ECE521 W17 Tutorial 4

Min Bai & Kaustav Kundu

UNIVERSITY OF TORONTO

**Reference: CSC411 lecture slides, Stanford's CS321n course*

Agenda

- Logistic Regression
- Neural Networks
- Assignment 1 Questions

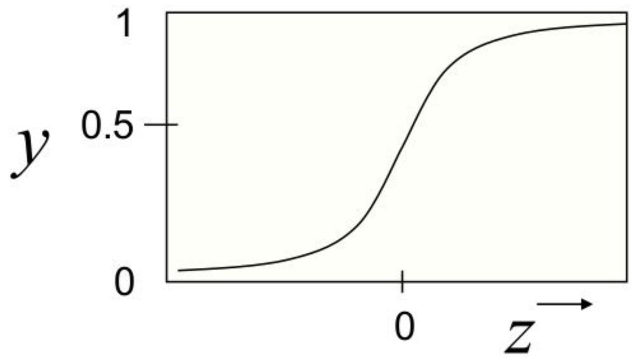
Logistic Regression

- We assumed a particular functional form: sigmoid applied to a linear function of the data

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



- The output is a smooth function of the inputs and the weights

Logistic Regression

- We assumed a particular functional form: sigmoid applied to a linear function of the data

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

  - One parameter per data dimension (feature)
  - Features can be discrete or continuous
  - Output of the model: value $y \in [0, 1]$

Logistic Regression

- If we have a value between 0 and 1, let's use it to model class probability

$$p(C = 0|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0) \quad \text{with} \quad \sigma(z) = \frac{1}{1 + \exp(-z)}$$

- Substituting we have

$$p(C = 0|\mathbf{x}) = \frac{1}{1 + \exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)}$$

- Suppose we have two classes, how can I compute $p(C = 1|\mathbf{x})$?
- Use the marginalization property of probability

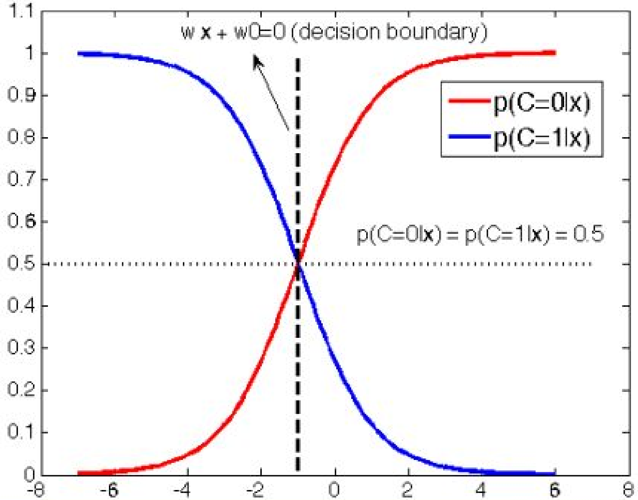
$$p(C = 1|\mathbf{x}) + p(C = 0|\mathbf{x}) = 1$$

- Thus

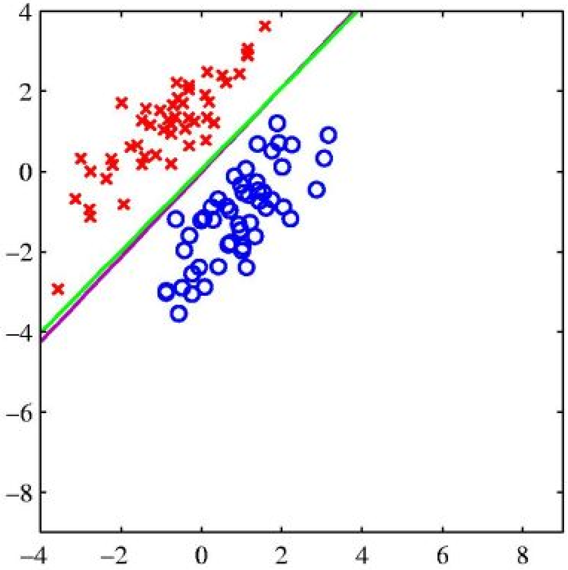
$$p(C = 1|\mathbf{x}) = 1 - \frac{1}{1 + \exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)} = \frac{\exp(-\mathbf{w}^T\mathbf{x} - w_0)}{1 + \exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)}$$

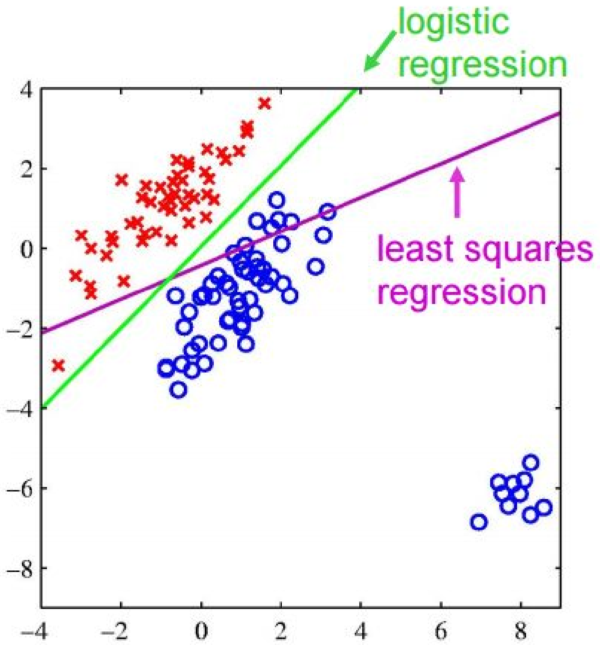
Logistic Regression

- What is the decision boundary for logistic regression?
- $p(C=1|\mathbf{x},\mathbf{w}) = p(C=0|\mathbf{x},\mathbf{w}) = 0.5$
- $p(C=0|\mathbf{x},\mathbf{w}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right) = 0.5$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$
- Decision boundary: $\mathbf{w}^T\mathbf{x} + w_0 = 0$
- Logistic regression has a linear decision boundary



Logistic Regression





If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being "too correct" (tilts away from outliers)

Logistic Regression

- We can also look at

$$p(\mathbf{w}|\{t\}, \{\mathbf{x}\}) \propto p(\{t\}|\{\mathbf{x}\}, \mathbf{w})\, p(\mathbf{w})$$

  with $\{t\} = (t^{(1)}, \cdots, t^{(N)})$, and $\{\mathbf{x}\} = (\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)})$

- We can define priors on parameters $\mathbf{w}$
- This is a form of regularization
- Helps avoid large weights and overfitting

$$\max_{\mathbf{w}} \log \left[p(\mathbf{w}) \prod_i p(t^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}) \right]$$

- What's $p(\mathbf{w})$?

