Machine Learning

Support Vector Machine

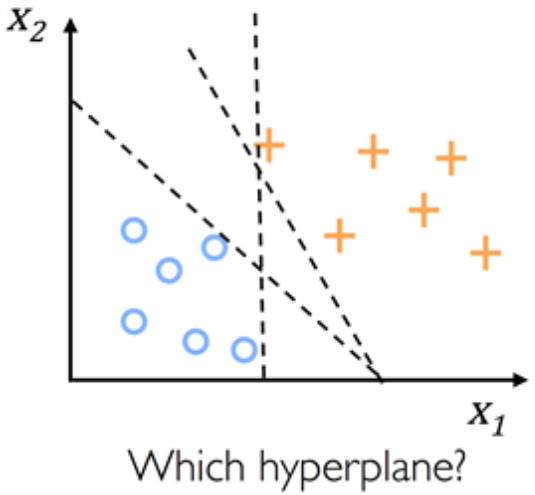
김선녕(ksycafe@gmail.com)

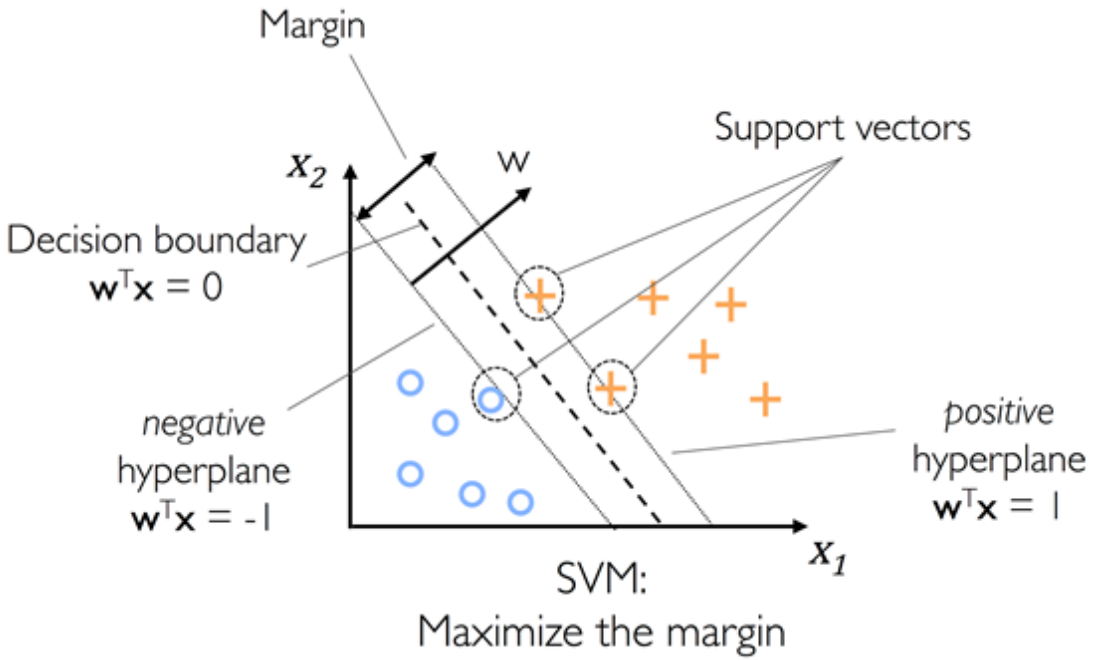
SVM분류

커널트릭

개요

- 결정함수 : n차원의 초평면$(hyperplane)$
- 결정경계 : 결정 함수의 값이 0인 점. $(n-1)$차원의 초평면$(hyperplane)$
- 마진$(Margin)$: 클래스를 구분하는 초평면(결정경계)과 이 초평면에 가장 가까운 훈련 샘플 사이의 거리. 이런 샘플을 **서포트 벡터**$(\boldsymbol{support\ vector})$라고 한다
- **SVM 최적화** : 마진$(Margin)$을 최대화 하는 것$(large\ margin\ classification)$





SVM 이론

- n차원 공간에 있는 입력 데이터 $x = [x_1 x_2 \cdots x_n]^T$
- $h(x) = w_1 x_1 + w_2 x_2 \cdots w_n x_n + b = w^T x + b = 0$
- 경계와 가장 가까운 support vector

$$w^T x_{pos} + b = 1 \quad \cdots \quad (1)$$
$$w^T x_{neg} + b = -1 \quad \cdots \quad (2)$$
$$w^T (x_{pos} - x_{neg}) = 2 \quad \cdots \quad (1) - (2)$$

$$\frac{w^T (x_{pos} - x_{neg})}{\|w\|} = \frac{2}{\|w\|}$$

$$\|w\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

$$r_{pos} = \frac{w^T x_{pos} + b}{\|w\|} = \frac{1}{\|w\|}$$

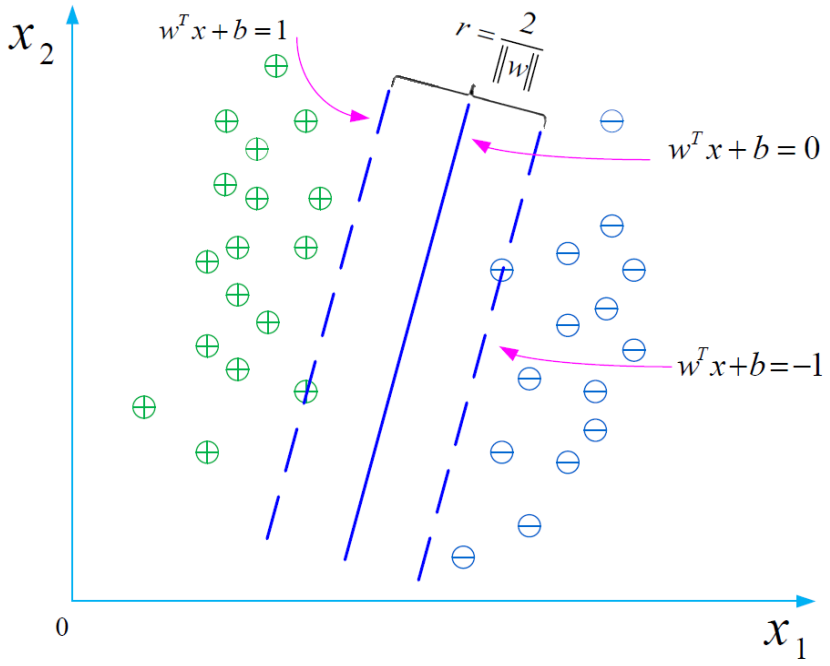
$$r_{neg} = \frac{w^T x_{neg} + b}{\|w\|} = -\frac{1}{\|w\|}$$

