# CSCI 599: Deep Learning and its Applications

**Lecture 4**

Fall 2017

Joseph J. Lim

Lots of help from **Shao-Hua Sun**, **Youngwoon Lee**, and **Te-Lin Wu**

# Disclaimer

- This course is taught for the 1st time @ USC. This course is 599, and thus an **experimental** course.

- The syllabus, course policy, and grading details **may change** over the semester (**check website!**)

- If you prefer a well-structured course, this is **NOT** a course for you, and I encourage you to take the course next year. We really mean this.

- But, it will be **fun** and **challenging**!

# Welcome to CSCI 599!

- 175 students
  - PhD: 61
  - Master's: 102
  - Undergraduate: 12
- About 20 different majors
  - CSCI: ~110
  - EE: ~30
  - Others: ~30

# Communication

- Please use **Piazza** for any general communication including questions
  https://piazza.com/usc/fall2017/csci599

- Use e-mail ONLY when it is necessary. Seriously I don't know when…
  But, the staff e-mail address is: ~~deeplearning-staff-l@usc.edu~~

- Any non-necessary e-mail will be ignored.

# Communication

- Please do NOT
  - e-mail us individually (**we will not reply**)
  - come to our office without appointment

# Assignment 1

- Assignment 1 will be released next week (not this week)
- DUE October 18th, 2017 (week 8)

# Important Dates

- **Midterm: week 7**
- Assignment 1: **week 8** (changed from week 6)
- Assignment 2: week 11
- Project
  - **Team formation: week 4**
  - Project proposal: week 8
    - **Project meeting with TA #1: between week 4 - week 8**
    - Project meeting with Instructor #1: week 8 (M-W)
  - Project mid-report: week 12
    - Project meeting with TA #2: between week 8 - week 12
    - Project meeting with Instructor #2: week 11 (M-W)
  - Project report + Final presentation: week 15 (5-9:30pm) **4.5 hours**
    - Project meeting with TA #3: between week 12 - week 15

**Subject to change!**

# Project Evaluation

- Creativity and difficulty of the problem setup
- Novelty of the approach
- Thoroughness of the experiments
- Quality of student's presentation, report, and meetings with TA/instructor

**Extra credit** for creating your own project (OK to discuss and get help from TAs)

# Team Formation

- Submit your team information now
- It is OK to change the team until the proposal day.
- Link: https://goo.gl/forms/i7XnI8y08qDEoeE73

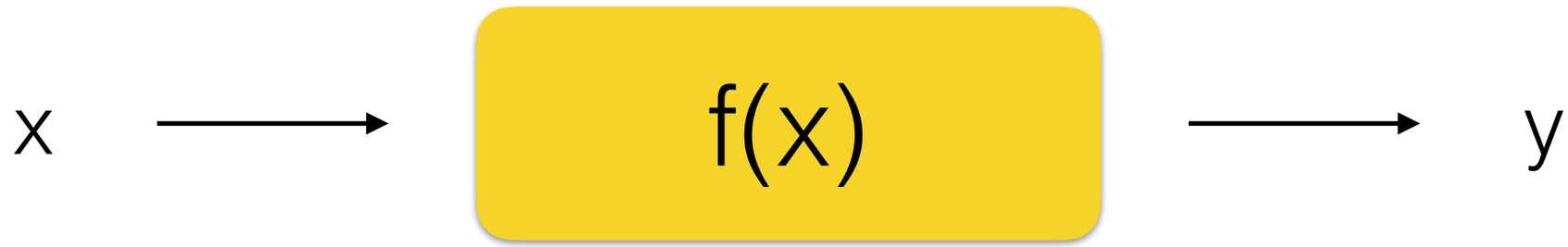
# Today's agenda

- Part 1
  - Loss function & Optimization
- Part 2
  - Neural Networks
- Part 3
  - Project Discussion

# Today's agenda

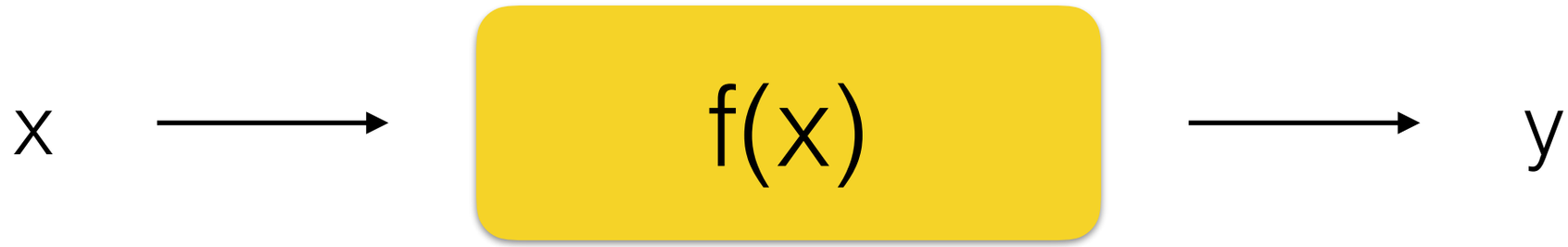
- Part 1
  - Loss function & Optimization
- Part 2
  - Neural Networks
- Part 3
  - Project Discussion

# Recap:Our course

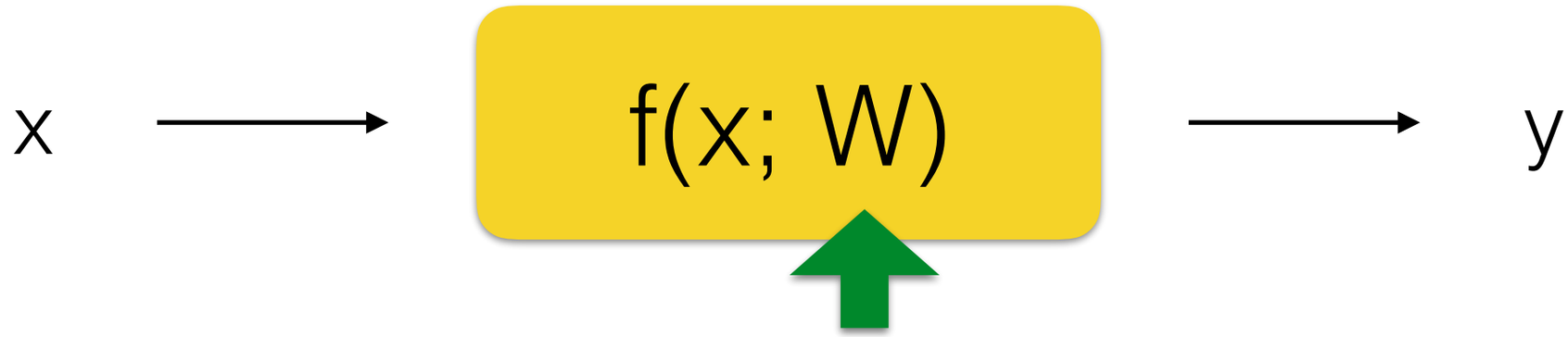



(1) How do we learn this function (using deep learning)?
(2) How to formulate a problem into this

# Loss function & Optimization

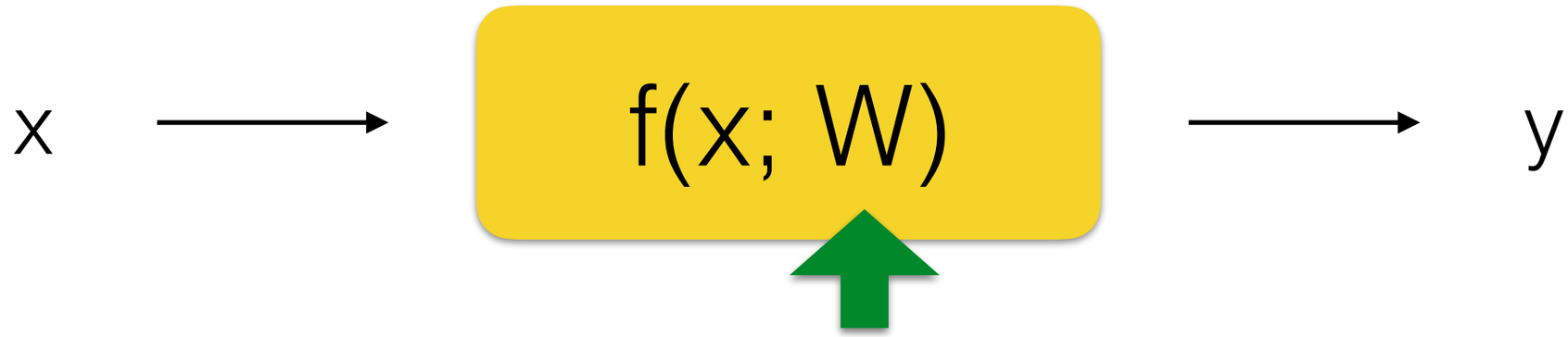


# Loss function & Optimization



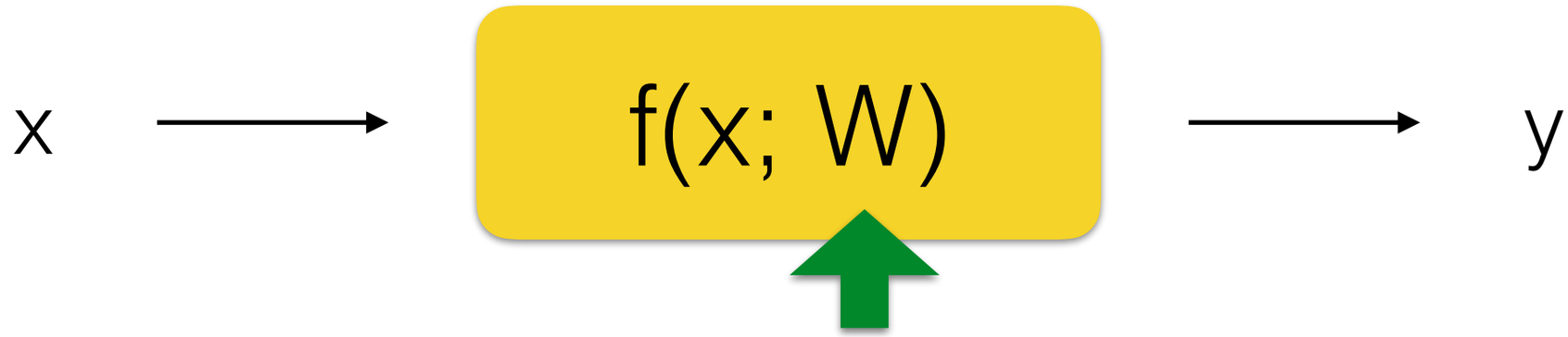

Showing W (to learn) explicitly
Parametric function

# Loss function & Optimization




- **Loss function (L)** measures how well learned W can map X to Y (compared to f*).

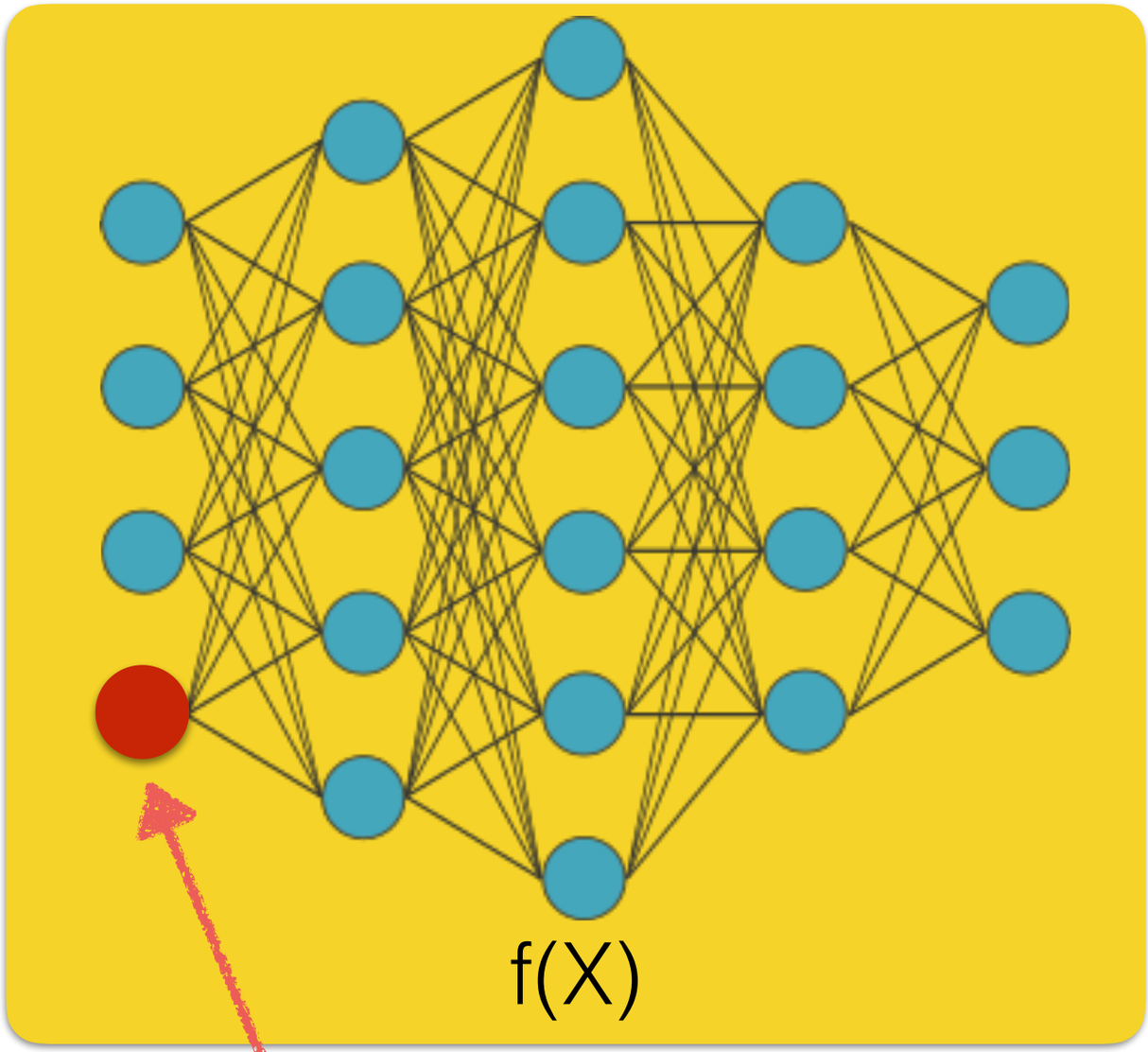
# Loss function & Optimization




- **Loss function (L)** measures how well learned W can map X to Y (compared to f*).

- **Optimization** finds the best W given a loss function L (i.e. finding W that minimizes L).

# Recap: Linear Classification

X → 

→ Y

Apparently, this is an unit used in deep neural networks too.

# Recap: Linear Classification

5x1 5x7500 7500x1

$$f(x; W) = Wx + b$$ (b: 5x1)

X → f(X; W) → Y

50x50x3 numbers
(7500 numbers)

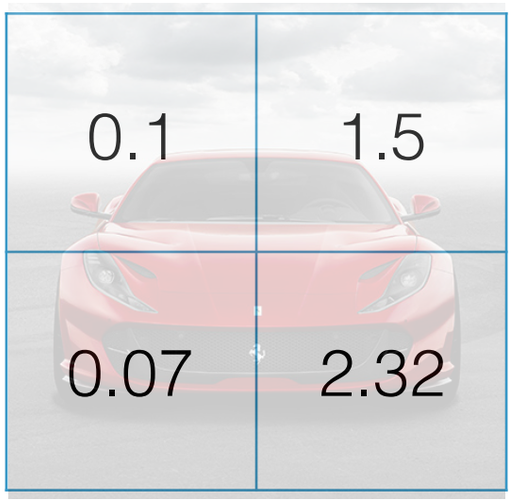
Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Y: output
X: input
W, b: **learned weight**

Modified from CS 231N @ Stanford

# Recap: Linear Classification




| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

| |
|---|
| 0.1 |
| 1.5 |
| 0.07 |
| 2.32 |

x

+

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

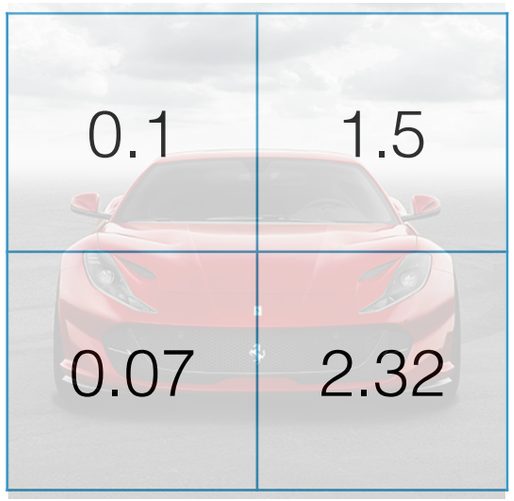
=

| | |
|---|---|
| 0.631 | Cat |
| 0.969 | Car |
| 0.71 | Airplane |

f(x; W,b)

Modified from CS 231N @ Stanford

# Recap: Linear Classification




| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

| |
|---|
| 0.1 |
| 1.5 |
| 0.07 |
| 2.32 |

x

+

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

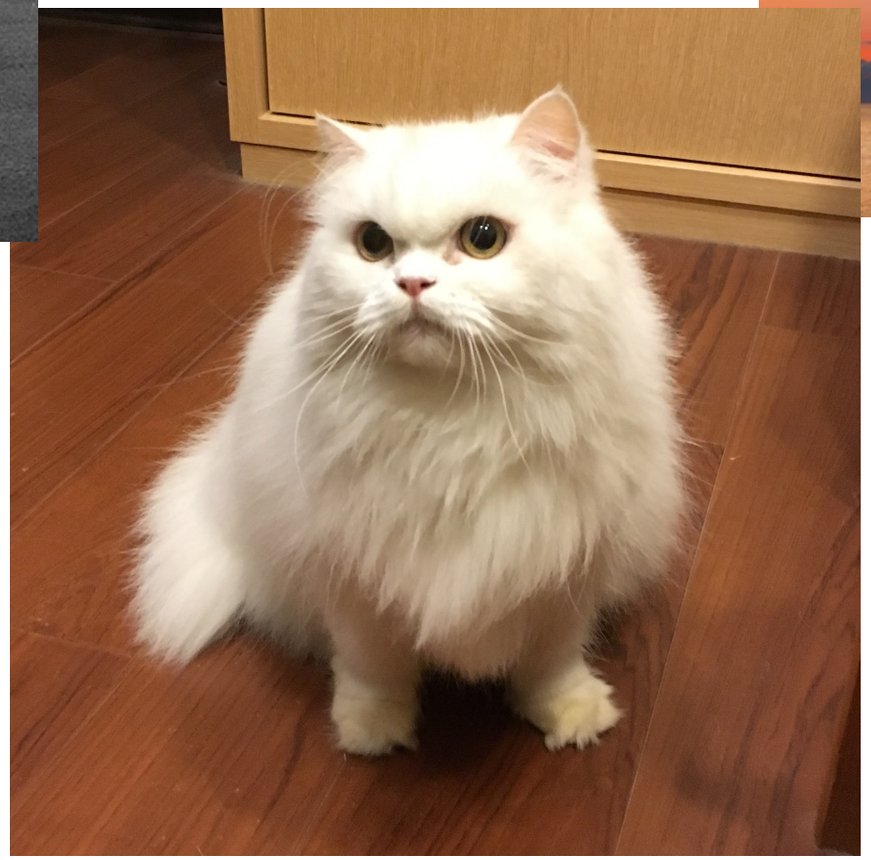
=

| | |
|---|---|
| 0.631 | Cat |
| 0.969 | Car |
| 0.71 | Airplane |

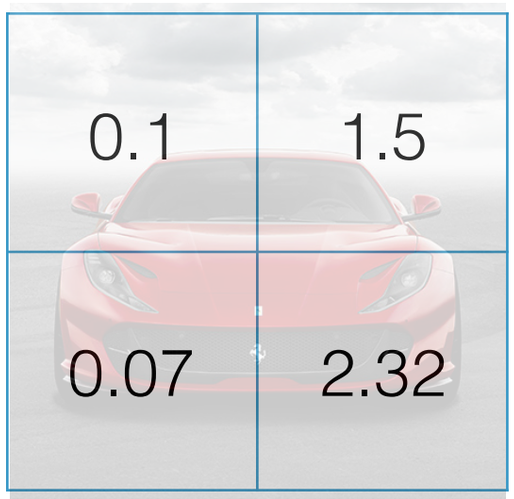
f(x; W,b)

Modified from CS 231N @ Stanford

# Linear Classification



# Linear Classification






| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

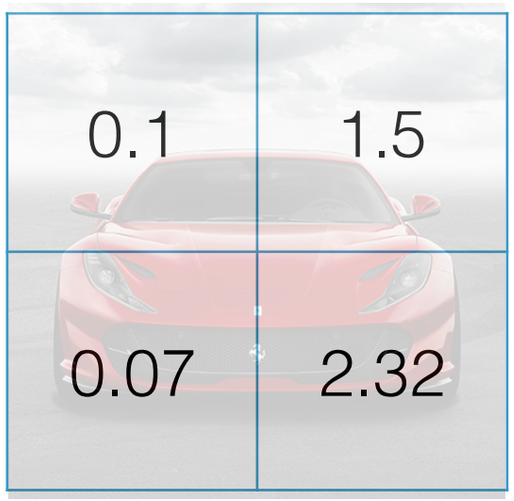
W

f(x; W,b)

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

# Linear Classification




| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

| |
|---|
| 0.1 |
| 1.5 |
| 0.07 |
| 2.32 |

x

+

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

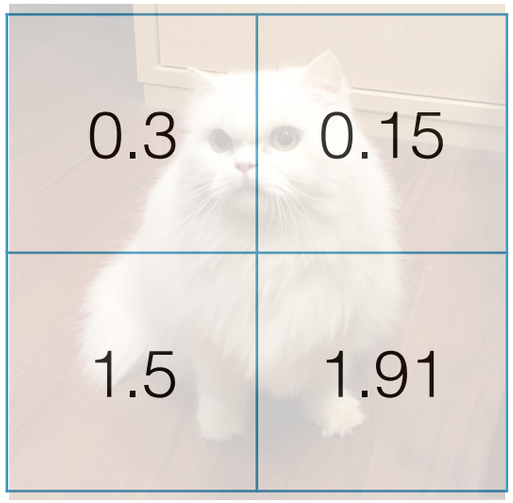
b

=

| | |
|---|---|
| 0.631 | Cat |
| 0.969 | Car |
| 0.71 | Airplane |

f(x; W,b)

# Linear Classification




| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

| |
|---|
| 0.3 |
| 0.15 |
| 1.5 |
| 1.91 |

x

+

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

=

| | |
|---|---|
| 0.73 | Cat |
| 0.598 | Car |
| 0.161 | Airplane |

f(x; W,b)

# Linear Classification


| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

| |
|---|
| 1 |
| 3 |
| 0.37 |
| 0.8 |

x

+

| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

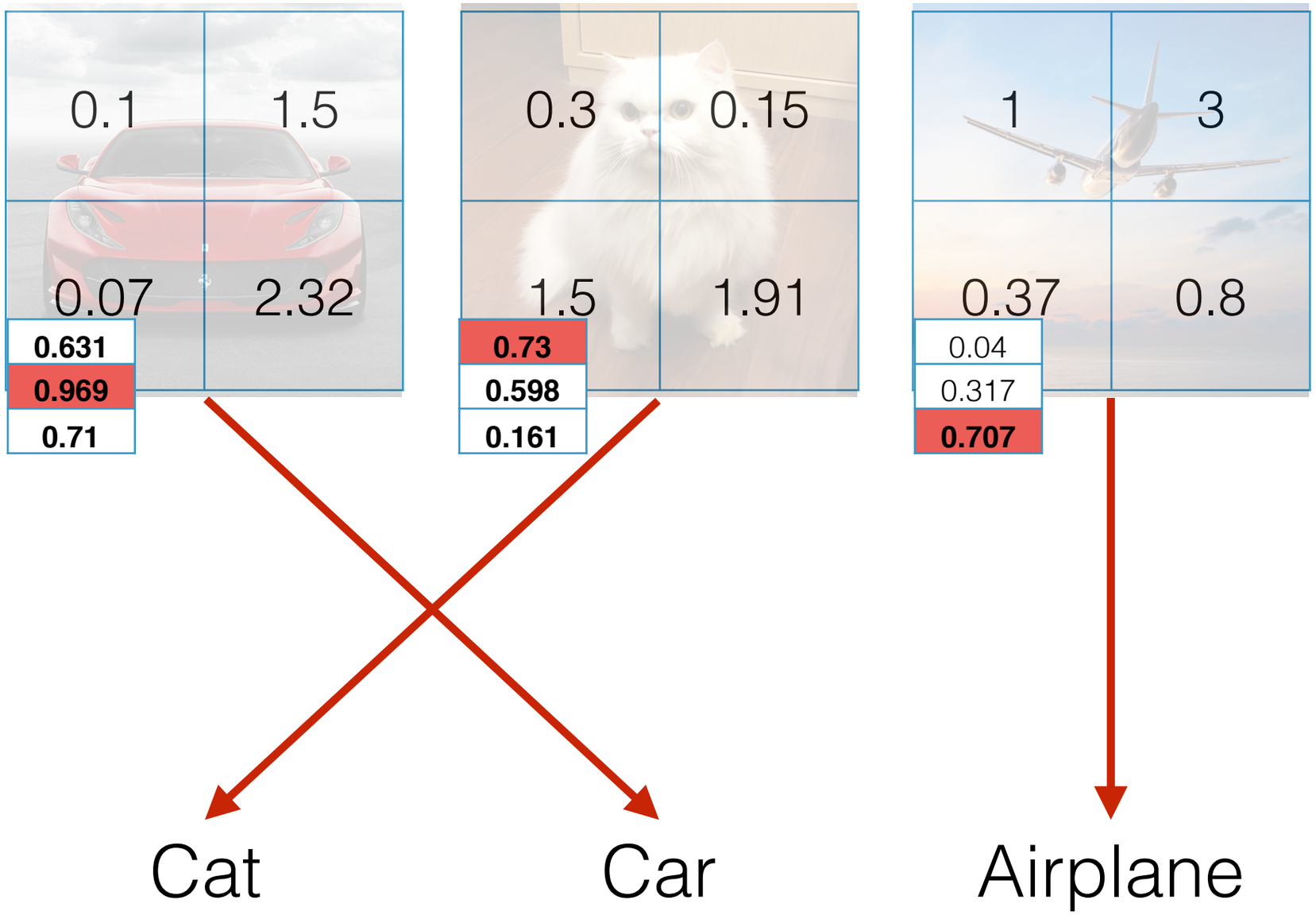
b

=

| | |
|---|---|
| 0.04 | Cat |
| 0.317 | Car |
| 0.707 | Airplane |

f(x; W,b)

# Linear Classification



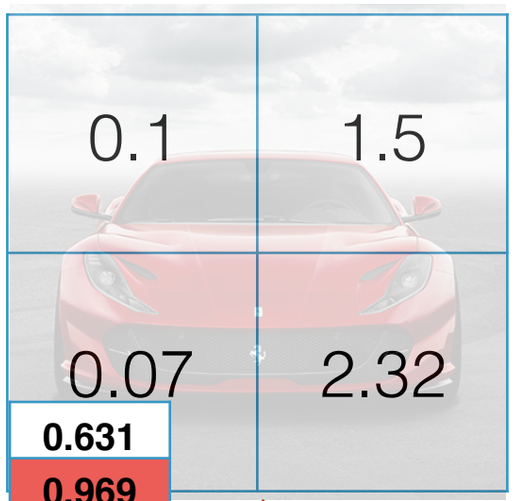

| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

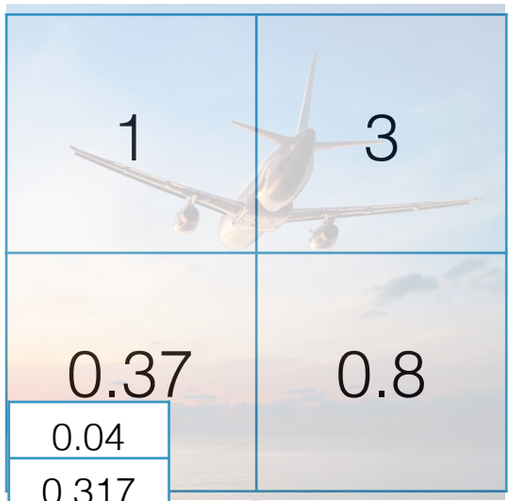
| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

# Linear Classification








| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

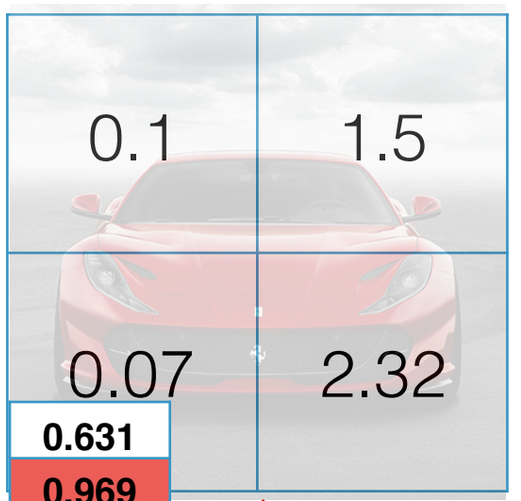
| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

Cat Car Airplane

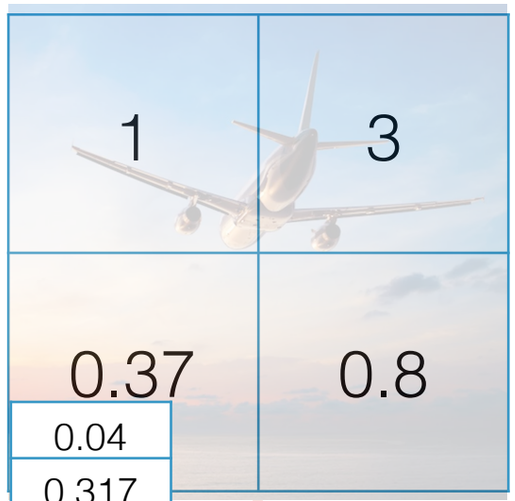
Looks perfect! Doesn't it?

# Linear Classification




**0.631**
**0.969**
**0.71**


**0.73**
**0.598**
**0.161**




0.04
0.317
**0.707**

| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

W

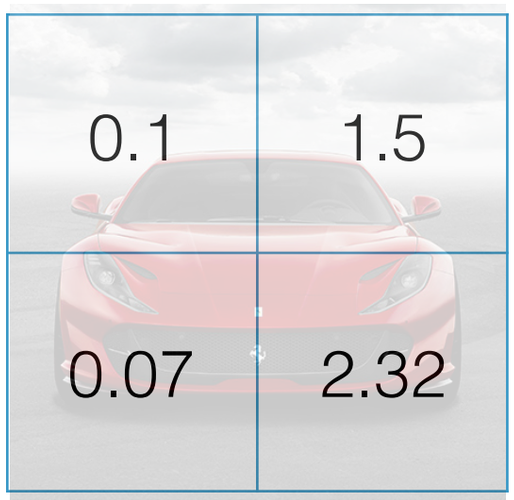
| |
|---|
| 0.15 |
| -0.1 |
| -0.2 |

b

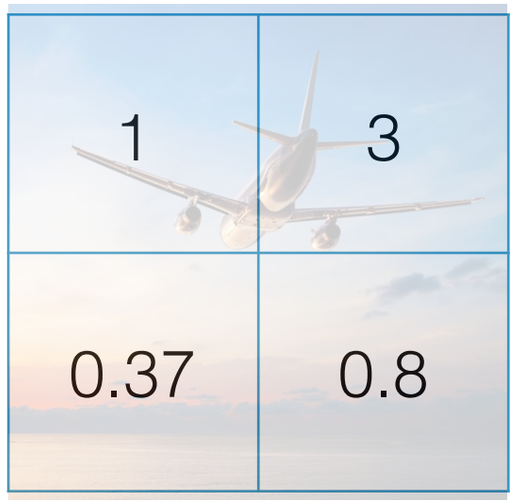
Cat Car Airplane

But… What if we have different W or b?

# Linear Classification








| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

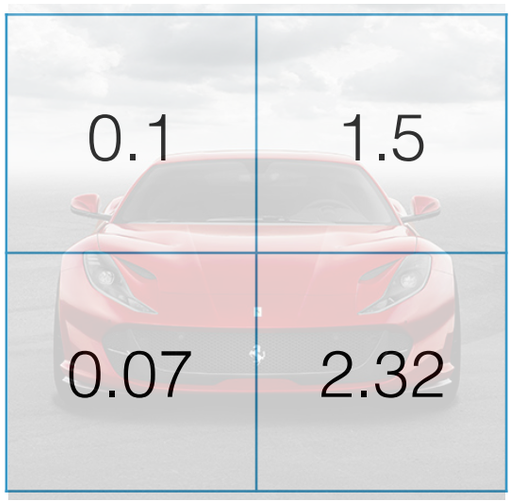
W

f(x; W,b)

| |
|---|
| 0.35 |
| 0.1 |
| 0 |

b

# Linear Classification




| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

W

| |
|---|
| 0.1 |
| 1.5 |
| 0.07 |
| 2.32 |

x

+

| |
|---|
| 0.35 |
| 0.1 |
| 0 |

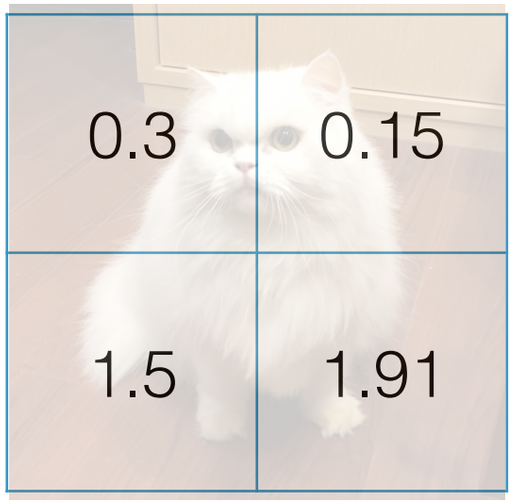
b

=

| | |
|---|---|
| 0.884 | Cat |
| 0.34 | Car |
| 0.455 | Airplane |

f(x; W,b)

# Linear Classification




| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

W

| |
|---|
| 0.3 |
| 0.15 |
| 1.5 |
| 1.91 |

x

+

| |
|---|
| 0.35 |
| 0.1 |
| 0 |

b

=

| | |
|---|---|
| 0.699 | Cat |
| 0.39 | Car |
| 0.181 | Airplane |

f(x; W,b)

# Linear Classification


| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

W

| |
|---|
| 1 |
| 3 |
| 0.37 |
| 0.8 |

x

+

| |
|---|
| 0.35 |
| 0.1 |
| 0 |

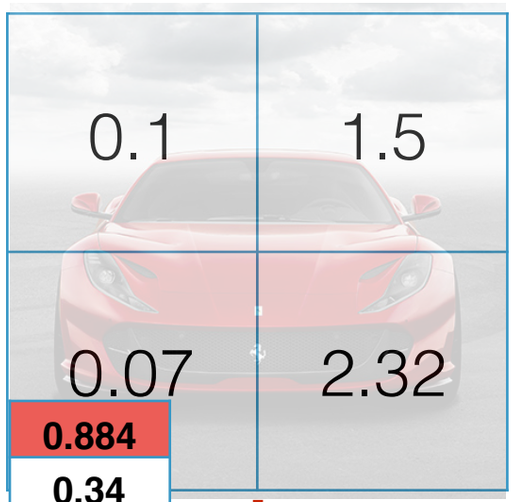
b

=

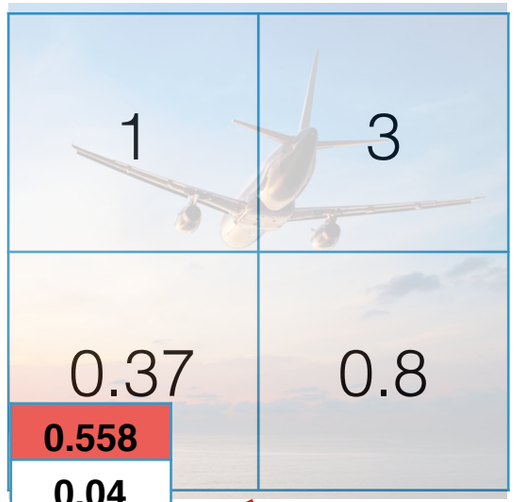
| | |
|---|---|
| 0.558 | Cat |
| 0.04 | Car |
| 0.453 | Airplane |

f(x; W,b)

# Linear Classification








| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

W

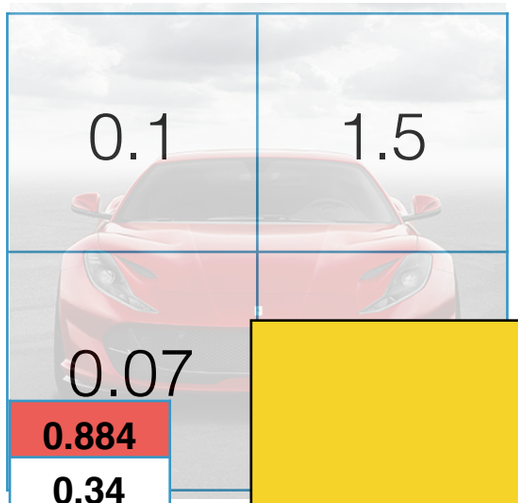
| |
|---|
| 0.35 |
| 0.1 |
| 0 |

b

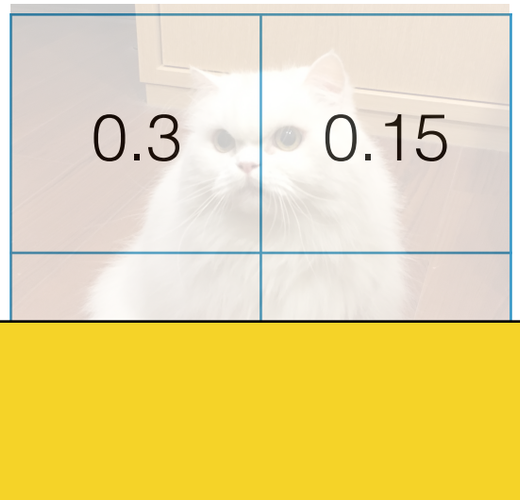
| | | |
|---|---|---|
| **0.884** | **0.699** | **0.558** |
| **0.34** | **0.39** | **0.04** |
| **0.455** | **0..181** | **0.453** |
| Cat | Car | Airplane |

Oops…

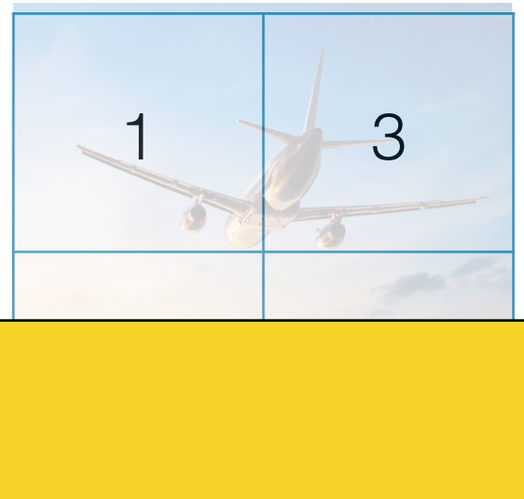
# Linear Classification



| | | | |
|---|---|---|---|
| 0.1 | 1.5 | 1.3 | 2.1 |
| 0.07 | 0.2 | 0.25 | 2.2 |
| 0.884 | 0.34 | 0.455 | 0.9 |



| | | | |
|---|---|---|---|
| 0.3 | 0.15 | 0.2 | 0.25 |
| 0.1 | 0.2 | 0.2 | 0.3 |
| 0.34 | 0.455 | 0.9 | 0.1 |



| | | | |
|---|---|---|---|
| 1 | 3 | 2.5 | 0.2 |
| 0.2 | 0.25 | 0.3 | 0.1 |
| 0.455 | 0.9 | 0.1 | 0.2 |

| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| 0.34 | 0.455 | 0.9 | 0.075 |

**W matters!**

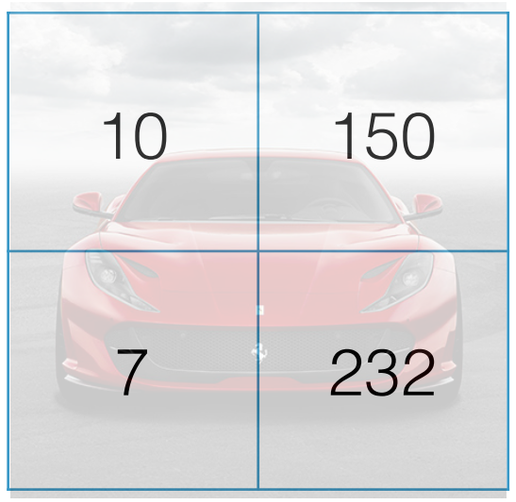
Cat Car Airplane

| |
|---|
| 0 |

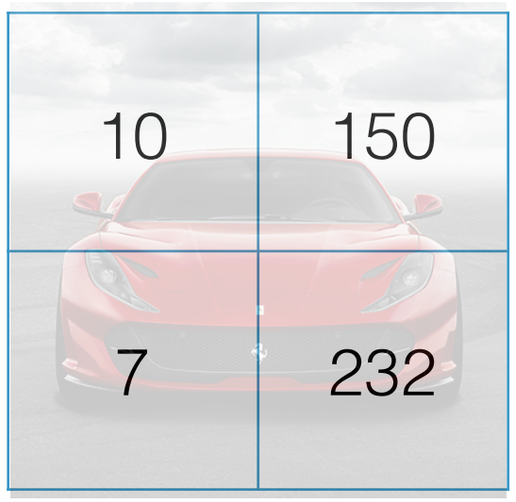
b

Oops…

# Loss functions



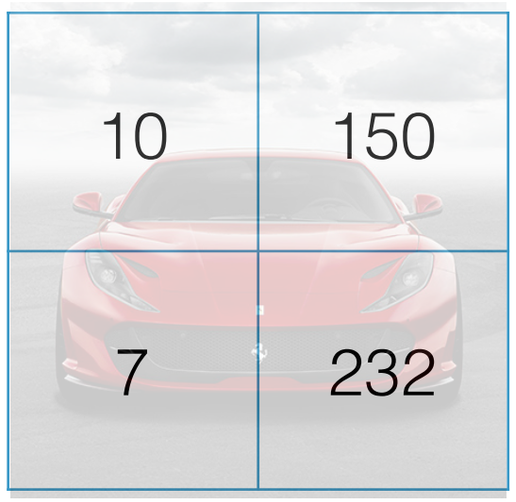
# Loss functions




| | |
|---|---|
| Cat | 0 |
| Car | 1 |
| Airplane | 0 |

Ground truth

# Loss functions




$W_1$

| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

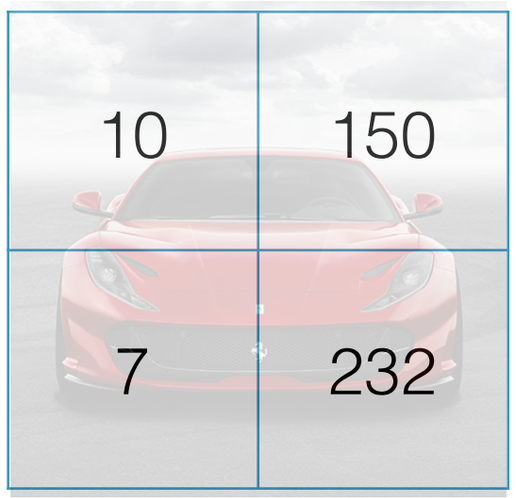
$W_2$

| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

| | |
|---|---|
| Cat | 0 |
| Car | 1 |
| Airplane | 0 |

Ground truth

# Loss functions




$W_1$

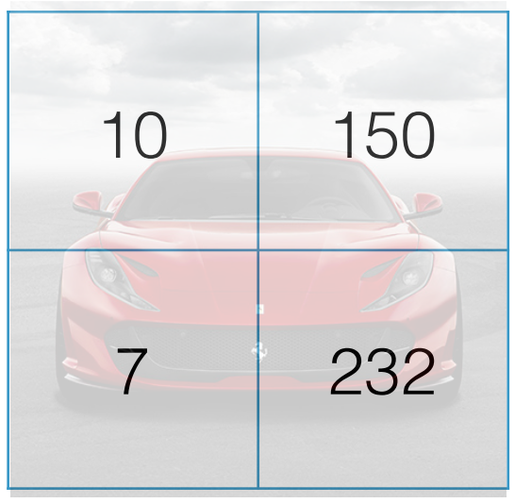
| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

$W_2$

| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

| | Ground truth | | |
|---|---|---|---|
| Cat | 0 | 0.631 | 0.884 |
| Car | 1 | 0.969 | 0.34 |
| Airplane | 0 | 0.71 | 0.455 |

# Loss functions




$W_1$

| | | | |
|---|---|---|---|
| 0.1 | -0.15 | 0 | 0.3 |
| -0.5 | 0.2 | 0.1 | 0.35 |
| -0.45 | 0.4 | 0.1 | 0.15 |

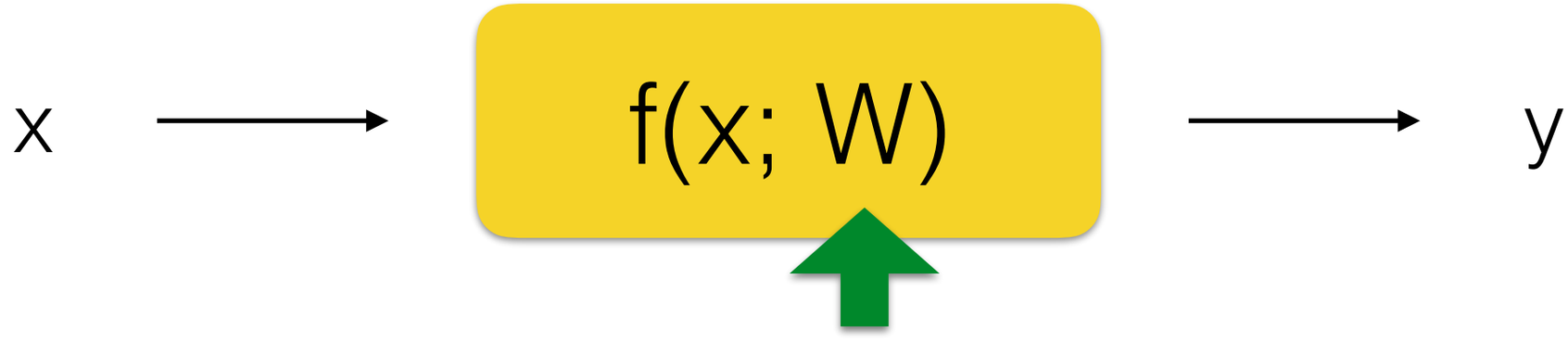
$W_2$

| | | | |
|---|---|---|---|
| -0.25 | 0.1 | 0.05 | 0.175 |
| 0.05 | -0.075 | 0 | 0.15 |
| -0.225 | 0.2 | 0.05 | 0.075 |

| | Ground truth | | |
|---|---|---|---|
| Cat | 0 | 0.631 | 0.884 |
| Car | 1 | 0.969 | 0.34 |
| Airplane | 0 | 0.71 | 0.455 |

Which one is better?

