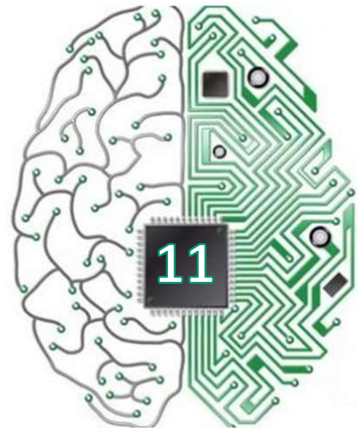
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#

Deep Learning

Unit-3: Artificial Neural Network (Part-II)

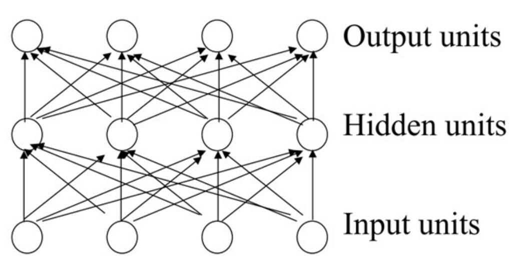


Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

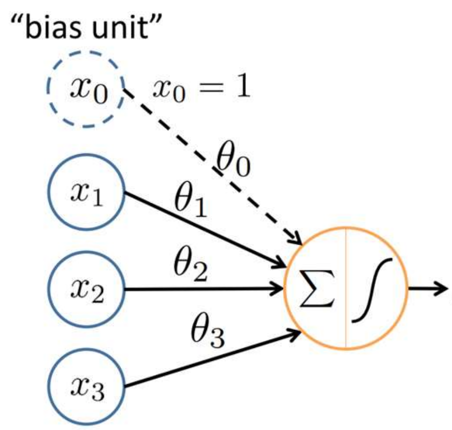
Neural Networks

- **Origins:** Algorithms that try to mimic the brain.
- Very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure
- Neural networks are made up of nodes or units, connected by links
- Each link has an associated weight and activation level
- Each node has an input function (typically summing over weighted inputs), an activation function, and an output



Layered feed-forward network

Neuron Model: Logistic Unit

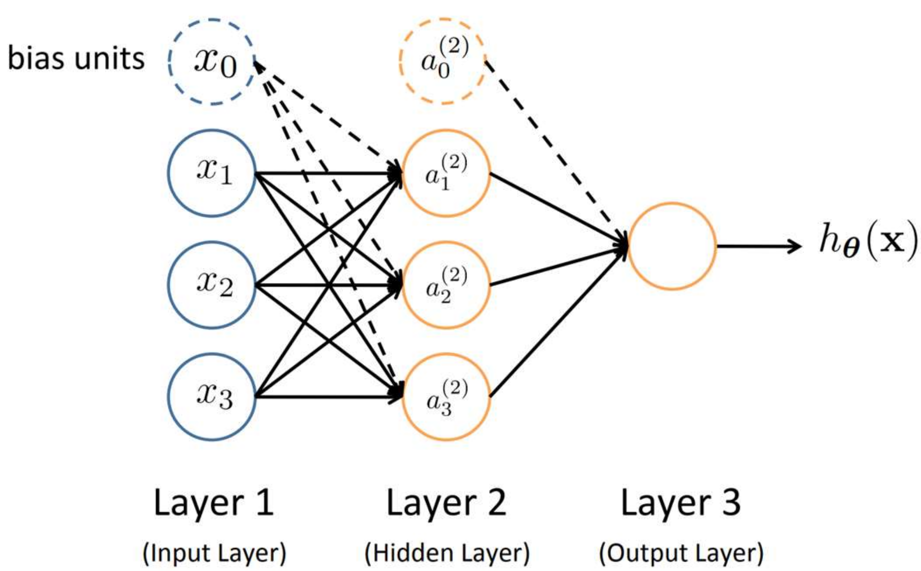


$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^\mathsf{T}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\mathsf{T}\mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

Neural Network



Feedforward Neural Network Structures

5

- They are called networks because they are composed of many different functions
- Model is associated with a directed acyclic graph describing how functions composed
  - E.g., functions $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ connected in a chain to form

  $f(x) = f^{(3)}[f^{(2)}[f^{(1)}(x)]]$
    - $f^{(1)}$ is called the first layer of the network
    - $f^{(2)}$ is called the second layer, etc
- These chain structures are the most commonly used structures of neural networks

Definition of Depth

6

- Overall length of the chain is the depth of the model
- The name *deep learning* arises from this terminology
- Final layer of a feedforward network is called the *output layer*

What are Hidden Layers?

