

Theano 딥러닝 실습 1

2016.4.15

인지과학산업협회 튜토리얼 13 딥러닝 실습

Eun-Sol Kim

Biointelligence Laboratory
Department of Computer Science and Engineering
Seoul National Univertisy

- 실습 코드 다운 받아 오신 분들
(최신 버전으로 업데이트)

- git pull

- 실습 코드 안 받아 오신 분들

- git clone https://github.com/bi-lab/deeplearning_tutorial.git

개요

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

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		
	Lua	O	O		RNN 및 다양한 Optimizer 제공. 기타 기본 ML 라이브러리 제공
	Python	O	O	O	RBM, DBN, AE, LSTM 등 대부분의 딥러닝 모델. 일반적인 확장 가능
	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)

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: $y = 2x^2 + 5x$ 함수의 구현

일반적인 Python	Theano
<pre>def compute(x): y=2*x^2+5*x return y compute(2)</pre>	<pre>x = T.scalar() y = 2*x^2+5*x compute = theano.function([x], y) compute(2)</pre>

Symbolic 미분 연산

- 예제: $y = 2x^2 + 5x$ 함수의 미분

일반적인 Python	Theano
<pre>def diff(x): y=4*x+5 return y diff(2)</pre>	<pre>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)</pre>

사람이 직접 미분한
식을 입력해야 함

Symbolic 미분을 통해 자동으로 도함수가 계산됨



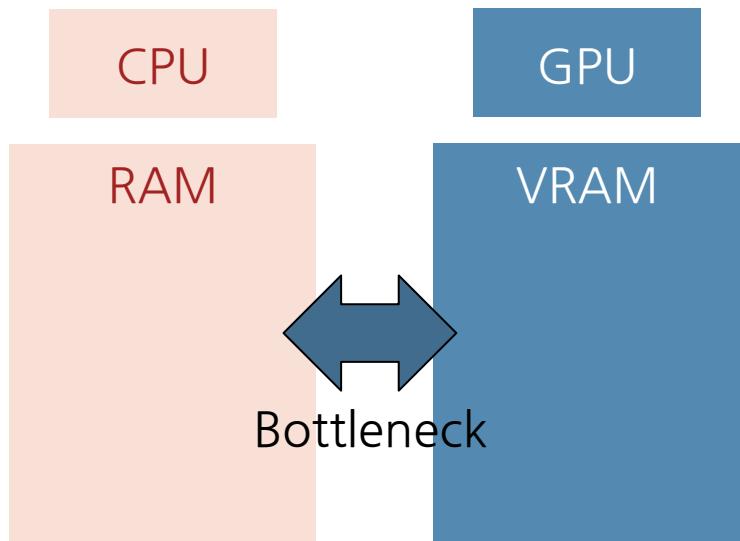
복잡한 Back-Propagation 계산을 직접 구현할 필요가 없음

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, x 에 10을 대입 계산하는 두 가지 구현 방법

- 방법1)

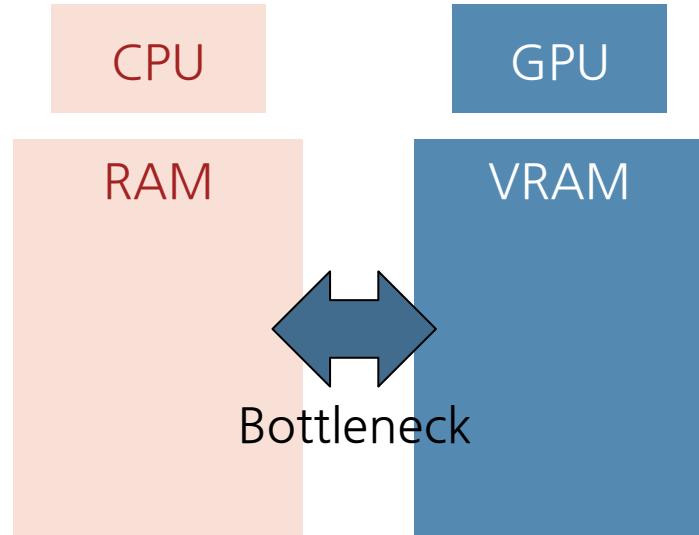
- `compute = theano.function([x], 2*x);`
- `compute(10)`

← 실행시
RAM → VRAM → GPU연산

- 방법2)

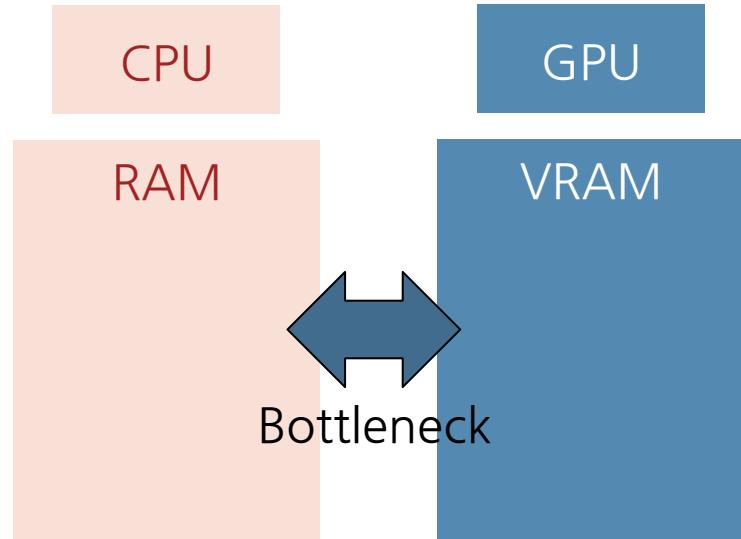
- `x_value = theano.shared(10)`
- `compute = theano.function([], 2*x,`
- `givens=[(x,x_value)])`
- `compute()`

← 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

Theano Basic 실습

Neural Network & Convolutional Neural Network

History of Neural Network Research

Neural network
Back propagation



Deep belief net
Science



2006

Speech

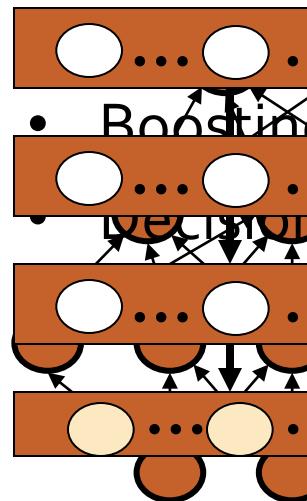
The New York Times
Google Hong Kong

IMAGENET



deep learning results

- Unsupervised & layer-wised pre-training



Rank	Name	Error rate	Description	Training (normal problems)
1	U. Toronto	0.15315	Deep Conv Net	P, HOG
2	U. Tokyo	0.26172	Hand-crafted features and learning models.	lectures
3	U. Oxford	0.26979		
4	Xerox/INRIA	0.27058	Bottleneck.	

1 frame from each (200x200)



Deep Networks Advance State-of-Art in Speech

Deep Learning leads to breakthrough in speech recognition at MSR.

