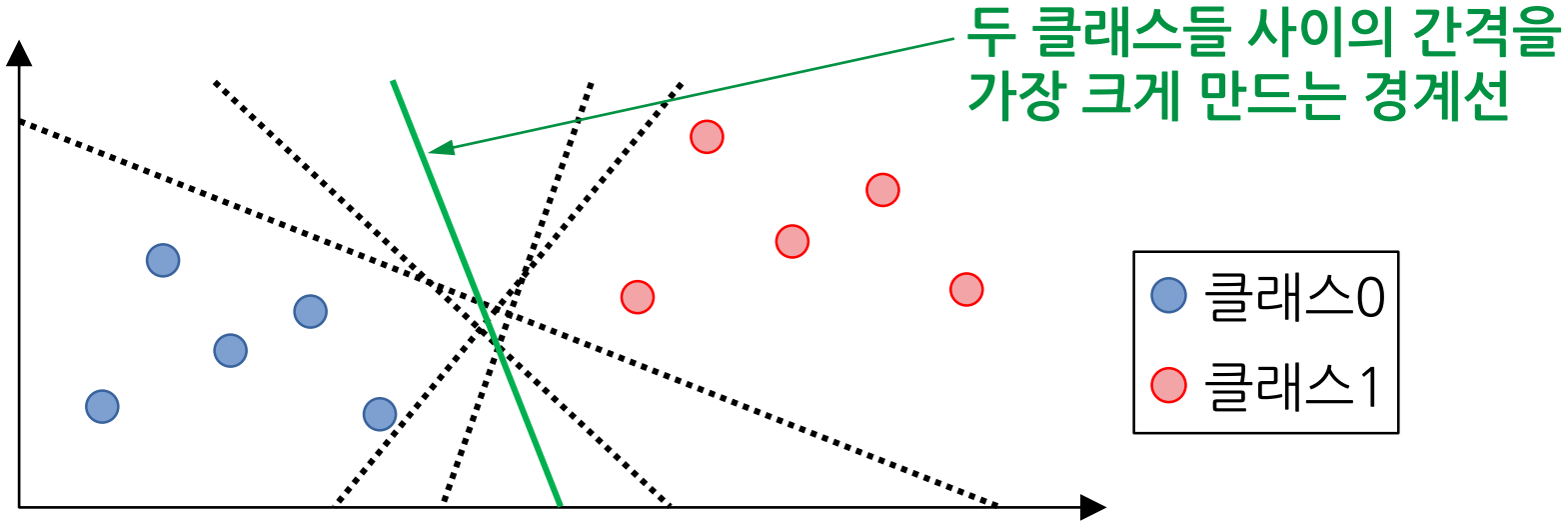
서포트 벡터 머신

서포트 벡터 머신

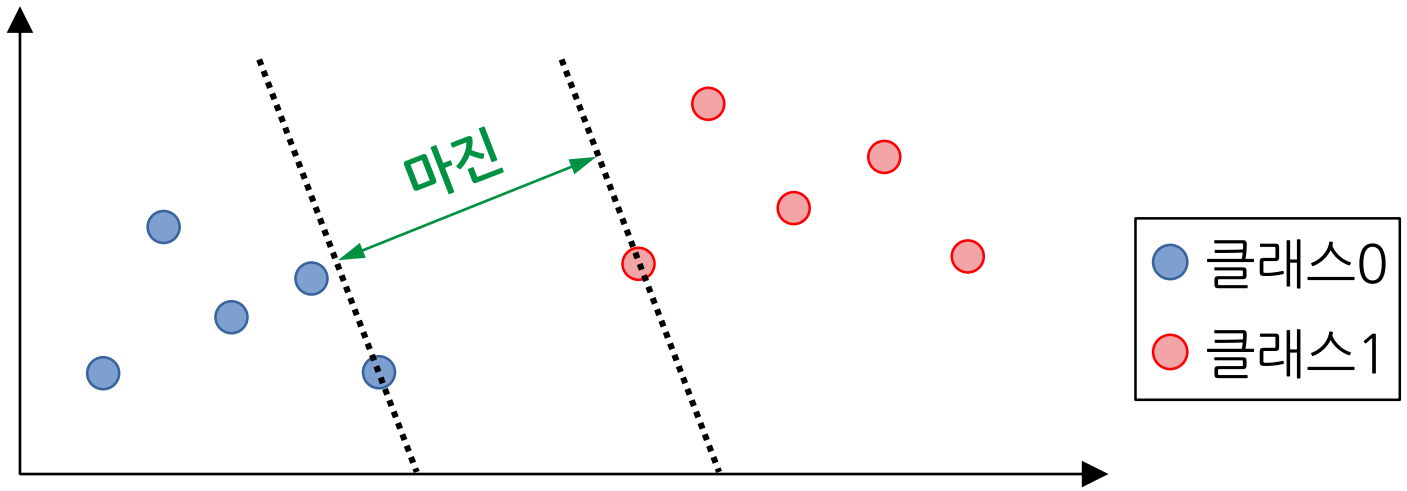
- 서포트 벡터 머신 (SVM; Support Vector Machine)
  - 이진 클래스에 대한 분류 기법 중 하나
  - 두 개의 클래스를 구분하는 경계(boundary)들 중 클래스들 사이의 최적의 경계를 찾는다.



  - 두 개의 클래스를 구분하는 경계선은 무수히 많다. 이들 중 가장 적합한 경계선을 긋는 것이 SVM의 목표이다.

