

학습계획서

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제	주요내용
1일차 (5 / 27)	임연지	파이썬 기초	파이썬 기본 문법 공부
2일차 (5 / 28)	박길준	딥러닝 기초	머신러닝의 개념과 용어 Linear Regression 의 개념 Linear Regression cost함수 최소화 여러개의 입력(feature)의 Linear Regression
3일차 (5 / 29)	유동관	Logistic (Regression) Classification	Hypothesis 함수 소개 Cost 함수 소개
4일차 (5 / 30)	임연지	Softmax Regression (Multinomial Logistic Regression)	Multinomial 개념 소개 Cost 함수 소개
5일차 (5 / 31)	박길준	ML의 실용과 몇가지 팁	학습 rate, Overfitting, 그리고 일반화 (Regularization) Training/Testing 데이터 셋
6일차 (6 / 3)	유동관	딥러닝의 기본 개념과, 문제, 그리고 해결	딥러닝의 기본 개념: 시작과 XOR 문제 딥러닝의 기본 개념2: Back-propagation 과 2006/2007 '딥'의 출현
7일차 (6 / 4)	임연지	Neural Network 1: XOR 문제와 학습방법, Backpropagation	XOR 문제 딥러닝으로 풀기 특별편: 10분안에 미분 정리하기 딥넷트웍 학습 시키기 (backpropagation)
8일차 (6 / 5)	박길준	Neural Network 2: ReLU and 초기값 정하기	XSigmoid 보다 ReLU가 더 좋아 Weight 초기화 잘해보자 Dropout 과 앙상블 레고처럼 넷트웍 모듈을 마음껏 쌓아 보자 딥러닝으로 MNIST 98%이상 해보기
9일차 (6 / 7)	유동관	Convolutional Neural Networks	ConvNet의 Conv 레이어 만들기 ConvNet Max pooling 과 Full Network ConvNet의 활용예
10일차	임연지	Recurrent	NN의 꽃 RNN 이야기

(6 / 10)		Neural Network	
------------	--	----------------	--

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
1일차 (5 / 27)	임연지	파이썬 기초

주요 내용 요약

- 출력하기
print()
- 입력하기
input()
- Data type
숫자형(Numeric), 문자열(String), 불린(Boolean)
- list
가변과 불변으로 나뉘어짐
여러 값을 한꺼번에 모을 수 있음
가변) 값을 변경할 수 있는 리스트, 딕셔너리
불변) 값을 변경할 수 없는 문자열, 튜플
>>> my_list = [] 처럼 구현
append함수로 데이터 추가 []안의 숫자로 인덱싱 가능
sort() 같은 함수로 정렬이 가능
- tuple
리스트와 거의 유사하지만 값을 변경할 수 없음
괄호를 쓰지 않아도 됨
>>> my_tuple = (1, 2, 3)
패킹) 여러 개의 값을 한꺼번에 묶는 것
언패킹) 묶여 있는 값을 풀어놓는 것
- dictionary
관련된 정보를 서로 연관시켜 놓은 것
키와 값의 쌍으로 이루어져 있습니다.
- casting
자료형끼리 변환할 수 있는 함수도 존재.
내장함수란 파이썬에서 기본으로 제공하는 함수를 의미.
int(): 정수형으로 변환.. 같은 방법으로
float(), str(), list()
- comment
주석, #을 사용
- module
비슷한 기능의 함수들을 모아둔 파일
직접 만들 수도 있고 가져와서 사용할 수도 있음
import 키워드로 모듈을 가져오고, 마침표(.)를 이용해 함수를 사용

```

ex) import 모듈이름
10. for
    기본 구조)
    for 변수 in 컨테이너:
        실행할 명령1
        실행할 명령2
    range() : for와 함께 자주 사용되는 내장함수
    range(stop)은 0부터 stop전까지의 숫자를 나열
    range(start, stop)은 start부터 stop전까지의 숫자를 나열
11. 논리연산자, 멤버쉽연산자
    파이썬의 논리연산자) and, or, not
    and : 둘다 true
    or : 하나라도 true
    not : true->false, false->true
    파이썬의 멤버쉽연산자) in, not in
    in은 포함되어있다, not in은 포함되어있지 않다
    결과는 true와 false로 출력
12. 함수 선언
    def 함수이름 (인자1, ...):
        실행할 명령1
        실행할 명령2
        ....
        return 결과

```

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
2일차 (5 / 28)	박길준	딥러닝 기초
주요 내용 요약		
1. 머신러닝, 인공지능, 딥러닝 <ul style="list-style-type: none"> a. 기계가 학습을 할 수 있도록 하는 연구 분야 b. 인공지능 연구의 한 분야로서 최근 들어 딥러닝을 통해서 빠르게 발전 2. 머신러닝의 특성 <ul style="list-style-type: none"> a. 학습 시스템과 머신러닝 		

