Machine learning 09

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

Contents

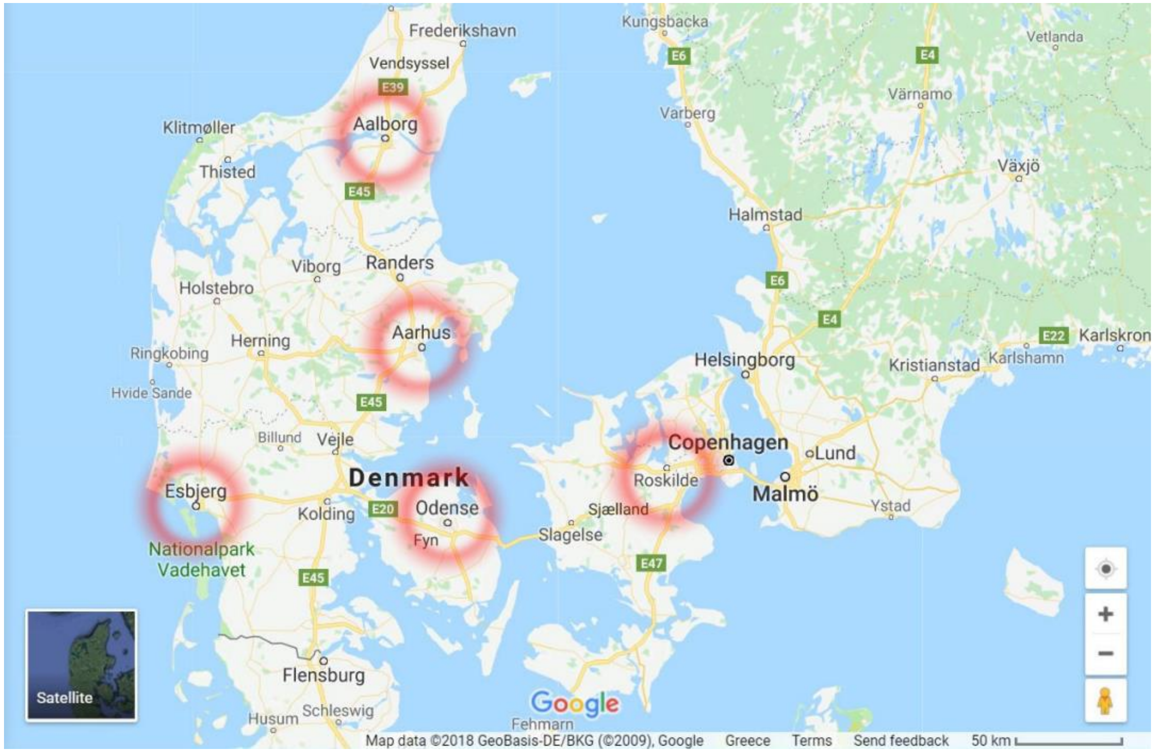
- Recurrent neural network (RNN)
  - time series data
  - recurrent neuron
  - introduction and training RNN
  - basic application
  - issues in RNN

Time series data

- data type
  - time series data, also referred to as time-stamped data, is a sequence of data points indexed in time order
  - data points typically consist of successive measurements made from the same source over a time interval

Time series data

- Weather information in Denmark



Gyeongsang National University 경상국립대학교

https://tykimos.github.io/warehouse/2018-5-16-ISS_Plant_DeepLearning_Model_in_SNRC_kbk_file.pdf

Time series data

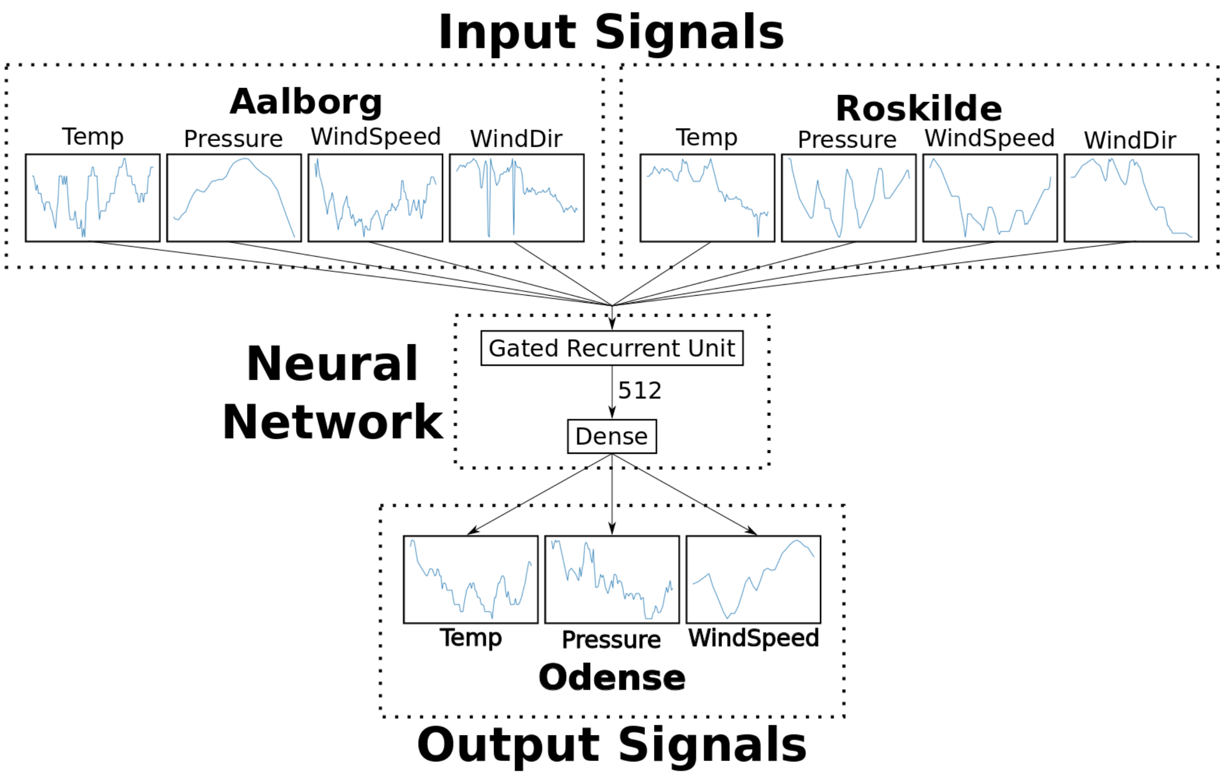
- Weather information in Denmark
  - we are trying to predict the weather for the Danish city "Odense" 24 hours into the future, given the current and past weather-data from 5 cities
  - we use a Recurrent Neural Network (RNN) because it can work on sequences of arbitrary length

