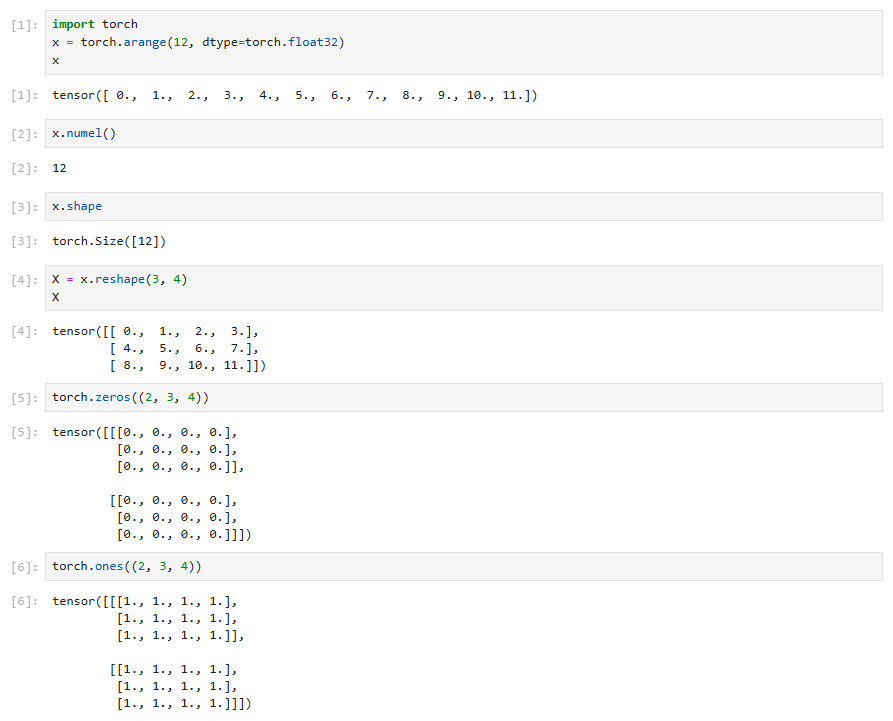
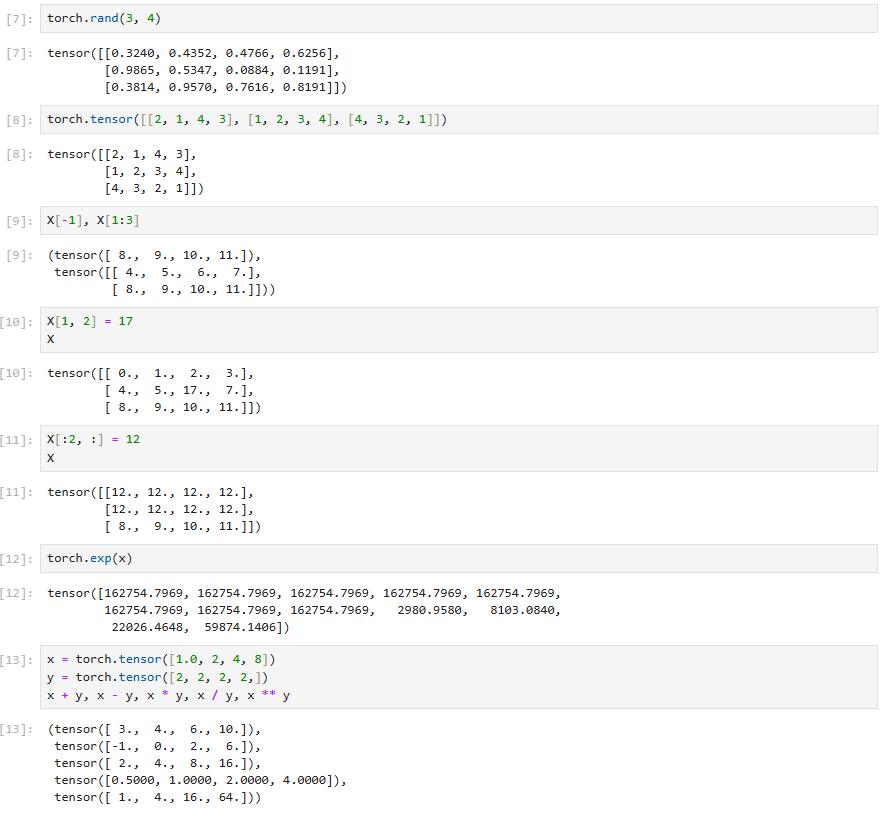
**딥러닝 HW 1**

