분류 알고리즘(지도학습)

SVM(SUPPORT VECTOR MACHINE)

SVM

- 분류 문제를 해결하기 위해 2개의 다른 범주에 속한 데이터 그룹 사이에 좋은 결정 경계(decision boundary)를 찾는다.

- Perceptron의 확장
  - 분류오차($y - \hat{y}$)의 최소화

- SVM의 학습
  - Margine(초평면=결정경계)의 최대화
  - 서로 다른 결과들을 가장 잘 분할하는 특징들의 선형 결합을 찾는다.

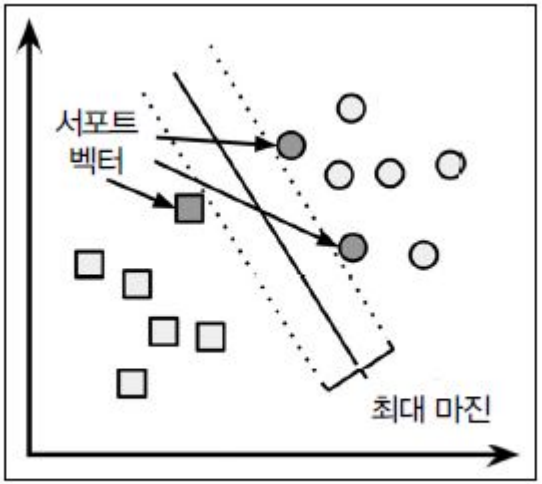
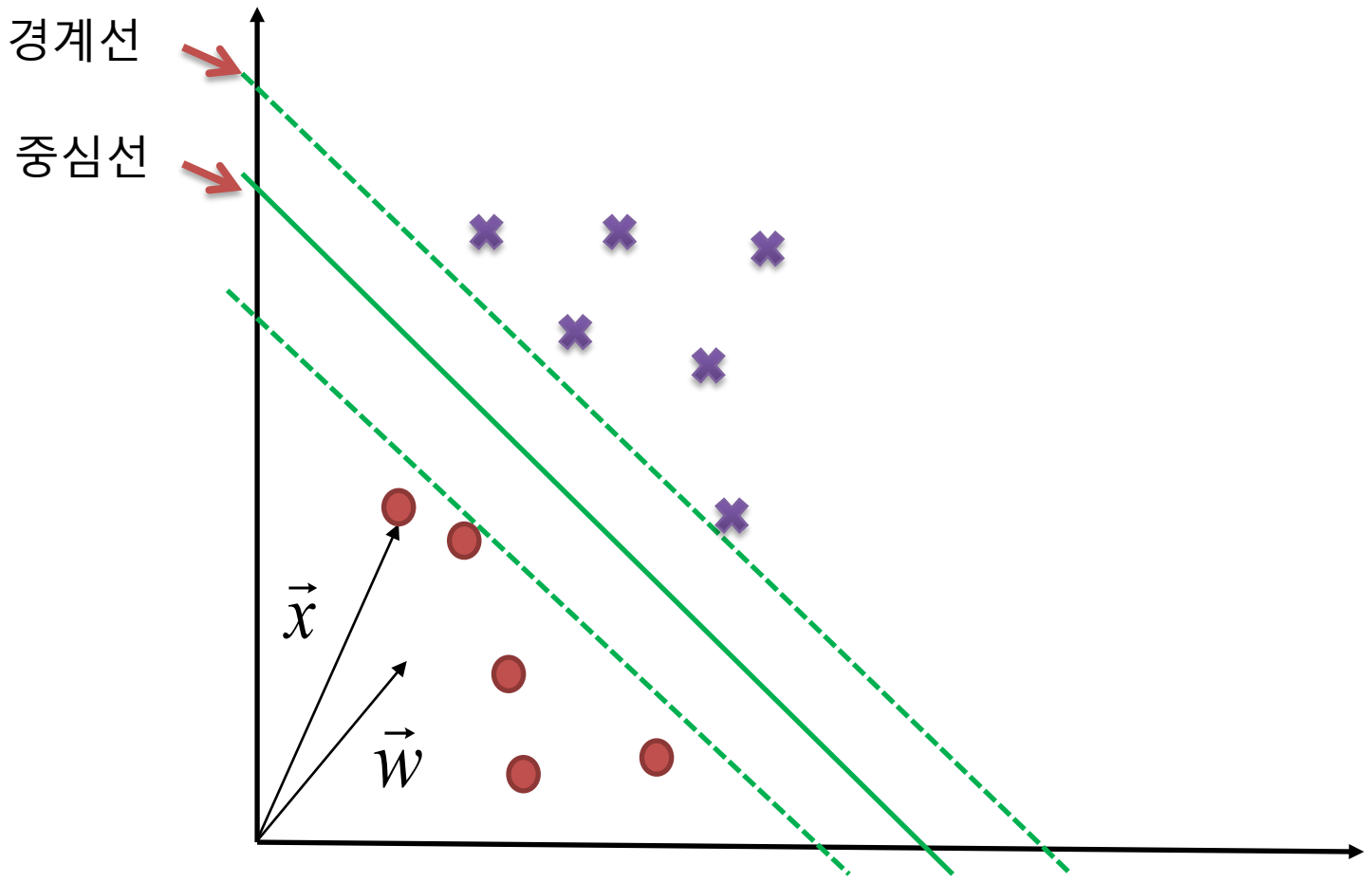
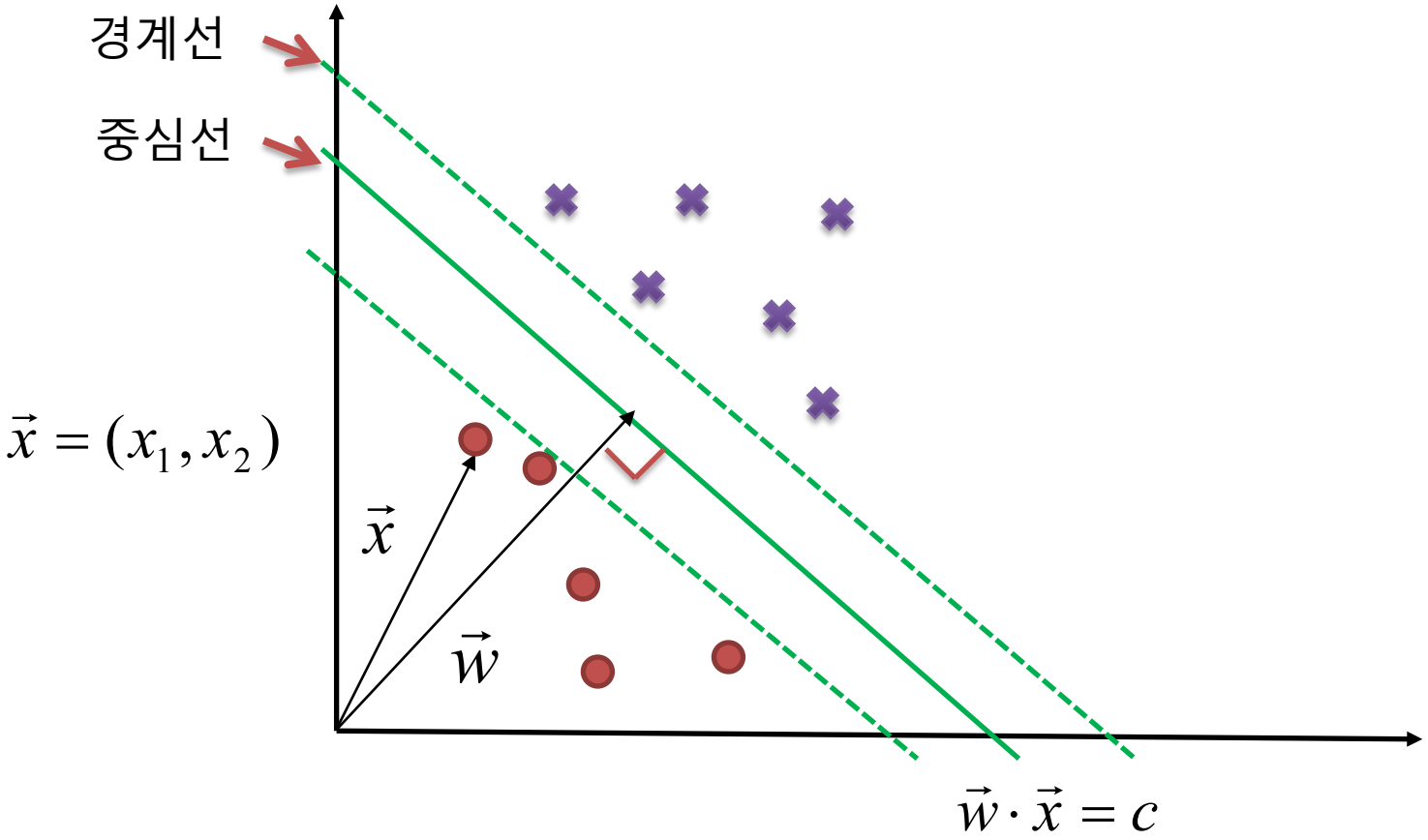