Time series data

- Weather information in Denmark
  - during training we will use sub-sequences of 1344 data-points (8 weeks) from the training-set, with each data-point or observation having 20 input-signals for the temperature, pressure, etc. for each of the 5 cities
  - we then want to train the neural network so it outputs the 3 signals for tomorrow's temperature, pressure and wind-speed

Time series data

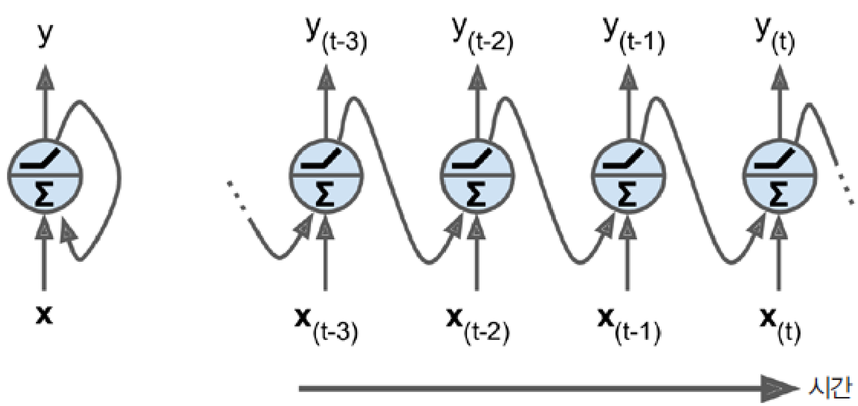
- Weather information in Denmark



Gyeongsang National University 경상국립대학교

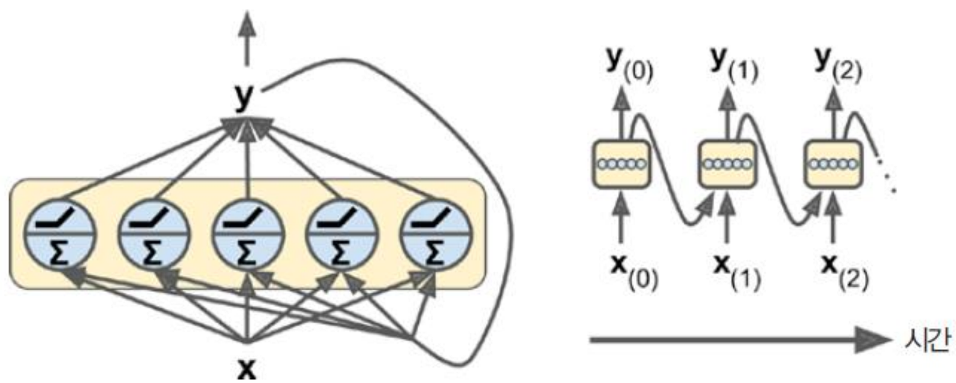
Recurrent neuron

- unrolling network through time
  - recurrent neuron is very similar to the feedforward neuron, but there are also backward recurrent connections



Recurrent neural network

- description for recurrent neuron



  - two types of weights
    - for input
    - for previous output

Gyeongsang National University 경상국립대학교

Recurrent neural network

- description for recurrent neuron
  - two types of weights
    - for input
    - for previous output

