



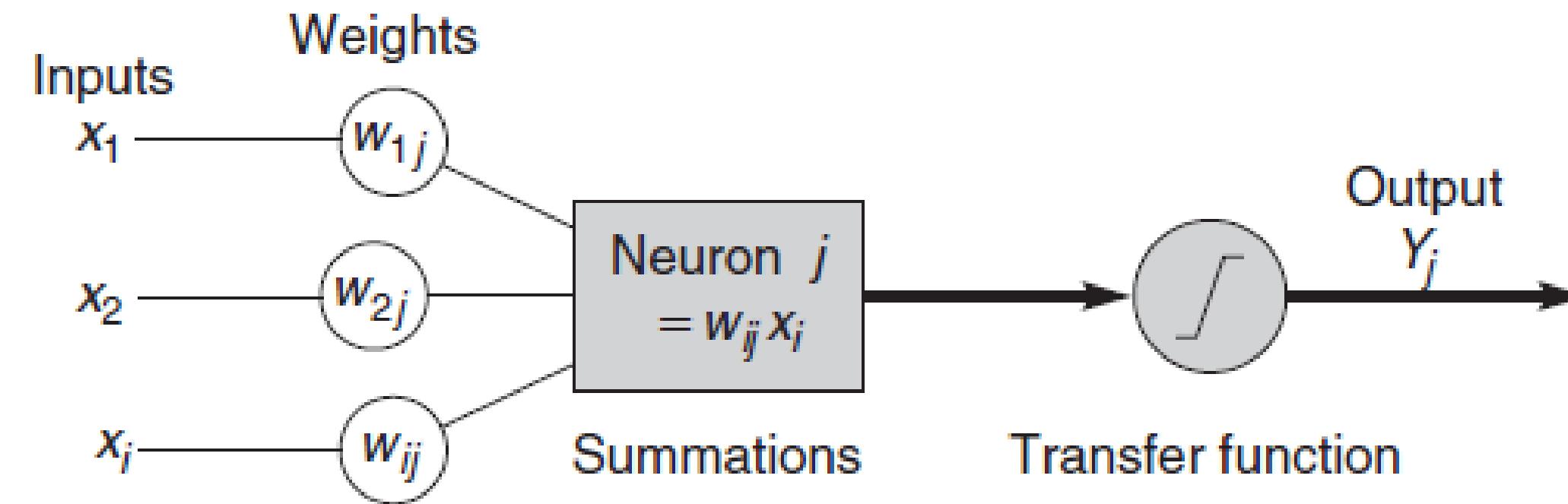
INTRODUCTION TO ARTIFICIAL INTELLIGENCE

PGS. TS. Nguyễn Thanh Bình



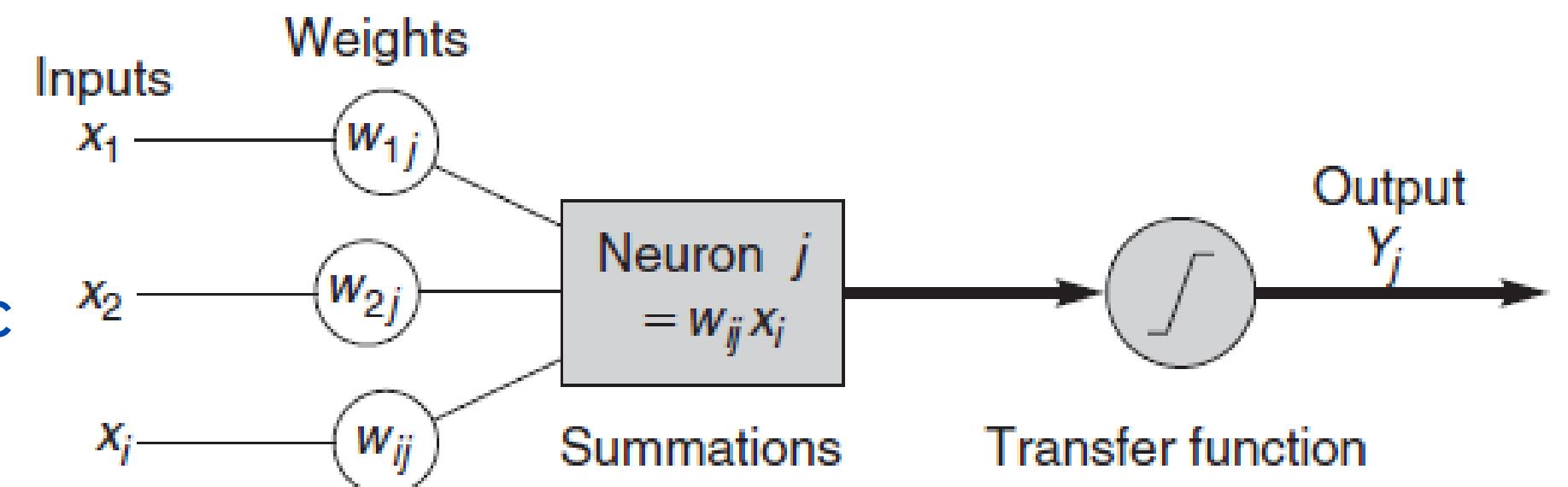
QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- Sau đây là mô hình chi tiết cho quá trình xử lý thông tin trên ANN



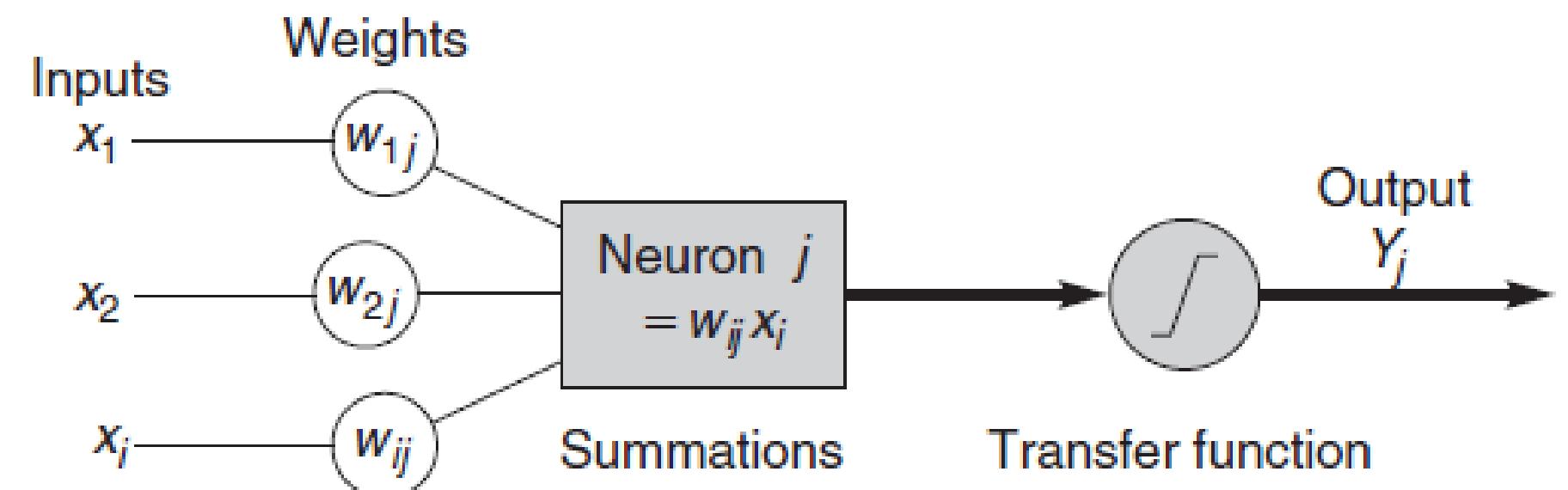
QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- **Inputs:** mỗi input tương ứng với một thuộc tính (*attribute*) của dữ liệu (*patterns*).
 - Ví dụ: xét hệ thống đánh giá mức độ rủi ro cho vay trong ngân hàng, mỗi input là một thuộc tính của khách hàng như thu nhập, nghề nghiệp, giới tính...



