CS 360: Machine Learning

Sara Mathieson, Sorelle Friedler

Spring 2024

HAVERFORD COLLEGE

Admin

- **Lab 7** due TODAY!
- Sorelle/Sara office hours **today 4-5pm (H110)**
- **Project proposal** due Monday
  - Email me by *Friday at midnight* for a random partner

Outline for April 4

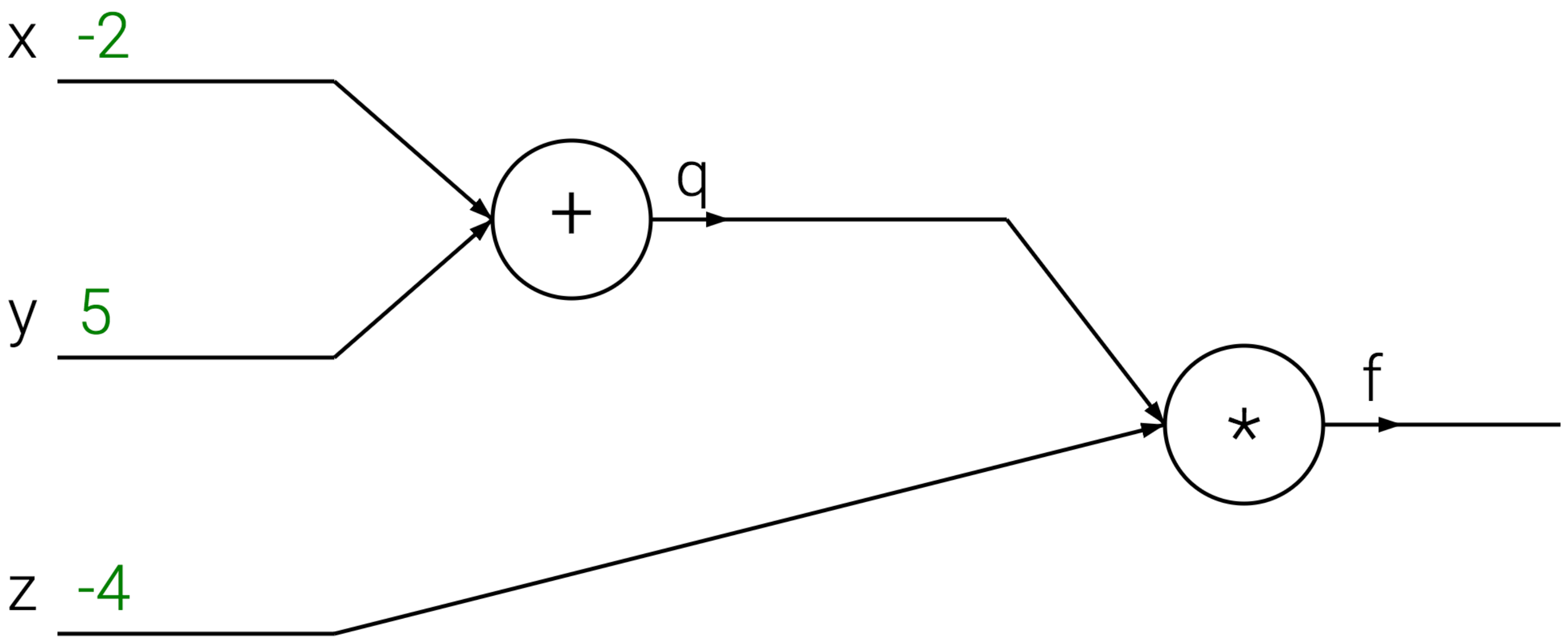
- Finish Backpropagation
- Recurrent neural networks
- Attention mechanisms
- Applications
- Transformers

Outline for April 4

- Finish Backpropagation
- Recurrent neural networks
- Attention mechanisms
- Applications
- Transformers

Backpropagation: Example

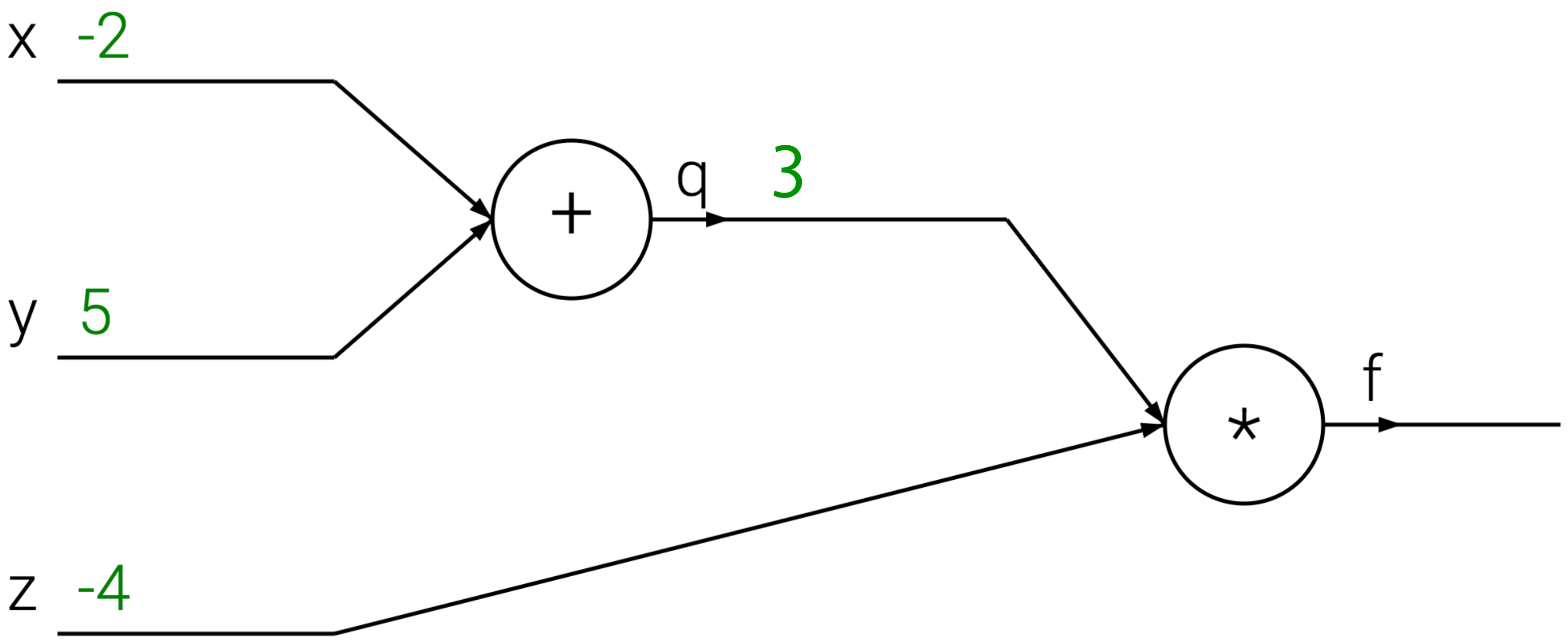
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

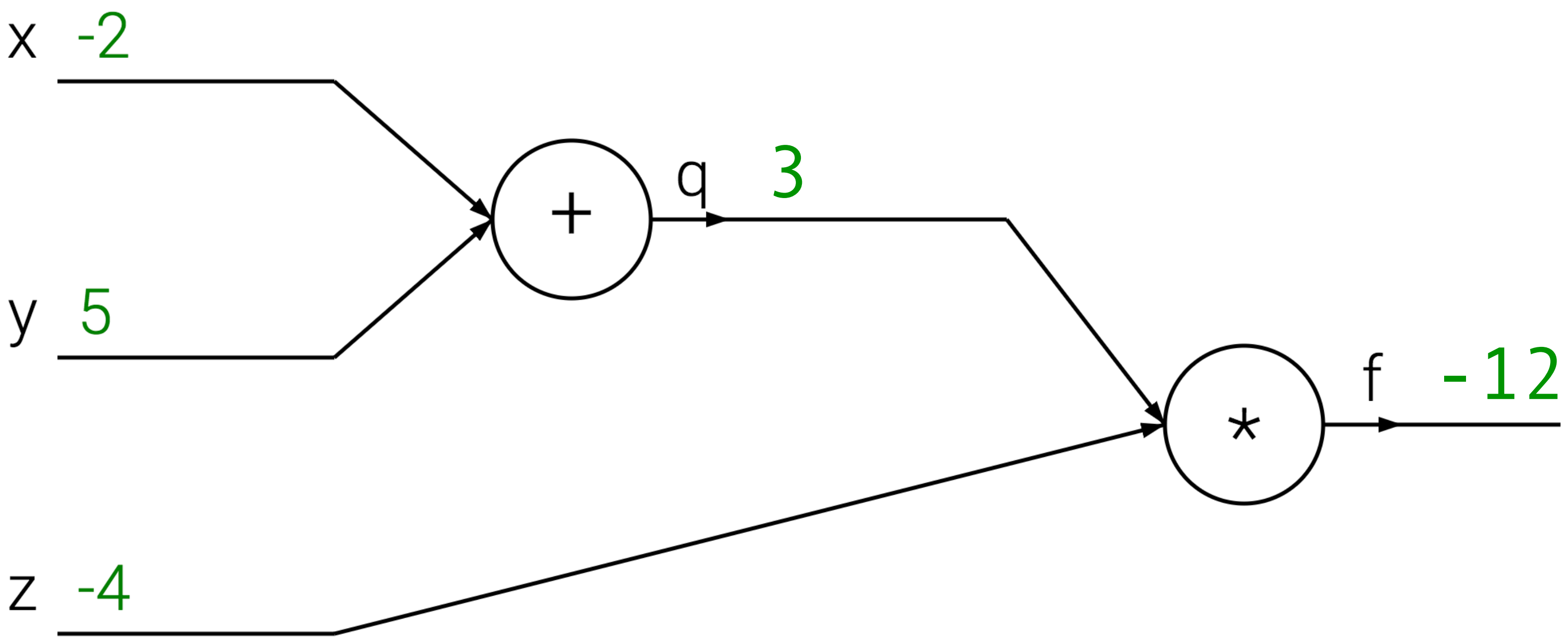
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

