Sequence Models

Bùi Tiến Lên

2022

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

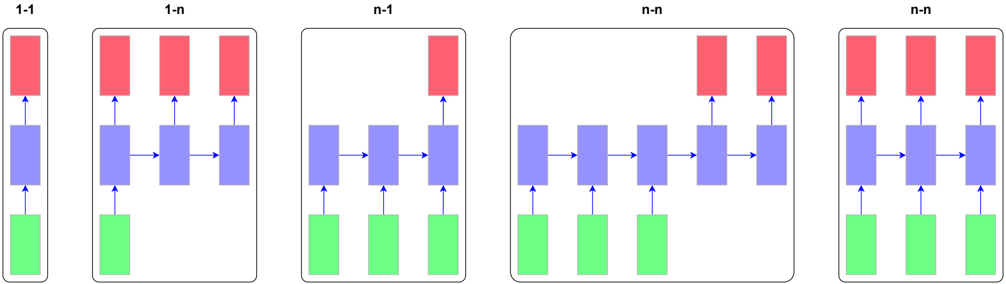
Notation

symbol	meaning
$a, b, c, N \ldots$	scalar number
$\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$	column vector
$\boldsymbol{X}, \boldsymbol{Y} \ldots$	matrix
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^D$	set of vectors
$\mathcal{X}, \mathcal{Y}, \ldots$	set
$\mathcal{A}$	algorithm

operator	meaning
$\boldsymbol{w}^\top$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse
$\boldsymbol{X} \odot \boldsymbol{Y}$	an element-wise matrix-vector multiplication

Sequential prediction tasks

Recurrent Neural Networks: Process Sequences



- Model 1-1: e.g. Image classification
- Model 1-n: e.g. Image captioning
- Model n-1: e.g. Sentiment classification
- Model n-n: e.g. Machine translation
- Model n-n: e.g. Intellisense

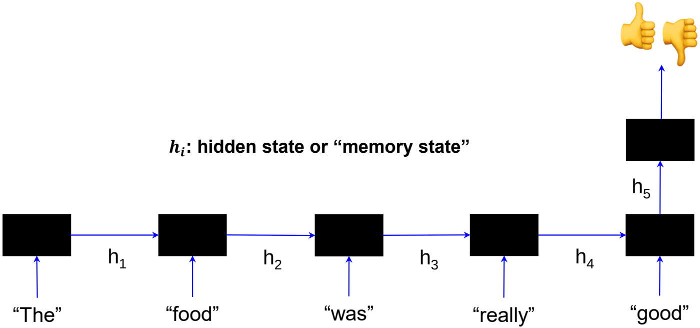
Example 1: Sentiment classification

- Goal: classify a text sequence (e.g., restaurant, movie or product review, Tweet) as having positive or negative sentiment

  "The food was really good"

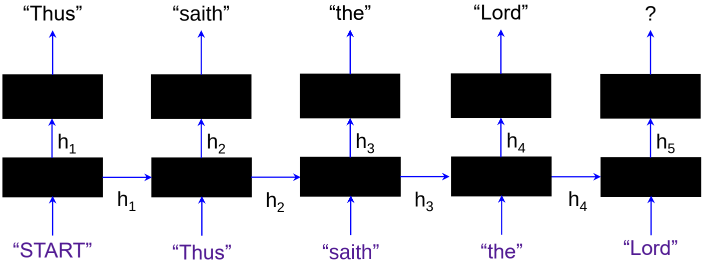
  "The vacuum cleaner broke within two weeks"

  "The movie had slow parts, but overall was worth watching"



Example 2: Text generation

- Goal: Sample from the distribution of a given text corpus (also known as language modeling)



Example 3: Image caption generation

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

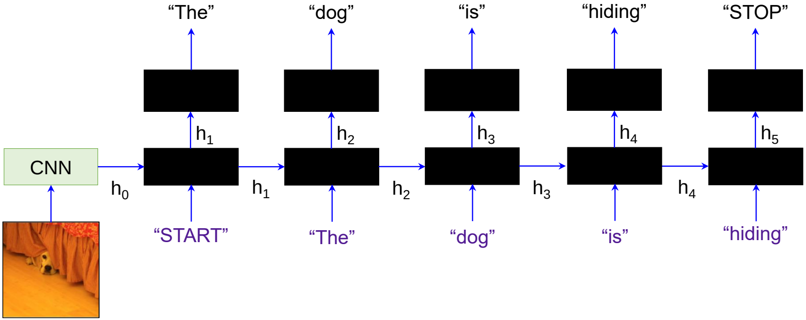
Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