- Behavior of other layers is not directly specified by the data
- Learning algorithm must decide how to use those layers to produce value that is close to y
- Training data does not say what individual layers should do
- Since the desired output for these layers is not shown, they are called *hidden layers*

What a Hidden Unit Does

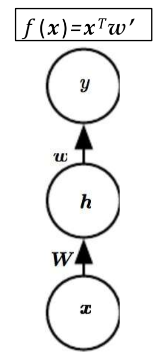
- Accepts a vector of inputs $\boldsymbol{x}$ and computes an affine transformation

  $z = W^T\boldsymbol{x} + \boldsymbol{b}$
- Computes an element-wise non-linear function $g(z)$
- Most hidden units are distinguished from each other by the choice of activation function $g(z)$
  - Examples: ReLU, Sigmoid and tanh, and other hidden units
- Design of hidden units is an active research area that does not have much guidance in theory

Linear vs Nonlinear functions

9

- If we choose both $f^{(1)}$ and $f^{(2)}$ to be linear, the total function will still be linear $f(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{w}'$
  - Suppose $f^{(1)}(\boldsymbol{x})= W^T x=\boldsymbol{h}$ and $f^{(2)}(\boldsymbol{h})=\boldsymbol{h}^T\boldsymbol{w}$
  - Then we could represent this function as
    $$f(\boldsymbol{x})=\boldsymbol{x}^T\boldsymbol{w}' \text{ where } \boldsymbol{w}'=W\boldsymbol{w}$$
- Since linear is insufficient, we must use a nonlinear function to describe the features
  - Use the design of neural networks by using a nonlinear activation function
    $$\boldsymbol{h}=g(W^T\boldsymbol{x}+\boldsymbol{c})$$



Activation Function

10

- In linear regression we used a vector of weights $\boldsymbol{w}$ and scalar bias b
  $$\boxed{f(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{x}^T\boldsymbol{w} + b}$$
  - This describes an affine transformation from an input vector to an output scalar
- Now we describe an affine transformation from a vector $\boldsymbol{x}$ to a vector $\boldsymbol{h}$, so an entire vector of bias parameters is needed
- Activation function g is typically chosen to be applied element-wise $\boldsymbol{h}_i=g(x^T W_{:,i}+c_i)$

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Logistic Sigmoid

11

- Prior to introduction of ReLU, most neural networks used logistic sigmoid activation

  $g(z)=\sigma(z)$

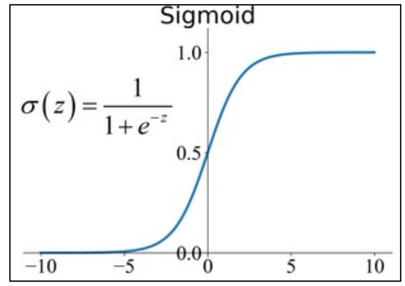
- Or the hyperbolic tangent

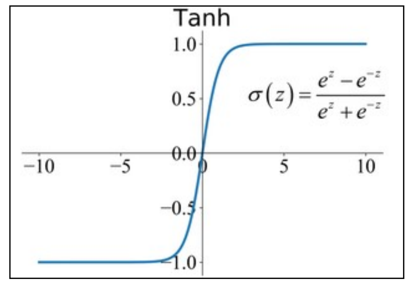
  $g(z)=\tanh(z)$

- These activation functions are closely related because

  $\tanh(z)=2\sigma(2z)-1$

- Sigmoid units are used to predict probability that a binary variable is 1

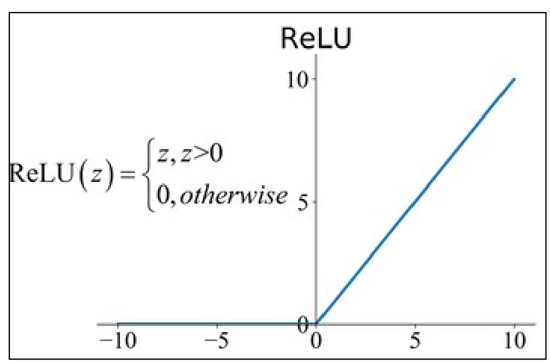




Rectified Linear Unit & Generalizations

12

- Rectified linear units use the activation function $g(z)=\max\{0,z\}$
  - They are easy to optimize due to their similarity to linear units
    - Only difference with linear units that they output 0 across half their domain
    - Derivative is 1 everywhere that the unit is active
    - Allows gradient computation that is far more useful than with activation functions that have second-order effects



