Deep Feedforward Networks

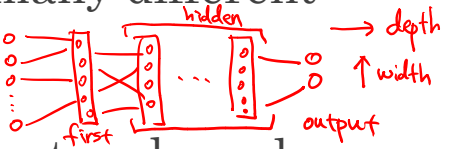
ADVANCED ARTIFICIAL INTELLIGENCE
JUCHEOL MOON

Terminology

- What are the differences between the names?
  - Artificial Neural Network
  - Multilayer Perceptron Model
  - Feedforward Neural Network
  - Deep Feedforward Network
  - Shallow Neural Network
  - Deep Deep Neural Network
  - Deep Deep Deep Neural Network

Feedforward neural networks

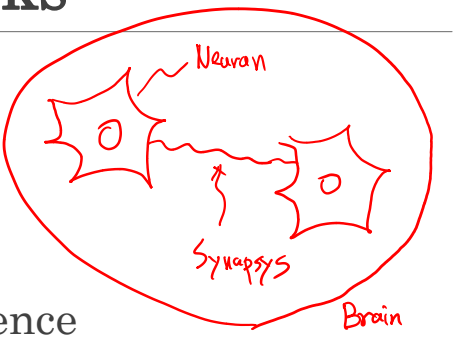
- They are called networks because they are typically represented by composing together many different functions.
  - $f(\vec{x}) = \cdots f^{(3)}(f^{(2)}(f^{(1)}(\vec{x})))$
  - $f^{(1)}$ is called first layer of the network, and so on.
  - The final layer of a feedforward network is called the output **layer**.
  - The layers between input and output are called hidden
  - The overall length of the chain gives the depth of the model.



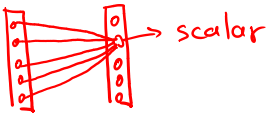
3

Feedforward neural networks

- What we want to do with $f(\vec{x})$?
  - We drive $f(\vec{x})$ to approximate $f^*(\vec{x})$
  - We assume noisy training data
  - Hence, $y \simeq f^*(\vec{x})$
- Why the networks are called **neural**?
  - They are loosely inspired by neuroscience



- The dimensionality of these hidden layers determines the width of the model
- The layer as consisting of many units (neurons), each representing a vector-to-scalar function



4

Linear vs. Nonlinear models

- Logistic regression and linear regression, are appealing
  - because they may be fit efficiently and reliably either in closed form or with convex optimization
  - The model capacity is limited to linear functions
- We can apply the linear model not to $\vec{x}$ itself but to a transformed input $\phi(\vec{x})$, where ϕ is a nonlinear transformation.
  - The strategy of deep learning is to learn ϕ
  - $y = f(\vec{x}; \vec{\theta}, \vec{w}) = \phi(\vec{x}; \vec{\theta})^T \vec{w}$ where $\vec{\theta}$ is parameters to learn ϕ
  - ϕ defines a hidden layer.

5

5

Learning XOR

- The XOR function ("exclusive or") is an operation on two binary values, x_1 and x_2.
  - Target function $y = f^*(\vec{x})$
  - Our model $y = f(\vec{x}; \vec{\theta})$
  - Make f as similar to f^*

- $\vec{X} = \begin{bmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \\ \vec{x}^{(3)} \\ \vec{x}^{(4)} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, $\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

6

6

Linear model

- Regression model with a mean squared error loss function
  - MSE loss function
  - $J(\vec{\theta}) = \frac{1}{4} \sum_{\vec{x} \in \vec{X}} \left(f^*(\vec{x}) - f(\vec{x}; \vec{\theta}) \right)^2$
- Suppose that we choose a linear model
  - $\vec{\theta}$ is consisting of $\vec{w}$ and b.
  - $f(\vec{x}; \vec{w}, b) = \vec{x}^T \vec{w} + b = \vec{w}^T \vec{x} + b$

7

Linear model

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

- $J(\vec{w}, b) = \frac{1}{4} \sum_{\vec{x}} \left(f^*(\vec{x}) - (\vec{x}^T \vec{w} + b) \right)^2$

$= \frac{1}{4} \sum_{i}^{4} \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b \right)^2$

$\frac{\partial}{\partial b} J(w_1, w_2, b) = \frac{2}{4} \sum_{i}^{4} \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b \right)(-1) = 0$

$\sum_{i}^{4} \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b \right) = \sum_i y^{(i)} - w_1 \sum_i x_1^{(i)} - w_2 \sum_i x_2^{(i)} - \sum_i b$

$= 2 - 2w_1 - 2w_2 - 4b = 0$

$w_1 + w_2 + 2b = 1$

8

Linear model

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

- $J(\vec{w}, b) = \frac{1}{4}\sum_i \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b\right)^2$

$$\frac{\partial}{\partial w_1} J(w_1, w_2, b) = \frac{2}{4}\sum_i \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b\right)\left(-x_1^{(i)}\right)$$

$$\sum_i \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b\right) x_1^{(i)} = 0$$

$$\Rightarrow \underbrace{\sum_i x_1^{(i)} y^{(i)}}_{0+1+0+0} - w_1 \sum_i x_1^{(i)2} - w_2 \underbrace{\sum_i x_1^{(i)} x_2^{(i)}}_{0+0+0+1} - b \sum_i x_1^{(i)}$$

$$1 - 2w_1 - w_2 - 2b = 0$$

$$\boxed{2w_1 + w_2 + 2b = 1}$$

9

Linear model

$f(\vec{x}) = \frac{1}{2}$

x_1	x_2	y
0	0	$(0-\frac{1}{2})^2$
1	0	$(1-\frac{1}{2})^2$
0	1	$(1-\frac{1}{2})^2$
1	1	$(0-\frac{1}{2})^2$

$4 \cdot (\frac{1}{2})^2 = 1$

- $J(\vec{w}, b) = \frac{1}{4}\sum_i \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b\right)^2$

$$\frac{\partial}{\partial w_2} J(w_1, w_2, b)$$

$$= \frac{2}{4}\sum_i \left(y^{(i)} - x_1^{(i)} w_1 - x_2^{(i)} w_2 - b\right)\left(-x_2^{(i)}\right)$$

$$\sum_i y^{(i)} x_2^{(i)} - w_1 \sum_i x_1^{(i)} x_2^{(i)} - w_2 \sum_i x_2^{(i)} - b \sum_i x_2^{(i)}$$

$$-w_1 - 2w_2 - 2b = 0$$

$$\boxed{w_1 + 2w_2 + 2b = 1}$$

$$\boxed{\begin{aligned} w_1 + w_2 + 2b &= 0 \\ 2w_1 + w_2 + 2b &= 1 \end{aligned}}$$