**2020320131**

**컴퓨터학과 박수빈**

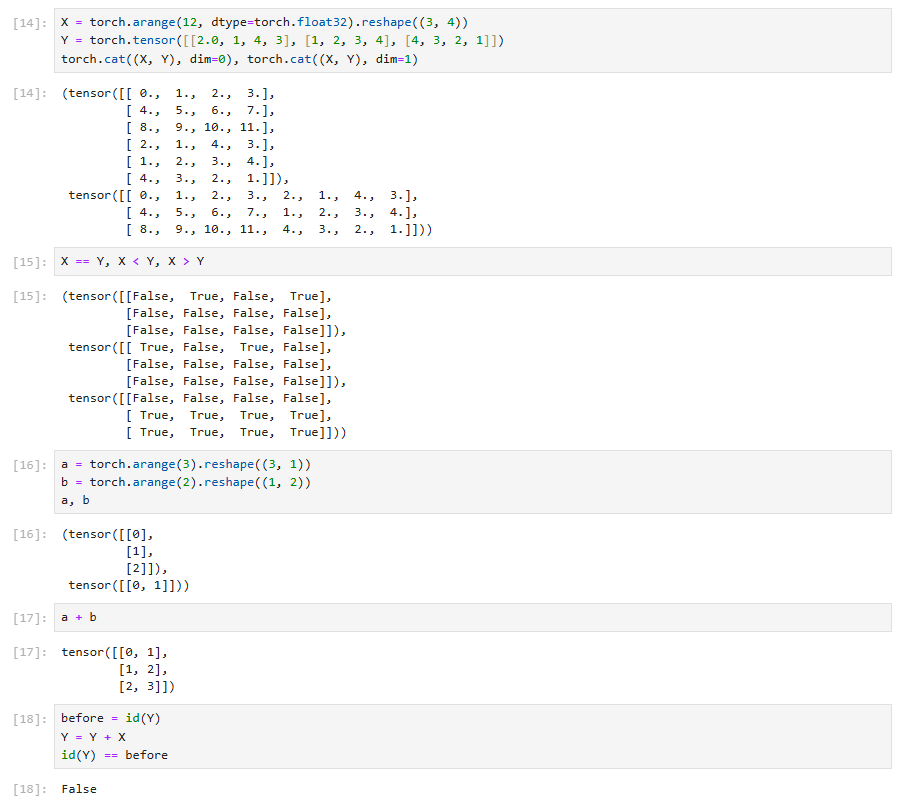
**2.1. Data Manipulation**



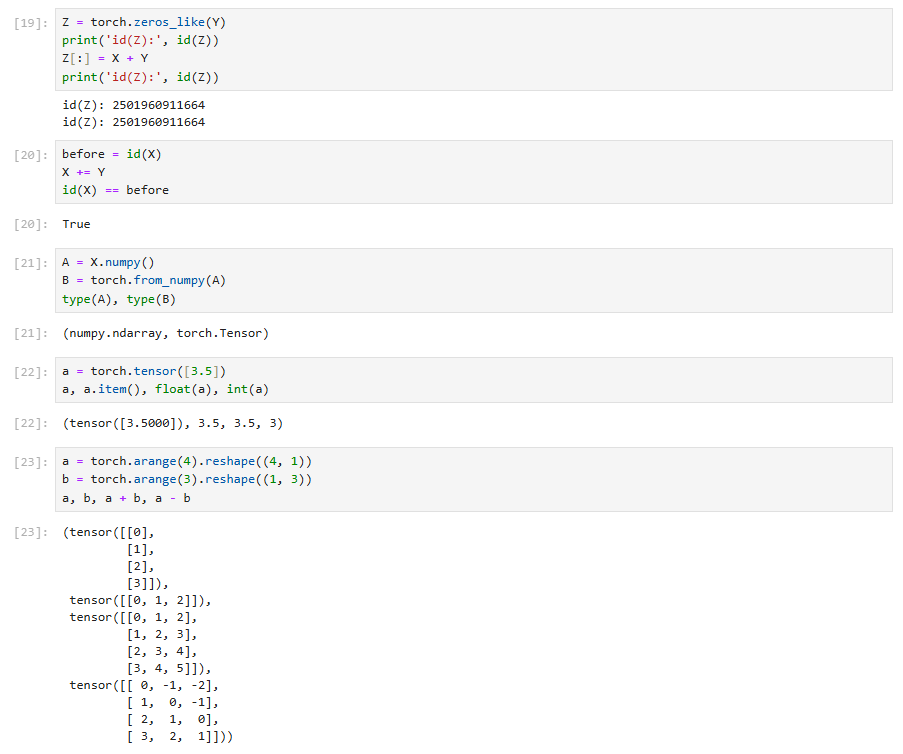
텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

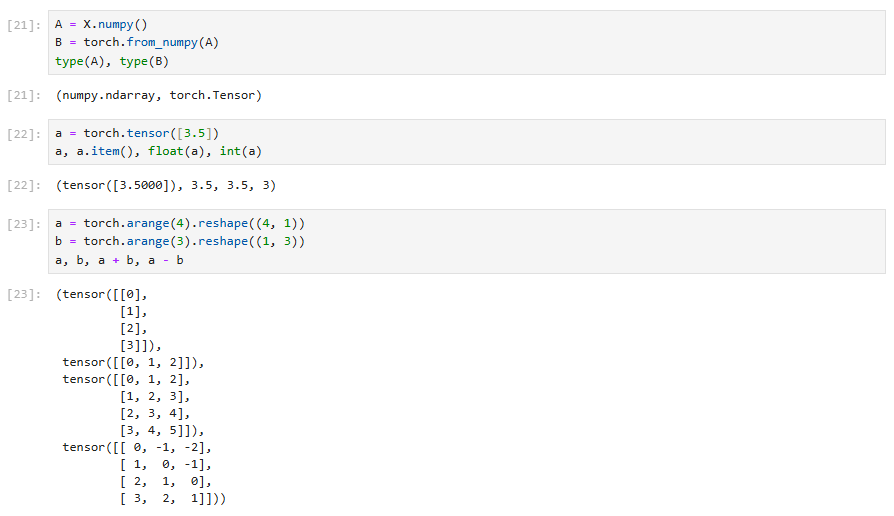
자동 생성된 설명

x의 exp 값을 봤을 때, X를 통해 값을 변경하면 x에서도 동일한 위치의 값이 함께 변경된다는 것을 알 수 있다. 이를 통해 torch.reshape()는 새로운 메모리를 할당하는 것이 아니며, 파이썬에서의 ‘=’ 기호는 shallow copy에 해당한다는 것을 추측할 수 있다. 또한 뒤에서 언급되는 x.numpy()도 기존의 텐서와 변환된 배열이 같은 메모리 공간을 공유한다고 한다.

