회귀(Regression)

회귀 개념

- 회귀(Regression) : ‘한바퀴를 돌아 제자리로 돌아가다’, 통계학에서 유래, 즉 어디로 회귀하는지를 아는 것이 중요

- “부모의 키가 크더라도 자식의 키가 대를 이어 무한정 커지지 않으며, 부모의 키가 작더라도 대를 이어 자식의 키가 무한정 작아지지 않는다" - 영국 통계학자 갈톤(Galton)

  ➡️ 회귀 분석은 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법

회귀 개념

- 회귀(Regression)는 여러 개의 독립변수와 한 개의 종속변수 간의 상관 관계를 모델링 하는 기법을 통칭

아파트 가격은?	=	방 갯수	+	아파트 크기	+	주변 학군	+	역세권

$$Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + \dots + W_n * X_n$$

- Y 는 종속 변수, 즉 아파트 가격
- $X_1, X_2, X_3, \dots, X_n$ 은 방 갯수, 아파트 크기, 주변 학군, 역세권등의 독립 변수
- $W_1, W_2, W_3, \dots, W_n$은 독립변수의 값에 영향을 미치는 회귀 계수

➡ 학습을 통해 최적의 회귀 계수를 찾자!

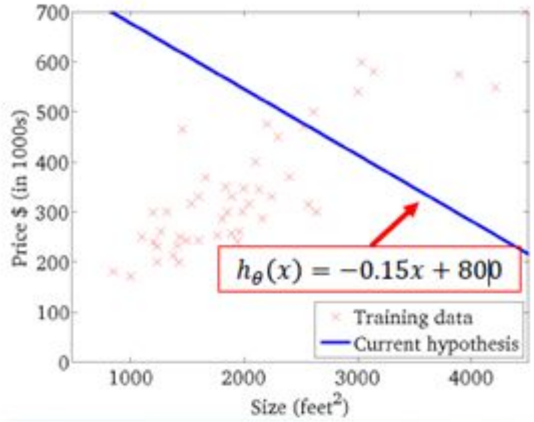
머신러닝 학습 방법

- Hypothesis 함수(H) : 머신러닝의 목적이 되는 모델
- Cost 함수(J) : Hypothesis 함수로 인해 찾아지는 예측값과 실제 값의 차이를 Cost라고 한다

➡️ 어떻게 정확한 Hypothesis를 찾아가는가?

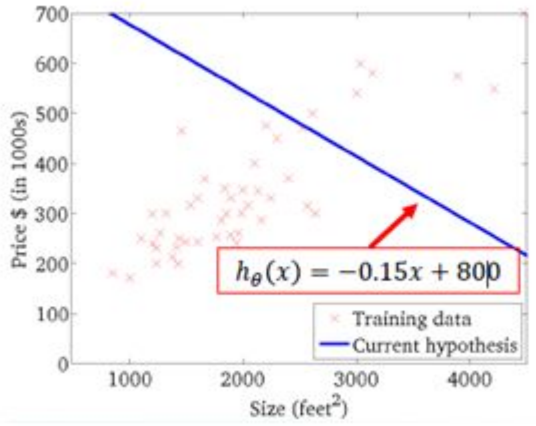
Hypothesis 함수 정리

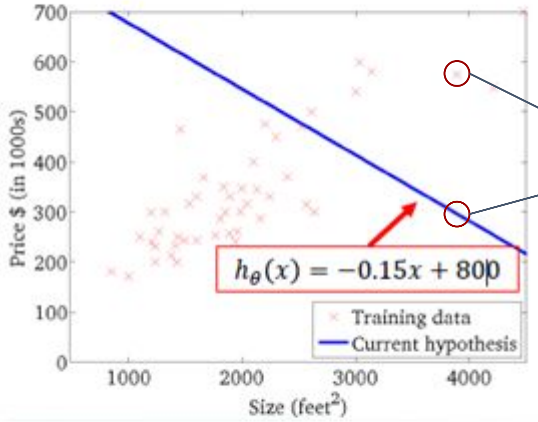
1. Hypothesis 함수는 입력 값 X 가 출력 값 Y에 영향을 미치는 정도를 의미하는 Weight 값 W(혹은 θ)로 이루어져 있다.
2. 출력값 Y는 결국 모델의 예측값이다.
3. 입력값 X는 주어지는 데이터이므로, 정확한 Hypothesis를 찾는다는 말은 Weight 값들을 찾는다는 것을 의미한다



