# Deep Learning의 이해

## 2017.3.10
## 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

- 기계 학습 개론
  - 인공지능, 기계 학습, 딥러닝의 개념
  - 기계 학습 및 인공지능의 역사
  - 최신 동향
- 딥러닝 이론
  - 퍼셉트론과 인공신경망
  - Deep Neural Network
  - Convolutional Neural Network
  - Deep Belief Network
  - Recurrent Neural Network
- 딥러닝 실습

# 인공지능 및 기계 학습 개론

## Getting Started:
# How to Build a Diagnosis System for Diabetes

if hungry_blood_sugar > 125mg/dl
then diabetes = yes
else diabetes = no

## Getting Started:

# How to Build a Prediction System for Pima Indians Diabetes

From Wikipedia.org♪

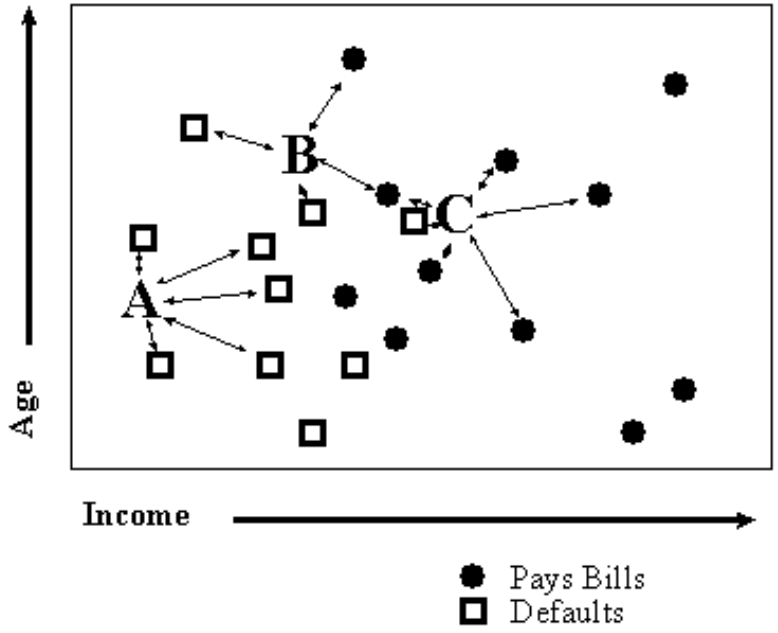
- UCI Machine Learning Repository [Link]
- Pima Indians Diabetes Data Set [Link]

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1) (Onset in 5 years)

**Getting Started:**

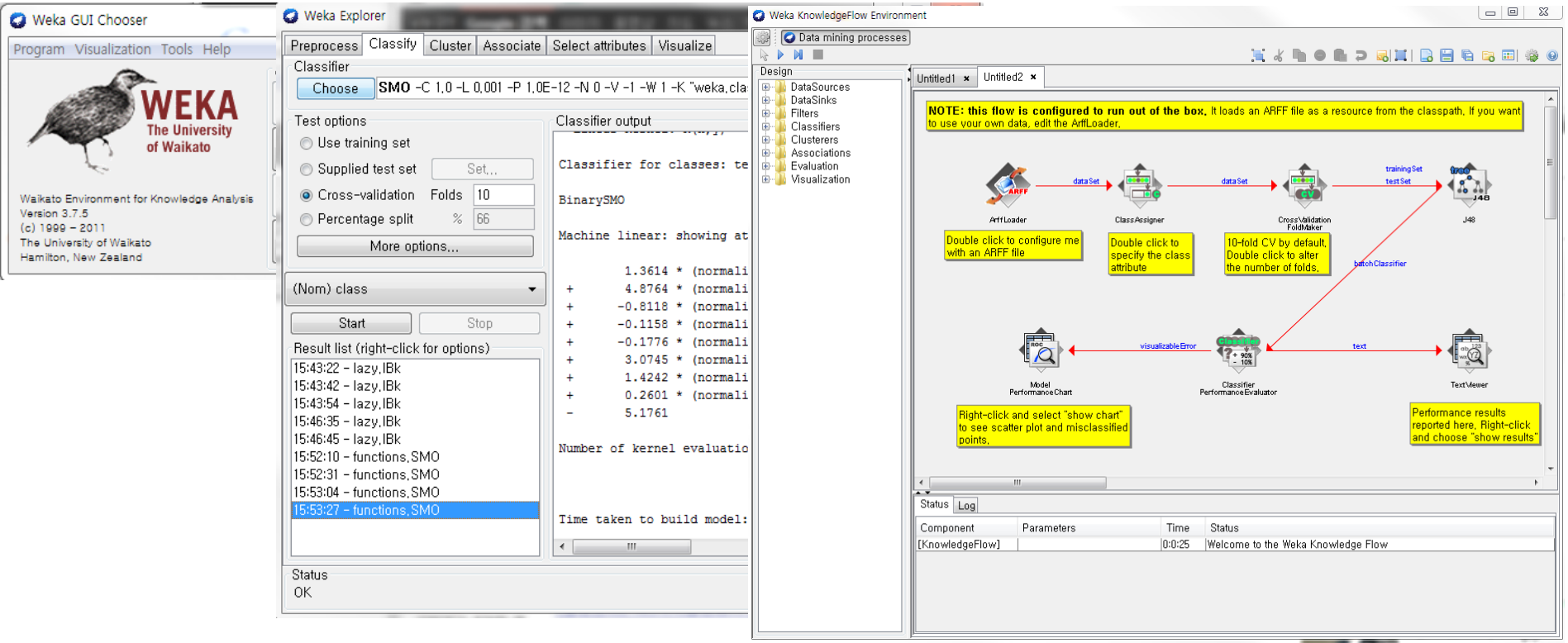
# k-Nearest Neighbor

- Classify based on the closest examples
  - Euclidean distance
- Instance-based learning (Lazy learning)
  - Remember all examples
- Properties
  - Fast learning
  - Slow running
  - Large storage
  - Good performance



# Getting Started:
# Machine Learning with WEKA

- WEKA: Free collection of machine learning algorithms for data mining [Link]
  - Explorer
  - Knowledge Flow



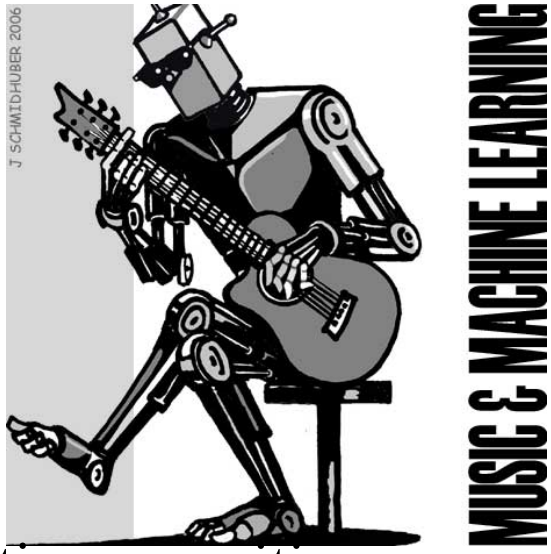
# Overview

- What is the **Machine Learning**?
- Experiment Scheme

# Machine Learning (ML)

- Definition: a branch of artificial intelligence, a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases

- Brief History
  - AI?
  - Data Mining?



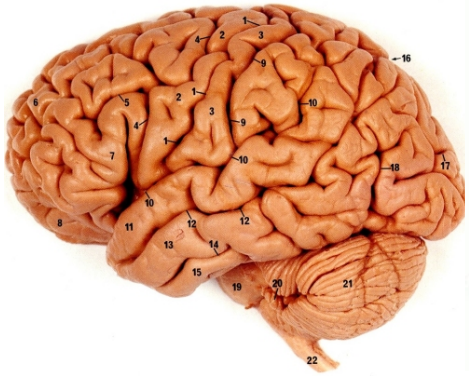

- Related Areas
  - probability theory, statistics, pattern recognition, cognitive science, data mining, adaptive control, computational neur oscience, theoretical computer science, ...

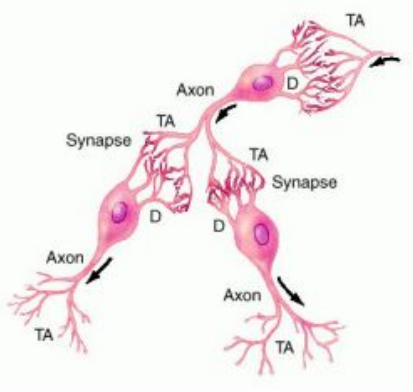
# Machine Learning: Applications

- Machine perception
- Computer vision
- Natural language processing
- Syntactic pattern recognition
- Search engines
- Medical diagnosis
- Bioinformatics
- Structural health monitoring
- Speech and handwriting recognition
- Object recognition in computer vision
- Brain-machine interfaces and cheminformatics
- ...

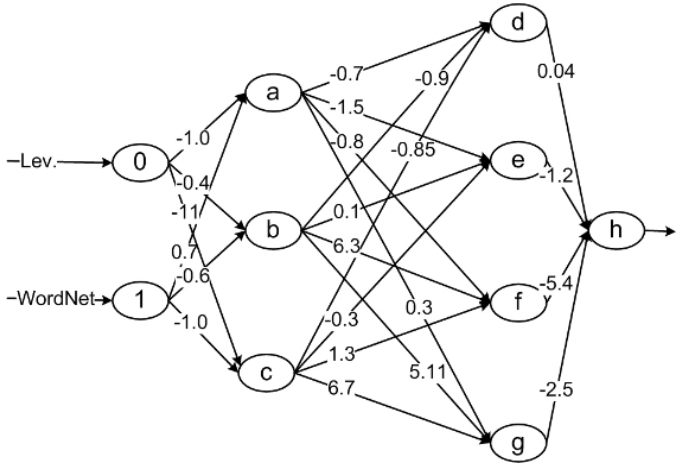
- Detecting credit card fraud
- Stock market analysis
- Classifying DNA sequences
- Game playing
- Software engineering
- Adaptive websites
- Robot locomotion

# What is "Machine"?






- Computer




- A system, not a (human) brain

- **Model**
  - Mathematical abstraction of a system
    - Equation
    - Function
  - Element, structure, parameters, state, dynamics, ...
  - Deterministic / Probabilistic

# What is "Learning"?

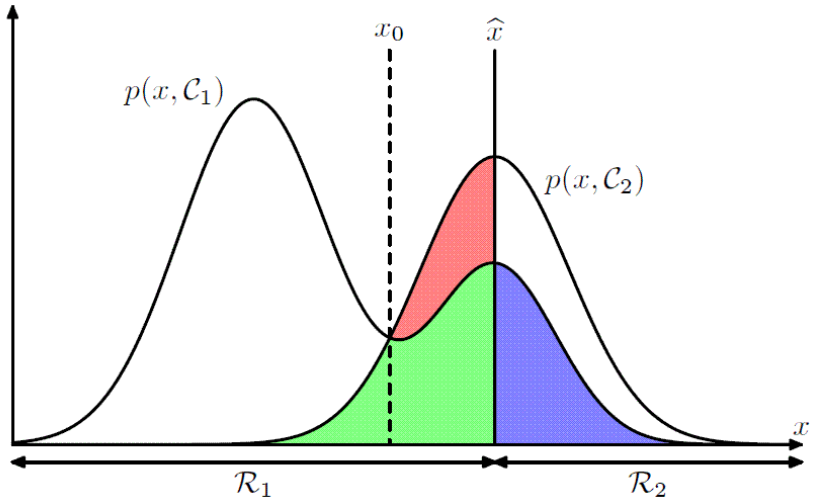
- E: experience
- T: tasks
- P: performance measure
- Learning = **improving P at T with E**

- Training?

Learning = improving P at T with E

# T (Task) = Something Done (Output)

- Examples
  - Diagnosing
    - Diabetes
    - Cancer
    - ...
  - Driving
    - Car
    - Helicopter
    - ...
  - ...
  - Forecasting
    - Weather
    - Stock price
    - ...
  - Recommending
    - Product
    - ...
- Decision theory



# E (Experience) = (Training) Data

- Supervised learning
  - Learn **w/** answers (class)
  - Classification
    - Yes/No, A/B/C/D, ...
  - Regression
    - Real number
- Unsupervised learning
  - Learn **w/o** answers (no class)
  - Clustering
  - Feature selection / Dimension reduction
- Semi-supervised learning
  - Web image auto-tagging
- Reinforcement learning
  - Chess

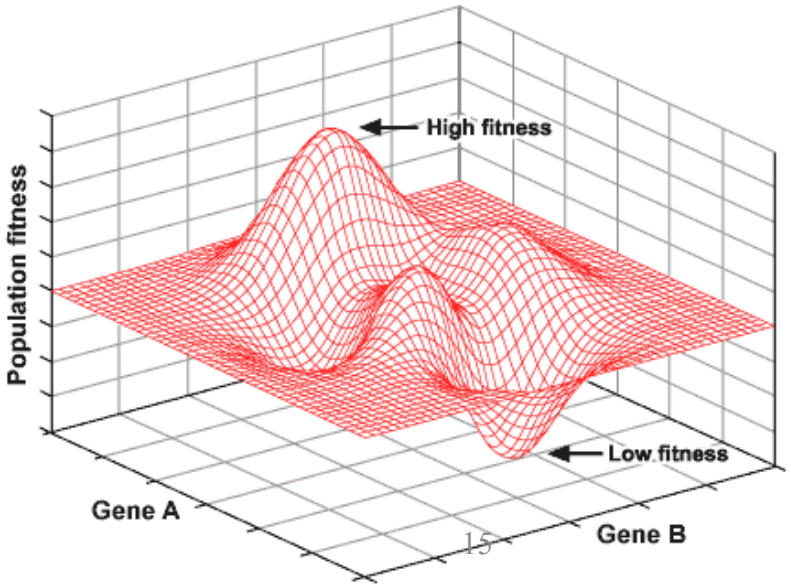
| No. | sepallength Numeric | sepalwidth Numeric | petallength Numeric | petalwidth Numeric | **class** Nominal |
|---|---|---|---|---|---|
| 48 | 4.6 | 3.2 | 1.4 | 0.2 | Iris-setosa |
| 49 | 5.3 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 50 | 5.0 | 3.3 | 1.4 | 0.2 | Iris-setosa |
| 51 | 7.0 | 3.2 | 4.7 | 1.4 | Iris-versicolor |
| 52 | 6.4 | 3.2 | 4.5 | 1.5 | Iris-versicolor |
| 53 | 6.9 | 3.1 | 4.9 | 1.5 | Iris-versicolor |
| 54 | 5.5 | 2.3 | 4.0 | 1.3 | Iris-versicolor |
| 55 | 6.5 | 2.8 | 4.6 | 1.5 | Iris-versicolor |

# P (Performance Measure) = Target (or Loss) Function

*"If you cannot measure it, you can not improve it."*

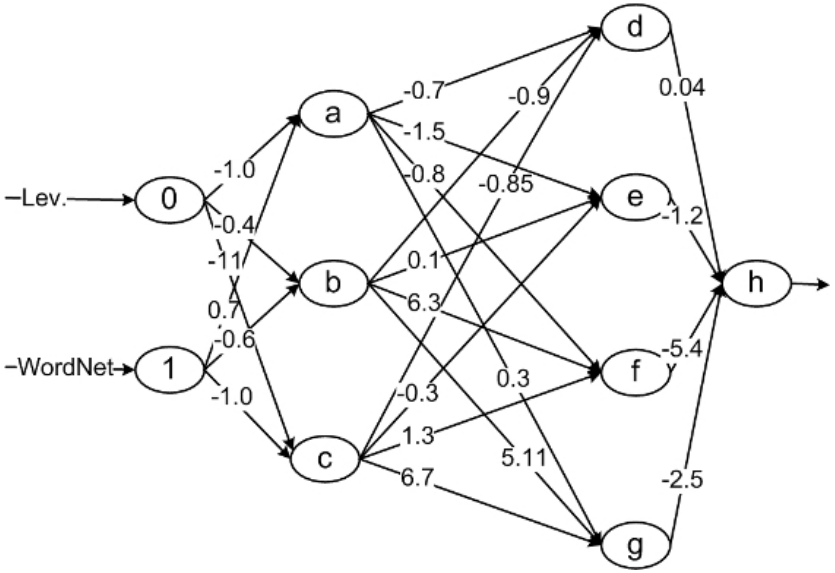
**Kelvin, Lord William Thomson (1824-1907)**

- Error rate?
- Euclidean distance
  - e.g. $\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$ where $e_j(n) = d_j(n) - y_j(n)$
- Log Probability
  - e.g. $L(\mathrm{w}) = -\Sigma \log \mathrm{P}(\mathrm{y}=\mathrm{d}^{(i)}|\mathrm{X}^{(i)})$
- Information theoretical measures
  - Mutual information
  - KL
- ...



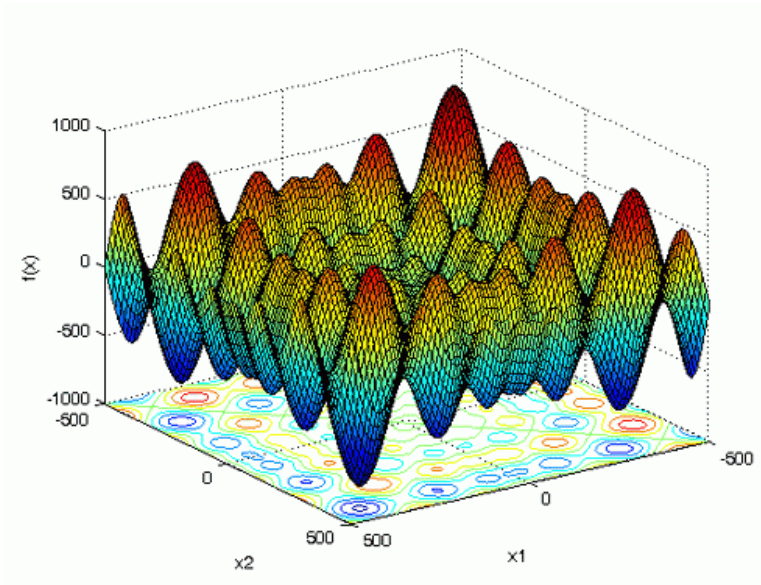
**Learning** = improving P at T with E

# How to "Learn"?



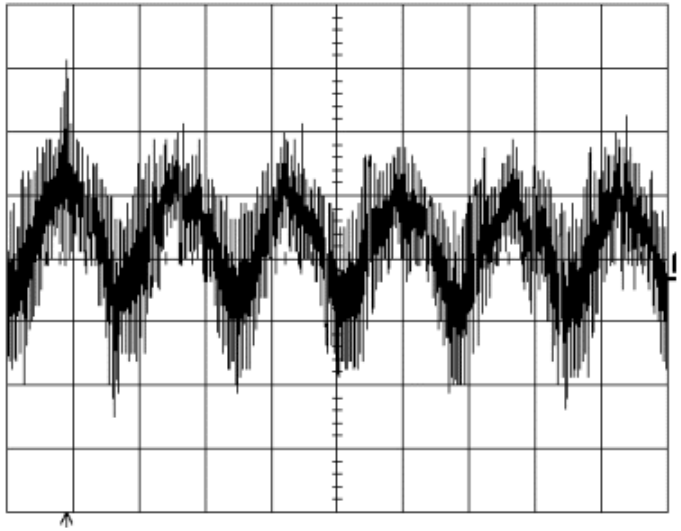

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

- Adjust (parameters of) model
- Minimize loss function
  ⇒ **Learning algorithm**
- Obstacles
  - Limited data ⇒ **Overfitting**
    - #
    - Noise
  - Limited computational power
    - **Local optima**
    - High dimension
  - ...



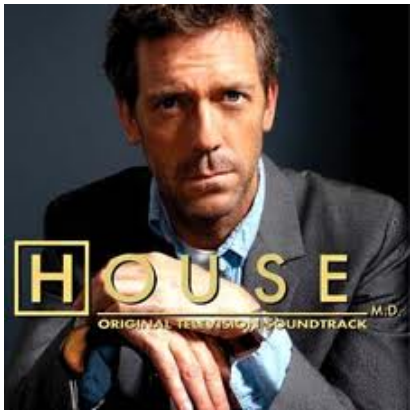
# Why Do Errors Happen?

- Noise
  - Human error
  - Resolution
  - Uncertainty
- Unobserved
  - Feature
  - Case



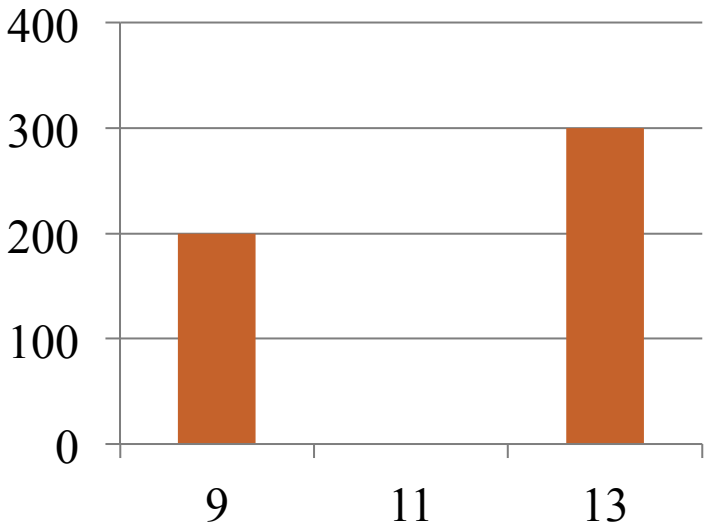
# Unobserved Cases

- Training
- Diagnosis
- Prediction
  - Black swan
- **We can’t learn all cases!!!**




# Model & Inductive Bias

- Answering to an unobserved case



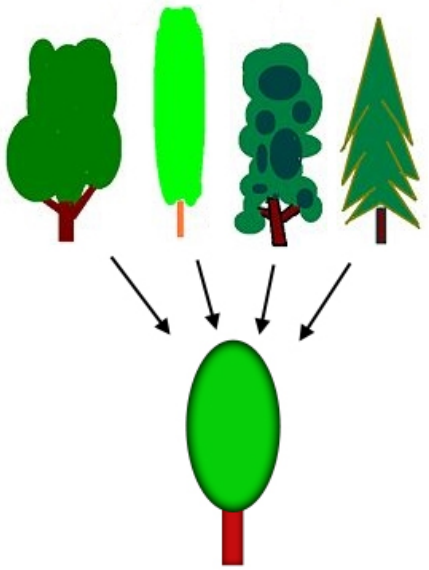
# Error Estimation

- A fortune-teller
  - Correct for past & present
  - Wrong for future!!
- Error for Observed Data
  = Error for Unobserved Data?
- Training error = Test error?
  - Training set / Test set
  - Cross-validation

# Generalization

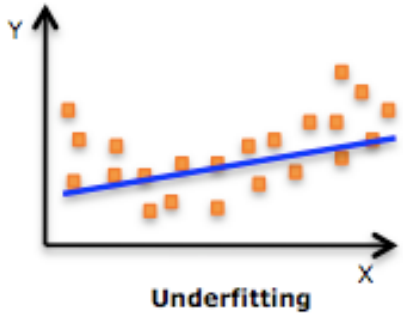
- Study & Exam
  - Training/Test error
- Understanding
  - Concept, Abstraction
  - Build good model for "True Nature"

  → **Model Selection**

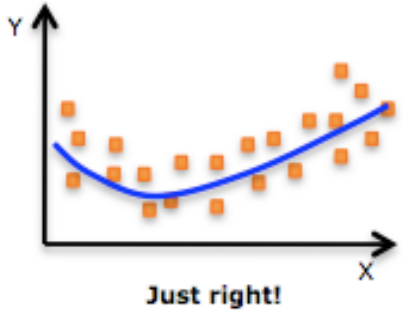


# Overfitting & Model Selection

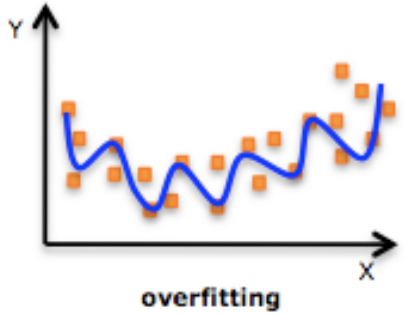
$$y = a_2x^2 + a_1x + a_0 + Noise$$




$$y = a_1x + a_0$$

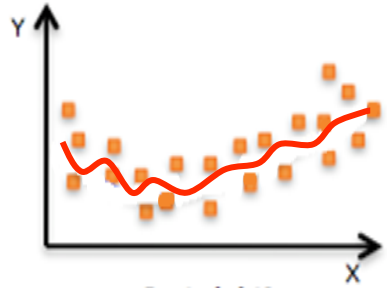


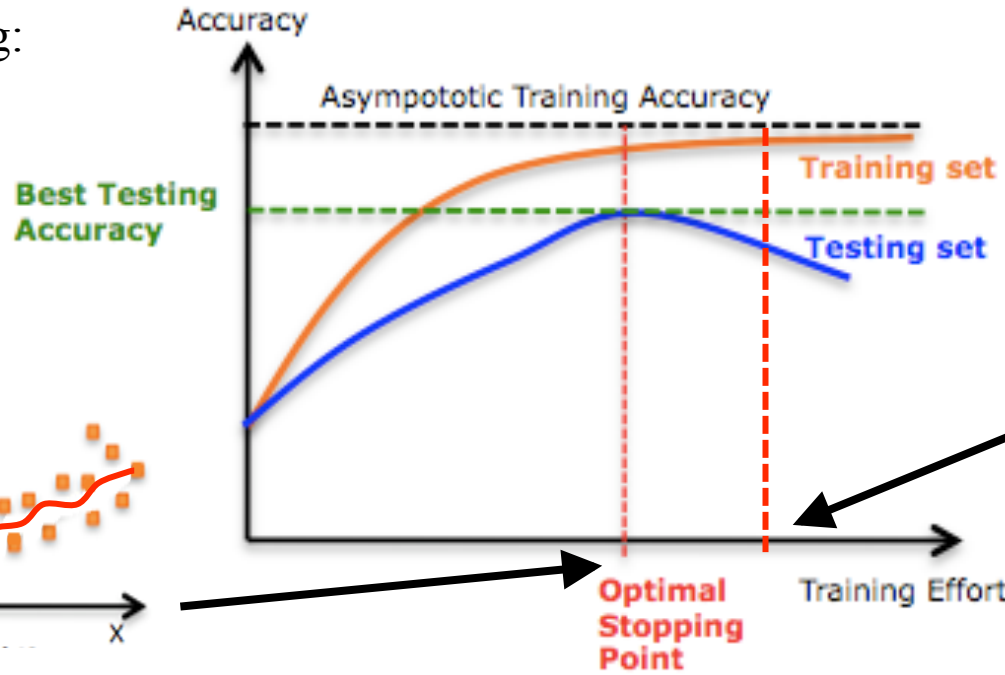

$$y = a_2x^2 + a_1x + a_0$$

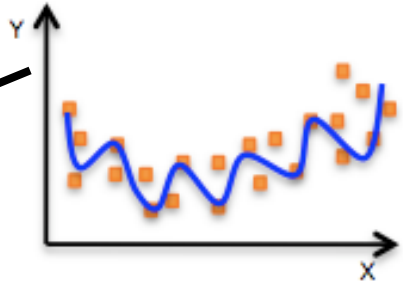



$$y = a_{10}x^{10} + \ldots + a_1x + a_0$$

Early Stopping:



# Occam's Razor & Model Selection

- Simpler explanations are, other things being equal, generally better than more complex ones.
  - http://homa.egloos.com/page/6
- ↔ Hickam's dictum
- Model of "True Nature"?
  - Shortage of data

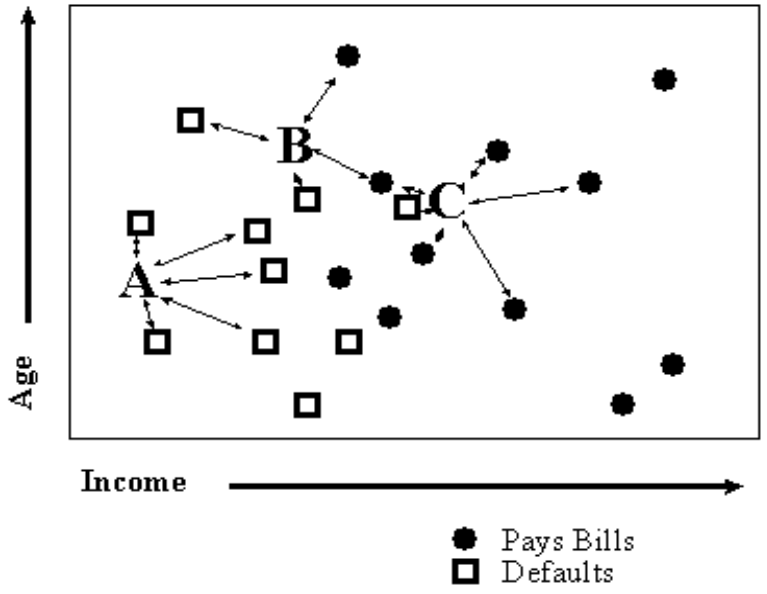
    → Simple model

    → Reasonable test error
  - More data or Regularization

# Experiment Scheme

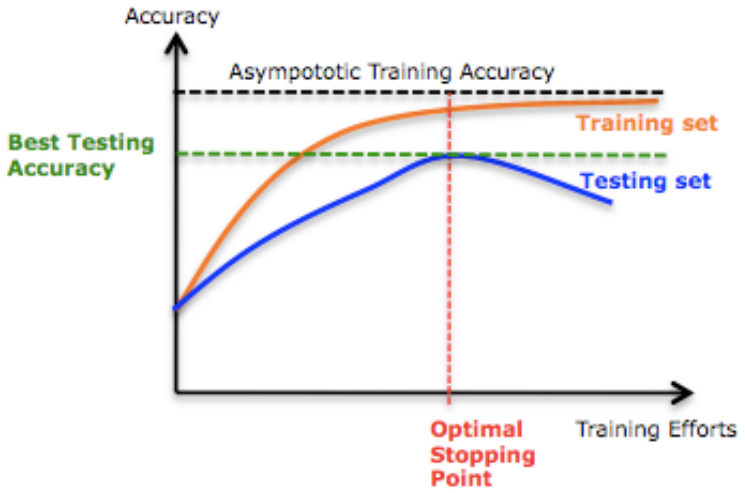
# Experiment Scheme in Supervised Learning

- Get Data
- Divide data set
  - Training / (Validation) / Test set
- Preprocess data (w/ training set)
  - Standardizing (Normalizing)
  - Feature Selection
  - Discretization
  - ...
- Choose model & algorithm
  - Set options
  - Initialize
- Run
- Record training & test error
- Repeat
- Analyze



# Analysis

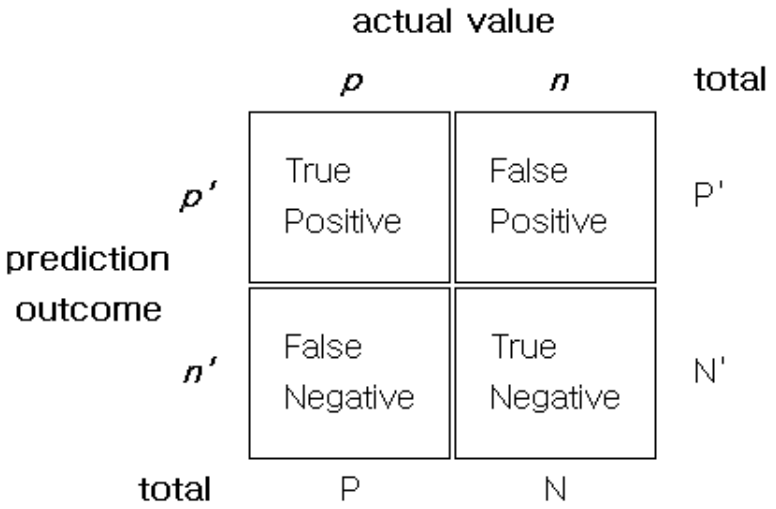
- Accuracy
  - Classification rate (Training & Test)
- MSE, RMSE(RMSD)
  - $$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}.$$
- Learning Curve



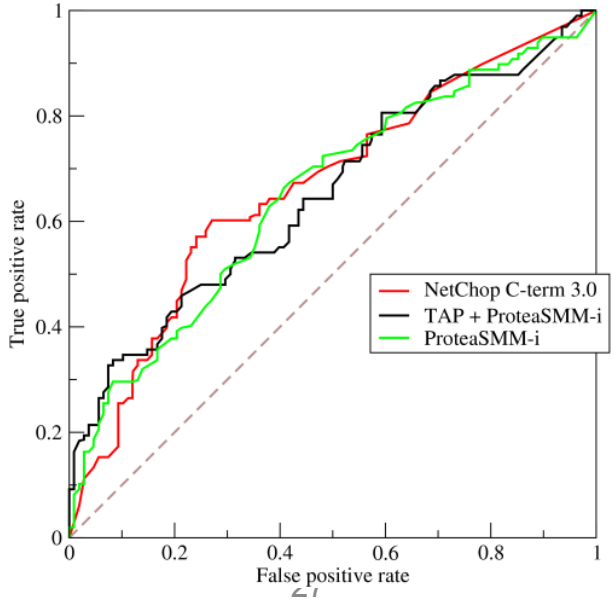

- ROC curve & AUC
- Cross-validation

# ROC Curve

- Radar
  - Bird or Bomber?
  - Trade-off
  - Receiver operating characteristic
- Confusion Matrix

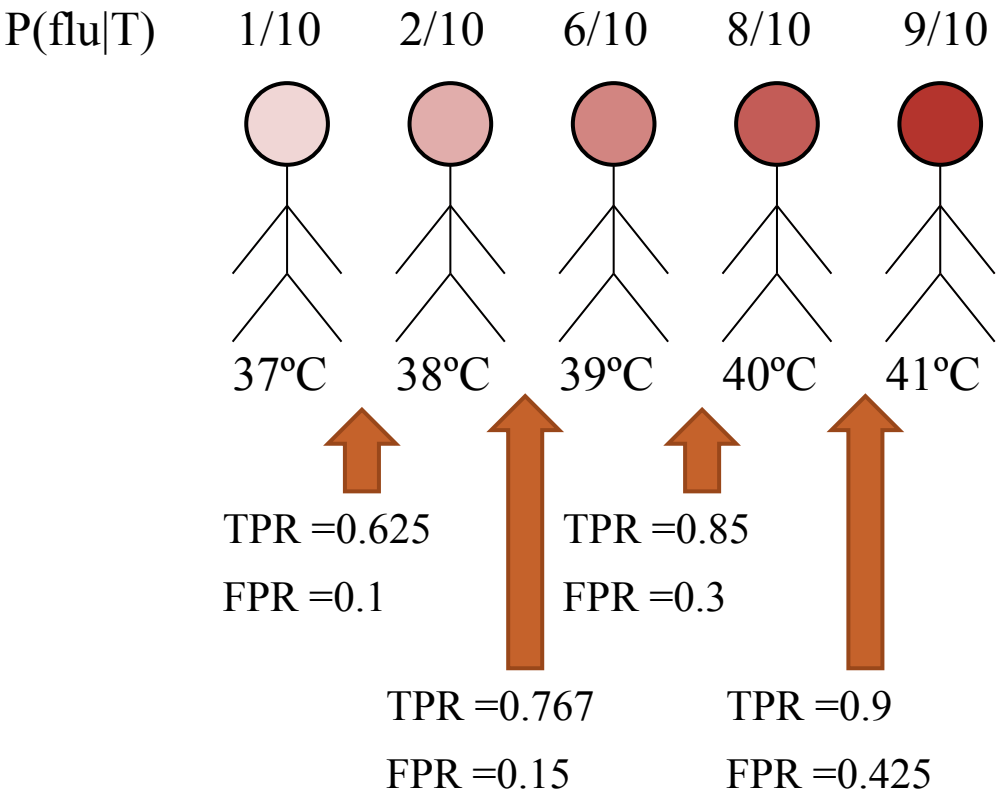


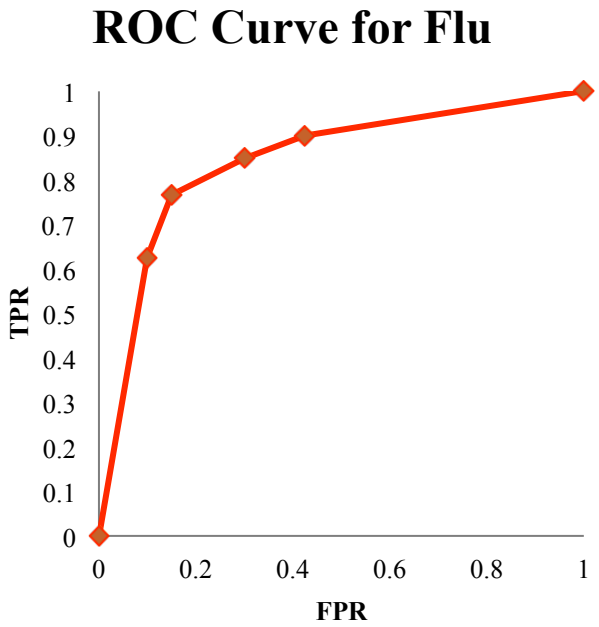

- Sensitivity
  = TP rate = TP/P
- 1-Specificity = 1 – TN/N
  = FP rate = FP/N
- AUC (Area under Curve)