- Goal : maximize the margin

$$\therefore max \frac{2}{\|w\|}$$

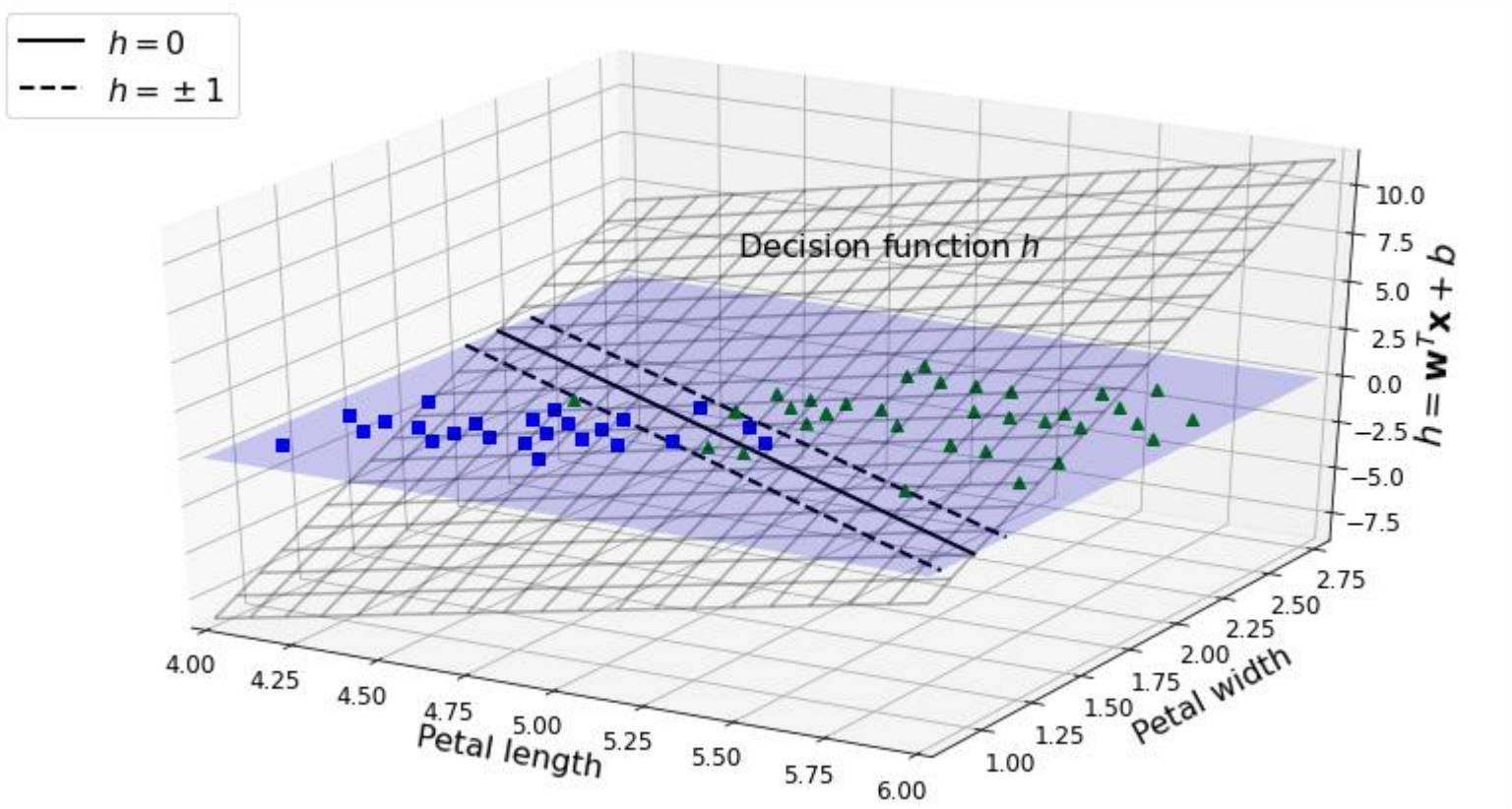
⇒ 콰드라틱$(quadratic)$ 프로그래밍 방법

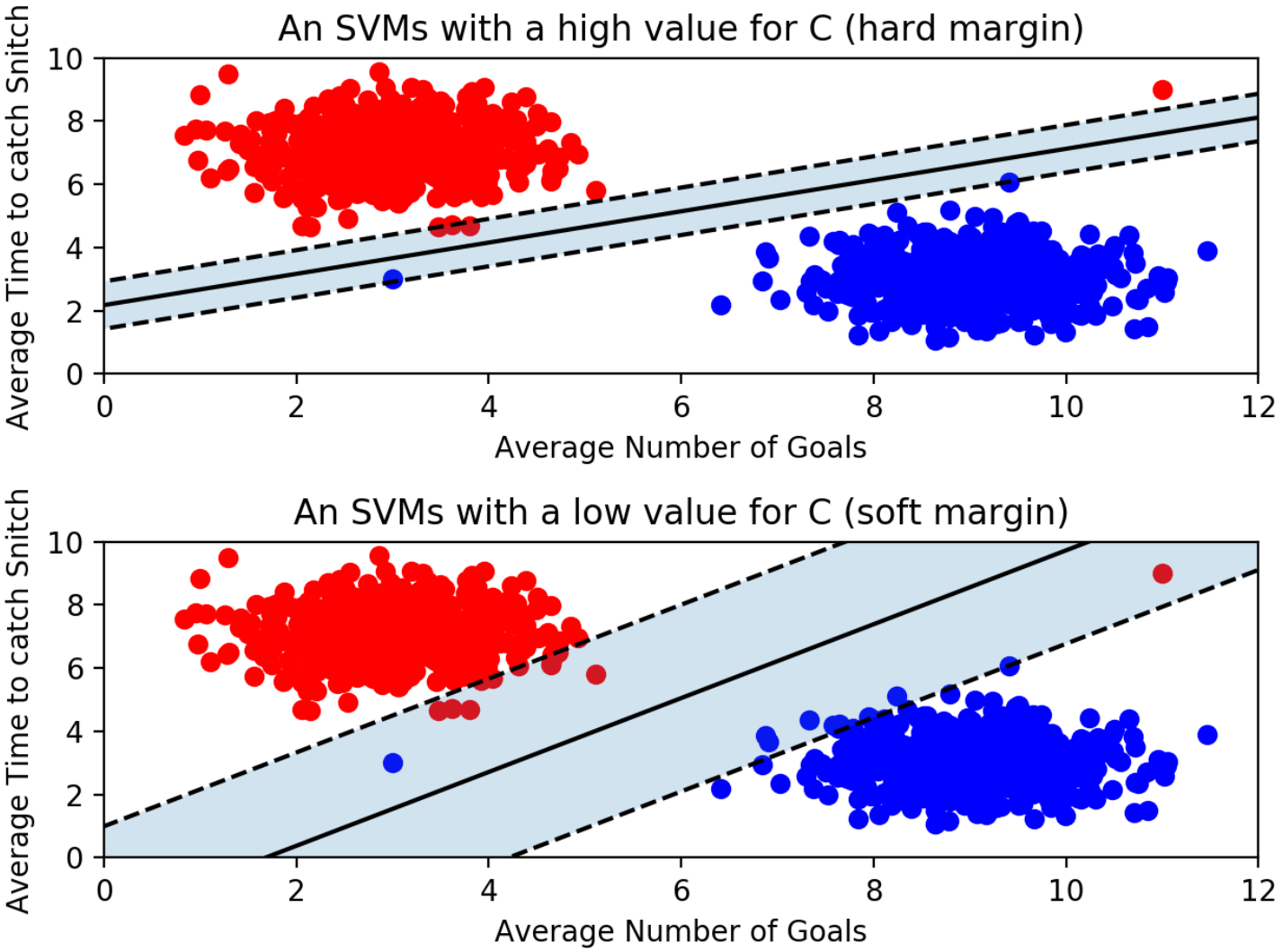
$min \frac{1}{2}\|w\|^2$ 하는 것이 더 쉽다



iris 데이터셋의 결정함수

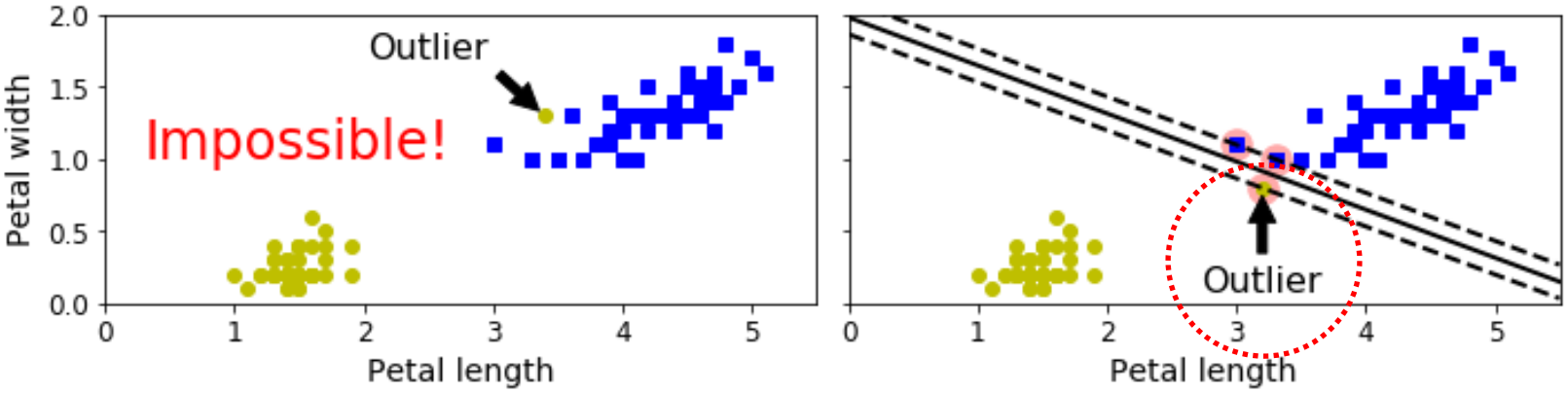
- SVM의 최적화는 가능한 한 마진을 크게 하는 $w^T x + b$의 w와 b를 찾는 것
- 점선 : 결정함수의 값이 1 또는 -1인 점



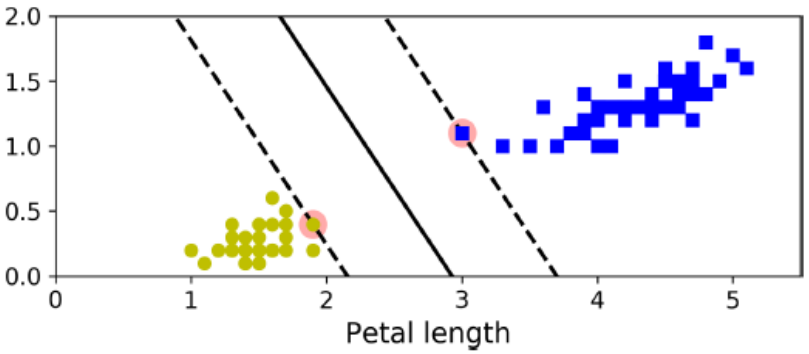

An SVMs with a high value for C (hard margin)
