핸즈온 머신러닝

Chap. 5 서포트 벡터 머신

한나영
bandi12424@naver.com

서포트 벡터 머신 (SVM)

- 선형, 비선형 분류, 회귀, 이상치 탐색에 사용할 수 있는 다목적 ML 모델
- 복잡한 분류 문제, 작거나 중간 크기의 데이터셋에 적합

선형 SVM

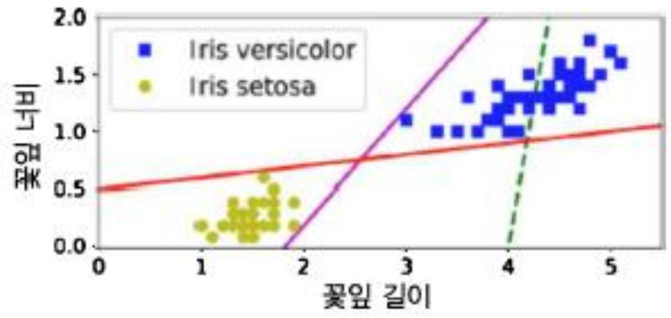
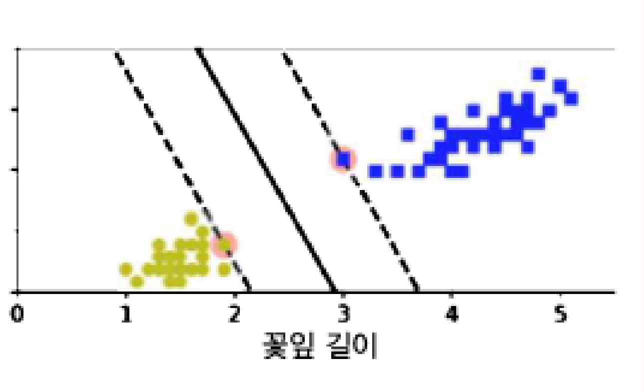


그림 5-1 라지 마진 분류

두 클래스가 선형적으로 구분. 그러나,

- 점선으로 나타난 결정 경계를 만든 모델: 클래스를 적절히 분류하지 못함
- 실선으로 나타난 모델: 경계가 샘플에 너무 가까움

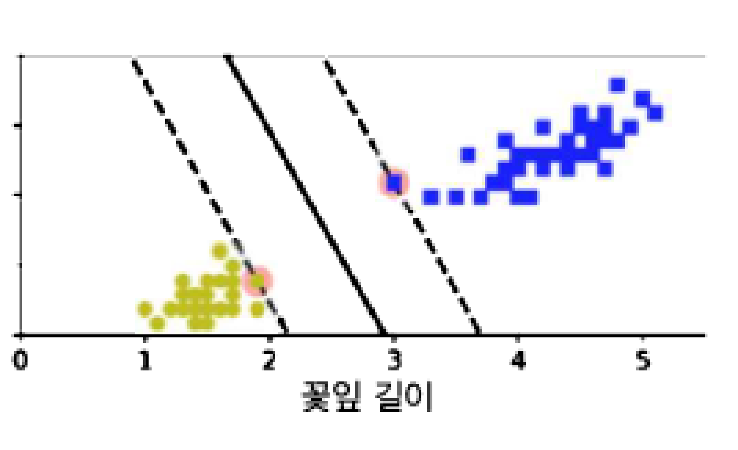
→ 새로운 샘플에 잘 작동하지 못할 가능성 ↑



실선: SVM 분류기의 결정 경계

→ 1. 분류 잘함

2. 가장 가까운 훈련 샘플로부터 가능한 한 멀리 떨어져있음



SVM 분류기 = 클래스 사이의 **가장 폭 넓은 도로**를 찾는 것 (라지 마진 분류)