그림 7.17: 서포트 벡터에 의해 정의된 최대 마진 초평면


경계선
중심선
$\vec{x}$
$\vec{w}$



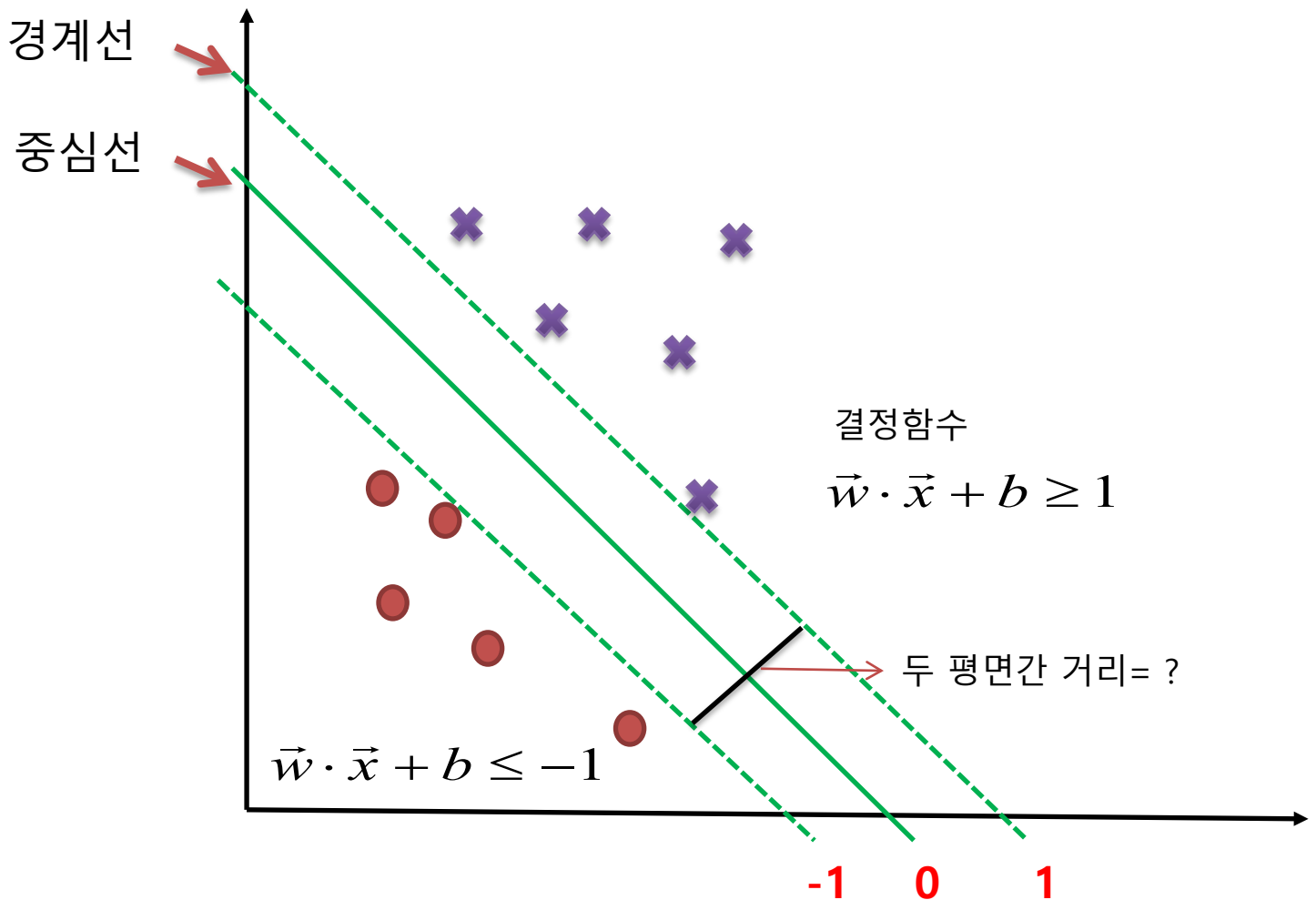
$\vec{w} = (w_1, w_2)$

$\vec{w} \cdot \vec{x} = w_1 x_1 + w_2 x_2$

중심선에 수직인 벡터

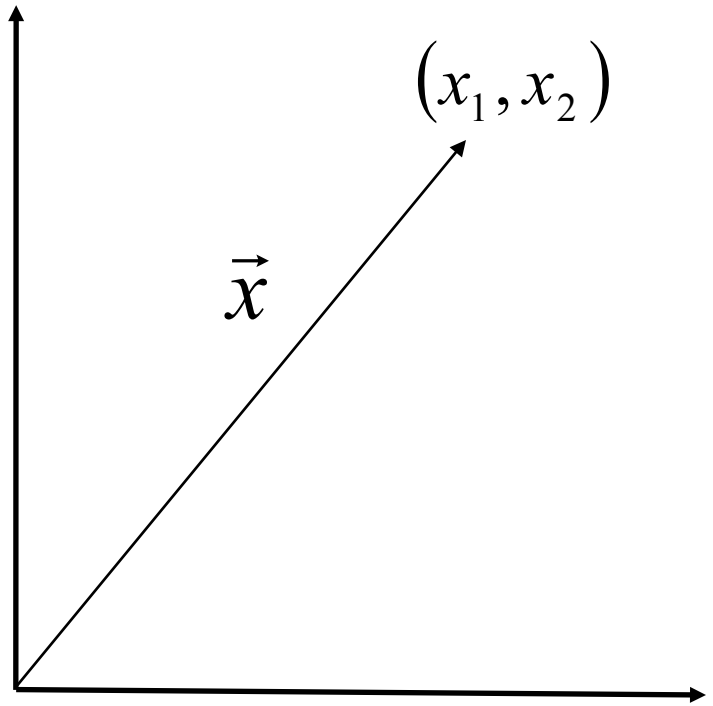
벡터 : 방향과 크기를 모두 포함하는 기하학적 대상

결정 조건(Decision rule)