# ROC Curve

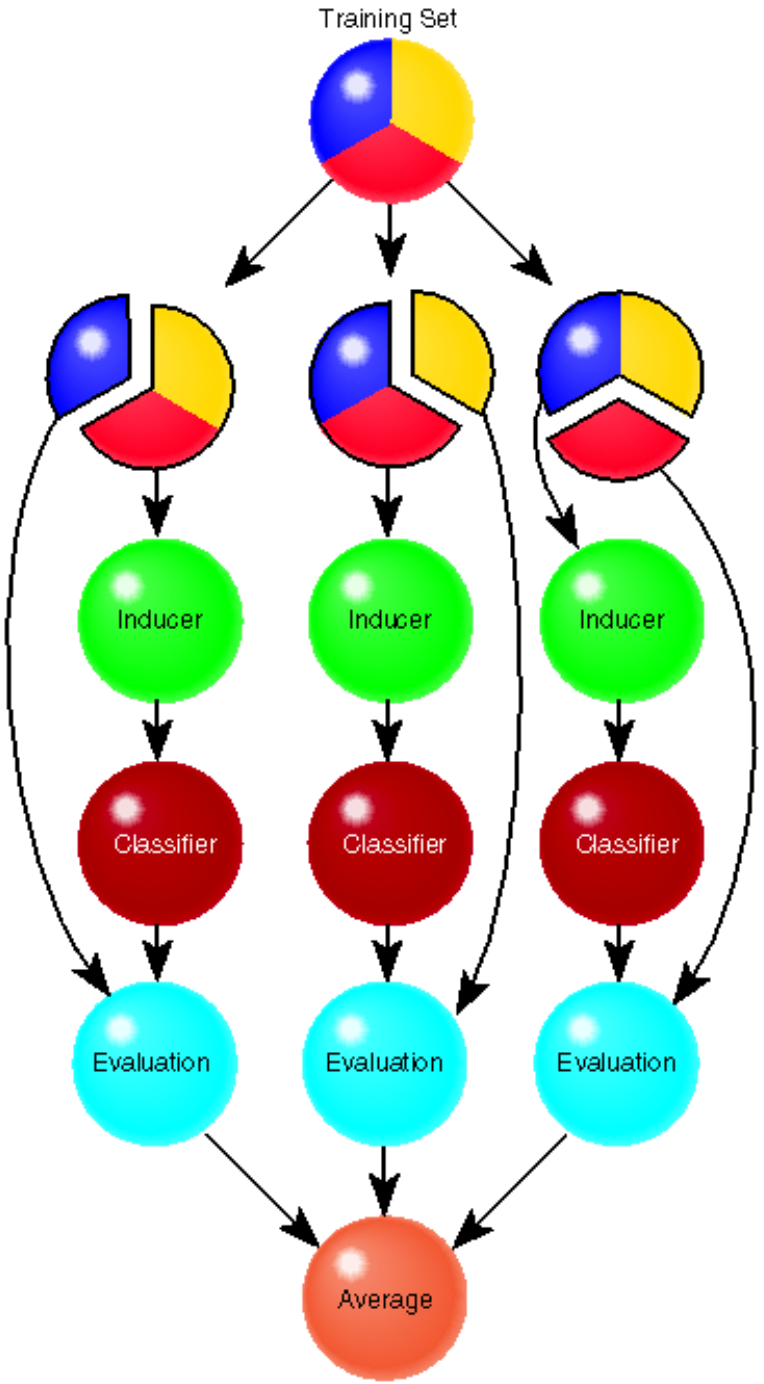
- Ex) Flu diagnosis with body temperature



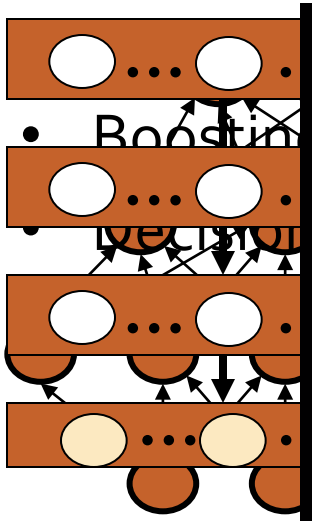
# Cross-Validation

- Ex) 3-fold CV



# 기계 학습 동향

# History of Neural Network Research



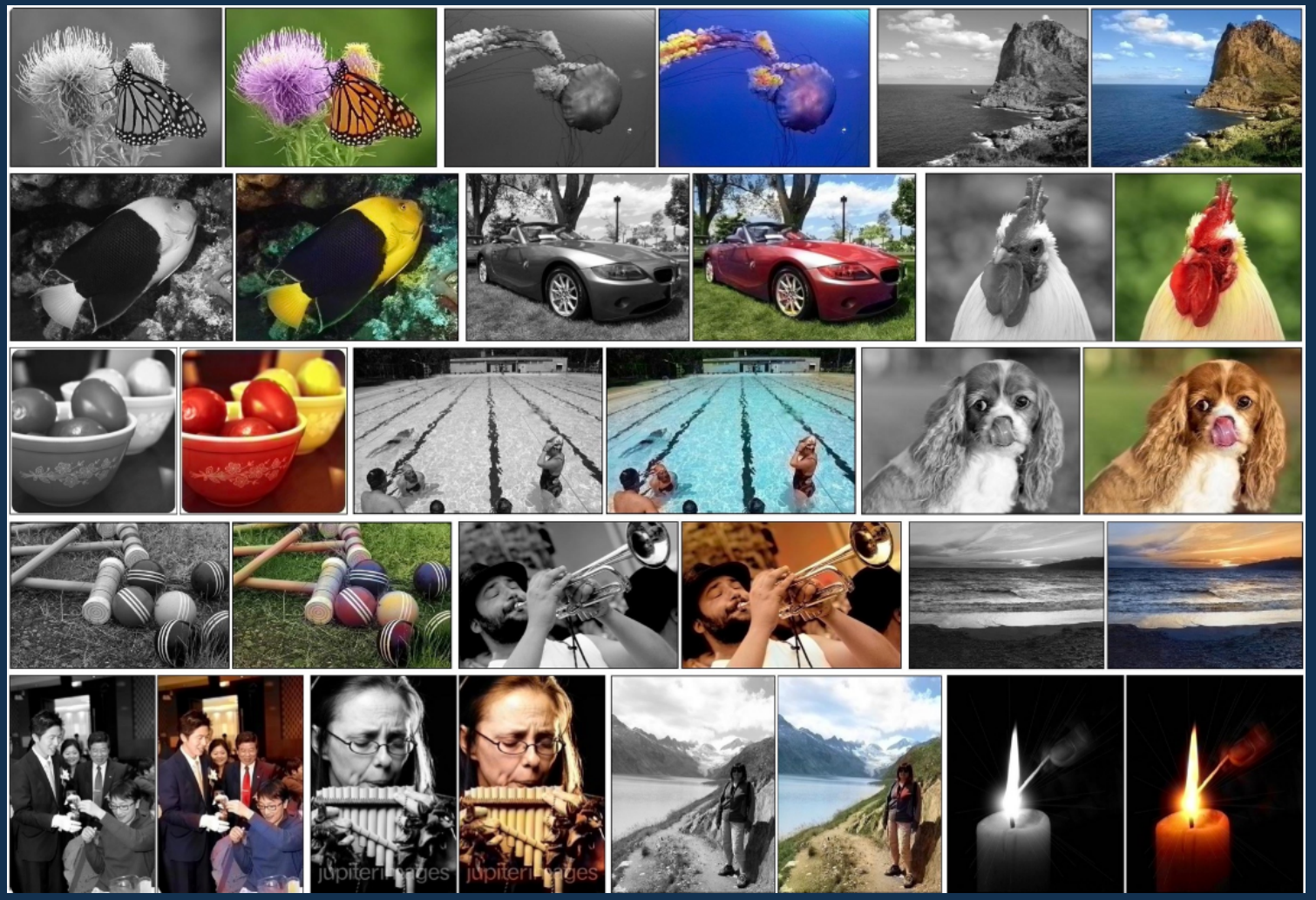
| Rank | Name | Error rate | Description |
|---|---|---|---|
| 1 | **U. Toronto** | 0.15315 | Deep Conv Net |
| 2 | U. Tokyo | 0.26172 | Hand-crafted features and learning models. Bottleneck. |
| 3 | U. Oxford | 0.26979 | |
| 4 | Xerox/INRIA | 0.27058 | |

## Deep Networks Advance State of Art in Speech

Deep Learning leads to breakthrough in speech recognition at MSR.

# What deep learning can do?

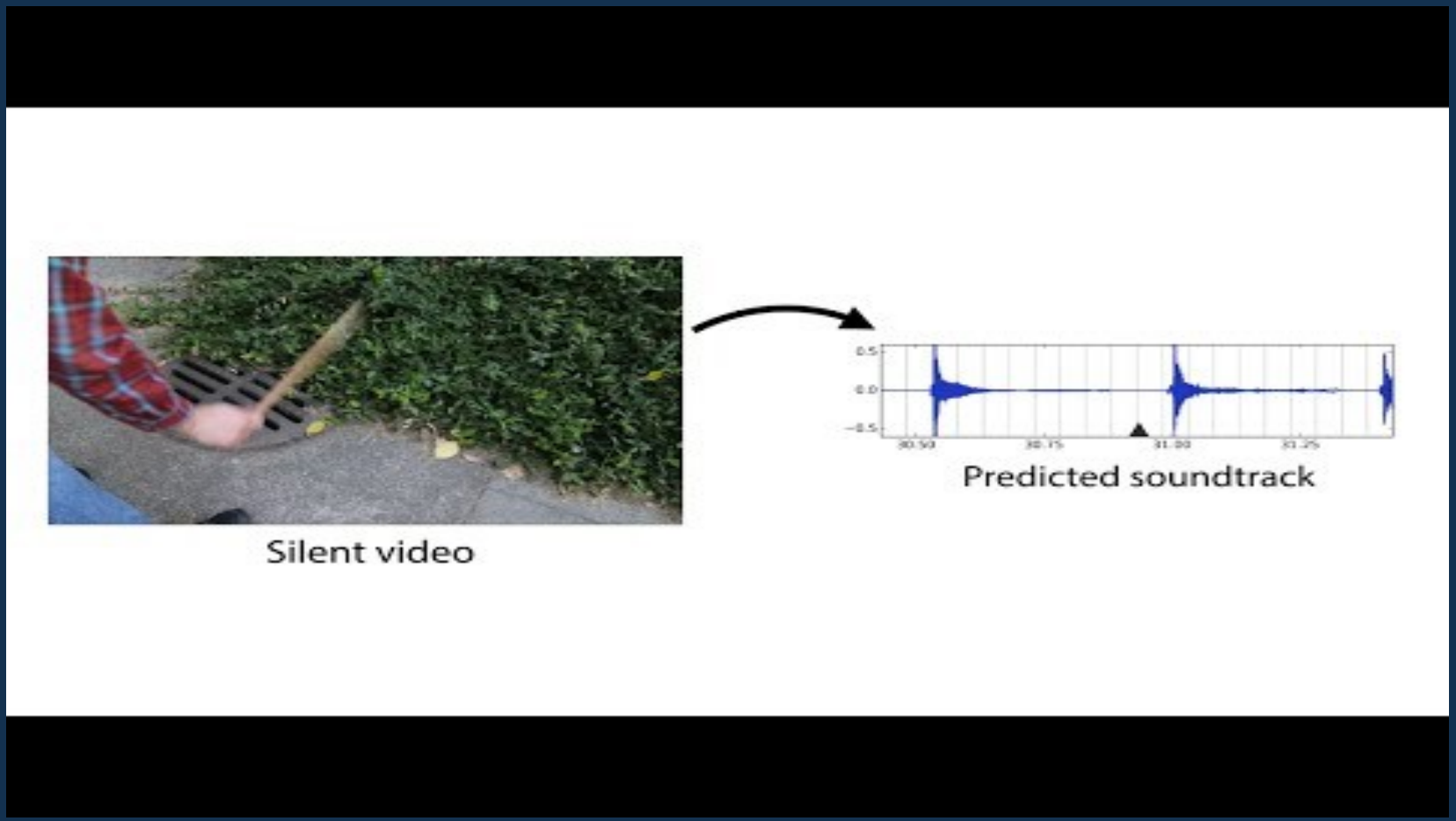
## 1. Automatic Colorization of Black and White images



http://richzhang.github.io/colorization/

# What deep learning can do?

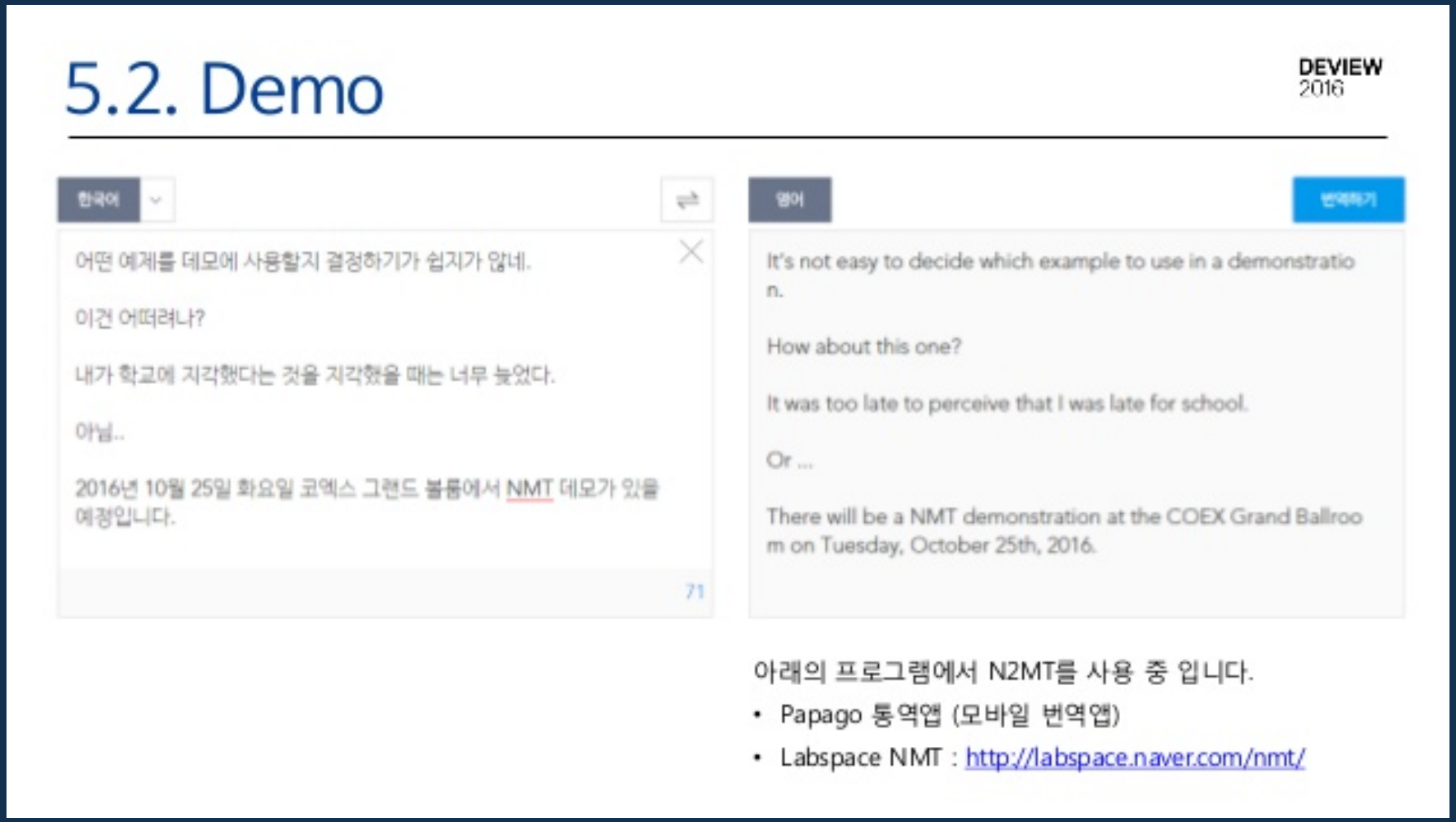
## 2. Automatically adding sounds to silent movies




http://vis.csail.mit.edu/

# What deep learning can do?

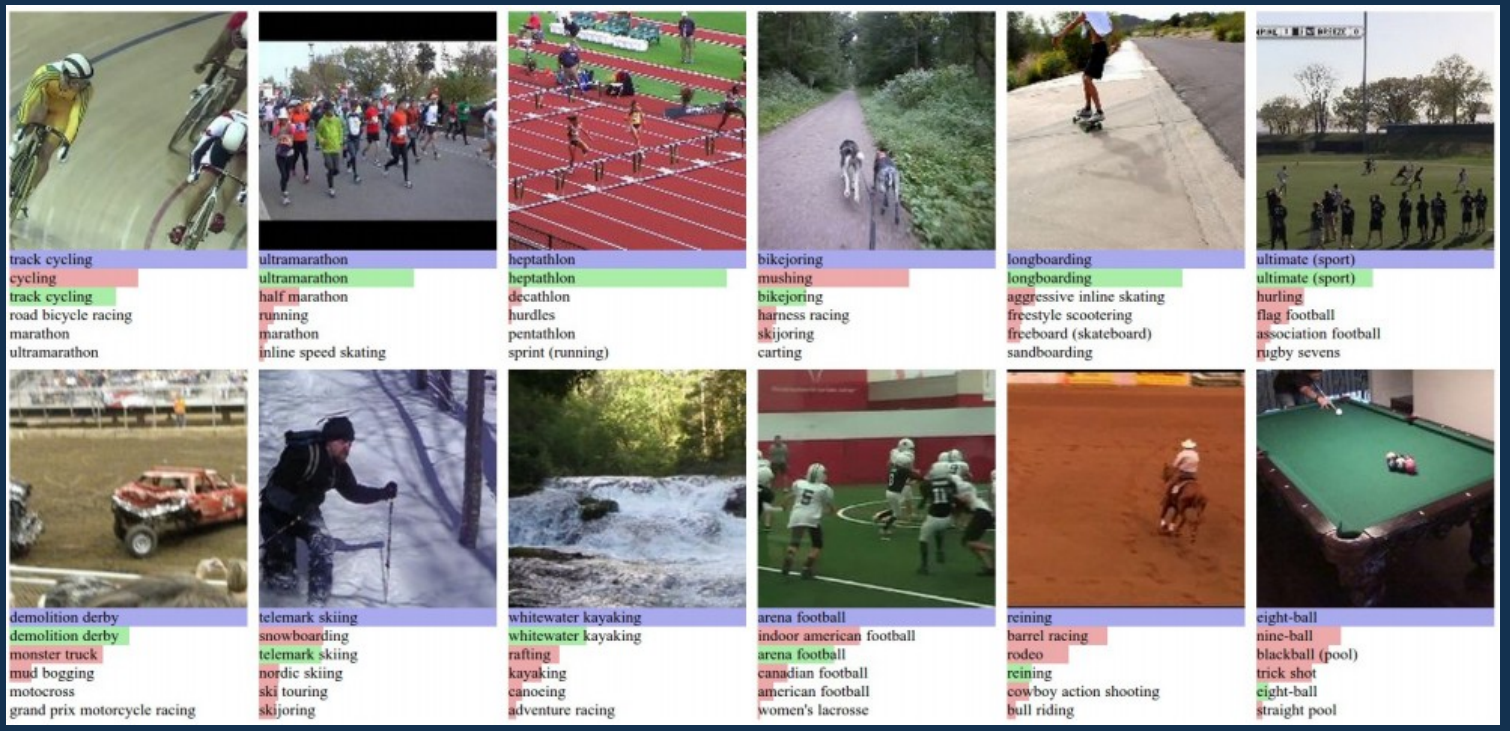
## 3. Automatic Machine Translation

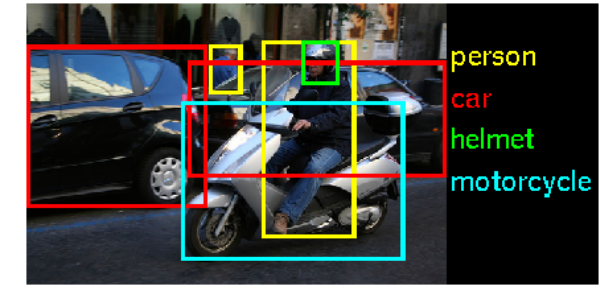



http://www.slideshare.net/deview/224-backend-neural-machine-translation-67608580

# What deep learning can do?

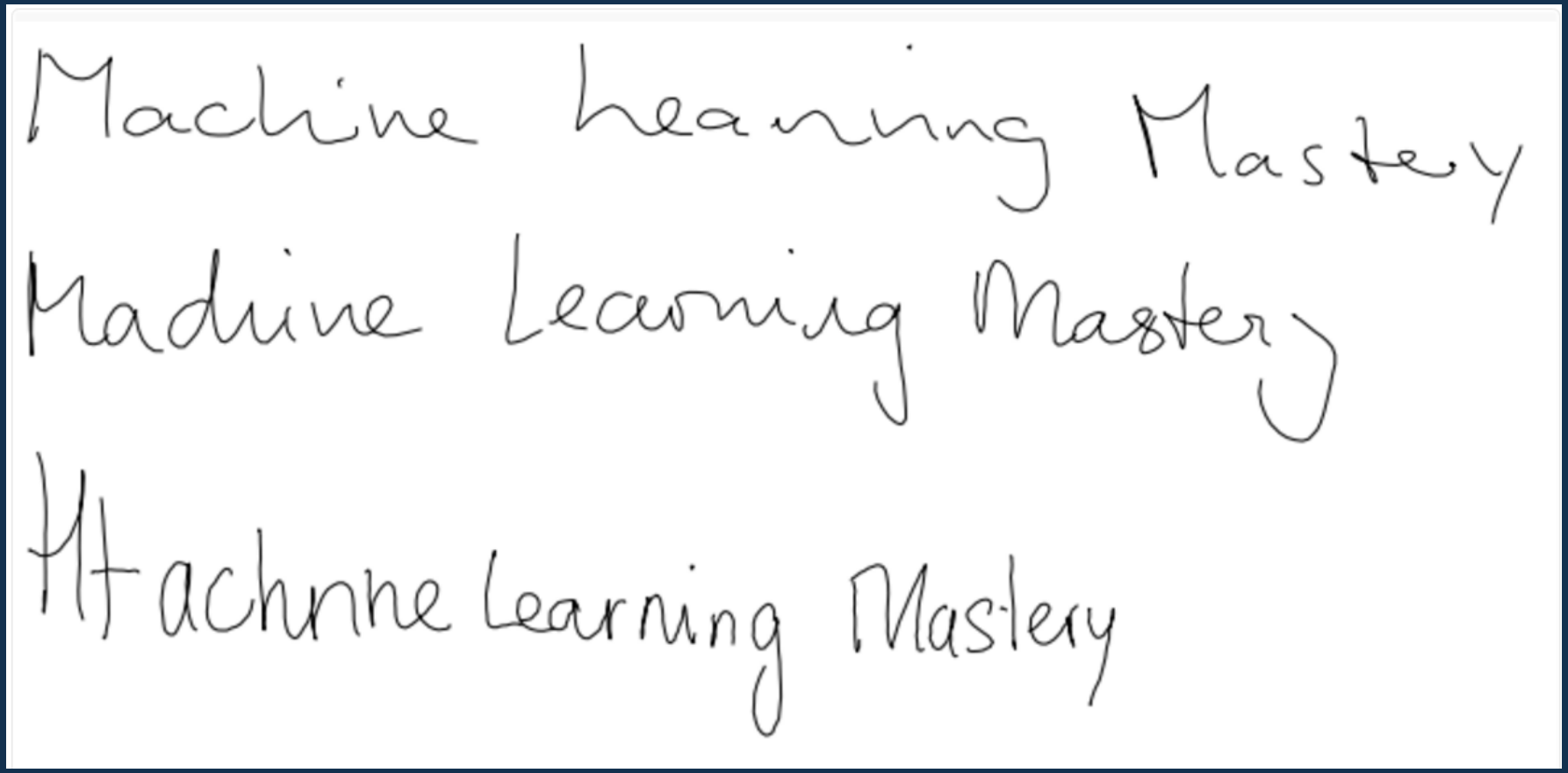
## 4. Object Classification and Detection in Photographs




ImageNet Classification with Deep Convolutional

## 5. Automatic Handwriting Generation




http://www.cs.toronto.edu/~graves/handwriting.html

# What deep learning can do?

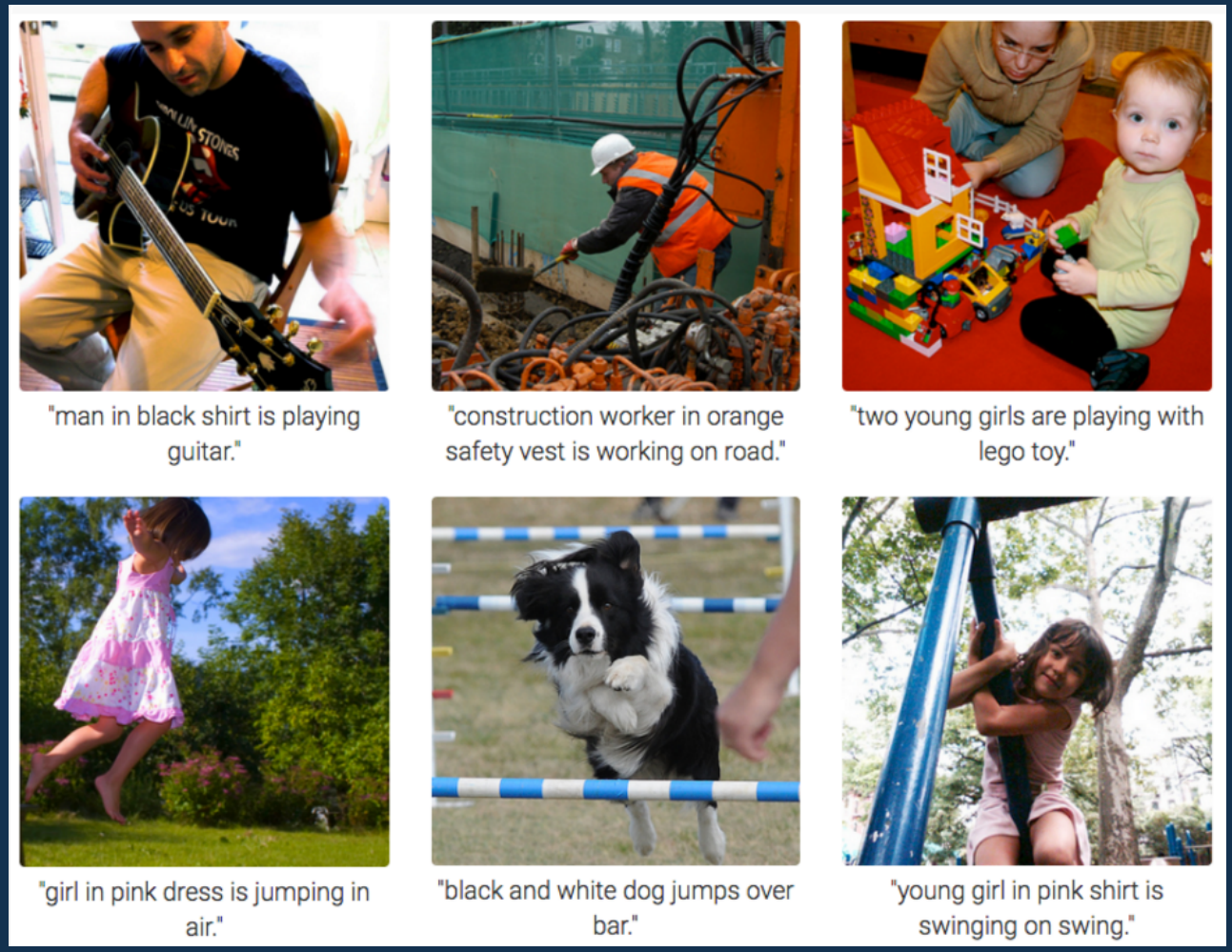
## 6. Automatic Text Generation

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# What deep learning can do?

## 7. Automatic Image Caption Generation




http://cs.stanford.edu/people/karpathy/deepimagesent/

## 8. Automatic Game Playing


https://youtu.be/TmPfTpjtdgg

# 딥러닝 이론

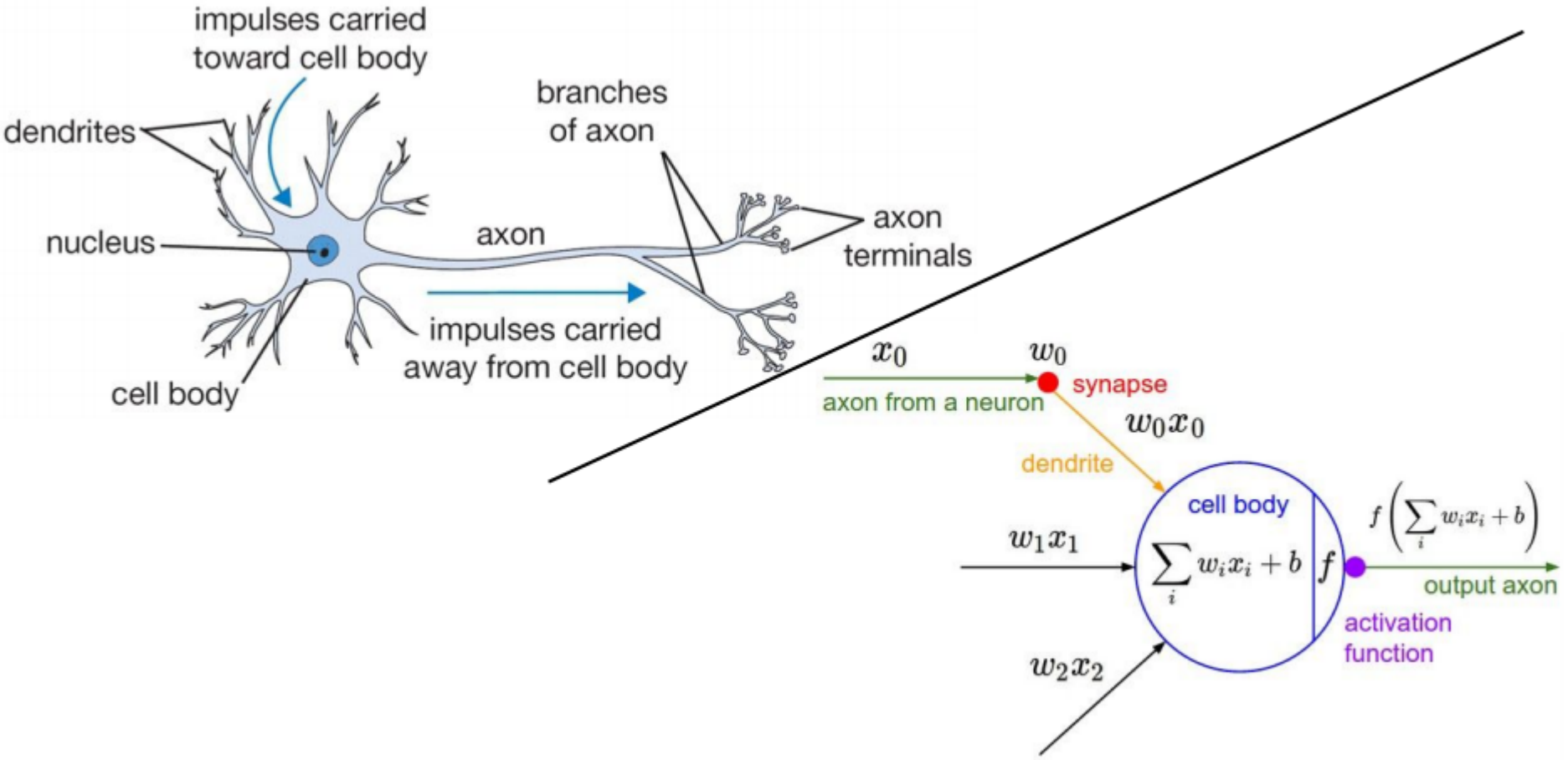
# Classification Problem

- 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)

