Yonsei University, Department of Applied Statistics

Chapter 9 Support Vector Machines

12기 YBIGTA 신보현

March 2, 2019

이번 챕터에서는 분류 모델 중 하나인 SVM에 대해서 배운다. SVM은 간단하고 직관적인 분류기인 maximal margin classifier의 일반적인 버전이다. maximal margin classifier은 간단하고 우아하지만 클래스들이 선형의 바운더리를 가져야하는 가정으로 인해서 현실에서는 대부분 쓰일 수 없다. 9.2에서는 maximal margin classifier의 확장인 support vector classifier을, 9.3에서는 비선형 바운더리를 수용하기 위해 이를 더 확정한 support vector machine을 소개한다. 9.4에서는 두 개 이상의 클래스에 대해서 SVM을 확장한다. 9.5에는 SVM과 다른 분류 모델과의 연관성에 대해서 논의한다.

9.1 Maximal Margin Classifier

9.1.1 What Is a Hyperplane?

p차원의 공간에서, hyperplane이란, p-1차원의 affine(원점을 지날 필요가 없음을 의미) subspace이다. 예를 들어, 2차원에서 hyperplane은 직선이고 3차원에서는 평면이다. 고차원이 되어도 이러한 개념은 여전히 적용된다.
hyperplane의 수학적인 정의는 굉장히 간단하다. 2차원에서 hyperplane은 아래와 같이 정의된다.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \tag{9.1}$$

(9.1)과 같이 hyperplane을 정의한다는 것은, (9.1)의 식이 유효한 모든 점 $X = (X_1, X_2)^T$이 그 hyperplane위에 있다는 것이다.
(9.1)은 p차원에서 아래와 같이 쉽게 확장된다.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \tag{9.2}$$

다시 한번 강조하면, p 차원의 벡터 $X = (X_1, \cdots, X_p)^T$가 (9.2)을 만족한다면 그 점은 해당 hyperplane 위에 있는 것이다. 이제 벡터 X가 (9.2)을 만족하지 않는 경우를 생각해보자.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \tag{9.3}$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \tag{9.4}$$

즉 hyperplane이 p차원 공간을 두 부분으로 나누고 있다고 생각할 수 있다. hyperplane 위쪽에 있는지, 아래 쪽에 있는지는 $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$을 계산해서 부호의 방향만 결정을하면 쉽게 알 수 있다.

9.1.2 Classification Using a Separating Hyperplane

이제 $n \times p$ 행렬 $\boldsymbol{X}$과 관측치들이 두 개의 클래스에 속하고 테스트 관측치도 가지고 있는 상황을 생각해보자. 늘 그래왔듯이, 훈련 데이터를 통해서 테스트 데이터를 잘 분류하는 분류기를 만들고 싶다. 4단원, 8단원에서 여러 분류기를 살펴보았는데 여기서는 separating hyperplane에 기반한 접근법을 살펴볼 것이다.
관측치의 반응변수 클래스에 따라서 훈련 데이터를 완벽하게 분류하는 hyperplane을 만들 수 있다고 가정해보자. 이러한 분류기의 예는 그림 9.2 왼쪽에 나와있다.

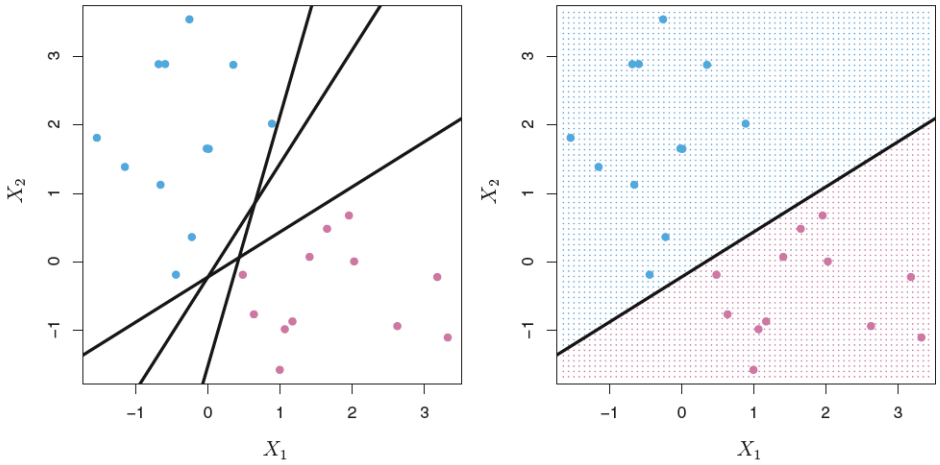

FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

파란색 관측치를 $y_i = 1$, 보라색 관측치를 $y_i = -1$이라고 하자. 그러면 그림 9.2 왼쪽의 separating hyplane은 아래와 같은 특징을 가진다.

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \; if \; y_i = 1$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \; if \; y_i = -1$$

또는 $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$이라고 다시 쓸 수 있다.
만약 separating hyperplane이 존재한다면, 테스트 관측치가 hyperplane의 어떤 면에 속하는지에 따라 분류하는, 매우 자연스러운 분류기를 만들 수 있다. 즉, 테스트 관측치인 x^*을 $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$의 부호에 따라 분류하는 것이다. 또한 $f(x^*)$의

magnitude를 사용할 수 있는데, 만약 $f(x^*)$가 0에서 멀리 떨어져 있다면 x^*가 hyperplane으로부터 멀리 떨어져 있다는 뜻이고 테스트 관측치 x^*의 클래스를 분류하는데 어느 정도 확신이 있을 것이다. 반면에 $f(x^*)$가 0에 가깝다면 x^* 클래스에 대해서 덜 확신할 것이다.

