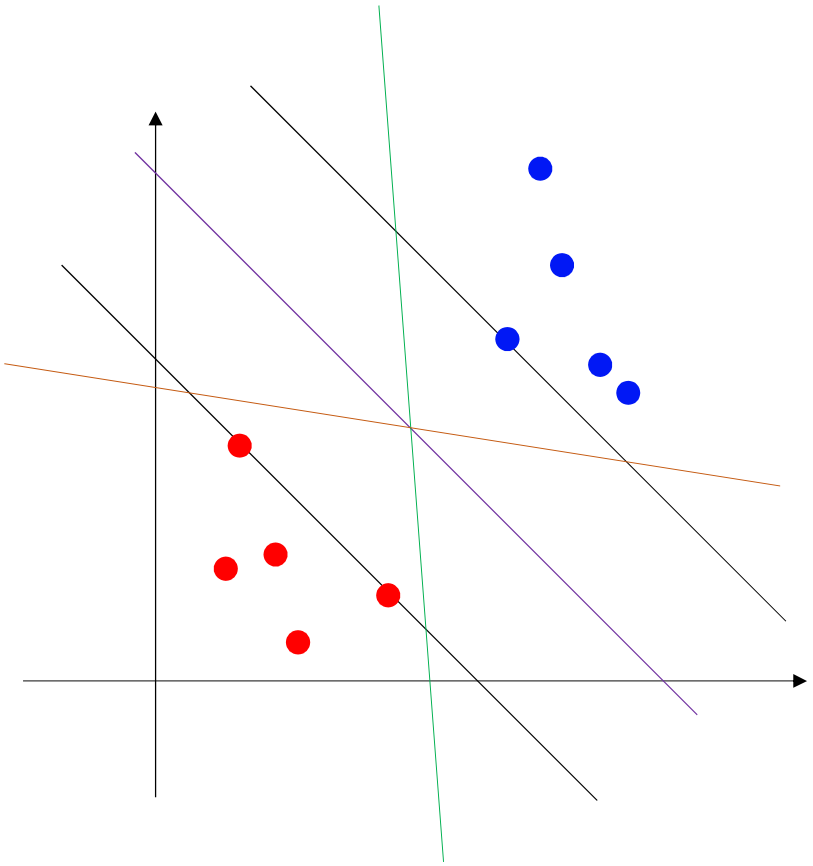
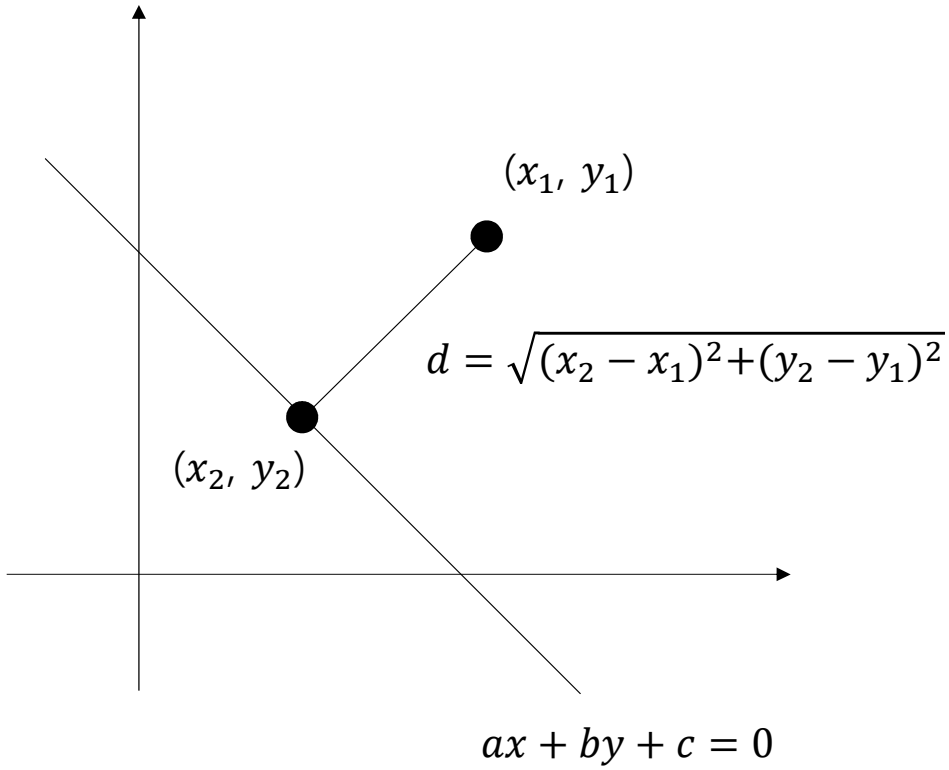
# 서포트벡터머신

