In [36]:

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

In [37]:

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

In [38]:

```
from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname="C:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

In [39]:

```
#import matplotlib as plt

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

In [40]:

```
from sklearn.neighbors import KNeighborsRegressor
```

In [41]:

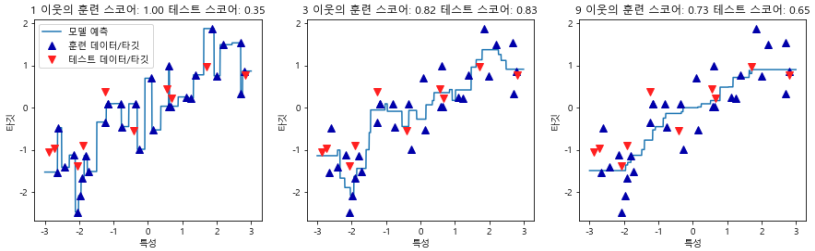
```
X, y = mglearn.datasets.make_wave(n_samples=40)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [42]:

```
fig, axes = plt.subplots(1, 3, figsize=(15, 4))
line = np.linspace(-3, 3, 1000).reshape(-1, 1)
for n_neighbors, ax in zip([1, 3, 9], axes):
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)
    ax.plot(line, reg.predict(line))
    ax.plot(X_train, y_train, '^', c=mglearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mglearn.cm2(1), markersize=8)

    ax.set_title("{} 이웃의 훈련 스코어: {:.2f} 테스트 스코어: {:.2f}".format(
    n_neighbors, reg.score(X_train, y_train), reg.score(X_test, y_test)))
    ax.set_xlabel("특성")
    ax.set_ylabel("타깃")
axes[0].legend(["모델 예측", "훈련 데이터/타깃", "테스트 데이터/타깃"], loc="best")
```

Out[42]:

<matplotlib.legend.Legend at 0x285d1999630>

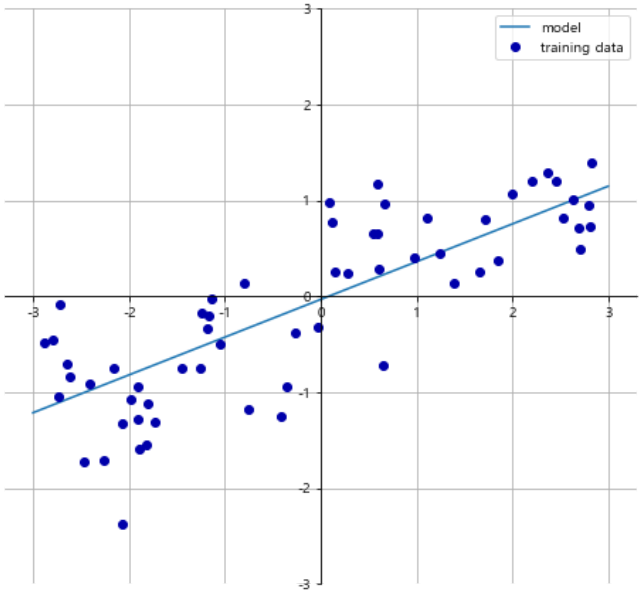


선형회귀

In [43]:

```
mglearn.plots.plot_linear_regression_wave()
```

```
w[0]: 0.393906  b: -0.031804
```



최소제곱법

In [44]:

```
from sklearn.linear_model import LinearRegression
X, y = mglearn.datasets.make_wave(n_samples=60)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

lr = LinearRegression().fit(X_train, y_train)
```

In [45]:

```
print("lr.coef_:", lr.coef_)
print("lr.intercept_:", lr.intercept_)
```

```
lr.coef_: [0.39390555]
lr.intercept_: -0.031804343026759746
```

In [46]:

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

```
훈련 세트 점수: 0.67
테스트 세트 점수: 0.66
```

In [47]:

```
X, y = mglearn.datasets.load_extended_boston()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
lr = LinearRegression().fit(X_train, y_train)
```

In [48]:

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

```
훈련 세트 점수: 0.95
테스트 세트 점수: 0.61
```

릿지 회귀

In [49]:

```
from sklearn.linear_model import Ridge

ridge = Ridge().fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(ridge.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(ridge.score(X_test, y_test)))
```

```
훈련 세트 점수: 0.89
테스트 세트 점수: 0.75
```

In [50]:

```
ridge10 = Ridge(alpha=10).fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(ridge10.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(ridge10.score(X_test, y_test)))
```

```
훈련 세트 점수: 0.79
테스트 세트 점수: 0.64
```

In [51]:

```
ridge01 = Ridge(alpha=0.1).fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(ridge01.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(ridge01.score(X_test, y_test)))
```

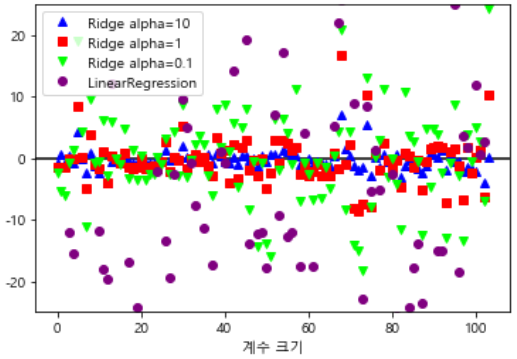
훈련 세트 점수: 0.93
테스트 세트 점수: 0.77

In [52]:

```
plt.plot(ridge10.coef_, '^', label="Ridge alpha=10", color='blue')
plt.plot(ridge.coef_, 's', label="Ridge alpha=1", color='r')
plt.plot(ridge01.coef_, 'v', label="Ridge alpha=0.1", color='lime')
plt.plot(lr.coef_, 'o', label="LinearRegression", color='purple')

plt.xlabel("계수 목록")
plt.xlabel("계수 크기")

xlims = plt.xlim()
plt.hlines(0, xlims[0], xlims[1])
plt.xlim(xlims)
plt.ylim(-25, 25)
plt.legend()
```

Out[52]:

<matplotlib.legend.Legend at 0x285d1a1bcf8>

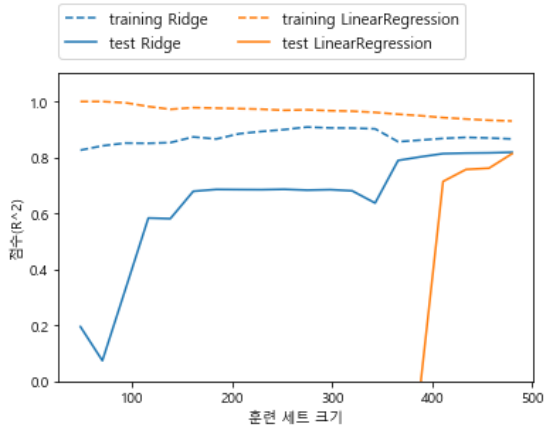


In [53]:

```
mglearn.plots.plot_ridge_n_samples()
plt.xlabel("훈련 세트 크기")
plt.ylabel("점수(R^2)")
```

Out[53]:

Text(0, 0.5, '점수(R^2)')



Lasso

In [54]:

```
from sklearn.linear_model import Lasso

lasso = Lasso().fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(lasso.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(lasso.score(X_test, y_test)))
print("사용한 특성의 개수:", np.sum(lasso.coef_ != 0))
print()

lasso001 = Lasso(alpha=0.01, max_iter=100000).fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(lasso001.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(lasso001.score(X_test, y_test)))
print("사용한 특성의 개수:", np.sum(lasso001.coef_ != 0))
print()

lasso00001 = Lasso(alpha=0.0001, max_iter=100000).fit(X_train, y_train)
print("훈련 세트 점수: {:.2f}".format(lasso00001.score(X_train, y_train)))
print("테스트 세트 점수: {:.2f}".format(lasso00001.score(X_test, y_test)))
print("사용한 특성의 개수:", np.sum(lasso00001.coef_ != 0))
```

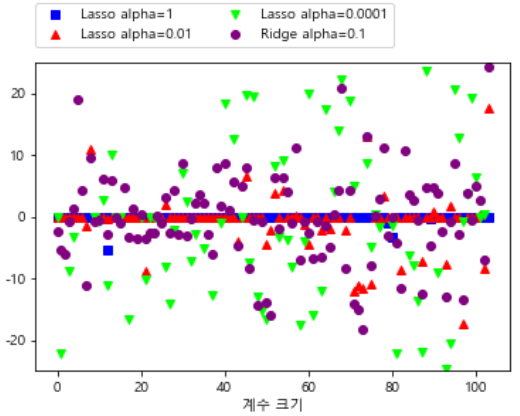
```
훈련 세트 점수: 0.29
테스트 세트 점수: 0.21
사용한 특성의 개수: 4

