In [1]:

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [2]:

```
import mglearn
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import sklearn
```

In [3]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [20]:

```
from matplotlib import font_manager, rc

plt.rcParams['axes.unicode_minus'] = False
font_name = font_manager.FontProperties(fname="C:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

plt.rcParams['axes.unicode_minus'] = False
```

In [5]:

```
cancer = load_breast_cancer()

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

gbrt = GradientBoostingClassifier(random_state=0)
gbrt.fit(X_train, y_train)

print("훈련 세트 정확도: {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(gbrt.score(X_test, y_test)))
```

훈련 세트 정확도: 1.000
테스트 세트 정확도: 0.958

In [6]:

```
gbrt = GradientBoostingClassifier(random_state=0, max_depth=1)
gbrt.fit(X_train, y_train)

print("훈련 세트 정확도: {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(gbrt.score(X_test, y_test)))
```

훈련 세트 정확도: 0.991
테스트 세트 정확도: 0.972

In [7]:

```
gbrt = GradientBoostingClassifier(random_state=0, learning_rate=0.01)
gbrt.fit(X_train, y_train)

print("훈련 세트 정확도: {:.3f}".format(gbrt.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(gbrt.score(X_test, y_test)))
```

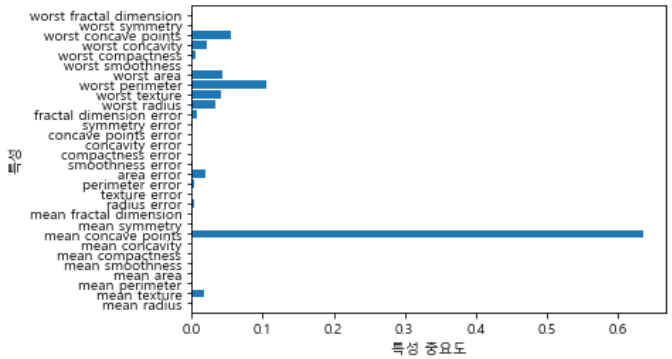
훈련 세트 정확도: 0.988
테스트 세트 정확도: 0.965

In [8]:

```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1]
    plt.barh(np.arange(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), cancer.feature_names)
    plt.xlabel("특성 중요도")
    plt.ylabel("특성")
    plt.ylim(-1, n_features)
```

In [21]:

```
gbrt = GradientBoostingClassifier(random_state=0, learning_rate=0.01)
gbrt.fit(X_train, y_train)

plot_feature_importances_cancer(gbrt)
```



In [10]:

```
from sklearn.datasets import make_blobs
```

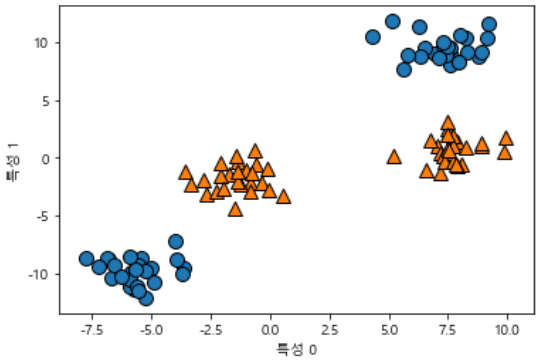
서포트 벡터 머신

In [22]:

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

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

Out[22]:

```
Text(0, 0.5, '특성 1')
```



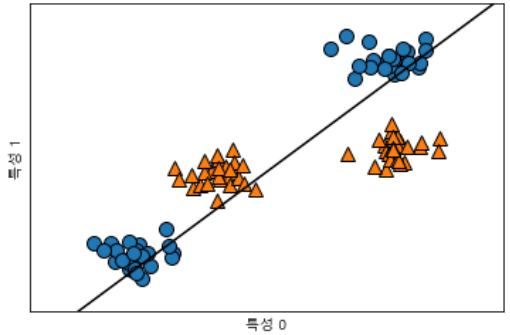
In [23]:

```
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("특성 0")
plt.ylabel("특성 1")
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩svm₩base.py:931: ConvergenceWar
ning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[23]:

```
Text(0, 0.5, '특성 1')
```

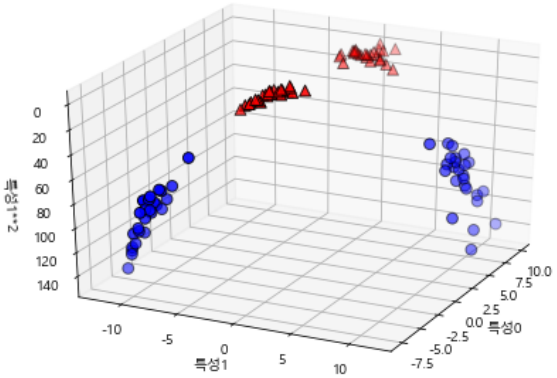


In [13]:

```
X_new = np.hstack([X, X[:, 1:] ** 2])
```

In [24]:

```
from mpl_toolkits.mplot3d import Axes3D, axes3d

figure = plt.figure()

ax = Axes3D(figure, elev=-152, azim=-26)
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("특성0")
ax.set_ylabel("특성1")
ax.set_zlabel("특성1**2")
plt.show()
```



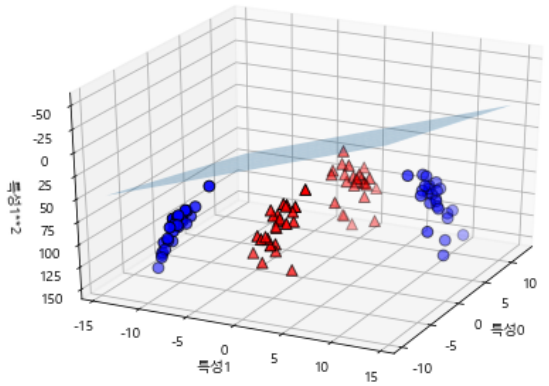
In [25]:

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

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("특성0")
ax.set_ylabel("특성1")
ax.set_zlabel("특성1**2")
```

```
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWsvmWbase.py:931: ConvergenceWar
ning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[25]:

```
Text(0.5, 0, '특성1**2')
```

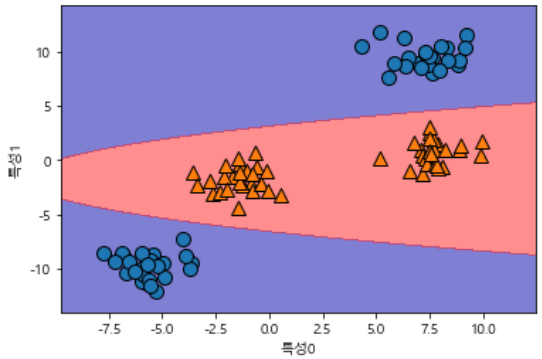


In [26]:

```
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("특성0")
plt.ylabel("특성1")
```

Out[26]:

```
Text(0, 0.5, '특성1')
```



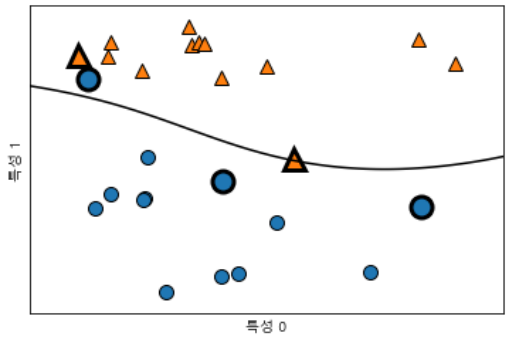
SVM 이해하기

In [27]:

```
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)

sv = svm.support_vectors_
sv_labels = svm.dual_coef_.ravel() > 0

mglearn.discrete_scatter(sv[:, 0], sv[:, 1], sv_labels, s=15, markeredgewidth=3)

plt.xlabel("특성 0")
plt.ylabel("특성 1")
```

Out[27]:

```
Text(0, 0.5, '특성 1')
```

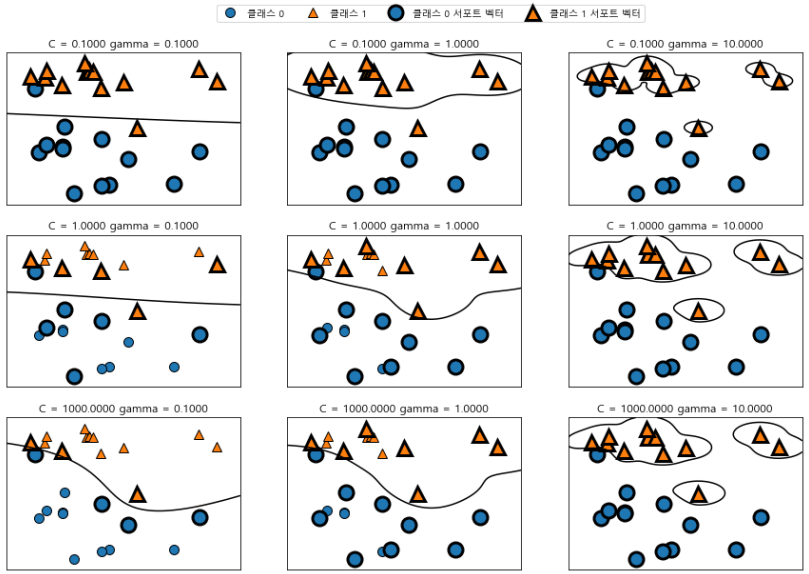


In [30]:

```
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(["클래스 0", "클래스 1",
                   "클래스 0 서포트 벡터", "클래스 1 서포트 벡터"], ncol=4,
                  loc=(.9, 1.2))
```

Out[30]:

<matplotlib.legend.Legend at 0x1370365c358>



In [31]:

```
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("훈련 세트 정확도: {:.2f}".format(svc.score(X_train, y_train)))
print("테스트 세트 정확도: {:.2f}".format(svc.score(X_test, y_test)))
```

훈련 세트 정확도: 1.00
테스트 세트 정확도: 0.63

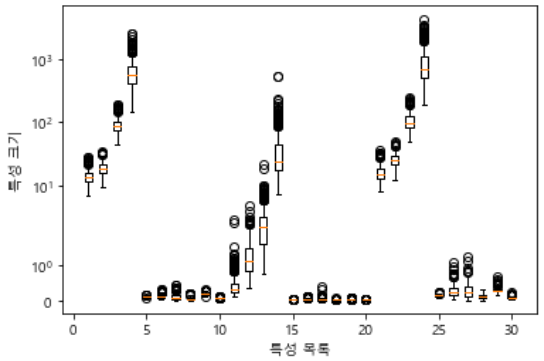
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩svm₩base.py:196: FutureWarning:
The default value of gamma will change from 'auto' to 'scale' in version 0.22 to a
ccount better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to
avoid this warning.
  "avoid this warning.", FutureWarning)

In [32]:

```
plt.boxplot(X_train, manage_xticks=False)
plt.yscale("symlog")
plt.xlabel("특성 목록")
plt.ylabel("특성 크기")
```

Out[32]:

Text(0, 0.5, '특성 크기')



In [33]:

```
min_on_training = X_train.min(axis=0)
range_on_training = (X_train - min_on_training).max(axis=0)

X_train_scaled = (X_train - min_on_training) / range_on_training

print("특성별 최솟값Wn{}".format(X_train_scaled.min(axis=0)))
print("특성별 최댓값Wn{}".format(X_train_scaled.max(axis=0)))
```

```
특성별 최솟값
[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.]
특성별 최댓값
[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.]
```

In [35]:

```
X_test_scaled = (X_test - min_on_training) / range_on_training

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

print("훈련 세트 정확도: {:.3f}".format(svc.score(X_train_scaled, y_train)))
print("테스트 세트 정확도: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

```
훈련 세트 정확도: 0.948
테스트 세트 정확도: 0.951

C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWsvmWbase.py:196: FutureWarning:
The default value of gamma will change from 'auto' to 'scale' in version 0.22 to a
ccount better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to
avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [36]:

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

print("훈련 세트 정확도: {:.3f}".format(svc.score(X_train_scaled, y_train)))
print("테스트 세트 정확도: {:.3f}".format(svc.score(X_test_scaled, y_test)))
```

```
훈련 세트 정확도: 0.988
테스트 세트 정확도: 0.972

C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWsvmWbase.py:196: FutureWarning:
The default value of gamma will change from 'auto' to 'scale' in version 0.22 to a
ccount better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to
avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [ ]:

```

```