

03. 특징공간과 모델선택

- Python언어와 실습환경구축
- 특징공간과 데이터
- 모델선택

Python과 실습환경 구축

Python이란?

- 1990년 암스테르담의 **귀도 반 로섬(Guido Van Rossum)**에 의해 만들어진 인터프리터 언어
 - 사전적 의미 : 고대 신화 속의 파르나수스(Parnassus) 산의 동굴에 살던 큰 뱀
- 사용 예
 - 구글에서 만들어진 소프트웨어의 50%이상이 파이썬으로 제작됨
 - Dropbox(파일 동기화 서비스) 등
- 공동작업과 유지보수가 매우 쉽고 편하기 때문에 이미 다른 언어로 작성된 많은 프로그램과 모듈들이 Python으로 다시 재구성되고 있는 상황

Python의 특징

- 인터프리터 언어
- 객체지향 지원 / 독립적 실행환경 지원
- 인간친화적 언어
 - `if 4 in [1,2,3,4]: print("4가 있습니다")`
 - 만일 1,2,3,4중에 4가 있으면 "4가 있습니다"를 출력하라.
- 쉬운 문법 / 간결성
- 강력함
 - 시스템 프로그래밍, 하드웨어 제어, 매우 복잡하고 많은 반복연산 등을 제외한 거의 모든 작업을 할 수 있음
 - 다른 언어의 모듈을 Python에 포함할 수 있음
 - (예) 뼈대는 Python, 속도를 요하는 부분은 C언어

Python 응용분야

- 할 수 있는 일
 - 시스템 유틸리티
 - OS의 시스템 명령어 사용가능한 도구 제공
 - GUI프로그래밍
 - Tkinter
 - C/C++언어와의 결합
 - Web 프로그래밍
 - 수치연산 프로그래밍
 - Numerical Python 모듈 제공
 - 데이터베이스 프로그래밍
 - DB연결, pickle
- 할 수 없는 일
 - OS제작
 - 많은 반복과 연산을 필요로 하는 프로그램
 - 데이터압축 알고리즘 등
 - 속도를 요하는 프로그램
 - H/W를 직접 제어하는 프로그램

실습을 위한 환경구축

■ 실습환경

- Python 3.8.x(64-bit)
- IDE: Jupyter Notebook, Visual Studio Code 등을 이용

■ 설치순서

1. Python 3.8.x 설치
2. Command Prompt에서... 라이브러리 설치
 - numpy, matplotlib, seaborn, bokeh
 - scikit-learn
 - Pandas 등
3. IDE 설치

파이썬 및 라이브러리 설치

- python 3.8.x 64-bit 버전 설치
 - python 3.9는 아직 tensorflow 지원 안됨.
- cmd prompt에서... 다음을 차례로 입력 (참고: anaconda가능)

```
python -m pip install --upgrade pip
```

```
pip install jupyter
```

```
pip install numpy
```

```
pip install matplotlib
```

```
pip install seaborn
```

```
pip install bokeh
```

```
pip install pandas
```

```
pip install scikit-learn
```

통합개발환경(IDE) 설치

■ 가능한 IDE

■ Python Shell(Python IDLE)

- python과 동시에 설치됨(반드시 PATH 확인)

■ PyCharm Community

- <https://www.jetbrains.com/ko-kr/pycharm/>
- 많이 사용됨

■ Visual Studio Code

- 크로스 플랫폼을 지원하는 editor
- Windows, macOS, Linux(Ubuntu)를 모두 지원

■ **jupyter notebook(또는 colab)**

- command prompt에서 server 실행 후, Chrome에서 해당 주소로 연결하여 사용
 - > jupyter notebook
- Code와 Markdown 입력 가능

■ Vscode 설치

- <https://code.visualstudio.com/>
- 설치시에 반드시 PATH에 경로 추가

■ Vscode 환경설정

- Extensions(Ctrl+Shift+X)에서 'python'으로 검색 후, Python Extention, Korean Language Pack, Beauty 등을 설치
- 글꼴설정
 - 필요시 D2Coding 폰트 설치
 - 설정 - 텍스트편집기 - 폰트에서 글꼴, 크기 설정

참고: **setting.json**

```
{  
    "terminal.integrated.fontSize": 16,  
    "terminal.integrated.fontFamily": "Consolas, D2Coding",  
    "editor.fontSize": 18,  
    "editor.fontFamily": "Consolas, D2Coding, 'Courier New', monospace",  
    "editor.mouseWheelZoom": true  
}
```

■ Vscode 사용방법

- 폴더를 선택/생성
- 파일을 추가(예: test.py)

■ Vscode 편집을 위한 유용한 단축키

- **Ctrl+C** / **Ctrl+V** / **Ctrl+X**
 - 현재라인 복사하기/붙이기/잘라내기(선택하지 않은 상태)
- **Ctrl+Shift+K**
 - 현재라인 삭제하기(선택하지 않은 상태)
- **Ctrl+Enter(아래)** / **Ctrl+Shift+Enter(위)**
 - 현재라인의 아래 / 위에 빈 라인 추가
- **Alt+↑** / **Alt+↓**
 - 현재라인을 위 / 아래로 이동
- **Ctrl+D**
 - 현재 커서의 단어와 같은 단어를 차례로 선택하여 한꺼번에 바꾸기
- **Alt+클릭** : 멀티 커서

jupyter notebook 설치

■ 준비

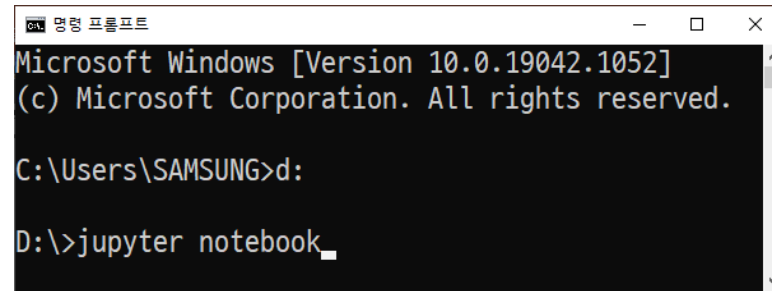
- jupyter notebook은 웹브라우저(IEE, Chrome 등)에서 실행됨
- 가급적 Chrome Browser 설치 권장

■ 설치

- 검색창(Win+R)에 cmd 입력한 후, “명령 프롬프트”에서 설치
- **pip install jupyter**

■ 실행방법

1. 명령 프롬프트 열기(cmd)
2. 노트북파일(*.ipynb)을 저장하기 원하는 디스크로 이동(필요시)
3. jupyter notebook 입력
 - 주의: 명령 프롬프트는 절대 종료시키면 안됨



```
명령 프롬프트
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SAMSUNG>d:

D:\>jupyter notebook_
```

■ 사용방법 요약

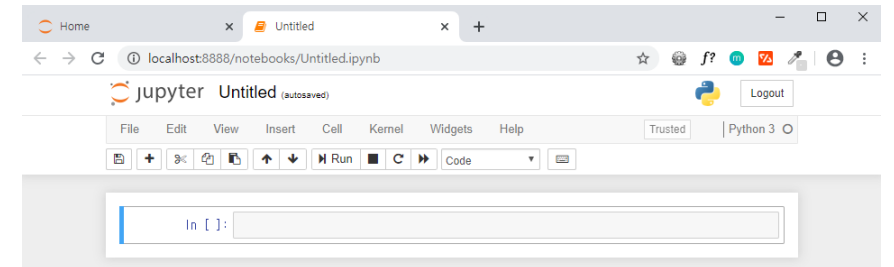
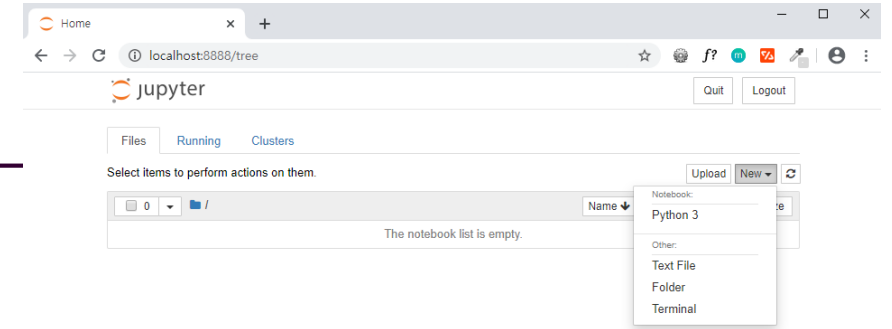
■ 노트북 만들기

