

1주차_박은혜

1. 딥러닝 소개

지도학습

- 정답이 주어져있는 데이터로 컴퓨터를 학습시키는 방법. 신경망도 지도학습에 한 종류이다.
- Standard Neural Network, CNN (이미지 데이터), RNN (오디오, 텍스트 데이터), Custom/Hybrid NN

Structured Data vs. Unstructured Data

- Structured Data : Database of data. Each of the features suggests well-defined meaning
- Unstructured Data : refers to audio, image, text data. The data here may be the pixel values of an image.

딥러닝의 부상 이유

딥러닝에 대한 기본적인 이론은 오랫동안 존재해왔다. 그렇다면 왜 이제서야 딥러닝이 부상하고 있는 것일까?

그동안 사용했던 모델들은 데이터의 양이 커져도 성능의 차이가 크게 나타나지 않았다. 심지어 데이터의 양이 적을 때 모델들간에 상대적 순위가 없다. 모델의 차이가 없을 뿐더러, 당시에는 데이터의 양이 많지 않았다.

하지만 그동안 인터넷이 활성화되고, 사람들이 데이터를 모으는데 더 많은 노력을 기울이게 되면서, 더 큰 데이터를 분석할 수 있는 모델에 대한 수요가 늘어났다



높은 성능을 가진 모델을 만들기 위해서는:

1. Large Neural Network
2. Large amount of data



“Scale has been driving deep learning progress”

단순히 신경망 사이즈와 데이터의 사이즈가 커질수록 더 좋은 성능을 낼 수 있게 된다.

초반에는 데이터의 크기와 컴퓨팅 성능 향상이 가장 큰 발전을 이루었다.

알고리즘의 성능 향상은 딥러닝 모델의 학습 속도를 가속화하기 위해 시작되었는데, sigmoid 함수가 아닌 ReLU를 사용하기 시작했다.

- Sigmoid의 단점은, 함수의 경사가 0에 가까워질 수록 파라미터가 천천히 바뀌기 때문에 학습 속도가 매우 느려진다.
- 하지만 ReLU는 sigmoid와 다르게 함수의 경사가 양수에서 항상 1이기 때문에 경사가 0에 수렴할 가능성이 줄어든다.
- 그렇기 때문에 단순히 ReLU함수로 바뀌움으로서 딥러닝 모델의 경사 하강법 알고리즘을 향상 시켜줬다.

빠른 계산 이 중요한 이유

딥러닝은 코드를 짜고, 실행해보고, 수정하는 것을 반복해야한다.

더 효율적으로 수정을 하고 데이터에 더 알맞은 모델을 구하기 위해 모델이 빠르면 빠를수록 좋다.

2. 신경망과 로지스틱 회귀

이진분류

64x64 픽셀의 이미지 데이터를 분석하여 고양이 사진인지 아닌지를 알아본다고 하자.

- 3개의 RGB 행렬의 값들을 하나의 긴 벡터로 만들어준다. 이 벡터의 길이는 $64 \times 64 \times 3 (=12288)$ 이 될 것이다
- 이 x 를 활용해 y 값을 추정한다
- 여기서 y 값은 0 또는 1이다 \rightarrow 1이면 고양이, 0이면 아니다.

(x, y) where $x \in \mathbb{R}^x, y \in 0, 1$

m training samples : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

왜 X 를 행의 형식으로 만들면 안되나?

→ It will make the implementation of the neural network difficult.

y의 예측값을 통해 무엇을 알 수 있나?

Given an x , you want an algorithm to know the probability for $y = 1$ when x is given.

$$\text{given } x, \hat{y} = P(y = 1|x)$$

We want the \hat{y} to tell us the change for x to be a cat picture → y의 예측값은 이 사진이 고양이일 확률을 알려준다



input : $x \in \mathbb{R}^x$

parameters : $w = \mathbb{R}^{n_x}$ and $b = \mathbb{R}$

output :

(1) $\hat{y} = w^T x + b \rightarrow$ linear function for regression

- 이 방법이 안좋은 이유 : \hat{y} is not between $(0, 1)$, as $w^T x + b$ can be much bigger than 1, or even negative.