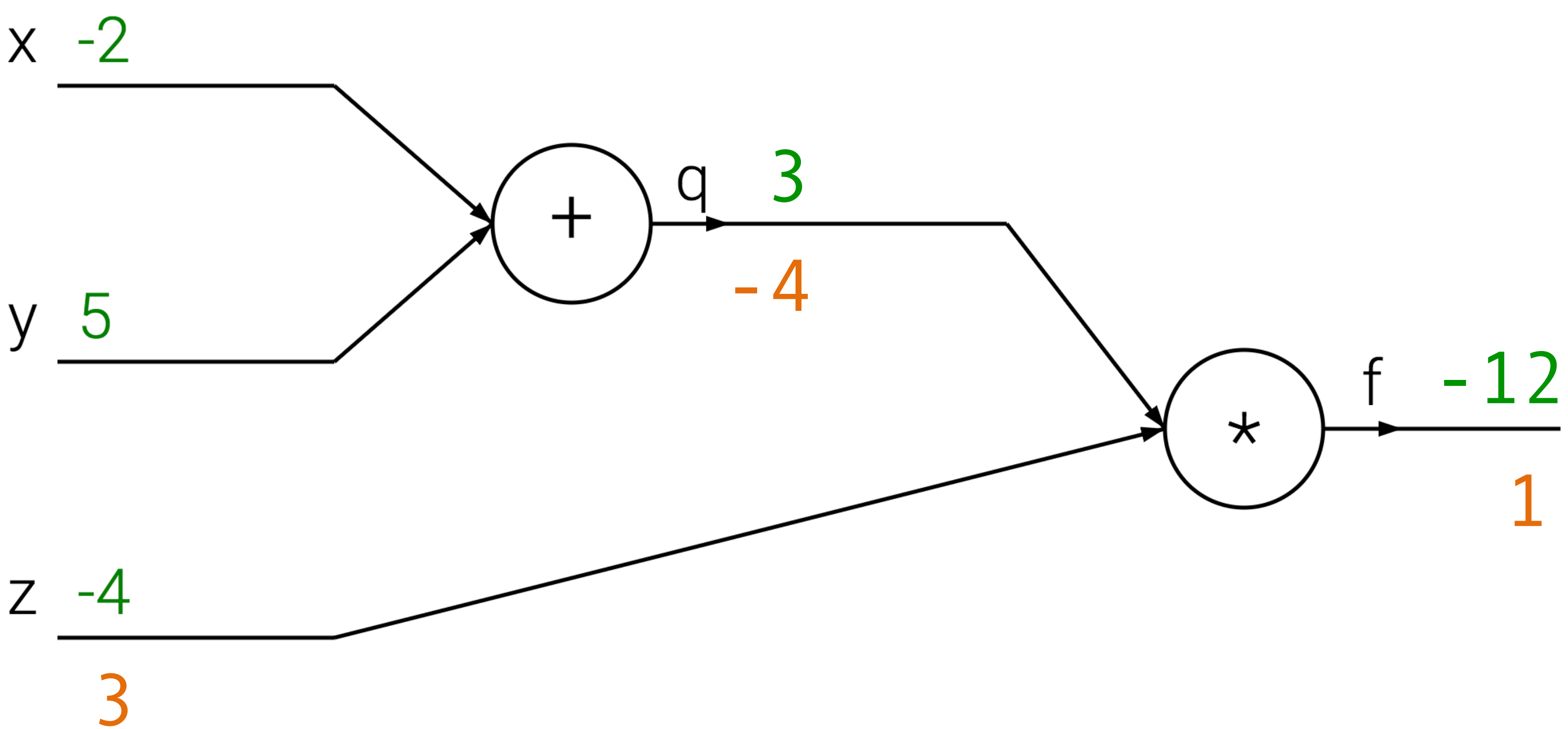
Forward pass: compute values



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

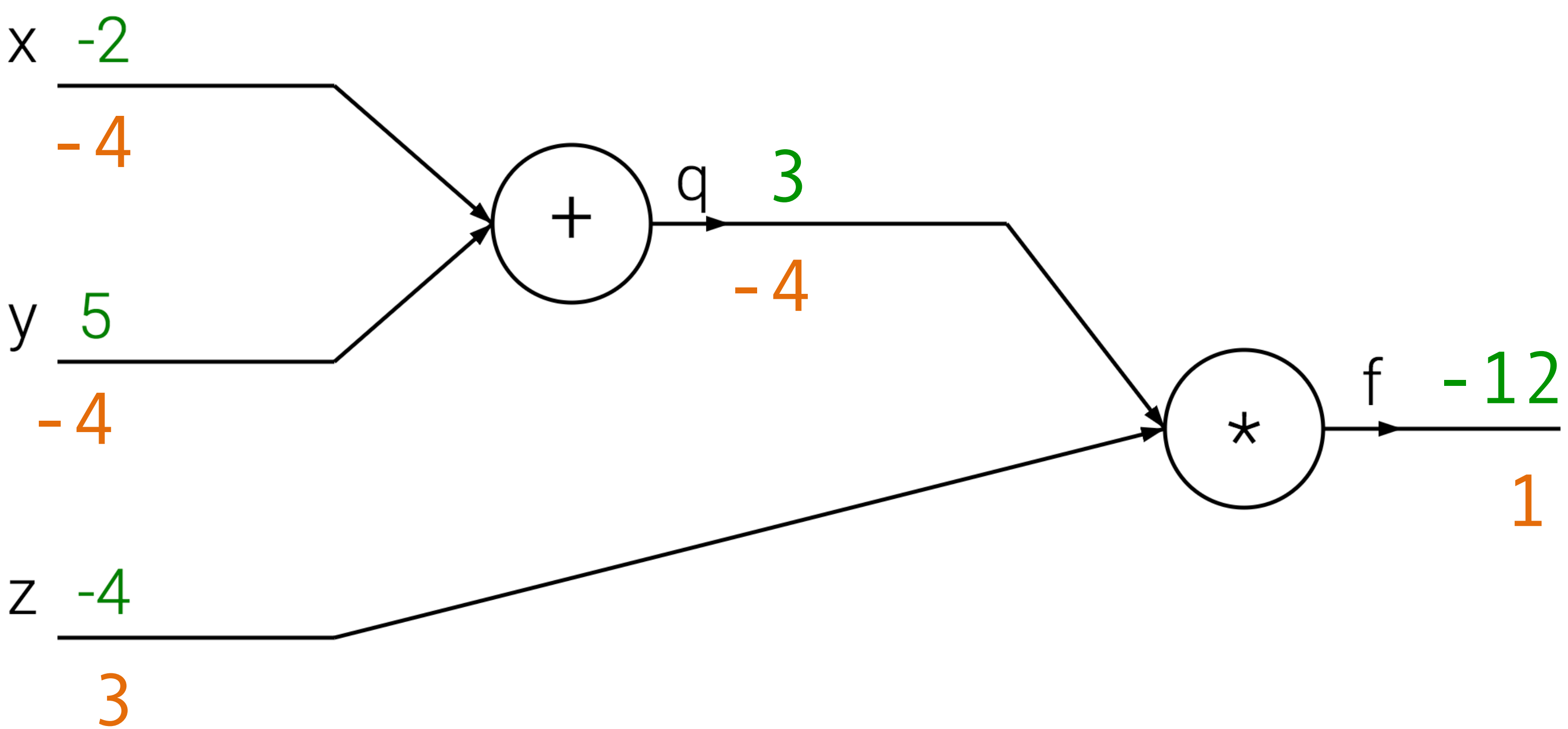
Backward pass: compute local gradients



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

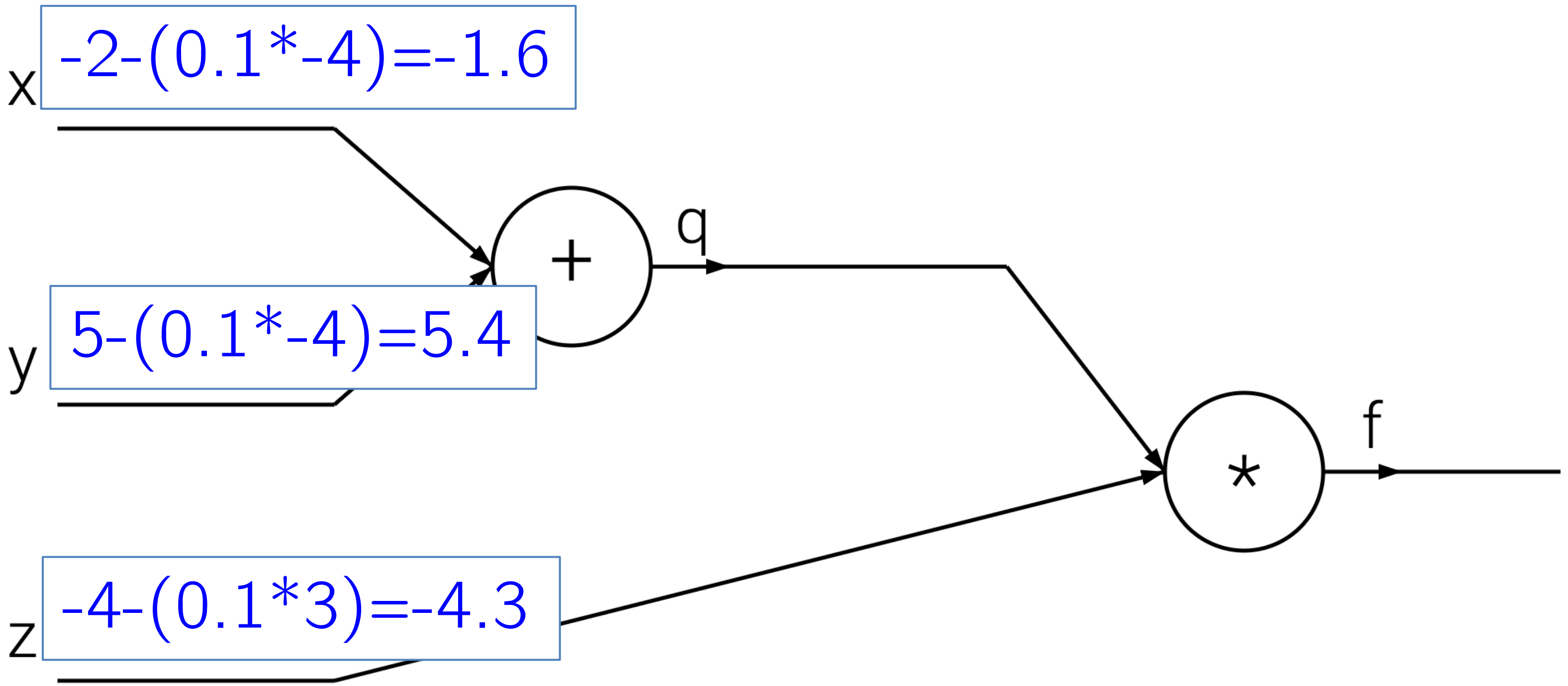
Backward pass: compute local gradients



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

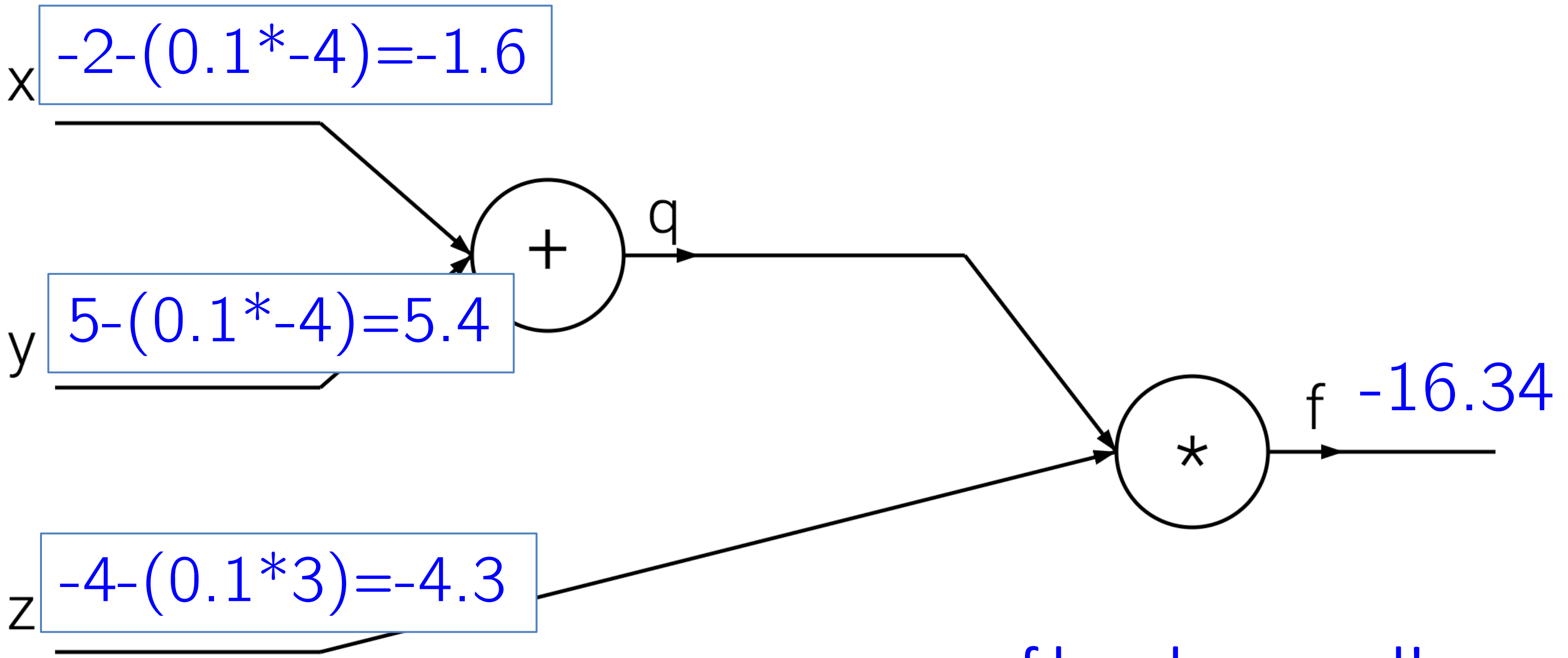
Now if we wanted to minimize f => opposite direction of gradient



Example from:
http://cs231n.github.io/optimization-2/

Backpropagation: Example

Now if we wanted to minimize f => opposite direction of gradient



f has decreased!

Example from:
http://cs231n.github.io/optimization-2/

ℓ = overall loss

$$X_{in} \rightarrow \boxed{f} \rightarrow Y_{out} = f(x_{in})$$

$$\underbrace{\frac{\partial f}{\partial x}\Big|_{x_{in}} \cdot \frac{\partial \ell}{\partial f}\Big|_{Y_{out}}}_{\text{chain rule}}$$

$$\underbrace{\frac{\partial f}{\partial x}}_{\text{function}} \qquad \underbrace{\underbrace{\frac{\partial \ell}{\partial f}}_{\text{function}}\Big|_{Y_{out}}}_{\text{value (Scalar)}}$$

$$3 \xrightarrow{} \boxed{x^2} \xrightarrow{} 9$$

f

$$-12 \qquad 2x \qquad -2$$

"evaluated at"

$$\frac{\partial f}{\partial x}\Big|_3 \cdot \frac{\partial \ell}{\partial f}\Big|_9$$

$$2(3) \cdot (-2) = -12$$

$x_1 = 5$

$\boxed{7} = 1\big|_5 \cdot 7$

f

$x_1 + x_2$

4

7

$x_2 = -1$

$\boxed{7} = 1\big|_{-1} \cdot 7$

$$\begin{bmatrix} \dfrac{\partial f}{\partial x_1} = 1 \\[2ex] \dfrac{\partial f}{\partial x_2} = 1 \end{bmatrix} = \text{gradient} = \nabla f$$

(n, p)

(n, p_1)

$\underbrace{p_1 = 1}_{\text{common}}$

(p, p_1)

(p_1, p_1)

p_1

$x_{in} = 2$

$\log(x)$

$\log(2)$

-4

