Chap. 5 Support Vector Machine

2022-05-18

Yeongcheol Kim

5.1 선형 SVM 분류

- 직선으로 구분하는 결정경계
- 선형 SVM 분류는 적절하면서도 훈련샘플에서 가능한 한 멀리 떨어지도록 하는 결정경계를 선택
- 라지마진분류(Large Margin Classification, LMC): 가장 넓은 도로를 찾아내는 일
- 서포트 벡터(Support Vector): 결정경계를 결정하는데 영향을 미치는 Sample
  * 도로 밖의 샘플은 결정경계에 영향을 미치지 않음
- 스케일의존성: Standard Scaler를 이용하여 결정경계를 만드는 것을 추천

그림 5-1. 라지 마진 분류

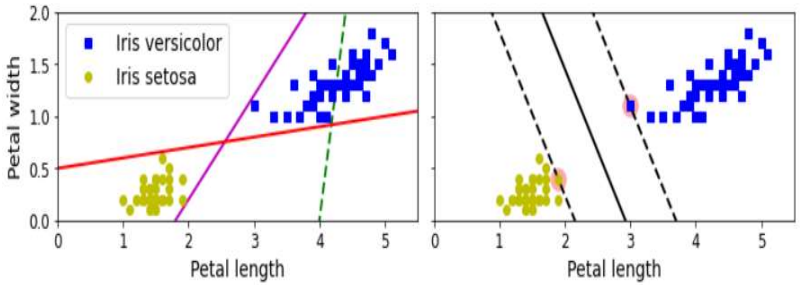
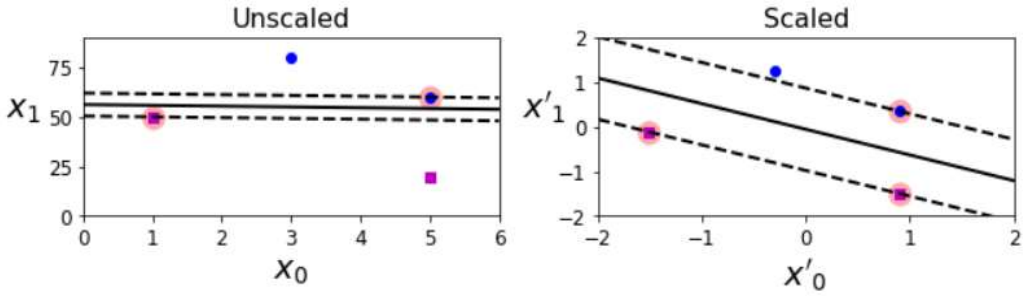


그림 5-2. 특성스케일에 따른 민감성



5.1.1 소프트마진분류

- 하드마진분류(Hard Margin Classification): 모든 샘플이 도로 바깥쪽에 분류된 경우
  > 반드시 데이터가 선형적으로 분류될 수 있어야 함.
  > 이상치에 민감
- 소프트마진분류(Soft Margin Classification): 보다 유연한 모델
  > 도로폭을 넓게 유지하면 , 일반화가 용이하나 마진오류(Margin Violation)가 증가
  > 도로폭을 좁게 유지하면 이상치에 더욱 민감해짐
  > 하이퍼파라미터 지정하여 도로폭을 넓게 유지하는 것과 마진오류 사이의 균형을 잡음.

그림 5-3. 이상치에 민감한 하드마진

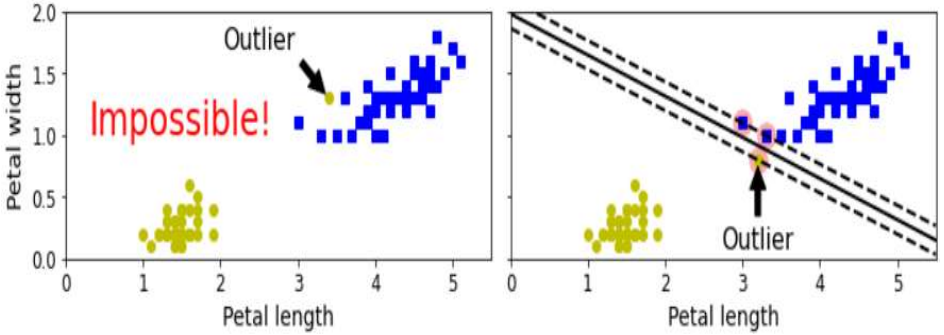
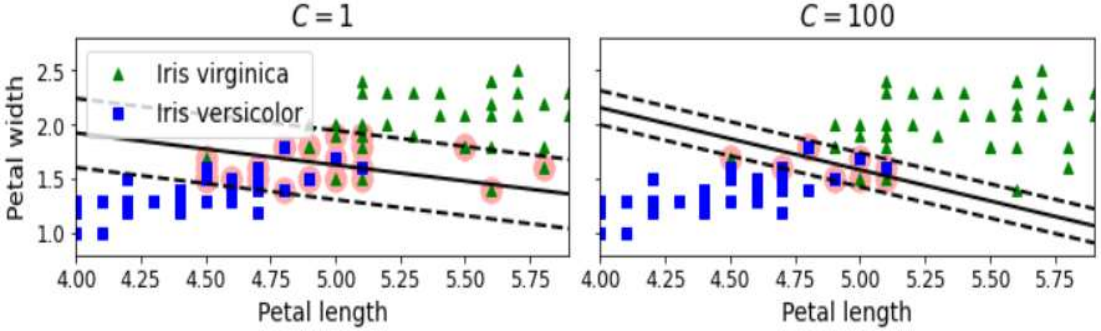


