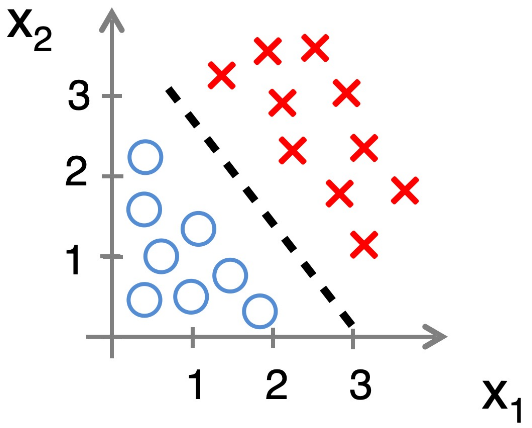
Lecture *4* – Multilayer Perceptron

- AI in Genetics
- *ZOO6927 / BOT6935 / ZOO4926*

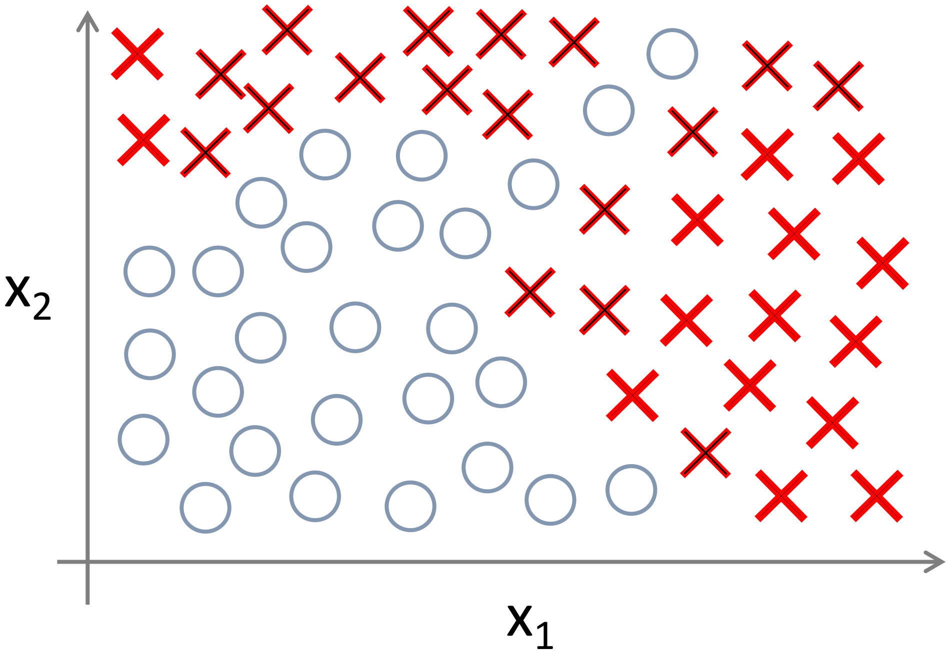
Book chapter: Murphy 13.1, 13.2

Logistic regression

$$p(y \mid x, \theta) = \text{Ber}\left(y \mid g(\theta^T x)\right)$$

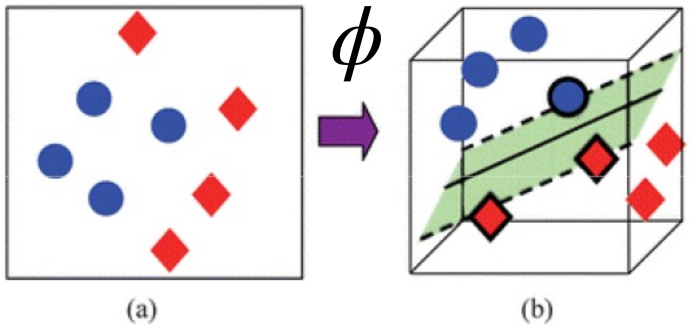


Non-linear Classification



$$
\begin{aligned}
x_1 &= \text{tumor size} \\
x_2 &= \text{age} \\
x_3 &= \text{sex} \\
x_4 &= \text{smoking} \\
&\ldots \\
x_{100} &
\end{aligned}
$$

