Machine Learning
Lecture 19 (Week 10)

Agenda

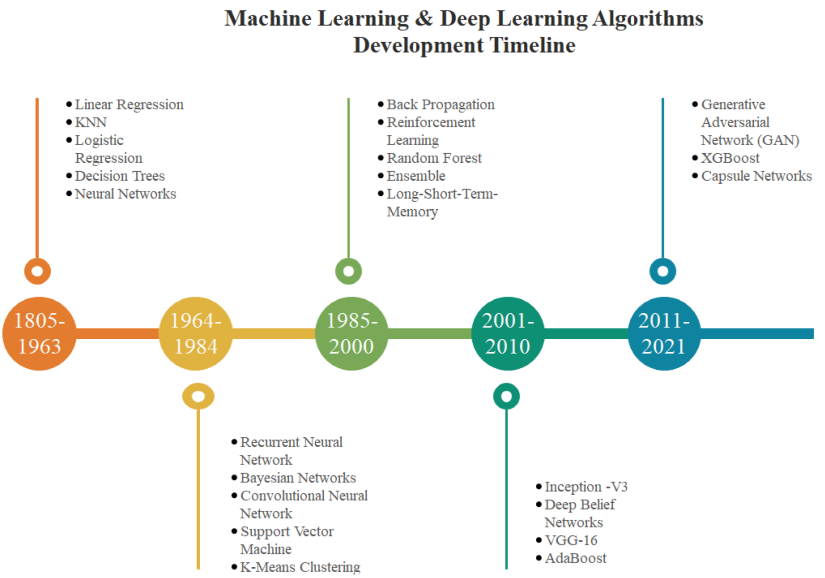
(Artifitial) Neural Networks

- Automated Differentiation
  - Computational Graphs
  - Forward, Backward, and Cross Modes
- Multilayer Perceptron (MLP)

ASIRRA

Source: `https://martin-thoma.com/captcha/`

Timeline



Source: `https://www.mdpi.com/2227-9032/10/3/541`

Neural Networks, Vol. 2, pp. 359–366, 1989
0893-6080/89 $3.00 + .00

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE

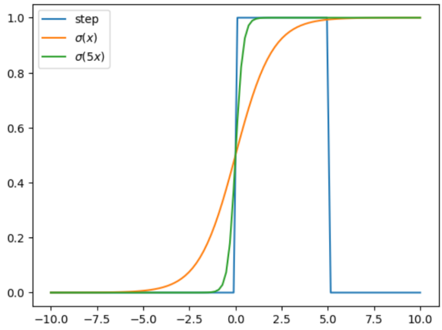
University of California, San Diego

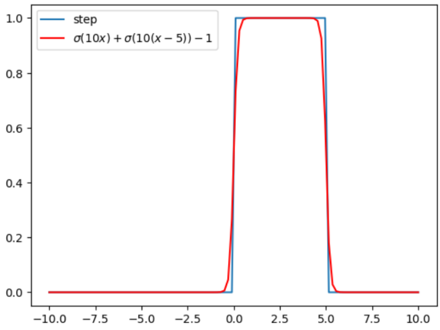
(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

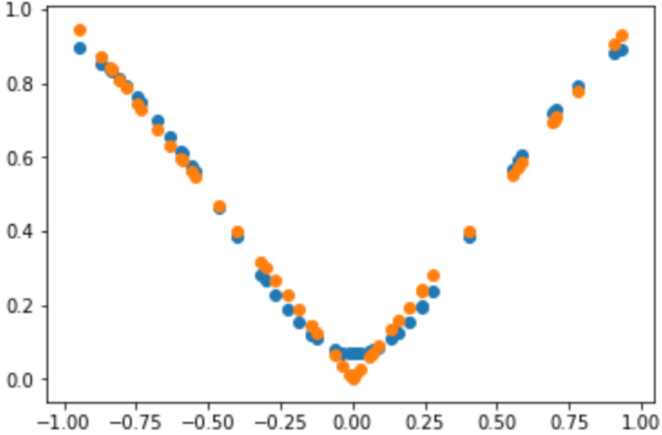
Keywords—Feedforward networks, Universal approximation, Mapping networks, Network representation capability, Stone-Weierstrass Theorem, Squashing functions, Sigma-Pi networks, Back-propagation networks.

