5장
서포트벡터머신(SVM)

- 선형 SVM분류
- 비선형 SVM분류
- SVM회귀
- SVM이론

Support Vector Machine(SVM)

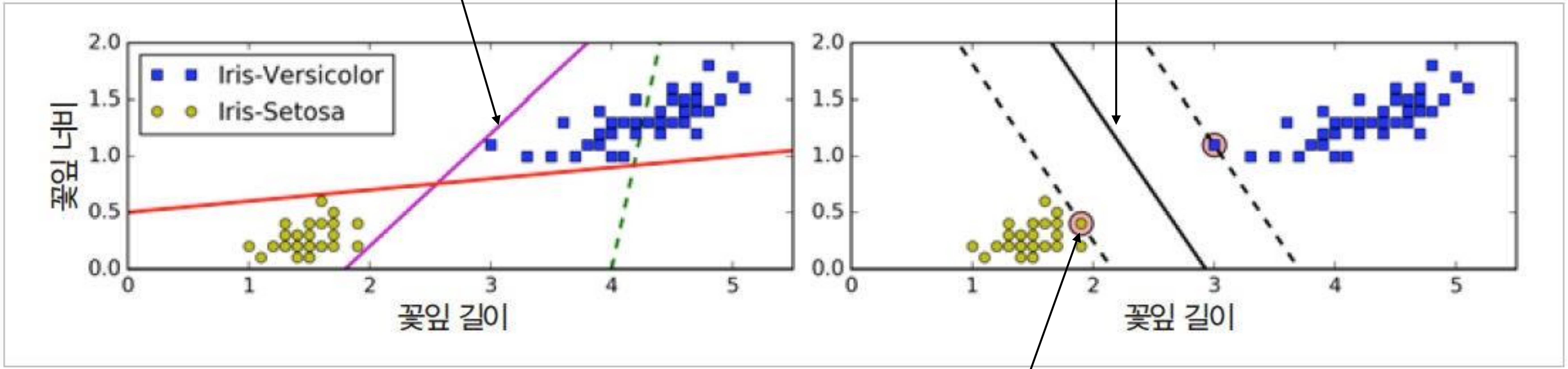
- 선형, 비선형 문제에 모두 적용 가능
- 분류, 회귀 문제에 모두 적용 가능
- 수학적으로 정의가 잘 되어 있음
- 복잡도를 조정할 수 있음
- 선형 : liblinear(coordinate descent)
  비선형 : libsvm(quadratic programming)

SVM의 개념

- 마진을 최대화하도록 결정경계 설정 (=서포트벡터 선택)
  ➔ Maximal Margin Classifier

결정 경계가 샘플에 너무 가깝습니다

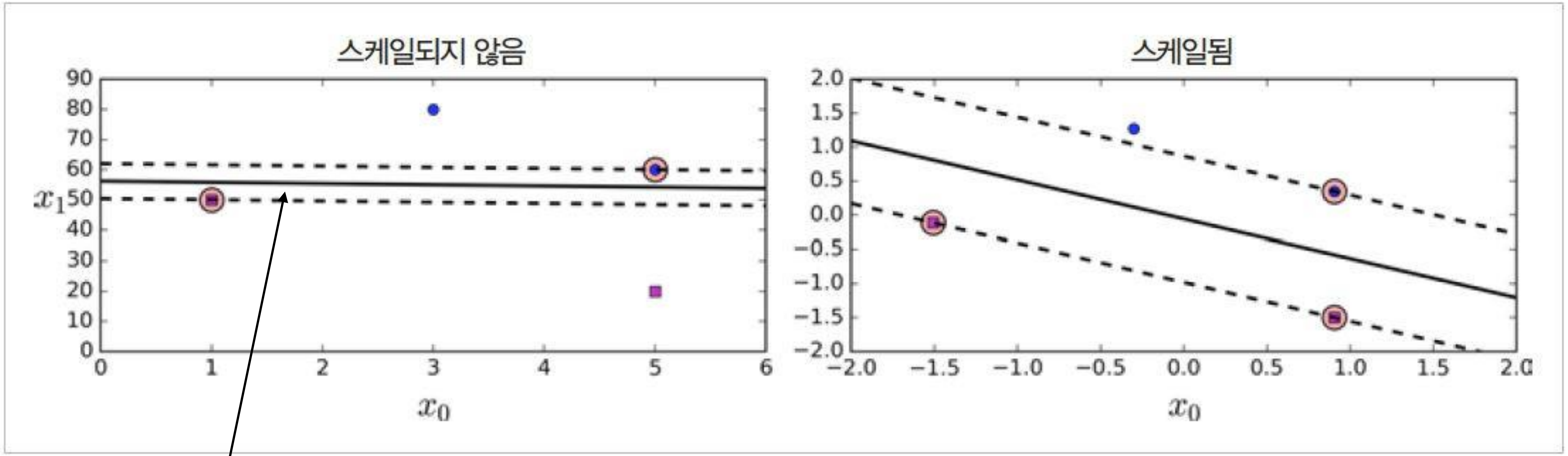
결정 경계가 가능한 샘플에서 멀리 떨어져 있습니다.
라지 마진(large margin) 분류라고도 부릅니다



경계에 위치한 샘플: 서포트 벡터(support vector)