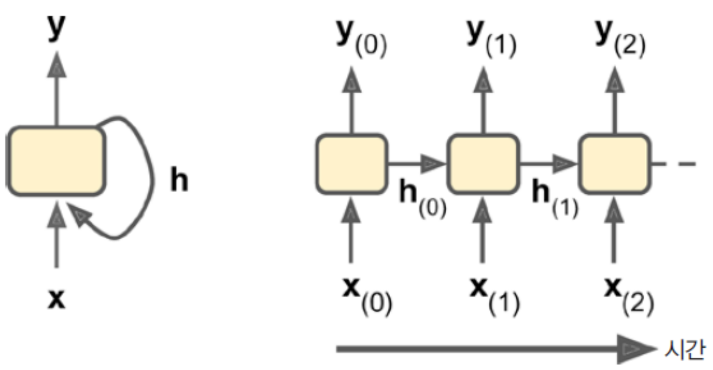
$$\mathbf{y}_{(t)} = \phi\left(\mathbf{W}_x{}^T \mathbf{x}_{(t)} + \mathbf{W}_y{}^T \mathbf{y}_{(t-1)} + \mathbf{b}\right)$$

$$\begin{aligned}\mathbf{Y}_{(t)} &= \phi\left(\mathbf{X}_{(t)} \mathbf{W}_x + \mathbf{Y}_{(t-1)} \mathbf{W}_y + \mathbf{b}\right) \\ &= \phi\left(\left[\mathbf{X}_{(t)} \ \mathbf{Y}_{(t-1)}\right] \mathbf{W} + \mathbf{b}\right) \text{ 여기에서 } \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_y \end{bmatrix}\end{aligned}$$

GNU Gyeongsang National University 경상국립대학교

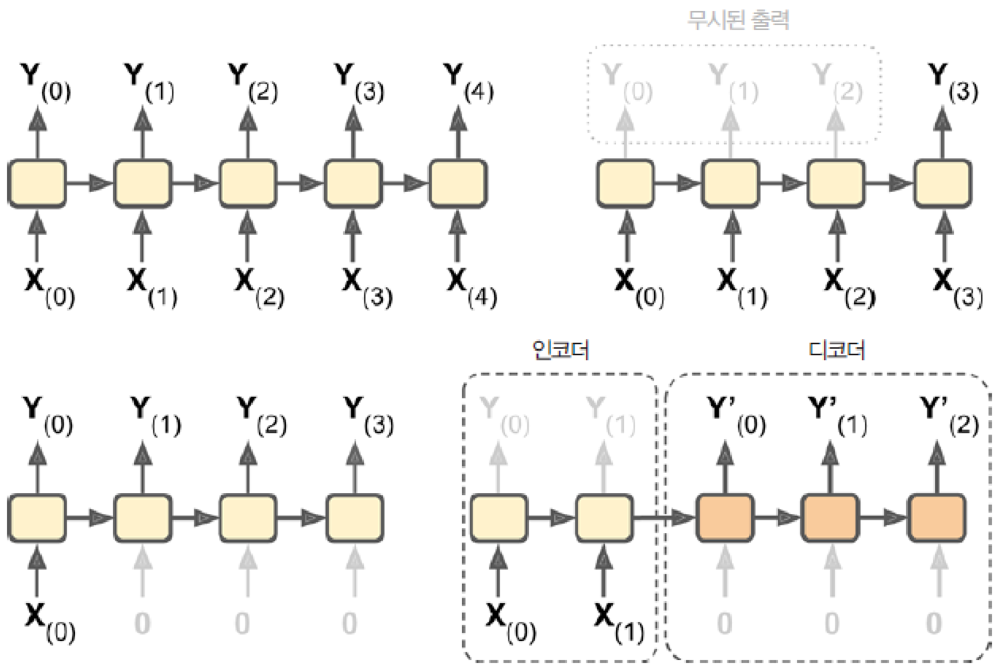
Recurrent neural network

- memory cell
  - remember previous results
  - basic form



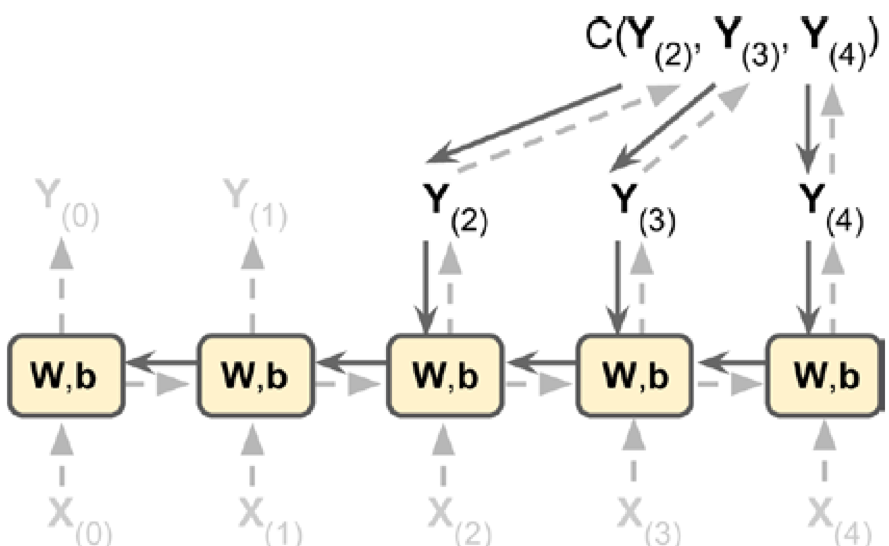
Recurrent neural network

- structure of memory cell
  - 4 types of network



Training recurrent neural network

- backpropagation through time (BPTT)
  - expand network with respect to time and use basic backpropagation

