MONASH University
Information Technology

FIT5201

Data Analysis Algorithms

Week 9 – Neural Networks

Outline

- Refresher about Neural Network
- Network Training

Neural Network, a little bit refresher

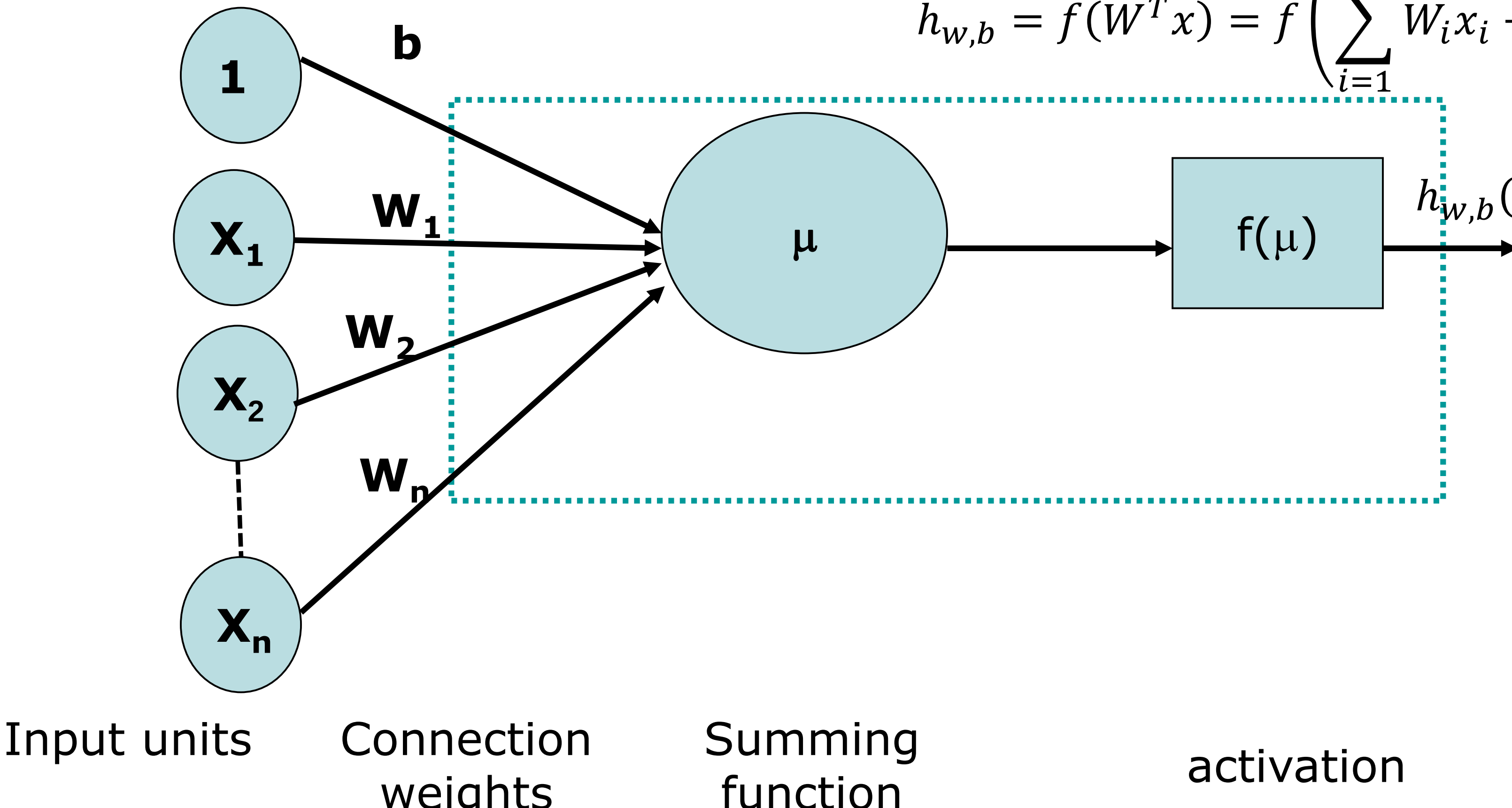
- Not new
- Human intelligence?