SVM 특성 스케일링

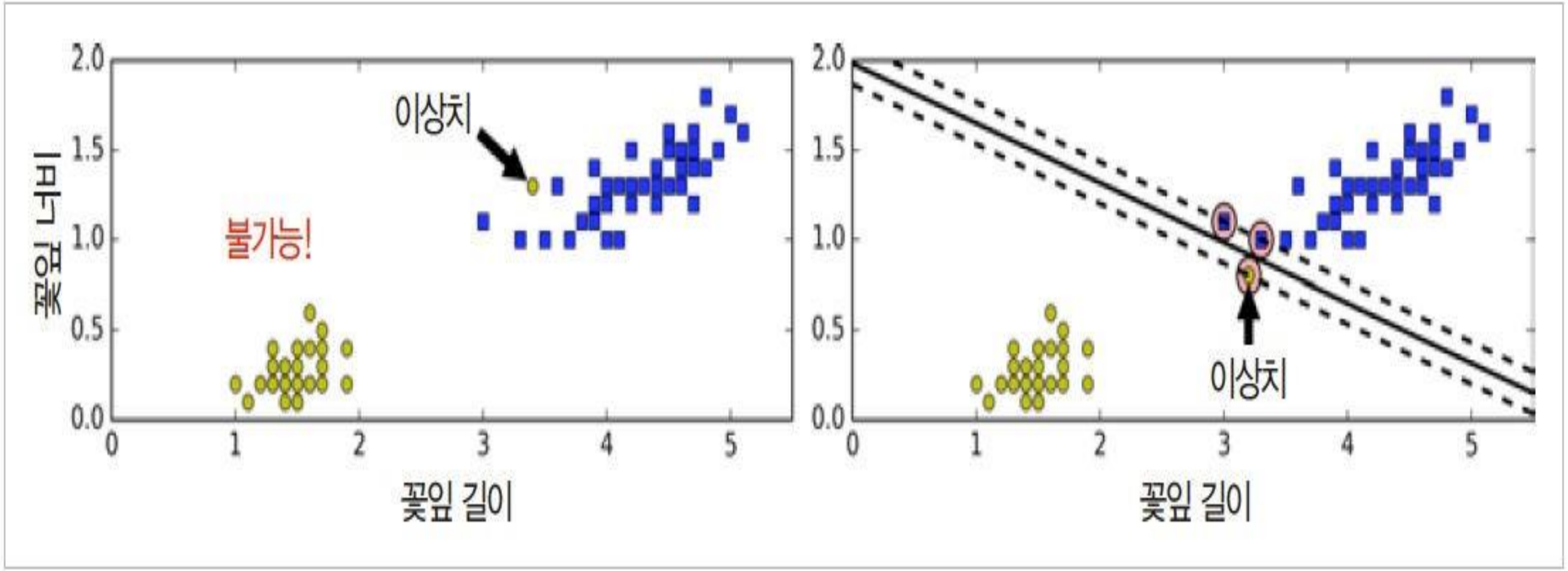
- SVM은 특성의 스케일에 영향을 많이 받음
  ➔ 사이킷런의 StandardScaler등을 이용해서 특징 정규화한 후 SVM



스케일이 큰 x1에 영향을 많이 받아 결정 경계가 수평에 가깝게 됩니다.

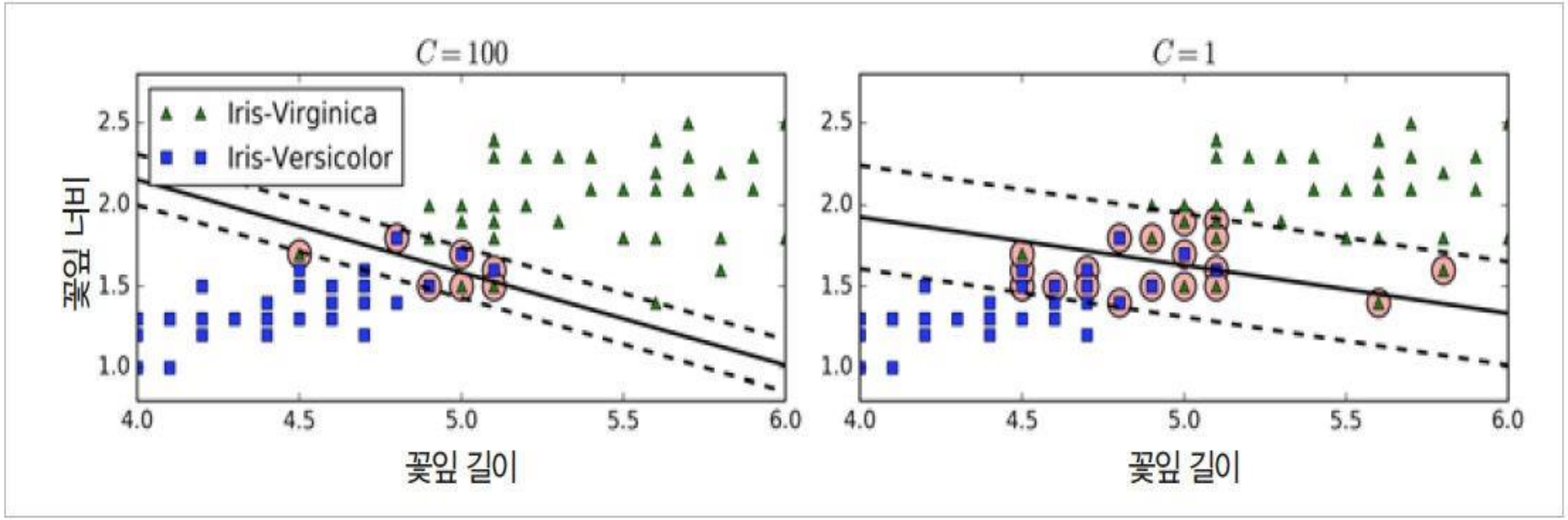
하드 마진(hard margin) 분류

- 모든 샘플을 마진 바깥 쪽에 분류 시킴
- 데이터가 선형적으로 구분될 수 있어야 함
- 이상치에 민감함



소프트 마진(soft margin) 분류

- 매개변수 C를 사용해서 마진을 조절할 수 있음 (마진 안에 있는 샘플들은 오류라고 판단하지 않음)
- 작은 C : 마진 커짐.   허용하는 마진 오류 늘어남. 모델 규제 커짐. 복잡도 작아짐
  큰 C : 마진 줄어듬.   허용하는 마진 오류 줄어듬. 모델 규제 작아짐. 복잡도 커짐



선형 SVM : LinearSVC 클래스

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

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # 꽃잎 길이, 꽃잎 너비
y = (iris["target"] == 2).astype(np.float64)  # Iris-Virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),
    ])