Transforming inputs to higher dimensional space



Cover's Theorem: A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

Nonlinear transformation of the inputs

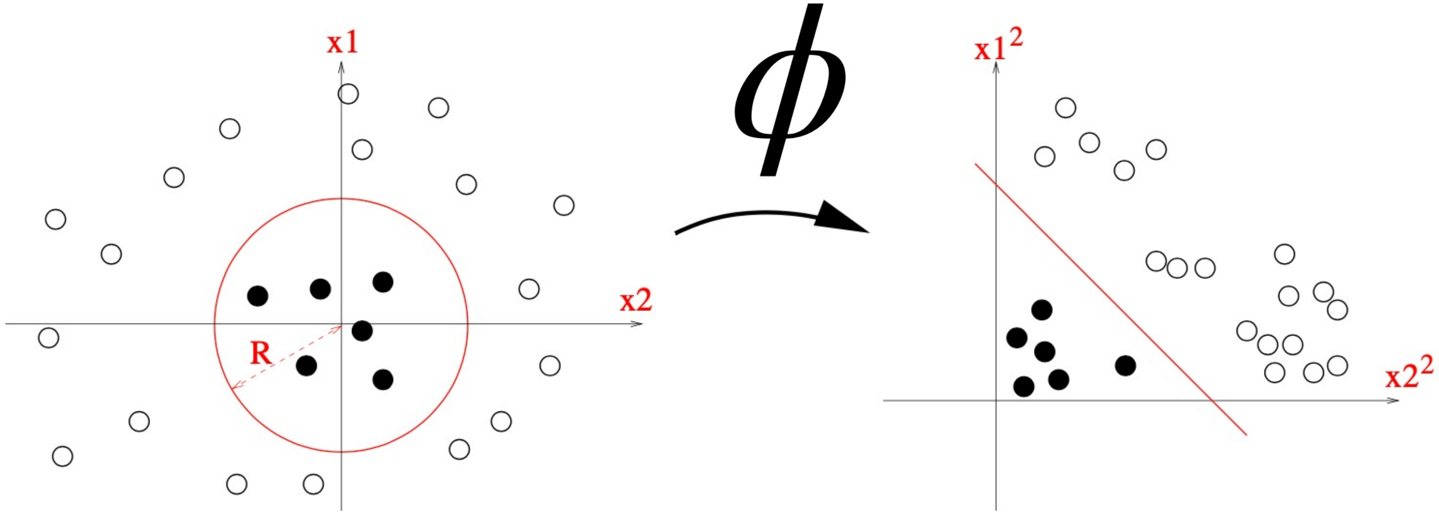
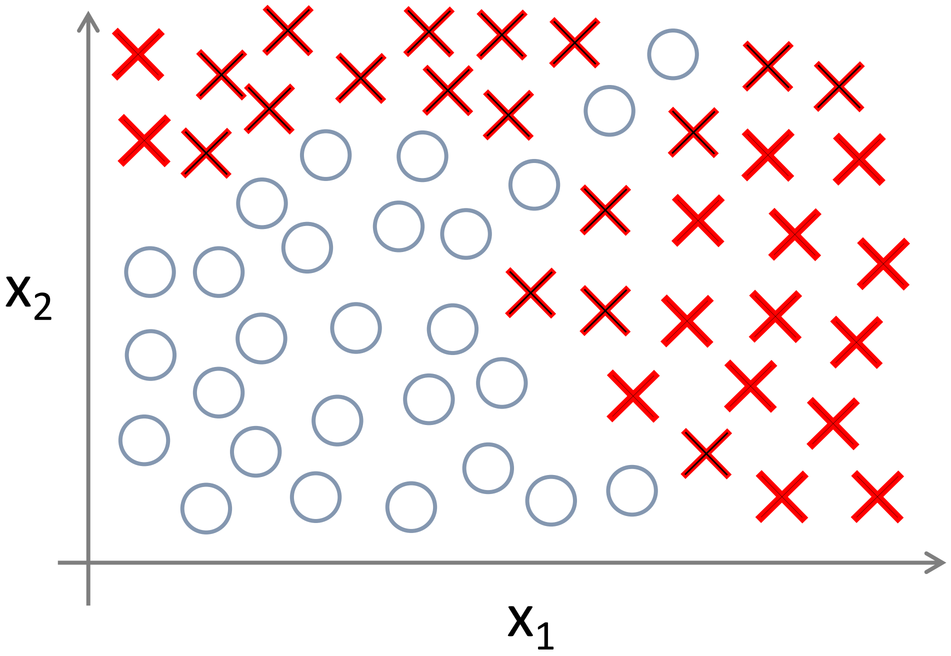


