Ch2. Support Vector Machine (SVM)

November 3, 2021

1 Support Vector Classifier Algorithm

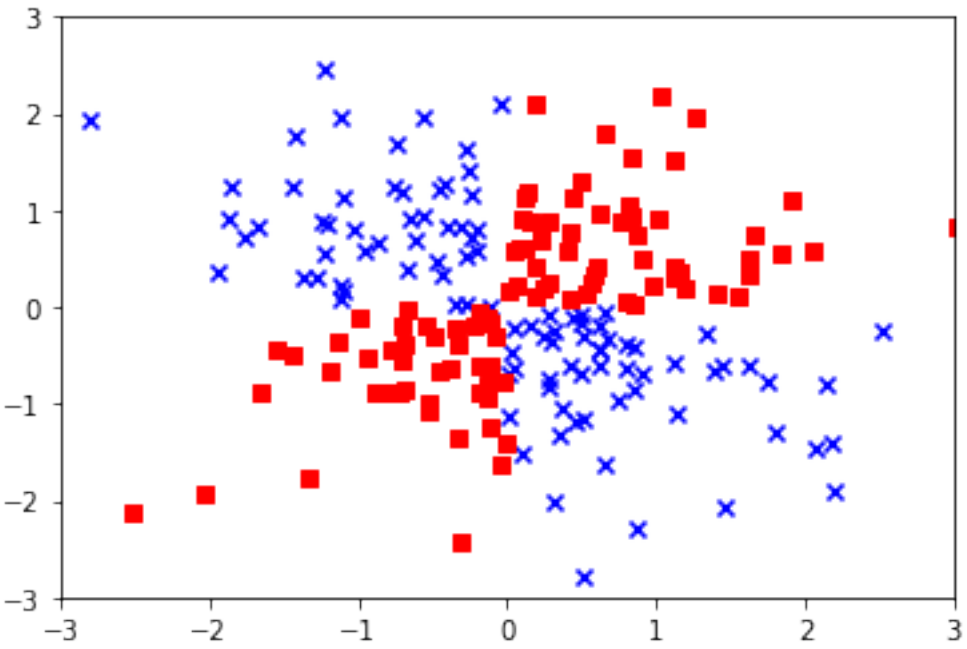
- **sklearn.svm 모듈의 SVC (서포트벡터분류기) 클래스**
  - *sklearn.svm.SVC(, C=1.0, kernel='rbf', degree=3, gamma='scale',...)**
  - 입력 파라미터
    - kernel : 커널함수 지정
      - linear: 선형 SVM
      - poly: 다항 커널 함수 사용, 비선형 SVM
      - rbf: RBF (Radial Basis Function (방사 기저 함수)) 커널 함수 사용, 비선형 SVM, 디폴트
    - C : 규제강도 하이퍼 파라미터
      - 클수록 하드마진. 마진이 작아짐. 오류를 허용하는 정도가 작음. 과대적합
      - 작을수록 소프트마진. 마진이 커짐. 오류 허용 정도가 큼. 과대적합 방지.
    - gamma : 커널 계수.
      - 커널 계수로 하나의 훈련 샘플이 미치는 영향의 범위를 결정함.
      - 클수록 커널폭이 좁아져 구불구불한 결정경계를 가짐. 과대적합.
      - 작을수록 커널폭이 넓어져 직선에 가까운 결정경계 가짐. 과대적합 방지.
      - 비선형 SVM에서만 이 옵션 사용되며, RBF의 경우 가우시안 커널 폭의 역수에 해당.
    - degree : 다항커널의 차수.
      - default는 3.
      - 다항 커널 이외의 다른 커널 이용시 이 옵션 무시됨.
  - 메서드
    - fit (X_train,y_train) 메서드로 모델을 학습.
    - predict (X_test) 메서드로 새로운 입력데이터에 학습된 모델을 적용한 예측.
  - 속성
    - support_ : 서포트 벡터의 인덱스
    - support_vectors_ : 서포트 벡터 (n_SV, n_features)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
X_xor = np.random.randn(200, 2)
y_xor = np.logical_xor(X_xor[:, 0] > 0, X_xor[:, 1] > 0)
tmp = np.hstack( [X_xor, y_xor.reshape(-1,1)] )
print( tmp[:10] )
```