svm_clf.fit(X, y)
```

liblinear 이용

이진 분류 문제로 변경

규제값 C

```
>>> svm_clf.predict([[5.5, 1.7]])
array([ 1.])
```

비선형 문제

- PolynomialFeatures + LinearSVC

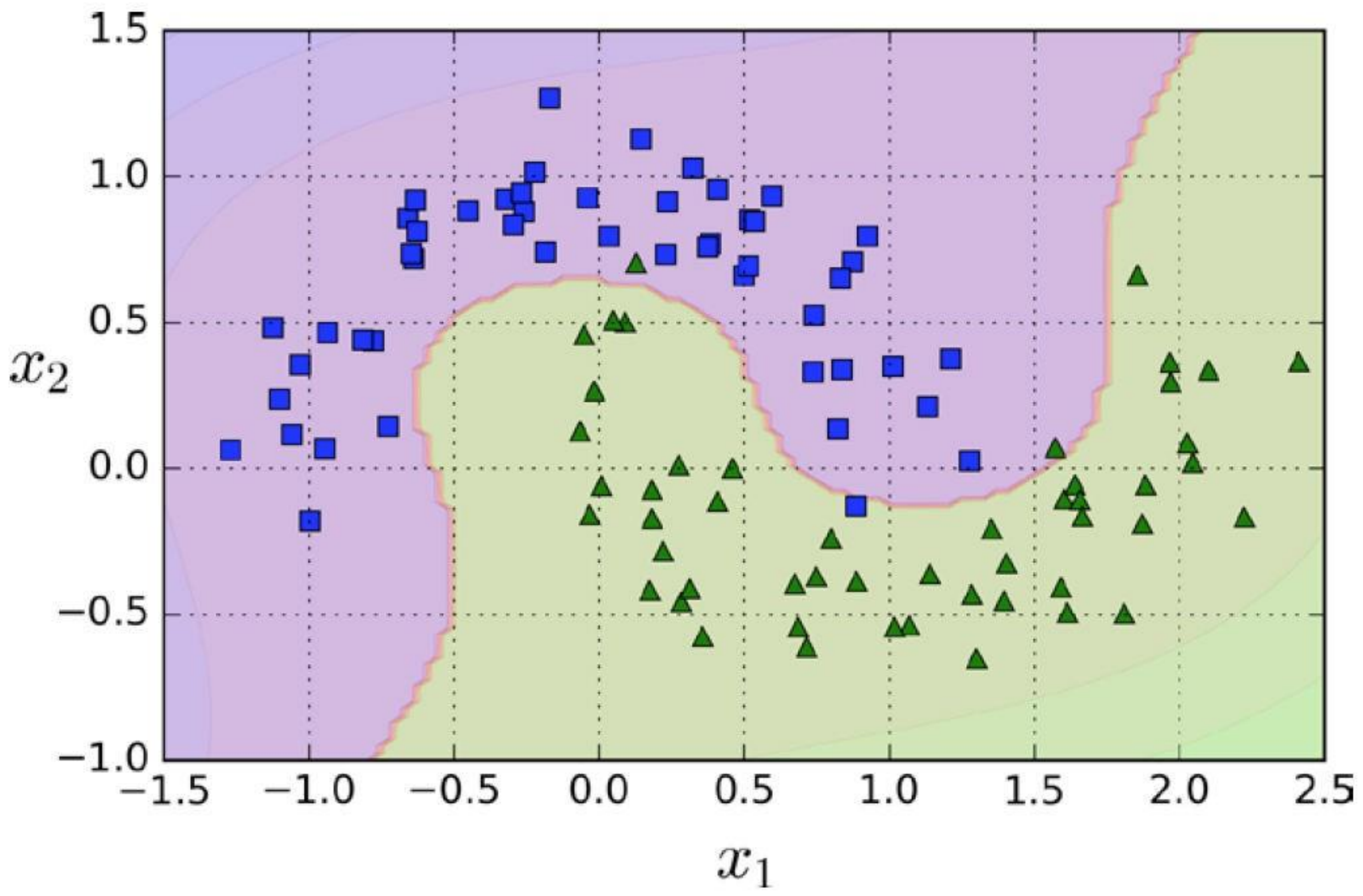
SVM에서는 이렇게 하는 대신 Kernel Trick을 이용함 (다음 슬라이드)

```python
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

polynomial_svm_clf = Pipeline([
        ("poly_features", PolynomialFeatures(degree=3)),
        ("scaler", StandardScaler()),
        ("svm_clf", LinearSVC(C=10, loss="hinge"))
    ])