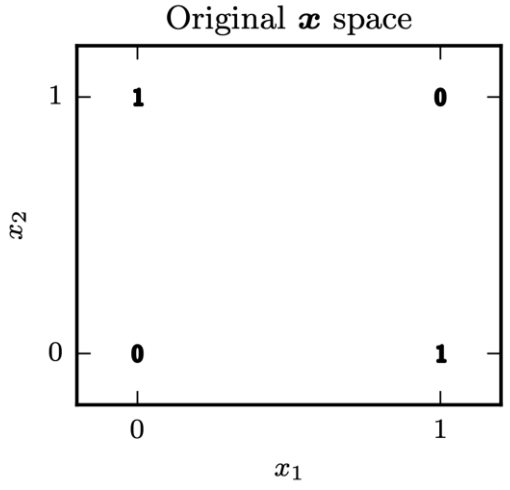
$$\boxed{\begin{aligned} w_1 &= 0 \\ w_2 &= 0 \\ b &= \frac{1}{2} \end{aligned}}$$

$$\hat{y} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

10

Linear model

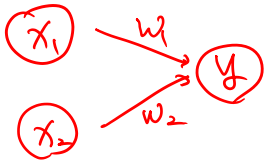
- $f(\vec{x}; \vec{w}, b) = \vec{0}^T \vec{x} + b$ minimize $J(\vec{w}, b)$



11

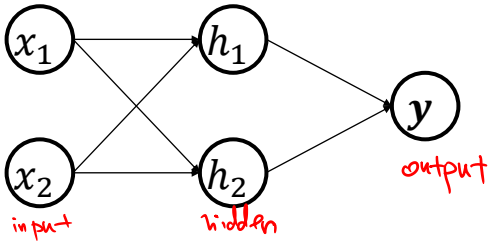
Hidden layer

- Feedforward network with one hidden layer containing two hidden units
  - $\vec{h} = f^{(1)}(\vec{x}; \vec{W}, \vec{c})$
  - $y = f^{(2)}(\vec{h}; \vec{w}, b)$
  - $f(\vec{x}; \vec{W}, \vec{c}, \vec{w}, b) = f^{(2)}(f^{(1)}(\vec{x}))$



$y = w_1 x_1 + w_2 x_2$

$= \vec{w}^T \vec{x}$



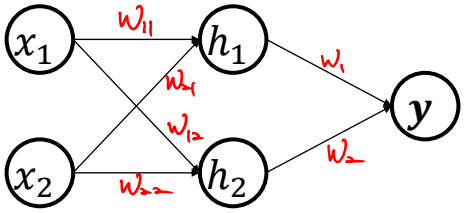
12

Hidden layer

- Suppose $\vec{h} = f^{(1)}(\vec{x}) = \vec{W}^T\vec{x}$ and $y = f^{(2)}(\vec{h}) = \vec{h}^T\vec{w}$

$h_1 = w_{11}x_1 + w_{21}x_2$
$h_2 = w_{12}x_1 + w_{22}x_2$
$y = w_1h_1 + w_2h_2$

- $y = \vec{h}^T\vec{w} = (\vec{W}^T\vec{x})^T\vec{w} = \vec{x}^T\vec{W}\vec{w} = (\vec{W}\vec{w})^T\vec{x} = \vec{A}\cdot\vec{x}$

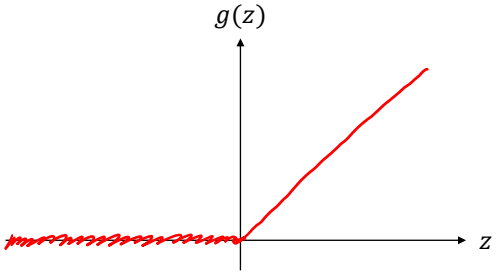


13

Nonlinear model

- Neural networks do so using an affine transformation
  - Nonlinear function called an activation function
  - $\vec{h} = g(\vec{W}^T\vec{x} + \vec{c})$
    - $\vec{W}$: weight, $\vec{c}$: bias
- In modern neural networks, the default activation function is $g(z) = \max\{0, z\}$

rectified linear unit
ReLU



14

Solution to XOR problem

- $f(\vec{x}; \overrightarrow{W}, \vec{c}, \vec{w}, b) = \vec{w}^T \vec{h} + b = \vec{w}^T g(\vec{W}^T \vec{x} + \vec{c}) + b$

$= \vec{w}^T \max(0, \vec{W}^T \vec{x} + \vec{c}) + b$

$\vec{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, $\vec{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\vec{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, $b = 0$

$\vec{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ $\vec{X}\vec{W} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$

15

15

Broadcasting

- $\overrightarrow{W}^T \vec{x} + \vec{c} = \vec{W}\vec{X} + \vec{c}$

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

$\max(0, \vec{W}\vec{X} + \vec{c})$

$= \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$

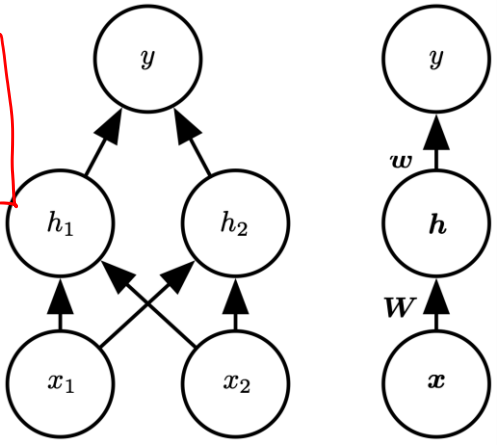
16

16

Solution to XOR problem

- $\max\{0, \vec{W}\vec{X} + \vec{C}\}\vec{w} + b$

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



17

17

Gradient-Based Learning

- Convex optimization converges starting from any initial parameters
- Stochastic gradient descent applied to non-convex loss functions has no such convergence guarantee, and is sensitive to the values of the initial parameters
- For feedforward neural networks,
  - All weights to small random values.
  - The biases may be initialized to zero

18

18

Cost function

- Most modern neural networks are trained using maximum likelihood
- $J(\vec{\theta}) = -\mathbb{E}_{\vec{x},\vec{y}\sim\hat{p}_{data}} \log p_{model}(\vec{y}|\vec{x})$

$x_1 > x_2 \equiv \log x_1 > \log x_2$

  - $\hat{p}_{data}$: sample distribution
  - $\vec{X}, \vec{Y}$: sample data
  - $p_{model}(\vec{y}|\vec{x})$: (probability) distribution that $\hat{\vec{y}} = \vec{y}$ with given $\vec{x}$

