Lecture 12. Neural Networks and Backpropagation

COMP90051 Statistical Machine Learning

Semester 2, 2015

Lecturer: Andrey Kan

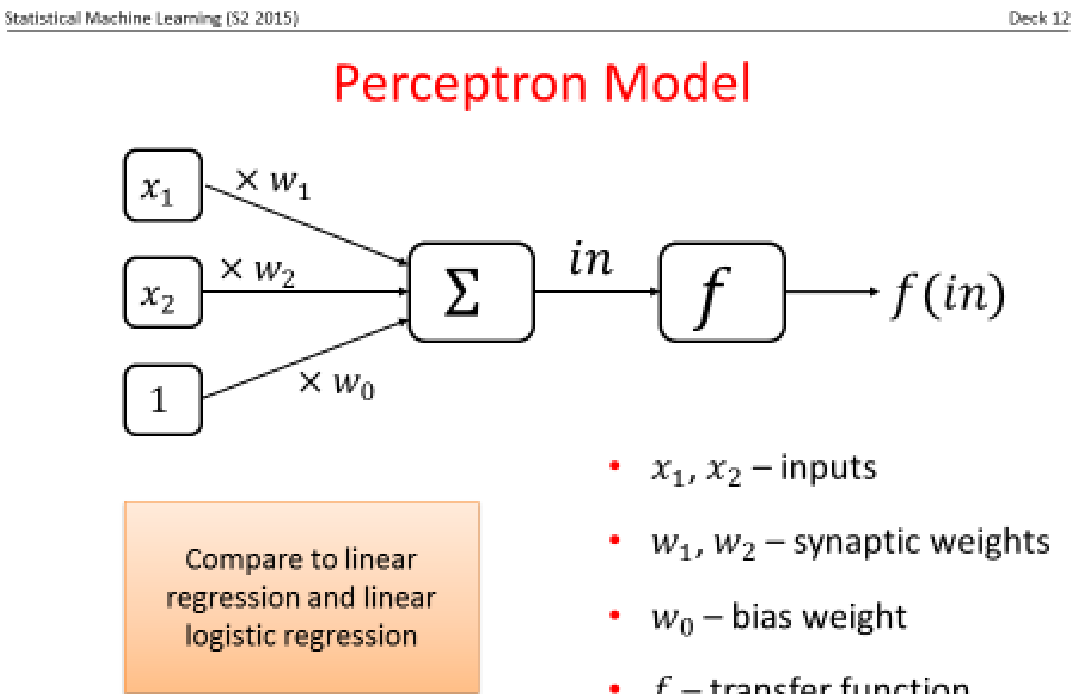
Content is based on slides provided by Jeffrey Chan and Ben Rubinstein

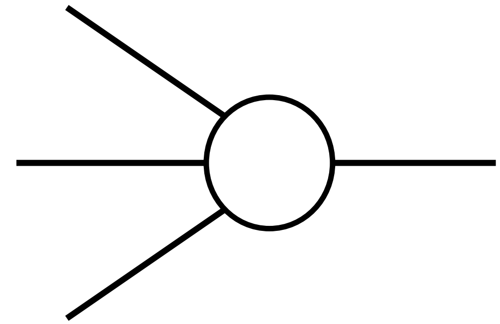
THE UNIVERSITY OF MELBOURNE

Multilayer Feed-Forward Neural Networks

The ANN approach to non-linearity.

