머신러닝
(Machine Learning)

LECTURE XV: 지도 학습 9
(Supervised Learning)

Dai-Gyoung Kim

Department of Applied Mathematics

Hanyang University ERICA

ⓒ dgkim@hanyang.ac.kr

지도 학습 (Supervised Learning)

Contents

 ⓒ dgkim@hanyang.ac.kr

▶ 커널 서포트 벡터 머신

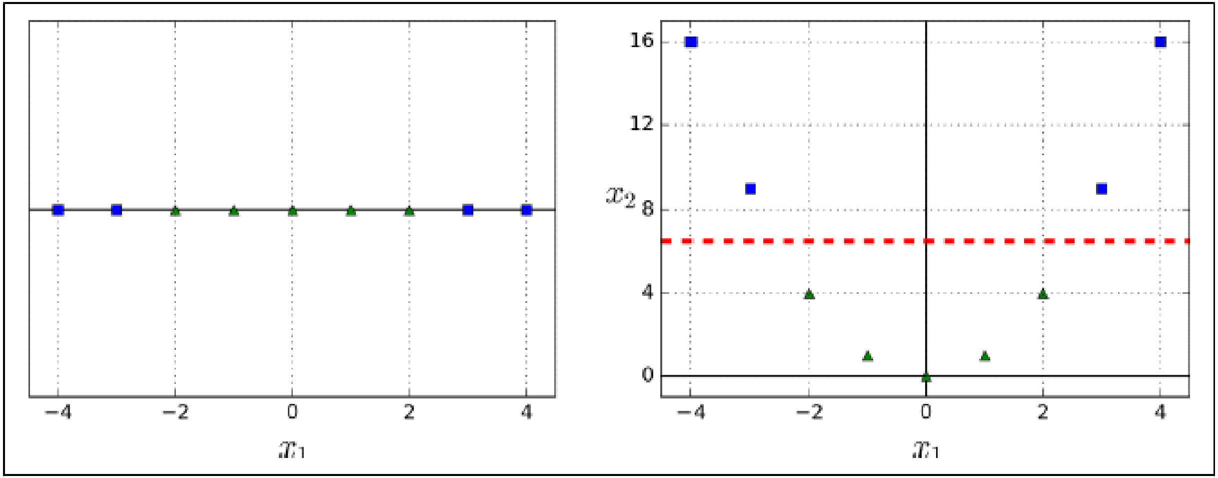
선형 SVM 분류기가 효율적이고 잘 작동하지만, 선형분류가 불가능한 데이터셋에 대해서는 비효율적임. 이를 극복하기 위해 커널의 개념을 도입함. 커널 서포트 벡터 머신(Kernel Support Vector Machine)은 초평면(hyperplane)으로 모델링할 수 없는 더 복잡한 모델(비선형)을 만들 수 있도록 기존의 선형 SVM을 확장한 것임.

❖ 선형 모델과 비선형 특성

- 직선과 초평면은 비선형 데이터에 대해 유연하지 못하여 저차원 데이터셋에서는 선형 모델이 매우 제한적임.
- 선형 모델을 유연하게 만드는 한 가지 방법은 특성끼리 곱하거나 특성을 거듭제곱으로 하여 새로운 특성 (다항 특성) 을 추가하여 차원을 높이는 것임.

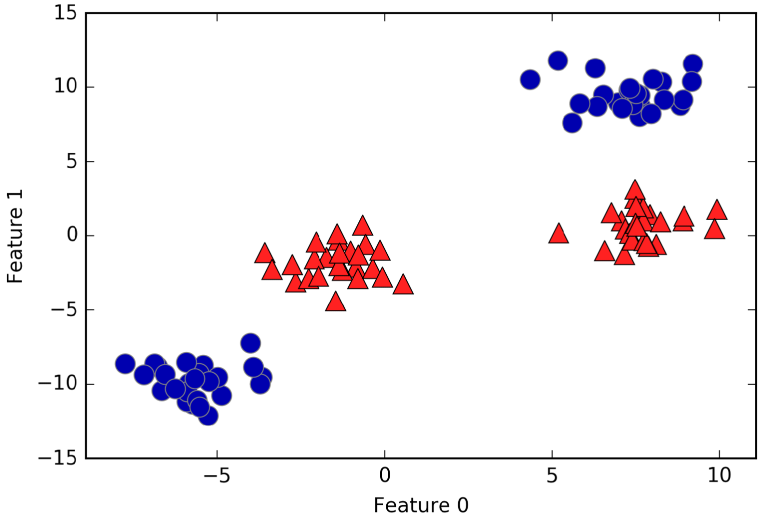
ⓒ dgkim@hanyang.ac.kr

- 다음은 하나의 특성 x_1에 두 번째 특성 $x_2 = (x_1)^2$을 추가하여 비선형 데이터를 선형적으로 구분되는 데이터셋을 생성한 것임.



