Optimization in ArtificialIntelligence

Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction

Group 3

2020147051 JANG GEON
2021111012 LEE SEUNG YEON

Contents

Contents

Source Paper

Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction

by Hyejung Chung and Kyung-shik Shin *

Ewha School of Business, Ewha Womans University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul 03760, Korea

* Author to whom correspondence should be addressed.

Sustainability **2018**, *10*(10), 3765; https://doi.org/10.3390/su10103765

Received: 28 September 2018 / Revised: 16 October 2018 / Accepted: 16 October 2018 / Published: 18 October 2018

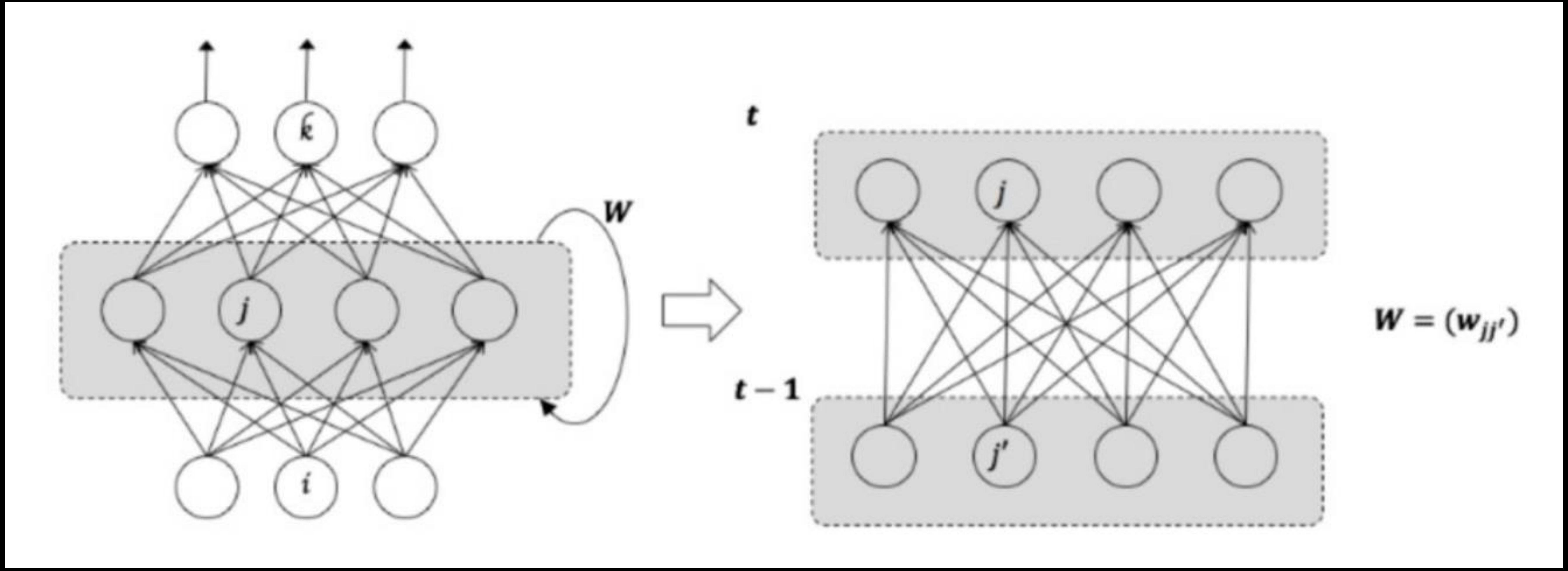
(This article belongs to the Special Issue **Expert Systems: Applications of Business Intelligence in Big Data Environments**)

Source Paper

- In the field of finance, great opportunities to create useful insights by analyzing tremendous amount of real-time data
- Develop a novel stock market prediction model using the available financial data
- Adopting deep learning technique : A hybrid approach integrating long short-term memory (LSTM) network and genetic algorithm (GA)
- Use daily Korea Stock Price Index (KOSPI) data to evaluate the proposed hybrid approach
- The hybrid model of LSTM network and GA outperforms the benchmark model

Methodology

▌Basic structure of a simple recurrent neural network (RNN)

