Session 48

Neural Network

Table of Content

Apa yang Akan Kita Pelajari Hari Ini?

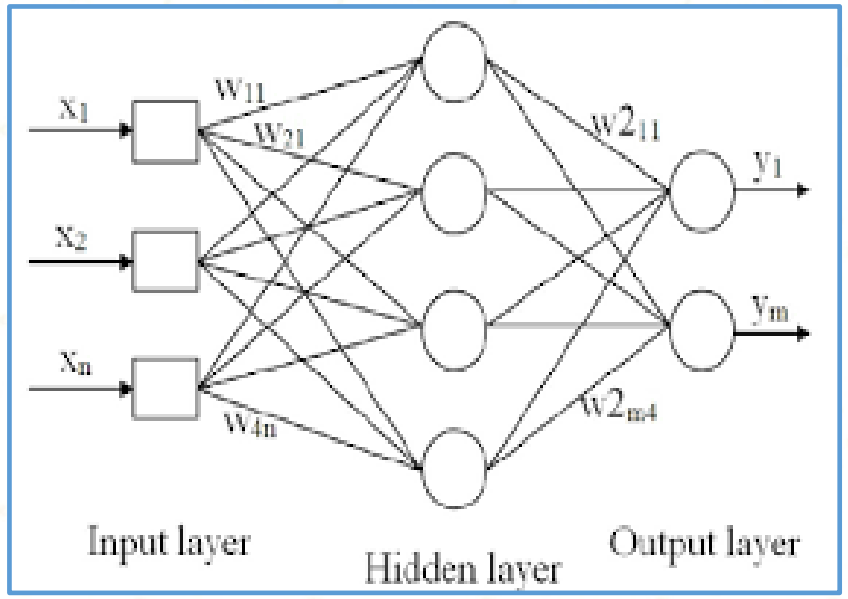
1. Perceptron
2. Multilayer perceptron
3. Backpropagation

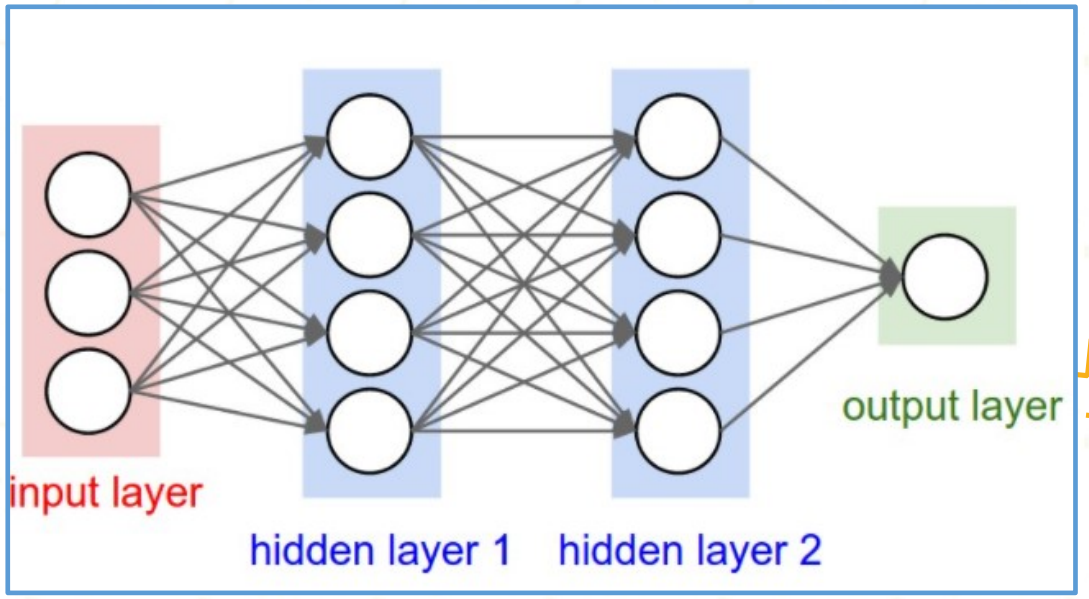
Brief History

- **History of ANN**
- 1943: McCulloch and Pitts: First mathematic model of neuron
- 1958: Rosenblatt: Perceptron - Single layer NN
- 1986: Rumelhart: Back Propagation algorithm
- 1995: Y. LeCun, Y. Bengio, et al.: Convolutional neural network
- 2006: G. E. Hinton, et al.: Deep belief nets.

Artificial Neural Network

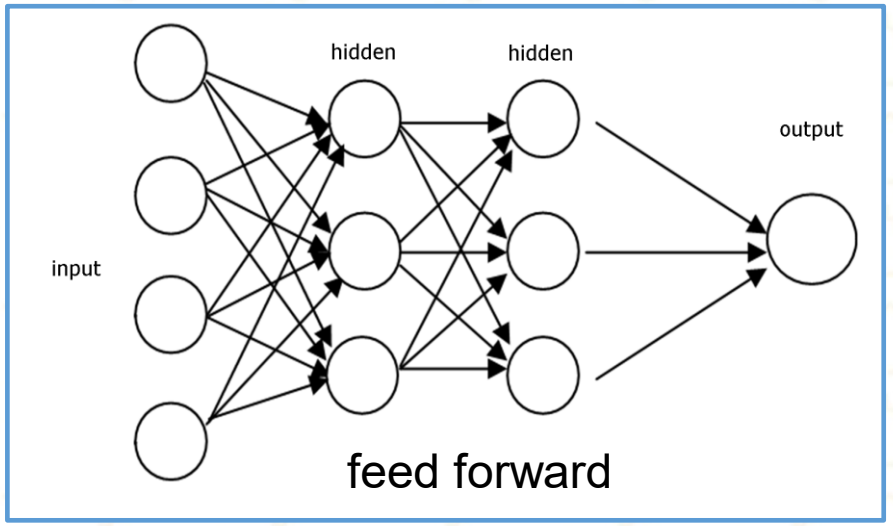
- Artificial Neural Network (Jaringan saraf tiruan) terdiri dari kumpulan unit pemrosesan sederhana yang berkomunikasi dengan mengirimkan sinyal satu sama lain melalui sejumlah besar koneksi berbobot.
- Model yang terinspirasi oleh bagaimana neuron dalam otak manusia bekerja



