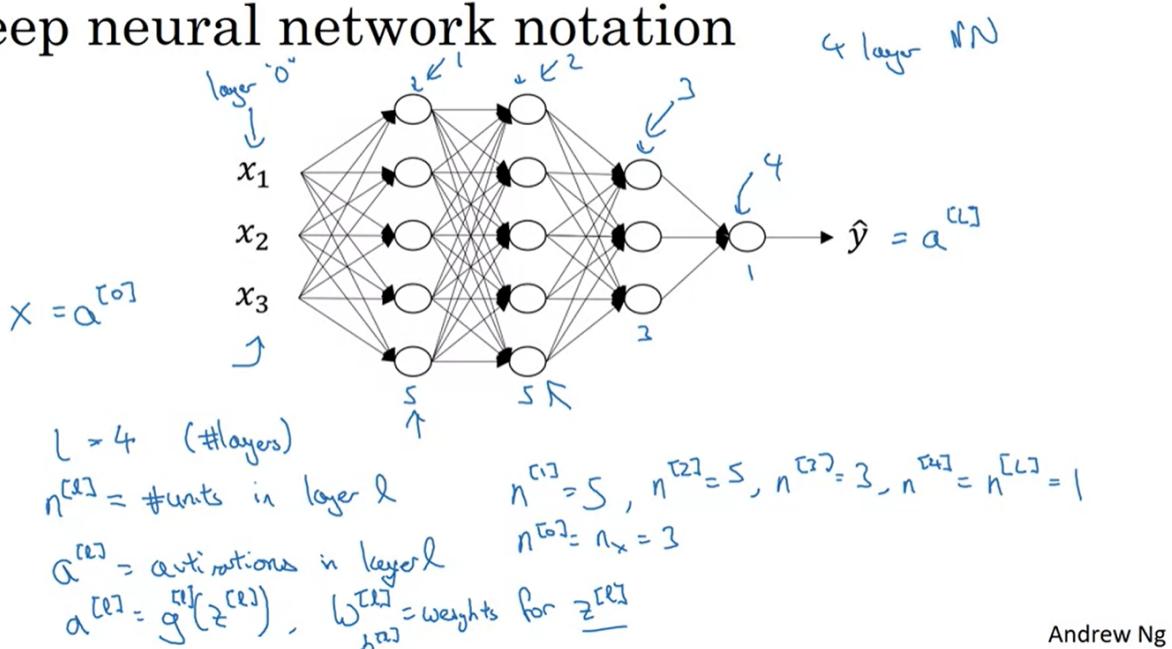




[4주차] Deep L-layer neural network

Deep neural network notation

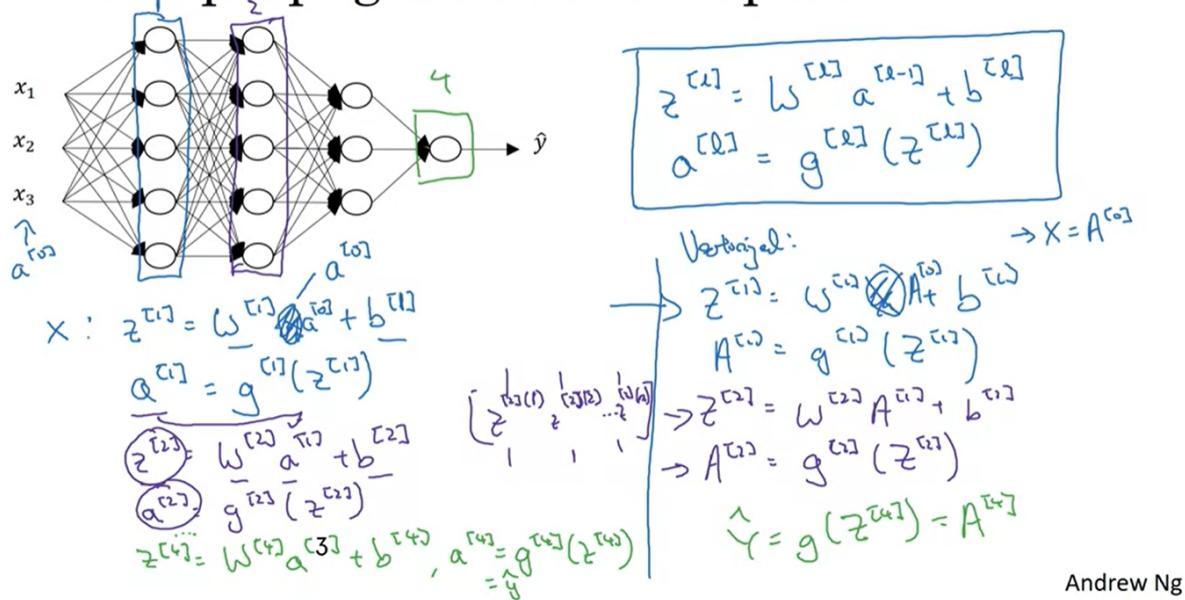


Andrew Ng

이 층에서 숨겨진 유닛은 5, 5, 3 그리고 1 유닛입니다. 여기서 쓸 표기법은, 대문자 L을 쓸것 이구요, 이 것은 네트워크에서의 층 개수를 나타낼 것입니다. 이 경우 L=4입니다. 이것은 층의 개수이죠. 그리고 N의 위첨자 [l]을 사용해서 노드의 개수를 나타낼 것입니다. 또는 소문자 l로 나타낸 유닛의 개수입니다. 이것을 인덱스화하면, 입력값을 "0" 층으로하고 이것을 1층, 이것은 2층, 이것은 3층, 이것은 4층 그렇게되면 이것처럼 $n[1]$ 이 있겠죠. 첫번째 층에서 나온 것이구요, 이 값은 5입니다. 그 이유는 5개의 숨겨진 유닛이 있기 때문입니다. 이것은 $n[2]$ 인데요 2번째 숨겨진 숨겨진 층에서의 유닛 개수인데요 이 값은 마찬가지로 5입니다. $n[3]$ 는 3입니다. $n[4]$ 는 $n[L]$ 입니다. upper unit의 개수는 1입니다. 대문자 L은 4와 같기 때문입니다. 그리고 여기에서는 입력 층으로 $n[0]$ 은 n_x 이고 이 값은 3이 됩니다. 이렇게해서 다른 층마다 있는 노드의 개수를 표기합니다. 각각의 L층에 대해서, $a[l]$ 을 이용해서 l 층에 대한

activation을 표기할 것입니다. 나중에 보겠지만, 방향전파를 위해 $a[l]$ 을 activation $g(z[l])$ 을 계산하게 될 것입니다. 아마도 activation은 l 층에 대해서도 이렇게 인덱싱 될 것입니다. 그리고 $W[l]$ 를 이용해서 l 층에서의 $z[l]$ 값을 계산하는 비중을 표기하도록 하겠습니다. 유사하게 $b[l]$ 은 $z[l]$ 을 계산하는데 쓰입니다. 