Machine Learning Basics

KIAS CAC 병렬컴퓨팅 여름학교

2018. 06.26

김 은 솔(epsilon.kim@kakaobrain.com)

kakaobrain

Contents

- Basic Concepts of Machine learning
- Feasibility of Learning
- Neural Network
  - Basics of Neural Networks
  - Convolutional Neural Networks
  - Recurrent Neural Networks

Learning

- Data
  - $\mathcal{D} = \{\mathrm{x}_i, y_i\}_{i=1}^{N} \sim P$

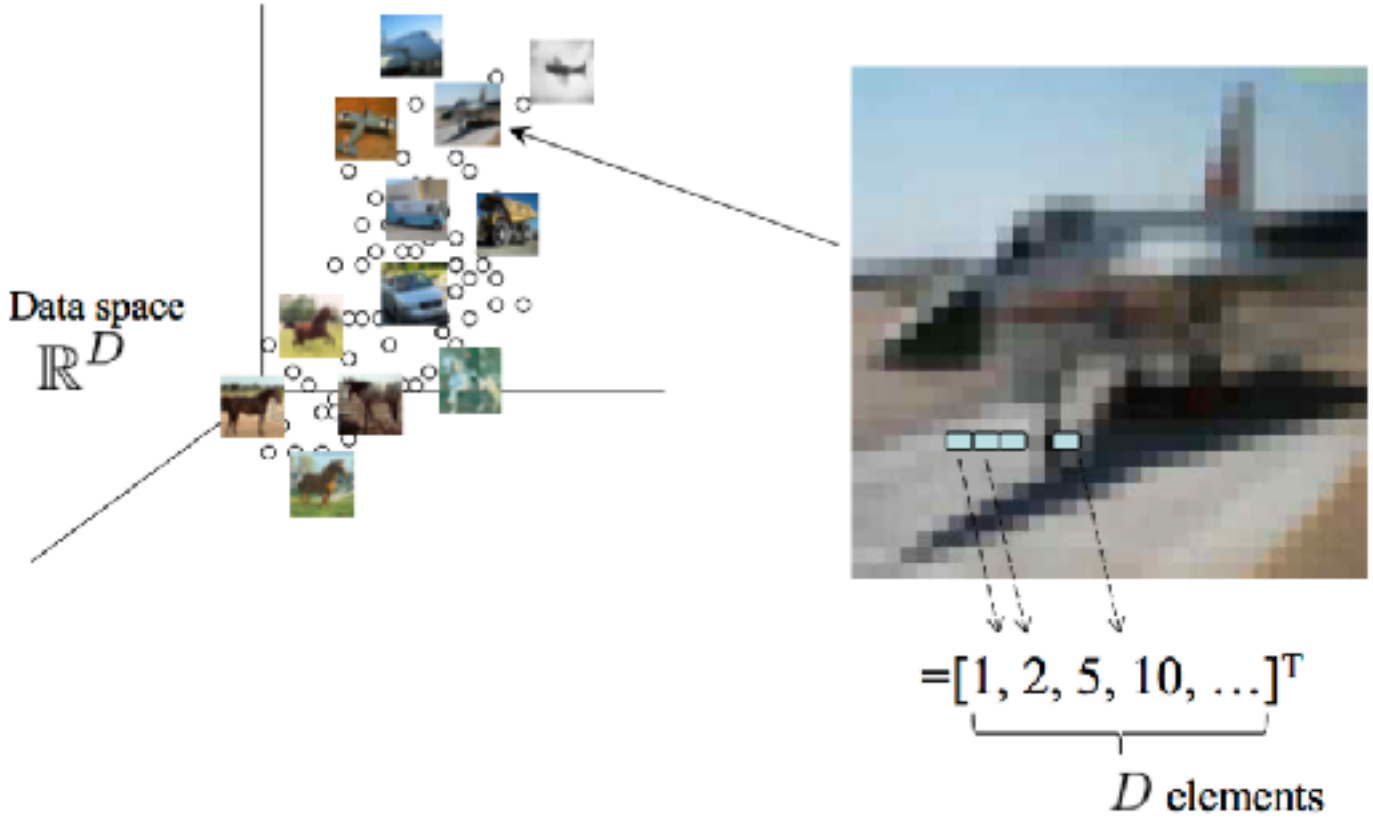
Regularity

- Prediction
  - $\mathrm{x} \in \mathbb{R}^{D} \xrightarrow{y = f(\mathrm{x})} y \in \{1,2,\ldots,C\}$ or $y \in \mathbb{R}$

- Learning
  - Learn prediction function $f(\mathbf{x}) \in \mathcal{H}$ from data $\mathcal{D}$
    - $\mathcal{H}$: Hypothesis set / Candidate set / Model set

Representation of Data

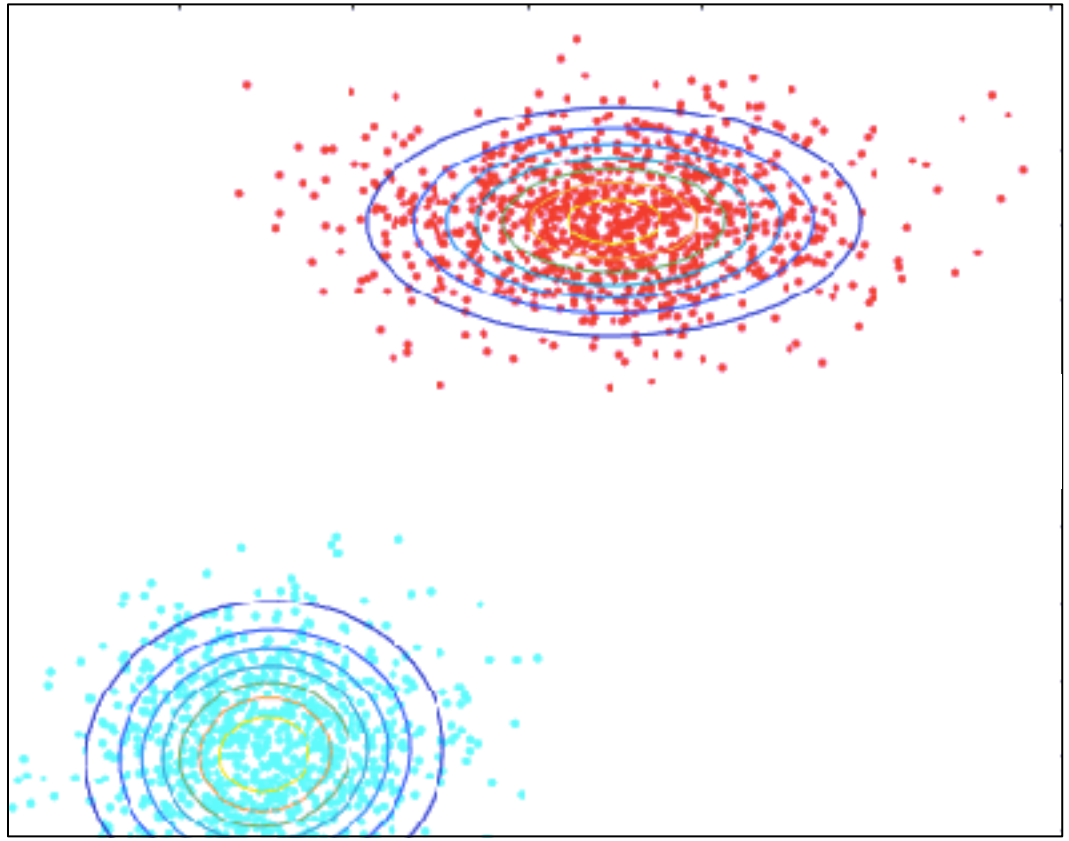
- Each datum is one point in a data space



$$\mathbf{x} \in \mathbb{R}^D$$
$$y \in \{1,2,\ldots,C\}$$

Assumptions for Learning

- Trained data and data for prediction
  are **randomly** chosen
  from the **same** population
  = **Randomly** generated from
  the **same** underlying density
  function
  = **Regularity** exists



Again, Learning

- Data
  - $\mathcal{D} = \{\mathrm{x}_i, y_i\}_{i=1}^{N} \sim P$

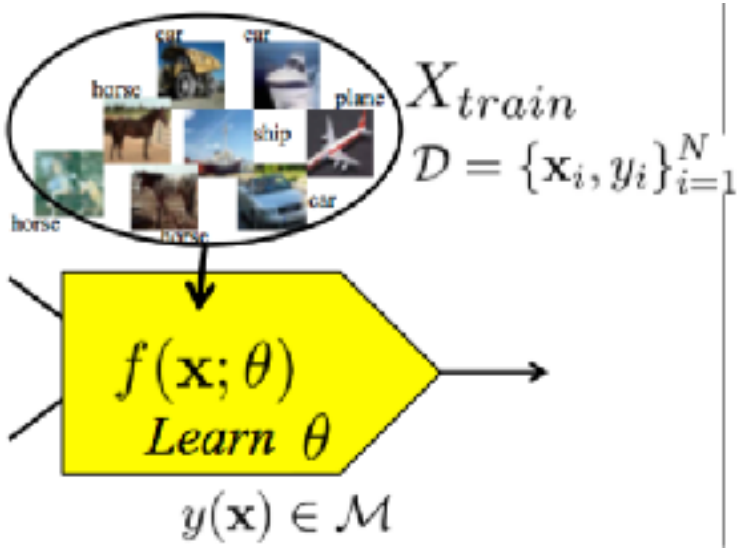
Regularity

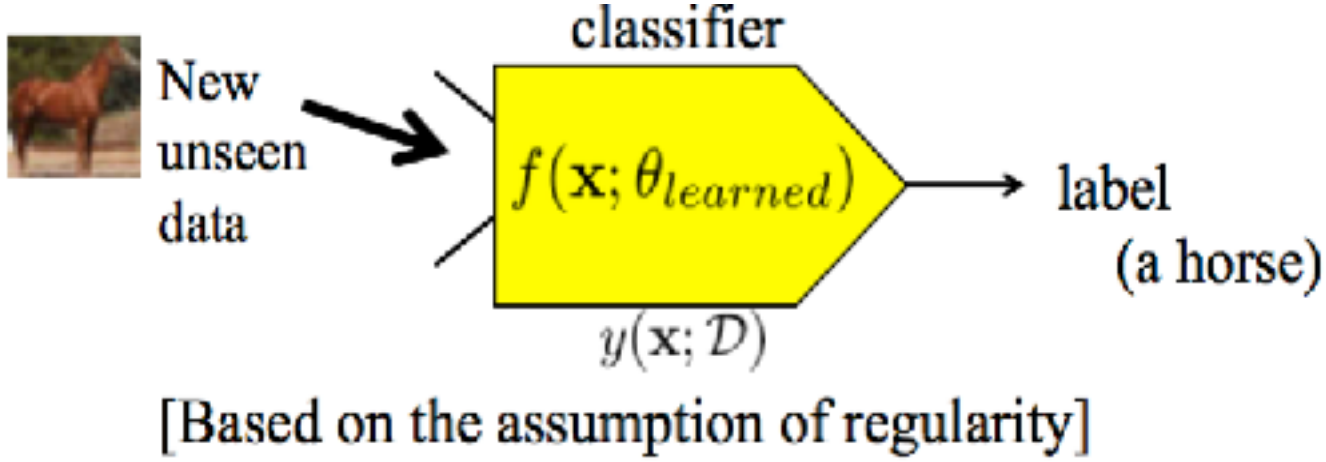
- Prediction
  - $\mathrm{x} \in \mathbb{R}^{D} \xrightarrow{y = f(\mathrm{x})} y \in \{1,2,\ldots,C\}$ or $y \in \mathbb{R}$

- Learning
  - Learn prediction function $f(\mathbf{x}) \in \mathcal{H}$ from data $\mathcal{D}$
    - $\mathcal{H}$: Hypothesis set / Candidate set / Model set

Supervised Learning and Prediction

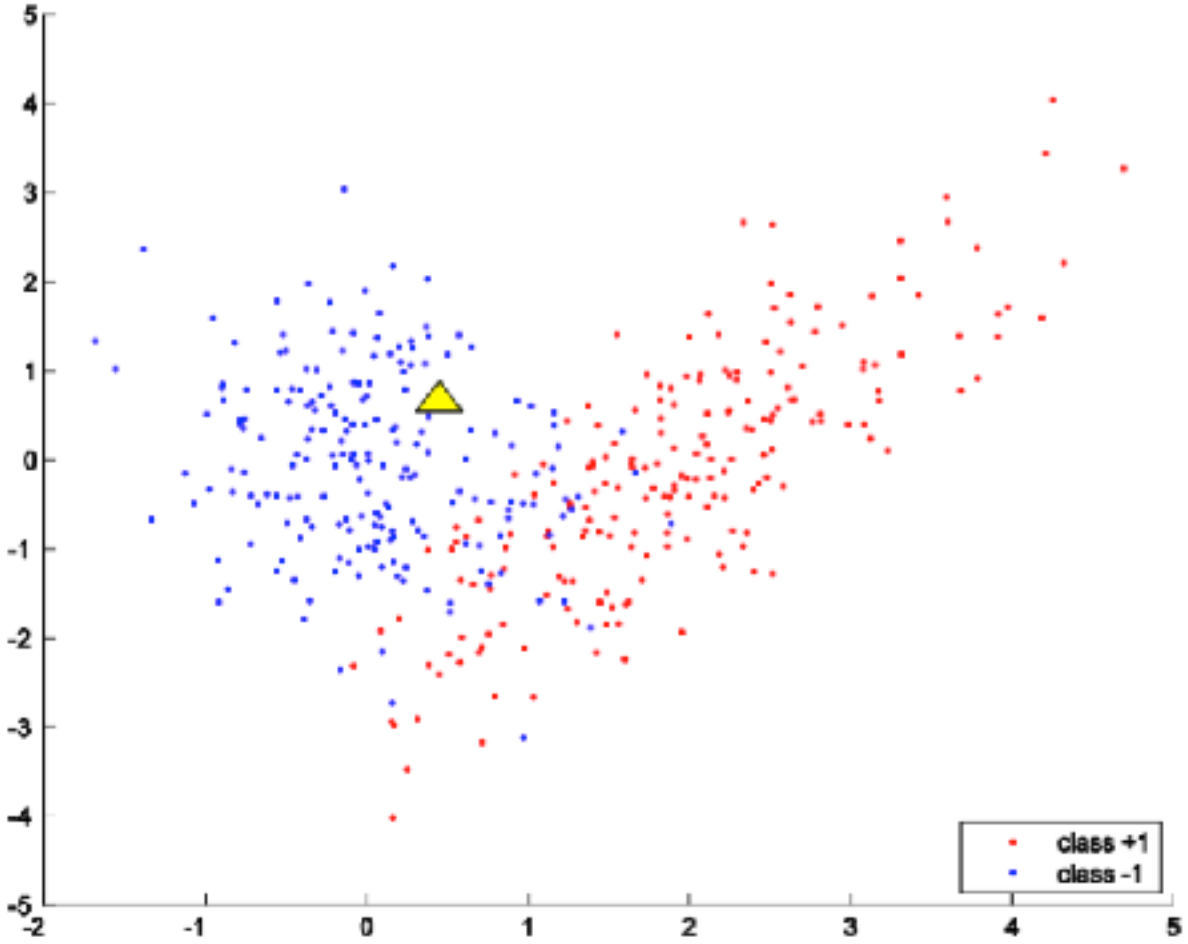
- Method
  - Learning from **examples** and classifying an **unseen data**





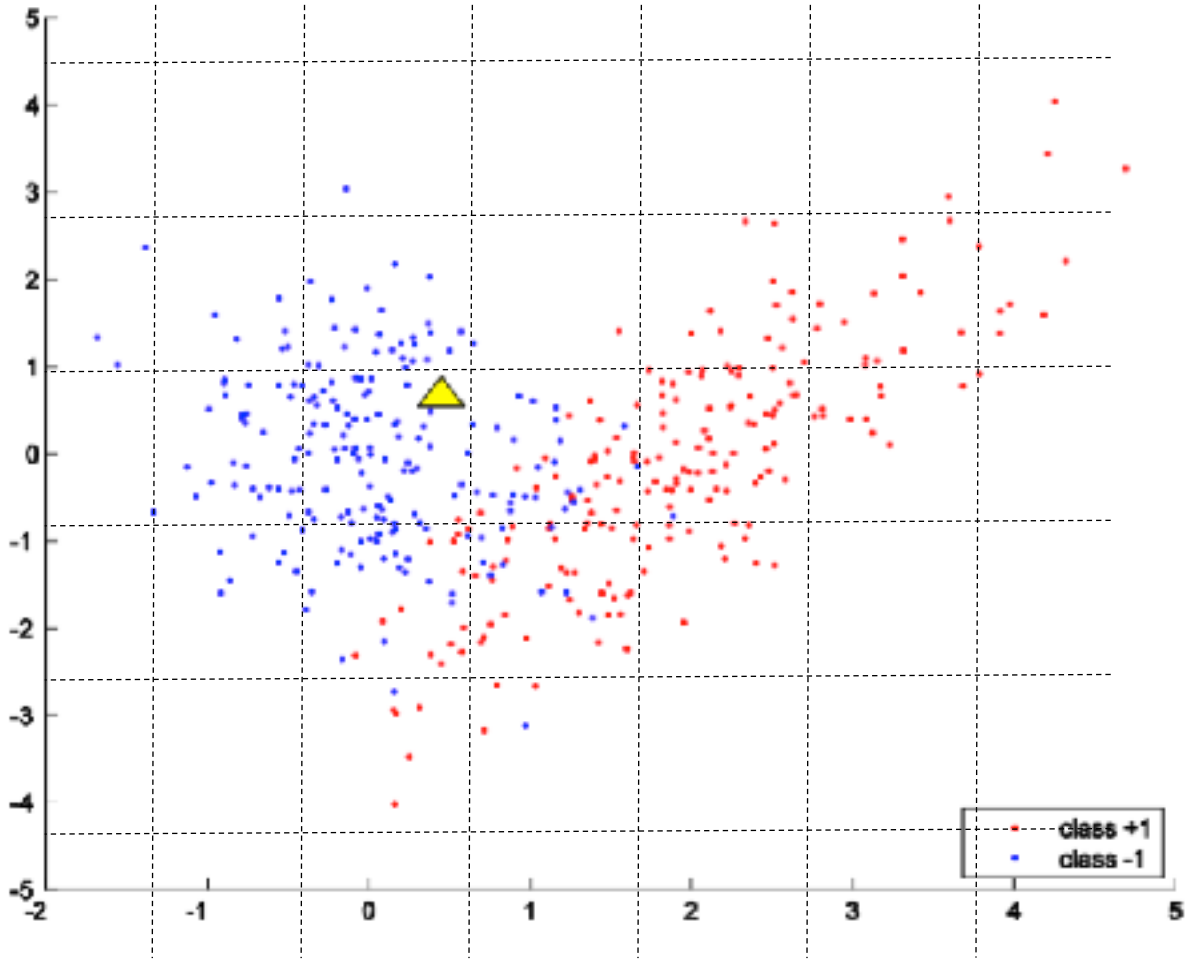
Comparisons on Classifiers

- Problem



$$x \in \mathbb{R}^2$$
$$y \in \{1, -1\}$$

Classifier 1: Divide Data Space into Bins



$$y \in \{+1, -1\}$$

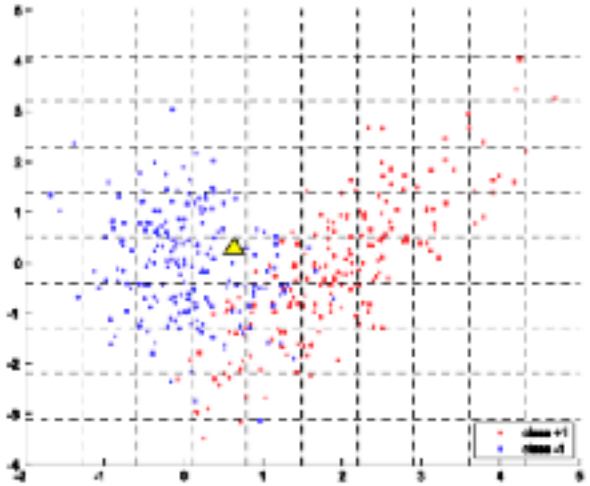
$$h(\mathbf{x}) = +1 \text{ if}$$

$$\sum_{i=1}^{N} y_i \mathbb{I}(\mathbf{x}_i \in \text{bin}(\mathbf{x})) \geq 0$$

$$h(\mathbf{x}) = -1 \text{ if}$$

$$\sum_{i=1}^{N} y_i \mathbb{I}(\mathbf{x}_i \in \text{bin}(\mathbf{x})) < 0$$

Classifier 1: Divide Data Space into Bins



$$y \in \{+1, -1\}$$

$$h(\mathbf{x}) = +1 \quad \text{if} \quad \sum_{i=1} N y_i \mathbb{I}(\mathbf{x}_i \in \text{bin}(\mathbf{x})) \geq 0$$

$$h(\mathbf{x}) = -1 \quad \text{if} \quad \sum_{i=1} N y_i \mathbb{I}(\mathbf{x}_i \in \text{bin}(\mathbf{x})) < 0$$

$|\theta| = 2^{\text{Number of bins}}$

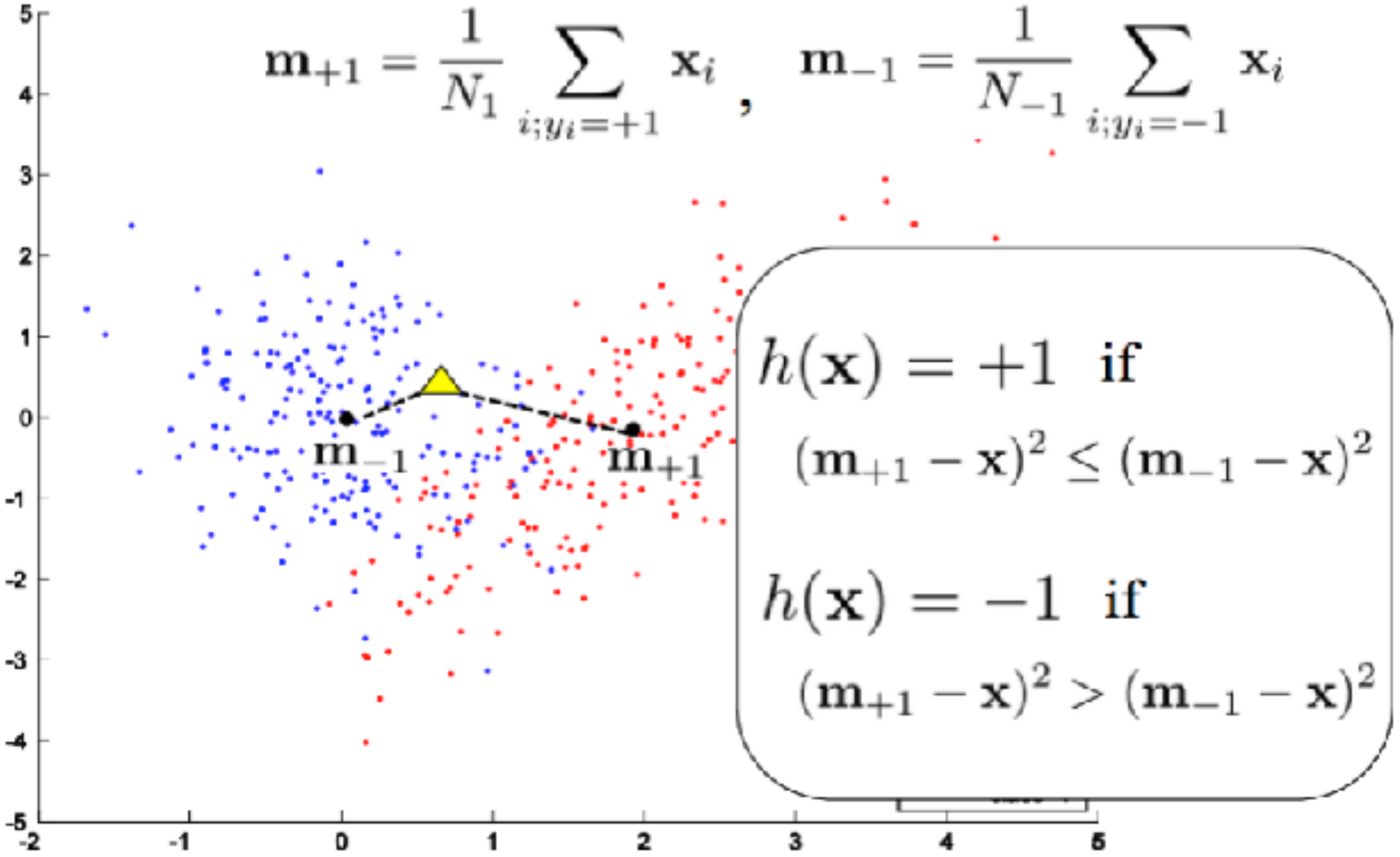
Data ↑ → Bayes error

Dimensionality ↑ → intractable

Local structure

Classifier 2: Compare Distances to Means

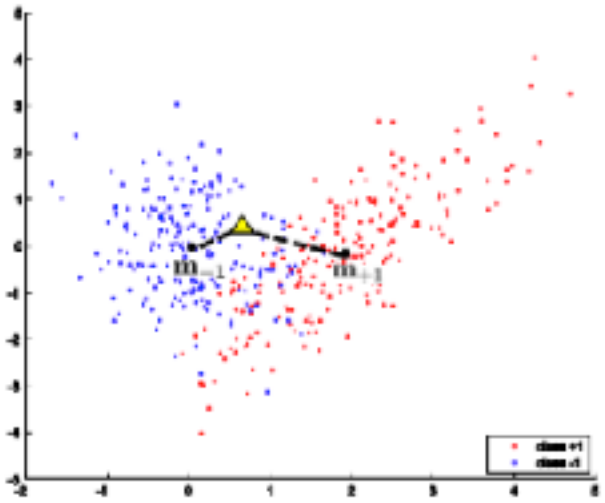
$$\mathbf{m}_{+1} = \frac{1}{N_1} \sum_{i;y_i=+1} \mathbf{x}_i \;, \quad \mathbf{m}_{-1} = \frac{1}{N_{-1}} \sum_{i;y_i=-1} \mathbf{x}_i$$



$$h(\mathbf{x}) = +1 \;\; \text{if}$$
$$(\mathbf{m}_{+1} - \mathbf{x})^2 \leq (\mathbf{m}_{-1} - \mathbf{x})^2$$

$$h(\mathbf{x}) = -1 \;\; \text{if}$$
$$(\mathbf{m}_{+1} - \mathbf{x})^2 > (\mathbf{m}_{-1} - \mathbf{x})^2$$

Classifier 2: Compare Distances to Means



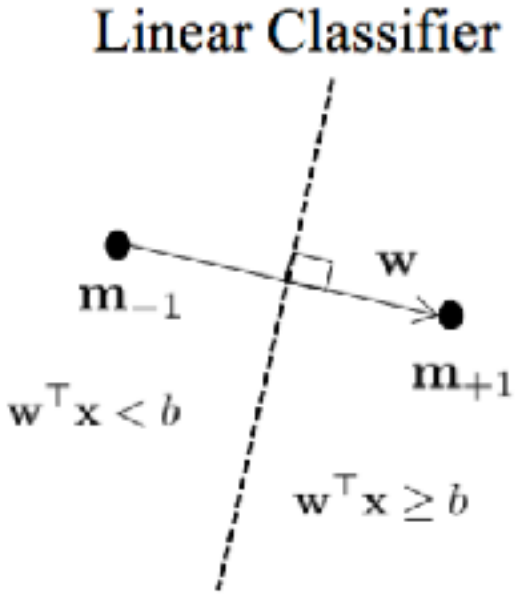
$$\mathbf{m}_{+1} = \frac{1}{N_1} \sum_{i;y_i=+1} \mathbf{x}_i, \quad \mathbf{m}_{-1} = \frac{1}{N_{-1}} \sum_{i;y_i=-1} \mathbf{x}_i$$

$$h(\mathbf{x}) = +1 \quad \text{if} \quad (\mathbf{m}_{+1} - \mathbf{x})^2 \leq (\mathbf{m}_{-1} - \mathbf{x})^2$$

$$h(\mathbf{x}) = -1 \quad \text{if} \quad (\mathbf{m}_{+1} - \mathbf{x})^2 > (\mathbf{m}_{-1} - \mathbf{x})^2$$

- Dimensionality ↑ → Can be handled
- Data ↑ → Bayes error is not achieved

Classifier 2: Compare Distances to Means



$$(\mathbf{m}_{+1} - \mathbf{x})^2 \gtrless (\mathbf{m}_{-1} - \mathbf{x})^2$$

$$\mathbf{m}_{+1}^2 - 2\mathbf{m}_{+1}^\top \mathbf{x} + \mathbf{x}^2 \gtrless \mathbf{m}_{-1}^2 - 2\mathbf{m}_{-1}^\top \mathbf{x} + \mathbf{x}^2$$

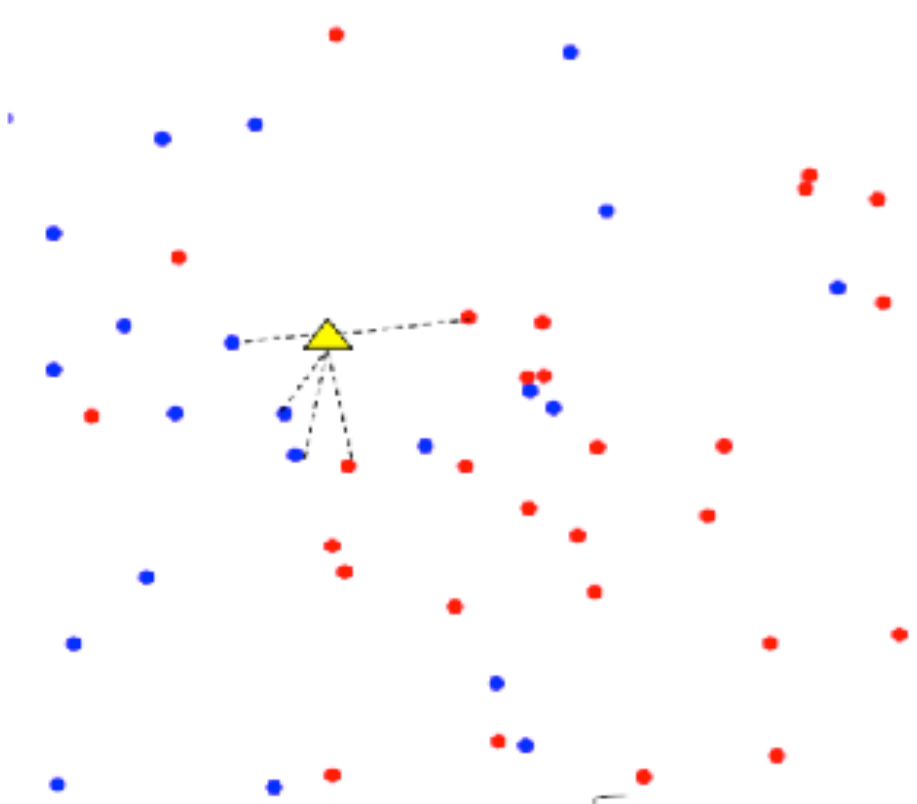
$$\mathbf{m}_{+1}^\top \mathbf{x} - \mathbf{m}_{-1}^\top \mathbf{x} \gtrless \frac{\mathbf{m}_{+1}^2 - \mathbf{m}_{-1}^2}{2}$$

$$\mathbf{w} = \mathbf{m}_{+1} - \mathbf{m}_{-1}, \quad b = \frac{\mathbf{m}_{+1}^2 - \mathbf{m}_{-1}^2}{2}$$

$$h(\mathbf{x}) = +1 \quad \text{if} \quad \mathbf{w}^\top \mathbf{x} - b \geq 0$$

$$h(\mathbf{x}) = -1 \quad \text{if} \quad \mathbf{w}^\top \mathbf{x} - b < 0$$

Classifier 3: k-Nearest Neighbor Classifier

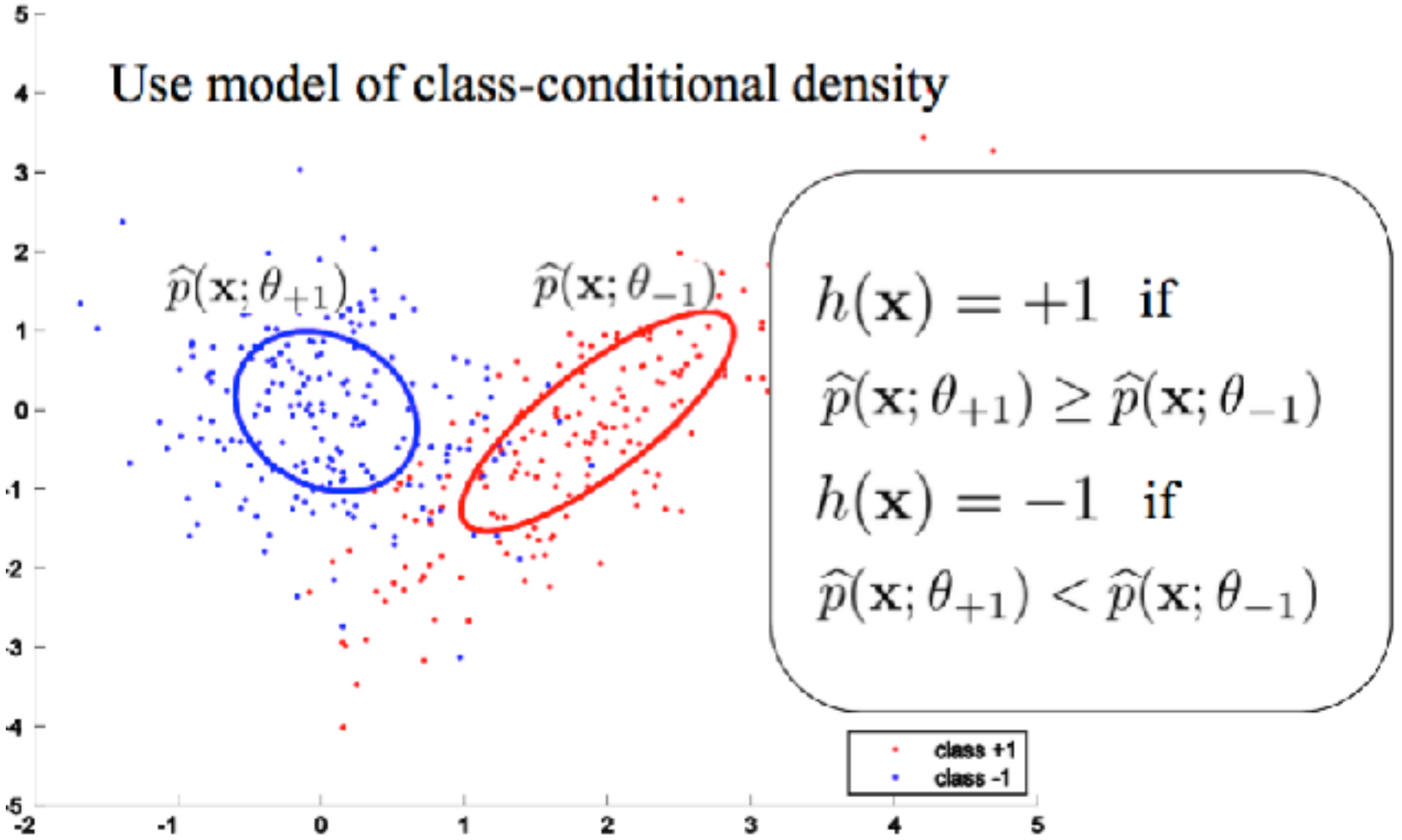


$k = 5$

Majority voting

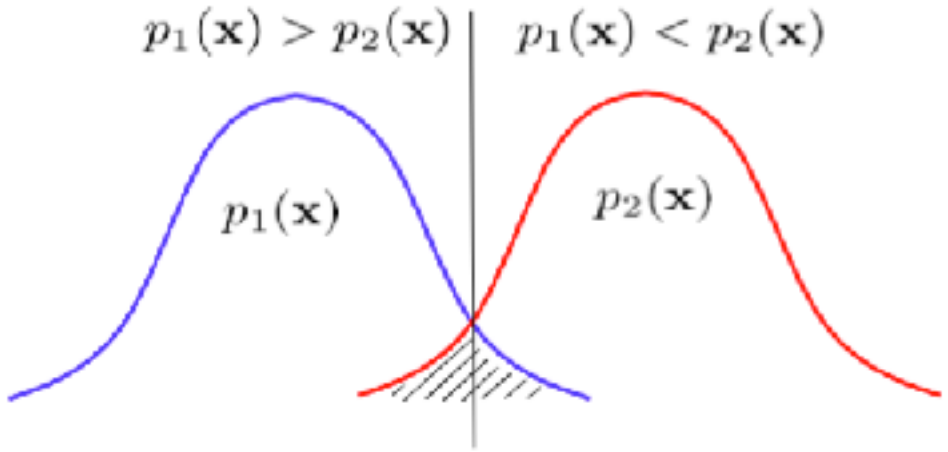
- Dimensionality ↑ → Can be handled
- Data ↑ → Bayes error is achieved

