CH6: Training/Testing and Regularization

6.1 Concept of Bias and Variance

머신러닝의 목적은 모델에 데이터를 학습 시켜 더 좋은 모델을 만들어 내도록 하는 것이다. 그래서 우리는 사용하고자하는 데이터셋을 train set, test set 그리고 validation set으로 나눈다. 각각에 대해서 알아보자.

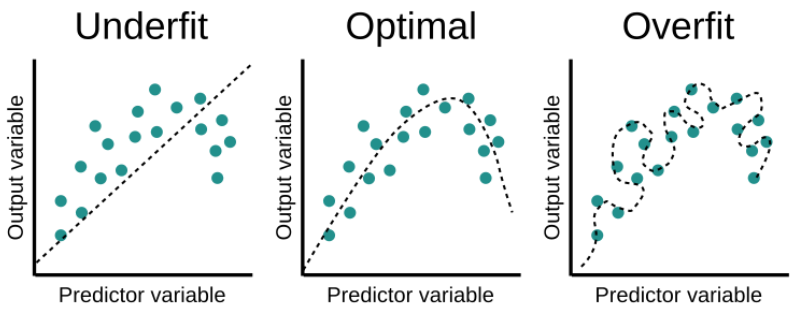
먼저 train set이란 학습 데이터이다. 이때, 명심해야 할 것은 '모델을 학습하는데 있어 오직 train set'만 사용해야 한다는 점이다. 우리는 train set을 통해서 1차 적으로 파라미터를 수정한다.

Validation set은 train set으로 만들어진 모델의 성능을 측정하기 위해 사용된다. 일반적으로 어떤 모델ㅇ리 가장 데이터에 적합한지 찾아내기 위해서 사양한 파라미터와 모델을 사용해보게 되며, 그 중 validation set으로 가장 성능이 좋았던 모델을 선택한다. 가끔 validation set이 없는 경우도 있는데, 이 경우에는 train set이 validation set의 역할까지 하는 것이다.

Test set은 사용할 모델이 결정 된 후, 마지막으로 모델의 성능을 예측하기 위해 사용된다. 이미 우리가 선택한 모델은 validation set에 적합된 상태이기 때문에, 새로운 데이터셋을 적합하면서 실제 모델의 성능을 측정하는 것이다. 즉, validation set은 여러번 사용하는 것이고, test set은 최종 모델에 딱 한 번 사용하는 것이다.

6.1.1 Over-Fitting and Under-Fitting

모델을 생성하는 방법에 따라 다양한 모델이 생성될 수 있다. 다음과 같은 상황이 주어졌다고 가정해보자. 우리는 N개의 데이터를 가지고 있고, 주어진 데이터에 대해 다항 선형 회귀 (Polynomial Linear Regression)을 사용하여 모델을 fit 했다고 가정해보자.

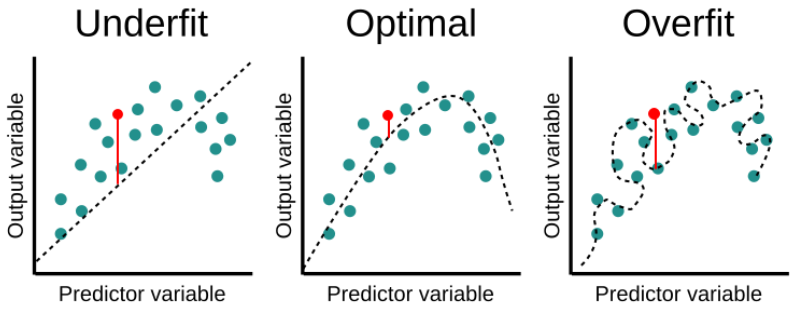


먼저 왼쪽 그림을 보면, 변수가 0 인 쪽의 데이터 몇 개에 대해서는 비교적 잘 근사하는 듯 하지만, 나머지 데이터는 회귀 그래프에서 떨어져 있다. 즉, 예측을 잘 못하고 있는 것을 볼 수 있다.

가운데 그려진 그래프는, 모데르이 전체 변수 범위에서 잘 예측을 하고 있음을 확인할 수 있다. 물론 100%의 적합도를 보이고 있지는 않지만, 새로운 데이터가 들어오더라도 어느정도는 예측할 수 있어보인다.

오른쪽에 그래프 같은 경우, 모델의 오차가가 없어보인다. 즉, 회귀 그래프가 모든 데이터를 지나가고 있다.

이제, 이 3개의 그래프에 대해 새로운 점이 추가되었을 때의 상황을 알아보자.



추가된 학습 데이터는 빨간색으로 표시되어 있다. 왼쪽 그래프의 경우, 오차가 상당히 많이 나타나는 것을 확인할 수 있다. 이는 훈련 데이터셋을 제대로 fitting하지 못했기 때문이다. 가운데 모델 같은 경우에는, 오차가 크지 않다. 이는 학습 데이터를 어느정도 근사 했기 때문에, 새로운 학습 데이터가 들어와도 어느정도 잘 수용하는 것이다. 오른쪽의 경우, 100%의 fitting 결과를 가졌기 때문에, 학습 데이터와의 오차가 없음에도 시험 데이터와는 큰 오차를 보이는 것을 확인할 수 있다.

위 이미지에서 왼쪽 그래프와 같이 너무 단순한 모델을 생성하여 학습 데이터와 잘 맞지 않을 때 모델이 과소적합(Underfitting) 되었다고 한다며, 이런 경우, 모델의 복잡도를 늘려줌으로써 문제를 해결해야 한다. 반대로 오른쪽은 과적합(Overfitting, 과대적합) 상태라고하며, 학습 데이터를 100% 학습한 경우를 뜻한다. 그리고 그래프에서 볼 수 있다싶이, 과소적합과 과대적합은 trade-off 관계가 존재한다.

위에서 살펴본 바와 같이 왼쪽 그래프와 오른쪽 그래프는 시험 데이터와의 오차가 발생하는 이유가 다르다. 전자는 훈련 데이터도 제대로 근사를 못할 만큼 모델이 단순한 것이 문제였다. 이런 경우에 발생하는 오차를 편향(Bias)이라고 한다. 후자는 훈련 데이터에 대해서는 근사를 매우 잘하지만 새로운 데이터가 들어왔을 때 제대로 근사하지 못한다는 문제가 있었는데, 이는 분산(variance) 때문이다.

