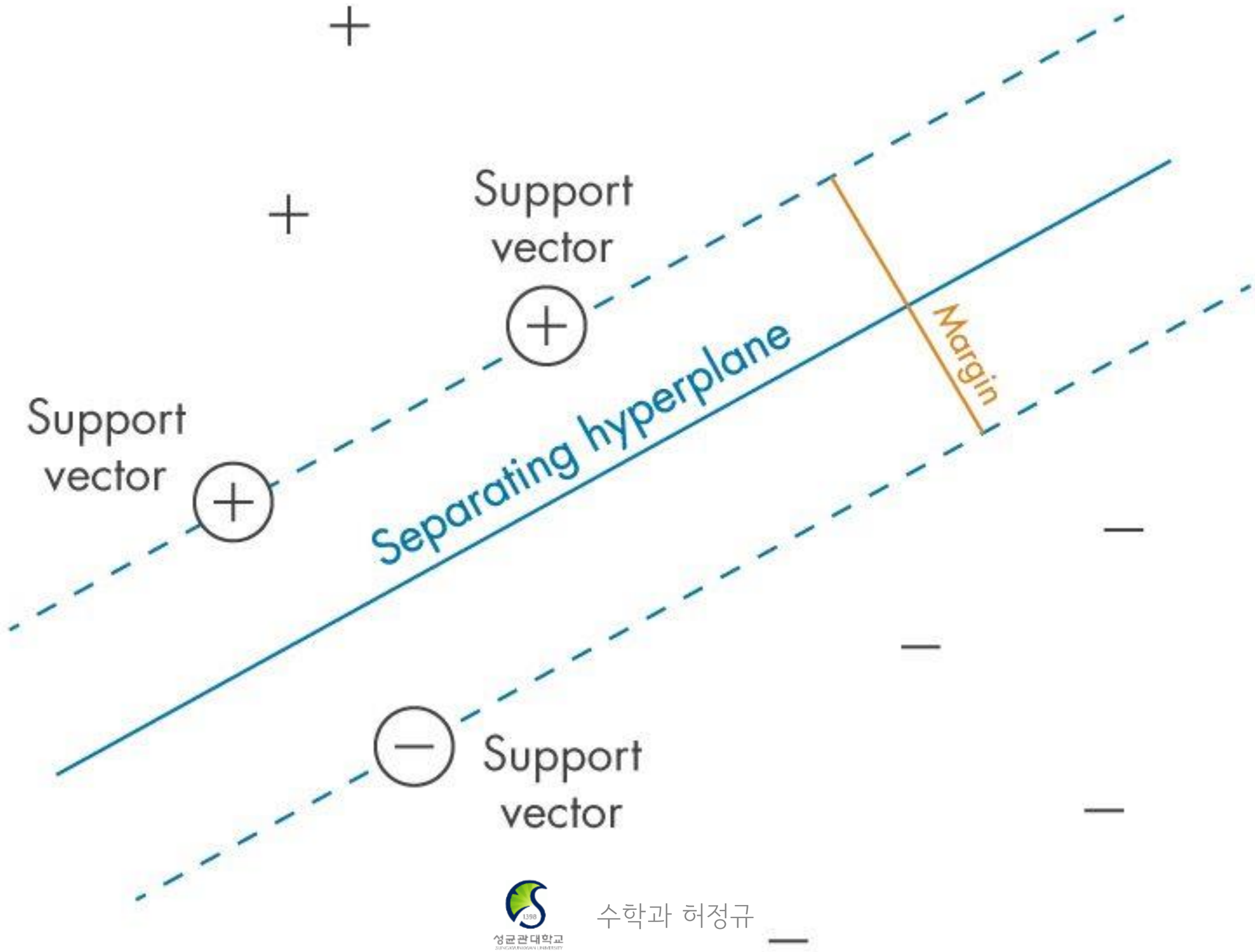
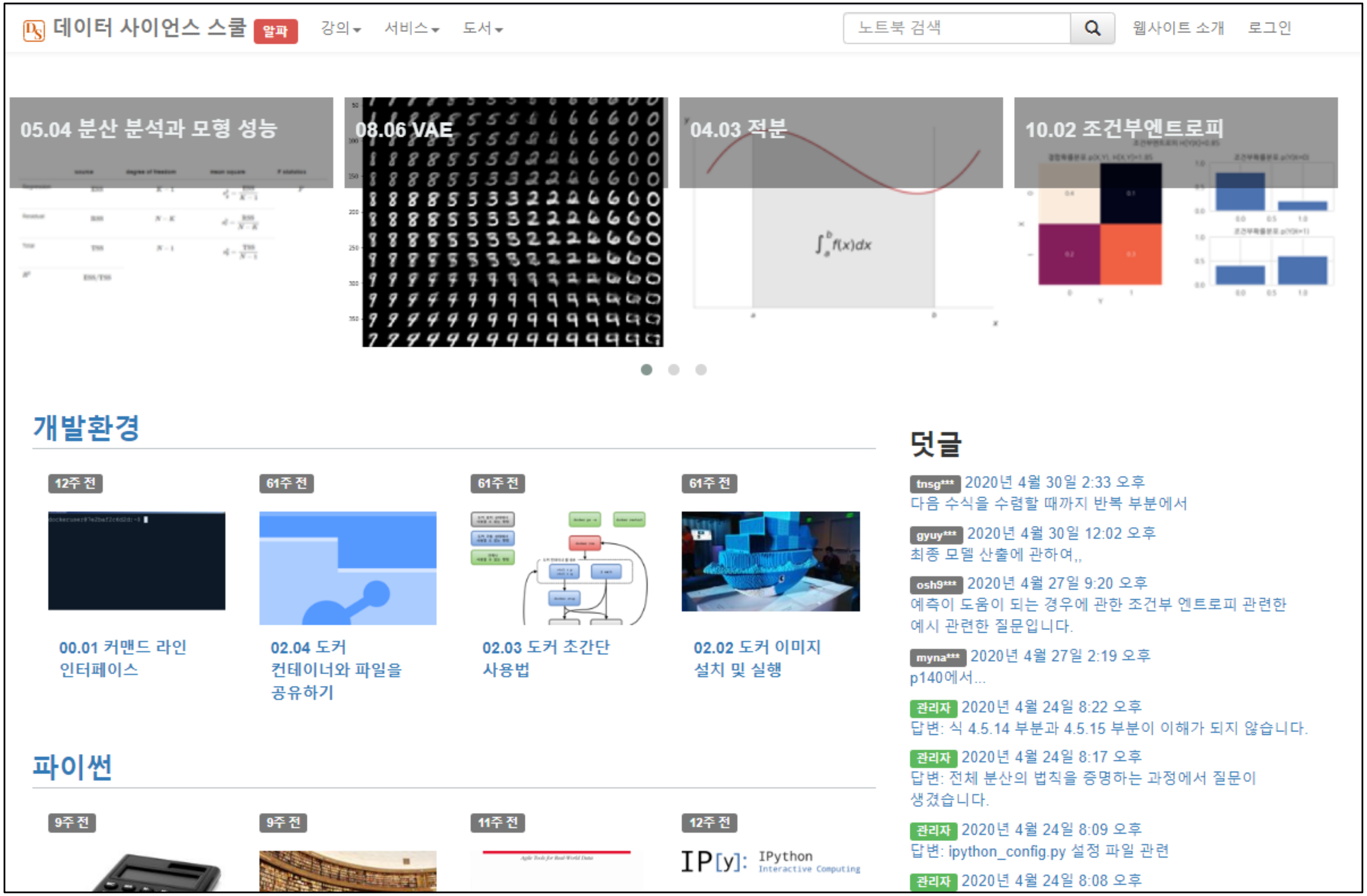
Support Vector Machine



