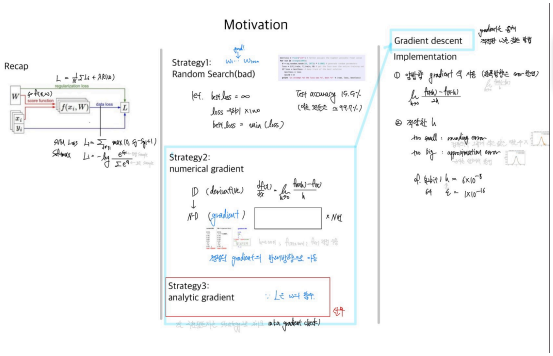
Lec4. back propagation

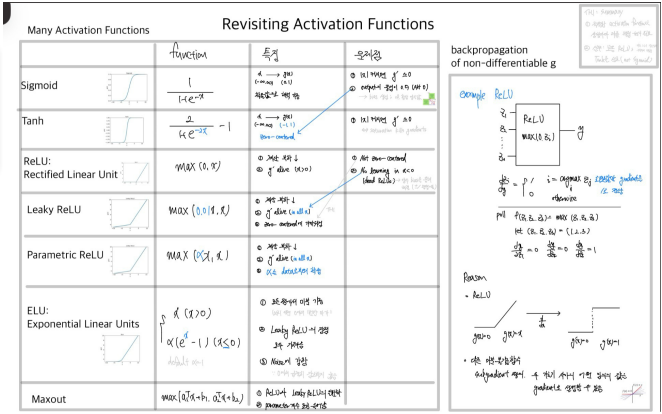
Motivation

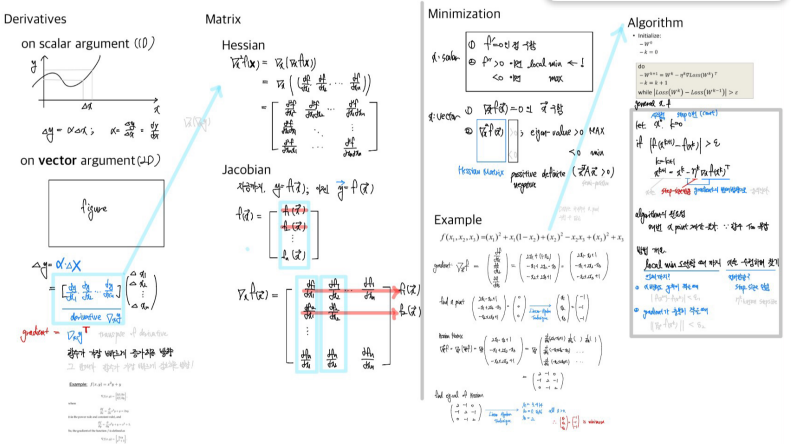
Revisiting activation functions

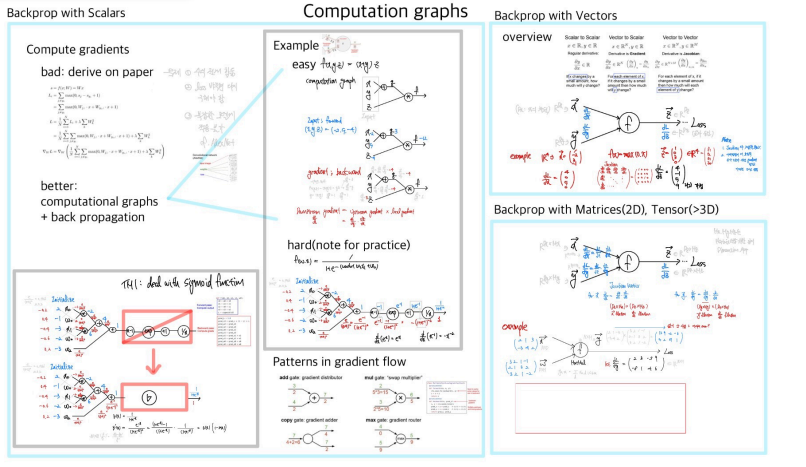
Calculating gradients

Computation graphs





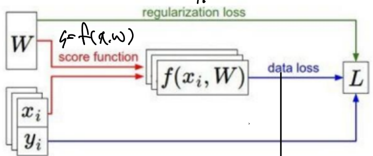




Motivation

Recap

$L = \frac{1}{N}\sum L_i + \lambda R(W)$



SVM Loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Softmax $L_i = -\log \frac{e^{s_{y_i}}}{\sum e^{s_j}}$ ← 정답 Sample, ← 모든 sample

Strategy1: Random Search(bad)

good! ↓ $w_1 \cdots w_{1000}$

```
bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)
```

ref. bestloss = ∞

loss 구하기 ×1000

bestloss = min (loss)

Test accuracy 15.5%

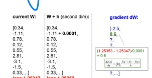
(다른 것들은 ≃ 99.7%)

Strategy2: numerical gradient

1D (derivative) $\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

↓

N-D (gradient) × N번



h=0.0001 ; f(x+0.0001), f(x) 계산 구함

계산된 gradient의 반대방향으로 이동

Strategy3: analytic gradient

∵ L은 w의 함수.

