개인화 추천 시스템 II 실습자료

서울대학교 통계학과[1]

6-5. Convolutional Neural Network

이미지 학습을 위한 NN

- 동물의 시각 인지 과정을 모방한 모형으로 이미지 분류 문제에 특히 높은 성능을 나타냄.
- Convolutional layer, pooling layer, convolutional layer, pooling layer, $\cdots$, fully-connected layer, output layer의 순서로 네트워크가 이루어짐.
- 일반적으로 역전파 알고리즘을 이용하여 모수 추정.

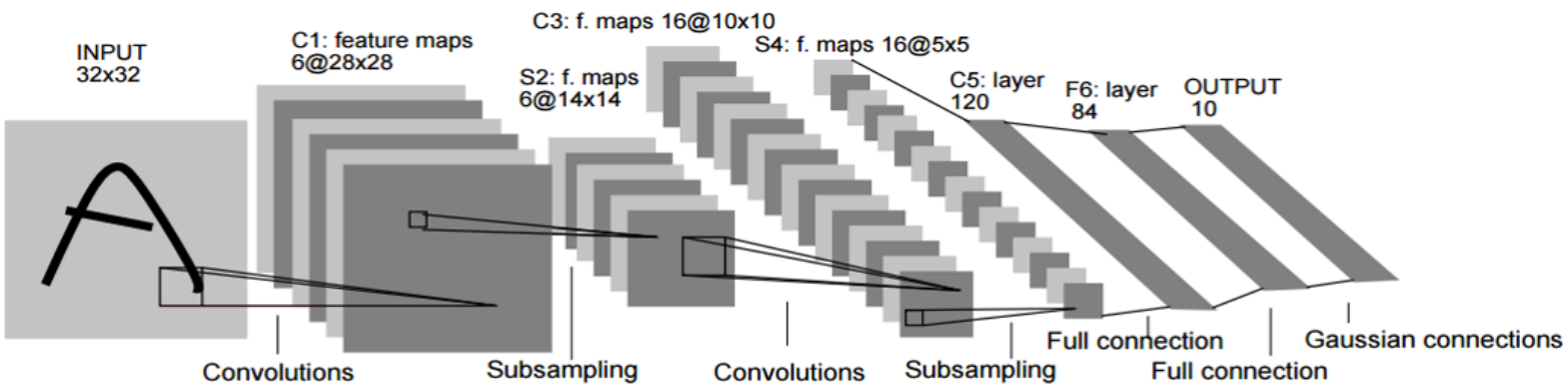
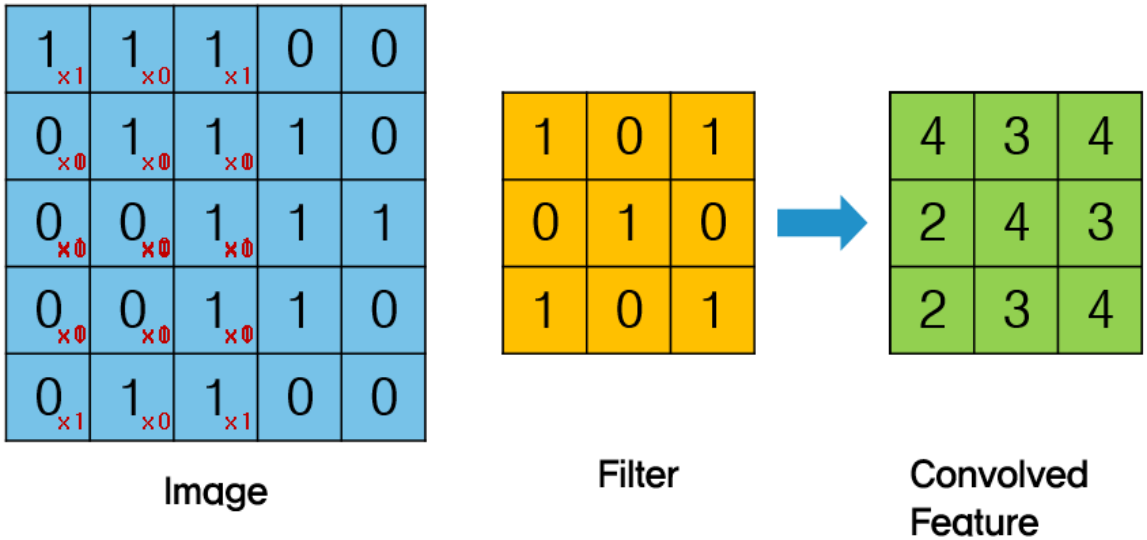


