

## Multi-layer neural networks

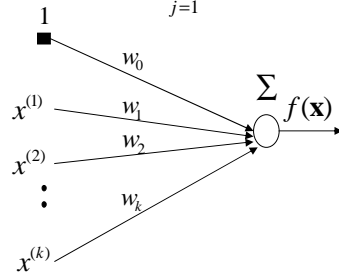
Vasileios Megalooikonomou

(some material adopted from notes by M. Hauskrecht)

### Summary of linear units

#### Linear regression

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^k w_j x^{(j)}$$



#### On-line gradient update:

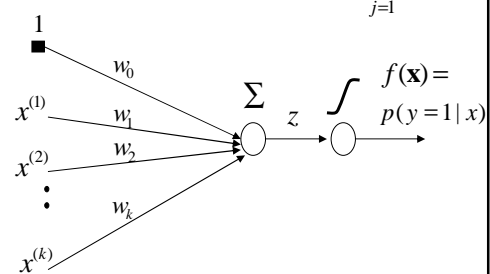
$$w_0 \leftarrow w_0 + \alpha(y - f(\mathbf{x}))$$

$$\vdots$$

$$w_j \leftarrow w_j + \alpha(y - f(\mathbf{x}))x^{(j)}$$

#### Logistic regression

$$f(\mathbf{x}) = p(y=1 | \mathbf{x}, \mathbf{w}) = g(w_0 + \sum_{j=1}^k w_j x^{(j)})$$



#### On-line gradient update:

$$w_0 \leftarrow w_0 + \alpha(y - f(\mathbf{x}))$$

$$\vdots$$

$$w_j \leftarrow w_j + \alpha(y - f(\mathbf{x}))x^{(j)}$$

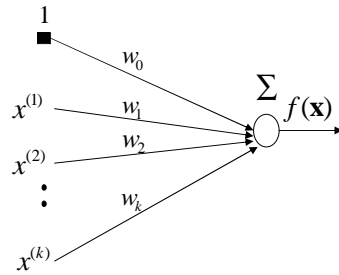
The same



## Limitations of basic linear units

### Linear regression

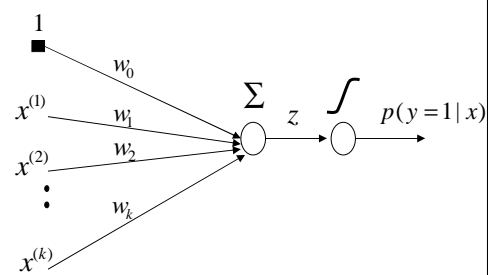
$$f(\mathbf{x}) = w_0 + \sum_{j=1}^k w_j x^{(j)}$$



**Function linear in inputs !!**

### Logistic regression

$$f(\mathbf{x}) = p(y=1 | \mathbf{x}, \mathbf{w}) = g(w_0 + \sum_{j=1}^k w_j x^{(j)})$$

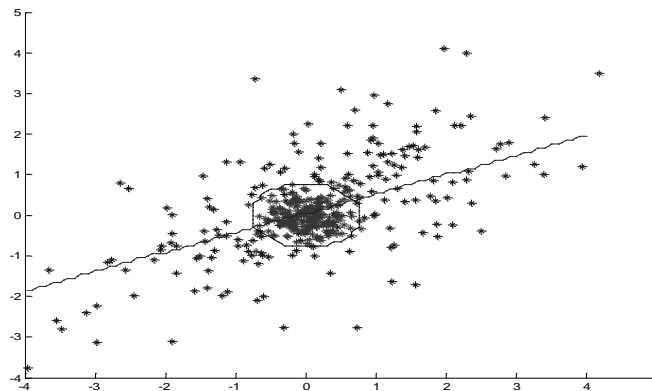


**Linear decision boundary !!**

CIS603 - AI

## Limitations of linear units

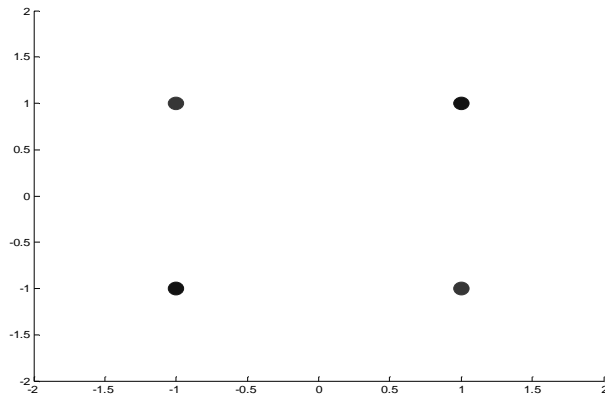
- Example in which the logistic regression model with linear decision boundary fails