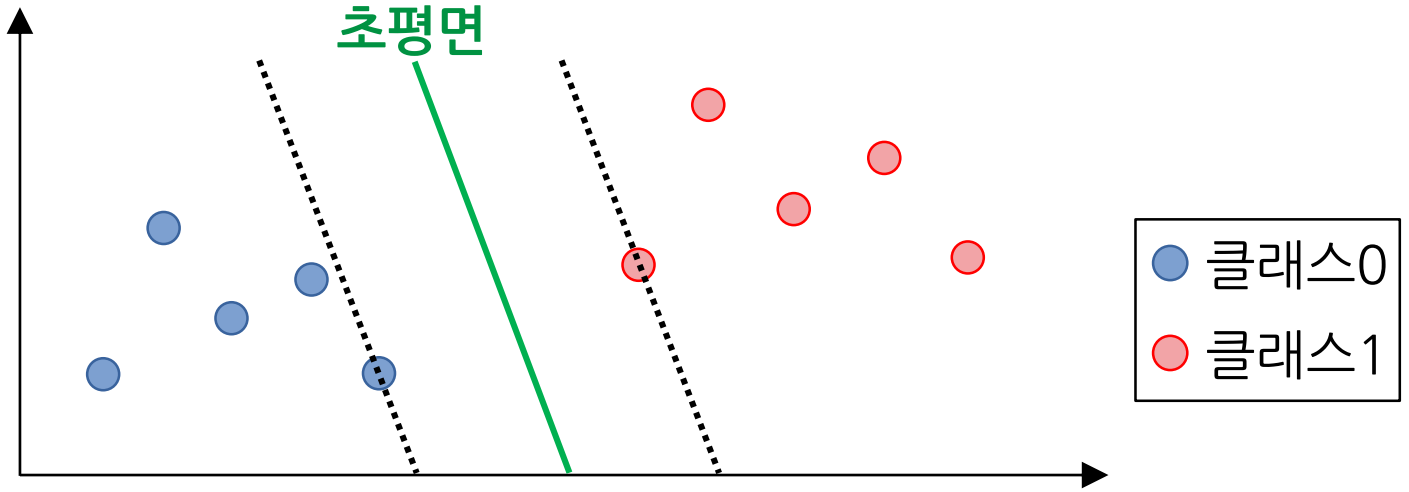
서포트 벡터 머신

- 서포트 벡터 머신
  - 마진 (margin)
    - 클래스들 사이의 간격, 즉 여백
    - 각 클래스들의 말단에 위치한 데이터들 사이의 거리에 해당한다.



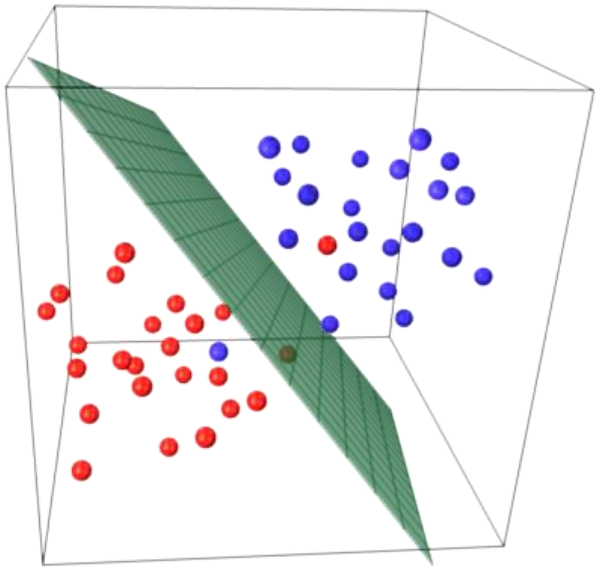
서포트 벡터 머신

- 서포트 벡터 머신
  - 초평면 (hyperplane)
    - 한 영역을 두 개의 부분으로 나누는 결정 경계 (decision boundary)
    - 데이터를 분리하는 초평면은 무한히 많이 존재한다.



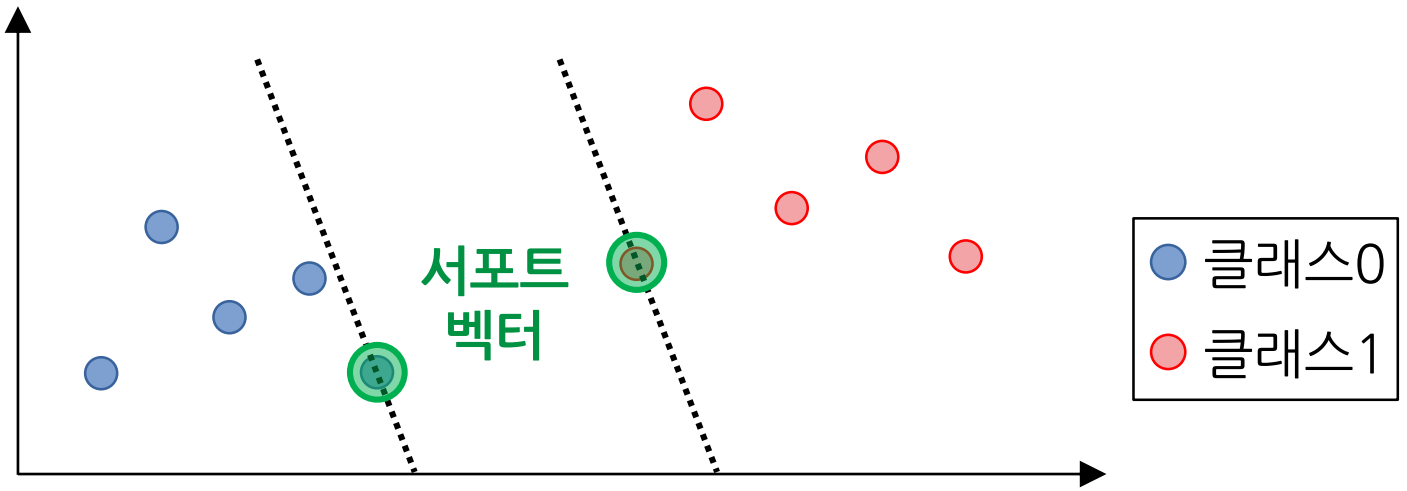
서포트 벡터 머신

- 서포트 벡터 머신
  - 초평면 (hyperplane)
    - 예를 들어, 2차원 공간 영역을 나누는 경계는 직선이고 3차원 공간을 나누는 경계는 평면이다.
    - 일반적으로 표현해서, 초평면은 N차원 공간을 (N-1)차원으로 나누는 경계를 뜻한다.



