04_머신러닝 분류

로지스틱 회귀 (LogisticRegression)

최근접 이웃(K-Neighbors)

서포트 벡터 머신(SVM)

머신러닝 분류

• 분류(Classification)

02

> 머신러닝 지도학습의 한 유형으로 ~~레이블~~ 값을 예측한다.

> 다양한 머신러닝 알고리즘으로 구현할 수 있다.

- 데이터 균일도에 따른 규칙 기반의 결정 트리(Decision Tree)
- 서로 다른 머신러닝 알고리즘을 결합한 앙상블(Ensemble)
- 독립변수와 종속변수의 선형 관계성에 기반한 로지스틱 회귀(Logistic Regression)
- 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신(Support Vector Machine)
- 거리가 가까운 데이터를 참조하는 최근접 이웃(K-Nearest Neighbor)
- 조건부 확률을 계산하는 방법을 적용한 나이브 베이즈(Naive Bayes)

제외

• 로지스틱 회귀

이진분류에 특화된 알고리즘: 딥러닝에서도 사용

> 분류 문제(classification task)를 해결하는 가장 기본적인 머신러닝 모델

> 로지스틱 회귀라는 이름에도 불구하고 회귀가 아닌 분류에 사용

> 로지스틱 회귀 모델은 주어진 이진 목표값 y의 클래스 레이블 값이 1이 나올 확률 p(y)와 피쳐 벡터 x 사이의 관계를 모델링하는 기법

선형

$$\log \frac{p(y_i \mid x_i, w)}{1 - p(y_i \mid x_i, w)} = w_0 + w_1 x_{i,1} + \cdots + w_p x_{i,p} = \sum_{j=0}^{p} w_j x_{i,j}$$

엔트로피와 비슷함

머신러닝 분류

• 로지스틱 회귀

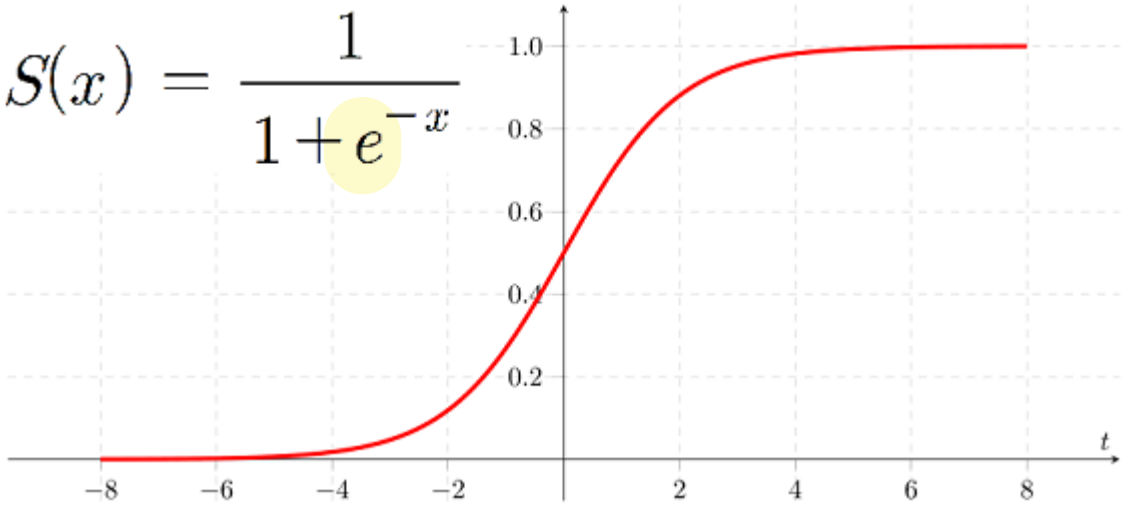
> 선형 회귀 방식을 분류에 적용한 알고리즘

> 아래와 같은 시그모이드 함수를 생성하고 이를 토대로 입력된 값이 0인지 1인지 예측하도록 수행한다.