그림 5-4. 좁은 마진과 넓은 마진



[예제] 붓꽃 데이터 훈련 및 예측

- 특성 스케일 변경
- 선형 SVM 모델 훈련
  > Linear SVC class 사용
  > C=1
  > Hinge Loss Function 적용

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # 꽃잎 길이, 꽃잎 너비
y = (iris["target"] == 2).astype(np.float64)  # Iris virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge", random_state=42)),
    ])

svm_clf.fit(X, y)
```

```
Pipeline(steps=[('scaler', StandardScaler()),
                ('linear_svc', LinearSVC(C=1, loss='hinge', random_state=42))])
```

```
[6]  svm_clf.predict([[5.5, 1.7]])
```

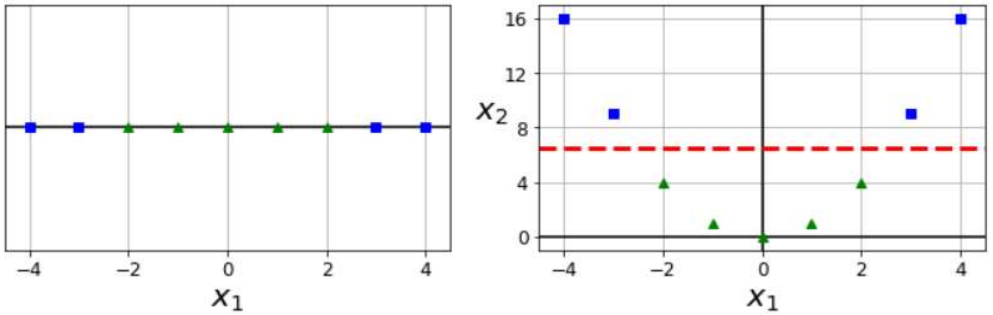
```
array([1.])
```

5.2 비선형 SVM 분류

- 선형적으로 분류할 수 없는 데이터셋
- 다항 특성과 같은 특성을 더 추가함 (4장 참조)

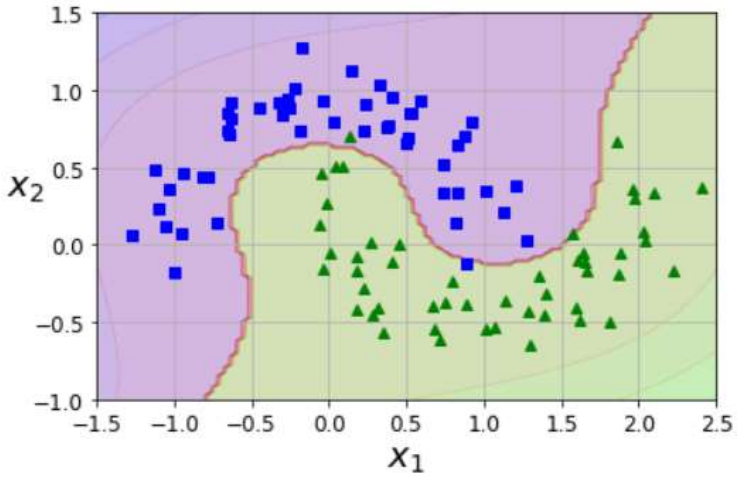
