빅데이터 분석 실무과정

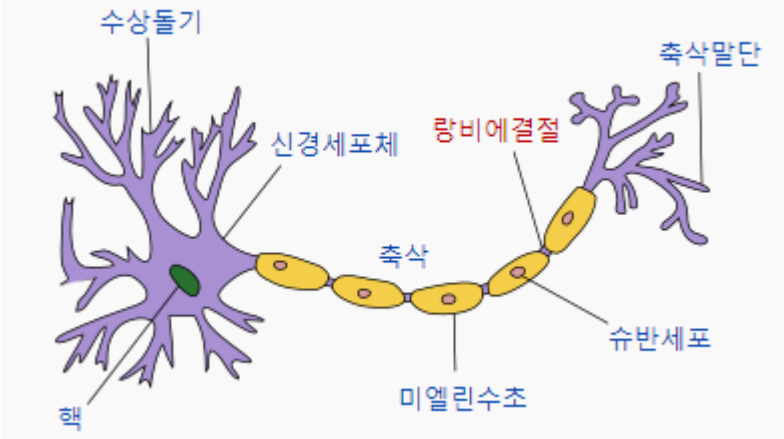
보충강의 : 딥러닝

2018.6.14 김현호

딥러닝 (Deep Learning)

딥러닝은 인간 뇌, 특히 뉴런의 구조를 흉내낸 기술이다

- 수상돌기에 다른 뉴런의 축삭돌기 들이 접속하여 전기신호를 보낸다
- **동시에 일정값 이상**의 전기신호가 수상돌기에 감지되면 신호는 핵으로 전달된다
- 핵은 신호를 처리하여 축삭돌기를 통해 다른 뉴런으로 신호를 전달한다
- 인간의 대뇌피질은 6층 정도의 신호전달 체계를 가지고 있다

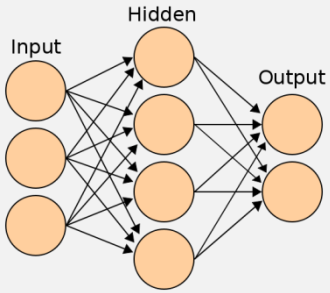


https://ko.wikipedia.org/wiki/신경_세포

딥러닝 (Deep Learning)

딥러닝 핵심 기술들

ANN(Artificial Neural Network)



DNN(Deep Neural Network)

CNN

RNN

DQN

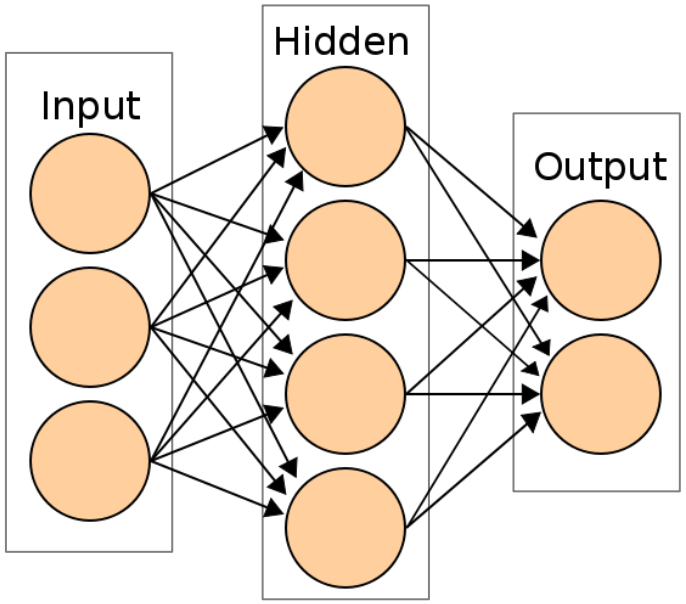
- CNN : Convolutionary Neural Network (이미지 처리)
- RNN : Recurrent Neural Network (자연어 처리)
- DQN : Deep Q-Network (강화학습)

ANN (Artificial Neural Network)

인공신경망 또는 그냥 신경망이라고 부른다

- 입력층(Input), 중간층(Hidden), 출력층(Output) 으로 이루어진다
- 중간층은 여러 층(Layer)로 구성될 수 있다 (또는 없을 수 있다)
- 입력층의 뉴런 갯수는 데이터의 속성 갯수이다
- 중간층의 뉴런 갯수는 경험적으로 정한다 (노하우)
- 출력층의 갯수는 일반적으로 분류할 클래스 갯수이다

