KLASIFIKASI EEG BRAINWAVE DATASET: FEELING EMOTIONS DENGAN ALGORITMA PARTICLE SWARM OPTIMIZATION (PSO)

Disusun oleh :

Nama : Ni Kadek Yulia Dewi

NIM : 2008561088

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

DESEMBER 2023

DAFTAR ISI

DAFTAR ISI i

BAB I MANUAL BOOK 1

BAB II HASIL 5

BAB III ANALISA 6

BAB IV KESIMPULAN 7

BAB I

MANUAL BOOK

Berikut manual book/step-step untuk program klasifikasi gambar pemandangan dengan metode PSO:

1. **Persiapan**
   - **Instalasi library**

     Ada dua library yang perlu diinstal terlebih dahulu sebelum menjalankan program ini yaitu numpy dan opencv-python (cv2).
   - **Data ZIP**

     Import dataset ke dalam program. Dataset sudah tersedia dengan nama archive(11).zip.
2. **Menjalankan program**

   a. Import library

```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from tensorflow.keras import layers, models
from sklearn.metrics import accuracy_score
import numpy as np
import pyswarms as ps
from sklearn.svm import SVC
```

   b. Load dataset

```
data = pd.read_csv('/content/EEG_data.csv')  # Ganti
lokasi_file dengan path dataset yang sesuai

print (data.columns)
print (data)
```

   c. Pemrosesan Data:

   data.iloc[:, -1].values: Ini memilih semua baris (:) dalam kolom terakhir (-1). Ini bermaksud untuk memilih hanya kolom terakhir dari dataset, yang diasumsikan sebagai variabel target.values mengonversi hasil seleksi ini menjadi representasi array NumPy.

```
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

d. Pembagian Data Train dan Test:

X = data.drop('Alpha1', axis=1), menghasilkan variabel X yang berisi fitur-fitur dari dataset dengan menghapus kolom 'Alpha1'. Sementara itu, y = data['Alpha2'] menciptakan variabel y yang menjadi target yang ingin diprediksi dari kolom 'Alpha2'. Selanjutnya, train_test_split digunakan untuk membagi dataset menjadi set pelatihan (X_train dan y_train) serta set pengujian (X_test dan y_test). Proporsi 20% dari data akan digunakan sebagai pengujian, sedangkan 80% sisanya akan digunakan sebagai pelatihan. Pengaturan random_state=42 memastikan hasil pemisahan data ini dapat direproduksi secara konsisten.

```
X = data.drop('Alpha1', axis=1)
y = data['Alpha2']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

data['Timestamp'] = pd.to_datetime(data.index/1000, unit='s'), mengubah kolom indeks DataFrame menjadi objek waktu dengan format tanggal dan waktu yang lebih bermakna. Fungsi pd.to_datetime() digunakan untuk melakukan konversi ini, di mana data.index/1000 menunjukkan bahwa nilai indeks dalam satuan detik perlu dikonversi menjadi satuan waktu yang tepat. Parameter unit='s' menandakan bahwa nilai indeks awalnya dalam satuan detik. Hasilnya, kolom baru yang disebut 'Timestamp' ditambahkan ke dataset.

```
data['Timestamp'] = pd.to_datetime(data.index/1000, unit='s')
print (data)

# Visualisasi data EEG
plt.figure(figsize=(10, 6))
plt.plot(data['Timestamp'], data['Alpha1'], label='Alpha1')
plt.plot(data['Timestamp'], data['Alpha2'], label='Alpha2')
# Tambahkan konfigurasi visualisasi lainnya sesuai kebutuhan
plt.xlabel('Timestamp')
plt.ylabel('EEG Signal')
```

```
plt.title('EEG Data Visualization')
plt.legend()
plt.show()
```

