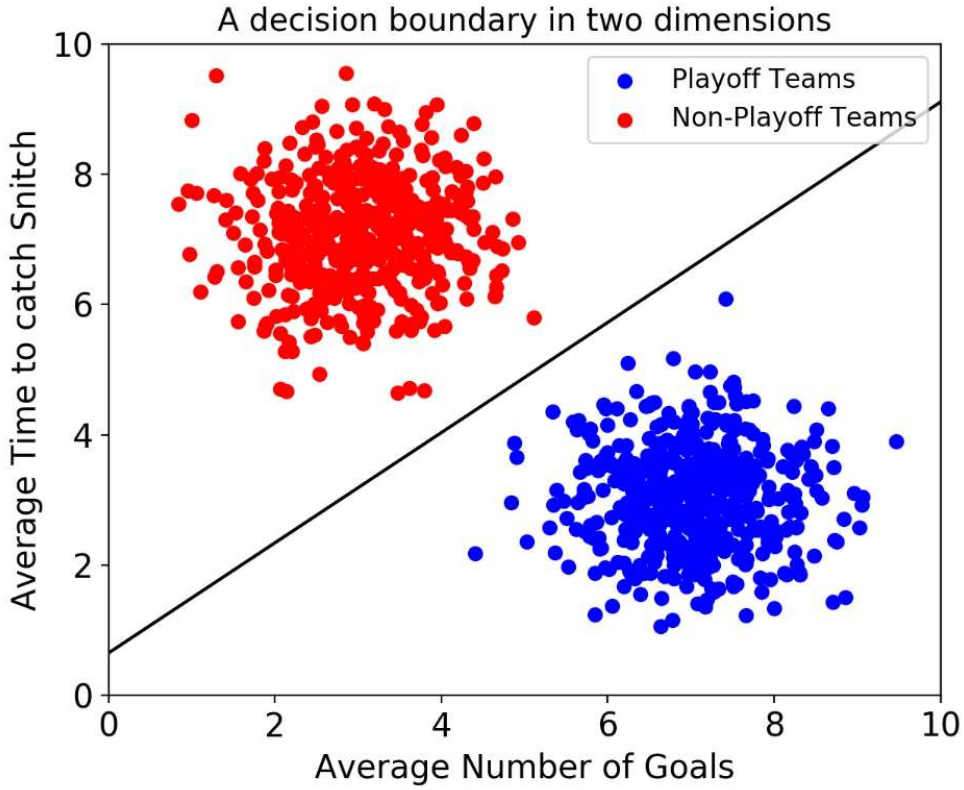
SVM

➢ **서포트 벡터 머신(SVM: Support Vector Machine)**

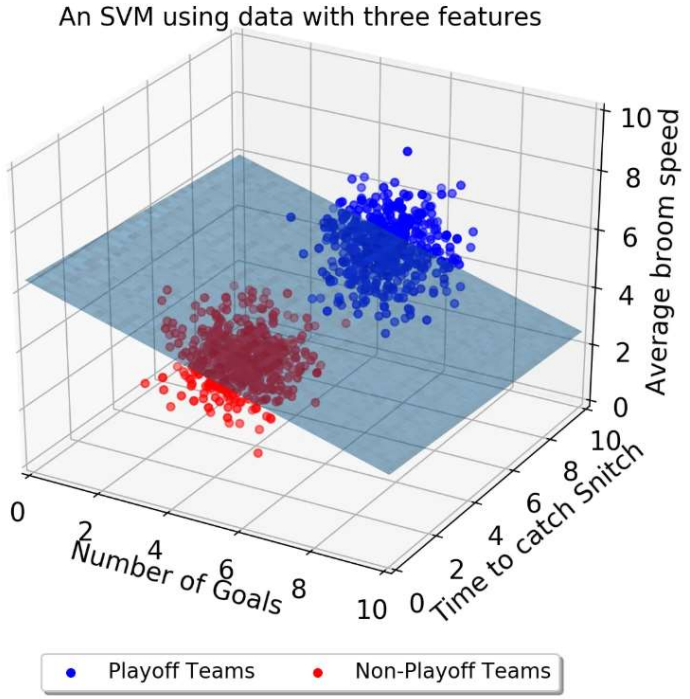
- 분류 과제에 사용할 수 있는 강력한 머신러닝 지도학습 모델
- 결정 경계(Decision Boundary), 즉 분류를 위한 기준 선을 정의하는 모델
- 분류되지 않은 새로운 점이 나타나면 경계의 어느 쪽에 속하는지 확인해서 분류 과제를 수행할 수 있게 된다.



데이터에 2개 속성(feature)만 있다면 결정 경계는 간단한 선 형태가 될 것이다.