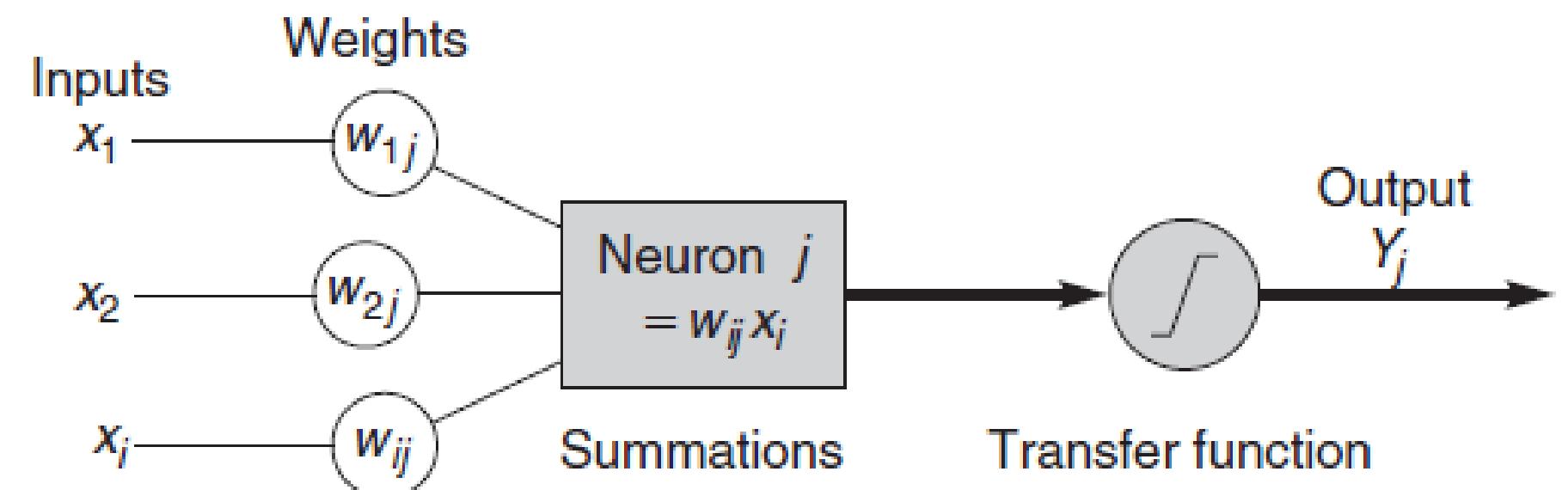
QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- **Outputs:** là kết quả của Artificial Neuron Networks hay một giải pháp cho một vấn đề.
 - Ví dụ: trong bài toán xem xét chấp nhận cho khách hàng vay tiền trong ngân hàng sẽ là cho vay hay không cho vay...



QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

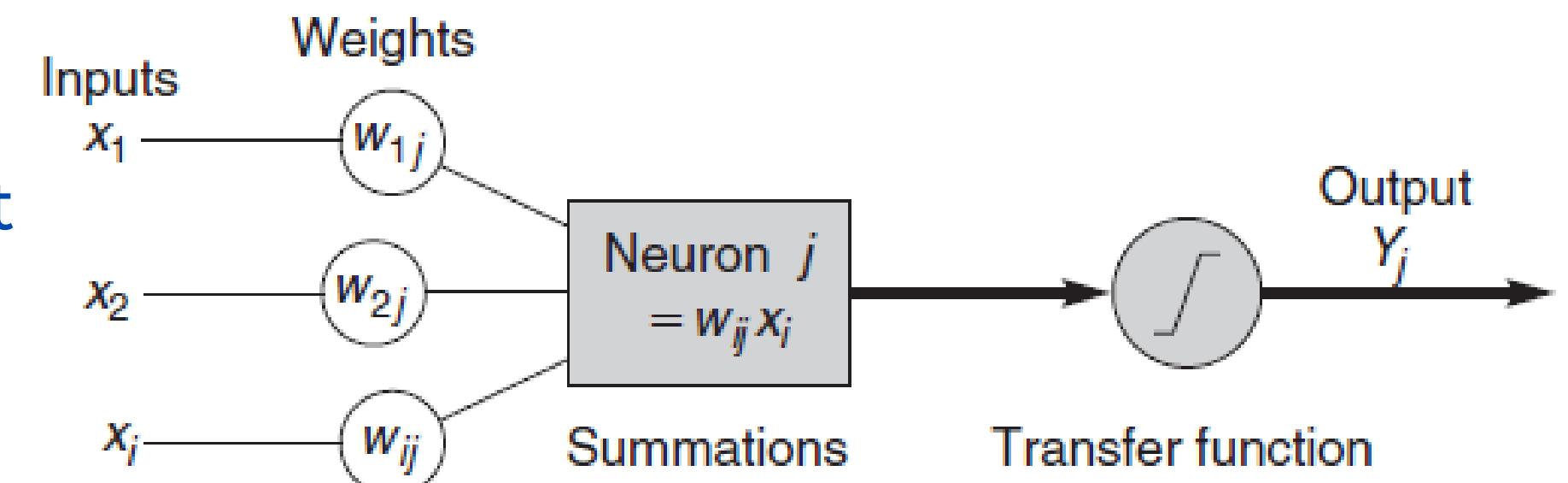
- **Trọng số liên kết (connection weights):** là thành phần vô cùng quan trọng trong một hệ thống mạng neuron nhân tạo. Nó thể hiện mức độ quan trọng của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu giữa các layer)



QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- **Hàm tổng (summation function):**
tính tổng trọng số của tất cả các input được đưa vào mỗi neuron. Một hàm tổng của một neuron đối với n input sẽ được tính theo công thức sau đây:

$$Y = \sum_{i=1}^n X_i W_i$$

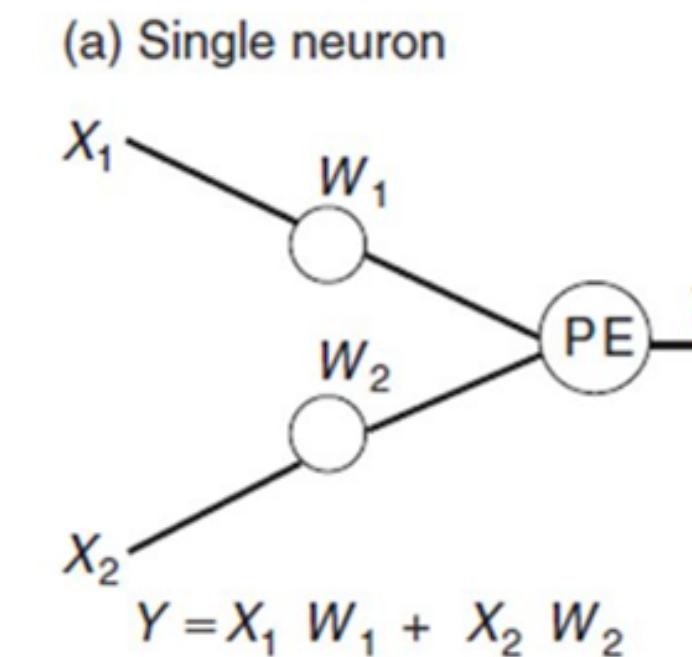




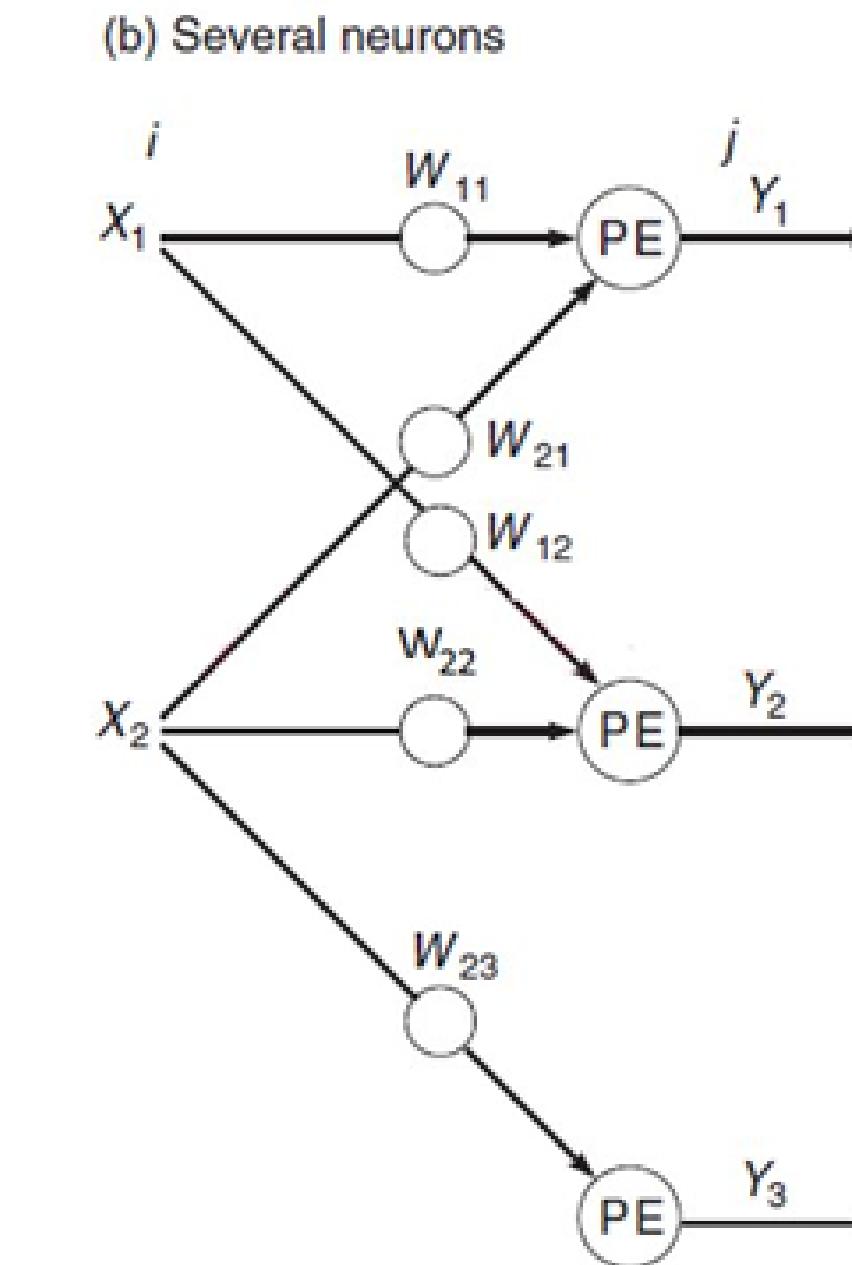
QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- Hàm tổng đối với nhiều neurons trong cùng một layer:

$$Y_j = \sum_{i=1}^n X_i W_{ij}$$



PE = processing element



QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

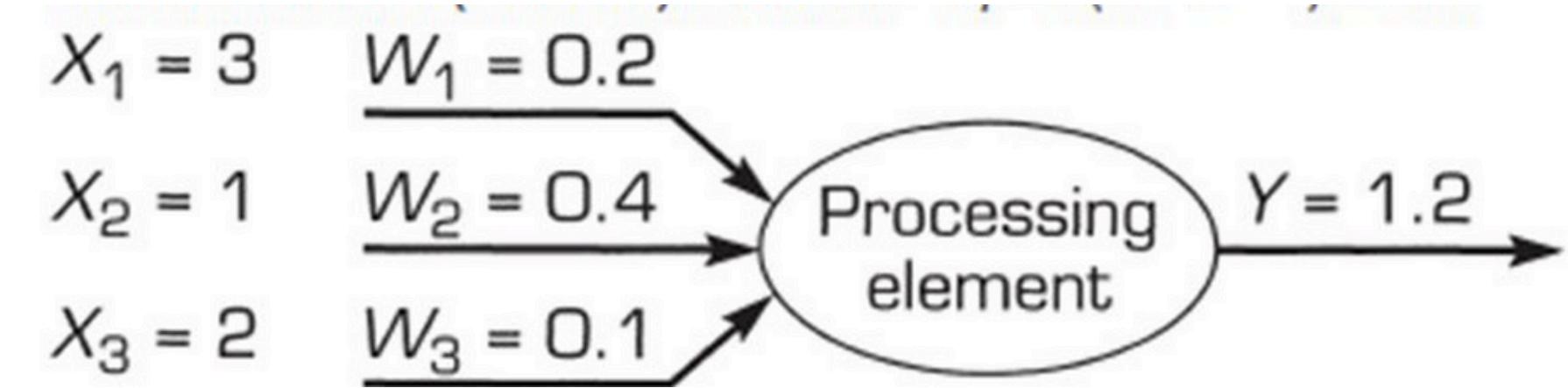
- **Hàm chuyển đổi** (*transformation function*)
 - Hàm tổng của một neuron cho chúng ta biết khả năng kích hoạt của neuron đó và còn gọi là kích hoạt bên trong (*internal activation*).
 - Các neuron này có thể sinh ra một output hoặc không trong hệ thống mạng neuron nhân tạo hay nói cách khác output của một neuron có thể được chuyển đến layer tiếp theo trong mạng neuron hay không.
 - Mỗi quan hệ giữa Internal Activation và kết quả (Output) được thể hiện bằng hàm chuyển đổi (*transfer function*)



QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- Việc lựa chọn hàm chuyển đổi có tác động lớn đến kết quả ANN. Hàm chuyển đổi phi tuyến hay sử dụng trong mạng neuron nhân tạo là **sigmoid** (logical activation) function

$$Y_T = 1/(1 + e^{-Y})$$



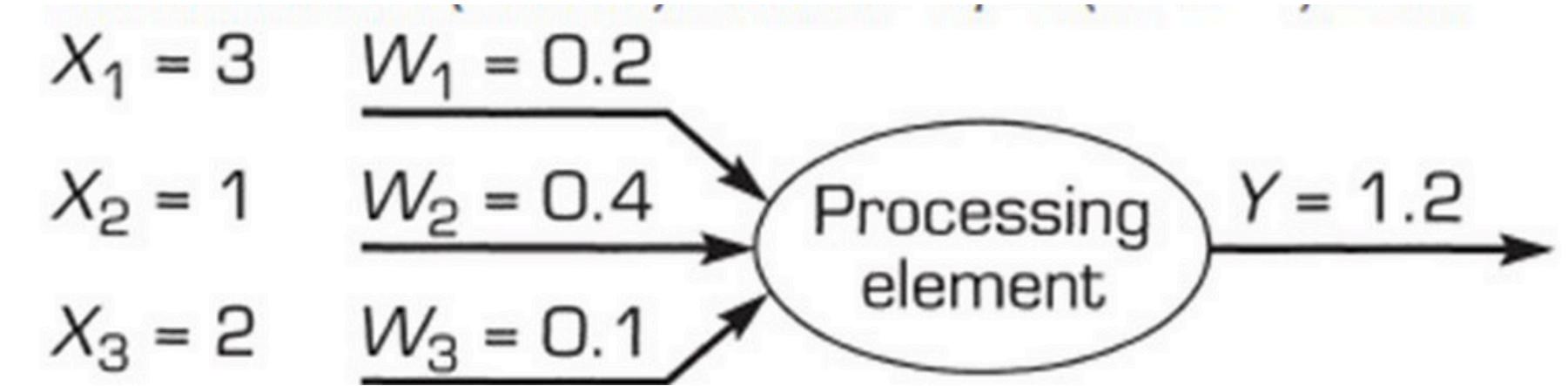
Summation function: $Y = 3(0.2) + 1(0.4) + 2(0.1) = 1.2$

Transformation function: $Y_T = 1/(1 + e^{-1.2}) = 0.77$



QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- Kết quả của sigmoid function thuộc khoảng [0,1] nên còn gọi là **hàm chuẩn hóa (normalized function)**. Kết quả xử lý tại các neuron đôi khi rất lớn. Chính vì thế, hàm chuyển đổi được sử dụng để xử lý những kết quả này trước khi chuyển đến layer tiếp theo



