Pertemuan 8

Jaringan Syaraf Tiruan

(Artificial Neural Network – ANN)

A. Konsep Neuron Tiruan dan Struktur Jaringan

1. Latar Belakang dan Motivasi

Jaringan Syaraf Tiruan (Artificial Neural Network – ANN) dikembangkan sebagai upaya meniru cara kerja otak manusia dalam mengenali pola dan mempelajari hubungan kompleks dalam data. Otak manusia terdiri dari sekitar 86 miliar neuron yang berkomunikasi melalui sinyal listrik dan kimia; ANN mencoba mengaproksimasi mekanisme ini secara matematis untuk menyelesaikan masalah prediksi, klasifikasi, maupun pengenalan pola.

ANN sangat efektif untuk:

- Data nonlinear dan kompleks
- Masalah yang sulit diselesaikan dengan pendekatan rule-based
- Pembelajaran dari contoh (supervised maupun unsupervised)

2. Konsep Dasar Neuron Tiruan

a. Struktur Neuron

Neuron tiruan adalah unit pemrosesan dasar dalam ANN. Setiap neuron menerima sejumlah input, melakukan perhitungan tertentu, lalu menghasilkan output.

Komponen neuron:

1. Input (x_1, x_2, …, x_n)

   Nilai fitur/variabel yang masuk ke neuron.

2. Bobot (w_1, w_2, …, w_n)

   Menyatakan seberapa penting tiap input.

3. Bias (b)

   Nilai tambahan yang membantu memindahkan fungsi aktivasi.

4. Fungsi Aktivasi (f)

   Mengubah nilai linear menjadi output non-linear.

5. Output (y)

b. Persamaan Neuron

Secara matematis, neuron melakukan operasi berikut:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

Kemudian nilai ini dilewatkan ke fungsi aktivasi:

$$y = f(z)$$

Neuron akan "aktif" jika nilai output melewati ambang tertentu (tergantung jenis aktivasi).

3. Struktur Dasar Jaringan Syaraf Tiruan

ANN terdiri dari banyak neuron yang dihubungkan dan tersusun dalam beberapa lapisan (*layers*).

a. Jenis Lapisan dalam ANN

1. Input Layer
   - Lapisan paling awal
   - Menerima data mentah
   - Tidak melakukan perhitungan, hanya meneruskan nilai input
2. Hidden Layer (1 atau lebih)
   - Tempat proses komputasi utama terjadi
   - Masing-masing neuron melakukan operasi *weighted sum + activation*
   - Banyaknya hidden layer mempengaruhi kemampuan ANN
     - Sedikit layer → ANN sederhana
     - Banyak layer → Deep Neural Network (DNN)
3. Output Layer
   - Menghasilkan prediksi
   - Bentuk output tergantung masalah

- Regresi → 1 neuron
- Klasifikasi 2 kelas → 1 neuron (sigmoid)
- Klasifikasi multi kelas → banyak neuron (softmax)

4. Pola Konektivitas Jaringan

a. Feedforward Neural Network (FNN)

- Arsitektur paling sederhana
- Aliran data hanya satu arah: input → hidden → output
- Tidak ada loop (acyclic)

b. Fully Connected / Dense Network

- Setiap neuron terhubung ke semua neuron pada layer berikutnya
- Banyak digunakan pada prediksi dan klasifikasi

c. Ciri Khas ANN

- Belajar dari data melalui penyesuaian bobot (*learning*)
- Mampu melakukan generalisasi
- Tahan terhadap noise data
- Mampu menangkap hubungan nonlinear

5. Mekanisme Kerja ANN Secara Konseptual

1. Forward Pass

   Data masuk ke jaringan → dihitung oleh neuron → menghasilkan prediksi awal.
2. Perhitungan Error

   Output ANN dibandingkan dengan nilai sebenarnya (label).
3. Backward Pass (Backpropagation)

   Error digunakan untuk memperbarui bobot agar prediksi selanjutnya lebih akurat.
4. Iterasi (Epoch)

   Proses diulang berkali-kali hingga error mengecil.

Tahapan ini memungkinkan ANN belajar pola data secara otomatis tanpa pemrograman eksplisit.

6. Contoh Representasi Neuron dalam Kehidupan Nyata

Neuron tiruan dapat dianalogikan seperti proses pengambilan keputusan manusia:

Contoh: Menentukan apakah akan pergi kuliah hari ini

Input:

- Cuaca cerah (x_1)
- Badan sehat (x_2)
- Ada jadwal kuliah penting (x_3)

Setiap faktor memiliki bobot (w_1, w_2, w_3) yang menunjukkan tingkat kepentingan.

