머신러닝
(Machine Learning)

LECTURE VIII: 지도 학습 2
(Supervised Learning)

Dai-Gyoung Kim

Department of Applied Mathematics

Hanyang University ERICA

ⓒ dgkim@hanyang.ac.kr

지도 학습 (Supervised Learning)

Contents

ⓒ dgkim@hanyang.ac.kr

▶ k-최근접 이웃

- k-최근접 이웃 (k-NN[k-Nearest Neighbors]) 분류기는 비모수적 확률 모형을 갖는 간단한 분류기임.
- 이 분류기의 기본 아이디어는 단순히 훈련 데이터셋에서 테스트 입력 포인트에 가장 가까이 있는 k개의 데이터 포인트(최근접 이웃)를 찾아 예측에 사용함.

k-NN의 수학적 표현

- 훈련 데이터셋 X에서 테스트 입력 포인트 x에 가장 가까이 있는 k개의 데이터 포인트를 찾아, 각 클래스의 맴버가 얼마나 이 집합에 있는지 개수를 계산하고, 그 비율을 추정 값으로 반환함.

ⓒ dgkim@hanyang.ac.kr

- **추정 확률 모형**

$$p(y=c \mid \boldsymbol{x}, D, k) = \frac{1}{k} \sum_{i \in N_k(\boldsymbol{x}, D)} \mathbb{I}(y_i = c)$$

▷ $N_k(\boldsymbol{x}, D)$: D 안에서 $\boldsymbol{x}$에 대한 k 근접 포인트의 인덱스의 집합.

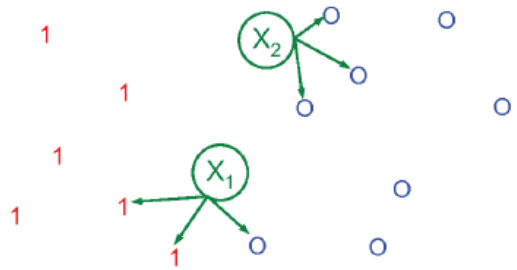
▷ $\mathbb{I}(s)$: 표시 함수(indicator function)

$$\mathbb{I}(s) = \begin{cases} 1 & \text{if } s \text{ is true} \\ 0 & \text{if } s \text{ is false} \end{cases}$$

- **오른쪽 그림은 $k=3$에 대한 k-NN 분류를 보여줌.**

✓ $p(y=1 \mid \boldsymbol{x}_1, D, k=3) = \frac{2}{3}$

✓ $p(y=1 \mid \boldsymbol{x}_2, D, k=3) = \frac{0}{3}$

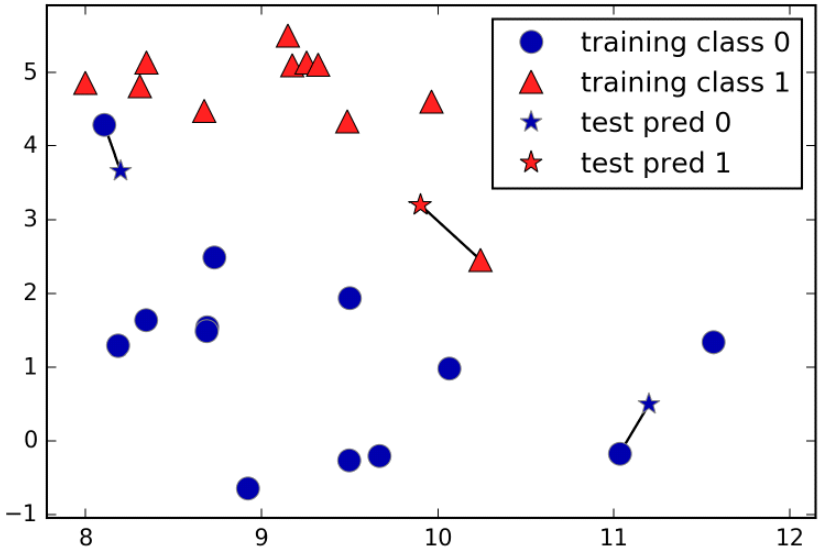


© dgkim@hanyang.ac.kr

❖ k-NN 분류

- 다음 코드는 forge 데이터셋에 대한 1-NN 모델의 예측을 보여줌.