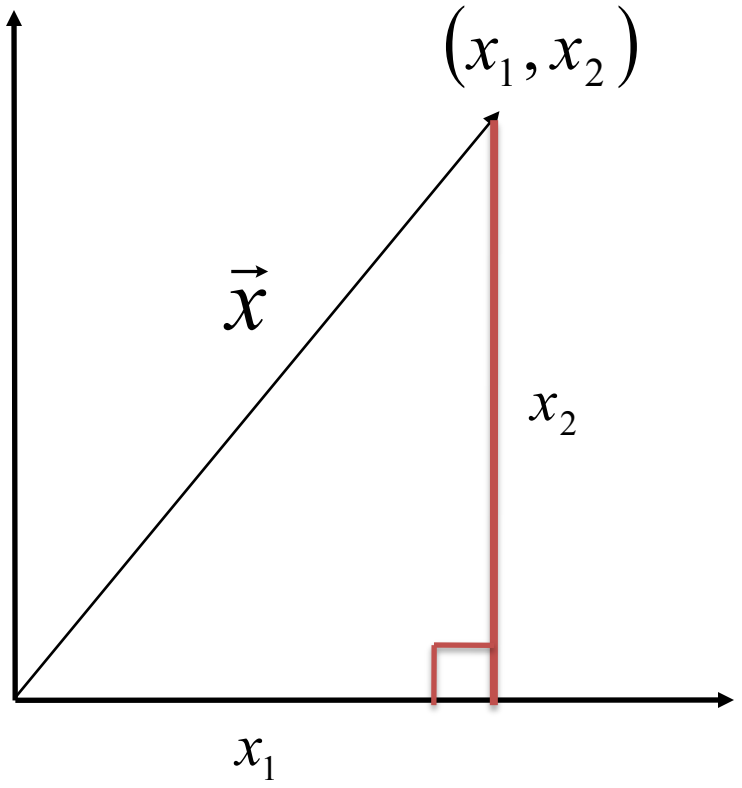
선형 SVM 분류기를 훈련한다는 것은 마진 오류를 하나도 발생하지 않거나(하드 마진)
제한적인 마진 오류를 가지면서(소프트 마진) 가능한 한 마진을 크게 하는 w와 b를 찾는 것

벡터의 길이 $\| \vec{x} \|$



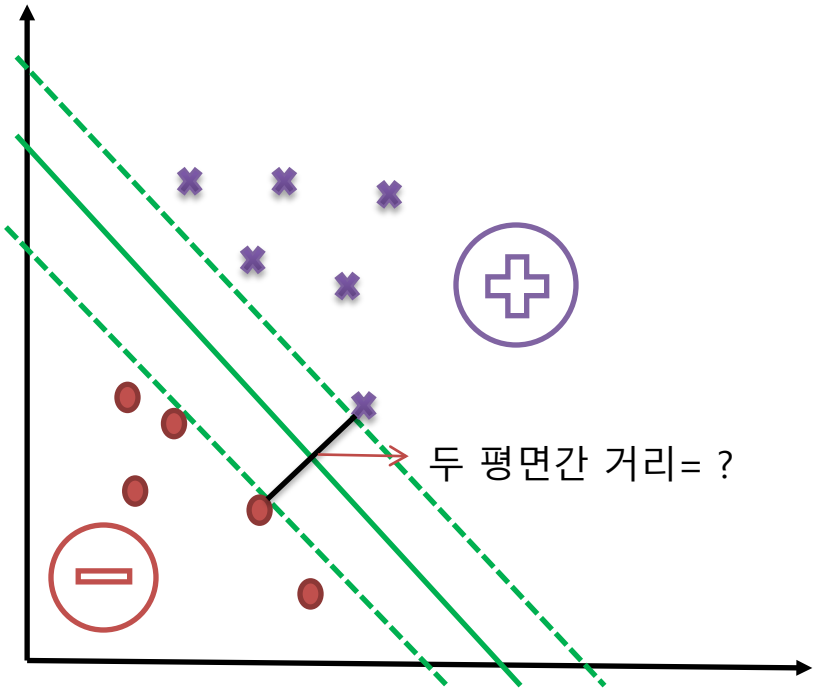
$$\|\vec{x}\| = \sqrt{x_1{}^2 + x_2{}^2}$$

피타고라스의 정리



$$\vec{x}^2 = x_1{}^2 + x_2{}^2$$

두 평면 간의 거리(마진)



서포트 벡트 간의 거리

$$\frac{2}{\|\vec{w}\|}$$

서포트 벡트 간의 거리 최대화

$$\min(\frac{1}{2}\|\vec{w}\|^2)$$

두 평면 간의 거리가 멀수록 좋은 분류 모형
가중치 벡터 w가 작을수록 마진은 커진다.

$$y_i = \begin{cases} 1 & \text{x가 } \oplus \text{ 영역에 있을 때} \\ -1 & \text{x가 } \ominus \text{ 영역에 있을 때} \end{cases}$$

$$\vec{w}\cdot\vec{x}_+ + b \geq 1 \quad \Longrightarrow \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

$$\vec{w}\cdot\vec{x}_- + b \leq -1 \quad \Longrightarrow \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

$$y_i(\vec{w}\cdot\vec{x}_i + b) - 1 \geq 0$$

+ 경계선에 있는 데이터 $\quad 1\times(\vec{w}\cdot\vec{x}_+ + b) - 1 = 0 \quad \vec{w}\cdot\vec{x}_+ = 1 - b$

- 경계선에 있는 데이터 $\quad (-1)\times(\vec{w}\cdot\vec{x}_- + b) - 1 = 0 \quad -\vec{w}\cdot\vec{x}_- = 1 + b$

$$(\vec{x}_+ - \vec{x}_-)\frac{\vec{w}}{\|\vec{w}\|} = \frac{\vec{w}\cdot\vec{x}_+ - \vec{w}\cdot\vec{x}_-}{\|\vec{w}\|} = \frac{1-b+1+b}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

$$\max\left(\frac{2}{\|\vec{w}\|}\right) \Longrightarrow \max\left(\frac{1}{\|\vec{w}\|}\right) \Longrightarrow \min(\|\vec{w}\|) \Longrightarrow \min\left(\frac{1}{2}\|\vec{w}\|^2\right)$$

