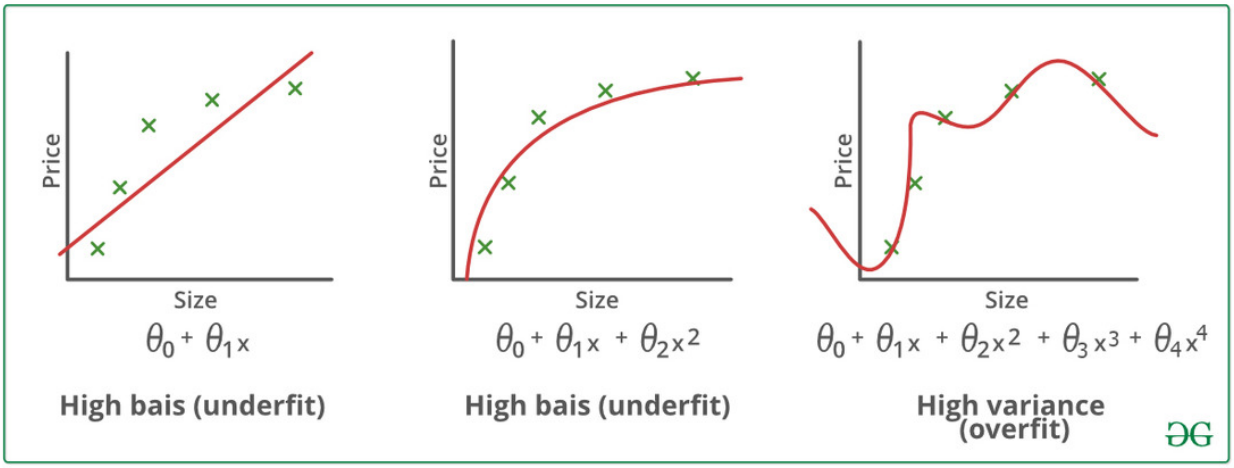
8강: Data Splits, Models & Cross-Validation

Part 6 Learning Theory

1. Bias/ Variance tradeoff

linear regression에 대해 배울 때, “simple” 모델인 $y = \theta_0 + \theta_1 x$에서부터 “complex”모델인 $y = \theta_0 + \theta_1 x + \dots \theta_5 x^5$를 살펴보았고, 이를 다음과 같이 그래프로 표현할 수 있다.



먼저 오른쪽 모델을 보면 , training set에서는 input feature x에 대해 output y를 잘 예측하는 것처럼 보이지만, 새로운 data에 대해 ouput y를 잘 예측할 것이라고 기대하지 않는다.

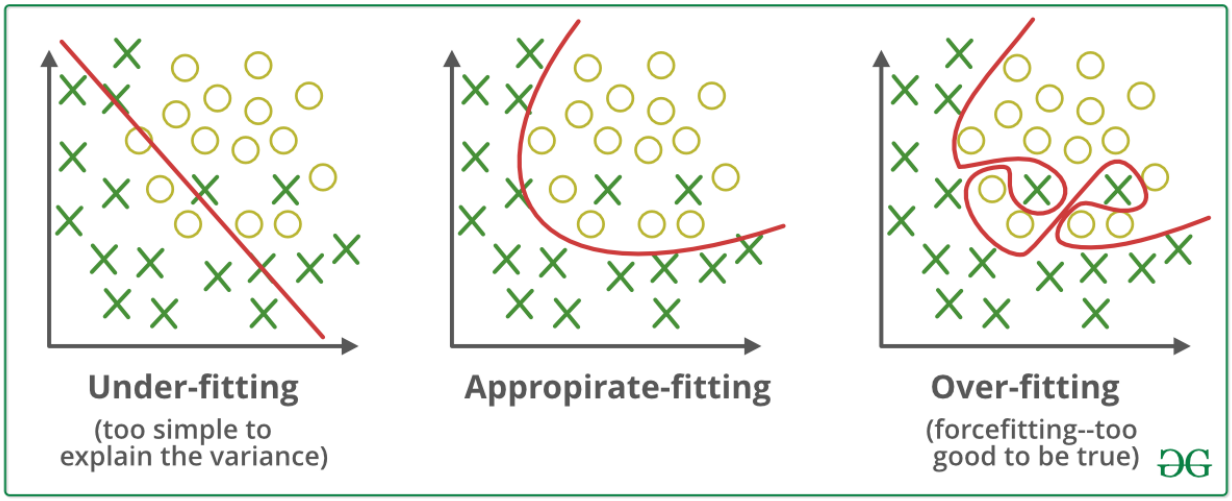
다른말로, training set에 의해 만들어진 모델이 다른 data에 대해 generalize하지 않는다는 것이고, 여기에서 **generalization error of hypothesis**를 정의할 수 있다. (training set에 의해 발생하지 않은 expected error를 의미)