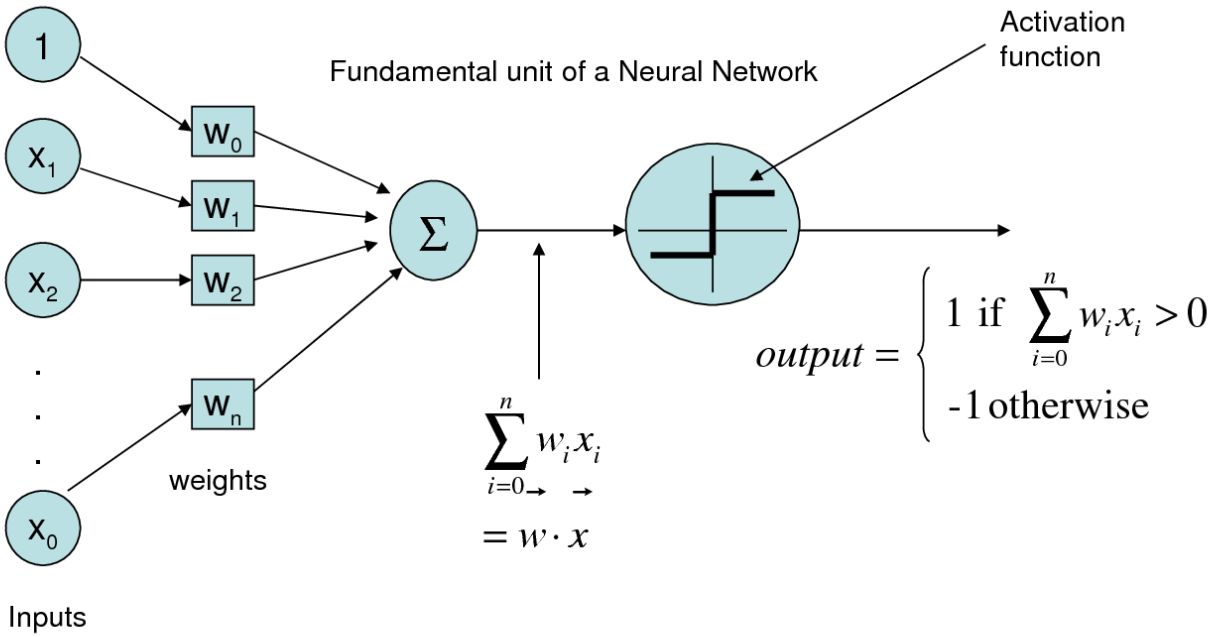



http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

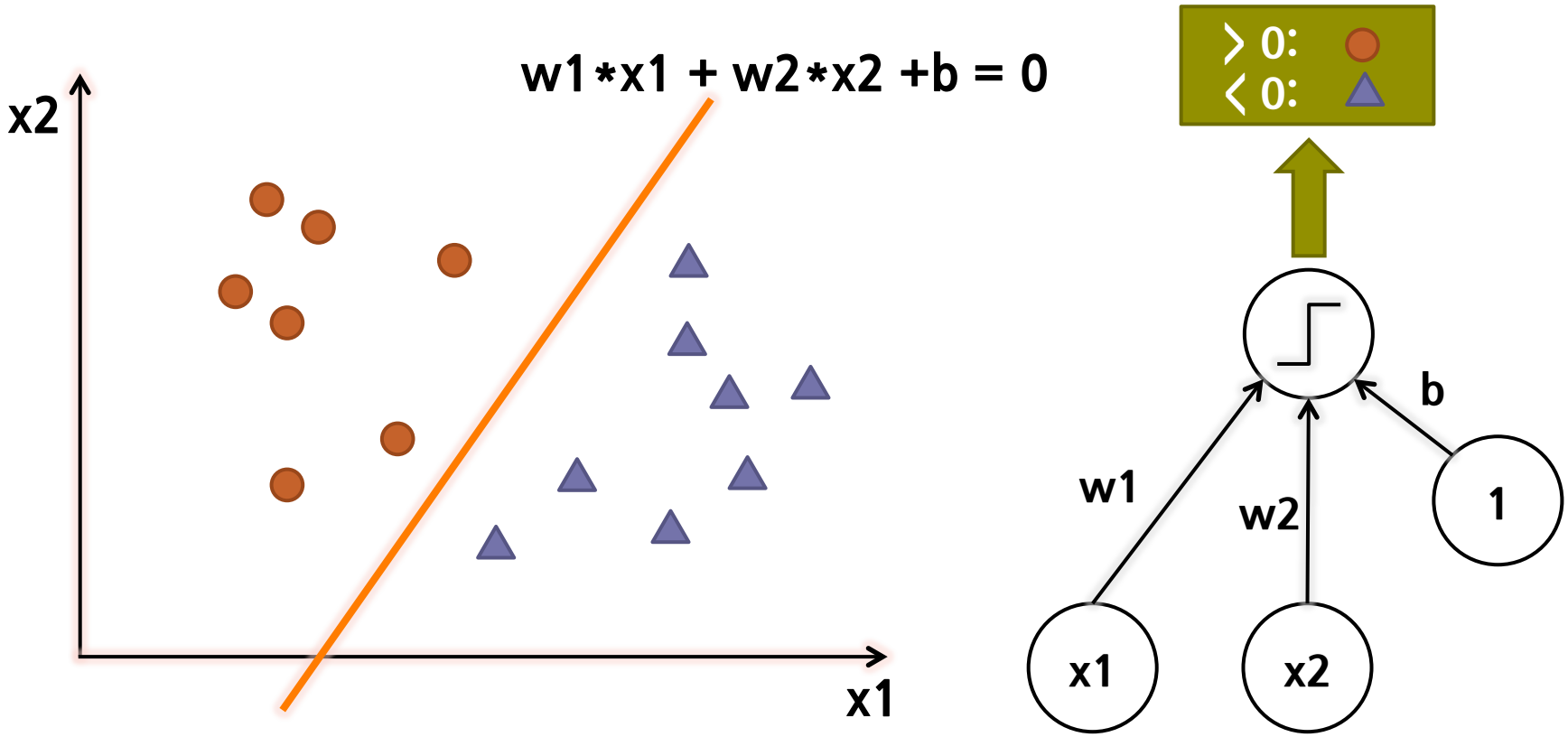
# Perceptron (2/3)

***Artificial Neural Networks***

***The Perceptron***

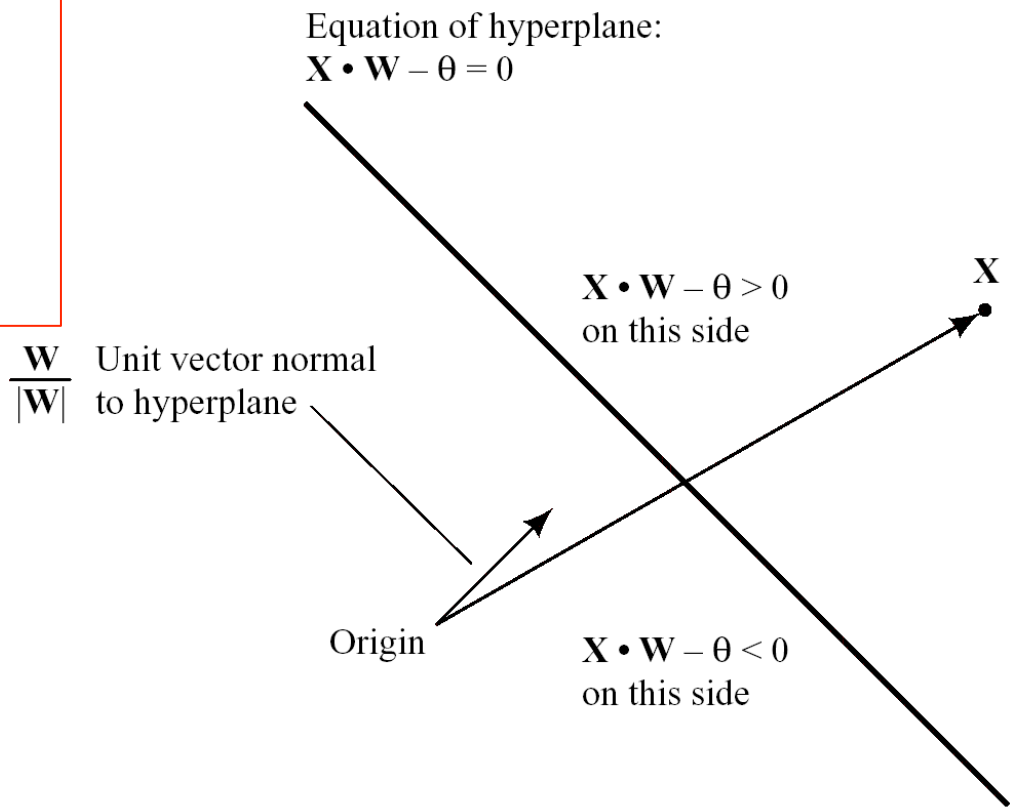


# Perceptron (3/3)

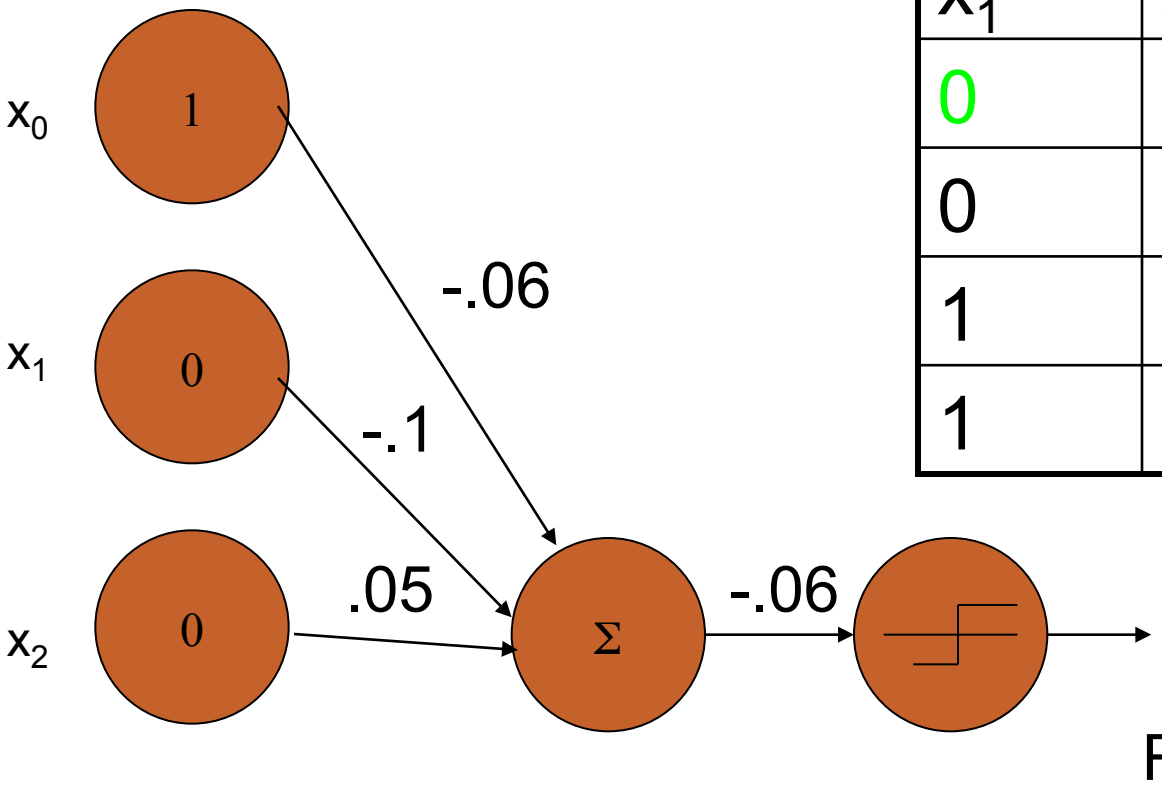


# Parameter Learning in Perceptron

**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

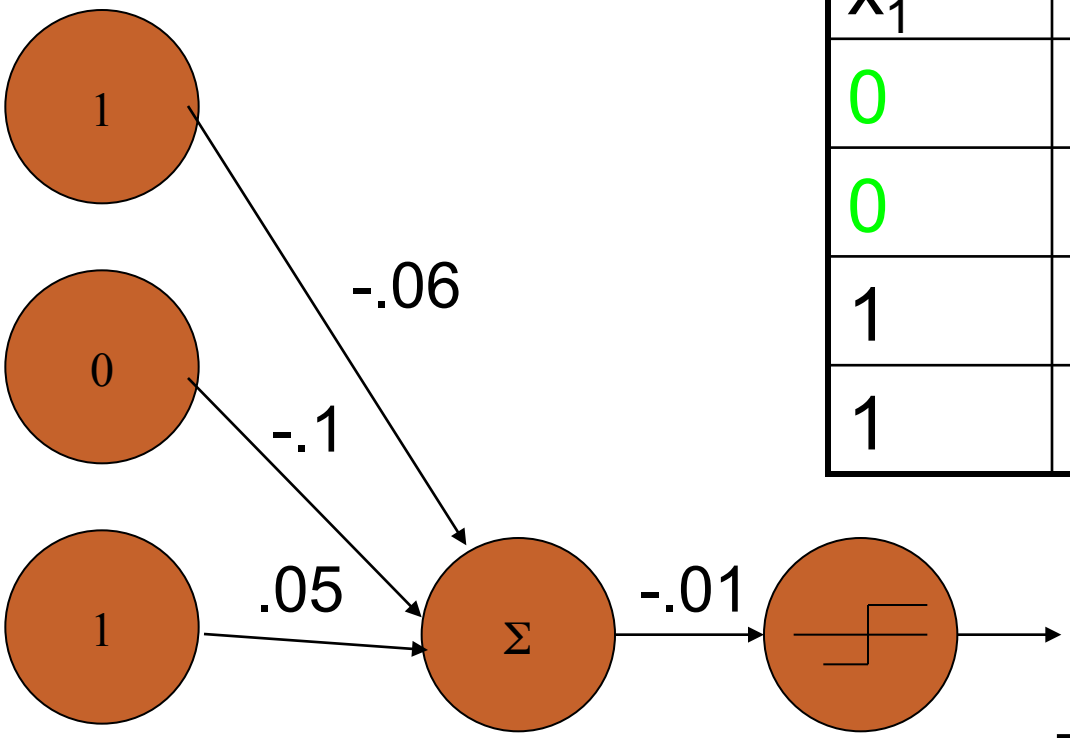


# Perceptrons



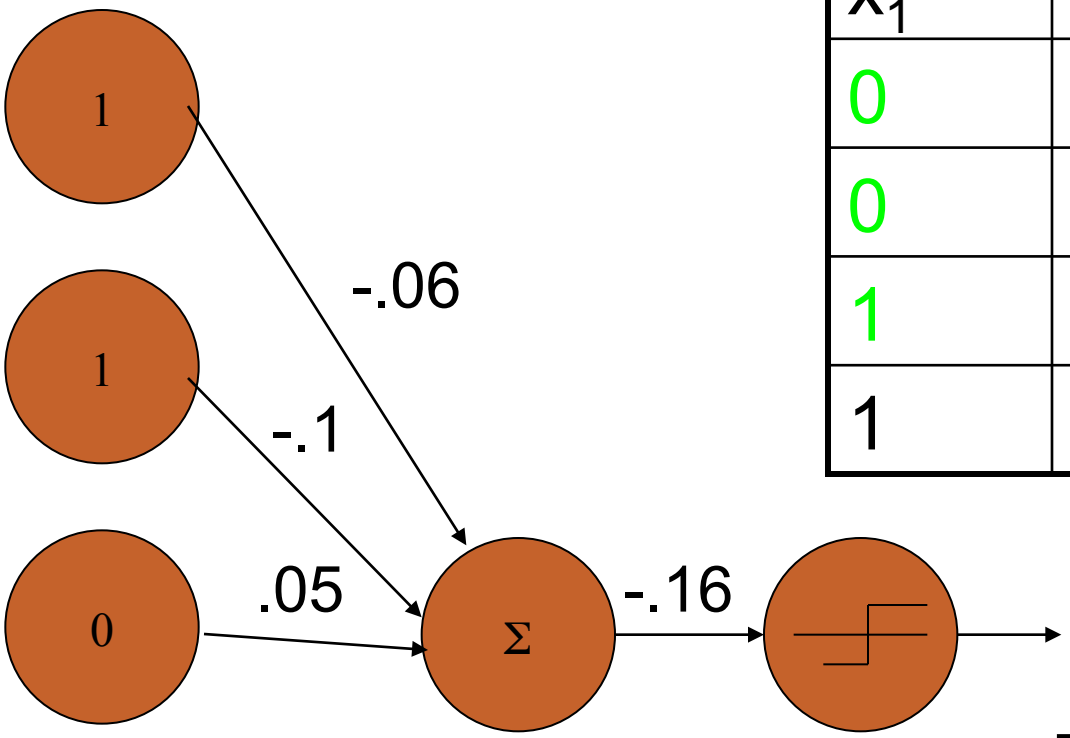

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



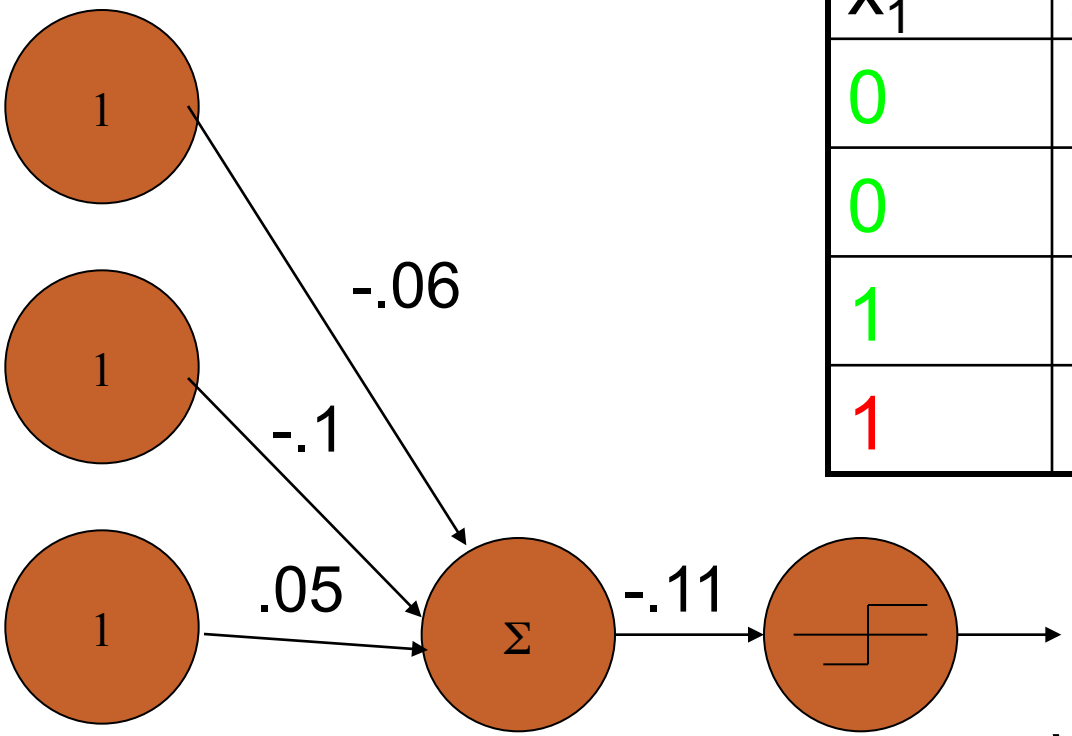

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



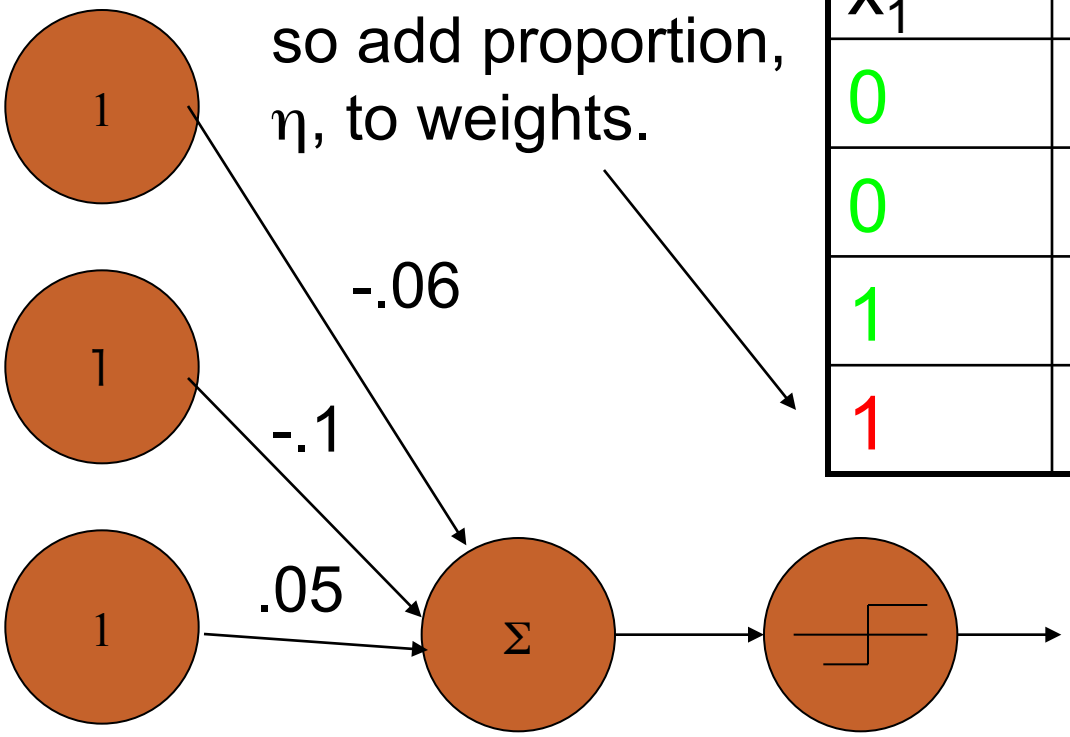

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



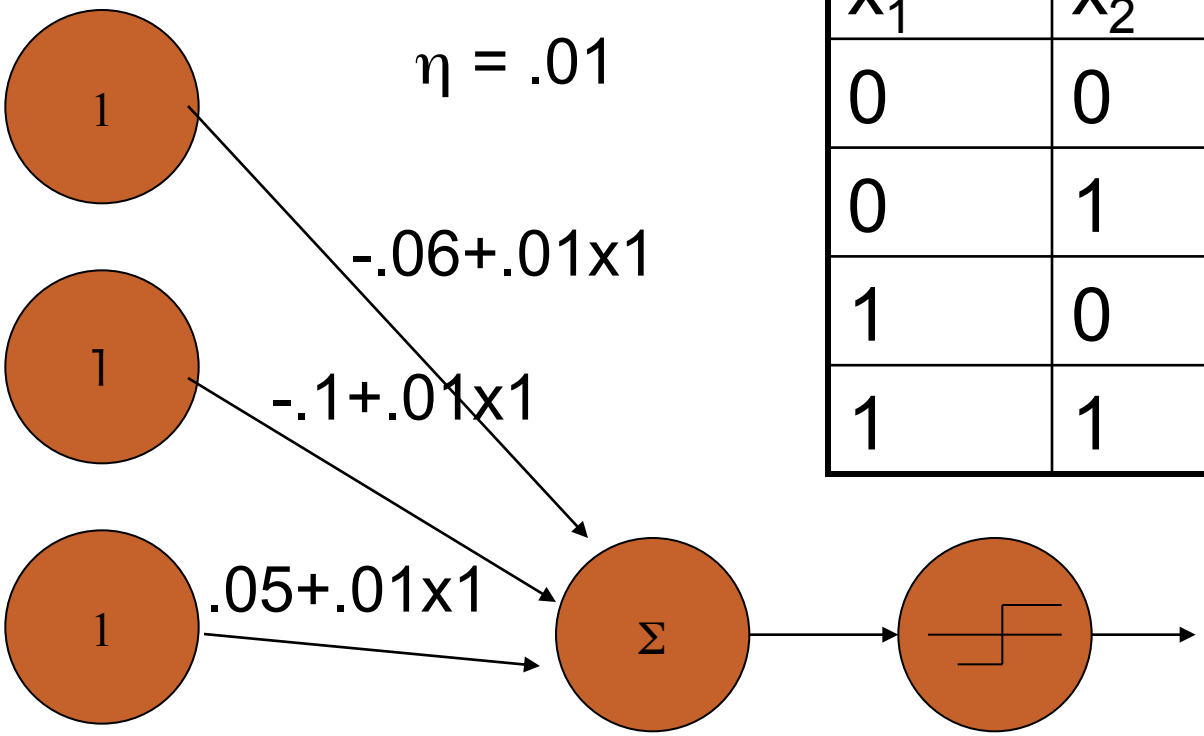

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



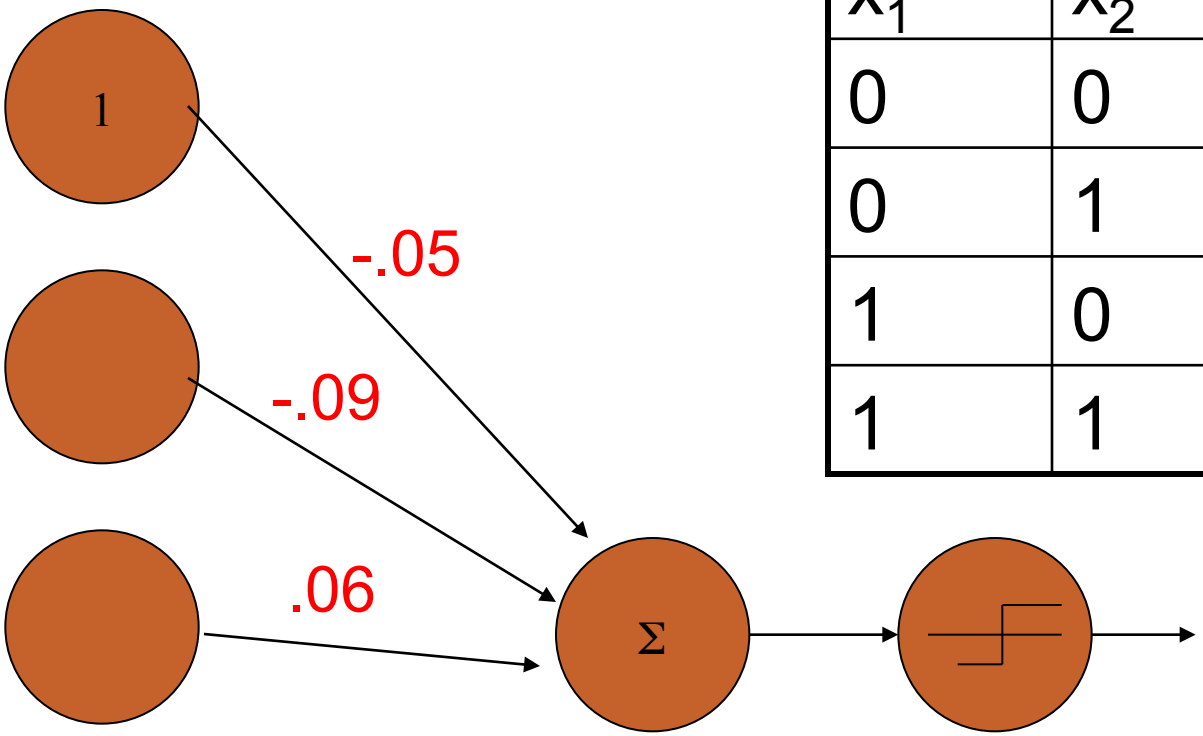

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



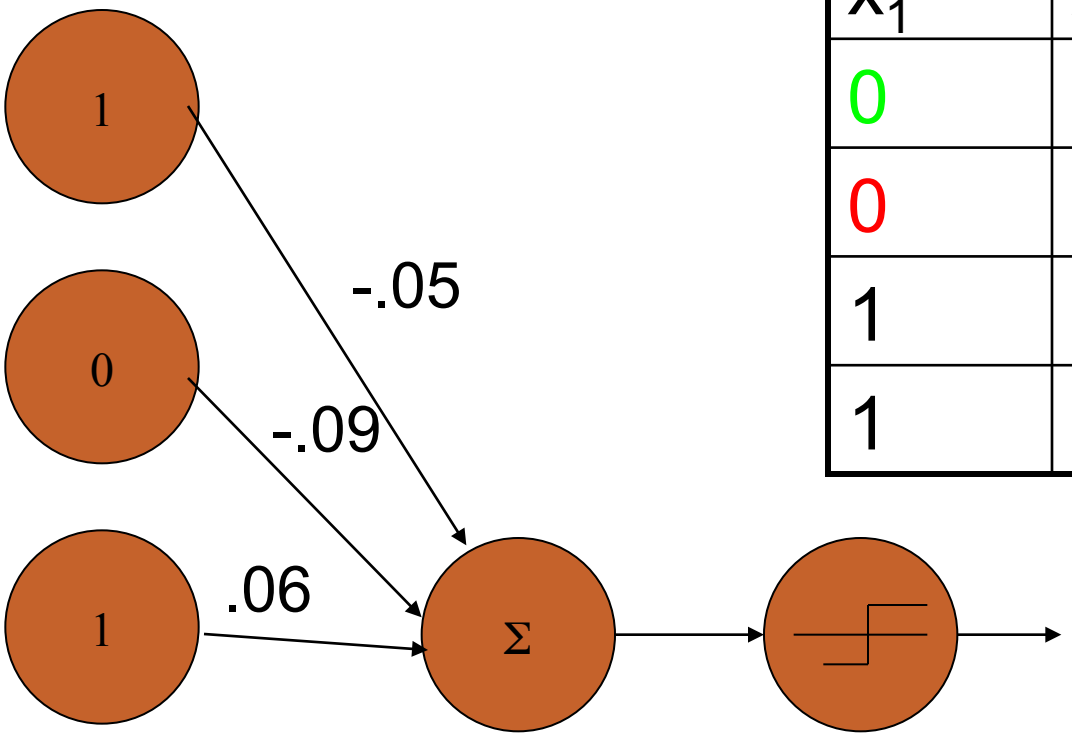

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



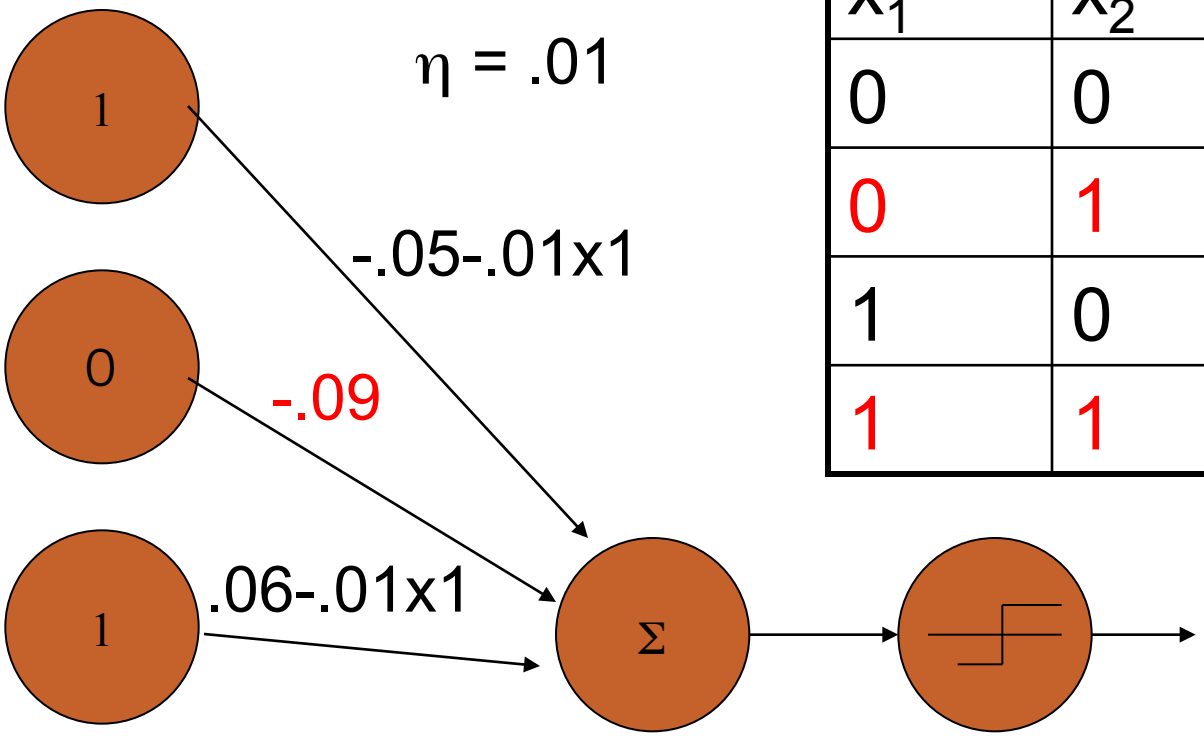

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



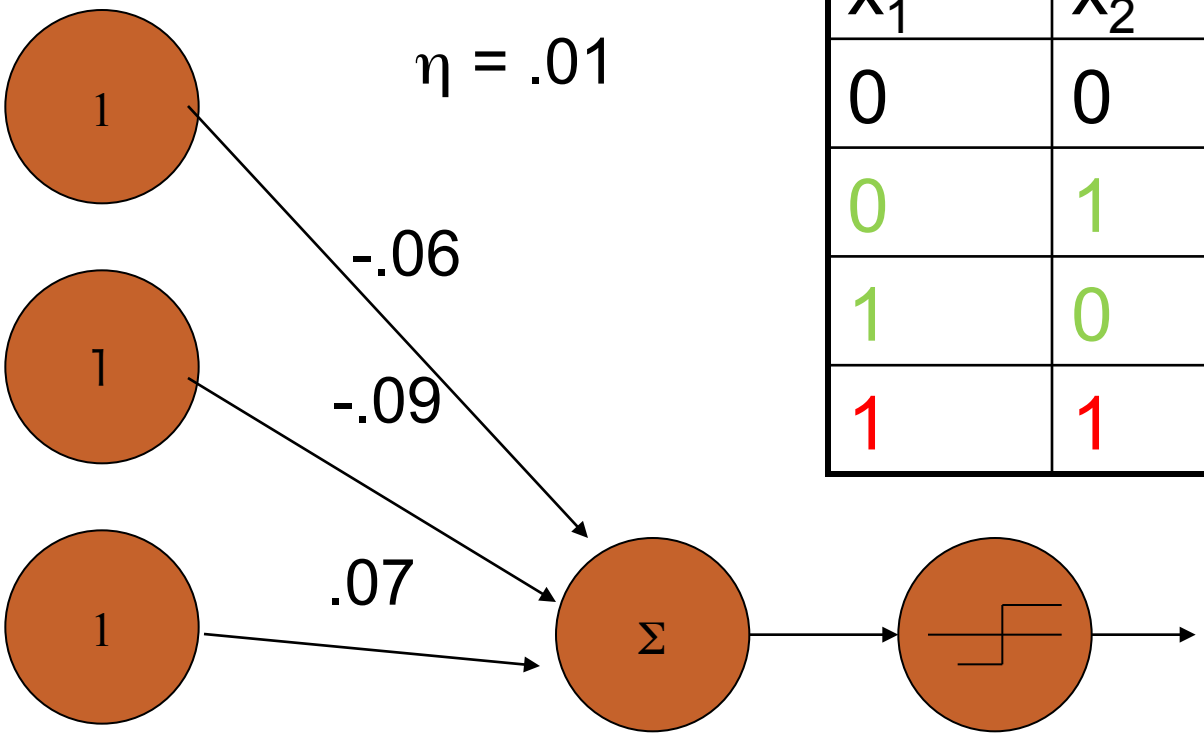

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



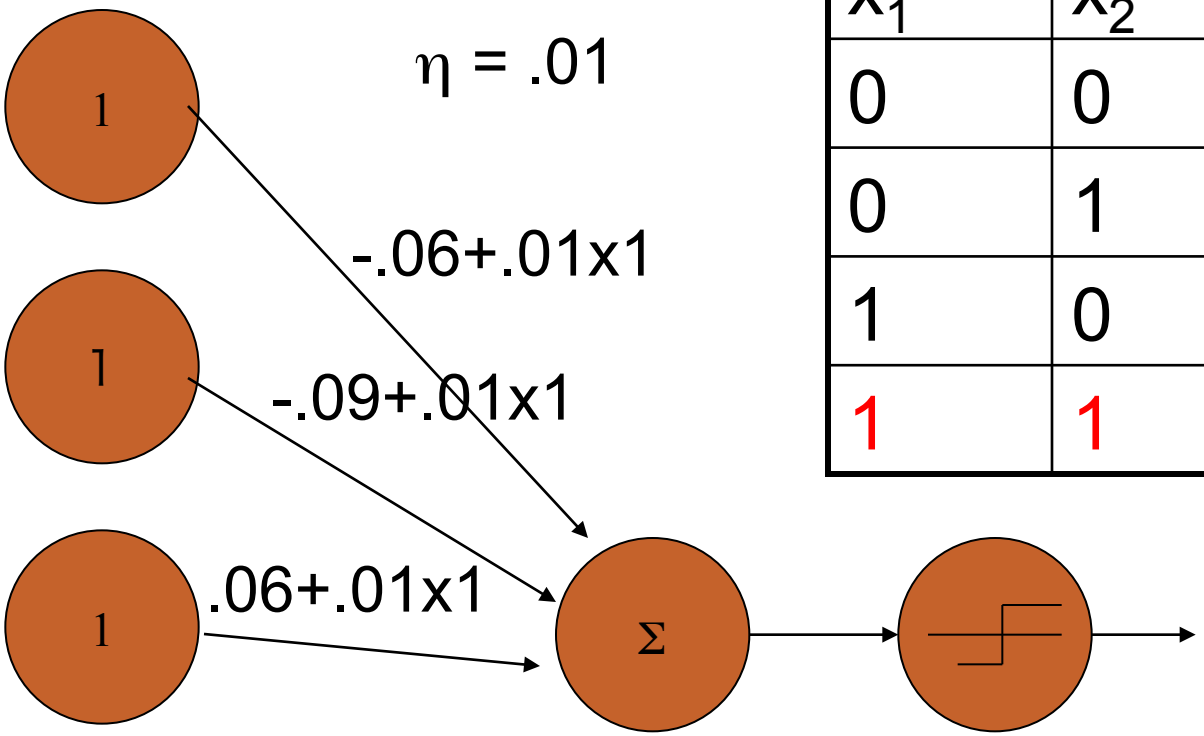

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



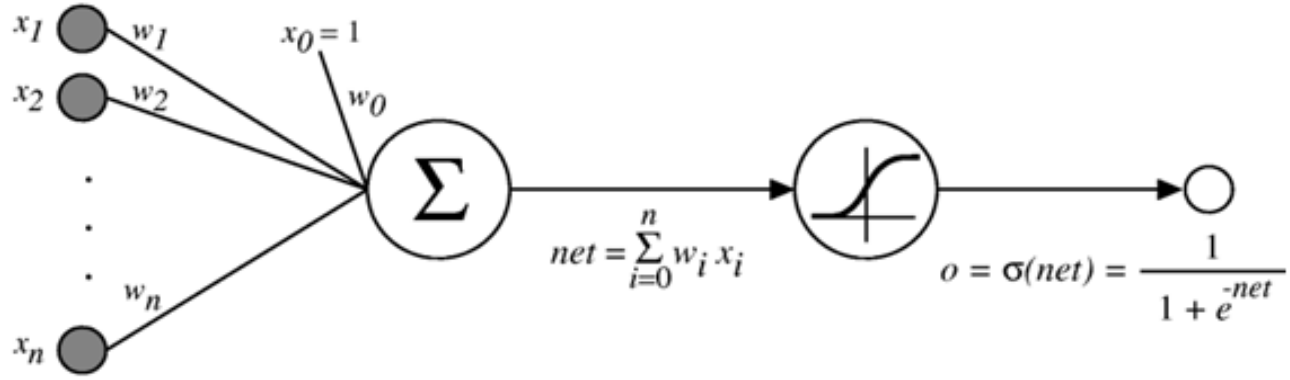

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Perceptrons



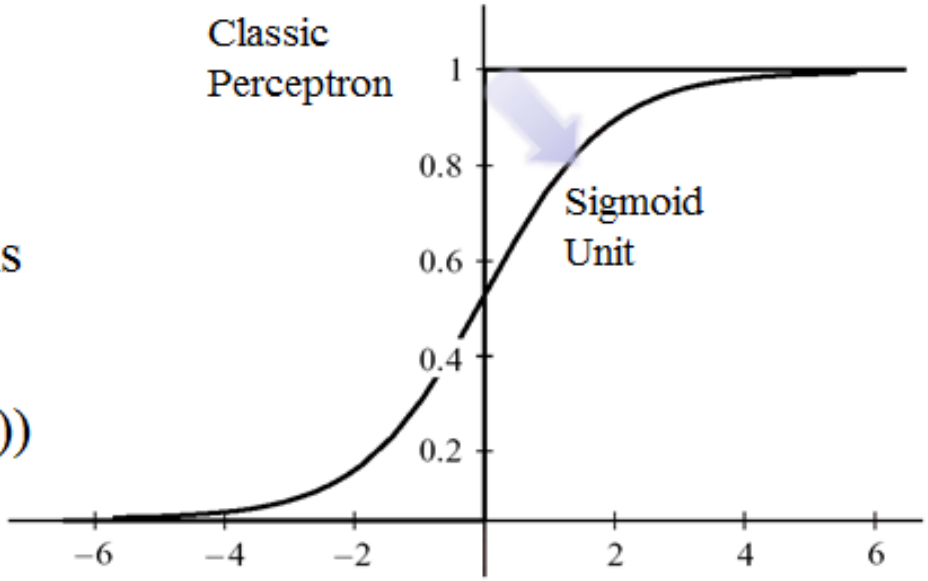

| $x_1$ | $x_2$ | o |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# 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