CIS603 - AI

## Limitations of linear units.

- Logistic regression does not work for **parity functions**
  - no linear decision boundary



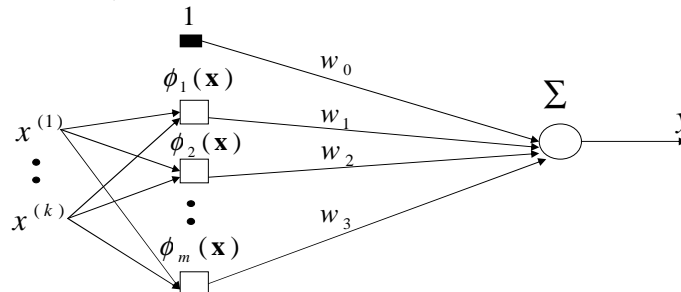
CIS603 - AI

## Extensions of simple linear units

Replace inputs to linear units with **feature (basis) functions** to model **nonlinearities**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^m w_j \phi_j(\mathbf{x})$$

$\phi_j(\mathbf{x})$  - an arbitrary function of  $\mathbf{x}$

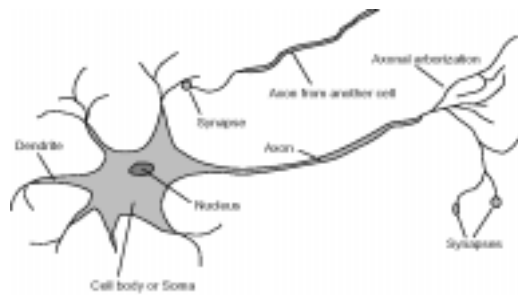


The same trick can be done for the logistic regression

CIS603 - AI

## Multi-layered neural networks

- Alternative way to introduce nonlinearities to regression/classification models
- **Idea:** Cascade several simple neural models (based on logistic regression). Much like neuron connections.



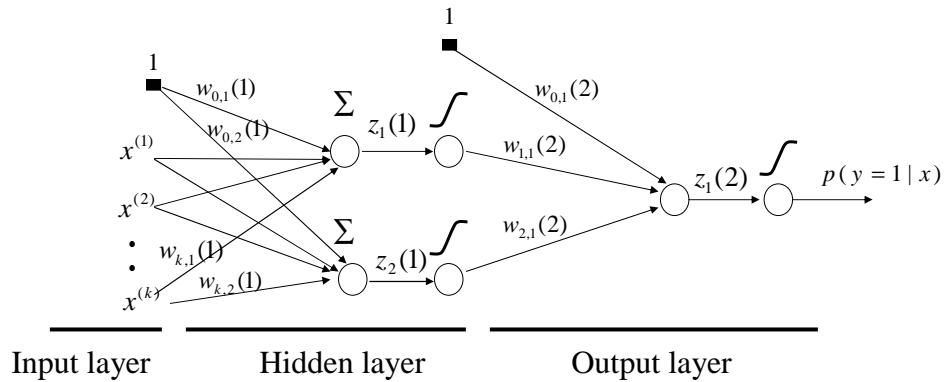
CIS603 - AI

## Multilayer neural network

Also called **multilayer perceptron (MLP)**

Cascades multiple logistic regression units

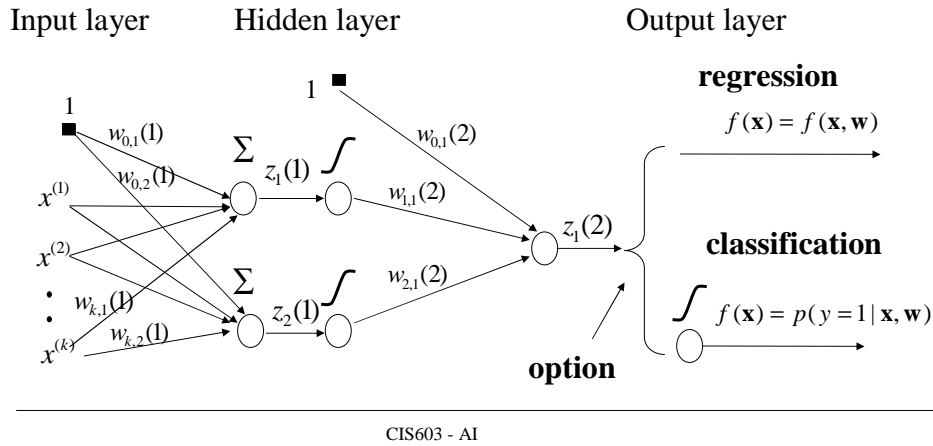
**Example:** classifier with non-linear decision boundaries