polynomial_svm_clf.fit(X, y)
```

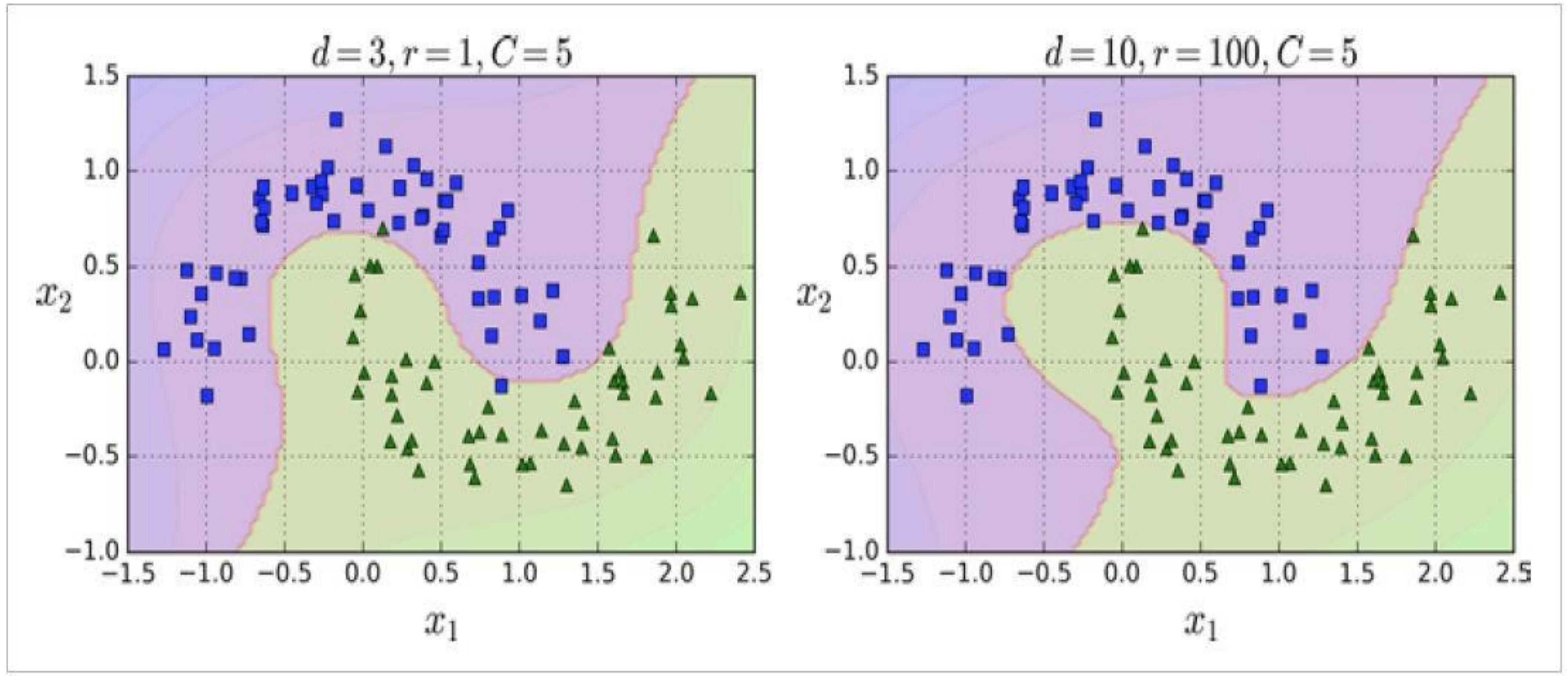


3차 다항식 특징 추가

커널 트릭(Kernel Trick)

- 실제 다항 특징을 추가하지 않고 비슷한 효과를 만드는 수학적 트릭
- 특징을 변환하는 대신 두 샘플사이의 유사도를 의미하는 커널을 정의
- 다항커널과 가우시안커널이 주로 사용됨.

- 다항 커널 $K(\mathbf{a},\mathbf{b}) = (\gamma\mathbf{a}^T \cdot \mathbf{b} + r)^d$

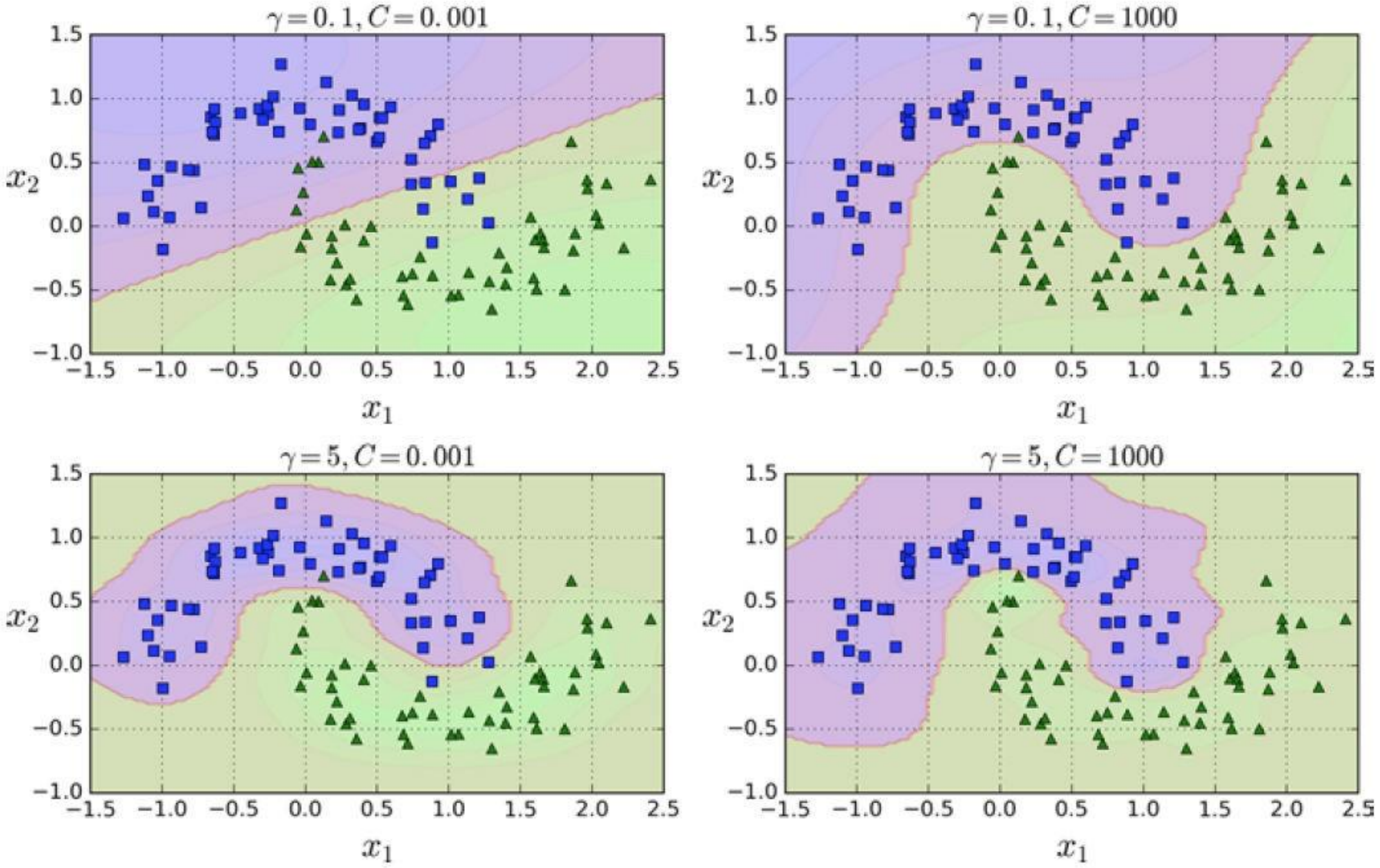



```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

가우시안 커널(Gaussian Kernel)

- RBF(Radial Basis Function)을 사용. $K(\mathbf{a},\mathbf{b}) = e^{-\gamma||\mathbf{a}-\mathbf{b}||^2}$
- 가장 많이 쓰는 커널
- γ가 커지면 모델 복잡도 커짐. c가 커지면 모델 복잡도 커짐

```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001
    ])
rbf_kernel_svm_clf.fit(X, y)
```



커널 선택 가이드

- 먼저 선형 커널을 시도 : LinearSVC가 SVC(kernel='linear') 보다 빠름
- 훈련 세트가 크면 LinearSVC를 선택
- 훈련 세트가 크지 않으면 가우시안 커널을 시도함