도로의 바깥쪽에 훈련 샘플 추가 → 결정 경계 영향 X
도로 경계에 위치한 샘플 (**서포트 벡터**) → 결정 경계 결정

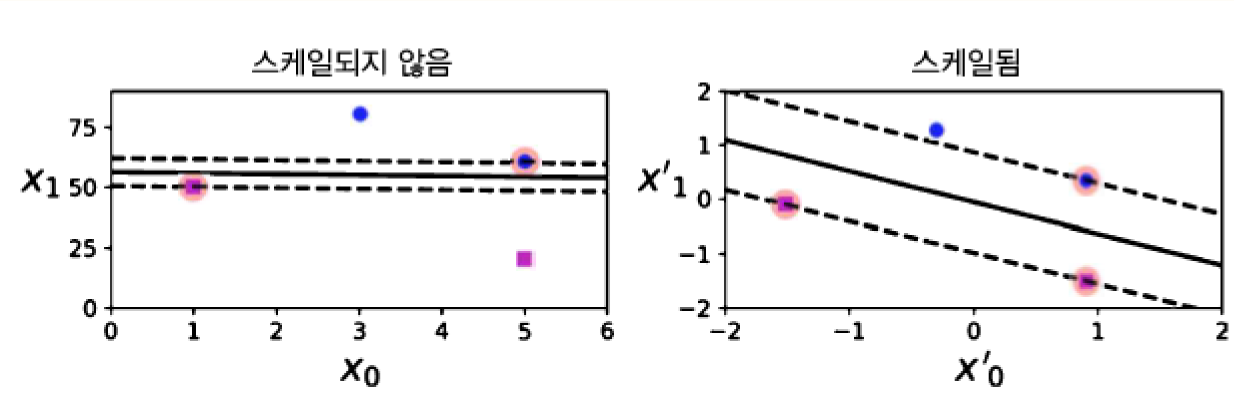


그림 5-2 특성 스케일에 따른 민감성

SVM은 특성의 스케일에 민감

b : 편향, $\boldsymbol{w}$: 가중치 벡터

선형 SVM 분류기 모델

$\boldsymbol{w}^{T}\boldsymbol{x} + b = w_1 x_1 + \ldots + w_n x_n + b$ 계산하여 새로운 샘플 $\boldsymbol{x}$의 클래스 예측

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

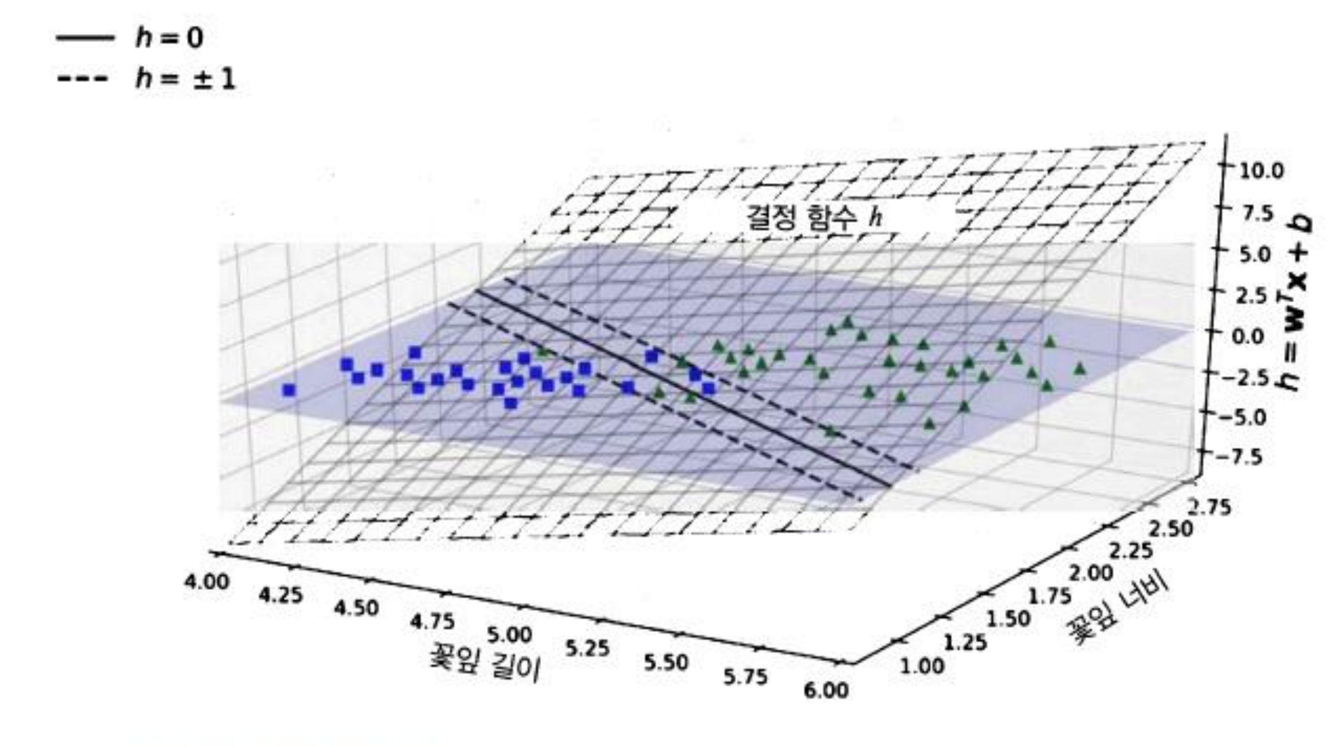


그림 5-12 iris 데이터셋의 결정 함수

결정 경계: 결정 함수의 값이 0인 점들로 구성,
두 평면의 교차점인 직선

임의의 점 x에서 초평면 $w^T x+b=0$ 까지의 거리 r

$$r = \frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b|}{||\boldsymbol{w}||}$$

서포트 벡터에서
초평면에 이르는 거리의 합 $\gamma = \frac{2}{||\boldsymbol{w}||}$, $||\boldsymbol{w}||$ ↓ 마진 γ ↑