Summation function: $Y = 3(0.2) + 1(0.4) + 2(0.1) = 1.2$

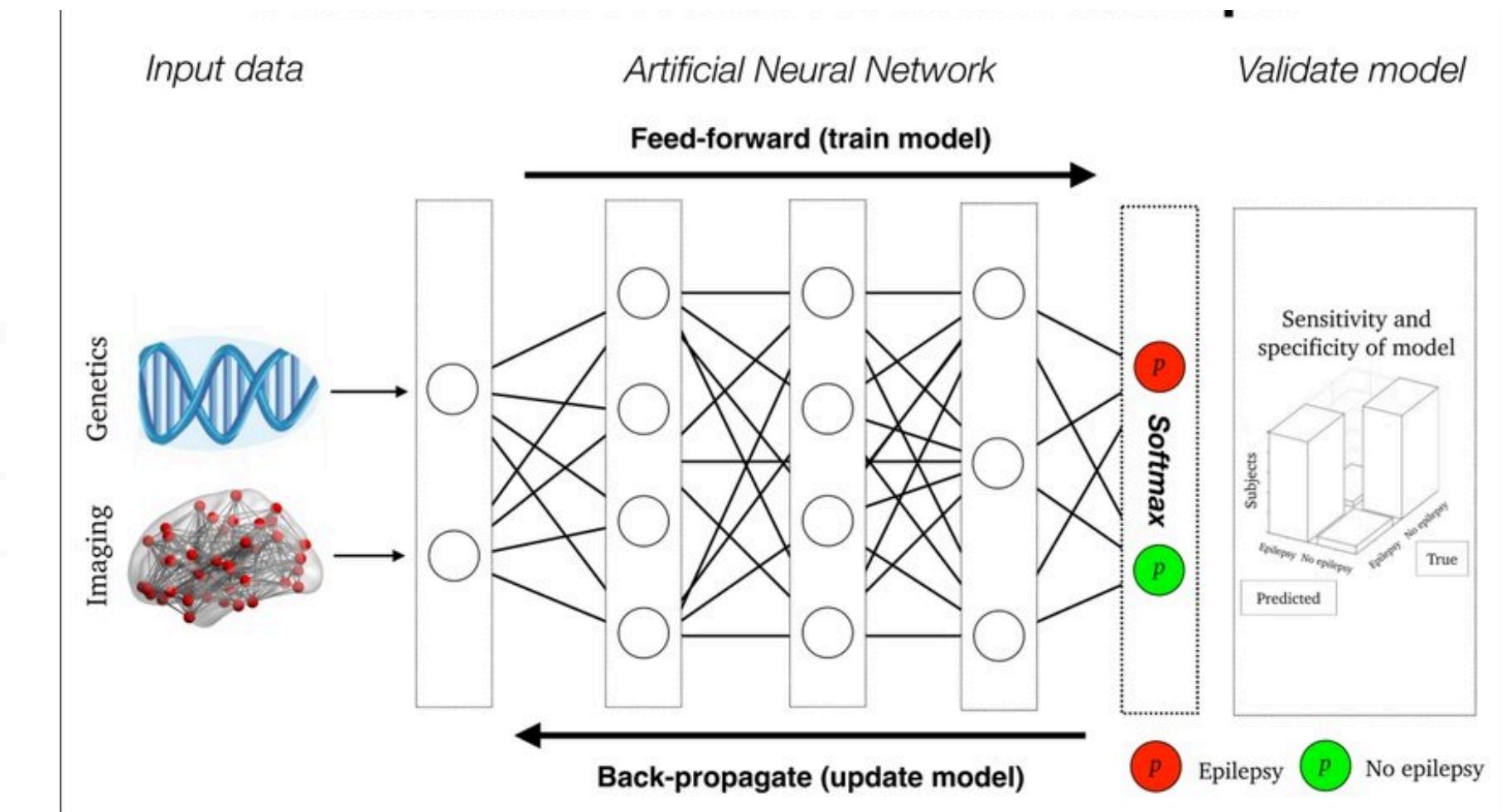
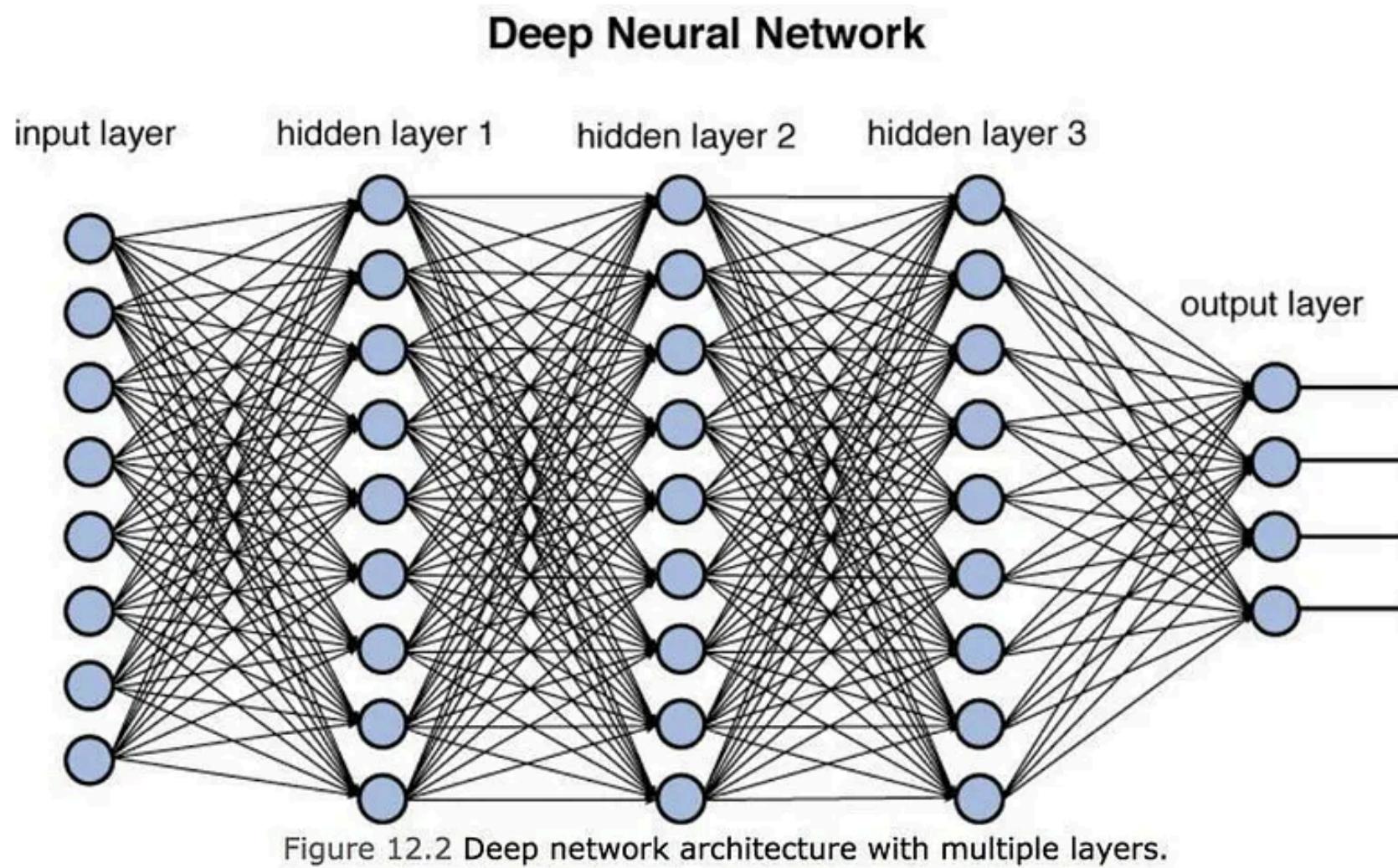
Transformation function: $Y_T = 1/(1 + e^{-1.2}) = 0.77$

QUÁ TRÌNH XỬ LÝ THÔNG TIN CỦA ANN

- Trong thực tế, thay vì sử dụng các hàm chuyển đổi đã nêu trên, người ta có thể sử dụng **giá trị ngưỡng** (*threshold value*) để kiểm soát các output của các neuron tại một layer nào đó trước khi chuyển các output này đến các layer tiếp theo. Nếu output của một neuron nào đó nhỏ hơn threshold thì nó sẽ không được chuyển đến layer tiếp theo.