6.1.2 Bais-Variance Tradeoff

단순한 모델에서 발생하는 편향을 줄이기 위해 모델의 복잡도를 늘리면 분산이 늘어나고, 복잡한 모델에서 분산이 줄이기 위해 모델을 단순화 하면 편향이 늘어나게 된다. 이러한 상황을 편향-분산 트레이드오프 (Bias-Variance Tradeoff)라고 한다.

Bias-Variance Tradeoff가 발생하는 과정을 수식적으로 알아보자. 일단 생성된 모델에서 발생한 오차의 합은 E_{out}이라고 하자. 이 오차는 알고리즘을 학습하는 과정(approximation)에서 발생하는 E_{in}과 데이터를 관측하는데 발생하는 오류인 Ω로 이루어져있다. 이를 수식적으로 나타내면 다음과 같다.

$$E_{out} \leq E_{in} + \Omega$$

우리가 찾고자 하는 목적 함수를 f라고 하고, 기계가 학습하여 생성한 모델의 함수를 g라고 하고, 존재하는 데이터 D에 대해서 생성되는 모델의 함수를 g^D라고 하자. 이런 경우, 데이터셋 D안에 있는 하나의 인스턴스에 의해 발생하는 오차는 다음과 같다.

$$E_{out}(g^D(x)) = E_X[(g^{(D)}(x) - f(x))^2]$$

이를 전체 데이터셋에 대한 오차를 구하는 수식으로 표현하면 다음과 같다.

$$E_D[E_{out}(g^{(D)}(x))] = E_D[E_X[g^{(D)}(x) - f(x))^2]] = E_X[E_D[(g^{(D)}(x) - f(x))^2]]$$

약간의 트릭을 사용하여 우의 식의 $E_D[(g^{(D)}(x) - f(x))^2]$ 부분을 간단하게 만들어 줄 수 있다.

$$\begin{aligned} E_D[(g^D(x) - f(x))^2] &= E_D[(g^D(x) - \bar{g}(x) + \bar{g}(x) - f(x))^2]] \\ &= E_D[(g^D - \bar{g}(x))^2 + (\bar{g}(x) - f(x))^2 + 2(g^D(x) - \bar{g}(x))(\bar{g}(x) - f(x))] \\ &= E_D[g^D(x) - \bar{g}(x))^2 + (\bar{g}(x) - f(x))^2)^2 + E_D[2(g^D(x) - \bar{g}(x))(\bar{g}(x) - f(x))] \\ &= E_D[g^D(x) - \bar{g}(x))^2] + (\bar{g}(x) - f(x))^2 \\ &\because \bar{g}(x) = E_D(g^D(x)) \end{aligned}$$

변형한 식을 원래 식에 대입하면, 데이터셋을 통해 생성된 모델로부터 발생하는 오차를 다음과 같이 나타낼 수 있다.

$$E_D[E_{out}(g^D(x))] = E_X[E_D[(g^D(x) - \bar{g}(x))^2] + (\bar{g}(x) - f(x))^2]$$

유도된 식에서 $E_D[(g^D(x) - \bar{g}(x))^2]$는 분산을 뜻한다. 즉, 이는 데이터셋으로부터 일끌어 낸 가설과 특정 데이터셋 D로부터 이끌어 낸 가설 사이의 차이를 뜻한다. $(\bar{g}(x) - f(x))$는 편향을 뜻하는데, 이는 데이터셋으로부터 이끌어낸 가설과 목적 함수의 차이를 뜻한다. 즉, 이는 모델을 근사하는 과정에서 학습의 한계 때문에 발생한 오차라고 생각하면 된다.

그러면 어떻게 편향과 분산을 줄일 수 있을까? 편향의 경우 모델을 더 복잡하게 만들면 되고, 분산은 더 많은 데이터를 수집하면 해결된다.

6.2 Performance Measurement

그렇다면 편향이 큰 모델과, 분산이 큰 모델 중 어떤 모델을 선택해야 할까? 특정 데이터셋을 통해 학습 시킨 결과, 전체 오차 E_{out}는 같지만, 편향과 분산이 다른 두 모델이 생성되었다고 가정하자. 이럴 경우 오컴의 면도날 (Occam's Razor)을 사용하면 된다.

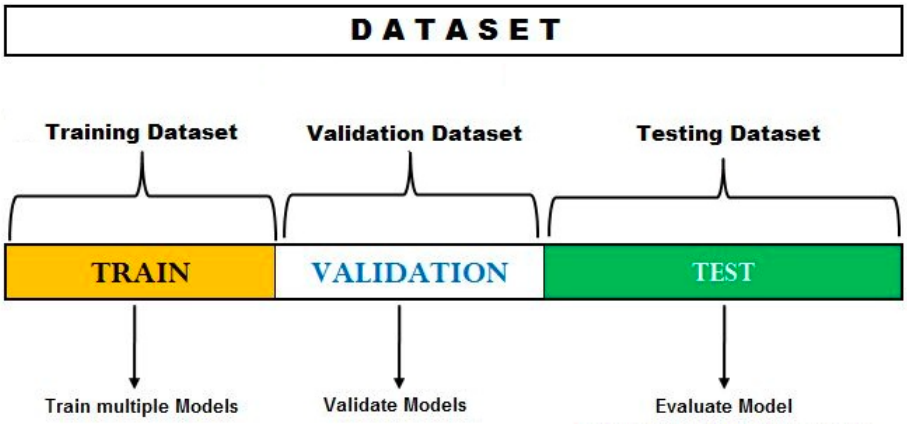
오컴의 면도날이란 '많은 것들을 필요없이 가정해서는 안된다','더 적은 수의 논리고 설명이 가능한 경우, 많은 수의 논리를 세우지 말라' 등, 같은 현상을 설명하는 주장이 있다면, 그 중 간단한 것을 선택하라는 것을 뜻한다. 즉, 우리가 항상 말하는 simple the better 인 것이다..! 그래서 우리는 주어진 모델중에 간단한 모델, 즉, 편향의 비율이 더 높은 모델을 선택하는 것이 맞는 것이다.

6.2.1 Validation

머신러닝 모델 검증 방법은 다양하지만 여기서는 3개만 다루겠다.

