MONASH University
Information Technology

FIT5201

Data Analysis Algorithms

Week 8 – Document Clustering and Introduction to Neural Networks

Document Clustering

- Given a collection of documents $\{d_1, d_2, \dots, d_N\}$ we would like to partition them into K clusters.
- Document representation
  - Each document is made of some text
  - *bag of word* representation of the document
  - We treat a document as a *set* of words in its text irrespective of their positions
  - Also, we assume the words appearing in our collection of documents come from a dictionary denoted by $\mathcal{A}$

Bag of Words

```
(1) John likes to watch movies. Mary likes movies too.
```

```
(2) John also likes to watch football games.
```

```
[
    "John",
    "likes",
    "to",
    "watch",
    "movies",
    "Mary",
    "too",
    "also",
    "football",
    "games"
]
```

```
(1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
(2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
```

Understanding the Model in Alexandria: an example

- d_1=this one has a little star
- d_2=this one has a little car
- d_3=I would not like them here or there
- d_4=I would not like them anywhere
- d_5=I do not like green eggs and ham

- Assume we know the clusters beforehand (in reality we don't)
  - K=2 (two clusters from two books)
  - C_1=(d_1, d_2), C_2=(d_3, d_4, d_5)
  - $\varphi_1 = 0.4$ (2/5), $\varphi_2 = 0.6$ (3/5)
  - Dictionary for C_1=(this, one, has, little, star, car)
  - μ_1=(2/10, 2/10, 2/10, 2/10, 1/10, 1/10) = (0.2, 0.2, 0.2, 0.2, 0.1, 0.1)
  - Dictionary for C_2=(I, would, not, like, them, here, there, anywhere, do, green, eggs, ham)
  - μ_2=(0.15, 0.1, 0.15, 0.15, 0.1, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05)

Quotes from Dr Suess's books *One Fish, Two Fish* and *Green Eggs and Ham*

Generating Words

- $\mathcal{A}$ ={this, one, has, little, star, car, I, would, not, like, them, here, there, anywhere, do, green, eggs, ham}
- P(this, one, has, little, star) = P(this)P(one)P(has) P(little) P(star)
- $P(d|k) = \prod_{w \in d} P(w|k) = \prod_{w \in \mathcal{A}} P(w|k)^{c(w,d)}$

Generative Model

- For each document d_n
  - Toss the K-face dice (with the probability parameter φ) to choose the face k (i.e., the cluster) that the n^{th} document belongs to
  - For each word placeholder in the document d_n
    - > Generate the word by tossing the dice (with the probability parameter μ_k) corresponding to the face k

- Parameters:
  - The clusters proportion $\varphi = (\varphi_1, \varphi_2, \dots, \varphi_K), \varphi_k \geq 0, \sum_{k=1}^{K} \varphi_k = 1$
  - The word proportion $\mu_k = \left(\mu_{k,1}, \mu_{k,2}, \dots, \mu_{k,|\mathcal{A}|}\right), \mu_{k,w} \geq 0, \sum_{w \in \mathcal{A}} \mu_{k,w} = 1$
  - These are constraints which allow us to use Lagrange model to learn the parameters

Generative Model

- The probability of generating a document and its cluster (k, d) is

$$p(k, d) = p(k)p(d|k) = \varphi_k \prod_{w \in d} \mu_{k,w} = \varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d)}$$

- $c(w, d)$ is the number of occurrences of the word w in the document d
- In practice,
  - The document cluster labels are not given to us

Complete Data

- Documents $\{d_1, d_2, \dots, d_N\}$
- We use latent variables $\boldsymbol{z}_n$ to denote the cluster assignments for n^{th} document
- $\boldsymbol{z}_n = (z_{n1}, z_{n2}, \dots, z_{nK})$
  - $z_{nk} = \begin{cases} 1, & d_n \in C_k \\ 0, & d_n \notin C_k \end{cases}$
  - Only one element in z_{nk} is 1. The rest are zero

Complete Data

$$p(d_1, z_1, \dots, d_N, z_N) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d_n)} \right)^{z_{nk}}$$

$$z_{nk} = \begin{cases} 1, & d_n \in \mathcal{C}_k \\ 0, & d_n \notin \mathcal{C}_k \end{cases}$$

- With the constraint that $\sum_{k=1}^{K} \varphi_k = 1$ and $\sum_{w \in \mathcal{A}} \mu_{k,w} = 1$
- Use Lagrange model to solve the parameters
  - Constrained optimization: convert it to unconstrained optimization problems which can be solved either find a solution analytically or use an iterative algorithm to find a solution
  - Lagrange model