Figure 10.3: Illustration of how we can transform a quadratic decision boundary into a linear one by transforming the features from $\boldsymbol{x} = (x_1, x_2)$ *to* $\boldsymbol{\phi}(\boldsymbol{x}) = (x_1^2, x_2^2)$*. Used with kind permission of Jean-Philippe Vert.*

Non-linear Classification



x_1 = tumor size
x_2 = age
x_3 = sex
x_4 = smoking
$\cdots$
x_{100}

ϕ is the feature extractor

$$\phi(x) = \begin{bmatrix} x_1 & x_2 & x_1x_2 & x_1^2x_2 & x_1^3x_2 & x_1x_2^2 & \cdots \end{bmatrix}^T$$

$$g(\theta^T\phi(x)) =$$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ +\theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 \\ +\theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \ldots)$$

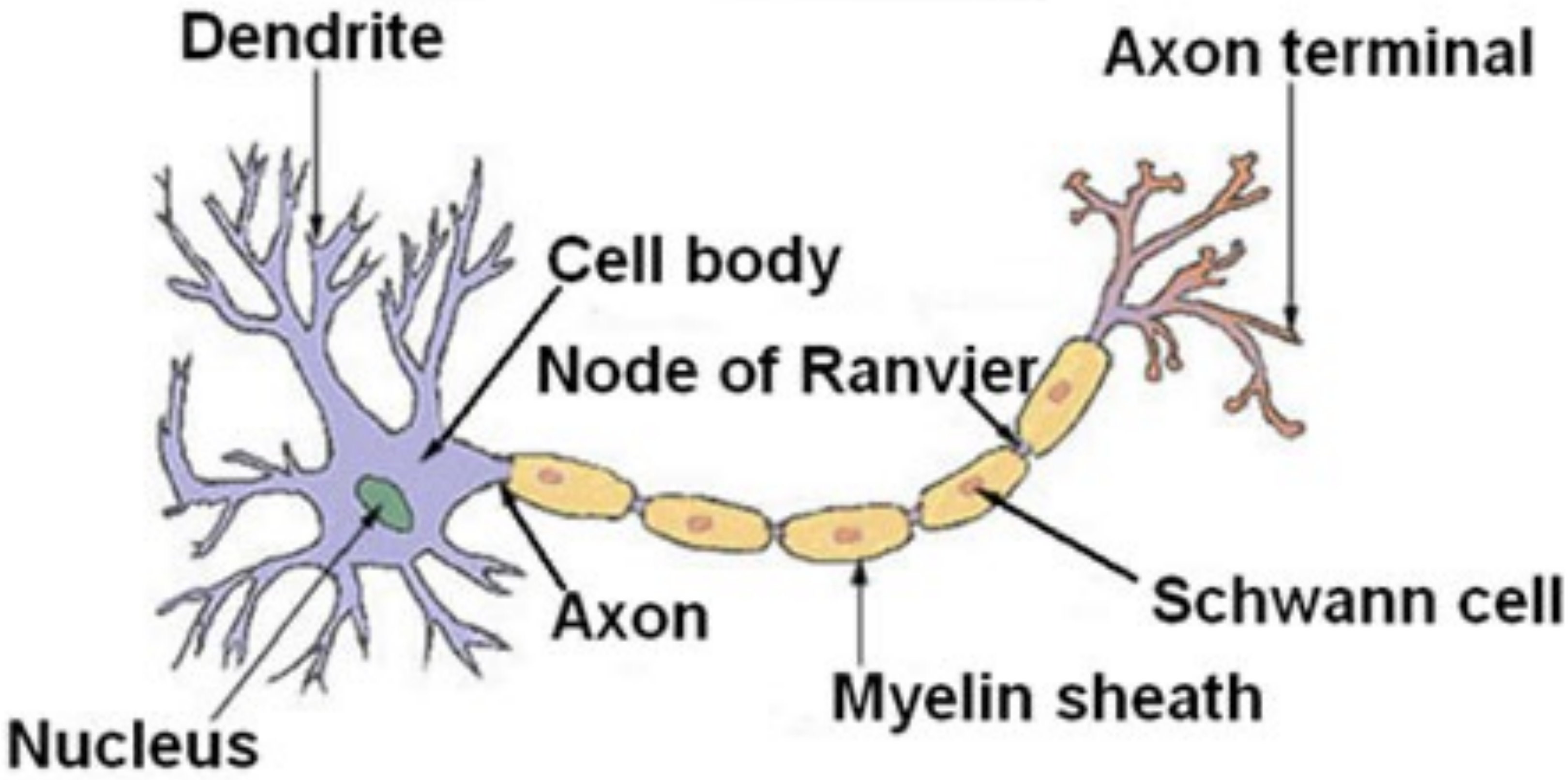
- **Having to specify the feature transformation by hand is very limiting**
  - Number of possible features is too many
  - Not all features are useful
- **Can we train the model to automatically select for useful features?**