이진의 답을 구하는 함수

소프트맥스

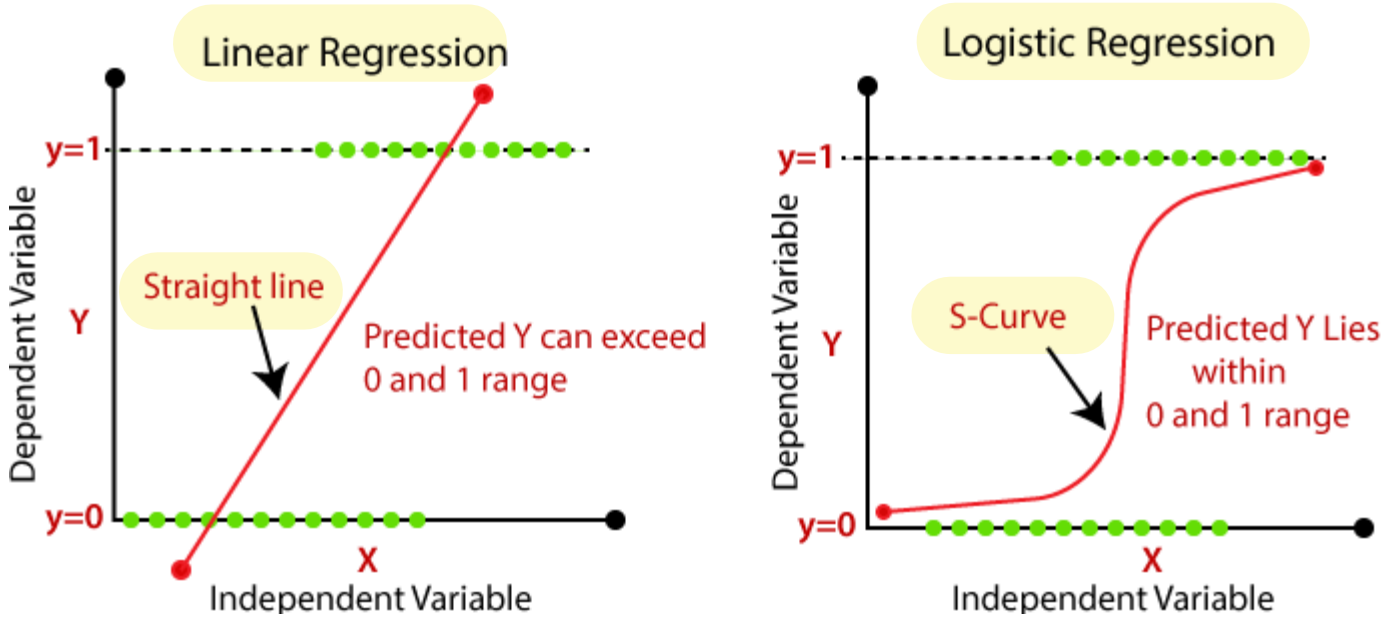
$$S(x) = \frac{1}{1+e^{-x}}$$



임곗값을 기준으로 0,1 나눔

• 로지스틱 회귀

> 선형 회귀와의 비교



• 로지스틱 회귀

> 로지스틱 회귀 실습

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score

cancer = load_breast_cancer()
scaler = StandardScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train , X_test, y_train , y_test = train_test_split(data_scaled, cancer.target,
                                                      test_size=0.3, random_state=0)
```

• 로지스틱 회귀

> 로지스틱 회귀 실습

```
lr_clf = LogisticRegression()
lr_clf.fit(X_train, y_train)
pred = lr_clf.predict(X_test)
pred_proba = lr_clf.predict_proba(X_test)[:,1]
acc = accuracy_score(y_test, pred)
auc = roc_auc_score(y_test, pred_proba)


print(f'accuracy: {acc:.3f}, roc_auc:{auc:.3f}')
```

Out : accuracy: 0.977, roc_auc:0.995

• 로지스틱 회귀

> 로지스틱 회귀 실습