Complete Data

- Use Lagrange model to solve the parameters
  - Constrained optimization: convert it to unconstrained optimization problems which can be solved either finding a solution analytically or using an iterative algorithm to find a solution
  - Lagrange multipliers

$$\begin{array}{ll} \underset{x}{\text{maximise}} & f(x) \\ \text{subject to} & g_i(x) = 0 \quad i = 1, \ldots, m \end{array}$$

    > Equality constraints

$$\mathcal{L}(x, \lambda_1, \ldots, \lambda_m) := f(x) - \lambda_1 g_1(x) - \ldots - \lambda_m g_m(x)$$

    > The stationary points for $f(x)$ are ensured to be the stationary points for the new function, but not conversely

Complete Data…

- Through the Lagrange multiplier on Maximum Likelihood Function

Mixing components: $\varphi_k = \frac{N_k}{N}$ where $N_k = \sum_{n=1}^{N} z_{nk}$

Word proportion parameters: $\mu_{kw} = \frac{\sum_{n=1}^{N} z_{nk} c(w, d_n)}{\sum_{w\prime \in \mathcal{A}} \sum_{n=1}^{N} z_{nk} c(w\prime, d_n)}$

Incomplete Data and EM

$$p(d_1, \ldots, d_N) = \prod_{n=1}^{N} p(d_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \left(\varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d_n)} \right)$$

- Hard to derive the analytical solutions
- Resort to EM algorithm

Refreshment about GMMs

- Training objective: find maximum likelihood solution for models having latent variables.
  - Observed data X, Latent variable Z, set of model parameters θ
  - Log likelihood function

$$\ln p(X|\theta) = \ln \sum_{Z} p(X, Z|\theta)$$

Refreshment about GMMs

$$\gamma(z_{nk}) = p(z_n = k|x_n) = \frac{\varphi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \varphi_j N\left(x_n|\mu_j, \Sigma_j\right)}$$

- $\gamma(z_{nk})$: posterior probability once we observed x_n
- $\sum_{k=1}^{K} \gamma(z_{nk}) = 1$
- Partial assignment or soft assignment
- φ_k prior probability of $z_n = k$
- We do simultaneously
  - Cluster prediction and parameter estimation
  - Use iterative Expectation Maximisation (EM)

Refreshment about GMMs

- Training objective: find maximum likelihood solution for models having latent variables.
  - Observed data X, Latent variable Z, set of model parameters θ
  - Log likelihood function

$$\ln p(X|\theta) = \ln \sum_{Z} p(X, Z|\theta)$$

- Algorithm:
  - Choose an initial setting for the parameters θ^{old}
  - While convergence is not met:
    > **E Step**: Evaluate $p\left(Z|X, \theta^{old}\right)$

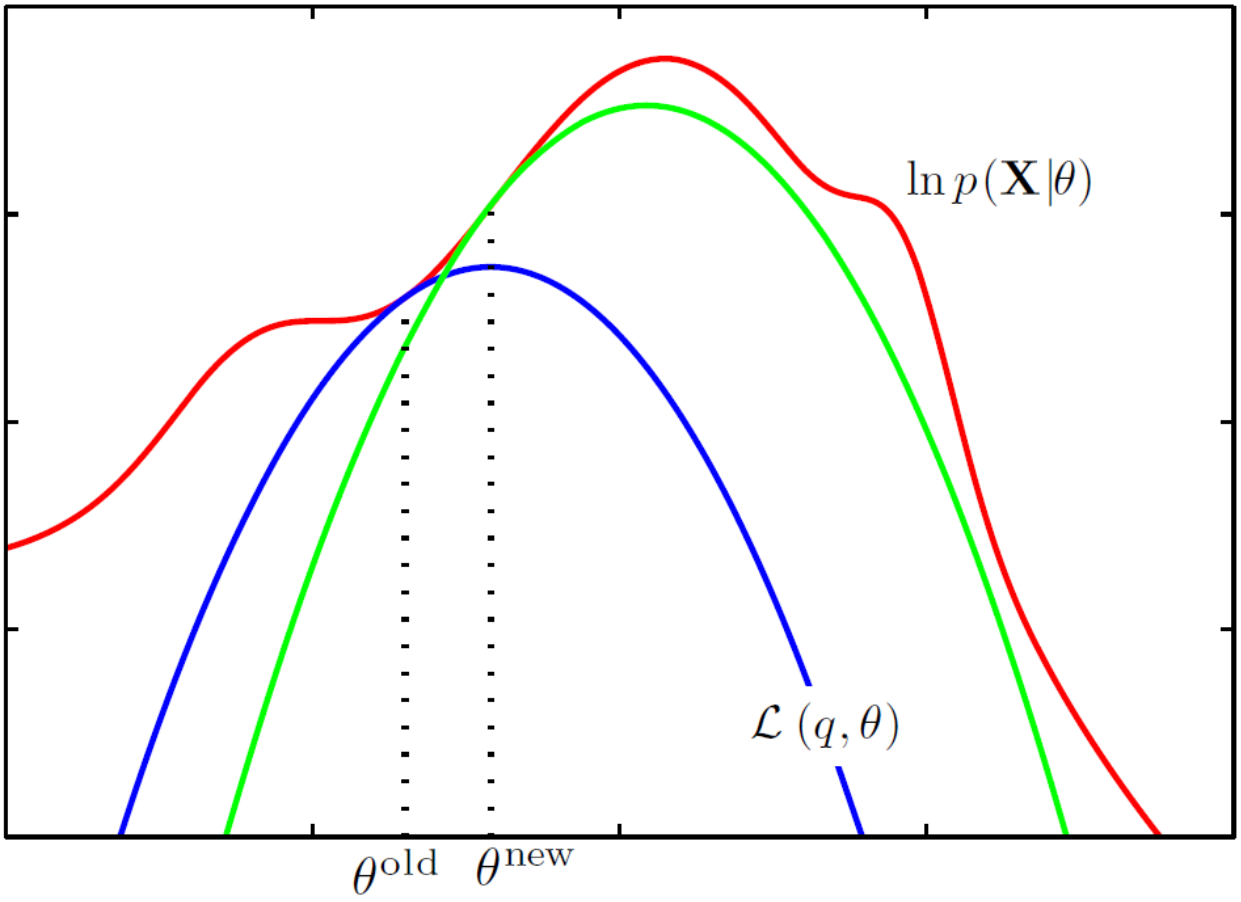
    > **M Step**: Evaluate θ^{new} given by