$\boxed{||w||^2 \text{ 을 최소화하는 문제}}$

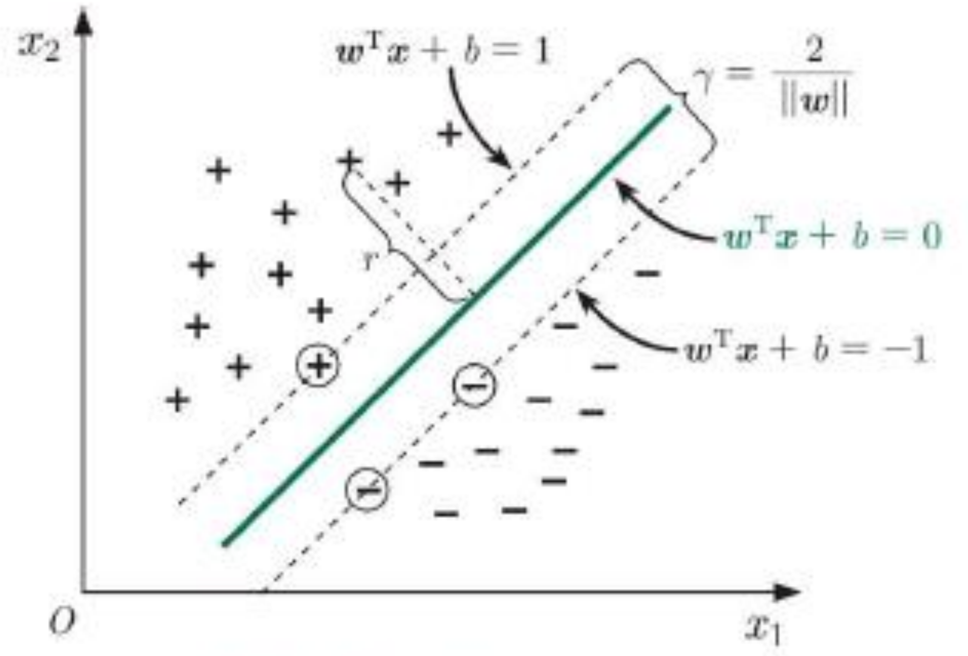


그림 6.2 서포트 벡터와 마진

하드 마진 분류 : 모든 샘플이 정확하게 분류되도록 하는 것

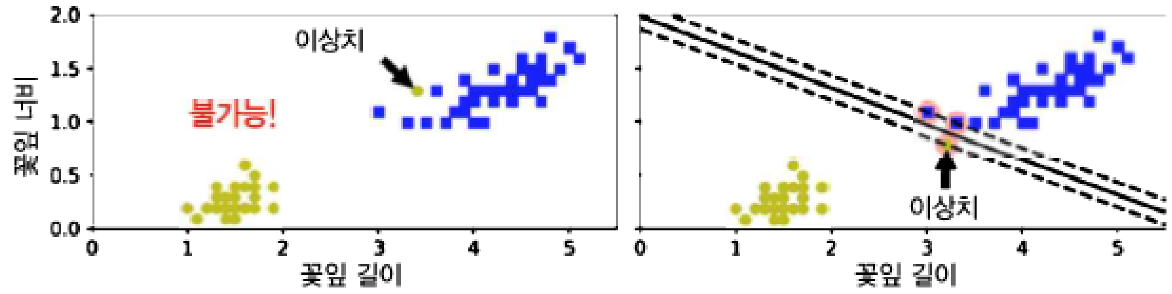


그림 5-3 이상치에 민감한 하드 마진

문제점

- 데이터가 선형적으로 구분될 수 있어야 제대로 동작
- 이상치에 민감

약간의 오류를 허용

소프트 마진 분류

선형 SVM 분류기 훈련	
하드마진	소프트 마진
마진 오류가 하나도 발생하지 않음	제한적인 마진 오류를 가지면서 가능한 한 마진을 크게 하는 $\boldsymbol{w}$와 b를 찾음

식 5-3 하드 마진 선형 SVM 분류기의 목적 함수

$$\underset{\mathbf{w},b}{\text{minimize}} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} = (1/2)||\boldsymbol{w}||^2$$

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

음성 샘플 일 때, $t^{(i)} = -1$,
양성 샘플 일 때, $t^{(i)} = 1$

식 5-4 소프트 마진 선형 SVM 분류기의 목적 함수[20]

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

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

슬랙변수 $\zeta^i \geq 0$ 도입: i번째 샘플이 얼마나 마진을 위반할지 결정

Note.

마진 오류를 최소화하려면 슬랙 변수의 값이 작아야 함
마진을 크게 하기 위해서는 $(1/2)||\boldsymbol{w}||^2$ 값이 작아야 함

동시에 만족할 수 없음
∴ 두 목표 사이의 트레이드 오프를 정의하는 C 값 설정

하드 마진 & 소프트 마진 문제 = 선형적인 제약 조건이 있는 볼록 함수의 이차 최적화 문제 (**QP 문제**)

식 5-5 QP 문제

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

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

$$\text{여기서} \begin{cases} \mathbf{p}\text{는 } n_p \text{차원의 벡터} \ (n_p = \text{모델 파라미터 수}) \\ \mathbf{H}\text{는 } n_p \times n_p \text{ 크기 행렬} \\ \mathbf{f}\text{는 } n_p \text{차원의 벡터} \\ \mathbf{A}\text{는 } n_c \times n_p \text{ 크기 행렬} \ (n_c = \text{제약 수}) \\ \mathbf{b}\text{는 } n_c \text{차원의 벡터} \end{cases}$$

- 하드 마진: 아래와 같이 QP 파라미터 지정

- $n_p = n+1$, 여기서 n은 특성 수입니다(편향 때문에 +1이 추가되었습니다).
- $n_c = m$, 여기서 m은 훈련 샘플 수입니다.
- $\mathbf{H}$는 $n_p \times n_p$ 크기이고 왼쪽 맨 위의 원소가 0(편향을 제외하기 위해)인 것을 제외하고는 단위행렬입니다.
- $\mathbf{f} = \mathbf{0}$, 모두 0으로 채워진 n_p차원의 벡터입니다.
- $\mathbf{b} = \mathbf{1}$, 모두 1로 채워진 n_c차원의 벡터입니다.
- $a^{(i)} = -t^{(i)}\dot{\mathbf{x}}^{(i)}$, 여기서 $\dot{\mathbf{x}}^{(i)}$는 편향을 위해 특성 $\dot{\mathbf{x}}_0 = 1$을 추가한 $\mathbf{x}^{(i)}$와 같습니다.

