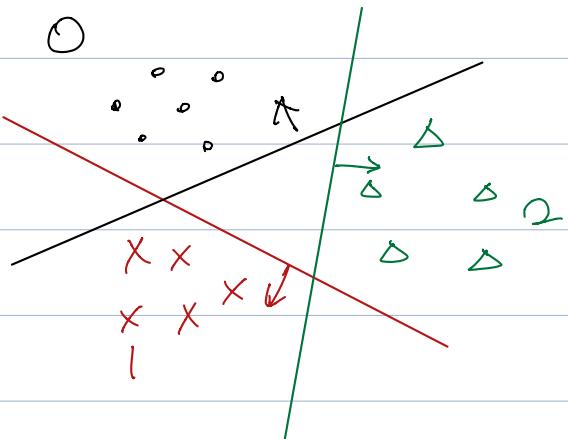


② OVR (One-versus-Rest) or OvA (One versus All)

다중 분류 (multi-class classification)에서 OVR(One versus Rest)란?

OVR은 이진 분류기 (binary classifier)를 이용하여 다중 분류기를 만들 때 쓰는 전략이다. 예를 들어 아래와 같이 세 가지 클래스를 가진 데이터가 있다.



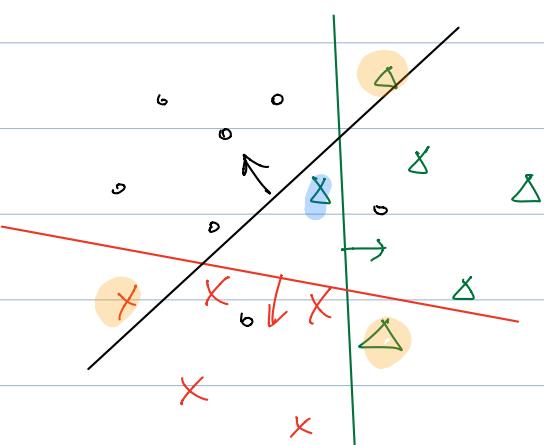
이진 분류기를 이용해 클래스 0을 분류하기
위해서는 클래스 0과 (One) 나머지 클래스들을
(Rest) 구별하는 분류기를 학습시키아 한다.

그럼의 같은 선이 클래스 0을 위해 학습된 분류기다.

빨간선은 클래스 1과 나머지를 분류하고

녹색선은 클래스 2와 나머지를 구별할 수 있다.

하지만 위와 같은 상황은 이승적일 경우고 대개는 다음과 같이 구분하기 어려운 상황이 많다.



여기서의 문제는 몇몇 샘플이 분류가 잘못된 것인 아나라

여러 분류기에서 동시에 "True" 판정을 받은 샘플들이 있다는 것이다.

이런 경우에는 각 분류기에서 "True"라는 결과만 보는 것이 아나라

activation이나 $P(y=True|x)$ 같은 확률값을 비교하여 더 높은 값이 나오는 쪽을 최종 분류 결과로 선택한다.

혹은 반대로 어느 분류기에서도 "True"가 나오지 않는 샘플이 있을 수 있다.

그런 경우에도 분류기들 사이에서 상대적으로 높은 확률이 나오는 쪽을 선택한다.

책 p.100 아래에 각 클래스에 대한 확률을 확인할 수 있는 predict_proba() 란 함수가 나온다. 결과적으로 보면 OVR은 클래스의 개수 만큼 분류기를 학습시키고

어떤 샘플을 분류할 때 각 분류기의 확률을 비교하여 분류한다.

① Perceptron 모델이 수렴할 수 없는 경우?

