4190.408 **2015-Spring**

Deep Learning

Byoung-Tak Zhang
TA: Hyo-Sun Chun
School of Computer Science and Engineering
Seoul National University

A Learning Robot

목차

1. **기계학습이란 무엇인가?**
   - 정의, 중요성, 역사, 원리, 종류, 응용분야
2. **신경망은 어떻게 학습하는가?**
   - 퍼셉트론, 다층퍼셉트론, 오류역전파 학습알고리즘
3. **왜 딥신경망인가?**
   - 학습의 문제점, 딥러닝의 혁신점
4. **어떤 딥학습 아키텍쳐들이 있는가?**
   - CNN, DBN, DHN, RNN
5. **딥모델 설계시 유의점은?**
   - 과다학습, 모델복잡도, Occam's Razor, 정규화, SRM, MAP, MDL
6. **어떤 응용에 어떤 딥모델을 사용할 것인가?**
   - 감독/무감독, 변별/생성모델, 예측/모듈이해, 추론가능성, 연결성, 깊이, 배치/온라인학습

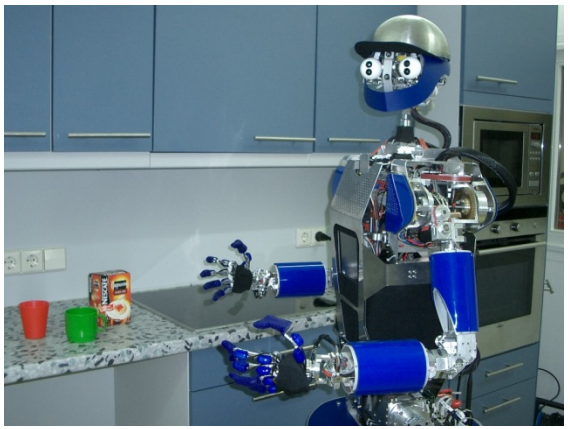
Conclusion

Theano 실습

1. 기계학습이란 무엇인가?

기계학습이란 무엇인가?

- 학습 시스템: "환경 E와의 상호작용으로부터 획득한 경험적인 데이터 D를 바탕으로 모델 M을 자동으로 구성하여 스스로 성능 P를 향상하는 시스템"
  - 환경 E
  - 데이터 D
  - 모델 M
  - 성능 P



- 특성 1: Self-improving Systems (인공지능)
- 특성 2: Knowledge Discovery (데이터마이닝)
- 특성 3: Data-Driven SW Design (SW공학)
- 특성 4: Automatic Programming (컴퓨터공학)

기계학습이 왜 중요한가?

- 프로그래밍이 어려운 문제들
  - No explicit knowledge
  - Hard to encode by humans
  - Environmental change

- IT 환경의 변화
  - Big data
  - Computing power
  - Mobile services

- 직접적인 비즈니스 가치 창출
  - User preference
  - Advertisement
  - Recommendation

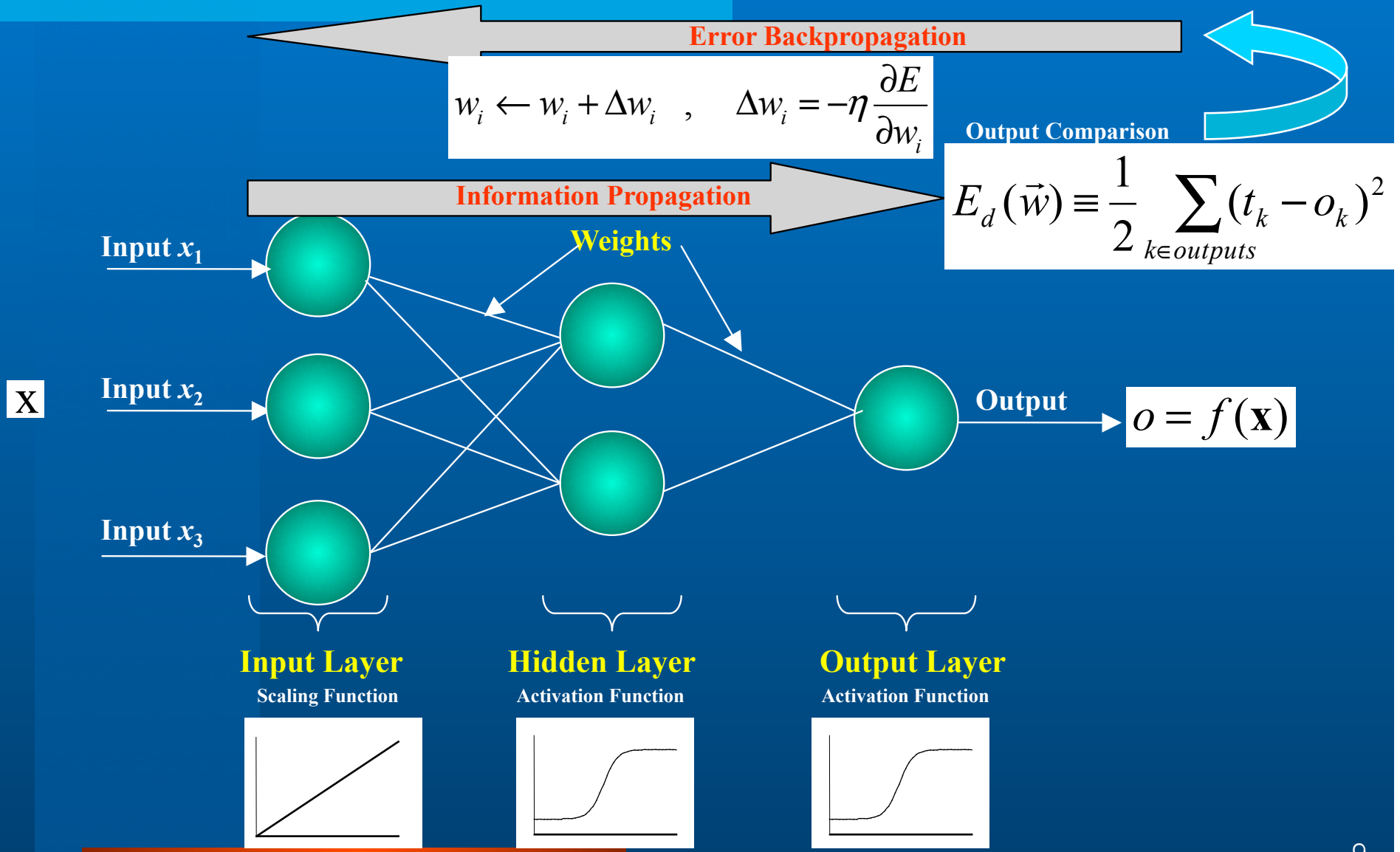
기계학습의 역사

- **1948: Checkers Player (Samuel)**
- 1958: Perceptron (Rosenblatt)
- 1975: Near miss concept learning (Winston)
- **1980: First ICML Conference (Machine Learning Workshop)**
- 1982: Self-organizing maps (Kohonen)
- 1983: Boltzmann machine (Hinton & Sejnowski)
- 1984: PAC computational learning theory (Valiant)
- **1986: Backpropagation algorithm (Rumelhart, Hinton, & Williams)**
- 1986: Decision trees (Quinlan)
- **1986: Machine Learning Journal**
- **1987: First NIPS Conference (Neural Information Processing Systems)**
- 1992: TD-Gammon (Tesauro)
- **1992: Support vector machines (Boser, Guyon, & Vapnik)**
- 1994: Learning Bayesian networks (Hackerman)
- 1995: Statistical learning theory (Vapnik)
- 1995: Neural Networks for Pattern Recognition (Bishop)
- **1997: Machine Learning Textbook (Mitchell)**
- 1998: Neural networks (Haykin)
- 1998: Reinforcement learning (Sutton & Barto)
- 1999: Learning in graphical models (Jordan)
- 1999: Kernel machines (Schoelkopf & Smolar)
- **2001: Journal of Machine Learning Research (JMLR)**
- 2003: Boosting algorithms (Freund)
- 2003: Information theory, inference, and learning algorithms (MacKay)

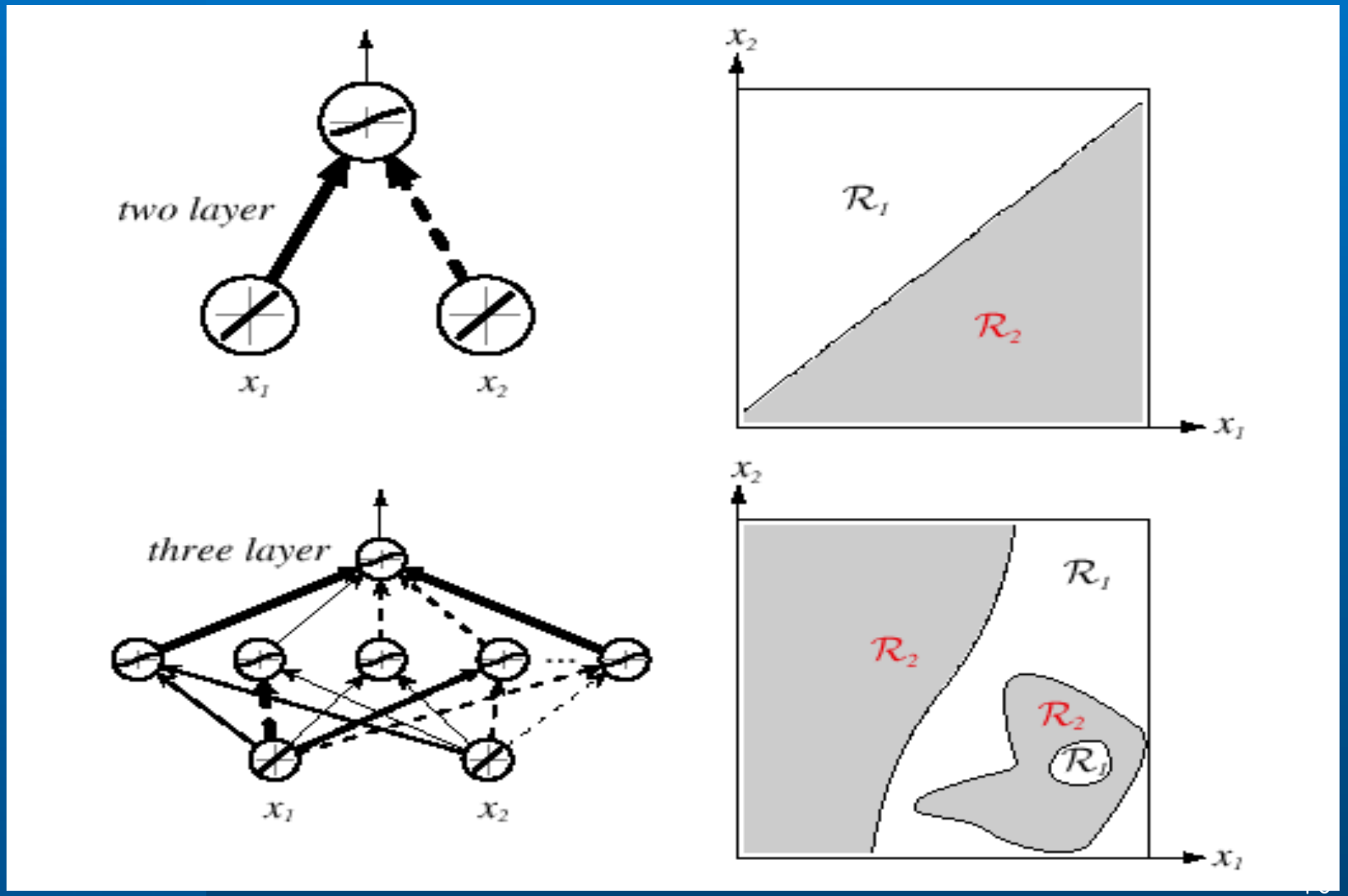
최근 기계학습 산업 동향

- **2005: DARPA Grand Challenge (Thrun)**
- 2005: Probabilistic Robotics (Thrun, Burgard, & Fox)
- **2006: Deep Neural Networks (Hinton)**
- 2006: Pattern Recognition and Machine Learning (Bishop)
- 2008: Neural Networks and Learning Machines (Haykin)
- 2009: Probabilistic Graphical Models (Koller)
- 2009: Siri personal assistant (Apple)
- 2009: Google Car (Thrun)
- **2010: Turing award in learning theory (Valiant)**
- 2011: Watson AI supercomputer (IBM)
- **2011: Turing award in Bayesian networks (Pearl)**
- 2012: DNNresearch deep learning (Hinton & Google)
- 2012: Large-scale image retrieval (Google)
- 2013: Human Brain Project HBP (EU)
- **2013: Quantum AI Lab for machine learning (Google)**
- **2013: AI Labs for commercial ML research (Facebook)**

기계학습의 원리: 반복적인 에러 교정

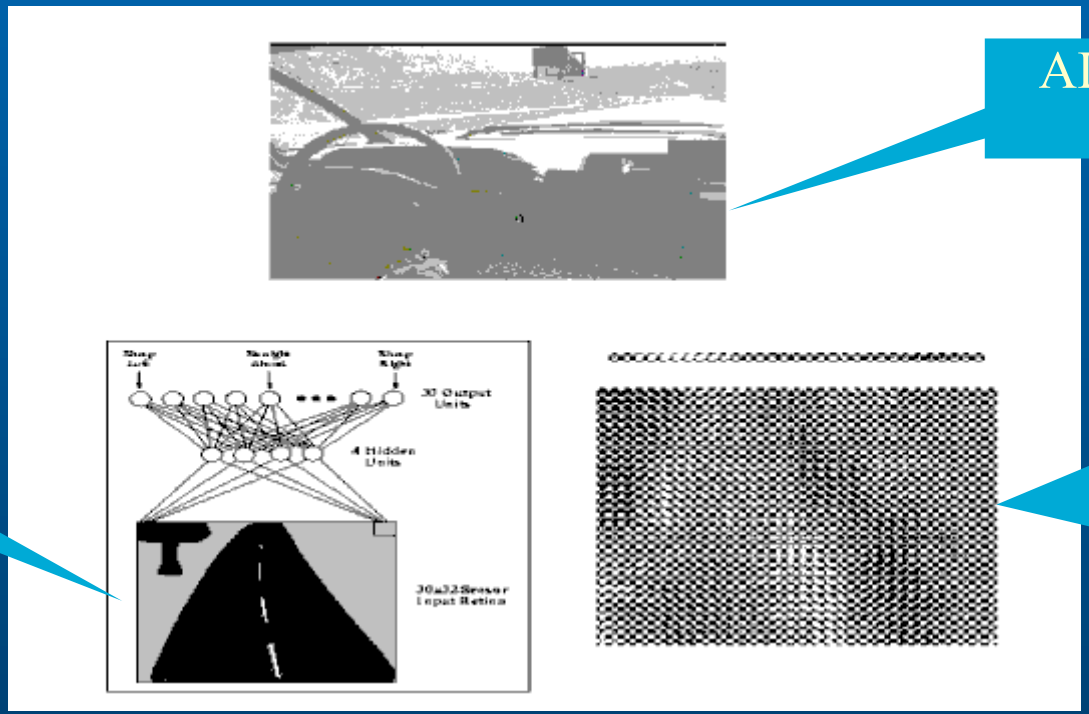


Learning as Finding Decision Boundaries from Labeled Data



기계학습 응용 예: 무인자동차
Autonomous Land Vehicle (ALV)

- NN learns to steer an autonomous vehicle.
- 960 input units, 4 hidden units, 30 output units
- Driving at speeds up to 70 miles per hour



기계학습: 종류 및 모델

장병탁, 차세대 기계학습 기술, 정보과학회지, 25(3), 2007

학습 방법	학습 문제의 예
감독 학습	인식, 분류, 진단, 예측, 회귀분석
무감독 학습	군집화, 밀도추정, 차원 축소, 특징추출
강화 학습	시행착오, 보상 함수, 동적 프로그래밍

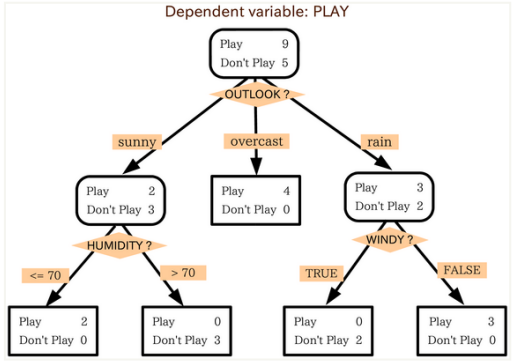
모델 구조	표 현	기계학습 모델 예
논리식	명제 논리, 술어논리, Prolog 프로그램	Version Space, 귀납적 논리 프로그래밍(ILP)
규칙	If-Then 규칙, 결정규칙	AQ
함수	Sigmoid, 다항식, 커널	신경망, RBF 망, SVM, 커널 머신
트리	유전자 프로그램, Lisp 프로그램	결정 트리, 유전자 프로그래밍, 뉴럴트리
그래프	방향성/무방향성 그래프, 네트워크	확률그래프 모델, 베이지안망, HMM

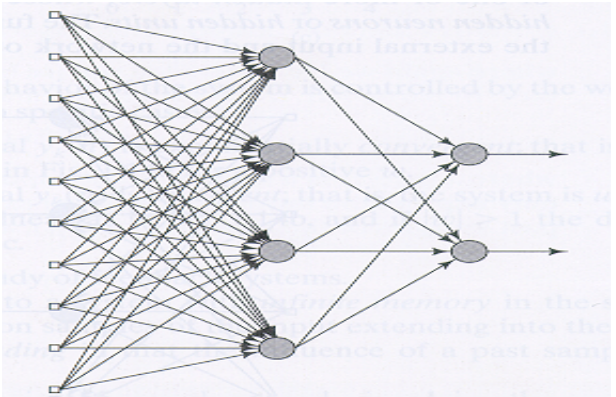
Machine Learning: Three Tasks

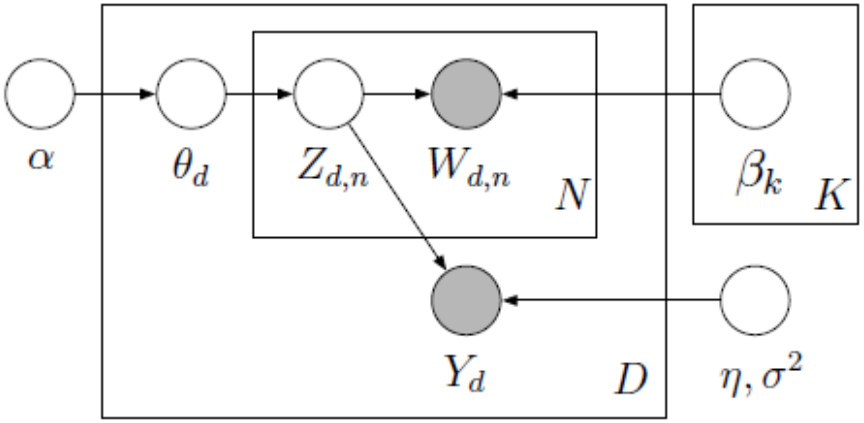
- Supervised Learning
  - Estimate an unknown mapping from known input and target out put pairs
  - Learn $f_{\mathbf{w}}$ from training set $D=\{(\mathbf{x},y)\}$ s.t. $f_{\mathbf{w}}(\mathbf{x}) = y = f(\mathbf{x})$
  - Classification: y is discrete
  - Regression: y is continuous
- Unsupervised Learning
  - Only input values are provided
  - Learn $f_{\mathbf{w}}$ from $D=\{(\mathbf{x})\}$ s.t. $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}$
  - Compression
  - Clustering
- Reinforcement Learning
  - Not target, but rewards (critiques) are provided $h_{\mathbf{w}}(\mathbf{x}, a, c)$
  - Learn a heuristic function $h_{\mathbf{w}}$ from $D=\{(\mathbf{x}, a, c)\}$ s.t.
  - Action selection
  - Policy learning

기계학습 종류: 모델 아키텍쳐 (표현구조)

- Instances
- Rules
- Equations
- Functions
- Decision trees
- Neural networks
- Graphical models
- Lattice models
- Hierarchical models
- Complex networks
- Model ensembles







기계학습 종류: 목적 함수 (성능 평가방법)

- Squared error
- Classification error
- Margin
- Accuracy
- Precision and recall
- Likelihood
- Posterior probability
- Cost, utility, value
- Risk
- Entropy
- Cross entropy
- Information gain
- Mutual information
- KL divergence

$$p(\theta \mid \mathbf{X}^N) = \frac{p(\mathbf{X}^N \mid \theta) p(\theta)}{p(\mathbf{X}^N)}$$

$$\theta^{MAP} = \arg\max_{\theta} \frac{p(\mathbf{X}^N \mid \theta) p(\theta)}{p(\mathbf{X}^N)}$$

$$= \arg\max_{\theta} p(\mathbf{X}^N \mid \theta) p(\theta)$$

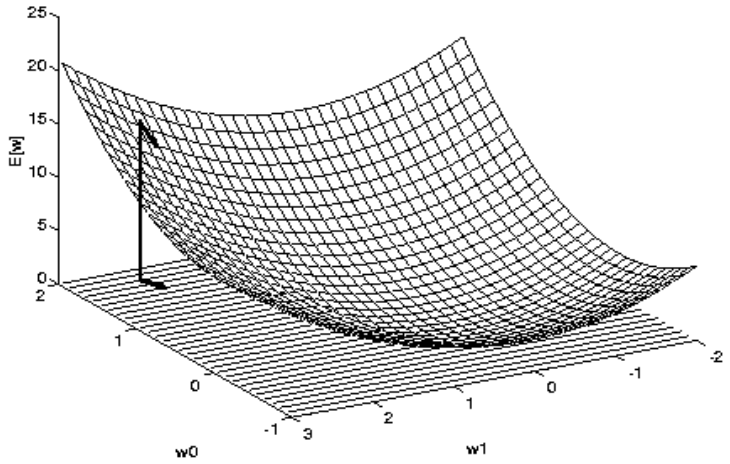
$$\underbrace{R_{pen}(\theta)}_{penalized\ risk} = \underbrace{R_{emp}(\theta)}_{empirical\ risk} + \alpha \underbrace{\Phi[f(\mathbf{x}, \theta)]}_{penalty}$$

$$KL(p \,\|\, q) = \sum_{i=1}^{N} p(x_i) \log_2 \frac{p(x_i)}{q(x_i)}$$

기계학습 종류: 학습 알고리즘 (최적화 기법)