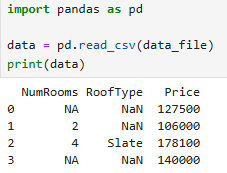
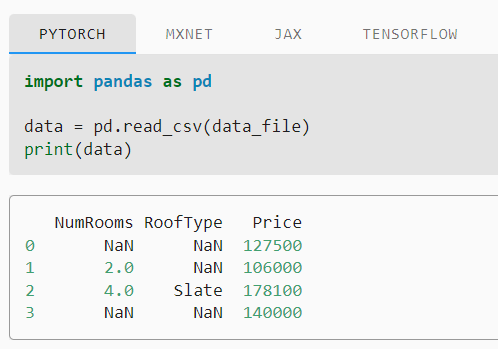
텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

또한 ‘X=X+Y’처럼 변수를 새로 선언하는 것은 변수가 가리키는 주소값이 변하게 하지만, ‘X[:]’처럼 slice notation을 이용하거나 식을 ‘X += Y’처럼 작성하는 것은 주소값이 변하지 않아 메모리 낭비를 막을 수 있다.



**2.2. Data Preprocessing**



NumRooms의 형식이 예시와 다르게 문자열로 출력되어서 to\_numeric을 통해 숫자로 변환했다.

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

get\_dummies를 통해 열의 값에 따른 새로운 열을 생성할 수 있다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

to\_numpy를 사용하면 flase는 0으로, true는 1로 변환된다.

**2.3. Linear Algebra**

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

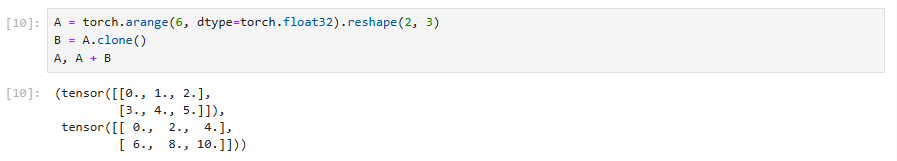
‘A == A.T’ 연산은 A 매트릭스 전체가 AT 매트릭스 전체와 동일한지를 판단하지 않고, 각 행과 열의 일치 여부를 모두 따져서 반환한다. 이를 통해 매트릭스 간 비교 연산은 같은 위치의 값들마다 이뤄짐을 알 수 있다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

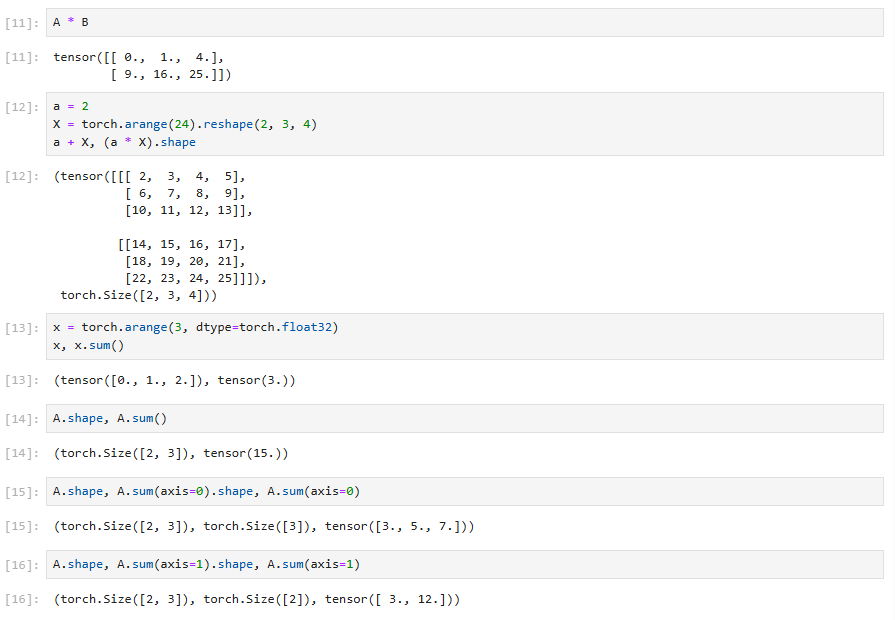
자동 생성된 설명 텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

reshape는 길이가 맞지 않으면 오류가 발생한다.



‘B = A.clone()’과 같이 clone을 이용하면 새 메모리를 할당하면서 복사할 수 있다.



cat에서 dim=0이 row 방향, dim=1이 column 방향이었듯이 sum도 axis=0은 같은 row에 있는 값들을 더하고, axis=1은 같은 column에 있는 값들을 더한다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 라인, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Hadamard product, dot product, 매트릭스-벡터/매트릭스 곱셈 등을 모두 함수 또는 기호로 계산할 수 있다.

**2.5. Automatic Differentiation**

텍스트, 라인, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

x.requires\_trad(True)를 통해 x의 기울기를 새 메모리를 할당하지 않고도 추적할 수 있다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

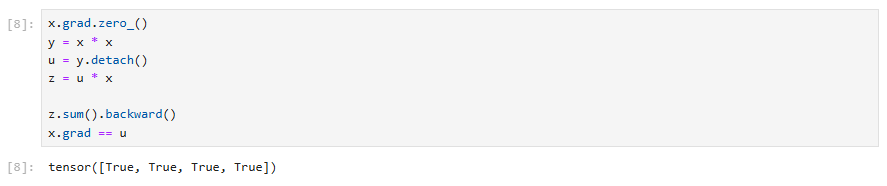
자동 생성된 설명

‘y.backward()’는 y를 x에 대해 미분하여 기울기를 계산하고, 그 결과를 x.grad에 저장한다. 또한 ‘x.grad.zero\_()’를 이용하여 기울기를 초과할 수 있다.