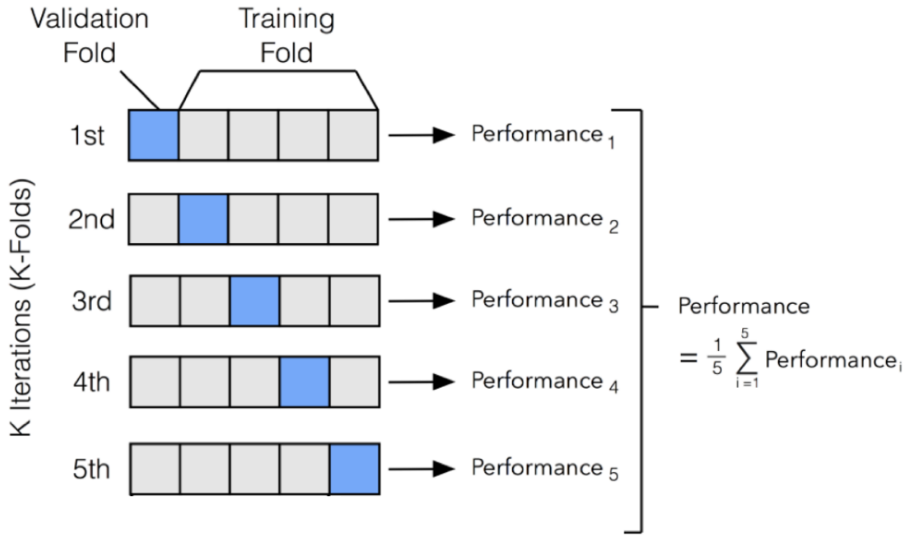
(1) Holdout

Holdout 검정은 가장 기본적인 모델 검증 방법이다. 전체 데이터셋에서 테스트 데이터를 분리하고 남은 학습 데이터의 일부를 검증 데이터셋으로 또 분리하는 방법이다. 즉, 전체 데이터를 학습 데이터, 검증 데이터, 데스트 데이터로 나누는 것이다. 이 방법은 간다한 만큼, 학습 데이터에 손실이 있고, 검증을 한 번만 할 수 있다는 단점이 있다. 또한, 학습, 검증 데이터에 이상치가 다 몰려있는 경우, 잘못된 모델을 만들수 있다.



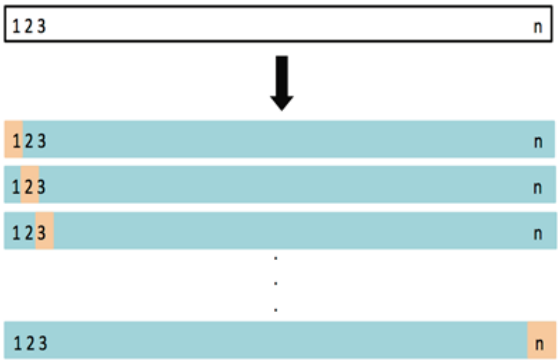
(2) K-fold CV

홀드아웃 방법을 개선한 방법이 교차검증(Cross validation, CV)이다. 교차 검증은 최소 2번 이상의 검증을 진행하므로 각 검증 결과 값의 평균을 모델의 검증 결과로 사용하게 된다. 교차 검증 방법 중 가장 유명하고 많이 쓰이는 방법은 K-fold 교차검증 (K-Fold CV) 이다. K-Fold cv는 학습 데이터를 k개로 나눈 뒤 순차적으로 1개는 검증 데이터셋으로, 나머지 k-1개는 학습 데이터로 사용해, k번 검증을 진행하는 방법이다. 이 방법도 학습 과정마다 학습 데이터에 손실이 있기는 하지만, 결국 전체 데이터를 살펴볼 수 있고, 검증 횟수를 늘릴 수 있다는 장점이 있다.

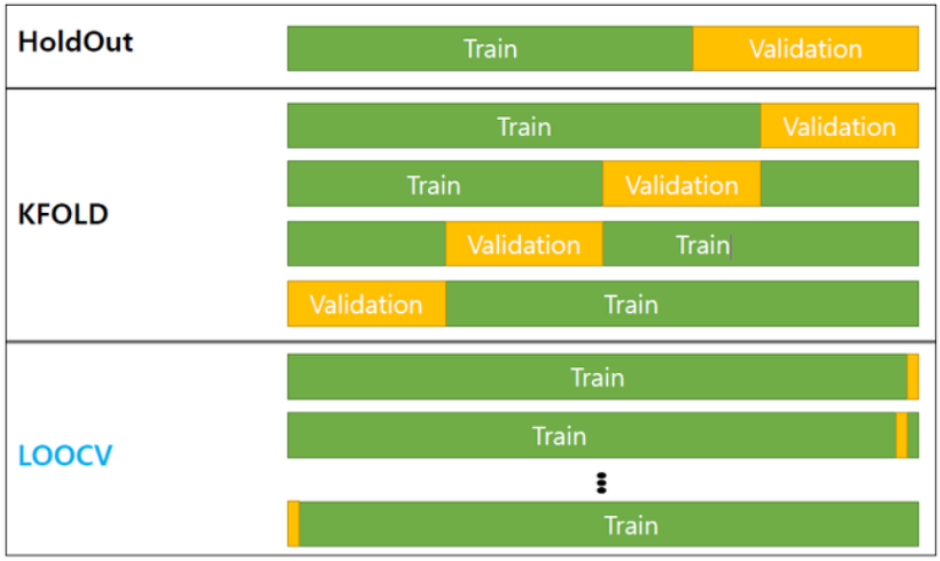


(3) LOOCV (Leave-One-Out Cross Validation)

LOOCV는 K-Fold cv의 극단적인 형태로, 학습 데이터가 작은 경우 사용하는 방법이다. LOOCV는 하나의 인스턴스만을 검증 데이터셋으로 남겨 놓는다. 즉, 검증 데이터가 하나이므로, 학습 데이터가 N개라면, 총 N번의 교차 검증을 진행하는 것이다. 이는 학습 데이터의 손실을 최소화할 수 있지만, 샘플의 데이터 수가 늘어나면 오랜 시간과 컴퓨팅 파워를 요구한다.



그래서 holdout, k-fold cv와 LOOCV를 비교하자면 다음과 같다.



6.2.2 Hyperparameter Optimization

검증을 통해 과적합을 방지하는 것도 중요하지만, 모델의 성능을 높이기 위해 파라미터들을 튜닝하는 것도 중요하다. 하이퍼파라미터(Hyperparamter)는 학습 과정에서 사람이 지정하는 파라미터를 의미한다. 대표적인 예시로는 경사 하강법의 학습률, 정칙화의 정도를 조정하는 α 등이 있습니다. 모델마다 설정해주어야 하는 하이퍼파라미터가 다르기 때문에, 튜닝을 진행하기 전에, 어떤 하이퍼파라미터를 튜닝해야 한 번 하는지 알아보는 것도 나쁘지 않다.

