LECTURE NOTE

2018 Fall Semester
11. 12. 2018

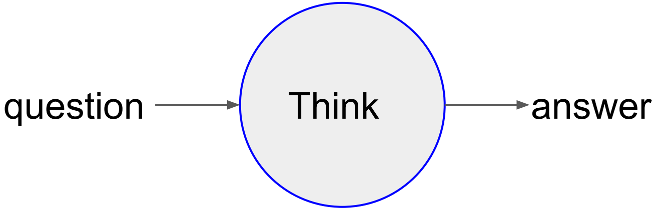
- Neural Network
- Supervised vs. Unsupervised Learning
- CNN
- RNN

Team 3: 20174252 김한솔 20186473 LIU LINGJUN 20183576 조경제 20174293 배병일

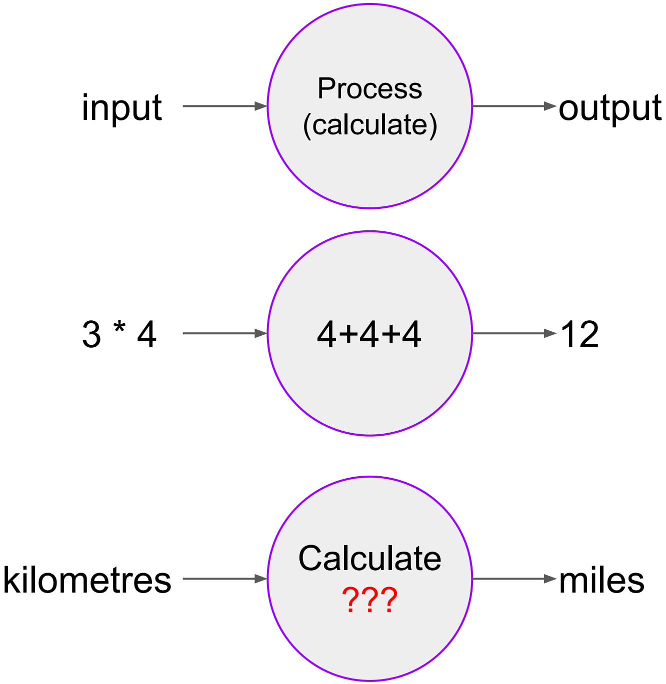
A Simple Predicting Machine

- Imagine a basic machine that takes a question, does some “thinking” and pushes out an answer