표기에 대한 내용을 마무리하자면, 입력 특성 값이 x 라고 불립니다. x 는 또한 0층의 activation인데요, 즉 $a[0]$ 은 x 입니다. 그리고 마지막 층의 activation은 $a[L] = \hat{y}$ 입니다. 결과적으로 $a[L]$ 은 신경망의 예상 결과값 \hat{y} 이 되는거죠.

Forward propagation in a deep network



첫번째 트레이닝 샘플은 z 벡터이고요, 이것은 2번째 트레이닝에 대한 z 벡터이고요, m 번째 트레이닝 샘플까지 반복합니다. 그리하여 세로줄에 쌓아 올립니다. 대문자 Z 로 표현하는데요, 대문자 A 도, 대문자 X 도 마찬가지로 이것은 세로 벡터를 왼쪽에서 오른쪽으로 쌓는 것입니다.

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

$$Z_{\text{all}}^{[l]} = W^{[l]} A_{\text{all}}^{[l-1]} + b^{[l]}$$

$$A_{\text{all}}^{[l]} = g^{[l]}(Z_{\text{all}}^{[l]})$$

요약하자면, 여기 위에 변경해서 적을텐데요, 다른 표기 방식을 통해 여기 소문자 z와 a를 각각 상응하는 대문자 값으로 변경해주겠습니다. 이것은 이미 대문자 Z로 보이네요, 이렇게하면 전 방향전파의 벡터화된 버전이 나오는 것입니다. 한번에 전체 트레이닝 세트에 대해 진행하는 경우 말이죠.

The diagram illustrates the forward pass of a neural network. It shows the flow of data from input \$X = A^{(0)}\$ through four layers (\$l=1 \dots 4\$) to output \$\hat{Y} = g(\hat{z}^{(4)}) = A^{(4)}\$. The equations are:

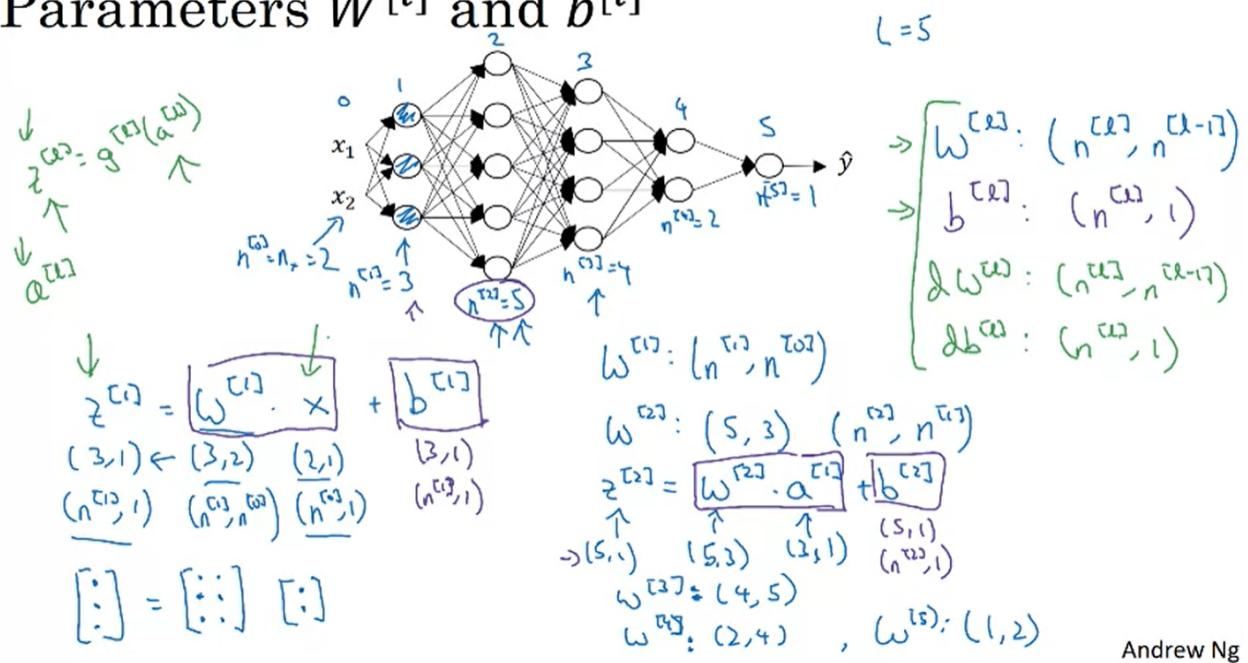
- \$z^{(1)} = \omega^{(1)} \times A^{(0)} + b^{(1)}
- \$A^{(1)} = g^{(1)}(z^{(1)})\$
- \$\rightarrow z^{(2)} = \omega^{(2)} A^{(1)} + b^{(2)}
- \$\rightarrow A^{(2)} = g^{(2)}(z^{(2)})\$
- \$\vdots
- \$\rightarrow z^{(4)} = \omega^{(4)} A^{(3)} + b^{(4)}
- \$\rightarrow A^{(4)} = g^{(4)}(z^{(4)})\$

A yellow spiral arrow on the right indicates the sequence of operations. The handwritten text "Verteilung:" is at the top left. The signature "Andrew Ng" is at the bottom right.

여기서 보는 벡터화의 도입을 보면, 여기에서 for loop이 있을 것처럼 보여집니다. for l=1에서 4까지이죠. for l은 1에서 대문자 L까지입니다. 그리고 첫번째 층에 대한 activation을 계산하고, 두번째 층, 그리고 세번째 층, 4번째 층 이런식으로 말이죠.

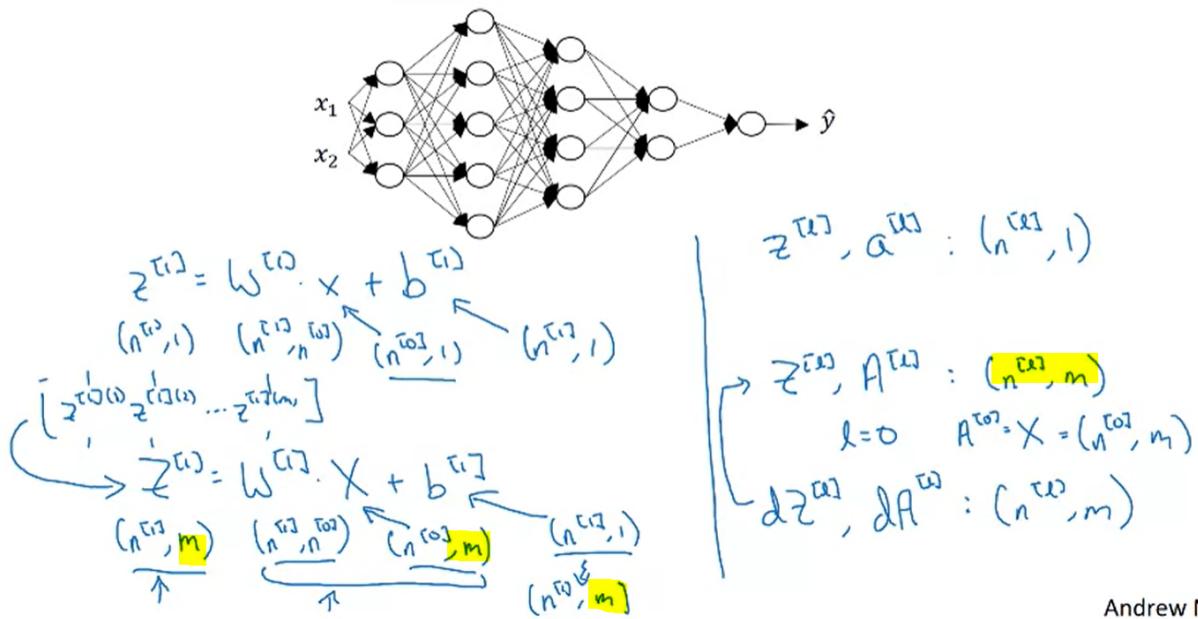
그리고 저는 네트워크를 도입하는 경우, explicit for loop를 제거하고 싶다는 것을 아는데요, 하지만 여기서는 아마 explicit for loop 없이 도입할 수 있는 방법이 없는 것 같습니다. 그러므로 전 방향전파를 도입할 때는 for loop이 있는 것이 괜찮습니다.