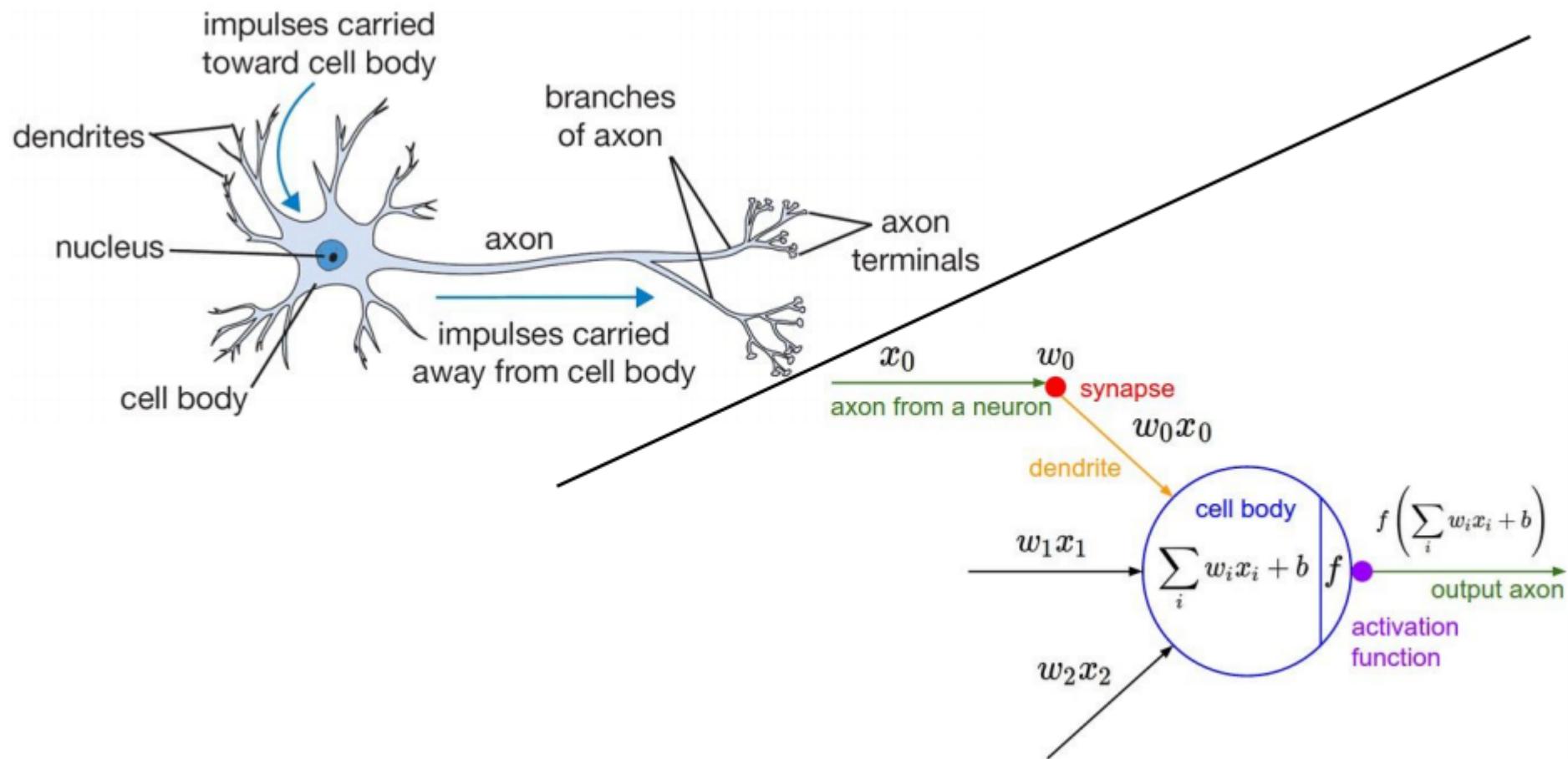
(2 GPU)



Classification Problem

- 데이터 x 가 주어졌을 때 해당되는 레이블 y 를 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)

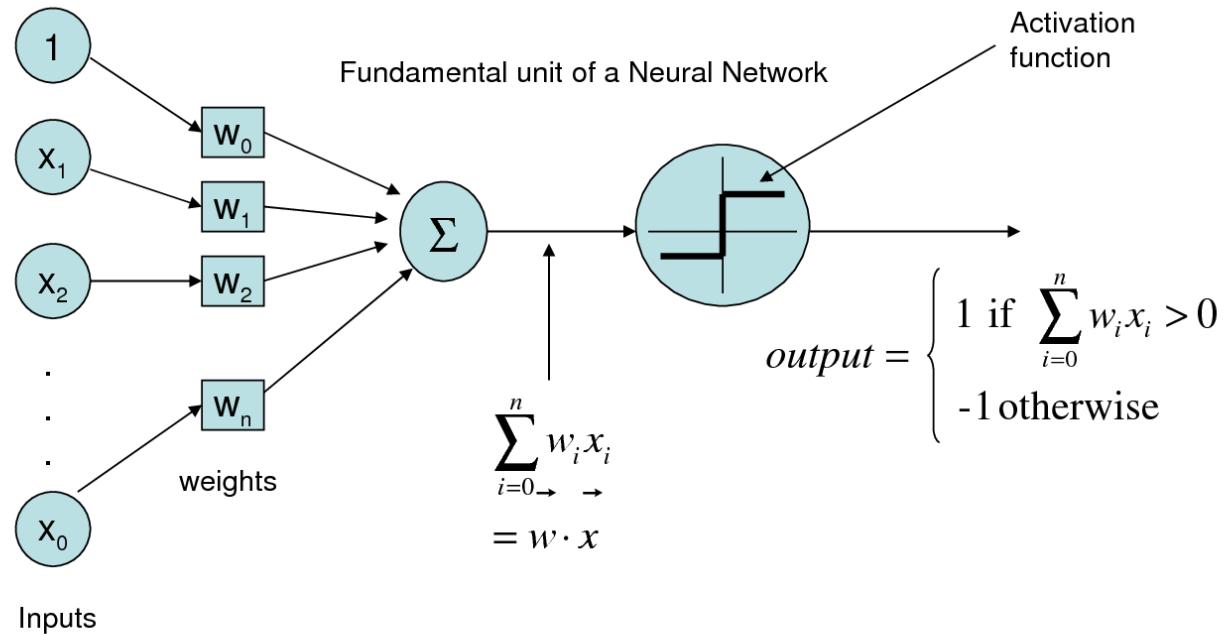


http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

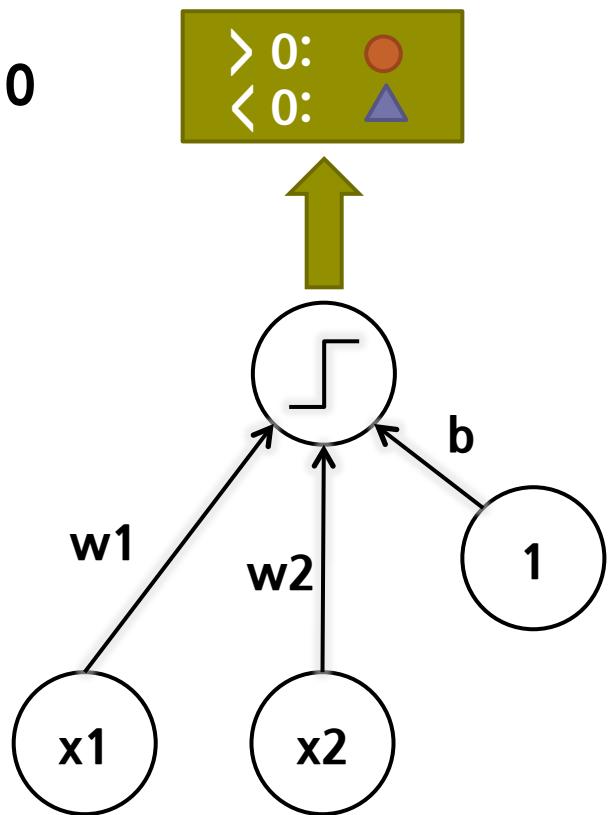
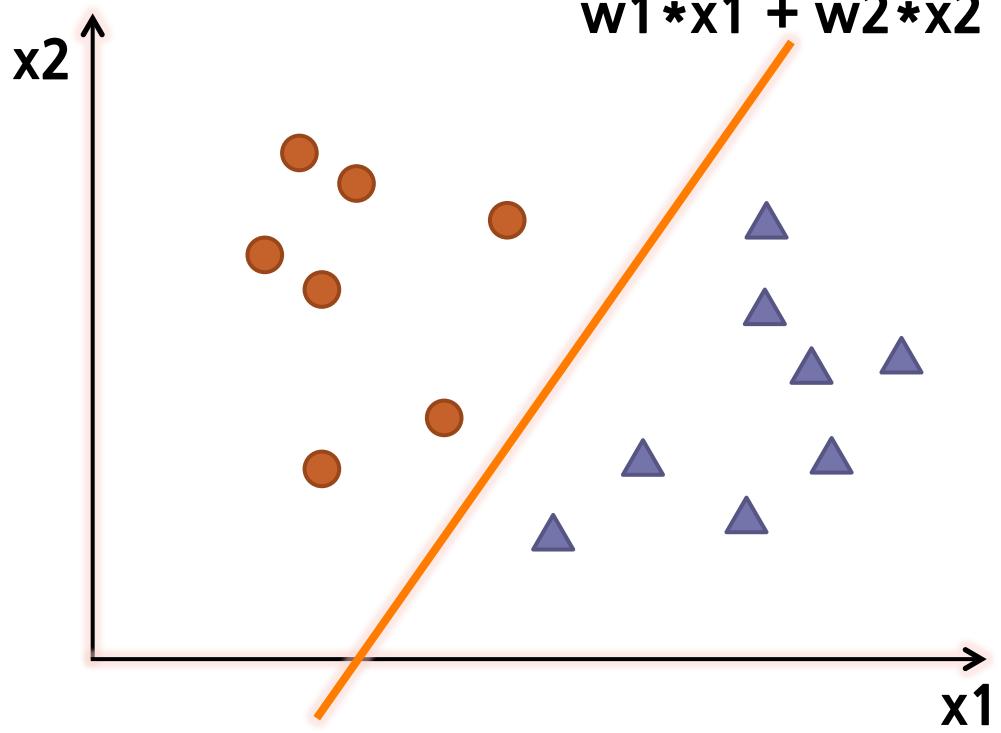
Perceptron (2/3)