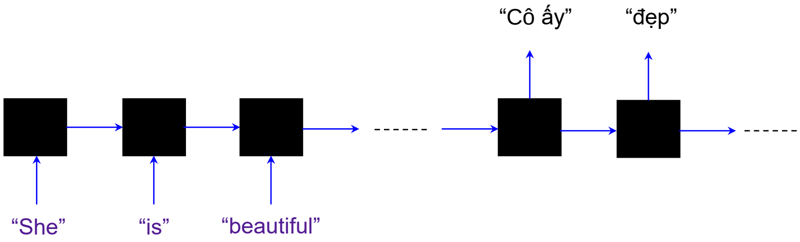
A man riding a dirt bike on a dirt track

Example 3: Image caption generation (cont.)



Example 4: Machine translation

- Translate English to Vietnamese

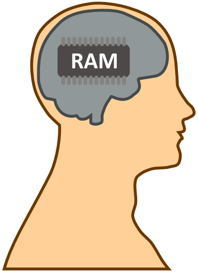


Recurrent Network Network

- Architecture
- Training
- Learning problems

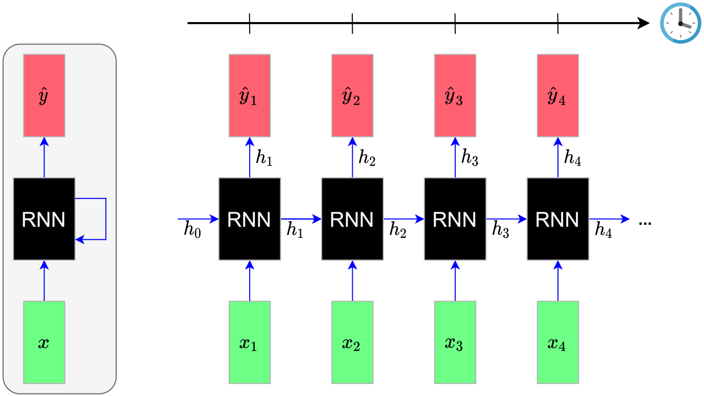
Human thoughts are persistence

- Humans don't start their thinking from scratch. They usually use their prior knowledge or experiences.



Recurrent Neural Network

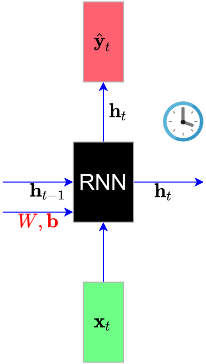
- **Key idea**: RNNs have an "internal state" h_t that is updated as a sequence is processed



Model **Run**

Vanilla RNN unit

- The **state** consists of a single "hidden" vector $\boldsymbol{h}_t$:



- Full formula

$$\begin{cases} \boldsymbol{h}_t = f_{W,\boldsymbol{b}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) \\ \hat{\boldsymbol{y}}_t = g_{W,\boldsymbol{b}}(\boldsymbol{h}_t) \end{cases} \begin{cases} \boldsymbol{h}_t = \tanh(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t + \boldsymbol{b}_h) \\ \hat{\boldsymbol{y}}_t = W_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y \end{cases} \tag{1}$$

where **parameters** $W = (W_{xh}, W_{hh}, W_{hy})$ and $\boldsymbol{b} = (\boldsymbol{b}_h, \boldsymbol{b}_y)$

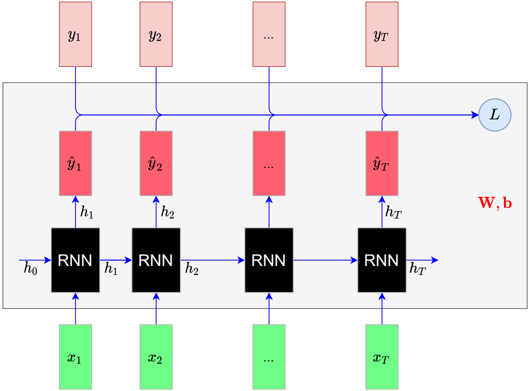
- Simple formula without bias vector

$$\begin{cases} \boldsymbol{h}_t = \tanh(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t) \\ \hat{\boldsymbol{y}}_t = W_{hy}\boldsymbol{h}_t \end{cases} \tag{2}$$

Cost function

- Given a sequence $\{(x_{1,}y_1),\ldots,(x_T,y_T)\}$, the lost function L is defined by

$$L(x_1,\ldots,x_T,y_1,\ldots,y_T \mid W,\mathbf{b}) = \sum_{t=1}^{T} l(y_t,\hat{y}_t) \quad (3)$$