- 오른쪽 New 버튼을 클릭한 뒤 Python 3을 클릭
- 노트북이름을 클릭하여, 노트북의 이름을 변경

■ Cell에 Code 또는 Markdown 입력

■ 유용한 단축키

- 편집모드 : 녹색(cell 클릭 / Enter)
 - Ctrl+Enter: 현재 cell 실행
 - Shift+Enter: 현재 cell 실행하고, 다음 cell로 이동(없으면 새로 만듦)
- 명령모드 : 청색(cell 바깥쪽 클릭 / ESC)
 - m: markdown, y: code
 - a: 현재 cell 위에 추가, b: 현재 cell 아래에 추가
 - dd: 현재 cell 삭제, z: 현재 cell 삭제 취소



참고: Jupyter notebook extentions

■ 설치

- `pip install jupyter_contrib_nbextensions && jupyter contrib nbextension install`

■ jupyter notebook에 Nbextensions 탭 생성됨

■ 유용한 확장기능

- ExecuteTime
- Hinterland
- Variable Inspector

참고: Google Colab

■ colab 사용하기

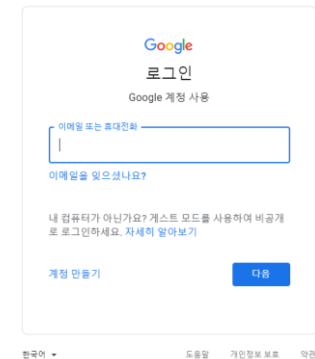
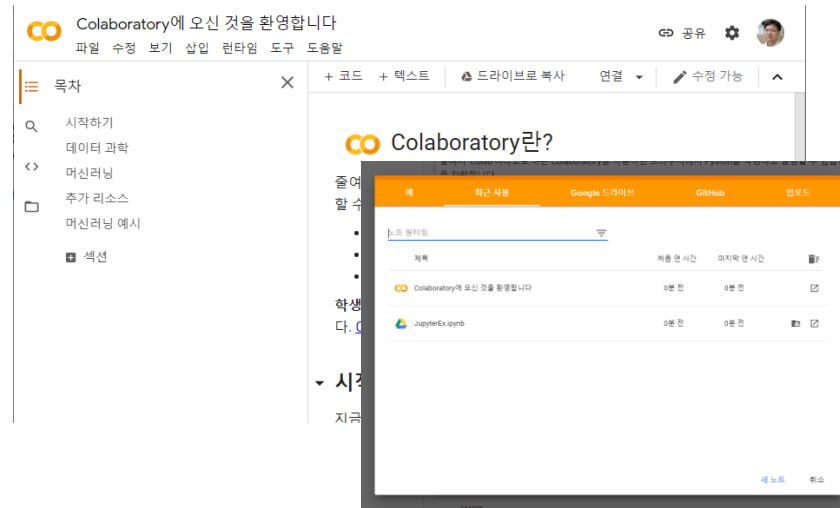
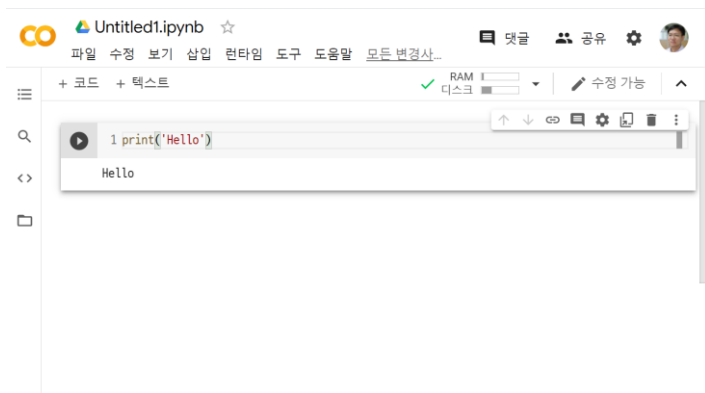
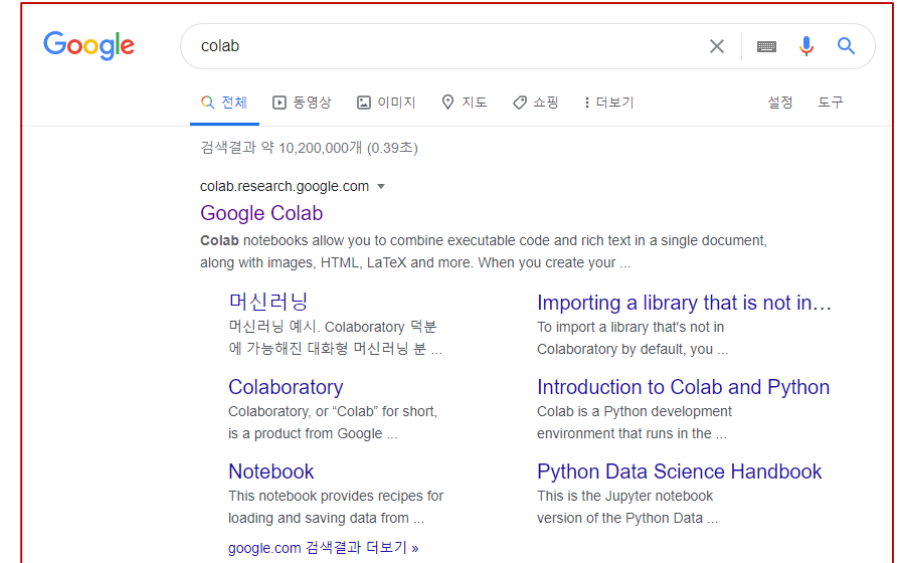
■ colab 사이트에 접속

- <https://colab.research.google.com/>

■ google 로그인

■ 새노트

- 새노트 만들기
- coding 후, 실행(Ctrl+Enter)



Colab 사용시 꼭 알아야 할 것들

■ 경고안내문 무시하기

```
import warnings

warnings.filterwarnings("ignore")
```

■ 한글폰트 설치하기

```
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

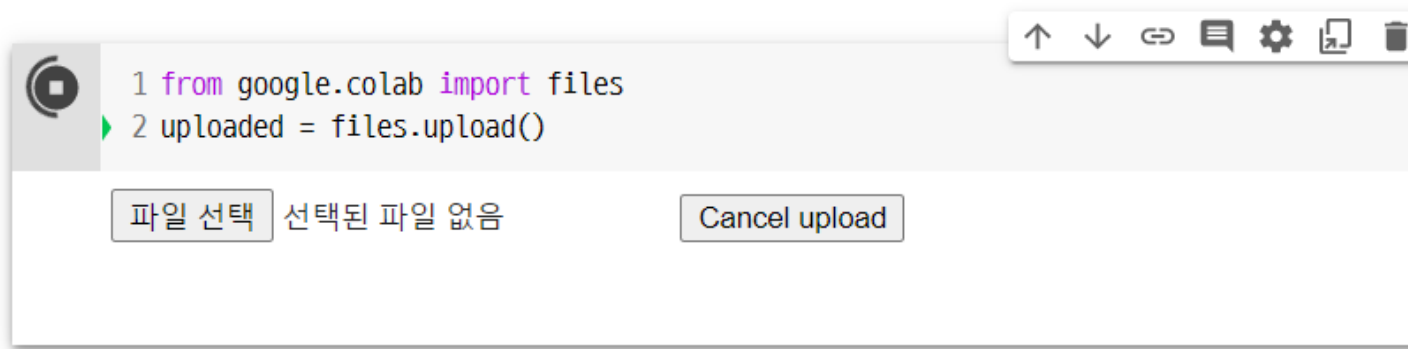
■ matplotlib에서 한글폰트 사용하기

```
import matplotlib.pyplot as plt

plt.rc('font', family='NanumBarunGothic')
```

■ Google drive로 로컬파일 업로드하기

```
from google.colab import files  
uploaded = files.upload()
```



■ Google drive로 부터 csv파일 불러오기