- **Parameter vs. Structure**
- **Deterministic vs. Stochastic**
- **Discriminative vs. Generative**

- Error minimization
  - E.g.: Least mean squares
- Convex optimization
  - E.g.: Gradient descent
- Stochastic search
  - E.g.: MCMC
- Combinatorial optimization
  - E.g.: Genetic algorithms
- Constrained optimization
  - E.g.: Linear programming



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

기계학습의 응용 분야

응용 분야	적용 사례
인터넷 정보검색	텍스트 마이닝, 웹로그 분석, 스팸필터, 문서 분류, 여과, 추출, 요약, 추천
컴퓨터 시각	문자 인식, 패턴 인식, 물체 인식, 얼굴 인식, 장면전환 검출, 화상 복구
음성인식/언어처리	음성 인식, 단어 모호성 제거, 번역 단어 선택, 문법 학습, 대화 패턴 분석
모바일 HCI	동작 인식, 제스쳐 인식, 휴대기기의 각종 센서 정보 인식, 떨림 방지
생물정보	유전자 인식, 단백질 분류, 유전자 조절망 분석, DNA 칩 분석, 질병 진단
바이오메트릭스	홍채 인식, 심장 박동수 측정, 혈압 측정, 당뇨치 측정, 지문 인식
컴퓨터 그래픽	데이터기반 애니메이션, 캐릭터 동작 제어, 역운동학, 행동 진화, 가상현실
로보틱스	장애물 인식, 물체 분류, 지도 작성, 무인자동차 운전, 경로 계획, 모터 제어
서비스업	고객 분석, 시장 클러스터 분석, 고객 관리(CRM), 마켓팅, 상품 추천
제조업	이상 탐지, 에너지 소모 예측, 공정 분석 계획, 오류 예측 및 분류

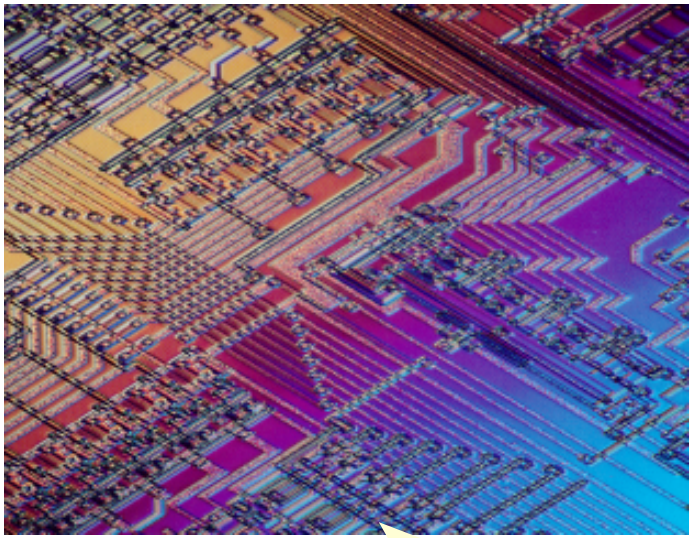
장병탁, 차세대 기계학습 기술, 정보과학회지, 25(3), 2007

2009: Google "Self-Driving Car"

- DARPA Grand Challenge (2005)
- DARPA Urban Challenge (2007)
- Google Self-Driving Car (2009)

2. 신경망은 어떻게 학습하는가?

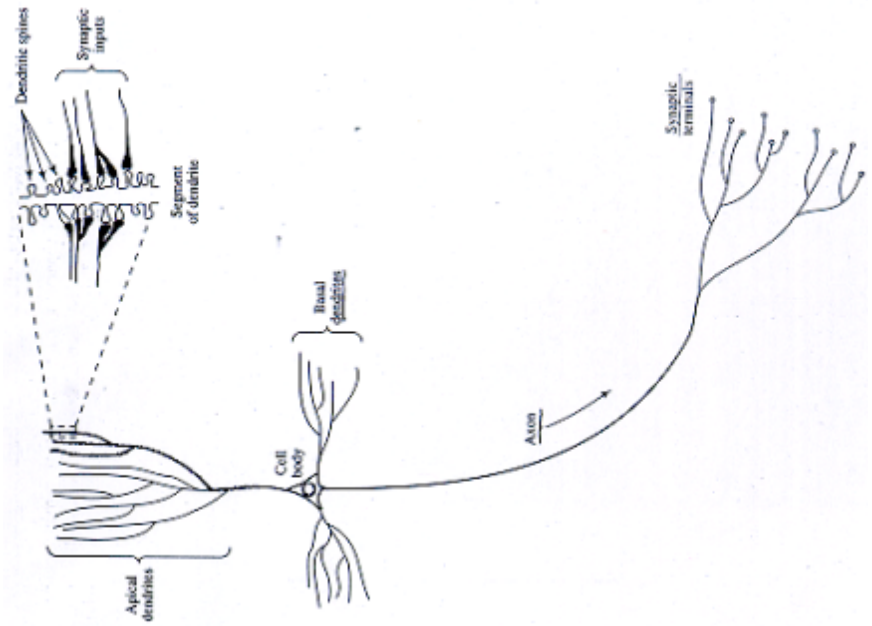
뇌와 컴퓨터의 비교

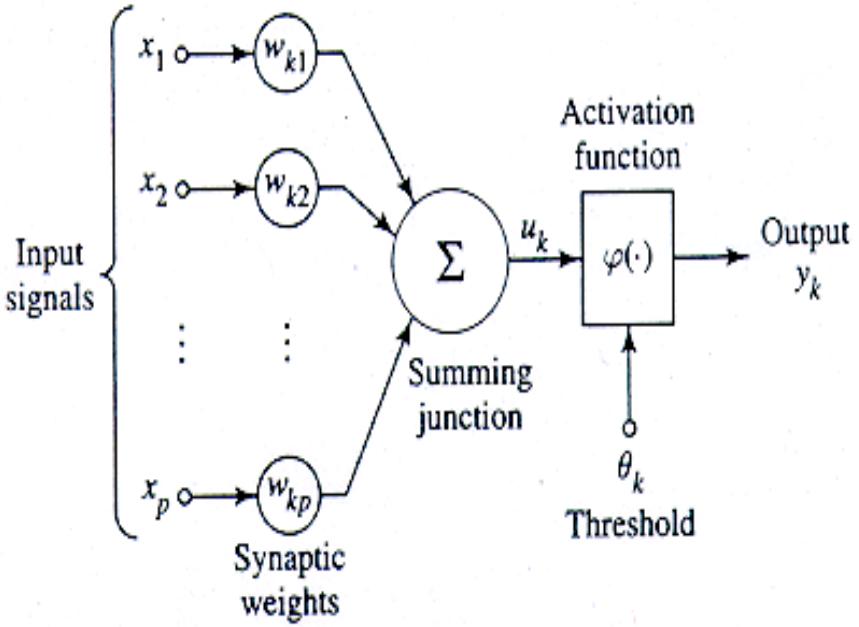


1. 10 billion neurons
2. 60 trillion synapses
3. Distributed processing
4. Nonlinear processing
5. Parallel processing

1. Faster than neuron (10^{-9} sec)
   cf. neuron: 10^{-3} sec
3. Central processing
4. Arithmetic operation (linearity)
5. Sequential processing

뉴런과 인공뉴런



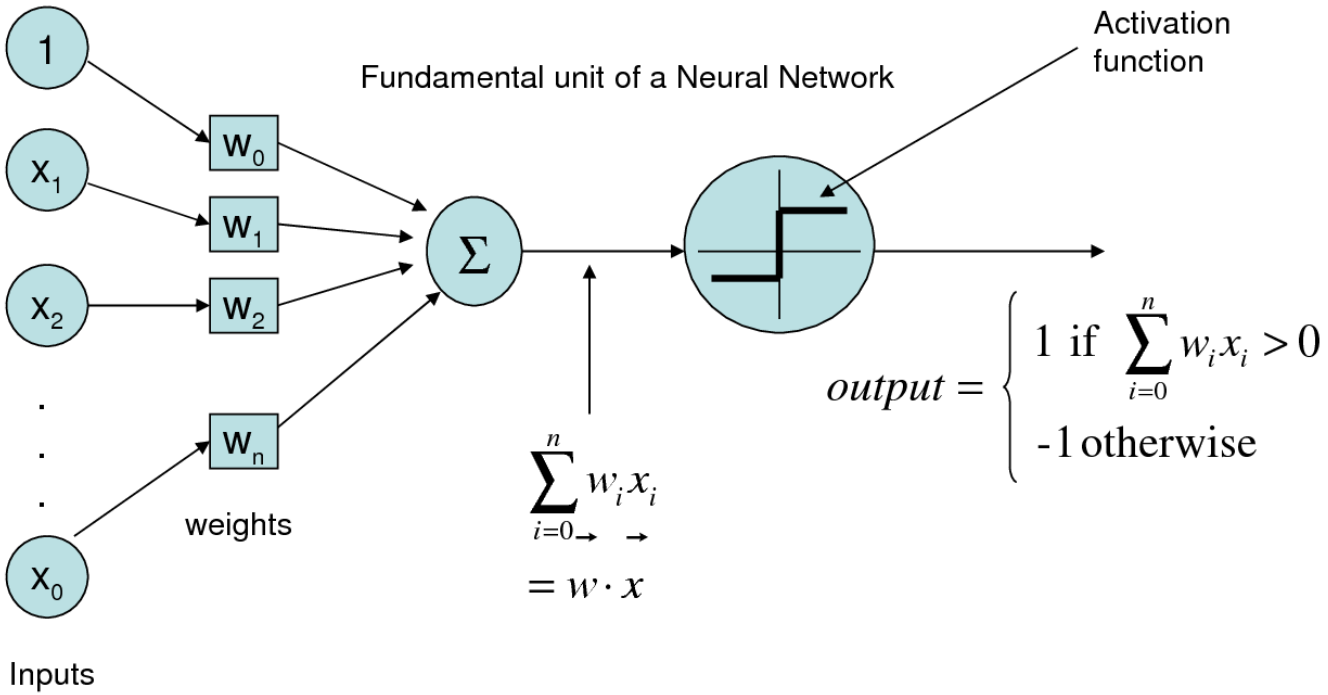


수상돌기 세포체 축색돌기

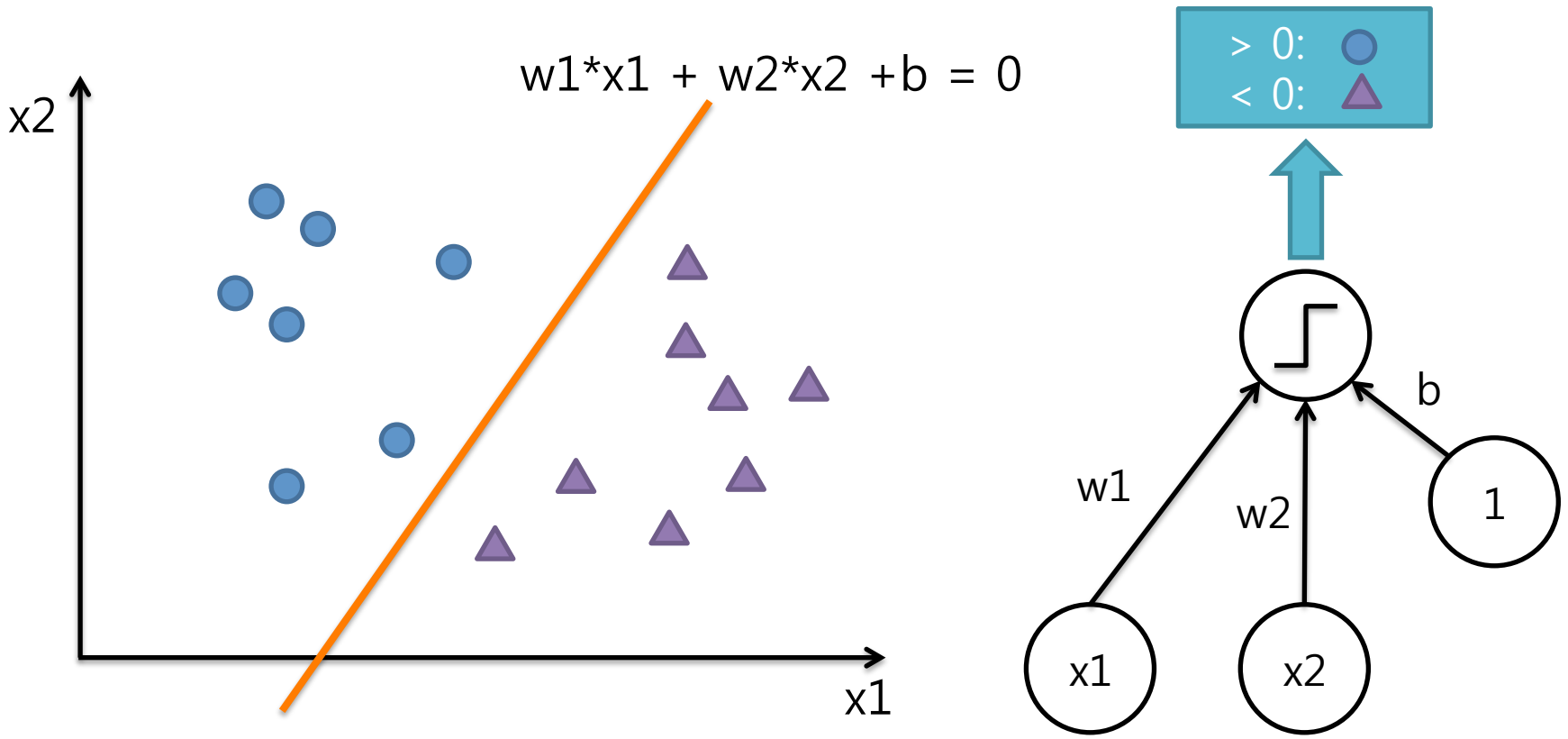
신경망: Simple Perceptron

Artificial Neural Networks

The Perceptron



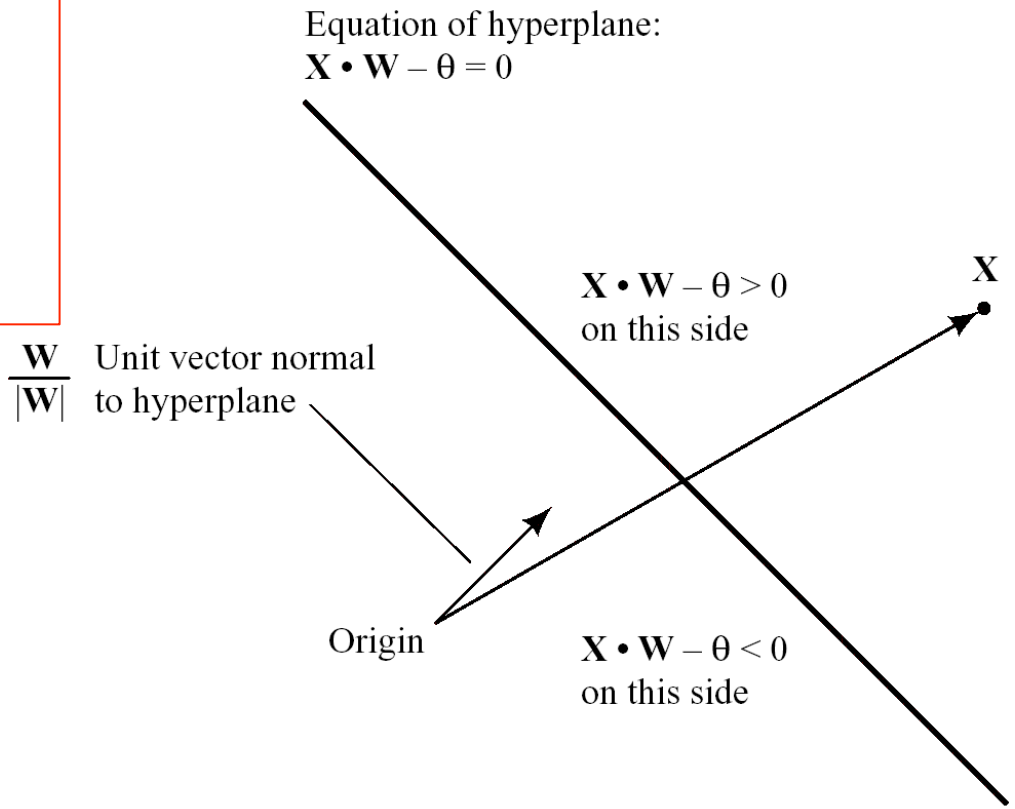
Simple Perceptron: 작동 원리



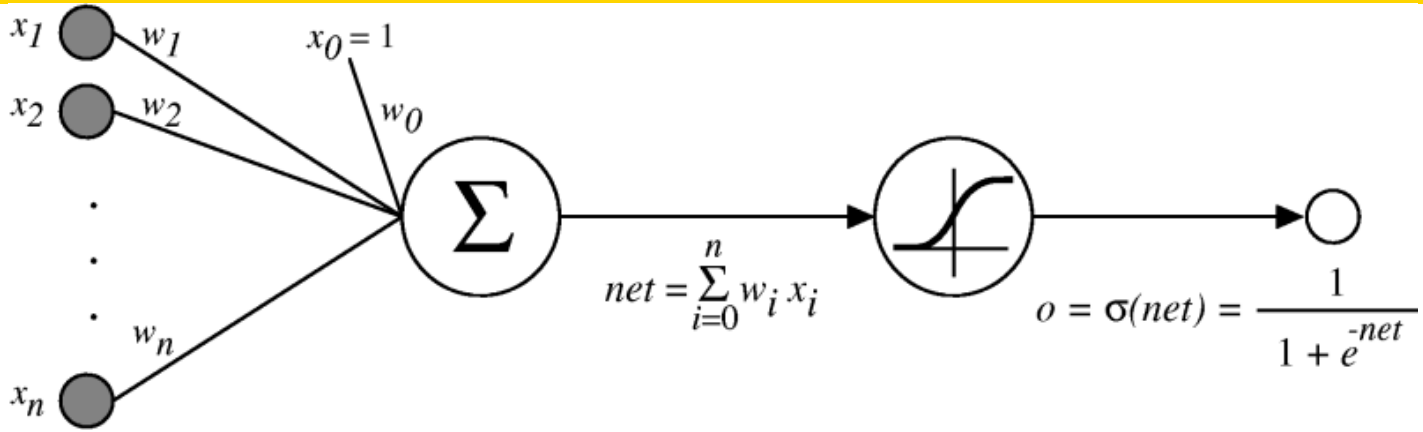
Simple Perceptron: 학습 원리

start:
The weight vector w is generated randomly
test:
A vector x ∈ P ∪ N is selected randomly,
If x∈P and w·x>0 goto test,
If x∈P and w·x≤0 goto add,
If x ∈ N and w · x < 0 go to test,
If x ∈ N and w · x ≥ 0 go to subtract.
add:
Set w = w+x, goto test
subtract:
Set w = w-x, goto test

