간단한 신경망 만들기

- 사인 함수 예측하기 -

사인 함수 예측하기

사인 함수 예측하기

- **모듈 : nn.Module 파이토치 기본 객체**
- **MES 평균 제곱 오차 :** (Mean Squared Error, MSE)
  - **두 값의 제곱의 평균, 회귀문제에 사용**
- **CE 크로스 엔트로피 : (Cross entropy)**
  - **두 확률 분포의 차, 분류에서 사용**

예시) 스팸메일 분류, 새 아파트 가격 예측

사인 함수 예측하기

- 모듈 : nn.Module 파이토치 기본 객체
- MES 평균 제곱 오차 : (Mean Squared Error, MSE) 두 값의 제곱의 평균, 회귀문제에 사용
- CE 크로스 엔트로피 : (Cross entropy) 두 확률 분포의 차, 분류에서 사용

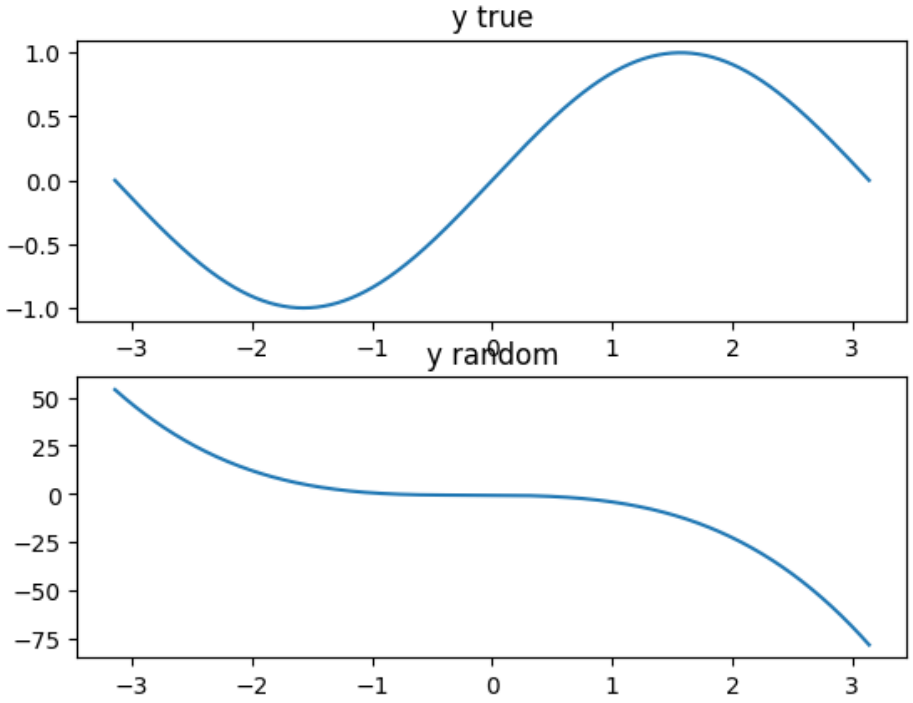
사인 함수 예측하기

```
import math
import torch
import matplotlib.pyplot as plt

x=torch.linspace(-math.pi, math.pi, 1000)
#sin 곡선 추출
y=torch.sin(x)
a=torch.randn(())
b=torch.randn(())
c=torch.randn(())
d=torch.randn(())

#3차 다항식 정의
y_random=a*x**3+b*x**2+c*x+d
```

사인 함수 예측하기




```
# 그래프 출력
plt.subplot(2,1,1)
plt.title("y true")
plt.plot(x,y)

# 예측용 y값으로 사인곡선 만들기
plt.subplot(2,1,2)
plt.title("y random")
plt.plot(x, y_random)

plt.show()
```

사인 함수 예측하기

```
learning_rate =1e-6

for epoch in range(2000):
  y_pred=a*x**3+b*x**2+c*x+d

  loss=(y_pred-y).pow(2).sum().item()
  if epoch % 100==0 :
    print(f"epoch{epoch+1} loss:{loss}")

  grad_y_pred=2.0*(y_pred-y)

  grad_a=(grad_y_pred*x**3).sum()
  grad_b=(grad_y_pred*x**2).sum()
  grad_c=(grad_y_pred*x).sum()
  grad_d=grad_y_pred.sum()
```