$\boxed{-2}$

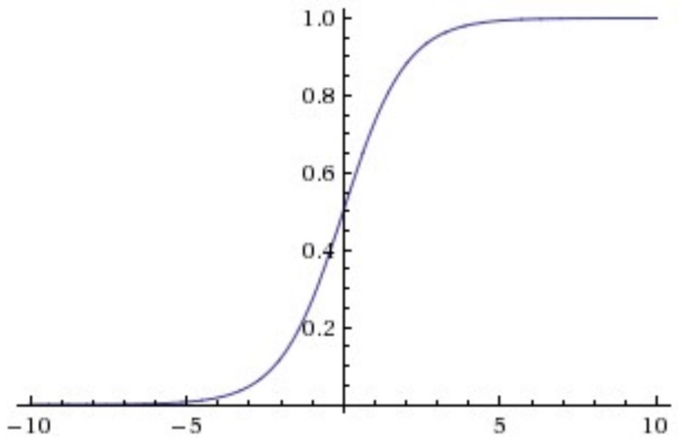
$\frac{1}{x}$

$$\frac{1}{x}\Big|_2 \cdot (-4) = -2$$

Option 1: sigmoid function

- Input: all real numbers, output: [0, 1]

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$



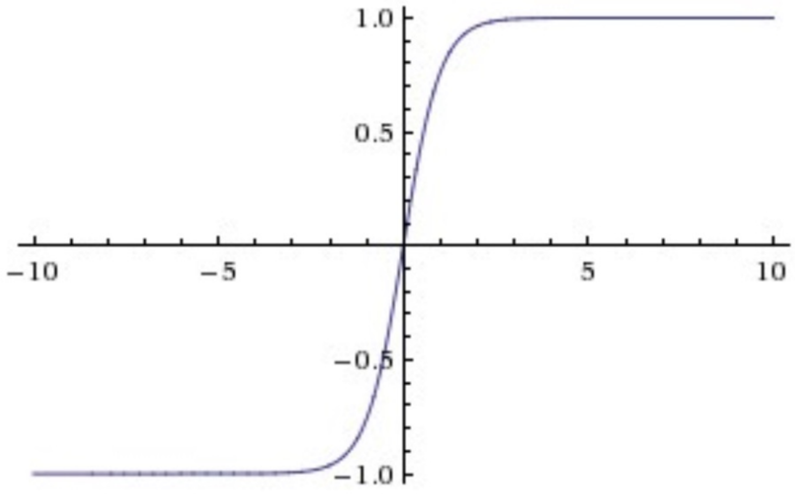
- Derivative is convenient

$$\boxed{\sigma'(x) = \sigma(x)(1-\sigma(x))}$$

Option 2: hyperbolic tangent

- Input: all real numbers, output: [-1, 1]

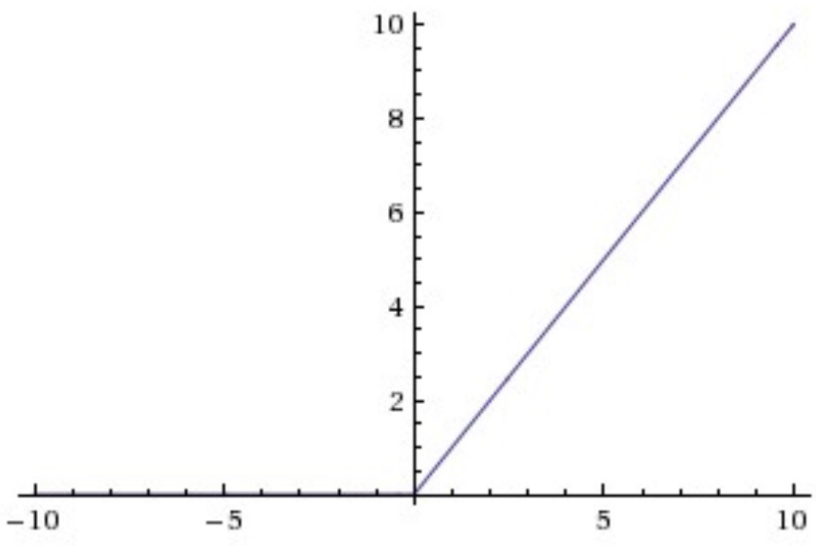
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Option 3: Rectified Linear Unit (ReLU)

- Return *x* if *x* is positive (i.e. threshold at 0)

$$\boxed{f(x) = \max(0, x)}$$



Pros and Cons of Activation Functions

1) Sigmoid

- (-) When input becomes very positive or very negative, gradient approaches 0 (saturates and stops gradient descent)
- (-) Not zero-centered, so gradient on weights can end up all positive or all negative (zig-zag in gradient descent)
- (+) Derivative is easy to compute given function value!

2) Tanh

- (-) Still has a tendency to prematurely kill the gradient
- (+) Zero-centered so we get a range of gradients
- (+) Rescaling of sigmoid function so derivative is also not too difficult

3) ReLU

- (+) Works well in practice (accelerates convergence)
- (+) Function value very easy to compute! (no exponentials)
- (-) Units can "die" (no signal) if input becomes too negative throughout gradient descent

More info:
http://cs231n.github.io/neural-networks-1/

Outline for April 4

- Finish Backpropagation
- Recurrent neural networks
- Attention mechanisms
- Applications
- Transformers

Recurrent Neural Networks

(RNNs)

arbitrary length

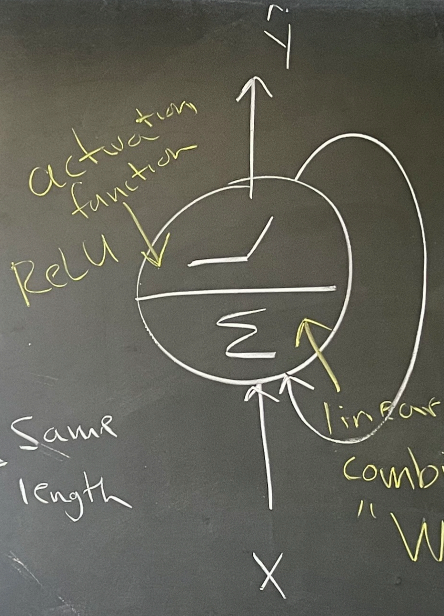
input $X = [X_{(0)}, X_{(1)} \dots X_{(t)} \dots]$

