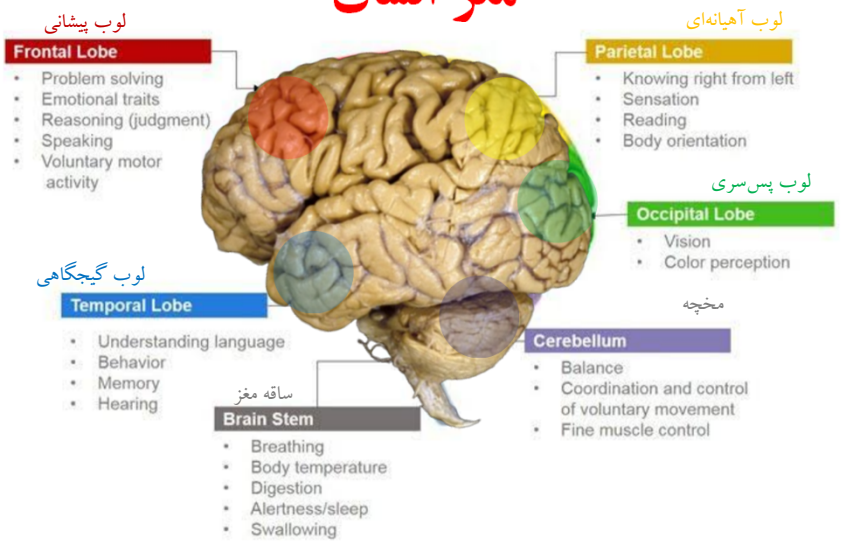
Advanced Application of Artificial Intelligence and Digital Transformation

K.N. Toosi University of Technology

Hasan Ghasemzadeh
http://wp.kntu.ac.ir/ghasemzadeh

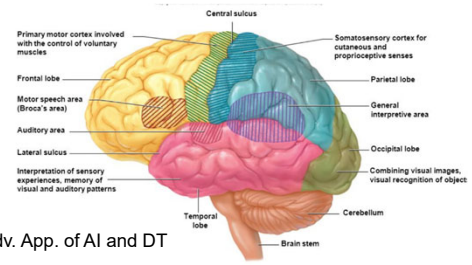
Neural Networks شبکه عصبی

مغز انسان

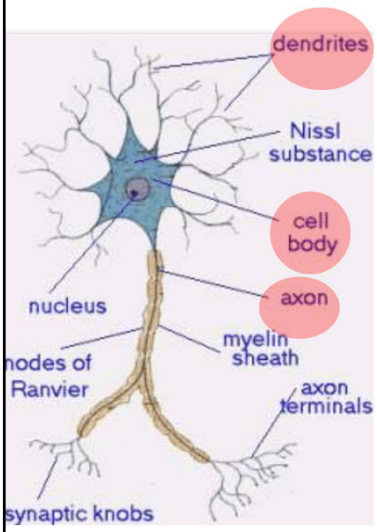


برخی ویژگیهای مغز

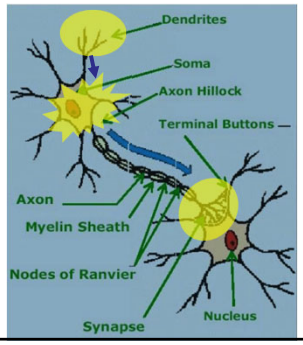
- یک توده بافت عصبی(مغز انسان – حدود ۱/۴ کیلوگرم و ۱/۵ لیتر)
- هدایت کل جسم بصورت خودآگاه و ناخودآگاه
- مرکز اندوختن تجربه، یادگیری و تصمیم گیری
- حجم حافظه بالا و سرعت پردازش زیاد
- نامشخص بودن نحوه عملکرد مغز(تاکنون)



برخی ویژگیهای مغز



- مغز شامل ۸۶ میلیارد نورون (۱۹ درصد در قشر حدود ۳ میلیمتری مغز و ۸۰ درصد در مخچه)
- نورون‌ها از یک جسم سلولی، دندریت و آکسون تشکیل شده‌اند.
- نورون اطلاعات را از طریق سیگنال‌های الکتریکی و شیمیایی منتقل می‌کند.
- نورون‌ها انرژی خود را از طریق یک شکاف کوچک به نام سیناپس(بیش از ۱۰۰ تریلیون سیناپس) به یکدیگر منتقل می‌کنند



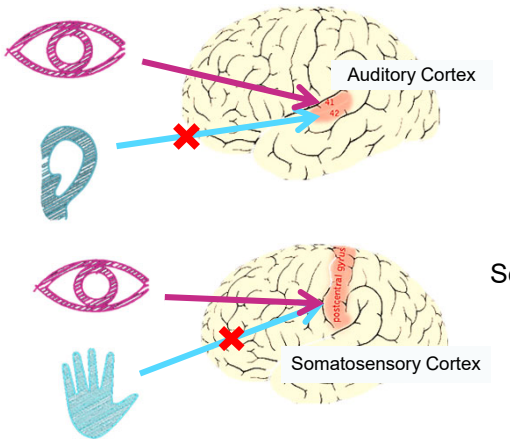
The soma, sums the incoming signals. When sufficient input is received, the cell fires; that is it transmit a signal over its axon to other cells

5

برخی ویژگیهای مغز

The “one learning algorithm” hypothesis

گرچه هر قسمت مغز وظیفه مخصوص خود را انجام می دهد ولی در شرایط خاص محرومیت از برخی سنسورها سعی بر جابجایی وظایف خود میکند



Auditory cortex learns to see

[Roe et al., 1992]

Somatosensory cortex learns to see

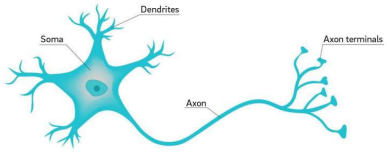
[Metin & Frost, 1989]

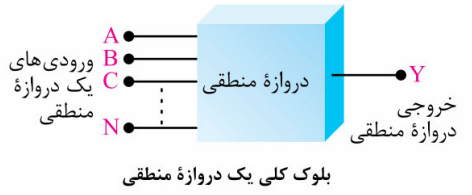
Adv. App. of AI and DT 6

مقایسه مغز انسان و رایانه

Feature	Human (Brain)	Computer
Processing Elements	~86 billion neurons	100 billion in modern chips (like Apple M2).
Interconnects	~10,000 synaptic connections per neuron	a few interconnects per transistor.
Cycles per Second	~200 Hz to 1000 Hz (neuron firing rate)	Up to 5 GHz
Memory	Dynamic and associative (estimated ~2.5 PB)	Explicit, structured (up to terabytes).
2X Improvement	Evolved over millions of years	~1.5–2 years

Neuron

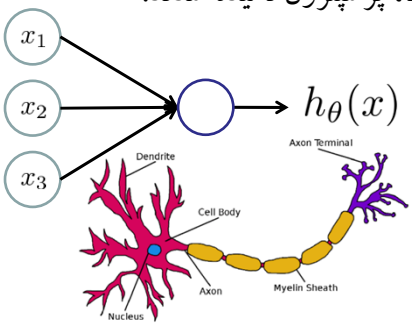




بلوک کلی یک دروازهٔ منطقی

پرسپترون

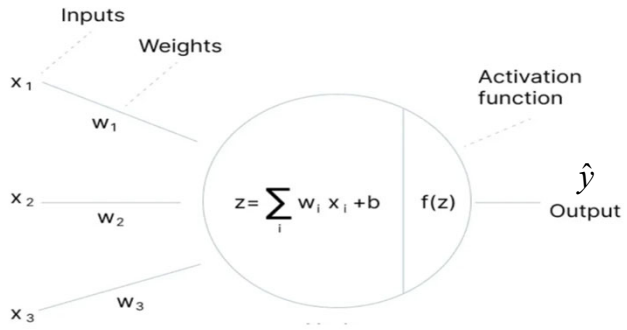
- شبکه عصبی مصنوعی از رفتار نورنهای مغز تقلید می کند.
- یک واحد منطقی مدلی از یک نورن است. Neuron model: Logistic unit
- شبکه‌های تک لایه، با توابع فعال سازی آستانه‌ای، توسط روزنبلات (۱۹۶۲) بنیان‌گذاری شدند که این نوع شبکه‌ها، پرسپترون نامیده شدند.



$$x = \begin{Bmatrix} x_0 \\ x_1 \\ x_2 \\ x_n \end{Bmatrix} \quad w = \begin{Bmatrix} w_0 \\ w_1 \\ w_2 \\ w_n \end{Bmatrix}$$