서포트 벡트 간의 거리 최대화 : 벡터 w 자기 자신과의 내적 값의 최소화

$$\min(\frac{1}{2}(\vec{w}\cdot\vec{w}^{T}))$$

제약이 있는 최적화(constrained optimization)

제약조건(경계선) : $y_i(\vec{w}\cdot\vec{x}+b)-1=0$

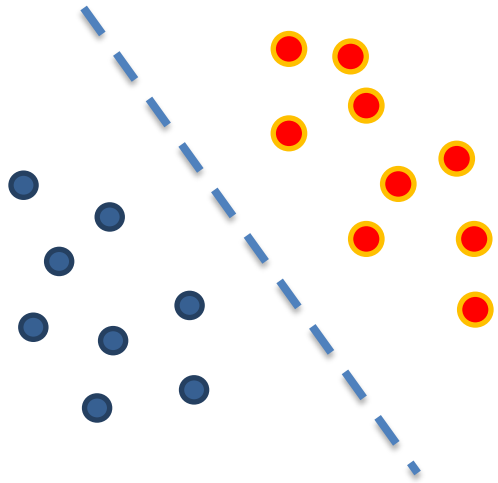
목적함수 : $\min(\frac{1}{2}\|\vec{w}\|^2)$

라그랑지안(Lagrangian) $L=\frac{1}{2}\|\vec{w}\|^2-\sum_{i=1}^{n}\alpha_i[y_i(\vec{w}\cdot\vec{x}+b)-1]$

최적값(optimal value) $\vec{w}=\sum_{i=1}^{n}\alpha_i y_i\vec{x}_i$

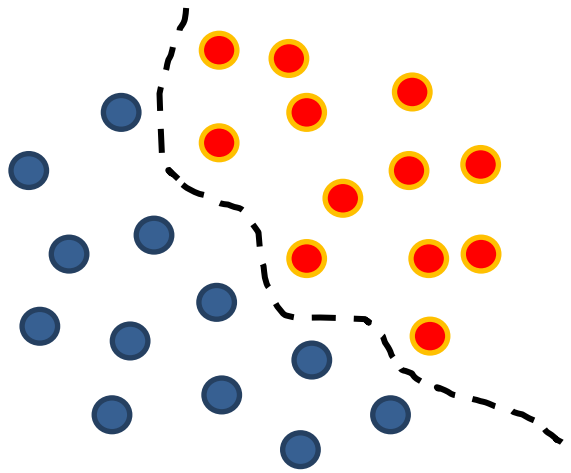
결정조건 $\vec{w}\cdot\vec{x}_i+b>0$

선형 서포트 벡터 머신



최대 마진

비선형 서포트 벡터 머신



최소 비용, C(cost) : 경계선의 복잡성을 컨트롤

kernel 함수 : 시그모이드 커널, 가우시안 RBF 커널

degree(다항식의 차수),

Scale(다항식의 파라미터를 스케일링),

Cost(비용함수)

예제 : 와인 품질 분류

- 분석 목적
  - 와인 class 분류를 통한 선형 SVM과 비선형 SVM 비교

- 데이터
  - wine.csv 파일
  - 178개 샘플, 13개 특성,
  - Class 목표변수(class = 1, 2, 3)

- 분석 알고리즘
  - Caret 패키지 활용
  - Kernlab 패키지 연동

R code

```
# caret 패키지 설치
install.packages("caret", dependencies = TRUE) # R은 오픈소스, 의존성
library(caret)

# 데이터 불러오기
wine_svm <- read.csv("wine.csv", header = TRUE)
wine_svm$Class <- as.factor(wine_svm$Class)
str(wine_svm)

# 훈련데이터(70%), 테스트데이터(30%) 분할
set.seed(2020)
svm_total <- sample(nrow(wine_svm), nrow(wine_svm)*0.7)
wine_svm_train <- wine_svm[svm_total,]
wine_svm_test <- wine_svm[-svm_total,]

train_x <- wine_svm_train[,1:13]
train_y <- wine_svm_train[,14]
test_x <- wine_svm_test[,1:13]
test_y <- wine_svm_test[,14]
```

```r
# 서포트 벡터 머신 구현

# 데이터 훈련과정의 파라미터 설정

ctrl <- trainControl(method = "repeatedcv", repeats = 5)


# 머신러닝 알고리즘을 이용해 데이터학습을 통한 모델 생성

svm_linear_fit <- train(Class ~.,

                        data = wine_svm_train,

                        method = "svmLinear",   #선형서포트벡터 머신방법

                        trControl = ctrl, # 학습 방법

                        preProcess = c("center","scale"), # 전처리(표준화)

                        metric = "Accuracy") # 모형평가 방법(정확도)

svm_linear_fit
```

```
ctrl <- trainControl(method = "repeatedcv", repeats = 5)

svm_poly_fit <- train(Class ~.,
                      data = wine_svm_train,
                      method = "svmPoly",   # 비선형 서포트 벡터 머신 방법
                      trControl = ctrl,
                      preProcess = c("center","scale"),
                      metric = "Accuracy")

svm_poly_fit
```

```
Support Vector Machines with Linear Kernel

124 samples
 13 predictor
  3 classes: '1', '2', '3'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 111, 112, 112, 111, 110, 112, ...
Resampling results:

  Accuracy   Kappa
  0.9738678  0.9604508
```

```
svm_linear_fit #선형 서포트벡터머신

pred_svm_linear_fit <- predict(svm_linear_fit, wine_svm_test)

# 혼동행렬

confusionMatrix(pred_svm_linear_fit, wine_svm_test$Class )

# 중요변수추출

importance_linear <- varImp(svm_linear_fit, scale=FALSE)

plot(importance_linear)
```

```
tb <- table(pred_svm_linear_fit, test_y)

error_tb <- 1-(sum(diag(tb))/sum(tb)) #오분류율 계산
```

