Seoul National University
College of Engineering
Department of Naval Architecture and Ocean Engineering
1, Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea

Fall 2020

딥러닝

PA # 3

Instructor name	김태완 교수님
Student name	이용준
Department	조선해양공학과
Student ID	2015-19595
Submission date	2020.10.26

Contents

1. Problem Definition

1.1 Problem 1

Types	Columns	Explanations
INPUT	CRIM	자치시(town) 별 1 인당 범죄율
	ZN	25,000 평방피트를 초과하는 거주지역의 비율
	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
	NOX	10ppm 당 농축 일산화질소
	RM	주택 1가구당 평균 방의 개수
	AGE	1940년 이전에 건축된 소유주택의 비율
	DIS	5개의 보스턴 직업센터까지의 접근성 지수
	RAD	방사형 도로까지의 접근성 지수
	TAX	10,000 달러 당 재산세율
	PTRATIO	자치시(town)별 학생/교사 비율
	B-1000	1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
	LSTAT	모집단의 하위계층의 비율(%)
OUTPUT	MEDV	본인 소유의 주택가격(중앙값) (단위: $1,000)

- 주어진 데이터에 적합한 Neural network를 코드로 작성하고 학습시키기
- Hidden layer의 수와 node의 수, activation function 등을 변경해가며 학습 결과 비교하기
- cost function으로 SSE를 사용하고 training, validation, test의 비율 8 : 1: 1
- input data와 output data를 같은 scale로 정규화 했을 때와 하지 않았을 때를 비교
- input data 중 결과에 가장 큰 영향을 미치는 요소들을 선정하고, 그 데이터만을 사용하여 neural network를 학습시키고 결과 비교하기

1.2 Problem 2

Types	Columns	Explanations
INPUT	age	나이
	sex	성별(남=1 여=0)
	cp	가슴 통증 타입(0~3)
	trestbps	휴식기 혈압
	chol	형청 콜레스테롤(mg/dl)
	fbs	공복시 혈당
	restecg	휴식기 심전도
	thalach	최대 심장박동 수
	ca	주요 혈관의 수
OUTPUT	target	Heart disease의 발병 여부

- 주어진 데이터에 적합한 Neural network를 코드로 작성하고 학습시키기

- Hidden layer의 수와 node의 수, activation function 등을 변경해가며 학습 결과 비교하기
- output activation은 sigmoid 혹은 softmax
- cost function으로 SSE 또는 CFCF를 사용하고 training, validation, test의 비율 8 : 1: 1
- input data 중 결과에 가장 큰 영향을 미치는 요소들을 선정하고, 그 데이터만을 사용하여 neural network를 학습시키고 결과 비교하기
- input data와 output data를 같은 scale로 정규화 했을 때와 하지 않았을 때를 비교

1.3 Problem 3

- mnist_400 data에 적합한 Neural network 코드 작성하고 학습시키기
- Hidden layer 개수 2개, activation function으로 sigmoid 혹은 ReLU 사용
- output activation function으로 softmax, cost function으로 CECF 사용
- training, validation, test 비율 2: 1: 1
- 가중치 초기화 방법을 적용해보고 결과 비교
- 배치 정규화 방식을 Hidden layer에 적용해보고 결과 비교
- 1-D 데이터의 한계점 논의

2. Problem Analysis

- Data의 정규화

```
def normalized(x):## x is array
    normalized_x = (x - np.min(x)) / (np.max(x)-np.min(x))
    return normalized_x
```

이번 과제를 수행하기 위해선 input data와 output data의 정규화 과정이 필요하다. 각 data별로 scale이 다르기 때문에 모든 data의 값들을 0부터 1까지의 값으로 조정한다. 즉, 최솟값을 0으로, 최댓값을 1로 모든 데이터 값을 조정한다.

- Problem 1

주택가격을 예측해내는 것이 목표이기 때문에 output layer의 node 개수는 1개로 설정한다. 전체 데이터 개수를 기준으로 training, validation, test data의 개수는 404, 50, 50으로 설정하였다.