Logistic Regression

- For example, define prior: normal distribution, zero mean and identity covariance $p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{I})$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(\mu - x)^2}{2\sigma^2}}$$

- This prior pushes parameters towards zero (why is this a good idea?)
- Including this prior the new gradient is

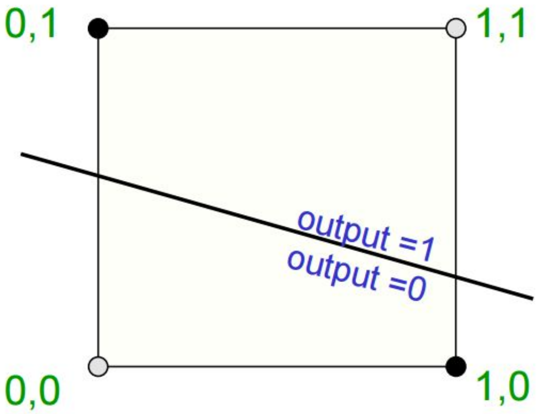
$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \frac{\partial \ell(\mathbf{w})}{\partial w_j} - \lambda \alpha w_j^{(t)}$$

  where t here refers to iteration of the gradient descent

- The parameter α is the importance of the regularization, and it's a hyper-parameter
- How do we decide the best value of α (or a hyper-parameter in general)?

Problems with Linear Classifiers

- Linear classifiers (e.g., logistic regression) classify inputs based on linear combinations of features x_i
- Many decisions involve non-linear functions of the input
- Canonical example: do 2 input elements have the same value?

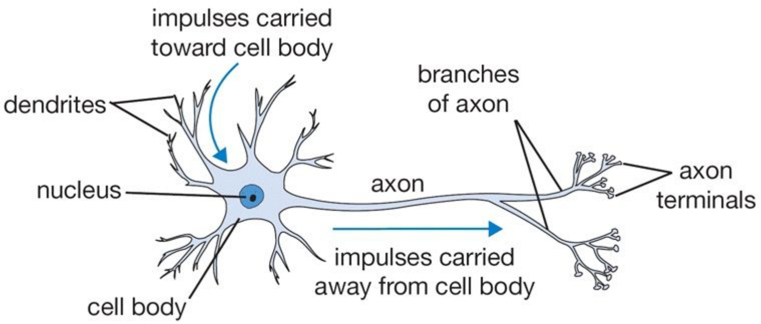


- The positive and negative cases cannot be separated by a plane
- What can we do?

Slide credit: CSC411 lecture slides

Intro to Neural Networks

- Neural networks are computational models ***inspired*** by the human brain.
- Our brain has $\sim 10^{11}$ neurons, each of which is connected to $\sim 10^{4}$ other neurons.

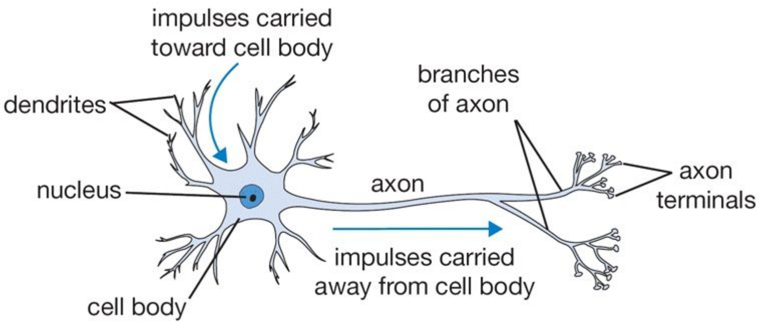


Cartoon diagram of a biological neuron

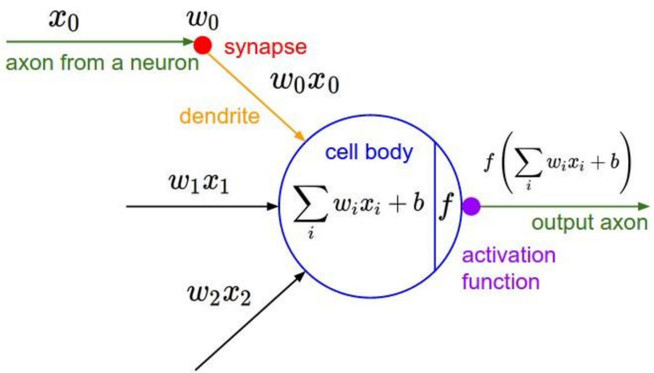
Pic credit: Stanford CS321n

Intro to Neural Networks

- Neural networks are functions of the neurons, also called hidden units.



Cartoon diagram of a biological neuron



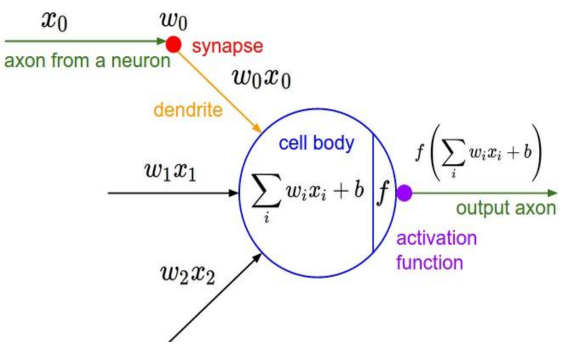
Mathematical model of a biological neuron