- i. 학습 시스템 : 환경과의 상호작용으로부터 획득한 경험적인 데이터를 바탕으로 지식을 자동으로 습득하여 스스로 성능을 향상하는 시스템
 - ii. 머신러닝 : 인공적인 학습 시스템을 연구하는 과학과 기술, 즉, 경험적인 데이터를 바탕으로 지식을 자동으로 습득하여 스스로 성능을 향상하는 기술
- b. 딥러닝
 - i. 많은 수의 신경층을 쌓아 입력된 데이터가 여러 단계의 특정 추출 과정을 거쳐 자동으로 고수준의 추상적인 지식을 추출하는 방식
 - ii. 특징 추출과 특징 분류를 특징 학습의 문제로 통합
 - iii. 딥러닝을 통해서 신경망에 대한 관심이 다시 늘어남에 따라 머신러닝 연구에 관심을 다시 갖기 시작함.
- c. 다양한 분야에의 응용
 - i. Google의 GoogLeNet, 백만여 장의 이미지로부터 천 가지 종류의 물체를 분류
 - ii. Facebook의 DeepFace, 사람의 얼굴을 인식하는 문제에서 인간 수준의 성능
 - iii. Microsoft의 딥러닝을 적용한 음성인식 기술
 - iv. Google의 DeepMind, 실제 사람처럼 비디오 게임을 학습하는 기술
- 3. 프로그래밍 방식과의 차이점
 - a. 일반적인 컴퓨터 프로그램
 - i. 사람이 알고리즘 설계 및 코딩
 - ii. 주어진 문제에 대한 답 출력
 - b. 머신러닝 프로그램
 - i. 사람이 코딩
 - ii. 기계가 알고리즘을 자동 프로그래밍
 - iii. 데이터에 대한 프로그램을 출력
 - c. 머신러닝이 필요한 문제
 - i. 프로그래밍이 어려운 문제(예: 음성인식)
 - ii. 지속적으로 변화하는 문제(예: 자율이동로봇)
 - d. 머신러닝이 더욱 중요해지는 이유
 - i. 빅데이터의 존재
 - ii. 컴퓨터 성능의 향상(고난도 학습이 가능)
 - iii. 서비스와 직접 연결(비즈니스적 효과)
 - iv. 비즈니스 가치 창출(회사 가치 향상)
- 4. 산업적 응용 사례 - 자율주행 자동차
 - a. 1992년, 신경망 구조를 이용한 자동차 자동 운전
 - b. 2005년, 사막 환경 무인 자동차 대회에서 성공적으로 운전을 수행
 - c. 2007년, 도시 환경 무인 자동차 대회에서 성공적으로 운전을 수행
 - d. 2010년, 구글이 무인자동차를 발표
- 5. 다양한 활용 분야
 - a. 데이터 마이닝
 - b. 자연어처리 음성인식, 기계번역, 챗봇
 - c. 컴퓨터비전, 물체인식
 - d. 로보틱스, 휴머노이드 로봇
 - e. 컴퓨터그래픽스
- 6. 최근 산업 동향
 - a. 머신러닝에 대한 기업들의 투자 증가
 - i. 글로벌 IT 기업들의 인공지능 연구소 설립 및 스타트업 인수
 - b. 스피커 타입 스마트 비서
 - i. 홈 서비스를 위한 보급형 스피커 타입 스마트 비서의 등장

- ii. 배송 주문, 음악 추천 등 실생활 응용과 긴밀히 연결
- c. 산업 전바에서 사물인터넷과 빅데이터를 기반으로 하는 제 4차 산업혁명의 물결이 일어나고 있고 머신러닝 기반의 인공지능은 그 동력원 역할을 할 것으로 예측된다.

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

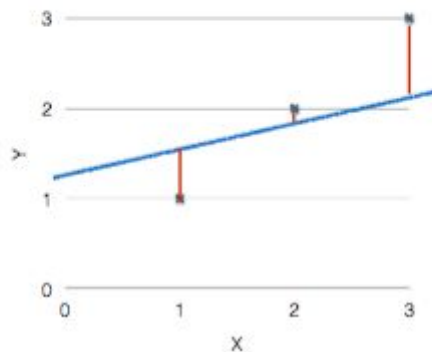
일정	발제자	주제
3일차 (5 / 29)	유동관	Logistic (Regression) Classification Hypothesis 함수 소개 Cost 함수 소개 TensorFlow 에서의 구현

주요 내용 요약

1. Linear Regression
 - a. 회귀분석
 - i. 점들이 퍼져있는 상태에서 패턴을 찾아내고, 이 패턴을 활용해서 무언가를 예측하는 분석. 새로운 표본을 뽑았을 때 평균으로 돌아가려는 특징이 있기 때문에 붙은 이름이다.
 - b. Linear Regression
 - i. 2차원 좌표에 분포된 데이터를 1차원 직선 방정식을 통해 표현되지 않은 데이터를 예측하기 위한 분석 모델. 머신러닝 입문에서는 2차원이나 3차원까지만 정리하면 된다. 간단하게 직선을 생각하면 된다.
2. Hypothesis
 - a. Linear Regression에서 사용하는 1차원 방정식을 가리키는 용어로, 우리말로는 가설이라고 한다. 수식에서는 $h(x)$ 또는 $H(x)$ 로 표현한다.
 - b. $H(x) = Wx + b$ 에서 $Wx + b$ 는 x 에 대한 1차 방정식으로 기울기에 해당하는 W (Weight)와 절편에 해당하는 b (bias)가 반복되는 과정에서 계속 바뀌고, 마지막 루프에서 바뀐 최종 값을 사용해서 데이터 예측(prediction)에 사용하게 된다. 최종 결과로 나온 가설을 모델(model)이라고 부르고, '학습되었다'라고 한다. 학습된 모델은 배포되어 새로운 학습을 통해 수정되기 전까지 지속적으로 활용된다.
3. Cost(비용)
 - a. Hypothesis 방정식에 대한 비용으로 방정식의 결과가 크게 나오면 좋지 않다고 이야기하고, 루프를 돌 때마다 W 와 b 를 비용이 적게 발생하는 방향으로 수정하게 된다.
 - b. 미분을 이용하면 스스로 최저비용을 찾아갈 수 있다고 하는데, 강의자도 신기하다고 하였음
4. Cost 함수
 - a. Hypothesis 방정식을 포함하는 계산식으로, 현재의 기울기(W)와 절편(b)에 대해 비용을 계산해 주는 함수이다. 매번 호출시 반환값으로 표현되는

비용이 줄어들도록 코딩되어야 한다.