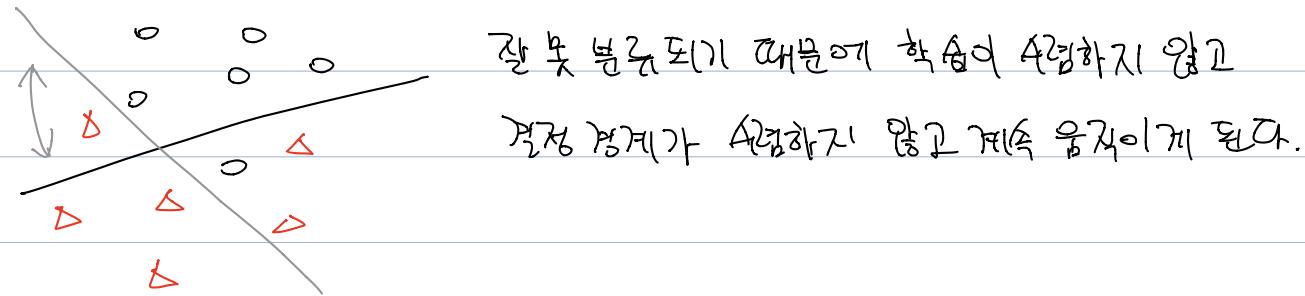
책 p.89에 “퍼셉트론 규칙은 ... 결과가 선형적으로 구분되지 않을 때는 수렴하지 않는다”는데 왜 그걸까?

퍼셉트론에서 파라미터 업데이트 식은 다음과 같다.

$$\omega := \omega + \Delta\omega = \omega + \eta (y^i - \hat{y}^i) x^i \quad (\text{perceptron})$$

퍼셉트론은 $y^i = \hat{y}^i$ 인 경우에는 학습하지 않고 오직 레이블(y^i)과 출력(\hat{y}^i)이 다른 경우, 즉 분류를 잘못한 경우에만 학습이 된다.

그런데 완전한 선형분류가 되지 않는 경우에는 아무리 학습해도 1개 이상은



반면에 logistic regression의 업데이트 식은 다음과 같다.

$$\omega := \omega + \Delta\omega = \omega + \eta \sum_i (y^i - \phi(z^i)) x^i, \quad \phi(z^i) = \frac{1}{1+e^{-z^i}}$$

퍼셉트론과의 차이점은 두 가지가 있다. 배치(batch) 학습을 하는 것과 학습의 강도($y^i - \phi(z^i)$)가 실수로 나온다는 것이다. 이렇게 함으로써 학습을 계속 진행하면 $\Delta\omega \approx 0$ 으로 수렴하게 된다. 즉 0을 (로 분류해서 발생하는 $\Delta\omega$)와 (을 0으로 분류해서 발생하는 $\Delta\omega$)가 정확히 반대방향으로 균형을 이루게 된다.

하지만 perceptron의 경우 배치 학습을 한하고 하더라도 학습의 강도($y^i - \hat{y}^i$)가 선형하게 조절되지 않으므로 (-2, 0, 2 등 중 하나) $\Delta\omega$ 가 0으로 수렴할 수 없다.

참고로 Adaline은 batch로 학습하고 학습의 강도가 선수가 때문에 선형분류가 불가능한 문제에서도 수렴 가능하다.

$$\omega := \omega + \Delta\omega = \omega + \eta (y^i - \phi(z^i)) x^i \quad (\phi(z^i) = \omega^T x^i)$$

⑥ Logistic Regression

Regression은 원래 입력으로부터 실수 출력을 내는 문제를 말한다. 하지만 logistic regression은 어떤 샘플의 들어왔을 때 샘플이 각 클래스에 속할 "확률"을 출력하는 regression 문제로 보고 결과적으로 확률이 높은쪽으로 분류 결과를 만든다.

이전 분류에서 $y \in \{0, 1\}$ 두 종류의 값을 갖고 $P(y=1|x) = p$,

$P(y=0|x) = 1-p$ 일 때 logistic regression은 activation의 도움과 같이 나오도록 파악하여 w 를 학습한다.

$$\text{activation: } \phi(z) = p = P(y=1|x) = \begin{cases} 1 & \text{if } y=1 \\ 0 & \text{if } y=0 \end{cases}$$

▷ Note

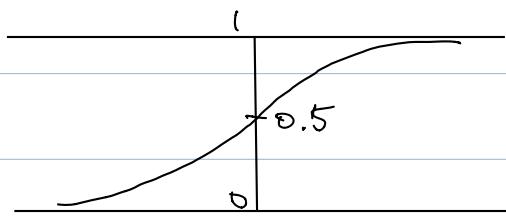
- 확률에서 $P(y|x)$ 는 x 에 따른 y 의 분포 확률이다. $P(y=y_0|x=x_0)$ 는 x 가 x_0 일 때 y 가 y_0 일 확률이다.

