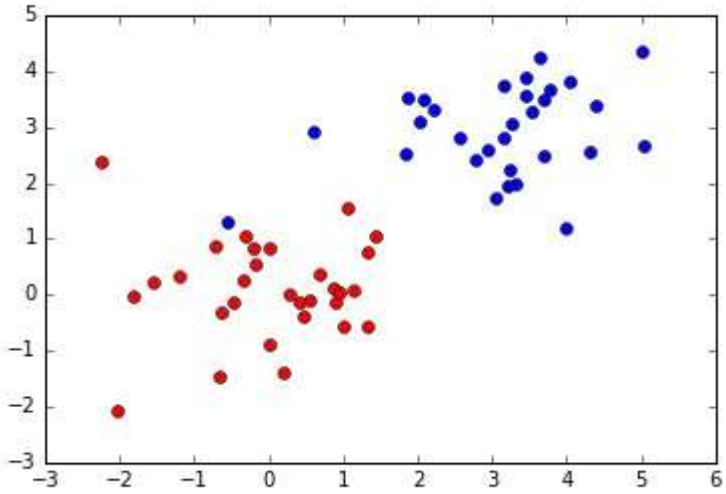
3. 선형 모형을 이용한 분류

지도학습에서 회귀(regression)는 연속적인 값을 예측하는 것인 반면, 분류(classification)는 이산적인 값을 예측하는 것이다. 예를 들어 생선 사진을 보고 넙치(광어)인지 도다리인지 구별하는 문제의 경우 회귀가 아닌 분류에 속한다. 넙치면 넙치고, 도다리면 도다리지 그 사이에 연속적인 것이 없기 때문이다.

위의 예에서 넙치, 도다리와 같은 것을 각각 클래스(class)라고 한다. 당연하지만 여기서 클래스는 '반'이 아니라 '종류'를 뜻한다. 분류는 클래스의 수에 따라 다시 클래스가 2개인 이항 분류(binary classification)와 클래스가 3개 이상인 다항 분류(multi class classification)으로 나눈다.

3.1. 이항 분류

간단한 이항 분류 문제부터 생각을 해보자.



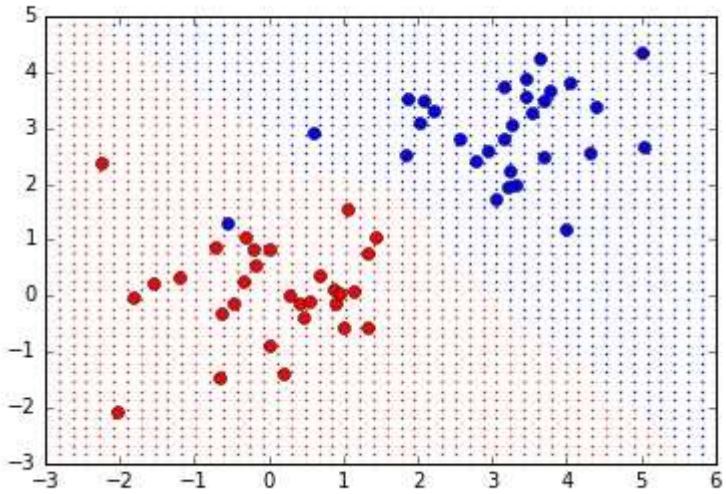
아래 그림에서 빨간 점과 파란 점은 두 개의 서로 다른 클래스이다. 어떤 클래스인지는 알아서 상상하면 된다. 넙치와 도다리일 수도 있고, 송이와 독버섯일 수도 있다. 가로축과 세로축은 두 클래스의 어떤 특성을 나타낸다. 어떤 특성인지도 역시 상상에 맡겨둔다. 클래스나 특성이 무엇인지는 구체적으로 중요하지 않다.

3.2. 최근접 이웃

어쨌든 이 문제를 접근 하는 방법 중에 여러 가지 방법을 생각해볼 수 있다. 첫째는 최근접 이웃(k Nearest Neighbor)의 방법이다. 이 방법은 학습 데이터를 모두 기억하고, 새로운 데이터가 들어왔을 때 기억하고 있는 학습 데이터 중에 가장 비슷한 사례를 찾는 방법이다. 만약 그 사례가 빨강 클래스이면 빨강으로 예측하고, 파랑 클래스이면 파랑으로 예측한다.

가끔씩은 예외도 있을 수 있으니 하나의 사례가 아니라 k개의 사례를 찾아서 일종의 다수결로 결정

할 수도 있다. 예를 들어 $k = 5$이면 새로운 데이터와 가장 비슷한 5개의 사례를 찾는다. 이 5개의 사례 중에 빨강이 더 많으면 빨강으로, 파랑이 더 많으면 파랑으로 예측한다. 아래 그림에서 큰 점은 기존 사례, 작은 점은 새로운 사례를 나타낸다.



최근접 이웃의 방법은 회귀든 분류든 가리지 않고 쓸 수 있다. 회귀 문제에서는 다수결 대신에 평균을 내면된다.

최근접 이웃은 간단하고 직관적이다. 특별한 수학적, 통계적 가정도 필요하지 않다. 심지어 '학습'의 과정조차 거의 없다. 단순히 비슷한 사례를 찾기만 하면 되기 때문이다. 게다가 데이터가 충분히 많다면 정확성도 높다.

과적합

하지만 최근접 이웃은 지나치게 기존 사례에 의존적이라는 문제가 있다. 데이터에는 항상 예외, 오차, 우연 등이 포함되어 있다. 위의 그림에서도 파란 점 하나가 빨간점들 사이에 섞여 있는 것을 볼 수 있다. 실제의 패턴을 반영하는 것일 수도 있고, 예외적인 사례일 수도 있다. 만약 예외적인 사례라면 이를 반영하는 것은 흔히 말해 '성급한 일반화의 오류'를 저지르게 된다. 통계적 용어로는 **과적합**(overfitting)이라고 한다.