$$\theta^{new} \leftarrow \arg\max_{\theta} \underbrace{\sum_{Z} p\left(Z|X, \theta^{old}\right) \ln p(X, Z|\theta)}_{Q\left(\theta, \theta^{old}\right)}$$

    > $\theta^{old \leftarrow \theta^{new}}$

Refreshment about GMMs

- Is each iteration guaranteed to increase the log likelihood function?
- What's the relationship between the Q function and log likelihood function?



Incomplete Data and EM

$$p(d_1, \dots, d_N) = \prod_{n=1}^{N} p(d_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \left(\varphi_k \prod_{w \in \mathcal{A}} \mu_{k,w}^{c(w,d_n)} \right)$$

- Q function

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &:= \sum_{n=1}^{N} \sum_{k=1}^{K} p(z_{n,k} = 1 | d_n, \boldsymbol{\theta}^{\text{old}}) \ln p(z_{n,k} = 1, d_n | \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} p(z_{n,k} = 1 | d_n, \boldsymbol{\theta}^{\text{old}}) \left(\ln \varphi_k + \sum_{w \in \mathcal{A}} c(w, d_n) \ln \mu_{k,w} \right) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{n,k}) \left(\ln \varphi_k + \sum_{w \in \mathcal{A}} c(w, d_n) \ln \mu_{k,w} \right) \end{aligned}$$

$$\gamma(z_n, k) := p(z_{n,k} = 1 | d_n, \boldsymbol{\theta}^{\text{old}})$$

Incomplete Data and EM

$$\varphi_k = \frac{N_k}{N} \text{ where } N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

$$\mu_{kw} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})\, c(w, d_n)}{\sum_{w' \in \mathcal{A}} \sum_{n=1}^{N} \gamma(z_{nk})\, c(w', d_n)}$$

- Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}} = (\boldsymbol{\varphi}^{\text{old}}, \boldsymbol{\mu}_1^{\text{old}}, \ldots, \boldsymbol{\mu}_K^{\text{old}})$
- While the convergence is not met:
  - **E step:** Set $\forall n, \forall k : \gamma(z_{n,k})$ based on $\boldsymbol{\theta}^{\text{old}}$
  - **M Step:** Set $\boldsymbol{\theta}^{\text{new}}$ based on the above equations
  - $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}}$

Other Methods for Document Clustering

- Other methods can be used to encode the documents, e.g., TF-IDF
- Use Euclidian distance to measure the similarity between documents/cluster vectors
- Can use K-Means to cluster the documents into K clusters
  - What we do in the tutorial