Classifier 4: Gaussian Model



Bayes Optimal Classifier

- Our ultimate goal is not a zero error



Bayes classifier

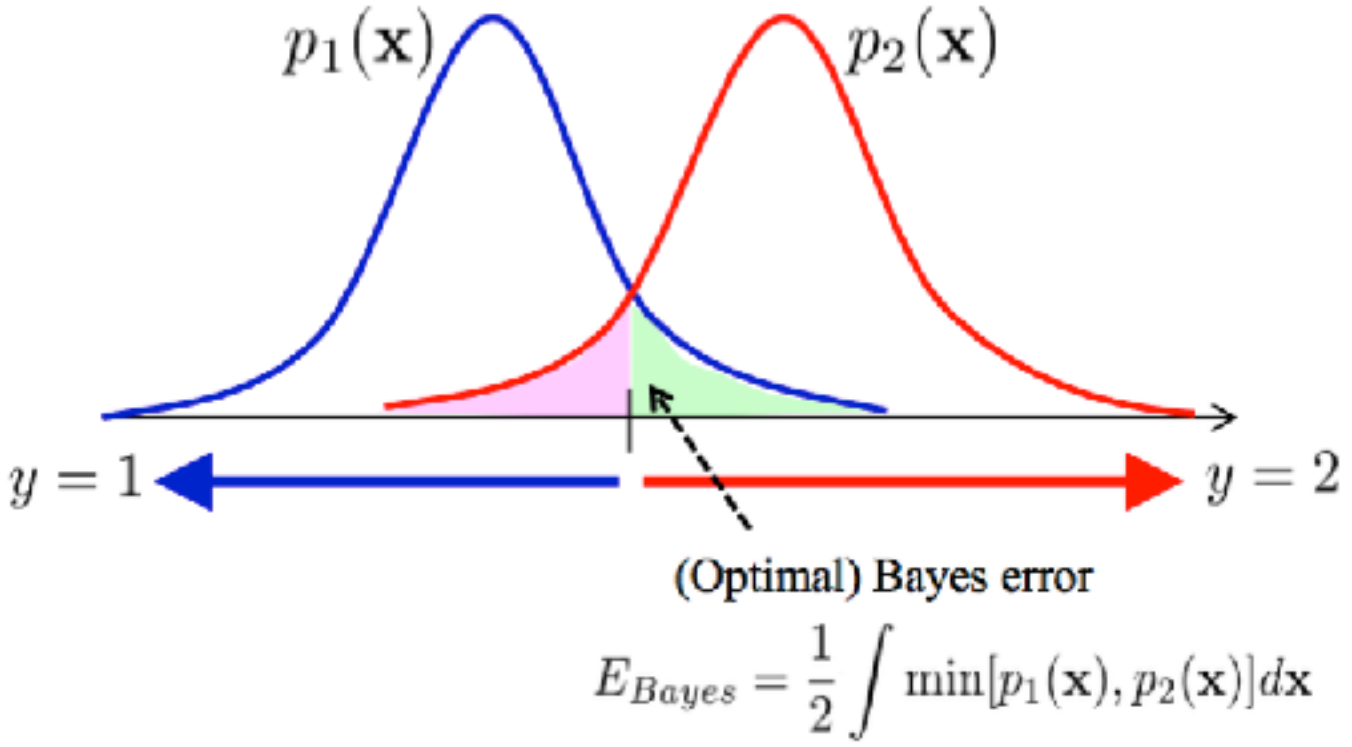


Figure credit: Masashi Sugiyama

Generative vs. Discriminative Methods

- Generative methods
  - Gaussian class-conditional model
  - Graphical model
  - Restricted Boltzmann Machines

$$p(\mathbf{x}, y)$$

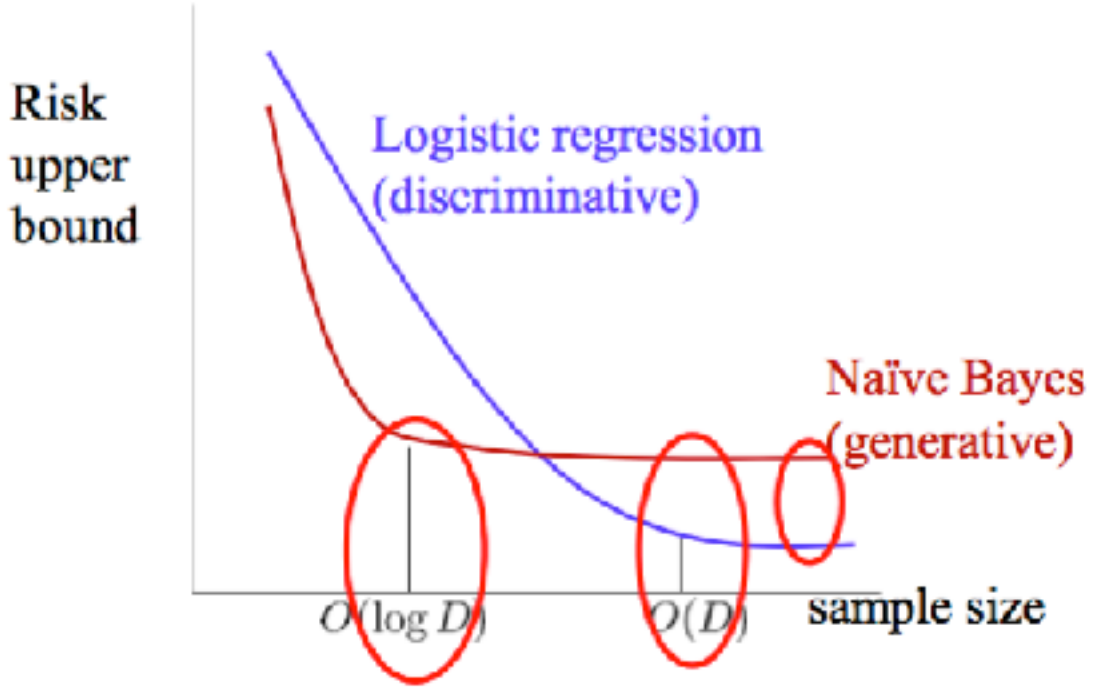
$$p(y = 1|\mathrm{x}) = \frac{p(\mathrm{x}|y = 1)p(y = 1)}{p(\mathrm{x})}$$

- Discriminative methods
  - Support Vector Machines (SVMs)
  - Logistic Regression
  - Artificial Neural Networks

Learn $p(y = 1|\mathbf{x})$ directly

Generative vs. Discriminative Methods

- Generative vs. discriminative pair
  - Same number of parameters, same form of f(x)



Ng & Jordan, 2001, NIPS

Quantify the Evaluation

- Measure of quality: expected loss
  - $L = \mathbb{E}_P[l(y, f(\mathbf{x}))]$

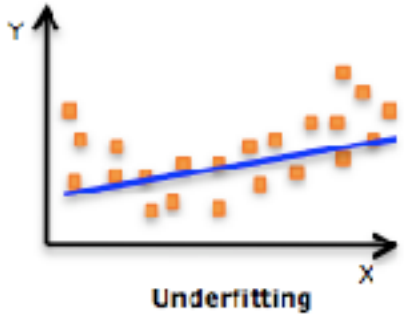
- Estimated error
  - $\hat{L} = \sum_n l\big(y_n, f(\mathbf{x}_n)\big),\ f(\mathbf{x}) \in \mathcal{H}$

- Examples
  - Classification
  - Regression
  - Clustering

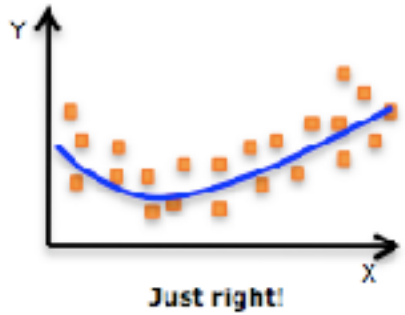
$$\hat{L} = \sum_n \mathbb{I}(y_n \neq f(\mathbf{x}_n))$$
$$\hat{L} = \sum_n ||y_n - f(\mathbf{x}_n)||^2$$
$$\hat{L} = \sum_n \min_c ||y_n - f(\mathbf{x}_n)||^2$$

Generalization and Overfitting

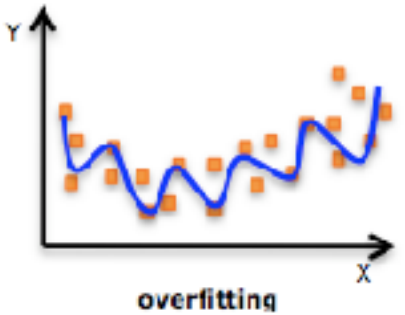
$$y = a_2x^2 + a_1x + a_0 + Noise$$



$$y = a_1x + a_0$$

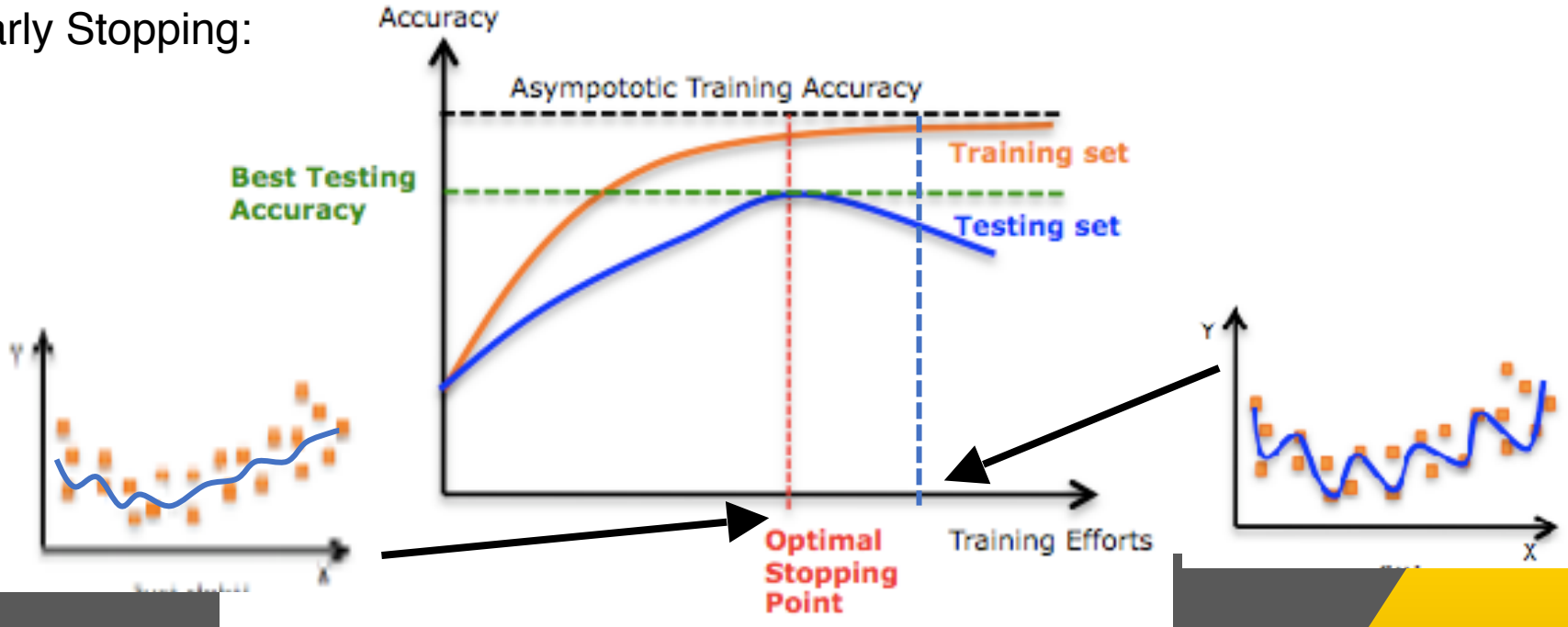


$$y = a_2x^2 + a_1x + a_0$$



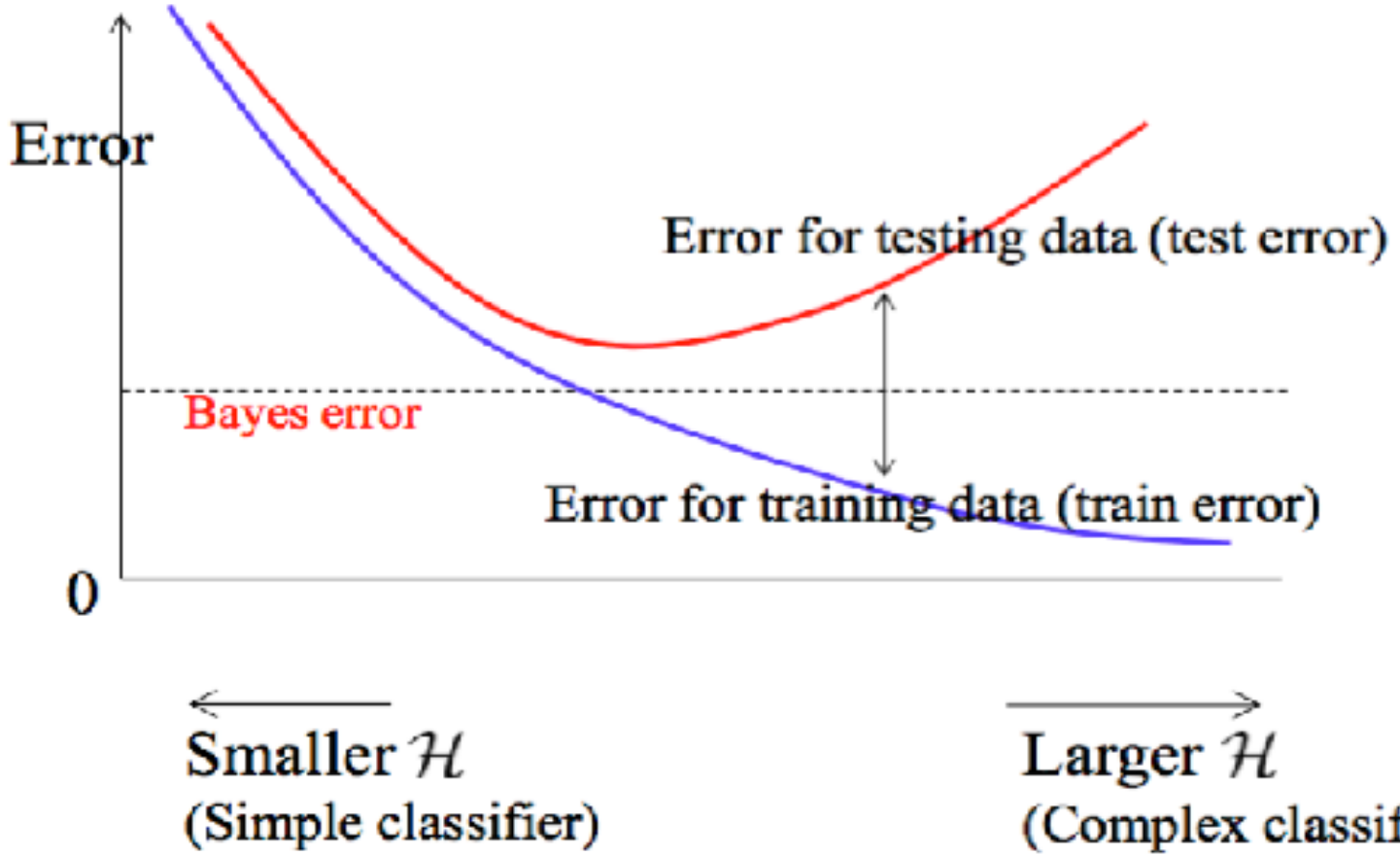
$$y = a_{10}x^{10} + \ldots + a_1x + a_0$$

Early Stopping:

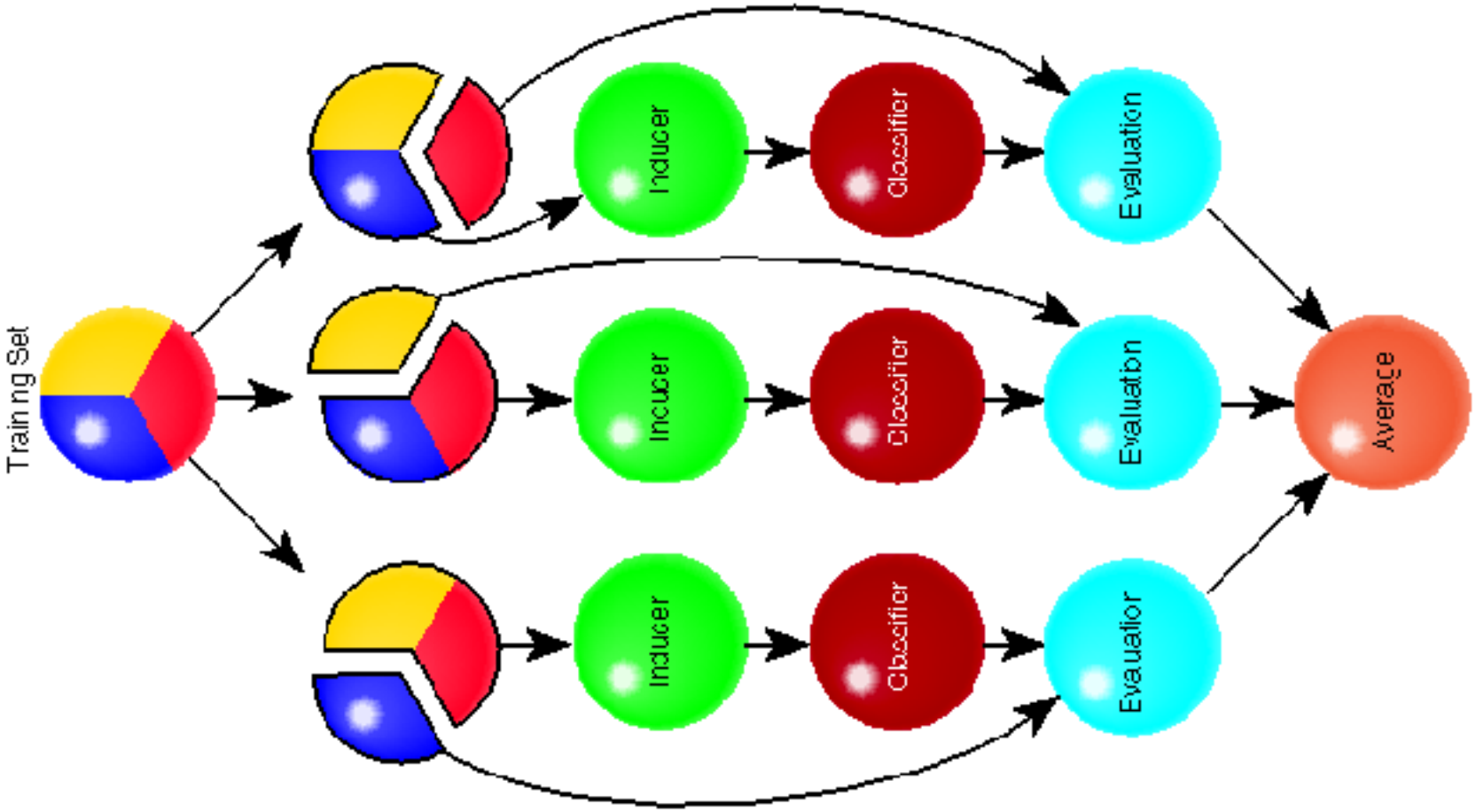


Generalization and Overfitting

- Minimizing estimated error



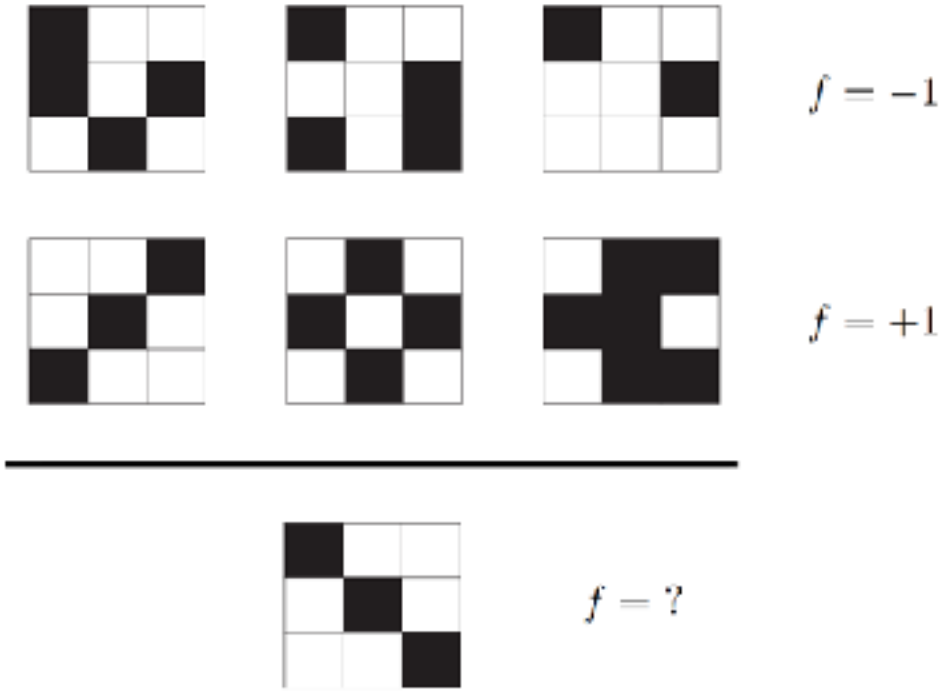
Generalization and Overfitting



Feasibility of Learning

A Puzzle Example

- A simple learning task
  - 6 training examples of a ± 1 target function



A learning puzzle

- There is more than one function that fits the 6 training examples
  - Did you say $f = +1$? (f measures symmetry)
  - Did you say $f = -1$? (f considers only the top left pixel)

- Which is correct?
  - We cannot rule out either possibility!

Inference outside the given data set

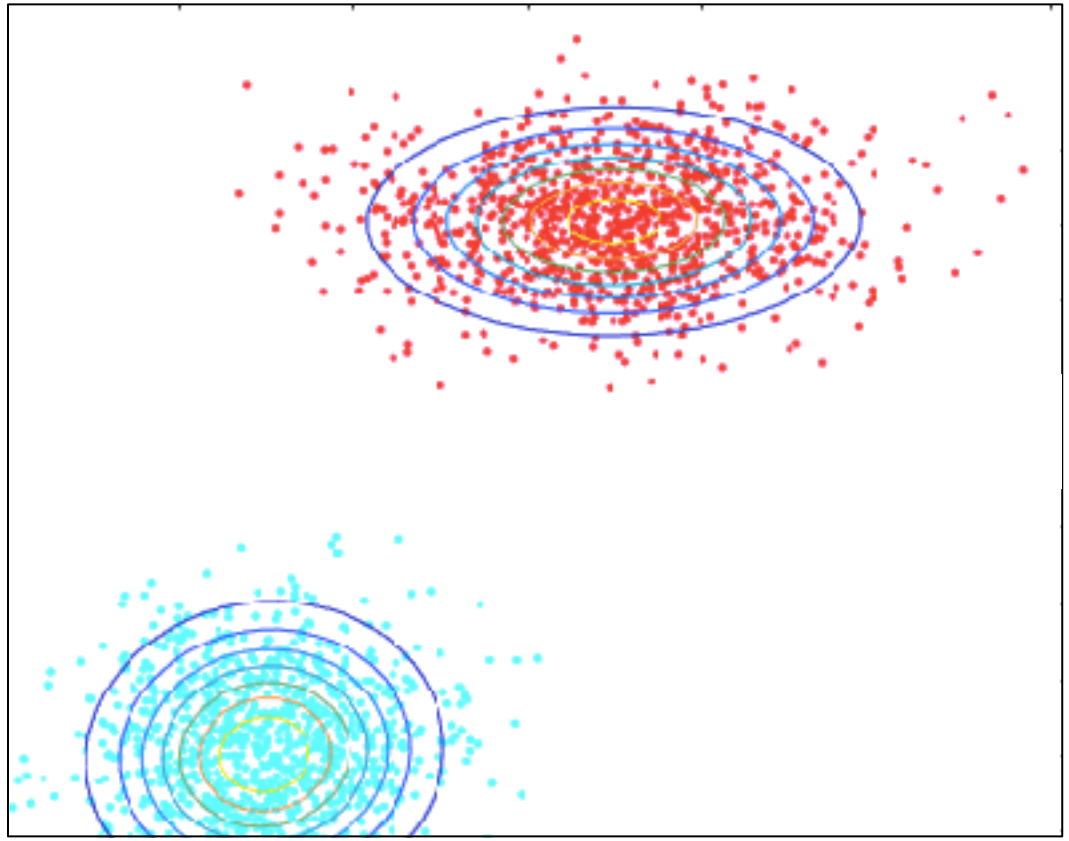
- In the training data D, we know the value of f on all the points
  - But this does not mean that we have learned f
  - No guarantee that we know anything about f outside of D
- We know what we have already seen
  - But that's not *learning* but *memorizing*
- Key question: does the data set D tell us
  - anything outside of D we did not know before?
- If the answer is yes, then we have learned *something*
  - *Otherwise, we can conclude that learning is not feasible*

Inference outside the given data set

- We will show: we can infer something outside D using only D
  - In a ***probabilisitic way***, giving up knowing about f "for sure"
- What we infer in this way:
  - Will establish the principle that we can reach outside D
- We will take this principle to the general learning problem
  - And determine what we can and cannot learn

Again! Assumptions for Learning

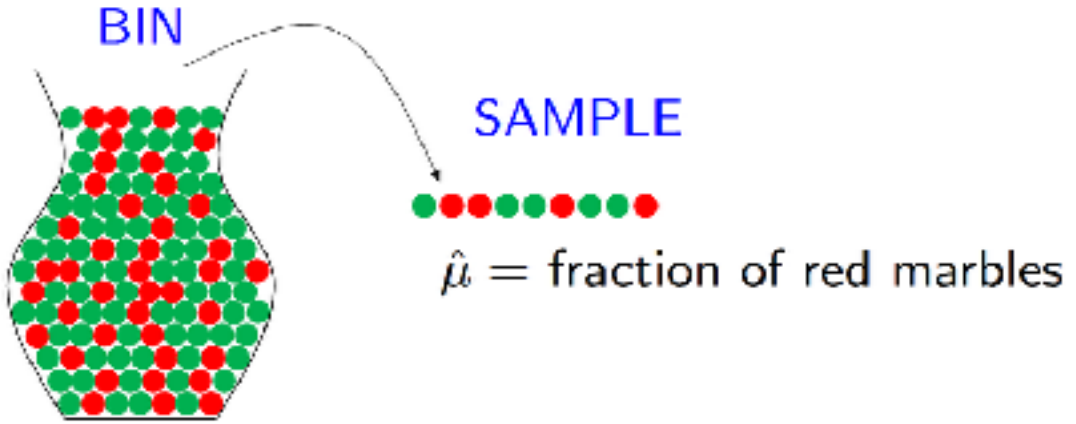
- Trained data and data for prediction
  are **randomly** chosen
  from the **same** population
  = **Randomly** generated from
  the **same** underlying density
  function
  = **Regularity** exists



A ‘bin’ experiment

- Consider a bin with red and green marbles (possibly infinitely many)
  - $P[picking\ a\ red\ marble] = \mu$, $P[picking\ a\ green\ marble] = 1 - \mu$
  - The value of μ : unknown

- Pick N marbles independently (with replacement)
  - The faction of red marbles within the sample $\triangleq \hat{\mu}$



Inferring with probability

- Do you **know** μ?

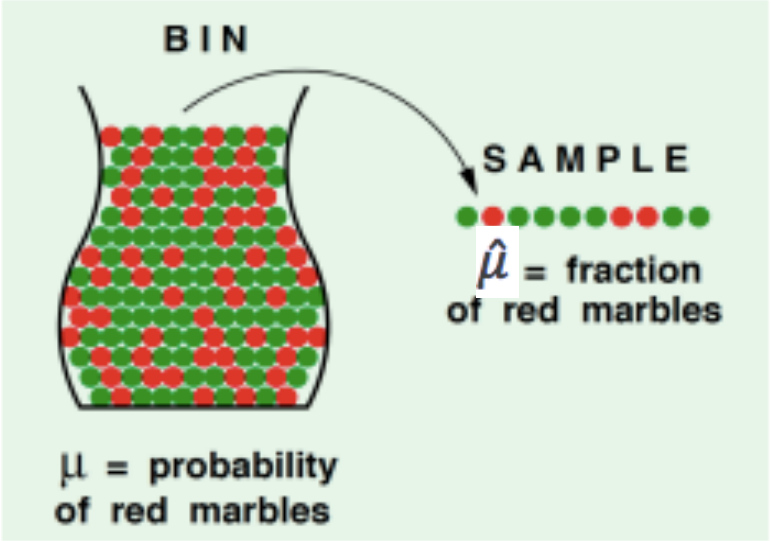
  No

  Sample can be mostly green while bin is mostly red

- Can you say something about μ?

  Yes

  $\hat{\mu}$ is "probably" close to μ



Hoeffding inequality

for any sample size N and $\epsilon > 0$:

$$P[|\hat{\mu} - \mu| > \epsilon] \leq 2e^{-2N\epsilon^2}$$

- P : probability of an event (here, with regard to random sample)
- The bound does not depend on μ or on the bin size
- In words
  - As the sample size N grows, it becomes exponentially unlikely that $\hat{\mu}$ will deviate from μ by more than our 'tolerance' ϵ

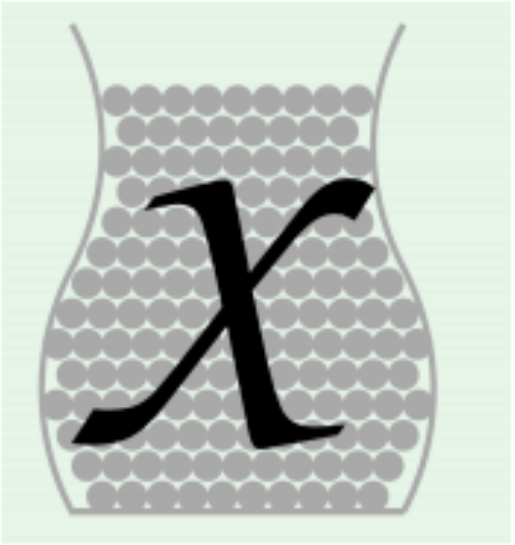
Hoeffding inequality

- Are the bin model and learning connected?
  - You may wonder how the bin model and learning are connected
    - The unknown in the bin problem: just the value of μ
    - The unknown in learning: an entire function $f: X \rightarrow Y$
  - The two situation can indeed be connected: we'll show that
    - $\mu \approx$ the error rate a hypothesis makes when approximating f

Verifying a Hypothesis

- Bin = input space X
  - f is unknown → the color of each point in the bin is unknown
  - But, if we pick $\boldsymbol{x}$ randomly w.r.t some probability distribution P over X
    - $\boldsymbol{x} = \begin{cases} red & \text{with some probability } \mu \\ green & \text{with probability } 1-\mu \end{cases}$

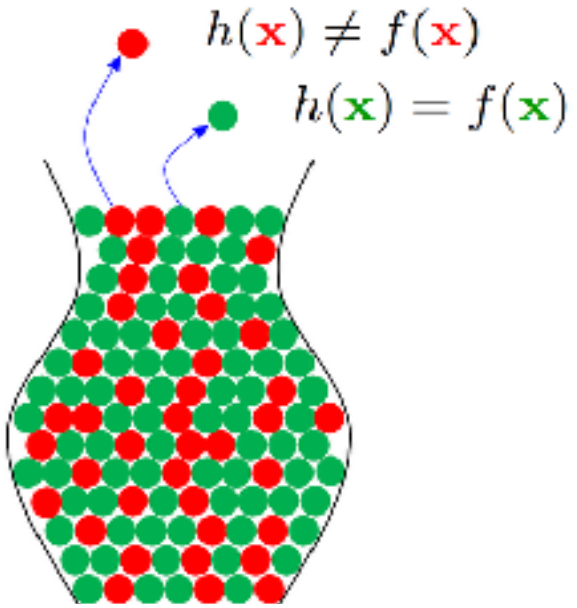
Again the bin experiment



- Bin coloring = hypothesis h
  - Take a particular hypothesis $h \in H$ and compare it to f on each $\boldsymbol{x} \in X$

$$\boldsymbol{x} = \begin{cases} green & if\ h(\boldsymbol{x}) = f(\boldsymbol{x}): hypothesis\ right \\ red & if\ h(\boldsymbol{x}) \neq f(\boldsymbol{x}): hypothesis\ wrong \end{cases}$$

  - Note: a different h will give a different coloring of the balls in the bin



Verifying a Hypothesis

- Sample = training data set D
  - Training examples play the role of a sample from the bin
  - If points $x_1, \ldots, x_N$ in D are picked independently according to P
    - We will get a random sample of red and green points
    - $x = \begin{cases} red & with\ probability\ \mu \\ green & with\ probability\ 1-\mu \end{cases}$
  - The color of each point in D will be known to us
    - Since both $h(x_i)$ and $f(x_i)$ are known for $i = 1, \ldots, N$
    - Remember: D is our training data set

Verifying a Hypothesis

- Both $\hat{\mu}$ and our belief in μ depend on which hypothesis h is used
  - $\hat{\mu}$ is in-sample error denoted by $E_{in}(h)$
  - μ is out-of-sample error denoted by $E_{out}(h)$

- For a particular function h:
  - $E_{\text{out}}(h) = E_{x \sim P}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})]$ (out-of-sample, unknown) $\Leftrightarrow \mu$
  - $E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} [h(\boldsymbol{x}_n) \neq y_n]$ (in-sample, known) $\Leftrightarrow \hat{\mu}$
  - Now we can **infer** $E_{\text{out}}(h)$ from $E_{\text{in}}(h)$!

Verifying a Hypothesis

- For any h, when sample size (N) is large:

$$P[|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

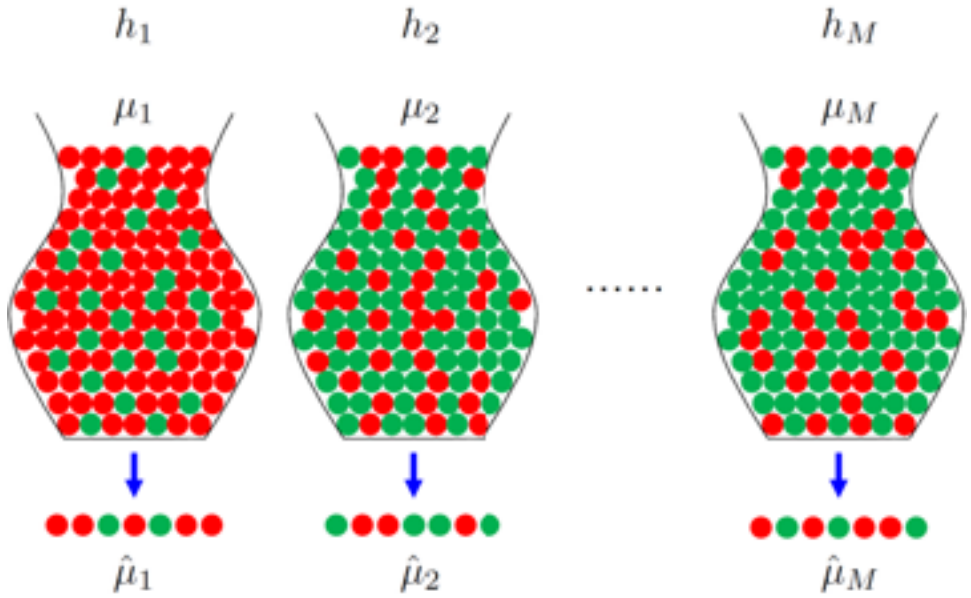
- "$E_{\text{in}}(h) = E_{\text{out}}(h)$" is probably approximately correct (PAC)
- How is this useful?
- If $E_{\text{in}}(h) \approx E_{\text{out}}(h)$ and $E_{\text{in}}(h)$ is small

  $\Rightarrow E_{\text{out}}(h)$ small

  $\Rightarrow h \approx f$ with respect to P
- **Given a hypothesis h $\Rightarrow$ sample N data $\Rightarrow$ $E_{in}(h)$ to "verify" the quality of h**
- Can we apply to mulitple hypothesis?

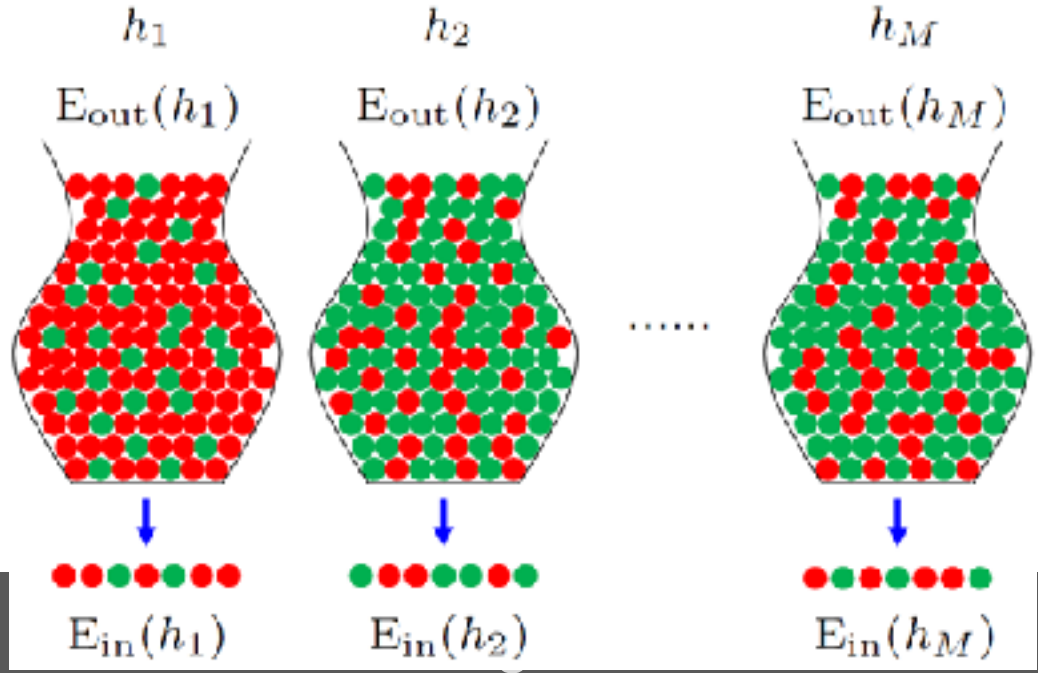
Generalizing the bin model

- In order to capture real learning
  - See if we can extend the bin equivalence to multiple hypotheses
  - Different h → different $\hat{\mu}$ → different approximation of μ



Generalizing the bin model

- Considering a finite number of hypotheses
  - Consider an entire hypothesis set H instead of just one h
    - Assume for now H has a *finite* number of hypotheses
      - $H = \{h_1, h_2, \ldots, h_M\}$
    - We can then construct a bin equivalent by having M bins



Generalizing the bin model

- Considering a finite number of hypotheses
  - Using the union bound

$$
\begin{aligned}
\mathbb{P}[|\mathrm{E}_{\mathrm{in}}(g) - \mathrm{E}_{\mathrm{out}}(g)| > \epsilon] \quad \leq \quad & \mathbb{P}[\quad |\mathrm{E}_{\mathrm{in}}(h_1) - \mathrm{E}_{\mathrm{out}}(h_1)| > \epsilon \\
& \quad \text{or } |\mathrm{E}_{\mathrm{in}}(h_2) - \mathrm{E}_{\mathrm{out}}(h_2)| > \epsilon \\
& \quad \cdots \\
& \quad \text{or } |\mathrm{E}_{\mathrm{in}}(h_M) - \mathrm{E}_{\mathrm{out}}(h_M)| > \epsilon] \\
\leq \quad & \sum_{m=1}^{M} \mathbb{P}[|\mathrm{E}_{\mathrm{in}}(h_m) - \mathrm{E}_{\mathrm{out}}(h_m)| > \epsilon] \\
\leq \quad & 2Me^{-2\epsilon^2 N}
\end{aligned}
$$

Generalizing the bin model

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

Two questions:

(1) Can we make sure $E_{out}(g) \approx E_{in}(g)$?

(2) Can we make sure $E_{in}(g) \approx 0$?