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,

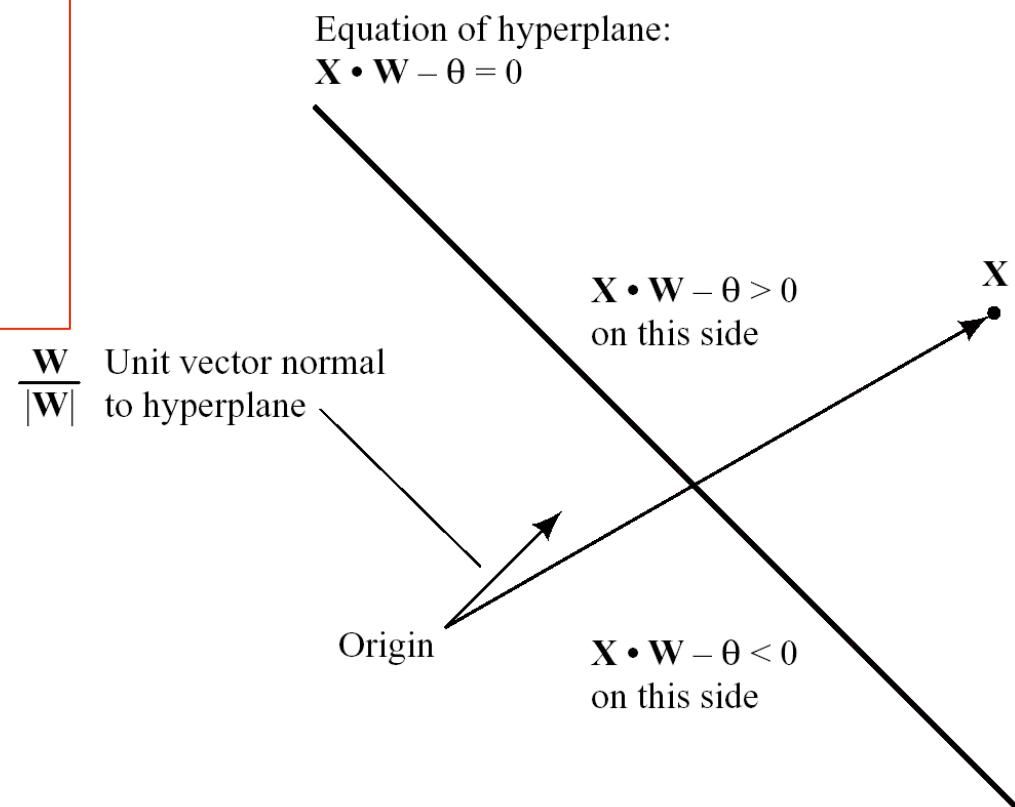
If $x \in N$ and $w \cdot x \geq 0$ go to subtract.

add:

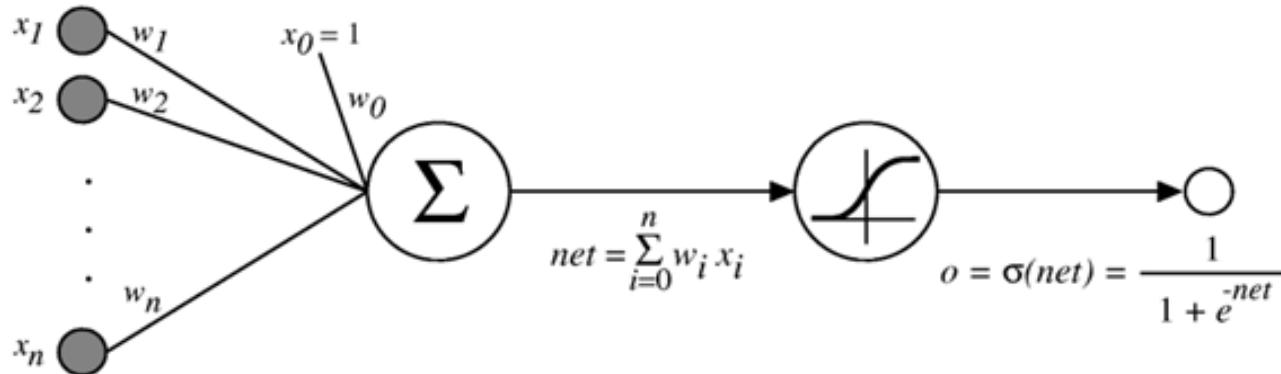
Set $w = w + x$, goto test

subtract:

Set $w = w - x$, goto test

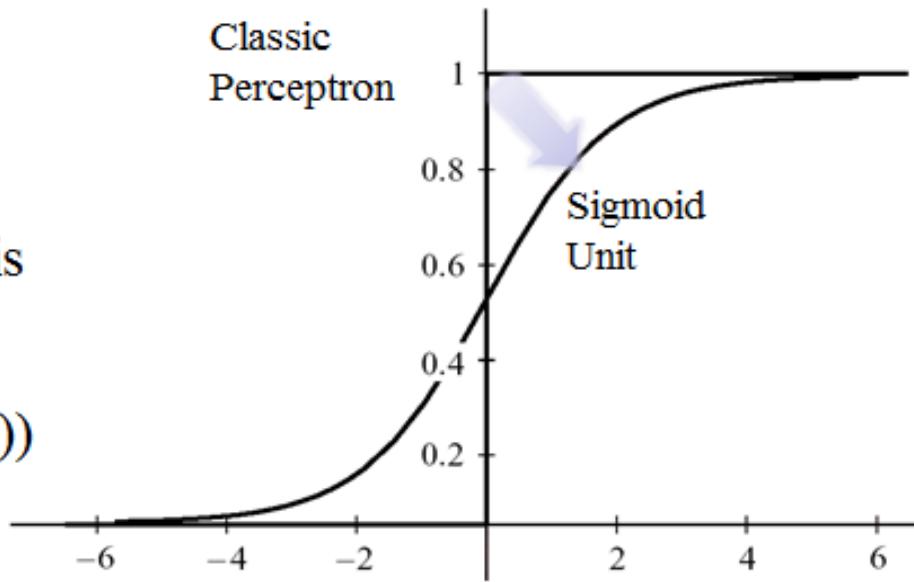


Sigmoid Unit



Sigmoid function is
Differentiable

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



Learning Algorithm of Sigmoid Unit

■ Loss Function

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

Target Unit Output

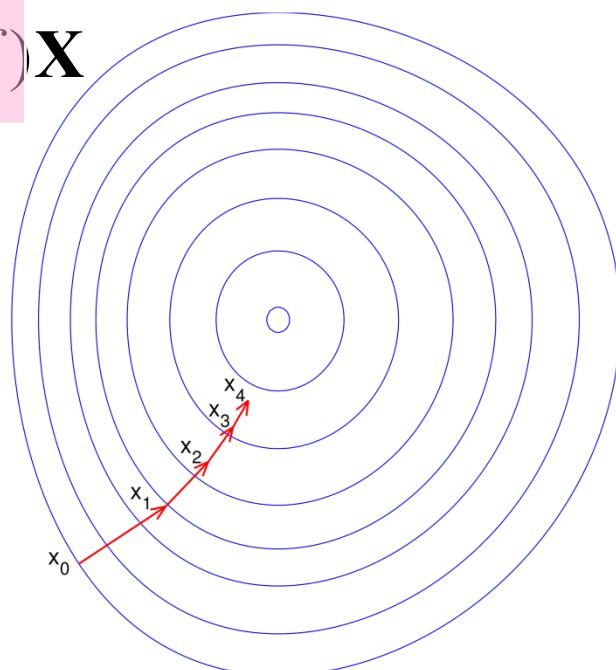
■ Gradient Descent Update

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

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

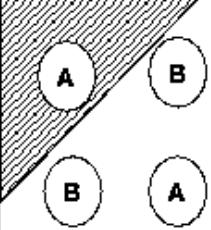
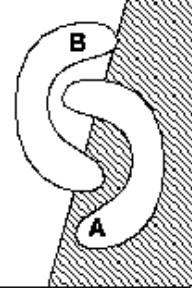
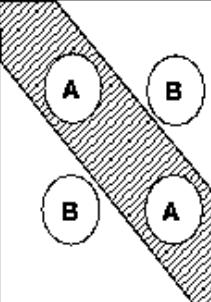
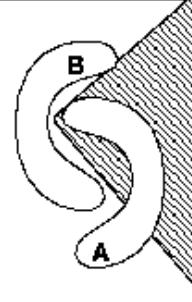
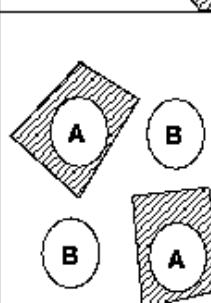
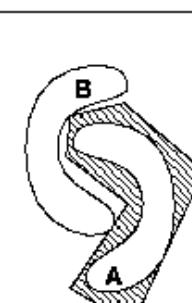
$$f'(s) = f(s)(1 - f(s))$$