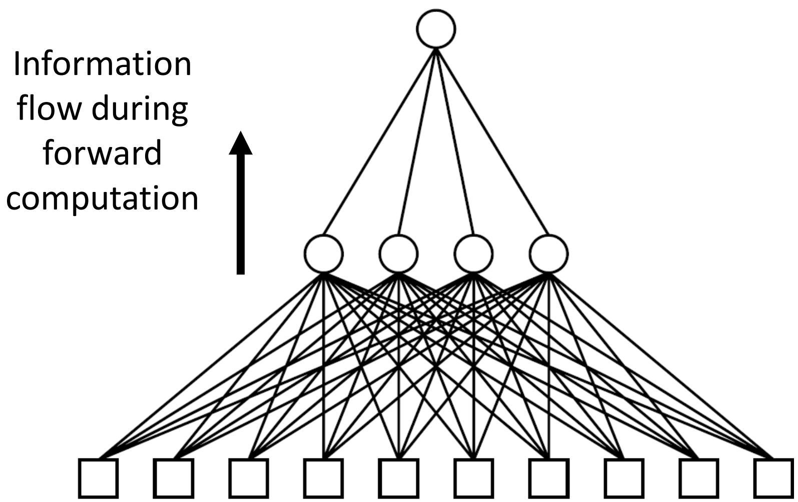
Simplified Graphical Representation

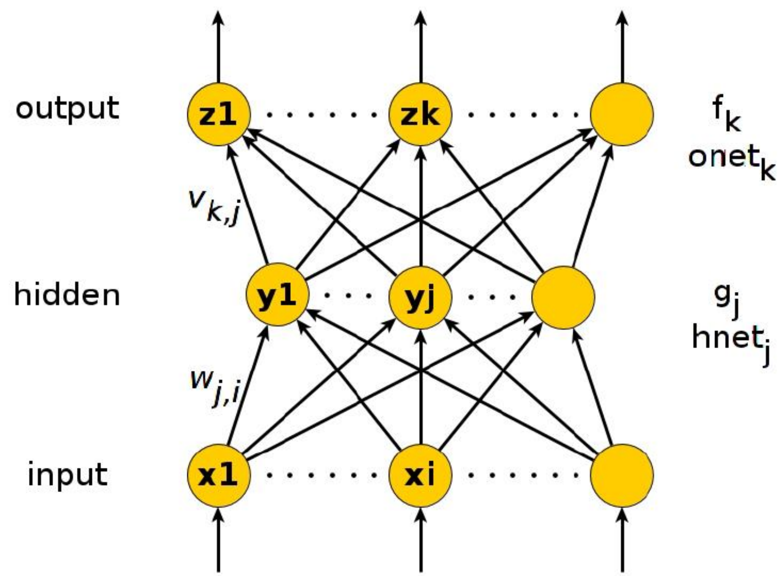




Network of Perceptrons

- Layers are often (but not always) fully connected; the numbers of hidden units typically chosen by hand
  - Note multidimensional output

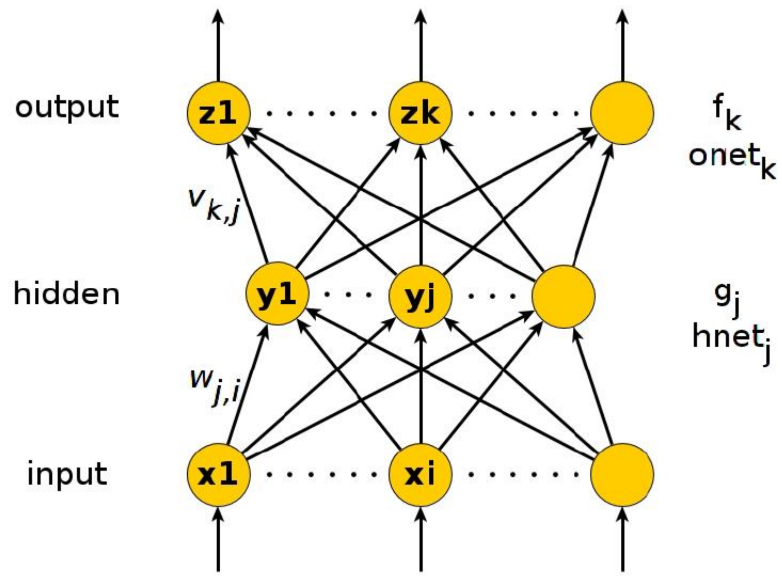




Feed Forward Neural Network

- How is the output z_k related to the inputs x's?

$$
\begin{aligned}
z_k &= f_k(s_k) \\
&= f_k\left(\sum_j v_{kj} y_j\right) \\
&= f_k\left(\sum_j v_{kj} g_i(u_j)\right) \\
&= f_k\left(\sum_j v_{kj} g_i\left(\sum_i w_{ji} x_i\right)\right)
\end{aligned}
$$



How to Train ~~Your Dragon~~the Network?