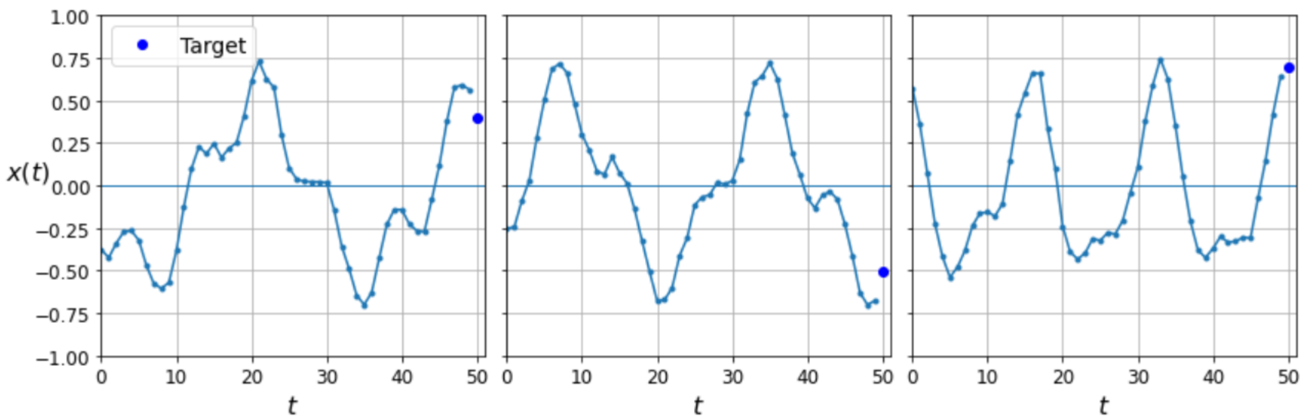


Gyeongsang National University 경상국립대학교

Forecasting time-series data

- basic example
  - making time-series data

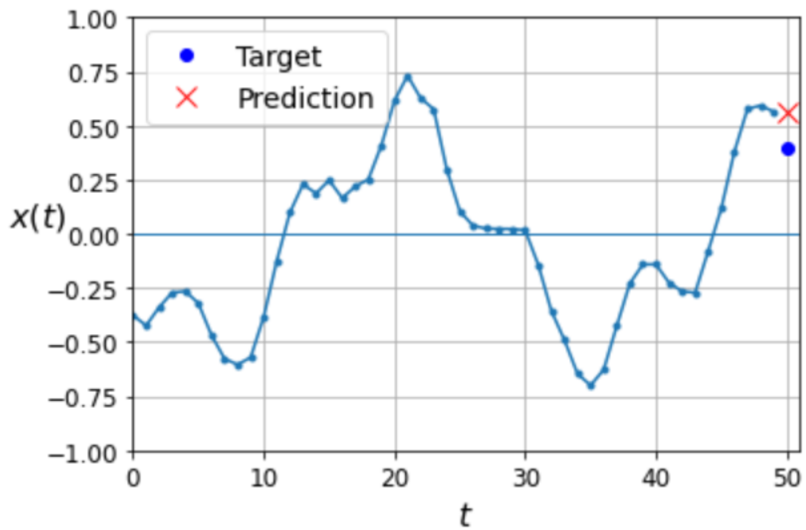
```
def generate_time_series(batch_size, n_steps):
    freq1, freq2, offsets1, offsets2 = np.random.rand(4, batch_size, 1)
    time = np.linspace(0, 1, n_steps)
    series = 0.5 * np.sin((time - offsets1) * (freq1 * 10 + 10))  #   웨이브 1
    series += 0.2 * np.sin((time - offsets2) * (freq2 * 20 + 20)) # + 웨이브 2
    series += 0.1 * (np.random.rand(batch_size, n_steps) - 0.5)   # + 잡음
    return series[..., np.newaxis].astype(np.float32)
```



Forecasting time-series data

- basic example
  - forecast using previous input

```
y_pred = X_valid[:, -1]
np.mean(keras.losses.mean_squared_error(y_valid, y_pred))
```