```
solvers = ['lbfgs', 'liblinear', 'newton-cg', 'sag', 'saga']
    오류/ 가중치 계산방법
for solver in solvers:
    lr_clf = LogisticRegression(solver=solver, max_iter=600)
    lr_clf.fit(X_train, y_train)
    pred = lr_clf.predict(X_test)
    pred_proba = lr_clf.predict_proba(X_test)[:,1]
    acc = accuracy_score(y_test, pred)
    auc = roc_auc_score(y_test, pred_proba)

    print(f'solver: {solver}, accuracy: {acc:.3f}, roc_auc:{auc:.3f}')
```

• 로지스틱 회귀

> 로지스틱 회귀 실습

Out :
```
solver: lbfgs, accuracy: 0.977, roc_auc:0.995
solver: liblinear, accuracy: 0.982, roc_auc:0.995
solver: newton-cg, accuracy: 0.977, roc_auc:0.995
solver: sag, accuracy: 0.982, roc_auc:0.995
solver: saga, accuracy: 0.982, roc_auc:0.995
```

하이퍼파라미터	의미	특징
'newton-cg'	켤레기울기법(conjugate gradient method)	희소 데이터셋(sparse dataset)에서 추천한다.
'lbfgs'	L-BFGS-B 알고리즘	작은 크기의 데이터셋에 추천하며, 데이터의 크기가 크다면 성능이 떨어질 수 있다.
'liblinear'	최적화된 좌표 하강법(coordinate descent) 알고리즘	좌표 하강법은 4장에서 자세히 다루도록 한다. 데이터의 크기가 작을 때 추천한다.
'sag'	SAG(stochastic average gradient descent) 알고리즘 경사하강법 이용하여 계산	샘플 개수와 피처 개수가 모두 큰 대형 데이터셋에서 빠르게 수렴한다. 하지만 빠른 수렴은 각 피처의 스케일이 비슷할 때만 보장되므로 sklearn.preprocessing 모듈의 클래스를 사용해 데이터를 전처리하는 것이 좋다.
'saga'	SAG 알고리즘의 변형	'sag'와 마찬가지로 대형 데이터셋에서 잘 동작하고, 피처 스케일에 대한 정규화를 선행하는 것이 좋다.

default

선형모델의 알고리즘 사용

대형데이터에서만 작동

• 로지스틱 회귀

> 로지스틱 회귀 실습

```
from sklearn.model_selection import GridSearchCV

params={'solver':['liblinear', 'lbfgs'],
        'penalty':['l2', 'l1'],   규제
        'C':[0.01, 0.1, 1, 1, 5, 10]}  패널티 강도

lr_clf = LogisticRegression()
grid_clf = GridSearchCV(lr_clf, param_grid=params, scoring='accuracy', cv=3 )
grid_clf.fit(data_scaled, cancer.target)

print(f'최적 하이퍼 파라미터:{grid_clf.best_params_}')
print(f'최대 평균 정확도:{grid_clf.best_score_:.3f}')
```

• 로지스틱 회귀

> 로지스틱 회귀 실습

Out : 최적 하이퍼 파라미터:{'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear ' }
최대 평균 정확도:0.979

하이퍼파라미터	주요값	기본값	의미
penalty	'l1', 'l2', 'elasticnet', 'none'	'l2'	규제 페널티 선택 - 'l1': L1 규제 적용 - 'l2': L2 규제 적용 - 'elasticnet': 엘라스틱 넷 규제 적용 - 'none': 규제 미부여
C	float>0	1.0	규제 페널티 크기의 역수

강도

기능	'liblinear'	'lbfgs'	'newton-cg'	'sag'	'saga'
다항 분류 + L2 규제	X	O	O	O	O
OVR + L2 규제	O	O	O	O	O
다항 분류 + L1 규제	X	X	X	X	O
OVR + L1 규제	O	X	X	X	O
엘라스틱 넷 규제	X	X	X	X	O
규제 미부여 기능	X	O	O	O	O

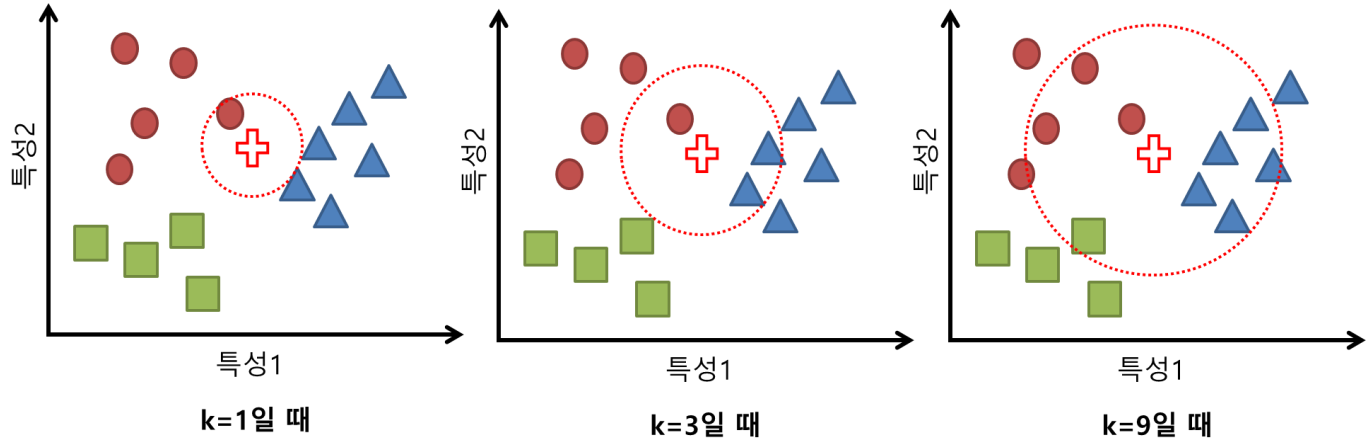
• 최근접 이웃(K-Neighbors)

> 데이터의 산포도를 기준으로 가까운 데이터를 같은 분류로 판단하여 학습하고 예측한다.

> 데이터의 분포만으로 결정을 하기에 간단하고 빠르게 분류한다.

> k라는 파라미터를 조절하면 결과가 달라진다.

차원축소 하고 분류



• 최근접 이웃(K-Neighbors)

> 최근접 이웃 실습