Why Neural Networks

- More advanced neural networks such as deep learning, convolutional, networks are all built on top of the basic neural networks
- Highly adoptable for many uses
  - Image recognition
    - > Automatic number plate recognition
  - Voice recognition
    - > Siri, OK Google
  - Handwriting recognition
    - > Post code on envelops
  - Self-driving cars

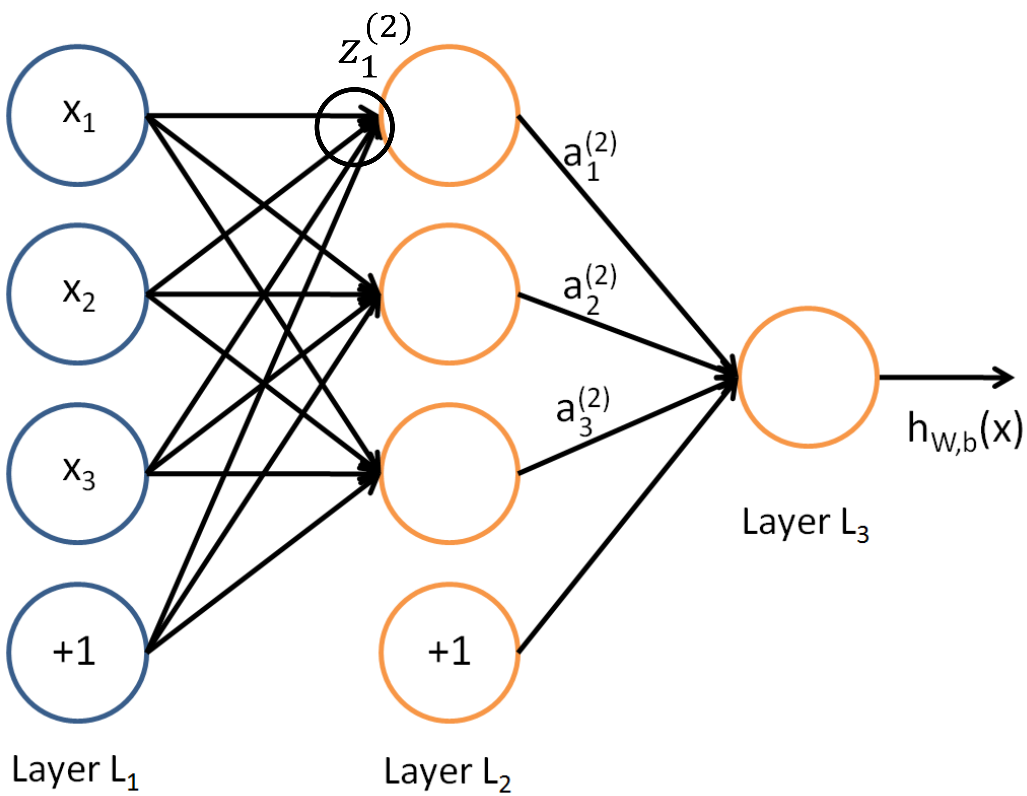
Model of a Neuron

$$h_{w,b} = f(W^T x) = f\left(\sum_{i=1}^{n} W_i x_i + b\right)$$

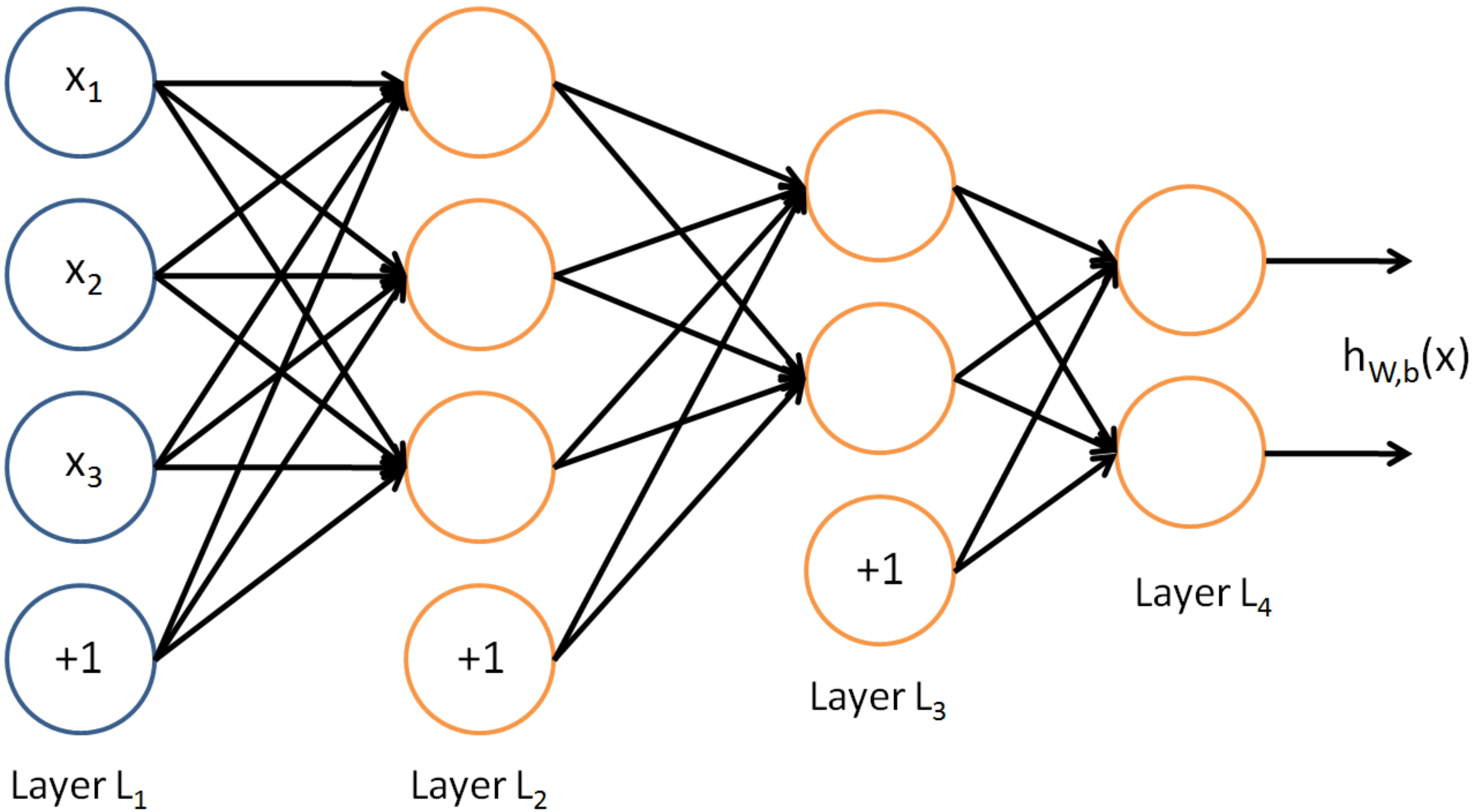


Neural Network

A collection of Neurons connected together



Neural Networks with Multiple Outputs



The power of neural networks

- The model class corresponding to neural networks can represent almost any function (given some minor conditions) provided the network has a sufficiently large number of hidden units
  - Have been widely studied
  - 9 layer can solve many low-level intelligence task pretty well

The power of neural networks

- Classification problem
  - Approximate the target decision boundary to any required precision
- Regression problem:
  - Approximate the target function to any precision
- Price:
  - Large number of neurons in the hidden layers
  - Large number of parameters