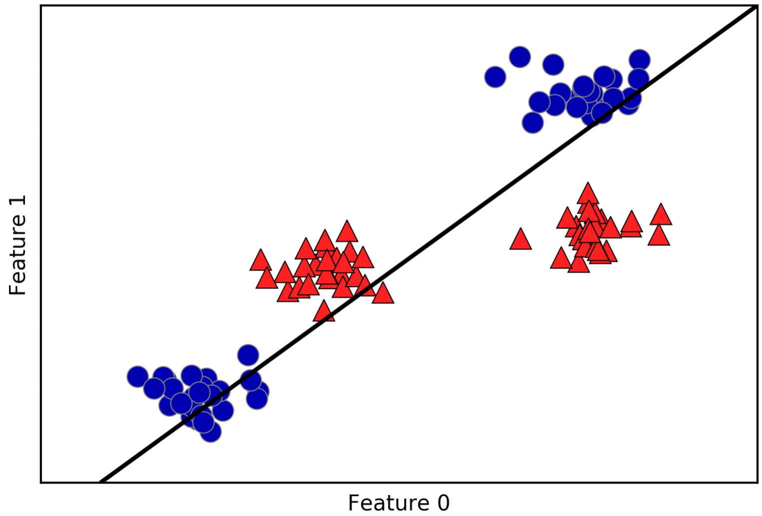
- 선형적으로 구분되지 않는 클래스를 가진 이진 분류 데이터셋의 예.

```
X, y = make_blobs(centers=4, random_state=8)
y = y % 2

mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



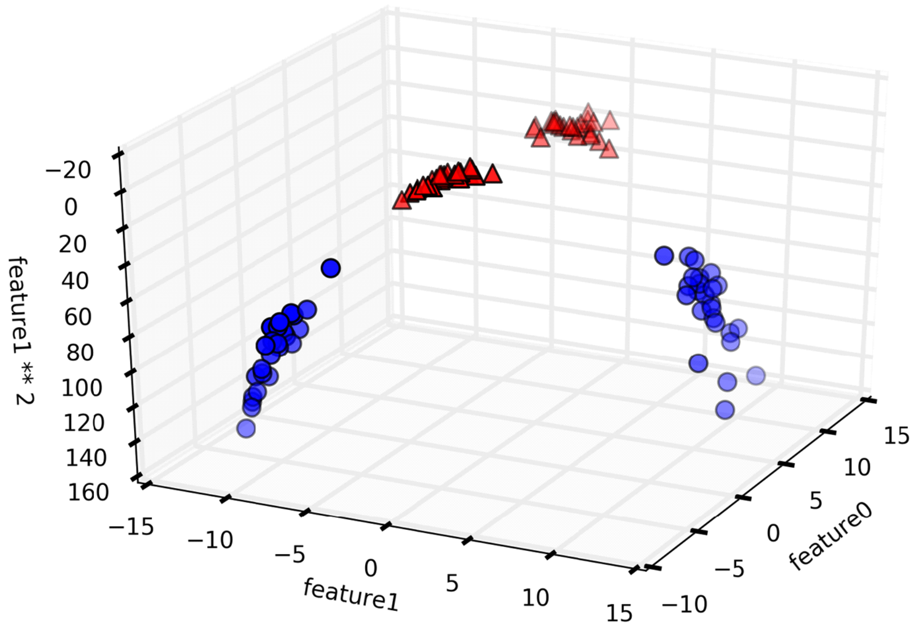
- 다음은 위 데이터셋에 적용한 선형 SVM으로 만들어진 결정 경계임.

```
from sklearn.svm import LinearSVC
linear_svm = LinearSVC().fit(X, y)

mglearn.plots.plot_2d_separator(linear_svm, X) mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



- 위 데이터의 두 번째 특성을 제곱하여 새로운 특성을 추가하여 입력 특성을 확장함.
  - ✓ (특성0, 특성1)에서 (특성0, 특성1, (특성1)^2)으로 확장.

```
# add the squared second feature
X_new = np.hstack([X, X[:, 1:] ** 2])

from mpl_toolkits.mplot3d import Axes3D, axes3d
figure = plt.figure()
# visualize in 3D
ax = Axes3D(figure, elev=-152, azim=-26)
# plot first all the points with y == 0, then all with y == 1
mask = y == 0
ax.scatter(X_new[mask, 0], X_new[mask, 1], X_new[mask, 2], c='b',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.scatter(X_new[~mask, 0], X_new[~mask, 1], X_new[~mask, 2], c='r', marker='^',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.set_xlabel("feature0")
ax.set_ylabel("feature1")
ax.set_zlabel("feature1 ** 2")
```

