**머신러닝 11주차 실습**

20145128

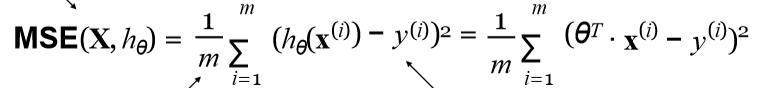
박인근

**선형 회귀**

선형 회귀 모델을 훈련시키려면 RMSE를 **최소화 하는 찾아야 한다.**

실제로는 RMSE보다 MSE(평균 제곱 오차)를 최소화하는 것이 같은 결과를 내면서 더 간단하다

가설의 MSE 식



**선형 회귀 : 정규 방정식**

비용 함수를 최소화하는 찾기 위한 해석적인 방법(수학 공식)이 있다.

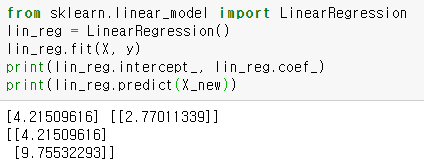
이를 정규방정식 이라고 한다.



: 비용함수를 최소화 하는 값

**선형 회귀 : 사이킷런**

사이킷런 코드



* Intercept : 편향
* Coef : 가중치

특성이 매우 많고 훈련 샘플이 너무 많아서 메모리에 모두 담을 수 없을때, 경사 하강법을 사용한다.

**경사 하강법**

최적화 알고리즘

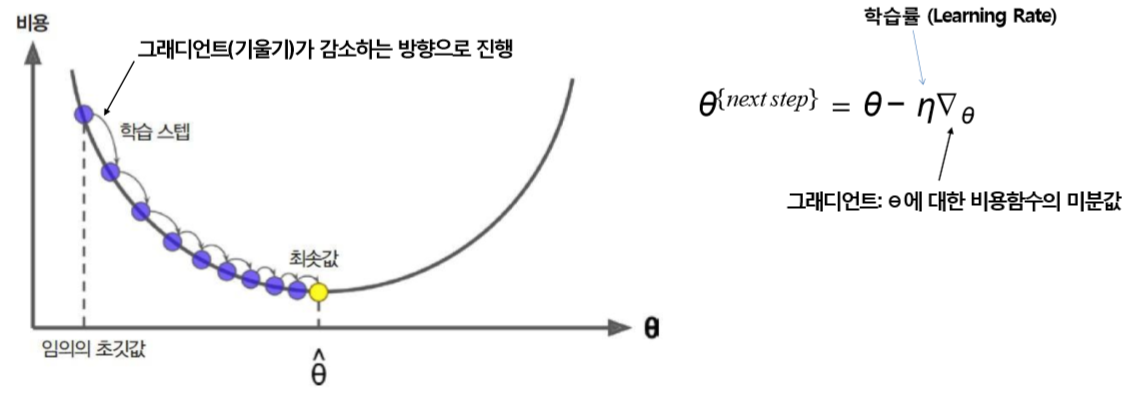
기본 아이디어 : 비용 함수를 최소화하기 위해 반복해서 파라미터를 조정

산속에서 길을 잃었으면 지면의 기울기를 이용해서 빠르게 내려가는, 가장 가파른 길을 따라 아래로 내려가는 것에 비유할 수 있다.

파라미터 대해 비용 함수의 현재 그래디언트를 계산한다. 그리고 그래디언트가 감소하는 방향으로 진행하고. 그래디언트가 0이 되면 최솟값에 도달한 것이다.

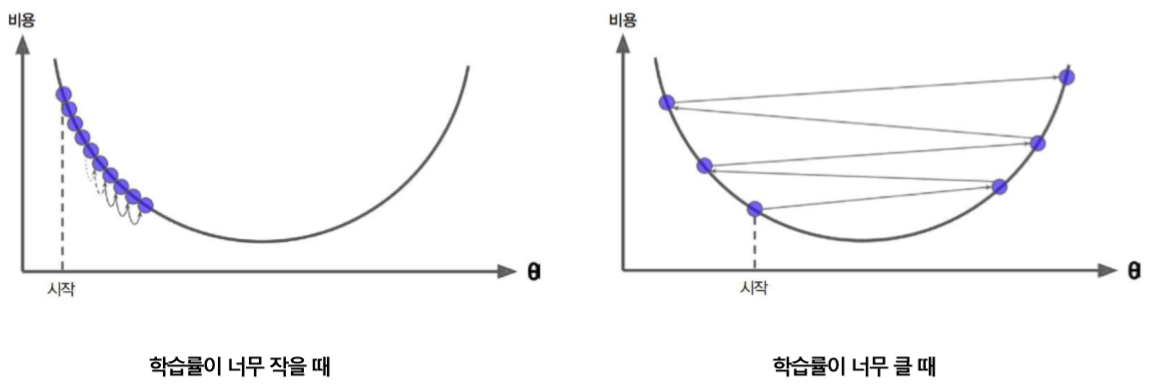
를 임의의 값으로 시작해서(무작위 초기화) 한번에 조금씩 비용 함수가 감소되는 방향으로 진행하여 알고리즘이 최솟값에 수렴할 때까지 점진적으로 향상시킨다.