선우

잘 구현됐는지는 strategy2로 체크 (a.k.a. gradient check)

Gradient descent

gradient를 통해 적절한 w을 찾는 방법

Implementation

① 양방향 gradient 식 사용 (한쪽방향은 error 발생)

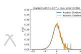
$\lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$

$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

② 적당한 h

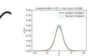
too small : rounding error

컴퓨터가 2개의 작은 값을 다룰 수 X



too big : approximation error

근사를 선에서 발생



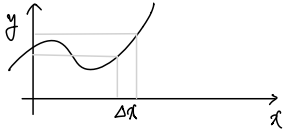
cf. 32bit) $h = 6 \times 10^{-8}$

64 $\epsilon = 1 \times 10^{-16}$

Calculating Gradients

Derivatives

on scalar argument ($\mathbb{R}^1$)



$\Delta y = \alpha \Delta x$; $\alpha = \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$

on **vector** argument ($\mathbb{R}^n$)

$\Delta y = \alpha \cdot \Delta x$

$= \left[\frac{\partial y}{\partial x_1} \; \frac{\partial y}{\partial x_2} \cdots \frac{\partial y}{\partial x_n} \right] \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix}$

derivative $\nabla_{\mathbf{x}} y$

gradient = $\nabla_{\mathbf{x}} y^T$ transpose of derivative

함수가 가장 빠르게 증가하는 방향

그 반대는 함수가 가장 빠르게 감소하는 방향!

Example: $f(x, y) = x^2 y + y$

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

where

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x} x^2 y + y = 2xy$$

(via the power rule and constant rule), and

$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y} x^2 y + y = x^2 + 1.$$

So, the gradient of the function f is defined as

$$\nabla f(x, y) = \begin{bmatrix} 2xy \\ x^2 + 1 \end{bmatrix}.$$

Matrix

Hessian

$\nabla_{\mathbf{x}}^2 f(\mathbf{x}) = \nabla_{\mathbf{x}} (\nabla_{\mathbf{x}} f(\mathbf{x}))$

$= \nabla_{\mathbf{x}} \left(\left(\frac{\partial f}{\partial x_1} \; \frac{\partial f}{\partial x_2} \cdots \frac{\partial f}{\partial x_n} \right) \right)$

$= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \ddots & & \vdots \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$

Jacobian

자유까지. $\mathbf{y} = f(\mathbf{x})$; 여러 $\vec{y} = f(\vec{x})$

$f(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_m(\vec{x}) \end{bmatrix}$

$\nabla_{\mathbf{x}} f(\vec{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & & \\ \vdots & & & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$ → $f_1(\vec{x})$, $f_2(\vec{x})$

Minimization

x: scalar

① $f' = 0$인 점 구함

② $f'' > 0$ 이면 local min ← !

< 0 이면 max

x: vector

① $\nabla_{\vec{x}} f(\vec{x}) = 0$ 인 $\vec{x}$ 구함

② $\nabla_{\vec{x}}^2 f(\vec{x})$; eigen-value > 0 MAX

< 0 min

Hessian Matrix positive definite ($\vec{x}^T A \vec{x} > 0$)

negative

Example

$f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2 x_3 + (x_3)^2 + x_3$

gradient: $\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{pmatrix} = \begin{pmatrix} 2x_1 + (1 - x_2) \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 + 1 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{pmatrix}$

find a point $\begin{pmatrix} 2x_1 - x_2 + 1 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ → Linear Algebra Technique → $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$

Hessian Matrix

$\nabla_{\vec{x}}^2 f = \nabla_{\vec{x}} (\nabla_{\vec{x}} f) = \nabla_{\vec{x}} \begin{pmatrix} 2x_1 - x_2 + 1 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{pmatrix} = \nabla_{\vec{x}} \begin{pmatrix} \frac{\partial}{\partial x_1}(2x_1 - x_2 + 1) & \frac{\partial}{\partial x_2}(\;) & \frac{\partial}{\partial x_3}(\;) \\ \frac{\partial}{\partial x_1}(-x_1 + 2x_2 - x_3) & \cdots & \\ \frac{\partial}{\partial x_1}(-x_2 + 2x_3 + 1) & \cdots & \end{pmatrix}$

$= \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$

find eigval of Hessian

$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$ → Linear Algebra Technique → $\lambda_1 = 3.414$, $\lambda_2 = 0.586$, $\lambda_3 = 2$ all $\lambda > 0$.