```
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score

cancer = load_breast_cancer()
scaler = StandardScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train , X_test, y_train , y_test = train_test_split(data_scaled, cancer.target,
                                                      test_size=0.3, random_state=0)
```

• 최근접 이웃(K-Neighbors)

> 최근접 이웃 실습

```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
pred_proba = knn.predict_proba(X_test)[:,1]
acc = accuracy_score(y_test, pred)
auc = roc_auc_score(y_test, pred_proba)


print(f'accuracy: {acc:.3f}, roc_auc:{auc:.3f}')
```

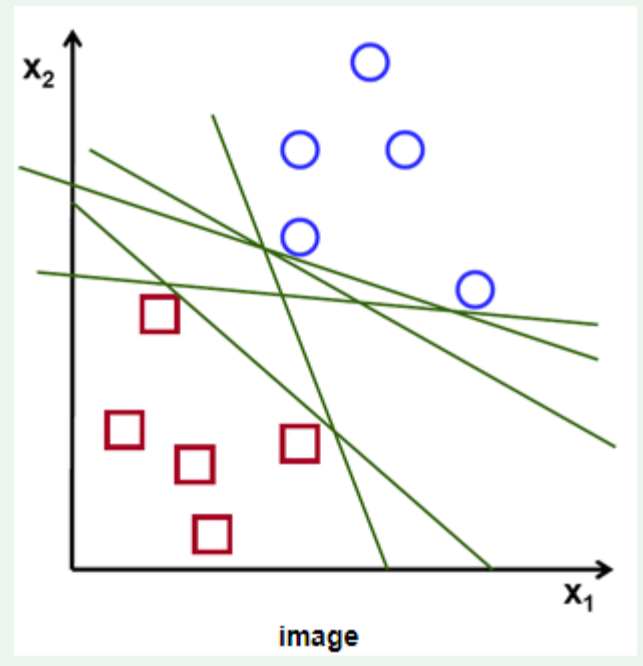
Out : accuracy: 0.953, roc_auc:0.992

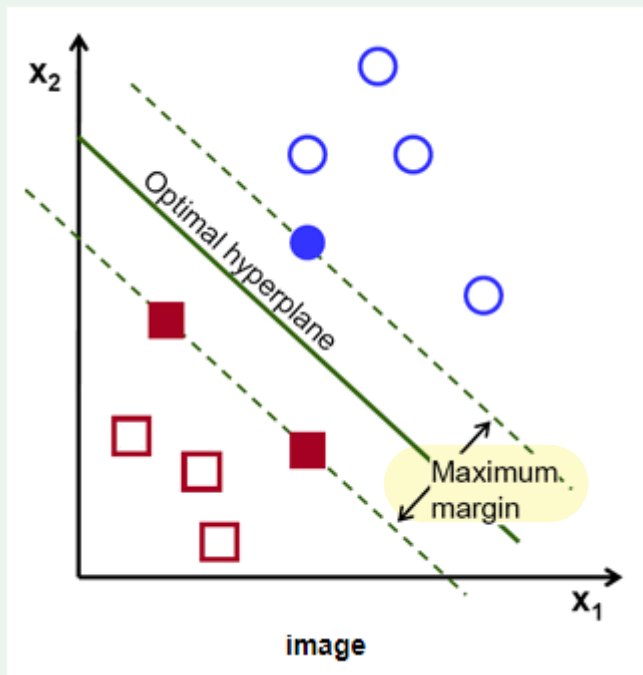
- 서포트 백터 머신(SVM)

분류 알고리즘이나 회귀도 가능

> SVM은 기본적으로 분류를 위한 기준 선을 정의하여 두 개의 데이터를 분리하는 방법으로 데이터들과 거리가 가장 먼(Margin 극대화) 초평면을 선택하여 분리하는 방법이다.

선을 구하는 것이 목표임

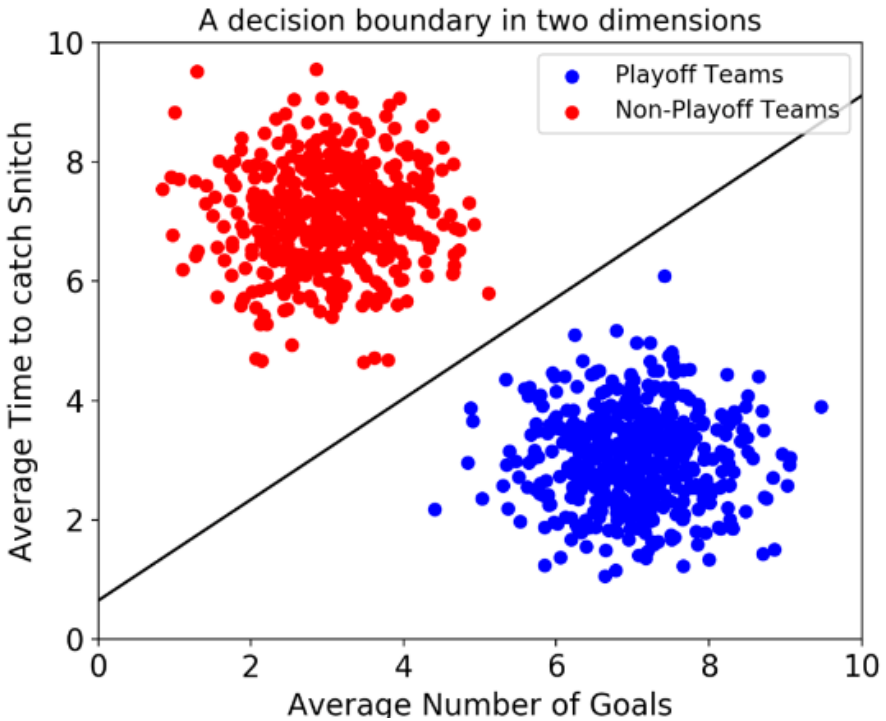


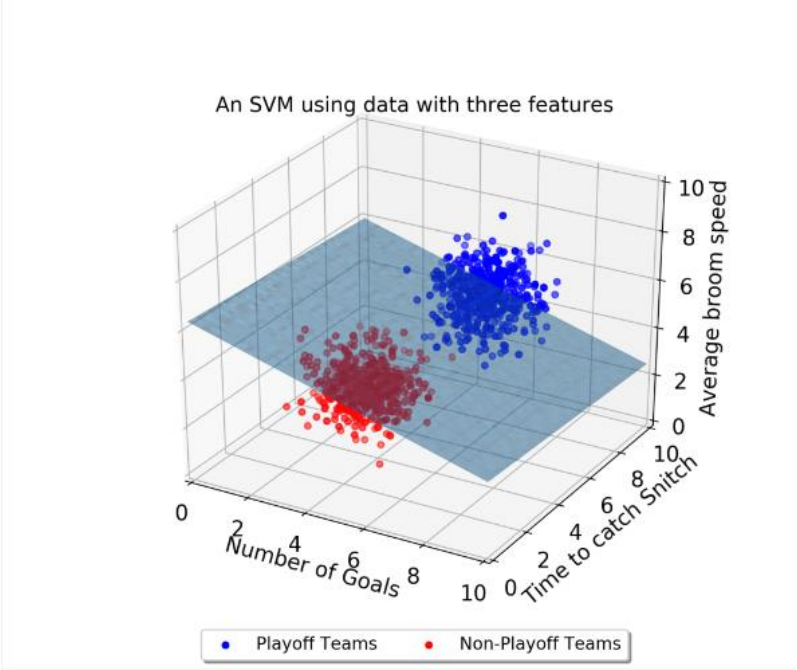


• 서포트 백터 머신(SVM)

> 일반적으로 선으로 보일 순 있으나 평면을 만들어 분리한다.

> 이를 인지할 수 있는 범위는 3차원 까지이다.(최대 독립변수 3개, 종속변수 1개)





• 서포트 백터 머신(SVM)

> SVM 실습