- Training example is a set of inputs and the desired outputs $a = [x_1, \dots, x_n, t_1, \dots, t_c]^T$
  * note that outputs denoted as t_i not y_i
- Aim is to learn (optimise) the weights v_{kj} and w_{ji} to minimise the difference between z_k (predicted) and t_k (training) for each of our training examples
- Define error function (discrepancy) <u>for one example</u> as

$$D = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2$$

Stochastic Gradient Descent

1. Initialisation: choose starting guess $\boldsymbol{\theta}^{(0)}, i = 0$
   * Here $\boldsymbol{\theta}$ is a set of all weights v_{kj} and w_{ji}
2. Randomly choose one training example $(\boldsymbol{x}, \boldsymbol{t})$
3. Compute discrepancy $D = 0.5 \cdot \sum_{k=1}^{c} (t_k - z_k)^2$
   * Computing $z_k(\boldsymbol{\theta}^{(i)})$ is called a forward propagation
4. Termination: decide whether to stop
5. Update: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \nabla D(\boldsymbol{\theta}^{(i)})$
6. Go to Step 2

Need to compute partial derivatives $\frac{\partial z_k}{\partial v_{kj}}$ and $\frac{\partial z_k}{\partial w_{kj}}$

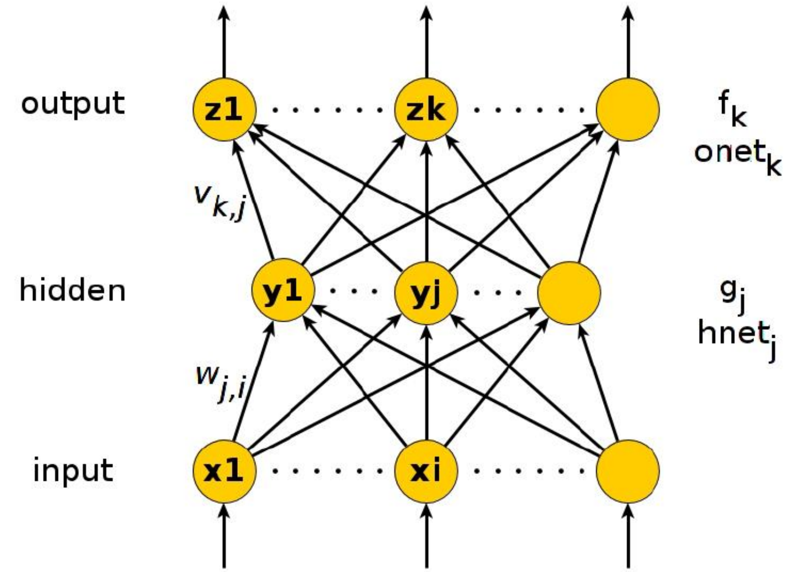
Backpropagation: Key Idea

- The structure of neural network allows efficient computation of partial derivatives $\frac{\partial z_k}{\partial v_{ki}}$ and $\frac{\partial z_k}{\partial w_{ji}}$

- Recall that $z_k = f_k(s_k)$
  * Where $s_k = \sum_j v_{kj} y_j$

- We can apply chain rule for derivatives

- We have $\frac{\partial z_k}{\partial v_{ki}} = \frac{\partial f_k}{\partial s_k} \frac{\partial s_k}{\partial v_{ki}}$



Backpropagation: Start from Outputs

- We have $\frac{\partial z_k}{\partial v_{ki}} = \frac{\partial f_k}{\partial s_k}\frac{\partial s_k}{\partial v_{ki}}$
- Consider $\frac{\partial f_k}{\partial s_k}$ first

Statistical Machine Learning (S2 2015) Deck 12

Transfer functions f

Step function $f(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & if\ s < 0 \end{cases}$