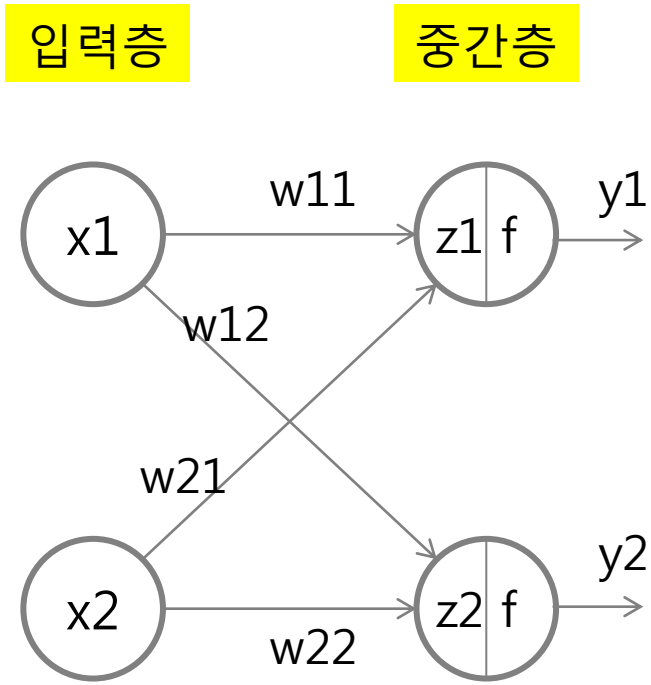
  (숫자이면 0~9 까지의 10개)



https://en.wikipedia.org/wiki/Artificial_neural_network

ANN (Artificial Neural Network)

계산 예시



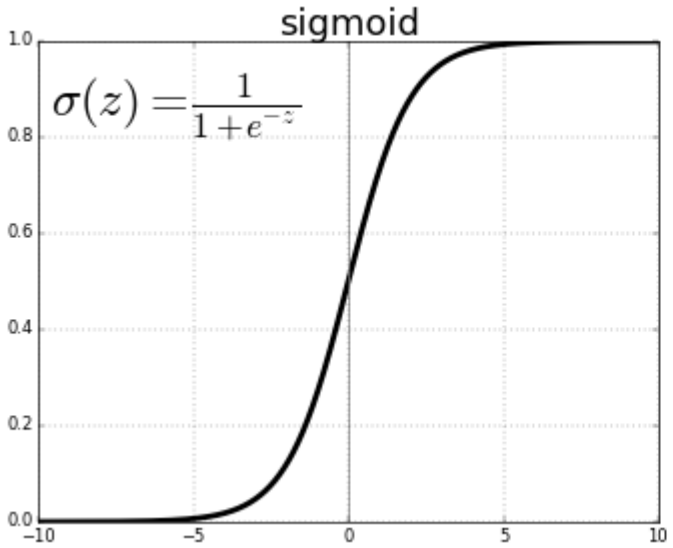
z1 = w11 * x1 + w21 * x2
y1 = f(z1)
= f(w11*x1+w21*x2)

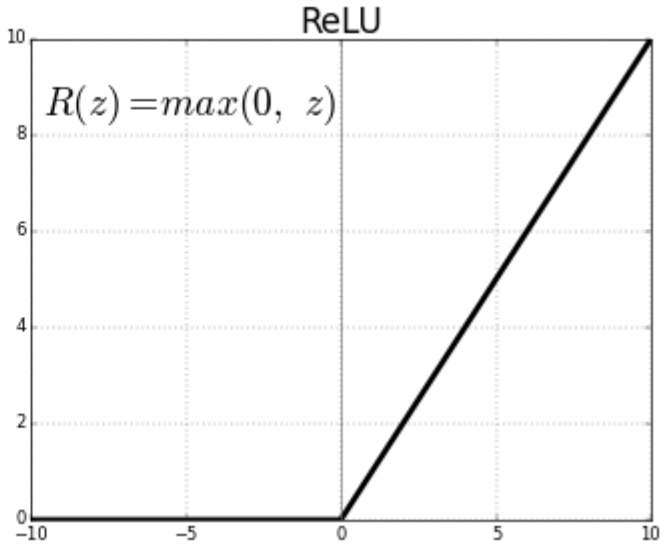
z2 = w12 * x1 + w22 * x2
y2 = f(z2)
= f(w12*x1+w22*x2)