따라서 맨 왼쪽, 맨 오른쪽 모델 둘다 큰 generalization error를 가지지만 처한 문제 상황은 다르다.

- 맨 왼쪽 : x, y 가 linear한 관계가 아닌데 linear model로 fit한 경우, 데이터 구조를 정확하게 나타내는데 실패했다. 이러한 경우 expected generalization error를 모델의 **bias**라고 정의할 수 있다. 따라서 위의 linear model은 large bias & underfitting 문제가 발생한 것이다.

- 맨 오른쪽: 4차의 다항 모델은 x,y 사이의 wider pattern을 반영하지 못하면서, 새로운 데이터가 들어올 경우 모델의 변동이 크다. 이러한 경우 expected generalization error를 모델의 variance로 정의할 수 있다. 따라서 위의 model은 large variance& overfitting 문제가 발생한 것이다.
- bias variance tradeoff :
  - model is too “simple” : bias크고 variance 작음
  - model if too “complex” : bias작고 variance크다.

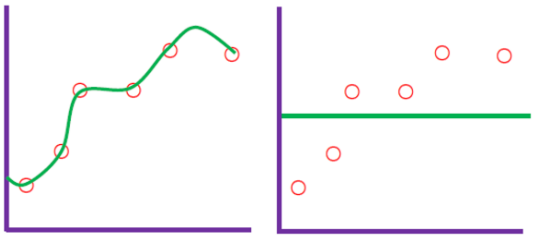
참고/ 이는 다음과 같이 binary classification 문제에서도 생각해볼 수 있다.



이렇게 너무 많은 feature가 존재하면 과적합 문제가 발생하는데 이를 예방할 수 있는 효과적인 방법 중 하나인 regularization를 알아보자

2. Regularization

정규화(regularization)이란 회귀계수가 가질 수 있는 값에 제약조건을 부여하는 방법이다. 정규화를 통해 variance를 줄이면서 overfitting을 예방하고 성능을 높인다.



이처럼 오른쪽을 정규화한 결과 왼쪽의 결과가 도출된다. 참고로 오른쪽에는 가장 강한 정규화를 수행한 결과인다.

> 학습데이터에 대한 설명력을 다소 포기하는 대신에 미래 데이터 변화에 상대적으로 안정적인 결과를 낸다.

(1) linear regression

먼저 linear regression의 optimization objective는 다음과 같은데, 우리는 data를 fit하기 위해 모델의 예측값과 실제 y값의 평균제곱오차(Mean Square error)를 작게 만드는 parameter θ를 구해야 한다는 것을 배웠다.

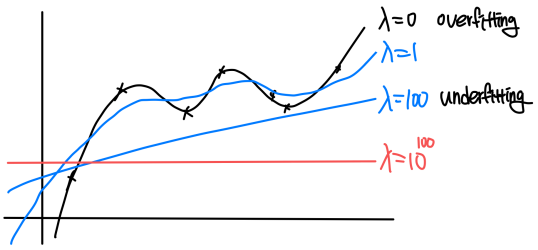
여기에 regularization이 적용된 optimization objective를 다시 살펴보자.

- $\lambda \parallel \theta \parallel^2$: regularization term

  λ : Regularization Paramter
  각 항의 계수인 파라미터가 너무 커지는 것을 방지하기 위해 패널티를 부여하는 부분이다.

이렇게 정규화 term을 추가하면서 파라미터가 커지는 것을 막으면서 데이터에 과적합하는 것을 방지할 수 있다.