Linear Neuron

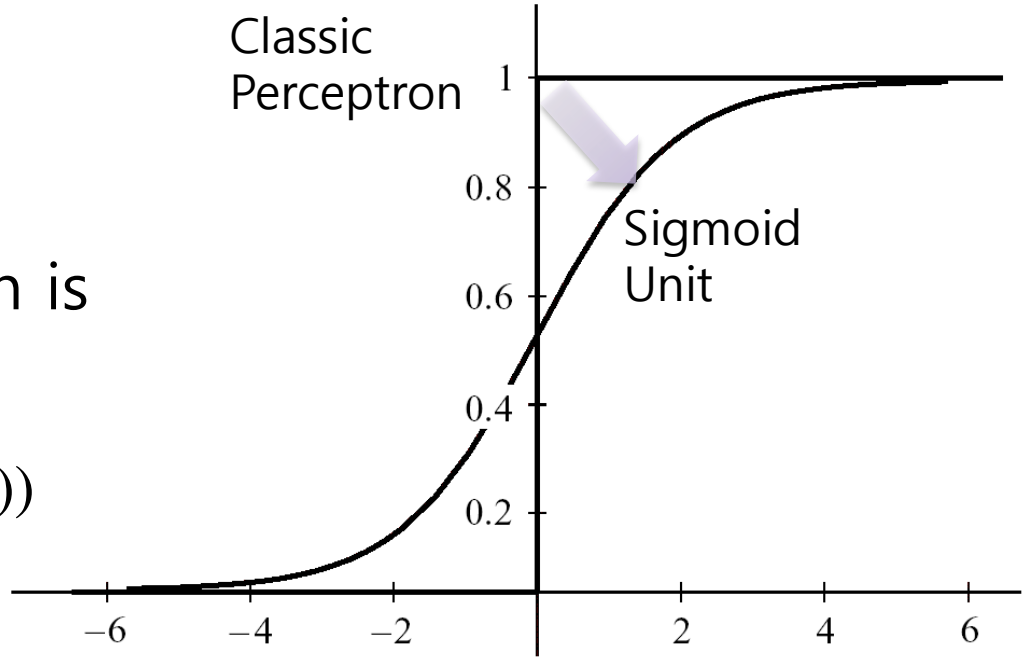


From Linear Neuron to Nonlinear Neuron: Sigmoid Unit



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**

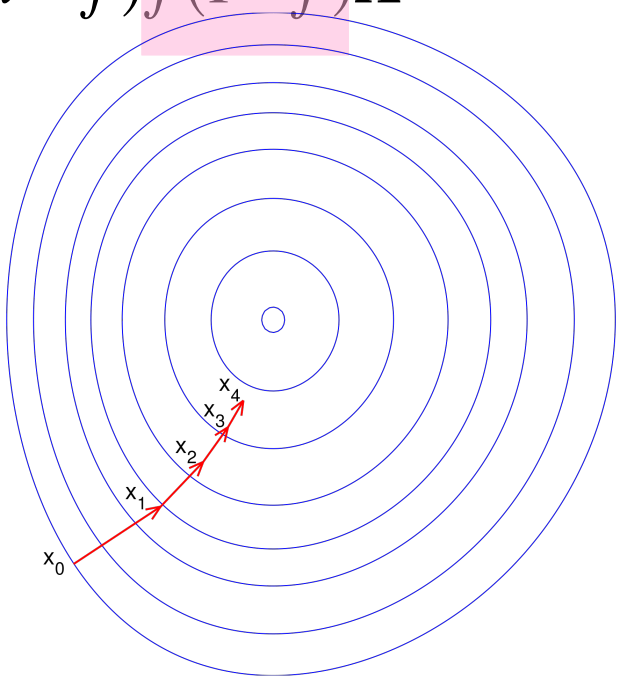
$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = \frac{\partial \varepsilon}{\partial f}\frac{\partial f}{\partial s}\frac{\partial s}{\partial \mathbf{W}} = -2(d-f)\frac{\partial f}{\partial s}\mathbf{X} = -2(d-f)f(1-f)\mathbf{X}$$

$$s = \mathbf{W}\mathbf{X}$$

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

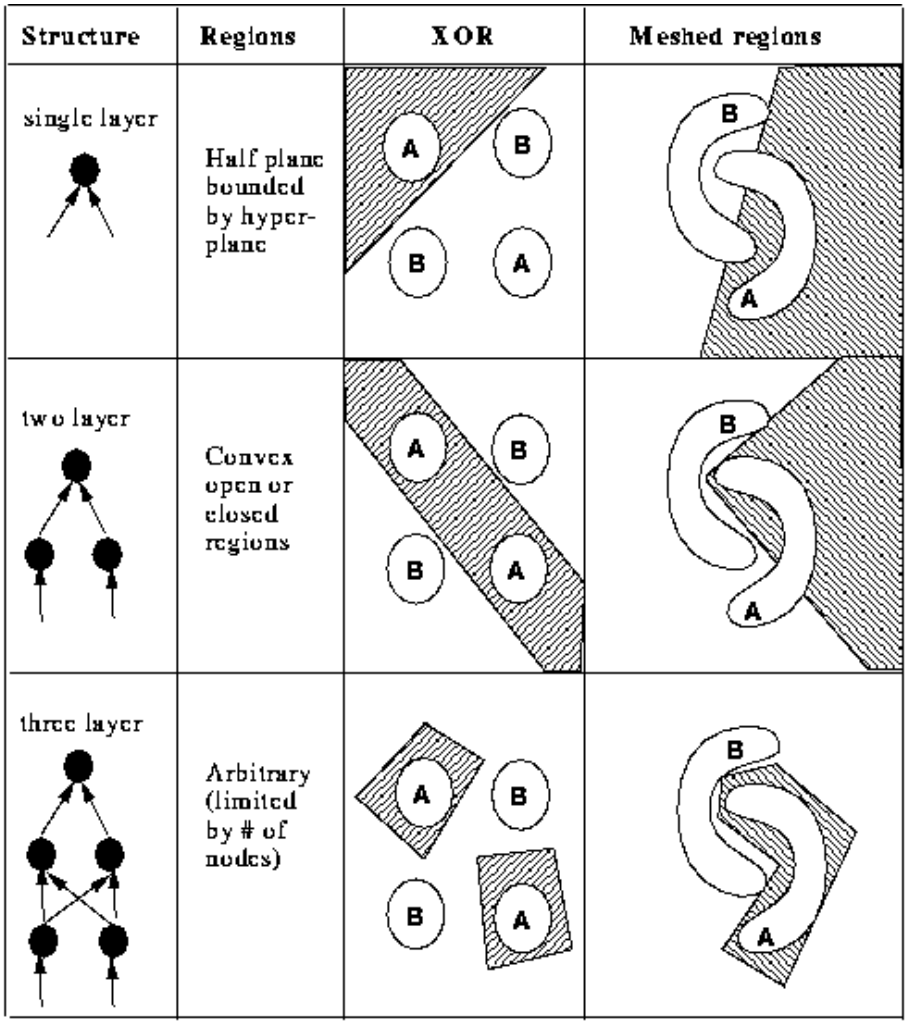
$$f'(s) = \frac{\partial f}{\partial s} = f(s)(1-f(s))$$

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

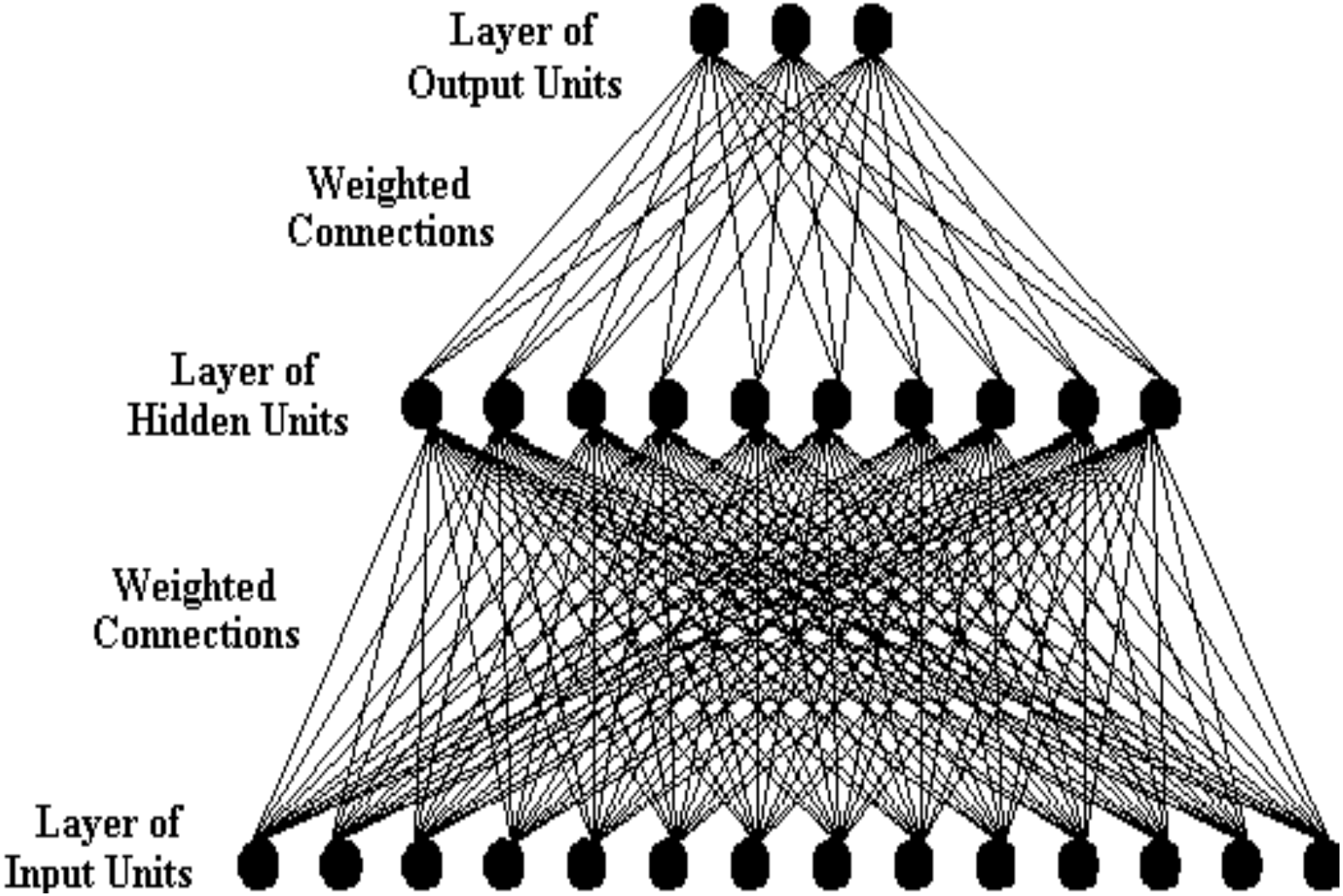


Need for Multiple Units and Multiple Layers

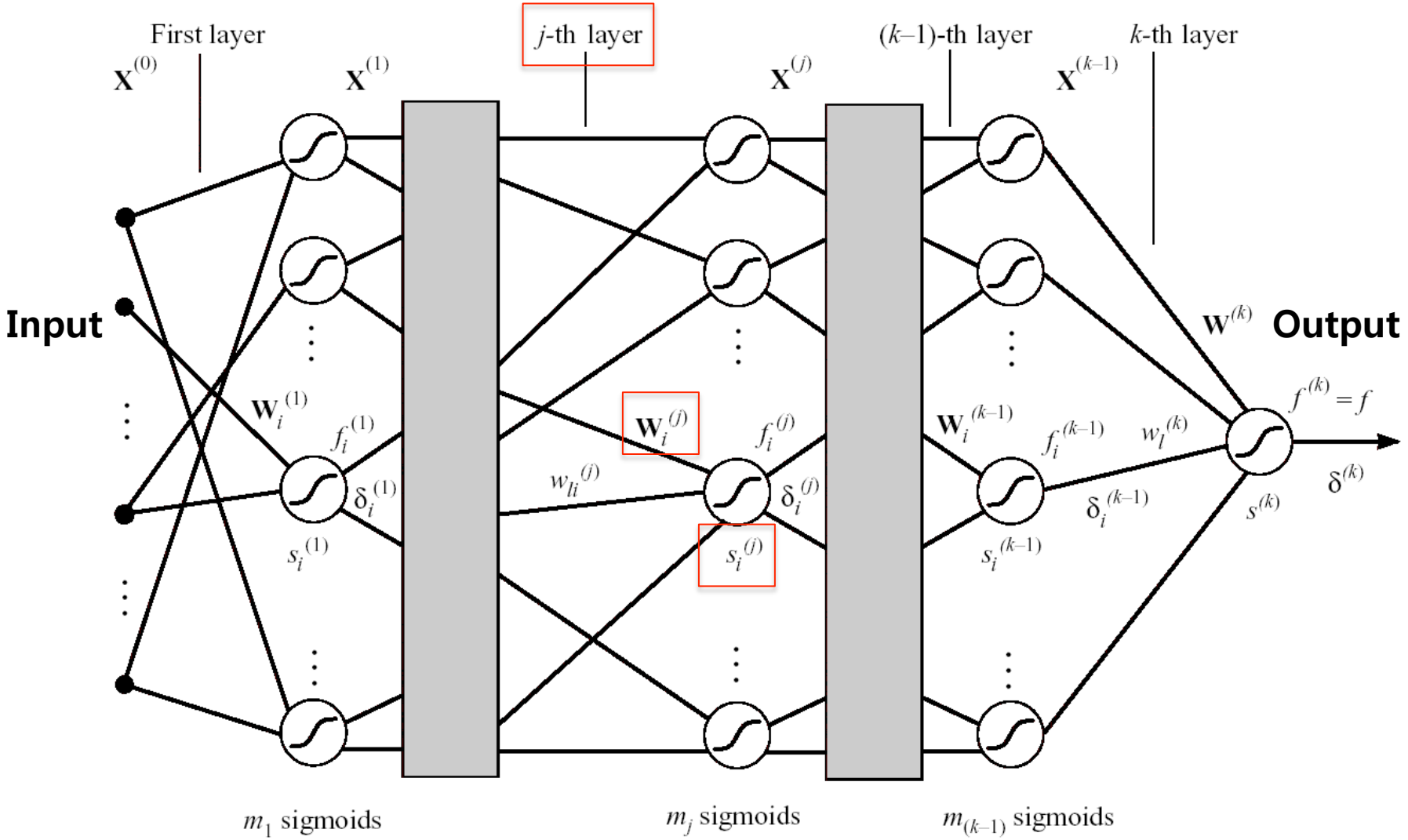
- Multiple boundaries are n eeded (e.g. XOR problem) → Multiple Units
- More complex regions are needed (e.g. Polygons) → Multiple Layers



다층퍼셉트론 신경망 (MLP)



From Perceptron to Multilayer Perceptron (MLP; Artificial Neural Network)



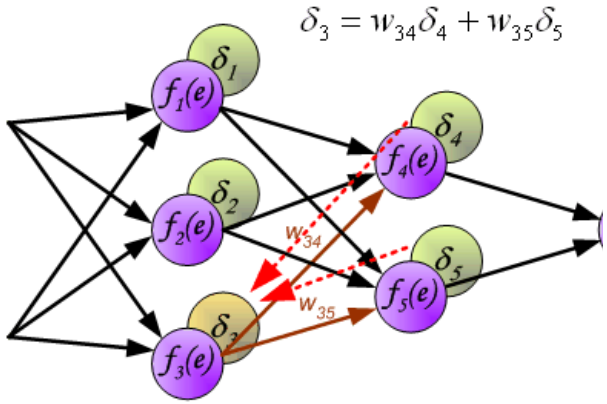
Learning Parameters of MLP

- **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**

Target, Unit Output

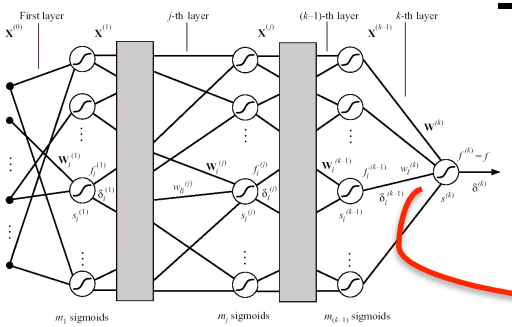
$$\varepsilon = (d - f)^2$$

- **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 (error backpropagation)**



Backpropagation Learning Algorithm (1/3)

- Gradients of top-layer weights and update rule



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

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

Gradient Descent update rule

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

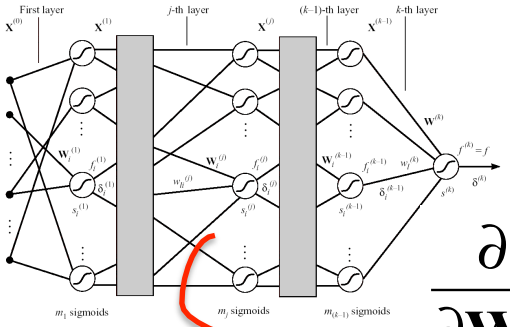
- Store intermediate value delta for later use of chain rule

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

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

Backpropagation 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

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

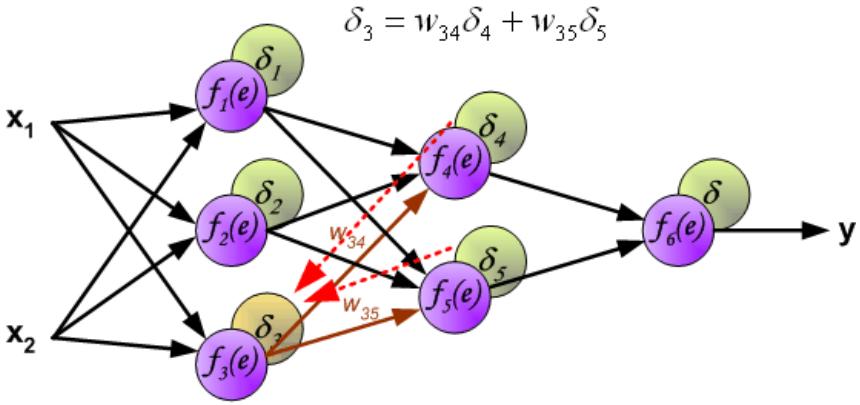
Gradient descent update rule for lower-layer weights

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

Backpropagation Learning Algorithm (3/3)

- Applying chain rule, recursive relation between delta's

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

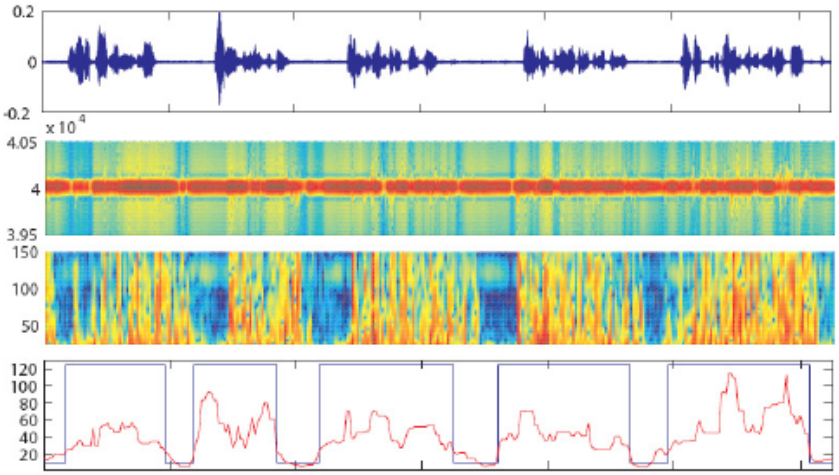


Algorithm: Backpropagation

1. Randomly initialize weight parameters
2. Calculate the activations of all units (with input data)
3. Calculate top-layer delta
4. Backpropagate 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

Application Examples

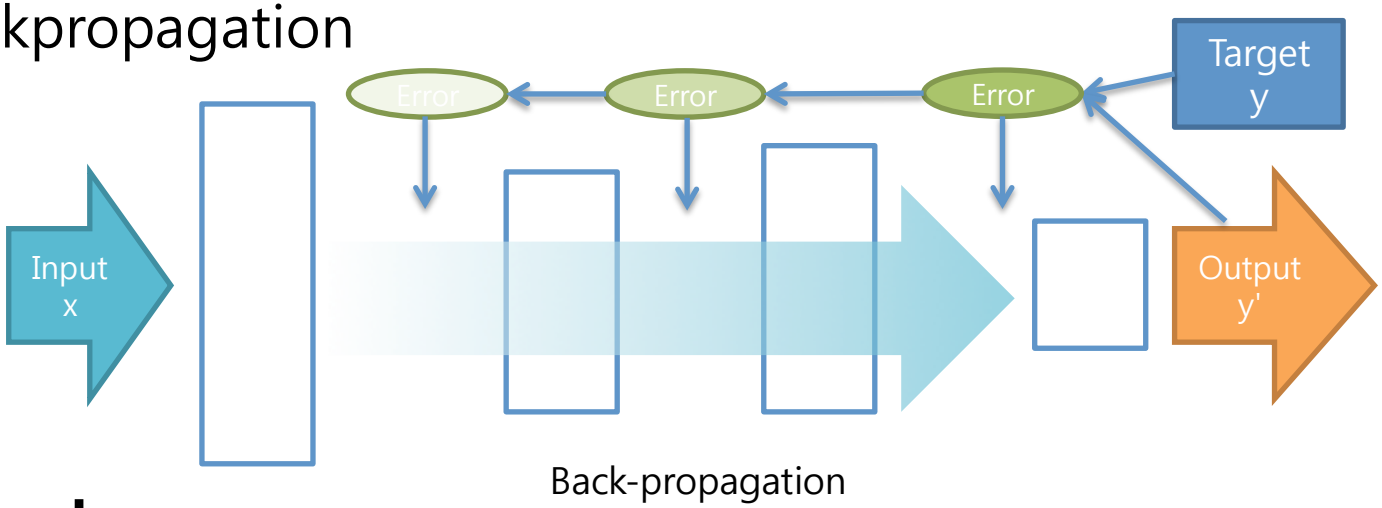
- **Almost All Classification Problems**
  - Face recognition
  - Object recognition
  - Voice recognition
  - Spam-mail detection
  - Disease detection
  - etc.



3. 왜 딥신경망인가?