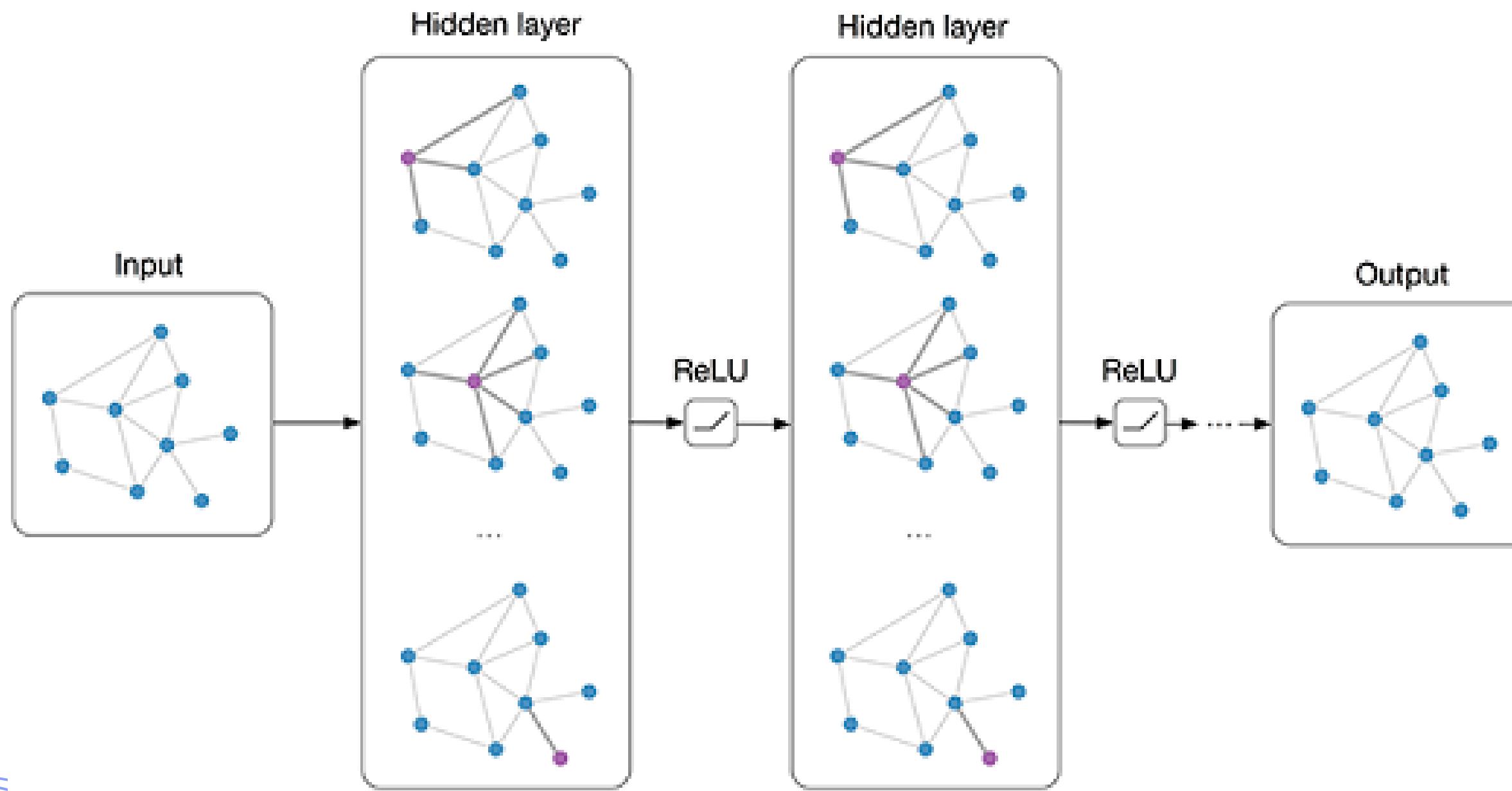


MỘT SỐ VÍ DỤ





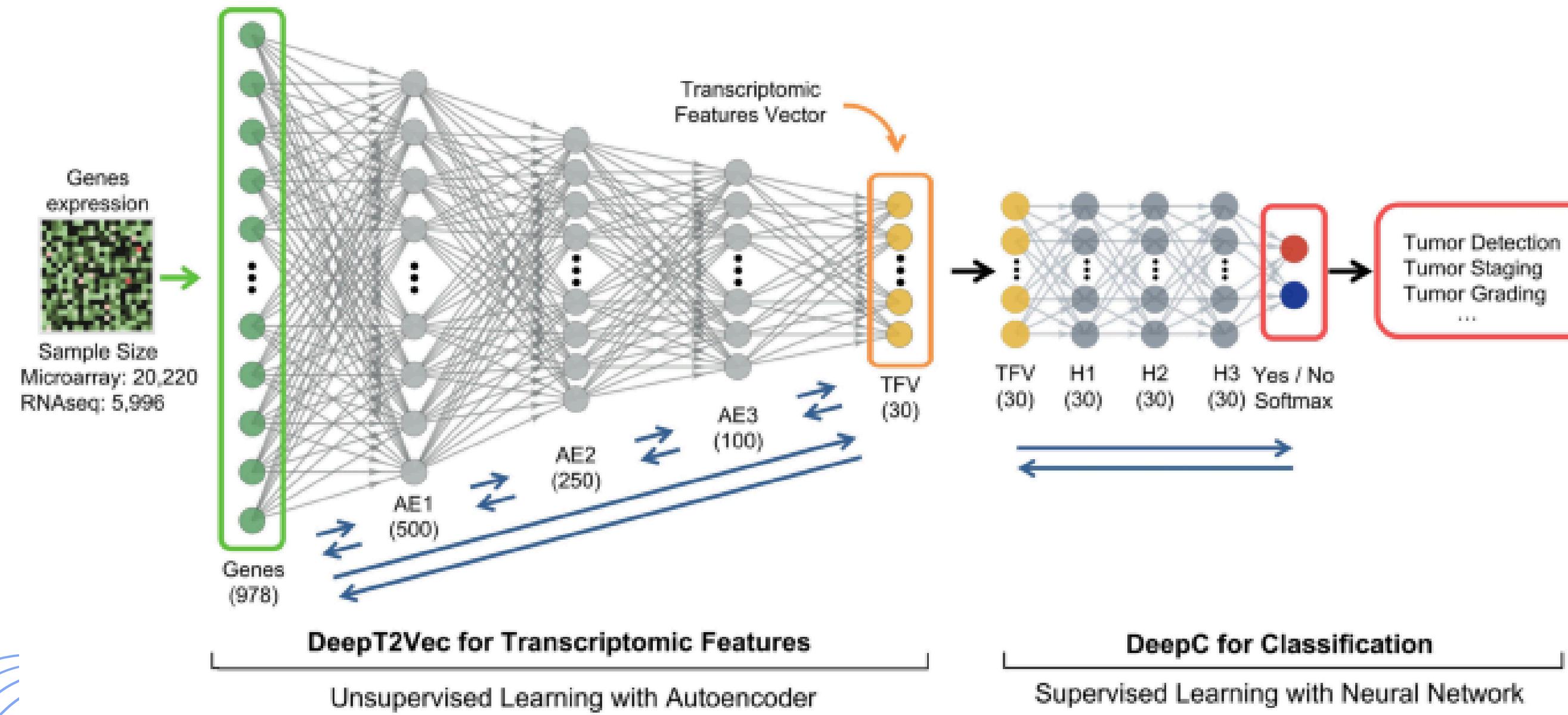
MỘT SỐ VÍ DỤ



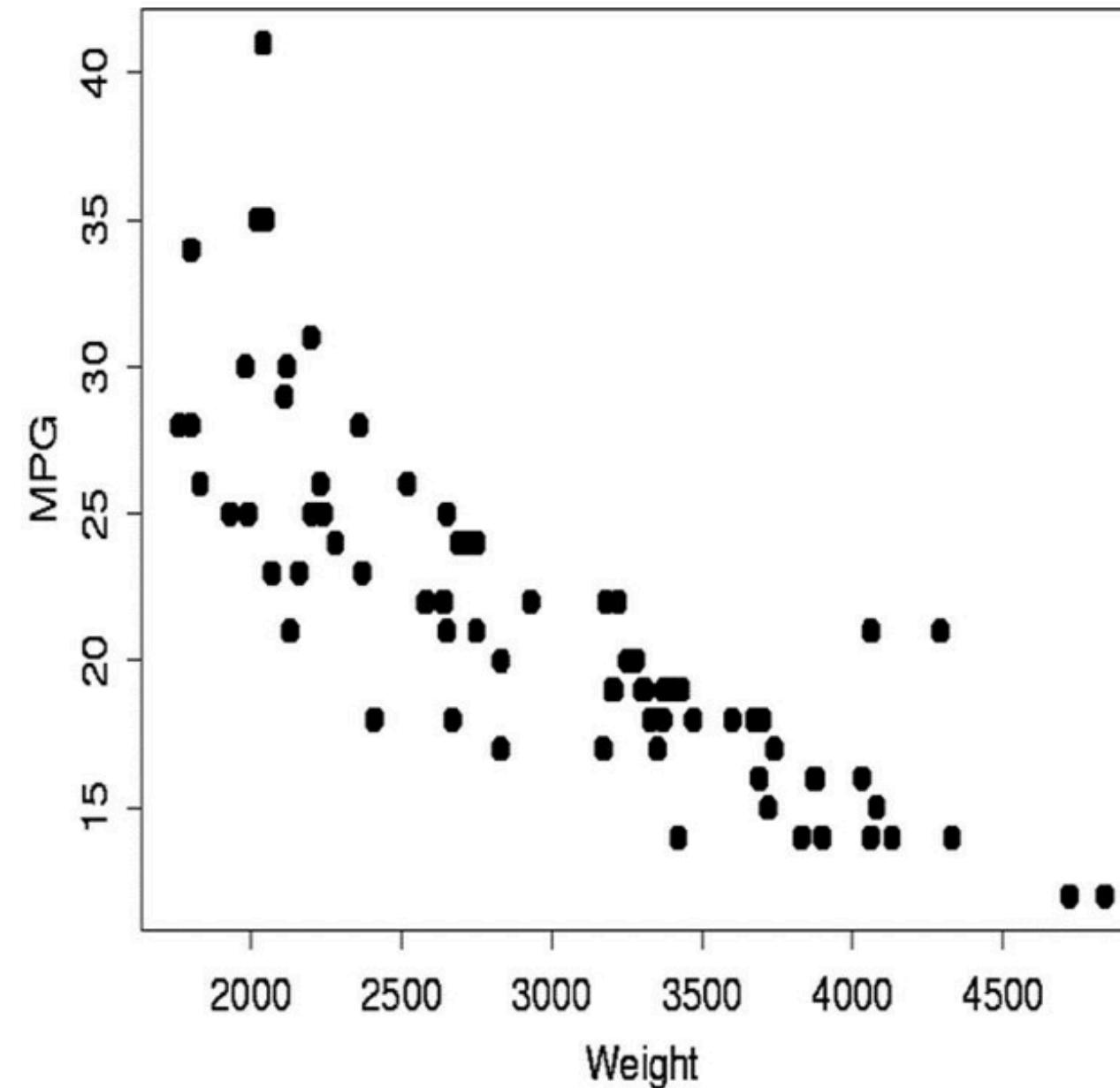


MỘT SỐ VÍ DỤ

Deep Diagnosis for Cancer (DeepDCancer)



