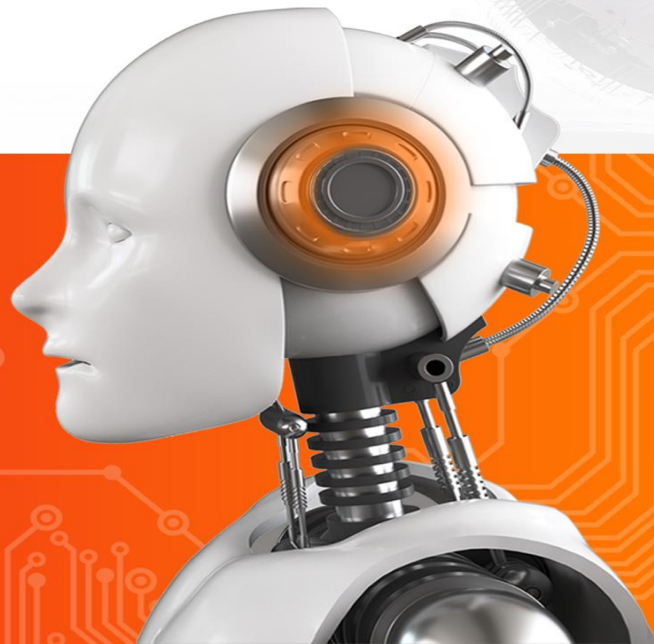




1. NOSQL의 특징





1.1 NOSQL의 특징



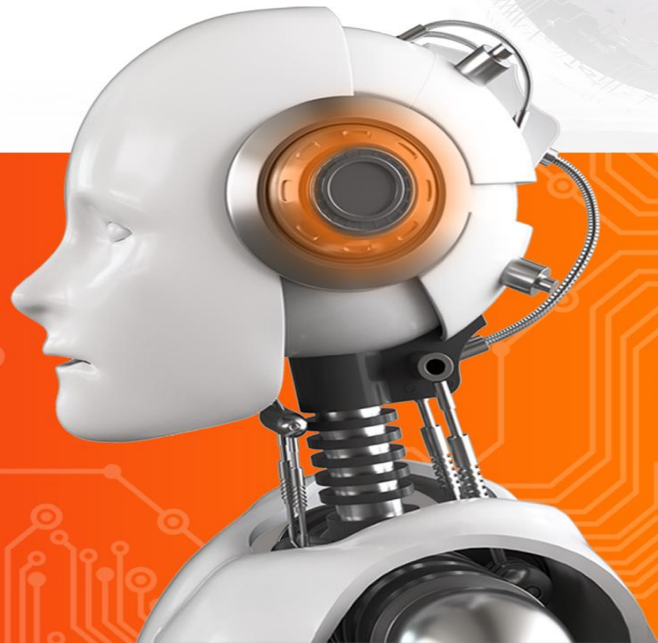
1. NOSQL 특징

● NOSQL 등장 배경

- 1) 대용량의 데이터를 저장 할 수 있는 단순한 형태의 데이터베이스
- 2) 주로 비정형 데이터(이미지, 동영상 등) 을 저장



1.2 NoSQL 종류



2. NOSQL 종류

● Key/Value Store

- 대부분의 NoSQL은 Key/Value 개념을 지원
- Unique Key에 하나의 Value를 가지고 있는 형태
- `put(key,value)`, `value := get(key)` 형태 API 사용

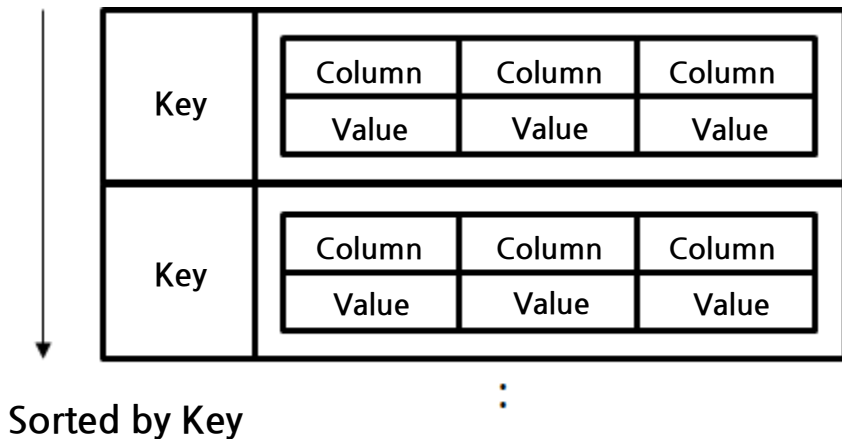
Key	Value
Key	Value

:

2. NOSQL 종류

Ordered Key/Value Store

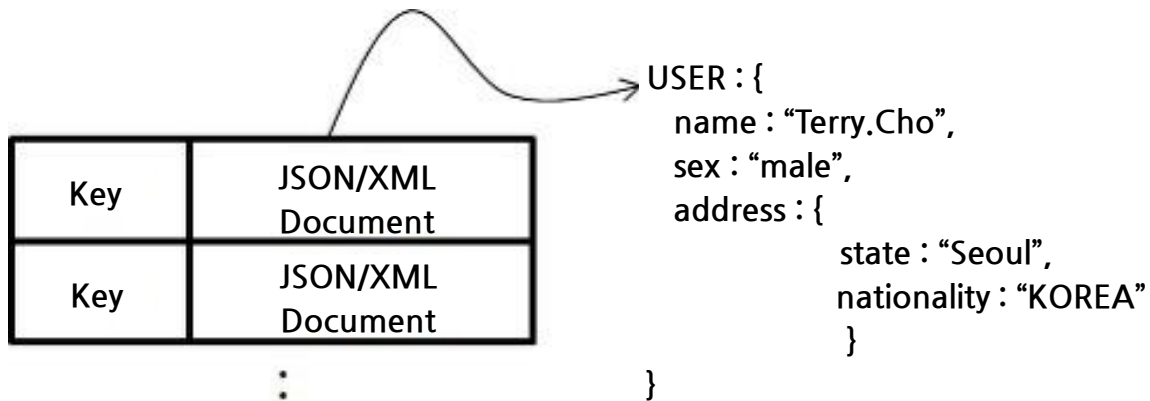
- 데이터가 내부적으로 Key를 순서로 Sorting되어 저장됨
- Key 안에 (column:value) 조합으로 된 여러개의 필드를 가지는 구조
- 대표 제품 : Hbase, Cassandra



2. NOSQL 종류

Document Key/Value Store

- Key/Value Store의 확장된 형태
- 저장되는 Value의 데이터 타입으로 “Document”라는 구조화된 데이터 타입(JSON,XML,YAML등)을 사용
- 복잡한 계층구조 표현 가능
- 제품에 따라 추가 기능(Sorting,Join,grouping) 지원



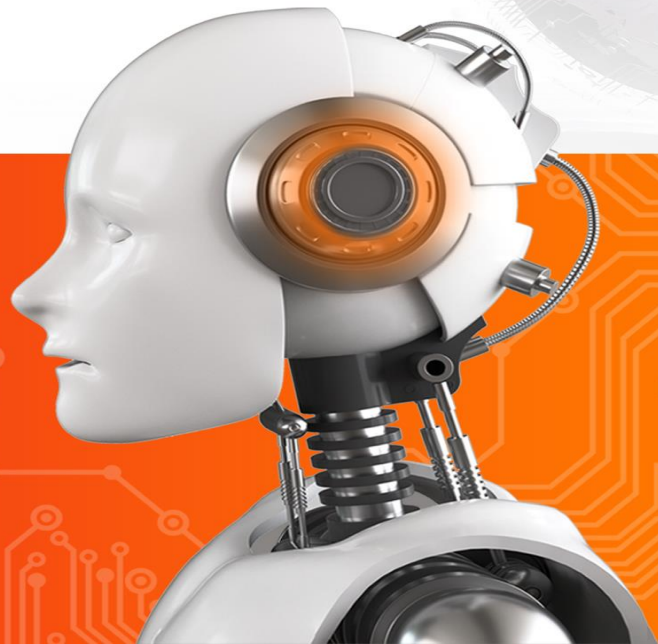
2. NOSQL $\frac{7}{0}$ $\frac{2}{11}$

● NoSQL System List

- Key-Value Stores: Oracle Coherence, Redis, Kyoto Cabinet
- BigTable-style Databases: Apache HBase, Apache Cassandra
- Document Databases: MongoDB, CouchDB
- Full Text Search Engines: Apache Lucene, Apache Solr
- Graph Databases: neo4j, FlockDB



2. MongoDB 기본개념





2.1 MongoDB 특징



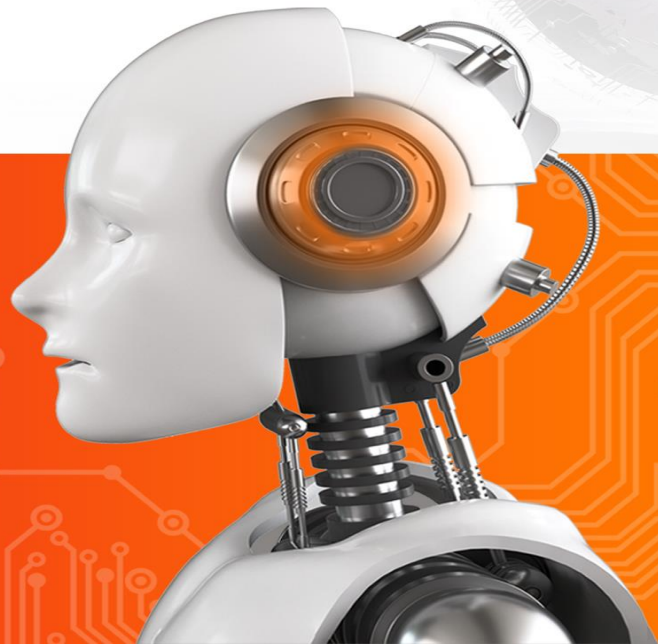
1. MongoDB 특징

● MongoDB 소개

- 10gen 사에서 개발한 솔루션 (C++)
- Key-value와 다르게 여러 용도로 사용이 가능 (범용적)
- 스키마를 고정하지 않는 형태
 - ✓스키마 변경으로 오는 문제 없음
 - ✓데이터를 구조화해서 json 형태로 저장 (데이터를 key-value화 저장)
- Join이 불가능하기 때문에 join이 필요없도록 데이터 설계 필요

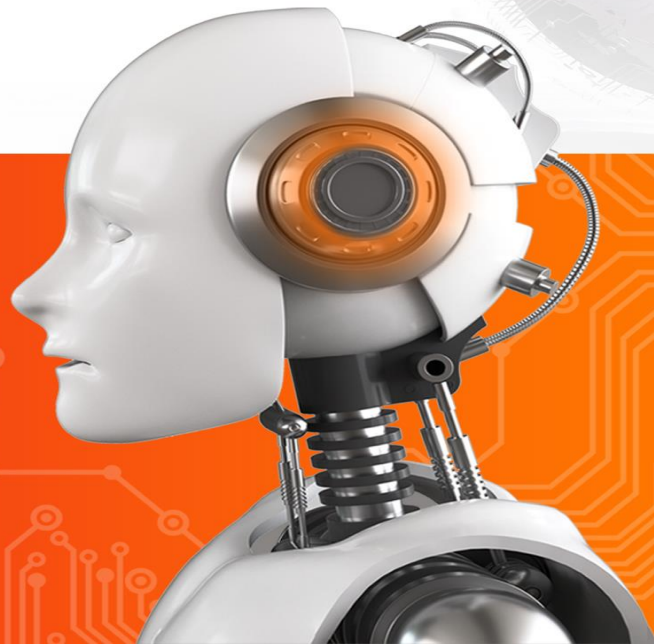


3. MongoDB 동작 방식





3.1 MongoDB 기본 동작 방식



1. MongoDB 기본 동작 원리

● MongoDB 설치

- MongoDB는 총 4개의 운영체제를 지원
- MongoDB를 사용하는 대부분의 유저는 90%가 리눅스 사용
 - ✓ 빅데이터 처리를 위해서는 많은 수의 머신이 필수적으로 필요
 - ✓ 운영체제에 대한 비용이 상대적으로 적은 리눅스를 사용
- MongoDB의 철학 : “메모리 관리는 운영체제에 맡기자”
 - ✓ 상대적으로 메모리 관리에 뛰어난 *nix 베이스의 운영체제를 선택
- 윈도우용 MongoDB는 설치 및 사용이 간편함

1. MongoDB 기본 동작 원리

● MongoDB 설치

▪ 지원 언어 :

- ✓ Java / Java Script
- ✓ Perl / PHP / Python
- ✓ Ruby / Scala / Erlang / Haskell

▪ 지원 운영체제

Windows	Windows 32Bit	Windows 64Bit
Linux	Linux 32Bit	Linux 64Bit
Unix Solaris	Unix Solaris 32Bit	Unix Solaris 64Bit
Mac OS X	Mac OS X 32Bit	Mac OS X 64Bit

1. MongoDB 기본 동작 원리

● MongoDB 설치

- MongoDB 공식 사이트 (<https://www.mongodb.com/download-center/community>) 접속
- 다운로드 페이지에서 운영체제에 맞는 배포판 선택 및 다운로드

MongoDB Enterprise Server

MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. The community version offers the flexible document model along with ad hoc queries, indexing, and real time aggregation to provide powerful ways to access and analyze your data. As a distributed system you get high availability through built-in replication and failover along with horizontal scalability with native sharding.

The MongoDB Enterprise Server gives you all of this and more. Review the Enterprise Server tab to learn what else is available.