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

6-5. Convolutional Neural Network

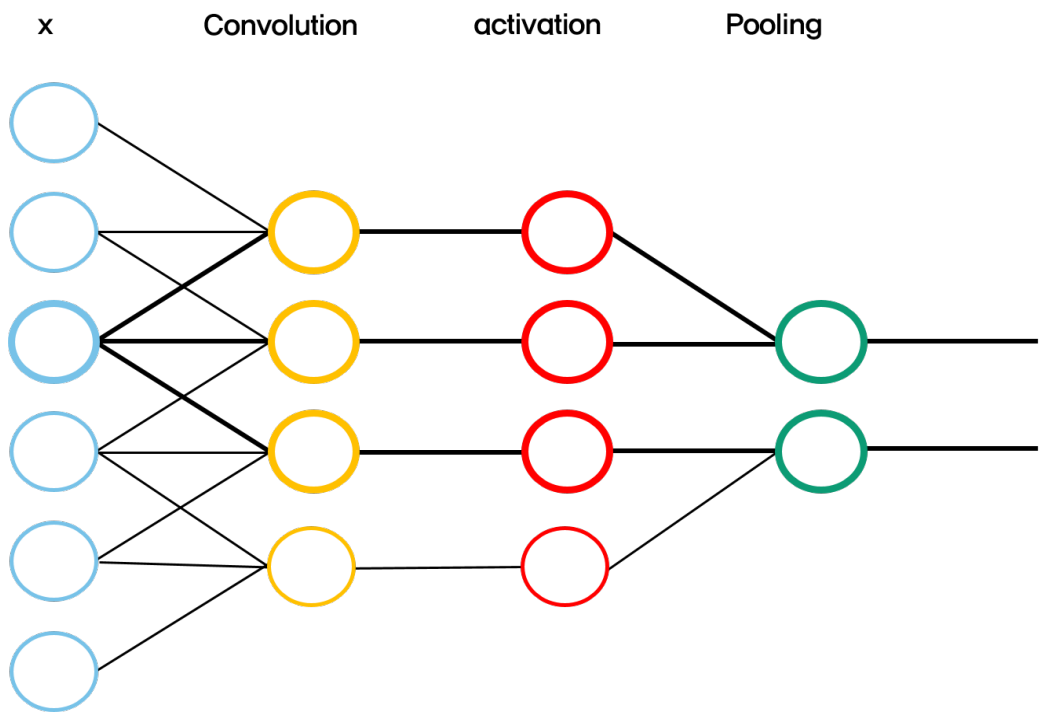
- Convolution



✓ 컬러 이미지인 경우에는 Red, Green, Blue 명도 각각에 대해서 filter를 걸어준다. 즉, filter가 3차원 cubic이 된다.

6-5. Convolutional Neural Network

- Convolution: dimension reduction



6-5. Convolutional Neural Network

- Max pooling

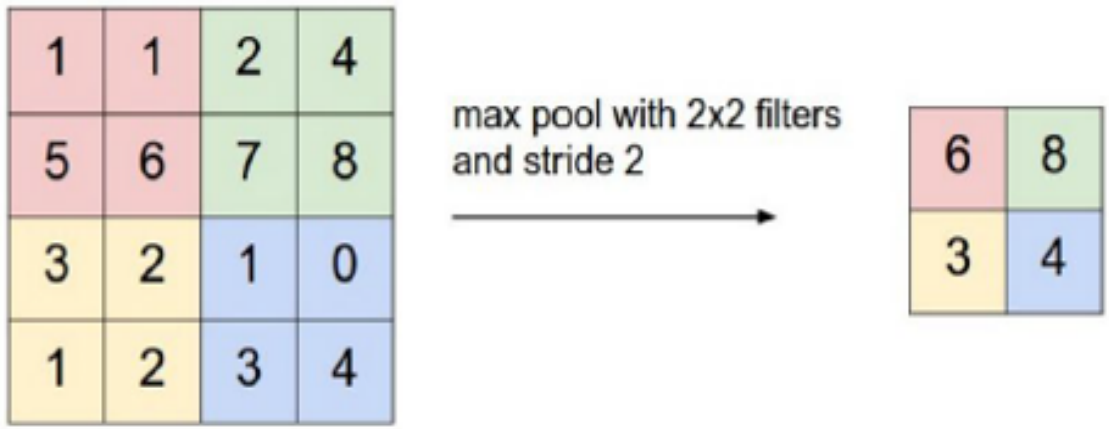


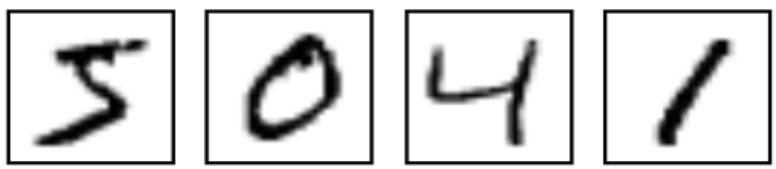
Figure: The max-pooling operation

6-5. Convolutional Neural Network의 실습

MNIST dataset

- 0-9까지 총 10가지의 수 중 하나를 손으로 직접 쓴 이미지 데이터
- 60,000개의 훈련 자료와 10,000개의 시험 자료가 있음
- 이미지가 주어지면 그 이미지에 쓰여진 숫자를 맞추는 분류 문제를 CNN을 이용하여 만들어보자.

- 모델 구현 프로그램
  - Python 내부에 있는 Keras 라이브러리 사용
  - Keras는 Python 딥러닝 프레임워크의 양대 산맥인 Tensorflow와 Theano를 감싸서 더 간결하고 쉬운 API를 제공하는 신경망 모형 라이브러리.