(2) $\hat{y} = \sigma(w^T x + b) \rightarrow$ using sigmoid function

- If $z = (w^T x + b)$ is very large, then $\sigma(z)$ approaches 1
- If $z = (w^T x + b)$ is very small, then $\sigma(z)$ approaches 0



Logistic Regression

We need to adjust w and b (our parameters) for us to get a better estimate of \hat{y}

또, 로지스틱 회귀의 매개 변수 w 와 b 를 학습하려면 비용함수를 정의해야한다.

손실함수 (Loss Function)

$L(\hat{y}, y) = 1/2(\hat{y} - y)^2 \leftarrow$ 최적화 함수가 **non-convex**이기 때문에 쓰지 못한다.

non-convex이면 지역 최소값들이 많이 생기기 때문에 이후 경사 하강법을 사용하기에 적합하지 않다.

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

- $y = 1 : L(\hat{y}, y) = -\log \hat{y} \rightarrow$ then you want $-\log \hat{y}$ to be as big as possible $\rightarrow \hat{y}$ should be **large**
- $y = 0 : L(\hat{y}, y) = -\log(1 - \hat{y}) \rightarrow$ then you want $-\log(1 - \hat{y})$ to be as big as possible $\rightarrow \hat{y}$ should be **small**

비용함수 (Cost Function)

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

- 전체 훈련 세트에서 얼마나 잘 추측되었는지
- 전체 훈련 세트에서의 손실함수의 평균값
- 매개 변수의 비용처럼 작용 \rightarrow 그래서 LR를 학습할 때 이 cost function을 최소화시켜야 한다.



로지스틱 회귀분석

- **손실함수 (loss function)** : 단일 훈련 샘플이 얼마나 잘 작동하나를 측정
- **비용함수 (cost function) J** : 매개변수 w, b 가 훈련세트 전체를 얼마나 잘 예측하는지를 측정 (손실함수의 평균값)
- **경사 하강법** : 매개변수 w, b 를 훈련하는 방법
경사 하강법의 목적 : 비용함수를 최소화시키는 w 와 b 를 찾는 것

경사 하강법


목적 : 비용함수값을 최소화시키는 w 와 b 를 찾는 것



위에 구한 비용함수가 convex의 형태를 갖고 있기 때문에 경사하강법을 활용할 수 있다.

또한 그렇기 때문에 initial point가 어디인지는 중요하지 않다

아래 **알고리즘**이 수렴할 때까지 w 값을 업데이트 하는 것을 반복한다.

 Repeat $\{ w := w - \alpha \frac{dJ(w)}{dw} \}$

- α : learning rate (학습률) : 경사 하강법의 각 단계의 크기를 결정한다
- $\frac{dJ(w)}{dw}$: 각 갱신할 때 매개변수 w 에 줄 변화

계산 그래프와 계산 방향

신경망계산은 두 가지 부분이다

1. Forward Propagation
2. Backward Propagation

계산 그래프가 왜 이렇게 두 부분으로 나뉘었는지 설명해준다.

역방향과 정방향 계산

- 역방향을 계산 할 때는 각 변수의 도함수를 계산할 때 사용한다.

왜? 결과값을 계산하는 과정에서 나오는 중간변수들의 도함수(변화)를 먼저 계산하는 것이 기존 변수들의 도함수를 계산할 때 더 수월하기 때문. 기존 변수들의 변화가 이 중간 값들의 변화에 영향을 미치기 때문에 그렇다.

- 그래서 도함수를 구해서 어디에 쓰느냐?

도함수는 w 값을 업데이트 할 때 필요하다. 역방향으로 w 값들의 도함수를 구해 w 값을 업데이트하기 위한 값을 구하는 것

- 정방향은 최종값을 최적화할 때 활용한다.

최종 알고리즘

- 약점 : 이렇게 구현하려면 for loop 2개를 많아야한다.
옆 방법에서는 n 이 2개밖에 없으니까 for loop 하나만 썼자만, x 가 많아지면 for loop이 필요하다.
- 딥러닝은 반복이 중요하기 때문에, 스피드가 중요하다.
그렇기 때문에 for loop을 계속 돌리기에는 시간이 너무 많이 걸린다.

For loop을 없애고 efficiency를 올리기 위해 백터화를 시킨다.