# 서포트벡터머신(SVM)

- 주어진 데이터가 어느 클래스에 속할지 판단하는 비확률적 이진 선형 분류 모델
- 분류와 회귀 문제 중 주로 분류 문제에 사용
- 서로 다른 클래스를 분류하는 기준인 결정 경계선에 가장 가까이 있는 데이터(서포트 벡터, Support Vector)와 직교하는 직선과의 거리(마진, Margin)가 최대가 되도록 학습하는 모델
- 오류 최소화가 아닌 마진 최대화를 목적으로 설계



# 직선과 점 간의 거리




수직인 두 직선의 기울기 곱 = -1

$$\frac{y_2 - y_1}{x_2 - x_1} \cdot \left(-\frac{a}{b}\right) = -1$$

$$\frac{y_2 - y_1}{b} = \frac{x_2 - x_1}{a} = z$$

$$x_2 = x_1 + az$$

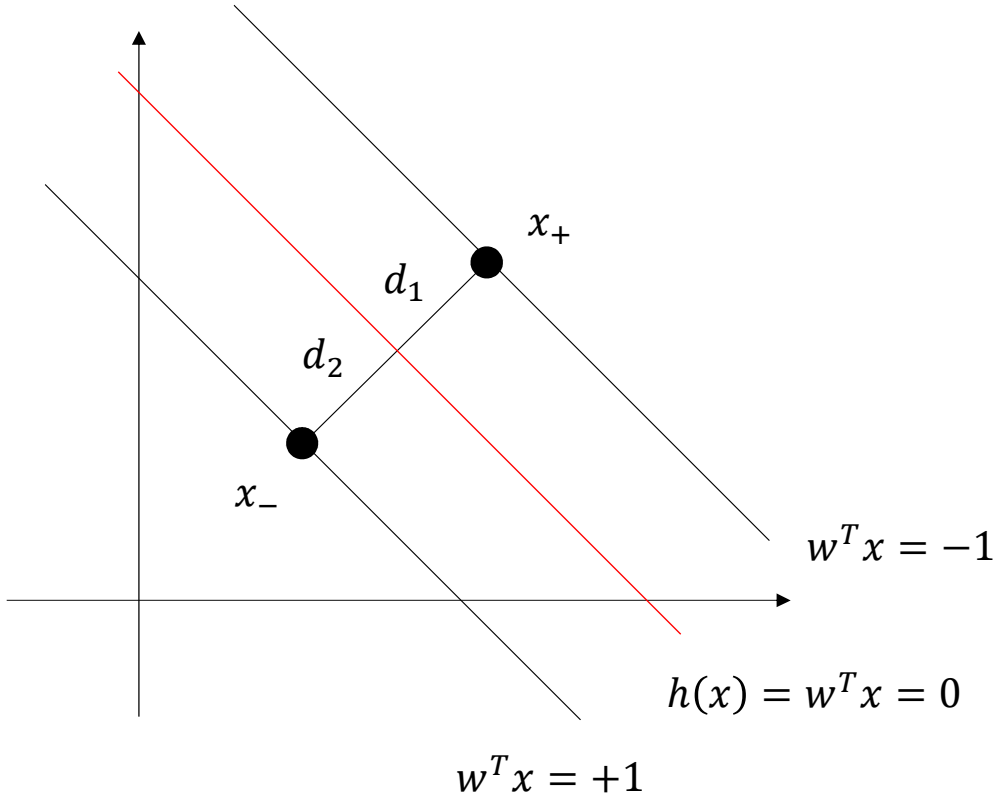
$$y_2 = y_1 + bz$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 = (a^2 + b^2) \cdot z^2$$