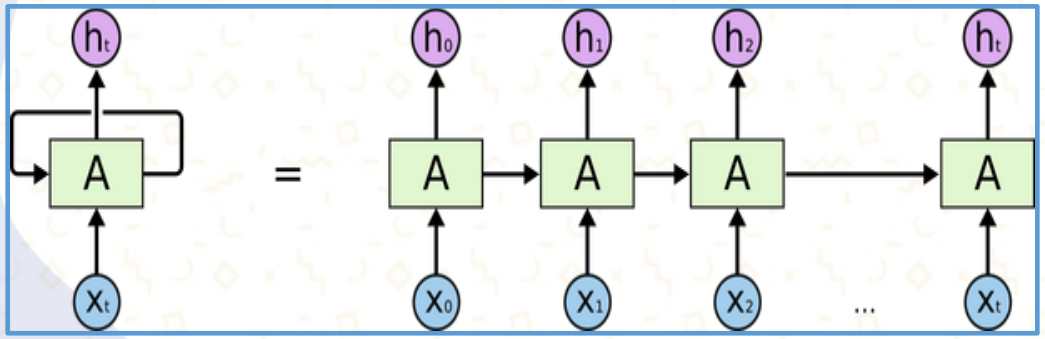


Types of ANN

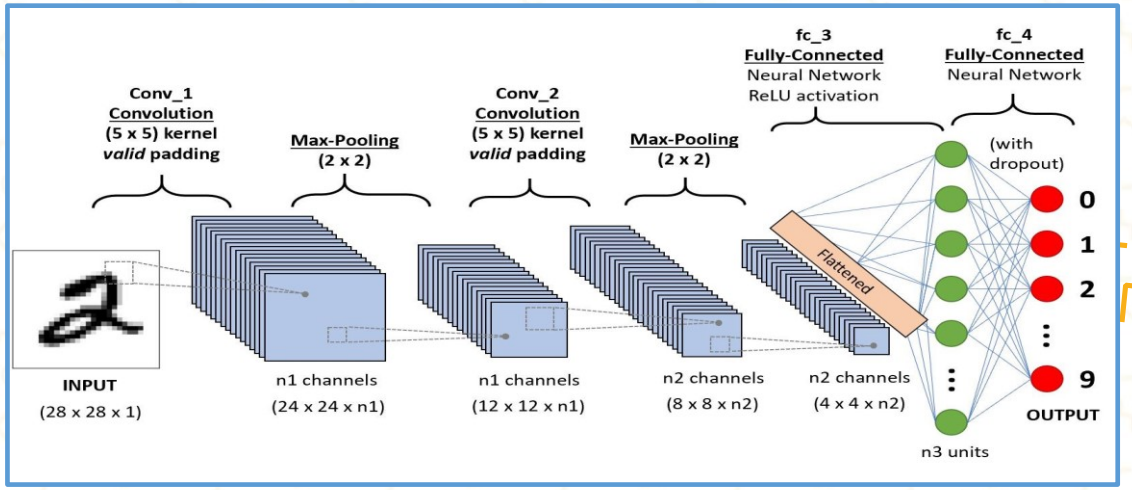
- **Types of ANN**
  - Feedforward neural network
  - Recurrent neural network (RNN)
  - Convolutional neural network (CNN)
  - etc,