9.1.3 The Maximal Margin Classifier

일반적으로, 데이터가 hyperplane을 통해서 완벽하게 분류될 수 있다면 그러한 hyperplane은 무한하개 존재할 것이다. 그림 9.2의 왼편에는 세 개의 hyperplane이 나와있지만 이를 아주 조금만 돌리면서 hyperplane을 달리하면 관측치를 완벽히 분류하는 무한히 많은 hyperplane을 만들 수 있을 것이다. 이러한 수 많은 hyplane 중 어떤 것을 사용할지 합리적으로 결정해야 한다.

한 방법은 maximal margin hyperplane(또는 optimal separating hyperplane이라고 알려져 있다)을 사용하는 것인데, 이는 훈련 관측치에서 가장 멀리 떨어져 있는 separating hyperplane이다. 즉, hyperplane과 모든 관측치로부터 수직 거리를 계산하고 가장 작은 거리는 관측치와 hyperplane의 최소 거리이며 이를 margin이라고 부른다. maximal margin hyperplane은 이 margin이 가장 큰 separating hyperplane이다. 즉, hyperplane과 훈련 관측치 사이의 거리를 모두 계산하였을 때, 가장 작은 거리가 있을 텐데, 각 hyperplane마다 가지고 있는 이 최소 거리 중 가장 큰 최소거리를 가지는 hyperplane을 선택하는 것이다. 그리고 테스트 관측치를 maximal margin hyperplane의 어느 쪽에 해당하는 지에 따라서 분류를 한다. 바로 이것이 maximal margin classifier의 기본 개념이다. maximal margin classsifier는 종종 성능이 좋지만, 고차원에서는 과적합의 우려가 있다.

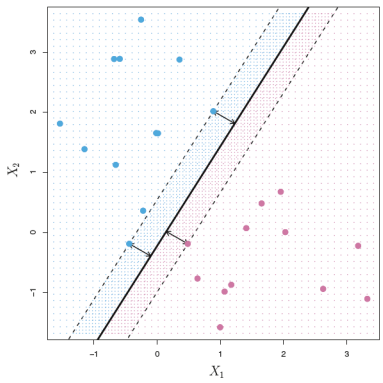

FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

그림 9.3은 그림 9.2의 데이터에 대한 maximal margin classifier이다. 이를 그림 9.2의 오른쪽 hyperplane과 비교해보자. 그림 9.2의 오른쪽 hyperplane과 관측치 사이의 거리의 최소값은

그림 9.3의 그것보다 확실히 작다. 즉, 그림 9.3의 hyperplane의 margin이 더 크다.
그림 9.3에서는 세 개의 관측치가 hyperplane과 동일한 거리에 있다. 이 세개의 관측치들은 supprot vectors라고 알려져 있는데, 그 이유는 이들은 p차원에서 백터이고 이들이 조금만 움직이면 maximal margin hyperplane도 또한 움직일거라는 관점에서 maximal margin hyperplane을 'support' 한다고 보기 때문이다. 흥미롭게도, maximal margin hyperplane은 support vectors에 직접적으로 의존하고 다른 관측치에는 의존하지 않는다. support vectors가 아닌 관측치들이 움직인다고 해서 maximal margin hyperplane은 변화하지 않는다. 이러한 측면은 이후 support vector machines에서 중요하게 쓰이는 개념이기도 하다.

9.1.4 Construction of the Maximal Margin Classifier

이제 어떻게 maximal margin classifier가 만들어지는지 살펴보자. maximal margin hyperplane은 아래의 최적화 문제에 대한 솔루션이다.

$$\underset{\beta_0,\beta_1,\cdots,\beta_p,M}{maximize}\ M \tag{9.5}$$

$$subject\ to \sum_{j=1}^{p} \beta_j^2 = 1 \tag{9.6}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > M\ \forall i = 1, \cdots, n \tag{9.7}$$

(9.5) ~ (9.7)의 최적화 문제는 생각보다 간단하다.
먼저 (9.7)은 M이 양수라는 가정 하에, 각 관측치가 hyperplane의 올바른 면에 있도록 보장한다. 사실 간단히 $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$의 조건으로도 각 관측치가 올바른 면에 있도록 할 수 있지만 M이 양수라는 가정 하에, 0이 아닌 M보다 크다는 조건을 둠으로써 추가적인 제한을 하는 느낌이다.
i번째 점 $(x_{i1}, \cdots, x_{ip})$와 hyperplane $\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p$의 거리는 $\frac{y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})}{\sqrt{\beta_1^2 + \cdots + \beta_p^2}}$
인데, (9.6)으로 인해서 $y_i(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_p x_{ip})$으로 바꿀 수 있고, (9.6), (9.7)을 모두 고려하면, 관측치와 hyperplane과의 거리가 최소 M보다는 커야 함을 유추할 수 있다. 즉, M은 hyperplane과 관측치 사이의 최소 거리이므로, hyperplane의 margin을 의미한다. 따라서 (9.5)로부터 이 최적화 문제는 결국 margin을 최대화하는 문제임을 알 수 있다. 이는 maximal margin hyperplane의 정의와 완전히 동일하다. 구체적으로 해를 찾아가는 과정은 이 책의 수준을 넘어서므로 생략하도록 한다.