결과 벡터 p = (b, $w_1, w_2, \dots w_n$)

- 소프트 마진: 아래와 같이 QP 파라미터 지정

- $\mathbf{H}$는 $\mathbf{H}'$의 오른쪽에 0으로 채워진 m개의 열이 있고 아래에 0으로 채워진 m개의 열이 있는 행렬입니다.

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}' & 0 & \cdots \\ 0 & 0 & \\ \vdots & & \ddots \end{pmatrix}$$

- $\mathbf{f}$는 $\mathbf{f}'$에 하이퍼파라미터 C와 동일한 값의 원소 m개가 추가된 벡터입니다.
- $\mathbf{b}$는 $\mathbf{b}'$에 값이 0인 원소 m개가 추가된 벡터입니다.
- $\mathbf{A}$는 $\mathbf{A}'$의 오른쪽에서 $-\mathbf{I}_m$이 추가되고 바로 그 아래에 $-\mathbf{I}_m$이 추가되며 나머지는 0으로 채워진 행렬입니다($\mathbf{I}_m$은 $m \times n$ 단위 행렬).

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' & -\mathbf{I}_m \\ \mathbf{0} & -\mathbf{I}_m \end{pmatrix}$$

원 문제 (primal problem) 라는 제약이 있는 최적화 문제 → 쌍대문제 (dual problem)으로 표현할 수 있음

원 문제

$$\begin{array}{ll} \min_x & f(x) \\ \text{subject to} & h_i(x) \le 0, i = 1, \cdots, m \\ & \ell_j(x) = 0, j = 1, \cdots, r. \end{array}$$

원 문제의 쌍대문제 원 문제

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \le b \\ & x \ge 0 \end{array} \iff \begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \ge c \\ & y \ge 0 \end{array}$$

일반적으로 원 문제의 해 ≥ 쌍대문제의 해
그러나, SVM 문제는 원 문제의 해 = 쌍대문제의 해 (strong duality)

식 C-4 SVM 문제의 쌍대 형식 이 라그랑주 함수를 최대값을 찾음

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

여기서 $\alpha^{(i)} \ge 0 \quad i = 1, 2, \cdots, m$이고 $\sum_{i=1}^{m} \alpha^{(i)} t^{(i)} = 0$일 때

식 5-6 선형 SVM 목적 함수의 쌍대 형식

$$\underset{\alpha}{\text{minimize}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} t^{(i)} t^{(j)} \mathbf{x}^{(i)^T} \mathbf{x}^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}$$

