지도학습

지도학습 개요

데이터셋의 소개

K – Nearest Neighbors algorithm :분류, 회귀

선형모델:선형회귀(최소자승법),Ridge회귀,Lasso 회귀,…

Yoon Joong Kim

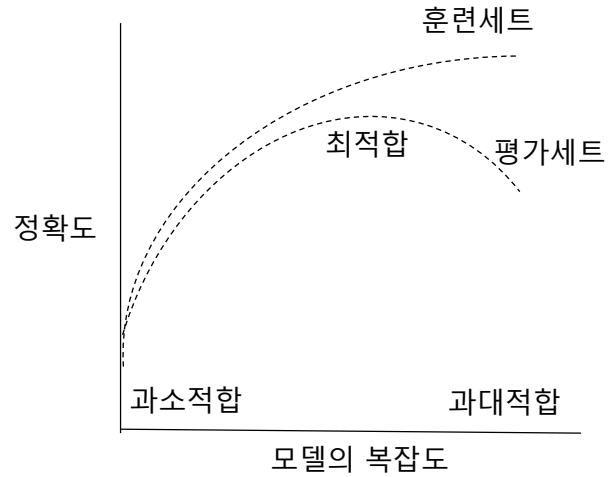
Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

content

1. 지도학습개요

- 지도학습의 정의
  - 데이터 샘플과 기대되는 결과 (X, y)의 데이터셋로
    알고리즘으로 모델을 학습(일반화) 시켜서
    임의의 샘플에 대하여 결과를 **예측**하는 방법
- 예측의 종류 : 분류와 회귀
  - 분류
    - 미리 정의된 가능성 있는 클래스의 하나를 예측하는 것이다.
    - 3 붓꽃품종 중 하나를 예측
    - 이진분류(binary classification),다중분류(multiclass classification)
  - 회귀
    - 연속적인 수, 실수를 예축하는 것이다
    - 나이, 성적점수
- 일반화,과대적합,과소적합
  - 일반화(학습데이터셋, 모델)
    - 지도학습에서 처음보는 테스트데이터가 훈련데이터와 특성이 같다면 정확하게 예측될 것이다.
      이와 같이 처음보는 데이터를 정확하게 예측할 수 있다면 훈련세트가 테스트세트로 일반화 되었다고 한다.
  - 과대적합(모델)
    - 너무 많은 정보를 이용하여 너무 복잡한 모델이 만들어 지는 현상을 '모델이 과대적합 되었다'고 한다.
    - 훈련세트의 모든 샘플에 너무 가까이 적합(학습)되어 새로운 데이터에 일반화되기 어렵다
  - 과소적합(모델)
    - 너무 적은 정보를 이용하여 너무 간단한 모델이 만들어 지는 현상을 '모델이 과소적합 되었다'고 한다.
  - 모델의 복잡도
    - 학습 데이터셋 크기에 비례한다.



모델의 복잡도에 따른 훈련 및 평가세트의 정확도변화

1. 지도학습개요(cont.)

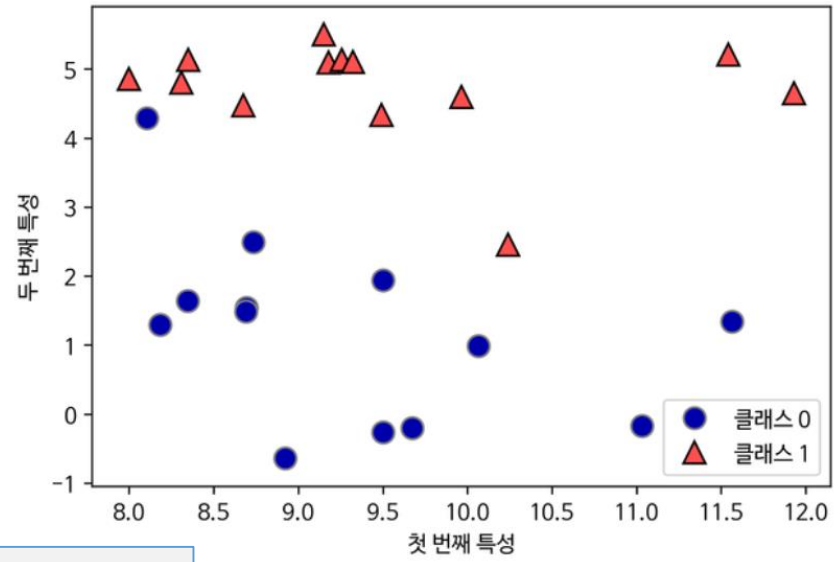
- 간단한 모델을 만드는 사례를 살펴본다
  - 문제의정의
    - 과거의 요트구매 정보를 이용하여 미래의 어떤 고객이 요트구매자 일수 있는 지를 예측하는 규칙을 찾는다.
  - 데이터 작성
    - 요트구매고객의 정보로 테이블을 작성한다.
  - 모델 (규칙) 작성
    - 테이블의 정보를 이용하여 유용한 규칙을 만든다.
    1. 나이가 45이상,차량을 보유,주택보유,자녀가2이상인 사람
       - 너무 복잡, 64세=>X
    2. 나이가 66,52,53,58세 인 사람
       - 100% OK
       - 새로운 데이터에 대하여 잘 맞을까?
    - 적합한 모델은 ?