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

‘y.backward(gradient=torch.ones(len(y)))’는 y의 각 요소에 1을 곱한 상태로 backward 계산을 하며, ‘y.sum().backward()’와 동일하다. ‘y.sum().backward()’가 더 빠르고 직관적이다.

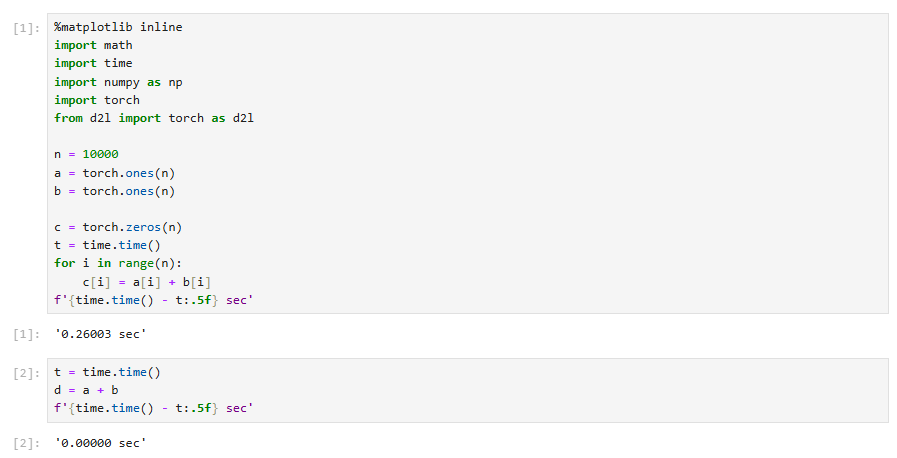


detach()를 통해 그래프에서 분리한 u는 상수 취급되기 때문에 z를 x에 대해 미분하면 u가 남게 된다.

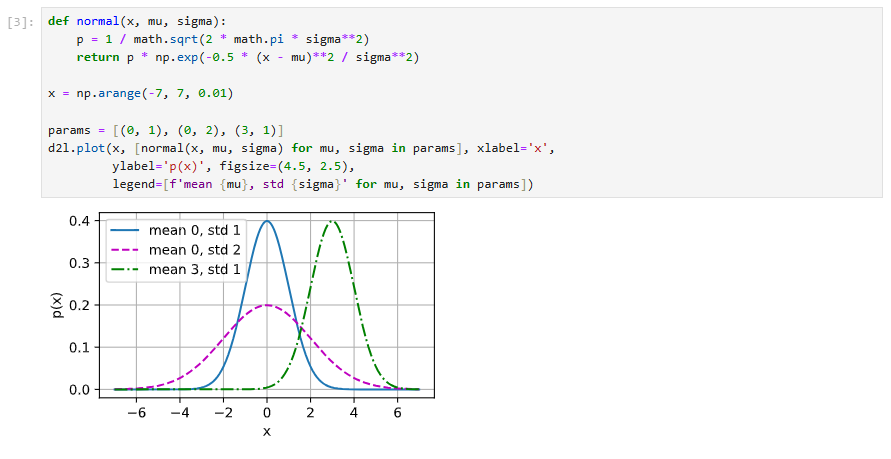


함수의 기울기도 계산할 수 있다.

**3.1. Linear Regression**



‘time.time()’으로 현재 시간을 구하고, 이를 통해 연산에 소요된 시간을 계산할 수 있다.



‘d2l.plot()’으로 선형 회귀식의 그래프를 그릴 수 있으며, x, y 축의 라벨 또한 지정할 수 있다.

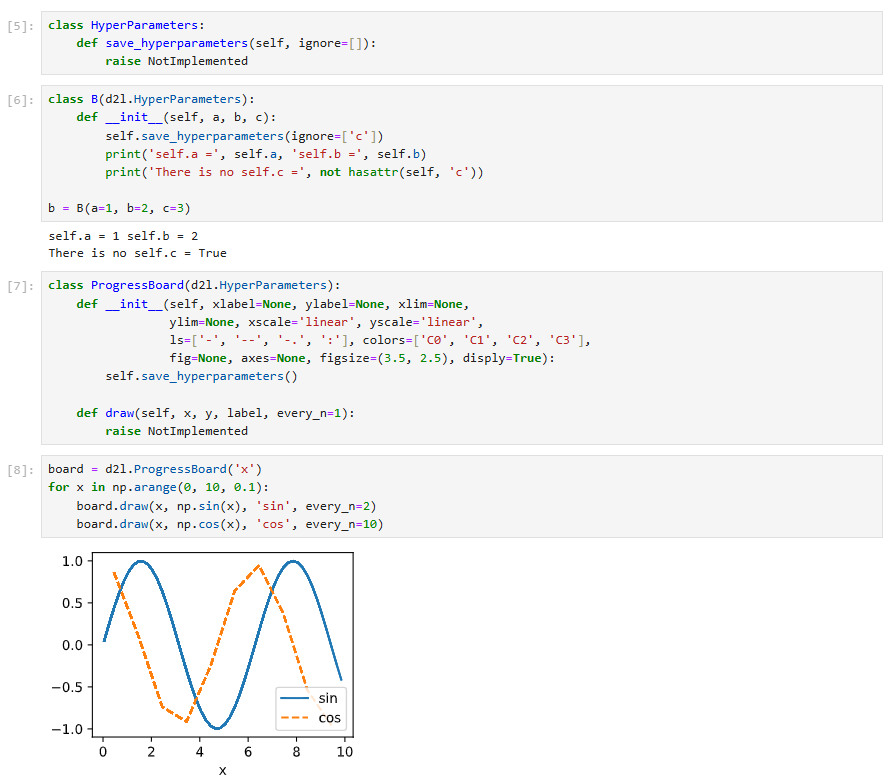
stochastic gradient descent optimizer를 통해 최적화할 수 있다.