Neural Networks

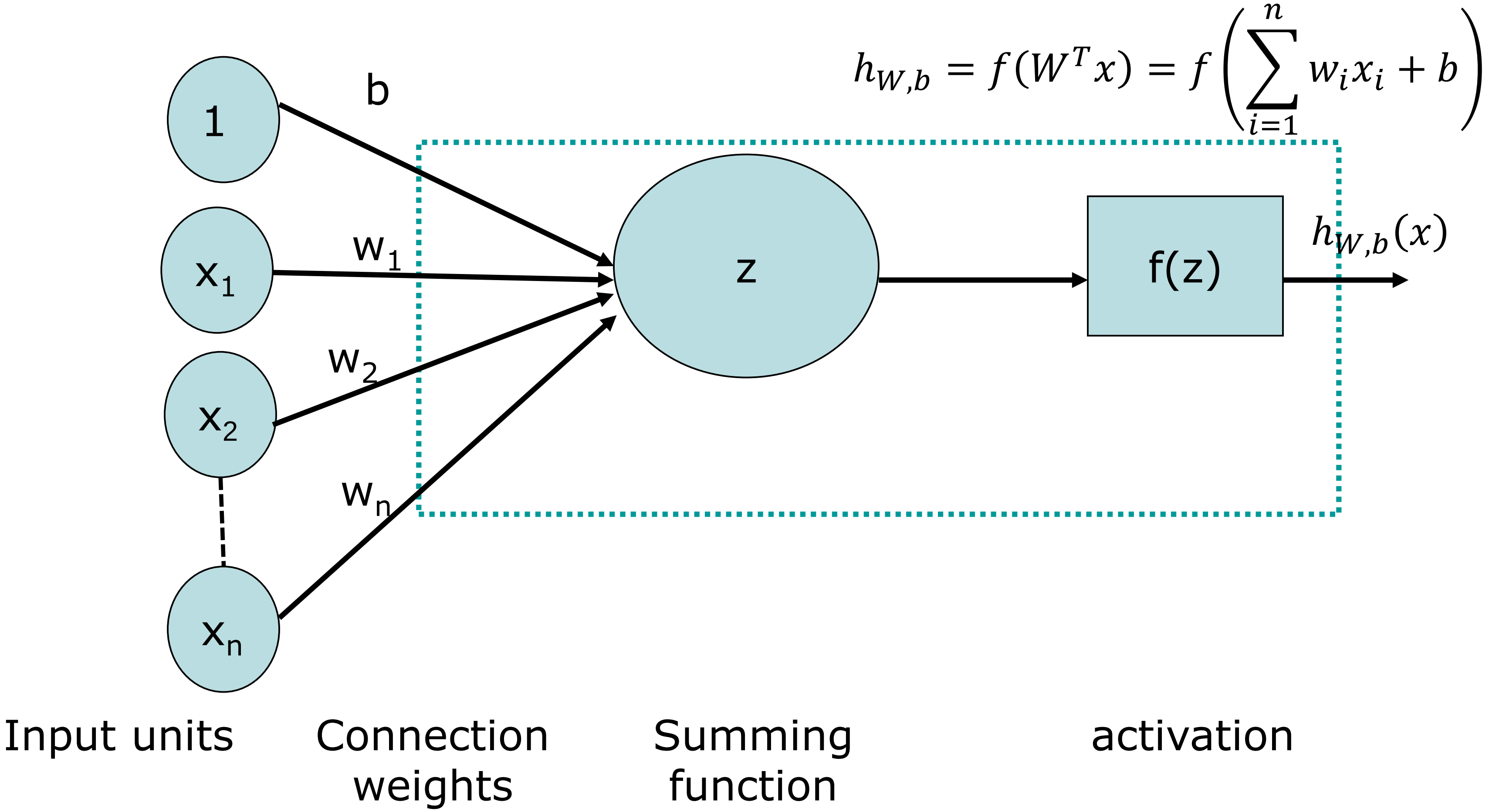
What is a Neural Network?

- Novelty:
  - The key element of this paradigm is the novel structure of the information processing system.
  - It is composed of a large number of highly interconnected processing elements (neurons).