Human

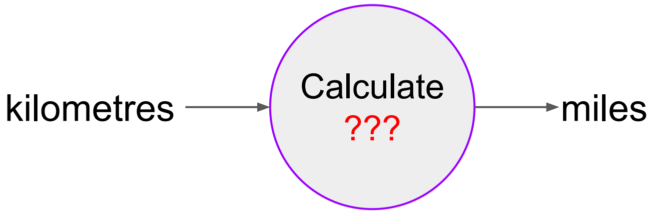


Computer

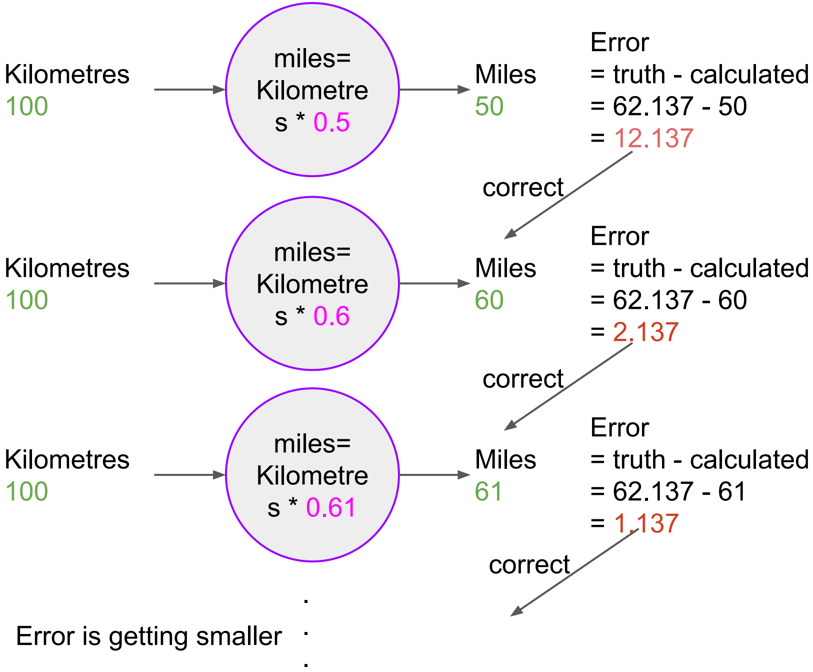


A Simple Predicting Machine

- Now imagine we don’t know the formula for converting between kilometres and miles, All we know is the relationship between the two is linear
- It needs to be of the form “miles = kilometres * **c**”