```
[[ 1.62434536 -0.61175641  1.        ]
```

```
[-0.52817175 -1.07296862  0.        ]
[ 0.86540763 -2.3015387   1.        ]
[ 1.74481176 -0.7612069   1.        ]
[ 0.3190391  -0.24937038  1.        ]
[ 1.46210794 -2.06014071  1.        ]
[-0.3224172  -0.38405435  0.        ]
[ 1.13376944 -1.09989127  1.        ]
[-0.17242821 -0.87785842  0.        ]
[ 0.04221375  0.58281521  0.        ]]
```

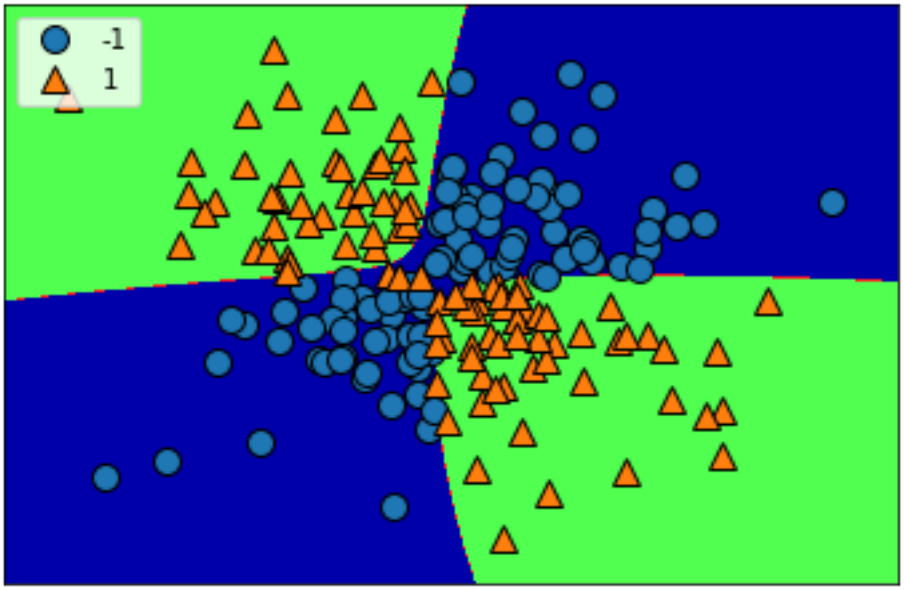
[2]:
```
y_xor = np.where(y_xor, 1, -1)
plt.scatter(X_xor[y_xor == 1, 0], X_xor[y_xor == 1, 1],
            c='b', marker='x', label='1')
plt.scatter(X_xor[y_xor == -1, 0], X_xor[y_xor == -1, 1],
            c='r', marker='s', label='-1')
plt.xlim([-3, 3])
plt.ylim([-3, 3])
plt.show()
```



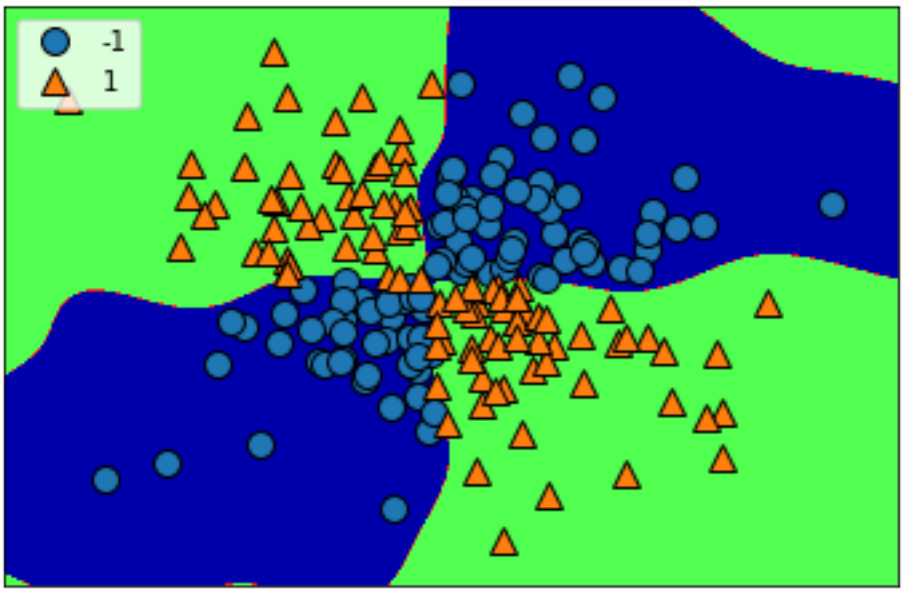
[3]:
```
from sklearn.svm import SVC
svm = SVC( kernel='rbf', random_state=1, gamma=0.10, C=100.0 )
svm.fit( X_xor, y_xor )
```

[3]:
```
SVC(C=100.0, gamma=0.1, random_state=1)
```

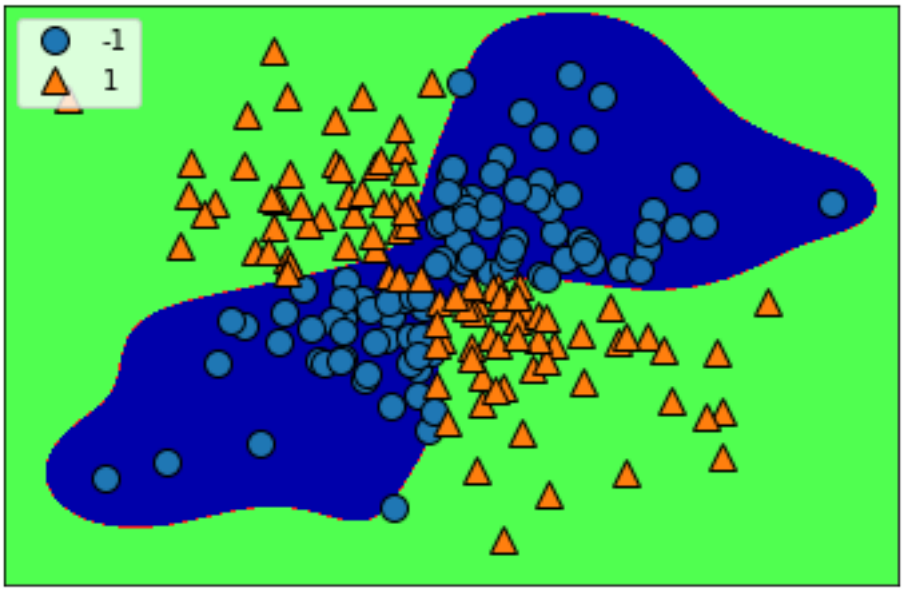
[4]:
```
import mglearn
mglearn.plots.plot_2d_classification(svm, X_xor)
mglearn.discrete_scatter(X_xor[:,0], X_xor[:,1], y_xor)
plt.legend(loc='upper left')
plt.show()
```



[5]:
```
svm = SVC(kernel='rbf', random_state=1, gamma=2.0, C=100.0)
svm.fit(X_xor, y_xor)