Average Time to catch Snitch
Average Number of Goals
0
2
4
6
8
10
12
An SVMs with a low value for C (soft margin)
Average Time to catch Snitch
Average Number of Goals

하드 마진 분류(hard margin classification)

- 모든 샘플이 경계선 바깥쪽에 올바르게 분류되어 있는 상태
- 데이터가 선형적으로 구분될 수 있어야 제대로 작동하며 이상치(outlier)에 **민감**

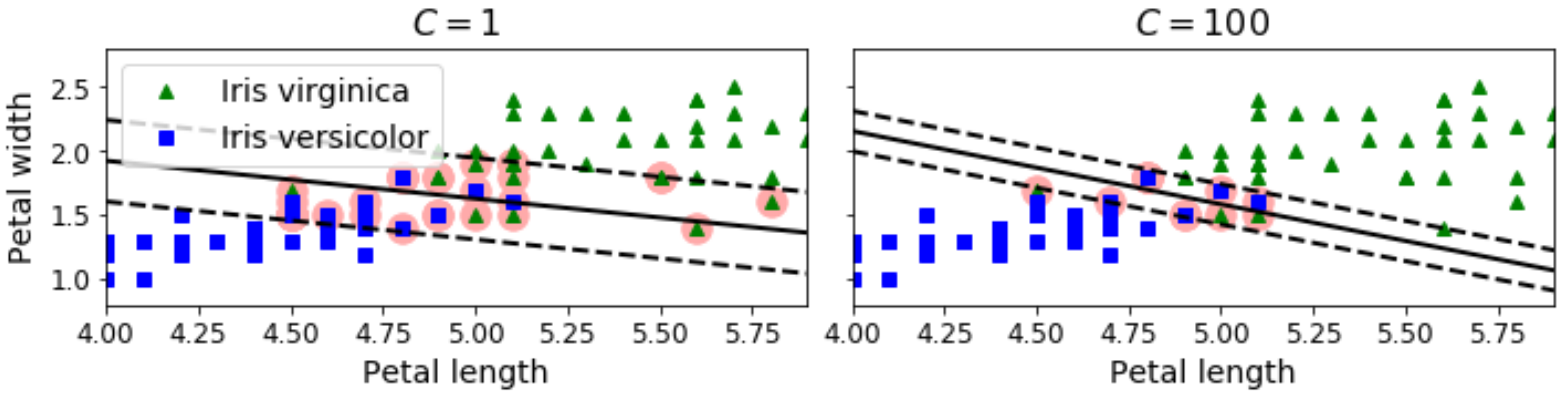


  - 왼쪽그래프 : 하드마진을 찾을 수 없다.
  - 오른쪽 그래프 : 결정경계는 이상치가 없는 경우(하단 그림)와 매우 다르다. 또한 일반화가 잘 될 것 같지 않다