정확한 Hypothesis를 찾는 과정

1. Hypothesis 함수를 구성하는 weight 값들에 임의의 초기값을 대입하여 첫번째 Hypothesis를 찾는다
2. 찾은 첫 번째 Hypothesis에 의한 예측값과 실제 데이터의 값의 차이를 계산한다

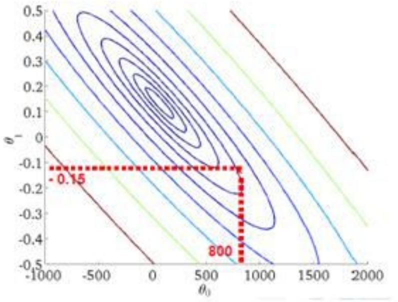


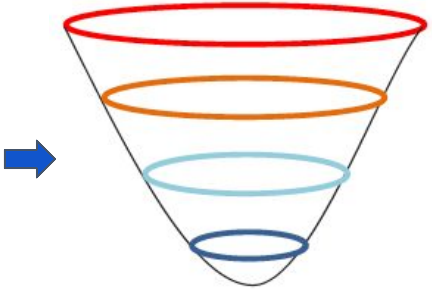


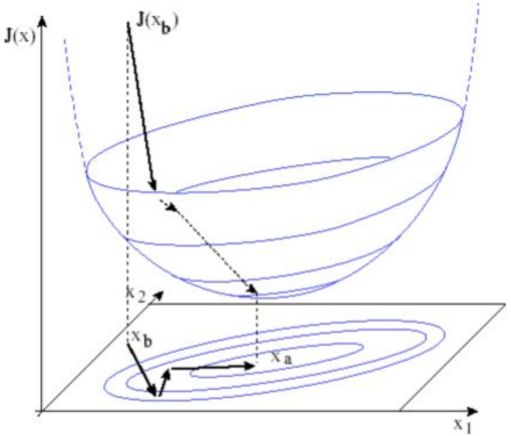
예측값과 실제 값의 차이
=> Error

Cost 함수

1. Cost 함수 : $(error)^2$ 값의 평균 (RSS : Residual Sum of Square)
2. Hypothesis 함수에 따라, 즉 Weight 값들에 따라 Cost는 달라진다. Weight 값들에 따른 Cost값들의 집합이 Cost 함수이다.