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

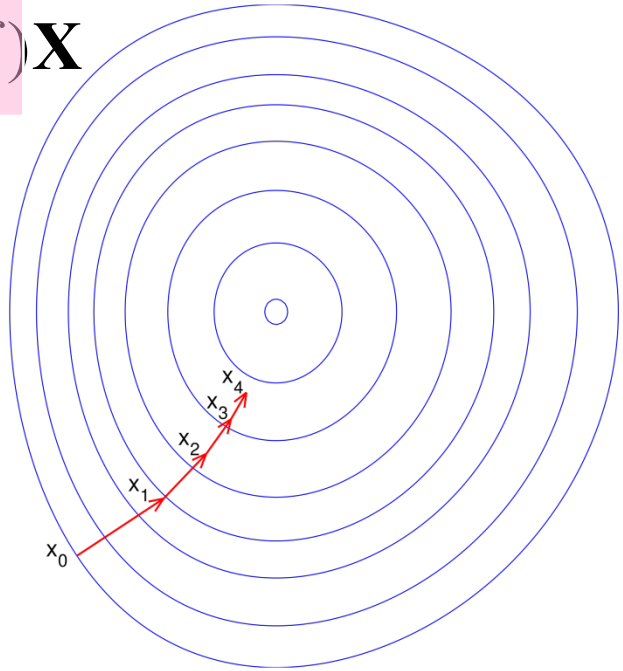
- Gradient Descent Update

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

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

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

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



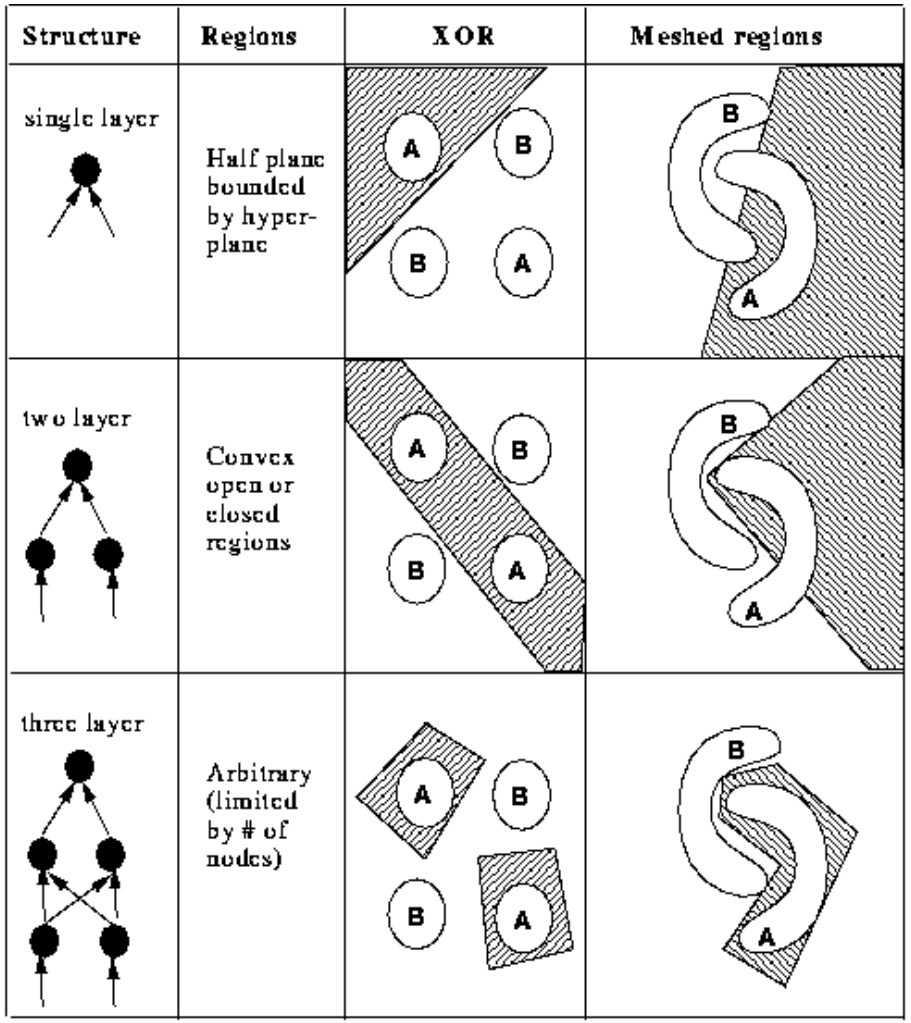
# Need for Multiple Units and Multiple Layers

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

→ Multiple Units

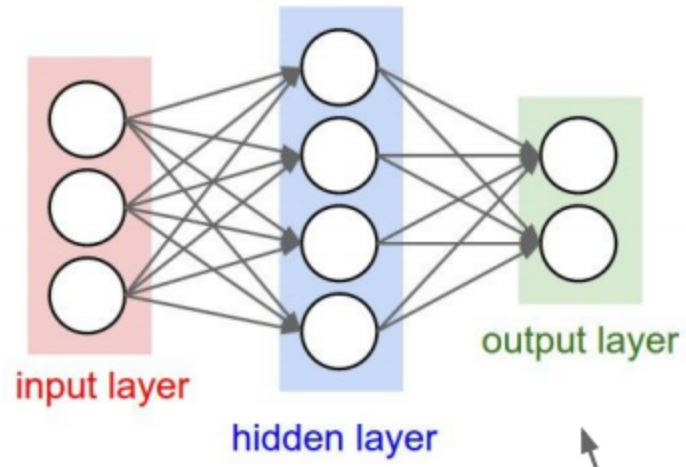
- More complex regions are needed (e.g. Polygons)

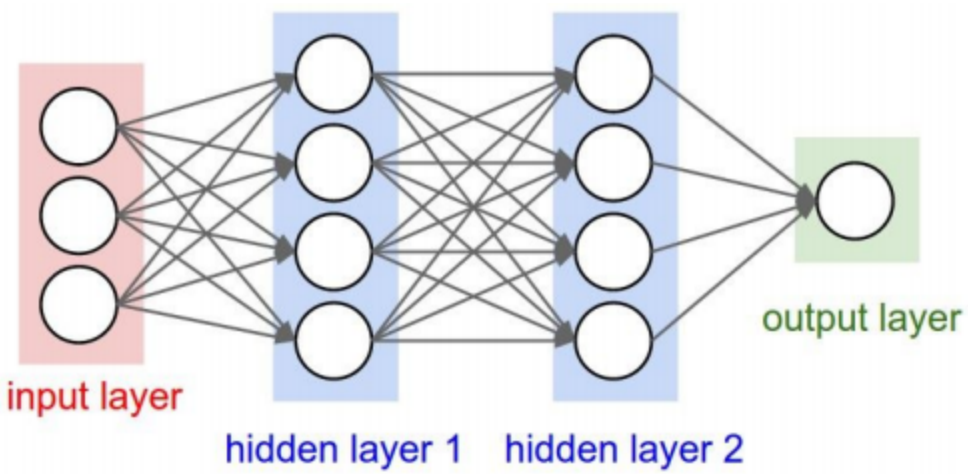
→ Multiple Layers




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

# Structure of Multilayer Perceptron



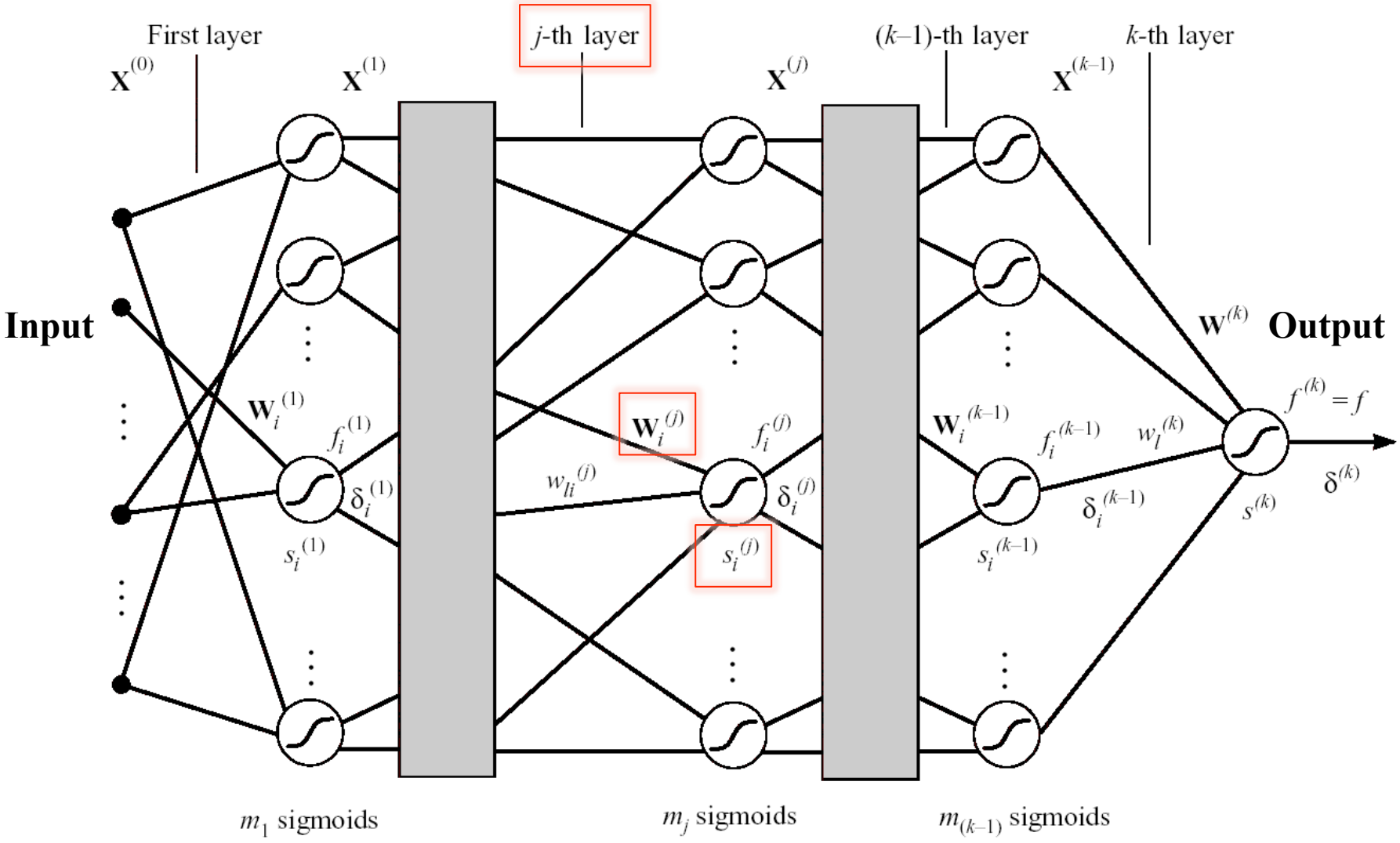

"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

**"Fully-connected" layers**

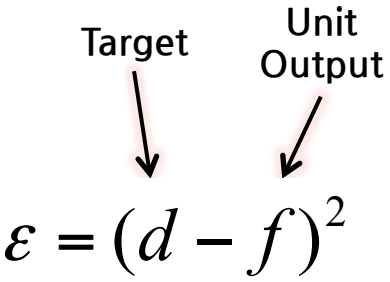
http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

# Structure of Multilayer Perceptron (MLP; Artificial Neural Network)

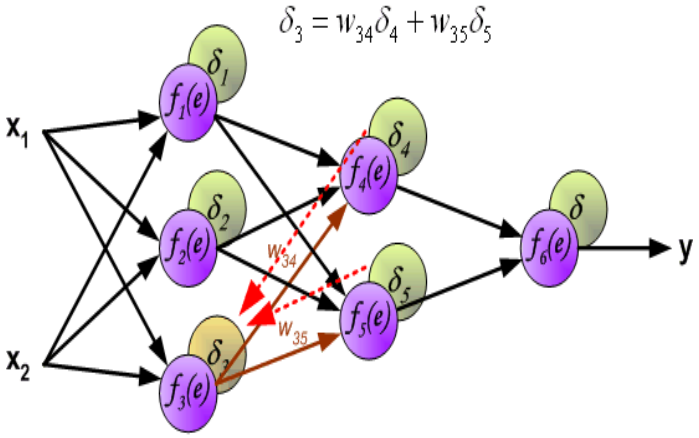


# Learning Parameters of MLP

- 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**

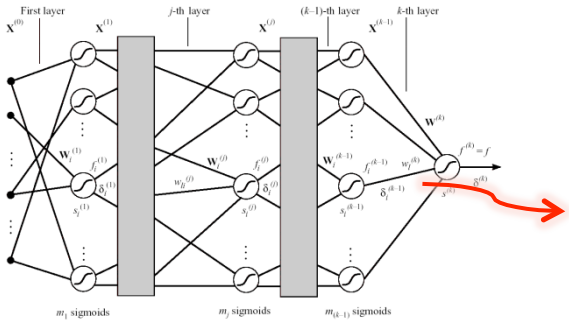



- Recursive Computation of Gradients
  - Computation of loss-gradient of **the top-layer** weights is **the same** as before
  - Using the **chain rule**, we can compute the loss-gradient of lower-layer weights **recursively (Back Propagation)**



# 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**

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

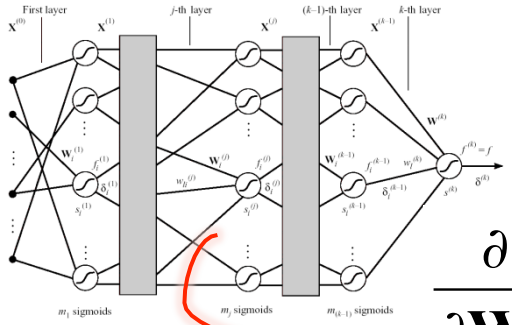
- Store intermediate value delta for later use of chain rule

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

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

# 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

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

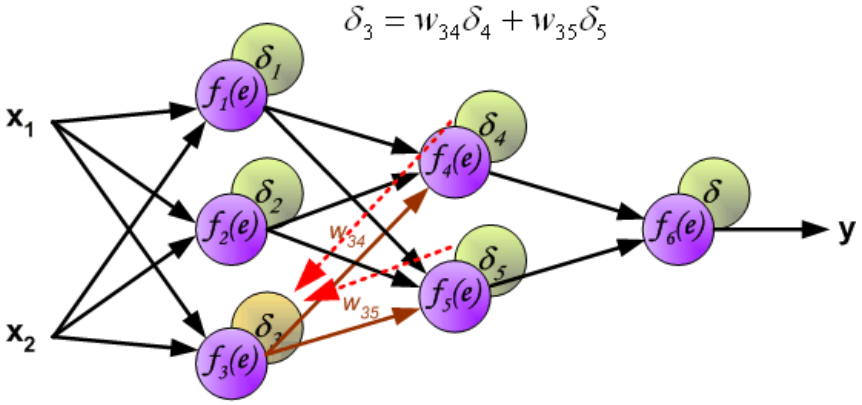
**Gradient Descent Update rule for lower-layer weights**

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

# Back Propagation Learning Algorithm (3/3)

- Applying chain rule, recursive relation between delta's

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



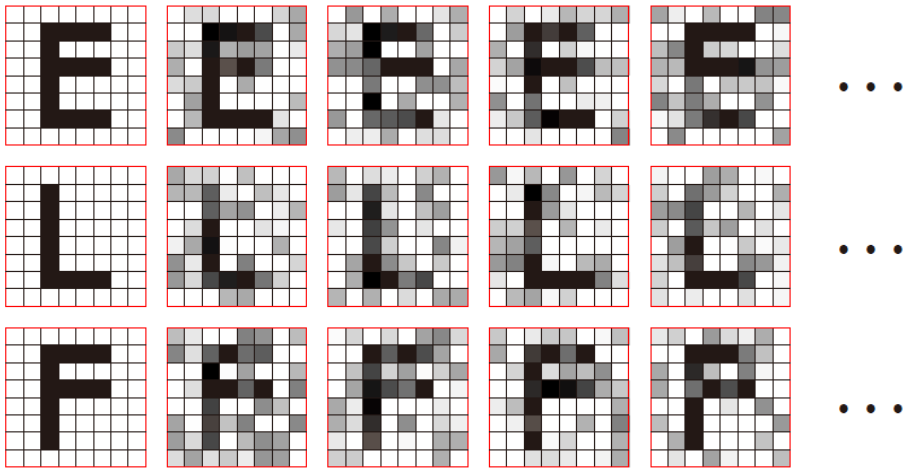

**Algorithm: Back Propagation**

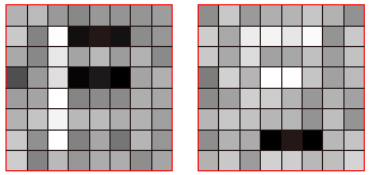
1. **Randomly Initialize weight parameters**
2. **Calculate the activations of all units (with input data)**
3. **Calculate top-layer delta**
4. **Back-propagate delta from top to the bottom**
5. **Calculate actual gradient of all units using delta's**
6. **Update weights using Gradient Descent rule**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

# An example of the MLP

- Example

(a) sample training patterns



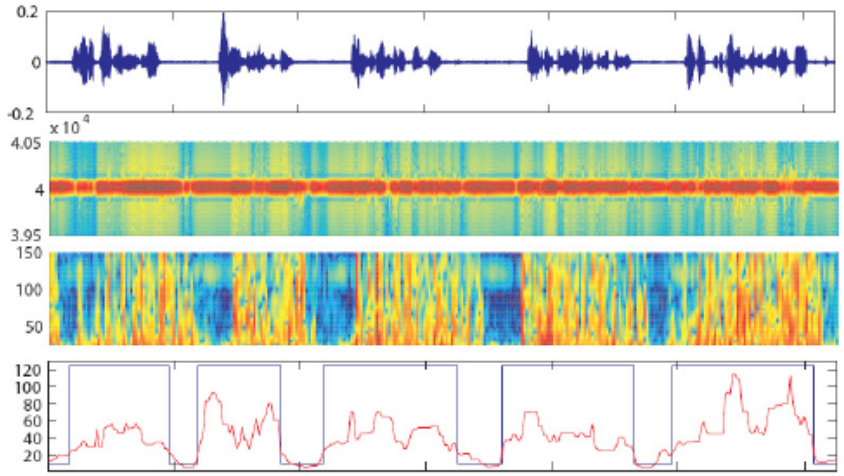


(b) learned input-to-hidden weights

- 64-2-3 network for classifying 3 characters
  - 64-dim inputs
  - 2 hidden units
  - 3 output units
- Learned i-to-h weights
  - Describe feature groupings useful for classification

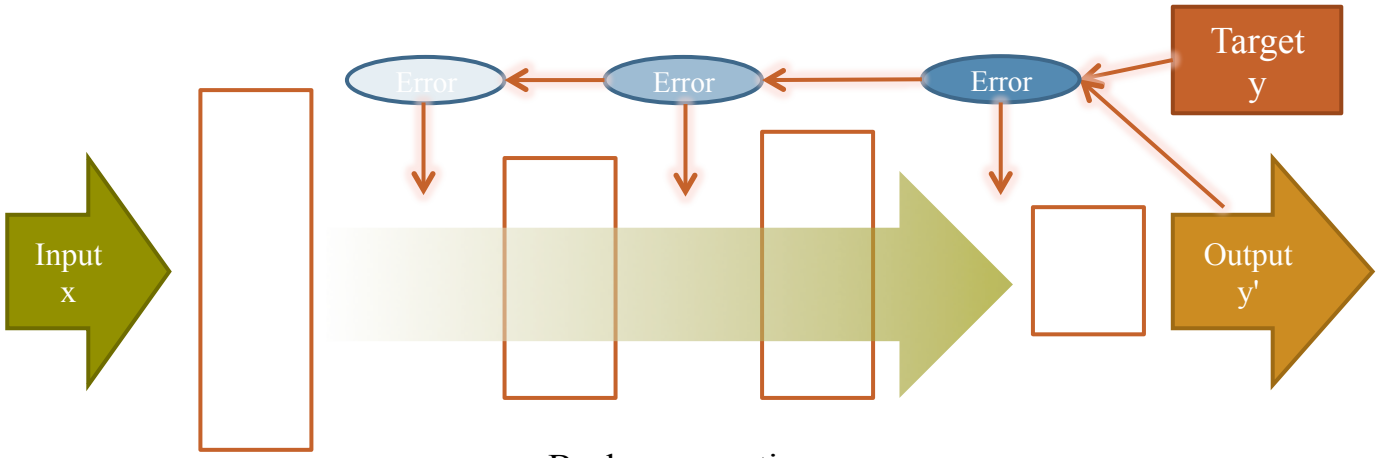
# Applications

- 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

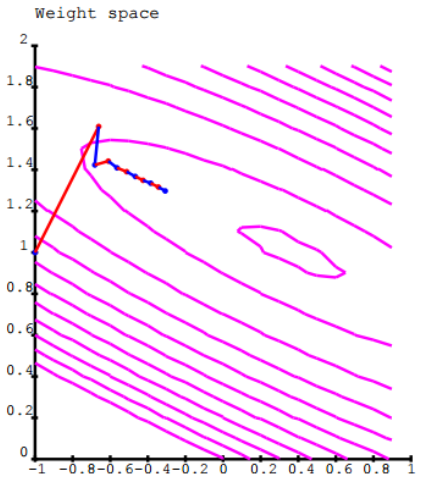


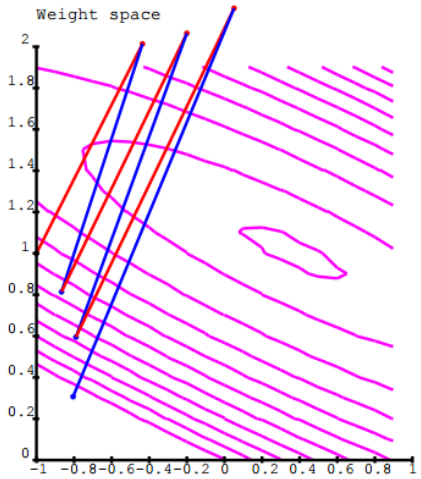

Back-propagation

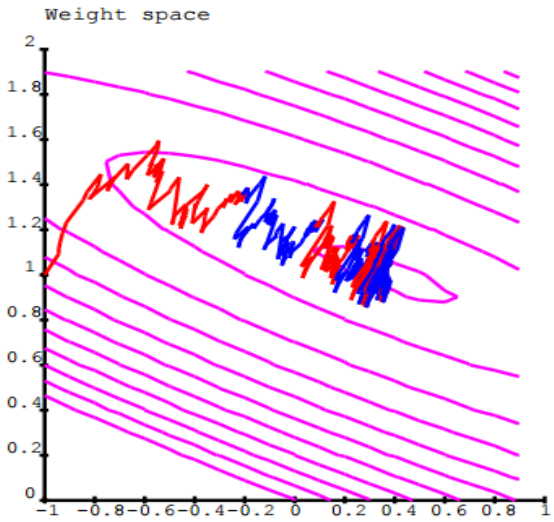
- Breakthrough
  - Deep Belief Networks (Unsupervised Pre-training)
  - Convolutional Neural Networks (Reducing Redundant Parameters)
  - Rectified Linear Unit (Constant Gradient Propagation)

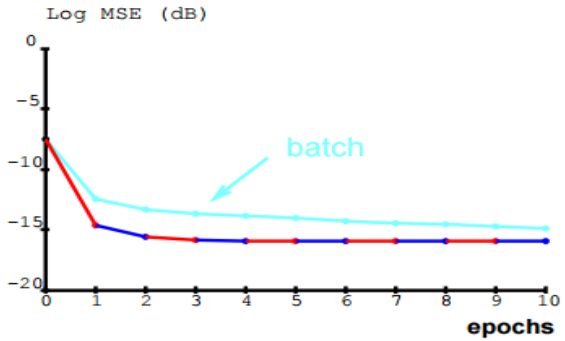
# Some Issues (1/3)

- Stochastic Gradient Descent



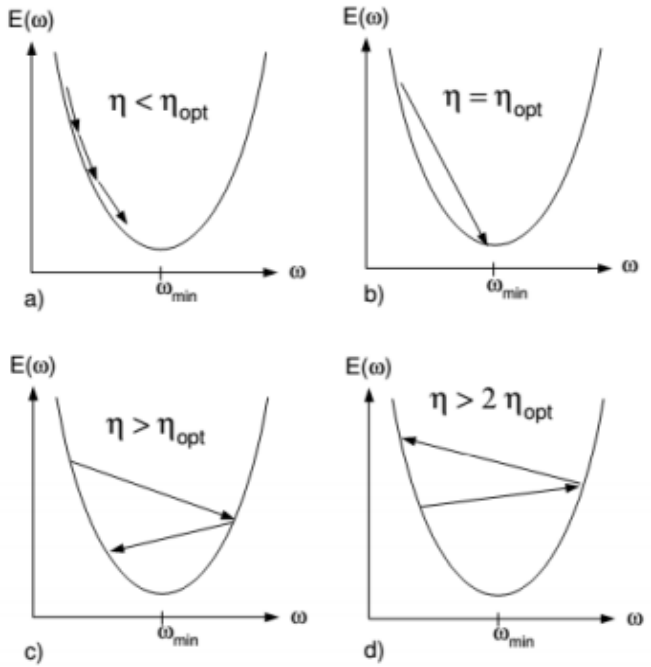



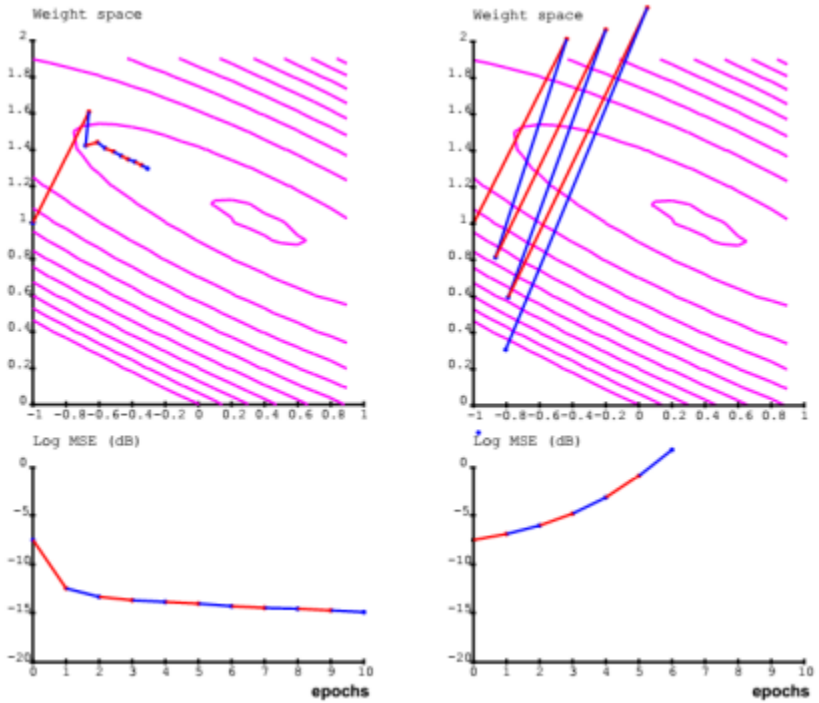




http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

# Some Issues (2/3)

- Learning Rate Adaptation
- Momentum
- Weight Decay



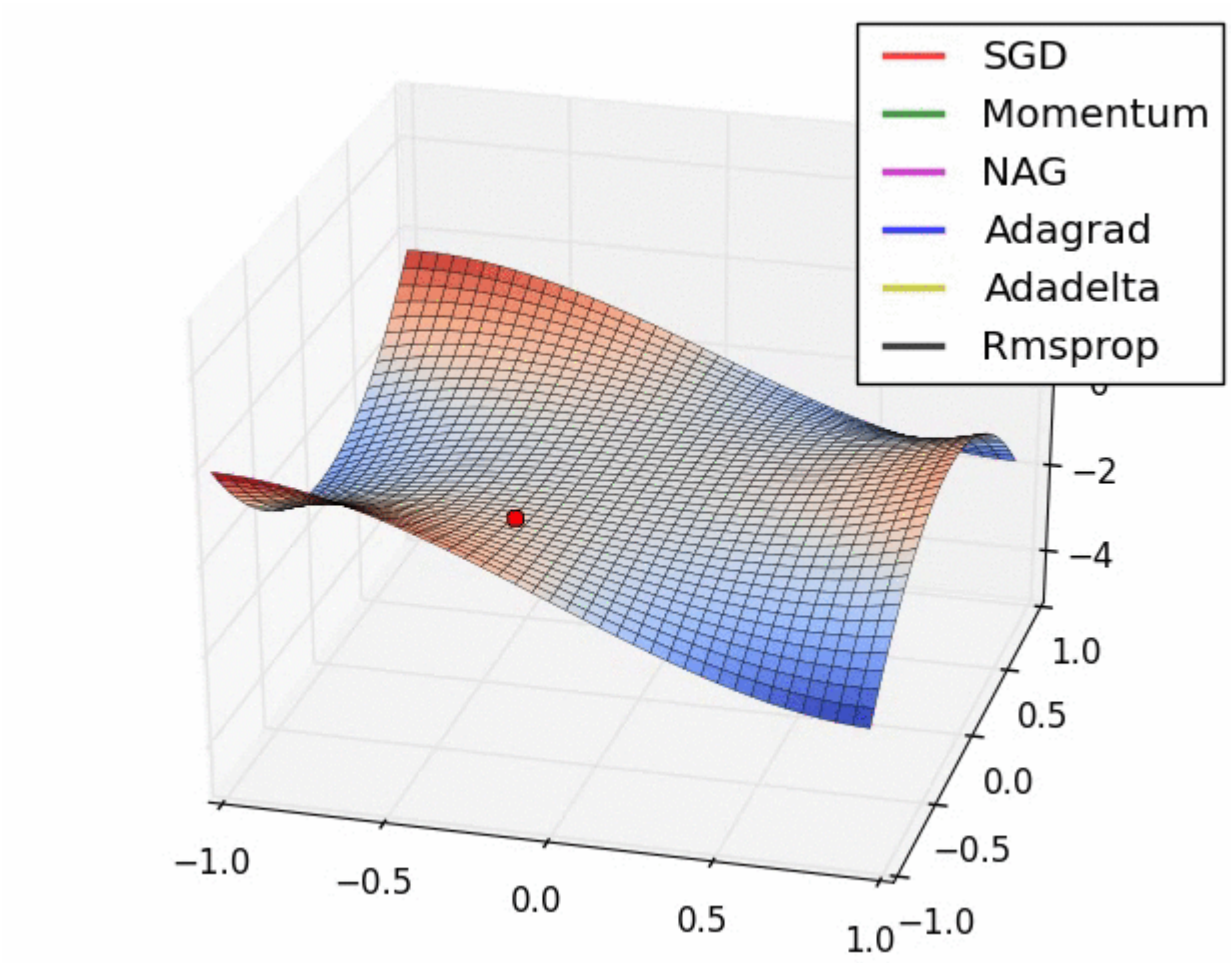




http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

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

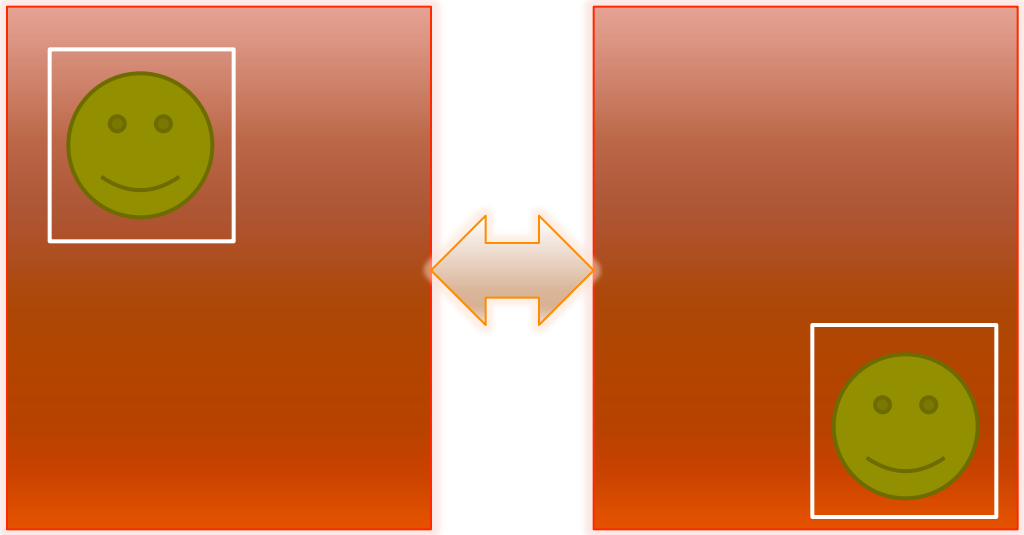
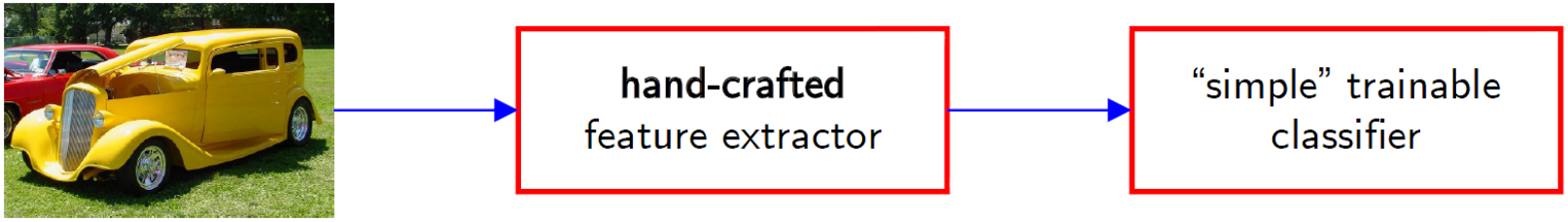


