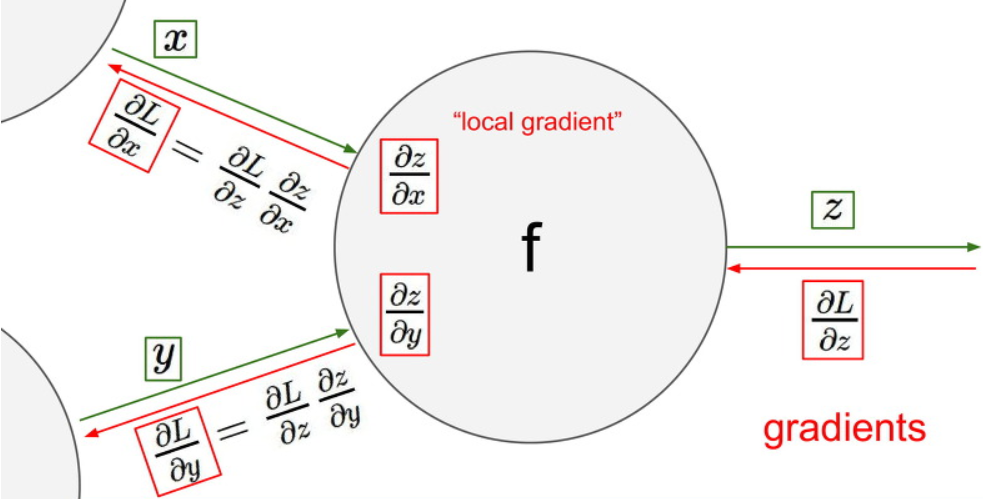
**CS231n 4. Introduction to Neural Networks**



**Computational graphs**

이전강의: Linear Classification에 사용하는 weight 값을 평가하기 위한 loss function, weight 값의 최적화를 위한 Gradient Descent(경사 하강법)과 같은 Optimization 기법

이제 하나의 Linear Classification이 아니라 임의의 복잡한 함수를 통해 analytic gradient값을 계산하는지 배운다.

먼저 이를 배우기 전에 우리가 앞서서 배운 내용들을 Computational Graphs형태로 표현할 수 있다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

예제: input이 x, W인 선형 분류기. 곱셈노드는 행렬곱셈을 나타내고 W와 x의 곱셈은 스코어벡터를 출력한다. Hinge loss(계산노드), regularization항(R). 최종Loss는 R항과 데이터항의 합이다.

Computational Graph를 사용해서 함수를 표현하면 backpropagation기술을 사용가능.

Computational Graphs로 표현하면 input 값과 local gradient값을 쉽게 파악할 수 있습니다.

또한 복잡한 형태의 Anaytic gradient를 보다 쉽게 계산해 낼 수 있습니다.

**Backpropagation**

Backpropagation (역전파법)은 gradient를 얻기위해 computational graph내부의 모든 변수에 대해 Chain Rule을 재귀적으로 사용. -> 매우 복잡한 함수를 이용하여 작업할 때 유용

예시:

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 함수 f를 이용해서 computational graph로 나타냄
2. X=-2, y=5, z=-4
3. 중간값들도 계산해주면 최종값은 -12
4. 중간 변수에 이름을 주자. q=x+y 그러면 f=qz
5. 이제 뒤에서 부터 차례대로 Gradient 값을 구해보자

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 마지막값의 Gradient 값은 1

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. z값에 대한 Gradient 값은q 따라서 q=3

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. q값에 대한 Gradient 값은z 따라서 z=-4

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 이제 x와 y에 대한 값을 계산하기 위해서 앞에서 계산한 Gradient 값과 현재 local Gradient 간에 Chain Rule를 적용하여 계산 -> q값에 대한 f의 Gradient와 y값에 대한 q의 Gradient를 곱하면 -4 \* 1 = -4
2. 그림에는 없지만 x gradient 값 또한 -4 \* 1 = -4를 가진다는 것을 Backpropagation (역전파법)을 이용하여 유추가능

텍스트, 도표, 원, 폰트이(가) 표시된 사진

자동 생성된 설명

자 위 설명을 좀더 보기 쉽게 그림으로 설명:

우선 x와y값을 입력으로 받고 f라는 함수에 집어넣어서 z라는 값을 얻어냅니다.

이때 우리는 z를 x로 미분한 값과 z를 y로 미분한 값을 얻어낼 수 있습니다.

이를 local gradient 값이라고 합니다.

이제 z를 끝으로 L, 즉 loss를 계산했다고 했다면,

우리는 먼저 마지막에서 부터 L를 z로 먼저 미분.

다음 L를 x, y로 미분한 값을 구할 때, 아까 구해놨던 z를 x, y로 미분한 값과 L를 z로 미분한 값을 곱하는 Chain rule를 이용하여 구할 수 있다.

이렇게 입력을 받고 loss값을 구하는 계산 과정 = forward pass

forward pass가 끝난 이후 역으로 미분해가며 기울기 값을 구해가는 과정 = backward pass

좀 복잡한 예시:

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

각 local gradient 값을 미리 구하고, 전에 있던 input 값을 이용하여 뒤에서부터 backpropagation를 한 결과 값.