Sigmoid (logistic) activation function

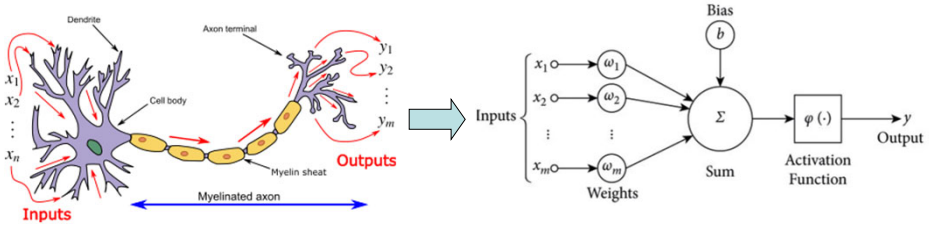
مدل ریاضی یک نرون (مک کلاخ–پیتس–۱۹۴۳)



Inputs | Weights | Summation | Activation | Output

پرسپترون

- From biological neuron to artificial neuron (perceptron)



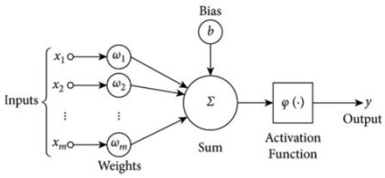
$$z = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$\hat{y} = activation(z) \quad \begin{cases} if \ z \geq 0 \ , \ \hat{y} = 1 \\ if \ z < 0 \ , \ \hat{y} = 0 \end{cases}$$

$$Error = \Delta y = y - \hat{y}$$

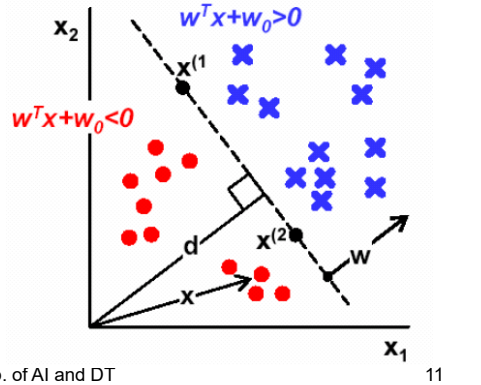
پرسپترون

- Activation function



$$z = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$\begin{cases} if\ z \geq 0\ , & \hat{y} = 1 \\ if\ z < 0\ , & \hat{y} = 0 \end{cases}$$



$$w_i + \eta.\Delta y.x_i \rightarrow w_i$$

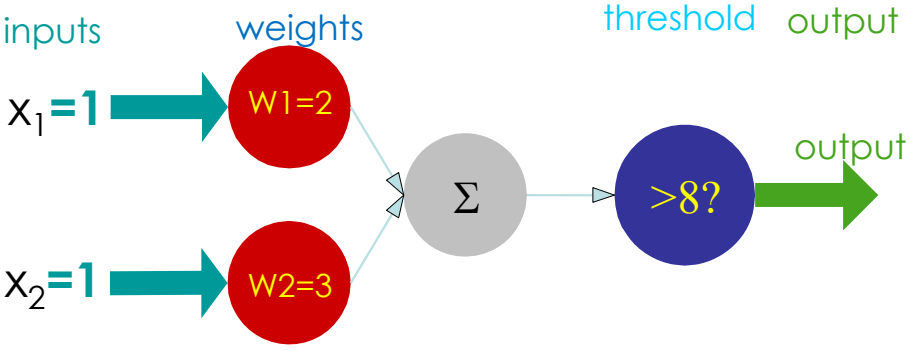
x_i Input feature value
w_i Activation function
Δy Error
η Learning rate

Adv. App. of AI and DT 11

مثال پرسپترون: Logical AND

	x1	x2	T
P1 =	0	1	0
P2 =	1	1	1
Pn =	...	...	...

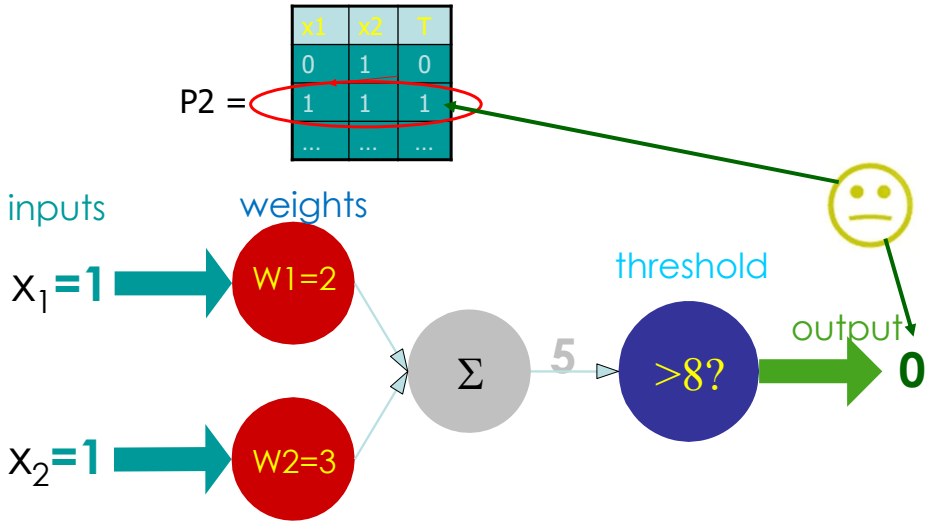
P2 الگویی است که باید پرسپترون یاد بگیرد:
یعنی، اگر هر دو ورودی ۱ هستند، خروجی با ید ۱ باشد



۱۲

Adv. App. of AI and DT

ادامه مثال – آموزش



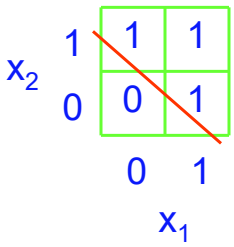
۱۳

Adv. App. of AI and DT

Linear Separability

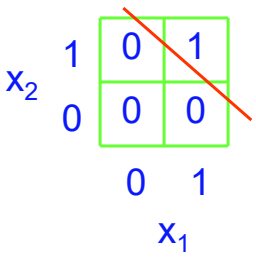
•Examples (two dimensions):

OR function



linearly separable

AND function

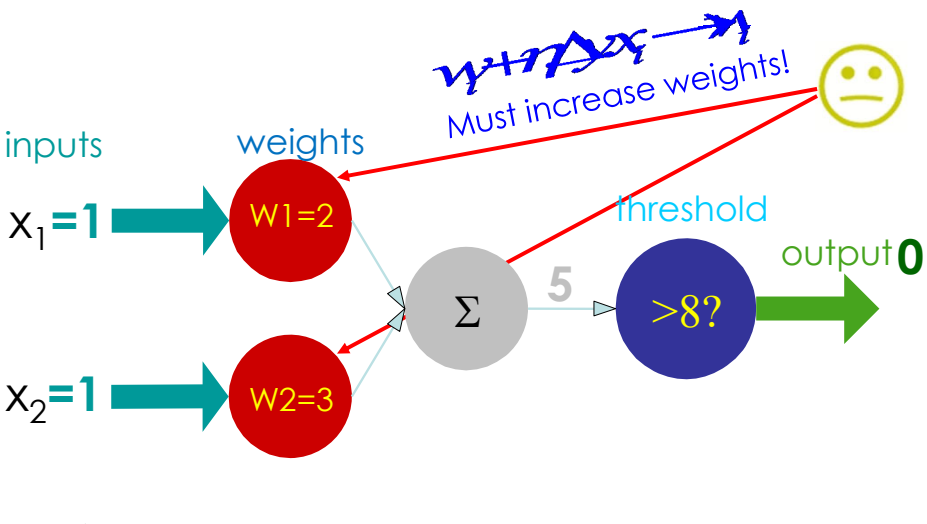


linearly separable

This means that a single threshold neuron can realize the OR and AND function.

Adv. App. of AI and DT

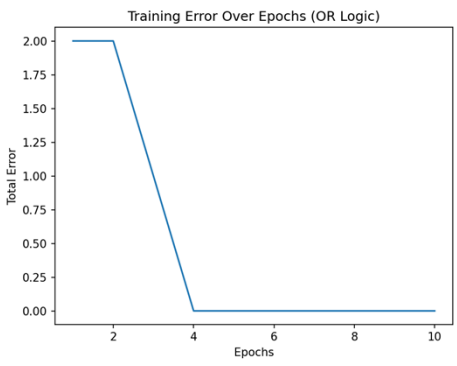
ادامه مثال – آموزش

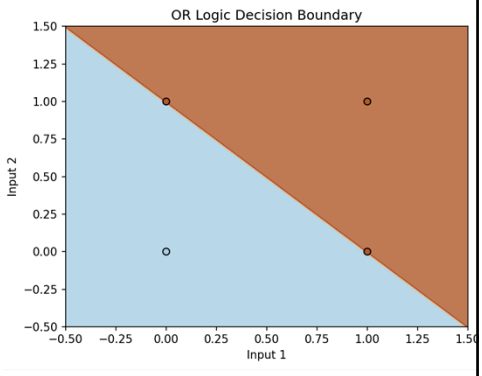


۱۵

Adv. App. of AI and DT

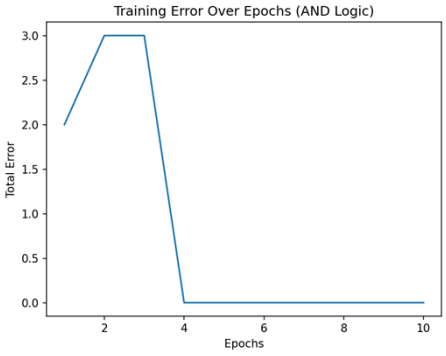
مثال پرسپترون: Logical OR

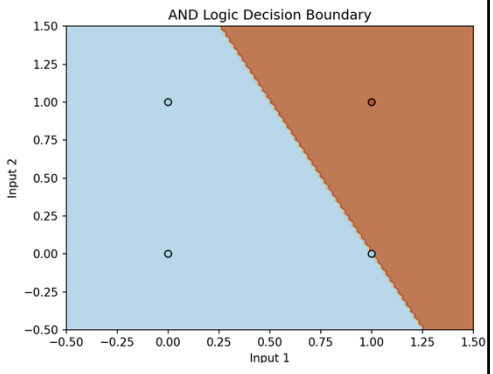




Adv. App. of AI and DT 16

مثال پرسپترون: Logical AND





Adv. App. of AI and DT 17

Linear Inseparable

•Examples (two dimensions):

