SVM (Support Vector Machine)

2025. 8

Yongjin Jeong, KwangWoon University

[1] Aurellian Geron, Hands-on Machine Learning with Scikit, Keras, and Tensorflow
[2] https://www.robots.ox.ac.uk/~az/lectures (Prof. Zisserman's lecture slide)
[3] ChatGPT, Gemini

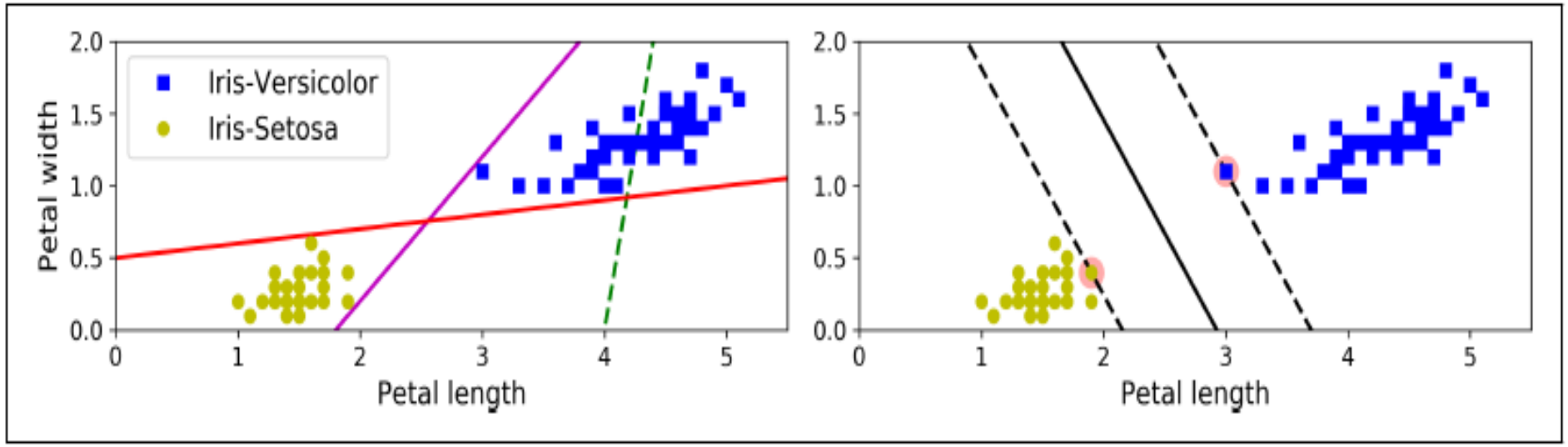
Contents

SVM(Support Vector Machine) 개요

- SVM은 데이터를 분리하는 최적의 결정 경계(Decision Boundary)를 찾는 데 특화됨.
- 최대 마진(Maximum Margin)의 개념
  - SVM의 핵심 아이디어는 최대 마진(Maximum Margin)이다. 마진이란 결정 경계(초평면)와 그 경계에 가장 가까이 있는 데이터 포인트들(서포트 벡터) 사이의 거리를 의미한다. SVM은 이 마진의 거리를 **최대화**하는 초평면을 찾는다.
- 선형 모델 vs. SVM 모델
  - **선형 모델:** 데이터셋을 분류할 수 있는 여러 개의 선을 찾을 수 있지만, 그중 어떤 선이 가장 좋은 분류기인지 알기 어렵다.
  - **SVM 모델:** 마진을 최대화함으로써 가장 견고하고 일반화 성능이 뛰어난 결정 경계를 찾아준다.

Key Idea: Maximum Margin (최대 마진)

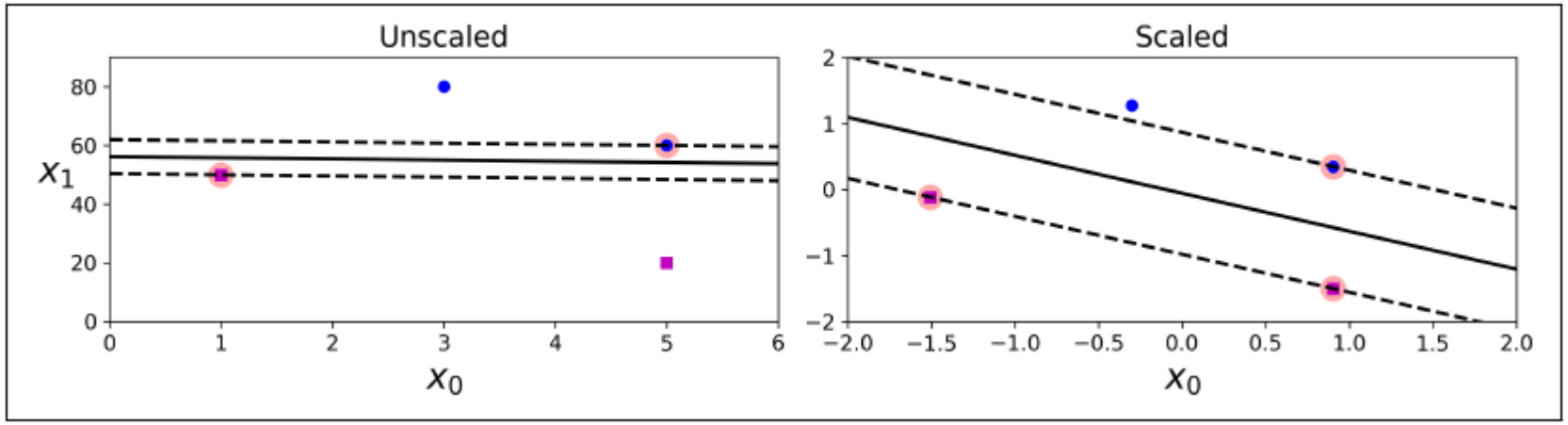
- **Linear Classifier and SVM Classifier**



- 선형모델: 붓꽃(Iris) 데이터셋을 분류하는 여러 개의 선을 보여주며, 이 선들은 모두 데이터를 잘 분류하지만, 어떤 선이 가장 좋은 분류기인지는 알기 어렵다.
- SVM 모델: 데이터를 분류하는 선(hyper-plane, 초평면)과 함께, 그 초평면으로부터 가장 가까운 데이터들(서포트 벡터)과의 거리를 최대화하는 '마진(margin)'을 보여준다. 이 마진을 최대화하는 선을 찾는 것이 SVM의 핵심 목표이다.

Maximum Margin

- **Sensitivity to feature scales**



before scaling

after scaling

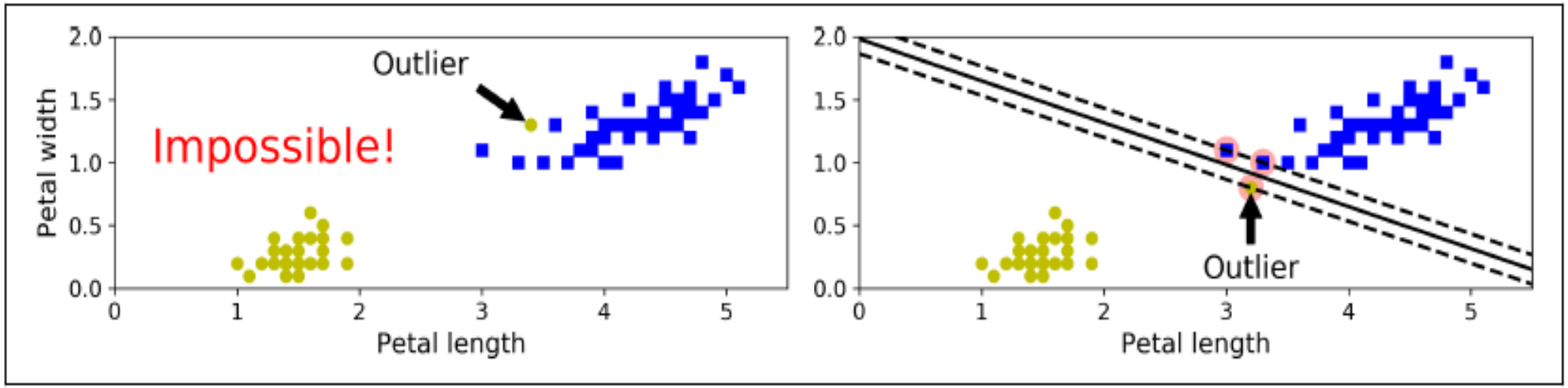
- Before Scaling: SVM은 각 데이터 포인트 간의 거리를 기준으로 최적의 결정 경계를 찾기 때문에, 스케일이 크게 다른 경우 스케일이 큰 축(x1) 의 영향력이 지나치게 커지게 된다. 결과적으로, 모델은 최적의 마진을 찾지 못하고 결정 경계가 한쪽으로 치우치게 된다.
- After Scaling: 모든 특성이 모델 학습에 동등하게 기여할 수 있고, SVM은 최적의 마진을 가진 초평면(결정 경계)을 찾을 수 있다. 이는 모델의 성능을 향상시키는 데 매우 중요한 전처리 과정이다.

Hard Margin and Soft Margin

- **하드 마진(Hard Margin)**
  - 모든 데이터가 완벽하게 분리될 수 있을 때 사용한다
  - 결정 경계가 마진 안에 어떤 데이터 포인트도 허용하지 않는다
  - 장점: 분류 오차가 전혀 없는 모델을 만들 수 있다
  - 단점: 현실의 많은 데이터는 완벽하게 분리되지 않으므로, 이상치(outlier)에 매우 민감하고 과적합(overfitting)될 가능성이 높다
- **소프트 마진(Soft Margin):**
  - 현실의 이상치나 완벽하게 분리되지 않는 데이터를 처리하기 위해 사용한다
  - 일부 데이터 포인트가 마진 안이나 심지어 결정 경계를 넘어가는 것을 허용한다
  - 장점: 이상치에 덜 민감하고 일반화 성능이 좋다.
  - 단점: 약간의 분류 오차를 허용한다.

Hard Margin

- **Hard Margin**
  - 모든 데이터 포인트를 완벽하게 분류하는 초평면(hyperplane)을 찾는다. 즉, 마진 경계선 안쪽이나 반대편에 어떤 데이터 포인트도 존재하지 않아야 한다.
  - 하지만 만약 이상치(outlier)가 존재하거나 데이터가 선형적으로 분리되지 않는 경우, 하드 마진 방식은 분류에 실패하거나 적절한 마진을 찾지 못하게 된다.



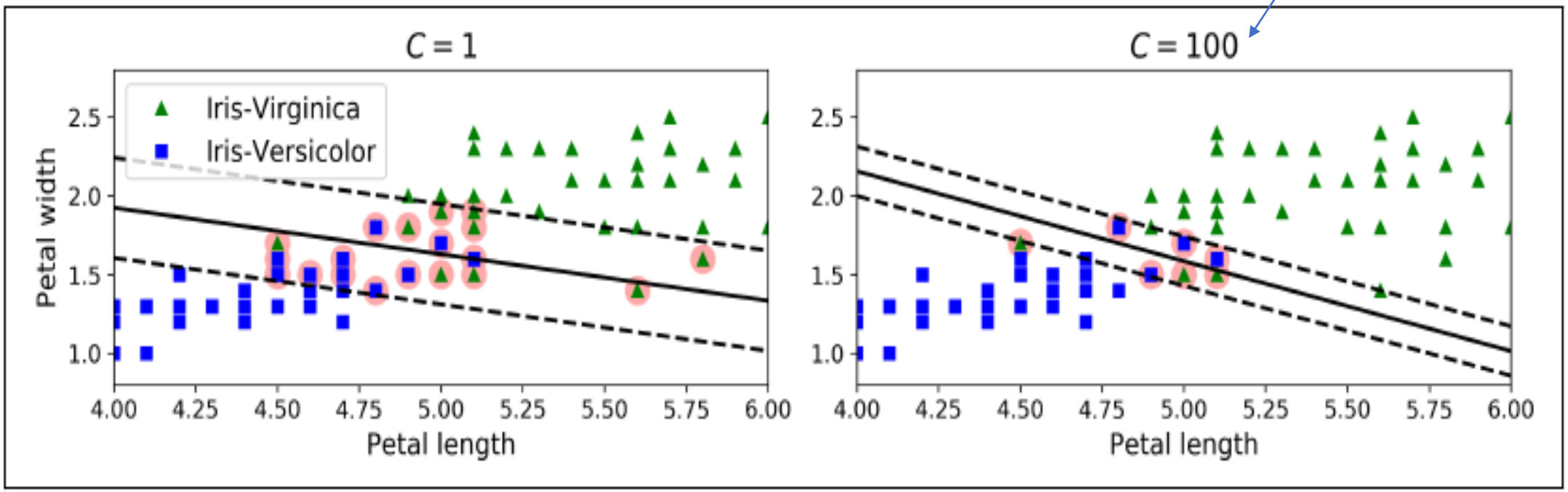
Soft Margin

- **Soft Margin**
  - 일부 이상치를 허용함(오분류를 허용함)으로써 더 합리적인 초평면을 찾는다.
  - 모델은 마진을 최대한 넓게 유지하면서, 동시에 마진을 침범하는 데이터에 대해 페널티를 부과하는 방식으로 학습한다. 이때 페널티의 강도를 조절하는 하이퍼파라미터가 C 이다.
- **하이퍼파라미 C**
  - C 값이 작을 때: 오분류에 대한 페널티가 작아지며 모델은 일부 오분류를 허용하더라도 마진을 최대한 넓게 설정한다. 지나치게 마진이 넓어지면 과소적합(Underfitting)이 발생할 수 있다.
  - C 값이 클 때: 오분류에 대한 페널티가 커진다. 모델은 마진 경계를 엄격하게 지키려고 하며, 마진을 침범하는 데이터를 최소화한다. 이 경우 결정 경계는 더 복잡해질 수 있으며, 훈련 데이터에 너무 맞춰져서 과적합될 가능성이 높아진다.