Why Neural Networks

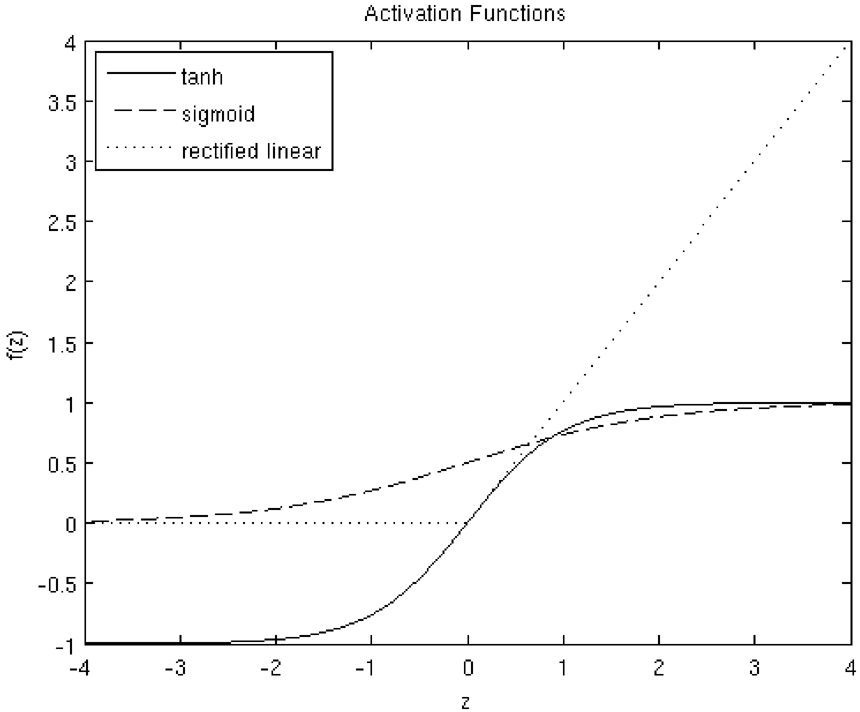
- Highly adoptable for many uses
  - Image recognition
    - > Automatic number plate recognition
  - Voice recognition
    - > Siri, OK Google
  - Handwriting recognition
    - > Post code on envelops
  - Self-driving cars
- More advanced neural networks such as deep learning, convolutional networks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

Model of a Neuron



Model of a Neuron…

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



- Activation function
  - Usually use the sigmoid function
    > $\sigma(z) = \frac{1}{1+e^{-z}}$
  - Other possibilities:
    > $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
      - Rescaled sigmoid function

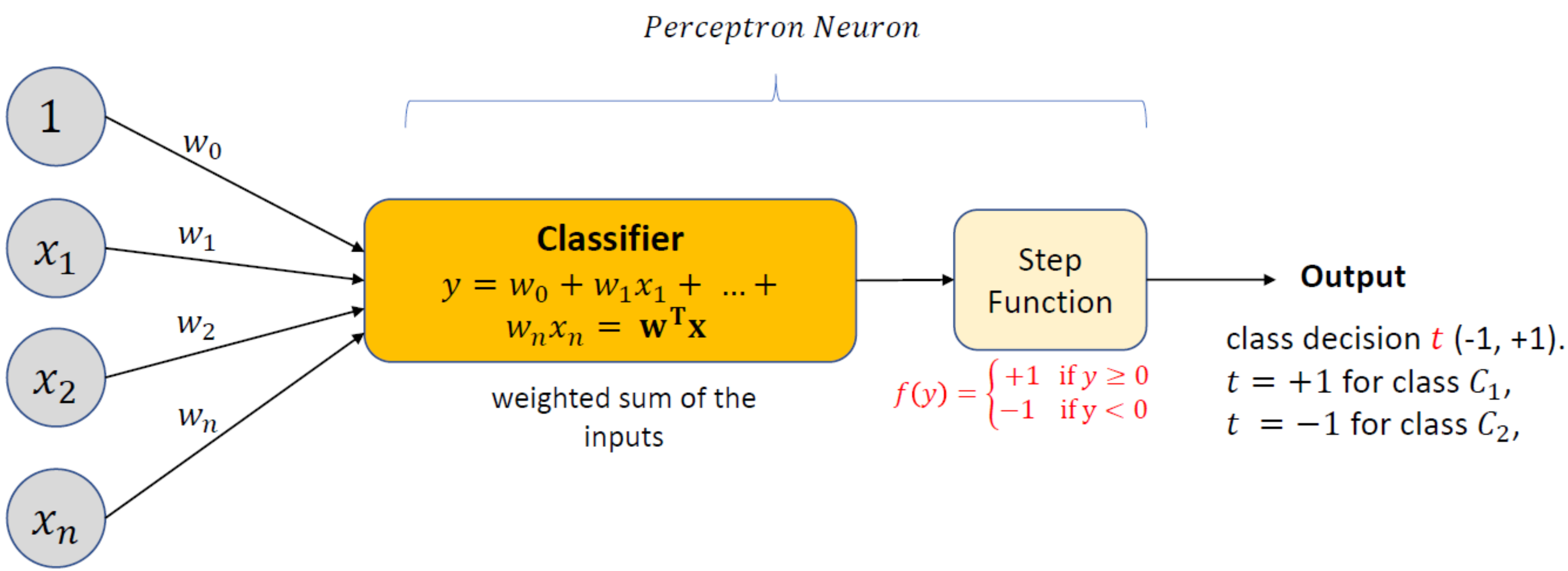
    > Rectified linear function
      - max(0,x)
      - Sparse activation

    > ….