- Problem 2

heart disease의 발병 여부를 예측해야 하기 때문에 output layer의 node 개수를 2개로 설정한다. Output layer의 activation function으로 sigmoid를 설정하여 0과1 사이의 값을 가지도록 한다. 이때 첫번째 노드의 값이 더 크면 발병 여부를 yes, 두번째 노드 값이 더 크면 발병 여부를 no로 산출하기로 정한다. 전체 데이터 개수를 기준으로 training, validation, test data의 개수는 240, 30, 30로 설정하였다. Cost function으로는 SSE를 사용하였다.

- Problem 3

0부터 9까지의 숫자를 예측해야 하기 때문에 output layer의 node 개수를 10개로 설정하였다. 10개의 값 중 가장 큰 값을 가지는 node의 인덱스 값을 예측한 숫자의 결과로 사용한다. 예를 들어, 10개의 node 값 중 5번째 값이 가장 크다면, 숫자를 4로 예측한다. 10개의 node 값 중 1번째 값이 가장 크다면, 숫자를 0으로 예측한다. Hidden layer의 개수는 2개이며, Hidden layer에 사용한 activation function은 sigmoid를 사용하였다. 따라서 가중치 초기화 방법은 Xavier initialization을 채택하였다.

3. Code Explanation

프로젝트에 사용된 함수에 관한 설명이다. 문제 별 사용한 함수는 동일하다.

sigmoid(x)		
Parameter	Type	Explanation
x	array	Sigmoid에 대입할 input array

identity_function(x)		
Parameter	Type	Explanation
x	array	Identity function에 대입할 input array

normalized(x)		
Parameter	Type	Explanation
x	array	정규화 할 input array

setting_data(data)		
Parameter	Type	Explanation
data	dataframe	csv파일의 내용을 담고 있는 dataframe

Hiddenlayer1

init_network()		
Parameter	Type	Explanation
-	-	Network의 초기값을 설정
training(network, x, t)		
Parameter	Type	Explanation
Network	dictionary	Weight, bias 값을 담고 있는 dictionary
X	array	Input layer의 값들을 저장하는 array
T	array	Target 값들을 저장하는 array
test(network, x, t)		
Parameter	Type	Explanation
Network	dictionary	Weight, bias 값을 담고있는 dictionary
X	array	Input layer의 값들을 저장하는 array
T	array	Target 값들을 저장하는 array

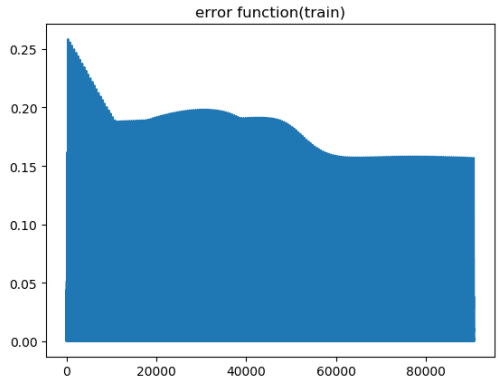
Hiddenlayer2		
init_network()		
Parameter	Type	Explanation
-	-	Network의 초기값을 설정
training(network, x, t)		
Parameter	Type	Explanation
Network	dictionary	Weight, bias 값을 담고있는 dictionary
X	array	Input layer의 값들을 저장하는 array
T	array	Target 값들을 저장하는 array
test(network, x, t)		
Parameter	Type	Explanation
Network	dictionary	Weight, bias 값을 담고있는 dictionary
X	array	Input layer의 값들을 저장하는 array
T	array	Target 값들을 저장하는 array