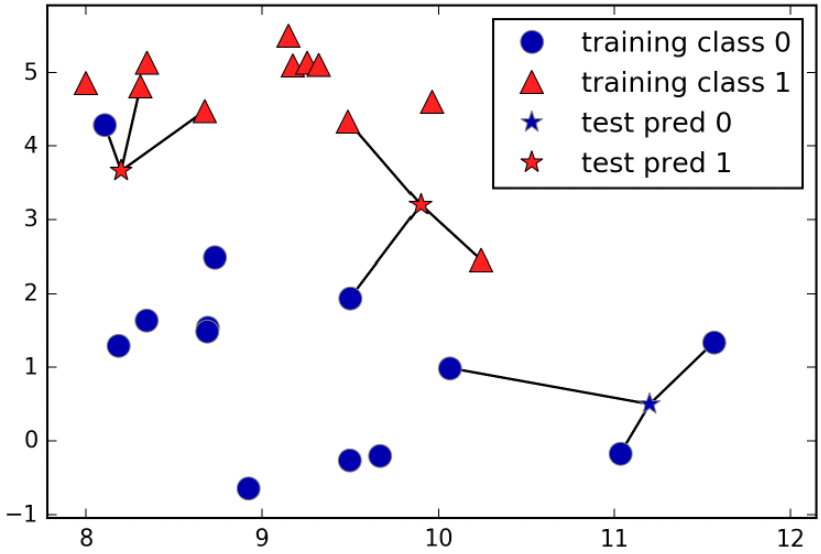
```
mglearn.plots.plot_knn_classification(n_neighbors=1)
```



© dgkim@hanyang.ac.kr

- 다음 코드는 forge 데이터셋에 대한 3-NN 모델의 예측을 보여줌.

```
mglearn.plots.plot_knn_classification(n_neighbors=3)
```



© dgkim@hanyang.ac.kr

k-NN 알고리즘의 적용

- "scikit-learn"을 사용하여 k-NN 알고리즘을 적용하기 위해 먼저 일반화 성능을 평가할 수 있도록 forge 데이터셋을 훈련 세트와 테스트 세트로 나눔.

```
from sklearn.model_selection import train_test_split
X, y = mglearn.datasets.make_forge()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

- 다음은 "KNeighborsClassifier"를 임포트하고 객체를 만듦.

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
```

- 이제 훈련 세트를 사용하여 분류 모델을 학습시킴

```
clf.fit(X_train, y_train)
```


ⓒ dgkim@hanyang.ac.kr

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

- 테스트 데이터에 대해 예측을 하기 위해 “predict” 메서드를 호출함.

```
print("Test set predictions:", clf.predict(X_test))
```

```
Test set predictions: [1 0 1 0 1 0 0]
```

- 일반화 성능을 평가하기 위해 “score” 메서드에 테스트 데이터와 테스트 레이블을 넣어 호출함.

```
print("Test set accuracy: {:.2f}".format(clf.score(X_test, y_test)))
```

```
Test set accuracy: 0.86
```

✓ 이 모델은 테스트 데이터셋에 있는 샘플 중 86%를 정확히 예측함.