Generalizations of ReLU

13

- Perform comparably to ReLU and occasionally perform better
- ReLU cannot learn on examples for which the activation is zero
- Generalizations guarantee that they receive a gradient everywhere

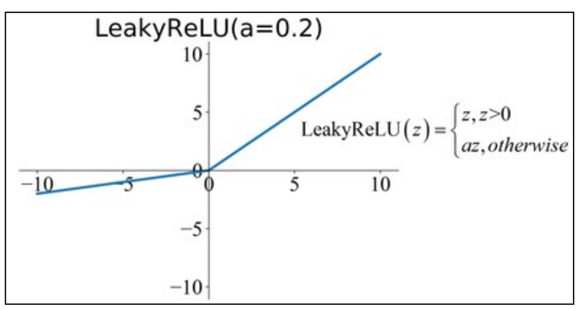
Three Generalizations of ReLU

14

- Three methods based on using a non-zero slope α_i when $z_i<0$:

$$h_i=g(z,\alpha)_i=\max(0,z_i)+\alpha_i \min(0,z_i)$$

1. Absolute-value rectification:
   - fixes $\alpha_i=-1$ to obtain $g(z)=|z|$
2. Leaky ReLU:
   - fixes α_i to a small value like 0.01
3. Parametric ReLU or PReLU:
   - treats α_i as a parameter



Other Hidden Units

15

- Many other types of hidden units possible, but used less frequently
  - Feed-forward network using $h = \cos(Wx + b)$
  - on MNIST obtained error rate of less than 1%
  - Radial Basis $\boxed{h_i = \exp\left(-\frac{1}{\sigma^2} ||W_{:,i} - \boldsymbol{x}||^2\right)}$
    - Becomes more active as x approaches a template $W_{:,i}$
  - Softplus $\boxed{g(a) = \zeta(a) = \log(1 + e^a)}$
    - Smooth version of the rectifier
  - Hard tanh
    - Shaped similar to tanh and the rectifier but it is bounded

$$g(a) = \max(-1, \min(1,a))$$

Is Differentiability necessary?

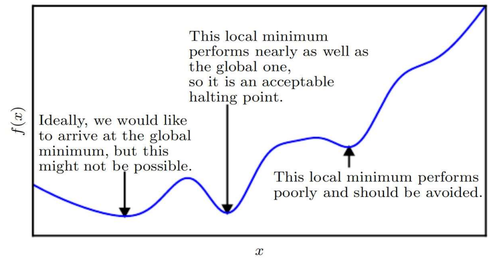
16

- Some hidden units are not differentiable at all input points
  - Rectified Linear Unit Function $g(z)=\max\{0,z\}$ is not differentiable at $z=0$
- Does this invalidate use in gradient-based learning
- In practice gradient descent still performs well enough for these models to be used

Differentiability ignored

17

- Neural network training
  - not usually arrives at a local minimum of cost function

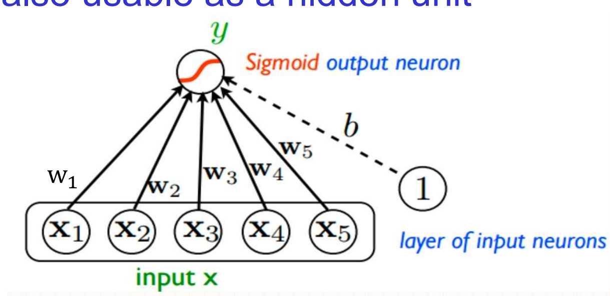


- Do not expect training to reach a point where gradient is 0,
  - Accept minima to correspond to points of undefined gradient
- Hidden units that are not differentiable are usually for only a small number of points