표 2-1 고객 샘플 데이터

나이	보유차량수	주택보유	자녀수	혼인상태	애완견	보트구매
66	1	yes	2	사별	no	yes
52	2	yes	3	기혼	no	yes
22	0	no	0	기혼	yes	no
25	1	no	1	미혼	no	no
44	0	no	2	이혼	yes	no
39	1	yes	2	기혼	yes	no
26	1	no	2	미혼	no	no
40	3	yes	1	기혼	yes	no
53	2	yes	2	이혼	no	yes
64	2	yes	3	이혼	no	no
58	2	yes	2	기혼	yes	yes
33	1	no	1	미혼	no	no

2. 예제에서 사용할 데이터 셋

- 머신러닝 알고리즘의 모델에서 고려하여야 할 사항
  - 모델의 사용법
  - 모델의 복잡도 와 성능
  - 모델의 장단점
  - 모델의 매개변수와 옵션
  - 학습/평가용 데이터셋, 샘플의 특성
- 2.1 forge 데이터셋
  - 두 개의 특성을 가진 forge 인위적으로 만들어지는 데이터셋




```
# 데이터셋을 만듭니다.
X, y = mglearn.datasets.make_forge()
# 산점도를 그립니다.
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.legend(["클래스 0", "클래스 1"], loc=4)
plt.xlabel("첫 번째 특성")
plt.ylabel("두 번째 특성")
plt.show()
print("X.shape: {}".format(X.shape))  #(26, 2)
```

Forge

	X[:,,0]	X[:,1]	y
0	9.963466	4.596765	1
1	11.032954	-0.168167	0
…			
24	9.150723	5.498322	1
25	11.563957	1.338940	0

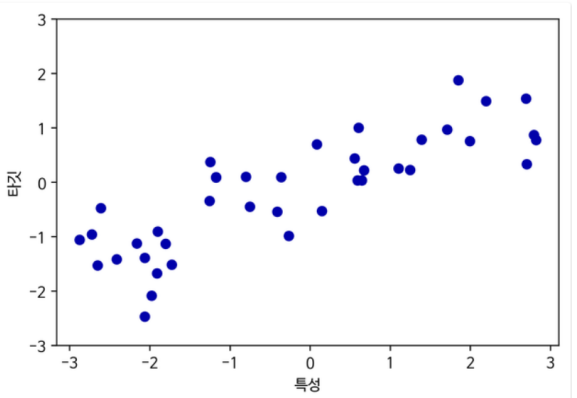
2. 예제에서 사용할 데이터 셋(cont.)

- 2.2 wave 데이터셋
  - 회귀 알고리즘 설명에는 인위적으로 만든 wave 데이터셋

```
import matplotlib.pyplot as plt
import mglearn
X, y = mglearn.datasets.make_wave(n_samples=40)  #X:(40, 1)   y:[(40,) float64]
print('X:{}   y:[{} {}]'.format(X.shape,y.shape,y.dtype)) #X:(40, 1)   y:[(40,) float64]
plt.plot(X, y, 'o')
plt.ylim(-3, 3)
plt.xlabel("특성")
plt.ylabel("타깃")
plt.show()
```

Wave

	X	y
0	-0.752759	-0.448221
1	2.704286	0.331226
…		
37	-2.413967	-1.415024
38	1.105398	0.254389
39	-0.359085	0.093989



- 2.3 위스콘신 유방암Wisconsin Breast Cancer 데이터셋
  - 암환자의 검진정보 30개를 특성으로 하고 결과 판정(암의 유무)한 데이터셋

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()  #data:(569, 30)   target:[(569,) int32 (2,)]
print(cancer.keys())
      #['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
print('data:{}   target:[{} {} {}]'.format(cancer.data.shape, cancer.target.shape,
      cancer.target.dtype,cancer.target_names.shape))
      #data:(569, 30)   target:[(569,) int32 (2,)]
```

2. 예제에서 사용할 데이터 셋(cont.)

- 2.4 보스턴 주택가격Boston Housing 데이터셋
  - 범죄율, 찰스강 인접도, 고속도로 접근성 등의 13개 정보를 특성으로 하고 1970년대 보스턴 주변의 주택 평균 가격을 기대 값으로 만들어진 데이터셋

```
from sklearn.datasets import load_boston
boston = load_boston()
print("데이터의 형태: {}".format(boston.data.shape))
# 데이터의 형태: (506, 13)
```

- 2.5 보스턴 주택가격Boston Housing 확장 데이터셋
  - 13개의 입력 특성뿐 아니라 특성끼리 곱하여(또는 상호작용이라 부름) 의도적으로 104개로 확장한 데이터 셋
  - 예, 범죄율과 고속도로 접근성의 개별 특성은 물론, 범죄율과 고속도로 접근성의 곱도 특성으로 생각한다는 뜻, **특성 공학**feature engineering

```
X, y = mglearn.datasets.load_extended_boston()
print("X.shape: {}".format(X.shape))
#X.shape: (506, 104)
```

- 요약