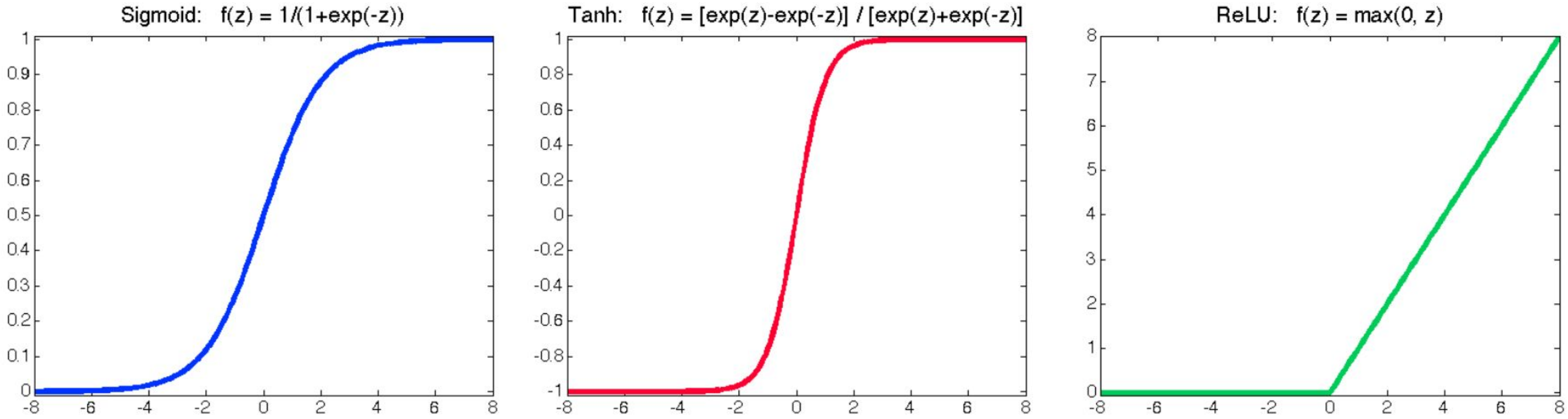
Pic credit: Stanford CS321n

Intro to Neural Networks

- Neural networks are functions of the neurons, also called hidden units.
- Commonly used activation functions
  - Sigmoid: $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$
  - Tanh: $\tanh(z) = \dfrac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$
  - ReLU (Rectified Linear Units):

$$\mathrm{ReLU}(x) = \max(0, z)$$





Pic credit: CSC411 slides

Multi layer Perceptrons

- Going deeper: a 3-layer neural network with two layers of hidden units

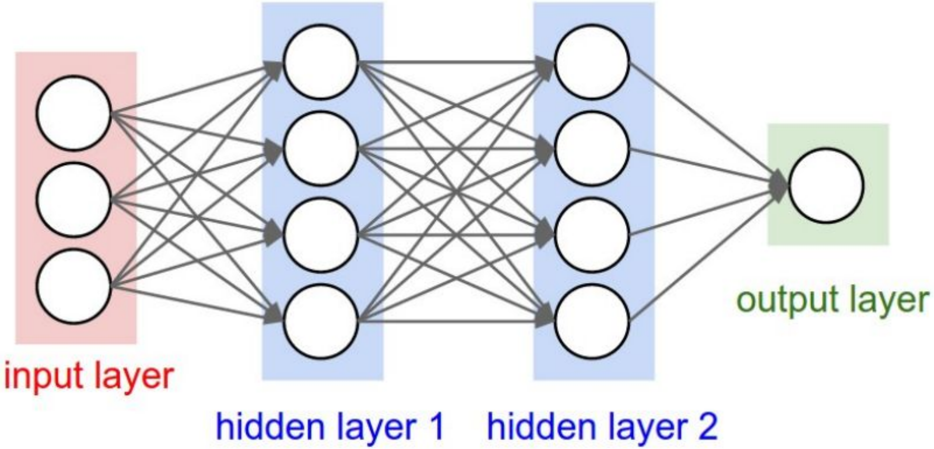


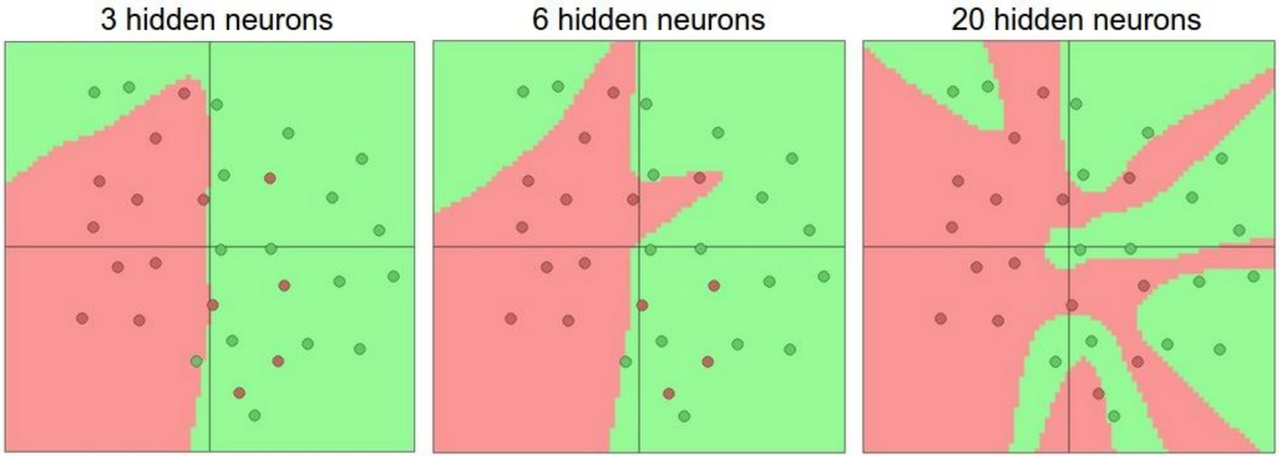
Figure : A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit

- Naming conventions; a N-layer neural network:
  - $N-1$ layers of hidden units
  - One output layer

Slide credit: CSC411 slides

Multi layer Perceptrons

- Neural Network with **at least one hidden layer** is a universal approximator (can represent any function)[1].