XOR function

x_2		x_1 = 0	x_1 = 1
	1	1	0
	0	0	1

linearly inseparable

XNOR function

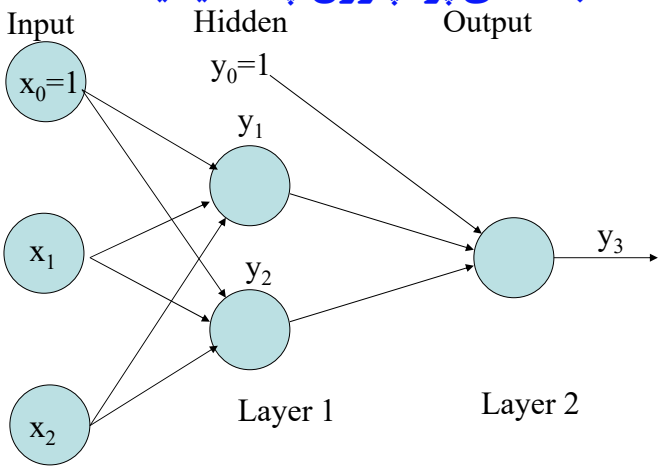
x_2		x_1 = 0	x_1 = 1
	1	0	1
	0	1	0

linearly inseparable

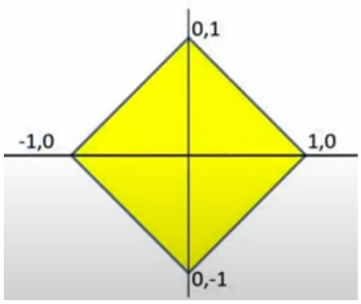
This means that a single threshold neuron cannot realize the XOR and XNOR function.

Adv. App. of AI and DT

شبکه های پرسپترون چند لایه یا MLP

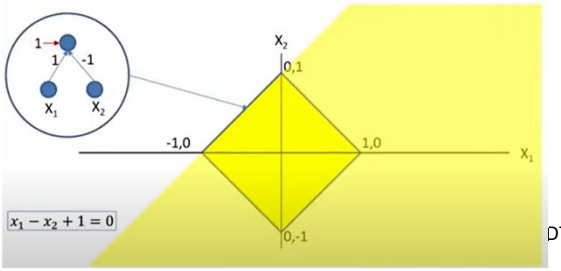


عملکرد پرسپترون چند لایه یا MLP



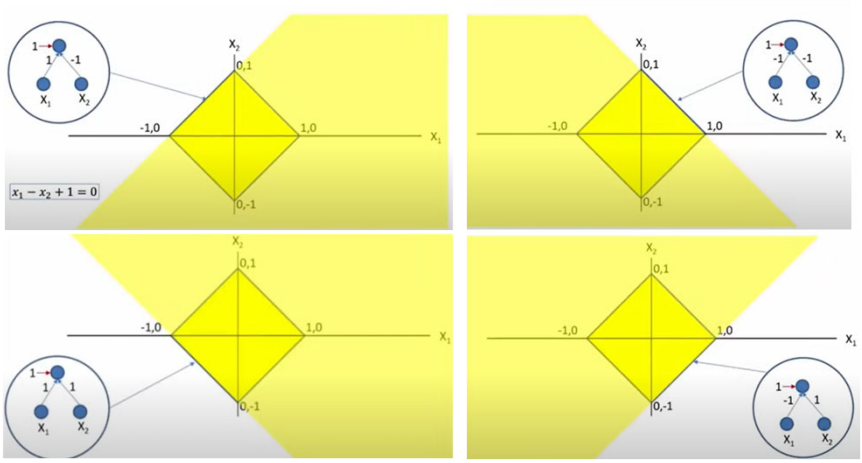
مثال:
چگونه می توان با پرسپترون نقاط داخل مربع را از سایر نقاط جدا کرد

حل:
به دلیل غیرخطی بودن کلاس وسط مربع با یک پرسپترون نمی‌توان طبقه‌بندی انجام داد