과적합의 문제는 데이터가 많으면 많을 수록 완화가 된다. 앞서 말한 빨간 점 사이의 파란점 주변에 더 많은 데이터를 모아보면 그것이 패턴인지 우연인지 가려볼 수 있기 때문이다. 그러나 최근접 이웃은 데이터가 많아지면 또 다른 문제가 생긴다.

대부분의 기계학습 모형은 데이터가 많아지면 학습 시간은 길어지지만 한 번 학습이 끝나고 나면 예측에 걸리는 시간에는 차이가 없다. 최근접 이웃은 새로운 사례가 들어올 때마다 기존의 사례 중 가장 비슷한 사례를 검색해야 하기 때문에, 데이터가 많으면 많을 수록 검색 시간이 길어진다.

거리 계산

최근접 이웃은 비슷한 사례를 찾는 방법이다. 바꿔말하면 거리(distance)가 가까운 이웃을 찾는 문제라고도 할수 있다. 이 거리 계산이 생각보다 쉽지 않다. 연속변수의 경우에는 두 사례의 차이를 구

하면 거리를 쉽게 구할 수 있다.

그렇지만 이산변수의 경우 거리를 수치화하기 어렵다. 예를 들면 충청도와 경기도와 강원도 중에 어느 쪽과 더 먼가? 여기서는 물리적 거리가 아니라 두 변수의 성질이 얼마나 다른가를 묻는 것이다. 이런 것은 딱 떨어지는 수치로 만들기 어렵다. 보통은 같으면 거리를 0, 다르면 1로 수치화하는 방법을 쓴다.

또 다른 문제는 변수가 여러 가지일 때 이들의 차이를 어떻게 결합하느냐이다. 키 차이 1cm와 몸무게 차이 100g 중 어느 것이 더 먼 거리인가?

주변부에서 왜곡

분류의 경우에는 크게 나타나지 않지만 회귀의 경우에는 주변부에서 왜곡 현상이 나타날 수 있다. 기존 사례의 최대치를 넘어서는 새로운 사례가 들어오면 기존의 사례만으로 예측을 하기 때문에 과소하게 예측을 하게 된다. 반대로 최소치를 밑도는 사례가 들어오면 반대로 과다하게 예측을 하게 된다.

차원의 저주

점을 하나찍고 여기서 좌우로 길이가 10인 선을 그려보자. 이 선을 '전체'라고 부르자. 다시 중심 점에서 좌우로 길이가 9인 선을 그려보자. 이 선은 '중심부'라고 하자. 그러면 전체에서 중심부가 차지하는 비율은 얼마인가? 90%이다.

이번에는 차원을 하나 높여서 반지름이 10인 원을 전체로 하고 그 중에 반지름 9인 중심부를 생각해보자. 원의 넓이는 반지름의 제곱에 비례하므로 중심부의 비율이 81%로 줄어든다. 차원을 하나 더 높여서 반지름이 10인 구를 전체로 하면 구의 부피는 반지름의 세제곱에 비례하므로 중심부의 비율이 73%로 줄어든다. 이런 식으로 차원을 계속 높여 나가면 43차원에서는 중심부가 차지하는 비율이 1%에 불과하게 된다. 거리로는 계속 전체의 90%에 달하는대도 말이다. 이를 **차원의 저주** (curse of dimensionality)라 한다.

데이터에서 변수들은 각각 하나의 차원을 이룬다. 2개의 변수가 있다면 2차원, 3개의 변수가 있다면 3차원이다. 변수가 많아지면 많아질 수록 차원도 높아진다. 그러면 데이터는 차원의 저주 때문에 중심부에는 드물고 주변부에 많아지게 된다.

그런데 앞서 말했듯이 최근접 이웃은 주변부에서 왜곡 현상이 나타난다. 게다가 중심부는 기존 사례가 드물기 때문에 정확성이 떨어진다. 따라서 변수가 다양한 데이터에 최근접 이웃을 적용하면 중심부건 주변부건 잘 맞지 않게 된다.

3.3. 선형 분류

최근접 이웃의 장점과 단점은 모두 기존의 사례들을 하나 하나 직접 활용하는데서 나온다. 그러면 이번에는 정반대의 접근을 취해서 각각의 사례 대신 전체적인 패턴을 학습하는 방법을 생각해보자. 위의 $k = 5$인 최근접 이웃의 그림을 보면 빨간색과 파란색 사이를 가로지르는 하나의 경계선을

볼 수 있다. 만약 각각의 사례 대신에 이 경계선 하나를 학습할 수 있다면 최근접 이웃의 여러 단점들을 극복할 수 있다.

먼저 가장 단순하게 이 경계선이 직선의 형태라고 생각해보자. 이를 선형 분류(linear classification)이라고 한다. 선형 분류는 단순하기 때문에 학습도 쉽지만 과적합에도 강하다.

위의 예에서 가로축을 x_1이라 하고 세로축을 x_2라고 하자. 다음 같은 선형 모형을 만든다.