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

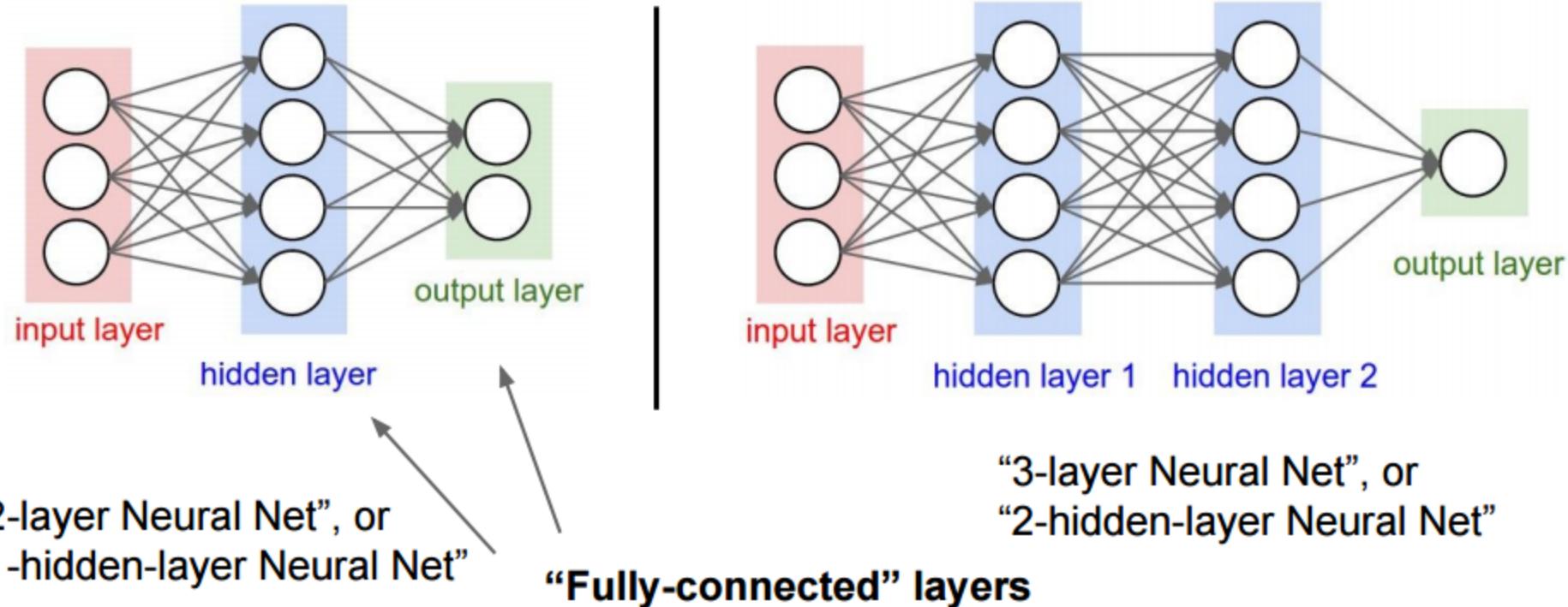


Need for Multiple Units and Multiple Layers

- Multiple boundaries are needed (e.g. XOR problem)
→ Multiple Units
- More complex regions are needed (e.g. Polygons)
→ Multiple Layers

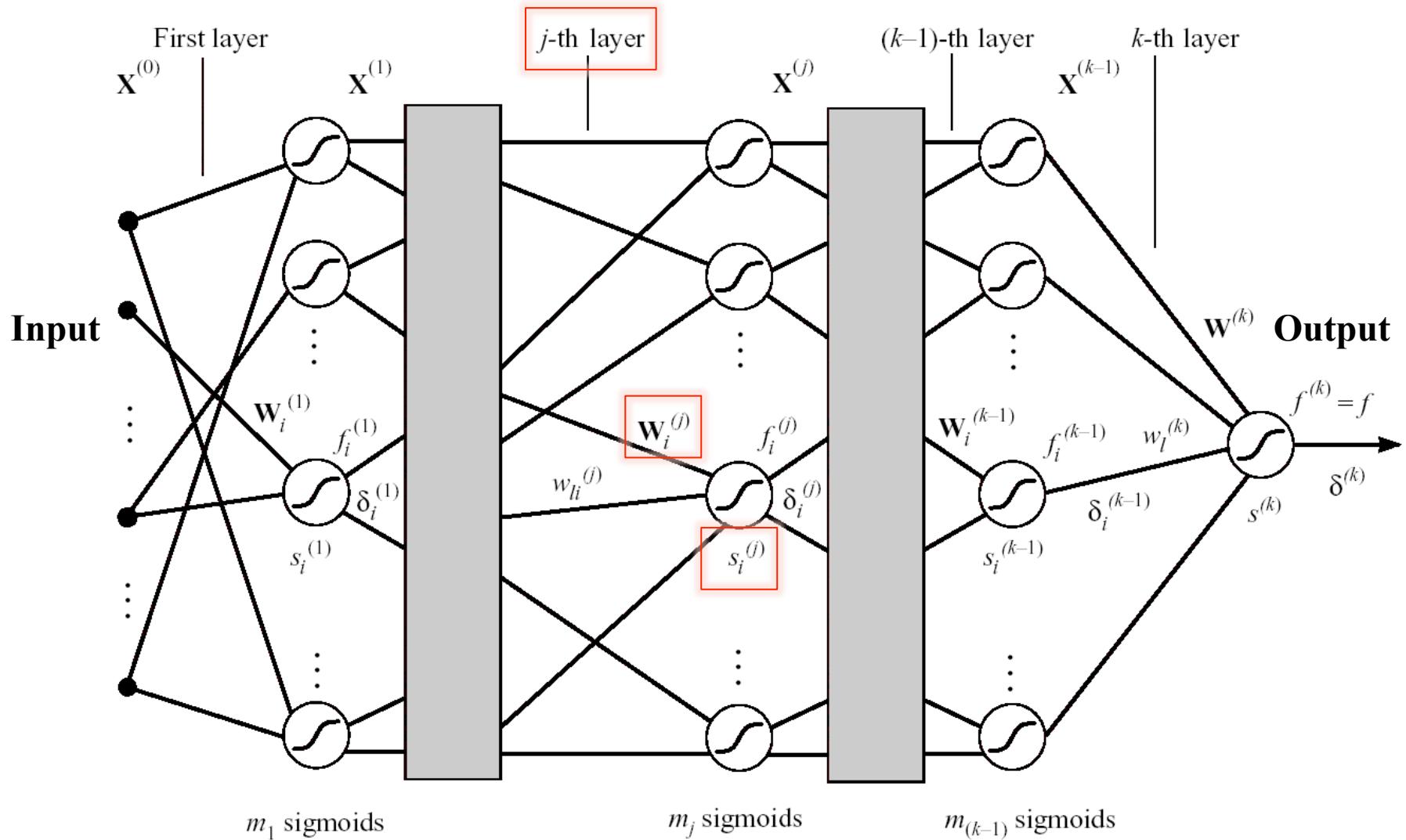
Structure	Regions	XOR	Meshed regions
single layer	Half plane bounded by hyperplane		
two layer	Convex open or closed regions		
three layer	Arbitrary (limited by # of nodes)		

Structure of Multilayer Perceptron



“3-layer Neural Net”, or
“2-hidden-layer Neural Net”

Structure of Multilayer Perceptron (MLP; Artificial Neural Network)



Learning Parameters of MLP

■ Loss Function

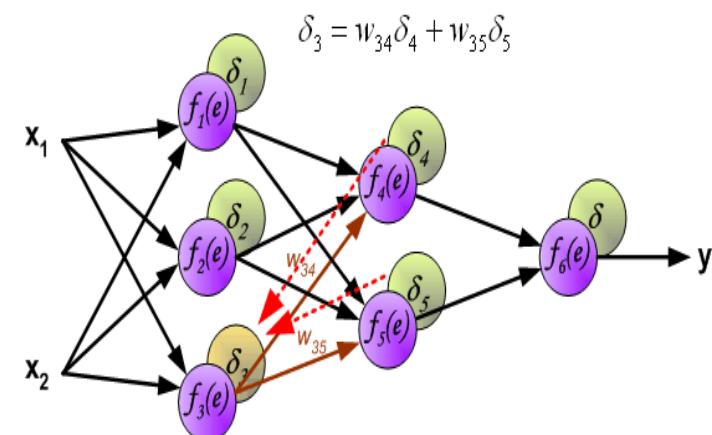
- We have the same Loss Function
- But the # of parameters are now much more (Weight for **each layer** and **each unit**)
- To use Gradient Descent, we need to calculate the **gradient for all the parameters**

■ 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**)

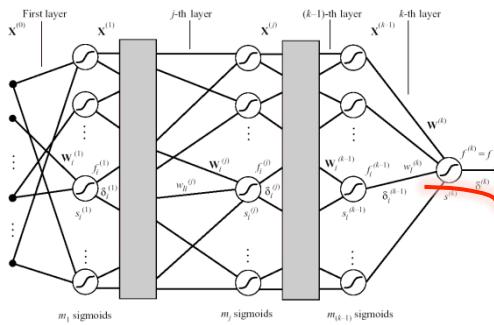
$$\varepsilon = (d - f)^2$$

Target Unit Output



Back Propagation Learning Algorithm (1/3)