Soft Margin

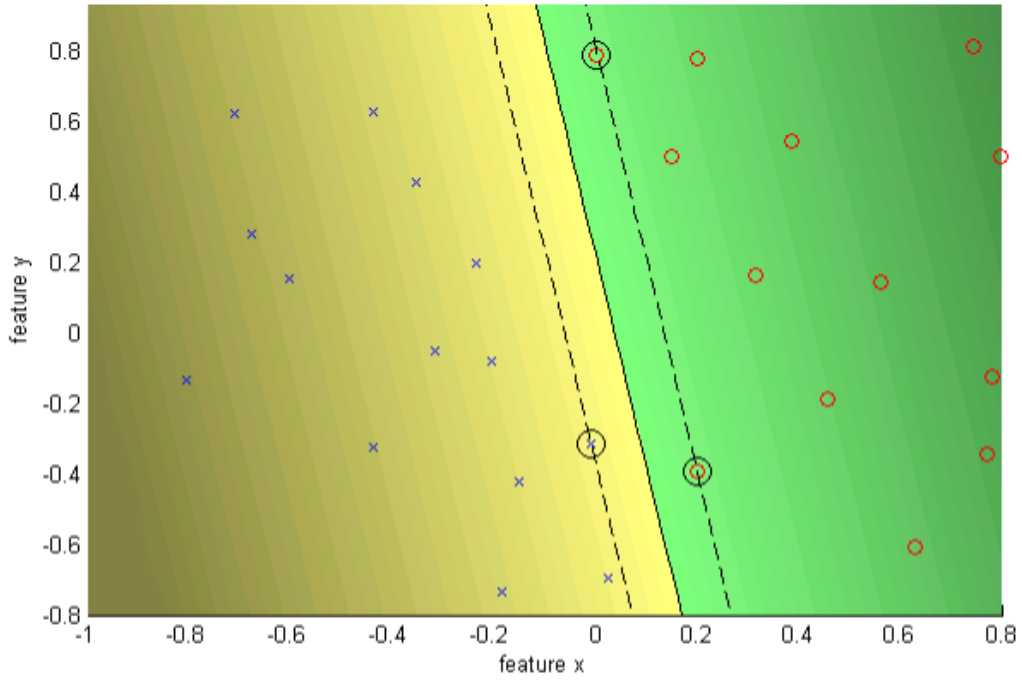
- **Soft Margin**

Overfitting 가능성

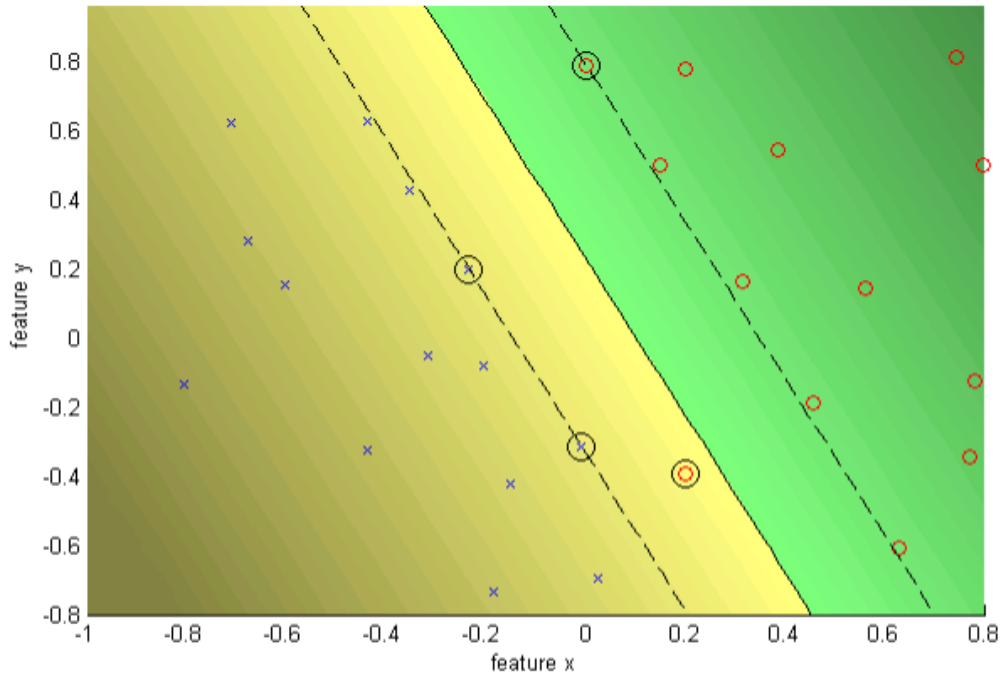


Hard Margin and Soft Margin with C

- **Hard margin and Soft margin**



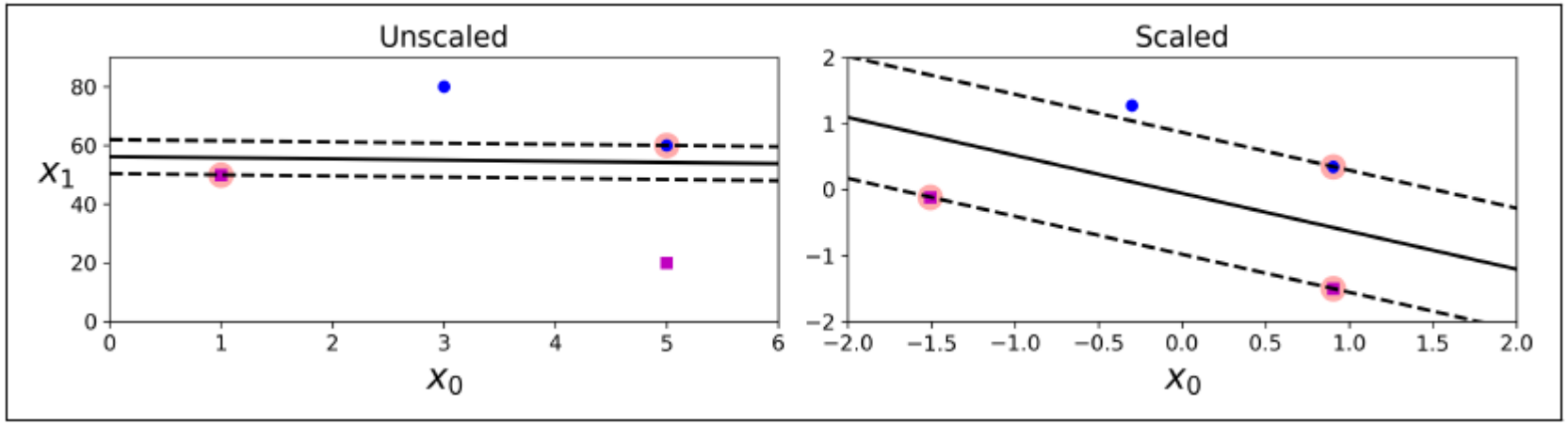
C = Infinity, hard margin



C = 10, Soft margin

SVM Classifier

- **Sensitivity to feature scales**



Before Scaling **After Scaling**

- Before Scaling: SVM은 각데이터 포인트 간의 거리를 기준으로 최적의 결정 경계를 찾기 때문에, 스케일이 크게 다른 경우 스케일이 큰 축(x1)의 영향력이 지나치게 커지게 된다. 결과적으로, 모델은 최적의 마진을 찾지 못하고 결정 경계가 한쪽으로 치우치게 된다.
- ·After Scaling: 모든 특성이 모델 학습에 동등하게 기여할 수 있고, SVM은 최적의 마진을 가진 초평면(결정경계)을 찾을 수 있다. 이는 모델의 성능을 향상시키는 데 매우 중요한 전처리 과정이다.

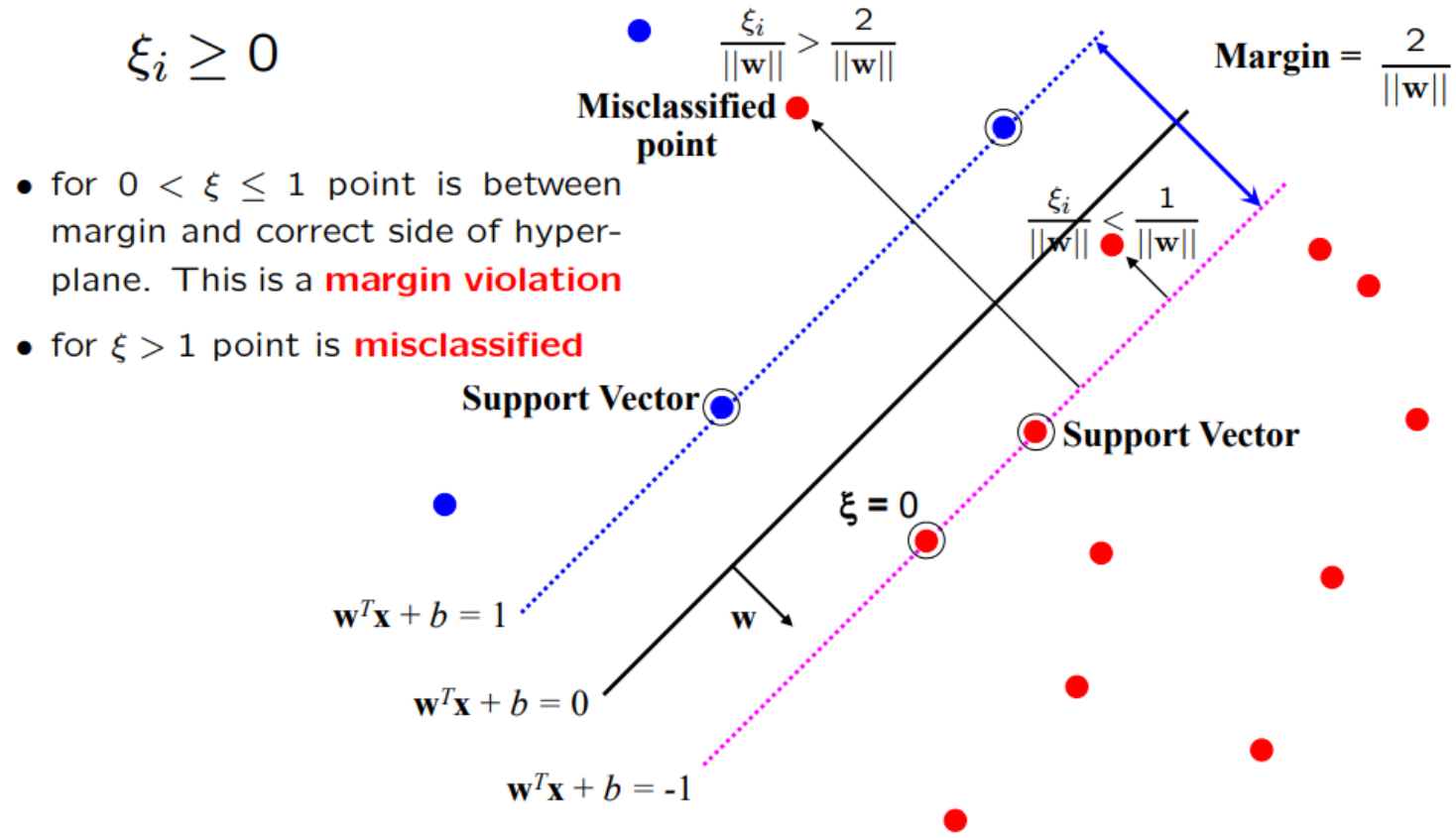
Slack Variable (슬랙 변수)

- **Introduce Slack Variables (for Soft Margin)**
  - 슬랙 변수 (ξ) 는 개별 데이터 포인트가 마진 경계를 얼마나 침범했는지를 측정하는 변수
  - ξ=0: 데이터 포인트가 마진 경계선 밖, 즉 올바른 위치에 있다.
  - 0<ξ≤1: 데이터 포인트가 마진 경계선 안쪽에 있지만, 올바른 클래스에 속해 있다. (Margin Violation)
  - ξ>1: 데이터 포인트가 결정 경계를 넘어 반대편에 잘못 분류된 경우이다. (Misclassification)
  - Soft Margin SVM은 슬랙 변수를 허용함으로써, 모델이 모든 데이터를 완벽하게 분류하지 않아도 되도록 만든다. 이 슬랙 변수들의 총합에 페널티를 부과하여, 마진을 최대한 넓게 유지하면서도 오분류의 양을 적절히 조절한다.