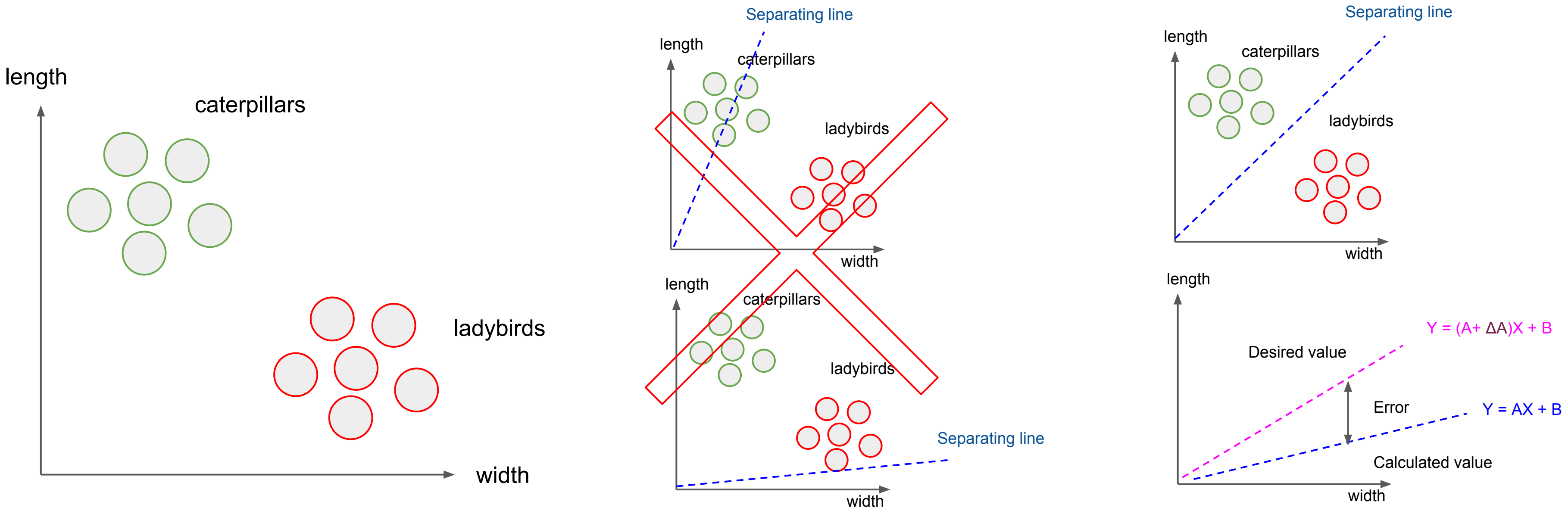


Truth Example	Kilometres	Miles
1	0	0
2	100	62.137



Classifying is Not very Different from Predicting

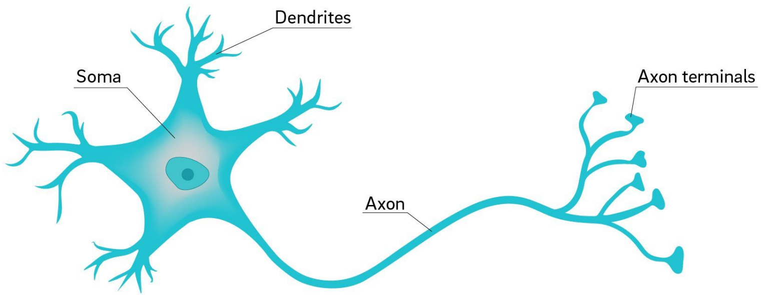
- That predictor had an adjustable linear function at it's heart. Remember, linear functions give straight lines when you plot their output against input. The adjustable parameter **C** changed the slope of that straight line
- Now look at the following graph showing the measured widths and lengths of garden bugs

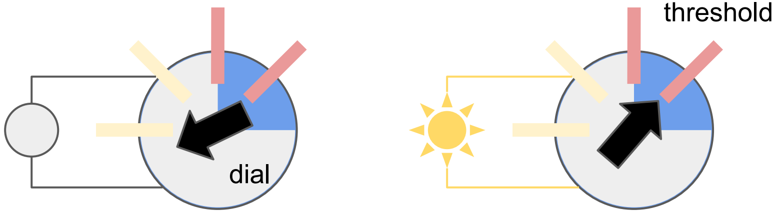


Neurons, Nature's Computing Machines

- Neurons all transmit an electrical signal forms one end to the other, form the dendrites along the axons to the terminals. These signals are then passed from one neuron to another.
- Observations suggest that neurons don't react readily, but instead suppress the input until it has grown so large that it triggers an output. You can think of this as a threshold that must be reached before any output is produced.

Neuron

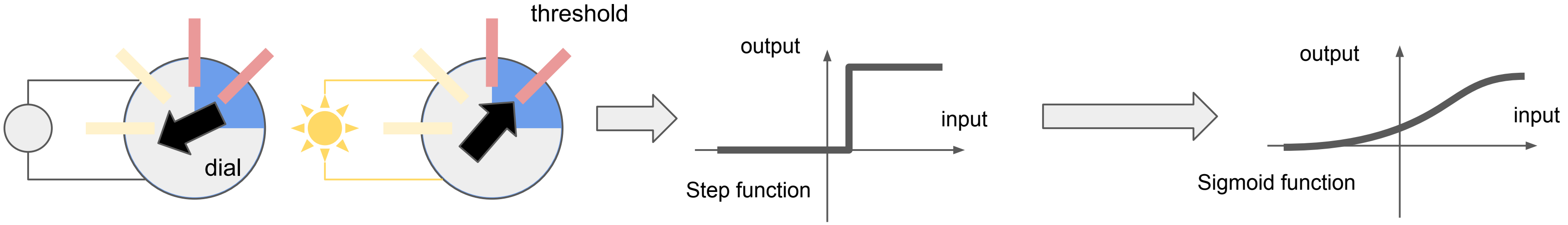




Neurons, Nature’s Computing Machines

- For low input values, the output is zero, However once the threshold input is reached, output jumps up. An artificial neuron behaving like this would be like a real biological neuron
- This smooth S-shaped sigmoid function is what we’ll be continue to use for making our own neural network. Artificial intelligence researchers will also use other, similar looking functions, but the sigmoid is simple and actually very common too.