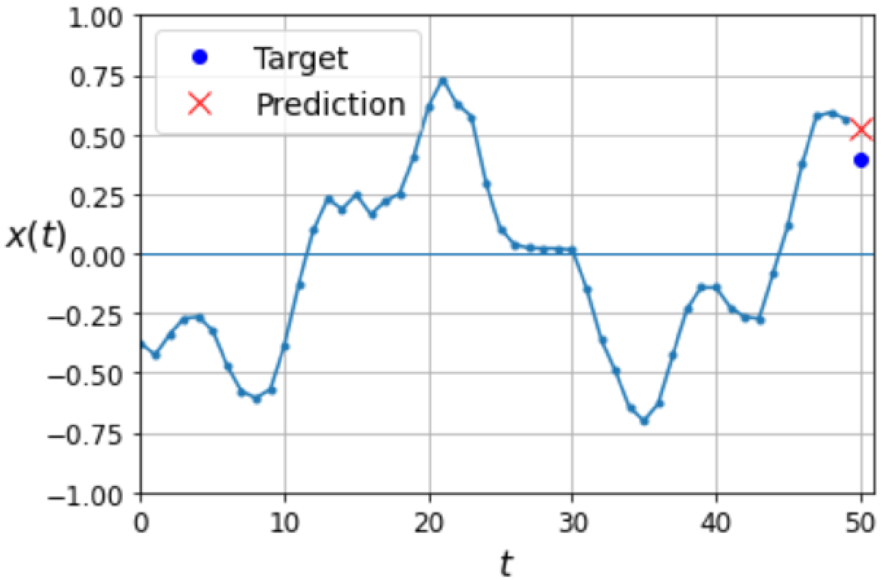


Gyeongsang National University 경상국립대학교

Forecasting time-series data

- basic example
  - forecast using typical neural network

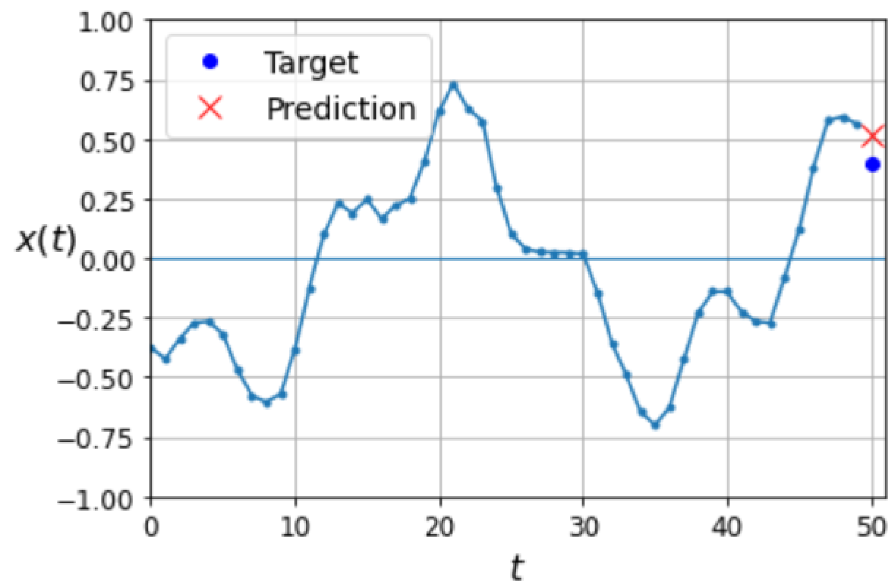
```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[50, 1]),
    keras.layers.Dense(1)
])

model.compile(loss="mse", optimizer="adam")
history = model.fit(X_train, y_train, epochs=20,
                    validation_data=(X_valid, y_valid))
```



Forecasting time-series data

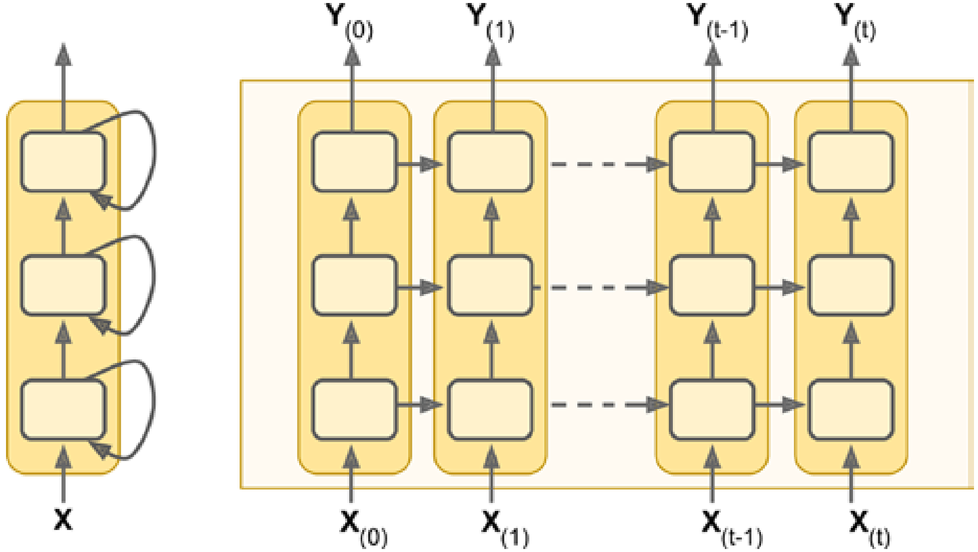
- basic example
  - forecast using simple RNN

```
model = keras.models.Sequential([
    keras.layers.SimpleRNN(1, input_shape=[None, 1])
])

optimizer = keras.optimizers.Adam(learning_rate=0.005)
model.compile(loss="mse", optimizer=optimizer)
history = model.fit(X_train, y_train, epochs=20,
                    validation_data=(X_valid, y_valid))
```



Forecasting time-series data

- Deep RNN
  - extension through accumulated memory cells
  - similar to the hidden layer in deep learning