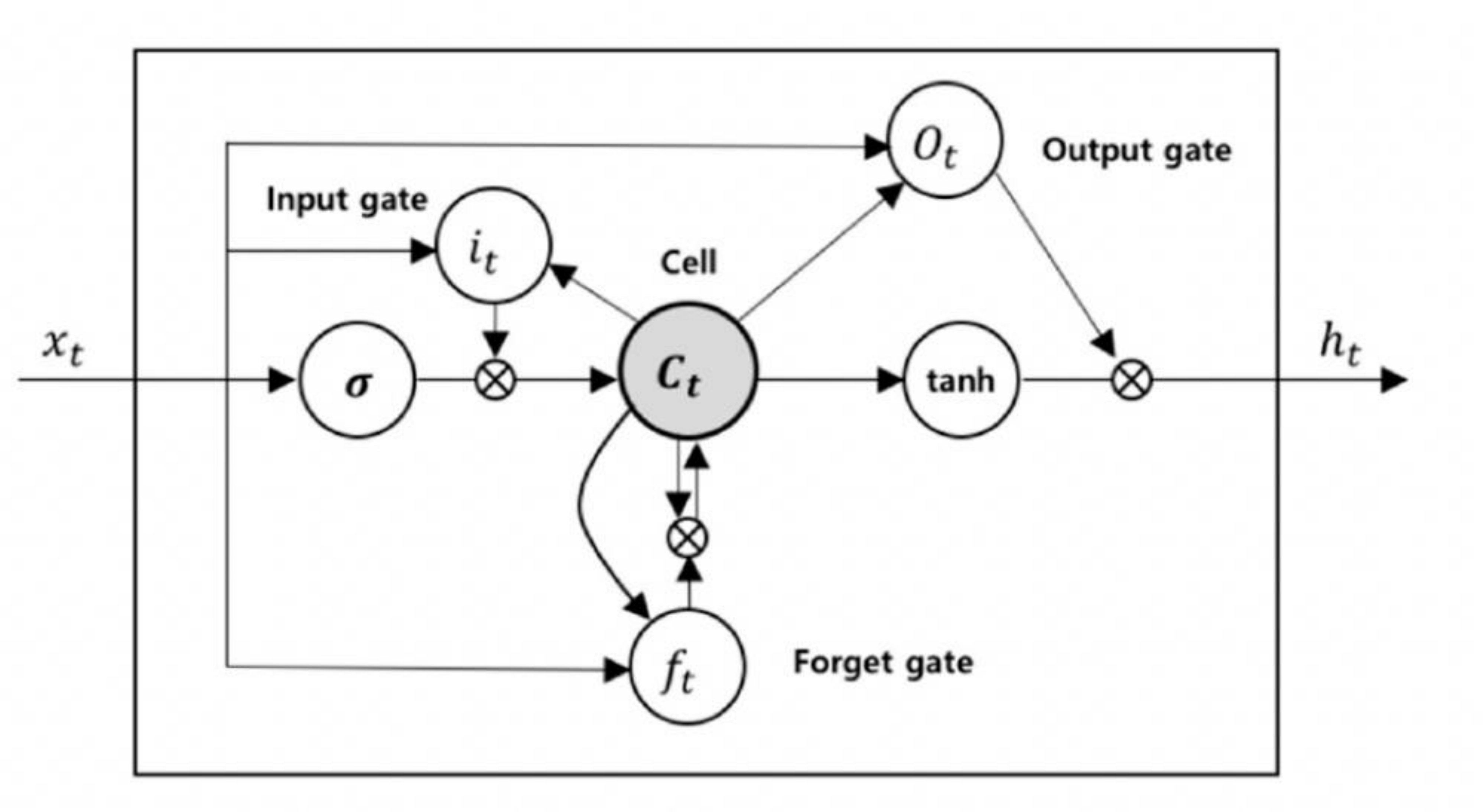


(Sustainability 2018, 10, 3765,6p)

- RNN architecture that produces an output at every time step
- recurrent connections among hidden neurons

Methodology

- Long short-term memory (LSTM) cell with gating units



(Sustainability 2018, 10, 3765,7p)

