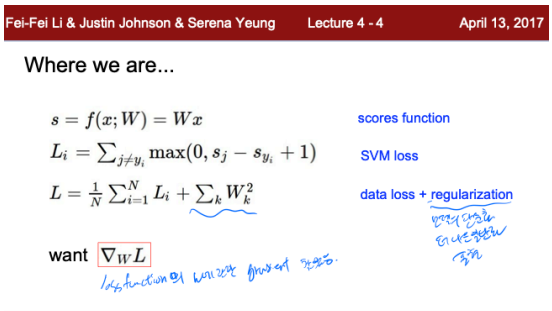
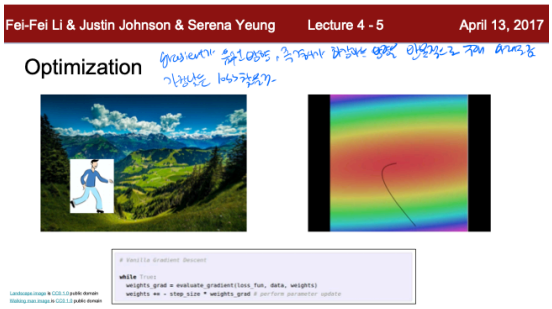
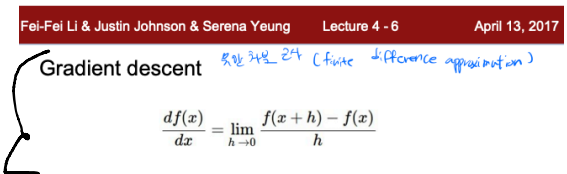






우리가 지금까지 배워온것중에

CS231 3~4강의를 되새겨 본다면

기본적인 인공신경망(Neural Network)에서

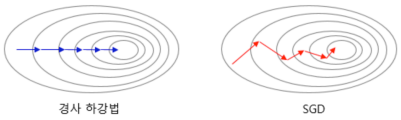
Scores function을 구하고

Loss function(SVM, Softmax등)을 구하여

산꼭대기에서 최저 (목표) 까지 내려올 수 있는

Optimizer를 배웠었다.

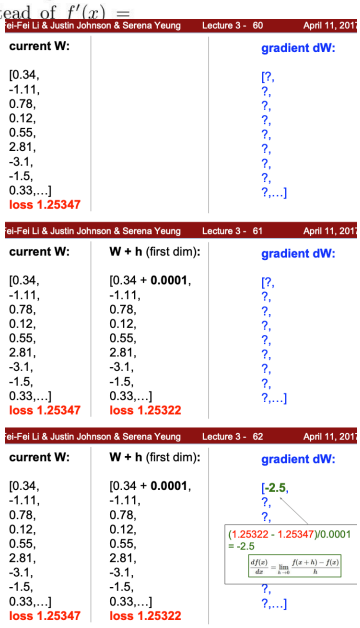
Optimizer에는 배치경사하강법(Batch Gradient Descent)
확률적 경사하강법(Stochastic Pradient Descent, SGD)
미니 배치 경사하강법.. 모멘텀, 아다 그라드, Adam 등등



근데 문제에서는 묻는다 The finite difference approximation of the gradient 중에 Centered Difference Formula를 쓰는 이유가 kinks를 피하는거때문이냐고?! 그럼 일단 The finite difference approximation 부터 알아보자

11. **True or False:** One reason that the centered difference formula for the finite difference approximation of the gradient is preferable to the uncentered alternative is because it is better at avoiding kinks in the objective. Recall: the centered formula is $f'(x) = (f(x+h) - f(x-h))/2h$ instead of $f'(x) = (f(x+h) - f(x))/h$.

위에 여기 살펴보면 Gradient Descent 방법 중 Numerical gradient와 Analytic gradient가 있따. 먼저 Numerica Gradient를 보면 산에서 내려오는 기울기를 발걸음 한폭 한폭 작은 발걸음 하나하나 구해서 기울기를 구하여 전체 모델 함수의 기울기를 통해 가중치의 최적값을 구할 수 있겠다..
하지만 3강이 기억날듯 안날듯 하지만 pdf 자료를 보면
옆에 그림처럼 수많은 여러 계산을 통해서 구해야만 값을 구할 수 있다. 즉, 느리다.