훈련 세트 점수: 0.90
테스트 세트 점수: 0.77
사용한 특성의 개수: 33

훈련 세트 점수: 0.95
테스트 세트 점수: 0.64
사용한 특성의 개수: 96
```

In [55]:

```
plt.plot(lasso.coef_, 's', label="Lasso alpha=1", color='blue')
plt.plot(lasso001.coef_, '^', label="Lasso alpha=0.01", color='r')
plt.plot(lasso00001.coef_, 'v', label="Lasso alpha=0.0001", color='lime')
plt.plot(ridge01.coef_, 'o', label="Ridge alpha=0.1", color='purple')

plt.xlabel("계수 목록")
plt.xlabel("계수 크기")
plt.ylim(-25, 25)
plt.legend(ncol=2, loc=(0, 1.05))
```

Out[55]:

```
<matplotlib.legend.Legend at 0x285d1ba2438>
```



In [56]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
```

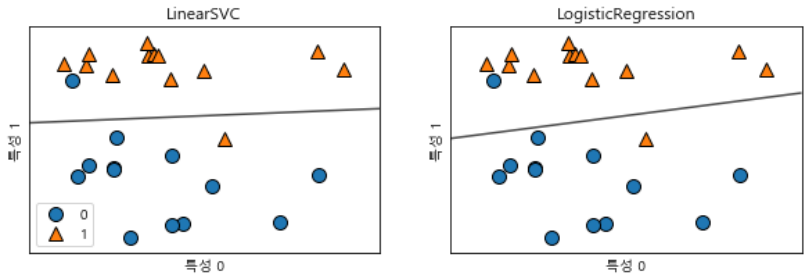
In [57]:

```
X, y = mglearn.datasets.make_forge()

fig, axes = plt.subplots(1, 2, figsize=(10, 3))

for model, ax in zip([LinearSVC(), LogisticRegression()], axes):
    clf = model.fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=False, eps=0.5, ax=ax, alpha=.7)
    mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
    ax.set_title(clf.__class__.__name__)
    ax.set_xlabel("특성 0")
    ax.set_ylabel("특성 1")
axes[0].legend()
```

```
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWutilsWdeprecation.py:77: Deprec
ationWarning: Function make_blobs is deprecated; Please import make_blobs directly
from scikit-learn
  warnings.warn(msg, category=DeprecationWarning)
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWsvmWbase.py:931: ConvergenceWar
ning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

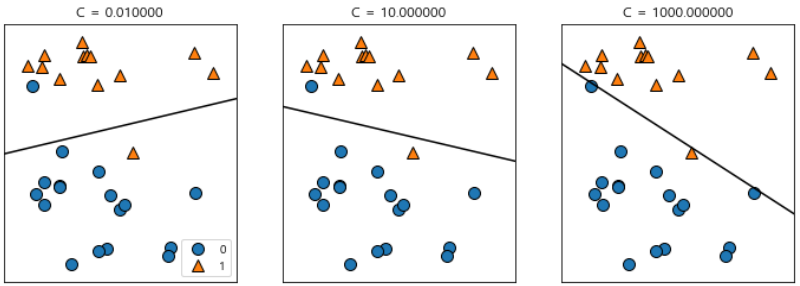
Out[57]:

```
<matplotlib.legend.Legend at 0x285d1b78780>
```



In [58]:

```
mglearn.plots.plot_linear_svc_regularization()
```



In [59]:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=42)
logreg = LogisticRegression().fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_train, y_train)))
print("훈련 세트 점수: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
훈련 세트 점수: 0.953
훈련 세트 점수: 0.958

C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [60]:

```
logreg100 = LogisticRegression(C=100).fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg100.score(X_train, y_train)))
print("훈련 세트 점수: {:.3f}".format(logreg100.score(X_test, y_test)))
```

```
훈련 세트 점수: 0.977
훈련 세트 점수: 0.965

C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [61]:

```
logreg001 = LogisticRegression(C=0.01).fit(X_train, y_train)
print("훈련 세트 점수: {:.3f}".format(logreg001.score(X_train, y_train)))
print("훈련 세트 점수: {:.3f}".format(logreg001.score(X_test, y_test)))
```