Sigmoid Approximation



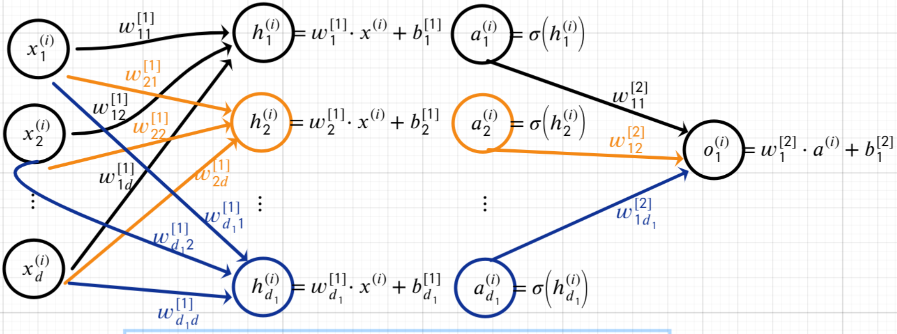


Two-layer Perceptron (one hidden layer)



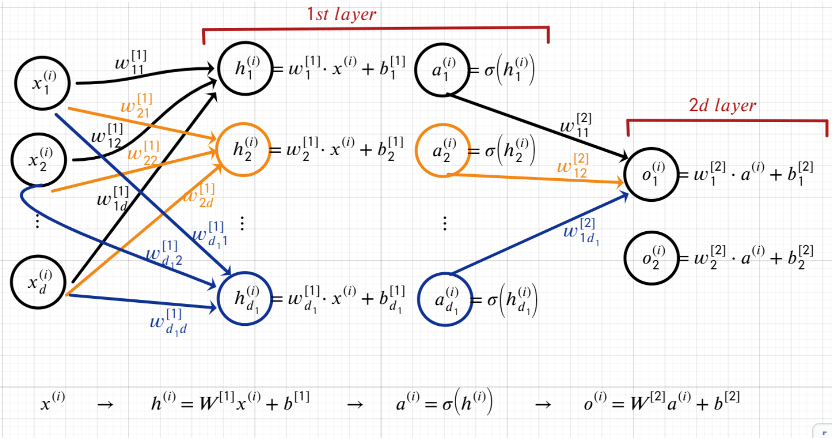
Two-layer Perceptron (one hidden layer)

$$
\begin{aligned}
o_1^{(i)} &= w_{11}^{[2]} a_1^{(i)} + w_{12}^{[2]} a_2^{(i)} + \ldots + w_{1d_1}^{[2]} a_{d_1}^{(i)} + b_1^{[2]} \\
&= w_{11}^{[2]} \sigma\left(w_1^{[1]} \cdot x^{(i)} + b_1^{[1]}\right) + w_{12}^{[2]} \sigma\left(w_2^{[1]} \cdot x^{(i)} + b_2^{[1]}\right) + \ldots + w_{1d_1}^{[2]} \sigma\left(w_{d_1}^{[1]} \cdot x^{(i)} + b_{d_1}^{[1]}\right) + b_1^{[2]}
\end{aligned}
$$



$$
L\left(w^{[1]}, w^{[2]}, b^{[1]}, b^{[2]}\right) = \frac{1}{N} \sum_{i=1}^{N} L^{(i)}, \qquad L^{(i)} = \left(o_1^{(i)} - y^{(i)}\right)^2
$$