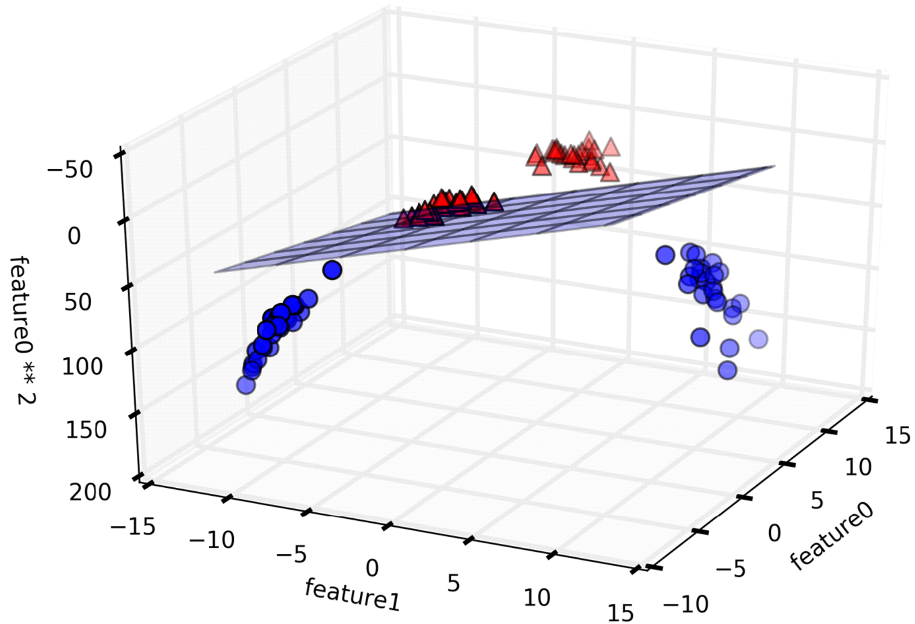


✓ 새로운 데이터셋에서는 3차원 공간에서 선형 모델(평면)을 사용해 두 클래스를 구분할 수 있음.

- 다음은 확장된 데이터셋에서 선형 모델을 만드는 코드임.

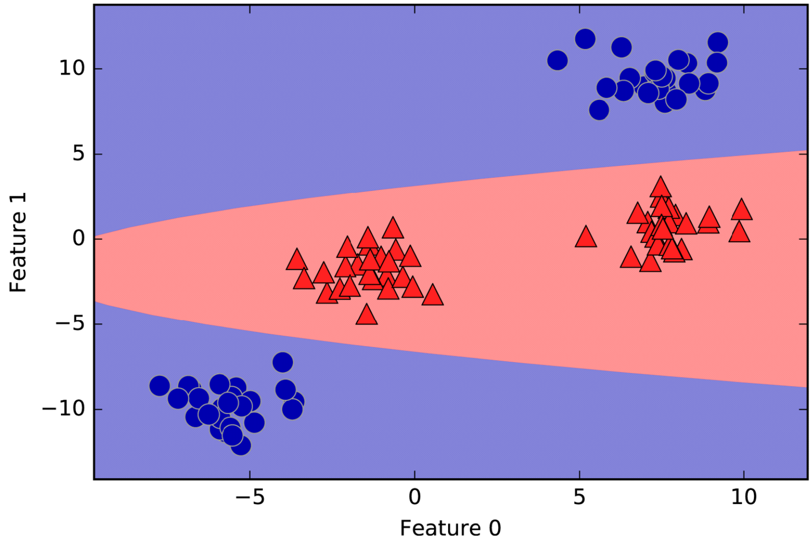
ⓒ dgkim@hanyang.ac.kr

```
linear_svm_3d = LinearSVC().fit(X_new, y)
coef, intercept = linear_svm_3d.coef_.ravel(), linear_svm_3d.intercept_

# show linear decision boundary
figure = plt.figure()
ax = Axes3D(figure, elev=-152, azim=-26)
xx = np.linspace(X_new[:, 0].min() - 2, X_new[:, 0].max() + 2, 50)
yy = np.linspace(X_new[:, 1].min() - 2, X_new[:, 1].max() + 2, 50)

XX, YY = np.meshgrid(xx, yy)
ZZ = (coef[0] * XX + coef[1] * YY + intercept) / -coef[2]
ax.plot_surface(XX, YY, ZZ, rstride=8, cstride=8, alpha=0.3)
ax.scatter(X_new[mask, 0], X_new[mask, 1], X_new[mask, 2], c='b',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.scatter(X_new[~mask, 0], X_new[~mask, 1], X_new[~mask, 2], c='r', marker='^',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.set_xlabel("feature0")
ax.set_ylabel("feature1")
ax.set_zlabel("feature1 ** 2")
```