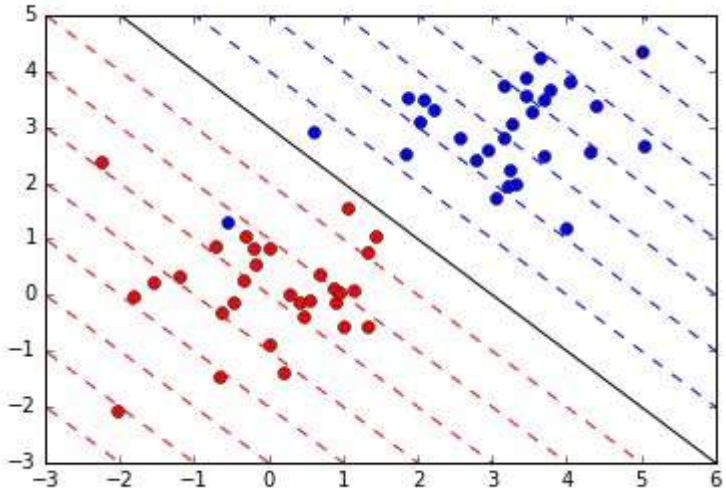
$$y = w_1 x_1 + w_2 x_2 + b$$

위의 식을 자세히 보면 선형 회귀와 식 자체는 동일하다는 것을 알 수 있다. 한 가지 다른 점은 y가 데이터에서 나오는 값이 아니라는 것이다.

만약 $w_1 = w_2 = 1$이고 $b = 0$이면 식은 단순히

$$y = x_1 + x_2$$

가 된다. 이제 y 값이 같은 점들을 직선으로 이으면 아래와 같은 그림이 된다.



이 직선들 중에 가운데 검은 실선은 $y = 6$인 경우이다. 여러 직선들 중에 가장 경계선으로 적절해 보인다. 여기서 $b = -6$이라고 하면 $y = x_1 + x_2 - 6$으로 바꿀 수 있다. 이 때는 위의 실선은 $y = 0$이 된다.

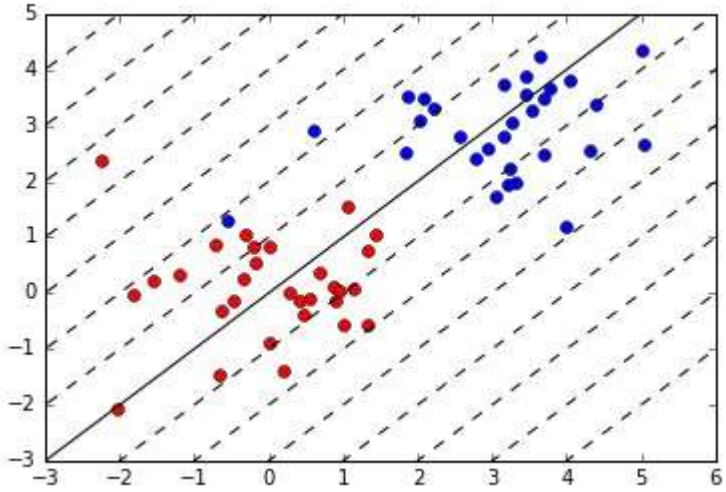
선형 분류는 $y = 0$이 가장 적절한 경계선이 되도록 w와 b를 결정하는 것이다. 그 '가장 적절함'은 손실 함수에 의해 정해진다.

3.3.1. 판별 분석

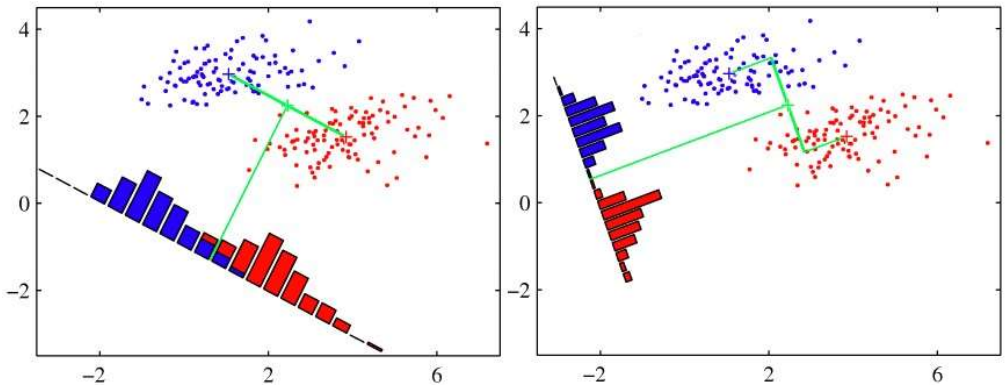
판별 분석(linear discriminant analysis)는 현대 통계학의 아버지 로널드 피셔(Ronal Fisher)가 만든 방법이다. 즉, 아주 오래된 방법이다.

판별 분석의 아이디어는 두 클래스의 중심점, 또는 평균이 서로 멀게 만드는 w가 좋은 파라미터라는 데서 출발한다.

예를 들어 $w_1 = 1, w_2 = -1$인 경우라면 y가 아래와 같은 형태로 변한다. 아무리 b를 잘 정해주더라도 좋은 경계를 찾기는 어려워 보인다. 중간의 굵은 실선은 빨강 클래스와 파랑 클래스의 중심을 똑같이 지나간다. 즉, 이 w 아래에서는 두 클래스의 평균이 같은 y를 가지기 때문에 적합한 경계선을 찾을 수 없다.



그러나 단순히 두 클래스의 평균을 멀게 만드는 것만으로는 충분치 않다. 아래 그림을 보자.



왼쪽 그림이 오른쪽 그림보다 평균 간의 거리(짧은 초록선)는 더 멀다. 그렇지만 경계선(긴 초록선)은 오른 쪽이 더 적합해 보인다. 왼쪽 그림은 경계선을 넘어가는 경우가 있지만 오른쪽 그림은 그렇지 않다.

원인을 생각해보면 왼쪽 그림에서는 한 클래스 내의 y 값들이 넓게 퍼지는 반면에 오른쪽 그림에서는 좁게 모여있는 것을 볼 수 있다. 다시 말하면 한 클래스 안의 데이터들은 y 값의 차이가 적다.