Neuron akan menjumlahkan:

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Jika nilai z "cukup tinggi", neuron mengeluarkan output = pergi kuliah.

B. Fungsi Aktivasi (sigmoid, ReLU, tanh)

1. Konsep Dasar Fungsi Aktivasi

Dalam jaringan syaraf tiruan, fungsi aktivasi adalah komponen yang mengubah output linear dari neuron menjadi nilai non-linear. Tanpa fungsi aktivasi, ANN hanya akan menjadi model linear biasa, sehingga tidak mampu mempelajari pola kompleks seperti:

- Hubungan nonlinear antar fitur
- Interaksi variabel yang saling berpengaruh
- Pola dinamis seperti suara, gambar, dan teks

Fungsi aktivasi membantu ANN untuk:

- Menangkap nonlinearitas data
- Mengontrol seberapa besar sinyal diteruskan ke layer berikutnya
- Menstabilkan proses pembelajaran
- Menghindari ledakan (exploding) atau hilangnya gradien (vanishing)

- Menentukan skala output sesuai kebutuhan tugas (regresi, klasifikasi, dll.)

Pada bagian ini, kita membahas tiga fungsi aktivasi utama yang sering digunakan Sigmoid, Tanh, dan ReLU (Rectified Linear Unit).

2. Fungsi Aktivasi Sigmoid

a. Rumus Matematis

$$f(z) = \frac{1}{1 + e^{-z}}$$

b. Karakteristik

- Menghasilkan output antara 0 dan 1
- Bersifat monoton meningkat
- Menyerupai kurva S (*S-shaped*)
- Dapat dipandang sebagai "probabilitas"

c. Kelebihan Sigmoid

1. Output terikat pada rentang [0,1] → cocok untuk binary classification
2. Interpretasinya mudah (bisa dianggap peluang)
3. Halus dan kontinu → memudahkan optimasi

d. Kekurangan Sigmoid

1. Vanishing Gradient Problem

   Pada nilai z sangat besar (+/-), gradien mendekati nol, sehingga bobot sulit diperbarui.
2. Output tidak terpusat di sekitar 0 (range 0–1)

   → memperlambat konvergensi.
3. Perhitungan melibatkan eksponensial → relatif lambat untuk jaringan besar.

e. Kapan Menggunakan Sigmoid

- Output layer untuk klasifikasi dua kelas
- Model yang menginginkan output berupa probabilitas

3. Fungsi Aktivasi Hyperbolic Tangent (tanh)

a. Rumus Matematis

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

b. Rentang Output

$$-1 \leq f(z) \leq 1$$

c. Karakteristik

- Mirip sigmoid, tetapi terpusat di sekitar **0**
- Kurva *S* yang lebih curam dan simetris

d. Kelebihan tanh

- Zero-centered output → pembelajaran lebih cepat
- Performa sering lebih baik dibanding sigmoid pada hidden layer
- Menangkap nonlinearitas lebih kuat karena perubahan nilai lebih tajam

e. Kekurangan tanh

- Tetap mengalami vanishing gradient pada nilai ekstrem
- Melibatkan operasi eksponensial → cukup mahal secara komputasi

f. Kapan Menggunakan tanh

- Pada hidden layer model sederhana hingga menengah
- Saat data memiliki nilai positif dan negatif
- Ketika diperlukan fungsi aktivasi yang lebih baik daripada sigmoid

4. Fungsi Aktivasi ReLU (Rectified Linear Unit)

a. Rumus Matematis

$$f(z) = \max(0, z)$$

b. Karakteristik

- Untuk $z \leq 0$ → output = 0
- Untuk $z > 0$ → output = z (linear)

c. Kelebihan ReLU

- Sangat efisien secara komputasi
  - Tidak perlu eksponensial
  - Hanya operasi maksimum sederhana
- Mengurangi vanishing gradient

  Karena gradien bernilai 1 untuk $z > 0$
- Mempercepat proses pelatihan jaringan dalam-dalam (deep learning)
- Sering menghasilkan performa lebih tinggi pada image processing dan data kompleks.

d. Kekurangan ReLU

1. Dying ReLU Problem

   Jika terlalu banyak nilai z negatif, neuron menghasilkan output 0 terus-menerus sehingga "mati".
2. Tidak cocok untuk data yang sangat bervariasi di sekitar nol jika tidak dinormalisasi dengan baik.

