Lecture 4: Backpropagation and Neural Networks part 1

박사과정 김성빈 <u>chengbinjin@inha.edu</u>, 지도교수 김학일 교수 <u>hikim@inha.ac.kr</u> 인하대학교 컴퓨터비전 연구실







Recall From Last Time...



Where we are...

$$s = f\left(x, \boldsymbol{W}\right) = \boldsymbol{W}x$$

Score function

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

SVM loss

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_{j} e^{s_j}}\right)$$

Softmax loss

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(\mathbf{W})$$

Data loss + Regularization loss

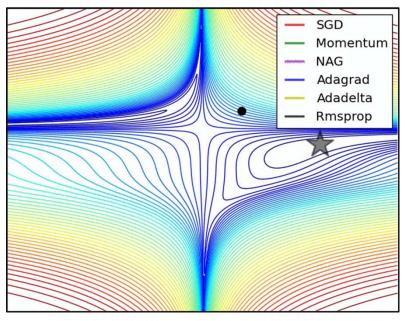
Want
$$\nabla_{\mathbf{W}} L = \dots$$

Recall From Last Time...



Optimization





```
# Vanilla Minibatch Gradient Descent
while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step size * weights grad # perform parameter update
```

Recall From Last Time...



Gradient Descent

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

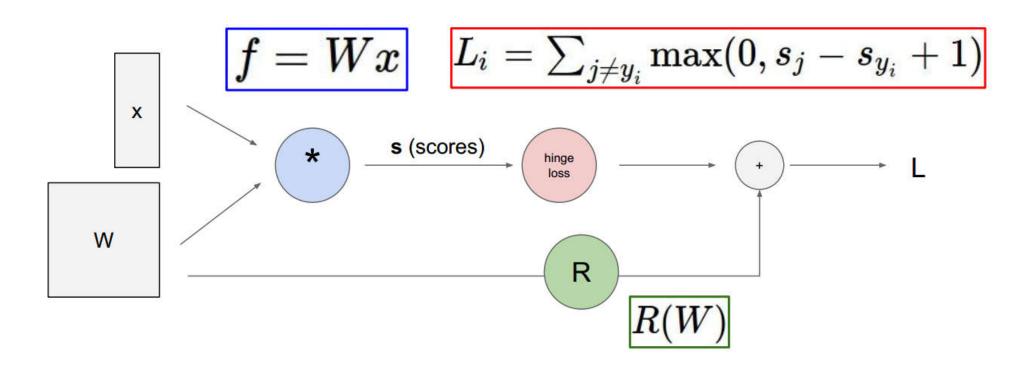
- > Numerical gradient: approximate, slow, easy to write
- > Analytic gradient (calculus): exact, fast, error-prone

> In practice: Derive analytic gradient, check your implementation with numerical gradient



Computational Graph

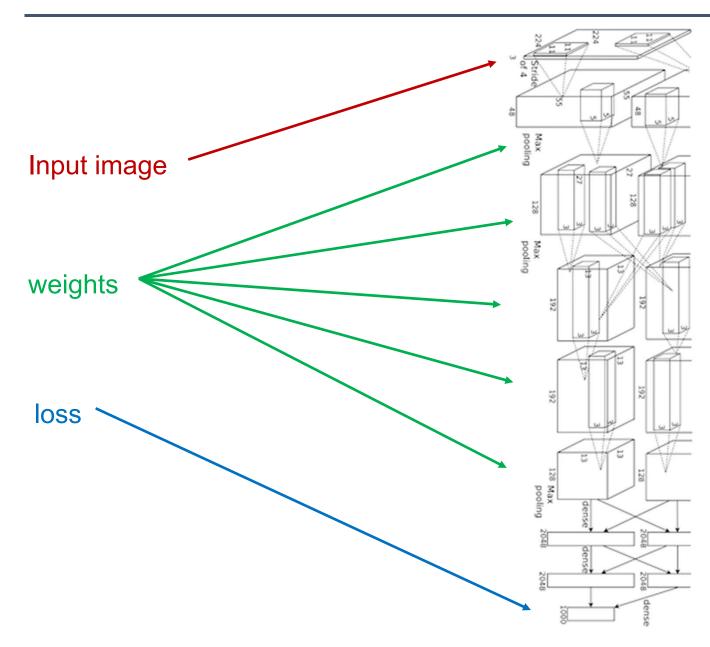




Computation graph of the SVM loss

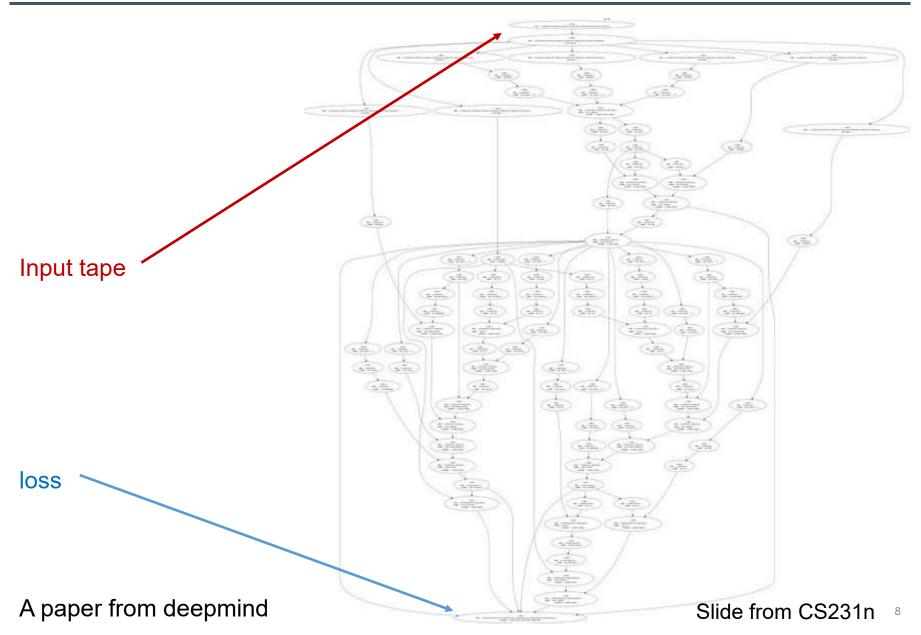
Convolutional Network (AlexNet)



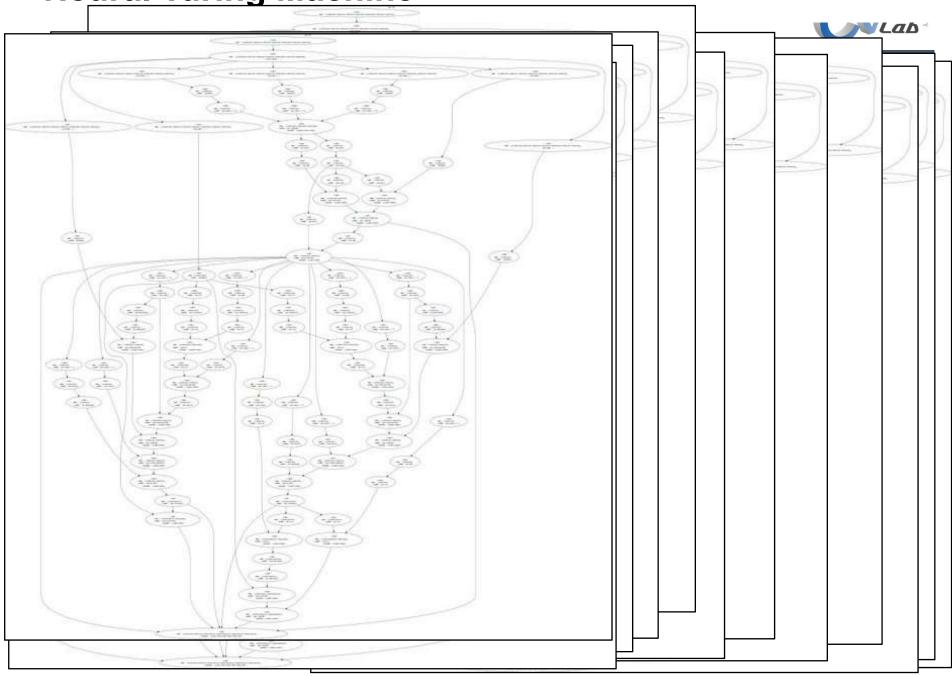


Neural Turing Machine





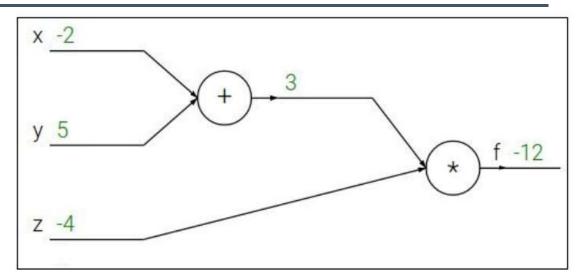
Neural Turing Machine





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

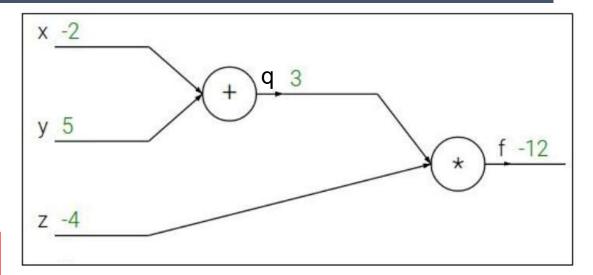
e.g. x = -2, y=5, z = -4





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

e.g. $x = -2$, $y=5$, $z = -4$



$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$

Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

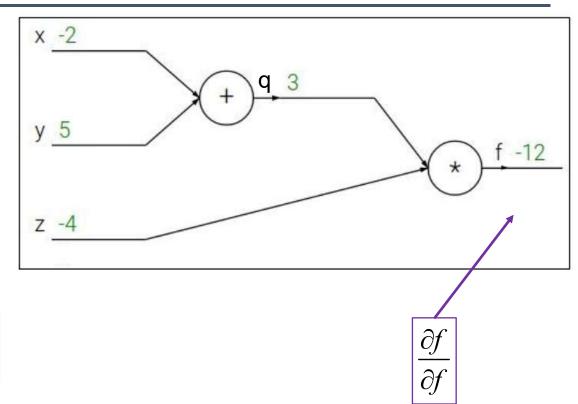


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

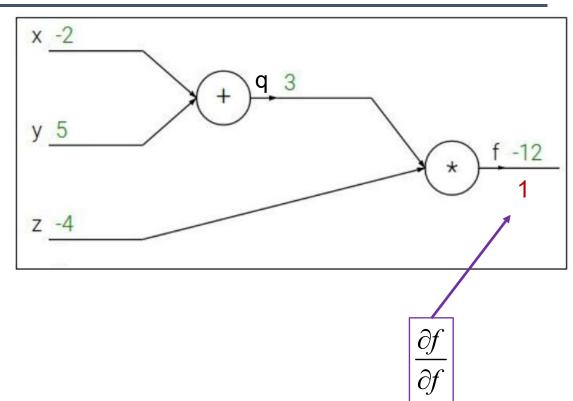


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$

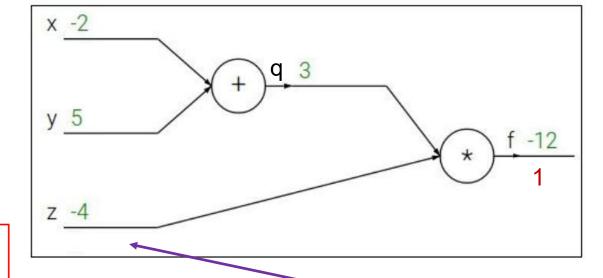


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



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

e.g. $x = -2$, $y=5$, $z = -4$



$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

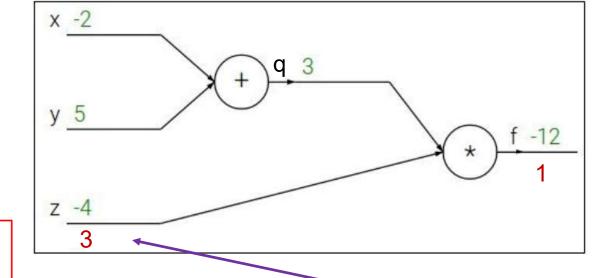
$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$

Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$



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

e.g. $x = -2$, $y=5$, $z = -4$



$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$

Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

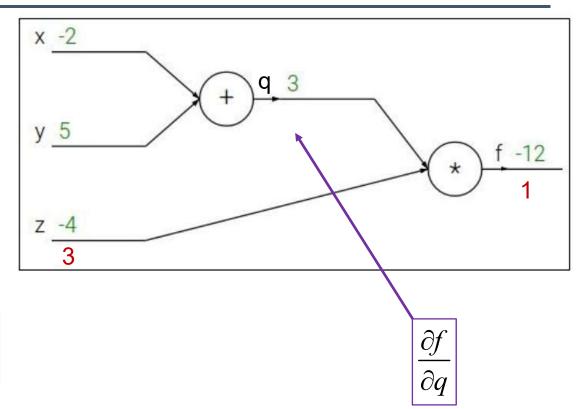


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

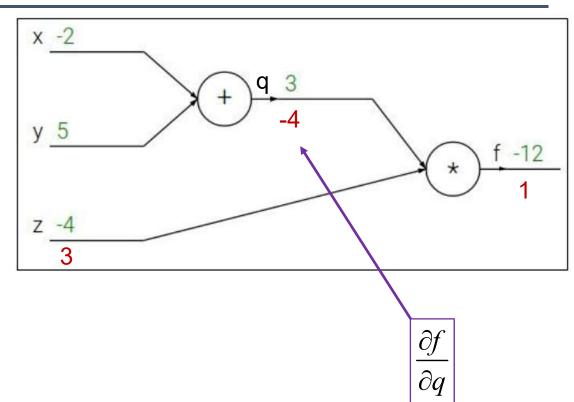


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

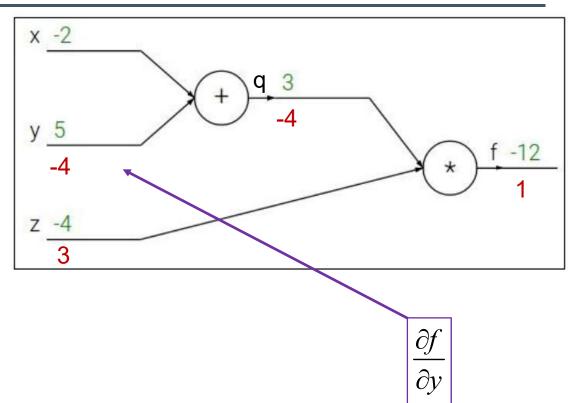


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

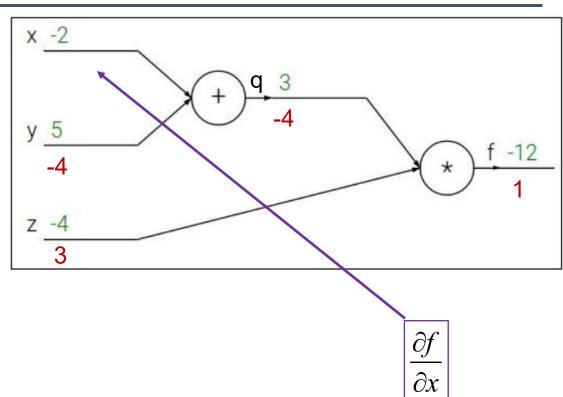


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \qquad rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



Want:
$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

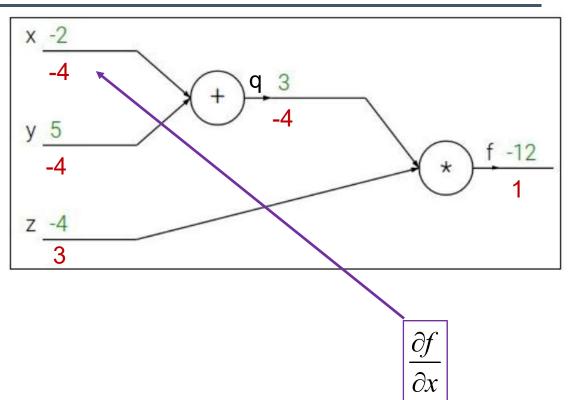


$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y=5$, $z = -4$

$$q=x+y \hspace{0.5cm} rac{\partial q}{\partial x}=1, rac{\partial q}{\partial y}=1$$

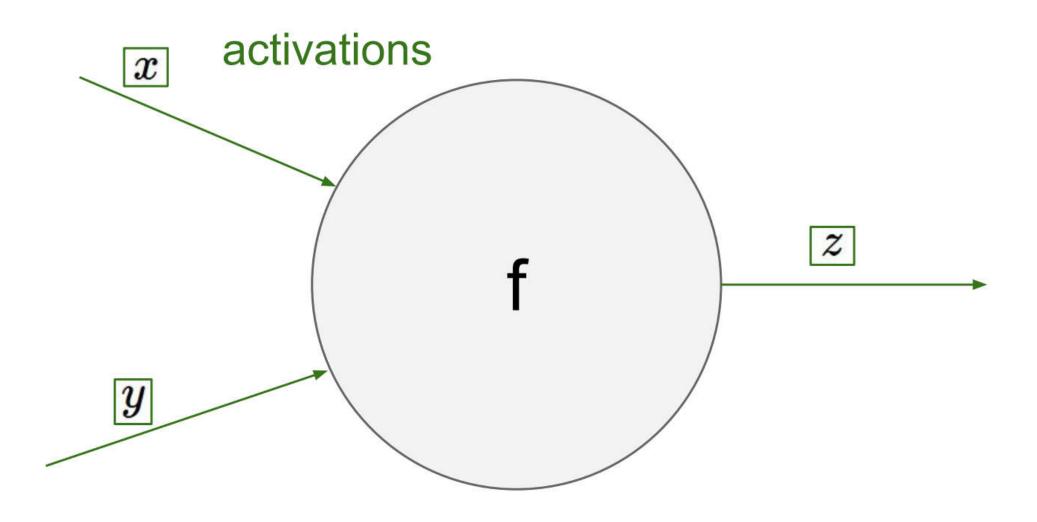
$$f=qz$$
 $rac{\partial f}{\partial q}=z, rac{\partial f}{\partial z}=q$



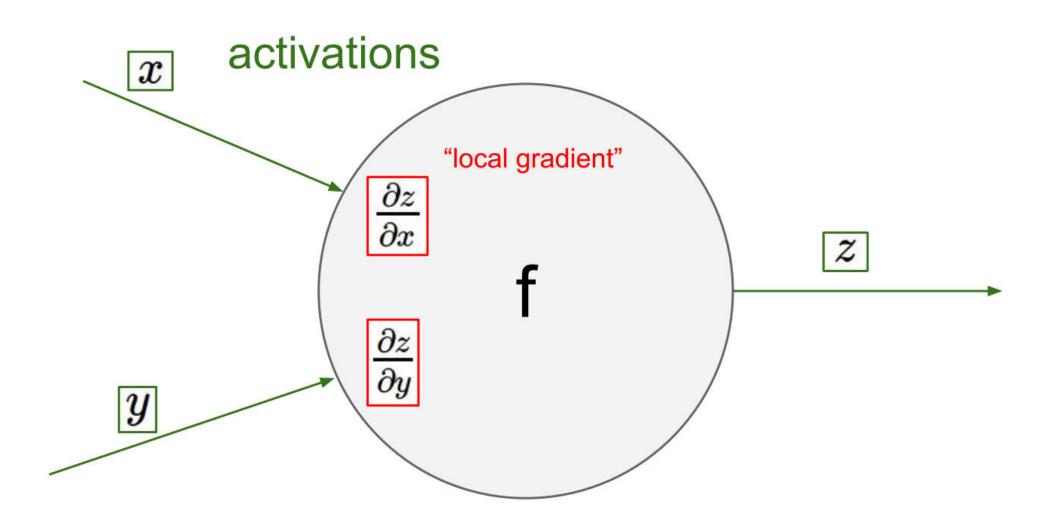
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