하이퍼파리미터의의 조합에 따라 모델의 결과도 달라지기 때문에, 검증 결과를 비교하여 최적의 하이퍼파라미터의 조합을 찾아야 한다. 최적의 하이터파라미터의 조합을 찾는 방법은 다양하기 때문에, 몇 개만 소개하겠다.

(1) Manual Search

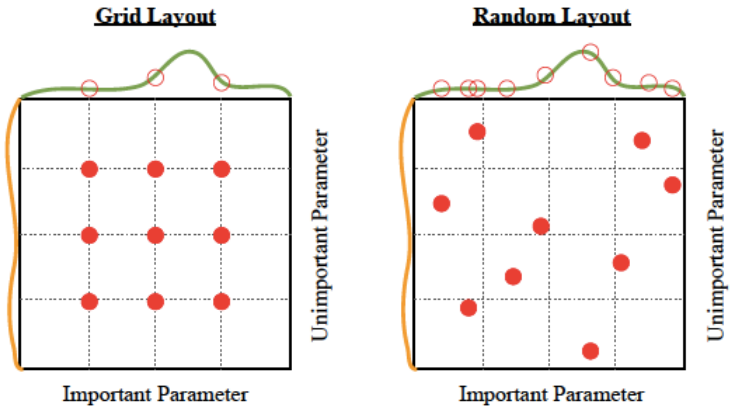
이름 그대로, 수동으 하이퍼파라미터를 조정하는 방법이다.

(2) Grid Search

하이퍼파라미터마다 적용하고 싶은 목록을 설정해둔 뒤 그 조합을 모두 시행하보고 최적의 조합을 찾아내는 방법이다. 시도해보고 싶은 조합이 많아지면, 시간이 오래 걸린다는 단점이 있다.

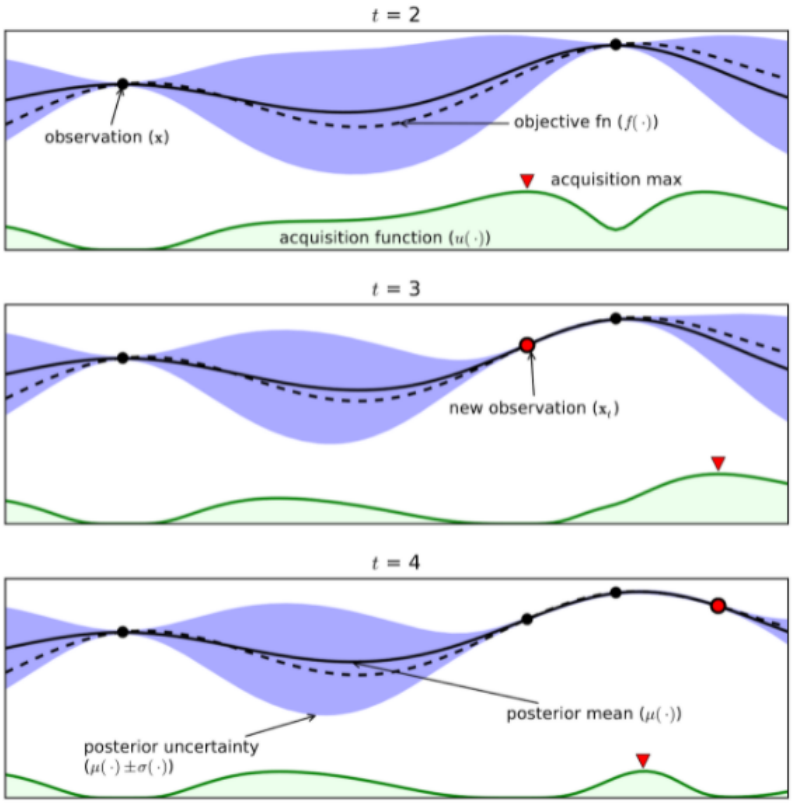
(3) Random Search

조정하고 싶은 하이퍼파라미터 마다 조정하고 싶은 범위를 정한다. 실제 학습 과정에서는 범위 내에 있는 임의의 값을 설정하여 각 파라미터에 대한 조합을 정한다. Grid search는 지정한 값에 대해서만 학습과 검증을 시도하지만 Random search는 범위 내에 있는 임의의 값에 대하여 예상치 못한 조합을 시도해볼 수 있다는 장점이 있다. 비교해보자면 다음과 같다.



(4) Bayesian Optimization

베이지안 최적화란 사전 정보를 최적 값 탐색에 반영하는 것이다. 베이지안 최적화에서는 surrogate model과 acquisition function을 통해 결과를 내고 학습하는 것처럼 돌아간다. Surrogate model이란 현재까지 조사된 입력-함수값 쌍들에 대해 목적 함수에 대한 확률적 추정을 수행하는 모델이다. Acquisiton function은 surrogate model을 활용해 다음 입력값 후보를 추전해주는 함수로 다음 그림과 같이 작동한다.



이를 이해하기 위해서는 gaussian proccess를 알아야 한다. Gaussian process 내용은 1주 뒤에 알아볼 예정이니 그때까지 참아보자.

베이지안 최적화로 하이퍼파라미터 수정하기 (Bayesian Optimization)

이번 포스트에서는 Auto ML 방법 중 하이퍼 파라미터 튜닝에 대해 다루며, Bayesian Optimization(베이지안 최적화) 방법을 소개하겠습니다. 베이지안 최적화는 Auto ML 분야에서도 Hyperparameter Optimization(하이퍼파라미터 튜닝)에 대한 내용입니다. 우선 Auto ML 분야에 대해 알아보겠습니다. Auto ML은 '머신러닝으로 설계

https://jihyun22.github.io/2020/10/07/Bayesian-Optimization/

6.2.3 Evaluation

이제부터 모델을 평가할 수 있는 지표에 대해서 알아보자.