9.1.5 The Non-separable Case

maximal margin classifier는 많은 경우에, separating hyperplane이 존재하지 않기 때문에

수행하기가 힘들다. 즉, (9.5) ~ (9.7)의 최적화 문제에 대한 해가 없는 것이다. 그림 9.4의 예시를 보자.

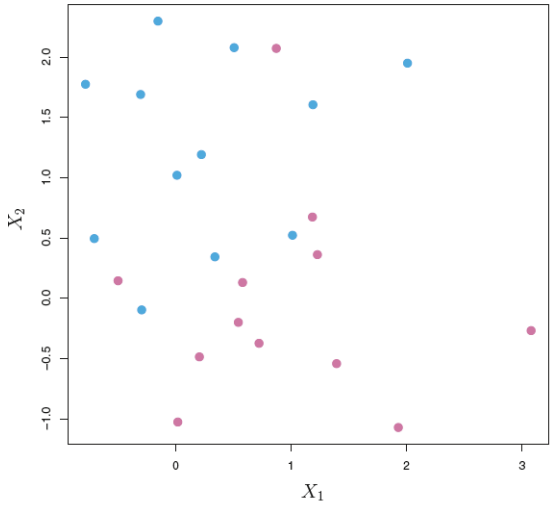

FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

두 클래스를 완전히 나눌 수 없다. 하지만 다음 단원에서는 완벽히 클래스를 분류하는 separating hyperplane의 개념을 soft margin이라고 불리는 기법을 이용하여 클래스를 거의 (almost) 구분하는 확장된 개념에 대해서 살펴볼 것이다. 이렇게 완벽히 구분을 못하는 문제에 대한 대안으로 나온 것이 바로 support vector classifier이다.

9.2 Support Vector Classifiers

9.2.1 Overview of the Support Vector Classifier

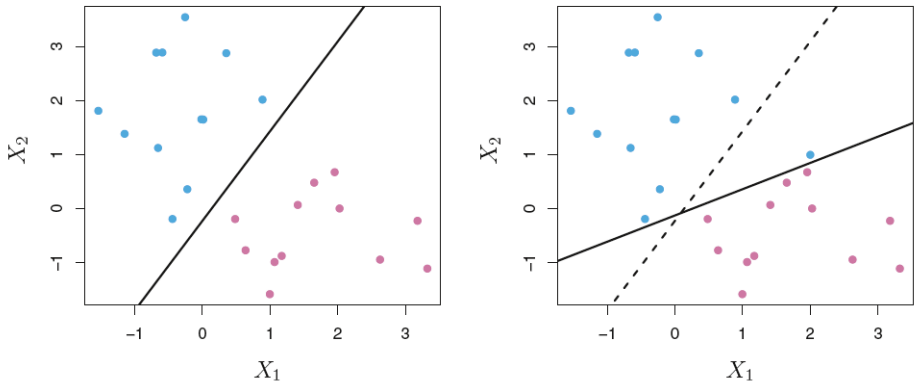

FIGURE 9.5. Left: *Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane.* Right: *An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.*

separating hyperplane에 기반한 분류기는 모든 훈련 데이터를 완벽하게 분류하지만 각 관측치에 대한 민감도로 이어질 수 있다. 그림 9.5의 오른쪽을 보자. 하나의 데이터가 추가되었을뿐인데 maximal margin hyperplane이 심하게 변화하였다. 변한 hyperplane은 작은 margin을 가지게 되었다. margin이 작을수록 분류기의 성능에 대해서 의심을 품을 수 있기 때문에 문제가 되는 상황이다. 또한 이렇게 하나의 데이터에 대해 민감하게 반응한다는 것은 훈련 데이터에 과적합될 우려가 있기도 하다. 이러한 경우에, 두 클래스를 완벽히 나누지 않은 hyperplane에 기반한 분류기를 고려한다. 이를 통해서 개인의 관측치에 대해서 더 나은 robustness을 지니고 훈련 데이터의 거의 대부분(most)에 대해 좋은 분류를 목표로 한다. 몇몇의 관측치에 대해서는 분류를 잘 못할수도 있지만 나머지 관측치에 대해서는 올바른 분류를 하는 것이 목적이다.

support vector classifier(soft margin classifier라고도 불린다)는 정확히 이것을 한다. 모든 관측치가 올바른 쪽에 있게 하기 위해서 가장 큰 margin을 찾는 것이 아니라 몇몇 관측치가 틀린 쪽에 있는 것을 허용하는 것이다(soft margin은 이러한 위반을 허용한다는 점에서 soft라는 이름이 붙여졌다) 그림 9.6의 예시를 보자. 모든 관측치가 올바른 쪽에 분류된것이 아니라 몇몇의 관측치는 틀린 margin 쪽에 있고 심지어 틀린 class로 구분된 경우도 있다. 틀린 hyperplane 면에 있다면 이는 support vector classifier가 잘못 분류한 훈련 데이터이다.