# Loss functions

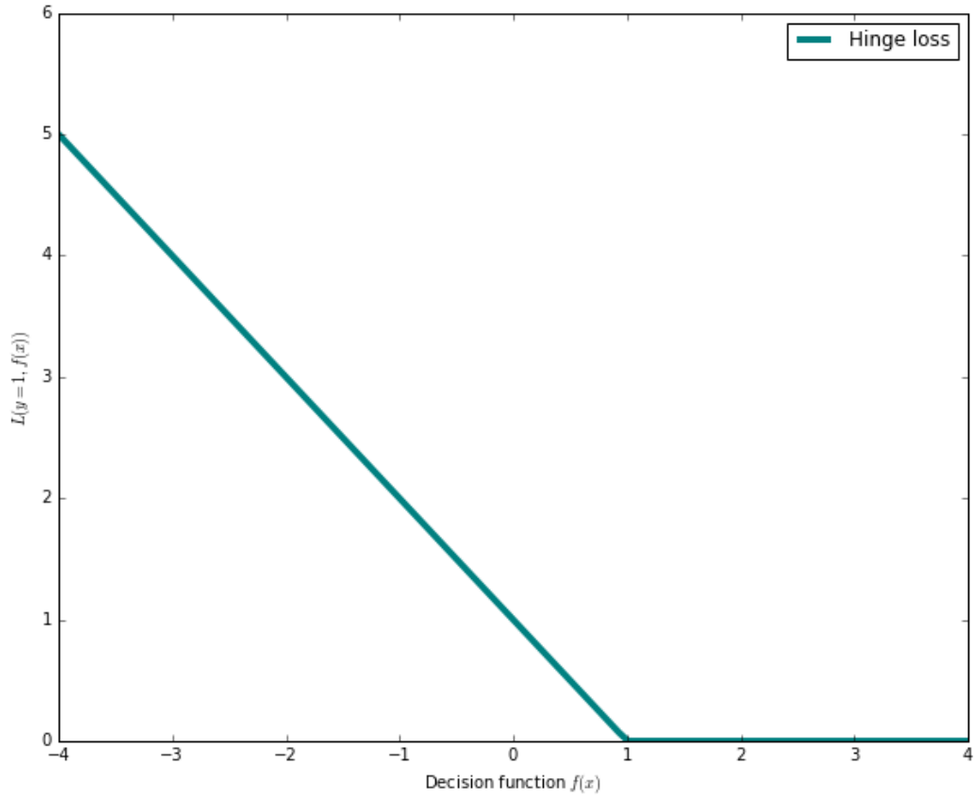



Showing W (to learn) explicitly
Parametric function

L measures how well learned W can map X to Y.

- Hinge Loss
- $L_1$, $L_2$ Loss
- Cross-Entropy

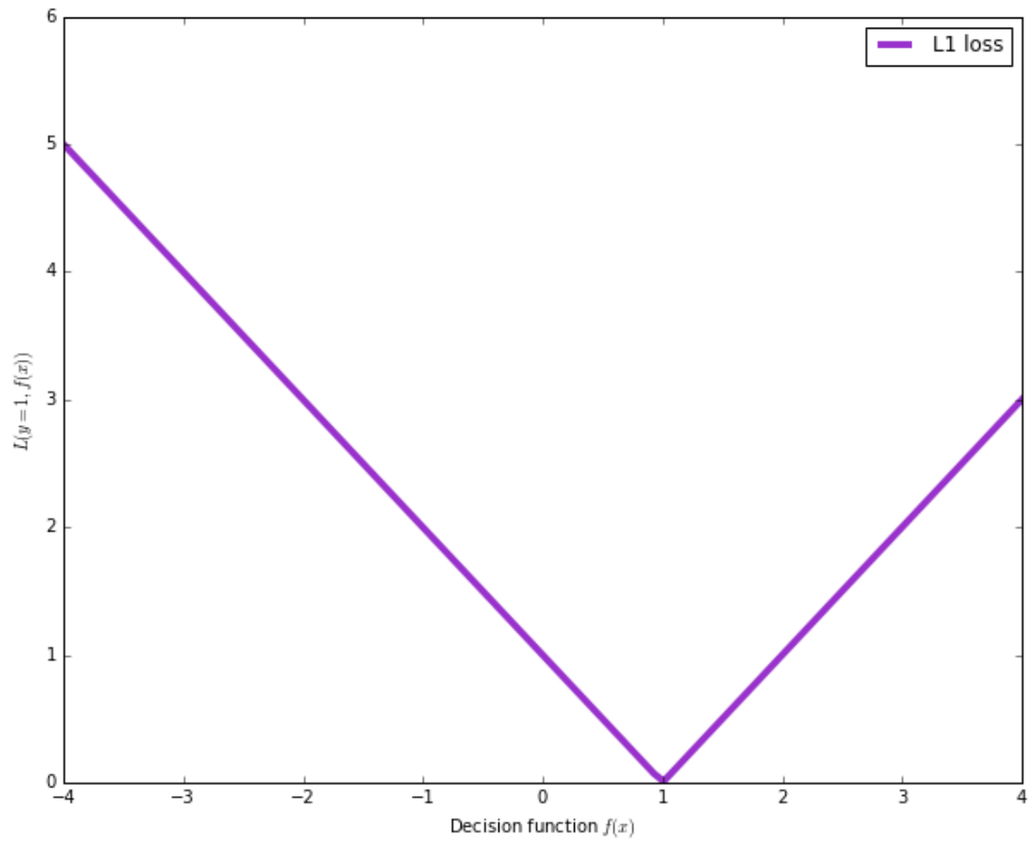
# Hinge Loss




- Given f(x; W,b) & examples ($x_i$, $y_i$), minimize:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} max(0, 1 + f(x_i; W, b)_j - f(x_i; W, b)_{y_i})$$

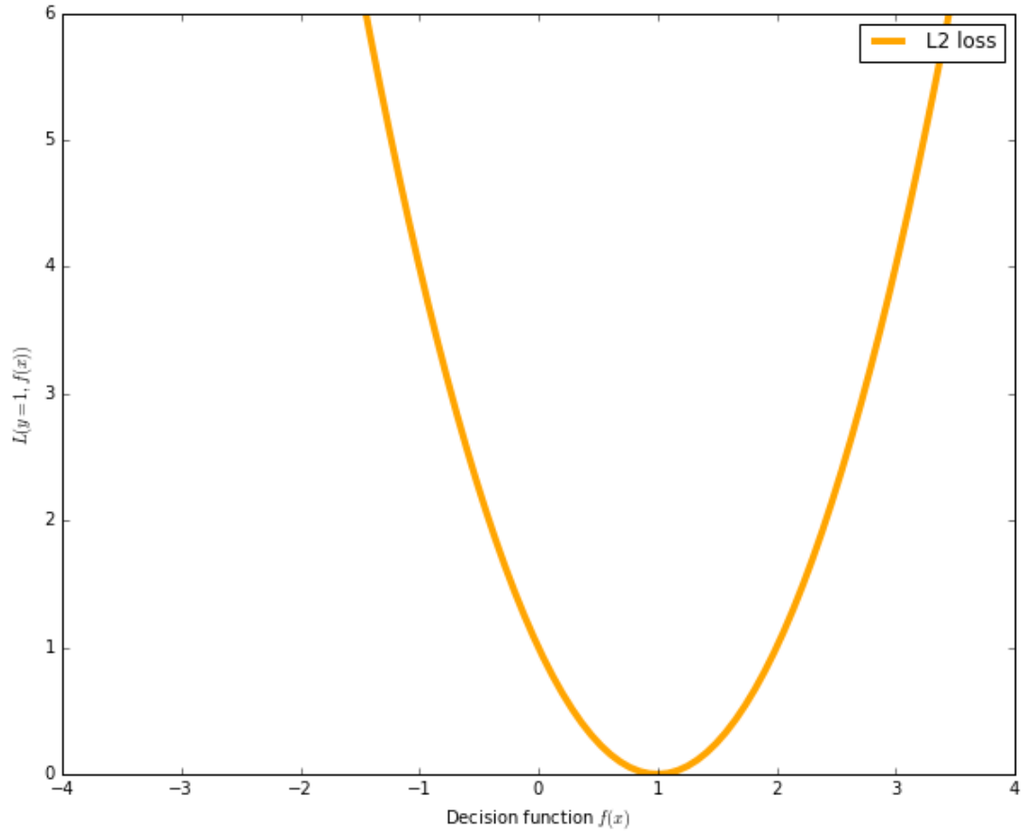
# $L_1$ Loss




- Given f(x; W,b) & examples ($x_i$, $y_i$), minimize:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_1$$

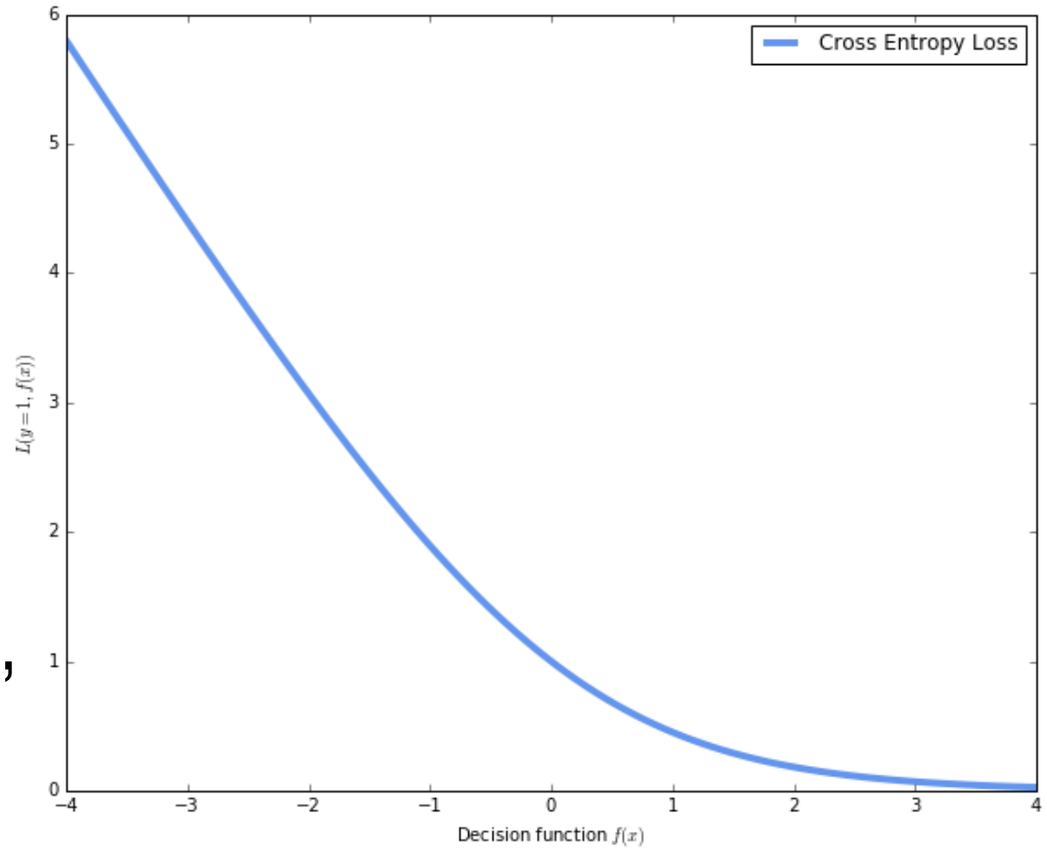
# $L_2$ Loss




- Given f(x; W,b) & examples ($x_i$, $y_i$), minimize:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2$$

# Cross-Entropy Loss

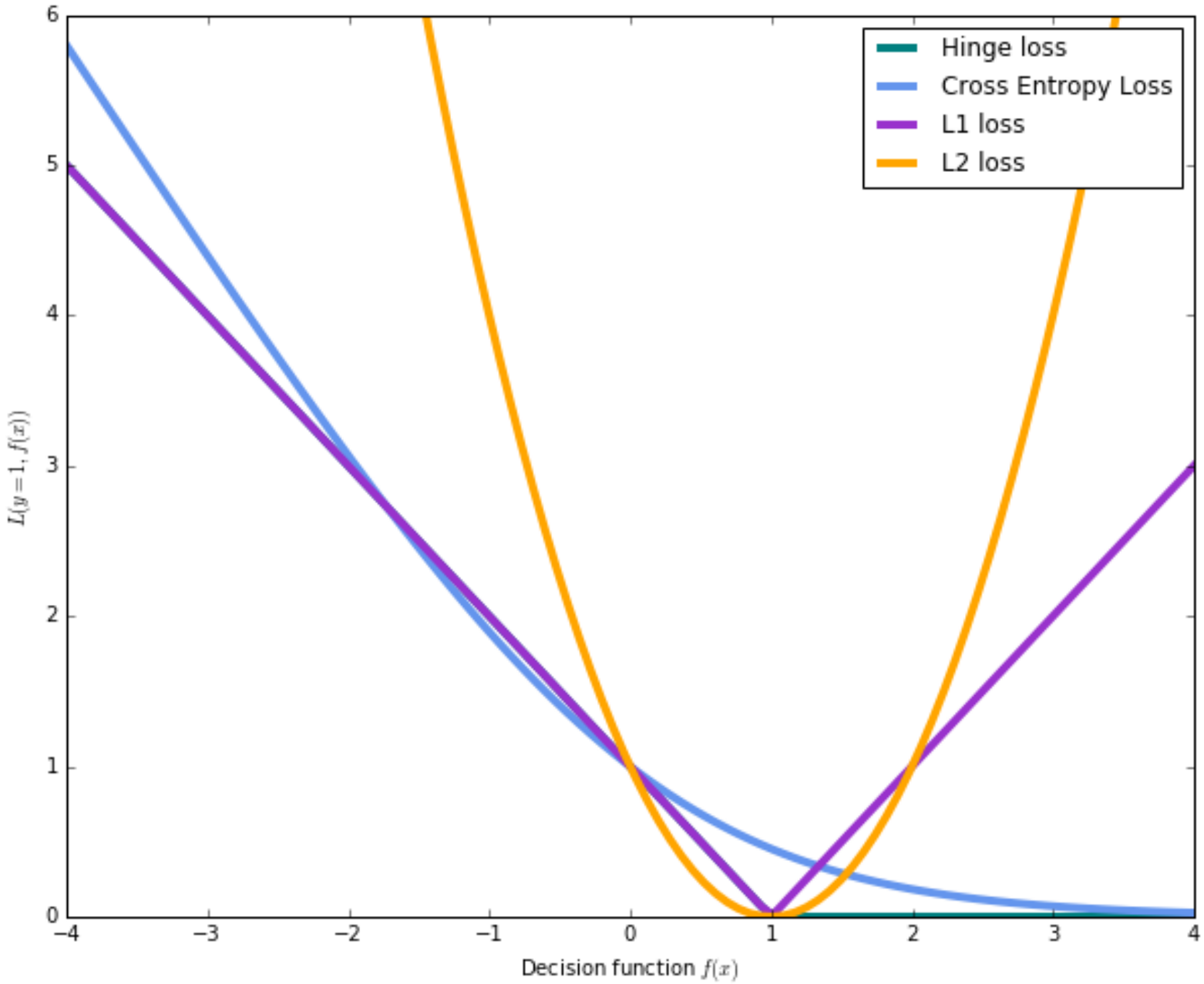



- Given f(x; W,b) & examples ($x_i$, $y_i$), minimize:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \hat{y_i}\, log(P(y_i = j\,|x_i))$$

$$P(y_i = j\,|x_i) = \frac{e^{f(x_i;W,b)}}{\sum_{j=1}^{C} e^{f(x_j;W,b)}}$$

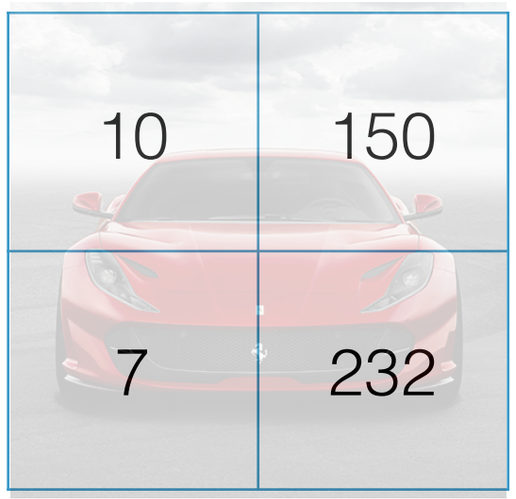
# Loss functions




Each loss function penalizes differently.

# Does a loss function matter?

# Does a loss function matter?




$W_1$

| | | | |
|---|---|---|---|
| $W_{11}$ | $W_{11}$ | $W_{11}$ | $W_{11}$ |
| $W_{11}$ | $W_{11}$ | $W_{11}$ | $W_{11}$ |
| $W_{11}$ | $W_{11}$ | $W_{11}$ | $W_{11}$ |

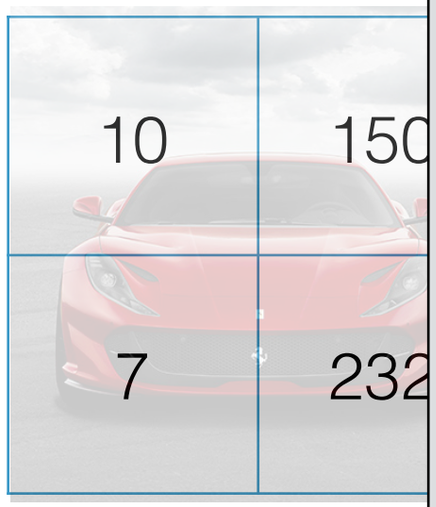
$W_2$

| | | | |
|---|---|---|---|
| $W_{21}$ | $W_{21}$ | $W_{21}$ | $W_{21}$ |
| $W_{21}$ | $W_{21}$ | $W_{21}$ | $W_{21}$ |
| $W_{21}$ | $W_{21}$ | $W_{21}$ | $W_{21}$ |

| | Ground truth | | |
|---|---|---|---|
| Cat | 0 | 0 | 0.07 |
| Car | 1 | 1.2 | 0.9 |
| Airplane | 0 | 0 | 0.05 |

Which one is better?

# Does a loss function matter?




| | f(x; W1) | f(x; W2) |
|---|---|---|
| **$L_1$-loss** | 0.2 | 0.22 |

$W_2$

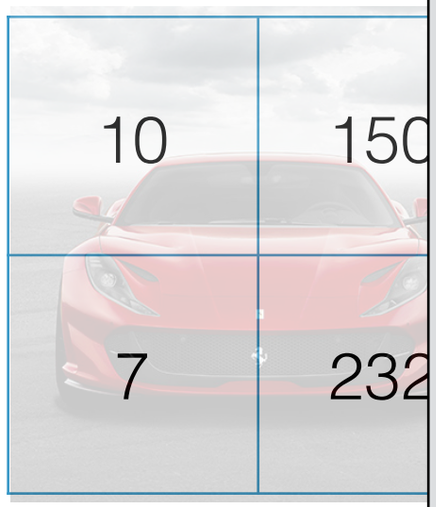
| | | |
|---|---|---|
| | $W_{21}$ | $W_{21}$ |
| | $W_{21}$ | $W_{21}$ |
| | $W_{21}$ | $W_{21}$ |

| | | | |
|---|---|---|---|
| Cat | 0 | 0 | 0.07 |
| Car | 1 | 1.2 | 0.9 |
| Airplane | 0 | 0 | 0.05 |

Ground truth

Which one is better?

# Does a loss function matter?




| | f(x; W1) | f(x; W2) |
|---|---|---|
| **$L_1$-loss** | 0.2 | 0.22 |
| **$L_2$-loss** | 0.04 | 0.0174 |

$W_2$

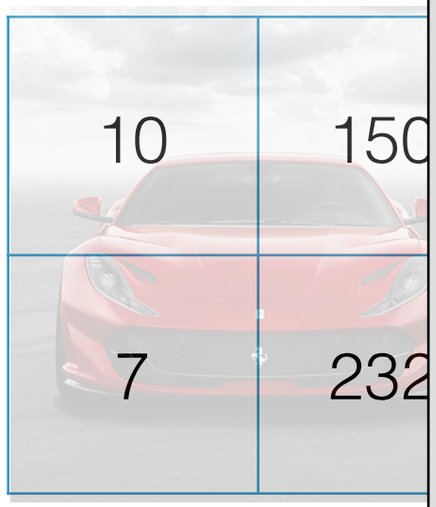
| | | |
|---|---|---|
| $W_{21}$ | $W_{21}$ | |
| $W_{21}$ | $W_{21}$ | |
| $W_{21}$ | $W_{21}$ | |

| | | | |
|---|---|---|---|
| Cat | 0 | 0 | 0.07 |
| Car | 1 | 1.2 | 0.9 |
| Airplane | 0 | 0 | 0.05 |

Ground truth

Which one is better?

# Does a loss function matter?




| | f(x; W1) | f(x; W2) |
|---|---|---|
| **$L_1$-loss** | 0.2 | 0.22 |
| **$L_2$-loss** | 0.04 | 0.0174 |

**Loss function matter!**

$W_2$

| | | |
|---|---|---|
| $W_{21}$ | $W_{21}$ | |
| $W_{21}$ | $W_{21}$ | |
| $W_{21}$ | $W_{21}$ | |

| | | | |
|---|---|---|---|
| Cat | 0 | 0 | 0.07 |
| Car | 1 | 1.2 | 0.9 |
| Airplane | 0 | 0 | 0.05 |

Ground truth

Which one is better?

# Regularization

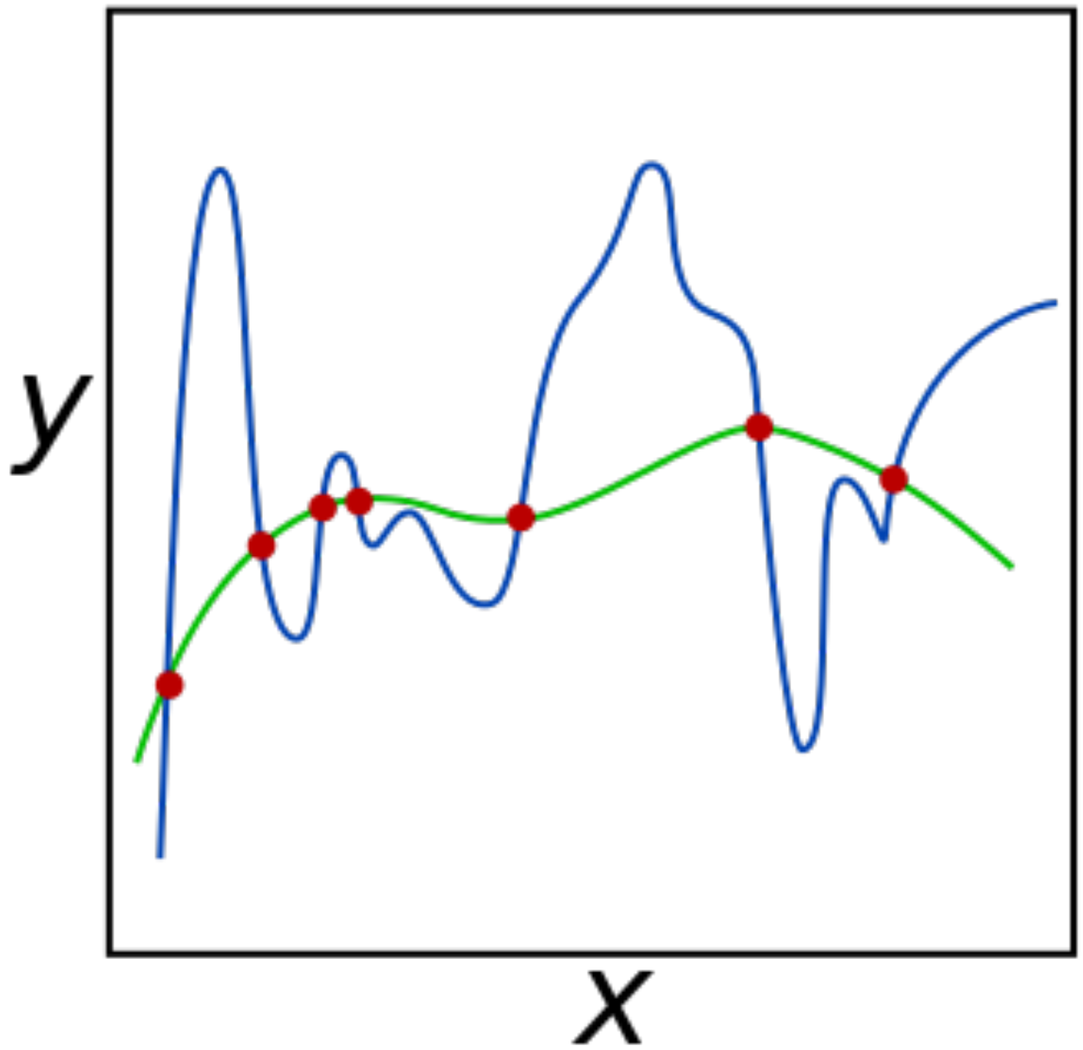
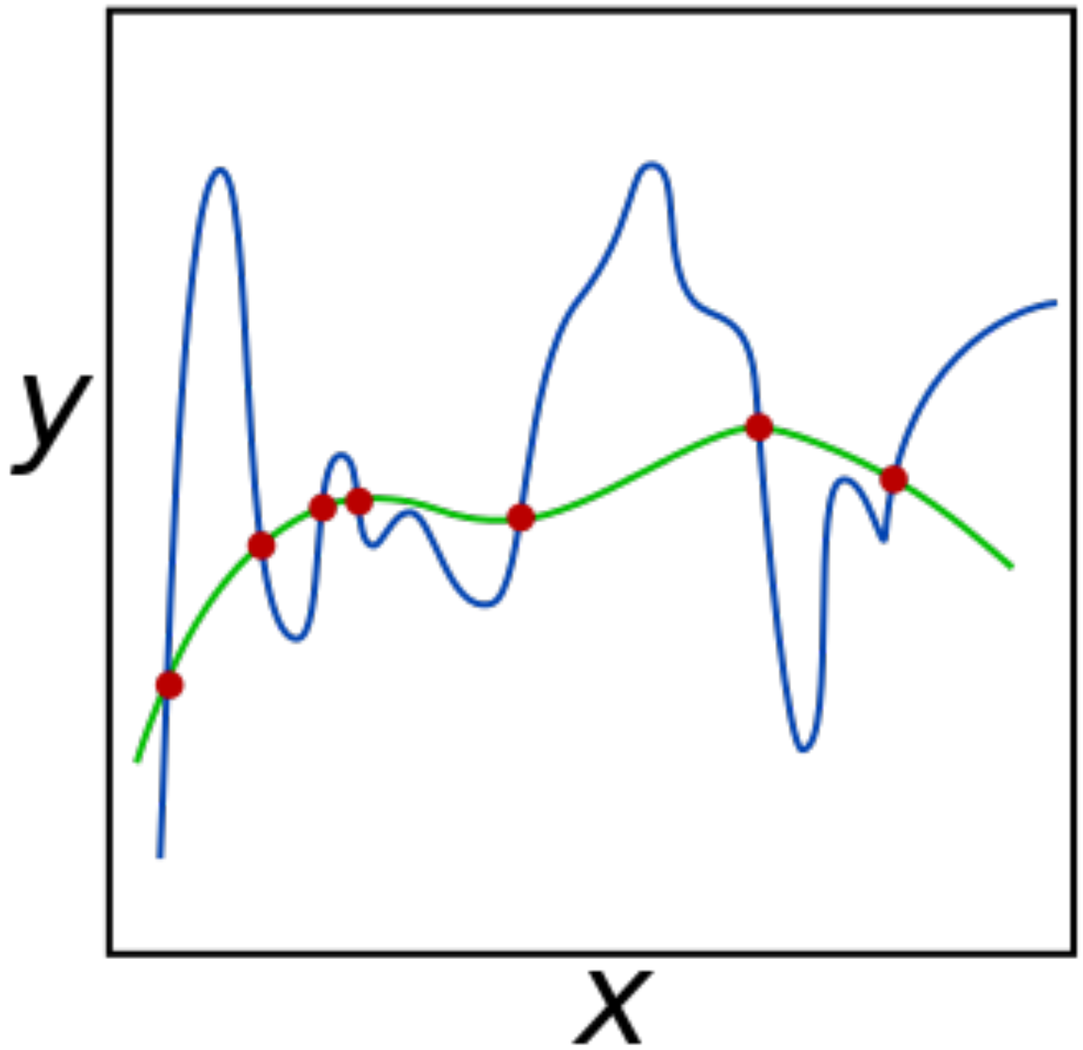



Image from Wikipedia

# Regularization




## Ockham's razor:

"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

Image from Wikipedia

# Regularization

Data loss Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2 + \lambda R(W)$$

## Ockham's razor:

"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

# Regularization

Data loss Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2 + \lambda R(W)$$

fitting to the data

## Ockham's razor:

"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

# Regularization

Data loss

Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2 + \lambda R(W)$$

fitting to the data

preferring a simple model

## Ockham's razor:

"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

# Optimization

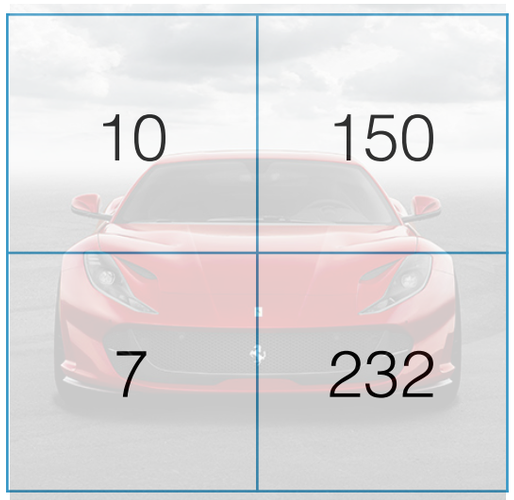
**Find W** that **minimize a loss function** (L).

# Optimization

**Find W** that **minimize a loss function** (L).

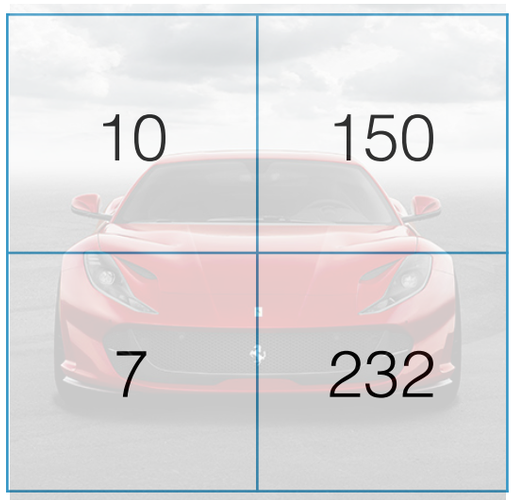
Note that we are finding W based on L, **NOT** f!

# Optimization




X

# Optimization

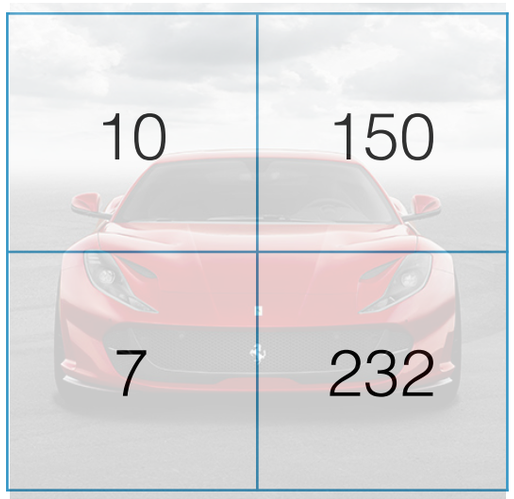



x

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# Optimization




x

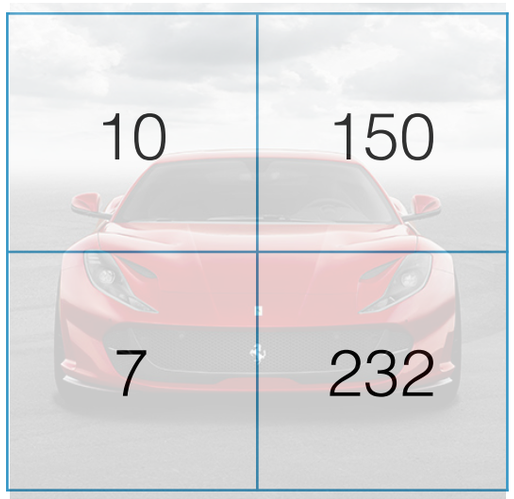
| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# Optimization




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

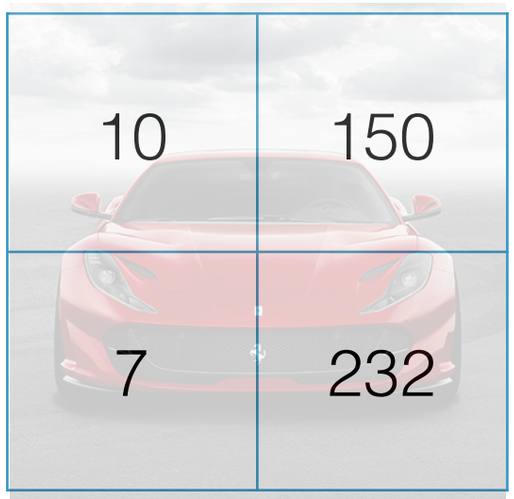
W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

f(x; W)

# Optimization




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

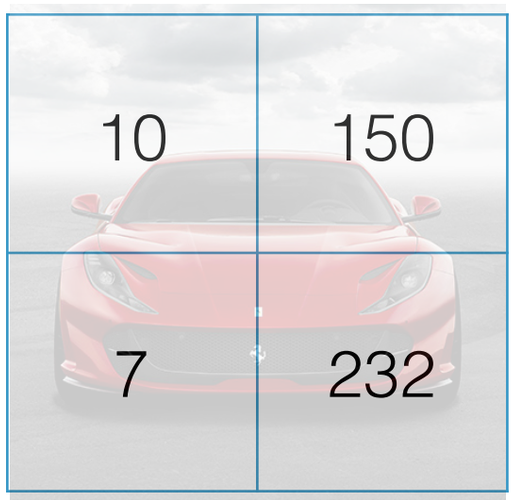
W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

f(x; W) → y*

# Optimization




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

f(x; W) → y*

**L(W; f,x,y*)**

# Optimization

How do we find W minimizing L?