성균관대학교

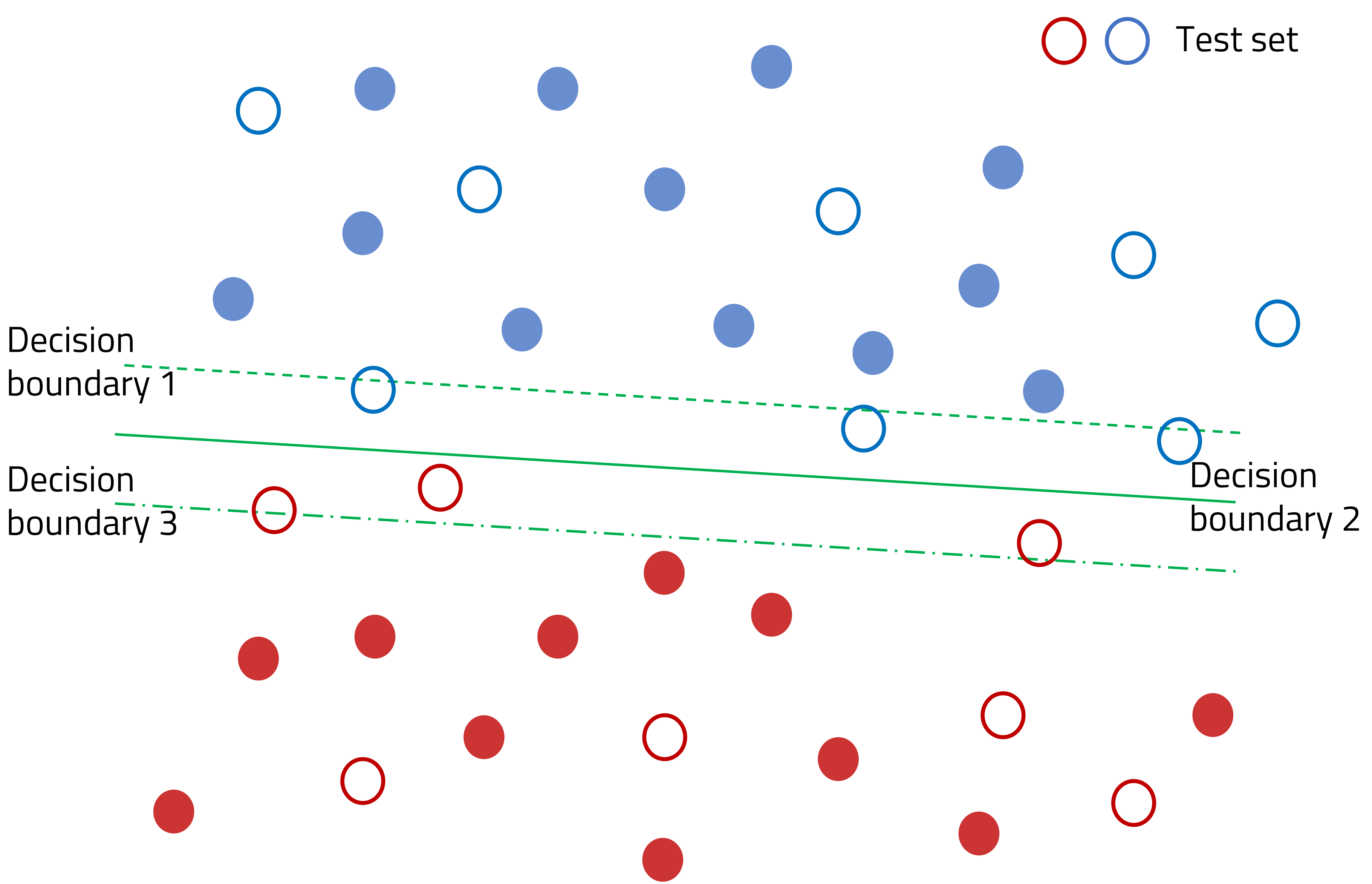
수학과 허정규

Data Science School (by Dr. Dohyeong Kim, datascienceschool.net)



수학과 허정규

What is the optimal decision boundary?



Data

$$\mathcal{D} = \{(x_j, y_j)\}_{j=1,\cdots,N}$$

$$y_i = \begin{cases} y_i^+ = +1 \\ y_i^- = -1 \end{cases}$$

$x_i^+ : x_i \in \{x_j\}$ such that $y_i = y_i^+$
$x_i^- : x_i \in \{x_j\}$ such that $y_i = y_i^-$

Model

$$f(x) = w^T x - w_0$$

such that

$$f(x_i^+) = w^T x_i^+ - w_0 > 0$$

$$f(x_i^-) = w^T x_i^- - w_0 < 0$$

Decision boundary

x such that $f(x) = 0$

Support vectors

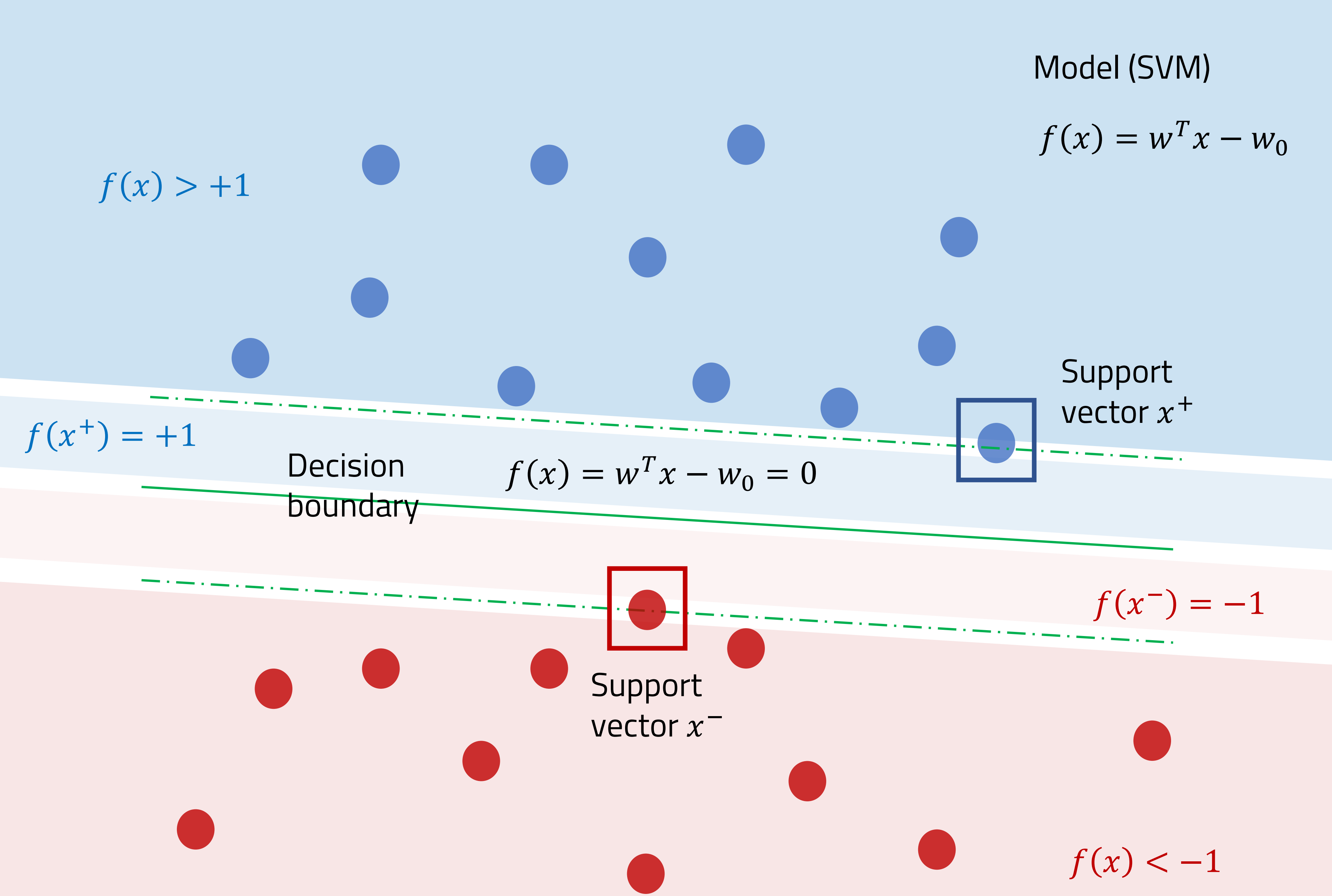
$x^+ : x \in \{x_j^+\}$ at which $f(x)$ is the smallest of $\{f(x_j^+)\}$

