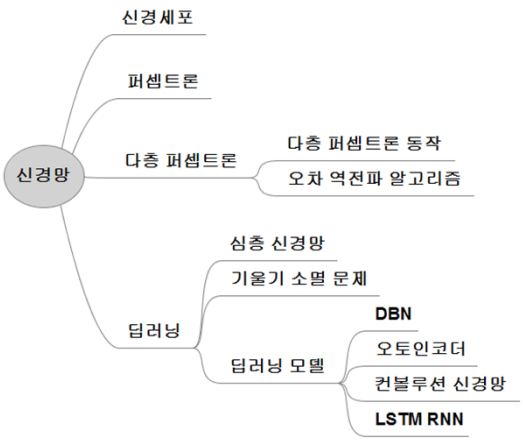
기계 학습

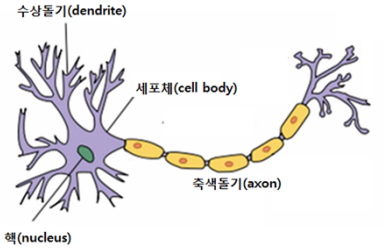
Part Ⅲ

충북대학교 소프트웨어학과
이건명



8. 신경망

❖ **신경망(neural network, artificial neural network)**
- 인간 두뇌에 대한 계산적 모델을 통해 인공지능을 구현하려는 분야
- **신경세포** (neuron)



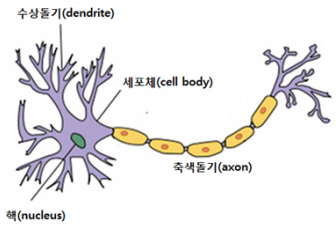
신경세포 8.6×10^{10} 개

신경연접 1.5×10^{14} 개

- **수상돌기(**樹狀突起, dendrite) : 다른 신경세포의 축색돌기와 연결되어 **전기화학적 신호**를 받아들이는 부위
- **축색돌기**(軸索突起, axon) : 수신한 전기화학적 신호의 합성결과 값이 특정 임계값이 이상이면 신호를 내보는 부위.
- **신경연접**(神經連接, synapse) : 수상돌기와 축색돌기 연결 부위
  - 전달되는 신호의 증폭 또는 감쇄

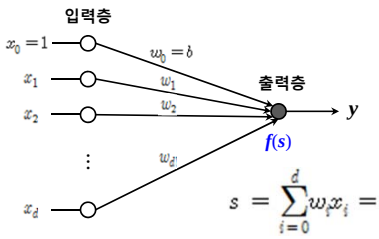
신경망

❖ **신경세포의 계산 모델**
- 신경세포

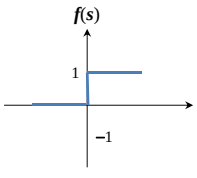


McClulloch & Pitts, 1943
신경세포의 계산 모델

- **퍼셉트론**(Perceptron)
  - 로젠블랏(Rosenblatt, 1957)이 제안한 학습가능한 신경망 모델



$$s = \sum_{i=0}^{d} w_i x_i = \sum_{i=1}^{d} w_i x_i + b$$

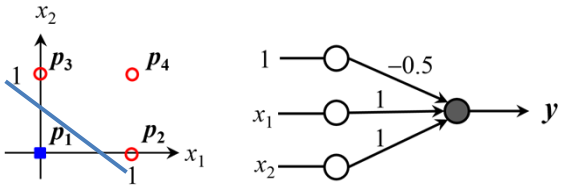


전달함수

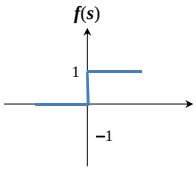
신경망

❖ **퍼셉트론**(Perceptron)

- **OR 연산을 수행하는 퍼셉트론**



$$y = f(s) = f(\sum_{i=1}^{2} w_i x_i + b) = f(\boldsymbol{w}\boldsymbol{x}^T) = x_1 + x_2 - 0.5$$

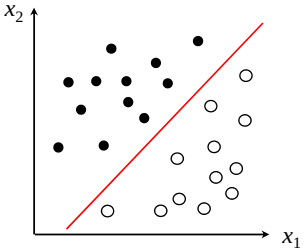


p_1 (0,0): $y = 0$
p_2 (1,0): $y = 0$
p_3 (0,1): $y = 0$
p_4 (1,1): $y = 1$

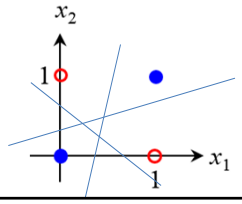
신경망

❖ **퍼셉트론**(Perceptron)