MNIST 자료의 예

6-5. Convolutional Neural Network의 실습

- 코드

In [1]:

```
from __future__ import print_function
import numpy as np
np.random.seed(1337)  # for reproducibility
import json

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras import backend as K

batch_size = 128
nb_classes = 10
nb_epoch = 20

# input image dimensions
img_rows, img_cols = 28, 28
# number of convolutional filters to use
nb_filters = 3
# convolution kernel size
nb_conv = 5
# size of pooling area for max pooling
nb_pool = 2

# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Using TensorFlow backend.
```

- MNIST 데이터 불러오기
- CNN에 필요한 operator 불러오기
- 모형 학습에 필요한 모수 설정
- 모형 학습에 필요한 모수 설정
- MNIST 자료 불러오기

6-5. Convolutional Neural Network의 실습

- 코드

In [2]:

```
X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)
```

- 이미지 데이터의 형태 변환
  (28,28) -> (28,28,1)

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print('X_train shape:', X_train.shape)
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

- 이미지 데이터의 형 변환
- 각 pixcel의 크기(0~255)를 0과 1 사이로 바꿈.
- 훈련 자료와 시험 자료의 형태 확인

```
# convert class vectors to binary class matrices
Y_train = to_categorical(y_train, nb_classes)
Y_test = to_categorical(y_test, nb_classes)
```

- 종속 변수를 one-hot 벡터로 변형
- One-hot 벡터란? :
  특정 카테고리에 해당하는 원소만
  1이고 나머지 원소들은 0인 벡터
- 예)
  0 -> (**1**,0,0,0,0,0,0,0,0,0)
  5 -> (0,0,0,0,0,**1**,0,0,0,0)
  3 -> (0,0,0,**1**,0,0,0,0,0,0)

```
X_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

6-5. Convolutional Neural Network의 실습

- 코드

In [3]:

```
model = Sequential()
```

- 모델 구축 시작

```
model.add(Conv2D(nb_filters, (nb_conv, nb_conv),
                 input_shape=input_shape))
model.add(Activation('relu'))
model.add(Conv2D(nb_filters, (nb_conv, nb_conv)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))
```

- Convolution layer -> Convolution layer -> max-pooling layer 순으로 아키텍처 생성

```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
```

- Fully-connected layer 생성
- 마지막 layer는 softmax

```
model.compile(loss='categorical_crossentropy',
              optimizer='adadelta', metrics=['accuracy'])
```

- 손실 함수로 cross-entropy 사용
- 최적화 방법으로는 역전파 알고리즘의 변형인 adadelta 이용
- 모형 적합의 성능 척도로 정분류율 사용

6-5. Convolutional Neural Network의 실습

- 결과

```
In [4]: model.summary()
```

- 모델 아키텍처 설명 요청

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 3)         78
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 3)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 3)         228
_________________________________________________________________
activation_2 (Activation)    (None, 20, 20, 3)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 3)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 3)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               38528
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_4 (Activation)    (None, 10)                0
=================================================================
Total params: 40,124
Trainable params: 40,124
Non-trainable params: 0
_________________________________________________________________
```

6-5. Convolutional Neural Network의 실습

- 결과

```
In [5]: hist = model.fit(X_train, Y_train, batch_size=batch_size,
                         epochs=nb_epoch, verbose=1, validation_split=0.2)
        score = model.evaluate(X_test, Y_test, verbose=0)
        print('Test score:', score[0])
        print('Test accuracy:', score[1])
```

- 모수 추정 시작
- 적합이 끝난 후에 시험 자료를 통해 모형의 성능 확인
  (score : 손실 함수 값과 정분류율)

```
Epoch 16/20
48000/48000 [==============================] - 13s - loss: 0.1512 - acc: 0.9550 - val_
loss: 0.0741 - val_acc: 0.9772
Epoch 17/20
48000/48000 [==============================] - 13s - loss: 0.1475 - acc: 0.9565 - val_
loss: 0.0731 - val_acc: 0.9780
Epoch 18/20
48000/48000 [==============================] - 13s - loss: 0.1395 - acc: 0.9577 - val_
loss: 0.0711 - val_acc: 0.9783
Epoch 19/20
48000/48000 [==============================] - 13s - loss: 0.1399 - acc: 0.9574 - val_
loss: 0.0703 - val_acc: 0.9787
Epoch 20/20
48000/48000 [==============================] - 13s - loss: 0.1357 - acc: 0.9592 - val_
loss: 0.0684 - val_acc: 0.9790
Test score: 0.0638477349755
Test accuracy: 0.9787
```

- 모수 추정 과정이 표시됨
- 최종 모형 결과

6-5. Convolutional Neural Network의 실습

- 결과

In [6]:
```
import os
os.chdir('D:₩₩by0705₩₩조교자료₩₩data_code₩₩MNIST')
json_string = model.to_json()
open('mnist_model_architecture.json','w').write(json_string)
model.save_weights('mnist_model_weights.h5')

# Save History
with open('mnist_model_history.json','w') as fp:
    json.dump(hist.history, fp)
```

- 모형의 결과 저장

In [7]:
```
# Plot history
hist = json.loads(open('mnist_model_history.json').read())
```

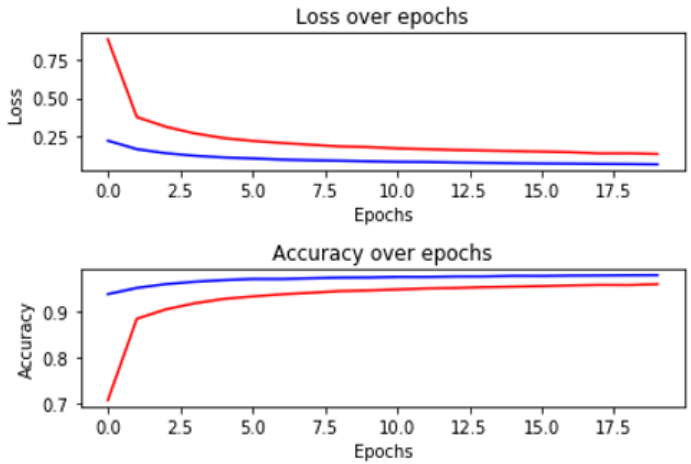
- 모형 불러오기

```
plt.figure('history')
plt.subplot(211)
plt.title('Loss over epochs')
plt.plot(hist['loss'],'r',label='loss')
plt.plot(hist['val_loss'], 'b',label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.subplot(212)
plt.title('Accuracy over epochs')
plt.plot(hist['acc'],'r',label='acc')
plt.plot(hist['val_acc'], 'b',label='val_acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.tight_layout()
plt.show()
```

