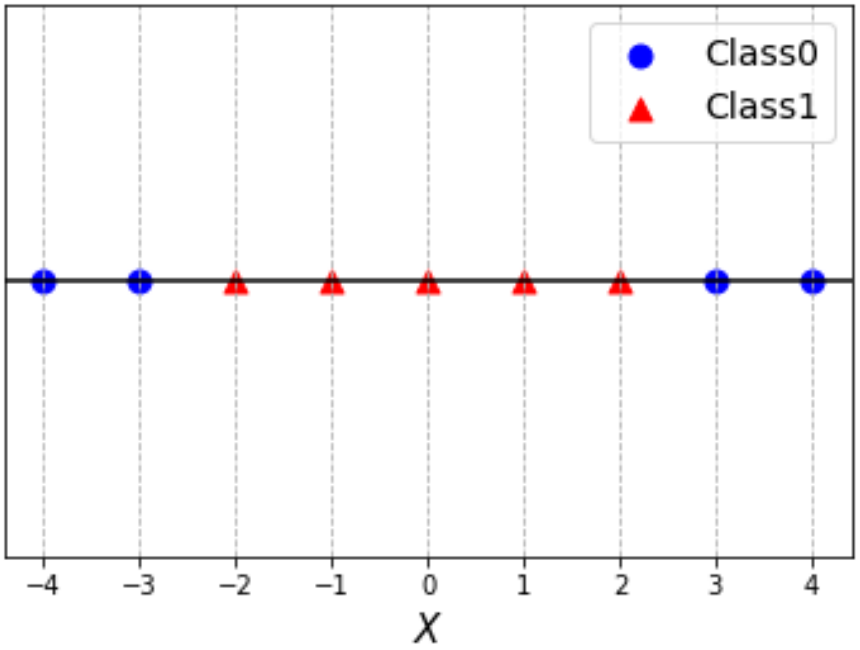
비선형 SVM

다항 특성 확장

비선형 SVM

- 선형 SVM 분류의 한계
  - 모든 데이터들을 선형적으로 분류할 수 있는 것은 아니다.

```
1  import numpy as np
2
3  X = np.linspace(-4, 4, 9).reshape(-1, 1)
4  y = np.array([0, 0, 1, 1, 1, 1, 1, 0, 0])
```



비선형 SVM

- 비선형 데이터의 특성 확장
  - 비선형 데이터의 특성을 다항식 형태로 변환하면 선형적으로 구분이 가능하게 될 수 있다.

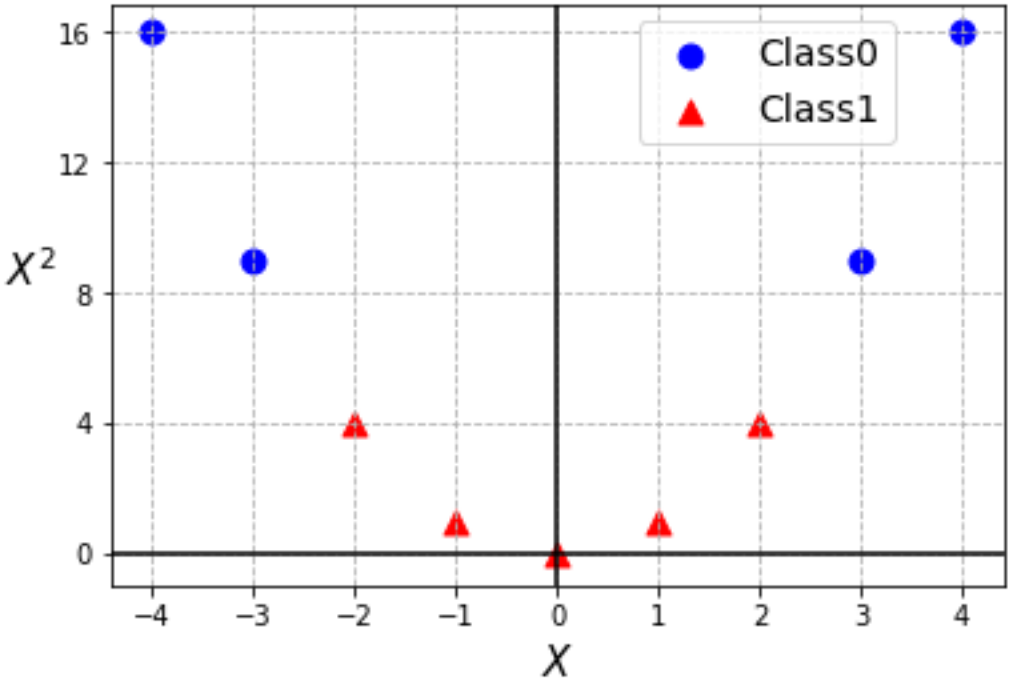
```
1  import numpy as np
2  import sklearn.preprocessing as pp
3
4  X = np.linspace(-4, 4, 9).reshape(-1, 1)
5  y = np.array([0, 0, 1, 1, 1, 1, 1, 0, 0])
6
7  poly = pp.PolynomialFeatures(degree=2)
8  X_poly = poly.fit_transform(X)
```

```
1  print(X_poly)
```

```
[[ 1. -4. 16.]
 [ 1. -3.  9.]
 [ 1. -2.  4.]
 [ 1. -1.  1.]
 [ 1.  0.  0.]
 [ 1.  1.  1.]
 [ 1.  2.  4.]
 [ 1.  3.  9.]
 [ 1.  4. 16.]]
```