- 분류된 데이터셋을 원래 특성 공간으로 투영하면 다음과 같이 비선형 곡선으로 분류됨.

```
ZZ = YY ** 2
dec = linear_svm_3d.decision_function(np.c_[XX.ravel(), YY.ravel(), ZZ.ravel()])
plt.contourf(XX, YY, dec.reshape(XX.shape), levels=[dec.min(), 0, dec.max()],
             cmap=mglearn.cm2, alpha=0.5)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



❖ 커널 기법

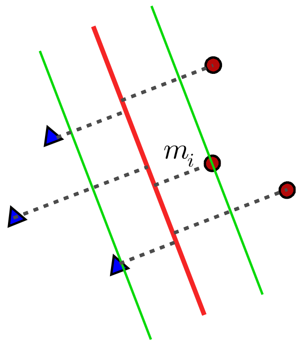
- 데이터셋에 비선형 특성을 추가하여 선형 모델을 만들려면 <u>어떠한 특성을 추가해야 할지 결정</u>해야 함.
  - ✓ 예를 들어 다항식 특성을 추가하는 기법은 간단하고 모든 머신러닝 알고리즘에서 잘 작동함.
- 한편 특성을 많이 추가하면 특성 차원 매우 커지고 연산 비용이 증가함.
  - ✓ 매우 복잡한 데이터셋을 잘 표현하기 위해 높은 차수의 다항식 특성을 많이 추가하면 모델이 느리게 작동하게 됨.
- SVM을 적용할 때는 커널 기법(kernel trick)이라는 (거의 기적적인) 수학적 방법을 도입하면 비선형 데이터셋으로 모델 효과적으로 학습할 수 있음.
  - ✓ 실제로 특성을 추가하지 않으면서 다항식 특성을 많이 추가한 것과 같은 결과를 얻을 수 있음.
  - ✓ 어떤 새로운 특성을 추가하지 않기 때문에 엄청난 수의 특성 조합이 생기지 않음.

하드 마진 선형 SVM 분류기의 최적화 문제

- 선형 서포트 벡터 머신은 다음의 최적화 문제의 해법을 기반으로 함.

$$
\text{(P)} \qquad \begin{aligned} &\underset{w,\,b}{\text{Minimize}} && h(w) = \frac{1}{2} w^T w \\ &\text{subject to} && \left(w^T \boldsymbol{x}_i + b\right) t_i \geq 1,\ \ i = 1,\ \ldots\ , m \end{aligned}
$$

✓ 위 최적화 문제를 풀어 w와 b를 결정함.



$$
m_i = \frac{\left(w^T \boldsymbol{x}_i + b\right) t_i}{\| \, w \, \|}
$$

- 위 문제 (P)는 쌍대 문제로 바꾸어 풀 수 있음.

쌍대 문제(Dual Problem)

- 커널 기법을 이용하려면 최적화 원 문제(P)를 쌍대 문제의 형태로 바꿔보아야 함.
- 먼저 KKT승수(Karush–Kuhn–Tucker multiplier)를 도입하여 다음과 같은 라그랑주 함수(Lagrange function)를 구성함.

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{m} \alpha_i \left[t_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) - 1 \right]$$

$$\alpha_i \geq 0, \; i = 1, \ldots, m$$

- 원 문제 (P)와 동치가 되는 라그랑주 쌍대 문제는 다음과 같음.

(D)

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{Maximize}} && h(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w}, b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \\ &\text{subject to} && \alpha_i \geq 0, \; i = 1, \ldots, m \end{aligned}$$

- $L(w, b, \alpha)$의 최적성 조건(optimality condition)에 의하여

$$\nabla_w L = 0 \implies \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i t_i \boldsymbol{x}_i$$

$$\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{m} \alpha_i t_i = 0$$

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{m} \alpha_i \left[t_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) - 1\right]$$

- 위 식을 $L(w, b, \alpha)$에 대입하면, α에 관한 함수로 바꿀 수 있음.

$$h(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

ⓒ dgkim@hanyang.ac.kr

- 이제 쌍대 문제 (D)를 다음과 같이 바꿔 쓸 수 있음.

(D*)

$$\underset{\boldsymbol{\alpha}}{\text{Minimize}} \quad h(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j t_i t_j \boldsymbol{x}_i^T\boldsymbol{x}_j - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^{m}\alpha_i t_i = 0, \quad \alpha_i \geq 0, \; i = 1, \dots, m$$

✓ 위의 최적화 문제는 최적화할 변수가 m개인 2차형 계획문제 (quadratic programming)임.

- 문제 (D*)의 해 $\hat{\alpha}$를 이용하면, 원 문제 (P)의 해를 다음과 같이 구할 수 있음.

$$\hat{\boldsymbol{w}} = \sum_{i=1}^{m}\hat{\alpha}_i t_i \boldsymbol{x}_i$$

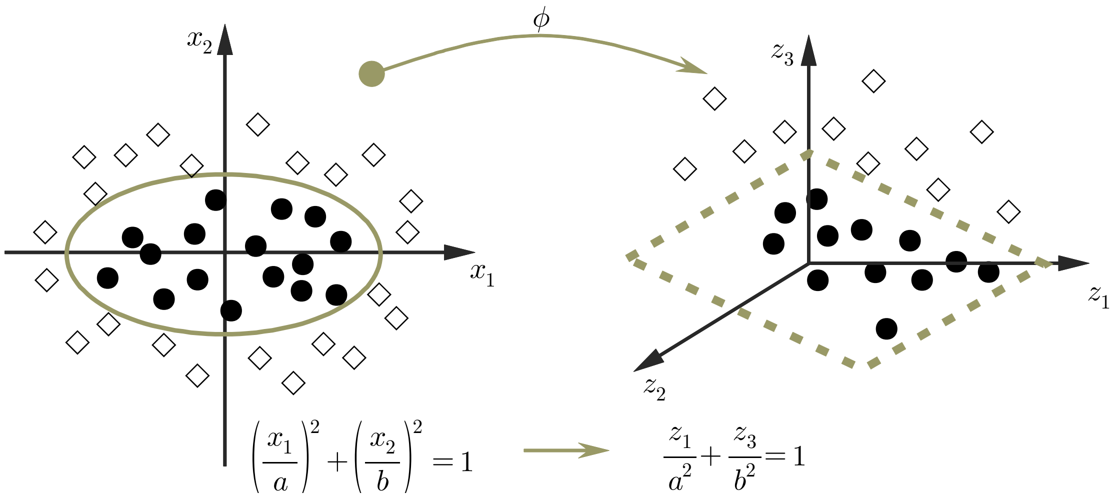
$$\hat{b} = \frac{1}{n_s}\sum_{\substack{i=1 \\ \hat{\alpha}_i > 0}}^{m}\left(t_i - \hat{\boldsymbol{w}}^T\boldsymbol{x}_i\right), \quad n_s = \#\left\{i \,\middle|\, \hat{\alpha}_i > 0\right\}$$

- 위 쌍대문제 해법을 이용하면 원 문제 (P)에 적용할 수 없는 커널 기법이 가능함.

커널 SVM의 아이디어

- 먼저 2차원 데이터셋에 2차 다항식 변환을 적용하고 선형 SVM 분류기를 변환된 훈련 세트에 적용하는 경우를 보고자 함.

$$\phi(x_1, x_2) = (x_1^2,\ \sqrt{2}\,x_1 x_2,\ x_2^2) = (z_1,\ z_2,\ z_3)$$