Confusion Matrix and Statistics

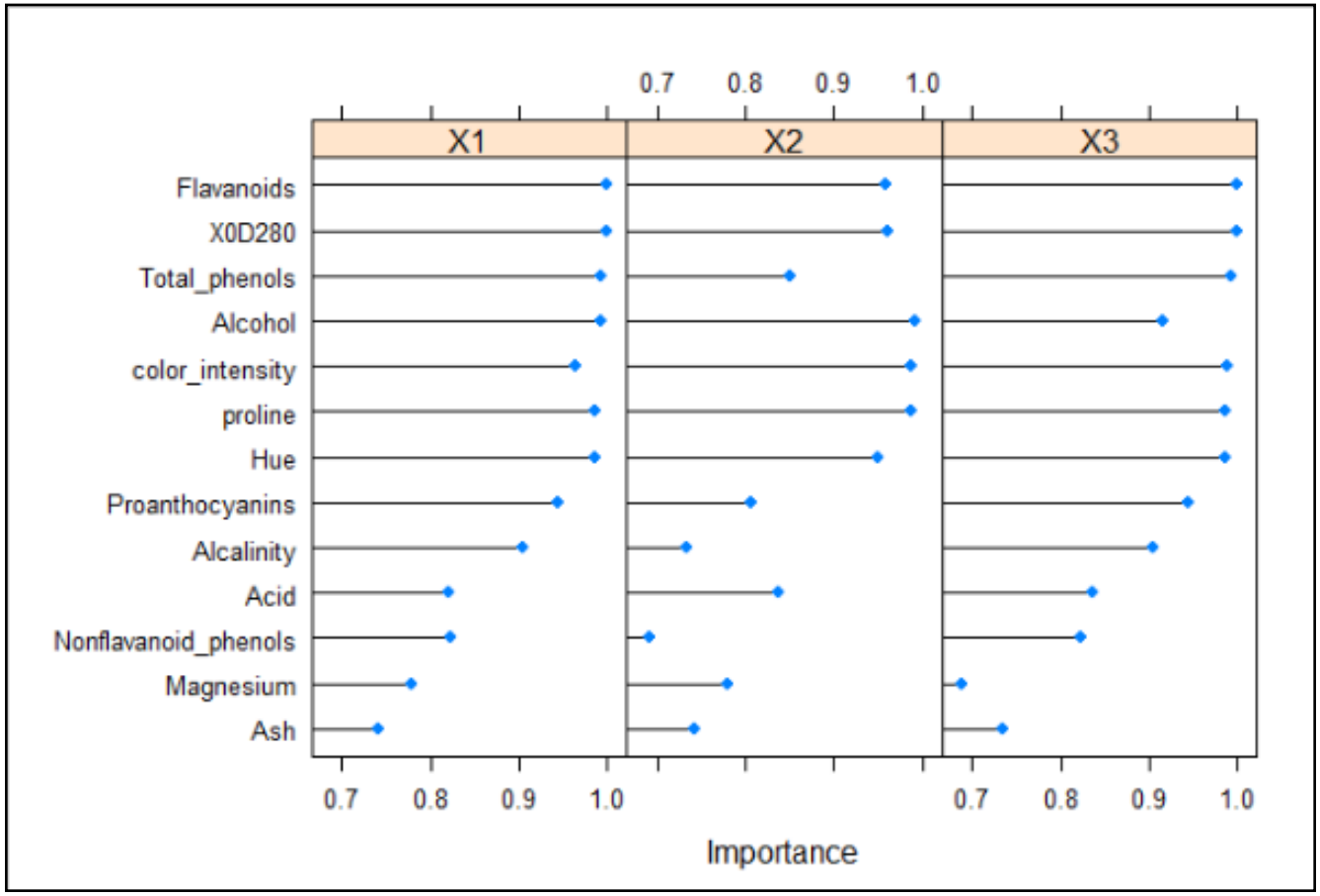
```
          Reference
Prediction  1  2  3
         1 14  1  0
         2  0 25  1
         3  0  1 12
```

Overall Statistics

```
               Accuracy : 0.9444
                 95% CI : (0.8461, 0.9884)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.459e-12

                  Kappa : 0.9117

 Mcnemar's Test P-Value : NA
```

Statistics by Class:

	Class: 1	Class: 2	Class: 3
Sensitivity	1.0000	0.9259	0.9231
Specificity	0.9750	0.9630	0.9756
Pos Pred Value	0.9333	0.9615	0.9231
Neg Pred Value	1.0000	0.9286	0.9756
Prevalence	0.2593	0.5000	0.2407
Detection Rate	0.2593	0.4630	0.2222
Detection Prevalence	0.2778	0.4815	0.2407
Balanced Accuracy	0.9875	0.9444	0.9493