feed forward



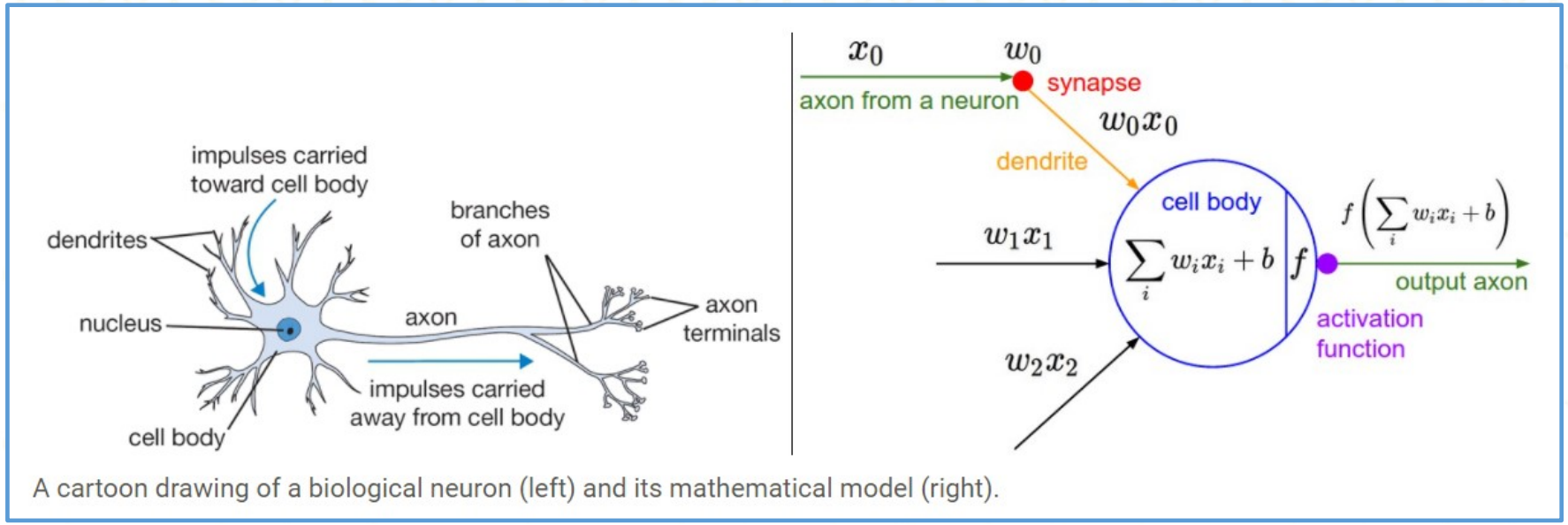
RNN



CNN

Biological motivation

- Neural Networks awalnya terinspirasi oleh tujuan pemodelan sistem saraf biologis.
- Unit komputasi dasar otak adalah neuron.
- Sekitar 86 miliar neuron dapat ditemukan di sistem saraf manusia