* Random search
* Analytic solution
* Numerical approach (gradient descent)

ut)


**L(W; f,x,y*)**

# Optimization

## How do we find W minimizing L?

* Random search
* Analytic solution
* Numerical approach (gradient descent)

ut)

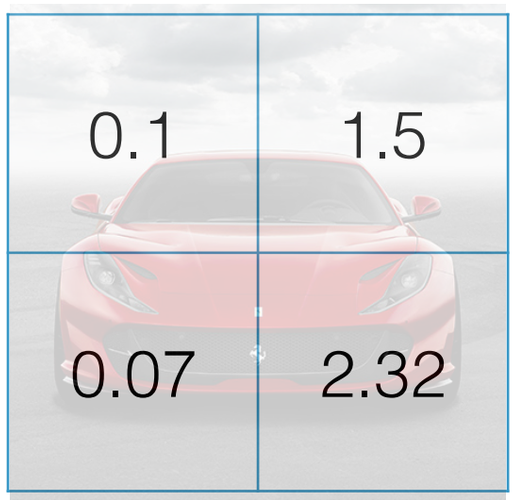
f(x; W) → y*

**L(W; f,x,y*)**

As an example, let's pick a simple loss function.

$$L(W) = || f(x; W) - y^* ||_2$$

# 1. Random Search




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

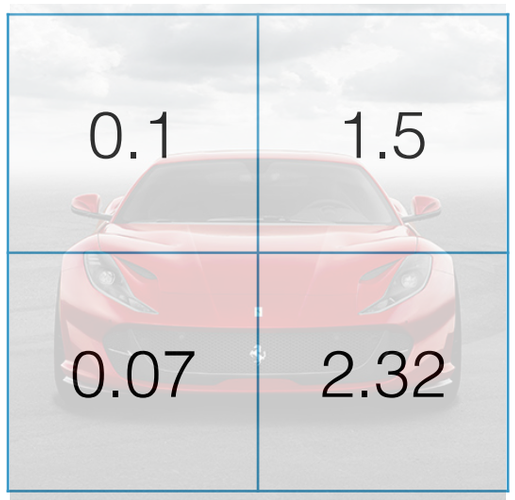
W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

$$L(W) = \| f(x; W) - y^* \|_2$$

# 1. Random Search




x

| | | | |
|---|---|---|---|
| 0.05 | 0.15 | -0.1 | 0.2 |
| -0.1 | 0.2 | 0.25 | 0.05 |
| 0.4 | -0.15 | 0.25 | 0.15 |

W

$$L(W) = \| f(x; W) - y^* \|_2$$

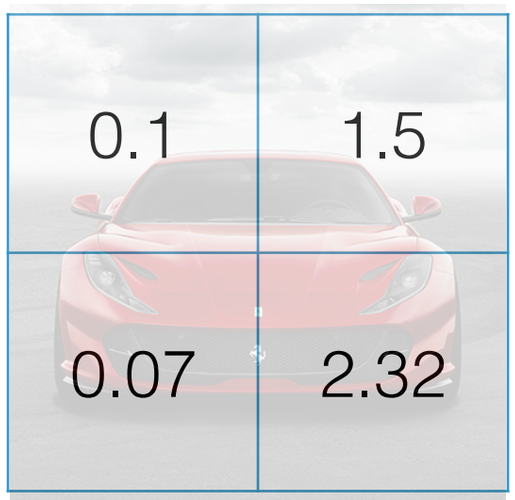
| |
|---|
| 0.987 |
| 0.223 |
| 0.38 |

f(x; W)

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 1. Random Search




x

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

$$L(W) = \| f(x; W) - y^* \|_2$$

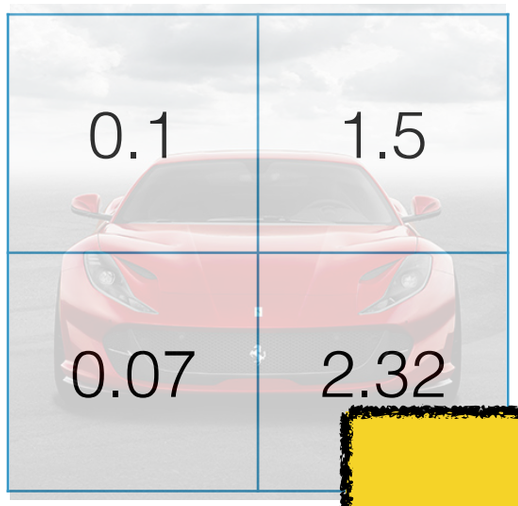
| |
|---|
| 0.416 |
| -0.118 |
| 1.167 |

f(x; W)

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 1. Random Search




| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| | | | |

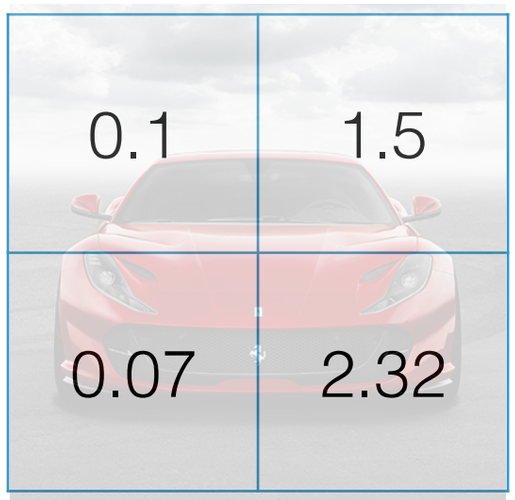
| |
|---|
| 0 |
| 1 |
| 0 |

x al output)

L(W) = |

Too slow

# 2. Analytic Solution




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

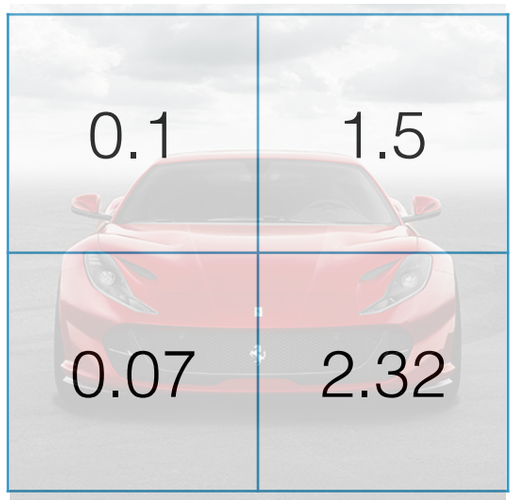
W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

$$L(W) = \| f(x; W) - y^* \|_2$$

# 2. Analytic Solution




x

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

$L(W) = \| f(x; W) - y^* \|_2$

$$\min_W L(W) = (X^T X)^{-1} X^T y^*$$

# 2. Analytic Solution

| | |
|---|---|
| 0.1 | 1.5 |
| 0.07 | 2.32 |

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| | | | |

| |
|---|
| 0 |
| 1 |
| 0 |

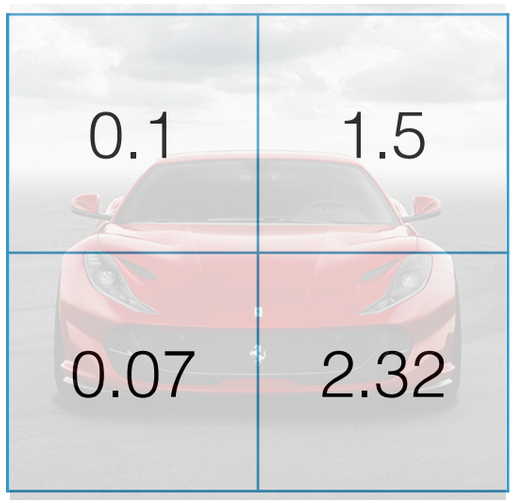
x al output)

L(W) = |

Very hard to find.

$$\min_W L(W) = (A^T A)^{-1} A^T y$$

# 3. Numerical Solution (gradient descent)




x

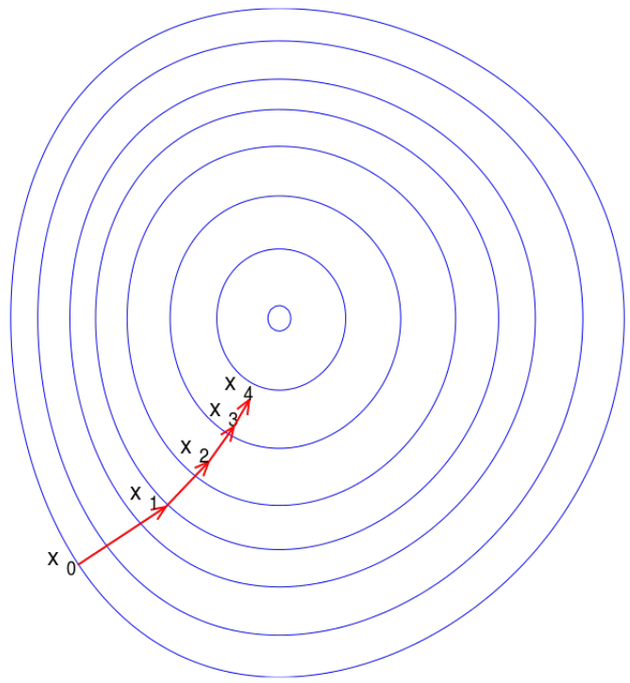
| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

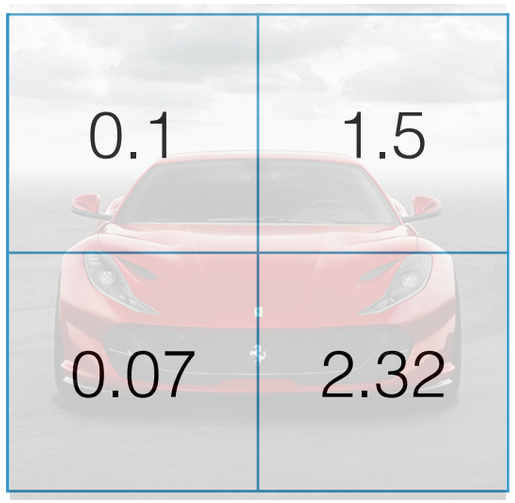
| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

$$L(W) = \| f(x; W) - y^* \|_2$$



# 3. Numerical Solution (gradient descent)




x

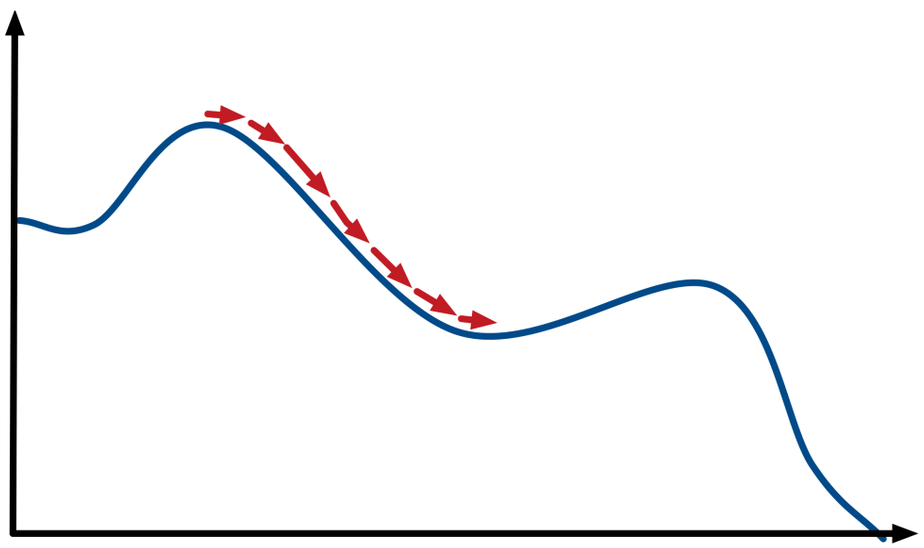
| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

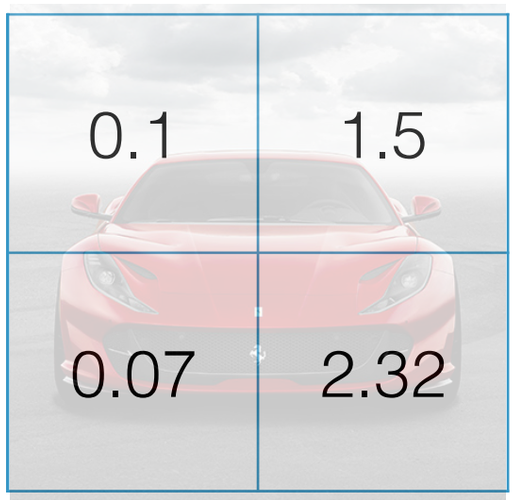
$L(W) = \| f(x; W) - y^* \|_2$



The derivative of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)




x

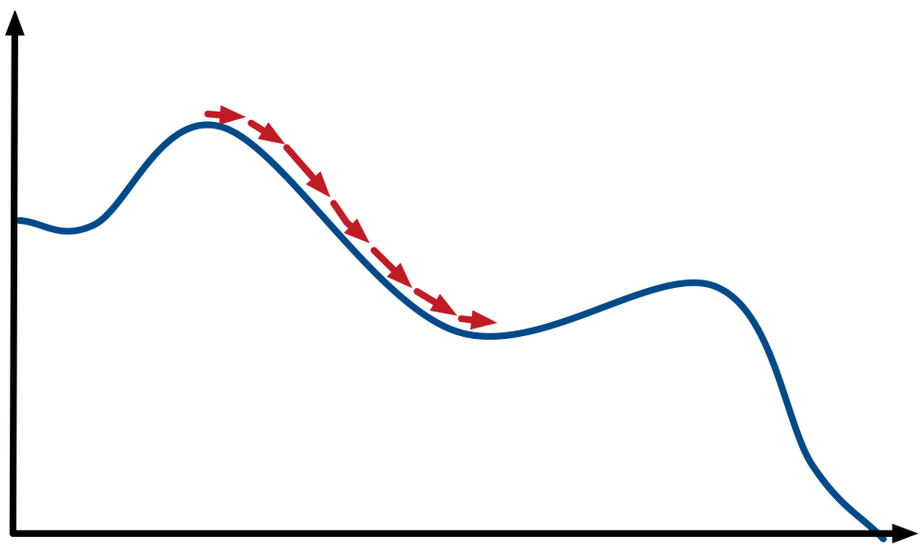
| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

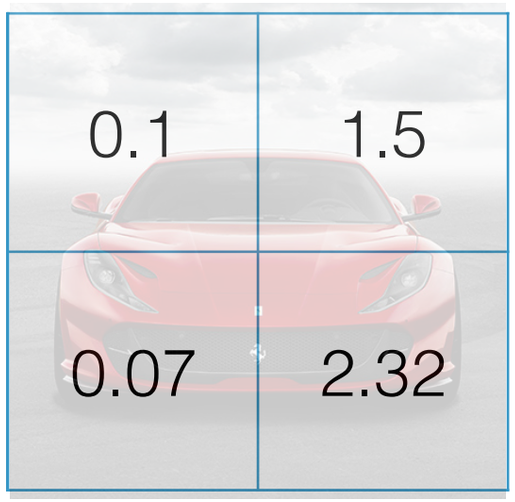
$L(W) = \| f(x; W) - y^* \|_2$



The **derivative** of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)




x

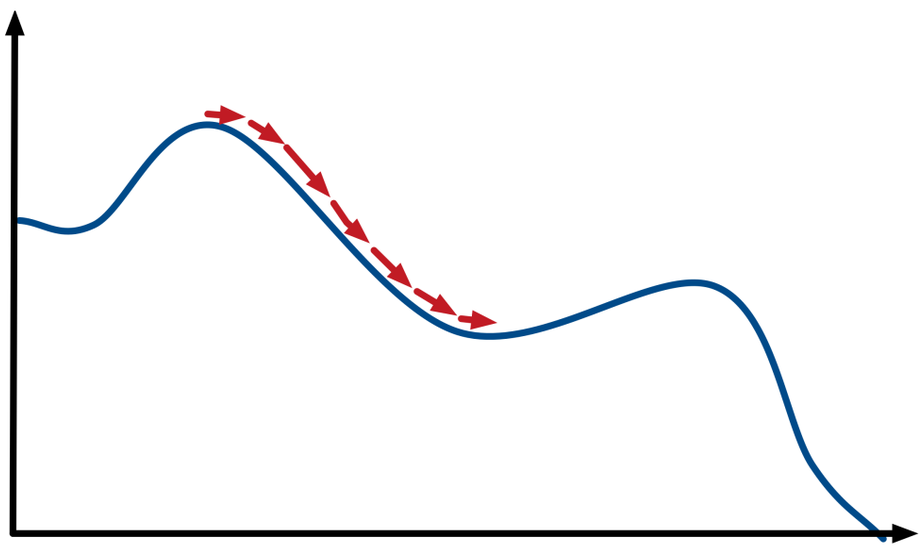
| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

W

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

$L(W) = \| f(x; W) - y^* \|_2$



The **derivative** of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The **gradient** is the vector of partial derivatives in a higher dimension

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

| |
|---|
| 0.416 |
| -0.118 |
| 1.167 |

f(x; W+h)

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

| |
|---|
| 0.416 |
| -0.118 |
| 1.167 |

f(x; W+h)

L(W+h) = 1.6693

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6693

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6693

Gradient =

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **?** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6693

Gradient = (1.6693-1.6688)/0.001 = 0.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| **-0.1+ 0.001** | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| **0.5** | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6693

Gradient = (1.6693-1.6688)/0.001 = 0.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+ 0.001 | **0.2+ 0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **?** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+ 0.001 | **0.2+ 0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **?** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

| |
|---|
| 0.418 |
| -0.118 |
| 1.167 |

f(x; W+h)

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+ 0.001 | **0.2+ 0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **?** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688

| |
|---|
| 0.418 |
| -0.118 |
| 1.167 |

f(x; W+h)

L(W+h) = 1.6697

| |
|---|
| 0 |
| 1 |
| 0 |

y* (ideal output)

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+<br>0.001 | **0.2+<br>0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **?** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6697

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+ 0.001 | **0.2+ 0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **?** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6697

Gradient = (1.6697-1.6688)/0.001 = 0.9

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)

| | | | |
|---|---|---|---|
| -0.1 | 0.2 | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W

| | | | |
|---|---|---|---|
| -0.1+ 0.001 | **0.2+ 0.001** | 0.15 | 0.05 |
| 0.25 | 0.05 | 0.2 | -0.1 |
| 0.25 | 0.15 | -0.15 | 0.4 |

W+h

| | | | |
|---|---|---|---|
| 0.5 | **0.9** | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

gradient (dW)

L(W) = 1.6688 L(W+h) = 1.6697

Gradient = (1.6697-1.6688)/0.001 = 0.9

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

# 3. Numerical Solution (gradient descent)

Various optimization strategy for stability, accuracy, or speed

- Gradient Descent
- Stochastic gradient descent
- Adam optimization
- Batch and Minibatch algorithms

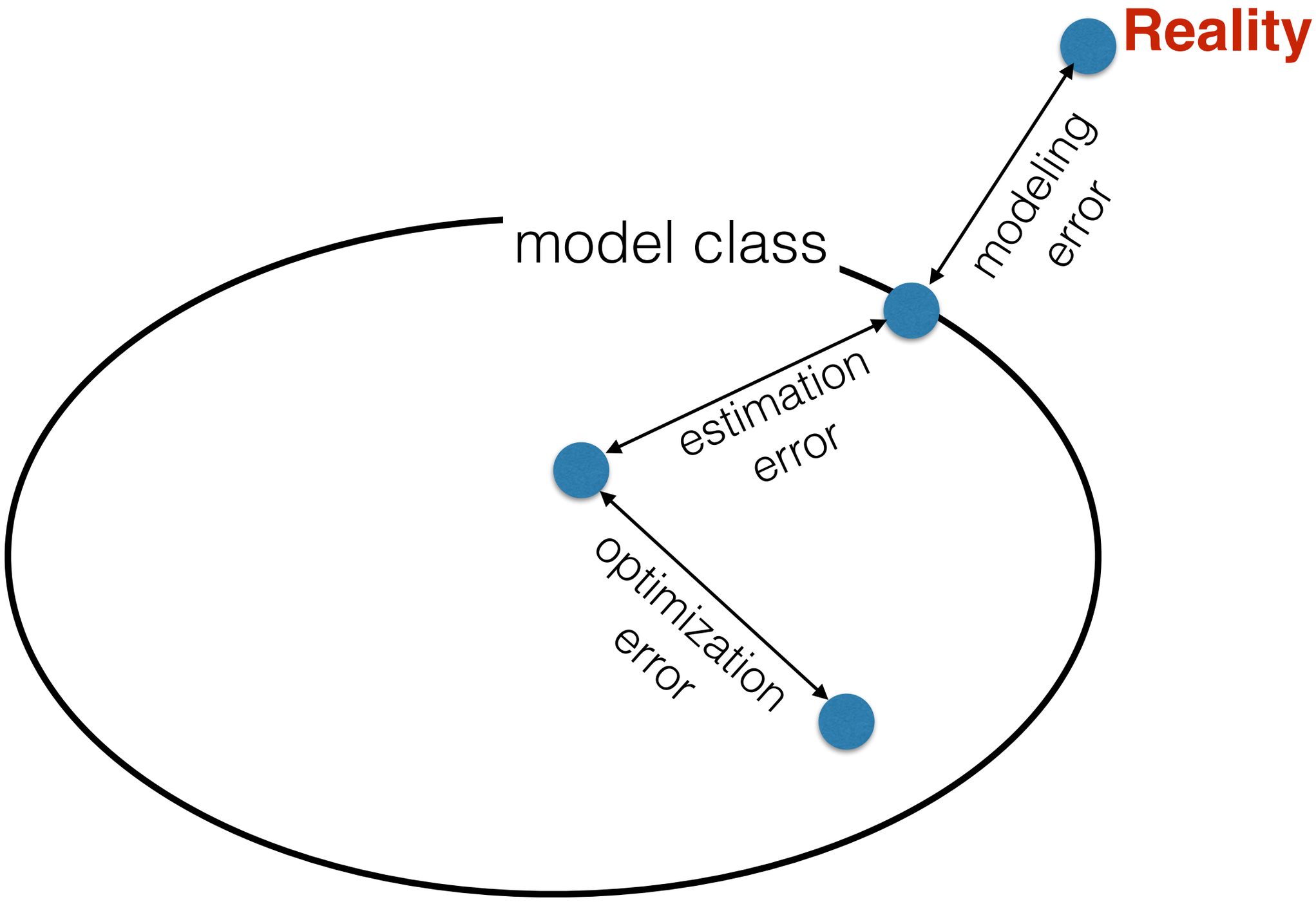
# 3. Numerical Solution (gradient descent)

- Gr
- St
- Ad
- Batch and Minibatch algorithms

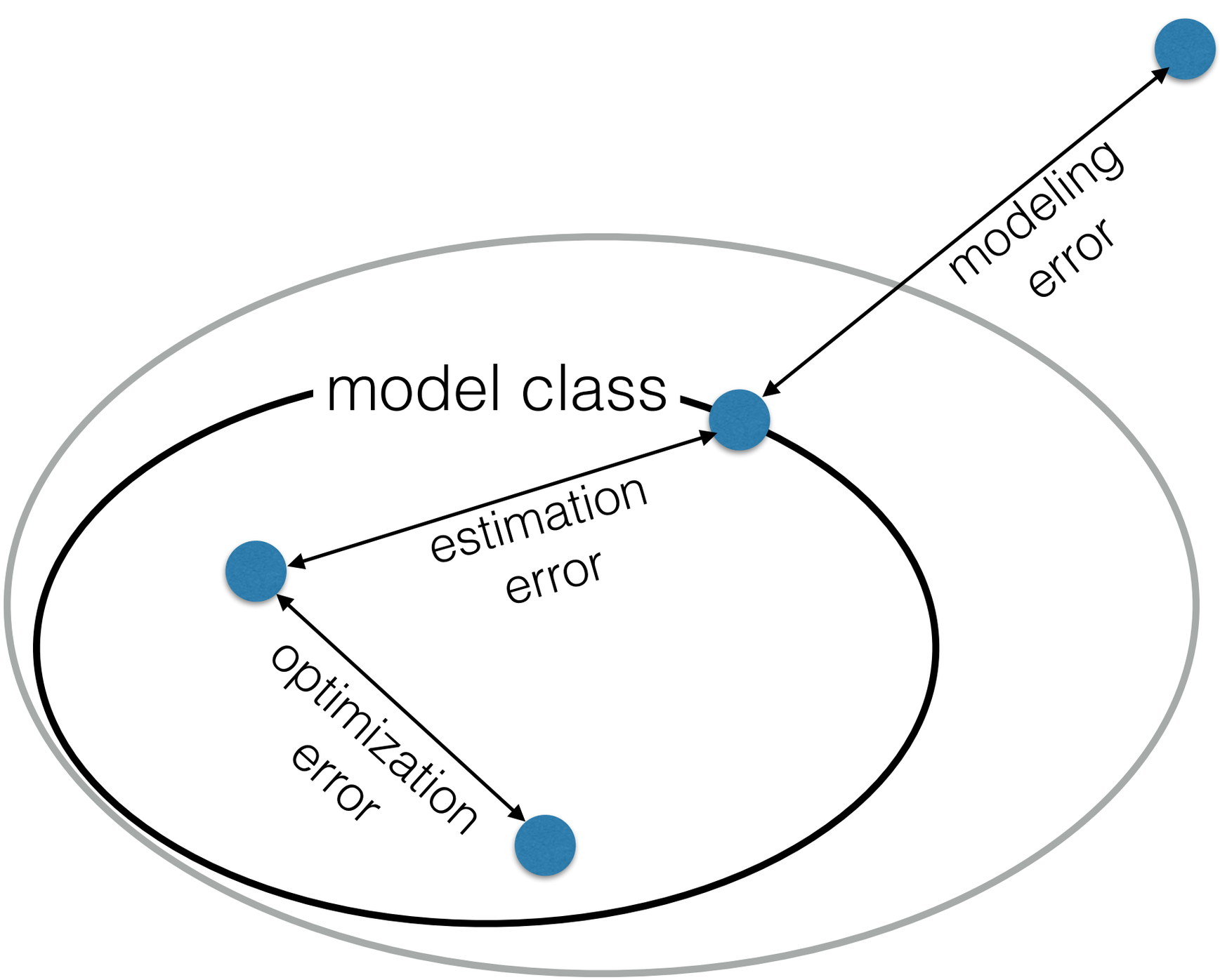
**Issues**

1. Local minima
2. Exploding gradient
3. Inexact gradient

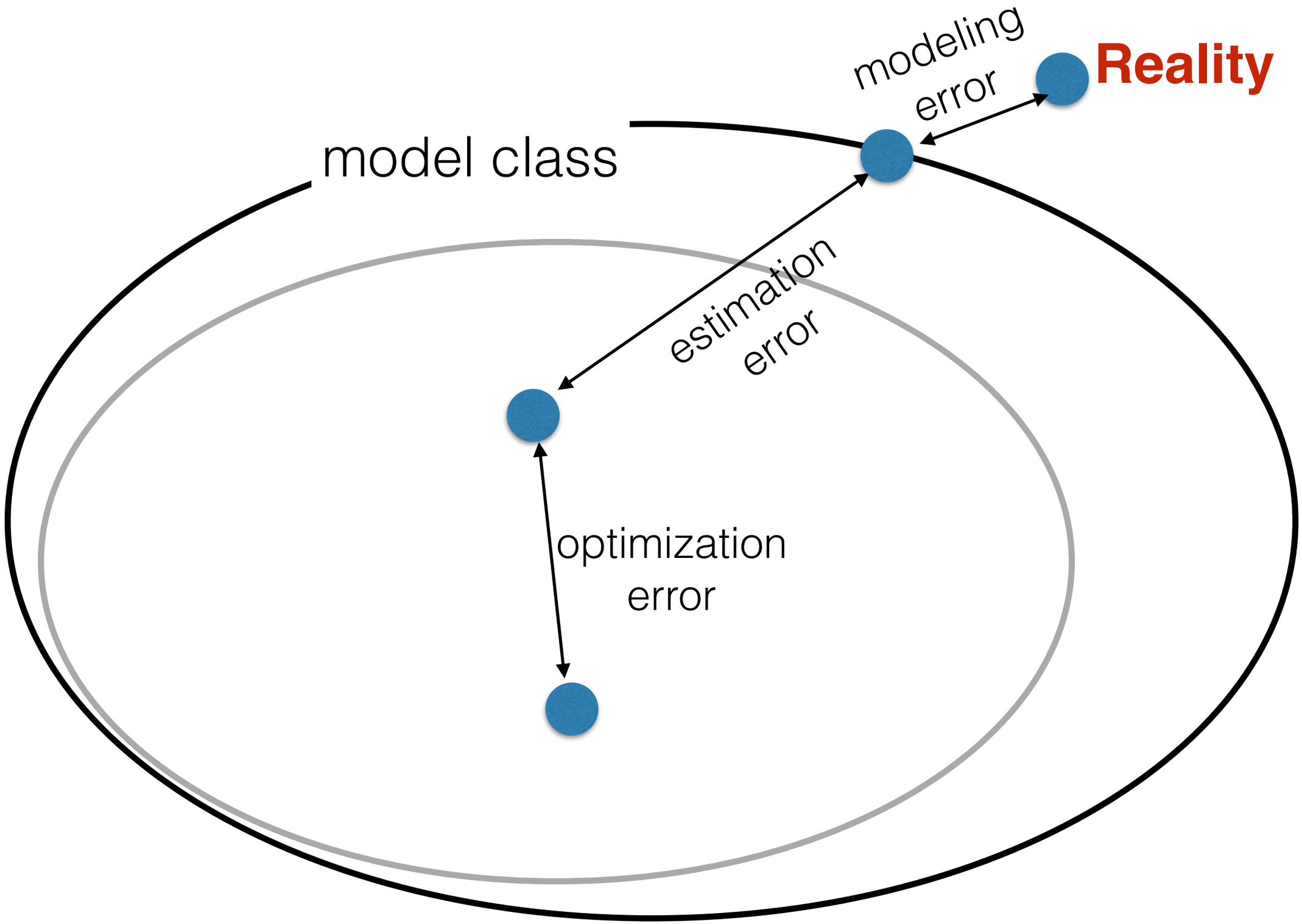
# Where do errors come from?



# Where do errors come from?



# Where do errors come from?



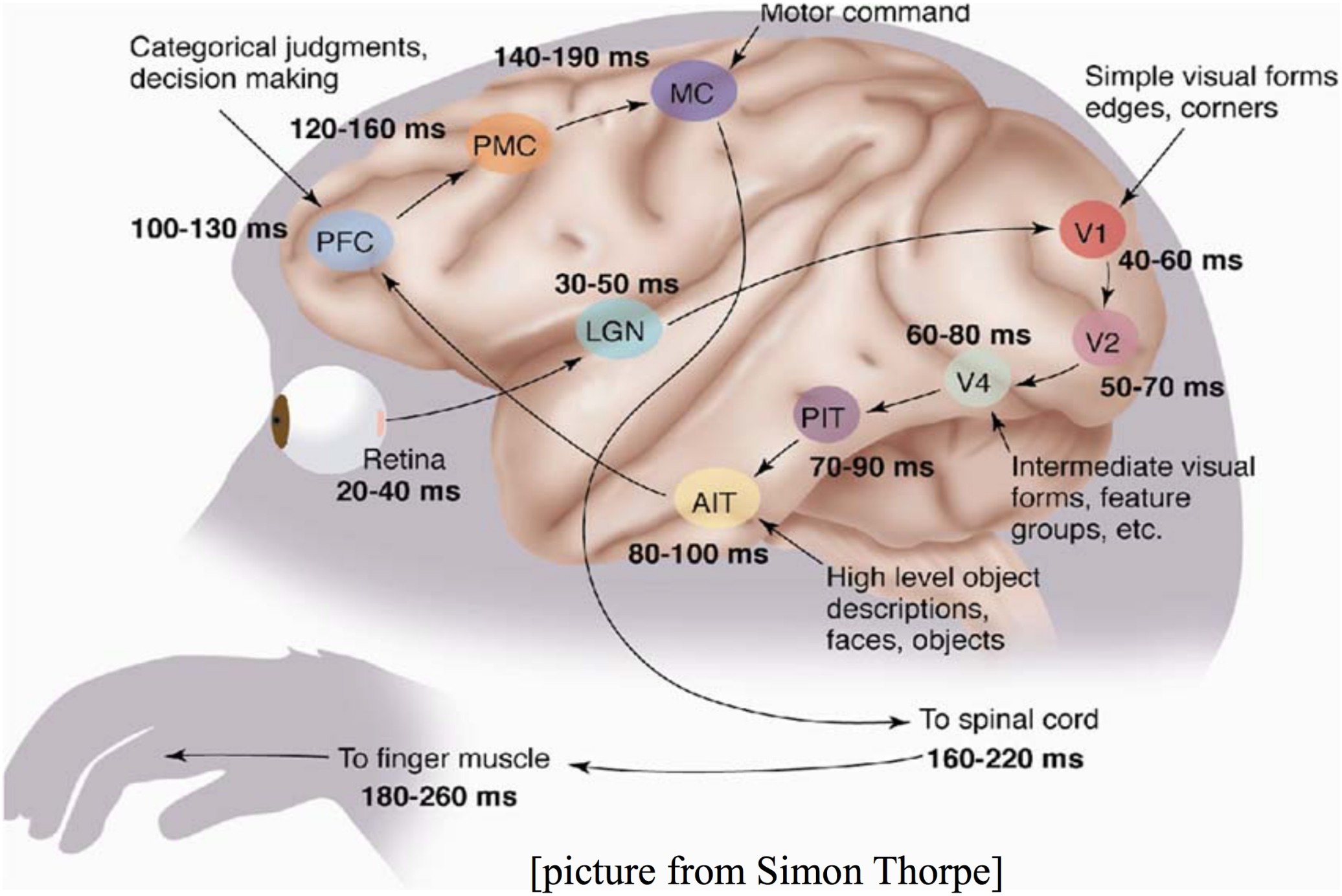
# Break Time

See you in 15 mins!

# Today's agenda

- Part 1
  - Loss function & Optimization
- Part 2
  - Neural Networks
- Part 3
  - Project Discussion

# Deep Learning is motivated by human brain




[picture from Simon Thorpe]

Slide credit: Marc'Aurelio Ranzato, Yann LeCun

# Neuron

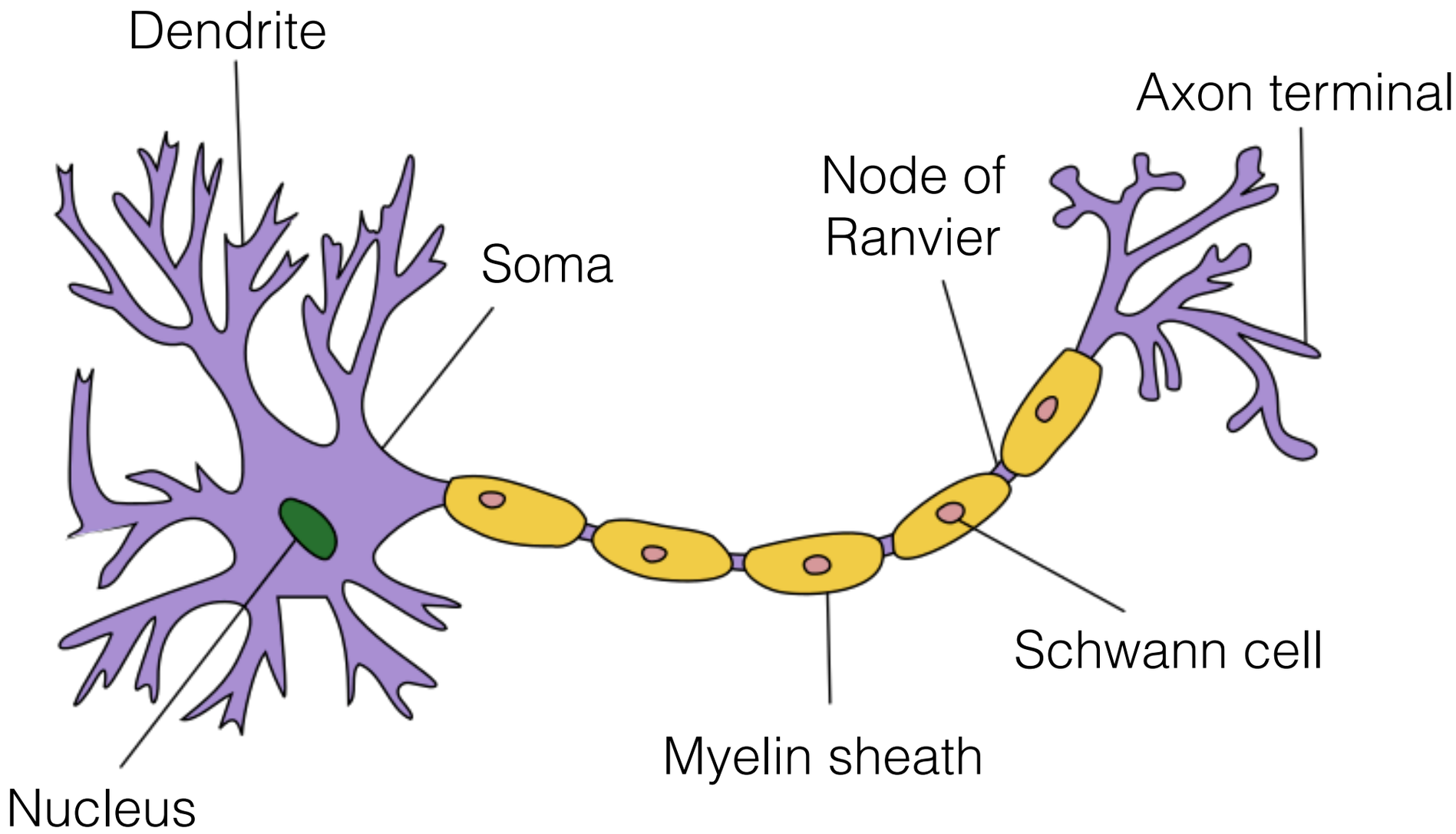



Image from Wikipedia

# Neuron

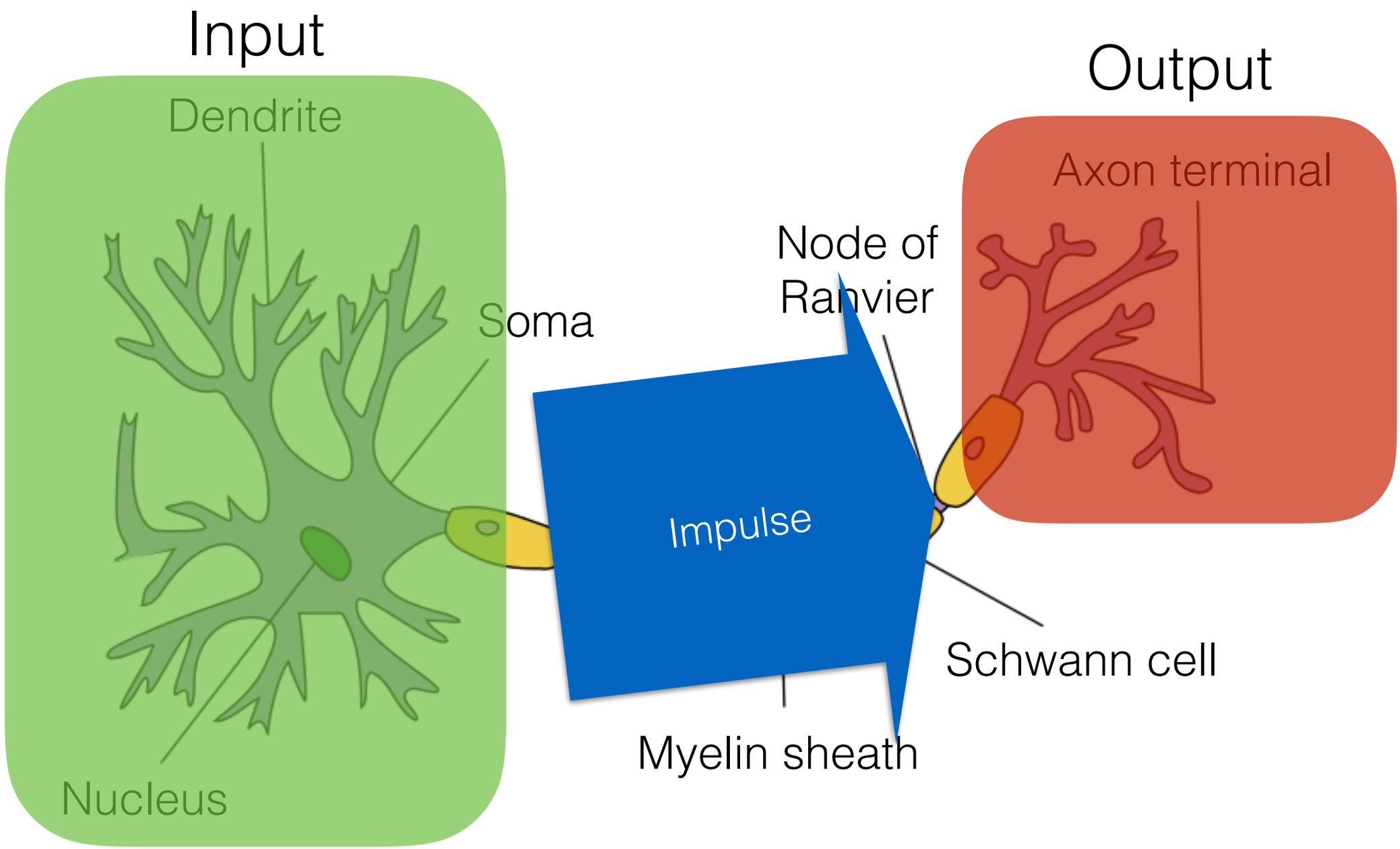
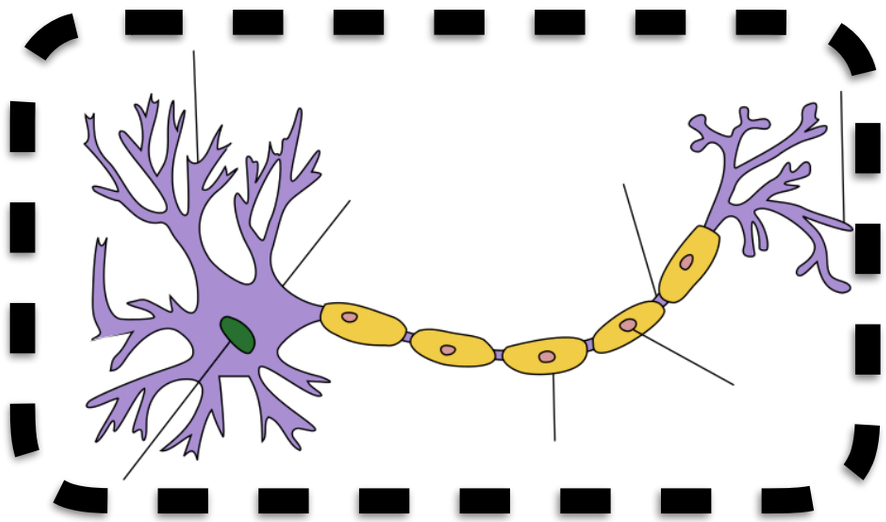
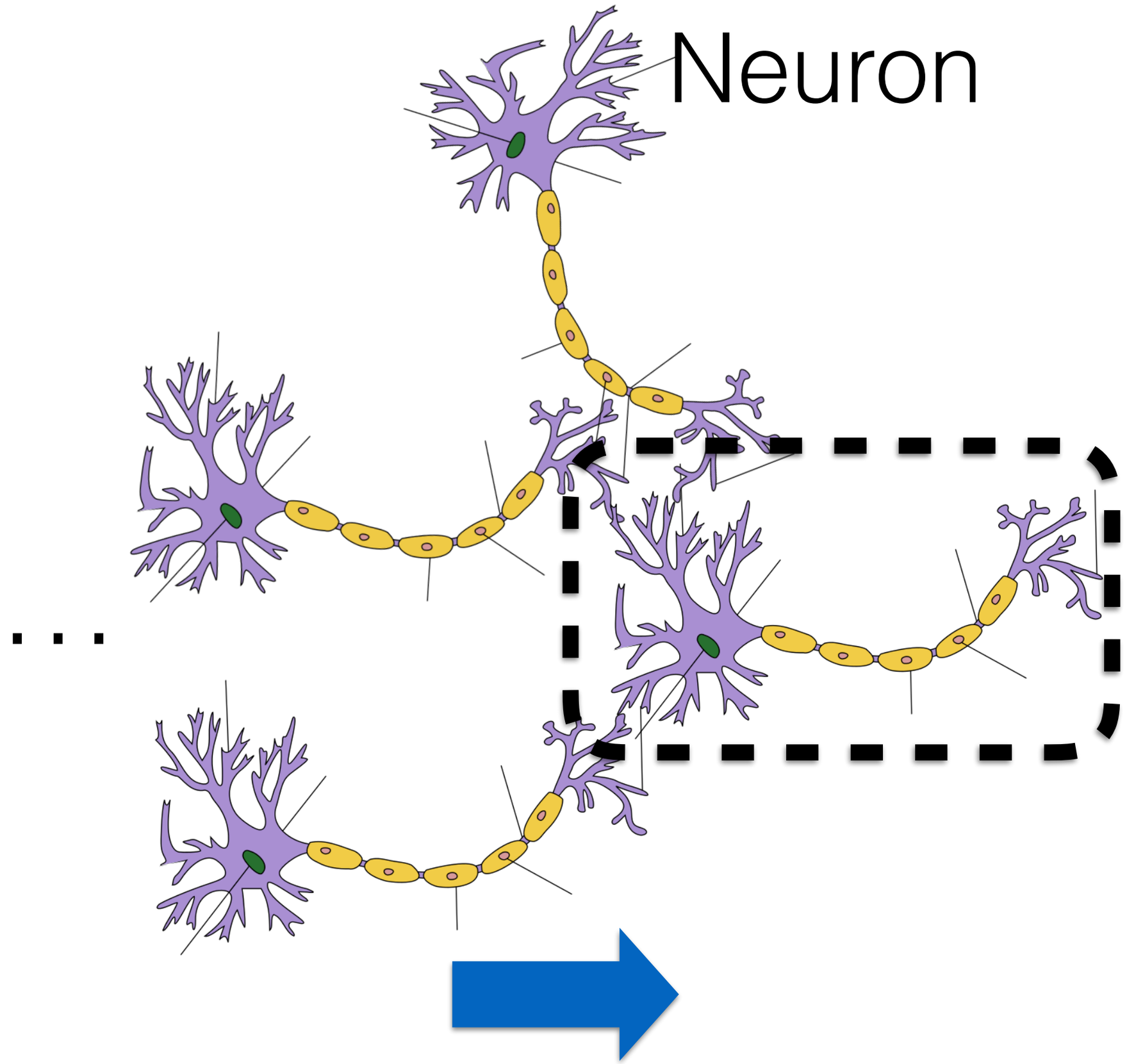