Image 1 Image 2

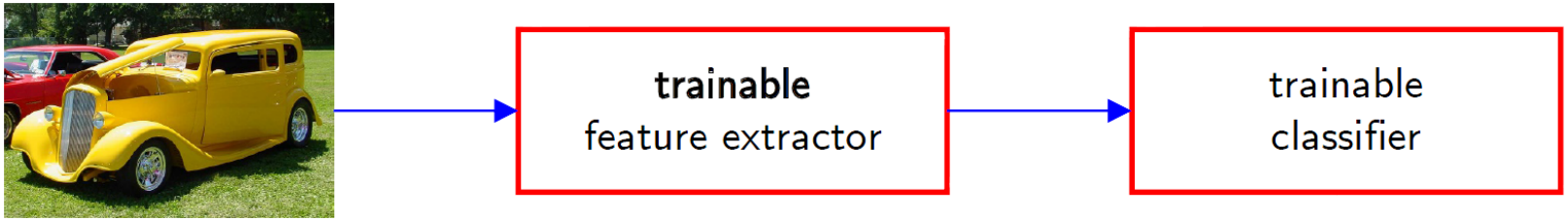
- DBN의 경우 다른 data
- CNN의 경우 같은 data

# End-to-End training

- Learning hierarchical representations
- Traditional model of pattern recognition (since the late 50's)
  - fixed / engineered features (or kernel) + trainable classifier

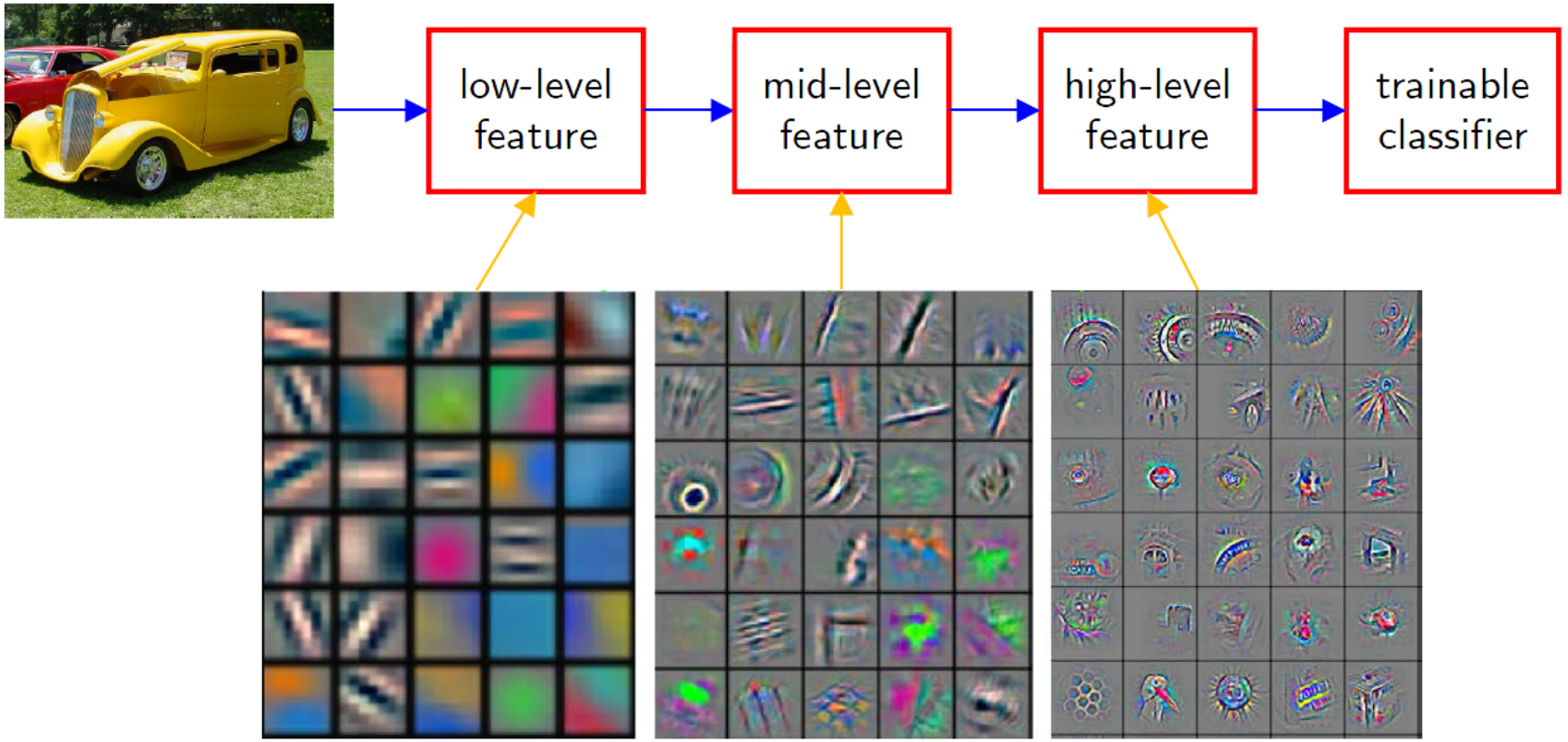



- End-to-end learning / feature learning / deep learning
  - Trainable features (or kernel) + trainable classifier



# Convolutional Neural Networks

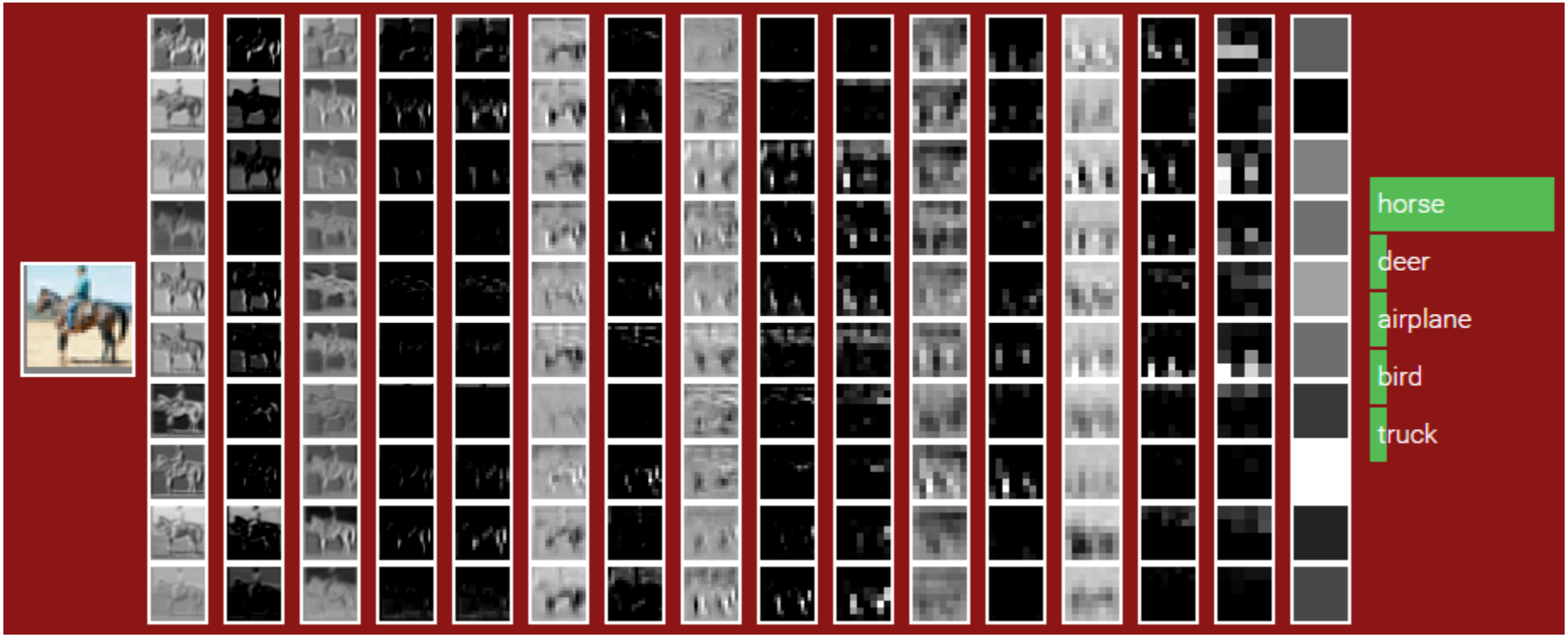
- Introduction
  - Learning hierarchical representations




feature visualization of CNN trained on ImageNet (Zeiler and Fergus 2013)

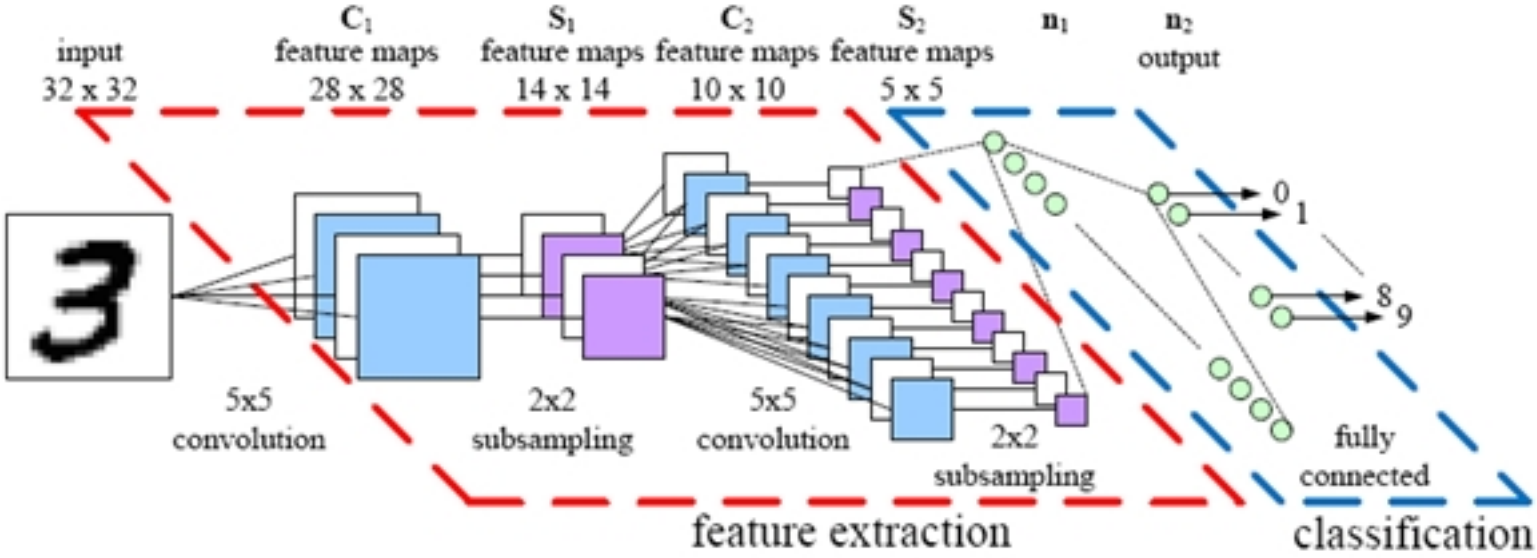
# Convolutional Neural Networks

- Introduction
  - Learning hierarchical representations



# Structure of Convolutional Neural Network (CNN)

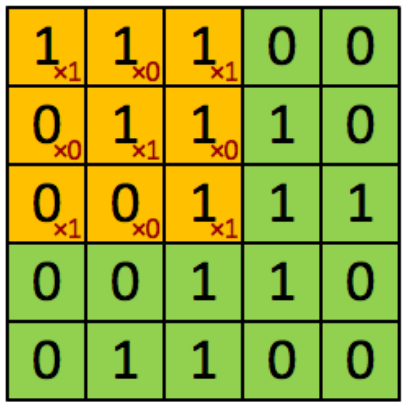
- Convolution과 Pooling (Subsampling)을 반복하여 상위 Feature를 구성
- Convolution은 Local영역에서의 특정 Feature를 얻는 과정
- Pooling은 Dimension을 줄이면서도, Translation-invariant한 Feature를 얻는 과정



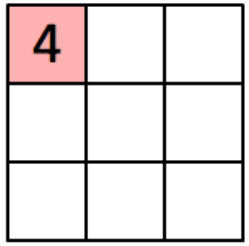

http://parse.ele.tue.nl/education/cluster2

# Convolution Layer

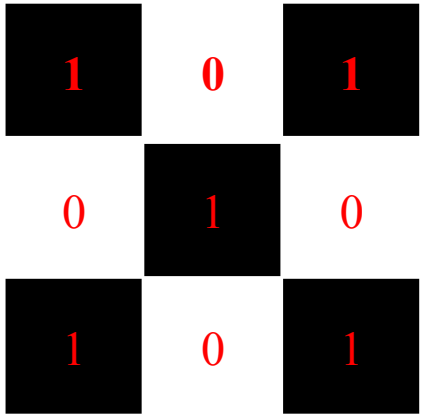
- The Kernel Detects pattern:




Image



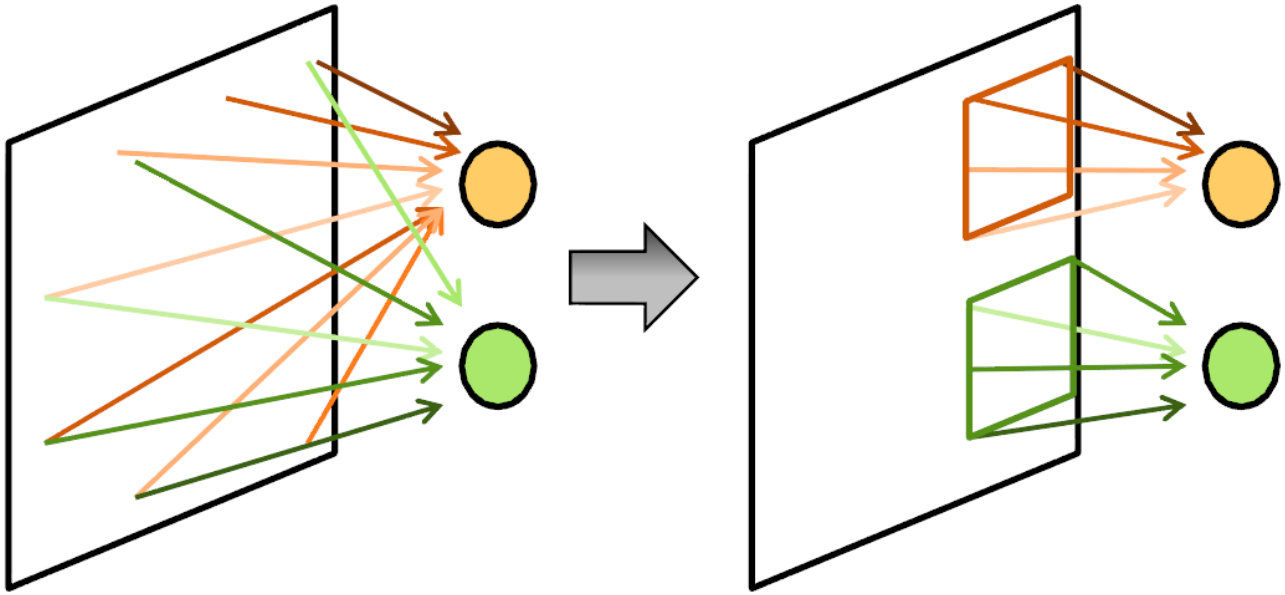

Convolved Feature




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

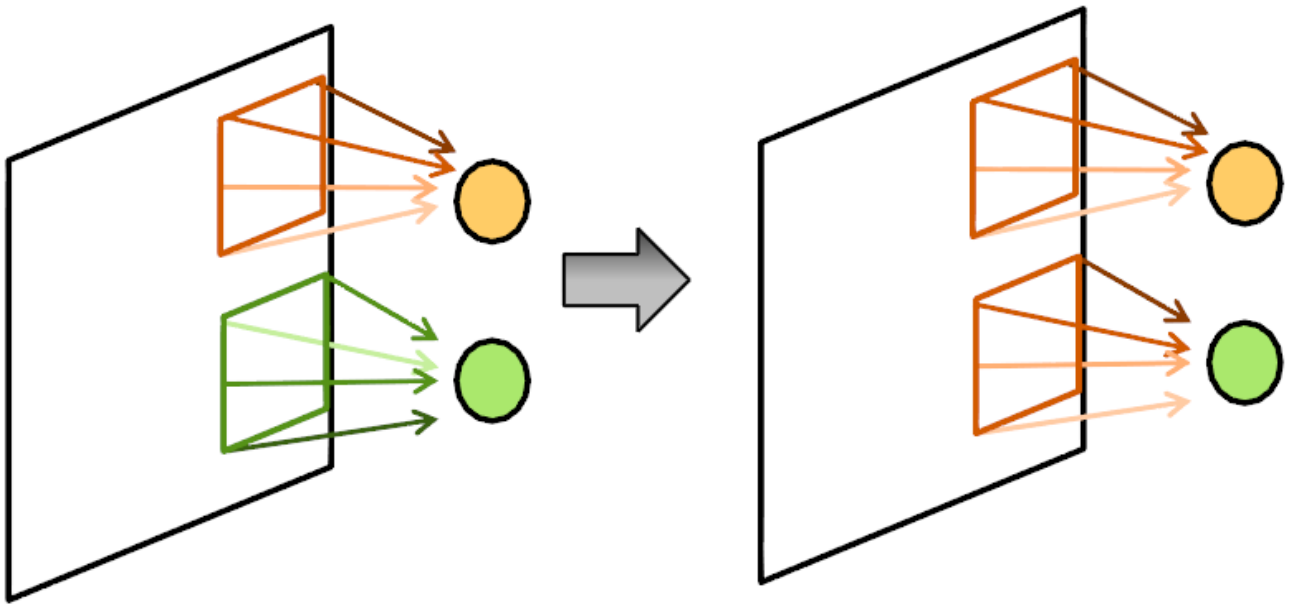
# Convolutional Neural Networks

- Convolutional Layer
  - Local connectivity
    - Connect a neuron to a local region of the input volume (Receptive field / Filter size / Kernel size)
    - Ex) input size [32X32X3], receptive field (5X5) then [5X5X3]region is connected



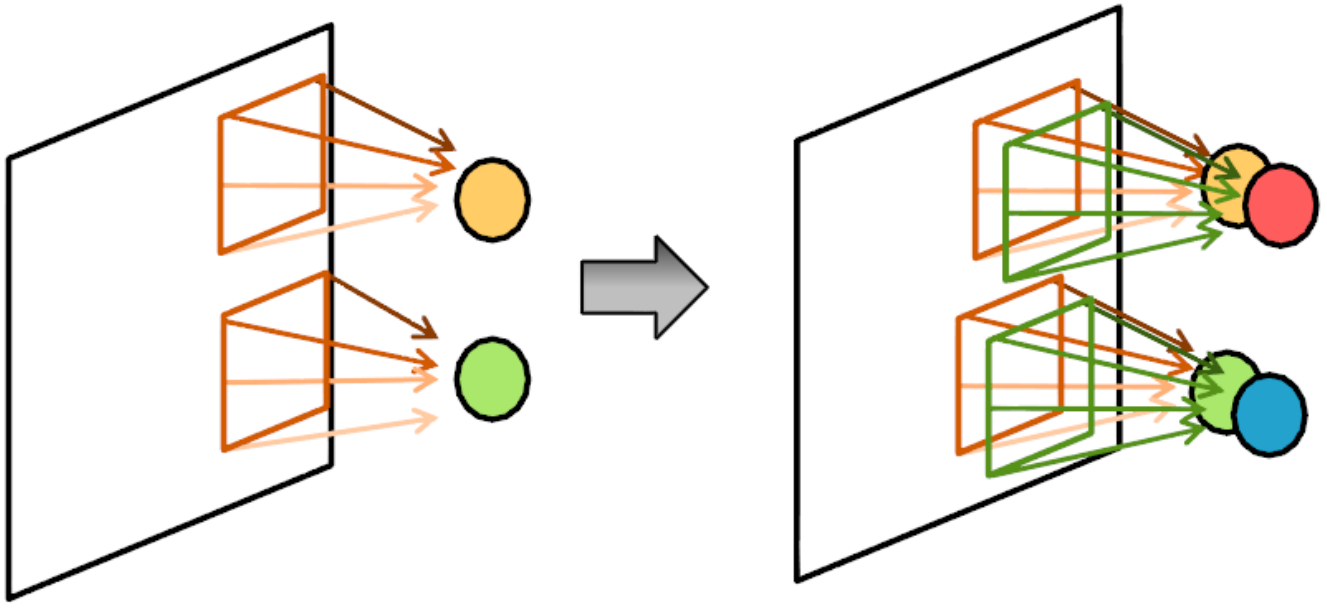
# Convolutional Neural Networks

- Convolutional Layer
  - Parameter sharing
    - Image statistics are invariant over all image location
    - Number of parameter reduced



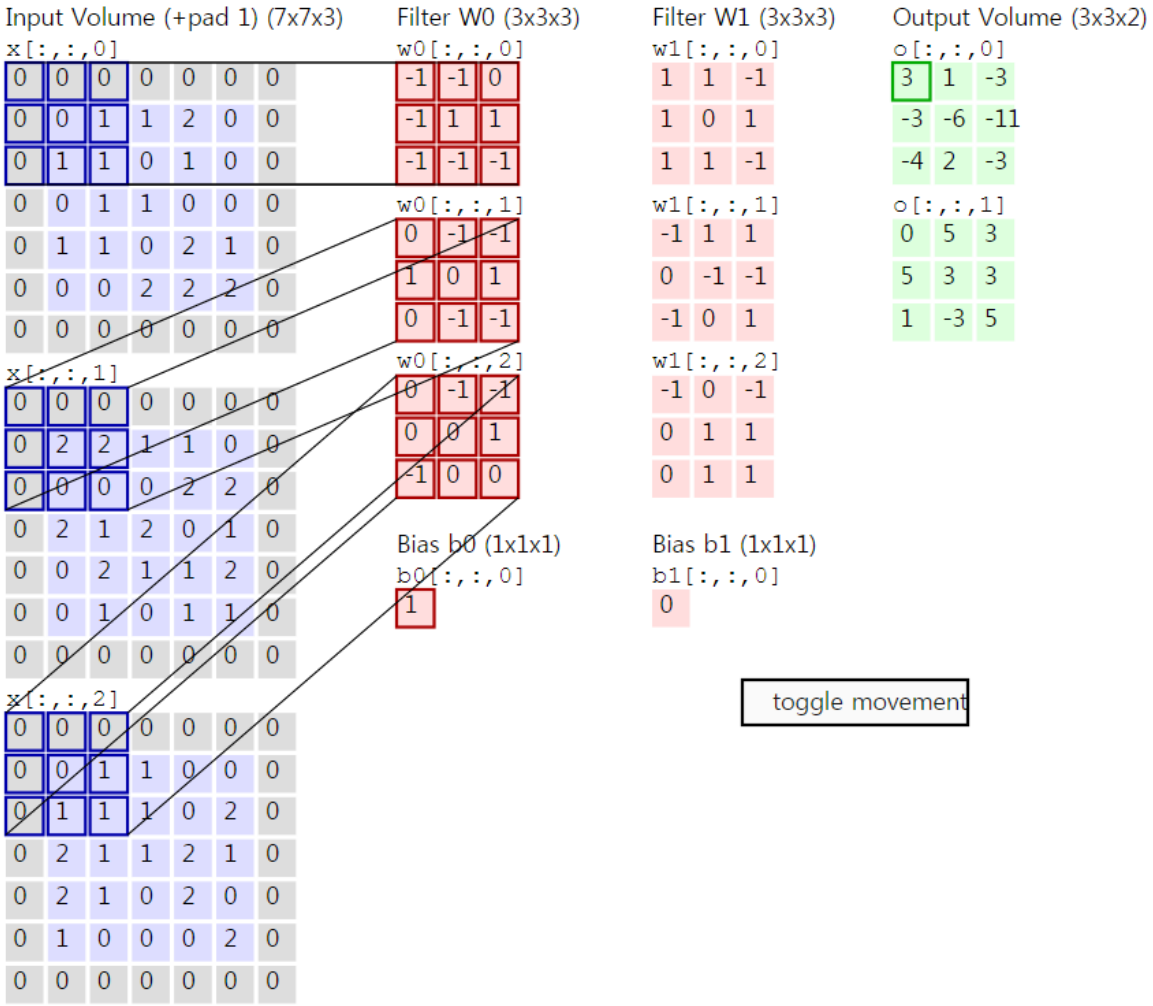
# Convolutional Neural Networks

- Convolutional Layer
  - Convolution kernels
    - Learn multiple kernels
    - Still much fewer parameters than fully connected model