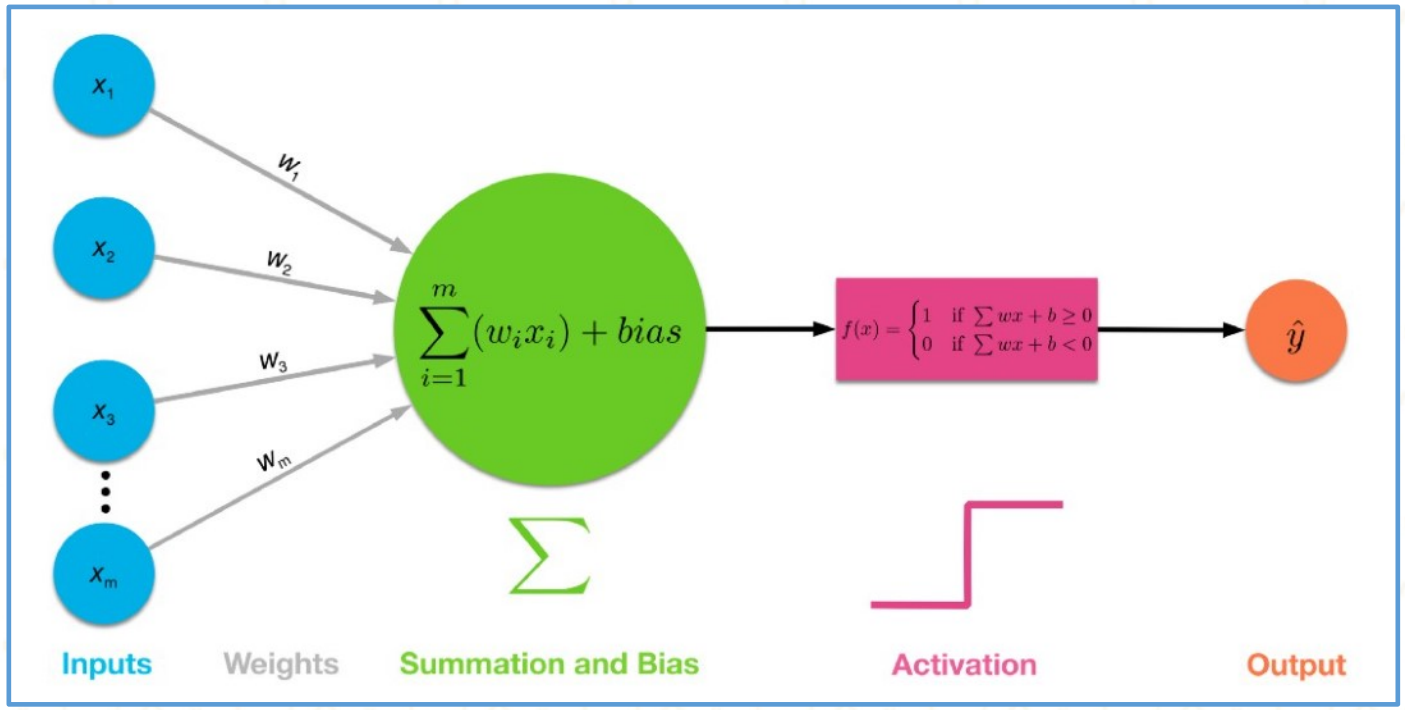


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

https://cs231n.github.io/neural-networks-1/

Perceptron

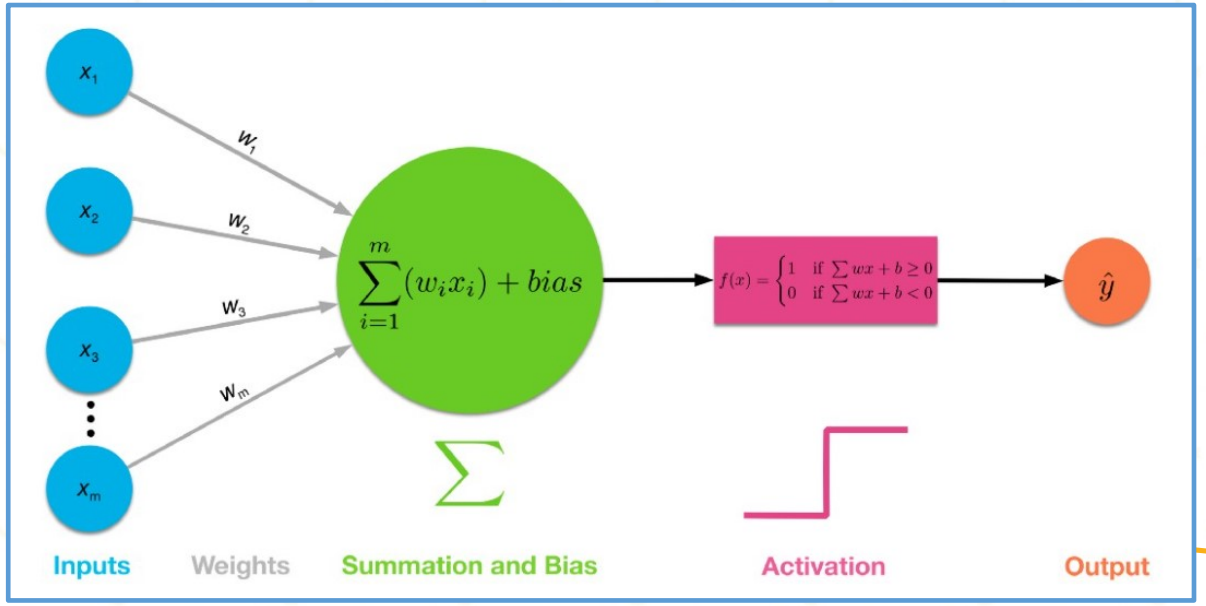
- Perceptron adalah pengklasifikasi linier (linear classifier).
- Frank Rosenblatt, seorang psikolog Amerika, mengusulkan model perceptron klasik pada tahun 1958



https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

Perceptron

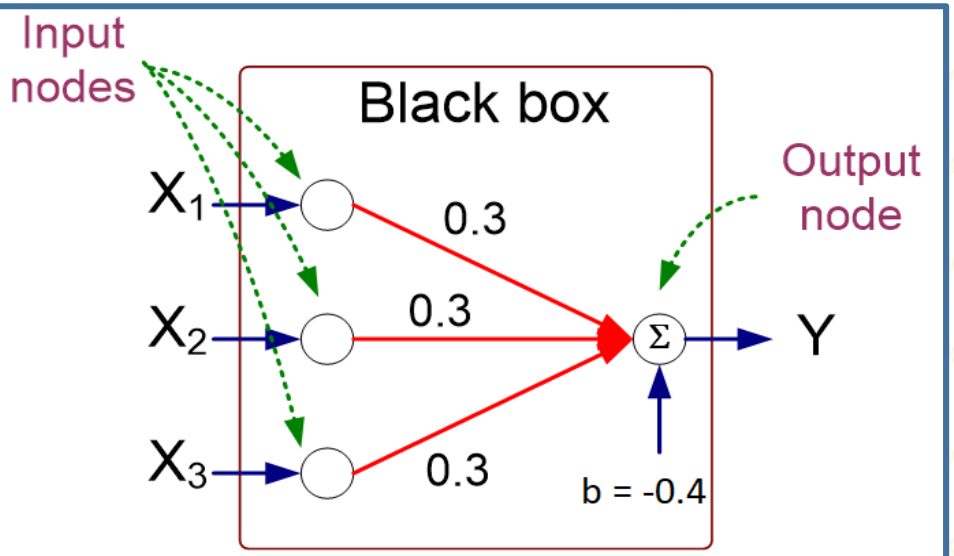
- Digunakan untuk mengklasifikasikan data input yang diberikan.