    > What will happen if the activation function is linear function to the inputs?
  - Depend on the nature of the data and the assumed distribution of target variables

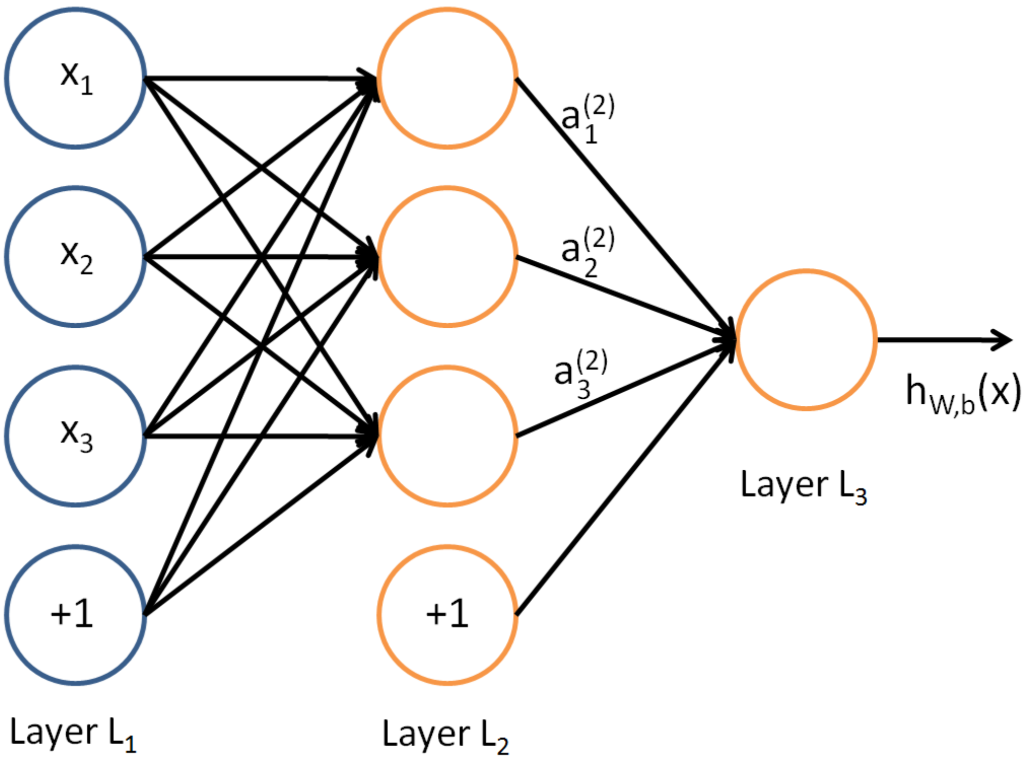
Comparison with Perceptron

- Perceptron has a binary output based on threshold
- Neuron has a continuous output (differentiable)
- Perceptron: more suitable for linearly separable data



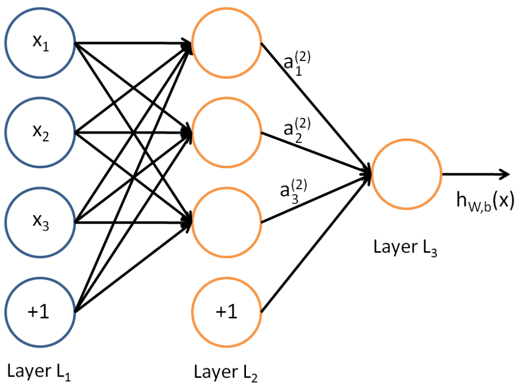
Neural Network

A collection of Neurons connected together



1. Input layer: leftmost layer
2. Output layer: rightmost layer
3. Hidden layer: hidden layer
4. Summary: this neural network has 3 input units, 3 hidden units and 1 output unit.

3-Layer Neural Network



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

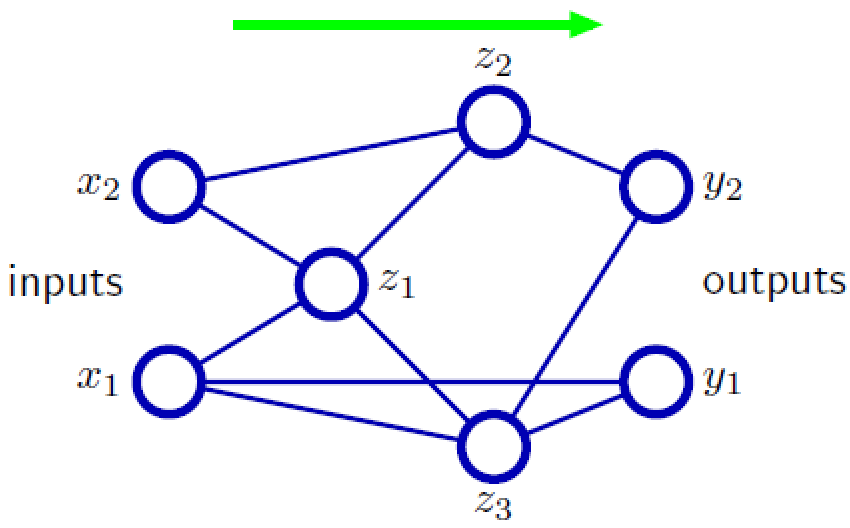
$$\begin{aligned}
a_1^{(2)} &:= f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \\
a_2^{(2)} &:= f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}) \\
a_3^{(2)} &:= f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}) \\
h_{\boldsymbol{\theta}}(\boldsymbol{x})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

