**Assignment 1**

**조선해양공학과 2017-35978 이성준**

**1. Data 취득**

교육 여건과 부동산 가격의 상관관계를 파악하기 위해 아래와 같은 데이터를 이용하였다.

1) **교육 여건 관련 지표**: 서울시 각 구별 인구 천 명 당 사설학원 수 (25개 구)

* 출처: 통계지리정보서비스 ([https://sgis.kostat.go.kr](https://sgis.kostat.go.kr/))
* 원출처: 행정안전부 주민등록인구현황, 각시도 통계연보

2) **부동산 가격 관련 지표**: 2018년 8월 아파트 평균 단위 매매가격

* 출처: 한국감정원 ([http://r-one.co.kr/rone/resis/statistics/statisticsViewer.do?  
  menuId=HOUSE\_21121&autoExec=Y](http://r-one.co.kr/rone/resis/statistics/statisticsViewer.do?menuId=HOUSE_21121&autoExec=Y))

단, 부동산 가격은 지역 별로 많은 차이를 보이며, 전국을 대상으로 조사할 경우 교육여건 외에도 다른 요소들이 많이 작용하기 때문에 서울시 내 25개 구에 대해서만 국한하여 분석 대상을 정하였다. 각각 얻어진 데이터를 x, y 그래프 상에 매핑하여 나타내면 아래와 같다. (x축은 인구 천 명 당 사설학원 수, y축은 단위면적 m2당 아파트 평균 매매가)

**2. Linear Regression**

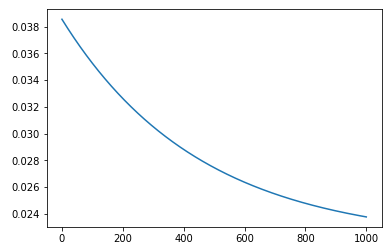
**2.1 Gradient Descent Method**

총 25개의 데이터를 학습과 테스트를 위해 20 : 5로 나누었으며, 학습을 위한 주요 코드는 다음과 같다.

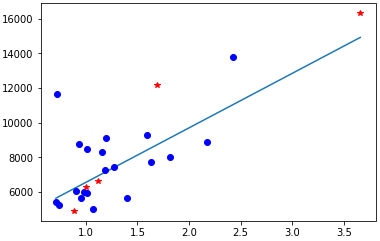
|  |
| --- |
| for i in range(epoch):  Y\_pred = w\*X\_ + b  Loss = 0.5 \* (Y\_ - Y\_pred) \*\* 2  mse = Loss.sum() / ndata\_tr    dLdw = (Y\_pred - Y\_) \* X\_  dLdb = (Y\_pred - Y\_)  dLdw\_mean = dLdw.sum() / ndata\_tr  dLdb\_mean = dLdb.sum() / ndata\_tr    w = w - alpha \* dLdw\_mean  b = b - alpha \* dLdb\_mean |

Loss function으로는 mean square error를 사용하였고, weight와 bias에 대한 gradient를 구하여 이를 반복 차감하였다. X\_와 Y\_는 각각 20개의 원소를 갖는 배열이다. 전체 학습 데이터에 대한 gradient 평균을 취하여 갱신하였다. (즉, mini-batch를 사용하지 않음)

Learning rate는 0.001, epoch는 1000으로 설정하여 학습을 진행하였다. 아래는 학습이 진행됨에 따라 각 epoch에서의 loss 값을 나타낸 것이다. 완만하게 수렴하고 있음을 볼 수 있다.



아래는 1000 epoch 학습이 완료된 후의 linear regression 결과를 보여준다. 실선이 linear regression에 해당되는 직선이며, 파란색 점이 training data, 빨간색 별 모양이 test data이다.



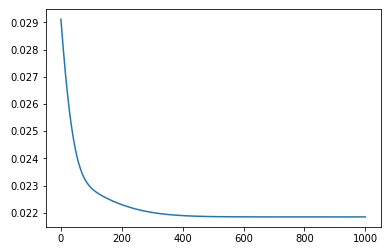
* Loss (MSE) for training data after training: **0.0231** (upon normalized data)
* Loss (MSE) for test data: **0.0202**

**2.2 Accelerated Optimization**

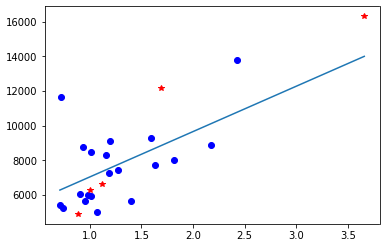
Accelerated 최적화 방법으로는 ADAM 알고리즘을 사용하였다. 아래는 이를 이용한 코드이다.

|  |
| --- |
| for i in range(epoch):  Y\_pred = w\*X\_ + b  Loss = 0.5 \* (Y\_ - Y\_pred) \*\* 2  mse = Loss.sum() / ndata\_tr    dLdw = (Y\_pred - Y\_) \* X\_  dLdb = (Y\_pred - Y\_)  dLdw\_mean = dLdw.sum() / ndata\_tr  dLdb\_mean = dLdb.sum() / ndata\_tr    grad = np.array([dLdw\_mean, dLdb\_mean])    # update momentum  m = beta1 \* m + (1 - beta1) \* grad    # update RMSProp  v = beta2 \* v + (1 - beta2) \* grad\*\*2    # unbias  t = i + 1  m\_hat = m / (1 - beta1\*\*t)  v\_hat = v / (1 - beta2\*\*t)    # update weights  [w, b] = [w, b] - alpha / np.sqrt(v\_hat + e) \* m\_hat |

Learning rate는 0.001로 동일하게 설정하여 학습을 진행하였다. 아래는 학습 진행에 따른 loss 감소를 보여주는데, 동일한 조건임에도 불구하고 gradient descent 방법보다 훨씬 빠르게 수렴함을 알 수 있다.