Parameters $W^{[l]}$ and $b^{[l]}$



Andrew Ng

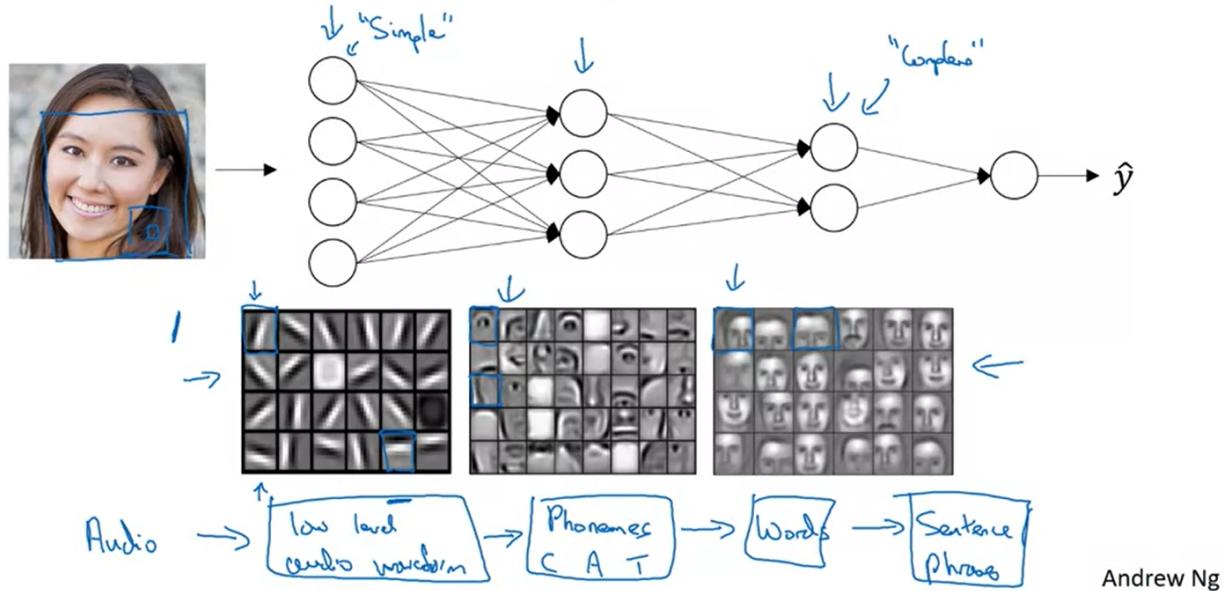
Vectorized implementation



Andrew Ng

벡터화 도입에서 그러나 z , a , x 의 다이멘션은 약간 변할 것입니다. 마지막으로 b_1 은 똑같이 $n_1 \times 1$ 입니다. 하지만 이 값을 갖고 b 에 더하면, 파이썬 broadcasting에서는 이것이 $n_1 \times m$ 매트릭스에 중복될 것입니다.

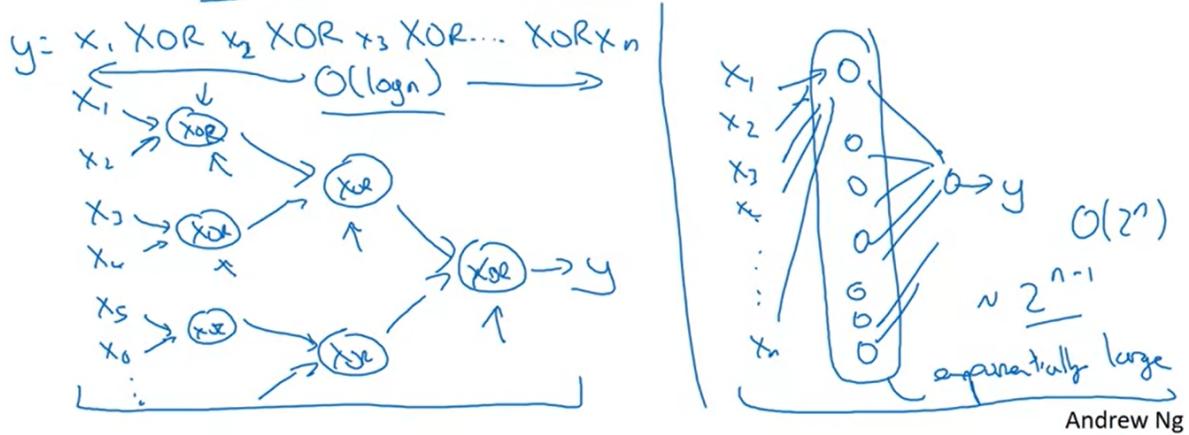
Intuition about deep representation



여러분이 음성 인식 시스템을 만드는 경우에, 음성을 시각화하기는 쉽지 않지만, 오디오 음성 파일을 삽입하면, 첫번째 층의 신경망이 낮은 레벨 음성 wave 형식을 감지하도록 배울 수 있습니다. 예를 들어, 이 톤이 올라가는 것인지 내려가는지 감지하는 것과 같이 말이죠. 백색소음인지, slithering 음성인지, 또는 피치는 어떤지 감지할 수 있습니다. 이렇게 low level wave form 특성들을 감지할 수 있도록 가르칠 수 있습니다. 기 이후에, low level wave forms를 취합하여 기본적인 음성유닛을 감지할 수 있도록 만들 수 있습니다. 음성학에서는 이것을 phonemes라고 하는데요. 예를 들어, cat이라는 단어에서 C가 phoneme이고 A가 phoneme입니다 T도 또 하나의 phoneme입니다. 이렇게 기본 음성 유닛을 감지하도록 배울 수 있고요, 이러한 phoneme를 취합하여 단어를 알아듣게 만들 수 있습니다. 그리고 단어를 합쳐서 전체 문구를 이해하거나 문장들을 이해할 수도 있겠죠.