💥 강의에서는 ridge regularization(L2)만을 설명하네여



응 교수님이 그리신 그래프를 보면

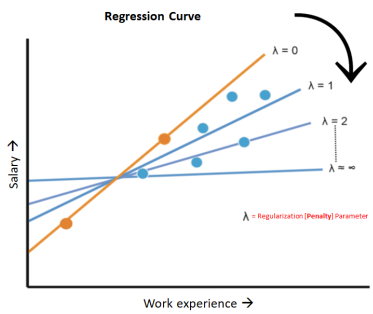
λ값이 증가할 때

$\lambda = 0$일때는 과적합 문제 발생

$\lambda = 1$ 일때는 적당한 모델

$\lambda = 100$ 일때 과소적합 문제 발생

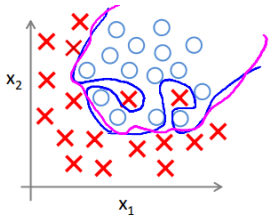
$\lambda = 10^{100}$ 처럼 엄청 큰 숫자일때는 straight line으로 fit



(2) Regularized logistic regression

이번에는 logistic regression의 cost function에 정규화를 추가하면 다음과 같다.

$$argmax_\theta \sum_{i=1}^{n} log\ p(y^{(i)}|x^{(i)};\theta) - \lambda \parallel \theta \parallel^2$$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

위의 그래프를 보면 λ작으면 파란색 곡선처럼 과적합된 decision boundary를 얻고 λ 값이 커질 수록 분홍색곡선의 적당한 decision boundary를 얻을 가능성이 높아진다.

참고로 정규화는 svm에 포함되어 있기 때문에 이로 인해 고차원에서 과적합이 일어나지 않는 것이다.

또한 우리는 이전에 text classification에서 naive bayes를 배웠는데 이러한 분류 문제에서 feature는 많고(10000) 이에 비해 example data(100)가 적을 때는 logistic regression에 regularization term을 추가한 모델이 성능이 더 좋다. 보통 feature가 example data보다 10배 이상 클 경우 regularization을 추가한다.

(3) gradient descent

로지스틱 회귀에서 경사하강법은 다음과 같다.

repeat {

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$

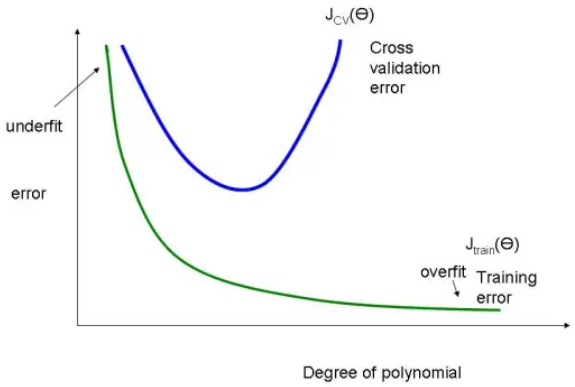
$\theta_j := \theta_j - \alpha [\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j]$ $(j = 1, 2, 3, ...n)$ ← 동시에 업데이트

}

θ_0과 θ_j를 분리하여 업데이트 하는데, θ_j열에만 정규화 term을 추가한다.

(4) regularization& choosing the degree of polynomial

다음의 그래프는 regularization을 하지 않고 모델의 복잡도에 따른 train error의 변화를 나타낸 그래프이다. 이 error는 generalization error와 같다고 볼 수 있다.



모델의 복잡도가 낮을 수록 underfitting이 일어나 train error가 크다.

그러나 모델 목잡도가 높아질수록 에러가 줄어들다가 중간에 error가 상승하면서 overfitting 문제가 발생한다.

곡선이 꺾이기 시작하는 직전이 가장 좋은 모델 복잡도를 가진 모델이라고 볼 수 있다.

이때 regularization을 추가하면 λ값을 과하게 설정하면 underfitting 되고 λ를 0으로 두어 규제를 하지 않으면 overfitting된다. λ값을 적절히 지정하여 중간처럼 가장 에러가 낮을 때를 만들 수 있다.