$\vec{x} \rightarrow$ Model $\rightarrow \hat{\vec{y}} = \vec{y}$

19

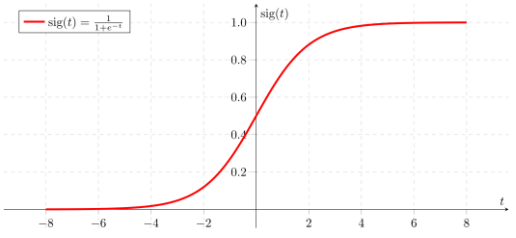
Bernoulli output

- Predicting the value of a binary variable y
  - $P(y=1|\vec{x})$ $\hat{y} = (y=1)$
- A sigmoid output unit

$\vec{x} \rightarrow$ Model $\rightarrow z \rightarrow \sigma(z) \downarrow \hat{y}$

$\hat{y} = \sigma(z)$

  - where σ is the logistic sigmoid function
  - and



20

Bernoulli output

- Unnormalized probability distribution $\tilde{P}(y)$, which does not sum to 1.
- Assume that the unnormalized log probabilities are linear in y and z

$\log \tilde{P}(y) = yz$ $\quad \tilde{P}(y) = e^{yz} = \exp(yz)$ $\quad y \in \{0, 1\}$

$$P(y) = \frac{e^{yz}}{\sum_{y'=0} e^{y'z}} = \frac{e^{yz}}{e^0 + e^z} = \frac{e^{yz}}{e^z + 1} = \begin{cases} \frac{e^z}{e^z+1} = \sigma(z) & y=1 \\ \frac{1}{e^z+1} = \sigma(-z) & y=0 \end{cases}$$

$= \sigma((2y-1)z)$ $\quad y=1 \to \sigma(z)$, $y=0 \to \sigma(-z)$

21

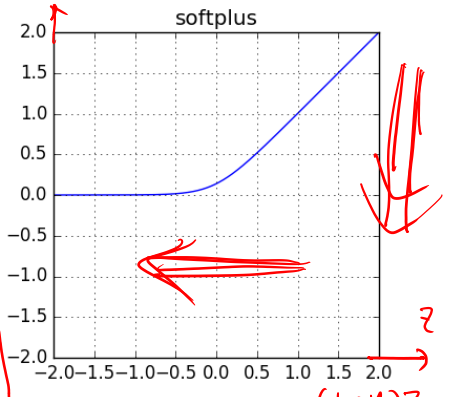
Bernoulli output

- The loss function for maximum likelihood learning
  - $L(\vec{\theta}) = -\log P(y|\vec{x})$

$= -\log \sigma((2y-1)z)$

$= \zeta((1-2y)z)$

ζ is the softplus function



$\vec{x} \to$ Model $\to z \to \sigma(z) \to \hat{y}$

- Saturation: when $y = 1$ and z is very positive
  - or when $y = 0$ and z is very negative

$(1-2y)z$: $y=1 \to -z$, $y=0 \to z$

22

Multinoulli output

- Probability distribution over a discrete variable with n possible values
- We need to produce a vector $\hat{\vec{y}}$, with $\hat{y}_i = P(y = i|\vec{x})$ and want $\hat{y}_i$ represents a probability distribution.

$$\begin{cases} 0 \le \hat{y}_i \le 1 \\ \sum_{i=1}^{n} \hat{y}_i = 1 \end{cases}$$

$$\hat{\vec{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

- The softmax function is given by
  - $g(\vec{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$

23

Cross Entropy

- Update z to $\vec{z}$

$$\vec{z} = \vec{W}^T \vec{h} + \vec{b}$$

$$\vec{x} \rightarrow \boxed{\text{Model}} \rightarrow \vec{z} = \begin{bmatrix} \vdots \end{bmatrix}$$

- In information theory, the cross entropy between two probability distributions $\vec{y}, \hat{\vec{y}}$ is defined
  - $H(\vec{y}, \hat{\vec{y}}) = -\sum_{i=1}^{n} y_i \log \hat{y}_i$

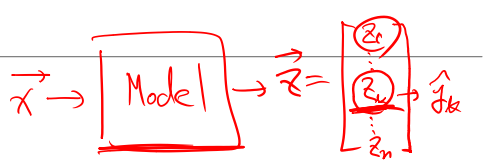
$$\vec{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \hat{\vec{y}} = \begin{bmatrix} 0.12 \\ 0.8 \\ 0.01 \\ \vdots \\ 0.09 \end{bmatrix}$$

- One-hot encoding
  - $\vec{y} = [0 \ 0 \ \cdots \ 1 \ \cdots \ 0 \ 0]^T$
  - if $\vec{y}$'s class is k^{th} one among n values

24

Cross Entropy

- Assume $\vec{y} = [0\ 0\ 0\ \ldots 1\ \ldots 0]^T$ ← cat
- $L(\vec{\theta}) = H(\vec{y}, \hat{\vec{y}}_i) = -\sum_{i=1}^{n} y_i \log \hat{y}_i$



$= -(0\log\hat{y}_1 + 0\log\hat{y}_2 + \cdots + 1\cdot\log\hat{y}_k + \cdots + 0\log\hat{y}_n) = -\log\hat{y}_k$

$= -\log\frac{\exp(z_k)}{\sum_j \exp(z_j)} = -\log e^{z_k} + \log\sum_j e^{z_j} \approx -z_k + \max_{j \neq k} z_i$

- The first term shows that the input z_k always has a direct contribution to the cost function.
- The second term shows that cost function penalizes the most incorrect prediction

25

25

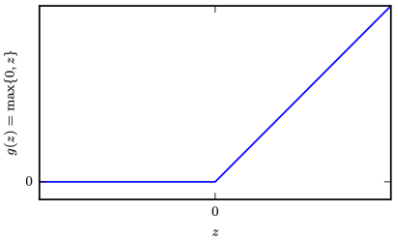
Hidden Units

- The design of hidden units is an active area of research and does not have many definitive guiding theoretical principles
  - It can be difficult to determine when to use which kind of hidden unit
  - In general, rectified linear units (ReLU) are a default choice of hidden unit
  - The design process consists of trial and error, intuiting that a kind of hidden unit may work well, and then training a network with that kind of hidden unit and evaluating its performance on a validation set.

