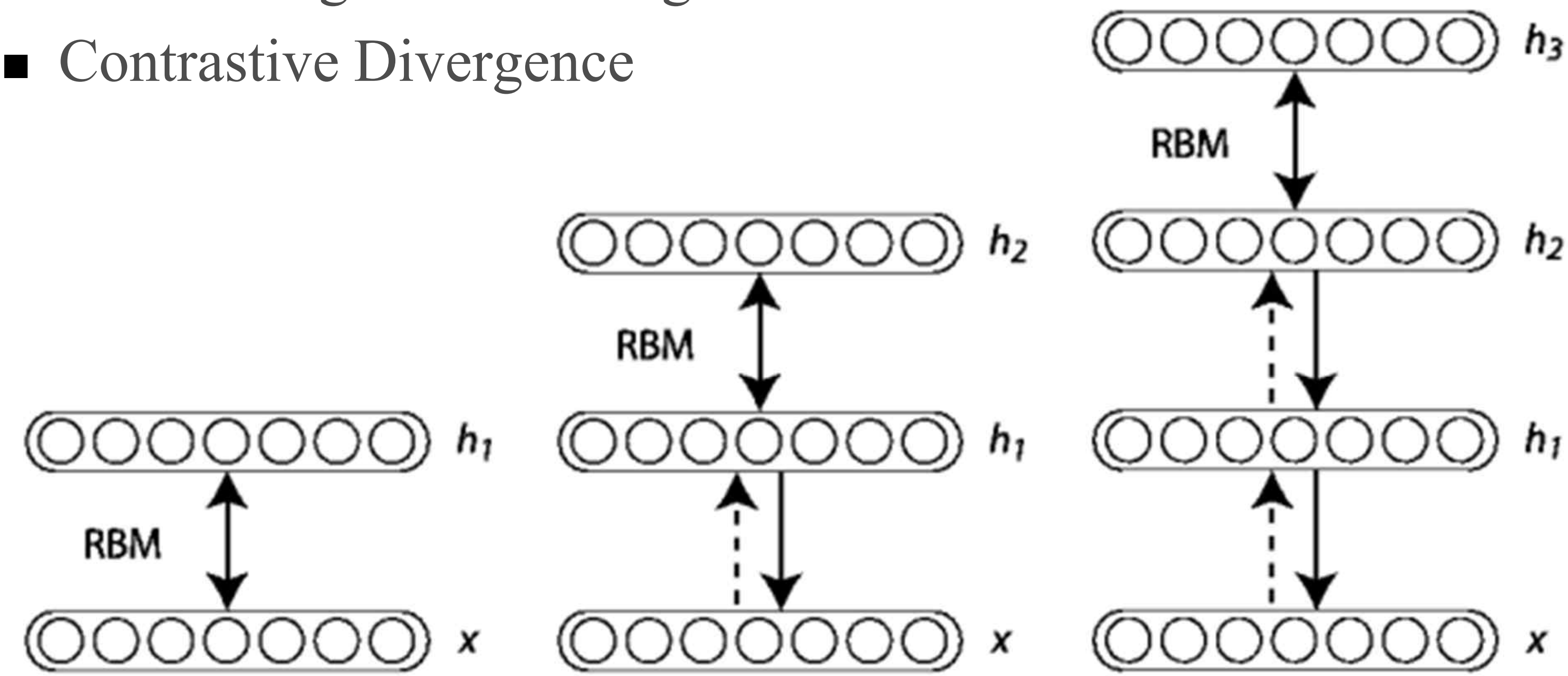
CNN outperforms all previous met
hods that uses GMM of MFCC

Deep Belief Networks

Slides by Jiseob Kim

Motivation

- 아이디어:
  - Greedy Layer-wise training
  - Pre-training + Fine tuning
  - Contrastive Divergence



Restricted Boltzmann Machine (RBM)

- Energy-Based Model

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

Joint (x, h) Probability

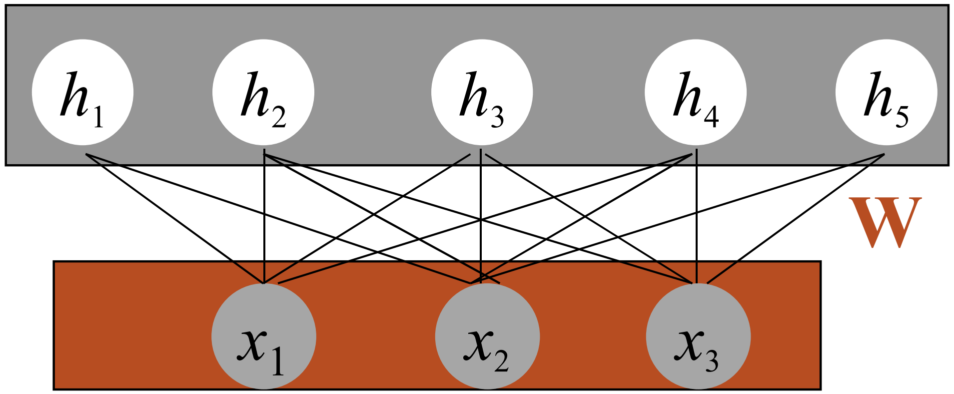
$$P(\mathrm{x}) = \frac{\sum_{\mathrm{h}} e^{-E(\mathrm{x},\mathrm{h})}}{\sum_{\mathrm{x},\mathrm{h}} e^{-E(\mathrm{x},\mathrm{h})}}$$

Marginal (x) Probability, or Likelihoo d

$$P(x_j = 1|\mathbf{h}) = \sigma(b_j + W'_{\bullet j} \cdot h)$$

$$P(h_i = 1|\mathbf{x}) = \sigma(c_i + W_{i\,.} \cdot x)$$

Conditional Probability



Remark:

- Conditional Independence

$$P(\mathbf{h}\,|\,\mathbf{x}) = \prod_i P(h_i\,|\,\mathbf{x})$$

$$P(\mathbf{x}\,|\,\mathbf{h}) = \prod_j P(x_j\,|\,\mathbf{h})$$

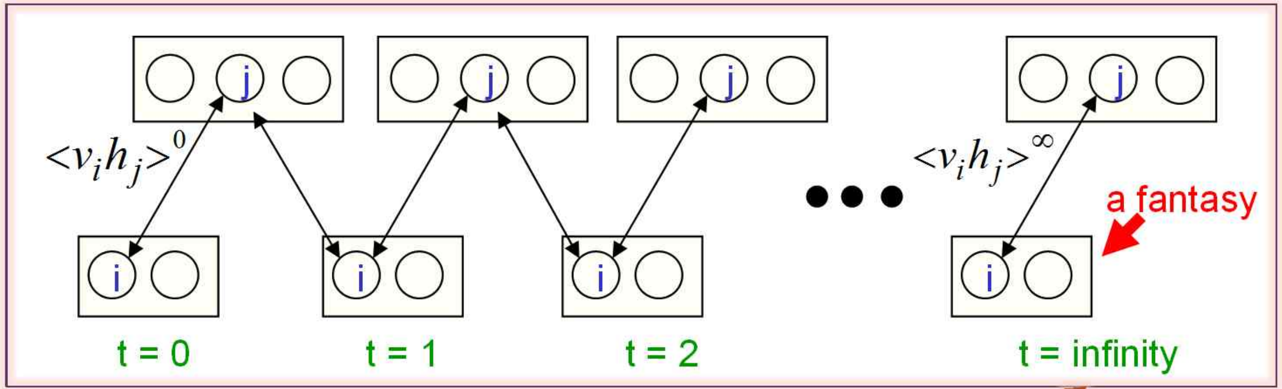
- Conditional Probability is the sa me as Neural Network