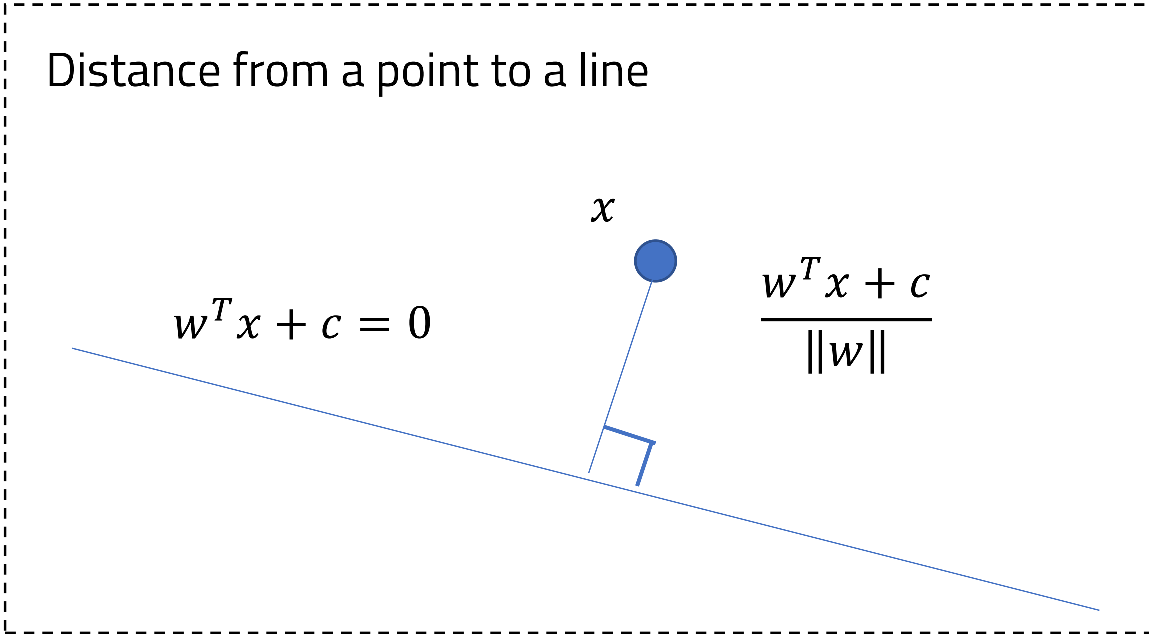
$x^- : x \in \{x_j^-\}$ at which $f(x)$ is the largest of $\{f(x_j^-)\}$

Constraints

$f(x^+) = 1$ $\qquad$ $f(x^-) = -1$

Support Vector Machine


Model (SVM)
$f(x) = w^T x - w_0$
$f(x) > +1$
$f(x^+) = +1$
Support vector x^+
Decision boundary
$f(x) = w^T x - w_0 = 0$
$f(x^-) = -1$
Support vector x^-
$f(x) < -1$

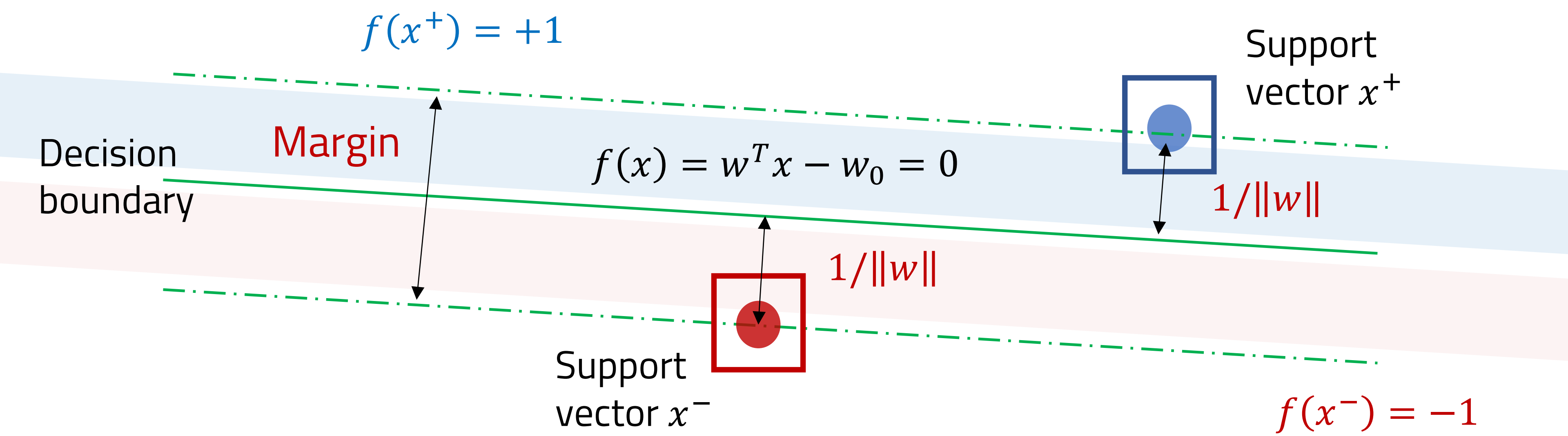


Margin maximization

$$w^* = \arg\max_w \frac{2}{\|w\|}$$

subject to

$$f(x^+) = +1$$
$$f(x^-) = -1$$



Margin maximization with inequality constraints

$$w^* = \arg\max_w \frac{2}{\|w\|}$$

subject to

$$f(x^+) = +1$$
$$f(x^-) = -1$$

$$w^* = \arg\min_w \|w\|^2$$

subject to

$$f(x_i^+) \geq +1$$
$$f(x_i^-) \leq -1$$

$$w^* = \arg\min_w \frac{1}{2} w^T w$$

subject to

$$f(x_i^+) y_i^+ \geq +1$$
$$f(x_i^-) y_i^- \geq +1$$

$$w^* = \arg\min_w \boxed{\frac{1}{2} w^T w}$$

subject to

$$\boxed{1 - f(x_i) y_i} \leq 0$$

Model (SVM)

$$f(x) = w^T x - w_0$$

$J(w)$

Primal Problem

computationally challenging

$g_i(w; x_i, y_i)$

Transformation of the primal problem to its dual problem

$$L(w,\lambda) = J(w) + \sum_{i=1}^{N} \lambda_i g(w; x_i, y_i)$$

$$= \frac{1}{2} w^T w + \sum_{i=1}^{N} \lambda_i \big(1 - y_i (w^T x_i - w_0)\big) \le J(w)$$

$\le$

$\min\limits_{w} J(w) \longleftrightarrow \max\limits_{\lambda} \min\limits_{w} L(w,\lambda)$

subject to $\quad$ subject to

$1 - f(x_i) y_i \le 0 \quad$ KKT condition

substitution

From KKT condition

(1) $\dfrac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \longrightarrow w = \sum_{i=1}^{N} \lambda_i y_i x_i$

(2) $\dfrac{\partial L}{\partial w_0} = \sum_{i=1}^{N} \lambda_i y_i = 0$

(3) $\lambda_i \dfrac{\partial L}{\partial \lambda_i} = 0 \longrightarrow \lambda_i \neq 0$ for $x = x^+, x^-$, $\lambda_i = 0$ otherwise

(4) $\lambda_i \ge 0$

Constructing a dual problem from the primal problem

$$\max_{\lambda} L(\lambda; x_i, y_i)$$

subject to

$$L(\lambda; x_i, y_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

(2) $\sum_{i=1}^{N} \lambda_i y_i = 0$ (4) $\lambda_i \geq 0$

Dual Problem

computationally easier

Matrix-vector form

$$L(\lambda; x_i, y_i) = c^T \lambda + \frac{1}{2}\lambda^T Q \lambda$$ where $c^T = [1,1,\cdots,1]$, $Q = -q^T q$,
$q = [q_1 \quad q_2 \quad \cdots \quad q_N]$, $q_i = y_i x_i$

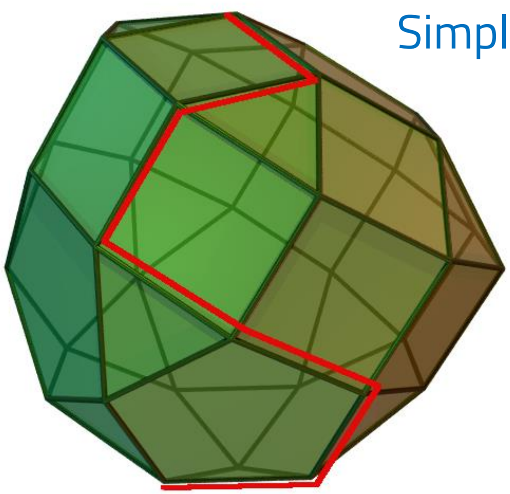
(2) $A\lambda = 0$ where $A = diag([y_1, y_2, \cdots, y_N])$ (4) $\lambda \geq 0$