BP의 문제점과 극복 방법

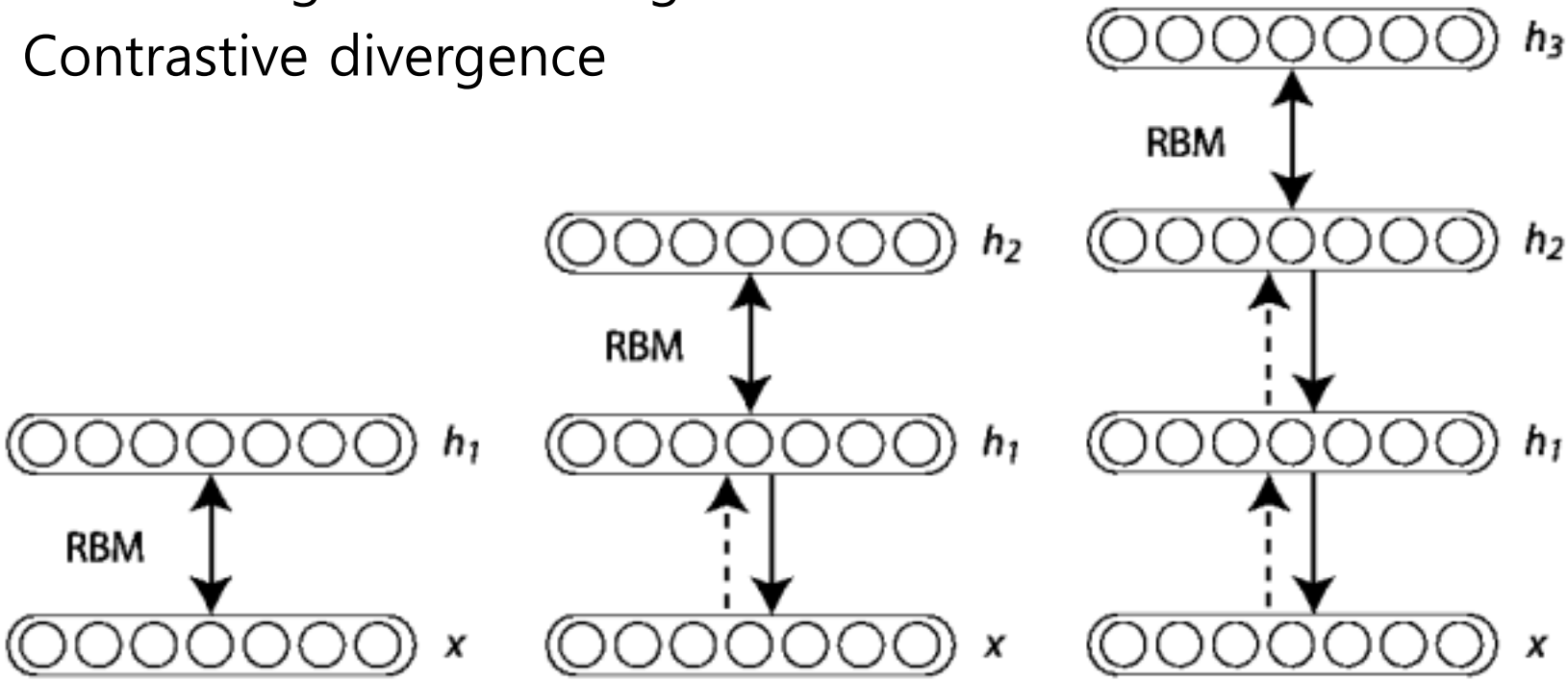
- **Limitations**
  - Backpropagation (BP) barely changes lower-layer parameters (vanishing gradient)
  - Therefore, deep networks cannot be fully (effectively) trained with backpropagation



- **Breakthrough**
  - Deep belief networks (unsupervised pre-training)
  - Convolutional neural networks (reducing redundant parameters)
  - Rectified linear unit (constant gradient propagation)

Three Innovations

- **Innovations:**
  - Greedy layer-wise training
  - Pre-training + fine tuning
  - Contrastive divergence



History of Neural Network Research

Neural network
Back propagation

Deep belief net
Science

Speech

The New York Times
Google Hong Kong
IMAGENET
Microsoft

1986 2006 2011 2012

deep learning results

- Unsupervised & Layer-wised pre-training

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	

ing
blems
n
P, HOG)
ectures

(2 GPU)

Deep Networks Advance State of Art in Speech

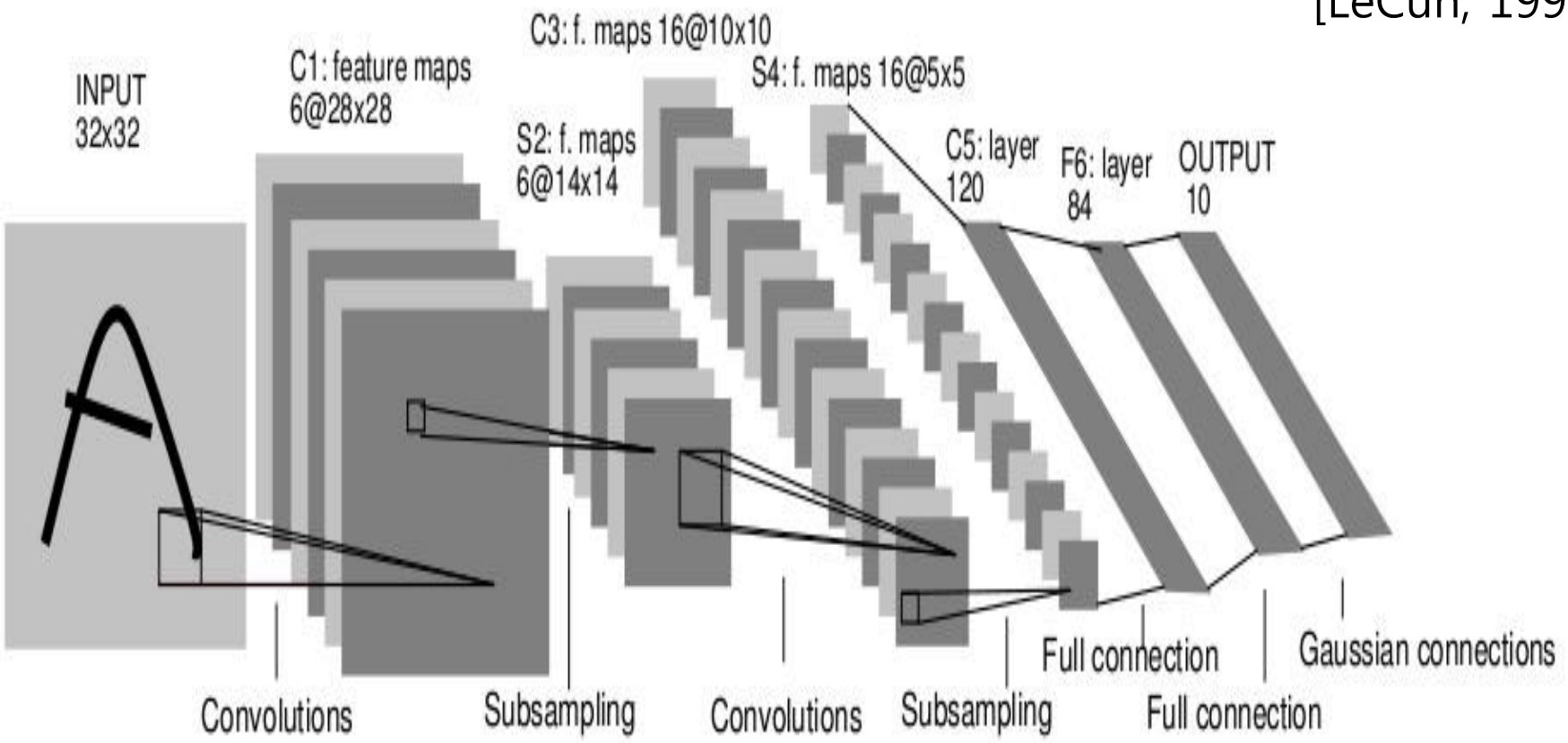
Deep Learning leads to breakthrough in speech recognition at MSR.

Microsoft

4. 어떤 딥아키텍쳐들이 있는가?

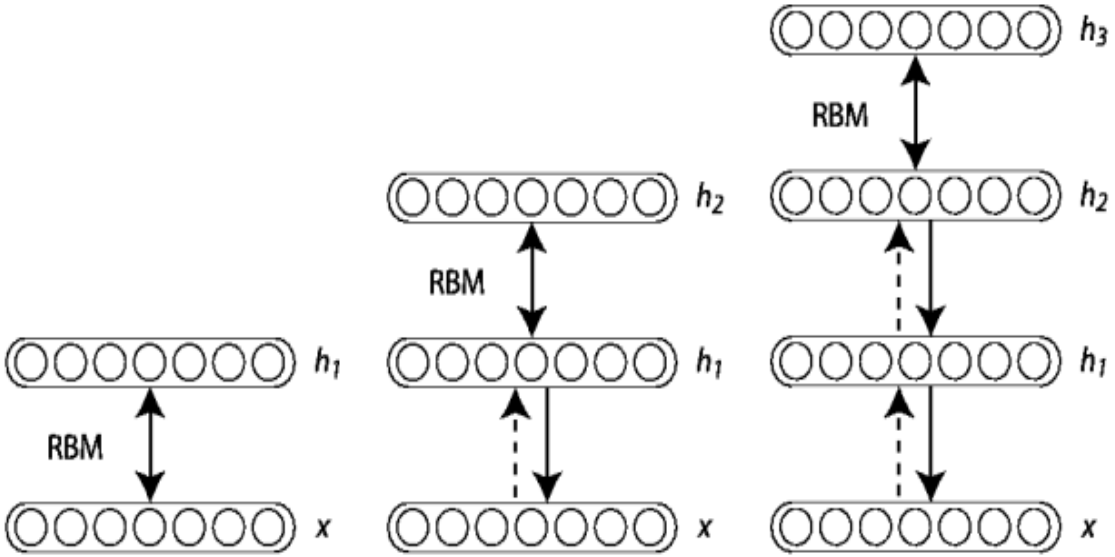
딥러닝 아키텍쳐 1:
Convolutional Neural Nets (CNN)

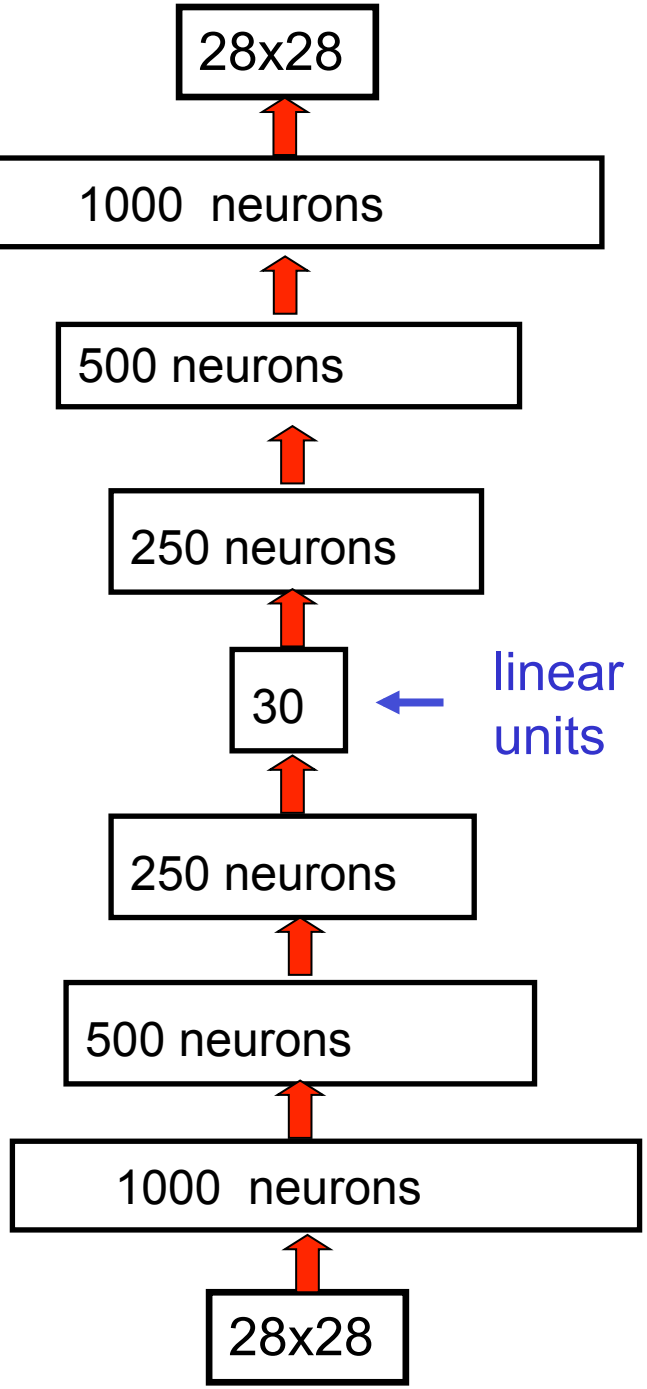
[LeCun, 1998]



딥러닝 아키텍쳐 2: Deep Belief Networks (DBN)

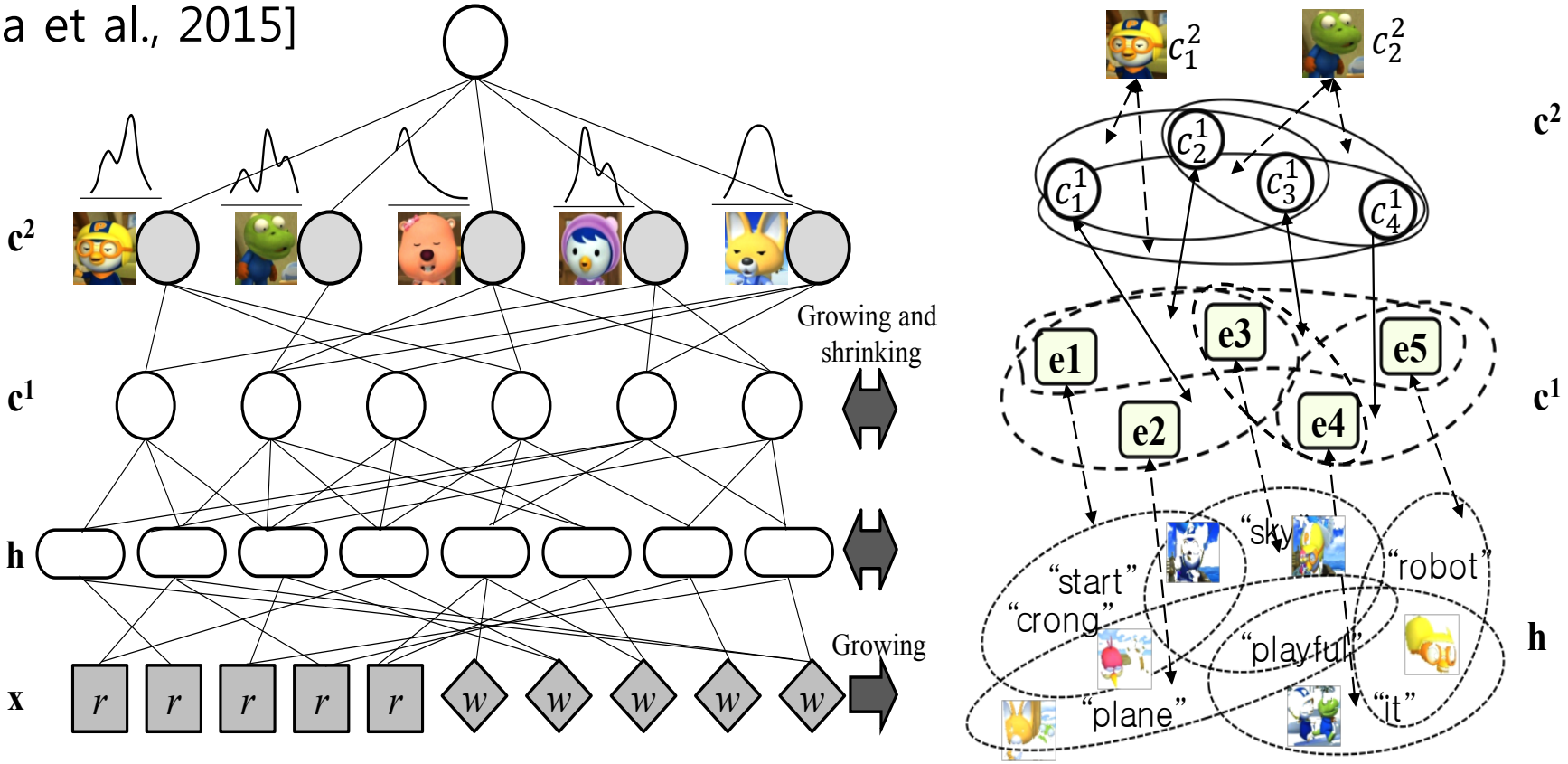
[Hinton et al., 2006]





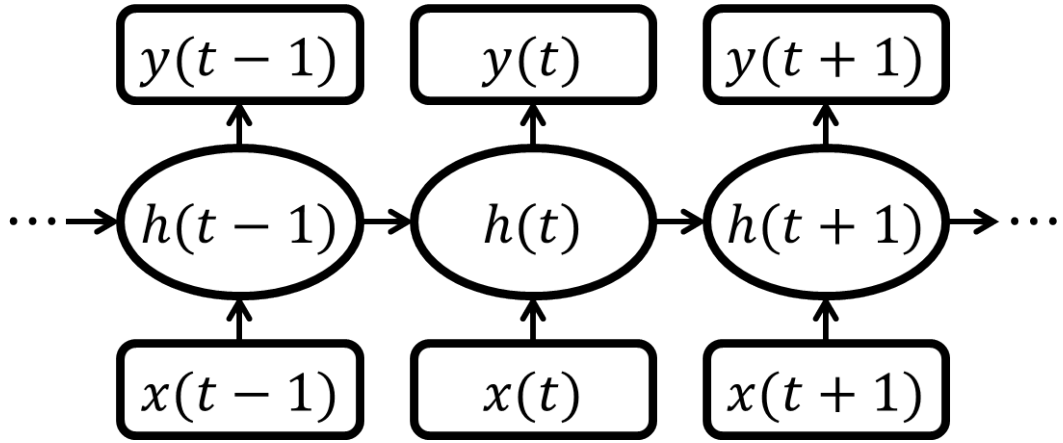
딥러닝 아키텍쳐 3:
Deep Hypernetworks (DHN)

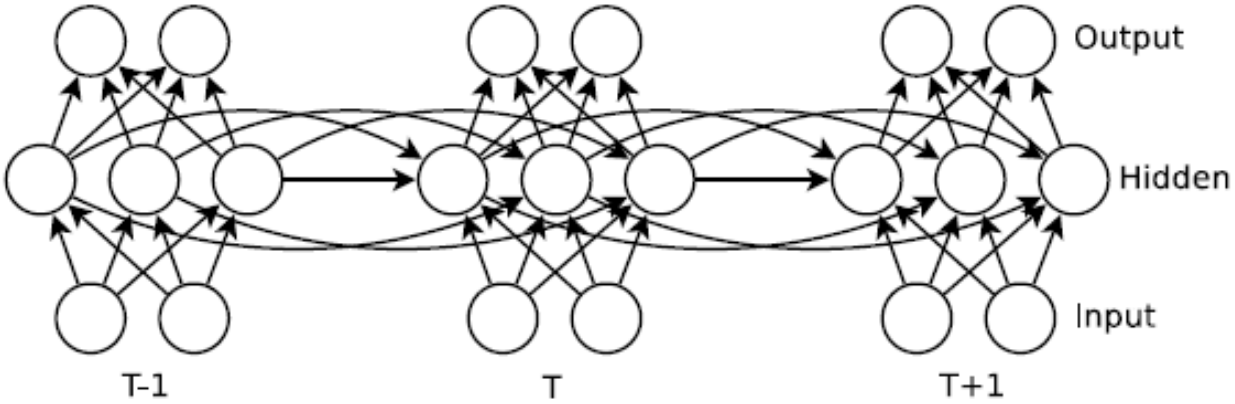
[Zhang et al., 2012]
[Ha et al., 2015]



(a) Example of deep concept hierarchy learned from Pororo videos (b) Hypergraph representation of (a)

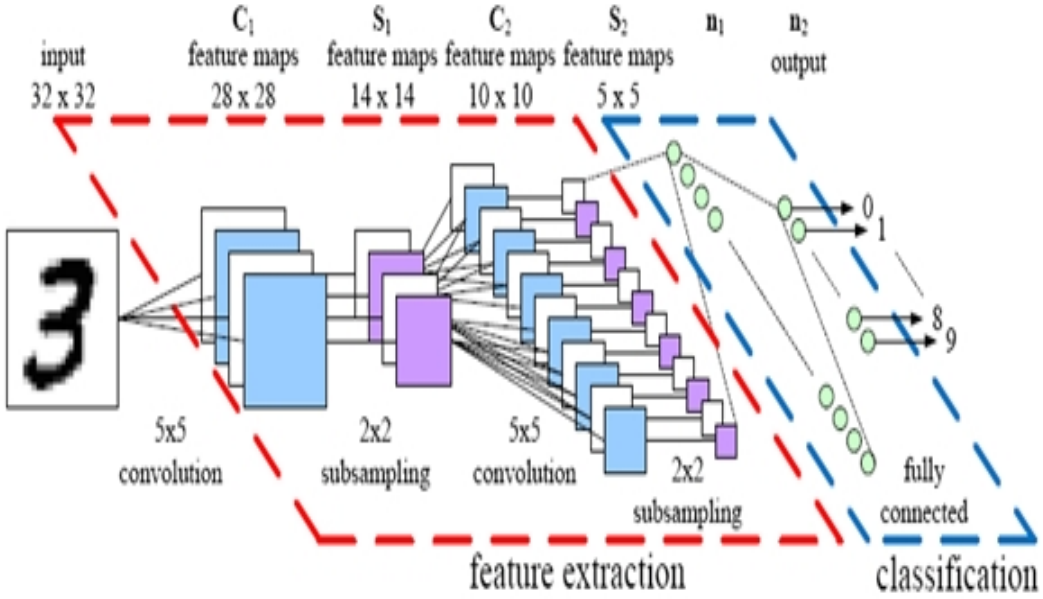
딥러닝 아키텍쳐 4:
Recurrent Neural Networks (RNNs)