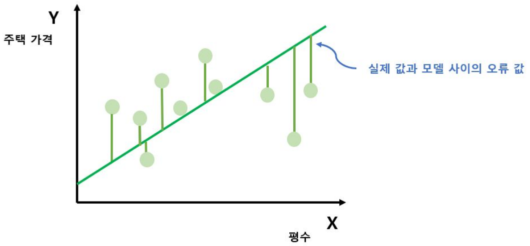
Cost 함수

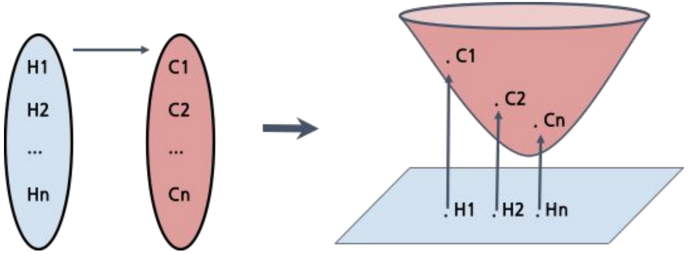
- 실제 값과 모델 사이의 오류 값(잔차)들의 합을 최소화

  = Cost 함수를 최소화

  = 최적의 회귀 계수를 찾는 과정

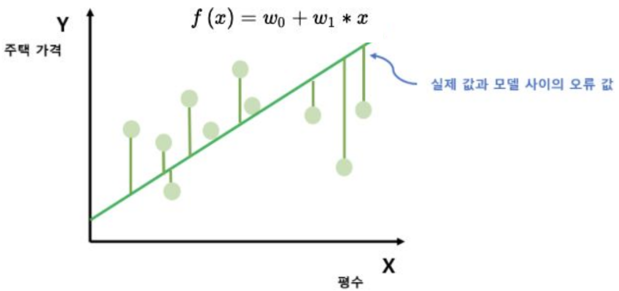
- 각각의 Hypothesis는 자신의 Cost를 가지고 있다





Cost 함수

- RSS (Residual Sum of Square) : 오류 값의 제곱을 구해서 더하는 방식



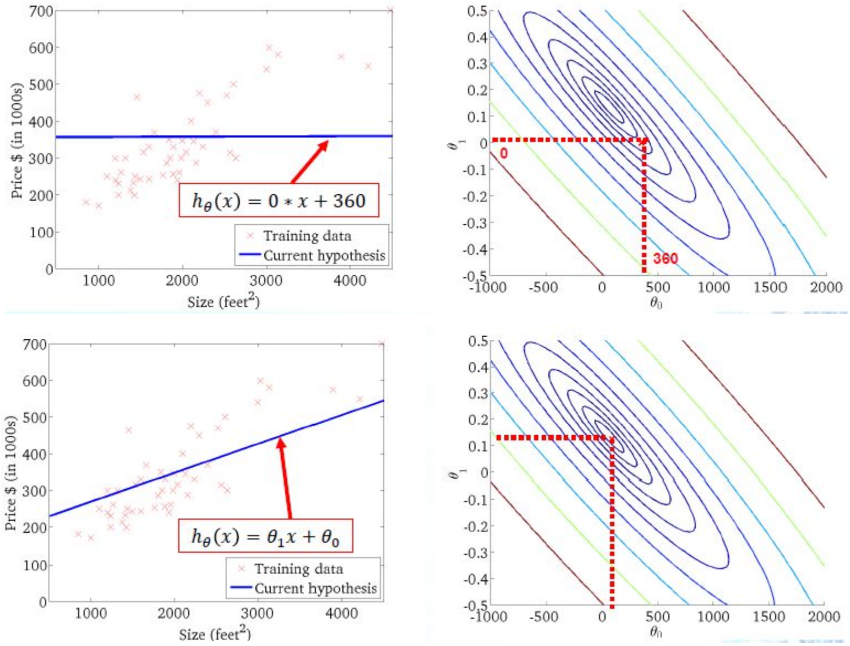
RSS = (#1 주택가격 -(w_0+w_1*#1주택 크기))^2
+ (#2 주택가격 -(w_0+w_1*#2주택 크기))^2
+ (#3 주택가격 -(w_0+w_1*#3주택 크기))^2
+ ...
+ (#n 주택가격 -(w_0+w_1*#n주택 크기))^2

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$

w변수(회귀 계수) 의 함수

정확한 Hypothesis를 찾는 과정

3. Cost 값이 낮아지는 방향으로 Weight 값들을 업데이트 하다가 Cost 값이 가장 낮을때 멈춘다.



Cost 함수 (비용 함수) 최소화

- 머신러닝 회귀 알고리즘은 데이터를 계속 학습하면서 이 비용 함수가 반환하는 값(즉, 오류 값)을 지속해서 감소시키고 더 이상 감소하지 않는 최소의 오류 값을 구하는 것
- 손실 함수(Loss function)