Circuit theory and deep learning

Informally: There are functions you can compute with a “small” L-layer deep neural network that shallower networks require exponentially more hidden units to compute.



같은 함수를 얇은 네트워크로 계산하려고 하면, 즉, 숨겨진 레이어가 충분하지 않으므로 계산을 하기 위해서는 기하급수적으로 많은 숨겨진 레이어의 수가 필요할 것입니다. 한가지 예제를 통해 한번 보여드리겠습니다. 여러분이 exclusive or를 계산하려고 한다 해보겠습니다. 입력특성 값의 차이를 구하는 것입니다. 즉 $x_1, \text{xor}, x_2, \text{xor} x_3, \text{xor}, x_n$ 까지 계산하려고 합니다. n 또는 $n \times n$ 개의 특성이 있는 경우 말이죠. xor tree를 이렇게 그리면, xor1에서 x_1 과 x_2 를 갖고, x_3 와 x_4 에 대해서 또 xor을 계산합니다. 엄밀히 이야기하면, AND 와 NOT 게이지를 이용하는 경우, XOR 함수를 계산하는데 2개정도의 층이 필요할 수 있습니다. 하지만 비교적 작은 circuit에서는 XOR를 계산할 수 있습니다. 그 다음에 XOR tree를 이와 같이 그릴 수 있습니다. 이렇게 Y 라는 결과값을 주는 circuit을 만들 때까지 말이죠. $y \hat{=}$ Y 입니다.

exclusive or 또는 parity of all these input입니다. XOR 를 계산하기 위해서는 네트워크의 깊이는 order of $\log N$ 일 것입니다. 이런 XOR tree의 경우 말이죠. 그러므로 노드의 개수, circuit components 개수, 또는 gates 숫자는 여기 네트워크에서는 그리 많지 않습니다.

exclusive OR를 계산하기 위해서 그리 많은 gates를 요하지 않습니다. 그러나 만약 복수의 숨겨진 레이어가 있는 신경망을 사용할 수 없는 경우, 이 경우는 order $\log n$ and 숨겨진 레이어s 인데요 하나의 숨겨진 레이어로 함수를 계산해야하는 경우, 여기 이 것들이 특정 숨겨진 유닛으로 가는 것인데요, 그러면 이 것들은 Y 를 결과값으로 줍니다. XOR 함수의 parity를 계산하기 위해서는 여기 숨겨진 레이어가 기하급수적으로 커야할 것입니다. 2 의 n 승의 배열을 열거해하기 때문이죠 즉, 2 의 n 승입니다, 가능한 입력값의 배치인데요, exclusive or 가 1 또는 0이 되는 경우입니다. 결과적으로 숨겨진 레이어가 기하급수적으로 큰 값이 필요하게 됩니

다. bit의 단위로 말이죠 제가 생각하기엔, 엄밀히 맑하면 이것은 2의 $n-1$ 승으로 할 수 있을 것입니다.

Q. 디어는 전국에 총 2000대 정도를 운용하고 있습니다. 디어의 매출은 어느 정도일 것 같 은가요? 필요한 단서는, 자유롭게 질문해도 좋습니다.

Q. 매출이라고 함은, 순이익을 말씀하시는 건가요? 단순히 매출을 말씀하시는 건가요?

A. 단순 매출을 의미합니다.

Q. 매출은 연 단위로 계산하나요, 월 단위로 계산하나요?

A. 월 단위로 계산합니다.

+ :: I

Q. 킥보드는 한 대당 평균 몇 분 정도 운용되나요?

A. 50분정도 운용됩니다.

Q. 일별로 운용되는 킥보드에 편차가 있거나, 전국 지역마다 이용습관 (이용 시간 간격이나 이용 특징 등) 에 차이 가 있나요?

A. 유의미한 차이는 없습니다.

Q. 몇몇 회사들의 경우 킥보드를 새벽에 전량 수거하고 재배치하는 정책을 취하고 있습니다. 디어의 경우는 24시 간 운용하고 있나요?

A. 저희는 개별 배터리 교체 정책을 취하고 있습니다.

Q. 그렇다고 하더라도, 정비소에 들어가있는 킥보드들이 있고 밖에 나와있는 킥보드들이 존재할텐데, 이것들이 전 체 중 어느정도의 비율을 차지하는지 알 수 있을까요? 주말과 평일에 그 편차가 있다면 알려주세요.

A. 운용률이라고 부르며, 80%정도를 차지합니다. 주말과 평일에 풀리는 양은 비슷합니다.

Q. 야간과 주간 주행요금에 차이가 있나요?

A. 없습니다.

Q. 디어의 경우 주행요금을 할인해주는 반납 허브가 존재하던데, 허브 환수율은 얼마나 되나요?

A. 7%입니다.

최종 A.