Available Downloads

Version

4.2.13

Platform

Windows

Package

zip

Download

Copy Link

Current releases & packages

Development releases

Archived releases

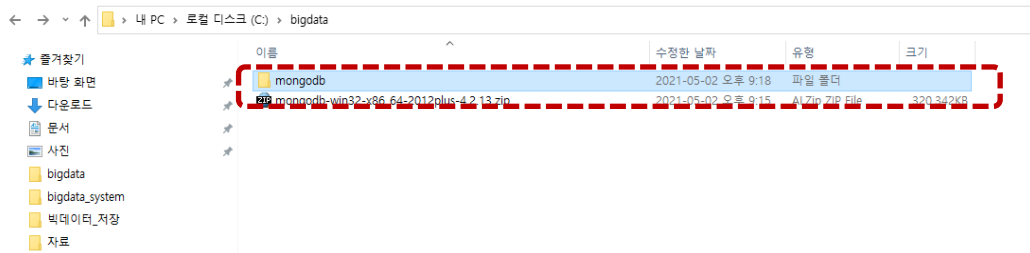
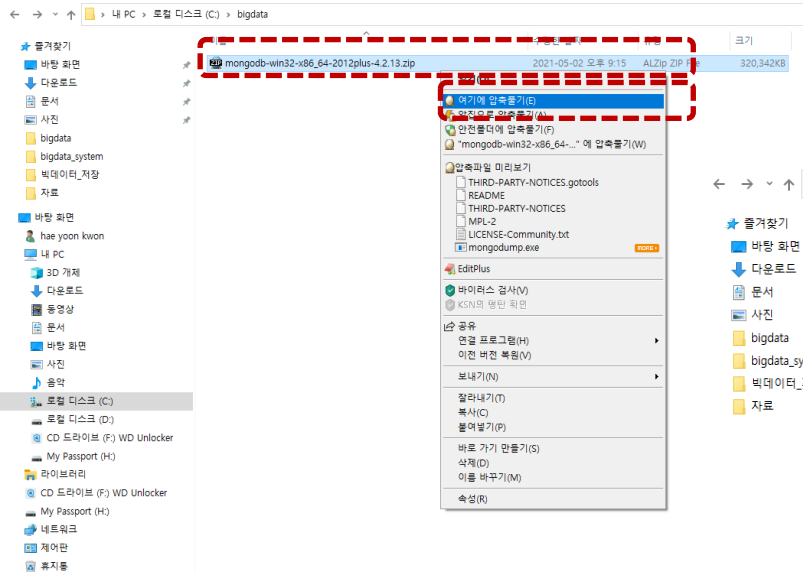
Changelog

Release Notes

1. MongoDB 기본 동작 원리

MongoDB 설치

- 다운로드받은 압축 파일을 c:\bigdata 폴더에 압축해제
- 폴더명을 mogodb로 수정





3.2 MongoDB 실행



2. MongoDB 실행

● MongoDB 실행

- 명령프롬프트 새창을 실행
- MongoDB 설치 폴더로 이동

✦ cd c:\bigdata\mongodb\bin

C:\Users\yujin>cd c:\bigdata\mongodb\bin

c:\bigdata\mongodb\bin>

2. MongoDB 실행

• MongoDB 실행

- MongoDB 데이터 저장 폴더 생성

✦ mkdir c:\bigdata\mongodb\var

```
c:\bigdata\mongodb\bin>mkdir c:\bigdata\mongodb\var
```

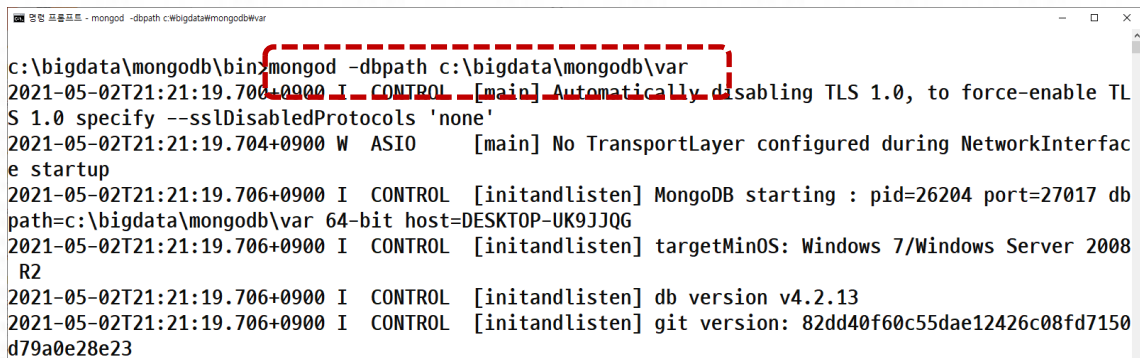
```
c:\bigdata\mongodb\bin>
```

2. MongoDB 실행

● MongoDB 실행

■ MongoDB 실행

✦ `mongod -dbpath c:\bigdata\mongodb\var`



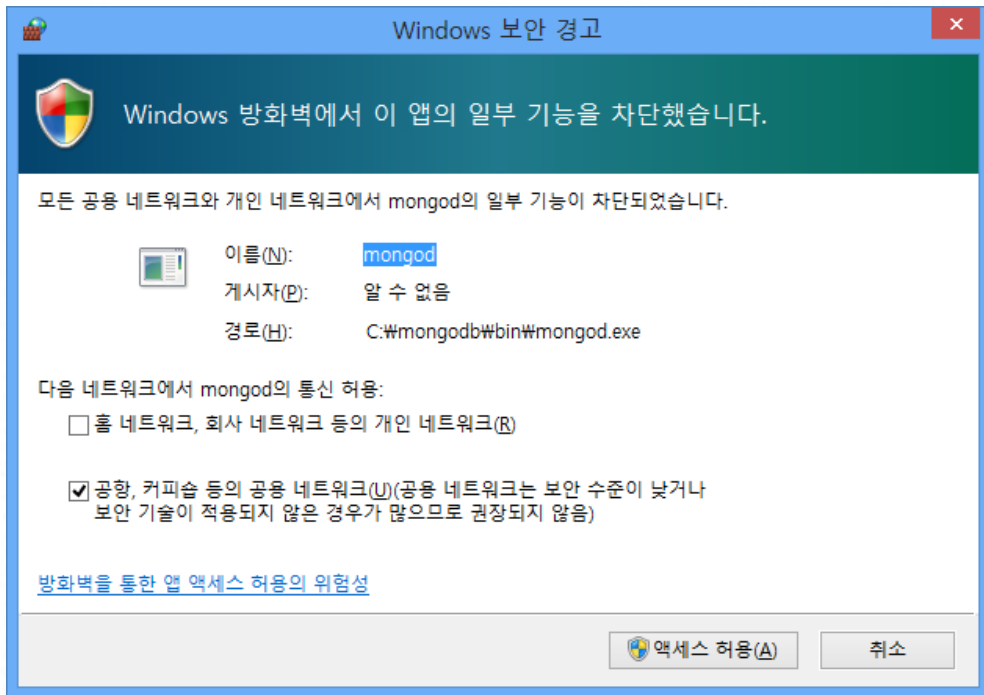
```
c:\bigdata\mongodb\bin>mongod -dbpath c:\bigdata\mongodb\var
2021-05-02T21:21:19.706+0900 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS
1.0 specify --sslDisabledProtocols 'none'
2021-05-02T21:21:19.704+0900 W ASIO [main] No TransportLayer configured during NetworkInterface
startup
2021-05-02T21:21:19.706+0900 I CONTROL [initandlisten] MongoDB starting : pid=26204 port=27017 db
path=c:\bigdata\mongodb\var 64-bit host=DESKTOP-UK9JJQG
2021-05-02T21:21:19.706+0900 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008
R2
2021-05-02T21:21:19.706+0900 I CONTROL [initandlisten] db version v4.2.13
2021-05-02T21:21:19.706+0900 I CONTROL [initandlisten] git version: 82dd40f60c55dae12426c08fd7150
d79a0e28e23
```

명령프롬프트 창을 종료하면 프로그램이 종료됨

2. MongoDB 실행

● MongoDB 실행

- 서버용 프로그램이기 때문에 외부에서도 접속 허용해주어야 함



2. MongoDB 실행

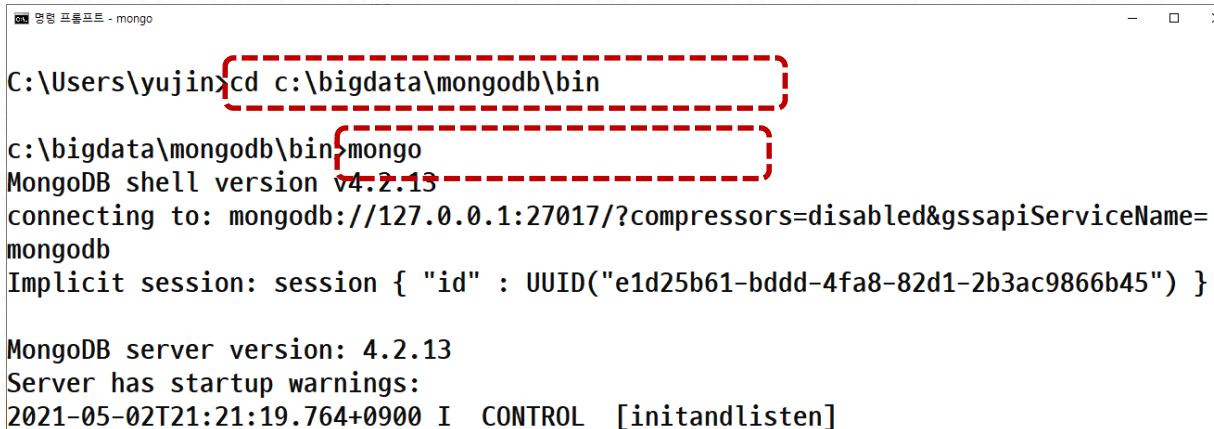
● MongoDB 실행

▪ MongoDB Shell 실행

새로운 명령프롬프트 창을 열고

✦ `cd c:\bigdata\mongodb\bin`

✦ `mongo`



```
명령 프롬프트 - mongo
C:\Users\yujin>cd c:\bigdata\mongodb\bin
c:\bigdata\mongodb\bin>mongo
MongoDB shell version 4.2.13
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("e1d25b61-bddd-4fa8-82d1-2b3ac9866b45") }

MongoDB server version: 4.2.13
Server has startup warnings:
2021-05-02T21:21:19.764+0900 I CONTROL [initandlisten]
```

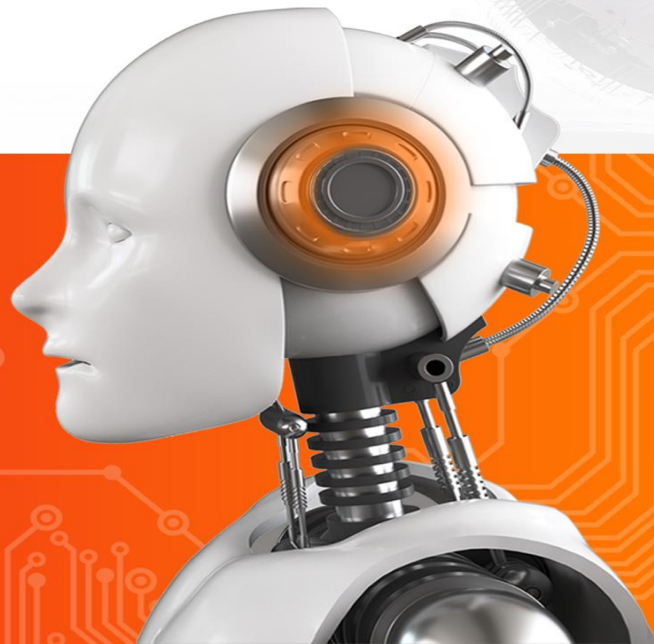


4 MongoDB 비정형 데이터 저장





4.2 MongoDB 비정형 데이터 저장 개요



GrideFS

- MongoDB 안에 비정형 데이터 (이미지, 동영상 등) 을 저장 할 수 있는 파일 시스템
- 1개 파일당 최대 2GB까지 저장 가능
- 각 파일은 256K (chunk) 단위로 나뉘어져서 저장

GrideFS 구조

- files : 파일의 정보 저장
- chunks : 파일의 내용을 256K 씩 나눠서 저장

GrideFS files

파일 크기, 저장한 날짜 등 파일의 정보가 저장됨

files의 document 구조를 알아보겠습니다.

```
{
  "_id"      : <unspecified>,    // unique ID for this file
  "length"   : data_number,      // size of the file in bytes
  "chunkSize" : data_number,      // size of each of the chunks. Default is 256k
  "uploadDate" : data_date,      // date when object first stored
  "md5"       : data_string      // result of running the "filemd5" command on th
is file's chunks
}
```

위 내용은 기본적으로 생성되는 필드이며, 아래와 같이 임의로 지정한 여러필드를 추가할 수 있습니다.

```
{
  "filename" : data_string,      // human name for the file
  "contentType" : data_string,  // valid mime type for the object
  "aliases" : data_array of data_string, // optional array of alias strings
  "metadata" : data_object,     // anything the user wants to store
}
```

chunks Top

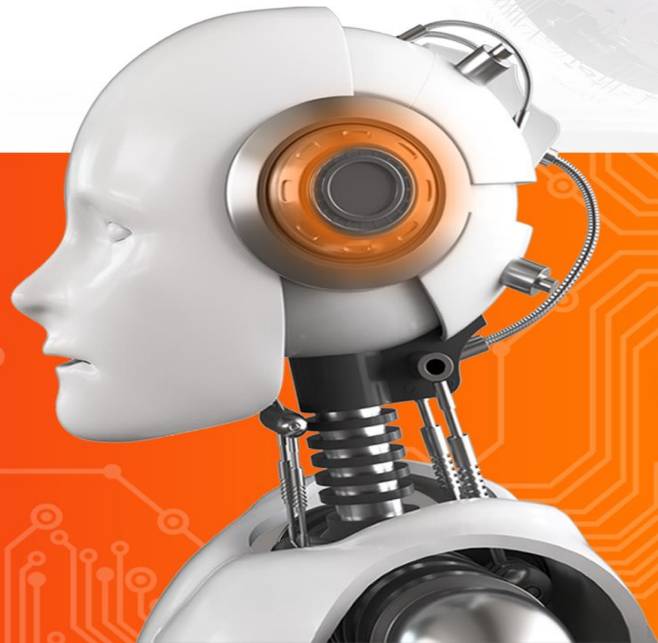
files collection 과 1:n 으로 관계지어지는 collection 입니다.

```
{
  "_id"      : <unspecified>,      // object id of the chunk in the _chunks collection
  "files_id" : <unspecified>,      // 일명 files.id FK 라고 생각하면 됩니다.
  "n"       : chunk_number,        // 256k 단위 chunk의 순번입니다. (예) 1,2,3
  "data"    : data_binary,         // BSON binary 형태의 데이터입니다.
}
```

files.id와 chunks.filesid 는 FK 형식으로 이어지는 구조입니다.