This makes it more natural and realistic

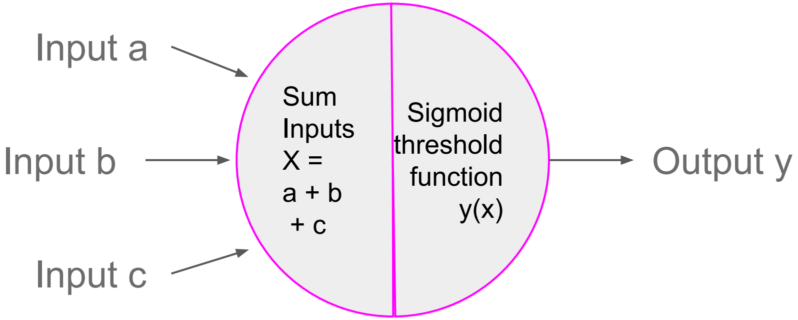


sometimes also called the logistic functions

$$y = \frac{1}{1 + e^{-x}}$$

Neurons, Nature’s Computing Machines

- Let’s get back to neurons, and consider how we might model an artificial neuron
- The first thing to realise is that real biological neurons take many inputs, not just one.
- Simply combine all input by adding them up, and the resultant sum is the input to the sigmoid function which controls the output
- If the combined signal is not large enough then the effect of the sigmoid threshold function is to suppress the output signal. If the sum **x** if large enough the effect of the sigmoid is to fire the neuron
- The electrical signals are collected by the **dendrites** and these combine to form an stronger electrical signal. If the signal is strong enough to pass the threshold, the neuron fires a signal down the **axon** towards the **terminals** to pass onto the **next neuron’s dendrites.**



General Procedure to Train a NN

1. Pre-training
   a. Train a sequence of shallow autoencoders
   b. Greedily one layer at a time
2. Fine-tuning (step1)
   a. Train the last layer **using supervised data**
3. Fine-tuning (step2)
   a. Use backpropagation to fine-tune the entire network **using supervised data**

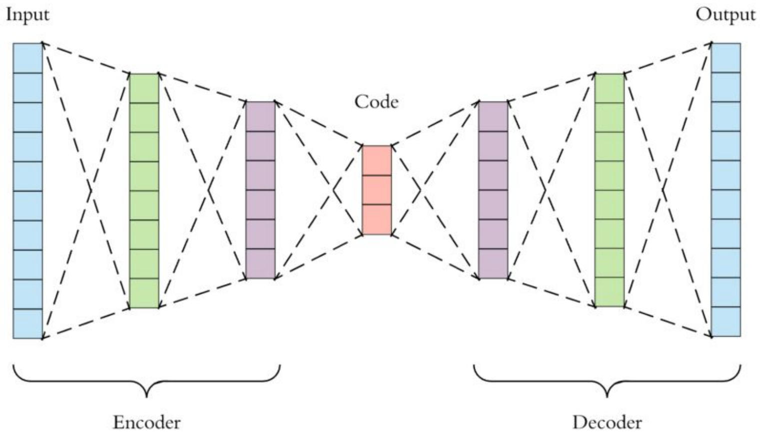


Figure 1 - An example of encoder-decoder

Machine Learning

Objectives: Build a neural network that maps input to output

Two main types of machine learning tasks

1. Supervised learning
   a. Uses datasets to fine-tune the model
   b. Use cases: user analytic, info and cyber security, asset management, stock exchange
2. Unsupervised learning
   a. **Does not** use other datasets to fine-tune the model
   b. Use cases: clustering, representation learning, and density estimation

Convolutional Neural Network (CNN)

Features

- Each edge has weights
- Many connections within the network
- Commonly used for image recognition, classifications, detections tasks

➔ **Fully Connected Layer** - connect all inputs previous layer to next layer.

➔ **Convolutional Layer(Locally connected network)** - each neuron is connected only to that of adjacent pixel

- ◆ Also called as “local receptive field”

➔ **Max-pooling Layer** - take the largest element from each of a cluster of neurons at previous layer

- ◆ Output of max-pooling neurons are invariant to shifts in the inputs(Translation invariance)

Convolutional Neural Network (CNN)

- General procedure
  - Takes an input image
  - Process the image
  - Classify the image under certain categories (e.g. animals, cars, etc.)