[조건] $i = 1, 2, \cdots, m$일 때 $\alpha^{(i)} \ge 0$

식 5-7 쌍대 문제에서 구한 해로 원 문제의 해 계산하기

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

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

Note. 훈련 샘플 수 < 특성 개수 이면,

- 쌍대 문제가 더 빠름
- 원 문제에서는 적용 안되는 커널 트릭을 가능하게 함

비선형 SVM 분류

샘플을 원시 공간에서 더 높은 차원의 특성 공간으로 투영, 특성 공간 내에서 선형 분리가 가능하도록 함

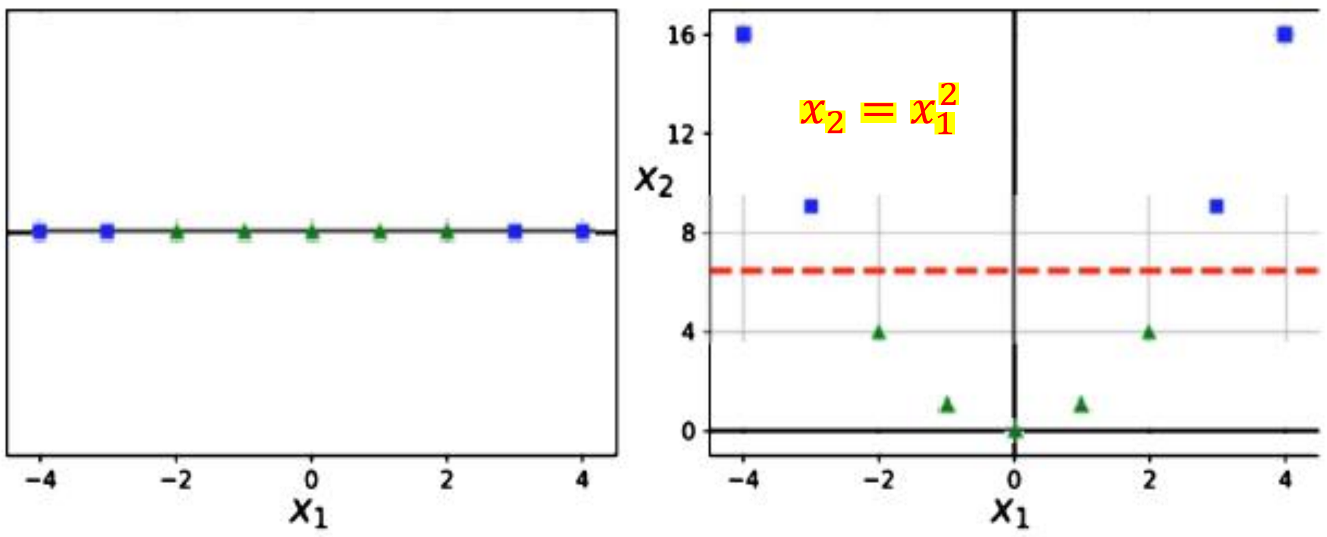


그림 5-5 특성을 추가하여 선형적으로 구분되는 데이터셋 만들기

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

polynomial_svm_clf = Pipeline([
        ("poly_features", PolynomialFeatures(degree=3)),
        ("scaler", StandardScaler()),
        ("svm_clf", LinearSVC(C=10, loss="hinge", random_state=42))
    ])