```
import mglearn
import sklearn
import numpy as np

X, y     = mglearn.datasets.make_forge()                 #X:(26, 2)              y:(26,),    int32, [0, 1], 분류용
X, y     = mglearn.datasets.make_wave(n_samples=40)      #X:(40, 1)              y:(40,),    float64,       회귀용
Cancer = sklearn.datasets.load_breast_cancer()           #data:(569, 30)  target:(569,),  int32, [0, 1]  분류용
Boston = sklearn.datasets.load_boston()                  #data:(506, 13)  target:(506,),  float64        회귀용
boston_ext= mglearn.datasets.load_extended_boston()      #data:(506, 104) target:(506,), float64        회귀용
X,y      =sklearn.datasets.make_blobs(random_state=42)#X:(100, 2)              y:(100,), int32, [0, 1, 2]] 분류용
```

Blobs

	X[:,0]	X[:,1]	y
0	-7.726421	-8.394957	2
1	5.453396	0.742305	1
…			
98	-5.796576	-5.826308	2
99	-3.348415	8.705074	0

Forge

	X[:,,0]	X[:,1]	y
0	9.963466	4.596765	1
1	11.032954	-0.168167	0
…			
24	9.150723	5.498322	1
25	11.563957	1.338940	0

Wave

	X	y
0	-0.752759	-0.448221
1	2.704286	0.331226
…		
38	1.105398	0.254389
39	-0.359085	0.093989

Cnacer

	0	1	...	29	target
0	17.990	10.38	...	0.11890	0
1	20.570	17.77	...	0.08902	0
…					
567	20.600	29.33	...	0.12400	0
568	7.760	24.54	...	0.07039	1

Boston

	0	1	...	12	target
0	17.990	10.38	…	0.11890	10000.1
1	20.570	17.77	…	0.08902	25000.0
…					
504	20.600	29.33	…	0.12400	12000.5
505	7.760	24.54	…	0.07039	50000.0

Boston

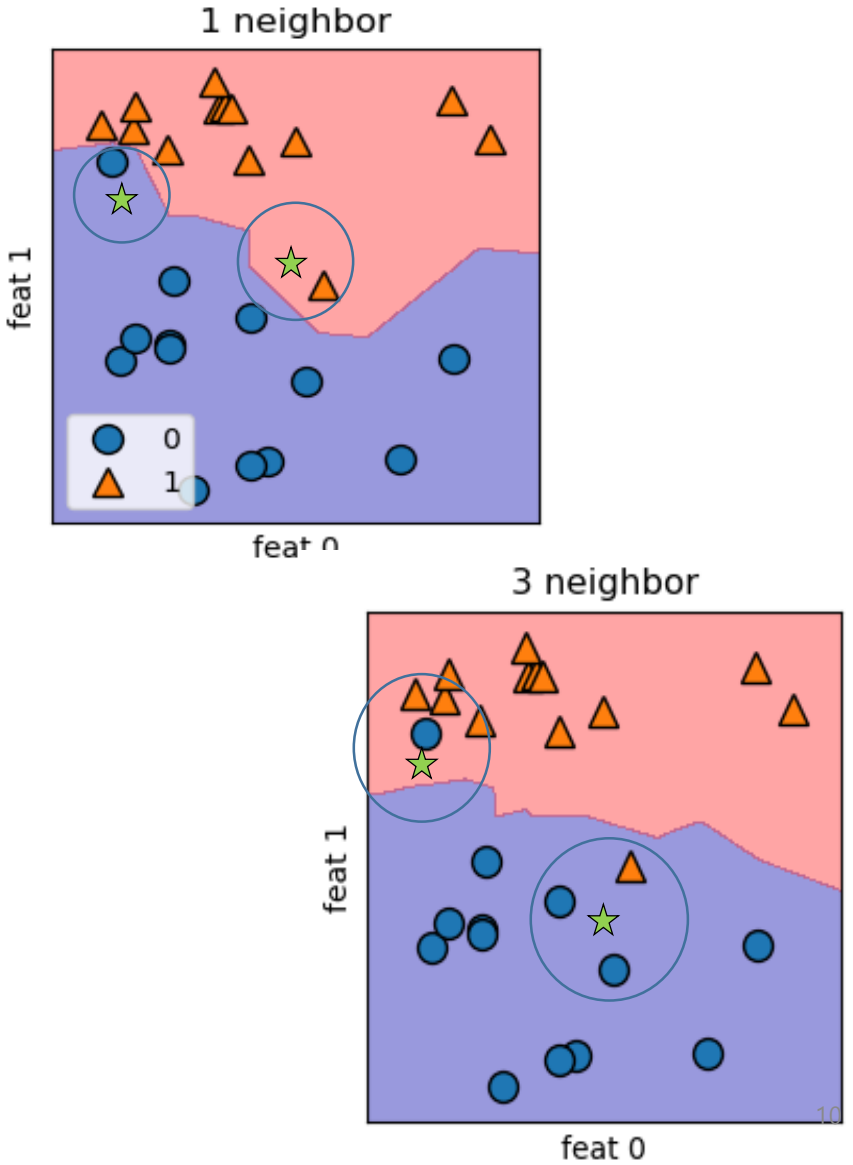
	0	1	...	103	target
0	17.990	10.38	…	0.11890	10000.1
1	20.570	17.77	…	0.08902	25000.0
…					
504	20.600	29.33	…	0.12400	12000.5
505	7.760	24.54	…	0.07039	50000.0

3. k-최근접 이웃

- 3. k-최근접 이웃 분류
  - k-NN(k-Nearest Neighbors 알고리즘(모델)
    - 평가 샘플로부터 최 근접에 위치한 K개의 모델 샘플의 정보를 이용하여 평가샘플이 속한 클래스를 예측하는 모델이다.
- 3.1 k-최근접 이웃 분류
  - 3.1.1 K-최근접 이웃 분류 개요
  - 3.1.2 K-최근접 이웃 분류 알고리즘 KNeighbordClassifiers
- 3.2 K-최근접 이웃 회귀
  - 3.2.1 K-최근접 이웃 회귀 개요
  - 3.2.2 K-최근접 이웃 회귀 알고리즘 KNeighborsRegressor

3.1.1 k-최근접 이웃 개요

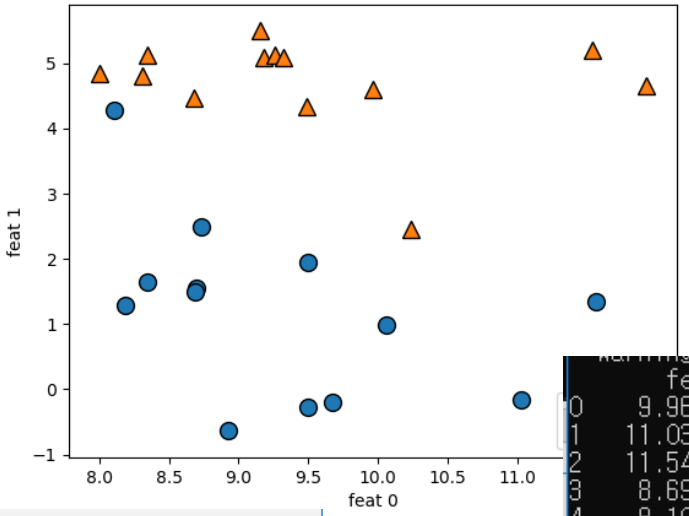
- 3.1.1 k-최근접 이웃 분류 개요
  - 모델 학습
    - 모델이 학습데이터를 그대로 보유
  - 분류방법
    - 임의로 주어지는 평가 데이터로부터
      모델데이터와 가장 가까운 k개의 대상데이터셋을 만들고
      대상데이터셋에서 최대 클래스를 분류결과로 판단한다.
  - 1 –Nearest Neighbor 모델의 예
    - 학습 : 13△ ,14● sample로 구성
    - 분류
      - 평가 샘플 ★ 대상클래스셋 ● 결과 ●
      - 평가 샘플 ★ 대상클래스셋 △ 결과 △
  - 3 –Nearest Neighbor의 예
    - 학습 : 13△ ,14● sample로 구성
    - 분류
      - 평가 샘플 ★ 대상클래스셋 △△● 결과 △
      - 평가 샘플 ★ 대상클래스셋 △●● 결과 ●



3.1.2 KNeighborsClassifier 알고리즘(cont.)

- 3-NeighborsClassifier 모델의 분석
  - forge 데이터 셋의 분석