Image from Wikipedia

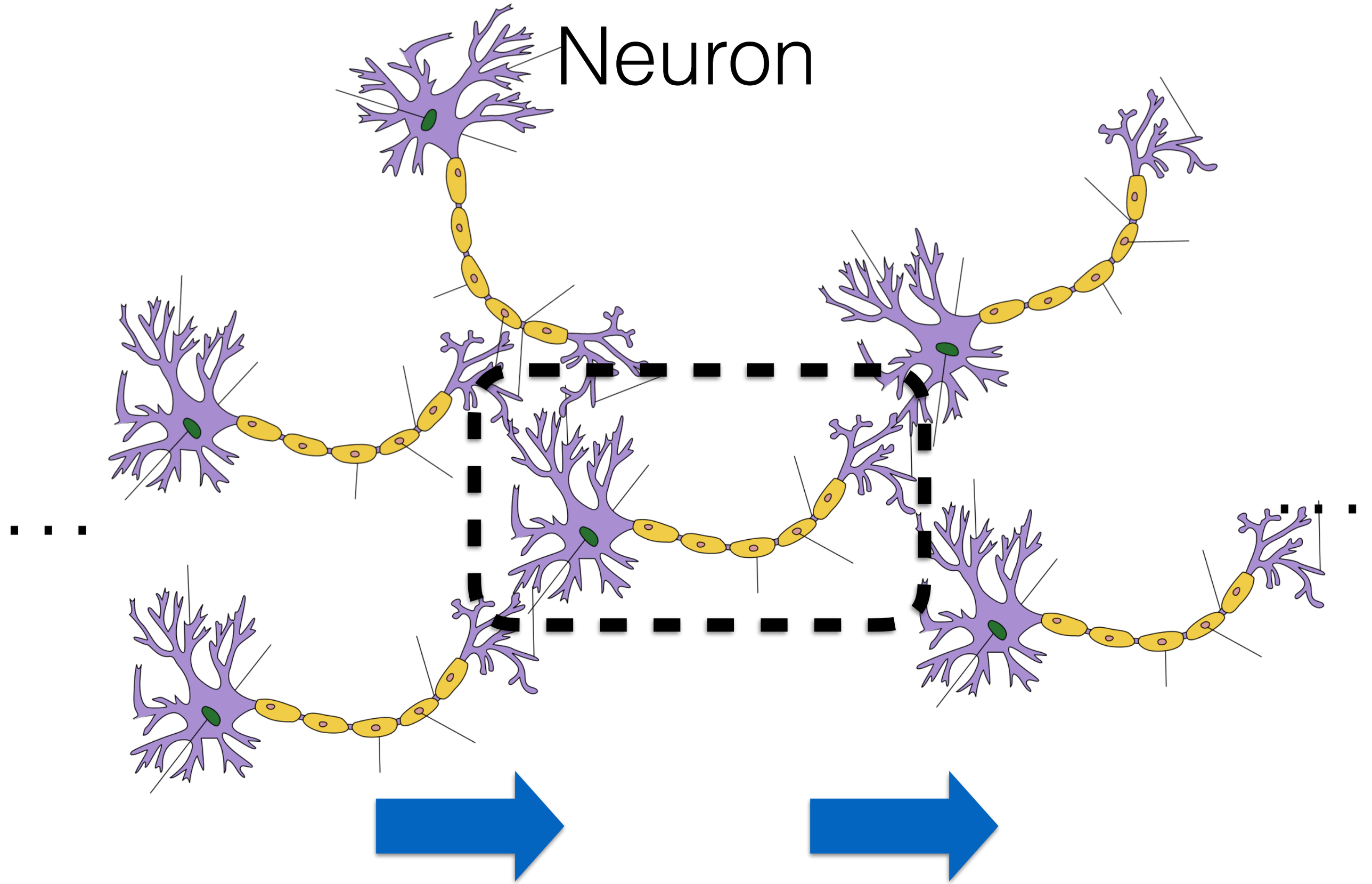
# Neuron



# Neuron

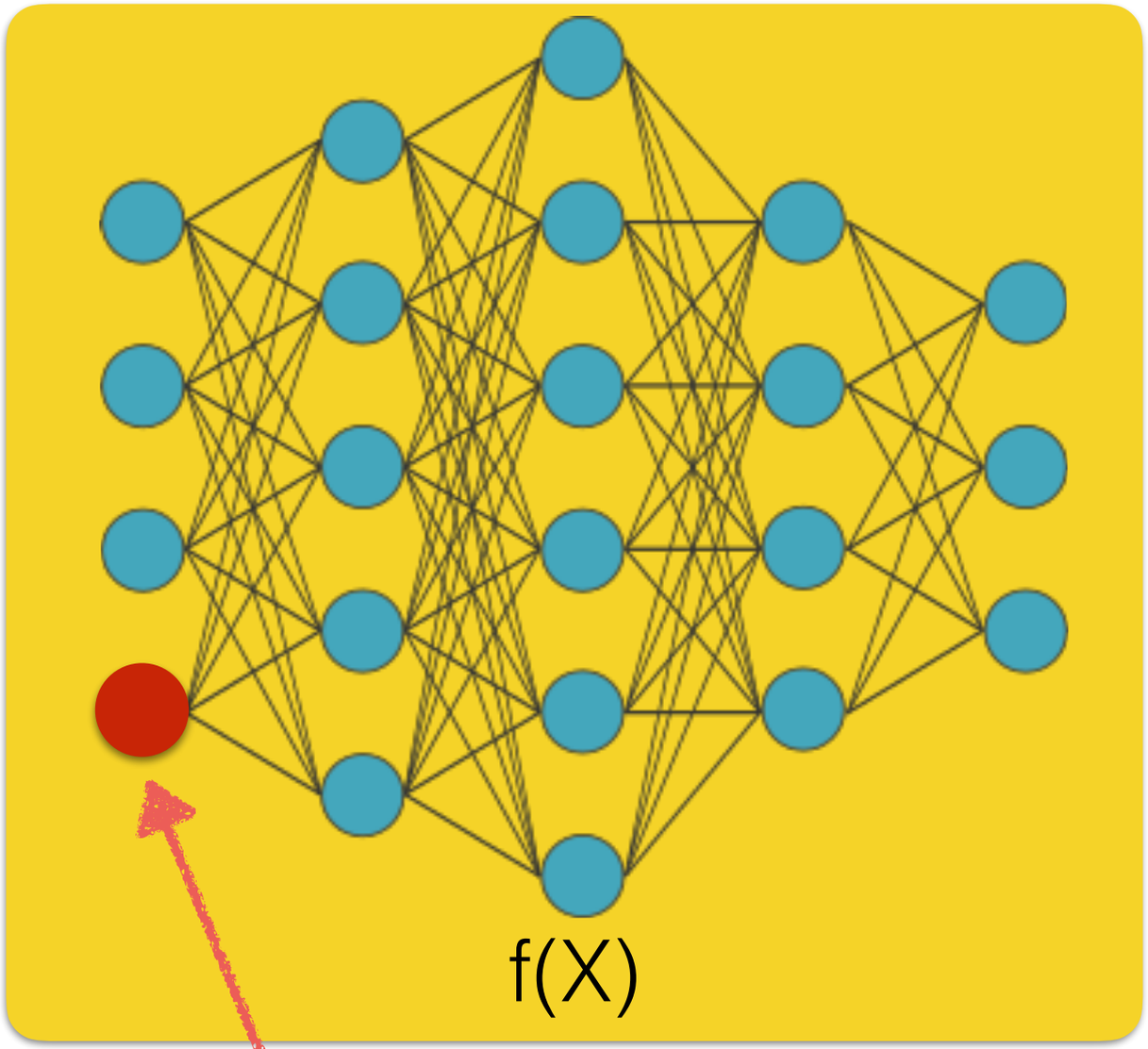


# Neuron



# Neural Networks

X




Y

Each unit represents “neuron”

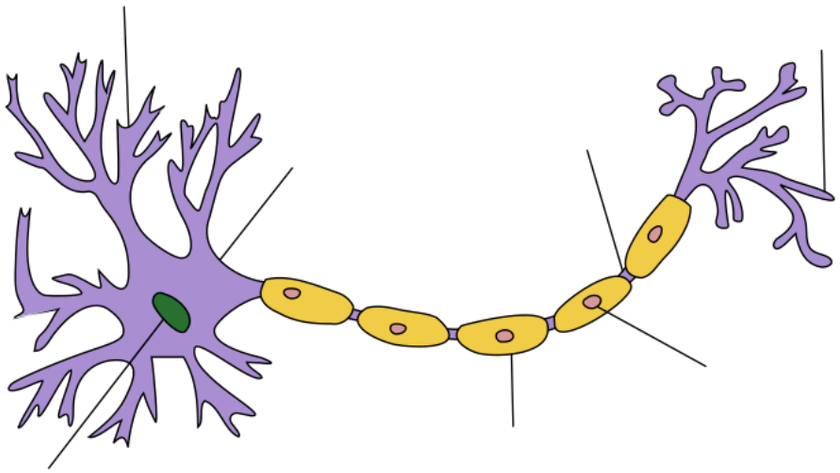
# What is Neural Networks?

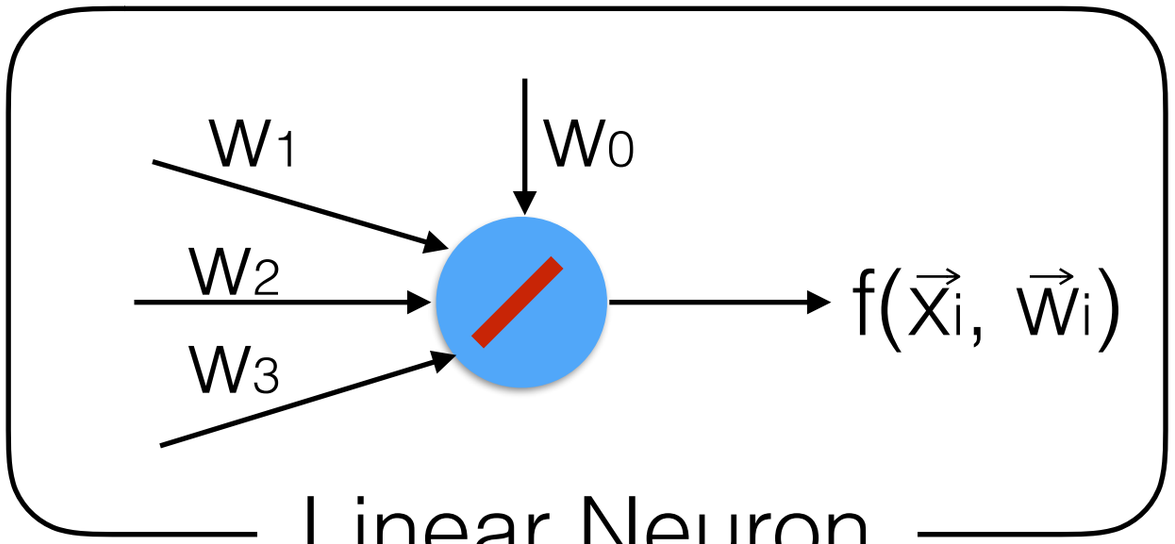
In simple terms,

a set of **neurons (atomic functions)**

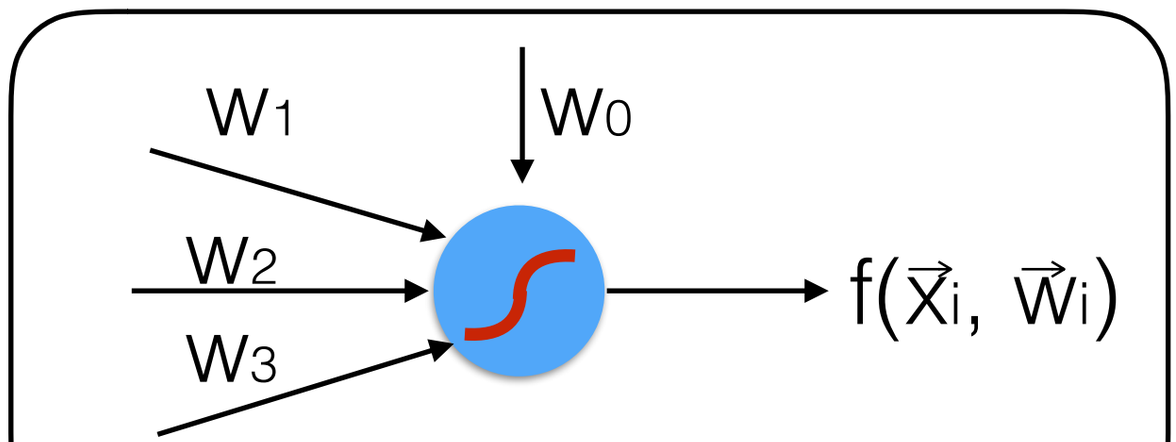
connected in a **non-linear** way

# “Neuron”






Linear Neuron

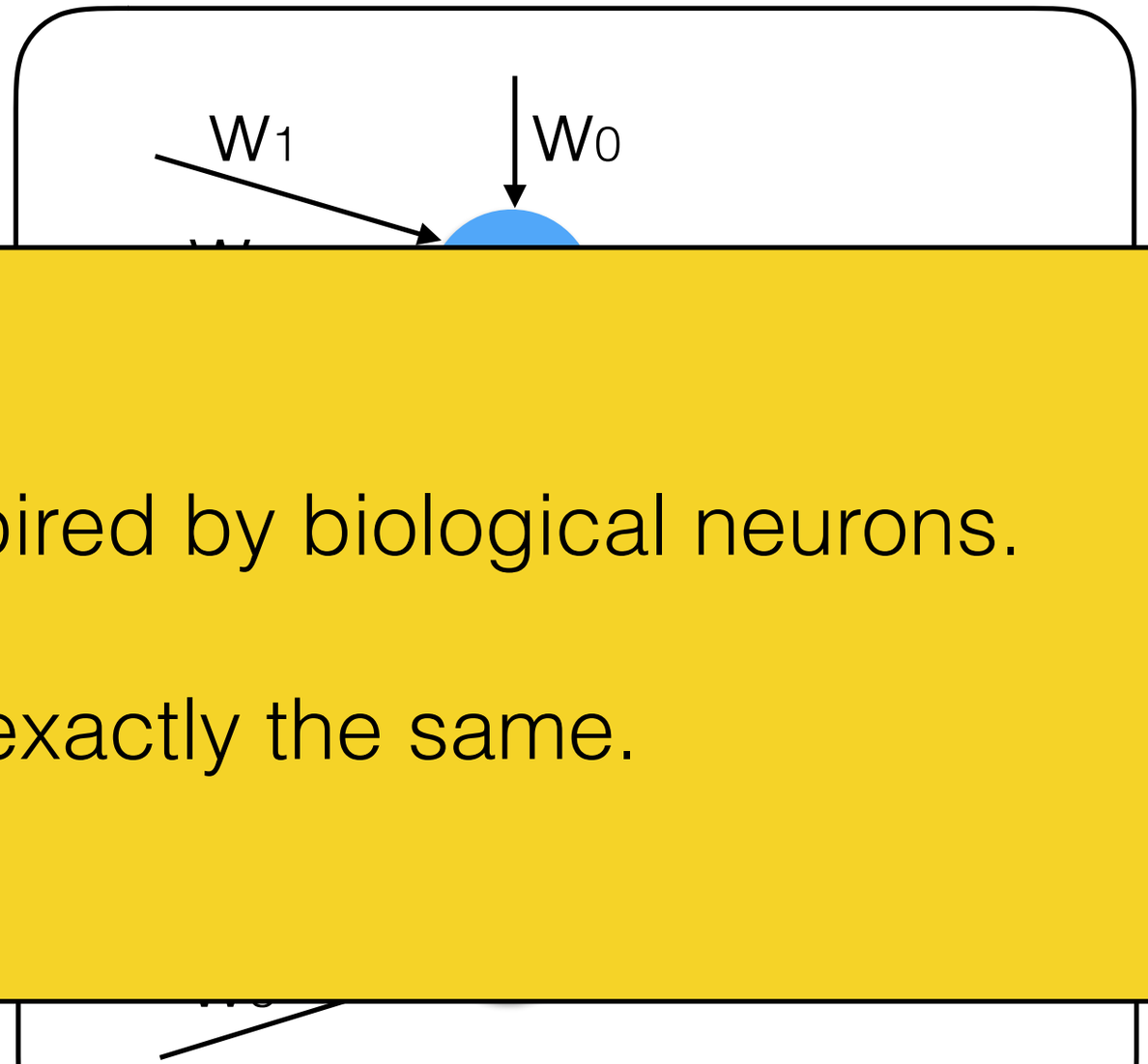



Logistic Neuron

More neurons!

Modified from the HKUST slide

# “Neuron”



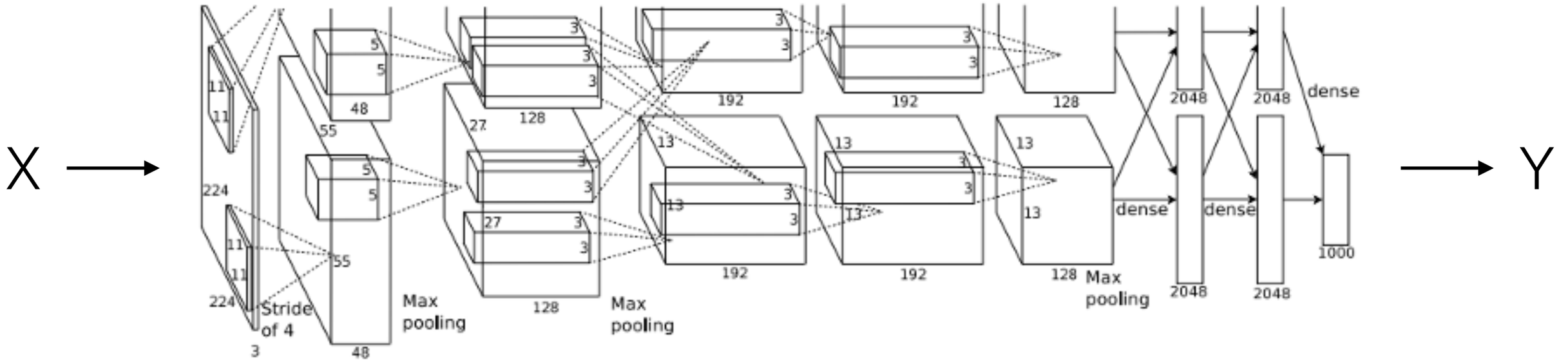

Artificial Neurons are inspired by biological neurons.

They are **NOT** exactly the same.

More neurons!