polynomial_svm_clf.fit(X, y)
```

다항식 커널

```
from sklearn.svm import SVC

poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5)) #coef0: 다항식 커널의 상수항, 모델이 차수에 얼마나 영향을 받을지를 조절
    ])
```

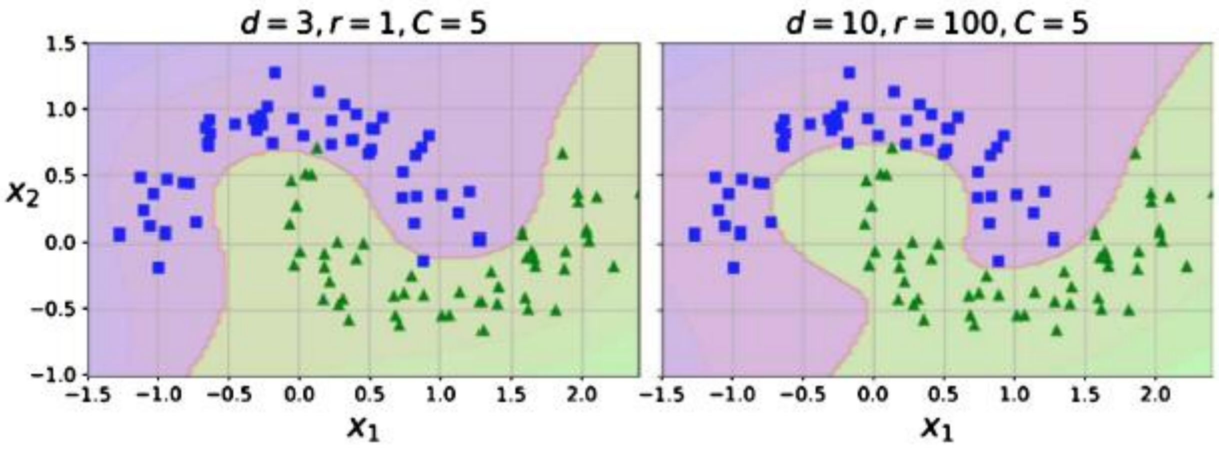


그림 5-7 다항식 커널을 사용한 SVM 분류기

유사도 특성

각 샘플이 특정 랜드마크와 얼마나 닮았는지를 측정하는 **유사도 함수로 계산한 특성을 추가**

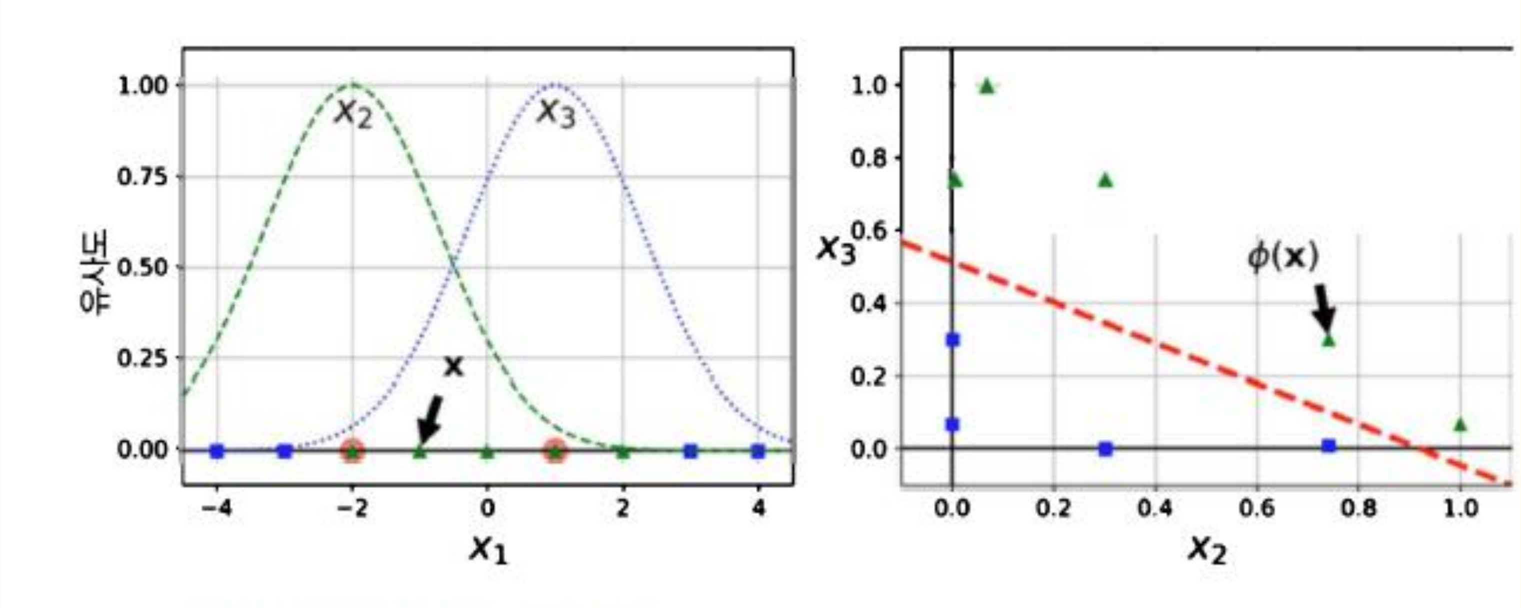


그림 5-8 가우시안 RBF를 사용한 유사도 특성

- 랜드마크 추가

  $\ell_1: x_1 = -2, \ \ell_2: x_1 = 1$

- 유사도 함수: 가우시안 RBF

  식 5-1 가우시안 RBF

  $$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \|\mathbf{x} - \ell\|^2\right)$$

  ℓ: 랜드마크 지점

  랜드마크와 가까울수록 1에 가까워짐

- 샘플 이용, 특성 추가: $\mathbf{x}: x_1 = -1$
  - $\mathrm{x}_2 = \exp(-0.3 \times ||\mathbf{x} - \ell_1||^2) \cong 0.74$
  - $\mathrm{x}_3 = \exp(-0.3 \times ||\mathbf{x} - \ell_2||^2) \cong 0.3$

  $\rightarrow \phi(\mathbf{x}) = (\mathrm{x}_2, \ \mathrm{x}_3)$

Q. 랜드마크를 어떻게 선택?

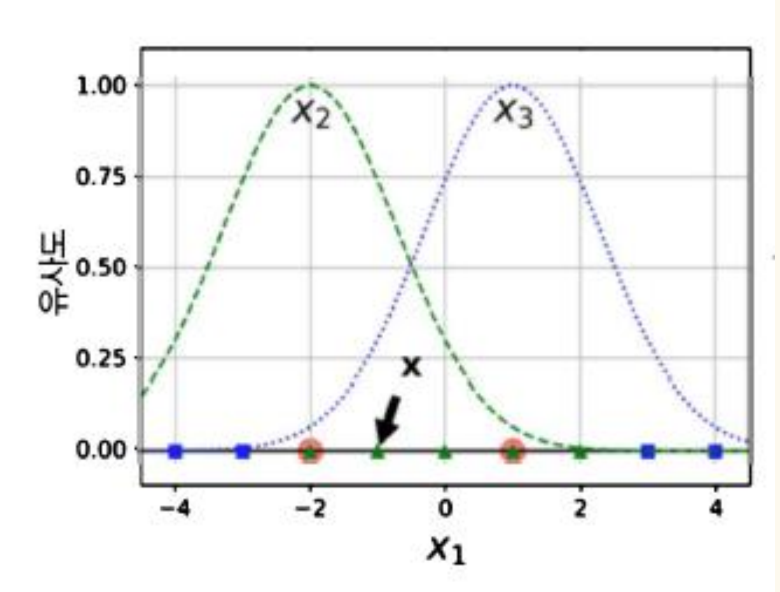
(simple) 데이터셋에 있는 모든 샘플 위치에 랜드마크를 설정 → (+) 차원 ↑, 훈련세트 선형적으로 구분

(-) 훈련 세트 크기 만큼 특성이 생성

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

식 5-1 가우시안 RBF

$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \left\| \mathbf{x} - \ell \right\|^2\right)$$



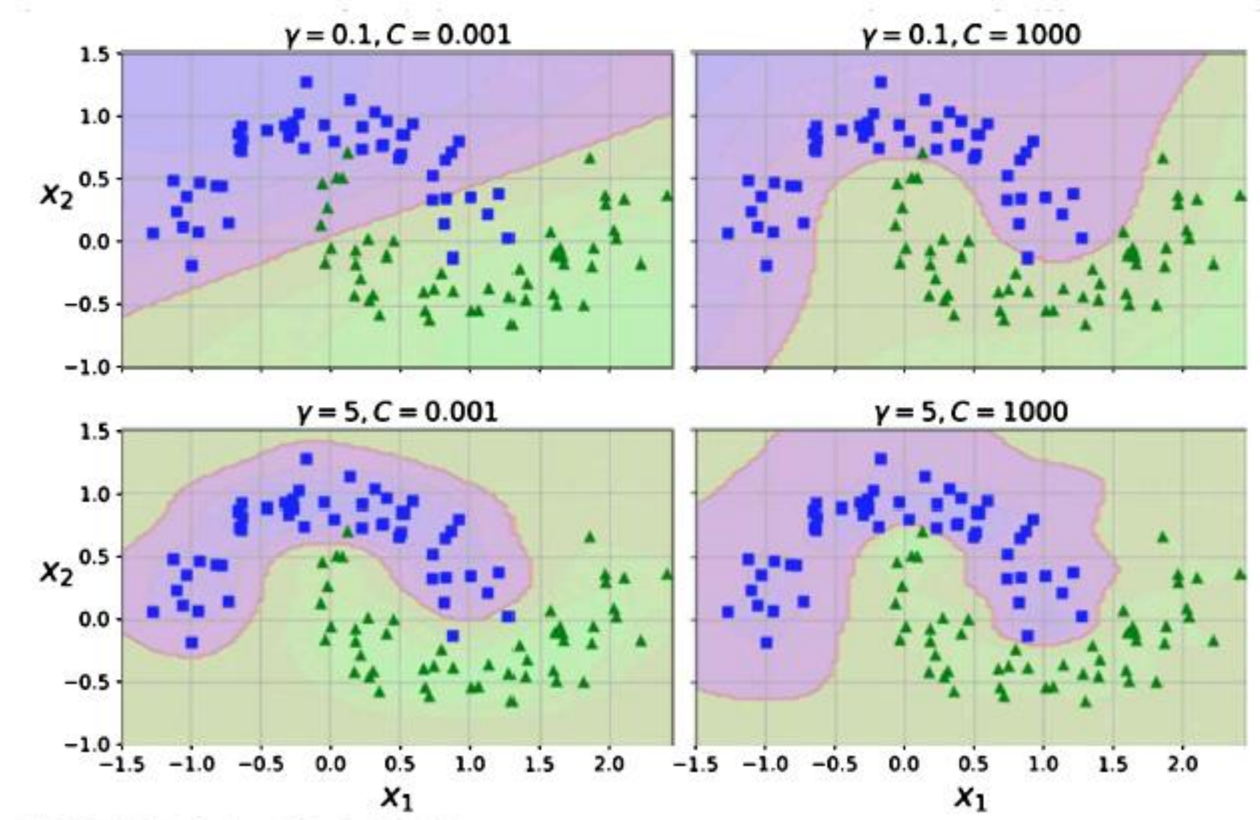


그림 5-9 RBF 커널을 사용한 SVM 분류기

γ ↑ 그래프 폭↓ 결정경계 불규칙성↑

→ 과대적합 시, γ↓

과소적합 시, γ↑

$\boxed{\phi\left(\mathbf{x}^{(i)}\right)^T \phi\left(\mathbf{x}^{(j)}\right)}$

식 5-6 선형 SVM 목적 함수의 쌍대 형식

$$\underset{\alpha}{\text{minimize}} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^{(i)}\alpha^{(j)}t^{(i)}t^{(j)}\underline{\mathbf{x}^{(i)^T}\mathbf{x}^{(j)}} - \sum_{i=1}^{m}\alpha^{(i)}$$