(1) Confusion Matrix (혼동 행렬)

혼동행렬이란, 분류 모델의 성능을 평가하는 지표로 모델에서 훈련을 통해 예측한 값($\hat{Y}$)과 실제값(Y)을 비교하기 위한 행렬이다 . 혼동행렬의 형태는 아래의 표와 같으며 우리는 반응변수가 이항변수인 경우를 중점적으로 살펴볼 것이다.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	TP	FP
	$\hat{Y}=0$	FN	TN

혼동행렬의 칸에 있는 두 문자는 각 칸이 어떠한 의미를 지니는지를 보여준다. 앞에 위치한 T (True) / F(False)는 실제 값과 예측 값의 일치 여부를 나타내고 , 뒤에 위치한 P(Positive)/ N(Negative)는 연구자가 긍정 혹은 부정으로 예측하였는지를 나타낸다.

- TP (True Positive) : 긍정으로 예측($\hat{Y}=1$)하였고 실제 값고 긍정 ($Y=1$)인 경우
- TN (True Negative): 부정으로 예측 ($\hat{Y}=0$)하였고, 실제 값도 부정 ($Y=0$)인 경우
- FP (False Positive): 긍정으로 예측하였지만 실제 값은 부정인 경우 $\rightarrow$ 1종 오류
- FN (False Negative): 부정으로 예측하였지만, 실제 값은 긍정인 경우 $\rightarrow$ 2종 오류

혼동행렬의 한계점

1) 정보의 손실 발생

로지스틱 회귀는 예측 값 $\hat{\pi}$으로 연속적인 확률을 반환하지만, 분류를 진행하면서 해당 예측 값을 cut off point 에 따라 이항 변수로 범주화한다. 이 과정에서 크기 처럼 본래 예측 값이 지니고 있던 정보가 손실되어진다.
예를 들어 $\hat{\pi}$= 0.72 라는 예측 확률 값이 주어졌을 때 cut off point 가 0.6 으로 정해져 있다면 해당 예측값은 1 이 되고, 0.72 라는 수치적인 의미는 사라지게 되는 것이다.

2) 임의적인 cut-off point 설정

보통 cut off point 를 0.5로 지정하여 분류를 진행하지만, cut off point 를 임의적으로 지정하는 것은 분석의 객관성을 떨어트린다. 더불어 cut off point에 따라 혼동행렬 값이 달라지기 때문에 분석결과가 크게 바뀔 수도 있다.

(2) 분류 평가지표

혼동행렬의 한계점을 보완하여 모델의 성능을 평가할 수 있지만, 혼동행렬을 이용하여 분류모델의 성능을 평가할 수 있는 다양한 지표들을 형성할 수 있다. 이때, 주어진 상황에 따라 성능을 오바르게 판단할 수 씨는 적절한 평가 지표를 사용하는 것이 중요하다.

1) 정확도

정확도(Accuracy/ACC/정분류율)는 전체 경우에서 실제값과 예측값이 일치하는 비율이다.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	TP	FP
	$\hat{Y}=0$	FN	TN

정확도는 직관적인 지표로 성능을 평가하는데 자주 사용되는 지표이다. 전체 중에 정확하게 예측한 비율을 나타냄으로 1에 가까울수록 좋은 모형이라고 판단할 수 있다. 하지만 unbalanced data가 주어졌을 때 해당 지표를 사용하게 되면 문제가 발생한다.

2) 정밀도

정밀도 (Precision/PPV/Positive Predictive Value) 란 긍정으로 예측($\hat{Y}$=1)한 것들 중 실제값도 긍정(Y=1)인 것의 비율이다. 즉, 긍정으로 예측한 값 중 올바르게 예측한 값의 비율을 의미한다. 정밀도도 정확도처럼 정밀도가 1에 가까울수록 좋은 모형이라고 평가되어진다.

$$Percision = \frac{TP}{TP + FP}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	TP	FP
	$\hat{Y} = 0$	FN	TN

정밀도는 False Positive, 즉 실제로는 아닌데 맞다고 오판하는 경우가 치명적인 경우에 사용한다. 예를 들어, 식수가 오염되었는데 위생적이라고 판단한다는 것이 식수가 위생적인데도 오염되었다고 판단하는 것보다 더 위험하다. 오염된 물을 마시면 식중독에 걸릴 수 있으므로 이처럼 FP 가 critical 한 경우에 정밀도를 사용하여 모형을 평가한다.

3) 민감도 (재현율)

민감도 (Sensitivity/TPR/True Positive Rate) 또는 재현율 (Recall) 로 지칭되는 평가지표는 실제 긍정(Y=1) 인 것들 중 예측값을 긍정($\hat{Y}$=1)으로 판단 한 비율이다. 즉, 실제 긍정인 것들 중 긍정으로 예측한 것에 관한 비율이므로, 민감도도 1에 가까울수록 성능이 좋다고 판단한다. 민감도는 이후에 배울 ROC 곡선의 Y축에 해당 한다.

$$Sensitivity = \frac{TP}{TP + FN}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	TP	FP
	$\hat{Y} = 0$	FN	TN

민감도가 중요한 지표로써 적용되는 경우는 False Negative 가 치명적인 경우이다. 의사가 오진을 하는 경우를 예시로 들어보자. 예를 들어 , 몸 속에 암세포가 없는데 있다고 진단하는 경우보다 몸 속에 암세포가 존재하는데 없다고 판단하는 경우가 더 치명적이다. 왜냐하면 몸 속에 암세포가 없는데 오진을 한 경우는 추가적인 시간과 비용을 지불하면 되지만, 암세포가 있는데 오진을 한다면 그 사람의 생명이 매우 위험해지기 때문이다. 이와같이 FN이 critical 한 경우에 정밀도를 사용하여 모형을 평가한다.

4)특이도

특이도 (Specificity/TNR/True Negative Rate) 은 실제로 부정 (Y=0) 인 것 중 부정 ($\hat{Y}$=0)으로 올바르게 예측한 비율을 나타낸다. 즉, 부정인 것을 잘 맞춘 정도를 나타냄으로 특이도도 1에 가까울수록 모델의 성능이 좋다고 판단하다.

$$Specificity = \frac{TN}{TN + FP}$$

		관측값(Y)	
		$Y = 1$	$Y = 0$
예측값($\hat{Y}$)	$\hat{Y} = 1$	TP	FP
	$\hat{Y} = 0$	FN	TN

특이도의 반대의 경우, 즉 실제로 부정인 것 중 긍정으로 잘못 예측한 비율을 FPR(False Positive Rate)이라고 부르며 식은 다음과 같다.

$$FPR = \frac{FP}{TN + FP} = 1 - Specificity$$

즉 FPR은 1- 특이도와 같다 그러므로 특이도는 1에 가까울수록 FPR은 0에 가까울수록 모델의 성능이 좋다고 판단한다. 또한, FPR은 ROC 곡선의 X 축에 해당 한다.