Matrix Representation



Forward mode

Backward mode

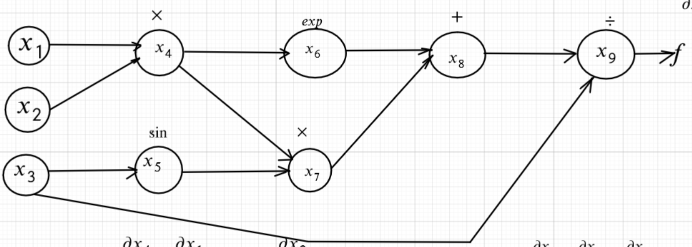
Cross mode

$$f(x_1,x_2,x_3)=\frac{x_1x_2\sin x_3+exp(x_1x_2)}{x_3}$$

Goal :

$$\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\frac{\partial f}{\partial x_3}$$

Computational Graph :



f	∇f
$O(n)$	$O(nd)$

$$\frac{\partial x_1}{\partial x_1}=1$$

$$\frac{\partial x_2}{\partial x_1}=0=\frac{\partial x_3}{\partial x_1}$$

$$\frac{\partial x_4}{\partial x_1}=\frac{\partial x_1}{\partial x_1}x_2+x_1\cdot\frac{\partial x_2}{\partial x_1}$$

$$\frac{\partial x_5}{\partial x_1}=\cos x_3\cdot\frac{\partial x_3}{\partial x_1}$$

$$\frac{\partial x_6}{\partial x_1}=e^{x_4}\cdot\frac{\partial x_4}{\partial x_1}$$

$$\frac{\partial x_7}{\partial x_1}=\frac{\partial x_4}{\partial x_1}x_5+x_4\cdot\frac{\partial x_5}{\partial x_1}$$

$$\frac{\partial x_8}{\partial x_1}=\frac{\partial x_6}{\partial x_1}+\frac{\partial x_7}{\partial x_1}$$

$$\frac{\partial x_9}{\partial x_1}=\frac{1}{x_3}\frac{\partial x_8}{\partial x_1}$$

Forward mode

Backward mode

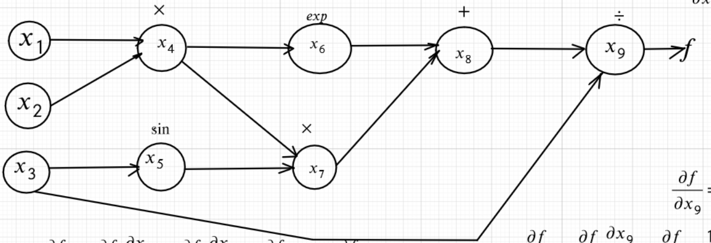
Cross mode

$$f(x_1,x_2,x_3)=\frac{x_1x_2\sin x_3+exp(x_1x_2)}{x_3}$$

Goal :

$$\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\frac{\partial f}{\partial x_3}$$

Computational Graph :



f	∇f
$O(n)$	$O(nd)$
$O(n)$	$O(n)$

$$\frac{\partial f}{\partial x_9}=1$$

$$\frac{\partial f}{\partial x_8}=\frac{\partial f}{\partial x_9}\frac{\partial x_9}{\partial x_8}=\frac{\partial f}{\partial x_9}\cdot\frac{1}{x_3}$$

$$\frac{\partial f}{\partial x_6}=\frac{\partial f}{\partial x_8}\frac{\partial x_8}{\partial x_6}=\frac{\partial f}{\partial x_8}\cdot 1$$

$$\frac{\partial f}{\partial x_7}=\frac{\partial f}{\partial x_8}\frac{\partial x_8}{\partial x_7}=\frac{\partial f}{\partial x_8}\cdot 1$$

$$\frac{\partial f}{\partial x_4}=\frac{\partial f}{\partial x_6}\frac{\partial x_6}{\partial x_4}+\frac{\partial f}{\partial x_7}\frac{\partial x_7}{\partial x_4}=\frac{\partial f}{\partial x_6}e^{x_4}+\frac{\partial f}{\partial x_7}x_5$$