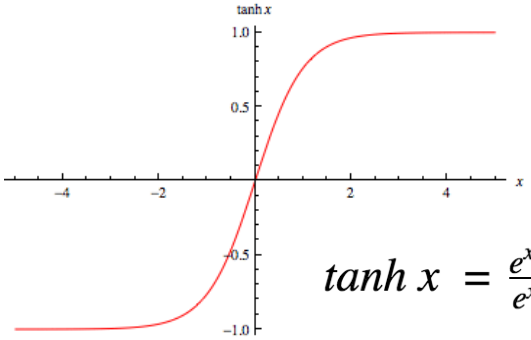
* f 는 활성화 함수
(가장 간단한 경우는 f(z)=z)

ANN (Artificial Neural Network)

활성화 함수 종류







$$tanh\ x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ANN (Artificial Neural Network)

신경망 문제 해결 방법

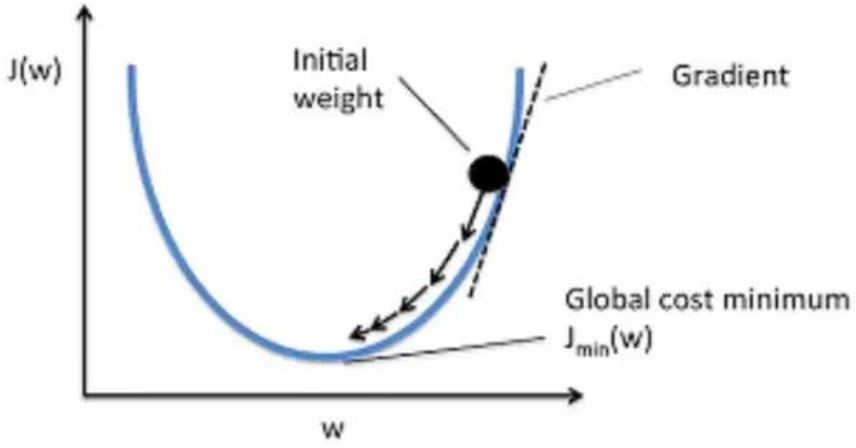
- 입력값(X) 와 타겟값(T) 는 고정된 상수값이다 (iris 데이터의 샘플과 레이블 값들)
- 신경망의 가중치값인 W 를 최적화하는 것이 문제이다
- 해결 방법

1. W 를 초기화한다. (0 또는 랜덤값)
2. X 와 W 를 이용하여 출력값인 Y 를 구한다
3. 타겟값인 T 와 출력값 Y 를 가지고 비용함수(cost function) 를 구한다 ➔ 크로스 엔트로피
4. 비용함수 값을 줄이도록 W 값을 변경한다 ➔ 경사하강법
5. 2번 부터 반복

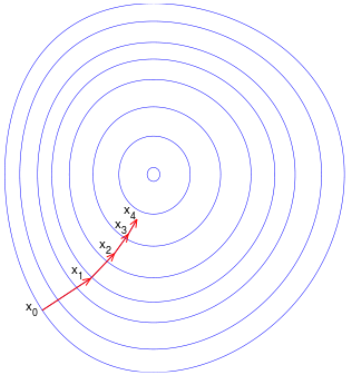
ANN (Artificial Neural Network)

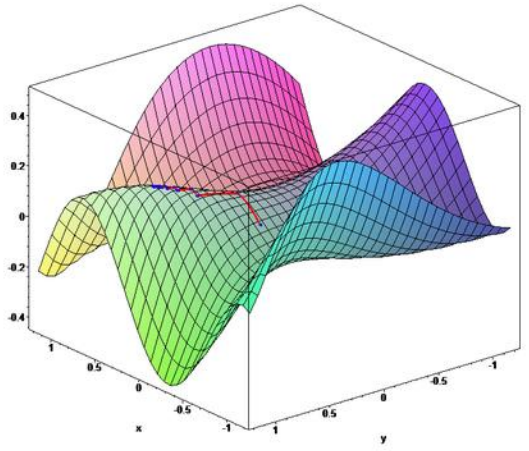
경사하강법

- w → w - 학습률 * (w-공간 에서의 기울기)