```
import io  
import pandas as pd  
  
data = pd.read_csv(io.BytesIO(uploaded['tips.csv']))  
data
```


특징공간과 데이터

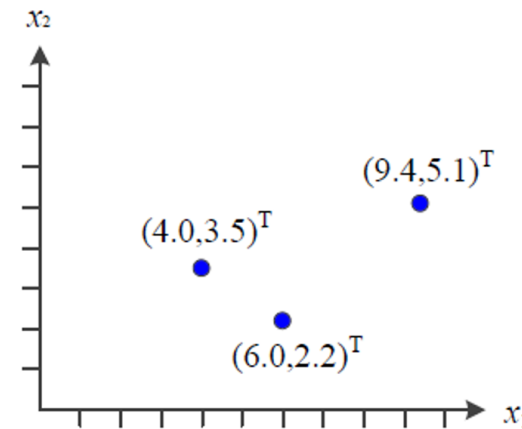
특징공간(feature space)

■ 1차원 특징공간

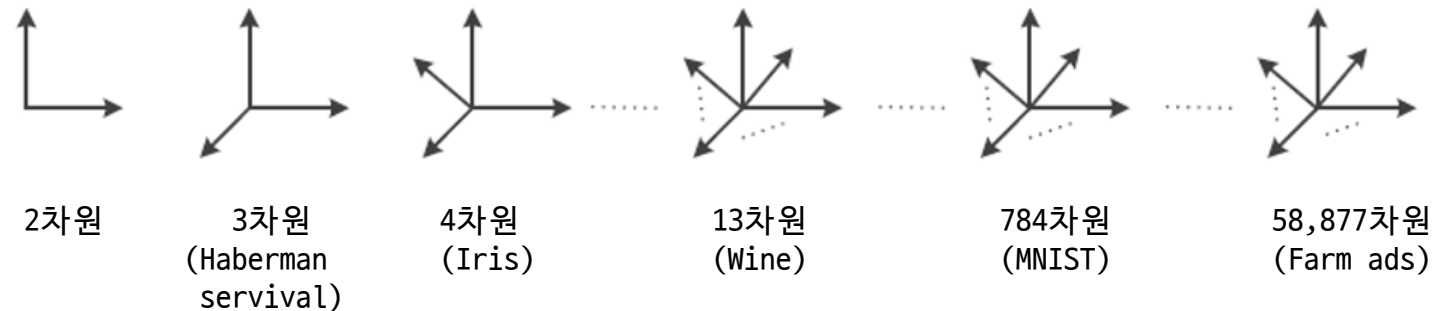
- 특징이 1개인 공간(특징값 x , 목표값 y)
- n 개의 data 각각이 1개의 특징(x)만 있는 경우

■ 2차원 특징공간

- 특징이 2개인 공간(vector)
- $\mathbf{x} = (x_1, x_2)^T$
- (예)
 - $\mathbf{x} = (\text{몸무게}, \text{키})^T, y = \text{비만지수}$
 - $\mathbf{x} = (\text{체온}, \text{두통})^T, y = \text{감기여부}$



■ 다차원 특징공간의 예



■ Haberman survival : 하버만생존 데이터셋

- $\mathbf{x} = (\text{나이}, \text{수술년도}, \text{양성림프샘개수})^T$
- $y = \{1 \mid 2\}$ # 1은 5년이상 생존, 2는 5년이내 사망

■ Iris : 붓꽃 데이터셋

- $\mathbf{x} = (\text{꽃받침길이}, \text{꽃받침너비}, \text{꽃잎길이}, \text{꽃잎너비})^T$
- $y = \{\text{setosa} \mid \text{versicolor} \mid \text{virginica}\}$ #붓꽃 종

■ Wine : 와인등급 데이터셋

■ MNIST : 필기체숫자 데이터셋

■ Farm ads : 농업의사결정시스템 데이터셋

Harberman servivial Dataset

- 데이터 개수 : 306개

- 데이터 정보

- 데이터 세트에는 유방암 수술을받은 환자의 생존에 대해 시카고 대학 빌링스 병원에서 1958 년에서 1970년 사이에 수행 된 연구 사례가 포함되어 있음

- 속성 정보

1. Age of patient at time of operation (numerical)
2. Patient's year of operation (year - 1900, numerical)
3. Number of positive axillary nodes detected (numerical)
4. Survival status (class attribute)
 - 1 = the patient survived 5 years or longer
 - 2 = the patient died within 5 year

Iris Dataset

- 데이터개수 : 150개
- 데이터 정보
 - 데이터 세트에는 각각 50 개 인스턴스의 3 개 클래스가 포함되어 있으며, 각 클래스는 붓꽃의 유형을 나타냄
- 데이터 속성
 1. sepal length in cm
 2. sepal width in cm
 3. petal length in cm
 4. petal width in cm
 5. class:
 - Iris Setosa
 - Iris Versicolour
 - Iris Virginica



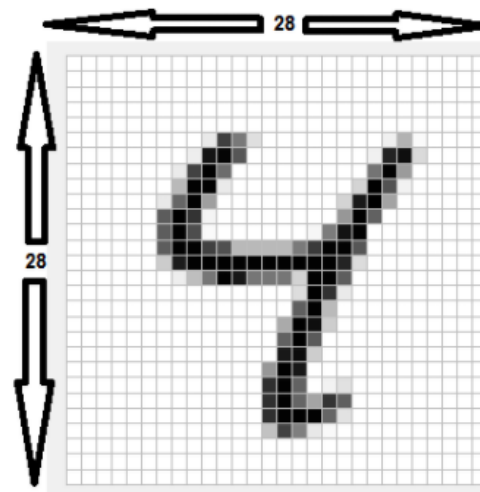
MNIST Dataset

■ 우편번호 필기숫자 데이터베이스

- 학습데이터 60,000개 / 테스트데이터 10,000개

- 4개의 파일로 구성

- train-images-idx3-ubyte.gz # training images
- train-labels-idx1-ubyte.gz # training set labels
- t10k-images-idx3-ubyte.gz # test images
- t10k-labels-idx1-ubyte.gz # test set labels



다차원 특징공간과 매개변수의 수

- d-차원 데이터