- 2차 다항식 매핑 함수:

$$\phi(\boldsymbol{x}) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}\,x_1 x_2 \\ x_2^2 \end{bmatrix}$$

- 2차 다항식 매핑을 위한 커널 기법

$$\phi(\boldsymbol{a})^T \phi(\boldsymbol{b}) = \begin{bmatrix} a_1^2 \\ \sqrt{2}\,a_1 a_2 \\ a_2^2 \end{bmatrix}^T \begin{bmatrix} b_1^2 \\ \sqrt{2}\,b_1 b_2 \\ b_2^2 \end{bmatrix} = a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2$$

$$= (a_1 b_1 + a_2 b_2)^2 = \left(\boldsymbol{a}^T \boldsymbol{b}\right)^2$$

✓ 위의 경우, 고차원 데이터를 내적한 결과와 저차원 데이터를 내적한 것을 커널함수로 변환한 결과가 같음.

✓ 위 예에서 커널 함수는 $K(a,b) = \phi(a)^T \phi(b)$로 정의되며

✓ $K(\boldsymbol{a},\boldsymbol{b}) = \left(\boldsymbol{a}^T \boldsymbol{b}\right)^2$

- 커널 기법의 아이디어는 **실제로 데이터를 확장하지 않고 확장된 특성에 대한 데이터 포인트들의 내적을 간접적(효율적)으로 계산**하는 것임.

[쌍대 문제에 적용]

- 쌍대 문제 (D*)에서 모든 훈련 데이터 포인트에 변환 ϕ를 적용하면 (D*)의 목적 함수의 첫째 항을 다음과 같이 바꿔 쓸 수 있음.

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j t_i t_j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j t_i t_j \left[\boldsymbol{x}_i^T\boldsymbol{x}_j\right]^2$$

✓ 위 식을 통하여 실제로 훈련 샘플을 변환할 필요가 없이, 훈련 샘플의 내적의 결과에 제곱을 가하면 됨.

✓ 이 기법이 전체 과정에 필요한 계산량 측면에서 획기적인 효율성을 제공함.

일반적인 커널

- 위 예에서 $K(a,b)=(a^Tb)^2$을 2차 다항식 커널이라 함.
- 커널 함수 ϕ는 원래 벡터 a와 b에 기반하여 $\phi(a)^T\phi(b)$을 계산할 수 있는 함수임.
- 범용적으로 쓰이는 커널:

1) 선형: $K(a,b)=a^Tb$

2) 다항식: $K(a,b)=(\gamma a^Tb+\epsilon)^d,\ \gamma>0$

3) 가우시안(Gaussian) RBF(Radial Basis Function):

$$K(a,b)=e^{-\gamma\|a-b\|^2},\ \gamma>0$$

4) 시그모이드(Sigmoid): $K(a,b)=\tanh(\gamma a^Tb+\epsilon),\ \gamma>0$

✓ 문제에 따라 최적의 커널 함수를 적용해야 함.

✓ 가우시안 커널과 시그모이드 커널은 무한차원의 특성 공간에 데이터를 매핑하는 효과가 있음. 즉 모든 차수의 다항 특성을 고려함.

[머서의 정리]

- 머서의 정리(Mercer's Theorem)에 의하면 머서의 조건을 만족하면 a와 b를 더 높은 차원의 다른 공간에 매핑하는 $K(a,b)=\phi(a)^T\phi(b)$를 만족하는 함수 ϕ가 존재함.
  - ✓ 가우시안 RBF 커널의 경우 ϕ는 각 훈련 샘플을 무한 차원 공간에 매핑함. 그러나 실제로 매핑하지 않고 $a^T b$ 결과 값에 ϕ를 적용함. 이것이 **커널 트릭**(기법)임.

- 새로운 샘플 x_n을 입력하여 커널 SVM으로 예측하기

$$
\begin{aligned}
h_{\hat{w},\hat{b}} = \hat{w}^T\phi(x_n) + \hat{b} &= \left(\sum_{i=1}^{m}\hat{\alpha}_i t_i \phi(x_i)\right)^T \phi(x_n) + \hat{b} \\
&= \sum_{i=1}^{m}\hat{\alpha}_i t_i \left(\phi(x_i)^T\phi(x_n)\right) + \hat{b} \\
&= \sum_{\substack{i=1 \\ \hat{\alpha}_i>0}}^{m}\hat{\alpha}_i t_i K(x_i, x_n) + \hat{b}
\end{aligned}
$$

✓ 위 식에서 서포트 벡터에 대해서만 $\hat{\alpha}_i \neq 0$이기 때문에 예측을 만드는 데 전체 샘플이 아니라 **서포트 벡터와 새로운 x_n간의 내적만**을 계산하면 됨.

- 커널 트릭을 적용한 가중치 벡터와 편향 계산

$$\hat{\boldsymbol{w}} = \sum_{i=1}^{m} \hat{\alpha}_i t_i \phi(\boldsymbol{x}_i)$$

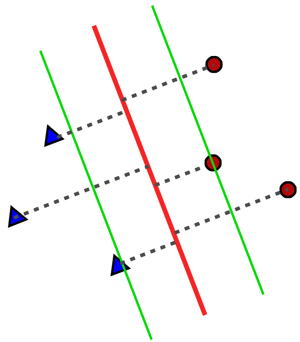
$$\hat{b} = \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}_i > 0}}^{m} \left(t_i - \hat{\boldsymbol{w}}^T \phi(\boldsymbol{x}_i) \right)$$

$$= \frac{1}{n_s} \sum_{\substack{i=1 \\ \hat{\alpha}_i > 0}}^{m} \left(t_i - \sum_{\substack{j=1 \\ \hat{\alpha}_j > 0}}^{m} \hat{\alpha}_j t_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) \right), \quad n_s = \#\left\{ i \,\middle|\, \hat{\alpha}_i > 0 \right\}$$

ⓒ dgkim@hanyang.ac.kr

❖ 예시로 SVM 이해하기

- 두 클래스 사이의 경계(support hyperplane)에 위치한 데이터 포인트는 **서포트 벡터임.**



- 새로운 데이터 포인트에 대해 예측하려면 각 서포트 벡터와 거리를 측정함.
- 분류 결정은 서포트 벡터까지의 거리에 기반하며 서포트 벡터의 중요도는 훈련 과정에서 학습함 (SVC객체의 ‘dual_coef_’ 속성에 저장됨).

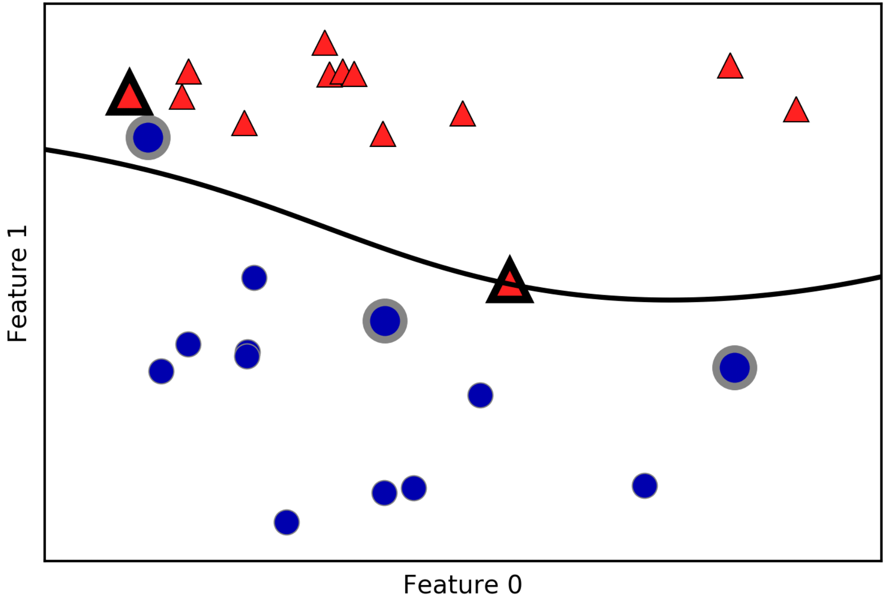
- 데이터 포인트 사이의 거리는 가우시안 커널에 의해 계산됨.

$$K_{rbf}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \| \boldsymbol{x}_i - \boldsymbol{x}_j \|^2}$$

- 다음은 두 개의 클래스를 가진 2차원 'forge' 데이터셋에 SVM을 학습시킨 결과를 보여 주는 코드임.