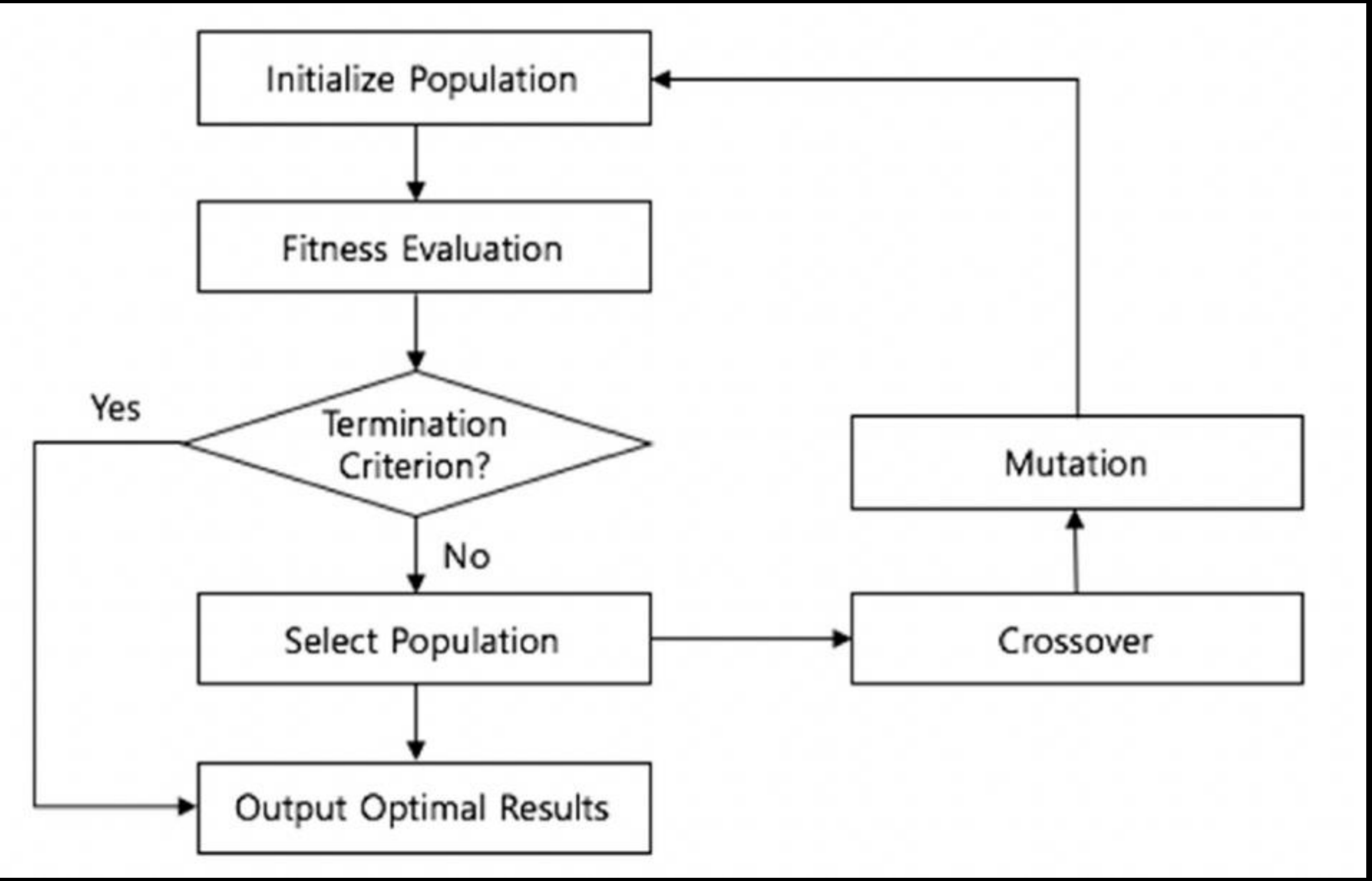
- memory cell, three multiplicative gating units; input, output, forget gate

- recurrent connections between the cells

- each gate provides continuous operations for the cells

Methodology

- Basic process of genetic algorithm (GA)