소프트 마진 분류(soft margin classification)

- 마진오류(margin violation): 샘플이 경계선 중간이나 반대쪽에 있는 경우
- 경계의 폭을 가능한 넓게 유지하는 것
- 마진 오류 사이에 적절한 균형을 잡는 것



- 사이킷런(sklearn)의 SVM모델에서 여러 하이퍼 파라미터 지정(C는 여러 하이퍼 파라미터 중 하나)
  - 왼쪽 그래프 : 낮게 설정
  - 오른쪽 그래프 : 높게 설정
- 마진 오류는 일반적으로 적은 것이 좋다.
  - 그렇지만, 위의 경우에 왼쪽 모델의 마진 오류가 많지만 일반화가 더 잘 될 것 같다

목적함수

- 결정함수의 기울기 : 가중치 벡터 노름 $\|w\|$. 가중치 벡터 w가 작을 수록 마진은 커진다
- 마진을 크게 하기 위해 $\|w\|$를 최소화 : $w^T w = \|w\|^2$
- 음성 샘플일 때 $t^{(i)} = -1$, 양성 샘플일 때 $t^{(i)} = 1$로 정의하면 $\boldsymbol{t^{(i)}\left(w^T x^{(i)} + b\right) \geq 1}$ 로 표현할 수 있다.

- 하드마진 선형 SVM 분류기의 목적함수

$$\underset{W,b}{\boldsymbol{minimize}} \quad \frac{1}{2}\|w\|^2$$

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

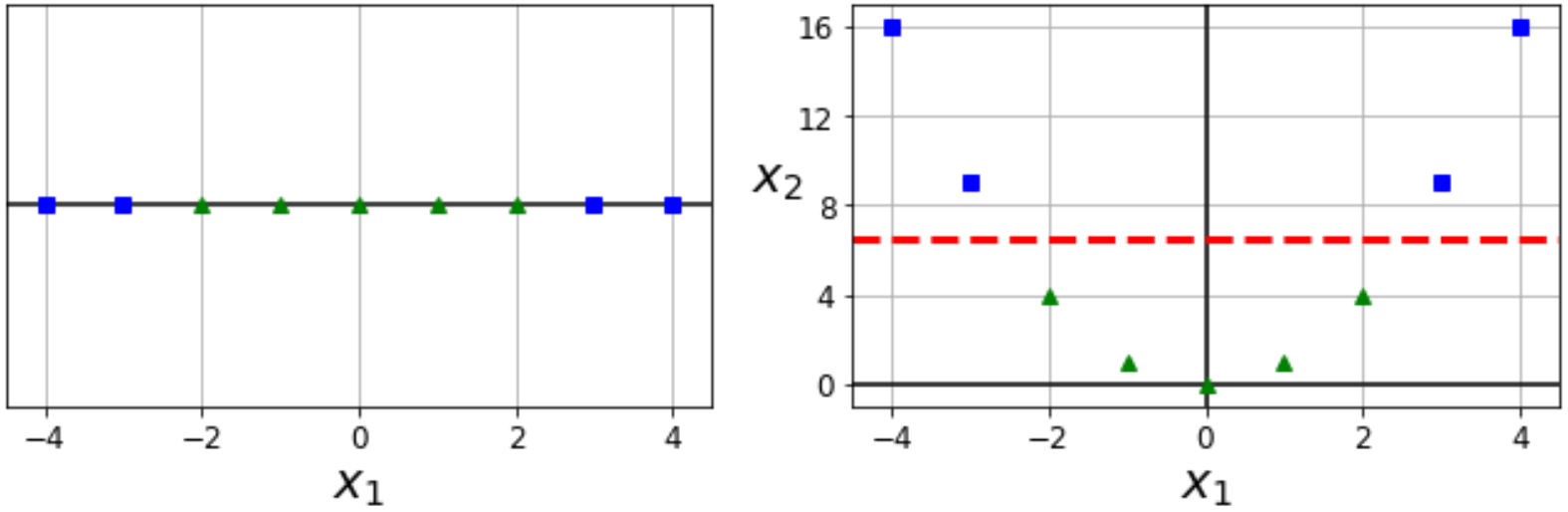
- 소프트마진 선형 SVM 분류기의 목적함수
  - 슬랙변수(slack variable) $\zeta^{(i)} \geq 0$을 도입. $\zeta^{(i)}$는 i번째 샘플이 얼마나 마진을 위반할지 결정
  - 하이퍼파라미터 C는 아래 두 목표 사이의 트레이드 오프를 정의
    - 슬랙변수의 값이 작으면 마진의 오류를 최소화
    - $\frac{1}{2}\|w\|^2$가 작으면 마진을 크게 한다

$$\underset{W,b,\zeta}{\boldsymbol{minimize}} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\zeta^{(i)}$$