- 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

- The inputs are fed forward to generate the output
- Therefore we call these neural networks Feedforward (Neural) Networks (FFN)
- No closed directed cycles
- Ensure that the outputs are deterministic functions of the inputs



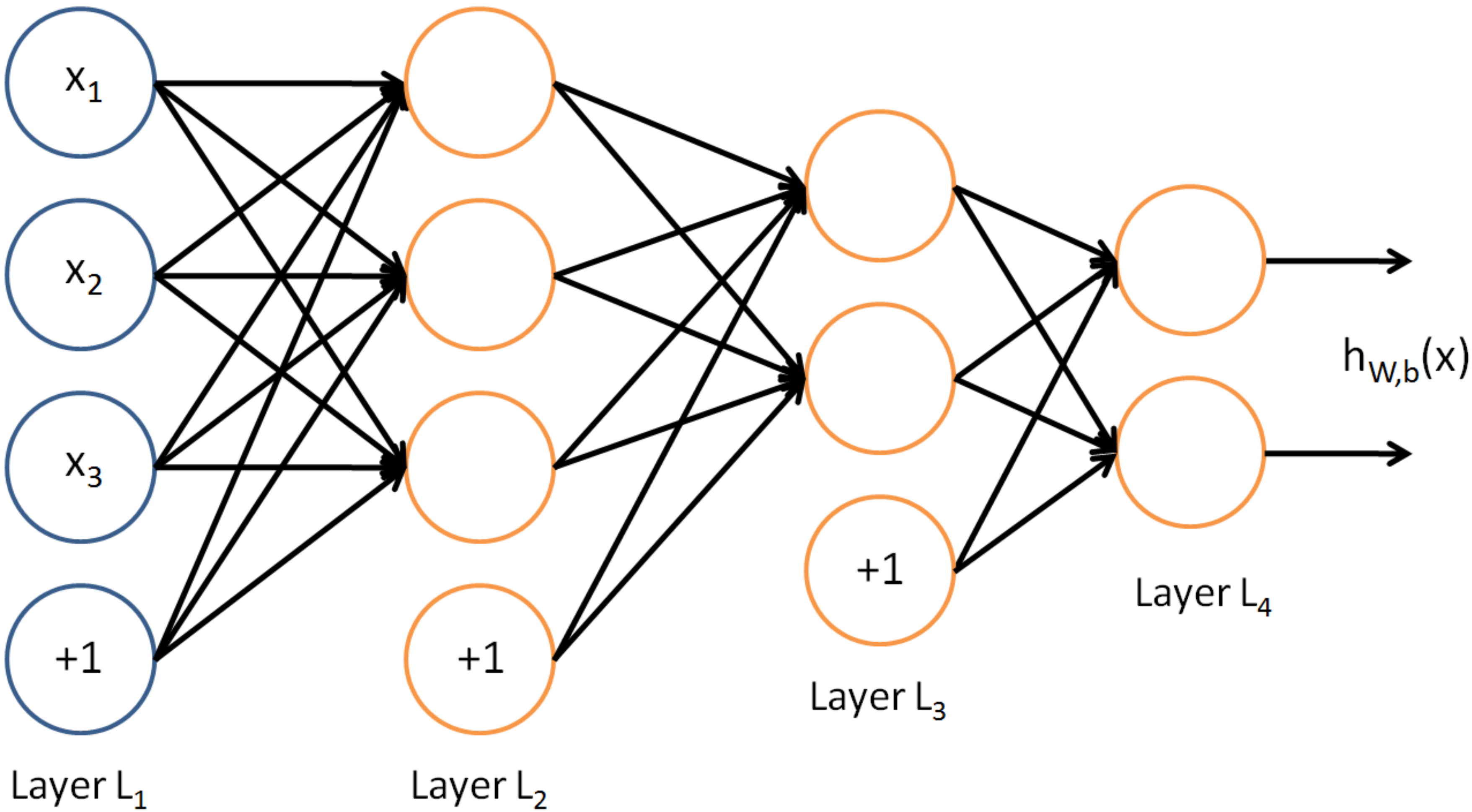
Relationship with Linear regression and classification