4.3 MongoDB 비정형 데이터 저장



GrideFS 에 저장 할 파일 생성

1. Mogodb GridFS에 저장할 파일 test.txt를 c:\wai\workspace\mongodb에 생성
2. test.txt의 내용을 입력하고 파일 저장



Mongodb GridFS에 test.txt저장

새로운 명령 프롬프트 창을 열어서 다음의 명령을 입력

선택 명령 프롬프트

```
C:\Users\yujin>cd c:\bigdata\mongodb\bin
```

```
c:\bigdata\mongodb\bin>mongofiles put c:\ai\workspace\mongodb\test.txt
```

```
2021-05-02T21:27:28.926+0900
```

```
connected to: mongodb://localhost/
```

```
2021-05-02T21:27:29.107+0900
```

```
added gridFile: c:\ai\workspace\mongodb\test.txt
```

```
c:\bigdata\mongodb\bin>
```


Mongodb GridFS에 저장된 데이터 확인

23페이지에서 실행한 mongodb 창에서 다음의 명령을 입력

명령 프롬프트 - mongo

```
> show collections
```

mongodb에 저장된 테이블 리스트 조회

```
fs.chunks
```

```
fs.files
```

```
> db.fs.files.find()
```

mongodb에 저장된 파일의 정보 조회

```
{ "_id" : ObjectId("608e9ab1e7fe1a256c4322d1"), "length" : NumberLong(35), "chunk  
Size" : 261120, "uploadDate" : ISODate("2021-05-02T12:27:29.106Z"), "filename" :  
"c:\\ai\\workspace\\mongodb\\test.txt", "metadata" : { } }
```

```
> db.fs.chunks.find()
```

mongodb에 저장된 파일의 내용 조회

```
{ "_id" : ObjectId("608e9ab1e7fe1a256c4322d2"), "files_id" : ObjectId("608e9ab1e7  
fe1a256c4322d1"), "n" : 0, "data" : BinData(0,"66q96r0gREIgR1JJREZT7JeQI0ygg0yep  
e2VoCDrgrTsmqk=") }
```

```
>
```



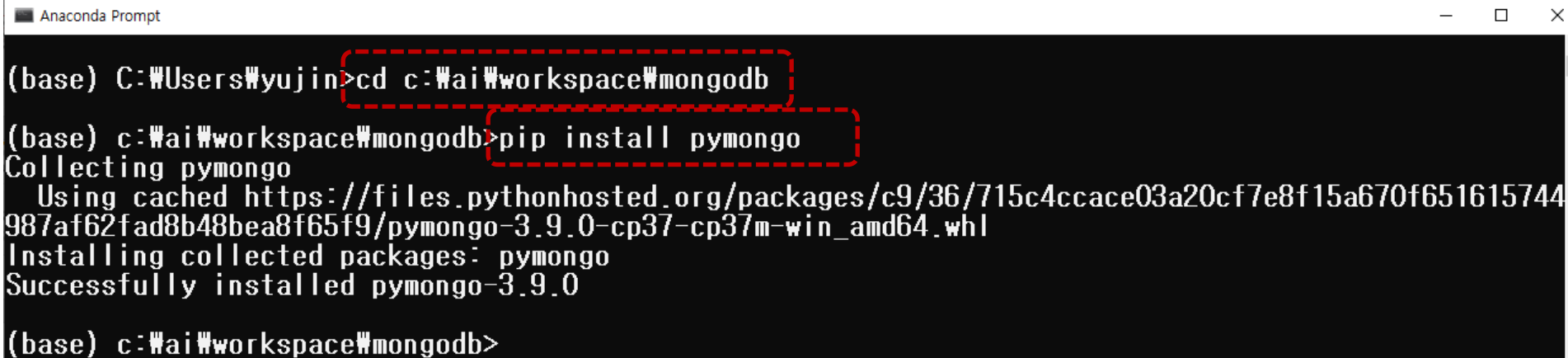
5 Python을 이용한 MongoDB test.txt파일 저장



라이브러리 설치

아나콘드 프롬프트에서 라이브러리 설치

pip install pymongo



```
Anaconda Prompt
(base) C:\Users\yuujin>cd c:\Wai\workspace\mongodb
(base) c:\Wai\workspace\mongodb>pip install pymongo
Collecting pymongo
  Using cached https://files.pythonhosted.org/packages/c9/36/715c4ccace03a20cf7e8f15a670f651615744987af62fad8b48bea8f65f9/pymongo-3.9.0-cp37-cp37m-win_amd64.whl
Installing collected packages: pymongo
Successfully installed pymongo-3.9.0
(base) c:\Wai\workspace\mongodb>
```

프로그램 개요

python을 이용해서 MongoDB에 test.txt 파일을 저장하고 읽어옴

```
In [30]: #라이브러리 임포트  
from pymongo import MongoClient  
from gridfs import GridFS  
from bson import ObjectId
```

```
In [31]: #mongodb에 python_test데이터 베이스에 접속  
db = MongoClient().python_test
```

```
In [32]: #python_test에 파일을 저장할 GridFS객체 생성  
fs = GridFS(db)
```

```
In [33]: #c:/ai/workspace/mongodb/test.txt를 읽을 객체 f생성  
with open("c:/ai/workspace/mongodb/test.txt", 'rb') as f:  
    #f를 통해서 데이터를 읽어서 GridFS에 저장  
    fs.put(f,filename="test.txt")
```

```
In [20]: #GridFS에 저장된 파일 조회
db.fs.files.find()
```

```
Out[20]: <pymongo.cursor.Cursor at 0x25113d8e588>
```

```
In [21]: list(db.fs.files.find())
```

```
Out[21]: [{'_id': ObjectId('5de3d0e45ec102f1fcb78935'),
  'filename': 'test.txt',
  'md5': '0b97c2704cfb0e1892bb47a0b62bef96',
  'chunkSize': 261120,
  'length': 35,
  'uploadDate': datetime.datetime(2019, 12, 1, 14, 40, 36, 704000)}]
```

```
In [27]: #GridFS에 저장된 test.txt파일을 읽을 객체 f생성
f = fs.get_last_version(filename="test.txt")
```

```
In [28]: #test.txt파일의 내용을 읽어서 data에 저장
data=f.read()
data
```

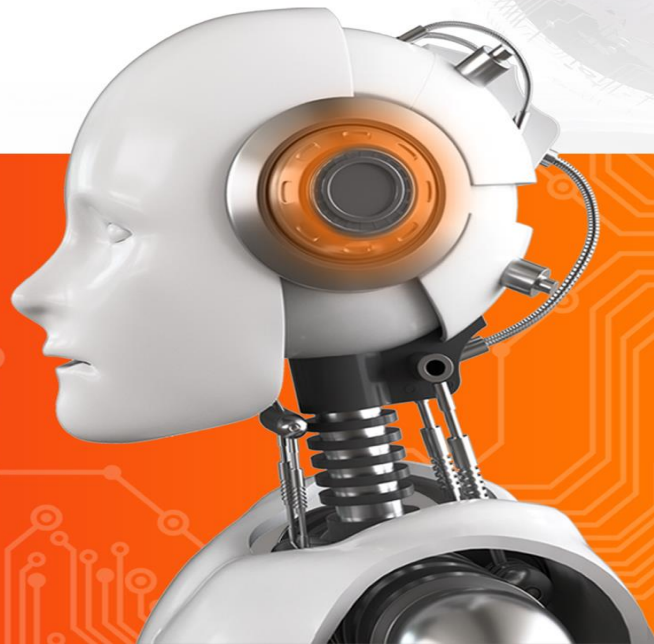
```
Out[28]: b'\xeb\xaa\xbd\xea\xb3\x0DB GRIDFS\x97\x90 \x0\x80\x9e\xa5\xed\x95\x0 \xeb\x82\xb4\xec\x9a\xa9'
```

```
In [29]: #data에 저장된 내용을 utf-8 로 인코딩하여 출력
data.decode('utf-8')
```

```
Out[29]: '몽고DB GRIDFS에 저장할 내용'
```



6 Python을 이용한 MongoDB 이미지 파일 저장



```
In [1]: #라이브러리 임포트  
from pymongo import MongoClient  
from gridfs import GridFS  
from bson.objectid import ObjectId  
from gridfs import GridFSBucket
```

```
In [3]: #mongodb에 python_test데이터 베이스에 접속  
db = MongoClient().python_test
```

```
In [4]: #python_test에 파일을 저장할 GridFS객체 생성  
fs = GridFS(db)
```

```
In [5]: import urllib.request  
#다운로드 받을 이미지 URL  
url = "https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png"
```

```
In [6]: #이미지의 확장자 리턴  
image_type=url.split(".")[-1]  
image_type
```

```
Out [6]: 'png'
```



```
In [10]: #이미지 파일을 저장할 객체 생성  
bucket = GridFSBucket(db)
```

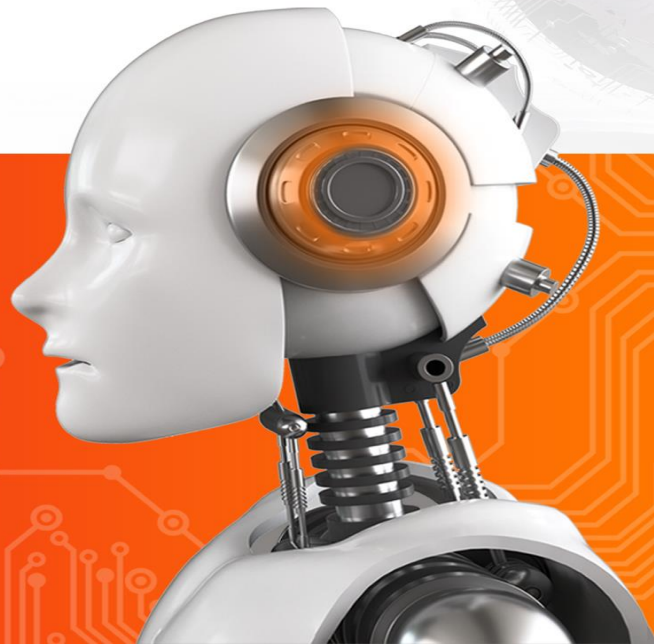
```
In [11]: #이미지 파일을 GridFS에 저장할 객체 grid_in 생성  
grid_in = bucket.open_upload_stream(  
    image_name, metadata={"contentType":content_type })
```

```
In [12]: #이미지의 내용을 GridFS에 저장  
grid_in.write(image)
```

```
In [13]: #이미지 저장 종료  
grid_in.close()
```



7 Python을 이용한 MongoDB 에 저장된 이밋 조회



라이브러리 설치

pip install image

pip install pillow

```
In [1]: #라이브러리 임포트  
from pymongo import MongoClient  
from gridfs import GridFS  
from bson.objectid import ObjectId  
from gridfs import GridFSBucket
```

```
In [2]: from PIL import Image  
import urllib.request  
from io import BytesIO
```

```
In [3]: #mongodb에 python_test데이터 베이스에 접속  
db = MongoClient().python_test
```

```
In [4]: #python_test에 파일을 저장할 GridFS객체 생성  
fs = GridFS(db)
```

```
In [7]: #GridFS 에 저장된 데이터 정보 조회  
db.fs.files.find()
```

```
Out [7]: <pymongo.cursor.Cursor at 0x2d63431c550>
```

```
In [8]: #GridFS 에 저장된 데이터 정보 조회
list(db.fs.files.find())
```

```
Out [8]: [{'_id': ObjectId('608ea36a2fb5d3c3998f3c02'),
  'filename': 'test.txt',
  'md5': '0b97c2704cfb0e1892bb47a0b62bef96',
  'chunkSize': 261120,
  'length': 35,
  'uploadDate': datetime.datetime(2021, 5, 2, 13, 4, 42, 298000)},
 {'_id': ObjectId('608ea3dd3f625d039c5239c0'),
  'filename': 'googlelogo_color_272x92dp.png',
  'metadata': {'contentType': 'image/png'},
  'chunkSize': 261120,
  'md5': '8f9327db2597fa57d2f42b4a6c5a9855',
  'length': 5969,
  'uploadDate': datetime.datetime(2021, 5, 2, 13, 6, 40, 74000)}]
```

```
In [12]: #GridFS 에 저장된 {번째} 데이터 조회
file_detail=db.fs.files.find()[1]
file_detail
```

```
Out [12]: {'_id': ObjectId('608ea3dd3f625d039c5239c0'),
  'filename': 'googlelogo_color_272x92dp.png',
  'metadata': {'contentType': 'image/png'},
  'chunkSize': 261120,
  'md5': '8f9327db2597fa57d2f42b4a6c5a9855',
  'length': 5969,
  'uploadDate': datetime.datetime(2021, 5, 2, 13, 6, 40, 74000)}
```

```
In [13]: #파일명 조회
file_detail["filename"]
```

```
Out [13]: 'googlelogo_color_272x92dp.png'
```

GridFS에 저장된
파일 리스트에서
구글 이미지의 인덱스를 입력



```
In [21]: #파일명이 일치하는 마지막 파일의 내용을 가져올 객체 f 리턴
f = fs.get_last_version(filename=file_detail["filename"])
```

```
In [22]: #파일의 내용을 읽어서 data에 저장
data=f.read()
data
```

```
0xb2wx94wx9fwxfb$xa2wx9dwx11wx3kwx19wxc7;Uwx86<wxa6;wxc1wxfeW;
wxe9xdbwxc2Qwx9bwx2wx3wx10Uwd7:wx0wxc%f%&Q_wx1ewxebwx04bwwx07.
0wx99wxfawxbawx11wxc1wx83lwx7fwxb2wxdceWxeawtwx91wx93wd8wx8dEwx;
wx0wx08 wxf6wxc2wx8c8wtwxa4wxcwxc8d/wx13wx1f >wxc4ewxe1wx1fVwx3;
7vwx0ewxc4wx9fwxc3jwx80.Ywxe2/vwxabx=wxc9cEwx88~wxdbwxfawx18wxc8wx;
8wxfawxadwx1fwrxwx8Gwd3wx95wxa3wxc1wx89,Bwxa0)/.wx02)dwxcdGwx89w;
a0wx9ewx8awxc1wx14wx02wxfdwxc0c*wx16z}wxc2Zwx15wx08bwx8f8wxrMwx97a~w;
swx12~wx9eXwxa2}6jwxa54w*wx07wx88wx8dwx3wx5wx8d9wx04wx82wxfcwx19;
wx94wx8GTAWxcfc<wxfbYwxebwx01wx1bwx8e1wxd3wxc4#wxcwxc.Ywxe6wx8e9cw;
$0wxe1wxdwx81wxa3wx0eSwx80jwxfwfwxeewx07wx11?wxc3Pwx5wx97lwx9f:w;
wxa2wxfawx84wxa0wx5wx82zL!<,wxd5wxdawx19wx8e8wx8a8wx1wx8bwx8b;
v}wxfawx87wx9dHwxadwxba0wx85wx10wxfwxdKwx8d3wx9cwx8b4wx8d8wx03wx;
hN>fwx94wxe8wxf5pwx1^wxd7wx08wxe2wx18hwx944wx10wx81wx14mwxe3wx0e;
b1wnwxd3wxf3".wxd2wwx15wx13wxc8wxbwxc10ywx10wxf4wxbwxcwDwxacwx82wx;
e6wxb5wx89wx04wx82wx1bwx8fwwx1eJwx82wx14%wxd5zwx15wx15TJwxe0%wx13w;
bwx01wxeacJ}wx1fwx83wx8b9wxbw<wtwx16S~wxc9wx8aMwx7fJwx8b6wxf5q"wx81;
wtwx046wx97wx84wxd2wxf83wxb~wxcfwxe4wxf7wx97wx8b8u)wx04wx8cwx11wx;
w'|=kwx11wxd8=wxc1wx81wxc4Mwxwxc1dw81wxdwxc8f)wx8fbpZwr><`7Gwx8cwx;
6wxe8y1wxdhwx8dwxcfwV|wx04wxfewxcfwxe6F wxc6+wx94<wx04wx8ewx12%wx;
wxa4wxbwxc8wxd5wx98wxbwxc7fFwxbwxcwxfawxf5wxbfMwx8dwx84wxe7wxf;
```

```
In [34]: from IPython.display import display
```

```
In [36]: #data의 내용을 이미지로 변환해서 image에 저장  
image = Image.open(BytesIO(data))  
image|
```