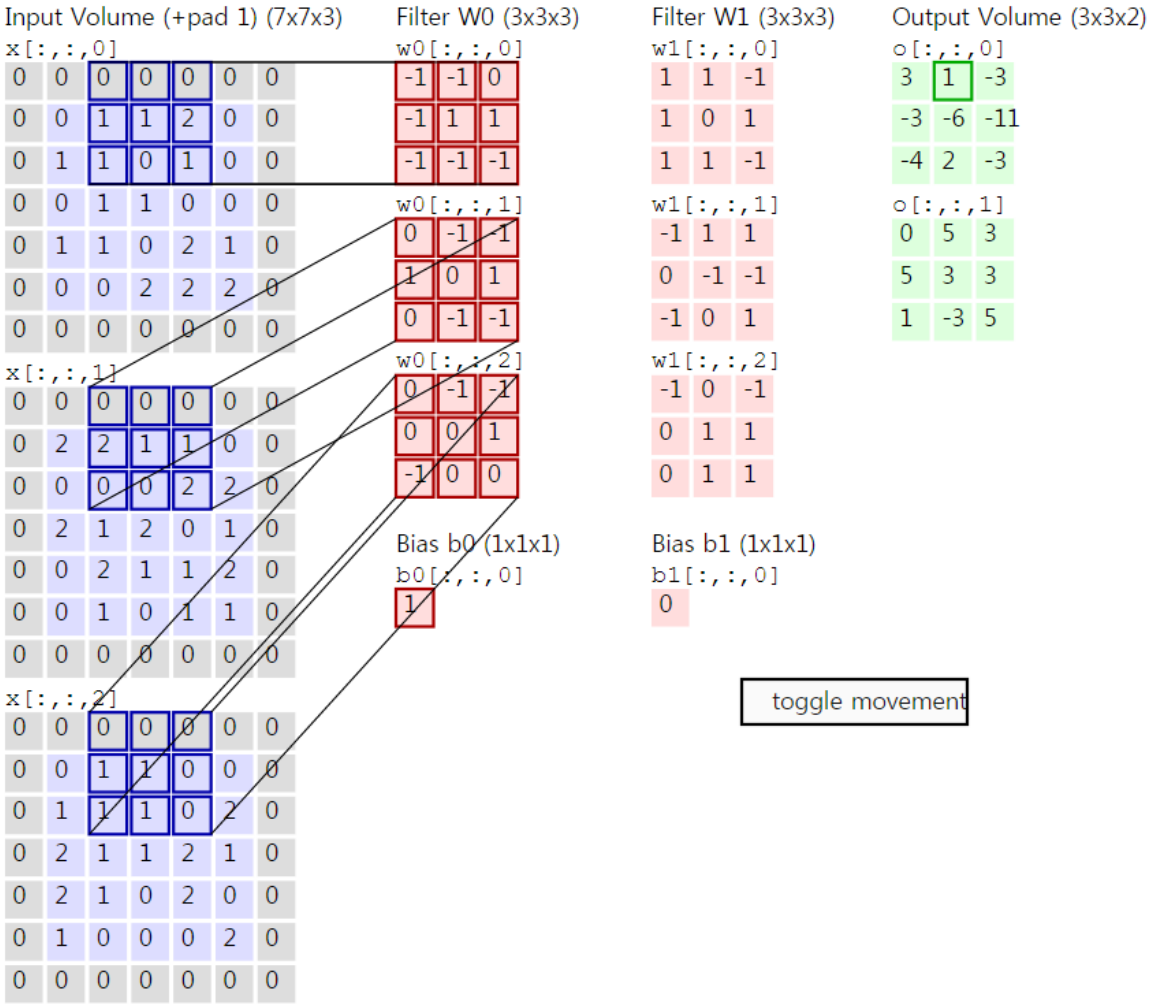
# Convolutional Neural Networks

- Convolutional Layer



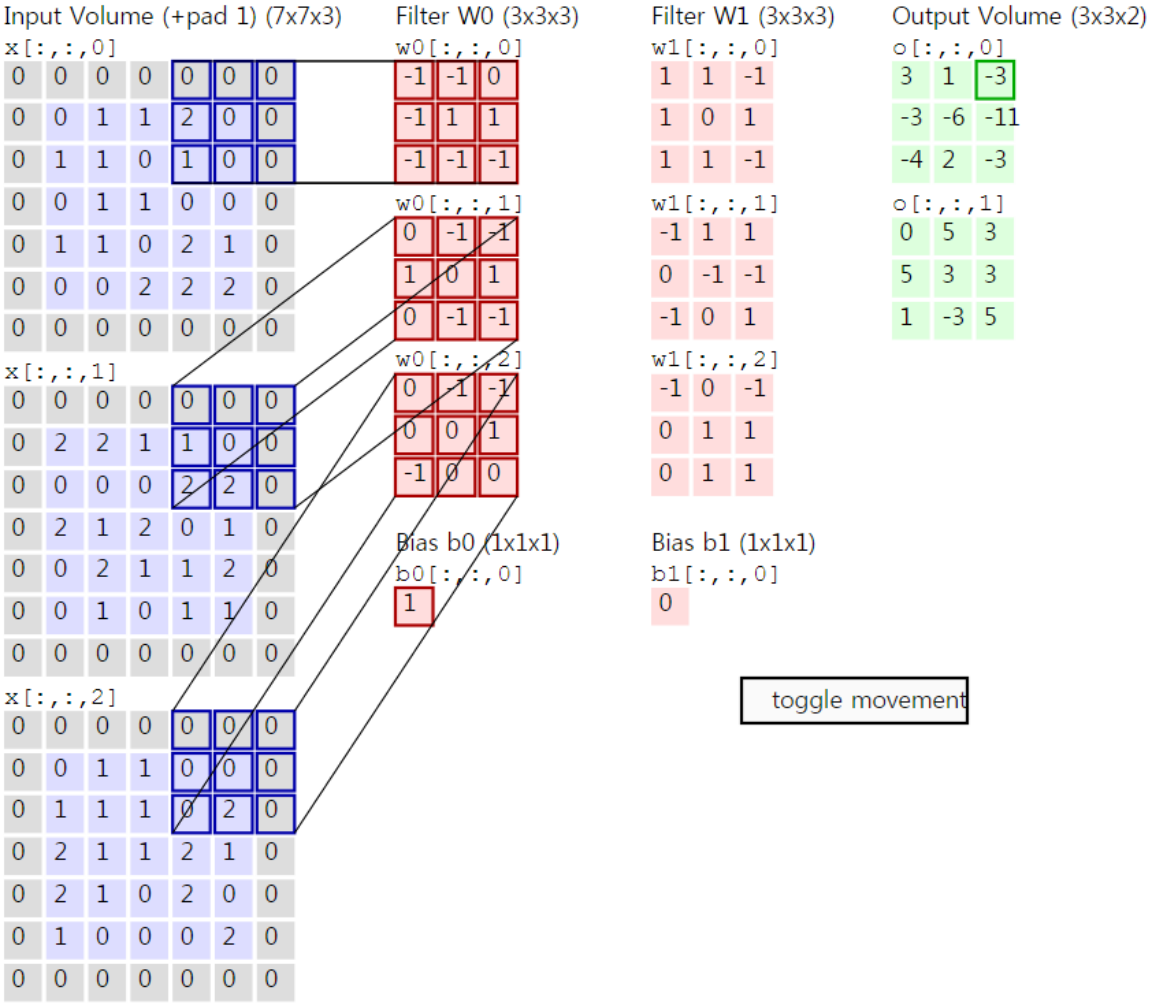
# Convolutional Neural Networks

- Convolutional Layer



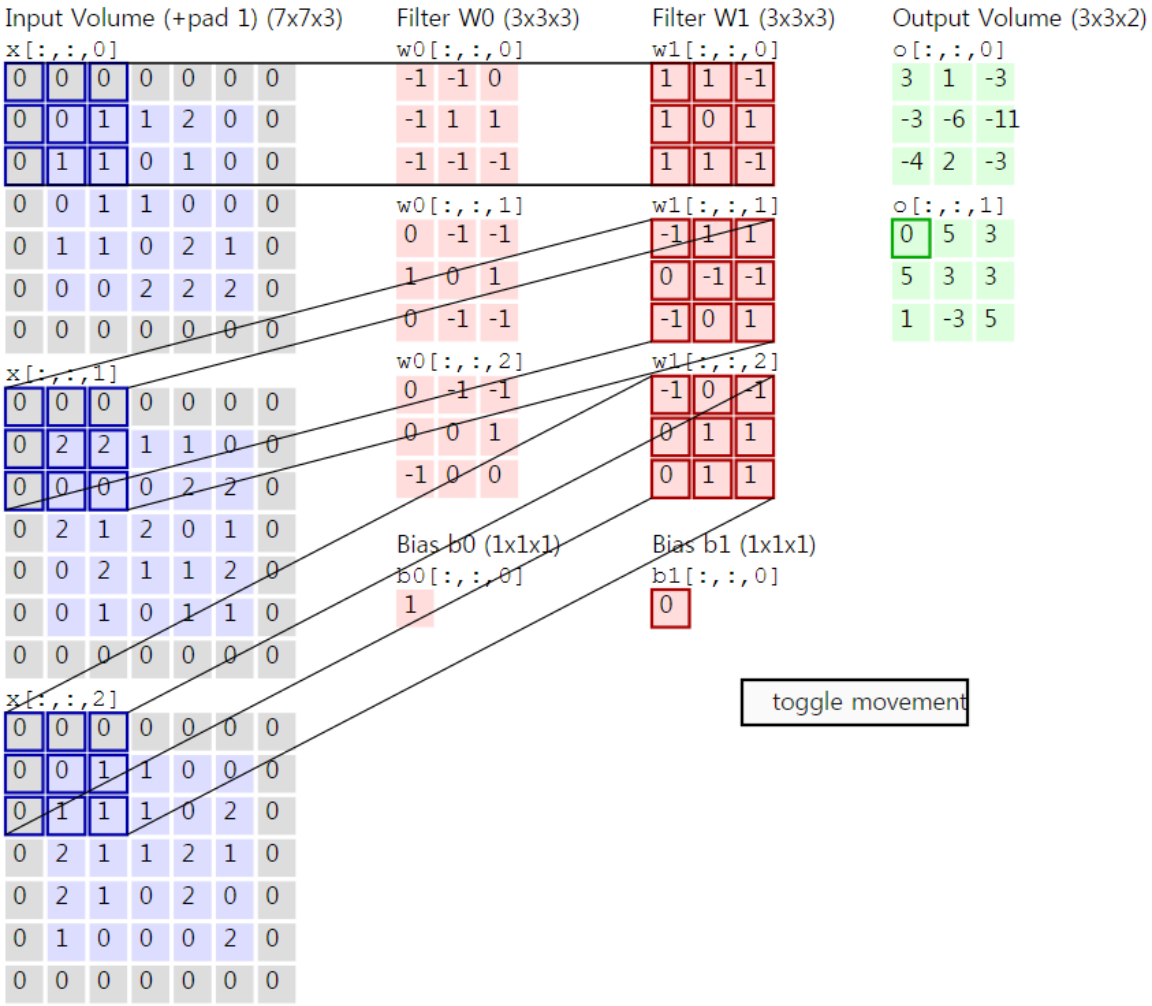
# Convolutional Neural Networks

- Convolutional Layer



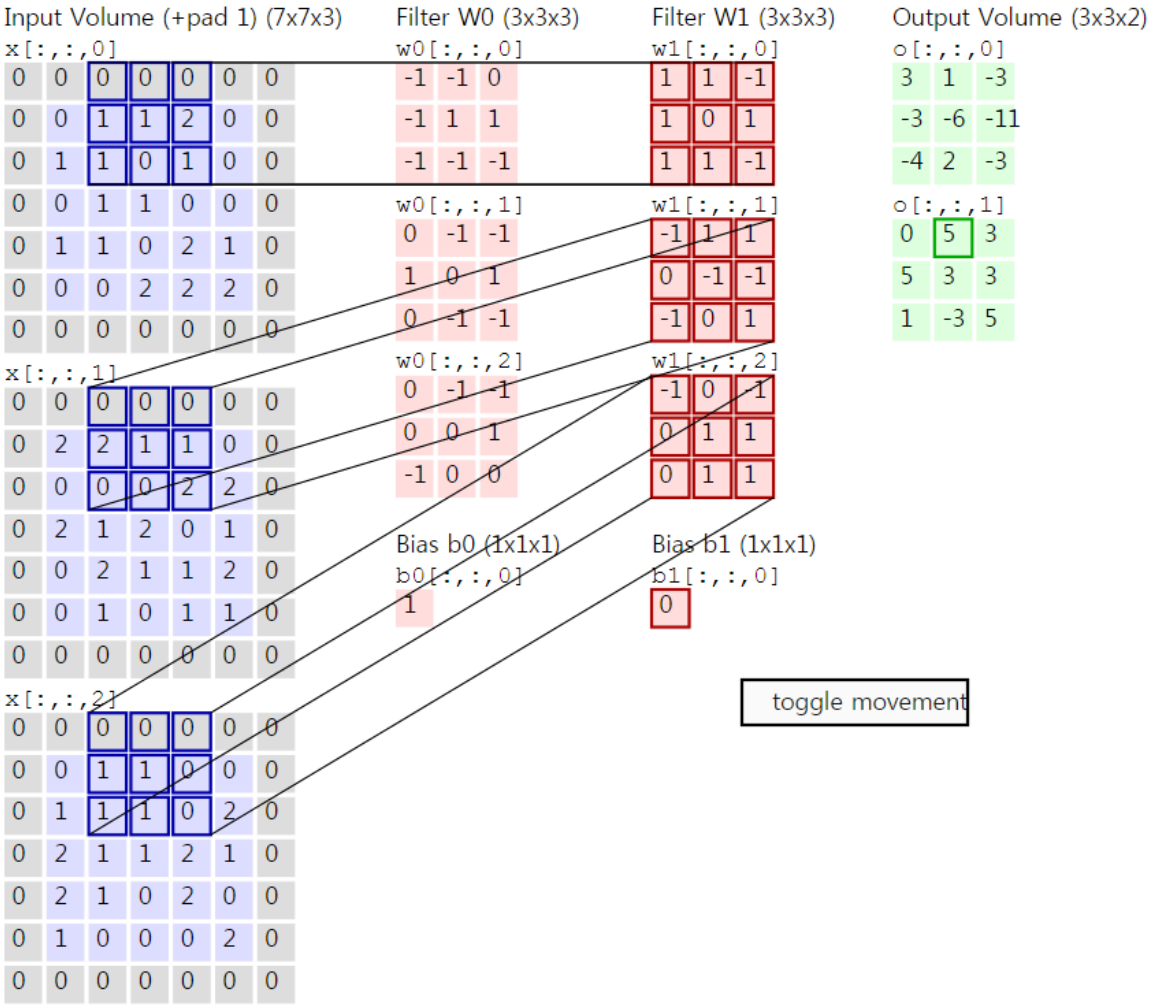
# Convolutional Neural Networks

- Convolutional Layer



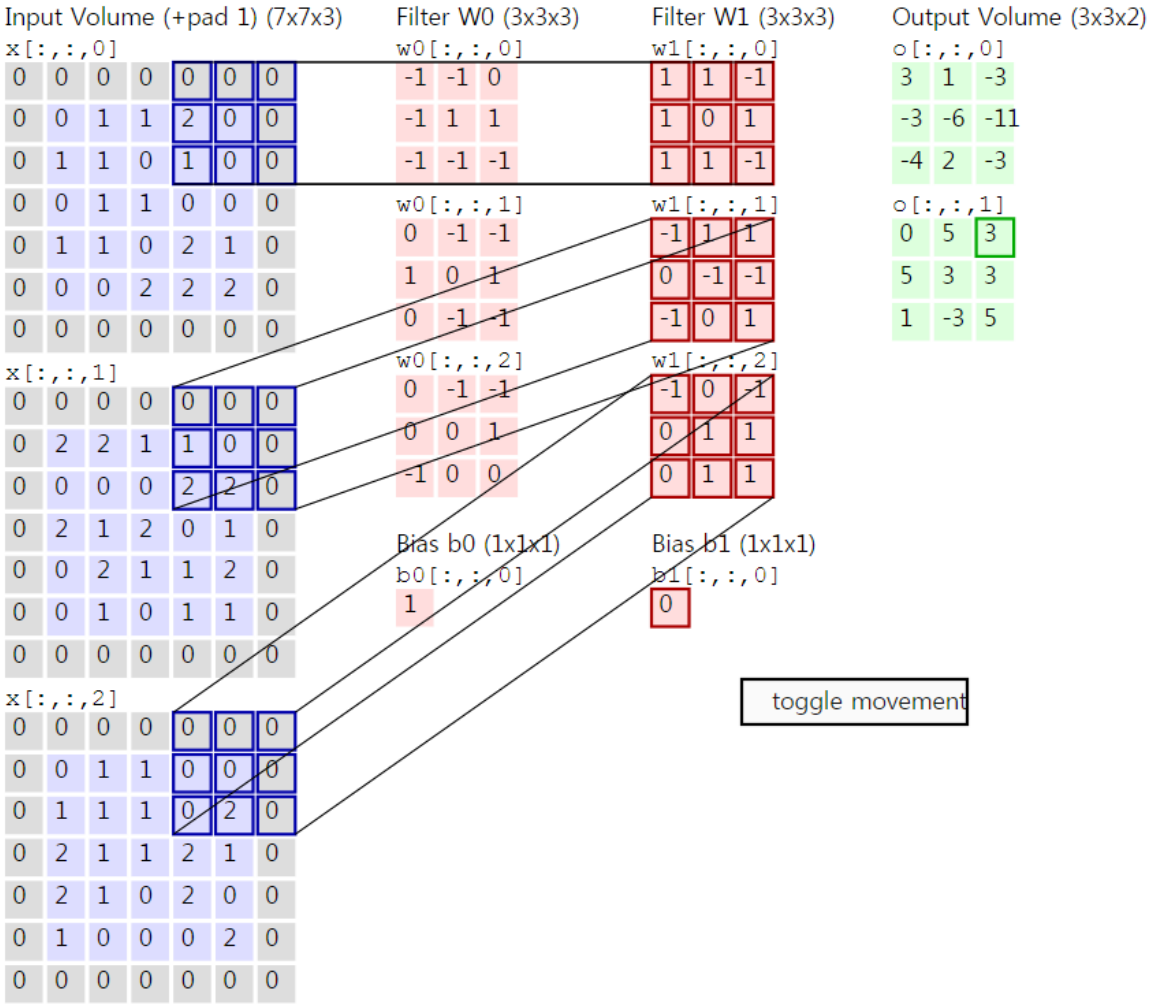
# Convolutional Neural Networks

- Convolutional Layer

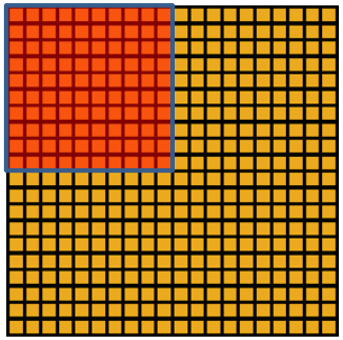


# Convolutional Neural Networks

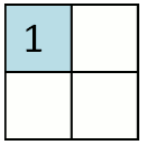
- Convolutional Layer



# Max-Pooling Layer



Convolved feature

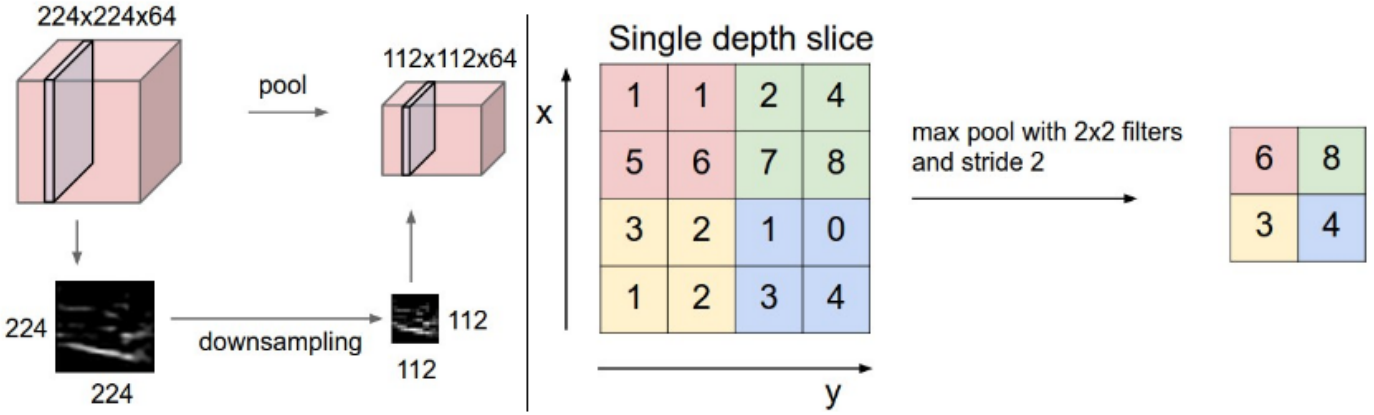



Pooled feature

- 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

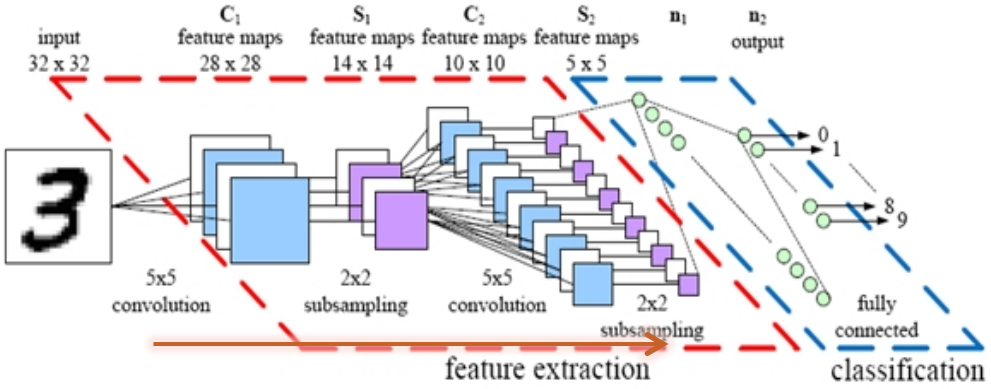
# Convolutional Neural Networks

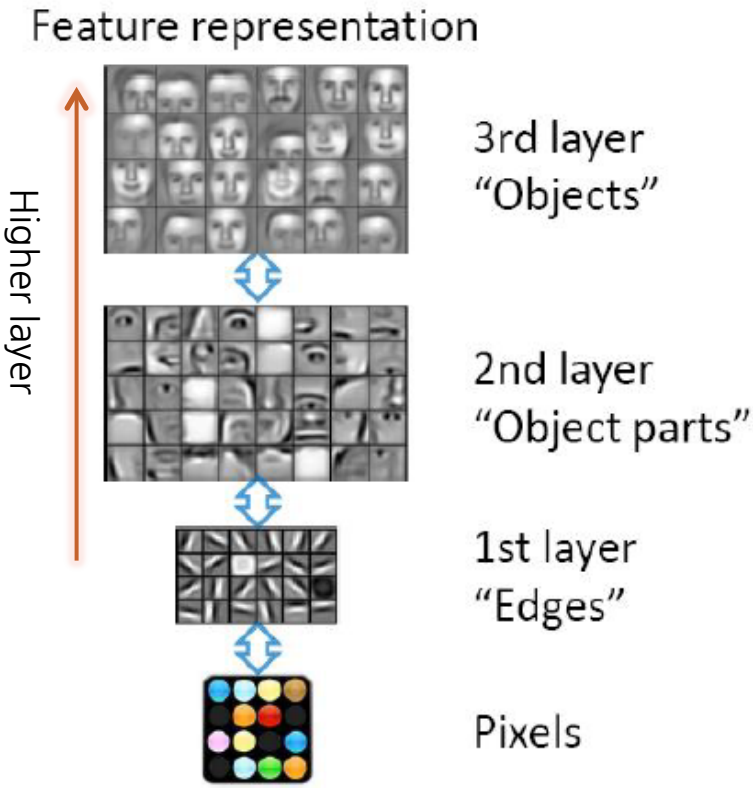
- Pooling layer
  - In-between successive Conv layers (commonly)
  - Reduce spatial size
    - Reduce amount of parameters
    - Control overfitting
  - **Max pooling**, average pooling, L2-norm pooling …




Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

# Remarks

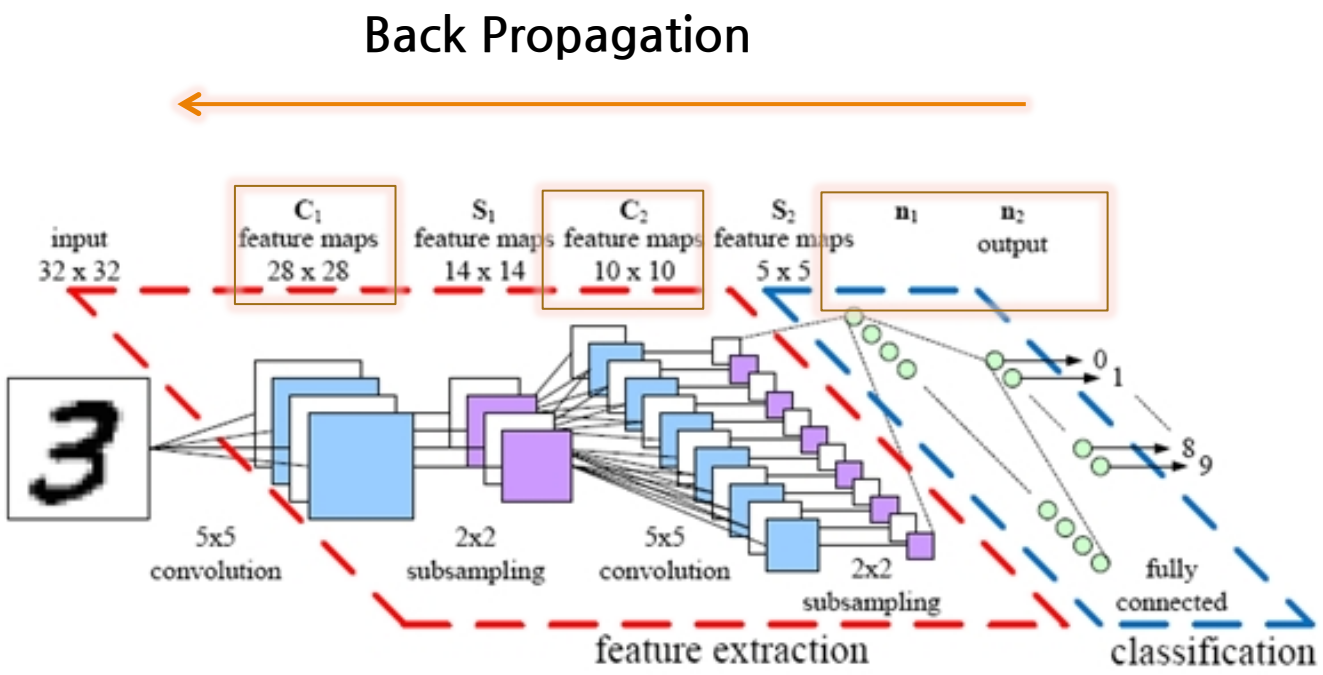



- Higher layer catches more specific, abstract patterns
- Lower layer catches more general patterns

# 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)

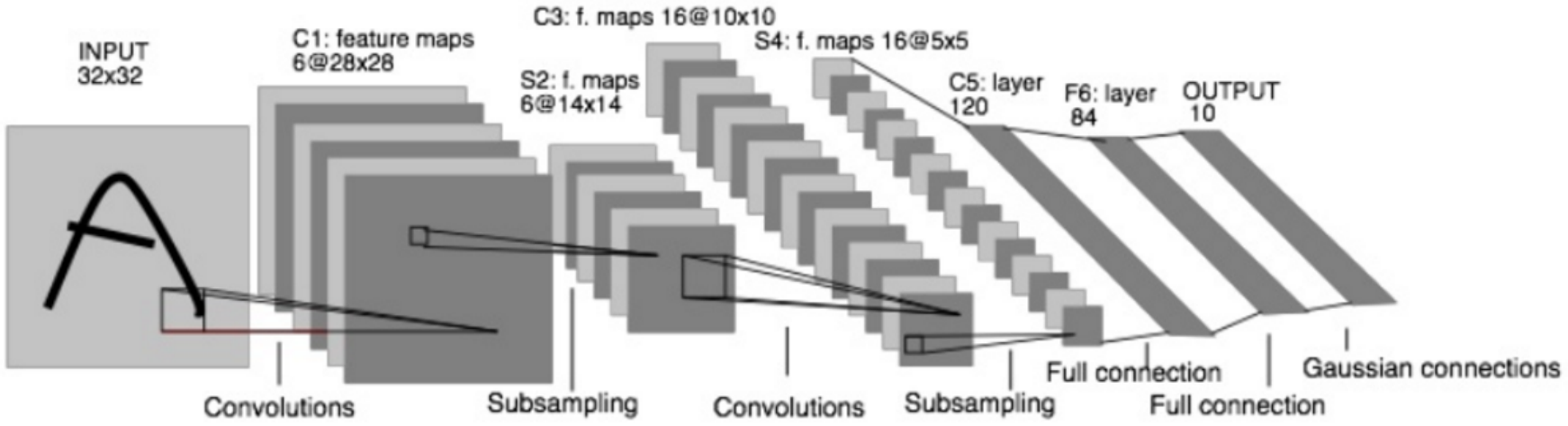
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional Neural Networks**
7. Adobe: **Deep Convolutional Neural Networks**
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!!

From Kyunghyun Cho's dnn tutorial

# Applications (Image Classification) (2/4)

- 1989, CNN for hand digit recognition, Yann LeCun

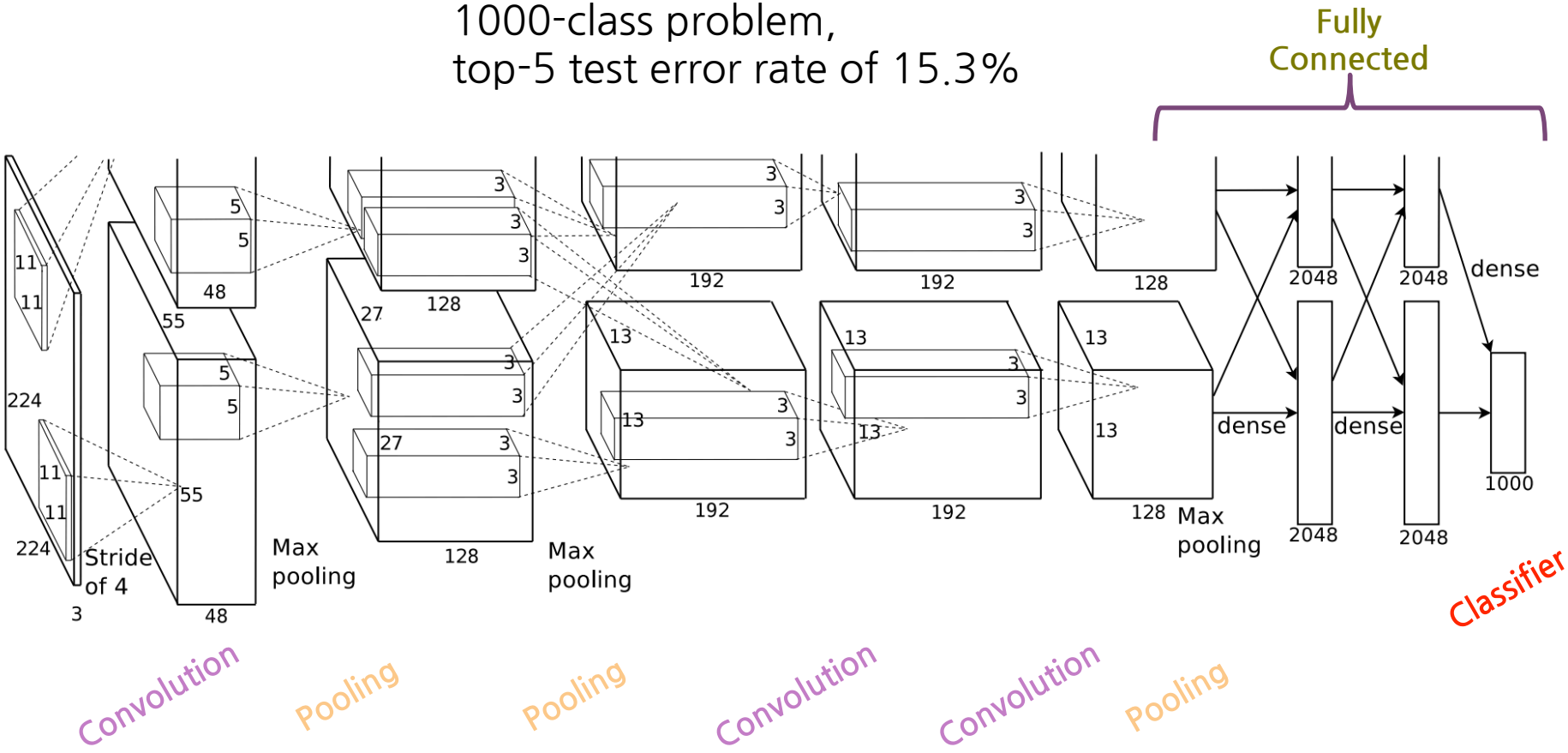



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ Yann LeCun; LeNet ]

# 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

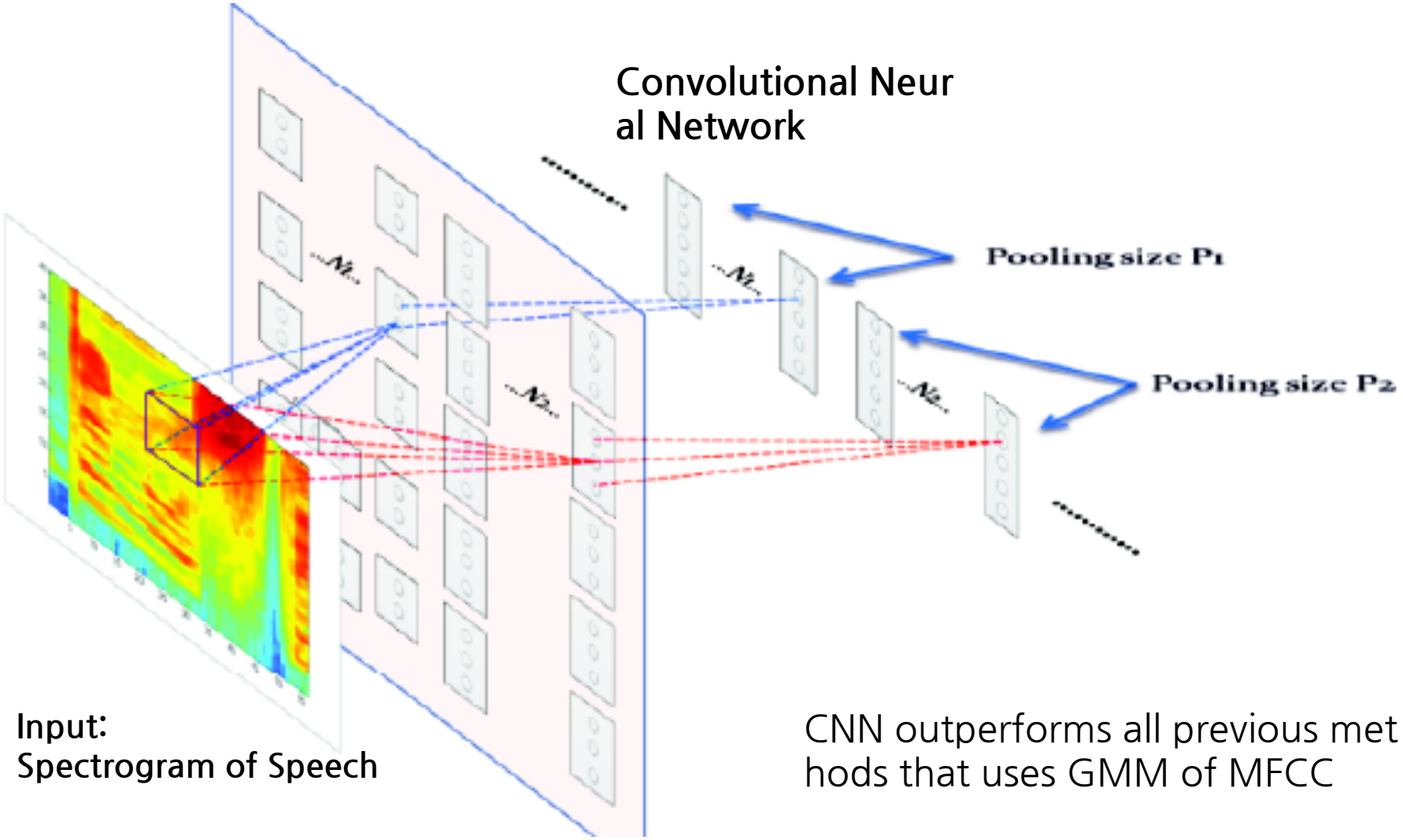
**Example: VGG**

19 layers
3x3 convolution
pad 1
stride 1



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

# Application (Speech Recognition)




**Input:**
**Spectrogram of Speech**

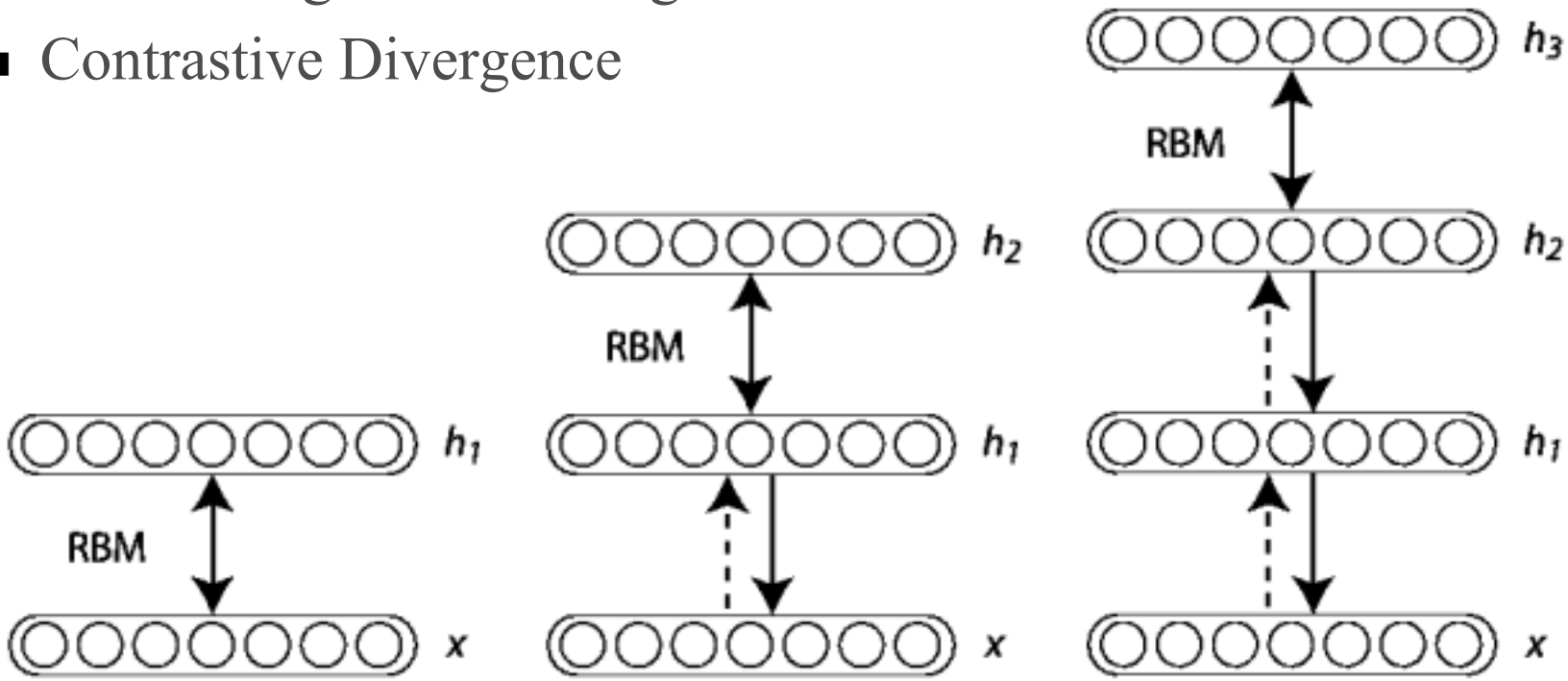
CNN outperforms all previous met
hods that uses GMM of MFCC

# 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

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

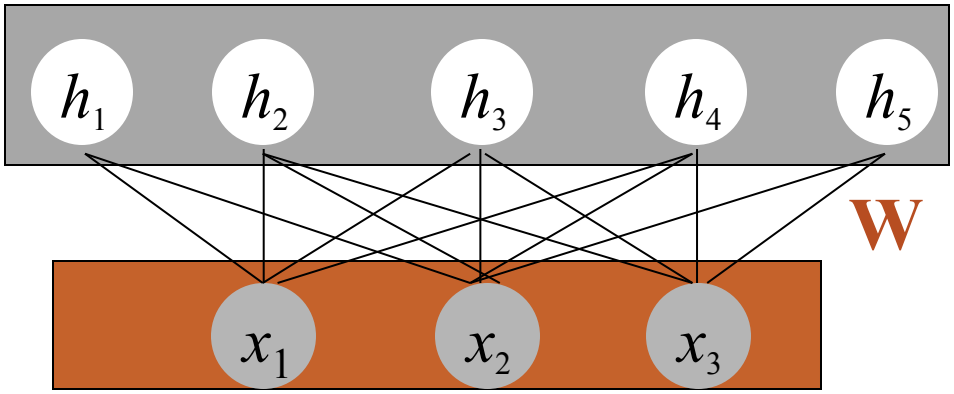
Marginal (x) Probability, or Likelihood

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

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

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

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

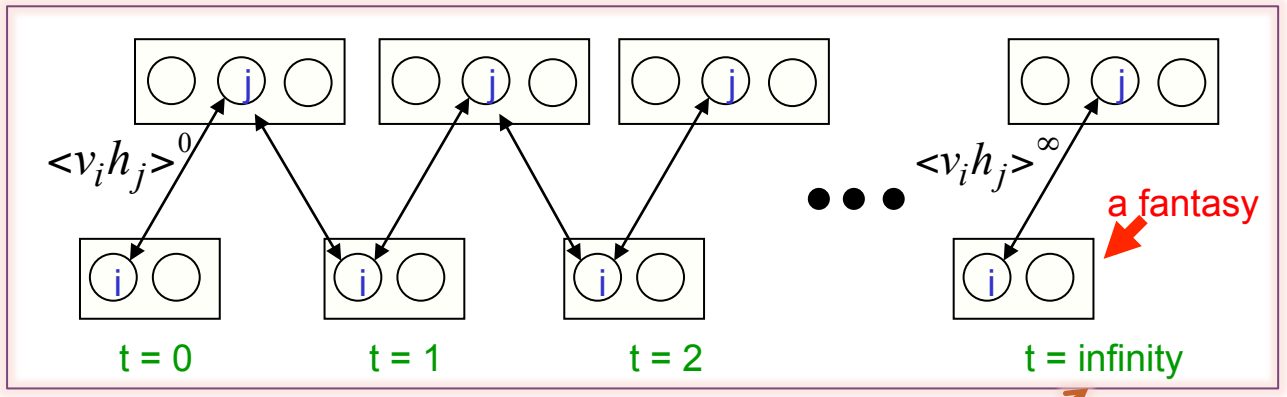
$$P(\mathbf{x}\,|\,\mathbf{h}) = \prod_j P(x_j\,|\,\mathbf{h})$$