Role of Output Units

18

- Any output unit is also usable as a hidden unit



- A feedforward network provides a hidden set of *features* $\boldsymbol{h} = f(\boldsymbol{x}; \boldsymbol{\theta})$
- Role of output layer is to provide some additional transformation from the features generated before the output layer

Types of output units

19

1. Linear units: no nonlinearity
   – for Gaussian Output distributions
2. Sigmoid units
   – for Bernoulli Output Distributions
3. Softmax units
   – for Multinoulli Output Distributions
4. Other Output Types
   – Not direct prediction of y but provide parameters of distribution over y

Architecture

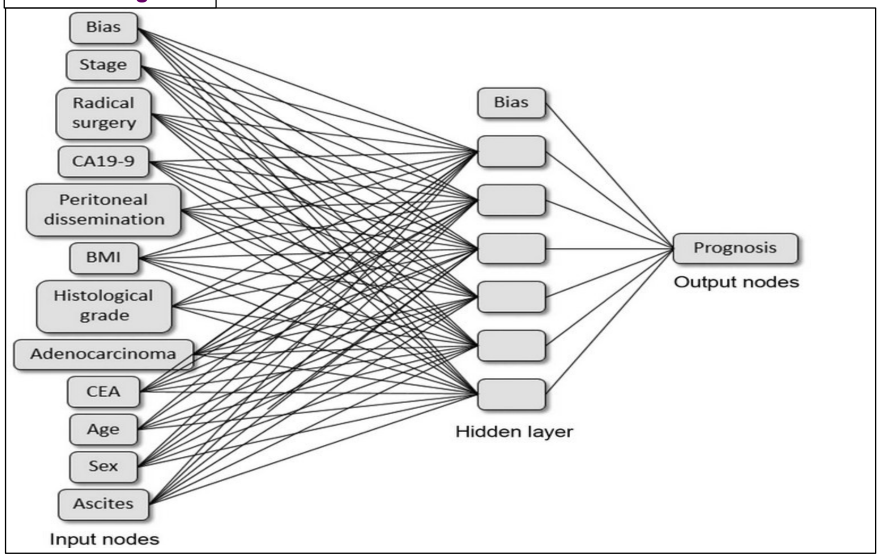
20

- *Architecture* refers to the overall structure of the network:
  – How many units should it have?
  – How are the units connected to each other?
- Most neural networks are organized into units called *layers*
  – Most neural network architectures arrange these layers in a chain structure
  – Each layer a function of the layer that proceeded it
- Main architectural consideration
  - Choice of depth of network
  - Choice of width of each layer

Specific Application Architectures

21

Cancer Prognosis



Architecture for multi-class classification

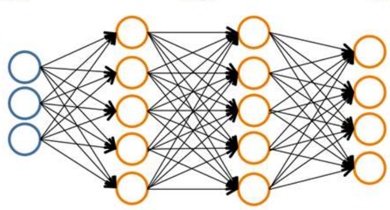
22


Pedestrian


Car


Motorcycle


Truck



$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ when pedestrian

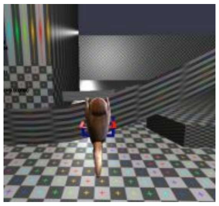
$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car

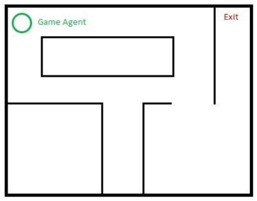
$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ when truck

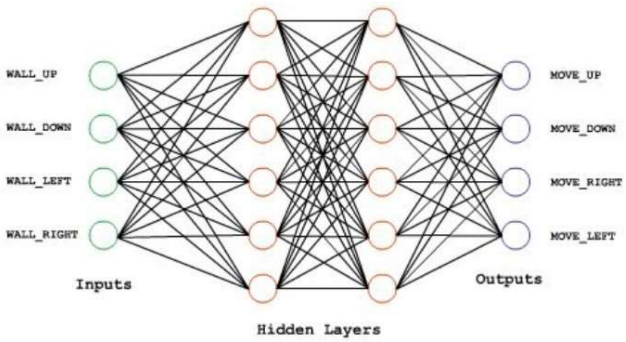
An Architecture for Game Design

23





Maze Game Recording Output File							
Inputs: Wall Directions				Outputs: Game Agent Direction			
UP	DOWN	LEFT	RIGHT	UP	DOWN	LEFT	RIGHT
1	0	0	0	0	1	0	0
1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1
0	0	1	0	0	0	0	1