- **Relation with Hyperparameter C**
  - 슬랙 변수에 대한 페널티의 강도는 하이퍼파라미터 C가 조절한다.
  - C 값이 크면, 슬랙 변수에 대한 페널티가 커지므로, 모델은 마진 위반이나 오분류를 최소화하려고 한다
  - C 값이 작으면, 페널티가 작아져 마진 위반을 더 많이 허용하며, 더 넓은 마진을 찾게 된다.

Slack Variable

- **Slack Variables (for Soft Margin)**
  - Amount of error (hinge loss) from the correct side of Hyperplane (오분류를 허용하는 정도)



Hinge Loss

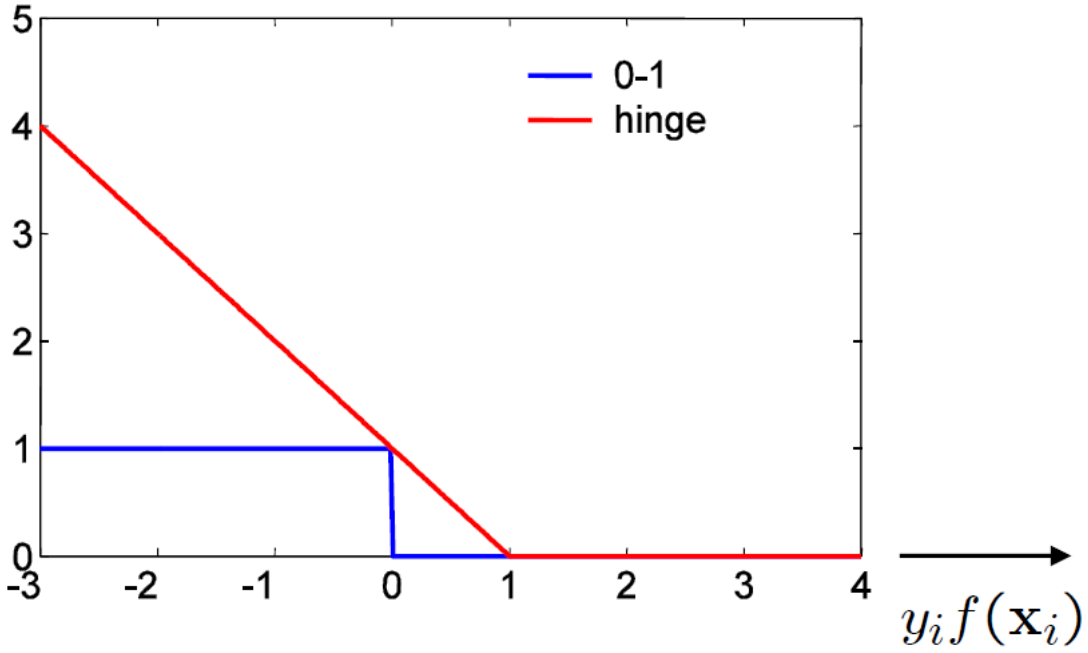
- 슬랙 변수의 합은 전체 오분류에 대한 페널티를 나타낸다. 힌지 손실은 이 페널티를 계산하는 데 사용되는 함수이다.

- 수식 $\max(0, 1 - y_i(w^T x_i + b))$

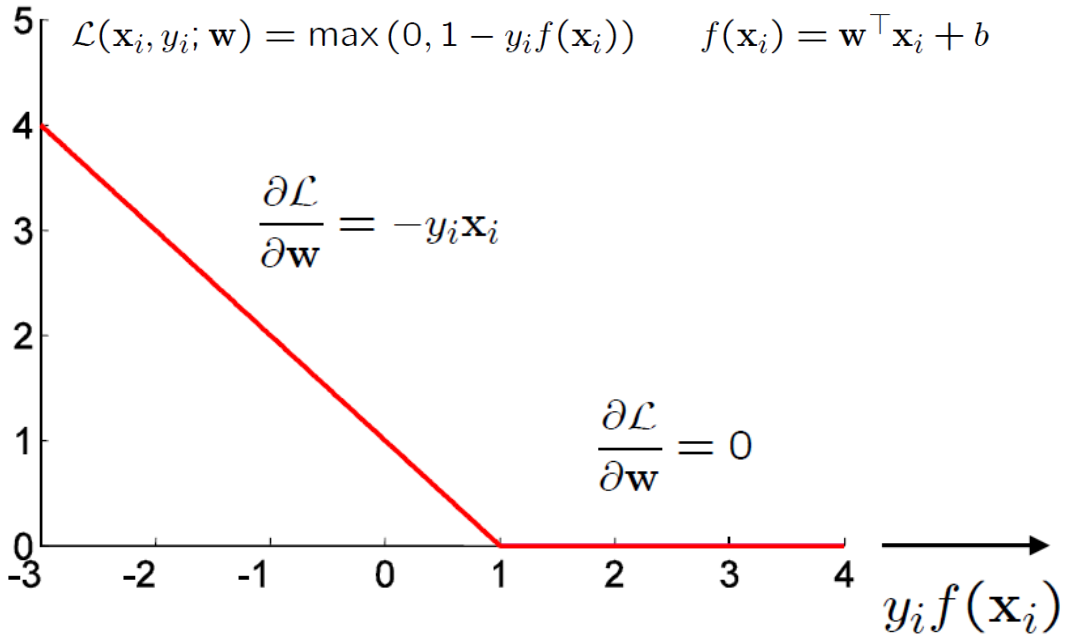
- 직관적 설명: 힌지 손실은 마진 경계(margin)를 벗어난 정도에 비례하여 페널티를 부과하는 함수'이다. 올바르게 분류된 데이터는 손실이 0이 되며, 마진 경계를 침범하거나 오분류된 데이터일수록 손실이 커진다.

Hinge Loss

- **Hinge loss**



- **Sub-gradient for Hinge loss**



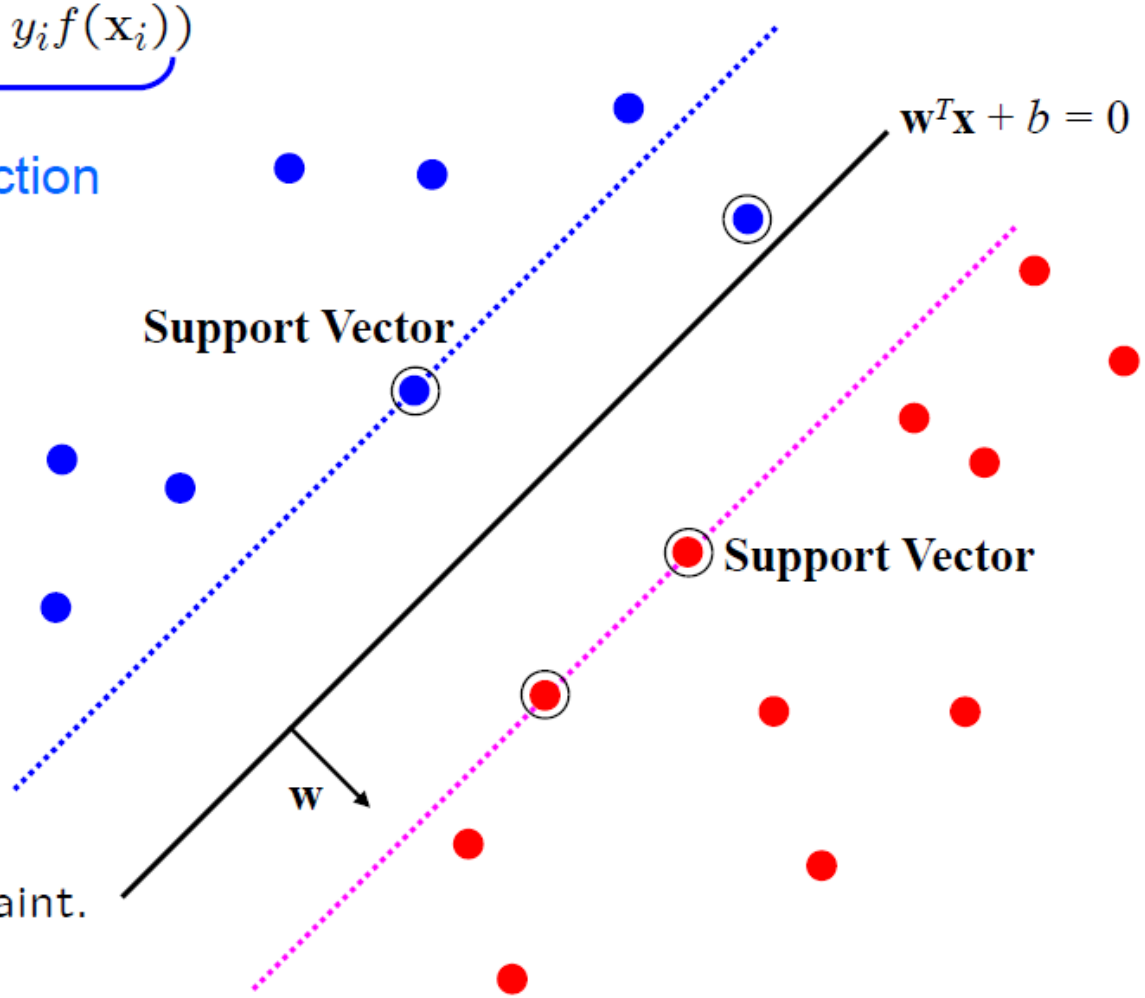
- SVM uses "hinge" loss $\max(0, 1 - y_i f(\mathbf{x}_i))$
- an approximation to the 0-1 loss

Hinge Loss

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss

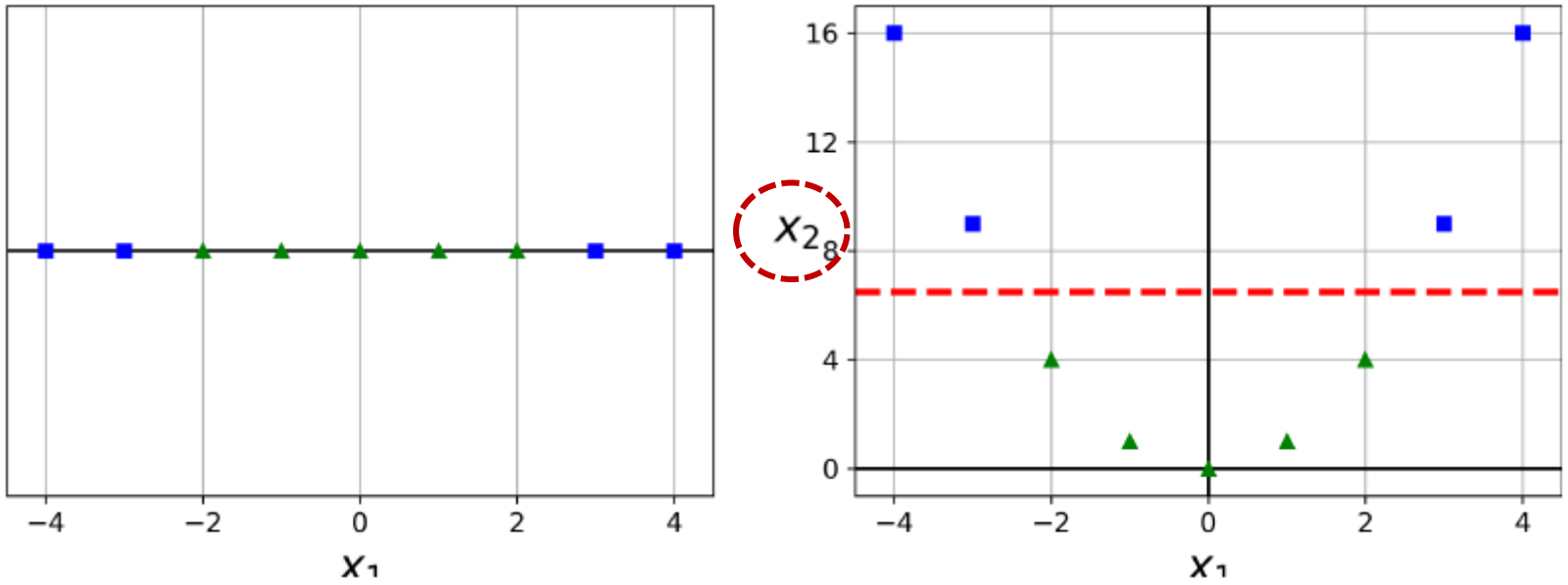


Nonlinear SVM

- **비선형 데이터 문제**
  - 많은 실제 데이터는 선형적으로 분리되지 않는다,
  - (예) 다음 슬라이드
- **고차원 매핑의 어려움**
  - 비선형 문제를 해결하기 위해 데이터를 고차원 공간으로 변환하여 선형적으로 분리할 수 있게 만드는 방법이 있다.
  - 예를 들어, x1이라는 1차원 특성을 $x2 = (x1)^2$라는 새로운 특성을 추가하여 2차원으로 변환하면, 다음 그림처럼 데이터가 포물선 형태로 나타나고 선형적으로 분리 가능해진다.
  - 하지만 데이터의 특성 수가 많아지면, 고차원으로의 변환은 계산량이 너무 많아져 비효율적이다. 특히 특성 수가 수백, 수천 개가 되면 계산이 거의 불가능해 진다.
  - Use Kernel Trick