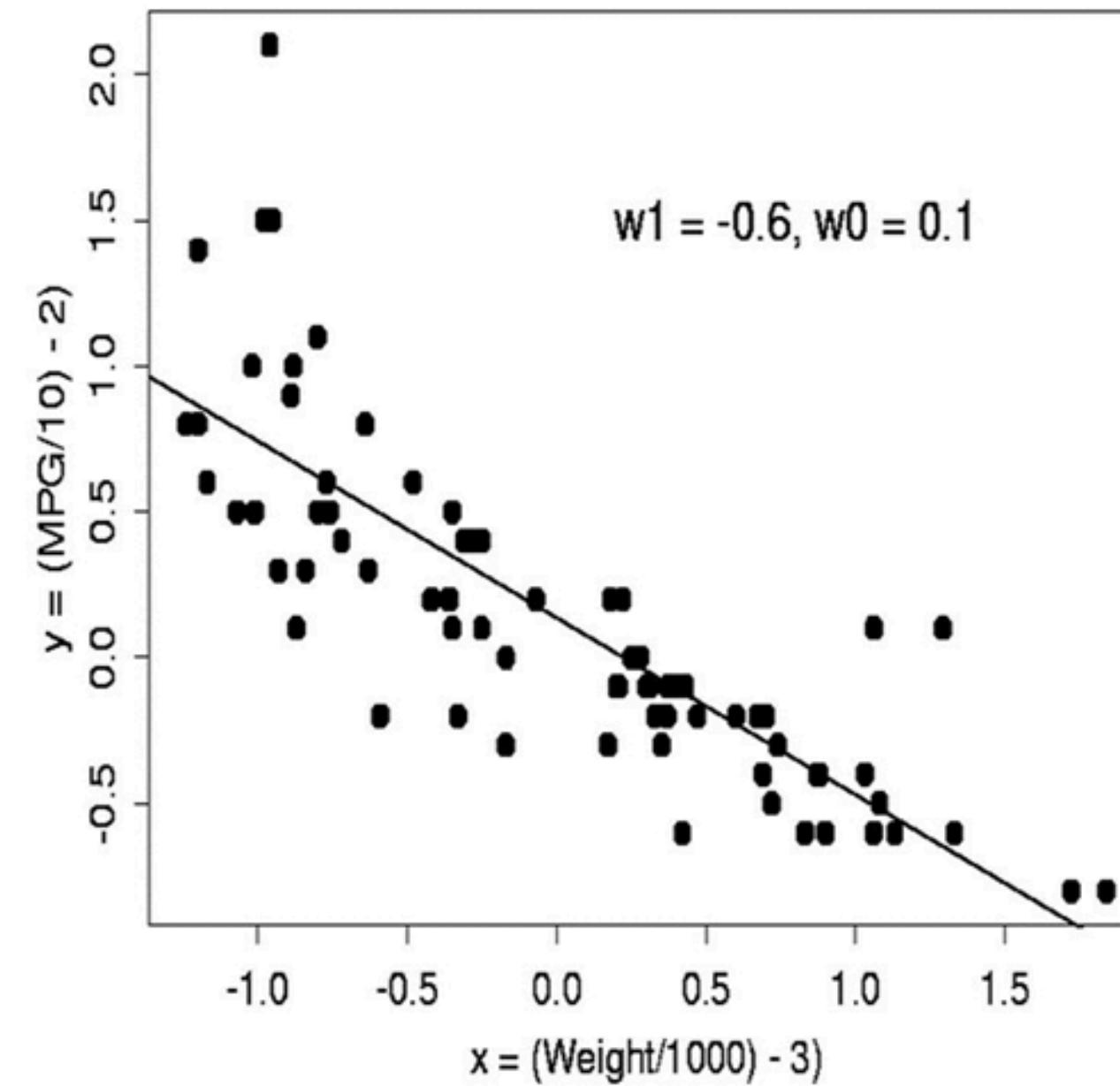
LINEAR REGRESSION



- Each dot provides the information about the weight and fuel consumption (mile per gallon) for one of 74 cars.
- Goal: given the 75th car. How to predict how much fuel it will use.
- Model:

$$y = w_1x + w_0$$

MINIMIZE THE LOSS FUNCTION



- For linear models, one can use linear regression to minimize the loss function or sum-squared error function:

$$E = \frac{1}{2} \sum_p (t_p - y_p)^2$$

- Here, t_p is a target value (actual fuel consumption)

MINIMIZE THE LOSS FUNCTION

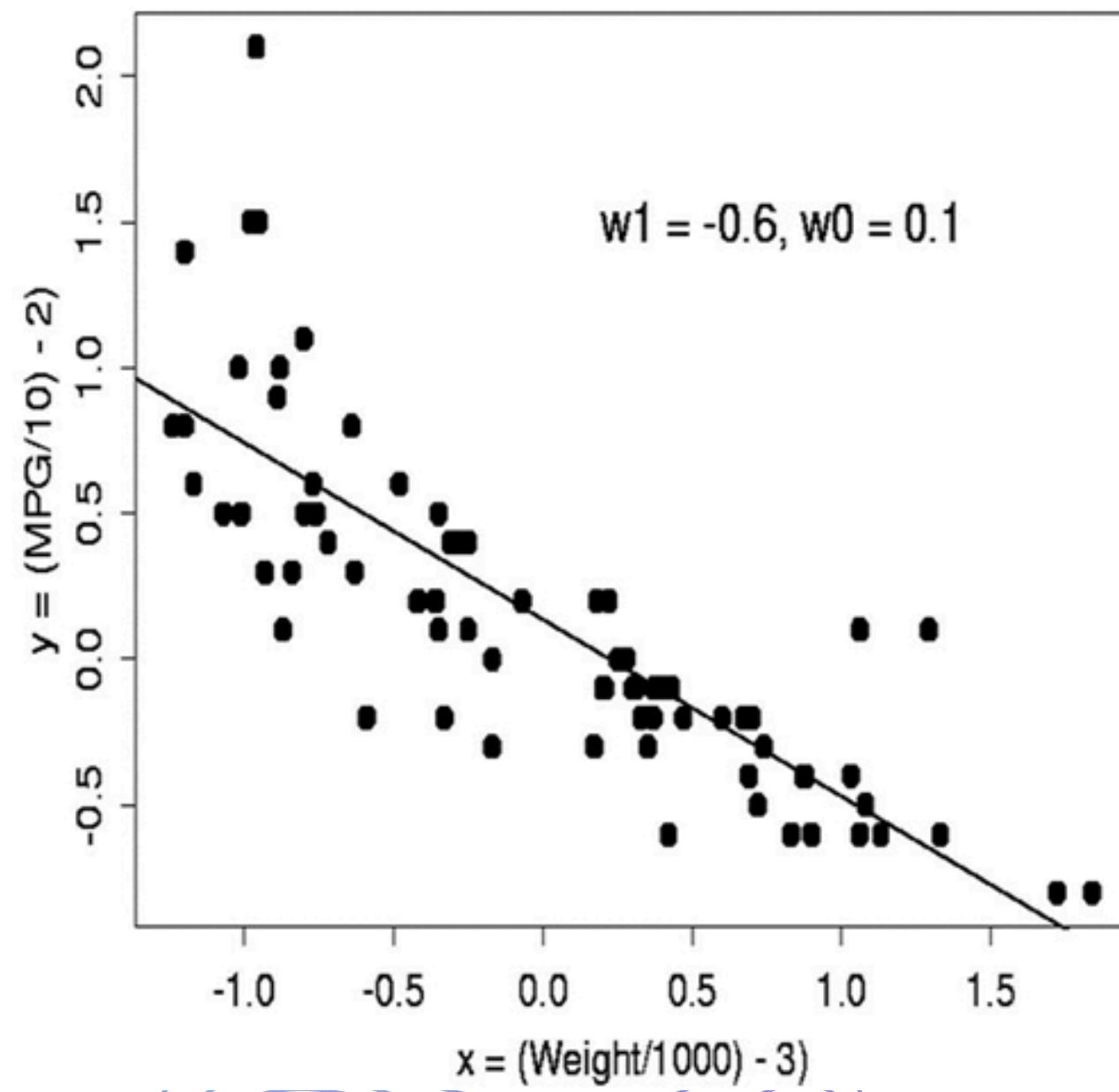
- The linear regression can be solved by an iterative numerical technique, named **gradient descent**:
 - Step 1: choose some initial values for the model parameters.
 - Step 2: Calculate the gradient G of the loss function with respect to each model parameter.
 - Step 3: change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error.
 - Step 4: repeat step 2 and 3 until G gets close to zero.