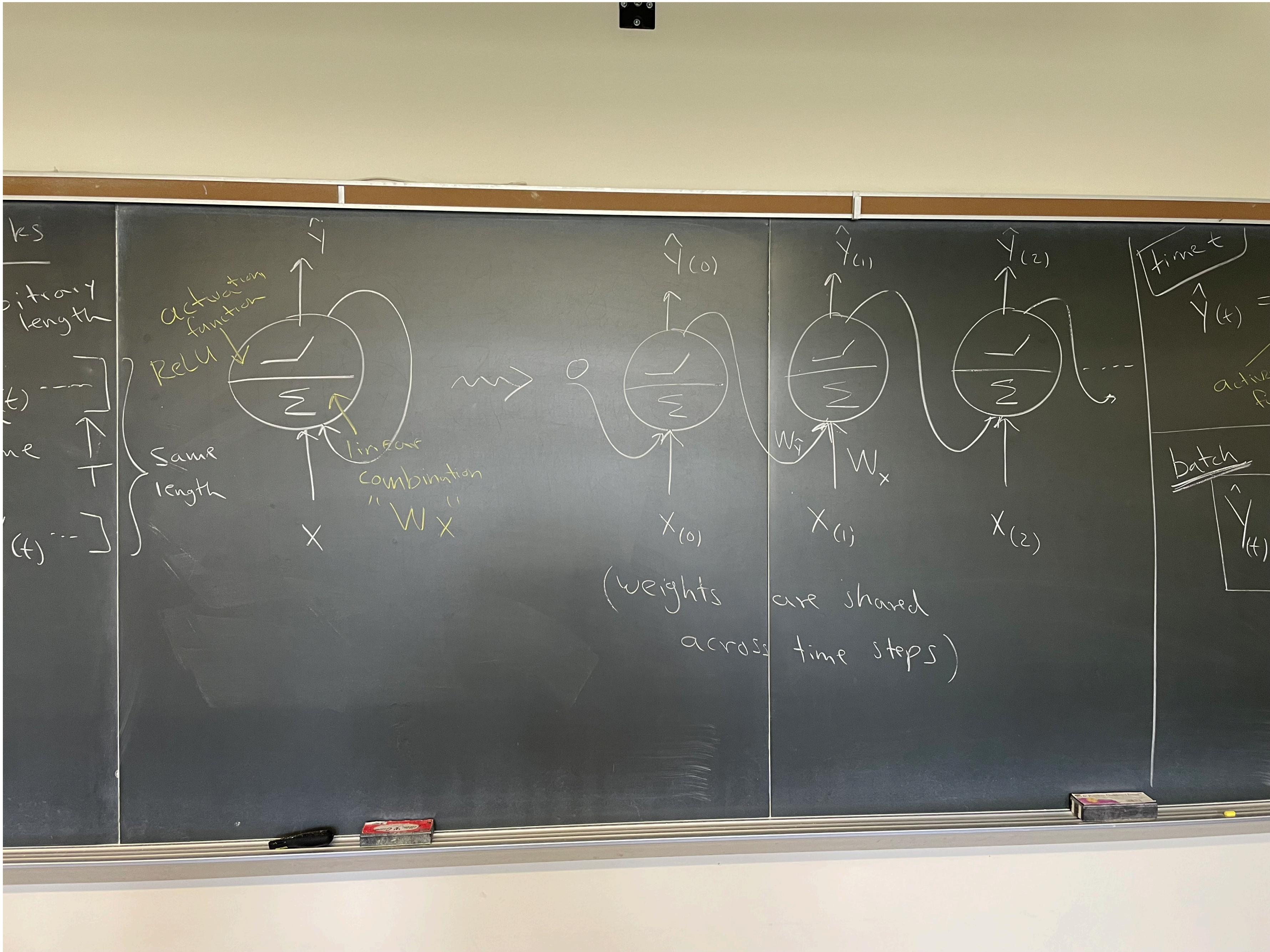
time

T

output $Y = [Y_{(0)}, Y_{(1)} \dots Y_{(t)} \dots]$

same length





ks
arbitrary length
same length
activation function
ReLU
linear combination
"Wx"
$\hat{Y}$
X
$\hat{Y}_{(0)}$
$\hat{Y}_{(1)}$
$\hat{Y}_{(2)}$
$W_{\hat{Y}}$
W_x
$X_{(0)}$
$X_{(1)}$
$X_{(2)}$
(weights are shared across time steps)
time t
$\hat{Y}_{(t)}$
batch

time t

$$\hat{y}_{(t)} = a\Big(W_x^T \underbrace{x_{(t)}}_{\text{input a time } t} + W_{\hat{y}}^T \underbrace{\hat{y}_{(t-1)}}_{\text{output from previous time}} + \underbrace{b}_{\text{bias}}\Big)$$

activation function (element wise)

batch

$$\boxed{\hat{Y}_{(t)} = a\left(X_{(t)} W_x + \hat{Y}_{(t-1)} W_{\hat{y}} + b\right)}$$

inputs to next time step

$X_{(t)} = (n, p)$

$\hat{Y}_{(t)} = (n, p_1)$

$W_x = (p, p_1)$

$W_{\hat{y}} = (p_1, p_1)$

$b = p_1$

$\underbrace{p_1 = 1}_{\text{common}}$

$x_1 = 5$

$\boxed{7} = 1/5 \cdot 7$

$x_2 = -1$

$\boxed{7} = 1/{-1} \cdot 7$

Recurrent neural networks

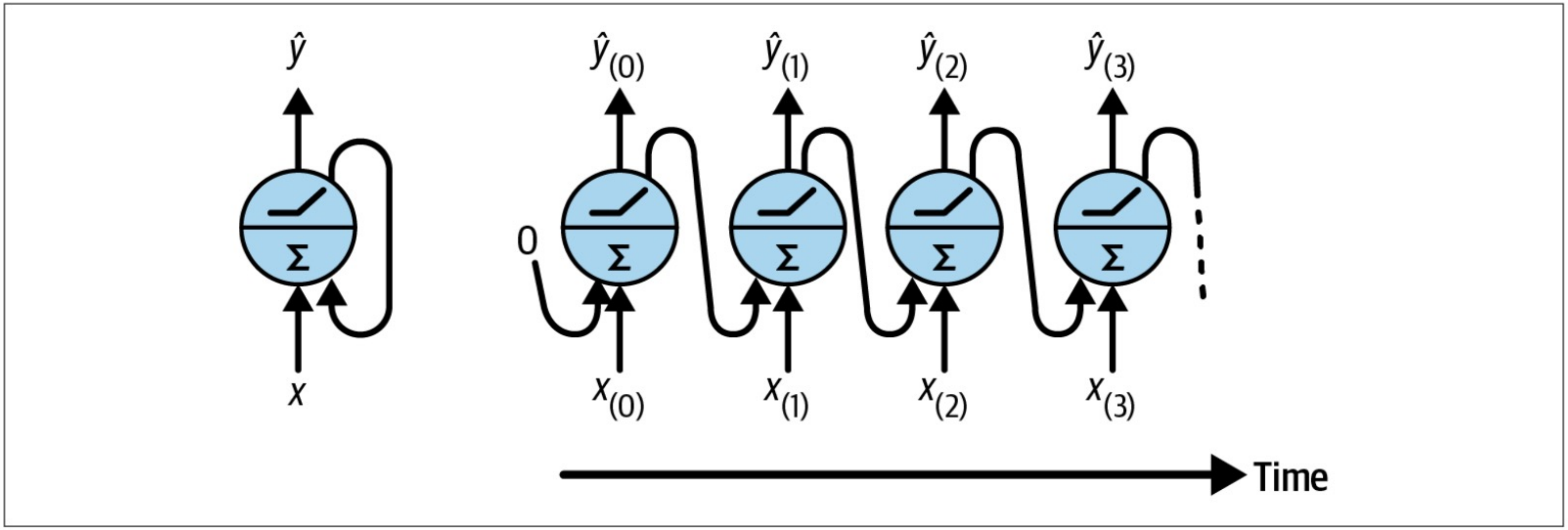


Figure 15-1. A recurrent neuron (left) unrolled through time (right)

Geron, Chap 15

Recurrent neural networks

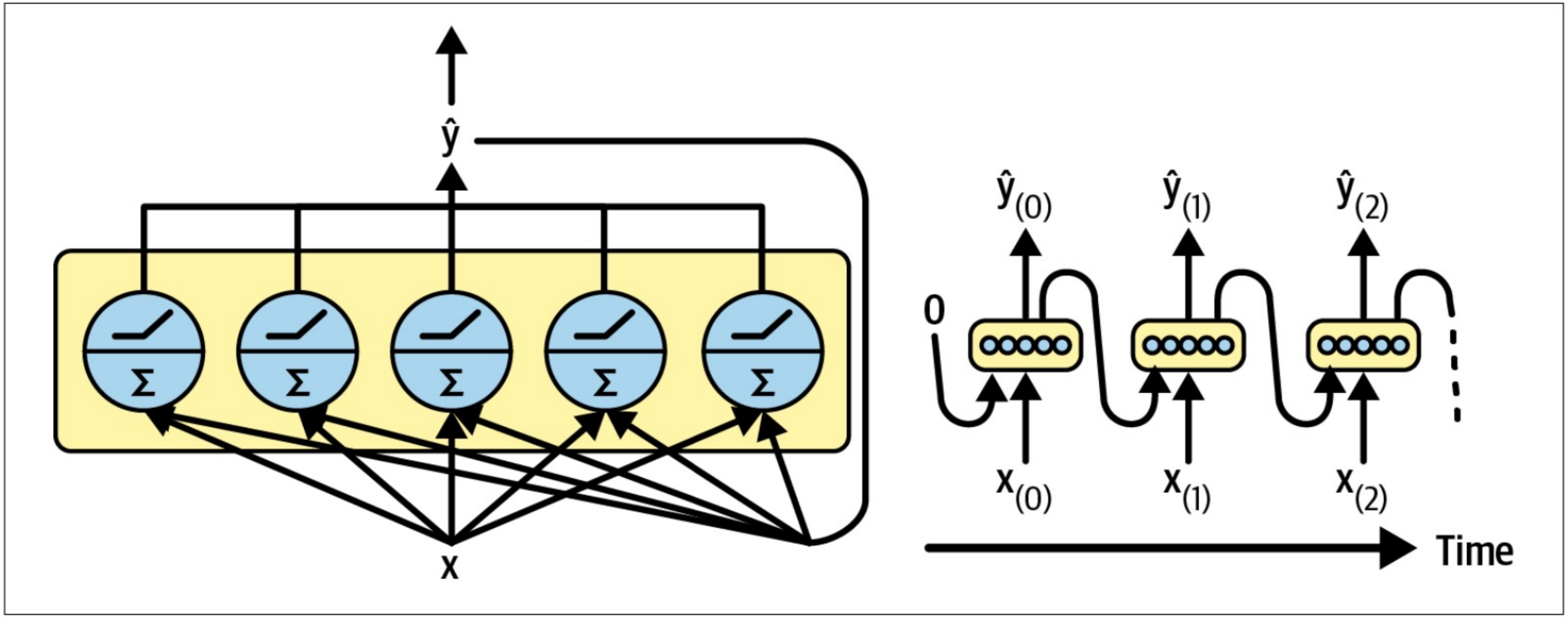
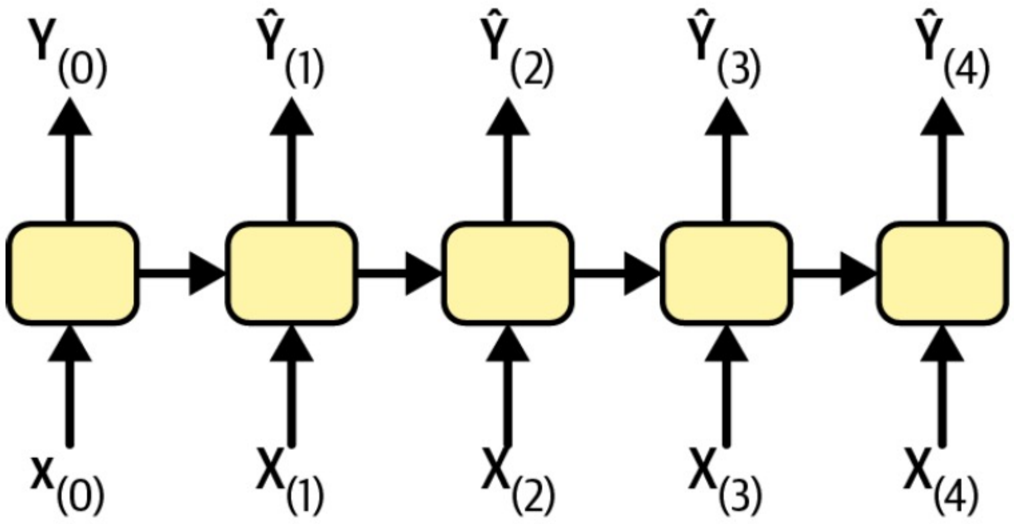