학습 완료 이후 linear regression 수행 결과는 아래의 직선과 같다.



* Loss (MSE) for training data after training: **0.0218** (upon normalized data)
* Loss (MSE) for test data: **0.0267**

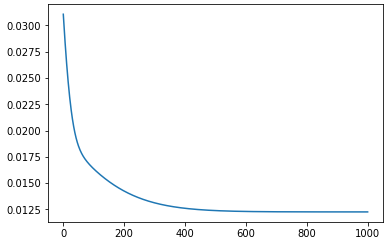
또한 본 보고서에는 비교자료를 수록하지 않았지만, 초기 weight 값을 바꿔가면서 학습을 수행했을 때 gradient descent 방법에서는 결과가 매우 민감하게 달라지는 것을 확인할 수 있었다. 위의 결과만 보면 Gradient descent 방법이 loss가 더 적게 나왔지만, 초기 값이 달라졌을 때 비교적 큰 loss 값으로 수렴하는 현상이 많이 나타났다. 그에 비해 ADAM 최적화 방법은 초기값에 대해 덜 민감하게 반응하였는데, 이는 gradient descent 방법에 비해 ADAM 최적화 방법이 local minimum에 덜 빠진다는 것을 의미한다.

**3. Logistic Regression**

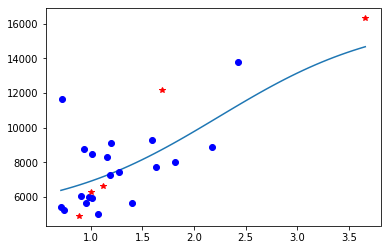
추가로 sigmoid 함수를 이용하여 logistic regression을 수행해 보았다. Loss function은 동일하게 mean square error를 사용하였고, 최적화 방법으로는 ADAM을 사용하였다.

|  |
| --- |
| for i in range(epoch):  **Y\_pred = 1 / (1 + np.exp(-1 \* (w\*X\_ + b)))**  Loss = 0.5 \* (Y\_ - Y\_pred) \*\* 2  mse = Loss.sum() / ndata\_tr    **dLdw = (Y\_pred - Y\_) \* Y\_pred \* (1 - Y\_pred) \* X\_**  **dLdb = (Y\_pred - Y\_) \* Y\_pred \* (1 - Y\_pred)**  dLdw\_mean = dLdw.sum() / ndata\_tr  dLdb\_mean = dLdb.sum() / ndata\_tr    grad = np.array([dLdw\_mean, dLdb\_mean])    # update momentum  m = beta1 \* m + (1 - beta1) \* grad    # update RMSProp  v = beta2 \* v + (1 - beta2) \* grad\*\*2    # unbias  t = i + 1  m\_hat = m / (1 - beta1\*\*t)  v\_hat = v / (1 - beta2\*\*t)    # update weights  [w, b] = [w, b] - alpha / np.sqrt(v\_hat + e) \* m\_hat |

Linear regression에 비해 달라지는 부분은 굵게 표시하였다. 예측값을 구하는 식이 linear에서 sigmoid logistic function으로 변경되었고, 이에 따라 gradient를 구하는 식이 달라졌다. 나머지는 동일하다. ADAM 최적화 방법을 적용하였기 때문에 마찬가지로 수렴속도가 빠른 것을 볼 수 있다.



아래는 학습이 완료된 이후 regression을 수행한 결과이다. 그래프가 약간 s자 모양을 갖는 것을 볼 수 있으며, 이 또한 추세를 잘 반영하고 있다는 것을 확인할 수 있다.



* Loss (MSE) for training data after training: **0.0123** (upon normalized data)
* Loss (MSE) for test data: **0.0206**

**4. 결론**

서울 25개 구의 인구 천 명 당 사설학원 수와 단위 면적당 평균 아파트 매매가에 대한 상관관계를 가지고 기계학습 방법을 사용하여 regression을 수행해 보았다. 이를 통해 확인한 것은 첫째, 두 지표가 비교적 강한 양의 상관관계를 갖는다는 것이고, 둘째는 기계학습의 최적화 방법을 사용하여 linear regression 또는 logistic regression을 훌륭히 수행할 수 있다는 것이다. 또한 gradient descent, ADAM 최적화 방법(accelerated 최적화 방법)의 수렴속도 등을 비교해 볼 수 있었다.

두 지표의 양의 상관관계에 대해 부연하자면 인구당 사설학원 수가 적음에도 불구하고 (거의 최소 수준) 부동산 가격이 높게 형성된 튀는 점(outlier)이 하나 있는데, 이는 주거기능보다는 도심기능을 수행하는 용산구이다. 이를 제외한다면 이 두 지표는 더욱 강한 상관관계로 묶이게 된다. 낯선 곳에 갔을 때 주변에 학원이 많이 보인다면 그 동네는 집값이 비싼 동네이다.