- **선형 분리가능 문제** (linearly separable problem)



- **선형 분리불가 문제**(linearly inseparable problem)
  - **XOR**(exclusive OR) **문제**



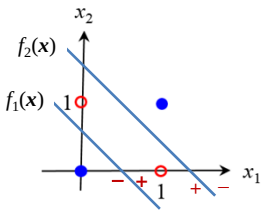
(0,0): 0
(1,0): 0
(0,1): 0
(1,1): 1

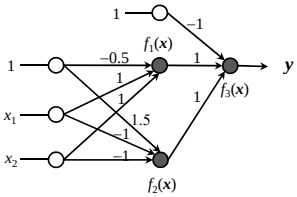
Minsky & Papert 1969

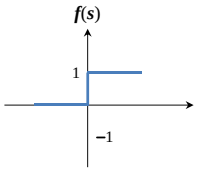
신경망

❖ **다층 퍼셉트론**(multilayer Perceptron, **MLP**)

- 여러 개의 퍼셉트론을 층 구조로 구성한 신경망 모델







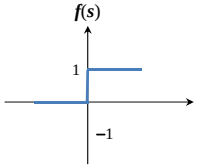
$$y = f(s) = f(\sum_{i=1}^{2} w_i x_i + b) = f(\boldsymbol{w}\boldsymbol{x}^T)$$

신경망

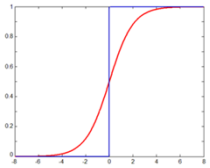
❖ **다층 퍼셉트론**(multilayer Perceptron, **MLP**) – cont.

- **다층 퍼셉트론의 학습**
  - 입력-출력 $(\boldsymbol{x}_i, y_i)$의 학습데이터에 대해서 출력값 $f(\boldsymbol{x}_i)$의 차이, **오차** (error)가 최소가 되도록 가중치 $\boldsymbol{w}$를 결정하는 것

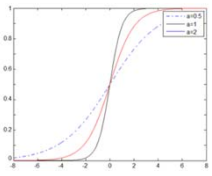
- **학습 가능한 다층 퍼셉트론**
  - 학습알고리즘 : **오차역전파** 알고리즘 발표 (Rumelhart&McClelland 1986)
  - 계단모양 전달함수를 미분가능한 시그모이드(sigmoid) 함수로 대체



계단 (step) 함수



시그모이드(sigmoid) 함수



$$f(s) = \frac{1}{1+e^{-as}}$$

$$f'(s) = \frac{ae^{-as}}{(1+e^{-as})^2} = af(s)(1-f(s))$$

신경망

❖ 다층 퍼셉트론 MLP의 동작

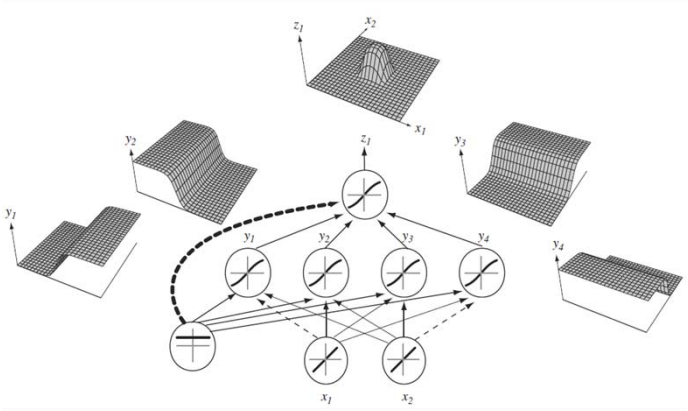
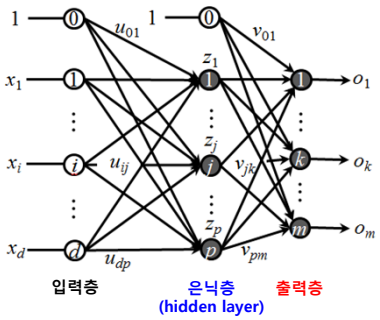