Figure 15-2. A layer of recurrent neurons (left) unrolled through time (right)

Geron, Chap 15

RNNs are flexible

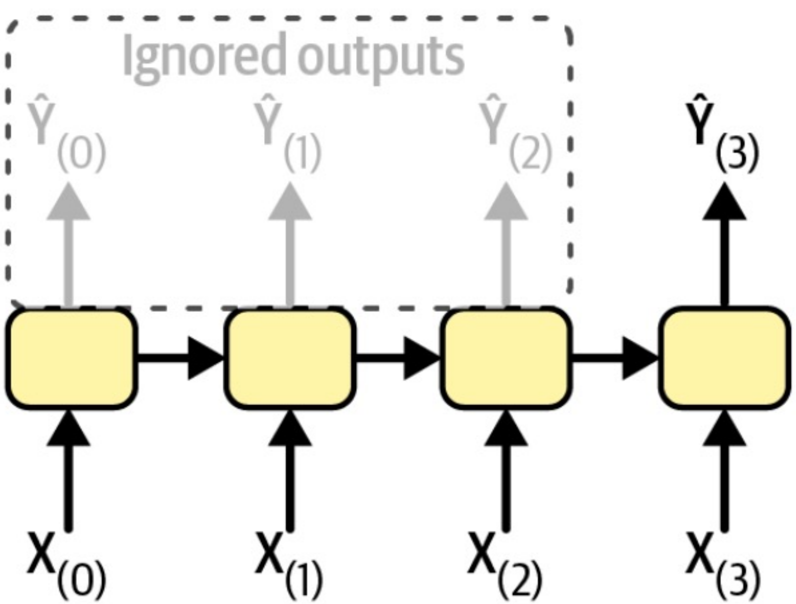
- **Sequence-to-sequence**
  - Example: predict power consumption
    - <u>Input</u>: power for N days
    - <u>Output</u>: power for N days shifted one day into the future



Geron, Chap 15

RNNs are flexible

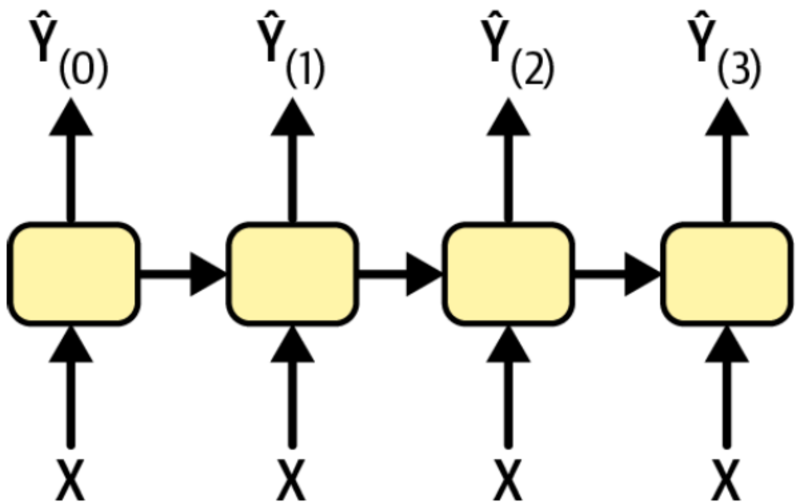
- **Sequence-to-vector**
  - Example: sentiment analysis
    - Input: text of review/tweet/post etc
    - Output: ignore all outputs but the last one and convert to 0 (negative) or 1 (positive), i.e. binary classification



Geron, Chap 15

RNNs are flexible

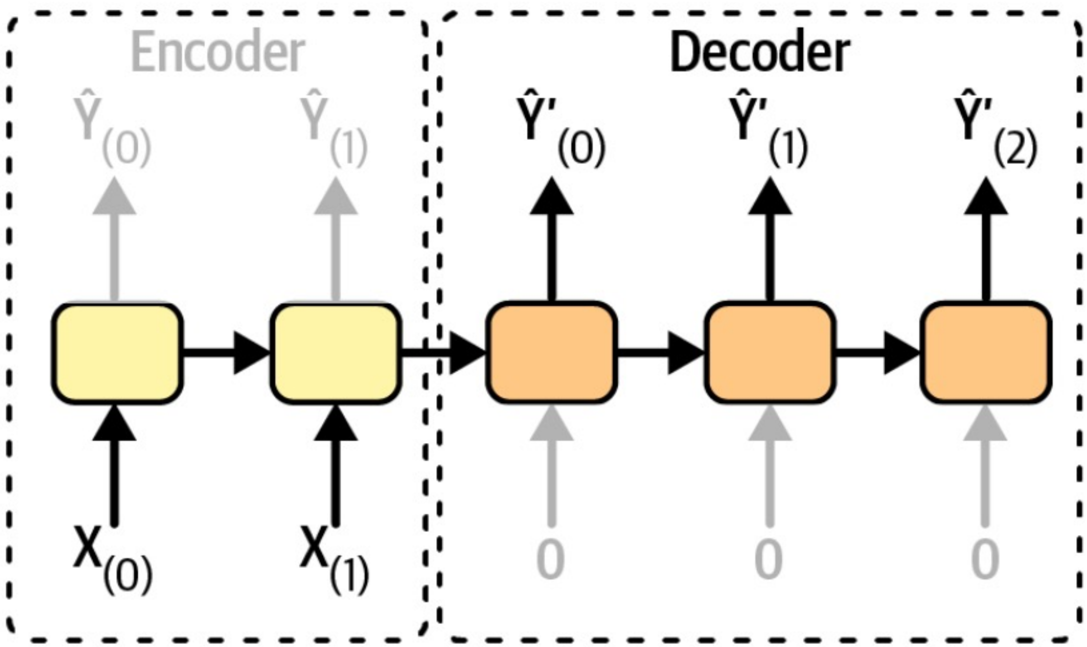
- **Vector-to-sequence**
  - Example: caption generation
    - Input: output from an image CNN (same at each "time")
    - Output: text caption for the image



Geron, Chap 15

RNNs are flexible

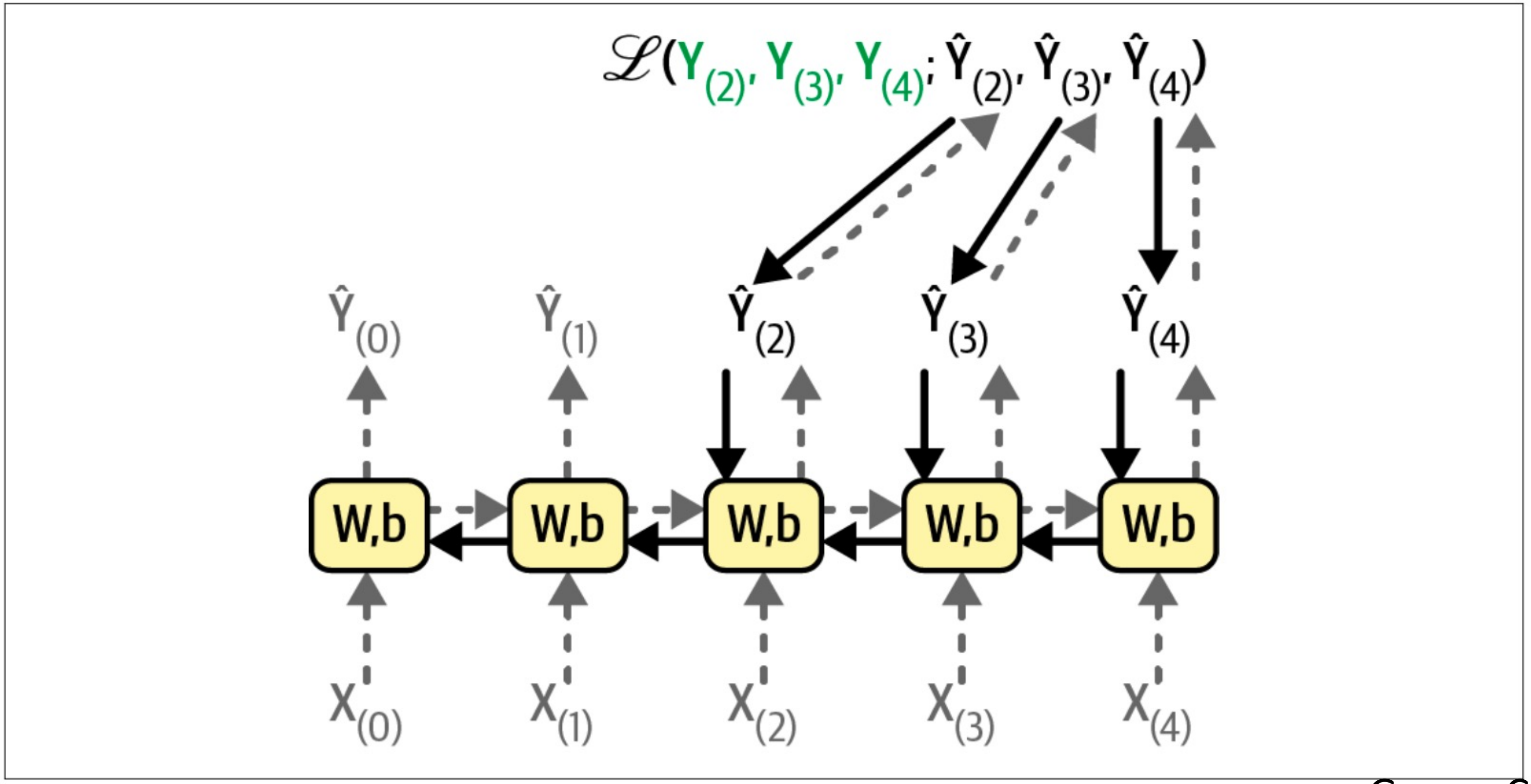
- **Encoder-decoder**
  - Example: machine translation (two sequence-to-vector networks)
    - Input: sentences in one language
    - Output: sentences in another language



Geron, Chap 15

Training RNNs

- Still backpropagation!
- Dashed lines: forward pass to compute outputs
- Solid lines: backward pass to compute gradients
- Note: loss function does not need to depend on all outputs