Which hypothesis is better?



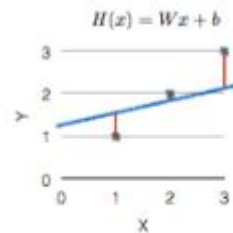
- b.
- c. 위의 그림처럼 한 직선에 대해 각 좌표까지의 일직선 거리를 cost라고 정의하고, 이 cost 길이가 가장 짧도록 직선을 긋는 것이 최소비용을 구하는 방법이다.
- d. 계산법

Cost function

- How fit the line to our (training) data

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m} \sum_{i=1}^m (H(x^{(i)}) - y^{(i)})^2$$



- i.
- ii. hypothesis의 결과에서 y 를 뺀 다음에 제공해주어야 한다. 통계의 표준편차와 분산의 원리를 이용한 것이다.
- iii. 크기를 계산하기 위해 절대값을 계산하는 것이 쉽지만, 제곱을 하는것도 방법이라고 한다(항상 양수가 나오기 때문에).
- iv. 제곱을 하면 가장 가까운 데이터는 작은 값이 나오고, 멀리 있는 데이터는 큰 값이 나오기 때문에 멀리 있는 데이터에 벌점(penalty)를 부과할 수 있다. 이 방법을 LSM(Least Square Method) 라고 한다. 통계학 관련 서적에서 보면, 절대값을 이용한 처리보다 튼튼하다고 알려져 있다.
- v. 여기서는 계산 후 점의 갯수대로 나누어준다.

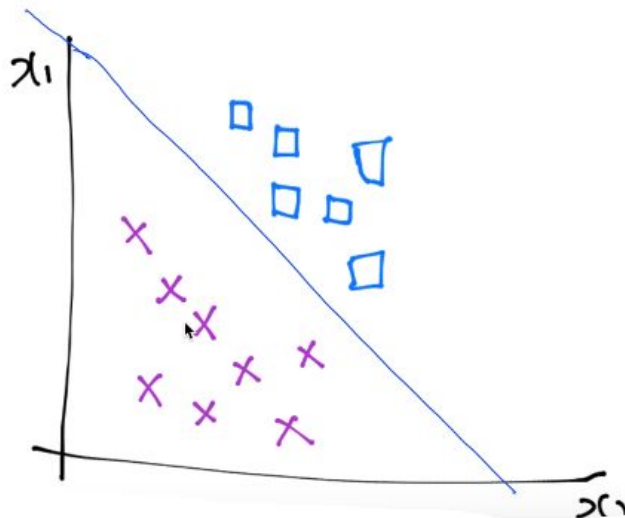
학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
4일차 (5 / 30)	임연지	<Softmax Regression> Multinomial 개념 소개 Cost 함수 소개

주요 내용 요약

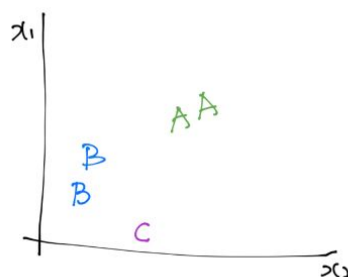
- Logistic regression에서 둘 중 하나를 고르는데에 적합하지않았음
Z가 아주 커지거나 작아져도 $g(z)$ 가 0이나 1 사이의 값으로 나왔으면 좋겠다라는 생각에서 발전되어 알아낸 것이 $g(z) = 1/(1+e^{-z})$ (logistic classification)
 $H(x) = g(H(x))$ 가 최종식
- X라는 입력이 들어올때 W를 가지고 유닛이 계산한 후 나오는 Z값을 sigmoid라는 함수에 한 번 통과시키면 그 수는 0과 1 사이의 어떤 값이 될 것이다(\hat{Y} 으로 표현, Y위에 -쓴 문자)
- Logistic regression을 학습한다 :



위와 같은 식의 데이터가 있다면 X와 네모를 분리하는 선을 찾는것을 학습하는 것

- Multinomial Classification
: 클래스가 여러개 있는것

x1 (hours)	x2 (attendance)	y (grade)
10	5	A
9	5	A
3	2	B
2	4	B
11	1	C



이런 경우에서 B와 C사이에 선이있으면 C or not C, B바로옆에 또 선이 있으면 B or not B 처럼 표현할 수 있다.

이를 토대로 어떤 X일 때 A, B, C가 분리되는 Y를 각각 구할 수 있다.