비선형 SVM

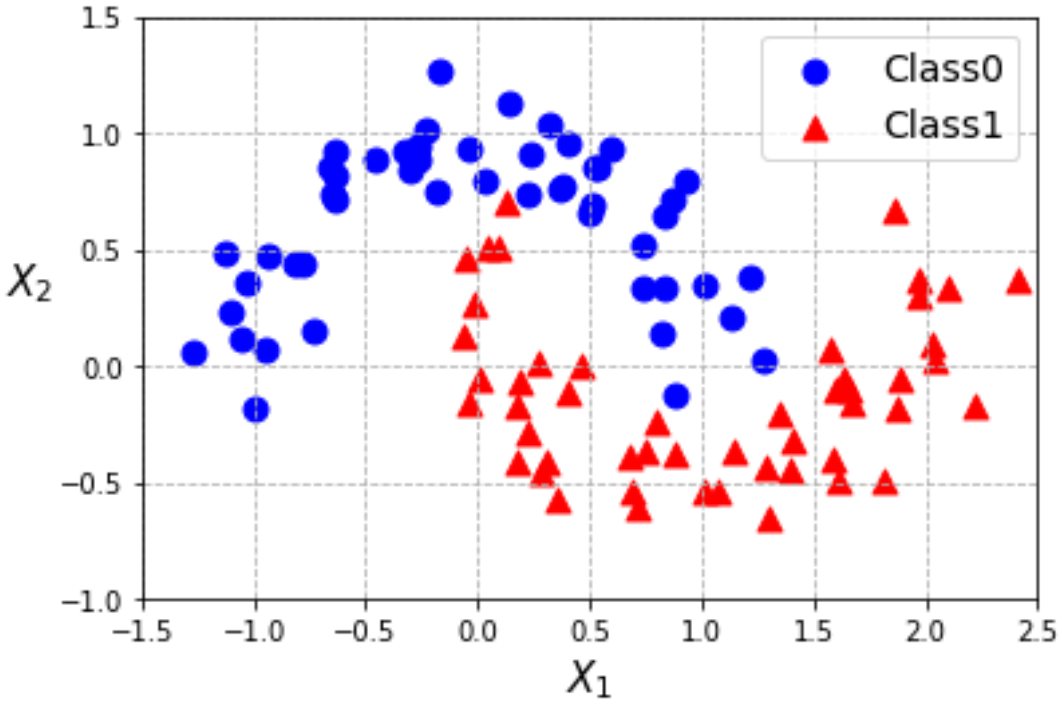
- 비선형 데이터의 특성 확장
  - 비선형 데이터의 특성을 다항식 형태로 변환하면 선형적으로 구분이 가능하게 될 수 있다.



비선형 SVM

- 비선형 데이터의 특성 확장 후 선형 SVM 분류
  - 다음과 같은 비선형 데이터에 대해서 선형 SVM을 수행한다.