```
from sklearn.svm import SVC

X, y = mglearn.tools.make_handcrafted_dataset()
svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X, y)
mglearn.plots.plot_2d_separator(svm, X, eps=.5)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)

# plot support vectors
sv = svm.support_vectors_
```

```
# class labels of support vectors are given by the sign of the dual coefficients
sv_labels = svm.dual_coef_.ravel() > 0
mglearn.discrete_scatter(sv[:, 0], sv[:, 1], sv_labels, s=15, markeredgewidth=3)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



❖ SVM 매개변수 튜닝

- 가우시안 커널 SVM의 매개변수: C, γ

1) C는 규제에 관한 매개변수:

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

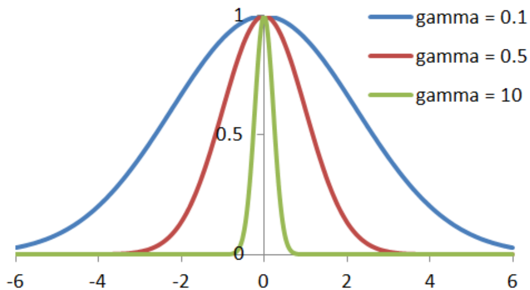
✓ 이 매개변수는 각 데이터 포인트의 중요도 (‘dual_coef_’ 값)를 제한함.