➢ 서포트 벡터 머신(SVM: Support Vector Machine)

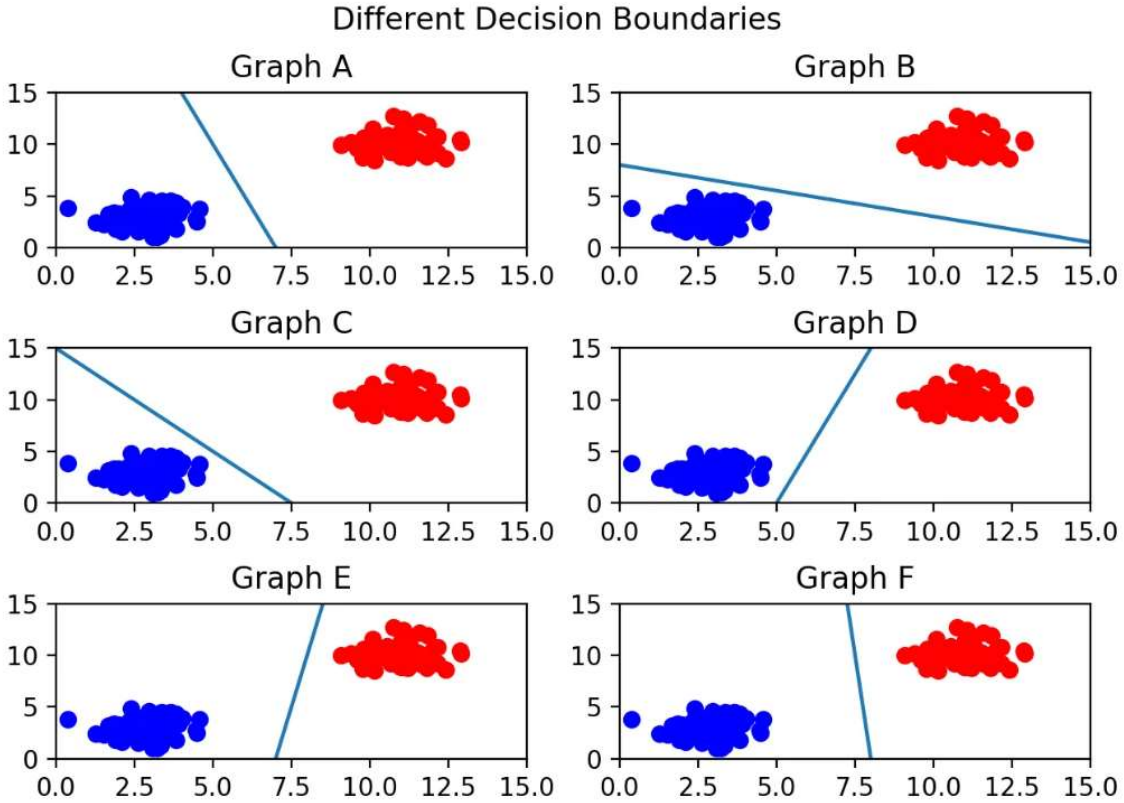
- 속성의 개수가 늘어날수록 결정 경계도 단순한 평면이 아닌 고차원이 되고 이를 "초평면(hyperplane)"이라고 부른다.



속성이 3개로 늘어난다면 이렇게 3차원으로 그려야 한다.
결정 경계는 '선'이 아닌 '평면'이 된다.

➢ **서포트 벡터 머신(SVM: Support Vector Machine)**

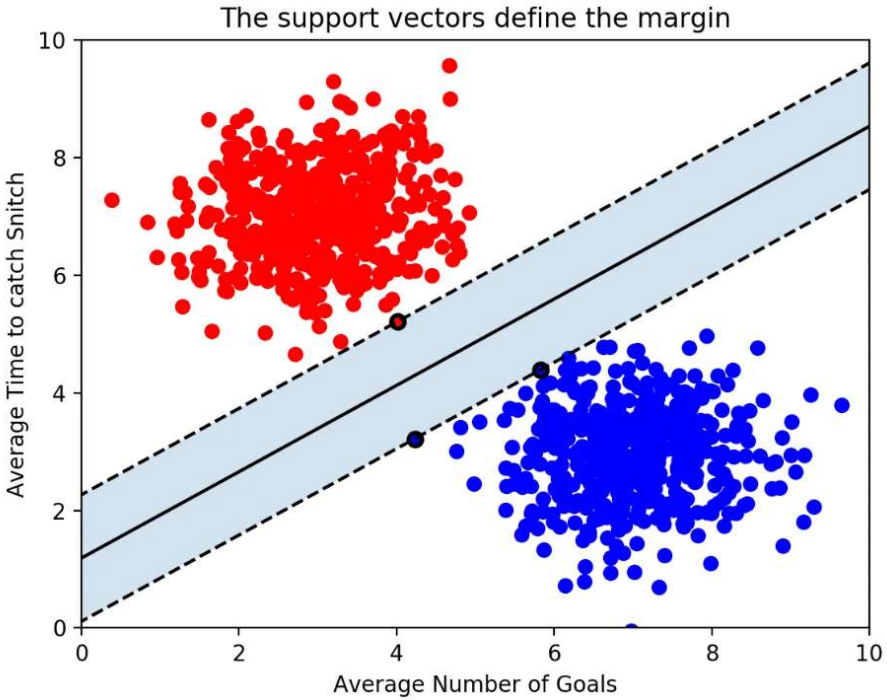
- 최적의 결정 경계(Decision Boundary) - 결정 경계는 데이터 군으로부터 최대한 멀리 떨어지는 게 좋다
- Support Vectors는 결정 경계와 가까이 있는 데이터 포인트들을 의미한다. 이 데이터들이 경계를 정의하는 결정적인 역할을 한다