가장 중요한 파라미터 : 학습률(learning rate)



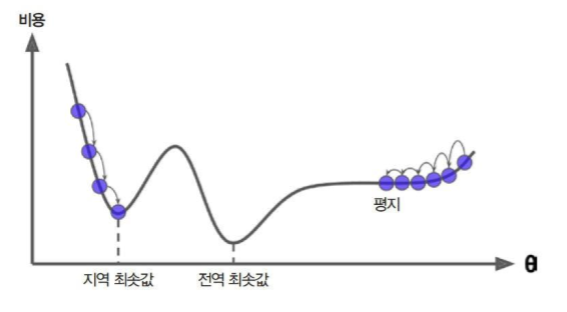
**학습률이 작을 때** : 반복을 많이 해야하기 때문에 시간이 많이 소요 된다.

**학습률이 클 때** : 그림과 같이 반대편으로 건너뛰게 되어서 이전보다 더 높은 곳으로 올라가게 될지도 모른다.



모든 비용 함수가 이전과 같이 이상적인 모양은 아니다.

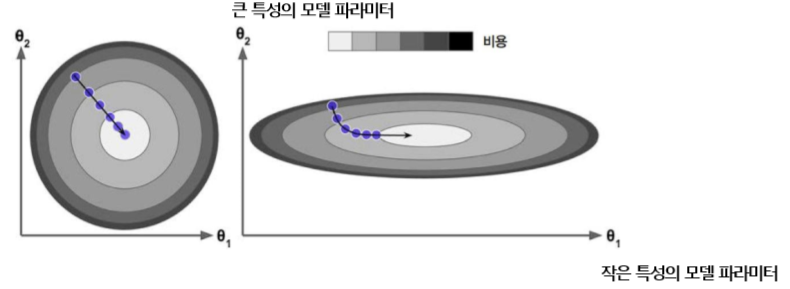
초기 값이 임의로 정해지기 때문에 아래와 같이 왼쪽에서 시작하면 지역 최소값(local minimum)에 수렴하게 되고, 오른쪽에서 시작하면 평탄한 지역을 지나기 위해 시간이 오래 걸린다.



선형 회귀를 위한 MSE 비용 함수는 곡선에서 어떤 두 점을 선택해서 선을 그어도 곡선을 가로지르지 않는 볼록 함수다. (전역 최솟값만 가진다.)

또한 연속적인 함수이고 기울기가 갑자기 변하지 않는다.

위의 두 사실로부터 경사 하강법이 전역 최솟값에 가깝게 접근할 수 있다는 것을 보장한다.



왼쪽은 두개의 특성의 스케일이 같은 훈련 세트이고, 오른쪽은 두개의 특성의 스케일이 다른 훈련 세트다.

왼쪽 : 빠르게 도달한다.

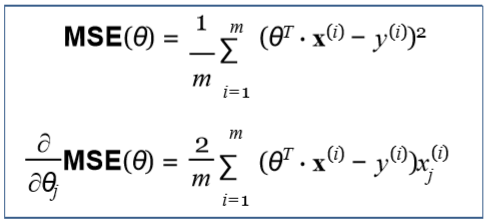
오른쪽 : 처음엔 빠르지만 점점 시간이 오래 걸린다. (최솟값에 도달하긴 한다.)

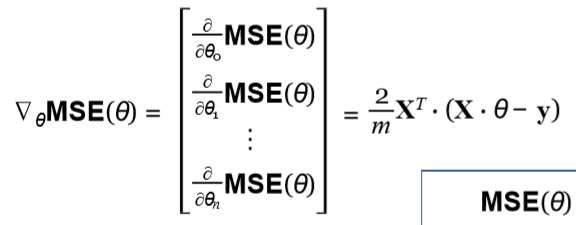
**배치 경사 하강법**

경사 하강법을 구현하려면 모델 파라미터에 j대해 비용 함수의 기울기를 계산해야한다.

