

기계학습

머신러닝..실전입문 Ch4
조남운

목차

- 머신러닝에 대하여
- 머신러닝 프레임워크 scikit-learn
- 머신러닝 실습1: 이미지 내부 문자인식
- 머신러닝 실습2: 외국어 문장 판별
- Support Vector Machine (SVM)
- Random Forest
- Validation

머신 러닝

- 컴퓨터로 학습을 구현하는 것
 - 예: 문자 인식, 음성 인식, ...
 - 딥러닝도 머신러닝의 일종
- 샘플 데이터를 통해 패턴 학습 --> 새로운 데이터에서 패턴을 분류
 - 예: 손글씨 인식 → 새로운 손글씨에서 글자를 추출

양적 변수와 질적 변수

- 컴퓨터는 숫자만을 다룰 수 있음: 숫자로 연결된 변수/상수만을 다룰 수 있음
 - 양적 변수: 숫자의 크기가 의미를 가짐
 - 예: 키, 몸무게, 나이
 - 질적으로 다른 양적 변수는 서로 더할 수 없음
- 질적 변수: 숫자의 크기는 의미가 없음
 - 예: 성별코드, 학번
 - 연산 자체의 의미가 없음

질적으로 다른 숫자의 취급

- 질적으로 다른 것은 벡터에서 다른 차원으로 취급
 - 예: 빨강 = $(1, 0, 0, 0)$, 파랑 = $(0, 1, 0, 0)$
- 결국 기계학습은 특징을 나타내는 양으로 이루어진 벡터들의 공간에서 식별하고자 하는 것들을 잘 분리해낼 수 있는 일종의 구분선을 찾는 과정으로 볼 수 있음
 - 복잡한 함수로 이루어진 일종의 회귀 분석

머신 러닝의 구조

- 특징 추출: 데이터의 특징을 추출하여 벡터로 만드는 일
 - 예: 글자의 출현 빈도, 단어의 출현 빈도
- 학습: 벡터화된 학습 데이터를 통해 목적 함수의 매개 변수를 조절
- 평가: 학습의 성과를 평가
 - 학습 데이터 일부를 평가를 위한 데이터로 준비함

머신러닝의 종류

- 교사학습
- 비교사학습
- 강화학습

교사학습

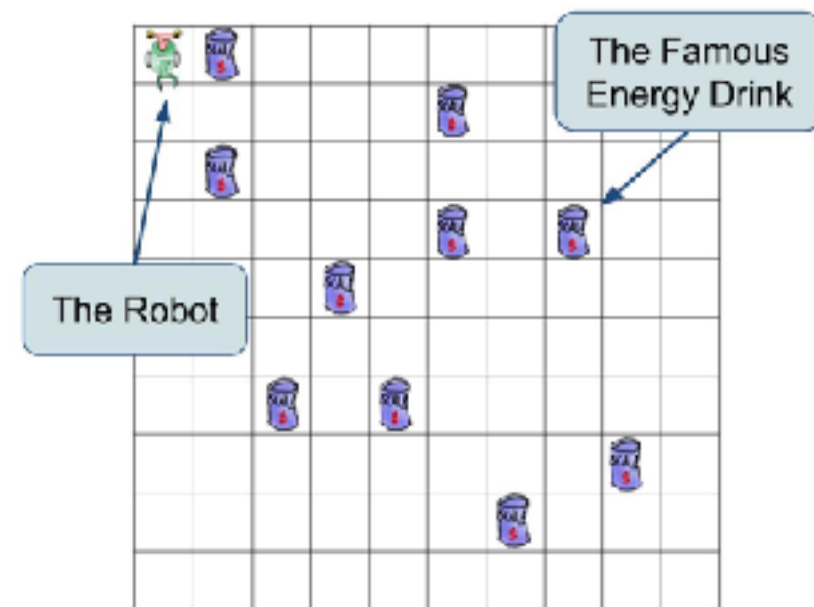
- Supervised Learning
- 데이터와 함께 그 데이터의 정답 (예: 패턴분류)을 함께 학습
 - 예:
 - 글자 이미지 데이터와 그 글자의 종류,
 - 사물 이미지들과 그 사물의 이름