```
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.svm import SVC
cancer = load_breast_cancer()
scaler = StandardScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train , X_test, y_train , y_test = train_test_split(data_scaled, cancer.target,
                                                      test_size=0.3, random_state=0)
```

• 서포트 백터 머신(SVM)

> SVM 실습

```
svc = SVC(probability = True) 확률계산 : predict_proba를 구할 경우 설정해야
svc.fit(X_train, y_train)
pred = svc.predict(X_test)
pred_proba = svc.predict_proba(X_test)[:,1]
acc = accuracy_score(y_test, pred)
auc = roc_auc_score(y_test, pred_proba)


print(f'accuracy: {acc:.3f}, roc_auc:{auc:.3f}')
```

Out : accuracy: 0.977, roc_auc:0.998

• 서포트 백터 머신(SVM)

> 커널 종류

- 사용하는 커널에 따라 초평면의 모양이 달라지고 하이퍼 파라미터가 조금 차이난다..

커널 명칭	형태	비고
선형 커널linear kernel	$K(x_1, x_2) = x_1^T x_2$	커널을 정의하지 않는 선형 SVM와 동일하다.
다항 커널polynomial kernel	$K(x_1, x_2) = (\gamma x_1^T x_2 + r)^d$	계수coefficient에 해당하는 γ, 절편intercept에 해당하는 r, 차수degree에 해당하는 d를 가진다. 선형 커널을 확장한 개념으로, γ=1, r=0, d=1일 때 선형 커널과 같다.
RBF 커널radial basis function kernel	$K(x_1, x_2) = e^{-\gamma\|x_1 - x_2\|^2}$	분산의 역수에 해당하는 계수 γ를 가진다. 가우시안 분포Gaussian distribution와 비슷한 형태이므로 가우시안 커널Gaussian kernel이라고도 한다. SVM 모델에서 가장 많이 사용하는 커널 함수의 하나이다.
시그모이드 커널sigmoid kernel	$K(x_1, x_2) = \tanh(\gamma x_1^T x_2 + r)$	γ는 함수의 기울기를 조정하는 계수이며, r은 절편에 해당한다.