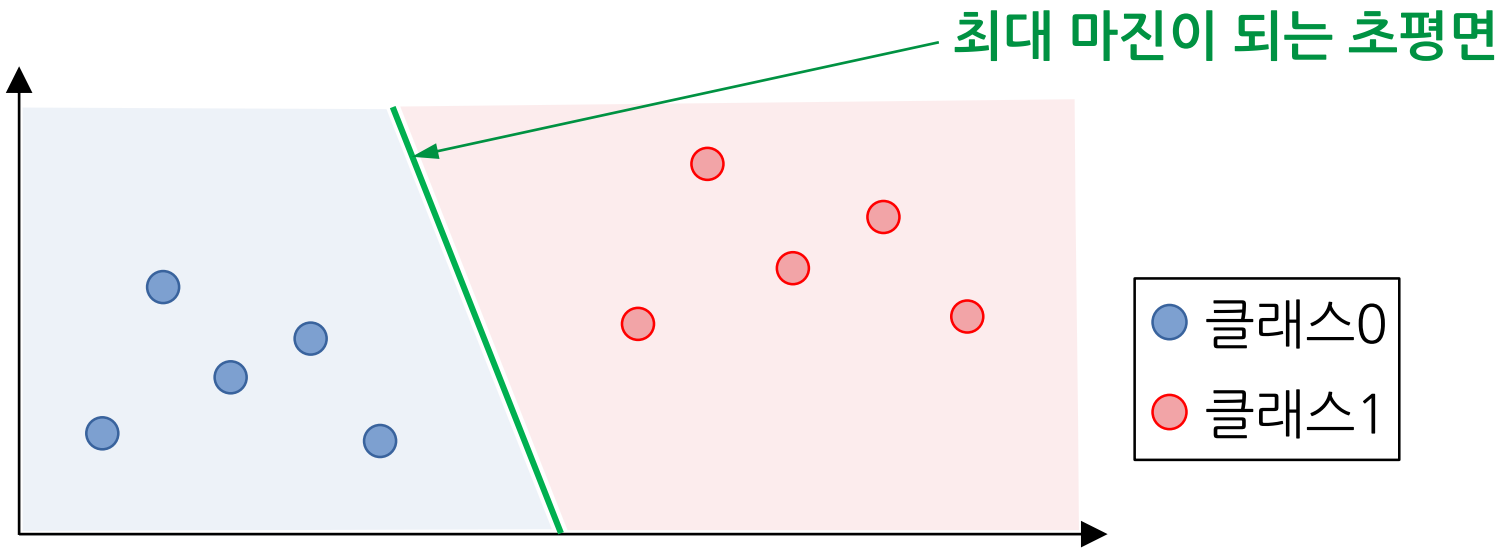
서포트 벡터 머신

- 서포트 벡터 머신
  - 서포트 벡터 (support vector)
    - 마진에서 서로 가장 가까이 위치해 있는 데이터
    - 이 데이터들의 위치에 따라 초평면의 위치도 달라질 것이다. 즉, 이 값들은 초평면 함수를 지지(support)한다.



서포트 벡터 머신

- 서포트 벡터 머신의 개념 요약
  - 데이터를 분리하는 최적의 초평면(hyperplane), 즉 최적의 결정 경계(decision boundary)를 찾는 기법
  - 결국, 최대 마진(maximum margin)이 되도록 클래스를 구분하는 것이 기본적인 목적이다.



선형 SVM

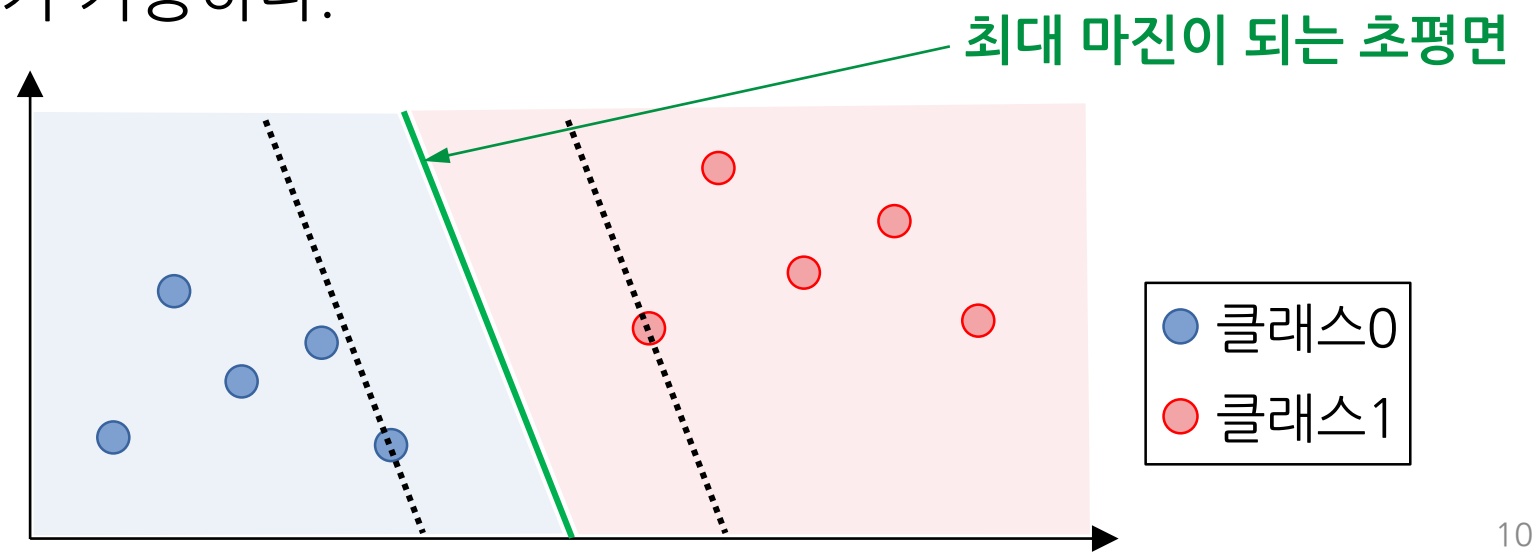
하드 마진 분류

선형 SVM

- 선형 SVM (Linear SVM)
  - 데이터를 선형으로 구분하는 최적의 초평면을 찾는 기법
  - 선형 SVM은 두 가지 경우로 구분할 수 있다.
    - 하드 마진 (hard margin) 분류 : 두 개의 클래스에 대해 최대 마진이 되는 초평면을 찾는다.
    - 소프트 마진 (soft margin) 분류 : 하드 마진 분류에서 초평면이 존재하지 않을 때, 오분류를 허용하여 찾는다.