- Capacity of the network increases with more hidden units and more hidden layers
- Why go deeper? Read Jimmy's [paper] or the paper on [the loss surface of multilayer networks].

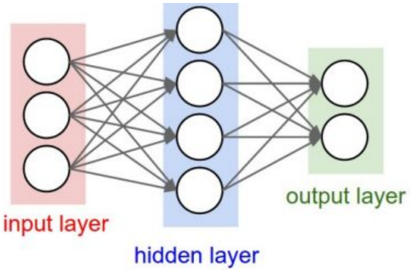
[1]Proof by Cibenko in *Approximation by Superpositions of a Sigmoidal Function*. Intuitive Explanation from Michael Nielson: link

Slide credit: CSC411 slides

Neural Networks

- We only need to know two algorithms
  - Forward pass: performs inference
  - Backward pass: performs learning

Inference: Forward Pass



- Output of the network can be written as:

$$
\begin{aligned}
h_j(\mathbf{x}) &= f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \\
o_k(\mathbf{x}) &= g(w_{k0} + \sum_{j=1}^{J} h_j(\mathbf{x}) w_{kj})
\end{aligned}
$$

(j indexing hidden units, k indexing the output units, D number of inputs)

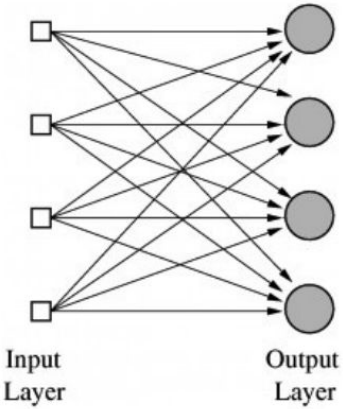
- Activation functions f, g: sigmoid/logistic, tanh, or rectified linear (ReLU)

$$\sigma(z) = \frac{1}{1+\exp(-z)}, \quad \tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}, \quad \mathrm{ReLU}(z) = \max(0, z)$$

Slide credit: CSC411 slides

Special Case

- What is a single layer (no hiddens) network with a sigmoid act. function?



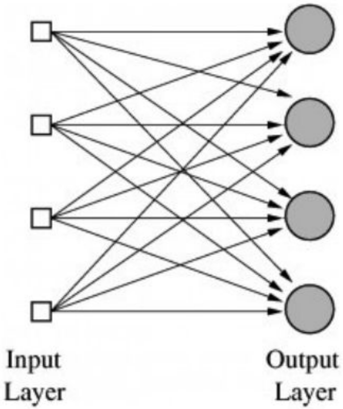
- Network:

$$o_k(\mathbf{x}) = \frac{1}{1 + \exp(-z_k)}$$

$$z_k = w_{k0} + \sum_{j=1}^{J} x_j w_{kj}$$

Special Case

- What is a single layer (no hiddens) network with a sigmoid act. function?



- Network:

$$
\begin{aligned}
o_k(\mathbf{x}) &= \frac{1}{1+\exp(-z_k)} \\
z_k &= w_{k0} + \sum_{j=1}^{J} x_j w_{kj}
\end{aligned}
$$

- Logistic regression!

Training: Backward Pass

- Find weights:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \operatorname{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

  where $\mathbf{o} = f(\mathbf{x}; \mathbf{w})$ is the output of a neural network

- Define a loss function, eg:
  - Squared loss: $\sum_k \frac{1}{2}(o_k^{(n)} - t_k^{(n)})^2$
  - Cross-entropy loss: $-\sum_k t_k^{(n)} \log o_k^{(n)}$

- Gradient descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \frac{\partial E}{\partial \mathbf{w}^t}$$

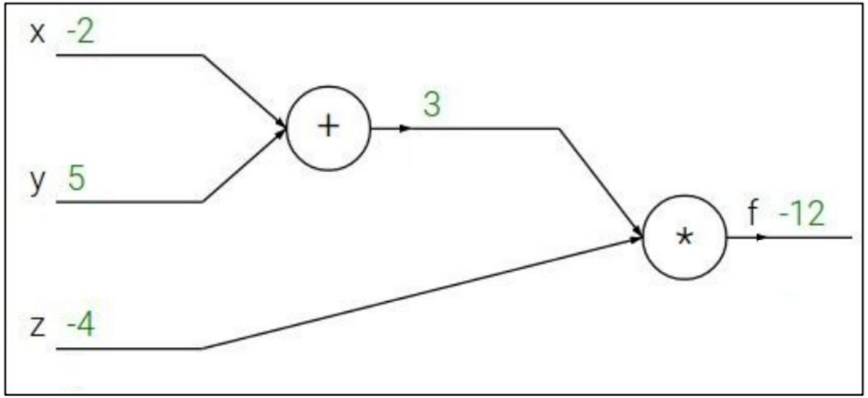
  where η is the learning rate (and E is error/loss)

Slide credit: CSC411 slides

Training: Backward Pass

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

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



Slide credit: Stanford CS321n

Training: Backward Pass

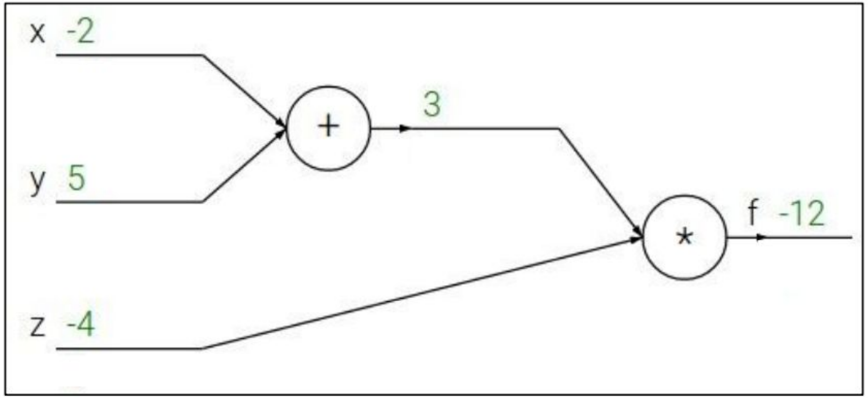
$$f(x,y,z) = (x+y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