- 모수를 추정해나감에 따라 얻어지는 손실 함수의 값과 정분류율을 그래프로 표시 (훈련 자료와 검증 자료 둘 다 표시)

6-5. Convolutional Neural Network의 실습

- 손실함수(위)와 정분류율(아래) 결과



- 파란색 : 훈련자료
- 빨간색 : 검증자료

6-5. Convolutional Neural Network의 실습

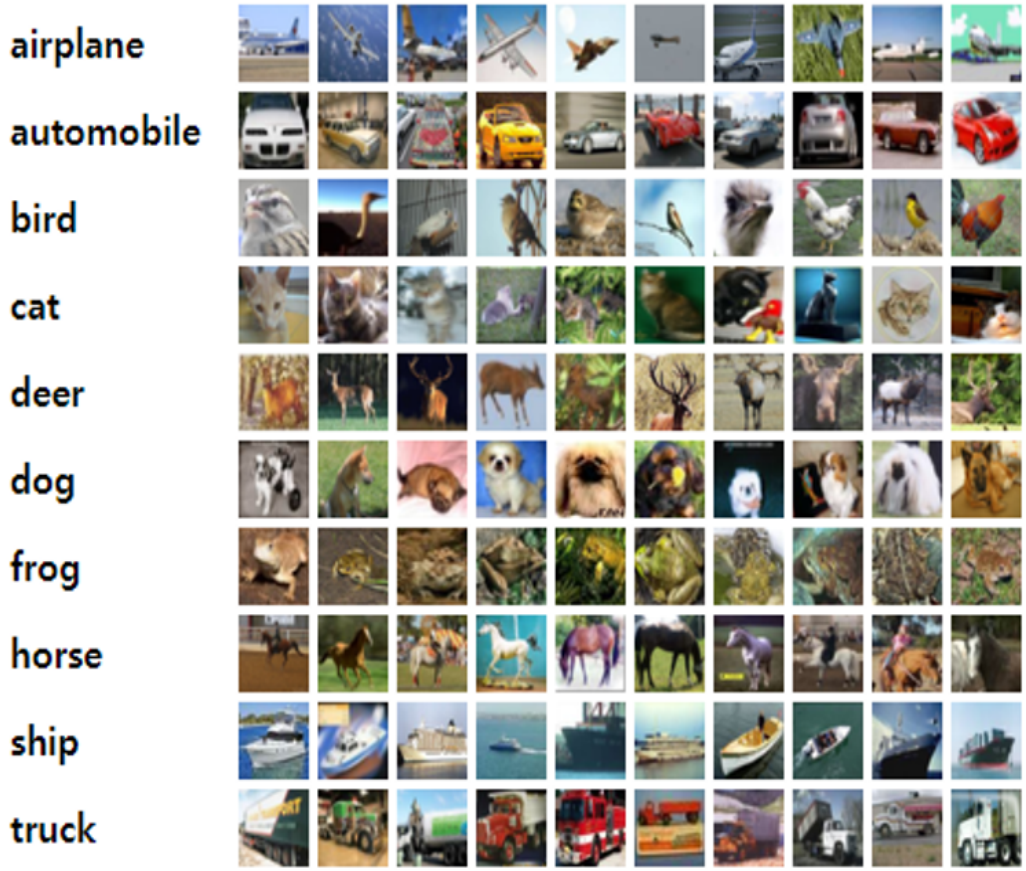
직접해보기 (CIFAR-10 dataset)

- 10개 class의 이미지 데이터. 각 class마다 32×32의 6,000개 RGB 이미지 존재.
- 50,000개의 훈련 자료와 10,000개의 시험 자료가 있음
- 이미지의 class를 맞추는 분류 문제를 CNN을 이용하여 만들어보자.

힌트 :

- MNIST 자료와 마찬가지로 keras 모듈에서 자료를 불러올 수 있다.
- CIFAR 10은 RGB image 이기 때문에 input_shape의 세번째 원소가 3이어야 한다.

6-5. Convolutional Neural Network의 실습

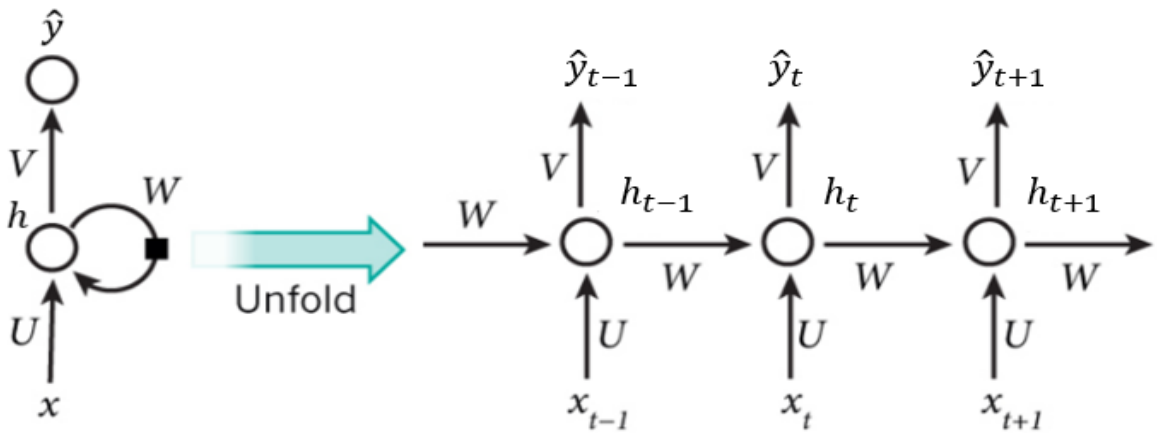


CIFAR-10 자료의 예

6-6. Recurrent Neural Network

RNN이란?