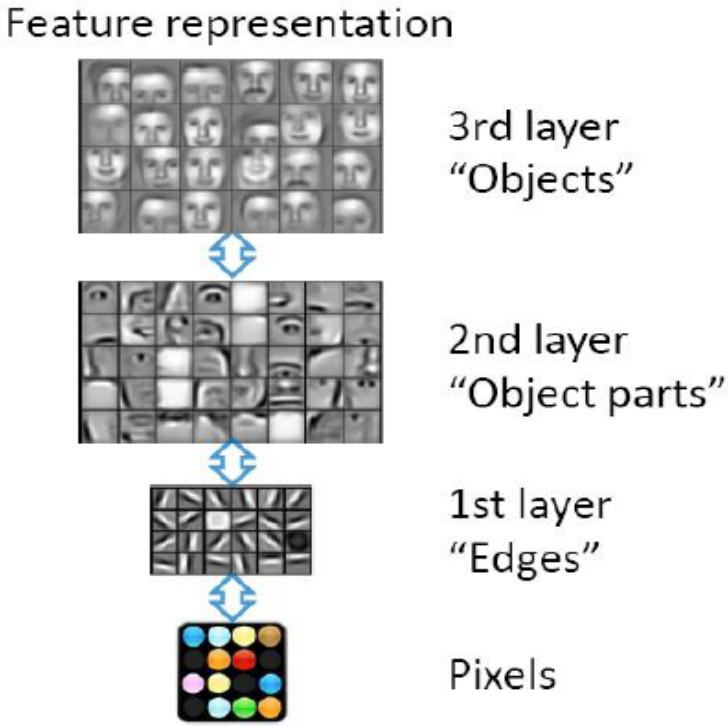




CNN (1/3): Architecture

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

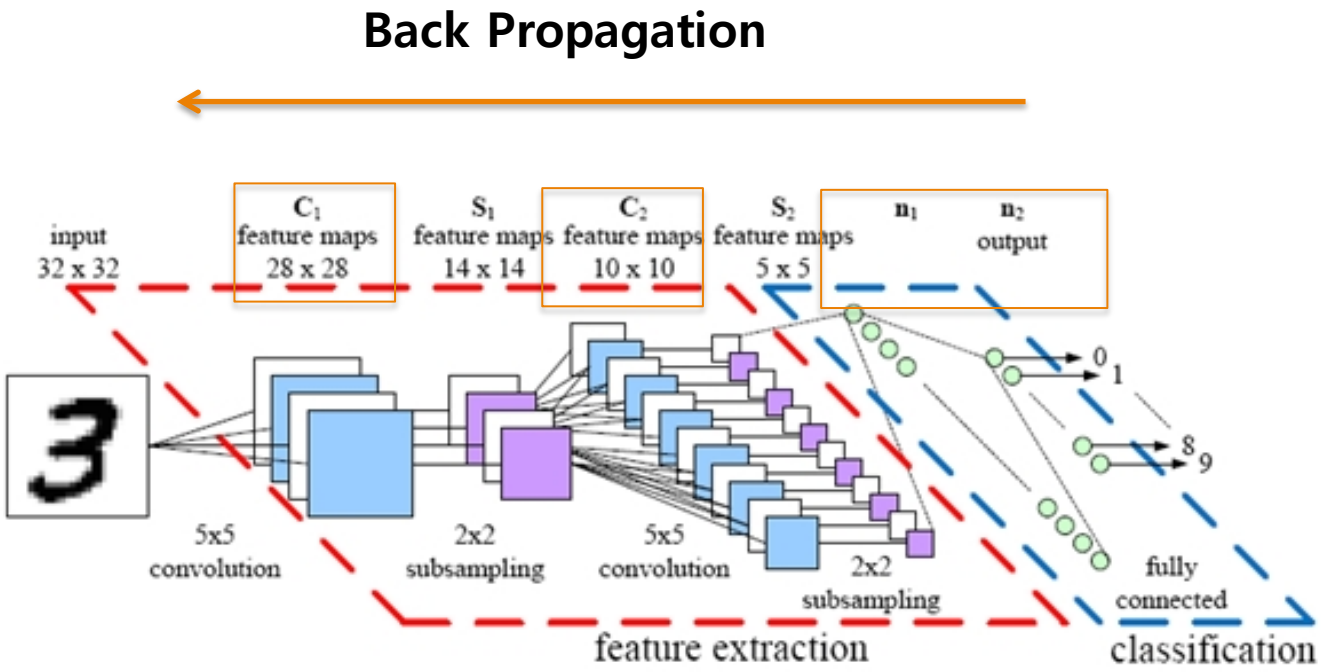




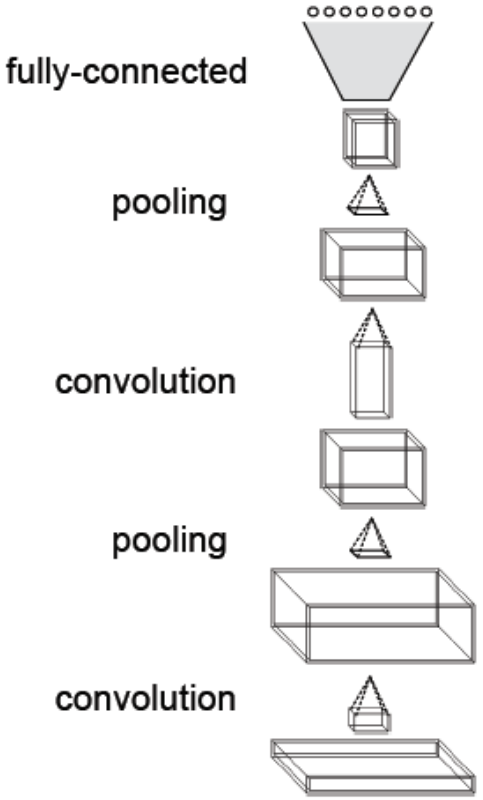
http://parse.ele.tue.nl/education/cluster2

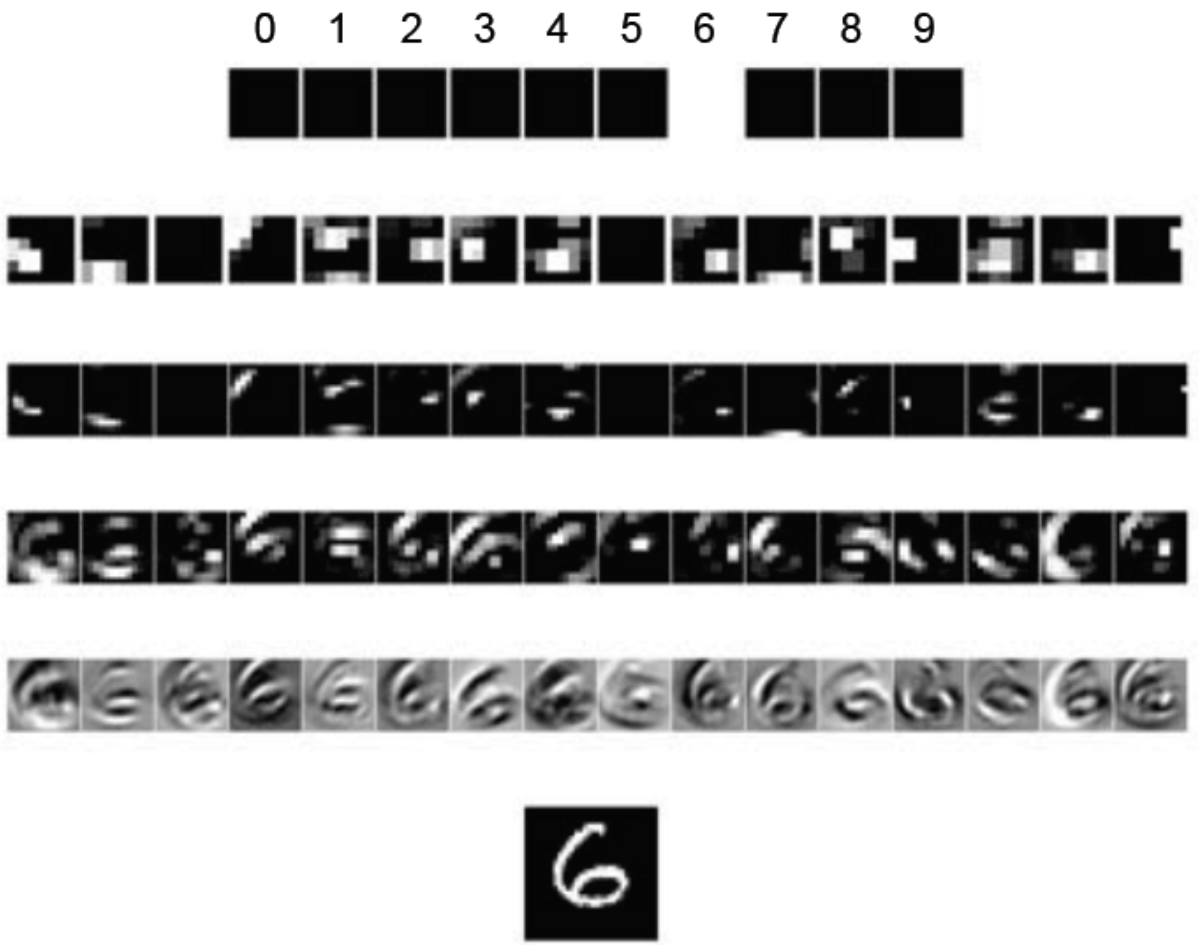
CNN (2/3): Learning

- **CNN is just another neural network with sparse connections**
- **Learning algorithm:**
  - Backpropagation on convolution layers and fully-connected layers



Behavior of CNN





CNN (3/3): Applications (Image Classification)

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

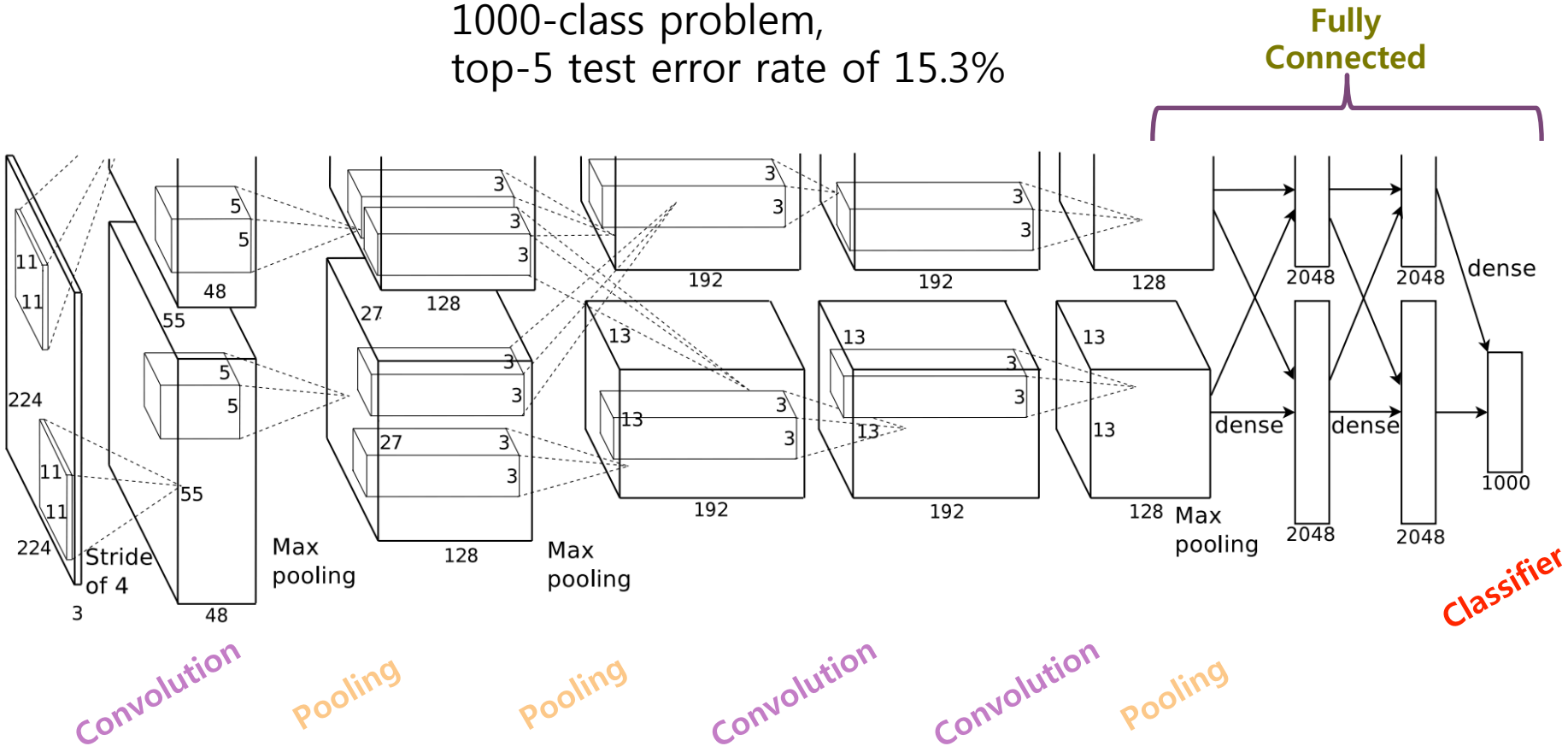
1. Clarifi (**0.117**): **Deep Convolutional Neural Networks (Zeiler)**
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional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

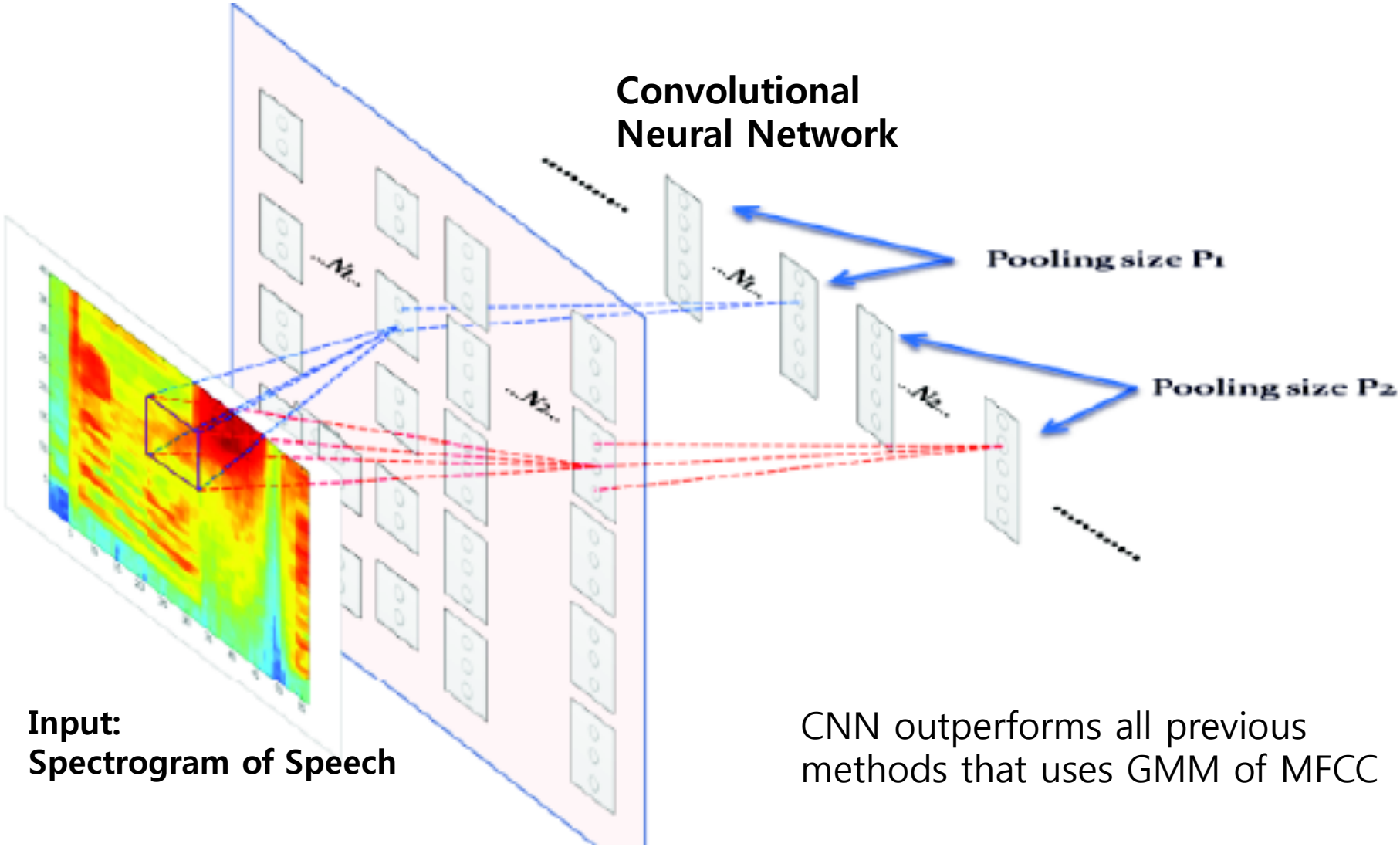
CNN Application: Image Classification

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

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



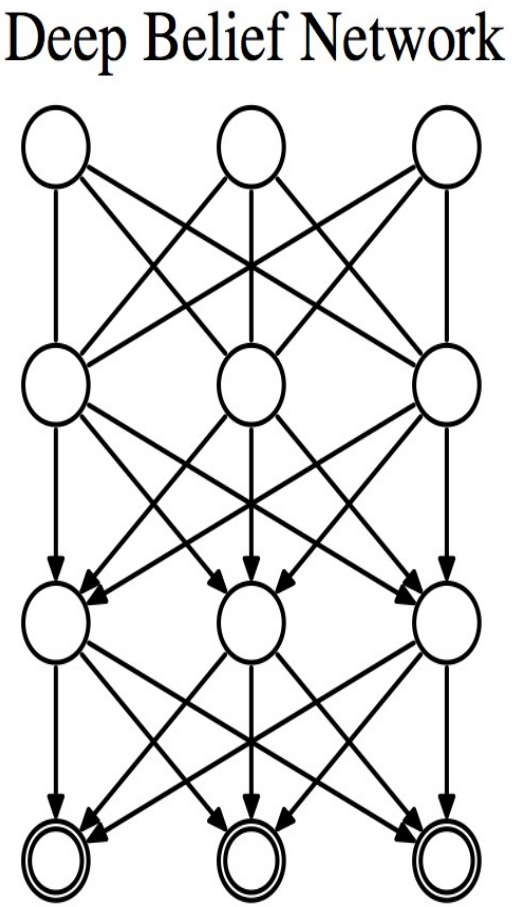
CNN Application: Speech Recognition



CNN outperforms all previous methods that uses GMM of MFCC

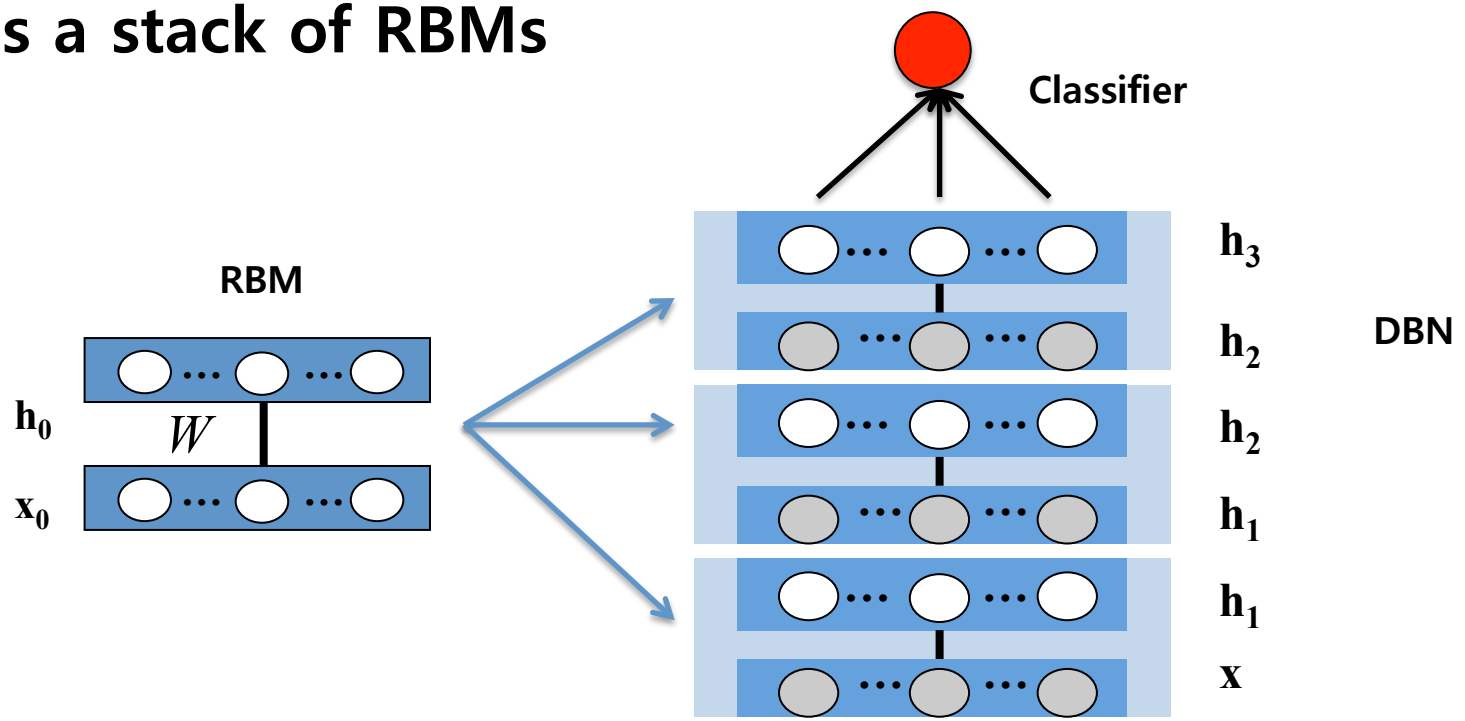
DBN (1/3): Architecture

- **Deep Belief Networks (Deep Bayesian Networks)**
  - Bayesian networks that have similar structure to neural networks
  - Generative model
  - Also, can be used as classifier (with additional classifier at top layer)
  - Resolves gradient vanishing by pre-training
  - There are two modes (classifiers & auto-encoders), but we only consider classifiers here