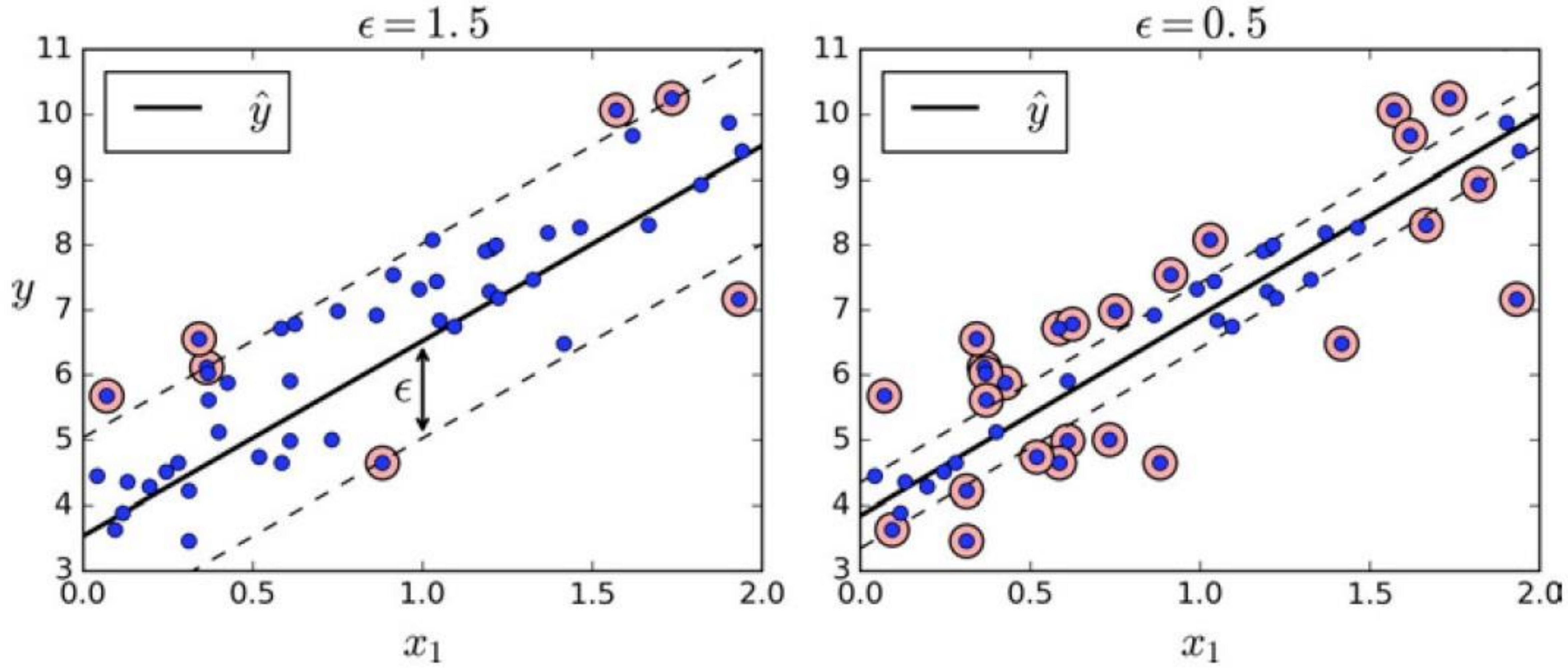
파이썬 클래스	시간 복잡도	외부 메모리 학습 지원	스케일 조정의 필요성	커널 트릭	
LinearSVC	$O(m \times n)$	아니오	예	아니오	liblinear 기반
SGDClassifier	$O(m \times n)$	예	예	아니오	
SVC	$O(m^2 \times n) \sim O(m^3 \times n)$	아니오	예	예	libsvm 기반

샘플 수에 민감

SVM 회귀 : 선형(LinearSVR)

- 사이킷런 svm.LinearSVR (선형회귀) 비선형회귀는 svm.SVR
- 분류와는 달리 마진 안에 최대한 많은 샘플을 포함하는 것이 목적임
- 마진 폭은 epsilon으로 조절
- 마진안에 샘플이 추가되어도 예측에 영향을 미치지 않음(e-insensitive).

```
from sklearn.svm import LinearSVR
svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```



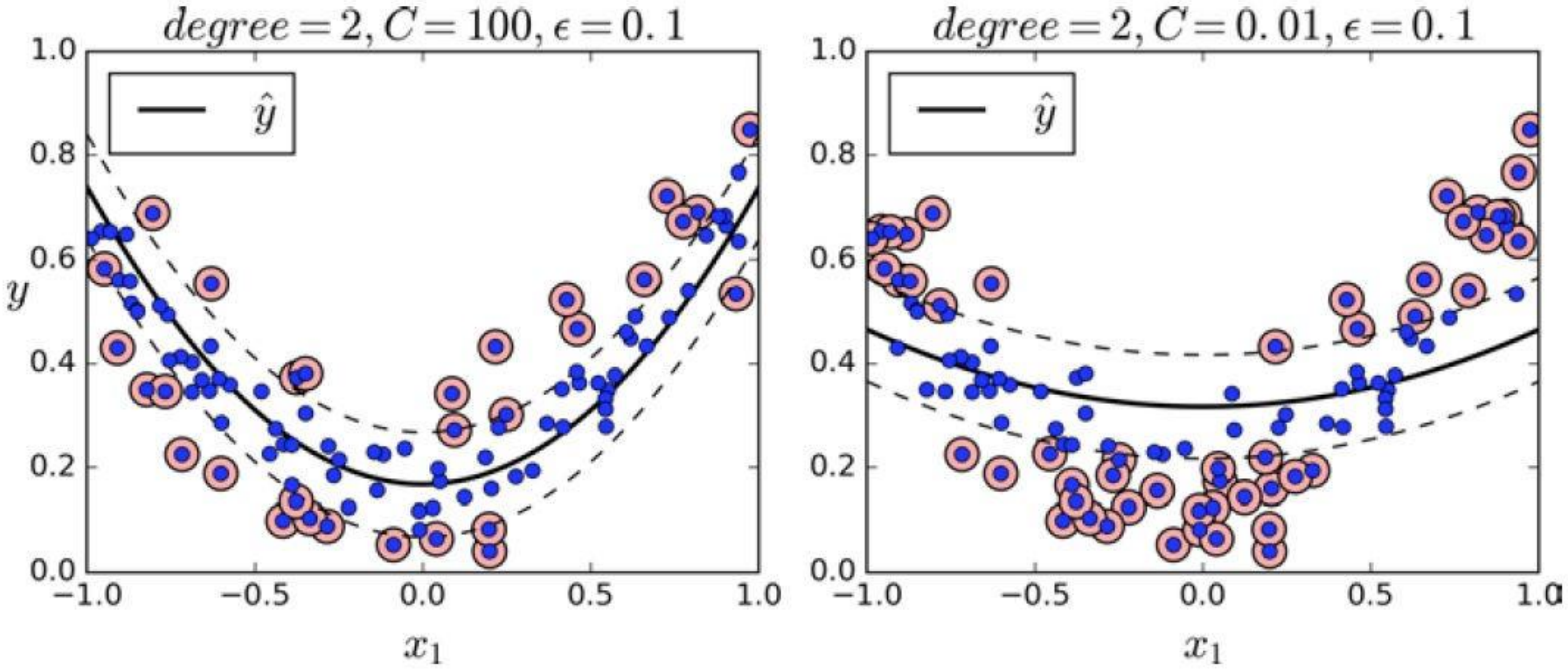
SVM 회귀 : 비선형(SVR)

```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```

SVC와 마찬가지로
훈련 세트의 크기가 커지면
속도가 많이 느려짐

규제 증가(C 감소)