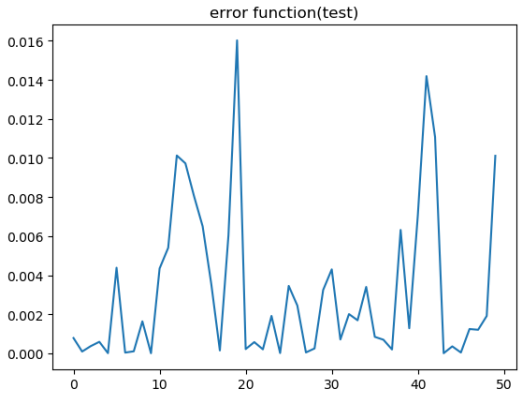
4. Result & Conclusion

4.1 Result

<Problem 1>

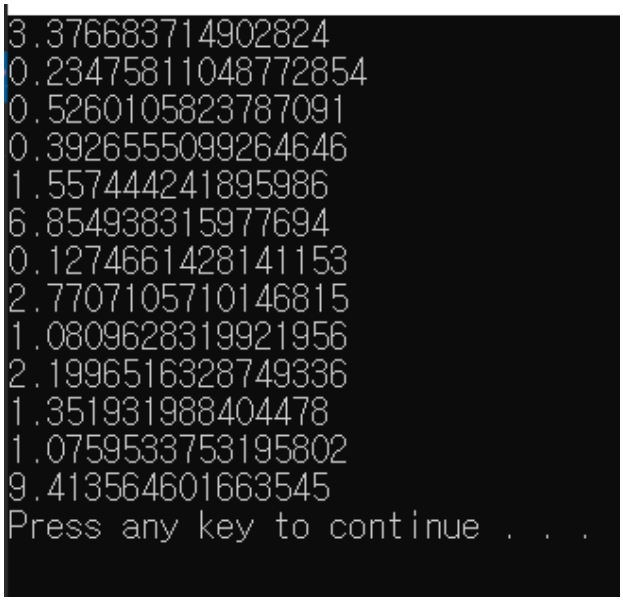
(a) 문제 1-(a)에 대한 결과로, 우선 output layer의 activation function을 sigmoid함수를 쓴 경우와 identity함수를 쓴 경우를 비교해보았다. Error function을 비교해본 결과 identity를 사용했을 경우가 학습이 더 잘 되었다. 그리고 다음으로 hidden layer가 1개인 경우와 hidden layer가 2개인 경우를 비교해 보았다. hidden layer 1개인 경우 node 수를 20개, hidden layer 2개인 경우 1층 노드 개수 10개, 2층의 노드 개수 5개로 설정하였다. Cost function은 SSE를 적용하였다. 아래의 그래프는 epoch가 각각 200일 때의 error function(왼쪽은 training, 오른쪽은 test)을 그린 것이다.



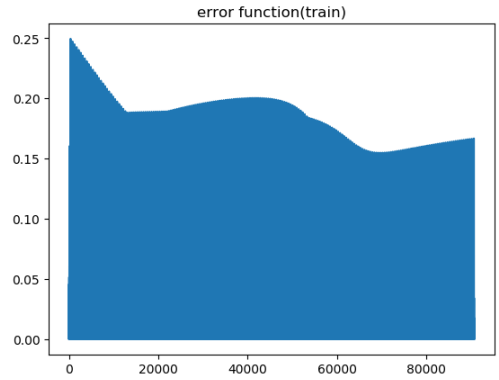


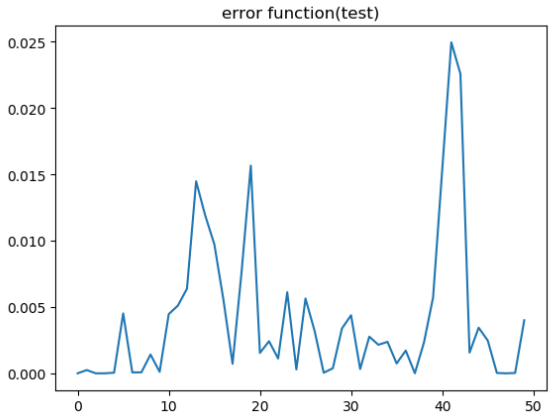
결과적으로 activation function이 identity, hidden layer 2층인 경우가 학습 결과가 가장 좋았다. 위의 error function 그래프를 보면 학습이 진행되면서 점차 오류 값이 감소하는 것을 확인할 수 있었다. 그리고 test 그래프를 보면 오차가 매우 작게 발생함을 확인할 수 있었다.

(b) training 완료 후 i 번째 input node에서 뻗어 나가는 weight_i들의 크기들의 합을 구해 비교를 해봤더니 가장 높은 값을 가진 input data는 LSTAT(모집단 하위계층 비율), RM(평균 방 개수), CRIM(1인당 범죄율)이었다. 이들을 주택가격에 가장 큰 영향을 주는 데이터들이라고 가정을 한 뒤, 이 세개의 데이터만으로 다시 학습을 시켜보았다.