- 특징벡터 표현 $\mathbf{x} = (x_1, x_2, \dots, x_d)^T$

- d-차원 데이터의 선형학습모델

- 1차원인 경우

- $y = ax + b$

- 다차원인 경우

- $y = \mathbf{w}\mathbf{x} + b$

- $= w_1x_1 + w_2x_2 + \dots + w_dx_d + b$

- 매개변수의 개수 : $d + 1$

■ d-차원 데이터의 2차곡선형학습모델

■ 1차원인 경우

- $y = w_1 x_1^2 + w_2 x_1 + b$

■ 다차원인 경우

- 생략

■ 매개변수의 개수 : $d^2 + d + 1$

- (예)

- Iris데이터 : $d = 4$

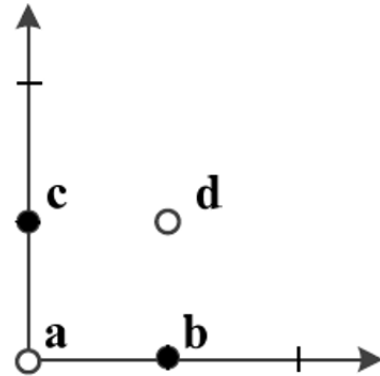
- 매개변수의 수 : $4^2 + 4 + 1 = 21$

- MNIST데이터 : $d = 784$

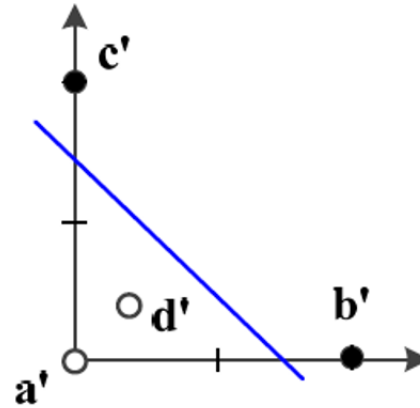
- 매개변수의 수 : $(784)^2 + 784 + 1 = 615,441$

특징공간 변환과 표현학습

- 선형분리 불가능한 특징공간 변환의 예
 - 선형모델은 최대 75%가 한계



주어진 특징공간



분리에 유리하게 변환된 특징공간

■ $\mathbf{x} = (x_1, x_2)^T$

- $a = (0, 0)$ →
- $b = (1, 0)$ →
- $c = (0, 1)$ →
- $d = (1, 1)$ →

$$\mathbf{x}' = \left(\frac{x_1}{2x_1x_2+0.5}, \frac{x_2}{2x_1x_2+0.5} \right)^T$$

- $a' = (0, 0)$
- $b' = (2, 0)$
- $c' = (0, 2)$
- $d' = (0.4, 0.4)$

■ 표현학습(representation learning)

- 좋은 특징공간을 자동으로 찾아내는 것
- 딥러닝은 다수의 은닉층을 가진 신경망을 이용하여 계층적인 특징 공간을 찾아냄
 - 왼쪽 은닉층은 저급 특징(에지, 구석점 등), 오른쪽은 고급 특징을 추출

■ 표현학습의 단적인 예(인위적)

- $\mathbf{x} = (x_1, x_2)^T$ $\mathbf{x}' = \left(\frac{x_1}{2x_1x_2+0.5}, \frac{x_2}{2x_1x_2+0.5} \right)^T$

■ 차원의 저주

- 차원이 높아짐에 따라 발생하는 현실적인 문제

- (예) $d=4$ 인 Iris 데이터

- 축마다 100개 구간으로 나누면 총 $100^4=1$ 억 개의 칸에 150개의 샘플이 있는 희소분포

- (예) $d=784$ 인 MNIST 샘플

- 화소가 0과 1값을 가진다면 $2^{784} = 1.017458 \times 10^{236}$ 개의 칸
 - 거대한 공간에 고작 60,000개의 샘플을 흩뿌린 매우 희소한 분포를 가짐

데이터의 생성과 중요성

■ 데이터의 생성과정을 아는 경우

- 특징 : x = 주사위 2개를 던져 나온 눈의 합
- 목표값 : $y = (x - 7)^2 + 1$
- 주사위 실험으로 데이터 생성
 - $x = \{3, 10, 8, 5\}$, $y = \{17, 10, 2, 5\}$
 - 이런 상황을 ‘데이터 생성 과정을 완전히 알고 있다’고 말함
 - x 를 알면 정확히 y 를 예측할 수 있음
 - $P(x)$ 를 알고 있으므로 새로운 데이터 생성 가능

■ 기계학습 문제

- 데이터의 생성과정을 알 수 없음
- 주어진 학습데이터로 예측모델(생성모델)을 역으로 추정하는 문제
=> 데이터의 품질이 매우 중요

■ 데이터베이스의 품질

- 주어진 응용에 맞는 충분히 다양한 데이터를 충분한 양만큼 수집하면 추정 정확도 높아짐
- (예) 정면 얼굴만 가진 데이터베이스로 학습하면, 기울어진 얼굴의 분류성능은 매우 낮음
- 주어진 응용 환경을 자세히 살핀 다음 그에 맞는 데이터베이스 확보는 아주 중요함

■ 공개 데이터베이스

- UCI Repository(2017년11월 기준으로 394개 데이터베이스 제공)