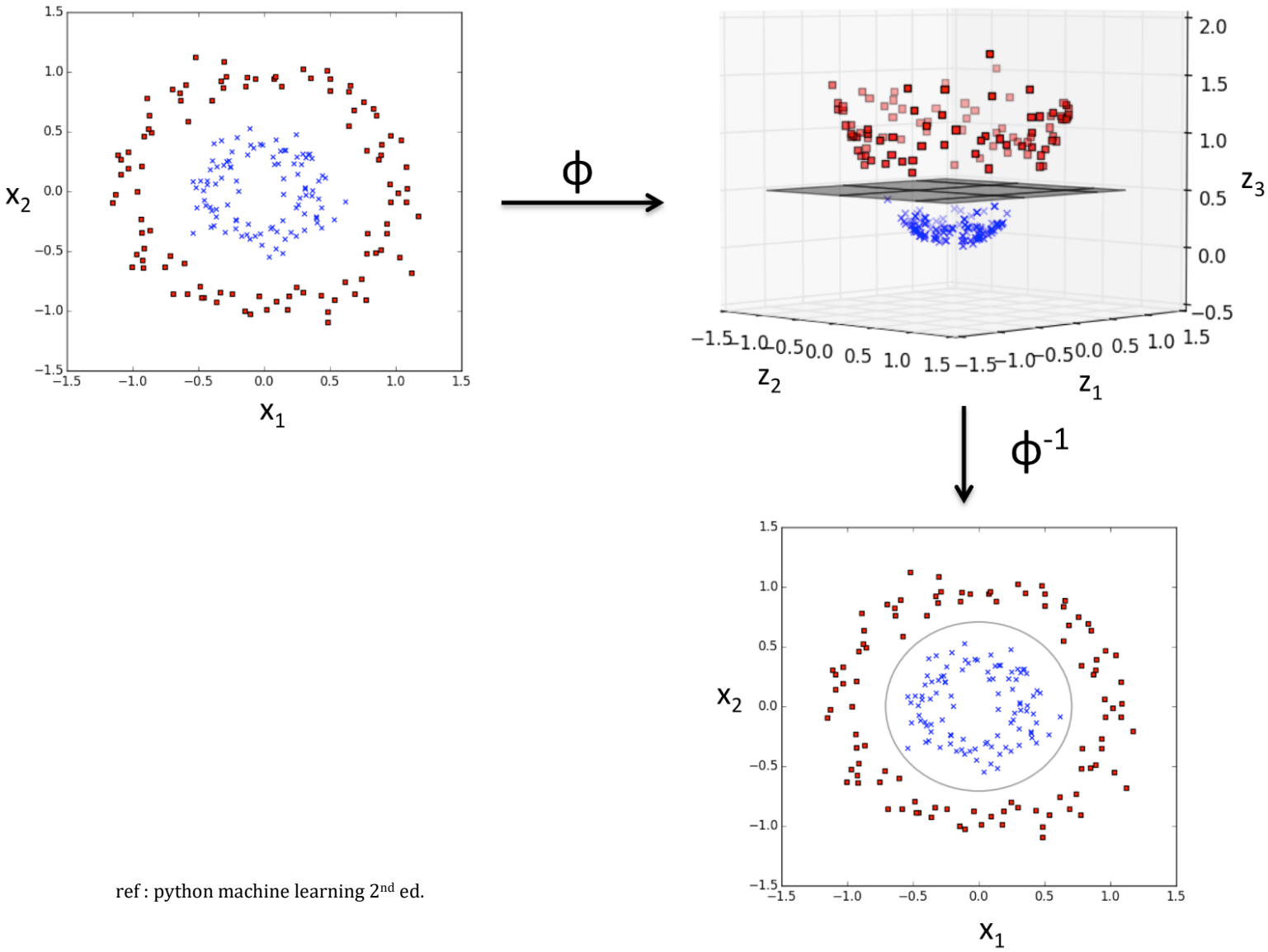
[조건] $i = 1,2,\cdots,m$일 때 $\quad t^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \zeta^{(i)}$이고 $\zeta^{(i)} \geq 0$

비선형 SVM분류

- $x_2 = (x_1)^2$로 하여 만들어진 2차원 데이터셋은 선형적으로 구분 가능(오른쪽 그래프)



고차원 공간에서 찾은 결정 경계의 예



ref : python machine learning 2nd ed.

PolynomialFeatures(degree=d)

- 다항식 특성
  - 낮은 차수의 다항식 : 매우 복잡한 데이터셋을 잘 표현하지 못함
  - 높은 차수의 다항식 : 매우 많은 특성을 추가하므로 모델이 느림
- 특성이 n개인 배열을 특성이 $\frac{(n+d)!}{d!n!}$ 개인 배열로 변환
  - $n!$ 로서 특성수가 교차항을 포함해 엄청나게 늘어날 수 있다.
- (예) 두 개의 특성 a, b가 있을 때 degree=3으로 적용하면
  - $a^2, a^3, b^2b^3, ab, a^2b, ab^2$

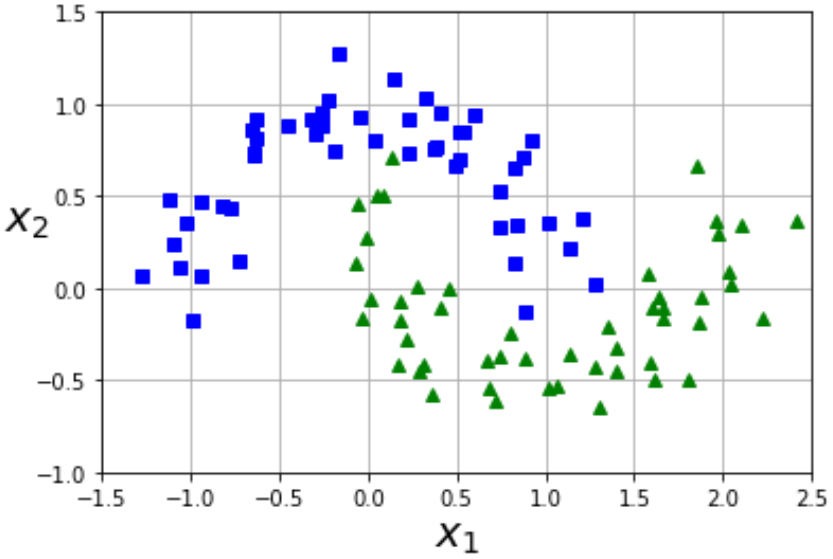
```
from sklearn.datasets import make_moons
from sklearn.preprocessing import PolynomialFeatures
X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

poly =PolynomialFeatures(degree=3)
poly.fit(X, y)
# 만들어진 특성의 차수 확인
poly.get_feature_names()
```

['1', 'x0', 'x1', 'x0^2', 'x0 x1', 'x1^2', 'x0^3', 'x0^2 x1', 'x0 x1^2', 'x1^3']

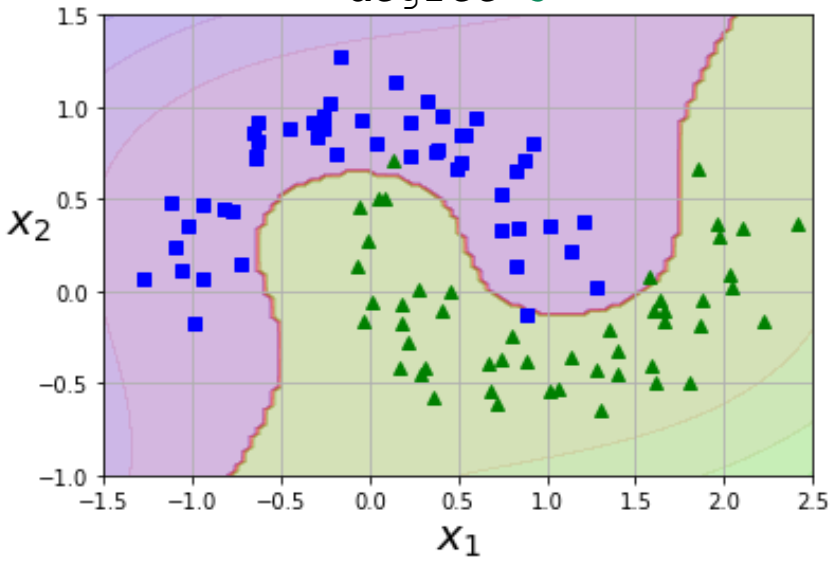
```
# interaction_only=True지정 : 거듭제곱이 포함된 항은 제외
poly =PolynomialFeatures(degree=3, interaction_only=True)
poly.fit(X, y)
poly.get_feature_names()
```