- Energy function
  - $\mathrm{E}(\mathbf{x}\,,\,\mathbf{h}) = \mathbf{b}'\,\mathbf{x} + \mathbf{c}'\,\mathbf{h} + \mathbf{h}'\,\mathbf{W}\mathbf{x}$

Unsupervised Learning of RBM

- Maximum Likelihood
  - Use Gradient Descent

$$L(X;\theta)=\frac{\sum_{h} e^{-E(x,h)}}{\sum_{x,h} e^{-E(x,h)}}$$



$$\frac{\partial L(X;\theta)}{\partial w_{ij}}=\int p(x,\theta)\frac{\partial \log f(x;\theta)}{\partial \theta}dx-\frac{1}{K}\sum_{k=1}^{K}\frac{\partial \log f(x^{(k)};\theta)}{\partial \theta}$$

$$=\left\langle x_i h_j \right\rangle_{p(x,\theta)}-\left\langle x_i h_j \right\rangle_{X}=\left\langle x_i h_j \right\rangle_{\infty}-\left\langle x_i h_j \right\rangle_{0}$$

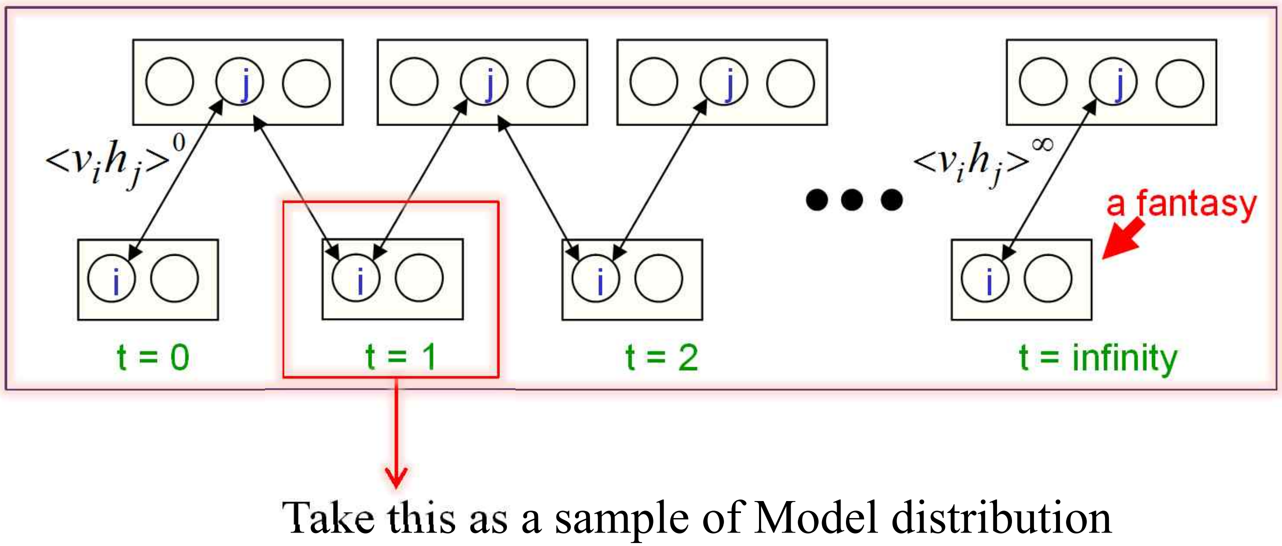
$$\approx\left\langle x_i h_j \right\rangle_{1}-\left\langle x_i h_j \right\rangle_{0}$$

Distribution of Dataset

Distribution of Model

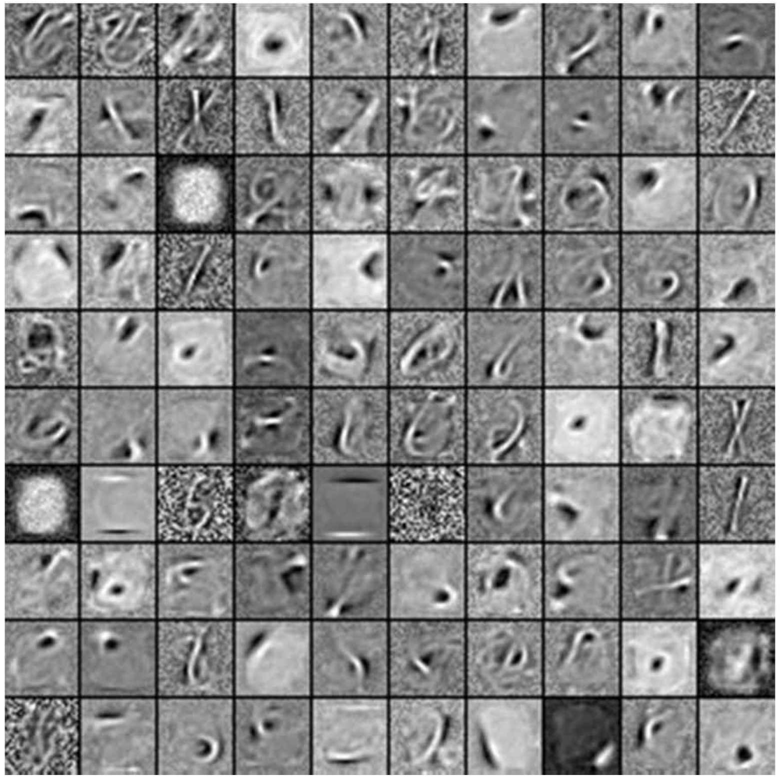
Contrastive Divergence (CD) Learning of RBM parameters

- k-Contrastive Divergence Trick
  - From the previous slide, to get distribution of model, we need to calculate many Gibbs sampling steps
  - And this is per a single parameter update
  - Therefore, we take the sample after only k-steps where in practice, k=1 is sufficient



Effect of Unsupervised Training

Unsupervised Training makes RBM successfully catch the essential patterns

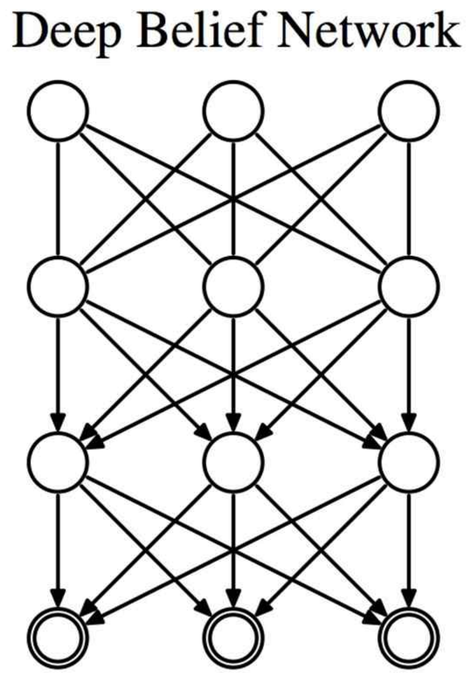


RBM trained on MNIST h and-written digit data:

Each cell shows the pattern e ach hidden node encodes

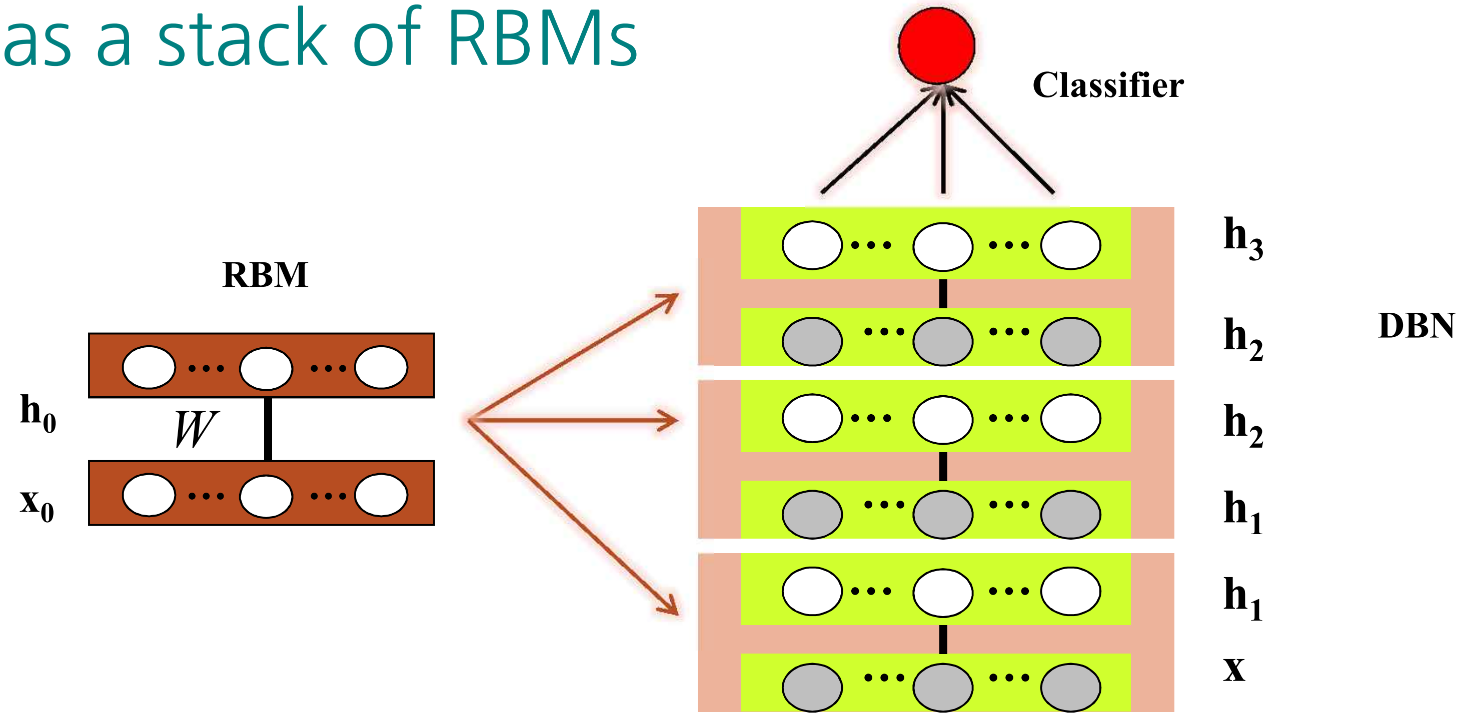
Deep Belief Network (DBN)

- Deep Belief Network (Deep Bayesian Network)
  - Bayesian Network that has similar structure to Neural Network
  - Generative model
  - Also, can be used as classifier (with additional classifier at top layer)
  - Resolves gradient vanishing by Pre-training
  - There are two modes (Classifier & Auto-Encoder), but we only consider Classifier here



Learning Algorithm of DBN

- DBN as a stack of RBMs

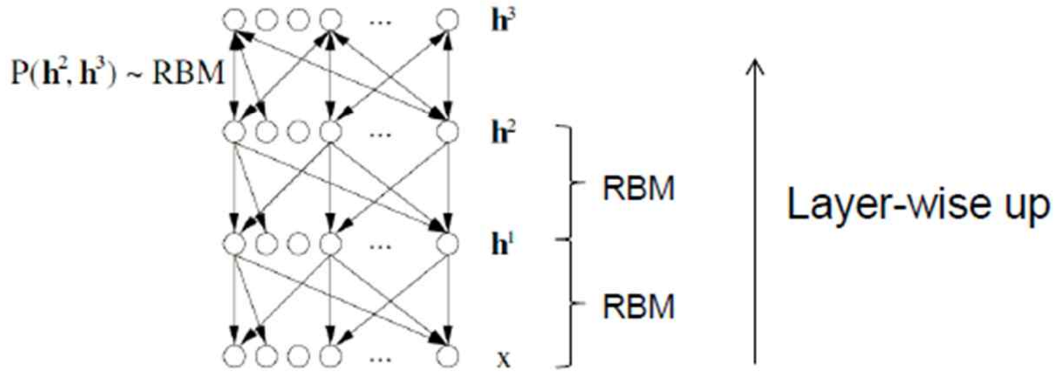


1. Regard each layer as RBM
2. Layer-wise Pre-train each RBM in Unsupervised way
3. Attach the classifier and Fine-tune the whole Network in Supervised way

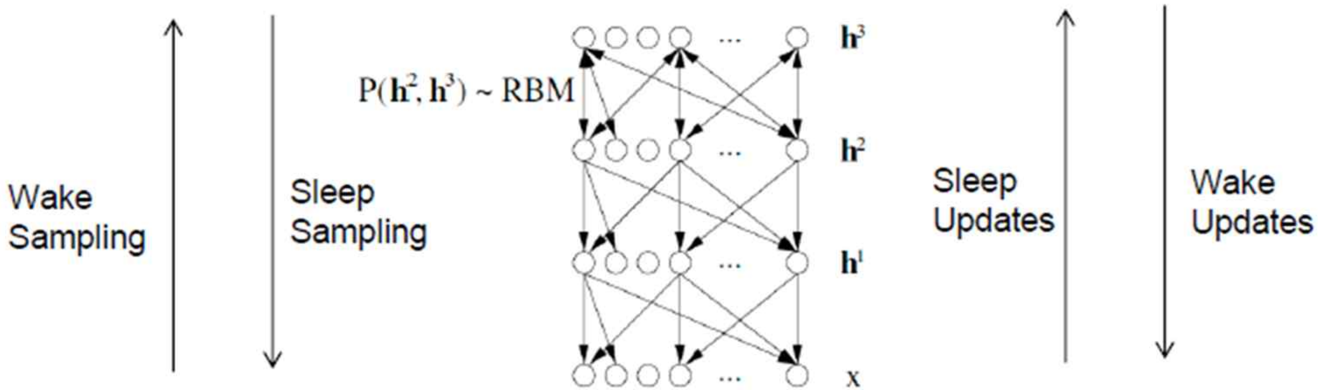
Viewing Learning as Wake-Sleep Algorithm

Training algorithms of DBNs and RBMs

- Training data: training set size: *n* (in millions), training data dimension: *d* (= number of observable nodes, in hundreds or thousands).
- RBM training as the basic module: maximum likelihood + stochastic gradient ascent
- Layer-wise greedy training:

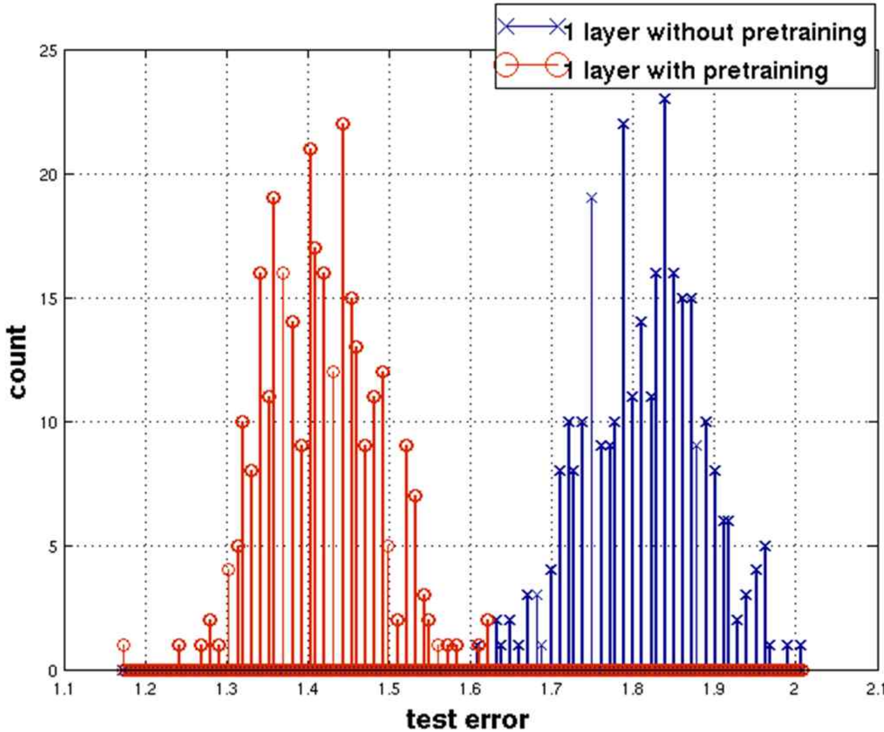


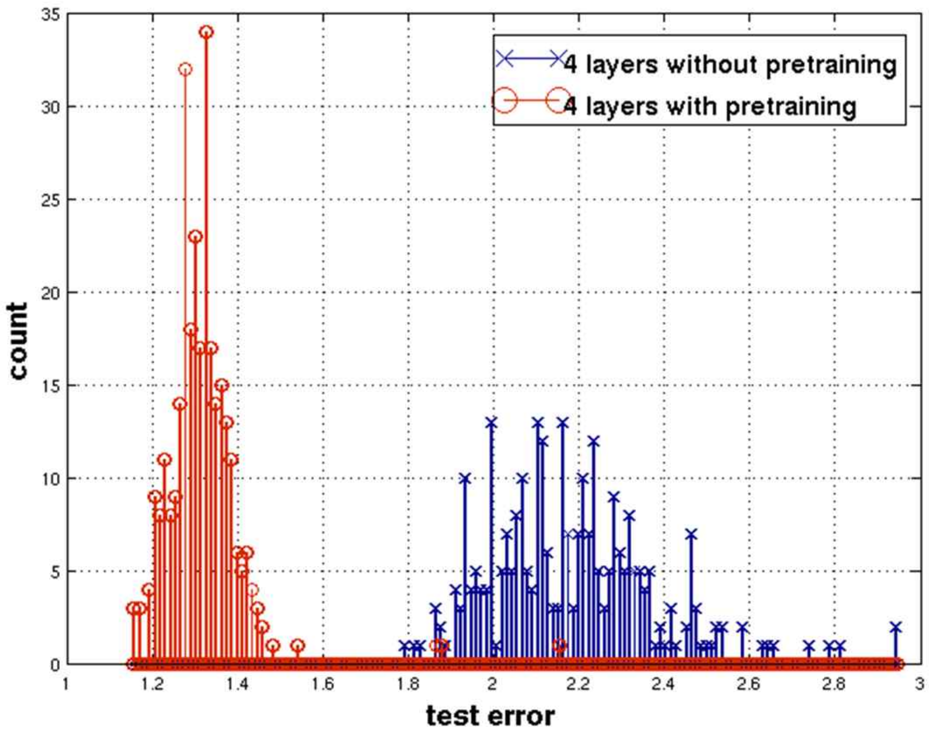
- Joint Wake-Sleep algorithm: Samplings+ Updates



Effect of Unsupervised Pre-Training in DBN (1/2)