정리하자면 선형 판별 분석은 클래스 **간**의 차이는 **크게**, 클래스 **내**의 차이는 **작게** 만드는 w를 가장 좋은 w로 본다. 이를 식으로 정리하자면 아래 J를 **최대화**하는 w를 찾는 것이다.

$$J = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

- m_i는 i 번째 클래스의 평균

- s_i는 i번째 클래스의 표준편차.

분산 분석(ANOVA)을 알고 있는 사람은 용도는 다르지만 논리가 비슷하다 생각이 들 것이다. 분산 분석도 로날드 피셔가 개발한 방법론이기 때문이다.

3.3.2. 로지스틱 회귀

이번에는 다른 접근을 취해보자. 판별 분석에서 y에는 특별한 의미가 없다. 단지 분류를 하기 위해 편의상 만들어낸 값에 지나지 않는다.

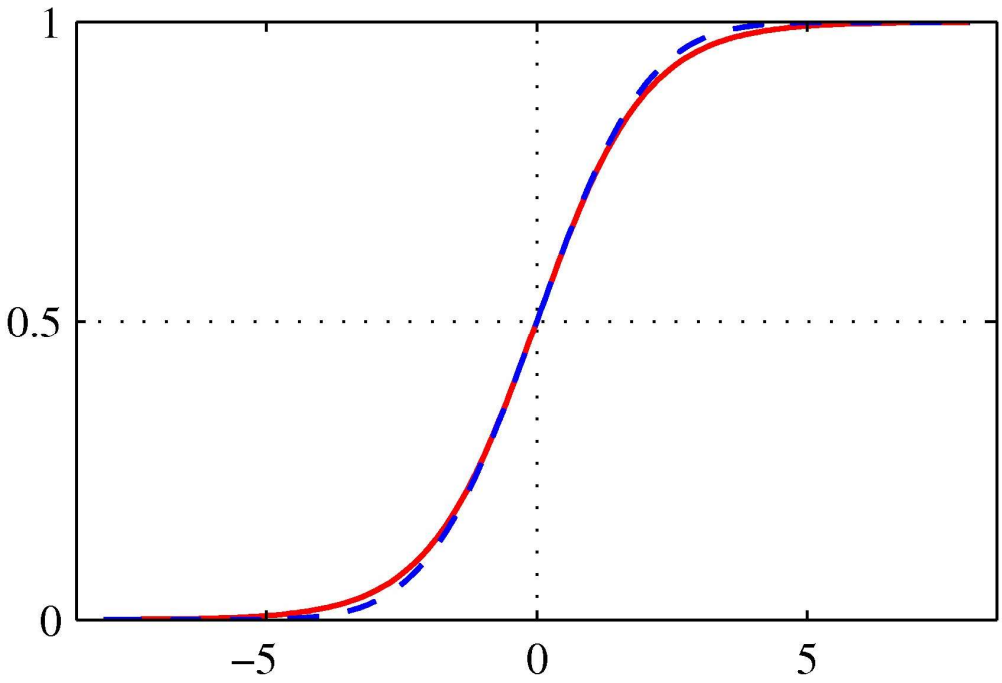
로지스틱 회귀는 이렇게 의미 없는 y 대신에 데이터 x가 특정 클래스에 속할 확률 p를 예측한다. 그리고 $p = .5$일 때, 즉 확률이 50%:50%일 때를 기준으로 경계선을 긋는다.

사실은 로지스틱 회귀도 식 자체는 선형 분류 모형과 같이 $y = wx + b$의 형태이다. 여기서 y를 로지스틱(logistic) 함수에 한 번 더 집어넣어주는 것이 차이다.

로지스틱 함수는 다음과 같은 형태의 함수로

$$p = \frac{1}{1 + e^{-y}}$$

아래 그림의 빨간 선과 같은 형태를 가진다.



로지스틱 함수는 $(-\infty, \infty)$ 범위의 y를 $(0, 1)$까지 범위의 p로 찌그러트린다(squash). 그래서 스쿼시 함수라고도 하고 S자 형태로 생겼다 해서 시그모이드(sigmoid) 함수라고도 한다. 확률이 $[0, 1]$ 범위의 값이기 때문에 p는 확률로 해석할 수 있게 된다.

참고로 범위를 나타낼 때 둥근 괄호는 범위의 양쪽 끝을 포함하지 않는다는 의미이고, 각 괄호는 범위의 양쪽 끝을 포함한다는 의미이다.

시그모이드가 S자 형태로 생겼다는 뜻인 이유는 '오이드(-oid)'가 "~의 형태로 생긴"이라는 뜻의 어미이기 때문이다. '시그마(sigma)'는 라틴 문자 S에 해당하는 그리스 문자를 가리킨다. 인간형 로봇을 휴머노이드(humanoid)라고 한다.

그렇다면 로지스틱 회귀분석의 손실 함수는 어떻게 정의할까? 가장 대표적인 방법은 최대우도법(maximum likelihood)을 이용한다. 우도(likelihood)란 모형의 파라미터를 정했을 때, 학습용 데이터의 확률을 말한다. 최대우도법은 이 우도를 가장 크게 만드는 파라미터를 찾는 방법이다.

로지스틱 회귀 분석은 각각의 x에 대해 한 클래스에 속할 확률 p를 알려주므로 그 확률들을 모두 곱하면 우도를 구할 수 있다.

$$\prod_k \left[p_k^{C_k} (1-p_k)^{1-C_k} \right]$$

그런데 확률의 곱셈은 번거로우므로 로그(log)를 사용해서 덧셈으로 바꿔서 **로그 우도**를 사용한다.

$$\sum_k \left[C_k \log p_k + (1-C_k) \log (1-p_k) \right]$$

- C_k: k번째 데이터의 클래스(0 또는 1)
- p_k: k번째 데이터의 클래스가 1일 확률

위의 로그 우도는 정보 이론의 크로스 엔트로피와 형태가 같기 때문에 크로스 엔트로피라고도 한다.