Quadratic programming

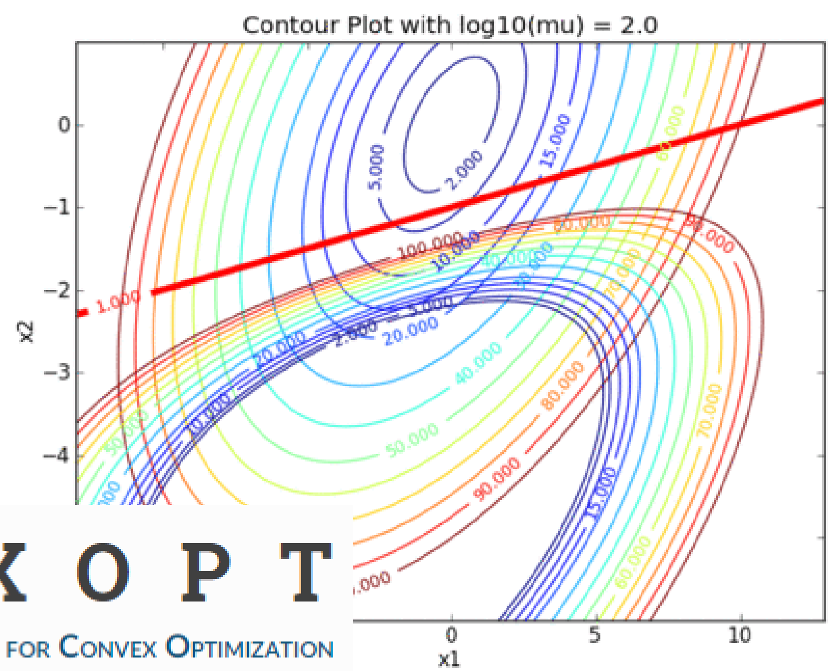
$$L(\{(x_i, y_i)\}, \lambda) = c^T\lambda + \frac{1}{2}\lambda^T Q\lambda \qquad \text{where } c^T = [1,1,\cdots,1],\ Q = -q^Tq,$$
$$q = [q_1 \quad q_2 \quad \cdots \quad q_N],\ q_i = y_i x_i$$

subject to

$A\lambda = 0$ where $A = diag([y_1, y_2, \cdots, y_N])$, $\lambda \geq 0$



Simplex method



CVXOPT

Python Software for Convex Optimization

https://cvxopt.org/userguide/index.html

Interior point method

Support Vector Machine

$$f(x) = w^T x - w_0$$

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

CVXOPT

PYTHON SOFTWARE FOR CONVEX OPTIMIZATION

$$f(x) = w^T x - w_0 = \lambda^+ (x^+)^T x - \lambda^- (x^-)^T x - w_0$$

$$w = \lambda^+ x^+ - \lambda^- x^-$$

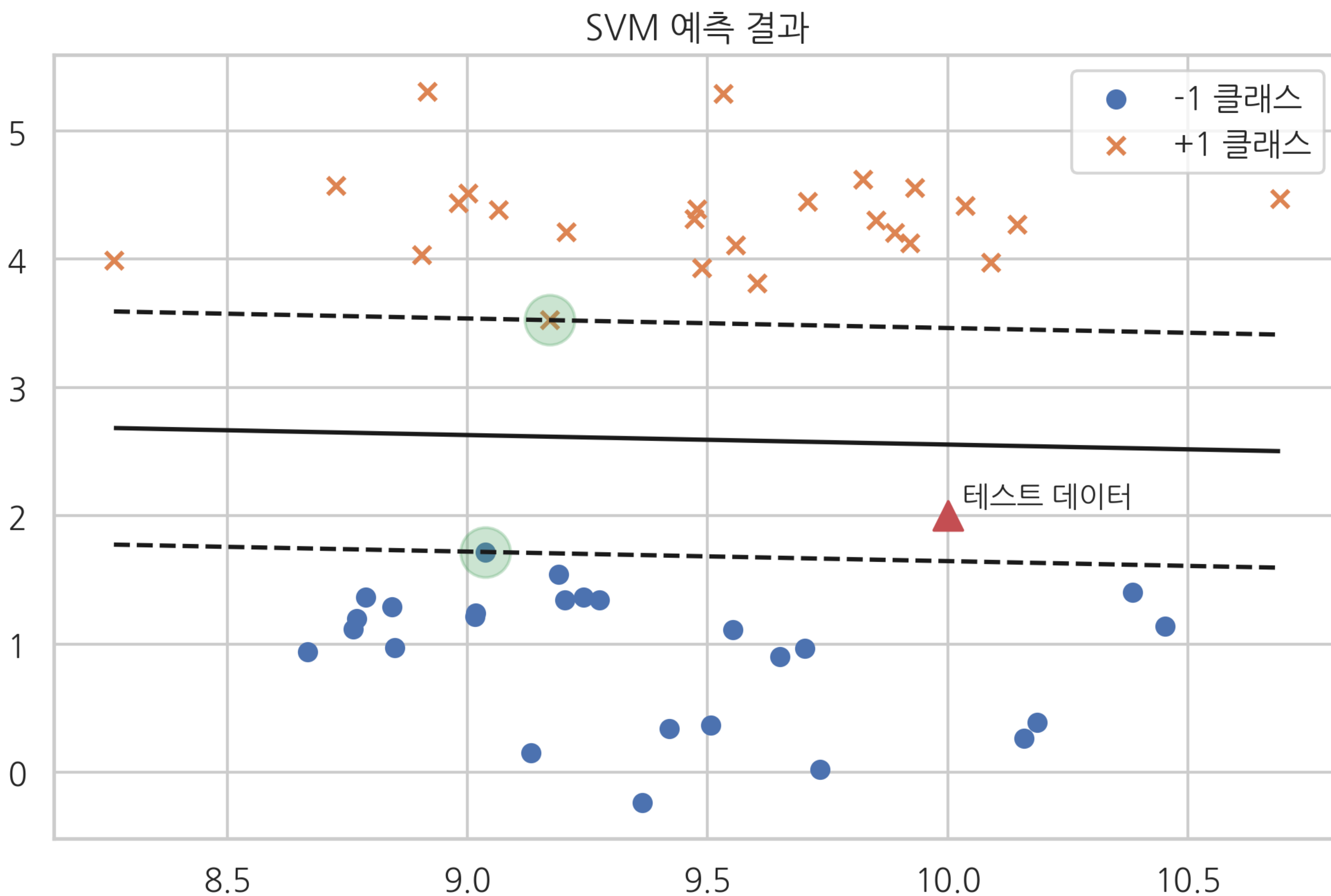
$$f(x^+) = w^T x^+ - w_0 = +1$$
$$f(x^-) = w^T x^- - w_0 = -1$$

$$w_0 = \frac{1}{2} w^T (x^+ + x^-)$$

$$f(x) = \lambda^+ \langle x, x^+ \rangle - \lambda^- \langle x, x^- \rangle - w_0$$

$\langle \cdot,\cdot \rangle$: inner product

similarity measure




```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=50, centers=2, cluster_std=0.5, random_state=4)
y = 2 * y - 1
```

Data Generation Code

Support Vector Machine (SVM) with Scikit-Learn

```
from sklearn.svm import SVC

model = SVC(kernel='linear', C=1e10).fit(X, y)
```

SVC : support vector classifier

```
model.support_
```

the index of the support vector for each class

```
y[model.support_]
```

the target values y^+, y^- of the support vectors

```
model.support_vectors_
```

the feature values x^+, x^- of the support vectors

```
x_new = [10, 2]
y_pred = model.decision_function(x_new)
```