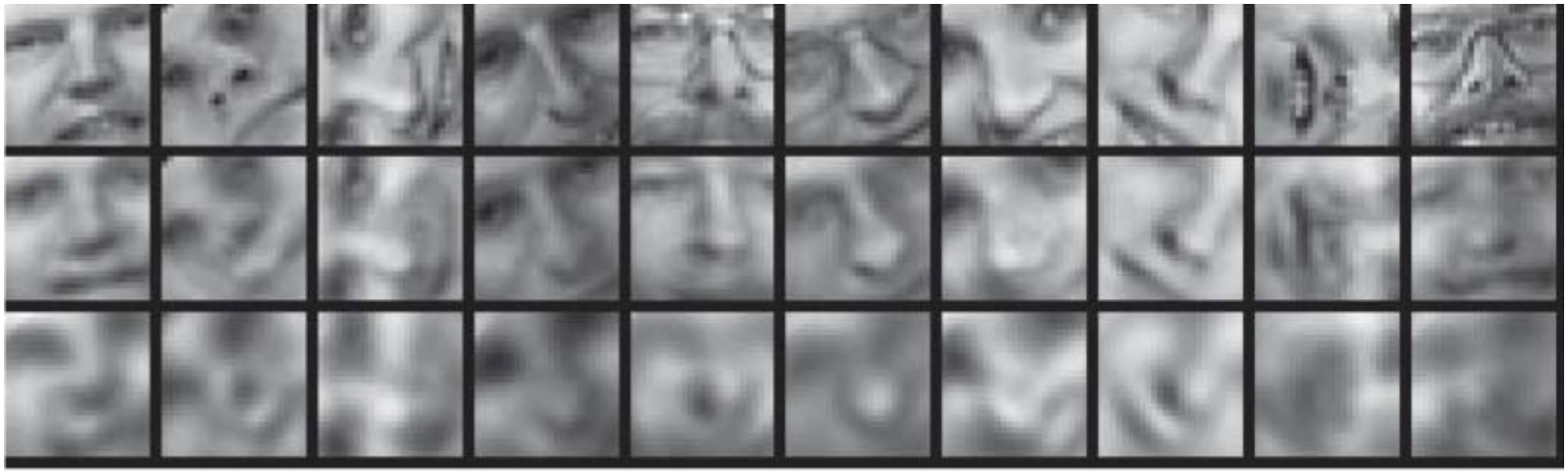
DBN (2/3): Learning

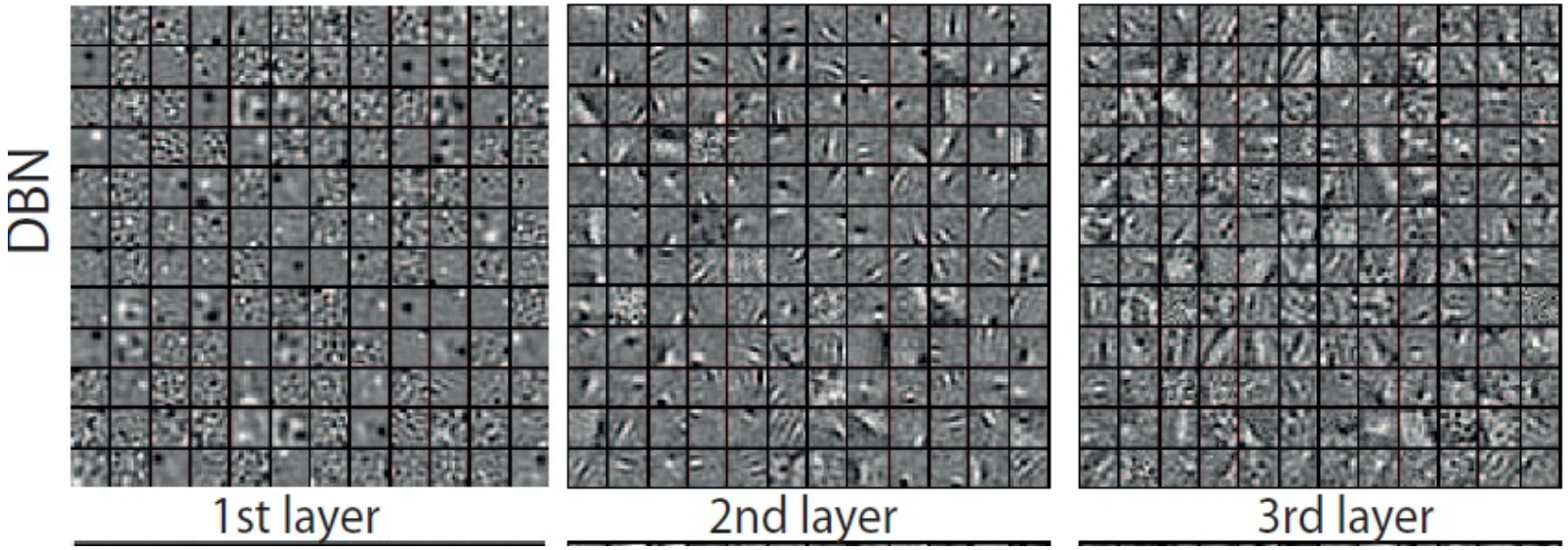
- **DBN as a stack of RBMs**



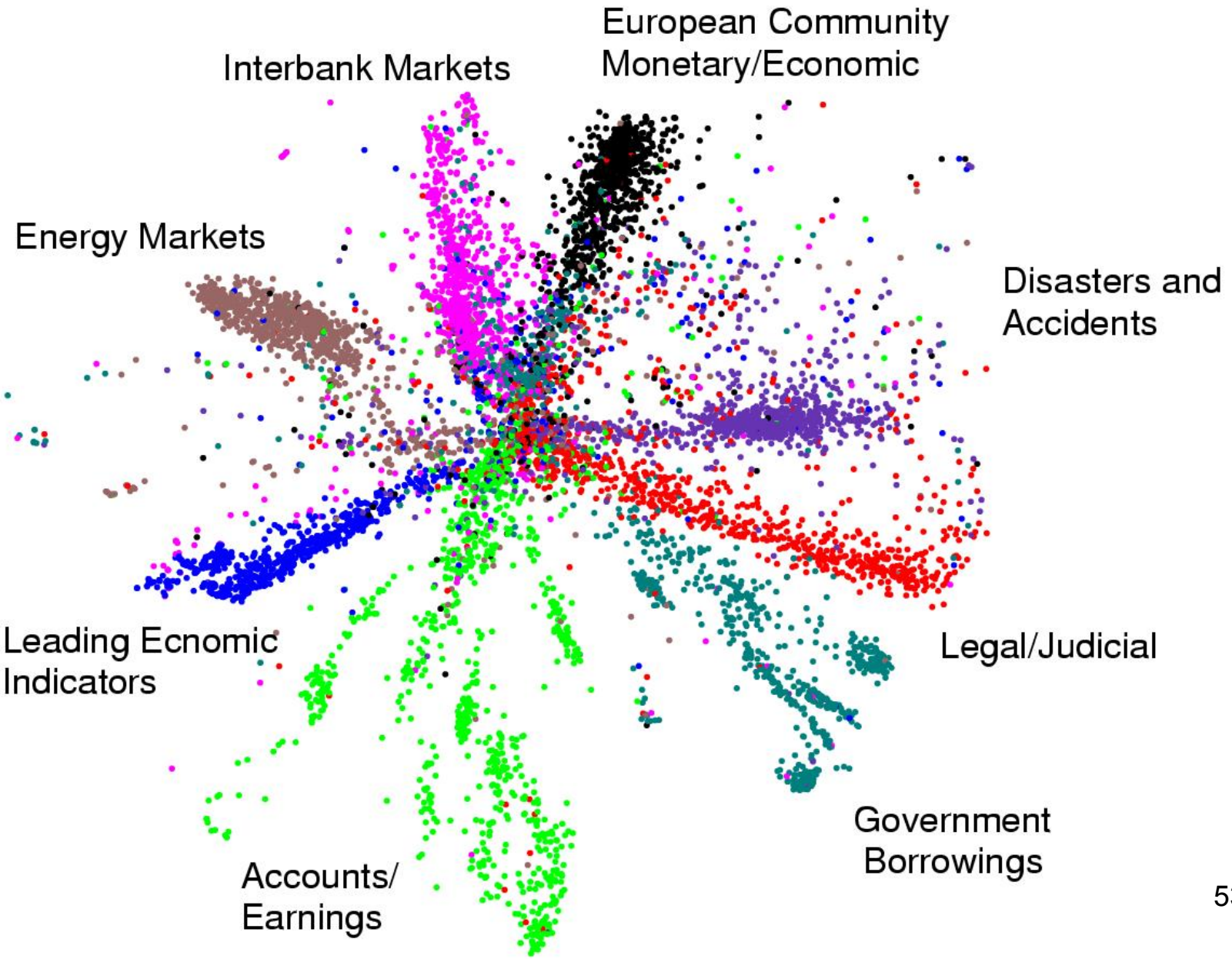
1. Regard each layer as RBM
2. Layer-wise pre-train each RBM in unsupervised way
3. Attach the classifier and fine-tune the whole network in supervise d way

DBN (3/3): Applications





First compress all documents to 2 numbers. Then use different colors for different document categories



Learning: 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

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

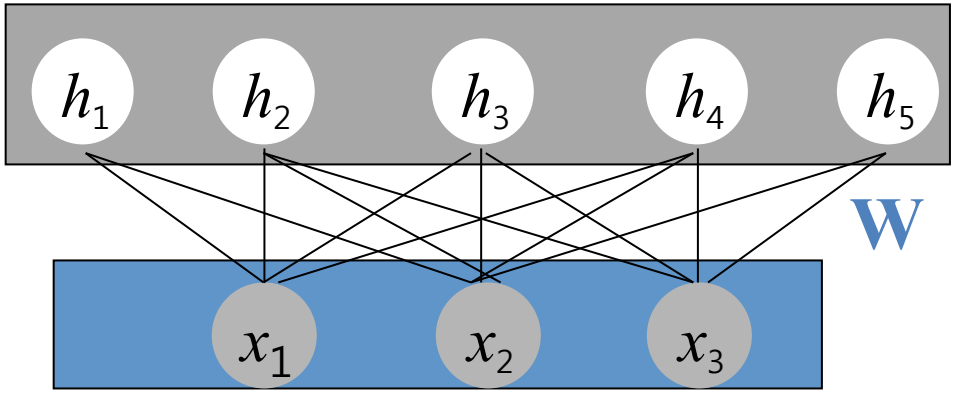
Marginal (x) Probability, or Likelihood

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

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

Conditional Probability

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



Remark:

- Conditional independence

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

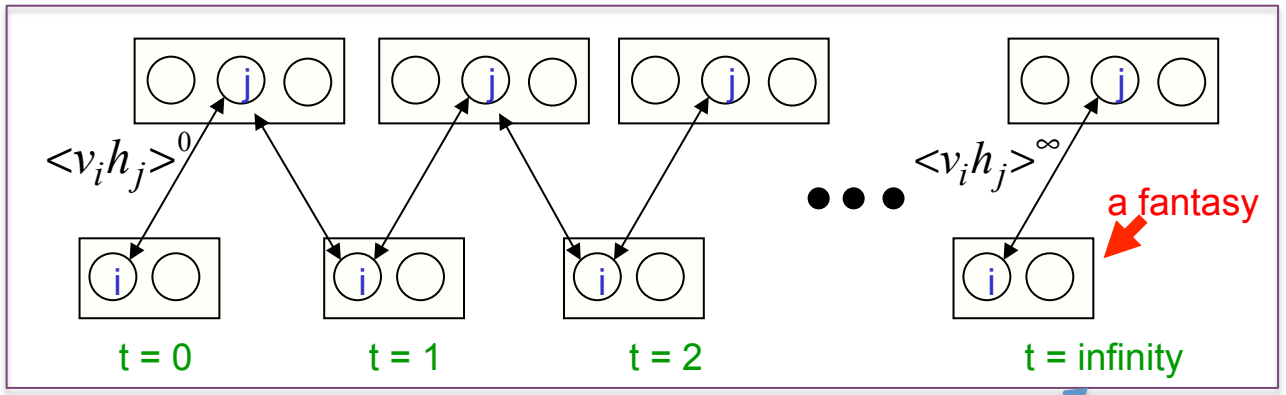
$$P(\mathbf{x}\,|\,\mathbf{h}) = \prod_j P(x_j\,|\,\mathbf{h})$$

- Conditional probability is the same as neural network

Learning: Unsupervised Learning of RBM

- **Maximum Likelihood**
  - Use gradient descent

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



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

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

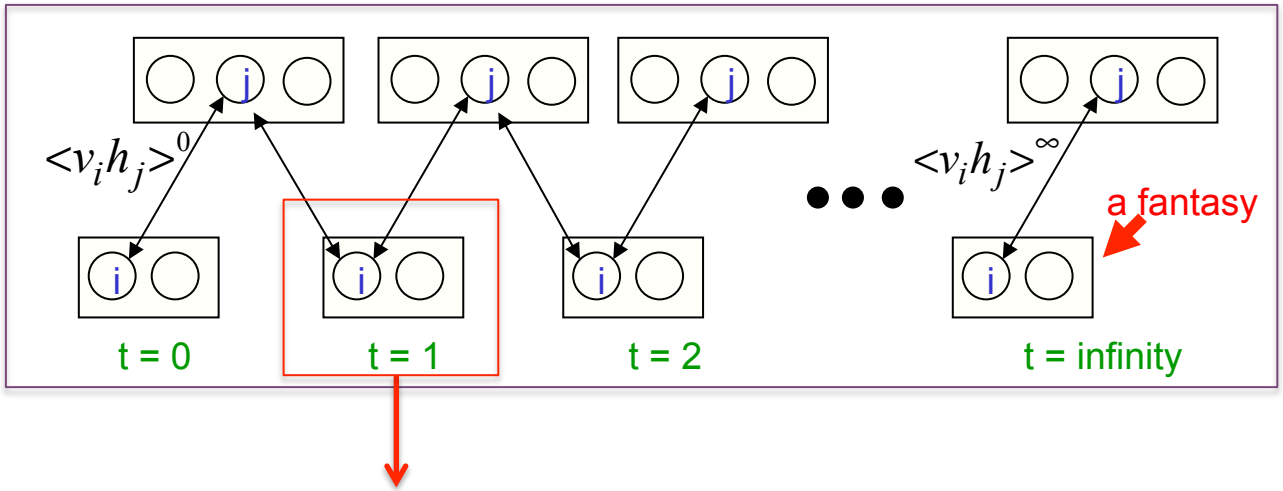
$$\approx \left\langle x_i h_j \right\rangle_{1} - \left\langle x_i h_j \right\rangle_{0}$$

Distribution of Dataset

Distribution of Model

Learning: Contrastive Divergence (CD)

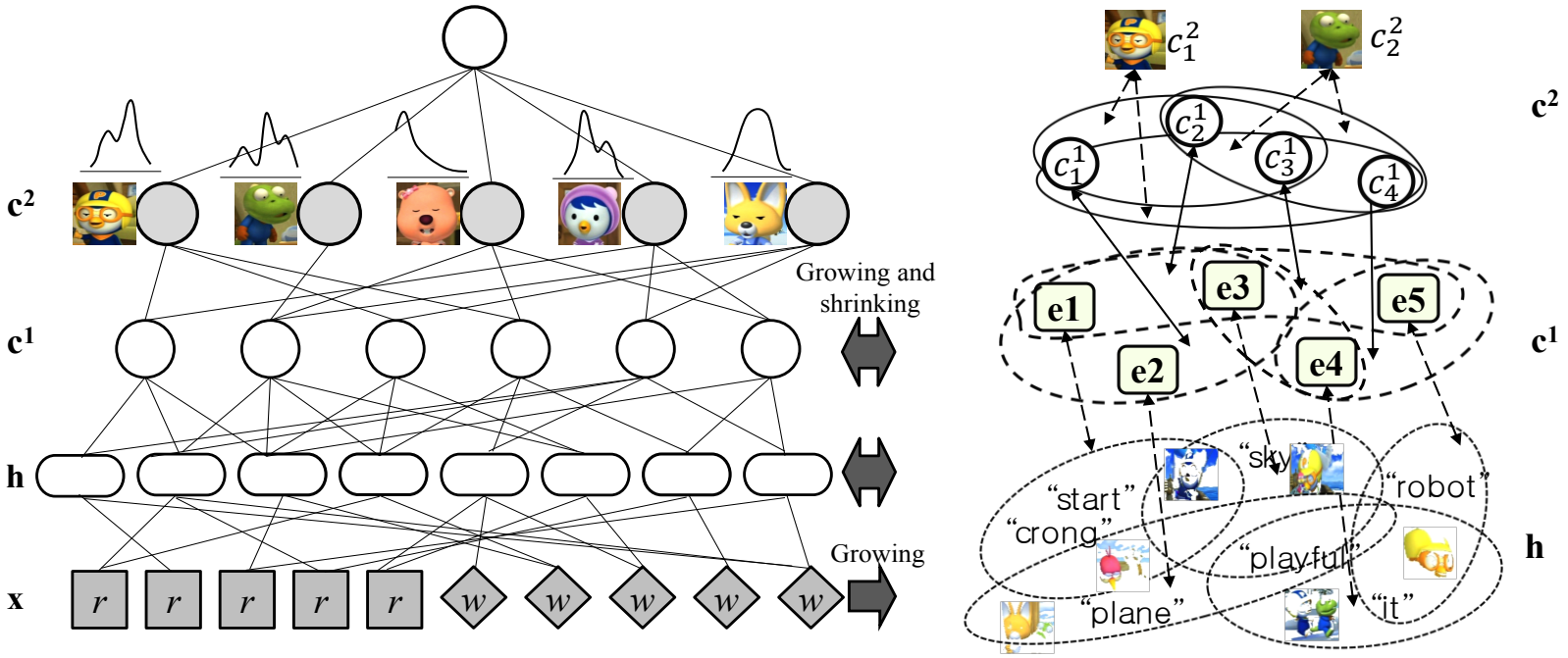
- **k-Contrastive Divergence Trick**
  - From the previous slide, to get distribution of model, we nee d to calculate many Gibbs sampling steps
  - And this is per a single parameter update
  - Therefore, we take the sample after only k-steps where in pra ctice, k=1 is sufficient



DHN (1/3): Architecture

- **Deep concept hierarchy of cartoon videos**
  - A concept hierarchy using SPC models (Zhang et al., CogSci-2012)
  - Empirical distribution of a scene-utterance pair

$$P(\mathbf{r},\mathbf{w}\,|\,\mathbf{c}^1,\mathbf{c}^2)=\sum_{\mathbf{h}} P(\mathbf{r},\mathbf{w}\,|\,\mathbf{h},\mathbf{c}^1,\mathbf{c}^2)P(\mathbf{h}\,|\,\mathbf{c}^1,\mathbf{c}^2)$$



(a) Example of deep concept hierarchy learned from Pororo videos (b) Hypergraph representation of (a)

DHN (2/3): Learning

- **Posterior distribution**

[Ha et al., AAAI-2015]

$$P_t(\mathbf{h},\mathbf{c}^1 \mid \mathbf{r},\mathbf{w},\mathbf{c}^2) = \frac{P(\mathbf{r},\mathbf{w} \mid \mathbf{h},\mathbf{c}^1,\mathbf{c}^2)P(\mathbf{c}^2 \mid \mathbf{c}^1,\mathbf{h})P_{t-1}(\mathbf{h},\mathbf{c}^1)}{P(\mathbf{r},\mathbf{w},\mathbf{c}^2)}$$

- **Empirical distribution**

$$P_t(\mathbf{h},\mathbf{c}^1 \mid \mathbf{r},\mathbf{w},\mathbf{c}^2) \propto \prod_{d=1}^{D_t}\left\{P(\mathbf{r}^{(d)},\mathbf{w}^{(d)} \mid \mathbf{h},\mathbf{c}^1,\mathbf{c}^2)P(\mathbf{c}^2 \mid \mathbf{c}^1)P(\mathbf{c}^1 \mid \mathbf{h})P_{t-1}(\mathbf{h})\right\}$$

  - 1st term: Data reconstruction from the models

$$\log P(\mathbf{r}^{(d)},\mathbf{w}^{(d)} \mid \mathbf{c}^2,\mathbf{c}^1,\mathbf{h}) = \sum_{n=1}^{N}\log P(r_n^{(d)} \mid \mathbf{c}^2,\mathbf{c}^1,\mathbf{h}) + \sum_{m=1}^{M}\log P(w_m^{(d)} \mid \mathbf{c}^2,\mathbf{c}^1,\mathbf{h})$$

$$P(w_m^{(d)}=1 \mid \mathbf{c}^2,\mathbf{c}^1,\mathbf{h}) = \exp\left(s_m^{\mathbf{w}} - \sum_{i=1}^{|\mathbf{e}^{\mathbf{c}}|}\alpha_i\right) \quad P(r_n^{(d)}=1 \mid \mathbf{c}^2,\mathbf{c}^1,\mathbf{h}) = \exp\left(s_n^{\mathbf{r}} - \sum_{i=1}^{|\mathbf{h}^{\mathbf{c}}|}\alpha_i\right)$$

$$\mathbf{s}^{\mathbf{w}} = \sum_{i=1}^{|\mathbf{e}^{\mathbf{c}}|}\alpha_i e_i^{\mathbf{w}} \quad \text{and} \quad \mathbf{s}^{\mathbf{r}} = \sum_{i=1}^{|\mathbf{e}^{\mathbf{c}}|}\alpha_i e_i^{\mathbf{r}}$$