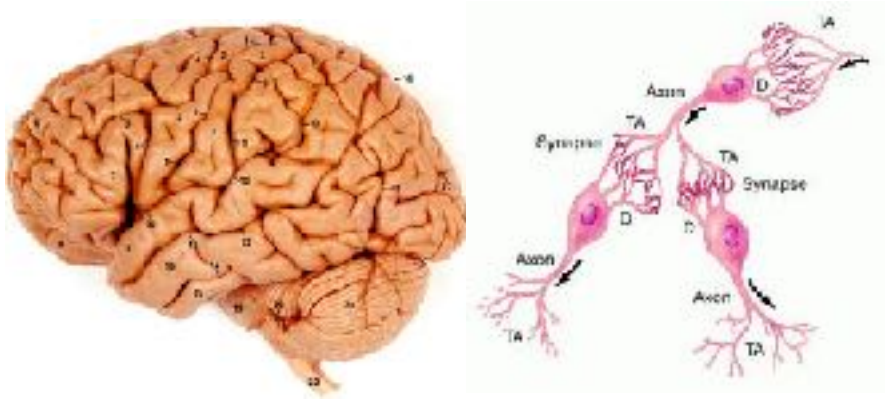
M: complexity of model

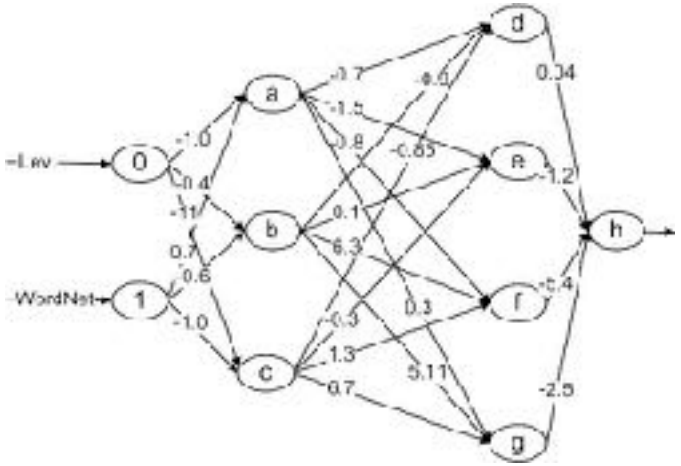
- Small M: (1) holds, but (2) may not hold (too few choices) (under-fitting)
- Large M: (1) doesn't hold, but (2) may hold (over-fitting)
- How to handle infinite M? (will discuss in next lectures)

기계 학습 알고리즘 학습하기

What is the ‘machine’?

- Computer
- A system, not a (human) brain
- **Model**
  - Mathematical abstraction of a system
    - Equation
    - Function
  - Element, structure, parameters, state, dynamics, ...
  - Deterministic / Probabilistic


Axon
Synapse
Axon
Axon



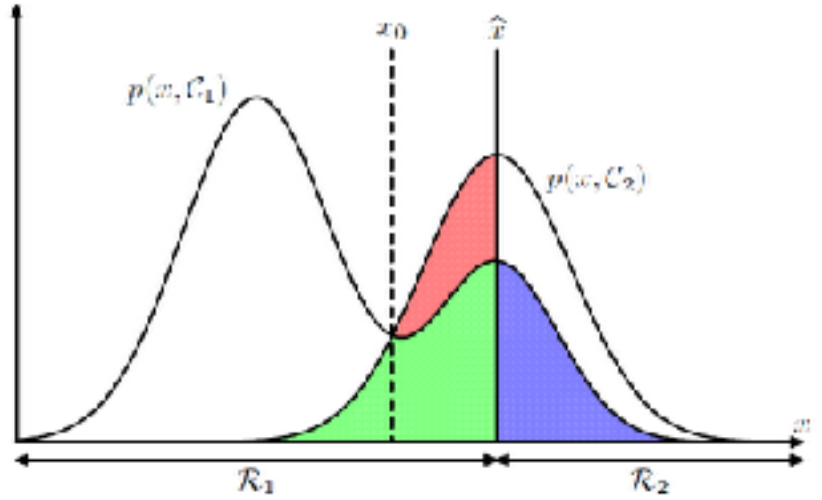
What is "Learning"?

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

- Training?

T (Task) = Something Done (Output)

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



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 reducti
- Semi-supervised learning
  - Web image auto-tagging
- Reinforcement learning
  - Chess

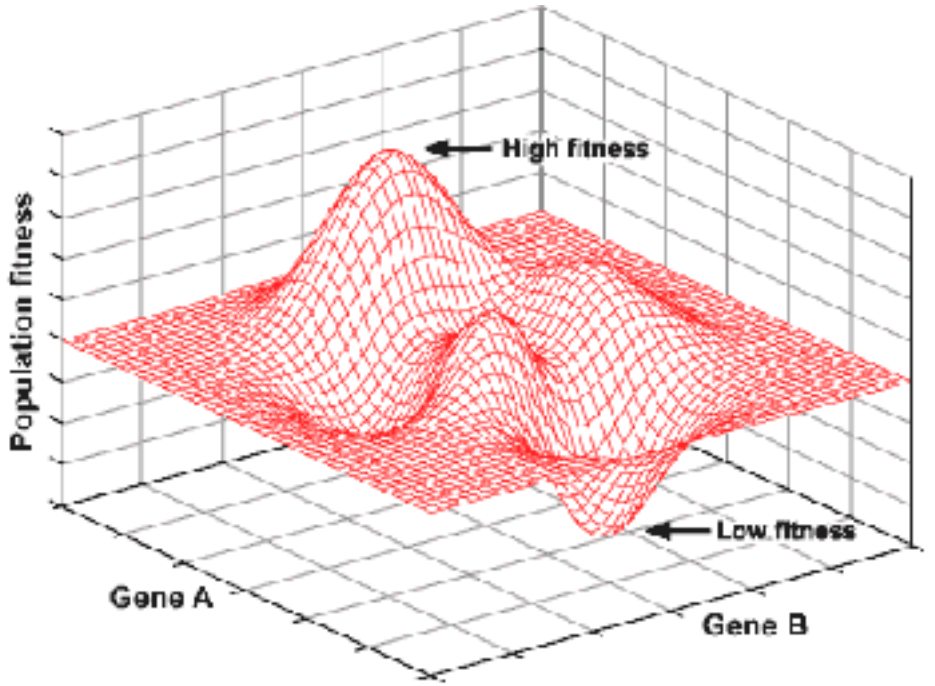
No.	sepallength Numeric	sepalwidth Numeric	petallength Numeric	petalwidth Numeric	**class** Nominal
48	4.6	3.2	1.4	0.2	Iris-setosa
49	5.3	3.7	1.5	0.2	Iris-setosa
50	5.0	3.3	1.4	0.2	Iris-setosa
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54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

- Error rate?
- Euclidean distance
  - e.g. , where $e_j(n) = d_j(n) - y_j(n)$

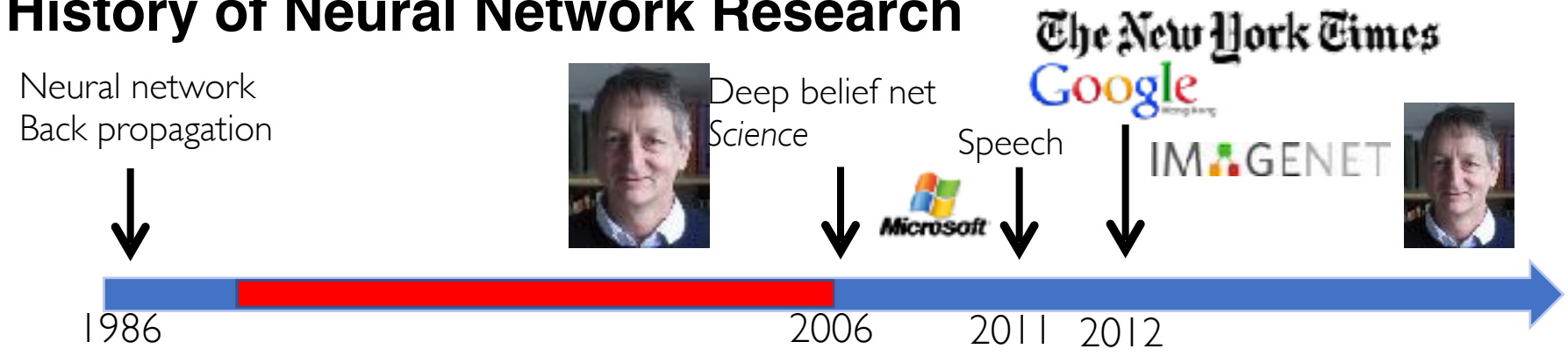
$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

- Log Probability
  - e.g. $L(w) = -\Sigma \log P(y=d^{(i)}|X^{(i)})$
- Information theoretical measures
  - Mutual information
  - KL
- ...



Neural Networks

History of Neural Network Research



deep learning results

- Unsupervised & Layer-wised pre-training
- Boosting

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	

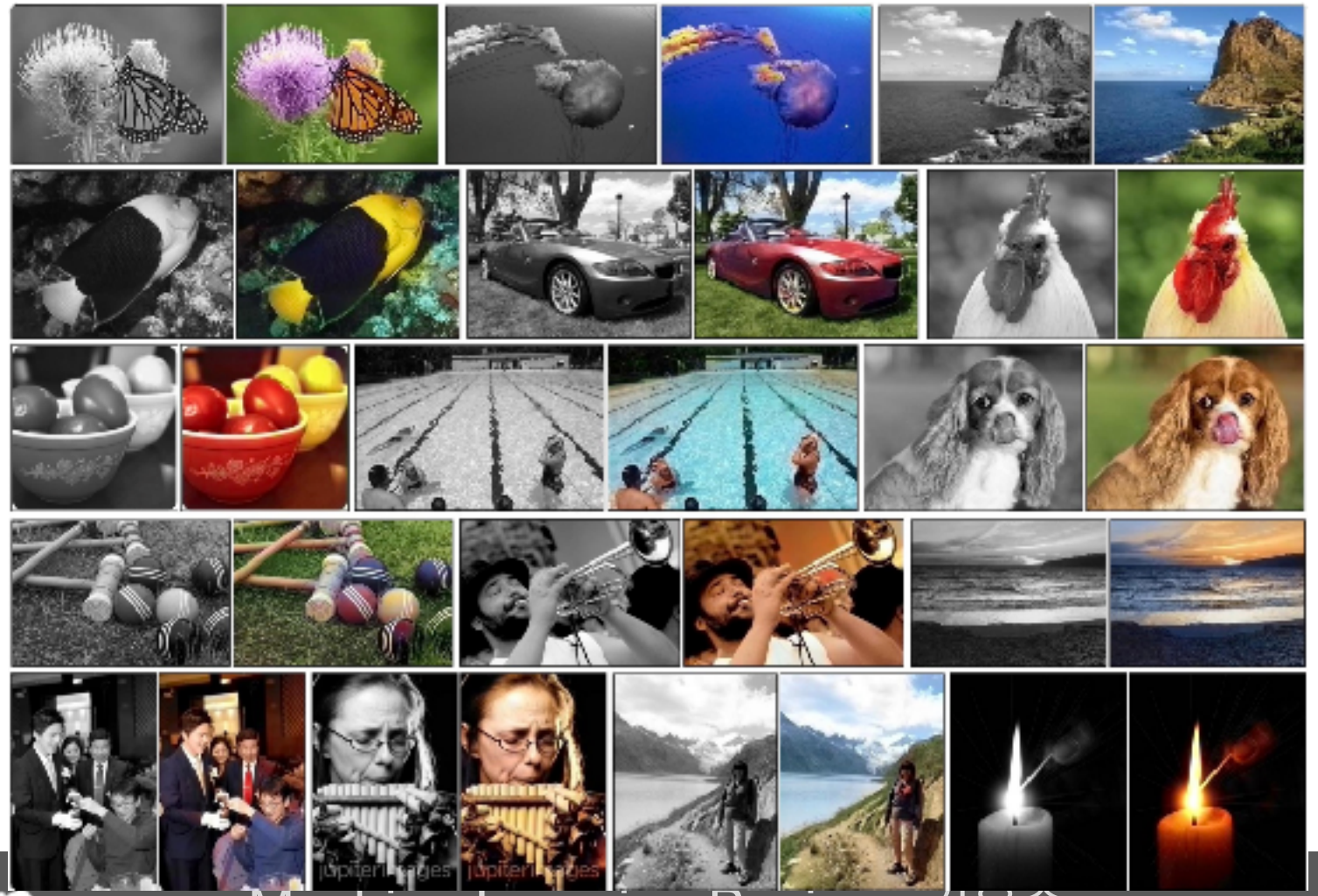
ms
HOG)
ctures

Now: Object recognition over 15M images, 22K categories
(2 GPU)

Deep Networks Advance State of Art in Speech

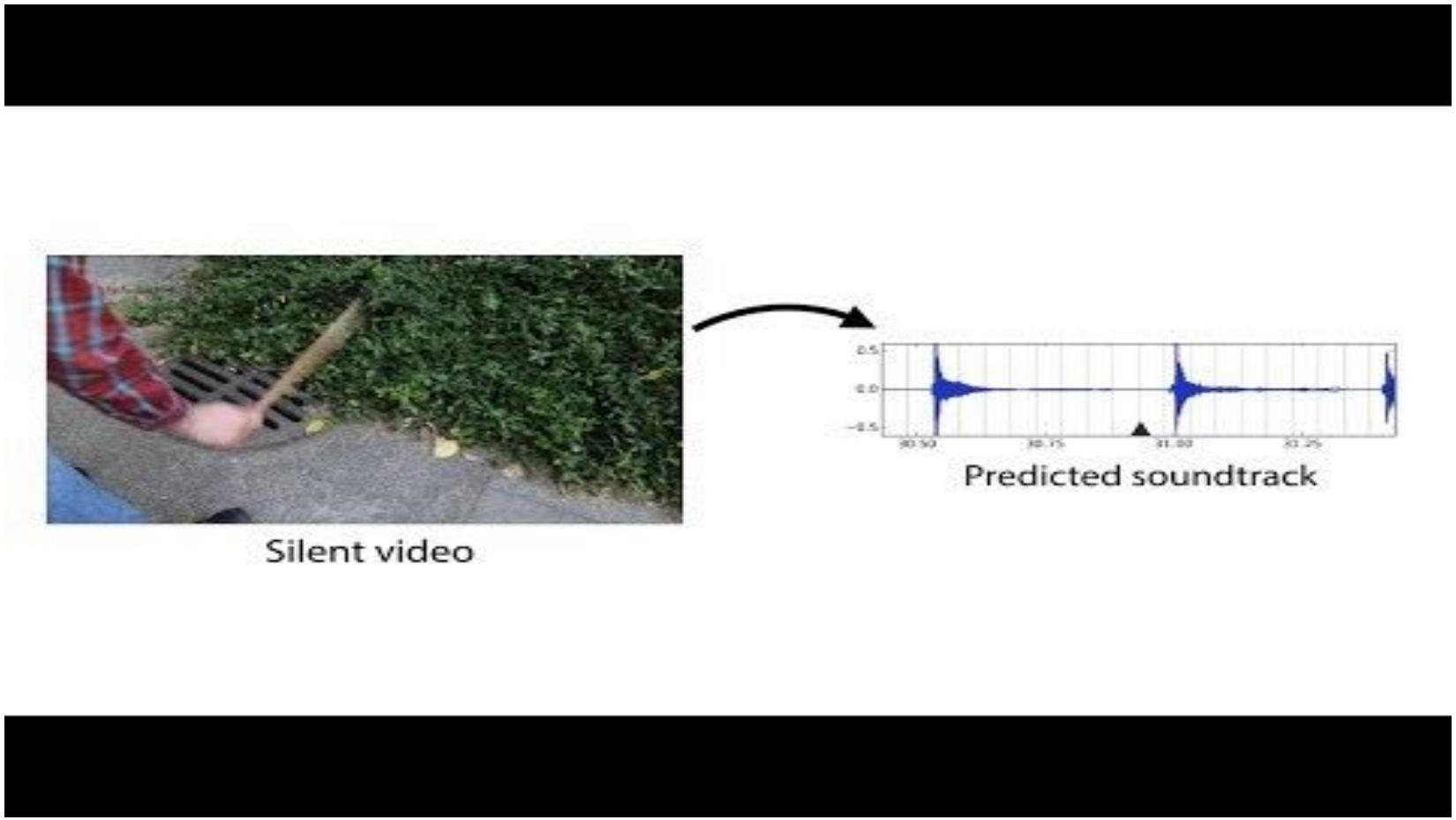
Deep Learning leads to breakthrough in speech recognition at MSR.

1. Automatic Colorization of Black and White images

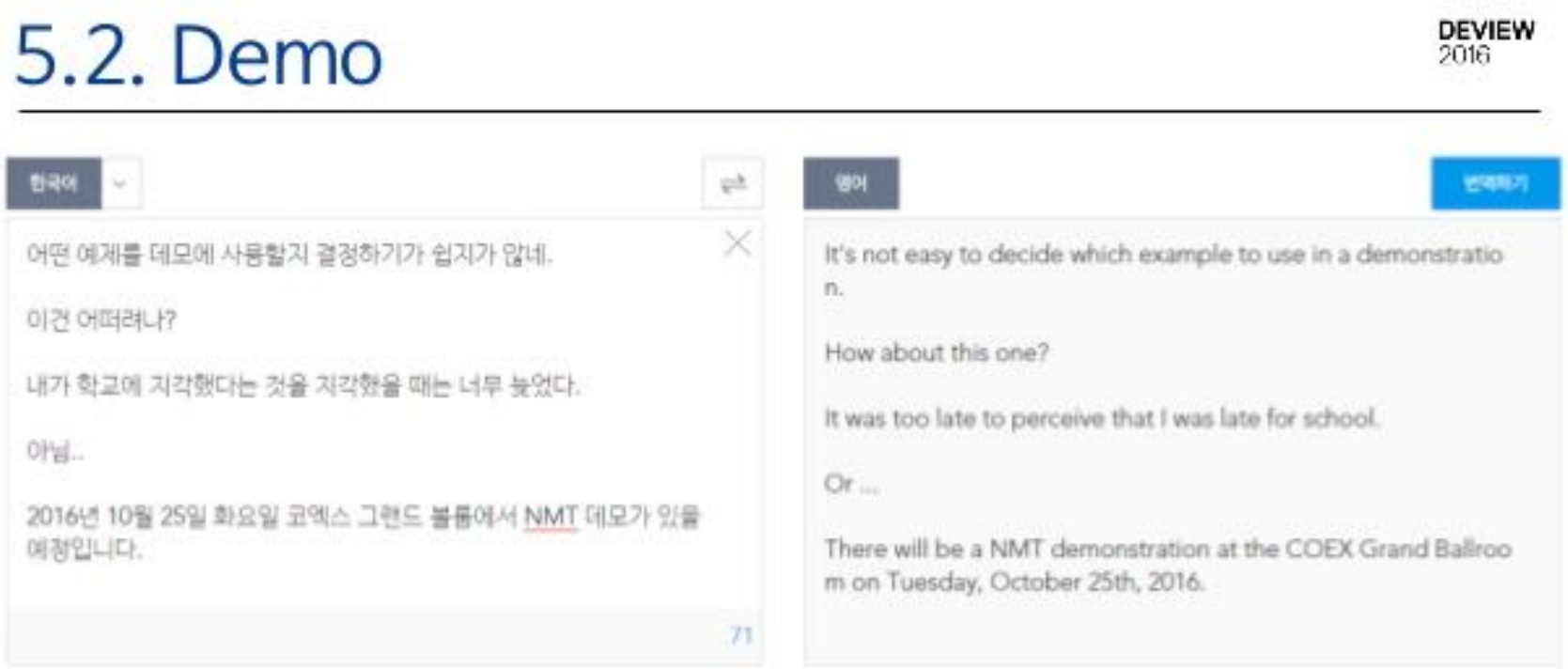


http://richzhang.github.io/colorization/

2. Automatically adding sounds to silent movies



3. Automatic Machine Translation

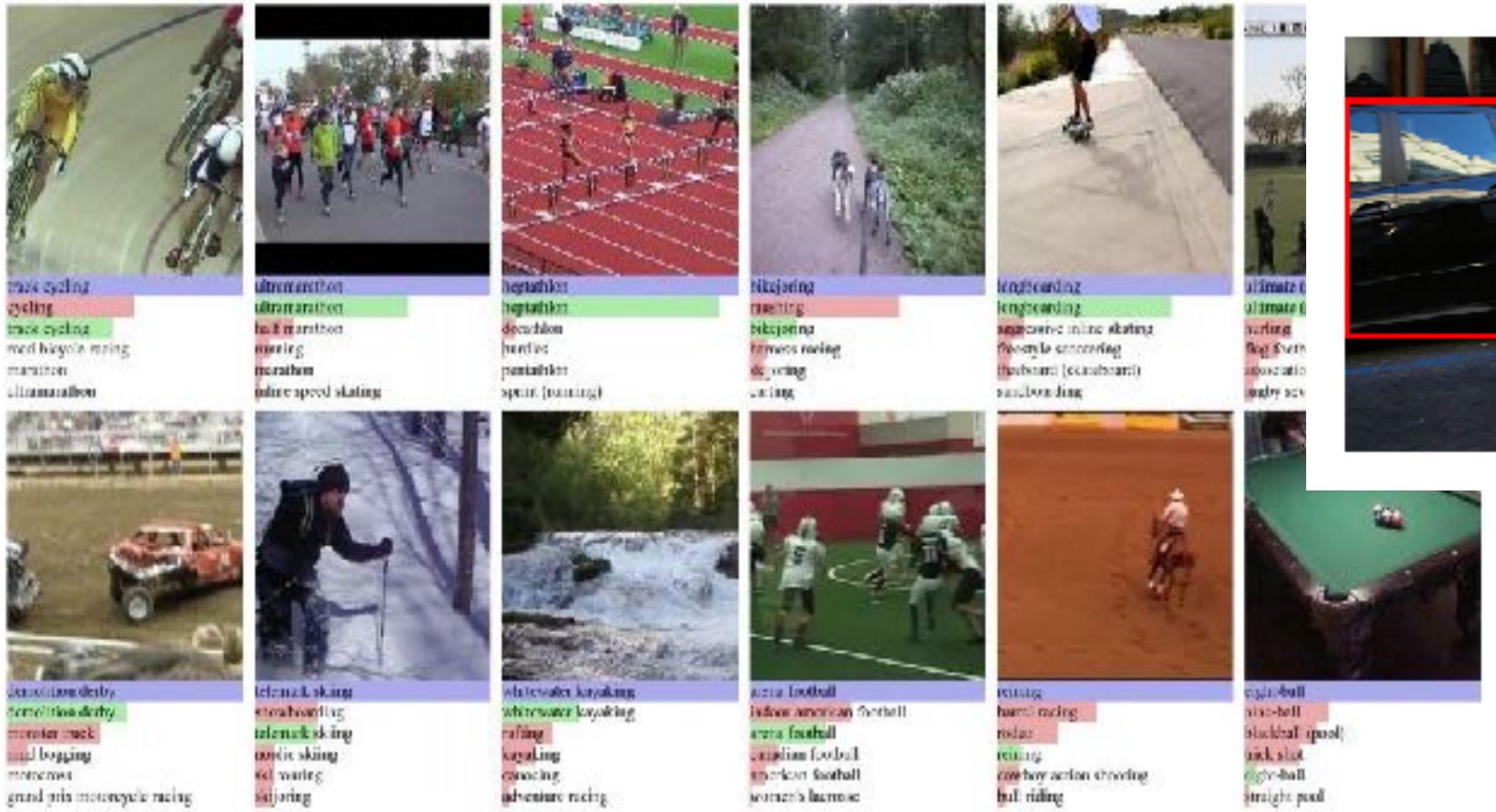


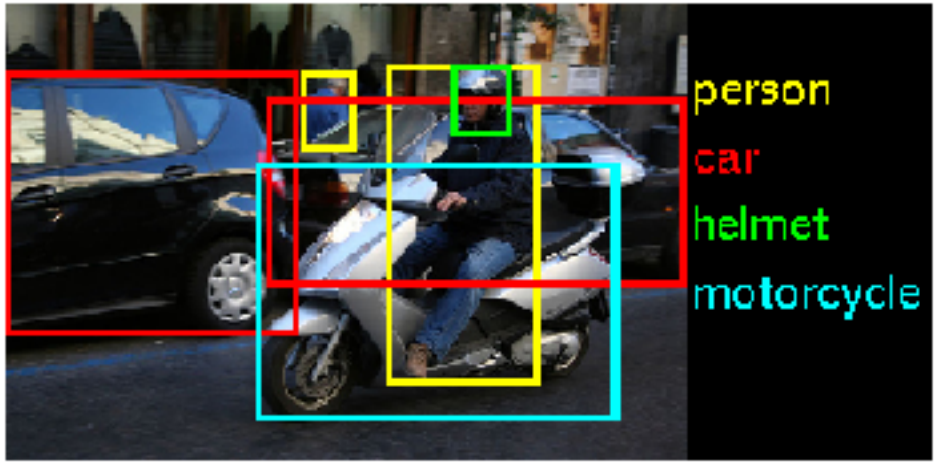
아래의 프로그램에서 N2MT를 사용 중 입니다.

- Papago 통역앱 (모바일 번역앱)
- Labspace NMT : http://labspace.naver.com/nmt/

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

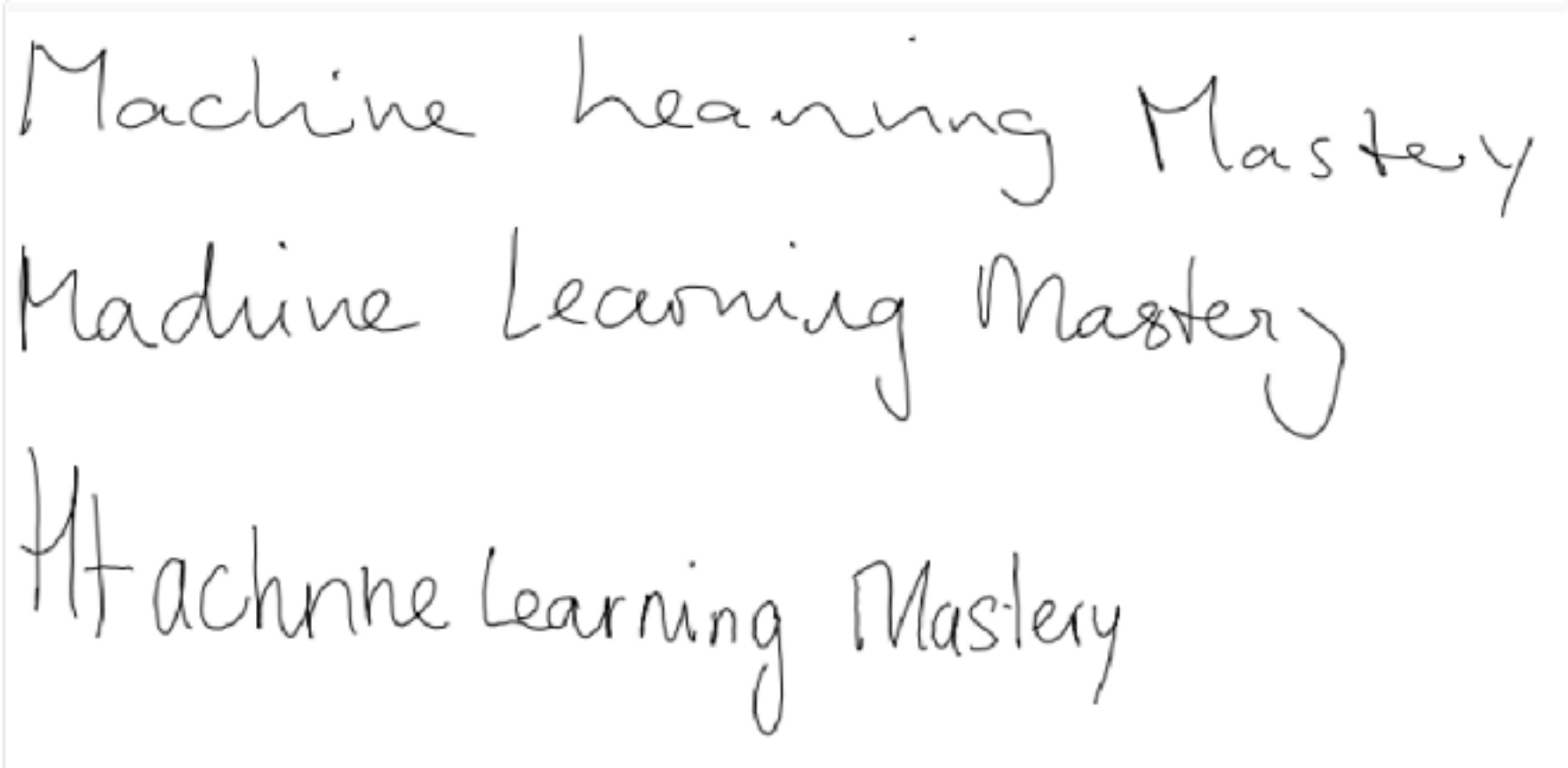
4. Object Classification and Detection in Photographs





ImageNet Classification with Deep Convolutional Neural Networks

5. Automatic Handwriting Generation



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

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/

7. Automatic Image Caption Generation

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

8. Automatic Game Playing

https://youtu.be/Tr

Basics of Neural Networks

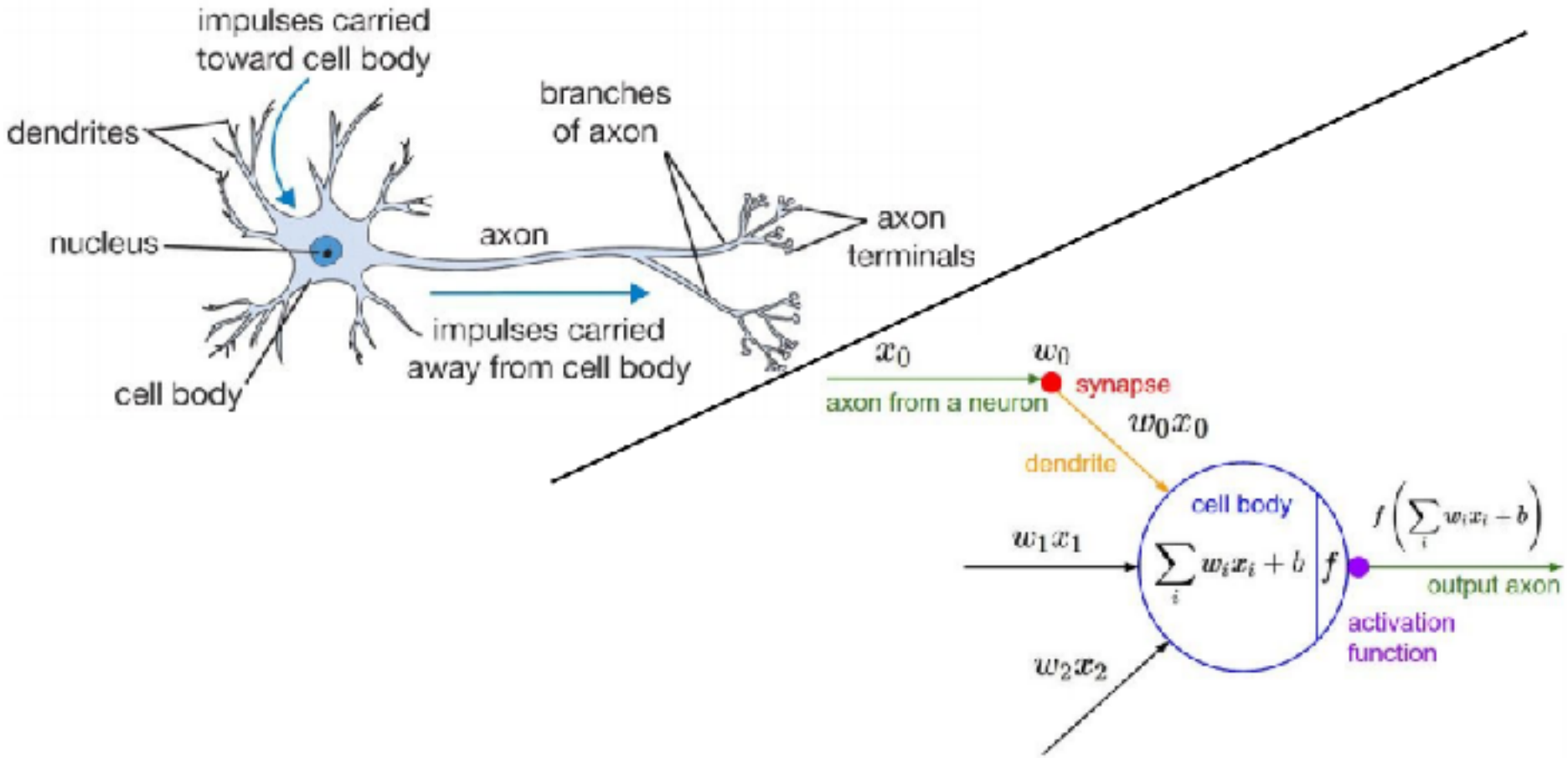
Classification Problem

- 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)

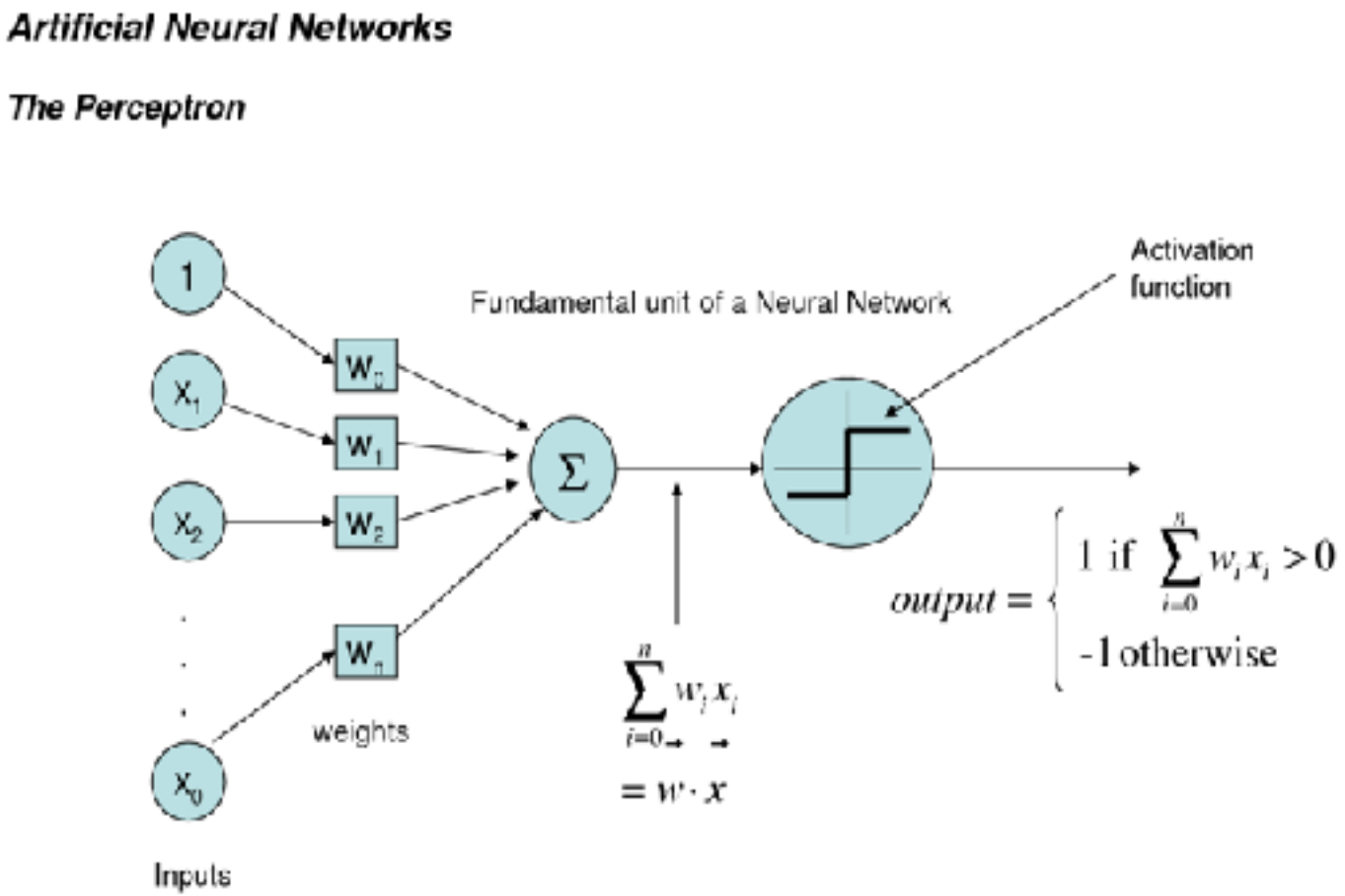
- 대표적인 패턴 인식 알고리즘
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbor
  - Multi-Layer Perceptron (Artificial Neural Network; 인공신경망)

Perceptron (1/3)

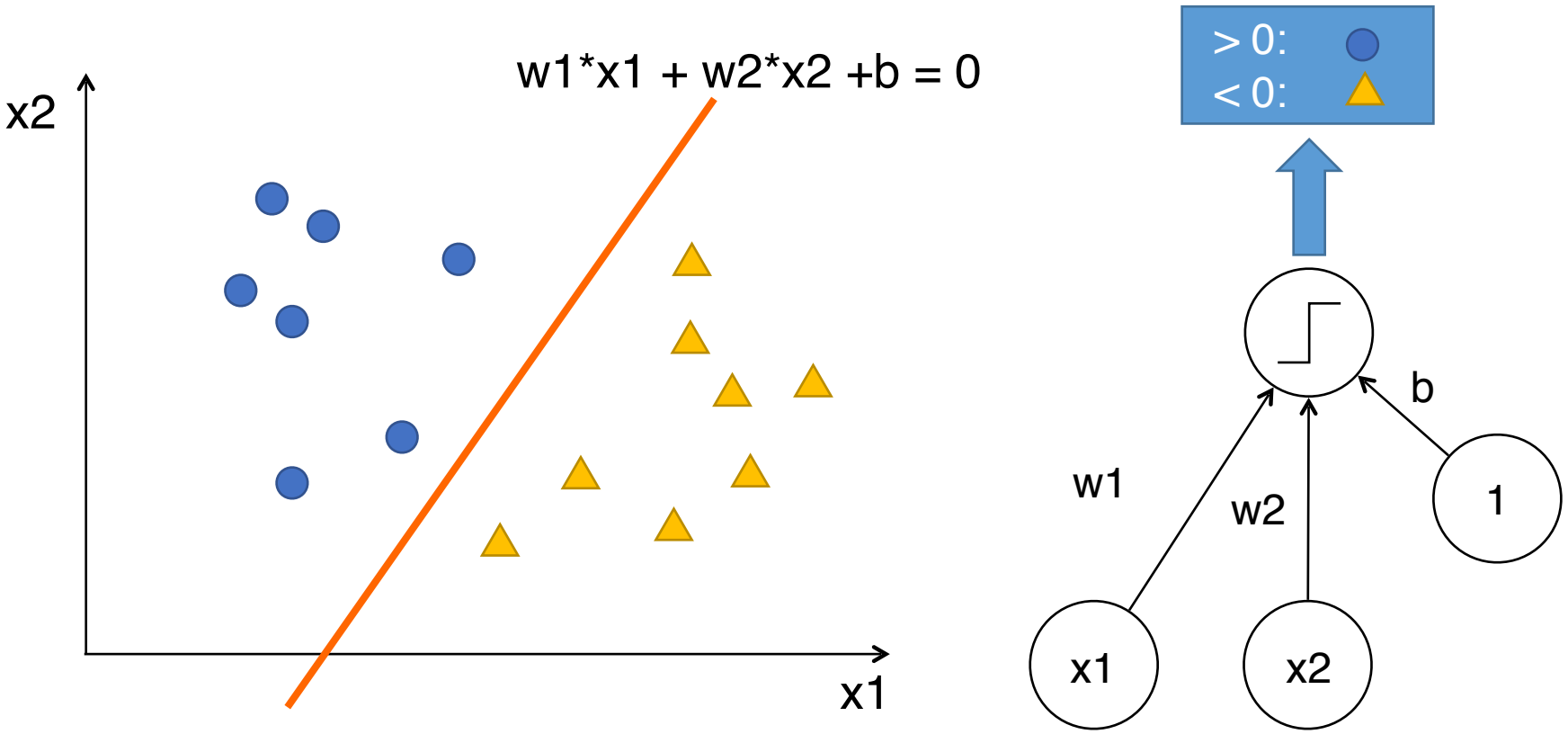


http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

Perceptron (2/3)