- 이전 분류기 때문에 y 는 0 또는 1의 값을 가지고 모든 경우에 대한 확률은 1이다. $\sum_c P(y=c|x) = 1$ (c 는 y 가 가질 수 있는 값)
즉 $P(y=0|x) + P(y=1|x) = 1$ 이므로 $P(y=1|x) = p$ 이면
 $P(y=0|x) = 1 - P(y=1|x) = 1 - p$ 다.

$z = w^T x$ (net input)은 값의 제약이 없으므로 $(-\infty, \infty)$ 사이의 값을 가질 수 있는데 activation function을 거쳐 나온 확률 p 는 $[0, 1]$ 사이의 값을 가져야 한다.

$z \in (-\infty, \infty) \rightarrow z \in [0, 1]$ mapping에 적합한 함수가 sigmoid 함수다.

$$\text{logistic Sigmoid function: } \phi(z) = \frac{1}{1+e^{-z}} = p$$



입력 범위가 무한하고 입력의 절대값이

큰 영역에서는 출력이 크기 변하지 않고

입력이 0 근처일 때 민감하게 반응한다.

반대로 확률 P 로부터 구를 구하는 함수를 logit 함수라 한다.

$$\phi(z) = \frac{e^z}{1+e^z} = P$$

$$1 = P(1+e^{-z}) \rightarrow 1-P = Pe^{-z} \rightarrow e^{-z} = \frac{1-P}{P}$$

$$\rightarrow -z = \log \frac{1-P}{P} \rightarrow z = -\log \frac{1-P}{P} = \log \frac{P}{1-P}$$

(logit function or log odds : $\text{logit}(P) = \log \frac{P}{1-P} = z$)

여기서 $\frac{P}{1-P}$ 는 odds ratio (승률, 배당률) 라 부르고 여기에 log까지 쓰운 것은 log odds 혹은 logit이다.

여기서 혼동하지 말아야 할 것은 $z = \log \frac{P}{1-P}$ 는 P 와 p 의 논리적 관계일

뿐이다. 실제 net input $z = w^T x$ 로 계산한다.

즉 logistic regression의 학습목표는

" $y=1$ 일 때 $P=1$ 이 되고 $y=0$ 일 때 $P=0$ 이 되는 것"

\Leftrightarrow " $y=1$ 일 때 $z=w^T x \rightarrow b=0$ 되고 $y=0$ 일 때 $z=w^T x \rightarrow -b$

되도록 w 를 학습시키는 것" 이 된다.

④ Bayes' Rule (Bayes' Theorem) in Probabilistics

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \quad y: 사건, x: 조건$$

$$\text{유도: } ① P(y|x) = \frac{P(x,y)}{P(x)}$$

특정 x 조건에서 어떤 사건 y 가 발생할 확률은 x 와 y 가 동시에 일어날

확률을 x 가 발생할 전체 확률로 나눈 것과 같다.

$$② P(x,y) = P(y|x)P(x) = P(x|y)P(y)$$

①에서 x, y 관계 (사건과 조건)를 바꿔도 동일한 관계가 유지된다.

$$①, ②를 모두 적용하면 \phi(P(y|x)) = \frac{P(x,y)}{P(x)} = \frac{P(x|y)P(y)}{P(y)}$$

Bayes' Rule의 각 부분을 설명하면

- $p(y|x)$: posterior라 불리며 조건에 따른 사건변수의 확률 분포이며 궁극적으로 구하고자 하는 목표다.

- $p(x|y)$: likelihood 라 불리며 조건과 사건의 바뀐 형태인데 어떤 사건이 일어났을 때 조건변수의 분포 확률이다.

- $p(y)$: prior 라 불리며 조건과 상관없는 y의 분포 확률이다.

- $p(x)$: 조건 x 자체의 확률 $p(x)$ 는 확률 전체 합을 1로 만들여주는 역할만 한다.

Bayes Rule을 해석하면 사건에 알고 있던 사건의 확률 분포 (prior)에 어떤 사건이 발생했을 때의 그 값의 분포 (likelihood)를 곱하면 사건 후 조건에 따른 사건의 확률 (posterior)을 구할 수 있다.

보통의 경우 prior $p(y)$ 는 조건이 되어지지 않았으므로 특정하기 어렵고 따라서 모든 y에 대해 균일한 값을 가지는 경우가 많다.

그러면 $p(y|x) \propto p(x|y)$ 가 된다. 즉 posterior는 likelihood에 비례한다.