the predicted value y_{pred} for a new feature value x_{new}

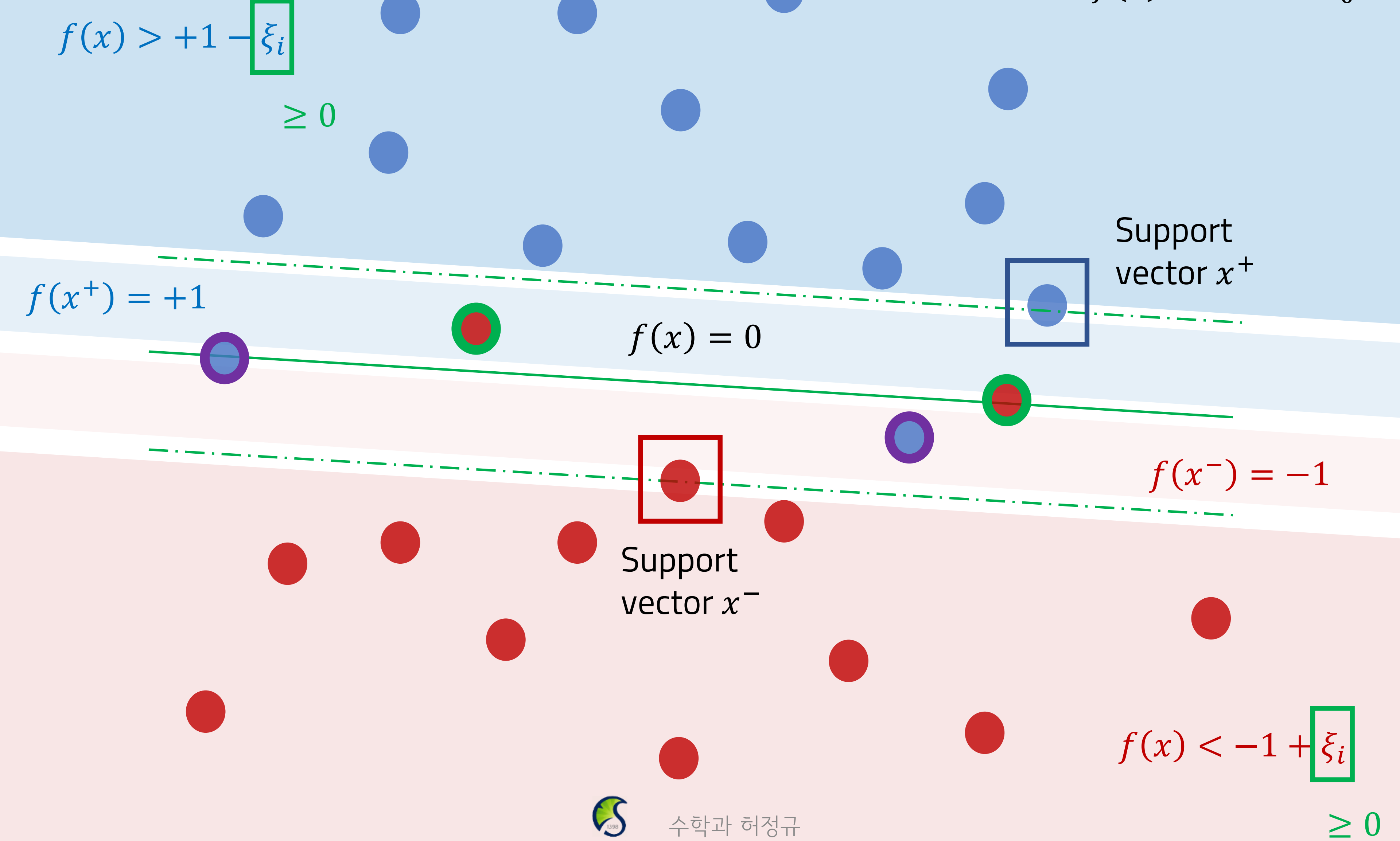
Exercise 1

Let's solve the iris problem with a support vector machine. Let's solve it by changing it to a binary classification problem using only the species 'versicolor', 'virginica'. Set the kernel type and the slack variable C as 'linear' and 1e10, respectively.

Not Linearly Separable Data and Slack Variable ξ_i

Model (SVM)

$$f(x) = w^T x - w_0$$



Lagrangian multiplier approach with inequality constraints

Model (SVM)

$$f(x) = w^T x - w_0$$

$$\min_{w} \left(J(w) + C \sum_{i=1}^{N} \xi_i \right)$$

slack variable

subject to

$$g_i(w; x_i, y_i) \leq \xi_i$$

$$\xi_i \geq 0$$

where

$$J(w) = \frac{1}{2} w^T w$$

$$g_i(w; x_i, y_i) = 1 - f(x_i) y_i$$

$$L(\lambda; x_i, y_i) = J(w) + \sum_{i=1}^{N} \lambda_i (g(w; x_i, y_i) - \xi_i) - \sum_{i=1}^{N} \mu_i \xi_i + C \sum_{i=1}^{N} \xi_i$$

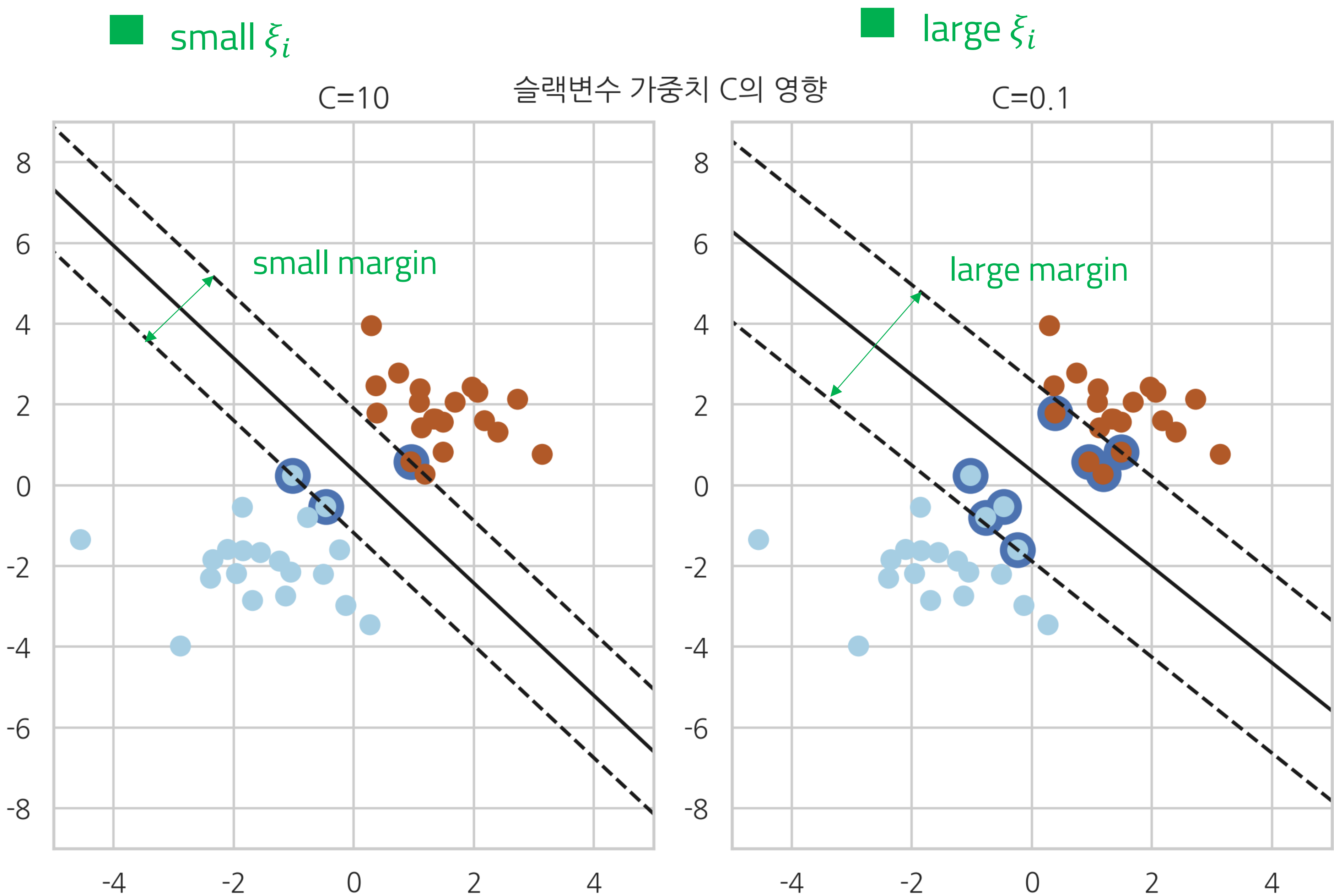
(1) $$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0$$

(2) $$\frac{\partial L}{\partial w_0} = \sum_{i=1}^{N} \lambda_i y_i = 0$$

(3) $$\lambda_i \frac{\partial L}{\partial \lambda_i} = 0 \ , \quad \mu_i \frac{\partial L}{\partial \mu_i} = 0$$