  - 2nd term: predicting characters from the clusters
  - 3rd term: similarity of each cluster
  - 4th term: model complexity

- **Incremental weight update**

$$\alpha_i = \sum_{d=1}^{D}\left\{g(e_i)f(\mathbf{r}^{(d)},\boldsymbol{w}^{(d)};e_i)\right\}, \quad \alpha_i^t = \lambda\alpha_i + (1-\lambda)\alpha_i^{t-1}$$

$$f(\mathbf{r}^{(d)},\boldsymbol{w}^{(d)};e_i) = \begin{cases} 1, \text{ if } \left(\mathbf{r}^{(d)}\cdot e_i^{\mathbf{r}} + \mathbf{w}^{(d)}\cdot e_i^{\mathbf{w}}\right) \Big/ e_i e_i^{\mathrm{T}} > \kappa \\ 0, \text{ otherwise} \end{cases}$$

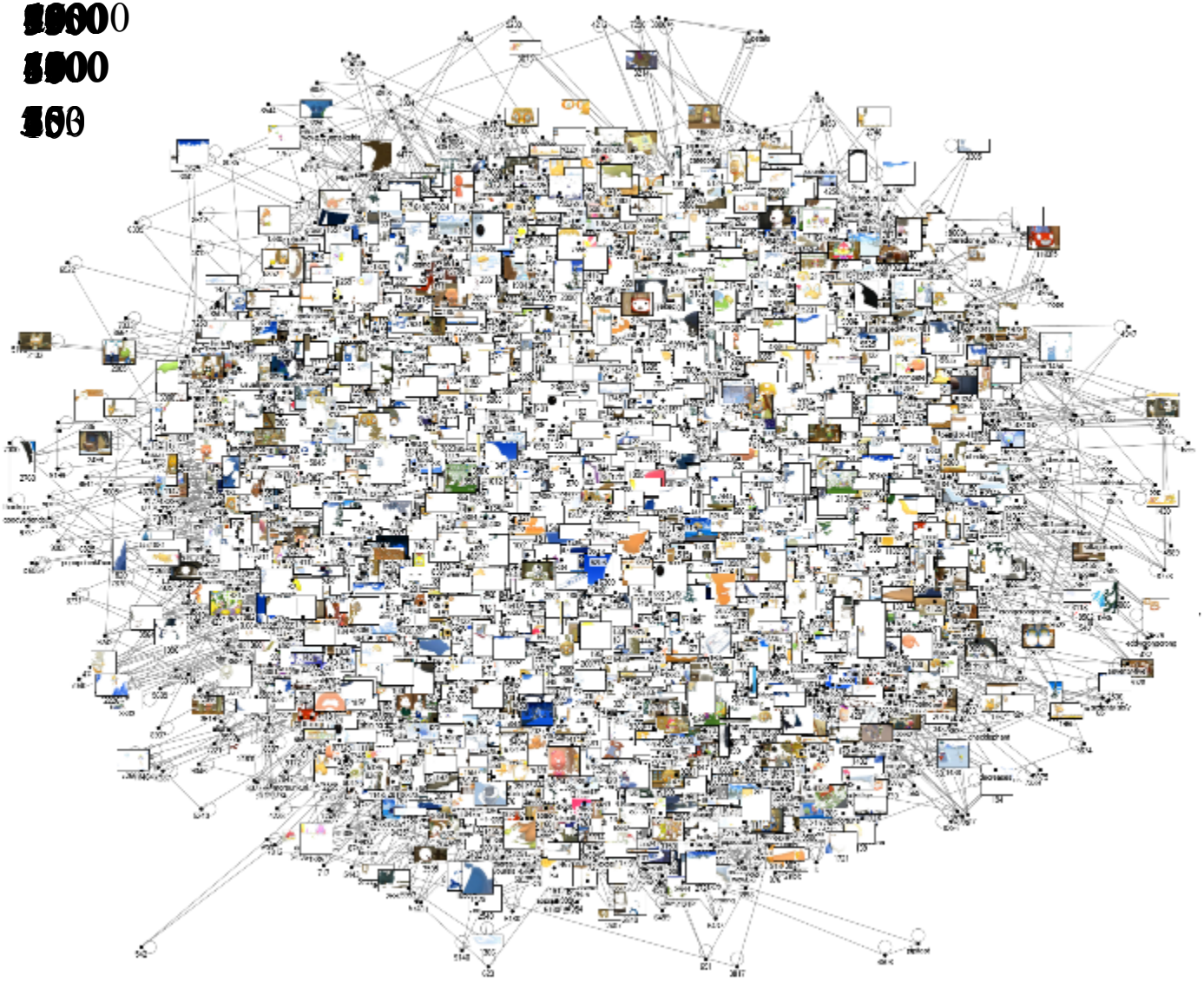
DHN (3/3): Applications

[Zhang et al., CogSci-2012]

[Ha et al., AAAI-2015]

Application: 뽀로로 만화영화 학습

Image 개수 : [illegible]0
Word 개수 : [illegible]
Episode 개수 : [illegible]



Application: 대사로부터 장면의 상상

- **Intermediate images generated from query sentences**

Query sentences	Episodes 1~52 (1 season)	Episodes 1~104 (2 seasons)	Episodes 1~183 (all seasons)
• Tongtong, please change this book using magic. • Kurikuri, Kurikuri-tongtong!			
• I like cookies. • It looks delicious • Thank you, loopy			

[Ha et al., AAAI-2015]

Application: 장면으로부터 대사의 생성

Query = {i, try}

Original subscript: clock, I have made another potion come and try it

Generated subscripts

- **as i don't have the right magic potion come and try it was nice**
- **ah, finished i finally made another potion come and try it we'll all alone?♪**

Query = {thanks, drink}

Original subscript: Oh, thanks Make him drink this

Generated subscripts

- **oh, thanks make him drink this bread**
- **oh, thanks make him drink this forest**

Query = {he, take}

Original: Tong. Tong Let. Me. Take. It. To. Clock

Generated

- **take your magic to know what is he doing?**
- **take your magic to avoid the house to know what is he keeps going in circles like this will turn you back to normal**

Query = {ship, pulled}

Original: The ship is being pulled

Generated

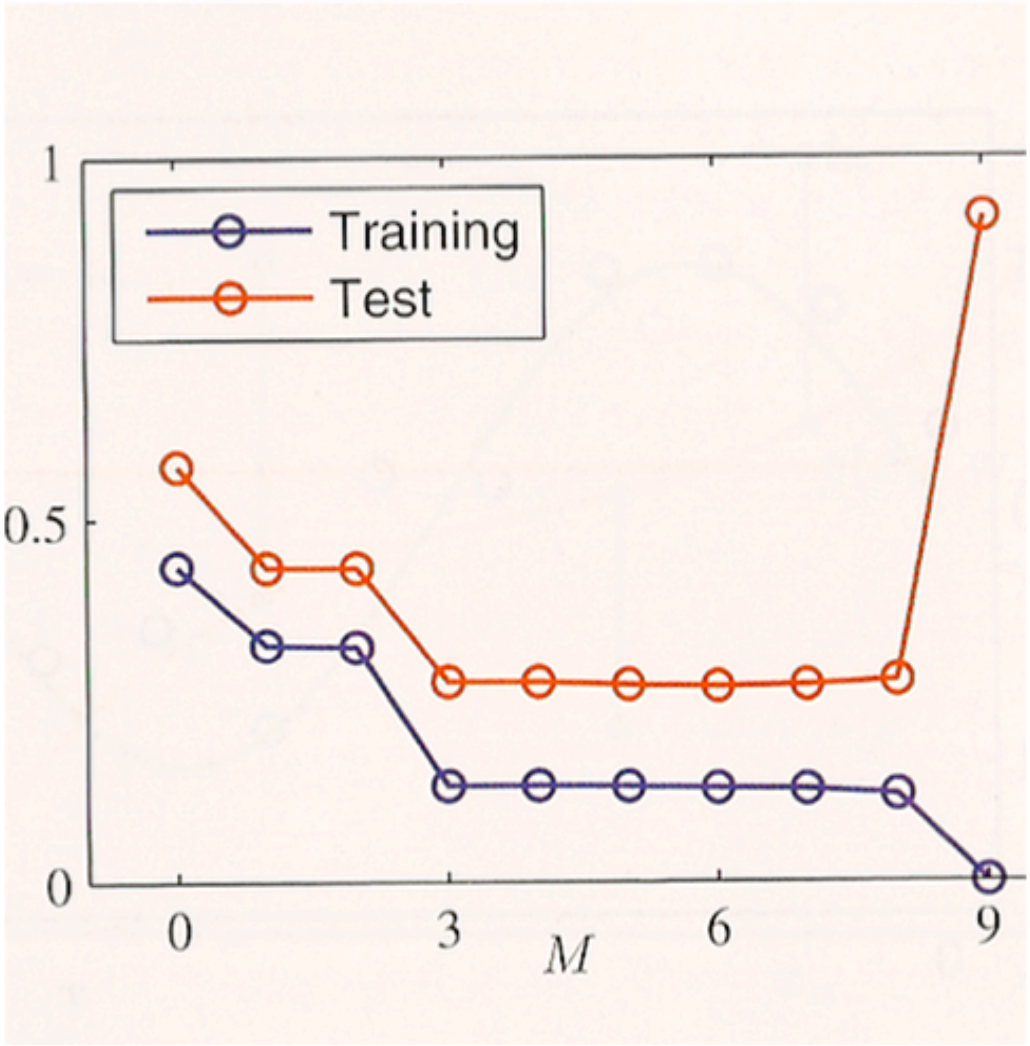
- **wow looks as if that's the ship is being pulled**
- **the ship is being pulled**

5. 딥 모델 설계시 유의점은?

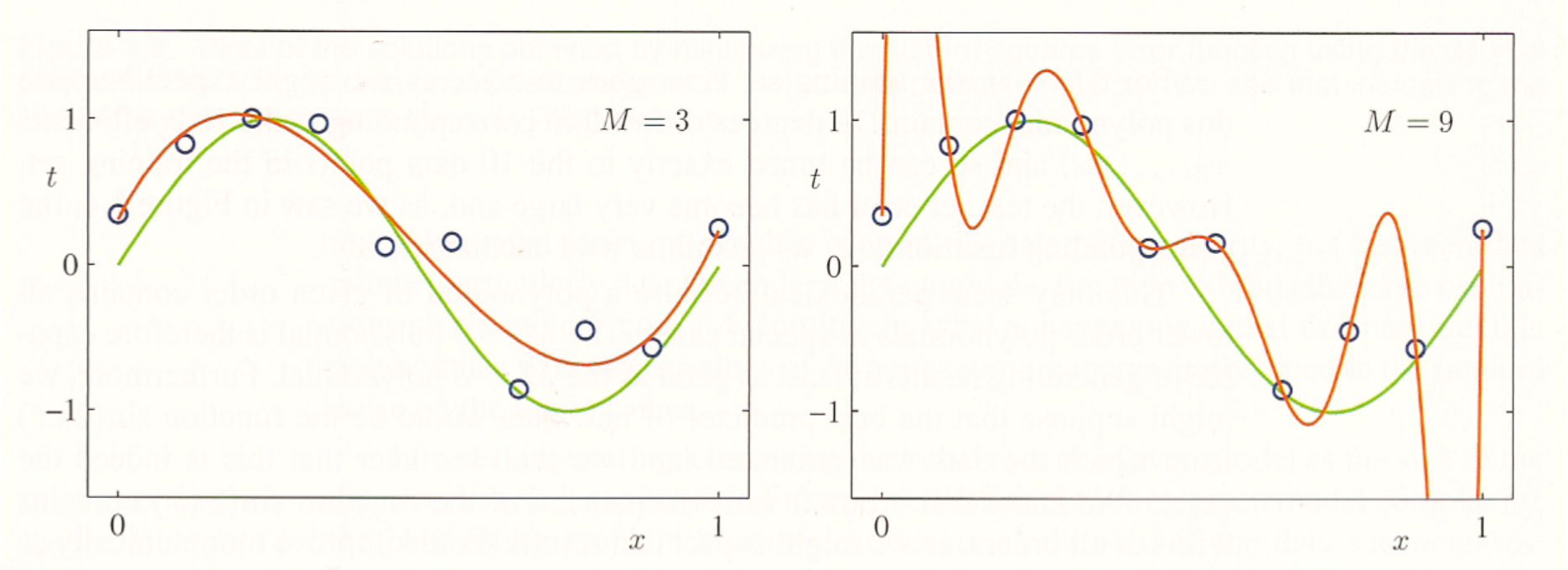
학습 이론

- 과다학습
- 모델복잡도
- Occam's Razor
- 정규화
- SRM
- MAP
- MDL

과다학습 (Overfitting)



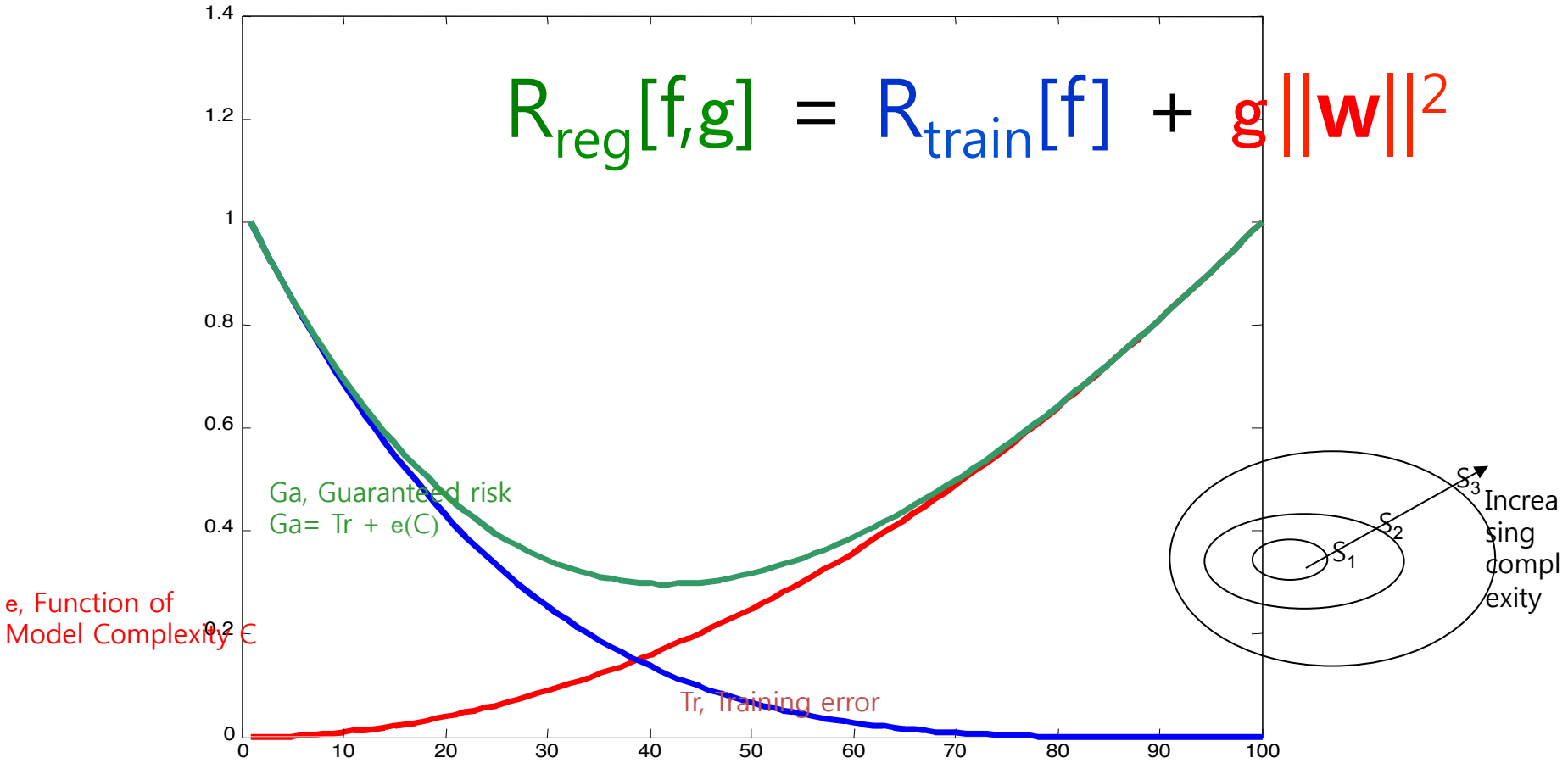
모델 복잡도



Occam's Razor

- Principle proposed by William of Ockham in the fourteenth century: "Pluralitas non est ponenda sine neccesitate".
- Of two theories providing similarly good predictions, prefer *the simplest one*.
- Shave off unnecessary parameters of your models.

Regularization & Structural Risk Minimization (SRM)



Bayesian MAP vs. SRM

- **Maximum A Posteriori (MAP):**

$$f = \text{argmax } P(f|D)$$
$$= \text{argmax } P(D|f)\ P(f)$$
$$= \text{argmin } \underbrace{-\log P(D|f)}_{\text{Negative log likelihood} = \text{Empirical risk } R_{emp}[f]} \quad \underbrace{-\log P(f)}_{\text{Negative log prior} = \text{Regularizer } W[f]}$$

- **Structural Risk Minimization (SRM):**

$$f = \text{argmin } R_{emp}[f] + W[f]$$

Minimum Description Length (MDL)

- MDL: minimize the total length of the "message".
- Two part code: transmit the model and the residual.
- $f = \text{argmin} -\log_2 P(D|f) -\log_2 P(f)$

Residual: length of the shortest code to encode the data given the model

Length of the shortest code to encode the model (model complexity)

Bias-Variance Tradeoff

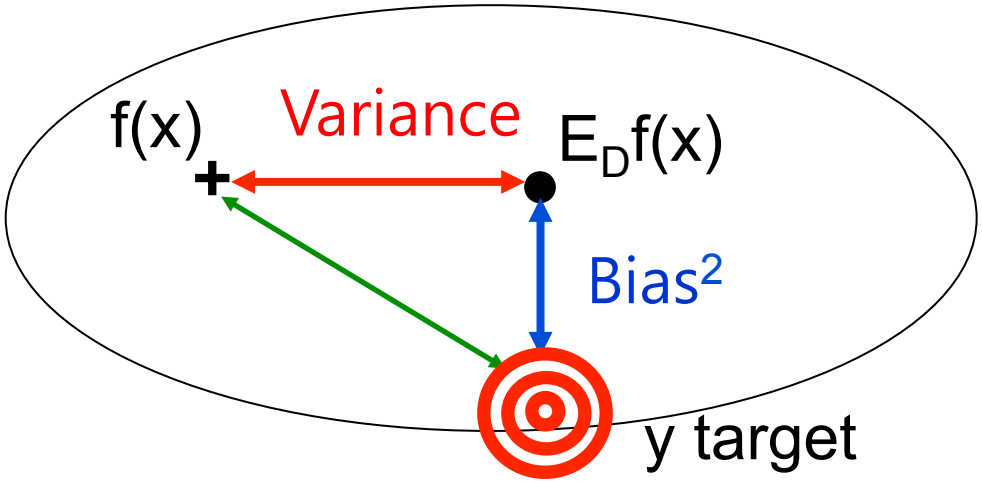
- f trained on a training set D of size m (m fixed)
- For the square loss:

$$E_D[f(x)-y]^2 = [E_D f(x)-y]^2 + E_D[f(x)-E_D f(x)]^2$$

Expected value of the loss over datasets D of the same size

Bias^2

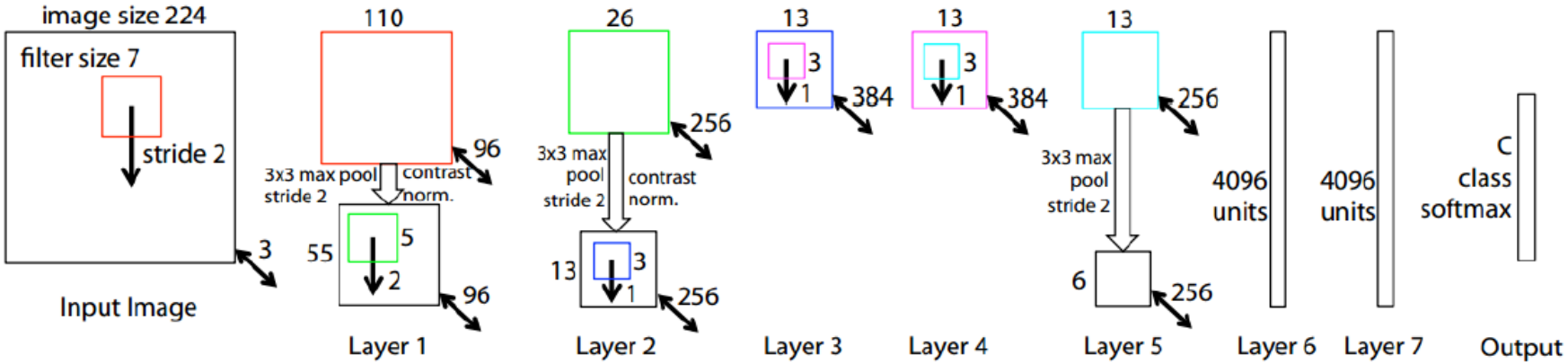
Variance



학습이론의 딥러닝 적용 사례

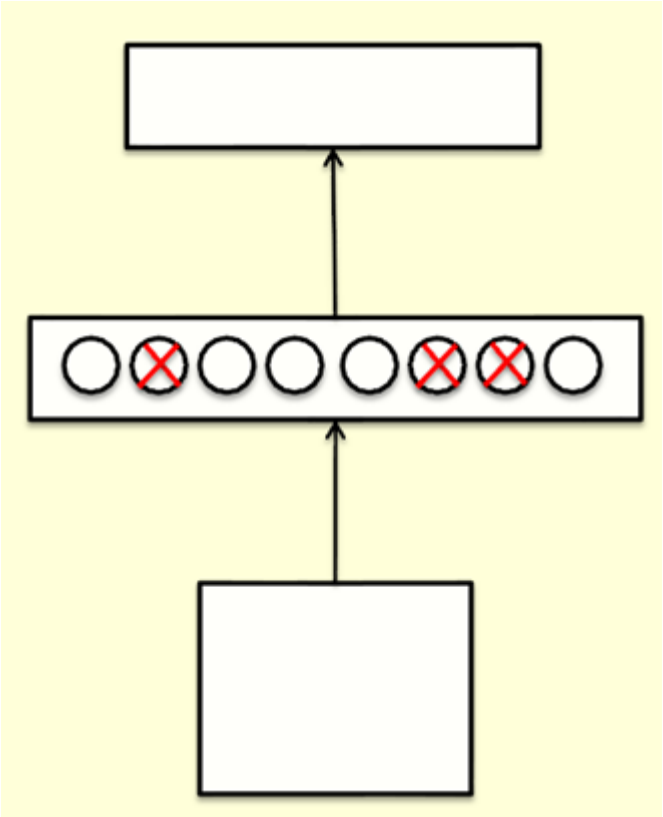
Convolutional Neural Networks - AlexNet

- New Elements
  - Dropout – Approximation of Exponential Number of Ensemble
    - Prevent overfitting
  - ReLU – Novel Non-Linear Unit
    - Prevent vanishing gradient
  - Normalization – Whitening
    - Smooth parameter search space
  - Deep – Compact Representation
    - Powerful inductive bias on classifier
    - 7 layers, 60M parameters