```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import mglearn
from sklearn.neighbors import KNeighborsClassifier
#데이터셋 적재
X,y=mglearn.datasets.make_forge()      #인위적으로 데이터셋 생성
print(pd.DataFrame(                    #데이터셋 출력
      [list(xx)+[yy] for xx, yy in zip(X,y)],     #X,y데이터셋 zip
      columns=['feat 0','feat 1','class']))   #특성 및 label

mglearn.discrete_scatter(X[:,0],X[:,1],y)#산점도 분석
plt.xlabel('feat 0')
plt.ylabel('feat 1')
plt.legend(['feat 0','feat 1'],loc=4)
plt.show()
```




```
      feat 0    feat 1  class
0   9.963466  4.596765      1
1  11.032954 -0.168167      0
2  11.541558  5.211161      1
3   8.692890  1.543220      0
4   8.106227  4.286960      0
5   8.309889  4.806240      1
6  11.930271  4.648663      1
7   9.672847 -0.202832      0
8   8.348103  5.134156      1
9   8.674947  4.475731      1
10  9.177484  5.092832      1
11 10.240289  2.455444      1
12  8.689371  1.487096      0
13  8.922295 -0.639932      0
14  9.491235  4.332248      1
15  9.256942  5.132849      1
16  7.998153  4.852505      1
17  8.183781  1.295642      0
18  8.733709  2.491624      0
19  9.322983  5.098406      1
20 10.063938  0.990781      0
21  9.500490 -0.264303      0
22  8.344688  1.638243      0
23  9.501693  1.938246      0
24  9.150723  5.498322      1
25 11.563957  1.338940      0
데이터 shape
X_train:(19, 2) X_test:(7, 2) y_train:(19,)  y_test:(7,)
```

3.1.2 KNeighborsClassifier 알고리즘

- 3-NeighborsClassifier 모델의 분석
  - 모델생성/학습/평가

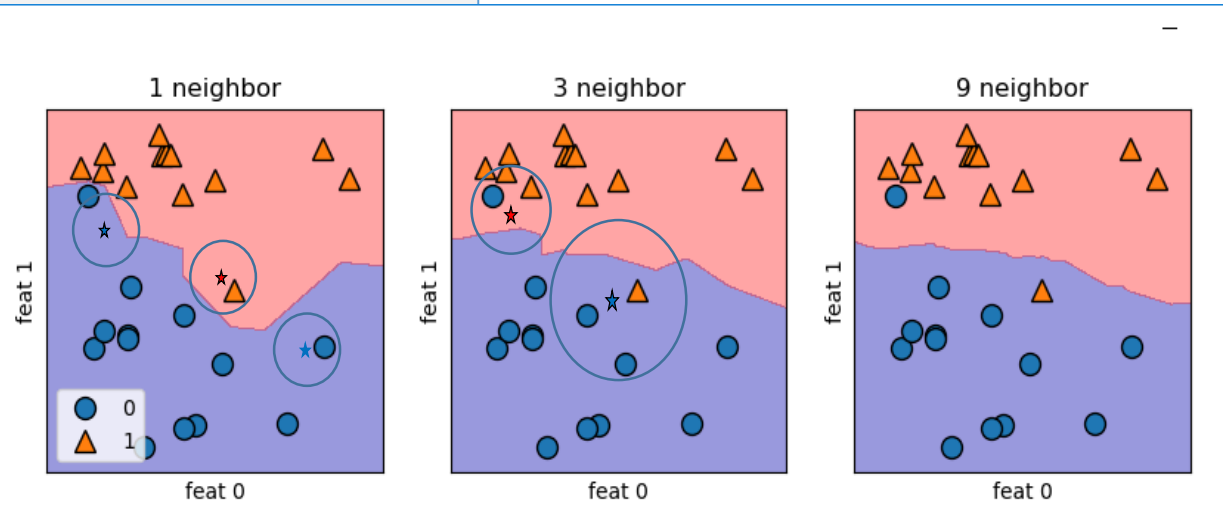
```
X,y=mglearn.datasets.make_forge()                          #데이터셋 적재
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0) #학습 및 평가셋으로 분리


#모델 생성 및 평가
print('\n\nKNeighborsClassifier 모델 생성 및 평가')
knn = KNeighborsClassifier(n_neighbors=3)                  #모델 정의(생성)
knn.fit(X_train, y_train)                                  #모델학습
print('모델예측  : {}'.format(knn.predict(X_test)))         #예측 [1 0 1 0 1 0 0]
print("모델정확도 : {:2f}".format(knn.score(X_test,y_test))) #모델평가 정확도:86%
```

```
KNeighborsClassifier 모델 생성 및 평가
모델예측   : [1 0 1 0 1 0 0]
모델타킷   : [1 0 1 0 1 1 0]
모델정확도 : 0.857143
Press any key to continue . . .
```

3.1.2 KNeighborsClassifier 알고리즘(cont.)

- K=[1,3,9] NeighborsClassifier 결정경계(성능)의 분석