C의 결정 경계는 파란색 부류와 너무 가까워서 아슬아슬해보인다.
F의 결정 경계는 두 클래스(분류) 사이에서 거리가 가장 멀기 때문에 가장 적절해 보인다

➢ **서포트 벡터 머신(SVM: Support Vector Machine)**

- 마진(Margin)은 결정 경계와 서포트 벡터 사이의 거리를 의미한다.
- 최적의 결정 경계는 마진을 최대화한다.
- x축과 y축 2개의 속성을 가진 데이터로 결정 경계는 총 3개의 데이터 포인트(서포트 벡터)가 필요하다
- n개의 속성을 가진 데이터에는 최소 n+1개의 서포트 벡터가 존재한다

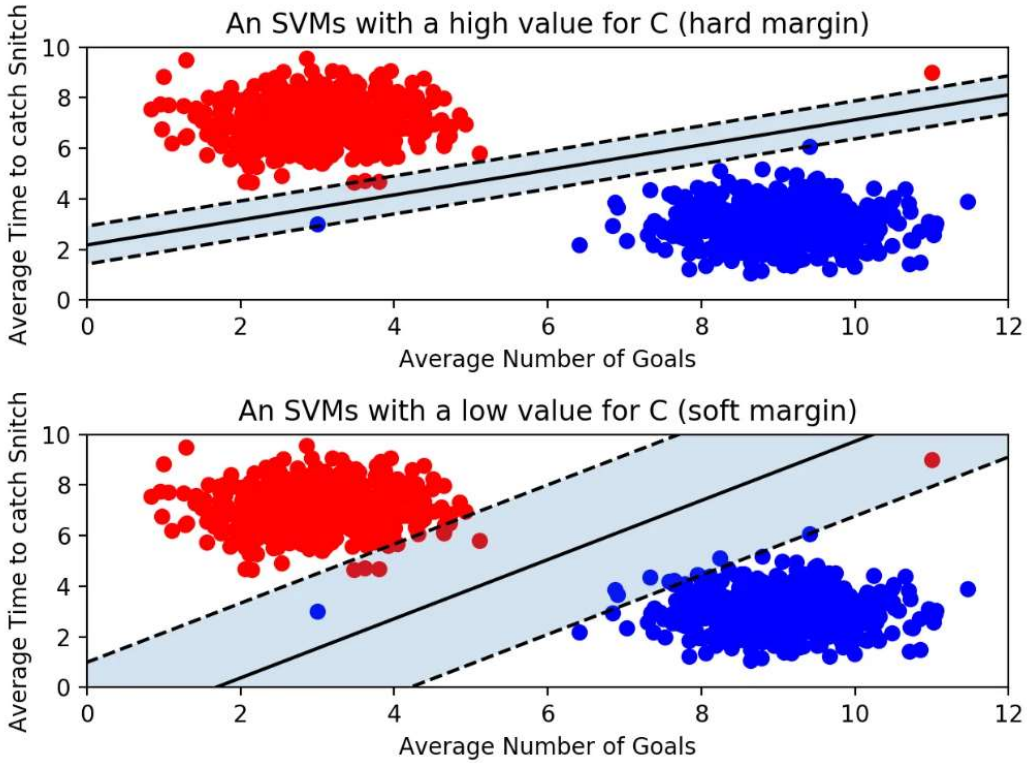


가운데 실선이 하나 그어져있는데, 이게 바로 '결정 경계'가 되겠다. 그리고 그 실선으로부터 검은 테두리가 있는 빨간점 1개, 파란점 2개까지 영역을 두고 점선을 그어놓았다. 점선으로부터 결정 경계까지의 거리가 바로 '마진(margin)'이다.