이러한 상황은 두 클래스를 완벽하게 구분하는 separating hyperplane이 없다면 피할 수 없는 상황이다.

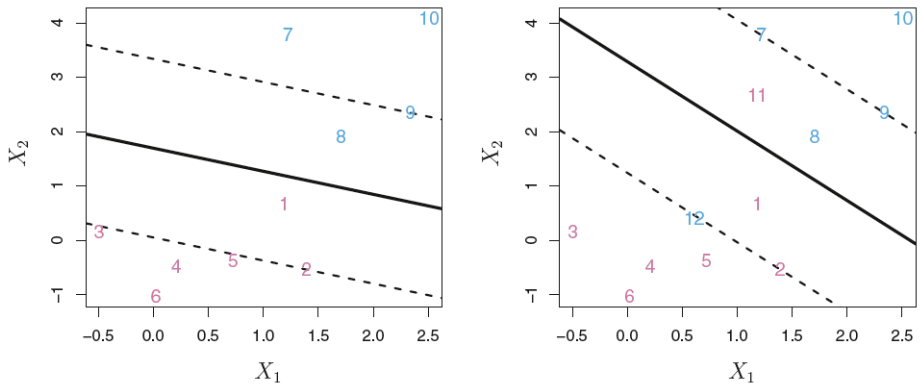

FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* $3, 4, 5$, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

9.2.2 Details of the Supprot Vector Classifier

support vector classifier은 테스트 관측치를 hyperplane의 어느 쪽에 있는지에 따라서 분류를 한다. 이러한 hyperplane은 아래 최적화 문제에 대한 해답으로 만들어진다.

$$\underset{\beta_0,\beta_1,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_p,M}{maximize} M \tag{9.8}$$

$$subject\ to \sum_{j=1}^{p} \beta_j^2 = 1 \tag{9.9}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > M(1 - \epsilon_i)\ \forall i = 1, \cdots, n \tag{9.10}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{9.11}$$

여기서 C는 음수가 아닌 tuning parameter이다. (9.10)에서 $\epsilon_1, \cdots, \epsilon_n$은 slack variable로, 각 관측치가 틀린 margin이나 hyperplane 쪽에 있도록 허락하는 역할을 한다. (9.8) ~ (9.11)의 최적화 문제를 풀었으면 테스트 관측치인 x^*가 놓여있는 hyperplane 면에 분류를 한다. 즉, $f(x^*)$의 부호에 따라서 분류를 한다.

(9.8) ~ (9.11)을 자세히 살펴보자. slack variable인 ϵ_i은 i번째 관측치가 hyperplane과 margin과 비교하여 어디에 있는지 알려준다. 만약 $\epsilon_i = 0$이라면 i번째 관측치는 margin에 대해서 올바른 쪽에 있는 것이고 $\epsilon_i > 0$이라면 해당 관측치가 margin을 위반(violated)했다고 하며 margin의 잘못된 쪽에 있는 것이다. 마지막으로 $\epsilon_i > 1$이라면 hyperplane의 잘못된 쪽에 있는 것이다.

이제 tuning parameter인 C의 역할에 대해서 논의한다. (9.11)에서 C는 slack variable의 합에 대한 한계점이다. 즉, C는 margin과 hyperplane에 대한 위반의 정도와 수를 결정한다. C를 관측치가 margin을 위반할 budget이라고 생각할 수 있다. 만약 $C = 0$이라면 모든 slack variable은 0이어야 하고 margin 위반을 해서는 안된다. 이러한 경우에는, maximal margin hyperplane과 동일해질 것이다. C가 증가할수록 margin의 위반에 대해서 더 관용적이고 margin이 더 넓어질 것이다. 반대로 감소할수록, margin은 좁아질 것이다.

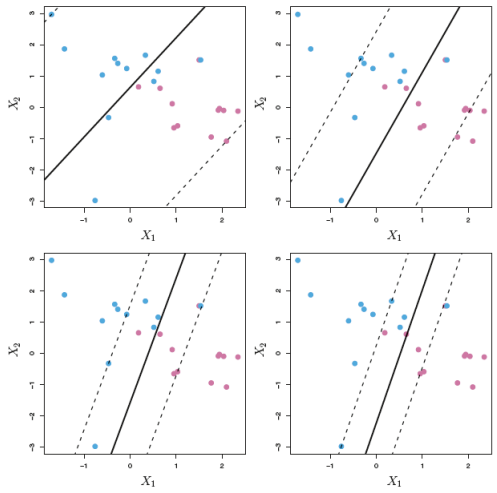

FIGURE 9.7. *A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.*

그림 9.7에는 C의 변화에 따른 margin의 변화가 나타나있다.

C는 CV를 통해서 선택된다. 여태껏 봐왔던 tuning parameter처럼, C는 bias-variance trade-off을 조정한다. 만약 C가 작다면, 위반을 잘 허용하지 않은 좁은 margin이 형성이 된다. 이는 훈련데이터에 과적합될 우려가 있어 분산은 크지만 낮은 편향을 가진다. 반면에 C가 크다면 위반을 상대적으로 더 허용하는 넓은 margin이 형성이 되고 이는 잠재적으로 높은 편향을 가지지만 분산은 작을 것이다.