(4) $$\lambda_i \geq 0 \ , \quad \mu_i \geq 0$$

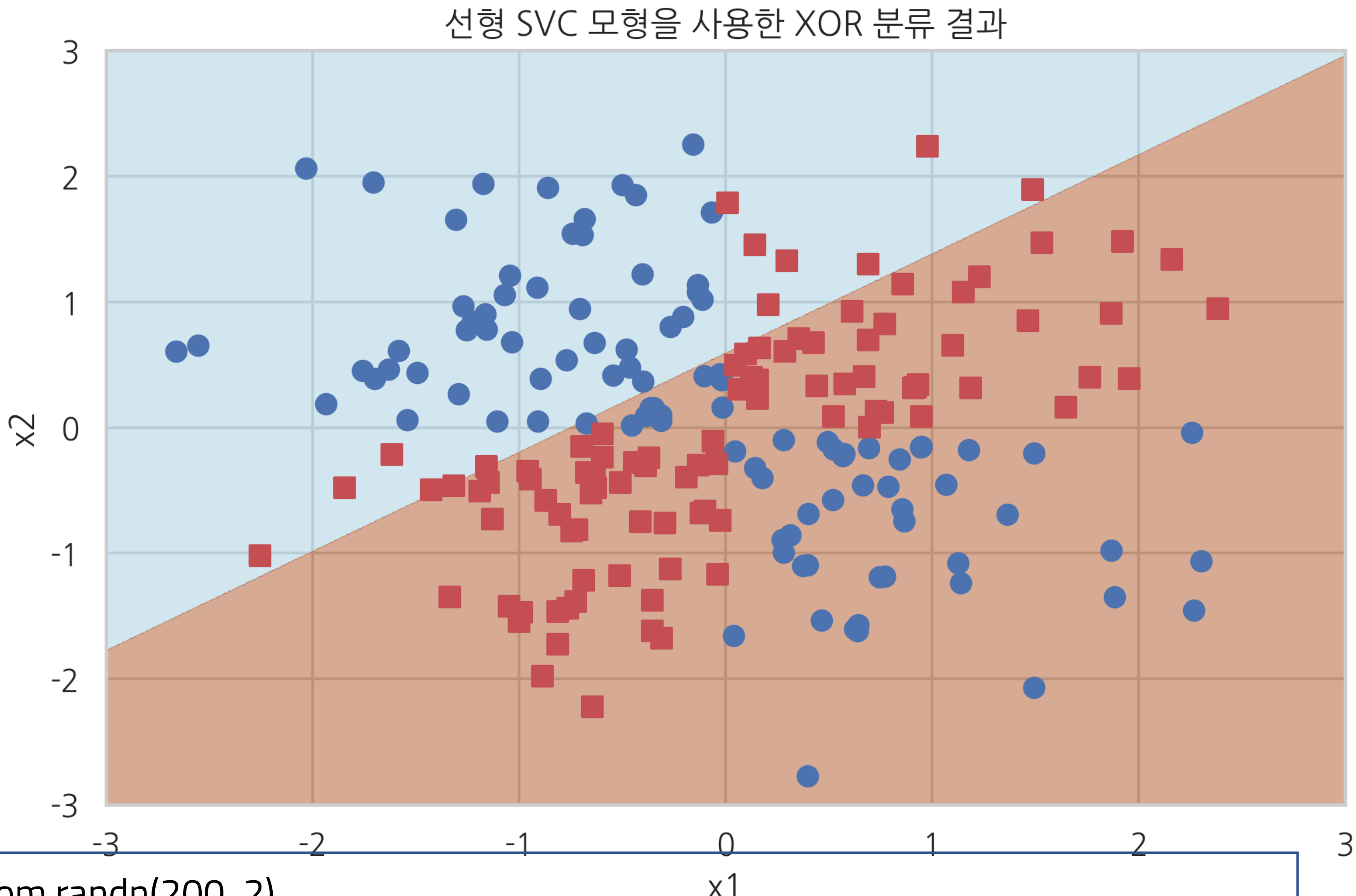
슬랙변수 가중치 C의 영향



Exercise 2

Let's solve the iris problem with a support vector machine. Let's solve it by changing it to a binary classification problem using only the species 'versicolor', 'virginica'. Fix the kernel type as 'linear', but find the optimal slack variable C while changing the value.

Linear SVM cannot solve the XOR problem



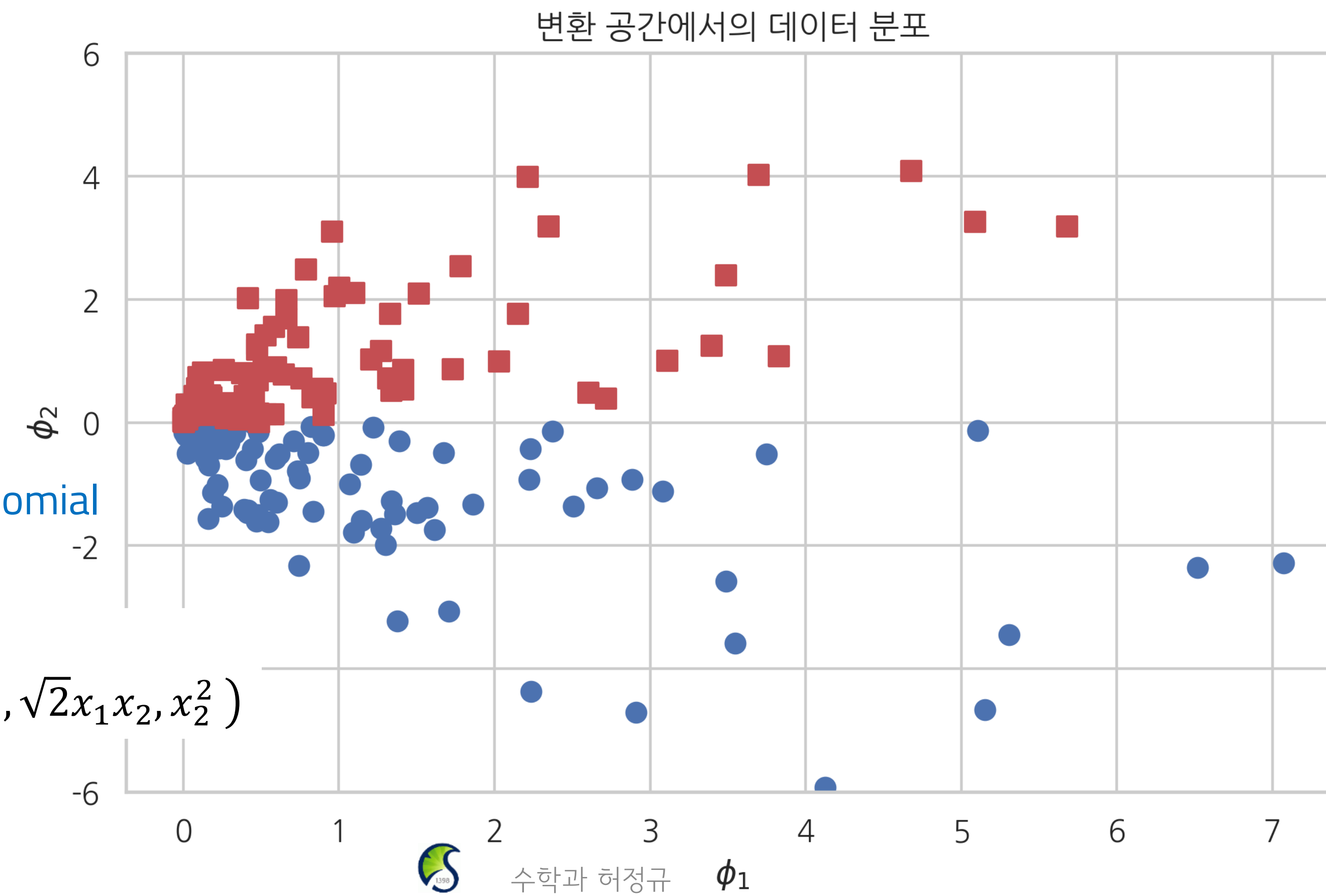

```
X_xor = np.random.randn(200, 2)
y_xor = np.logical_xor(X_xor[:, 0] > 0, X_xor[:, 1] > 0)
y_xor = np.where(y_xor, 1, 0)
```

Data Generation Code

Transform of data using a feature map

$\phi(\cdot): R^D \to R^M$ feature map

s.t. $x = (x_1, x_2, \cdots, x_D) \to \phi(x) = \left(\phi_1(x), \phi_2(x), \cdots, \phi_M(x)\right)$