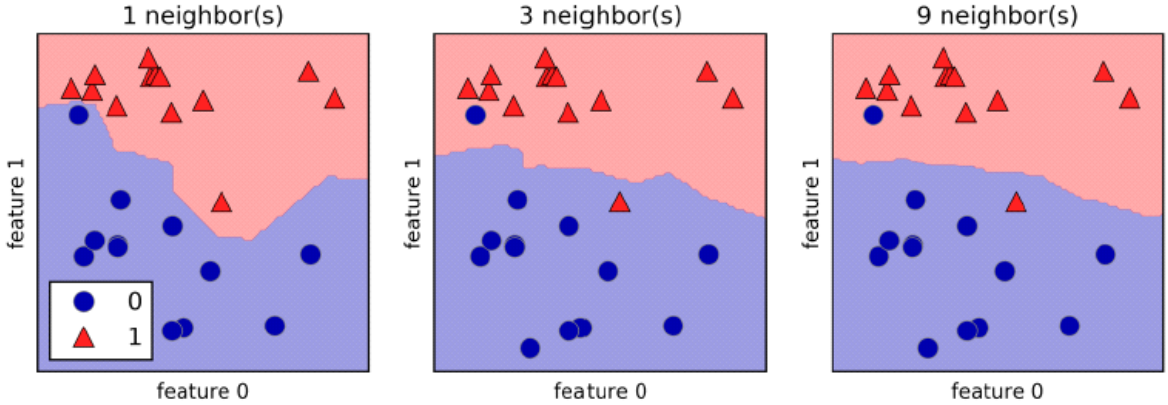
ⓒ dgkim@hanyang.ac.kr

❖ KNeighborsClassifier 분석

- 다음 코드는 $k = 1,\ 3,\ 9$일 때의 **결정 경계(decision boundary)**를 보여줌.
- 결정 경계는 두 개의 클래스 (0, 1)로 지정한 영역을 나누는 최적의 경계임.

```
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

for n_neighbors, ax in zip([1, 3, 9], axes):
    # fit 메서드는 self 객체를 반환함
    # 따라서 객체 생성과 fit 메서드를 한 줄에 쓸 수 있음
    clf = KNeighborsClassifier(n_neighbors=n_neighbors).fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=True, eps=0.5, ax=ax, alpha=.4)
    mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
    ax.set_title("{} neighbor(s)".format(n_neighbors))
    ax.set_xlabel("feature 0")
    ax.set_ylabel("feature 1")
axes[0].legend(loc=3)
```


© dgkim@hanyang.ac.kr



- ✓ 이웃의 수 k를 늘릴수록 모델의 복잡도는 낮아지고, 결정 경계는 더 부드러워짐.
- ✓ 이웃의 수 k를 적게 할수록 모델의 복잡도는 높아지고, 결정 경계는 복잡해짐.

모델의 복잡도와 일반화의 관계

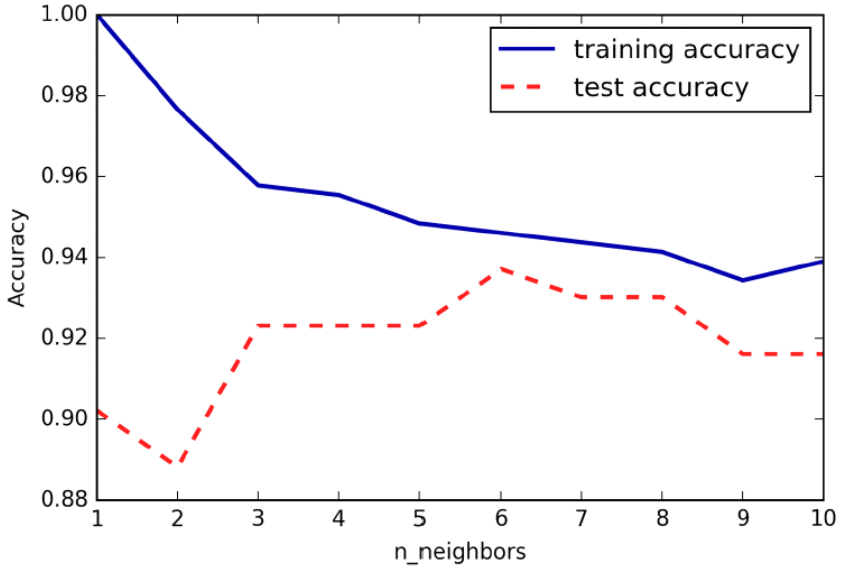
- 유방암 데이터셋을 사용하여 훈련 세트와 테스트 세트의 성능을 평가해보고자 함.
- 다음은 유방암 데이터셋을 훈련 세트와 테스트 세트로 나누고, 이웃의 수를 달리하여 훈련 세트와 테스트 세트의 성능 평가를 구현하는 코드임.