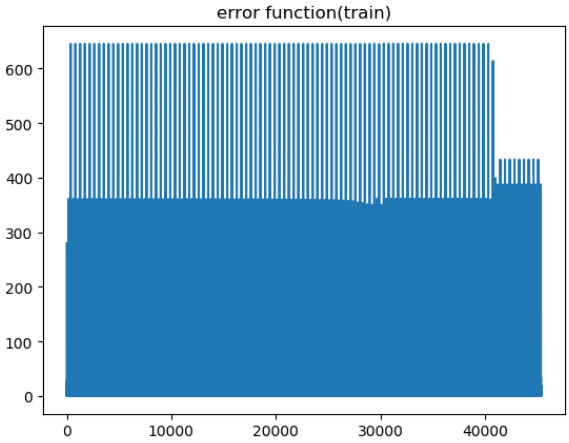
Epoch = 200으로 학습시키면서 training시의 error function, test시의 error function을 다시 구해보았다.

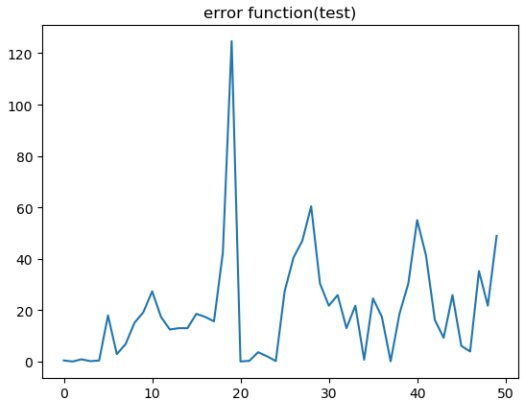




이전 그래프와 비교해보았을 때 학습성과에 별 차이가 없었고, test error function의 경우 error값이 약간 증가했다.

(c) input data와 output data를 같은 scale로 정규화 하지 않았을 경우는 다음과 같았다.





Normalize를 하지 않을 경우 학습이 전혀 되지 않는 결과를 낳았다. Error function이 수렴하지 않고 계속 진동하였다. Test의 error값 역시 매우 크게 나타났다.

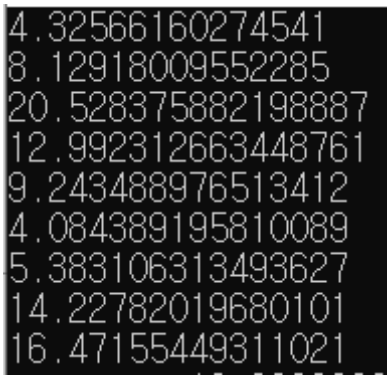
<Problem 2>

(a) hidden layer가 1개인 경우와 hidden layer가 2개인 경우를 비교해 보았다. hidden layer 1개인 경우 node 수를 12개, hidden layer 2개인 경우 1층 노드 개수 5개, 2층의 노드 개수 5개로 설정하였다. Output activation function은 sigmoid를 사용하여 결과를 0~1로 mapping하였다. Cost function으로 CECF를 사용하였다. Epoch =200으로 학습하였다.

Epoch= 200	Hidden layer 1개	Hidden layer 2개
Error rate	10%	13.33%

Output activation function을 sigmoid로 설정하였을 때, 동시에 hidden layer가 1층일때가 미미하지만 학습효과가 더 높았음을 알 수 있었다.

(b) training 완료 후 i 번째 input node에서 뻗어 나가는 weight_i들의 크기들의 합을 구해 비교를 해봤더니 가장 높은 값을 가진 input data는 cp, ca, thalach, trestbps였다. 이들을 herat disease에 가장 큰 영향을 주는 데이터들이라고 가정을 한 뒤, 이 네 개의 데이터만으로 다시 학습을 시켜보았다.



그 결과는 다음과 같다. Hidden layer는 2층으로 설정하였다.

Epoch = 200	All data	cp, ca, thalach, trestbps
Error rate	13.33%	16.66%

학습결과 네 개의 데이터만으로도 비슷한 결과를 얻을 수 있었다. 그러다 정확도는 약간 떨어졌다.