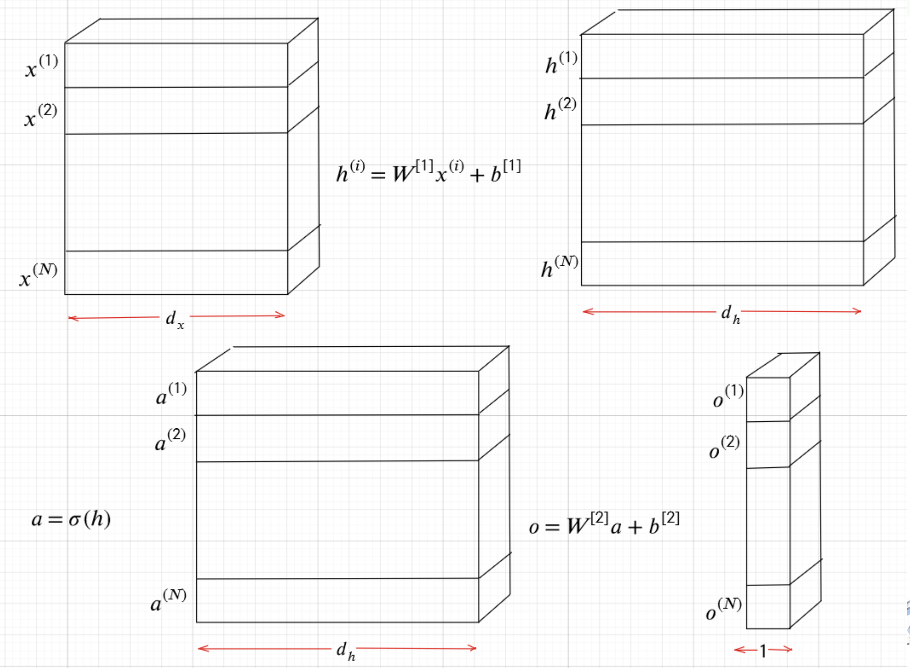
$$\frac{\partial f}{\partial x_5}=\frac{\partial f}{\partial x_7}\frac{\partial x_7}{\partial x_5}=\frac{\partial f}{\partial x_7}x_4$$

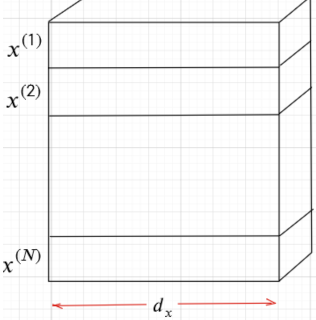
$$\frac{\partial f}{\partial x_1}=\frac{\partial f}{\partial x_4}\frac{\partial x_4}{\partial x_1}$$

$$\frac{\partial f}{\partial x_2}=\frac{\partial f}{\partial x_4}\frac{\partial x_4}{\partial x_2}$$

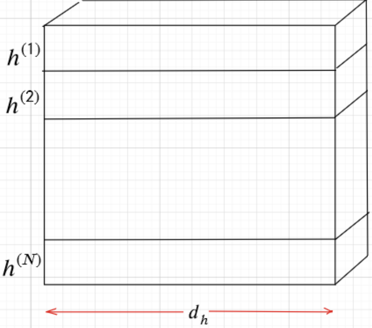
$$\frac{\partial f}{\partial x_3}=\frac{\partial f}{\partial x_5}\frac{\partial x_5}{\partial x_3}+\frac{\partial f}{\partial x_9}\frac{\partial x_9}{\partial x_3}$$