Image source : Pascal Vincent

신경망

❖ 다층 퍼셉트론 MLP의 동작



입력 $(x_1, x_2, \cdots, x_d)$

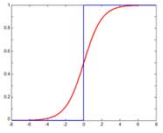
출력 $(y_1, y_2, \cdots, y_m)$

은닉층 $zsum_j = \sum_{i=1}^{d} u_{ij} x_i + u_{0j} \quad (1 \le j \le p)$

$z_j = f(zsum_j)$

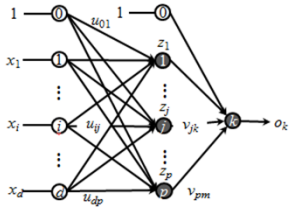
출력층 $osum_k = \sum_{j=1}^{p} v_{jk} z_k + v_{0k} \quad (1 \le k \le m)$

$o_k = f(osum_k)$



신경망

❖ 다층 퍼셉트론 MLP의 학습



입력 : $(x_1, x_2, \ldots, x_d)$

기대 출력 : y_k

MLP 출력 : o_k

- **학습 목표**
  - **기대 출력**과 **MLP 출력**이 최대한 **비슷**해지도록 **가중치**를 **변경**하는 것

$$E = \frac{1}{2}(o_k - y_k)^2$$

  - **최대 경사법**(gradient descent method) 사용

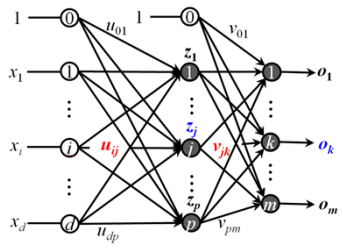
$$v_{jk}^{(t+1)} = v_{jk}^{(t)} - \eta \frac{\partial E}{\partial v_{jk}}$$

$$u_{ij}^{(t+1)} = u_{ij}^{(t)} - \eta \frac{\partial E}{\partial u_{ij}}$$

신경망

❖ 다층 퍼셉트론 MLP의 학습

- **오차 역전파 알고리즘**(Error back propagation algorithm, Backprop algorithm)



입력 $(x_1, x_2, \cdots, x_d)$

출력 $(y_1, y_2, \cdots, y_m)$

$$E = \frac{1}{2}\sum_{k=1}^{n}(o_k - y_k)^2 \quad \text{오차함수}$$

$$\boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \eta \frac{\partial E}{\partial \boldsymbol{v}}$$

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} - \eta \frac{\partial E}{\partial \boldsymbol{u}}$$

$$osum_k = \sum_{j=1}^{p} v_{jk} z_k + v_{0k} \quad (1 \le k \le m)$$

$$o_k = f(osum_k)$$

$$zsum_j = \sum_{i=1}^{d} u_{ij} x_i + u_{0j} \quad (1 \le j \le p)$$

$$z_j = f(zsum_j)$$

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial v_{jk}} = (o_k - y_k) f'(osum_k) z_j = \delta_k z_j$$

$$\frac{\partial E}{\partial u_{ij}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial u_{ij}} = \sum_{k=1}^{m}\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial z_j} f'(zsum_j) x_i$$

$$= \sum_{k=1}^{m}(o_k - y_k) f'(osum_k) v_{jk} f'(zsum_j) x_i$$

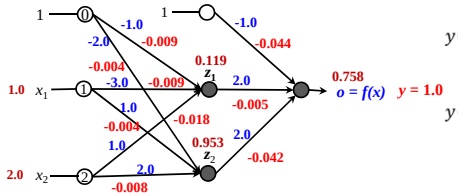
$$= \sum_{k=1}^{m}\delta_k v_{jk} f'(zsum_j) x_i$$

신경망

❖ 다층 퍼셉트론 MLP의 학습

▪ 오차 역전파 알고리즘 – cont.

$$y(x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$y(x_1, x_2) = \frac{1}{1 + e^{-(-1.0 - 3.0x_1 + x_2)}}$$

$$y(1.0, 2.0) = \frac{1}{1 + e^{-(-1.0 - 3.0(1.0) + 20.0)}} = 0.119$$

$$E = \frac{1}{2}(o - y)^2$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{\partial f(x)}{\partial x} = \frac{e^{-x}}{1 + e^{-x}} = f(x)(1 - f(x))$$

$$\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial v_{jk}} = (o_k - y_k) f'(osum_k) z_j = \delta_k z_j$$

$$\frac{\partial E}{\partial u_{ij}} = \sum_{k=1}^{m} (o_k - y_k) f'(osum_k) v_{jk} f'(zsum_j) x_i$$

$$= \sum_{k=1}^{m} \delta_k v_{jk} f'(zsum_j) x_i$$

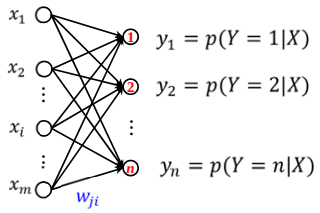
$$v_{jk}^{(t+1)} = v_{jk}^{(t)} - \eta \frac{\partial E}{\partial v_{jk}} \qquad u_{ij}^{(t+1)} = u_{ij}^{(t)} - \eta \frac{\partial E}{\partial u_{ij}}$$