행렬을 통해서 독립적으로 구할 수 있는데 하나로 합친 후 W를 늘려서 한다고 가정

$$\begin{bmatrix} W_{a1} & W_{a2} & W_{a3} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W_{a1}X_1 + W_{a2}X_2 + W_{a3}X_3 \end{bmatrix} = \begin{bmatrix} Y_a \end{bmatrix}$$

$$\begin{bmatrix} W_{b1} & W_{b2} & W_{b3} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W_{b1}X_1 + W_{b2}X_2 + W_{b3}X_3 \end{bmatrix} = \begin{bmatrix} Y_b \end{bmatrix}$$

$$\begin{bmatrix} W_{c1} & W_{c2} & W_{c3} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W_{c1}X_1 + W_{c2}X_2 + W_{c3}X_3 \end{bmatrix} = \begin{bmatrix} Y_c \end{bmatrix}$$

학습 정리


팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
5일차 (5 / 31)	박길준	ML의 실용과 몇가지 팁

주요 내용 요약

- Learning rate를 조절하는 방법
 - Gradient descent 알고리즘을 사용할 때, a값을 말함
 - Learning rate를 잘 정하는 것이 중요하다.
 - 굉장히 큰 값을 정하면 Cost가 커지면, Overshooting이 발생할 수 있다.
 - 반대로 작은 값을 정하면, 최저점이 아닌 경우에서도 멈추는 경우가 발생할 수 있다.
 - 따라서 Learning rate을 조금씩 값을 변경하면서 Overshooting이나 멈춤 현상이 발생 안하도록 조절해야한다.
- 데이터를 선처리하는 방법
 - zero-centered data
 - 0에 가깝게 하도록 모아주는 방식
 - normalized data
 - 특정 범위 안에 들어가도록 한다.

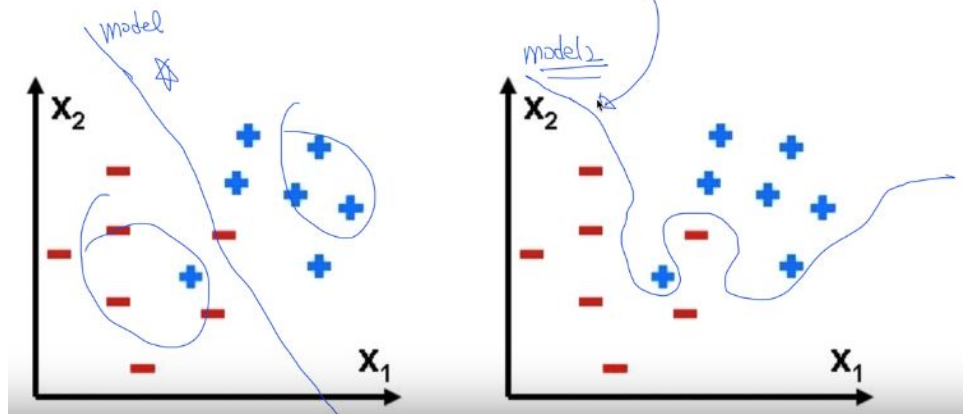
$$\underline{\underline{x'_j}} = \frac{x_j - \mu_j}{\sigma_j}$$

 `x_std[:,0] = (x[:,0] - x[:,0].mean()) / x[:,0].std()`

- cost가 발산하게 된다면 쓰는 방식.

- 머신러닝의 가장 큰 문제인 Overfitting을 방지하는 방법
 - 머신러닝이 학습을 통해서, 모델을 만들어 가는데, 학습 데이터의 경우에는 굉장히 잘 맞지만, 실제 데이터를 입력했을 때 정상적으로 맞지 않는 현상을 의미한다.

Overfitting



b. 줄이는 방법

- i. 트레이닝 데이터가 많으면 많을수록 overfitting을 줄일 수 있다.
- ii. 가지고 있는 feature의 중복된 것들을 줄이는 것도 하나의 방법이다.
- iii. 일반화를 실행하면 된다.
 1. 일반화는 너무나 큰 값을 가지지 않는 것을 의미한다.

- Let's not have too big numbers in the weight

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$$

LOSS (pointing to \mathcal{L})
TRAINING SET (pointing to x_i)
regulation strength (pointing to λ)

2. 공식

4. 성능 측정

- a. 30%정도를 잘라서 앞부분을 training data set / test set이라고 부른다.
- b. test data set은 숨겨놓고 사용하면 안된다.
- c. training data set을 가지고 모델을 학습시킨다.
- d. 그리고 완벽하게 끝났다고 생각이 들었을 때, test data set을 입력한다.
- e. 예측값과 실제값을 비교하여, 성능을 확인한다.
- f. Validation set
 - i. 알파나 람다를 사용하여 training data set외에 모의고사처럼 Validation data set을 사용해본다.

5. Online learning

- a. 데이터나 학습시켜야할 set이 많이 필요할 때 사용하는 방법
- b. 100만개의 데이터가 있을 때, 10만 개씩 잘라서 학습시킨다.
 - i. 이 때, 모델이 할 일은 개별적으로 10만개씩 작업을 할때마다 있는 데이터에 추가로 학습을 시켜야한다.
 - ii. 학습의 하나의 방법이다.

학습 정리

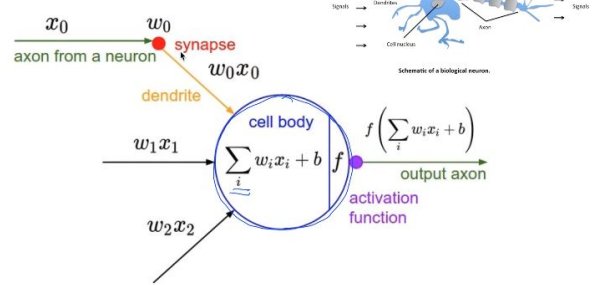
팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
6일차 (6 / 3)	유동관	딥러닝의 기본 개념: 시작과 XOR 문제 딥러닝의 기본 개념2: Back-propagation 과 2006/2007 '딥'의 출현

주요 내용 요약

1. 뉴런에 대한 이해
 - a. 뉴런 : $\text{input} \times w$ 의 합 + bias = (일정 값 이상이 되면 활성화) 아니면 활성화되지 않는다.
 - b. 이 과정을 activation function이라고 한다.