MINIMIZE THE LOSS FUNCTION

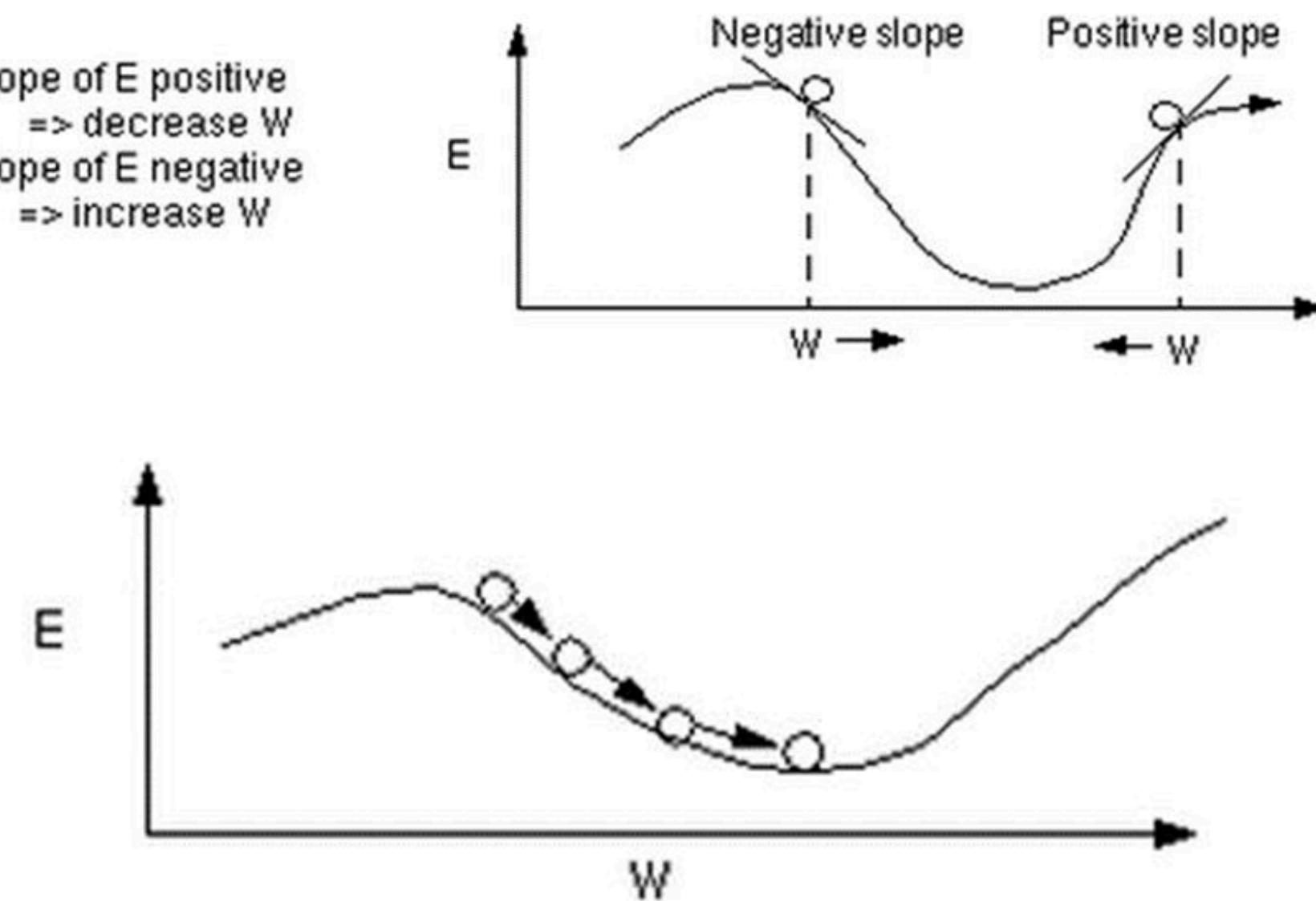
- The linear regression can be solved by an iterative numerical technique, named **gradient descent**:
 - Step 1: choose some initial values for the model parameters.
 - Step 2: Calculate the gradient G of the loss function with respect to each model parameter.
 - Step 3: change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error.
 - Step 4: repeat step 2 and 3 until G gets close to zero.