$\therefore \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$ is minimum

Algorithm

- Initialize:
  - W^0
  - $k = 0$
- do
  - $W^{k+1} = W^k - \eta^k \nabla Loss(W^k)^T$
  - $k = k + 1$
- while $|Loss(W^k) - Loss(W^{k-1})| > \varepsilon$

general x, f

시작점 step 0번 (start)

let x^0, $k = 0$

if $|f(x^{k+1}) - f(x^k)| > \varepsilon$

$k = k + 1$

$x^{k+1} = x^k - \eta^k \nabla_{\vec{x}} f(x^k)^T$

또는 Step-size만큼 gradient의 반대방향으로 감소한다

algorithm의 한계점

여러 x point 계산 필요: ∵ 함수 too 복잡

방법 개념.

local min 도달할 때까지 → 언제까지? ① 미분값 충분히 작을 때 $|f(x^{k+1}) - f(x^k)| < \varepsilon_1$ ② gradient가 충분히 작을 때 $\|\nabla_{\vec{x}} f(x^k)\| < \varepsilon_2$

x는 수렴하여 찾기 → 어떻게? Step-size 안됨 η^k: 너무 stepsize

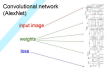
Computation graphs

Backprop with Scalars

Compute gradients

bad: derive on paper

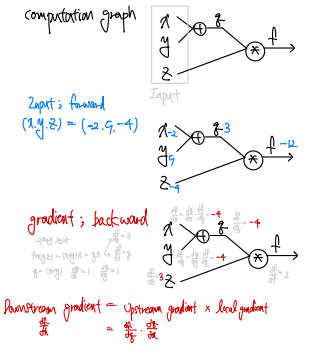
$$s = f(x; W) = Wx$$
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$
$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$
$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$
$$\nabla_W L = \nabla_W \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$



better:
computational graphs
+ back propagation

Example

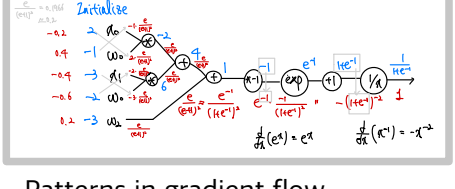
easy $f(x,y,z) = (x+y)z$



Upstream gradient = Upstream gradient × local gradient

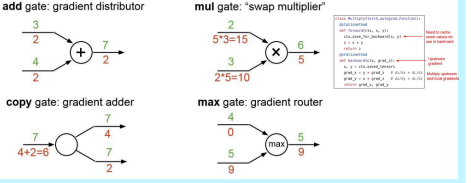
hard(note for practice)

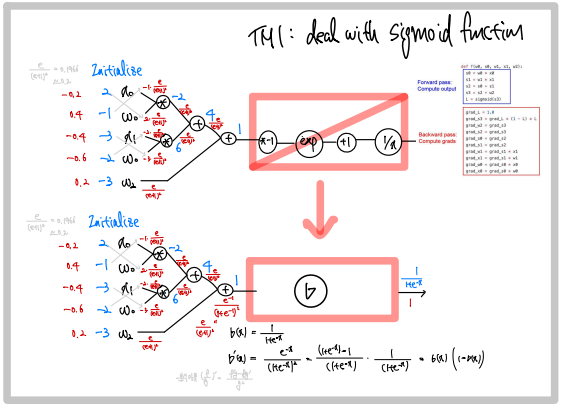
$$f(w,x) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$\frac{d}{dx}(e^x) = e^x$ $\quad \frac{d}{dx}(x^{-1}) = -x^{-2}$