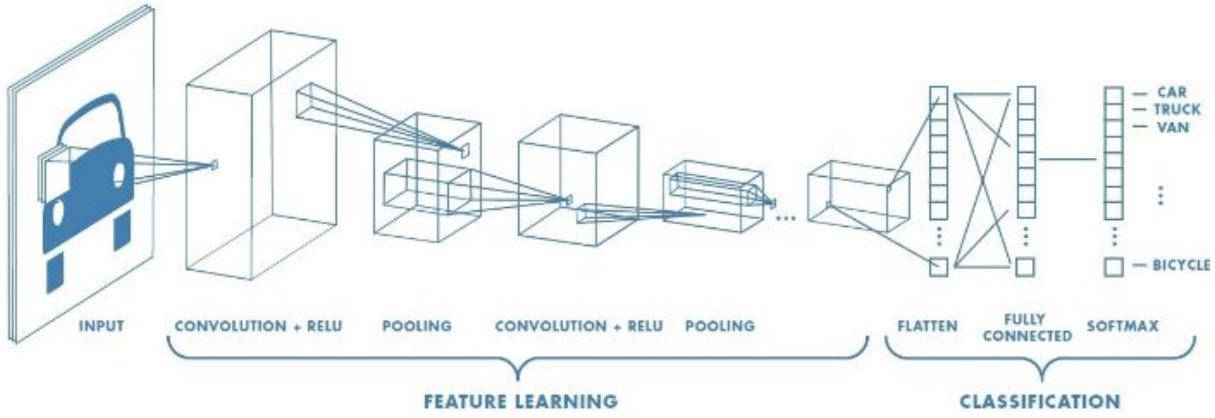
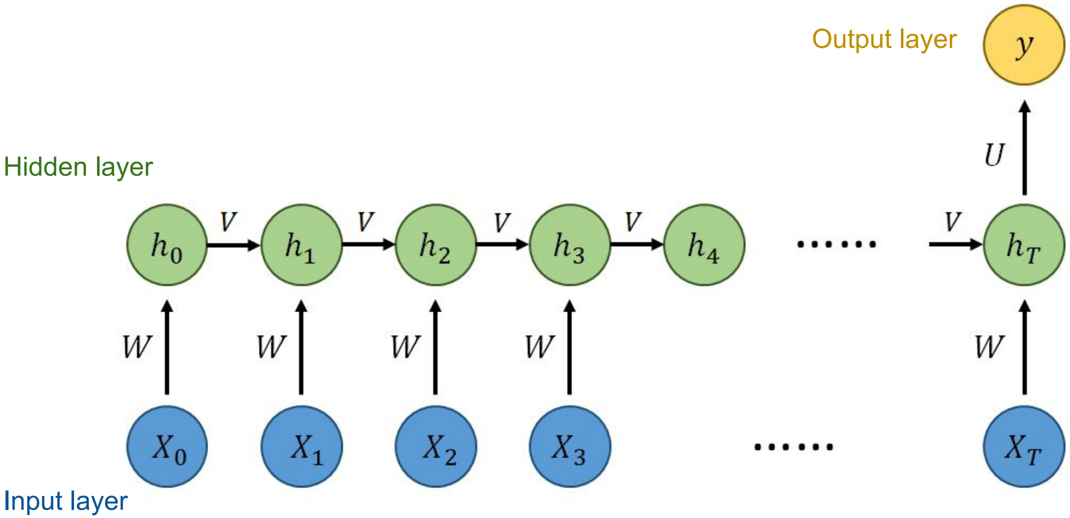


Figure 2 - An example of CNN with many convolutional layers

Recurrent Neural Network (RNN)

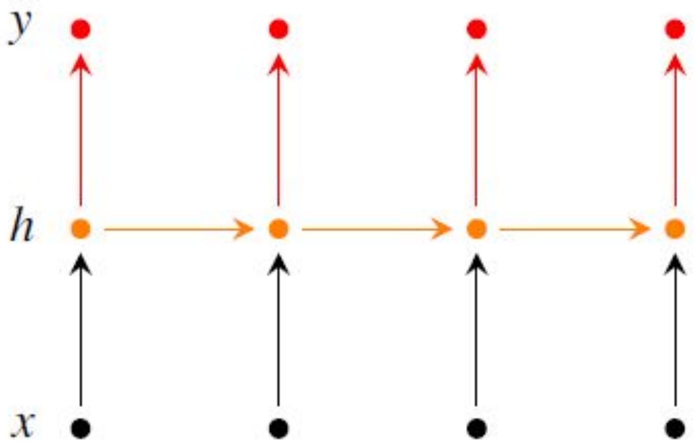
- Designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario
- E.g. predict tomorrow’s stock price