Out [36]:

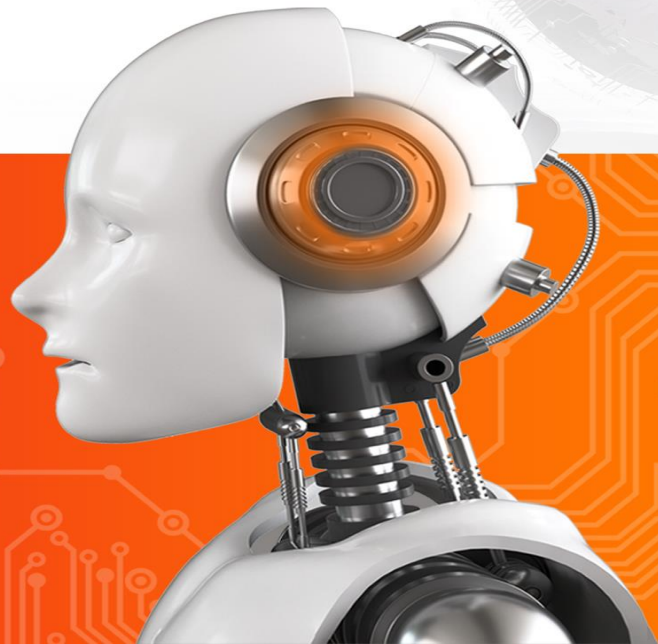


```
In [37]: #image출력  
display(image)
```





8 Python을 이용한 이미지 분류

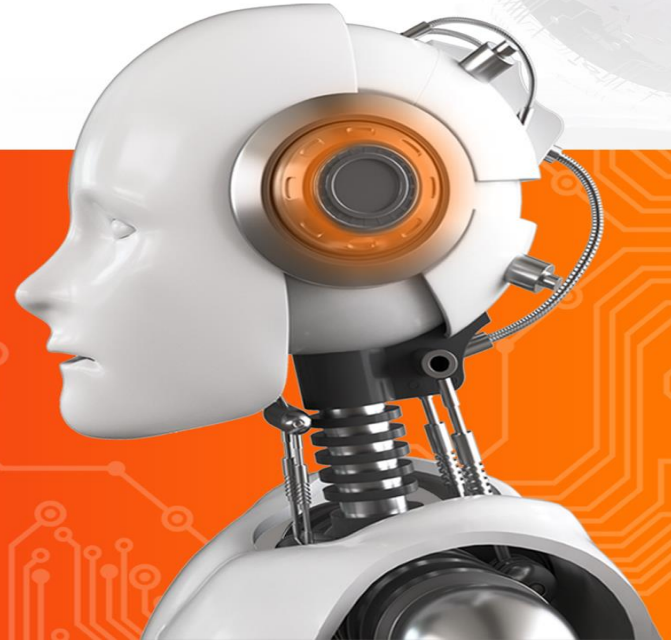


프로젝트 개요

1. 강아지와 고양이 이미지를 수집해서 MongoDB에 저장
2. Support Vector Machine 알고리즘을 이용하여 강아지와 고양이 이미지 분류 학습
3. c:\wai\workspace\mongodb폴더에 테스트 이미지를 저장하고 강아지 이미지인지 고양이 이미지 인지 판별



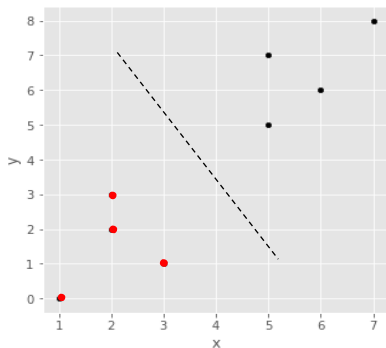
8.1 Support Vector Machine



Support Vector Machine(SVM) 개요

1) SVM 개념

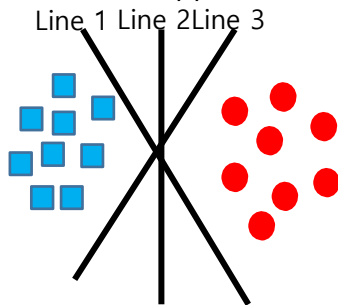
- "Support Vector Machine"은 분류 또는 회귀 문제에 사용할 수 있는 기계 학습 알고리즘
- 대부분 분류 문제에 사용
- SVM 알고리즘에서는 각 데이터 항목을 n 차원 공간 상 하나의 점으로 표시
- 아래와 같이 이질적인 두 개 또는 그 이상의 데이터 집단을 잘 구분하는 최적의 초평면 (Optimal Hyper Plane)을 찾는 방법을 제공



Support Vector Machine이란?

1) SVM 개념(예시)

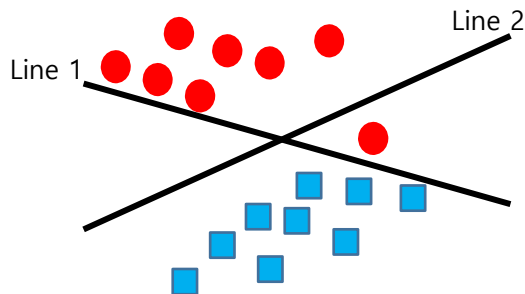
- SVM을 이용하여 빨간 원과 하늘색 사각형을 구분(Classify) 한다면 어떤 선으로 구분해야 하나?
- 답은 Line 2임
- Line 2인 이유는 Margin 때문
- Margin이란 초평면 가까이에 있는 Support Vector에서 초평면까지의 거리의 합을 의미함 (얼마나 가까이에 있는 Support Vector까지 사용할이지는 Parameter 설정에 따라 다름)
- Line 1이나 Line 3같은 경우 가까이에 있는 Support Vector가 조금 위치를 바꾸면 다른 그룹에 포함하게 됨



Support Vector Machine이란?

1) SVM 개념(예시)

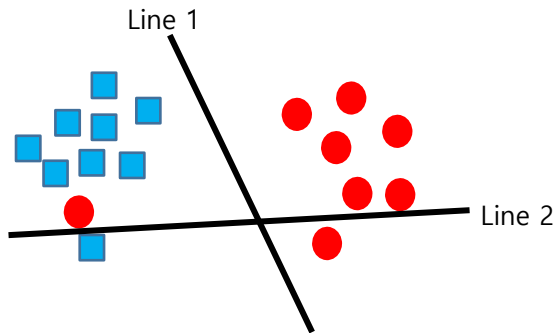
- 다음 그림에서는 SVM을 사용하면 어떤 선으로 분류를 하나?
- Classification 관점에서 보면 Line 1이 맞고 Margin 관점에서 보면 Line 2가 맞지만 정답은 Line 1임
- 이유는 SVM은 Classification을 한 다음에 Margin이 가장 큰 선을 선택하기 때문에



Support Vector Machine이란?

1) SVM 개념(예시)

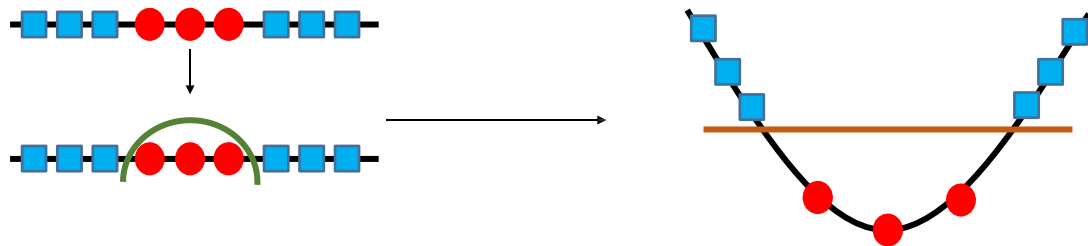
- 다음 그림에서는 SVM을 사용하면 어떤 선으로 분류를 하나?
- 정답은 Line 1임
- 선형 SVM은 Outlier를 어느 정도 무시하며 최선의 선택을 함



Support Vector Machine이란?

2) 비선형 분류

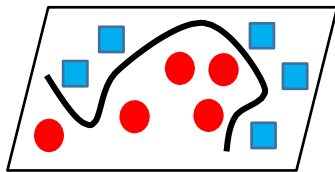
- SVM은 선형분류와 더불어 비선형 분류에서도 사용
- 비선형 분류를 하기 위해서는 주어진 데이터를 고차원 특징 공간으로 사상하는 작업이 필요
- 효율적으로 실행하기 위해서 커널트릭을 사용
- 이러한 Kernel을 이용하여 차원을 변경하게 되면, 흩어져있는 데이터에 대해서도 차원을 변경하여 간단하게 나눌 수 있다는 장점을 가짐
- 주요 Kernel은 Linear Kernel, Polynomial Kernel, RBF(Radial Basis Function)이 있음



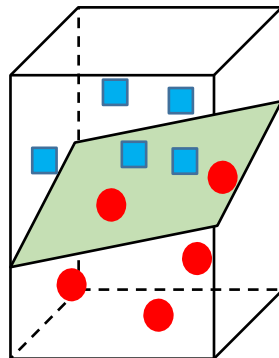
Support Vector Machine이란?

2) 비선형 분류

- 각각의 커널에서는 최적화를 도와주는 파라미터들이 따로 존재함
- 일반적으로 각 문제에 대해서 어떠한 커널의 파라미터를 선택하는 것이 가장 좋은지를 자동적으로 알려주는 방법은 없음
- 실험을 통해 모든 조건을 바꾸면서 SVM의 학습과 예측을 반복해서 최적의 예측률을 보여주는 조건을 찾아야 함



입력 공간

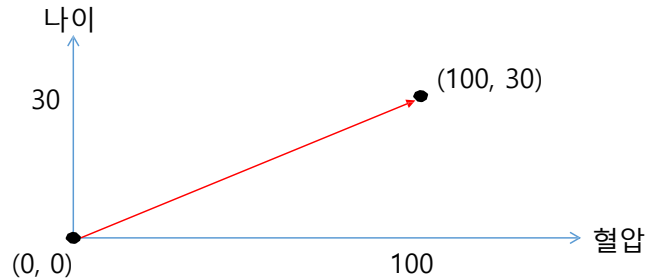


특징 공간

Support Vector Machine(SVM) 개요

3) 벡터 이론 복습 - 초평면을 구하기 위한 기초 이론

- n 개의 특징 (특징, 변수)으로 사람을 기술함
- 어떤 사람의 특징 값은 아래 그림과 같이 화살표로 표시할 수 있음
 - 예 :
 - 2 가지 특징으로 설명되는 사람 : 혈압 = 100, 나이 = 30

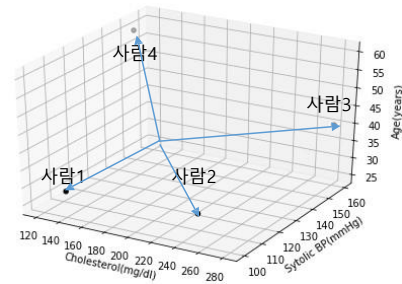


Support Vector Machine(SVM) 개요

3) 벡터 이론 복습 – 초평면을 구하기 위한 기초 이론

- 아래와 같은 사람 4명이 있다고 가정
- 이 때 각 사람별 건강상태를 콜레스테롤, 혈압, 나이 등의 속성값으로 표현할 수 있음
- 이 각 속성의 집합을 특징이라고 함.
- 각 사람별 특징값은 3차원 벡터로 나타낼 수 있음
- 이것을 그림으로 표현하면 오른쪽 그림과 같이 시작점이 (0, 0, 0)이고 끝점이 각 특징값으로 이루어지는 화살표 모양

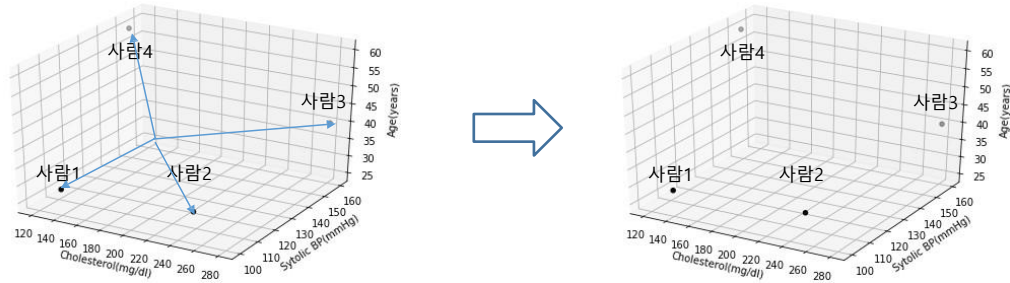
사람	콜레스테롤 (mg/dl)	혈압 (mmHg)	나이 (years)	벡터 시작점	벡터 끝점
1	140	100	30	(0, 0, 0)	(140, 100, 30)
2	230	115	25	(0, 0, 0)	(230, 115, 25)
3	120	150	60	(0, 0, 0)	(120, 150, 60)
4	280	160	40	(0, 0, 0)	(280, 160, 40)