A natural extension is to endow the feature extractor with its own parameters

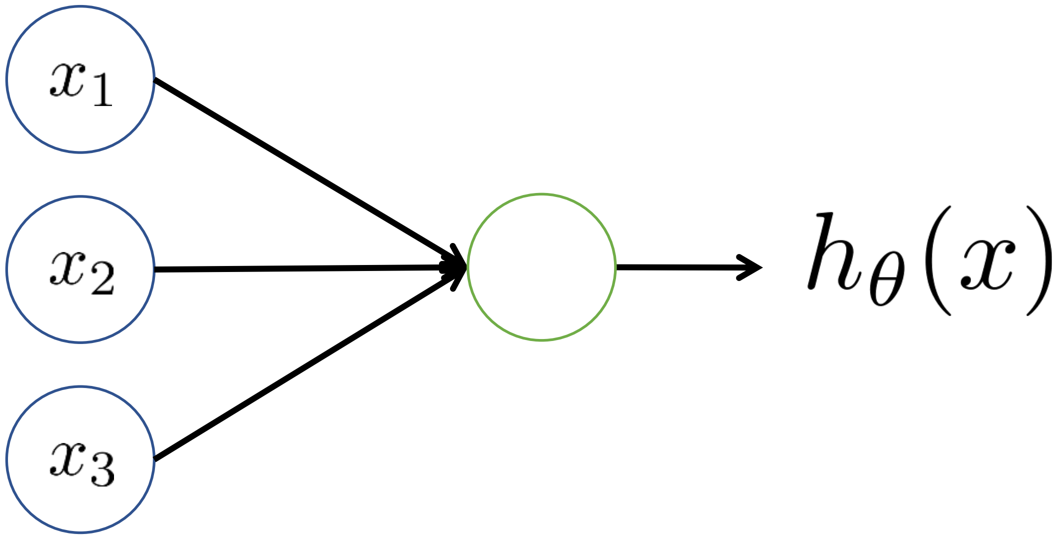
$$\phi_{\theta'}(x)$$

$$\phi_{\theta'}(x) = [? \quad ? \quad ? \quad ? \quad ? \quad ? \quad \cdots]^T$$