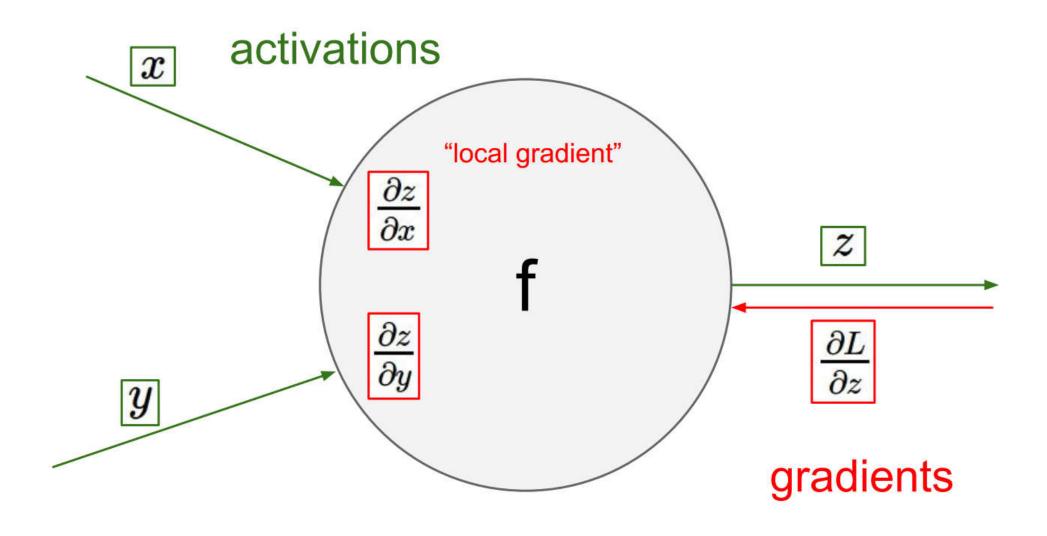




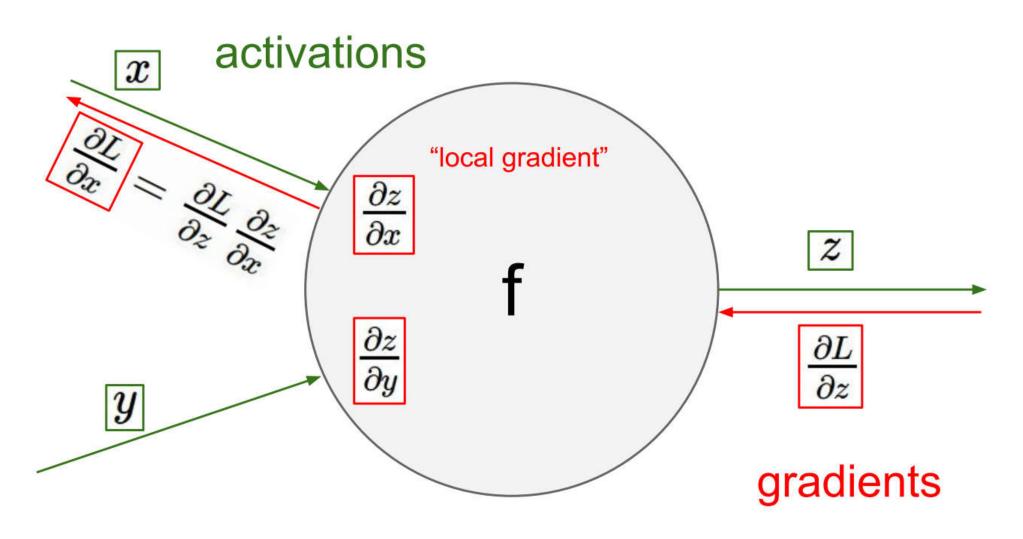






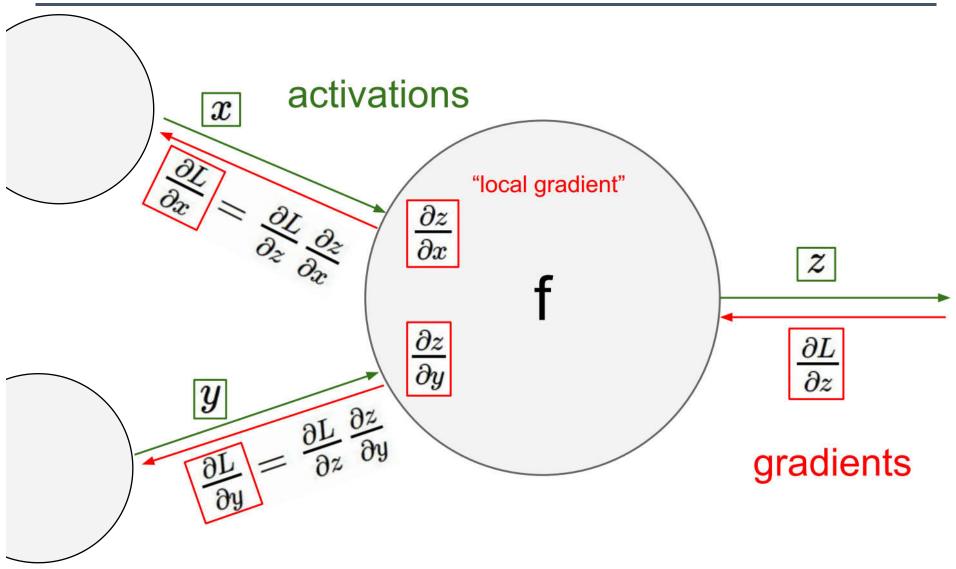






Chain Rule

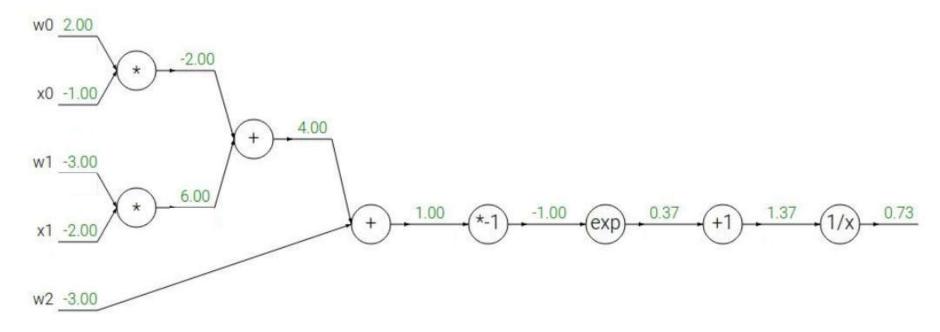




Chain rule



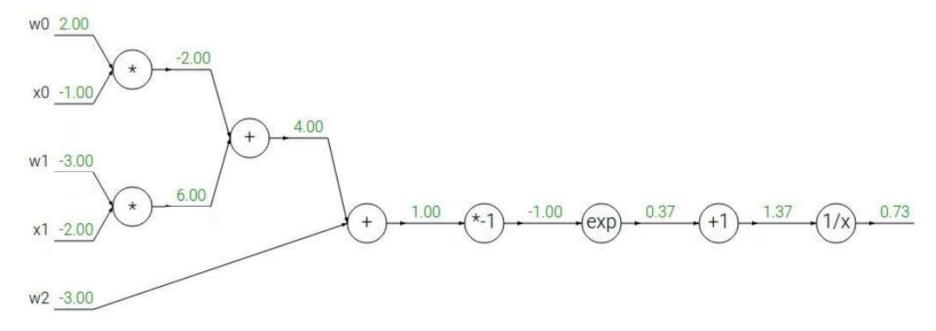
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



Computation graph of sigmoid neuron

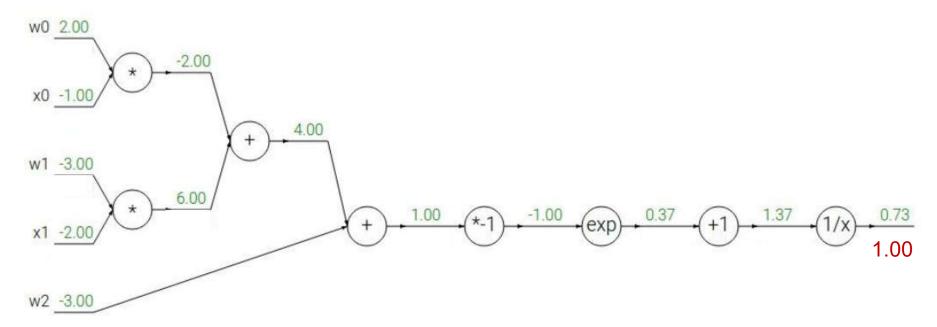


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$





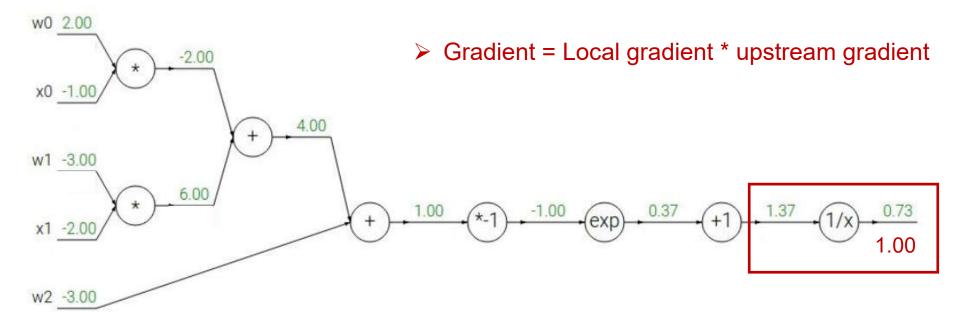
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$egin{aligned} f(x) = e^x &
ightarrow & rac{df}{dx} = e^x & f(x) = rac{1}{x} &
ightarrow & rac{df}{dx} = -1/x^2 \ f_a(x) = ax &
ightarrow & rac{df}{dx} = a & f_c(x) = c + x &
ightarrow & rac{df}{dx} = 1 \end{aligned}$$



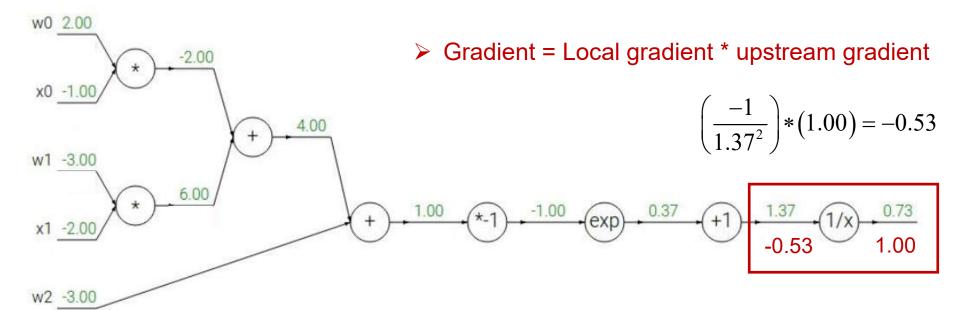
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x)=e^x \hspace{1cm} o \hspace{1cm} rac{df}{dx}=e^x \hspace{1cm} f(x)=rac{1}{x} \hspace{1cm} o \hspace{1cm} rac{df}{dx}=-1/x^2 \hspace{1cm} f_c(x)=c+x \hspace{1cm} o \hspace{1cm} rac{df}{dx}=1$$



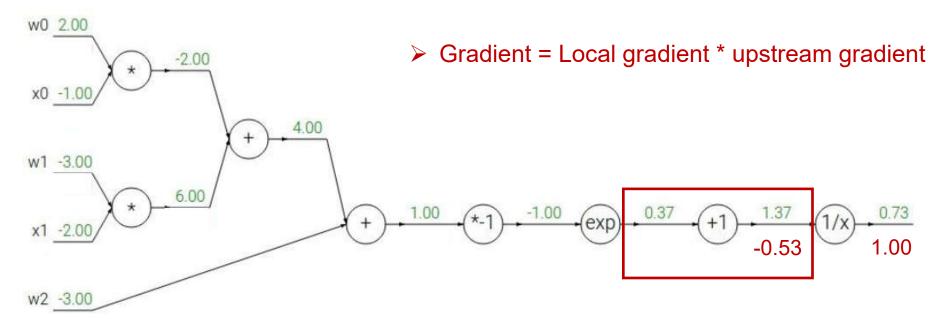
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x) = e^x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = e^x \hspace{1cm} f(x) = rac{1}{x} \hspace{1cm} o \hspace{1cm} rac{df}{dx} = -1/x^2 \hspace{1cm} f_c(x) = c + x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = 1$$