$$ax_2 + by_2 + c = 0 \rightarrow z = -\frac{ax_1 + by_1 + c}{a^2 + b^2}$$

$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

# 마진 최대화




$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

$$d_1 = \frac{|w^T x_+|}{\sqrt{w^2}} = \frac{|-1|}{\|w\|} \qquad d_2 = \frac{|w^T x_-|}{\sqrt{w^2}} = \frac{|1|}{\|w\|}$$

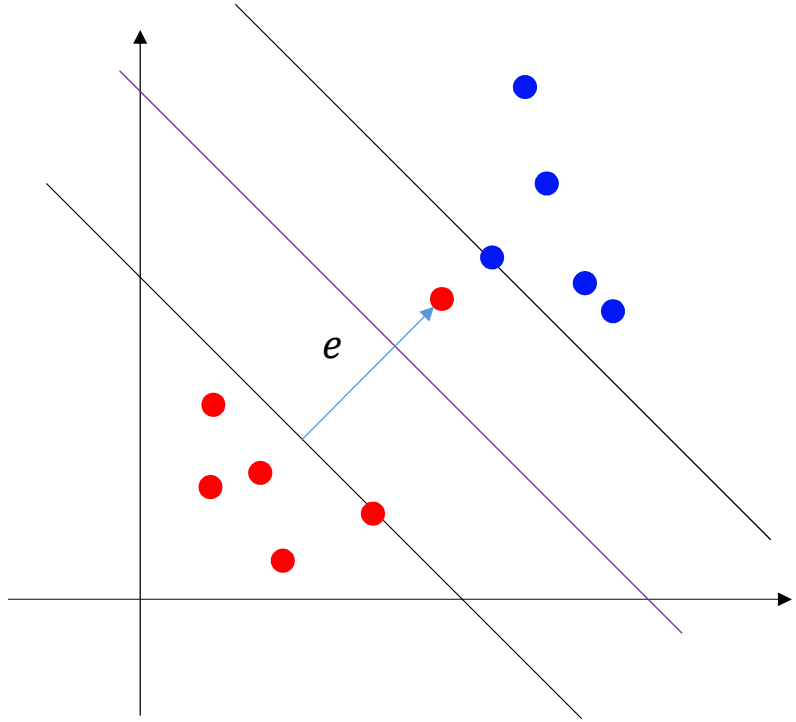
$$d = d_1 + d_2 = \frac{2}{\|w\|}$$

$$max \frac{1}{\|w\|} \leftrightarrow min \frac{1}{2} \|w\|^2$$

# Soft Margin SVM

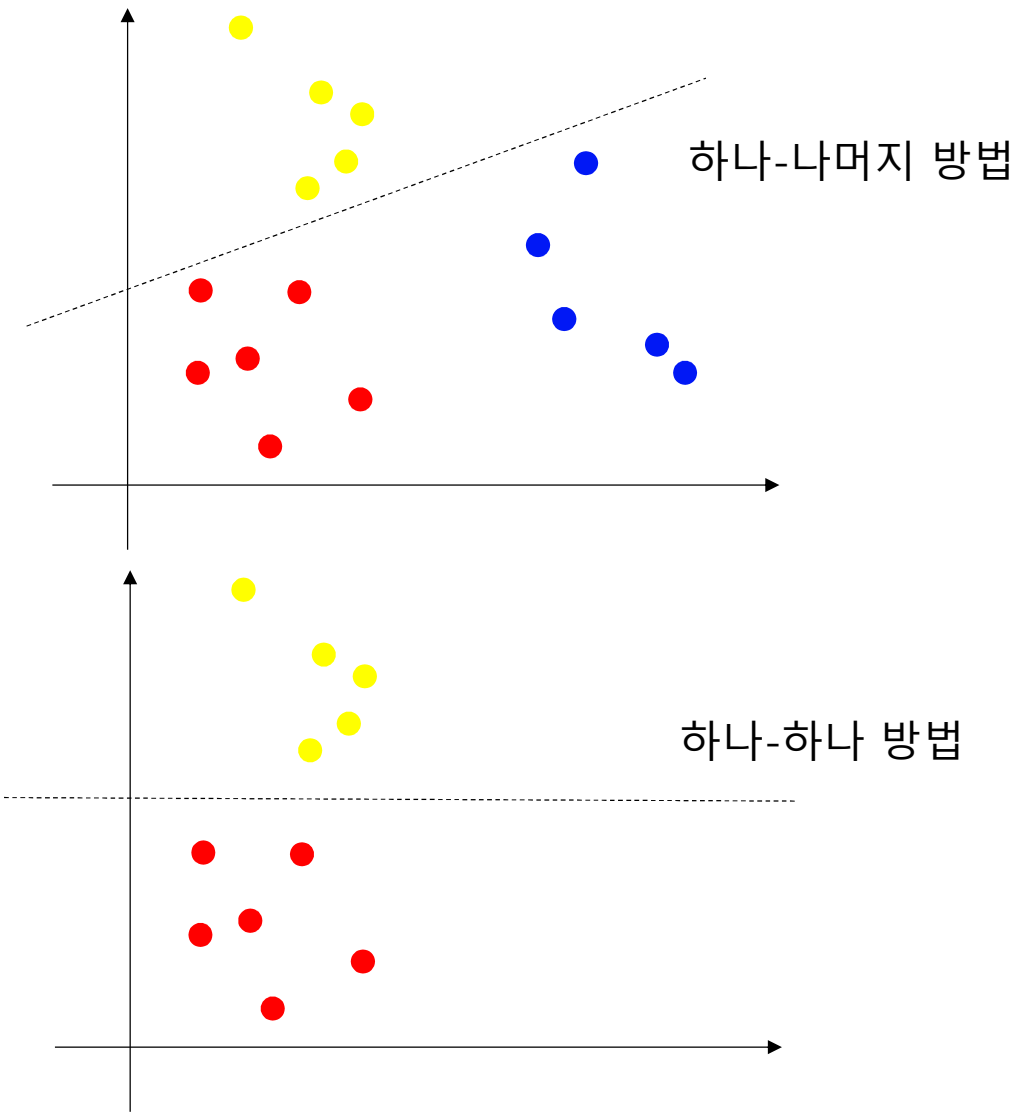
- 학습 데이터의 에러가 0이 되도록 완벽히 분류하는 것은 불가능
- 학습 데이터에 대해 잘못 분류되는 것을 허용하면서 마진을 최대화하는 방식
- 패털티항을 추가하여 C계수를 조정하면서 오분류에 대한 불이익 정도를 조정
- C가 커질수록 마진이 작아지고(Hard Margin), C가 작아질수록 마진이 커짐(Soft Margin)
- C가 무한대면 원래 SVM과 동일

$$\min(\frac{1}{2}\|w\|^2 + C\sum e)$$



# 다중분류

- 하나-나머지(One-vs-the-Rest) 방법
  - 결정점수가 가장 큰 값의 클래스 결정
  - N개의 모델을 생성
  - sklearn.svm.LinearSVC 사용
- 하나-하나(One-vs-One) 방법
  - 주어진 특징들에 대해 가장 많이 할당된 클래스로 결정
  - N(N-1)/2개의 모델 생성
  - sklearn.svm.SVC 사용



# sklearn.svm.LinearSVC

*class* sklearn.svm.LinearSVC(*penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*) [source]

Linear Support Vector Classification.