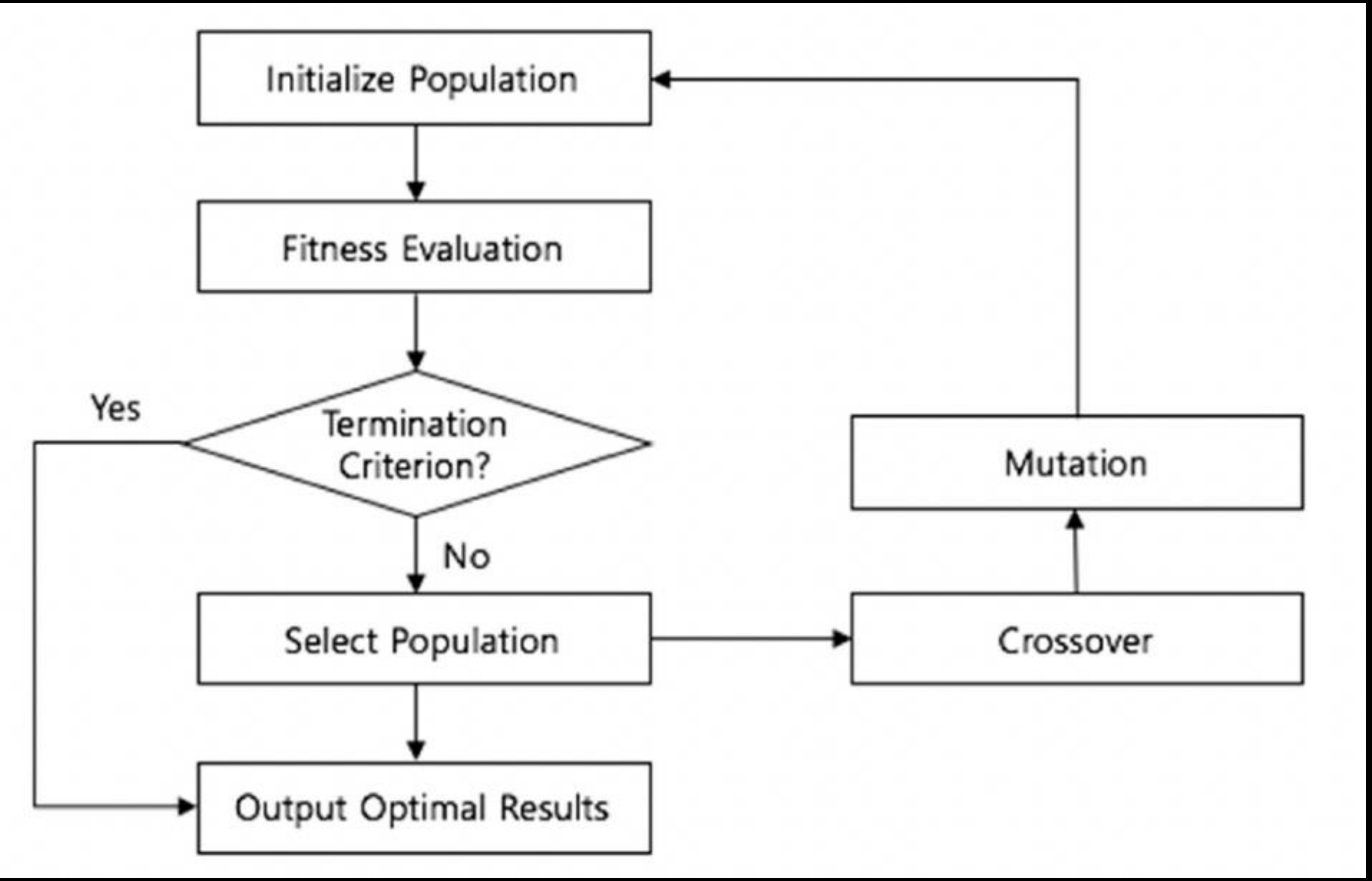


(Sustainability 2018, 10, 3765,8p)

- divided into six stages: initialization, fitness calculation, termination condition check, selection, crossover, and mutation
- The standard procedure of GA is over when the termination criteria have been satisfied
- If some termination criteria are not satisfied, the selection, crossover, and mutation processes are repeated