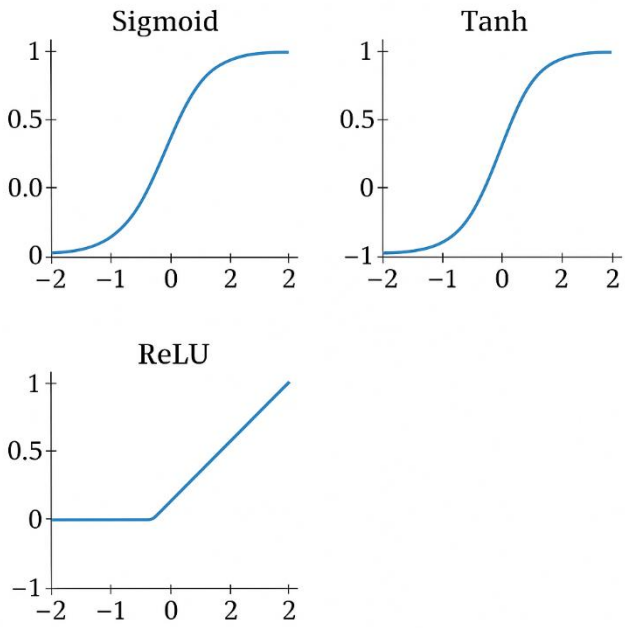
e. Kapan Menggunakan ReLU

- Hampir semua hidden layer dalam deep learning
- Model yang membutuhkan training cepat
- Komputasi besar (CNN, DNN)

5. Perbandingan Ketiga Fungsi Aktivasi

Aspek	Sigmoid	tanh	ReLU
Range	0 – 1	-1 – 1	0 – ∞
Sifat Nonlinear	Ada	Ada	Ada
Biaya Komputasi	Tinggi	Tinggi	Sangat rendah
Zero-centered	✗ Tidak	✓ Ya	✗ Tidak
Vanishing Gradient	Tinggi	Sedang	Rendah
Cocok untuk	Output biner	Hidden layer klasik	Hidden layer modern (DL)

6. Contoh Visualisasi



- Sigmoid: naik perlahan, mendatar di bagian atas & bawah.
  → mirip "S" lembut.
- tanh: mirip sigmoid tetapi merentang dari -1 ke 1.
  → simetris terhadap titik (0,0).
- ReLU: garis lurus di kanan, garis datar nol di kiri.
  → seperti huruf "L" terbalik.

Visualisasi ini membantu memahami sifat masing-masing fungsi aktivasi.

7. Mengapa Hidden Layer Jarang Menggunakan Sigmoid/Tanh Saat Ini?

Dalam deep learning modern, ReLU lebih populer karena:

- ReLU menghindari vanishing gradient pada banyak layer
- Training jauh lebih cepat
- Lebih stabil pada data berukuran besar
- Lebih mampu belajar fitur-fitur kompleks

Namun sigmoid dan tanh tetap digunakan pada situasi tertentu, khususnya lapisan output dan model-model tertentu seperti RNN klasik.

C. Proses Pembelajaran (feedforward dan backpropagation)

Proses pembelajaran dalam Jaringan Syaraf Tiruan (Artificial Neural Network – ANN) merupakan inti dari kemampuan jaringan untuk *belajar dari pengalaman*, yaitu dari contoh data yang diberikan. Pembelajaran ini terjadi melalui penyesuaian bobot (*weights*) dan bias secara bertahap sehingga jaringan mampu menghasilkan output yang semakin mendekati target yang benar. Dengan kata lain, ANN berusaha menemukan suatu fungsi matematis terbaik yang memetakan input ke output berdasarkan pola-pola yang tersembunyi di dalam data.

Pada awal pelatihan, bobot biasanya diinisialisasi secara acak. Bobot awal yang acak tersebut menyebabkan jaringan menghasilkan prediksi yang masih jauh dari benar. Melalui proses pelatihan berulang, jaringan memperbaiki bobot-bobot tersebut sehingga arah prediksinya makin mendekati nilai sebenarnya. Proses ini terjadi secara iteratif, seperti seorang manusia yang terus berlatih dan mengoreksi kesalahan sampai akhirnya mencapai performa optimal.

Secara umum, terdapat dua tahap utama dalam siklus pembelajaran ANN yang terus berulang sepanjang proses training:

1. Feedforward (Propagasi Maju)

Feedforward adalah tahap ketika input mengalir melalui jaringan dari lapisan input menuju hidden layer dan akhirnya ke lapisan output. Pada tahap ini tidak ada proses pembelajaran; jaringan hanya melakukan perhitungan matematis berdasarkan bobot sementara yang dimilikinya. Setiap neuron diaktifkan melalui fungsi aktivasi tertentu, menghasilkan sinyal yang kemudian diteruskan ke neuron berikutnya. Proses ini menghasilkan *prediksi awal* dari jaringan.

