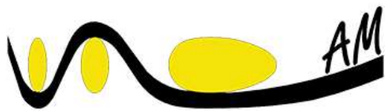
LECTURE 8. RNN

(Recurrent Neural Network)

MANU 465

Ahmad Mohammadpanah

PhD, PEng



AIntelligentManufacturing.com

Deep Learning Algorithms & Theirs Applications

- **ANN**----------→ for Regression & Classification
- **CNN**----------→ for Computer Vision
- **RNN**----------→ for Time Series Analysis

} Supervised learning

X → labels y

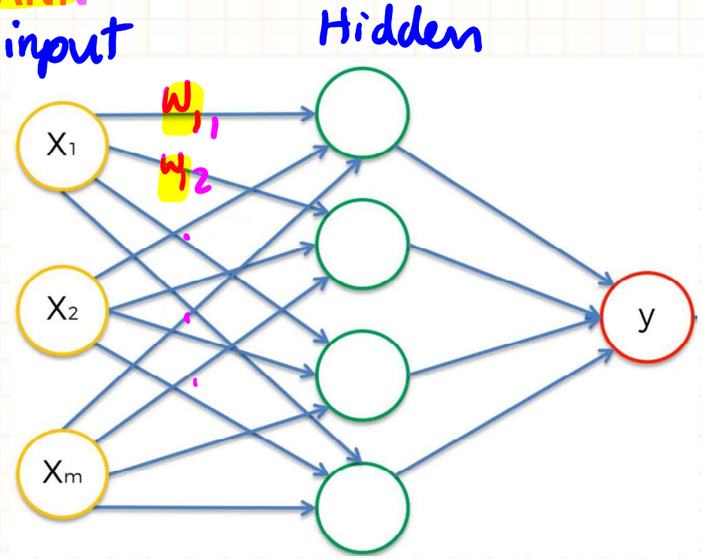
- **Self-Organizing Maps**----------→ for Feature Detection
- **Boltzmann Machines**----------→ for Recommendation Systems
- **AutoEncoder**----------→ for Recommendation Systems

} unsupervised

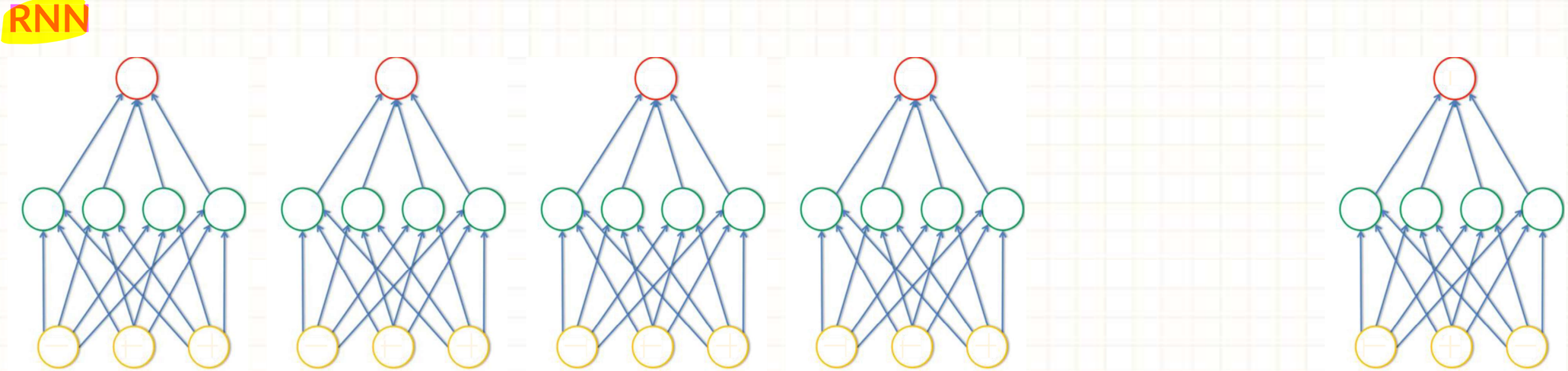
X

Recurrent Neural Network (RNN)

ANN



RNN



RNN Applications

- One-to-Many,

one image

A cat eatting watermelon.

many info.