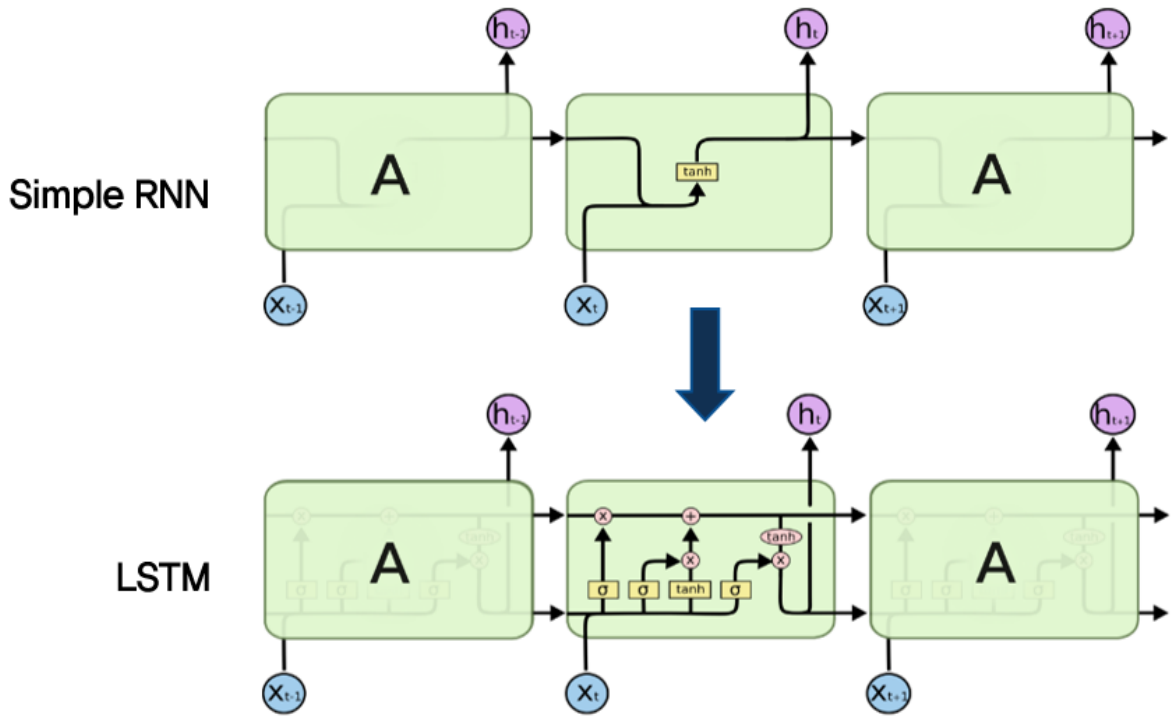
- 순차적으로 들어오는 정보를 처리하는 Neural Network.
- 예를 들어, 문장에서 다음에 나올 단어를 추측하고 싶은 경우 이전에 나온 단어의 정보가 필요.
- 동일한 task를 입력값 시퀀스 (x_t)의 모든 요소마다 적용하고, 출력결과 ($\hat{y}_t$)에서는 이전 시간의 계산결과 (h_{t-1})에 영향을 받는다.



6-6. Recurrent Neural Network: LSTM

Long Short Term Memory (Hochreiter, S., & Schmidhuber, J. ,1997)

- Vanishing gradient 문제를 해결하기 위한 대표적인 방법 중 하나.
- Memory cell이라는 새로운 노드를 추가하여 과거의 정보가 현재에도 영향을 잘 미칠 수 있도록 함.



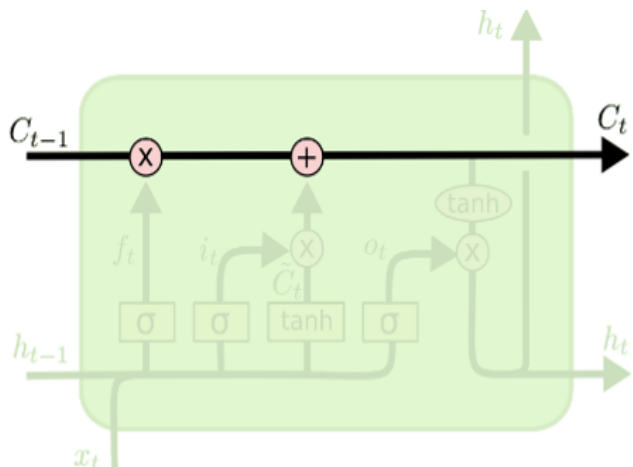
6-6. Recurrent Neural Network: LSTM

LSTM 의 핵심 아이디어

- Memory cell : t시점에서의 memory cell 을 C_t 라고 표기. 이 때 C_t 은 기존 RNN의 h_t와 달리 self-recurrent connection으로 업데이트가 되어 과거의 정보를 잘 보존할 수 있음.