SVM에서는 결정 경계를 정의하는 게 결국 서포트 벡터이기 때문에 데이터 포인트 중에서 서포트 벡터만 잘 골라내면 나머지 쓸 데 없는 수많은 데이터 포인트들을 무시할 수 있다. 그래서 매우 빠르다.

➢ 서포트 벡터 머신(SVM: Support Vector Machine)

- 포트 벡터 결정 경계를 정의하는 서포트 벡터를 확인하려면 classifier.support_vectors_를 출력해본다
- scikit-learn에서는 SVM 모델이 오류를 어느정도 허용할 것인지 파라미터 C를 통해 지정할 수 있다.
- C값이 클수록 하드마진(오류 허용 안 함), 작을수록 소프트마진(오류를 허용함)이다.

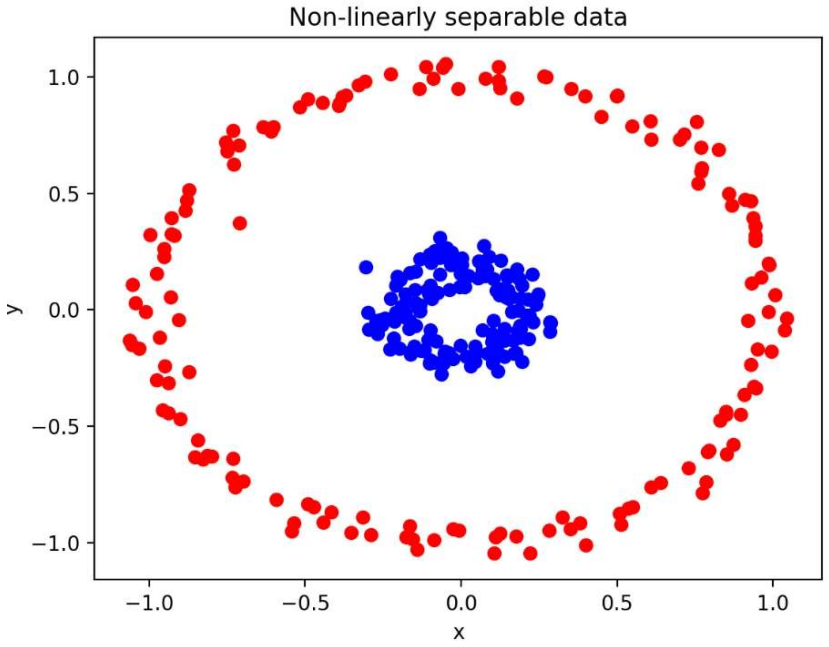


하드 마진(hard margin) -아웃라이어를 허용하지 않고 기준을 까다롭게 세운 모양
서포트 벡터와 결정 경계 사이의 거리가 매우 좁다. 즉, 마진이 매우 작아진다.
이렇게 개별적인 학습 데이터들을 다 놓치지 않으려고 아웃라이어를 허용하지 않는 기준으로 결정 경계를 정해버리면 오버피팅(overfitting) 문제가 발생할 수 있다.

소프트 마진(soft margin) - 아웃라이어들이 마진 안에 어느정도 포함되도록 기준을 잡은 모양
서포트 벡터와 결정 경계 사이의 거리가 멀어졌다. 즉, 마진이 커진다.
언더피팅(underfitting) 문제가 발생할 수 있다.

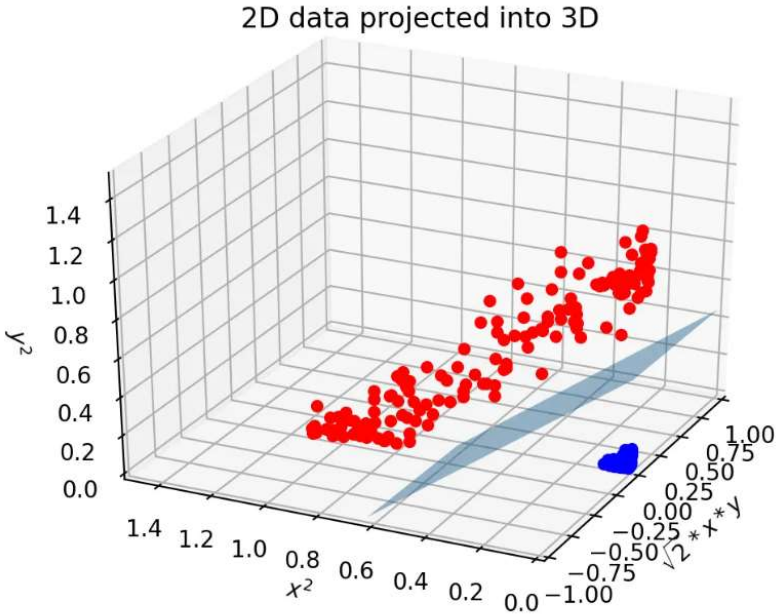
➢ 서포트 벡터 머신(SVM: Support Vector Machine)