Geron, Chap 15

Deep RNNs

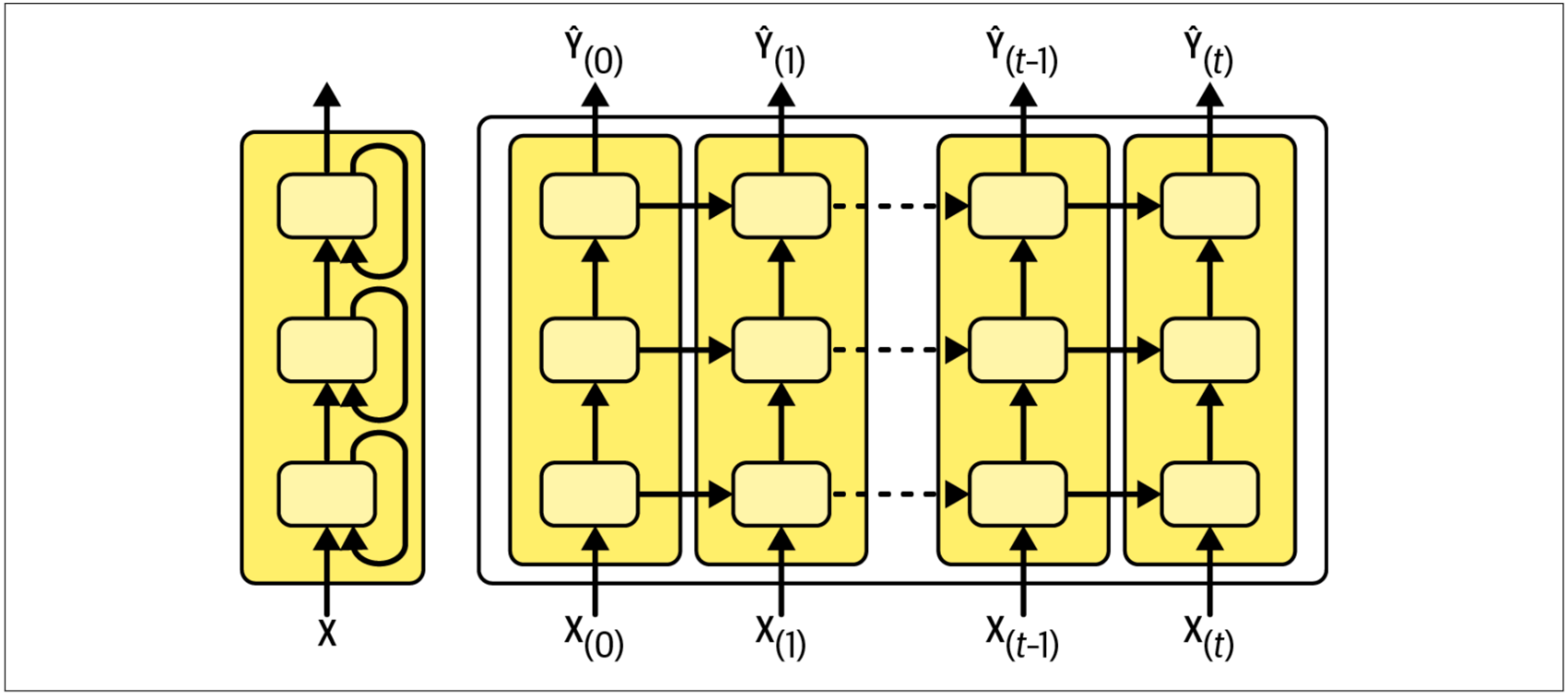


Figure 15-10. A deep RNN (left) unrolled through time (right)

Geron, Chap 15

RNN training problems

- Weights are shared across time steps
- In backpropagation, weights changes might accumulate across the entire sequence!
- ReLU can make this worse (doesn't saturate)

- Additionally, RNNs can lose long-term memory over time
- Set up to focus more on short-term memory

LSTM: hold on to long-term memory

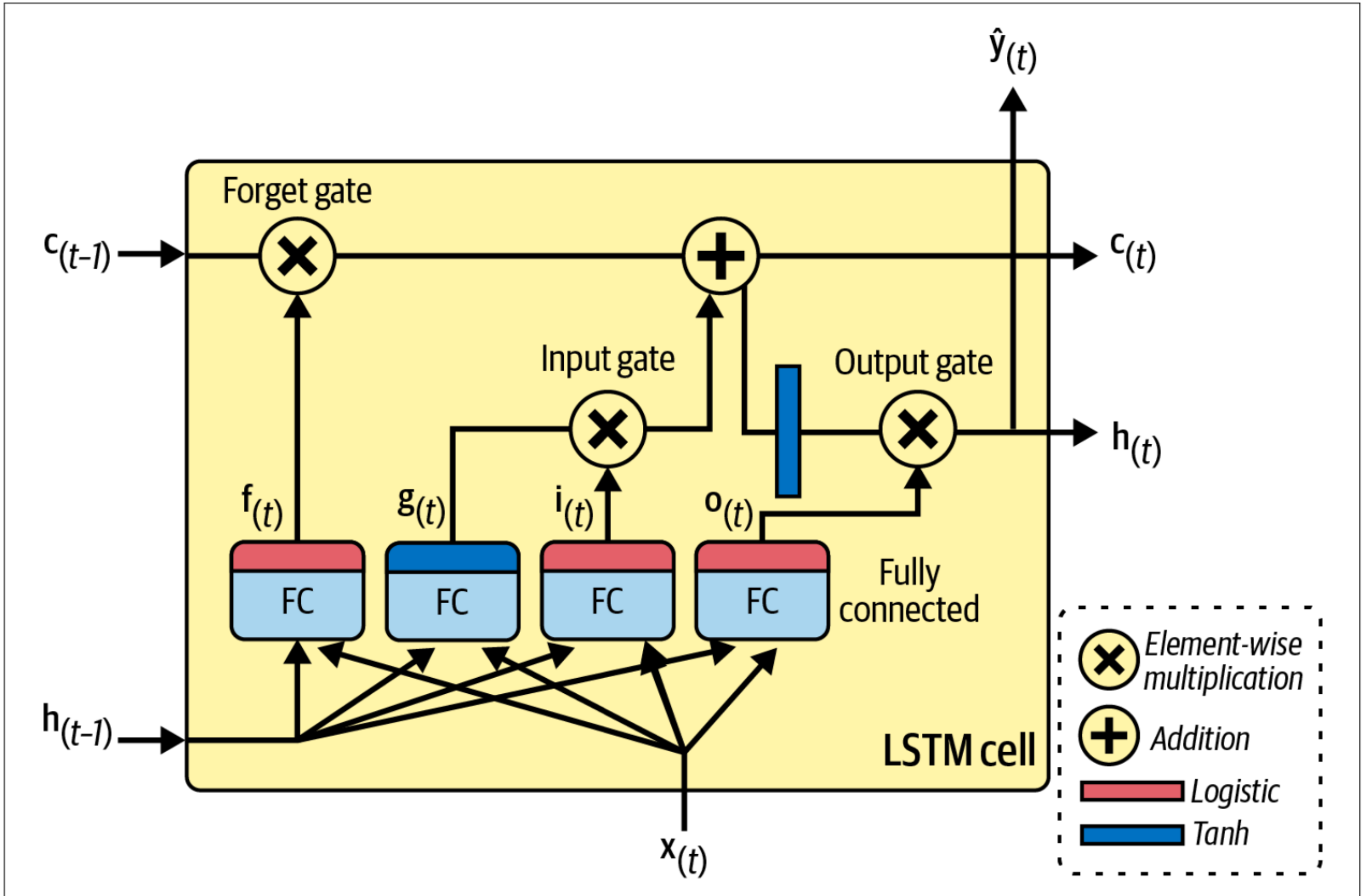
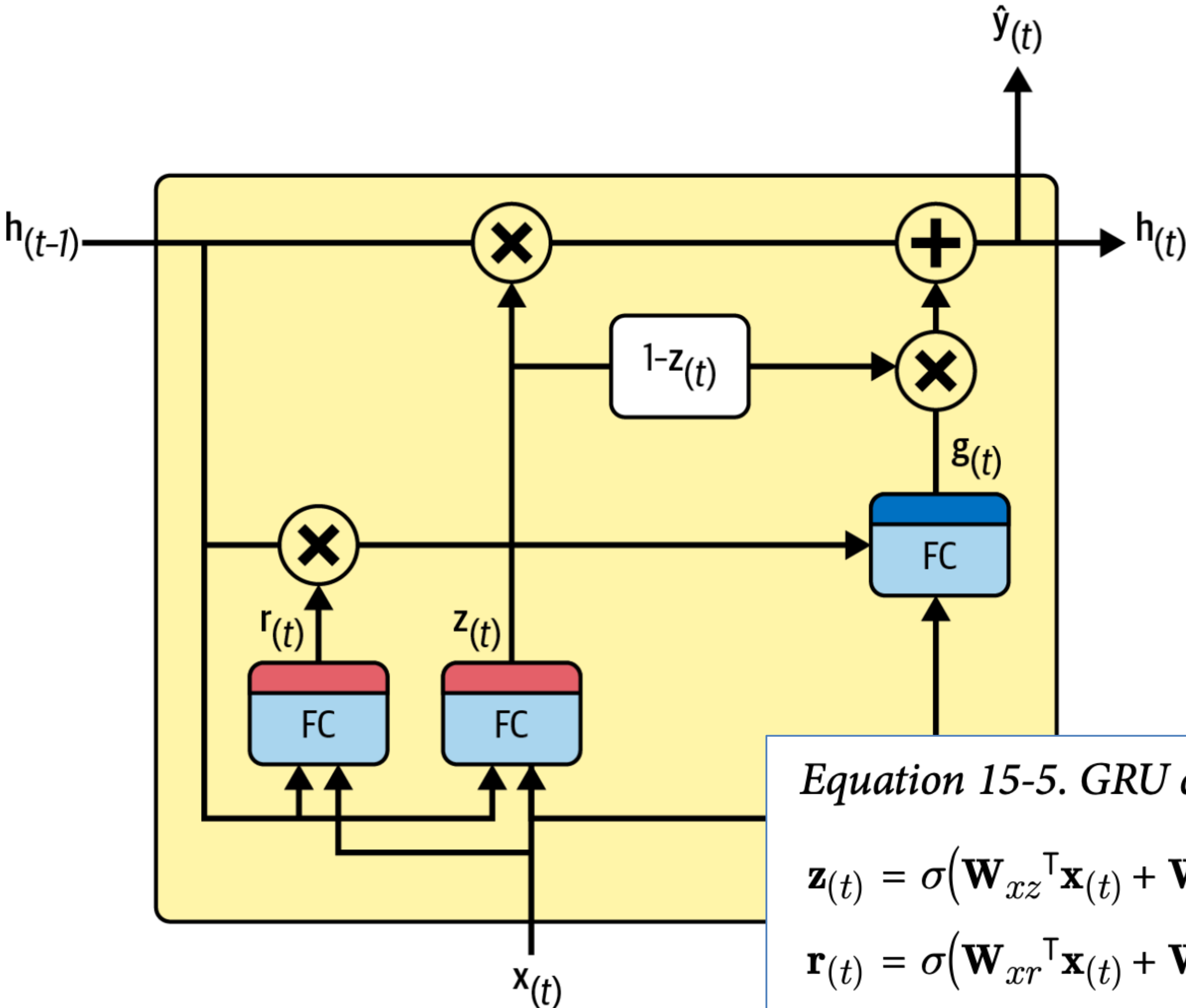