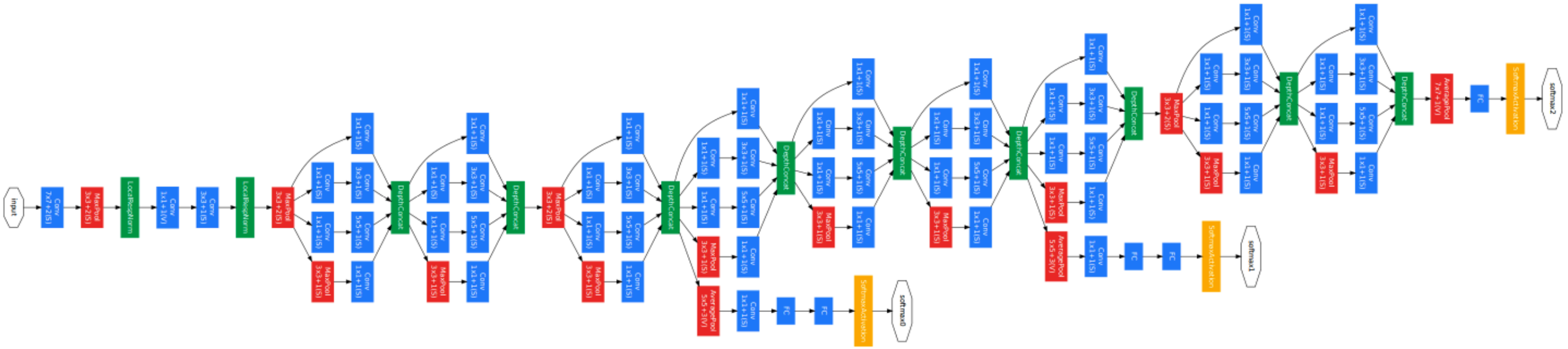
Modified from the HKUST slide

# (Convolutional) Neural Networks



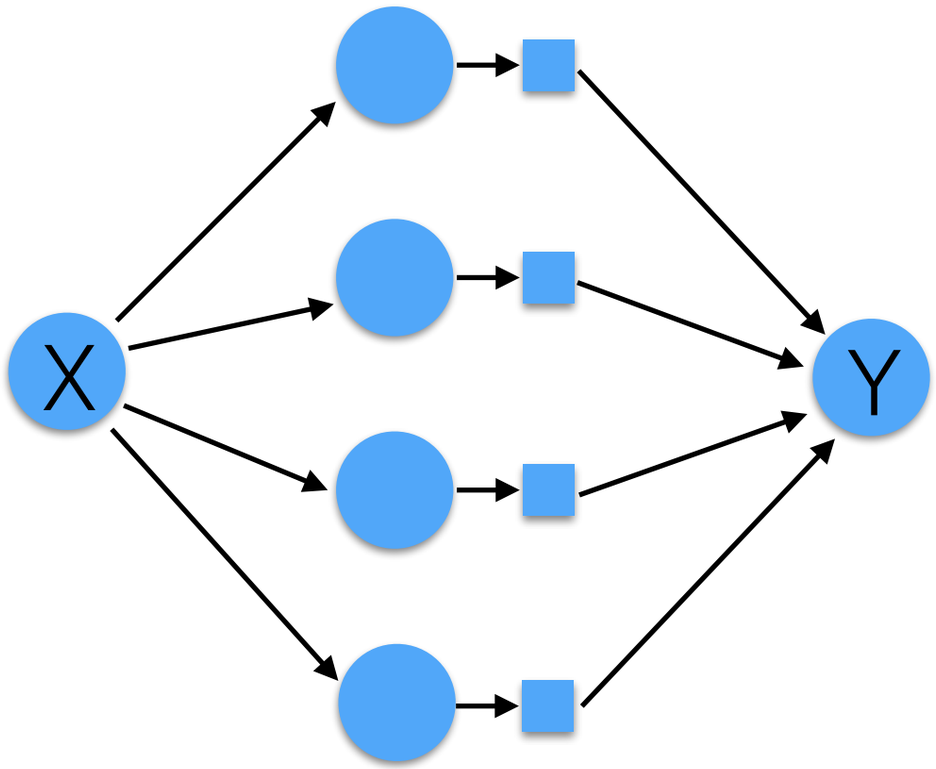

A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

# Deeper Neural Networks

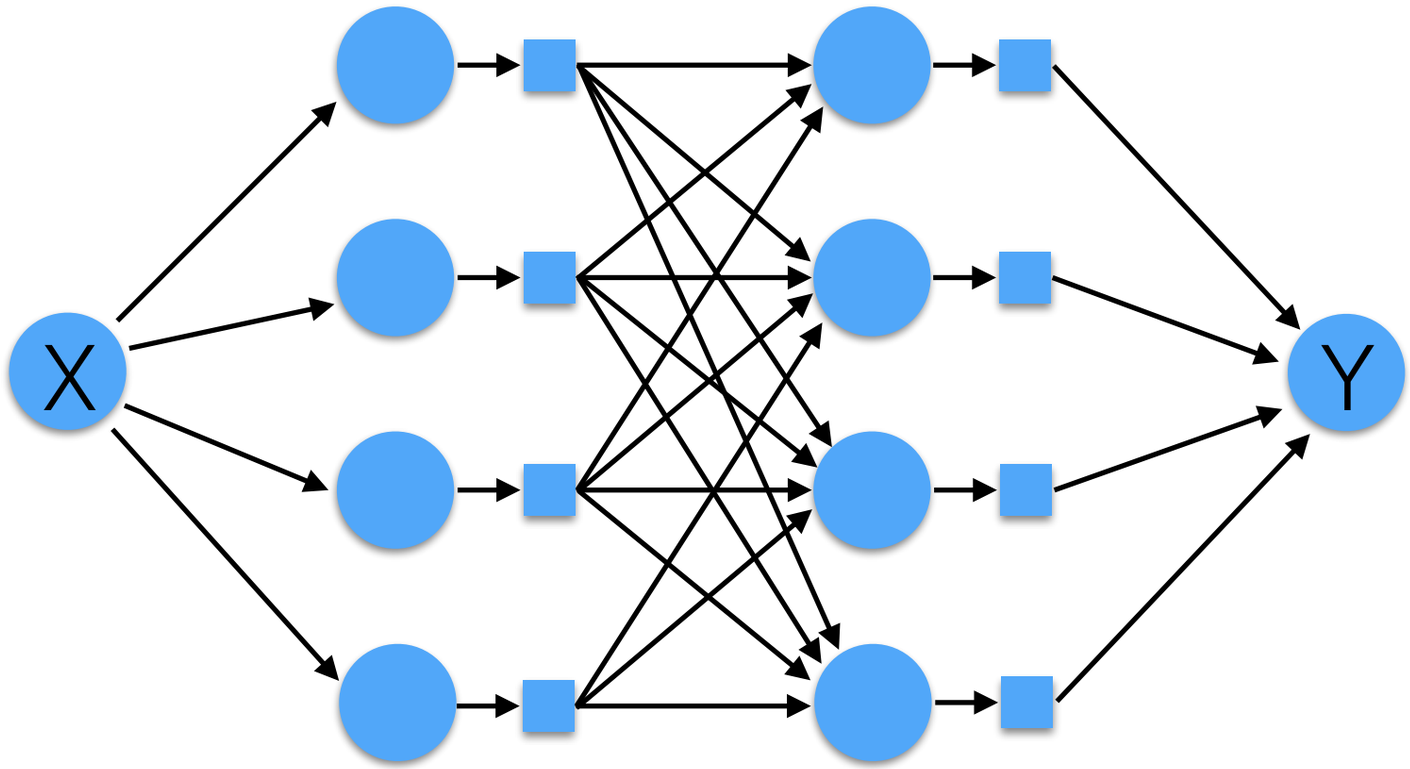



Szegedy et. al., Going deeper with convolutions, CVPR 2015

# Why “Deep Learning”?



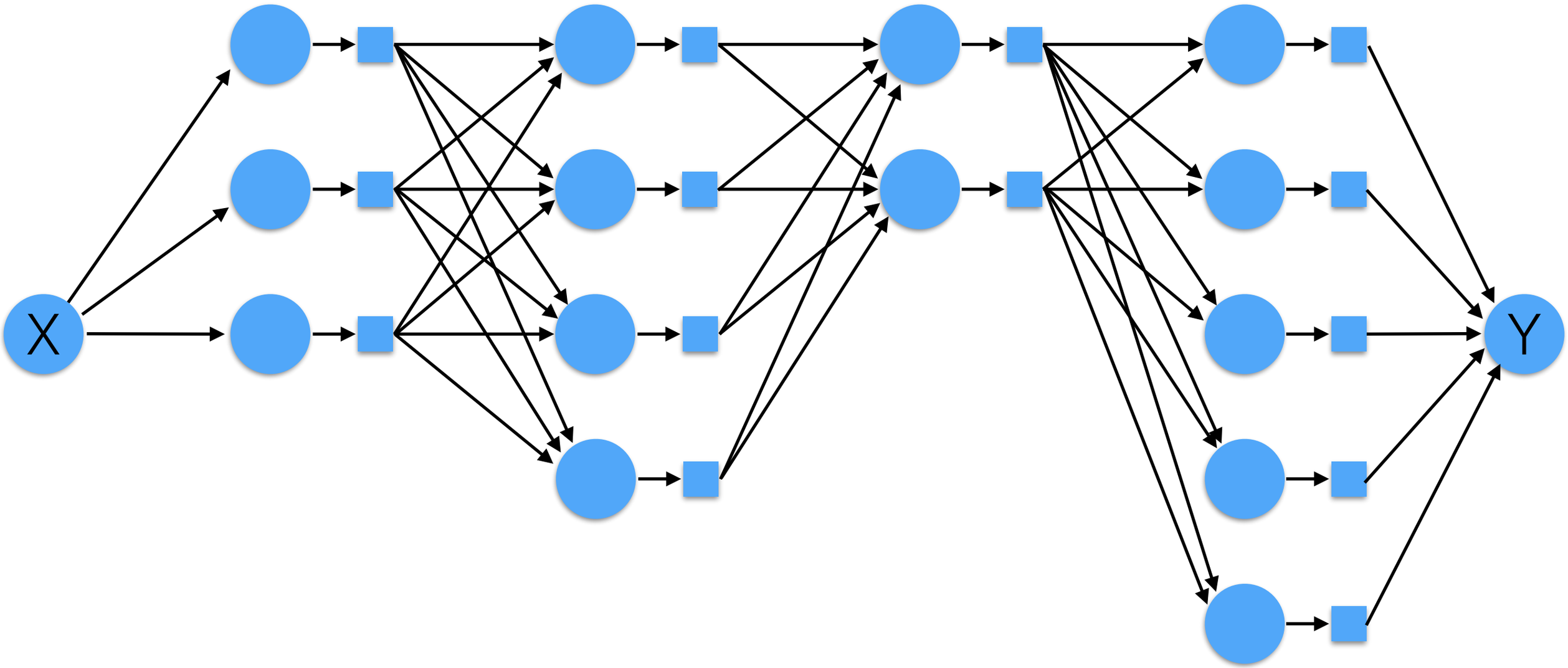

Shallow network



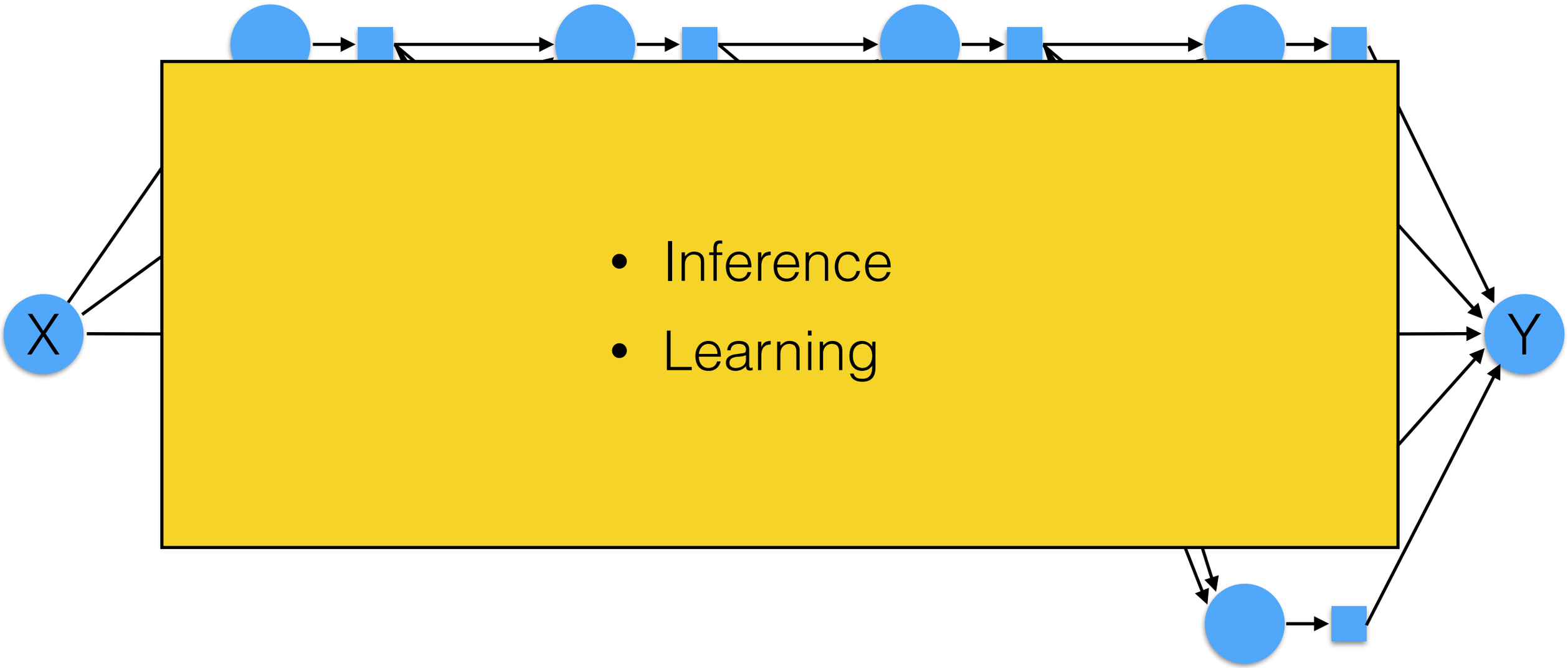

Deep network

The algorithm for training a **deep** network

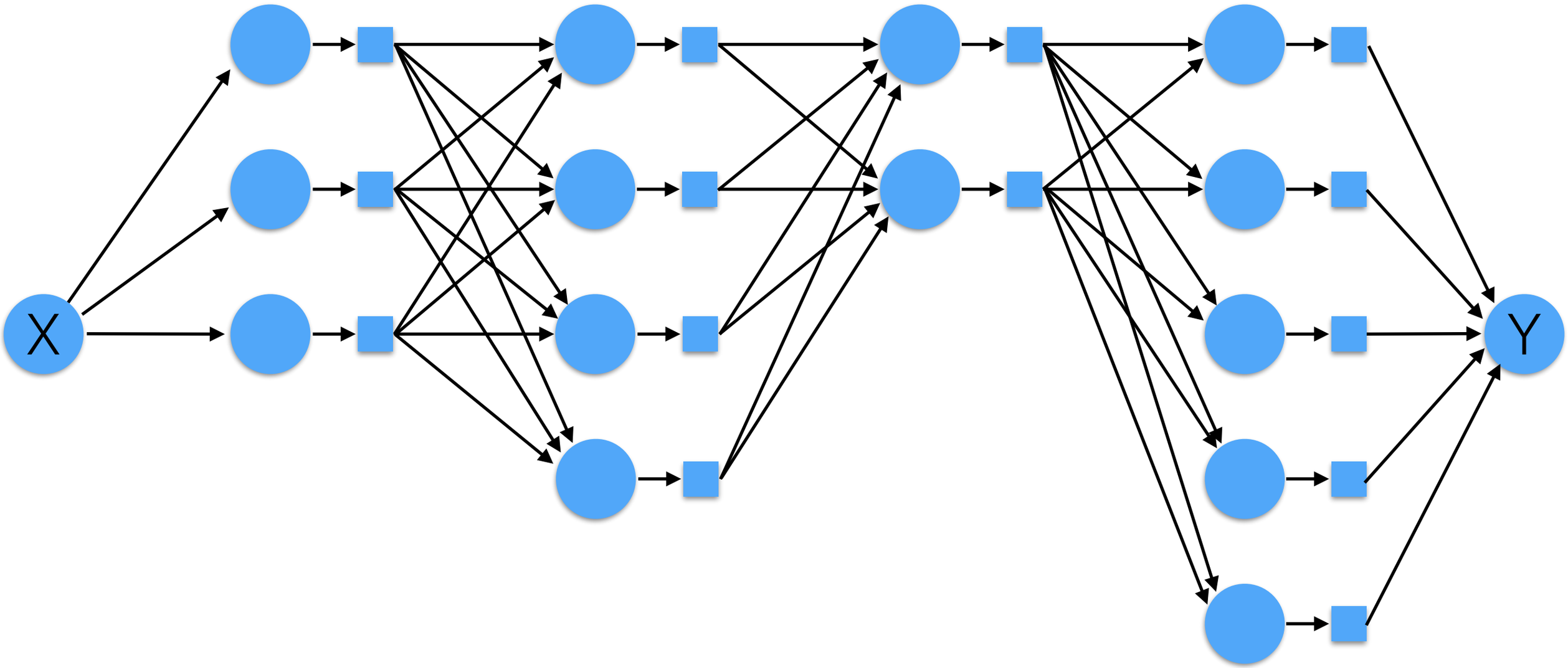
# Neural Networks Example



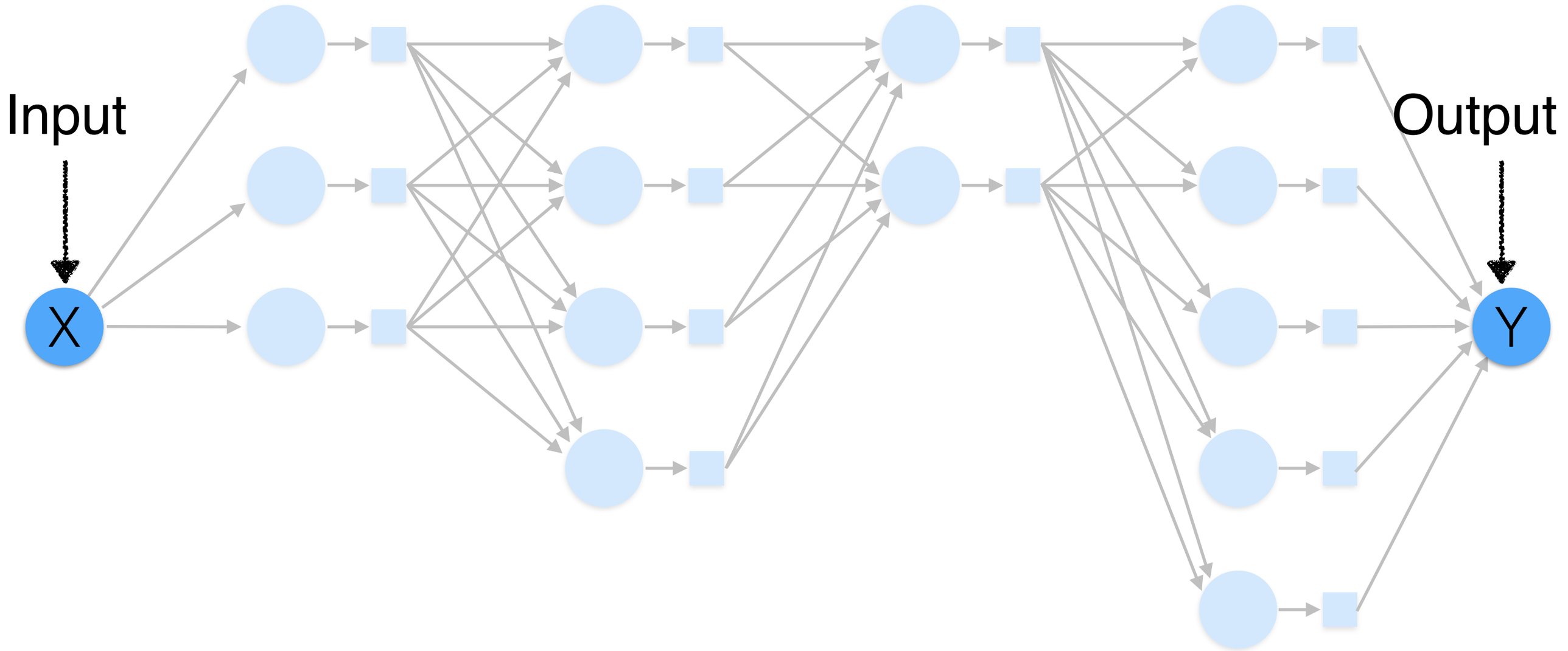
# Neural Networks Example



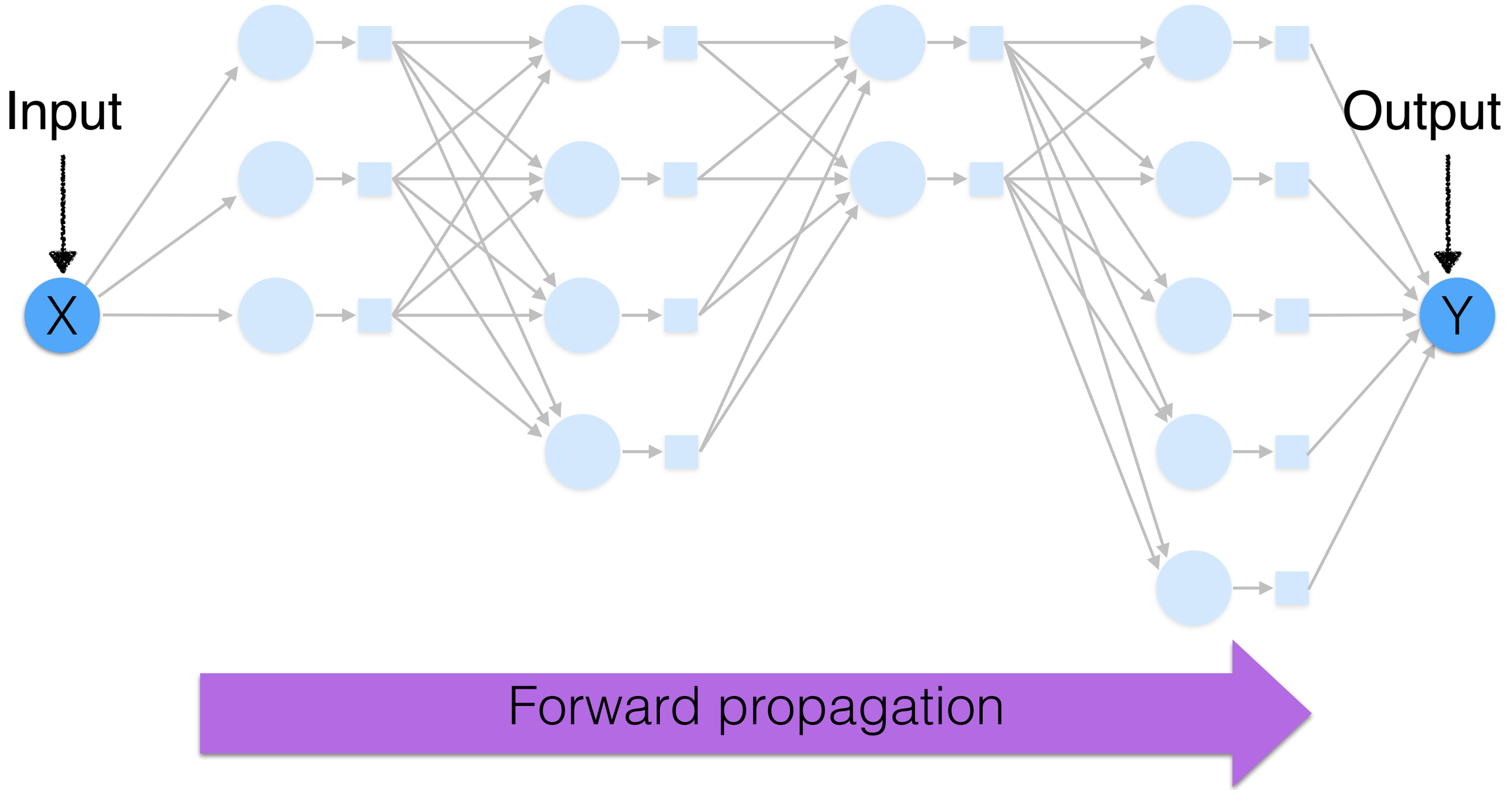
# Forward propagation



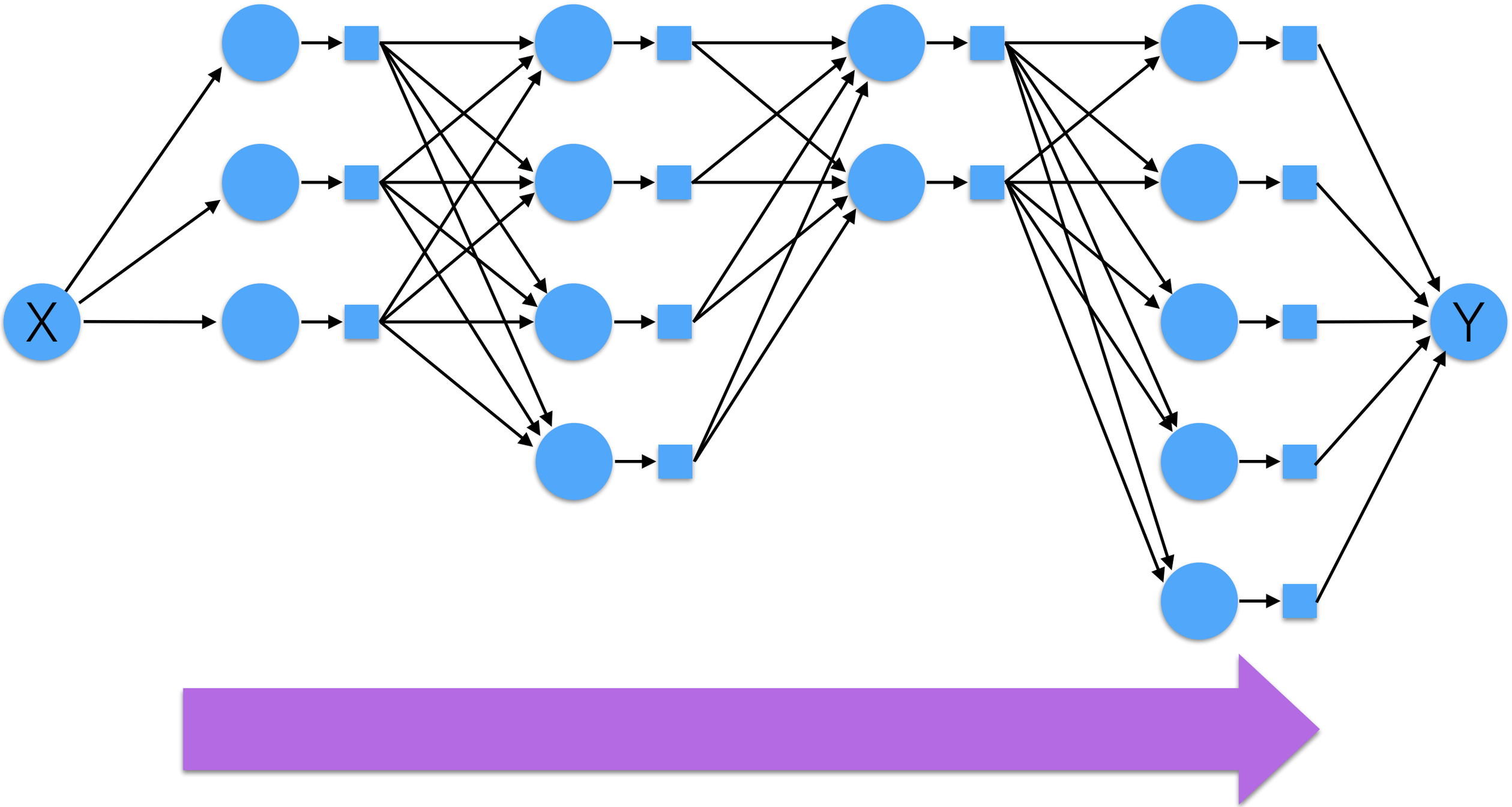
# Forward propagation



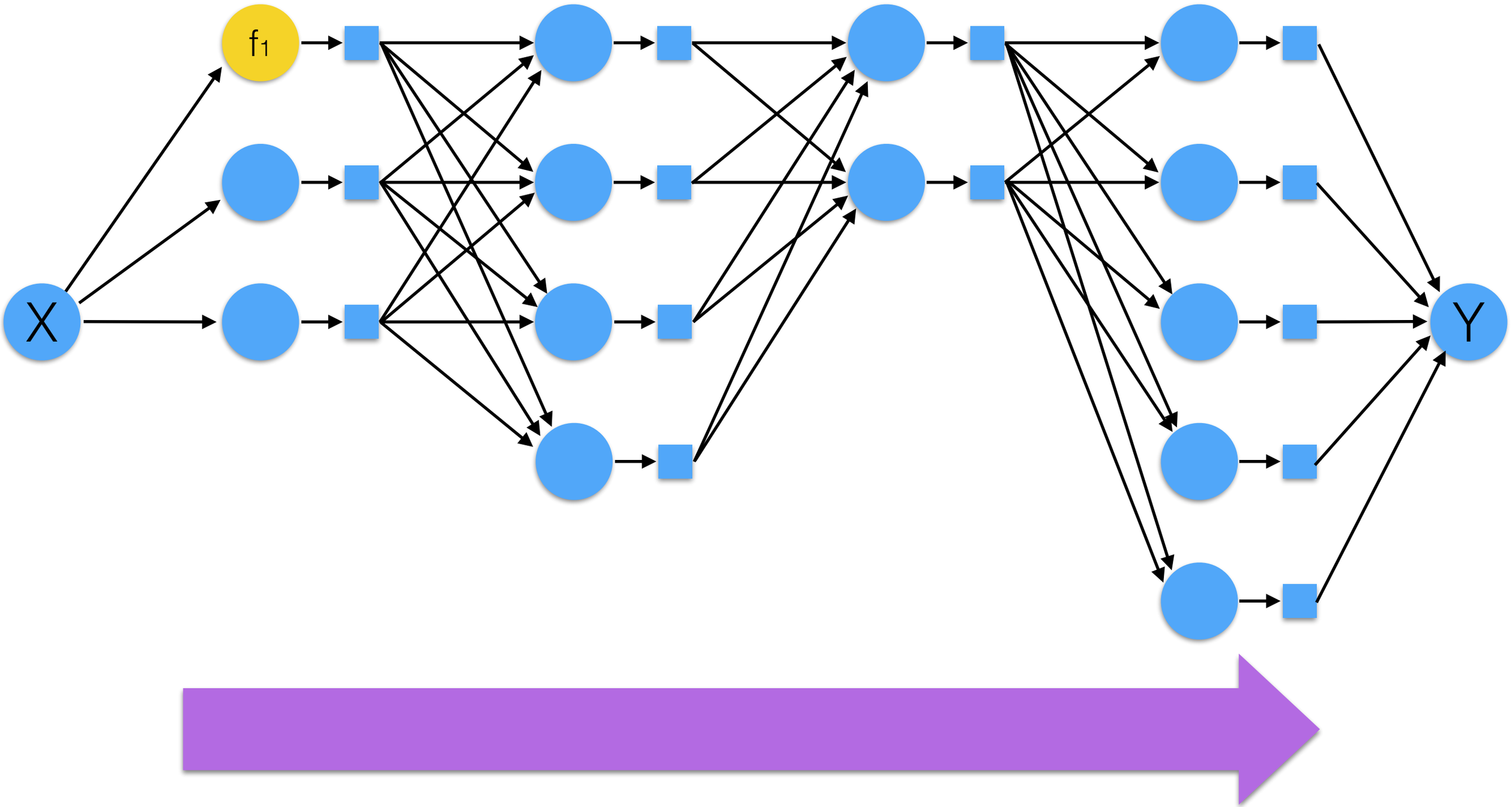
# Forward propagation



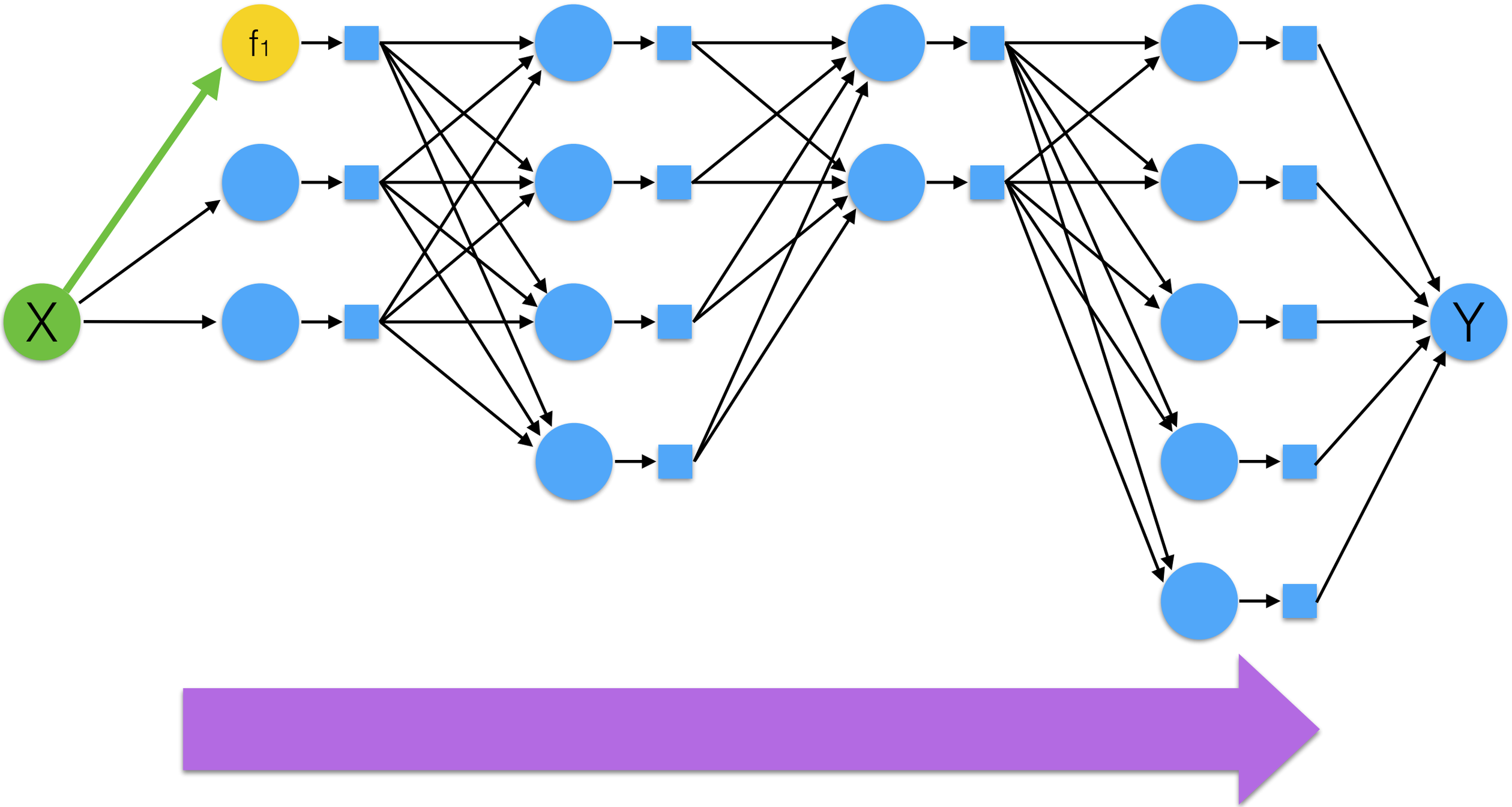
# Forward propagation



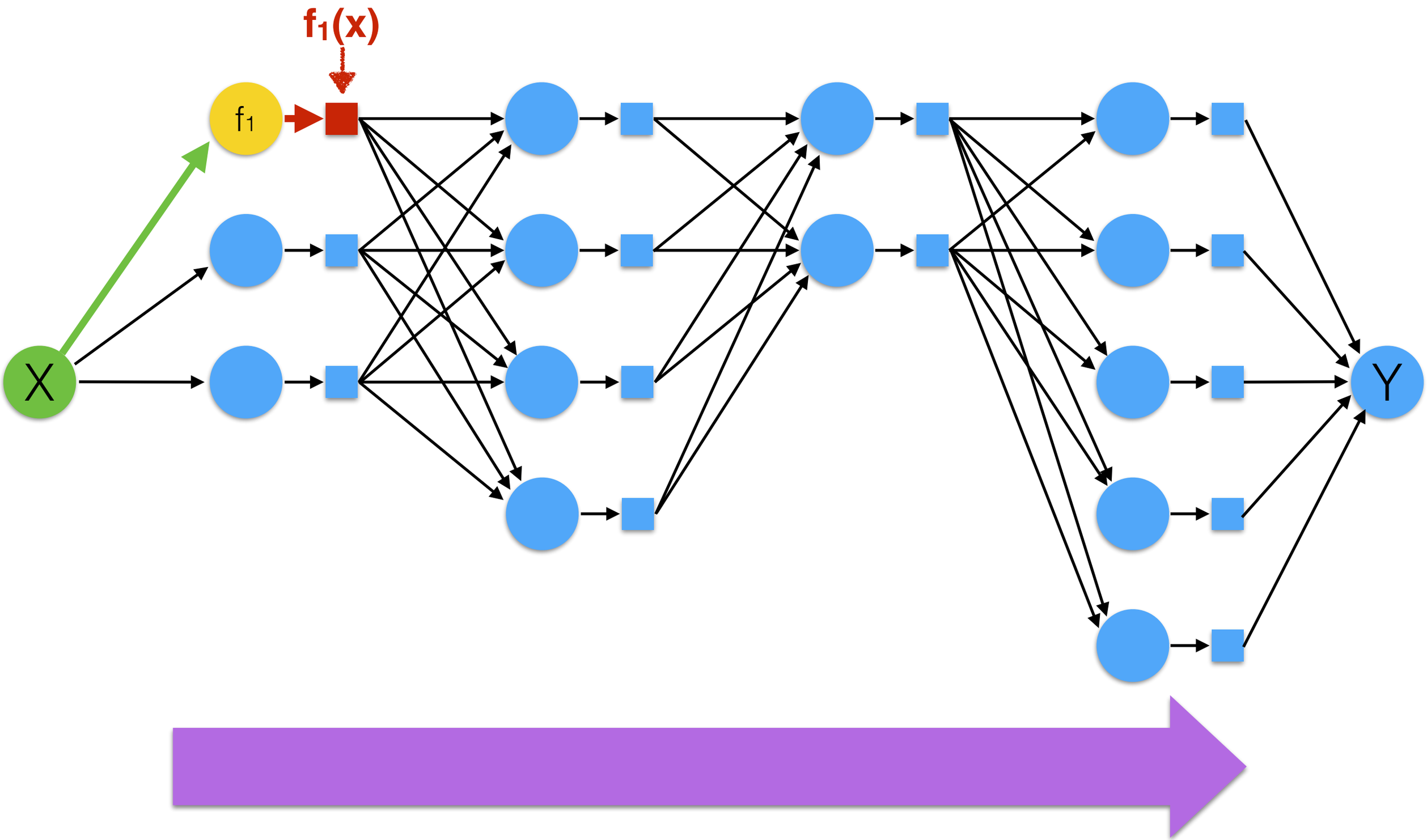
# Forward propagation



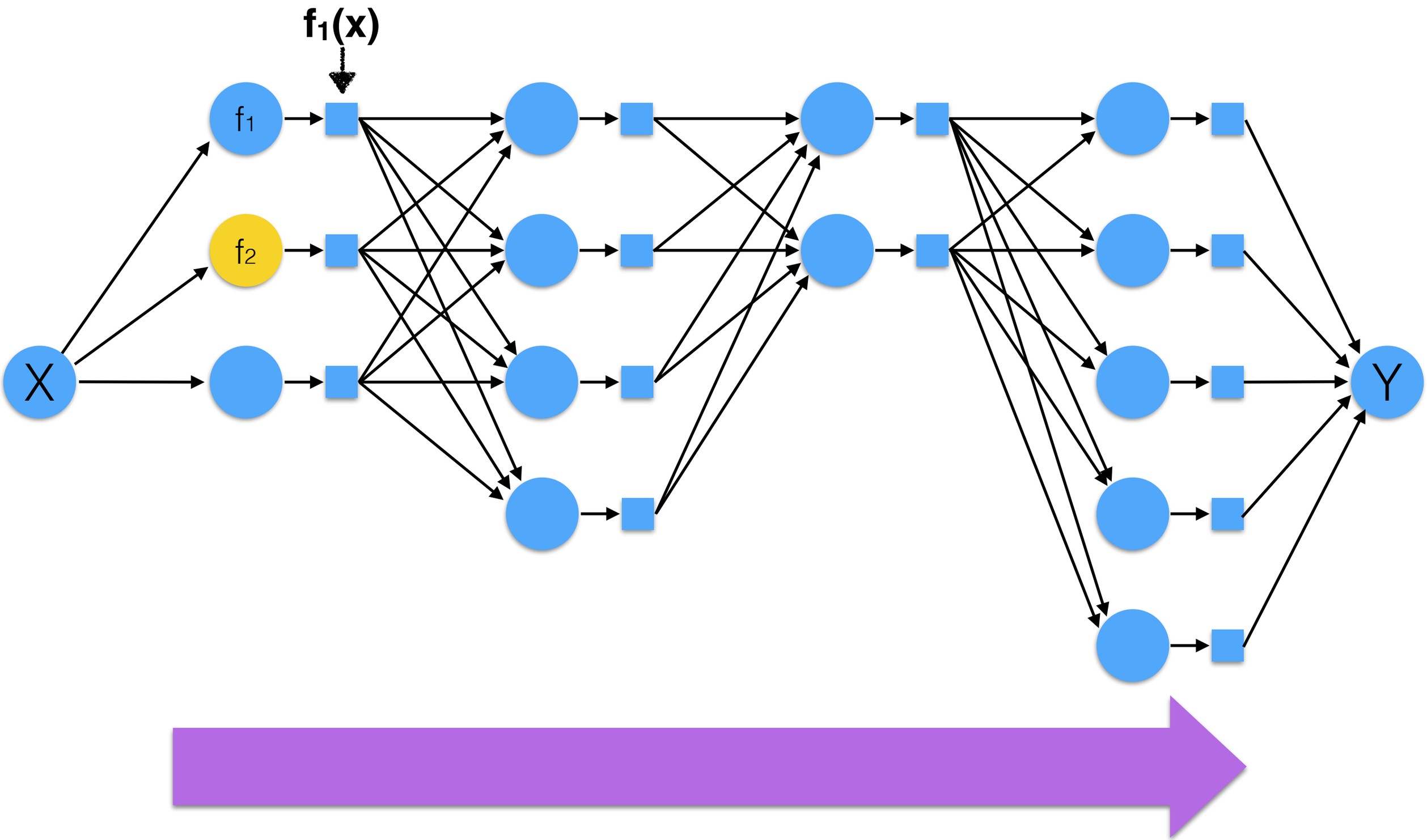
# Forward propagation



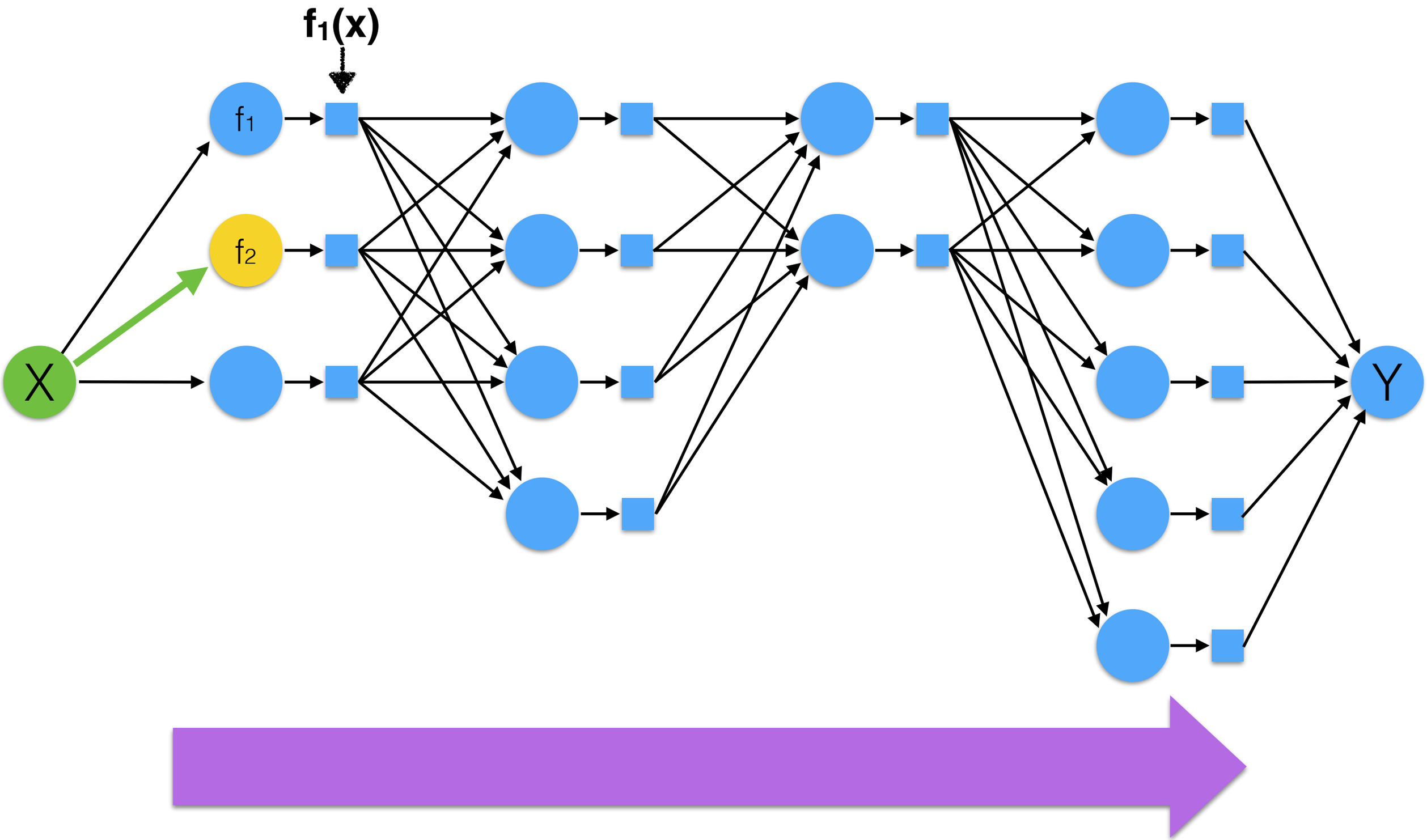