신경망

❖ 소프트맥스 층 (softmax layer)

▪ 최종 출력을 **분류 확률**(classification probability)로 변환하는 층

• 출력의 합 = 1



$$y_1 = p(Y = 1|X)$$

$$y_2 = p(Y = 2|X)$$

$$y_n = p(Y = n|X)$$

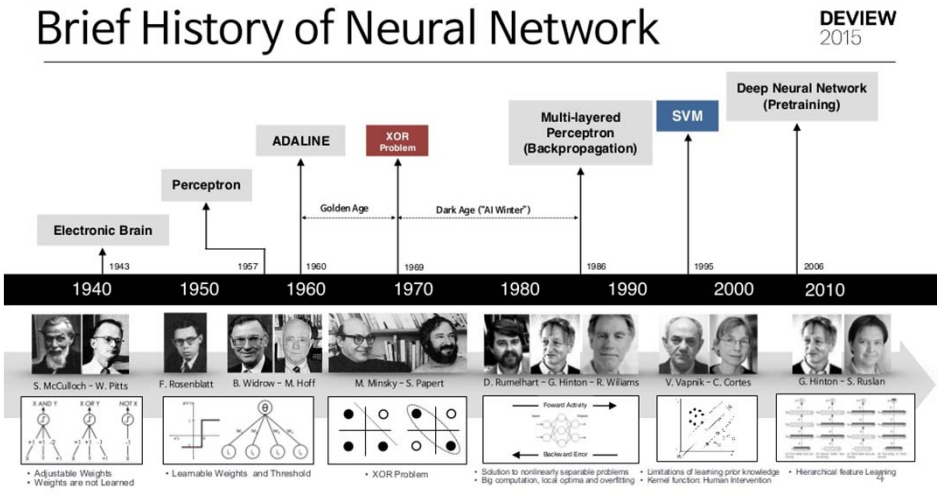
$$y_k = \frac{e^{\sum_{j=1}^{m} w_{jk} x_j}}{\sum_{l=1}^{n} e^{\sum_{j=1}^{m} w_{jl} x_j}}$$

▪ 오차 함수

• **목표 분포 p**와 **출력분포 q**의 **교차 엔트로피**(cross entropy)

$$H(\boldsymbol{p}, \boldsymbol{q}) = -\sum_{i=1}^{n} [p(x_i) \log q(x_i) + (1 - p(x_i)) \log(1 - q(x_i))]$$

신경망의 역사

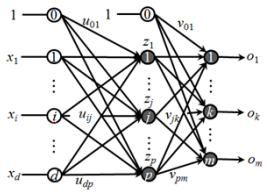


출전: 이예하

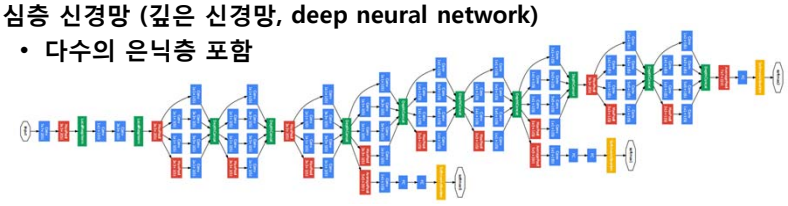
9. 딥러닝

❖ **딥러닝**(deep learning, **심층학습**, **깊은 학습**, **심화학습**)

- **일반 신경망**
  - **소수의 은닉층 포함**



- **심층 신경망 (깊은 신경망, deep neural network)**
  - **다수의 은닉층 포함**

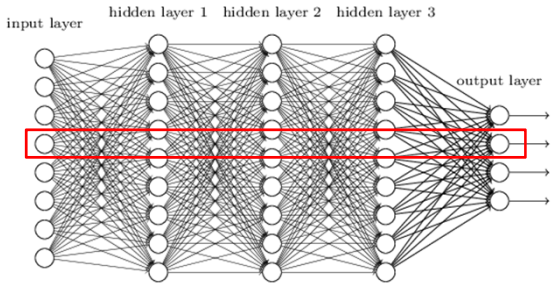


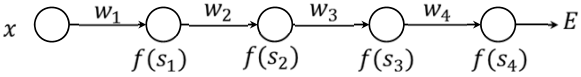
출전: googLeNet (google)