Support Vector Machine(SVM) 개요

3) 벡터 이론 복습 – 벡터 표시의 단순화

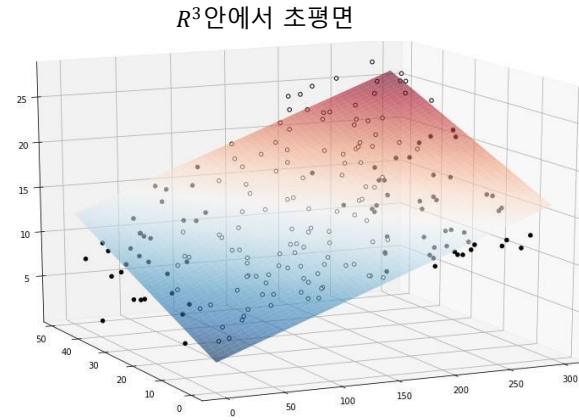
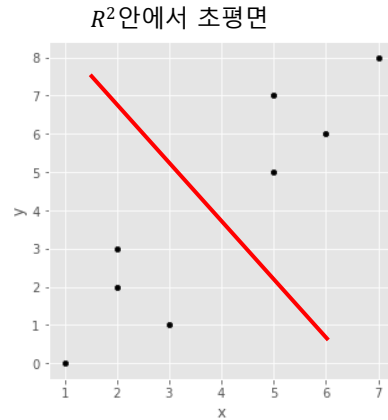
- 모든 벡터는 시작점이 0이므로 편의상 끝점만을 표시할 수 있음
- 왼쪽의 예제 벡터는 오른쪽 그림과 같이 점만 표시하기로 함



Support Vector Machine(SVM) 개요

4) 초평면의 개념

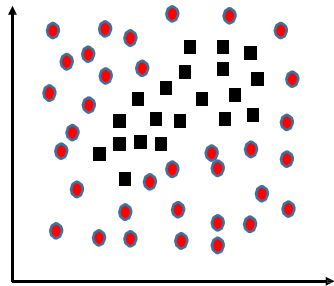
- 2차원 공간(R^2)에서의 초평면은 아래 왼쪽 그림과 같이 나타낼 수 있고
- 3차원 공간(R^3)에서의 초평면은 아래 오른쪽 그림과같이 나타낼 수 있음



Support Vector Machine(SVM) 개요

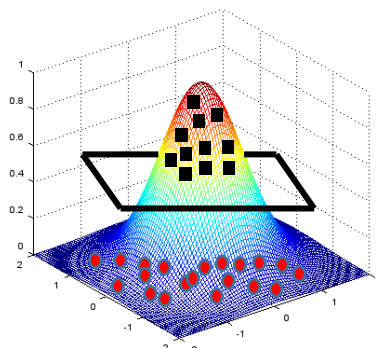
9) 커널트릭

- 선형으로 분리할 수 있는 데이터가 아닐 때 사용



데이터가 입력 공간에서
선형으로 분리되지 않음

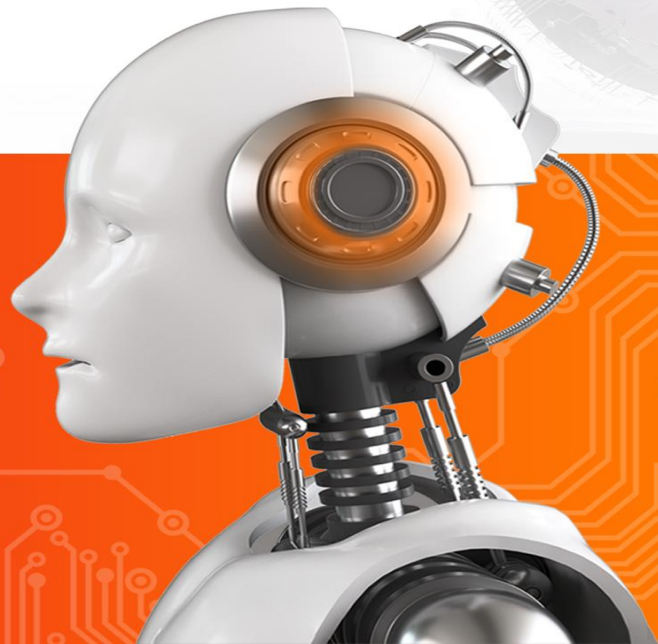
커널
 Φ



커널에 의해 얻어진 특정 공간
에서 선형으로 분리 가능함



8.2 강아지 고양이 이미지 분류

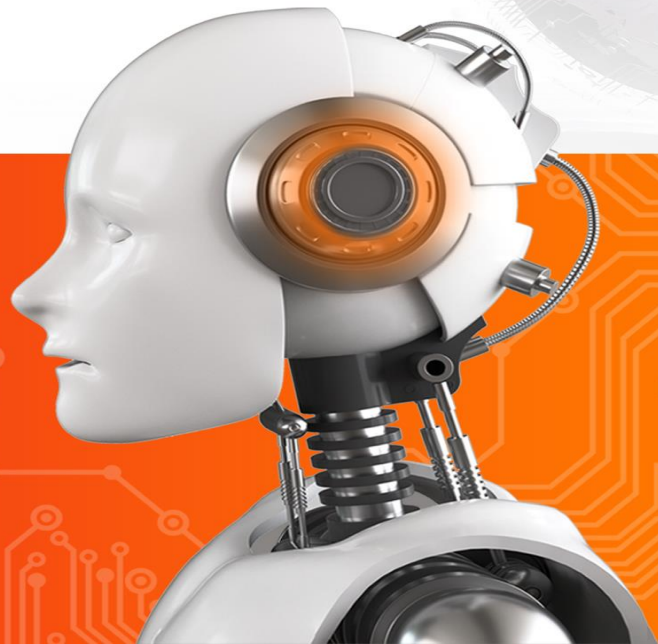


라이브러리 설치

pip install opencv-python



8.3 강아지 고양이 이미지 검색



개요

open API를 이용해서 고양이이미지, 강아지 이미지의 url을 각각 1000개씩 리스트에 저장

```
In [1]: import requests

#API 인증 정보
client_id = "udzt3jcCA8xTKi3UjGH_"
client_secret = "Hak0g9n8cs"

#header에 추가될 내용
headers = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}
```

```
In [2]: # 네이버 OEPN API에서 고양이와 강아지 이미지를 검색
# keyword : 검색어 (고양이, 강아지)
# display : 검색 결과의 개수
# start : 시작 인덱스
def get_api_result(keyword, display, start):
    # 네이버 이미지 검색 URL
    url = "https://openapi.naver.com/v1/search/image?query=" + keyword +
        "&display=" + str(display) +
        "&start=" + str(start)
    # 네이버 이미지 검색 URL 를 실행하고 검색 결과를 result에 저장
    result=requests.get(url, headers=headers)
    # 이미지 검색 결과 리턴
    return result.json()
```

```
In [3]: # keyword : 검색어 (고양이, 강아지)
# total_page : 검색할 전체 페이지 (100개씩 10페이지 전체 1000 개의 이미지를 검색)
def call_and_print(keyword, total_page=10):
    # 네이버 이미지 검색 결과에서 이미지를 다운 받을 url을 저장 할 리스트
    link_list=[]
    # 1에서 부터 total_page(10페이지)+1 미만 만큼 반복
    for page in range(1,total_page+1):
        # 한페이지당 100개의 이미지 정보를 검색
        display = 100
        # 페이지의 시작 인덱스 한페이지당 100개씩 검색 하기 때문에 1페이지는 1, 2페이지는 101, 3페이지는 201 ...
        start = ((page-1)*display)+1
        # get_api_result(keyword, display, start) : 네이버 이미지 검색 API를 호출하고 결과를 리턴받아서 json_obj 에 대입
        json_obj = get_api_result(keyword, display, start )
        # 검색 결과에서 json_obj['items'] 에 이미지를 다운 받을 URL이 저장되 있으므로 json_obj["item"]을 link_list에 저장
        for item in json_obj['items']:
            link_list.append(item['link'])
    # 이미지를 다운로드할 url이 저장된 link_list를 리턴
    return link_list
```

```
In [4]: keyword = "고양이"
# call_and_print 함수를 호출해서 고양이 이미지를 검색하고 이미지를 다운 받을 URL을 리턴 받음
link1=call_and_print(keyword)
link1
```

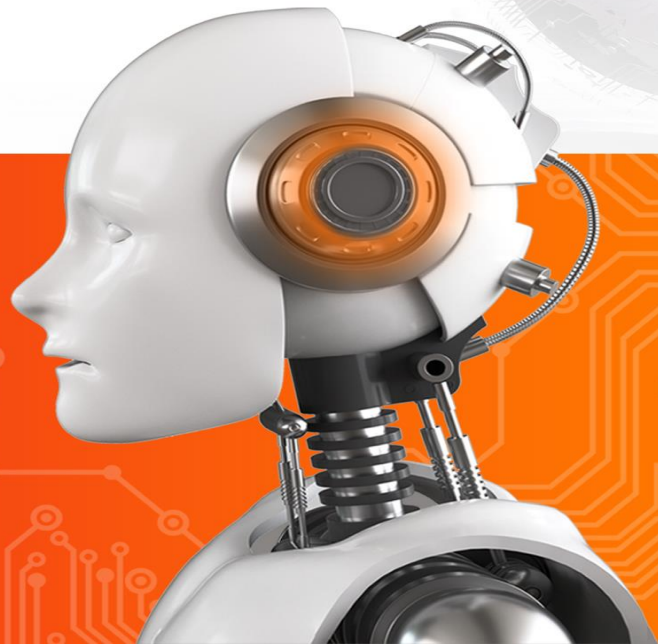
```
Out [4]: ['http://post.phinf.naver.net/MjAyMTAxMjdfMjcw/MDAxNjExNzUxOTExNjMw.QFi-nG__LR0tBP_dUII8Z24weuJyY7iRd423YnQ2acog.0tRRXMDMU
dyjbYpGfU0VFMDFN0F07I5xSlg.JPEG/IJJfWmwGLyY4DbIdHWZJ35cN4dhU.jpg',
'http://post.phinf.naver.net/MjAyMTA0MTJfNjk4/MDAxNjE4MTk4Njc3ODAy.ywIDcIB-Dqp-MuuVn_Kj_NZHcER9z9XH2iguU0pS1MugID26PUanGz
87ZU_Wd8qjjjAXpC_qIzGJwg.JPEG/I5moydFuUMwFi3oEOPX959RIIZqY.jpg',
```

```
In [5]: keyword = "강아지"
# 네이버 API에서 강아지 이미지를 검색하고 다운로드 받을 URL을 리턴 받음
link2=call_and_print(keyword)
link2
```

'http://img.dogpre.com/web/dogpre/reviewphoto/3/thumb/34283_detail_32301492399366.jpg',
 'http://imgnews.naver.net/image/038/2013/02/27/2013022713364299_1_rstarcby_59_20130227140623.jpg',
 'http://post.phinf.naver.net/MjAyMDEyMzBfNzUgMDAxNjA5Mjk1ODIzOTI2L3DlFrs1ic2yyLNnQJRIygcMbChQZns_ru7jjcIA220g.HPmuwABq26rfG0mWmD
 5xfGmHD9mz_hFB0JZ6YRQaKWog.JPEG/IMLfIqMgVKPKdKXnCO3Skz-XpUQ.jpg',
 'http://shop1.phinf.naver.net/20210224_128/1614174156140bwmNU_JPEG/15309998831862944_1375725873.JPG',
 'http://post.phinf.naver.net/MjAxNzEyMDVfOTEgMDAxNTEyNDU3ODA5NjE3.UKBjGH2bQ_JnGr53p6dnc4WTJy6MDxrX6v6bmFC6G-Qg.X5jKz-_AUGIUci-00_
 7t1iuz0vXX3EDJ8rRhXznRC-wg.JPEG/IAtDAKFzaXi78IMNsQeMeK1Yck8g.jpg',
 'https://phinf.pstatic.net/shop/20201111_115/1605067031348NIdf2_JPEG/6202859150194180_925289682.jpg',
 'http://post.phinf.naver.net/MjAyMDEyMDdfMjA3MDAxNjA3Mjk2NTMOMzc2.M7EmrsEEEnCG4p7znL_UIIgWZWAU84H5I1JbgFYKISGAgn.dnS0ThMyn8VX4VwkDj
 vpNKYP76eedy3SPAETelSMaSug.JPEG/lxOyxhLUjrwwTo3HvwTtLpr6an4A.jpg',
 'http://post.phinf.naver.net/MjAxNjEyMTJfMTQ1MDAxNDgxNTQ0NzQ1MzQ2.OjJ3-2PA8BNl2ty4oAmvRRo-tgPU6OFKq0m9YYCt_uYg.J9Ww6hup__eYB8qm1j
 bjubltC2tZ_pq3p6_l18lrIOkg.JPEG/INkNHVPEoJaG-d3Bd8CIISucJd4s.jpg',
 'http://img.dogpre.com/web/dogpre/reviewphoto/3/thumb/2632_detail_81001552380185.jpg',
 'https://i.pinimg.com/736x/8a/9c/75/8a9c7510892169d63dde39ce66f98f21--springer-spaniel-puppies-english-springer-spaniels.jpg',
 'http://post.phinf.naver.net/MjAxNzExMDZfMTQwMDAxNTA5ODk3MzkxNzIx.twr0le2livisRD03BlufOE8xFbQDC8jqYDitLhz2lbcbg.0IBEt04S9-UgdqkAc
 Zsm9vEQBNdyJx0ZE1r5IEnllog.JPEG/lJ5PqJuGBdfeI9AQAz5yivj8lpTO.jpg',
 'http://imgnews.naver.net/image/5511/2020/11/11/0000046301_001_20201111105425940.jpg',
 'https://i.pinimg.com/736x/c3/06/3e/c3063e7fef56f4ad31501f5a8667a158.jpg',
 'http://imgnews.naver.net/image/5511/2020/10/08/0000043391_001_20201008085328795.jpg',
 'http://post.phinf.naver.net/MjAyMTYyMTMfMTQwMDAxNjEzNjI2ODkwNDcy.uVEKjVfknyxtVkWgdAMlWVCYBuJhg3PHG9SVJm5joN8g.aruCmo-e9Xh-UKEuoq



