Jaringan dan Infrastruktur Sistem Cerdas (JSC)

Pertemuan 9 : Praktik Pengolahan Data

Alauddin Maulana Hirzan

Fakultas Teknologi Informasi dan Komunikasi
Universitas Semarang

Outline

Sumber Data

Data mentah (*raw data*) adalah data yang diperoleh langsung dari sumber tanpa melalui proses pengolahan. Dalam konteks sistem cerdas dan jaringan, sumber utama meliputi:

- Sensor IoT :
  - Sensor menghasilkan data kontinu (time-series)
    - Suhu (temperature)
    - Kelembaban (humidity)
    - Tekanan (pressure)
  - Karakteristik:
    - Real-time
    - Rentan noise
    - Sampling periodik

Sumber Data

- Network Logs
  - Log jaringan mencatat aktivitas sistem:
    - Traffic data (packet size, latency)
    - Access logs
    - Error logs
  - Karakteristik:
    - Volume besar
    - Tidak selalu terstruktur
    - Mengandung anomali

Permasalahan Umum

1. Noise (Gangguan Sinyal)
   - Noise adalah fluktuasi acak yang tidak merepresentasikan kondisi sebenarnya.
   - Contoh: Sensor suhu membaca 25°C → tiba-tiba 80°C → kembali 26°C
2. Outlier
   - Nilai ekstrem yang jauh dari distribusi normal.
   - Contoh: Data suhu normal: 24–28°C, Outlier: 100°C
3. Data Spike
   - Lonjakan tiba-tiba dalam waktu singkat.
   - Contoh: Traffic jaringan melonjak drastis dalam 1 detik

Contoh


IoT Raw Data with Noise, Outliers and Spikes
Raw IoT Sensor Data
Outliers
Spikes
Sensor Value (°C)
Time (arbitrary units)
0
20
40
60
80
100
5
10
15
20
25
30
35
40

Generate Random Data - Bagian 1

```
import numpy as np
import matplotlib.pyplot as plt

# Generate synthetic IoT sensor data (e.g. temperature over time)
np.random.seed(42)
time = np.linspace(0, 100, 1000)  # 1000 data points

# Base signal: normal trend + daily-ish variation
base = 25 + 5 * np.sin(2 * np.pi * time / 50)

# Add Gaussian noise
noise = np.random.normal(0, 0.8, len(time))
```

Generate Random Data - Bagian 2

```
# Add random outliers (rare large deviations)
outliers = np.random.normal(0, 0.3, len(time))
outlier_idx = np.random.choice(len(time), 8, replace=False)
outliers[outlier_idx] += np.random.choice([-15, 12, -18, 14], 8)

# Add spikes (short sudden bursts)
spikes = np.zeros(len(time))
spike_idx = np.random.choice(len(time), 6, replace=False)
spikes[spike_idx] = np.random.choice([8, -10, 12, -9], 6)
```

Generate Random Data - Bagian 3

```
# Final raw IoT data
y = base + noise + outliers + spikes
X = np.linspace(0, 100, 1000).reshape(-1, 1)

print(X[:10])
print(y[:10])
```

Ilustrasi

[4]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Generate synthetic IoT sensor data (e.g. temperature over time)
np.random.seed(42)
time = np.linspace(0, 100, 1000)  # 1000 data points

# Base signal: normal trend + daily-ish variation
base = 25 + 5 * np.sin(2 * np.pi * time / 50)

# Add Gaussian noise
noise = np.random.normal(0, 0.8, len(time))
# Add random outliers (rare large deviations)
outliers = np.random.normal(0, 0.3, len(time))
outlier_idx = np.random.choice(len(time), 8, replace=False)
outliers[outlier_idx] += np.random.choice([-15, 12, -18, 14], 8)

# Add spikes (short sudden bursts)
spikes = np.zeros(len(time))
spike_idx = np.random.choice(len(time), 6, replace=False)
spikes[spike_idx] = np.random.choice([8, -10, 12, -9], 6)

# Final raw IoT data
y = base + noise + outliers + spikes
X = np.linspace(0, 100, 1000).reshape(-1, 1)

print(X[:10])
print(y[:10])
```

```
[[0.        ]
 [0.1001001]
 [0.2002002]
 [0.3003003]
 [0.4004004]
 [0.5005005]
 [0.6006006]
 [0.7007007]
 [0.8008008]
 [0.9009009]]
[25.81717795 25.22967175 25.66181617 26.21298232 25.27361715 25.2450025
 26.90893854 26.24419387 25.4415955  25.83832183]
```

[ ]:

Teknik Filtering

Moving Average Filtering

```
import pandas as pd

def moving_average(y, window_size=15):
    return pd.Series(y).rolling(window=window_size,
                                center=True,
                                min_periods=1).mean().values

window_size = 15
y_filtered = moving_average(y, window_size)
```

Teknik Filtering

Menampilkan Gambar 1