Activation Functions



- c.
 - d. xor 은 값이 다를때 1이 활성화 된다.
 - e. 뉴럴 네트워크는 처음에는 하지 못할 것이라고 했었다.
2. Backpropagation
 - a. backward 를 이용한 알고리즘.
3. Convolutional Neural Networks
 - a. 쪼개서 인식하는 알고리즘.
 - b. 이게 성공적이어서 인기가 많았음. 자동주행차도 만들어줌.
 - c. 앞의 에러를 뒤로 보내는 과정에서 의미가 약해지면서 값이 전달이 잘 안된다.
 - d. 두번째 침체기를 맞이하게 된다.
4. CIFAR
 - a. 연구를 장려하는 단체. 인공지능 연구를 장려해주었다.
 - b. 아무도 일을 하지 않고 있을 때, 연구비를 주어 연구를 할 수 있게 되었다.
 - c. 초기값을 잘 주면 적응하기 좋다. 이름을 바꿔 딥 러닝이라고 하게 되었다.
5. Deep Api Learning
 - a. 넷플릭스, 페이스북, 아마존 등등 많은 곳에서 쓰이고 있음.

Deep API Learning*

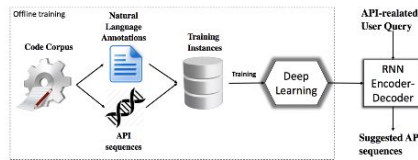


Figure 3: The Overall Workflow of DEEPAPI

b. ⁶⁶ copy a file and save it to ⁶⁵ -your destination path  `FileInputStream.new FileOutputStream.new FileInputStream.getChannel FileOutputStream.getChannel FileChannel.size FileChannel.transferTo FileInputStream.close FileOutputStream.close FileChannel.close FileChannel.close`

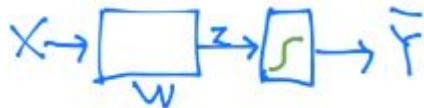
학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

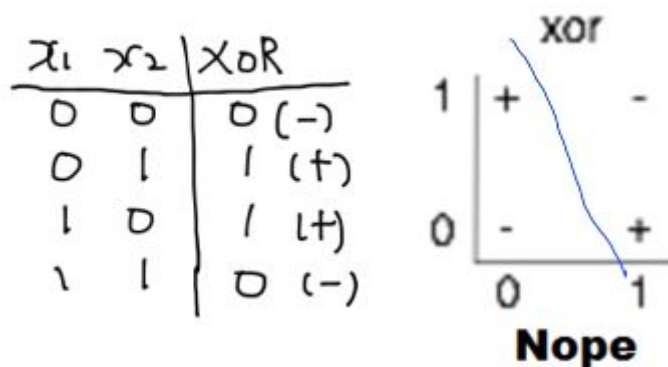
일정	발제자	주제
7일차 (6/4)	임연지	Neural Network 1: XOR 문제와 학습방법, Backpropagation

주요 내용 요약

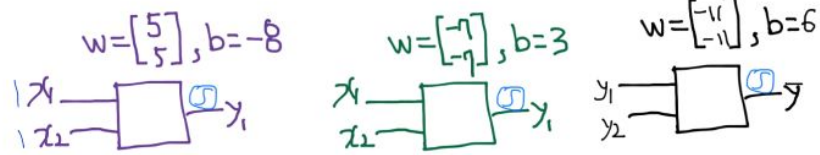
- XOR 문제 딥러닝으로 풀기
XOR은 초창기에 전문가들에게 절망을 줬던 문제



위 모델로는 절대 해결할수 없다고 수학적으로까지 증명했었기 때문
이 모델을 여러 개 합치면 풀 수 있다는 주장이 있었으나, 이걸 각각 복잡한
네트워크에 들어가있는 w, b를 어떻게 학습하는가에 대한 주제가 나왔음



위의 사진처럼 XOR은 Linear하게 +와 -를 구분할 수 없다는 문제점이 있다.



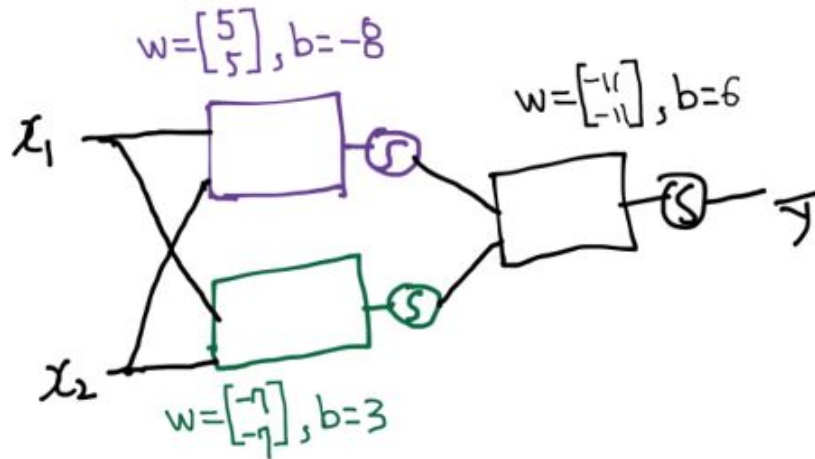
$$[1 \ 1] \begin{bmatrix} 5 \\ 5 \end{bmatrix} - 8 = 5 + 5 - 8 = 2, \text{sigmoid}(2) = 1$$

$$[1 \ 1] \begin{bmatrix} -1 \\ -1 \end{bmatrix} + 3 = -1 + -1 + 3 = 1, \text{sigmoid}(-1) = 0$$

$$[1 \ 0] \begin{bmatrix} -1 \\ -1 \end{bmatrix} + 6 = -1 + 0 + 6 = 5, \text{sigmoid}(5) = 1$$

x_1	x_2	y_1	y_2	\bar{y}	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

이런식으로 계산해서 bar y 를 유추한다 (초록색 y는 y2), sigmoid는 양수일때 1 음수일때 0이다.