Draw inspiration from the brain



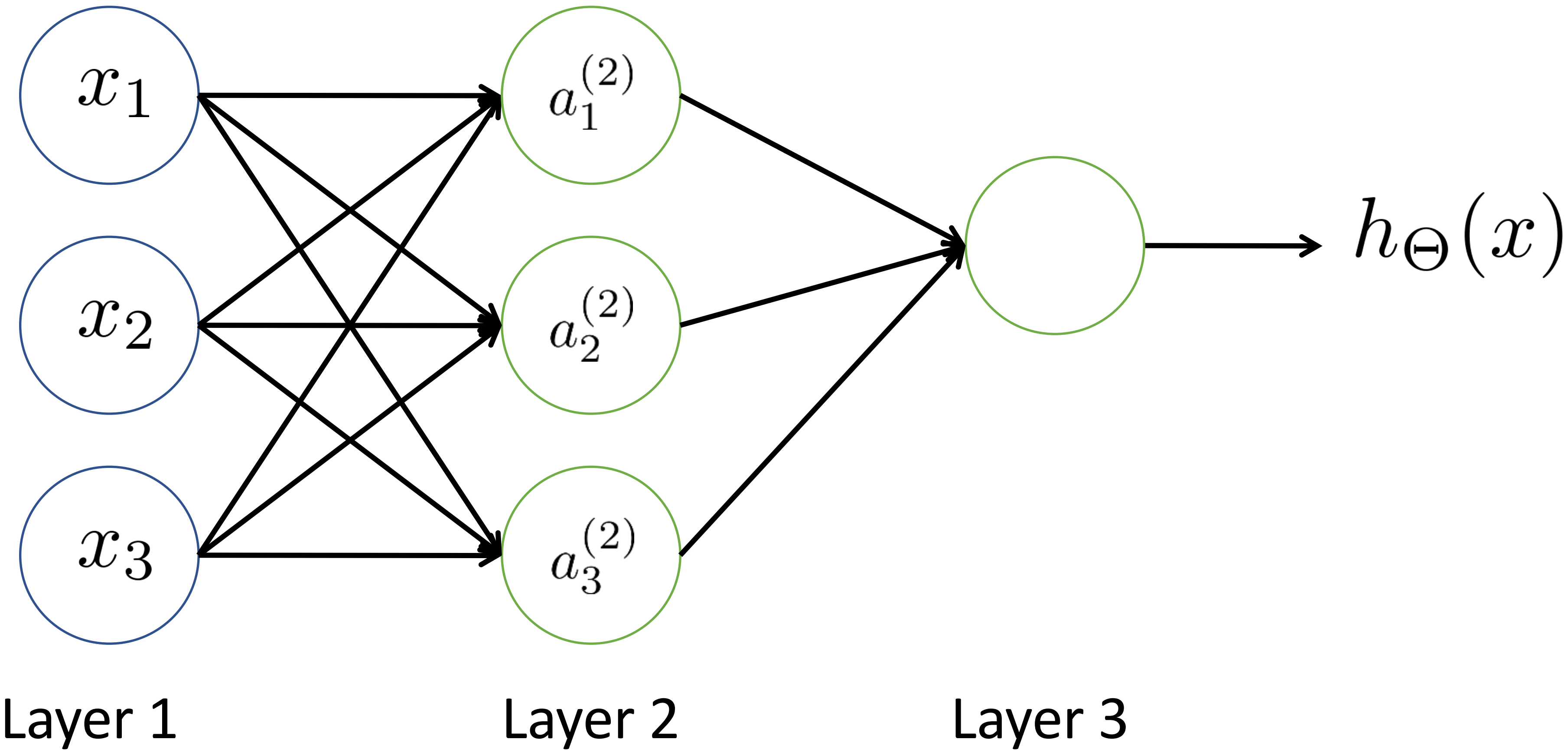
Neuron model: Logistic unit ('perceptron')