- SVM이 선형으로 분리 할 수 없는 데이터 세트에 SVM 모델을 만들 때 kernel을 지정하여 해결할 수 있다.
- 일부 아웃라이어에 맞추기 위해 비선형으로 결정 경계를 만들 필요가 없다
- 다항식(polynomial) 커널을 사용하면 2차원에서 x, y 좌표로 이루어진 점들을 따라 3차원으로 표현하게 된다.



$$(x,\ y) \rightarrow (\sqrt{2} \cdot x \cdot y,\ x^2,\ y^2)$$

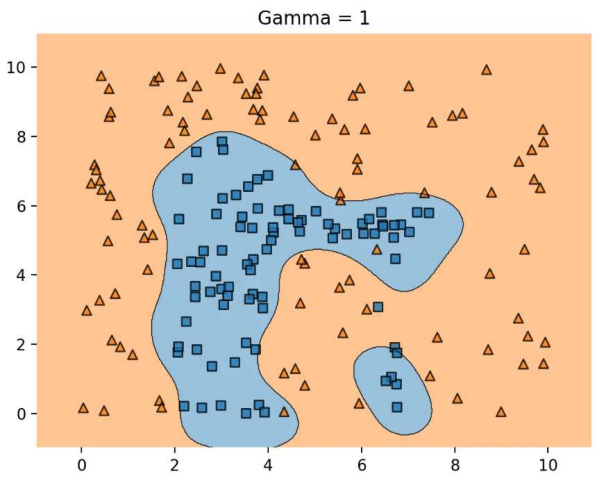
$$(1,\ 2) \rightarrow (2\sqrt{2},\ 1,\ 4)$$

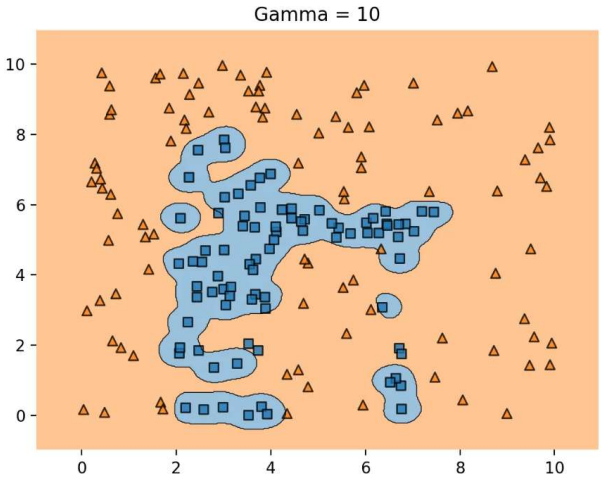


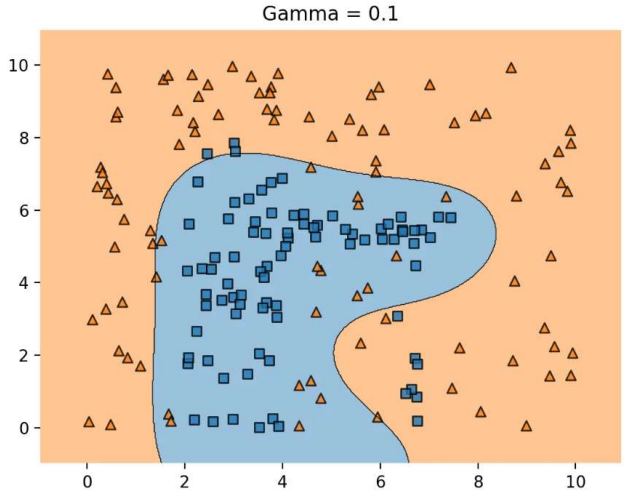
다항식(polynomial) 커널을 사용하면 데이터를 더 높은 차원으로 변형하여 나타냄으로써 초평면(hyperplane)의 결정 경계를 얻을 수 있다.

➢ 서포트 벡터 머신(SVM: Support Vector Machine)

- 방사 기저 함수 (RBF: Radial Bias Function를 RBF 커널 혹은 가우시안 커널이라고 부르기도 한다
- RBF 커널은 2차원의 점을 무한한 차원의 점으로 변환한다
- gamma는 결정 경계를 얼마나 유연하게 그을 것인지 정해주는 파라미터이다
- 학습 데이터에 얼마나 민감하게 반응할 것인지 모델을 조정
- gamma값을 높이면 학습 데이터에 많이 의존해서 결정 경계를 구불구불 긋게 된다. 이는 오버피팅을 초래할 수 있다.
- 반대로 gamma를 낮추면 학습 데이터에 별로 의존하지 않고 결정 경계를 직선에 가깝게 긋게 된다. 이러면 언더피팅이 발생할 수 있다.