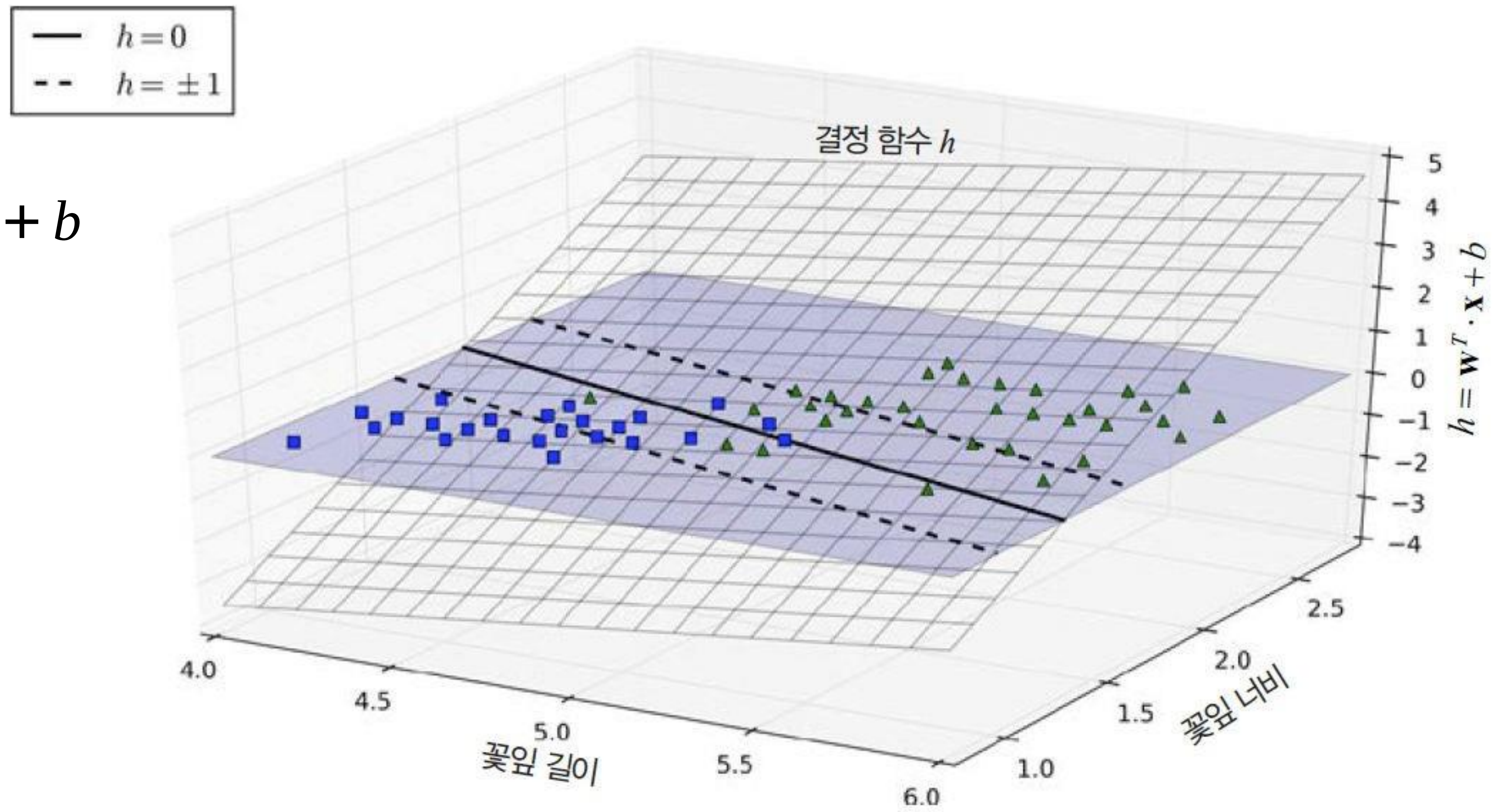
SVM 이론 : 선형 SVM 모델

- 마진 오류가 전혀 없거나(하드 마진), 어느 정도 오류를 가지면서(소프트 마진) 최대한 마진을 크게하는 **w**와 **b**를 찾는 것입니다.

결정함수 : $\mathbf{w}^T \cdot \mathbf{x} + b = w_1 x_1 + \cdots + w_n x_n + b$

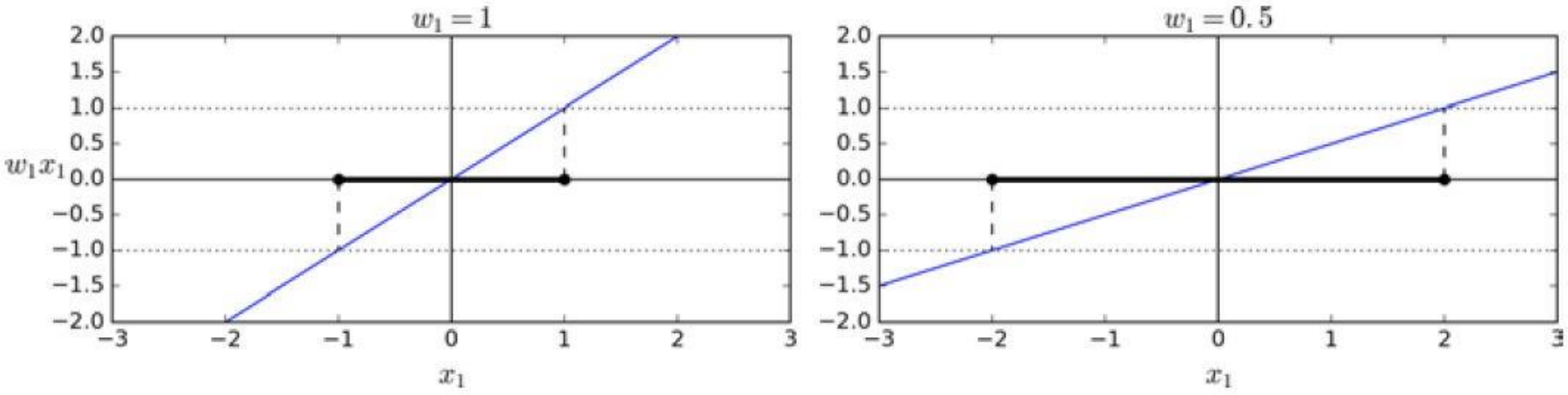
예측 :

$$\hat{y} = \begin{cases} 0 & \mathbf{w}^T \cdot \mathbf{x} + b < 0 \text{일 때} \\ 1 & \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \text{일 때} \end{cases}$$



SVM 이론 : 결정 함수의 가중치 효과

- 가중치가 줄어들면 결정 함수 기울기가 줄어들고 +1~-1 사이 마진이 늘어납니다.
- 마진을 크게 하기 위해 가중치를 최소화합니다.