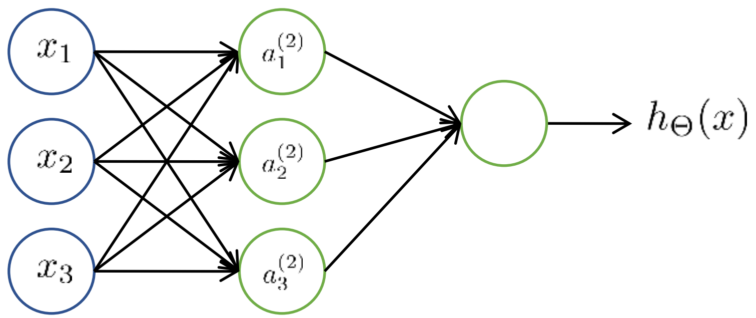
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

Sigmoid (logistic) activation function.

Neural Network



Neural Network: what goes on under the hood



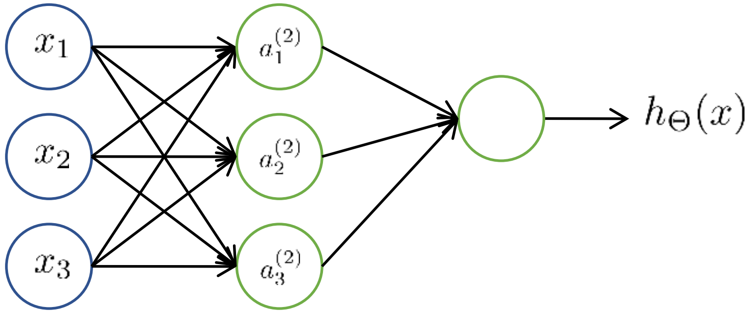
$a_i^{(j)} =$ "activation" of unit i in layer j

$\Theta^{(j)} =$ matrix of weights 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, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Forward propagation: Vectorized implementation



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

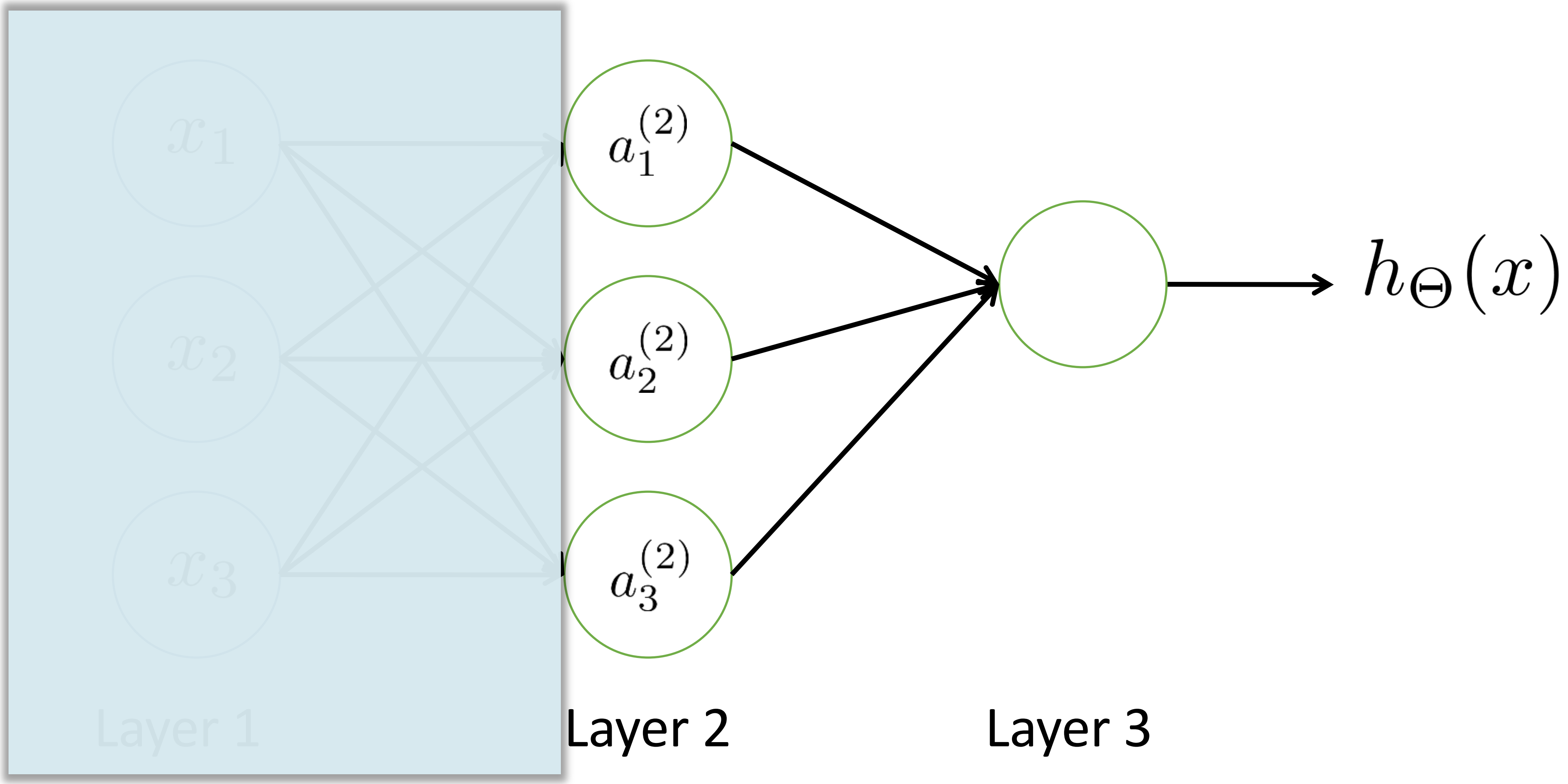
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)}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