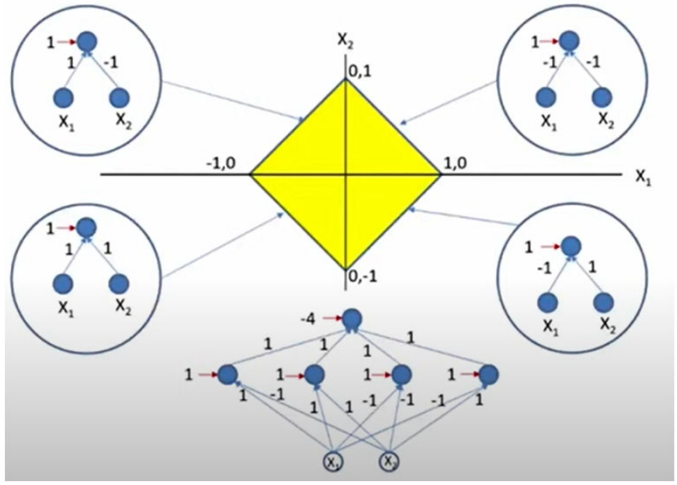
طبقه بندی با یک پرسپترون فقط خطی است

عملکرد پرسپترون چند لایه یا MLP

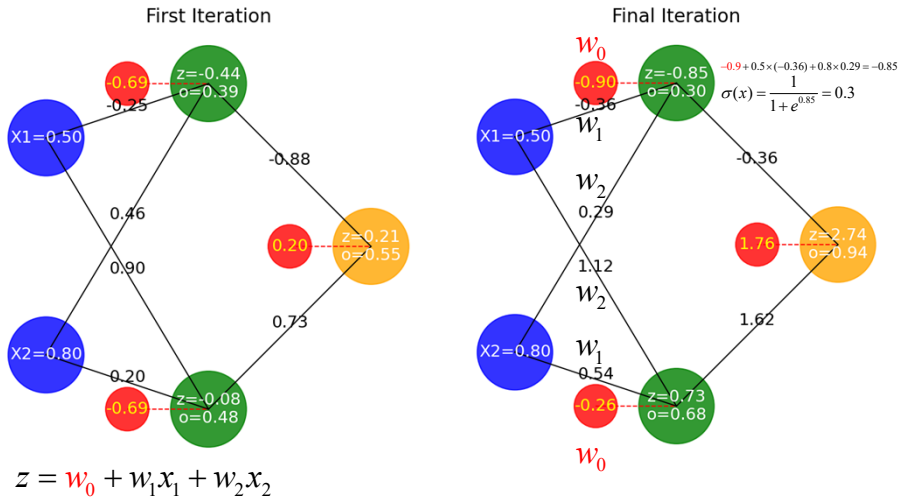


چهار طبقه بندی با یک پرسپترون می توان جواب مساله باشد

عملکرد پرسپترون چند لایه یا MLP



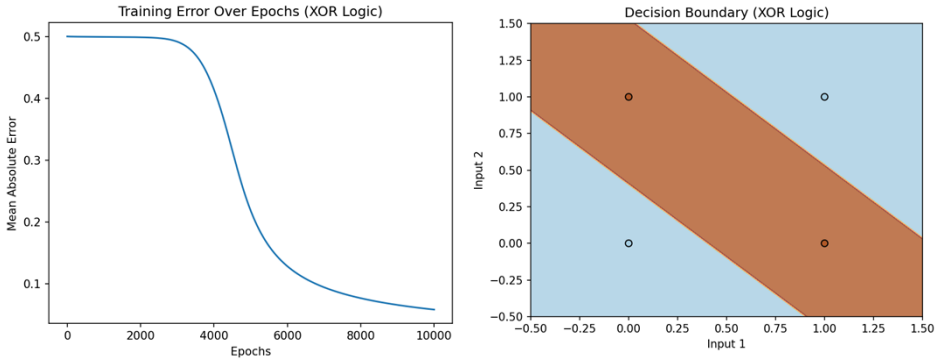
مثال پرسپترون چندلایه: Logical XOR



$$z = w_0 + w_1 x_1 + w_2 x_2$$

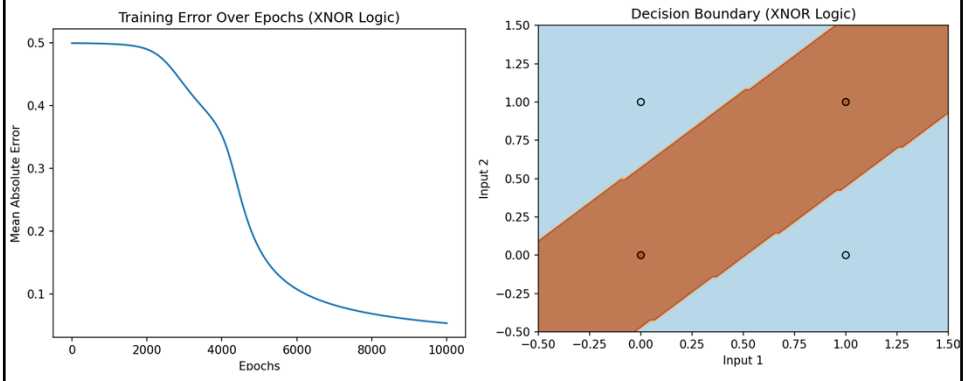
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

مثال پرسپترون چندلایه: Logical XOR



از روشهای مختلف از جمله گرادیان کاهشی برای تصحیح ضرایب وزنه استفاده می‌شود

مثال پرسپترون چندلایه : Logical XNOR

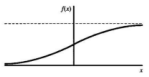


پرسپترون چند لایه، شبکه عصبی یا شبکه ژرف نیز نامیده می شود

شبکه عصبی – ویژگی های تابع فعال سازی

- **ویژگی‌های مورد نیاز**
  - **پیوسته**
  - **مشتق‌پذیر**
    - دارا بودن کارایی محاسباتی (به راحتی قابل محاسبه باشد)
    - مشتق تابع را بتوان برحسب مقدار خود تابع نوشت
  - **به صورت یکنوا غیرنزولی**
  - **قابلیت اشباع (Saturate)**
    - به صورت مجانبی به مقادیر بیشینه و کمینه خود نزدیک شود

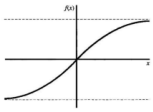
- **سیگموید دودویی**



$$f_1(x) = \frac{1}{1+\exp(-x)}$$

$$f_1'(x) = f_1(x)[1 - f_1(x)]$$

- **سیگموید دوقطبی**
  - شباهت به تانژانت هیپربولیک

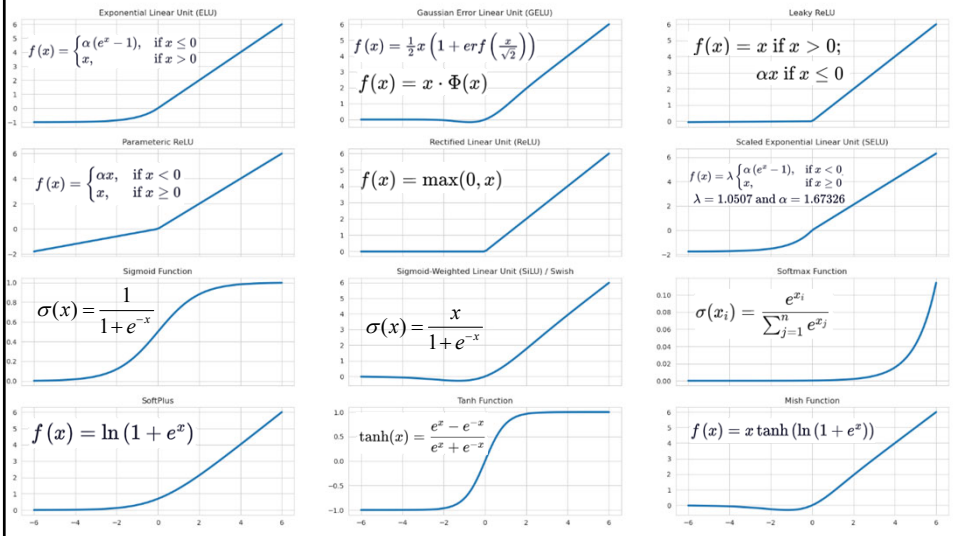


$$f_2(x) = \frac{2}{1+\exp(-x)} - 1$$

$$f_2'(x) = f_1(x) = \frac{1}{1+\exp(-x)}$$

با کشف الگوریتم پس-انتشار توسط رملهات، هینتن و ویلیامز در سال ۱۹۸۶ مطالعات جدید بر روی شبکه‌های عصبی مجددا شروع شد. اهمیت ویژه این الگوریتم این بود که شبکه‌های عصبی چند لایه توسط آن می‌توانستند آموزش داده شوند.

Comparison of Activation Functions

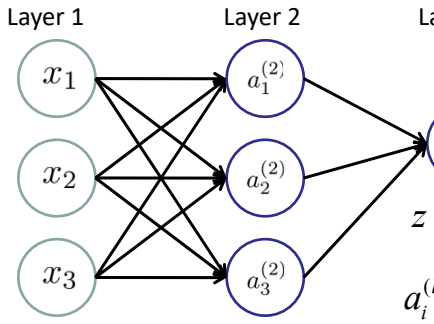


Comparison of Activation Functions

Activation Function		Range	Use Case	Advantages	Limitations
$\sigma(x) = \frac{1}{1+e^{-x}}$	Sigmoid	(0, 1)	Binary classification	Smooth, probabilistic output	Vanishing gradient, not zero-centered
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Tanh	(-1, 1)	Hidden layers	Zero-centered output	Vanishing gradient
$f(x) = \max(0, x)$	ReLU (Rectified Linear Unit)	[0, ∞)	Hidden layers	Efficient, avoids vanishing gradient	Dying neurons
$f(x) = x \text{ if } x > 0;$ $\alpha x \text{ if } x \le 0$	Leaky ReLU	(-∞, ∞)	Hidden layers	Avoids dying neurons	Arbitrary slope choice
$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$	Softmax	(0, 1)	Output for multi-class problems	Probabilistic interpretation	Expensive for large datasets
$f(x) = x \cdot \sigma(x)$	Swish	(-∞, ∞)	Deep networks	Smooth and flexible	Slightly more computationally intensive
$f(x) = x \cdot \Phi(x)$	GELU	(-∞, ∞)	Transformer-based architectures	Smooth and efficient	Relatively new, computationally expensive

شبکه عصبی مصنوعی

افزایش تعداد لایه‌های برای پردازش مسایل پیچیده‌تر



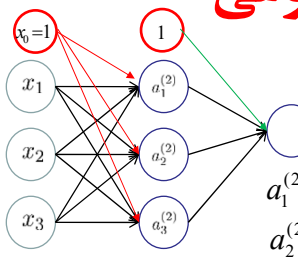
$$z = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$a_i^{(l+1)}(x) = b_i^{(l)} + \sum_j w_{ji}^{(l)} a_i^{(l)}$$

$a_i^{(l+1)}$ activation of unit i in layer $l+1$

$w_{ji}^{(l)}$ Weight of unit i in layer $l+1$ receiving from unit j in layer l which control function mapping from layer l to layer $l+1$

شبکه عصبی مصنوعی



$$a_1^{(2)}(x) = g(b_1^{(1)} x_0^{(1)} + w_{11}^{(1)} x_1^{(1)} + w_{21}^{(1)} x_2^{(1)} + w_{31}^{(1)} x_3^{(1)})$$

$$a_2^{(2)}(x) = g(b_2^{(1)} x_0^{(1)} + w_{12}^{(1)} x_1^{(1)} + w_{22}^{(1)} x_2^{(1)} + w_{32}^{(1)} x_3^{(1)})$$

$$a_3^{(2)}(x) = g(b_3^{(1)} x_0^{(1)} + w_{13}^{(1)} x_1^{(1)} + w_{23}^{(1)} x_2^{(1)} + w_{33}^{(1)} x_3^{(1)})$$

$$h_w(x) = a_1^{(3)}(x) = g(b_1^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{21}^{(2)} a_2^{(2)} + w_{31}^{(2)} a_3^{(2)})$$

Learnable parameter for l layer

$$\sum_1^{l-1} (N_L + 1)(N_{L+1}) = (N_1 + 1)N_2 + (N_2 + 1)N_1 \ldots (N_{L-1} + 1)N_l$$

N_l Number of neuron in l layer

شبکه عصبی مصنوعی

pip install tensorflow

pip install intel-tensorflow Use **Intel-optimized TensorFlow**

```
import os
import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay

# Optional: Disable oneDNN custom operations to avoid floating-point warnings
os.environ["TF_ENABLE_ONEDNN_OPTS"] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' #Suppress TensorFlow INFO and WARNING logs

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input

# Set random seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)
```