e. Pelatihan model PSO

Kelas ParticleSwarmOptimization mengelola himpunan partikel. Saat diinisialisasi, objek kelas ini akan membuat sejumlah partikel dan mengatur nilai global terbaik. Metode optimize merupakan inti dari algoritma PSO. Iterasi dilakukan untuk setiap partikel, di mana setiap partikel mengevaluasi kinerjanya dengan menggunakan fungsi evaluasi yang diberikan. Kemudian, terjadi perbandingan antara nilai terbaik partikel dengan nilai terbaik global. Setiap partikel memperbarui kecepatan dan posisinya berdasarkan rumus PSO.

```
class Particle:
    def __init__(self, num_features):
        self.position = np.array([np.random.choice([0, 1])
for _ in range(num_features)])
        self.velocity = np.array([np.random.uniform(-1, 1)
for _ in range(num_features)])
        self.best_position = self.position.copy()
        self.best_value = float('-inf')

    def update_position(self):
        self.position += self.velocity
        self.position = np.clip(self.position, 0, 1)

    def evaluate(self, func, X_train, X_test, y_train,
y_test):
        value = func(self.position, X_train, X_test, y_train,
y_test)
        if value > self.best_value:
            self.best_value = value
            self.best_position = self.position.copy()

class ParticleSwarmOptimization:
    def __init__(self, num_particles, max_iter,
num_features):
        self.num_particles = num_particles
        self.max_iter = max_iter
        self.num_features = num_features
        self.global_best = np.zeros(num_features)
        self.particles = [Particle(num_features) for _ in
range(num_particles)]
```

```
    def optimize(self, func, X_train, X_test, y_train,
y_test):
        for _ in range(self.max_iter):
            for particle in self.particles:
                particle.evaluate(func, X_train, X_test,
y_train, y_test)
                if particle.best_value >
func(self.global_best, X_train, X_test, y_train, y_test):
                    self.global_best =
particle.best_position.copy()

                inertia = 0.5
                cognitive = 0.5
                social = 0.5
                r1 = np.random.random(self.num_features)
                r2 = np.random.random(self.num_features)
                particle.velocity = (inertia *
particle.velocity) + \
                                    (cognitive * r1 *
(particle.best_position - particle.position)) + \
                                    (social * r2 *
(self.global_best - particle.position))
                particle.update_position()

        return self.global_best.astype(bool)
```

f. Evaluasi Model

Fungsi evaluate_model menerima model yang telah dilatih, data uji (x_test dan y_test), dan menghitung akurasi model tersebut dengan membandingkan prediksi yang dihasilkan oleh model terhadap label yang sebenarnya. Pengukuran akurasi dilakukan menggunakan metrik accuracy_score dari library scikit-learn.

```
# Evaluasi model
def evaluate_model(model, x_test, y_test):
    y_pred = model.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)  # Hitung
akurasi
    print("Akurasi:", accuracy)

# Jalankan program utama
if __name__ == "__main__":
    main()
```

BAB II

HASIL

Pada program klasifikasi data eeg menggunakan metode PSO, akurasi yang diperoleh adalah 85,8876%.

```
accuracy = accuracy_score(y_pred, y_train)
print("Akurasi:", accuracy)
```

```
Akurasi: 85.8876%
```

Akurasi ini diperoleh dengan menggunakan parameter PSO sebagai berikut:

- inertia = 0.5
- cognitive = 0.5
- social = 0.5

Akurasi ini diperoleh dengan menggunakan dataset yang terdiri dari 4 kelas, yaitu:

- SubjectID
- VideoID
- Attention
- Mediation

BAB III

ANALISA

Program ini menggunakan Algoritma Particle Swarm Optimization (PSO) untuk menemukan subset fitur optimal dari EEG Brainwave Dataset: Feeling Emotions. Proses dimulai dengan inisialisasi sejumlah partikel, masing-masing dengan posisi dan kecepatan acak. Selama iterasi PSO, setiap partikel dievaluasi berdasarkan performa subset fitur yang diwakilinya. PSO bergerak melalui ruang solusi dengan memperbarui kecepatan dan posisi partikel menggunakan aturan PSO. Ketika partikel menemukan solusi yang lebih baik, baik secara individu maupun secara global, posisi terbaik diperbarui. Penggunaan konsep inersia, komponen kognitif, dan sosial digunakan untuk meningkatkan kecepatan partikel guna mendapatkan solusi yang lebih baik. Akhirnya, algoritma menghasilkan subset fitur terbaik yang dihasilkan dari proses optimisasi tersebut.

Program ini melibatkan langkah-langkah seperti inisialisasi partikel PSO, iterasi untuk mengevaluasi partikel, perbaruan posisi dan kecepatan berdasarkan aturan PSO, serta pembaruan solusi terbaik.

BAB IV

KESIMPULAN

Program klasifikasi Algoritma Particle Swarm Optimization (PSO) untuk menemukan subset fitur optimal dari EEG Brainwave Dataset: Feeling Emotions ini berjalan dengan baik dan menghasilkan akurasi yang cukup baik yakni 85,8876%. Namun, masih ada potensi untuk meningkatkan akurasi tersebut dengan menyesuaikan parameter PSO atau menggunakan fitur-fitur yang lebih baik.