$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

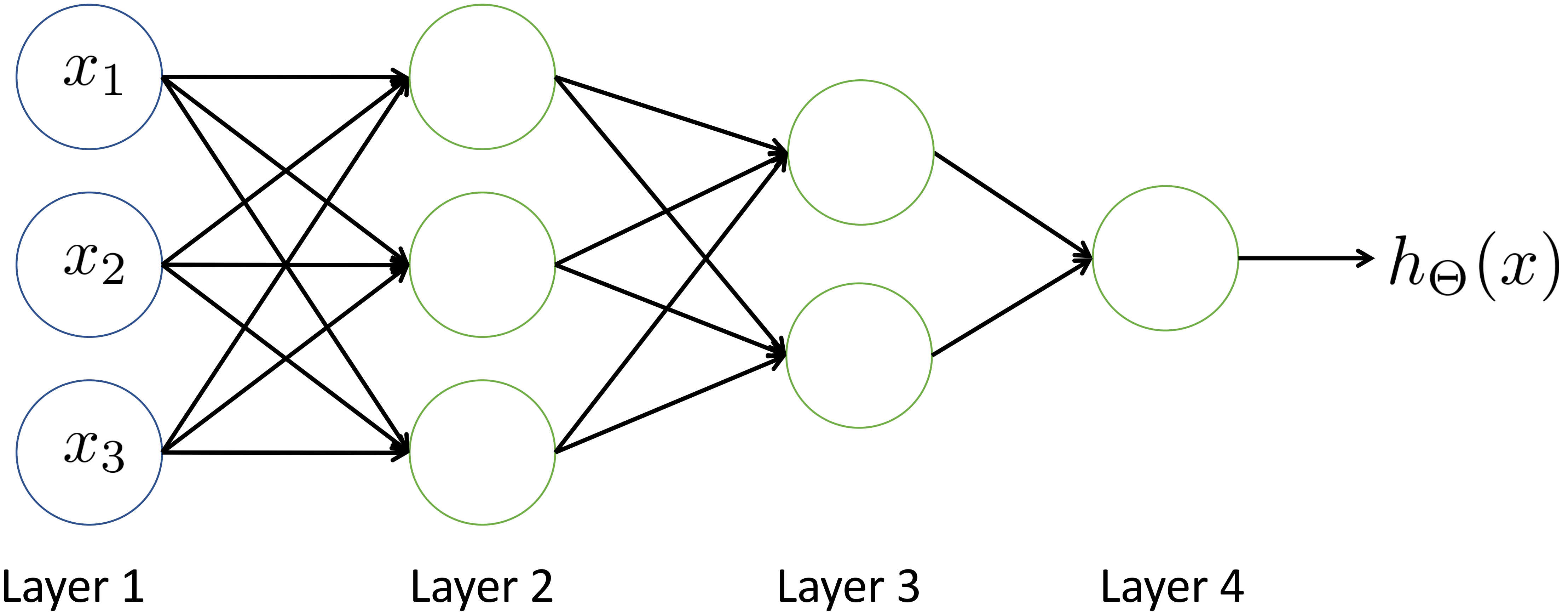
$$z^{(2)} = \Theta^{(1)} x$$
$$a^{(2)} = g(z^{(2)})$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Neural Network learning its own features