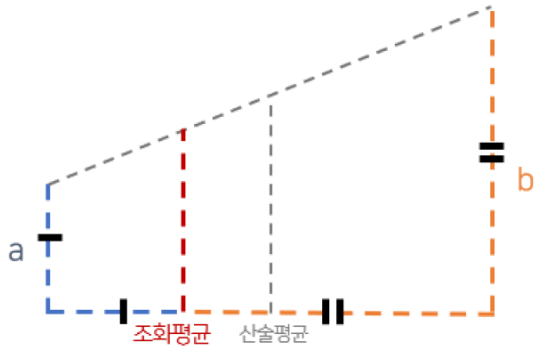
5) F1-Score

F1 Score 는 정밀도 (Precision) 와 재현도 (Recall= 민감도, Sensitivity)의 조화평균이다. 조화평균은 역수의 산술평균의 역수를 의미한다. 즉, a와 b 두 상수가 주어졌다면 이에 대한 조화평균은

$\frac{2}{\frac{1}{a}+\frac{1}{b}}$ 인 것이다. 조화평균을 활용한 F1 Score 을 구하는 식은 다음과 같다.

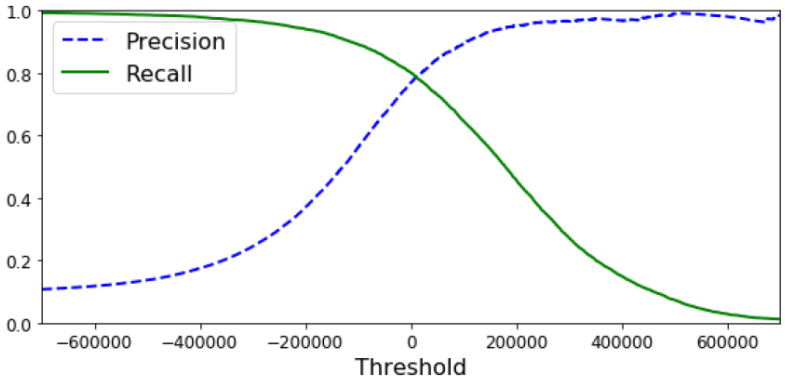
$$F1 - Score = \frac{2}{\frac{1}{Percision} + \frac{1}{Recall}} = 2 \times \frac{Percision \times Recall}{Percision + Recall} = \frac{2TP}{2TP + FN + FP}$$

앞서서 불균형 데이터가 주어졌을 때, 정확도 (Accuracy) 를 평가지표로써 사용한다면 성능을 잘못 평가할 수있다고 언급하였다. 하지만 F1 Score은 조화평균을 사용하여 성능을 평가하기에 데이터가 불균형한 경우에도 성능을 정확히 평가 할 수 있다.



조화평균을 기하학적으로 접근하면 위의 그림과 같이 나타낼 수 있다. 위의 그림은 밑변이 a와 b 로 이루어진 사다리꼴임을 알 수 있다. 이때, 각 변의 길이와 동일한 거리에 떨어진 지점에서 사다리꼴의 빗변으로의 높이가 바로 a 와 b 에 대한 조화평 균이 되는 것이다.

즉, 단순히 산술평균을 구하는 것이 아니라 큰 값을 가지는 수치에 패널티를 주어 작은 값에 가까운 평균을 도출하기 위해 조화평균이 사용된 것이다. 이러한 조화평균을 이용한 F1 Score를 불균형 데이터에 적용하면, 관측값이 많은 클래스에 패널티가 주어짐에 따라, 보다 더 정확하게 성능을 평가할 수 있는 것이다.
조화평균을 사용하는 또 다른 이유는 F1 Score를 구성하는 정밀도와 재현도를 모두 균형 있게 반영하기 위해서다. 앞서서 정밀도와 민감도는 모두 1에 가까운 값을 가질 때 성능이 좋다고 판단할 수 있다고 했는데, 정밀도와 민감도는 서로 상충관계 (Trade off) 이기 때문에 동시에 큰 값을 지닐 수 없다. 즉, 정밀도가 커지면 민감도는 낮아지고, 반대로 민가도가 커지면 정밀도가 낮아지는 것이다.



즉 임계값 (Threshold, cutoff point)이 낮아지면 바뀌기 전보다 positive로 예측하는 값(임계값보다 클 때)이 많아질 것이므로, recall 값은 오르지만 반대로 precision의 수치는 감소할 것이다. 따라서 어느 한 평가지표만 1에 가깝다고 좋은 성능은 지닌다고 판단할 수 없기에, precision과 recall을 모두 활용한 F1 score 를 사용하여 적합한 모형을 선정하고자 하는 것이다.

F1 Score 또한 1에 가까울수록 해당 모형의 성능이 우수하다고 판단한다. 하지만, F1 Score는 FN(False Negative)을 고려하지 않는다. 즉 , 많아진 관측 값에 대해 Negative로 올바르게 예측한 경우가 많아지더라도 F1 Score에는 전혀 반영되지 않는 것이다. 이러한 한계를 보완할 수 있는 지표로써 마지막으로 MCC에 대해 알아보자.

6) MCC

MCC(Matthews Correlation Coefficient) 매튜 상관계수 파이계수는 다른 지표들과 다르게 혼동 행렬에 있는 모든 부분으로 식이 구성 되어져 있다. 이로 인해 평가 지표 중 가장 균형 잡힌 척도로써 여겨진다.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

지표의 이름처럼 해당 지표는 상관계수를 나타내며, 다른 상관계수들과 동일하게 -1~1 사이의 값 을 가진다. 즉, 1에 가까울 수록 완전예측, 0 이면 랜덤예측, 그리고 -1 에 가까울수록 역예측으로 판단하는 것이다. 해당 지표 또한 불균형 데이터에서 유용하게 사용할 수 있다.
Imbalance data의 경우 F1 Score 와 MCC 를 비교해보며 두 평가지표가 어떠한 특징을 지니고 있는지 다시 한번 살펴보자.