['1', 'x0', 'x1', 'x0 x1']

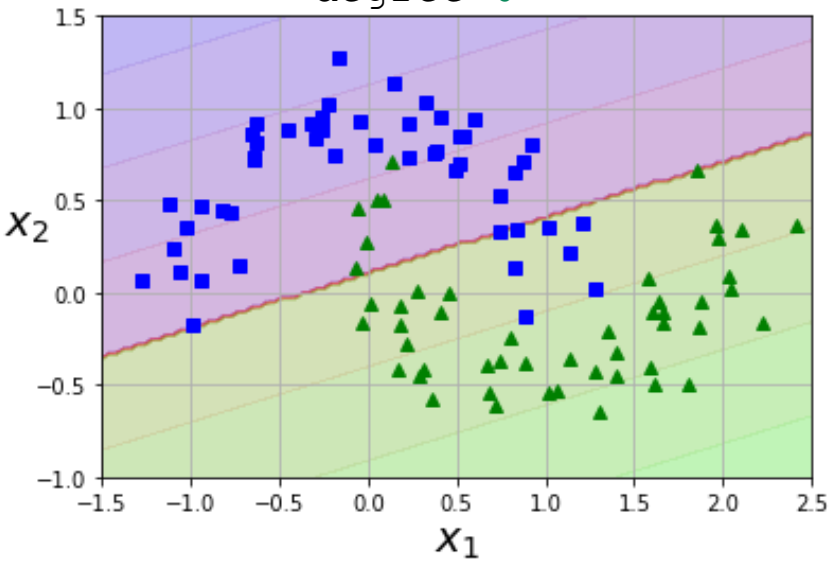
다항 특성을 사용한 선형 SVM분류



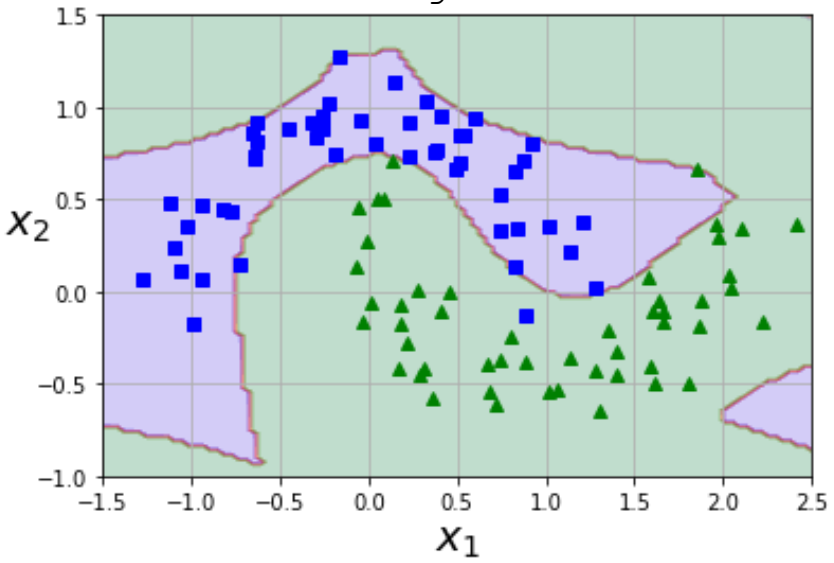
degree=3



degree=0



degree=300



SVM분류

커널트릭

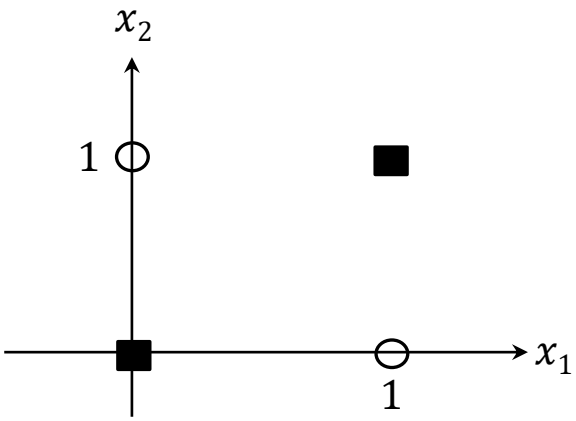
커널트릭(kernel trick)

- 고차원 공간으로 매핑
  - SVM으로 비선형을 해결하기 위하여 **매핑함수 ϕ**를 사용하여 훈련 데이터를 고차원 특성 공간으로 변환
  - 새로운 특성을 만드는 계산 비용이 매우 비싸다(특히 고차원 데이터일 때)
  - ϕ를 고차원으로 변환하여 계산하지 않고 원래 데이터에서 계산. 따라서 높은 비용을 절감하기 위해 커널함수를 정의
- 커널(kernel)
  - 샘플간의 유사도 함수(similarity function)
  - 0과 1사이의 범위
- 커널 트릭(kernel trick)
  - 원래 공간에서 더 높은 차원의 새로운 공간으로 매핑
  - 실제로 특성을 추가하지 않으면서 다항식 특성을 많이 추가한 것과 같은 결과를 얻을 수 있다

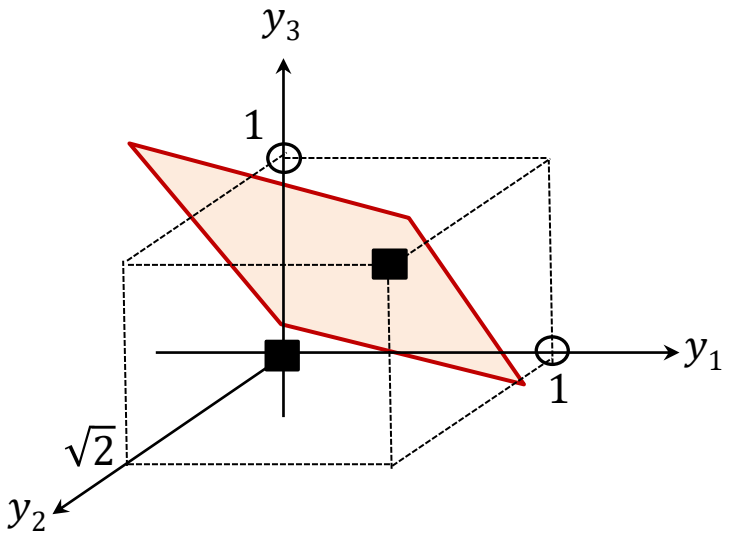
커널트릭(kernel trick) - 예

XOR의 2차원 입력(x_1, x_2)를 **매핑함수 $\boldsymbol{\phi(x_1, x_2)}$** 를 이용하여 3차원으로 변환

$$\phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right) = (y_1, y_2, y_3)$$