This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

**penalty : *{'l1', 'l2'}, default='l2'***

Specifies the norm used in the penalization. The 'l2' penalty is the standard used in SVC. The 'l1' leads to `coef_` vectors that are sparse.

**C : *float, default=1.0***

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.

**max_iter : *int, default=1000***

The maximum number of iterations to be run.

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

# Ont-vs-the-Rest 방법

```
import seaborn as sns
iris = sns.load_dataset("iris")

data = iris.drop("species", axis = 1)
t = iris[["species"]].copy()
t[t["species"] == "setosa"] = 0
t[t["species"] == "versicolor"] = 1
t[t["species"] == "virginica"] = 2
t = t["species"].astype("int")
```

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    data, t, test_size = 0.3, random_state = 42, stratify = t)

from sklearn.svm import LinearSVC
svm = LinearSVC(random_state = 42, max_iter = 3000)
svm.fit(train_data, train_target)
print("Train-Eval:", svm.score(train_data, train_target))
print("Test-Eval :", svm.score(test_data, test_target))
```

```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(test_target, dt.predict(test_data))
print(conf)
```

```
from sklearn.svm import LinearSVC
svm = LinearSVC(random_state = 42)
svm.fit(train_data, train_target)
print("Train-Eval:", svm.score(train_data, train_target))
print("Test-Eval :", svm.score(test_data, test_target))
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
Train-Eval: 0.9619047619047619
Test-Eval : 0.9111111111111111
[[15  0  0]
 [ 0 12  3]
 [ 0  1 14]]
```

# sklearn.svm.SVC

*class* sklearn.svm.SVC(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=- 1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*) [source]

C-Support Vector Classification.

The multiclass support is handled according to a one-vs-one scheme.

**C : *float, default=1.0***

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

**kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'***

Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape

**max_iter : *int, default=-1***

Hard limit on iterations within solver, or -1 for no limit.

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

# Ont-vs-One 방법

구현(ramdom_state=42, max_inter=5000)

```
print("Train-Eval:", svm.score(train_data, train_target))
print("Test-Eval :", svm.score(test_data, test_target))
```

```
print(svm.coef_)
print(svm.intercept_)
```

```
[[-0.23893126  0.48214934 -0.86551817 -0.60257466]
 [-0.15449394  0.15924522 -0.4652571  -0.22983524]
 [ 0.57753985  0.18079714 -1.89503132 -1.86158481]]
[2.27230402 2.15877744 8.2688225 ]
```

```
[[ 0.18491555  0.43524487 -0.79287366 -0.47343027]
 [ 0.07868075 -0.96874741  0.44431518 -1.06863357]
 [-0.80609785 -0.73155445  1.23970978  1.56128372]]
[ 0.11900691  1.73097348 -1.5384255 ]
```

```
Train-Eval: 0.9809523809523809
Test-Eval : 1.0
[[15  0  0]
 [ 0 15  0]
 [ 0  0 15]]
```

하나-하나 방법

```
Train-Eval: 0.9619047619047619
Test-Eval : 0.9111111111111111
[[15  0  0]
 [ 0 12  3]
 [ 0  1 14]]
```

하나-나머지 방법

# 참고자료

- 지능기전공학부 최유경 교수님 자료, https://github.com/sejongresearch/2021.MachineLearning
- 코랩(Colab), https://colab.research.google.com/
- 파이썬(Python), https://www.python.org/doc/
- 사이킷런(sckit-learn), https://scikit-learn.org/stable/index.html
- 판다스(pandas), https://pandas.pydata.org/
- 맷플롯립(matplotlib), https://matplotlib.org/
- 씨본(seaborn), https://seaborn.pydata.org/
- 캐글(Kaggle), https://www.kaggle.com/
- 넘파이(numpy), https://numpy.org/doc/stable/
- 스택오퍼플러우(stackoverflow), https://stackoverflow.com/