다시 말해 j가 조금 변경될 때 비용 함수가 얼마나 바뀌는지 계산해야 한다.

이것을 **편도 함수**라고 한다.





편도 함수를 각각 계산하는 대신 위에 식을 사용해서 한꺼번에 계산 할 수 있다.

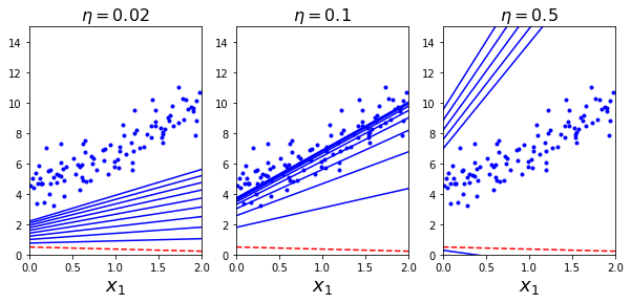
위의 수식은 매 경사 하강법 스텝에서 전체 훈련 세트 X에 대해 계산한다.

이 알고리즘을 배치 경사 하강법이라고 하고, 매우 큰 훈련 세트에서는 아주 느리다.

하지만 특성 수에 민감하지 않으면서 수 십만 개의 특성에서 선형 회귀를 훈련시키려면 정규 방정식 보다 경사 하강법을 사용하는 편이 훨씬 빠르다.

위로 향하는 기울기 벡터가 구해지면 반대 방향인 아래로 진행해야한다. 따라서 서 를 빼야 한다. 여기서 학습률이 사용된다.





왼쪽은 학습률이 너무 낮다.

-> 최적점에 도달 하겠지만 시간이 오래 걸린다.

가운데는 학습률이 적당하다.

오른쪽은 학습률이 너무 높다.

-> 알고리즘의 진행 형태가 발산한다.

적절한 학습률을 찾으려면 그리드 탐색을 사용한다.

-> 하지만 수렴하는데 너무 오래 걸리는 모델을 막기 위해 반복 횟수를 제한

반복횟수는 너무 작으면 최적점에 도달하기 전에 알고리즘이 멈추며, 너무 크면 모델 파라미터가 더 이상 변화하지 않는 동안 시간을 낭비하게 된다.

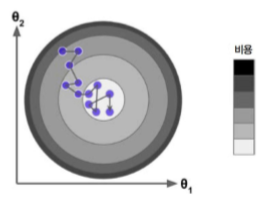
해결책은 반복 횟수를 크게 지정하고 기울기 벡터가 아주 작아지면, 알고리즘을 중지하는 것이다.

학습률과 같은 하이퍼파라미터를 이론적으로 구할 수 있는 방법은 없다.

-> 특히 딥러닝은 많은 하이퍼파라미터를 가지기 때문에 과학보다는 기술에 가깝다.

**확률적 경사 하강법**

* 확률적 경사 하강법은 매 스텝에서 딱 한 개의 샘플을 무작위로 선택하고 그 하나의 샘플에 대한 기울기를 계산한다.
  + 확실히 빠르다.
  + 매 반복에서 하나의 샘플만 메모리에 있으면 되므로 매우 큰 훈련 세트도 훈련시킬 수 있다.
  + 확률적이기 때문에 불안정 하다.
  + 최소값에 다다를 때 까지 부드럽게 감소하지 않고 위아래로 요동치면서 평균적으로 감소한다.
  + 비용 함수가 불규칙할 경우 지역 최솟값을 건너뛸 수 있게 도와주므로 전역 최솟값을 찾을 가능성이 높다.



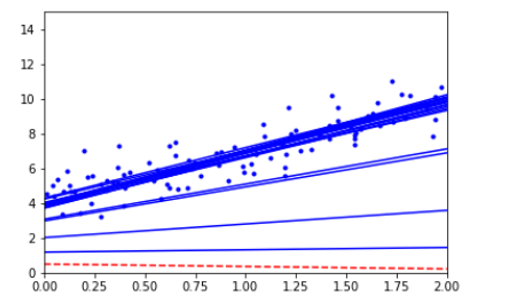
무작위성은 지역 최솟값에서 탈출시켜줘서 좋지만 알고리즘을 전역 최솟값에 다다르지 못하게 한다는 점에서 좋지 않다.