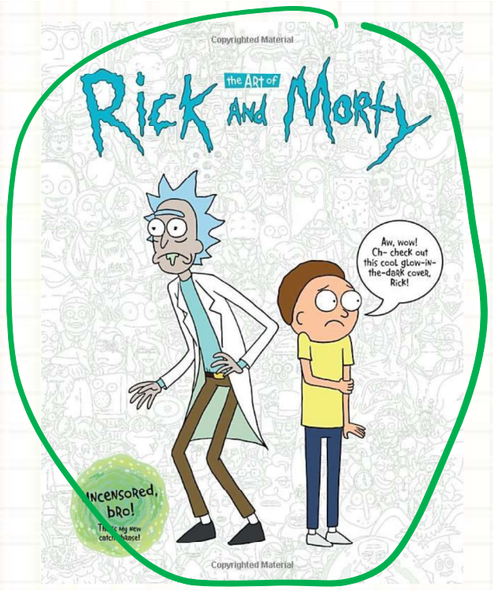
- Many-to-One,

" Food was good, price was reasonable ; waitren was unprofessional. "

→ . Positive

. score 85

- Many-to-Many



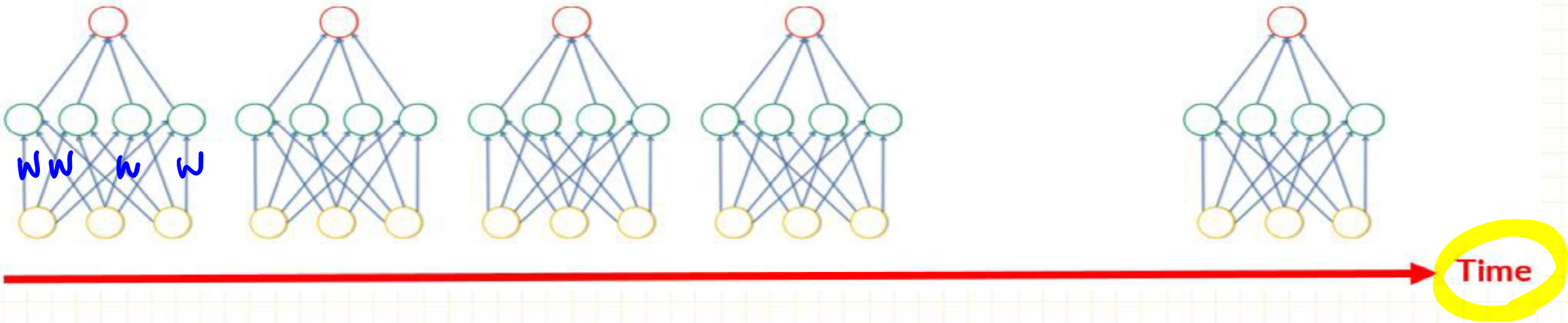
many info in movie

- Cartoon
- Family
- Scicnse fiction
- Curse langnages
- dysfunction family

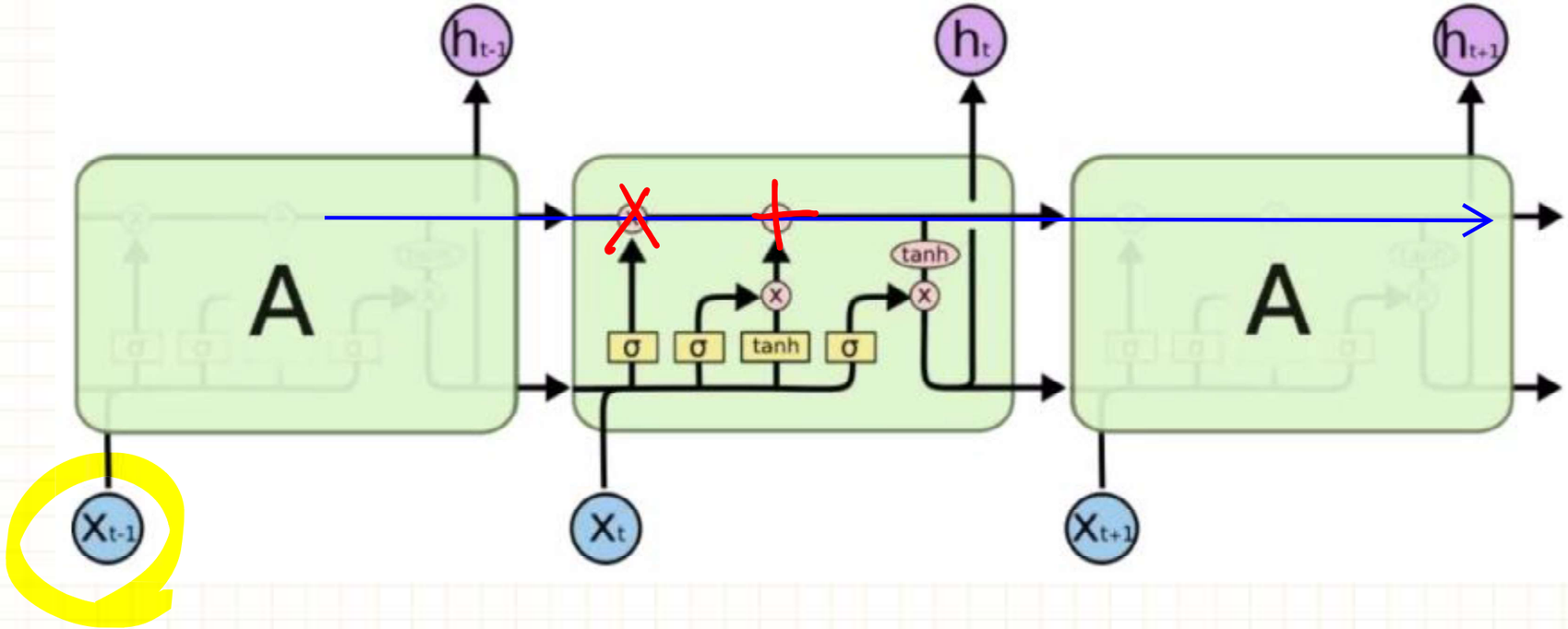
⋮

many out put

The Issue with ANN in Solving Time Series Problems: Vanishing Gradient



The Solution: Long-Short Term Memory (LSTM)



Example) Stock Market Trend Prediction (Code & Data on Canvas)

X train

Date	Price
1	300
2	310
3	280
⋮	⋮
1000	500

exchange volume

7552

2000

0	1	2	...	59
300	310	280	...	240
310	280	...	240	245
280	...	...	245	258
⋮	⋮			

y train

60
245
258 ← 61
232 ← 62
⋮

Implement in Python:

Step 1. Scaling

Note: For Feature Scaling in RNN Models, the recommendation is Normalization, and Not Standardization.