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$

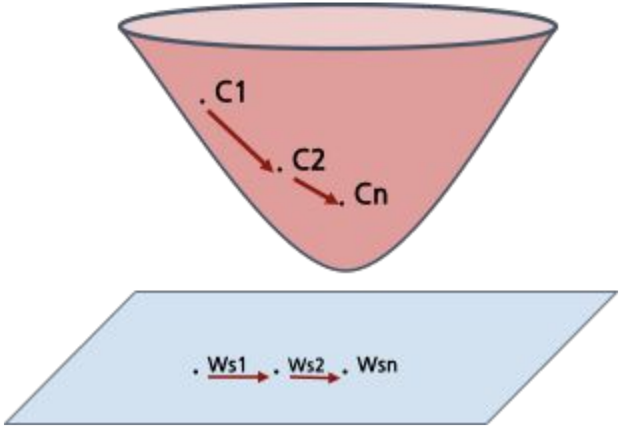
- w 파라미터의 개수가 적다면 고차원 방정식으로 비용 함수가 최소가 되는 w 변수값을 도출하겠지만 그 개수가 많다면 고차원 방정식으로 풀기 어려움

➡️ **경사 하강법 (Gradient Descent) !**

경사 하강법 (Gradient Descent)

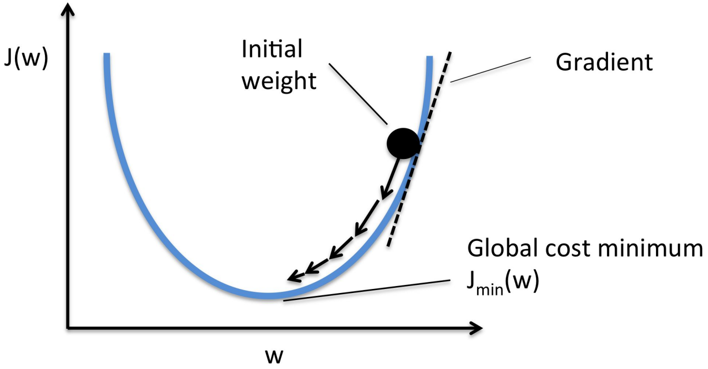
- Gradient : 기울기
- Descent : 하강

➡️ 점진적으로 반복적인 계산을 통해서 W 파라미터 값을 업데이트 해가면서 (기울기가 감소하는 방향으로 이동하면서) 오류 값이 최소가 되는 W 파라미터를 구하는 방식입니다



최적화 (Optimization)

- 어떻게 하면 오류가 작아지는 방향으로 w값을 보정할 수 있을까?

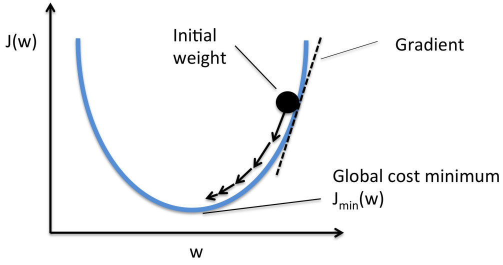


$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$

➡️ w에서 부터 미분을 적용한 뒤 이 미분 값이 감소하는 방향으로 순차적으로 w를 업데이트하면서 기울기가 0일때 멈춘다.

최적화 (Optimization)

- 비용함수를 w_0, w_1에 대해 편미분



$$\boxed{RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2}$$

➡ $$\frac{\partial R(\omega)}{\partial \omega_1} = \frac{2}{N}\sum_{i=1}^{N} - x_i * \left(\underbrace{y_i}_{실제값_i} - \underbrace{\left(w_0 + w_1 x_1\right)}_{예측값_i}\right) \qquad \frac{\partial R(\omega)}{\partial \omega_0} = \frac{2}{N}\sum_{i=1}^{N} - \left(y_i - \left(w_0 + w_1 x_1\right)\right)$$

➡ 즉, 비용함수 RSS(w0, w1) 이 최소가 되는 w1, w0를 구할 수 있다.