Nonlinear SVM Classifier

- **Adding features to make a dataset linearly separable**
  - Add a second feature $x_2 = (x_1)^2$



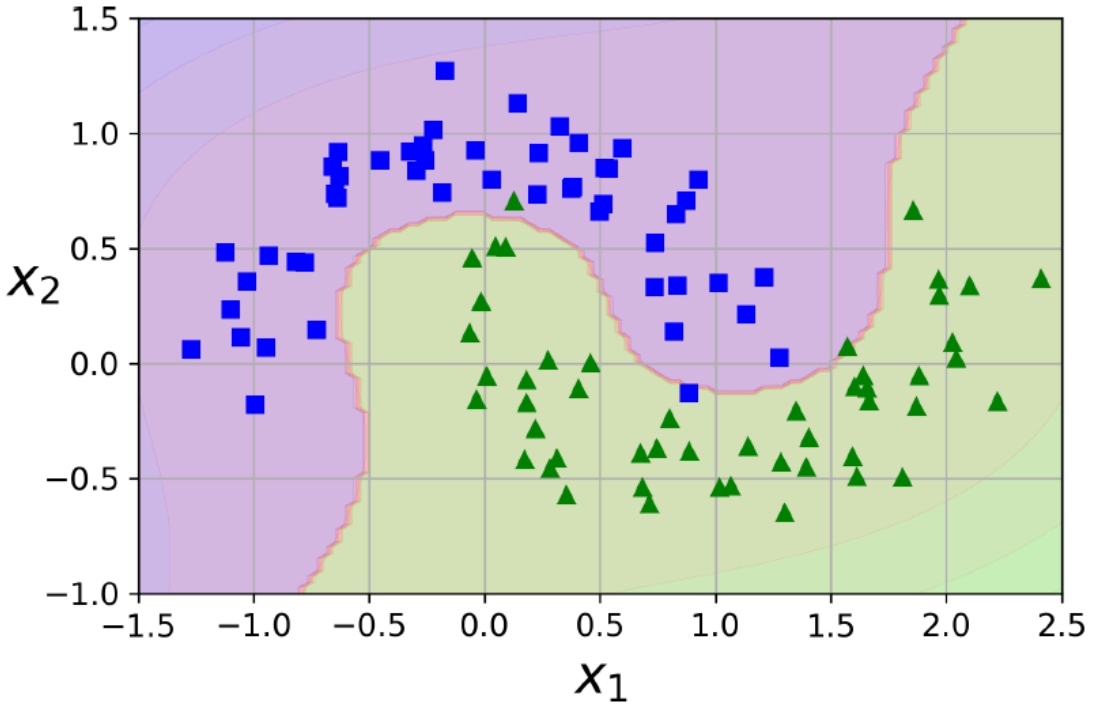
Kernel Trick

- **커널 트릭의 원리**
  - 커널 트릭은 데이터를 실제로 고차원 공간으로 변환하지 않고, 마치 변환된 것처럼 두 데이터 포인트 간의 유사도(내적)를 저차원 공간에서 효율적으로 계산해 주는 방법
  - SVM은 내적 계산을 통해 데이터가 결정 경계에 대해 어떤 위치에 있는지 판단하는데, 커널 트릭은 이 내적 계산을 고차원 공간으로 확장한 것,
  - 즉, 커널 함수 K(a,b)는 두 데이터 포인트 a와 b를 고차원 공간으로 매핑한 후의 내적 값 ($\Phi(a)^T\Phi(b)$)을 저차원에서 직접 계산한다.
- **주요 커널 함수**

- **선형 커널 (Linear Kernel):** $K(a,b) = a^T b$. 선형적으로 분리 가능한 데이터에 사용한다.
- **다항식 커널 (Polynomial Kernel):** $K(a,b) = (a^T b + r)^d$. 비선형 데이터에 사용하며, d는 다항식의 차수, r은 계수를 결정한다.
- **방사 기저 함수 커널 (RBF Kernel):** $K(a,b) = \exp(-\gamma||a-b||^2)$. SVM에서 가장 널리 사용되는 커널로, 데이터 포인트들을 무한 차원의 특징 공간으로 매핑하는 효과를 낸다. γ 값은 커널의 영향력 범위를 조절한다.
- **시그모이드 커널 (Sigmoid Kernel):** $K(a,b) = \tanh(\gamma a^T b + r)$.

Nonlinear SVM Classifier

- **Linear SVM classifier using Polynomial Features**
  - Kernel 사용 대신 명시적으로 기존 특성을 다항식 특성 (e.g. x2, x3)으로 확장한 후 고차원 공간에서 선형 분류기 학습시킨다. (명시적으로 고차원 공간으로 변환)
  - $K(a,b)=(a^Tb+r)^d$ 를 사용하면, 데이터를 직접 고차원으로 변환하지 않고도 다항식 특성을 사용하는 것과 같은 효과를 얻는다. (Kernel Trick)



10 features: c, x1, x2, $x1^2$, $x2^2$, x1x2, $x1^3$, $x2^3$, $x1^2x2$, $x1x2^2$

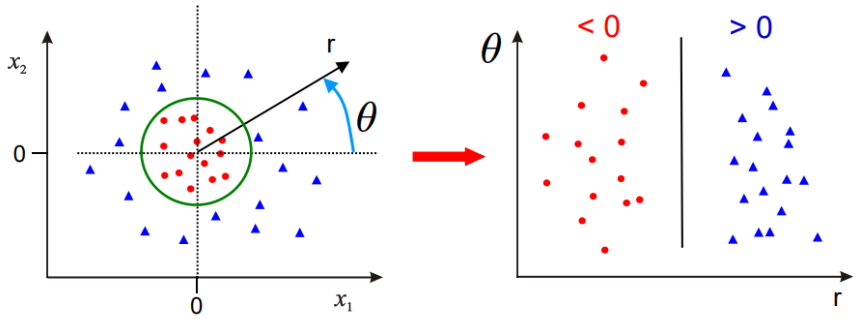
```
polynomial_svm_clf = Pipeline([
        ("poly_features", PolynomialFeatures(degree=3)),
        ("scaler", StandardScaler()),
        ("svm_clf", LinearSVC(C=10, loss="hinge", random_state=42))
    ])

polynomial_svm_clf.fit(X, y)
```

Nonlinear SVM Classifier

- **General (linearly) non-separable data**

Use polar coordinates

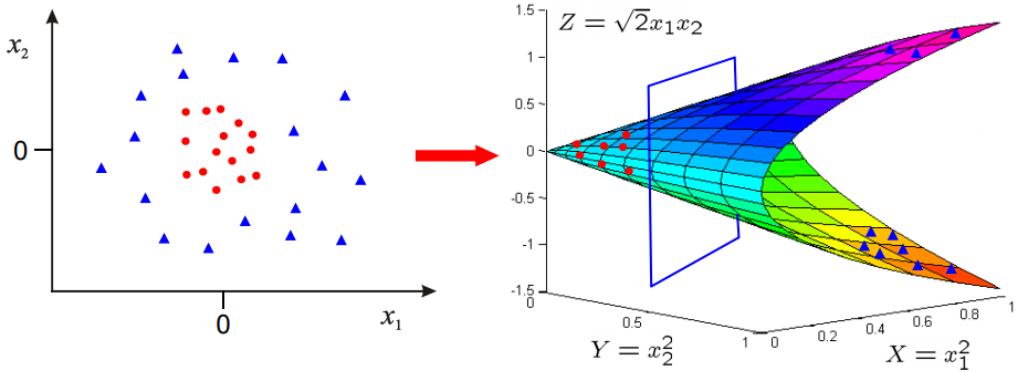


- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Map data to higher dimension

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



Feature space

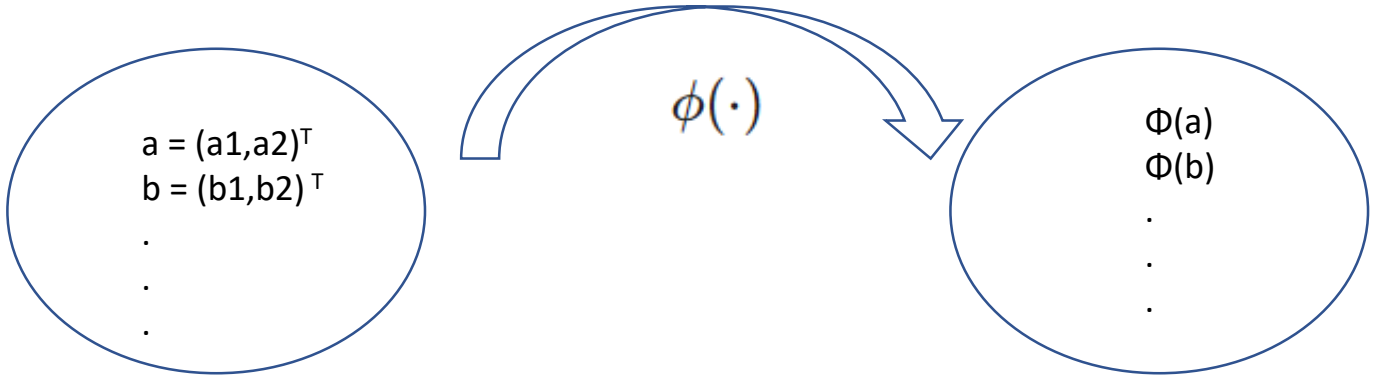
Mapping to Higher Dimension

- Problem:
  - 데이터를 실제로 고차원으로 변환하면 계산량이 너무 많아져 비효율적이다.
  - 특히 데이터의 특성 수가 수백, 수천 개가 되면 계산이 거의 불가능해진다.
- Solution -> **Kernel Trick** (커널 트릭)
  - 데이터를 실제로 고차원으로 변환하지 않고, 마치 고차원으로 변환된 것처럼 두 데이터 포인트 간의 유사도(내적)를 효율적으로 계산해 주는 함수
- Inner Product and Similarity (내적과 유사도)
  - 선형 분류기는 $w^Tx + b = 0$라는 결정 경계(초평면)를 사용해 데이터를 분류 (w^Tx는 w와 x의 내적 연산), 즉 내적 계산은 데이터 포인트가 결정 경계에 대해 어떤 위치에 있는지 판단하는 가장 기본적인 연산이다.
  - 커널 트릭은 이 내적 계산을 고차원 공간으로 확장한다. 커널함수 K(a,b) 는 두 데이터 포인트 a와 b를 고차원 공간으로 매핑한 후의 내적 값($\phi(a)^T\phi(b)$)을 저차원에서 직접 계산한다.
  - 이 계산을 통해 데이터를 실제로 고차원으로 변환하지 않고도, 고차원에서의 유사도를 측정할 수 있다.

Kernel Trick (Example)

$$\phi(a) = (a_1^2, a_2^2, \sqrt{2}a_1a_2, \sqrt{2}a_1, \sqrt{2}a_2, 1)^T$$
$$\phi(b) = (b_1^2, b_2^2, \sqrt{2}b_1b_2, \sqrt{2}b_1, \sqrt{2}b_2, 1)^T$$

2-dimensional data space



6-dimensional kernel space

$$K(a,b) = (a^Tb + r)^d \qquad \text{(r=1, d=2)}$$
$$= (a^Tb + 1)^2 = (a_1b_1 + a_2b_2 + 1)^2$$
$$= a_1^2b_1^2 + a_2^2b_2^2 + 2a_1a_2b_1b_2 + 2a_1b_1 + 2a_2b_2 + 1$$

$$\phi(a)^T\phi(b) = (a_1^2)(b_1^2) + (a_2^2)(b_2^2) + (\sqrt{2}a_1a_2)(\sqrt{2}b_1b_2) + (\sqrt{2}a_1)(\sqrt{2}b_1) + (\sqrt{2}a_2)(\sqrt{2}b_2) + (1)(1)$$
$$= a_1^2b_1^2 + a_2^2b_2^2 + 2a_1a_2b_1b_2 + 2a_1b_1 + 2a_2b_2 + 1$$
$$= (a_1b_1 + a_2b_2 + 1)^2 = (a^Tb + 1)^2$$

low-dimensional Kernel function = dot product in the high-dimensional space

Kernel Functions

- Kernel Functions 의 수학적 조건: 머서의 정리(Mercer's Theorem)

The theorem states that a symmetric function $K(x, z)$ is a kernel if and only if it satisfies the following condition: for any finite set of points $\{x_1, \ldots, x_m\} \subset \mathcal{X}$ and any coefficients $\{c_1, \ldots, c_m\} \subset \mathbb{R}$, the quadratic form is non-negative:

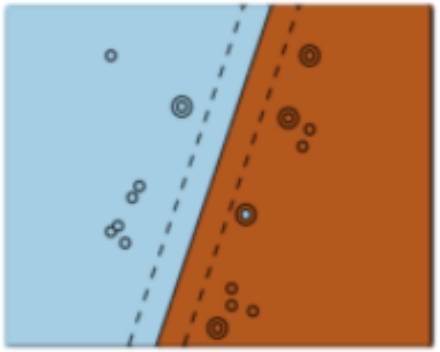
$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j K(x_i, x_j) \geq 0$$

This condition is equivalent to stating that the Gram matrix (or kernel matrix) K with entries $K_{ij} = K(x_i, x_j)$ must be **positive semi-definite**.