13 ANN.py

شبکه عصبی مصنوعی

```
# Generate a 2D dataset for visualization
X, y = make_classification(
    n_samples=200, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=42
)

# Standardize the dataset
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Build the ANN model
def build_model(input_dim, output_dim):
    """
    Build a simple feedforward ANN model with two hidden layers.
    """
    model = Sequential([
        Input(shape=(input_dim,)),  # Explicit Input layer
        Dense(4, activation='relu'),  # 4 neuron First hidden layer
        Dense(2, activation='relu'),  # 2 neuron Second hidden layer

        Dense(output_dim, activation='sigmoid')  # Output layer for
binary classification
    ])
    return model
```

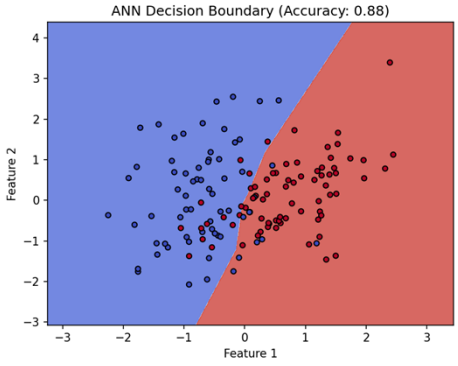
شبکه عصبی مصنوعی

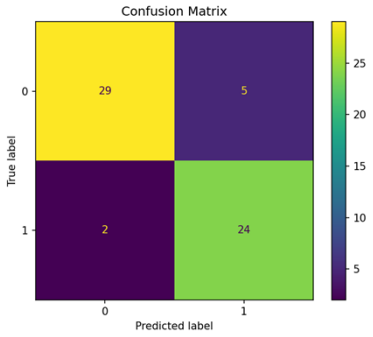
```
# Train, evaluate, and plot ANN
def train_evaluate_ann(epochs=50, batch_size=8):
    """
    Train the ANN, evaluate on test data, and visualize results.
    """
    print("Training Artificial Neural Network...")

    # Build and compile the model
    model = build_model(input_dim=2, output_dim=1)
    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

    # Train the model
    history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, verbose=0)

    # Evaluate on test set
    y_pred = (model.predict(X_test) > 0.5).astype(int)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Test Accuracy: {accuracy:.2f}")

    # Plot decision boundary
    plot_ann_with_boundary(X_train, y_train, model, accuracy)

# Run the ANN training and visualization
if __name__ == "__main__":
    try:
        train_evaluate_ann(epochs=50, batch_size=8)
    except Exception as e:
        print(f"An error occurred: {e}")
# Predict a new example
    predict_new_example(model, scaler, target_names)
```


مثال

حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

- اجرای اول



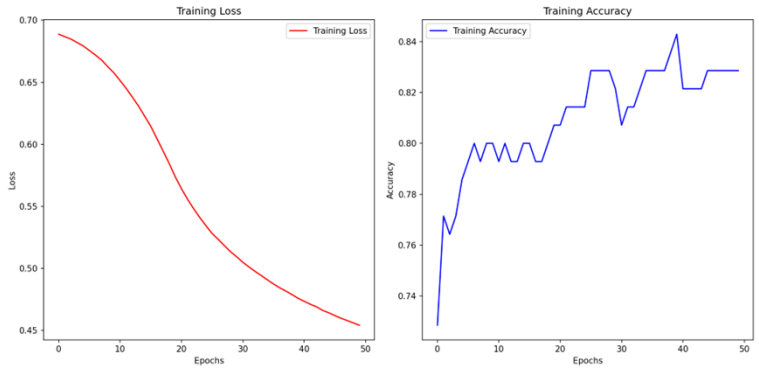



```
model = Sequential([
    Input(shape=(input_dim,)),  # Explicit Input layer
    Dense(4, activation='relu'),  # 4 neuron First hidden layer
    Dense(2, activation='relu'),  # 2 neuron Second hidden layer
    Dense(output_dim, activation='sigmoid')  # Output layer for binary classification
```

مثال

حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

- اجرای اول

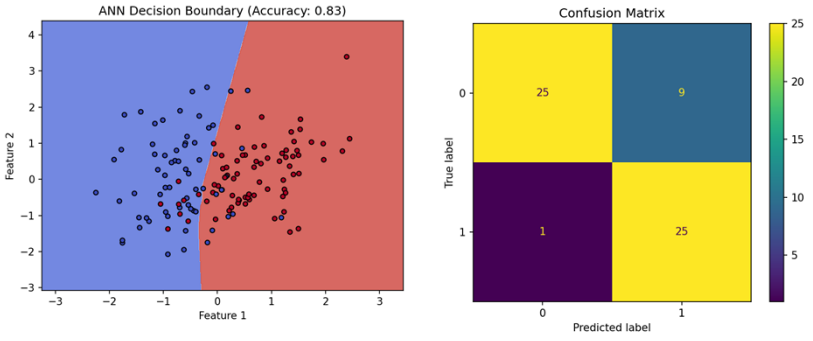


Adv. App. of AI and DT 35

مثال

حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

- اجرای دوم - حساسیت به شرایط اولیه



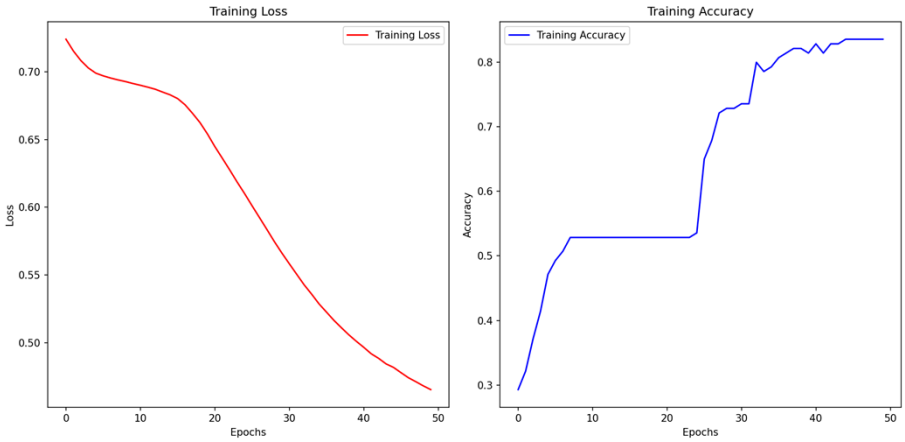

```
model = Sequential([
    Input(shape=(input_dim,)),  # Explicit Input layer
    Dense(4, activation='relu'),  # 4 neuron First hidden layer
    Dense(2, activation='relu'),  # 2 neuron Second hidden layer
    Dense(output_dim, activation='sigmoid')  # Output layer for binary classification
```

Adv. App. of AI and DT 36

مثال

حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

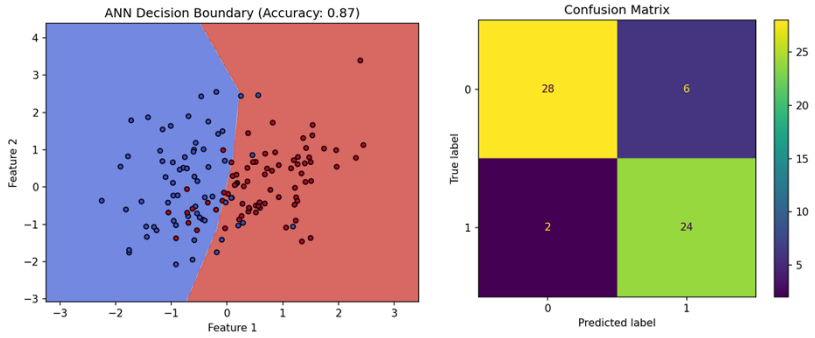
- اجرای دوم



مثال

حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

- اجرای سوم – شرایط اولیه یکسان



Set Random Seeds:

•Fixing the random seed ensures reproducibility across runs.