선형 회귀는 신경과학에서 많이 사용된다.

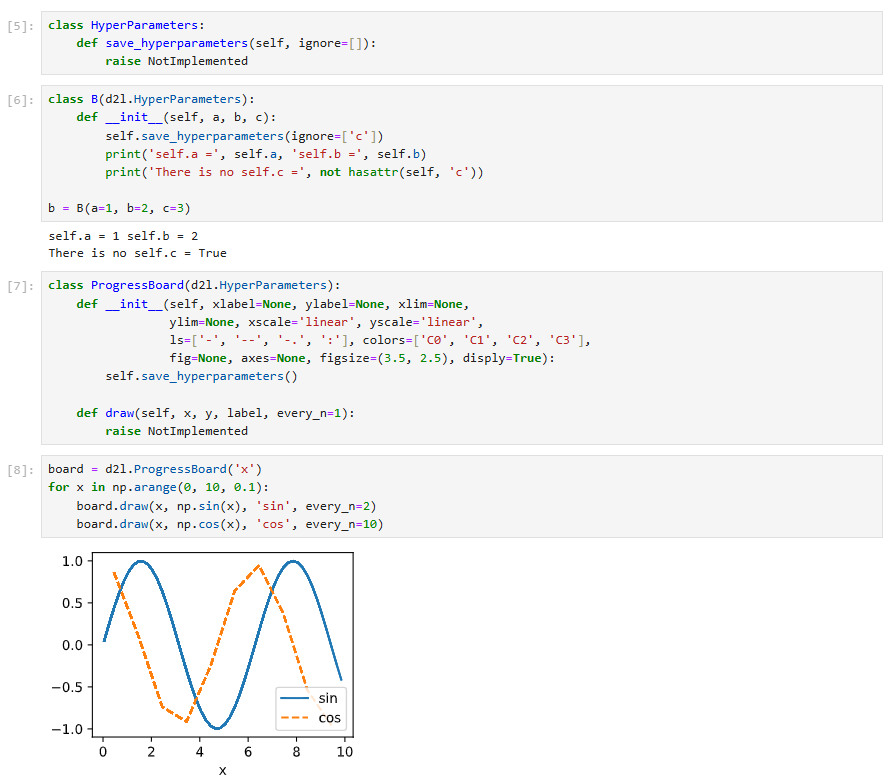
**3.2. Object-Oriented Design for Implementation**



‘@add\_to\_class’ 데코레이터를 통해 do 함수를 A 클래스에 추가하는 코드이다.



c 인자는 저장하지 않는다.



그래프가 애니메이션으로 그려지며, sin 함수는 2 간격으로, cos 함수는 10 간격으로 그려진다.



모델 클래스를 만들고,



Training dataset을 불러올 클래스를 만든 후,



Trainer 클래스로 모델 클래스를 학습시킬 수 있다.

**3.4. Linear Regression Implementation from Scratch**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

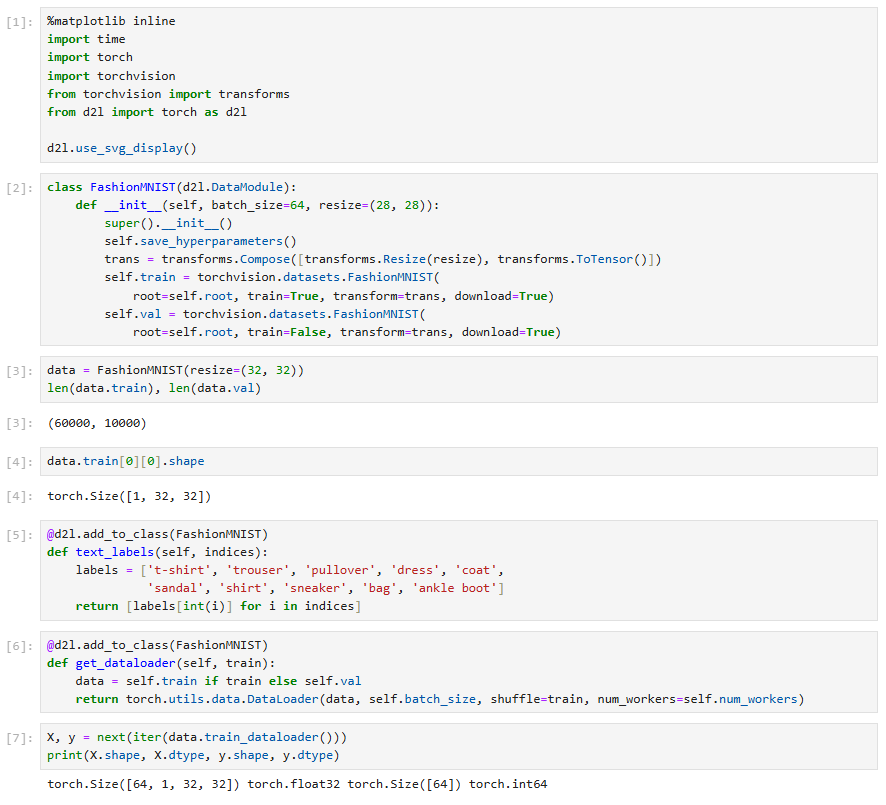
선형 회귀 모델을 가중치 [2, -3.4], 편향 4.2를 가진 트레이닝 데이터셋으로 epoch=3만큼 반복하여 학습시키고, 훈련 손실(train loss)과 검증 손실(val loss)을 기록하여 그래프로 시각화 해준다. 학습이 반복될수록 손실이 감소하는 것을 그래프로 확인할 수 있다.