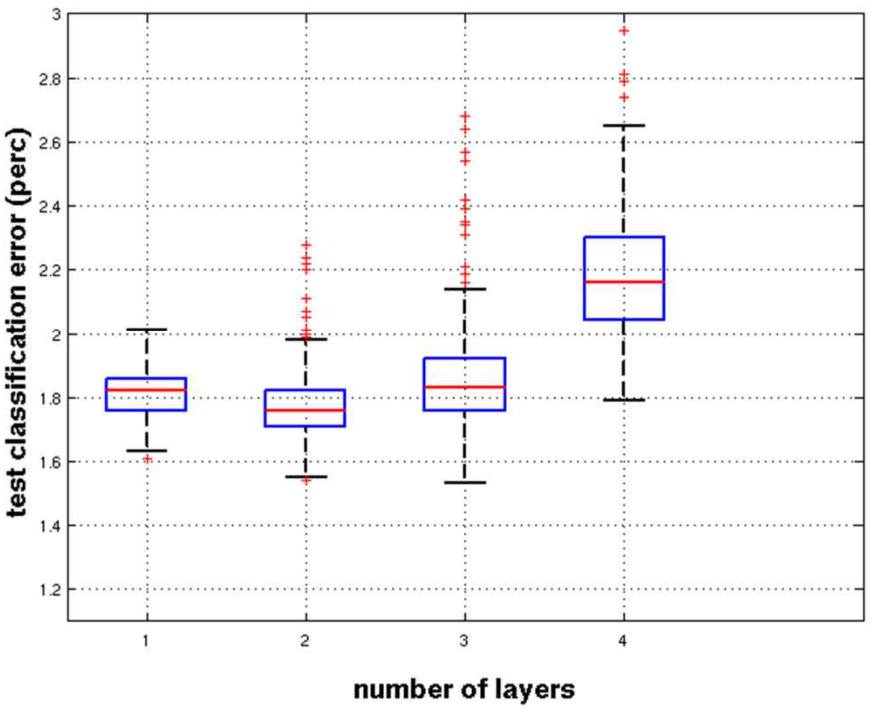
Erhan et. al. AISTATS'2009



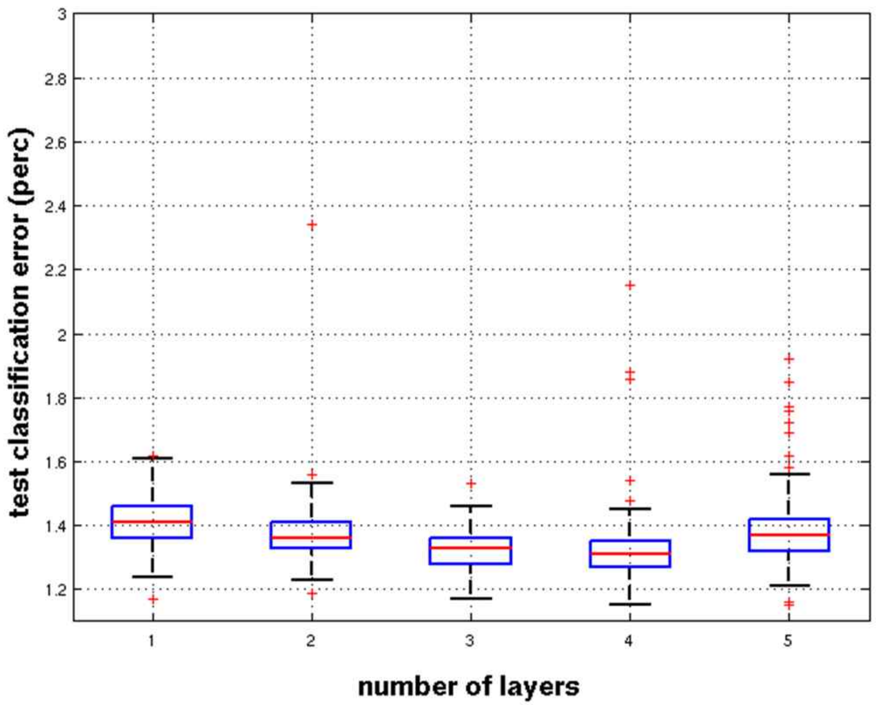


Effect of Unsupervised Pre-Training in DBN (2/2)

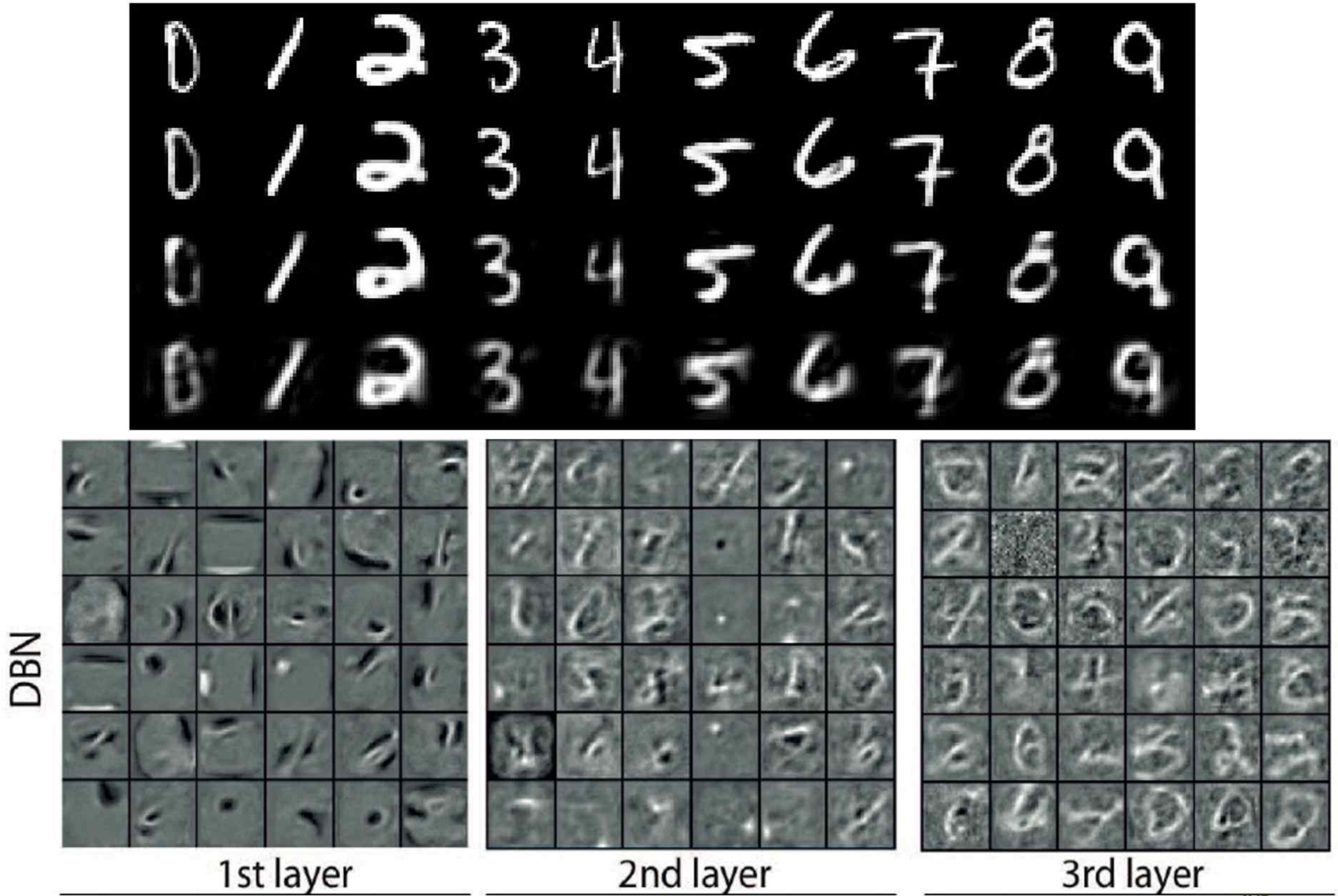
without pre-training



with pre-training

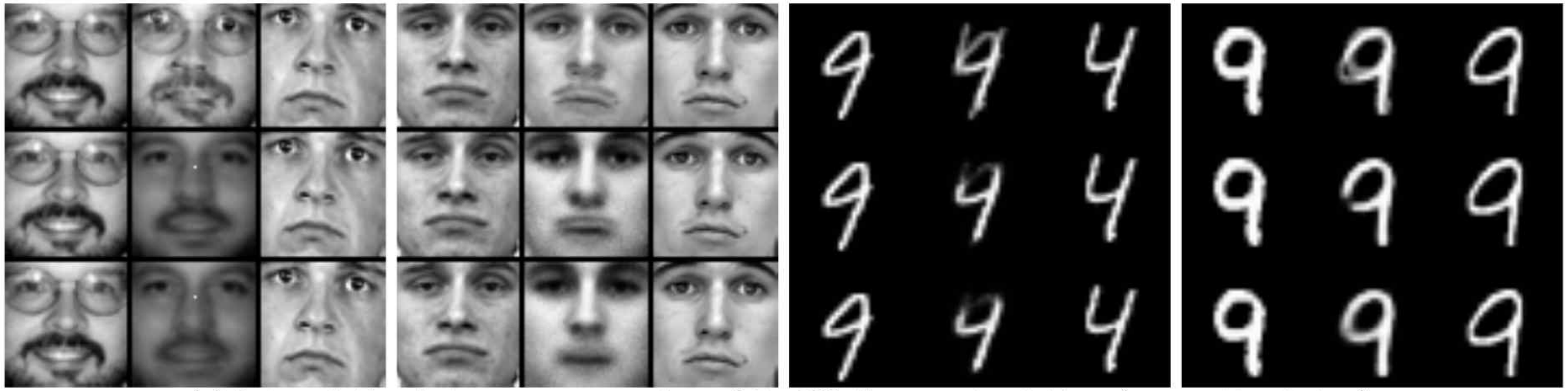


Internal Representation of DBN



Representation of Higher Layers

- Higher layers have more abstract representations
  - Interpolating between different images is not desirable in lower layers, but natural in higher layers