8.4 이미지 MongoDB 저장



이미지 파일 MongoDB 저장

이미지 파일 url을 읽어서 mongoDB에 저장

이미지 파일을 MongdoDB에 저장 전에 기존 파일 삭제

2. MongoDB 실행

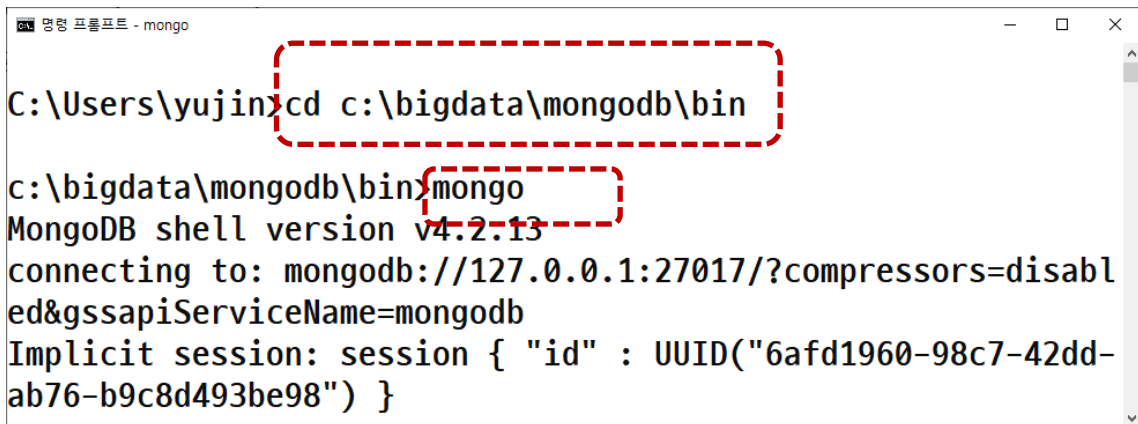
● MongoDB 실행

▪ MongoDB Shell 실행

새로운 명령프롬프트 창을 열고

✦ cd c:\bigdata\mongodb\bin

✦ mongo



```
C:\Users\yujin>cd c:\bigdata\mongodb\bin

c:\bigdata\mongodb\bin>mongo
MongoDB shell version v4.2.13
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6afd1960-98c7-42dd-ab76-b9c8d493be98") }
```


Mongodb 기존 이미지 삭제

```
> use python_test
switched to db python_test
> db.fs.chunks.remove({})
WriteResult({ "nRemoved" : 3832 })
> db.fs.files.remove({})
WriteResult({ "nRemoved" : 1934 })
>
```

```
In [6]: #라이브러리 임포트  
from pymongo import MongoClient  
from gridfs import GridFS  
from bson.objectid import ObjectId  
from gridfs import GridFSBucket
```

```
In [7]: #mongodb에 python_test데이터 베이스에 접속  
db = MongoClient().python_test
```

```
In [8]: #python_test에 파일을 저장할 GridFS객체 생성  
fs = GridFS(db)
```

```
In [9]: #이미지 파일을 저장할 객체 생성  
bucket = GridFSBucket(db)
```

```
In [10]: import urllib.request
```

```

In [11]: # link1 : 고양이 이미지를 다운로드 할 URL 1000 개가 저장된 리스트
for url in link1:
    try:
        # url에서 고양이 이미지를 다운 받아서 변수 image에 저장
        image=urllib.request.urlopen(url).read()

        # url을 / 을 기준으로 분리하고 마지막의 이미지파일명과 확장자를 리턴
        image_name=url.split("/")[-1]

        # image_name 에서 ? 가 있으면
        if image_name.find("?") !=-1:
            # image_name을 ? 를 기준으로 분리하고 ? 의 첫번째 문자열을 image_type에 대입
            image_name = image_name.split("?")[0]

        # 이미지의 기본 타입
        image_type = "jpg"

        #image_name 에서 . 를 포함하고 있으면
        if image_name.find(".") != -1:
            # image_name(고양이 이미지의 파일명과 확장자) 을 . 기준으로 분리하고 마지막 문자열 즉 이미지의 확장자 리턴
            image_type=image_name.split(".")[1]

        # image/이미지의 확장자를 content_type에 대입
        content_type="image/{}".format(image_type)

        #이미지 파일의 정보를 저장한 grid_in 생성
        grid_in = bucket.open_upload_stream(
            image_name, # 이미지 파일의 이름
            metadata={"contentType":content_type,"type":"cat" } # 이미지 파일 타입
        )

        #이미지의 내용을 GridFS에 저장
        grid_in.write(image)
        #이미지 저장 종료
        grid_in.close()
        print("image_type:",image_type,"content_type:",content_type,
            ":image_name:", image_name)
    except:
        print("에러 발생")

```

```
image_type: jpg :content_type: image/jpg :image_name: 12b0f777425e101c041025c1cef19d90.jpg
image_type: jpg :content_type: image/jpg :image_name: loNB1TxuziqQManUCbJV5S50MM3s.jpg
image_type: jpg :content_type: image/jpg :image_name: 5e97e7eed07d929587ec0b888f738748--ginger-kitten-ginger-cats.jpg
image_type: jpg :content_type: image/jpg :image_name: l_aEE6mC8JdXbaz3A1B3pQld7UJ0.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000028733_001_20200523123807150.jpg
image_type: jpg :content_type: image/jpg :image_name: 83707bb3cf3516cc298af9062d0e7f77--kucing-do-you.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000652200_001_20200619153105914.jpg
image_type: JPG :content_type: image/JPG :image_name: 2112721023_8kj2cT0v_IMG_5168.JPG
image_type: jpg :content_type: image/jpg :image_name: 7fs2h0x_20201020183913123364.jpg
image_type: jpg :content_type: image/jpg :image_name: lWY8iP_lCQJ4Lmpmo3Yv0Vs58Z8U.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000026392_001_20200501184653000.jpg
image_type: jpg :content_type: image/jpg :image_name: PYH2013110303290001300_P2_59_20131103153803.jpg
image_type: jpg :content_type: image/jpg :image_name: l7vVC-H_urT_nMZxJnLkqggsV0uY.jpg
image_type: png :content_type: image/png :image_name: 0000100294_001_20210308160250631.png
image_type: jpg :content_type: image/jpg :image_name: lOUiPRZXUv-xnQ4Ep3EmfonGM-X4.jpg
image_type: jpg :content_type: image/jpg :image_name: lspwt0VdV1JSFcMPcmfTiQzJcVGv.jpg
image_type: jpg :content_type: image/jpg :image_name: lWkXVHVZ7K1lSYuLpECttNtRkBS0.jpg
image_type: php :content_type: image/php :image_name: viewimage.php
image_type: jpg :content_type: image/jpg :image_name: 22276902eeb7685974936ee589c43ba7--funny-kitties-kitty-cats.jpg
image_type: png :content_type: image/png :image_name: 0000763775_001_20201215120011591.png
```

```

In [12]: # link2 : 강아지 이미지를 다운로드 할 URL 1000 개가 저장된 리스트
for url in link2:
    try:
        # url에서 강아지 이미지를 다운 받아서 변수 image에 저장
        image=urllib.request.urlopen(url).read()

        # url을 / 을 기준으로 분리하고 마지막의 이미지파일명과 확장자를 리턴
        image_name=url.split("/")[-1]

        # image_name 에서 ? 가 있으면
        if image_name.find("?") != -1:
            # image_name을 ? 를 기준으로 분리하고 ? 의 첫번째 문자열을 image_type에 대입
            image_name = image_name.split("?")[0]

        # 이미지의 기본 타입
        image_type = "jpg"

        #image_name 에서 . 를 포함하고 있으면
        if image_name.find(".") != -1:
            #image_name (강아지 이미지의 파일명과 확장자) 을 . 기준으로 분리하고 마지막 문자열 즉 이미지의 확장자 리턴
            image_type=image_name.split(".")[1]

        # image/이미지의 확장자를 content_type에 대입
        content_type="image/{}".format(image_type)

        #이미지 파일의 정보를 저장한 grid_in 생성
        grid_in = bucket.open_upload_stream(
            image_name, # 이미지 파일의 파일명과 확장자
            metadata={"contentType":content_type,"type":"dog" } # 이미지 파일의 정보
        )

        #이미지의 내용을 GridFS에 저장
        grid_in.write(image)
        #이미지 저장 종료
        grid_in.close()
        print("image_type:", image_type, ":content_type:", content_type,
              ":image_name:", image_name)
    except:
        print("에러 발생")

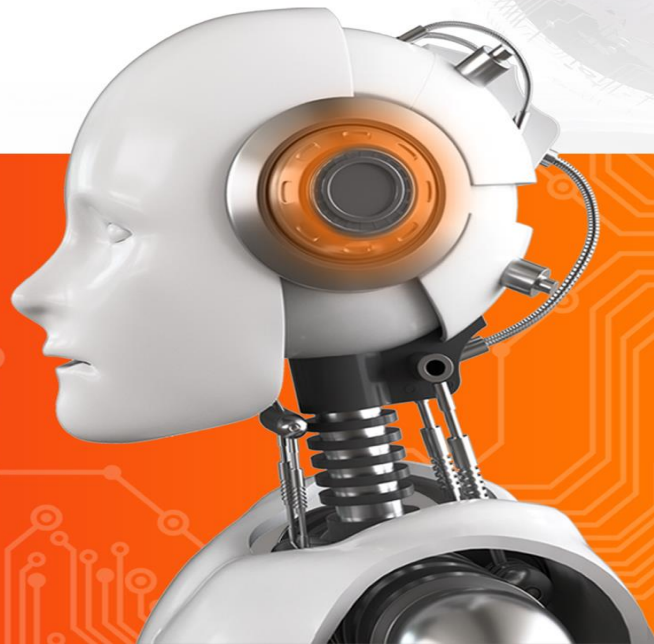
```

image_type: jpg :content_type: image/jpg :image_name: 000000010_001_2021000112000000.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000027662_001_20200513095605791.jpg
image_type: jpg :content_type: image/jpg :image_name: mug_obj_140297010838565278.jpg
image_type: jpg :content_type: image/jpg :image_name: 16661265075004797_1260062299.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000046063_005_20201108120155205.jpg
image_type: jpg :content_type: image/jpg :image_name: 17465409405756329_1331555580.jpg

image_type: jpg :content_type: image/jpg :image_name: 1711hrsLYgeSX-ZbfmQDP5PYGxkg.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000051477_001_20210219112912575.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000714044_001_20210119135102004.jpg
image_type: jpg :content_type: image/jpg :image_name: 20161120000275_0_99_20161120150104.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000052553_001_20210311151819612.jpg
image_type: jpg :content_type: image/jpg :image_name: 18039654142655229_1335458914.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000048465_001_20201219143903297.jpg
image_type: jpeg :content_type: image/jpeg :image_name: mosa7LY15n.jpeg
image_type: jpg :content_type: image/jpg :image_name: 0000053741_003_20210331103634676.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000030433_001_20200609100416993.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000049371_001_20210110123552470.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000049332_001_20210109132421013.jpg
image_type: jpg :content_type: image/jpg :image_name: 0000030587_001_20200610160415802.jpg
image_type: jpg :content_type: image/jpg :image_name: 25899788997_20210206203951.jpg



8.5 MongoDB 저장된 이미지 읽기



개요

1. MongoDB에 저장된 이미지를 읽어서 리스트에 저장
2. 1을 numpy 배열로 변환


```
In [1]: #Mogodb 연관 라이브러리 임포트  
from pymongo import MongoClient  
from gridfs import GridFS  
from bson.objectid import ObjectId  
from gridfs import GridFSBucket
```

```
In [2]: #이미지 연관 라이브러리 임포트  
from PIL import Image  
import urllib.request  
from io import BytesIO
```

```
In [3]: #mongodb에 python_test데이터 베이스에 접속  
db = MongoClient().python_test
```

```
In [4]: #python_test에 파일을 저장할 GridFS객체 생성  
fs = GridFS(db)
```

```
In [5]: #GridFS 에 저장된 데이터 정보 조회
image_list=list(db.fs.files.find())
image_list
```

```
Out [5]: [{'_id': ObjectId('5de4080191955d6fa4751c49'),
  'filename': '0000019325_004_20190416171424294.jpg',
  'metadata': {'contentType': 'image/jpg', 'type': 'cat'},
  'chunkSize': 261120,
  'md5': '7cd40ee41a78ea622f3856d5cba18487',
  'length': 99292,
  'uploadDate': datetime.datetime(2019, 12, 1, 18, 35, 45, 982000)},
 {'_id': ObjectId('5de4080291955d6fa4751c4b'),
  'filename': '20150203001206_0_99_20150204074103.jpg',
  'metadata': {'contentType': 'image/jpg', 'type': 'cat'},
  'chunkSize': 261120,
  'md5': '4b784cbf72b85e54976c92f18dcf26b9',
  'length': 34465,
  'uploadDate': datetime.datetime(2019, 12, 1, 18, 35, 46, 3000)},
 {'_id': ObjectId('5de4080291955d6fa4751c4d'),
  'filename': 'lx-Q2Hr7WP-HyMNdYFg8at7R6MKg.jpg',
  'metadata': {'contentType': 'image/jpg', 'type': 'cat'},
  'chunkSize': 261120,
  'md5': '996402d44bafa6e02ad469694e99fd22',
  'length': 57746}
```

```
In [6]: #이미지의 가로 세로  
IM_WIDTH = 200  
IM_HEIGHT = 200
```

```
In [7]: #Mongodb에서 이미지를 읽어서 저장할 리스트  
images=[]  
#고양이 이미지 1 강아지 이미지 0 을 저장할 리스트  
labels=[]
```

```
In [8]: import numpy as np
```

```

In [9]: for file_detail in image_list:
        #파일명 조회
        file_name=file_detail["filename"]
        print("file_name:",file_name)
        #파일명이 일치하는 마지막 파일의 내용을 가져올 객체 f 리턴
        f = fs.get_last_version(filename=file_name)
        #파일의 내용을 읽어서 data에 저장
        data=f.read()
        if len(data)>0 :
            #data의 내용을 이미지로 변환해서 im에 저장
            im = Image.open(BytesIO(data))
            #컬러이미지를 흑백으로 변환 convert('L')
            im = im.convert('L')
            #이미지 파일의 크기를 100x100 으로 수정
            im = im.resize((IM_WIDTH,IM_HEIGHT))
            #이미지 파일에 저장된 RGB값은 0~255사이의 값이 저장된 있음
            #255로 나눴서 값들을 0~1 사이의 실수로 변환

            #이미지를 2차원 배열로 변환 :np.array(im)
            #2차원 배열을 1차원 배열로 변환 : .flatten( )
            im = np.array(im).flatten( ) / 255.0
            #im에 저장된 데이터의 타입을 float32로 변환
            im = im.astype("float32")
            #images에 im을 추가
            images.append(im)
            # file_detail["metadata"]["type"] 에 cat 이 포함되 있으면 1 아니면 0을 label에 저장
            label = 1 if "cat" in file_detail["metadata"]["type"] else 0
            #labels에 label추가
            labels.append(label)