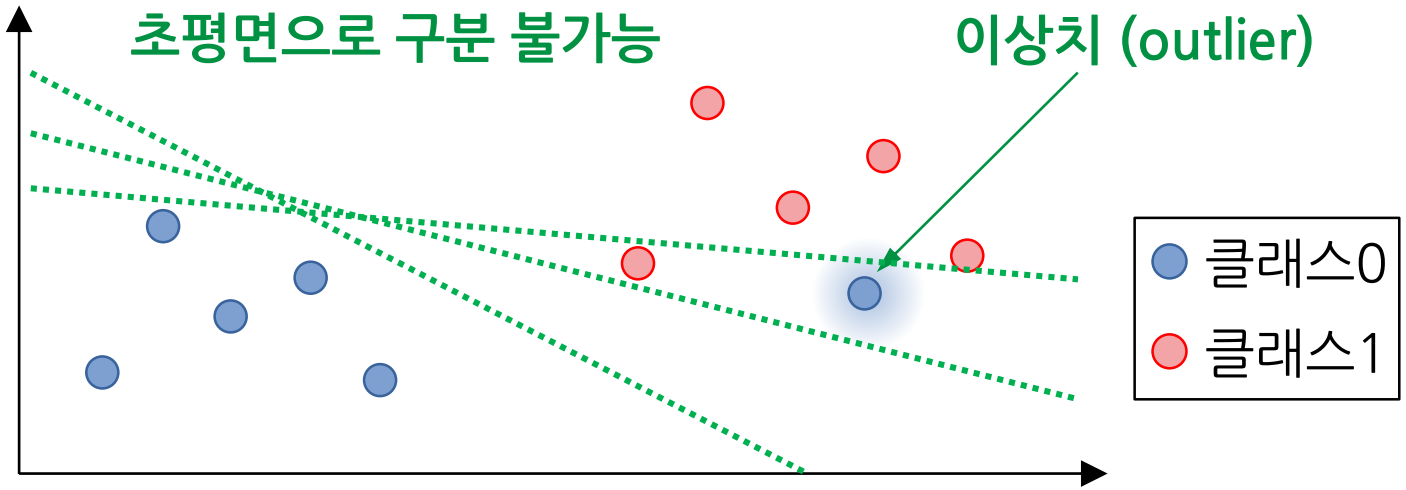
선형 SVM

- 하드 마진 (Hard Margin) 분류
  - 두 개의 클래스에 대해 최대 마진이 되는 초평면을 찾는다.
    - 이와 같이 하드 마진 분류를 수행하는 선형 SVM을 최대 마진 분류기(maximum margin classifier)라고 한다.
  - 모든 훈련 데이터들은 마진의 바깥쪽에 위치하게 된다.
  - 데이터들이 정확하게 선형적으로 구분되는 경우에만 분류가 가능하다.



선형 SVM

- 하드 마진 분류의 한계
  - 모든 경우에 반드시 초평면이 존재하는 것은 아니다.
    - 데이터가 정확하게 선형적으로 구분되지 않는 경우에는 결정 경계를 찾는 것이 불가능하다.
  - 분류 모형이 일반화되기 어렵다.
    - 이상치가 존재할 경우, 초평면이 없거나 잘 일반화되지 않는다.



선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (1)
  ① svm 모듈에 있는 LinearSVC를 이용하여 선형 SVM 객체를 생성한다.
    - 이 때 필요하다면 매개변수들을 설정할 수 있으나, 일단 처음에는 기본 상태로 적용한다.

```
1  import sklearn.svm as svm
2
3  svm_clf = svm.LinearSVC()
```

※ 학습 데이터는 연습용으로 다음과 같이 준비한다.

```
1  X_train = [[1, 2], [5, 6], [2, 3], [6, 7], [3, 1], [7, 5]]
2  y_train = [0, 1, 0, 1, 0, 1]
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (1)

② 선형 SVM 객체에 대하여 **fit** 메소드를 이용하여 훈련한다.

```
1  clf = svm_clf.fit(X_train, y_train)
```

③ 실행 객체 또는 분류 모형에 대하여 **predict** 메소드를 이용하여 예측을 수행한다.

```
1  X_test = [[1.5, 3.1], [4, 4.5], [6.7, 4.7]]
2  y_test = [0, 0, 1]
3
4  y_pred = clf.predict(X_test)
```

```
1  print(y_pred)
```

```
[0 1 1]
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (1)
  ④ 분석 결과를 평가한다. (정확도)
  - **metrics** 모듈에 있는 **accuracy_score** 함수를 이용하여 정확도를 구한다.

```
1  import sklearn.metrics as mt
2
3  score = mt.accuracy_score(y_test, y_pred)
4  print("정확도: {:.3f}".format(score))
```

```
정확도: 0.667
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (1)
  ④ 분석 결과를 평가한다. (정확도)
  - 또는, 실행 객체 또는 분류 모형에 대하여 **score** 메소드를 호출하여 정확도를 구할 수도 있다.
  - 이 때 첫 번째 매개변수는 검증 데이터의 특성이고, 두 번째 매개변수는 검증 데이터의 클래스이다.

```
1  score = clf.score(X_test, y_test)
2  print("정확도: {:.3f}".format(score))
```

```
정확도: 0.667
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (2)
  ① **svm** 모듈에 있는 **SVC**를 이용하여 선형 SVM 분류를 할 수 있다.
    - 매개변수 kernel을 “linear”로 지정한다. 선형으로 분류 결정 경계를 구하겠다는 의미이다.

```
1  import sklearn.svm as svm
2
3  svm_clf = svm.SVC(kernel="linear")
```

※ kernel에 대한 자세한 내용은 추후 설명한다.

※ 학습 데이터는 이전과 동일한 것을 이용한다.

```
1  X_train = [[1, 2], [5, 6], [2, 3], [6, 7], [3, 1], [7, 5]]
2  y_train = [0, 1, 0, 1, 0, 1]
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (2)

② LinearSVC와 동일한 방식으로 **fit** 메소드를 이용하여 훈련한다.

```
1  clf = svm_clf.fit(X_train, y_train)
```

③ 마찬가지로 **predict** 메소드를 이용하여 예측을 수행한다.

```
1  X_test = [[1.5, 3.1], [4, 4.5], [6.7, 4.7]]
2  y_test = [0, 0, 1]
3
4  y_pred = clf.predict(X_test)
```

```
1  print(y_pred)
```

```
[0 1 1]
```

선형 SVM

- 사이킷런으로 선형 SVM (하드 마진) 분류 수행 (2)
  ④ 마찬가지로 **accuracy_score** 함수 또는 **score** 메소드를 이용하여 정확도를 구한다.

```
1  import sklearn.metrics as mt
2
3  score = mt.accuracy_score(y_test, y_pred)
4  print("정확도: {:.3f}".format(score))
```

```
정확도: 0.667
```

```
1  score = clf.score(X_test, y_test)
2  print("정확도: {:.3f}".format(score))
```