26

26

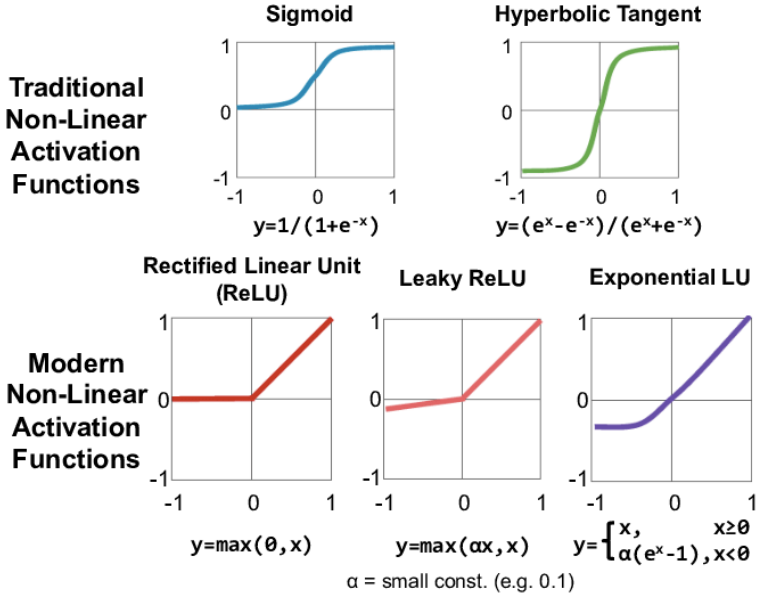
ReLU



- $g(z) = \max\{0, z\}$
  - Not differentiable at $z = 0$
- Invalid for a gradient based learning algorithm?
- In general, a function $g(z)$ has
  - a left derivative defined by the slope of the function immediately to the left of z, which is 0
  - a right derivative defined by the slope of the function immediately to the right of z, which is 1

27

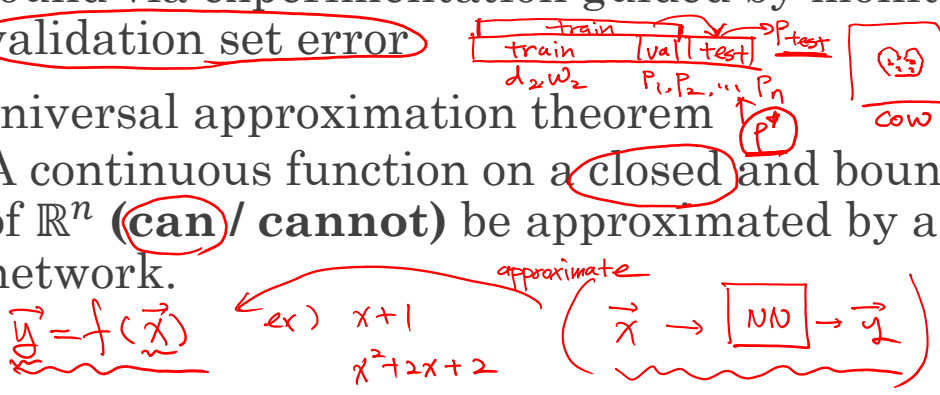
Hidden Units



28

Architecture Design

- How to choose the depth of the network and the width of each layer?
  - The ideal network architecture for a task must be found via experimentation guided by monitoring the validation set error
- Universal approximation theorem
  - A continuous function on a closed and bounded subset of $\mathbb{R}^n$ **(can / cannot)** be approximated by a neural network.



29

Universal approximation theorem

- It means that regardless of what function we are trying to learn, we know that a large MLP will be able to **represent** this function.
- However, we are not guaranteed that the training algorithm will be able to learn that function.
  - Optimization algorithm used for training may not be able to find the value of the parameters
  - Training algorithm might choose the wrong function due to overfitting

30

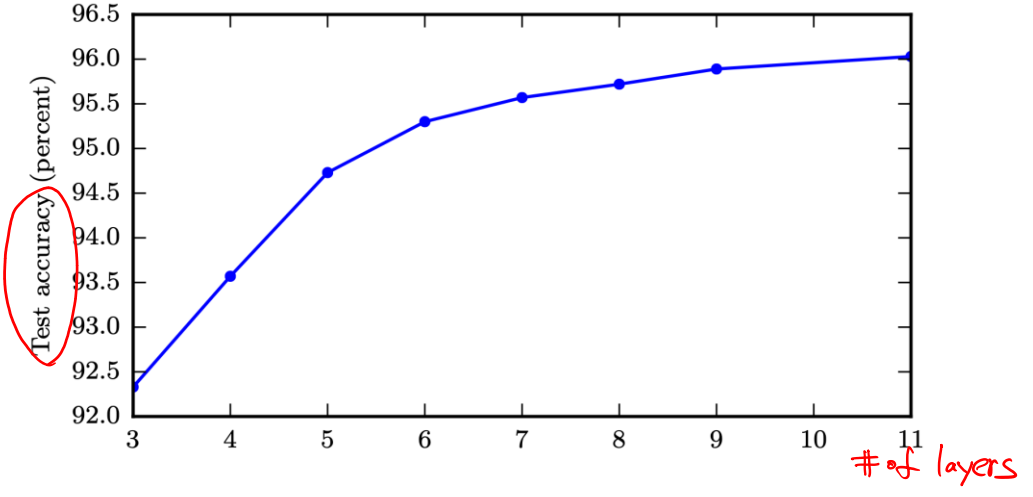
A single-layer network

- There **(does / does not)** exists a single-layer network large enough to achieve any degree of accuracy we desire.
  - For binary classification problem, $\mathcal{O}(2^n)$ of hidden units are required.
- A feedforward network with a single layer is sufficient to represent any function
  - but the layer may be infeasibly large and may fail to learn and generalize correctly.

31

Multi-layer network

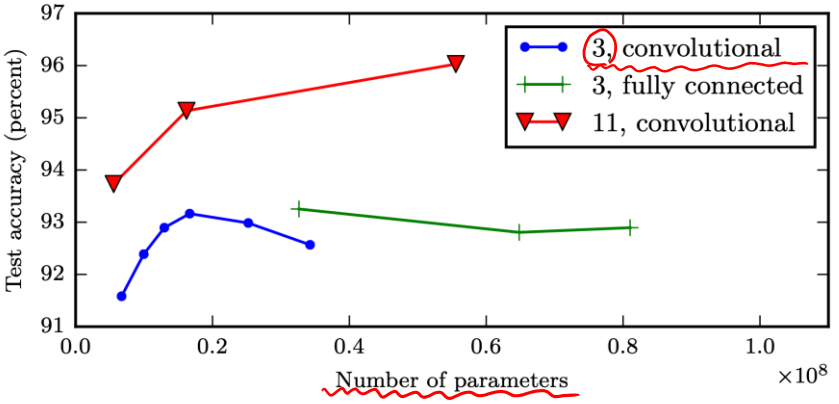
- Empirical results showing that deeper networks generalize better