- The most common Kernel Functions
  - **Linear Kernel**: $K(a, b) = a^T b$
  - **Polynomial Kernel**: $K(a, b) = (a^T b + r)^d$
  - **Radial Basis Function (RBF) Kernel**: $K(a, b) = \exp(-\gamma \|a - b\|^2)$
  - **Sigmoid Kernel**: $K(a, b) = \tanh(\gamma a^T b + r)$

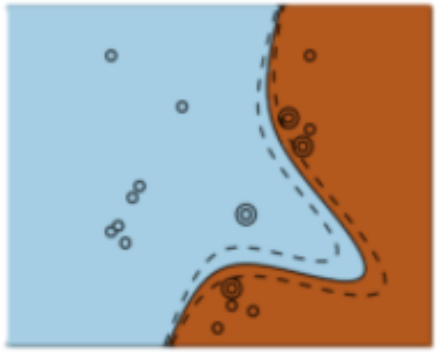
Common SVM Kernels

Linear Kernel



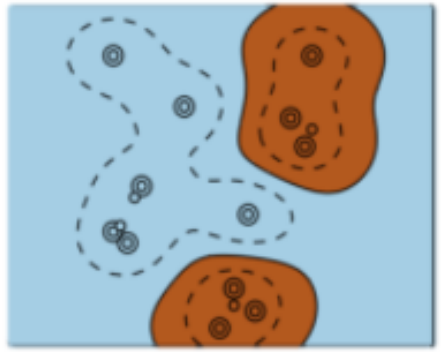
C hyperparameter

Polynomial Kernel



C plus gamma, degree and coefficient hyperparameters

RBF Kernel



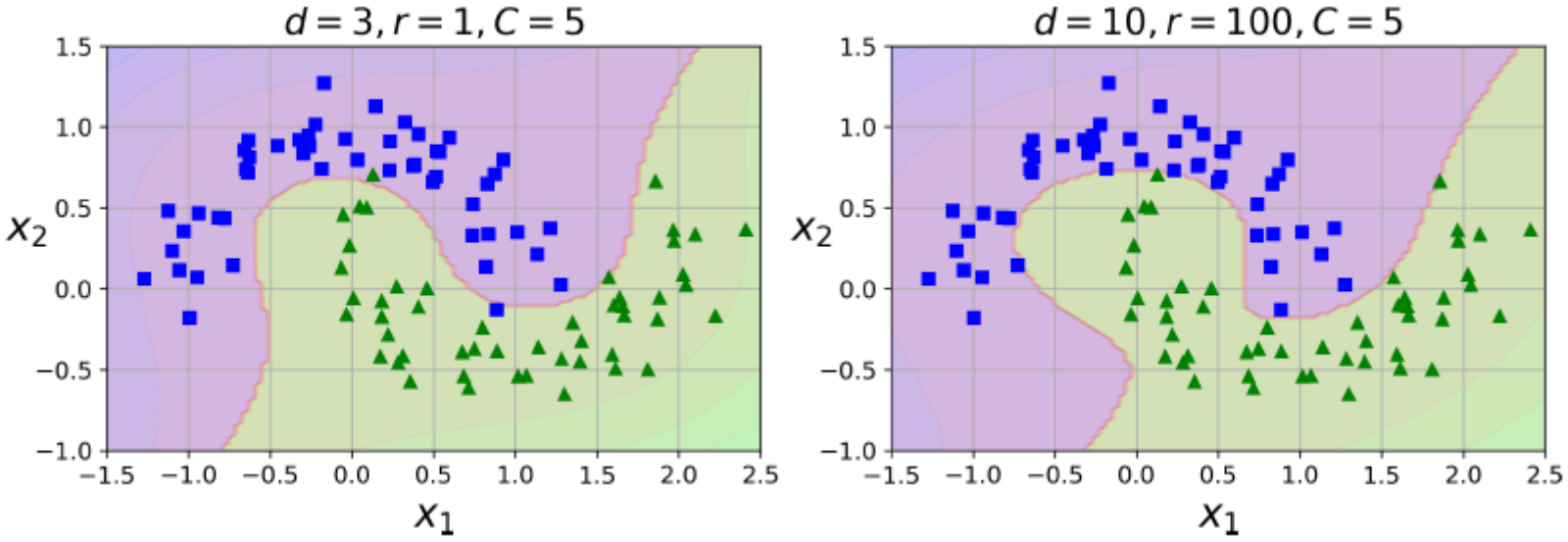
C plus gamma hyperparameter

SVM Kernels

- **SVM classifiers with a Polynomial Kernel**

K(a,b) = $(a \times b + r)^d$

- a, b: 서로 다른 데이터
- r: polynomial의 coefficient를 결정
- d: polynomial의 차수



RBF Kernel

- **RBF Kernel**
  - SVM에서 가장 널리 사용되는 커널 함수로, 데이터 포인트들을 무한 차원의 특징 공간으로 매핑하는 효과를 낸다. 이 덕분에 복잡한 비선형 데이터도 효과적으로 분류할 수 있다.
  - RBF 커널 함수는 두 데이터 포인트 a와 b 사이의 유클리드 거리($\|a-b\|^2$)를 기반으로 유사도를 측정 (여기서 γ는 하이퍼파라미터로, 커널의 영향력 범위를 조절한다.)

$$K(a,b) = \exp(-\gamma\|a-b\|^2)$$

  - γ 값이 작을 때:
    - 커널의 영향력이 넓어져, 멀리 떨어진 데이터 포인트들도 서로 영향을 미치게 된다. 결과적으로 결정 경계가 부드럽고 넓어진다.
    - 이 경우 모델이 지나치게 단순해져 과소적합(underfitting)이 발생할 수 있다.
  - γ 값이 클 때:
    - 커널의 영향력이 좁아져, 가까운 데이터 포인트들만 서로 영향을 미치게 된다. 결정 경계가 데이터 포인트 하나하나에 맞춰 불규칙하고 복잡해진다.
    - 이 경우 모델이 훈련 데이터에 너무 맞춰져 과대적합(overfitting)될 가능성이 높아진다.

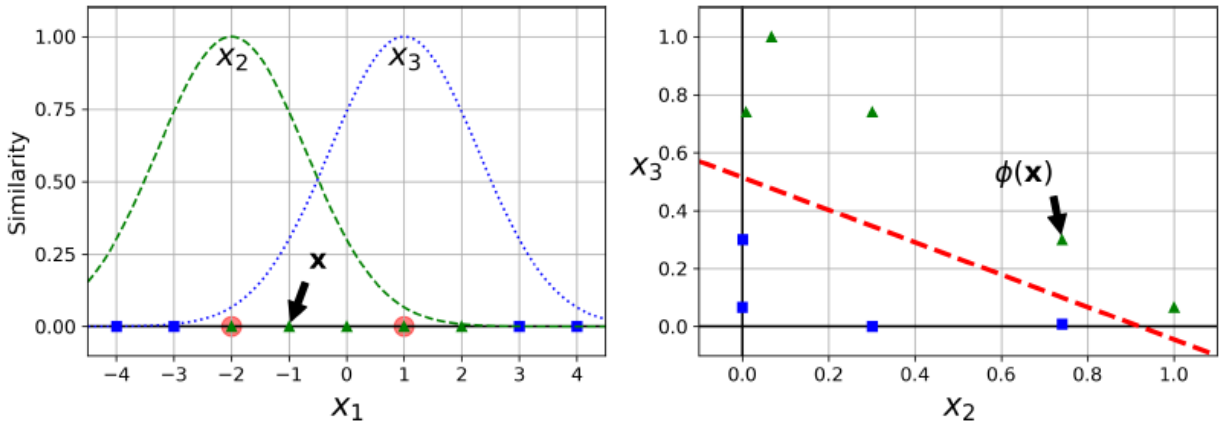
RBF Kernel

- **SVM with Similarity features**
  - Use a **similarity function** that measures how much each instance resembles a particular *landmark*.
  - define the similarity function: **Gaussian Radial Basis Function (*RBF*)**

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

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

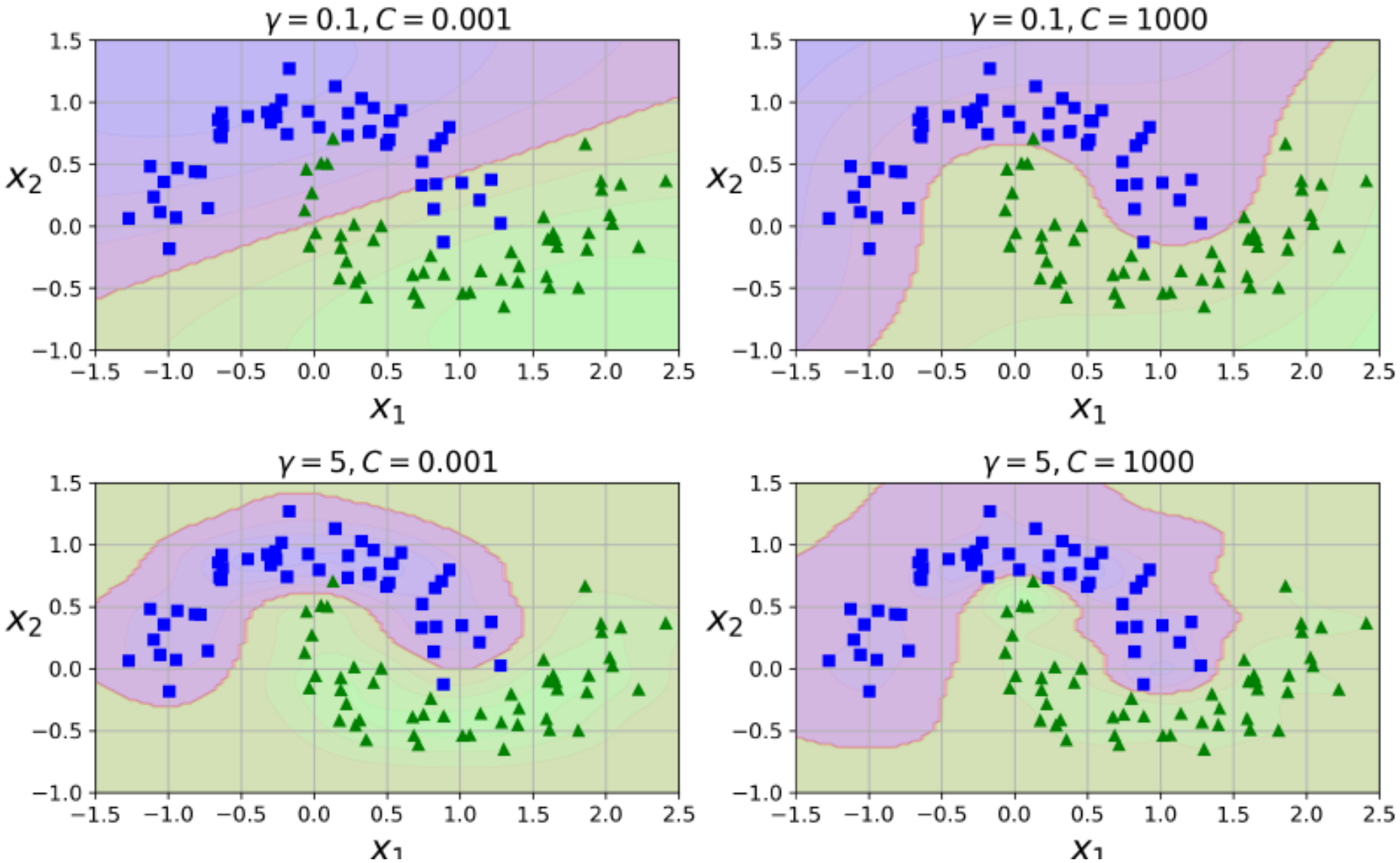
original feature:
x_1 = -1



Take two landmarks at x_1=-2, x_1=1
then, new features are:
x_2 = exp (–0.3 × 1^2) ≈ 0.74
x_3 = exp (–0.3 × 2^2) ≈ 0.30
now, linearly separable.