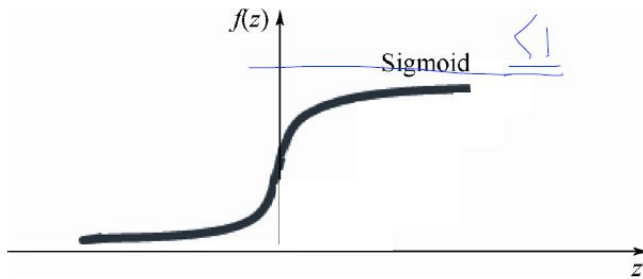
위와같이 정리할수 있고, 이것이 뉴럴네트워크이다.

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

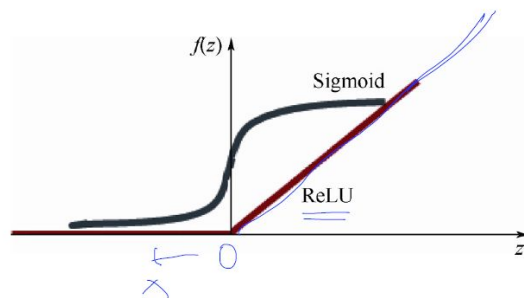
일정	발제자	주제
8일차 (6 / 5)	박길준	Neural Network 2: ReLU and 초기값 정하기 (2006/2007 breakthrough)
주요 내용 요약		
1. Sigmoid의 문제점 a.		

Sigmoid!



- b. 어떤 값이 주어지면 1보다 무조건 작다.
- c. 1보다 작은 값을 계속 곱해나가면 최종값은 작아진다.
- d. 그래서 이러한 문제를 해결하기 위해, ReLU를 사용
 - i. 0보다 작은 경우, 꺼버리고
 - ii. 0보다 큰 경우엔 끝까지 간다.

Sigmoid!



- iii.
- iv. 새로운 함수

ReLU: Rectified Linear Unit

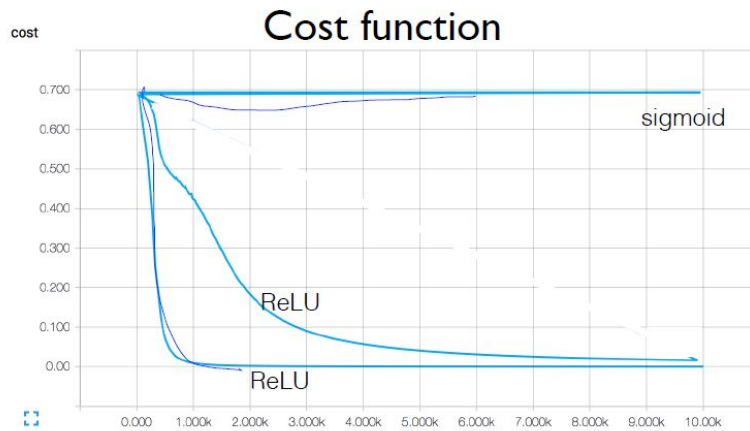
$$\max(0, x)$$



$$L1 = \text{tf.sigmoid}(\text{tf.matmul}(X, W1) + b1)$$

$$L1 = \text{tf.nn.relu}(\text{tf.matmul}(X, W1) + b1)$$

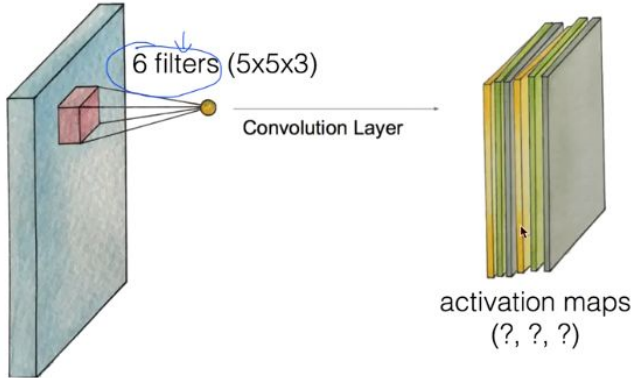
V.