이때, Analytic gradient로 넘어가기전에 Numerical Gradient 방법에 pdf에 있는 기본 공식보다 이론적으로 더 정확도가 높은 근사 공식이 있다.
이것이 바로 The Centered Formula 이다,.

왼쪽 일반 finite difference approximation, 오른쪽 centered finite.. approximation

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x)}{h} \text{ (bad, do not use)} \qquad \frac{df(x)}{dx} = \frac{f(x+h) - f(x-h)}{2h} \text{ (use instead)}$$

식으로 보면 X 기준으로 +h, -h를 해주기 때문에 최초 식보다 계산량이 두배 많지만 !!
훨씬 정확한 근사를 제공한다. (이는 테일러 전개를 고려하면 이유를 금방 알 수 있음)

테일러 전개에서

$f(x+h) = f(x) + hf'(x) + O(h)$ 정리하

$f'(x) - \frac{f(x+h) - f(x)}{h} = O(h)$

two-point (centered) difference $f'(x) = \frac{f(x+h) - f(x-h)}{2} \Rightarrow O(h^2)$

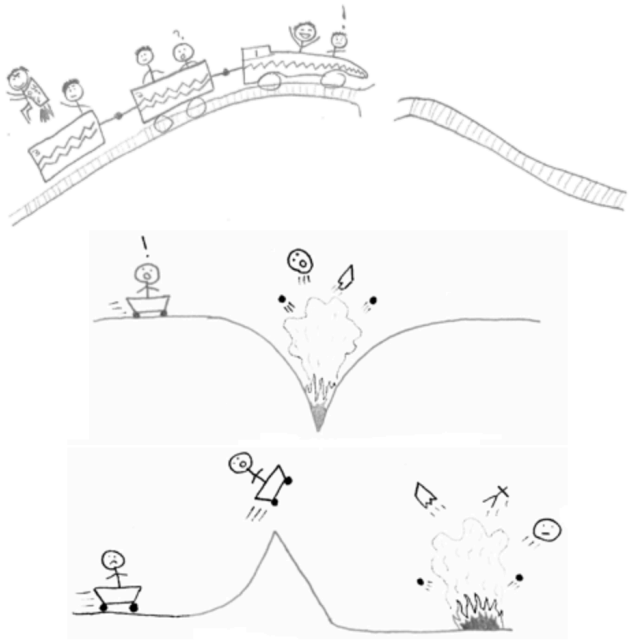
$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x)\frac{h^3}{6}$

$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2} - f'''(x)\frac{h^3}{6}$

$\therefore O(h^2)$ 정도는 안될까요 ... ㅠㅠ

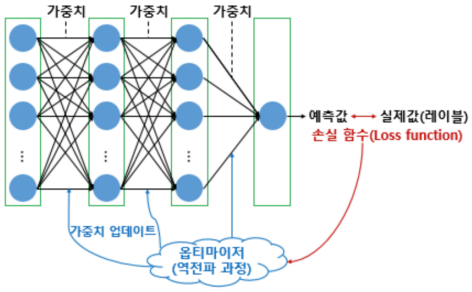
테일러 전개란.. 위키에서 정의된걸 정리해보자면
주어진 함수를 다항식의 극한(멱급수)로 표현하는 것을 말한다.
여러번 미분 가능한 함수에 대해 f(x)에 접하는 다항함수로 표현하는 방법이라고 하는데..
'접선'을 통해 함수를 근사하는 선형 근사를 일반화한 다항함수 형태라고 생각하면 이해하기 쉽다는데
일단 이 문제에서는 centered finite difference approximation을 하니깐 오차가 훨씬 적고 정확한 근사를 제공한다고 이해하자.