		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	92	4
	$\hat{Y}=0$	3	1

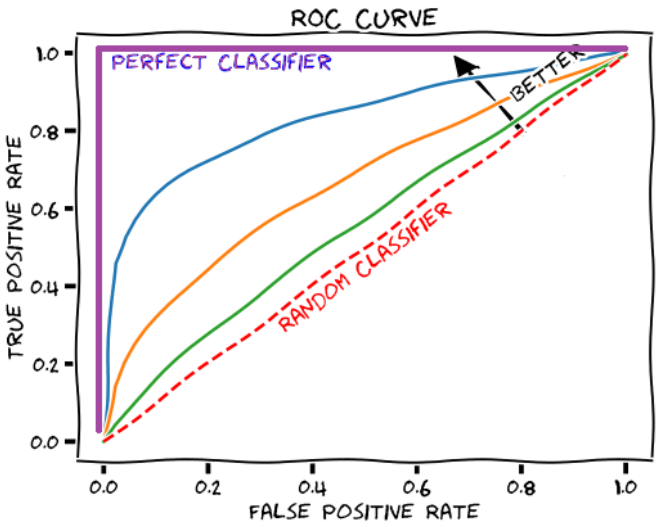
		관측값(Y)	
		$Y=1$	$Y=0$
예측값($\hat{Y}$)	$\hat{Y}=1$	1	3
	$\hat{Y}=0$	4	92

혼동행렬	F-1 Score	MCC
왼쪽	$\frac{2\times 92}{2\times 92+4+3}=0.96$	$\frac{(92\times 1)-(4\times 3)}{\sqrt{(92+4)(92+3)(1+4)(1+3)}}=0.18$
오른쪽	$\frac{2\times 1}{2\times 1+3+4}=0.22$	$\frac{(1\times 92)-(3\times 4)}{\sqrt{(1+3)(1+4)(92+3)(92+4)}}=0.18$

위의 두 평가지표를 비교한 표를 살펴보면 두 혼동행렬 모두 MCC 에 대해 동일한 값을 지니지만 F1 Score에 대해선 약 0.74 정도 큰 차이를 보인다. 이는 MCC 는 혼동행렬의 모든 성분을 사용하지만 F1 Score 는 평가지표 값을 구하기 위해서 TN을 활용하지 않기 때문에, 위 예시처럼 TN 값의 차이가 크면 F 1 Score 또한 큰 차이를 보이는 것이다. 그러므로 모델의 성능을 평가할 때 F1 Score 하나만 가지고 판단하면 문제가 발생 할 수 있다.
그렇다고 F1 Score가 MCC 보다 좋지 않은 평가지표라는 의미는 아니다. 예를 들어 분석의 목적이 클래스가 positive인지 아닌지의 여부에 상관없이 클래스에 대한 균형적인 평가라면, MCC 를 사용해야 한다 하지만, F1 Score는 TP에 더 중점을 두어, 성능을 평가하기에, 희귀 질환처럼 흥미롭지만 관측치가 적은 경우, 해당 클래스를 Positive로 두어 F1 Score 를 사용하여 성능을 평가하는 것이 더 효과적이다.
앞서서 6가지의 평가지표에 대해 살펴보았지만, 모델의 성능을 평가 할수 있는 지표들은 이보다 훨씬 다양하다. 위의 내용들을 통해 평가지표가 어떻게 구성 되었는지 깨닫는 것도 좋지만, 분석 목적과 주어진 데이터에 따라 적절한 평가 지표를 사용하는 것이 가 장 중요하다. 즉, 최고의 평가 지표는 없는 것이다.

(3) ROC curve

분류 모델 성능 평가를 위해 AUC(Area Under Curve) 또한 많이 사용된다. AUC는 말 그래도 ROC 곡선의 아래의 면적을 뜻한다.
ROC 곡선 (Receiver Operative Characteristic Curve)은 2차대전 때 통신 장비의 성능을 평가하기 위해 나온 수치이다. 앞서 말했듯, ROC 곡선은 FPR을 X 축으로 TPR을 Y축으로 가진 그래프이다.



6.3 Model Regularization

정규화(Regularization)는 오버피팅을 방지하기 위한 하나의 방법이다. 정규화를 적용한 모델은 데이터에 민감하게 반응하지 않기 때문에 분산에 의한 오차를 줄일 수 있다. 이는 원래 수식에 비용 함수에 regualrization termdmf 추가하여 모델 조정을 진행한다.

선형회귀 모델을 규제하는 방법부터 알아보자. 선형회귀 모델의 regularizaiton은 크게 2가지 방법이 이다. 첫 번째는 L1 Regularization 이라고 불리는 라쏘(Lasso)다. 두 번째는 L2 norm을 사용하는 릿지(Ridge)이다. 그리고 가끔 두 방법을 혼합한 엘라스틱 넷(Elastic Net)을 사용한다.

각 방법을 사용할 때의 비용 함수 $E(w)$는 다음과 같다.

$$\text{without Regulation} : E(w) = \frac{1}{2}\sum_{n=0}^{N}(Train_n - g(x_n, w))^2$$

$$\text{Ridge} : E(w) = \frac{1}{2}\sum_{n=0}^{N}(Train_n - g(x_n, w))^2 + \frac{\lambda}{2}||w||^2$$

$$\text{Lasso} : E(w) = \frac{1}{2}\sum_{n=0}^{N}(Train_n - g(x_n, w))^2 + \lambda||w||$$