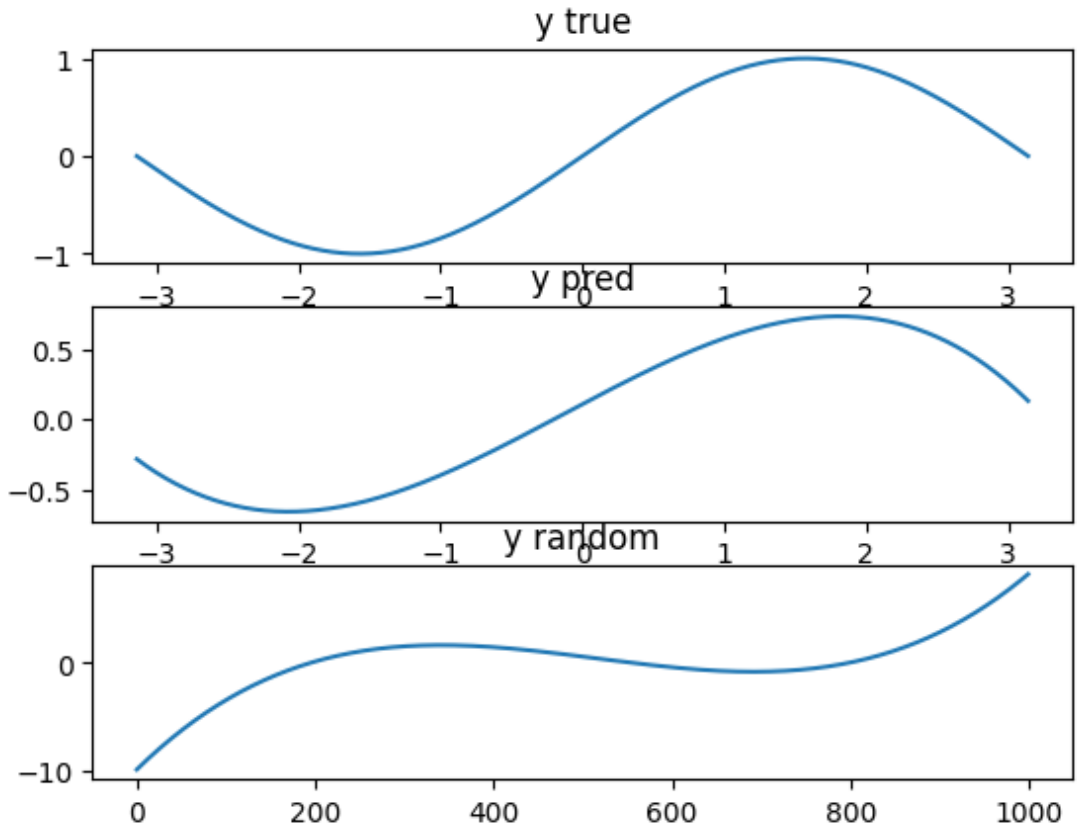
사인 함수 예측하기

```
    a-=learning_rate * grad_a
    b-=learning_rate * grad_b  # 경사하강법
    c-=learning_rate * grad_c
    d-=learning_rate * grad_d

    #그래프 그리기
plt.subplot(3,1,1)
plt.title('y true')
plt.plot(x,y)

plt.subplot(3,1,2)
plt.title('y pred')
plt.plot(x,y_pred)

plt.subplot(3,1,3)
plt.title('y random')
plt.plot(y_random)
plt.show()
```



감사합니다

참고 – Random

https://docs.scipy.org/doc/numpy-1.15.0/reference/routines.random.html

Random sampling (numpy.random)

rand(d0, d1, …, dn)	Random values in a given shape.
randn(d0, d1, …, dn)	Return a sample (or samples) from the “standard normal” distribution.
randint(low[, high, size, dtype])	Return random integers from low (inclusive) to high (exclusive).
random_integers(low[, high, size])	Random integers of type np.int between low and high, inclusive.
random_sample([size])	Return random floats in the half-open interval [0.0, 1.0).
random([size])	Return random floats in the half-open interval [0.0, 1.0).
ranf([size])	Return random floats in the half-open interval [0.0, 1.0).
sample([size])	Return random floats in the half-open interval [0.0, 1.0).
choice(a[, size, replace, p])	Generates a random sample from a given 1-D array
bytes(length)	Return random bytes.
Permutations	
shuffle(x)	Modify a sequence in-place by shuffling its contents.
permutation(x)	Randomly permute a sequence, or return a permuted range.