$$C_t = C_{t-1} + function(x_t, h_{t-1})$$

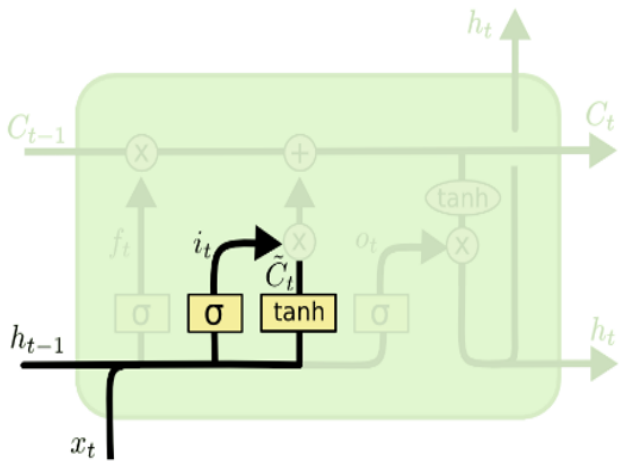
- 게이트 : 특정 노드의 정보를 보존하거나 지우도록 해주는 역할을 함. 0과 1 사이의 값을 가지며 1에 가까울수록 더 많은 정보를 보존. 출력값이 0인 경우 해당 게이트를 지나는 모든 정보가 지워짐.
- LSTM은 셀 상태의 정보를 제어하기 위한 세 종류의 게이트가 있다.



6-6. Recurrent Neural Network: LSTM

LSTM

- Input 게이트
  - 어떤 새로운 정보를 셀 상태에 저장할지 결정.
  - $\tilde{C}_t$은 셀 상태에 더해질 수 있는 새로운 후보값들의 벡터를 생성하고, i_t은 생성된 후보값들을 얼마나 저장할 지 결정.

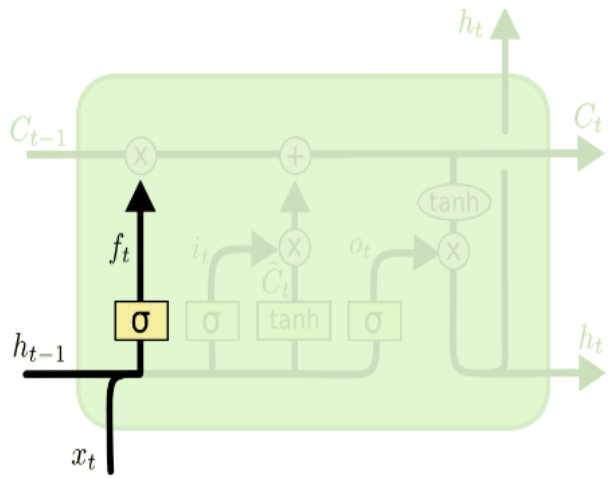


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

6-6. Recurrent Neural Network: LSTM

LSTM

- Forget 게이트 (Gers & Schmidhuber, 2000)
  - 이전 시점 $t-1$에서 어떠한 정보를 버릴지 결정.
  - 제일 처음에 나왔던 LSTM의 방법에서는 forget 게이트가 없이 기존에 정보를 계속 유지하도록 설정했지만 불필요한 정보도 계속 유지하고 있기 때문에 좋지 않은 성능을 나타낼 수 있음.
  - Forget 게이트를 추가한 LSTM 모형을 보편적으로 사용.

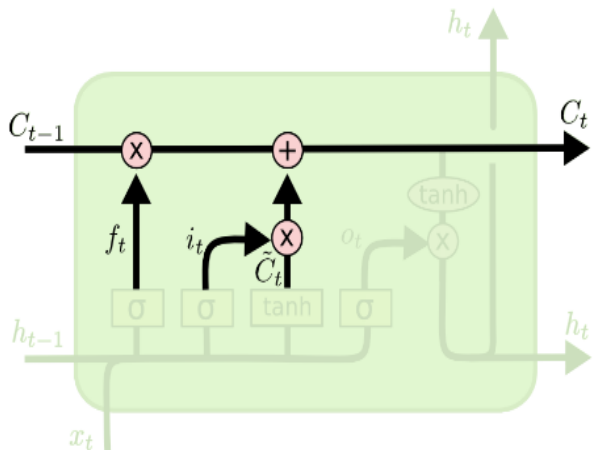


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

6-6. Recurrent Neural Network: LSTM

LSTM

- 셀 상태 업데이트
  - 앞의 Input 게이트에서 계산한 값과 이전 시점의 셀 상태의 값을 더하여 계산한다. 여기서 이전 시점의 셀 상태에 f_t을 곱해줌으로써 이전 정보를 얼마나 유지할지 결정한다.
  - 일반적으로 $C_0 = 0$으로 설정한다.

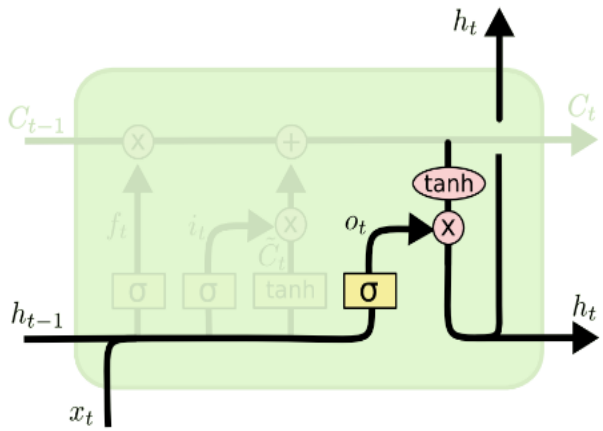


$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

6-6. Recurrent Neural Network: LSTM

LSTM

- Output 게이트
  - o_t의 값으로 셀 상태에서 어떤 부분을 출력할지 결정.
  - 업데이트한 셀 상태의 값에 *tanh*을 취한 뒤, 결정된 부분만 출력하도록 o_t를 곱함.