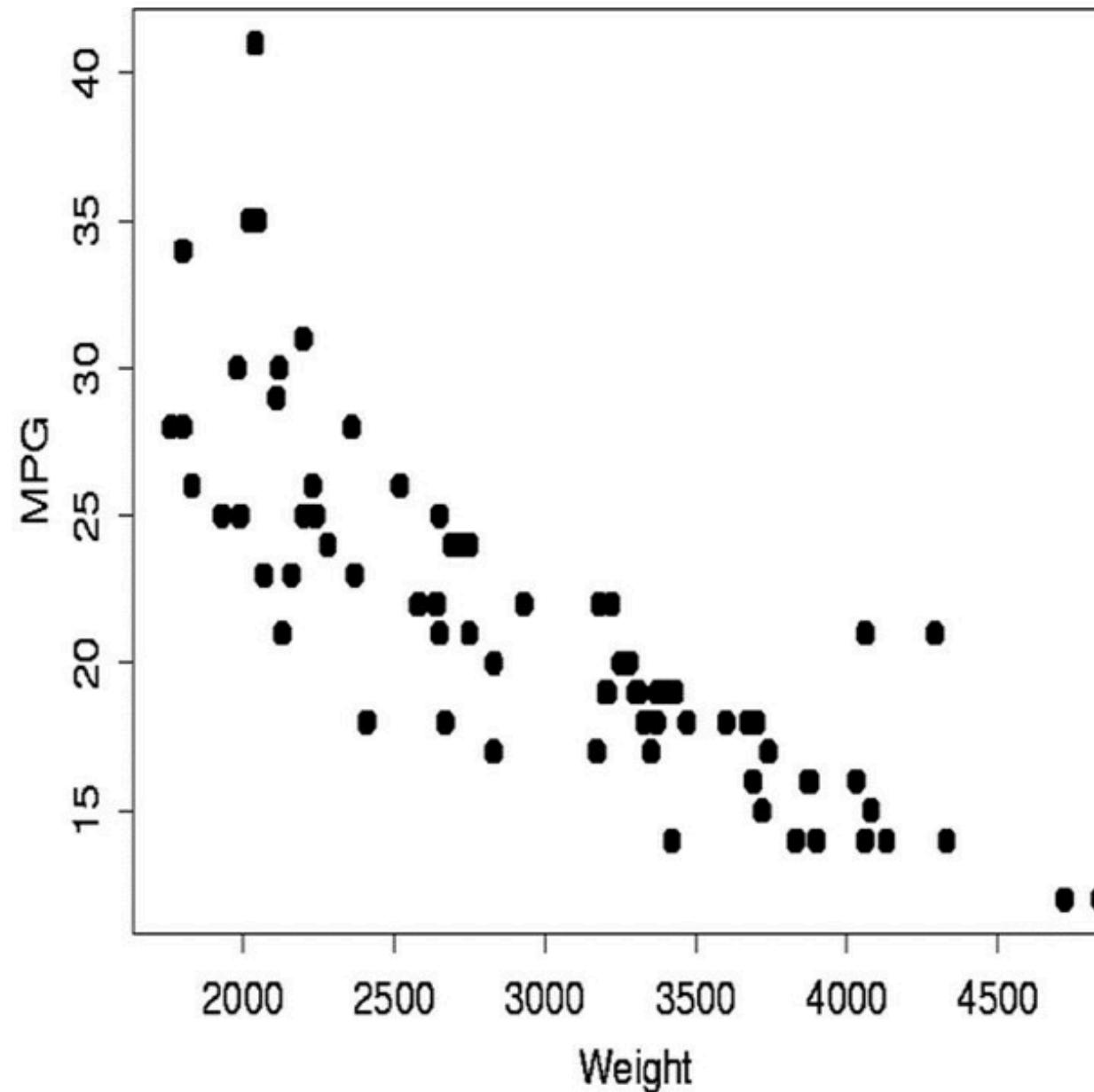
MINIMIZE THE LOSS FUNCTION



Slope of E positive
=> decrease W
Slope of E negative
=> increase W



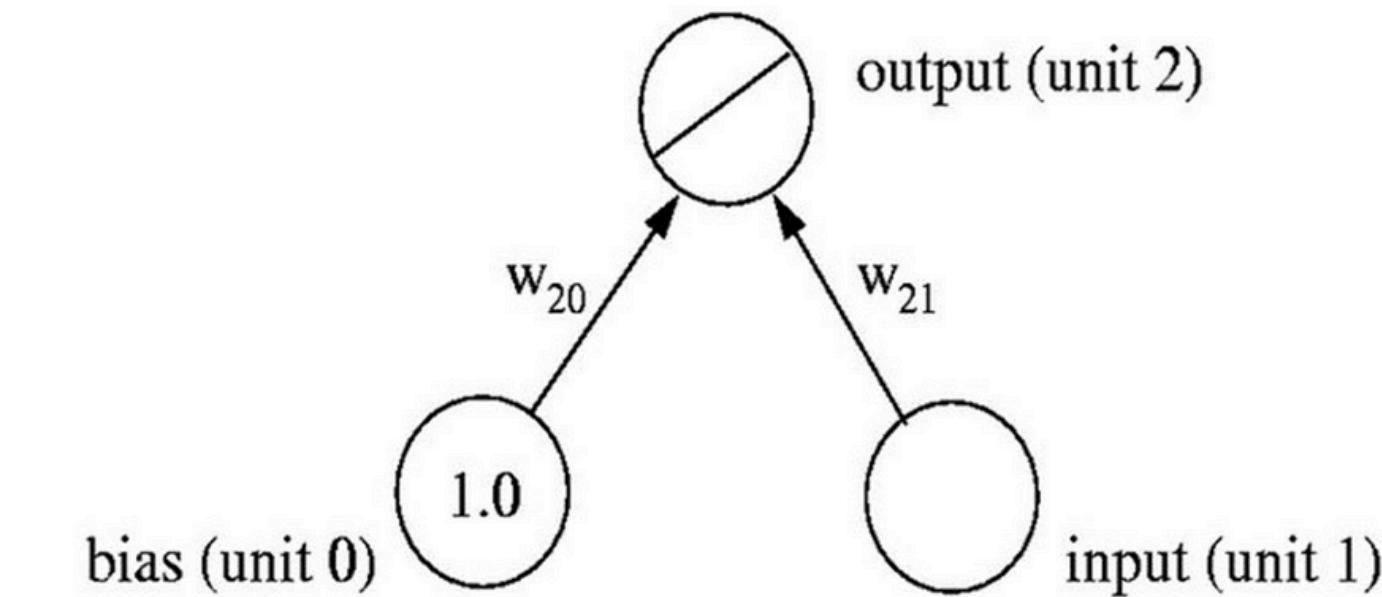
USING NEURAL NETWORKS



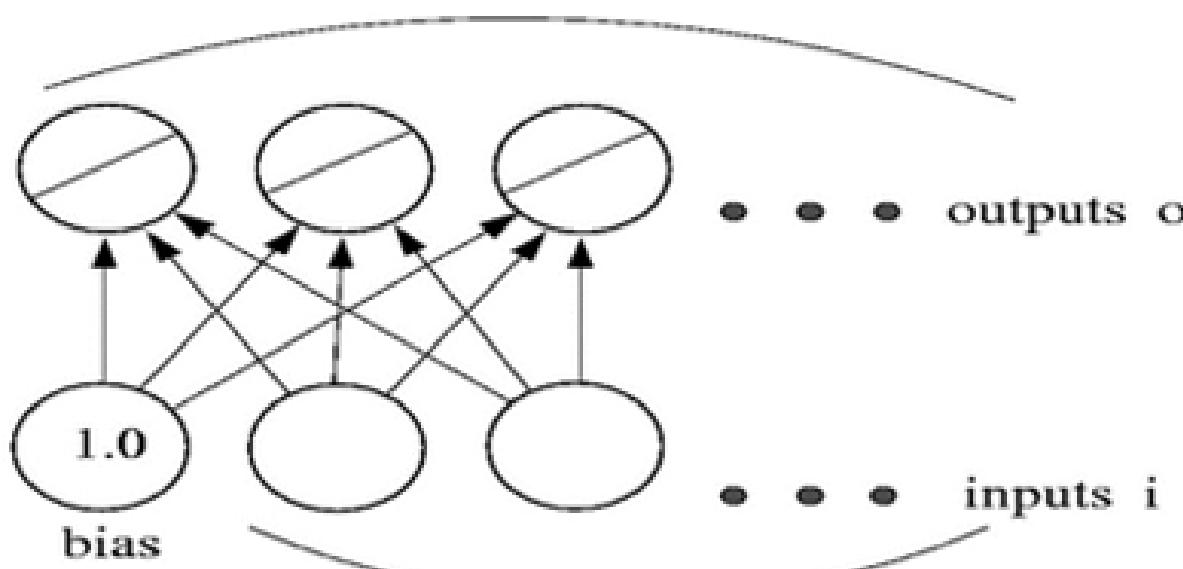
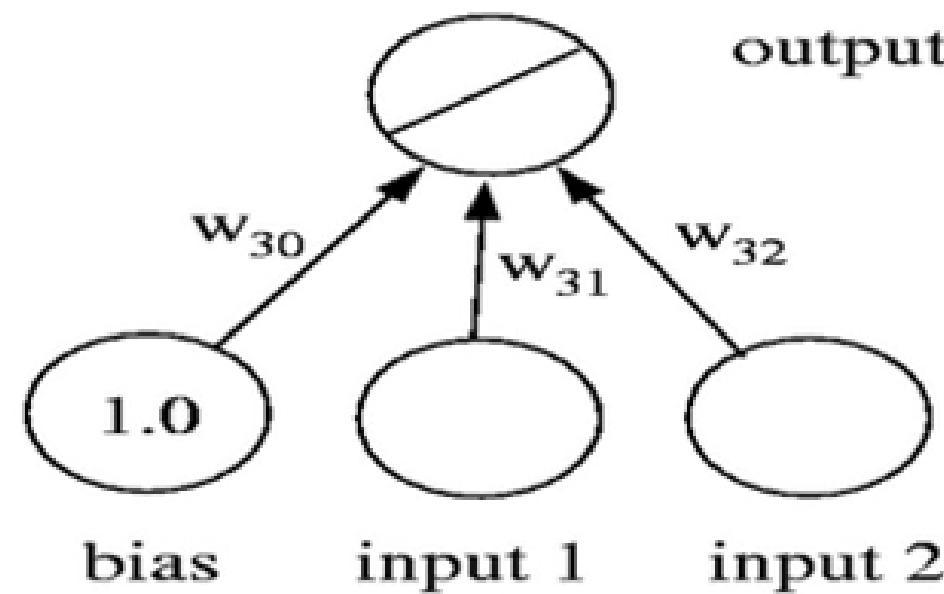
- Model

$$y_2 = y_1 w_{21} + 1.0 w_{20}$$

- Here, it consists of a bias unit, an input unit and a linear output unit:



LINEAR NEURAL NETWORKS



- The neural network has a typical layered structure: a layer of input units (including the bias), connected by a layer of weights to a layer of output units.
- The corresponding loss function is:

$$E = \sum_p E^p, \quad E^p = \frac{1}{2} \sum_p (t_0^p - y_0^p)^2$$

for each point p in the training data

THE GRADIENT DESCENT ALGORITHM

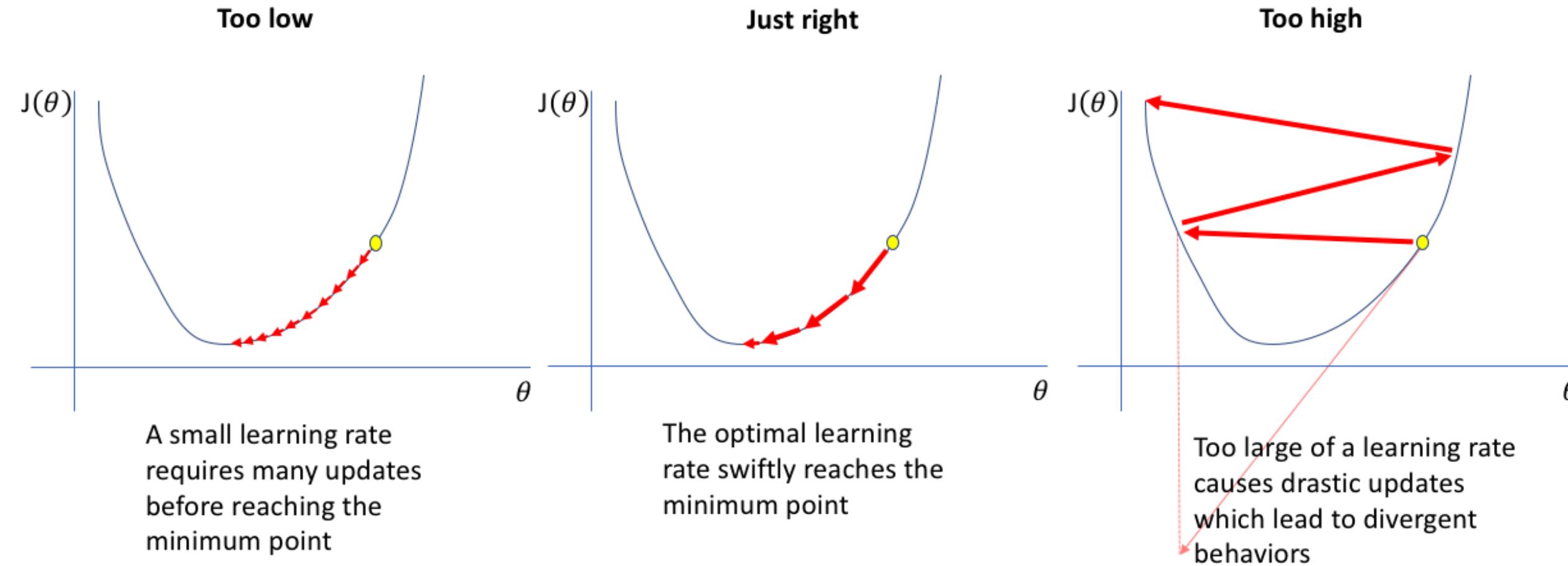
- Initialize all weights to a small random values.
- REPEAT until done
 - For each weight w_{ij} , set $\Delta w_{ij} = 0$.
 - For each data point $(x,t)^p$:
 - Set input units to x .
 - Compute the value of output units.
 - For each weight w_{ij} , set: $\Delta w_{ij} = \Delta w_{ij} + (t_i - y_i) y_j$
 - For each weight w_{ij} , set: $w_{ij} = w_{ij} + \mu \Delta w_{ij}$
 - Here μ is the learning rate.



THE GRADIENT DESCENT ALGORITHM

	general case	linear network
Training data	(\mathbf{x}, t)	(\mathbf{x}, t)
Model parameters	\mathbf{w}	\mathbf{w}
Model	$\mathbf{y} = g(\mathbf{w}, \mathbf{x})$	$y_o = \sum_j w_{oj} y_j$
Error function	$E(\mathbf{y}, \mathbf{t})$	$E = \sum_p E^p, \quad E^p = \frac{1}{2} \sum_o (t_o^p - y_o^p)^2$
Gradient with respect to w_{ij}	$\frac{\partial E}{\partial w_{ij}}$	$- (t_i - y_i) y_j$
Weight update rule	$\Delta w_{ij} = -\mu \frac{\partial E}{\partial w_{ij}}$	$\Delta w_{oi} = \mu (t_o - y_o) y_i$

THE GRADIENT DESCENT ALGORITHM



A small learning rate
requires many updates
before reaching the
minimum point

The optimal learning
rate swiftly reaches the
minimum point

Too large of a learning rate
causes drastic updates
which lead to divergent
behaviors