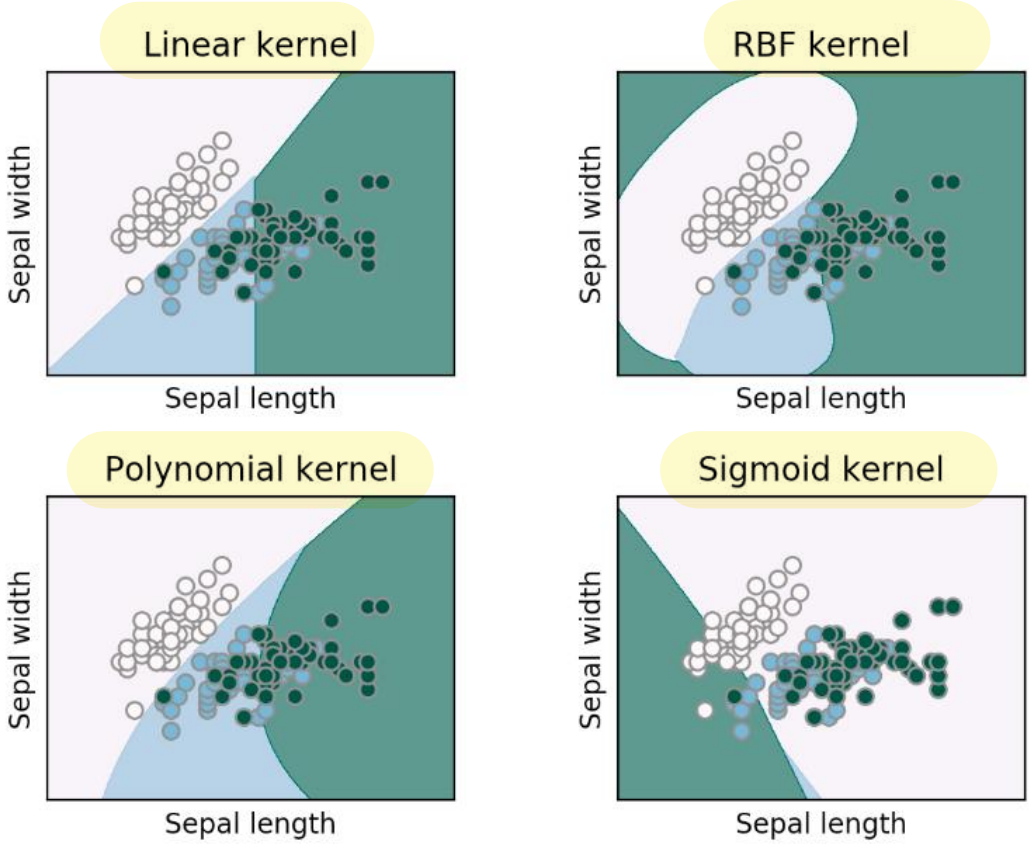
y=ax+b

비선형

• 서포트 백터 머신(SVM)

> 커널 종류

- 사용하는 커널에 따라 초평면의 모양이 달라지고 하이퍼 파라미터가 조금 차이난다..



• 서포트 백터 머신(SVM)

> 하이퍼 파라미터 종류

하이퍼파라미터	주요값	기본값	의미
C	float>0	1.0	규제 강도의 역수
kernel	'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'	'rbf'	사용할 커널의 종류. 사용자 지정 함수를 사용할 수 있다.
probability	bool	False	확률적 추정을 사용할지의 여부. 교차검증법cross validation을 사용하므로 속도가 떨어질 수 있다.
tol	float>0	1e-3	정지 조건에 대한 허용 오차
class_weight	None, 'balanced', dict	None	클래스 가중치 설정. 클래스 불균형class imbalance 효과를 줄이고자 할 때 유용하다. - None: 모든 클래스에 같은 가중치 부여 - 'balanced': 클래스 빈도에 반비례하는 가중치 설정 - dict: 사용자 지정 가중치 부여
max_iter	int>0 또는 int=-1	-1	이터레이션의 상한선 - max_iter=-1: 상한선 없음. tol에 의한 정지 조건에 도달할 때까지 계속 학습한다.
decision_function_shape	'ovr', 'ovo'	'ovr'	다중 클래스일 때 반환할 결정 함수의 종류 설정. 이진 분류라면 무시한다. - 'ovr': OVRone-vs-rest 결정 함수 - 'ovo': libsvm의 OVOone-vs-one 결정 함수
random_state	None, int	None	probability=True에 한해 데이터 셔플링data shuffling에 대한 랜덤성을 제어

특정 커널사용시에만 필요

degree: kernel='poly'일 때 설정할 수 있으며 차수

gamma : kernel이 'rbf', 'poly', 'sigmoid'일 때 커널 계수
선의 휘어짐의 정도

coef0 : kernel이 'poly'나 'sigmoid'일 때 설정할 수 있으며 커널 함수의 절편

• 서포트 백터 머신(SVM)

> 특징

- 두 집단을 잘 분리하기위해
  경계점(Suport Vector)를 만들어야하는데
  다음과 같이 이상치에 의해 마진이 작다.

- 다른 데이터에 비해 이상치에 민감하고
  스케일링 유무의 차이가 심하게 난다.