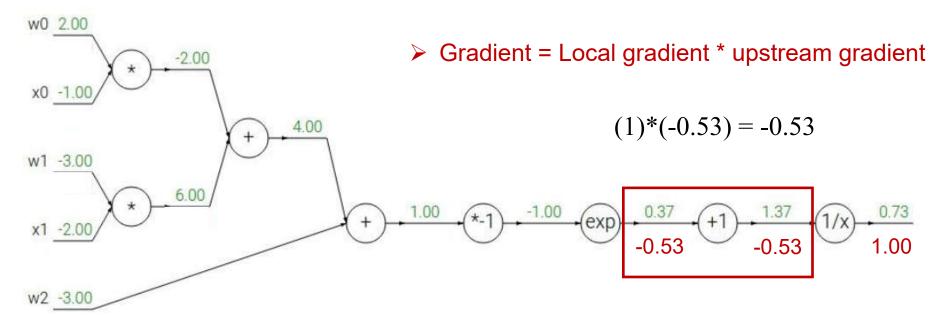
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x) = e^x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = e^x \hspace{1cm} f(x) = rac{1}{x} \hspace{1cm} o \hspace{1cm} rac{df}{dx} = -1/x^2 \hspace{1cm} f_c(x) = c + x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = 1 \hspace{1cm}$$



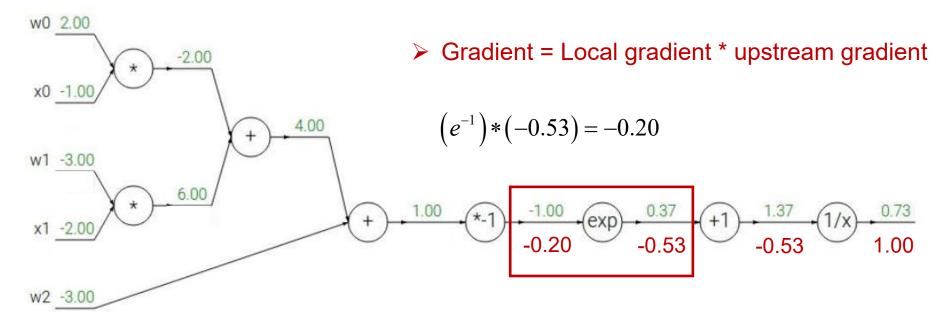
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x) = e^x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = e^x \hspace{1cm} f(x) = rac{1}{x} \hspace{1cm} o \hspace{1cm} rac{df}{dx} = -1/x^2 \hspace{1cm} f_c(x) = c + x \hspace{1cm} o \hspace{1cm} rac{df}{dx} = 1 \hspace{1cm}$$

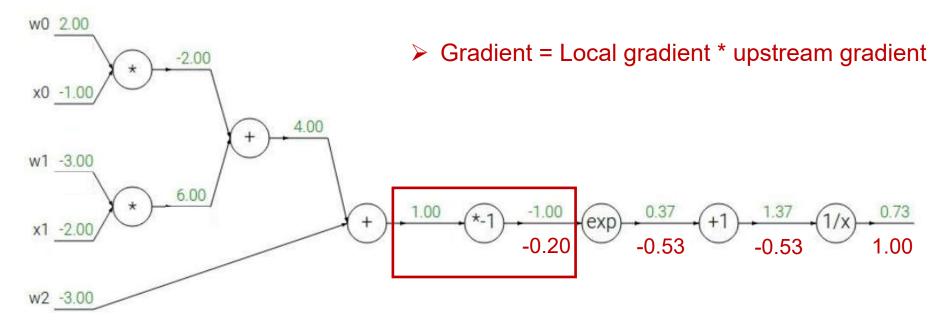


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$





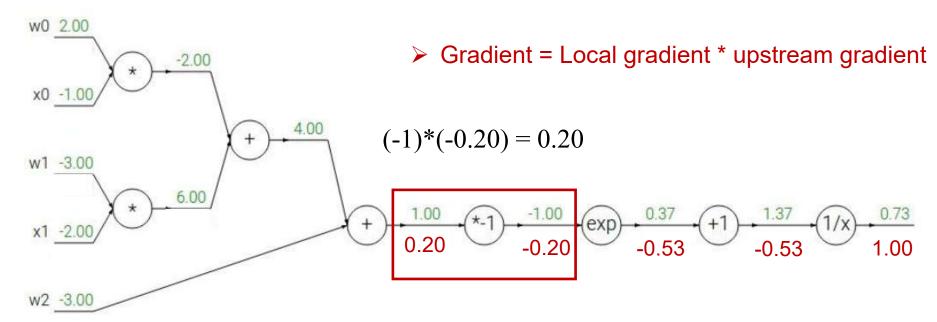
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x)=e^x \qquad o \qquad rac{df}{dx}=e^x \qquad f(x)=rac{1}{x} \qquad o \qquad rac{df}{dx}=-1/x^2 \ f_a(x)=ax \qquad o \qquad rac{df}{dx}=a \qquad f_c(x)=c+x \qquad o \qquad rac{df}{dx}=1$$



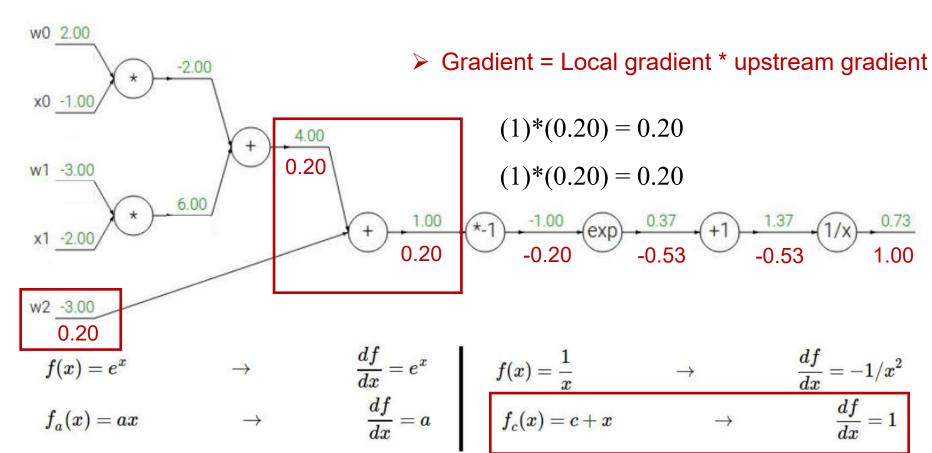
Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x)=e^x \qquad \qquad \qquad \qquad \qquad \qquad rac{df}{dx}=e^x \qquad \qquad f(x)=rac{1}{x} \qquad \qquad
ightarrow \qquad \qquad rac{df}{dx}=-1/x^2 \ f_a(x)=ax \qquad \qquad
ightarrow \qquad \qquad rac{df}{dx}=a \ \qquad \qquad f_c(x)=c+x \qquad \qquad
ightarrow \qquad \qquad rac{df}{dx}=1$$