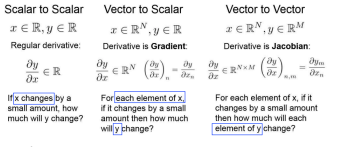
Patterns in gradient flow

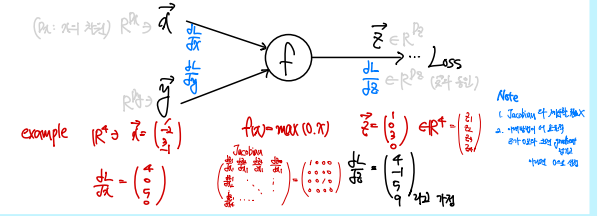




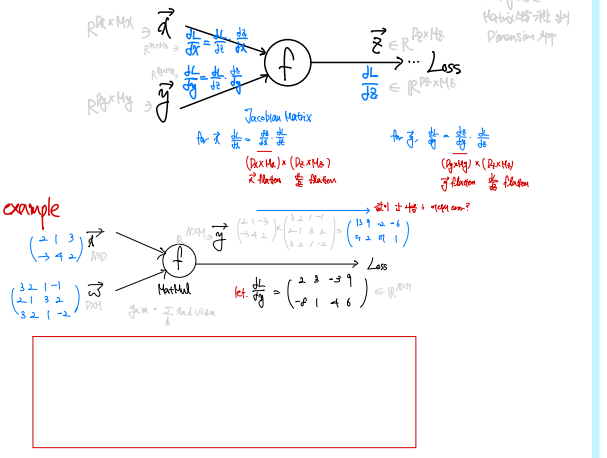
Backprop with Vectors

overview



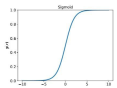
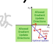
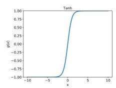
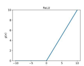
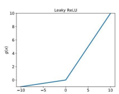
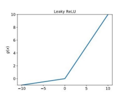
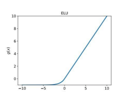


Backprop with Matrices(2D), Tensor(>3D)



Revisiting Activation Functions

Many Activation Functions

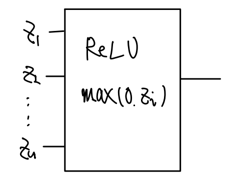
	function	특징	문제점
Sigmoid 	$\frac{1}{1+e^{-x}}$	$x \longrightarrow g(x)$ $(-\infty, \infty)$ $(0, 1)$ 확률값으로 해석 가능	① $\lvert x \rvert$ 커지면 $g' \simeq 0$ ② output의 중심이 0.5 (Not 0) → bias 생김 ; cf. 항상 양수값 
Tanh 	$\frac{2}{1+e^{-2x}} - 1$	$x \longrightarrow g(x)$ $(-\infty, \infty)$ $(-1, 1)$ zero-centered	① $\lvert x \rvert$ 커지면 $g' \simeq 0$ ⇔ saturation kills gradients
ReLU: Rectified Linear Unit 	$\max(0, x)$	① 계산 부하 ↓ ② g' alive $(x>0)$	① Not zero-centered ② No learning in $x<0$ (dead ReLUs) → 양수 bias를 통해 해결 (그래프 평행이동)
Leaky ReLU 	$\max(0.01x, x)$	① 계산 부하 ↓ ② g' alive (in all x) ③ zero-centered에 가까워짐	
Parametric ReLU 	$\max(\alpha x, x)$	① 계산 부하 ↓ ② g' alive (in all x) ③ α는 data로부터 학습	
ELU: Exponential Linear Units 	$\begin{cases} x & (x>0) \\ \alpha(e^{x}-1) & (x \le 0) \end{cases}$ default $\alpha=1$	① 모든 점에서 미분 가능 ($\alpha \neq 1$ 이면 0에서 1번만 미가) ② Leaky ReLU의 장점 모두 가져옴 ③ Noise에 강함 ∵ 0에서 급격히 감소하지 않음	
Maxout	$\max(a_1^T x + b_1, a_2^T x + b_2)$	① ReLU와 Leaky ReLU의 일반화 ② parameter 개수 2배 늘어남	

TMI

TMI : summary
① 적절한 activation function은 상황마다 다름. 직접 test 필요
② 실무 : 보통 ReLU ; 다른 LU는 중간중간 추가하며 사용함
Tanh은 선호 (not Sigmoid)

backpropagation of non-differentiable g

example ReLU



$$\frac{dz_i}{dy} = \begin{cases} 1 & i = \arg\max_j z_j \\ 0 & \text{otherwise} \end{cases}$$

최댓값만 gradient를 1로 전달

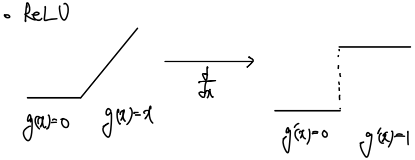
poll $f(z_1, z_2, z_3) = \max(z_1, z_2, z_3)$

let. $(z_1, z_2, z_3) = (1, 2, 3)$

$$\frac{dy}{\partial z_1} = 0 \quad \frac{dy}{\partial z_2} = 0 \quad \frac{dy}{\partial z_3} = 1$$

Reason

- ReLU



- 다른 미분 불가능 함수
  subgradient 정의. 두 기울기 사이의 어떤 임의의 값을 gradient로 설정할 수 있음