- The perceptron terdiri dari 4 bagian
  - Input values atau one input layer
  - Weights dan Bias
  - Net sum
  - Activation Function



Perceptron (how does it work?)

- Semua inputs x dikalikan dengan weights w
- Jumlahkan semua hasil perkalikan tersebut (kita sebut *Weighted Sum*)
- Mengaplikasikan activation function ke weighted sum

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



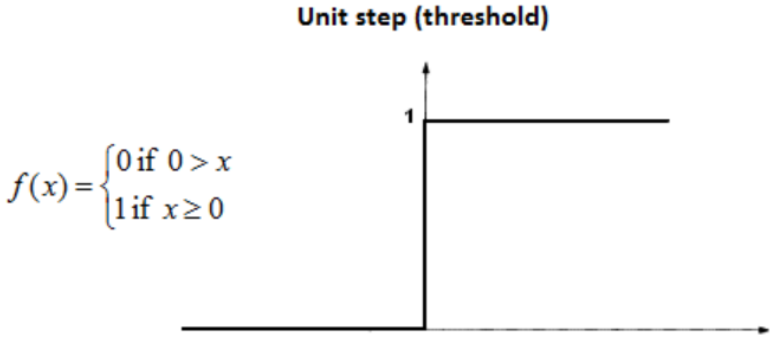
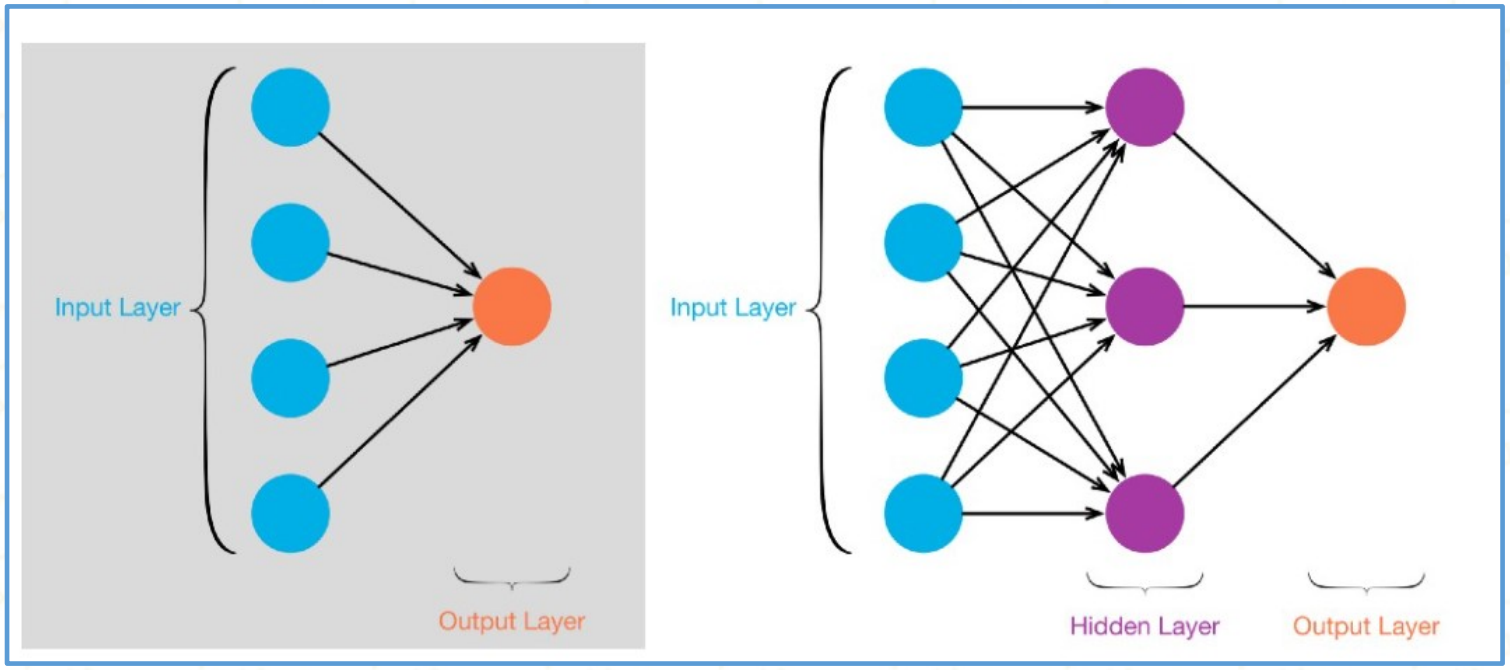