Figure 15-12. An LSTM cell

Geron, Chap 15

Gated Recurrent Unit (GRU)



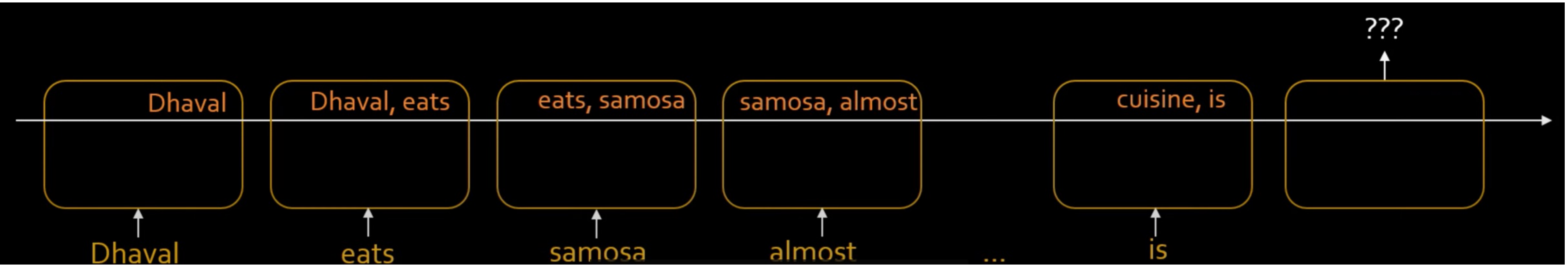
Equation 15-5. GRU computations

$$
\begin{aligned}
\mathbf{z}_{(t)} &= \sigma\left(\mathbf{W}_{xz}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_z\right) \\
\mathbf{r}_{(t)} &= \sigma\left(\mathbf{W}_{xr}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_r\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{\top}\left(\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}\right) + \mathbf{b}_g\right) \\
\mathbf{h}_{(t)} &= \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + \left(1 - \mathbf{z}_{(t)}\right) \otimes \mathbf{g}_{(t)}
\end{aligned}
$$

Geron, Chap 15

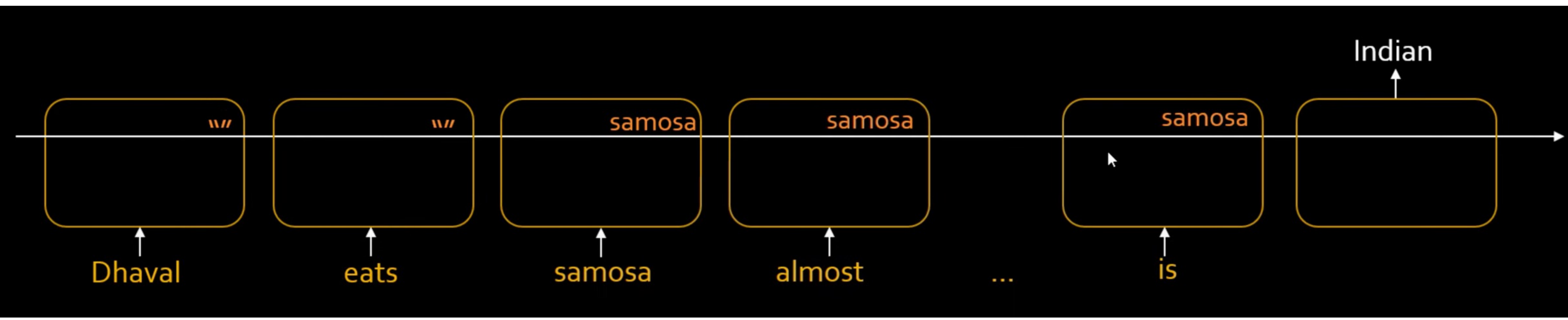
Traditional RNN example

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian



GRU example

Dhaval eats samosa almost everyday, it shouldn’t be hard to guess that his favorite cuisine is Indian



RNN-style machine translation

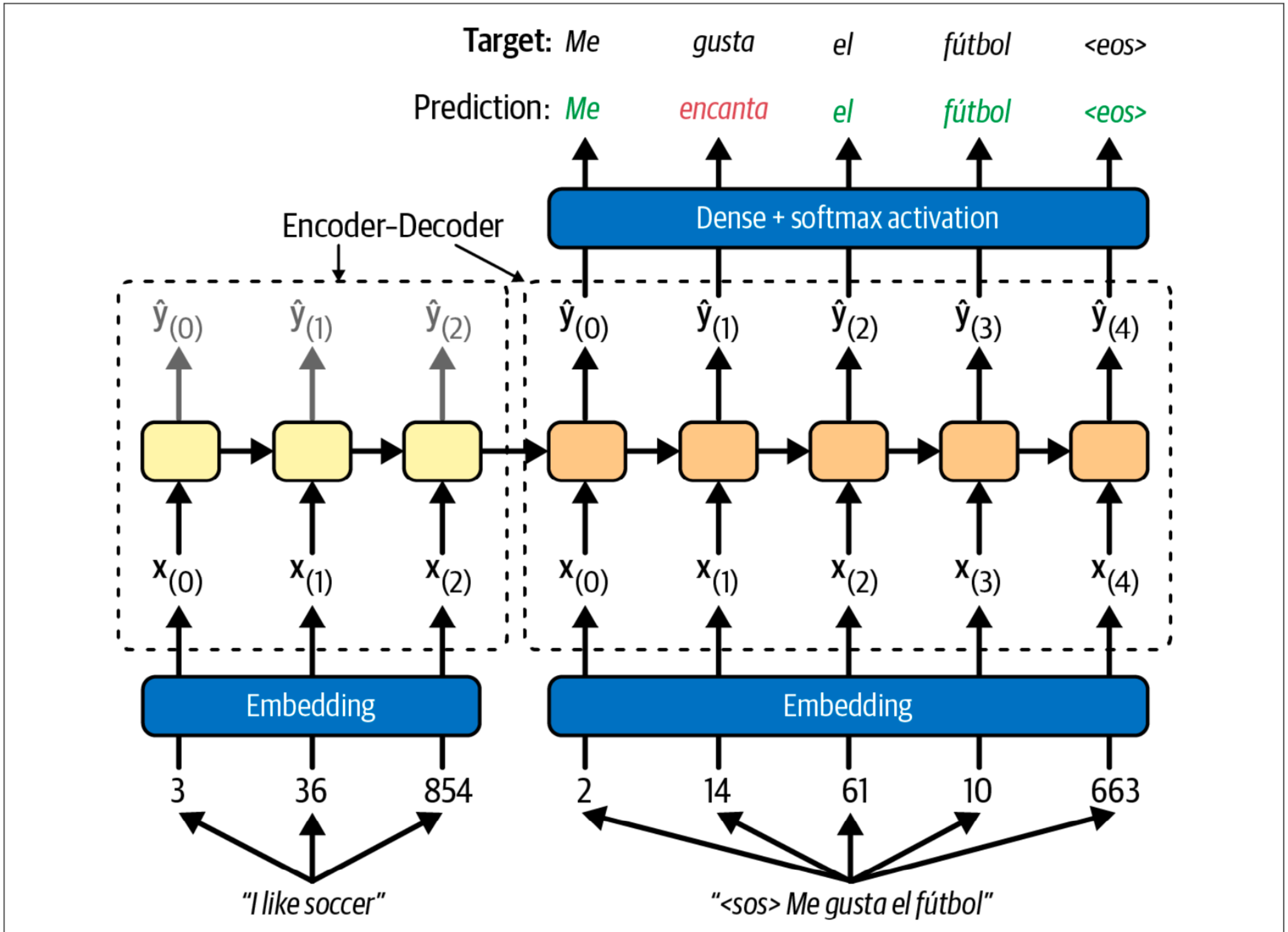


Figure 16-3. A simple machine translation model

Geron, Chap 16

Outline for April 4

- Finish Backpropagation
- Recurrent neural networks
- Attention mechanisms
- Applications
- Transformers

GitHub Copilot examples

```python
def compute_pi(matrix):
    from random import random

    inside = 0
    total = 0
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            if (i - 0.5) ** 2 + (j - 0.5) ** 2 <= 0.25:
                inside += 1
            total += 1
    return 4 * inside / total
```

prompt

Computing pairwise heterozygosity (biology summary statistic)

```python
# write a function to compute the average number of differences
# between all pairs of rows in a 2D matrix
def average_differences(matrix):
    total = 0
    count = 0
    for i in range(len(matrix)):
        for j in range(i + 1, len(matrix)):
            for k in range(len(matrix[i])):
                if matrix[i][k] != matrix[j][k]:
                    total += 1
            count += 1
    return total / count
```

prompt

Risks of language models

- Reliability
- Social bias
- Toxicity
- Disinformation
- Security
- Legal considerations
- Cost and environmental impact
- Access

Discussion question: given these risks, should language models remain public?