© dgkim@hanyang.ac.kr

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=66)

training_accuracy = [ ]
test_accuracy = [ ]
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 11)

for n_neighbors in neighbors_settings:
    # build the model
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(clf.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(clf.score(X_test, y_test))
```


© dgkim@hanyang.ac.kr

```
plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```



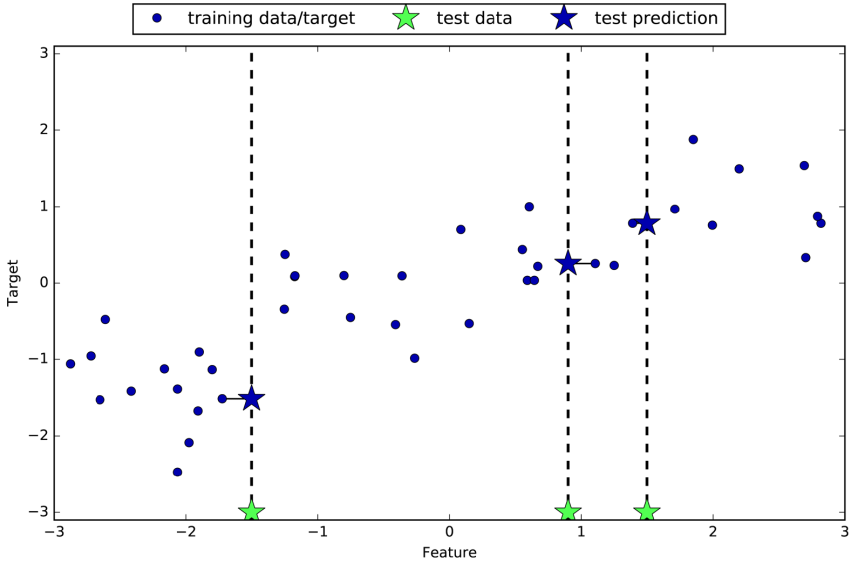
- ✓ 최근접 이웃의 수가 하나일 때는 훈련 데이터에 대한 예측이 완벽함.
- ✓ 이웃의 수가 늘어나면 모델이 단순해지고 훈련 데이터의 정확도는 줄어듦.
- ✓ 테스트 데이터의 경우 적은 이웃의 수(복잡한 모델)를 사용할 때와 많은 이웃의 수(단순한 모델)를 사용할 때 정확도가 줄어드는 경향이 있음.
- ✓ 테스트 데이터의 경우 최적의 정확도는 중간 정도의 이웃의 수를 사용할 때 얻어짐.

ⓒ dgkim@hanyang.ac.kr

❖ k-최근접 이웃 회귀

- k-최근접 이웃 회귀(k-neighbors regression)은 k-NN 알고리즘을 적용한 것임.
- 1-최근접을 사용하면, 새로운 데이터 포인트의 예측은 가장 가까운 이웃의 타깃값임.

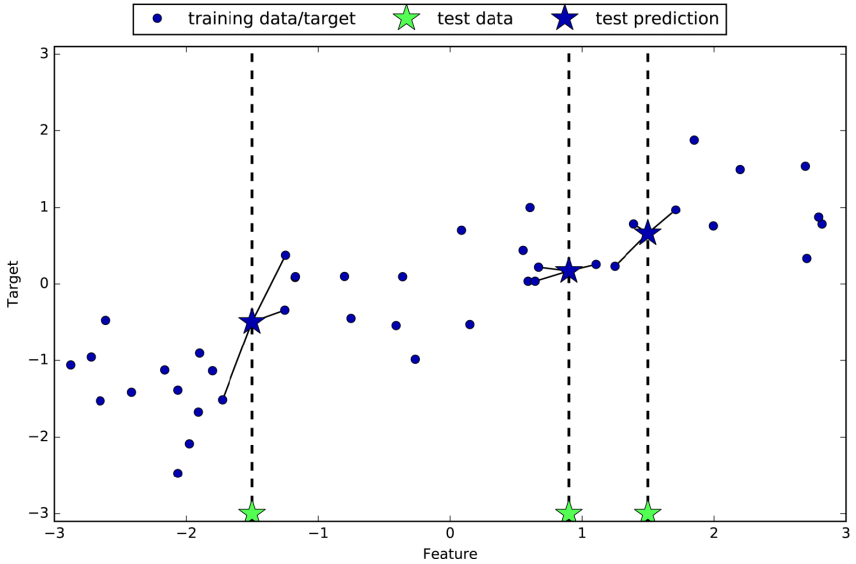
```
mglearn.plots.plot_knn_regression(n_neighbors=1)
```