- Conditional Probability is the same as Neural Network

# Unsupervised Learning of RBM

- Maximum Likelihood
  - Use Gradient Descent

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




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

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

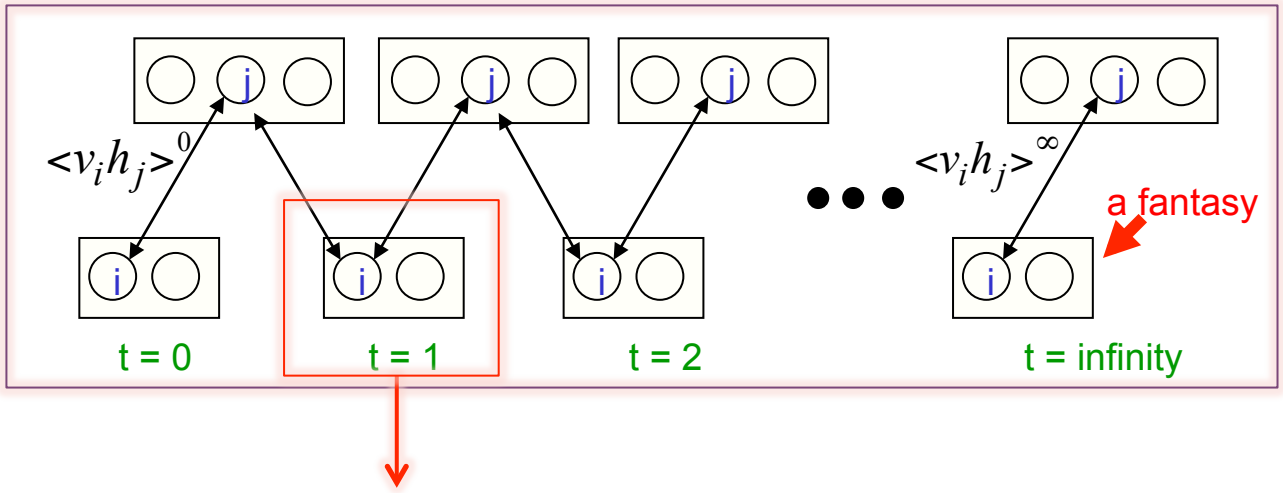
$$\approx \left\langle x_i h_j \right\rangle_{1} - \left\langle x_i h_j \right\rangle_{0}$$

Distribution of Dataset

Distribution of Model

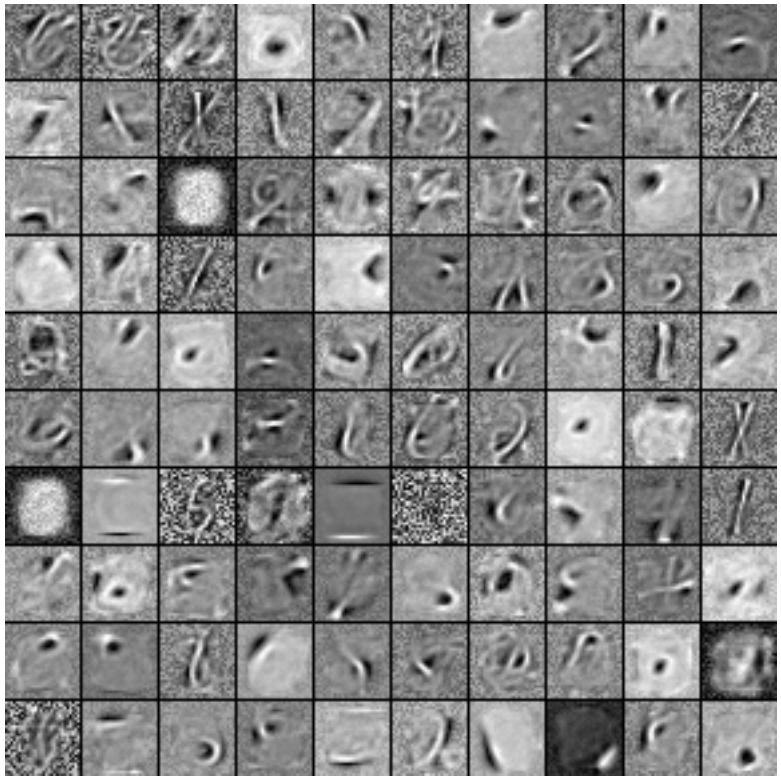
# Contrastive Divergence (CD) Learning of RBM parameters

- 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

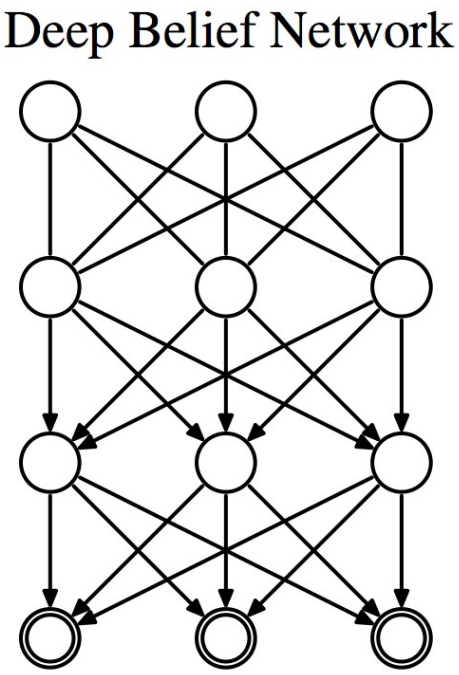


**RBM trained on MNIST h and-written digit data:**

Each cell shows the pattern e ach hidden node encodes

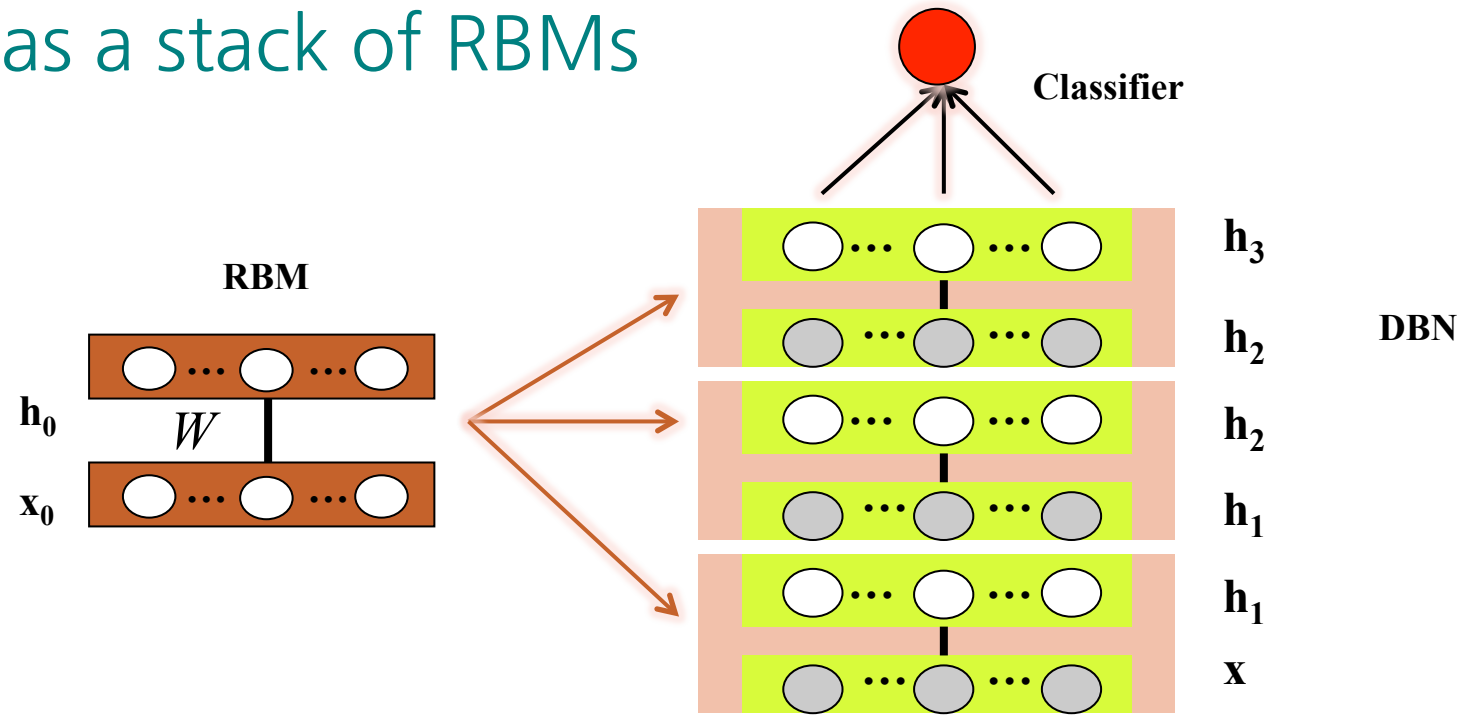
# Deep Belief Network (DBN)

- 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

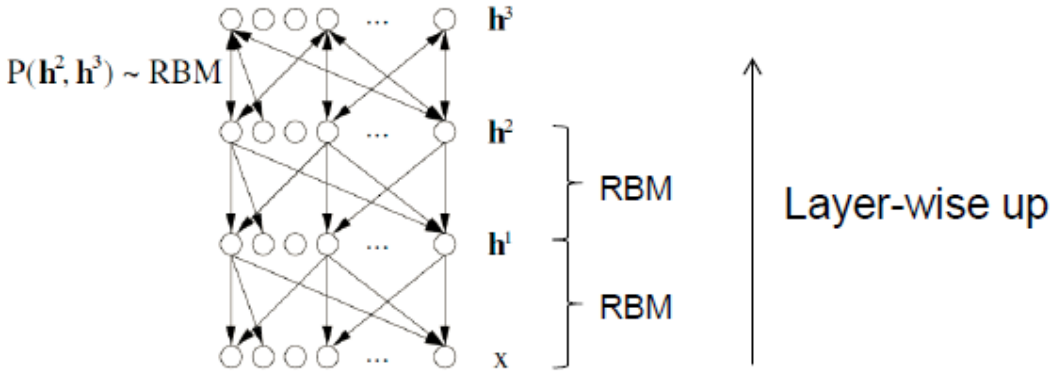



1. Regard each layer as RBM
2. Layer-wise Pre-train each RBM in Unsupervised way
3. Attach the classifier and Fine-tune the whole Network in Supervised way

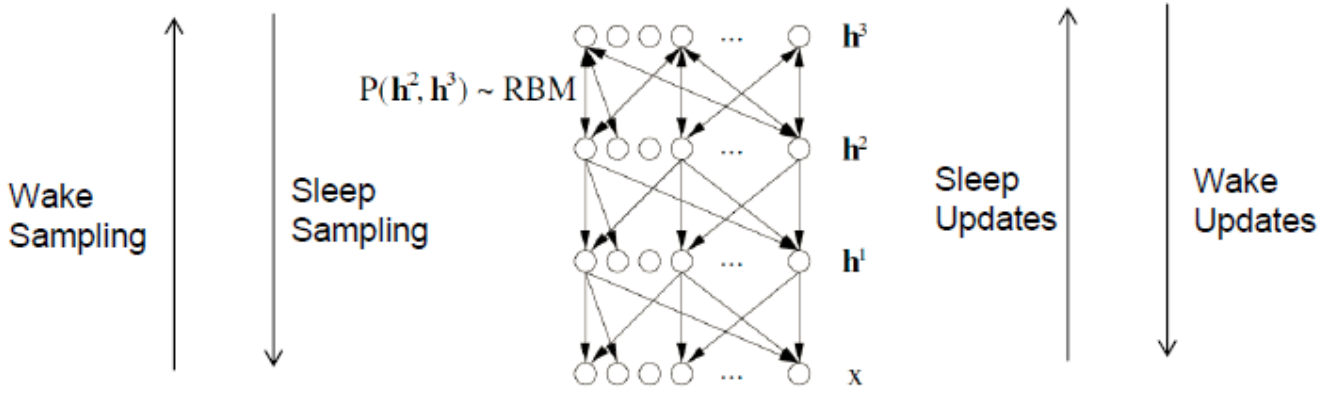
# Viewing Learning as Wake-Sleep Algorithm

## 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:

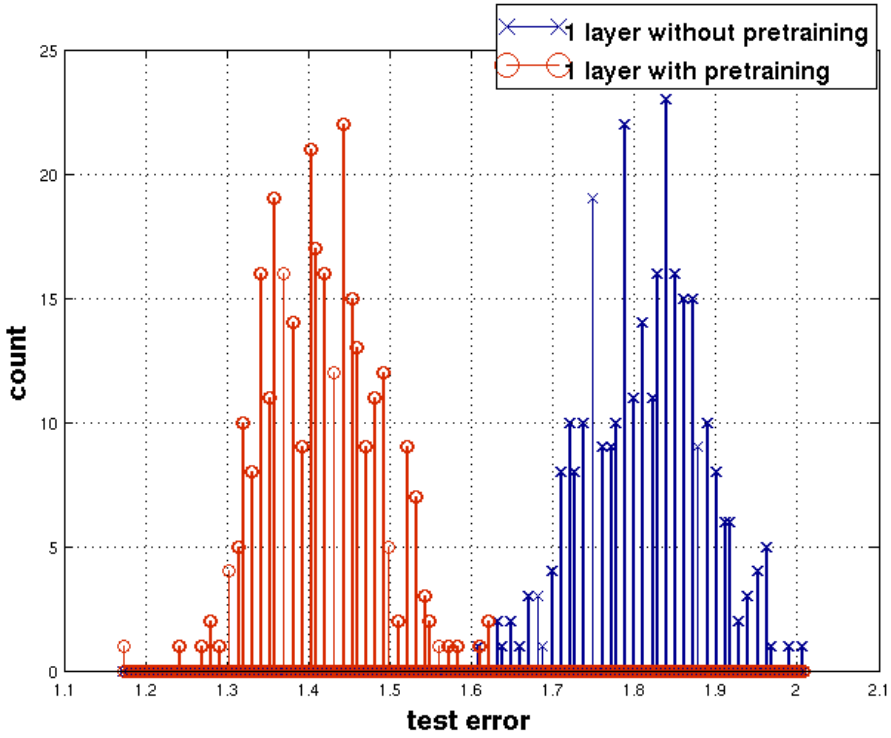


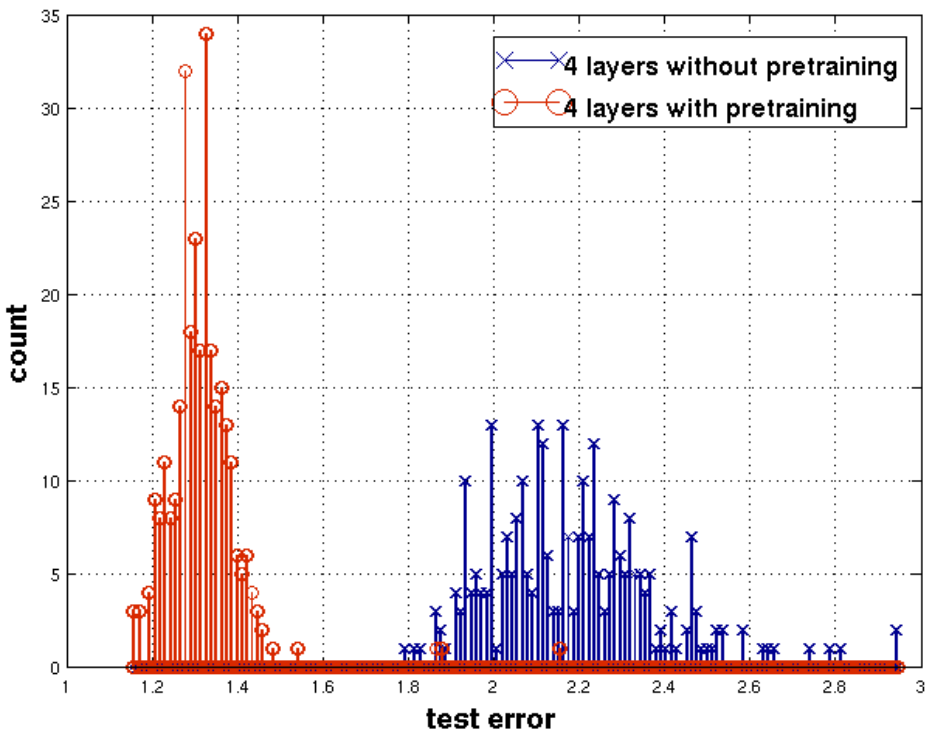

- Joint Wake-Sleep algorithm: Samplings+ Updates



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

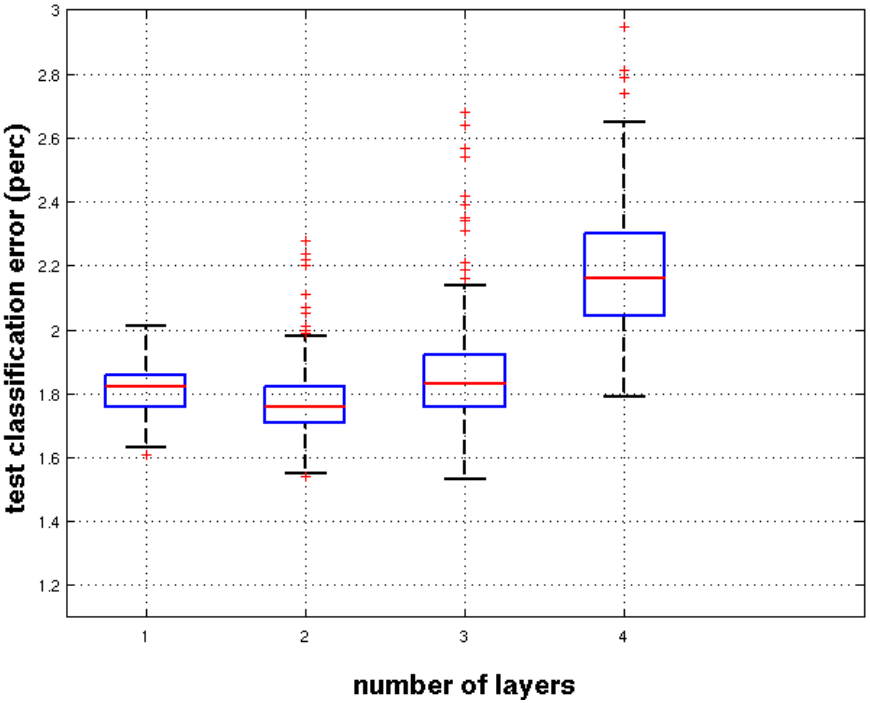
Erhan et. al. AISTATS'2009



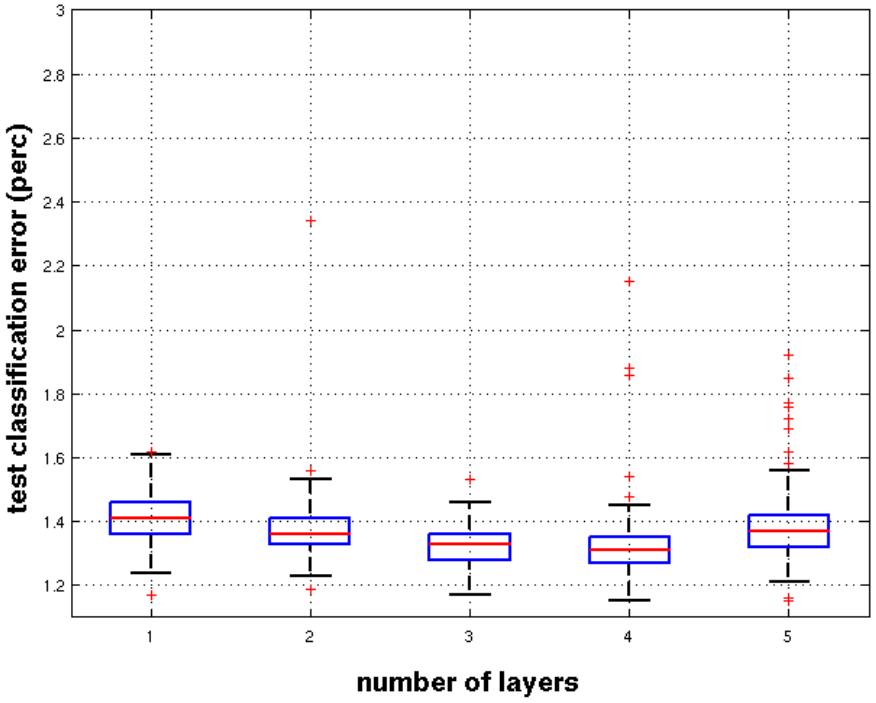



# Effect of Unsupervised Pre-Training in DBN (2/2)

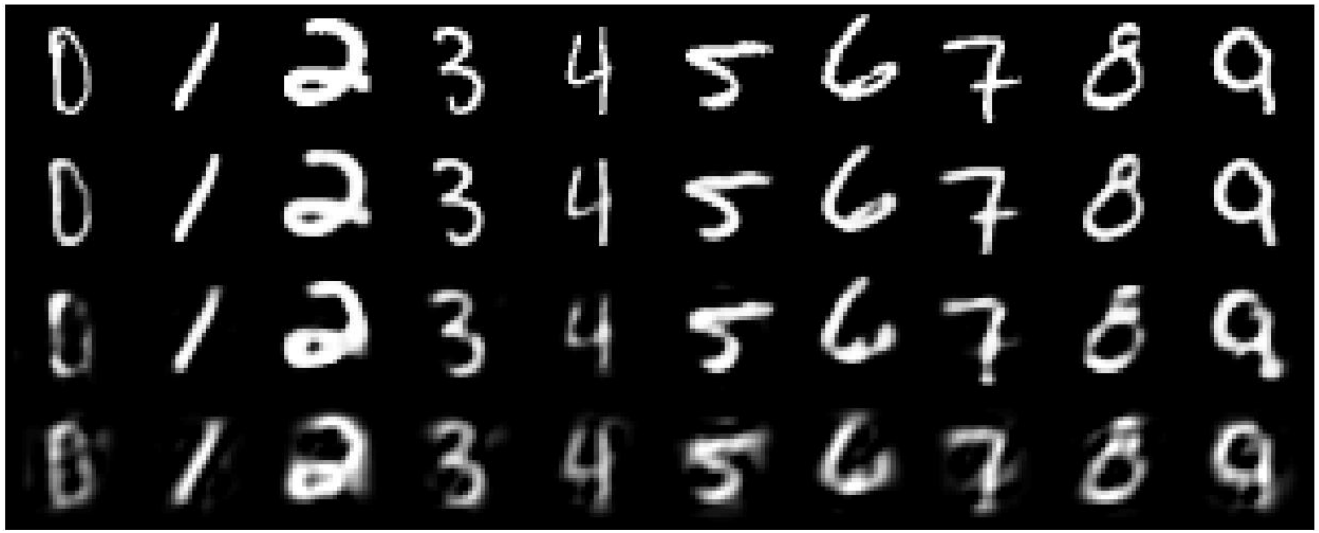
without pre-training

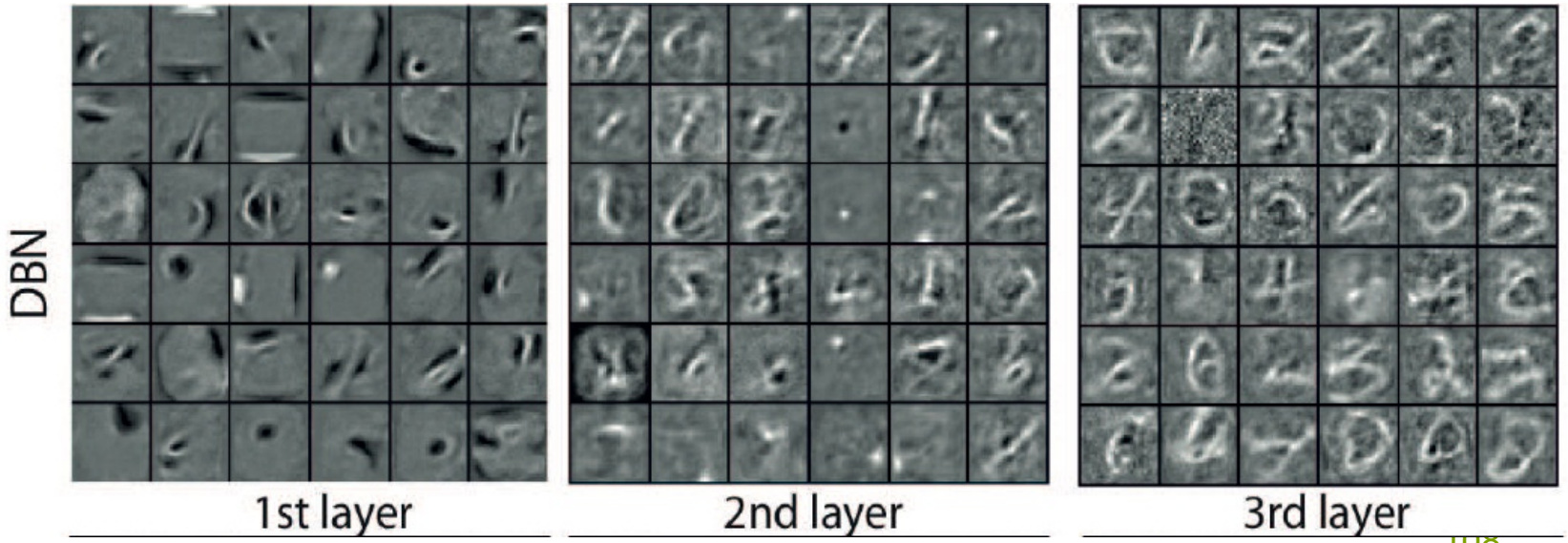



with pre-training



# Internal Representation of DBN





# Representation of Higher Layers

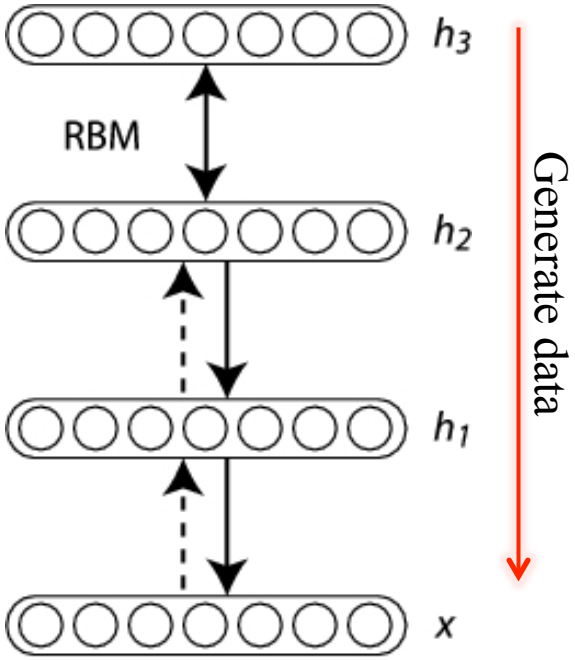
- Higher layers have more abstract representations
  - Interpolating between different images is not desirable in lower layers, but natural in higher layers

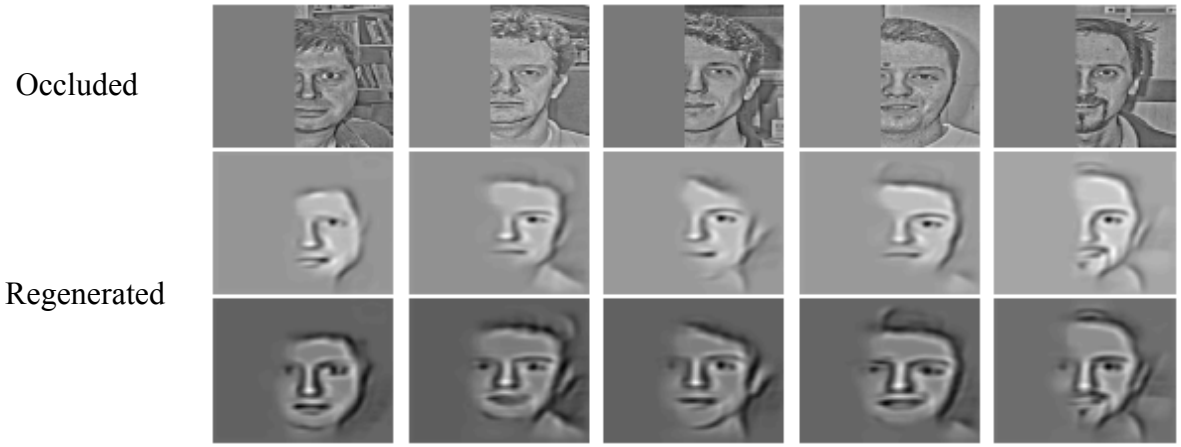
neighbors are close enough for a linear interpolation to be meaningful, while interpolaing with the 200-th nea
shows the failure of interpolation in raw input space but successful interpolation in deeper levels. In (c),
between samples of *different classes*, at different depths (top=raw input, middle=1st layer, bottom=2nd
how in lower levels one has to go through unplausible patterns, whereas in the deeper layers one almost
high-density region of one class to another (of the other class).

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

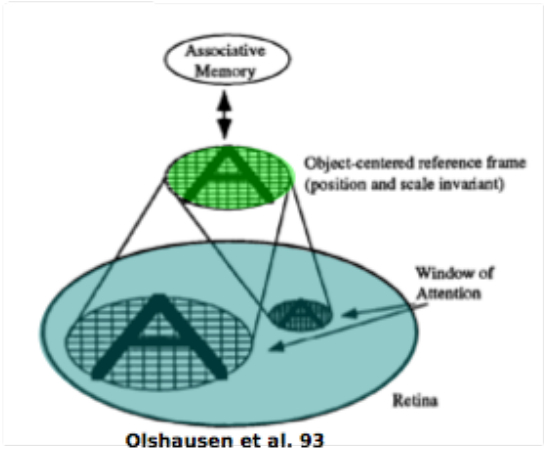


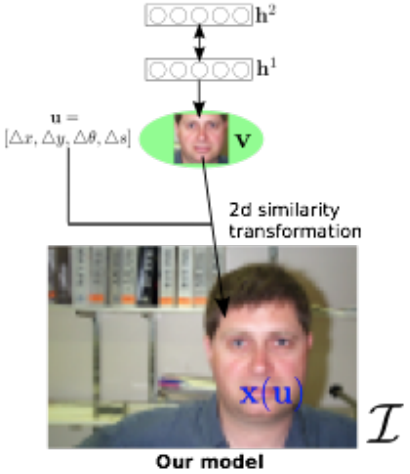




Lee, Ng et al., ICML 2009

# 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



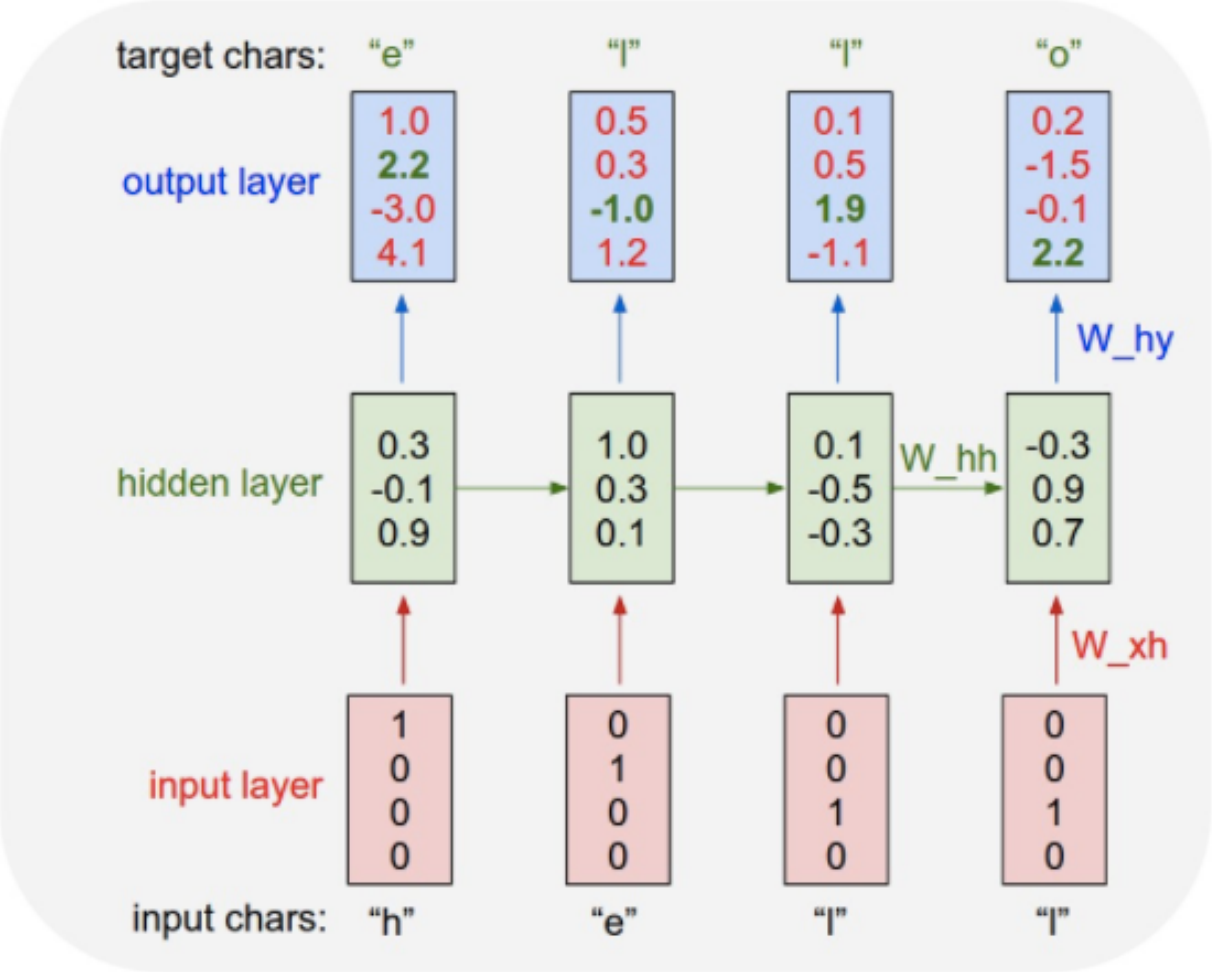

Generate Face patches
Tang, Srivastava, Salakhutdinov, NIPS 2014

# Recurrent Neural Networks

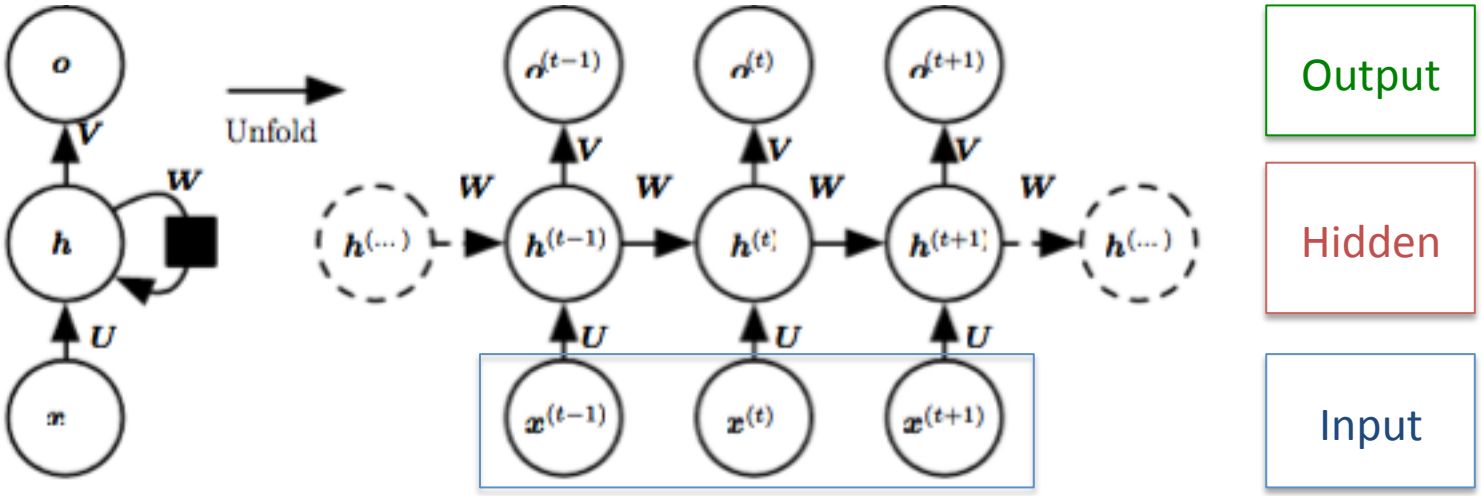
# Introduction

- 시계열 데이터를 모델링 할 수 있는 신경망 기반 모델
- 학습 해야하는 parameter들이 많고 학습이 잘 되지 않아 어려운 모델로 인식되었음
- 최근 언어 데이터를 학습하는 문제에서 뛰어난 성능을 보여 주목을 받고 있음
  - 기계 번역
  - 문서 분류
  - 문장 생성