딥러닝

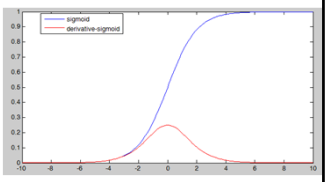
❖ **심층 신경망 학습의 어려움**

- **기울기 소멸 문제(vanishing gradient problem)**
  - 은닉층이 많은 다층 퍼셉트론에서, 출력층에서 아래 층으로 갈 수록 전달되는 오차가 크게 줄어들어, 학습이 되지 않는 현상





$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f(s_4)} f'(s_4)\ w_4\ f'(s_3)\ w_3\ f'(s_2)\ w_2\ f'(s_1)\ x$$



http://neuralnetworksanddeeplearning.com/chap6.html

딥러닝

❖ **심층 신경망 학습 전략**

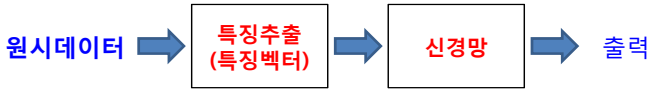
- **사전 학습(pre-training)**
  - 가중치를 무작위로 초기화하는 대신
  - 데이터를 사용하여 비지도학습에 의한 초기 가중치 결정
- **층(layer)간 노드 연결 수 제한**
  - 직전 층의 모든 노드와 연결하는 대신
  - 일부 지역적 노드와 연결
- **가중치 공유**
  - 같은 층의 여러 노드가 **동일한 가중치 사용**
- **전달함수 변경**
  - 그레디언트(gradient) 값이 0에 가까워지는 것을 회피하기 위해
  - **시그모이드**(sigmoid) 함수 대신 **부분적 선형 함수**(linear function) 사용
- **부분적 가중치 학습**
  - 각 학습 단계에서 모든 가중치를 조정하는 대신
  - 무작위로 선택한 **일부 가중치**만 조정

http://neuralnetworksanddeeplearning.com/chap6.html

딥러닝

❖ **일반 신경망과 심층 신경망**

- **일반 신경망 모델**
  - 원시 데이터(original data)에서 직접 특징(**handcrafted feature**)을 추출해서 만든 **특징 벡터**(feature vector)를 입력으로 사용
  - 특징 벡터들의 품질에 영향



- **심층 신경망**
  - **특징추출**과 **학습**을 함께 수행
  - 데이터로부터 **효과적인 특징**을 학습을 통해 추출 ➔ 우수한 성능

딥러닝

❖ **딥러닝**(deep learning, **심층학습, 깊은 학습, 심화학습**)

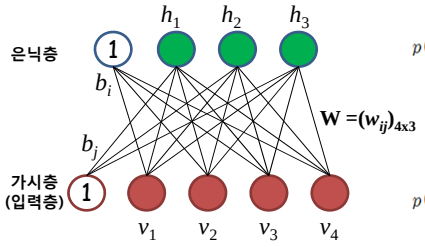
- **심층 신경망**을 사용하여 **특징추출**과 **문제해결**을 위한 **학습**을 동시에 하는 기계학습 방법

- 대표적인 깊은 학습 신경망
  - **DBN**(Deep Belief Network, DBN)
  - **오토인코더**(autoencoder)
  - **컨볼루션 신경망**(Convolutional Neural Network, CNN)
  - **LSTM RNN**(Long short-term memory RNN)

9.1 Deep Belief Network

❖ **DBN(Deep Belief Network) 모델**

- **Restricted Boltzmann Machine**(제한된 볼쯔만 머신, RBN)을 여러 층으로 쌓아 만든 심층 신경망
  - **RBM 모델**
    - **이진 입력 v 및 출력 h**