➡ $$W_1 = w_1 - \left(-\frac{2}{N}\sum_{i=1}^{N} x_i \times \left(y_i - \hat{h}_i\right)\right) \qquad \hat{h}_i = w_0 + w_1 x_i$$

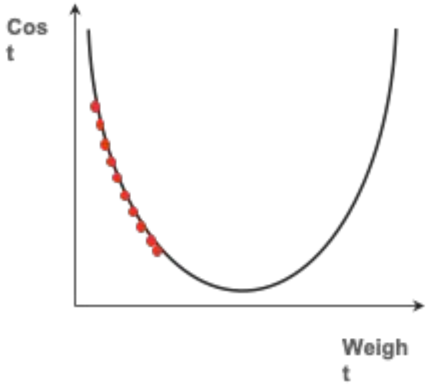
➡ 편미분 값이 클 수 있기 때문에 보정 η 수 를 곱하는데 이를 '학습률'이라고 한다.

➡ $$\boxed{W_1 = W_1 - \eta\left(-\frac{2}{N}\sum_{i=1}^{N} x_i \times \left(y_i - \hat{h}_i\right)\right)}$$

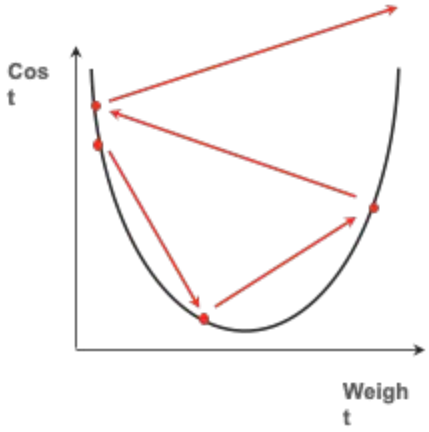
학습률 (Learning Rate)

- 한번의 학습으로 얼마만큼 학습해야 할지, 즉 매개변수 w값을 얼마나 갱신하느냐를 정하는 것

- too small ⇒ 너무 오래 걸린다



- too large ⇒ 학습이 안된다



최적화 (Optimization) 프로세스

✓ Step 1 : w_1, w_0 를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산

✓ Step 2 : w_1 을 $w_1 - \eta \frac{2}{N}\sum_{i=1}^{N} x_i * \left(y_i - \hat{h}_i\right)$, w_0 을 $w_0 - \eta \frac{2}{N}\sum_{i=1}^{N} \left(y_i - \hat{h}_i\right)$ 으로 업데이트 한 후 다시 비용 함수의 값을 계산

✓ Step 3 : 비용함수의 값이 감소했으면 다시 Step 2를 반복합니다. 더 이상 비용 함수의 값이 감소하지 않으면 그 때의 w_1, w_0 를 구하고 반복을 중지합니다.

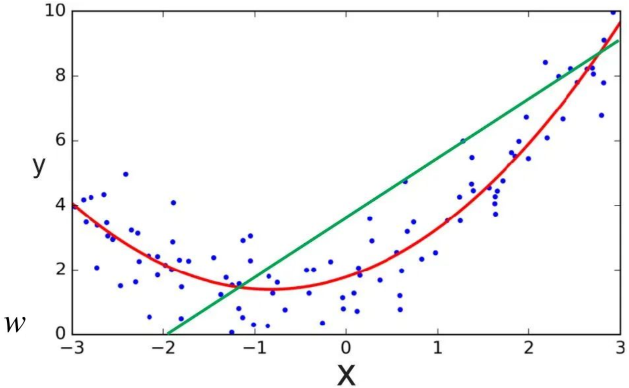
회귀(Regression)