- Iris, MNIST, ImageNet 등

데이터베이스 크기와 학습성능

■ 데이터베이스의 왜소한 크기

- (예) MNIST: 28×28 흑백 비트맵이라면 서로 다른 총 샘플 수는 2^{784} 가지이지만, MNIST는 고작 60,000개 샘플

■ 그런데 왜소한 데이터베이스로 어떻게 높은 성능(현재까지 99.8%)을 달성하는가?

- 방대한 공간에서 실제 데이터가 발생하는 곳은 매우 작은 부분 공간임
- 발생하는 데이터가 매우 심한 변화를 겪지만, 일정한 규칙에 따라 매끄럽게 변화함(manifold 가정)



데이터의 시각화

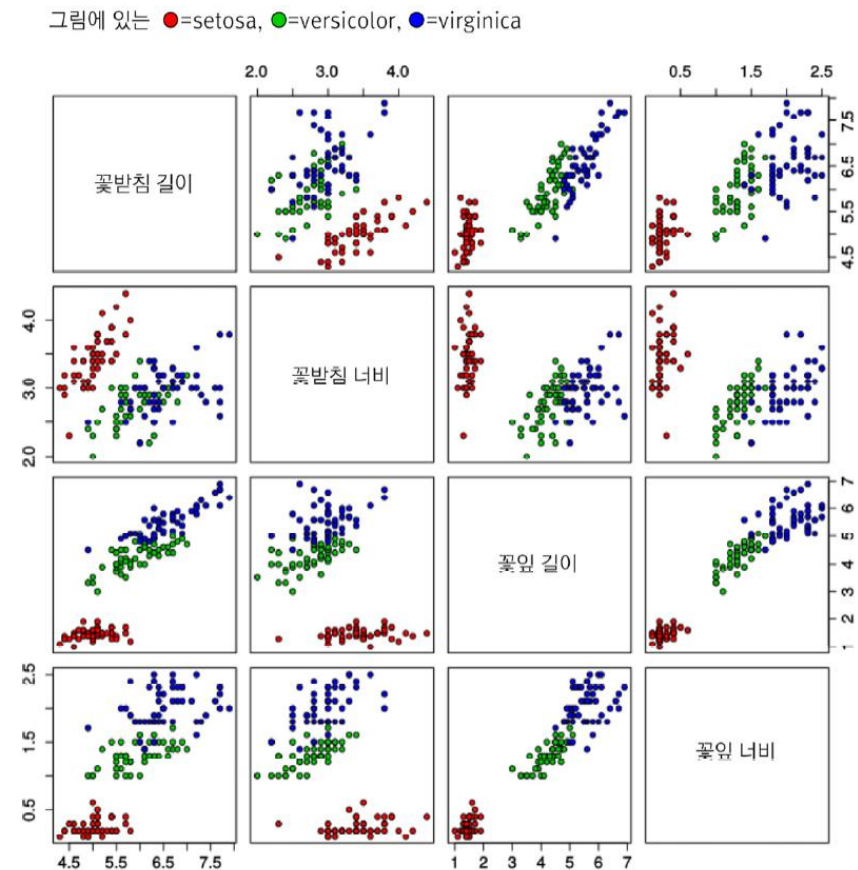
■ 4차원 이상의 초공간은 한꺼번에 가시화 불가능

■ 여러 가지 가시화 기법

- 2개씩 조합하여
여러개의 그래프로
나타냄

- PCA(주성분분석)

- 차원의 축소

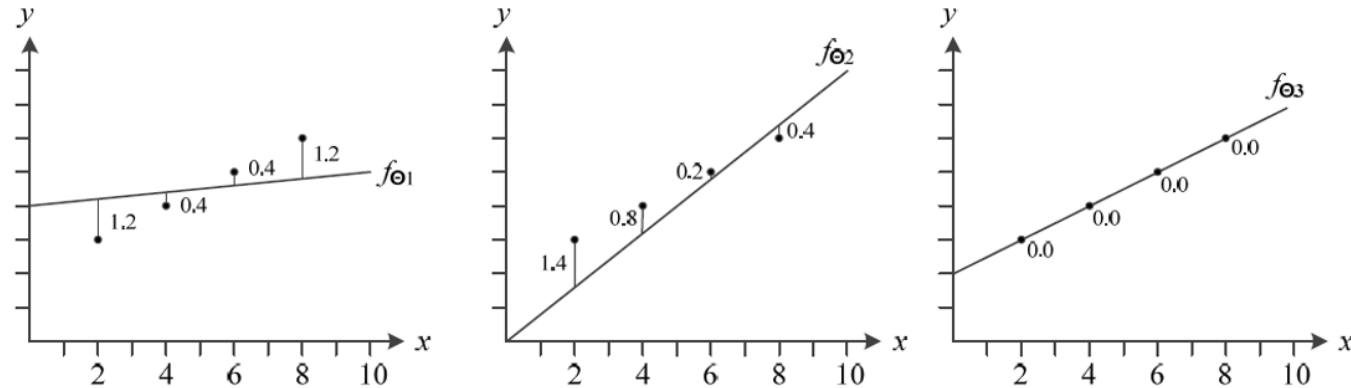


모델선택

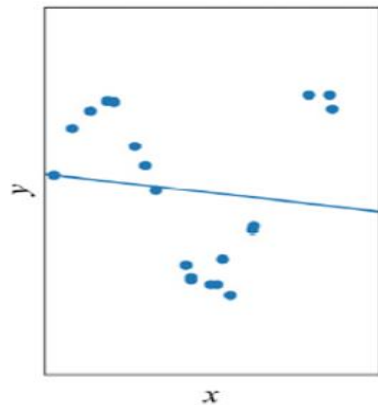
- 과소적합과 과대적합
- 바이어스와 분산
- 검증집합과 교차검증
- 규제
- 기계학습의 과거와 현재, 미래

기계학습 모델

- 선형회귀의 경우 학습모델 : $y = wx + b$



- 잘못된 모델선택의 예

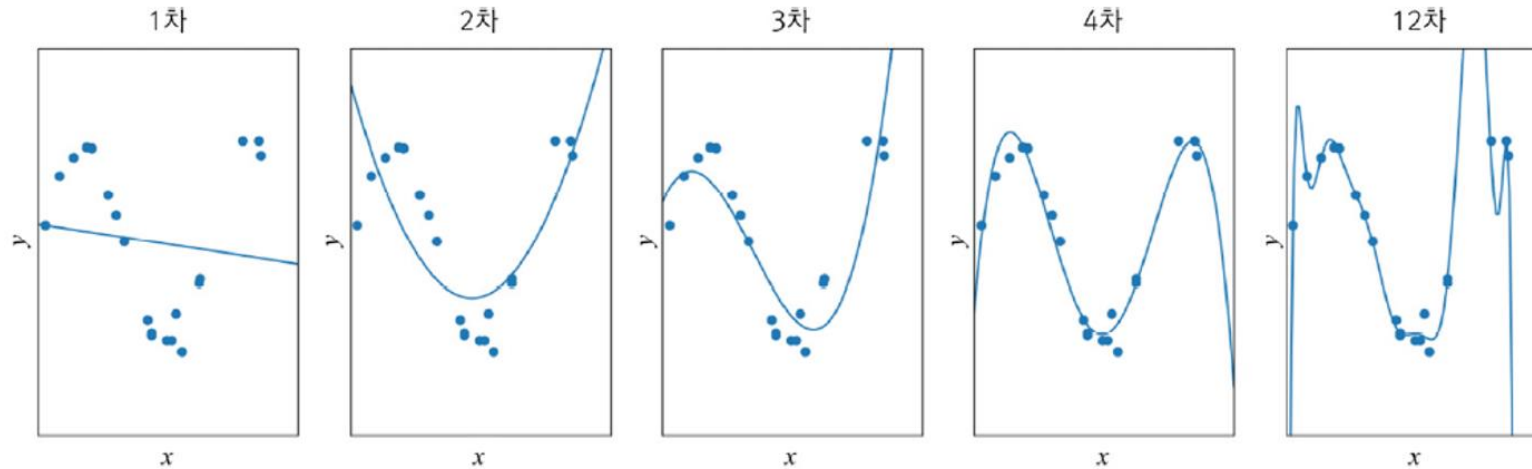


← 과소적합

모델이 필요

과소적합과 과대적합

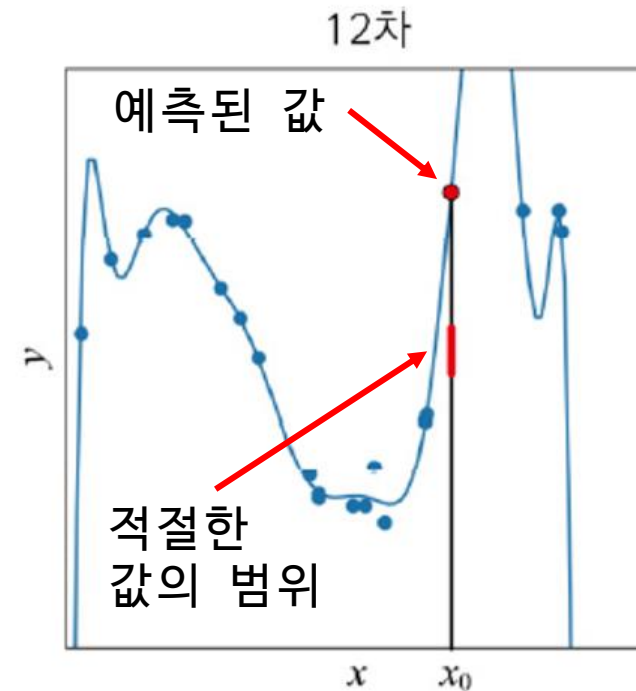
■ 비선형 다항식 곡선모델의 사용



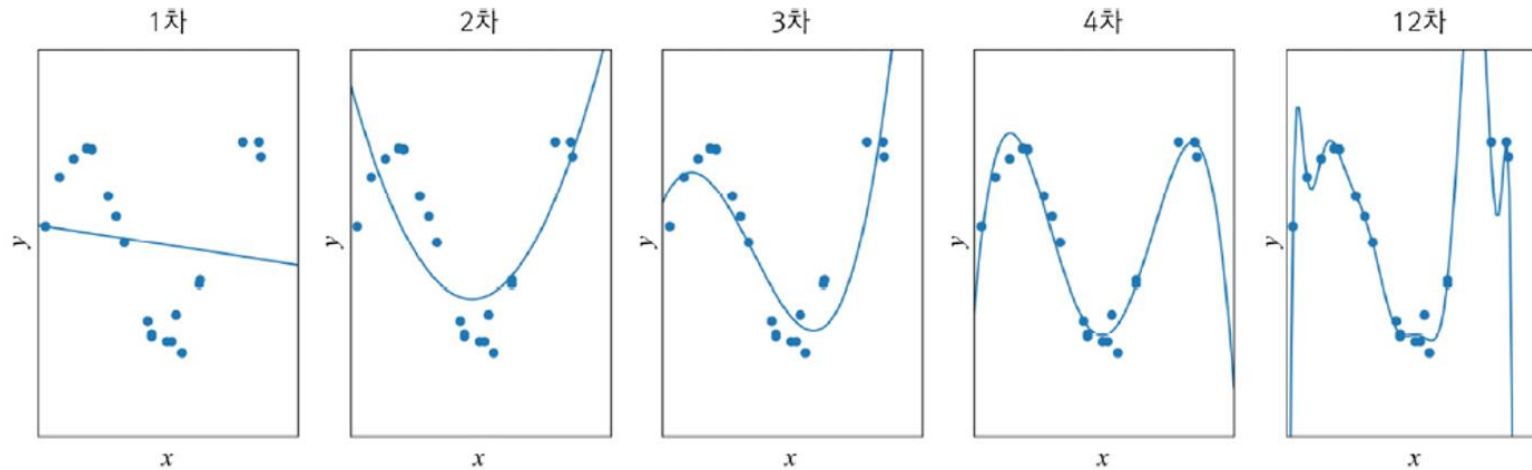
- 선형모델(1차)에 비해 오차가 크게 감소됨
- 12차 다항식모델이 가장 오차가 작음
 - 과연 이렇게 하는 것이 좋을까?

■ 과대적합

- 12차 다항식모델은 학습데이터에 대해서는 거의 완벽하게 근사됨
- 그러나 학습데이터에는 없는 새로운 데이터에 대해 예측한다면?
 - 잘못된 예측
 - 왜?
 - 학습에서 쓸데없는 학습데이터인 잡음까지 수용됨
- 데이터검증은 거의 불가능
- 적절한 모델의 선택이 매우 중요



■ 그렇다면 가장 적절한 모델은?



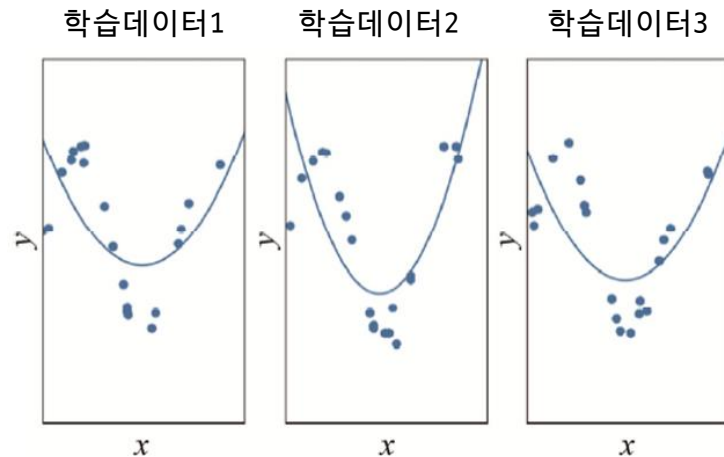
- 3~4차 다항식 모델이 가장 적절함
- 적절한 모델인지 어떻게 판단하나?
 - 바이어스(bias : 차이)와 분산(variance)을 고려하면 됨

바이어스와 분산

■ 학습데이터를 여러 개 사용하여 실험

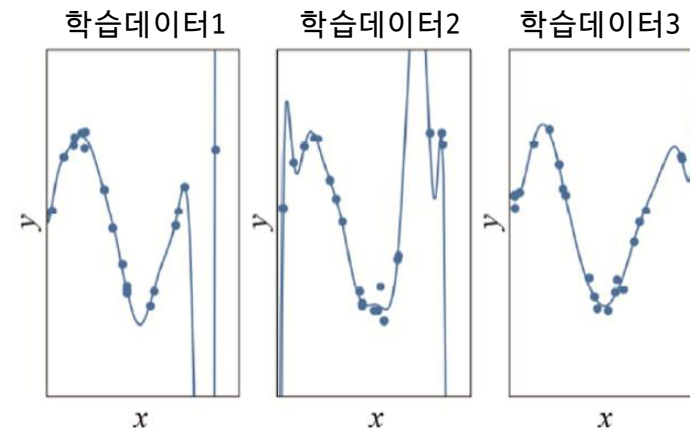
■ 2차 다항식 모델

- 매번 큰 오차(큰 바이어스). 매번 비슷한 모델(낮은 분산)



■ 12차 다항식 모델

- 매번 작은 오차(작은 바이어스). 매번 크게 다른 모델(높은 분산)



■ 일반적인 경향

■ 용량이 작은 모델

- 바이어스는 크고 분산은 작음

■ 복잡한 모델

- 바이어스는 작고 분산은 큼

■ 바이어스와 분산은 trade-off 관계

■ 일반화 능력

- 결국, 가능한한 작은 바이어스(small bias)와 낮은 분산(low variance)을 가지도록 하는 모델을 선택하는 것이 필요
- 테스트 데이터에 대해 높은 성능을 보이도록...

검증데이터와 교차검증

- 좋은 모델을 알고 있는 경우
 - 학습데이터로 모델을 학습하고, 테스트데이터로 일반화능력을 측정
- 좋은 모델을 모르는 경우
 - 여러 모델들을 학습시킨 후, 그 중에서 가장 좋은 모델을 선택
 - 검증데이터셋(validation dataset)
 - 모델을 비교할 때 사용하는 데이터

1. for 모델 in 모델집합
 - 모델을 학습데이터로 학습시킨다.
 - 검증데이터로 학습된 모델의 성능을 측정한다.
2. 가장 높은 성능을 보인 모델을 선택한다.
3. 테스트데이터로 모델의 일반화능력(성능)을 측정한다.

교차검증(cross-validation)

■ 기계학습을 위한 데이터의 종류

- 학습데이터, 검증데이터, 테스트데이터