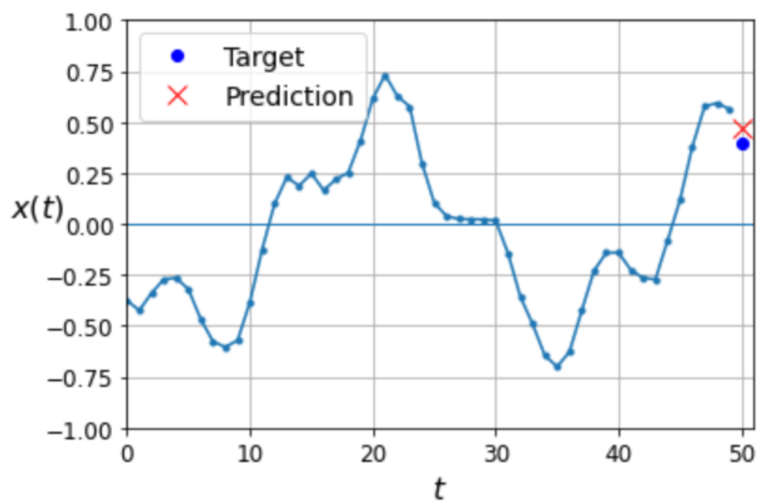


Gyeongsang National University 경상국립대학교

Forecasting time-series data

- applying deep RNN to basic example

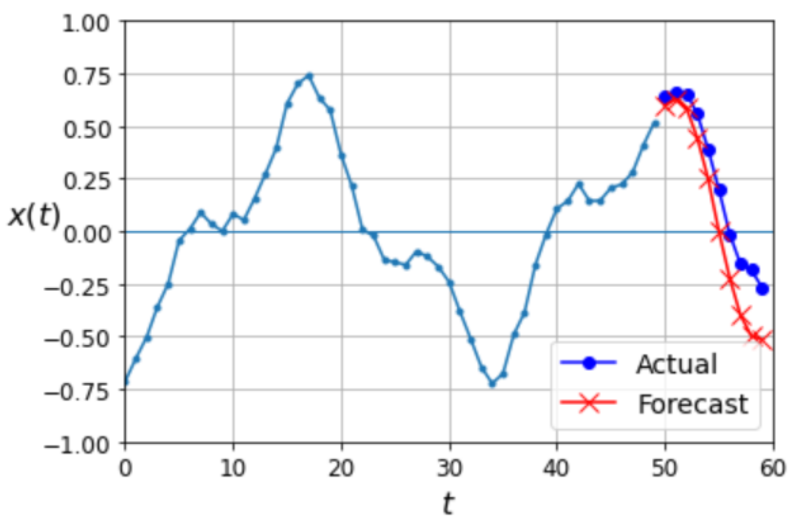
```
model = keras.models.Sequential([
    keras.layers.SimpleRNN(20, return_sequences=True, input_shape=[None, 1]),
    keras.layers.SimpleRNN(20, return_sequences=True),
    keras.layers.SimpleRNN(1)
])

model.compile(loss="mse", optimizer="adam")
history = model.fit(X_train, y_train, epochs=20,
                    validation_data=(X_valid, y_valid))
```



Forecasting time-series data

- forecasting multiple time steps

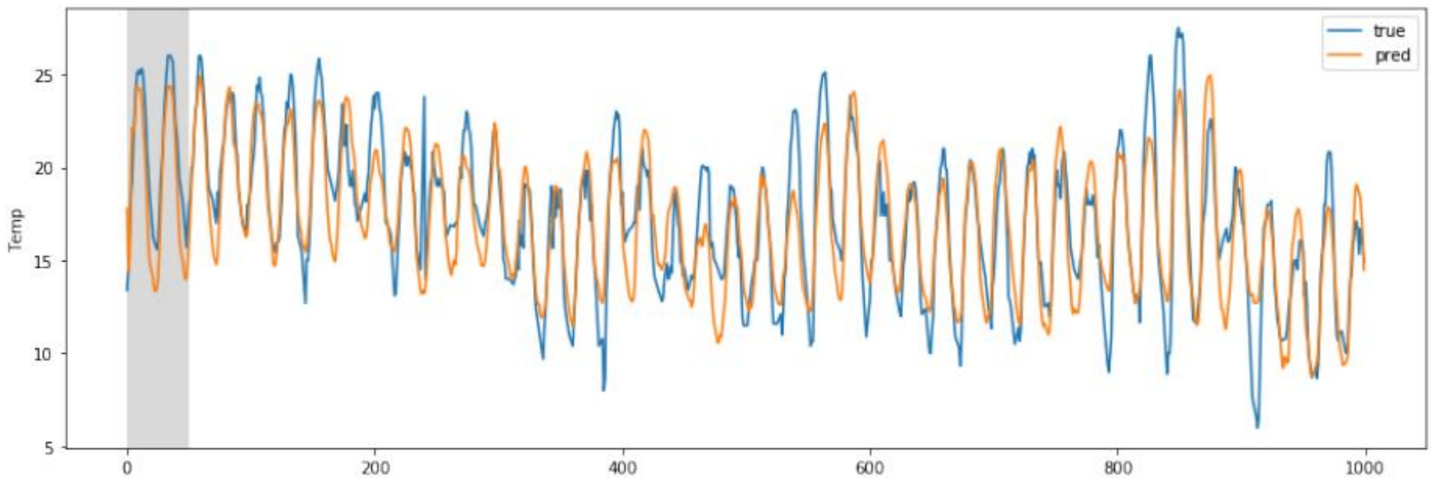
```
series = generate_time_series(1, n_steps + 10)
X_new, Y_new = series[:, :n_steps], series[:, n_steps:]
X = X_new
for step_ahead in range(10):
    y_pred_one = model.predict(X[:, step_ahead:])[:, np.newaxis, :]
    X = np.concatenate([X, y_pred_one], axis=1)

Y_pred = X[:, n_steps:]
```



Gyeongsang National University 경상국립대학교

Time series data revisited

- Weather forecast example
  - Refer to https://tykimos.github.io/warehouse/2018-5-16-ISS_Plant_DeepLearning_Model_in_SNRC_kbk_file.pdf