비교사학습

- Unsupervised Learning
- 사람도 패턴을 분류하기 어려운 본질적 구조를 확인할 때 사용
- 예
 - 클러스터 분석
 - 주성분 분석
 - 벡터 양자화
 - 자기 조직화

강화 학습

- Reinforcement Learning
- 행동 주체와 환경
 - ex) sugarscape, 로봇 미로찾기 등
 - 행동을 수정해가면서 가장 보상 (성공함수 혹은 비용함수*(-1))을 크게 만들 행동 패턴을 찾아나가는 행위
- 완전한 답을 제공해주는 것은 아님



<http://blog.papauschek.com/wp-content/uploads/2014/02/Scala-Akka-Workshop-2.jpg>

머신러닝으로 할 수 있는 것

- 분류 (Classification)
- 그룹 나누기 (Clustering)
- 추천 (Recommendation)
- 회귀 (Regression)
- 차원 축소 (Dimension Reduction)

Curse of Dimensionality

- 특성량의 차원수에 비해 학습 데이터가 한정적일 경우 목표하고자 하는 성능을 내지 못하는 상황
- 학습된 데이터 분류는 해내지만 새로운 데이터에 대해서는 분류를 잘 해내지 못하는 경우
- 학습 데이터가 한정적일 경우 문제가 복잡할수록 문제가 됨

scikit-learn

scikit-learn

- Python 머신러닝 프레임워크
 - 머신러닝에 필요한 클래스, 함수들을 제공
- `$ pip3 install -U scikit-learn
scipy matplotlib scikit-image`
- `$ pip3 install -U pandas`

nand-train.py

Train Data (4)

Preprocessing

- `_sandbox/_ch04.ML/nand-train.py`

Learning

Prediction

```
# NAND data

nand_data = [
    #P, Q, P NAND Q
    [0,0,1],
    [0,1,1],
    [1,0,1],
    [1,1,0]
]

# divide data and label for learning

data = []
label = []
for row in nand_data:
    p = row[0]
    q = row[1]
    r = row[2]
    data.append([p,q])
    label.append(r)

# learning

clf = svm.SVC()
clf.fit(data,label)

# prediction

pre = clf.predict(data)
print(" 예측결과:",pre)

# display accuracy

ok, total = 0,0
for idx, answer in enumerate(label):
    p = pre[idx]
    if p == answer: ok +=1
    total +=1
print("정답률:",ok,"/",total,"=",ok/total)
```

Train Data

- P, Q : input (data)
- P NAND Q : output (label)
- 통상적으로 학습 데이터는 방대하기 때문에 이 부분은 별도의 데이터셋을 읽는 것으로 해결함
 - pandas

```
# NAND data

nand_data = [
    #P, Q, P NAND Q
    [0, 0, 1],
    [0, 1, 1],
    [1, 0, 1],
    [1, 1, 0]
]
```


Preprocessing

- 데이터를 변수로 할당하여 학습에 적합한 형태로 맞춤
 - normalisation
 - 자료형 일치

```
# divide data and label for learning
```

```
data = []  
label = []  
for row in nand_data:  
    p = row[0]  
    q = row[1]  
    r = row[2]  
    data.append([p,q])  
    label.append(r)
```

Learning

- sklearn의 SVM 객체를 생성
 - 학습 모형
- fit() 으로 데이터와 레이블의 연결을 설명할 패턴을 찾음
 - fit의 첫번째 인자: 데이터 (문제)
 - fit의 두번째 인자: 레이블 (정답)
- 이 형태를 맞추기 위해 preprocessing을 한 것임

```
# learning
```

```
clf = svm.SVC()  
clf.fit(data, label)
```

Prediction

- data만 가지고 앞에서 fit한 모형에 대입하여 추론한 결과를 출력
- 통상적으로는 별도의 테스트 셋을 사용함

```
# prediction
```

```
pre = clf.predict(data)  
print(" 예측결과:", pre)
```