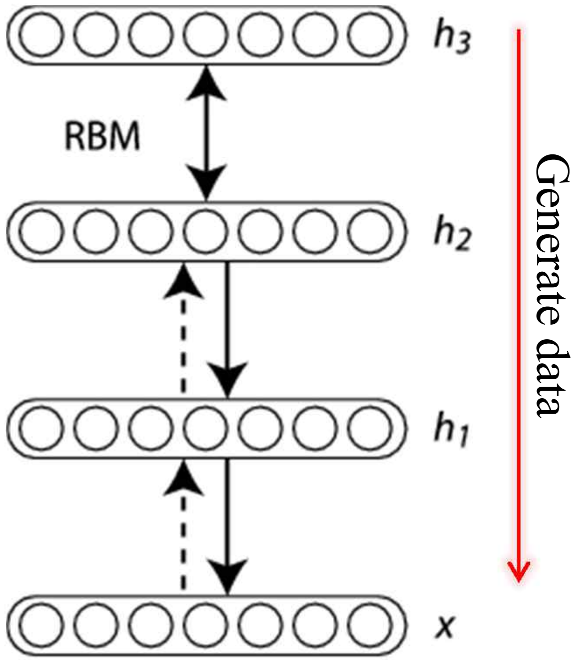
(a) Interpolating between an example and its 200-th nearest neighbor (see caption below).

(c) Sequences of points interpolated at different depths

Bengio et al., ICML 2013

Inference Algorithm of DBN

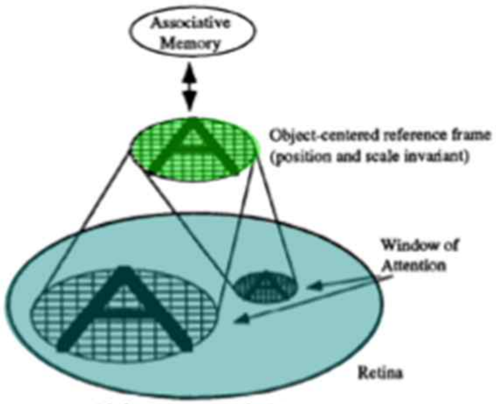
- As DBN is a generative model, we can also regenerate the data
  - From the top layer to the bottom, conduct Gibbs sampling to generate the data samples



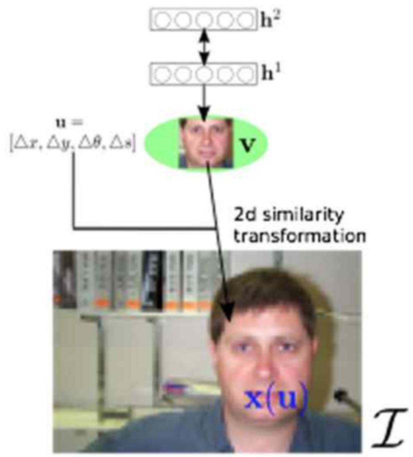
Lee, Ng et al., ICML 2009

Applications

- Nowadays, CNN outperforms DBN for Image or Speech data
- However, if there is no topological information, DBN is still a good choice
- Also, if the generative model is needed, DBN is used



Olshausen et al. 93



Our model

Generate Face patches
Tang, Srivastava, Salakhutdinov, NIPS 2014

Theano 실습

2016.3.30
Kmobile 딥러닝 1-day 워크샵

Eun-Sol Kim

Biointelligence Laboratory
Department of Computer Science and Engineering
Seoul National Univertisy

http://bi.snu.ac.kr

개요

- 심층 컨볼루션 신경망(Deep Convolutional Neural Network) 소개
- 다양한 딥러닝 프레임워크 비교 설명
- Theano의 특징 및 문법
  - Symbolic Variable
  - Shared Variable
- Theano 실습
  - Logistic Regression
  - Image classification
    - MNIST
- Theano Demo
  - Image classification with ImageNet dataset

실습 준비

- Kitematic 실행
- nzer0/theano_kit 다운
- CLI 창 열기
- docker run –it –p 8888:8888 nzer0/theano_kit
- cd ~
- python setup_nbserver.py
- 비밀번호 설정
- ipython notebook --profile=nbserver
- Virtual box->setting->network->port forwarding->8888:8888

GPU 컴퓨팅의 필요성

- 다루는 문제의 복잡도가 증가할수록 모델이 커지고 보다 많은 연산이 요구됨
- 컨볼루션 연산, 통합 연산은 모두 행렬 연산
- CUDA를 사용하여 컨볼루션 신경망을 구현한 경우 CPU에 비해 약 10배 이상 속도 향상
  - AlexNet으로 ILSVRC 데이터 학습시 CUDA를 이용한 경우 약 4~5일 정도 소모됨

다양한 딥러닝 프레임워크

	기반 언어	CNN	CUDA	Symbolic 연산	기타 모델 지원
Decaf / Caffe a Berkeley Vision Project	C++, Protobuf	O	O		
torch	Lua	O	O		RNN 및 다양한 Optimizer 제공. 기타 기본 ML 라이브러리 제공
theano	Python	O	O	O	RBM, DBN, AE, LSTM 등 대부분의 딥러닝 모델. 일반적인 확장 가능
Keras	Python, Theano	O	O	O	RBM, DBN, AE, LSTM, GRU 등 최신 모델. 다양한 Activation과 Optimizer 제공
MatConvNet	MATLAB	O	O		

다양한 딥러닝 프레임워크

	Caffe	Torch	Theano	TensorFlow
Language	C++, Python	Lua	Python	Python
Pretrained	Yes ++	Yes ++	Yes (Lasagne)	Inception
Multi-GPU: Data parallel	Yes	Yes cunn. DataParallelTable	Yes platoon	Yes
Multi-GPU: Model parallel	No	Yes fbcunn.ModelParallel	Experimental	Yes (best)
Readable source code	Yes (C++)	Yes (Lua)	No	No
Good at RNN	No	Mediocre	Yes	Yes (best)

http://cs231n.stanford.edu/slides/winter1516_lecture12.pdf

Theano

- 개요
  - LISA Lab에서 만든 Python 기반의 오픈소스 Package (http://deeplearning.net/software/theano/)
  - Symbolic 연산 철학

- 장점
  - Symbolic 연산 철학으로 간결하고 빠르게 모델 구현 가능
  - Symbolic 미분이 가능하므로 Back-Propagation 등을 직접 구현할 필요가 없음
  - 동일한 코드를 CPU와 GPU에서 모두 사용 가능
  - Python 기반이므로, numpy, scipy, matplotlib, ipython 등 다양한 python 패키지와의 연동 용이

- 단점
  - 에러 메세지가 번잡한 편
  - GPU연산의 경우 float만 지원

기본 Symbolic 연산

- 예제: y = 2*x^2+5*x 함수의 구현

일반적인 Python	Theano

def compute(x):
    y=2*x^2+5*x
    return y
compute(2)
	x = T.scalar() ← Symbolic 변수 정의
	y = 2*x^2+5*x ← Symbolic Expression
	compute = theano.function([x], y) ← 컴파일
	compute(2)