RNN forward

Given a sequence data$\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_T, \boldsymbol{y}_T)\}$

- For each $t = 1 \ldots T$, compute using simple formula

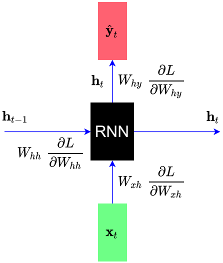
$$\begin{cases} \boldsymbol{h}_t = \tanh(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t) \\ \hat{\boldsymbol{y}}_t = W_{hy}\boldsymbol{h}_t \end{cases}$$

- Compute the lost function

$$L(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T \mid W) = \sum_{t=1}^{T} l(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$$

RNN backward

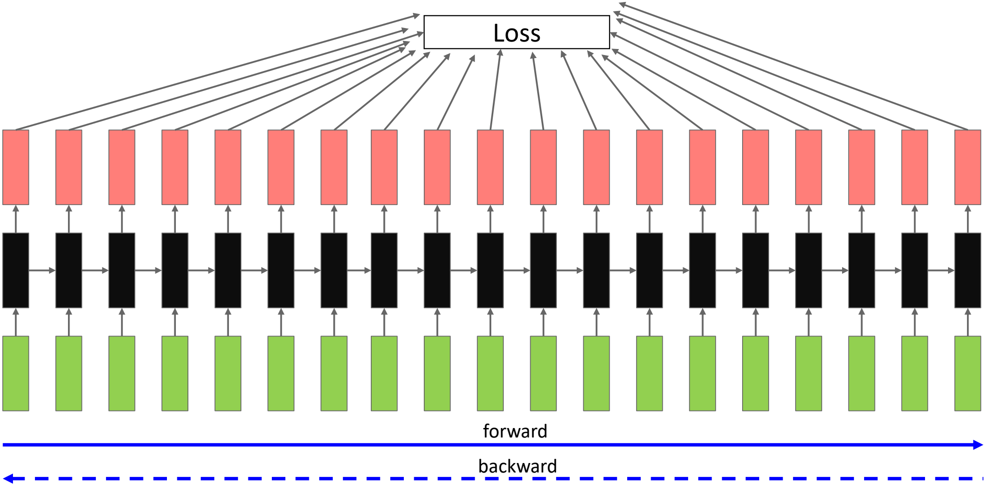
- For each RNN unit



$$
\begin{aligned}
\frac{\partial L}{\partial W_{hh}} &= \frac{\partial L}{\partial \boldsymbol{h}_t} \odot (1 - \tanh^2(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t))\boldsymbol{h}_{t-1}^\mathsf{T} \\
\frac{\partial L}{\partial W_{xh}} &= \frac{\partial L}{\partial \boldsymbol{h}_t} \odot (1 - \tanh^2(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t))\boldsymbol{x}_t^\mathsf{T} \\
\frac{\partial L}{\partial W_{hy}} &= ? \\
\frac{\partial L}{\partial \boldsymbol{h}_{t-1}} &= W_{hh}^\mathsf{T}(1 - \tanh^2(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t)) \odot \frac{\partial L}{\partial \boldsymbol{h}_t}
\end{aligned} \tag{4}
$$

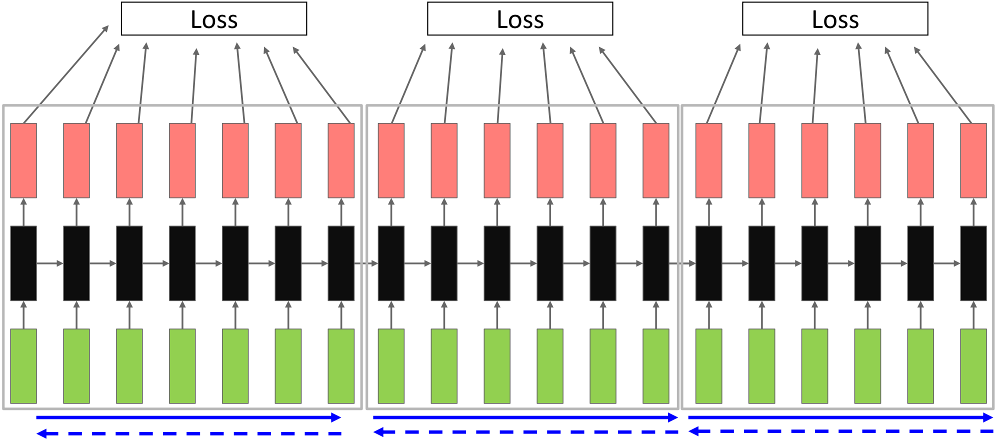
Backpropagation through time (BPTT)

- Problem: Takes a lot of memory for long sequences!