```

file_name: 2013022713364299_1_rstarcby_59_20130227140623.jpg
file_name: 1MLflqMgVKPDKkGxnC03Skz-XpUQ.jpg
file_name: 15309998831862944_1375725873.JPG
file_name: 1AtDAKfzaXi78IMNsQeMek1YCK8g.jpg
file_name: 6202859150194180_925289682.jpg
file_name: 1x0yxhLUjrwvTo3HvwTtLpr6an4A.jpg
file_name: 1NkNHVPEoJaG-d3Bd8CII SucJd4s.jpg
file_name: 2632_detail_81001552380185.jpg
file_name: 8a9c7510892169d63dde39ce66f98f21--springer-spaniel-puppies-english-springer-spaniels.jpg
file_name: 1j5PqJuGBdfei9AQAz5yivj8lpT0.jpg
file_name: 0000046301_001_20201111105425940.jpg
file_name: c3063e7fef56f4ad31501f5a8667a158.jpg
file_name: 0000043391_001_20201008085328795.jpg
file_name: 1A4JAiuHB079fAfg7LwJ34-rUSWw.jpg
file_name: 18069285788676172_1283340283.PNG
file_name: 34289_detail_46151535625770.jpg
file_name: 0000028478_001_20200521082004529.jpg
file_name: hu_1455326476_4250385189.jpeg
file_name: 0000048462_001_20201219143901889.jpg
file_name: 1TLaD02dY6Cfm2Qvk9z-eTnkuFUQ.jpg

```
In [10]: #고양이와 강아지 이미지가 저장된 리스트 images를 배열로 변환해서 리턴  
images_arr=np.array(images)  
#고양이 이미지 0 강아지 이미지 1이 저장된 labels를 배열로 변환해서 리턴  
labels_arr=np.array(labels)
```

```
In [11]: images_arr
```

```
Out[11]: array([[0.24705882, 0.21960784, 0.18431373, ..., 0.24705882, 0.2509804 ,  
                0.2509804 ],  
               [0.6313726 , 0.6431373 , 0.654902  , ..., 0.4627451 , 0.45882353,  
                0.45490196],  
               [0.5372549 , 0.54509807, 0.5529412 , ..., 0.43529412, 0.4       ,  
                0.3764706 ],  
               ...,  
               [0.39607844, 0.39607844, 0.41568628, ..., 0.0627451 , 0.05882353,  
                0.05882353],  
               [0.99607843, 0.99607843, 1.         , ..., 1.         , 1.         ,  
                1.         ],  
               [1.         , 1.         , 1.         , ..., 1.         , 1.         ,  
                1.         ]], dtype=float32)
```

```
In [12]: labels_arr
```

```
Out[12]: array([1, 1, 1, ..., 0, 0, 0])
```

```
In [13]: from sklearn.model_selection import train_test_split
#학습데이터와 테스트데이터로 분리
#image_arr, labels_arr을 7.5 : 2.5 로 분리
X_train, X_test, y_train, y_test = train_test_split(images_arr, labels_arr)
```

```
In [14]: #학습데이터 이미지 조회
X_train
```

```
Out[14]: array([[0.05098039, 0.05098039, 0.04705882, ..., 0.10588235, 0.09019608,
0.0627451 ],
[0.09019608, 0.09019608, 0.10980392, ..., 0.4627451 , 0.29803923,
0.20392157],
[0.64705884, 0.64705884, 0.64705884, ..., 0.7294118 , 0.7294118 ,
0.7294118 ],
...,
[1. , 1. , 1. , ..., 0.92156863, 0.92156863,
0.92156863],
[1. , 1. , 1. , ..., 1. , 1. ,
1. ],
[1. , 1. , 1. , ..., 0.99607843, 1. ,
1. ]], dtype=float32)
```

```
In [15]: #테스트 데이터 이미지 조회  
X_test
```

```
Out [15]: array([[0.9843137 , 0.98039216, 0.9843137 , ..., 0.6745098 , 0.6745098 ,  
                  0.67058825],  
                [1.          , 1.          , 1.          , ..., 1.          , 1.          ,  
                  1.          ],  
                [0.25490198, 0.25882354, 0.25490198, ..., 1.          , 1.          ,  
                  1.          ],  
                ...,  
                [0.9607843 , 0.96862745, 0.93333334, ..., 0.7529412 , 0.7529412 ,  
                  0.7529412 ],  
                [0.28235295, 0.29411766, 0.30588236, ..., 0.23529412, 0.23137255,  
                  0.23137255],  
                [0.69803923, 0.6862745 , 0.6862745 , ..., 0.65882355, 0.65882355,  
                  0.6627451 ]], dtype=float32)
```

```
In [16]: #학습데이터의 분류 조회 (고양이 0 강아지 1)  
y_train
```

```
Out [16]: array([0, 0, 1, ..., 0, 0, 1])
```



```
In [17]: #테스트데이터의 분류 조회 (고양이 0 강아지 1)  
y_test
```

```
Out[17]: array([0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,  
                1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,  
                0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,  
                0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1,  
                1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,  
                1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,  
                1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,  
                1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0,  
                1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,  
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,  
                0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1,  
                0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,  
                1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,  
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,  
                1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,  
                1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,  
                1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,  
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,  
                1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,  
                1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,  
                0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,  
                1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0])
```



8.6 SVM을 이용한 학습



```
In [18]: from sklearn import svm
```

```
In [27]: #X_train과 y_train을 학습  
SVM = svm.SVC().fit(X_train, y_train)
```

```
C:\ai\python\anaconda3\lib\site-packages\sklearn\svm\base.py:193:  
o 'scale' in version 0.22 to account better for unscaled features  
  "avoid this warning.", FutureWarning)
```

```
In [28]: #테스트 데이터의 정확도 출력  
SVM.score(X_test, y_test)
```

```
Out [28]: 0.5743801652892562
```



8.7 SVM을 이용한 예측



예측 이미지 저장

1. 예측 하고자 하는 이미지를 다운로드 받음
2. 1의 이미지를 c:\wai\workspace\mongodb\test01.xxx 로 저장
확장자는 다운로드 받은 파일의 확장자 그대로 유지

```
In [21]: #새로 예측할 이미지를 저장할 리스트  
new_image=[]
```

```
In [22]: #예측할 이미지를 읽어서 image에 저장  
#Image.open(예측하고자 하는 이미지 경로)  
image = Image.open("c:/ai/workspace/mongodb/test01.jpg")  
#컬러이미지를 흑백으로 변환 convert('L') 해서 im에 대입  
im = image.convert('L')  
#이미지 파일의 크기를 200x200 으로 수정  
im = im.resize((IM_WIDTH,IM_HEIGHT))  
  
#이미지를 2차원 배열로 변환 :np.array(im)  
#2차원 배열을 1차원 배열로 변환 : .flatten( )  
#이미지 파일에 저장된 RGB값은 0~255사이의 값이 저장되 있음  
#255로 나뉘서 값들을 0~1 사이의 실수로 변환  
im = np.array(im).flatten( ) / 255.0  
#im에 저장된 데이터의 타입을 float32로 변환  
im = im.astype("float32")  
#images에 im을 추가  
new_image.append(im)
```

```
In [23]: #new_image를 numpy배열로 변환
new_image_arr=np.array(new_image)
new_image_arr
```

```
Out [23]: array([[0.5411765 , 0.5411765 , 0.54901963, ..., 0.5372549 , 0.5294118 ,
                  0.5372549 ]], dtype=float32)
```

```
In [24]: #이미지가 고양이 인지 강아지인지 분류
#고양이 1 강아지 0
predict=SVM.predict(new_image_arr)
#분류 결과 조회
predict[0]
```

```
Out [24]: 1
```

```
In [25]: import matplotlib.pyplot as plt
```

```
In [26]: plt.rc('font', size=20)
#이미지를 출력 (전체 가로 30,세로 30)
fig = plt.figure(figsize=(30,30))
#예측값 대입
model_out = predict[0]
#예측값이 1일때
if model_out == 1:
    #cat대입
    str_label='cat'
else:
    #dog 대입
    str_label='dog'
#이미지의 제목으로 예측한 이미지의 이름 출력
plt.title('predict: %s' % str_label)
#이미지 출력
plt.imshow(image)
#그래프를 화면에 그림
plt.show()
```






9. CNN 을 이용한 이미지 분류

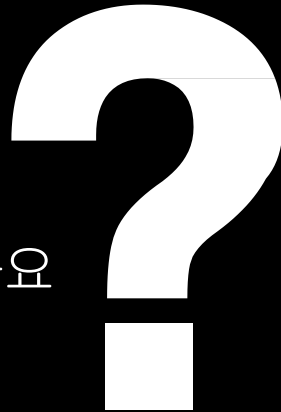




9.1 CNN (Convolutional Neural Network) 개요



우리 눈에는 당연한 것이
왜 컴퓨터에겐 어려운 걸까요







둘을 구분할 수 있겠습니까?

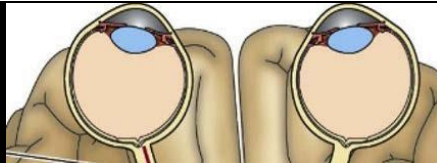


그렇다면, 컴퓨터의 관점에서 볼까요?

8	55
180 181 181 182 182 183 183 182 178 176 17	255 212 220 219 209 255 255 255 255 255 2
6	55
176 176 176 176 176 175 175 173 172 172 17	255 211 214 214 214 255 255 255 255 255 2
2	55
179 179 179 180 180 181 181 181 183 184 18	255 211 211 212 212 255 255 255 255 255 2
4	19
185 186 187 187 187 182 180 180 180 180 18	255 211 213 213 213 255 255 255 211 211 2
0	14
180 180 180 179 178 177 176 176 182 182 18	255 214 214 214 214 255 255 212 212 210 2
2	09
183 183 184 184 184 186 186 187 188 189 19	255 214 215 215 211 255 215 209 212 220 2
0	19
190 189 185 183 183 183 183 183 183 183 18	255 35 22 213 212 212 212 211 211 214
2	214
182 181 180 179 179 185 185 185 186 186 18	214 35 31 214 214 214 212 212 211 211
7	212
187 187 188 189 189 190 192 193 193 194 18	212 213 213 216 210 213 219 219 211 213 2
8	13
188 188 188 188 188 188 188 188 187 186 18	213 212 212 212 211 211 214 214 214 214 2
5	14
185 185 194 194 194 194 194 195 195 195 19	214 214 214 212 212 214 212 212 216 219 2
6	55
196 196 197 197 198 198 197 199 196 196 19	223 224 222 228 226 221 207 181 145 212 2
5	55
195 194 194 193 194 194 194 194 194 194 19	212 212 212 212 212 212 214 214 255 255 2
9	55
199 199 199 199 199 199 199 200 200 200 20	214 214 214 212 212 216 219 221 255 255 2
1	55
201 202 202 202 201 201 201 200 200 199 19	217 217 216 199 169 144 209 208 255 255 2
9	55
198 199 199 199 199 199 199 203 203 203 20	211 210 209 214 214 216 218 218 255 255 2
3	55
203 203 203 203 204 204 204 205 205 206 20	212 211 212 211 216 222 223 255 255 255 2
6	55
206 204 203 203 203 202 202 202 202 203 20	213 212 212 212 211 211 214 255 255 255 2
3	55
203 203 203 203 207 207 207 206 206 206 20	214 214 214 212 212 211 211 255 255 255 2
5	55
205 206 207 207 207 208 208 209 209 206 20	214 214 214 212 212 211 211 255 255 255 2
6	55

이제, 둘을 구분할 수 있겠나요?

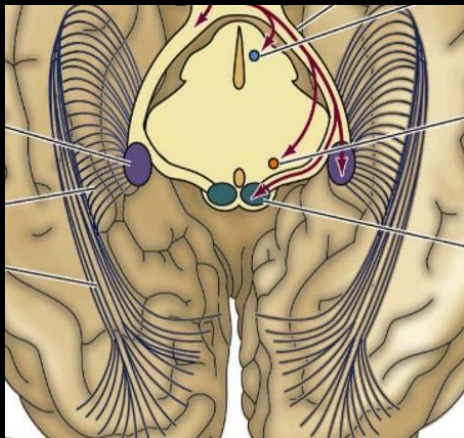
보는 것과 이해하는 것은 다릅니다.



우리가보는데걸린시간
110년

우리가 본 것을 이해하는데 걸린 시간

540만 년



단순한 숫자들의 나열로부터
어떻게 이미지를 이해할 수 있을까요?

해결의 출발점은 퍼즐을 맞추듯



간단한 원리에서 출발합니다



173 173 173 172 170 170 170 170 170 170 170
 169 168 167 166 166 166 166 172 173 173 173 174
 174 175 175 175 177 178 178 178 178 179 179 178
 174 174 174 174 175 176 176 176 176 177 177 172
 171 170 170 170 171 172 172 176 182 182 182 178
 180 181 181 181 182 183 183 182 182 182 182 176
 176 176 176 176 176 176 175 175 173 173 172 172
 179 179 179 181 181 181 181 181 181 184 184 184
 185 186 187 187 187 187 187 180 180 180 180 180
 180 180 180 179 179 184 184 184 187 182 182 182
 183 183 184 184 184 184 184 187 188 189 190 190
 190 189 185 185 183 183 183 183 183 183 183 182
 182 181 181 181 179 179 185 185 185 186 186 187
 187 187 187 188 188 189 190 192 193 193 194 188
 188 188 188 188 188 188 188 188 187 186 185 185
 185 185 184 194 194 194 194 194 196 196 195 196
 196 196 197 197 198 198 197 199 199 196 196 195
 195 195 194 194 194 194 194 194 194 194 199 199
 199 199 199 199 199 199 199 200 200 200 201 201
 201 202 202 202 201 201 201 200 200 199 199 199
 198 199 199 199 199 199 199 203 203 203 203 203
 203 203 203 203 203 203 203 206 206 206 206 206
 206 204 203 203 203 203 203 203 203 203 203 203
 203 203 203 203 207 207 207 207 206 206 206 205
 205 206 207 207 207 208 208 209 209 206 206

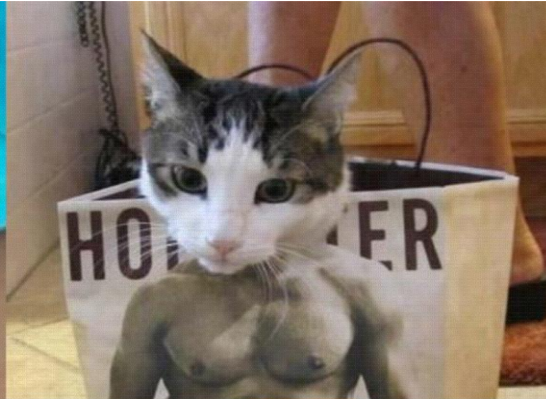
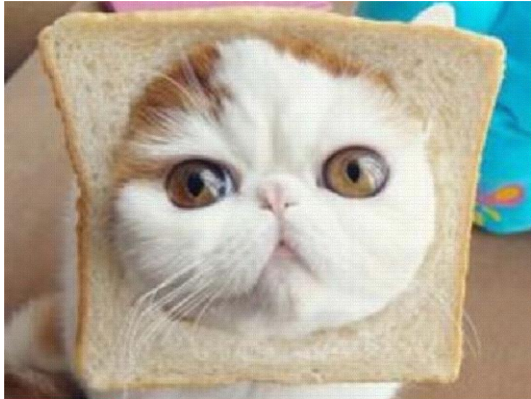
“Cat”

그렇듯해 보이는 이 방법은
전혀 좋은 효과를 거두지 못합니다.