Symbolic 미분 연산

- 예제: y = 2*x^2+5*x 함수의 미분

일반적인 Python	Theano

def diff(x):
    y=4*x+5
    return y
diff(2)
	x = T.scalar()
	y = 2*x^2+5*x
	y_prime = T.grad(y, x) ← Symbolic 미분
	diff = theano.function([x], y_prime)
	diff(2)

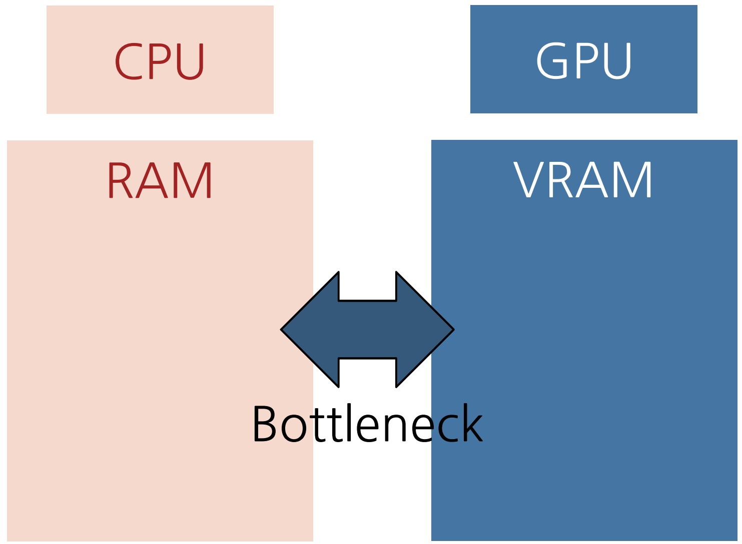
사람이 직접 미분한 식을 입력해야 함

Symbolic 미분을 통해 자동으로 도함수가 계산됨

복잡한 Back-Propagation 계산을 직접 구현할 필요가 없음

GPU 연산 관련 문법: shared

- 기능
  - VRAM과 RAM 사이의 데이터 전송
- shared_var = theano.shared(numpy_array)
- numpy_array = shared_var.get_value()

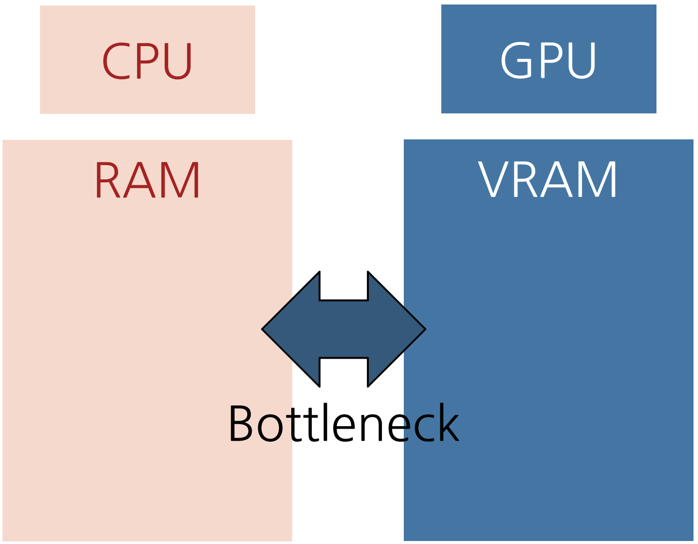


GPU 연산 관련 문법: Givens

- 기능: Symbolic 변수에 Shared 데이터를 대입

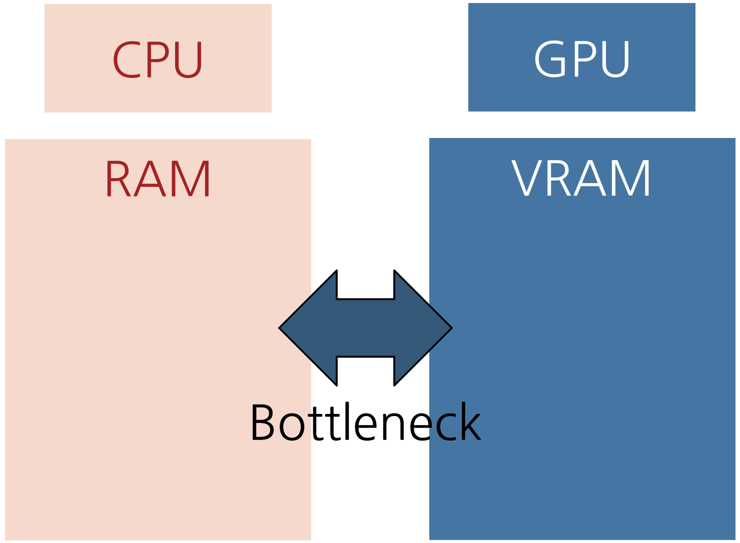
[예제] y = 2*x 일때, x에 10을 대입 계산하는 두 가지 구현 방법

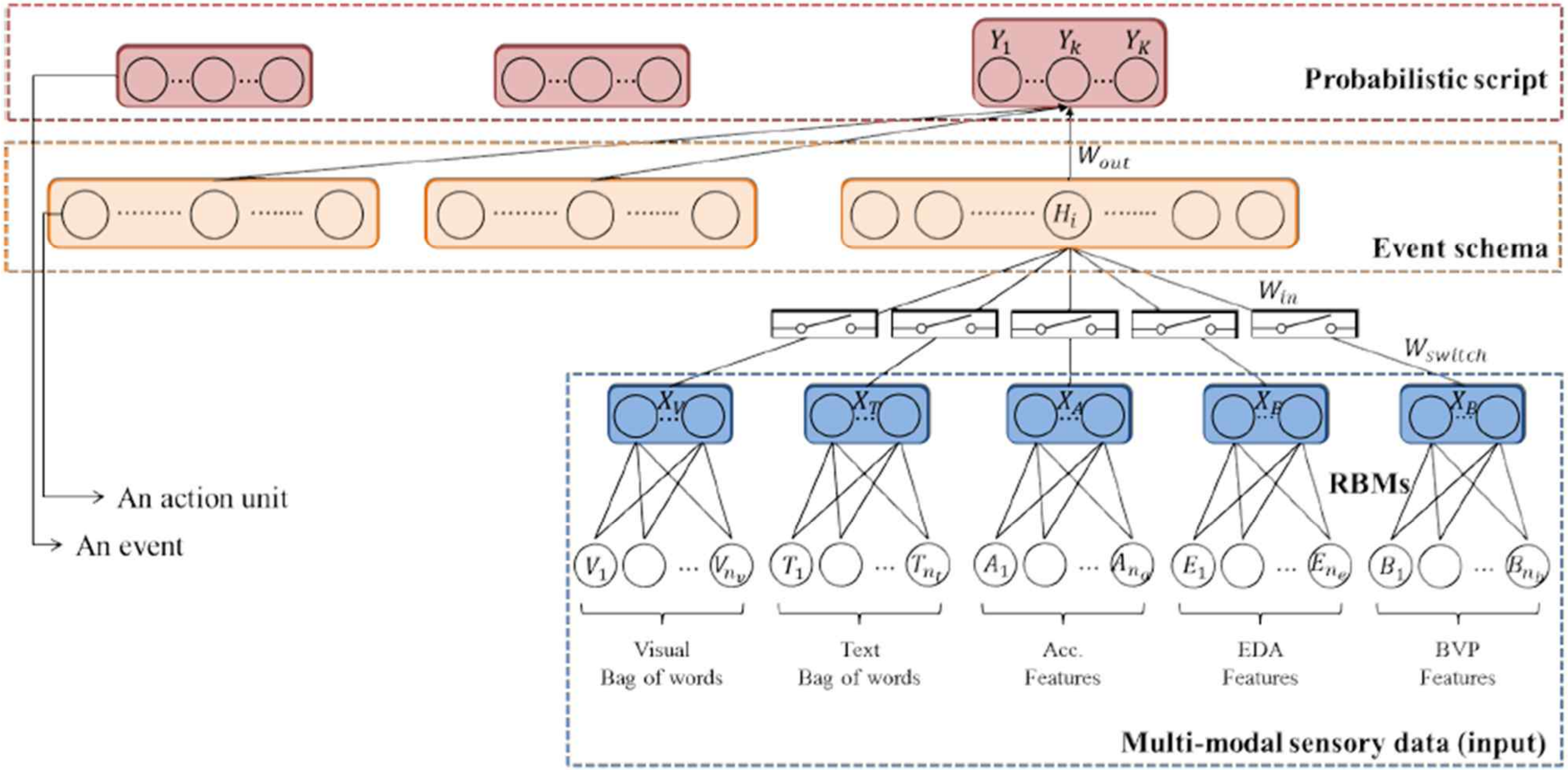
- 방법1)
  - compute = theano.function([x], 2*x);
  - compute(10) ← 실행시 RAM→VRAM→GPU연산
- 방법2)
  - x_value = theano.shared(10)
  - compute = theano.function([], 2*x, givens=[(x,x_value)])
  - compute() ← 실행시 VRAM→GPU연산