로지스틱 회귀에는 판별 분석에 비해 몇 가지 장점이 있다. 그 중에 한 가지는 상대적으로 이상점에 강하다는 것이다. 위의 예에서 오른쪽 먼 곳에 파란 점이 하나 더 있다고 생각해보자. 이 점은 경계선과 별 상관 없는 먼 곳에 있지만 파란색의 평균을 오른쪽으로 끌어 당기고 분산을 키우게 된다. 따라서 경계선도 변하게 된다.

로지스틱 회귀에서는 각 점의 확률만을 고려하기 때문에 먼 곳의 파란 점은 단지 우도가 높은 데이터가 한 건 추가된 것에 불과하고, 경계선에는 별 영향을 주지 않는다. 따라서 로지스틱 회귀는 판별 분석보다 좀 더 이상점에 강하게 된다.

하지만 로지스틱 회귀에도 약점은 있다. 흔히 있는 경우는 아니지만 두 클래스가 겹치는 부분이 전혀 없이 완전히 깨끗하게 구별되면, 각 점의 확률이 1이나 0이 되야하기 때문에 y가 무한히 커지거나 작아지는 형태로 발산 한다. 판별 분석에는 이러한 약점이 없다.

3.4. 혼돈 행렬

분류 문제의 경우 크로스 엔트로피는 그 값의 크기를 직접적으로 이해하기가 쉽지 않다. 그래서 혼돈 행렬(confusion matrix)이라는 형태로 성능을 나타낸다.

이항 분류 문제의 경우 두 가지 클래스 중에 한 클래스에 더 관심이 많은 경우가 보통이다. 질병 진단의 경우에는 질병 vs. 건강 중 질병에 더 관심이 있을 것이다. 화재 경보기라면 화재 vs. 일상 중 화재에 더 관심이 있을 것이다. 이렇게 관심이 더 있는 클래스를 양성(positive), 다른 클래스를 음성(negative)이라고한다.

양성이라고 예측했을 때 양성 경우는 진양성(true positive), 마찬가지로 음성이라고 예측했을 때 음성인 경우는 진음성(true negative)라고 한다. 양성이라고 예측했으나 실제로는 음성인 경우는 위양성(false positive) 또는 오경보(false alarm)이라 하고, 음성이라 예측했는데 실제로는 양성인 경우는 위음성(false negative) 또는 놓침(miss)라고 한다.

예측 \ 실제	양성	음성
양성	진양성	위양성
음성	위음성	진음성

이렇게 4가지 경우가 생기기 때문에 이들을 조합해서 여러 가지 지표를 만들 수 있다. 자주 사용되는 지표로는 정확도(accuracy), 정밀도(precision), 재현도(recall)가 있다.

정확도는 전체 4가지 경우 중에 진양성과 진음성이 차지하는 비율이다. 이는 양성과 음성을 가리지 않고 단순히 맞춘 경우만을 따진다.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{N}$$

정밀도는 양성으로 예측한 경우(진양성 + 위양성) 중에 진양성의 비율이다. 이는 양성 예측이 얼마나 정확한지를 따진다.

$$\mathrm{PRC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

양성 예측의 경우 특별한 행동이 뒤따르는 경우가 많다. 질병 진단에는 치료가 필요하고, 화재 경보에는 대피가 필요하다. 만약 정밀도가 낮다면 불필요한 치료나 대피를 하게 된다.

또한 음성 예측의 경우에는 그 예측이 맞았는지 틀렸는지 확인 어려운 경우도 많다. 신입 사원 채용의 경우 탈락시킨(음성예측) 지원자가 유능한 지원자였는지는 알 길이 없을 것이다. 공항 세관에서도 통과시킨(음성예측) 입국자가 밀수범이었는지 사후에 알기는 어렵다. 이런 경우 음성의 진위여부를 알 수 없으므로 실질적으로 확인할 수 있는 것은 정밀도 밖에 없을 수 있다.

재현도는 실제 양성(진양성 + 위음성) 중에 진양성의 비율이다. 이는 실제 양성을 얼마나 예측해낼 수 있는지를 따진다. 의학 등 분야에서는 민감도(sensitivity)라는 표현을 더 많이 사용한다.

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

전염병이 퍼지는 경우에 조금만 유사 증상을 보여도 환자를 격리시키는 경우가 있다. 격리된 환자

들 중에 상당수는 감염되지 않은 것으로 밝혀지기도 하지만, 전염병은 한 명이라도 놓치면(위음성) 피해가 막심하기 때문이다. 이럴 경우에는 재현도를 높이는 것이 중요하다.

정밀도와 재현도가 모두 요구될 때는 이 둘의 조화 평균인 지표 **F1**을 사용한다. 조화평균이란 역수의 평균의 역수로 다음과 같이 계산한다.

$$
\begin{aligned}
F_1 &= \frac{1}{\frac{\frac{1}{\mathrm{PRC}}+\frac{1}{\mathrm{REC}}}{2}} \\
&= \frac{2}{\frac{1}{\mathrm{PRC}}+\frac{1}{\mathrm{REC}}} \\
&= \frac{2}{\frac{\mathrm{PRC}+\mathrm{REC}}{\mathrm{PRC}\cdot\mathrm{REC}}} \\
&= \frac{2\cdot\mathrm{PRC}\cdot\mathrm{REC}}{\mathrm{PRC}+\mathrm{REC}}
\end{aligned}
$$

흔히 사용하는 산술 평균이 아닌 조화 평균을 사용하는 이유는 속도, 비율 등에서 분자가 같고 분모가 다른 경우가 많아 조화 평균이 더 정확하기 때문이다. 예를 들어 120Km 떨어진 지점에 120Km/h로 갔다가 60Km/h로 돌아왔다고 하자. 갈 때 1시간, 올 때 2시간 합쳐서 3시간 동안 왕복 240Km를 달린 셈이므로 평균 속력은 산술 평균인 90Km/h가 아니라 조화 평균인80Km/h가 된다.