Fig: Unit Step Activation Function

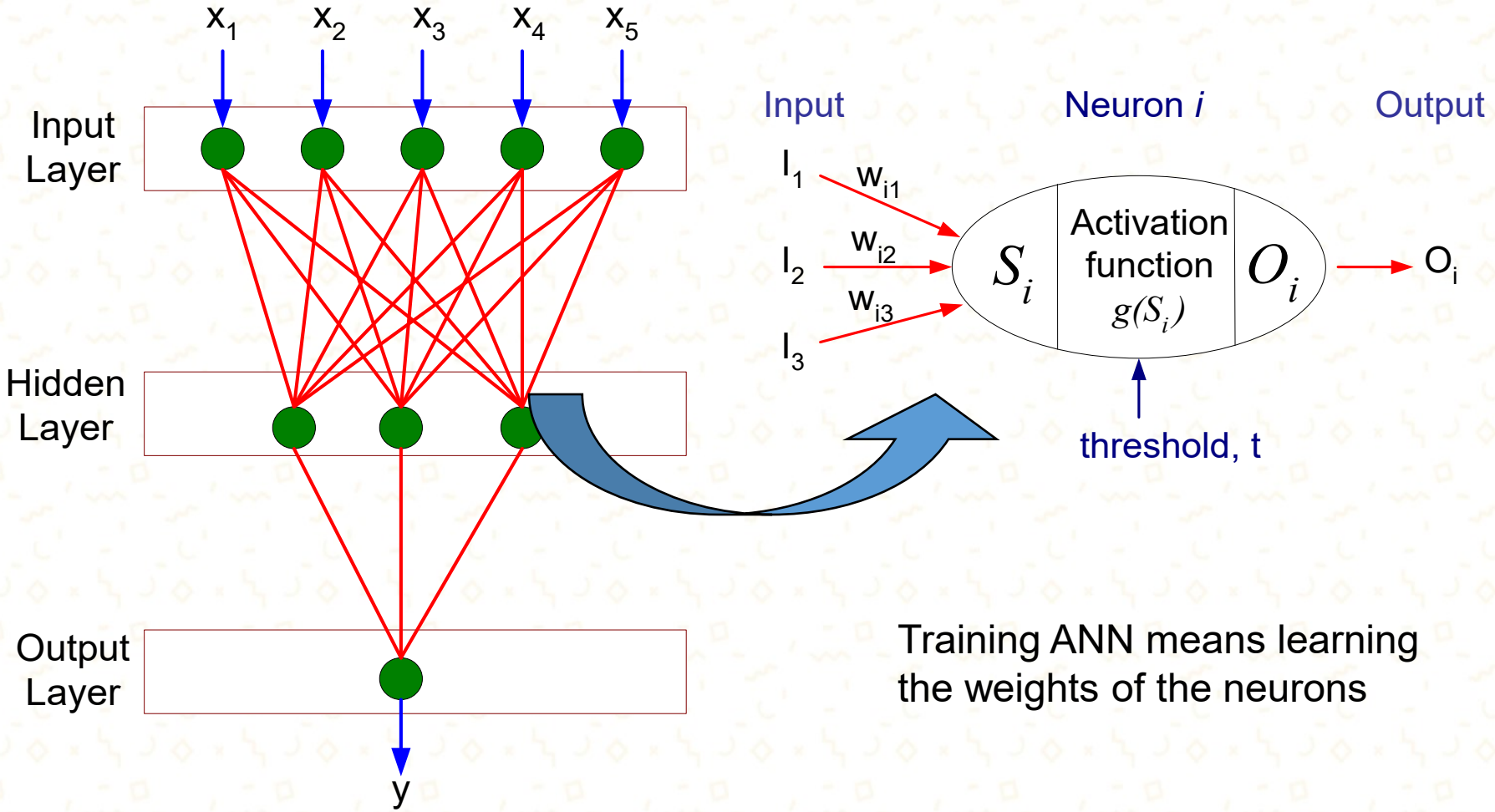
y = f(0.3*x1 + 0.3*x2 + 0.3*x3 - 0.4)

Multilayer perceptron

- Model ini terdiri dari tiga jenis layer — input layer, hidden layer, output layer.
- Kecuali node input, setiap node adalah neuron yang menggunakan fungsi aktivasi nonlinier.
- MLP menggunakan backpropagation untuk training-nya.