(9.8) ~ (9.11)의 최적화 문제는 매우 흥미로운 측면이 있다. margin위에 있거나 margin을 위반하는 관측치들만이 hyperplane에, 그에 따라서 분류기에 영향을 준다는 것이다. 다시 말하면 올바른 쪽의 margin에 있는 관측치는 support vector classifier에 영향을주지 않을

것이다. 직접적으로 margin위에 있거나 틀린 margin 쪽에 있는 관측치들(이들을 support vectors라고 부른다)이 support vector classifier에 영향을 준다.

오직 support vectors만이 분류기에 영향을 준다는 사실은 C가 분류기의 bias-variance trade-off를 조절한다는 이전의 주장과 일맥상통한다. 만약 C가 크다면 많은 관측치들이 margin을 위반할 것이고 그에 따라서 suppor vectors도 많이 있을 것이다. 이러한 경우에 많은 관측치들이 hyperplane을 결정하는데 관련이 있다. 그림 9.7의 좌상단을 보자. 이 분류기는 margin이 넓어 많은 support vectors을 가지고 그에 따라서 분산은 작지만 편향은 잠재적으로 높다. 하지만 C가 작다면 support vectors가 적을 것이고 그로 인해서 생기는 분류기는 작은 편향과 높은 분산을 가질 것이다. 우하단 그림은 이러한 상황을 보여준다.

support vector classifier의 결정 규칙이 훈련 데이터의 작은 일부(support vectors)만에 기반한다는 것은 hyperplane으로부터 멀리 떨어져 있는 관측치에 대해서는 꽤 robust하다는 것을 의미한다. 이는 LDA와는 확연히 다른 특징이다. LDA는 모든 관측치의 평균과 그 클래스 내의 모든 관측치를 이용하여 계산한 공분산 행렬에 의존한다. 이와는 다르게 로지스틱 회귀는 결정 바운더리로부터 멀리 떨어진 관측치에는 덜 민감하다. 사실 support vector classifier와 로지스틱 회귀는 밀접하게 연결되어 있는데 이는 9.5에서 살펴본다.

9.3 Support Vector Machines

먼저 선형 분류기를 비선형 결정 바운더리로 변환하는 일반적인 메커니즘에 대해서 얘기하고 이를 자동으로 하는 SVM에 대해서 논의한다.

9.3.1 Classification with Non-linear Decision Boundaries

support vector classifier은 선형 바운더리로 분류를 하지만 실제로 종종 비선형 클래스 바운더리에 직면할 수 있다. 예를 들어서 그림 9.8 왼쪽의 데이터를 살펴보자.

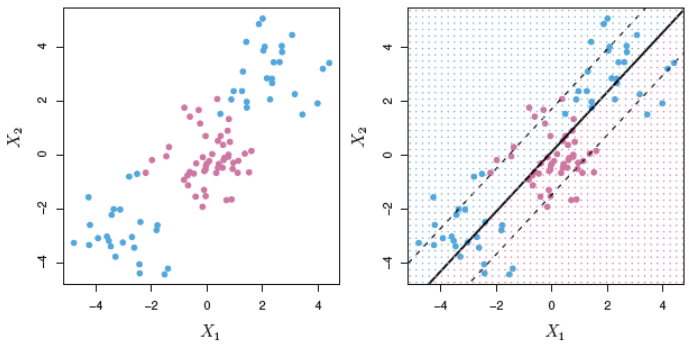

FIGURE 9.8. Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

이러한 데이터에 선형 결정 바운더리를 적용한다면 별로 좋지 못한 성능을 낼 것이다.
7단원에서, 비슷한 상황에 직면했었다. 반응변수와 예측변수간에 선형이라는 강력한 가정이 존재하지 않을 때에는, 비선형을 해결하기 위해 예측 변수의 함수를 사용하여 feature space을 확장했다. 비선형 바운더리의 문제도 이와 비슷하게, 예측 변수의 2차, 3차 또는 그 이상의 다항 함수를 사용하여 feature space을 확장하는 방식으로 해결한다. 예를 들어서 p개의 예측변수를 사용하는 support vector classifier을 $X_1, X_1^2, \cdots, X_p, X_p^2$의 2p개 예측변수를 사용하여 support vector classifier을 적합할 수도 있다.
이렇게 여러 방법으로 feature space을 확장할 수 있고 주의를 하지 않으면 매우 많은 변수로 적합할 수 있다. 이는 과정합의 위험과 계산량이 어마어마할 것이다. SVM은 feature space을 효율적인 계산을 할 수 있도록 확장한다.

9.3.2 The Support Vector Machine

SVM은 support vector machine의 확장인데, feature space를 kernels을 사용하여 확장하는 특징을 가진다. 핵심은 클래스들 간의 비선형 바운더리를 수용하기 위해 feature space을 확장하는 것이다. kernal 접근법은 이러한 생각을 실행하기 위한 효율적인 계산 접근법이다. support vector classifier의 최적화 문제에 대해서 자세하게 논의 하지는 않았지만, 구체적인 증명을 생략하면 이에 대한 해답은 관측치의 내적이다. 두 관측치 $x_i, x_{i'}$의 내적은 다음과 같다.

$$< x_i, x_{i'} >= \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{9.12}$$

이를 이용하면아래와 같은 결론을 얻을 수 있다(구체적인 증명은 생략)

- linear support vector classifier은

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i > \tag{9.13}$$

  이고 α_i는 훈련 데이터 하나당 부여된 n개의 모수이다.

