REPORT

Modeling XOR with a shallow neural network Assignment

과목명	딥러닝
담당교수	정우환 교수님
학생이름	박준우
학과	인공지능학과
학번	2021006253
제출일	2023.09.24

HANYANG UNIVERSITY

Source code for model

```
class shallow_neural_network():                                          // SNN 클래스 선언
    def __init__(self, num_input_features, num_hiddens):
        self.num_input_features=num_input_features                       //입력 특성 개수
        self.num_hiddens=num_hiddens                                     //은닉층 뉴런 개수

        self.W1=np.random.normal(size=(num_hiddens, num_input_features)) //가중치 행렬 W1 정규분포 초기화
        self.b1=np.random.normal(size=num_hiddens)                       //은닉층 편향 벡터 b2 초기화
        self.W2=np.random.normal(size=(1,num_hiddens))                   //가중치 행렬 W2 정규분포 초기화
        self.b2=np.random.normal(size=1)                                 //출력층 편향 벡터 b2 초기화

    def sigmoid(self,z):                                                 //시그모이드(활성화) 함수 정의
        return 1/(1+np.exp(-z))

    def predict(self,x):                                                 // predict 함수 선언
        z1=np.matmul(self.W1,x)+self.b1                                  //가중 합 계산 및 편향 값 덧셈
        a1=np.tanh(z1)                                                   //tanh 활성화 함수 적용해 a1 계산
        z2=np.matmul(self.W2,a1)+self.b2                                 //가중 합 z2 계산 + 출력층 편향값
        a2=self.sigmoid(z2)                                              //시그모이드 적용 a2 계싼
        return a2, (z1,a1,z2,a2)                                         //a2 와 중간값들 튜플로 반환
```

Source code for training

```
def train (X,Y, model,lr=0.1):                                  //학습 함수 정의
    dW1=np.zeros_like(model.W1)                                 //가중치, 편향 미분값 초기화
    db1=np.zeros_like(model.b1)
    dW2=np.zeros_like(model.W2)
    db2=np.zeros_like(model.b2)
    m=len(X)                                                    //입력 데이터 x 샘플 수
    cost=0.0                                                    //오차 초기화
    for x,y in zip(X,Y):                                        //훈련데이터 x,y 에 대한 반복문
        a2,(z1,a1,z2,_)=model.predict(x)                        //입력 x 에 대한 예측과 중간 값
        if y==1:                                                //loss func (Binary Cross Entropy)
            cost-=np.log(a2)
        else:
            cost-=np.log(1-a2)

        diff=a2-y                                               //a2 와 실제값 y 의 차이

        dW2+=np.outer(diff,a1)                                  //출력 가중치 W2 기울기 업뎃
        db2+=diff                                               //출력층 편향 값 b2 업뎃

        dz1=np.multiply((1-np.square(a1)),np.dot(model.W2.T,diff))  //은닉층 z1 기울기 계산
        dW1+=np.outer(dz1,x)                                    //은닉 가중치 W1 기울기 업뎃
        db1+=dz1                                                //은닉 편향값 b1 기울기 업뎃

    cost/=m                                                     //평균 cost 값
    model.W1-=lr*dW1/m
    model.b1-=lr*db1/m
    model.W2-=lr*dW2/m
    model.b2-=lr*db2/m

    return cost
```

Plotshow for XOR Operator and Predicted results

In [71]:
```
for epoch in range(100):
    cost=train(X,Y,model,1.0)
    if epoch %10==0:
        print(epoch,cost)
```

```
0 [0.78200101]
10 [0.63262951]
20 [0.57732848]
30 [0.47444905]
40 [0.38416924]
50 [0.31599153]
60 [0.26860725]
70 [0.23604187]
80 [0.21321833]
90 [0.19675363]
```

In [72]:
```
model.predict((1,1))[0].item()
```

Out[72]: 0.06111210618140697

In [73]:
```
model.predict((1,0))[0].item()
```

Out[73]: 0.9059690199254549

In [74]:
```
model.predict((0,1))[0].item()
```

Out[74]: 0.9787933212388599

In [75]:
```
model.predict((0,0))[0].item()
```

Out[75]: 0.0559807158862618

In [76]:
```
idxs_1=np.where(Y==1)
idxs_0=np.where(Y==0)
```

In [77]:
```
X_0=X[idxs_0]
Y_0=Y[idxs_0]
```

In [78]:
```
X_1=X[idxs_1]
Y_1=Y[idxs_1]
```

In [80]:
```
#plt.clf()
plt.plot(X_0[:,0],X_0[:,1],"r^")
plt.plot(X_1[:,0],X_1[:,1],"bx")
plt.show()
```