dnn_xor_QNJL

Xor problem¶

Quistian Navarro Juan Luis \ A341807@alumnos.uaslp.mx \ Ing. Sistemas Inteligentes, Gen 2021 \ Machine Learning, Group 281601

Mar/09/24¶

In [67]:

```
import numpy as np
from sklearn.neural_network import BernoulliRBM
from sklearn.model_selection import ParameterGrid
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [68]:

```
# Data Preparation for XOR
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

y = np.array([0, 1, 1, 0])

TrainData = X
TrainLabels = y

def reconstruct_from_hidden(hidden, rbm):
    #Reconstruct visible layer from hidden activations
    p = 1.0 / (1 + np.exp(-np.dot(hidden, rbm.components_) - rbm.intercept_visible_))
    return (p > 0.5).astype(int)

# Define hyperparameter grid for grid search
param_grid = {
```

```
    'n_components': [2, 3, 4],
    'learning_rate': [0.01, 0.05, 0.1],
    'batch_size': [1, 2, 3],
    'n_iter': [1000, 1500, 2000]
}

results = []

start_time_total = time.time()

# Perform grid search
for params in ParameterGrid(param_grid):
    start_time = time.time()

    rbm = BernoulliRBM(**params, verbose=0)
    rbm.fit(TrainData)
    hidden_data = rbm.transform(TrainData)
    reconstructed_data = reconstruct_from_hidden(hidden_data, rbm)
    accuracy = np.mean(reconstructed_data == TrainData)

    end_time = time.time()

    execution_time = end_time - start_time

    results.append({
        'n_components': params['n_components'],
        'learning_rate': params['learning_rate'],
        'batch_size': params['batch_size'],
        'n_iter': params['n_iter'],
        'accuracy': accuracy,
        'execution_time': execution_time
    })
end_time_total = time.time()

results_df = pd.DataFrame(results)
```

In [69]:

```
# Sort results by descending precision

results_df.head(10)
```

Out[69]:

	n_components	learning_rate	batch_size	n_iter	accuracy	execution_time
23	3	0.10	1	2000	0.750	0.293093

	n_components	learning_rate	batch_size	n_iter	accuracy	execution_time
52	4	0.10	2	1500	0.750	0.117508
42	4	0.05	2	1000	0.750	0.083694
18	2	0.10	1	1000	0.750	0.144099
16	4	0.05	1	1500	0.625	0.221584
14	3	0.05	1	2000	0.625	0.303556
25	4	0.10	1	1500	0.625	0.218129
22	3	0.10	1	1500	0.625	0.216475
12	3	0.05	1	1000	0.625	0.148033
51	4	0.10	2	1000	0.500	0.081509

In [70]:

```
# # Sort results by ascending time
# results_df = results_df.sort_values(by='execution_time', ascending=True)
# results_df.head(10)
```

In [71]:

```
print("\nTotal execution time:", end_time_total - start_time_total, "seconds")
```

```
Total execution time: 13.421512365341187 seconds
```