---

3. Bayesian statistics and regularization

이번에도 overfitting 문제를 해결하기 위한 또 다른 tool에 대해 알아볼 것이다

먼저 우리는 parameter를 fit할 때 다음과 같이 maximum likelihood(ML)을 사용해서 다음과 같이 parameter를 구한다.

$$\theta_{\mathrm{ML}} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta).$$

응 교수님 왈

> 우리는 maximum likelihood와 같이 통계적인 절차를 통해서 이 파라미터를 추정하기 위해 노력하는 일을 한다고 한다. 이렇게 파라미터를 바라보는 통계적인 접근에는 두가지 관점이 존재하는데 이는 freqentist statistic과 Bayesian statistic이다.

먼저 **frequentist statistics**는 θ를 constant-valued but unknown으로 취급한다. 그들은 θ는 랜덤하지 않고 그냥 unknown이라고 생각하는 것이다.

이와 달리 **Bayesian**은 θ를 값이 알려지지 않은 random variable로 취급한다. 이 접근 방식으로 우리는 사전분포 $p(\theta)$를 구체화 할 수 있다. 이때 θ는 파라미터에 대한 “prior beliefs”을 나타낸다.

training set $S = \{(x^{(i)}, y^{(i)})\}$ $i = 1, \ldots, m$이 주어져 있고

새로운 x에 대한 예측을 하고 싶을 때, 우리는 파라미터에 대한 사후 분포(posterior)를 계산한다.

$$\begin{aligned} p(\theta|S) &= \frac{p(S|\theta)p(\theta)}{p(S)} \\ &= \frac{\left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)\right) p(\theta)}{\int_{\theta} \left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)p(\theta)\right) d\theta} \end{aligned}$$

이 식에서 $P(y^{(i)}|x^{(i)}; \theta)$ 는 상황에 따른 모든 model에서 도출될 수 있다.

예를 들어, Bayesian logistic regression을 이용할 경우, 다음 식을 대입한다.

(1)

$$\begin{gathered} P(y^{(i)}|x^{(i)}; \theta) = h_\theta(x^{(i)})^{y^{(i)}}(1 - h_\theta(x^{(i)}))^{1-y^{(i)}}, \\ where\ h_\theta(x^{(i)}) = 1/(1 + exp(-\theta^T x^{(i)}))^3 \end{gathered}$$

새로운 test example x에 대해 예측을 하기 위해, class label에 대한 사후분포를 θ의 사후 분포를 통해 계산한다.

(2)

$$p(y|x, S) = \int_{\theta} p(y|x, \theta)p(\theta|S)d\theta$$

이 식에서 $p(\theta|S)$는 (1)번 식에서 왔다.

그러므로 예를 들어 y|x의 기댓값을 예측하는 것이 목표라면, ouput은 다음과 같다.

$$\mathrm{E}[y|x, S] = \int_{y} yp(y|x, S)dy$$

이렇게 $p(\theta|S)$의 사전확률을 통해 기댓값 구하면서 예측하는 방식은 완전 베이지안 접근방식이라고 볼 수 있다. 그러나 컴퓨터 계산시 식(1)을 적분하는 것처럼 많은 계산량이 필요하고, closed form으로 끝나지 않기 때문에 대신 θ에 대해 approximate한 사후분포를 구해볼 것이다.

보통 θ의 사후분표를 single point estimate로 대체한다. 이러한 MAP(maximum a posteriori) estimate는 다음과 같다.

(3)

▼ ~~위에 올라가기 귀찮아서~~

$$\begin{aligned} argmax_\theta P(\theta|s) &= argmax_\theta p(s|\theta)p(\theta) \\ &= argmax_\theta \prod_{i=1}^{m} P(y^{(i)}|x^{(i)}; \theta)P(\theta) \end{aligned}$$

$$\theta_{\text{MAP}} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)p(\theta).$$

이때 우리가 주목해야 할 점은 끝에 $p(\theta)$가 있다는 점을 제외하고

이 공식이 ML(maximum likelihood) estimate 공식과 같다는 것이다.