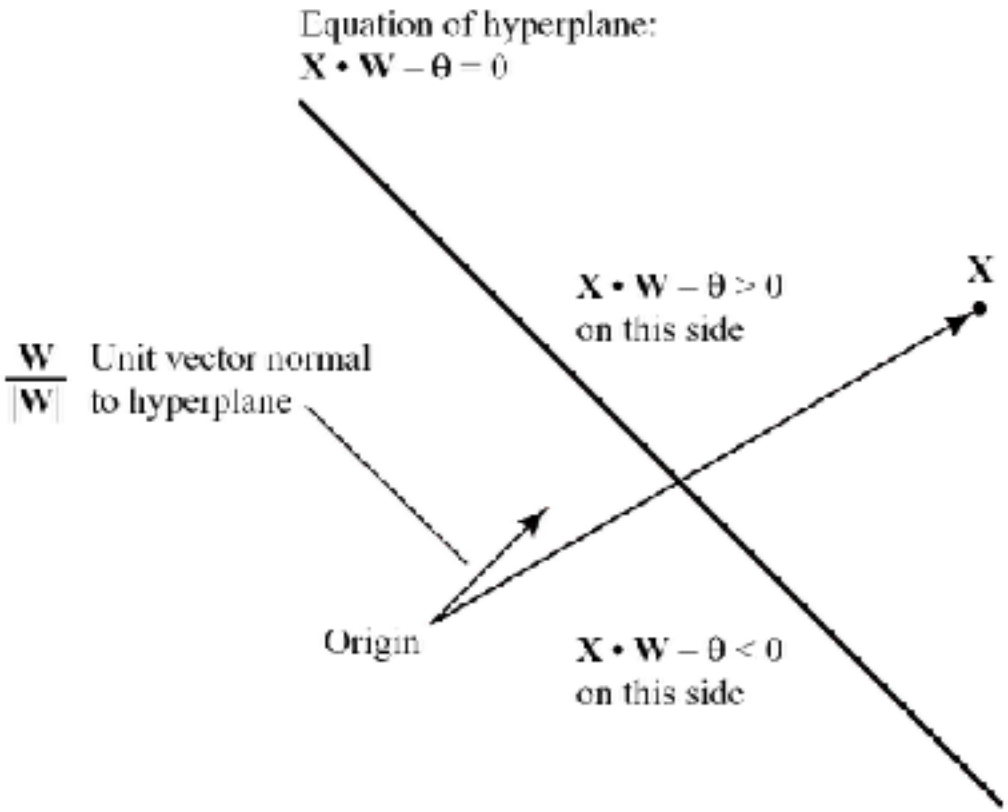


Perceptron (3/3)

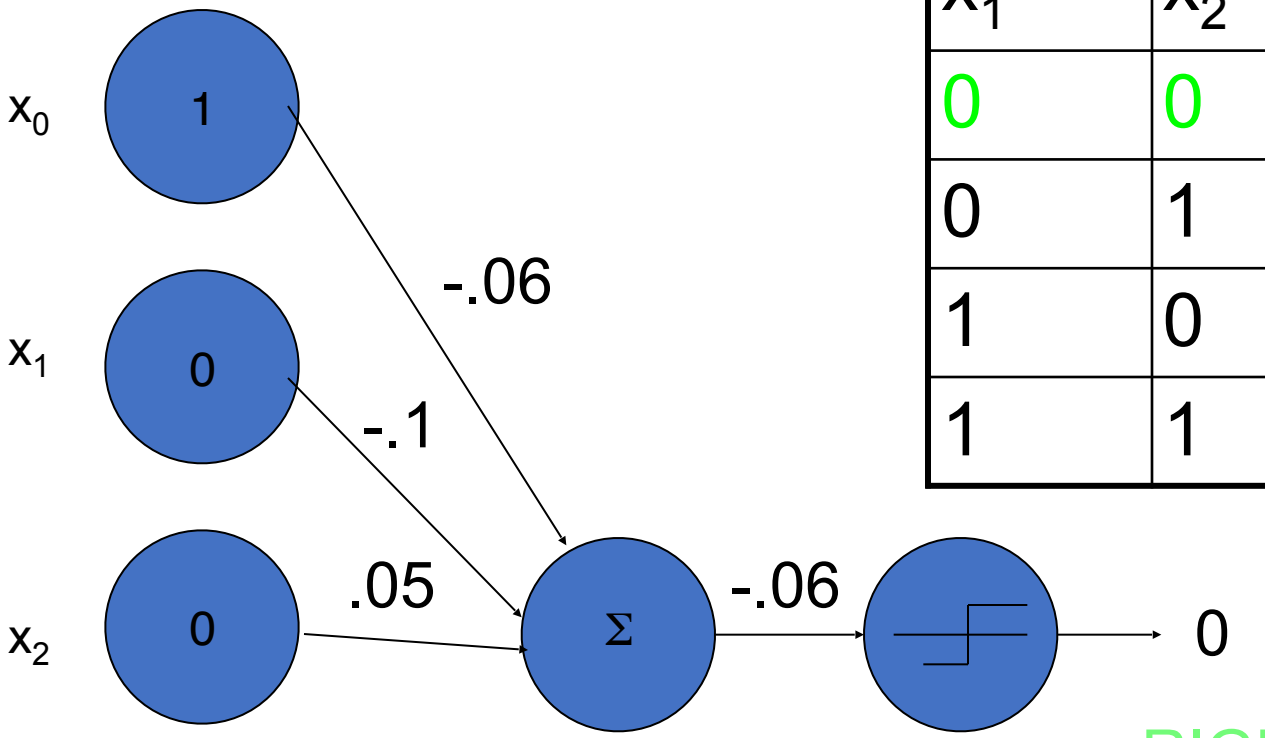


Parameter Learning in Perceptron

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



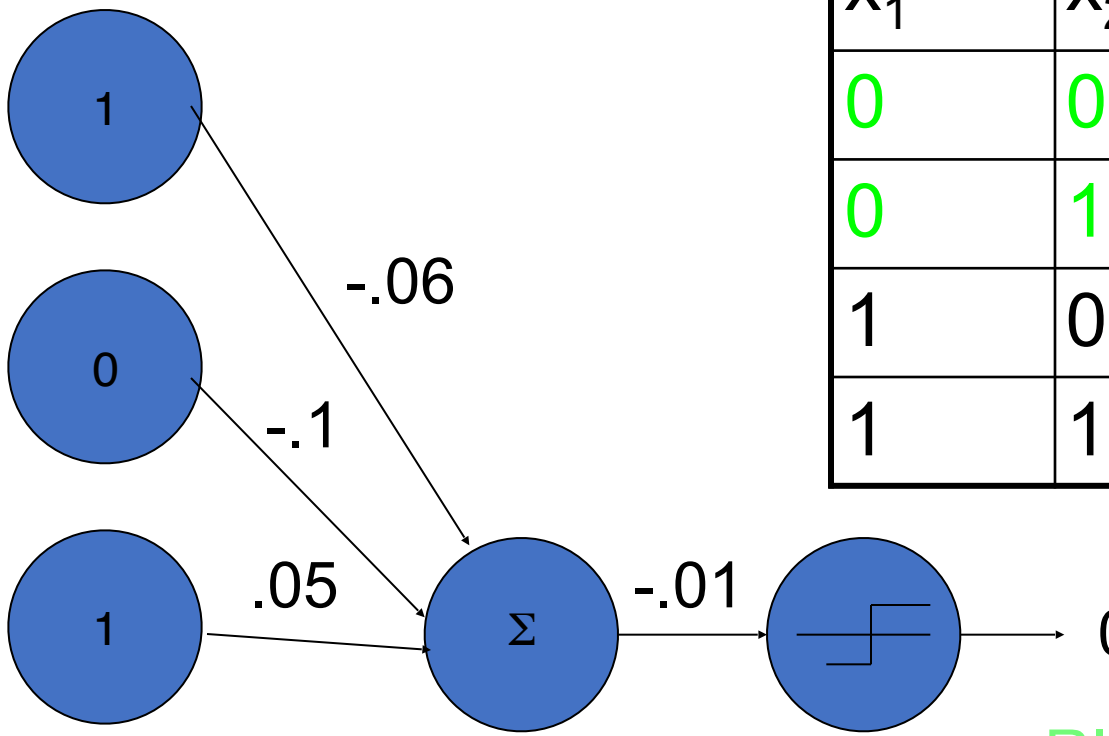
Perceptrons



x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

RIGHT

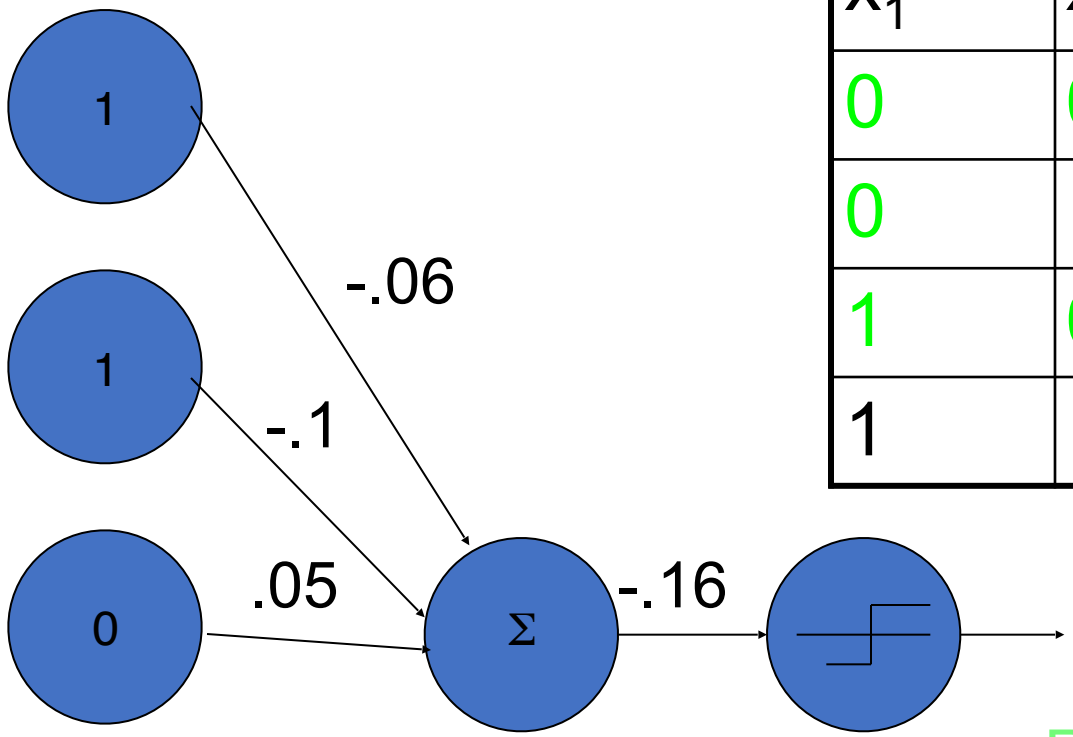
Perceptrons



x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

RIGHT

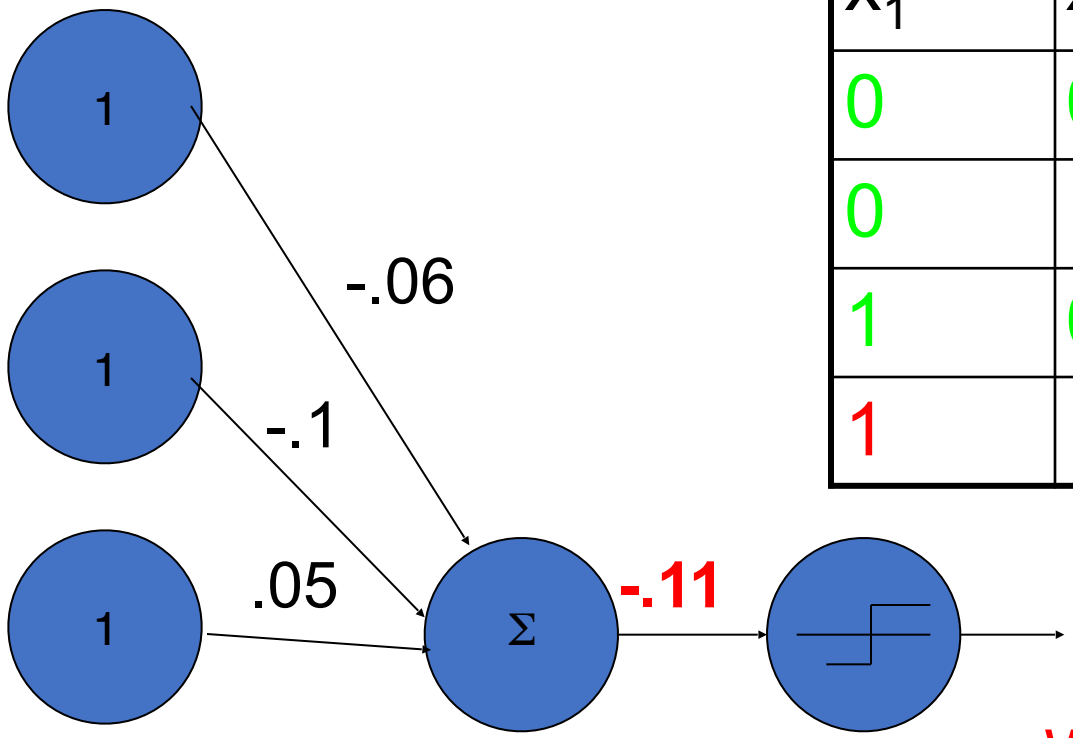
Perceptrons



x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

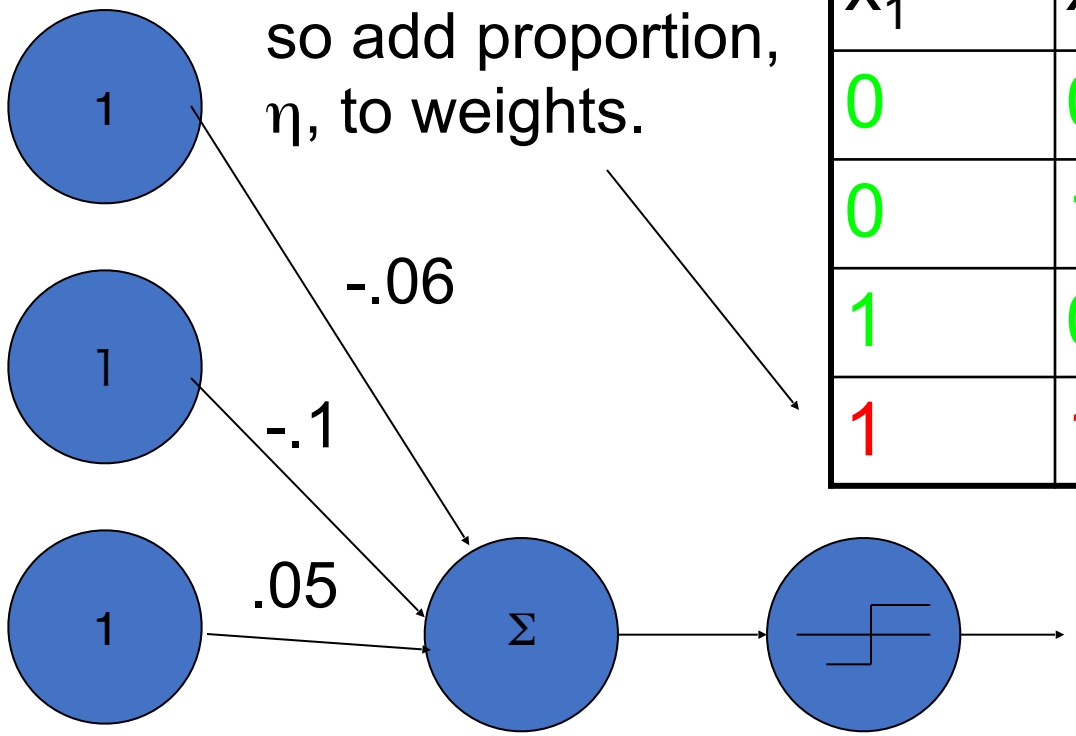
RIGHT

Perceptrons



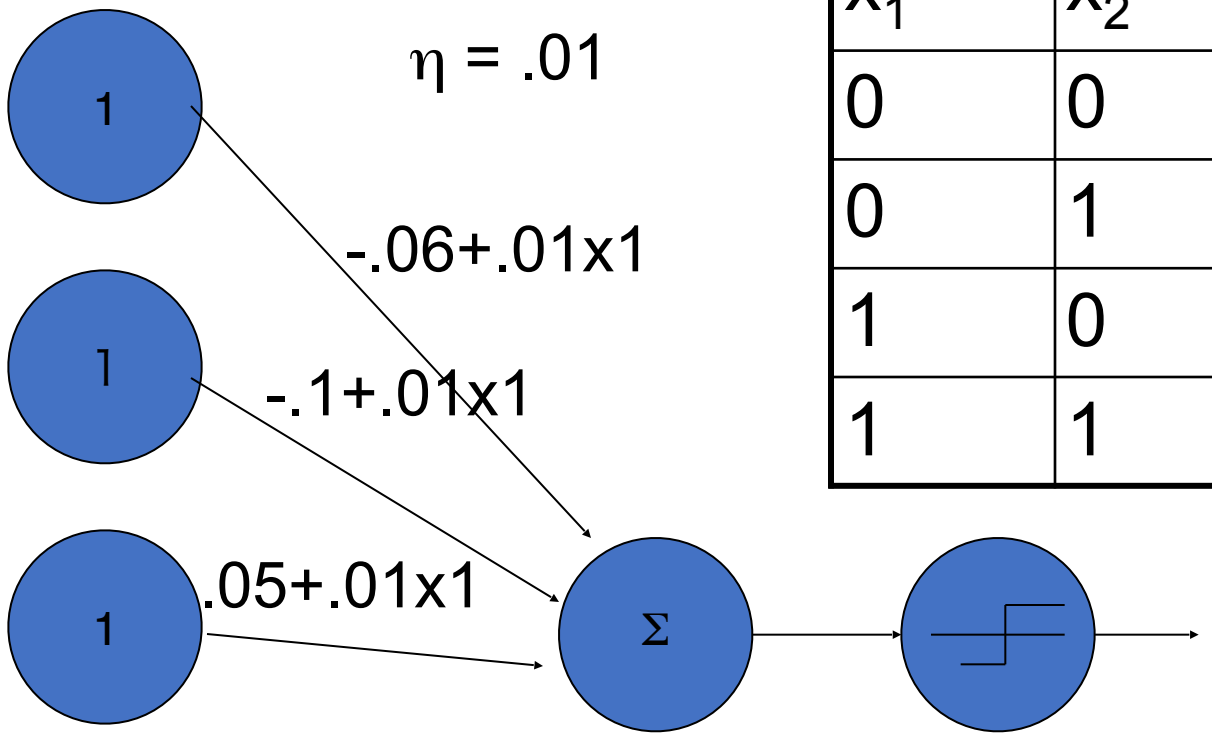
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



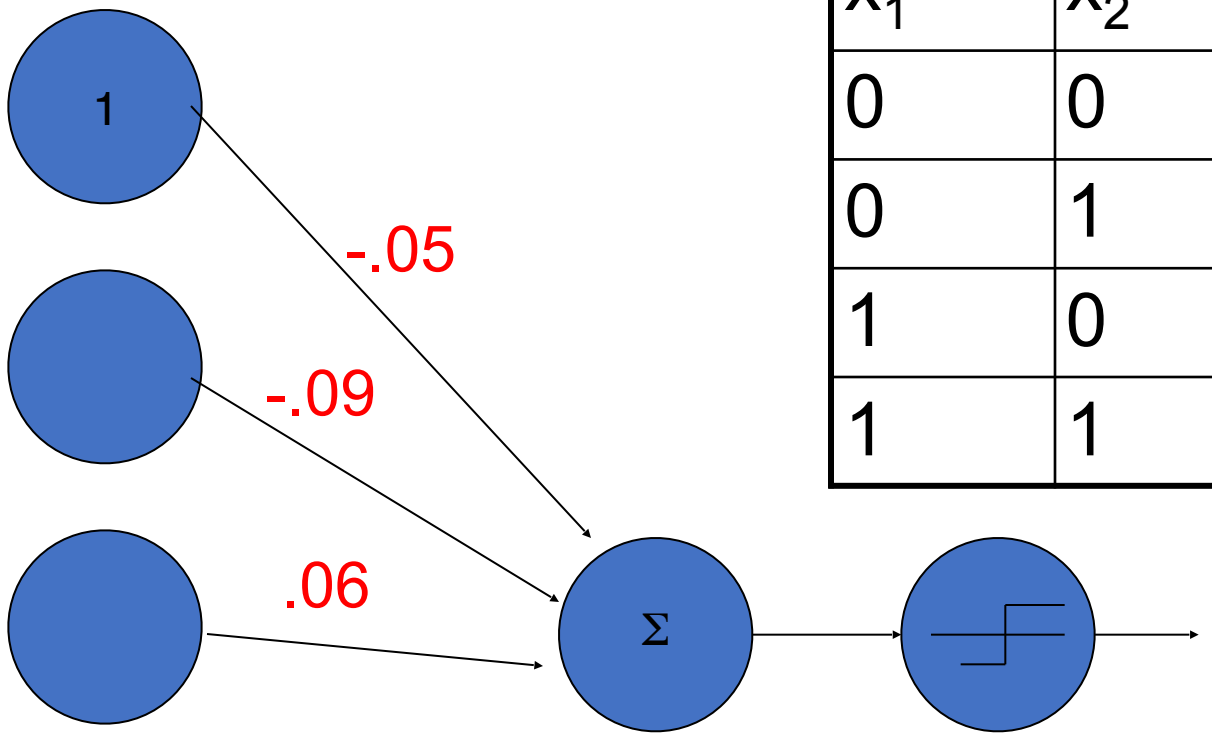
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



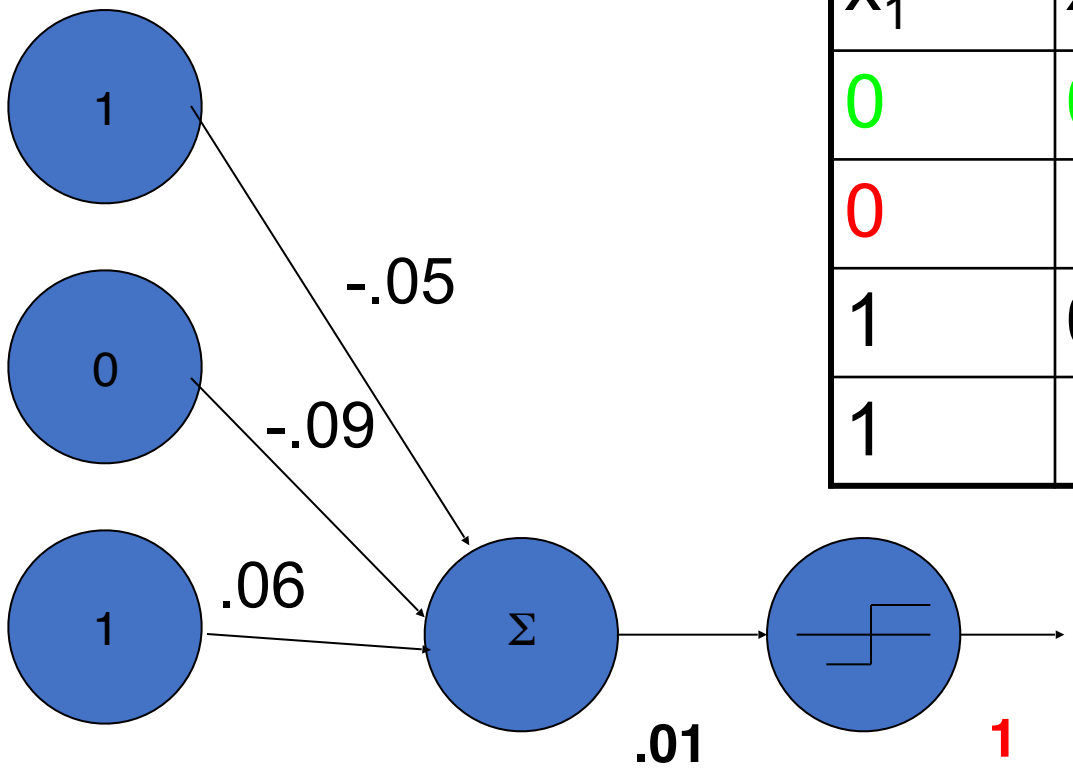
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



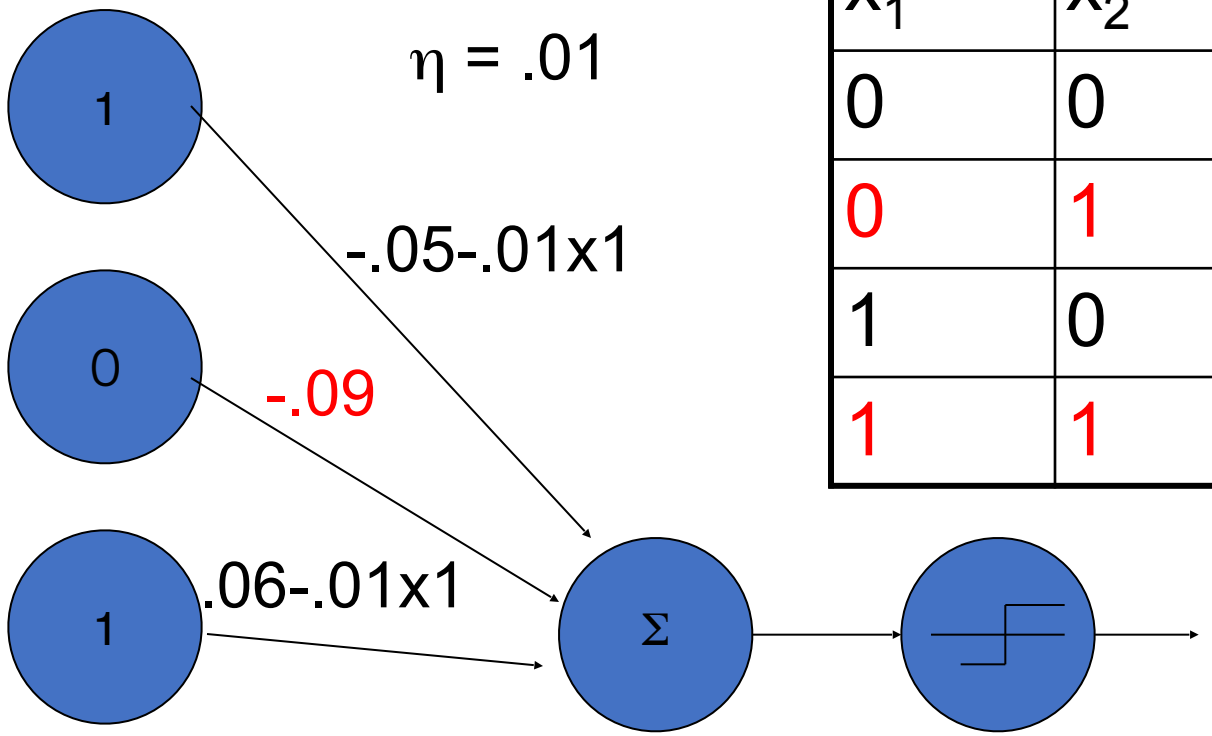
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



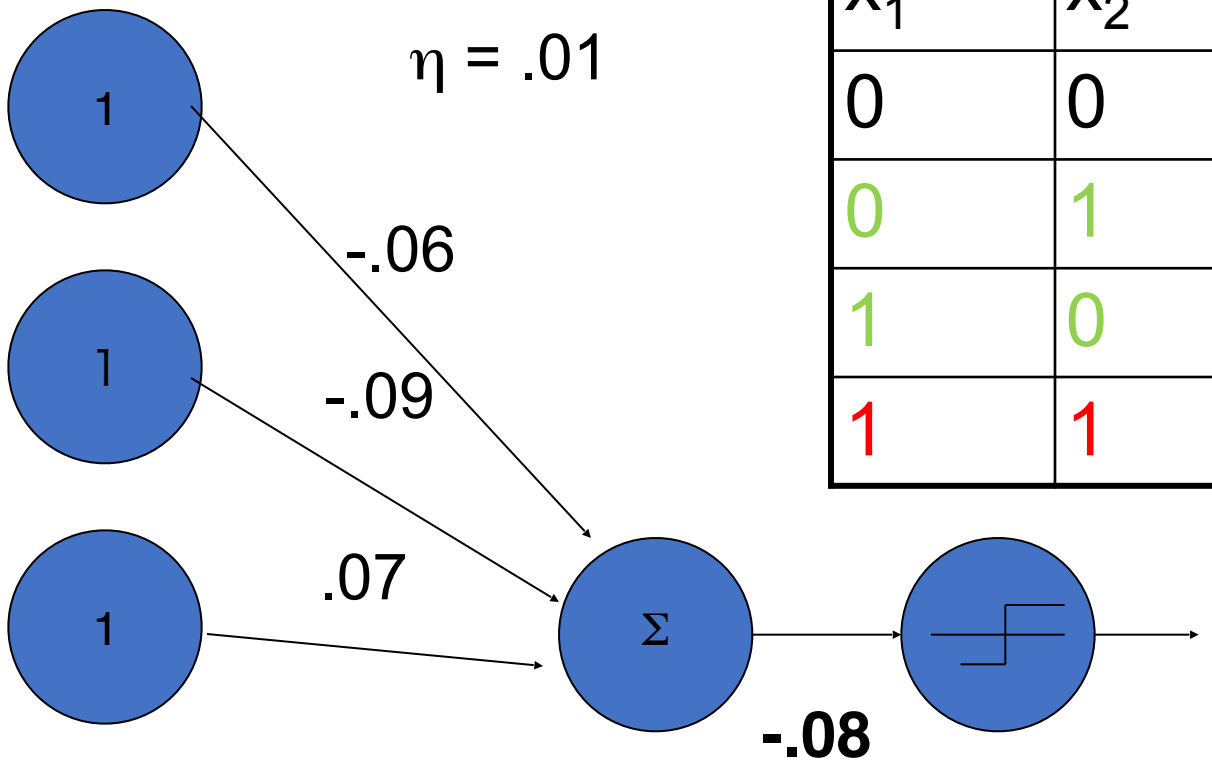
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



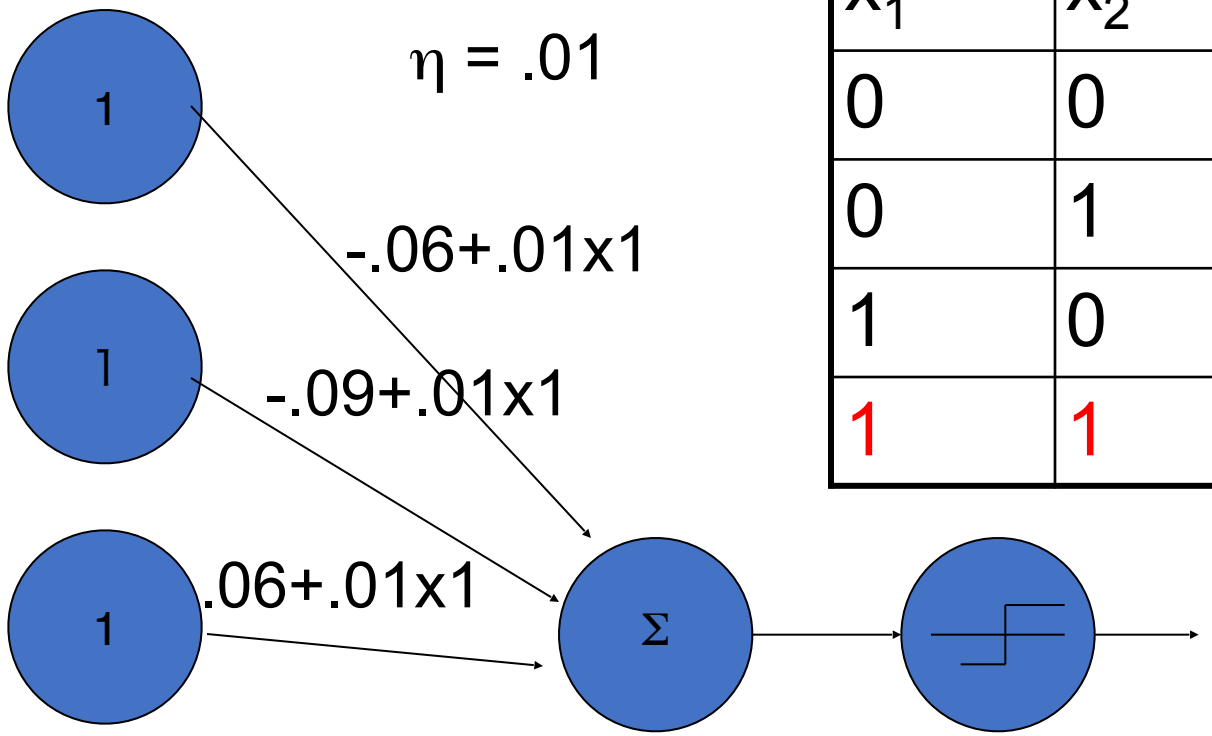
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



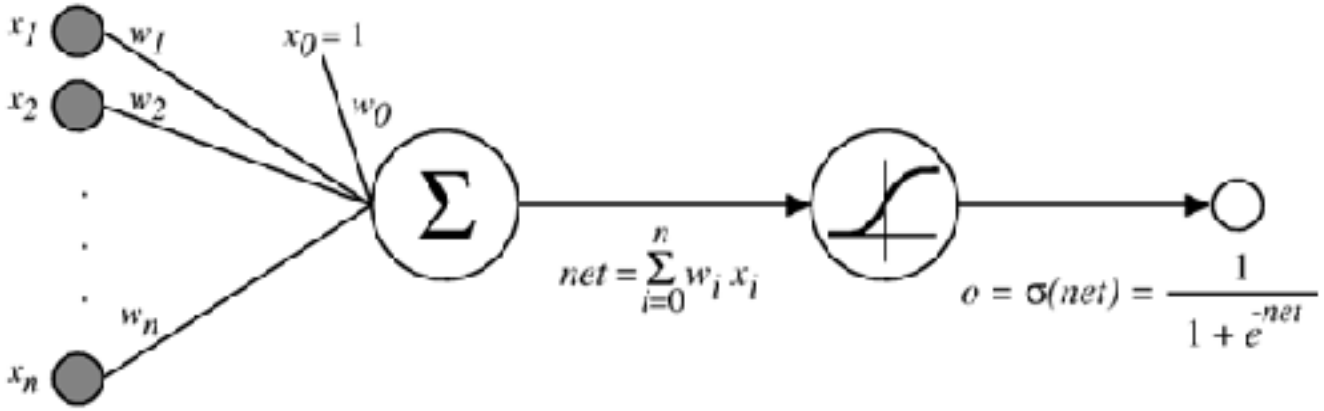
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons



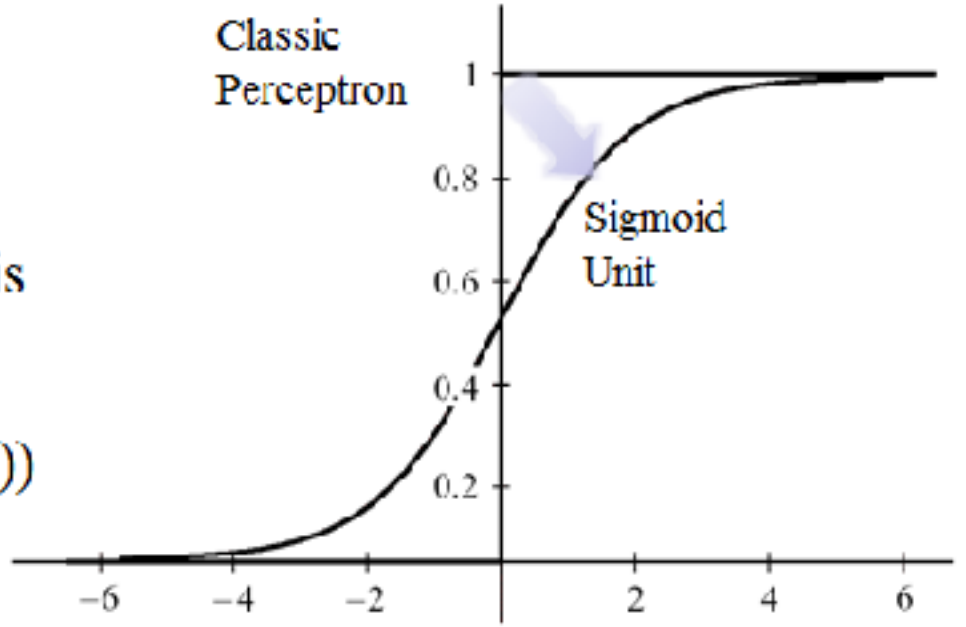
x_1	x_2	o
0	0	0
0	1	0
1	0	0
1	1	1

Sigmoid Unit



Sigmoid function is **Differentiable**

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



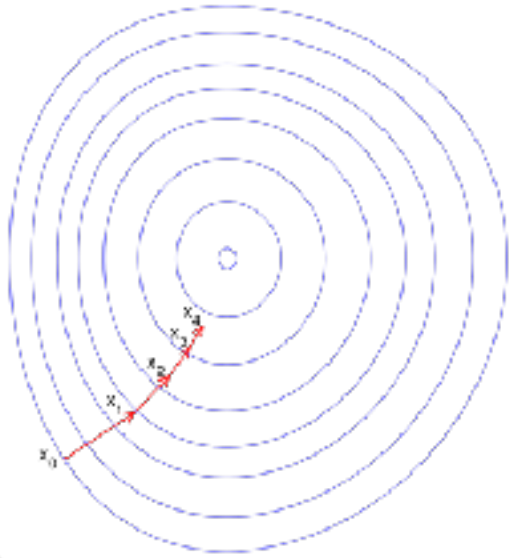
Learning Algorithm of Sigmoid Unit

- Loss Function

Target, Unit Output

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

- Gradient Descent Update



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

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

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

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

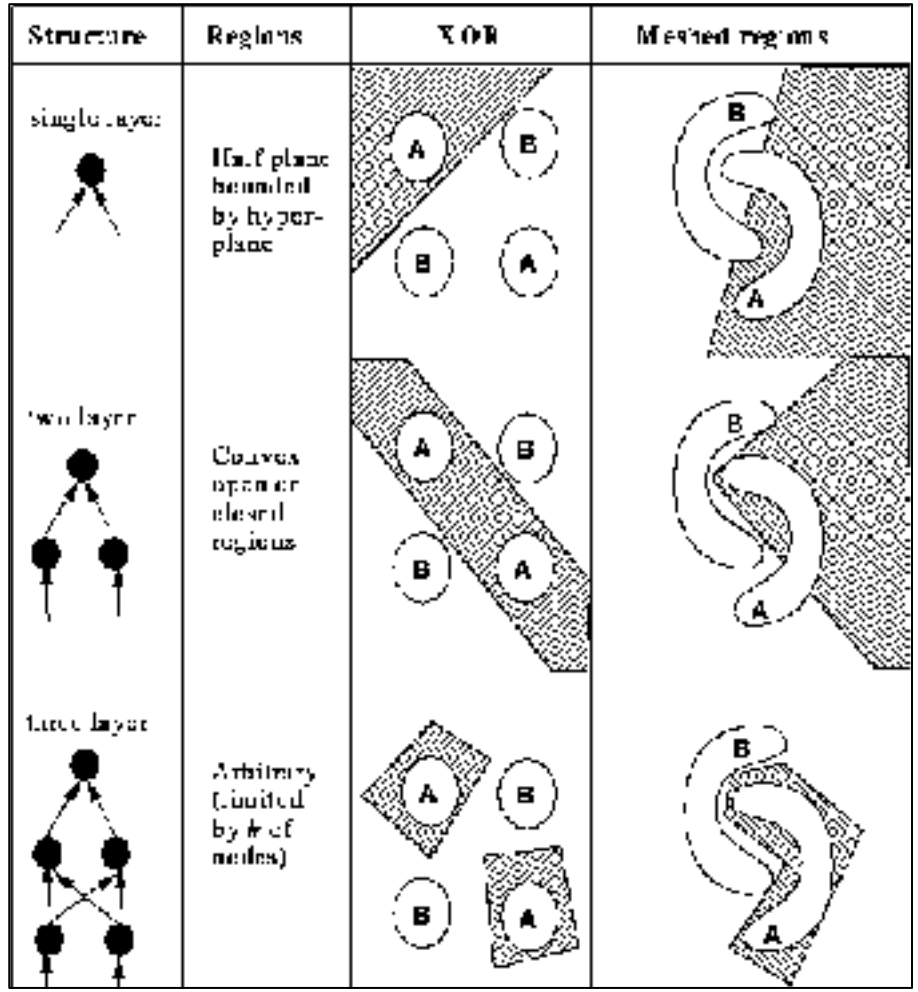
Need for Multiple Units and Multiple Layers