Deep neural network



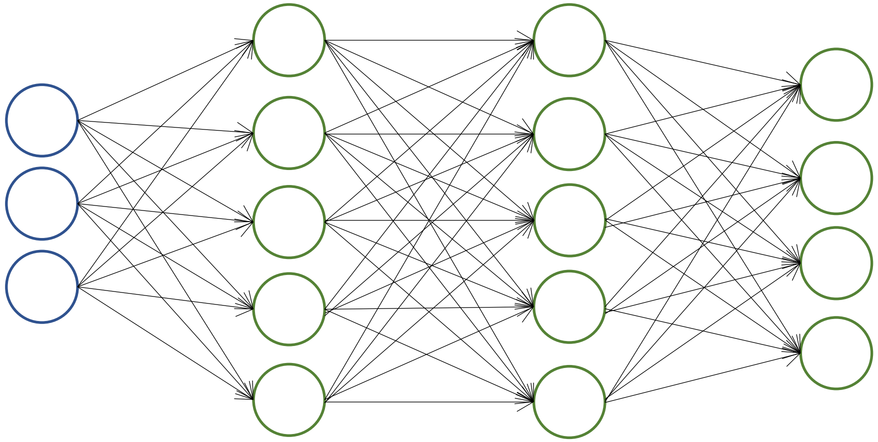
Multi-class classification

Pedestrian

Car

Motorcycle

Truck

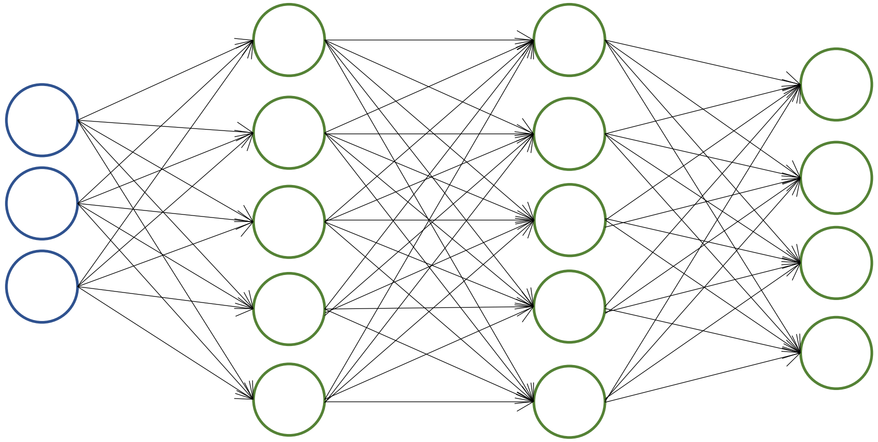


$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian   when car   when motorcycle

Multiple output units: One-vs-all.



$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, , $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

pedestrian car motorcycle truck