# An illustrative example



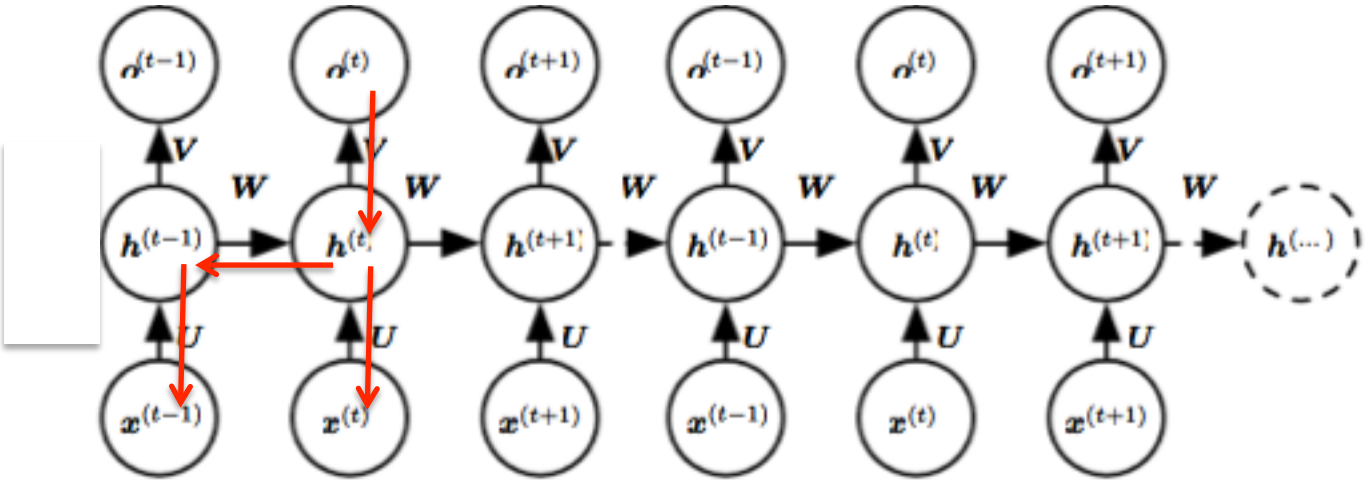
# Architecture




* Parameters: U, W, V

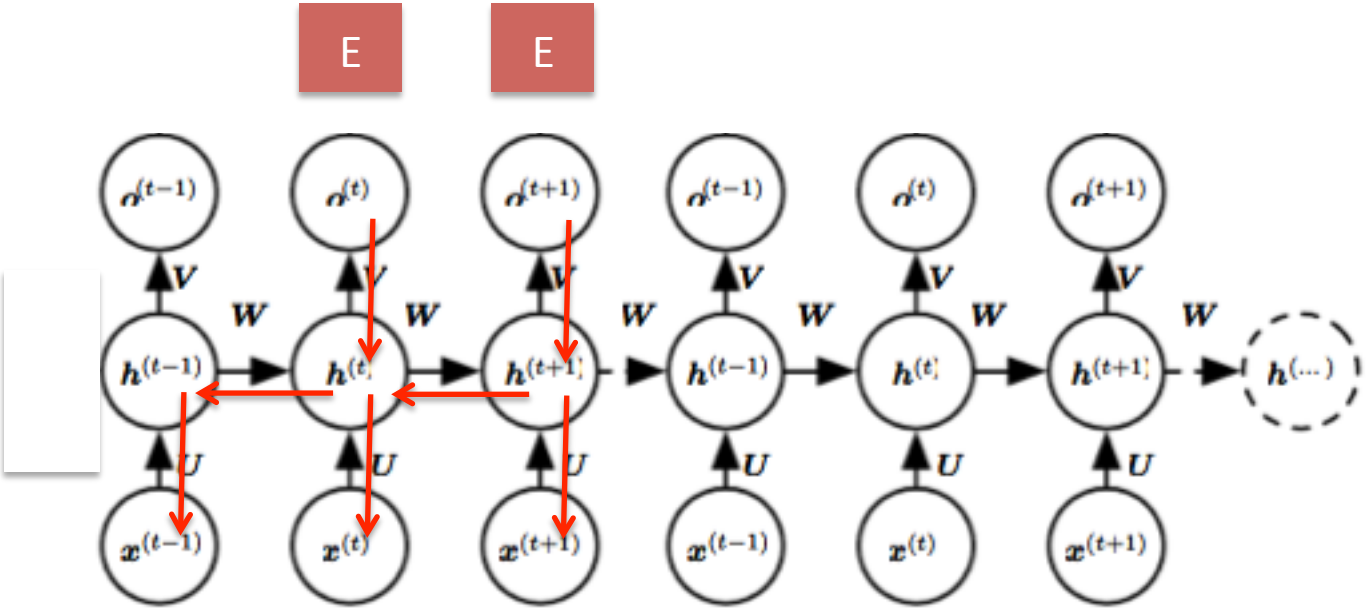
# Process – Next word prediction

E



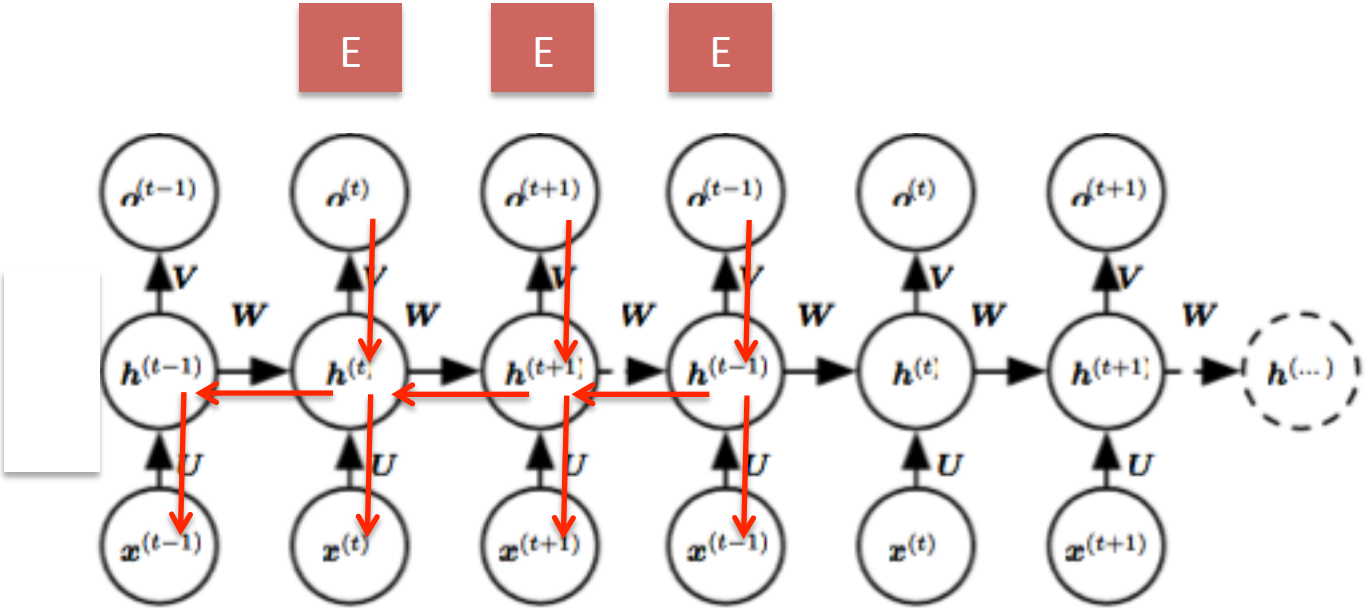

| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

# Process – Next word prediction



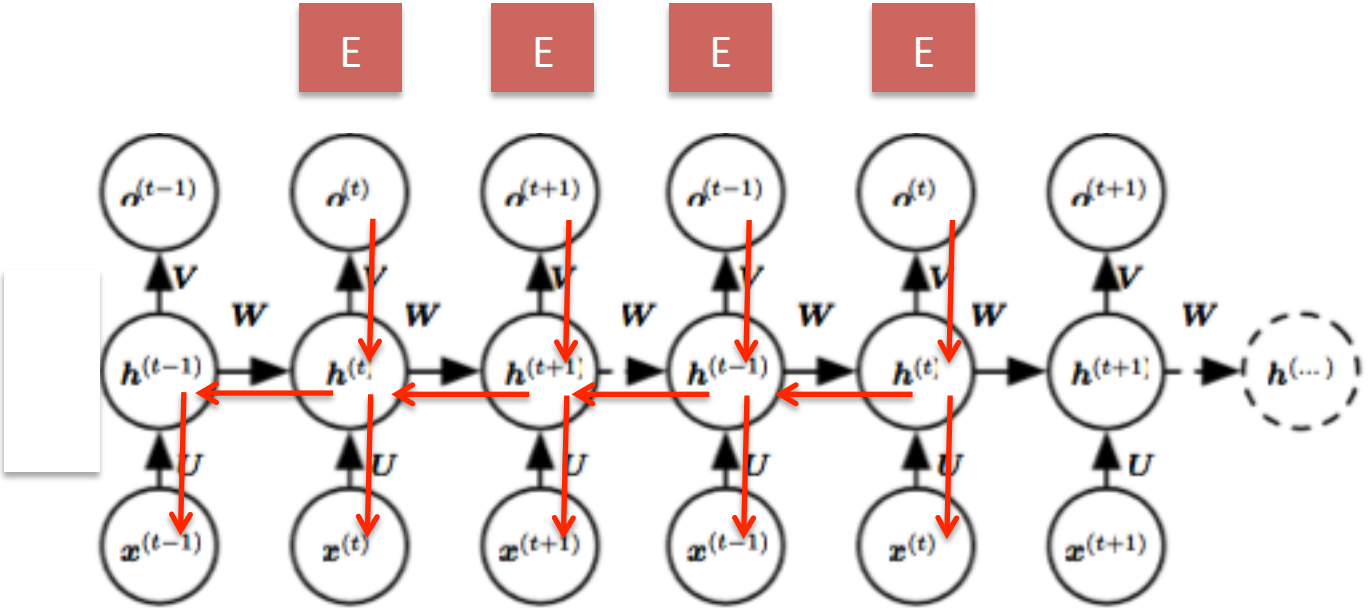

| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

# Process – Next word prediction




| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

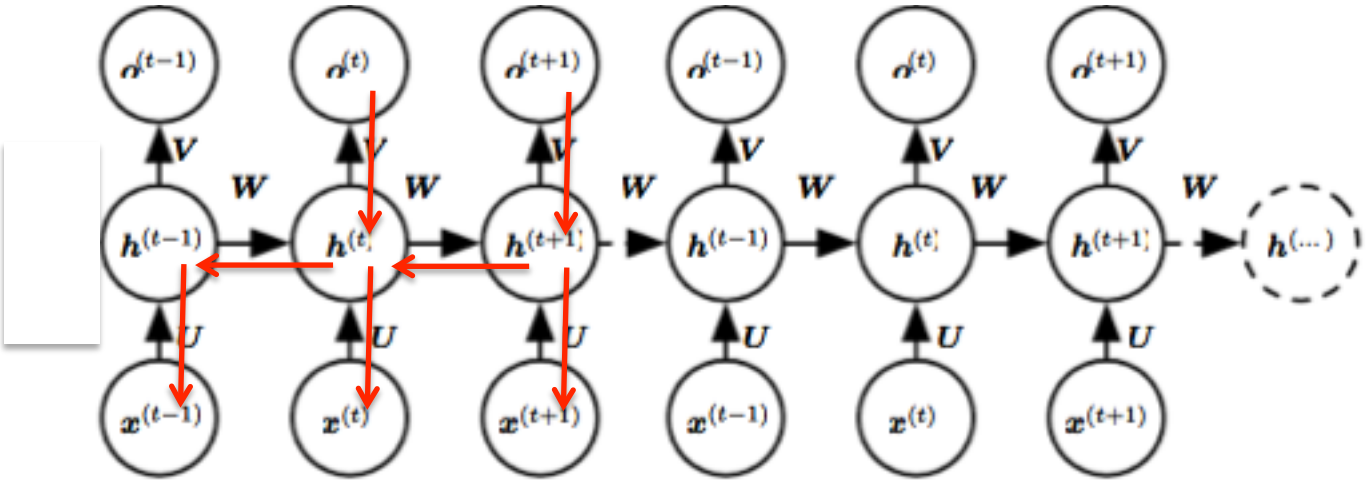
# Process – Next word prediction




| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

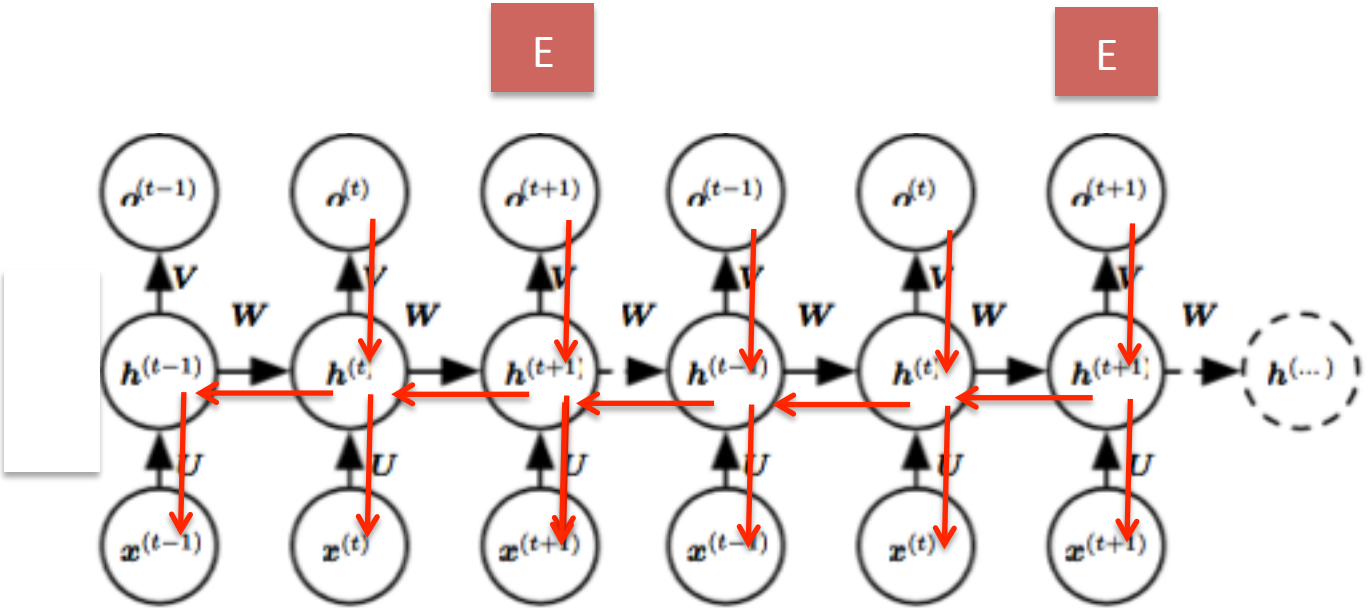
# Process – Sentence classification

E



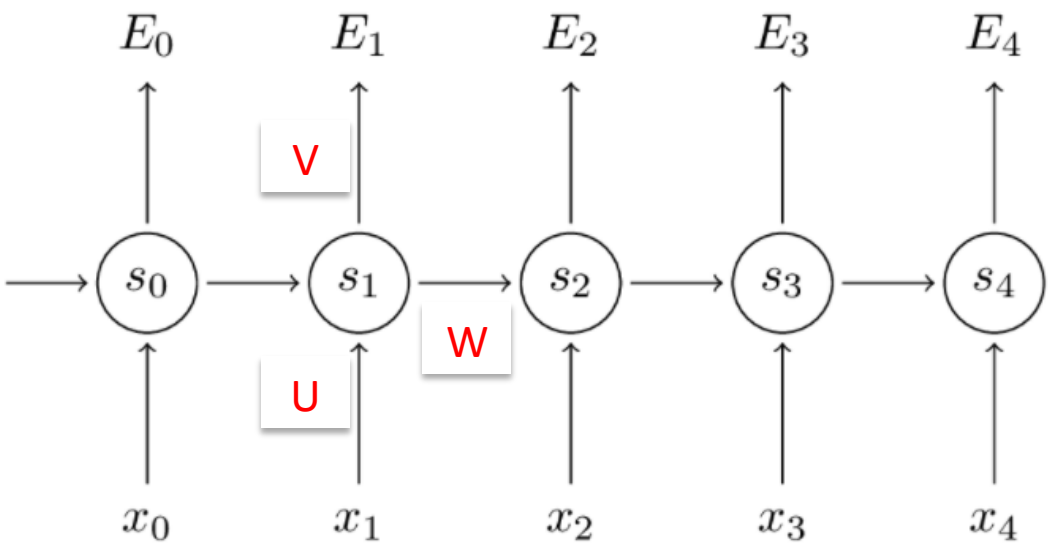

| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

# Process – Sentence classification




| Hello | World | ! | It's | an | apple |
|---|---|---|---|---|---|
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |

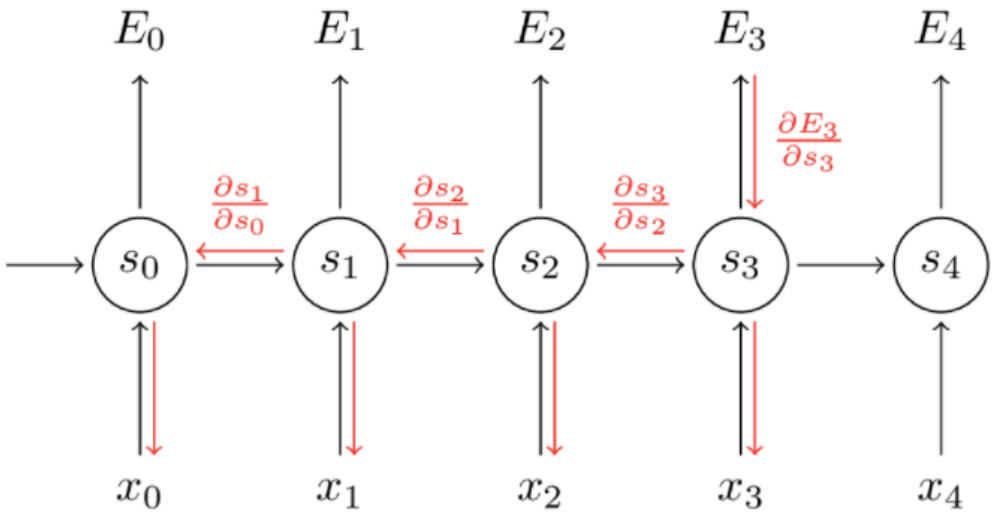
# Model




$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = softmax(Vs_t)$$

$$E(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$
$$E(y, \hat{y}) = -\sum_t E_t(y_t, \hat{y}_t)$$
$$= -\sum_t -y_t \log \hat{y}_t$$

# Learning




$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V}$$

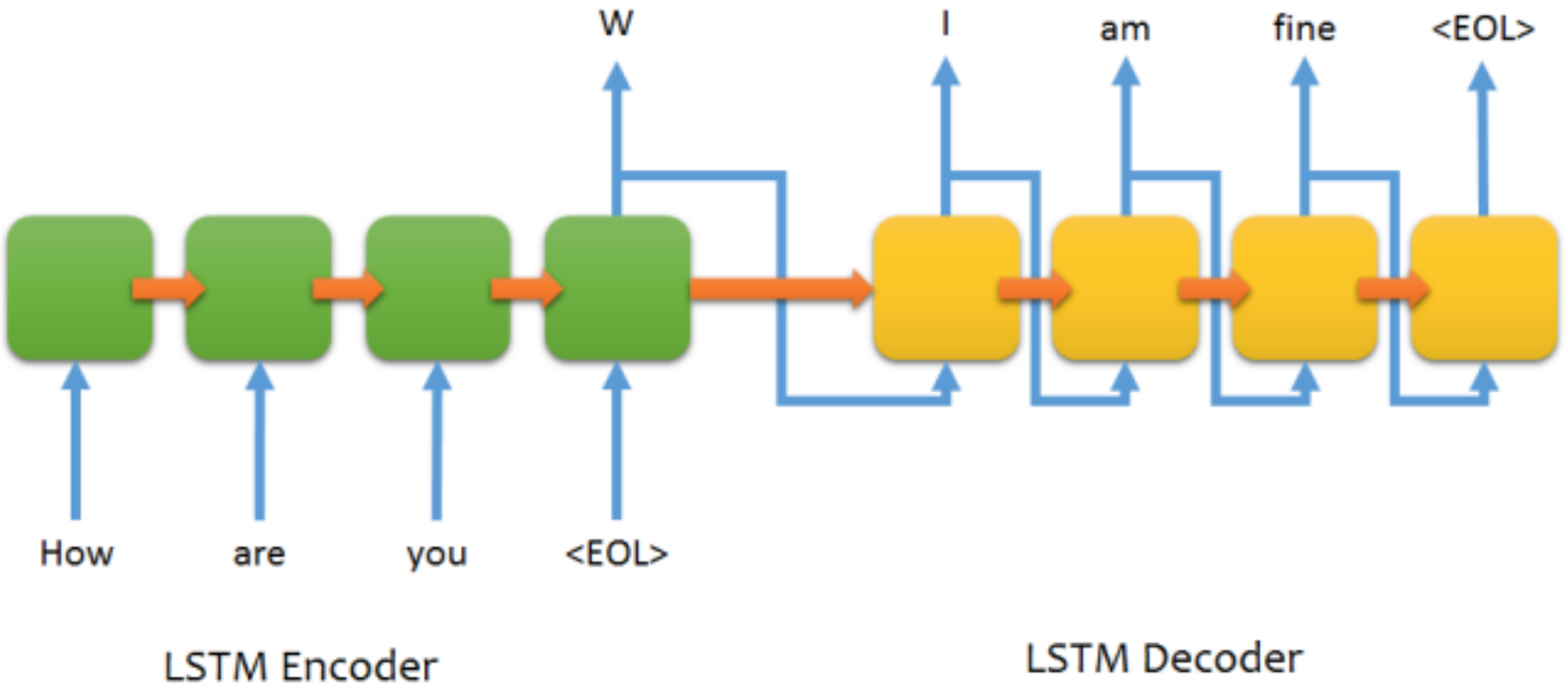
$$= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V}$$

$$= (\hat{y}_3 - y_3) \otimes s_3$$

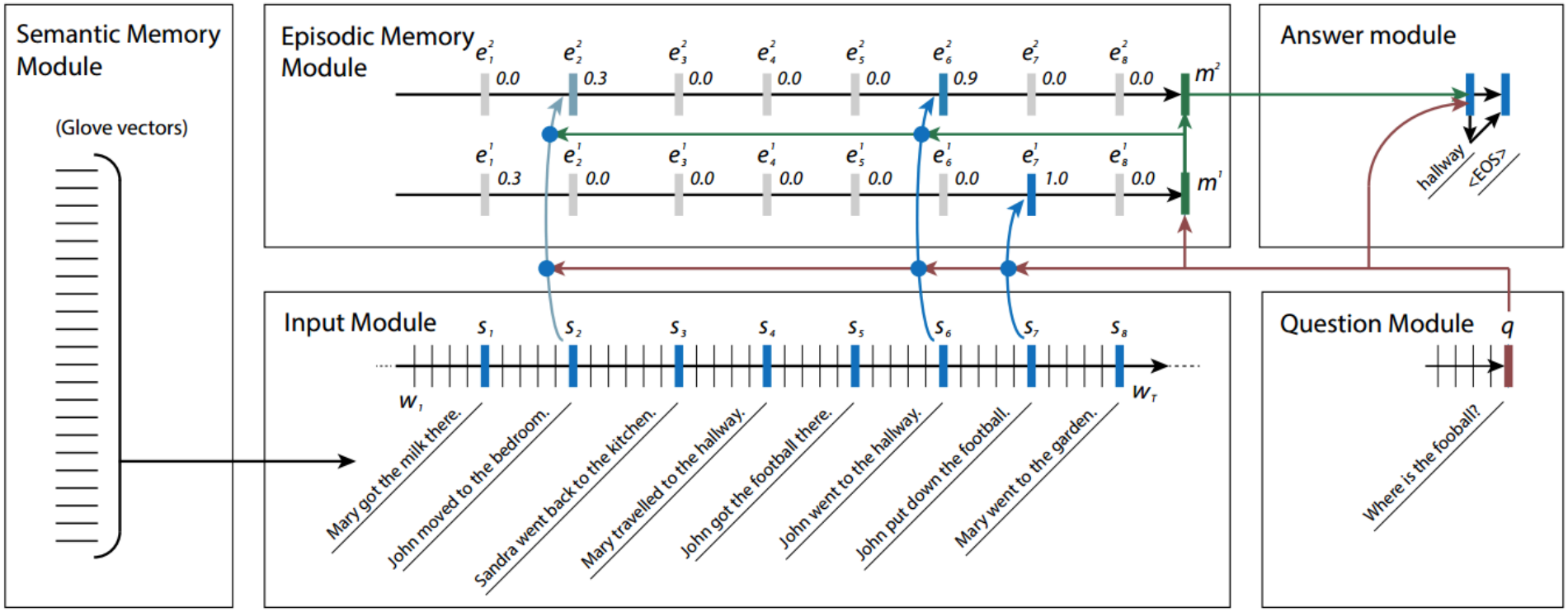
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

# 기계 번역



# 질의 응답


Semantic Memory Module
(Glove vectors)
Episodic Memory Module
Input Module
Answer module
Question Module
Mary got the milk there.
John moved to the bedroom.
Sandra went back to the kitchen.
Mary travelled to the hallway.
John got the football there.
John went to the hallway.
John put down the football.
Mary went to the garden.
Where is the fooball?
hallway
<EOS>

# IMPLEMENTATION

## Subject 1

Hansel and Gretel are lost in the woods.
Where oh where is the way home?
They are lost in the woods, lost in the woods.
Help them find their way home.
Hansel and Gretel are lost in the woods.
Where oh where is the way home?
They are lost in the woods, lost in the woods.
Help them find their way home.
There is a poor woodcutter.
He has a son and a daughter.
The boy is Hansel, and the girl is Gretel.
One day, their father marries an evil stepmother.
Their stepmother says to their father, "We are starving."
"So I will leave the children in the forest."
Hansel hears that.

## Subject 2

Snow White is pretty.
Very, very pretty.
She's the prettiest of them all.
The Queen is jealous.
Very, very jealous.
She gives Snow White a red apple.
Snow White becomes sick.
Very, very sick.
She needs a prince to wake her up.
A handsome prince comes.
He comes to save her.
She wakes up and they are happy.
Far away, there is a princess in a big castle.
She is very pretty.
Her name is Snow White.

## Subject 3

I wish I could have human legs.
I wish I could be one of them.
I wish I could tell the truth.
I wish I could marry the prince.
If I can't… if I can't…if I can't.
If I can't I would bless.
my dear prince, and I would say goodbye.
Goodbye my dear prince!
Deep inside the sea, there lived a little mermaid.
She was a princess who loved to see the outside world.
One night, she visited the seaside.
The mermaid went up to the surface.
She was so excited to see a big ship.
"How fantastic!"
In the sky, the sparkling lights were bright.

## Subject 4

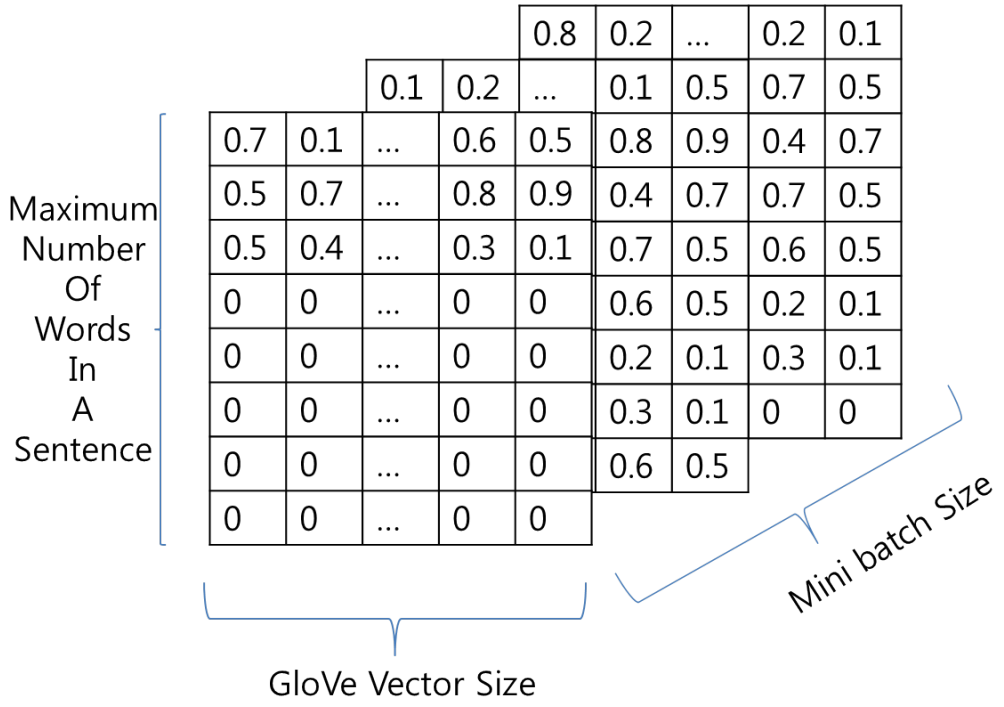
One little fat pig, two little fat pigs, three little fat pigs, let me in.
I will blow your house down, piggy.
Hurry, open up.
One little fat pig, two little fat pigs, three little fat pigs, let me in.
You will eat us, big bad wolf.
We won't let you in.
One little fat pig, two little fat pigs, three little fat pigs, let me in.
So go away you, big bad wolf.
We won't let you in.
There are three little pigs.
They are Biggie, Middie and Junior.
Each pig builds a house to keep the big bad wolf away.
Uh ah! Uh ah!
Biggie builds a house with straw.
And he says, "The wolf will never come in."

| | | | | | |
|---|---|---|---|---|---|
| Hansel | 0.7 | 0.1 | ... | 0.6 | 0.5 |
| hears | 0.5 | 0.7 | ... | 0.8 | 0.9 |
| that | 0.5 | 0.4 | ... | 0.3 | 0.1 |

| | | | | | |
|---|---|---|---|---|---|
| Hansel | 0.1 | 0.2 | ... | 0.1 | 0.5 |
| and | 0.5 | 0.7 | ... | 0.8 | 0.9 |
| Gretel | 0.1 | 0.2 | ... | 0.4 | 0.7 |
| are | 0.2 | 0.5 | ... | 0.7 | 0.5 |
| lost | 0.7 | 0.1 | ... | 0.6 | 0.5 |
| in | 0.8 | 0.2 | ... | 0.2 | 0.1 |
| the | 0.5 | 0.4 | ... | 0.3 | 0.1 |
| woods | 0.7 | 0.1 | ... | 0.6 | 0.5 |

| | | | | | |
|---|---|---|---|---|---|
| Where | 0.8 | 0.2 | ... | 0.2 | 0.1 |
| oh | 0.2 | 0.5 | ... | 0.7 | 0.5 |
| Where | 0.1 | 0.2 | ... | 0.4 | 0.7 |
| is | 0.2 | 0.5 | ... | 0.7 | 0.5 |
| the | 0.7 | 0.1 | ... | 0.6 | 0.5 |
| way | 0.8 | 0.2 | ... | 0.2 | 0.1 |
| home | 0.5 | 0.4 | ... | 0.3 | 0.1 |

Mini batch



# 딥러닝 실습

# 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

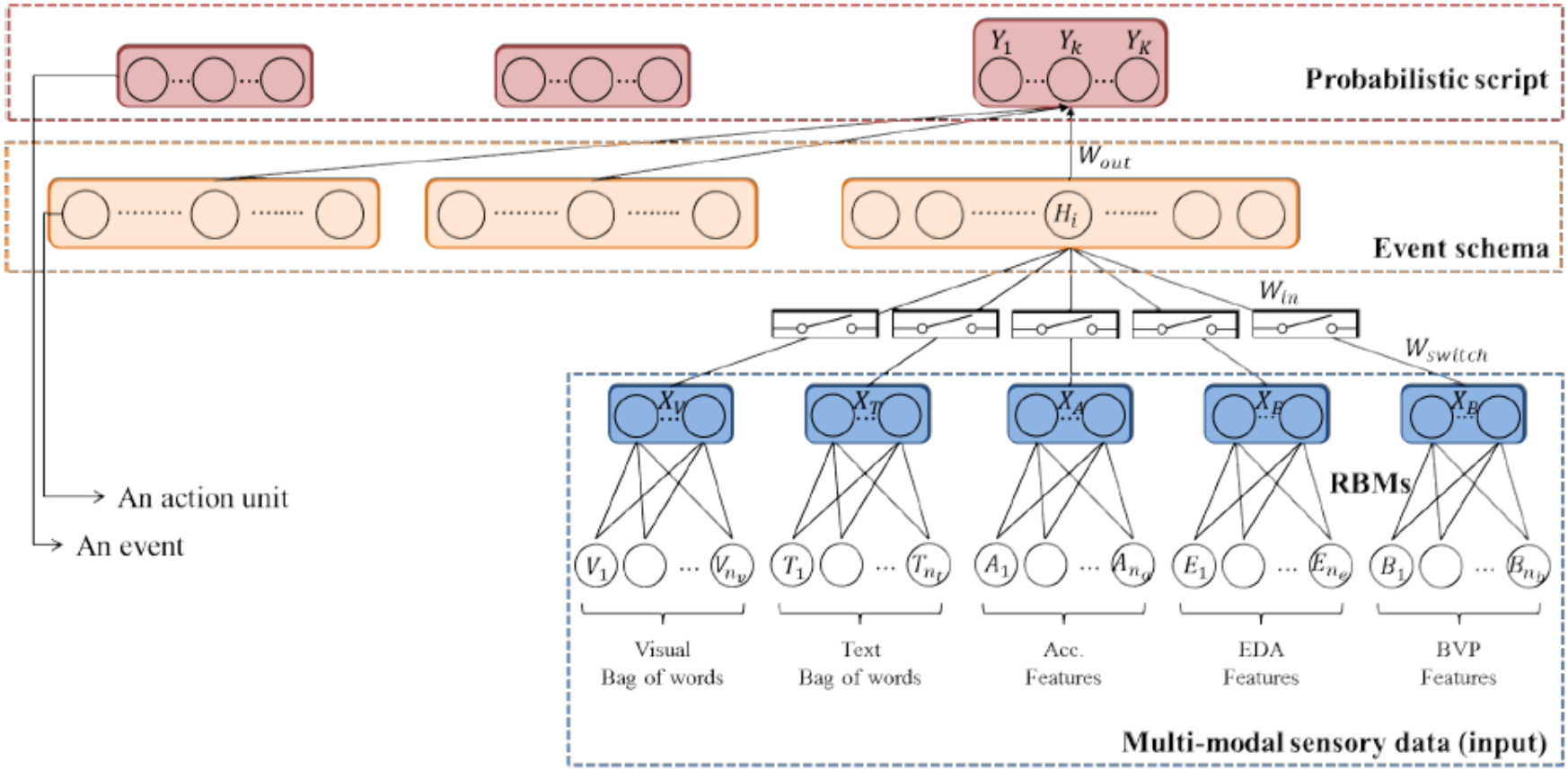
# 다양한 딥러닝 프레임워크

| | 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


Probabilistic script
Event schema
Multi-modal sensory data (input)
RBMs
An action unit
An event
Visual
Bag of words
Text
Bag of words
Acc.
Features
EDA
Features
BVP
Features

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

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

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

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

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

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

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

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

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

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

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

```
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]
```

```
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))
```