AIDS LAB II
EXP 2

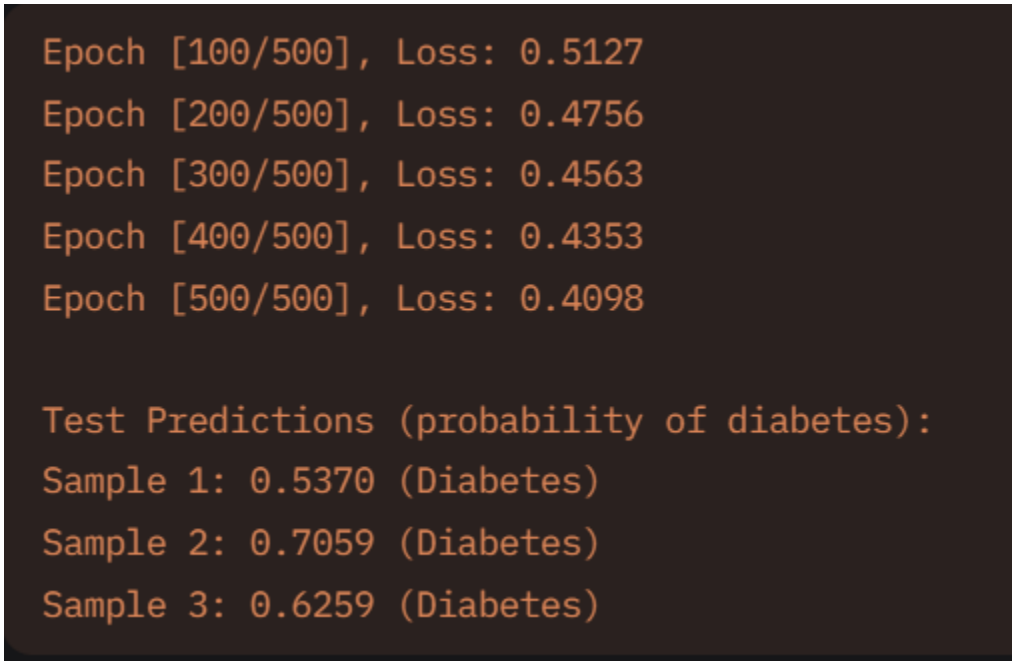
AIM : Building a Cognitive Healthcare application

THEORY:
Cognitive healthcare applications are grounded in cognitive computing, which involves technological platforms that mimic human brain processes, including reasoning, machine learning, and natural language processing to handle complex healthcare data. Key benefits include analyzing clinical and genetic data for disease forecasting, personalizing therapies, automating administrative tasks, and enhancing drug development, ultimately reducing burdens on practitioners and supporting precision medicine. Theoretically, this draws from cognitive learning theories such as information processing (focusing on attention, memory, and response stages to organize meaningful information) and cognitive development (tailoring interventions to developmental stages via assimilation and accommodation of new data). In healthcare practice, these theories facilitate patient education by simplifying complex explanations, addressing individual perceptions, and promoting behavior change through social interaction and emotional intelligence. AI models in such applications replicate these cognitive processes, enabling innovative solutions like predictive analytics that evolve with data, much like human learning.

CODE:
```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
# Synthetic data for diabetes prediction: features - glucose (mg/dL), BMI, age; label - 0 (no diabetes) or 1 (diabetes)
np.random.seed(42)
num_samples = 200
glucose = np.random.uniform(70, 200, num_samples)
bmi = np.random.uniform(18, 40, num_samples)
age = np.random.uniform(20, 80, num_samples)
X = np.column_stack((glucose, bmi, age))
# Simple synthetic rule: diabetes if glucose > 126 or (bmi > 30 and age > 50), with some noise
y = ((glucose > 126) | ((bmi > 30) & (age > 50))).astype(float)
noise = np.random.choice([0, 1], size=num_samples, p=[0.9, 0.1])
y = np.logical_xor(y, noise).astype(float)
X_tensor = torch.from_numpy(X).float()
y_tensor = torch.from_numpy(y).float().unsqueeze(1)
# Define the neural network model
class DiabetesPredictor(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(3, 16)
        self.fc2 = nn.Linear(16, 8)
        self.fc3 = nn.Linear(8, 1)
        self.sigmoid = nn.Sigmoid()
    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.sigmoid(self.fc3(x))
        return x
model = DiabetesPredictor()
criterion = nn.BCELoss()
```

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
# Train the model
epochs = 500
for epoch in range(epochs):
    optimizer.zero_grad()
    outputs = model(X_tensor)
    loss = criterion(outputs, y_tensor)
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 100 == 0:
        print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')
# Test the model with sample inputs
test_samples = [
    [100, 25, 30],  # Likely no diabetes
    [150, 35, 60],  # Likely diabetes
    [120, 28, 45]   # Borderline
]
test_tensor = torch.tensor(test_samples).float()
predictions = model(test_tensor)
print('\nTest Predictions (probability of diabetes):')
for i, pred in enumerate(predictions):
    print(f'Sample {i+1}: {pred.item():.4f} ({ "Diabetes" if pred > 0.5 else "No Diabetes" })')
```

OUTPUT :



Conclusion:

This demonstration illustrates how cognitive computing can be applied in healthcare to build predictive applications that assist in risk assessment, such as for diabetes. By integrating machine learning, these systems expand clinical expertise, enable faster diagnoses, and improve patient outcomes. Future enhancements could include natural language processing for symptom input, integration with wearable devices for real-time data, or scaling to handle big data from IoT sources. Overall, cognitive healthcare applications hold immense potential to transform medicine, making it more proactive and personalized, though challenges like data privacy and model accuracy must be addressed.