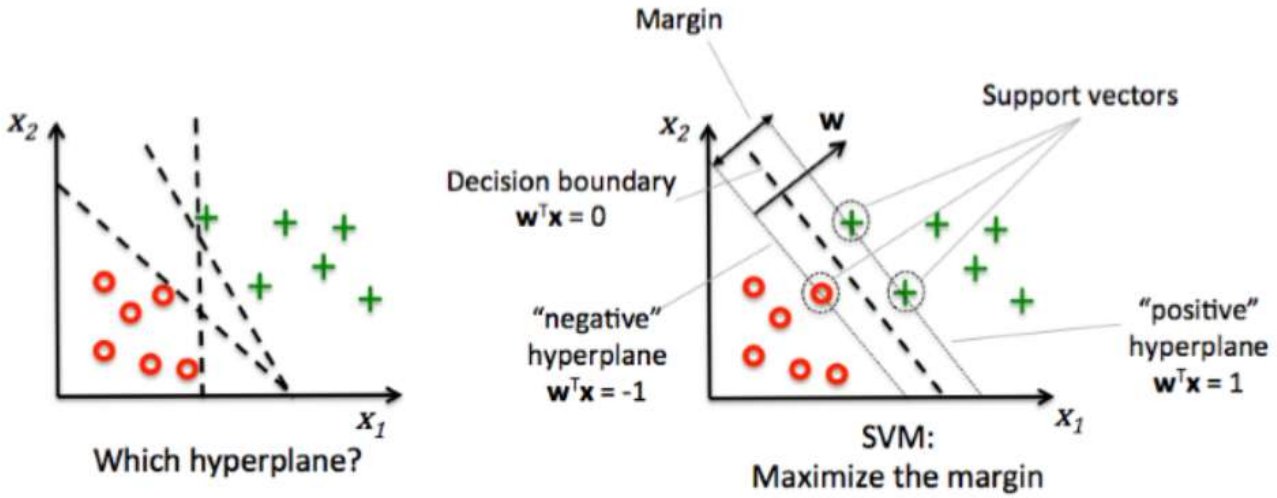


➢ 서포트 벡터 머신(SVM: Support Vector Machine)

- SVM은 분류에 사용되는 지도학습 머신러닝 모델이다.
- SVM은 서포트 벡터(support vectors)를 사용해서 결정 경계(Decision Boundary)를 정의하고, 분류되지 않은 점을 해당 결정 경계와 비교해서 분류한다.
- 서포트 벡터(support vectors)는 결정 경계에 가장 가까운 각 클래스의 점들이다.
- 서포트 벡터와 결정 경계 사이의 거리를 마진(margin)이라고 한다.
- SVM은 허용 가능한 오류 범위 내에서 가능한 최대 마진을 만들려고 한다.
- 파라미터 C는 허용되는 오류 양을 조절한다. C 값이 클수록 오류를 덜 허용하며 이를 하드 마진(hard margin)이라 부른다. 반대로 C 값이 작을수록 오류를 더 많이 허용해서 소프트 마진(soft margin)을 만든다.
- SVM에서는 선형으로 분리할 수 없는 점들을 분류하기 위해 커널(kernel)을 사용한다.
- 커널(kernel)은 원래 가지고 있는 데이터를 더 높은 차원의 데이터로 변환한다. 2차원의 점으로 나타낼 수 있는 데이터를 다항식(polynomial) 커널은 3차원으로, RBF 커널은 점을 무한한 차원으로 변환한다.
- RBF 커널에는 파라미터 감마(gamma)가 있다. 감마가 너무 크면 학습 데이터에 너무 의존해서 오버피팅이 발생할 수 있다.

SVM(Support Vector Machine)

- SVM(Support Vector Machine)
  - 로지스틱 회귀와 함께 분류를 위한 강력한 머신러닝 알고리즘
  - 퍼셉트론의 개념을 확장하여 적용한 알고리즘
  - 퍼셉트론이 분류 오류를 최소화하는 알고리즘인 반면 SVM은 margin을 최대가 되도록 하는 알고리즘
  - margin은 분류를 위한 경계선과 이 경계선에 가장 가까운 트레이닝 데이터 사이의 거리를 의미
  - 경계선에 가장 가까운운 트레이닝 데이터들을 support vector라고 부릅니다.