정밀도와 재현도도 식을 보면 분자는 똑같이 진양성이고, 분모에만 위양성이 들어가느냐 위음성이 들어가느냐로 달라지는 것을 볼 수 있다. F1은 다시 정리하면 아래 식과도 같다.

$$
F_1 = \frac{TP}{\mathrm{TP}+\frac{\mathrm{FP}+\mathrm{FN}}{2}}
$$

추가로 **특이도**(specificity)라는 지표도 있다. 이 지표는 전체 음성 중에 진음성의 비율로 계산한다.

$$
\mathrm{SPC} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}
$$

기타 다른 지표에 대해서는 아래 그림을 참고하라.

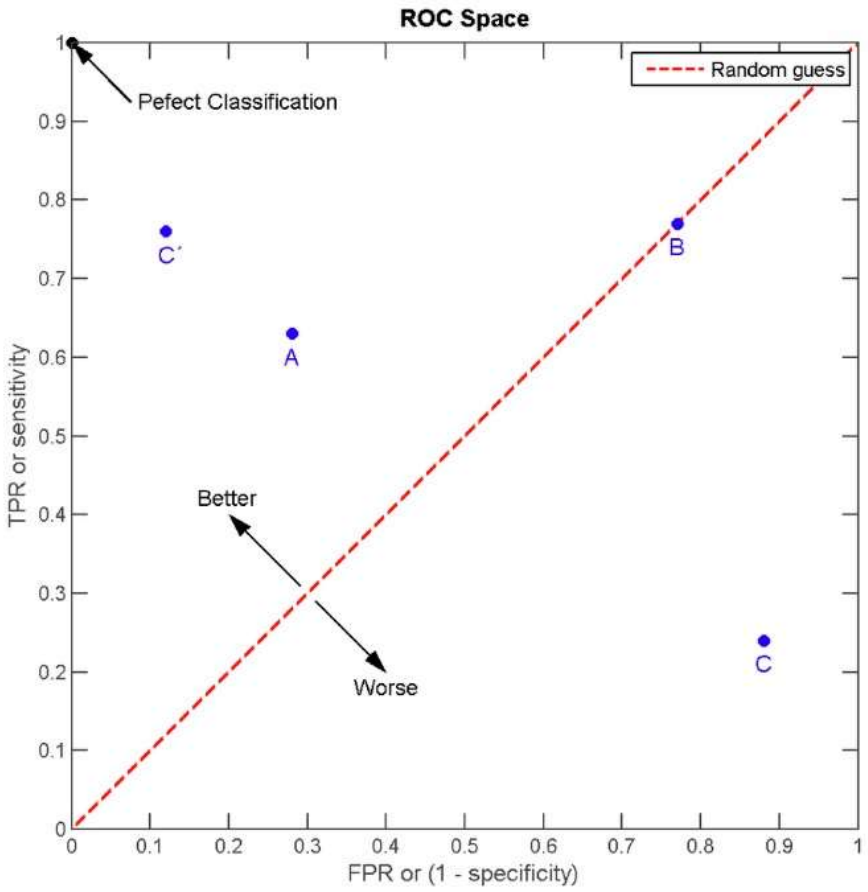
		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

3.4.1. 수신자 조작 특성 곡선과 곡선하 면적

정밀도, 재현도, 특이도는 상충되는 관계에 있다. 양성 예측을 적극적으로 하면 진양성(TP)이 늘어날 수 있지만 위양성(FP)도 늘어날 수 있다. 재현도는 TP에만 영향을 받으므로 같이 올라가지만 FP가 분모에 들어가는 정밀도나 특이도는 반대로 떨어질 수 있다. 양성 예측을 소극적으로 하면 반대의 결과가 벌어진다.

이상적으로는 양성 예측을 높일 때 진양성만 늘어나고 위양성은 늘어나지 않는 것이 바람직하다. 이를 확인하기 위한 그래플를 수신자 조작 특성 곡선(Reciever Operating Charateristic Curve, 이하 **ROC 곡선**)이라고 한다. 신호 이론(signal theory)에서 기원했기 때문에 이와 같은 이름이 붙었다.

ROC 곡선은 세로축은 재현도를 나타낸다. 가로축은 1 - 특이도를 나타내는데 이는 음성 중 위양성의 비율(FPR)에 해당한다.



예측을 무작위로 할 경우에 ROC 곡선은 위 그림의 빨간 점선에 가까운 형태가 된다. 예를 들어 무작위로 30%를 양성으로 꼽는다고 하자. 그러면 실제 양성인 경우 중에 30%는 양성으로 예측하고, 실제 음성인 경우 중에도 30%는 양성으로 예측하게 된다. 그러면 TPR과 FPR이 모두 30%가 되는 것이다.

예측을 좀 더 잘 한다면 ROC 곡선은 빨간 점선보다 위쪽으로 볼록한 형태의 곡선이 된다.

ROC 곡선을 하나의 수치로 나타낸 것이 곡선하 면적(Area Under Curve, 이하 AUC)이다. 위의 사각형은 가로 세로 1이므로 면적이 1이다. 빨간선은 그 절반이므로 0.5가 된다. ROC 곡선이 완벽한 형태가 될 수록 AUC는 1에 가까워지고 무작위적 예측에 가까울 수록 0.5에 가까워지게 된다.