Multilayer Perceptron

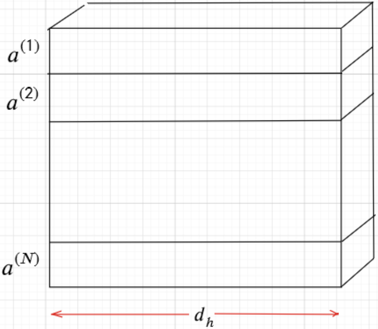




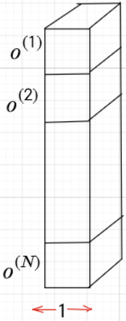
$$h^{(i)} = W^{[1]}x^{(i)} + b^{[1]}$$



$$a = \sigma(h)$$

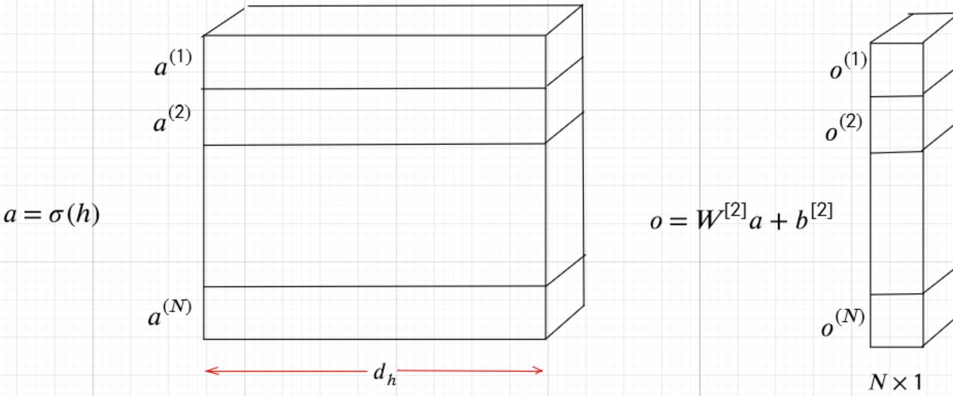


$$o = W^{[2]}a + b^{[2]}$$



$$Loss(o,y) = \frac{1}{N}\left(\left(o^{(1)} - y^{(1)}\right)^2 + \left(o^{(2)} - y^{(2)}\right)^2 + \ldots + \left(o^{(N)} - y^{(N)}\right)^2\right)$$

Multilayer Perceptron

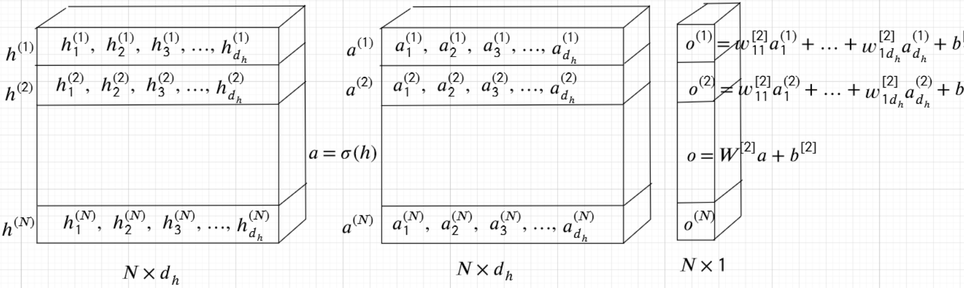


$$Loss(o, y) = \frac{1}{N}\left(\left(o^{(1)} - y^{(1)}\right)^2 + \left(o^{(2)} - y^{(2)}\right)^2 + \ldots + \left(o^{(N)} - y^{(N)}\right)^2\right)$$

$$\frac{\partial L}{\partial o^{(i)}} = \frac{2}{N}\left(o^{(i)} - y^{(i)}\right) \qquad \frac{\partial L}{\partial o} = \frac{2}{N}(o - y)$$

Multilayer Perceptron

```
class Loss():
  def __init__(self):
    self.x_in = None
    self.y_in = None

  def forward(self, x_in, y_in):
    self.x_in = x_in
    self.y_in = y_in
    return np.sum((self.x_in-self.y_in)**2)/len(self.x_in)

  def backward(self):
    return 2*(self.x_in-self.y_in)/len(self.x_in)
```



$$\frac{\partial L}{\partial a_k^{(i)}} = \frac{\partial L}{\partial o^{(i)}} \frac{\partial o^{(i)}}{\partial a_k^{(i)}} = \frac{\partial L}{\partial o^{(i)}} \cdot w_{1k}^{[2]}$$