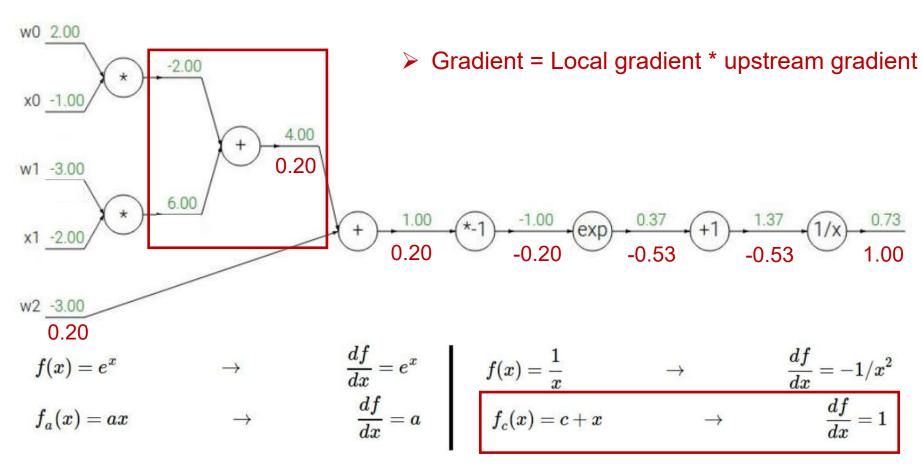


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



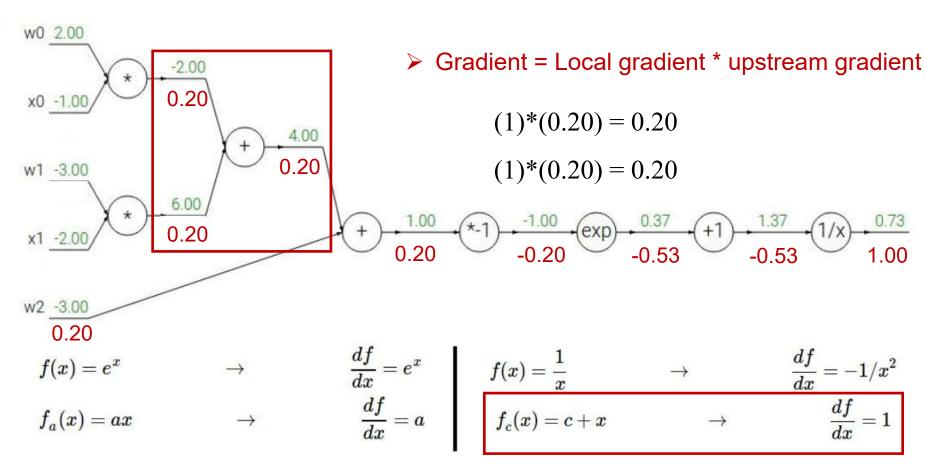


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



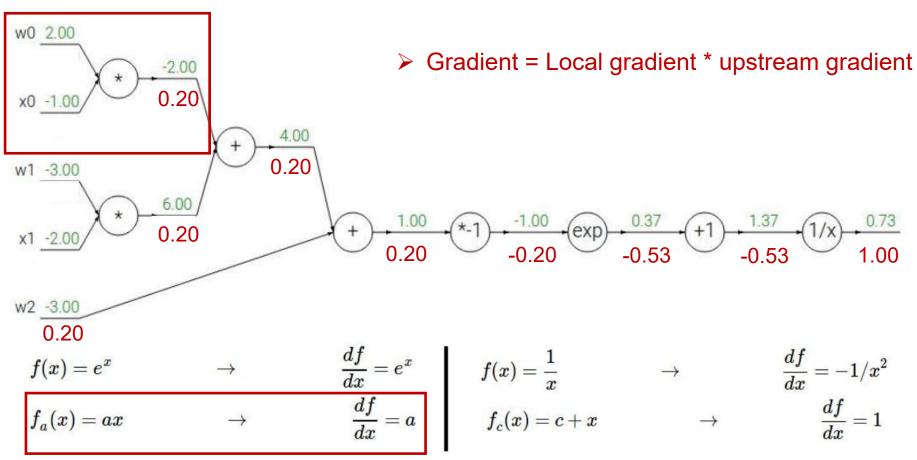


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



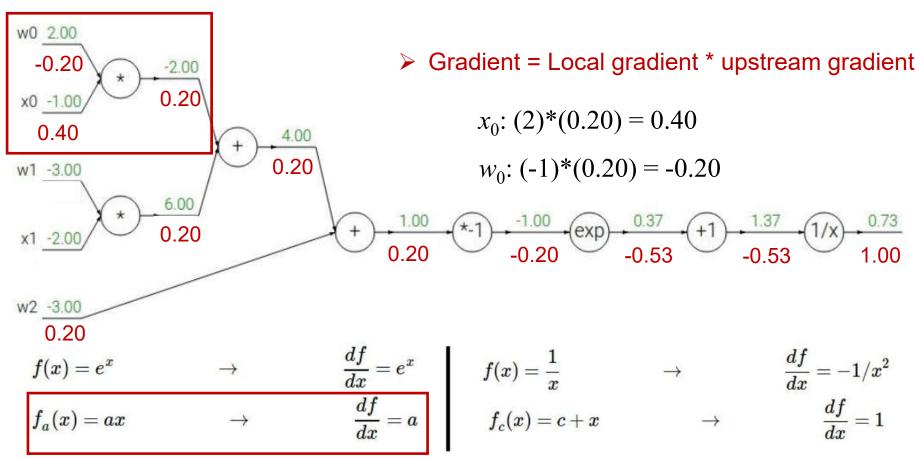


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



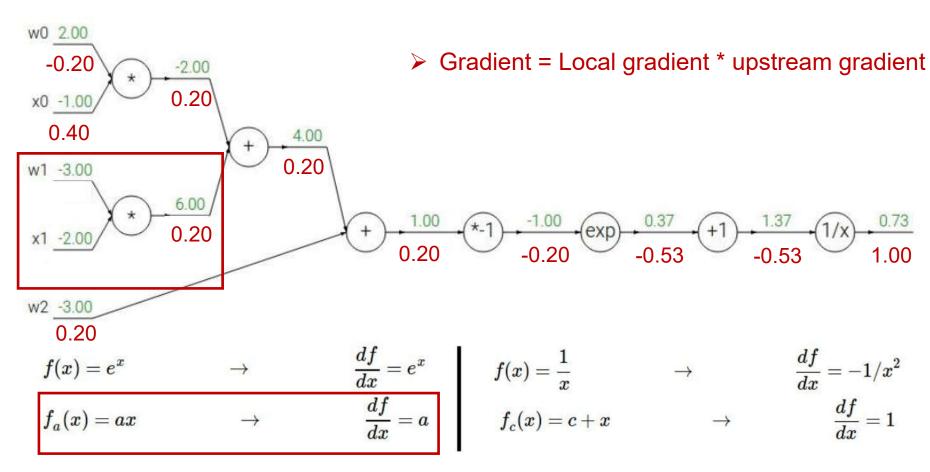


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



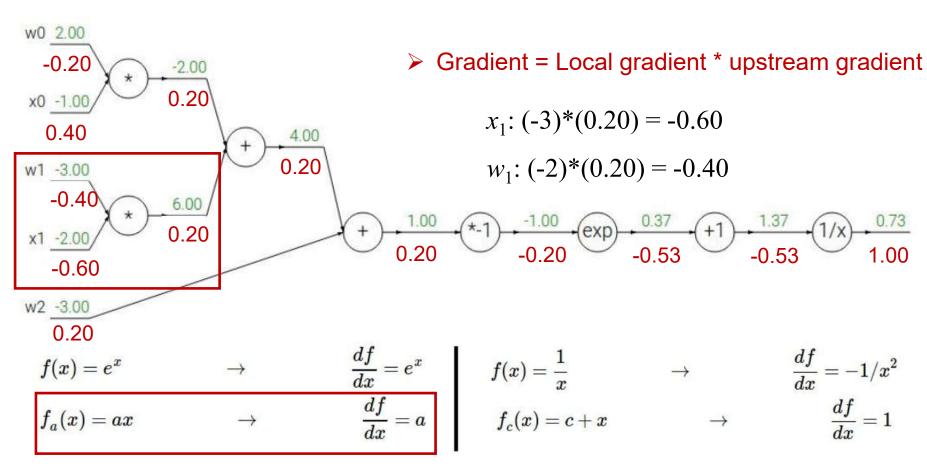


Sigmoid Neuron:
$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$





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



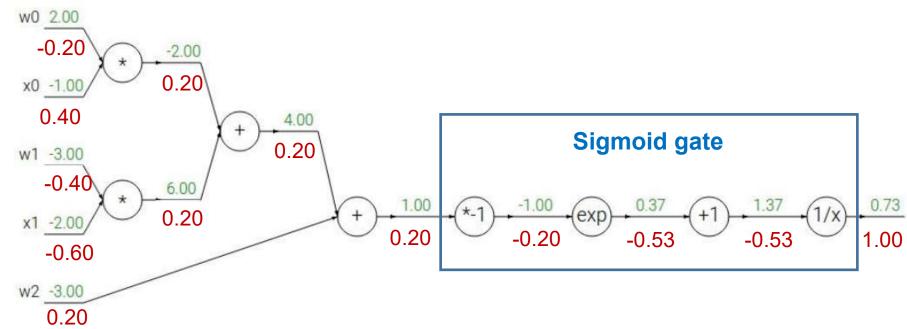
Sigmoid Function



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

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right) \left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



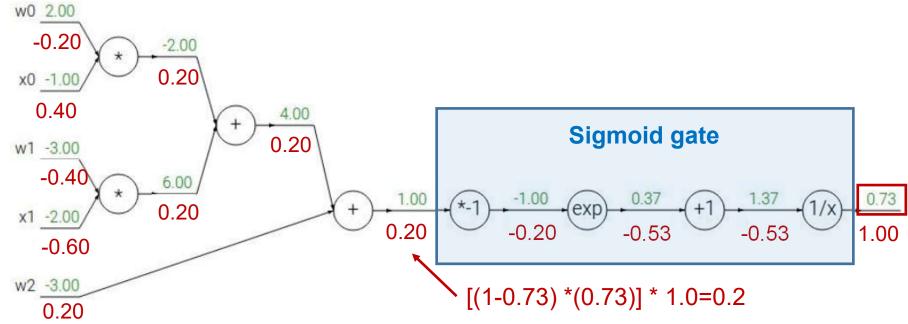
Sigmoid Function



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

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right) \left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$



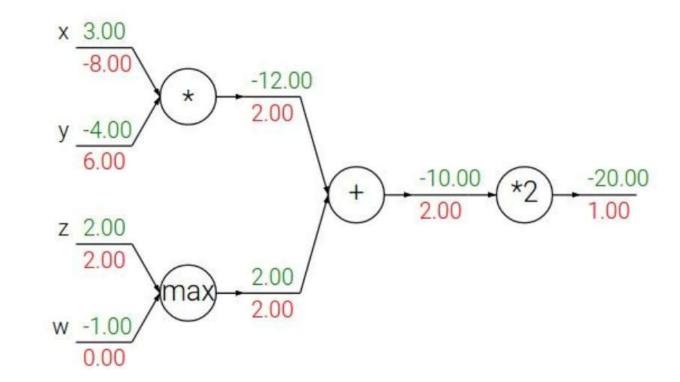
Patterns in Backward Flow



add gate: gradient distributor

max gate: gradient router

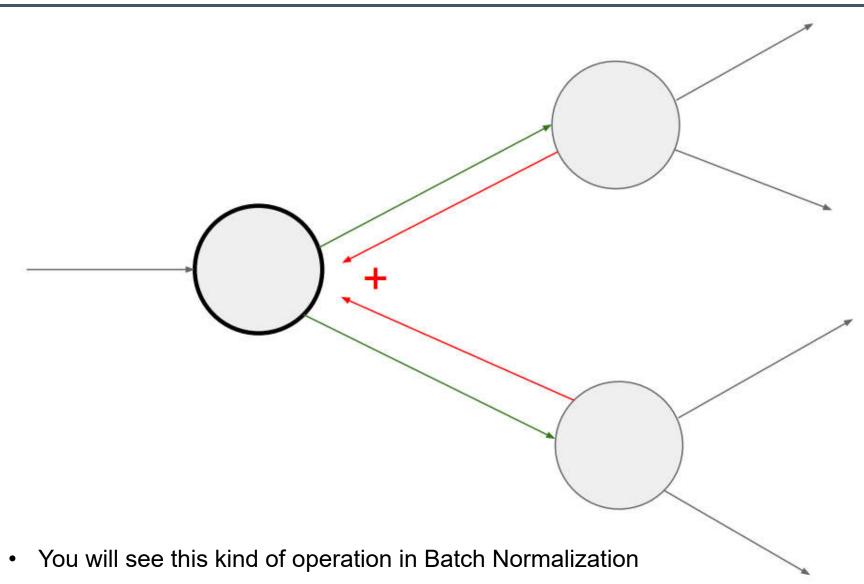
mul gate: gradient "switcher"



max(-1, 2) = 2 $max(-1+\delta, 2) = 2$ $max(-1, 2+\delta) = 2+\delta$

Gradients Add at Branches



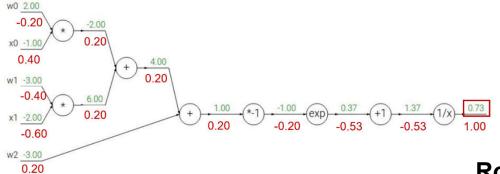




Forward/Backward API

Implementation: Forward/Backward API



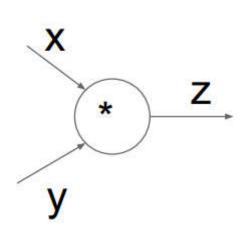


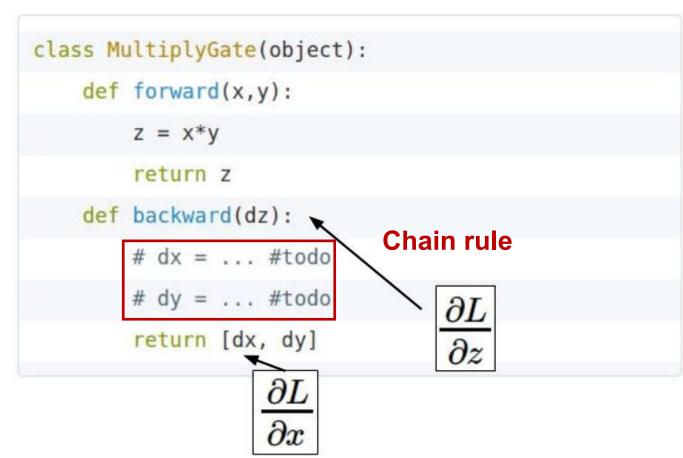
Rough pseudo code

```
class ComputationalGraph(object):
   # . . .
   def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes topologically sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
   def backward():
        for gate in reversed(self.graph.nodes topologically sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs gradients
```

Implementation: Forward/Backward API



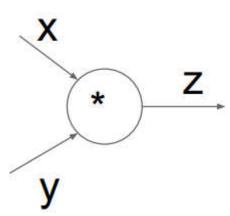




(x, y, z are scalars)