quadratic polynomial feature map

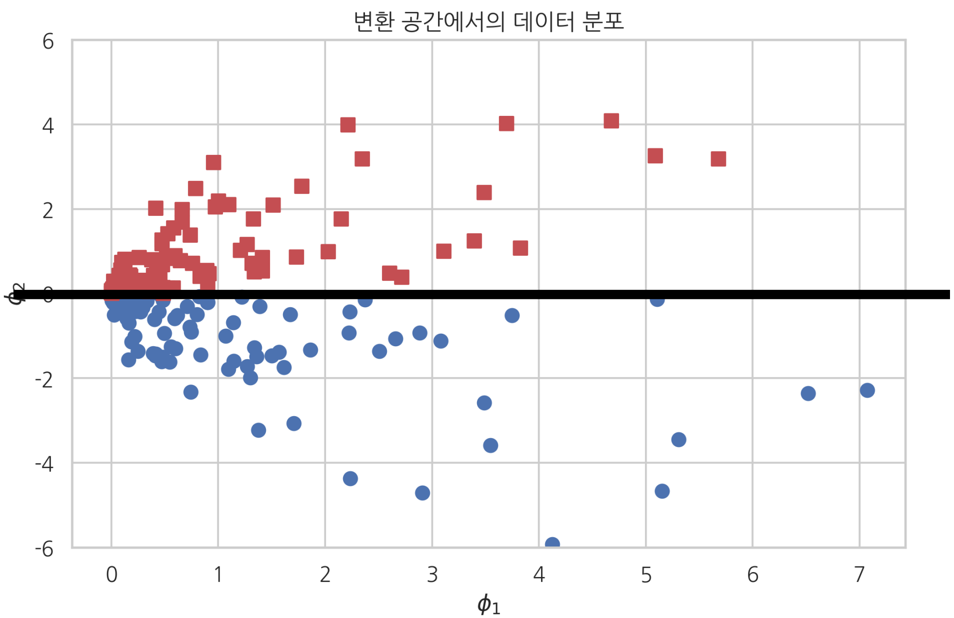
$x = (x_1, x_2)$

$\to \phi(x) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$

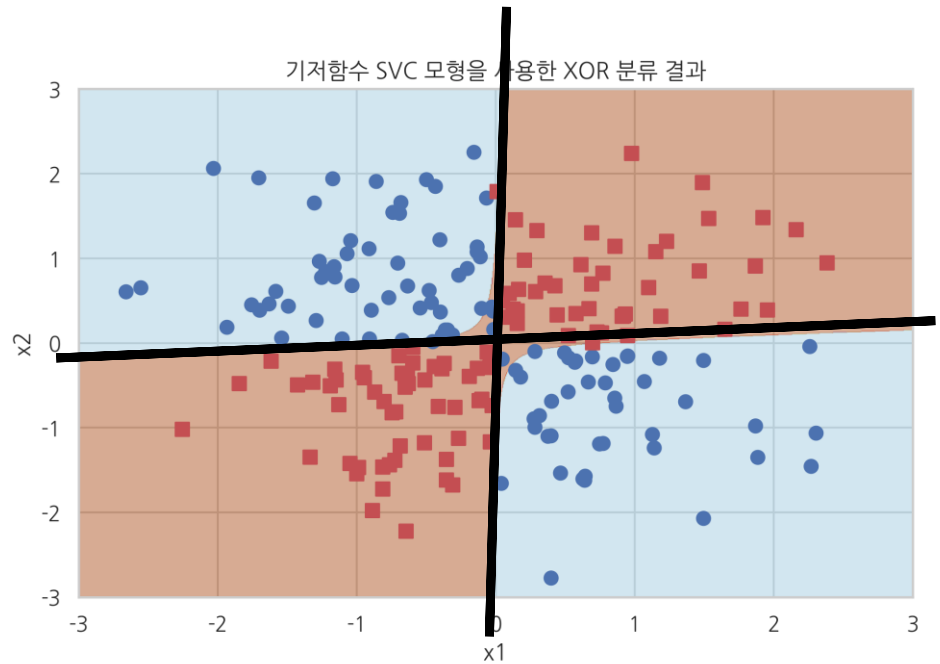
"Linear SVM + Quadratic polynomial kernel" solve the XOR problem

quadratic polynomial feature map

$$x = (x_1, x_2)$$
$$\rightarrow \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



kernel feature space



original feature space

Support Vector Machine (SVM) with Scikit-Learn

```
from sklearn.preprocessing import FunctionTransformer

def kernel(X):
  return np.vstack([X[:, 0]**2, np.sqrt(2)*X[:, 0]*X[:, 1], X[:, 1]**2]).T

X_xor2 = FunctionTransformer(kernel).fit_transform(X_xor)
```

quadratic polynomial feature map

$$x = (x_1, x_2) \to \phi(x) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

```
from sklearn.pipeline import Pipeline

basismodel = Pipeline([("kernel", FunctionTransformer(kernel)),
                       ("svc", SVC(kernel="linear"))]).fit(X_xor, y_xor)
```

pipeline construction
using the polynomial feature map and a support vector machine

Kernel Trick

* Model (SVM)

$$f(x) = w^T x - w_0$$

* Feature map

$$\phi(\cdot):\ R^D \rightarrow R^M \text{ s.t. } x \rightarrow \phi(x)$$

$$\min_w J(w)$$

subject to

$$g_i(w; x_i, y_i) \leq 0$$

where

$$J(w) = \frac{1}{2} w^T w$$

$$g_i(w; x_i, y_i) = 1 - f\big(\phi(x_i)\big) y_i$$

$$\max_\lambda L(\lambda; \phi(x_i), y_i)$$

subject to

$$\sum_{i=1}^{N} \lambda_i y_i = 0\ ,\ \ \lambda_i \geq 0$$

where

$$L(\lambda; \phi_{x_i}, y_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \lambda_i \lambda_j y_i y_j \phi(x_i)^T \phi(x_j)$$

kernel

$$k(u, v) = \phi(u)^T \phi(v)$$

Kernel Support Vector Machine

$$f_\phi(x) = w^T\phi(x) - w_0$$

$$w = \sum_{i=1}^{N} \lambda_i y_i \phi(x_i)$$

CVXOPT

Python Software for Convex Optimization

$$f_\phi(x) = w^T\phi(x) - w_0 = \lambda^+\left(\phi(x^+)\right)^T\phi(x) - \lambda^-\left(\phi(x^-)\right)^T\phi(x) - w_0$$

$$w = \lambda^+\phi(x^+) - \lambda^-\phi(x^-)$$

$$f_\phi(x^+) = w^T\phi(x^+) - w_0 = +1$$
$$f_\phi(x^-) = w^T\phi(x^-) - w_0 = -1$$

$$w_0 = \frac{1}{2}w^T\left(\phi(x^+) + \phi(x^-)\right)$$

$$f_\phi(x) = \lambda^+\langle x, x^+\rangle_\phi - \lambda^-\langle x, x^-\rangle_\phi - w_0$$

$\langle\cdot,\cdot\rangle_\phi$: inner product kernel induced by ϕ which is defined as $\langle u, v\rangle_\phi = \phi(u)^T\phi(v)$

similarity measure

Commonly Used Kernels

- Linear Kernel

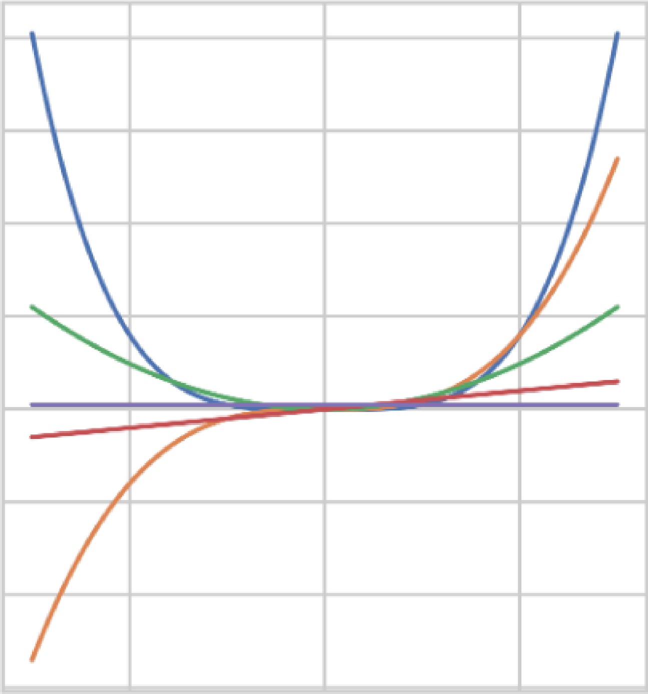
$$k(u,v) = u^T v$$

- Polynomial Kernel

$$k(u,v) = (\gamma(u^T v) + \theta)^d$$

components ϕ_j of the feature map

$$\phi = (\phi_1, \cdots, \phi_M)$$



$$k(u,v) = \phi(u)^T \phi(v)$$

- Gaussian Kernel (or called Radial Basis Function)

$$k(u,v) = \exp(-\gamma \| u - v \|^2)$$

- Sigmoid Kernel

$$k(u,v) = \tanh\big((\gamma(u^T v) + \theta)\big)$$

Kernel Support Vector Machine

* polynomial kernel

```
polysvc = SVC(kernel="poly", degree=2, gamma=1, coef0=0).fit(X_xor, y_xor)
```

* Gaussian kernel

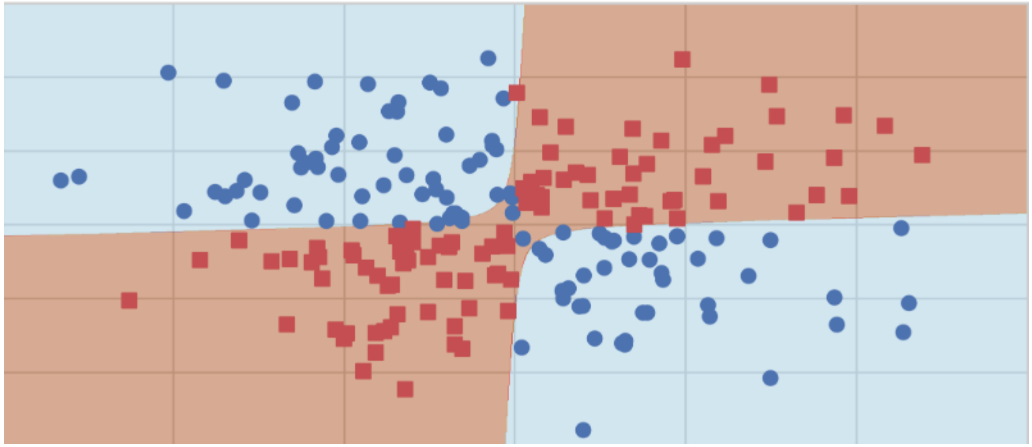
```
rbfsvc = SVC(kernel="rbf").fit(X_xor, y_xor)
```