디어는 5분에 780원 + 분당 150원의 정책을 취하고 있고, 하루에 대당 15분 15분 5분 5분 5분 5분 총 5번을 운 용한다고 가정하면, $780 * 5 + 150 * 20 = 7800$ 이므로, 대당 하루에 7800원정도의 매출을 올립니다.

$7800 * (2000 \text{ 의 } 80\%)$ 이므로, 하루에 약 12,480,000 원의 매출이 전체 킥보드로부터 잡히게 되고, 한달에는 약 3억 5천~4억의 매출이 잡히게 됩니다. 허브반납률은 충분히 작아 이를 고려하지 않은 수치입니다.

- 거의 일치했다고 합니다.

Q. 만약 당신에게 1년 내에 건국대학교 1바퀴를 자율주행하도록 마일스톤이 떨어진다면, 어 떻게 문제를 해결해 나갈 것 같은가요?

우선 단계별로 해야 하는 일들의 중간다리를 충분히 만들어놓고, 그것을 위해 완수해야 하는 일들의 리스트를 만드 어 병렬적으로 실행해 나갈 것 같습니다.

Q. 만약 당신에게 1년 내에 건국대학교 1바퀴를 자율주행하도록 마일스톤이 떨어진다면, 어떻게 문제를 해결해 나갈 것 같은가요?

우선 단계별로 해야 하는 일들의 중간다리를 충분히 만들어놓고, 그것을 위해 완수해야 하는 일들의 리스트를 만들어 병렬적으로 실행해 나갈 것 같습니다.

Q. 혹시 이것이 기술적인 답변을 듣고 싶어서 질문을 하신 건가요? 아니면 프로젝트 진행과 관련된 것을 질문하고 싶으셨던 건지 알 수 있을까요?

A. 문제상황에 대한 대처태도를 보기 위함이었고, 충분한 답변이 되었습니다.

Q. 지금까지 해 왔던 프로젝트들 중 가장 마음이 가는 프로젝트는 무엇인가요? 그리고 가장 아쉬웠던 점들은 무엇인가요?

키워드

- 임베디드시스템 수업 과제로 했던 프로젝트
- end-to-end

아쉬웠던 점 키워드

- HW부터 SW, Demo, 시연까지 혼자서 전부 다 했어야 함.
- 나는 한 분야도 잘 모르는데 모든 분야를 신경썼어야 함.
- 내가 만들고 싶은 기능들에 비해, 최소요구조건만 달성하는 것에 만족해야 했음.
- 이를 개선하려고 마치고 2달동안 공부 중이었음.