정답률 계산

- 75%

```
# display accuracy
```

```
ok, total = 0, 0
for idx, answer in enumerate(label):
    p = pre[idx]
    if p == answer: ok += 1
    total += 1
print("정답률:", ok, "/", total, "=", ok / total)
```

프레임워크의 사용

- pandas를 사용하여 작업 단순화
- metrics.accuracy_score 사용

```
import pandas as pd
from sklearn import svm, metrics
```

```
#NAND
```

```
nand_input = [
    [0,0,1],
    [0,1,1],
    [1,0,1],
    [1,1,0]
]
```

```
# preprocessing
```

```
nand_df = pd.DataFrame(nand_input)
nand_data = nand_df.ix[:,0:1] # data
nand_label = nand_df.ix[:,2] #label
```

```
# learning and prediction
```

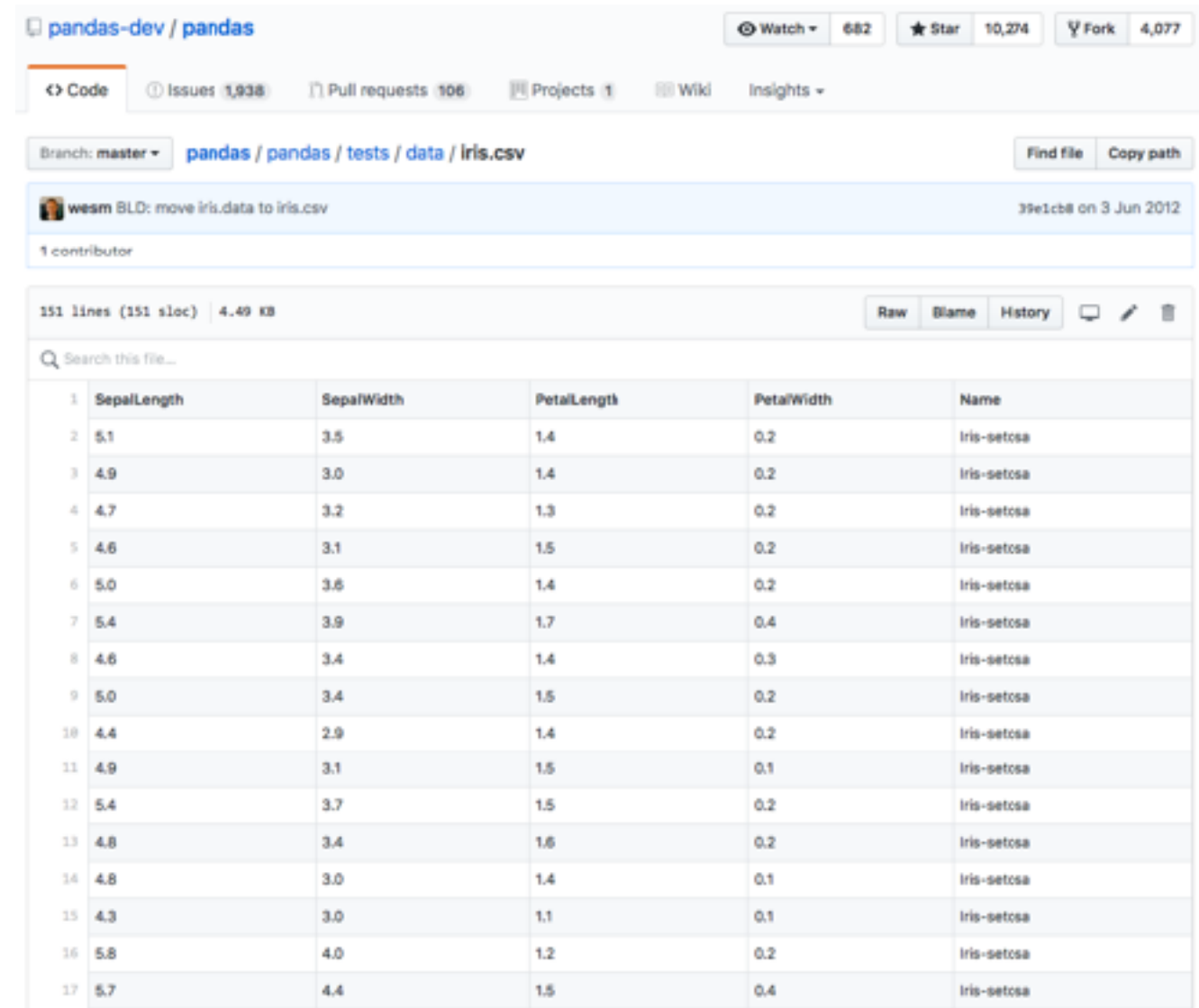
```
clf = svm.SVC()
clf.fit(nand_data,nand_label)
pre = clf.predict(nand_data)
```

```
# accuracy
```

```
ac_score = metrics.accuracy_score(nand_label,pre)
print("정답률 = ",ac_score)
```