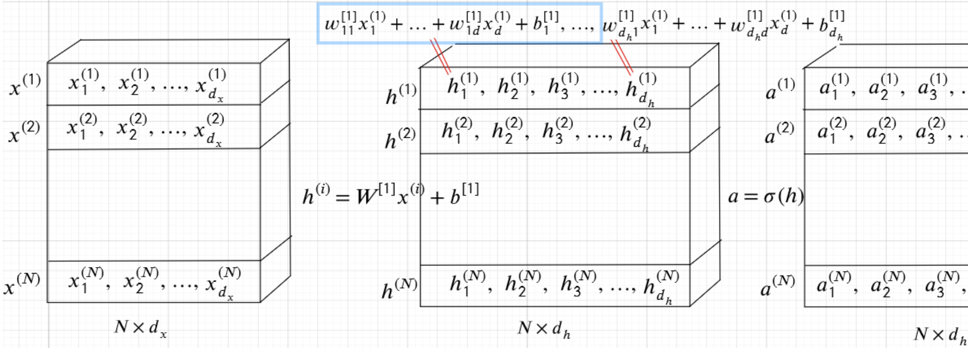
$$\frac{\partial L}{\partial a} = \begin{bmatrix} \frac{\partial L}{\partial o^{(1)}} \left(w_{11}^{[2]}, \ldots, w_{1d_h}^{[2]} \right) \\ \vdots \\ \frac{\partial L}{\partial o^{(N)}} \left(w_{11}^{[2]}, \ldots, w_{N1d_h}^{[2]} \right) \end{bmatrix} = \frac{\partial L}{\partial o} W^{[2]}$$

$$\frac{\partial L}{\partial h_k^{(i)}} = \frac{\partial L}{\partial a_k^{(i)}} \frac{\partial a_k^{(i)}}{\partial h_k^{(i)}} = \frac{\partial L}{\partial a_k^{(i)}} \sigma'\left(h_k^{(i)} \right)$$

Multilayer Perceptron

```python
class Activation():
  def __init__(self):
    self.x_in = None

  def forward(self, x_in):
    self.x_in = x_in
    return 1/(1+np.exp(-self.x_in))

  def backward(self, grad, lr=1e-3):
    return grad* np.exp(self.x_in)/(1+np.exp(self.x_in))**2
```

Multilayer Perceptron



$$\frac{\partial L}{\partial w_{11}^{[1]}} = \frac{\partial L}{\partial h_1^{(1)}} x_1^{(1)} + \ldots + \frac{\partial L}{\partial h_1^{(N)}} x_1^{(N)}$$

$$\boxed{\frac{\partial L}{\partial W^{[1]}} = \left(\frac{\partial L}{\partial h}\right)^T X}$$

$$d_h \times N \quad N \times d_x$$

$$\frac{\partial L}{\partial h}$$

$$\frac{\partial L}{\partial b_1^{[1]}} = \frac{\partial L}{\partial h_1^{(1)}} + \ldots + \frac{\partial L}{\partial h_1^{(N)}}$$

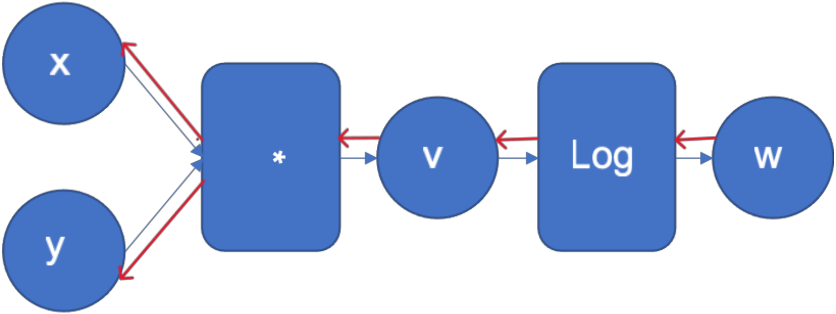
$$\boxed{\frac{\partial L}{\partial b^{[1]}} = \left(\frac{\partial L}{\partial h}\right)^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}}$$