(예) 특성을 추가하여 선형적으로 구분되는 데이터셋 만들기
[X1]→ [X1,X2] where X2 = X1 * X1

그림 5-5. 특성을 추가하여 선형적으로 구분되는 데이터셋 만들기



(예) 다항특성을 사용한 선형 SVM 분류기의 사이킷런 구현
Pipeline: PolynomialFeatures변환기
+ StandardScaler + LinearSVC

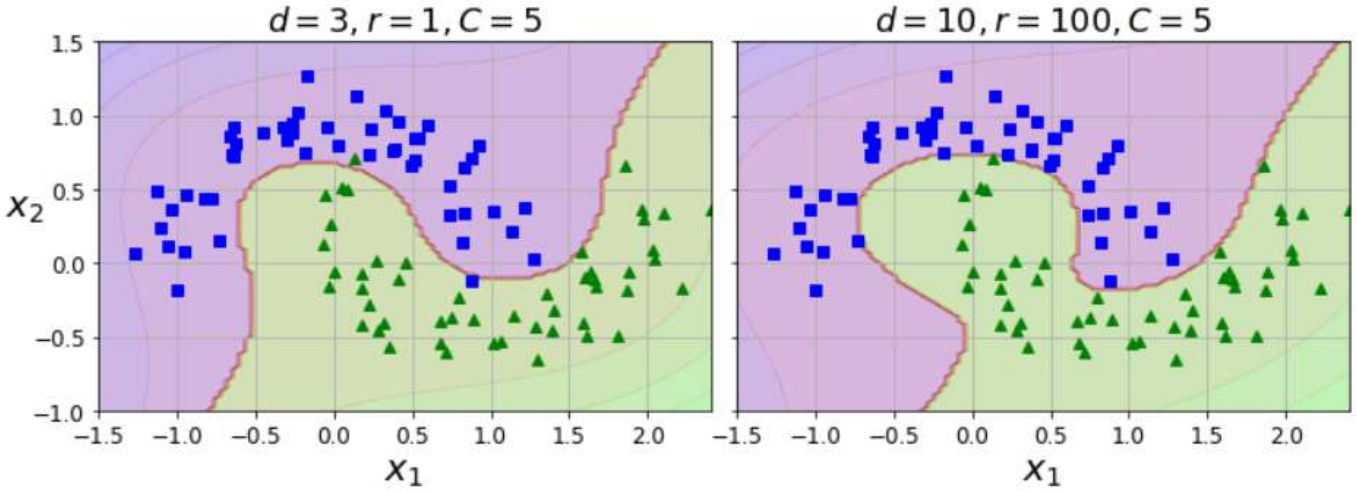
그림 5-6. 다항 특성을 사용한 선형 SVM 분류기



5.2.1 다항식 커널

- 다항식특성 추가
  > 낮은 차수: 복잡한 데이터셋을 표현하기 어려움
  > 높은 차수: 모델을 느려지게 함
- 커널 트릭(Kernel Trick): 수학적 기교
  > 실제로 특성을 추가하지 않지만, 다항식 특성을 많이 추가한 것과 같은 결과
  > 엄청난 수의 특성조합이 생기지 않음
  > 과대적합/과소적합의 문제

그림 5-7. 다항식 커널을 사용한 SVM 분류기



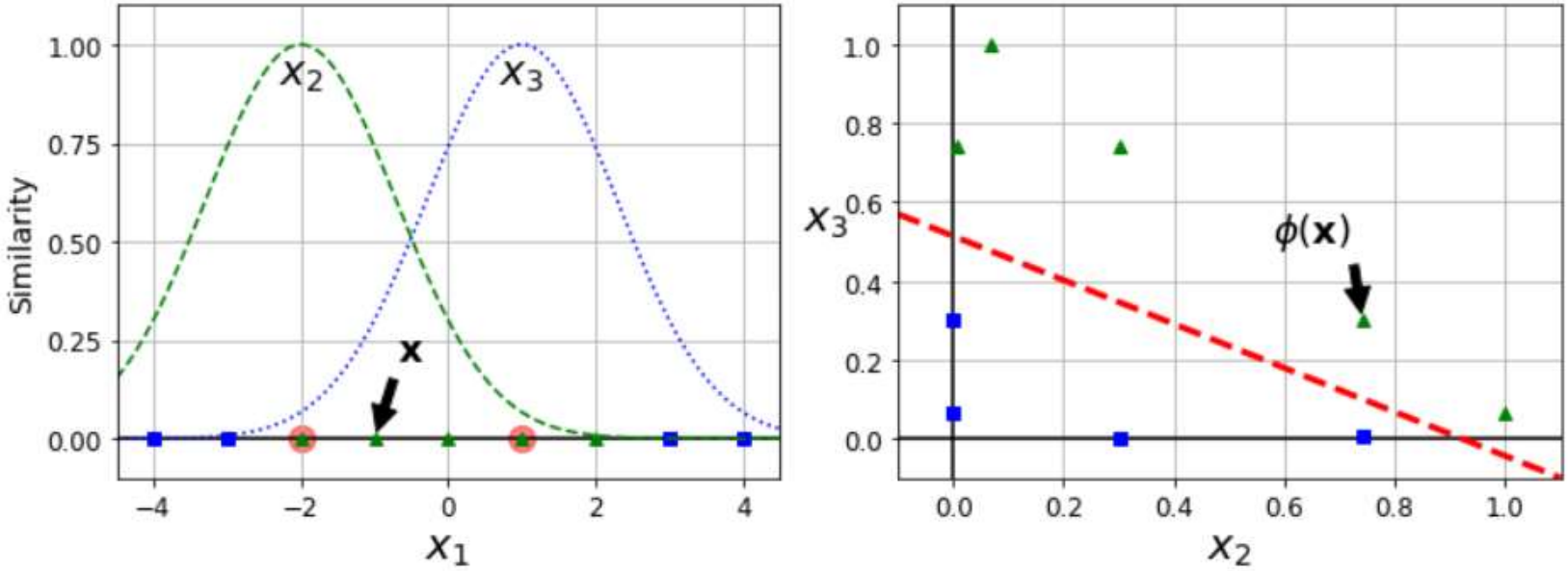
5.2.2 유사도 특성

- 특정 Landmark와 얼마나 닮았는지를 표현하는 Similarity Function 도입하여 비선형 특성을 다룸
- 유사도 함수로 계산된 특성값을 추가

식 5-1: 가우시안 RBF

$$\phi_\gamma(\mathbf{x}, \boldsymbol{\ell}) = \exp(-\gamma\|\mathbf{x} - \boldsymbol{\ell}\|^2)$$