- e.
- f. ReLU함수를 사용하면 위의 그림과 마찬가지로 정확도를 높일 수 있다.
- 2. 초기값을 어떻게 할 것인가.
 - a. Vanishing gradient 문제
 - b. 초기값을 멍청하게 설정한 것을 어떻게 할 지 고민해보자
 - i. 초기값으로 주는 랜덤값에 따라서 값이 변한다.
 - c. 똑똑하게 학습시키는 방법
 - i. 절대 전체를 0으로 넣으면 안된다.
 - ii. RBM을 사용한다.(현재는 많이 사용하지 않는다.)
 - iii. Xavier initialization
 - 1. 입력의 갯수와 출력의 갯수에 따라 초기값을 정하면 좋은 초기값을 줄 수 있다.
 - iv. He initialization
 - 1. Xavier initialization에서 개량화된 버전
 - d. 초기값을 세팅하는 부분은 아직도 연구가 많이 필요한 분야임.
- 3. Overfitting 관련
 - a. 자주 학습된 경우는 높은 정확도를 보이지만, 처음 보는 케이스일 때, 낮은 정확도를 보이는 경우.
 - b. 학습데이터를 최대한 많이 하면 처음 보는 케이스가 줄어들
 - c. 가중치를 너무 큰 값을 안주는 방법을 사용한다.
- 4. Dropout
 - a. neural network에서 오버피팅을 예방하는 심플한 방법
 - b. 랜덤하게 어떤 뉴런들을 잘라버리는 방법
 - i. 사람의 경우에도, 이목구비 중 하나가 없다고 해서 특정 사물을 구별을 아예 못하진 않기 때문이다.
- 5. Ensemble
 - a. 학습시킨 모델들을 합치는 방법
 - b. 2% ~ 4%정도 성능이 향상된다.
- 6. Feedforward neural network
 - a. 모델들을 지속적으로 쌓으면서 좋은 결과를 도출하는 네트워크를 형성
 - b. fast forward와 split & merge 형식으로도 형성 가능
 - c. Recurrent 방식도 가능(옆으로 가지를 치는 형식) -> RNN
 - d. 어떤 방식으로도 본인이 상상하는 모델을 만들어 낼 수 있다.

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
9일차 (6 / 7)	유동관	ConvNet의 Conv 레이어 만들기 ConvNet Max pooling 과 Full Network ConvNet의 활용예
주요 내용 요약		
<ol style="list-style-type: none"> 특정 그림이 특정 영역의 뉴런만 활성화 시킬 수 있음. 그 특정 뉴런의 입력을 나누어 보는 것이다. 이미지를 잘라 각각의 입력으로 넘기게 된다. 이것을 convNet이라 한다. pooling conv relu 여러번 반복하면 그림이 어떤건지 측정할 수 있게 된다. filter 가 하는 일. 이미지에서 숫자를 뽑아낸다. 5*5*3의 값을 한개의 숫자로 만들어내는 방법 : $Wx+b$ 를 이용한다. <p>Output size: $(N - F) / \text{stride} + 1$ </p> <p>e.g. $N = 7, F = 3$:</p> <p>stride 1 $\Rightarrow (7 - 3)/1 + 1 = 5$</p> <p>stride 2 $\Rightarrow (7 - 3)/2 + 1 = 3$</p> <p>stride 3 $\Rightarrow (7 - 3)/3 + 1 = 2.33 \therefore 2$</p> <ol style="list-style-type: none"> 7*7 output! 입력의 이미지와 출력의 이미지가 같게 만들어주는 것을 일반적으로 사용하고 있다. filter weight 를 계속 만들어주는 작업을 진행한다. <p>Swiping the entire image</p>  <ol style="list-style-type: none"> 32x32x3 image filter size를 계속 조절해주면서 activation map을 조정할 수 있다. 이것을 여러번 하면 어떤 이미지인지 학습이 된다. Max pooling <ol style="list-style-type: none"> conv , relu , pool 중에 pool 이란? pooling layer : sampling -> resizing 후 layer 쌓기. 이미지 겹겹이 쌓아서 학습한다. 		

학습 정리

팀	대전3반	구성원	박길준 유동관 임연지
---	------	-----	-------------

일정	발제자	주제
10일차 (6 / 10)	임연지	NN의 꽃 RNN 이야기

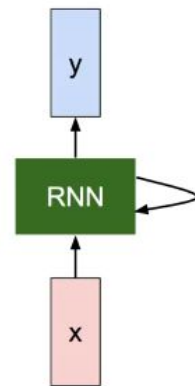
주요 내용 요약

- Sequence data : 음성인식, 자연어 등은 시퀀스 데이터로 되어있음.
(이전에 했던 말이나 단어들을 알아야 맥락이 이해가 되는 개념)
이전에 했던 연산이 그 뒤에 영향을 미쳐야 함.

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

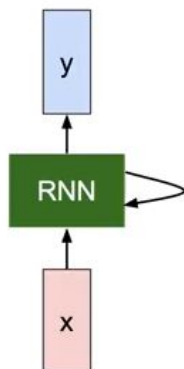
new state some function with parameters W old state input vector at some time step



- RNN은 state라는 개념이있음. 가장 먼저 state(h_t)를 계산해야함. h_t 는 x 라는 입력값과 이 이전의 h_{t-1} 을 가지고 어떤 함수 f_W 를 이용하여 계산함
RNN을 하나의 그림으로 나타내는 이유 : 모든 RNN에 대해 연산하는 function이 동일하기 때문

(Vanilla) Recurrent Neural Network

The state consists of a single "hidden" vector \mathbf{h} :