- 데이터의 수집은 비용이 많이 듦

- 일반적으로 데이터의 양이 부족함. 따라서 검증데이터를 마련하기 어려움
- 해결방법 : k-교차검증

■ 교차검증에 의한 모델선택

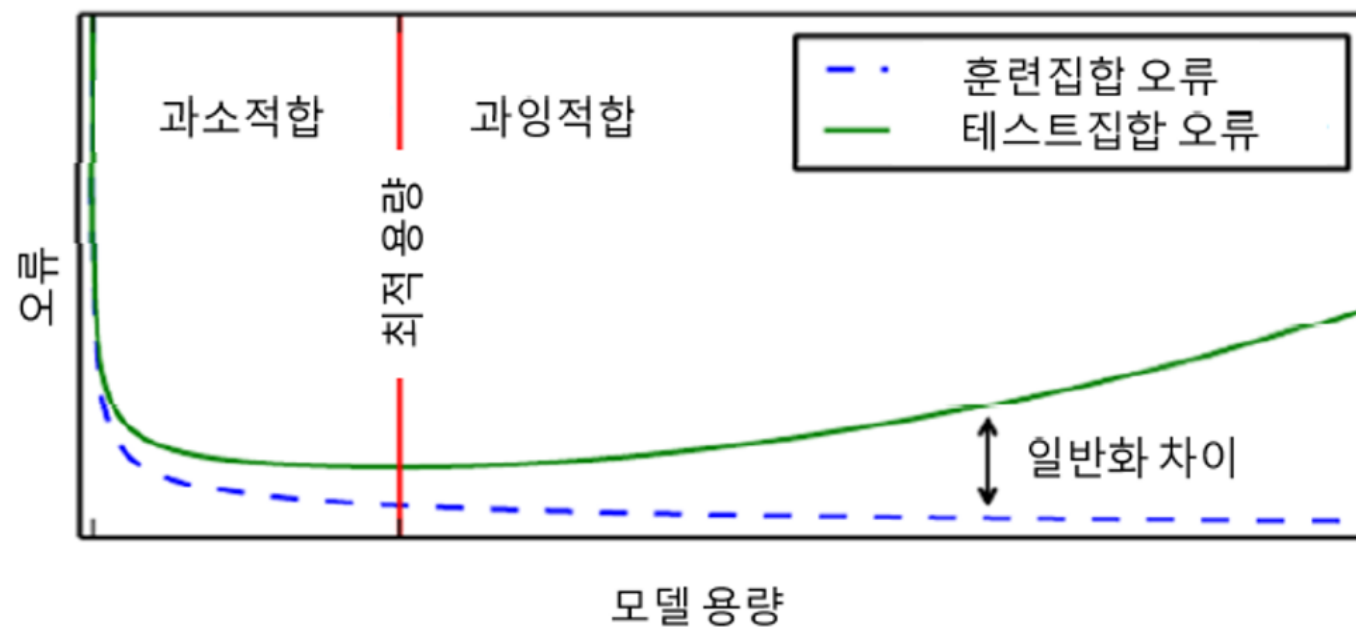
1. 학습데이터를 k개 그룹으로 분할한다.
2. for 모델 in 모델집합
 - 2.1 for i in range(k)
 - 모델을 학습데이터(i그룹제외)로 학습시킨다.
 - 학습된 모델의 성능을 i그룹 데이터로 측정한다.
 - 2.2 k개 성능을 평균하여 모델의 성능으로 정한다.
3. 가장 높은 성능을 보인 모델을 선택한다.
4. 테스트데이터로 모델의 일반화능력(성능)을 측정한다.

모델집합의 구성

- 실제로는 고차원 데이터에 대해 다항식 모델을 사용하지 않음
 - 신경망, 강화학습, SVM, 트리분류기 등을 사용
- 현실적인 학습모델의 선택
 - 경험과 지식을 통해 모델을 선택
 - (예) 신경망 MPL를 사용
 - 선택된 모델의 세부 사항을 결정
 - (예) 신경망 MLP의 은닉층 개수, 노드의 개수, 활성화함수, 비용함수, learning rate 등
 - 규제(regularization)
 - 충분히 크고 깊은 모델을 선택한 후, 각종 규제를 적용하는 방법을 사용

규제(regularization)

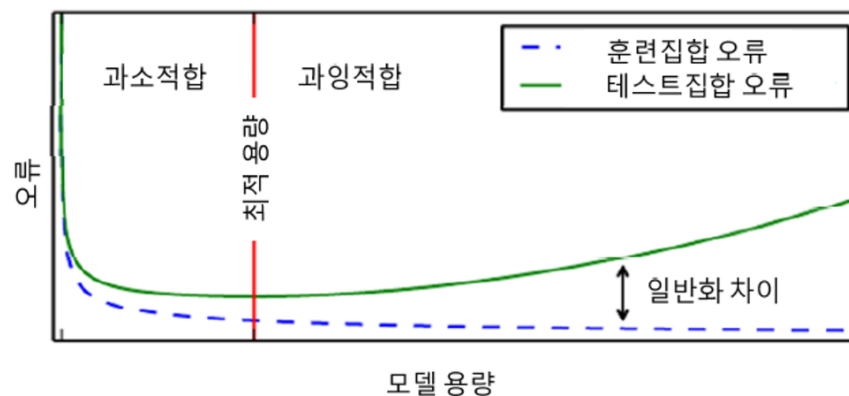
- 기계학습이 높은 일반화 능력을 확보하는 기본적인 접근방법
 - 충분히 큰 모델을 선택 -> 각종 규제를 적용



규제의 필요성

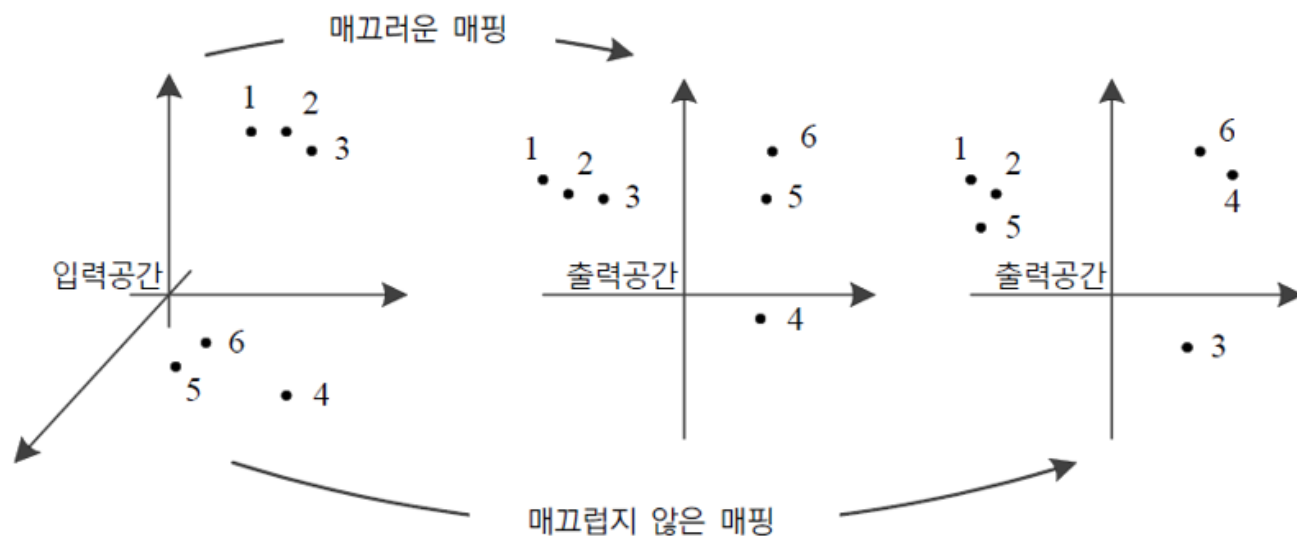
■ 과잉적합(과대적합)에 빠지는 이유

- 모델이 크면 훈련집합(학습데이터)의 오류는 0에 가까워지지만, 테스트집합에 대해서는 오히려 성능이 악화됨. => 하지만 대부분의 모델 설계자는 작은 용량의 모델을 사용하지 않으려고 함
- 학습알고리즘이 매개변수값을 조정하는 일을 하다가 결국 학습데이터를 **단순 암기(?)하는 상태**가 될 수 있음

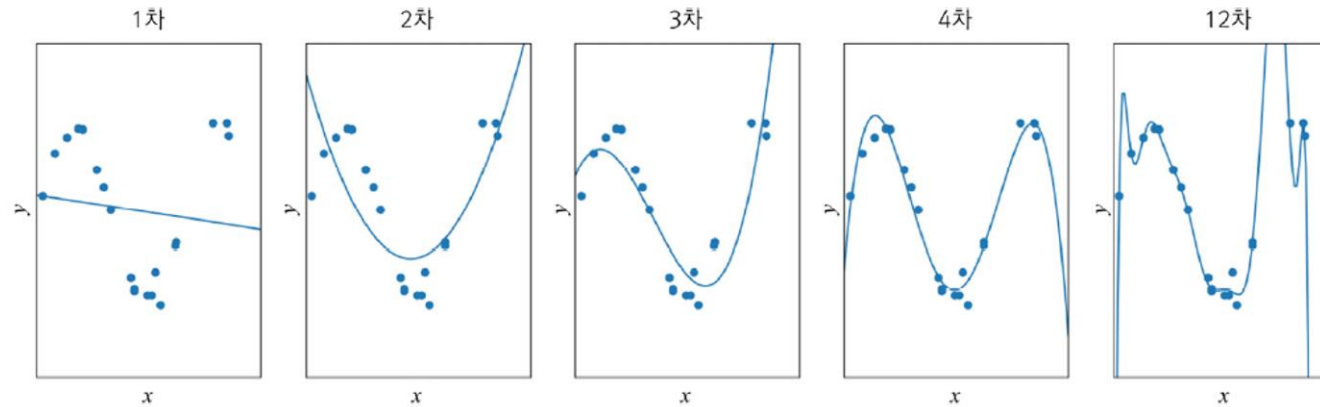


규제의 정의

- 오래 전부터 수학과 통계학에서 연구해온 주제
 - 모델 용량에 비해 데이터가 부족한 경우의 불량 문제(ill-posed problem)를 푸는 데 사용
 - 데이터의 대부분은 매끄러운 성질을 만족한다는 사전지식(prior knowledge)를 이용



■ (예) 4차는 매끄러우나, 12차는 매끄럽지 못함



■ 티호노프(Tikhonov)의 규제기법 - 수치적 용량 제한

■ $J_{reg}(\Theta) = J(\Theta) + \lambda R(\Theta)$

■ $J(\Theta)$: 목적함수

■ $R(\Theta)$: 규제항(매끄러운 정도) - 벌칙

■ 매끄럽지 못하면 큰 값을 부여하여 벌칙을 강화

■ 즉, 가중치가 커지면 벌칙을 부가하여 모델 용량이 폭발적으로 커지지 않도록 방지

규제의 종류

■ 명시적 규제

■ 가중치 감쇠(weight decay)

- 가중치를 강제로 축소

■ 드롭아웃(dropout)

- 노드의 일정비율을 제거

■ 암시적 규제

■ 조기 멈춤(early stopping)

- 검증데이터의 오류가 최저인 지점을 만나면 멈춤

■ 데이터 확대

- 충분히 큰 데이터셋을 사용하기 위해, affine변환 등으로 데이터를 강제로 늘림

■ 앙상블(ensemble)

- 여러 개의 모델이 예측한 결과를 결합

가중치 벌칙

■ 가중치에 규제를 가하여 모델의 수치적 용량을 제한하는 기법

- $J_{regularized}(\Theta; X, Y) = J(\Theta; X, Y) + \lambda R(\Theta)$

- 규제가 적용된 함수 = 목적함수 + 규제항
- 규제항은 목적함수의 값이 커지면 벌칙을 부가해서 작은 가중치를 유지하도록 함

- 규제항으로는 L2 norm을 많이 사용

■ L2 norm의 적용