32

Multi-layer network

- Increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance



33

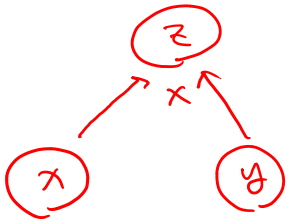
Back-Propagation

- Forward propagation
  - The inputs $\vec{x}$ provide the initial information that then propagates up to the hidden units at each layer and finally produces $\hat{\vec{y}}$
- Back-propagation (backprop)
  - allows the information from the cost to then flow backwards through the network, in order to compute the gradient
  - refers only to the method for computing the gradient

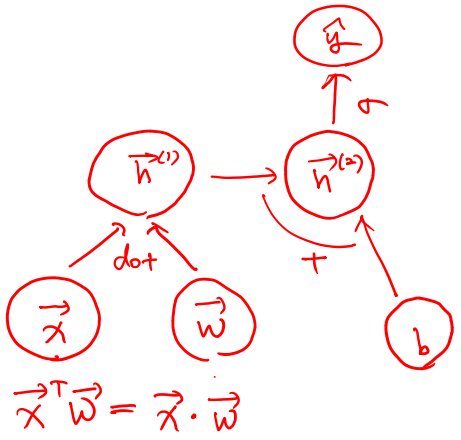
34

Computational Graphs

- $z = xy$



- $\hat{y} = \sigma(\vec{x}^T \vec{w} + b)$

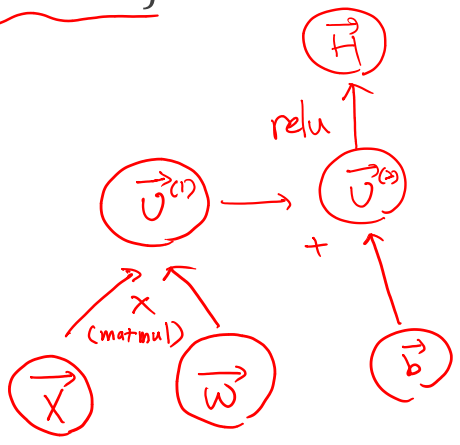


35

35

Computational Graphs

- $\vec{H} = \max\{0, \vec{X}\vec{W} + \vec{b}\}$



36

36

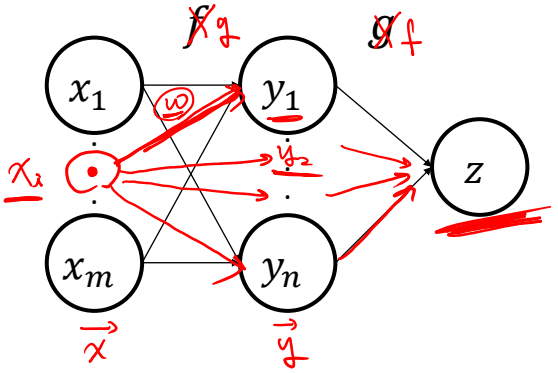
Chain Rule of Calculus

- Let x be a real number and let f and g both be functions mapping from a real number to a real number.
  - Suppose that $y = g(x)$ and $z = f(g(x)) = f(y)$.
  - $\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$

37

Chain Rule of Calculus

- Suppose that $\vec{x} \in \mathbb{R}^m$, $\vec{y} \in \mathbb{R}^n$, g maps from $\mathbb{R}^m$ to $\mathbb{R}^n$, and f maps from $\mathbb{R}^n$ to $\mathbb{R}$. If $\vec{y} = g(\vec{x})$ and $z = f(\vec{y})$
  - $\frac{\partial z}{\partial x_i} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_i} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_i} + \cdots = \sum_j^n \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}$



38

Chain Rule of Calculus

- In vector notation
- $\nabla_{\vec{x}} z = \left[\frac{\partial z}{\partial x_1} \cdots \frac{\partial z}{\partial x_m}\right]^T = \left[\sum_j^n \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_1} \cdots \sum_j^n \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_m}\right]^T$

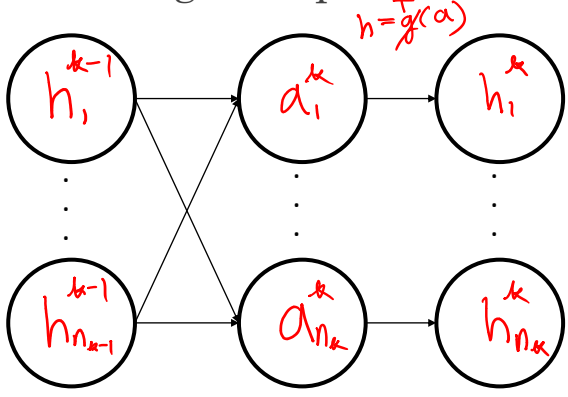
$= \left[\frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \cdots + \frac{\partial z}{\partial y_n}\frac{\partial y_n}{\partial x_1} \quad \cdots \quad \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_m} + \cdots + \frac{\partial z}{\partial y_n}\frac{\partial y_n}{\partial x_m}\right]^T$ $(m \times 1)$

$= \underbrace{\left[\frac{\partial z}{\partial y_1} \cdots \frac{\partial z}{\partial y_n}\right]}_{(\nabla_{\vec{y}} z)^T} \underbrace{\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \\ \frac{\partial y_n}{\partial x_1} & & \frac{\partial y_n}{\partial x_m} \end{bmatrix}^T}_{\text{Jacobian matrix } \frac{\partial \vec{y}}{\partial \vec{x}}} = \left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)^T \nabla_{\vec{y}} z$

39

Forward propagation

- Input:
  - Network depth: l, activation function: g (f)
  - $\vec{W}^{(i)}$: $i \in \{1, \ldots, l\}$, the weight matrices of the model
  - $\vec{b}^{(i)}$: $i \in \{1, \ldots, l\}$, the bias parameters of the model
  - $\vec{x}$: the input to process, $\vec{y}$: the target output
- $\vec{h}^{(0)} = \vec{x}$
- for $k = 1, \ldots, l$ do
  - $\vec{a}^{(k)} = \vec{W}^{(k)} \vec{h}^{(k-1)} + \vec{b}^{(k)}$
  - $\vec{h}^{(k)} = g(\vec{a}^{(k)})$ (f)
- $\vec{y} = \vec{h}^{(l)}$
- $J = L(\hat{\vec{y}}, \vec{y})$