```
정확도: 0.667
```

선형 SVM

- LinearSVC와 SVC의 차이

구분	LinearSVC	SVC
용도	선형 SVM 분류를 수행한다.	선형 SVM 및 비선형 SVM 분류를 수행할 수 있다.
수행 기법 (알고리즘)	liblinear	libsvm
특징	훈련 데이터가 크거나 특성 수가 많아도 수행이 빠르다.	복잡하지만 규모가 크지 않은 데이터의 학습에 적합하다.
커널	없다. (선형 분류 전용)	여러 가지 커널 트릭을 지원한다.
기본 손실 함수	제곱 힌지 손실 (squared hinge loss)	힌지 손실 (hinge loss)

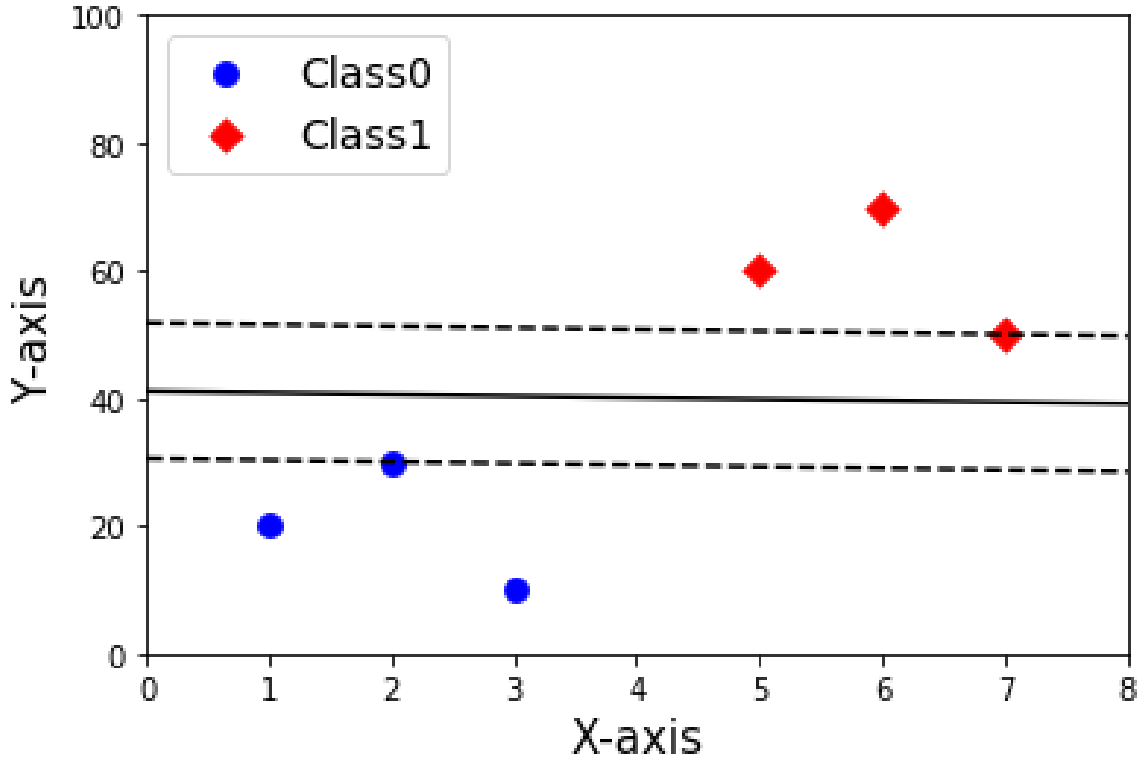
선형 SVM

- SVM의 스케일링
  - SVM은 데이터 특성의 스케일에 민감하다.
    - 아래와 같이 X축보다 Y축 값들의 스케일이 훨씬 크면 결정 경계가 거의 수평에 가깝게 된다.

```
1  import sklearn.svm as svm
2
3  # 원래 X_train = [[1, 2], [5, 6], [2, 3], [6, 7], [3, 1], [7, 5]]
4  X_train = [[item[0], item[1] * 10] for item in X_train]
5  y_train = [0, 1, 0, 1, 0, 1]
6
7  svm_clf = svm.SVC(kernel="linear")
8
9  clf = svm_clf.fit(X_train, y_train)
```

선형 SVM

- SVM의 스케일링
  - SVM은 데이터 특성의 스케일에 민감하다.
    - 특성 값의 스케일 차이로 인해, 결정 경계가 거의 수평선으로 표현되는 것을 확인할 수 있다.



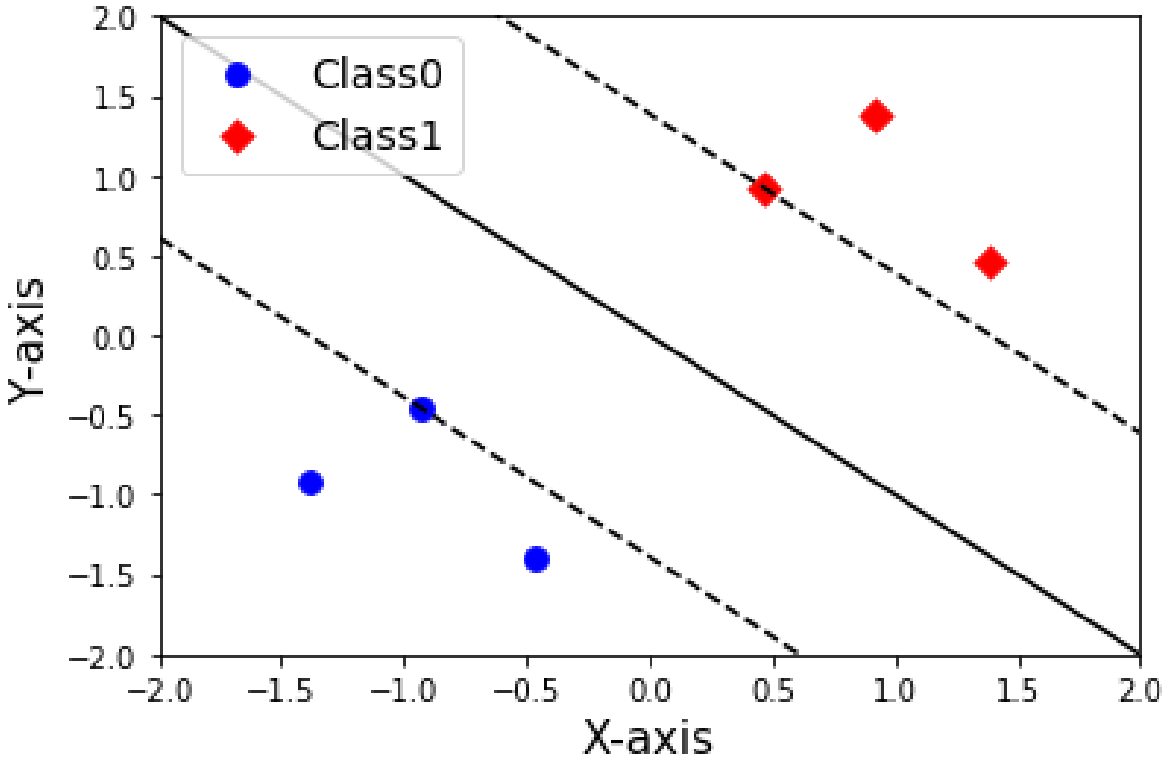
선형 SVM

- SVM의 스케일링
  - 스케일링을 통해, 보다 적합한 결정 경계를 도출할 수 있다.
    - preprocessing 모듈의 StandardScaler에서 fit_transform 메소드를 이용하여 표준 정규 분포로 전처리한다.