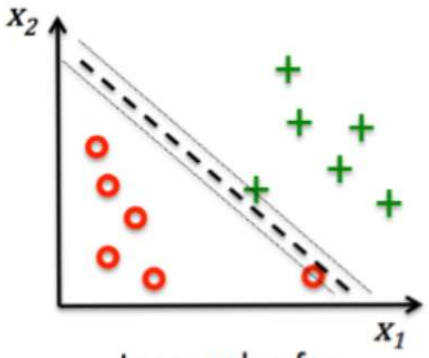


  - 빨간색 원으로 표시된 집단과 초록색 더하기 기호로 표시된 집단을 분류하는 경계선은 다양하게 존재할 수 있습니다.
  - SVM은 두 집단을 분류하는 경계선 중 support vector와의 거리가 가장 멀리 떨어져 있는 경계선을 찾아내는 알고리즘입니다.

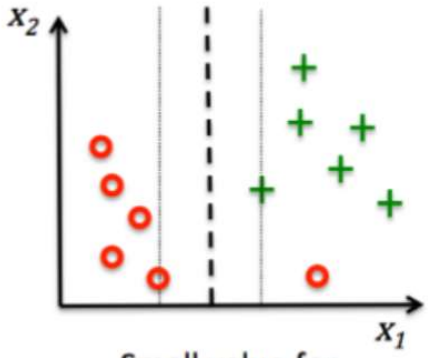
SVM(Support Vector Machine)

- SVM(Support Vector Machine)
  - SVM 알고리즘은 경계선에 가장 가까이 있는 support vector를 지나는 선과 거리가 최대가 되는 경계선을 구함으로써 그 목적을 달성하게 됩니다.
  - 경계선은 다차원 공간의 초평면입니다.
  - SVM이 퍼셉트론의 개념으로부터 출발하고 확장된 것이므로 입력되는 트레이닝 데이터, 입력값과 곱해지는 가중치와 관련됩니다.
  - j는 트레이닝 데이터의 특성값의 개수가 그 범위가 되며, i는 트레이닝 데이터의 개수가 그 범위가 됩니다.
  - C는 로지스틱 회귀에서 설명했던 정규화와 관련된 상수와 비슷한 개념이며 ξ는 여유 변수(slack variables)라고 부르는 변수는 분류 오류가 생기는 데이터에 적절한 패널티를 부여함으로써 비선형적으로 분리되는 모델을 완화시켜 최적화된 수렴값을 가지도록 하기 위한 선형 제약(linear contraints)값입니다.
  - C값을 변화시키면 알고리즘이 결정하는 경계선이 달라집니다.

$$\frac{1}{2}\sum_{j} w_j^2 + C\sum_{i} \xi^{(i)}$$



Large value for parameter *C*



Small value for parameter *C*

SVM(Support Vector Machine)

- 선형 분류기 훈련
  - 초평면은 n차원 공간에 있는 n-1 부분 공간으로 정의합니다.
  - SVM은 훈련 데이터를 분류하기 위해 클래스 사이의 마진을 최대화하는 초평면을 찾습니다.
  - 사이킷런의 LinearSVC는 클래스 사이 마진이 최대화되는 초평면을 찾습니다. (초평면은 새로운 샘플을 분류하기 위한 결정 경계입니다.)
  - 초평면의 마진을 최대화하는 SVC와 분류 오차를 최소화하는 것 사이에 균형을 잡아야 합니다.
  - SVC 모델의 매개변수 C는 잘못 분류된 데이터 포인트에 부여하는 페널티입니다.
  - C가 작으면 분류기는 잘못 분류된 데이터 포인트를 허용합니다
  - C가 크면 분류기는 잘못 분류된 데이터에 큰 페널티를 부과합니다.

```
from sklearn.svm import LinearSVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np

iris = datasets.load_iris()  # 데이터  로드
features = iris.data[:100,:2]  #두 개의 클래스와 두 개의 특성만 선택
target = iris.target[:100]

scaler = StandardScaler()  # 특성 표준화
features_standardized = scaler.fit_transform(features)
svc = LinearSVC(C=1.0)  # 서포트 벡터 분류기 생성
model = svc.fit(features_standardized, target)  # 모델 훈련
```

SVM(Support Vector Machine)

- 선형 분류기 훈련
  - LinearSVC 클래스는 예측 확률을 제공하지는 않지만 decision_function()를 사용해 분류에 대한 신뢰도를 확인할 수 있습니다.

```
from matplotlib import pyplot as plt

# 클래스를 색으로 구분한 산점도를 그립니다.
color = ["black" if c == 0 else "lightgrey" for c in target]
plt.scatter(features_standardized[:,0], features_standardized[:,1], c=color)

w = svc.coef_[0]   # 초평면을 만듭니다.
a = -w[0] / w[1]
xx = np.linspace(-2.5, 2.5)
yy = a * xx - (svc.intercept_[0]) / w[1]

plt.plot(xx, yy)  # 초평면을 그립니다.
plt.axis("off"), plt.show();

new_observation = [[ -2,  3]]   # 새로운 샘플을 만듭니다.

svc.predict(new_observation)   # 새로운 샘플의 클래스를 예측
svc.decision_function(new_observation)
```