  - Tend to over fit the training data

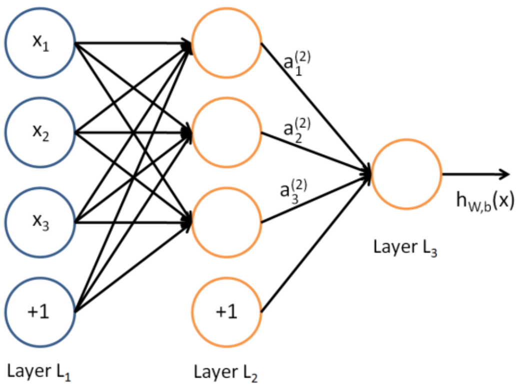
The power of neural networks

- Methods to prevent overfitting
  - Use a large training data
  - Use regularization methods
  - Use deep architecture instead of wide and shallow architecture
    - > Given same number of neurons, deep design performs better
    - > Given same performance, deep architecture needs smaller number of neurons
- A toy neural network
  - http://playground.tensorflow.org

Outline

- Recall about Neural Network
- Network Training

3-Layer Neural Network



$$\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)})$$

$$\begin{aligned}
a_1^{(2)} &:= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &:= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &:= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{\boldsymbol{\theta}}(\boldsymbol{x}) &:= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}$$

- W_{ij}^{l}: denote the weight associated with the connection between unit j in layer l and unit i in layer $l+1$
- $a_i^{(l)}$: the output of the i^{th} neuron in layer l
- $z_i^{(l)}$: the total weighted sum of inputs to the i^{th} neuron in layer l

$$z_i^l := \sum_{j=1}^{n} W_{ij}^{l-1} x_j + b_i^{l-1} \qquad a_i^{(l)} := f(z_i^{(l)}).$$