mglearn.plots.plot_2d_classification(svm, X_xor)
mglearn.discrete_scatter(X_xor[:,0], X_xor[:,1], y_xor)
plt.legend(loc='upper left')
plt.show()
```



[6]:
```
svm = SVC(kernel='rbf', random_state=1, gamma=2.0, C=0.01)
svm.fit(X_xor, y_xor)
mglearn.plots.plot_2d_classification(svm, X_xor)
mglearn.discrete_scatter(X_xor[:,0], X_xor[:,1], y_xor)
plt.legend(loc='upper left')
plt.show()
```



1.1 resign 데이터에 대한 SVM 적용 사례

- 어느 회사의 인사관리에 관한 사례
- 타겟 변수
  - regisn : 각 직원의 퇴사여부(0: 퇴사하지 않음, 1:퇴사함)
- 특성 변수
  - satisfaction : 업무만족도
  - evaluation : 인사평점
  - project : 프로젝트 수행횟수
  - workhour : 월평균업무시간
  - years : 근속연수
  - accident : 사고 여부
  - promotion : 최근 5년 이내 승진여부

[7]:
```
import pandas as pd
resign_df = pd.read_csv("resign.csv")
resign_df.head()
```

[7]:

	satisfaction	evaluation	project	workhour	years	accident	resign	promotion	good
0	0.38	0.53	2	157	3	0	1	0	0
1	0.80	0.86	5	262	6	0	1	0	1
2	0.11	0.88	7	272	4	0	1	0	1
3	0.72	0.87	5	223	5	0	1	0	1
4	0.37	0.52	2	159	3	0	1	0	0

[8]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
scaler = StandardScaler()
resign_df_X = scaler.fit_transform( resign_df[['evaluation','workhour']] )
X_tr, X_ts, y_tr, y_ts = train_test_split(
    resign_df_X, resign_df['resign'], test_size=0.2, random_state=0)
```