**4.1. Softmax Regression**

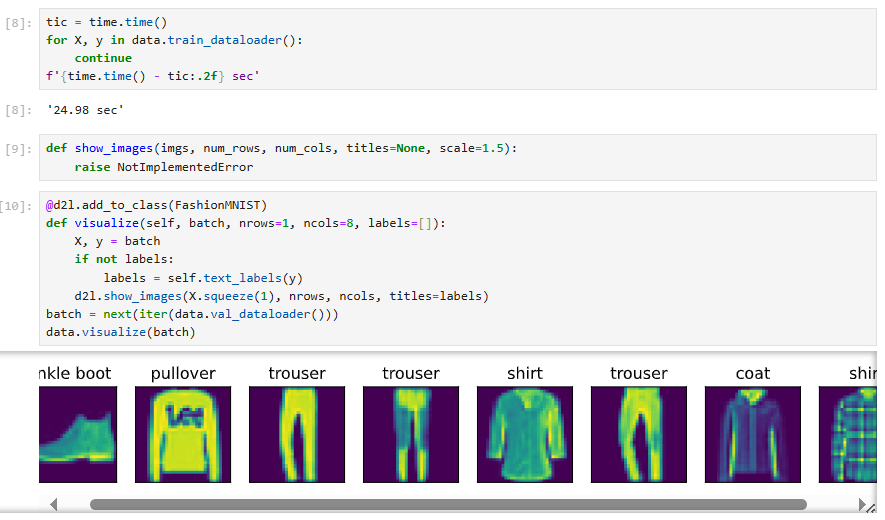
분류는 입력 데이터가 어느 범주(클래스)에 속하는지 예측하는 것이다. 분류의 방법으로는 선형 모델, 소프트맥스, 벡터화가 있다. 이 중 소프트맥스는 nonnegativity와 각 클래스의 확률값의 합이 1이 되는 것을 보장한다.

분류의 손실 함수에는 Log-Likelihood와 Softmax and Cross-Entropy Loss가 있다. 소프트맥스는 확률을 계산하고, 크로스 엔트로피가 확률이 실제와 얼마나 차이 나는지 측정한다.

**4.2. The Image Classification Dataset**



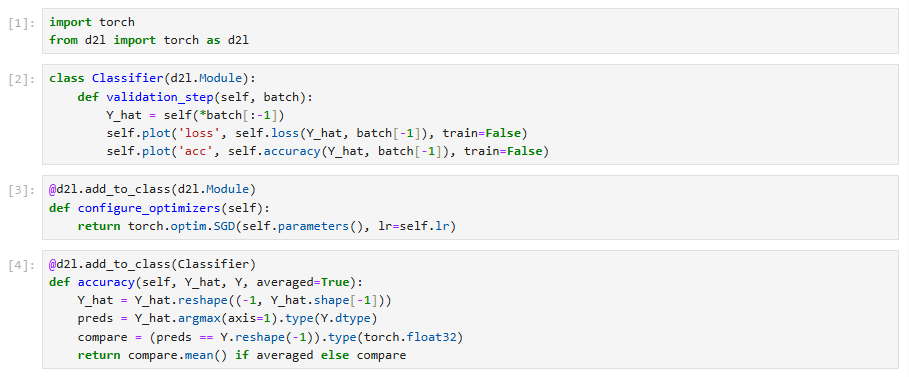
64개의 이미지로 구성된 미니배치를 불러온다.

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

visualize 메서드를 만들어 분류 이미지를 시각화 할 수 있다.

**4.3. The Base Classification Model**



손실 값과 정확도를 측정하고, 선형 회귀처럼 stochastic gradient descent optimizer를 이용해 최적화할 수 있다.

**4.4. Softmax Regression Implementation from Scratch**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Softmax로 분류한 확률의 합이 각각 1임을 확인할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