[조건] $i = 1, 2, \cdots, m$일 때 $\alpha^{(i)} \geq 0$

커널: 변환 Φ 을 계산하지 않고 원래 벡터 a, b에 기반하여 $\Phi(a)^T\Phi(b)$를 계산할 수 있는 함수

식 5-10 일반적인 커널

선형: $K(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}$

다항식: $K(\mathbf{a},\mathbf{b}) = \left(\gamma\mathbf{a}^T\mathbf{b} + r\right)^d$

가우시안 RBF: $K(\mathbf{a},\mathbf{b}) = \exp\left(-\gamma\|\mathbf{a}-\mathbf{b}\|^2\right)$

시그모이드[26]: $K(\mathbf{a},\mathbf{b}) = \tanh\left(\gamma\mathbf{a}^T\mathbf{b} + r\right)$

식 5-9 2차 다항식 매핑을 위한 커널 트릭

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

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

식 5-8 2차 다항식 매핑

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

3차원

실제로 훈련 샘플을 변환할 필요가 전혀 없다

커널 트릭: 실제로는 특성을 추가하지 않으면서 다항식 특성을 많이 추가한 것과 같은 결과를 줌

식 5-11 커널 SVM으로 예측하기

$$
\begin{aligned}
h_{\hat{\mathbf{w}},\hat{b}}\left(\phi\left(\mathbf{x}^{(n)}\right)\right) &= \hat{\mathbf{w}}^T\phi\left(\mathbf{x}^{(n)}\right)+\hat{b} = \left(\sum_{i=1}^{m}\hat{\alpha}^{(i)}t^{(i)}\phi\left(\mathbf{x}^{(i)}\right)\right)^T\phi\left(\mathbf{x}^{(n)}\right)+\hat{b} \\
&= \sum_{i=1}^{m}\hat{\alpha}^{(i)}t^{(i)}\left(\phi\left(\mathbf{x}^{(i)}\right)^T\phi\left(\mathbf{x}^{(n)}\right)\right)+\hat{b} \\
&= \sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m}\hat{\alpha}^{(i)}t^{(i)}\underline{K\left(\mathbf{x}^{(i)},\mathbf{x}^{(n)}\right)}+\hat{b}
\end{aligned}
$$

원시 공간에서의 내적을 계산!

Note.
서포트 벡터만 $\alpha^{(i)} \neq 0$ 이므로 ,
예측을 만드는 데는 서포트 벡터와 입력 벡터 간의 점곱(내적)만 계산

식 5-12 커널 트릭을 사용한 편향 계산