- $\alpha_1, \cdots, \alpha_n, \beta_0$을 추정하기 위해서 (모든 훈련 데이터에 대한) $\begin{pmatrix} n \\ 2 \end{pmatrix}$개의 내적 $< x_i, x_{i'} >$이 필요하다.

(9.13)을 구하기 위해 새로운 점 x와 기존 훈련 데이터들인 x_i간의 내적을 구해야 함을 주목하자. 하지만, α_i은 support vectors에 대해서만 0이 아니다. 다시 말하면, 훈련 데이터가 support vector가 아니면 α_i는 0이다(구체적인 증명 생략) 따라서 $\mathcal{S}$를 이러한 support

points의 인덱스의 모임이라고 하면 (9.13)을 아래와 같이 다시 쓸 수 있다.

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i < x, x_i > \tag{9.14}$$

요약하면 linear classifier인 $f(x)$을 나타내기 위해, 그리고 계수를 계산하기 위해서는 내적만 필요하다.
이제 (9.12)의 내적이 (9.13)에 나타날때마다, 또는 support vector classifier을 계산할때마다 이를 내적의 일반화 형태로 바꾼다고 생각해보자.

$$K(x_i, x_{i'}) \tag{9.15}$$

여기서 K는 나중에 kernel이라고 부를 어떤 함수이다. kernel은 두 관측치 간에 유사도를 나타내는 함수이다. 예를 들어서 간단하게 kernel을

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{9.16}$$

로 한다면 이는 support vector classifier이다. (9.21)은 support vector classifier의 kernel이므로 linear kernel로 알려져 있다. linear kernel은 Pearson 상관계수를 이용해서 관측치들의 유사도를 측정한다.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d \tag{9.17}$$

(9.22)은 d차의 polynomial kernel이다. (9.16)과 같이 linear kernel을 사용하는 대신 (9.17)의 kernel을 사용하면 결정 바운더리가 훨씬 더 flexible해진다. 이는 원래의 feature space가 아닌 d 차원의 고 차원 공간에 support vector classifier을 적합하는 것과 동일하다. (9.17)과 같이 support vector classifier가 non-linear kernel과 결합되어 생기는 분류기를 SVM이라고 부른다. 이 때, non-linear 함수는 아래와 같은 형태를 가진다.

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i) \tag{9.18}$$

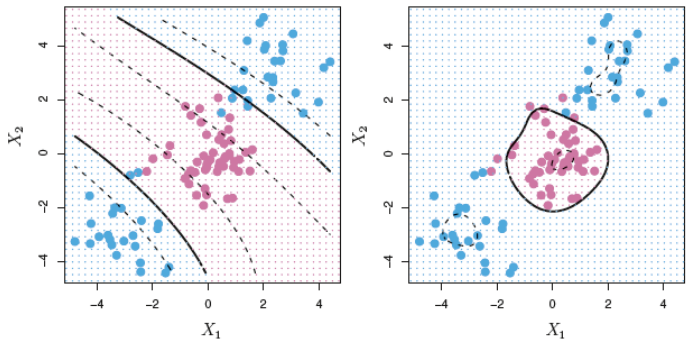

FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

그림 9.9는 비선형 데이터에 다항 kernel을 적용시킨 SVM 예이다. 오른쪽 그림은 radial kernel을 사용한 SVM 결과이다. radial kernel은 아래와 같은 형태이다.

$$K(x_i, x_{i'}) = exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) \tag{9.19}$$

(9.19) 에서 γ 은 양의 상수이다. radial kernel이 어떻게 작동할까? 만약 주어진 테스트 관측치, $x^* = (x_1^*, \cdots, x_p^*)^T$ 가 유클리디안 거리의 관점에서 훈련 데이터와 멀리 있다면 $\sum_{j=1}^{p}(x_j^* - x_{i'j})^2$ 는 커질 것이고 (9.19) 는 작아질 것이다. 이는 (9.18) 에서 훈련 데이터들인 x_i 가 $f(x^*)$ 에 영향을 거의 안 미친다는 뜻이다. 테스트 관측치의 라벨은 $f(x^*)$ 의 부호에 의해서 결정되었음을 상기해보자. 다시 말해, 테스트 데이터인 x^* 와 멀리 떨어져 있는 훈련 데이터는 $f(x^*)$ 에 거의 영향을 주지 않고 그에 따라서 x^* 의 라벨을 결정하는 데에도 거의 역할이 없다. 이는 radial kernel이 매우 지역적 (local) 인 행동을 보이는 것을 의미하는데, 테스트 데이터의 라벨을 결정할 때 이와 가까운 훈련 데이터에 영향을 많이 받기 때문이다.

9.3.3 An Application to the Heart Disease Data

Heart 데이터에 대해서 개인이 심장병을 가지고 있는지 아닌지에 대해 SVM과 LDA가 얼마나 잘 분류하는지 비교해본다.

우선 훈련 데이터에 LDA와 support vector classifier을 적합한다. support vector classifier는 1차인 polynomial kernel을 사용한 SVM과 동일한 모델이다.