유래의 문제가 $\hat{y} = \arg \max_y p(y|x)$ 였다면 posterior를 적용 구하기

어려운 경우에는 $\hat{y} = \arg \max_y p(x|y)$ 를 구해도 같은 결과가 나온다.

어려운 문제를 해결하는 방식은 "maximum likelihood"라 한다.

④ Maximum Likelihood 와 Logistic Regression

Logistic Regression은 입력 기와 파라미터 w가 주어졌을 때 분류 결과 y를 찾는 것, 즉 $p(y|w,x)$ 를 구하는 것이라고 생각하기 쉬우나 그것은 학습이 끝난 후 예측 (Prediction) 할 때의 문제다. 학습할 때의 문제는 입력 특성 x와 그에 해당하는 출력 y, 즉 데이터가 주어졌을 때 그 데이터를 최대한

잘 설명해 줄 수 있는 모델 파라미터 w를 찾는 것이다. 즉 학습 문제의 posterior는 $p(w|x,y)$ 이고 학습 문제는 다음과 같이 정의할 수 있다.

$$\hat{\omega} = \arg \max_{\omega} p(\omega | x, y)$$

이 posterior를 직접 구하려면 x, y 를 통해서 ω 의 예측값을 구하고 "실제" ω 값과의 차이에 따라 확률을 계산해야 한다. 하지만 logistic regression 모델은 x, y 를 통해서 y 를 예측하는 모델이고 "실제" ω 값이라는 개념이 없다. 따라서 이 문제는 likelihood를 통해서 계산하는 것이 낫다.

$$\hat{\omega} = \arg \max_{\omega} p(y | \omega, x) \quad (\text{Maximum Likelihood})$$

Likelihood $P(y | \omega, x)$ 는 다음과 같이 모델링 할 수 있다.

$$L(\omega) = p(y | \omega, x) = \prod_i p(y^i | x^i, \omega)$$

학습데이터셋에 출력 y 가 여러개 있고 각각의 사건 (e.g. 각 Iris 꽃의 종류)이 서로 독립적이라면 여러 사건이 동시에 일어날 확률은 각 사건의 확률의 곱과 같다.

If y_1, y_2, \dots, y_n are independent, $p(y_1, \dots, y_n) = \prod_i p(y_i)$

$$p(y^i | x^i, \omega) = \phi(z^i), \quad p(y^i = 0 | x^i, \omega) = 1 - \phi(z^i)$$

$$p(y^i | x^i, \omega) = \begin{cases} \phi(z^i), & \text{if } y=1 \\ 1 - \phi(z^i), & \text{if } y=0 \end{cases} = \phi(z^i)^{y^i} (1 - \phi(z^i))^{1-y^i}$$

$$\begin{aligned} \therefore L(\omega) &= p(y | \omega, x) = \prod_i p(y^i | x^i, \omega) \\ &= \prod_i \phi(z^i)^{y^i} (1 - \phi(z^i))^{1-y^i} \end{aligned}$$

Likelihood 함수에 log를 써운 것을 log likelihood라고 한다.

log 변환을 하는 이유는

- 원래 함수의 증감 형태와 convex/concave 성질을 유지시키고
- 극점의 위치를 유지해서 최적해가 원래 함수와 같게 나오고
- 곱셈으로 표현된 식을 선형합의꼴로 풀어준다.

$$\hat{\omega} = \arg \max_{\omega} L(\omega) = \arg \max_{\omega} \log L(\omega)$$

Log likelihood $l(\omega)$ 는 다음과 같다.

$$l(\omega) = \log L(\omega) = \sum_i [y^i \log(\phi(z^i)) + (1-y^i) \log(1-\phi(z^i))]$$

$$\text{최적 파라미터는 } \hat{\omega} = \arg_{\omega}^{\max} l(\omega) = \arg_{\omega}^{\min} J(\omega)$$

비용함수는 보통 최소화(내려가기) 때문에 $J(\omega) = -l(\omega)$ 로 정의하여

logistic regression은 다음 $J(\omega)$ 를 최소화하는 $\hat{\omega}$ 을 찾는 문제가 된다.

$$J(\omega) = \sum_i [-y^i \log(\phi(z^i)) + ((1-y^i) \log(1-\phi(z^i)))]$$