Methodology

- Basic process of genetic algorithm (GA)



(Sustainability 2018, 10, 3765,8p)

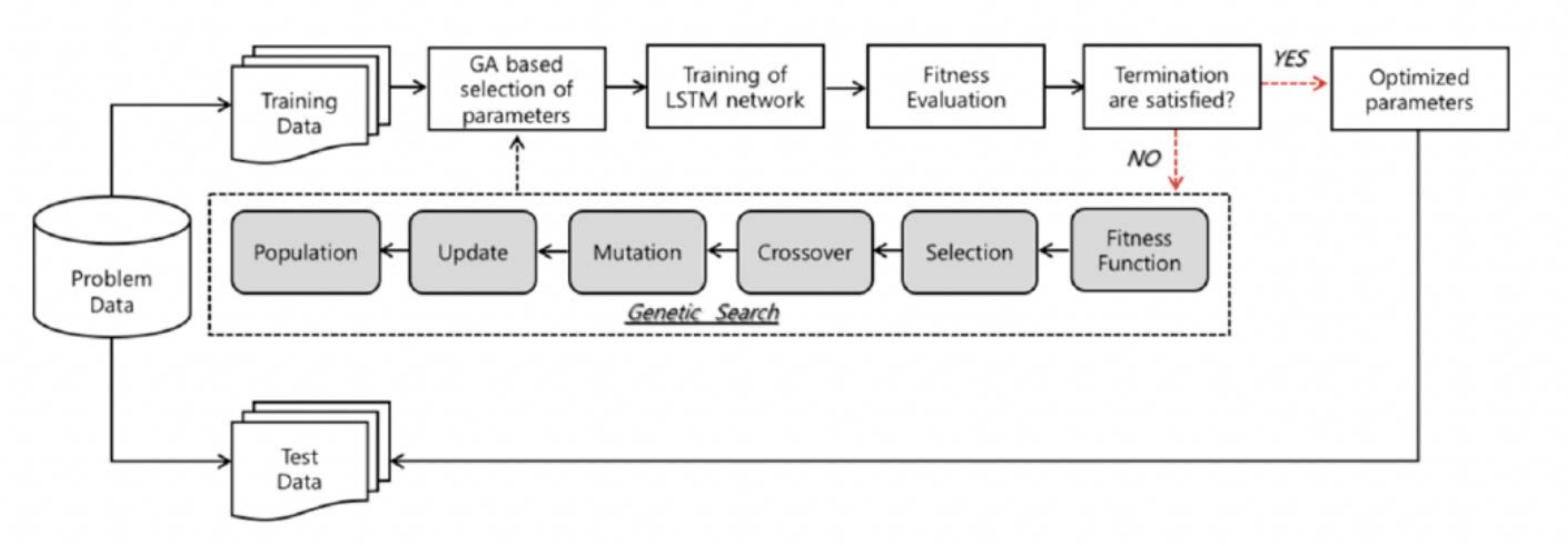
- GA parameters

population size of 70
0.7 crossover rate
0.15 mutation rate in the experiment

Stopping condition:
the number of generations 10

Methodology

▌Flowchart of the GA-LSTM model



(Sustainability 2018, 10, 3765,9p)

- a hybrid approach of LSTM network integrating GA is used to find the customized time window and number of LSTM units for financial time series prediction

Methodology

- Summary statistics of selected input variables

Name of Input	Max	Min	Mean	Standard Deviation
High price	2231.47	472.31	1439.90	541.51
Low price	2202.92	463.54	1420.12	539.65
Opening price	2225.95	466.57	1431.28	541.15
Closing price	2228.96	468.76	1430.54	540.70
Trading volume	2,379,293,952	136,328,992	420,925,245	192,841,151
Simple 10-day moving average	2208.53	474.37	1430.06	540.35
Weighted 10-day moving average	2210.60	473.76	1430.22	540.42
Relative strength index (RSI)	99.06	3.29	53.38	17.21
Stochastic K%	100.00	0.00	56.17	32.29
Stochastic D%	100.00	0.00	56.18	26.73

(Sustainability 2018, 10, 3765,11p)