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

→ Multiple Units

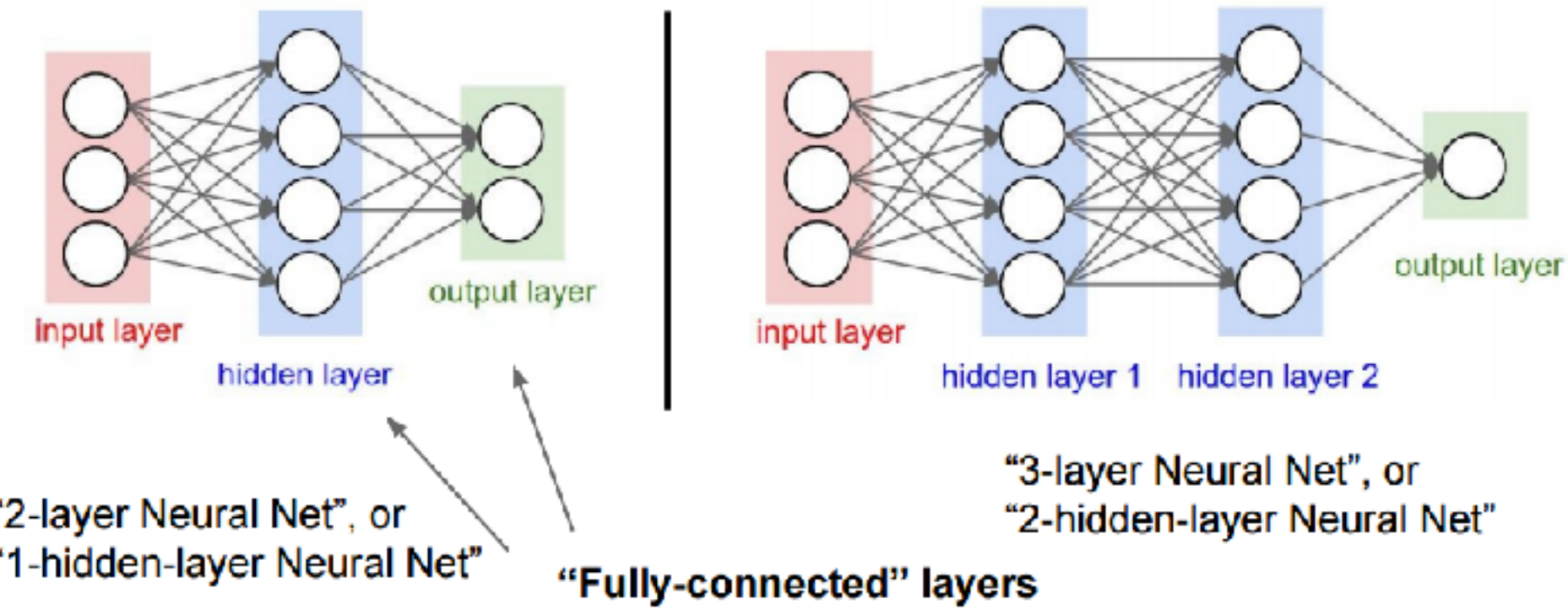
- More complex regions are needed (e.g. Polygons)

→ Multiple Layers



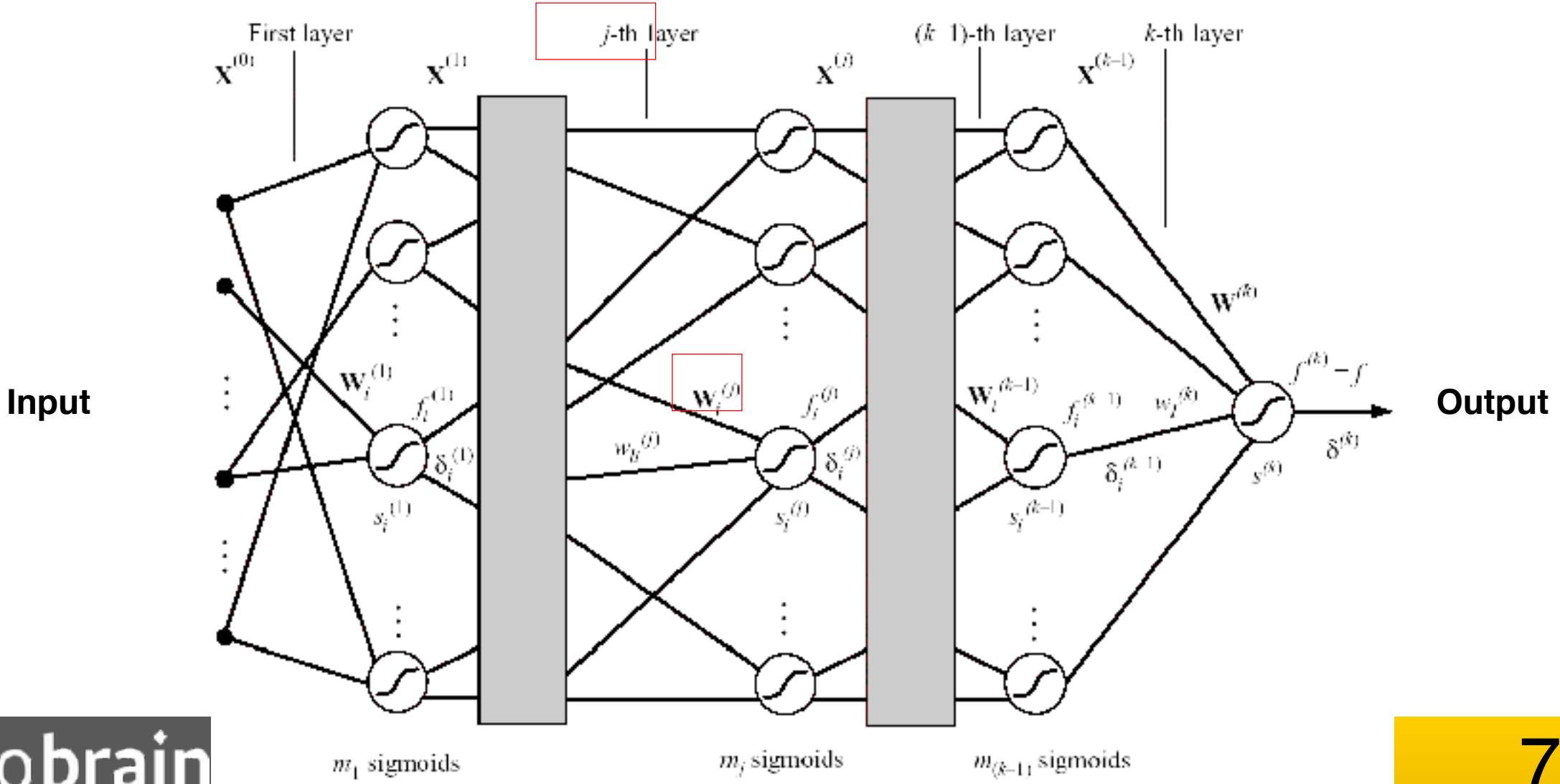
http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Structure of Multilayer Perceptron



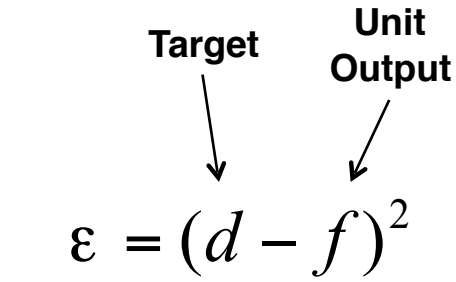
http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

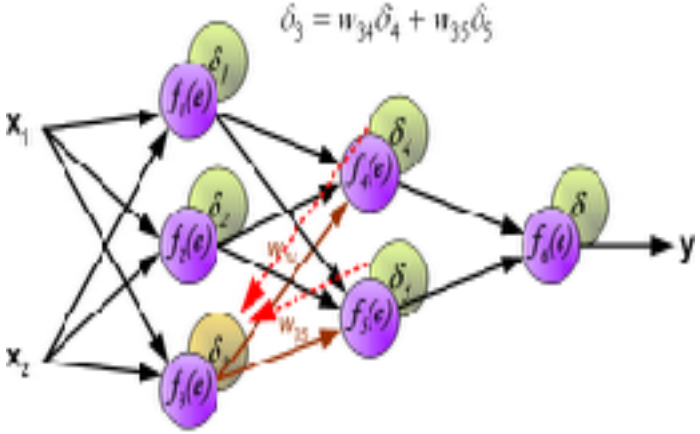
Structure of Multilayer Perceptron (MLP; Artificial Neural Network)



Learning Parameters of MLP

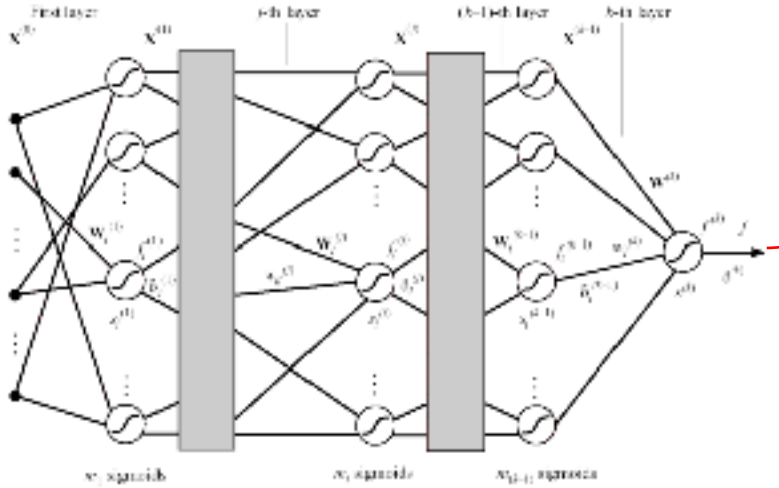
- Loss Function
  - We have the same Loss Function
  - But the # of parameters are now much more (Weight for **each layer** and **each unit**)
  - To use Gradient Descent, we need to calculate the **gradient for all the parameters**
- Recursive Computation of Gradients
  - Computation of loss-gradient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



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

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

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

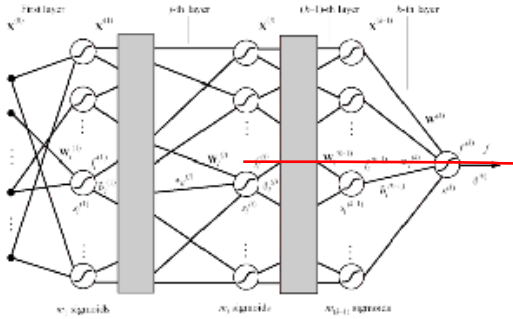
- Store intermediate value delta for later use of chain rule

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

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

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

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

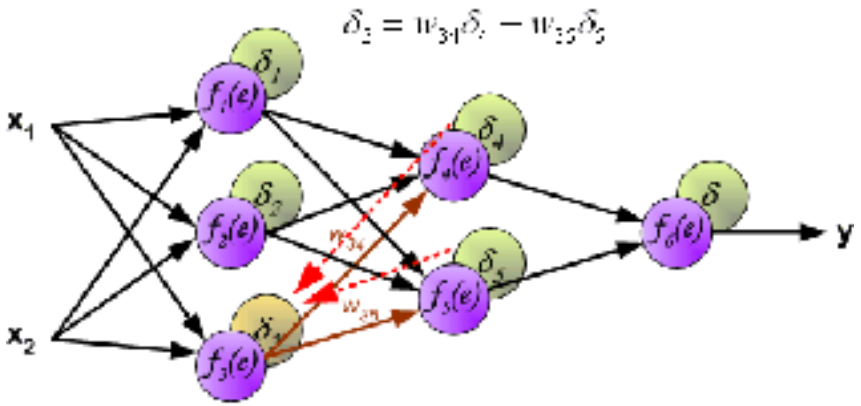
Gradient Descent Update rule for lower-layer weights

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

Back Propagation Learning Algorithm (3/3)

- Applying chain rule, recursive relation between delta's

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

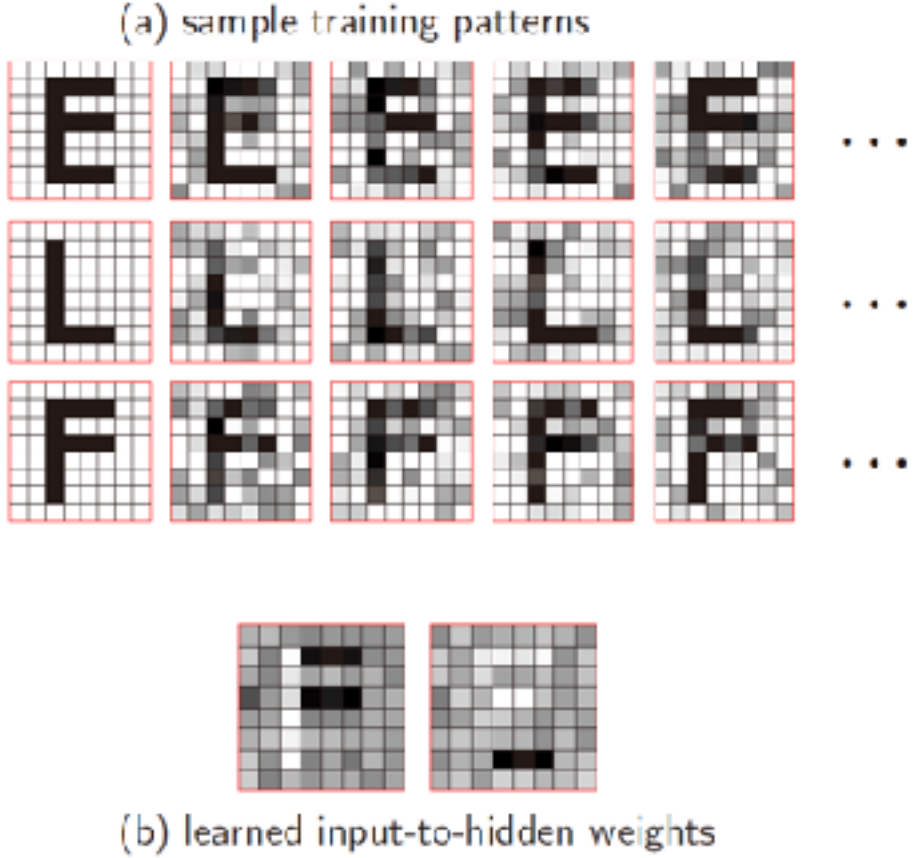


Algorithm: Back Propagation

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

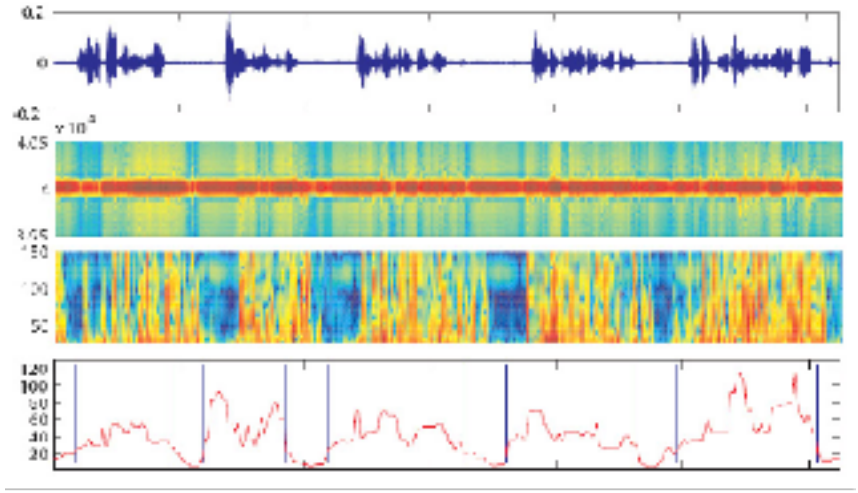


- 64-2-3 network for classifying 3 characters
  - 64-dim inputs
  - 2 hidden units
  - 3 output units

- Learned i-to-h weights
  - Describe feature groupings useful for classification

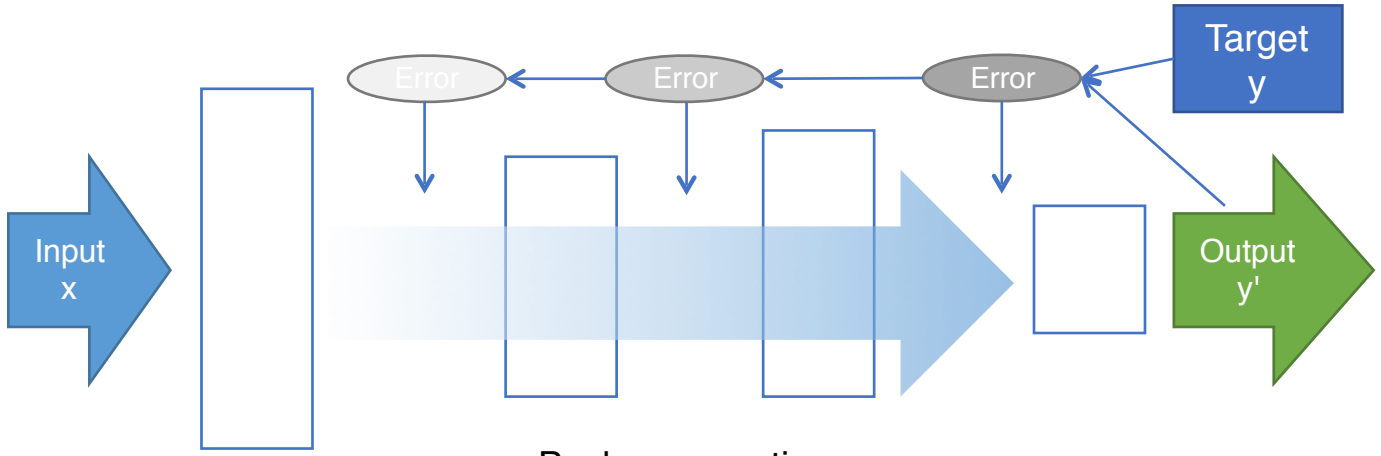
Applications

- 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

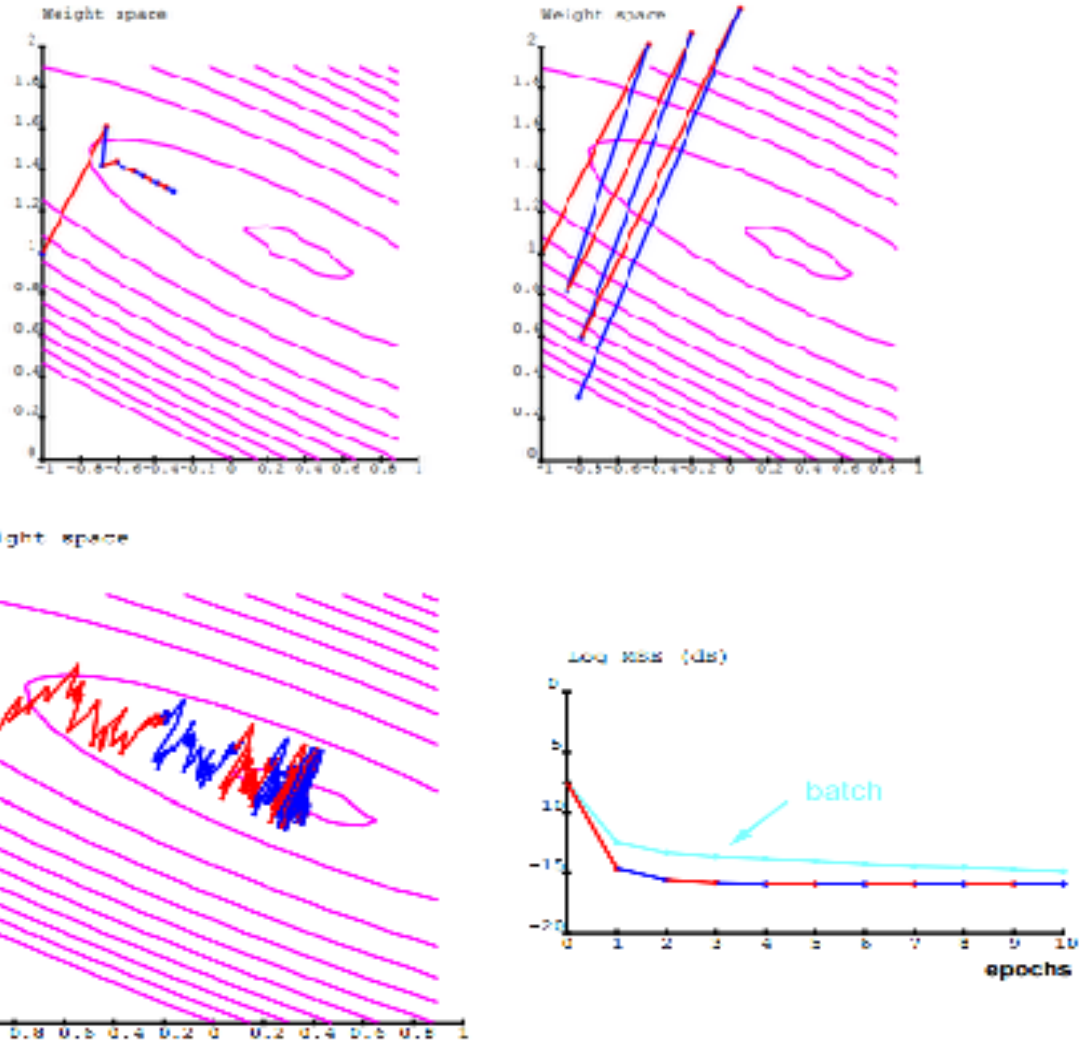


Back-propagation

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

Some Issues (1/3)

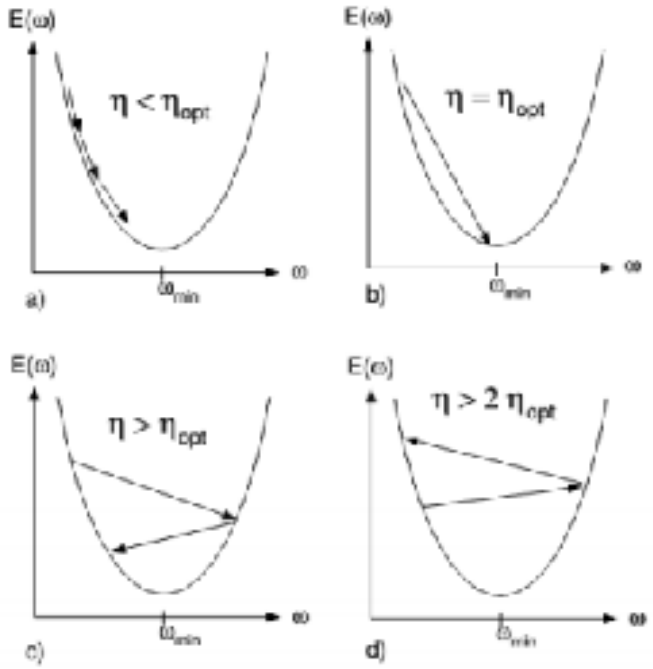
- Stochastic Gradient Descent

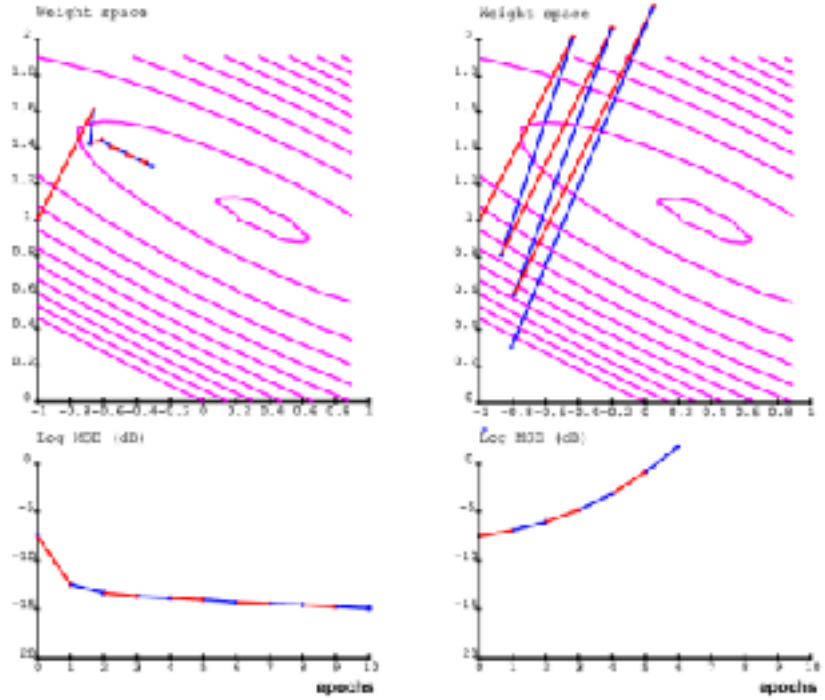


http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

Some Issues (2/3)

- Learning Rate Adaptatic
- Momentum
- Weight Decay



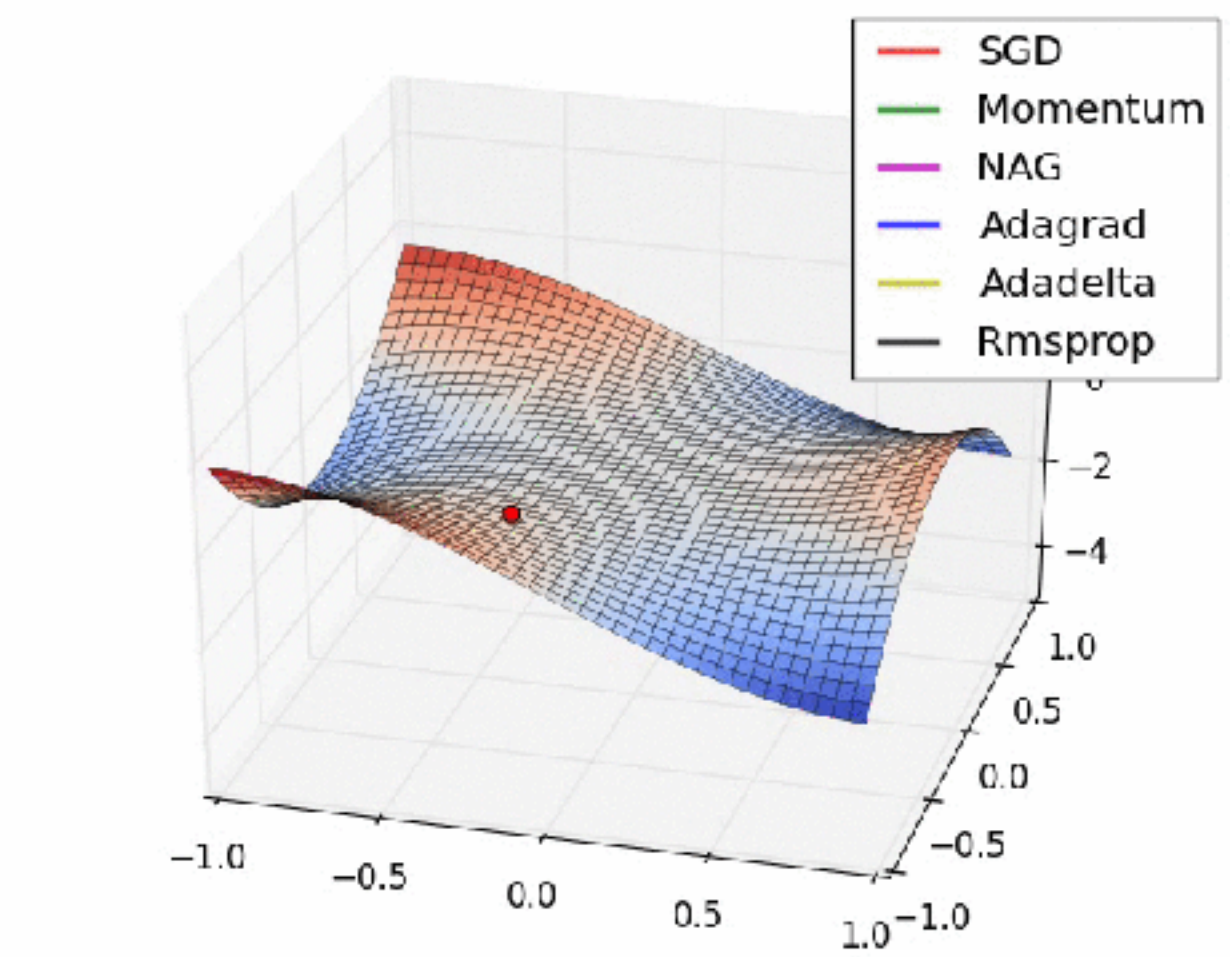


http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

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

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

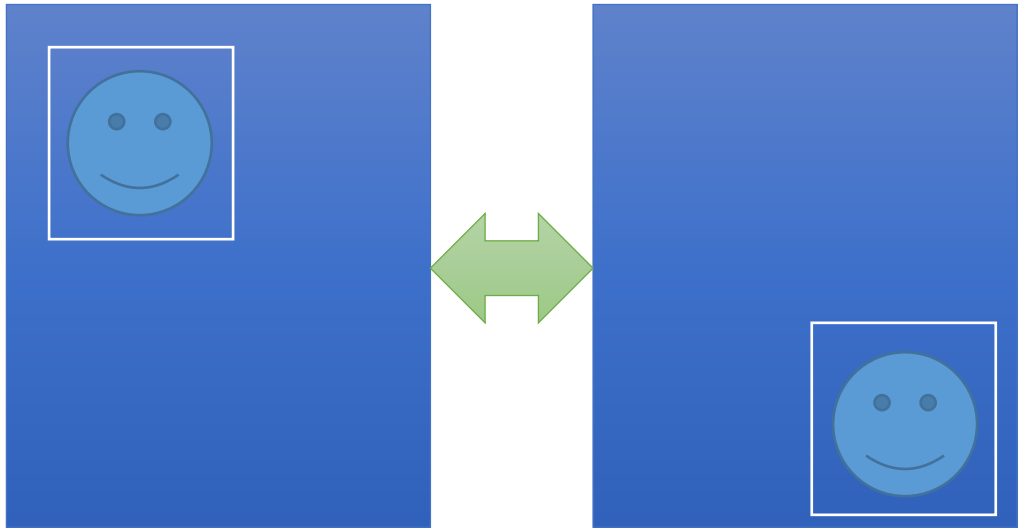
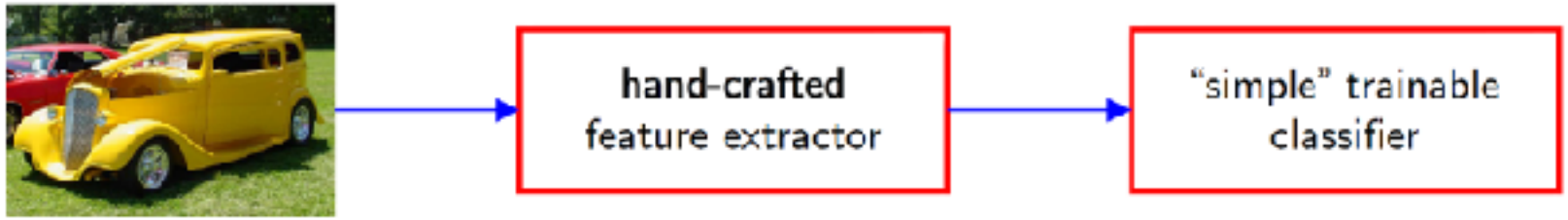


Image 1 Image 2

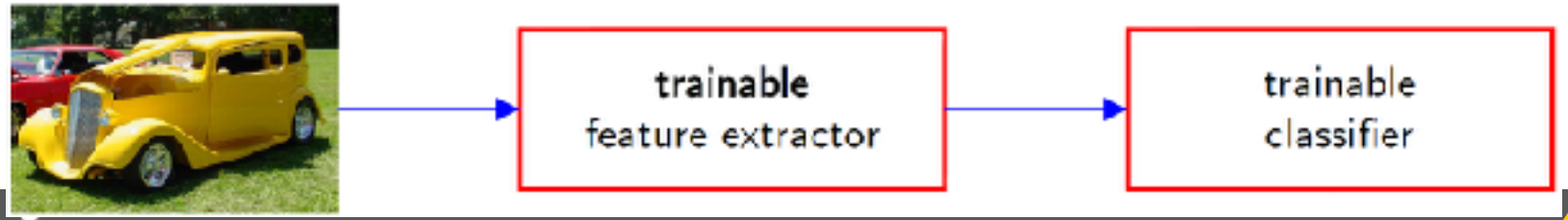
- DBN의 경우 다른 data
- CNN의 경우 같은 data

End-to-End training

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

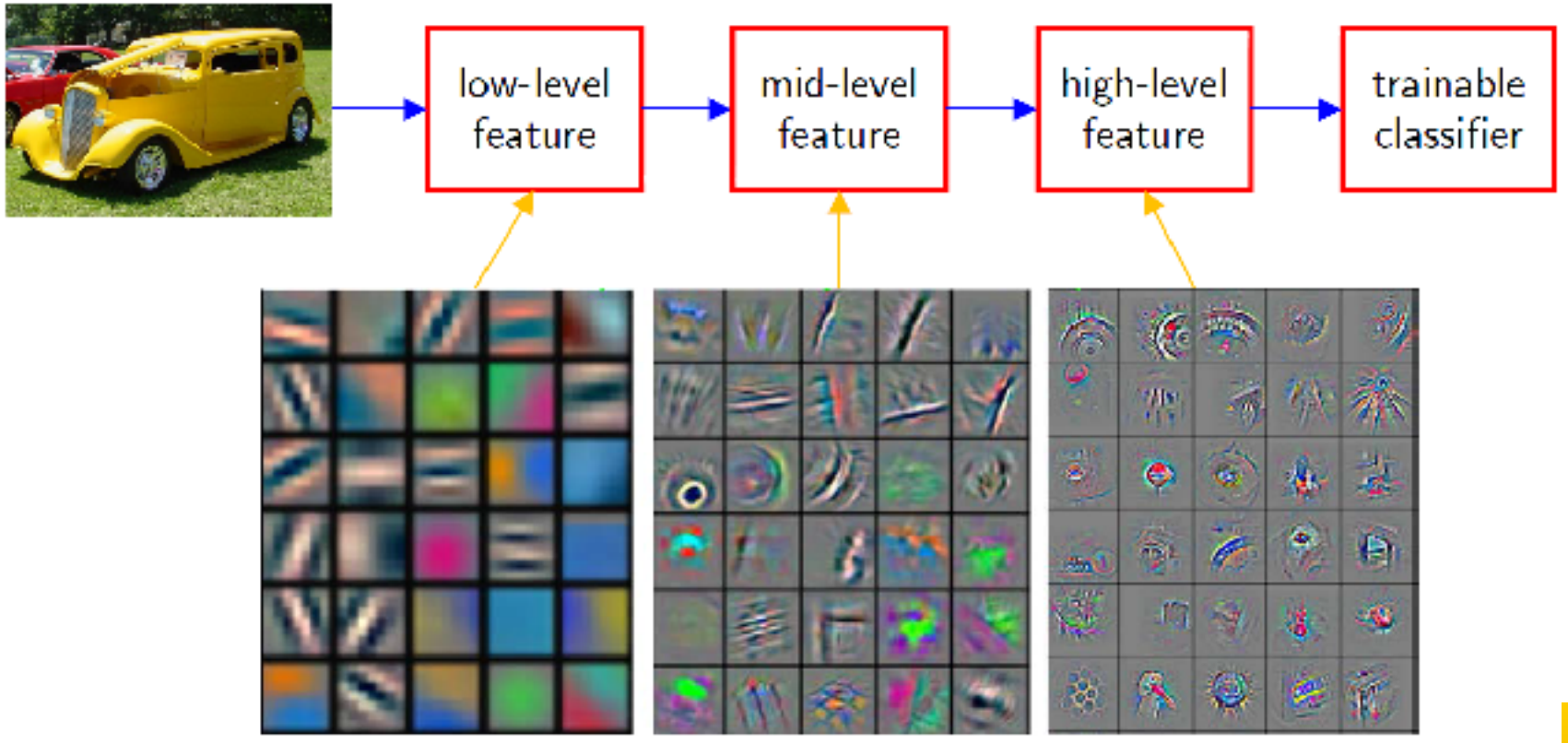


- End-to-end learning / feature learning / deep learning
  - Trainable features (or kernel) + trainable classifier