Slide credit: Stanford CS321n

Training: Backward Pass

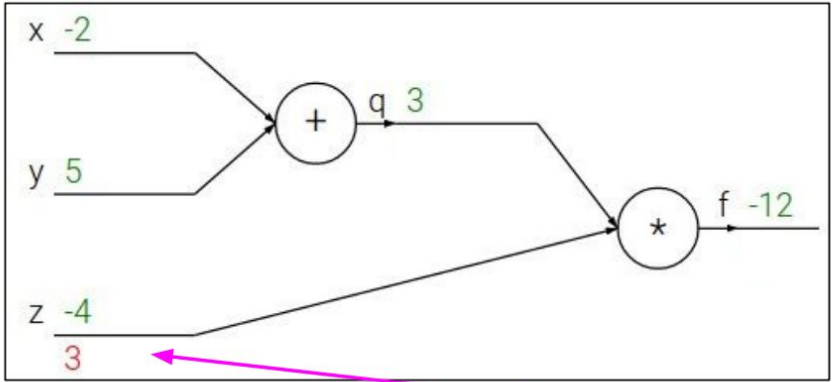
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



$$\frac{\partial f}{\partial z}$$

Slide credit: Stanford CS321n

Training: Backward Pass

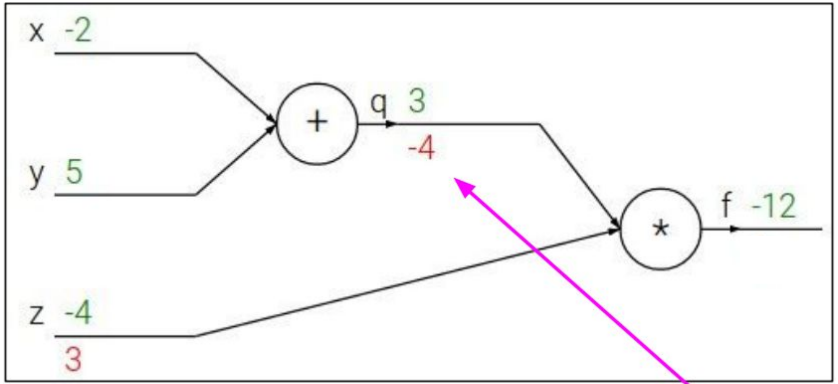
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

Training: Backward Pass

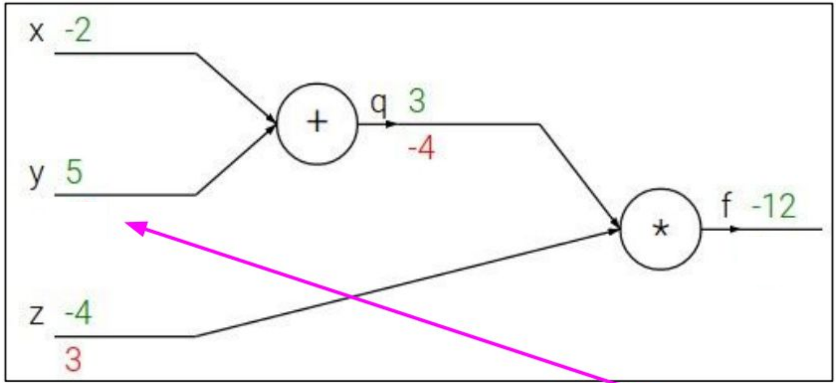
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

Slide credit: Stanford CS321n

Training: Backward Pass

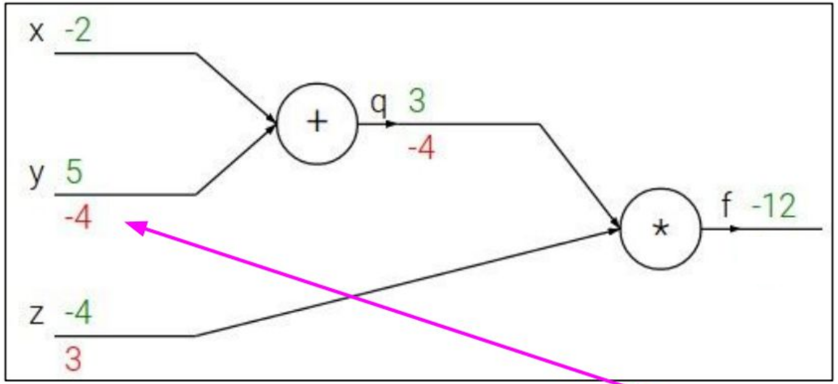
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



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

Chain rule:

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

Slide credit: Stanford CS321n

Training: Backward Pass

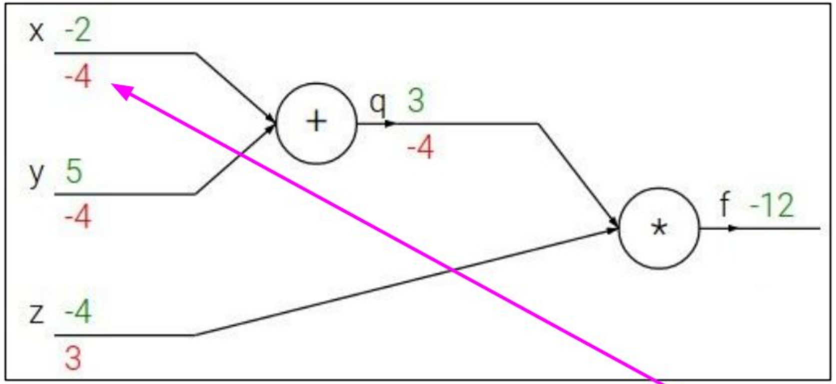
$$f(x, y, z) = (x + y)z$$

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

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

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

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



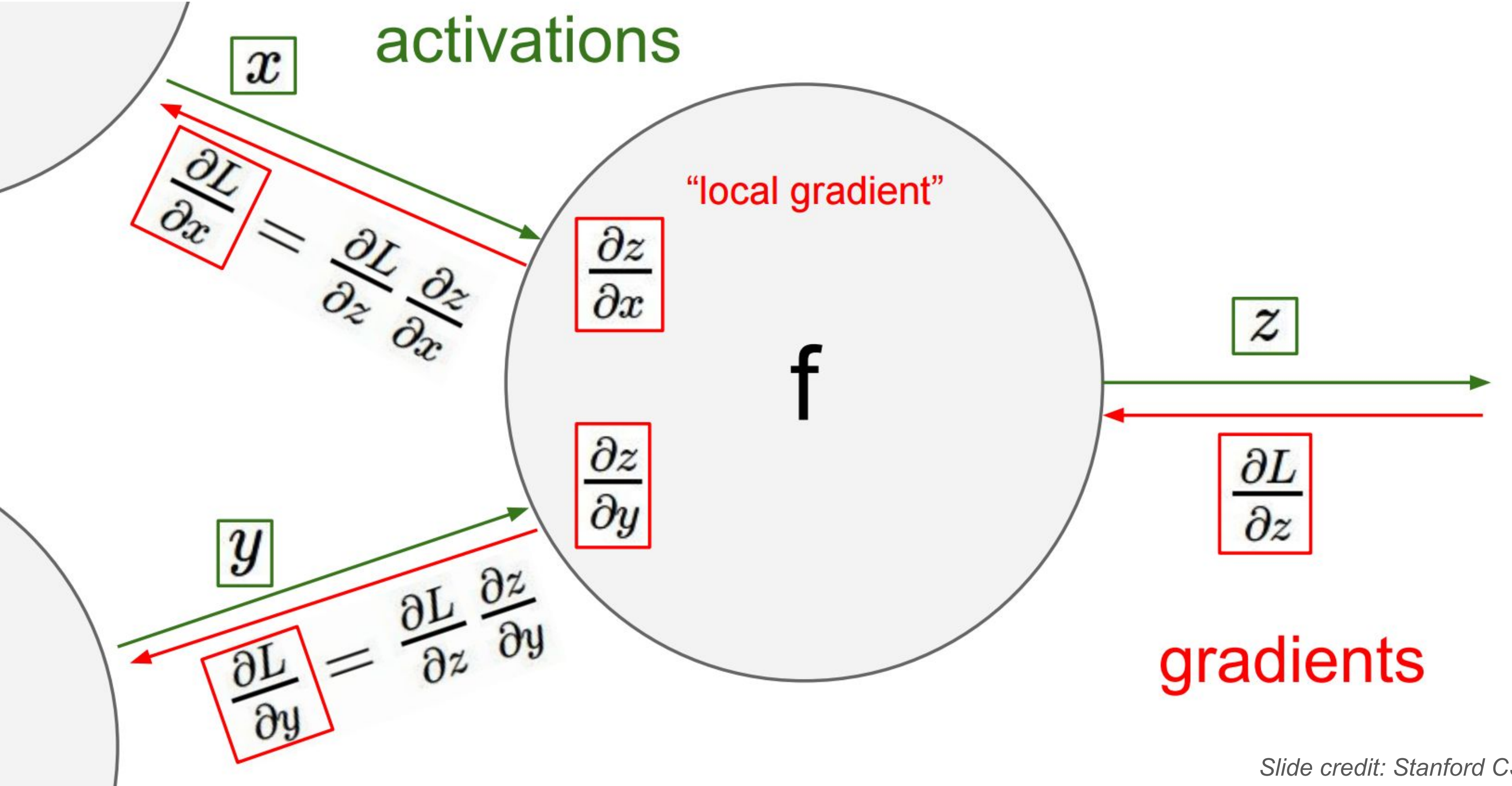
$$\boxed{\frac{\partial f}{\partial x}}$$

Chain rule:

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

Slide credit: Stanford CS321n

Training: Backward Pass

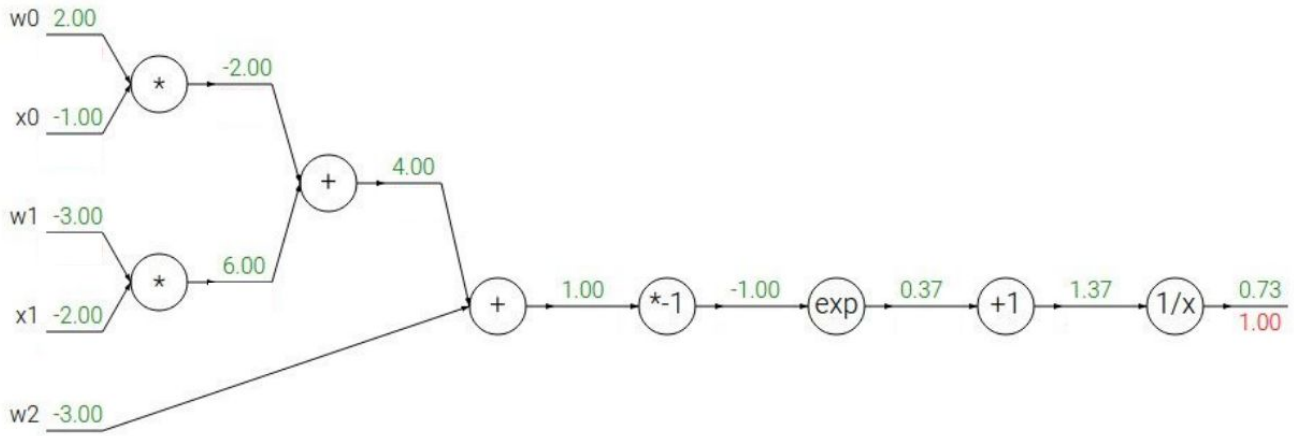


Slide credit: Stanford CS321n

Another Example

- Let's assume a two class problem

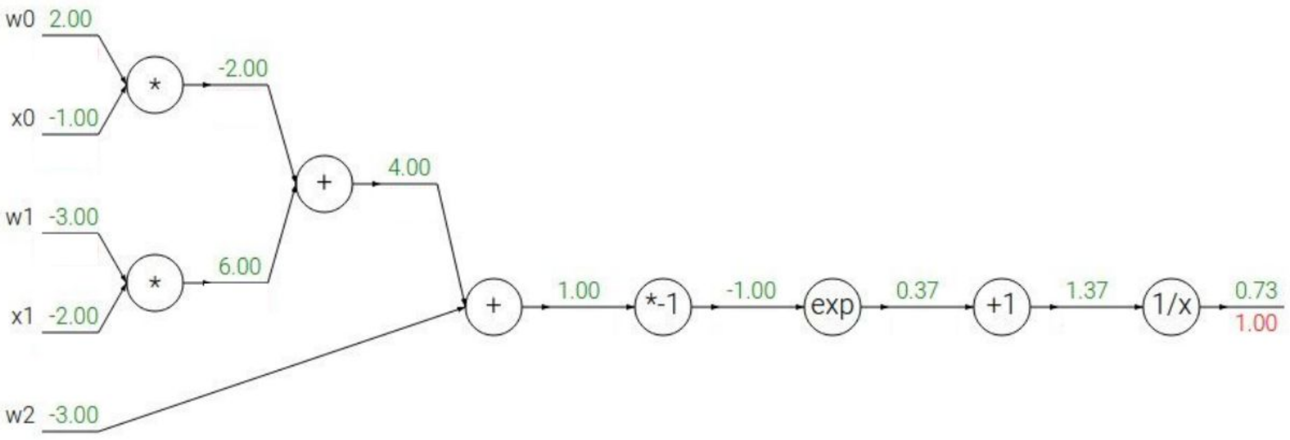
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



Slide credit: Stanford CS321n

Another Example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

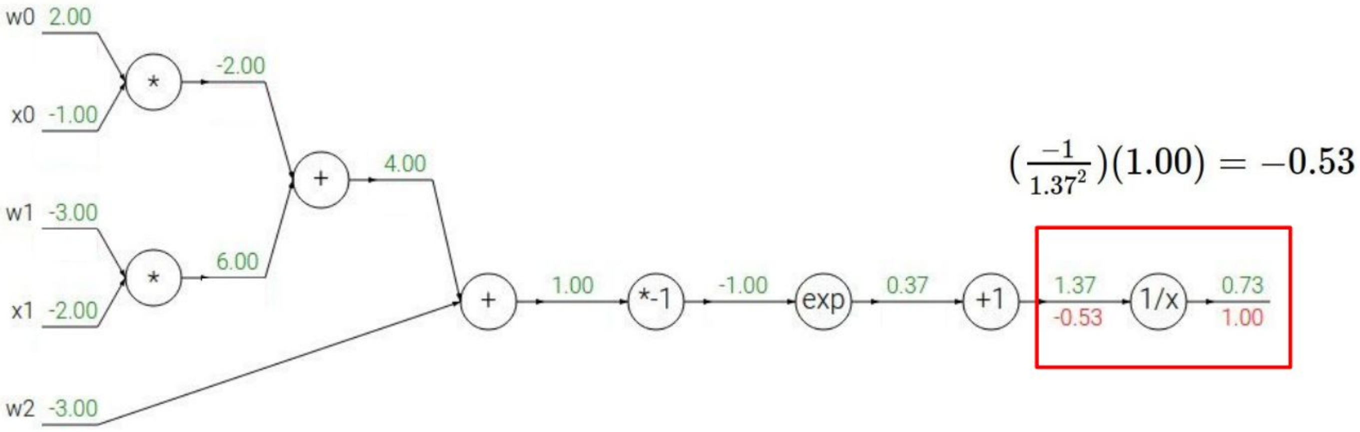


$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

Slide credit: Stanford CS321n

Another Example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

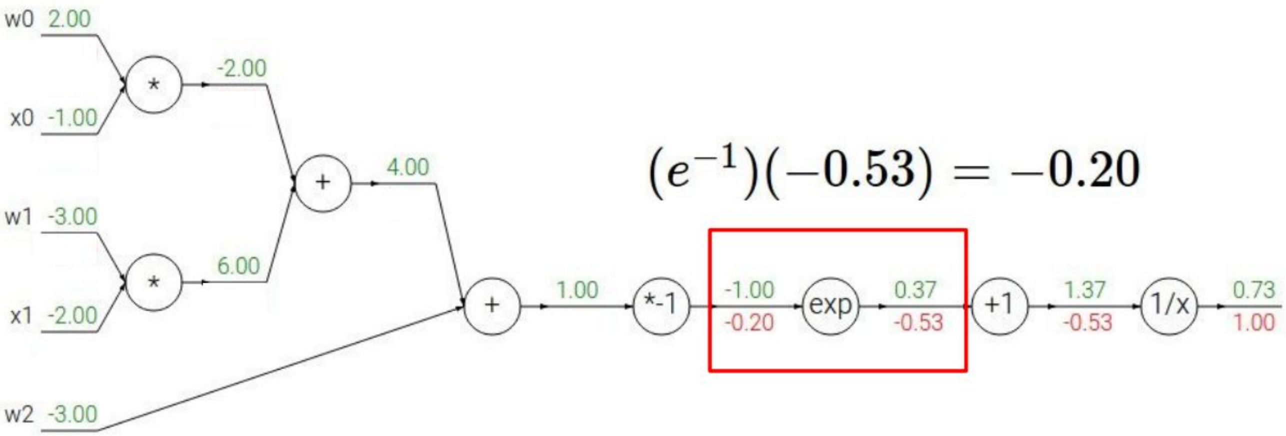
$$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Slide credit: Stanford CS321n