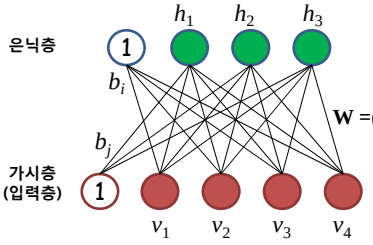
$$p(h_j = 1|V) = sigm(b_j + \sum_i w_{ij} v_i)$$

$$p(v_i = 1|H) = sigm(b_i + \sum_j w_{ij} h_j)$$

Deep Belief Network

❖ **DBN(Deep Belief Network) 모델**

- **RBN 모델의 동작**



$$p(h_j = 1|V) = sigm(b_j + \sum_i w_{ij} v_i)$$

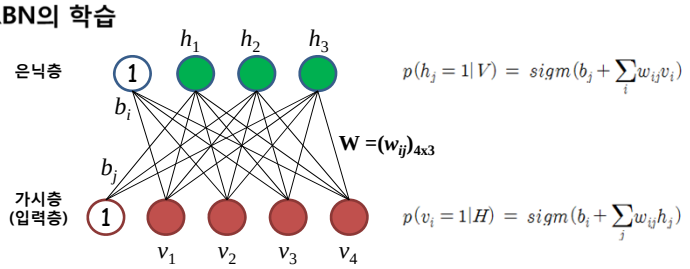
$$p(v_i = 1|H) = sigm(b_i + \sum_j w_{ij} h_j)$$

- **입력** v를 입력층에 부가
- 입력 v를 사용하여 은닉층의 확률 $p(h|v)$ 계산
- $p(h|v)$의 확률로 은닉층의 값 h 결정
- 은닉층 h를 사용하여 입력층 확률 $p(v|h)$ 계산
- $p(v|h)$의 확률로 입력층의 값 v 결정

Deep Belief Network

❖ **DBN 모델 – cont.**

- **RBN의 학습**



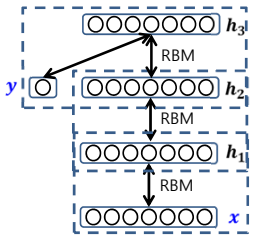
- V층의 노드값이 원래 입력값을 그대로 재구성하도록 즉, 같은 값이 나오도록 가중치를 결정하는 것
- **(H 노드 계산 → V 노드 계산 → H 노드 계산 → V 노드 계산)** 반복
- [**V노드 계산 → H노드 계산**] 단계에서 얻어진 $v_i h_j$의 값과 다음 단계의 $v_i' h_j'$ 차이를 최소화하도록 학습

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta(v_i h_j - v_i' h_j')$$

Deep Belief Network

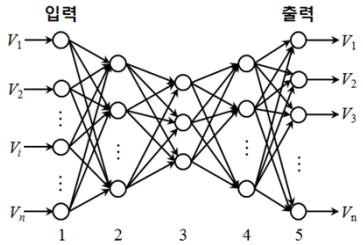
❖ **DBN 모델 – cont.**

- **DBN의 학습**
  - 층별로 학습이 점진적으로 이루어지는 **층별 학습**(layer-wise learning)



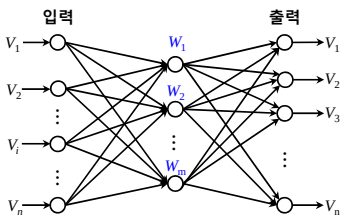
9.2 오토인코더

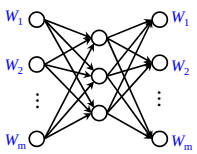
❖ **오토인코더**(authoencoder)
- 입력층의 노드 개수와 출력층의 노드 개수가 같은 다층 퍼셉트론
- 입력과 출력을 동일하게 하여 학습
- 입력과 같은 출력이 나오도록 학습한 다층 퍼셉트론

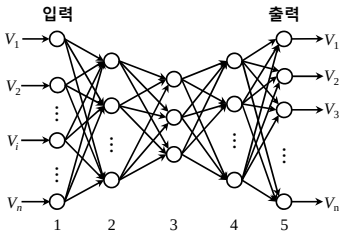


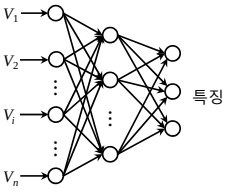
오토인코더

❖ **오토인코더 – cont.**
- **오토인코더의 학습**





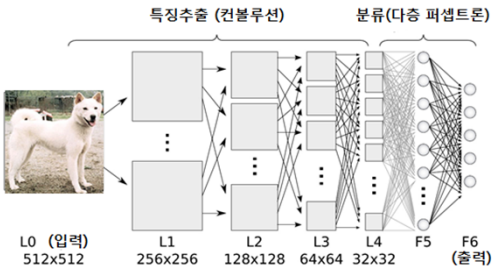




- 3번째 층을 입력에 대해 **특징 추출**(feature extraction)한 것으로 볼 수 있음

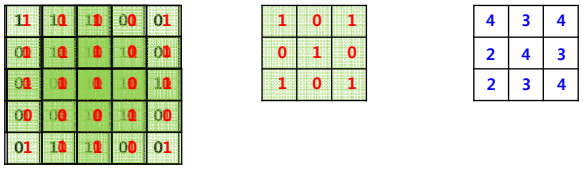
9.3 컨볼루션 신경망 CNN

❖ **컨볼루션 신경망**(Convolutional Neural Network, **CNN**)
- 전반부 : 컨볼루션 연산을 수행하여 **특징 추출**
- 후반부 : 특징을 이용하여 **분류**
- 영상분류, 문자 인식 등 인식문제에 높은 성능