Sign function $f(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$

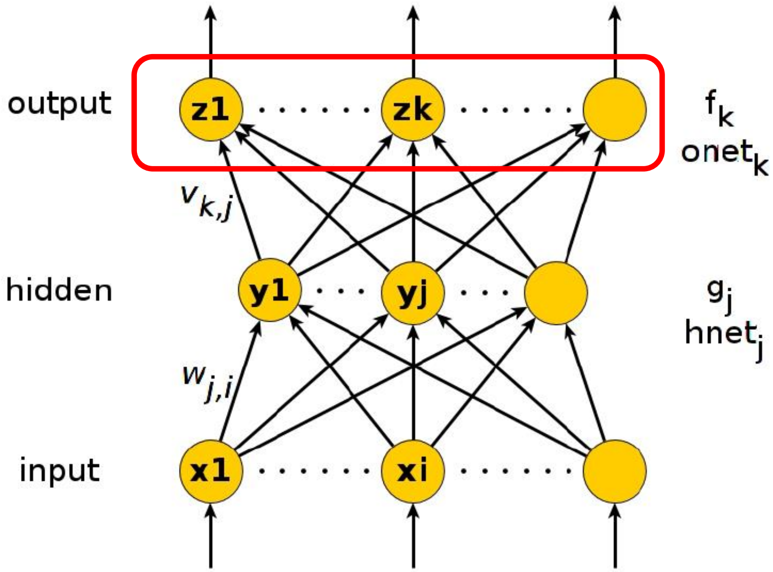
Logistic function $f(s) = \frac{1}{1+e^{-s}}$

9

- It is convenient to choose $f(s) = \frac{1}{1+e^{-s}}$
- Then $\frac{\partial f_k}{\partial s_k} = f_k(s_k)(1 - f_k(s_k))$
- Recall $f_k(s_k) = z_k$ is already computed during the forward propagation

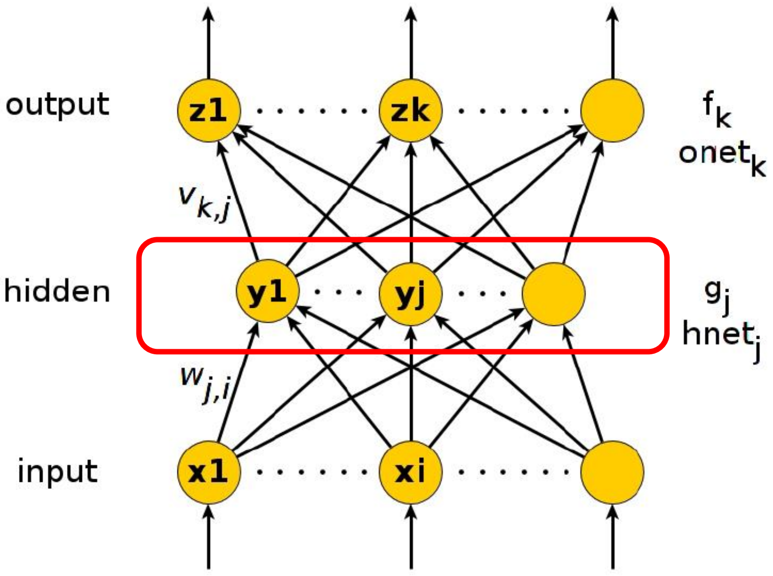
Backpropagation: Start from Outputs

- We have $\frac{\partial z_k}{\partial v_{ki}} = \frac{\partial f_k}{\partial s_k} \frac{\partial s_k}{\partial v_{ki}}$
- $\frac{\partial f_k}{\partial s_k} = z_k\,(1 - z_k)$