Convolutional Neural Networks

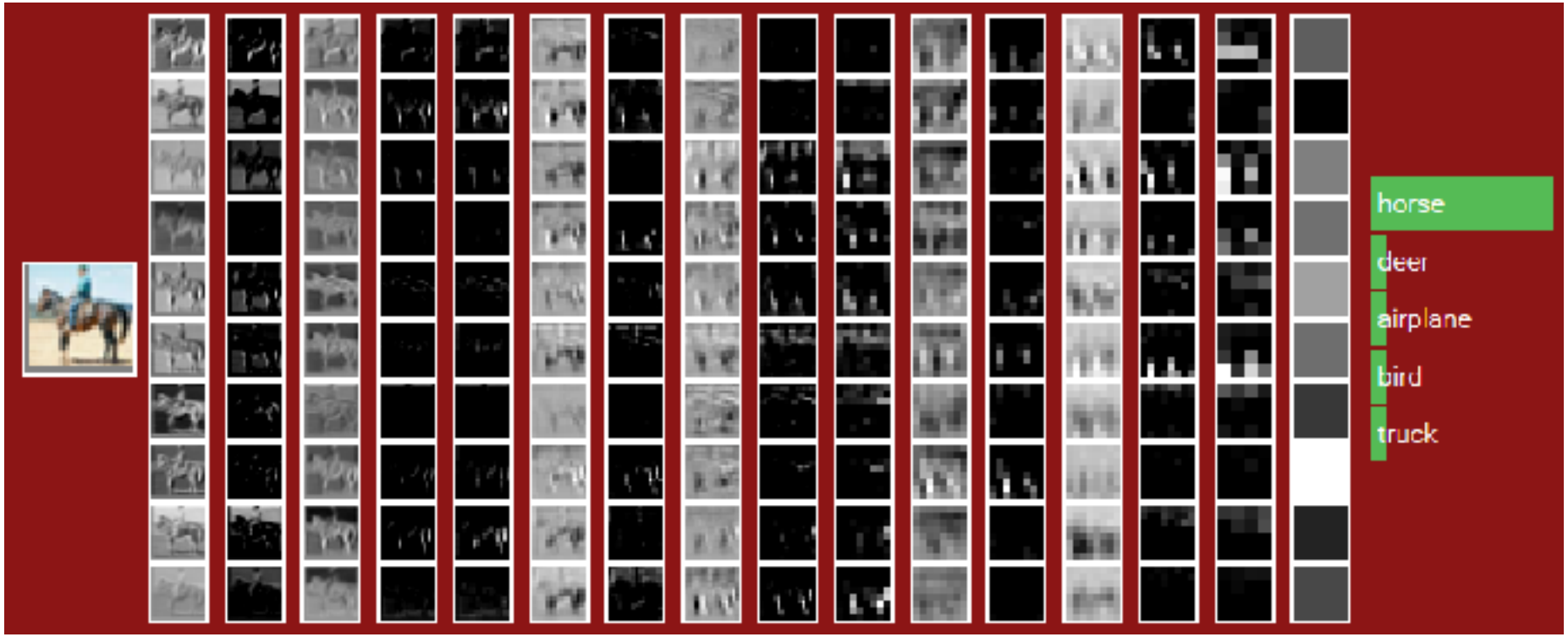
- Introduction
  - Learning hierarchical representations



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

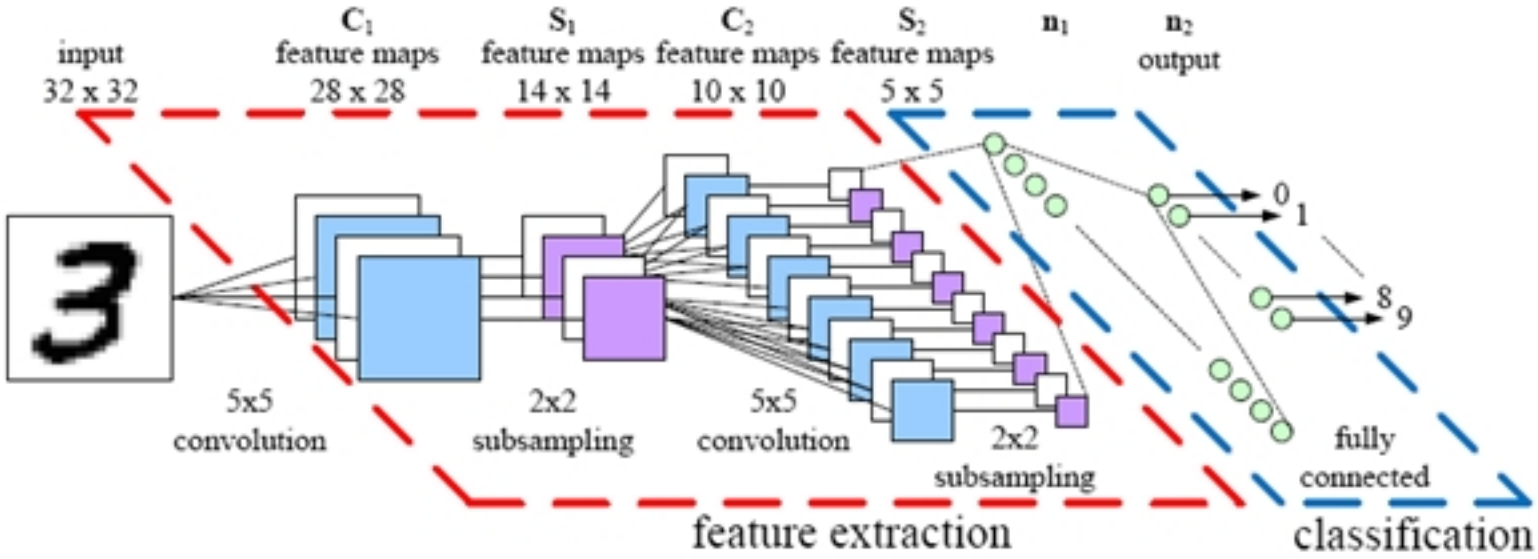
Convolutional Neural Networks

- Introduction
  - Learning hierarchical representations



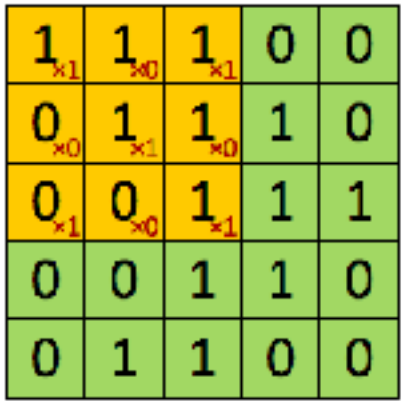
Structure of Convolutional Neural Network (CNN)

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

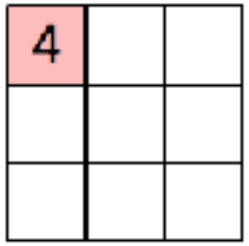


http://parse.ele.tue.nl/education/cluster2

Convolution Layer

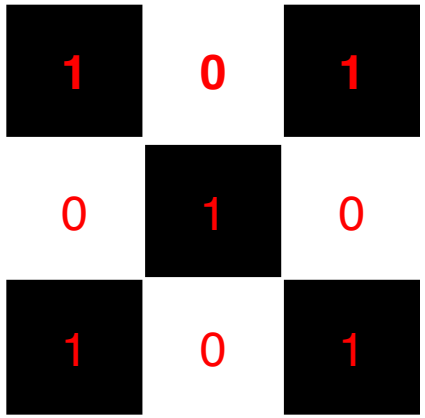


Image



Convolved Feature

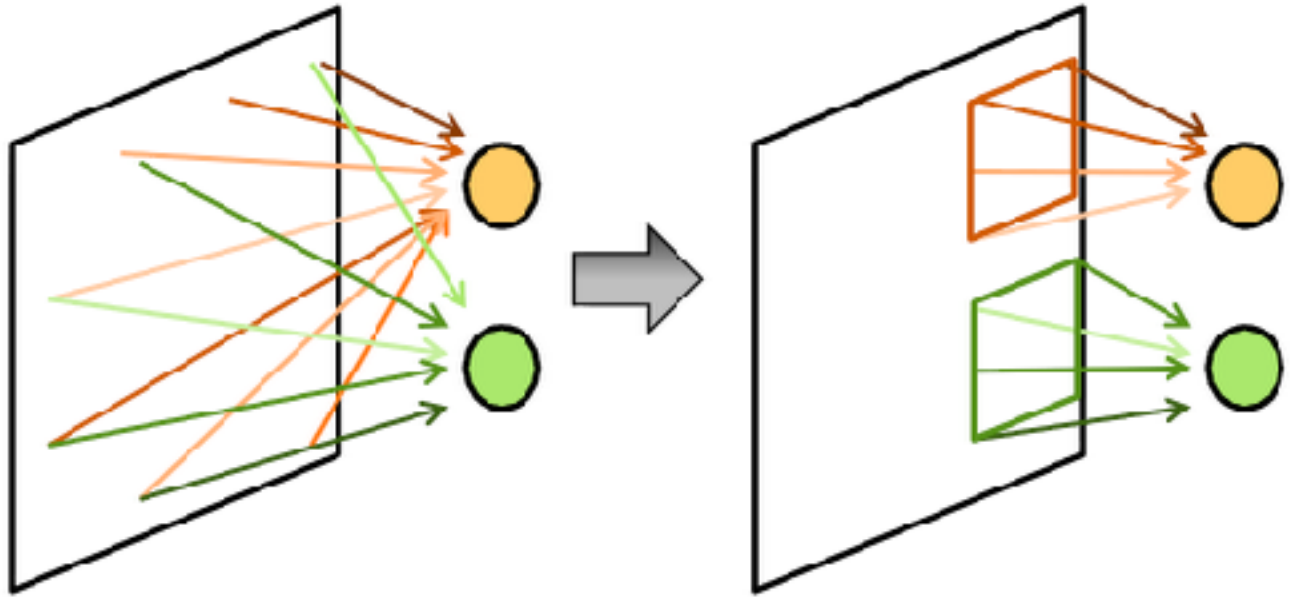
- The Kernel Detects pattern:



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

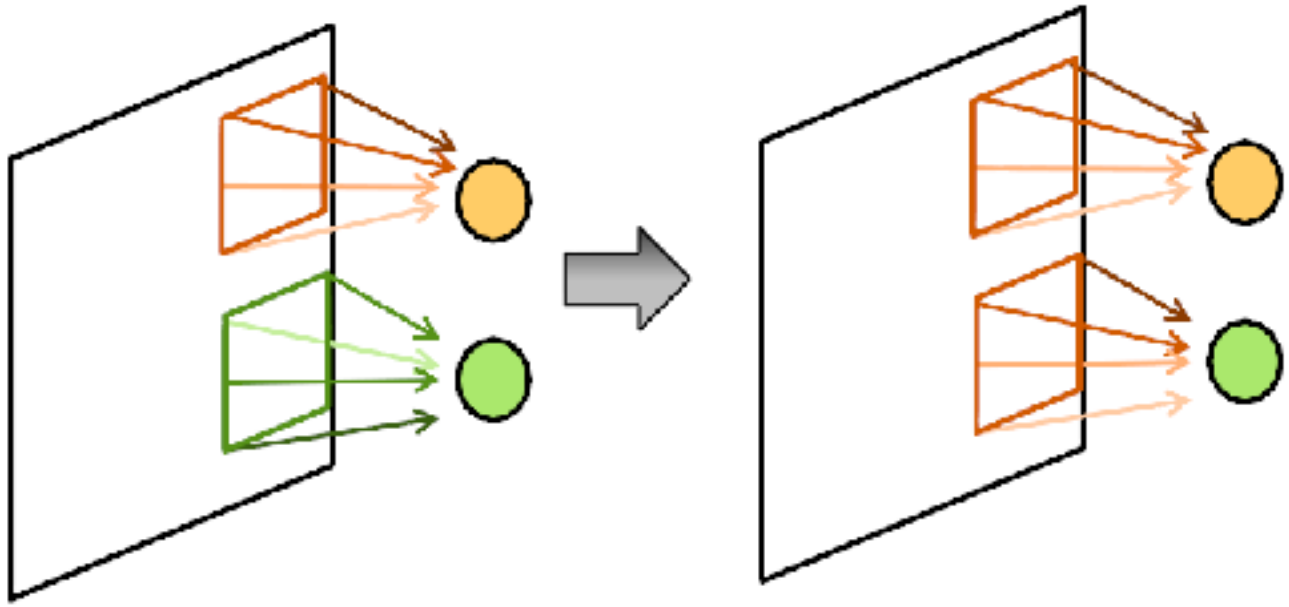
Convolutional Neural Networks

- 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



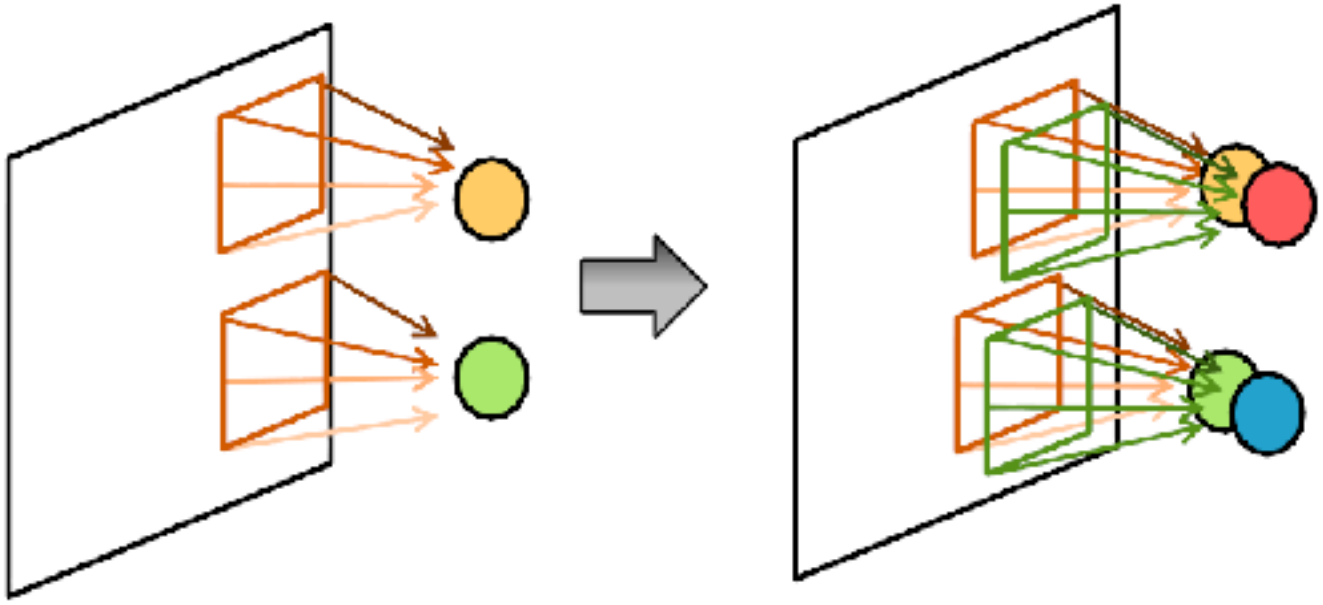
Convolutional Neural Networks

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



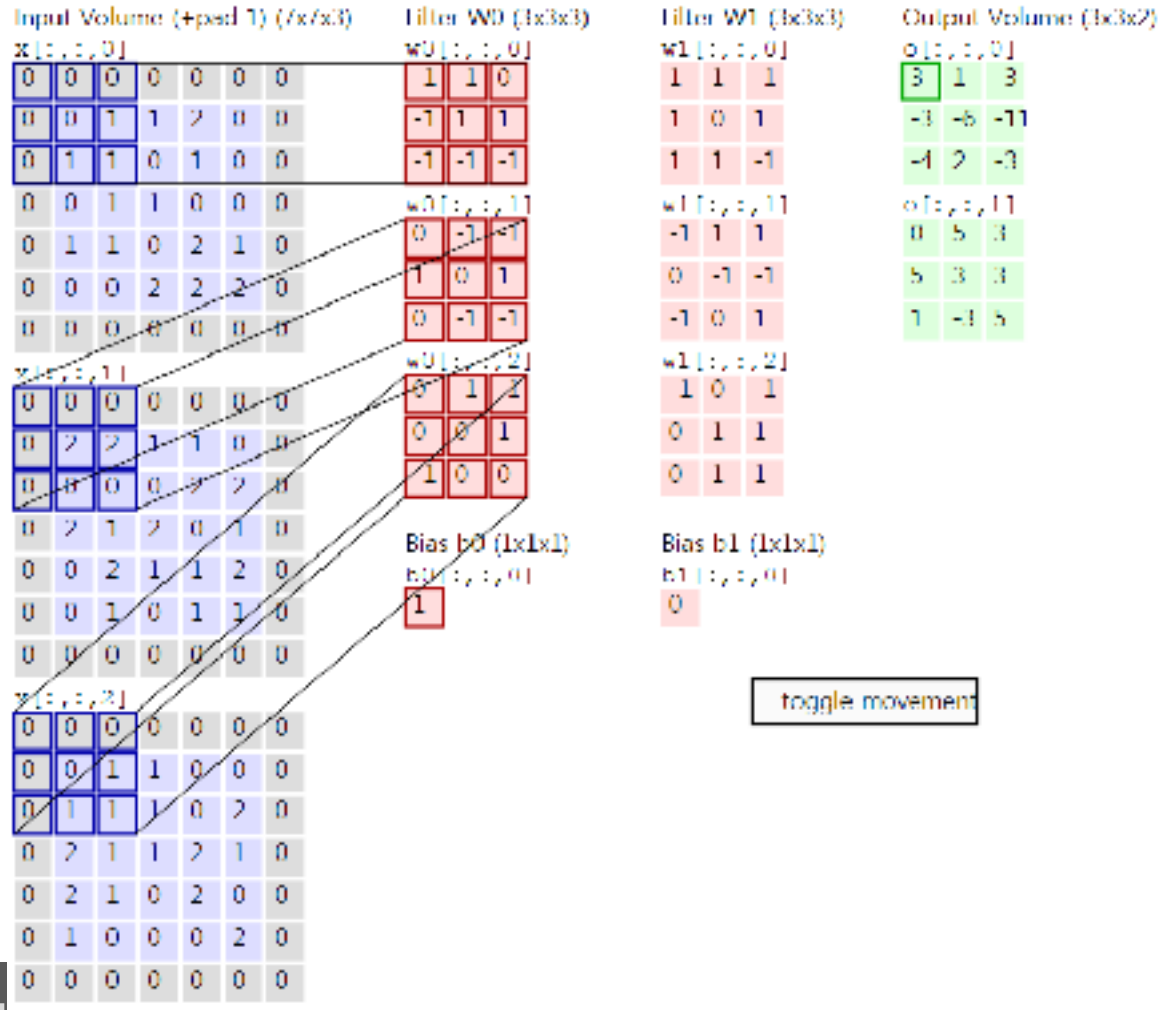
Convolutional Neural Networks

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



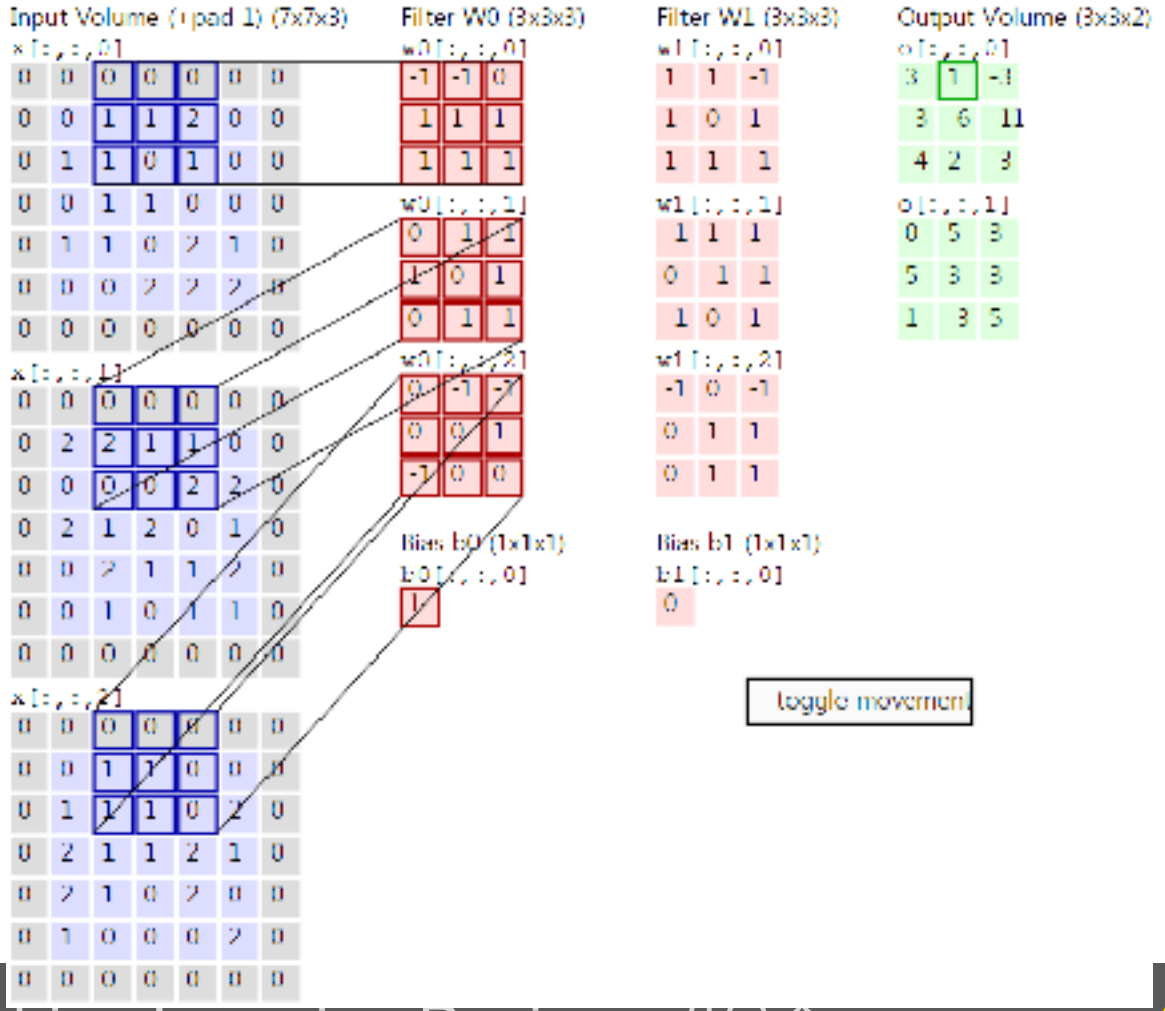
Convolutional Neural Networks

- Convolutional Layer



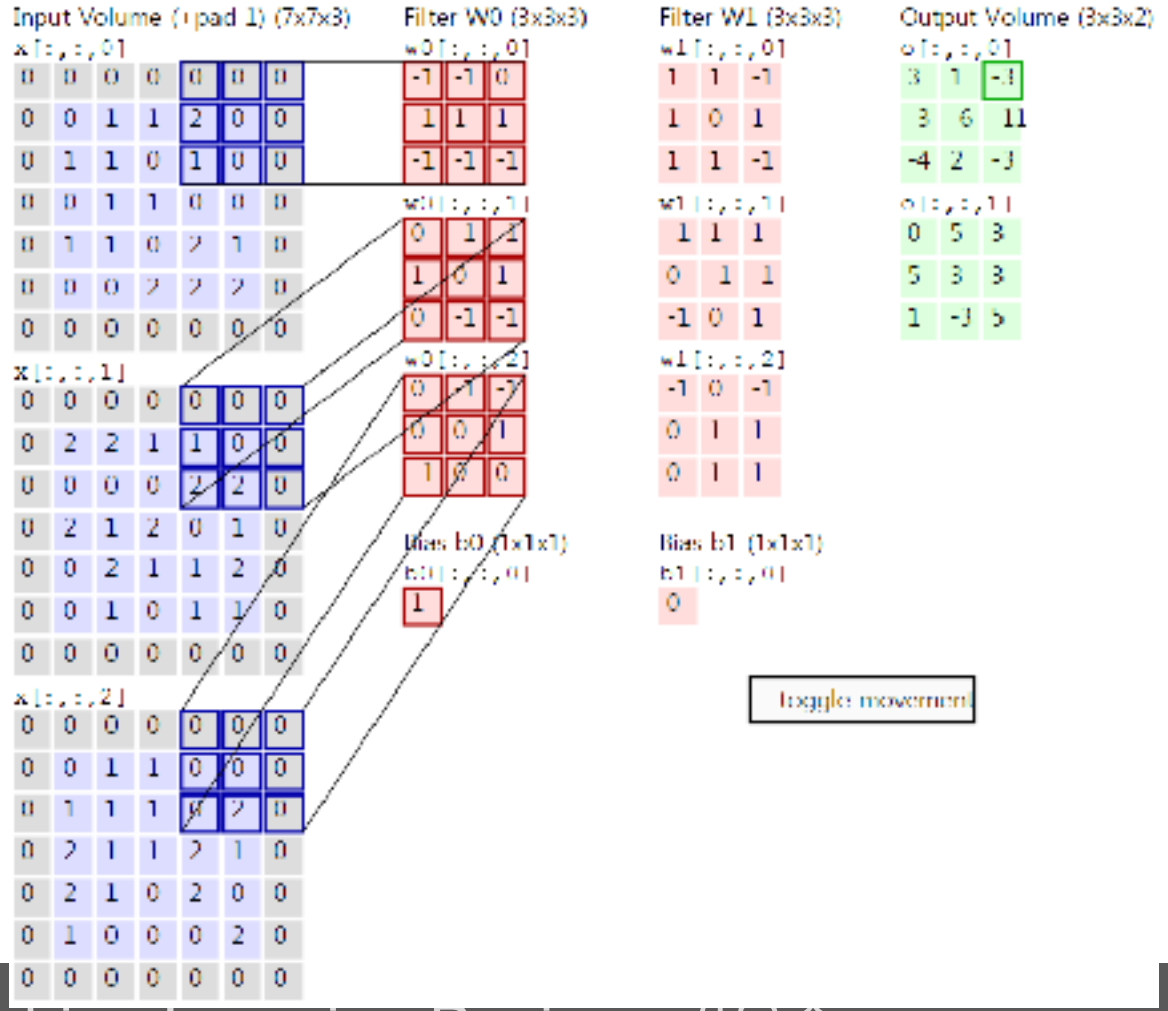
Convolutional Neural Networks

- Convolutional Layer



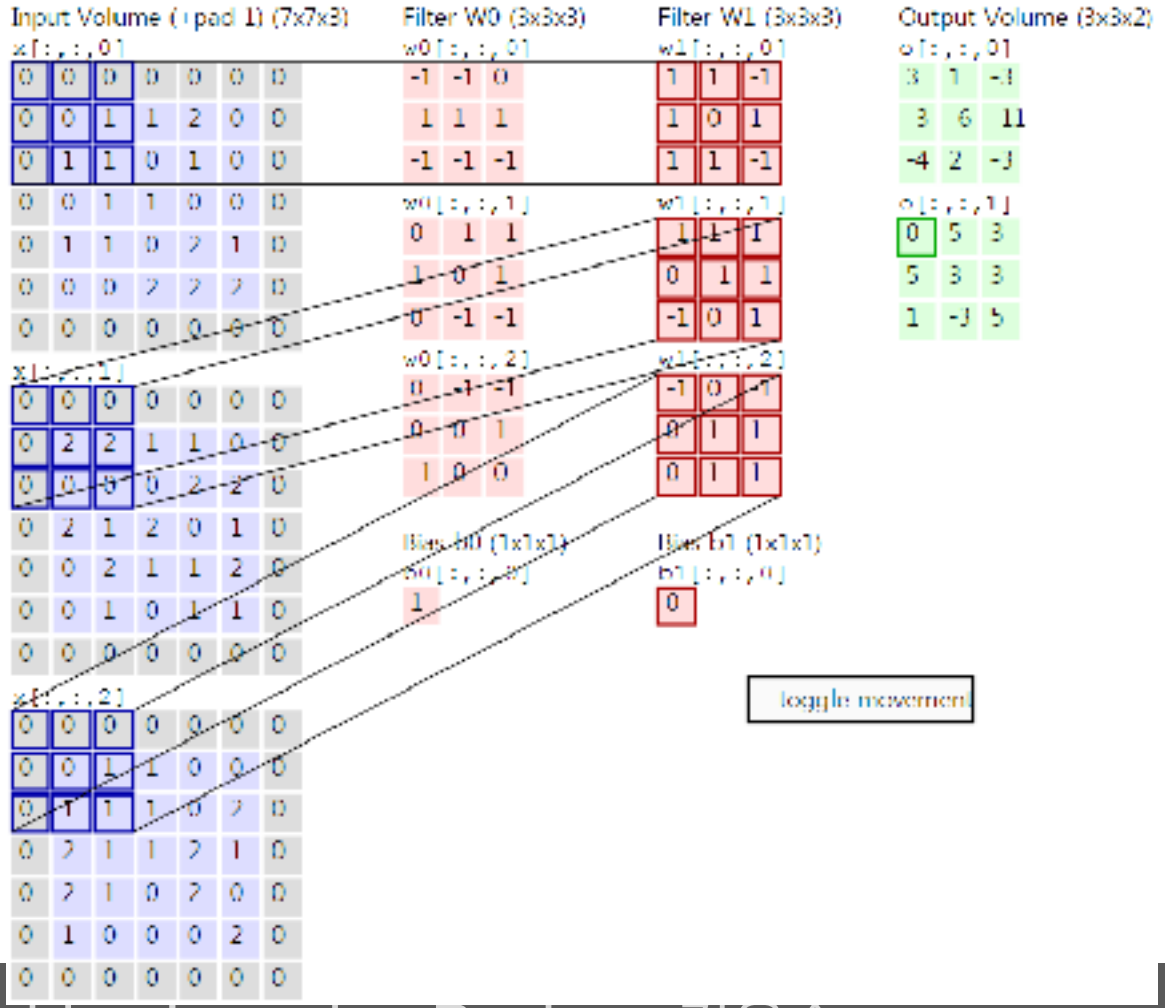
Convolutional Neural Networks

- Convolutional Layer



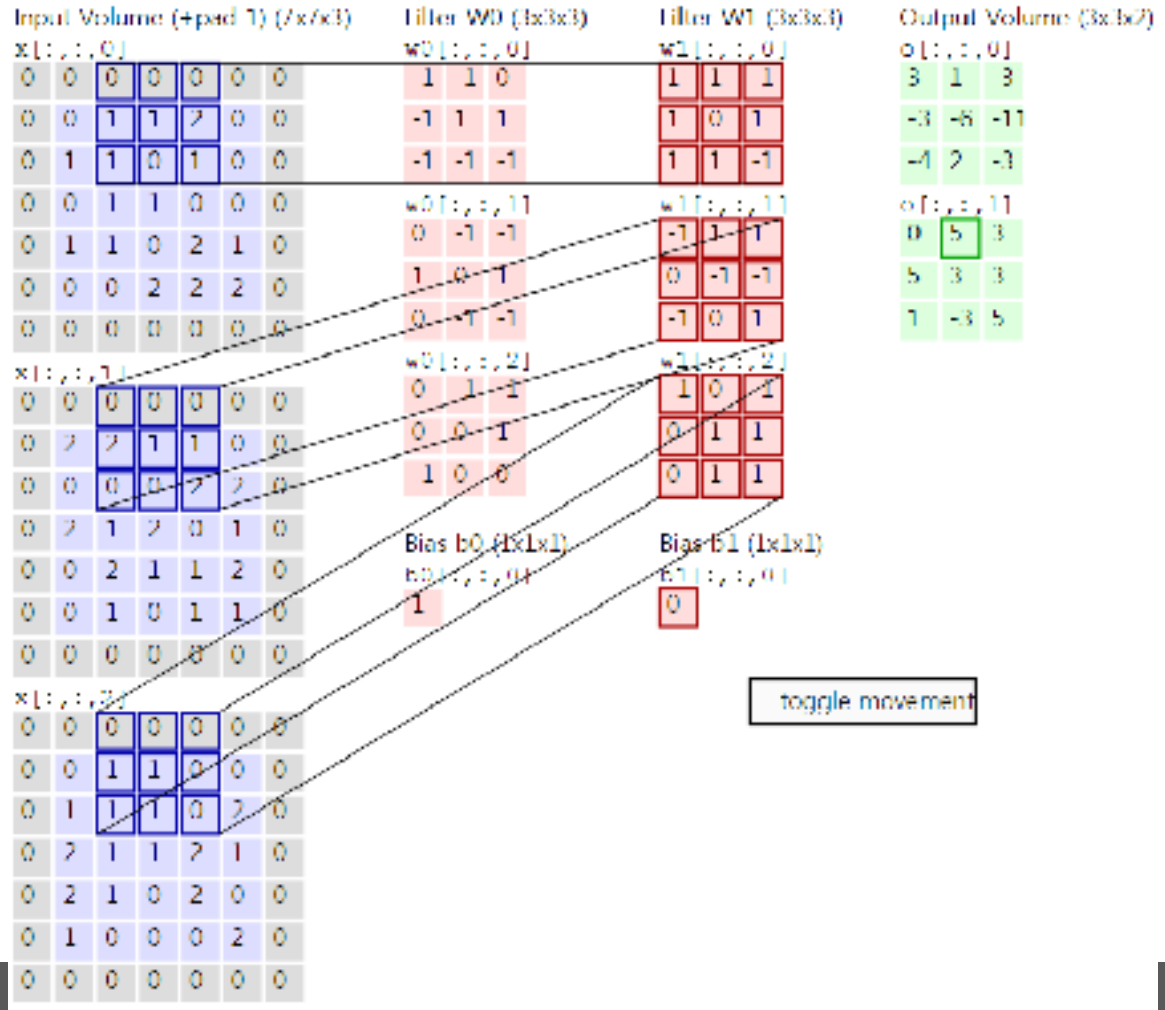
Convolutional Neural Networks

- Convolutional Layer



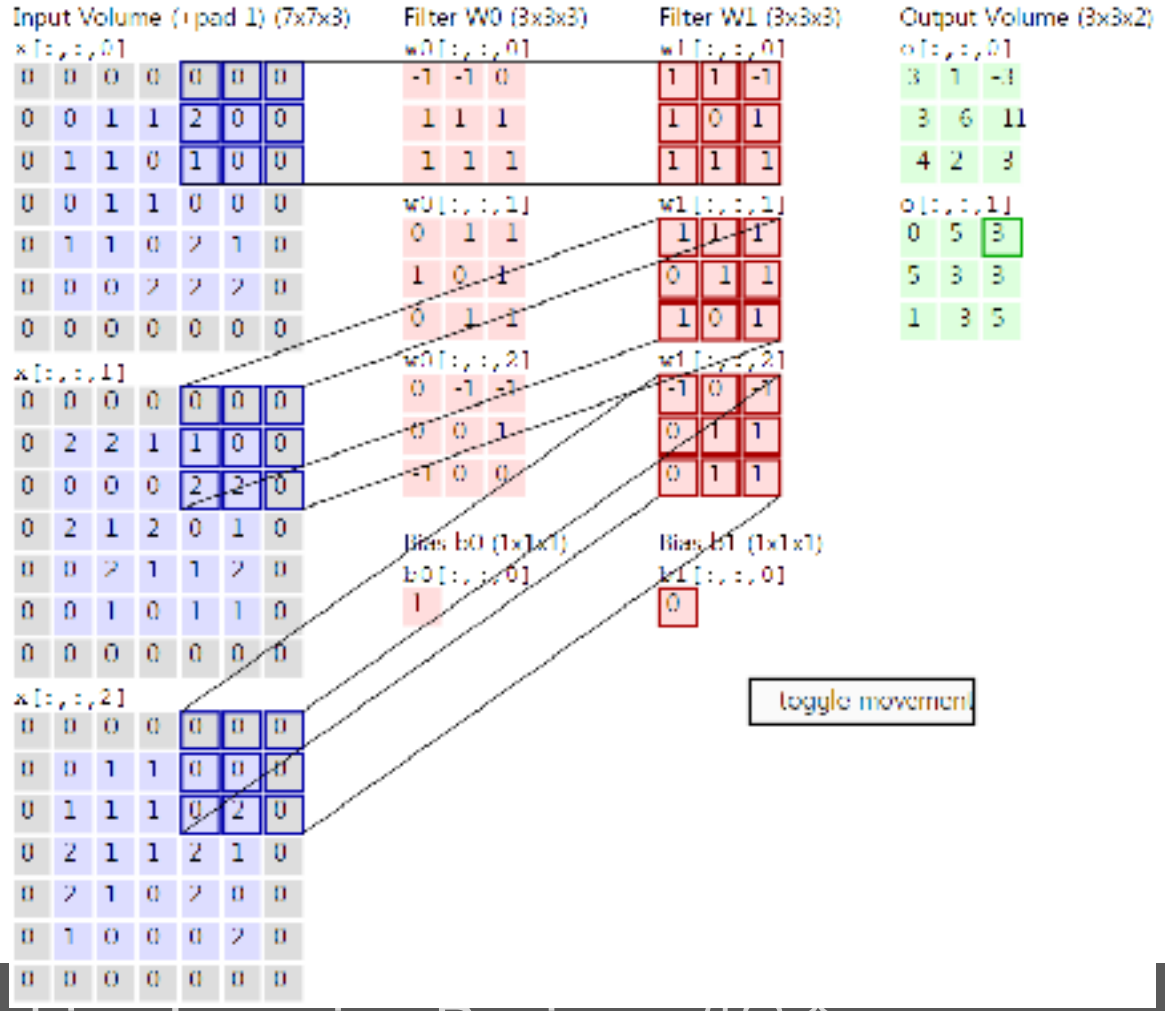
Convolutional Neural Networks

- Convolutional Layer



Convolutional Neural Networks

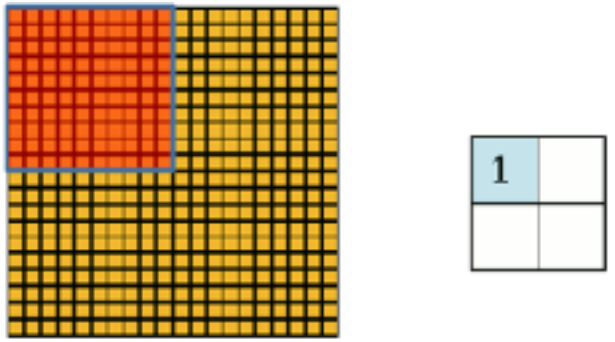
- Convolutional Layer



Max-Pooling Layer

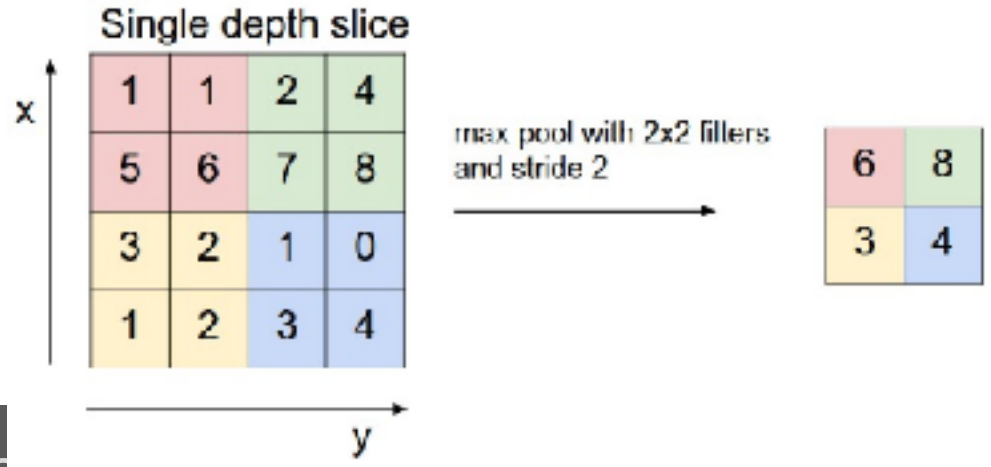
- The Pooling Layer summarizes the results of Convolution Layer
  - e.g.) 10x10 result is summarized into 1 cell