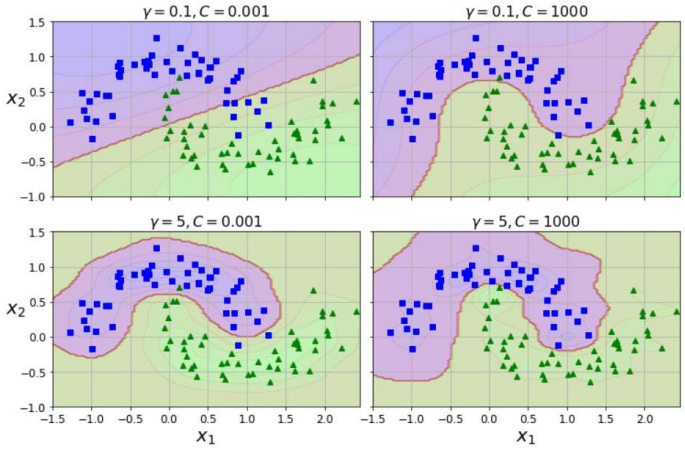
그림 5-8. Gaussian RBF(Radial Basis Function) 특성을 사용한 유사도 특성



5.2.3 가우시안 RBF 커널

- 훈련세트가 커질 경우, 유사도 측정 추가특성을 계산하려면 많은 연산비용
- Kernel Trick 적용하여 유사도 측정 특성을 추가한 것과 유사한 효과를 얻음

그림 5-9. RBF 커널을 사용한 SVM 분류기



5.2.3 계산 복잡도(표 5-1. SVM 분류를 위한 사이킷런 파이썬 클래스 비교)

파이썬클래스	시간복잡도	외부메모리학습지원	스케일조정의 필요성	커널트릭
LinearSVC	O(mxn)	No	Yes	No
SGDClassifier	O(mxn)	Yes	Yes	No
SVC	O(m2xn)~O(m3xn)	No	Yes	Yes

5.3 SVM 회귀

- SVM 알고리즘은 선형/비선형 분류 뿐 아니라, 선형/비선형 회귀에도 활용 가능
- SVM 분류: 일정한 마진 오류 안에서 두 클래스간의 도로폭을 최대화
- SVM 회귀: 제한된 마진 오류 안에서 도로 안에 들어가는 샘플수를 최대화
  > 도로의 폭은 하이퍼파라미터 ε으로 조절

그림 5-10. 선형 SVM 회귀

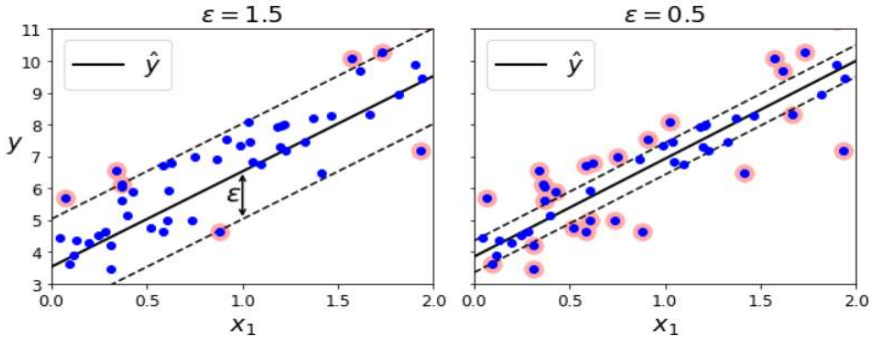


그림 5-11. 2차 다항커널을 사용한 SVM 회귀