# Forward propagation



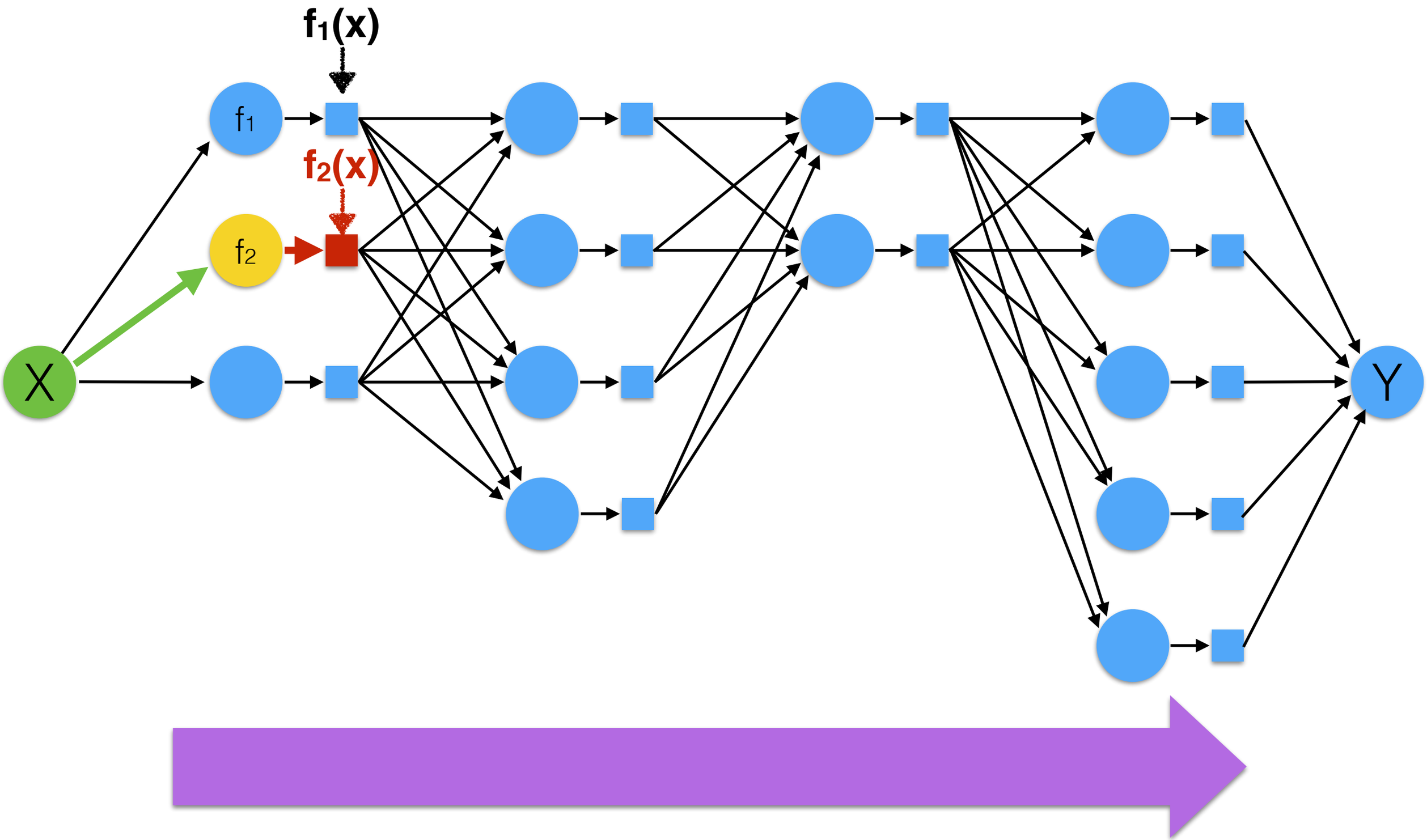
# Forward propagation



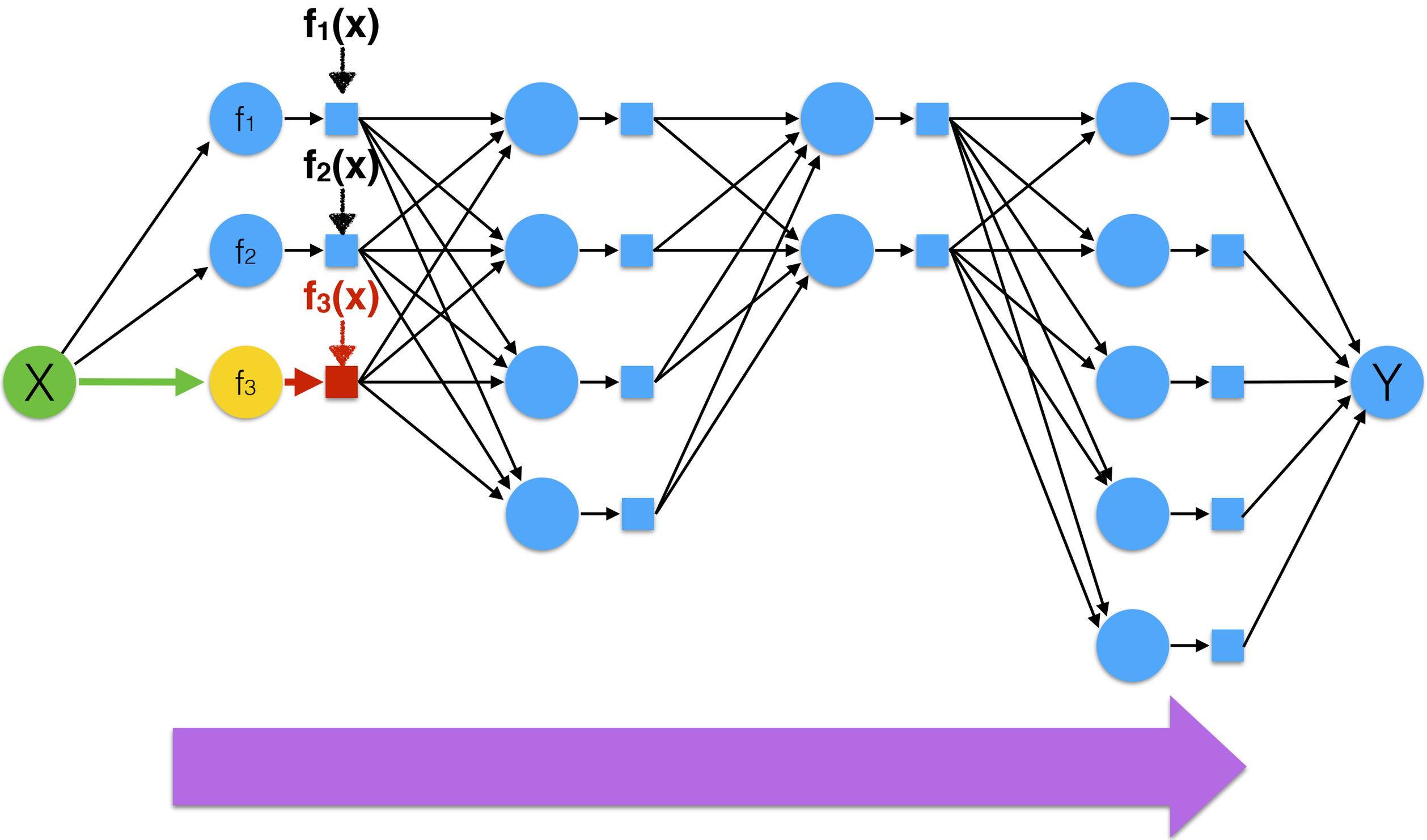
# Forward propagation



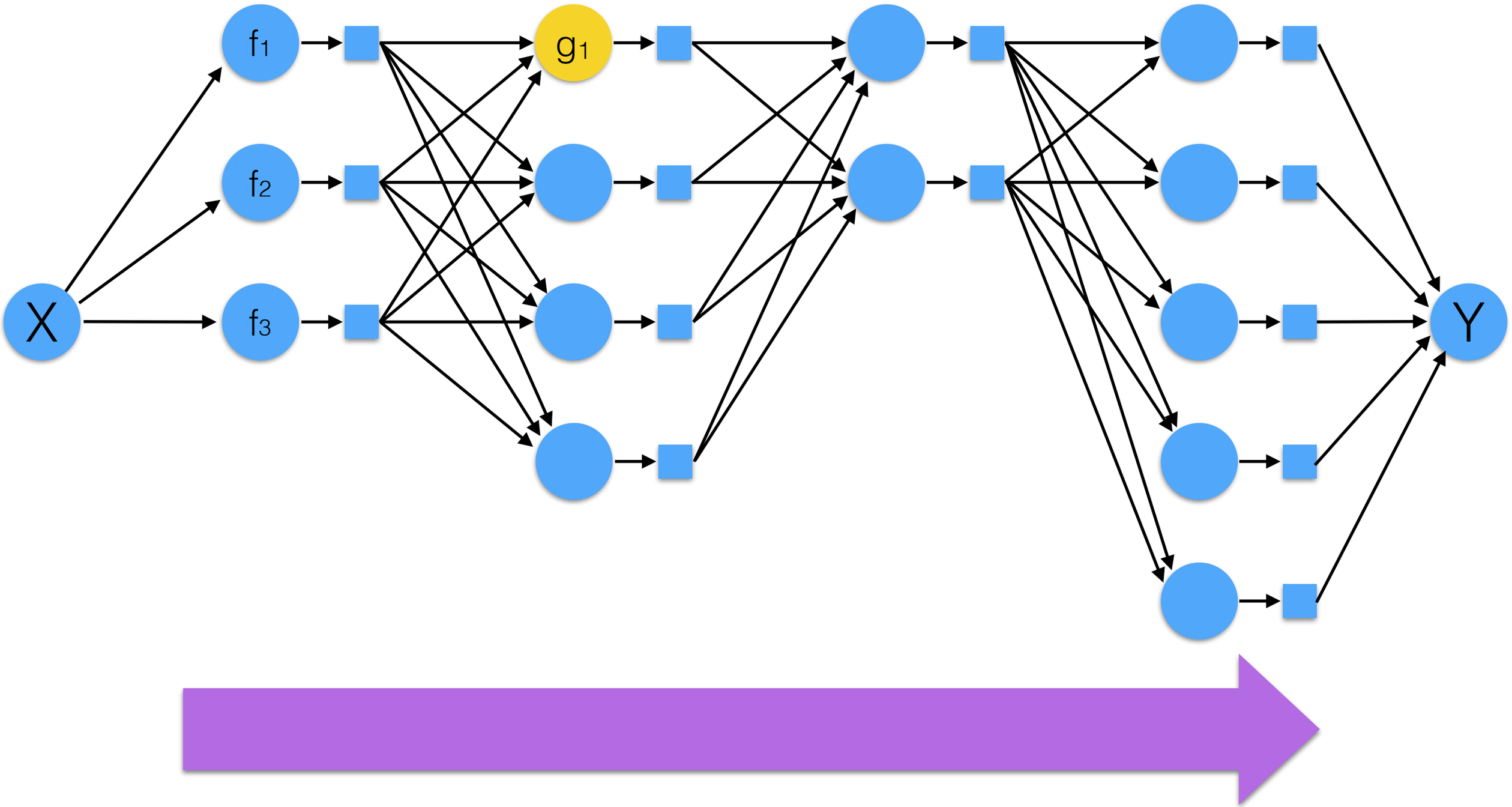
# Forward propagation



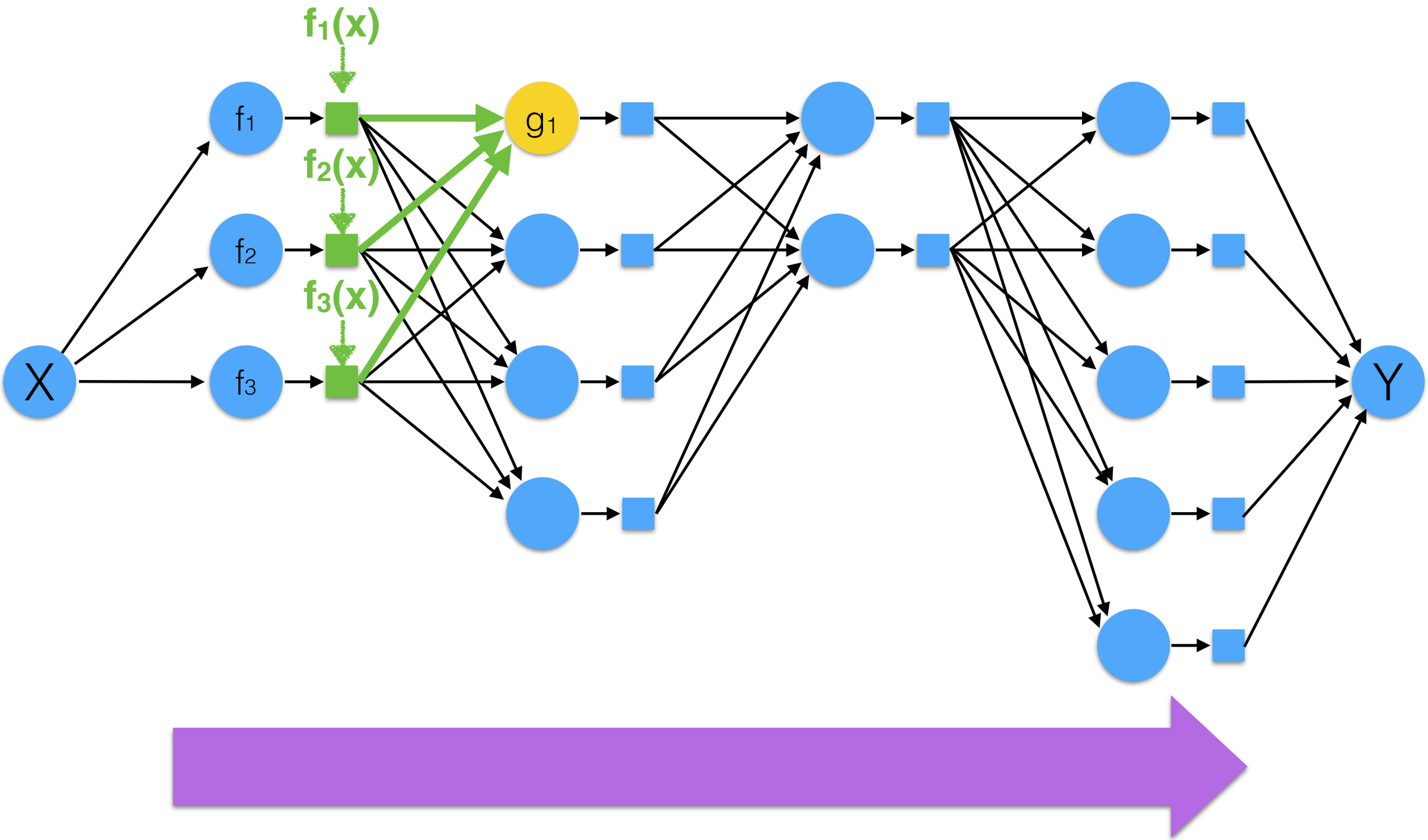
# Forward propagation



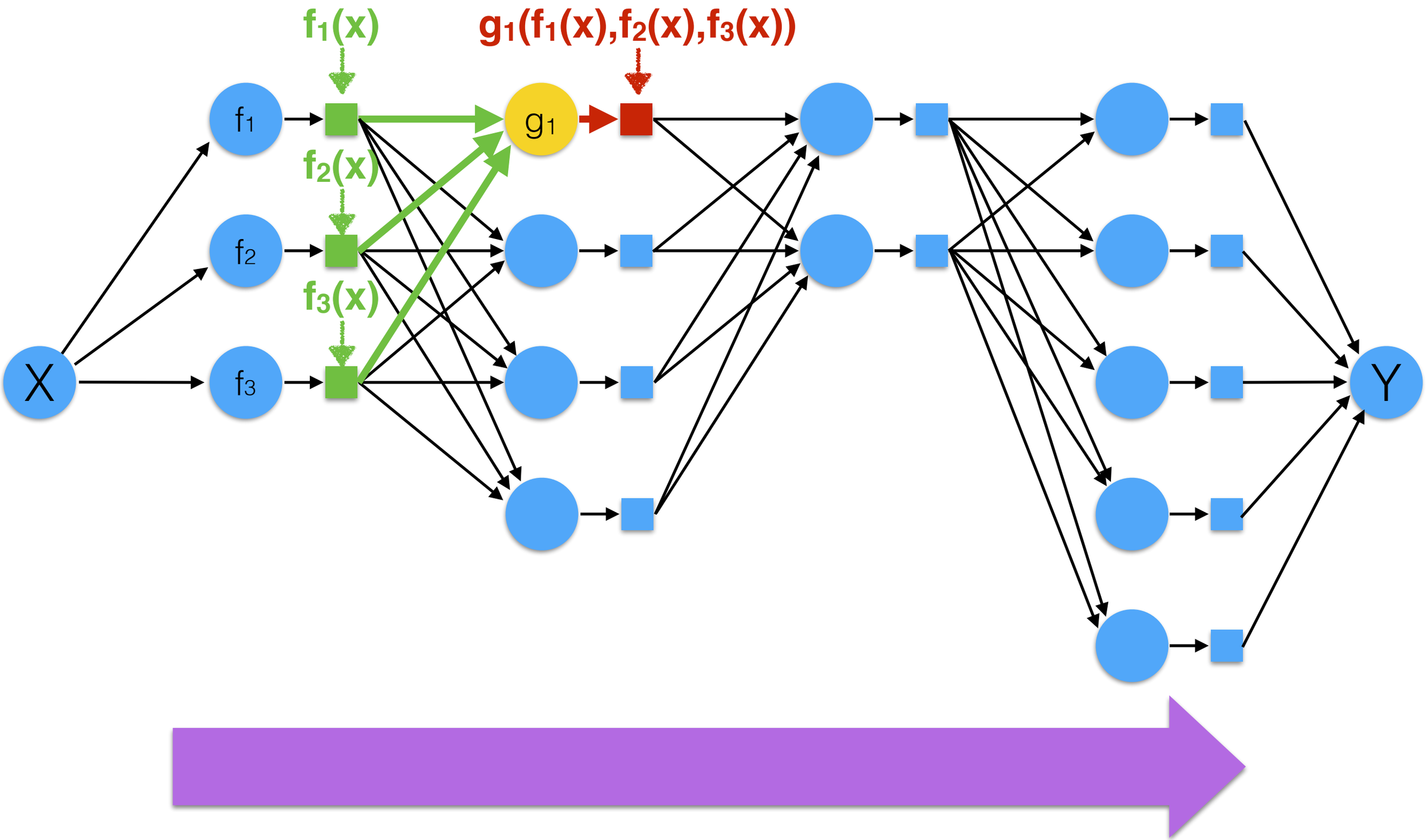
# Forward propagation



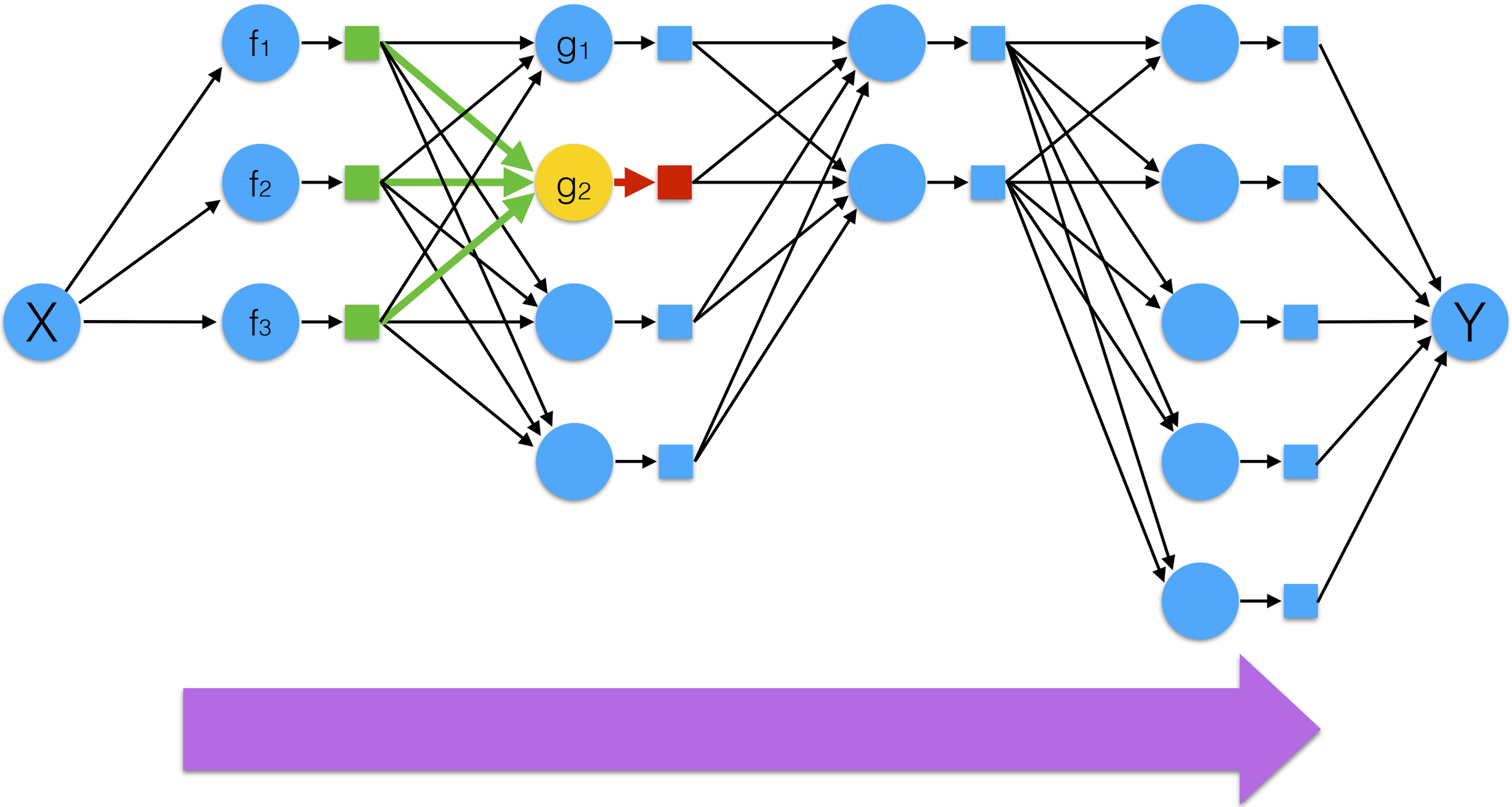
# Forward propagation



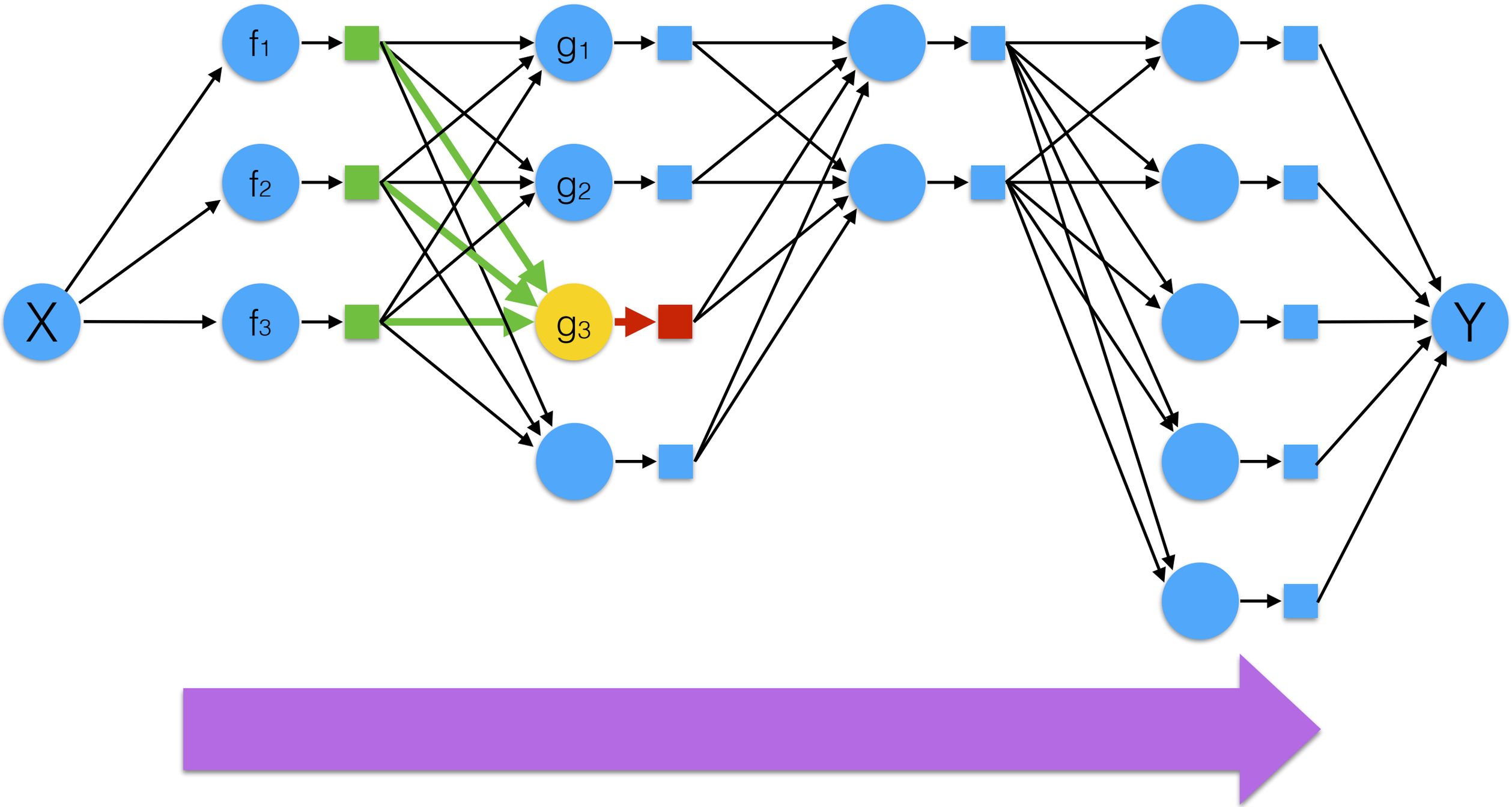
# Forward propagation



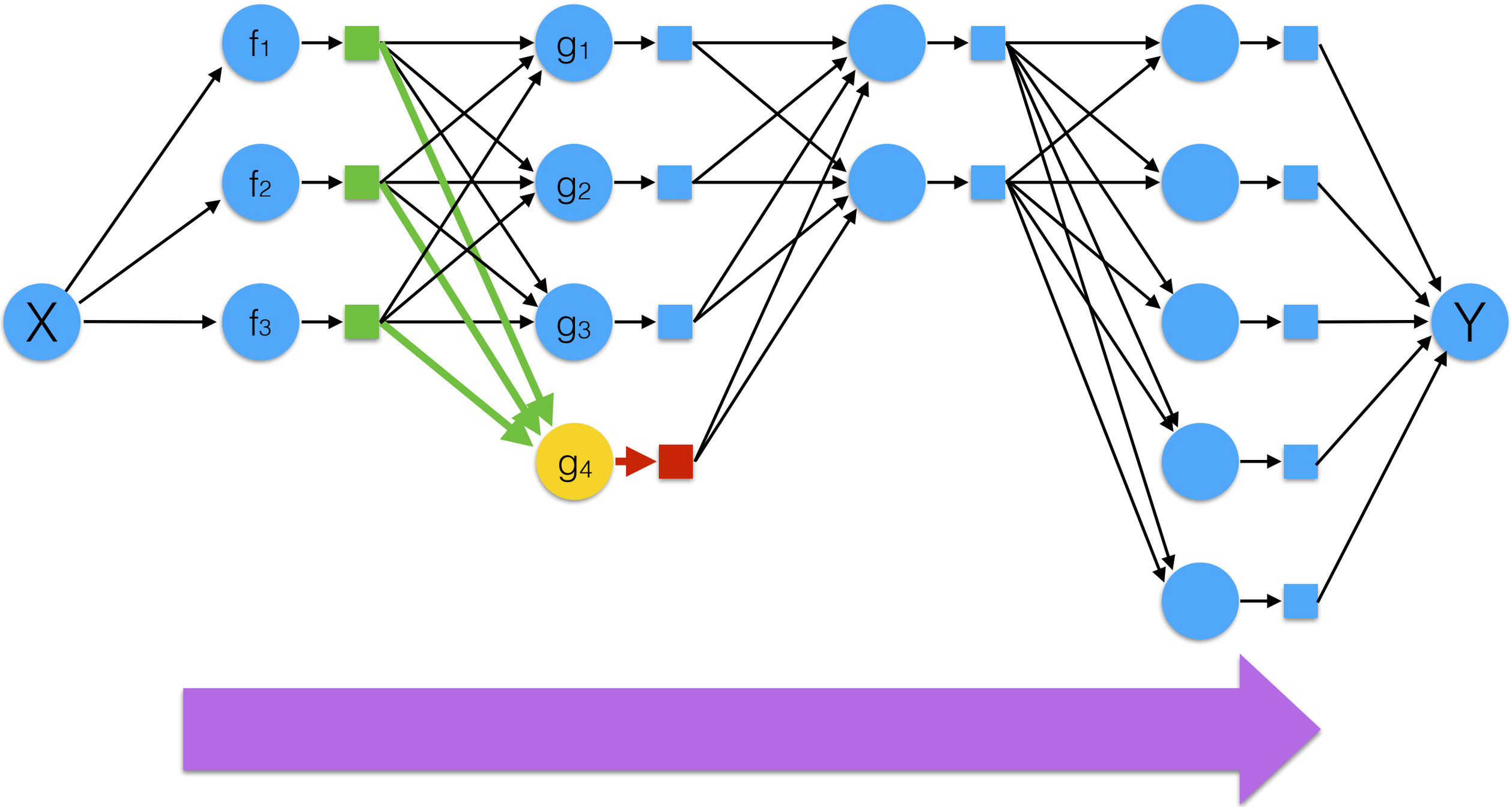
# Forward propagation



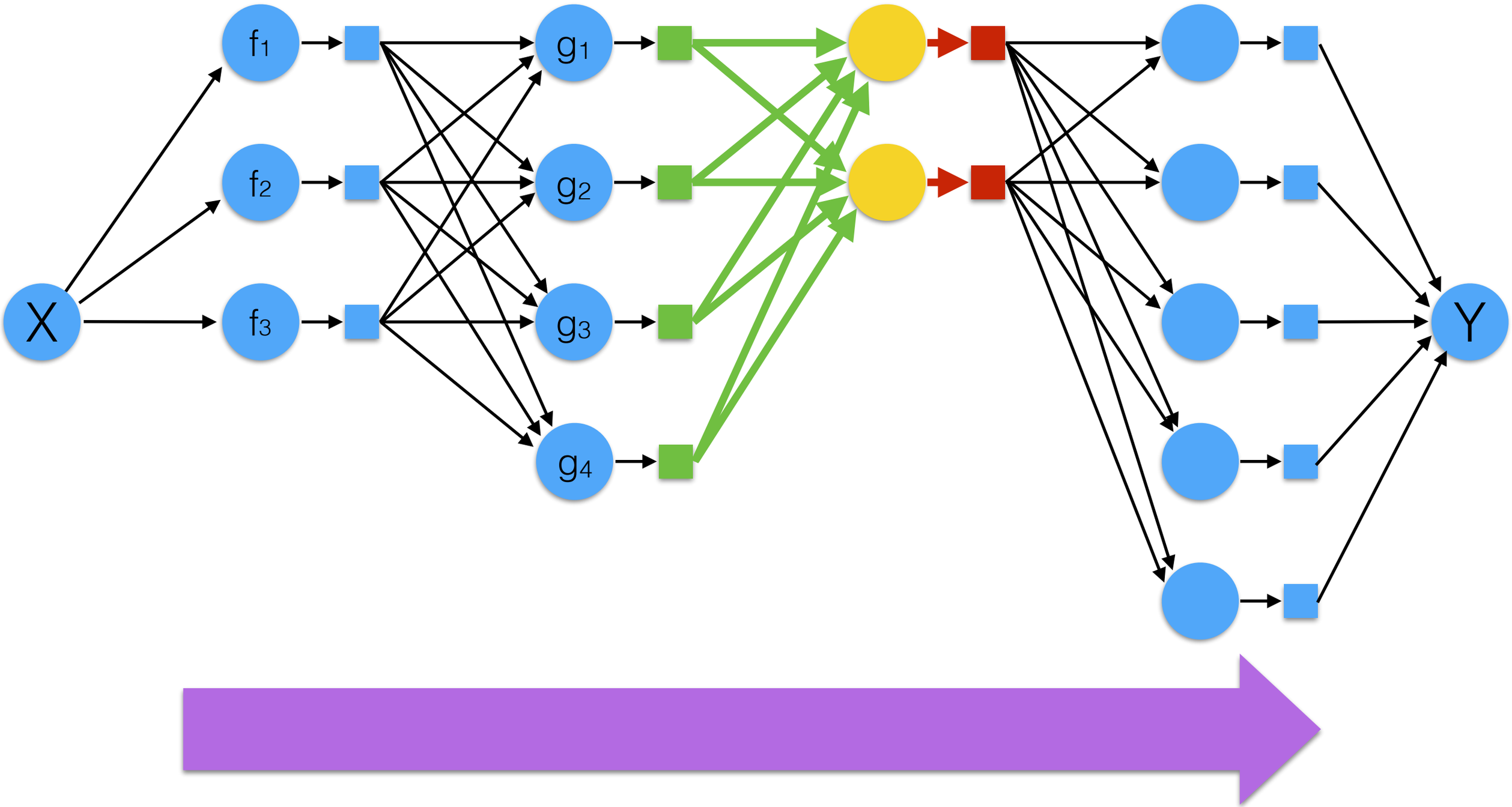
# Forward propagation



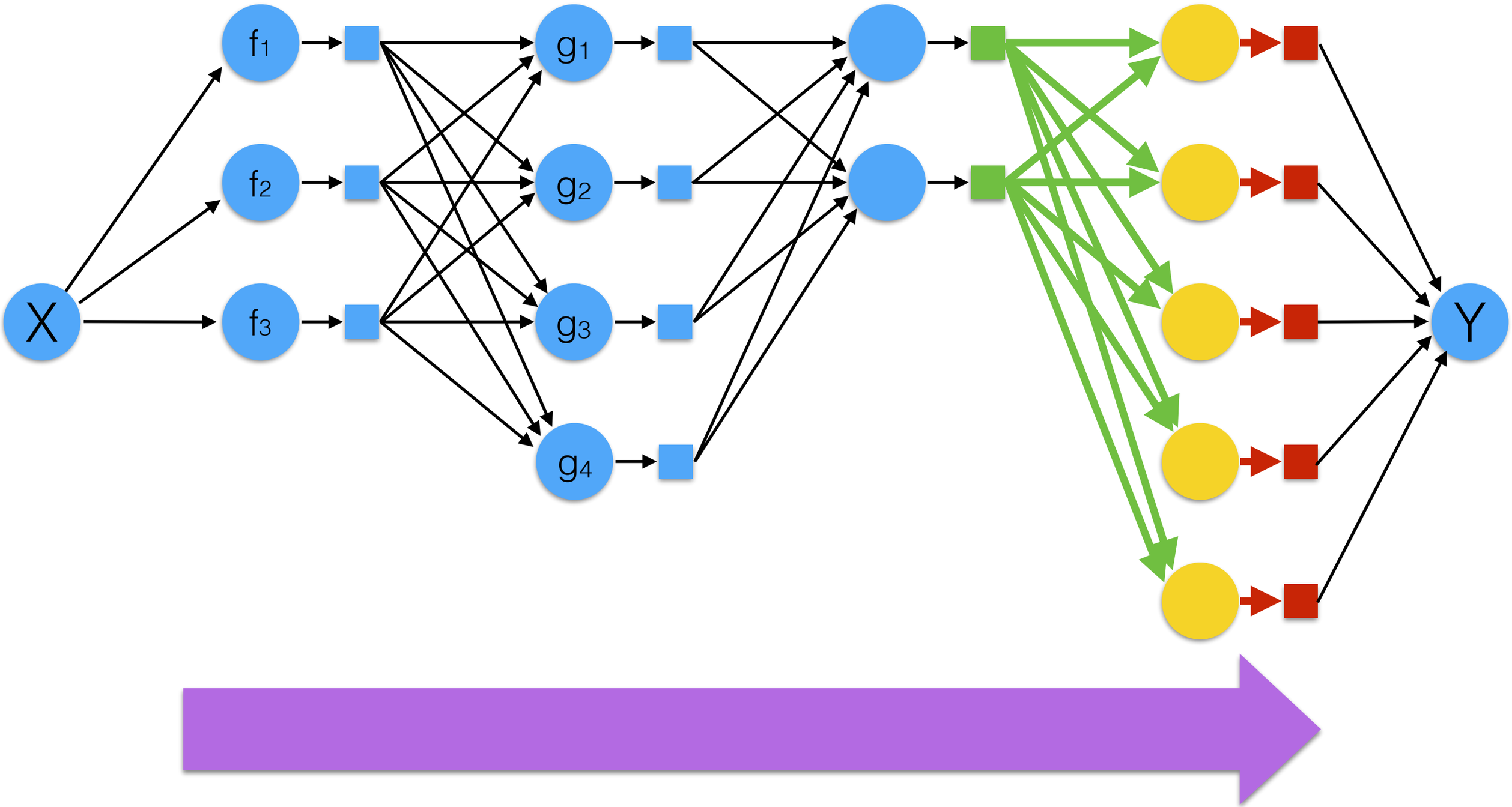
# Forward propagation



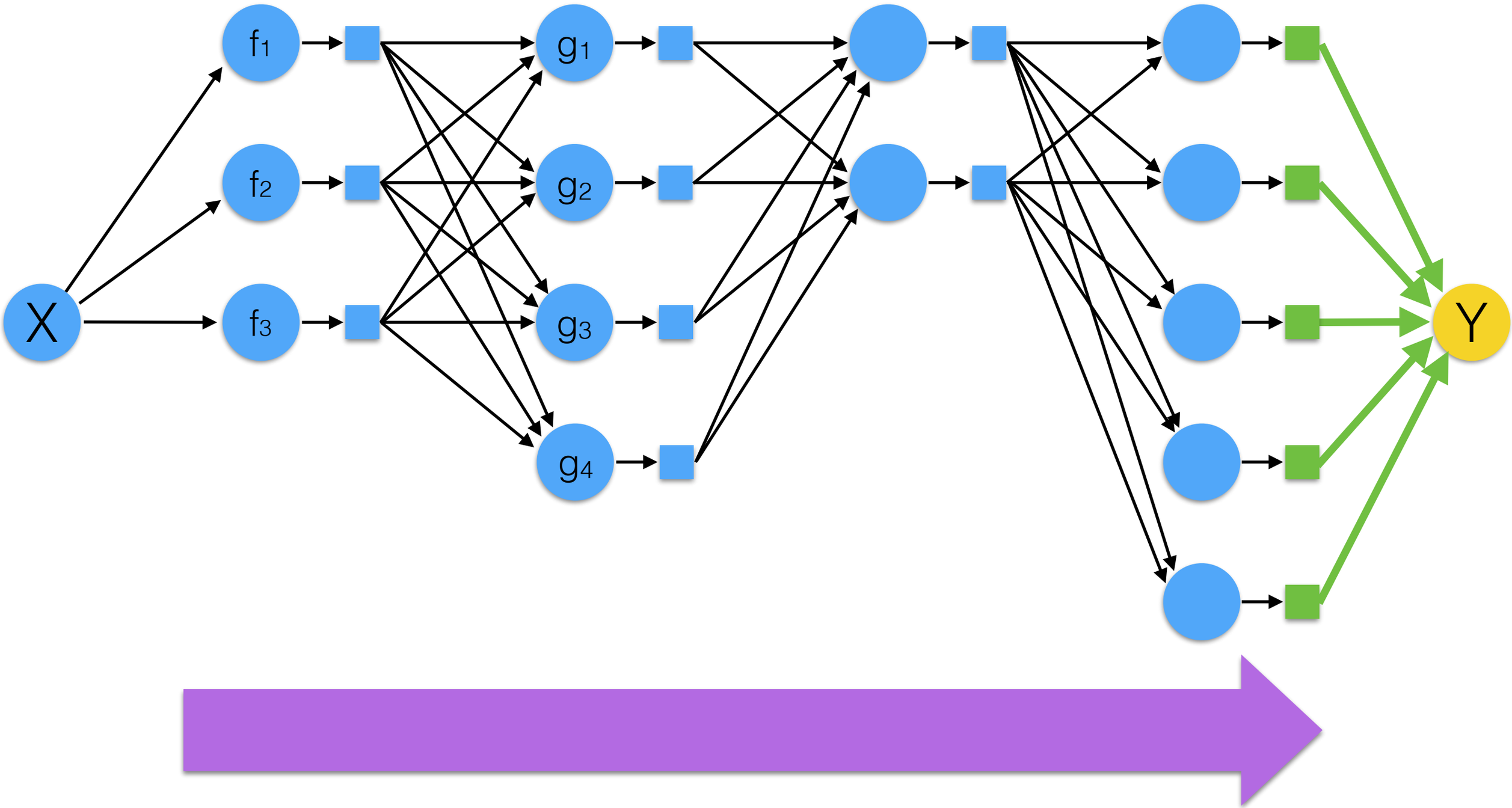
# Forward propagation



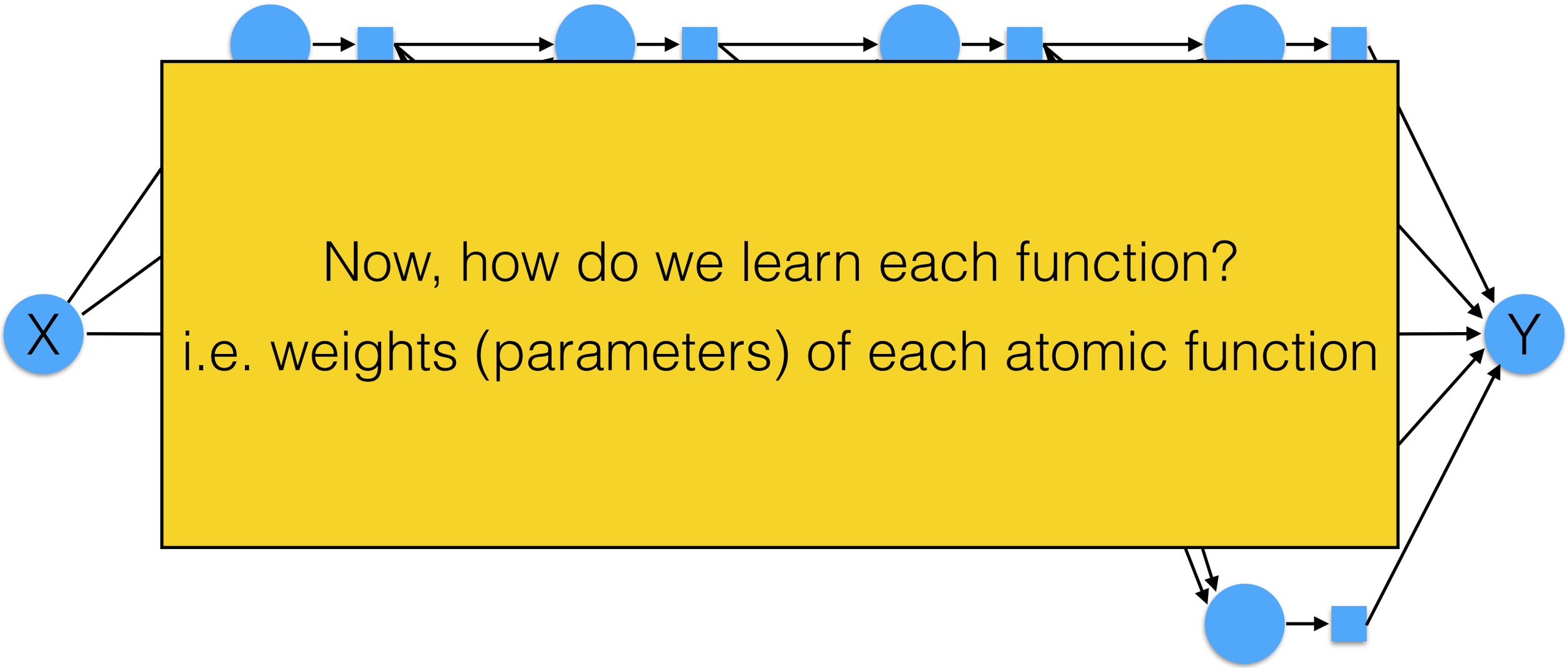
# Forward propagation



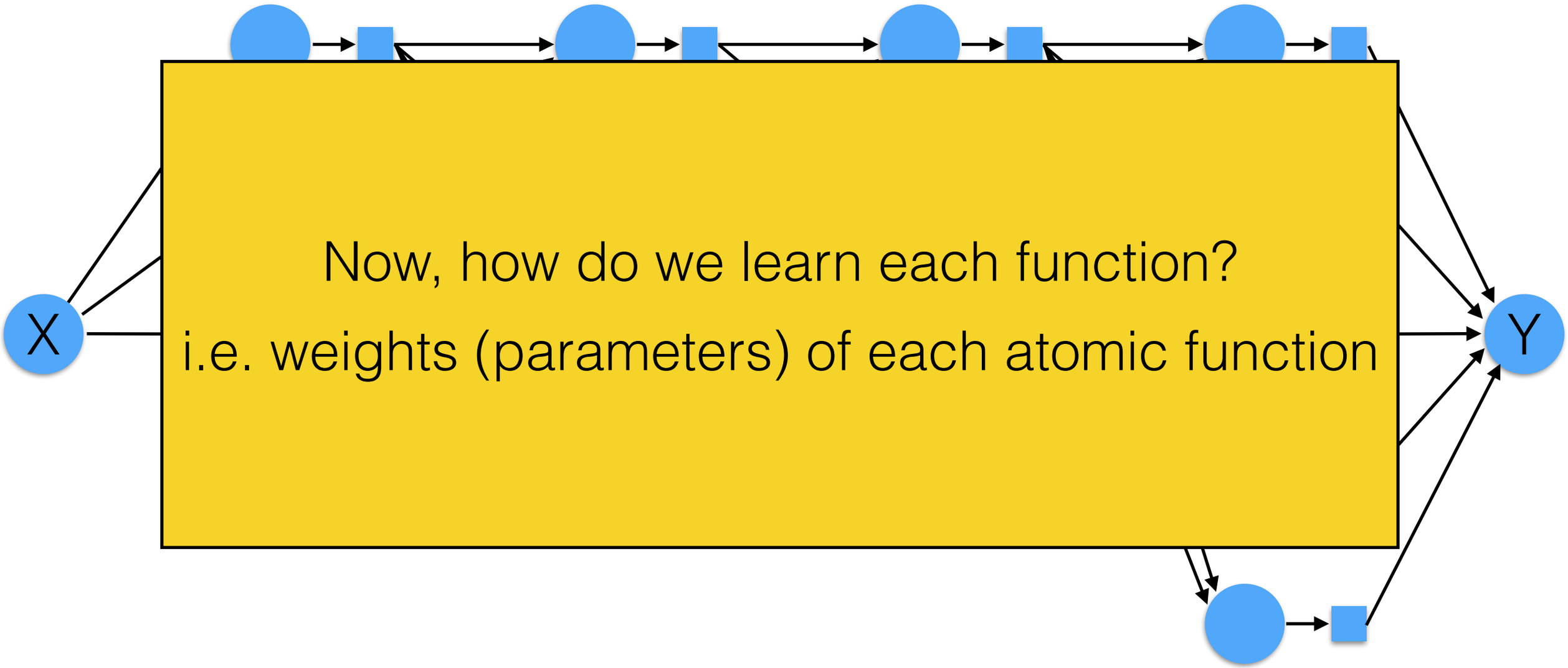
# Forward propagation



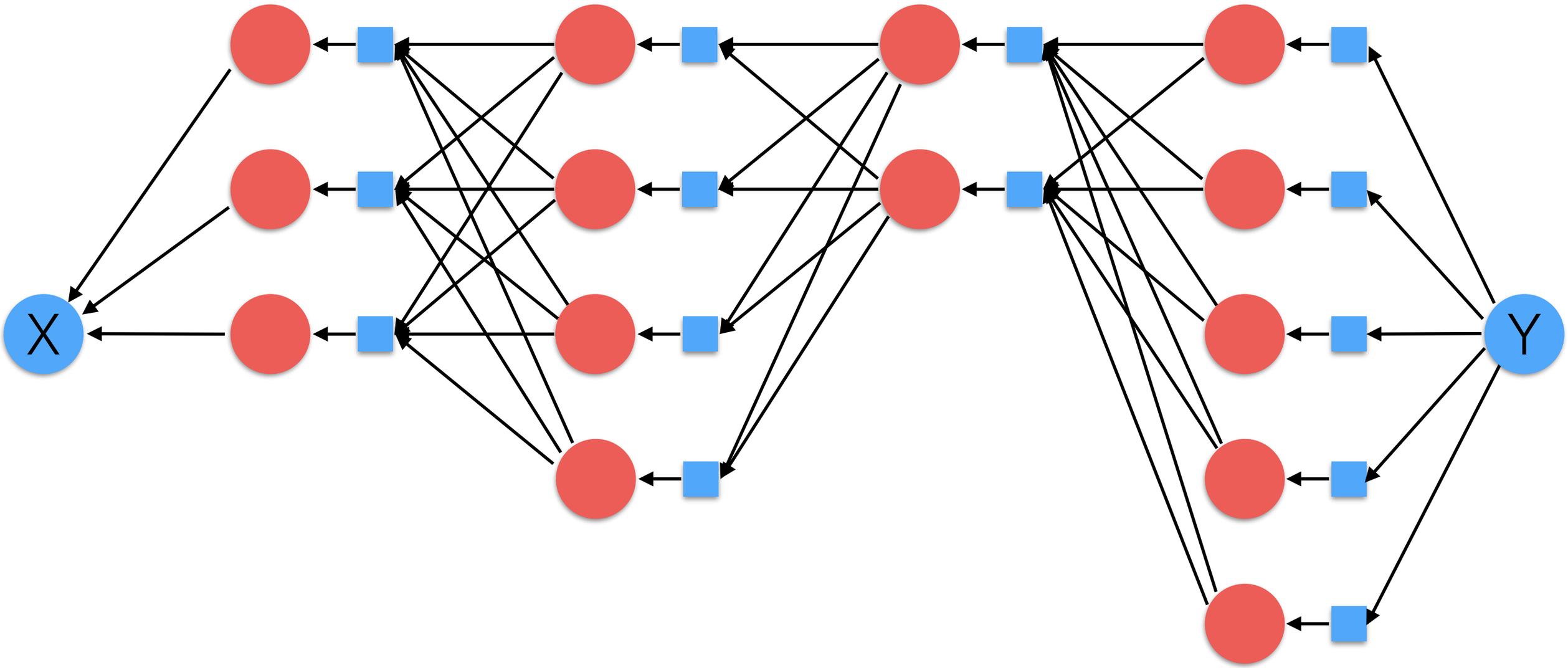
# Neural Networks Example



# Neural Networks Example



# Backward propagation

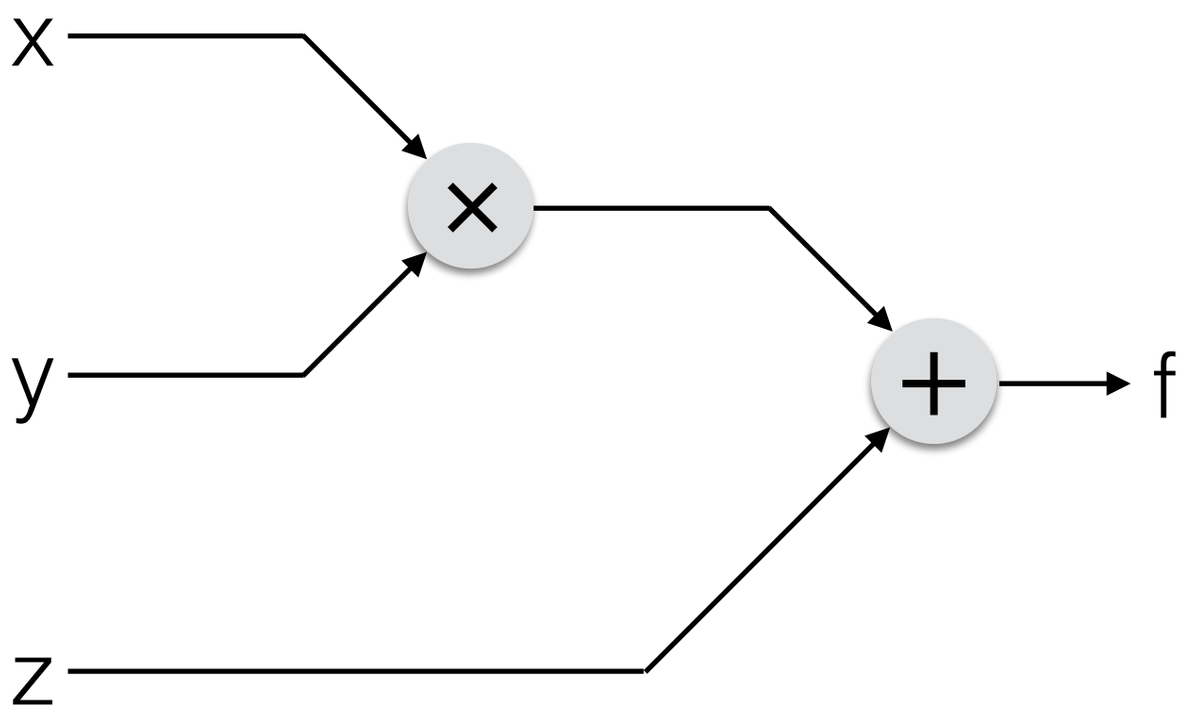


# Computational Graph

Let’s step back.