3.5. 선형 분류

3.5.1. 데이터 불러오기

```
red.url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/wi
white.url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
```

```
red.wine = read.csv(url(red.url), sep = ';')
white.wine = read.csv(url(white.url), sep = ';')
```

```
red.wine$color = 'red'
white.wine$color = 'white'
```

```
wine = rbind(red.wine, white.wine)
```

```
head(wine)
```

```
fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.4             0.70        0.00            1.9     0.076
2           7.8             0.88        0.00            2.6     0.098
3           7.8             0.76        0.04            2.3     0.092
4          11.2             0.28        0.56            1.9     0.075
5           7.4             0.70        0.00            1.9     0.076
6           7.4             0.66        0.00            1.8     0.075
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  11                   34  0.9978 3.51      0.56     9.4
2                  25                   67  0.9968 3.20      0.68     9.8
3                  15                   54  0.9970 3.26      0.65     9.8
4                  17                   60  0.9980 3.16      0.58     9.8
5                  11                   34  0.9978 3.51      0.56     9.4
6                  13                   40  0.9978 3.51      0.56     9.4
  quality color
1       5   red
2       5   red
3       5   red
4       6   red
5       5   red
6       5   red
```

```
tail(wine)
```

```
     fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
6492           6.5             0.23        0.38            1.3     0.032
6493           6.2             0.21        0.29            1.6     0.039
6494           6.6             0.32        0.36            8.0     0.047
6495           6.5             0.24        0.19            1.2     0.041
6496           5.5             0.29        0.30            1.1     0.022
6497           6.0             0.21        0.38            0.8     0.020
     free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
6492                  29                  112 0.99298 3.29      0.54     9.7
6493                  24                   92 0.99114 3.27      0.50    11.2
6494                  57                  168 0.99490 3.15      0.46     9.6
6495                  30                  111 0.99254 2.99      0.46     9.4
6496                  20                  110 0.98869 3.34      0.38    12.8
6497                  22                   98 0.98941 3.26      0.32    11.8
     quality color
6492       5 white
6493       6 white
6494       5 white
6495       6 white
6496       7 white
6497       6 white
```

3.5.2. 시각화

```
library(dplyr)
```

```
indep = which(names(wine) != 'color')
```

```
set.seed(1234)
```

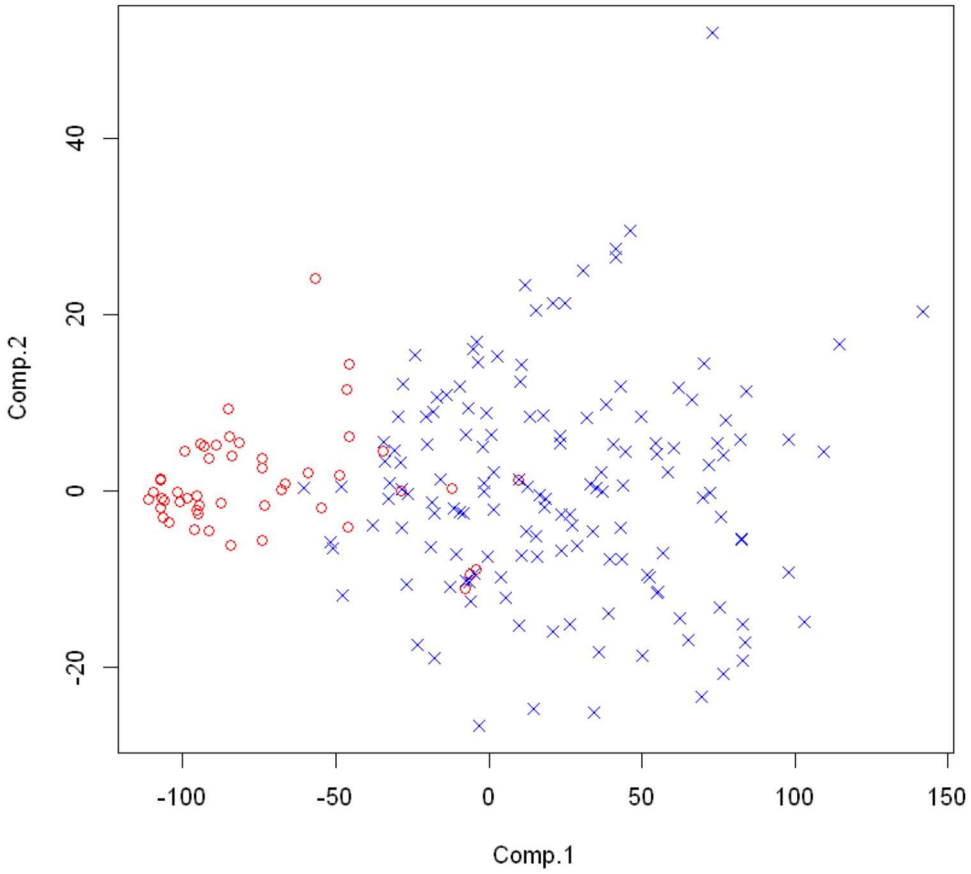
```
rows = sample(1:nrow(wine), 200)
```

```
pc = princomp(wine[indep])
```

```
col = recode(wine$color[rows], red=2, white=4)
pch = recode(wine$color[rows], red=1, white=4)
```

```
plot(pc$scores[rows,c(1,2)], col=col, pch=pch)
```

```
plot without title
```