그렇듯해 보이는 이 방법은
전혀 좋은 효과를 거두지 못합니다.

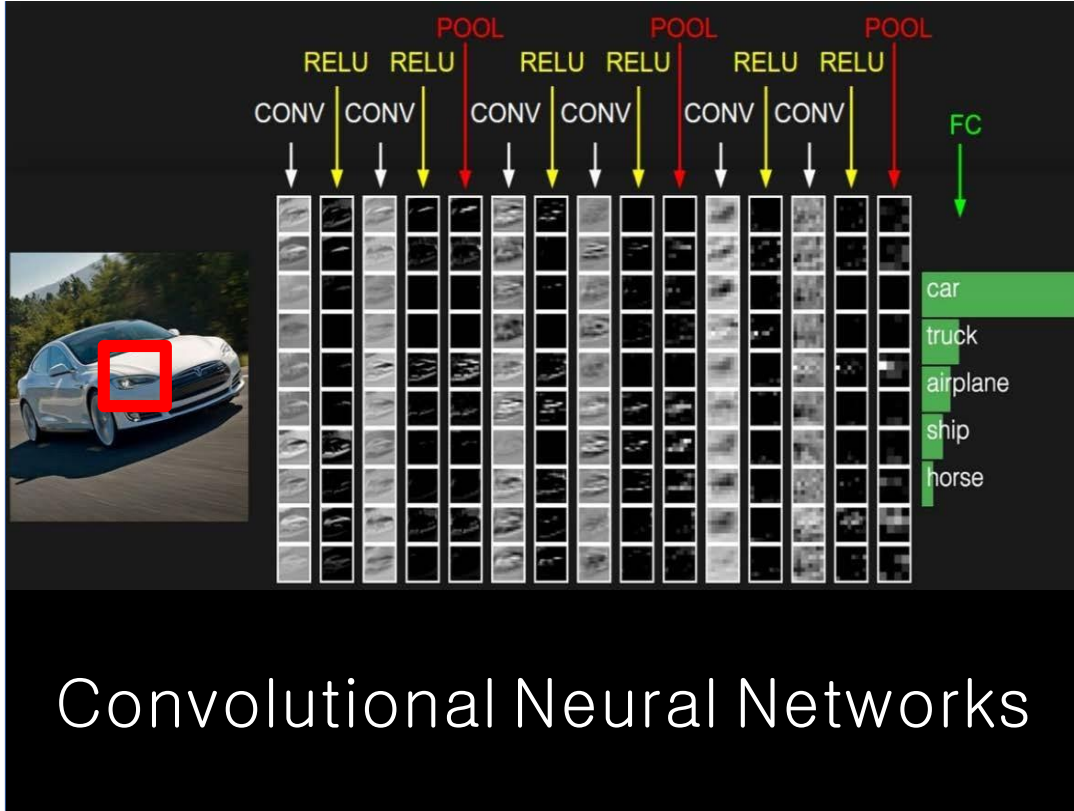
왜?





- 패턴은 예외 상황에 대한 대응이 불가능했고 결국 성능은 매우 미미하였습니다.

2012년, 혁명적인 알고리즘이 탄생하였는데
그것은바로...





예외 상황에
취약했던 기존의
알고리즘에 반해



CNN은 이렇게작은
filter를 순환시키며



CNN은 이렇게작은
filter를 순환시키며



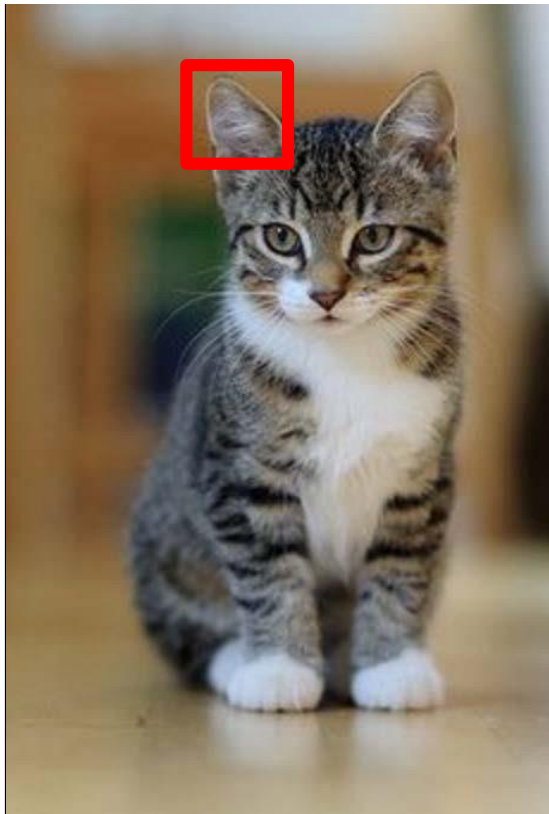
CNN은 이렇게 작은
filter를 순환시키며



CNN은 이렇게 작은
filter를 순환시키며

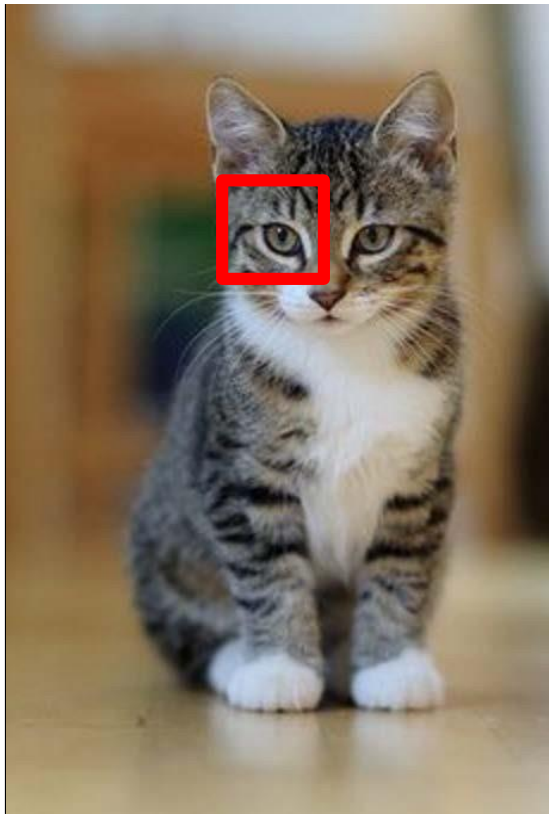


CNN은 이렇게 작은
filter를 순환시키며



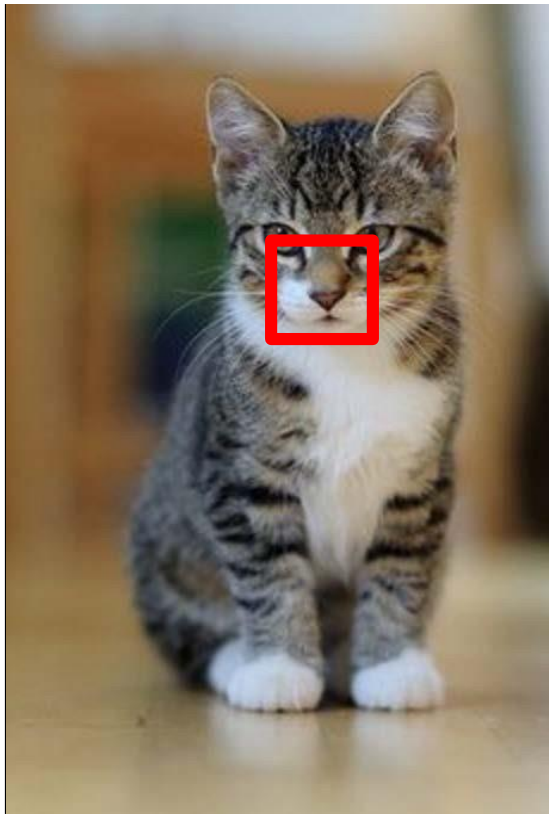
CNN은 이렇게작은
filter를 순환시키며

‘음? 고양이귀다!’



CNN은 이렇게작은
filter를 순환시키며

‘음 ? 고양이귀다!’
‘어 ? 고양이눈이다!’

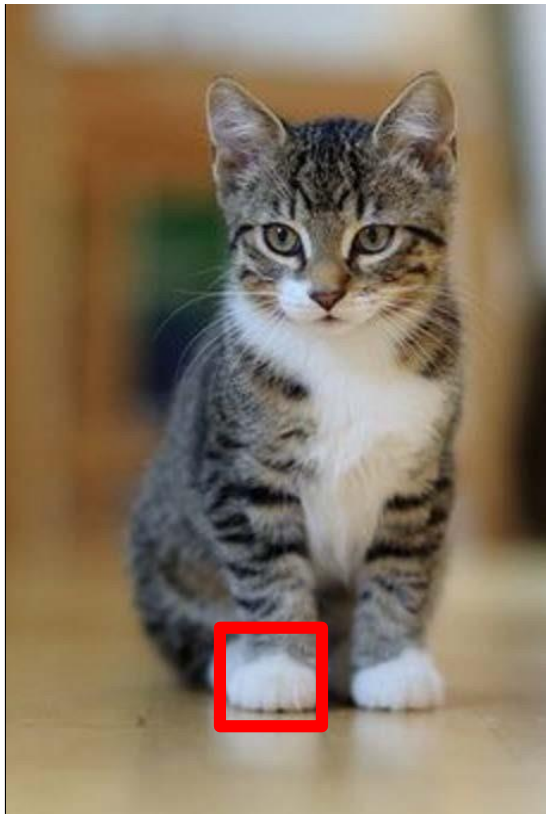


CNN은 이렇게작은
filter를 순환시키며

‘음 ? 고양이귀다!’

‘어 ? 고양이눈이다!’

‘오 ? 고양이코다!’



CNN은 이렇게작은
filter를 순환시키며

- ‘음 ? 고양이귀다!’
- ‘어 ? 고양이눈이다!’
- ‘오 ? 고양이코다!’
- ‘고양이발이네?’



filter를 순환시키며

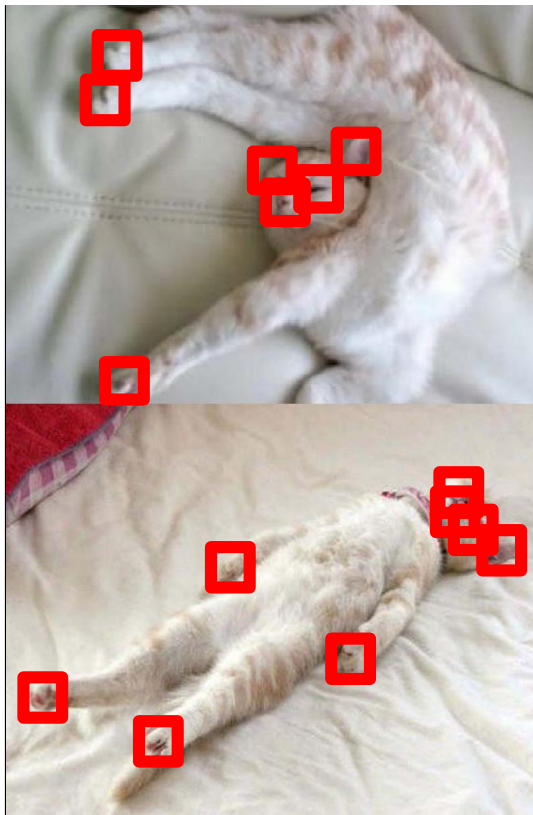
- ‘음 ? 고양이귀다!’
- ‘어 ? 고양이눈이다!’
- ‘오 ? 고양이코다!’
- ‘오호, 고양이발이네?’



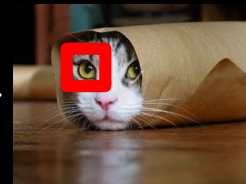
- “이건고양이군!”



이런 방식을 통해
각 특징들이기묘하게
분포되어있어도

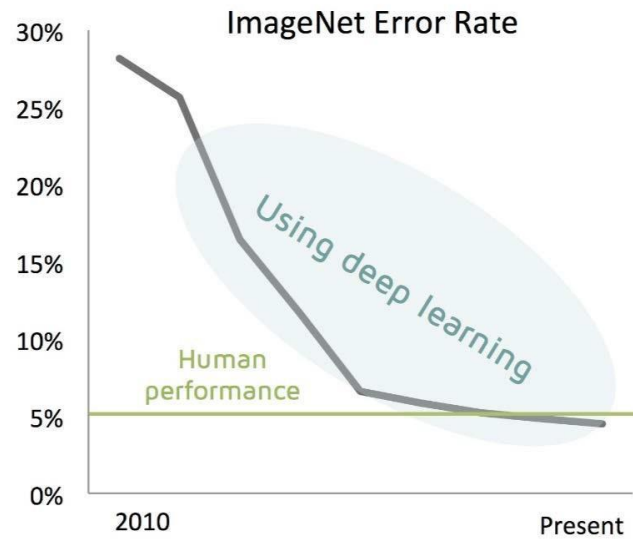


이런 방식을 통해
각 특징들이기묘하게
분포되어있어도
정확하게 찾아내는
높은적응도를가집니다



고양이눈

또한striding 기법을통해다양한크기의
물체에도높은적응도를갖습니다.



CNN(통틀어 Deep Learning)



GPU의 등장에 힘입어 발달은 가속화되었고,
급기야 **인간 성능**을 뛰어넘기에 이르렀습니다