2) γ는 가우시안 커널 함수 커널 폭의 역수:

$$K(\boldsymbol{a},\boldsymbol{b}) = e^{-\gamma\|\boldsymbol{a}-\boldsymbol{b}\|^2}$$

✓ $\gamma\downarrow$ 이면 커널 폭이 커지며, 훈련 샘플의 영향 범위가 커짐.

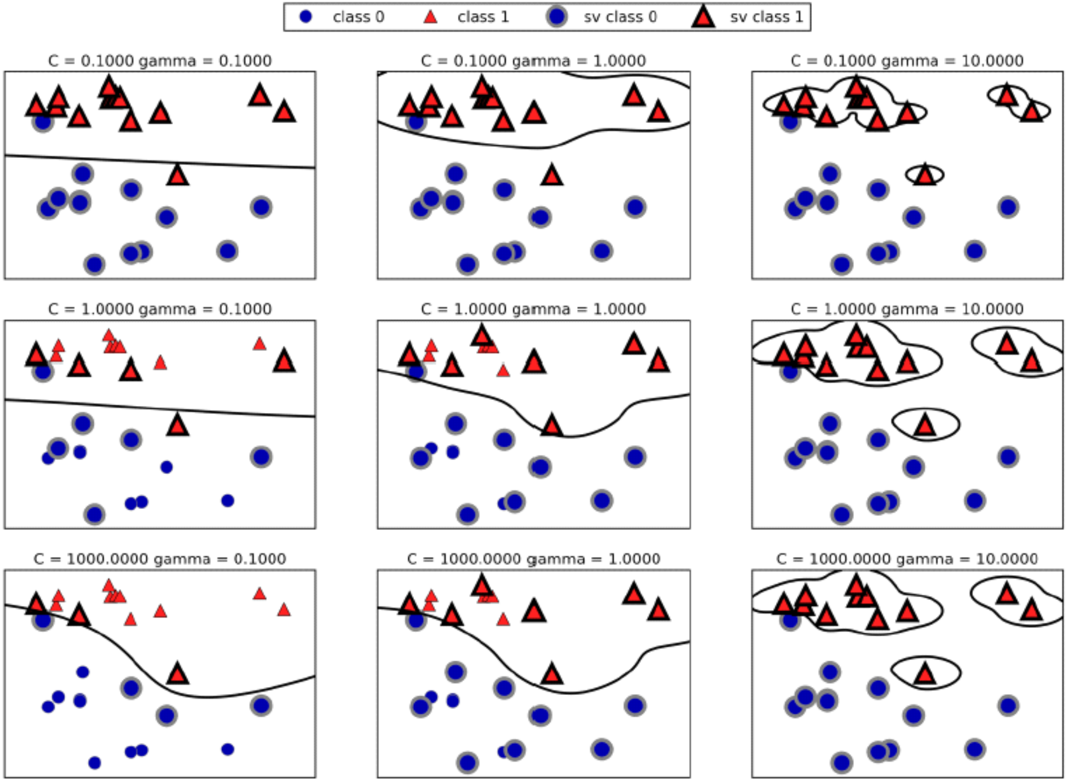
✓ $\gamma\uparrow$ 이면 커널 폭이 작아지며, 훈련 샘플의 영향 범위가 작게 집중됨.



	가중치 벡터의 분산	모델의 적합도
$C\uparrow$ 또는 $\gamma\uparrow$	커짐	과대적합
$C\downarrow$ 또는 $\gamma\downarrow$	작아짐	과소적합

- 다음은 매개변수를 다르게 했을 때 변화되는 모델을 보여주는 코드임.

```
fig, axes = plt.subplots(3, 3, figsize=(15, 10))

for ax, C in zip(axes, [-1, 0, 3]):
    for a, gamma in zip(ax, range(-1, 2)):
        mglearn.plots.plot_svm(log_C=C, log_gamma=gamma, ax=a)

axes[0, 0].legend(["class 0", "class 1", "sv class 0",
                  "sv class 1"], ncol=4, loc=(.9, 1.2))
```



✓ 매개변수 C와 γ 설정에 따른 결정 경계와 서포트 벡터

- 다음 예는 가우시안 커널 SVM을 유방암 데이터셋에 적용한 모델임.
  - ✓ 매개변수 기본값은 C=1, gamma=1/n_features:

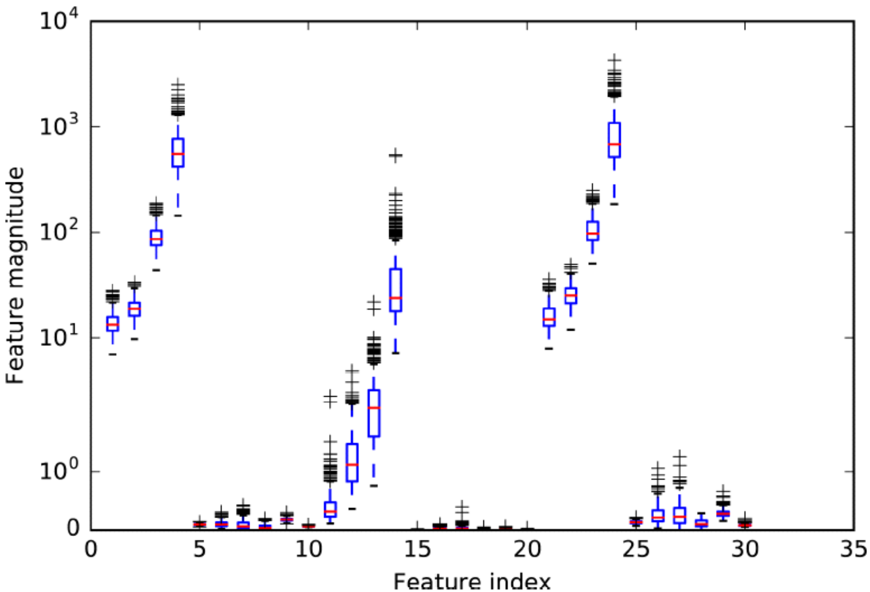
```
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, random_state=0)

svc = SVC()
svc.fit(X_train, y_train)

print("Accuracy on training set: {:.2f}".format(svc.score(X_train, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(X_test, y_test)))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.63
```