CIS603 - AI

## Multilayer neural network

- Models **non-linearities through logistic regression units**
- Can be applied to both **regression and binary classification**



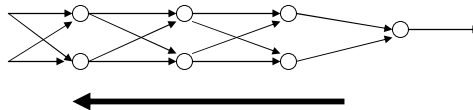
## Learning with MLP

- How to learn the parameters of the neural network?
- **Online gradient descent algorithm**
  - Weight updates based on

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J_{\text{online}}(d_i, \mathbf{w})$$

$J_{\text{online}}(d_i, \mathbf{w})$  - online component of the error function

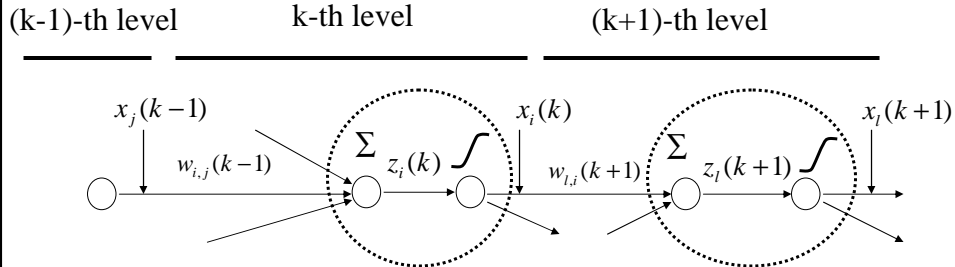
- We need to compute gradients for weights in all units
- Can be computed in one backward sweep through the net !!!



- The process is called **back-propagation**

CIS603 - AI

## Backpropagation



$x_i(k)$  - output of the unit  $i$  on level  $k$

$z_i(k)$  - input to the sigmoid function on level  $k$

$w_{i,j}(k)$  - weight between units  $j$  and  $i$  on levels  $(k-1)$  and  $k$

$$z_i(k) = w_{i,0}(k) + \sum_j w_{i,j}(k)x_j(k-1)$$

$$x_i(k) = g(z_i(k))$$

CIS603 - AI

## Backpropagation

Update weight  $w_{i,j}(k)$  using a data point  $d$

$$w_{i,j}(k) \leftarrow w_{i,j}(k) - \alpha \frac{\partial}{\partial w_{i,j}(k)} J_{online}(d, \mathbf{w})$$

$$\text{Let } \delta_i(k) = \frac{\partial}{\partial z_i(k)} J_{online}(d, \mathbf{w})$$

$$\text{Then: } \frac{\partial}{\partial w_{i,j}(k)} J_{online}(d, \mathbf{w}) = \frac{\partial J_{online}(d, \mathbf{w})}{\partial z_i(k)} \frac{\partial z_i(k)}{\partial w_{i,j}(k)} = \delta_i(k)x_j(k-1)$$

S.t.  $\delta_i(k)$  is computed from the next layer  $\delta_i(k+1)$  backwards

$$\delta_i(k) = \sum_l \delta_l(k+1)w_{l,i}(k+1)$$

**Last unit** (is the same as for regular linear units):

$$\delta_i(K) = -(y - f(\mathbf{x}, \mathbf{w}))$$

Exactly the same for classification with the log-likelihood measure and linear regression with least-squares !!!

CIS603 - AI

## Learning with MLP

- **Online gradient descent algorithm**

– Weight update:

$$w_{i,j}(k) \leftarrow w_{i,j}(k) - \alpha \frac{\partial}{\partial w_{i,j}(k)} J_{\text{online}}(d_i, \mathbf{w})$$

$$\frac{\partial}{\partial w_{i,j}(k)} J_{\text{online}}(d, w) = \frac{\partial J_{\text{online}}}{\partial z_i(k)} \frac{\partial z_i(k)}{\partial w_{i,j}(k)} = \delta_i(k) x_j(k-1)$$

