Homework 2

Modeling XOR with a shallow neural network

2021050300
김재민

1. 모델 및 학습 코드

- 다음은 Shallow Neural Network를 이용하여 XOR 모델을 구현하고 이를 학습한 코드이다.
- 이러한 코드는 SNN을 이용하여 XOR 모델링을 진행하며, Logistic Regression의 경우 XOR 모델링을 진행하며 비선형적 결과가 나왔지만, SNN을 이용하여 이를 해결하고자 한다.

1-1. XOR Data

```
x_seeds = np.array([(0,0), (1,0), (0,1), (1,1)], dtype = np.float64)
y_seeds = np.array([0,1,1,0])
N =1000
idxs = np.random.randint(0,4,N)
X = x_seeds[idxs]
Y = y_seeds[idxs]
X += np.random.normal(scale =0.25, size = X.shape)
```

1-2. Shallow Neural Network 코드

```
class shallow_neural_network():
    def __init__(self, num_input_features, num_hiddens):
        self.num_input_features = num_input_features
        self.num_hiddens = num_hiddens
        self.W1 = np.random.normal(size=(num_hiddens, num_input_features))
        self.b1 = np.random.normal(size=num_hiddens)
        self.W2 = np.random.normal(size=num_hiddens)
        self.b2 = np.random.normal(size=1)
    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))
    def predict(self, x):
        z1 = np.matmul(self.W1, x) +self.b1
        a1 = np.tanh(z1)
        z2 = np.matmul(self.W2, a1) +self.b2
        a2 =self.sigmoid(z2)
        return a2, (z1, a1, z2, a2)
model = shallow_neural_network(2,3)
```

1-3. 각 논리 연산자 코드

```
# 벡터 계산으로 변경된 train 모델
def train(X, Y, model, lr=0.1):
        # 모델 예측
        Z1 = np.dot(X, model.W1.T) + model.b1
        A1 = np.tanh(Z1)
        Z2 = np.dot(A1, model.W2) + model.b2
        A2 = model.sigmoid(Z2)

        # 손실 계산 (이진 교차 엔트로피)
        cost =-np.mean(Y * np.log(A2.flatten()) + (1 - Y) * np.log(1 - A2.flatten()))

        # 출력층 오차 계산
        diff = A2.flatten() - Y

        # 출력층 가중치 및 편향 업데이트 (벡터 연산)
        db2 = np.sum(diff)
        dW2 = np.dot(A1.T, diff)

        # 은닉층 오차 계산 및 가중치 및 편향 업데이트 (벡터 연산)
        delta1 = (1 - A1**2) * (diff.reshape(-1, 1) @ model.W2.T.reshape(1, -1))
        db1 = np.sum(delta1, axis=0)
        dW1 = np.dot(delta1.T, X)

        # 가중치 및 편향 업데이트
        model.W1 -= lr * dW1 / len(X)
        model.b1 -= lr * db1 / len(X)
        model.W2 -= lr * dW2 / len(X)
        model.b2 -= lr * db2 / len(X)

        return cost
```

2. epoch 100개를 10개 단위로 돌리기 위한 코드 및 cost 결과

2-1. 코드

```
for epoch in range(100):
    cost = train2(X,Y, model, 1.0)
    if epoch % 10 ==0:
        print(epoch, cost)
```

2-2. 결과 (epoch / cost 순서로 출력)

```
0 1.1937657565540831
10 0.6574494353530694
20 0.6314191276463123
30 0.5953509180803152
40 0.5514637944614542
50 0.520142909971489
60 0.5014468200265604
70 0.48871419689126194
80 0.4794272451702476
90 0.4723666991866248
```

3. XOR 모델의 Training

3-1. 코드

```
model.predict((0,0))[0].item()
model.predict((0,1))[0].item()
model.predict((1,0))[0].item()
model.predict((1,1))[0].item()
```

3-2. model.predict() 함수를 통한 예측값

```
(0,0) → 0.06848097090270096
(0,1) → 0.9011783965454454
(1,0) → 0.881318422217531
(1,1) → 0.06708234304255642
```

4. Data Plotting 진행

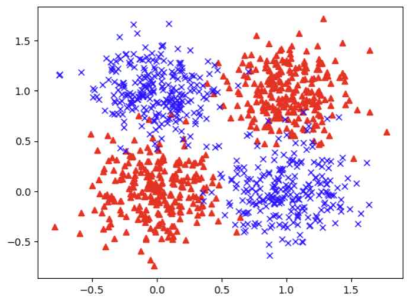
4-1. 코드

```
idxs_1 = np.where(Y==1)
idxs_0 = np.where(Y==0)

X_0 = X[idxs_0]
Y_0 = Y[idxs_0]
X_1 = X[idxs_1]
Y_1 = Y[idxs_1]

# plot
plt.plot(X_0[:,0], X_0[:,1],"r^")
plt.plot(X_1[:,0], X_1[:,1],"bx")
plt.show()
```

4-2. 결과



5. 결론

- XOR은 SNN을 통해 성공적으로 학습되었음을 test결과와 matplot을 통해 알 수 있다.
- 이러한 코드는 SNN을 이용하여 XOR 모델링을 진행하며, Logistic Regression의 경우 XOR 모델링을 진행하며 비선형적 결과가 나왔지만, SNN을 이용하여 이를 해결하였음을 보여준다.