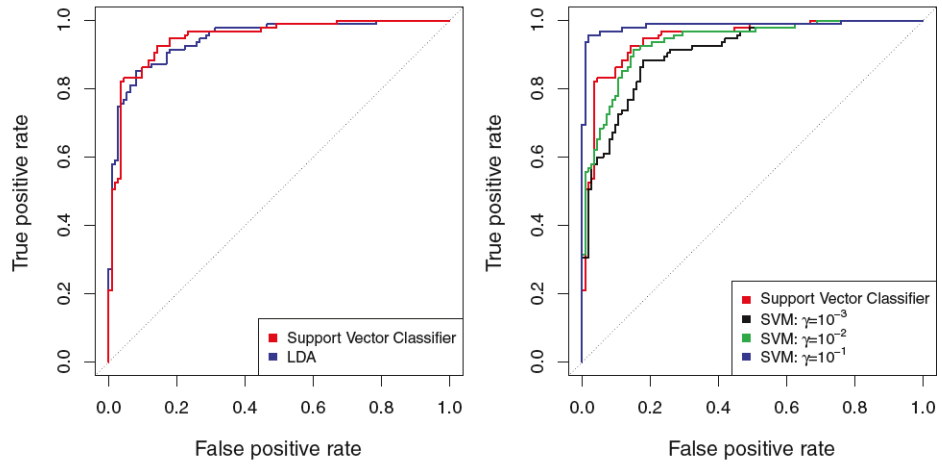

FIGURE 9.10. *ROC curves for the* `Heart` *data training set.* Left: *The support vector classifier and LDA are compared.* Right: *The support vector classifier is compared to an SVM using a radial basis kernel with* $\gamma = 10^{-3}$, 10^{-2}, *and* 10^{-1}.

그림 9.10의 왼쪽은 LDA와 support vector classifier을 훈련 데이터에 적합시킨 ROC 커브이다. 두 분류기는 어떤 점수 $\hat{f}(X)$를 계산할 것이고 cut point인 t를 정해서 $\hat{f}(X) < t$ *or* $\hat{f}(X) \geq t$에 따라서 심장병 여부를 분류한다. 그리고 False positive rate, True positive rate를 계산하여 ROC 커브를 그린다. 최적의 분류기는 ROC 커브가 좌상단을 품을 것이다. 여기서는 support vector classifier가 성능이 조금 더 좋은 것으로 나타났따.

그림 9.10의 오른쪽은 다양한 γ로 radial kernel을 사용한 SVM 적합 결과를 보여준다. γ가 커질수록 비선형에 가까운 적합이 되어 ROC 커브가 향상된다. 하지만 이는 훈련 오차 비율이므로 이를 너무 맹신하지는 말자.

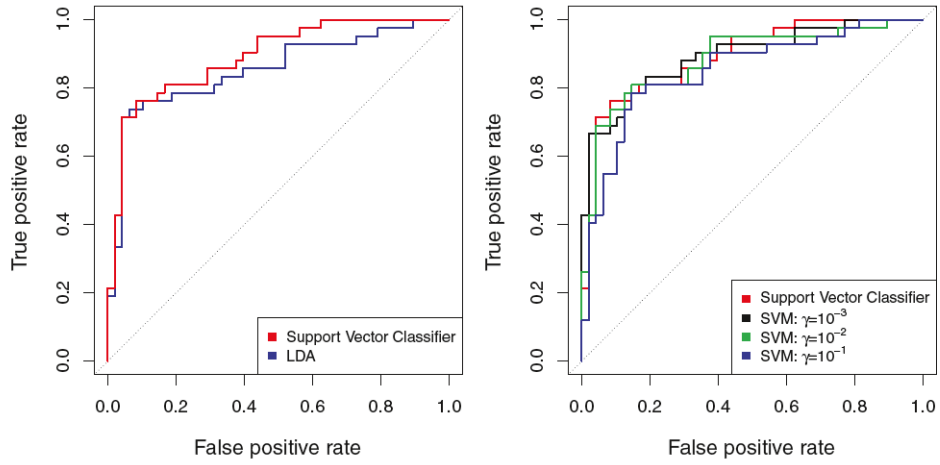

FIGURE 9.11. *ROC curves for the test set of the* `Heart` *data.* Left: *The support vector classifier and LDA are compared.* Right: *The support vector classifier is compared to an SVM using a radial basis kernel with* $\gamma = 10^{-3}$, 10^{-2}, *and* 10^{-1}.

그림 9.11은 테스트 데이터에 대한 적합 결과이다. 오른쪽을 보면, γ가 가장 클 때, 가장 좋지 못한 성능을 보이는데, 이는 flexible한 모델이 훈련 데이터에 대해서는 좋은 성능을 보이지만 테스트 데이터에 대해서는 분산이 커서 테스트 오차 비율은 높아지는 것과 일맥상통한다.

9.4 SVMs with More than Two Classes

여태까지 이진 분류 문제에 국한해서 논의를 진행했다. SVMs이 일반적인 분류 문제에서 어떻게 확장될까? K개의 클래스에 SVMs을 적용하는 두 가지 접근법으로 여기서는 가장 인기가 있는 one-versus-one과 one-versus-all을 소개한다.