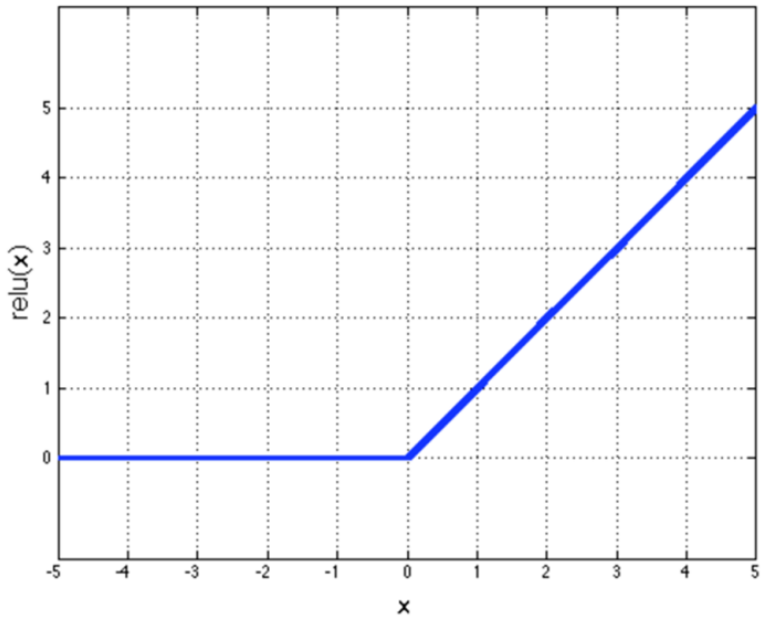
Dropout: An efficient way to average many large neural nets.

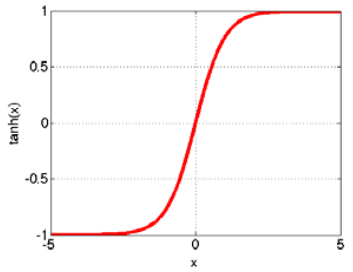
- Consider a neural net with on hidde n layer
- Each time we present a training exa mple, we randomly omit each hidde n unit with probability 0.5.
- So we are randomly sampling from 2^H different architectures
  - All architectures share weights

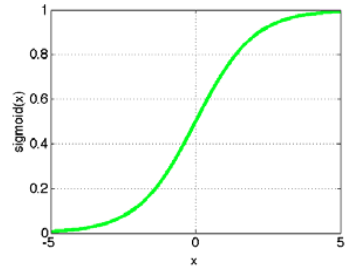


Rectified Linear Unit (ReLU)

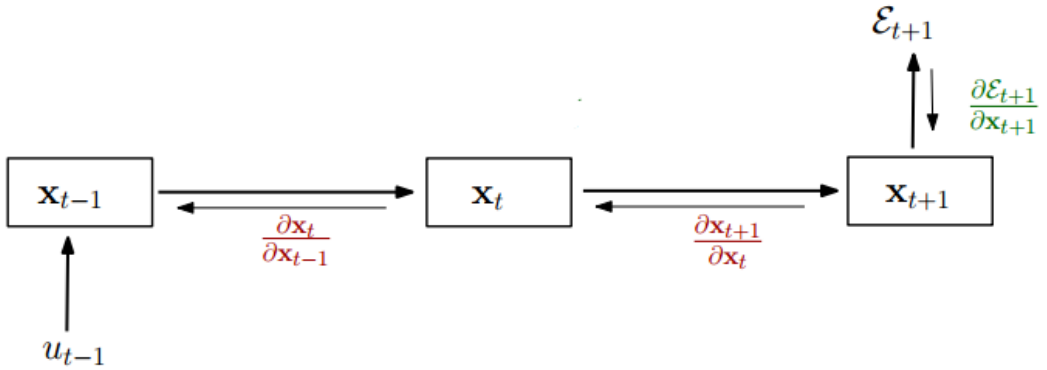
- Rectified linear unit (ReLU)
  - Simplifies backpropagation
  - Makes learning faster
  - Avoids saturation issues







Vanishing Gradient



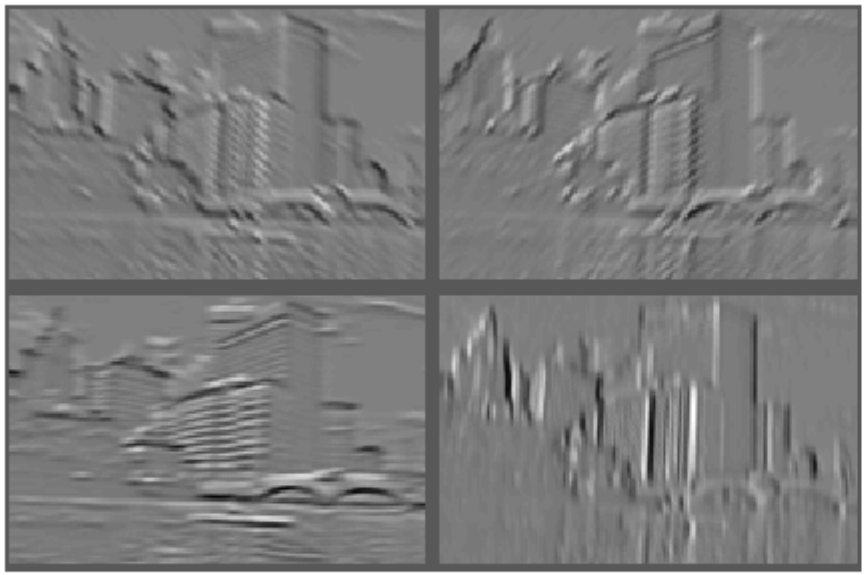
$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right)$$

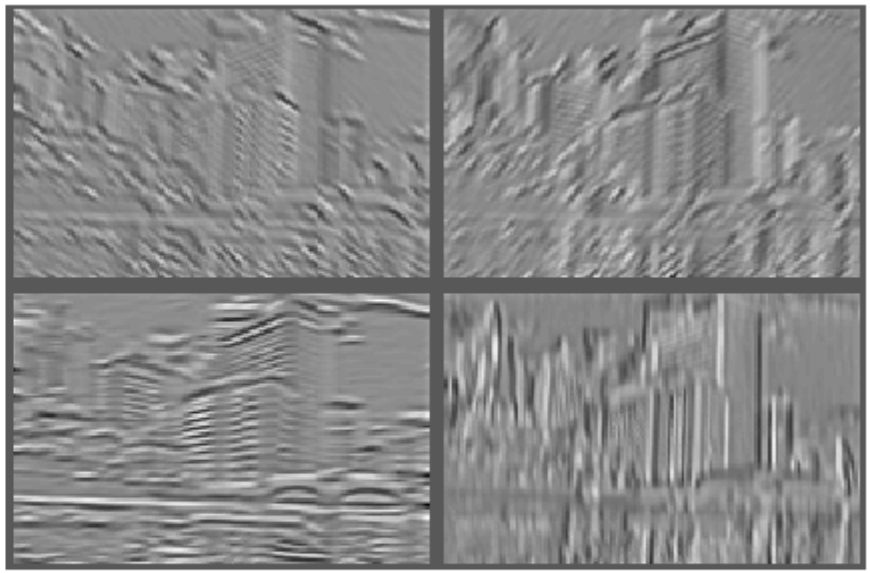
$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

Normalization

- Within or across feature maps
- Before or after spatial pooling

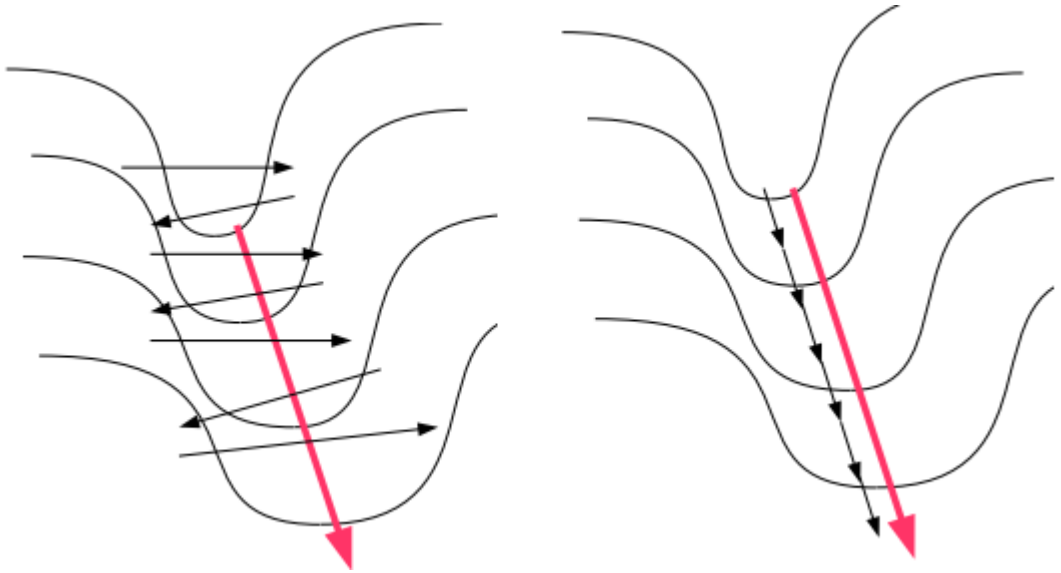


Feature Maps



Feature Maps
After Contrast Normalization

Long Narrow Valley

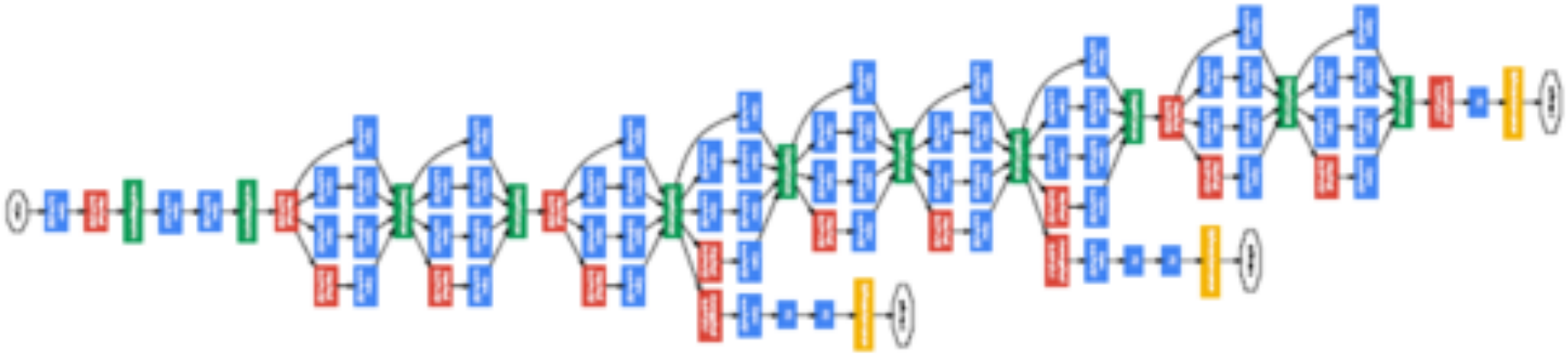


Example of Deep: GoogLeNet (2014)

- 22-layer network trained on 16K CPU cores
- 9 “Interception” modules
- Auxiliary Classifiers
- 12x fewer parameters than AlexNet (60M -> 5M)

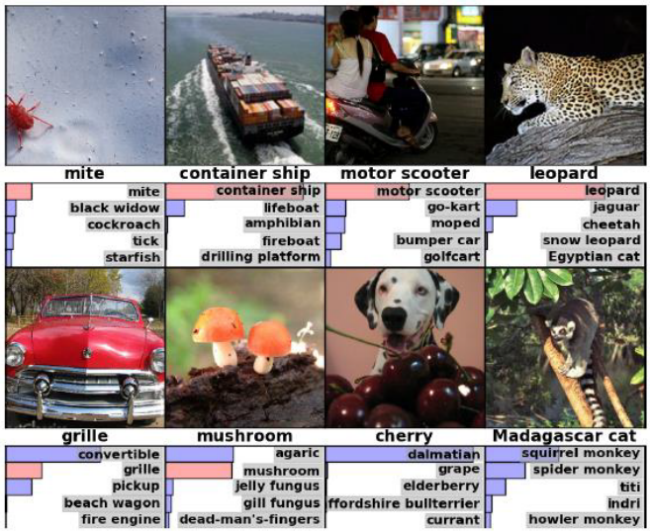
5M parameters?
MLP 32*32*3-1200-1000

60M parameters?
MLP 224*224*3-400-1000



Example: ImageNet Classification Task

- ImageNet Large-Scale Visual Recognition Challenge
  - Image Classification/Localization
  - 1.2M labeled images, 1000 classes
  - CNN has been dominating the contest since..
    - 2012 non-CNN: 26.2% (top-5 error)
    - 2012: (Hinton, AlexNet)15.3%
    - 2013: (Clarifai) 11.2%
    - 2014: (Google, GoogLeNet) 6.7%
    - (pre-2015): (Google) 4.9%
      - Beyond human-level performance



How dare can we learn Deep Neural Net?

- Obstacle
  - Overfitting
  - Vanishing Gradient Problem
  - Non-convex Search Space
  - High Dimensionality
  - Slow Learning Time
- Solution
  - Large Data, Tailored Hardware
  - Small-param Deep, Max Pooling, ReLU, Dropout, Normalization
  - Not unsupervised learning trick until now

6. 어떤 응용에 어떤 딥모델을 사용할 것인가?

고려할 점들

- 감독/무감독 학습
- 변별/생성 모델
- 예측/모듈이해
- 추론가능성
- 연결성
- 깊이
- 배치/온라인 학습

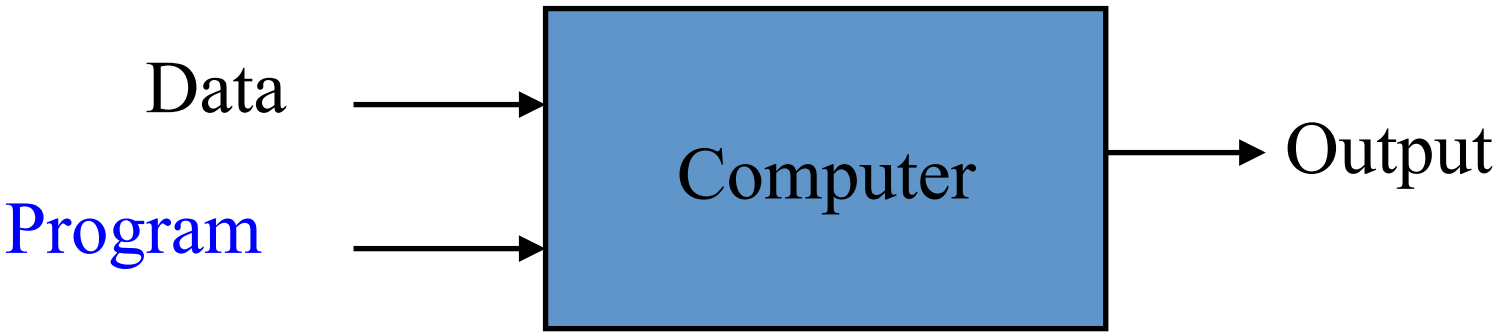
딥모델 적용분야 비교

	Deep Belief Net (DBN)	Convol. NN (CNN)	Deep Hypernet (DHN)
감독/무감독	**감독/무감독**	감독	감독/무감독
변별/생성 모델	생성	**변별**	생성
예측/모듈이해	예측++/**모듈-**	**예측+++**/모듈+	예측+/**모듈+++**
추론가능성	추론++	추론-	**추론++++**
연결성	**Full/Compact**	Partial/Convolved	Partial/**Sparse**
깊이	깊이+++	**깊이++++**	깊이++
배치/온라인 학습	배치	배치	**온라인**

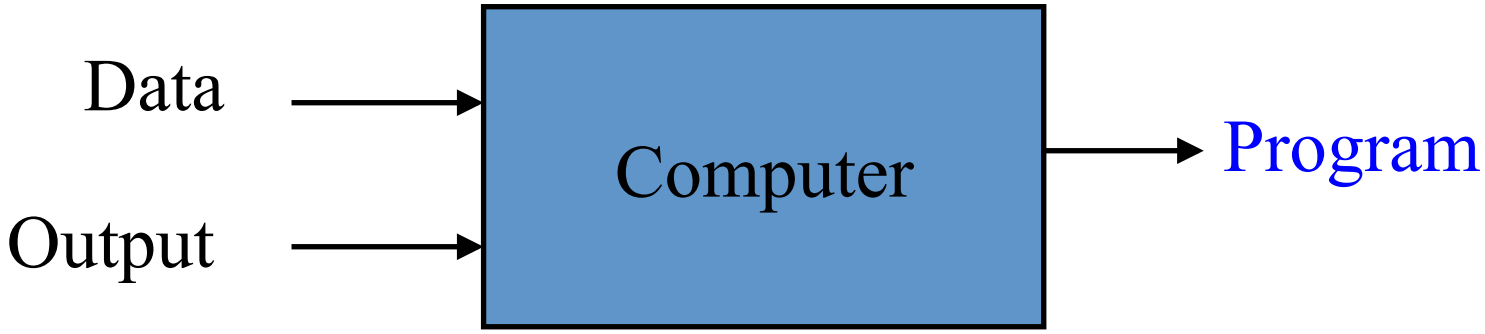
CONCLUSION

자동프로그래밍으로서의 기계학습과 딥러닝

Traditional Programming



Machine Learning



미래: 인간수준지능의 자가학습

Chappie: Self-learning Robot

참고: Resources on Deep Learning

- **Webpages:**
  - Geoffrey E. Hinton's readings (with source code available for DBN) http://www.cs.toronto.edu/~hinton/csc2515/deeprefs.html
  - Notes on Deep Belief Networks http://www.quantumg.net/dbns.php
  - MLSS Tutorial, October 2010, ANU Canberra, Marcus Frean http://videolectures.net/mlss2010au_frean_deepbeliefnets/
  - Deep Learning Tutorials http://deeplearning.net/tutorial/
  - Hinton's Tutorial, http://videolectures.net/mlss09uk_hinton_dbn/
  - Fergus's Tutorial, http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf
  - CUHK MMlab project : http://mmlab.ie.cuhk.edu.hk/project_deep_learning.html
- **People:**
  - Geoffrey Hinton http://www.cs.toronto.edu/~hinton
  - Andrew Ng http://www.cs.stanford.edu/people/ang/index.html
  - Ruslan Salakhutdinov http://www.utstat.toronto.edu/~rsalakhu/
  - Yoshua Bengio www.iro.umontreal.ca/~bengioy
  - Yann LeCun http://yann.lecun.com/
  - Marcus Frean http://ecs.victoria.ac.nz/Main/MarcusFrean
  - Rob Fergus http://cs.nyu.edu/~fergus/pmwiki/pmwiki.php
  - Alex Graves http://www.cs.toronto.edu/~graves/

- **Acknowledgement**
  - **Many materials in this ppt are retrieved from the internet.**
  - **Editorial assistance by Ji-Seob Kim, Sang-Woo Lee, and Kyoung-Min Kim.**