40

Backward propagation

- $\vec{g} \leftarrow \nabla_{\hat{y}} J = \nabla_{\hat{y}} L(\hat{y}, y)$
- for $k = l, l-1, \ldots, 1$ do
  - $\vec{g} \leftarrow \nabla_{\vec{a}^{(k)}} J = \vec{g} \odot f'(\vec{a}^{(k)})$
  - $\nabla_{\vec{W}^{(k)}} J = \vec{g} \odot \vec{h}^{(k-1)}$
  - $\nabla_{\vec{b}^{(k)}} J = \vec{g}$
  - $\vec{g} \leftarrow \nabla_{\vec{h}^{(k-1)}} J = \vec{W}^{(k)\top} \vec{g}$

f: activation

$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h} \cdot \frac{\partial h}{\partial a}$

$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial a} \frac{\partial a}{\partial w}$

$\frac{\partial a}{\partial w} = h$

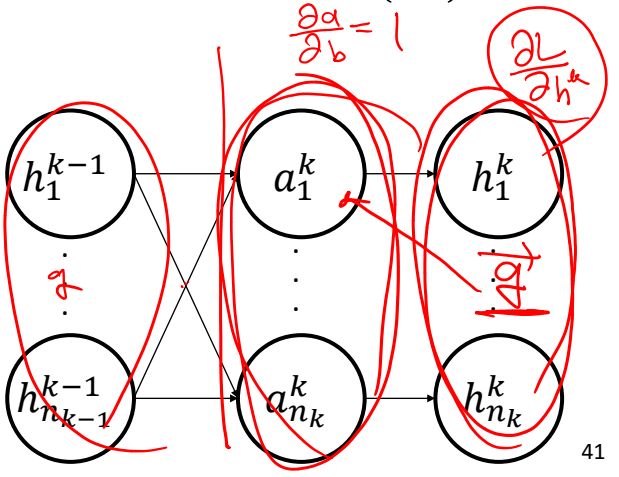
$\vec{a}^{(k)} = \vec{b}^{(k)} + \vec{W}^{(k)} \vec{h}^{(k-1)}$

$\vec{h}^{(k)} = f(\vec{a}^{(k)})$

$\frac{\partial a}{\partial b} = 1$

element-wise ×

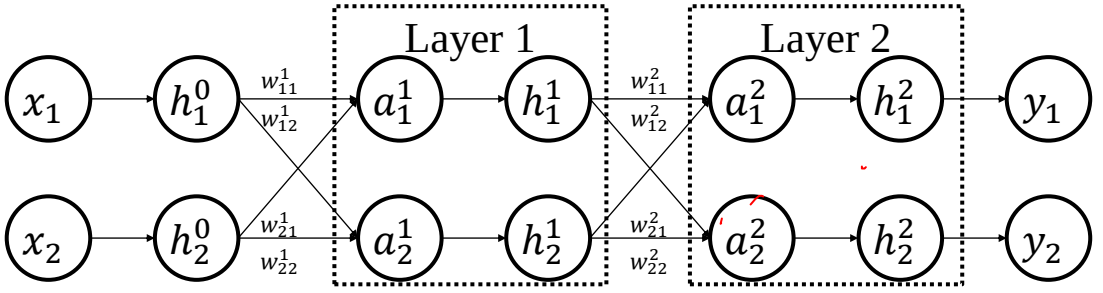
dot prod.



41

Backward propagation

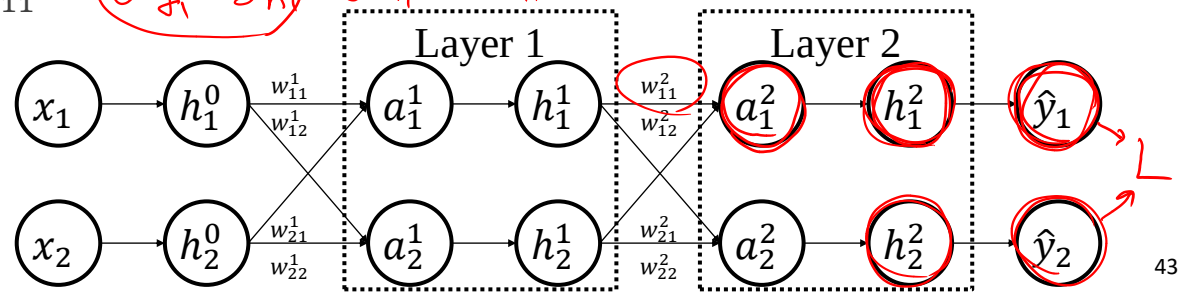
- $\vec{h}^{(k)} = \begin{bmatrix} h_1^{(k)} \\ h_2^{(k)} \end{bmatrix} = f(\vec{a}^{(k)}) = f\left(\begin{bmatrix} a_1^{(k)} \\ a_2^{(k)} \end{bmatrix}\right) = f\left(\vec{W}^{(k-1)\top} \vec{h}^{(k-1)}\right)$
- $\vec{W}^{(k)} = \begin{bmatrix} W_{11}^{(k)} & W_{12}^{(k)} \\ W_{21}^{(k)} & W_{22}^{(k)} \end{bmatrix}$



42

Backward propagation

- Loss function
  - $L(\vec{y}, \hat{\vec{y}}) = \frac{1}{2}\left\|\vec{y} - \hat{\vec{y}}\right\|_2^2 = \frac{1}{2}\left((y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2\right)$
- Update w_{11}^2?
  - $\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial h_1^2} \cdot \frac{\partial h_1^2}{\partial a_1^2} \cdot \frac{\partial a_1^2}{\partial w_{11}^2}$ $\left(\because \hat{y}_1 = h_1^2,\ \frac{\partial \hat{y}_1}{\partial h_1^2} = 1,\ \frac{\partial L}{\partial \hat{y}_1} = \frac{\partial L}{\partial h_1^2}\right)$

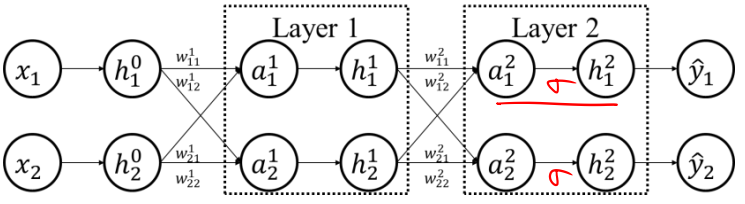


43

Backward propagation

$L(\vec{y}, \hat{\vec{y}}) = \frac{1}{2}((y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2)$

- $\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial h_1^2}\frac{\partial h_1^2}{\partial a_1^2}\frac{\partial a_1^2}{\partial w_{11}^2}$
- $\frac{\partial L}{\partial h_1^2} = \frac{\partial L}{\partial \hat{y}_1} = \hat{y}_1 - y_1$