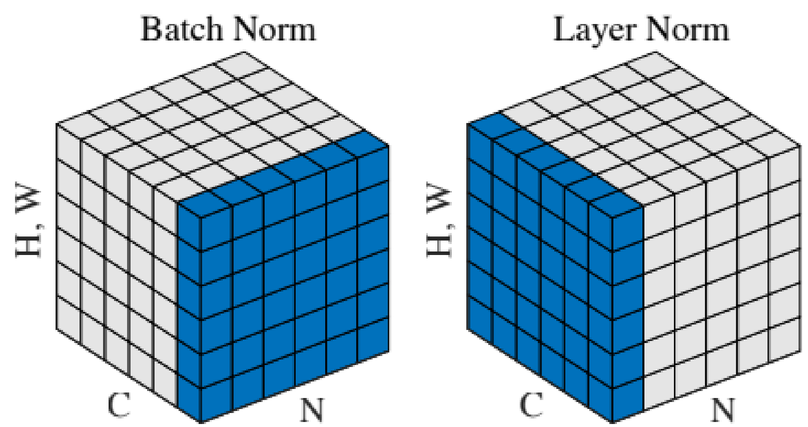


Issues in RNN

- Gradient explosion
  - repeated calculation in memory cell
  - modification in activation function
    - ReLU → hyperbolic tangent
  - gradient clipping

Issues in RNN

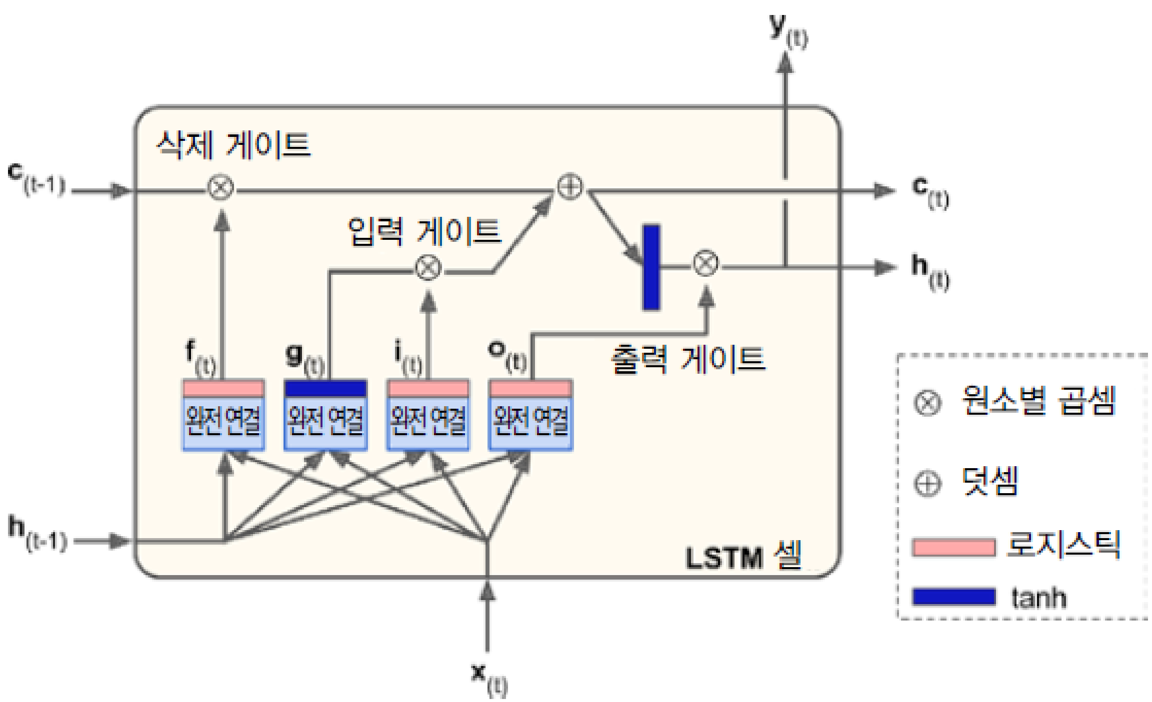
- Normalization
  - batch normalization
  - layer normalization



https://arxiv.org/pdf/1803.08494.pdf

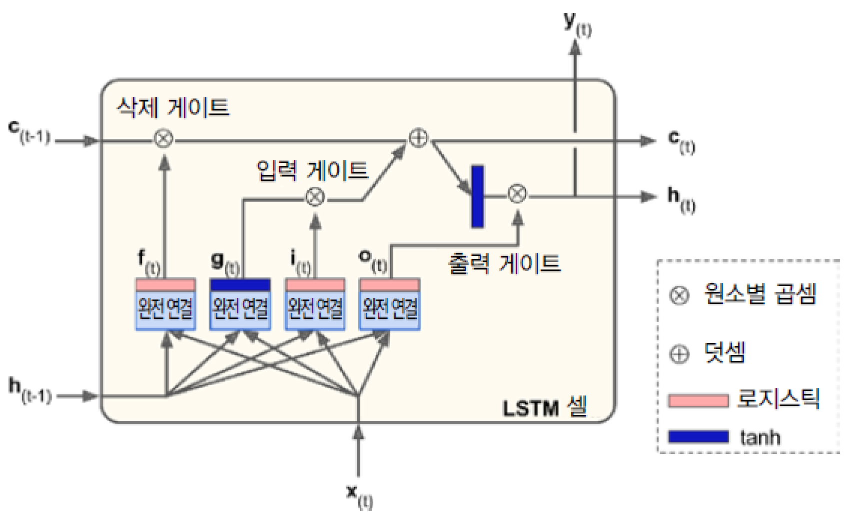
Expanded network

- Long short-term memory (LSTM)