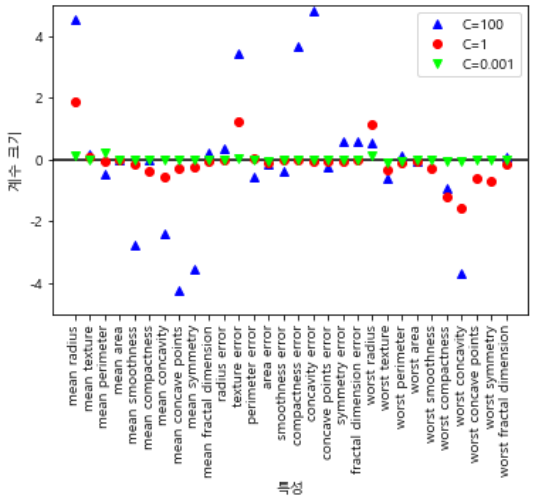
```
훈련 세트 점수: 0.934
훈련 세트 점수: 0.930

C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [62]:

```
plt.plot(logreg100.coef_.T, '^', label="C=100", color="blue")
plt.plot(logreg.coef_.T, 'o', label="C=1", color="red")
plt.plot(logreg001.coef_.T, 'v', label="C=0.001", color="lime")

plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)

xlims = plt.xlim()

plt.hlines(0, xlims[0], xlims[1])
plt.xlim(xlims)
plt.ylim(-5, 5)
plt.xlabel("특성")
plt.ylabel("계수 크기")
plt.legend()
```

Out[62]:

```
<matplotlib.legend.Legend at 0x285d0373400>
```



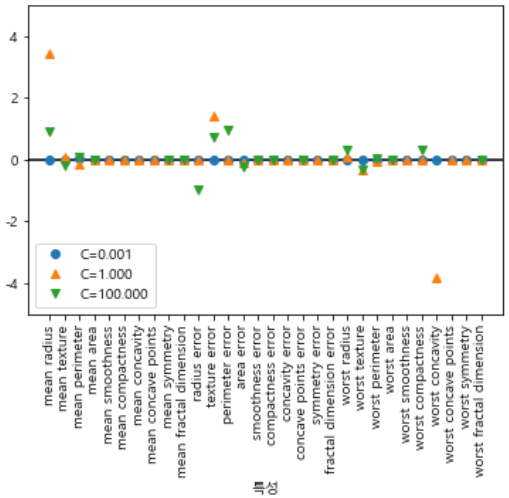
In [63]:

```
for C, marker, color in zip([0.001, 1, 100], ['o', '^', 'v'], ['blue','red','lime']):
    lr_l1 = LogisticRegression(C=C, penalty="l1").fit(X_train, y_train)
    print("C={:.3f}인 l1 로지스틱 회귀의 훈련 정확도: {:.2f}".format(
        C, lr_l1.score(X_train, y_train)))
    print("C={:.3f}인 l1 로지스틱 회귀의 테스트 정확도: {:.2f}".format(
        C, lr_l1.score(X_test, y_test)))
    plt.plot(lr_l1.coef_.T, marker, label="C={:.3f}".format(C))

plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)

xlims = plt.xlim()
plt.hlines(0, xlims[0], xlims[1])
plt.xlim(xlims)
plt.xlabel("특성")

plt.ylim(-5, 5)
plt.legend(loc=3)
```

```
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWsvmWbase.py:931: ConvergenceWar
ning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)

C=0.001인 l1 로지스틱 회귀의 훈련 정확도: 0.91
C=0.001인 l1 로지스틱 회귀의 테스트 정확도: 0.92
C=1.000인 l1 로지스틱 회귀의 훈련 정확도: 0.96
C=1.000인 l1 로지스틱 회귀의 테스트 정확도: 0.96
C=100.000인 l1 로지스틱 회귀의 훈련 정확도: 0.99
C=100.000인 l1 로지스틱 회귀의 테스트 정확도: 0.98
```

Out[63]:

```
<matplotlib.legend.Legend at 0x285d05705c0>
```



In [64]:

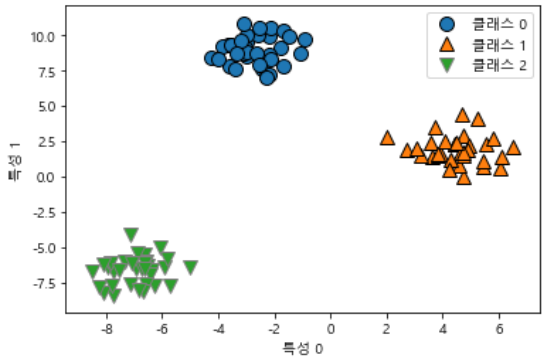
```
from sklearn.datasets import make_blobs
```

In [65]:

```
X,y = make_blobs(random_state=42)

mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("특성 0")
plt.ylabel("특성 1")
plt.legend(["클래스 0", "클래스 1", "클래스 2"])
```

Out[65]:

```
<matplotlib.legend.Legend at 0x285d2c11588>
```



In [66]:

```
linear_svm = LinearSVC().fit(X, y)
print("계수 배열의 크기: ", linear_svm.coef_.shape)
print("절편 배열의 크기: ", linear_svm.intercept_.shape)
```

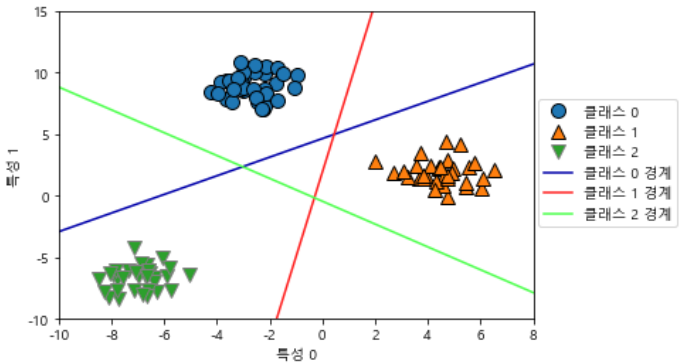
```
계수 배열의 크기:  (3, 2)
절편 배열의 크기:  (3,)
```

In [67]:

```
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
line = np.linspace(-15, 15)
for coef, intercept, color in zip(linear_svm.coef_, linear_svm.intercept_, mglearn.cm3.colors):
    plt.plot(line, -(line * coef[0] + intercept) / coef[1], c=color)

plt.ylim(-10, 15)
plt.xlim(-10, 8)
plt.xlabel("특성 0")
plt.ylabel("특성 1")
plt.legend(["클래스 0", "클래스 1", "클래스 2", "클래스 0 경계", "클래스 1 경계", "클래스 2 경계"], loc=(1.01, 0.3))
```

Out[67]:

<matplotlib.legend.Legend at 0x285d2c8ae48>

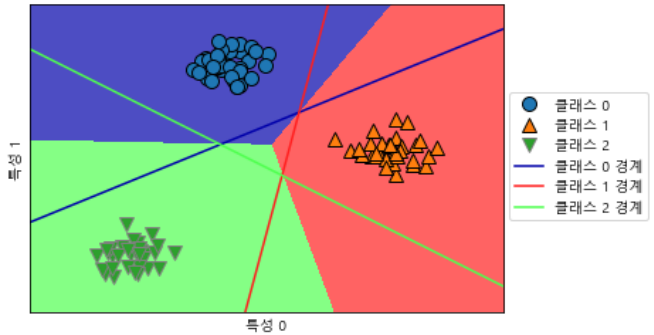


In [68]:

```
mglearn.plots.plot_2d_classification(linear_svm, X, fill=True, alpha=.7)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
line = np.linspace(-15, 15)
for coef, intercept, color in zip(linear_svm.coef_, linear_svm.intercept_, mglearn.cm3.colors):
    plt.plot(line, -(line * coef[0] + intercept) / coef[1], c=color)

plt.xlabel("특성 0")
plt.ylabel("특성 1")
plt.legend(["클래스 0", "클래스 1", "클래스 2", "클래스 0 경계", "클래스 1 경계", "클래스 2 경계"], loc=(1.01, 0.3))
```

Out[68]:

<matplotlib.legend.Legend at 0x285d2ce9828>



In [69]:

```
logreg = LogisticRegression().fit(X_train, y_train)
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [70]:

```
logreg = LogisticRegression()
y_pred = logreg.fit(X_train, y_train).predict(X_test)
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩sklearn₩linear_model₩logistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [71]:

```
y_pred = LogisticRegression().fit(X_train, y_train).predict(X_test)
```

```
C:WProgramDataWAnaconda3WlibWsite-packagesWsklearnWlinear_modelWlogistic.py:433: F
utureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

In [ ]:

```

```