(a)원래공간 x



(b)변환 공간 $\phi(x)$

커널함수(kernel function)

- 어떤 특징공간에 정의된 두 특징벡터$x^{(i)}, x^{(i)}$에 대해$K\left(x^{(i)}, x^{(j)}\right) = \phi\left(x^{(i)}\right) \cdot \phi\left(x^{(j)}\right)$인 변환함수 ϕ 가 존재하면 $K\left(x^{(i)}, x^{(j)}\right)$를 커널함수라 부른다
- **선형 커널**

  $K\left(x^{(i)}, x^{(j)}\right) = (x^{(i)})^T \cdot x^{(j)}$
- **다항식 커널(polynomial kernel)**

  $K\left(x^{(i)}, x^{(j)}\right) = \left(\Upsilon (x^{(i)})^T \cdot x^{(j)} + r\right)^d, \qquad d: \text{integer}(+)$
- **방사기저 함수(radial basis function, RBF), 가우시안 커널(Gaussian kernel)**

  $$K\left(x^{(i)}, x^{(j)}\right) = exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}\right)$$

  $$K\left(x^{(i)}, x^{(j)}\right) = exp\left(-\Upsilon\left\|x^{(i)} - x^{(j)}\right\|^2\right), \qquad \gamma = \frac{1}{2\sigma^2}$$
- **쌍곡 탄젠트 커널(hyperbolic tangent), 시그모이드(sigmoid)**

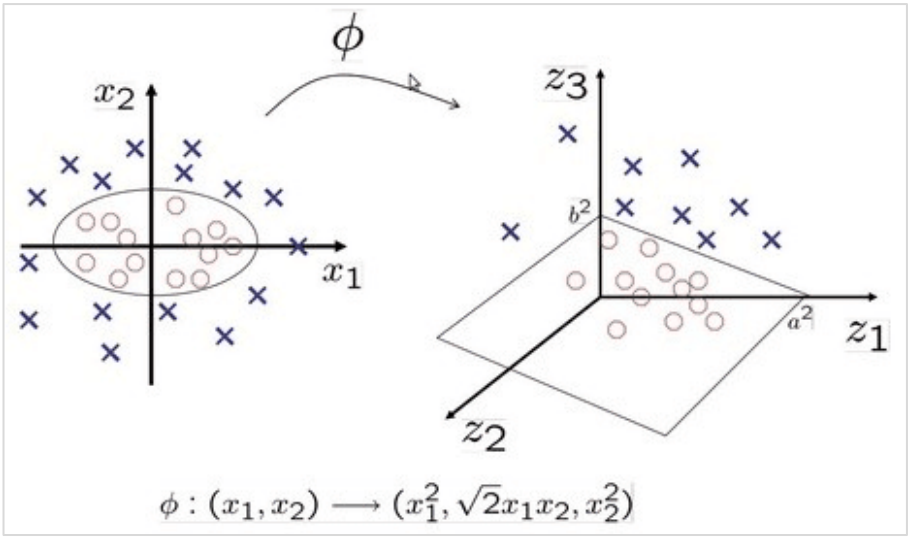
  $K\left(x^{(i)}, x^{(j)}\right) = tanh\left(\Upsilon (x^{(i)})^T \cdot x^{(j)} + r\right)$

- 사이킷런의 SVC, SVR에서 매개변수 kernel에 지정할 수 있는 함수
  - “linear”, “poly”, “rbf”, “sigmoid”

2차원 다항식 커널

- 2차 다항식 **매핑 함수 ϕ**

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



- 2차 다항식 매핑을 위한 커널 트릭

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

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

$$\therefore\ \boldsymbol{\phi(a)^T\phi(b) = \left(a^Tb\right)^2}$$

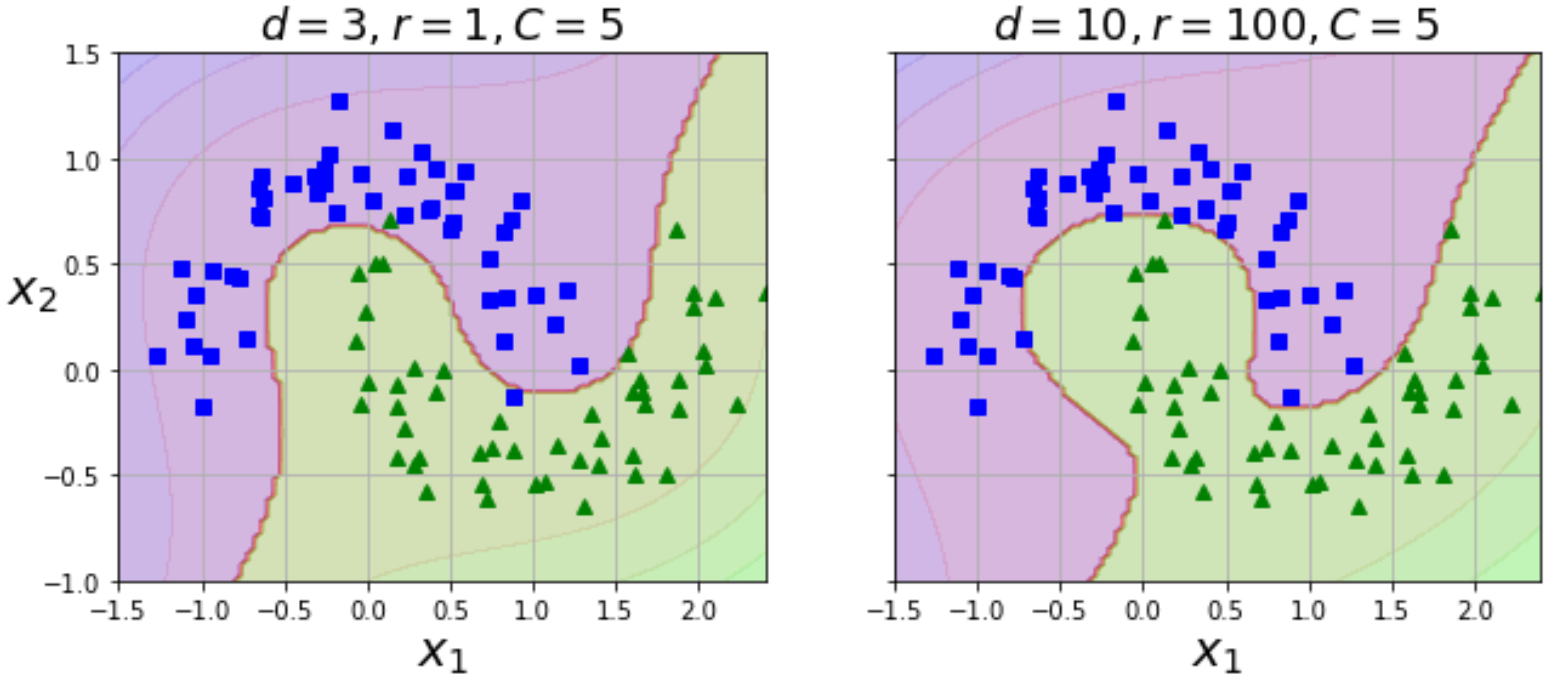
비선형 SVM분류 - 다항식 커널(1)


```
# 3차 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))
    ])
poly_kernel_svm_clf.fit(X, y)
```

```
# 10차 다항식 커널
poly100_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=10, coef0=100, C=5))
    ])
poly100_kernel_svm_clf.fit(X, y)
```