https://www.quora.com/What-is-Stochastic-Gradient-Descent

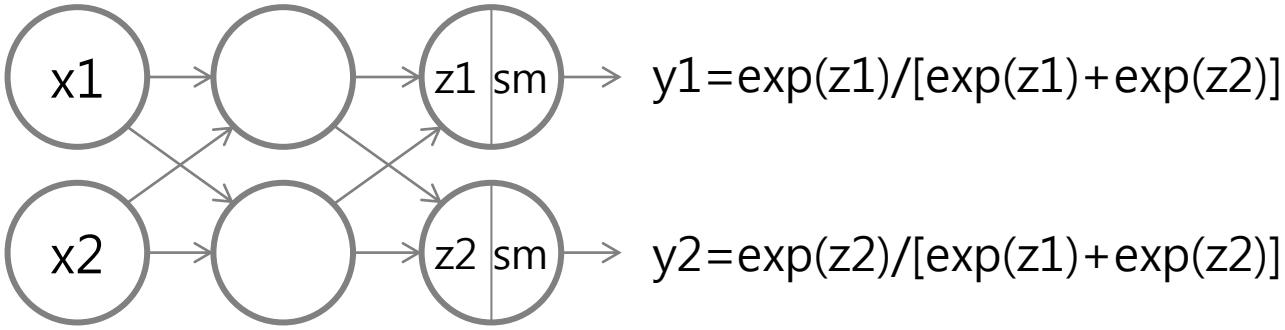




ANN (Artificial Neural Network)

분류 문제의 출력층 처리 (소프트맥스와 크로스 엔트로피)

- 출력층의 입력값이 (z1, z2) 이고, 타겟값 (t1,t2)가 (1,0) 또는 (0,1)
- 아래 그림과 같이 '소프트맥스' 를 적용하면, 출력값이 확률값으로 변환된다



- 크로스 엔트로피

cost = - [t1*log(y1) + t2*log(y2)]

= -[1*log(y1) + 0*log(y2)] = -log(y1) (* 타겟값이 (1,0) 일때)

ANN (Artificial Neural Network)

역전파 (Backpropagation)

- 경사하강법에서 W 값을 변경하기 위한 기울기를 구하는 방법임
- 출력층의 값을 가지고 단계적으로 중간층을 내려가며 W 값을 바꿔나간다

경사하강법 종류

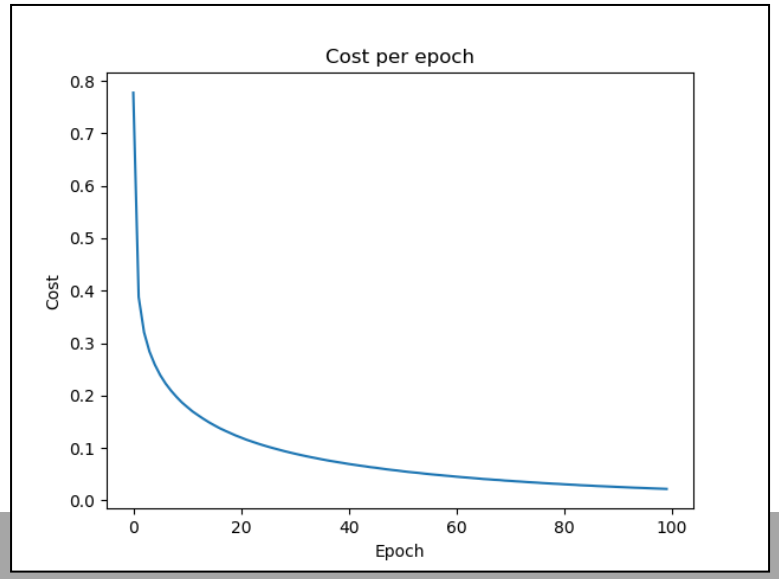
- batch gradient descent : 전체 데이터셋에 대해 W 변경 후 반복
- stochastic gradient descent(SGD) : 한개의 데이터에 대해서 W 변경
- 미니배치(mini-batch) : 전체 데이터셋의 부분집합에 대해 W 변경 후 반복

ANN – 예제

```
'''
MNIST
two hidden layer
softmax, cross entropy, SGD, minimax
'''

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data
mnist=input_data.read_data_sets('./mnist/data/',one_hot=True)

X=tf.placeholder(tf.float32, [None,784])
y=tf.placeholder(tf.float32, [None,10])

W1=tf.Variable(tf.random_normal([784,256],stddev=0.1))
b1=tf.Variable(tf.random_normal([256],stddev=0.1))
hidden_1=tf.nn.relu(tf.matmul(X,W1)+b1)

W2=tf.Variable(tf.random_normal([256,256],stddev=0.1))
b2=tf.Variable(tf.random_normal([256],stddev=0.1))
hidden_2=tf.nn.relu(tf.matmul(hidden_1,W2)+b2)

W3=tf.Variable(tf.random_normal([256,10],stddev=0.1))
b3=tf.Variable(tf.random_normal([10],stddev=0.1))
output=tf.nn.softmax(tf.matmul(hidden_2,W3)+b3)

cost=tf.reduce_mean(tf.reduce_sum(-y*tf.log(output),1))

optimizer=tf.train.GradientDescentOptimizer(learning_rate=0.01)
train_op=optimizer.minimize(cost)

cost_list=[]
```

