다중 클래스 분류용 선형 모델

학습 내용

- 3개 이상의 클래스를 갖는 경우의 분류 알고리즘에 대해 알아봅니다.

- 많은 선형 분류 모델은 태생적으로 이진 분류만을 지원. 다중 클래스를 지원안함.
- 이진 분류를 **다중 클래스 분류 알고리즘으로 확장하는 보편적인 기법은 일대다방법**입니다.
- 클래스의 수만큼 이진 분류 모델을 만들게 된다.
- 예측을 할때 이렇게 만들어진 **모든 이진 분류기가 작동하여 가장 높은 점수를 내는 분류기의 예측값**으로 선택

한글 표시

In [1]:

```
# 한글
import matplotlib
from matplotlib import font_manager, rc
font_loc = "C:/Windows/Fonts/malgunbd.ttf"
font_name = font_manager.FontProperties(fname=font_loc).get_name()
matplotlib.rc('font', family=font_name)
matplotlib.rcParams['axes.unicode_minus'] = False

%matplotlib inline
```

01 세 개의 클래스를 가진 간단한 데이터 셋

- 출력 : 100개, 클래스 값(0, 1, 2)
- 입력 : 100개, 2개열

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import mglearn
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
```

In [3]:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(random_state=42)
print(X.shape, y.shape)
print(X[:5], y[:5])
```

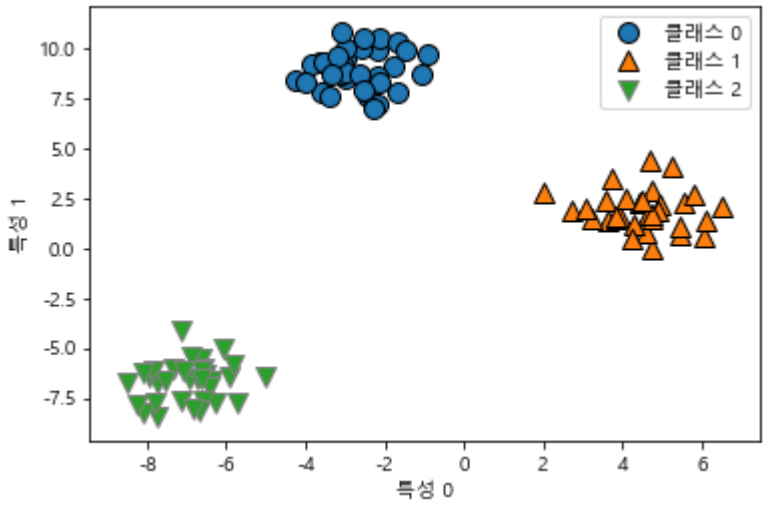
```
(100, 2) (100,)
[[-7.72642091 -8.39495682]
 [ 5.45339605  0.74230537]
 [-2.97867201  9.55684617]
 [ 6.04267315  0.57131862]
 [-6.52183983 -6.31932507]] [2 1 0 1 2]
```

In [4]:

```
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[4]:

```
<matplotlib.legend.Legend at 0x2421e7563d0>
```



02 LinearSVC 분류기로 학습을 진행

In [5]:

```
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,)
```

- coef_ 배열의 크기는 (3,2) coef_의 행은 세 개의 클래스에 각각 대응하는 계수 벡터를 담고 있음.

In [6]:

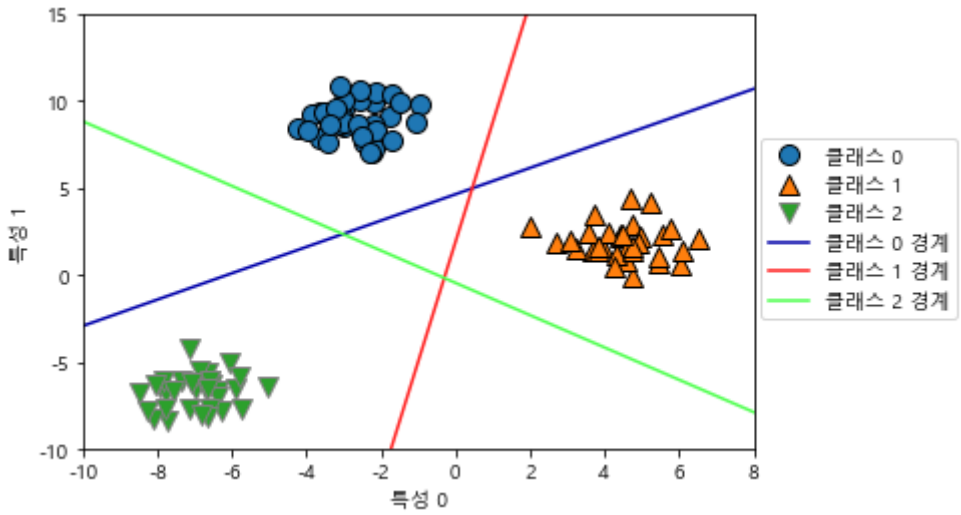
```
import numpy as np
```

In [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.ylim(-10, 15)
plt.xlim(-10, 8)
plt.xlabel("특성 0")
plt.ylabel("특성 1")

label_list = ['클래스 0', '클래스 1', '클래스 2',
              '클래스 0 경계', '클래스 1 경계', '클래스 2 경계']

plt.legend(label_list, loc=(1.01, 0.3))
```

Out[7]:

```
<matplotlib.legend.Legend at 0x2421ef150d0>
```

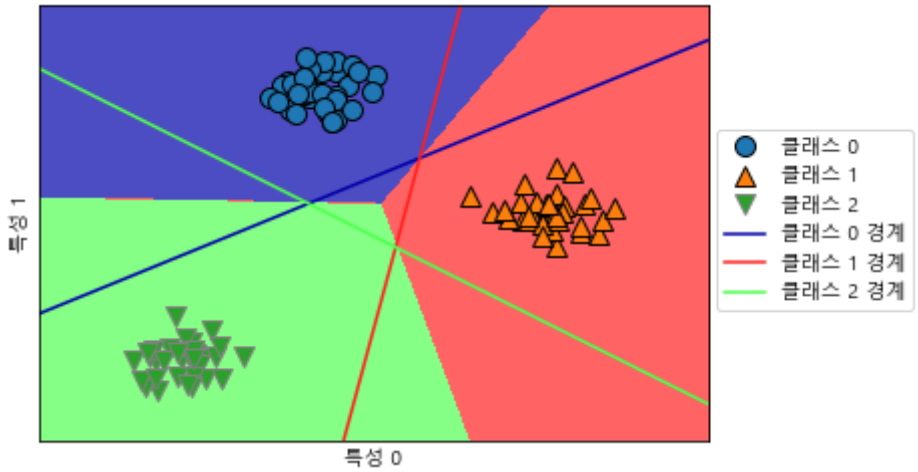


In [8]:

```
mglearn.plots.plot_2d_classification(linear_svm, X, fill=True, alpha=0.7)
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)


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

Out[8]:

```
Text(0, 0.5, '특성 1')
```



장단점과 매개변수

- 회귀 모델에서는 alpha가 주요 매개변수
- LinearSVC와 LogisticRegression에서는 C가 매개변수
- alpha값이 클수록, C값이 작을수록 모델이 단순해 집니다.(규제가 많다의 의미)
- 중요한 특성이 많지 않다고 한다면 L1규제 사용, 그렇지 않으면 기본적으로 L2규제를 사용.

선형 모델

- 선형모델의 대용량 처리 버전으로 SGDClassifier와 SGDRegressor를 사용 가능
- 수십만 수백만 개의 샘플로 이루어진 것일 때, 기본설정으로 빨리 처리하도록 LogisticRegresion과 Ridge에 solver='sag'옵션
- 선형 모델은 샘플에 비해 특성이 많을 때 잘 작동
- 다만, 저차원 데이터셋에서는 다른 모델들이 일반화 성능이 좋음.

history

- Machine Learning with sklearn @ DJ,Lim

- date : 2020/10/22 -- version 02