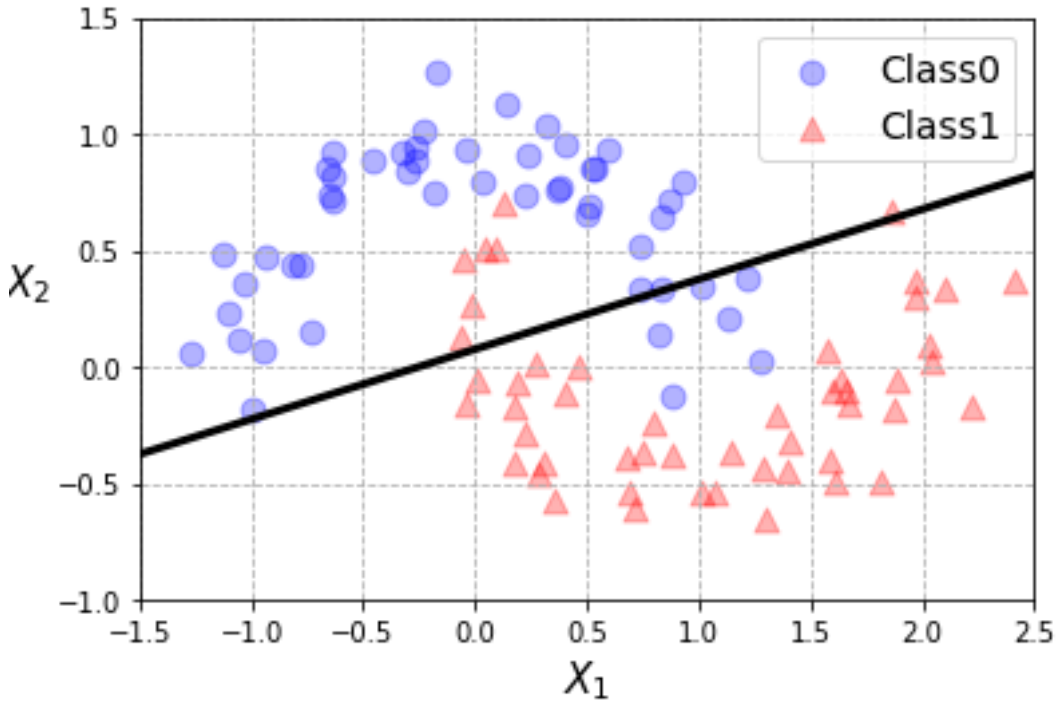
```
1  import sklearn.datasets as d
2
3  X, y = d.make_moons(n_samples=100, noise=0.15, random_state=42)
```



비선형 SVM

- 비선형 데이터의 특성 확장 후 선형 SVM 분류
  - 비선형 데이터를 그대로 이용하여 분류한 결과는 좋지 않다.

```
1  import sklearn.svm as svm
2
3  clf = svm.SVC(C=10, kernel="linear").fit(X, y)
```



비선형 SVM

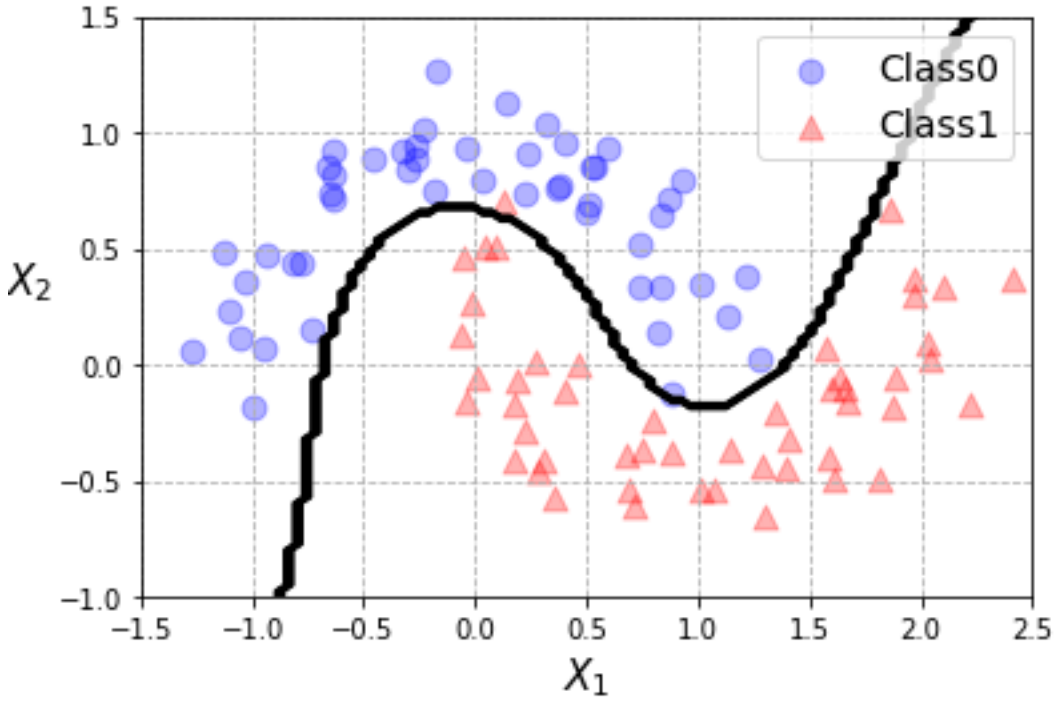
- 비선형 데이터의 특성 확장 후 선형 SVM 분류
  - 데이터의 특성을 다항식 형태로 변환한 뒤, 선형 SVM 분류를 수행한다.

```
1  import sklearn.preprocessing as pp
2
3  poly = pp.PolynomialFeatures(degree=3)
4  X_poly = poly.fit_transform(X)
```

```
1  import sklearn.svm as svm
2
3  clf = svm.SVC(C=10, kernel="linear").fit(X_poly, y)
```