Truncated backpropagation through time

- In practice, truncated BPTT is used: run the RNN forward k time steps, propagate backward for k time steps



Long-term dependencies

Vanilla RNNs trained with BPTT have difficulties learning long-term dependencies.

- **Able** when the **gap** between the relevant information is **small**.
- As that gap grows, **unable** to learn to connect the information.

Vanishing/Exploding gradients

Computing gradient of h_0 involves many factors of W (and repeated tanh)

- Largest singular value > 1: **Exploding gradients**
  Gradient clipping: Scale gradient $\boldsymbol{g}$ if its norm is too big

$$\boldsymbol{g} \leftarrow \min\left(1, \frac{\text{threshold}}{\|\boldsymbol{g}\|}\right)\boldsymbol{g} \tag{5}$$

- Largest singular value < 1: **Vanishing gradients**
  Change RNN architecture

Modern Recurrent Neural Networks

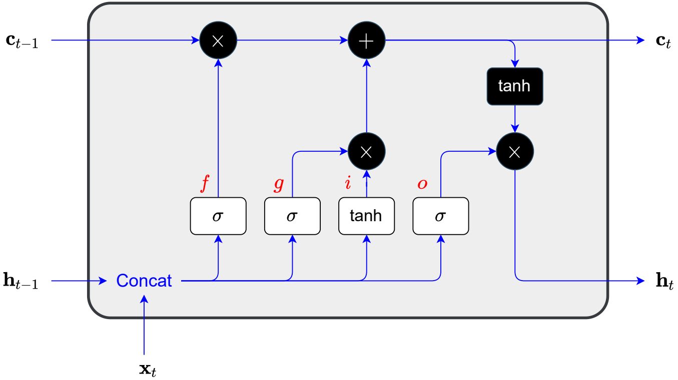
- Long Short Term Memory (LSTM)
- Gated Recurrent Units (GRU)
- Deep Recurrent Neural Networks
- BidirectionalRecurrent Neural Networks
- Encoder-Decoder Architecture

Key Concepts

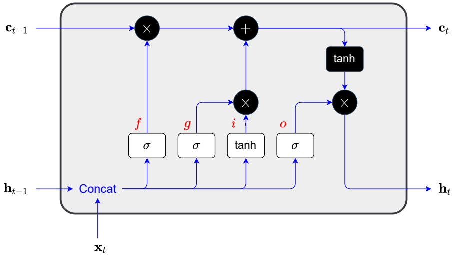
1. Maintain a **separate cell state** c_t from what is outputted
2. Use **gates** to control the **flow of information**
   - Forget gate gets rid of irrelevant information
   - Selectively update cell state
   - Output gate returns a filtered version of the cell state
3. Backpropagation from c_t to c_{t-1} doesn't require matrix multiplication $\rightarrow$ avoid vanishing gradient problem (**uninterrupted gradient flow**)

Long Short Term Memory Unit

- i (input gate): Whether to write to cell?
- f (forget gate): Whether to erase cell?
- o (output gate): How much to reveal cell?
- g (candiate gate): How much to write to cell?



Long Short Term Memory Unit (cont.)



$$\begin{pmatrix} \boldsymbol{i}_t \\ \boldsymbol{f}_t \\ \boldsymbol{o}_t \\ \boldsymbol{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{pmatrix} + \boldsymbol{b}_h \right) \tag{6}$$

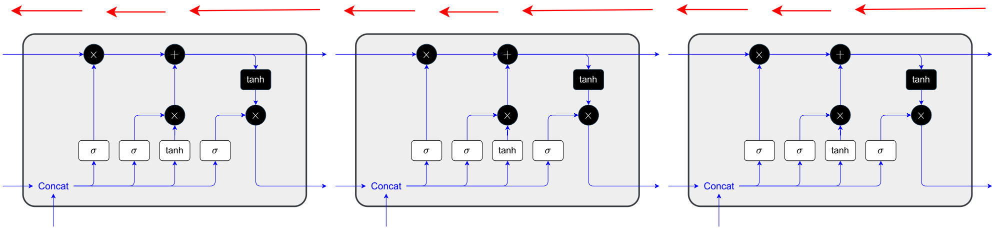
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)$$