$$d_h \times N \quad N \times 1$$

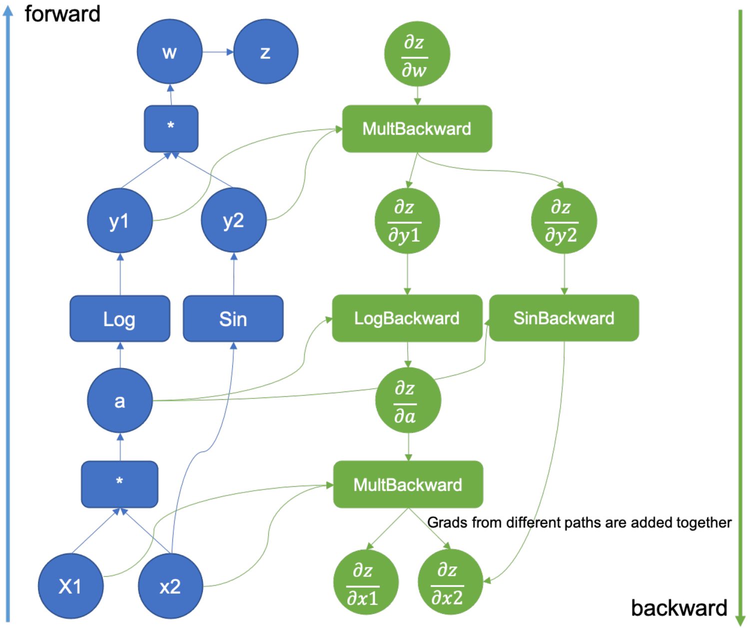
$$\boxed{\frac{\partial L}{\partial X} = \frac{\partial L}{\partial h} W^{[1]}}$$

$$N \times d_h \quad d_h \times d_x$$

```python
class Linear():
  def __init__(self, input_size, output_size):
    self.W = np.random.random((output_size, input_size))*0.01
    self.b = np.random.random((output_size, 1))*0.01

    self.grad_W = np.zeros(self.W.shape)
    self.grad_b = np.zeros(self.b.shape)

    self.x_in = None

  def forward(self, x_in):
    self.x_in = x_in
    return (self.W.dot(self.x_in.T) + self.b).T

  def backward(self, grad, lr=1e-3):
    self.grad_W = grad.T @ self.x_in
    self.grad_b = grad.T @ np.ones((len(self.x_in),1))
    grad = grad @ self.W
    self.W -= lr * self.grad_W
    self.b -= lr * self.grad_b
    #grad = grad @ self.W
    return grad
```



https://pytorch.org/blog/computational-graphs-constructed-in-pytorch/



https://pytorch.org/blog/computational-graphs-constructed-in-pytorch/

Frameworks

TensorFlow	Caffe	Keras
PyTorch	Apache MXNet	CNTK
Deeplearning4j	Torch	Chainer
Theano	H2O	Accord.NET
Apache SINGA	Graph	Onnx
Horovod	Scikit-learn	Fast.ai
BigDL	CatBoost	Flux

https://developer.nvidia.com/deep-learning-frameworks

Frameworks

Quora

Search for questions, people, and topics

Should I go for TensorFlow or PyTorch? — Related

Ismail Elezi

Computer Science student · Upvoted by Alexander Serebriansky, Software engineer, Researcher, Ph.D. in Computer Science and Ibrahim Musa, PhD Computer Science & Data Science, Chungbuk National University (2019)Author has **158** answers and **1.1M** answer views · 5y

To be fair, the only reason to use TF instead of PyTorch is if you are forced to do so (the company you work uses Tensorflow). I am one of those people who is forced to use Tensorflow in work, and I do every side project in PyTorch. PyTorch is much cleaner, being Pythonic, easier to write on OOP, much more easier to debug, and I even think that it has a better documentation. Sure, TF has more things but whom on Earth needs 7 functions which do a 2d convolution. Also, I have found responds in PyTorch forums quicker than in Tensorflow stackoverflow.

Continue Reading

PyTorch

INSTALL PYTORCH

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

PyTorch Build	Stable (2.1.0)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.8	CUDA 12.1	~~ROCm 5.6~~	CPU

Run this Command:

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

NOTE: PyTorch LTS has been deprecated. For more information, see this blog.

https://pytorch.org/