비선형 SVM

- 비선형 데이터의 특성 확장 후 선형 SVM 분류
  - 다항식 형태로 변환된 데이터에 대해 선형 SVM 분류를 수행한 결과, 적절한 결정 경계가 나타난 것을 확인할 수 있다.



비선형 SVM

- 결정 경계와 예측값의 관계
  - 예측값은 **predict** 메소드를 호출하여 반환되는 결과이며, 이진 분류에서는 0 또는 1로 도출된다.
  - 결정 경계에 대해 적용한 값은 **decision_function** 메소드를 호출하여 반환되는 결과이며, 그 값의 부호에 따라 예측값이 결정된다.
    - 결과 값이 양수이면 결정 경계의 위(양성 초평면 상)에 존재한다는 의미이다. 따라서 예측값은 1이다.
    - 결과 값이 음수이면 결정 경계의 아래(음성 초평면 상)에 존재한다는 의미이다. 따라서 예측값은 0이다.

비선형 SVM

- 결정 경계와 예측값의 관계
  - 어떤 데이터에 대한 **decision_function** 메소드의 반환 값은 그 데이터가 어떤 초평면 상에 있는지 판단의 근거가 된다.

```
1  X_new = np.array([[1, -0.5], [1, 0], [1, 0.5]])
2  X_new_poly = poly.fit_transform(X_new)
3
4  y_new_pred = clf.predict(X_new_poly)
5  print(y_new_pred)
```

```
[1 0 0]
```

```
1  X_new = np.array([[1, -0.5], [1, 0], [1, 0.5]])
2  X_new_poly = poly.fit_transform(X_new)
3
4  y_new_decision = clf.decision_function(X_new_poly)
5  print(np.round(y_new_decision, 3))
```

```
[ 1.842 -0.752 -2.676]
```

참고 : 비선형 분류 결정 경계의 플롯 표현 예

비선형 SVM

- 비선형 SVM 분류 결정 경계의 플롯
  - 가로축(X)과 세로축(Y)의 각 좌표들마다 높이축(Z) 값이 존재할 때, 이 데이터를 3차원 육면체가 아니라 2차원 평면상에서 등고선으로 그리거나 등고선의 내부를 채워서 높이축의 값을 표현할 수 있다.
    - **pyplot**의 **contour** 함수를 이용하여 등고선을 그린다.
    - **pyplot**의 **contourf** 함수를 이용하여 등고선의 내부를 채운다.
  - 이 함수를 이용하여 결정 경계 곡선을 표시하겠다는 것이 핵심 발상이다.

비선형 SVM

- 비선형 SVM 분류 결정 경계의 플롯
  - 가로축(X)과 세로축(Y)의 각 좌표들마다 높이축(Z) 값이 존재할 때, 이 데이터를 3차원 육면체가 아니라 2차원 평면상에서 등고선으로 그리거나 등고선의 내부를 채워서 높이축의 값을 표현할 수 있다.

```
# plt.contour 함수
# 가로축 X, 세로축 Y, 높이축 Z 값들을 인자로 받아서
# 이 높이에 해당하는 Z 값들을 기준으로 등고선을 표현한다.
plt.contour(X, Y, Z, ...)

# plt.contourf 함수
# 가로축 X, 세로축 Y, 높이축 Z 값들을 인자로 받아서
# 이 높이에 해당하는 Z 값들을 기준으로 등고선의 내부를 채운다.
plt.contourf(X, Y, Z, ...)

# 위 함수들에 대한 자세한 설명과 예제는 pyplot document를 참고
```

비선형 SVM

- 비선형 SVM 분류 결정 경계의 플롯
  - 전체 과정의 예제 구문은 별도의 참고 자료 “참고) 비선형 분류_결정경계의_플롯표현_예” 파일의 내용을 확인한다.