비선형 SVM분류 - 다항식 커널(2)

- 과대적합(overfitting) : 다항식의 차수를 줄인다
- 과소적합(underfitting) : 다항식의 차수를 늘린다
- coef0 : 모델이 높은 차수와 낮은 차수에 얼마나 영향을 받을 지 조절
  - 차수가 높아질수록 1보다 작은 값과 1보다 큰 값의 차이가 크게 벌어지므로 coef0을 적절한 값으로 지정하면 고차항의 영향을 줄일 수 있다.
  - 다항식 커널에 있는 상수항 r (기본값은 0)

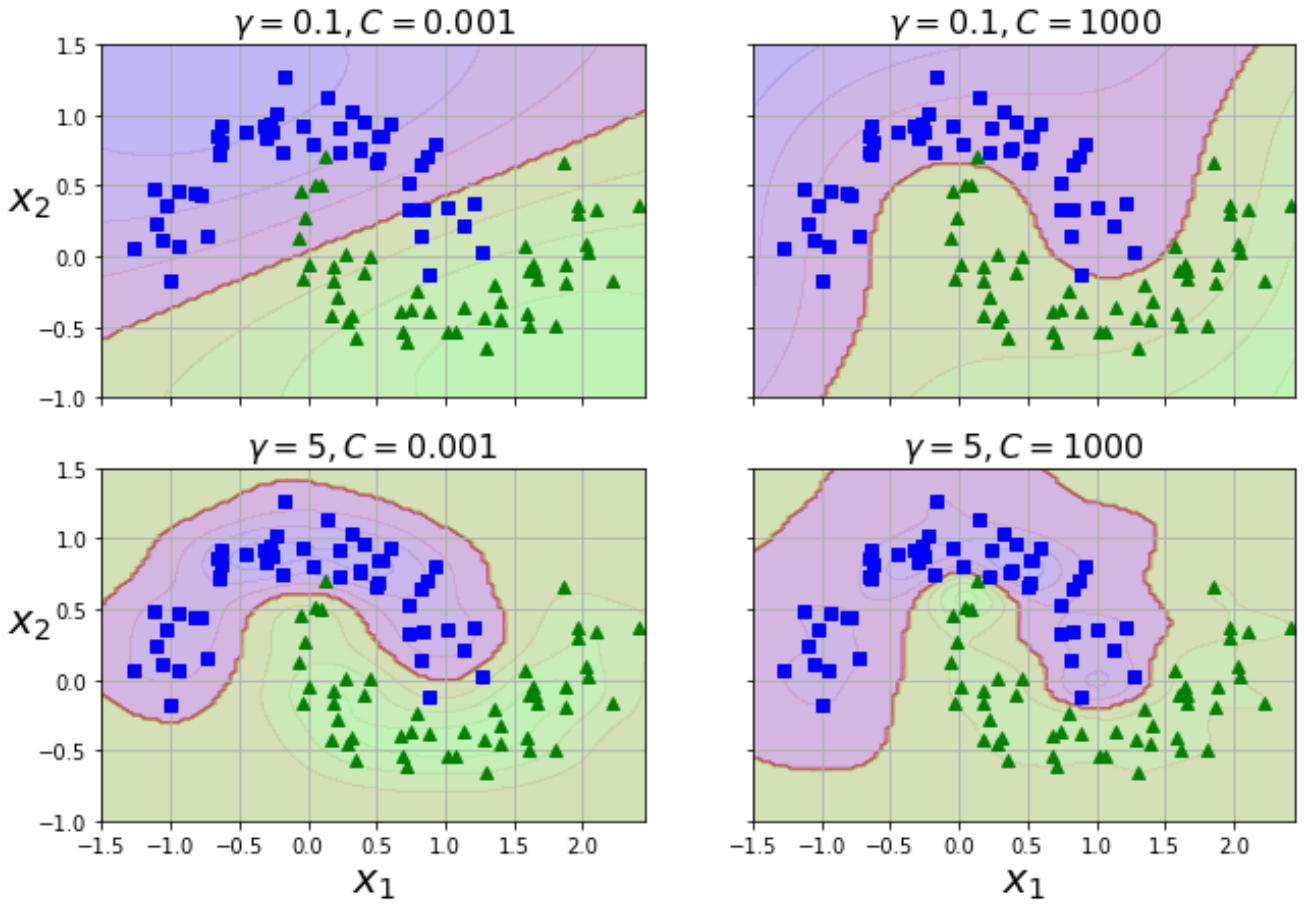


비선형 SVM분류 - RBF(1)


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

```
from sklearn.svm import SVC

gamma1, gamma2 = 0.1, 5
C1, C2 = 0.001, 1000
hyperparams = (gamma1, C1), (gamma1, C2), (gamma2, C1), (gamma2, C2)

svm_clfs = []
for gamma, C in hyperparams:
    rbf_kernel_svm_clf = Pipeline([
            ("scaler", StandardScaler()),
            ("svm_clf", SVC(kernel="rbf", gamma=gamma, C=C))
        ])
    rbf_kernel_svm_clf.fit(X, y)
    svm_clfs.append(rbf_kernel_svm_clf)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10.5, 7), sharex=T
rue, sharey=True)
```

비선형 SVM분류 - RBF(2)



- gammer가 규제역할 : 과대적합일 경우 감소, 과소적합일 경우 증가
- γ 를 증가시키면 결정경계가 불규칙해지고 각 샘플을 따라 구불구불하게 휘어진다
- 작은 γ 값은 샘플이 넓은 범위에 걸쳐 영향을 주므로 결정경계가 더 부드러워진다
- 모델의 복잡도를 조절하려면 γ 와 C 를 함께 조정하는 것이 좋다