$X_0,\ldots X_T$: stock prices up to today
$h_0,\ldots h_T$: hidden states. h is calculated based on the previous hidden state and the input at the current step
W : input to hidden weights
V : hidden to hidden weights
U : hidden to output weights
y : tomorrow’s stock price

Recurrent Neural Network (RNN)

Each node in a given layer is connected with a directed connection to every other node in the next layer.

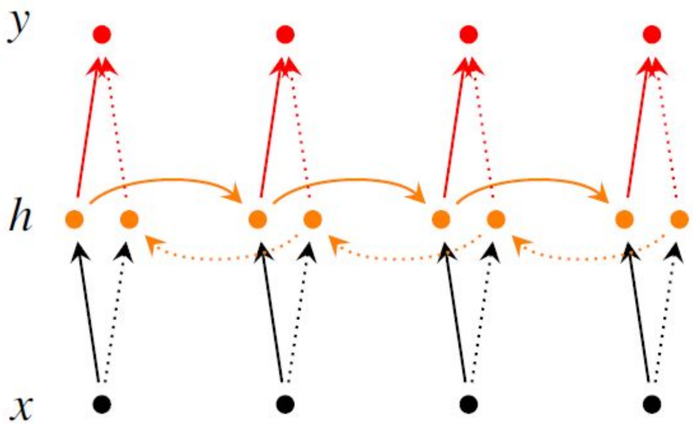


$$h_t = f(Wx_t + Vh_{t-1} + b)$$

$$y_t = g(Uh_t + c)$$

Bidirectional Recurrent Neural Network(BRNN)

- Use a finite sequence to predict or label each element of the sequence based on the element’s past and future contexts.
- Especially useful when combined with LSTM



$$\overrightarrow{h}_t = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

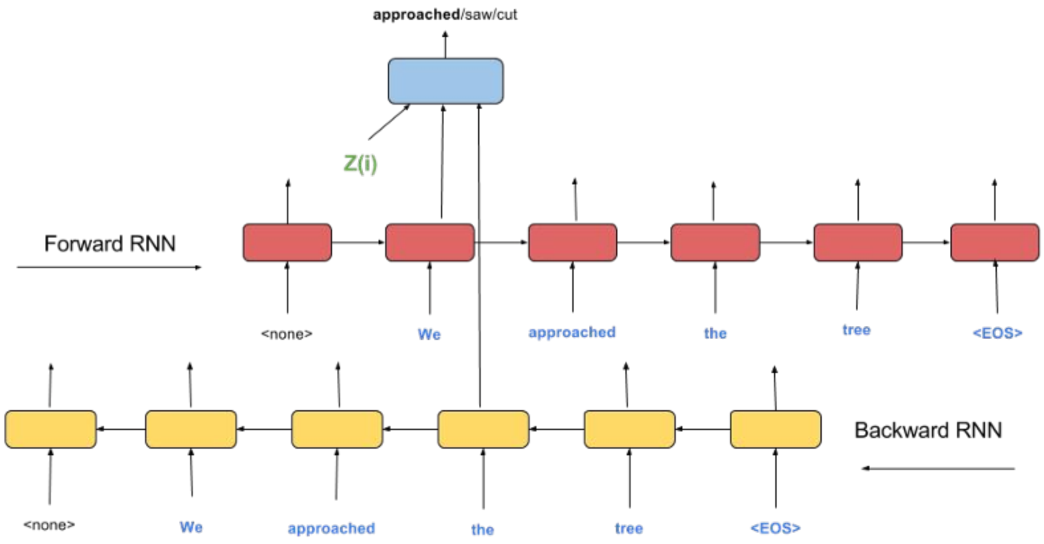
$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t]+c)$$

Example of Bidirectional RNN

Sentence: “We approached the tree.”