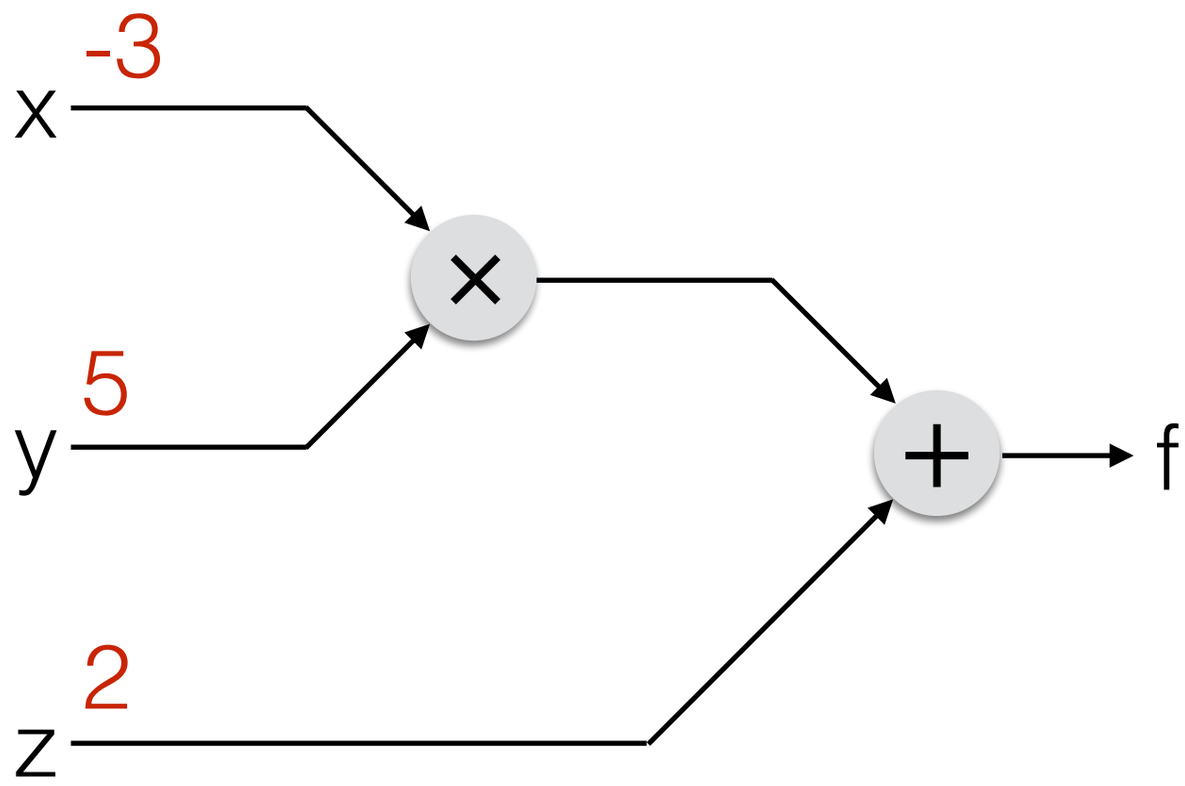
# Computational Graph

Function: f(x, y, z) = x*y+z



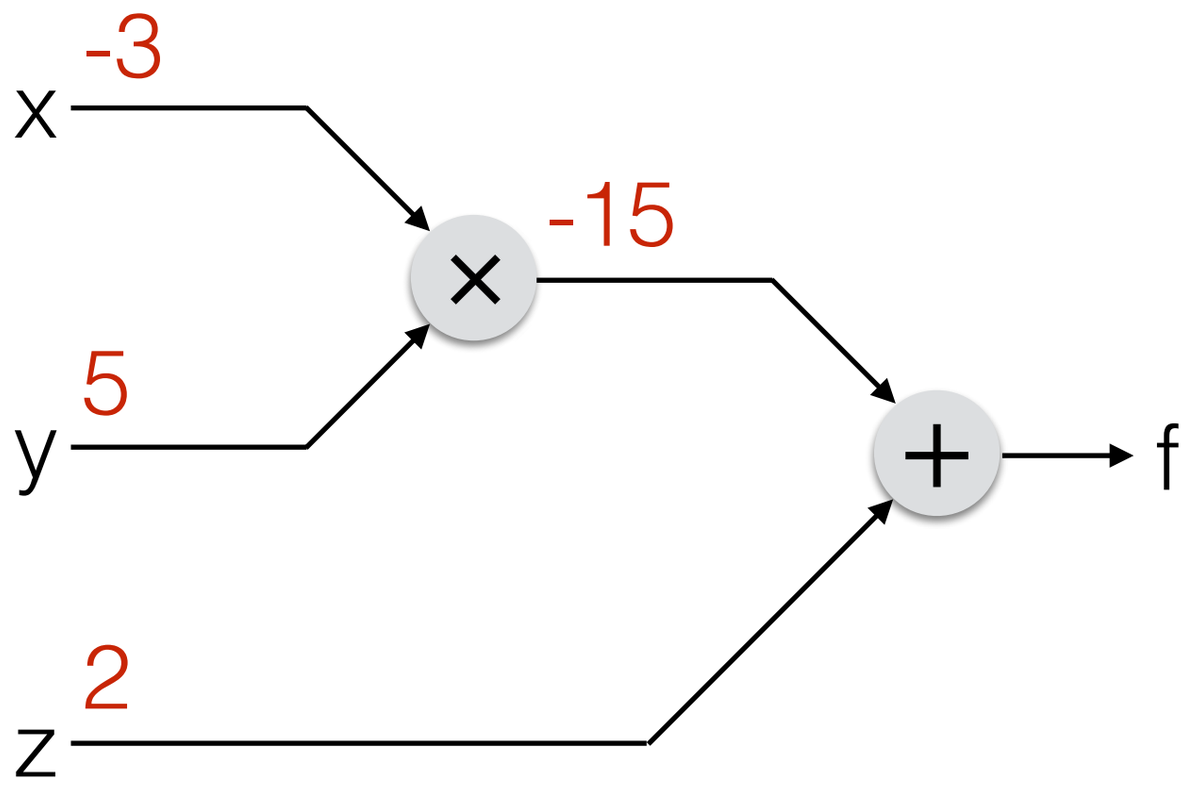
# Computational Graph

Function: f(x, y, z) = x*y+z



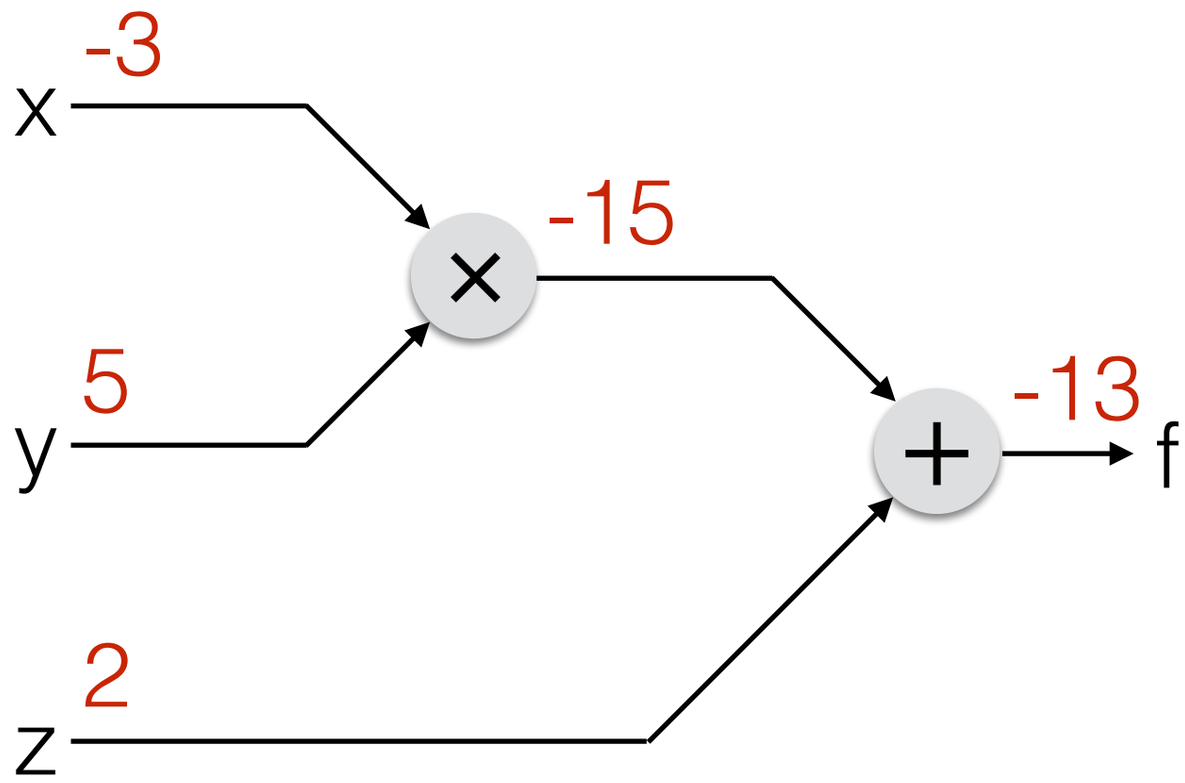
# Computational Graph

Function: f(x, y, z) = x*y+z



# Computational Graph

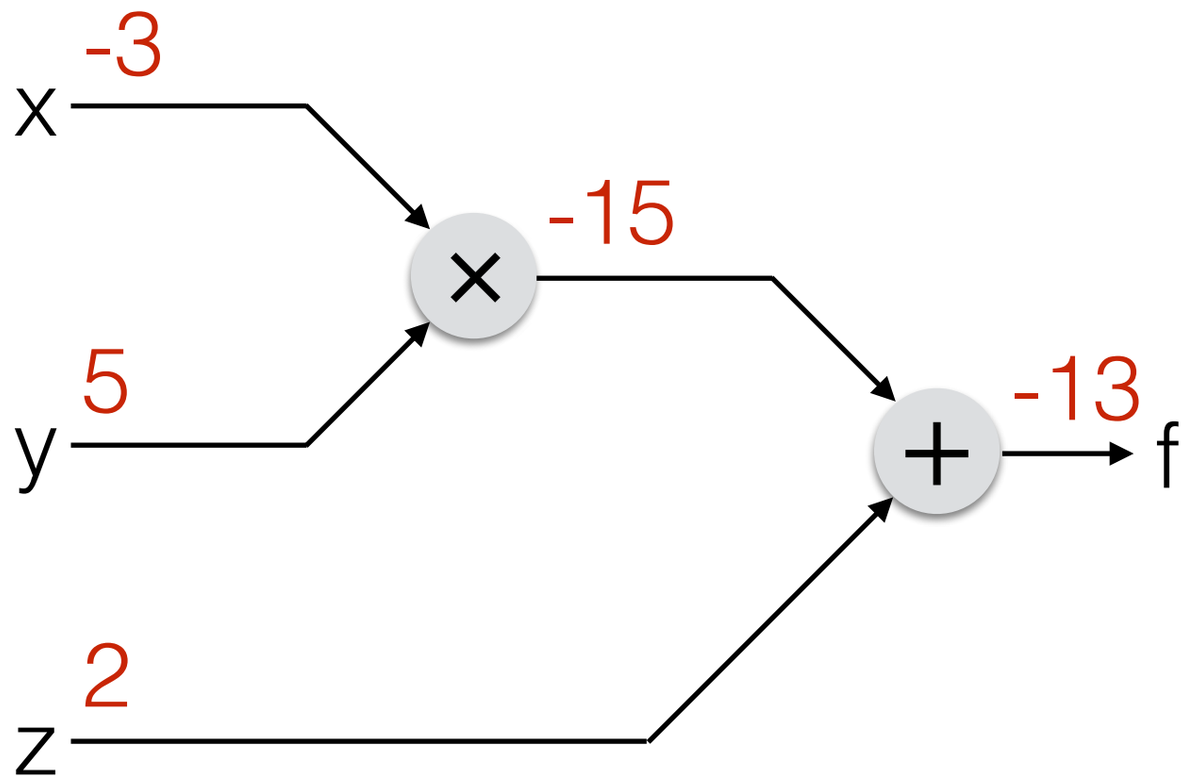
Function: f(x, y, z) = x*y+z



# Computational Graph

Function: f(x, y, z) = x*y+z

Backpropagation



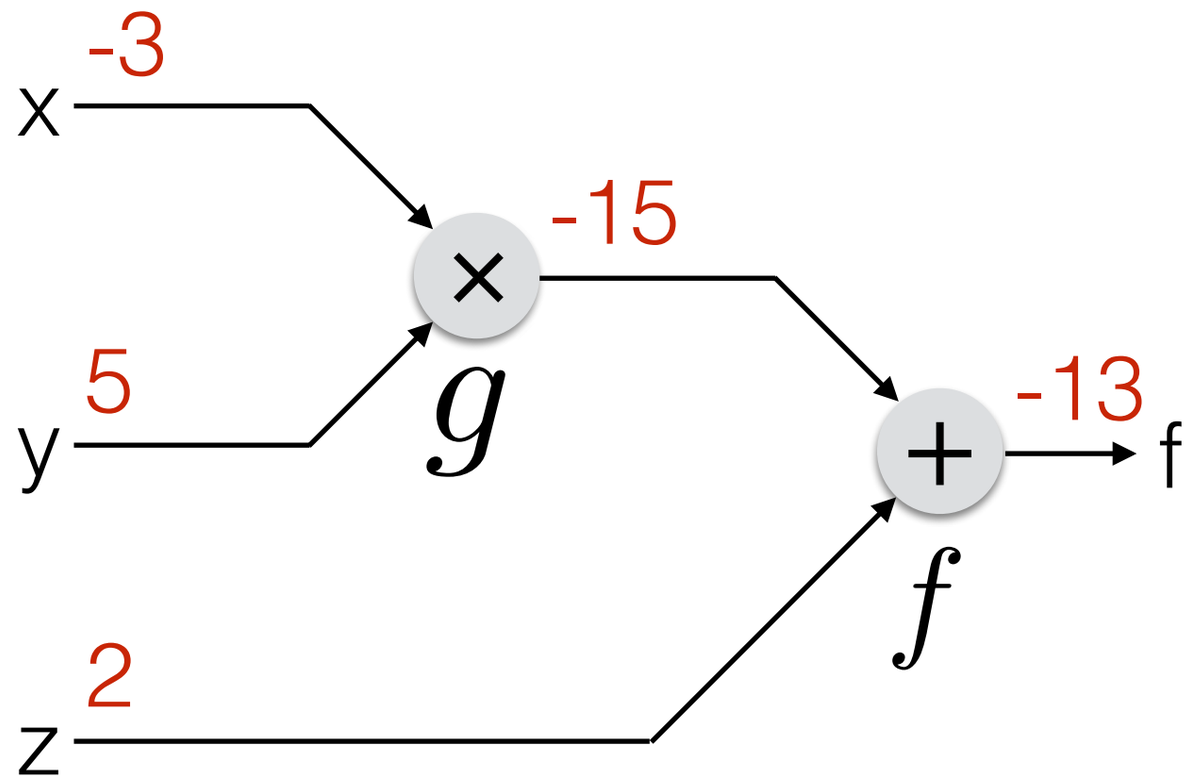
# Computational Graph

Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y$$

$$f = g + z$$



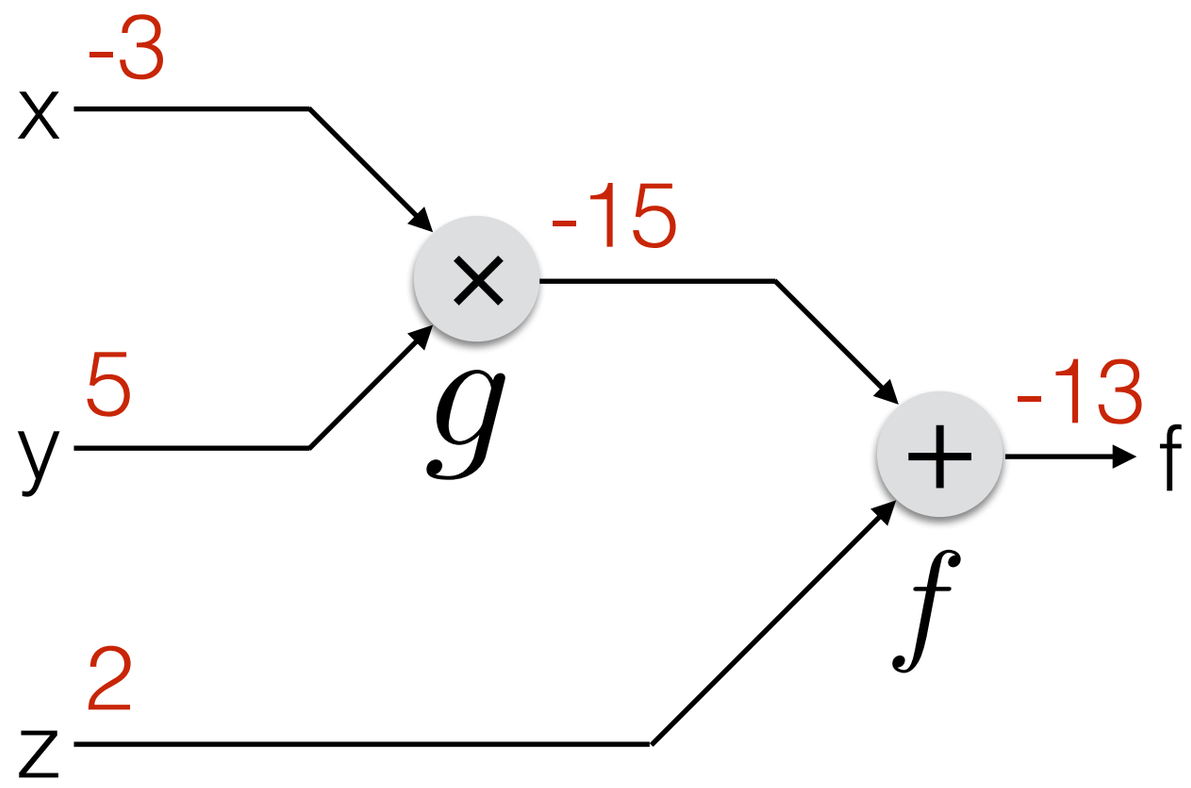
# Computational Graph

Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$



# Computational Graph

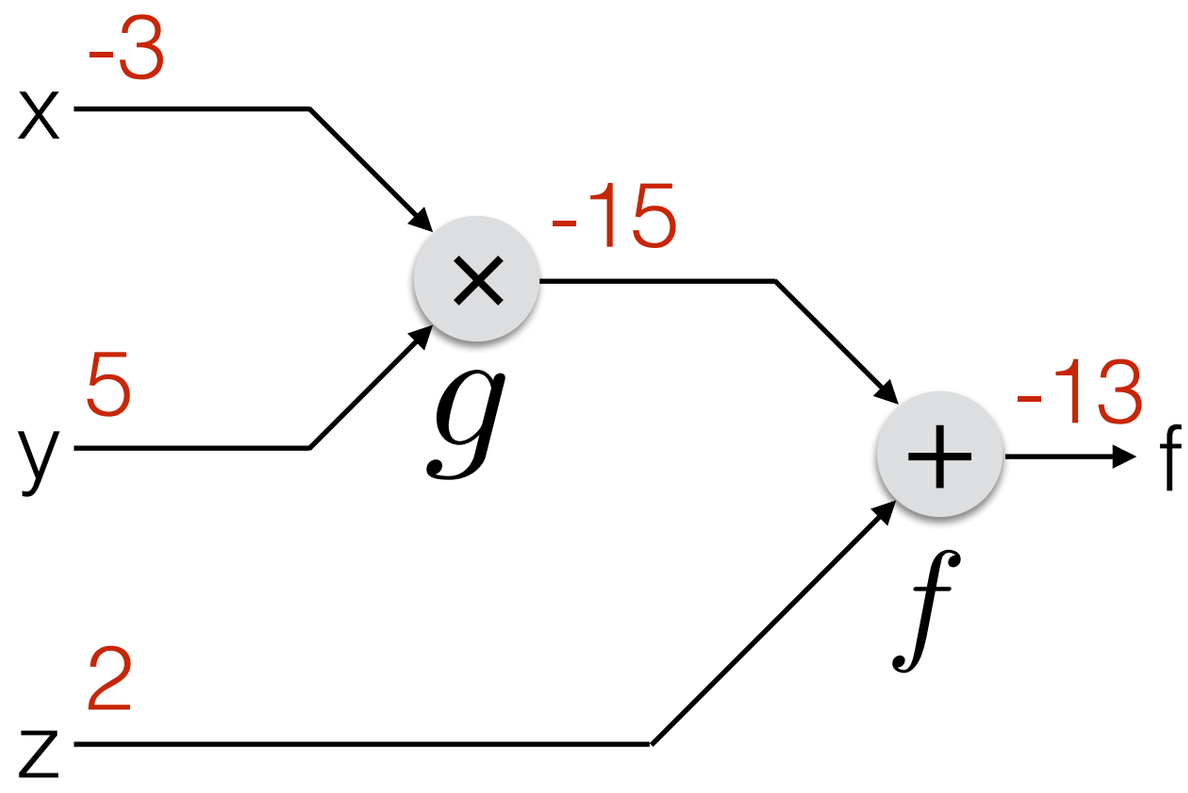
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

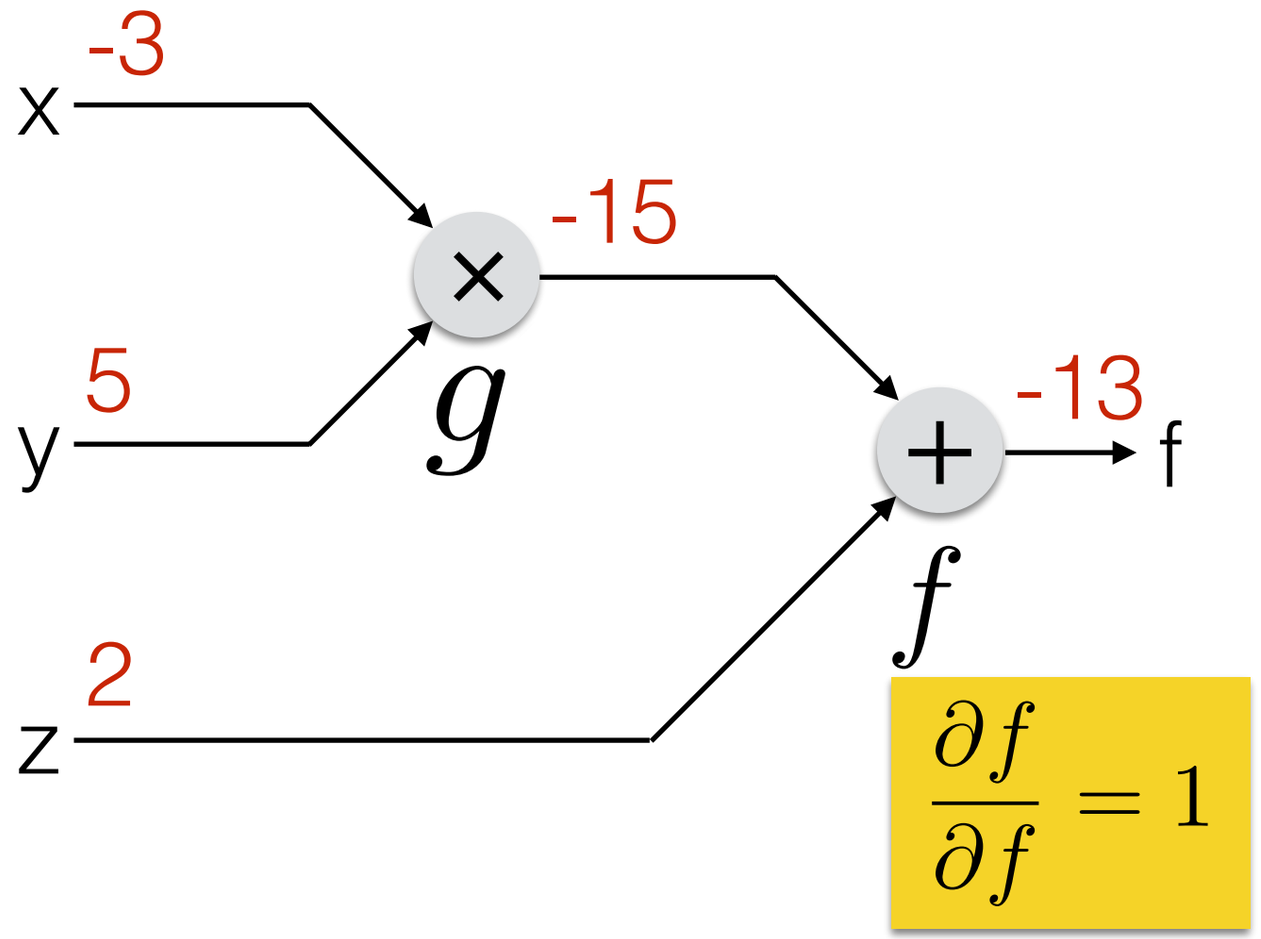
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

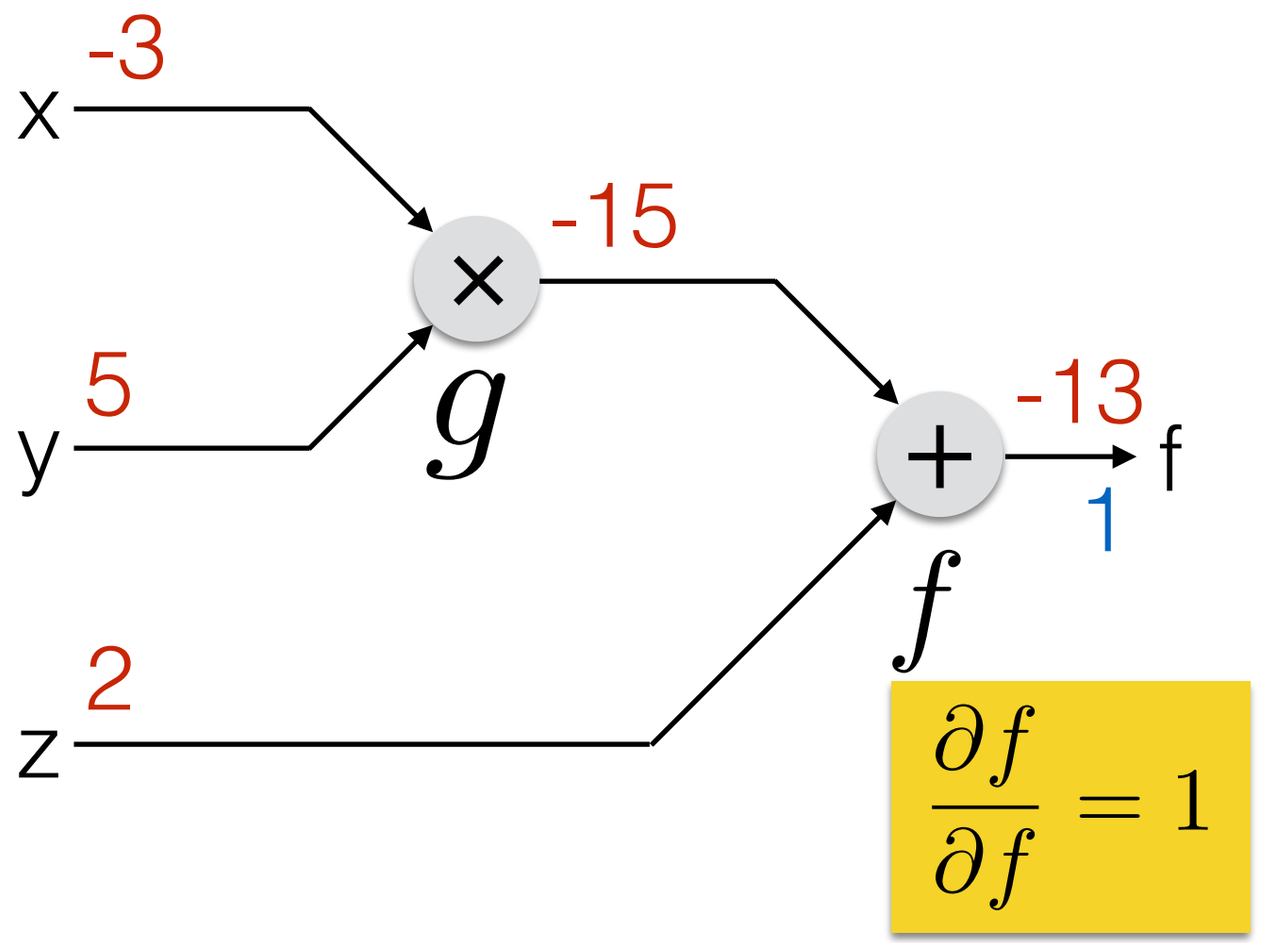
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

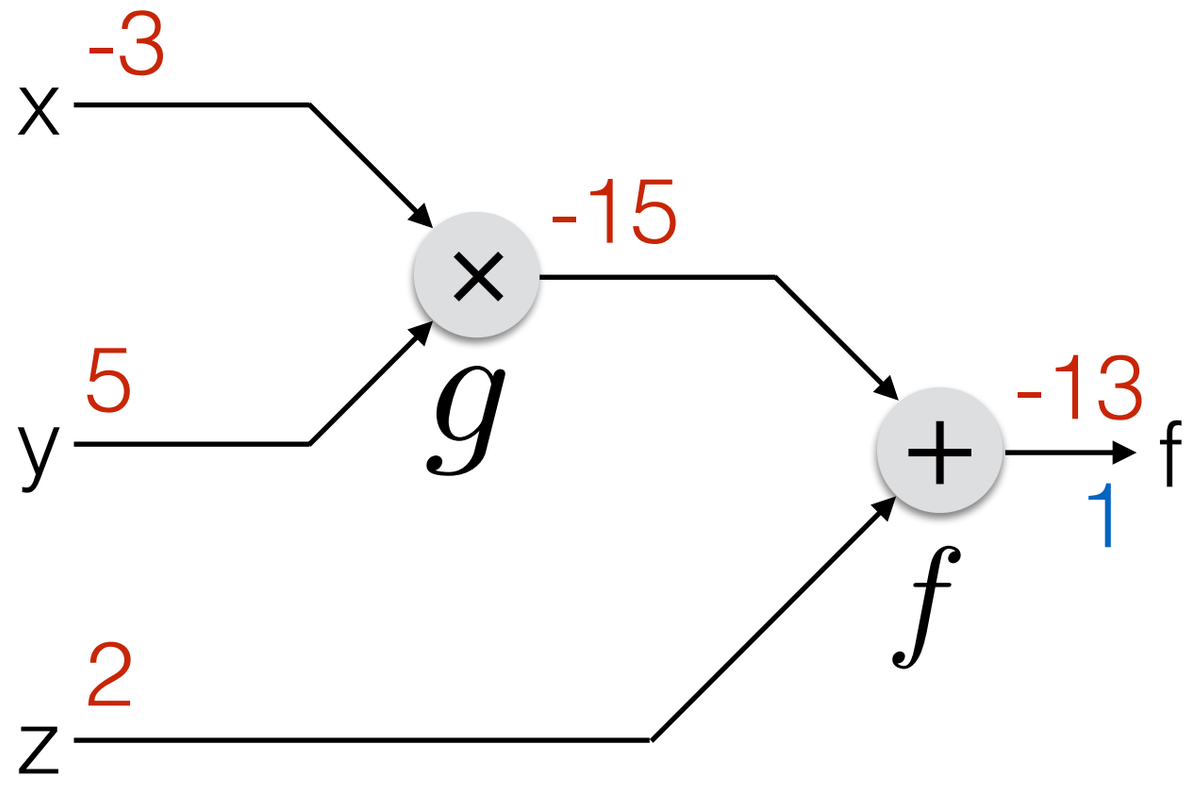
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$$\frac{\partial f}{\partial z} = 1$$

# Computational Graph

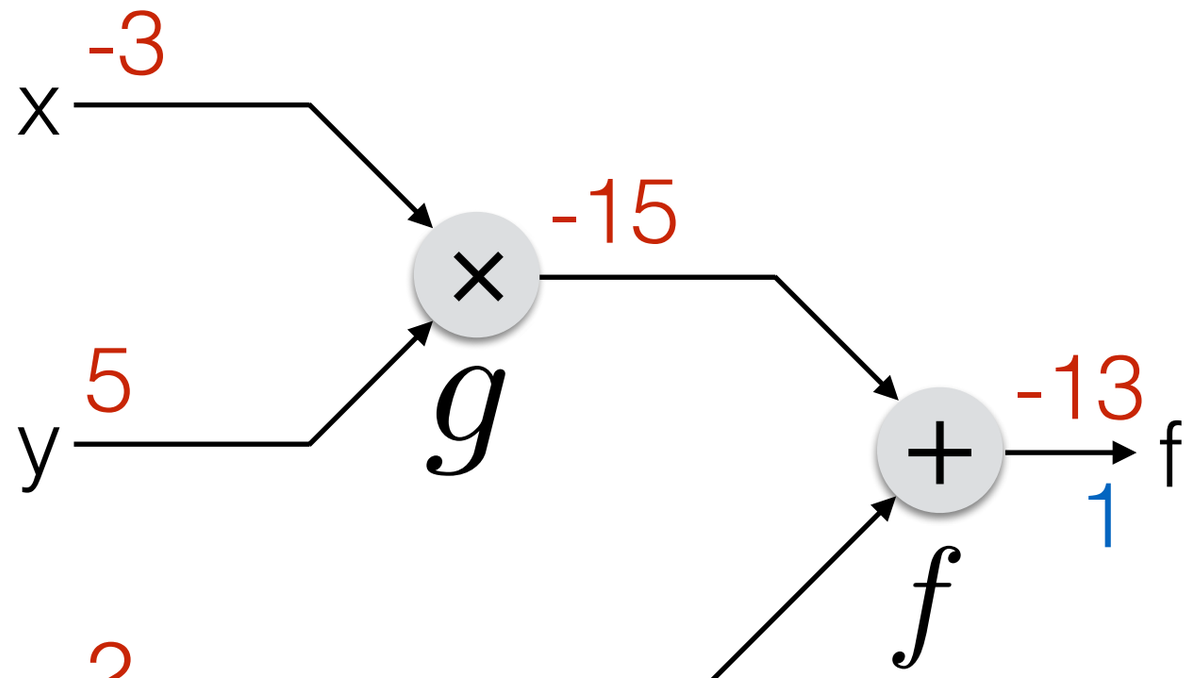
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$




$$\frac{\partial f}{\partial z} = 1$$

# Computational Graph

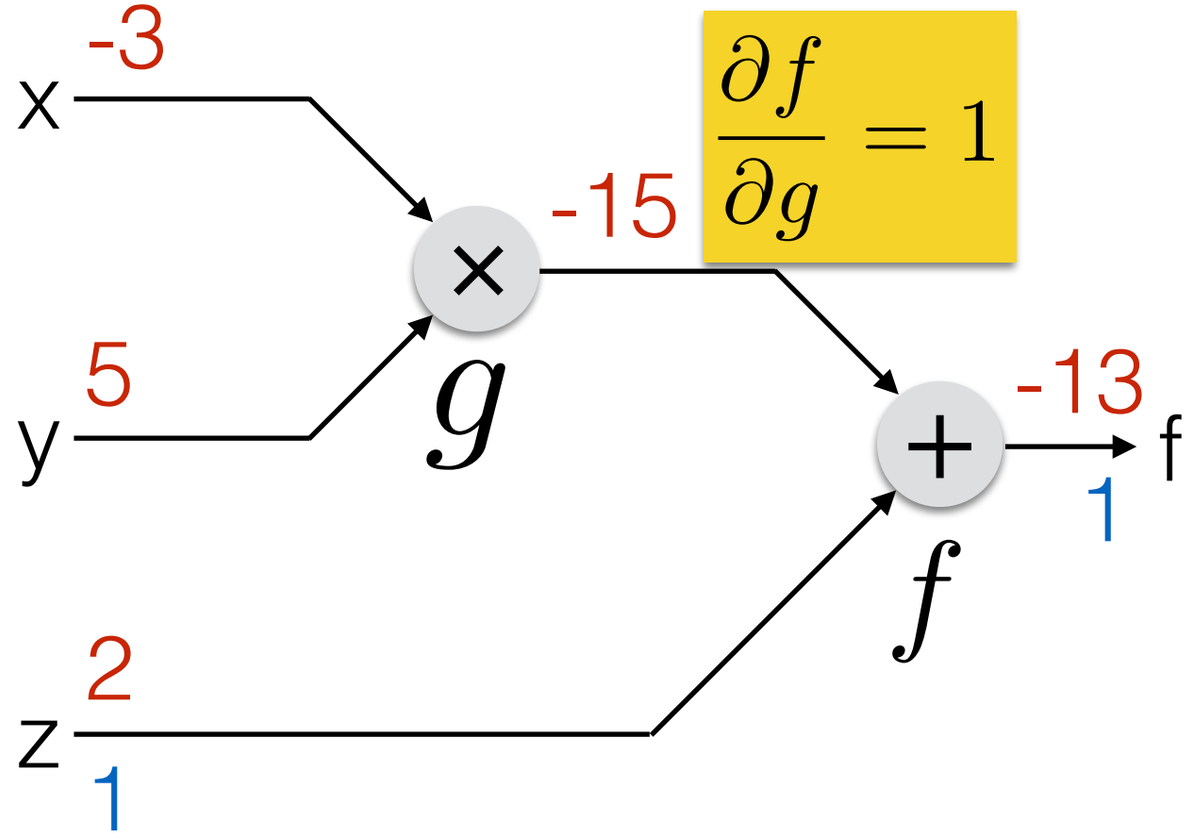
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

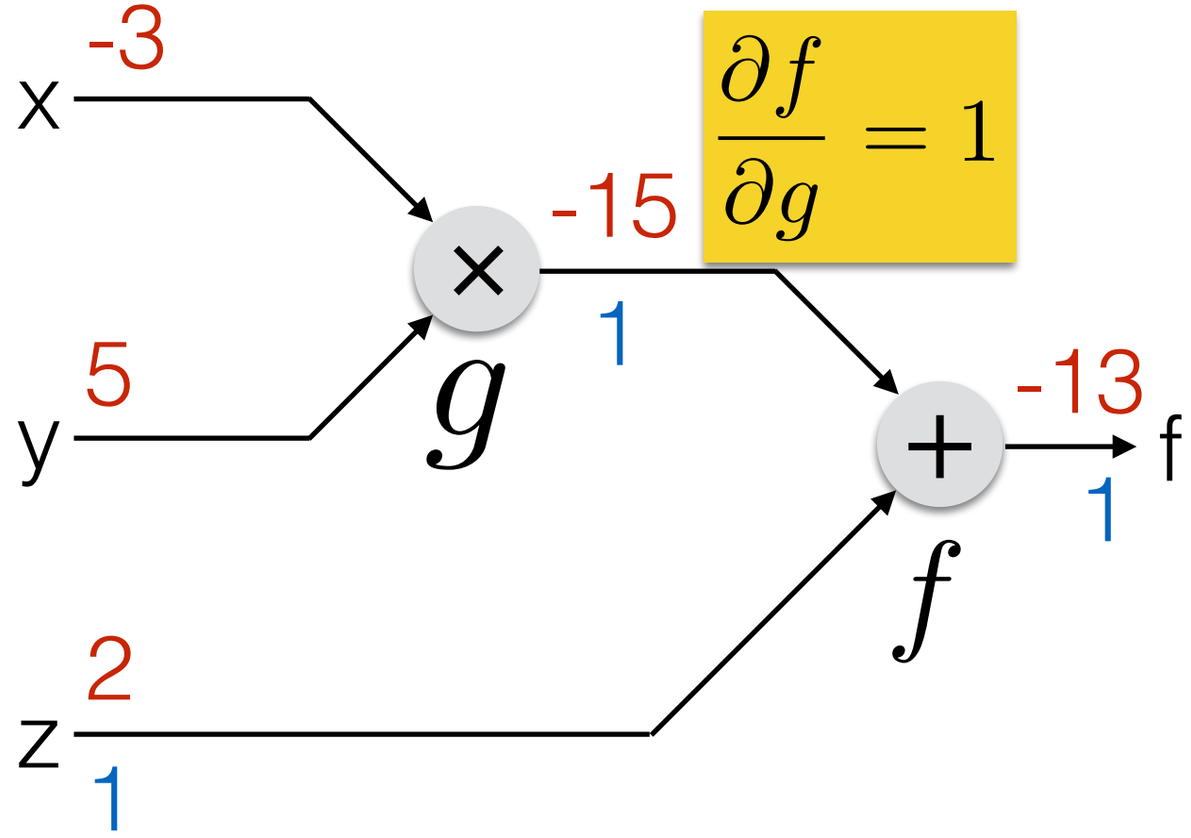
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

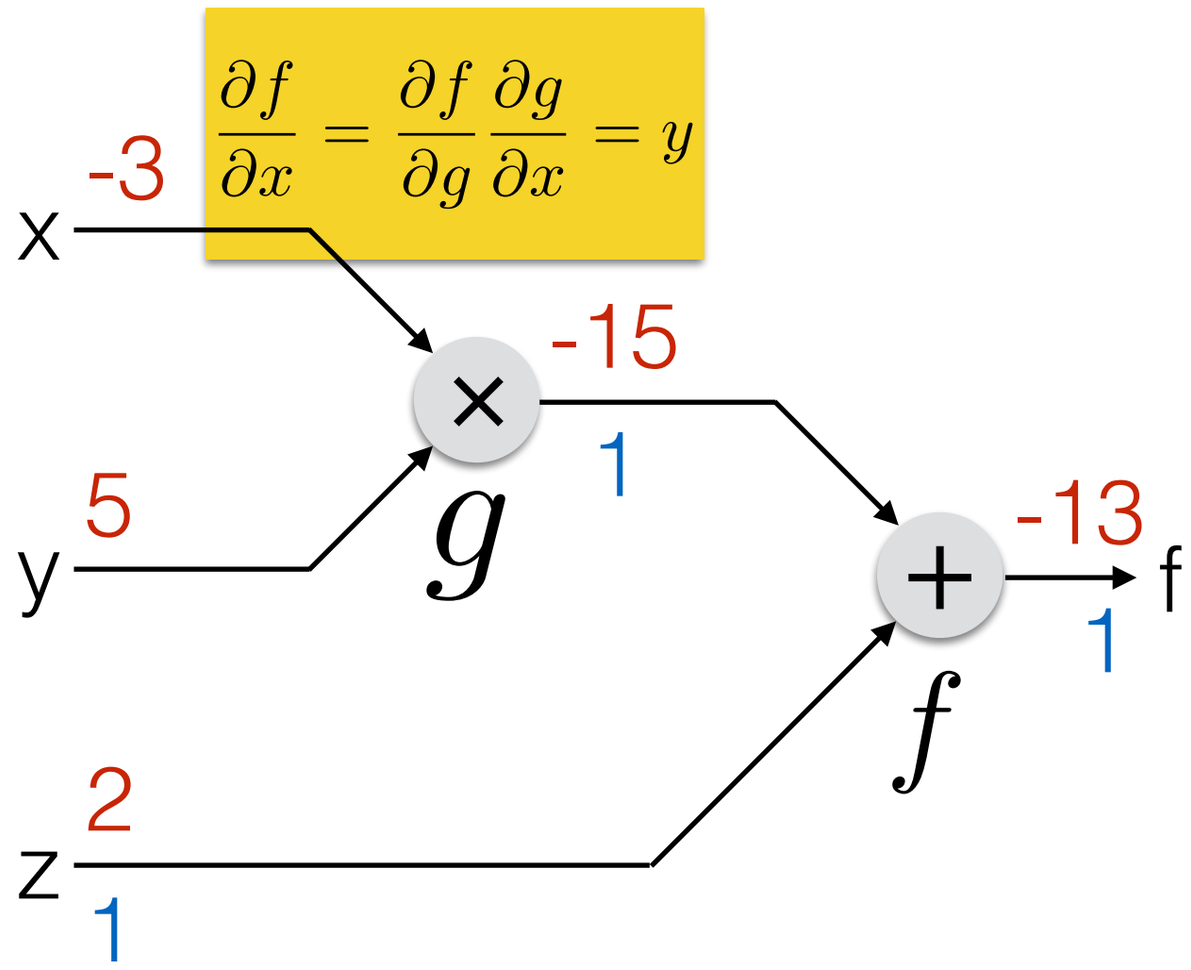
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