Proceed to the Next Layer

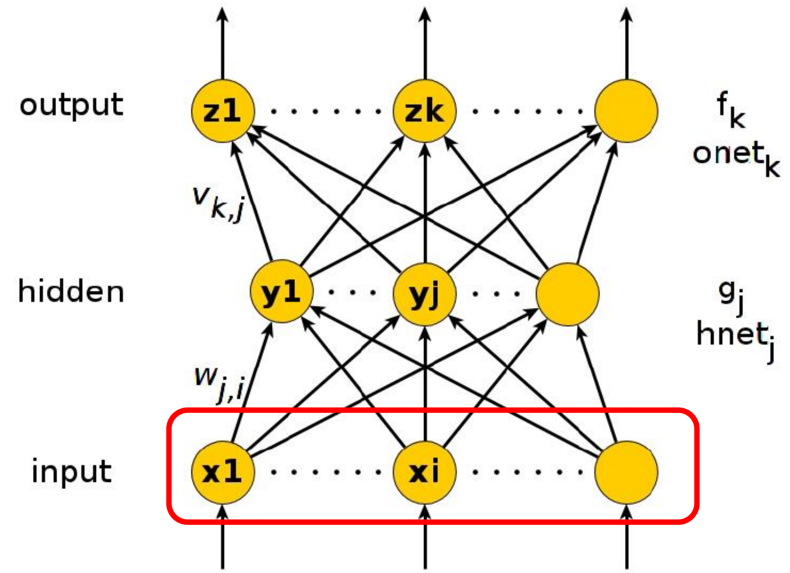
- We have $\frac{\partial z_k}{\partial v_{ki}} = \frac{\partial f_k}{\partial s_k} \frac{\partial s_k}{\partial v_{ki}}$
- Next consider $\frac{\partial s_k}{\partial v_{ki}}$
- Here $s_k = \sum_j v_{kj} y_j$
- Therefore $\frac{\partial s_k}{\partial v_{ki}} = y_j$



- $\frac{\partial z_k}{\partial v_{kj}} = [z_k(1 - z_k)][y_j]$

Reaching the Final Layer

- Similarly $\frac{\partial z_k}{\partial w_{kj}} = \frac{\partial f_k}{\partial s_k}\frac{\partial s_k}{\partial y_j}\frac{\partial y_j}{\partial u_j}\frac{\partial u_j}{\partial x_i}$
- Here $z_k = f(s_k)$
- and $s_k = \sum_j v_{kj} y_j$
- and $y_j = g(u_j)$
- and $u_j = \sum_i w_{ji} x_i$



- $\frac{\partial z_k}{\partial w_{kj}} = [z_k(1 - z_k)][v_{kj}][g'(u_j)][x_i]$

Stochastic Gradient Descent

1. Initialisation: choose starting guess $\boldsymbol{\theta}^{(0)}$, $i = 0$
   * Here $\boldsymbol{\theta}$ is a set of all weights v_{kj} and w_{ji}
2. Randomly choose one training example $(\boldsymbol{x}, \boldsymbol{t})$
3. Compute discrepancy $D = 0.5 \cdot \sum_{k=1}^{c}(t_k - z_k)^2$
   * Computing $z_k(\boldsymbol{\theta}^{(i)})$ is called a forward propagation
4. Termination: decide whether to stop
5. Update: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \nabla D(\boldsymbol{\theta}^{(i)})$
6. Go to Step 2

Deriving the Update Rule

- Discrepancy $D = 0.5 \cdot \sum_{k=1}^{c}(t_k - z_k)^2$
- Partial derivatives $\frac{\partial D}{\partial v_{kj}} = \frac{\partial D}{\partial z_k}\boxed{\frac{\partial z_k}{\partial v_{kj}}}$ and $\frac{\partial D}{\partial w_{kj}} = \frac{\partial D}{\partial z_k}\boxed{\frac{\partial z_k}{\partial w_{kj}}}$
- Recall that we have already derived these
- Define $\delta_k \stackrel{\text{def}}{=} -(t_k - z_k)z_k(1 - z_k)$
- $\frac{\partial D}{\partial v_{kj}} = \delta_k y_j$
- $\frac{\partial D}{\partial w_{kj}} = g'(u_j)x_i \sum_{k=1}^{c} \delta_k \, v_{kj}$

Exercise: prove this

Deriving the Update Rule

- <u>The Update Rule</u>
- $v_{kj} \leftarrow v_{kj} - \eta \delta_k y_j$
- $w_{kj} \leftarrow w_{kj} - \eta g'(u_j) x_i \sum_{k=1}^{c} \delta_k v_{kj}$

- Define $\delta_k \stackrel{\text{def}}{=} -(t_k - z_k) z_k (1 - z_k)$
- $\frac{\partial D}{\partial v_{kj}} = \delta_k y_j$
- $\frac{\partial D}{\partial w_{kj}} = g'(u_j) x_i \sum_{k=1}^{c} \delta_k v_{kj}$

Summary

- Neural network is a non-linear model
  - Cumbersome equation
  - Fancy graphical representation
- Structure of the model allows efficient computation of partial derivatives
  - Training is based on stochastic gradient descent