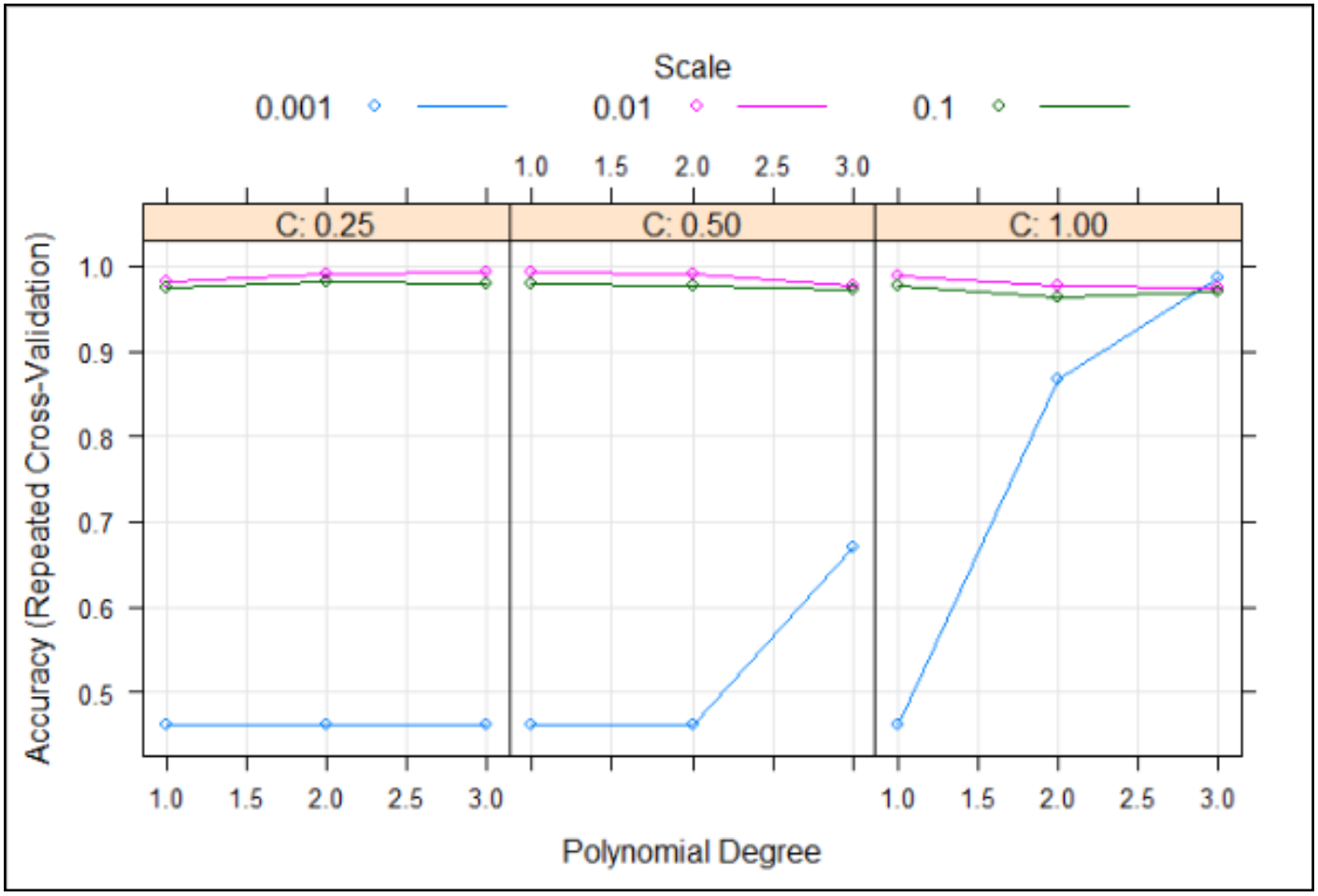


상위에 있을수록 중요 변수, 하위에 있을수록 덜 중요한 변수

```
svm_poly_fit #비선형 서포트벡터머신

plot(svm_poly_fit)

pred_svm_poly_fit <- predict(svm_poly_fit, wine_svm_test)

# 혼동행렬

confusionMatrix(pred_svm_poly_fit, wine_svm_test$Class)

# 중요변수추출

importance_poly <- varImp(svm_poly_fit, scale=FALSE)

plot(importance_poly)
```