Q. 지금 여러 가지 일을 하고 계신 것 같은데, 이것들에 대해서 디어와 조정할 수 있는 부분에 대해서 이야기를 해 주셨으면 합니다. 현재 해당 기업들 (및 단체들) 에 얼마나 일을 하고 있나요?

Q. 킥보드 자율주행에 있어 가장 어려울 것 같은 점은 무엇인가요?

I

Q. 차량 자율주행과 킥보드 자율주행에 있어 가장 큰 차이가 무엇일 것 같나요?

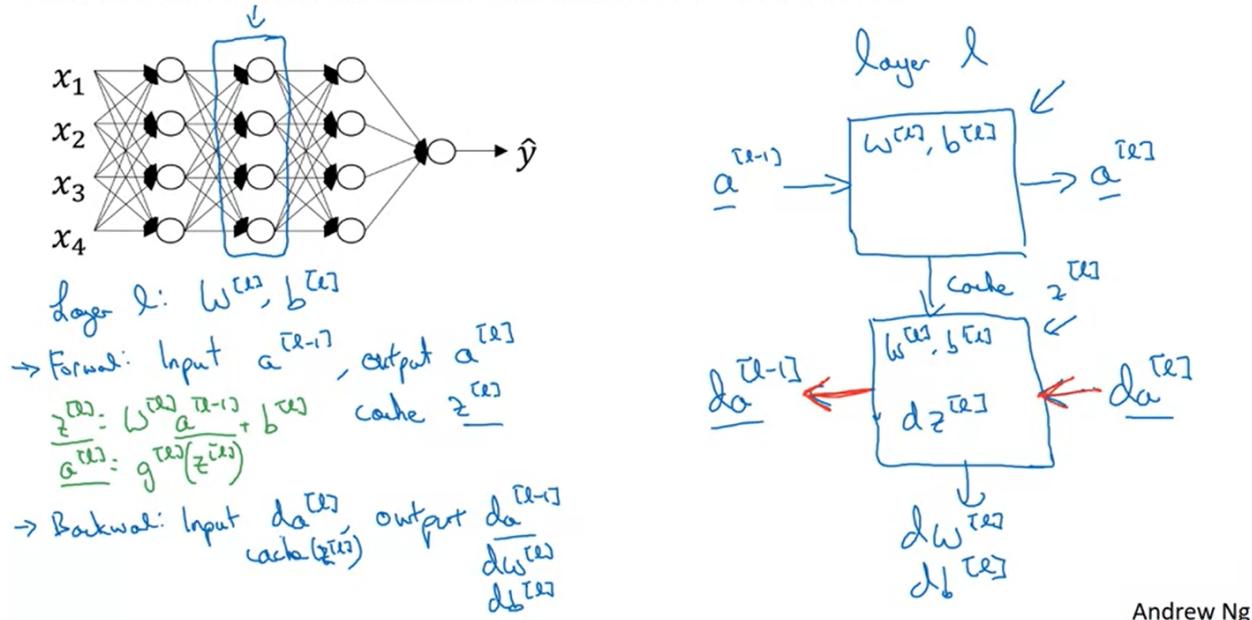
다행인 점 먼저 이야기

- 사고에 대해서 조금 부담이 적은 편이다.

더 어려운 점

- 오히려 차가 낫다 : 대한민국 골목길은 너무 거칩니다. 때로는 인도주행이 불가피하기도 하고, 작은 턱 하나를 넘지 못해 쓰러지고는 할 것입니다. 주행 환경이 훨씬 복잡합니다. 도로같은 경우 선이라도 있지, 이건 어디로 가야 할지 정말 변수가 많아도 너무나 많습니다.

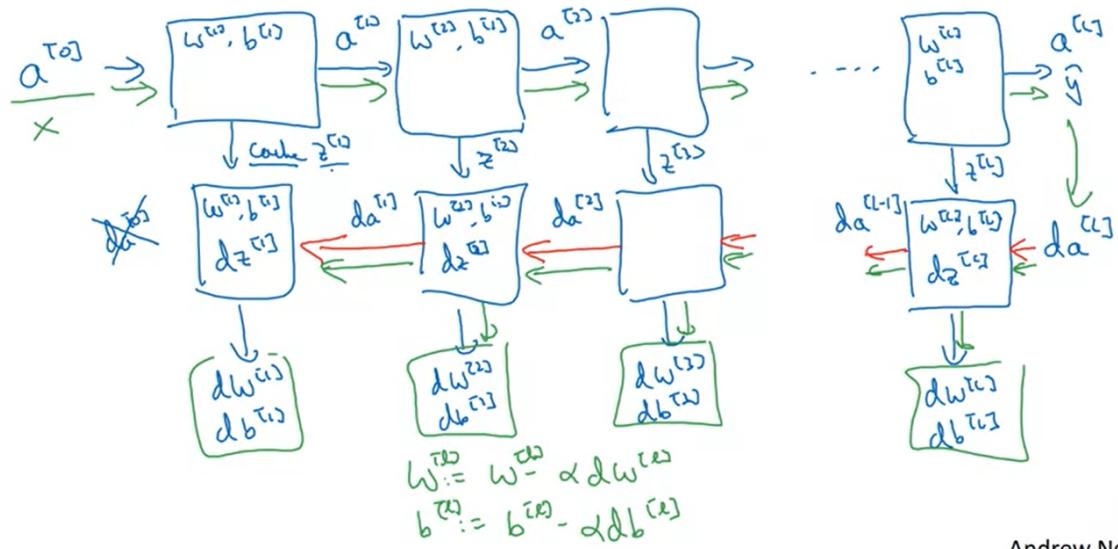
Forward and backward functions



Andrew Ng

이건 입력 feature에 대한 미분이므로, 지도 신경망의 가중치 훈련에는 별 쓸모가 없습니다
그러니 여기서 멈춰도 됩니다.

Forward and backward functions



Andrew Ng

신경망 학습의 한 이터레이션은 A0인 X 로 시작하여 다음과 같이 순전파 과정을 거쳐서 \hat{Y} 를 계산합니다. 그리고 이 값을 사용하여 역전파를 합니다 이제 필요한 모든 미분값들이 있으니 각 레이어에서 W 는 기존값에서 학습률과 dW 의 곱을 뺀 값이 될껍니다 마찬가지로 b 도 역전파를 통해 필요한 모든 미분값을 얻어 업데이트됩니다 여기까지가 신경망 경사 하강법의 한 이터레이션입니다

Forward propagation for layer l

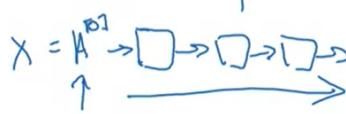
→ Input $a^{[l-1]} \leftarrow$

→ Output $a^{[l]}$, cache ($\underline{z}^{[l]}$)

$$z^{[l]} = W^{[l]}. a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

$$\begin{matrix} a^{[0]} \\ A^{[0]} \end{matrix}$$



Vertwijf:

$$z^{[l]} = W^{[l]}. A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(\underline{z}^{[l]})$$