Gradient flow

- Similar to ResNet

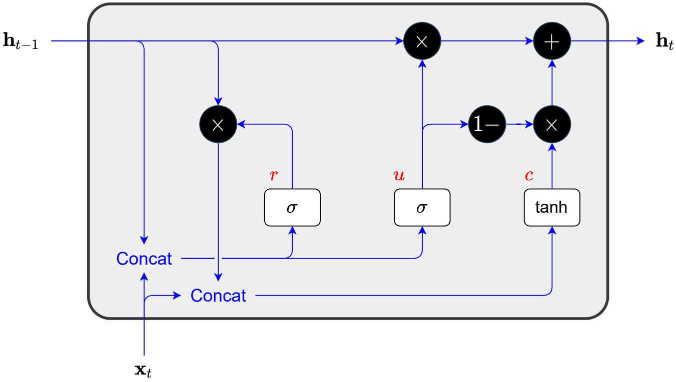
Uninterrupted gradient flow



GRU: Key Concepts

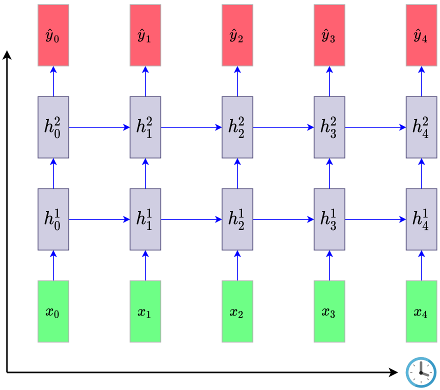
GRUs get rid of separate cell states; only use:

- **Reset gates** r help capture short-term dependencies in sequences.
- **Update gates** u help capture long-term dependencies in sequences.
- **Candiate gates** c



Deep Recurrent Neural Networks

- Multilayer LSTMs



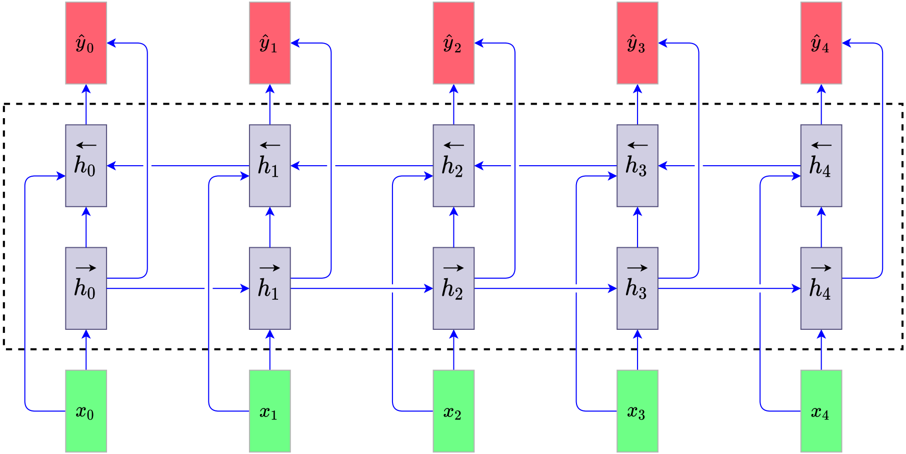
$$\begin{pmatrix} \boldsymbol{i}_t^\ell \\ \boldsymbol{f}_t^\ell \\ \boldsymbol{o}_t^\ell \\ \boldsymbol{g}_t^\ell \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} \boldsymbol{h}_{t-1}^\ell \\ \boldsymbol{h}_t^{\ell-1} \end{pmatrix} + \boldsymbol{b}_h^\ell \right)$$

$$\boldsymbol{c}_t^\ell = \boldsymbol{f}_t^\ell \odot \boldsymbol{c}_{t-1}^\ell + \boldsymbol{i}_t^\ell \odot \boldsymbol{g}_t^\ell$$

$$\boldsymbol{h}_t^\ell = \boldsymbol{o}_t^\ell \odot \tanh(\boldsymbol{c}_t^\ell)$$

Bidirectional Model

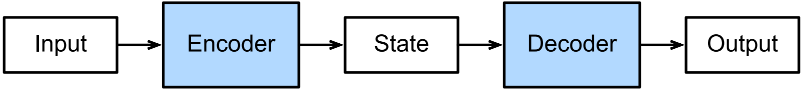
- Bidirectional RNNs add a hidden layer that passes information in a backward direction



Encoder-Decoder Architecture

A encoder-decoder architecture includes two major components

- The first component is an **encoder**: it takes a variable-length sequence as the input and transforms it into a state with a fixed shape.
- The second component is a **decoder**: it maps the encoded state of a fixed shape to a variable-length sequence.



Visualizing and Understanding

Applications

- Sequence classification
- Language modeling
- Image captioning
- Machine translation

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.