© dgkim@hanyang.ac.kr

- 다음은 3-최근접 이웃 회귀 모델의 예측임.

```
mglearn.plots.plot_knn_regression(n_neighbors=3)
```



✓ 여러 개의 최근접 이웃을 사용할 때는, 이웃 타깃값의 평균으로 예측함.

© dgkim@hanyang.ac.kr

- scikit-learn에서 회귀를 위한 k-최근접 이웃 알고리즘은 “KNeighborsRegression”으로 구현할 수 있음.

```
from sklearn.neighbors import KNeighborsRegressor

X, y = mglearn.datasets.make_wave(n_samples=40)
# wave 데이터셋을 훈련 세트와 테스트 세트로 나눔
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# 이웃의 수를 3으로 하여 모델의 객체를 만듦
reg = KNeighborsRegressor(n_neighbors=3)
# 훈련 데이터와 타깃을 사용하여 모델을 학습시킴
reg.fit(X_train, y_train)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=3, p=2, weights='uniform')
```


© dgkim@hanyang.ac.kr

- 다음은 테스트 세트에 대해 예측을 함.

```
print("Test set predictions:\n{}", reg.predict(X_test))
```

```
Test set predictions:
[-0.054  0.357  1.137 -1.894 -1.139 -1.631  0.357  0.912 -0.447 -1.139]
```

모델 평가

- score 메서드를 사용해 위 모델을 평가할 수 있음.
- score 메서드는 회귀의 경우 결정 계수라고 하는 R^2값으로 모델을 평가함.

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

© dgkim@hanyang.ac.kr

▷ y_i: 훈련 데이터의 타깃값

▷ $\hat{y}_i$: 회귀 모델 예측의 타깃값

▷ $\bar{y}$: 훈련 데이터의 타깃값의 평균

▷ $\sum(y_i - \bar{y})^2$: 총변동(total variation)

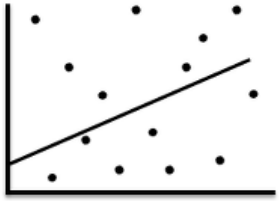
▷ $\sum(y_i - \hat{y}_i)^2$: 잔차 제곱합(total sum of square of error)

▷ $\sum(\hat{y}_i - \bar{y})^2$: 회귀변동(regression variation)

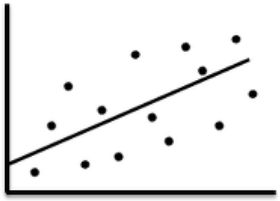
- 결정 계수의 아이디어는 회귀직선이 데이터의 특성과 타깃간의 함수관계(인과관계)를 얼마나 잘 설명하는가를 측정하는 것임.

- $0 \le R^2 \le 1$: $R^2 = 1$은 완벽한 예측인 경우이고, $R^2 = 0$은 훈련 데이터 타깃값의 평균으로 예측한 경우임.