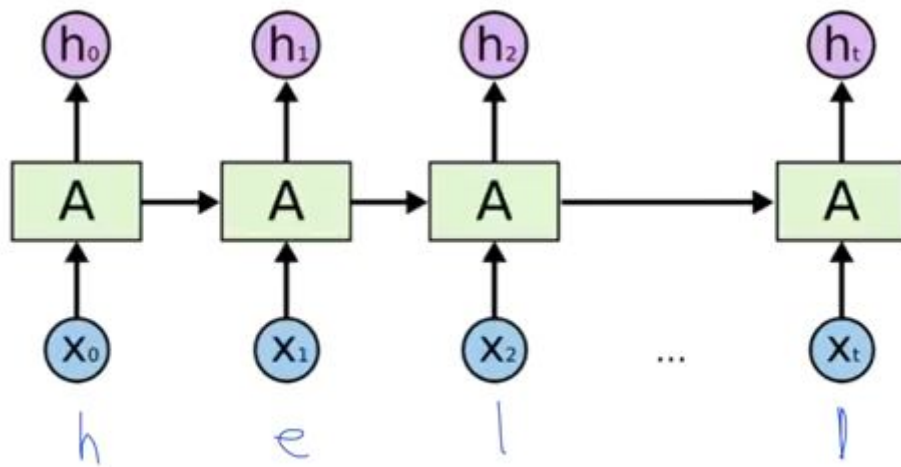
$$h_t = f_W(h_{t-1}, x_t)$$

↓

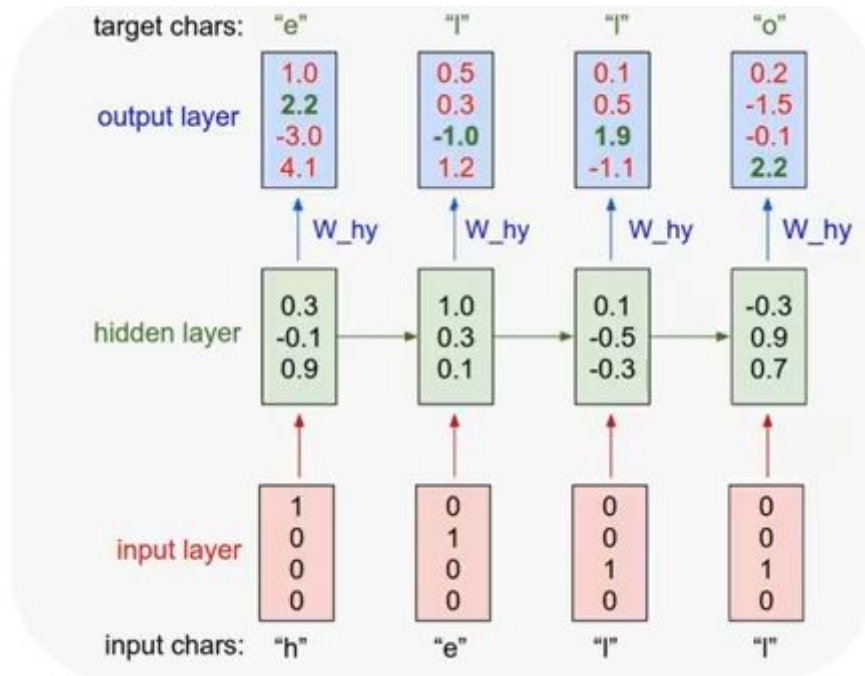
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

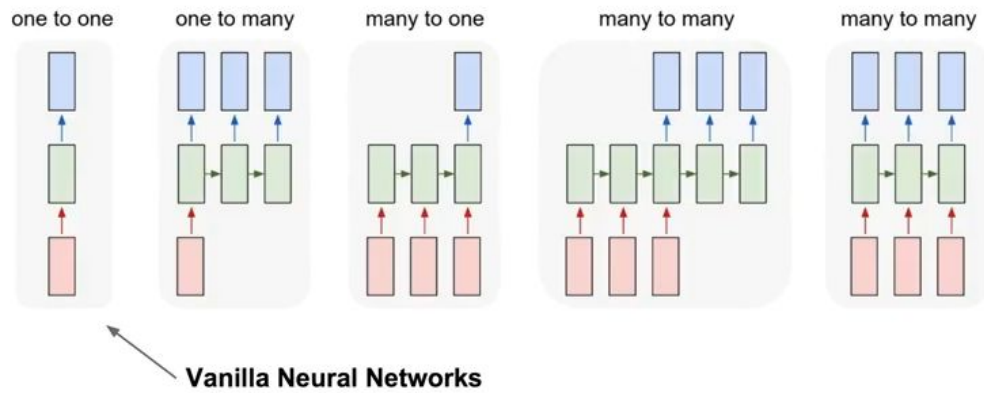
- (Vanilla) RNN : 가장 기본적인 RNN
 h_t 는 다음상태로 넘겨주기 위해 사용하는 것
 W_{hh} , W_{xh} , W_{hy} 3가지 값은 전부 똑같이 들어가게됨



- 현재 글씨를 알고있을때 다음글씨가 무엇일지 구현한다고 가정,



- 각각의 입력을 vector로 표현 : 해당하는 위치를 1로 표현
- hidden state값이 처음값일때 $W_{hh}h_{t-1}$ 값은 없다고 가정
- 그 다음 단계는 다음 input layer값과 이전 hidden layer값을 더한 것이 됨
- output layer에서 가장 큰 수가 원하는 값이 나와야하는데 첫번째, 두번째 값은 가장 큰것을 채택하고 있지 않으므로 error라고 판단할 수 있음
- RNN은 어떠한 sequence를 입력받고 새로운 sequence를 출력하는 것이다



- RNN의 여러가지 형태
 - 1) one to many 부분이 Image Captioning의 예시
 - 2) many to one 부분이 여러 문자열을 받고 하나의 문장을 판단하는 출력값을 받을 수 있음
 - 3) many to many 부분은 sequence of words를 받고 그 형태로 출력해줄 수 있음
 - 4) Multi-Layer RNN : 더 복잡한 학습 가능해짐