Implementation: Forward/Backward API





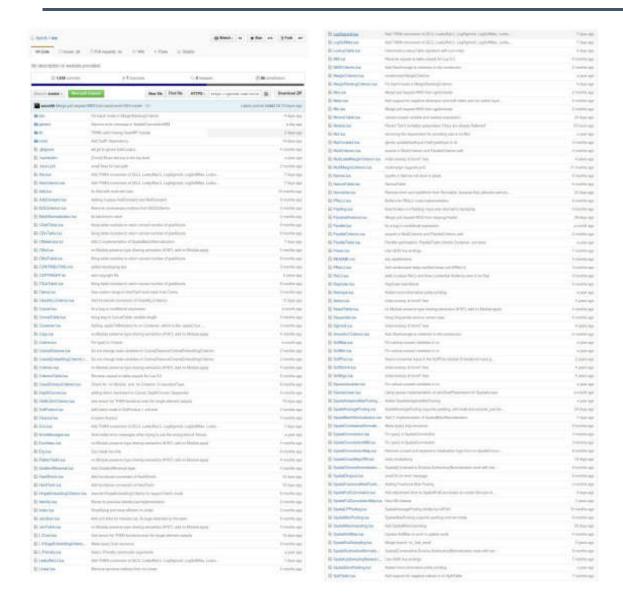
```
class MultiplyGate(object):
   def forward(x,y):
       z = x*y
       self.x = x # must keep these around!
                                                  cache
       self.y = y
       return z
   def backward(dz):
       dx = self.y * dz # [dz/dx * dL/dz]
       dy = self.x * dz # [dz/dy * dL/dz]
       return [dx, dy]
```



(x, y, z are scalars)

Example: Torch Layers





Example: Torch Layers





N substitute	Aut 1978 - Comment of (EV) Complete Conference Conferen	7994
Bughten in	And the contract of \$110 (and \$411) (and \$441) and \$1400 (and \$1400) (and \$140	1700
Si Long here no	THE RESIDENCE OF STREET	1 turn
Di ORENA	Name and Administration of the Edit	100000
the second second	Section of the last of the las	Product
E teachments		1000
S. Bargellerberg Street Land	Name and Address of the Owner, when the Park of the Owner, when the Owner, which the Owne	A Sept 4
S. Decor	And be look Mill to optional	Contra
S description of the last of t	Advanced in colors from a color from an incident	1000
E-Street	Ship of tune 85 to spinore	Contract
E-montance.		JP from to
E-Bress Inc	New York in Nation automore Than an elevan Automat	Title 1 to
S-Maria	months for expension for printing and a triallar	1984
E-Salvenia .	Series and Control of the Parties of the	institute.
E-Salvania.	Search of Mariel Search and Parameters and	Freeton.
S SALAMADON IN	resource of confidence	5000
Si Walley Committee	Address Australia	7 (47)
Ri Floriso Art	Starts of Reliance of State States.	FREE
E recol we be	Sendan	America,
C females to	Assistant accounts for female based his discovering.	25 (64) (6
E Philipson	Selection Trial Contemporations	Shirthy
iii Proteștiii	Section of hits street more being	Since
Territories	they are record that the representation	String is
E Vanda Lor	To a fingle conditional pagestion	and a
E Paris Cristone	Anadric Stationers and Planta Charter and	Same
Si Prodeficie pa	Study principal Psychillaminists Delaine program	1000
Shante	UNITED BY A PROPERTY.	Foreign
Contract or	to service	T-STATE OF
(Maller)	Advantage and property and prop	Transfer 6
The later	with trades that the first control for the control of the	1000
C TopTota for	The latest materials and the latest materials	- Indian
District Inc.	Name and Administration of the Control of the Contr	1000
S Security .	Common at the Common Co	Page 1
Stanfords.	to being a service and the service of the service o	1-000
S Named Street	the bear are as	1-00
N Sprak ju	Secretary Contract	11000
S Proposition in	Astronomy is present a security	1777
E tolking on		
		0.044
E bytto ca	A-14-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1	0.000-0
Elifornia	Professional Anna F. Park (1970) (1974) (1974) (1974)	1100-0
E School Lab	Charles of the Control of the Contro	* (40)
E telepola	Communication of the Communica	
E Name in	Name and Address of the Owner o	1000
C families to:	Strip come representation in particular framework for Survivione	10107
S Special control of the Comp.	Anti-Special Conference of the	1907
S. Santangellering by	Specifical property and the second party and the second party.	20.044.4
-	And represent the section of the sec	Frage
the second	the color represent	Torrier o
The second second	Proper Characteristics	(Company
Name and Address	School Schoolsenson	i-man a
San Company of	Secretaria de la como de destra de Como de Mario.	- Francisco
Santing Street	NAME OF TAXABLE PARTY.	16 Hyris
Sand Street, or other Desiration of the last of the la	September 1 to a lower of the september 1 to 1 t	from the s
SalaDopolie	pulled marriage	l-min-
Special automobile Cont.	Association for Facility	10000
Specific Committee	All all and the A Table of Street As a part from a	Stone
Special Constitution	No. of Concession Conc	
		VIII
Sand Photogram		Transfer .
Specific Policy or	Supplied the Strategy on product year the part of the Strategy	- American
Samulah noong ter	ACTORNOLOGY	Ti for i
Constitution by	Spite Sittle Start in patients	Children
Spinistration by the	Shiprines's to Job and	Specia
	Salad and a Salad	Empelica
C facilities and the sale.	San Control of the Association and the san Association	E-cortos E-cortos







Example: Torch MulConstant



```
local MulConstant, parent = torch.class('nn.MulConstant', 'nn.Module')
    function MulConstant: init(constant scalar,ip)
                                                                                                 f(X) = aX
      parent. init(self)
      assert(type(constant_scalar) == 'number', 'input is not scalar!')
      self.constant scalar = constant scalar
8
      -- default for inplace is false
9
       self.inplace = ip or false
       if (ip and type(ip) -= 'boolean') then
                                                                                         Initialization()
          error('in-place flag must be boolean')
    end
14
    function MulConstant:updateOutput(input)
      if self.inplace then
16
        input:mul(self.constant_scalar)
        self.output = input
18
        self.output:resizeAs(input)
        self.output:copy(input)
                                                                                          Forward()
        self.output:mul(self.constant_scalar)
      return self.output
24
    function MulConstant:updateGradInput(input, gradOutput)
      if self.gradInput then
        if self.inplace then
          gradOutput:mul(self.constant_scalar)
          self.gradInput = gradOutput
                                                                                          Backward()
          -- restore previous input value
          input:div(self.constant_scalar)
34
        else
          self.gradInput:resizeAs(gradOutput)
          self.gradInput:copy(gradOutput)
          self.gradInput:mul(self.constant scalar)
        return self.gradInput
```

Example: Caffe Layers



hour ease. neffe sm/ path / layers /		New Tee Find the 1880er	
L'esperant motors	and the second	And something the section of	
2000,000.00	Server in agent reported	10000	
Section is	Stringer but houses	4.000.00	
Description of the last	description and the second second	11100	
Experience .	dune to last human	4.000.00	
and the last	man time more said.	5500	
Street, Street, Square, Street	danielis jeur jeune).	4.000.00	
Street, Street, Street, or	Serverin Sper Teams	4 10 10 10	
Dally services and	Serverile successions	Treet a	
State over the la	Sunsets later masses	a media	
E made, merce dans con	SHIPS SHE'SHIP!	10000	
State Jennie, Sam in	directly less houses	4 7007-0	
THE RESERVE		1007.0	
N. Sec.	Street has been	17107-0	
and the same of	derivation to the extrements.	1 4 4 4 4 4	
and the same	Street and Street	4.000.00	
	dillostas	4-000-0	
	Annah Karmada	1000	
E-MARKET THE	ad space to 25 blys produces	7300	
E-100 (II)	demand in their beautiful	1007.4	
State on the state	District Controlled	4.000.40	
Contractor (See 1)	PS DESTRUMENTAL SET SCHOOL	1100.0	
Emply, lockers day	Reside sorchality	47000	
School Strategie	Strain Statement	1117.0	
Surry Inches de	desired functionals	0.000000	
- University	State of Spiritual Co.	4,000.4	
Damping areas	Statut de Cale Calenda	1995.4	
non-pany nero	According to the country	4 1000 40	
Color por parties	count be into	1000	
CONTRACTOR OF THE PARTY OF	Security and county	emetra	
and the latest	SOLD SECTION .	4000	
Court, Agreed, Service	BOARD INCOME.	+107.0	
Contractor and	deleta la luer luciona.	A hosting	
Coltania and	design from the control	inter a	
Committee and the		10000	
Service and the	desire and search	1107.0	
Company of	strain to bear	1100.0	
day be to	AND DOCUMENTS	(5.54) (4)	
Error, me to	desired and harms	1000 4	
Print, new part	decided against the	1.000.0	
PRINT BEIN	desert sur-reserv	1 100 4	
Street, Str. (Art 10)	describe the retired	11170	
Defense Japan com	districts and based	4.000-0	
Service Service	State State State S.	4100.4	
and arms	State of the Person	4,000.0	
Employee to	State of the Control	1105.0	
water, but her our	Secretary and regions	1998.0	
	Stratill last raises	4 4 1 1 1	
	Annual territory	1000	
and the last	destable lead hading	1000	
APRIL DE	Scools for heart Scools for heart	17074	
THE RESERVE			

Edward State State Co.	distante sun Telepis	is morning
Emirane spre, m. sc.	Benarie age teatre	a north in
El mort, homoso	Mesons iven teatre.	is month as
Emm, Japan co.	districts have beginned.	2.000.0
Emm, april	districtly law health	action as
Contractor on	phospile layer happen.	aneta
Barra James	dangels over houses	1997
Special Company	Blocards look houses	0.000 N
Daver, net to	Blowth aut Yesses	a nom a
Services as	disharite law heaters.	a some a
Earn, ager to	Binanti Sym Yeales	a north as
Similar density	According to the Control	a residence
Section Services	Altherita layer beasing	a menti q
Employing.	districtly bus headers	a married
Emilyen	moves ov hours	a moon a
Ember 2011	STORES LINE TARRES	a Horitical
Esperit, rest, respected.	Strante Juni heates	0.00000
Expert, ma, may, in.	Sharakita layer Nessero.	2000
Espekt, kemissi	Michelle law Needes	4 1007 4
Singman, June 14	Shimedia layer Seatiers	anen a
Queun, borrow	stangets laser beamers	4 900 4
Since house	duranta layer hapsins	a nom a
Entra John Con	Moneto con terres.	a morth as
Edite Joe de	shipunte Guet Nasterii.	a nom a
Entracement.	Birtistik Son Yessex	s Horth as
Coffra, Service	Annually Superference	is moreth as
Entractor per ou	dimension report headers	a month o
Estimation points	Browle aux resses.	(it this life, a ₀
Earl American	districtly duri beauty	is more as
Earl, low or	thruste are heater.	a mark as
Energe av	distracts (syncholos)	in (Hellich) as
Sein, Name and	stropic suriness	9.999 8
Energies au	durieds (sp) Nation	in Health as
Desiration in	Blourita Sque Newton	a mem u
Special like to	sharquite (por futales)	a work a
Eth_horas	Barnetti operhassira	a Health as
EM, NIPDI	Birnelli lun teoles	a more as
Switcher data lawrence	daturia sent hastes.	A1000 A