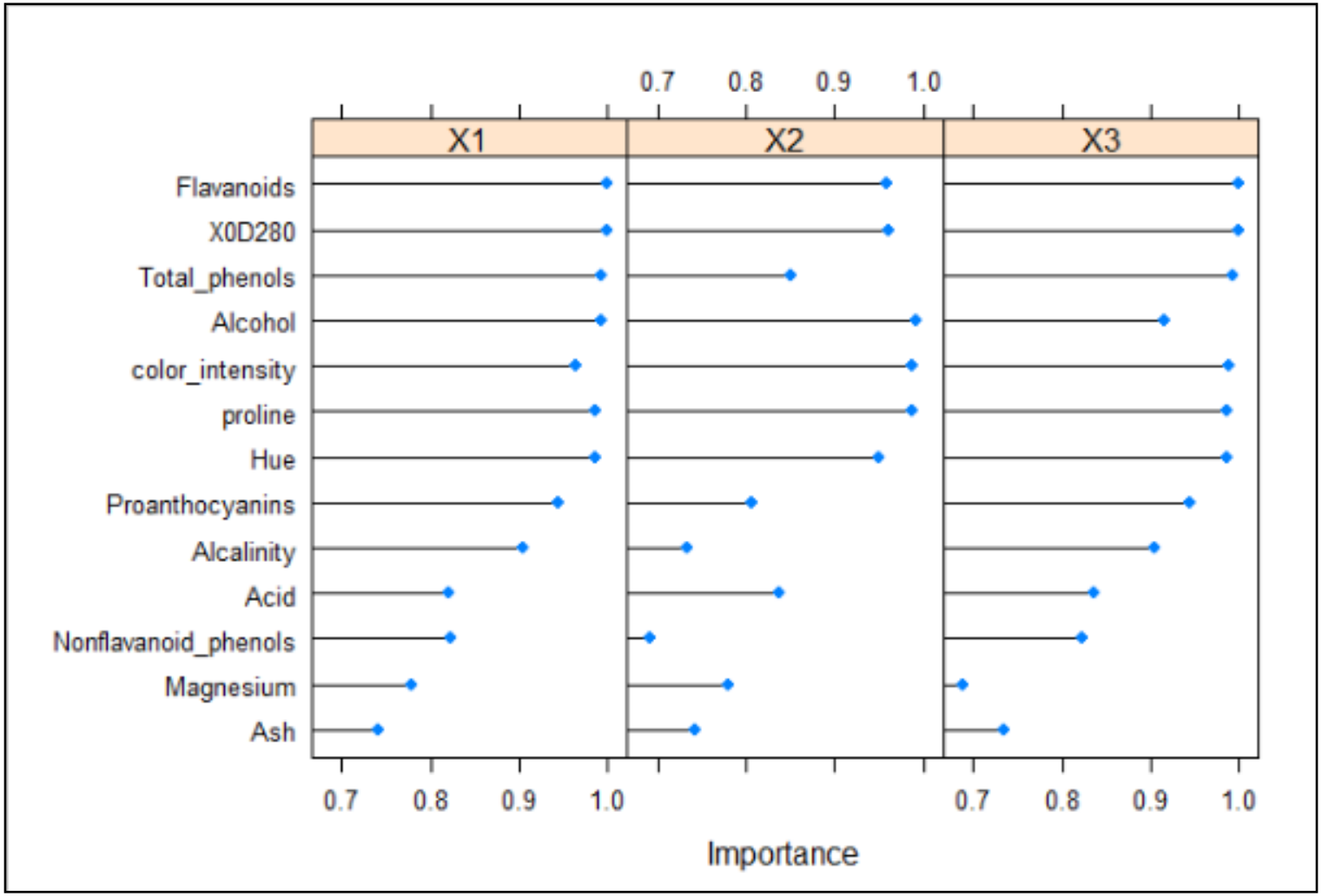
Support Vector Machines with Polynomial Kernel

degree	scale	C	Accuracy	Kappa
1	0.001	0.25	0.4624642	0.1779802
1	0.001	0.50	0.4624642	0.1779802
1	0.001	1.00	0.4624642	0.1779802
1	0.010	0.25	0.9817483	0.9722875
1	0.010	0.50	0.9919231	0.9878193
1	0.010	1.00	0.9885897	0.9828193
1	0.100	0.25	0.9755994	0.9633987
1	0.100	0.50	0.9788462	0.9682017
1	0.100	1.00	0.9757276	0.9635560
2	0.001	0.25	0.4624642	0.1779802
2	0.001	0.50	0.4624642	0.1779802
2	0.001	1.00	0.8678671	0.7973709
2	0.010	0.25	0.9901049	0.9850345
2	0.010	0.50	0.9902564	0.9853193
2	0.010	1.00	0.9774176	0.9661147
2	0.100	0.25	0.9805128	0.9706707
2	0.100	0.50	0.9757276	0.9634849
2	0.100	1.00	0.9623477	0.9432571
3	0.001	0.25	0.4624642	0.1779802
3	0.001	0.50	0.6694289	0.4893888
3	0.001	1.00	0.9849534	0.9772497
3	0.010	0.25	0.9919231	0.9878193
3	0.010	0.50	0.9757509	0.9636147
3	0.010	1.00	0.9740609	0.9610564
3	0.100	0.25	0.9802331	0.9702383
3	0.100	0.50	0.9721562	0.9578710
3	0.100	1.00	0.9690793	0.9532194


Scale
0.001
0.01
0.1
C: 0.25
C: 0.50
C: 1.00
Accuracy (Repeated Cross-Validation)
Polynomial Degree
1.0
1.5
2.0
2.5
3.0
0.5
0.6
0.7
0.8
0.9

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 14  3  0
         2  0 23  0
         3  0  1 13
```

```
Overall Statistics

               Accuracy : 0.9259
                 95% CI : (0.8211, 0.9794)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.901e-11

                  Kappa : 0.8848

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.8519   1.0000
Specificity            0.9250   1.0000   0.9756
Pos Pred Value         0.8235   1.0000   0.9286
Neg Pred Value         1.0000   0.8710   1.0000
Prevalence             0.2593   0.5000   0.2407
Detection Rate         0.2593   0.4259   0.2407
Detection Prevalence   0.3148   0.4259   0.2593
Balanced Accuracy      0.9625   0.9259   0.9878
```


0.7
0.8
0.9
1.0
X1
X2
X3
Flavanoids
X0D280
Total_phenols
Alcohol
color_intensity
proline
Hue
Proanthocyanins
Alcalinity
Acid
Nonflavanoid_phenols
Magnesium
Ash
0.7
0.8
0.9
1.0
0.7
0.8
0.9
1.0
Importance

선형 SVM과 비선형 SVM의 정확도 비교