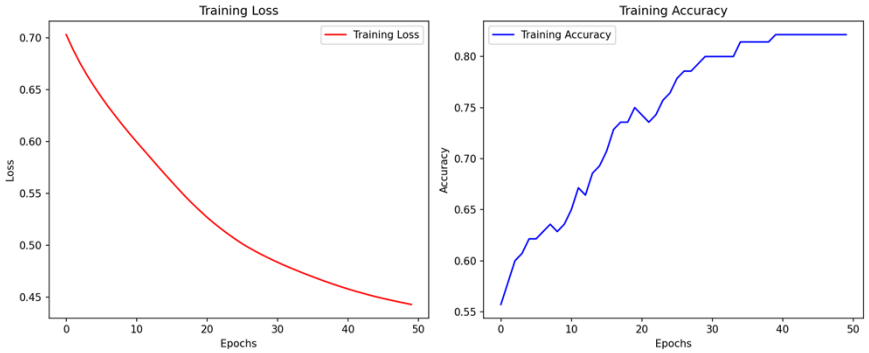
```
np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)
```

13 ANN replace SVM.py

مثال

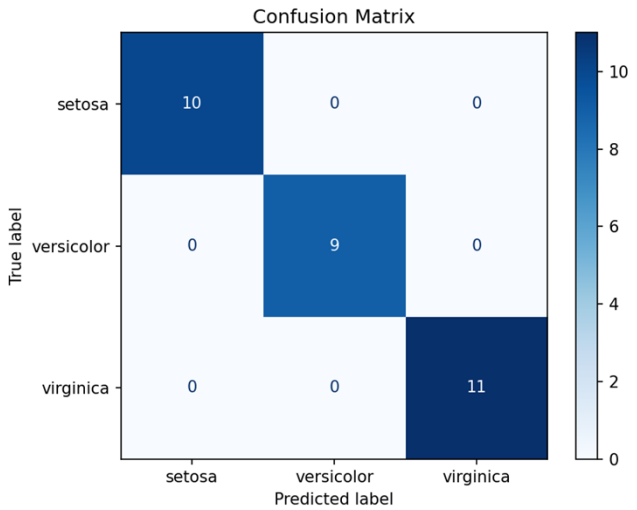
حل مساله به روش شبکه عصبی برای مثال دسته بندی در قسمت SVM

- اجرای سوم



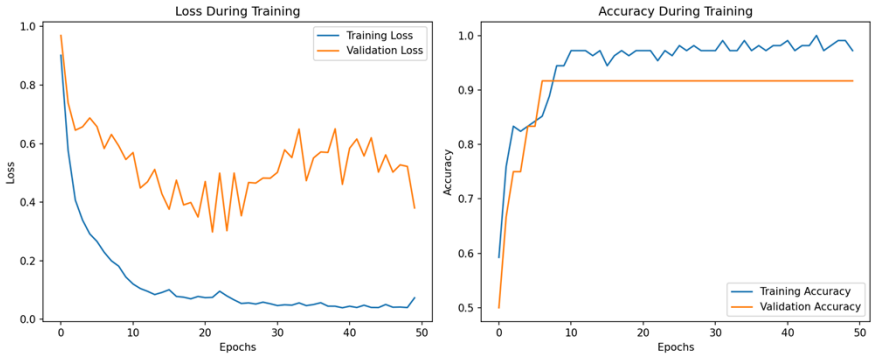
مثال

حل مساله گلهای زنبق



مثال

حل مساله گلهای زنبق



New Example Features: [5.9 3. 5.1 1.8]
Predicted Class: virginica

معماری شبکه هوش مصنوعی

•First hidden layer, the number of neurons is often: $n > n_{input\ features} + n_{output\ targets}$

•Subsequent layers, try reducing the number of neurons progressively (e.g., 64 → 32 → 16).

Example

1. Binary Classification
•Input Features: 10
•Output Classes: 2

•Suggested Architecture:
- •Input Layer: 10 neurons (input size).
- •Hidden Layer 1: 16 neurons (capture interactions).
- •Hidden Layer 2: 8 neurons (reduce complexity).
- •Output Layer: 2 neuron (sigmoid activation).

2. Multiclass Classification
•Input Features: 20
•Output Classes: 5

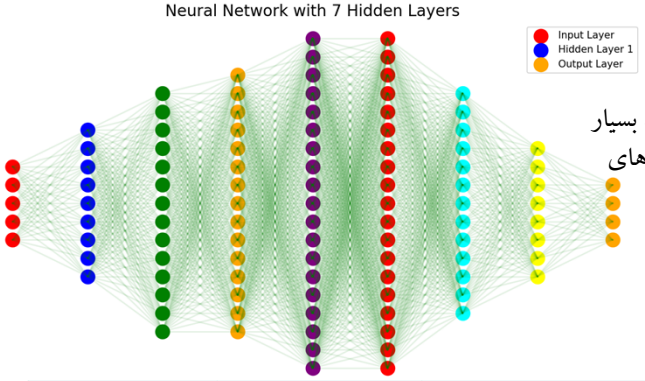
•Suggested Architecture:
- •Input Layer: 20 neurons.
- •Hidden Layer 1: 32 neurons (capture higher-level features).
- •Hidden Layer 2: 16 neurons.
- •Output Layer: 5 neurons (softmax activation).

3. Regression
•Input Features: 5
•Output: Continuous value

•Suggested Architecture:
- •Input Layer: 5 neurons.
- •Hidden Layer 1: 10 neurons (nonlinear mapping).
- •Output Layer: 1 neuron (linear activation).

Use learning curves to analyze underfitting (high training loss) or overfitting (low training loss, high validation loss)

شبکه عصبی مصنوعی



شبکه عصبی می‌تواند بسیار پیچیده شده و پارامترهای زیادی داشته باشد

Feature	GPT-3	GPT-4 (Speculated)
Number of Layers	96	Likely >120
Parameters	175 billion	500 billion to 1 trillion
Context Length	2,048 to 4,096 tokens	Up to 32,768 tokens (or more)
Training Data	570 GB of filtered text	Larger and more diverse dataset
Vector Dimension	12,288	Likely 16,384 or more

شبکه عصبی مصنوعی کانولوشنی

CNN: Convolutional Neural Networks

نوعی از **شبکه‌های عصبی مصنوعی** است که می‌تواند داده‌های تصویری را **به‌صورت خودکار تحلیل و یاد بگیرد**.
برخلاف شبکه‌های عصبی سنتی که نیاز به پیش‌پردازش دستی ویژگی‌ها دارند، CNN ویژگی‌های مهم تصویر را به‌صورت خودکار استخراج می‌کند

مؤلفه	توضیح
لایه کانولوشن Convolution Layer	فیلترهایی (Kernel) را روی تصویر حرکت می‌دهد تا ویژگی‌هایی مثل لبه، گوشه یا بافت را شناسایی کند.
لایه فعال‌سازی ReLU	عملیات غیرخطی (مثل ReLU: max(0, x)) برای افزایش توانایی مدل در یادگیری روابط پیچیده.
لایه زیرنمونه‌گیری Pooling	اندازه تصویر را کاهش می‌دهد و به مدل کمک می‌کند تا نسبت به جابه‌جایی‌ها مقاوم شود.
لایه اتصال کامل Fully Connected	پس از استخراج ویژگی‌ها، تصمیم‌گیری نهایی (مثلاً طبقه‌بندی) را انجام می‌دهد.
Softmax	خروجی نهایی برای دسته‌بندی (مثلاً احتمال تعلق تصویر به هر کلاس).

تفسیرپذیری شبکه عصبی مصنوعی

عدم تفسیرپذیری یک معضل مهم **شبکه‌های عصبی مصنوعی** است
روشهایی جهت تفسیر پذیری شبکه عصبی مصنوعی:
امروزه تمرکز از روش‌های کلاسیک مانند
SHAP و
LIME
به سمت روش‌های مبتنی بر مفهوم،
مدارهای داخلی شبکه و
مدل‌های علّی حرکت کرده است.

SHAP – روش توضیحات جمع‌پذیر شپلی (Shapley Additive Explanations)

لوید شپلی(Lloyd Shapley)، ریاضیدان و اقتصاددان برنده جایزه نوبل در سال ۲۰۱۷ مقدار شپلی (Shapley Value) را در تئوری بازی همکارانه تعریف کرد که بیانگر نقش هر بازیکن در موفقیت کلی تیم است.

روش SHAP در سال ۲۰۱۷ در دانشگاه واشنگتن معرفی شد. در یادگیری ماشین این روش با هر ویژگی مثل یک بازیکن رفتار می کند و نقش هر ویژگی در نتیجه نهایی را مشخص می کند.