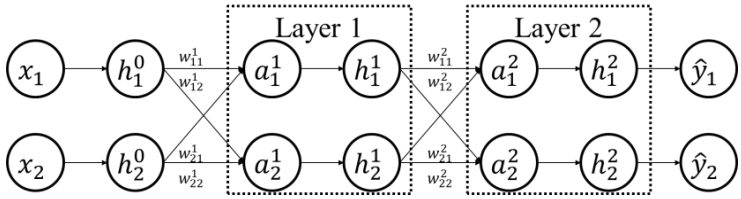
- Assume we use the sigmoid activation function: σ
- $\frac{\partial h_1^2}{\partial a_1^2} = \frac{\partial}{\partial a_1^2}\sigma(a_1^2) = \sigma(a_1^2)(1 - \sigma(a_1^2))$ $\left(h_1^2 = \sigma(a_1^2)\right)$
  $= h_1^2(1 - h_1^2)$

44

Backward propagation

- $\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial h_1^2}\frac{\partial h_1^2}{\partial a_1^2}\frac{\partial a_1^2}{\partial w_{11}^2}$
- $\frac{\partial a_1^2}{\partial w_{11}^2}$? $= h_1^1$
- $a_1^2 = w_{11}^2 h_1^1 + w_{21}^2 h_2^1$
- $\frac{\partial L}{\partial w_{11}^2} = (\hat{y}_1 - y_1)\cdot h_1^2(1-h_1^2)\cdot h_1^1 = (\hat{y}_1 - y_1)\cdot \hat{y}_1(1-\hat{y}_1)\cdot h_1^1$
- $w_{11}^2 \leftarrow w_{11}^2 - \varepsilon \frac{\partial L}{\partial w_{11}^2}$

current value; small constant

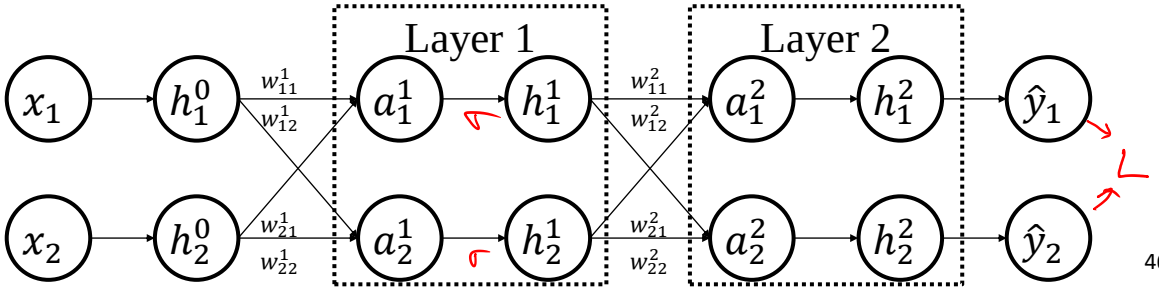


45

Backward propagation

- Update w_{11}^1?
- $\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial h_1^1}\cdot\frac{\partial h_1^1}{\partial a_1^1}\cdot\frac{\partial a_1^1}{\partial w_{11}^1}$

$\frac{\partial h_1^1}{\partial a_1^1}\cdot\frac{\partial a_1^1}{\partial w_{11}^1} = h_1^1(1-h_1^1)\cdot\frac{h_1^0}{x_1} = h_1^1(1-h_1^1)\cdot x_1$



46

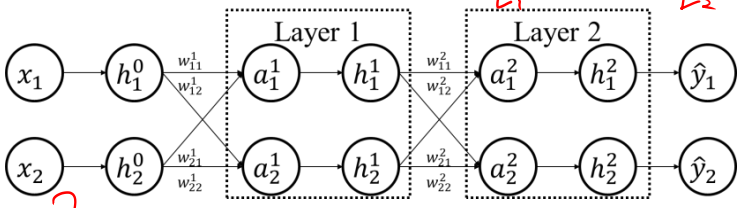
Backward propagation

$$L(\vec{y},\hat{\vec{y}}) = \frac{1}{2}\left((y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2\right)$$

L_1 L_2

- Let $L = L_1 + L_2$
  - $L_1 = \frac{1}{2}(y_1-\hat{y}_1)^2$
  - $L_2 = \frac{1}{2}(y_2-\hat{y}_2)^2$
- $\frac{\partial L}{\partial h_1^1} = \frac{\partial}{\partial h_1^1}(L_1+L_2) = \frac{\partial}{\partial h_1^1}L_1 + \frac{\partial}{\partial h_1^1}L_2$
- $\frac{\partial L_1}{\partial h_1^1} = \frac{\partial L_1}{\partial h_1^2}\cdot\frac{\partial h_1^2}{\partial a_1^2}\cdot\frac{\partial a_1^2}{\partial h_1^1}$

$= (\hat{y}_1-y_1)\cdot h_1^2(1-h_1^2)\cdot w_{11}^2$



$\frac{\partial a_1^2}{\partial h_1^1} = w_{11}^2 h_1^1 + w_{21}^2 h_2^1$

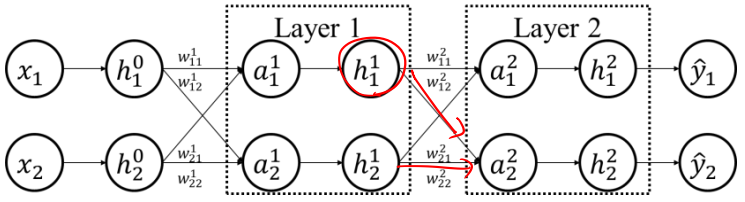
47

Backward propagation

$$L(\vec{y},\hat{\vec{y}}) = \frac{1}{2}\left((y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2\right)$$

- $\frac{\partial a_1^2}{\partial h_1^1}? = w_{11}^2$
- $a_1^2 = w_{11}^2 h_1^1 + w_{21}^2 h_2^1$
- $\frac{\partial L}{\partial h_1^1} = (\hat{y}_1-y_1)\hat{y}_1(1-\hat{y}_1)w_{11}^2$
- $\frac{\partial L_2}{\partial h_1^1} = \frac{\partial L}{\partial h_2^2}\cdot\frac{\partial h_2^2}{\partial a_2^2}\cdot\frac{\partial a_2^2}{\partial h_1^1}$