여기서 중요하게 보아야 할 부분은 아래의 파란색 박스 부분이다.

이 게이트의 연산부분은 sigmoid 함수와 같은 형태.

이러한 sigmoid 함수의 미분 결과는 (1-sigmoid(x)) \* sigmoid(x)와 같은 형태를 가진다.

이를 이용하여 여러 과정을 스킵하고 간단하게 계산 할 수 있다.

아래의 예시 같은 경우 sigmoid(x)의 출력값인 0.73를 이용하여

loss = (1-0.73) \* 0.73 = 0.2 값을 가지는 것을 볼 수 있다.

Backpropagation을 하면서 자주 보일 패턴

텍스트, 도표, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명  
add gate는 덧셈+,  
max gate는 최대값을,  
mul gate는 곱셈x를 구하는 gate다.

이는 backward pass를 할때에는 조금더 간단하게 계산이 가능.

* add gate: 이미 가지고 있던 gradient를 각각의 노드에 분배.
* max gate: 더 큰 쪽에만 gradient를 전달하고 작은 쪽은 0값을 줌.
* mul gate: 현재의 gradient를 각각 숫자에 곱해서 바꿔치기.

원, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

노드 하나에서 다른 노드 두개로 이어지는 경우, backpropagation을 수행할 때

앞 노드 gradient는 뒤의 두개의 노드에서 오는 미분 값을 더한 값이 된다.

반대로 앞의 노드 하나만 바뀌어도 뒤의 노드 두 개가 모두 바뀐다는 사실!

**Gradient for vectorized code**

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

자 앞에서 한 내용들은 어떤 한 변수 값으로 Gradient을 계산했다.

하지만 우리가 실제로 사용하는 입력값은 스칼라값이 아닌 vector값 이므로 다변수벡터함수(vector-valued function of multiple variables)에 대한 일차미분값이 필요하다.

이를 Jacobian (야코비안) matrix 라고 한다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

예시:4096차원의 input 벡터-> Jacobian matrix = 4096 x 4096의 크기를 가짐

여기서 minibatch를 100이라고 한다면 Jacobian matrix의 크기는? 409600 x 409600

여기서는 Jacobian matrix가 input matrix의 길이를 변으로 가지는 정방형 행렬임을 알고 넘어가면 된다.

벡터 입력에 대한 Gradient 값을 구하는 과정:

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

먼저 Linear Classifier 이후에 L2 Regularization으로 loss를 사용한다고하자.

이때에 L2에 입력 q=Wx라고 할때, L2의 Gradient 값은 2q임을 알수 있다.

텍스트, 도표, 폰트, 친필이(가) 표시된 사진

자동 생성된 설명

이후에 W에 대한 Jacobian matrix 값과 앞에서 L2에 대해 계산한 결과를 Chain rule를 이용하여 계산

텍스트, 폰트, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

마지막으로 입력 X에 대한 Jacobian matrix에 대해서 앞에 과정을 반복하여 계산

Backpropagation 코드로 구현:

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

먼저 Forward pass에서는 노드의 출력을 계산하는 함수를 구현하고

Backward pass에서는 gradient를 계산한다.

여기서 Forward의 값(연산 결과)은 이후에 Backward pass에서 사용하기에 저장해야함!

**Neural Networks**

Neural Networks (인공 신경망)는 앞에서 배운 Linear Classifier을 2개 이상 쌓아올리는 형태이다.

여기서 중요한 부분은 이 사이에 Non-linear function를 사용해야 한다는 점이다.

예시: 3072개의 입력 x와 W1과 linear classification 결과로 100개의 h 값을 계산 ->

이 값에 Non-linear max 함수를 취한 뒤에 -> W2값을 linear classification으로 사용 -> 10개의 결과

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

이러한 과정은 기존에는 선형 레이어 하나만 쌓아서 빨간색 자동차만을 찾았다면,

Neural Networks를 이용하면 빨간색, 노란색 자동차와 같이 여러 색깔을 찾을 수 있는 모델을 구축할수있음.

이러한 방법으로 레이어를 쌓아가면서 여러 특징을 추출 할 수 있다

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Neural Network을 2개뿐 아니라 3개 이상으로 구성할 수 있다.

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

Neural network는 뉴런의 구조와 Computation Graph의 구조의 유사성을 따온 구조이다.

하지만 실제의 뉴런 구조와 같다고 생각하면 오산이다.

텍스트, 도표, 스크린샷, 원이(가) 표시된 사진

자동 생성된 설명

중간중간에 모든 노드가 다음의 모든 노드에 영향을 끼치는 레이어를 fully-connected layer라고 한다.

Neural Networks의 예에서도, 모든 W값들이 다음 값들에 영향을 미쳤으니 fully-connected layer다.

위 그림의 2-layer에는 hidden layer가 1개, 3-layer에는 hidden layer가 2개 이므로, 각각 1-hidden-layer 인공신경망, 2-hidden-layer 인공신경망이라고 불린다.