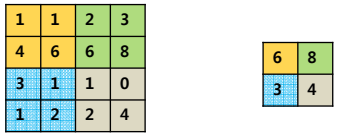


컨볼루션 신경망 CNN

❖ **CNN의 연산**
- **컨볼루션(convolution) 연산**



- **Max pooling (subsampling)연산**

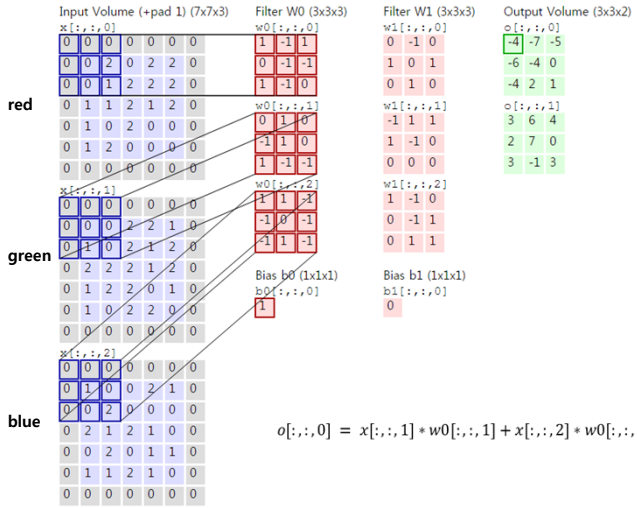


- **ReLU (rectified Linear Unit)**
  - $f(x) = max\{0, x\}$

컨볼루션 신경망 CNN

❖ **CNN의 연산**

- **3차원 입력에 대한 컨볼루션 연산의 예**



$$o[:,:,0] = x[:,:,1] * w0[:,:,1] + x[:,:,2] * w0[:,:,2] + x[:,:,3] * w0[:,:,3]$$

image : cs231n.stanford.edu

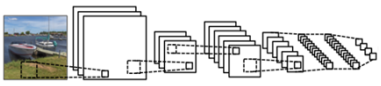
컨볼루션 신경망 CNN

❖ **CNN의 특징**

- **지역적 연결(local connection)**
  - 이전 층의 일부 뉴런에 대해서만 연결

- **가중치 공유(shared weight)**
  - 같은 층에서는 동일한 컨볼루션 필터 사용
- **풀링(pooling) 사용**
  - 위치가 다른 부분에 나온 유사한 특징을 결합
  - 특징공간을 축소

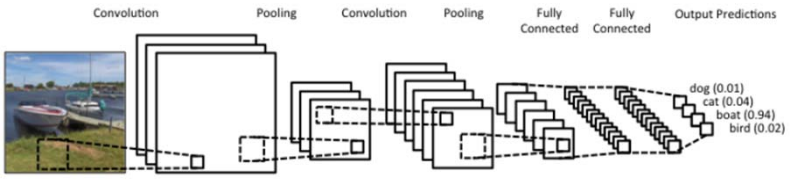
- **다층 사용**
  - 추상화된 다층의 특징 추출

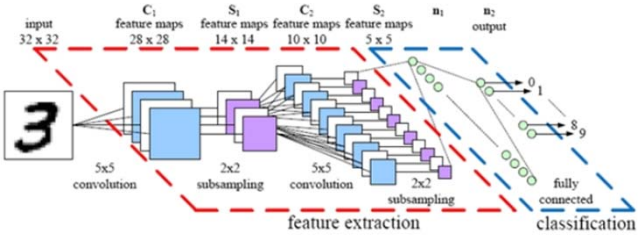


images: www.kdnuggets.com

컨볼루션 신경망 CNN

❖ CNN의 구조





images: www.kdnuggets.com http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