Feedforward Function

- Put $\boldsymbol{a}^{(1)} = \boldsymbol{x}$
- Then given layer l's activations $\boldsymbol{a}^{(l)}$, we can compute layer $(l+1)$'s activations $\boldsymbol{a}^{(l+1)}$ as

$$\boldsymbol{z}^{(l+1)} = \boldsymbol{w}^{(l)}\boldsymbol{a}^{(l)} + \boldsymbol{b}$$
$$\boldsymbol{a}^{(l+1)} = f\left(\boldsymbol{z}^{(l+1)}\right)$$

Training Objective

- Provide input values x_i and obtain an output y_i
- Find the optimal values for the weights that provide the correct output for the given input
  - Can be used for regression or classification

Training Objective

- Provide input values x_i and obtain an output y_i
- Find the optimal values for the weights that provide the correct output for the given input
  - Can be used for regression or classification
  - Can have one output or multiple outputs
  - There is a natural choice of both output unit activation function and matching error function

Training Objective

- Natural choice of both output unit activation function and matching error function
  - For regression
    - > one output
    - > linear outputs (i.e., identity activation function) $h_\theta(x) = z_1^{(n_l)}$
    - > sum of square error to evaluate the model
  - For binary classification
    - > one output (or two)
    - > logistic sigmoid activation function (or two outputs with softmax output activation function)
    - > cross-entropy error function
  - For K-class classification,
    - > K output
    - > softmax output activation function
    - > multiclass cross-entropy error function

Training Objective

- Natural choice of both output unit activation function and matching error function
  - For regression
    - > one output
    - > linear outputs (i.e., i ... $= z_1^{(n_l)}$
    - > sum of squares e
  - For binary classificat
    - > one output (or two)
    - > logistic sigmoid activatio ... outputs with softmax output activation function)
    - > cross-entropy error function
  - For K-class classification,
    - > K output
    - > softmax output activation function
    - > multiclass cross-entropy error function

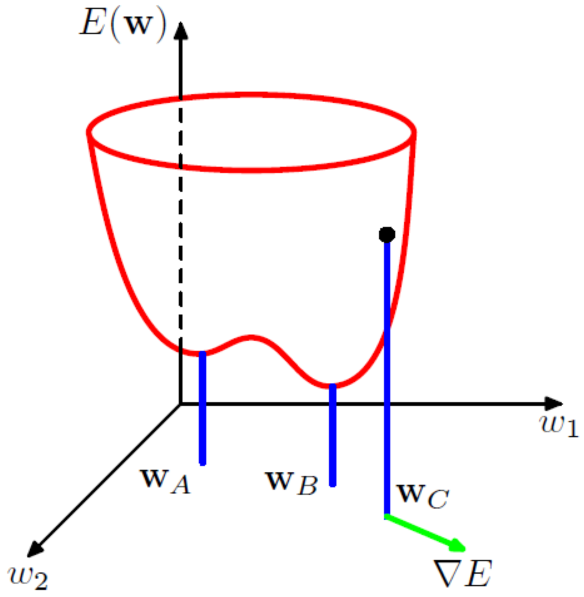
Refer to the hand written materials

Parameter Optimization

- For a 3 layer Neural Network

$$\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)})$$

- Find optimal value for θ that minimises the error $E(\theta)$
- Use gradient descent
- Start from regression as an example



Gradient Descent

- For regression problems the cost function to minimize is

$$J(\boldsymbol{\theta}) := \underbrace{\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}||h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(n)}) - \boldsymbol{y}^{(n)}||^2}_{E(\boldsymbol{\theta})} + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2$$

  – Weight decay is usually not applied to the bias term, why?

- Initialize all $w_{ij}^{(l)}$ and $b_i^{(l)}$ to random values near zero.
  – If initialized with zeros or equal values, the hidden units will be learning the same function WRT the input variables

- One iteration of GD updates the parameters as follows:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \eta \frac{\partial}{\partial W_{ij}^{(l)}} J(\boldsymbol{\theta})$$

$$b_i^{(l)} = b_i^{(l)} - \eta \frac{\partial}{\partial b_i^{(l)}} J(\boldsymbol{\theta})$$

Where η is the learning rate

Gradient Descent

- How to derive gradients?
- Back propagation algorithm!
  - For general idea, refer to https://www.youtube.com/watch?time_continue=1&v=An5z8lR8asY
  - For detailed algorithm, refer to handwritten material