تفسیرپذیری شبکه عصبی مصنوعی

LIME -تفسیرپذیر محلی مستقل از مدل(Local Interpretable Model-agnostic Explanations)

LIME این مدل بر روی یک پیش‌بینی واحد تمرکز می‌کند و می‌پرسد: «چرا مدل این پیش‌بینی خاص را برای این ورودی خاص انجام داده است؟ با استفاده از یک مدل هوش مصنوعی تفسیرپذیر مثل رگرسیون یا درخت تصمیم این کار انجام می شود.
روش کار
یک نمونه ورودی درنظر گرفته می شود مثل
[Soil type = clay, Depth = 10m, Water table = 2m, Load = 200kPa]
نمونه هایی تصادفی نزدیک نمونه ساخته می شود مثل
[Soil type = clay, Depth = 10.5m, Water table = 2.2m, Load = 200kPa]
پیش بینی توسط مدل اصلی صورت می پذیرد
به ورودی های با شباهت بیشتر به نمونه اولیه وزن بیشتر داده می شود
یک مدل تفسیرپذیر بر روی این داده ها آموزش داده می شود
ضرایب این مدل ساده (معمولا خطی) تاثیر هر پارامتر را مشخص می کند مثلا تاثیر عمق آب زیر زمینی روی نشست مشخص می شود

تفسیرپذیری شبکه عصبی مصنوعی

PINN – شبکه‌های عصبی فیزیک آگاه (Physics-Informed Neural Network)

در PINN، قوانین فیزیک حاکم بر مسئله (مانند معادلات تعادل، بقای جرم، یا قوانین ترمودینامیک) به عنوان جملات تابع هزینه (Loss Function) به شبکه عصبی اضافه می‌شوند. شبکه ملزم است هم خطای پیش‌بینی داده‌ها را کم کند و هم قوانین فیزیک را نقض نکند

برای مثال در ژئوتکنیک با این روش رفتار الاستوپلاستیک خاک را مدل‌سازی کرده‌اند. آنها قوانین افزایشی کرنش (تجزیه کرنش به الاستیک و پلاستیک) را در تابع هزینه شبکه قید کرده‌اند. نتیجه، مدلی است که تنش، کرنش پلاستیک و نسبت تخلخل را به طور همزمان و با پایبندی به اصول مکانیک خاک پیش‌بینی می کند و بسیار فراتر از یک رابطه همبستگی ساده عمل می‌نماید.

Variant	Description
PINN	Basic version with PDE residuals
Deep Ritz	Uses energy minimization instead of PDE residual
VPINN	Variational PINN (uses weak form of PDE)
XPINN	Domain decomposition for complex geometries
PPINN	Parallel PINN for large-scale problems
PINN with hard constraints	Enforces BC/IC exactly via basis functions

47
Adv. App. of AI and DT

تفسیرپذیری شبکه عصبی مصنوعی

Causal Explainability

به جای همبستگی، علت را پیدا می کند.
پرسش: اگر این ویژگی حذف شود چه اتفاقی می‌افتد؟

ابزارها

- Structural Causal Models
- Counterfactual Explanations
- Causal Mediation Analysis

مثال در مدل ترک‌یابی: آیا ترک باعث تصمیم شبکه شده یا فقط رنگ تیره تصویر؟

Counterfactual Explanations

مثال مدل می گوید: این قطعه معیوب است.
سپس توضیح می‌دهد:
اگر طول ترک ۲۰٪ کمتر بود، قطعه سالم تشخیص داده می‌شد.

48
Adv. App. of AI and DT

تفسیرپذیری شبکه عصبی مصنوعی

Prototype-Based Networks

شبکه تصمیم خود را با نمونه‌های واقعی توضیح می‌دهد.

مثلاً این تصویر ترک است زیرا شبیه نمونه شماره ۱۷ و ۳۴ در مجموعه آموزش است.

- ProtoPNet
- Case-Based Reasoning Networks

Explainable Vision Transformers

- Attention Rollout
- Attention Flow
- Transformer Attribution
- Token Attribution

برای مدل‌های جدید YOLO و ViT کاربرد دارند.

Physics-Aware Interpretability

در PINNها می‌توان تحلیل کرد کدام بخش شبکه قانون فیزیکی را یاد گرفته است؟

کدام قسمت داده را یاد گرفته است؟ کدام نواحی PDE را نقض می‌کنند؟

- PDE Attribution Maps
- Physics Residual Analysis
- Physics-Guided Attention

49

Adv. App. of AI and DT

تفسیرپذیری شبکه عصبی مصنوعی

Explainable Graph Neural Networks (XGNN)

برای شبکه‌های گرافی مناسب است

- GNNExplainer
- PGExplainer
- GraphMask

در تحلیل سازه‌ها، شبکه برق و سیستم‌های حمل‌ونقل کاربرد فراوان دارند.

پیشرفته‌ترین روندهای پژوهشی ۲۰۲۶

در حال حاضر بیشترین توجه جامعه تحقیقاتی روی این پنج حوزه است:

Mechanistic Interpretability
Sparse Autoencoders (SAE)
Concept Bottleneck Models
Causal Explainability
Counterfactual Explanations

50

Adv. App. of AI and DT

ANN vs PINN example

مثال: حل توزیع حرارت با استفاده از شبکه عصبی و شبکه عصبی فیزیک آگاه و مقایسه با جواب تحلیلی بدست آمده از سری فوریه

```
# -------------------- Problem Parameters -----------
L = 1.0               # length of rod (m)
T_max = 1.0           # total time (s)
alpha = 0.01          # thermal diffusivity (m²/s)
T_left = 100.0        # boundary temp at x=0 (°C)
T_right = 0.0         # boundary temp at x=1 (°C)
T_initial = 0.0       # initial temp (°C)
```

$$u_t = \alpha u_{xx} \quad or \quad \frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial t^2}$$

13 PINN vs ANN 1d heat transfer.py

ANN vs PINN example

13 PINN vs ANN 1d heat transfer.py

```
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim

# SETTINGS
torch.manual_seed(42)
np.random.seed(42)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Device:", device)

# PHYSICAL PARAMETERS
L = 1.0; alpha = 0.01; T_left = 100.0; T_right = 0.0; T_max = 10.0

# ANALYTICAL SOLUTION
def analytical_solution(x, t, n_terms=300):
    x = np.asarray(x)
    steady = 100.0 * (1.0 - x)
    transient = np.zeros_like(x, dtype=np.float64)
    for n in range(1, n_terms + 1):
        transient += (
            -200.0/(n*np.pi)
            * np.sin(n*np.pi*x)
            * np.exp(
                -alpha*(n*np.pi)**2*t   )      )
    return steady + transient
```

ANN vs PINN example

```
# CRANK NICOLSON SOLVER
def crank_nicolson():
    Nx = 101;     Nt = 1000
    dx = L/(Nx-1);    dt = T_max/(Nt-1)
    r = alpha*dt/dx**2
    x = np.linspace(0,L,Nx)
    t = np.linspace(0,T_max,Nt)
    u = np.zeros((Nt,Nx))
    u[:,0] = T_left
    u[:,-1] = T_right
    N = Nx-2
    A = np.zeros((N,N))
    B = np.zeros((N,N))
    for i in range(N):
        A[i,i] = 1+r
        B[i,i] = 1-r
        if i>0:
            A[i,i-1] = -r/2
            B[i,i-1] = r/2
        if i<N-1:
            A[i,i+1] = -r/2
            B[i,i+1] = r/2
    Ainv = np.linalg.inv(A)
    for n in range(Nt-1):
        rhs = B @ u[n,1:-1]
        rhs[0] += r*T_left
        u[n+1,1:-1] = Ainv @ rhs
    return x,t,u
print("Running finite difference solution ...")
x_fd,t_fd,u_fd = crank_nicolson()
```


ANN vs PINN example

```
# TRAINING DATA
N_data = 40; noise_std = 5.0
x_data = np.random.uniform(0,L,N_data)
t_data = np.random.uniform(0,T_max,N_data)
u_exact = analytical_solution(x_data,t_data)
u_noisy = u_exact + np.random.normal(
    0,
    noise_std,
    N_data)

# TORCH DATA
x_train = torch.tensor(
    x_data,
    dtype=torch.float32,
    requires_grad=True
).reshape(-1,1).to(device)
t_train = torch.tensor(
    t_data,
    dtype=torch.float32,
    requires_grad=True
).reshape(-1,1).to(device)
u_train = torch.tensor(
    u_noisy,
    dtype=torch.float32
).reshape(-1,1).to(device)
```

ANN vs PINN example

```
# COLLOCATION POINTS
N_colloc = 10000
x_c = torch.tensor(
    np.random.uniform(0,L,N_colloc),
    dtype=torch.float32,
    requires_grad=True
).reshape(-1,1).to(device)
t_c = torch.tensor(
    np.random.uniform(0,T_max,N_colloc),
    dtype=torch.float32,
    requires_grad=True
).reshape(-1,1).to(device)

# NETWORK
class BaseNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(2,64),
            nn.Tanh(),
            nn.Linear(64,64),
            nn.Tanh(),
            nn.Linear(64,64),
            nn.Tanh(),
            nn.Linear(64,1)        )
    def forward(self,x,t):
        x = x/L
        t = t/T_max
        X = torch.cat([x,t],dim=1)
        return self.net(X)
```


ANN vs PINN example

```
# STANDARD NN
# ============================================================

class StandardNN(nn.Module):

    def __init__(self):
        super().__init__()
        self.net = BaseNet()

    def forward(self,x,t):
        return self.net(x,t)

    def loss(self,x,t,u):
        return ((self(x,t)-u)**2).mean()
```