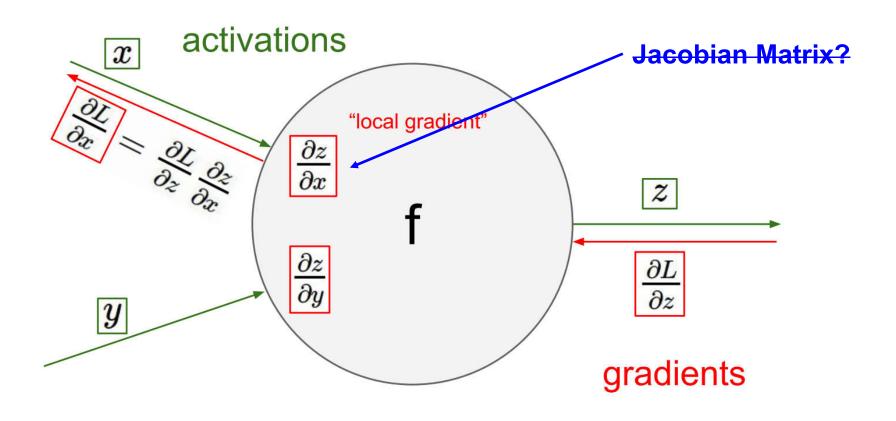
Caffe Sigmoid Layer



```
#include <cmath>
    #include <vector>
    #include "caffe/layers/sigmoid_layer.hpp"
    namespace caffe {
    template <typename Dtype>
    inline Dtype sigmoid(Dtype x) {
     return 1. / (1. + exp(-x));
                                            Forward_cpu
    template <typename Dtype>
  void SigmoidLayer<Dtype>::Forward_cpu(const vector<Blob<Dtype>">& bottom,
        const vector<Blob<Dtype>*>& top) {
      const Dtype* bottom data = bottom[0]->cpu data();
      Dtype* top_data = top[0]->mutable_cpu_data();
      const int count = bottom[0]->count();
      for (int 1 = 0; 1 < count; ++1) {
        top_data[1] = sigmoid(bottom_data[1]);
    template <typename Dtype>
    void SigmoidLayer<Dtype>::Backward_cpu{const vector<Blob<Dtype>*>& top,
        const vector<bool>& propagate_down,
       const vector<Blob<Otype>*>& bottom) { Backward cpu
      if (propagate_down[0]) {
        const Dtype* top_data = top[0]->cpu_data();
        const Dtype* top_diff = top[0]->cpu_diff();
        Dtype* bottom_diff = bottom[0]->mutable_cpu_diff();
                                                                                                (1-\sigma(x))\sigma(x) *top_diff (chain rule)
        const int count = bottom[0]->count();
        for (int i = 0; i < count; ++i) {
         const Dtype sigmoid_x = top_data[i];
          bottom_diff[i] = top_diff[i] * sigmoid_x * (1. - sigmoid_x);
36
    #1fdef CPU ONLY
    STUB_GPU(SigmoidLayer);
    #endif
    INSTANTIATE_CLASS(SigmoidLayer);
47 } // namespace caffe
```

Gradients for Vectorized Code

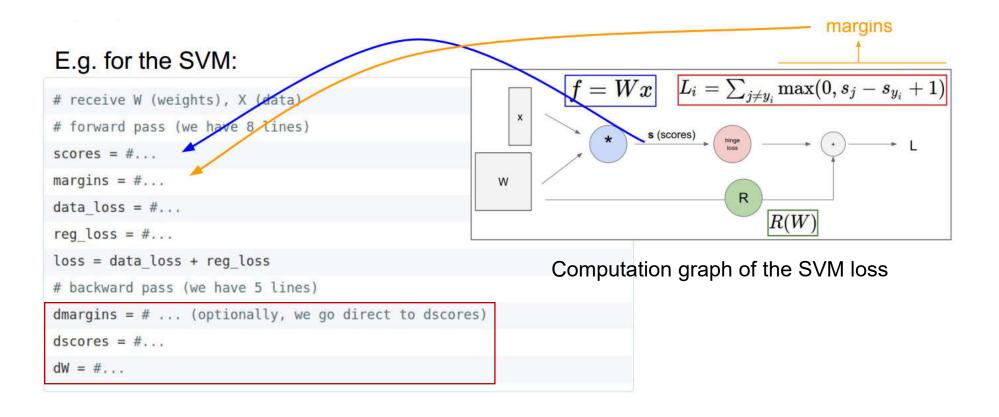




Assignment #1(Q2,Q3): Writing SVM/Softmax



Stage your forward/backward computation!



Summary so Far



- Neural nets will be very large: no hope of writing down gradient formula by hand for all parameters
- **Backpropagation** = recursive application of the <u>chain rule</u> along a computational graph to compute the gradients of all inputs/parameters/intermediates
- Implementations maintain a graph structure, where the nodes implement the forward() / backward() API
- Forward: compute result of an operation and save any intermediates needed for gradient computation in memory
- **Backward:** apply the <u>chain rule</u> to compute the gradient of the loss function with respect to the inputs.

Neural Network





Neural Network: without the Brain Stuff



(Before) Linear score function: f = Wx

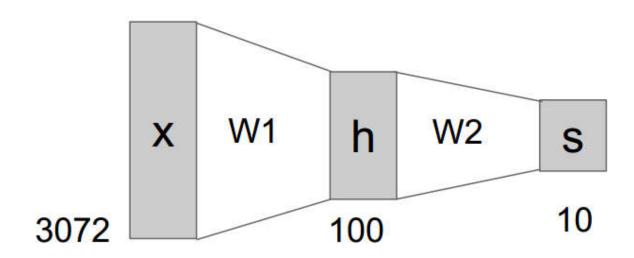
Neural Network: without the Brain Stuff



(Before) Linear score function: f = Wx

(Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

more complex mathematic expression of X



Neural Network: without the Brain Stuff



(Before) Linear score function: f = Wx

3072

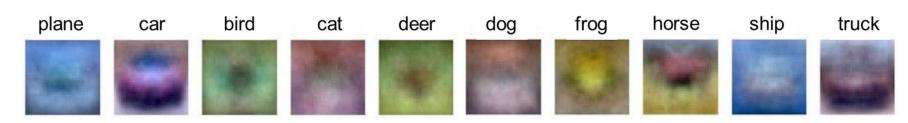
(Now) **2-layer Neural Network** $f = W_2 \max(0, W_1 x)$

$$f = W_2 \max(0, W_1 x)$$

more complex mathematic expression of X

➤ h is a hyperparameter, as big as possible fits in your computer

mode W1 W2 X h S 10 100



Neural Network: Without the Brain Stuff



(Before) Linear score function: f = Wx

more complex mathematic (Now) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$ expression of X

or **3-layer Neural Network**

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Python Code for 2-layer Neural Network



• Full implementation of training a 2-layer Neural Network needs ~11 lines:

```
01.
      X = \text{np.array}([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
02.
      y = np.array([[0,1,1,0]]).T
      syn0 = 2*np.random.random((3,4)) - 1
03.
      syn1 = 2*np.random.random((4,1)) - 1
04.
      for j in xrange(60000):
05.
                                                     Loss = (y - y_predict)
06.
          l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
          12 = 1/(1+np.exp(-(np.dot(l1,syn1))))
07.
          12 \text{ delta} = (y - 12)*(12*(1-12))
08.
          l1 delta = l2 delta.dot(syn1.T) * (l1 * (1-l1))
09.
10.
          syn1 += l1.T.dot(l2 delta)
          syn0 += X.T.dot(l1 delta)
11.
```

- Sigmoid activation function
- Logistic regression loss

Assignment: Writing 2-Layer Net



Stage your forward/backward computation!

```
# receive W1,W2,b1,b2 (weights/biases), X (data)
# forward pass:
h1 = #... function of X,W1,b1
scores = #... function of h1,W2,b2
loss = #... (several lines of code to evaluate Softmax loss)
# backward pass:
dscores = #...
dh1, dW2, db2 = #...
dW1, db1 = #...
                                       X
                                            W1
                                                          W2
                                                                   S
                                                                   10
                                                   100
                               3072
```