9.4.1 One-Versus-One Classification

$K > 2$개의 클래스를 가지는 분류 문제에 SVMs을 적용하려 한다고 하자. one-versus-one 또는 all-pairs 접근법은 $\binom{K}{2}$ 개의 SVMs을 적합하는데 각각은 클래스들의 쌍이다. 예를 들어, k번째 클래스와 k'번째 클래스를 비교하는 SVM을 생각해보자. 한 테스트 관측치를 각 $\binom{K}{2}$ 개의 분류기를 이용하여 분류하고 그 테스트 관측치가 K개의 클래스에 배정된 횟수를 합한다. 마지막 분류는 테스트 관측치를 가장 많이 배정된 클래스로 배정한다.

9.4.2 One-Versus-All Classification

K개의 SVMs을 적합하는데 매번 K개의 클래스 중 하나와 나머지 모든 K-1개의 클래스와 비교한다. $\beta_{0k}, \beta_{1k}, \cdots, \beta_{pk}$를 이와 같은 SVM을 하여 발생하는 모수라고, x^*을 테스트

관측치라고 하자. 테스트 관측치를 $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$가 가장 큰 관측치에 배정하는데 그 이유는 이것이 나머지 다른 K-1개의 클래스보다는 K번째 클래스에 테스트 관측치가 속하다고 높게 확신하는 것과 동일하기 때문이다.

9.5 Relationship to Logistic Regression

support vector classifier에 대한 최적화 문제인 (9.8) ~ (9.11)은 아래와 같이 다시 쓸 수 있다(구체적인 설명 생략)

$$\underset{\beta_0,\cdots,\beta_p}{minimize} \left\{ \sum_{i=1}^{n} max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{9.20}$$

만약 λ가 크다면 $\beta_1, \cdots, \beta_p$는 작고 margin에 대한 더 많은 위반이 허용될 것이며 작은 분산과 높은 편향을 가진 분류기가 생길 것이다. 만약 λ가 작다면 margin에 대한 더 적은 위반이 허용되며 높은 분산과 작은 편향을 가진 분류기가 생길 것이다. 따라서 작은 λ값은 (9.11)에서 작은 C값과 동일하다. (9.20)의 penalty항은 ridge의 penalty항과 동일하며 bias-variance trade-off를 조절한다.
이제 (9.20)은 여태껏 많이 봐왔던 Loss + Penalty 형태를 취한다.

$$\underset{\beta_0,\cdots,\beta_p}{minimize}\{L(\boldsymbol{X}, \boldsymbol{y}, \beta) + \lambda P(\beta)\} \tag{9.21}$$

(9.21)에서 $L(\boldsymbol{X}, \boldsymbol{y}, \beta)$dms β에 의해서 parametrized된 모델이 데이터 $(\boldsymbol{X}, \boldsymbol{y})$에 얼마나 잘 적합되는지를 수치화한 어떤 loss function이고 $P(\beta)$은 tuning parameter λ에 의해서 효과가 조절되는 penalty function이다. 예를 들어 ridge와 lasso의 Loss function은 모두 아래와 같다.

$$L(\boldsymbol{X}, \boldsymbol{y}, \beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

그리고 패널티항에 대해서, ridge는 $P(\beta) = \sum_{j=1}^{n} \beta_j^2$, lasso는 $P(\beta) = \sum_{j=1}^{n} |\beta_j|$이다. support vector classifier의 Loss function은 아래와 같다.

$$L(\boldsymbol{X}, \boldsymbol{y}, \beta) = \sum_{i=1}^{n} max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})]$$

이는 hinge loss라고 알려져 있는데, 그림 9.12에 나타나있다.

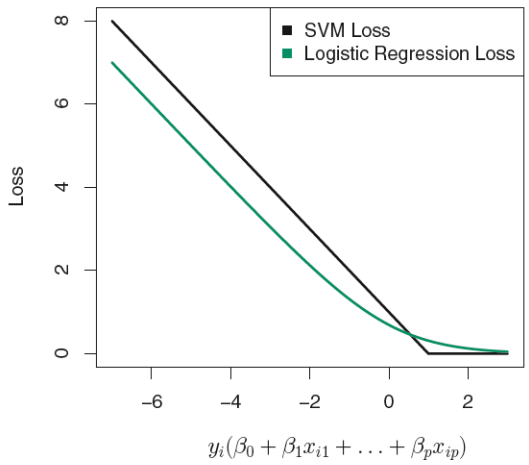

FIGURE 9.12. *The SVM and logistic regression loss functions are compared, as a function of* $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$. *When* $y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$ *is greater than 1, then the SVM loss is zero, since this corresponds to an observation that is on the correct side of the margin. Overall, the two loss functions have quite similar behavior.*

그런데 LR의 Loss function이 hinge loss과 깊게 연관되어 있음이 밝혀졌다. support vector classifier의 흥미로운 특징 중 하나는 오직 support vectors만이 분류기에 영향을 미친다는 것이다. 올바른 margin 쪽에 있는 관측치는 영향이 없다. 그 이유는 그림 9.12에 보여진 loss function이 $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1$인 관측치에 대해서 정확히 0이기 때문이다. 이는 곧, 올바른 margin쪽에 있는 관측치를 의미한다. 그와 반면에 그림 9.12에서 LR의 loss function은 정확히 0이 되는 지점은 없다. 하지만 결정 바운더리에서 매우 멀리 떨어진 관측치에 대해서는 매우 작은 값을 가진다. 이러한 loss functions의 유사성 때문에 LR과 support vector classifier는 매우 유사한 결과를 낸다.