```
fig,axes=plt.subplots(1,3, figsize=(10,3))      #크기 (10,3)인치의 캔버스, 3개의 하위그림 축 생성
for k_neighbor, ax in zip([1,3,9],axes) :       #K=[1,3,9]를 축[0,1,2]을 zip
    knn = KNeighborsClassifier(k_neighbors=n_neighbor).fit(X,y)   #kNN모델 생성 및 학습
    mglearn.plots.plot_2d_separator(            # ax축에 knn모델의 결정경계를 생성한다.
        knn, X, fill=True, eps=0.5,ax=ax, alpha=.4)
    mglearn.discrete_scatter(                   #ax축에 특징의 산점도를 배치한다.
        X[:,0], X[:,1], y, ax=ax)         #
    ax.set_title( " {} neighbor " .format(      #축의 title 설정
        k_neighbor))          #현재의 이웃 수
    ax.set_xlabel('feat 0')     #ax축의 x축 라벨
    ax.set_ylabel('feat 1')     #ax축의 y축 라벨
axes[0].legend(loc=3)           #범례를 ax0의 모서리 3에 배치
plt.show()                      #화면에 표시
```



- 1 neighbor 모델
  - 가장복잡,복잡한 경계, 과대적합, 훈련데이터 100%
- 3 neighbor 모델
  - 적합
- 9 neighbor 모델
  - 단순모델

3.1.2 KNeighborsClassifier 알고리즘(cont.)

- K=[1,2,..10] KNeighborsClassifier 복잡도와 정확도(성능) 분석
  - 유방암 데이터 셋 (569,30)

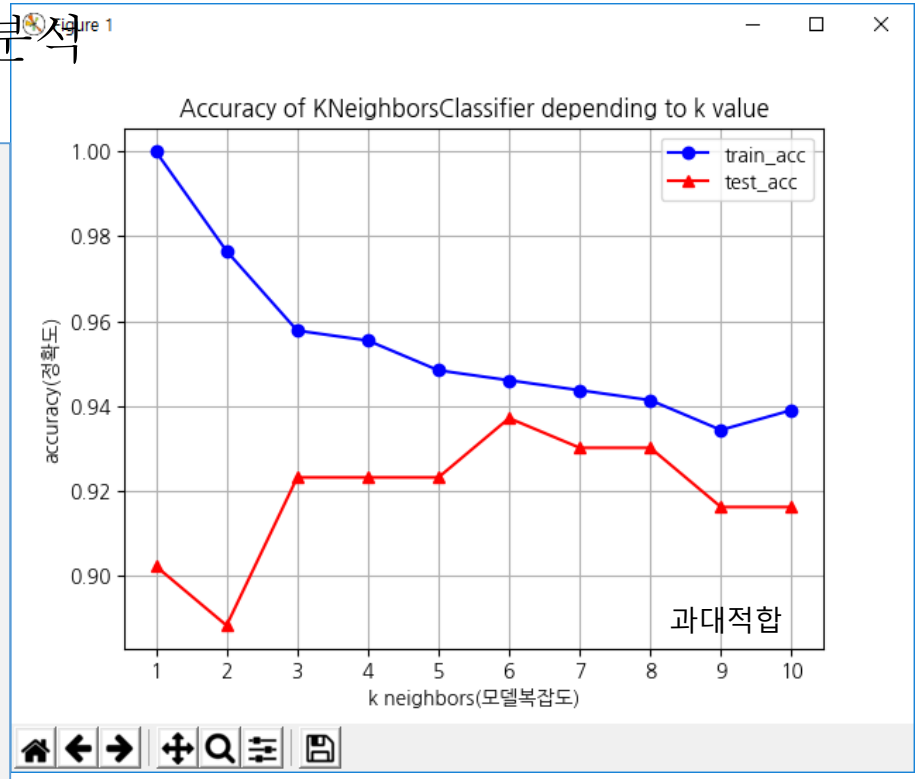
```
from sklearn.model_selection import train_test_split
from sklearn import datasets as skds
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import mglearn
from sklearn.neighbors import KNeighborsClassifier

#유방암 데이터 셋 (569,30) 적재
ds_cancer=skds.load_breast_cancer() #유방암 데이터셋
print(pd.DataFrame(                 #데이터셋의 기본정보  출력
    [list(xx)+[yy] for xx, yy in zip(ds_cancer.data,ds_cancer.target)]   )) # 특징,타켓 통합

X_train,X_test,y_train,y_test=train_test_split(                          #학습 및 평가셋 생성
    ds_cancer.data,ds_cancer.target,stratify=ds_cancer.target,random_state=66)

print('데이터 shape₩nX_train:{} X_test:{} y_train:{}  y_test:{}          #데이터셋 형태  출력
    '.format(X_train.shape,X_test.shape,y_train.shape,y_test.shape))
```

```
        0      1       2       3       4       5       6   ...      24      25      26      27      28      29  30
0   17.99  10.38  122.80  1001.0  0.11840  0.27760  0.30010  ...  0.16220  0.66560  0.7119  0.2654  0.4601  0.11890   0
1   20.57  17.77  132.90  1326.0  0.08474  0.07864  0.08690  ...  0.12380  0.18660  0.2416  0.1860  0.2750  0.08902   0
2   19.69  21.25  130.00  1203.0  0.10960  0.15990  0.19740  ...  0.14440  0.42450  0.4504  0.2430  0.3613  0.08758   0
3   11.42  20.38   77.58   386.1  0.14250  0.28390  0.24140  ...  0.20980  0.86630  0.6869  0.2575  0.6638  0.17300   0
4   20.29  14.34  135.10  1297.0  0.10030  0.13280  0.19800  ...  0.13740  0.20500  0.4000  0.1625  0.2364  0.07678   0
..    ...    ...     ...     ...      ...      ...      ...  ...      ...      ...     ...     ...     ...  ..
564 21.56  22.39  142.00  1479.0  0.11100  0.11590  0.24390  ...  0.14100  0.21130  0.4107  0.2216  0.2060  0.07115   0
565 20.13  28.25  131.20  1261.0  0.09780  0.10340  0.14400  ...  0.11660  0.19220  0.3215  0.1628  0.2572  0.06637   0
566 16.60  28.08  108.30   858.1  0.08455  0.10230  0.09251  ...  0.11390  0.30940  0.3403  0.1418  0.2218  0.07820   0
567 20.60  29.33  140.10  1265.0  0.11780  0.27700  0.35140  ...  0.16500  0.86810  0.9387  0.2650  0.4087  0.12400   0
568  7.76  24.54   47.92   181.0  0.05263  0.04362  0.00000  ...  0.08996  0.06444  0.0000  0.0000  0.2871  0.07039   1