ANN - 예제

```
with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        batch_size=100
        total_batch=int(mnist.train.num_examples/batch_size) # 550

        for epoch in range(100):
                total_cost=0

                for i in range(total_batch):
                        batch_X, batch_y = mnist.train.next_batch(batch_size)
                        _, rcost = sess.run([train_op,cost],feed_dict={X: batch_X, y: batch_y})
                        total_cost+=rcost

                cost_list.append(total_cost/total_batch)
                print('Epoch: %03d, Avg. cost = %f' % (epoch,total_cost/total_batch))

        pred_y=sess.run(tf.argmax(output,1), feed_dict={X: mnist.test.images})
        accuracy=np.mean(pred_y==np.argmax(mnist.test.labels,1))
        print('Accuracy = %.2f%%' % (accuracy*100))

plt.plot(cost_list)
plt.title('Cost per epoch')
plt.xlabel('Epoch')
plt.ylabel('Cost')
plt.show()
```



DNN (Deep Neural Network)

- **DNN 은 중간층과 뉴런의 갯수를 획기적으로 늘린 것이다**
  - 컴퓨팅 파워
  - 기존 기술적 난점들을 해결 (역전파, 과적합, 속도향상 등)
- **DNN 은 인간 뇌의 감각 및 인지 기능에 주목하였다 (시청각, 언어처리 등)**
  - CNN (이미지)
  - RNN (언어, 소리)

CNN (Covolutional Neural Network)

CNN 은 이미지 처리를 위한 신경망이다

- CNN = 영상처리기술 + 뇌과학(인지과학) + DNN
- 영상처리기술 발달 : 필터링, 리샘플링, 외곽선인식 등
- 뇌과학 발달 : 인간 뇌의 시신경 구조