- Create landmarks at the location of each instance, then (assuming drop of the original features)
  m instances, *n* features -> *m* instances, *m* features (large features !)

SVM RBF Kernel



SVM Optimization (Training)

- **Hard Margin**

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{Subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

- **Soft Margin**

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

SVM 예측 및 결론

- **선형 SVM 예측**
  - 훈련을 통해 구한 가중치 벡터 w와 편향 b를 이용하여 새로운 데이터 x에 대한 예측을 수행한다
  - 결정 함수:

$$f(x) = w^T x + b$$

  - f(x)>0 이면 클래스 +1로 예측하고, f(x)<0 이면 클래스 −1로 예측한다
- **비선형 SVM 예측**
  - 커널 트릭을 사용한 비선형 SVM에서는 결정 함수에 커널 함수 $K(x_i, x)$를 사용한다.
  - 결정 함수:

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b$$

  - SVM의 예측은 서포트 벡터(SV)에 의해서만 결정되므로, 모든 훈련 데이터가 아닌 일부 데이터만으로도 효율적인 추론이 가능하다.

QP (Quadratic Programming)

- **QP (Quadratic Programming)의 중요성**
  - SVM의 최적화 문제는 2차 함수 형태의 목적 함수와 선형 제약 조건을 가지므로, Quadratic Programming(QP) 문제에 속한다
  - QP는 목적 함수가 2차식이므로, 유일한 전역 최솟값(unique minimum)을 가지는 경우가 많아 안정적인 해를 찾을 수 있다
  - SVM은 원형(Primal) 문제를 쌍대(Dual) 문제로 변환하여 해결한다. 이 쌍대 형식에서도 최적화 문제는 여전히 2차 함수 형태를 유지한다.
- **Gradient Descent (경사하강법) vs. QP**
  - 경사하강법(Gradient Descent): 1차 미분 정보(경사)만 사용하여 최적의 해를 찾아가는 방법이다. 수렴 속도가 느리고 최적의 학습률(learning rate)을 찾는 튜닝 과정이 필요하다.
  - QP (Interior-Point Methods): 2차 미분 정보(헤시안 행렬)를 사용하여 더 빠르고 안정적으로 수렴한다. 특히 선형 제약 조건을 직접 처리할 수 있도록 설계되어 SVM 문제에 매우 적합하다.

Why QP is Preferred

- **Why QP is preferred?**
  - 수렴 속도 및 안정성: SVM의 목적 함수는 2차 함수 형태를 띠고 선형 제약 조건을 가지므로 , QP 알고리즘이 훨씬 빠르고 안정적으로 수렴한다. Subgradient descent 는 1차 정보(경사)만 사용하기 때문에 수렴이 느릴 수 있다.
  - 제약 조건 처리: QP는 등식/부등식 제약 조건을 직접 처리하도록 설계되어 있어 SVM 문제에 매우 적합하다. 반면 subgradient descent는 제약 조건을 직접 처리하기 어렵다.
  - 최적화 보장: SVM의 2차 함수 최적화 문제는 유일한 최솟값을 가지므로 , QP 알고리즘을 사용하면 전역 최솟값을 안정적으로 찾을 수 있다.

SVC and SVR

- SVC (Support Vector Classification)
  - 목표: 두 클래스를 분리하는 최대 마진 초평면을 찾는다
  - 출력: 초평면을 기준으로 클래스 +1 또는 -1을 예측한다
  - 핵심개념: 클래스를 나누는 경계선(boundary)이 중요하다. 마진은 두 클래스에 속한 가장 가까운 데이터 포인트(서포트 벡터) 사이의 거리다
- SVR (Support Vector Regression):
  - 목표: 예측값과 실제값의 오차를 최소화하는 회귀 함수를 찾는다.
  - 핵심 개념: ϵ-불변 튜브(ϵ-insensitive tube)를 설정하는 것이 핵심이다. 모델은 오차 허용 범위(ϵ) 내에 최대한 많은 데이터 포인트를 포함하도록 회귀선을 최적화한다.
  - 손실 함수: 예측 오차가 ϵ보다 작으면 손실을 0으로 간주하고, ϵ보다 클 때만 페널티를 부과하여 이상치에 강한 특징을 가진다.

SVM Regression

- It also supports linear and nonlinear regression.
- Object:
  - SVM Regression tries to fit as many instances as possible *on* the street while limiting margin violations (i.e., instances *off* the street).
  - The width of the street is controlled by a hyper-parameter ϵ.

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \quad \text{subject to} \quad \begin{aligned} & y_i - (w^T x_i + b) \leq \epsilon + \xi_i \\ & (w^T x_i + b) - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

regularization term: 회귀 함수를 가능한 한 평평하게 만들어 과적합(overfitting)을 방지하는 역할

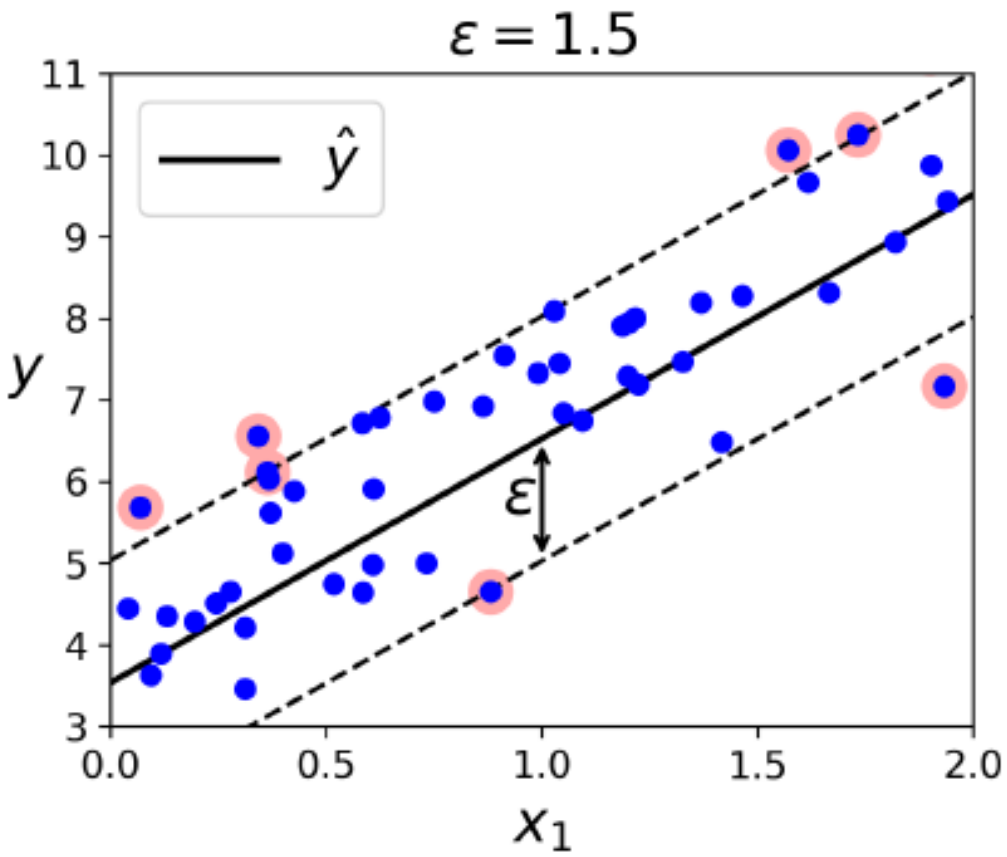
loss term: 마진 ϵ을 벗어나는 오차에 페널티를 부과

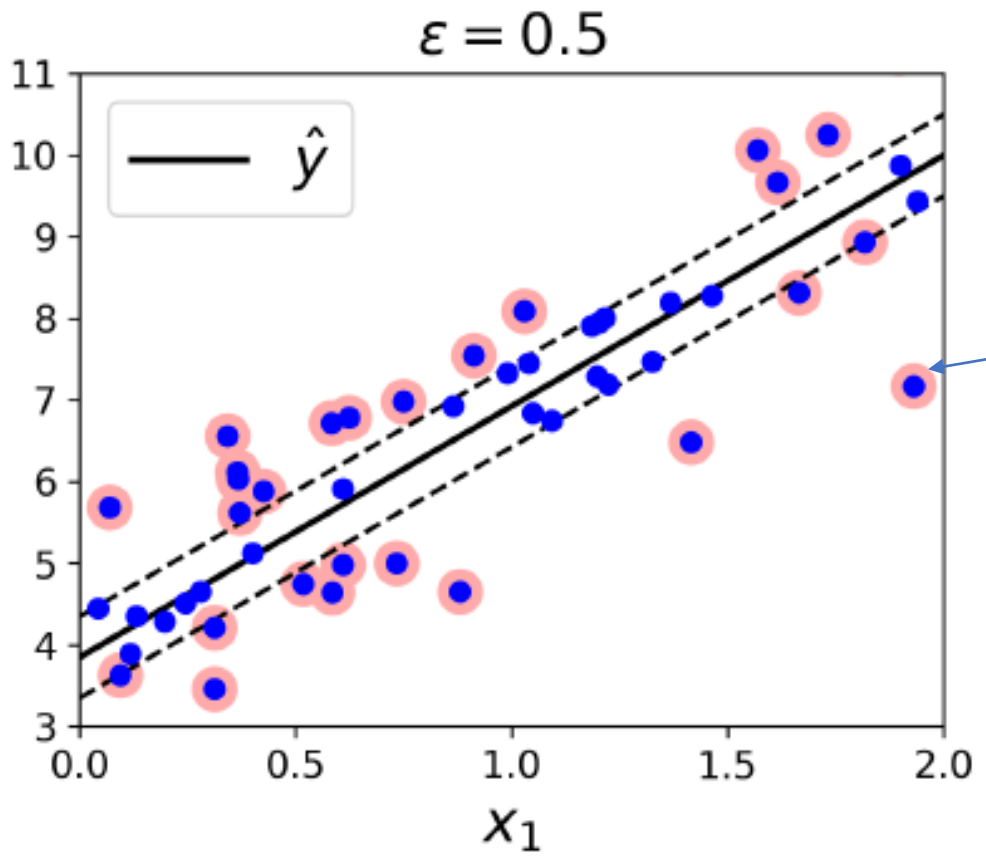
- ϵ: ϵ-insensitive tube의 폭을 정의 (이 튜브 안의 오차는 무시되며, 페널티를 받지 않는다)
- ξi and ξi*: 각 데이터 포인트가 ϵ 튜브를 벗어난 정도를 측정하는 슬랙 변수(slack variables) - 튜브를 벗어난 경우 두 슬랙 변수 중 하나만 0보다 큼 (튜브 안에 있는 경우는 둘 다 0)
- C: 회귀 함수의 평평함(∥w∥2 최소화)과 ϵ 튜브를 벗어나는 오차 (Σ(ξi+ξi*) 최소화) 사이의 균형을 조절

SVM Regression

- Loss Function:
  - ϵ-불변 손실(ϵ-insensitive loss)을 사용한다. 이 손실 함수는 예측값과 실제값 사이의 오차가 ϵ보다 작으면 손실을 0으로 간주하고, ϵ보다 클 때만 페널티를 부과한다. 이 때문에 SVR은 이상치(outlier)에 강한(robust) 특징이 있다.
- Hyperparameters
  - C: 튜브를 벗어나는 데이터에 대한 페널티를 조절
  - ϵ: 오차를 허용하는 튜브의 폭을 직접 조절
  - gamma: RBF 커널의 경우, 커널의 영향력 범위를 조절