3.5.3. 판별 분석

```
library(MASS)
```

```
model = lda(color ~ ., wine)
```

```
model
```

```
Call:
lda(color ~ ., data = wine)

Prior probabilities of groups:
      red     white
0.2461136 0.7538864

Group means:
      fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
red        8.319637        0.5278205   0.2709756       2.538806 0.08746654
white      6.854788        0.2782411   0.3341915       6.391415 0.04577236
      free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
red              15.87492             46.46779 0.9967467 3.311113 0.6581488
white            35.30808            138.36066 0.9940274 3.188267 0.4898469
       alcohol  quality
red   10.42298 5.636023
white 10.51427 5.877909

Coefficients of linear discriminants:
                              LD1
fixed.acidity           0.33203356
volatile.acidity       -3.22217380
citric.acid             0.86454436
residual.sugar          0.35714801
chlorides              -5.15774110
free.sulfur.dioxide    -0.01857162
total.sulfur.dioxide    0.01982966
density              -919.18988007
pH                      1.15987401
sulphates              -0.77838019
alcohol                -0.79641352
quality                -0.11587257
```

3.5.4. 로지스틱 회귀

```
y = recode(wine$color, red=1, white=0)
```

```
X = as.matrix(wine[indep])
```

```
model = glm(y ~ X, family=binomial(link='logit'))
```

```
summary(model)
```

```
Call:
glm(formula = y ~ X, family = binomial(link = "logit"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.8678  -0.0582  -0.0188  -0.0012   5.6178

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.876e+03  1.868e+02 -10.043  < 2e-16 ***
Xfixed.acidity        -4.005e-01  2.334e-01  -1.716   0.0861 .
Xvolatile.acidity      6.722e+00  1.061e+00   6.337 2.34e-10 ***
Xcitric.acid          -2.617e+00  1.185e+00  -2.209   0.0272 *
Xresidual.sugar       -9.562e-01  1.012e-01  -9.449  < 2e-16 ***
Xchlorides             2.201e+01  3.984e+00   5.524 3.31e-08 ***
Xfree.sulfur.dioxide   6.080e-02  1.456e-02   4.177 2.96e-05 ***
Xtotal.sulfur.dioxide -5.229e-02  4.990e-03 -10.479  < 2e-16 ***
Xdensity               1.875e+03  1.904e+02   9.846  < 2e-16 ***
XpH                   -1.959e+00  1.424e+00  -1.376   0.1689
Xsulphates             2.693e+00  1.249e+00   2.156   0.0311 *
Xalcohol               1.792e+00  2.795e-01   6.412 1.43e-10 ***
Xquality               4.339e-01  2.041e-01   2.126   0.0335 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7250.98  on 6496  degrees of freedom
Residual deviance:  424.23  on 6484  degrees of freedom
AIC: 450.23

Number of Fisher Scoring iterations: 9
```

3.5.5. caret

```
library(caret)
```

3.5.6. 데이터 분할

```
idx = createDataPartition(wine$color, p=.8, list=F)
```

```
data.train = wine[idx, ]
data.test = wine[-idx, ]
```

3.5.7. 훈련

```
trControl = trainControl(method='none')
```

판별 분석

```
lda.model = train(
    color ~ .,
    data = data.train,
    metric = 'Accuracy',
    preProc = c('center', 'scale'),
    trControl = trControl,

    method='lda'
)
```

로지스틱 회귀

```
lg.model = train(
    color ~ .,
    data = data.train,
    metric = 'Accuracy',
    preProc = c('center', 'scale'),
    trControl = trControl,

    method = 'glmnet',
    tuneGrid = data.frame(.alpha=0, .lambda=0)
)
```

최근접 이웃

```
knn.model = train(
    color ~ .,
    data = data.train,
    metric = 'Accuracy',
    preProc = c('center', 'scale'),
    trControl = trainControl(method='none'),

    method = 'knn',
    tuneGrid = data.frame(.k=15)
)
```

3.5.8. 성능 검증

LDA

```
y.lda = predict(lda.model, data.test)
```

```
confusionMatrix(y.lda, data.test$color)
```

```
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   317     2
     white   2   977

               Accuracy : 0.9969
                 95% CI : (0.9921, 0.9992)
    No Information Rate : 0.7542
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9917
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9937
            Specificity : 0.9980
         Pos Pred Value : 0.9937
         Neg Pred Value : 0.9980
             Prevalence : 0.2458
         Detection Rate : 0.2442
   Detection Prevalence : 0.2458
      Balanced Accuracy : 0.9958

       'Positive' Class : red
```

```
y.lg = predict(lg.model, data.test)
```

```
confusionMatrix(y.lg, data.test$color)
```

```
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   308     4
     white  11   975

               Accuracy : 0.9884
                 95% CI : (0.981, 0.9935)
    No Information Rate : 0.7542
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9686
 Mcnemar's Test P-Value : 0.1213

            Sensitivity : 0.9655
            Specificity : 0.9959
         Pos Pred Value : 0.9872
         Neg Pred Value : 0.9888
             Prevalence : 0.2458
         Detection Rate : 0.2373
   Detection Prevalence : 0.2404
      Balanced Accuracy : 0.9807

       'Positive' Class : red
```

```
y.knn = predict(knn.model, data.test)
```

```
confusionMatrix(y.knn, data.test$color)
```

```
Confusion Matrix and Statistics

          Reference
Prediction red white
     red   313     6
     white   6   973

               Accuracy : 0.9908
                 95% CI : (0.9839, 0.9952)
    No Information Rate : 0.7542
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9751
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9812
            Specificity : 0.9939
         Pos Pred Value : 0.9812
         Neg Pred Value : 0.9939
             Prevalence : 0.2458
         Detection Rate : 0.2411
   Detection Prevalence : 0.2458
      Balanced Accuracy : 0.9875

       'Positive' Class : red
```