$$
\begin{aligned}
\hat{b} &= \frac{1}{n_s}\sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m}\left(t^{(i)}-\hat{\mathbf{w}}^T\phi\left(\mathbf{x}^{(i)}\right)\right) = \frac{1}{n_s}\sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m}\left(t^{(i)}-\left(\sum_{j=1}^{m}\hat{\alpha}^{(j)}t^{(j)}\phi\left(\mathbf{x}^{(j)}\right)\right)^T\phi\left(\mathbf{x}^{(i)}\right)\right) \\
&= \frac{1}{n_s}\sum_{\substack{i=1 \\ \hat{\alpha}^{(i)}>0}}^{m}\left(t^{(i)}-\sum_{\substack{j=1 \\ \hat{\alpha}^{(j)}>0}}^{m}\hat{\alpha}^{(j)}t^{(j)}\underline{K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)}\right)
\end{aligned}
$$

계산 복잡도

표 5-1 SVM 분류를 위한 사이킷런 파이썬 클래스 비교

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

LinearSVC	SVC
liblinear 라이브러리 (선형 SVM을 위한 최적화된 알고리즘 구현)	libsvm 라이브러리
커널트릭 지원하지 않음	커널 트릭 알고리즘 구현

SVM 회귀

일정한 마진 오류 안에서 두 클래스 간의 도로 **폭이 가능한 한 최대**가 되도록 하는 대신,
제한된 마진 오류 안에서 도로 **안에 가능한 한 많은 샘플**이 들어가도록

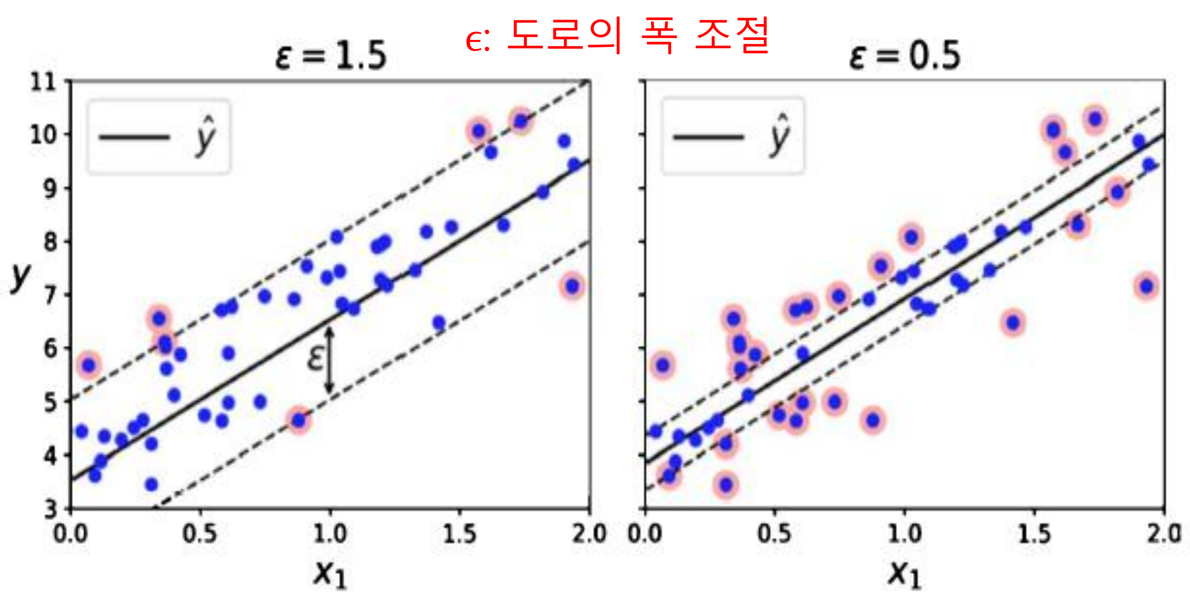


그림 5-10 SVM 회귀

마진 안에서는 훈련 샘플이 추가되어도
모델의 예측에는 영향이 없음

즉, ϵ 에 민감하지 않음

```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5, random_state=42)
svm_reg.fit(X, y)
```

선형 SVM 회귀

```
from sklearn.svm import SVR

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

비선형 SVM 회귀

온라인 SVM

새로운 샘플이 생겼을 때 점진적으로 학습

식 5-13 선형 SVM 분류기 비용 함수

$$J(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\max\left(0, 1 - t^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right)\right)$$

마진을 크게 만듦

모든 마진 오류를 계산

이 비용 함수를 최소화하기 위한 경사 하강법 사용
* 경사 하강법은QP기반의 방법보다 훨씬 느리게 수렴

힌지 손실 함수