- $J_{regularized}(\Theta; X, Y) = J(\Theta; X, Y) + \lambda \|\Theta\|_2^2$

- 즉, 학습 시 비용함수에 가중치의 제곱(λ 계수에가중치의 제곱을 곱한)을 더함

■ norm

- 벡터의 크기를 측정하는 방법 (두 벡터 사이의 거리를 측정)

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

- p : norm의 차수. 1이면 L1, 2이면 L2
- n : 벡터의 요소 수

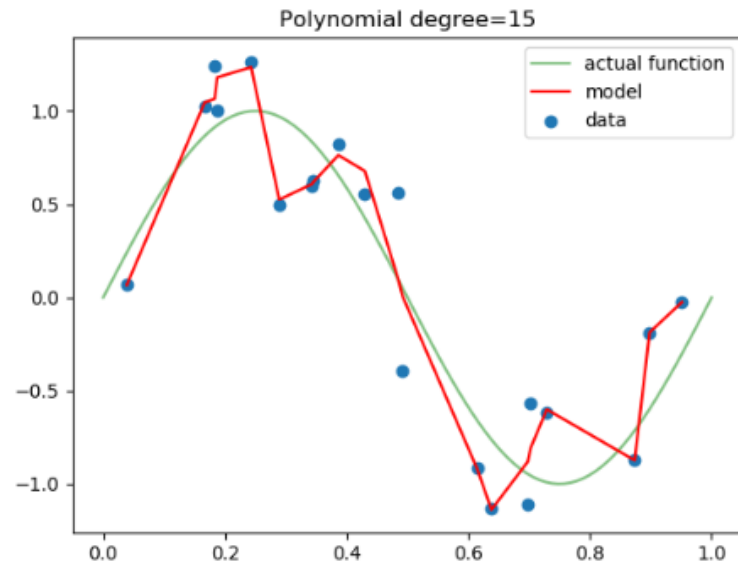
■ L1 norm

L2 norm

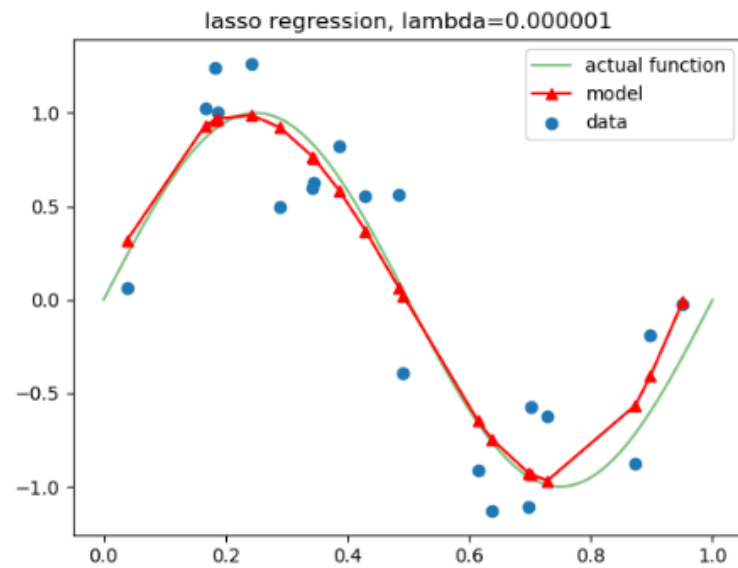
$$\|p - q\|_1 = \sum_i^n |p_i - q_i|$$

$$\|p - q\|_2 = \left(\sum_i^n |p_i - q_i|^2 \right)^{1/2}$$

벡터 p와 q사이의 직선거리



특정 가중치가 너무 큰 값을 가지면
model의 일반화능력이 떨어짐



규제항(가중치의 제곱에 λ 를 곱한 값)을
더해줌으로써 과적합을 방지

- 가중치 벌칙을 학습알고리즘에 적용하기 위해, 기울기(gradient)를 계산

- $J_{regularized}(\Theta; X, Y) = J(\Theta; X, Y) + \lambda \|\Theta\|_2^2$ 을 미분하면,
 $\nabla J_{regularized}(\Theta; X, Y) = \nabla J(\Theta; X, Y) + 2\lambda\Theta$

- 매개변수의 갱신

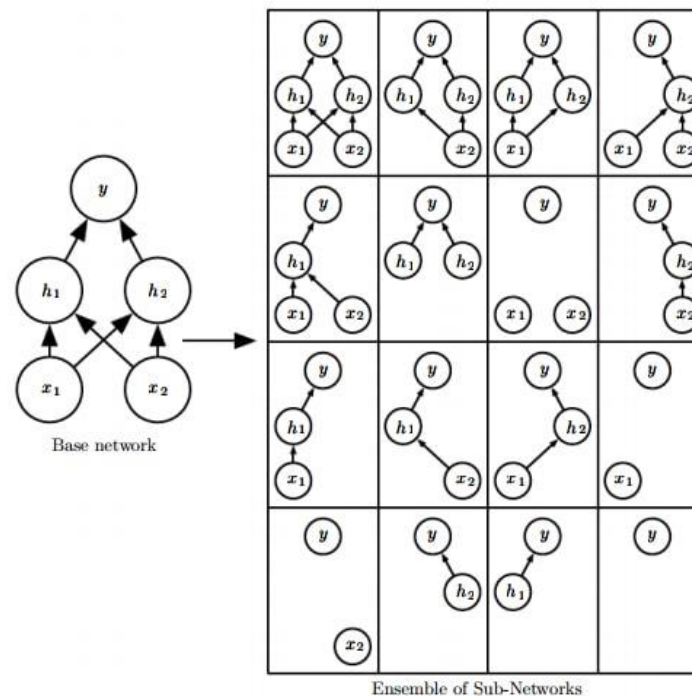
$$\begin{aligned}\Theta &= \Theta - \alpha \nabla J_{regularized}(\Theta; X, Y) \\ &= \Theta - \alpha (\nabla J(\Theta; X, Y) + 2\lambda\Theta) \\ &= (1 - 2\alpha\lambda)\Theta - \alpha \nabla J(\Theta; X, Y)\end{aligned}$$

- α : learning rate

- α 는 1보다 훨씬 작은 수를 사용하므로 $2\alpha\lambda$ 는 1보다 작음. 즉, 가중치를 작게 감쇠시키는 효과가 있음

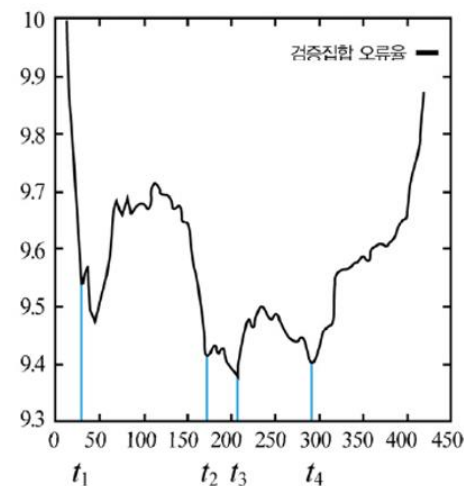
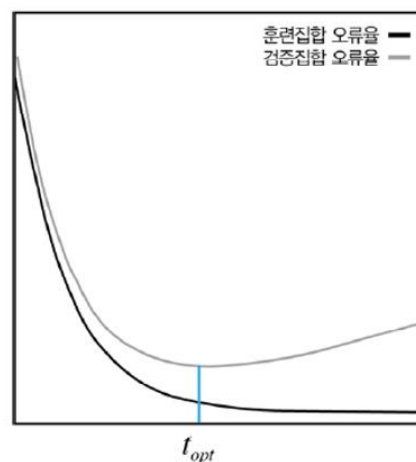
드롭아웃

- 입력층과 은닉층 노드 중 일정비율을 임의로 선택하여 제거하는 기법
 - 남아있는 부분신경망으로만 학습을 진행
 - 많은 부분신경망을 만들고, 테스트시에 앙상블 결합하는 효과가 있음
 - 실제로는 1개의 신경망에 적용
 - 효과
 - voting
 - 여러 부분 신경망들의 평균
 - co-adaptation 방지
 - 특정 데이터나 뉴런의 영향을 최소화



조기 멈춤

- 일정 시간이 지나면 과대적합이 발생
 - 학습데이터에 대해서는 오류가 거의 없지만, 테스트데이터에 대해서는 오류가 커짐
- 조기멈춤
 - 검증데이터의 오류가 최저인 지점에서 학습을 중단
 - 지그재그 현상을 고려할 필요가 있음

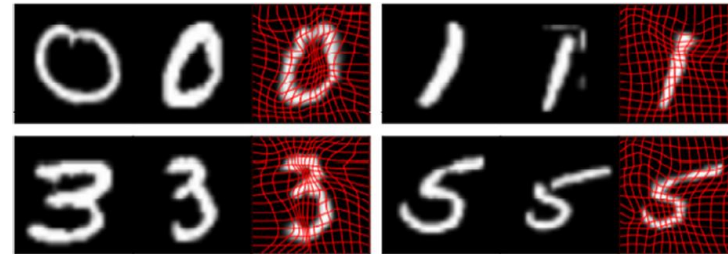


데이터 확대

- 데이터를 인위적으로 변형하여 확대

- (예)

- Affine 변환(이동, 회전, 크기) 적용
- Morphing
- 자르기, 이동, 반전
- PCA를 이용한 색상변환
- Noise 추가



앙상블 기법

■ 앙상블(ensemble)

- 서로 다른 여러 개의 모델들을 결합하여 일반화 오류를 줄이는 기법

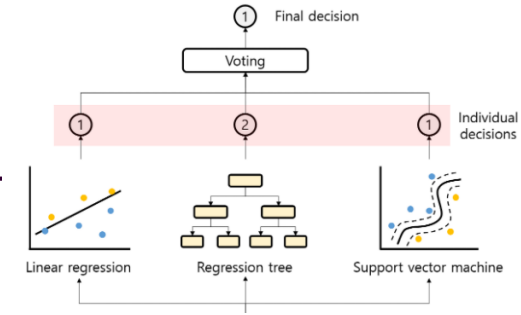
■ 종류

■ 서로 다른 예측기를 학습

- (예) 서로 다른 구조의 신경망 여러 개를 학습 또는 같은 구조를 사용하되 서로 다른 초기값과 매개변수를 설정하고 학습