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \cdot tanh(C_t)$$

6-6. Recurrent Neural Network의 실습

IMDB dataset

- 영화에 대한 text review 와 0 또는 1 (Good / Bad)의 labeling 데이터
- 25,000개의 훈련 자료와 25,000개의 시험 자료가 있음
- Text review는 단어 index로 코딩되어 있음
  - 정수 "3"에 해당하는 단어는 3번째로 많이 등장하는 단어
- 영화 review의 단어들을 이용하여 주어진 label을 맞추는 분류 문제를 LSTM을 이용하여 만들어보자.

6-6. Recurrent Neural Network의 실습

- 코드

In [1]:
```
import numpy as np
np.random.seed(10) #for reproducibility

# LSTM for sequence classification in the IMDB dataset
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence

batch_size=64
nb_epoch=3
nb_classes=1

# load the dataset but only keep the top n words
n_words = 5000
# number of maximum words length in review
max_length = 500
# embedding vector length
embedding_length = 32
```

```
Using TensorFlow backend.
```

- IMDB 데이터 불러오기
- LSTM에 필요한 operator 불러오기
- 모형 학습에 필요한 모수 설정
- 모형 학습에 필요한 모수 설정

6-6. Recurrent Neural Network의 실습

- 코드

In [2]:
```
#the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=n_words)
print('X_train shape: ',X_train.shape)
print(len(X_train[0]))
print(X_train[0][:10])
print(X_train[0][-10:])
```

```
X_train shape:  (25000,)
218
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65]
[4472, 113, 103, 32, 15, 16, 2, 19, 178, 32]
```

- IMDB 자료 불러오기
- 빈도수 많은 5000개의 단어만 불러온다.
- 훈련자료 형태확인: 첫번째 영화 review는 152개의 단어로 구성, 처음 단어 10개와 마지막 단어 10개를 확인할 수 있다.

In [3]:
```
# truncate and pad input sequences
X_train = sequence.pad_sequences(X_train, maxlen=max_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_length)
print('X_train shape: ',X_train.shape)
print(len(X_train[0]))
print(X_train[0][:10])
print(X_train[0][-10:])
```

```
X_train shape:  (25000, 500)
500
[0 0 0 0 0 0 0 0 0 0]
[4472  113  103   32   15   16    2   19  178   32]
```

- Padding
- RNN 구조에서 시간의 길이가 다를 때 사용
- 500단어 이상이면 중간에 잘라주고, 500단어 미만이면 앞에 0 으로 구성된 열을 붙어준다.
- Padding 후 형태 확인

6-6. Recurrent Neural Network의 실습

- 코드

In [4]:
```
# create the model
model = Sequential()

model.add(Embedding(n_words, embedding_length,
                    input_length=max_length))
model.add(Dropout(0.1))
model.add(LSTM(100))
model.add(Dropout(0.2))

model.add(Dense(nb_classes))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

- 모델 구축 시작

- Embedding -> LSTM
- Embedding이란?
  딥러닝 구조들은 문자열이나 일반 텍스트를 처리하지 못하기 때문에 하나의 단어를 실수의 저차원 벡터로 요약.

- Fully-connected layer 생성
- 마지막 layer는 binary-classification이기 때문에 sigmoid

- 손실 함수로 binary_crossentropy 사용
- 최적화 방법으로 adam 이용
- 모형 적합의 성능 척도로 정분류율 사용

6-6. Recurrent Neural Network의 실습

- 결과

```
In [5]: model.summary()
```

- 모델 아키텍처 설명 요청

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           160000
_________________________________________________________________
dropout_1 (Dropout)          (None, 500, 32)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
```

6-6. Recurrent Neural Network의 실습

- 결과

In [6]:
```
model.fit(X_train, y_train, epochs=nb_epoch, batch_size=batch_size,
          validation_split=0.2)

scores = model.evaluate(X_test, y_test, verbose=0)
print("Test score: ", scores[0])
print("Test Accuracy: ", scores[1])
```

- 모수 추정 시작
- 적합이 끝난 후에 시험 자료를 통해 모형의 성능 확인
  (score : 손실 함수 값과 정분류율)

```
Train on 20000 samples, validate on 5000 samples
Epoch 1/3
20000/20000 [==============================] - 206s - loss: 0.5127 - acc: 0.7344 - val
_loss: 0.3674 - val_acc: 0.8414
Epoch 2/3
20000/20000 [==============================] - 205s - loss: 0.3170 - acc: 0.8713 - val
_loss: 0.3266 - val_acc: 0.8620
Epoch 3/3
20000/20000 [==============================] - 206s - loss: 0.2632 - acc: 0.8986 - val
_loss: 0.3184 - val_acc: 0.8694
```