```
plt.plot(X.flatten(), y, 'b-',
          linewidth=1.0, alpha=0.6,
          label='Raw IoT Data (with noise, spikes & outliers)')
plt.plot(X.flatten(), y_filtered, 'r-',
          linewidth=2.0,
          label=f'Moving Average Filter (window={window_size})')

plt.scatter(X[spike_idx].flatten(), y[spike_idx],
                  color='orange', s=70, marker='^',
                  zorder=5, label='Spikes')
```

Teknik Filtering

Menampilkan Gambar 2

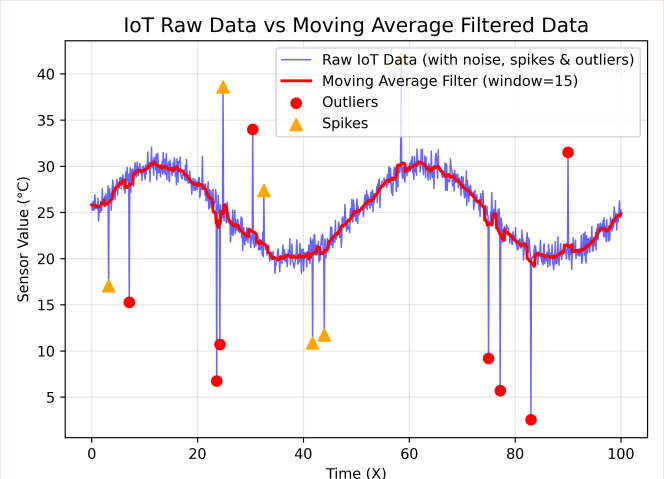
```
plt.scatter(X[outlier_idx].flatten(), y[outlier_idx],
                  color='red', s=50, zorder=5, label='Outliers')

plt.title('IoT Raw Data vs Moving Average Filtered Data',
          fontsize=14)
plt.xlabel('Time (X)')
plt.ylabel('Sensor Value (°C)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```

Ilustrasi

[6]:
```python
import pandas as pd

def moving_average(y, window_size=15):
    return pd.Series(y).rolling(window=window_size, center=True, min_periods=1).mean().values

window_size = 15
y_filtered = moving_average(y, window_size)
```

[14]:
```python
plt.plot(X.flatten(), y, 'b-', linewidth=1.0, alpha=0.6, label='Raw IoT Data (with noise, spikes & outliers)')
plt.plot(X.flatten(), y_filtered, 'r-', linewidth=2.0, label=f'Moving Average Filter (window={window_size})')

# Highlight anomalies
plt.scatter(X[outlier_idx].flatten(), y[outlier_idx],
            color='red', s=50, zorder=5, label='Outliers')
plt.scatter(X[spike_idx].flatten(), y[spike_idx],
            color='orange', s=70, marker='^', zorder=5, label='Spikes')

plt.title('IoT Raw Data vs Moving Average Filtered Data', fontsize=14)
plt.xlabel('Time (X)')
plt.ylabel('Sensor Value (°C)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.savefig("moving-average.png", dpi=600)
```

Moving Average



Teknik Filtering

Low Pass Filtering

```
from scipy import signal

def low_pass_filter(y, cutoff_freq=0.05, fs=1.0, order=4):
    nyquist = fs / 2
    normal_cutoff = cutoff_freq / nyquist
    b, a = signal.butter(order, normal_cutoff, btype='low',
                         analog=False)
    y_filtered = signal.filtfilt(b, a, y)  # Zero-phase filtering
    return y_filtered

# Apply Low-Pass Filter
y_filtered = low_pass_filter(y, cutoff_freq=0.08, fs=1.0, order=5)
```

Teknik Filtering

Menampilkan Gambar 1

```
plt.plot(X.flatten(), y, 'b-',
           linewidth=1.0, alpha=0.6,
           label='Raw IoT Data (with noise, spikes & outliers)')
plt.plot(X.flatten(), y_filtered, 'r-',
           linewidth=2.0,
           label=f'Low Pass Filter (window={window_size})')

plt.scatter(X[spike_idx].flatten(), y[spike_idx],
                  color='orange', s=70, marker='^',
                  zorder=5, label='Spikes')
```