(c) input data와 output data를 같은 scale로 정규화 하였을 경우와 하지 않았을 경우를 비교한 표는 다음과 같았다.

Epoch = 200	Normalized	Non-normalized
Error rate	13.33%	13.33%

정규화 이전의 결과와 같은 결과를 얻었다. 기본 정답률이 50%이고, Test data의 개수가 30개로 매우 작아서 이와 같은 결과가 나올 수 있었다고 예상된다.

<Problem 3>

(a)

hidden layer의 수는 2개, activation function은 sigmoid, output activation function은 softmax를 사용했다. Cost function은 CECF를 사용하였다. Training, validation, test 개수는 각각 200, 100, 100 (2: 1: 1)으로

설정하였다. Epoch =500으로 학습시킨 결과는 다음과 같았다.

Epoch = 500	Activation function : sigmoid Output activation function : softmax Cost function : CECF
Error rate	76%

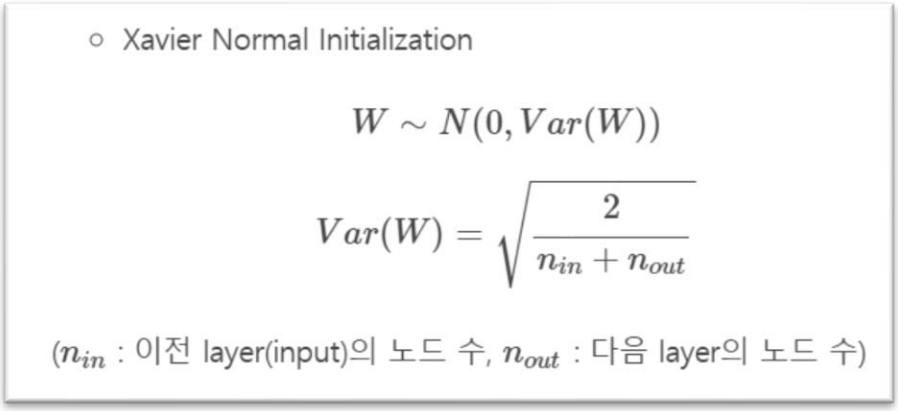
학습성과가 보이긴 하였지만 높은 에러율을 보이는 것을 확인하였다. 기존 90프로에서 76프로로 에러율을 줄였다.

(b)

원래의 데이터는 두개의 축을 가진 2차원 데이터이다. 그렇기 때문에 위아래로 연결되어 있는 픽셀들 사이에는 색데이터의 유사성이 존재한다. 하지만 이 데이터를 1차원으로 변환시키면 이러한 특성을 반영하기 어려워진다.

(c)

다음으로 Xavier initialization으로 가중치를 초기화한 뒤 같은 방법으로 학습시킨 결과이다.




```
network = {}
network['w1'] = np.random.normal(0,np.sqrt(2/(784+100)),[784,100]) ##xavier initialization
##network['w1'] = np.ones((784,100))*0.05
network['b1'] = np.ones((1,100))*0.2
network['w2'] = np.random.normal(0,np.sqrt(2/(100+30)),[100,30]) ##xavier initialization
##network['w2'] = np.ones((100,30))*0.1
network['b2'] = np.ones((1,30))*0.2
network['w3'] = np.random.normal(0,np.sqrt(2/(30+10)),[30,10]) ##xavier initialization
##network['w3'] = np.ones((30,10))*0.1
network['b3'] = np.ones((1,10))*0.2
return network
```

Epoch = 500	Xavier initialization
Error rate	17%

학습 결과, Xavier initialization을 적용하지 않았을 때에 비해서 학습 효과가 매우 상승했음을 보였다. Error rate를 크게 줄일 수 있었다. 100번의 test 중 83번을 예측에 성공했다.

(d)

4.2 Conclusion

- Hidden layer의 개수, node 수, activation function을 바꿔가면서 다양한 방법을 학습하고 이를 비교해 볼 수 있었다.
- 데이터를 정규화 하지 않았을 때에 비하여 데이터를 정규화해주면 학습 효과를 높일 수 있었다.
- Xavier initialization방법을 통해 가중치 초기화를 해보니 학습 효과를 크게 높일 수 있었다.