이에 대한 해결책은 학습률을 점진적으로 감소시키는 것이다.

시작할때는 학습률을 크게하고 점차 작게 줄여서 알고리즘이 전역 최솟값에 도달하게 해준다.

한 반복에서 m번 반복

이때 각 반복 : 에포크(epoch)라고 한다



샘플을 무작위로 선택하기 때문에 어떤 샘플은 한 에포크에서 여러 번 선택될 수 있다.

에포크마다 모든 샘플을 사용하게 하려면 훈련 세트를 섞은 후 차례대로 하나씩 선택하고 다음 에포크에서 다시 섞는 식의 방법을 사용할 수 있다.

하지만 위의 방법은 더 늦게 수렴하는 결과를 만든다.

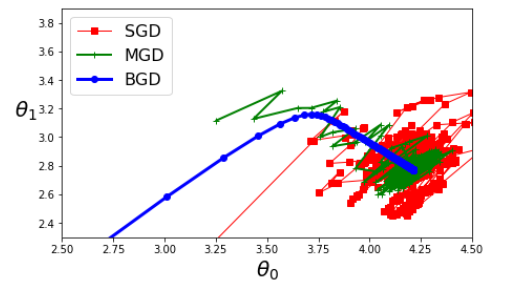
사이킷런에서는 SGDRegressor을 제공한다.

**미니배치 경사 하강법**

각 스텝에서 전체 훈련 세트, 하나의 샘플을 기준으로 기울기를 계산하지 않고 **미니배치**라고 불리는 임의의 작은 샘플 세트에 대해 기울기를 계산한다.

미니배치를 어느정도 크게 하면 SGD보다 덜 불규칙하게 움직이고 SGD보다 최소값에 더 가까이 도달하게 될 것이다.

하지만 한편으로는 로컬 미니멈에서 빠져나오기는 더 힘들지도 모른다.



세가지의 경사 하강법을 비교:

모두 최소값 근처에 도달했지만 배치 경사 하강법의 경로는 실제로 최소값에서 멈춘 반면 확률적 경사 하강법과 미니배치 경사 하강법은 근처에서 맴돈다.

하지만 배치 경사 하강법에는 매 스텝에서 많은 시간이 소요되고, 확률적 경사 하강법과 미니배치 경사 하강법도 적절한 학습 스케줄을 이용하면 최소값에 도달할 수 있다.

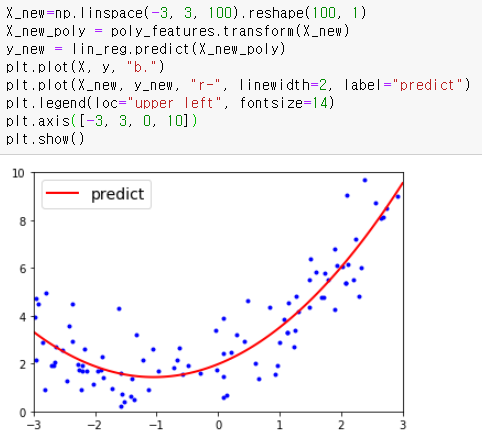
**다항 회귀**

직선이 아닌 곡선 형태인 경우

간단한 방법으로는 각 특성의 거듭제곱을 새로운 특성으로 추가하고, 이 확장된 특성을 포함하는 데이터셋에 선형 모델을 훈련시킴.

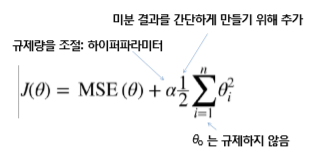
사이킷런의 PolynomialFeatures를 사용해 훈련 데이터를 반환한다.

Degree가 2면 ab, a2, b2 가 추가된다.



**릿지 회귀**

규제가 추가된 선형 회귀 버전



학습 알고리즘을 데이터에 맞추는 것뿐만 아니라 모델의 가중치가 가능한 작게 유지되도록 노력한다.

규제항은 훈련하는 동안 비용 함수에 추가 된다.

모델의 훈련이 끝나면 모델의 성능은 규제가 없는 성능 지표로 평가한다.