■ Gradients of top-layer 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

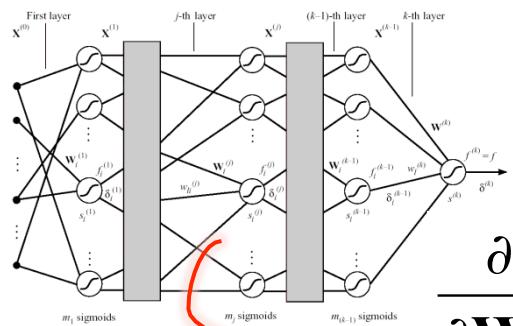
$$\mathbf{W} \leftarrow \mathbf{W} + c(d - f)f(1 - f)\mathbf{X}$$

■ 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

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

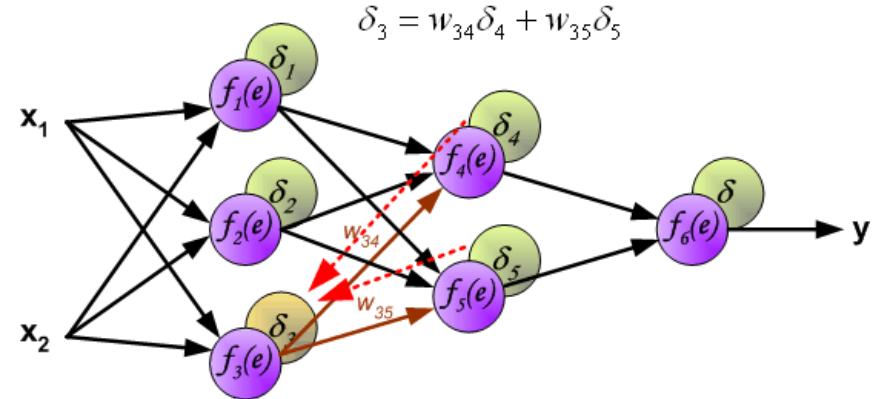
**Gradient Descent Update rule
for lower-layer weights**

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

Back Propagation Learning Algorithm (3/3)

- Applying chain rule, recursive relation between delta's

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



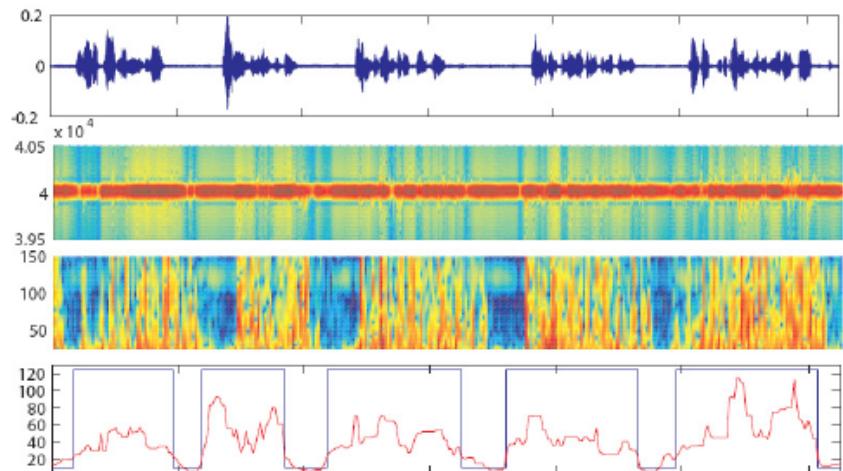
Algorithm: Back Propagation

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

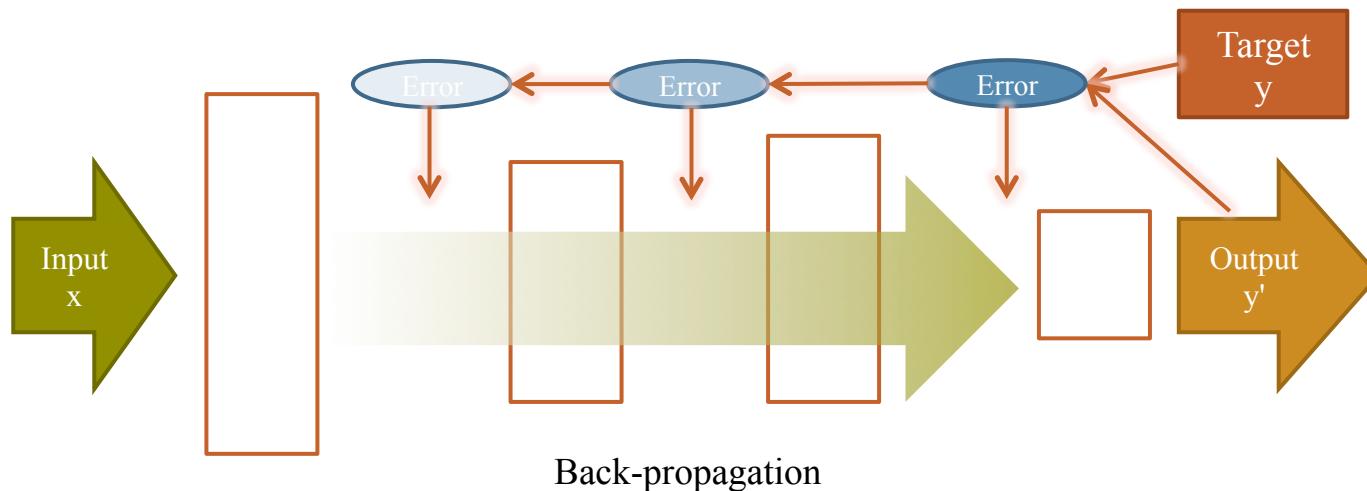
- Face Recognition
- Object Recognition
- Voice Recognition
- Spam mail Detection
- Disease Detection
- etc.



Limitations and Breakthrough

■ Limitations

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

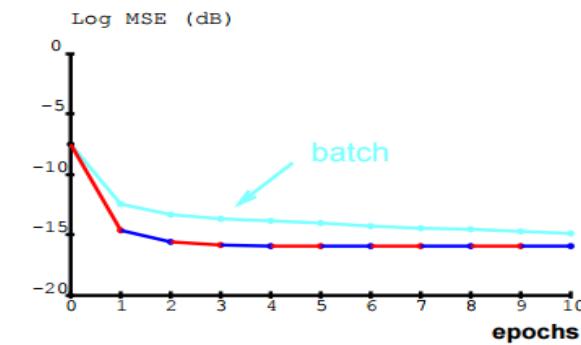
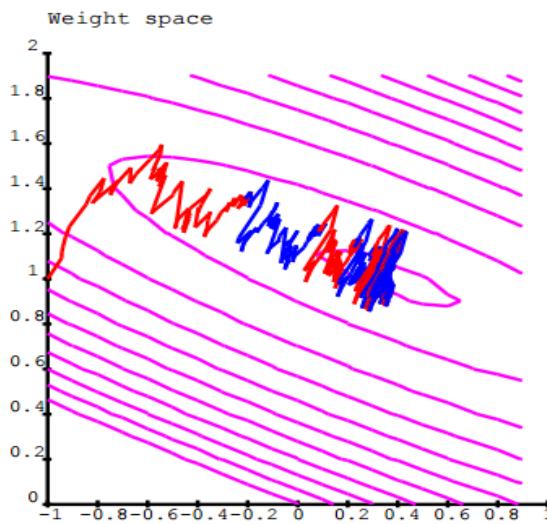
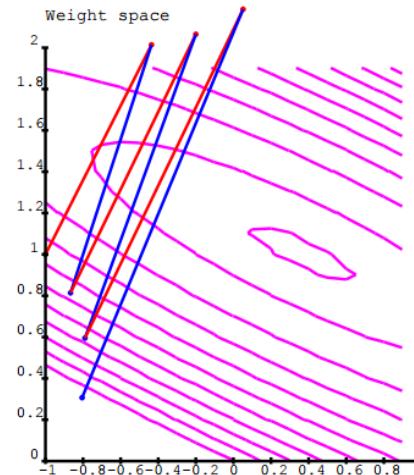
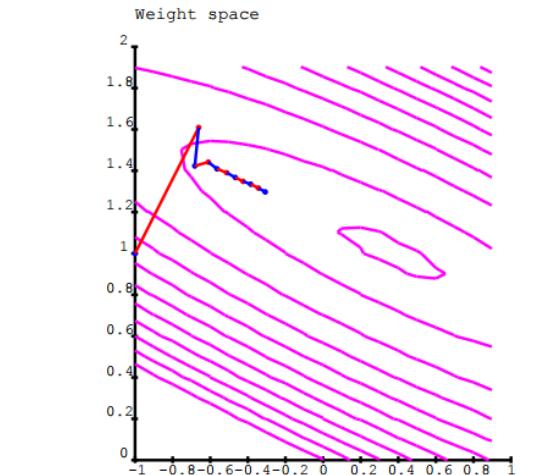