- The Result of Pooling Layer is Translation-invariant

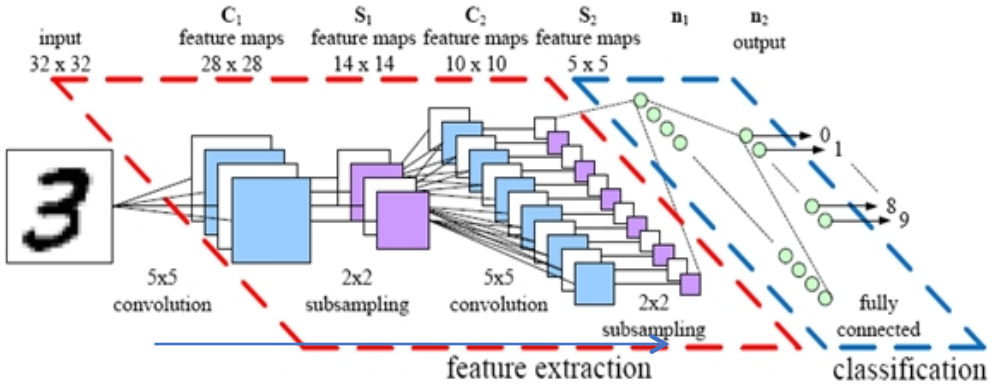


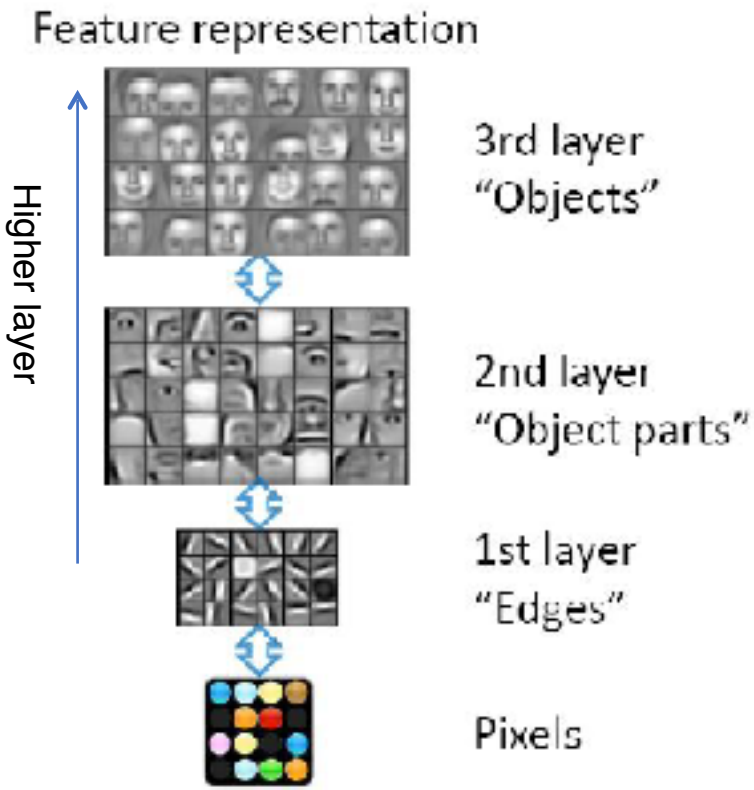
Convolutional Neural Networks

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



Remarks

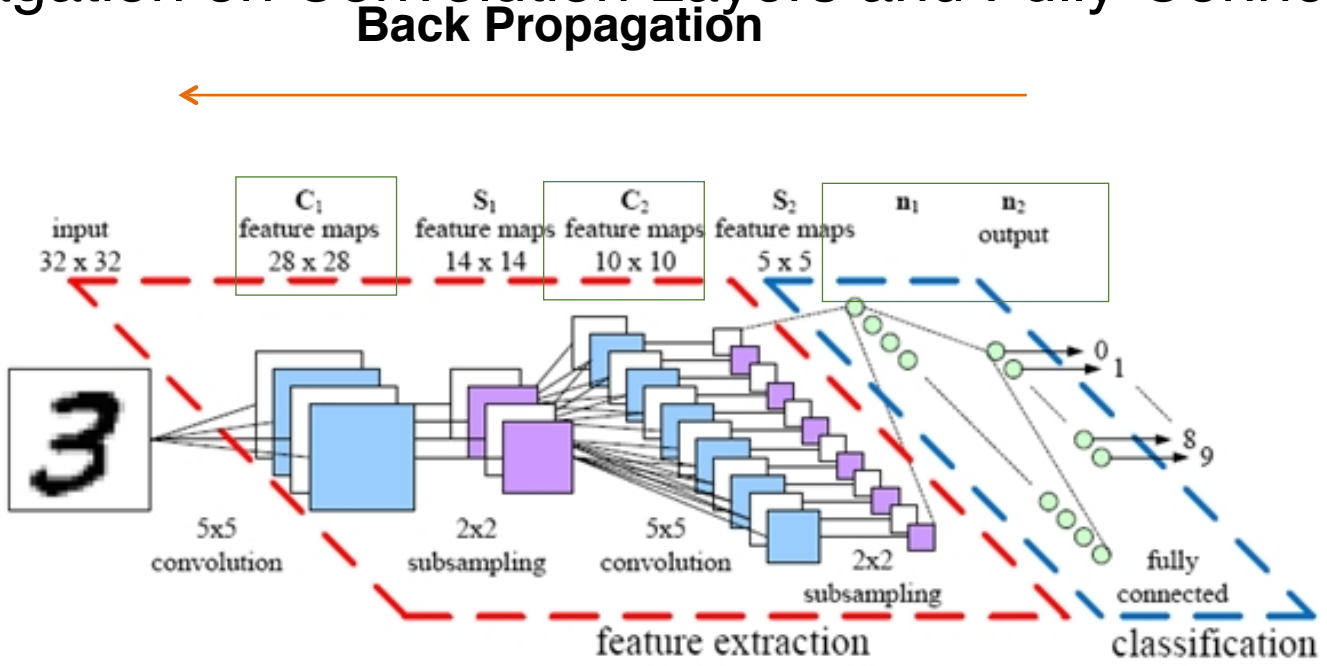




- Higher layer catches more specific, abstract patterns
- Lower layer catches more general patterns

Parameter Learning of CNN

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



Applications (Image Classification) (1/4)

Image Net Competition Ranking

(1000-class, 1 million images)

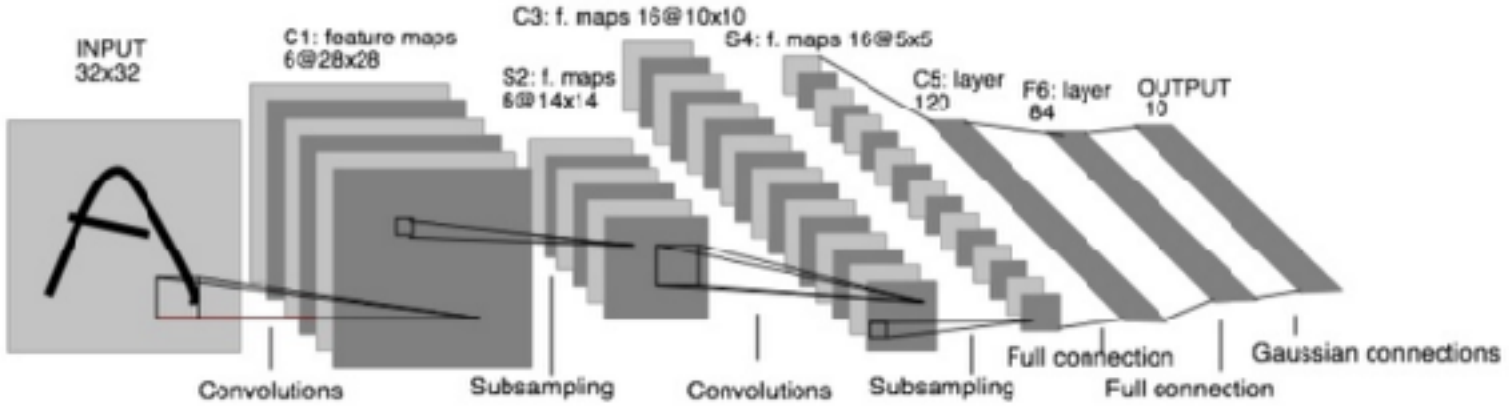
1. Clarifi (**0.117**): **Deep Convolutional Neural Networks** (Zeiler)
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional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

ALL CNN!!

From Kyunghyun Cho's dnn tutorial

Applications (Image Classification) (2/4)

- 1989, CNN for hand digit recognition, Yann LeCun

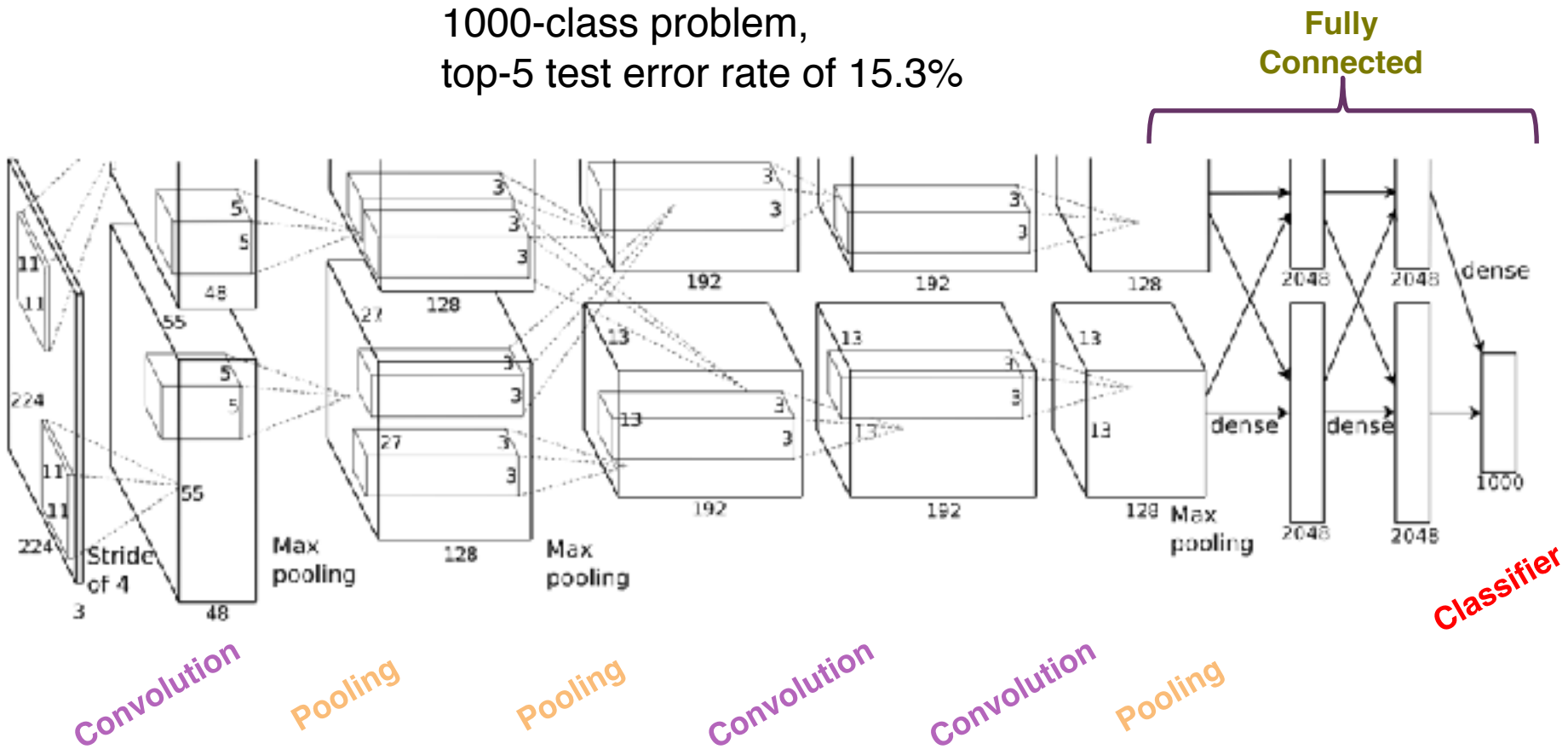


LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [Yann LeCun; LeNet]

Applications (Image Classification) (3/4)

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

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

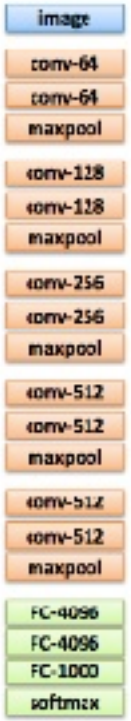


Applications (Image Classification) (4/4)

- 2014 ILSVRC winner, ~6.6% Top 5 error

Example: VGG

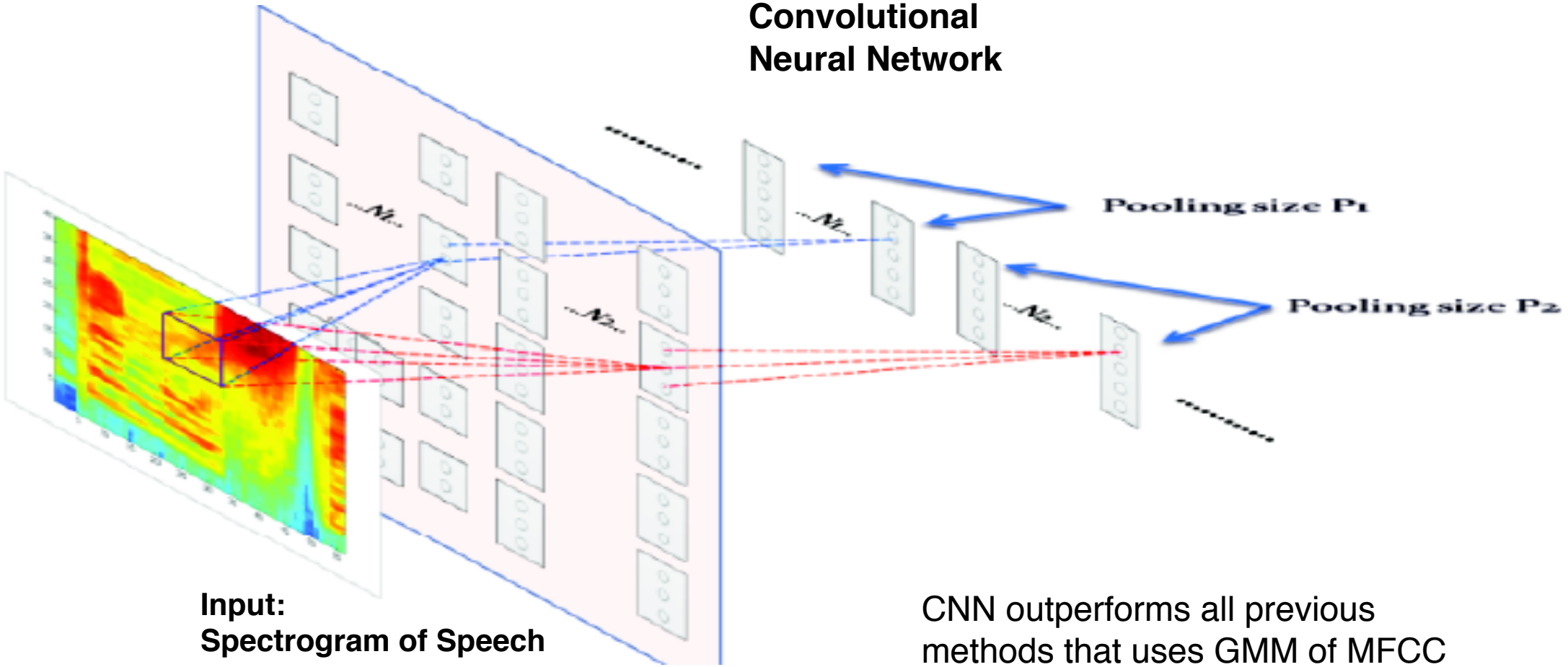
19 layers
3x3 convolution
pad 1
stride 1



Convolution
Pooling
Softmax
Other

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

Application (Speech Recognition)



CNN outperforms all previous methods that uses GMM of MFCC

Recurrent Neural Networks

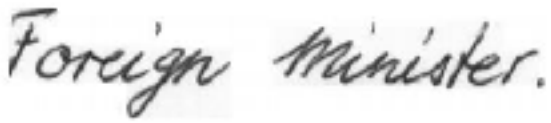
RNN의 활용 분야

- 순서 데이터에 꼬리표 달기(Sequence labeling)

문서 감정 분류
Sentiment classification

★★★★★ 10/10

필기체 인식



FOREIGN MINISTER.

음성 인식

THE SOUND OF

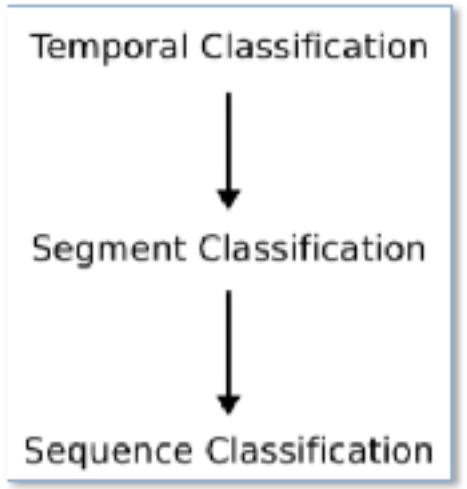
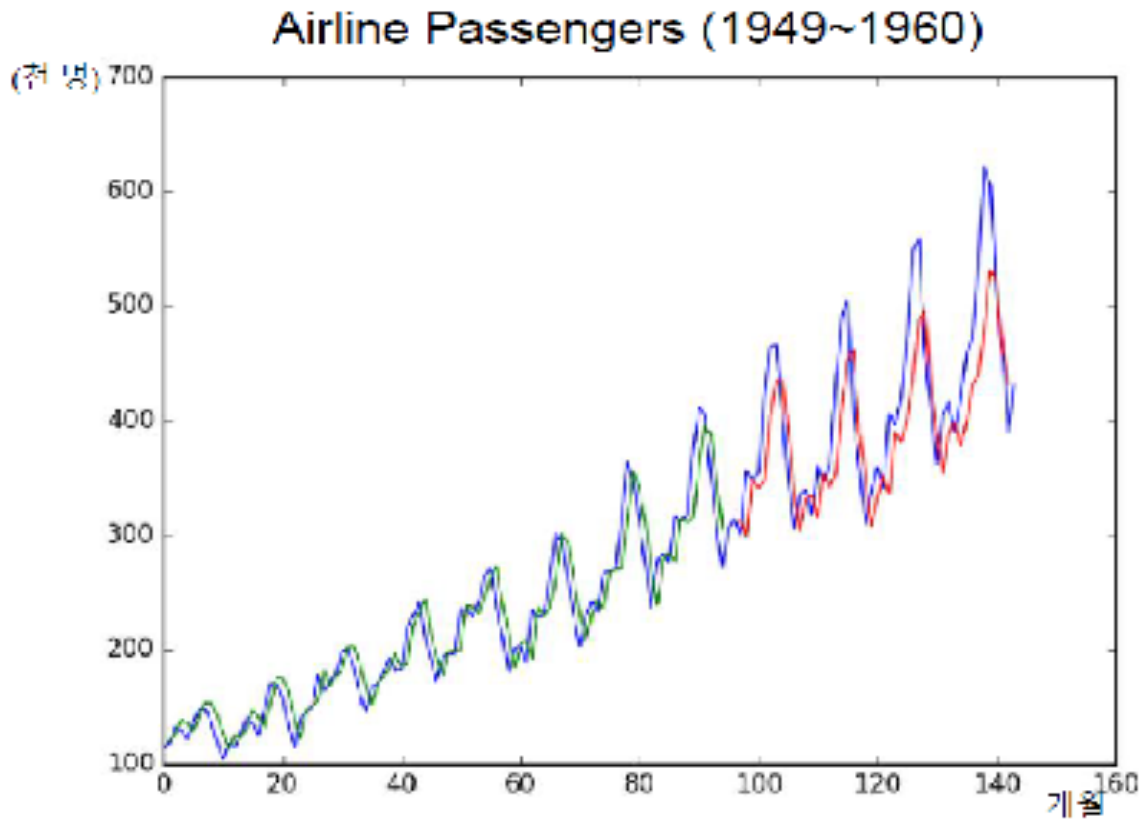


Figure 2.1: **Sequence labelling.** The algorithm receives a sequence of input data, and outputs a sequence of discrete labels.

RNN의 활용 분야

- 시계열 예측(time series prediction)



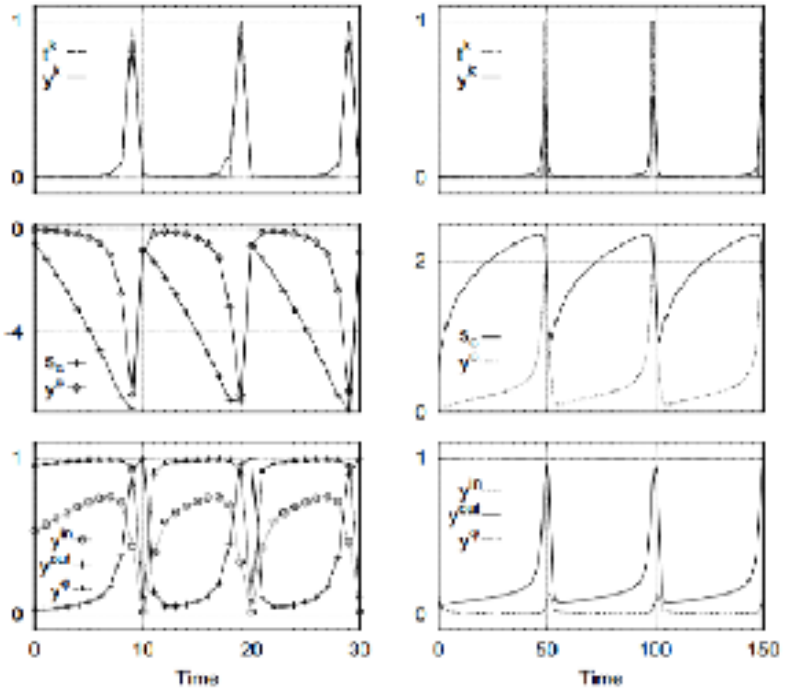
Blue=Whole Dataset, Green=Training, Red=Predictions

RNN의 활용 분야

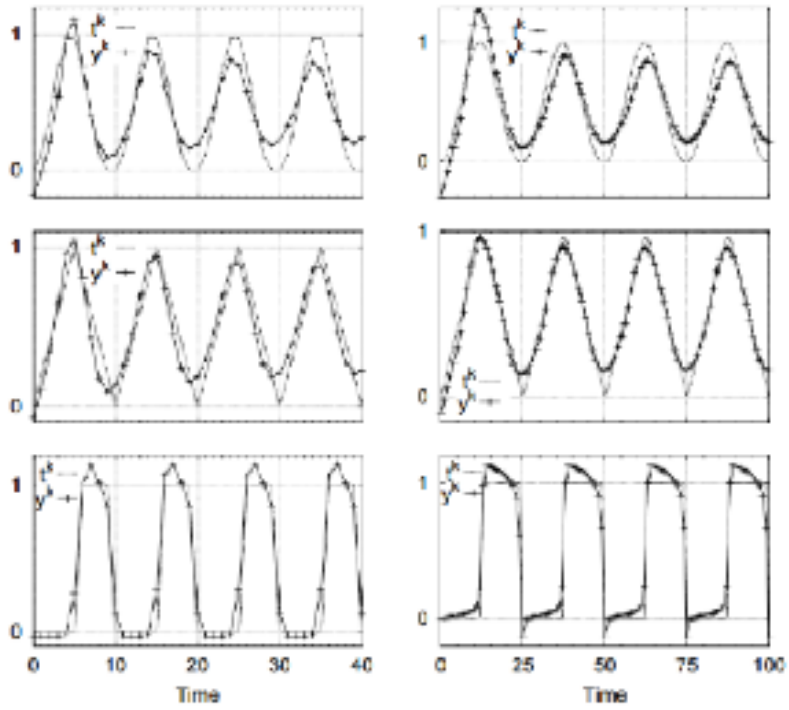
- 파형 생성(Spike or periodic function generation)

Gers, Schraudolph and Schmidhuber, Learning Precise Timing with LSTM Recurrent Networks, JMLR, 2002.

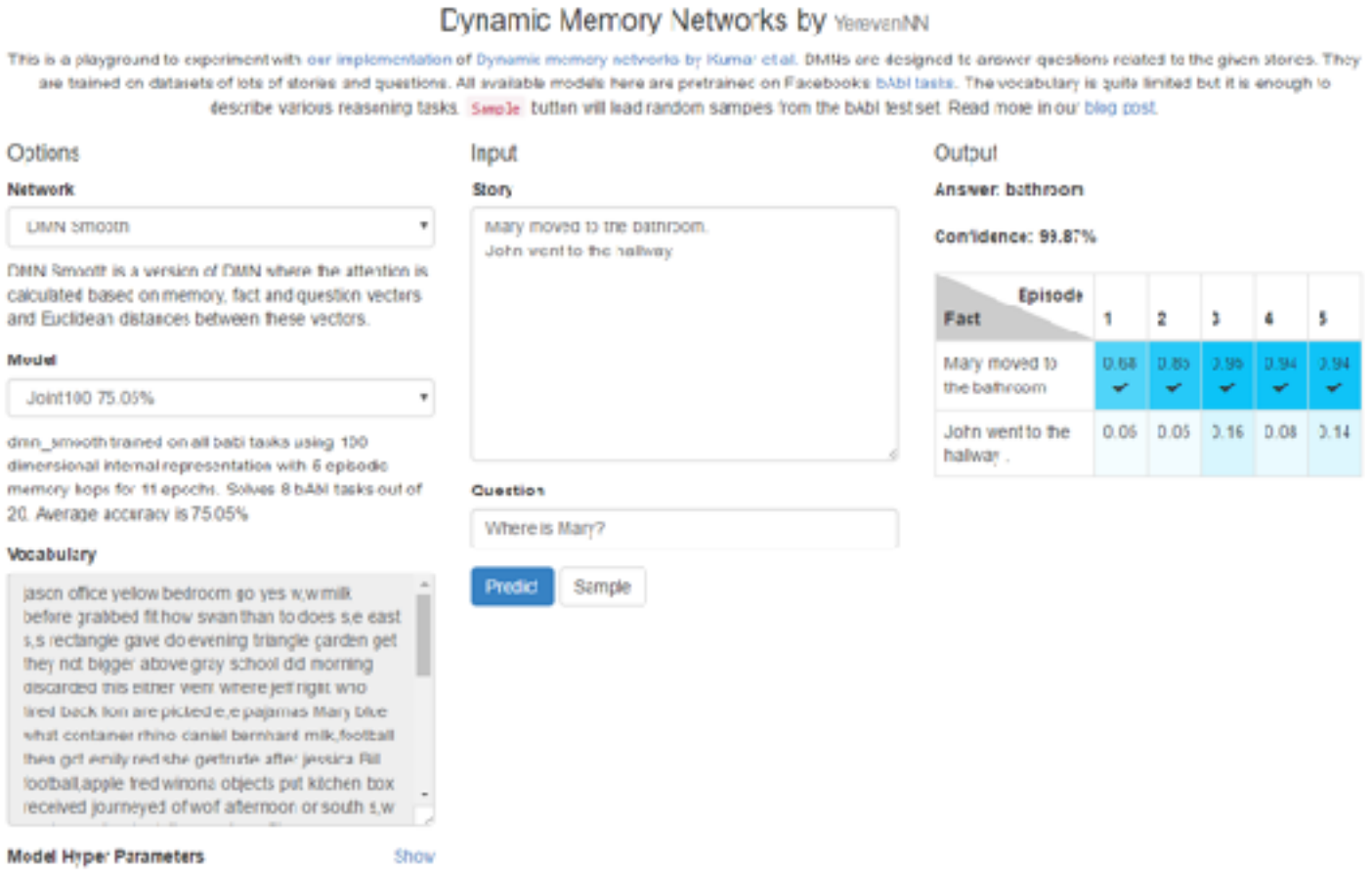
Generating Timed Spikes (GTS)



Periodic Function Generation (PFG)



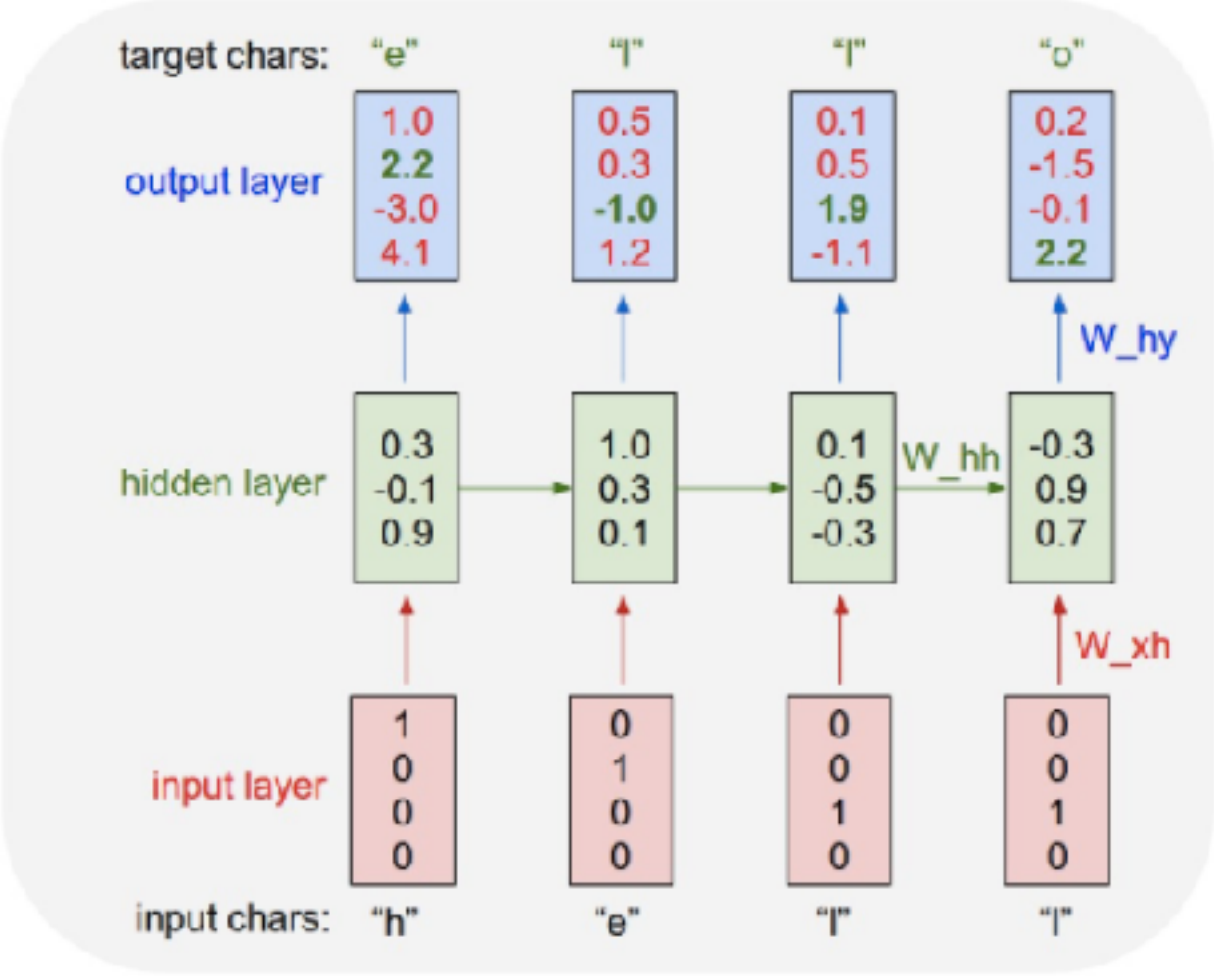
RNN의 활용 분야


Dynamic Memory Networks by YerevanNN
This is a playground to experiment with our implementation of Dynamic memory networks by Kumar et al. DMNs are designed to answer questions related to the given stories. They are trained on datasets of lots of stories and questions. All available models here are pretrained on Facebooks bAbI tasks. The vocabulary is quite limited but it is enough to describe various reasoning tasks. Sample button will load random samples from the bAbI test set. Read more in our blog post.
Options
Network
DMN Smooth
DMN Smooth is a version of DMN where the attention is calculated based on memory, fact and question vectors and Euclidean distances between these vectors.
Model
Joint100 75.05%
dmn_smooth trained on all babi tasks using 100 dimensional internal representation with 6 episodic memory hops for 11 epochs. Solves 8 bABI tasks out of 20. Average accuracy is 75.05%
Vocabulary
Model Hyper Parameters
Show
Input
Story
Mary moved to the bathroom.
John went to the hallway
Question
Where is Mary?
Predict
Sample
Output
Answer: bathroom
Confidence: 99.87%
Episode
Fact
1
2
3
4
5
Mary moved to the bathroom
0.68
0.85
0.99
0.94
0.94
John went to the hallway .
0.06
0.05
0.16
0.08
0.14