```
1  import sklearn.preprocessing as pp
2  import sklearn.svm as svm
3
4  # 원래 X_train = [[1, 2], [5, 6], [2, 3], [6, 7], [3, 1], [7, 5]]
5  X_train = [[item[0], item[1] * 10] for item in X_train]
6  y_train = [0, 1, 0, 1, 0, 1]
7
8  svm_clf = svm.SVC(kernel="linear")
9
10 scl = pp.StandardScaler()
11 X_train = scl.fit_transform(X_train)
12
13 clf = svm_clf.fit(X_train, y_train)
```

선형 SVM

- SVM의 스케일링
  - 스케일링을 통해, 보다 적합한 결정 경계를 도출할 수 있다.
    - 스케일을 조정한 결과, 원래의 경우보다 더 나은 결정 경계가 표현되는 것을 확인할 수 있다.



참고 : 2차원 데이터 분류에서의 결정 경계

선형 SVM

- 선형 SVM 분류의 초평면 (결정 경계)
  - 2차원 데이터에 대한 결정 경계 식은 다음과 같이 쓸 수 있다.

$$w_0 + w_1 \cdot X + w_2 \cdot y = 0$$

(이 때, 각 w_i는 계수와 절편이다.)

  - 이 식을 y에 대해서 정리하면 다음과 같다.

$$y = -\frac{w_1}{w_2} \cdot X - \frac{w_0}{w_2}$$

선형 SVM

- 사이킷런의 실행 결과
  - 분류 모형의 결과로 계수와 절편 값이 들어 있으며, 계수는 **coef_** 속성, 절편은 **intercept_** 속성에 값이 할당되어 있다.
  - 이 값들을 이용하여 결정 경계 식을 표현한다.

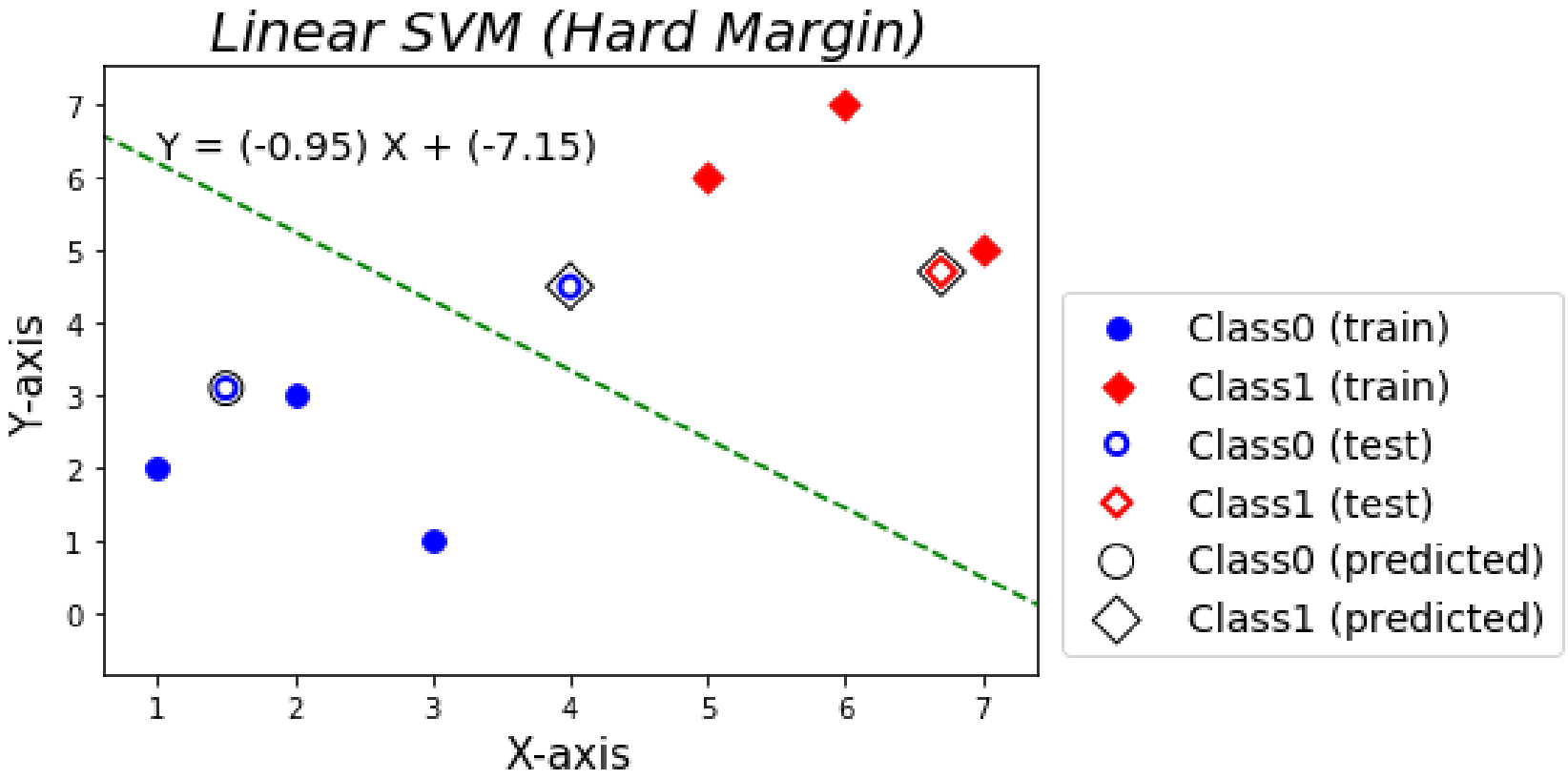
$$y = -\frac{w_1}{w_2} \cdot X - \frac{w_0}{w_2}$$