■ Breakthrough

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

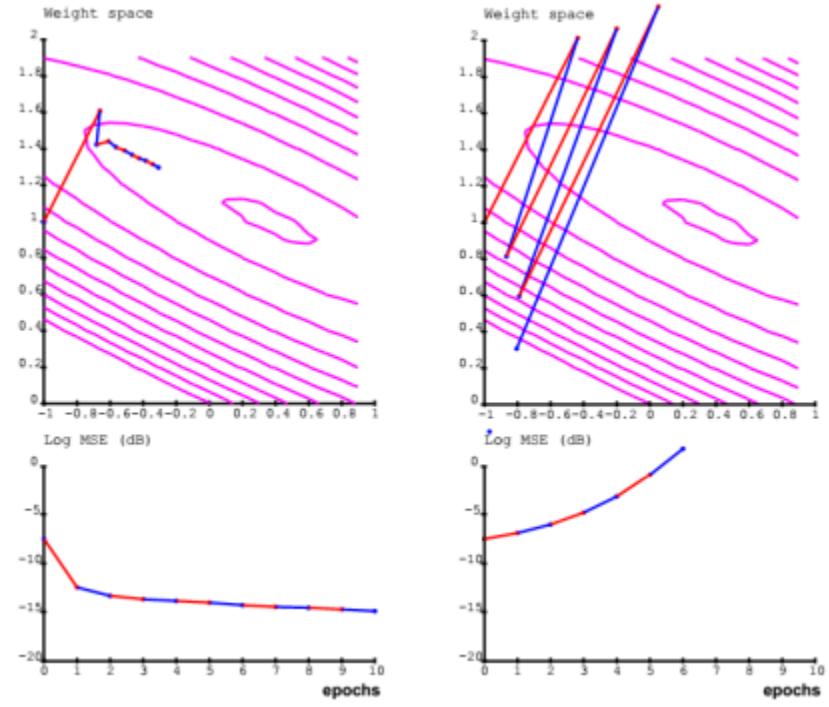
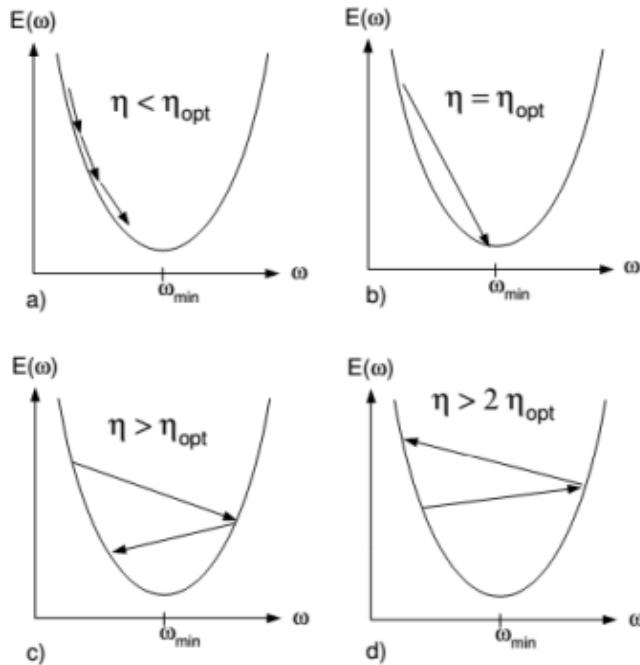
Some Issues (1/3)

■ Stochastic Gradient Descent



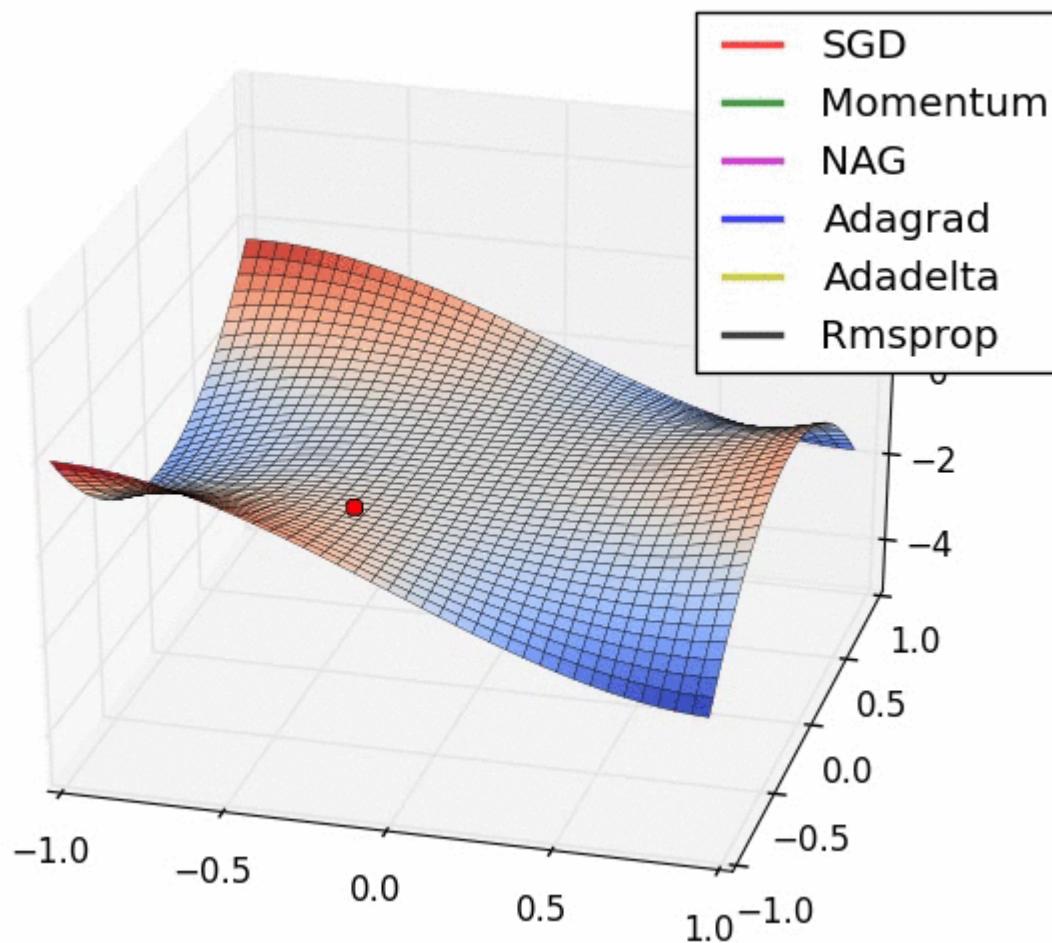
Some Issues (2/3)

- Learning Rate Adaptation
- Momentum
- Weight Decay



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 구조**가 있는 경우 **Local한 패턴을 학습**하는 것이 효과적