Data : Korea Composite Stock Price Index (KOSPI) for 2000–2016

Implementation

Preprocessing - Normalization

```
# normalization
y_col=raw_df['clsing price']
y_df=pd.DataFrame(y_col)

scaler = MinMaxScaler()
scale_cols = ['opening price', 'high price', 'low price', 'volume']

x_df = scaler.fit_transform(raw_df[scale_cols])
x_df = pd.DataFrame(scaled_df, columns=scale_cols)

scaled_df = pd.concat([y_df, x_df], axis = 1)
```

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)},$$

Preprocessing – Data split

```
# train: validation: test = 0.68 : 0.12 : 0.2
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
X_test, X_val, y_test, y_val = train_test_split(X_train, y_train, test_size=0.15)
```

train : validation : test = 0.68 : 0.12 : 0.2

Implementation

GA fuction code

selection function

```python
# Define the selection function (tournament selection)
def selection(population, k=3):
    selected = []
    population_list = population.values.tolist()  # Convert DataFrame to a list
    for _ in range(population_size):
        if len(population_list) < k:
            competitors = population_list
        else:
            competitors = random.sample(population_list, k)
        competitors_fitness = [fitness_function(individual, X_train, y_train, X_val, y_val) for individual in competitors]

        # Check if all fitness values are the same
        if len(set(competitors_fitness)) == 1:
            selected.append(random.choice(competitors))
        else:
            max_fitness_idx = random.choice([i for i, f in enumerate(competitors_fitness) if f == max(competitors_fitness)])
            selected.append(competitors[max_fitness_idx])
    return selected
```

mutation function

```python
# Define the mutation function
def mutation(individual):
    mutated = list(individual)
    for i in range(len(mutated)):
        if random.random() < mutation_rate:
            if i == 0:
                mutated[i] = random.choice(window_sizes)
            elif i == 1:
                mutated[i] = random.choice(lstm_units)
            else:
                mutated[i] = random.choice(hidden_layers)
    return tuple(mutated)
```

crossover function

```python
# Define the crossover function
def crossover(parent1, parent2):
    crossover_point = random.randint(1, len(parent1) - 1)
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]
    return child1, child2
```

Implementation

GA parameters

population_size= 70
generations = 10
mutation_rate = 0.15
crossover_rate = 0.7

Example of population set

```
Generation 1/10
Population: [(15, 22, 1), (5, 7, 3), (10, 22, 3), (10, 22, 3), (15, 22, 1), (15, 15, 3), (5, 15, 3), (5, 22, 3), (15, 15, 2), (5, 2
2, 2), (10, 22, 2), (15, 7, 3), (10, 22, 2), (10, 15, 3), (10, 15, 2), (10, 22, 1), (5, 7, 3), (15, 7, 2), (5, 15, 1), (15, 15, 3),
(5, 22, 3), (10, 7, 3), (15, 22, 2), (10, 7, 2), (10, 22, 1), (15, 7, 1), (15, 22, 1), (15, 15, 3), (10, 15, 2), (15, 22, 1), (5, 2
2, 3), (10, 22, 3), (15, 7, 1), (5, 7, 1), (5, 22, 3), (10, 7, 2), (15, 7, 2), (5, 15, 2), (10, 7, 2), (10, 7, 2), (15, 15, 1), (5,
7, 2), (15, 22, 3), (5, 15, 1), (5, 7, 1), (15, 15, 2), (10, 22, 1), (10, 7, 1), (5, 7, 2), (5, 22, 3), (5, 15, 1), (5, 15, 1), (5,
15, 3), (10, 15, 1), (10, 15, 2), (10, 22, 3), (15, 7, 1), (10, 7, 2), (5, 7, 1), (10, 7, 2), (10, 7, 3), (15, 7, 1), (15, 22, 2),
(10, 15, 1), (5, 15, 1), (15, 7, 3), (5, 15, 2), (5, 7, 2), (5, 15, 2), (5, 7, 1)]
```