General Structure of MLP

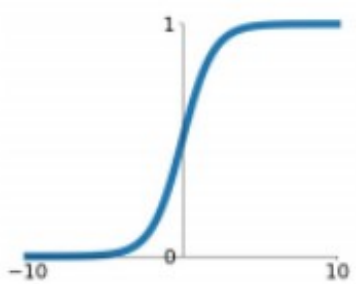


Training ANN means learning the weights of the neurons

Activation function

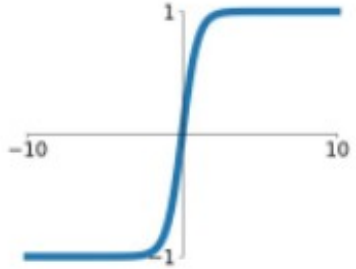
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



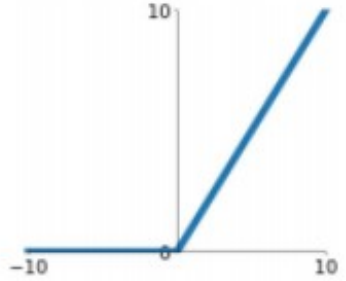
tanh

$$\tanh(x)$$

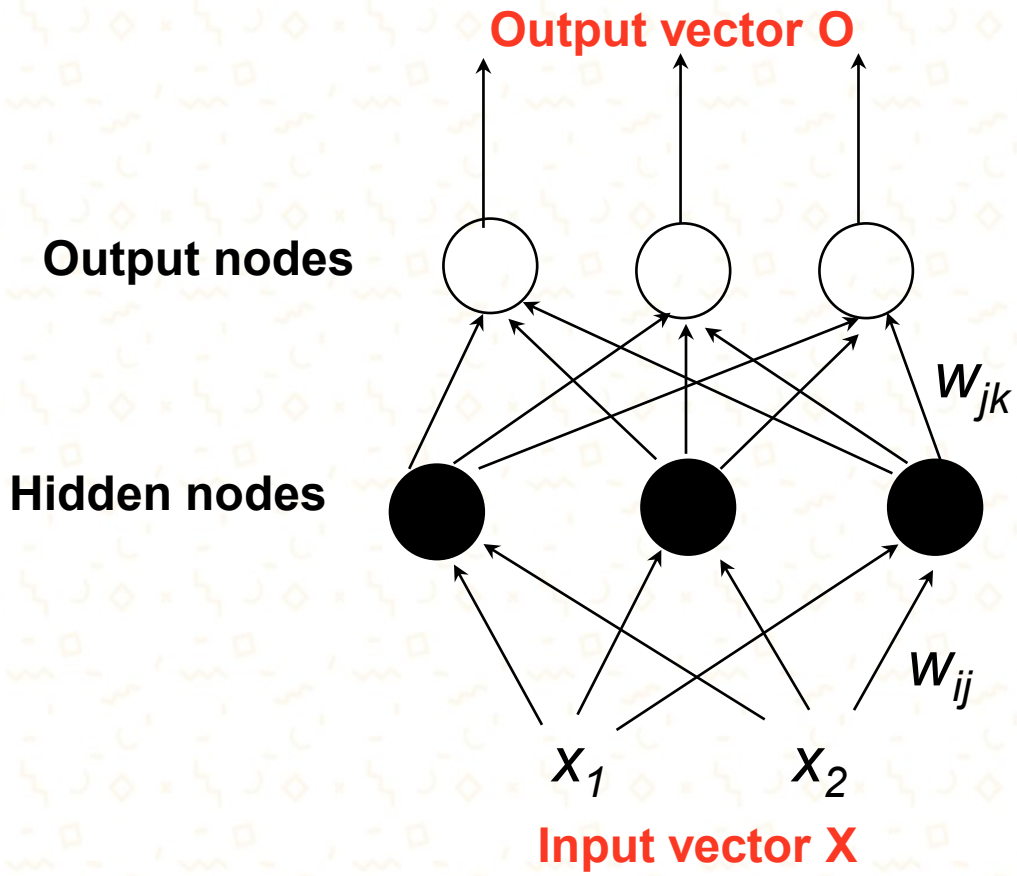


ReLU

$$\max(0, x)$$



Multilayer perceptron



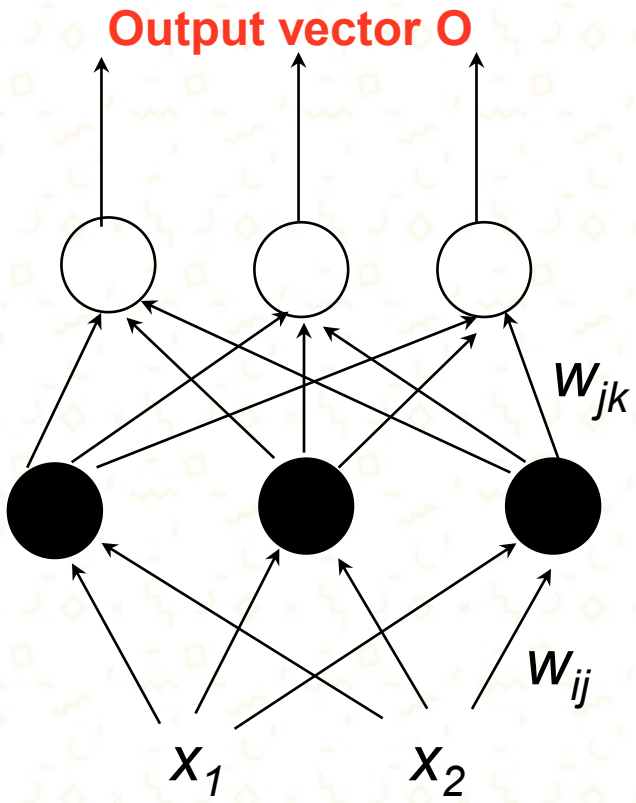
$$O_k = \frac{1}{1+e^{-\sum h_j w_{jk} + \theta_k}}$$

$$h_j = \frac{1}{1+e^{-\sum x_i w_{ij} + \theta_j}}$$

Network Training

- Tujuan dari training
  - Mendapatkan weights yang membuat hampir semua sampel dalam training data dapat diklasifikasikan dengan benar
- Langkah-langkah
  - Inisialisasi weights w_{ij} dengan nilai acak(random)
  - Masukkan (feed) sample training ***X*** ke dalam jaringan satu persatu
  - Untuk setiap unit
    - Hitung output value ***O*** dengan mengaplikasikan activation function
    - Hitung error ***E***
    - Update weights w_{ij} dan biases

Backpropagation Learning



$$Err_k = (T_k - O_k)$$

O_k

h_j

Given (X, T)