Theoretical Results

24

- Mathematical theory regarding artificial neural networks
  - Linear versus Nonlinear Models
  - Universal Approximation Theorem (UAT)
- No Free Lunch Theorem
  - There is no universal procedure for examining a training set of samples and choosing a function that will generalize to points not in training set
- Size of network

Summary/Implications of UAT

25

- A feedforward network with a single layer is sufficient to represent any function
- However, the layer may be unreasonably large and may fail to generalize well
- Using deeper models can reduce the number of units required and reduce generalization error

Advantage of Deeper Networks

26

- Deep networks usually have
  – Far fewer units in each layer
  – Far fewer parameters
  – Often generalize well on the test set
  – But are usually more difficult to optimize
- Best network architecture must be found via experimentation guided by validation error

Other Architectural Considerations

27

- Specialized architectures can have a *significant impact!*
- Convolutional Networks
  - Used for computer vision
- Recurrent Neural Networks
  - Used for sequence processing
  - Have their own architectural considerations
  - Handles stateful problems
    - What is a stateful problem?
- Skipping: can skip from layer i to layer $i+2$ or higher
  - During learning, this makes it easier for a gradient to flow from output layers to a layer nearer its input

Connecting a Pair of Layers

28

- Consider the default neural network layer described by a linear transformation via a matrix W
- Every input unit connected to every output unit
- Specialized networks have fewer connections
  - Each unit in input layer is connected to only small subset of units in output layer
  - Reduces number of parameters and computation for evaluation
  - E.g., CNNs use specialized patterns of sparse connections that are effective for computer vision

Training the Network

29

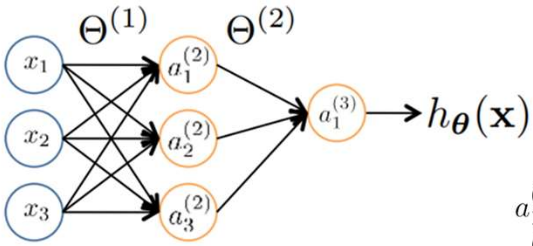
- In network training we drive $f(x)$ to match $f^*(x)$
- Training data provides us with noisy, approximate examples of $f^*(x)$ evaluated at different training points
- Each example accompanied by label $y \approx f^*(x)$
- Training examples specify directly what the output layer must do at each point x
  - It must produce a value that is close to y

Feed-Forward Process

30

- Input layer units are set by some exterior function (think of these as **sensors**), which causes their output links to be **activated** at the specified level
- Working forward through the network, the **input function** of each unit is applied to compute the input value
  - Usually this is just the weighted sum of the activation on the links feeding into this node
- The **activation function** transforms this input function into a final value
  - Typically this is a **nonlinear** function, often a **sigmoid** function corresponding to the "threshold" of that node

Neural Network



$a_i^{(j)} =$ "activation" of unit i in layer j

$\Theta^{(j)} =$ weight matrix controlling function mapping from layer j to layer $j + 1$

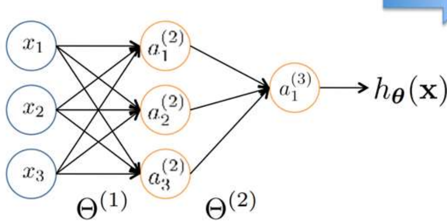
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

If network has s_j units in layer j *and* s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j+1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4} \qquad \Theta^{(2)} \in \mathbb{R}^{1\times 4}$$

Vectorization

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$
$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$
$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$
$$\text{Add } a_0^{(2)} = 1$$
$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$
$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Exercise

33

- Which of the following activation functions can lead to vanishing gradients?
  - (i) ReLU
  - (ii) Tanh
  - (iii) Leaky ReLU
  - (iv) None of the above

- You are solving the binary classification task of classifying emails as spam vs. not spam. You have designed a NN with a single output neuron. Let the net input to this neuron be z. The final output of your network, $\hat{y}$, is given by:
  $\hat{y} = \sigma(ReLU(z))$ and you classify all inputs with a final value $\hat{y} \geq 0.5$ as cat images.
  What problem are you going to encounter?

34

Questions?