[569 rows x 31 columns]
데이터 shape
X_train:(426, 30) X_test:(143, 30) y_train:(426,)  y_test:(143,)
```

3.1.2 KNeighborsClassifier 알고리즘(cont.)

- K=[1,2,..10] KNeighborsClassifier 복잡도와 정확도(성능) 분석
  - 10개 모델 생성/학습/평가

```
#유방암 데이터셋 적재 , 학습 및 평가셋 생성
ds_cancer=skds.load_breast_cancer()
X_train,X_test,y_train,y_test=train_test_split(
ds_cancer.data,ds_cancer.target,stratify=ds_cancer.target,random_state=66)

train_acc=[];test_acc=[]
ks=range(1,11)      #최근 이웃 수 k리스트 생성
for k in ks :
      knn=KNeighborsClassifier(k).fit(X_train,y_train)#knn모델 생성 및 학습
      train_acc.append(knn.score(X_train,y_train))#모델의 훈련데이터 정확도추가
      test_acc.append( knn.score(X_test,y_test))   #모델의 평가데이터 정확도추가
plt.plot(ks,train_acc,'bo-',label='train_acc')   #훈련데이터정확도 그래프
plt.plot(ks,test_acc,'r^-',label='test_acc')      #평가데이터정확도 그래프
plt.xlabel('k neighbors(모델복잡도)')   #x축 라벨
plt.ylabel('accuracy(정확도)')          #y축 라벨
plt.xticks(ks)                          #x축 눈금 설정
plt.legend(loc='best')                  #범례 배치
plt.title('Accuracy of KNeighborsClassifier depending to k value') #제목
plt.grid()                              #그리드 배치
plt.show()                              #그림 출력
```



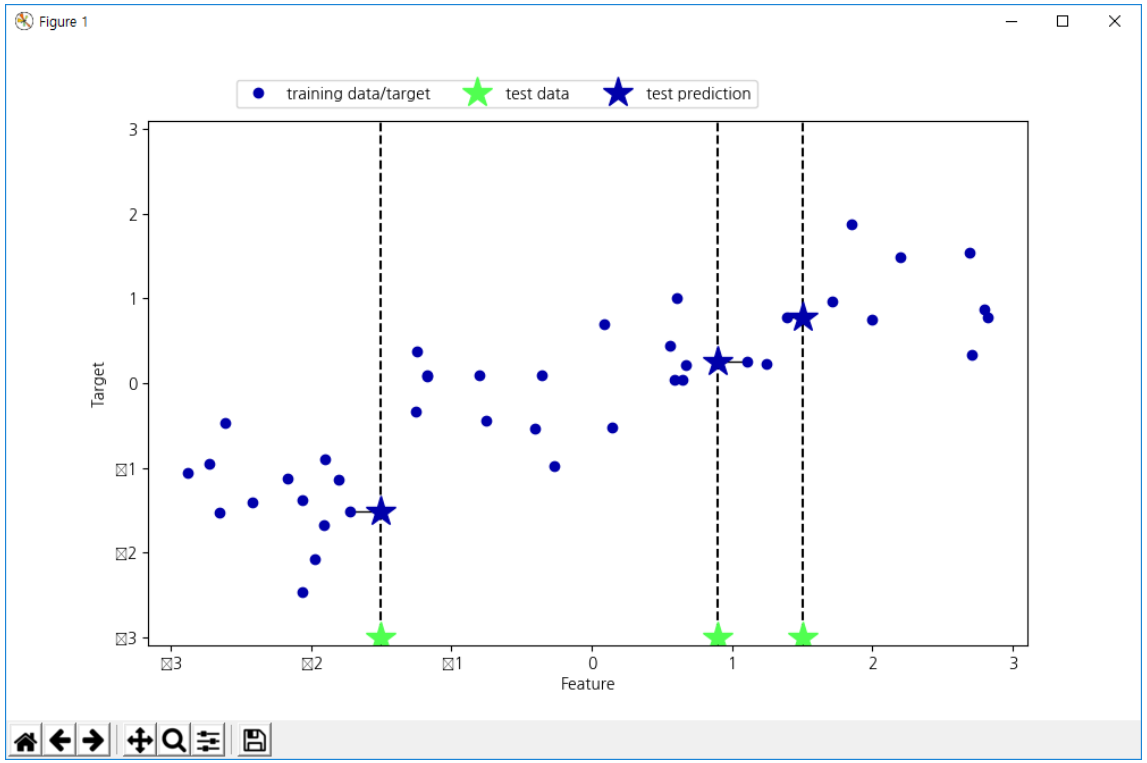
복잡도
정확도

K=1 모델 복잡도, 훈련데이터 100%, 평가데이터 최저
k증가시 점점 단순 정확도 낮아진다.
K=6에서 최적합 (평가데이터 정확도 최대)
k=10 모델 최소단순

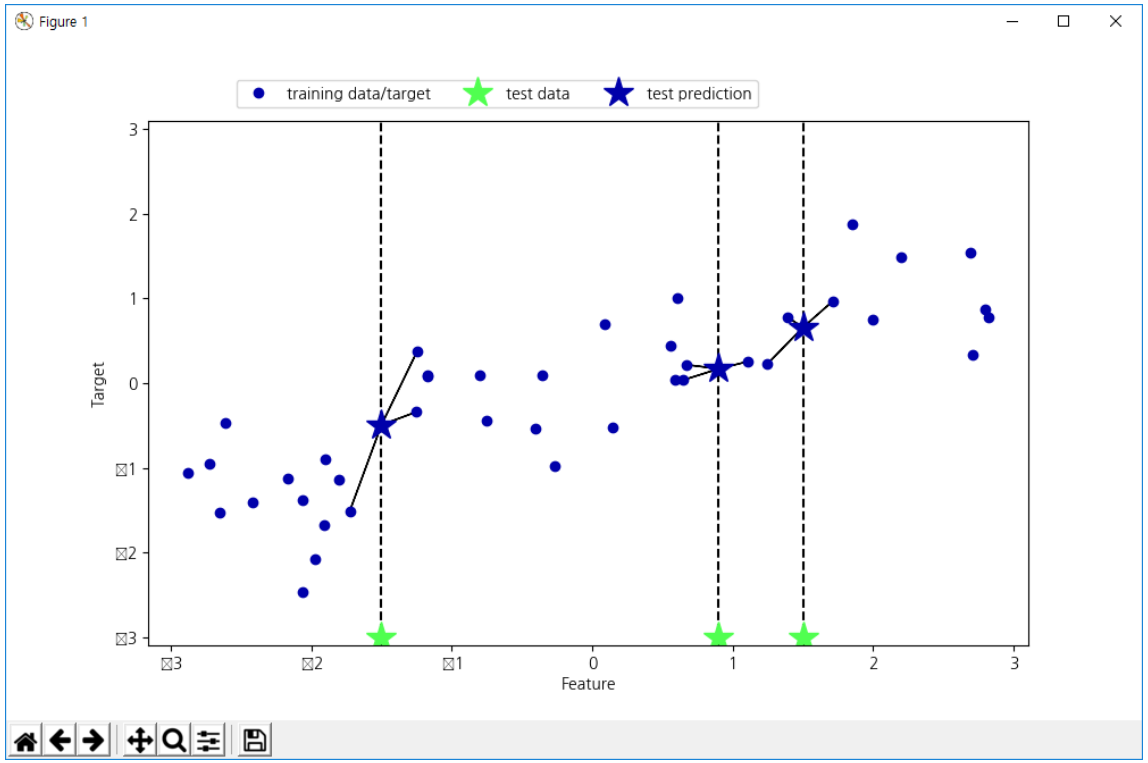
3.2 k 최근접 이웃 회귀

- 3.2.1 k 최근접 이웃 회귀(KNeighborsRegressor)개요
  - 최근접 이웃 분류기를 이용한 회귀–KNeighborsRegressor