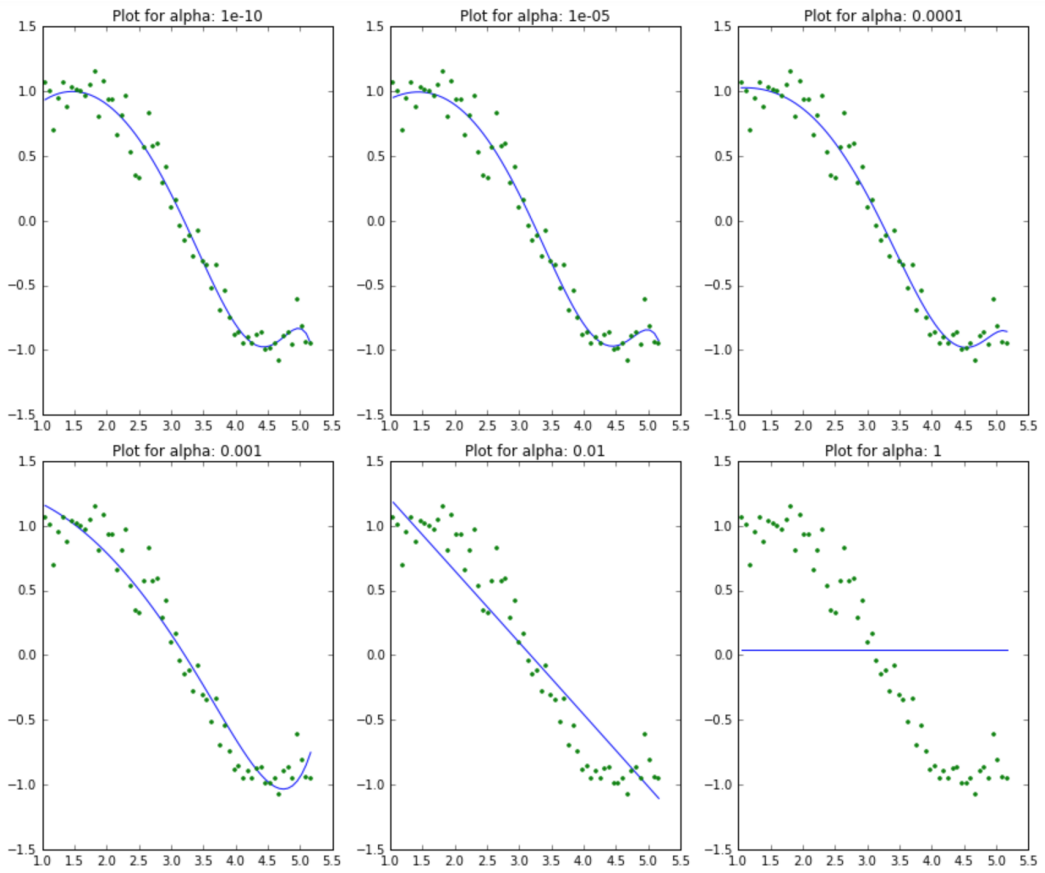
릿지에 의해서 구해지는 파라미터 w가 어떻게 달라지는 지를 수직적으로 알아보자. 이는 앞에서 했던 것과 동일하게 비용함수를 최소화하는 w를 구하는 것이 목적이므로 미분하여 0이 되는 점을 최소제곱법으로 구하면 된다.

$$\begin{aligned}
\frac{d}{dw}E(w) &= \frac{d}{dw}(\frac{1}{2}||train - Xw||^2 + \frac{\lambda}{2}||w||^2 \\
&= \frac{d}{dw}(\frac{1}{2}||train - Xw||^T||train - Xw|| + \frac{\lambda}{2}w^Tw) \\
&= \frac{d}{dw}(\frac{1}{2}(train^Ttrain - 2X^Tw \cdot train + X^TXw^Tw) + \frac{\lambda}{2}w^Tw) \\
&= \frac{d}{dw}(\frac{1}{2}train^Ttrain - X^Tw \cdot train + \frac{1}{2}X^TXw^Tw + \frac{\lambda}{2}w^Tw) \\
&= -X^T \cdot train + X^TXx + \lambda w = 0
\end{aligned}$$

$$\therefore w = (X^T X + \lambda I)^{-1} X^T \cdot Train$$

λ는 regularlization에서 등장하는 하이퍼파라미터로 regularization의 정도를 결정한다. λ가 커지면 $X^T X + \alpha I \approx \alpha I$가 된다. 결국 파라미터 행렬은 $w = \frac{1}{\lambda} X^T \cdot Train$이 되어 모델이 단순해진다. 반대로 λ가 0에 가까우면 regularlization을 적용하지 않았을 때와 식이 같아지기 때문에 regularlization의 영향력이 사라진다. 즉, 너무 작은 λ인 경우, 높은 분산을 가진다.

릿지와 라쏘 모델을 적용했을 때의 차이는 w중 regularlization의 대상이 되는 변수들, 즉 영향이 없는 특이값에 부여되는 변수가 어떤 방식으로 변하느냐에 있다. 릿지는 이런 변수들의 값을 0에 가깝게 바꾼다. 하지만, 라쏘는 이런 변수들을 아예 0으로 만들어 특성을 사라지게 한다. 그래서, 라쏘의 경우 변수 선택이 가능한 것이다. 이를 그래프로 알아보자!



선형 회귀 뿐만 아니라 다른 모델에서도 regularization을 적용할 수 있다. 로지스틱 회귀에 regularization을 적용하면 다음과 같아진다. 아래의 식은 비용 함수를 규제항에 더해주는 것이 아니라 우도에 정칙화 항을 뺴주는 방식으로 regularizaiton을 적용하고 있다. 이때도 동일하게 λ값을 변화시켜 regularization의 정도를 조정하며, λ가 커질수록 강한 regularization을 수행한다.

$$\arg\max_{\theta} \sum_{i=1}^{m} \log p(y_i | x_i, \theta) - \lambda R(\theta)$$

$$L1 : R(\theta) = ||\theta||_1 = \sum_{i=1}^{m} |\theta_i|$$

$$L2 : R(\theta) = ||\theta||_2^2 = \sum_{i=1}^{m} \theta_i^2$$

SVM에도 규제를 적용할 수 있다. 규제를 적용한 비용함수 f를 수식으로 아래와 같이 나타낼 수 있다. 이때도 마찬가지로 C 값을 변화시켜 규제의 정도를 조절하면 된다. 선형회귀, 로지스틱 회귀와 달리 하이퍼파라미터 C가 커질수록 규제의 정도는 약해진다.

$$f = \arg\min_{f \in H} \{ \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(x_i)) + \lambda ||f||_H^2 \}$$

$$V(y_i, f(x_i) = (1 - y(f(x)) + (s)_+ = \max(s, 0)$$

$$f = \arg\min_{f \in H} \{ \frac{1}{n} \sum_{i=1}^{n} (1 - yf(x)) + \lambda ||f||_H^2$$

$$f = \arg\min_{f \in H} \{ C \sum_{i=1}^{n} (1 - yf(x)_+ + \frac{1}{2} ||f||_H^2$$

$$C = \frac{1}{2\lambda n}$$

아래는 C가 작은 값(강한 regularization)일 때와 큰 값(약한 regularization)일 때의 결정 경계가 어떻게 변화하는 지를 보여준다. 강한 regularization이 적용되었을 때에는 아웃라이어 값을 무시하는 것을 볼 수 있다. 맨 아래 그림은 추가적인 데이터가 들어왔을 때 정칙화의 효과를 보여준다. 약한 정칙화가 적용된 모델은 X표시된 몇 개의 인스턴스를 잘못 분류하는 것을 볼 수 있다.