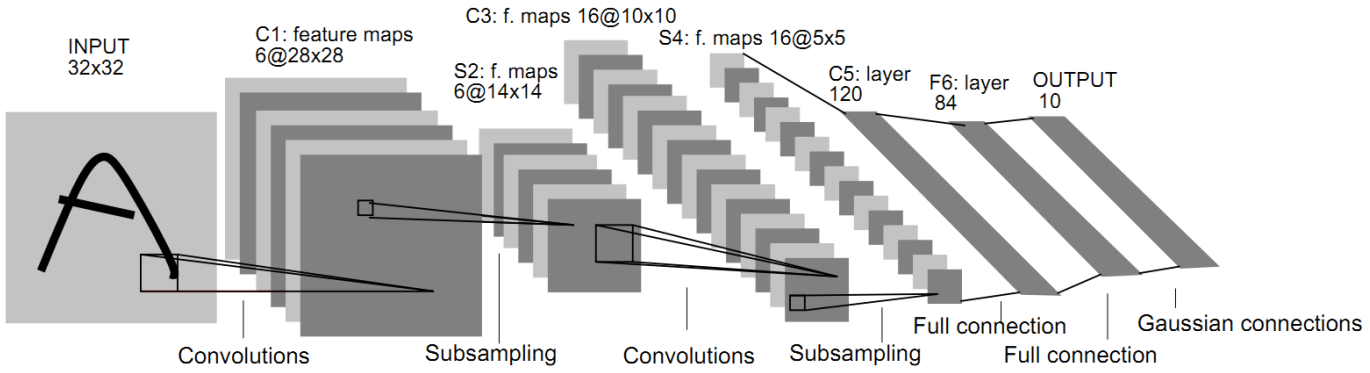


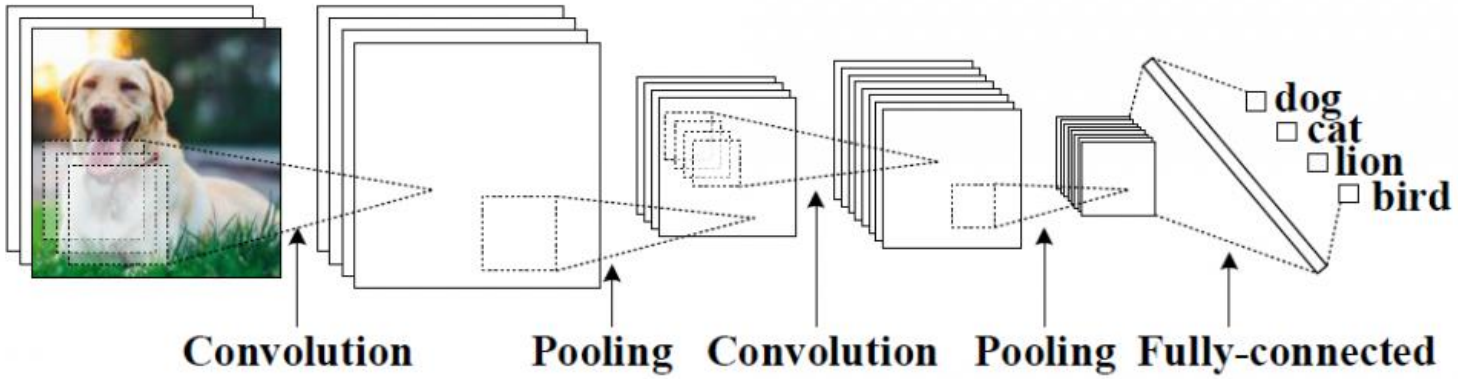
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

http://yann.lecun.com/exdb/lenet/

CNN (Covolutional Neural Network)

CNN 은 Convolution 과 Pooling 레이어로 이루어져있다

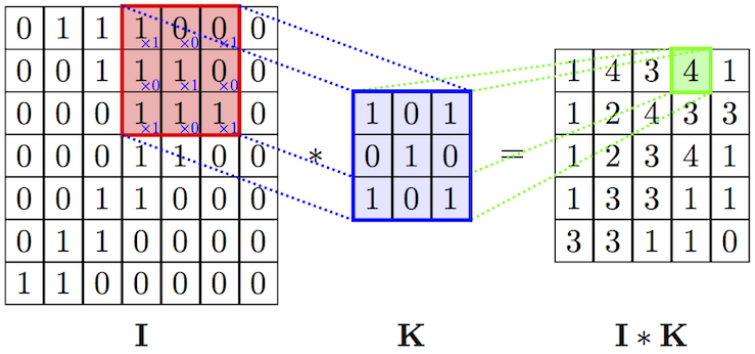
- 입력 : 3채널(RGB)의 컬러 이미지
- Convolution Layer : 입력이미지에 여러개의 작은 필터를 적용하여 신규 이미지 생성
- Pooling Layer : 입력 이미지의 너비와 폭을 줄인다



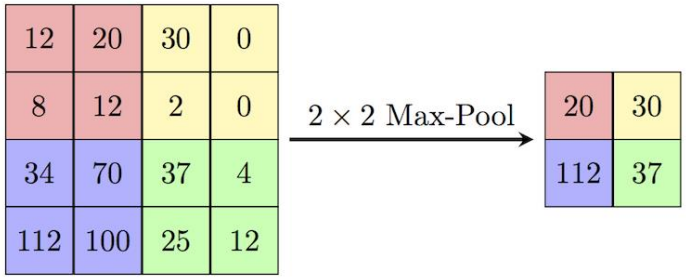
https://ren-fengbo.lab.asu.edu/content/gpu-outperforming-fpga-accelerator-architecture-binary-convolutional-neural-networks

CNN (Covolutional Neural Network)

Convolution (필터링)



Pooling (sub-samplling)



이미지 출처 : https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

CNN - 예제

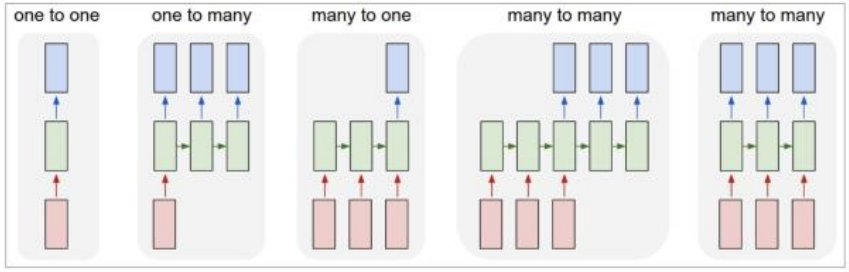
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data
mnist=input_data.read_data_sets('./mnist/data/',one_hot=True)