컨볼루션 신경망 CNN

❖ CNN을 이용한 영상 분류

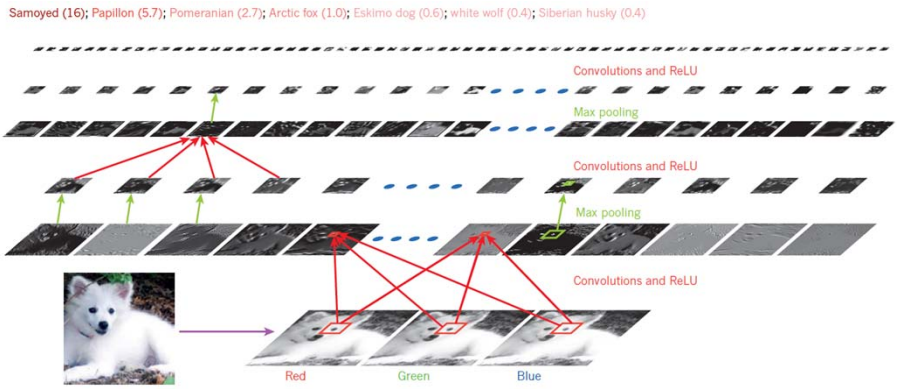
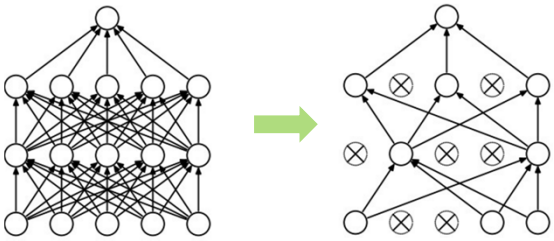


Image : Nature

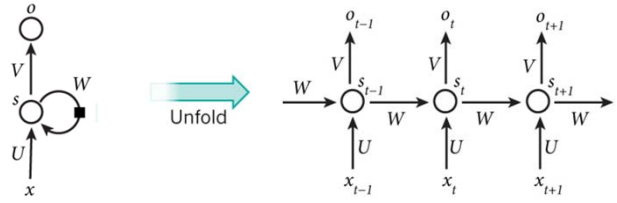
컨볼루션 신경망 CNN

❖ **컨볼루션 신경망 CNN의 학습**
- **오차 역전파 알고리즘** 사용
- **dropout 방법**
  - $1-p$의 확률로 노드를 학습에서 무시
  - 노드에 연결된 에지가 없는 것으로 간주
  - **과적합**(overfitting) 문제 완화
  - 학습 속도 향상



9.4 LSTM-RNN 모델

❖ **RNN(Recurrent Neural Network, 재귀 신경망) 모델**
- 이전 출력이 입력으로 들어가는 것이 있는 신경망



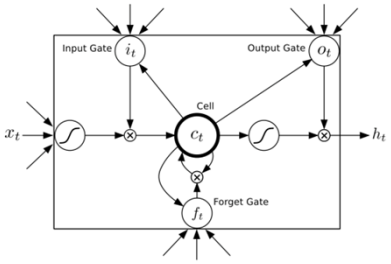
- **RNN 학습**
  - 오차역전파 알고리즘 사용
  - **기울기 소멸 문제**(vanishing gradient problem) 심각

Image : Nature

LSTM-RNN 모델

❖ **LSTM-RNN (**Long Short-Term Memory Recurrent Neural Network**)모델**

- **LSTM(Long Short-term memory) cell**
  - 뉴론 내에서 값을 저장하는 역할
  - 값의 **입력**(input), **출력**(output), **망각**(forget)을 관리하는 **문지기** (**gate**) 회로를 가지고 있음
  - 기울기 소멸 문제 해결



- 언어 모델(language model), 음성인식, 언어 번역 등에 우수한 성능

image: Alex Graves

딥러닝

❖ **깊은 학습의 특징**

- 첫째, 여러 개의 은닉층을 갖는다.
- 둘째, 신경망에서 특징 추출을 담당한다.
- 셋째, 계층적으로 볼 때 위쪽으로 올라갈수록 더 추상화된 특징을 추출한다.
- 넷째, 학습이 점진적으로 이루어진다.
- 다섯째, 학습하는데 많은 계산량을 필요로 한다.
- 여섯째, 학습 데이터의 수가 많아져도 적용할 수 있기 때문에, 빅데이터 처리에도 사용할 수 있다.

딥러닝

❖ **딥러닝 개발 환경**

- **라이브러리**
  - Caffe
  - Torch
  - Theano
  - RNNLIB
  - Cuda-ConvNet
  - Pylarn
  - DL4J
  - TensorFlow

- **계산 환경**
  - CUDA
  - Apache Spark