[9]:
```
svm = SVC(kernel='rbf', random_state=1, gamma=5, C=0.1 )
svm.fit( X_tr, y_tr )
```

[9]:
```
SVC(C=0.1, gamma=5, random_state=1)
```

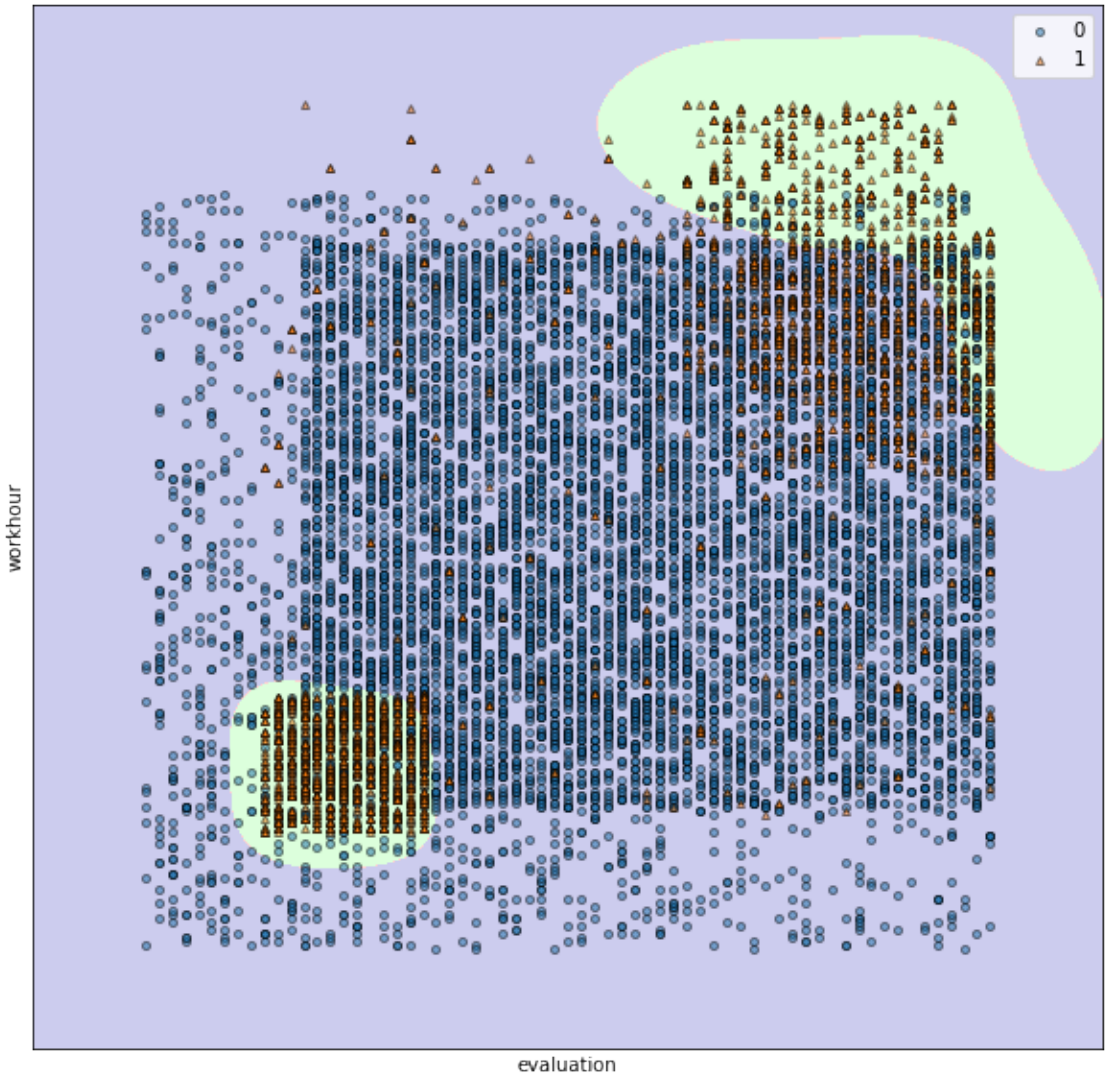
[10]:
```
svm.support_
```

[10]:
```
array([    8,    12,    16, ..., 11965, 11975, 11982])
```

[11]:
```
svm.support_vectors_
```

[11]:
```
array([[ 1.60021507,  1.14032895],
       [ 1.42494396,  0.73985987],
       [ 1.54179137,  1.4206573 ],
       ...,
       [ 1.07440174,  1.32054003],
       [-1.55466492, -0.94211024],
       [-1.02885159, -0.98215715]])
```

[12]:
```
X_tr_arr= np.array(X_tr)
y_tr_arr= np.array(y_tr)
plt.figure(figsize=(10, 10))
mglearn.plots.plot_2d_classification(svm, X_tr_arr, fill=True, alpha=0.2)
mglearn.discrete_scatter(X_tr_arr[:,0], X_tr_arr[:,1], y_tr_arr, alpha=0.5, s=4)
plt.legend()
plt.xlabel('evaluation')
plt.ylabel('workhour')
plt.show()
```



[13]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

[14]:
```
y_pred = svm.predict( X_ts )
```

[15]:
```
confusion_matrix ( y_ts, y_pred )
```

[15]:
```
array([[2139,  160],
       [ 262,  439]], dtype=int64)
```

[16]:
```
accuracy_score ( y_ts, y_pred )
```

[16]:
```
0.8593333333333333
```