Percy Liang

Preprocessing for a language model

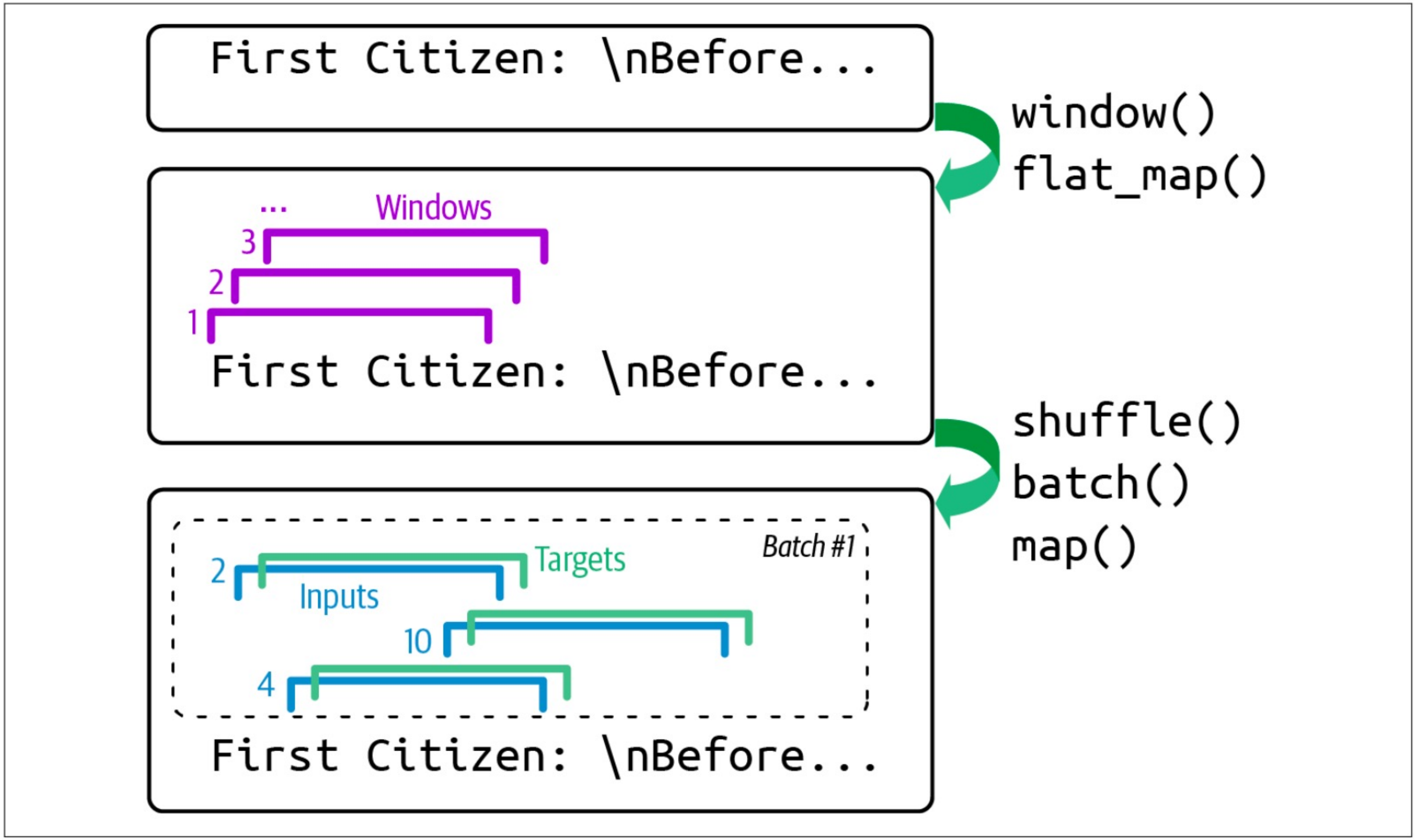


Figure 16-1. Preparing a dataset of shuffled windows

Geron, Chap 16

embedding

a	b	c	......	z	" "
↓	↓	↓		↓	↓
1	2	3		26	27

"letter by letter"

$\boxed{l = 5}$

block size
window size

input ⇒ "to be␣"

embedding ⇒ $[20, 15, 27, 2, 5] = \vec{x}$

$[15, 27, 2, 5, 27] = \vec{y}$

the␣the␣

Word Embeddings

- If we have 50,000 words and one-hot encoding, doesn't scale! (Very sparse matrix)
- Instead: embed in a lower dimension space

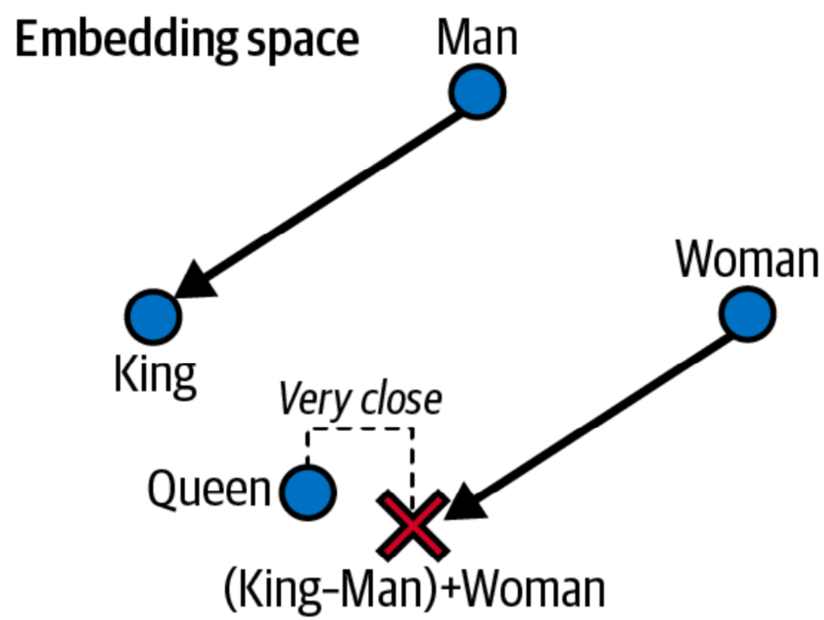


Figure 13-7. Word embeddings of similar words tend to be close, and some axes seem to encode meaningful concepts

Temperature: don’t always pick the letter with maximum probability

```
>>> print(extend_text("To be or not to be", temperature=0.01))
To be or not to be the duke
as it is a proper strange death,
and the
```

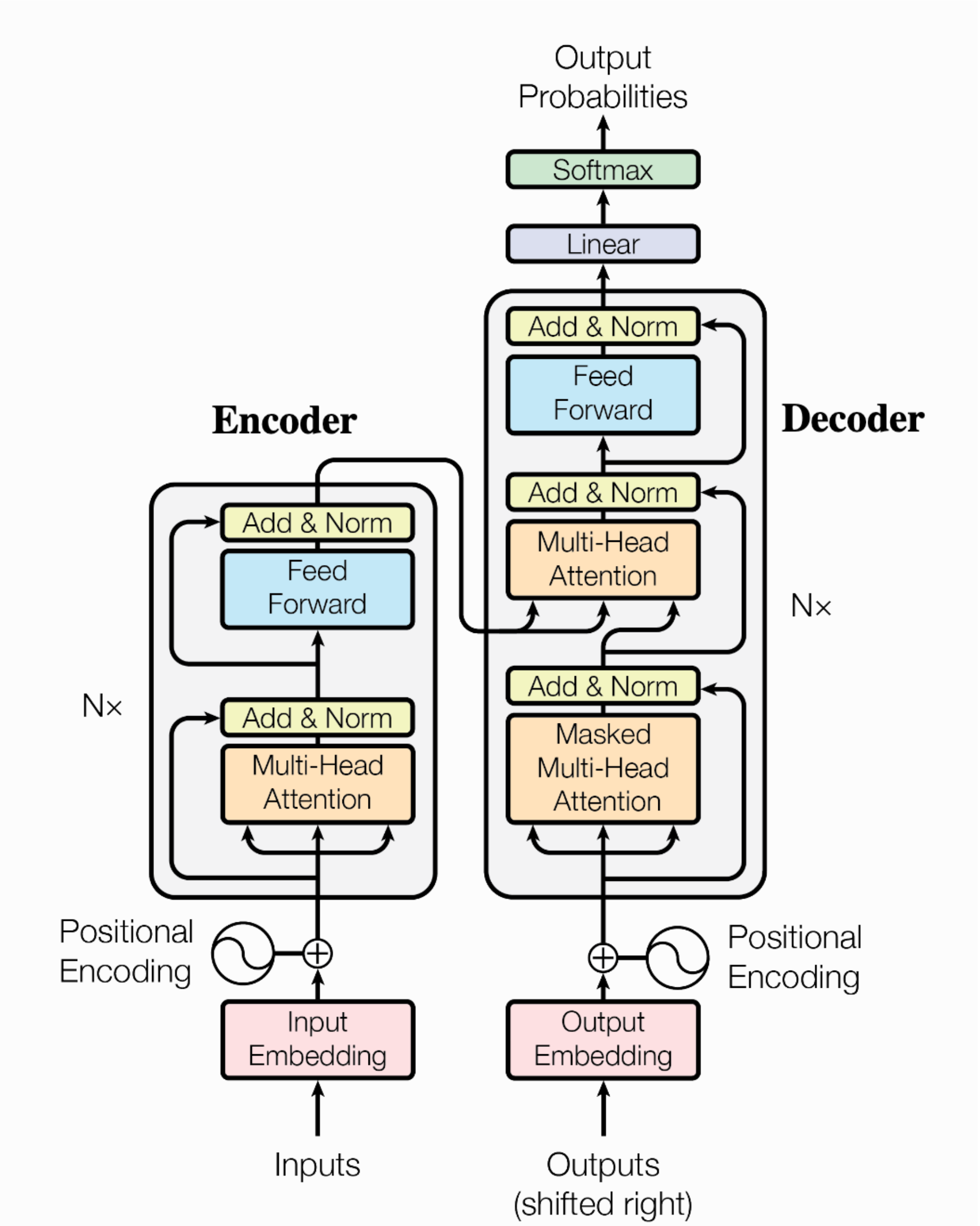
```
>>> print(extend_text("To be or not to be", temperature=1))
To be or not to behold?

second push:
gremio, lord all, a sistermen,
```

```
>>> print(extend_text("To be or not to be", temperature=100))
To be or not to bef ,mt'&o3fpadm!$
wh!nse?bws3est--vgerdjw?c-y-ewznq
```

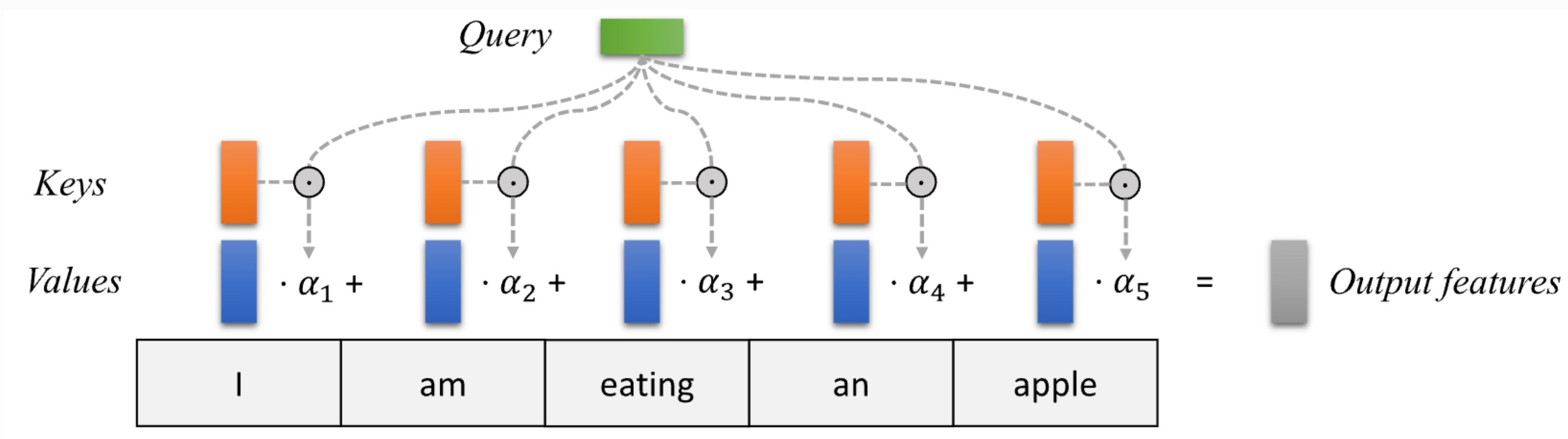
Geron, Chap 16

Transformer Architecture



“Attention is all you need”

Attention mechanisms



$$\alpha_i = \frac{\exp\left(f_{attn}\left(\text{key}_i, \text{query}\right)\right)}{\sum_j \exp\left(f_{attn}\left(\text{key}_j, \text{query}\right)\right)}, \quad \text{out} = \sum_i \alpha_i \cdot \text{value}_i$$

"Transformers and Multi-Head Attention" by Phillip Lippe