- Linear regression and classification:

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right)$$

  - For classification: f is a nonlinear activation function
  - For regression: the identity
- Remember how to introduce the nonlinearities?
- Each basis function in Neural networks is itself a nonlinear function of a linear combination of the inputs from the previous layer, do it recursively…..
- What will happen if the activation function is linear? Linear
- A general class of parametric nonlinear functions (i.e., in terms of the input variables) from a vector of input variables to a vector of output variables

Neural Networks with Multiple Outputs



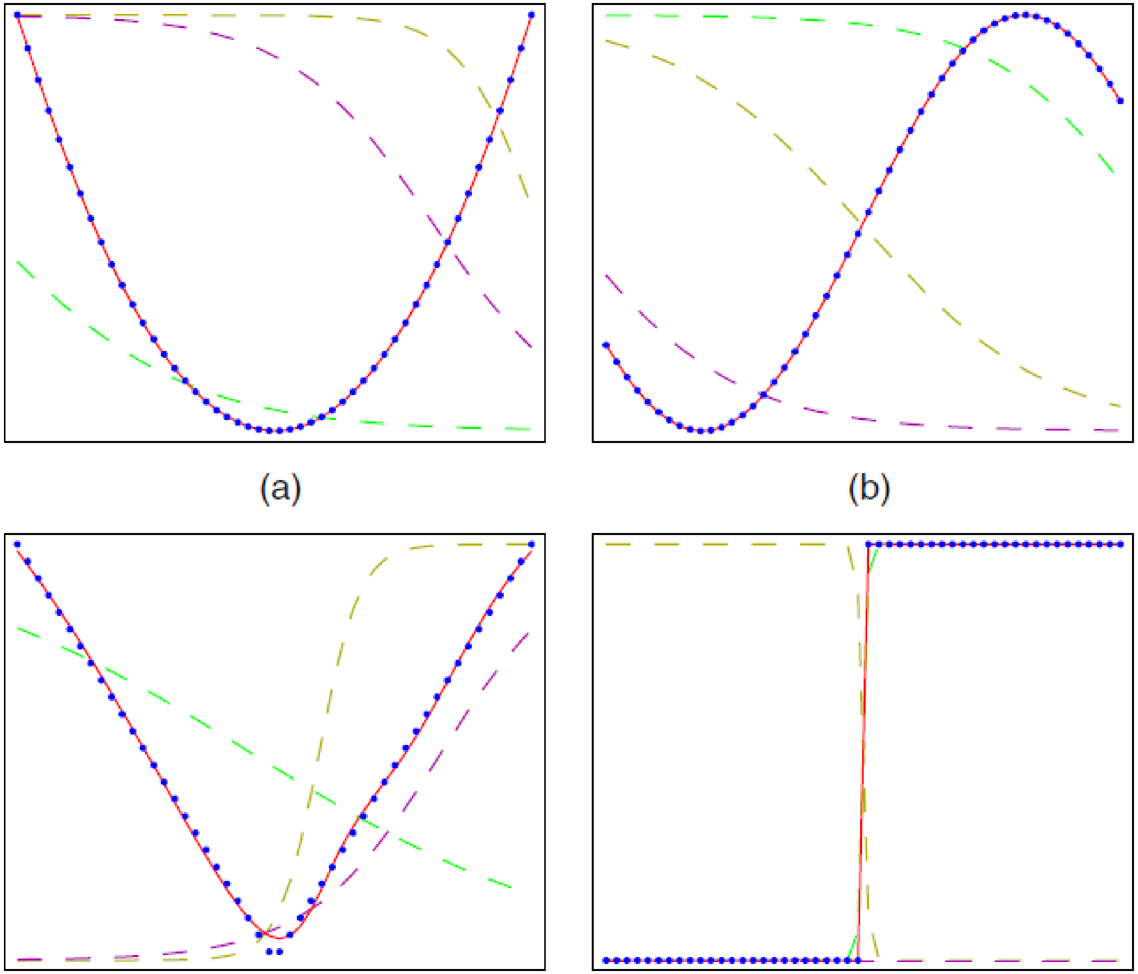
Neural Networks with other possibilities

- Skip layer
- Sparse (not all possible connections within a layer being present)

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

The power of neural networks


(a) (b) (c) (d)

1. Four functions:
   - a: $f(x) = x^2$
   - b: $f(x) = \sin(x)$
   - c: $f(x) = |x|$
   - d: $f(x) = H(x)$
2. 50 data points are sampled uniformly in x over the interval (-1,1)
3. Use these data points to train a 3-layered neural network having 3 hidden units with 'tanh' activation functions
4. Three dashed curves: outputs of the 3 hidden units

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 overfit 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