X=tf.placeholder(tf.float32, [None,28,28,1])
y=tf.placeholder(tf.float32, [None,10])

W1=tf.Variable(tf.random_normal([3,3,1,32],stddev=0.01))
b1=tf.Variable(tf.random_normal([32],stddev=0.01))
conv_1=tf.nn.conv2d(X,W1,strides=[1,1,1,1],padding='SAME')+b1 # [N,28,28,32]
conv_1_relu=tf.nn.relu(conv_1)
pool_1=tf.nn.max_pool(conv_1_relu,ksize=[1,2,2,1],strides=[1,2,2,1],padding='SAME') # [N,14,14,32]

W2=tf.Variable(tf.random_normal([3,3,32,64],stddev=0.01))
b2=tf.Variable(tf.random_normal([64],stddev=0.01))
conv_2=tf.nn.conv2d(pool_1,W2,strides=[1,1,1,1],padding='SAME')+b2 # [N,14,14,64]
conv_2_relu=tf.nn.relu(conv_2)
pool_2=tf.nn.max_pool(conv_2_relu,ksize=[1,2,2,1],strides=[1,2,2,1],padding='SAME') # [N,7,7,64]

W3=tf.Variable(tf.random_normal([7*7*64,256],stddev=0.1))
b3=tf.Variable(tf.random_normal([256],stddev=0.1))
hidden_1=tf.nn.relu(tf.matmul(tf.reshape(pool_2,[-1,7*7*64]),W3)+b3)

W4=tf.Variable(tf.random_normal([256,10],stddev=0.1))
b4=tf.Variable(tf.random_normal([10],stddev=0.1))
output=tf.nn.softmax(tf.matmul(hidden_1,W4)+b4)

cost=tf.reduce_mean(tf.reduce_sum(-y*tf.log(output),1))

optimizer=tf.train.AdamOptimizer(learning_rate=0.001)
train_op=optimizer.minimize(cost)

cost_list=[]
```

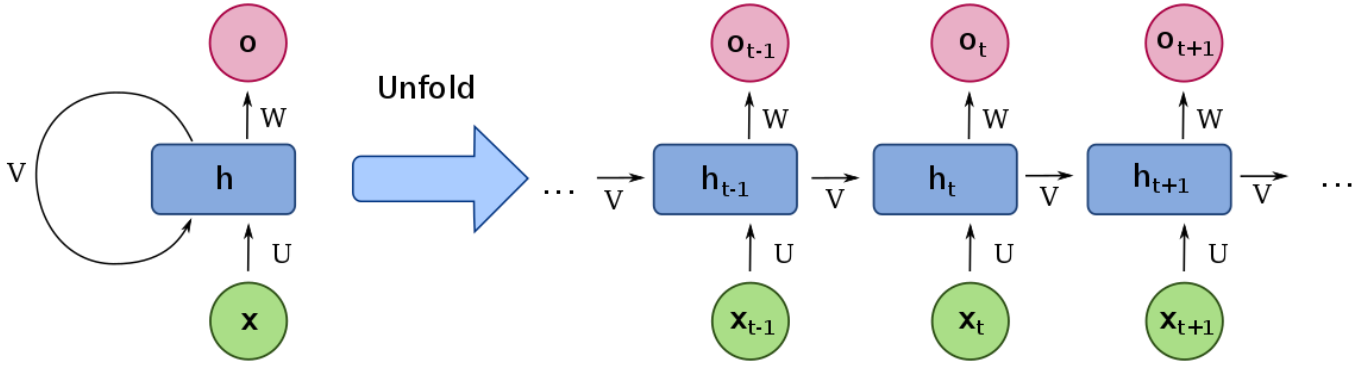
CNN - 예제

```
with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        batch_size=100
        total_batch=int(mnist.train.num_examples/batch_size) # 550

        for epoch in range(10):
                total_cost=0

                for i in range(total_batch):
                        batch_X, batch_y = mnist.train.next_batch(batch_size)
                        _, rcost = sess.run([train_op,cost],feed_dict={X: batch_X.reshape(-1,28,28,1), y: batch_y})
                        total_cost+=rcost
                        print('### %d-%d %f' % (epoch,i,rcost))

                cost_list.append(total_cost/total_batch)
                print('Epoch: %03d, Avg. cost = %f' % (epoch,total_cost/total_batch))

        pred_y=sess.run(tf.argmax(output,1), feed_dict={X: mnist.test.images.reshape(-1,28,28,1)})
        accuracy=np.mean(pred_y==np.argmax(mnist.test.labels,1))
        print('Accuracy = %.2f%%' % (accuracy*100))