Implementation

Result of GA algorithm

```
Best Individual: Window Size=10.0, LSTM Units=7.0, Hidden Layers=3.0
Fitness (MSE): 216.778769531, Fitness (MAE): 11.314928436279297, Fitness (MAPE): 1.519520133733749

Best Individual: Window Size=5.0, LSTM Units=7.0, Hidden Layers=3.0
Fitness (MSE): 357.73718, Fitness (MAE): 14.174722290039062, Fitness (MAPE): 1.82381018996238

Best Individual: Window Size=10.0, LSTM Units=7.0, Hidden Layers=3.0
Fitness (MSE): 201.5424218, Fitness (MAE): 11.202888488769531, Fitness (MAPE): 1.533407866954803

Best Individual: Window Size=10.0, LSTM Units=7.0, Hidden Layers=3.0
Fitness (MSE): 181.82542968, Fitness (MAE): 11.161949920654297, Fitness (MAPE): 1.587774902582168

Best Individual: Window Size=15.0, LSTM Units=22.0, Hidden Layers=1.0
Fitness (MSE): 174.2593593, Fitness (MAE): 39.597802734375, Fitness (MAPE): 5.65658867359161

Best Individual: Window Size=15.0, LSTM Units=22.0, Hidden Layers=2.0
Fitness (MSE): 801.827656, Fitness (MAE): 25.437088012695312, Fitness (MAPE): 4.20361697673797

Best Individual: Window Size=10.0, LSTM Units=15.0, Hidden Layers=2.0
Fitness (MSE): 182.892050781, Fitness (MAE): 11.158654022216797, Fitness (MAPE): 1.582613289356231

Best Individual: Window Size=15.0, LSTM Units=15.0, Hidden Layers=2.0
Fitness (MSE): 246.73746093, Fitness (MAE): 13.72898712158203, Fitness (MAPE): 2.231214940547943

Best Individual: Window Size=10.0, LSTM Units=15.0, Hidden Layers=3.0
Fitness (MSE): 169.849125, Fitness (MAE): 35.482226562, Fitness (MAPE): 2.708595693111419
```

Optimal parameters

```
Genetic Algorithm finished.
Best Fitness values across all generations:
MSE: 216.778769531
MAE: 11.314928436279297
MAPE: 1.519520133733749
```

Implementation

Many to one LSTM summary

```
Model: "sequential_43"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_46 (LSTM)              (None, 5, 15)             1020

 lstm_47 (LSTM)              (None, 5, 7)              644

 lstm_48 (LSTM)              (None, 7)                 420

 dense_140 (Dense)           (None, 1)                 8

=================================================================
Total params: 2,092
Trainable params: 2,092
Non-trainable params: 0
_________________________________________________________________
```

Result

```
MSE: 216.778769531
MAE: 11.314928436279297
MAPE: 1.519520133733749
```

Implementation

to one bidirectional LSTM summary

```
Model: "sequential_70"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_19 (Bidirecti  (None, 5, 30)            2040
 onal)

 bidirectional_20 (Bidirecti  (None, 5, 30)            5520
 onal)

 bidirectional_21 (Bidirecti  (None, 14)               2128
 onal)

 dense_163 (Dense)           (None, 1)                 15

=================================================================
Total params: 9,703
Trainable params: 9,703
Non-trainable params: 0
_________________________________________________________________
```

Result

```
90/90 [==============================] - 3s 3ms/step
MSE: 217.35592
MAE: 11.024706
MAPE: 0.98030167
```

Implementation

Many to on LSTM summary

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_34 (Bidirecti  (None, 5, 30)            2040
 onal)

 bidirectional_35 (Bidirecti  (None, 5, 14)            2128
 onal)

 time_distributed_5 (TimeDis  (None, 5, 1)             15
 tributed)

=================================================================
Total params: 4,183
Trainable params: 4,183
Non-trainable params: 0
_________________________________________________________________
```

Result

```
90/90 [==============================] - 1s 2ms/step
MSE: 175.9421.8
MAE: 10.044233
MAPE: 0.9063663
```

Conclusion

- **Performance measure & comparison**

	Paper	LSTM	Bi-LSTM	Many to many bi-LSTM
MSE	181.99	216.78	217.35	175.94
MAE	10.21	11.31	11.02	10.04
MAPE	0.91	1.51	1.51	0.90

LSTM < bi-LSTM < paper < many to many bi -LSTM

Thank you for listening!