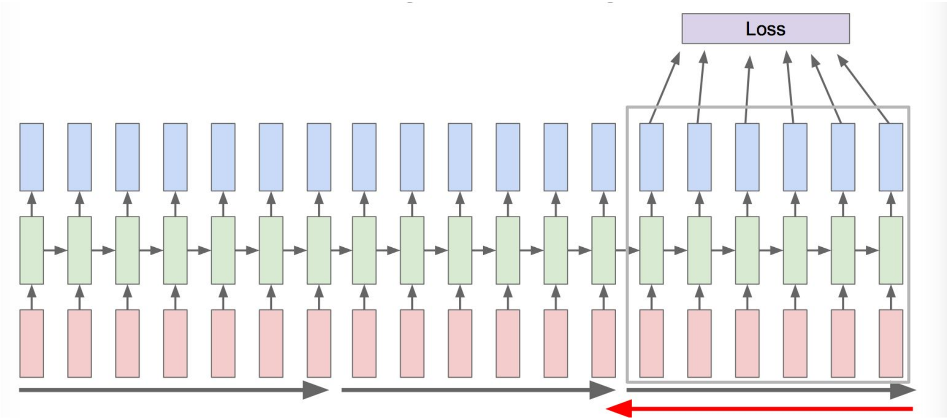
Predicts “approached” using

- “we” from forward RNN
- “the” from backward RNN



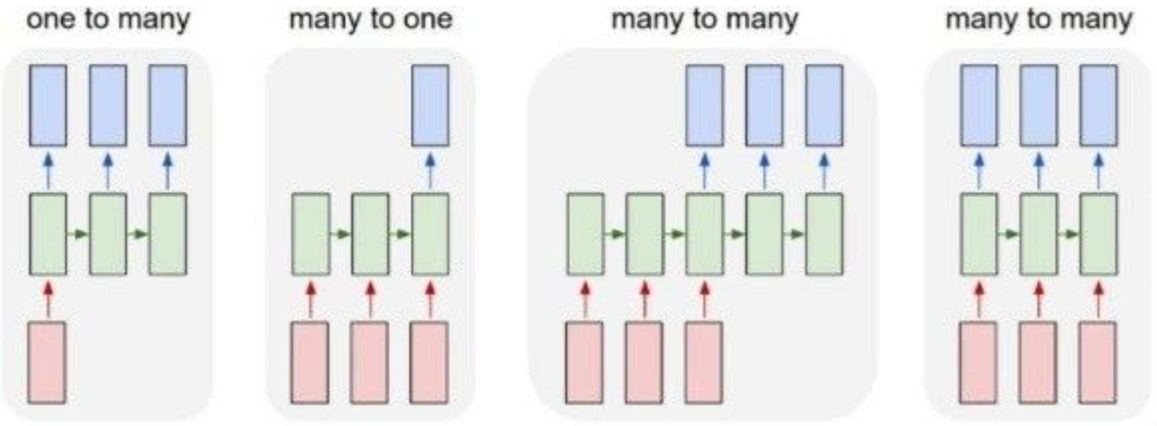
Backpropagation through time

- Forward through entire sequence to compute loss, and then backward through entire sequence to compute gradient.
- Usually use **Truncated BTT**
  - Run forward and backward through chunks of sequence instead of whole sequence
- Problem : When dealing with a time series, it tends to forget old information. When there is a distant relationship of unknown length, we want to have a **memory** to it.



RNN applications

➔ Speech recognition, Music generation, Sentiment classification, DNA sequence analysis, Machine translation, Video activity recognition and Name entity recognition



Music generation

Sentiment classification

Machine translation

Name entity recognition