보통

assume

$$P(\theta) : \theta \sim N(0, \tau^2 I)$$
$$P(\theta) = \frac{1}{\sqrt{2\pi}(\tau^2 I)^{\frac{1}{2}}} exp(-\frac{\theta^T(\tau I)^{-1}\theta}{2})$$

이때 **MAP값은 표준적으로 MLE보다 작기 때문에 Baysian MAP estimate가 MLE보다 overfitting에 덜 민감하다..!!**

예를들어 bayesian logistic regression은 feature의 개수가 매우 많은 상황에서도 text classification에서 효과적인 알고리즘으로 들어났다.

지윤이 왈

> 적은 데이터 셋에 효과적이다. 데이터 셋이 커지면 MLE와 MAP가 거의 유사하다.

---

4. Data split

(1) train/dev/test splits

- notation

데이터 S → Strain / Sdev / Stest

우리는 데이터를 train/ development / test으로 분리하여 모델을 만들고 평가하게 되는데 이때 사용되는 데이터와 그 목적에 대해 정확하게 알고 있는 것이 중요하다.

1. Strain를 통해 모델 M_i 을 학습하여 hypothesis h_i 를 얻는다. (option of degree of polynomial)
2. Sdev를 통해 error를 측정하고 가장 작은 error를 지닌 모델을 선택한다.

   💥 이 단계에서는 Strain를 사용하면 안된다. 왜냐하면 training set에서는 더 복잡한 모델일수록training error가 작기 때문에 항상 더 나은 모델로 판단하기 때문이다.

3. optional : Stest를 통해 알고리즘을 평가하고 이를 통해 얻은 error를 report한다.

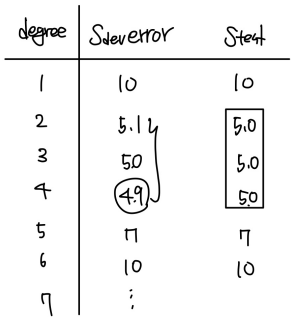
5. Model & Cross-Validation

위에서 data split과 각 데이터의 쓰임에 대해 정리하면서 모델의 평가와 선택에 대해 언급했지만 우리는 위험을 최소화 하기 위해 model selection을 진행하고 이를 위한 몇 가지 유용한 방법이 존재한다.

(1) hold-out cross validation (simple cross validation)

이는 위에 언급한 3step과 같은데, 일단 “development set” = “cross validation set”이다.

한 예시를 살펴보자. 다음과 같이 Sdev error, Stest error값이 나타난 경우



Sdev error에서 5.1, 5.0, 4.9 값은 모두 5.0정도라고 판단하면 된다. 그 이유는 Sdev error는 biased estimate로 dev set에서 미세하게 overfitting되는 경향이 있기 때문이다. Stest error값을 보면서 확인해볼 수 있는데 그래도 우리는 Sdev error를 보고 error값이 가장 작은 hypothesis를 best로 선택하면 된다.

그렇다면 data는 어떤 비율로 3개의 subset으로 나누면 될까?

- 예전에는 데이터 S에 대해
  - Strain : 70% / Sdev: 30%
  - Strain : 60% / Sdev:20% / Stest: 20%
  정도로 나눴다.
  적당한 규모의 데이터를 가진 경우에는 위에처럼 나눠서 사용하면 된다.

그러나 대규모의 데이터의 경우에는 Sdev Stest 개수가 상당히 많아지기 때문에 온라인 추천이나 매우 작은 소수점 아래자리까지 알고리즘의 성능을 고려하는 경우에는 90% / 5% / 5%로 나누는 것이 좋다. 그리고 여기에 원하는 정확도의 차이에 따라 Sdev/ Stest수를 조정하면 된다.

(2) k-fold cv

이때는 k-fold cross validation을 이용한다. k는 통상적으로 10을 사용한다.

여기서는 편의를 위해 k= 5를 사용한다.

- algorithm

1. random하게 데이터를 k개의 disjoint subset으로 나눈다.
2.