  - ✓ 이 모델은 훈련 세트에 100% 정확하지만, 테스트 세트에 대해서는 63%의 정확도를 갖고 있음. 이는 상당히 과대적합되어 있음.
  - ✓ <u>SVM은 매개변수 설정과 데이터 특성간의 스케일에 매우 민감함</u>.

- 유방암 데이터셋의 각 특성의 최솟값과 최댓값을 로그스케일로 보기.

```
plt.boxplot(X_train, manage_xticks=False)
plt.yscale("symlog")
plt.xlabel("Feature index")
plt.ylabel("Feature magnitude")
```



▶ SVM을 위한 전처리

- 데이터에 적용할 가장 중요한 변환 중 하나는 특성 스케일링(feature scaling)임.
- 입력 특성 스케일이 많이 다르면 대부분 머신러닝 알고리즘은 잘 작동하지 않음 (타깃값에 대한 스케일링은 일반적으로 수행하지 않음).
- 모든 특성의 범위가 같아지도록 만들어 주는 범용적인 방법은 다음과 같음.

1) min-max 스케일링

✓ 특성값이 0~1범위에 들도록 변환함.

✓ 사이킷런에서 'MinMaxScaler' 변환기를 제공함.

2) 표준화(standardization)

✓ 특성값이 평균이 0, 표준편차가 1이 되도록 표준화함.

✓ 사이킷런에서 'StandardScaler' 변환기를 제공함.

- 다음은 유방암 데이터셋을 min-max 스케일링으로 전처리 한 후 SVM 모델을 학습한 결과임.

```
# compute the minimum value per feature on the training set
min_on_training = X_train.min(axis=0)
# compute the range of each feature (max - min) on the training set
range_on_training = (X_train - min_on_training).max(axis=0)

# subtract the min, and divide by range
# afterward, min=0 and max=1 for each feature
X_train_scaled = (X_train - min_on_training) / range_on_training

print("Minimum for each feature\n{}", X_train_scaled.min(axis=0))
print("Maximum for each feature\n{}", X_train_scaled.max(axis=0))
```

```
Minimum for each feature
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Maximum for each feature
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

```
# use THE SAME transformation on the test set,
# using min and range of the training set (see Chapter 3 for details)
X_test_scaled = (X_test - min_on_training) / range_on_training

svc = SVC()
svc.fit(X_train_scaled, y_train)

print("Accuracy on training set: {:.3f}".format(
svc.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

✓ 테스트 세트의 스케일링에 대해서는 훈련세트에서 계산한 최솟값과 범위를 사용함.

```
Accuracy on training set: 0.948
Accuracy on test set: 0.951
```

✓ 테스트 세트에 대해서 정확도가 많이 개선됨.

- 다음은 C 값을 증가 시켜 학습하여 좀 더 복잡하게 만든 모델의 성능을 보여줌.

```python
svc = SVC(C=1000)
svc.fit(X_train_scaled, y_train)

print("Accuracy on training set: {:.3f}".format(
svc.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

```
Accuracy on training set: 0.988
Accuracy on test set: 0.972
```

✓ 테스트 세트에 대해서 모델 성능이 97.2%로 좋아짐.

❖ 장단점과 매개변수

[장점]

- 커널 SVM은 다양한 데이터셋에서 잘 작동하는 강력한 모델임.
- 데이터의 특성이 몇 개 안 되더라도 복잡한 결정 경계를 만들 수 있음.

[단점]

- 데이터 샘플이 많을 때는 잘 작동하지 않음.
  - ✓ 100,000개 이상의 데이터셋에 대해서는 속도와 메모리 관점에서 비효율적임
- 효과적인 결과를 위해 데이터의 전처리가 필수적임.
- 매개변수 설정에 많은 신경을 써야함.
  - ✓ 랜덤 포레스트나 그래디언트 부스팅 같은 트리기반 모델은 전처리가 필요 없으며 매개변수 조정이 용이함.
- 커널 SVM 모델은 분석하기 여려움.
  - ✓ 예측이 어떻게 결정되었는지 이해하기 여려움.

[매개변수]

- 커널 SVM 모델에서 중요한 매개변수와 선택사항은 다음과 같음.

  1) 규제 매개변수: C

  2) 커널의 선택: 가우시안 RBF 커널 등등

  3) 커널에 따른 매개변수: γ 등등