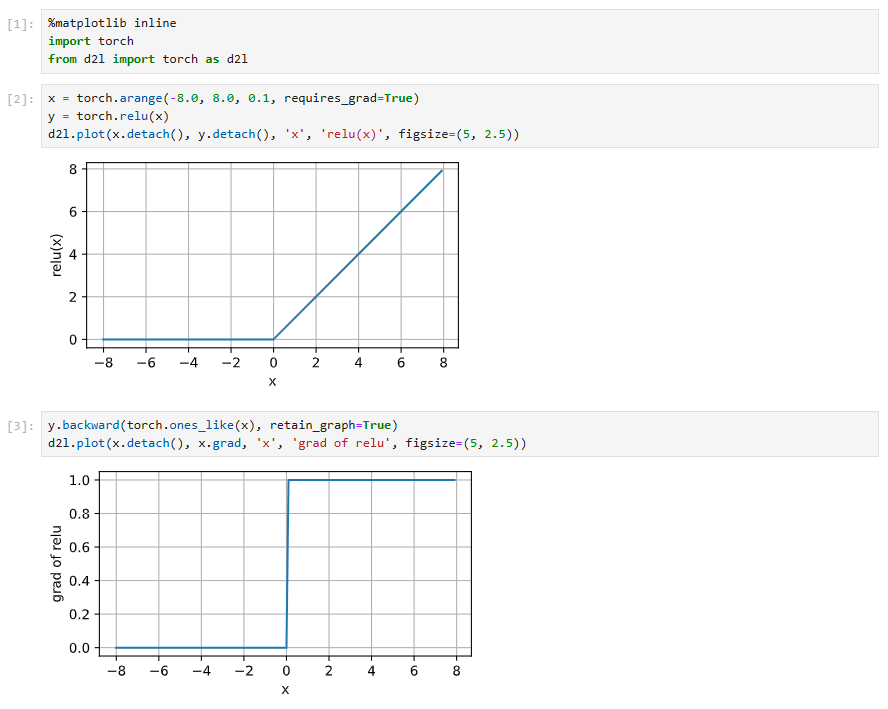
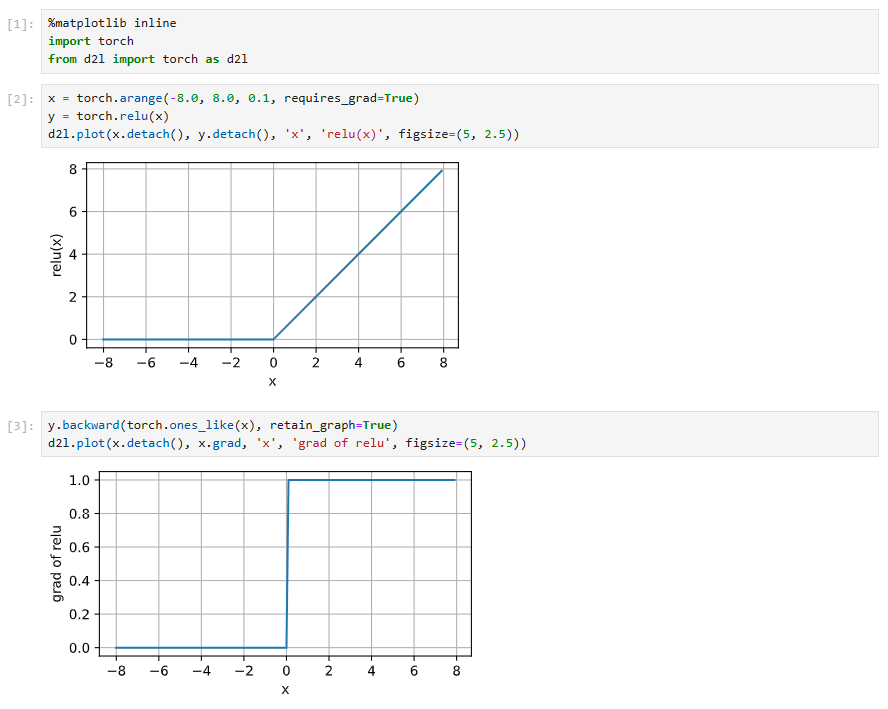
자동 생성된 설명

선형회귀 분류와 마찬가지로 시각화 할 수 있다.

**5.1. Multilayer Perceptrons**

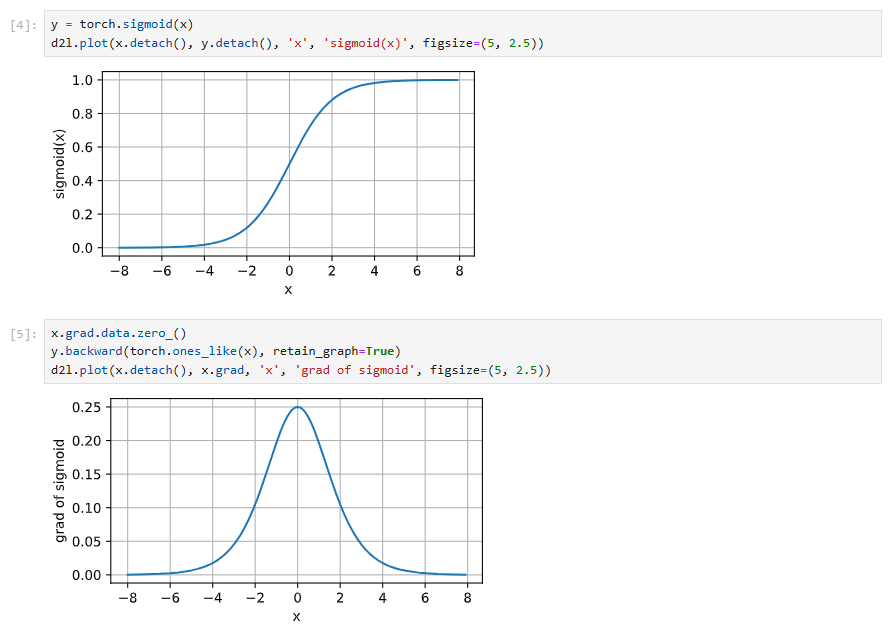
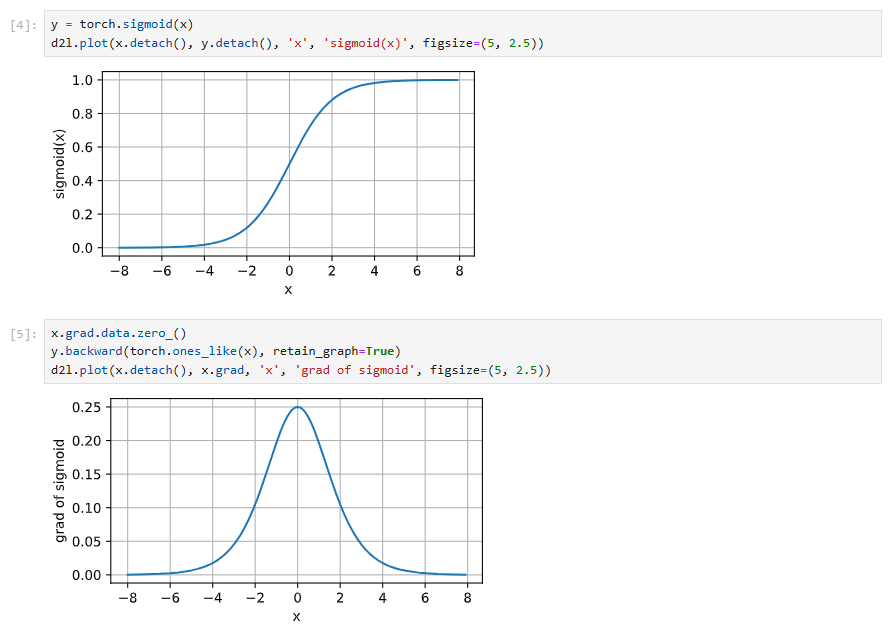
선형 모델은 입력의 증가가 항상 출력의 증가/감소로 이어진다는 한계를 가진다. 이 한계를 극복하기 위해 숨겨진 레이어를 사용하는 multilayer perceptrons을 활용할 수 있다.