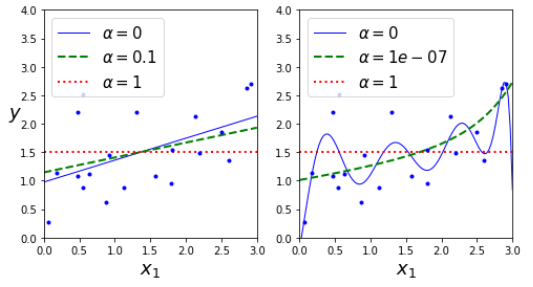
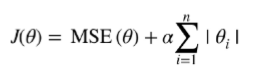
가 커지면 가중치가 0에 가까워지고, 가 0에 가까우면 MSE만 남는다.

릿지 회귀는 입력 특성의 스케일에 민감하기 때문에 수행하기 전에 데이터의 스케일을 맞추는 것이 중요하다.

**라쏘 회귀**

선형 회귀의 또 다른 규제된 버전

릿지 회귀 처럼 비용 함수에 규제항을 더하지만 아래와 같이 사용한다.



덜 중요한 특성의 가중치를 완전히 제거하려고 한다.

예를 들어, 오른쪽 그래프에서 점선이 2차 방정식 처럼 보이고 선형이다.

차수가 높은 다항 특성의 가중치가 모두 0이 되었다.

다시 말해서 라쏘 회귀는 자동으로 특성 선택을 하고 희소 모델을 만든다.

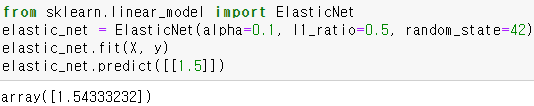
릿지 회귀는 규제가 강해지면 함수가 부드러운 선의 형태로 그려지지만 차수는 유지된다.

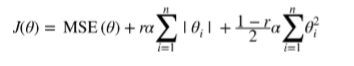
라쏘 회귀는 규제가 강해지면 다항식에서 일부 항이 없어지고 차수가 낮아지며 특징 선택으로 사용할 수 있다.

**엘라스틱넷**

릿지 회귀와 라쏘 회귀를 절충한 모델

규제항은 릿지와 회귀의 규제항을 단순히 더해서 사용하며, 혼합 정도는 혼합 비율 r을 사용해 조절한다.

r=0이면 엘리스틱넷은 릿지 회귀와 같고, r=1이면 라쏘 회귀와 같다.



선형회귀, 랏지, 라쏘, 엘라스틱넷을 사용하는 경우

* 규제가 약간이라도 있는 것이 대부분의 경우에 좋으므로 일반적으로 평범한 선형 회귀는 피해야한다.
* 릿지가 기본이 되지만 실제로 쓰이는 특성이 몇 개뿐이라고 의심되면 라쏘나 엘라스틱넷이 낫다.
* 이 모델들은 이전에 이야기 한 것 처럼 불필요한 특성의 가중치를 0으로 만든다.
* 특성 수가 훈련 샘플 수보다 많거나 특성 몇 개가 강하게 연관되어 있을 때는 엘라스틱넷을 사용한다

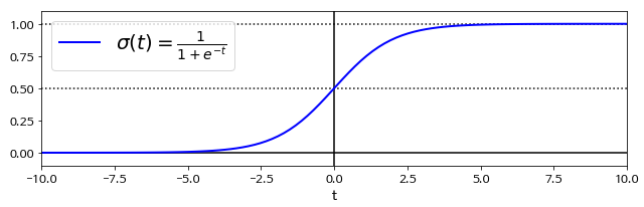
**로지스틱 회귀**

분류에서도 사용할 수 있는 회귀 알고리즘

* 샘플이 특정 클래스에 속할 확률을 추정하는데 사용
* 선형 방정식을 시그모이드 함수에 통과시켜서 0 ~ 1 사이의 확률을 계산

로지스틱 회귀 모델 : 

로지스틱 함수 : 



로지스틱 회귀 모델이 샘플 x가 양성 클래스에 속할 확률을 추정하면 이에 대한 예측을 쉽게 구할 수 있다.



**소프트맥스 회귀**

로지스틱 회귀 모델을 여러 개의 이진 뷴류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원하도록 일반화

**개념**

회귀 모델이 각 클래스에 대한 점수를 계산하고, 그 점수에 소프트맥스 함수를 적용하여 각 클래스의 확률을 추정



**비용 함수는 크로스-엔트로피 함수를 사용**

두 개의 클래스만 존재한다면(K=2), 이 비용 함수는 로지스틱 회귀의 비용 함수와 같다.