SVM 이론 : 제약이 있는 최적화 문제

$$\hat{y} = \begin{cases} 0 & \mathbf{w}^T \cdot \mathbf{x} + b < 0 \text{일 때} \\ 1 & \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \text{일 때} \end{cases} \qquad \begin{matrix} t = -1 \\ t = 1 \end{matrix} \longrightarrow t^{(i)}(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b) \geq 1$$

- 하드 마진 선형 분류기의 목적 함수

$$\underset{\mathbf{w},b}{\text{minimize}} \ \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}$$

[조건] $i = 1, 2, \cdots, m$일 때 $t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \geq 1$

SVM 이론 : 소프트 마진을 위한 분류 목적 함수

- 슬랙 변수(slack variable)를 도입하여 각 샘플이 마진을 얼마나 위배할 지 정합니다
.

하이퍼파라미터(양쪽을 절충): C가 커지면 마진 오류가 중요해짐,
마진 폭이 줄고 가중치가 커짐, 규제 효과

가중치를 작게 만듬(마진 폭을 넓히는 효과)

마진 오류를 줄임(마진 폭을 줄이는 효과)

$$\underset{w,b,\varsigma}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{m} \varsigma^{(i)}$$

[조건] $i = 1, 2, \cdots, m$일 때 $t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \geq 1 - \varsigma^{(i)}$이고 $\varsigma^{(i)} \geq 0$

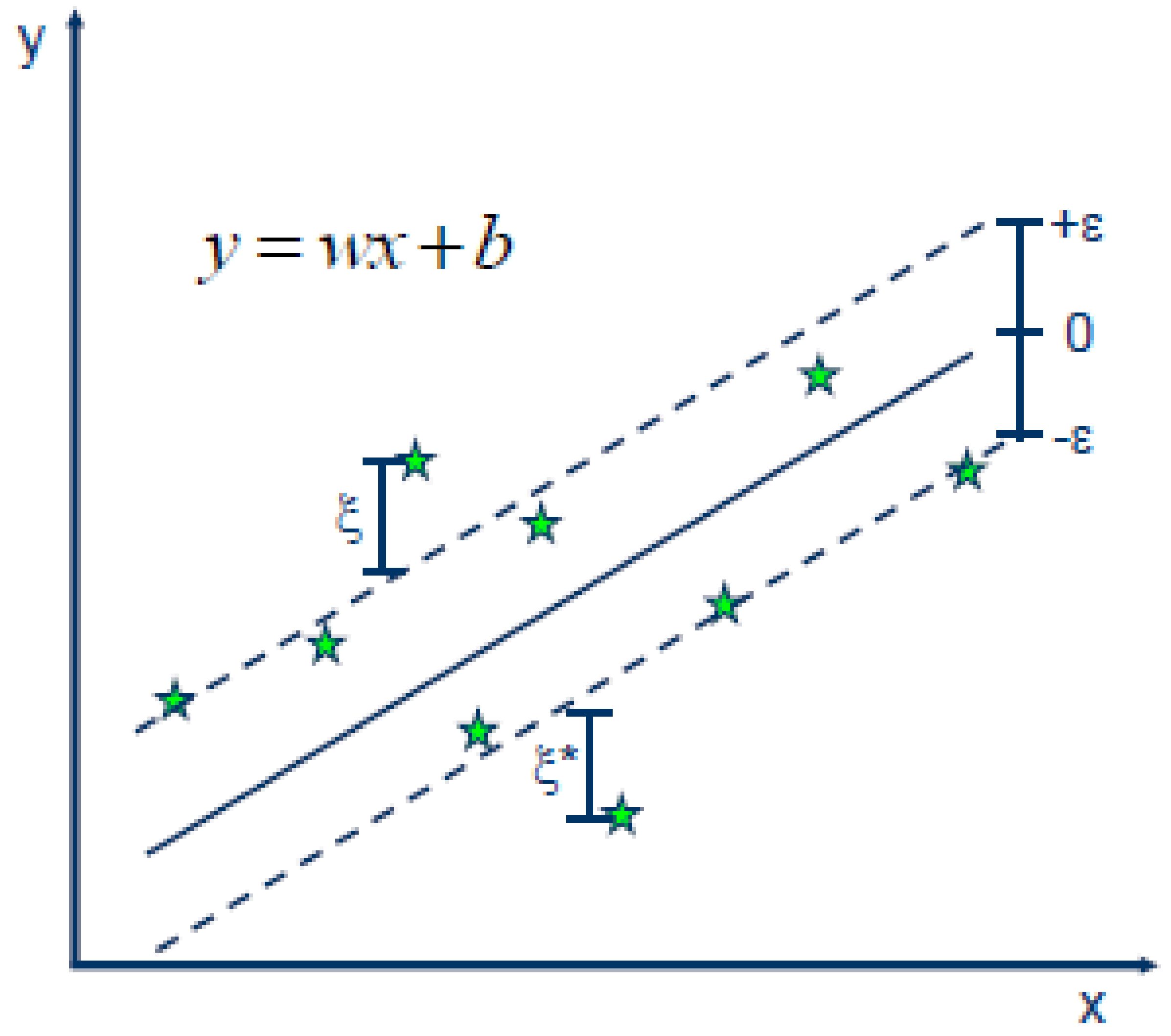
SVM 이론 : 소프트 마진을 위한 회귀 목적 함수

- 결정 경계 양쪽 마진에 해당하는 두 개의 슬랙 변수를 도입합니다

$$\underset{w,b,\varsigma,\zeta^*}{\text{minimize}} \ \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{m}(\zeta^{(i)} + \zeta^{(i)*})$$

[조건] $i = 1, 2, \cdots, m$일 때 $y^{(i)} - \left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \le \varepsilon + \varsigma^{(i)}$

$$\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) - y^{(i)} \le \varepsilon + \varsigma^{(i)*}$$



SVM 이론 : 콰드라틱 프로그래밍

- 제약 조건이 있는 볼록 함수의 이차 최적화 문제

$$\underset{\mathbf{w},\, b}{\text{minimize}}\ \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w}$$

[조건] $i = 1, 2, \cdots, m$일 때 $t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) \ge 1$

$$\underset{\mathbf{p}}{\text{minimize}}\ \frac{1}{2}\mathbf{p}^T \cdot \mathbf{H} \cdot \mathbf{p} + \mathbf{f}^T \cdot \mathbf{p}$$

[조건] $\mathbf{A} \cdot \mathbf{p} \le \mathbf{b}$

n+1

여기서
- $\mathbf{p}$는 n_p차원의 벡터(n_p = 모델 파라미터 수)
- $\mathbf{H}$는 $n_p \times n_p$ 크기 행렬 — 첫 번째는 0인 단위 행렬
- $\mathbf{f}$는 n_p차원의 벡터 — 0
- $\mathbf{A}$는 $n_c \times n_p$ 크기 행렬 (n_c = 제약 수)
- $\mathbf{b}$는 n_c차원의 벡터

m

1

SVM 이론 : 쌍대 문제

- 쌍대 문제(dual problem)는 원 문제(primal problem)의 하한값이거나 같습니다.
- LinearSVC, LinearSVR은 dual 매개변수의 기본값 True를 False로 바꾸면 원 문제 를 풉니다. SVC, SVR은 쌍대 문제를 풉니다.
- 샘플 개수가 특성 개수보다 작을 때 쌍대 문제가 더 빠릅니다.
- 커널 트릭을 가능하게 합니다.