Activate function은 뉴런이 활성화되어야 할지 아닌지를 판단하는데, 그 종류로는 ReLU, Sigmoid, Tanh function이 있다.

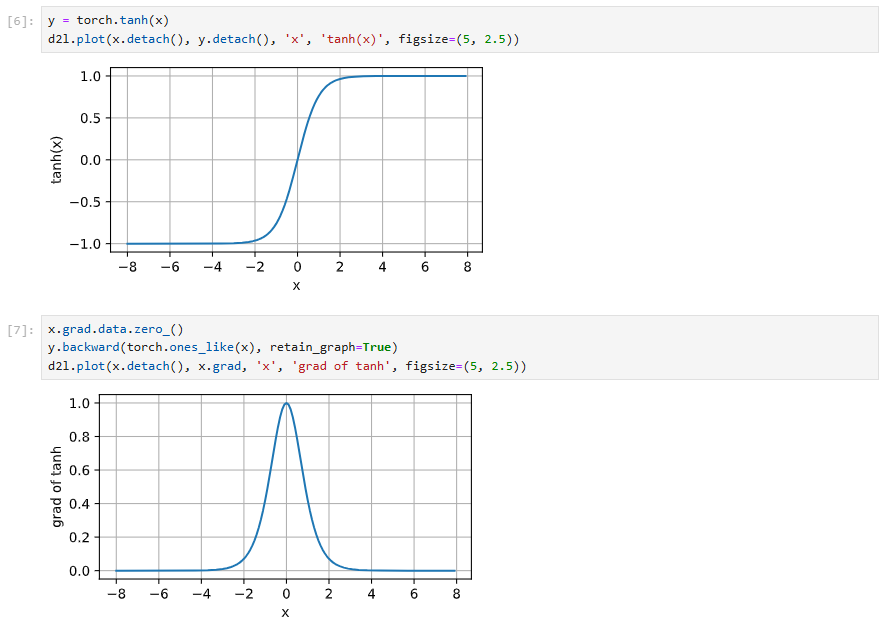


ReLU 함수는 식을 가지며, 식에서 알 수 있듯 음수값을 0으로 대체한다. 따라서 ReLU와 그 기울기에 대한 그래프는 위와 같이 그려진다.

또한, ReLU 함수는 parametrized ReLU() 등의 변형이 많다.



Sigmoid 함수는 식을 가지며, 입력값을 0과 1 사이의 출력값으로 바꾼다. 미분에 따른 기울기는 이며, 각각의 그래프는 위와 같다.



반면 Tanh 함수는 식을 가지며, -1과 1 사이의 결과를 반환한다. 기울기는 이며, 각각의 그래프는 위와 같다.

**5.2. Implementation of Multilayer Perceptron**

텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

ReLU 함수를 만들어 훈련하고, 그래프로 시각화 할 수 있다.

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

또한 high-level API를 이용해 더 간단하게 작업을 수행할 수도 있다.

**5.3. Forward Propagation, Backward Propagation, and Computational Graphs**

Forward propagation은 신경망에 대한 중간 변수를 입력 레이어부터 숨겨진 레이어, 출력 레이어까지 순서대로 계산하고 저장하는 것이다. 손실값도 순서대로 계산한다.

Backpropagation은 신경망 매개변수의 기울기를 계산하는 방법이다. 이 방법은 출력 레이어부터 입력 레이어까지 역순으로 미분과 prod 연산을 이용해 계산한다.

두 방법은 신경망을 훈련할 때 모두 사용되며 상호 의존적이다. 그 과정에서 중간 값을 저장해야 하기 때문에 예측(prediction)보다 훈련(training)에 많은 메모리가 필요하다. 이 중간 값은 레이어 수와 배치 크기에 비례한다.