Andrew Ng

Backward propagation for layer l

→ Input $\overline{da^{[l]}}$

→ Output $\overline{da^{[l-1]}}, \overline{dW^{[l]}}, \overline{db^{[l]}}$

$$\overline{dz^{[l]}} = \overline{da^{[l]}} * g^{[l]}'(z^{[l]})$$

$$\overline{dW^{[l]}} = \overline{dz^{[l]}} \cdot a^{[l-1]T}$$

$$\overline{db^{[l]}} = \overline{dz^{[l]}}$$

$$\overline{da^{[l-1]}} = W^{[l]T} \cdot \overline{dz^{[l]}}$$

$$\overline{dz^{[l]}} = W^{[l-1]T} \cdot \overline{dz^{[l-1]}} + g^{[l-1]}'(z^{[l]})$$

$$\overline{dz^{[l]}} = \overline{dA^{[l]}} * g^{[l]}'(z^{[l]})$$

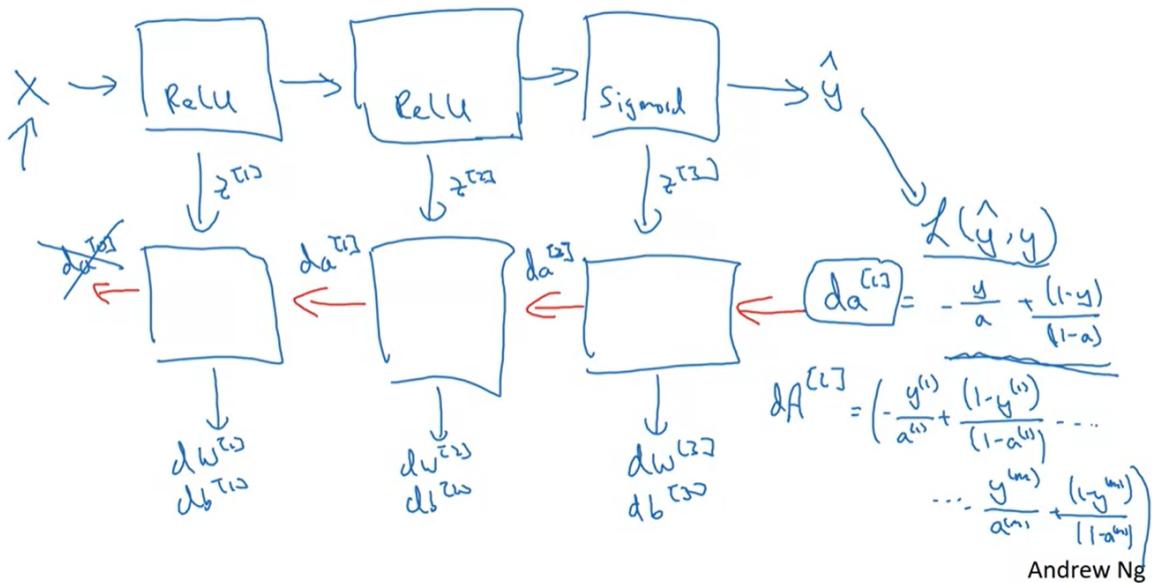
$$\overline{dW^{[l]}} = \frac{1}{m} \overline{dz^{[l]}} \cdot A^{[l-1]T}$$

$$\overline{db^{[l]}} = \frac{1}{m} np.sum(\overline{dz^{[l]}}, axis=1, keepdims=True)$$

$$\overline{dA^{[l-1]}} = W^{[l]T} \cdot \overline{dz^{[l]}}$$

Andrew Ng

Summary



Clarification about What does this have to do with the brain video

Note that the formulas shown in the next video have a few typos. Here is the correct set of formulas.

$$dZ^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L-1]T}$$

$$db^{[L]} = \frac{1}{m} np.sum(dZ^{[L]}, axis=1, keepdims=True)$$

$$dZ^{[L-1]} = W^{[L]T} dZ^{[L]} * g'^{[L-1]}(Z^{[L-1]})$$

Note that * denotes element-wise multiplication)

:

$$dZ^{[1]} = W^{[2]} dZ^{[2]} * g'^{[1]}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]T}$$

Note that $A^{[0]T}$ is another way to denote the input features, which is also written as X^T

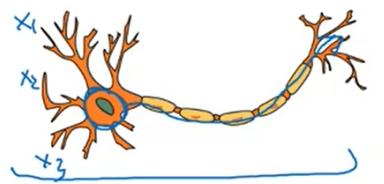
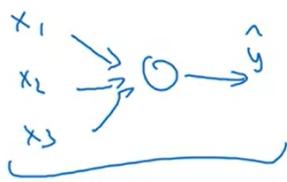
$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis=1, keepdims=True)$$

Forward and backward propagation

$$\begin{aligned} Z^{[1]} &= W^{[1]}X + b^{[1]} \\ A^{[1]} &= g^{[1]}(Z^{[1]}) \\ Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\ A^{[2]} &= g^{[2]}(Z^{[2]}) \\ &\vdots \\ A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y} \end{aligned}$$

"It's like the brain."

$$\begin{aligned} dZ^{[L]} &= A^{[L]} - Y \\ dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\ db^{[L]} &= \frac{1}{m} np.\text{sum}(dZ^{[L]}, axis=1, keepdims=True) \\ dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\ &\vdots \\ dZ^{[1]} &= dW^{[1]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\ dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\ db^{[1]} &= \frac{1}{m} np.\text{sum}(dZ^{[1]}, axis=1, keepdims=True) \end{aligned}$$



Andrew Ng