```
1  import numpy as np
2
3  xx = np.linspace(1, 7)
4
5  yy = -(clf.coef_[0][0] / clf.coef_[0][1]) * xx - ₩
6  clf.intercept_ / clf.coef_[0][1]
```

※ 클래스가 2종류일 때 coef_ 속성은 1행, 2열 형태인 2차원 ndarray이다.

선형 SVM

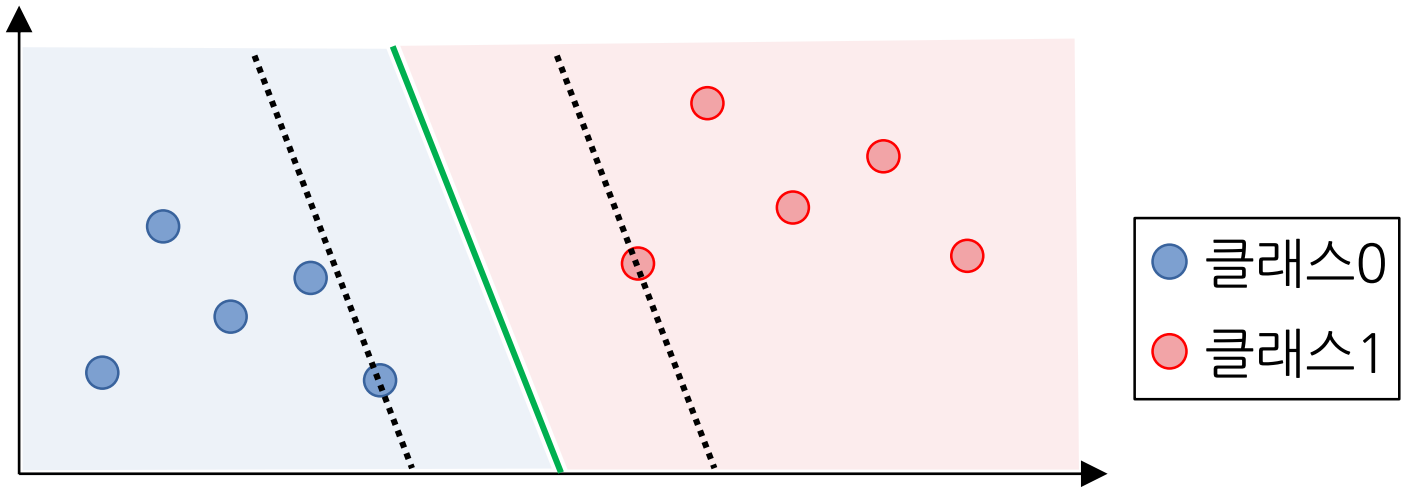
- 데이터 산점도 및 결정 경계 플롯



참고 : 힌지 손실 함수와 초평면 상의 값

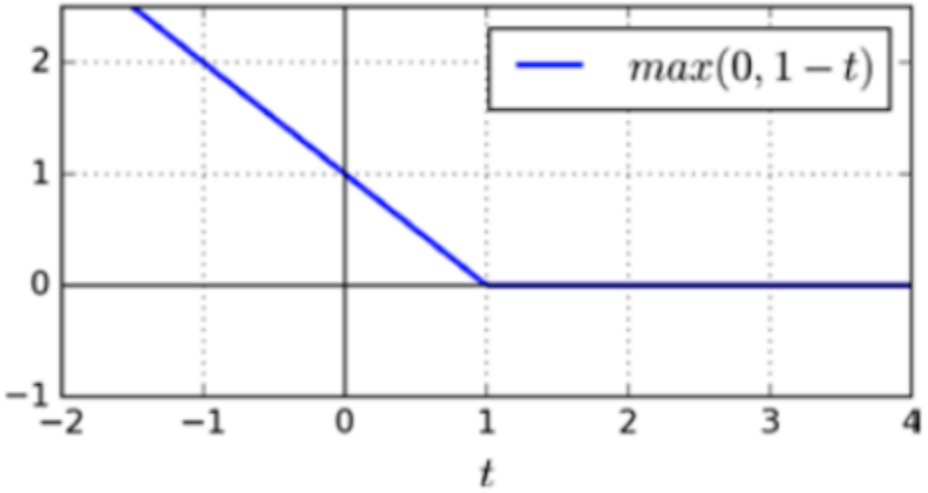
선형 SVM

- 초평면과 결정 경계에 따른 값의 분포
  - 양성 초평면 : 결과 값이 +1 이상이 되는 초평면 영역이다.
  - 음성 초평면 : 결과 값이 -1 이하가 되는 초평면 영역이다.
  - 결정 경계 : 이 경계 상에 있는 데이터의 결과 값은 0이다.
    - 데이터의 결과 값이 0 이상이면 클래스 1로 분류된다.
    - 데이터의 결과 값이 0 미만이면 클래스 0으로 분류된다.



선형 SVM

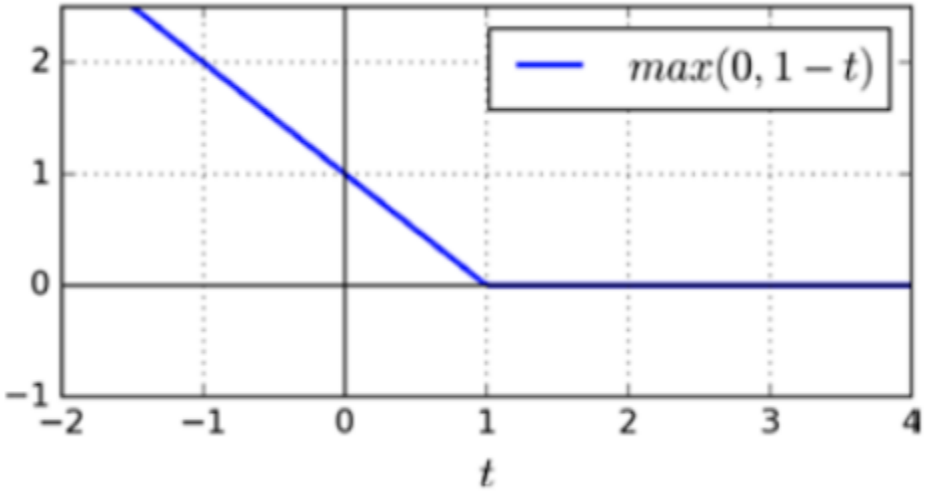
- 힌지 손실 함수 (Hinge Loss Function)
  - 힌지 손실 함수



  - t가 1 이상이면 함수 결과 값은 0이다.
  - t가 1 미만이면 함수 결과 값은 1-t이다.
  - 실제 클래스 집합이 +1 및 -1인 이진 분류 문제에서 t는 (실제값)×(예측값)으로 정의한다.

선형 SVM

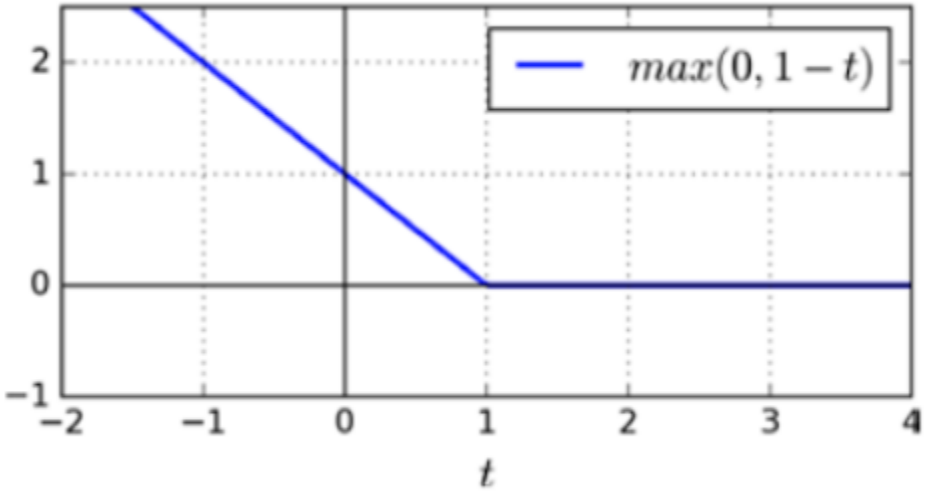
- 힌지 손실 함수 (Hinge Loss Function)
  - 힌지 손실 함수와 손실 값