SVM Regression

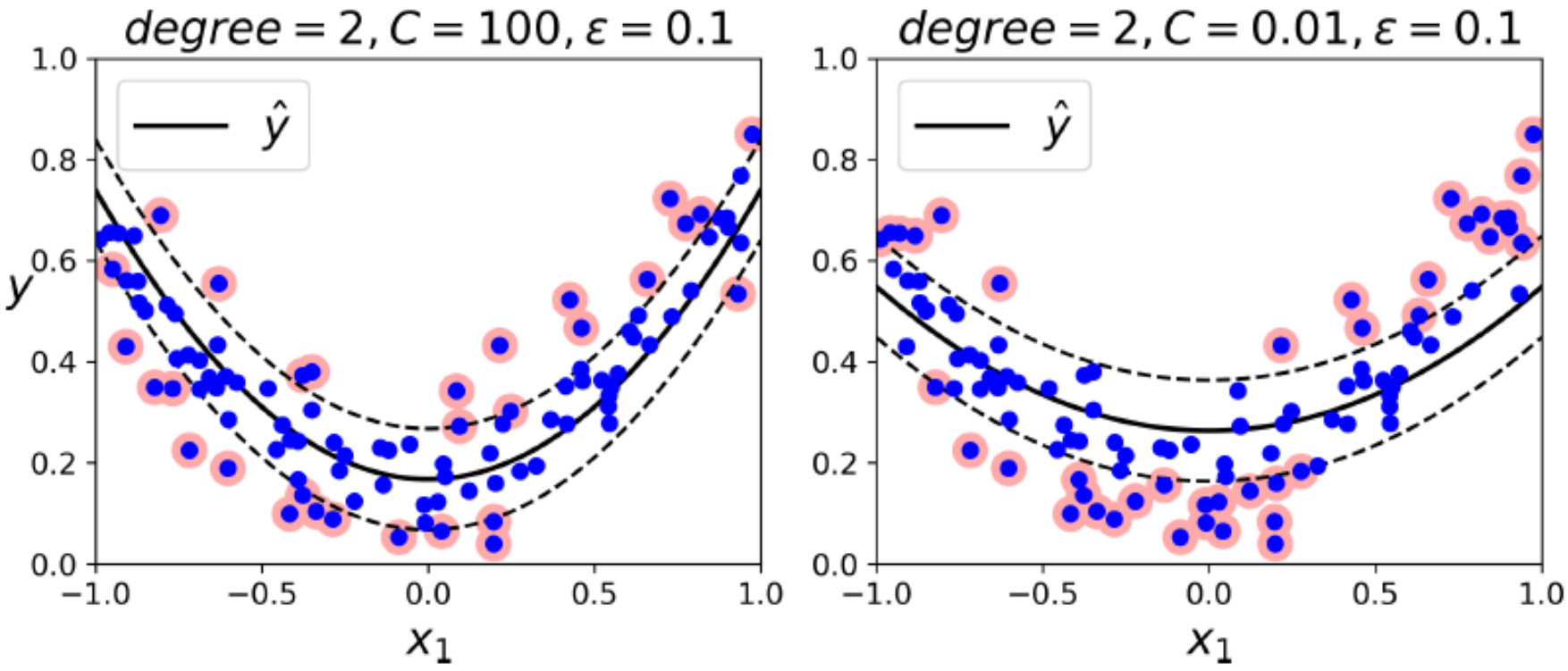




margin violation

SVM Regression

- **SVM Regression using 2nd degree Polynomial Kernel**



SVM Regression

- **Loss function: ε-insensitive loss:**
  - Ignores errors that are within ε distance by treating them as zero
  - Measured based on the distance between observed value y and the ε boundary

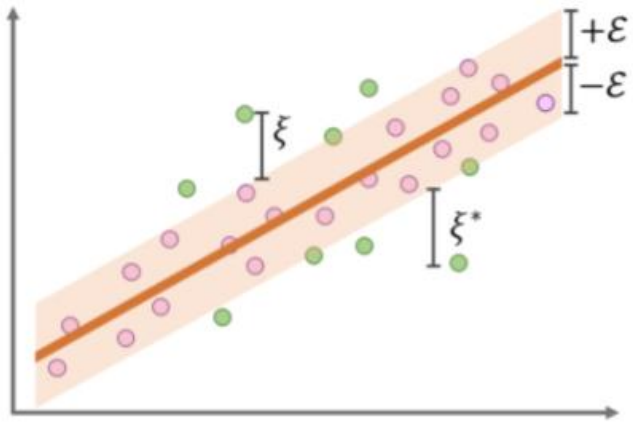
$$L_\varepsilon = \begin{cases} 0 & \text{if } |y - f(x)| \le \varepsilon \\ |y - f(x)| - \varepsilon & \text{otherwise} \end{cases}$$

$$L_{SVR} = \min \overbrace{\frac{1}{2}\|w\|^2}^{\text{Robustness}} + C\underbrace{\sum_{i=1}^{n}(\xi_i + \xi_i^*)}_{\text{loss funciton}}$$

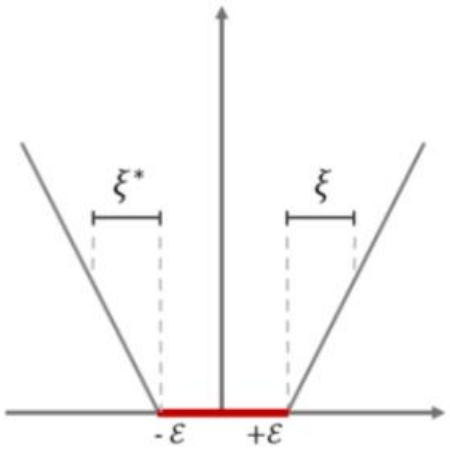
$$s.t. \quad (w^T x_i + b) - y_i \le \epsilon + \xi_i$$

$$y_i - (w^T x_i + b) \le \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$



● : *Support Vector*



ε − insensitive loss

SVC (Summary)

- **SVC**
  - the goal: to find a **hyperplane** that separates data points from two classes with the maximum margin while correctly classifying as many data points as possible
  - Primal form (soft margin):

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \qquad \text{subject to:} \qquad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

  - Dual form (can handle non-linear data via the kernel-trick):

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \qquad \text{subject to:} \qquad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C$$

quadratic

  - Quadratic Programming in SVC
    - objective function: quadratic because of $\|\mathbf{w}\|^2$ (primal) and Lagrangian (dual)
    - constraints: Linear inequality constraints involving $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$

SVR (Summary)

- **SVR**
  - the goal: to find a **regression function** that deviates from the actual target values by no more than a margin ϵ, while penalizing deviations greater than ϵ using slack variables.
  - Primal form:

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \quad \text{subject to:} \quad \begin{aligned} y_i - (\mathbf{w}^T\mathbf{x}_i + b) &\leq \epsilon + \xi_i \\ (\mathbf{w}^T\mathbf{x}_i + b) - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0 \end{aligned}$$

  - Dual form:

$$\max_{\alpha,\alpha^*} \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i - \frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{subject to:} \quad \begin{aligned} \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) &= 0 \\ 0 \leq \alpha_i, \alpha_i^* &\leq C \end{aligned}$$

SVC and SVR

Aspect	SVC (Classification)	SVR (Regression)
Goal	Maximize the margin between two classes.	Minimize prediction error while allowing tolerance ϵ.
Output	A **decision boundary** (hyperplane) separating classes.	A **regression function** that approximates continuous data.
Formulation	$\min \frac{1}{2}\|\mathbf{w}\|^2$ with classification constraints.	$\min \frac{1}{2}\|\mathbf{w}\|^2$ with ϵ-insensitive loss.
Slack Variables	ξ_i for allowing some misclassification.	ξ_i and ξ_i^* for allowing deviations beyond ϵ.
Quadratic Program	Dual form uses quadratic programming for optimizing Lagrange multipliers.	Dual form also uses quadratic programming for optimizing Lagrange multipliers.

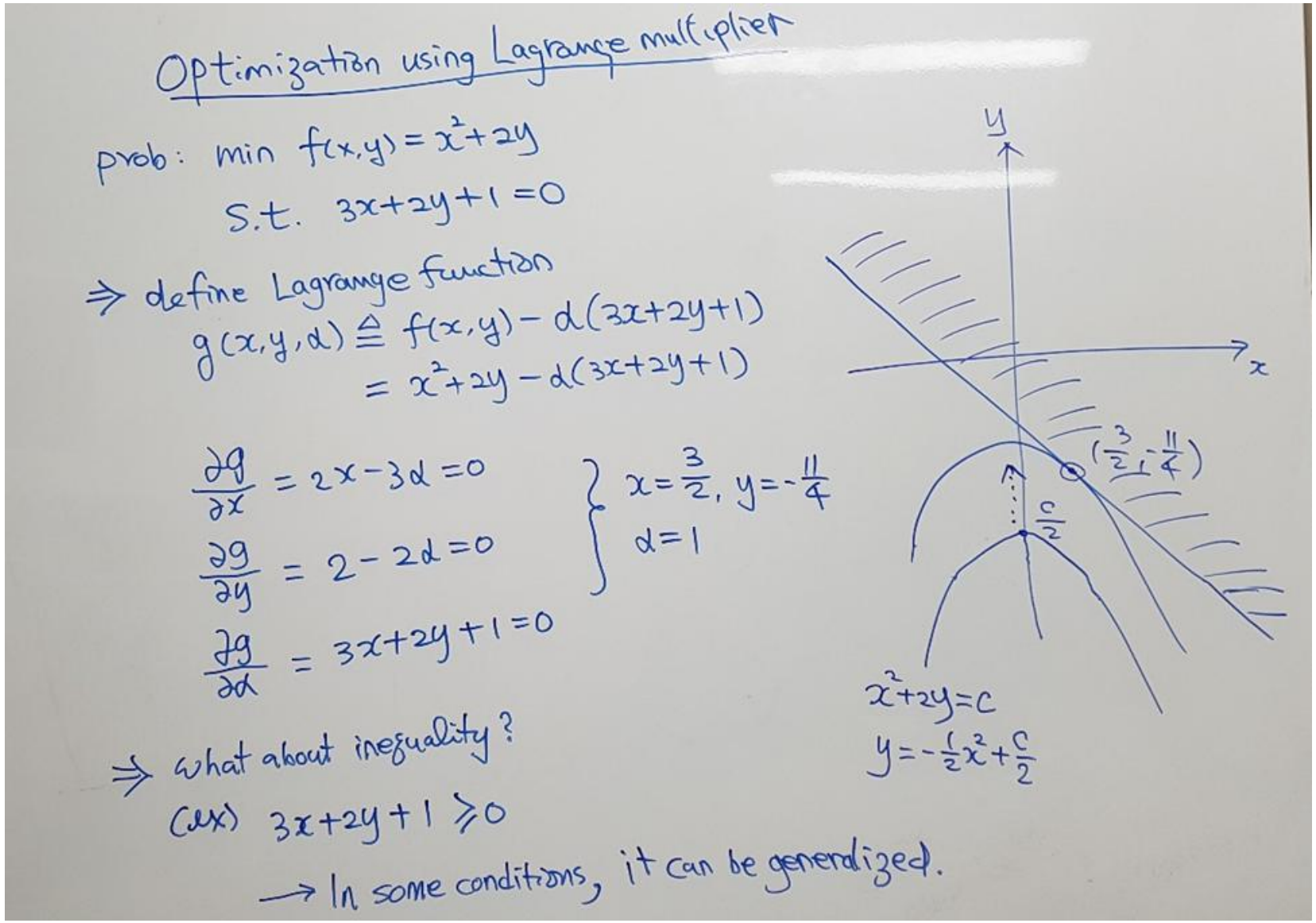
- Both SVC and SVR use **Quadratic Programming (QP)** to solve their respective optimization problems.

Summary

- SVM은 최대 마진이라는 명확한 목표를 가진 분류/회귀 알고리즘이다.
- 커널 트릭을 통해 비선형 데이터도 효과적으로 처리할 수 있다.
- QP를 통해 안정적이고 효율적인 최적화가 가능하다.
- 서포트 벡터만을 이용해 예측을 수행하므로 효율적이다.

[참고] SVM Optimization

SVM Optimization – Lagrangian Multiplier



SVM (Hard Margin)

1. Original (Primal) Formulation:

In the primal form of SVM, the objective is to minimize the **norm of the weight vector** $\mathbf{w}$, which is a quadratic function of $\mathbf{w}$. This can be expressed as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

This objective function is quadratic in $\mathbf{w}$.

2. Constraints:

The classification constraints in the primal form are:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

If we are dealing with a soft-margin SVM, there are additional **slack variables** ξ_i, but the fundamental structure involves linear inequality constraints.

SVM (Hard Margin)

3. Lagrangian Formulation:

To convert the primal problem into its dual, we use the **Lagrangian function**, which introduces **Lagrange multipliers** α_i for each constraint.

The Lagrangian for the SVM problem is:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

SVM (Hard Margin) - Dual formulation

4. Deriving the Dual Form:

Step 1: Minimize the Lagrangian with Respect to $\mathbf{w}$ and b:

To get the dual form, we first minimize the Lagrangian with respect to the primal variables $\mathbf{w}$ and b. We take partial derivatives of $\mathcal{L}(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b, and set them to zero to eliminate these variables:

- **Partial Derivative w.r.t. $\mathbf{w}$:**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- **Partial Derivative w.r.t. b:**

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Step 2: Substitute $\mathbf{w}$ Back into the Lagrangian:

Substituting $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ back into the Lagrangian gives the **dual objective function**. After simplification, the dual objective function becomes:

$$\mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

SVM (Hard Margin) - Dual Formulation

5. Why is it Quadratic? ← 2차함수 최적화 문제

- **Objective Function**: The dual objective function contains the **term**:

$$-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j)$$

  This term is **quadratic** in the **Lagrange multipliers** α_i, since it involves products of the multipliers $\alpha_i\alpha_j$.