Another Example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



$$(e^{-1})(-0.53) = -0.20$$

$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

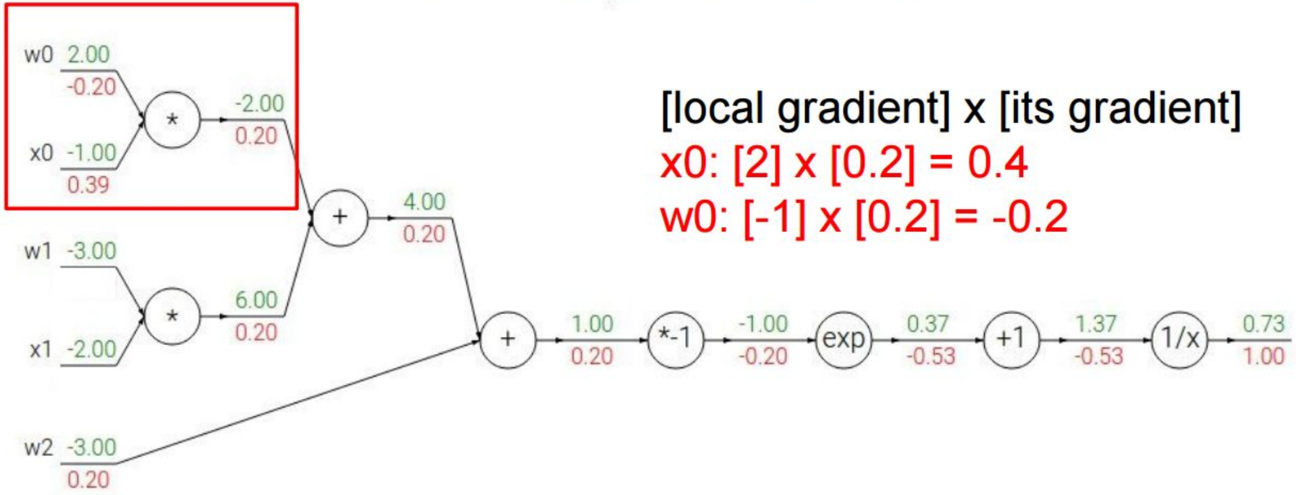
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another Example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad\qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad\qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

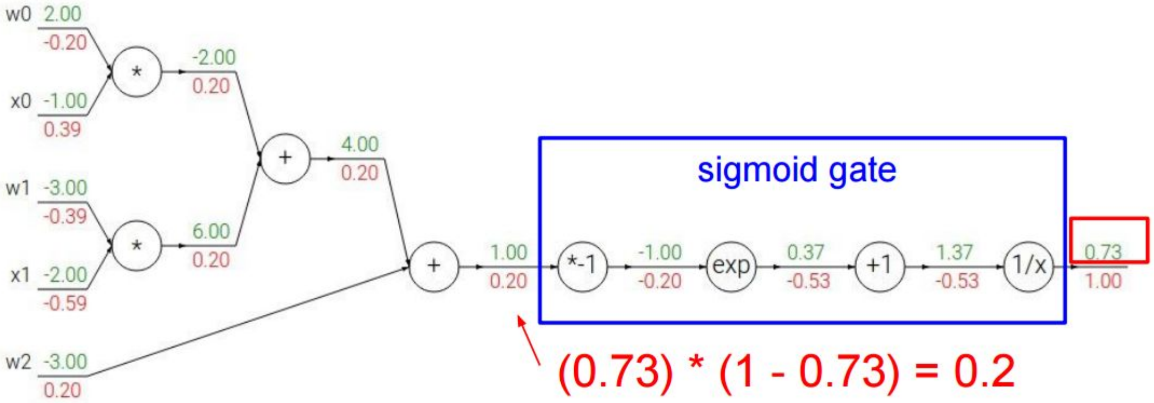
Slide credit: Stanford CS321n

Another Example

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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

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



Slide credit: Stanford CS321n

Other useful derivatives

name	function	derivative
Sigmoid	$\sigma(z) = \frac{1}{1+\exp(-z)}$	$\sigma(z) \cdot (1 - \sigma(z))$
Tanh	$\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$	$1/\cosh^2(z)$
ReLU	$\mathrm{ReLU}(z) = \max(0, z)$	$\begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z \leq 0 \end{cases}$

Slide credit: CSC411 slides

Cross Entropy

- Entropy of a distribution p, H(p), measures expected number of bits to represent data from this distribution using the optimal code from p distribution.

$$H(p) = -\sum_{x} p(x)\log p(x) = -E_{x\sim p(x)}[\log p(x)]$$

- Cross entropy of p w.r.t. q measures how many bits you require on average to represent data from p distribution using the optimal code from q distribution.

$$H(p, q) = -\sum_{x} p(x)\log q(x) = -E_{x\sim p(x)}[\log q(x)]$$

Cross Entropy

- Defined between two distributions:

$$
\begin{aligned}
\mathrm{H(p,\ q)} &= -\sum_x p(x)\log q(x) \\
&= -\sum_x p(x)\log p(x) + \sum_x p(x)\log\frac{p(x)}{q(x)} \\
&= \mathrm{H(p)} + \mathrm{D}_{KL}(p||q)
\end{aligned}
$$

- $D_{KL}(p||q)$ is the extra number of bits required to code data from p distribution using the optimal code for q distribution.

Cross Entropy

- Defined between two distributions:

$$H(p, q) = -\sum_x p(x) \log q(x)$$
$$= H(p) + D_{KL}(p||q)$$

- $p = [0, \ldots, 0, 1, 0, \ldots, 0]$, puts the entire probability mass on the ground truth class.
- $q(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ is the estimated class "probabilities"

$$L(y, y_{\text{gt}}; w) = -\sum_i \sum_j y_{\text{gt}}^{(i)}(x) \log y^{(i)}(x) = -\sum_i \log(y^{(i)}_{y_{\text{gt}}^{(i)}})$$

Cross Entropy + Weight Decay

- The total loss:

$$L(y, y_{\mathrm{gt}}; w) = -\sum_i \log(y^{(i)}_{y^{(i)}_{\mathrm{gt}}}) + ||w||^2$$

- Can be interpreted as a MAP estimation with a Gaussian prior on the weights.

Weight Initialization

- Logistic Regression
  - Convex optimization - initialization doesn't matter much
  - Initialize weights and biases to zero
- Neural Network
  - Initialize to random numbers - symmetry breaking
    - Otherwise, all branches have same weight derivatives and updates
  - Sample from Gaussian / truncated Gaussian
  - Mean = 0
  - Variance?
    - Xavier initialization guideline:

$$\mathrm{var}(w) = \sqrt{\frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}}}$$

Data Normalization

- Want each dimension of input data to be zero centered and have approximately same variance
  - Eg N(0,1)
  - For each channel:
    - Mean subtraction
    - Scaling