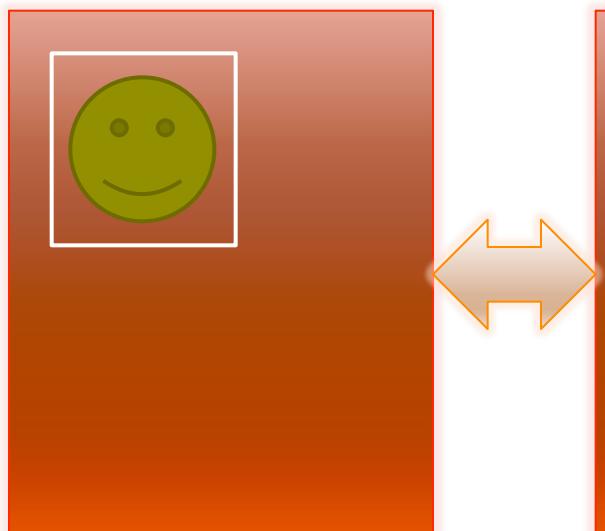


Image 1

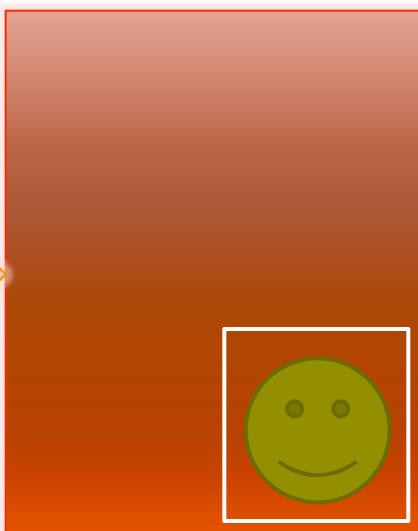
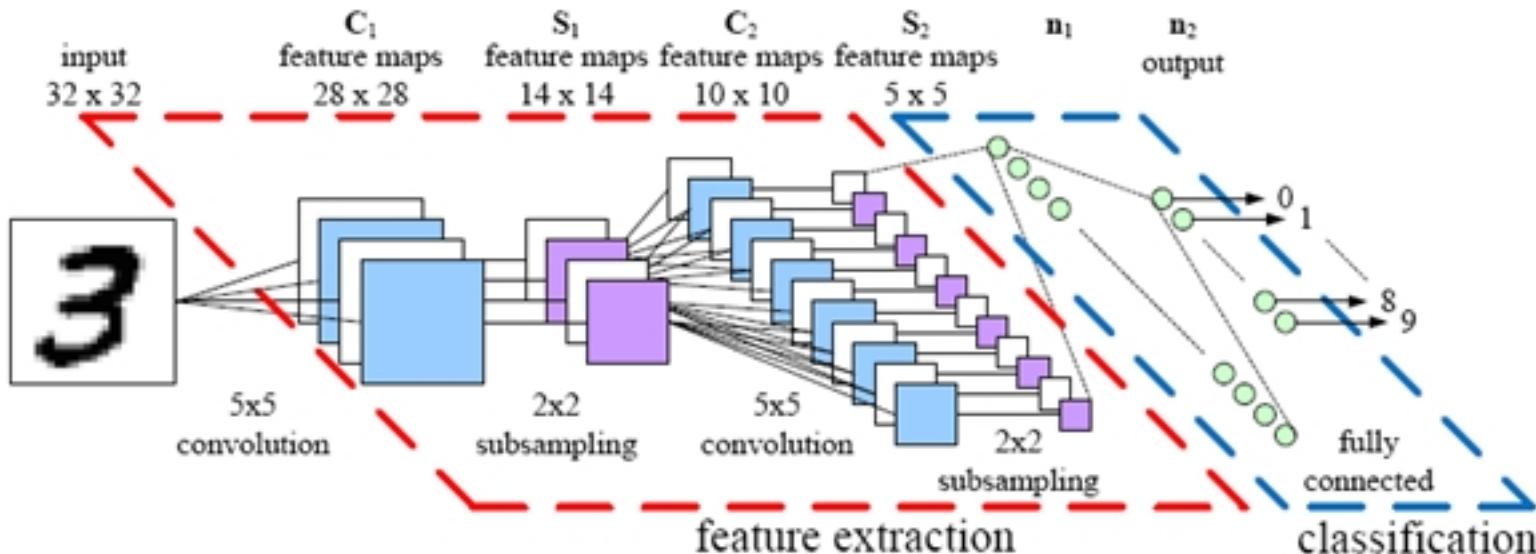


Image 2

- DBN의 경우 다른 data
- CNN의 경우 같은 data

Structure of Convolutional Neural Network (CNN)

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



Convolution Layer

- The Kernel Detects pattern:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

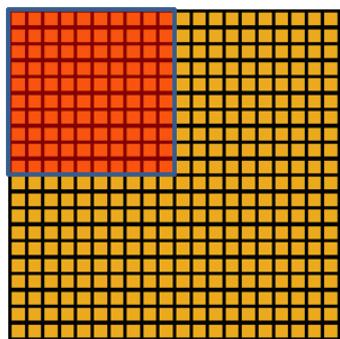
4		

Convolved Feature

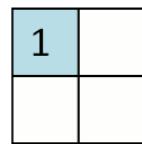
1	0	1
0	1	0
1	0	1

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

Max-Pooling Layer



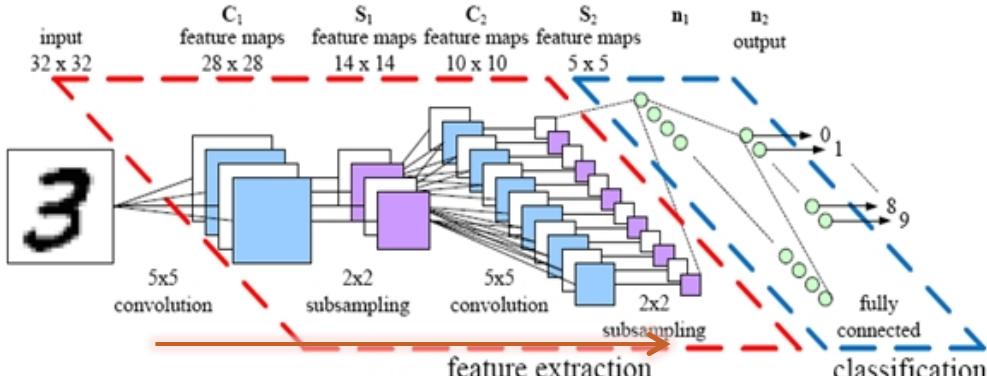
Convolved
feature



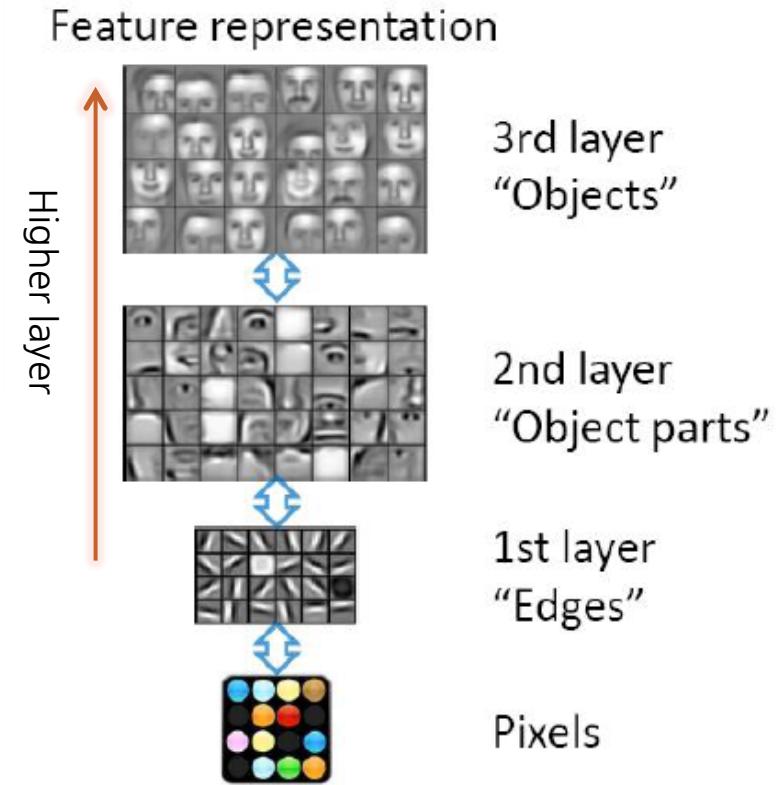
Pooled
feature

- 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



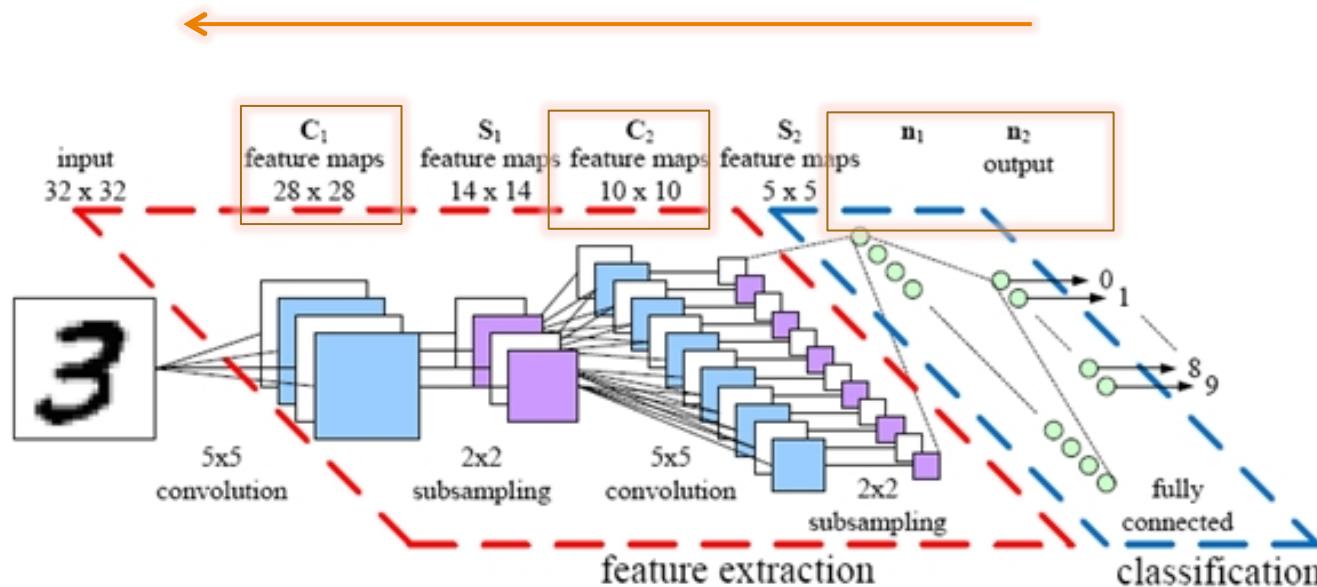
- Higher layer catches more specific, abstract patterns
- Lower layer catches more general patterns