* sigmoid kernel

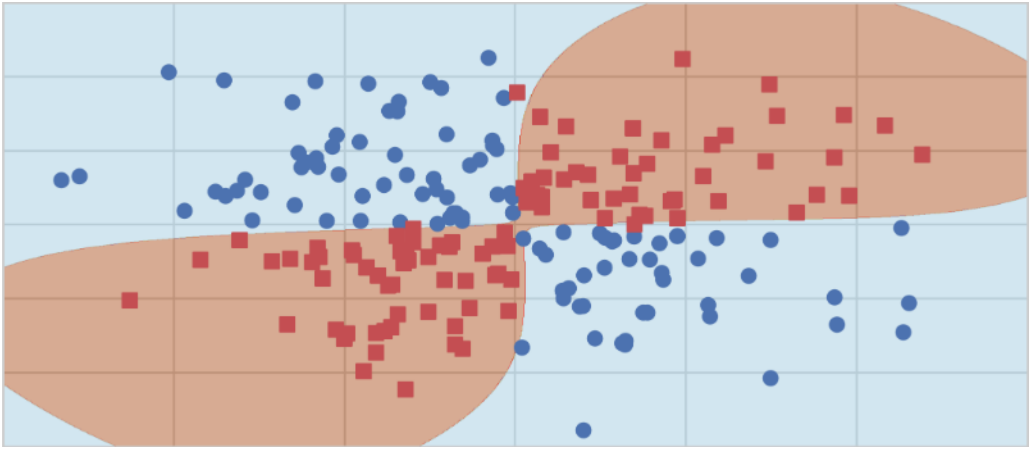
```
sigmoidsvc = SVC(kernel="sigmoid", gamma=2, coef0=2).fit(X_xor, y_xor)
```

- `kernel = "linear"`: 선형 SVM. $k(x_1, x_2) = x_1^T x_2$
- `kernel = "poly"`: 다항 커널. $k(x_1, x_2) = (\gamma (x_1^T x_2) + \theta)^d$
    - `gamma`: γ
    - `coef0`: θ
    - `degree`: d
- `kernel = "rbf"` 또는 `kernel = None`: RBF 커널. $k(x_1, x_2) = \exp\left(-\gamma ||x_1 - x_2||^2\right)$
    - `gamma`: γ
- `kernel = "sigmoid"`: 시그모이드 커널. $k(x_1, x_2) = \tanh(\gamma (x_1^T x_2) + \theta)$
    - `gamma`: γ
    - `coef0`: θ

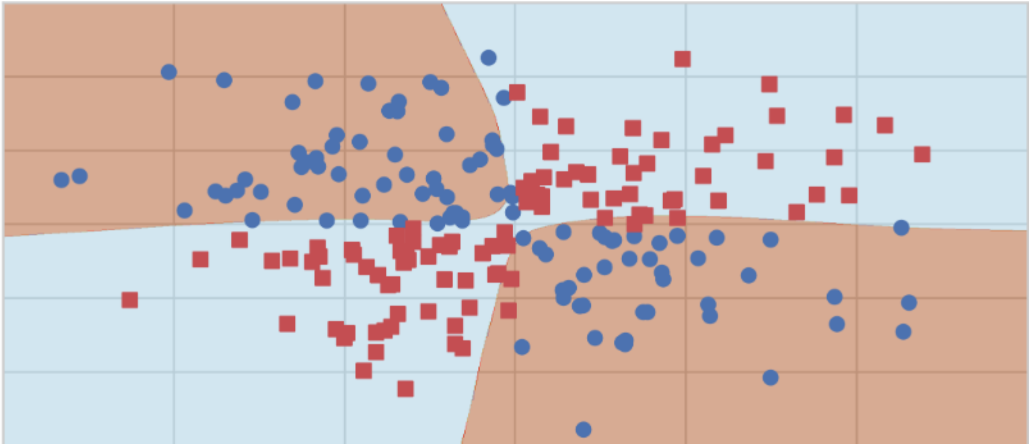
Polynomial
kernel



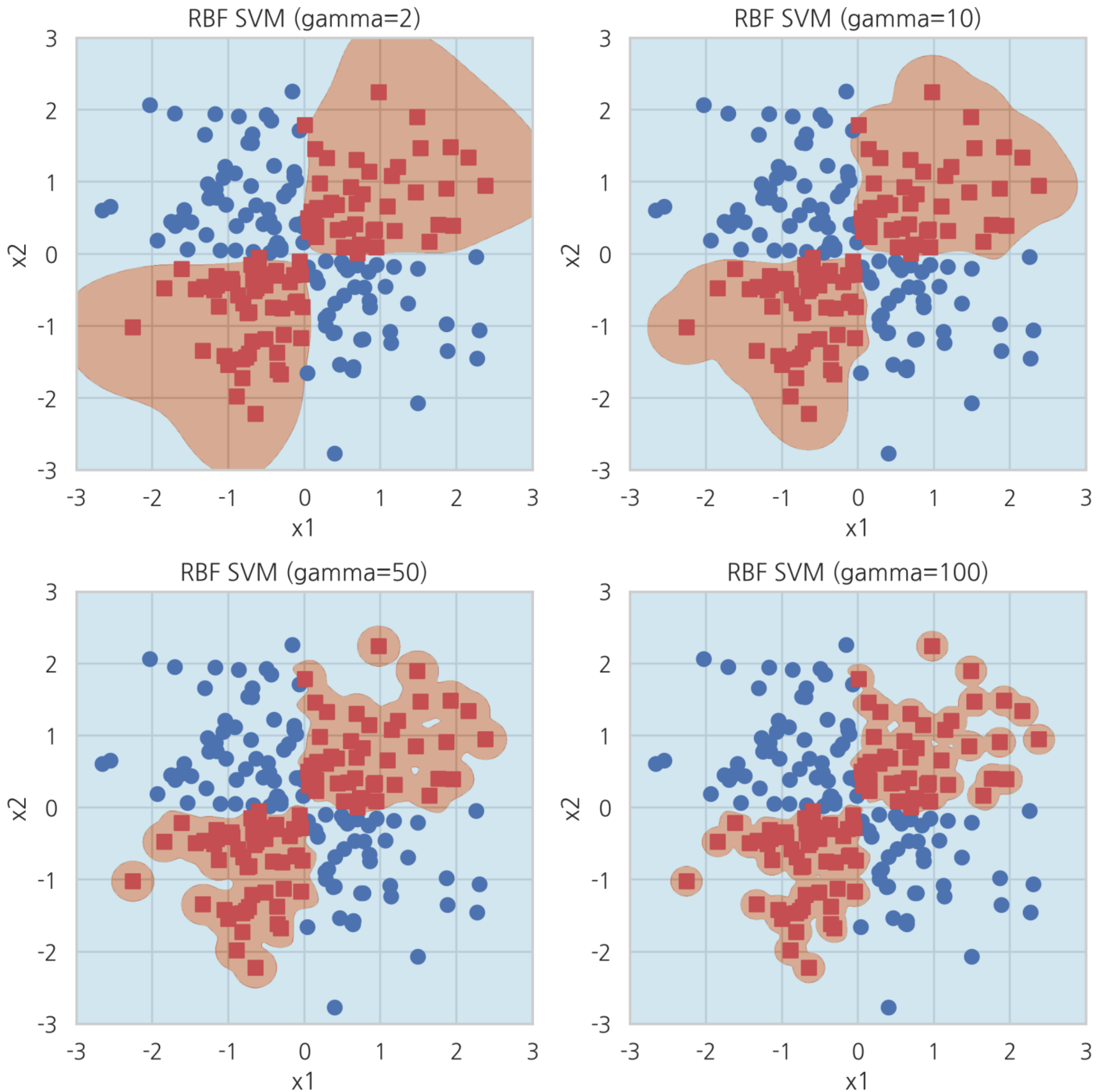
Gaussian
kernel



Sigmoid
kernel



Kernel Parameter Effect



Exercise 3

Let's solve the iris problem with a support vector machine. Let's solve it by changing it to a binary classification problem using only the species 'versicolor', 'virginica'. Find the optimal kernel type and slack variable C while changing the values.