- **Kernel or Feature Map**: In the dual form, the data points are only involved through the dot product $(\mathbf{x}_i^T\mathbf{x}_j)$, or more generally, through the **kernel function** $K(\mathbf{x}_i, \mathbf{x}_j)$. This dot

- Thus, the dual form retains a **quadratic structure**, but now the optimization is done over the **Lagrange multipliers** α_i rather than directly over the weight vector $\mathbf{w}$.

SVM (Hard Margin) - Dual Formulation

6. Dual Problem (Final Form):

The dual optimization problem is:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C$$

The objective function remains quadratic because of the $\alpha_i \alpha_j$ product terms.

SVM (Soft Margin)

1. Soft-Margin SVM: Primal Formulation

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

subject to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

2. Soft-margin SVM Lagrangian

$$\mathcal{L}(\mathbf{w}, b, \alpha, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{n}\mu_i\xi_i$$

Where:

- $\alpha_i \geq 0$ are the Lagrange multipliers for the classification constraints.
- $\mu_i \geq 0$ are the Lagrange multipliers for the slack variables.

SVM (Soft Margin)

3. Solving for w and b

1. **Minimizing with respect to $\mathbf{w}$:**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

2. **Minimizing with respect to b:**

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

3. **Minimizing with respect to ξ_i:** By setting the derivative of $\mathcal{L}$ with respect to ξ_i to zero, we get:

$$\alpha_i + \mu_i = C$$

SVM (soft margin) - Dual Formulation

4. Dual Formulation of Soft-Margin SVM

Substituting these expressions into the Lagrangian, we get the **dual form** of the soft-margin SVM:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

αi : 각 데이터에 대한 Lagrangian (각 support vector 에 대한 가중치이며 support vector 에 대해서만 nonzero 값을 갖게 됨)

subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C$$

1. The Lagrange multipliers α_i are now **bounded** by the parameter C (i.e., $0 \leq \alpha_i \leq C$). This constraint ensures that the slack variables ξ_i are appropriately penalized based on the value of C.
2. The slack variables ξ_i allow for margin violations, which means that not all the data points must satisfy the strict margin constraint.

SVC Prediction (or Inference)

- **Calculate w and b**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$\mathbf{x}_i$: support vectors (SV)
y_i : corresponding class labels (± 1)
α_i : lagrange multiplier for corresponding SV's
(SV 에 대한 가중치, 즉, non-zero 데이터 포인트 (SV)들 만이 결정 경계에 영향을 미침)

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1 \quad \Rightarrow \quad b = y_i - \mathbf{w}^T \mathbf{x}_i$$

SV 가 여러 개면 평균

- **Prediction**

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

SV 만을 고려하여 추론이 이루어지므로 효율적

- $f(\mathbf{x}) > 0$이면 클래스 **+1**을 예측.
- $f(\mathbf{x}) < 0$이면 클래스 **-1**을 예측.

SVC Prediction (or Inference)

1 Linear SVM: Prediction for New Input $\mathbf{x}$

For a **linear SVM**, the decision function for a new input $\mathbf{x}$ is:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

where:

- $\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$, which is the sum of the support vectors weighted by their Lagrange multipliers α_i and their labels y_i.
- b is the bias term found during training.

To make a **prediction** for a new input $\mathbf{x}$:

- If $f(\mathbf{x}) > 0$, the model predicts **class +1**.
- If $f(\mathbf{x}) < 0$, the model predicts **class -1**.

Thus, the prediction is based on the **sign** of the decision function:

$$\hat{y} = \text{sign}(f(\mathbf{x})) = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right)$$

SVC Prediction (or Inference)

2 Non-Linear SVM (with Kernel Trick)

For **non-linear SVMs**, where you use a **kernel function** to map the input into a higher-dimensional feature space, the prediction for a new input $\mathbf{x}$ is similar, but the decision function uses the **kernel function** $K(\mathbf{x}_i, \mathbf{x})$ instead of the dot product $\mathbf{x}_i^T \mathbf{x}$.

The decision function for a **non-linear SVM** is:

$$f(\mathbf{x}) = \sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

where:

- α_i are the Lagrange multipliers for the support v
- y_i are the labels of the support vectors.
- $K(\mathbf{x}_i, \mathbf{x})$ is the **kernel function** (e.g., **linear**, **pol**
- b is the bias term.

Make the Prediction: The class prediction $\hat{y}$ is determined by the **sign** of the decision function:

$$\hat{y} = \text{sign}(f(\mathbf{x})) = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

- If $f(\mathbf{x}) > 0$, predict **class +1**.
- If $f(\mathbf{x}) < 0$, predict **class -1**.

Quadratic Programming

Quadratic Programming (example)

- **General form:**

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

$$A\mathbf{x} = \mathbf{b}$$

Subject to

$$G\mathbf{x} \leq \mathbf{h}$$

x: variable to optimize
Q: positive semi-definite matrix
($x^TQx >= 0$, or all $e_i >= 0$)

- **Example**

$$\min_{\mathbf{x}} \frac{1}{2}\left(x_1^2 + x_2^2\right) + (-2x_1 - 3x_2)$$

$$x_1 + x_2 = 1 \quad \text{(Equality constraint)}$$

$$x_1 \geq 0, \quad x_2 \geq 0 \quad \text{(Inequality constraints)}$$

$$\mathcal{L}(x_1, x_2, \lambda) = \frac{1}{2}\left(x_1^2 + x_2^2\right) - 2x_1 - 3x_2 + \lambda(x_1 + x_2 - 1)$$

$$\frac{\partial \mathcal{L}}{\partial x_1} = x_1 - 2 + \lambda = 0 \quad \Rightarrow \quad x_1 = 2 - \lambda$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = x_2 - 3 + \lambda = 0 \quad \Rightarrow \quad x_2 = 3 - \lambda$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x_1 + x_2 - 1 = 0 \quad \Rightarrow \quad (2 - \lambda) + (3 - \lambda) = 1$$

$$5 - 2\lambda = 1 \quad \Rightarrow \quad 2\lambda = 4 \quad \Rightarrow \quad \lambda = 2$$

$$x_1 = 2 - 2 = 0, \quad x_2 = 3 - 2 = 1$$

Quadratic Programming (example)

```python
import cvxopt
from cvxopt import matrix, solvers

Q = matrix([[1.0, 0.0],
            [0.0, 1.0]])  # Coefficients for x1^2 and x2^2
c = matrix([-2.0, -3.0])  # Coefficients for -2x1 and -3x2

# Inequality constraints (Gx <= h) for x1 >= 0 and x2 >= 0
G = matrix([[-1.0, 0.0], [0.0, -1.0]])  # Inequality matrix for x1 >= 0 and x
h = matrix([0.0, 0.0])  # Inequality vector

# Equality constraint (Ax = b) for x1 + x2 = 1
A = matrix([[1.0, 1.0]]).T  # Equality matrix for x1 + x2
b = matrix([1.0])  # Equality vector

# Solve the QP problem
sol = solvers.qp(Q, c, G, h, A, b)

# Display the solution from cvxopt
optimal_x1, optimal_x2 = sol['x']
print(f"Optimal solution from solver: x1 = {optimal_x1}, x2 = {optimal_x2}")
```

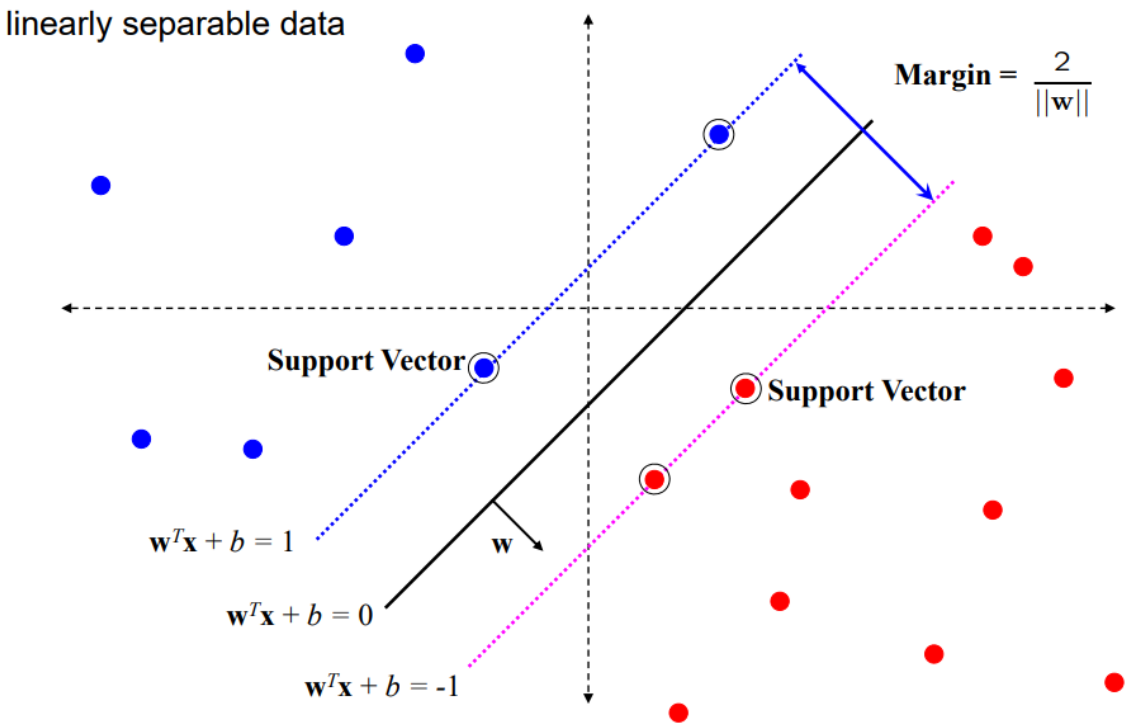
Quadratic Programming

- Quadratic Programming package (cvxopt) uses Interior-Point Methods.

Feature	Gradient Descent	Interior-Point Methods
Type of Optimization	First-order (uses only the gradient).	Second-order (uses gradient and Hessian for better steps).
Constraints	Cannot directly handle constraints.	Specifically designed to handle constraints (inequality/equality).
Convergence	Slower and often requires fine-tuning of the learning rate.	Usually faster convergence due to Newton's method and barrier functions.
Problem Type	Good for unconstrained problems or simple constraints.	Designed for constrained problems, such as QP, LP, etc.
Step Size	Requires careful tuning of the learning rate.	Step size is dynamically computed using second-order information (Hessian).

SVM Optimization Problem

- **Quadratic optimization problem subject to linear constraints**
  - There is a unique minimum.



$$\vec{w} \bullet \vec{x_+} + b \geq \delta \qquad \vec{w} \bullet \vec{x_-} + b \leq -\delta$$

normalize

$$\vec{w} \bullet \vec{x_+} + b \geq 1 \qquad \vec{w} \bullet \vec{x_-} + b \leq -1$$

- SVM is formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{matrix} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{matrix} \text{ for } i = 1 \ldots N$$

Or equivalently

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ subject to } y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 \text{ for } i = 1 \ldots N$$

training

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x >= 0 \\ 0 & \text{otherwise} \end{cases}$$

testing(inferencing)

SVM Optimization for Soft Margin

- **Soft margin**

The optimization problem becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored → large margin
  - large C makes constraints hard to ignore → narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

Or,
cost parameter (or penalty parameter)

SVM Optimization

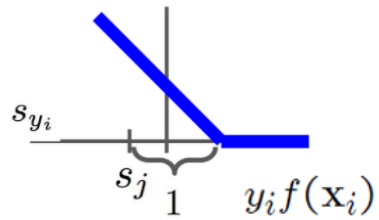
- **Constrained optimization problem**

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i \text{ subject to } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1\ldots N$$

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

$$\Rightarrow \quad \xi_i = \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$



- **The learning problem is now equivalent to the unconstrained optimization problem over w**

$$\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_i^N \underbrace{\max\left(0, 1 - y_i f(\mathbf{x}_i)\right)}_{\text{loss function}}$$

- **Regularization term**: $\frac{1}{2}||w||^2$, encourages simplicity and margin maximization.
- **Error (loss) term**: $C\sum_{i=1}^N \text{Loss}(w, x_i, y_i)$, penalizes misclassifications or points too close to the margin.

Sub-gradient descent algorithm for SVM

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_{i}^{N}\left(\frac{\lambda}{2}||\mathbf{w}||^2 + \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})\right)$$

← λ ~ 1/C

The iterative update is

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta\nabla_{\mathbf{w}_t}\mathcal{C}(\mathbf{w}_t)\\ &\leftarrow \mathbf{w}_t - \eta\frac{1}{N}\sum_{i}^{N}(\lambda\mathbf{w}_t + \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}_t))\end{aligned}$$

where η is the learning rate.

Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda\mathbf{w}_t - y_i\mathbf{x}_i) && \text{if } y_i f(\mathbf{x}_i) < 1\\ &\leftarrow \mathbf{w}_t - \eta\lambda\mathbf{w}_t && \text{otherwise}\end{aligned}$$

(*) QP에 비해 비효율적