  - 양성 초평면 상에 있는 데이터들의 실제값들은 +1이고, 예측값들은 +1 이상이다.
  - 따라서, t = (실제값)×(예측값)은 1 이상이다.
  - 즉, 힌지 손실 값은 max(0, 0 이하의 값) = 0이다.

선형 SVM

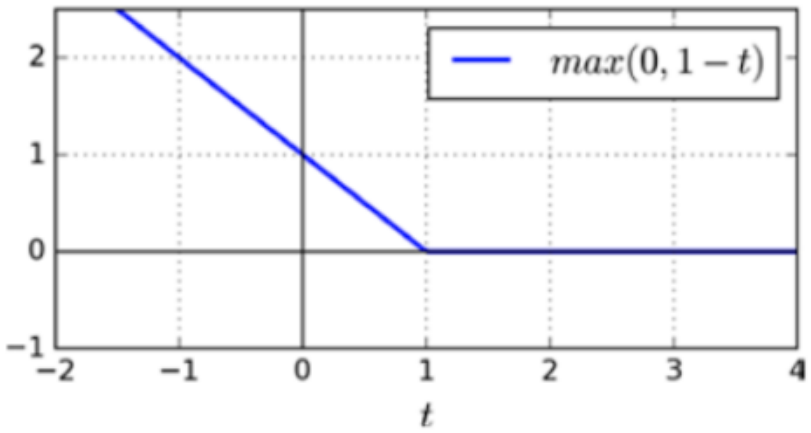
- 힌지 손실 함수 (Hinge Loss Function)
  - 힌지 손실 함수와 손실 값

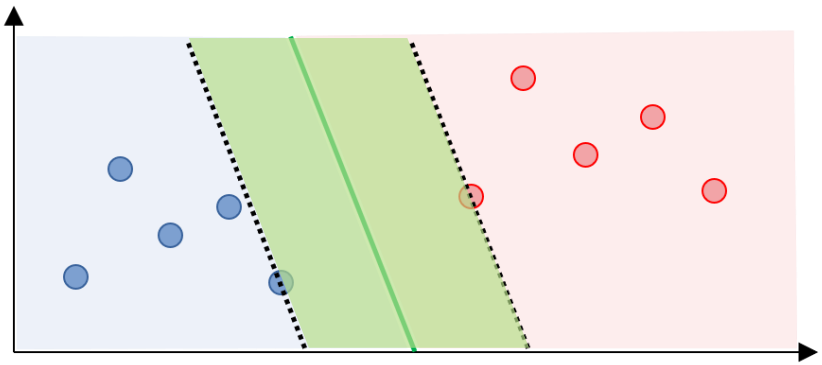


  - 음성 초평면 상에 있는 데이터들의 실제값들은 -1이고, 예측값들은 -1 이하이다.
  - 따라서, t = (실제값)×(예측값)은 1 이상이다.
  - 즉, 힌지 손실 값은 max(0, 0 이하의 값) = 0이다.

선형 SVM

- 힌지 손실 함수 (Hinge Loss Function)
  - 힌지 손실 함수와 손실 값

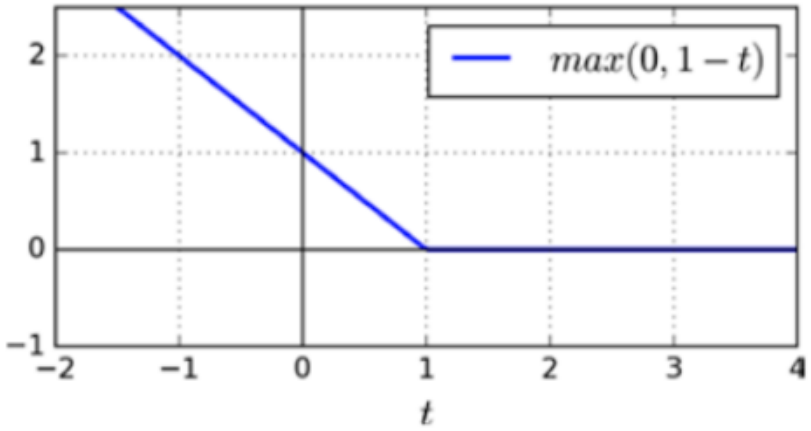


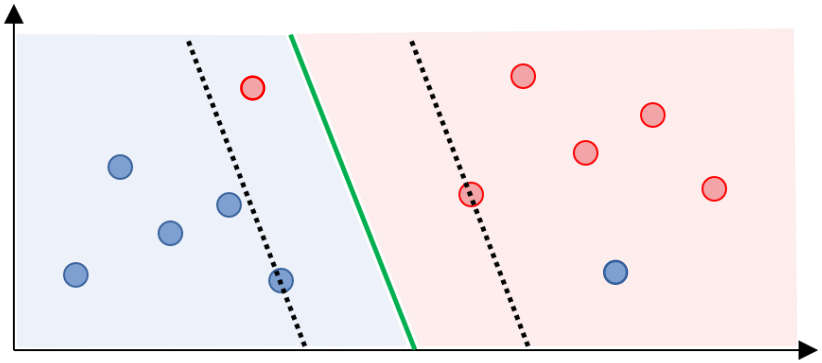


  - 결정 경계와 초평면 사이에 있는 (즉, 마진 내의 영역에 존재하는) 데이터에 대한 t의 값은 1 미만이다.
  - 이 때의 힌지 손실 값은 1 미만의 값(= 1-t)이다.
  - 분류 자체가 올바르다고 할지라도, 예측값과 실제값의 차이가 발생하면 그만큼의 손실을 고려한다.

선형 SVM

- 힌지 손실 함수 (Hinge Loss Function)
  - 힌지 손실 함수와 손실 값





  - 예측값과 실제값의 부호가 다르면 (즉, 분류 자체가 잘못 되었으면) 그 데이터에 대한 t의 값은 음수이다.
  - 이 때의 힌지 손실 값은 1-(음수)이므로 1을 초과하는 값이다.
  - 예측값과 실제값의 차이가 클수록 손실을 크게 간주한다.