$$x_{Standard} = \frac{x - \bar{x}}{StD(x)}$$

-2 , 2

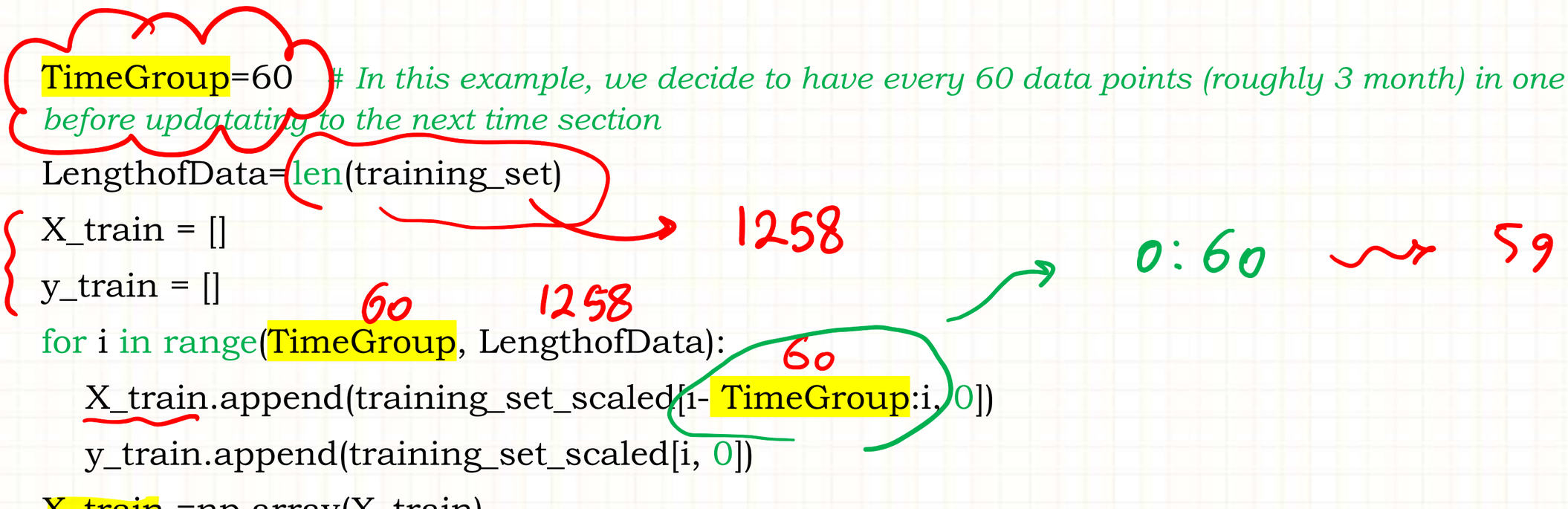
$$x_{Normal} = \frac{x - min}{max - min}$$

0 - 1

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
training_set_scaled = sc.fit_transform(training_set)
```

Step 2. Creating a data Structure with Timesteps

```
TimeGroup=60  # In this example, we decide to have every 60 data points (roughly 3 month) in one group,
before updatating to the next time section
LengthofData=len(training_set)
X_train = []
y_train = []
for i in range(TimeGroup, LengthofData):
    X_train.append(training_set_scaled[i- TimeGroup:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train =np.array(X_train)
y_train =np.array(y_train)
```



Step 3. Reshaping

The structure which is expected by RNN is always (# of rows, # of columns, # of indicators).

stock price

Here, we have only one indicator, which is Stock Price; you may have examples with more than 1 indicators, for example the Stock price, and the number of exchanges, etc.

```
#X_train.shape[0]      or      len(X_train[:,0])    # gives you the number of rows
#X_train.shape[1] or len(X_train[0,:])              # gives you the number of columns
```

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

# of indicators

# of rows

# of Columns

Step 4. Building the RNN

```
Initiating the Model----------
regressor = Sequential()
First layer ----------
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))
Second Layer ----------
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
Third layer ----------
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
Last Layer ----------
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
Output----------
regressor.add(Dense(units = 1))
Complile----------
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
Fit & Run ----------
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

Step 5. Prediction

in column

```
dataset_total = pd.concat((dataset_train['Price'], dataset_test['price']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values      # recall, we chose TimeGroup=60
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 80):    # 80=60+ the length of the dqta which we want to predict "len(dataset_test)"
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

only transform

Please see the full code on Canvas/Concepts/07 RNN----------

Note: Tutorial 7. LSTM (Mon, Nov 7th) is an important Tutorial where you can learn more details on this topic.

Assignment 7 (the last one)is also a good practice on this topic.