Pada tahap ini, jaringan *belum belajar*; hanya melakukan perhitungan berdasarkan bobot saat ini.

a. Langkah-langkah Feedforward

Misalkan input X masuk ke neuron:

1) Menghitung kombinasi linear

$$z = \sum(w_i x_i) + b$$

2) Mengaktifkan neuron

$$a = f(z)$$

di mana *f* adalah fungsi aktivasi (misalnya sigmoid, tanh, atau ReLU).

3) Meneruskan sinyal ke layer berikutnya

Setiap output dari neuron menjadi input bagi neuron di layer berikutnya.

4) Menghasilkan output akhir

Output layer mengeluarkan prediksi (\hat{y}).

b. Ilustrasi Singkat Feedforward

Input → Hidden Layer → Output Layer

Melalui operasi berulang:

(Weighted Sum) → (Activation) → (Pass to next layer)

Contoh kecil:

- Input: tinggi badan, berat badan
- Output: prediksi "tinggi/BMI normal atau tidak"

Model hanya menghitung secara matematis tanpa mengetahui benar/salahnya dulu.

2. Perhitungan Error

Setelah menghasilkan output y, jaringan menghitung error terhadap nilai target (y). Contoh fungsi error untuk regresi:

$$E = \frac{1}{2}(y - \hat{y})^2$$

Untuk klasifikasi, sering digunakan cross-entropy.

Error inilah yang kemudian digunakan pada proses backpropagation.

3. Backpropagation (Propagasi Balik)

Backpropagation adalah algoritma inti yang digunakan ANN untuk *belajar* dengan menyesuaikan bobot berdasarkan kesalahan prediksi. Konsepnya adalah: Jika prediksi salah, bobot harus diperbaiki. Perbaikan dilakukan dengan menghitung seberapa besar kontribusi setiap bobot terhadap error. Backpropagation menggunakan turunan (gradien) untuk menentukan arah perubahan bobot yang mengurangi error.

a. Tujuan Backpropagation

1. Meminimalkan error (loss function)
2. Menghitung gradien dengan efisien
3. Menentukan perubahan bobot menggunakan optimasi seperti Gradient Descent

b. Tahapan Backpropagation

1) Hitung error di output layer

$$\delta_{output} = (y - \hat{y}) \cdot f'(z)$$

2) Backpropagate error ke hidden layer

Error dipropagasi ke belakang melalui bobot:

$$\delta_{hidden} = \delta_{output} \cdot w \cdot f'(z)$$

3) Hitung perubahan bobot

Gunakan **gradient descent**:

$$w_{baru} = w_{lama} + \eta \cdot \delta \cdot x$$

di mana:

- n = learning rate (antara 0.01 – 0.1 biasanya)
- δ = error term
- (x) = input ke bobot tersebut

4) Perbarui bias

$$b_{baru} = b_{lama} + \eta \cdot \delta$$

5) Ulangi hingga error minimum tercapai

Proses feedforward → backpropagation terus berulang dalam *epoch*.

4. Peran Gradient Descent dalam Backpropagation

Gradient descent memastikan bobot bergerak ke arah yang **menurunkan error paling cepat**.

Analogi:

- Anda berdiri di atas bukit (error besar).
- Tujuan Anda adalah turun ke lembah (error minimal).
- Gradien memberi tahu arah paling curam untuk turun.

Learning rate menentukan besar "langkah kaki"; terlalu besar → lompat-lompat, terlalu kecil → lambat.

5. Contoh Alur Lengkap Feedforward & Backpropagation

Untuk memperjelas, berikut alur satu siklus (epoch):

1. Masukkan input data ke jaringan
2. Feedforward → menghasilkan prediksi (y)
3. Hitung error → selisih antara y dan (y)
4. Backpropagation
    - Error dihitung di output
    - Error dipropagasi ke hidden layer
    - Hitung gradien dan update bobot
5. Bobot diperbarui
6. Latih ulang dengan data berikutnya

Setelah ribuan iterasi, jaringan akan memiliki bobot yang optimal sehingga prediksi semakin akurat.

6. Pentingnya Normalisasi & Learning Rate

Agar proses feedforward dan backprop berjalan baik, sering dilakukan:

a. Normalisasi data

- Mencegah nilai input terlalu besar
- Mempercepat konvergensi

b. Learning rate yang tepat

- Terlalu kecil: lambat belajar
- Terlalu besar: model tidak stabil