```
#wave 데이터셋을 이용한  k-최근접 이웃 회귀, x축에 평가용 1개의 녹색 별에 대한 예측방법
mglearn.plots.plot_knn_regression(n_neighbors=1)
plt.show()
```




```
#wave 데이터셋을 이용한  k-최근접 이웃 회귀, x축에 평가용 3개의 녹색 별에 대한 예측방법
mglearn.plots.plot_knn_regression(n_neighbors=3)
plt.show()
```



3.2.2 k 최근접 이웃 회귀(kNeighborsRegressor)

- 2.3.2 k 최근접 이웃 회귀(KNeighborsRegressor)

```
# wave 데이터셋을 적재, 훈련, 테스트 세트 생성
X, y = mglearn.datasets.make_wave(n_samples=40) #(40,1), (40,)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
#(30,1), (10,1), (30,), (10,),

# 이웃의 수를 3으로 하여 모델의 객체를 만듭니다.
reg = KNeighborsRegressor(n_neighbors=3)
reg.fit(X_train, y_train)              # 모델을 학습시킵니다.

print("테스트 sample     : {}".format([X_test[0]]))
print("테스트 sample 예측: {}".format(reg.predict([X_test[0]])))

print("테스트 세트      :{}".format(X_test))
print("테스트 세트 예측 :{}".format(reg.predict(X_test)))
print("테스트 세트 라벨 :{}".format(y_test))
print("테스트 성능 R^2  : {:.2f}".format(reg.score(X_test, y_test)))
```

```
테스트 sample      :  [array([-1.24713211])]
테스트 sample 예측 : [-0.05396539]

테스트 세트        : [[-1.24713211], [ 0.67111737], [ 1.71105577], [-2.06388816], [-2.87649303],  [-1.89957294], [ 0.55448741], [ 2.81945911], [-0.40832989], [-2.72129752]]

테스트 세트 예측 : [-0.05396539  0.35686046  1.13671923 -1.89415682 -1.13881398 -1.63113382  0.35686046  0.91241374 -0.44680446 -1.13881398]

테스트 세트 라벨 : [ 0.37299129  0.21778193  0.96695428 -1.38773632 -1.05979555 -0.90496988  0.43655826  0.7789638  -0.54114599 -0.95652133]