- (예) Bagging(학습데이터를 여러 번 샘플링하여 k개의 서로 다른 집합을 구성하고 k개의 신경망 학습. 예측시에는 voting)
- (예) Boosting(i번째 예측기가 틀린 샘플을 i+1번째 예측기가 잘 인식하도록 연계성을 고려)- AdaBoost 알고리즘 참고

■ 학습된 예측기들을 결합

- 주로 투표(voting) 방식을 사용



기계학습의 과거, 현재, 미래

- 1843 에이다 “... 해석엔진은 꽤 복잡한 곡을 작곡할 수도 있다.”라는 논문 발표[Ada1843]
- 1950 인공지능 여부를 판별하는 튜링 테스트[Turing1950]
- 1956 최초의 인공지능 학술대회인 다트머스 컨퍼런스 개최. ‘인공지능’ 용어 탄생[McCarthy1955]
- 1958 로젠블랫이 퍼셉트론 제안[Rosenblatt1958]
 인공지능 언어 Lisp 탄생
- 1959 사무엘이 기계 학습을 이용한 체커 게임 프로그램 개발[Samuel1959]
- 1969 민스키가 퍼셉트론의 과대포장 지적. 신경망 내리막길 시작[Minsky1969]
 제1회 IJCA[International Joint Conference on Artificial Intelligence] 개최
- 1972 인공지능 언어 Prolog 탄생
- 1973 Lighthill 보고서로 인해 인공지능 내리막길, 인공지능 겨울AI winter 시작
- 1974 웨어보스가 오류 역전파 알고리즘을 기계 학습에 도입[Werbos1974]
- 1975경 의료진단 전문가 시스템 Mycin – 인공지능에 대한 관심 부활
- 1979 『IEEE Transactions on Pattern Analysis and Machine Intelligence』 저널 발간
- 1980 제1회 ICML[International Conference on Machine Learning] 개최
 후쿠시마가 NeoCognitron 제안[Fukushima1980]
- 1986 『Machine Learning』 저널 발간
 『Parallel Distributed Processing』 출간
 다층 퍼셉트론으로 신경망 부활

-
- 1987 Lisp 머신의 시장 붕괴로 제2의 인공지능 겨울
 UCI 리포지토리 서비스 시작
 NIPSNeural Information Processing Systems 컨퍼런스 시작
- 1989 「Neural Computation」 저널 발간
- 1993 R 언어 탄생
- 1997 IBM 딥블루가 세계 체스 챔피언인 카스파로프 이김
 LSTMLong short-term memory 개발됨
- 1998경 SVM이 MNIST 인식 성능에서 신경망 추월
- 1998 르쿤이 CNN의 실용적인 학습 알고리즘 제안[LeCun1998]
 『Neural Networks: Tricks of the Trade』 출간
- 1999 NVIDIA 사에서 GPU 공개
- 2000 「Journal of Machine Learning Research」 저널 발간
 OpenCV 최초 공개
- 2004 제1회 그랜드 챌린지(자율 주행)
- 2006 층별학습 탄생[Hinton2006a]
- 2007경 딥러닝이 MNIST 인식 성능에서 SVM 추월
- 2007 GPU 프로그래밍 라이브러리인 CUDA 공개

	어번 챌린지(도심 자율 주행)
	Scikit-learn 라이브러리 최초 공개
2009	Theano 서비스 시작
2010	ImageNet 탄생
	제1회 ILSVRC 대회
2011	IBM 왓슨이 제퍼디 우승자 꺾음
2012	MNIST에 대해 0.23% 오류율 달성
	AlexNet 발표 (3회 ILSVRC 우승)
2013	제1회 ICLR International Conference on Learning Representations 개최
2014	Caffe 서비스 시작
2015	TensorFlow 서비스 시작
	OpenAI 창립
2016	알파고와 이세돌의 바둑 대회에서 알파고 승리[Silver2016]
	『Deep Learning』 출간
2017	알파고 제로[Silver2017]