실습

gradient descent

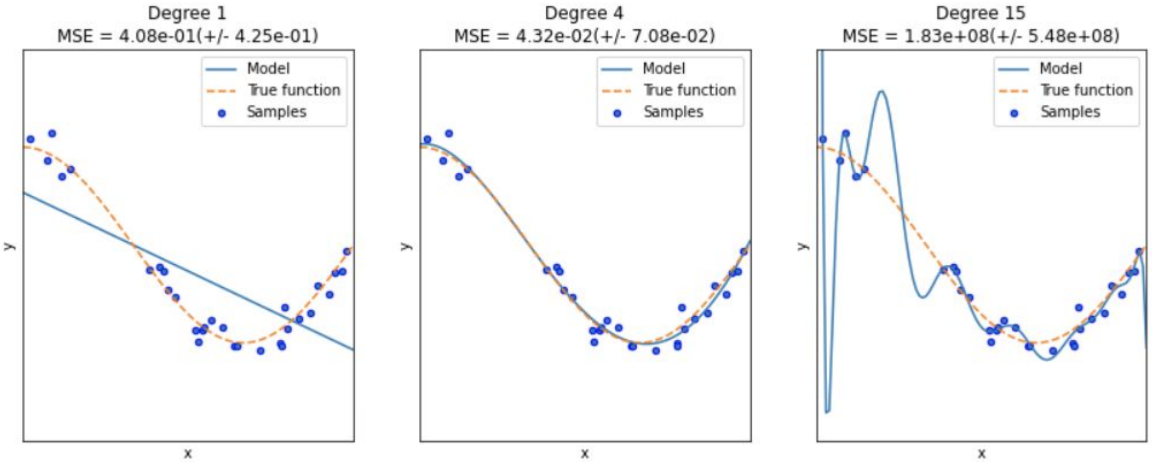
다항회귀 (Polynomial Regression)

- 회귀식이 $y = w_0 + w_1*x_1 + w_2*x_2 + w_3*x_1*x_2 + w_4*x_1^2 + w_5*x_2^2$ 과 같이 2차 3차 방정식과 같은 다항식으로 표현되는 것



- 데이터에 대해 독립변수(특성, Feature)에 대해 Target Y 값의 관계를 단순 선형 회귀 직선형으로 표현한 것 보다 곡선형으로 표현한 것이 더 예측 성능이 높다

과대적합(Overfitting) / 과소적합(Underfitting)



Underfitting	Well fitting	Overfitting
train data에 대한 예측을 잘 못한다	new data에 대한 예측을 잘한다	new data에 대한 예측을 잘 못한다

정규화(Regularization)

- Degree = 15의 다항회귀는 지나치게 모든 데이터에 적합한 회귀식을 만들기 위해서 다항식이 복잡해지고 회귀 계수가 매우 크게 설정이 되면서 과대적합이 되고 평가 데이터 세트에 대해서 형편없는 예측 성능을 보임

  ➡️ 회귀 모델은 적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야 함
  RSS 만 최소화 했을때 과적합

비용 함수 목표 = $$Min\left(\underbrace{RSS(w)}_{\text{학습데이터 잔차 오류 최소화}} + \underbrace{\alpha * \|W\|_2^2}_{\text{회귀계수 크기 제어}}\right)$$

α : 학습 데이터 적합 정도와 회귀 계수 값을 크기 제어를 수행하는 튜닝 파라미터

- 비용함수에 α값으로 페널티를 부여해 회귀 계수 값의 크기를 감소시켜 과적합을 개선하는 방식을 규제(정규화, regularization) 이라고 함

정규화(Regularization) 의 유형

- L2 규제 (릿지, Ridge) : $\alpha * \|W\|_2^2$ 와 같이 W에 제곱에 대해 패널티를 부여하는 방식, L2 규제를 적용한 회귀를 릿지 회귀라고 함
- L1 규제 (라쏘, Lasso) : $\alpha * \|W\|_1$ 와 같이 W의 절대값에 대해 패널티를 부여하는 방식, L1 규제를 적용한 회귀를 라쏘 회귀라고 함. L1 규제를 적용하면 영향력이 크지 않는 회귀 계수 값을 0으로 변환
- Elastic Net : L2 규제와 L1 규제를 결합한 회귀, 라쏘 회귀가 서로 상관관계가 높은 피처들의 경우에 이들 중에서 중요 피처만을 선택하고 다른 피처들은 모두 회귀 계수를 0으로 만드는 특징으로 인해 α 값에 따라 회귀 계수의 값이 급격히 변동할 수도 있는데, 엘라스틱넷 회귀는 이를 완화하기 위해 L2 규제를 라쏘 회귀에 추가, 수행 시간이 오래 걸림