테스트 성능 R^2  : 0.83
```

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2},$$

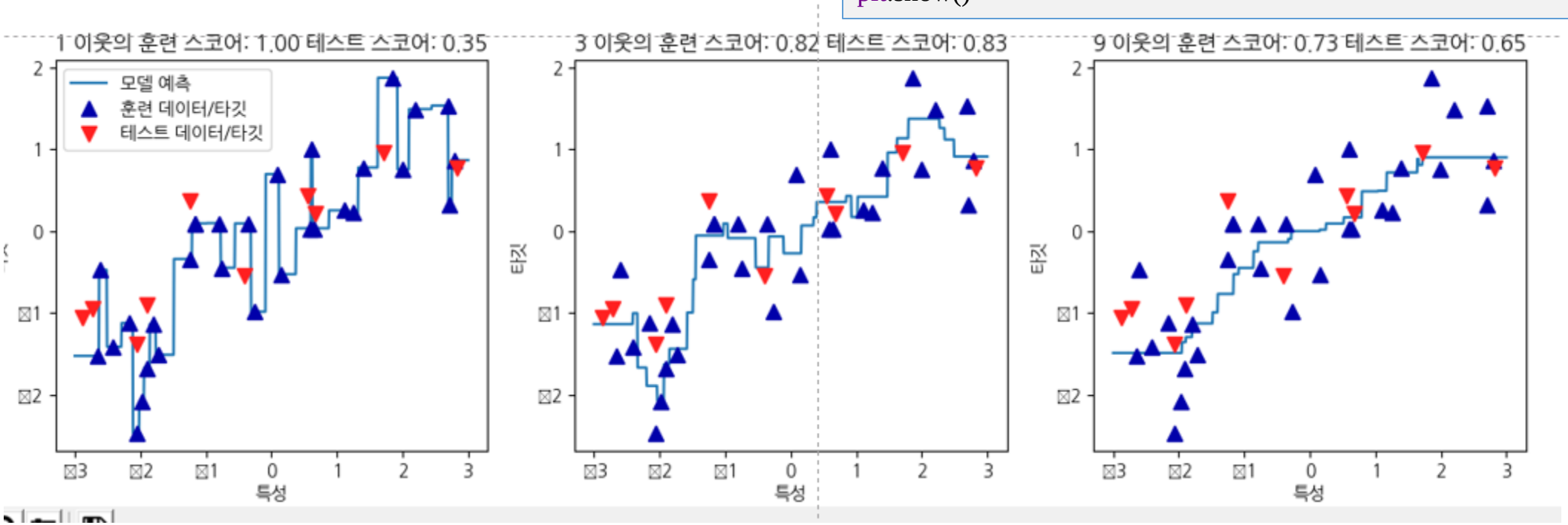
y_i : target
$\bar{y}$: target의 mean
$\hat{y}_i$:예측치

3.2.2 2 k 최근접 이웃 회귀(KNeig

- 3.3.2 k 최근접 이웃 회귀(KNeighborsRegressor)분석

```
# wave 데이터셋 적재, 훈/평가셋 생성
X, y = mglearn.datasets.make_wave(n_samples=40)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

fig, axes = plt.subplots(1, 3, figsize=(15, 4))
# -3과 3 사이에 1,000개의 데이터 포인트를 만듭니다.
line = np.linspace(-3, 3, 1000).reshape(-1, 1)
```

```
for n_neighbors, ax in zip([1, 3, 9], axes):
    # 1, 3, 9 이웃을 사용한 예측을 합니다.
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)
    ax.plot(line, reg.predict(line))
    ax.plot(X_train, y_train, '^', c=mglearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mglearn.cm2(1), markersize=8)

    ax.set_title(
    "{} 이웃의 훈련 스코어: {:.2f} 테스트 스코어: {:.2f}".format(
        n_neighbors, reg.score(X_train, y_train),  reg.score(X_test,
y_test)))
    ax.set_xlabel("특성");    ax.set_ylabel("타깃")
axes[0].legend(["모델 예측", "훈련 데이터/타깃", "테스트 데이터/타
깃"],
               loc="best")
plt.show()
```



3.2.2 k 최근접 이웃 회귀(KNeighborsRegressor)분석

- 일반적으로 KNeighbors 분류기에 중요한 매개변수는 두 개입니다.
  - 데이터 포인트 사이의 **거리계산방법**과 **이웃의 수**입니다. 실제로 이웃의 수는 3개나 5개 정도로 적을 때 잘 작동하지만, 이 매개변수는 잘 조정해야 합니다.
  - **거리계산방법**은 기본적으로 여러 환경에서 잘 동작하는 유클리디안 거리 방식을 사용합니다.
- k-NN의 장점은 이해하기 매우 쉬운 모델이라는 점입니다.
  - 그리고 많이 조정하지 않아도 자주 좋은 성능을 발휘합니다. 더 복잡한 알고리즘을 적용해보기 전에 시도해볼 수 있는 좋은 시작점입니다.
  - 보통 최근접 이웃 모델은 매우 빠르게 만들 수 있지만, 훈련 세트가 매우 크면 (특성의 수나 샘플의 수가 클 경우) 예측이 느려집니다.
  - k-NN 알고리즘을 사용할 땐 데이터를 전처리하는 과정이 중요합니다. 그리고 (수백 개 이상의) 많은 특성을 가진 데이터셋에는 잘 동작하지 않으며,
  - 특성 값 대부분이 0인 (즉 희소한) 데이터셋과는 특히 잘 작동하지 않습니다.
- k-최근접 이웃 알고리즘이 이해하긴 쉽지만,
  - 예측이 느리고 많은 특성을 처리하는 능력이 부족해 현업에서는 잘 쓰지 않습니다.
  - 이런 단점이 없는 알고리즘이 다음에 설명할 선형 모델입니다.

content

- 지도학습
  1. 지도학습 개요
     1. 분류,회귀,일반화,과대적합,과소적합,복잡도, 데이터셋크기
  2. 예제에서 사용할 데이터셋
     1. X, y = mglearn.datasets.make_forge() #X:(26, 2)y:(26,) int32 [0, 1]
     2. X, y = mglearn.datasets.make_wave(n_samples=40) #X:(40, 1)y:(40,) float64]
     3. cancer = sklearn.datasets.load_breast_cancer() #data:(569, 30) target:(569,) int32 [0, 1]
     4. boston = sklearn.datasets.load_boston() #data:(506, 13) target:(506,) float64
     5. boston_ext= mglearn.datasets.load_extended_boston() #data:(506, 104) target:(506,) float64
     6. X,y =sklearn.datasets.make_blobs(random_state=42) #X:(100, 2) y:[(100,) int32 [0, 1, 2]]
  3. K-최근이웃
     1. K-최근접 이웃 분류
        - 개요, 2. KNeighbordClassifiers 분석
     2. K-최근접 이웃 회귀
        - 개요, 2. KNeighborsRegressor 분석
  4. 선형모델
     1. 회귀선형모델
     2. 선형회귀(최소자승법)
     3. Ridge회귀
     4. Lasso 회귀
     5. 분류용 선형 모델
     6. 다중클래스 분류용 선형 모델
     7. 장단점과 매개변수