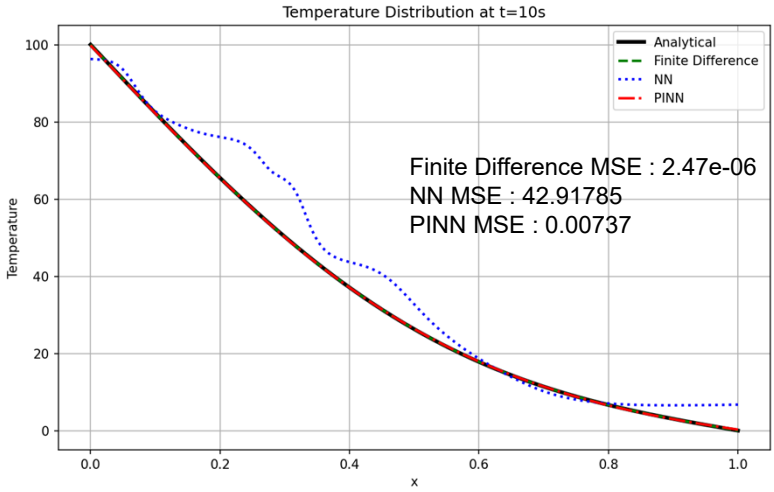
ANN vs PINN example

```
# PINN
class PINN(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = BaseNet()
    def forward(self,x,t):
        return self.net(x,t)
    def residual(self,x,t):
        u = self(x,t)
        u_t = torch.autograd.grad(u,t,
            grad_outputs=torch.ones_like(u),
            create_graph=True        )[0]
        u_x = torch.autograd.grad(u,x,
            grad_outputs=torch.ones_like(u),
            create_graph=True        )[0]
        u_xx = torch.autograd.grad(u_x,x,
            grad_outputs=torch.ones_like(u_x),
            create_graph=True        )[0]
        return u_t - alpha*u_xx
    def loss(self):
        pred = self(x_train,t_train)
        loss_data = ((pred-u_train)**2).mean()
        loss_pde = (self.residual(x_c,t_c)**2).mean()
        # BC
        t_bc = torch.linspace(0,T_max,200,device=device
        ).reshape(-1,1)
        left = self(torch.zeros_like(t_bc),t_bc        )
        right = self(torch.ones_like(t_bc),t_bc        )
        loss_bc = ( ((left-T_left)**2).mean()+
            ((right-T_right)**2).mean()        )
        # IC
        x_ic = torch.linspace(0,L,200,device=device).reshape(-1,1)
        ic = self(x_ic, torch.zeros_like(x_ic)        )
        loss_ic = ((ic**2).mean()        )
        total = (loss_data+100*loss_pde+loss_bc+50*loss_ic  )
        return total,loss_data,loss_pde
```


ANN vs PINN example

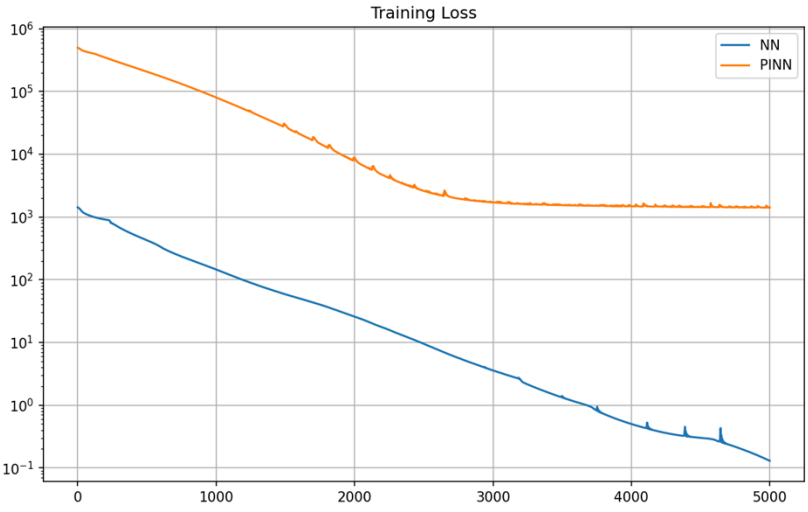
```
# TRAIN

nn_model = StandardNN().to(device)
pinn_model = PINN().to(device)
.
.
.

# COMPARISON AT t = 10s
x_plot = np.linspace(0,L,200)
.
.
.

# PLOT
plt.figure(figsize=(10,6))
.
.
.

# LOSS
plt.figure(figsize=(10,6))
.
.
.

# ERRORS
mse_nn = np.mean(
    (u_nn-u_exact)**2
)
.
.
.
```

ANN vs PINN example



Adv. App. of AI and DT 59

ANN vs PINN example



Adv. App. of AI and DT 60

نمونه شبکه عصبی مصنوعی

چارچوب ترکیبی (FEM-ANN- Experimental) برای تحلیل ژئوتکنیک شهری

یک پلتفرم ترکیبی که نقاط قوت سه روش را یکپارچه می‌کند: دقت روش‌های تجربی کلاسیک (ترزاقی)، توانایی شبیه‌سازی روش المان محدود و قدرت پیش‌بینی شبکه‌های عصبی برای تخمین پارامترهای گمشده خاک.

این روش توانست خطای پیش‌بینی را نسبت به روش‌های سنتی **بیش از ۲۰ درصد کاهش دهد** و قابلیت اطمینان بالایی در پروژه واقعی متروی جاکارتا از خود نشان دهد .

Chairul Salam M, Hendra Rezkie, Halim, Orhan Kural,
Comprehensive geotechnical analysis for urban underground construction in Jakarta,
Physics and Chemistry of the Earth, Parts A/B/C, Volume 141, Part 1, 2025,

نمونه شبکه عصبی مصنوعی

تخمین مقاومت برشی مخلوط‌های زباله‌های دانه‌ای (یک کاربرد زیست‌محیطی)

مدل **شبکه عصبی مصنوعی با آموزش بیزی (ANN-BR)** که تنها با استفاده از **۵ پارامتر کلیدی** (مانند درصد لاستیک، اندازه ذرات، تناسب محصورکننده) قادر به پیش‌بینی است.

این مدل با ساختاری ساده (۷ نود در لایه پنهان) موفق به پیش‌بینی زاویه اصطکاک اوج با **ضریب تعیین (R^2) برابر ۰,۹۴** شده است . این رویکرد، نیازی به انجام آزمایش‌های سه محوری زمان‌بر و پرهزینه را کاهش می دهد.

Hunt, H, *et al.*
Predicting the shear strength of rubber-incorporated granular waste mixtures with a machine learning approach.
Acta Geotech. **21**, 2405–2426 (2026). https://doi.org/10.1007/s11440-025-02890-7

نمونه شبکه عصبی مصنوعی

مدل‌سازی الاستوپلاستیک خاک با رویکرد ترکیبی FEM-PINN

یک شبکه PINNکه به طور همزمان **تنش، نسبت تخلخل و کرنش پلاستیک** را پیش‌بینی می‌کند. قانون رفتاری (مانند سطح تسلیم) به طور صریح تعریف نمی‌شود، بلکه به عنوان یک قید فیزیکی در تابع هزینه شبکه لحاظ می‌گردد .

این شبکه با موفقیت در روش اجزای محدود (FEM) جاسازی شده است (**FEM-PINN coupling**). در این روش، ماتریس سختی المان‌ها در هر تکرار نیوتن–رافسون با استفاده از خروجی شبکه PINNبه‌روزرسانی می‌شود .

این رویکرد ترکیبی در حل مسائل دشوار مقدار مرزی مانند **انبساط و انقباق حفره‌ای** و **بارگذاری فونداسیون**، تطابق عالی با نتایج مرجع نشان داده است و یک جایگزین قدرتمند و تفسیرپذیر برای مدل‌های رفتاری متداول است.

Mingpeng Liu, Qinghua Zhang, Raul Fuentes,
A FEM-PINN approach to modelling elastoplastic soil behaviour in boundary value problems,
Computers and Geotechnics, Volume 190, 2026,

تمرین برنامه نویسی

تمرین: یک برنامه به زبان پایتون بنویسید که یک فایل داده را خوانده و به روش شبکه عصبی مصنوعی موارد زیر را انجام دهد.

۱- تعریف یک مدل MLP
۲- تعیین و تنظیم تعداد لایه‌های پنهان و نرونها
۳- تعیین توابع فعالسازی مناسب
۴- پیش بینی مناسب مقاومت فشاری بتن
۵- تعیین میزان خطا

Data set: "Concrete Compressive Strength"
Input.: 8 Features
Output: Concrete strength
Sample No.: 1030

PyTorch : کتابخانه متن‌باز برای یادگیری عمیق که در سال ۲۰۱۶ توسط فیسبوک توسعه داده شده است. علاوه بر سادگی و پشتیبانی قوی در انجمن‌ها و شبکه‌های اجتماعی دارای ویژگیهای زیر است:

- کتابخانه‌های غنی. مثل torchvision (کار با تصاویر)، torchtext (پردازش متن) و torchaudio (کار با صدا).
- محاسبات گراف‌های پویا (قابلیت تعریف و تغییر مدل و گراف محاسباتی حین اجرا)
- سازگاری با سایر کتابخانه ها نظیر Numpy و Scikit-Learn
- پشتیبانی از کارت‌های گرافیک