$$\delta_k = O_k(1-O_k)(T_k - O_k)$$

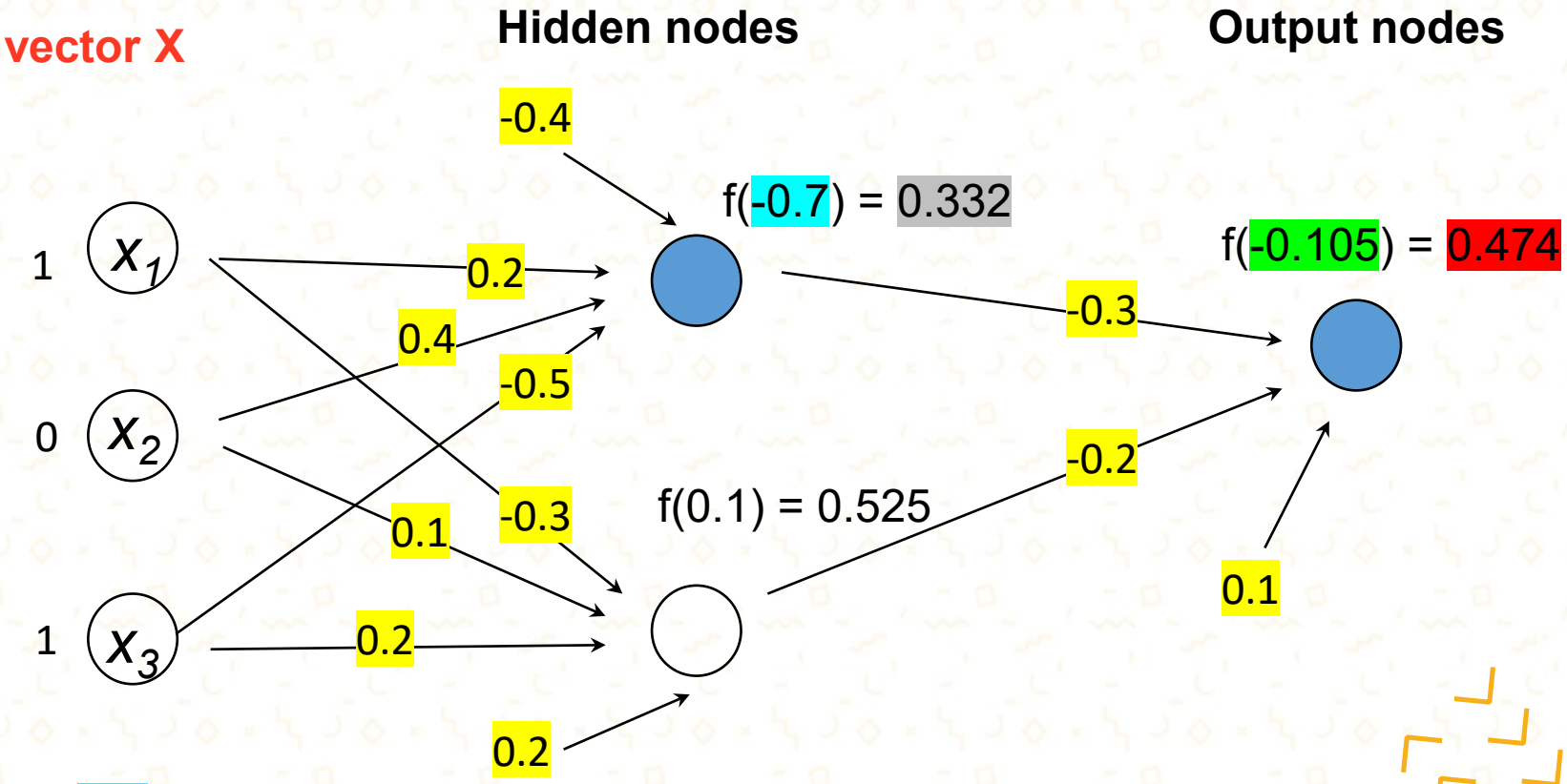
$$w_{jk} = w_{jk} + (l)\delta_k O_k$$

$$\delta_j = h_j(1-h_j)\sum_k \delta_k w_{jk}$$

$$w_{ij} = w_{ij} + (l)\delta_j h_j$$

Contoh

X_1	X_2	X_3	Y
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

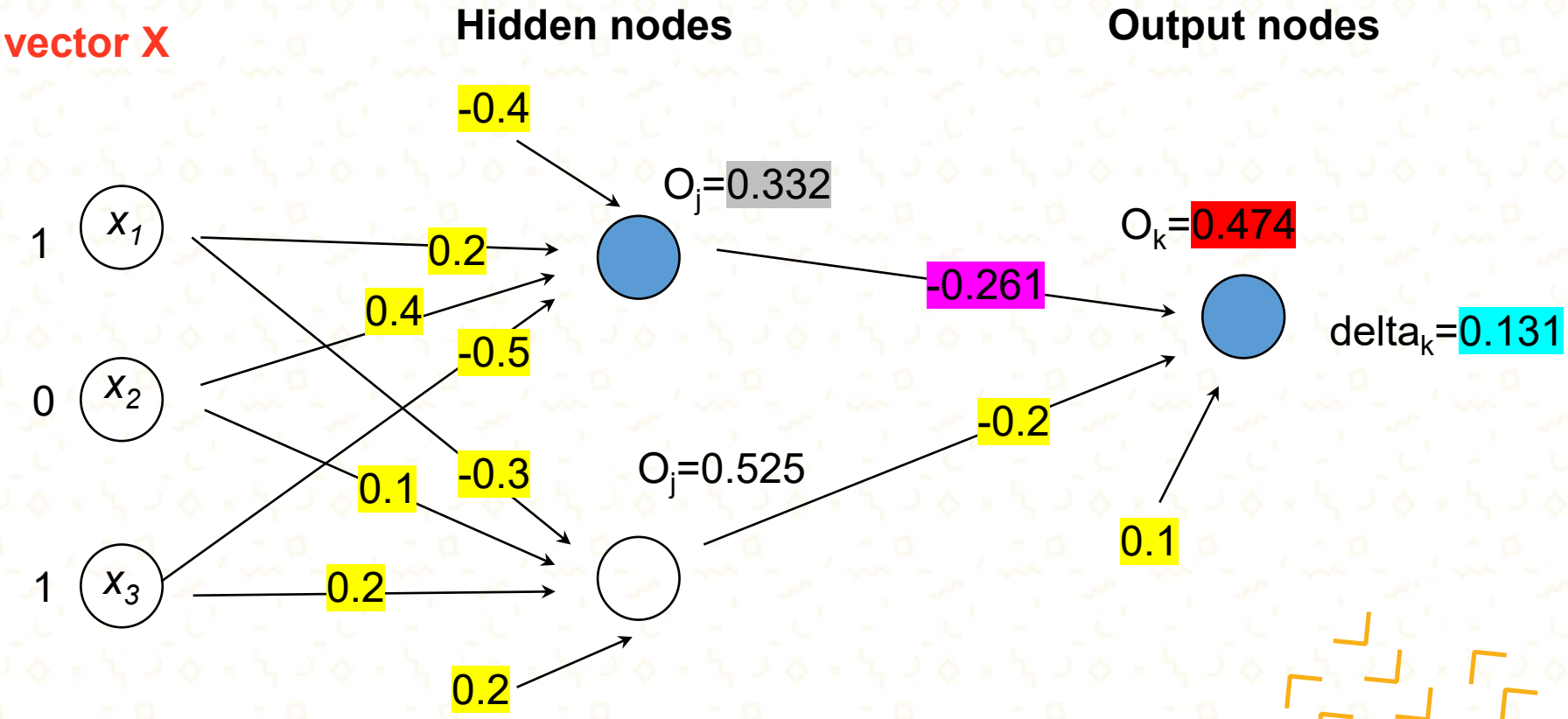


- Net Input I_j = 0.2 + 0 + -0.5 -0.4 = -0.7
- Output O_j = $(1/(1+e^{0.7})$=0.332

- Net Input I_k = -0.3(0.332) - (0.2)(0.525) + 0.1 = -0.105
- Output O_k = $(1/(1+e^{0.105})$ = 0.474

Contoh

X_1	X_2	X_3	Y
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



- $delta_k$=(0.474)(1-0.474)(1-0.474)=0.131
- w_{jk} = -0.3+(0.9)(0.1311)(0.332)=-0.261

- Dan seterusnya..

Discussion on NN

- Keuntungan
  - **Robust** -berfungsi baik ketika training set mengandung error
  - Output bisa discrete, real-valued, atau vector
- Criticism
  - Waktu yang lama saat training
  - Sulit untuk dipahami
  - Tidak mudah memasukkan *domain knowledge*

MLP in Sklearn

- Sklearn menyediakan class MLPClassifier

```
class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100), activation='relu', *, solver='adam', alpha=0.0001,
batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None,
tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)   [source]
```

Hyperparameter	Description
hidden_layer_sizes	Jumlah hidden layer dan node-nya
activation	Activation function
max_iter	Jumlah iterasi

How to use it?

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
#scaling
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

model=MLPClassifier(max_iter=1000, random_state=42, activation='logistic')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('Accuracy ',accuracy_score(y_test, y_pred))
print('Precision ',precision_score(y_test, y_pred, average='macro'))
print('Recall ',recall_score(y_test, y_pred, average='macro'))
print('Confusion matrix ', confusion_matrix(y_test, y_pred))
plot_confusion_matrix(model, X_test, y_test, cmap=plt.cm.Blues)
plt.show()
```

DigitalSkola

Uncover The World Of Digital Skills With Us

Thank YOU