붓꽃 (iris) 품종 분류

- data:: (#150)
 - SepalLength (꽃받침 길이), SepalWidth (꽃받침 폭), PetalLength (꽃잎길이), PetalWidth (꽃잎폭)으로
- label::
 - Iris-setosa, Iris-versicolor, Iris-virginica 품종을 분류
- <https://github.com/pandas-dev/pandas/blob/master/pandas/tests/data/iris.csv>



The screenshot shows the GitHub repository for pandas-dev/pandas. The file path is pandas / pandas / tests / data / iris.csv. The commit message is "wesm BLD: move iris.data to iris.csv" by 39e1cb8 on 3 Jun 2012. The file size is 4.49 KB. The table below is a preview of the data in the file.

| | SepalLength | SepalWidth | PetalLength | PetalWidth | Name |
|----|-------------|------------|-------------|------------|-------------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 8 | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 12 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 13 | 4.8 | 3.0 | 1.4 | 0.1 | Iris-setosa |
| 14 | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| 15 | 5.8 | 4.0 | 1.2 | 0.2 | Iris-setosa |
| 16 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |

iris-train2.py

- NAND-train.py와의 차이

- 데이터를 외부파일에서 읽음
- 훈련용, 테스트용셋을 분리함

```
import pandas as pd
from sklearn import svm, metrics
from sklearn.model_selection import train_test_split
```

```
# reading data
```

```
csv = pd.read_csv("_data/iris.csv")
```

```
# preprocessing
```

```
csv_data = csv[["SepalLength", "SepalWidth", "PetalLength", "PetalWidth"]]
csv_label = csv["Name"]
```

```
# separating learning data and test data
```

```
train_data, test_data, train_label, test_label = train_test_split(csv_data, csv_label)
```

```
# learning and prediction
```

```
clf = svm.SVC()
clf.fit(train_data, train_label)
pre = clf.predict(test_data)
```

```
# getting accuracy
```

```
acc_score = metrics.accuracy_score(test_label, pre)
print("정답률: ", acc_score)
```

MNIST 손글씨 숫자 인식

MNIST database

- <http://yann.lecun.com/exdb/mnist/>
- train-images-idx3-ubyte.gz: training set images (9912422 bytes)
- train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
- t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
- t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU

Corinna Cortes, Google Labs, New York

Christopher J.C. Burges, Microsoft Research, Redmond

se of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. The test set is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a square frame.

This database is for people who want to try learning techniques and pattern recognition methods on real-world data. It is not intended for people who want to do research on the effects of preprocessing and formatting.

Available on this site:

[train-images-idx3-ubyte.gz](#): training set images (9912422 bytes)
[train-labels-idx1-ubyte.gz](#): training set labels (28881 bytes)
[t10k-images-idx3-ubyte.gz](#): test set images (1648877 bytes)
[t10k-labels-idx1-ubyte.gz](#): test set labels (4542 bytes)

Your browser may uncompress these files without telling you. If the files you download have been uncompressed by your browser, simply rename them to remove the .gz extension (your application can't open your image files). These files are not in any standard image format (e.g., PNG) program to read them. The file format is described at the bottom of this page.



Data preprocessing

- download (mnist-download.py)
- uncompressing (mnist-download.py)
- binary to csv (mnist-tocsv.py)
- learning (mnist-train.py)