Teknik Filtering

Menampilkan Gambar 2

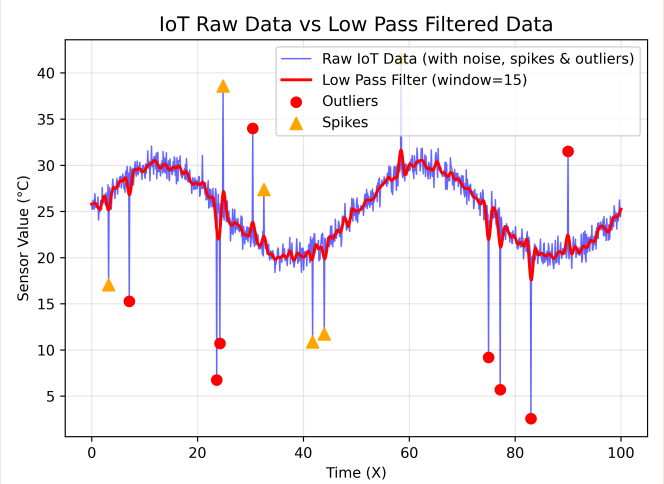
```
plt.scatter(X[outlier_idx].flatten(), y[outlier_idx],
                  color='red', s=50, zorder=5, label='Outliers')

plt.title('IoT Raw Data vs Low Pass Filtered Data',
          fontsize=14)
plt.xlabel('Time (X)')
plt.ylabel('Sensor Value (°C)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```

Ilustrasi

[16]:
```python
from scipy import signal

def low_pass_filter(y, cutoff_freq=0.05, fs=1.0, order=4):
    nyquist = fs / 2
    normal_cutoff = cutoff_freq / nyquist
    b, a = signal.butter(order, normal_cutoff, btype='low', analog=False)
    y_filtered = signal.filtfilt(b, a, y)  # Zero-phase filtering
    return y_filtered

# Apply Low-Pass Filter
y_filtered = low_pass_filter(y, cutoff_freq=0.08, fs=1.0, order=5)
```

Ilustrasi

Low Pass



Teknik Filtering

Median Filtering

```
from scipy import ndimage

def median_filter(y, window_size=15):
    return ndimage.median_filter(y, size=window_size)

# Apply Median Filter
window_size = 15
y_filtered = median_filter(y, window_size)
```

Teknik Filtering

Menampilkan Gambar 1

```
plt.plot(X.flatten(), y, 'b-',
          linewidth=1.0, alpha=0.6,
          label='Raw IoT Data (with noise, spikes & outliers)')
plt.plot(X.flatten(), y_filtered, 'r-',
          linewidth=2.0,
          label=f'Median Filter (window={window_size})')

plt.scatter(X[spike_idx].flatten(), y[spike_idx],
                 color='orange', s=70, marker='^',
                 zorder=5, label='Spikes')
```

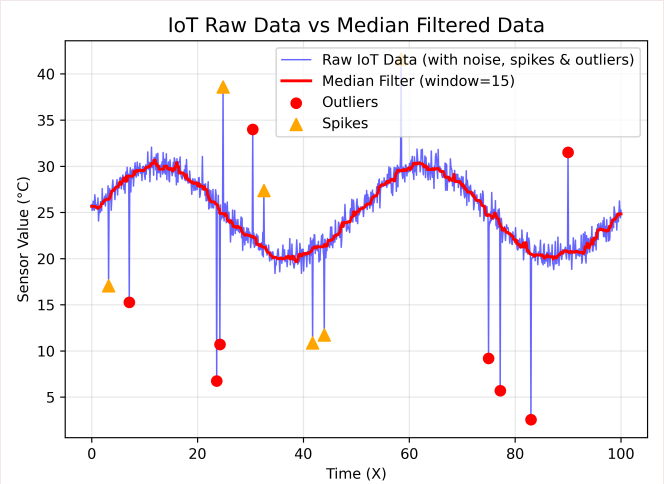
Menampilkan Gambar 2

```
plt.scatter(X[outlier_idx].flatten(), y[outlier_idx],
                  color='red', s=50, zorder=5, label='Outliers')

plt.title('IoT Raw Data vs Median Filtered Data',
          fontsize=14)
plt.xlabel('Time (X)')
plt.ylabel('Sensor Value (°C)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```

Ilustrasi

[21]:
```
from scipy import ndimage

def median_filter(y, window_size=15):
    return ndimage.median_filter(y, size=window_size)

# Apply Median Filter
window_size = 15
y_filtered = median_filter(y, window_size)
```

Median



Penyimpanan Data

Mengapa Data Perlu Disimpan

Penyimpanan data merupakan komponen penting dalam sistem cerdas, khususnya pada IoT dan jaringan.

1. Analisis Historis
   - Data yang tersimpan memungkinkan analisis pola masa lalu (time-series analysis)
2. Training Model
   - Machine learning membutuhkan data historis sebagai dataset
3. Monitoring
   - Data real-time dan historis

Jenis Penyimpanan

1. Penyimpanan Lokal
   - File CSV
   - File JSON
2. Database
   - SQL (Relation Database)
     - Data terstruktur (tabel)
     - Mendukung query kompleks (JOIN, aggregation)
   - NoSQL Database
     - Tidak berbasis tabel
     - Format dokumen (JSON-like)
     - Fleksibel untuk data IoT
3. Cloud Storage
   - Firebase
   - AWS IoT

Lastly

Terima Kasih