For d=1,2,3,4,5 (degree of polynomial) {

    For i=1,…,k {

        Train on (k-1) pieces

        Test on remaining 1 piece

    average(test errors)

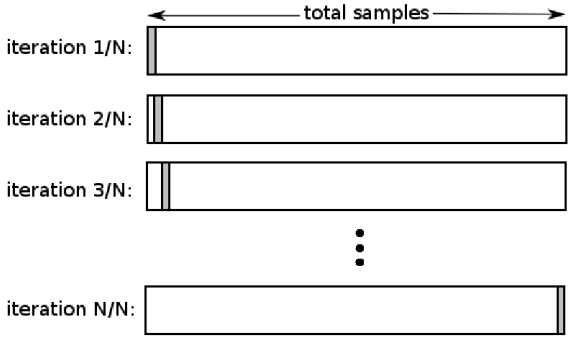
    }

}

- 장점 : 이전의 simple CV에서는 반복할때마다 데이터의 30%를 Sdev로 사용하기 위해 남겨놓아야 했다. 그러나 예를들어 10-fold cv에서는 한번 반복할 때마다 데이터의 10%만 Sdev로 사용하기 때문에 데이터 낭비가 덜하다는 장점이 있다.
  small dataset에서 효율적이다.
- 단점: 모델을 반복적으로 fitting해야 하기 때문에 컴퓨팅 파워가 중요하다.

(3) Leave one out CV

LooCV는 k-fold CV에서 k=m 즉, total data수와 K가 같은 경우이며, 데이터가 상당히 적을때 사용한다. 보통 데이터 수가 30이하일 때 사용한다.



6. Feature Selection

model selection의 특별하고 중요한 케이스에 Feature selection이 있다. supervised learning에서 feature의 개수(n)이 example의 수보다 매우 큰경우를 생각해보자. 만약 feature 중에 relevant한 경우가 있다면 simple liner classifier를 사용할 경우 training set의 수가 많지 않는 이유로 overfitting문제가 발생할 가능성이 있다.

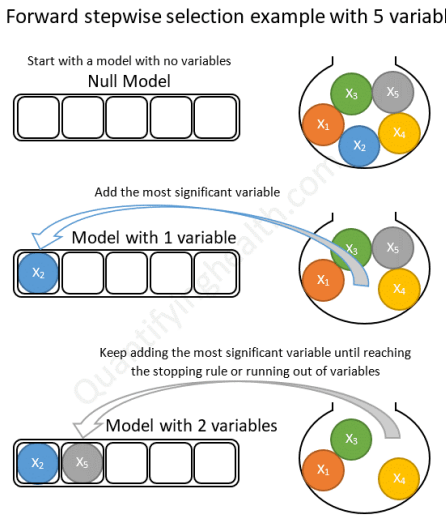
이를 위해서 우리는 feature의 개수를 줄이기 위해 feature selection을 적용한다.

(0) Exhaustive Search

완전탐색(Exhaustive search)은 feature의 모든 조합에 대한 모델을 만들고 가장 성능이 좋은 것을 고르는 방법니다. 이 방법은 언제나 global optimum을 찾을 수 있다는 장점이 있다. 그러나 많은 모델을 평가해야하기 때문에 시간이 오래 걸린다는 단점이 있다.

(1) forward search

전진선택 (forward search)는 가장 유의미한 특성을 선택해나가는 방식이다. 아무런 특성이 없는 상태에서 시작하여 특성을 늘려나가는 방향으로 나아간다. 매 단계마다 가장 성능이 좋은 특성을 선택한 후에 유의미한 성능이 없을 때까지 과정을 실행한다.



feature가 5개 x_1, x_2, x_3, x_4, x_5인 데이터셋에 전진선택방식을 적용하여 feature selection을 해보자.

1. 먼저 1개의 특성만을 가진 모델의 성능을 비교한다.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 \qquad R^2_{\text{adj}} = 0.32$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 \qquad R^2_{\text{adj}} = 0.45$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 \qquad R^2_{\text{adj}} = 0.53$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_4 x_4 \qquad R^2_{\text{adj}} = 0.35$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_5 x_5 \qquad R^2_{\text{adj}} = 0.46$$

2. x_3을 선택했을 때 성능이 가장 좋기 때문이 이 feature을 고정하고 동일한 과정을 반복한다.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_1 x_1 \qquad R^2_{\text{adj}} = 0.55$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_2 x_2 \qquad R^2_{\text{adj}} = 0.58$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 \qquad R^2_{\text{adj}} = 0.71$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_5 x_5 \qquad R^2_{\text{adj}} = 0.66$$