$$w_{i,j}(k) \leftarrow w_{i,j}(k) - \alpha \delta_i(k) x_j(k-1)$$

$x_j(k-1)$  - j-th output of the (k-1) layer

$\delta_i(k)$  - derivative computed via backpropagation

CIS603 - AI

## Online gradient descent algorithm for MLP

**Online-gradient-descent** ( $D$ , number of iterations)

**Initialize** all weights  $w_{i,j}(k)$

**for**  $i=1:1$ : number of iterations

**do**     **select** a data point  $d = \langle x, y \rangle$  from  $D$

**set**  $\alpha = 1/i$

**compute** outputs  $x_j(k)$  for each unit

**compute** derivatives  $\delta_i(k)$  via backpropagation

**update** all weights (in parallel)

$$w_{i,j}(k) \leftarrow w_{i,j}(k) - \alpha \delta_i(k) x_j(k-1)$$

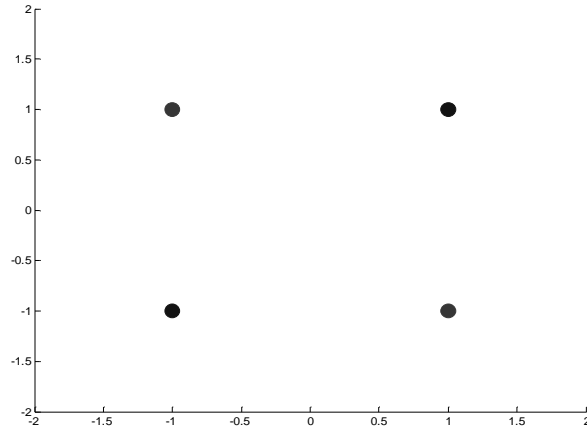
**end for**

**return** weights

CIS603 - AI

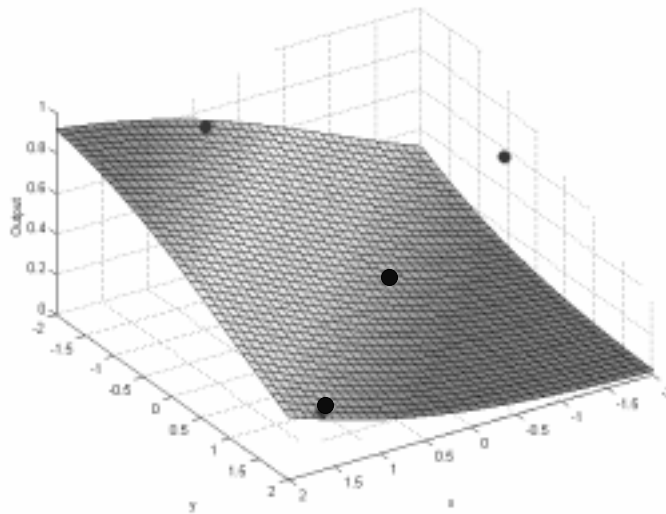
## Xor Example.

- No linear decision boundary



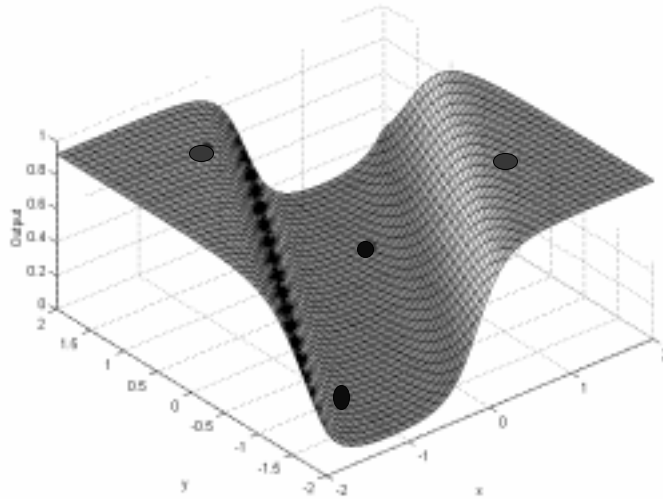
CIS603 - AI

## Xor example. Linear unit



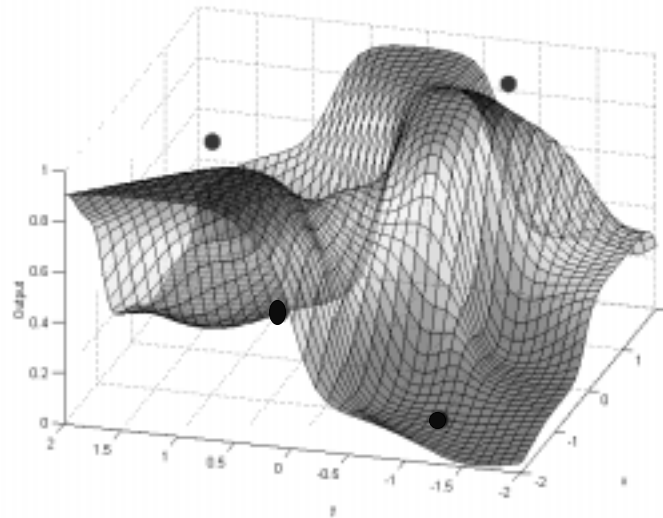
CIS603 - AI

**Xor example.**  
**Neural network with 2 hidden units**



CIS603 - AI

**Xor example.**  
**Neural network with 10 hidden units)**



CIS603 - AI