Expanded network

- Long short-term memory (LSTM)
  - main layer: $g_{(t)}$
  - gate controller: $f_{(t)}, i_{(t)}, o_{(t)}$



$$\mathbf{i}_{(t)} = \sigma\left(\mathbf{W}_{xi}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{i}\right)$$

$$\mathbf{f}_{(t)} = \sigma\left(\mathbf{W}_{xf}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{f}\right)$$

$$\mathbf{o}_{(t)} = \sigma\left(\mathbf{W}_{xo}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{o}\right)$$

$$\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{g}\right)$$

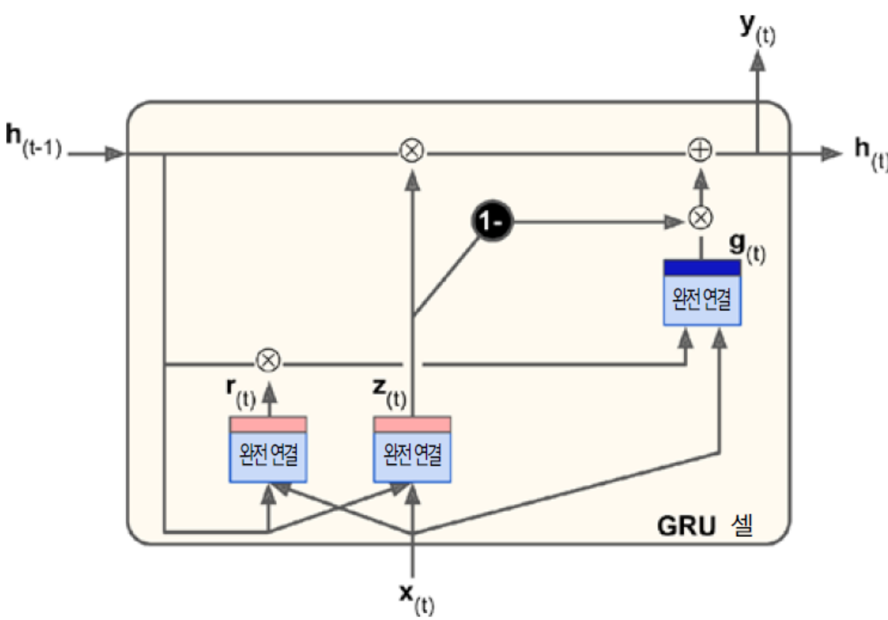
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\left(\mathbf{c}_{(t)}\right)$$

Gyeongsang National University 경상국립대학교

Expanded network

- Gated recurrent unit (GRU)



$$\mathbf{z}_{(t)} = \sigma\left(\mathbf{W}_{xz}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{z}\right)$$

$$\mathbf{r}_{(t)} = \sigma\left(\mathbf{W}_{xr}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{r}\right)$$

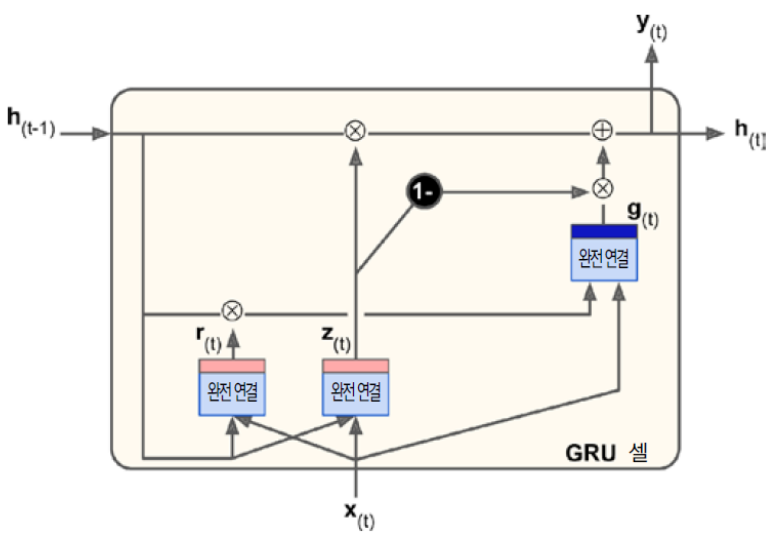
$$\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T}\left(\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}\right) + \mathbf{b}_{g}\right)$$

$$\mathbf{h}_{(t)} = \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + \left(1 - \mathbf{z}_{(t)}\right) \otimes \mathbf{g}_{(t)}$$

Gyeongsang National University 경상국립대학교

Expanded network

- Gated recurrent unit (GRU)
  - combined state variable: $h_{(t)}$
  - gate controller: $z_{(t)}$



$$\mathbf{z}_{(t)} = \sigma\left(\mathbf{W}_{xz}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{z}\right)$$

$$\mathbf{r}_{(t)} = \sigma\left(\mathbf{W}_{xr}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{r}\right)$$

$$\mathbf{g}_{(t)} = \tanh\left(\mathbf{W}_{xg}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T}\left(\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}\right) + \mathbf{b}_{g}\right)$$

$$\mathbf{h}_{(t)} = \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + \left(1 - \mathbf{z}_{(t)}\right) \otimes \mathbf{g}_{(t)}$$

Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함

Gyeongsang National University 경상국립대학교