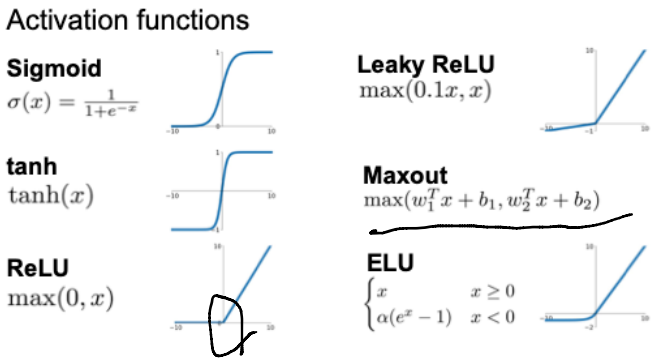
정리하면 Centered .... 를 사용하면 centered 가 아닌 numérica gradient 보다 더 정확하다고 한다.
근데 우리는 오래 걸리는 numerical이 아닌 analytic gradient를 사용하지 않는가?
다시 문제로 돌아가서 kinks를 피하기 위함이라는데 여기서 kinks는 꺾임목이라 ㅎ라는 미분이 불가능한 부분을 일컫는 용어이다.
미분이 불가능한 지역은.. 아래와 같다.



근데 왜 미분불가능 지역을 고려해야할까.??
Optimizer는 아래 그림과 같이 백프롭.. 역전파 과정이라 할 수 있따.
그럼 우리가 레이어를 쌓을때 비선형을 위해 사용했던 Activation Function을 떠올려보자.
이 중에 미분 불가능한 부분은 ReLU함수 (Max(0, x), 서포트벡터머신(SVM), 맥스아웃 뉴런 등을 사용하면 발생한다고 한다.

3. 옵티마이저(Optimizer)





지금 이 문제는 Gradient Check. 다시 말해 우리가 수식으로 계산한 Analytic Gradient로 계산한 값과 수치적 numerical 근사값 과 비교하여 상대 오차를 확인하는 과정에 대한 얘기이다.

즉, Analytic Gradient로 계산한 값이 정확한지 Numerical Gradient의 오차가 최~대한 적은 정확한 값을 통해 비교하여 Analytic Gradient가 제대로 계산이 되었는지 부정확한 계산이 발생하지는 않았는지 확인하는 것이 Gradient Check 이다.
꺾임목에서는 부정확한 계산이 발생할 수 있는데 이를 이 Gradient Check과정에서 확인할 수가 있고 몇가지 계산을 통해 꺾인점을 넘어 계산되었는지 여부를 알수가 있다.
꺾임목을 피하기 위한 해결책은 더 적은 데이터를 쓰는것이다. 활성화함수가 꺾인점을 포함하고 있으면 데이터가 적을수록 더 적은 꺾인점을 포함할 것이고 , 따라서 유한차분근사(finite differenct appromixation) 과정에서 꺽인점을 가로지르는 경우가 더 적을 것이다.

이 문제는 Gradient Checks 에 대한 포괄적인 내용에 대한 이해를 요구하는 문제이다.
Cs231 3,4강에서는 간단하게 수치적 계산 (numerical)과 수식으로 계산한 (analytic) Gradient를 비교하는 정도로만 배웠지만 실제 이 작업을 하기 위해서는 훨씬 복잡하고 뜬금없는 오차(꺾임목 같은..)가 발생하는것이 쉽다는것을 깨달을 수 있다.

문제에 대한 답을 정리하자면
수식(analytic)으로 계산한 Gradient가 정확한지 Numerical gradient 의 값과 비교하려 하는데
Numerical gradient 중에서도 Centered finite difference approximation 을 사용하여 비교하면
좀 더 적은 오차를 가진 값과 비교해볼 수 있으므로 Centered를 권장하고 있다.
이 외 여러가지 Gradient checks를 위한 팁이 있는데 혹 ReLU와 같은 꺾임목이 있는 활성화 함수를 사용했다면
꺾임목 있는 곳을 피하기 위해 적은 데이터만 사용하여 finite different approximation을 사용하여 비교해보면 된다.

꺾임목(kinks)를 피하기 위해 Centered를 사용하는것이 좋은것이 아니라 Gradient Checks에 좀 더 정확한 비교를 하기 위해 Centered finite difference approximation을 사용하는 것이다.