SVM(Support Vector Machine)

- 커널을 사용해 선형적으로 구분되지 않는 클래스 다루기
  - 클래스가 선형적으로 구분되지 않을 때 비선형 결정 경계를 만들기 위해 커널 함수를 사용한 서포트 벡터 머신으로 훈련합니다.

```
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np

np.random.seed(0)  # 랜덤 시드를 지정
features = np.random.randn(200, 2)  # 두 개의 특성을 만듭니다.

# XOR 연산(이것이 무엇인지 알 필요는 없습니다)을 사용하여
# 선형적으로 구분할 수 없는 클래스를 만듭니다.
target_xor = np.logical_xor(features[:, 0] > 0, features[:, 1] > 0)
target = np.where(target_xor, 0, 1)

# 방사 기저 함수 커널을 사용한 서포트 벡터 머신을 만듭니다.
svc = SVC(kernel="rbf", random_state=0, gamma=1, C=1)

model = svc.fit(features, target)  # 분류기 훈련
```

SVM(Support Vector Machine)

- 커널을 사용해 선형적으로 구분되지 않는 클래스 다루기

```
# 샘플과 결정 경계를 그립니다.
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

def plot_decision_regions(X, y, classifier):
    cmap = ListedColormap(("red", "blue"))
    xx1, xx2 = np.meshgrid(np.arange(-3, 3, 0.02), np.arange(-3, 3, 0.02))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.1, cmap=cmap)

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap.colors[idx],
                    marker="+", label=cl)

# 선형 커널을 사용한 서포트 벡터 분류기를 만듭니다.
svc_linear = SVC(kernel="linear", random_state=0, C=1)
svc_linear.fit(features, target) # 모델 훈련
plot_decision_regions(features, target, classifier=svc_linear) # 샘플과 초평면을 그립니다.
plt.axis("off"), plt.show();
```

SVM(Support Vector Machine)

- 예측 확률 계산
  - 사이킷런의 SVC 클래스를 사용할 때 probability=True로 지정하여 모델을 훈련하면 predict_proba()에서 보정된 확률을 확인할 수 있습니다.

```
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np

iris = datasets.load_iris()  # 데이터 로드
features = iris.data
target = iris.target

scaler = StandardScaler()  # 특성 표준화
features_standardized = scaler.fit_transform(features)

# 서포트 벡터 분류기 객체 생성
svc = SVC(kernel="linear", probability=True, random_state=0)
model = svc.fit(features_standardized, target)  # 분류기  훈련
new_observation = [[.4, .4, .4, .4]]  #New Sample Data
model.predict_proba(new_observation)  # 예측 확률 확인
```

SVM(Support Vector Machine)

- 서포트 벡터 식별
  - 사이킷런의 SVC를 훈련하고 support_vectors_를 사용하여 모델에 있는 서포트 벡터를 식별할 수 있습니다.
  - support_ 속성을 사용하여 서포트 벡터의 인덱스를 확인할 수 있습니다.

```
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np

iris = datasets.load_iris() #데이터 로드
features = iris.data[:100,:]    #두 개의 클래스만 선택
target = iris.target[:100]

scaler = StandardScaler() # 특성을 표준화
features_standardized = scaler.fit_transform(features)

svc = SVC(kernel="linear", random_state=0) # 서포트 벡터 분류기 객체 생성
model = svc.fit(features_standardized, target) # 분류기 훈련

model.support_vectors_ # 서포트 벡터를 확인
model.support_
```

SVM(Support Vector Machine)

- 불균형한 클래스 다루기
  - class_weight 매개변수를 사용하여 작은 클래스를 잘못 분류했을 때 페널티를 증가시킵니다.

```
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np

iris = datasets.load_iris()  #데이터 로드
features = iris.data[:100,:]   #두 개의 클래스만 선택
target = iris.target[:100]
features = features[40:,:]  # 처음 40개 샘플을 제거
target = target[40:]          #불균형한 클래스를 만듭니다.

# 타깃 벡터에서 0이 아닌 클래스는 모두 1로 만듭니다.
target = np.where((target == 0), 0, 1)
scaler = StandardScaler()  # 특성을 표준화
features_standardized = scaler.fit_transform(features)

svc = SVC(kernel="linear", class_weight="balanced", C=1.0, random_state=0)
model = svc.fit(features_standardized, target)  # 분류기 훈련
```