ⓒ dgkim@hanyang.ac.kr



$R^2 = 0$



$R^2 = 0.5$

$R^2 = 1$

```
print("Test set R^2: {:.2f}".format(reg.score(X_test, y_test)))
```

```
Test set R^2: 0.83
```


© dgkim@hanyang.ac.kr

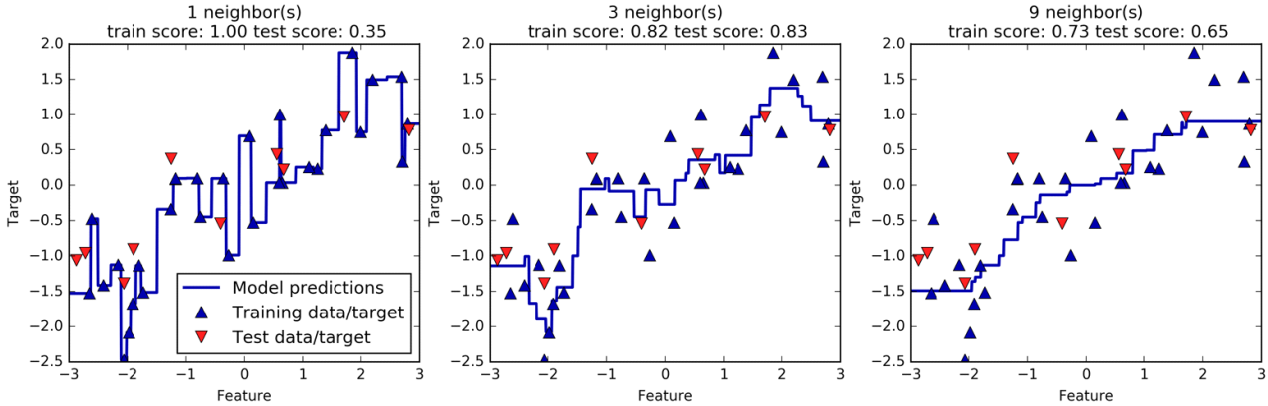
❖ KNeighborsRegressor 분석

- KNeighborsRegressor 분석을 위해 1차원 데이터셋에 많은 특성 값을 생성하여 예측을 구현함.

```
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# create 1,000 data points, evenly spaced between -3 and 3
line = np.linspace(-3, 3, 1000).reshape(-1, 1)

for n_neighbors, ax in zip([1, 3, 9], axes):
    # make predictions using 1, 3, or 9 neighbors
    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)
```


ⓒ dgkim@hanyang.ac.kr

```
    ax.set_title(
            "{} neighbor(s)\n train score: {:.2f} test score: {:.2f}".format(
                n_neighbors, reg.score(X_train, y_train),reg.score(X_test, y_test)))
    ax.set_xlabel("Feature")
    ax.set_ylabel("Target")
axes[0].legend(["Model predictions", "Training data/target",
                "Test data/target"], loc="best")
```



✓ 위 그림은 “n_neighbors” 값에 따라 최근접 이웃 회귀로 만들어진 예측 비교를 보여줌.

© dgkim@hanyang.ac.kr

❖ 장단점과 매개변수

매개변수

- KNeighbors 분류기 및 회귀에서 중요한 매개변수는 다음 두 가지임.
  1) 데이터 포인트 사이의 거리를 재는 방법 (유클리디안 거리, ℓ_p 거리, 지오데식 거리 등등)
  2) 이웃의 수(3~5개 정도가 잘 작동함).

장단점

- k-NN의 장점:
  1) 이해하기 쉬운 모델
  2) 조정할 파라미터수가 적음
  3) 복잡한 알고리즘을 적용하기 전에 초기에 시도해 볼만한 모델

ⓒ dgkim@hanyang.ac.kr

- k-NN의 단점:
  1) 훈련 세트가 매우 크면 (특성의 수나 샘플의 수가 클 경우) 예측이 느려지거나 잘 작동하지 않음. 특히, 특성 값 대부분이 0인 (희소한 특성, sparse) 데이터셋과는 잘 작동하지 않음.
  2) 데이터의 전처리가 필요함 (스케일 정규화). 이웃 간의 거리를 계산할 때 특성마다 값의 범위가 크게 다르면 범위가 작은 특성에 크게 영향을 받음.

ⓒ dgkim@hanyang.ac.kr