- 모수 추정 과정이 표시됨

```
Test score:  0.330381559658
Test Accuracy:  0.86444
```

- 최종 모형 결과

6-7. Deep learning 과 추천 알고리즘

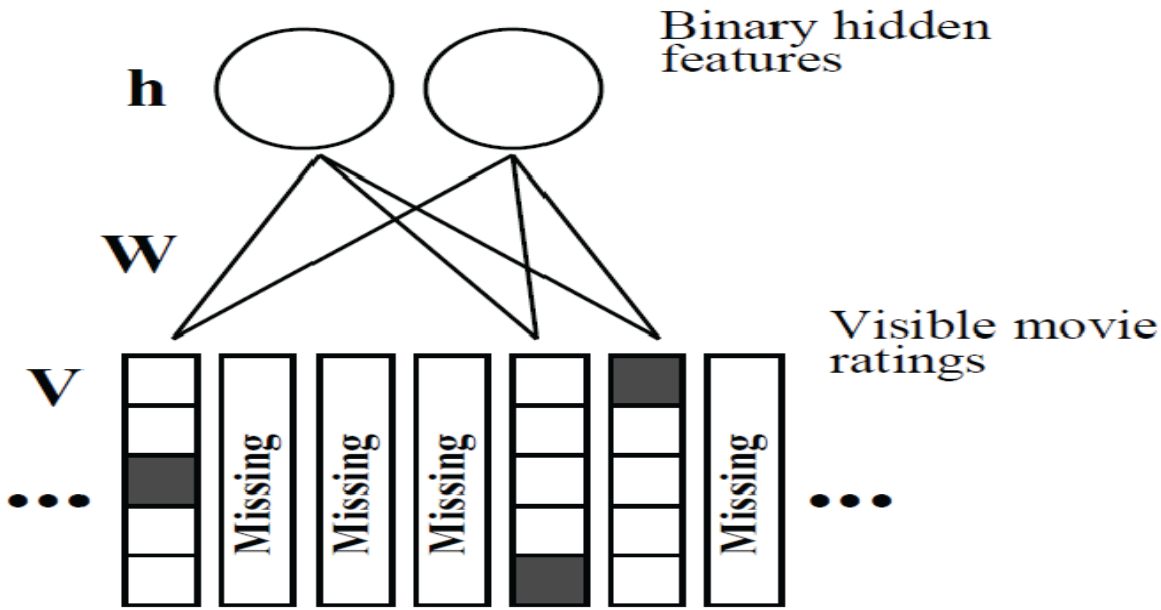
Restricted boltzmann machines for collaborative filtering
(R. Salakhutdinov et al., 2007)

- RBM 모형 사용하여 선호도 예측
- 고객마다 missing 인 rating이 다르기 때문데, 각 고객마다 다른 RBM 모형 사용
- 하지만 RBM들의 모수들은 서로 공유

- 가정 : 한 고객이 m개의 상품에 rating을 했고, $1, \ldots, K$의 정수로 rating을 한다고 가정하자.
- 상품 i 를 k라고 rating했으면 v_i^k=1, 아니면 0.
- $\mathbf{V}$: $K \times m$의 binary maxrix가 입력값이다.

6-7. Deep learning 과 추천 알고리즘

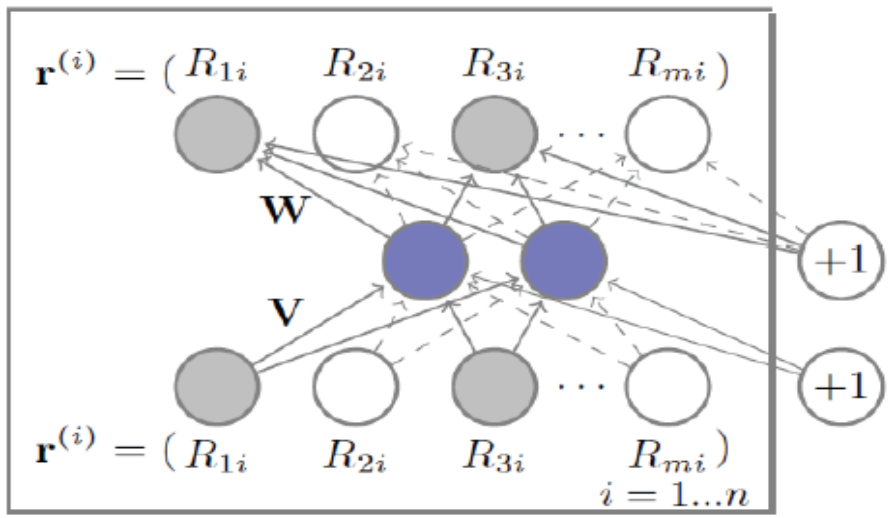
- 한명의 고객에 대한 RBM 모형



6-7. Deep learning 과 추천 알고리즘

AutoRec : Autoencoders Meet Collaborative Filtering
(S. Sedhain et al., 2015)

- 앞의 RBM 모형과 비슷하나, Auto-Encoder 모형 이용
- 입력값과 출력값이 실수이다.



6-8. 협력적 정화 방법과 Deep learning 방법

자료

자료	고객수	상품수	점수수	점수 스케일
Jester5k	1,000	100	72,358	-10 -10 실수
MovieLense	943	1,664	99,392	1-5 정수

- 자료의 30% 를 랜덤하게 추출하여 검증자료로 함

실험 결과(test RMSE) 비교

Method	Jester5k	MovieLense
Neighborhood(고객중심)	4.4235	1.0143
Neighborhood(고객중심, 스케일보정)	4.4410	0.9477
행렬분해	4.3245	**0.7808**
행렬분해(스케일 보정)	4.3455	1.0200
RBM-CF(user-based)	NA	0.9707
RBM-CF(item-based)	NA	0.9466
U-AutoRec	4.3187	1.0858
I-AutoRec	**4.1912**	1.0415