x

K-Digital Training 스마트 팩토리 3기

다중선형회귀

- https://colab.research.google.com/drive/1DepdQVNMmez2Z004bxrB3eETqv3WSzKd?usp=sharing

```
# optimizer 설정
optimizer = optim.SGD([w1, w2, w3, b], lr=1e-5)
```

```
nb_epochs = 1000
for epoch in range(nb_epochs + 1):

    # H(x) 계산
    hypothesis = x1_train * w1 + x2_train * w2 + x3_train * w3 + b

    # cost 계산
    cost = torch.mean((hypothesis - y_train) ** 2)

    # cost로 H(x) 개선
    optimizer.zero_grad() # 미분한 값들은 누적되는 특징이 있기 때문에 0으로 초기화
    cost.backward() # 역전파
    optimizer.step() # 역전파시에 계산된 값으로 매개변수 수정
```

다중선형회귀

- 행렬 적용
  - https://colab.research.google.com/drive/1-Tj7QmlFH5lwbVu47AJ2hBBZ4mNkQi4M?usp=sharing

```
# 모델 초기화
W = torch.zeros((3, 1), requires_grad=True)
b = torch.zeros(1, requires_grad=True)
```

```
hypothesis = x_train.matmul(W) + b
```

다중선형회귀

- Class 적용
  - https://colab.research.google.com/drive/1jAbq5Y7Bt-5gsT_Ve-_b3c-qrMRasuHI?usp=sharing

```python
# 다중 선형 회귀 클래스
class MultiLinearRegression(nn.Module):
    def __init__(self):
        super(MultiLinearRegression, self).__init__()
        self.linear = nn.Linear(3, 1)

    def forward(self, x):
        return self.linear(x)


model = MultiLinearRegression()
```

```python
# 손실 함수와 옵티마이저
loss_fn = nn.MSELoss()  # 평균 제곱 오차
optimizer = optim.SGD(model.parameters(), lr=1e-5)

# 학습
epochs = 2000
for epoch in range(epochs + 1):
    prediction = model(x_data)

    loss = loss_fn(prediction, y_data)
```