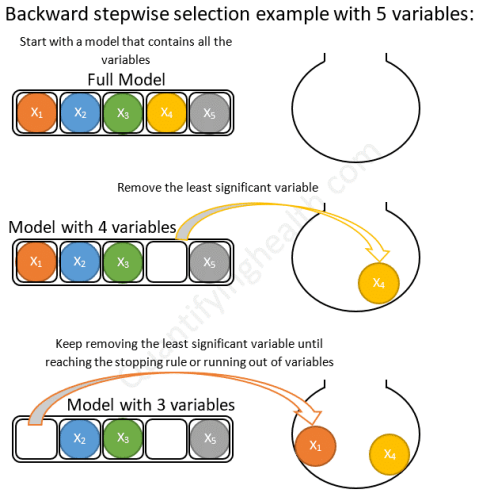
3. x_4를 추가적으로 선택했을때 성능이 가장 좋기 때문에 이 feature도 고정하고 동일한 과정을 반복한다.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_1 x_1 \qquad R^2_{\text{adj}} = 0.71$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_2 x_2 \qquad R^2_{\text{adj}} = 0.70$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_3 x_5 \qquad R^2_{\text{adj}} = 0.69$$

4. x_1, x_2, x_5 중에 어떤 특성을 추가하더라도 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4$ 보다 성능이 좋아지지 않기 때문에 이를 최종적인 feature selection의 결과로 선택한다.

(2) backward search

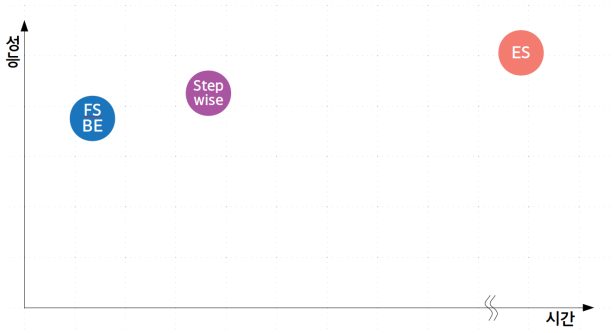
후진제거는 *무의미한 특성을 제거해나가는* 방식이다. 전진 선택과 달리 모든 feature을 가진 모델에서 feature을 하나씩 줄여나가는 방향으로 진행한다. feature을 제거했을때 가장 성능이 좋은 모델을 선택하면된다.



(3) Stepwise Selection

전진 선택의 단점은 한번 선택된 특성을 제거되지 않는다는 것이고, 후진 선택의 단점은 한번 제거된 특성은 다시 다시 선택되지 않는다는 것이다.

Stepwise selection은 전진 선택과 후진 제거 방식을 매 단계마다 반복하여 적용하는 방식입니다. 이전 두 방법보다는 더 오래 걸리지만 최적의 변수 조합을 찾을 확률이 높습니다. Stepwise selection을 시간-성능 그래프에 표시하면 아래와 같이 나타나게 됩니다.



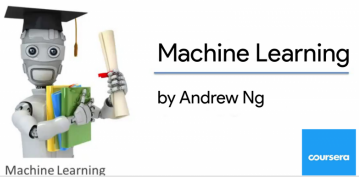
Regularized Linear Regression

https://ratsgo.github.io/machine%20learning/2017/05/22/RLR/

앤드류 응의 머신러닝(7-4): 정규화된 로지스틱 회귀

온라인 강의 플랫폼 코세라의 창립자인 앤드류 응 (Andrew Ng) 교수는 인공지능 업계의 거장입니다. 그가 스탠퍼드 대학에서 머신 러닝 입문자에게 한 강의를 그대로 코세라 온라인 강의 (Coursera.org)에서 무료로 배울 수 있습니다. 이 강의는 머신러닝 입문자들의 필수코스입니다. 인공지능과 머신러닝을 혼자 공부하면서 자연스럽게 만나게 되

https://brunch.co.kr/@linecard/487



전진 선택(Forward Selection)과 후진 제거(Backward Elimination)

Increscunt animi virescit volnere virtus by Friedrich Nietzsche

https://yngie-c.github.io/machine%20learning/2020/09/06/fs_be/