$= (\hat{y}_2-y_2)\hat{y}_2(1-\hat{y}_2)w_{12}^2$

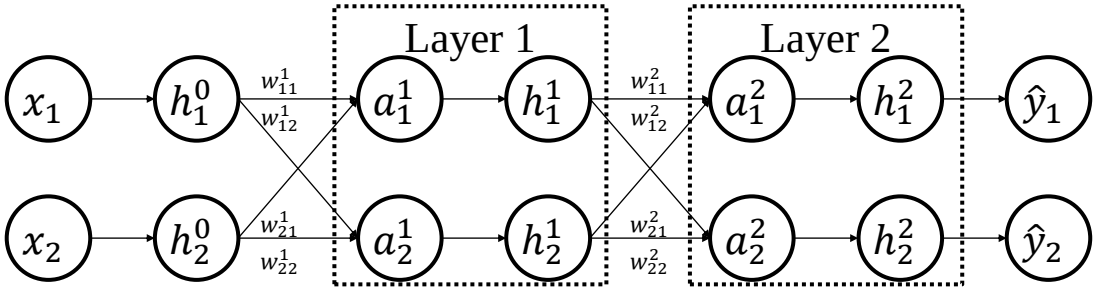


$\frac{\partial a_2^2}{\partial h_1^1} = w_{12}^2 h_1^1 + w_{22}^2 h_2^1$

48

Backward propagation

- $\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial h_1^1}\frac{\partial h_1^1}{\partial a_1^1}\frac{\partial a_1^1}{\partial w_{11}^1} = \left(\frac{\partial L_1}{\partial h_1^1} + \frac{\partial L_2}{\partial h_1^1}\right)\frac{\partial h_1^1}{\partial a_1^1}\frac{\partial a_1^1}{\partial w_{11}^1}$
- $= \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{11}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{12}^2\right) \cdot h_1^1(1-h_1^1)x_1$
- $w_{11}^1 \leftarrow w_{11}^1 - \varepsilon\frac{\partial L}{\partial w_{11}^1}$

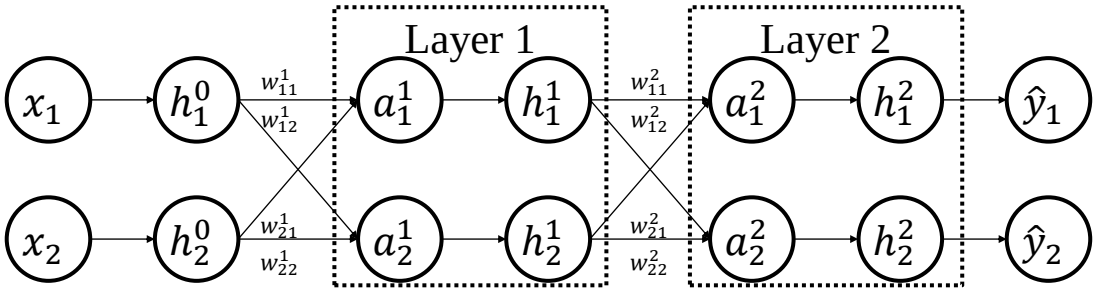


49

49

Backward propagation

- $\frac{\partial L}{\partial w_{11}^1} = \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{11}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{12}^2\right) \times h_1^1(1-h_1^1)x_1$
- $\frac{\partial L}{\partial w_{11}^2} = (\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)h_1^1$



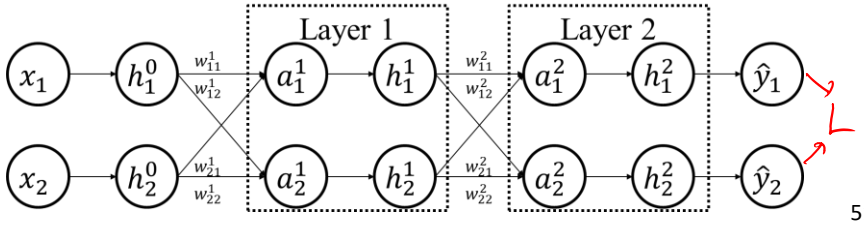
50

50

Backward propagation

- $\frac{\partial L}{\partial w_{11}^2} = (\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)h_1^1$
- $\frac{\partial L}{\partial w_{12}^2} = (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)h_1^1$
- $\frac{\partial L}{\partial w_{21}^2} = (\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)h_2^1$
- $\frac{\partial L}{\partial w_{22}^2} = (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)h_2^1$

$$\frac{\partial L}{\partial \vec{W}^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^2} & \frac{\partial L}{\partial w_{12}^2} \\ \frac{\partial L}{\partial w_{21}^2} & \frac{\partial L}{\partial w_{22}^2} \end{bmatrix}$$

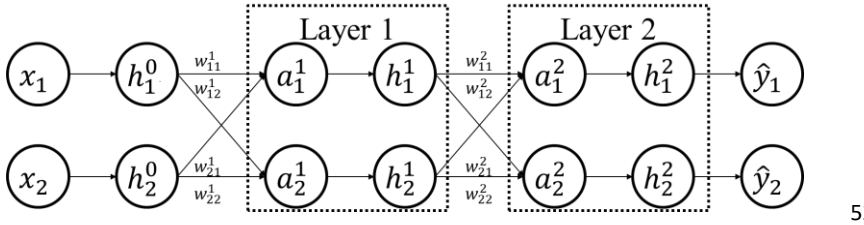


51

Backward propagation

- $\frac{\partial L}{\partial w_{11}^1} = \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{11}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{12}^2\right)h_1^1(1-h_1^1)x_1$
- $\frac{\partial L}{\partial w_{12}^1} = \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{21}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{22}^2\right)h_2^1(1-h_2^1)x_1$
- $\frac{\partial L}{\partial w_{21}^1} = \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{11}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{12}^2\right)h_1^1(1-h_1^1)x_2$
- $\frac{\partial L}{\partial w_{22}^1} = \left((\hat{y}_1 - y_1)\hat{y}_1(1-\hat{y}_1)w_{21}^2 + (\hat{y}_2 - y_2)\hat{y}_2(1-\hat{y}_2)w_{22}^2\right)h_2^1(1-h_2^1)x_2$

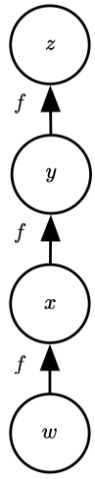
$$\frac{\partial L}{\partial \vec{W}^{(1)}} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^1} & \frac{\partial L}{\partial w_{12}^1} \\ \frac{\partial L}{\partial w_{21}^1} & \frac{\partial L}{\partial w_{22}^1} \end{bmatrix}$$



52

Repeated Subexpressions

- A computational graph that results in repeated subexpressions when computing the gradient.
  - $f: \mathbb{R} \to \mathbb{R}$
  - $x = f(w), y = f(x), z = f(y)$
  - $\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} = f'(y)f'(x)f'(w)$
    $= f'(f(y))f'(f(w))f'(w)$
    $f(x)$ $f(w)$
  - compute the value of $f(w)$ only once and store it in the variable x



53

53