Parameter Learning of CNN

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

Back Propagation



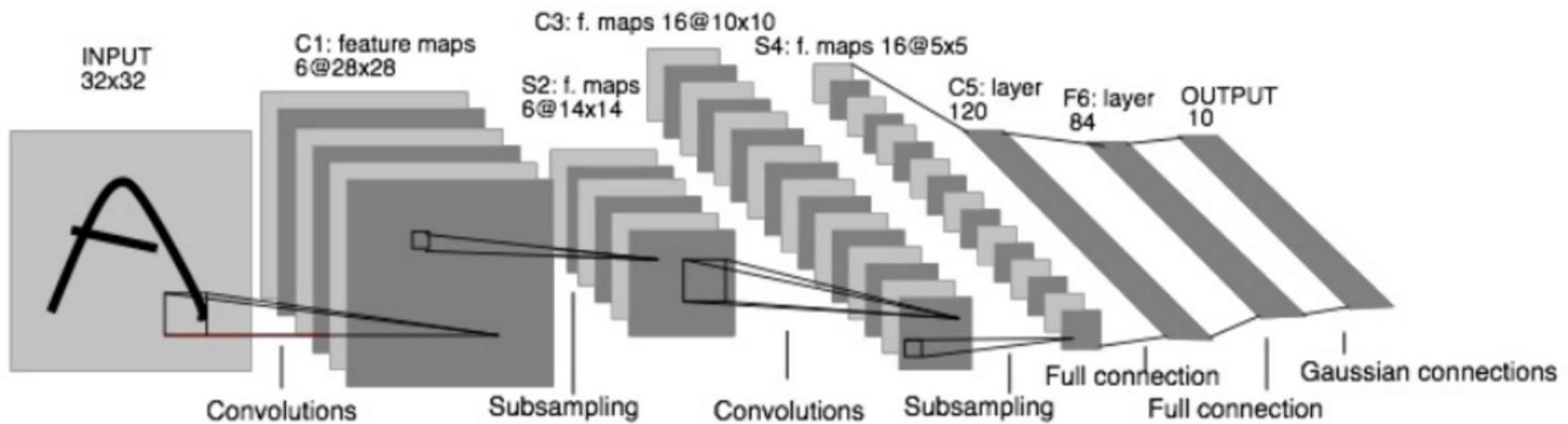
Applications (Image Classification) (1/4)

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

- 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
- ALL CNN!!

Applications (Image Classification) (2/4)

- 1989, CNN for hand digit recognition, Yann LeCun

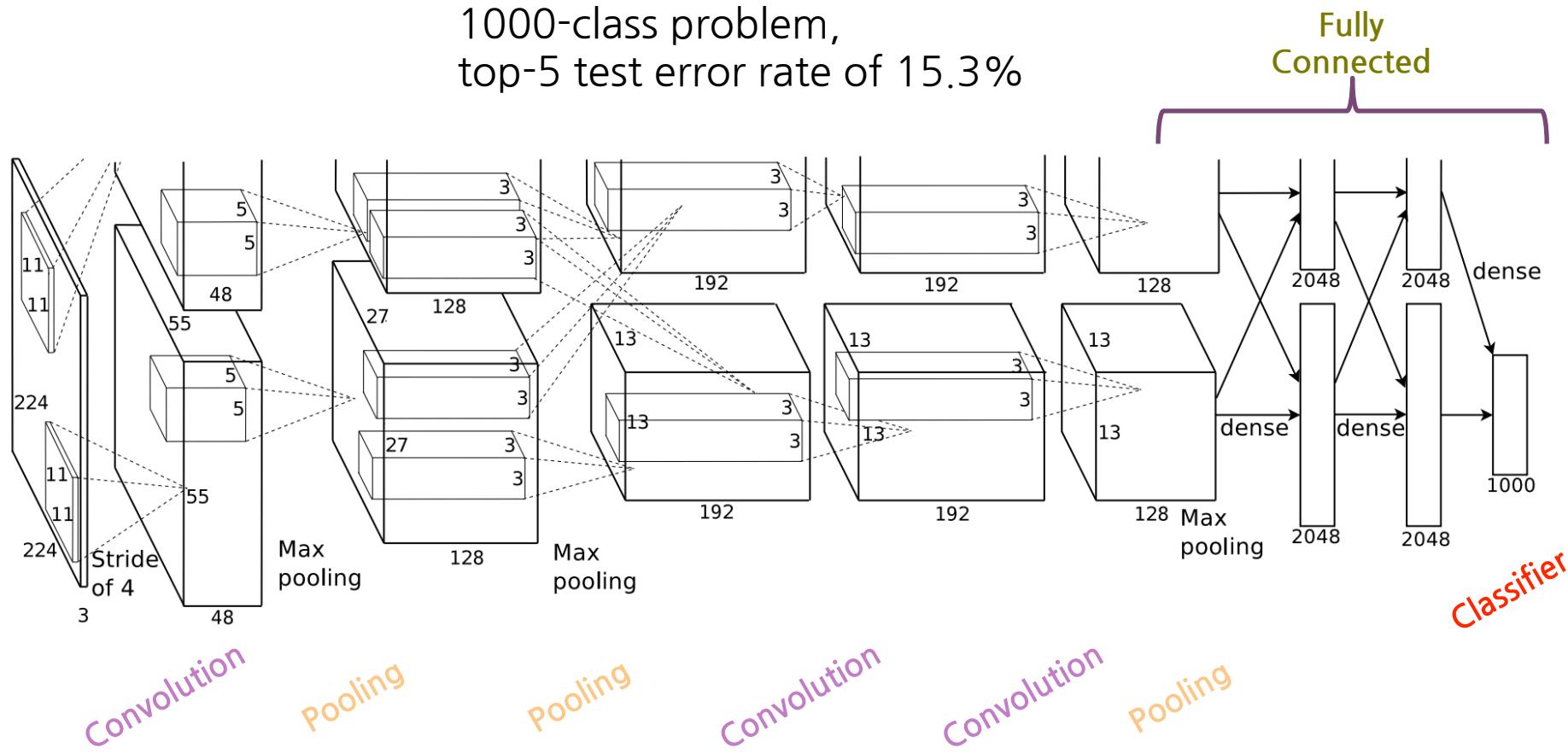


LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [Yann LeCun; LeNet]

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%

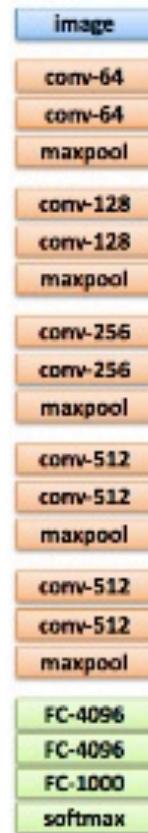


Applications (Image Classification) (4/4)

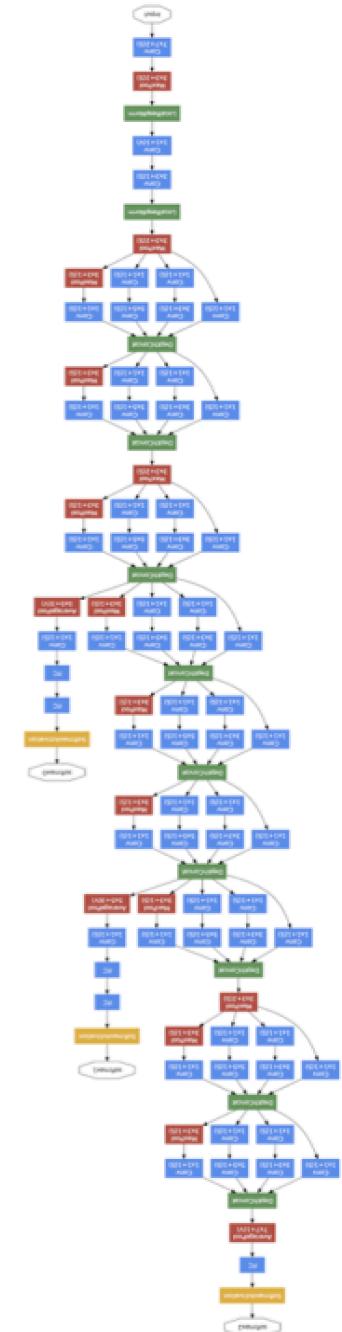
- 2014 ILSVRC winner, ~6.6% Top 5 error

Example: VGG

19 layers
3x3 convolution
pad 1
stride 1



Convolution
Pooling
Softmax
Other



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

Application (Speech Recognition)