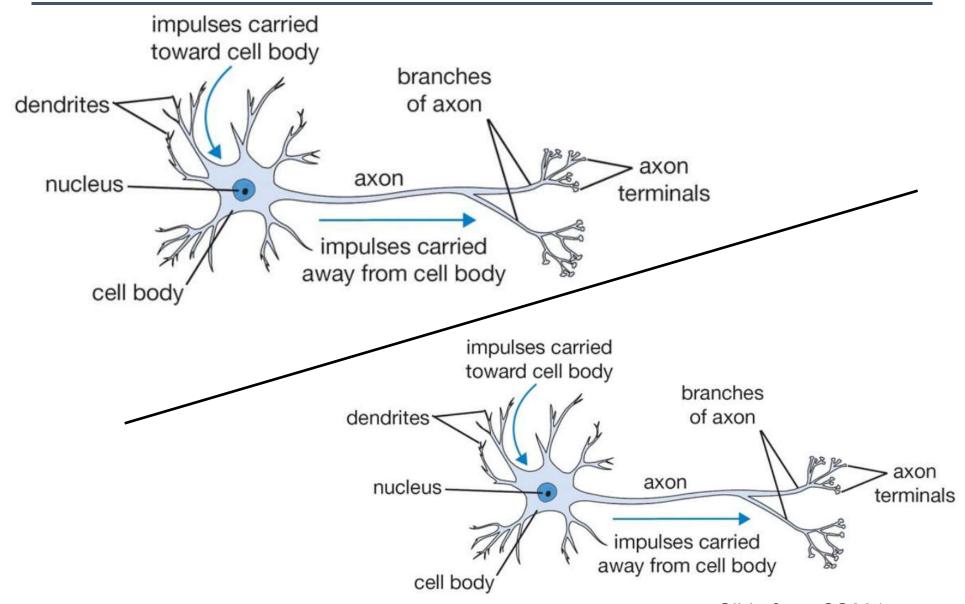
Neural Network Without Brain Stuff





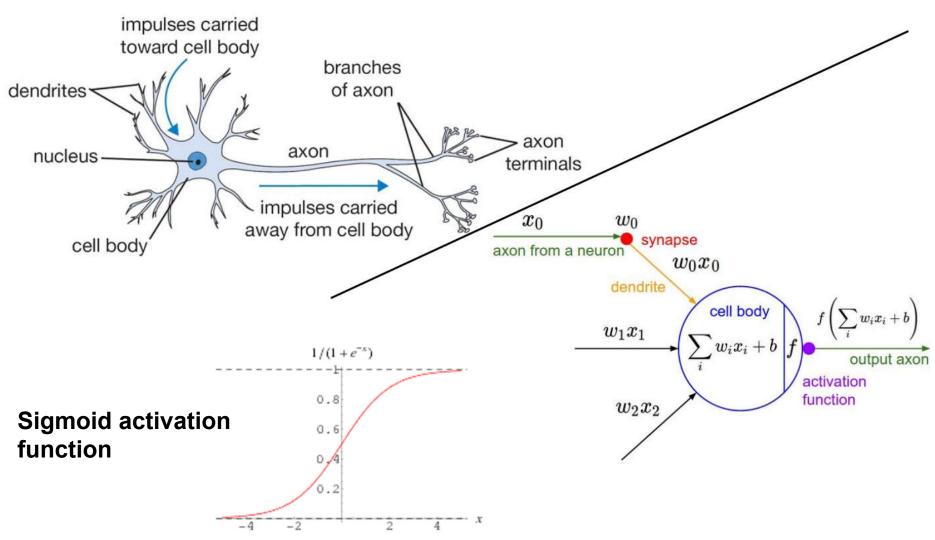
Biological Neuron Without Brain Stuff





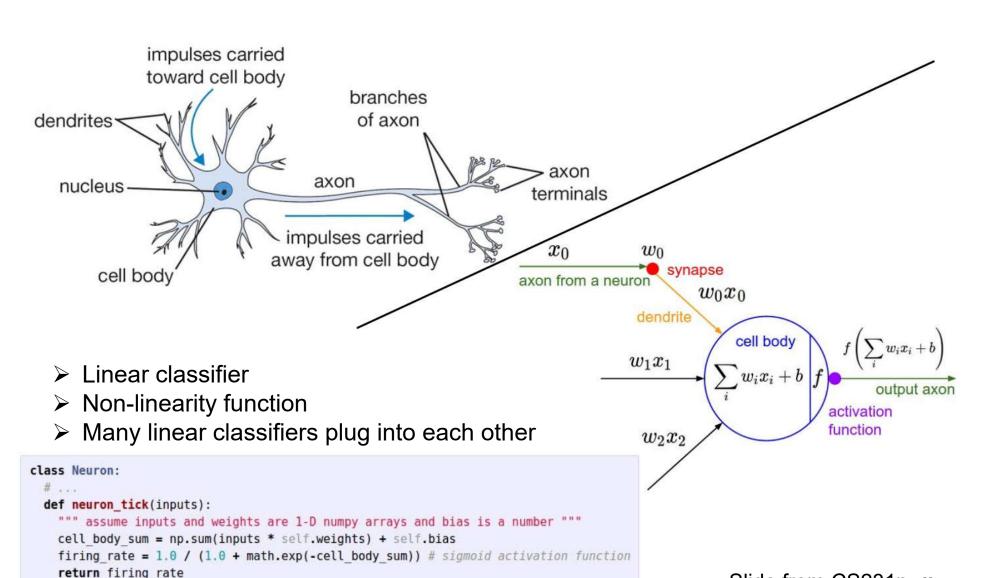
Biological Neuron Without Brain Stuff





Biological Neuron Without Brain Stuff





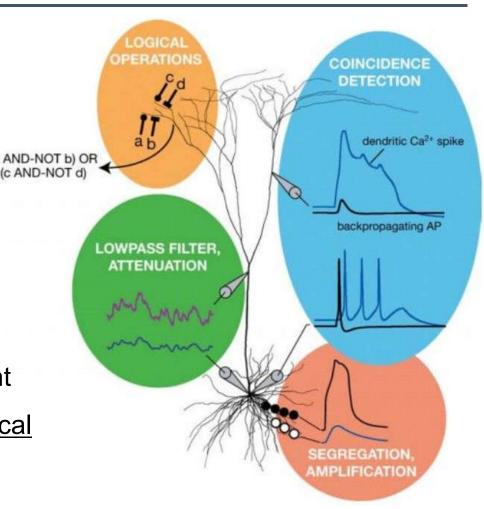
Biological Neuron



Be very careful with your Brain analogies:

Biological Neurons:

- Many different types of neurons
- **Dendrites** can perform <u>complex</u> non-linear computations
- > Synapses are not a single weight but a complex non-linear dynamical <u>system</u>
- > Rate code may not be adequate



Activation Functions



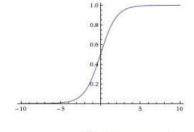
- Historically people use Sigmoid and tanh
- From 2012 **ReLu** became quite popular (default recommendation)
- A few kind of hipster activation functions: Leaky Relu, Maxout, and ELU

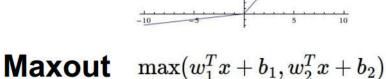
Question #2: why we need activation function?

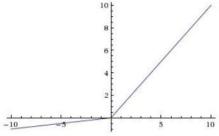
Leaky ReLU max(0.1x, x)

Sigmoid

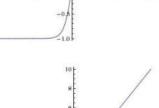
$$\sigma(x)=1/(1+e^{-x})$$







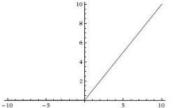
tanh(x) tanh

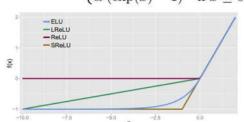


ELU

$$f(x) = \begin{cases} x & \text{if } x > 0\\ \alpha \left(\exp(x) - 1 \right) & \text{if } x \le 0 \end{cases}$$

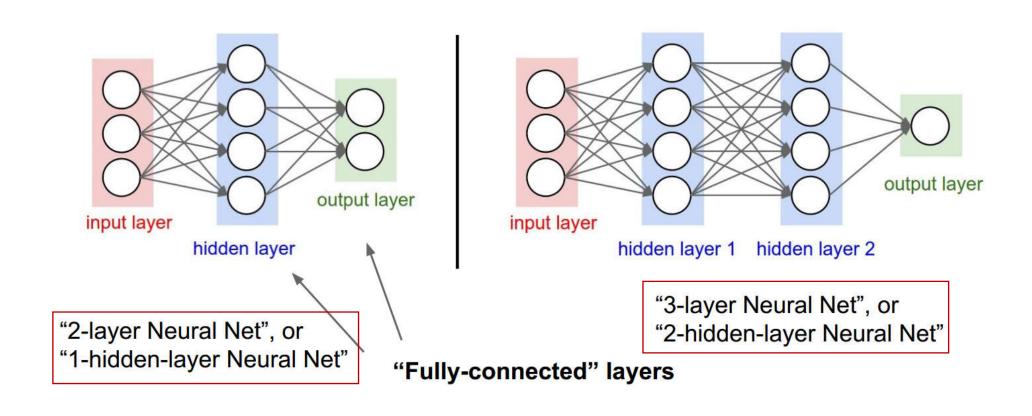
ReLU max(0,x)





Neural Networks: Architectures





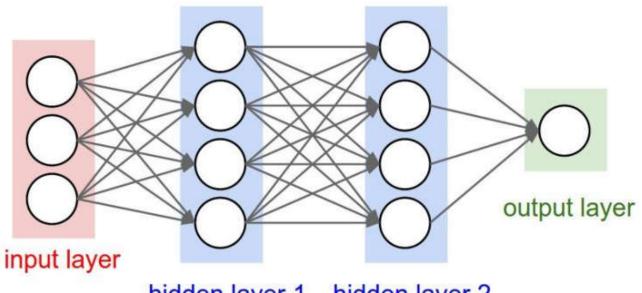
Example Feed-Forward Computation of a Neural Network

Assume inputs and weights are 1-D numpy arrays and bias is a number

```
class Neuron:
 def neuron_tick(inputs):
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
   cell body sum = np.sum(inputs * self.weights) + self.bias
   firing rate = 1.0 / (1.0 + math.exp(-cell body sum)) # sigmoid activation function
    return firing rate
```

Example Feed-Forward Computation of a Neural Network

 We can compute an entire set of neurons in a single hidden layer using a single time matrix multiplication



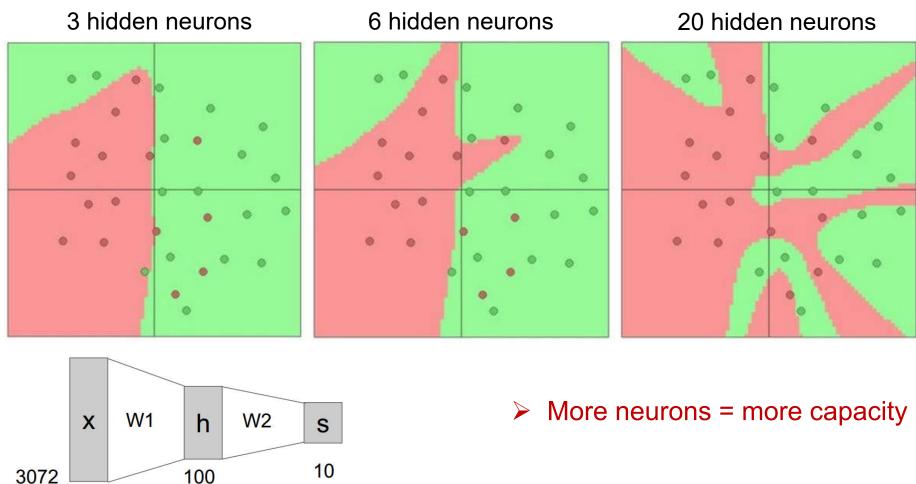
hidden layer 1 hidden layer 2

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
Slide from CS231n 74
```

Example of 2-Layer Neural Network



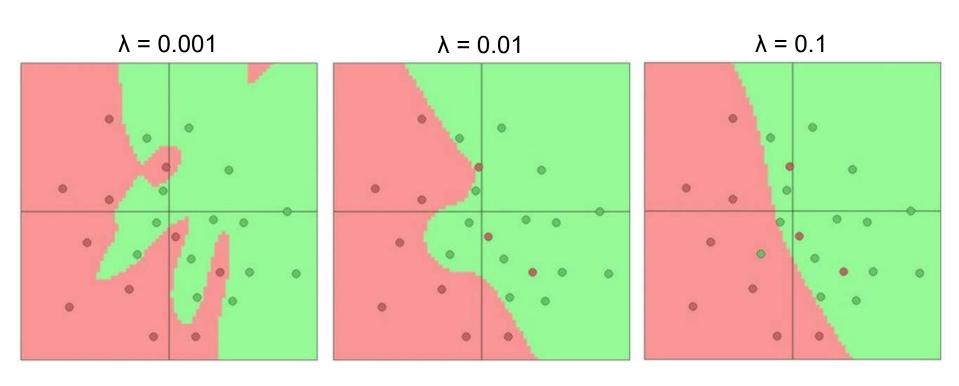
- Setting the **number of neurons** in the hidden layer
- h is a hyperparameter



Example of 2-Layer Neural Network



Do not use size of neural network as a regularizer. Use stronger regularization instead



$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(\mathbf{W})$$

You can play with this demo over at ConvNetJs:

http://cs.Stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Summary



- We arrange neurons into fully-connected layers
- The abstraction of a layer has the nice property that it allows us to use efficient vectorized code (e.g. matrix multiplies)
- Neural networks are **not really neural**
- Neural networks: **bigger = better** (but might have to regularize more strongly)

Next Lecture



More than you ever wanted to know about Neural Networks and how to train them.



Thank you for your attention!