'''
# Epoch: 009, Avg. cost = 0.008241
# Accuracy = 99.15%

# cost_list
[0.26349139610474759,
 0.060066648079082373,
 0.040152242099866273,
 0.030062449230304496,
 0.022707773680079053,
 0.017138848179707896,
 0.013540809828396463,
 0.011449726234740493,
 0.011226309404228231,
 0.008240539504093414 4]
'''
```

RNN (Recurrent Neural Network)

- RNN 은 시계열 데이터(순서가 있는 데이터)에 특화되어 있다
  - RNN 이해하기 (https://dreamgonfly.github.io/rnn/2017/09/04/understanding-rnn.html)



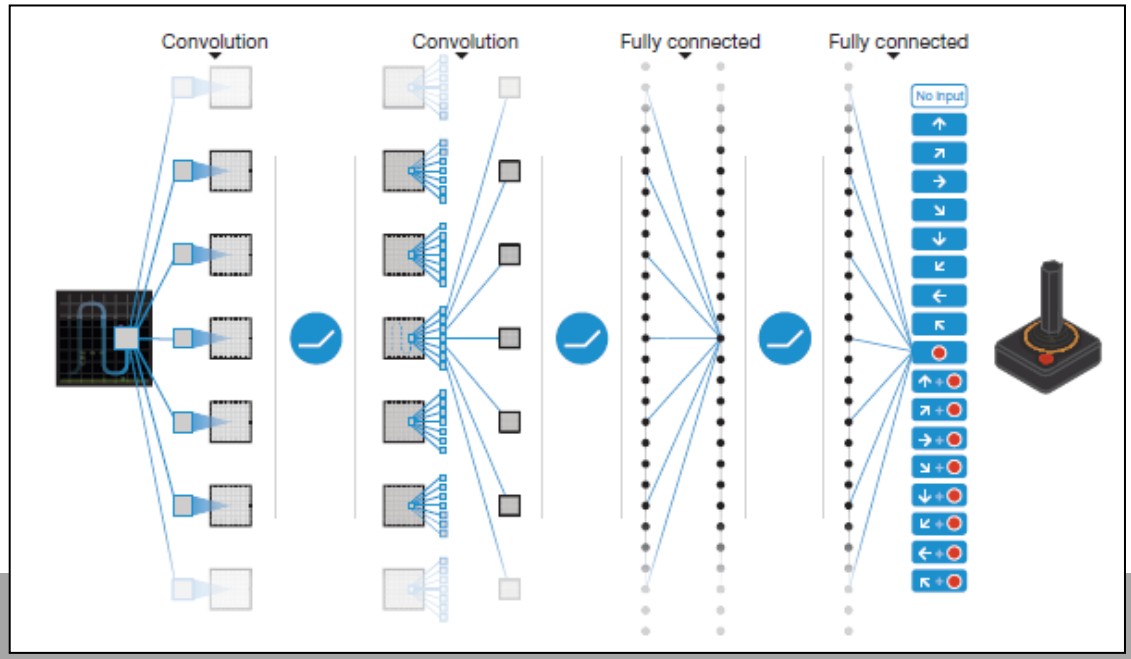


https://en.wikipedia.org/wiki/Recurrent_neural_network

DQN (Deep Q-Network)

DQN 은 알파고와 같은 복잡한 강화학습에 사용한다

- DQN = 강화학습 + 딥러닝
- 강화학습은 시뮬레이션을 통해 샘플을 자동으로 생성한다
  - 이에 반해, 기존의 딥러닝은 막대한 양의 샘플과 타겟값을 사전에 확보하여야 한다
  - (상태1,행동1) → (상태2,행동2) → … → (상태N,행동N) → 승리/패배
- 강화학습은 바둑, 비디오게임 과 같이 **<u>규칙/환경/보상</u>** 등이 사전에 규정되어야 한다
  - 모든 상태(state)와, 각 상태에서 가능한 모든 행동(action)이 사전에 정의되어야 함
  - 과업을 끝마쳤을 때 환경으로 부터 받을 명확한 보상값이 정의되어야 함



https://www.nature.com/articles/nature14236