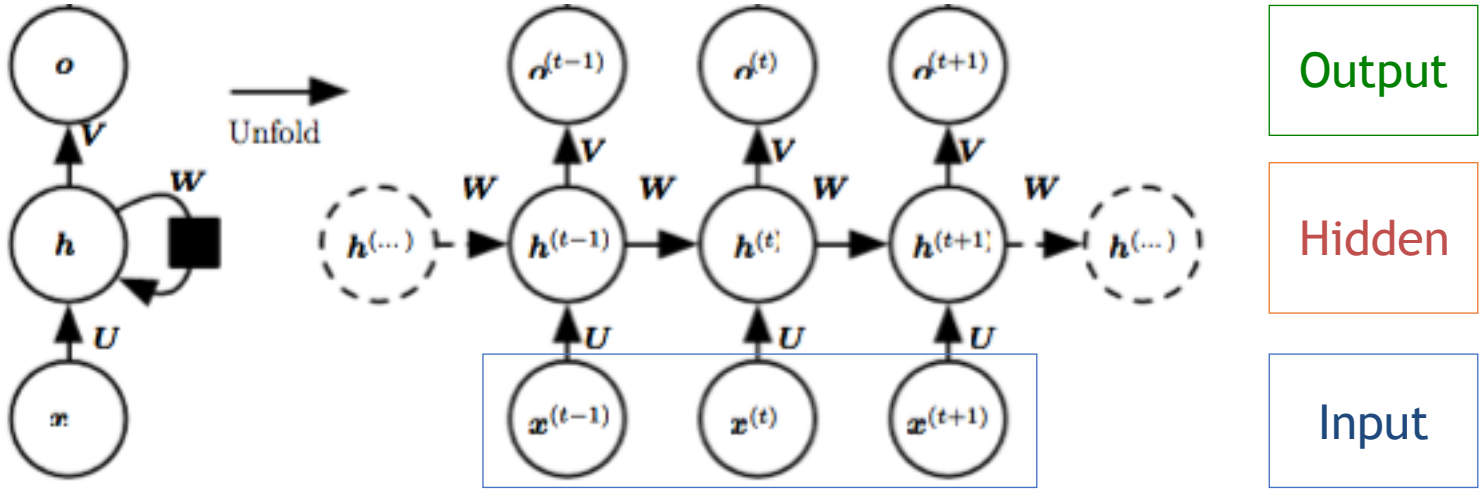
Introduction

- 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



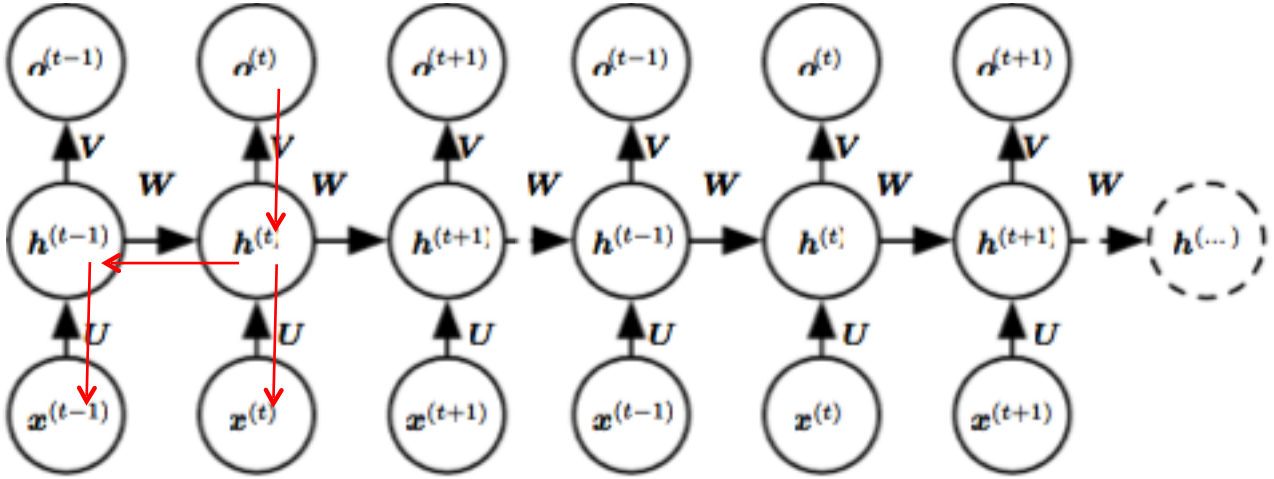
Architecture



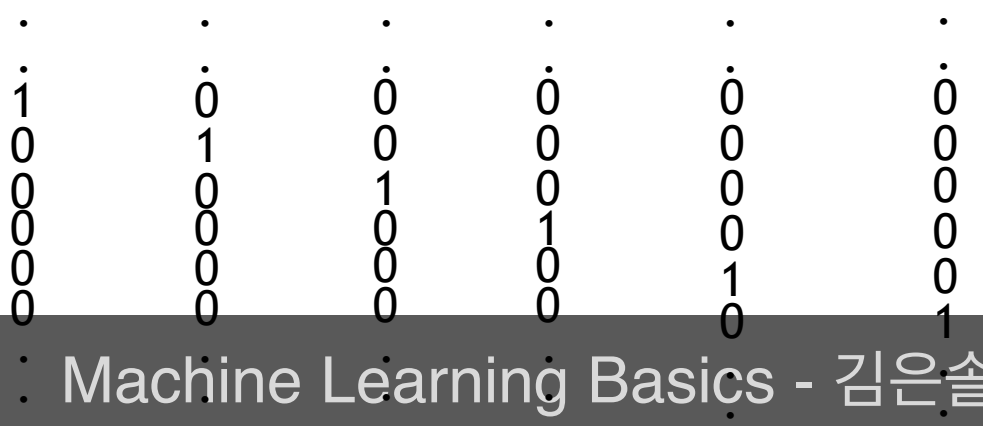
* Parameters: U, W, V

Process – Next word prediction

E

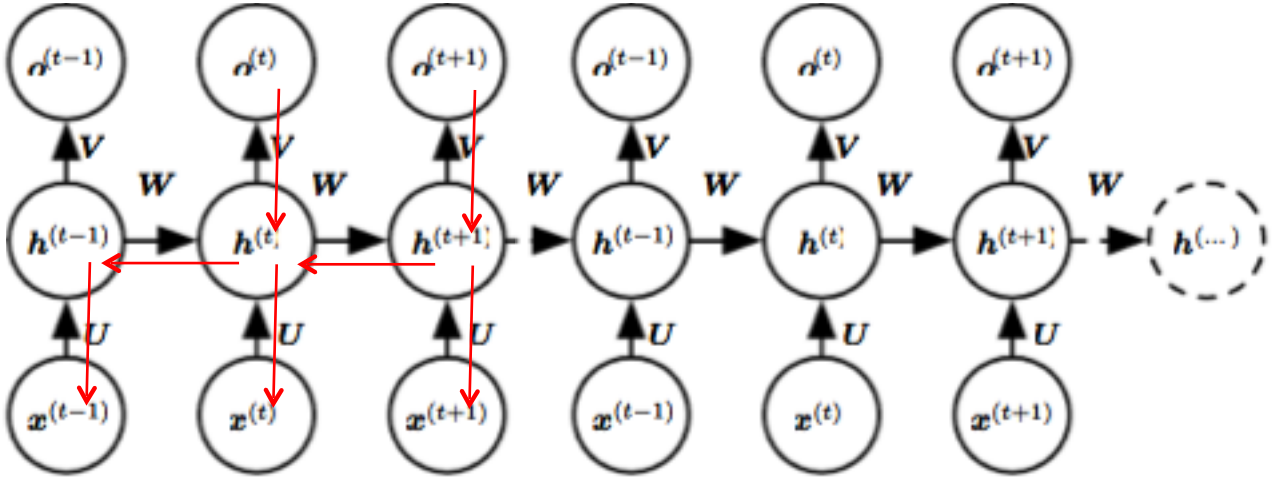


Hello World ! It's an apple

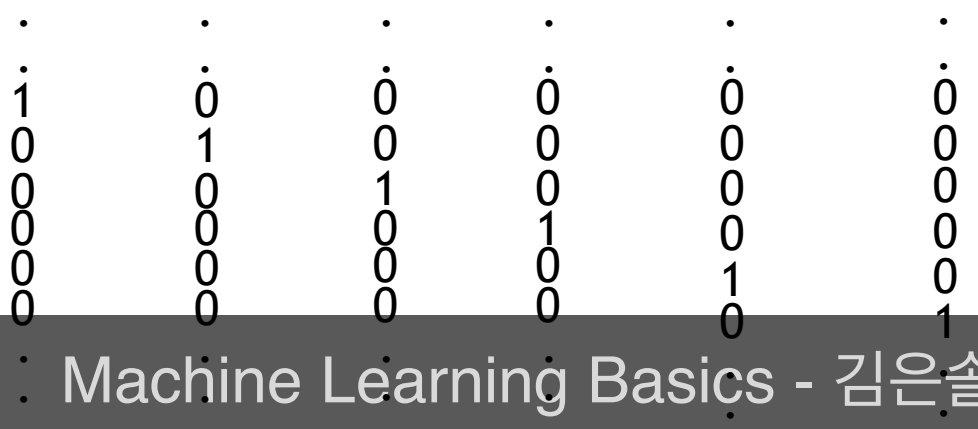


Process – Next word prediction

E E

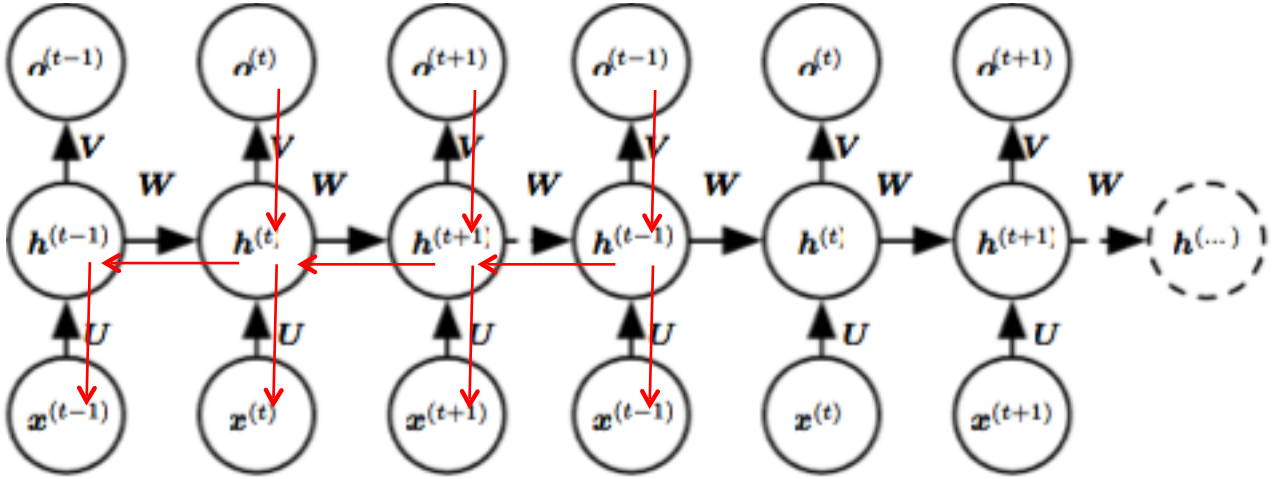


Hello World ! It's an apple

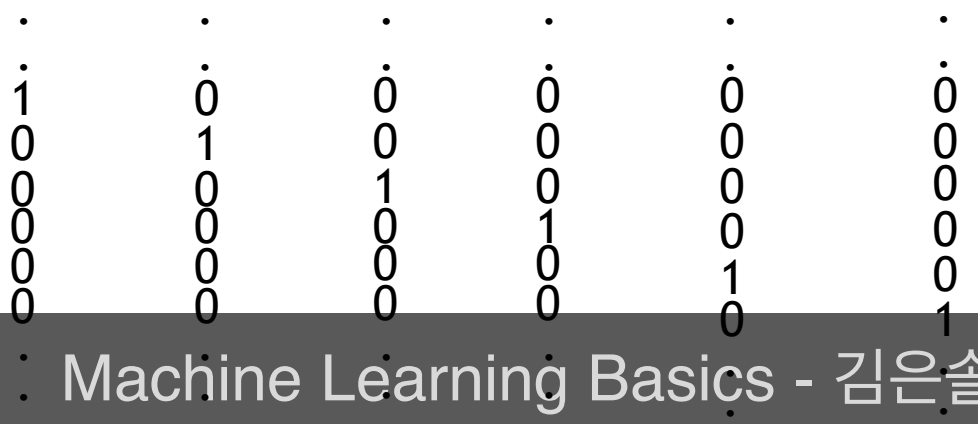


Process – Next word prediction

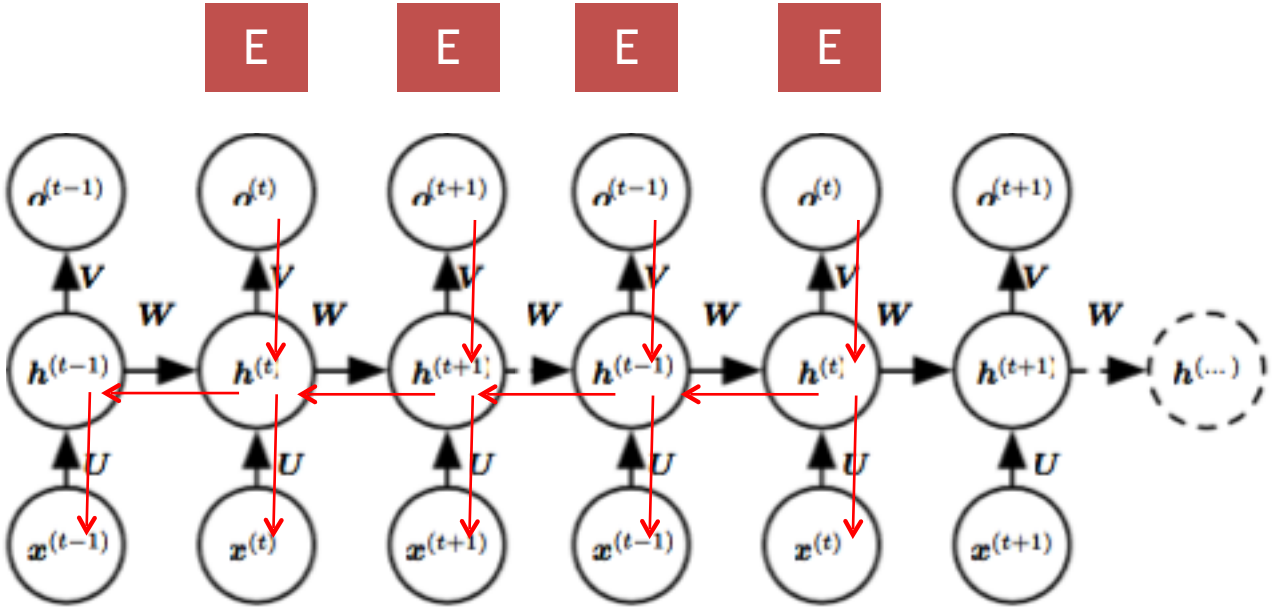
E E E



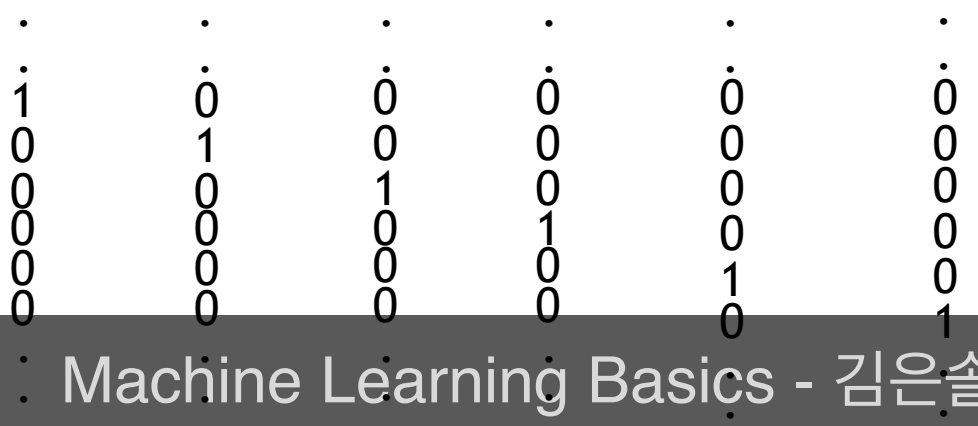
Hello World ! It's an apple



Process – Next word prediction

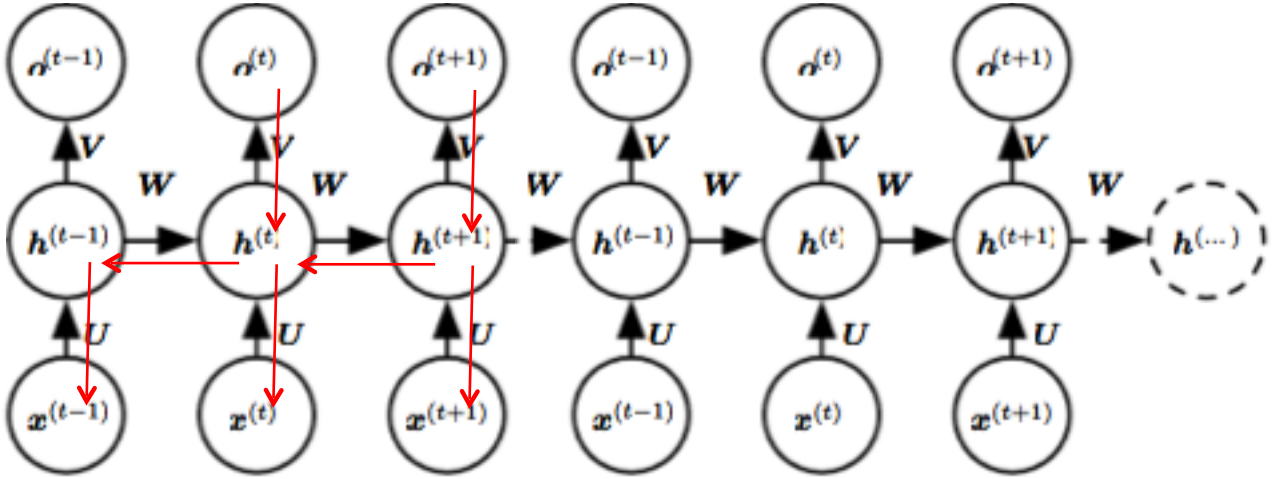


Hello World ! It's an apple

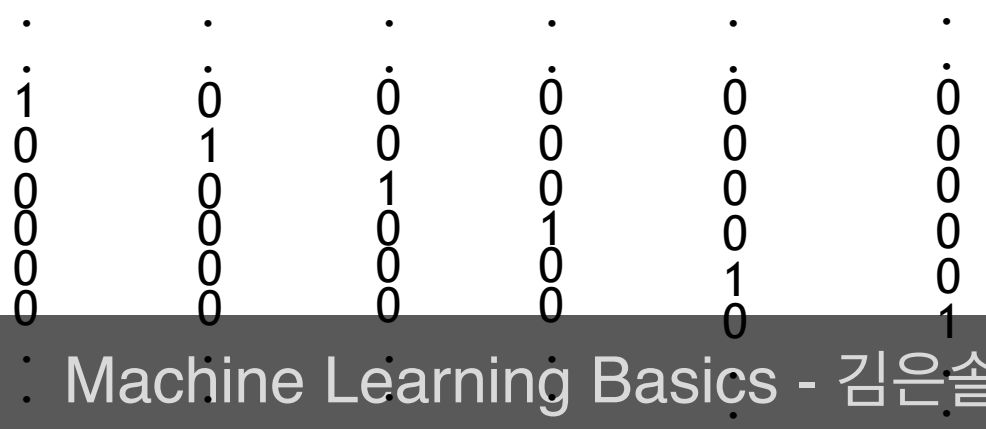


Process – Sentence classification

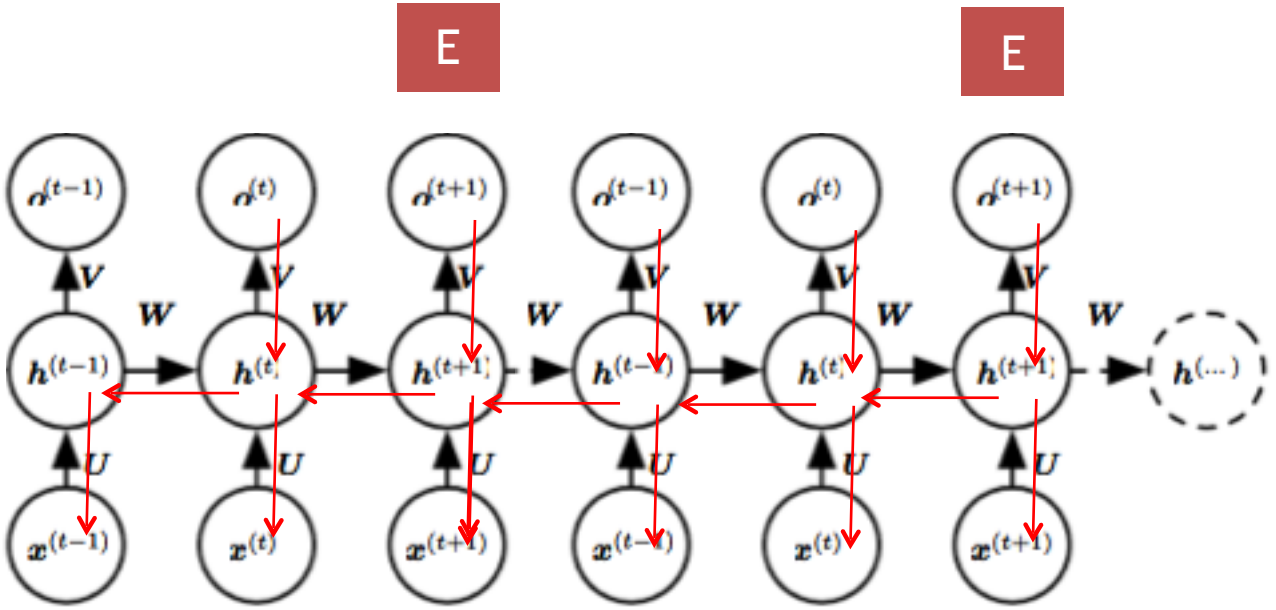
E



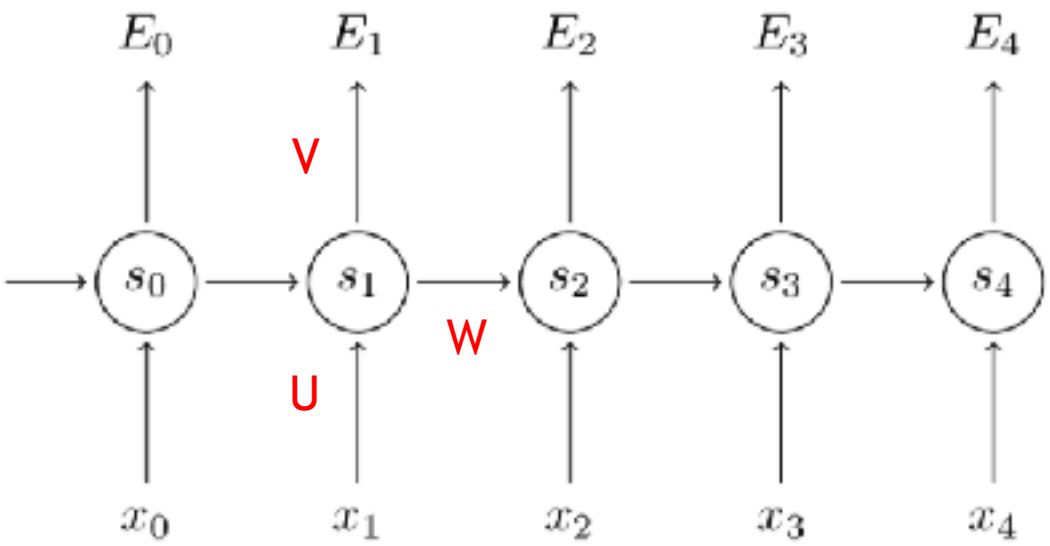
Hello World ! It's an apple



Process – Sentence classification



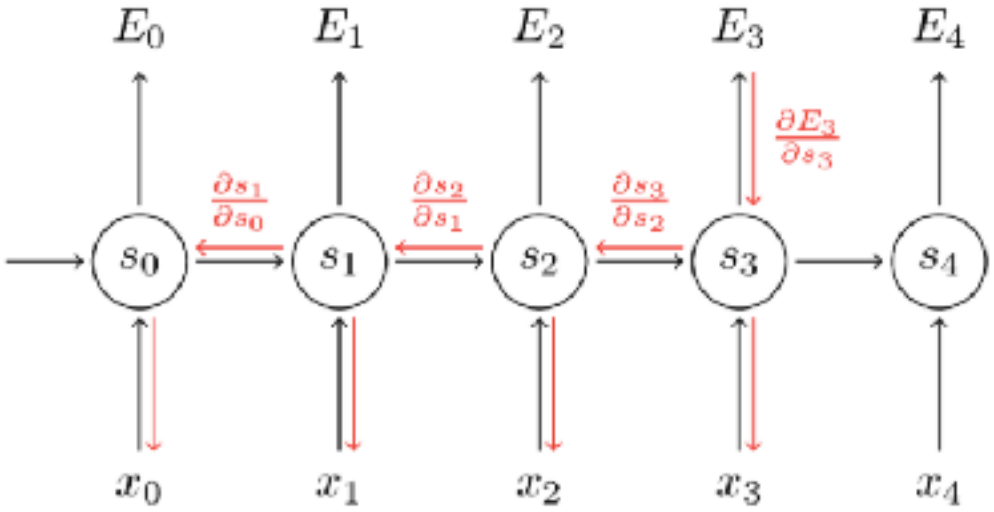
Model



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

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

Learning



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

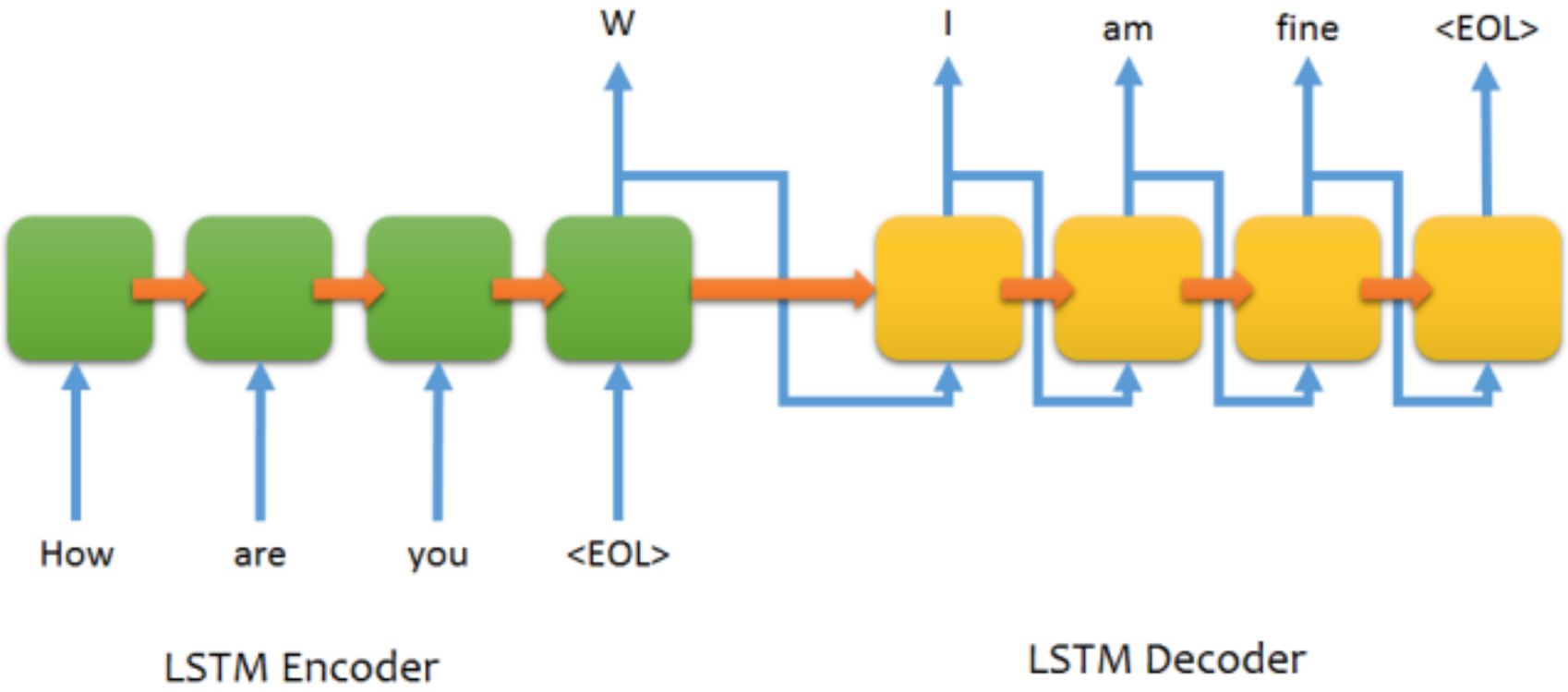
$$= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V}$$

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

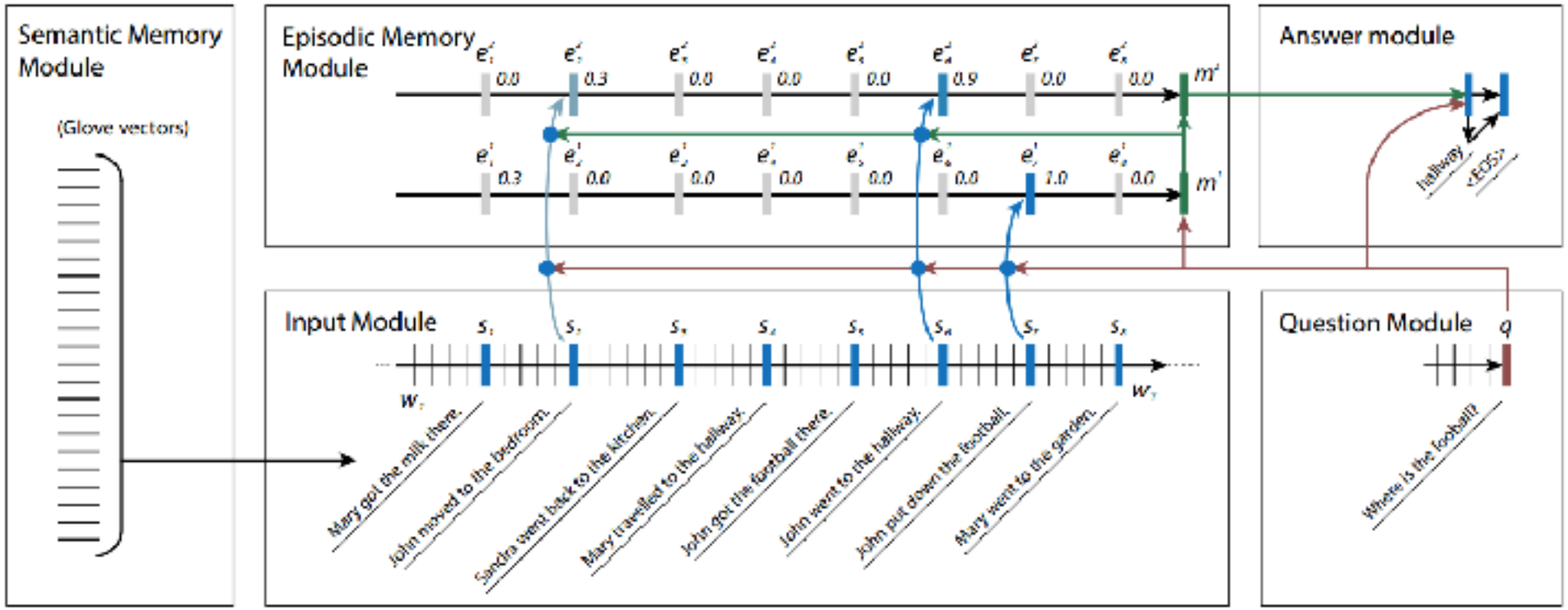
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

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

기계 번역



질의 응답


Semantic Memory Module
(Glove vectors)
Episodic Memory Module
0.0
0.3
0.0
0.0
0.0
0.9
0.0
0.0
0.3
0.0
0.0
0.0
0.0
0.0
1.0
0.0
Answer module
hallway
Input 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.
Question Module
Where is the football?

딥러닝 실습

GPU 컴퓨팅의 필요성

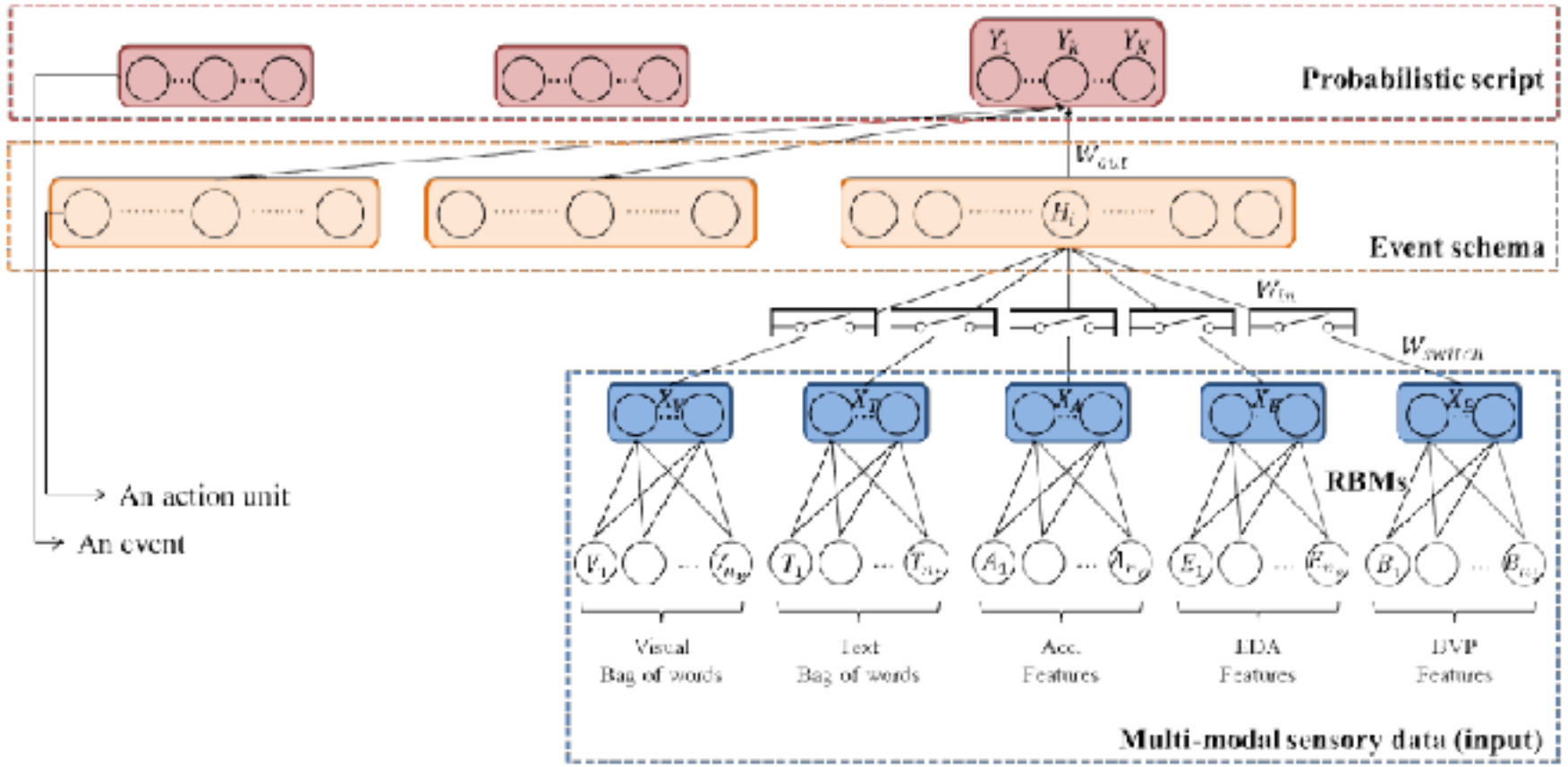
- 다루는 문제의 복잡도가 증가할수록 모델이 커지고 보다 많은 연산이 요구됨
- 컨볼루션 연산, 통합 연산은 모두 행렬 연산
- CUDA를 사용하여 컨볼루션 신경망을 구현한 경우 CPU에 비해 약 10배 이상 속도 향상
  - AlexNet으로 ILSVRC 데이터 학습시 CUDA를 이용한 경우 약 4~5일 정도 소모됨

다양한 딥러닝 프레임워크

	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

구현 예시


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

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

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

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

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

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

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

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

$$\frac{\partial lnE}{\partial \mathbf{w}_m^k} \quad \frac{\partial lnE}{\partial y_{n,k}} \frac{\partial y_{n,k}}{\partial \mathbf{w}_m^k}$$

$$= \begin{pmatrix} t_{n,k} & 1-t_{n,k} \\ y_{n,k} & 1-y_{n,k} \end{pmatrix} (y_{n,k}(1 \quad y_{n,k}) s_m \mathbf{x}_m^n) \quad (4)$$

$$- (\quad t_{n,k}(1 \quad y_{n,k}) \quad (1 \quad t_{n,k}) y_{n,k}) \, s_m \mathbf{x}_m^n$$

$$(y_{n,k} - t_{n,k}) s_m \mathbf{x}_m^n$$

$$\frac{\partial lnE}{\partial \mathbf{u}_m} = \frac{\partial lnE}{\partial y_{n,k}} \frac{\partial y_{n,k}}{\partial s_m} \frac{\partial s_m}{\partial \mathbf{u}_m}$$

$$- (y_{n,k} - t_{n,k}) s_m (1 - s_m) \mathbf{X} \sum_{k=1}^{K} (\mathbf{w}_m^k)^\top \mathbf{x}_m^n \quad (5)$$

$$\frac{\partial lnE}{\partial b_m^k} - \frac{\partial lnE}{\partial y_{n,k}} \frac{\partial y_{n,k}}{\partial b_m^k} \quad (6)$$

$$= (y_{n,k} - t_{n,k})$$

$$\frac{\partial lnE}{\partial a_m} \quad \frac{\partial lnE}{\partial y_{n,k}} \frac{\partial y_{n,k}}{\partial s_m} \frac{\partial s_m}{\partial a_m}$$

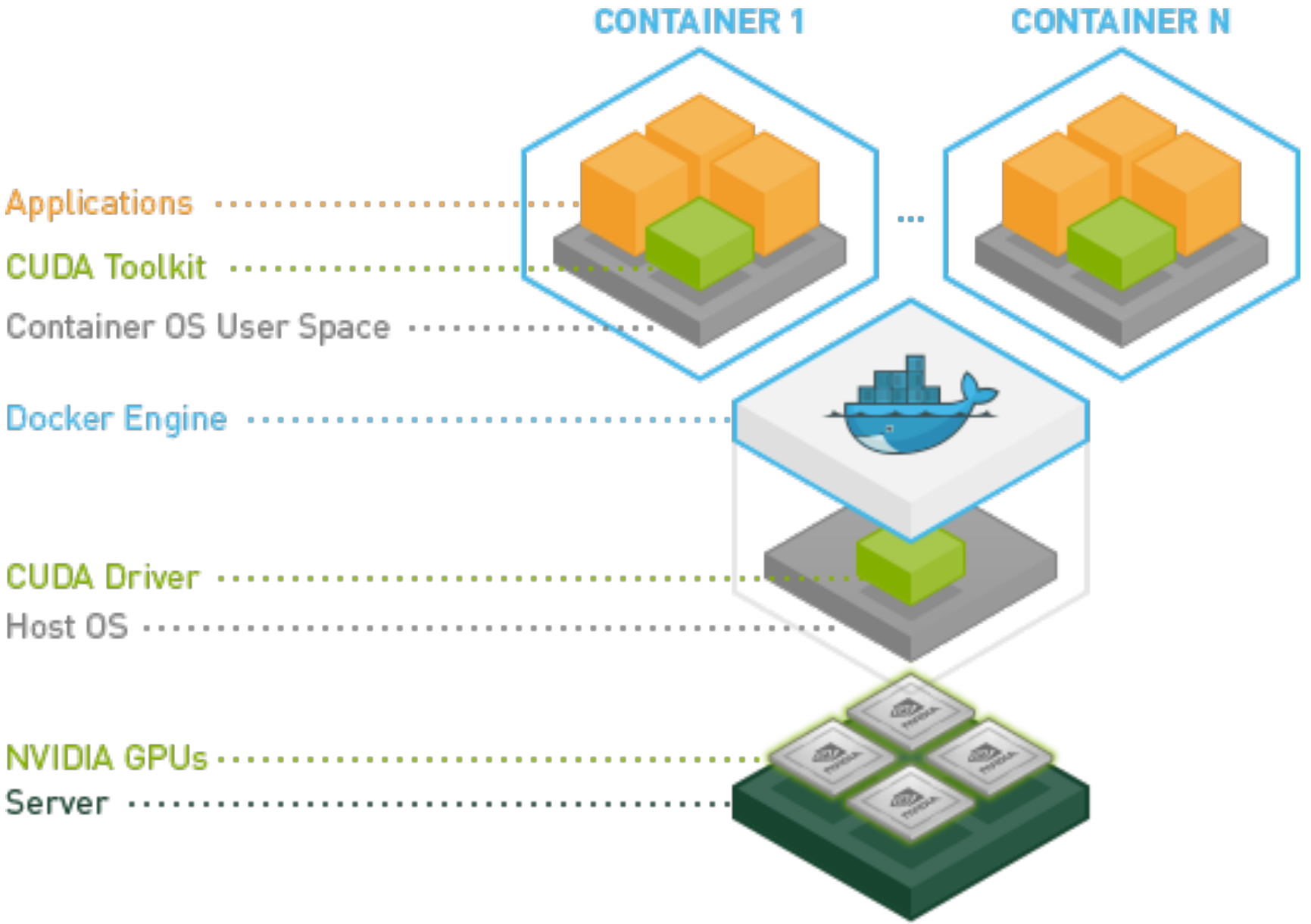
$$= (y_{n,k} - t_{n,k}) s_m (1 - s_m) \sum_{k-1}^{K} (\mathbf{w}_m^k) \, \mathbf{x}_m^n \quad (7)$$

```
switched = True

A_sw = theano.shared(0.01*np.asarray(rng.uniform(low= 4*np.sqrt(6. / (n_switch + n_con_input)),
                                                 high=4*np.sqrt(6. / (n_switch +   con_input)),
                                                 size=(n_con_input, n_switch)), dtype=floatX),
                     name='A_sw',  borrow=True)

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


A = theano.shared(0.01*np.asarray(rng.uniform( low= 4*np.sqrt(6. / (n_output + n_con_input)),
                                               high=4*np.sqrt(6. / (n_output +   con_input)),
                                               size=(n_con_input, n_output)), dtype=floatX),
                  borrow=True)

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

```
switch = T.  et.sigmoid(T.do(inp, W_sw)+b_sw) # (n_data x n_switch)
sw_indicator = T.cast(switch, grp_dtype) # (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(inp*switch, W) + b) # (n_data x n_output)
    params = [A_sw, b_sw, A, b]
else:
    output = T.nnet.sigmoid(T.dot(inp, A)+b)
    #output = T.nnet.sigmoid(T.dot(inp*switch, W)+b) # (n_data x n_output)
    params = [A, b]
```

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

서버 구축


CONTAINER 1
CONTAINER N
Applications
CUDA Toolkit
Container OS User Space
Docker Engine
CUDA Driver
Host OS
NVIDIA GPUs
Server

Logistic Regression 구현

Logistic Regression

$$E(x_1, x_2) = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + b)}}$$

$$loss = -\frac{1}{B} \sum y log(E(\mathbf{x})) + (1-y) log(1 - E(\mathbf{x}))$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} loss$$

실습

- ssh guest@gpu.kias.re.kr -p 6645
- ./go_server.sh
- vim lr_(본인이름).py

Data Generation

```
import tensorflow as tf
import numpy as np

n_data_per_class = 50
num_class = 2
dim_data = 2
n_data = n_data_per_class * num_class
n_train = 50

data1 = np.random.multivariate_normal([4,0], [[1,0],[0,1]], n_data_per_class).astype(np.float32)
data2 = np.random.multivariate_normal([0,4], [[1,0],[0,1]], n_data_per_class).astype(np.float32)
data_x = np.vstack([data1,data2])
data_y = np.hstack([np.ones((n_data_per_class,)), -np.ones((n_data_per_class,))])
```

Model Define

```
x = tf.placeholder(tf.float32, [None, dim_data])
W = tf.Variable(tf.zeros([dim_data,1]))
b = tf.Variable(tf.zeros([1]))
y_true = tf.placeholder(tf.float32, [None])

a = tf.matmul(x, W) + b
e = tf.div(1., 1. + tf.exp(-a))
loss = -tf.reduce_mean(y_true * tf.log(e) + (1-y_true)*tf.log(1-e))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(loss)
```

Training and Testing

```
for i in range(200):
    train_x = data_x[idx]
    train_y = data_y[idx]
    sess.run(train_step, feed_dict={x: train_x, y_true: train_y})
    if i % 20 == 0:
        print(i, sess.run(loss, feed_dict={x:train_x, y_true:train_y}))

print('[6.0, 0.0] : ', sess.run(e, feed_dict={x: [[6.0, 0.0]]} ) > 0.5 )
```

github.com/kimeunsol