SVM 이론 : 라그랑주 함수

- 제약이 있는 최적화 문제에서 제약을 목적 함수를 옮겨서 푸는 방법입니다.
- 목적 함수에서 라그랑주 승수를 곱한 제약을 뺍니다(라그랑주 함수).
- 최적화 문제에 해가 있다면 라그랑주 함수의 정류점(안장점) 중 하나여야 합니다. 즉 라그랑주 함수의 도함수가 0인 지점입니다.

SVM 이론 : 하드 마진 문제의 라그랑주 함수

maxmize a, minimize w, b

$$\mathscr{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} - \sum_{i=1}^{m} \alpha^{(i)} \left(t^{(i)} \left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b \right) - 1 \right)$$

여기서 $\alpha^{(i)} \geq 0 \qquad i = 1, 2, \cdots, m$에 대해

라그랑주 함수의 편도 함수 :

$$\nabla_{\mathbf{w}} \mathscr{L}(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_{i=1}^{m} \alpha^{(i)} t^{(i)} \mathbf{x}^{(i)}$$

$$\frac{\partial}{\partial b} \mathscr{L}(\mathbf{w}, b, \alpha) = -\sum_{i=1}^{m} \alpha^{(i)} t^{(i)}$$

정류점 :

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \mathbf{x}^{(i)}$$

$$\sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} = 0$$

SVM 이론 : 쌍대 형식

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} - \sum_{i=1}^{m} \alpha^{(i)}\left(t^{(i)}\left(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b\right) - 1\right)$$

여기서 $\alpha^{(i)} \geq 0 \qquad i = 1, 2, \cdots, m$에 대해

$\xleftarrow{\text{대입}}$

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \mathbf{x}^{(i)}$$

$$\sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} = 0$$

쌍대 형식 : minimize α

$$\mathcal{L}\left(\hat{\mathbf{w}}, \hat{b}, \alpha\right) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha^{(i)}\alpha^{(j)} t^{(i)} t^{(j)} x^{(i)^T} \cdot x^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\hat{b} = \frac{1}{n_s}\sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m}\left(t^{(i)} - \hat{\mathbf{w}}^T \cdot \mathbf{x}^{(i)}\right)$$

여기서 $\alpha^{(i)} \geq 0 \qquad i = 1, 2, \cdots, m$일 때

위 문제의 해를 $\hat{\alpha}$ 이라고 하면, 오른쪽 위 식에 대입해서 $\hat{\mathbf{w}}$ 를 구하고 오른쪽 아래 식에서 $\hat{b}$ 를 구한다.

SVM 이론 : 커널 함수

$$\phi(\mathbf{x}) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

$$\mathscr{L}\left(\hat{\mathbf{w}}, \hat{b}, \alpha\right) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^{(i)}\alpha^{(j)}t^{(i)}t^{(j)}x^{(i)^T}\cdot x^{(j)} - \sum_{i=1}^{m}\alpha^{(i)}$$

머서의 조건 : $K(\mathbf{a},\mathbf{b}) = \phi(\mathbf{a})^T \cdot \phi(\mathbf{b})$

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = \begin{pmatrix} a_1^2 \\ \sqrt{2}a_1a_2 \\ a_2^2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}b_1b_2 \\ b_2^2 \end{pmatrix} = a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2$$

$$= (a_1b_1 + a_2b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)^2 = (\mathbf{a}^T \cdot \mathbf{b})^2$$

샘플을 커널로 변환한후
커널공간에서 내적

= =

원래 샘플공간에서 내적 후
변환. 훨신 간단함

"커널 트릭"
Mercer 조건을 만족하는
커널들은 이런 성질을 가짐

Linear:	$K(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T \cdot \boldsymbol{b}$
Polynomial:	$K(\boldsymbol{a}, \boldsymbol{b}) = (\gamma \boldsymbol{a}^T \cdot \boldsymbol{b} + r)^d$
Gaussian RBF:	$K(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma \lVert \boldsymbol{a} - \boldsymbol{b} \rVert^2)$
Sigmoid:	$K(\boldsymbol{a}, \boldsymbol{b}) = \tanh(\gamma \boldsymbol{a}^T \cdot \boldsymbol{b} + r)$

SVM 이론 : 커널 트릭을 사용한 예측

$$h_{\hat{\mathbf{w}}\hat{b}}\left(\phi\left(\mathbf{x}^{(n)}\right)\right) = \hat{\mathbf{w}}^T \cdot \phi\left(\mathbf{x}^{(n)}\right) + \hat{b} = \left(\sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \phi\left(\mathbf{x}^{(i)}\right)\right)^T \cdot \phi\left(\mathbf{x}^{(n)}\right) + \hat{b}$$

$$= \sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \left(\phi\left(\mathbf{x}^{(i)}\right)^T \cdot \phi\left(\mathbf{x}^{(n)}\right)\right) + \hat{b}$$

$$= \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m} \hat{\alpha}^{(i)} t^{(i)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(n)}\right) + \hat{b}$$

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}^{(i)} t^{(i)} \mathbf{x}^{(i)}$$

$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m} \left(t^{(i)} - \hat{\mathbf{w}}^T \cdot \mathbf{x}^{(i)} \right)$$

$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m} \left(t^{(i)} - \hat{\mathbf{w}}^T \cdot \phi\left(\mathbf{x}^{(i)}\right)\right) = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m} \left(t^{(i)} - \left(\sum_{j=1}^{m} \hat{\alpha}^{(j)} t^{(j)} \phi\left(\mathbf{x}^{(j)}\right)\right)^T \cdot \phi\left(\mathbf{x}^{(i)}\right)\right)$$

$$= \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m} \left(t^{(i)} - \sum_{\substack{j=1 \\ \alpha^{(j)}>0}}^{m} \hat{\alpha}^{(j)} t^{(j)} K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)\right)$$

감사합니다