GPU 연산 관련 문법: updates

- 기능: GPU연산 결과를 이용해 Shared 데이터를 수정
- x_val = theano.shared(0)
- increase = theano.function([], x_val, updates=(x_val, x_val+1))
- increase() ← 실행시 RAM을 거치지 않고, GPU내에서 계속 x_val을 1씩 증가시킴





Probabilistic script
Y_1
Y_k
Y_K
W_{out}
H_i
Event schema
W_{in}
W_{switch}
X_V
X_T
X_A
X_E
X_B
RBMs
An action unit
An event
V_1 ... V_{n_v}
T_1 ... T_{n_t}
A_1 ... A_{n_a}
E_1 ... E_{n_e}
B_1 ... B_{n_b}
Visual Bag of words
Text Bag of words
Acc. Features
EDA Features
BVP Features
Multi-modal sensory data (input)

$$y_{n,k} = \sigma\left(\sum_{m=1}^{M} s_m (\mathbf{w}_m^k)^\top \mathbf{x}_m^n + b_m^k\right)$$

$$s_m = \sigma\left((\mathbf{u}_m)^\top \mathbf{X} + a_m\right)$$

$$lnE = -\sum_{n=1}^{N}\sum_{k=1}^{K}\{t_{n,k} ln y_{n,k} + (1-t_{n,k}) ln(1-y_{n,k})\}$$

$$\mathbf{w}_m^k \leftarrow \mathbf{w}_m^k - \delta \cdot \frac{\partial lnE}{\partial \mathbf{w}_m^k}$$

$$\mathbf{u}_m \leftarrow \mathbf{u}_m - \delta \cdot \frac{\partial lnE}{\partial \mathbf{u}_m}$$

$$b_m^k \leftarrow b_m^k - \delta \cdot \frac{\partial lnE}{\partial b_m^k}$$

$$a_m \leftarrow a_m - \delta \cdot \frac{\partial lnE}{\partial a_m}$$

$$\begin{aligned}\frac{\partial lnE}{\partial \mathbf{w}_m^k} &= \frac{\partial lnE}{\partial y_{n,k}}\frac{\partial y_{n,k}}{\partial \mathbf{w}_m^k} \\ &= \left(-\frac{t_{n,k}}{y_{n,k}} + \frac{1-t_{n,k}}{1-y_{n,k}}\right)(y_{n,k}(1-y_{n,k})s_m \mathbf{x}_m^n) \\ &= (-t_{n,k}(1-y_{n,k}) + (1-t_{n,k})y_{n,k})\, s_m \mathbf{x}_m^n \\ &= (y_{n,k}-t_{n,k}) s_m \mathbf{x}_m^n\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{\partial lnE}{\partial \mathbf{u}_m} &= \frac{\partial lnE}{\partial y_{n,k}}\frac{\partial y_{n,k}}{\partial s_m}\frac{\partial s_m}{\partial \mathbf{u}_m} \\ &= (y_{n,k}-t_{n,k}) s_m (1-s_m)\mathbf{X}\sum_{k=1}^{K}(\mathbf{w}_m^k)^\top \mathbf{x}_m^n\end{aligned} \tag{5}$$

$$\begin{aligned}\frac{\partial lnE}{\partial b_m^k} &= \frac{\partial lnE}{\partial y_{n,k}}\frac{\partial y_{n,k}}{\partial b_m^k} \\ &= (y_{n,k}-t_{n,k})\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial lnE}{\partial a_m} &= \frac{\partial lnE}{\partial y_{n,k}}\frac{\partial y_{n,k}}{\partial s_m}\frac{\partial s_m}{\partial a_m} \\ &= (y_{n,k}-t_{n,k}) s_m (1-s_m)\sum_{k=1}^{K}(\mathbf{w}_m^k)^\top \mathbf{x}_m^n\end{aligned} \tag{7}$$

```python
switched = True

W_sw = theano.shared(0.001*np.asarray(rng.uniform(low=-4*np.sqrt(6. / (n_switch + n_con_input)),
                                                   high=4*np.sqrt(6. / (n_switch + n_con_input)),
                                                   size=(n_con_input, n_switch)), dtype=floatX),
                     name='W_sw', borrow=True)

b_sw = theano.shared(np.zeros(n_switch, dtype=floatX), name='b_sw', borrow=True)


W = theano.shared(0.001*np.asarray(rng.uniform(low=-4*np.sqrt(6. / (n_output + n_con_input)),
                                                   high=4*np.sqrt(6. / (n_output + n_con_input)),
                                                   size=(n_con_input, n_output)), dtype=floatX),
                     borrow=True)

b = theano.shared(np.zeros(n_output, dtype=floatX), name='b', borrow=True)

switch = T.nnet.sigmoid(T.dot(inp, W_sw)+b_sw) # (n_data x n_grps)
sw_indicator = T.dot(switch, grp_indicator) # (n_data x n_con_input)
sw_inp = inp * sw_indicator # (n_data x n_con_input)

if switched:
    output = T.nnet.sigmoid(T.dot(sw_inp, W)+b)
    #output = T.nnet.sigmoid(T.dot(hid_value, W) + b) # (n_data x n_output)
    params = [W_sw, b_sw, W, b]
else:
    output = T.nnet.sigmoid(T.dot(inp, W)+b)
    #output = T.nnet.sigmoid(T.dot(hid_value, W)+b) # (n_data x n_output)
    params = [W, b]
```

```python
lbls = T.matrix('lbls')
cross_entropy = -T.sum(lbls*T.log(output) + (1-lbls)*T.log(1-output), axis=1)
cost = T.mean(cross_entropy)
gparams = T.grad(cost, params)
updates = []
for param, gparam in zip(params, gparams):
    updates.append((param, param - learning_rate * gparam))
```

실습