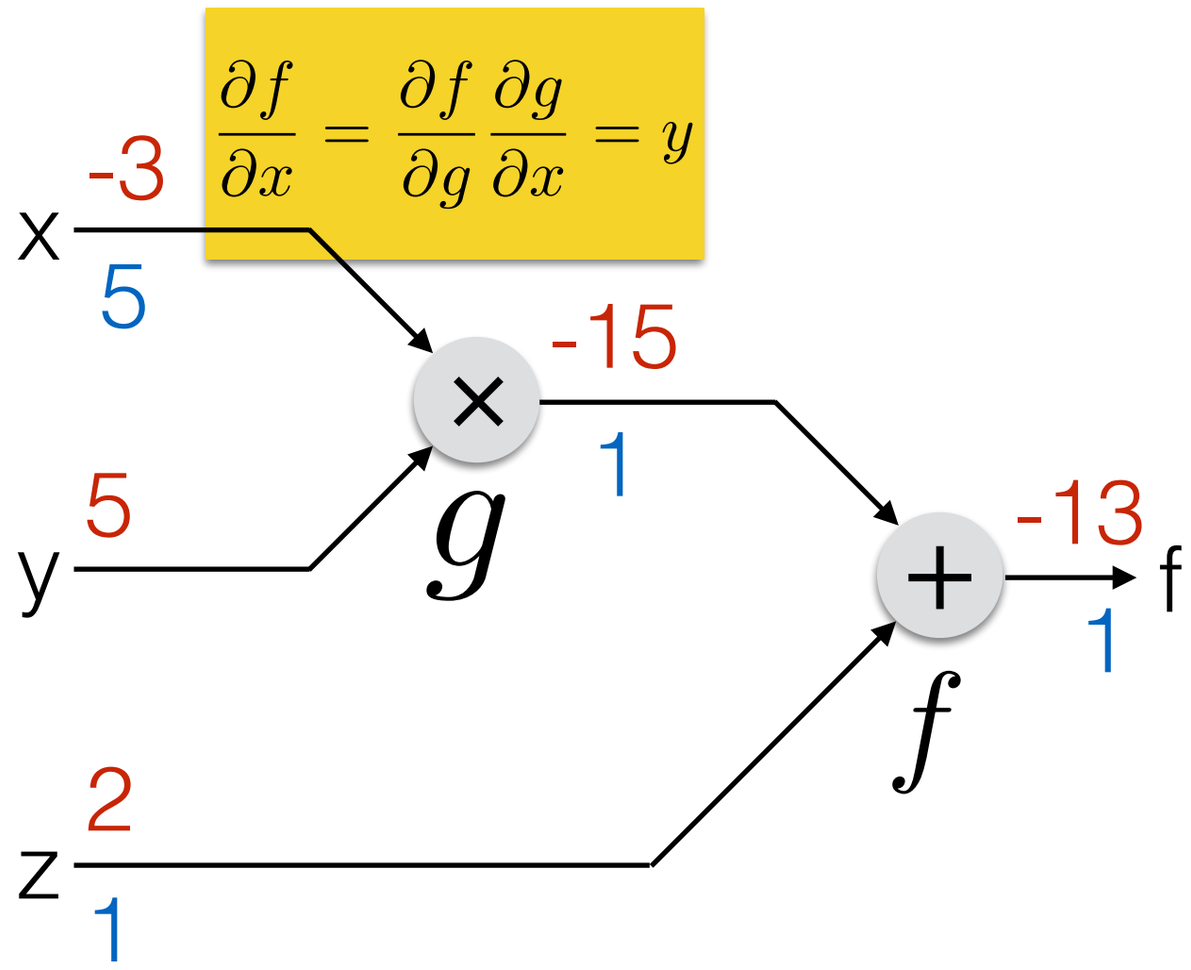
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

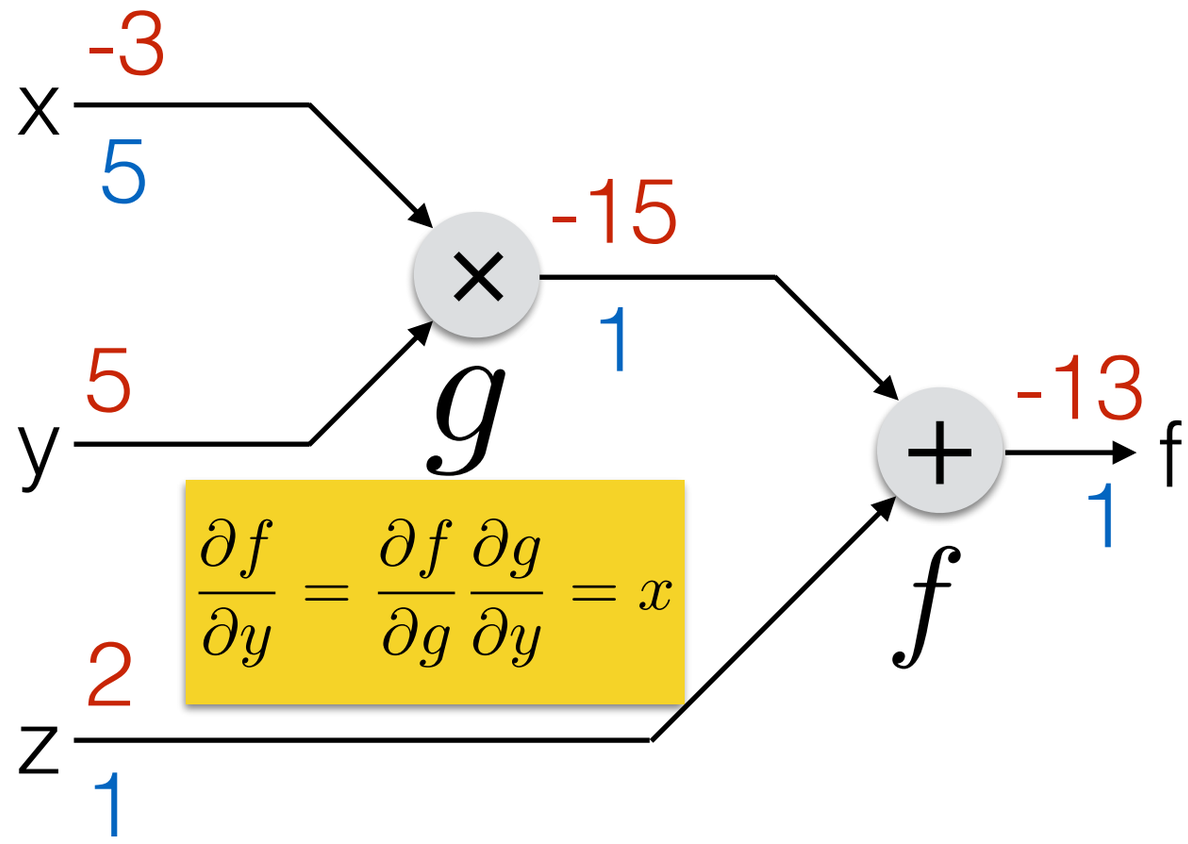
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

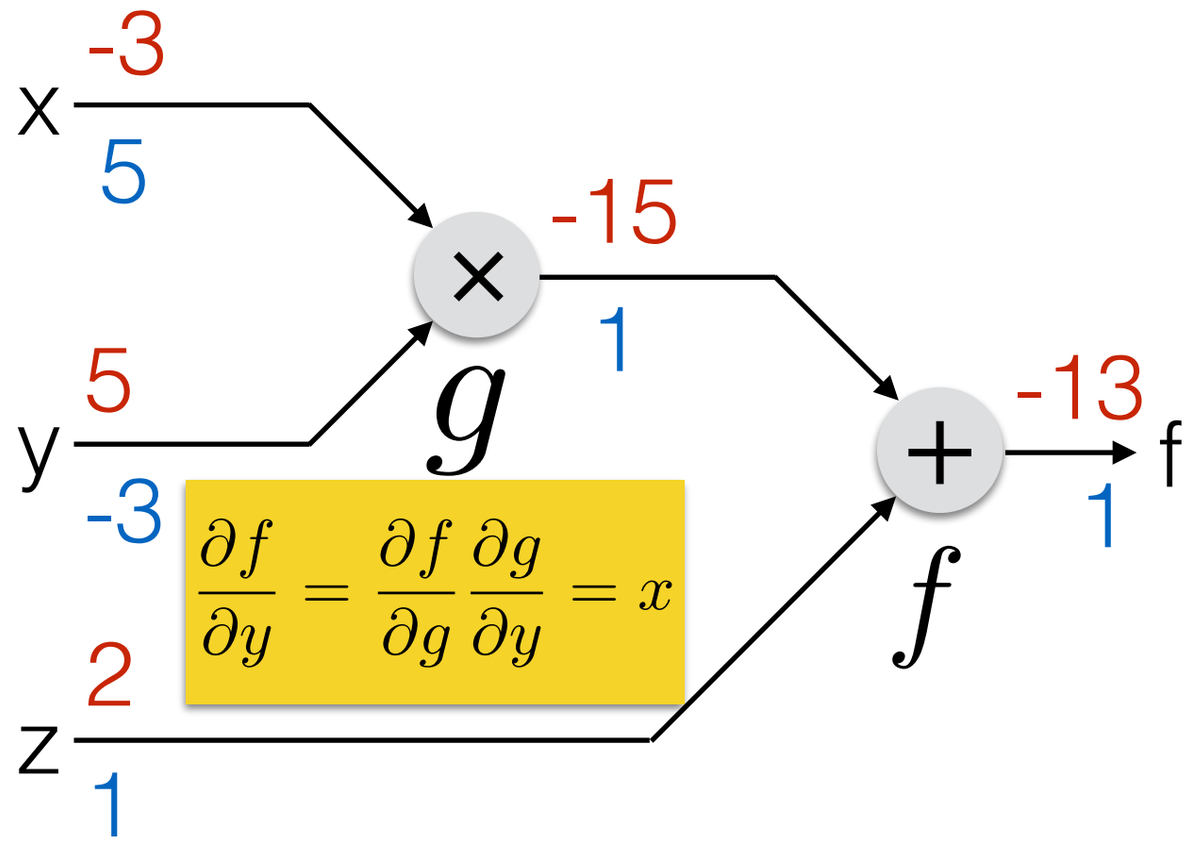
Function: f(x, y, z) = x*y+z

Backpropagation

$$g = x \times y \qquad \frac{\partial q}{\partial x} = y \qquad \frac{\partial q}{\partial y} = x$$

$$f = g + z \qquad \frac{\partial f}{\partial g} = 1 \qquad \frac{\partial f}{\partial z} = 1$$

We want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



# Computational Graph

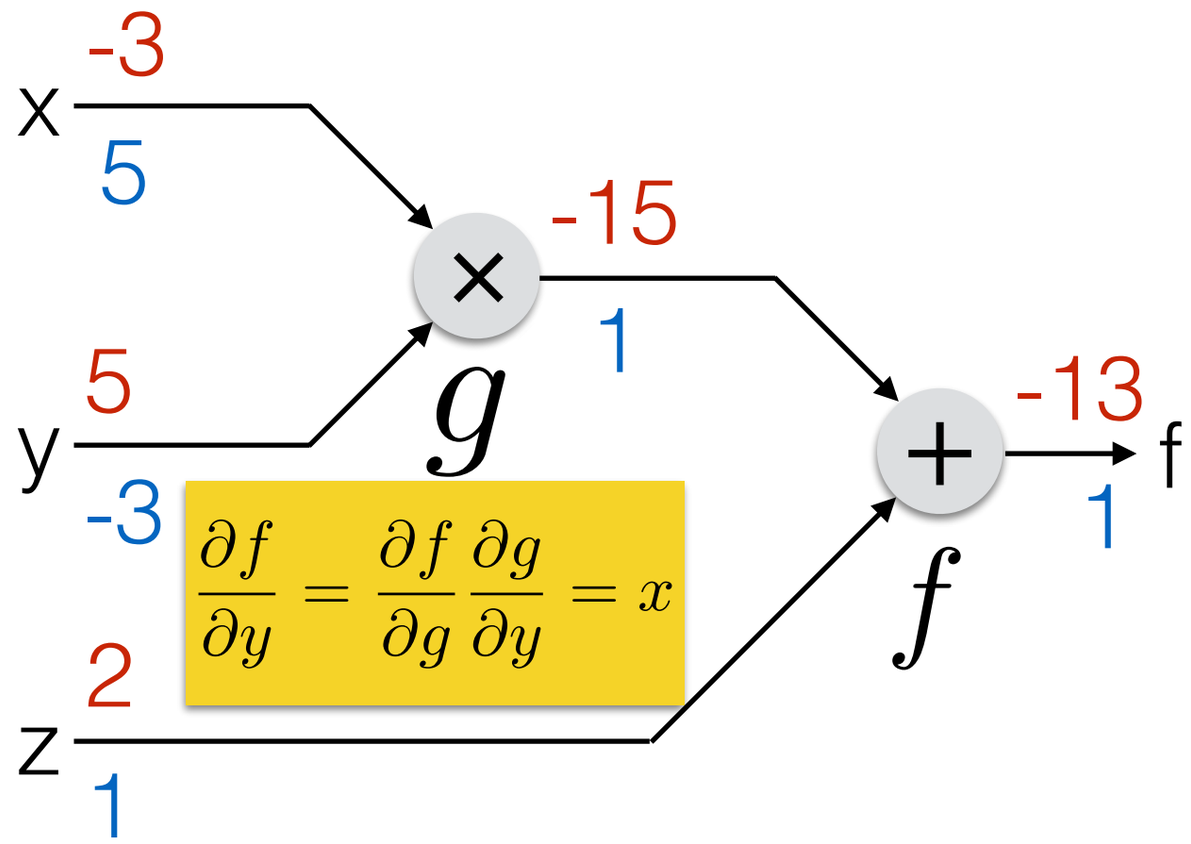
Function: f(x, y, z) = x*y+z

Backpropagation

$$\frac{\partial f}{\partial x} = 5$$

We get: $$\frac{\partial f}{\partial y} = -3$$

$$\frac{\partial f}{\partial z} = 1$$



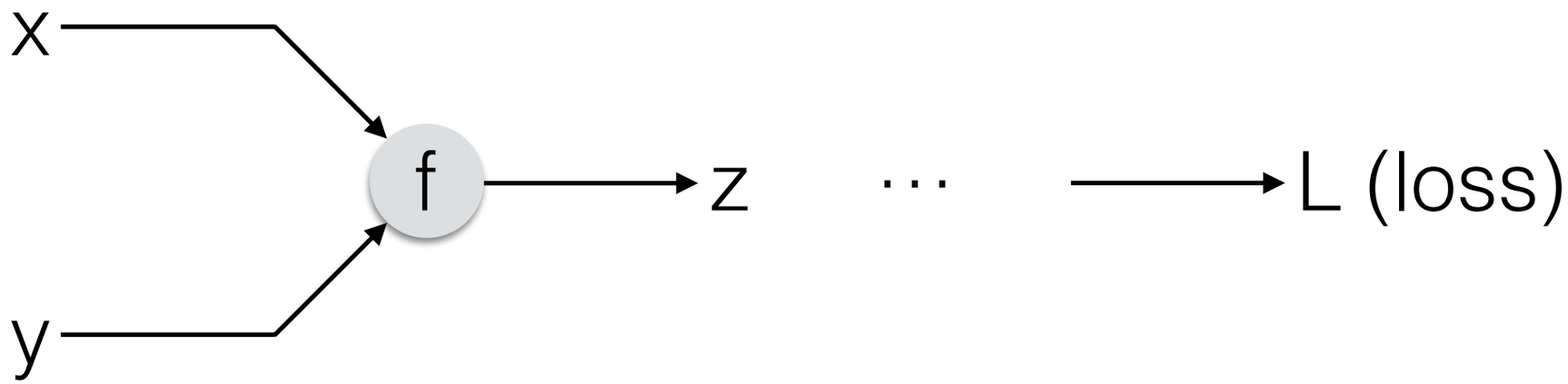
# Computational Graph

Note: a local view of it

# Computational Graph

Function: f(x, y, z) = x*y+z

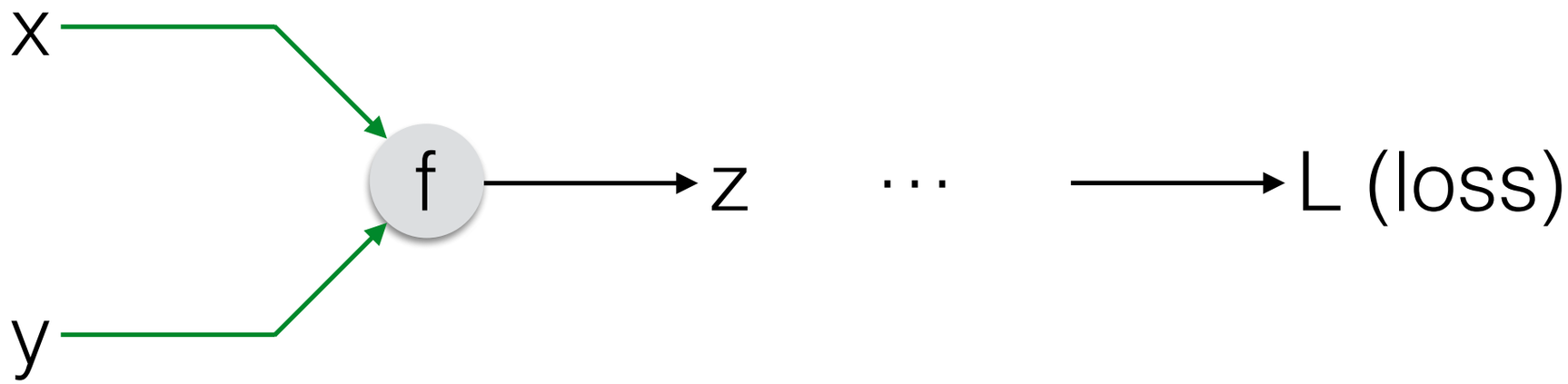
**A local view** of backpropagation
Forward propagation



# Computational Graph

Function: f(x, y, z) = x*y+z

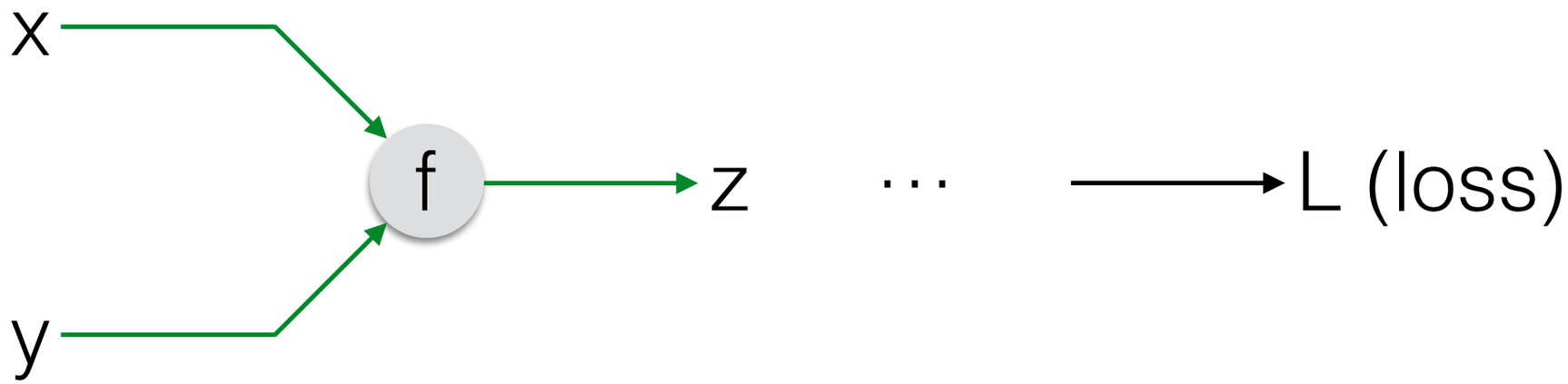
**A local view** of backpropagation
Forward propagation



# Computational Graph

Function: f(x, y, z) = x*y+z

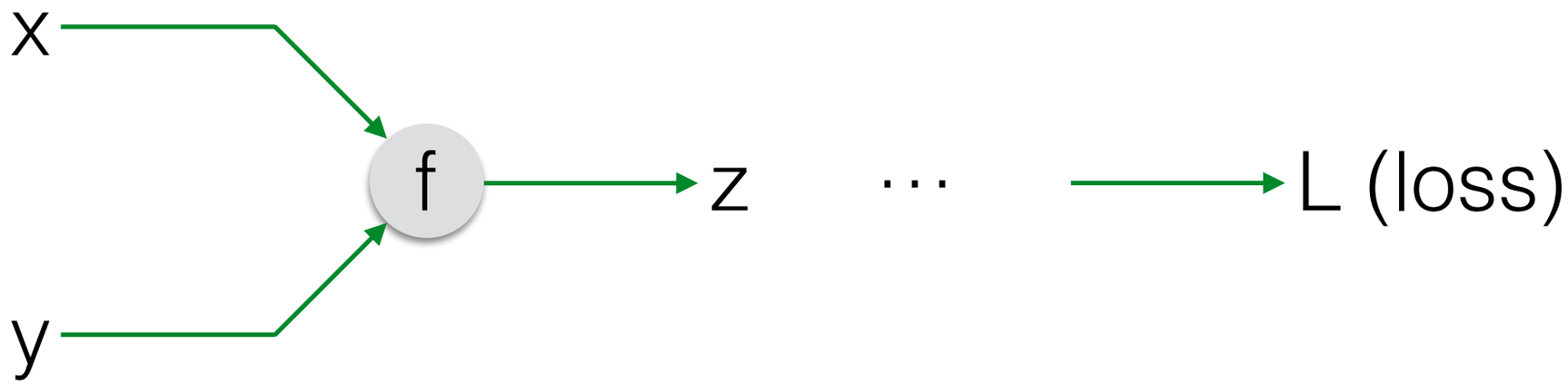
**A local view** of backpropagation
Forward propagation



# Computational Graph

Function: f(x, y, z) = x*y+z

**A local view** of backpropagation
Forward propagation

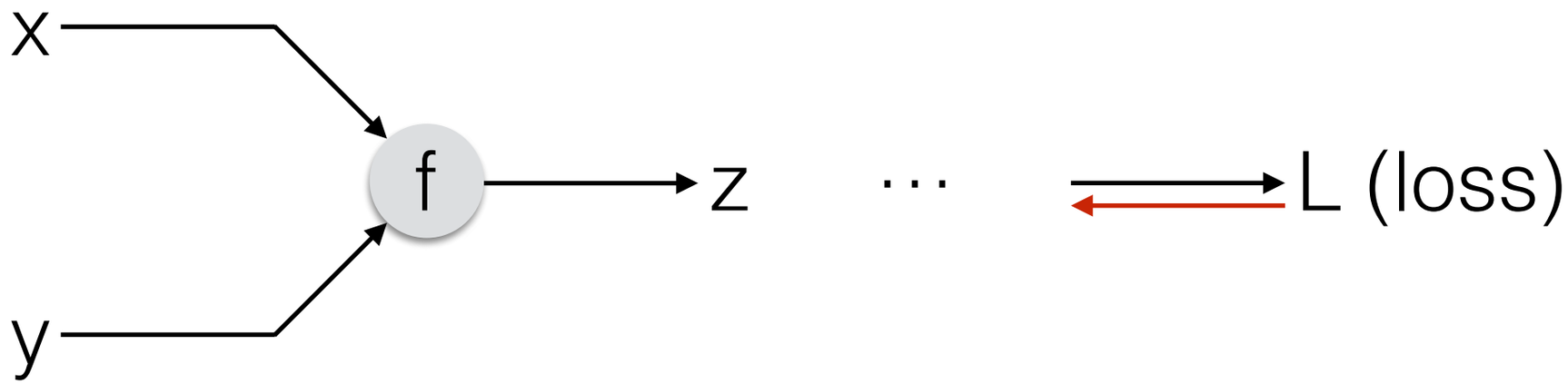


# Computational Graph

Function: f(x, y, z) = x*y+z

**A local view** of backpropagation

Backpropagation

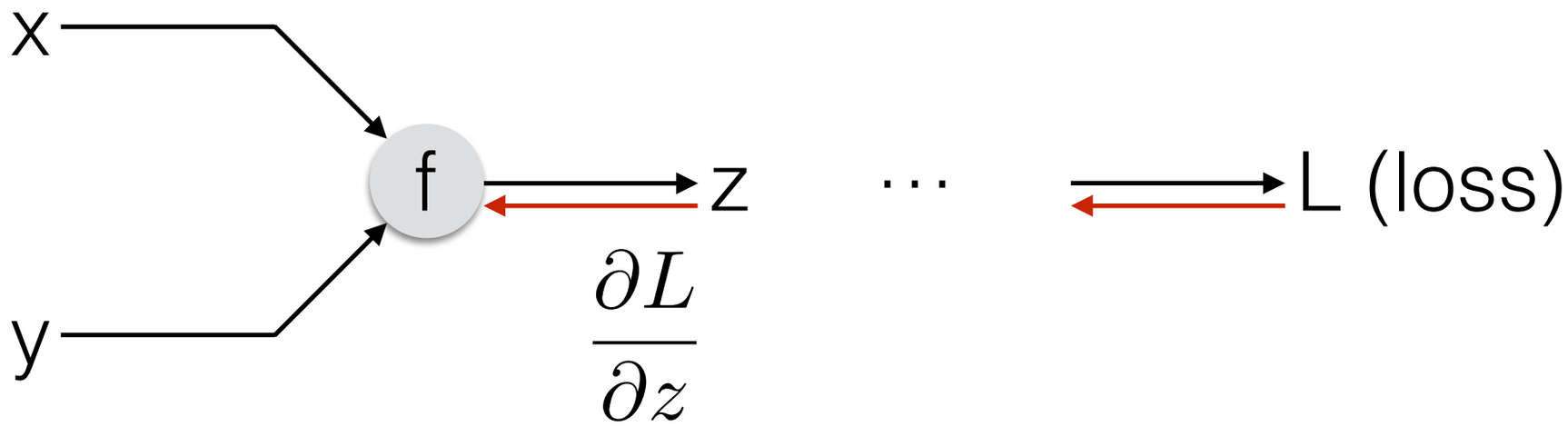


# Computational Graph

Function: f(x, y, z) = x*y+z

**A local view** of backpropagation

Backpropagation



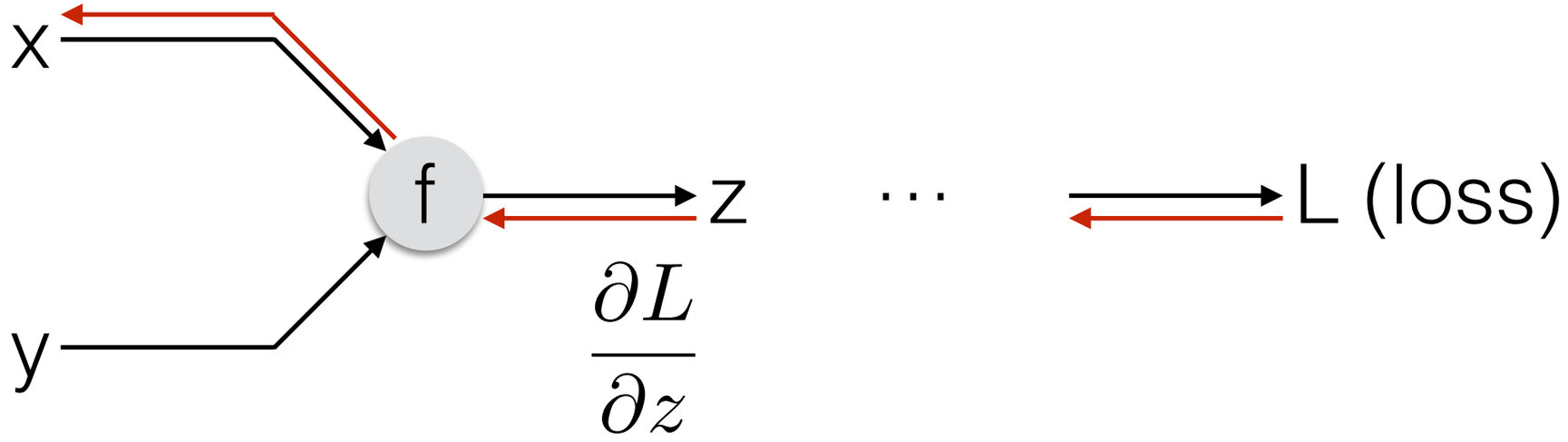
# Computational Graph

Function: f(x, y, z) = x*y+z

**A local view** of backpropagation

Backpropagation

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$



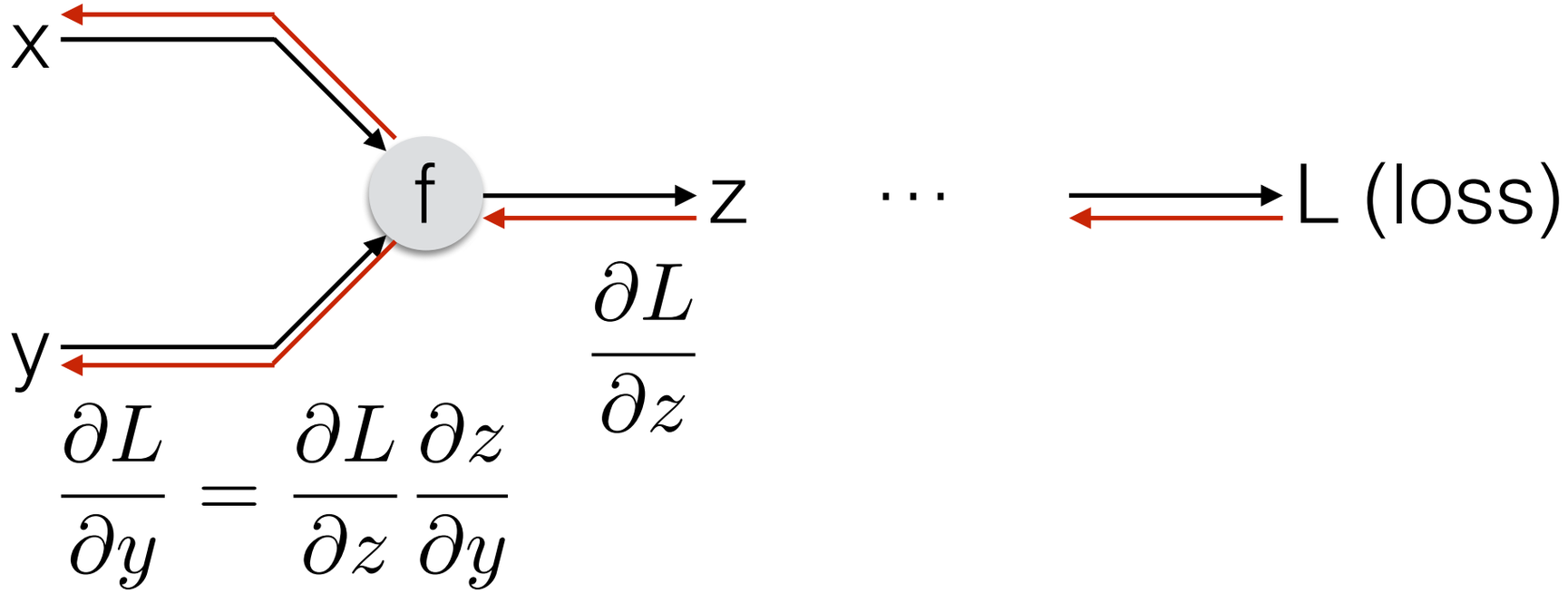
# Computational Graph

Function: f(x, y, z) = x*y+z

**A local view** of backpropagation

Backpropagation

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$



# Computational Graph

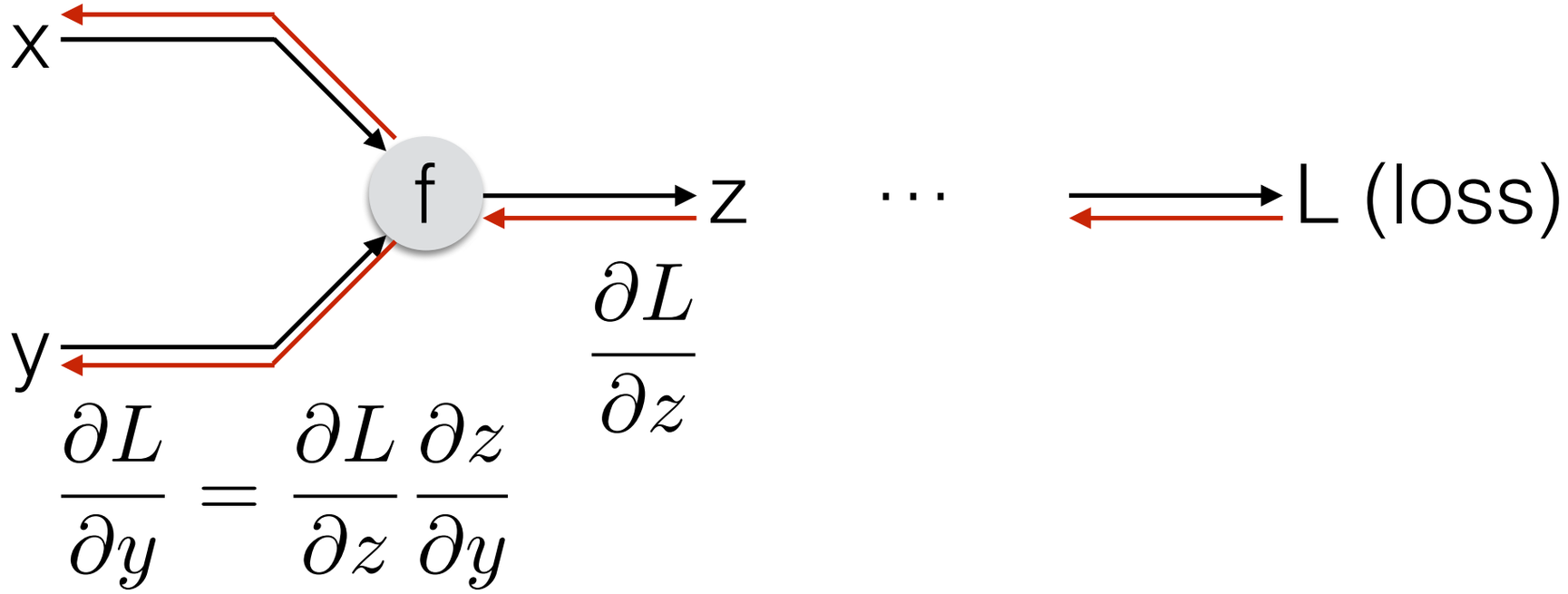
Function: f(x, y, z) = x*y+z

**A local view** of backpropa

Backpropagation

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

Requires a function to be differentiable

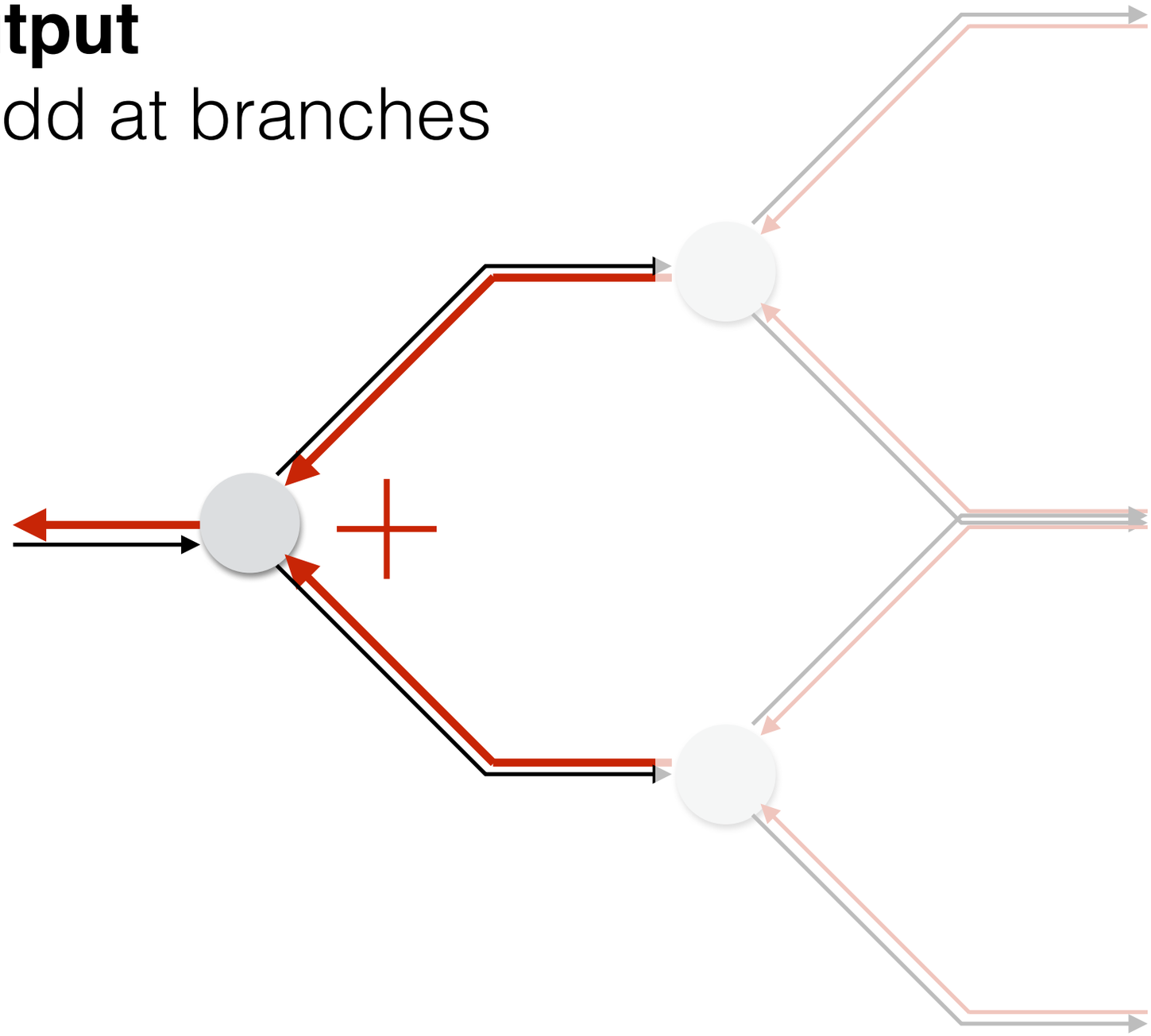


# Computational Graph

Function: f(x, y, z) = x*y+z

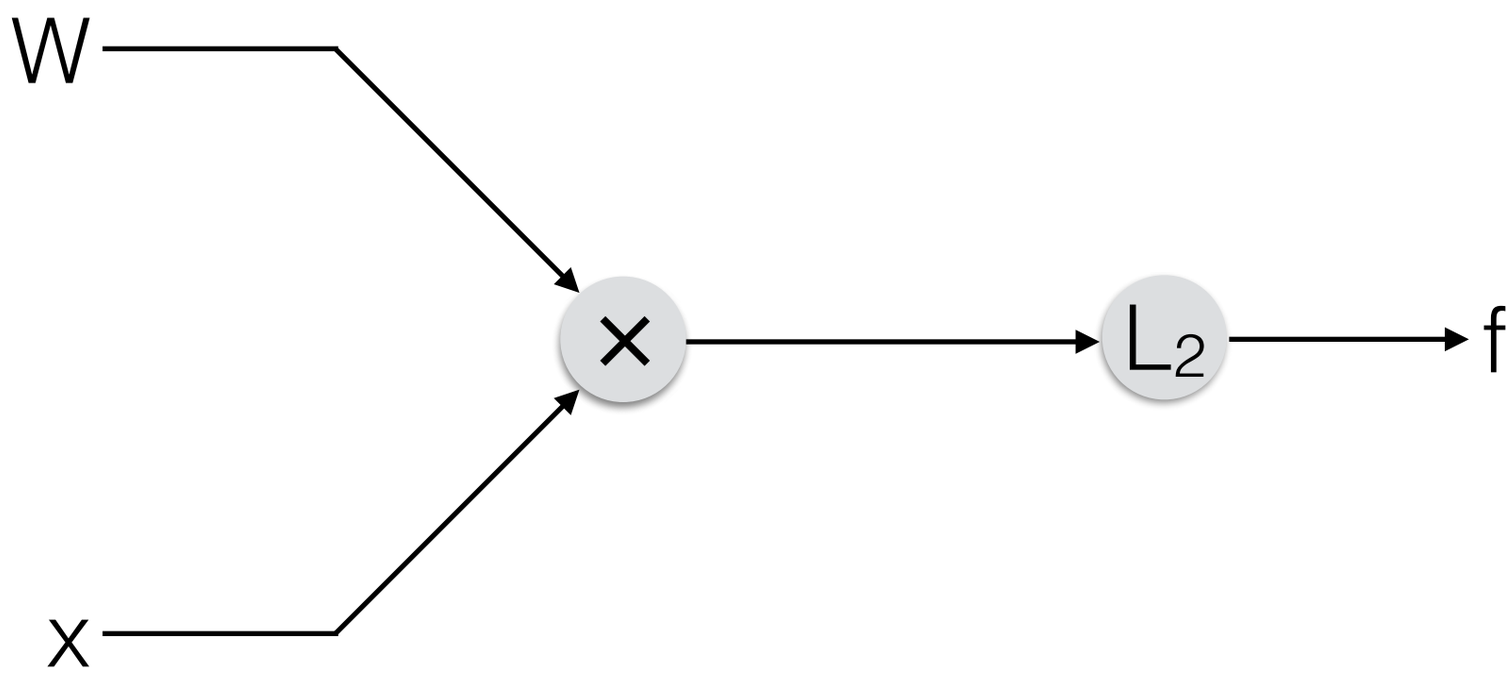
**Multiple output**

Gradients add at branches



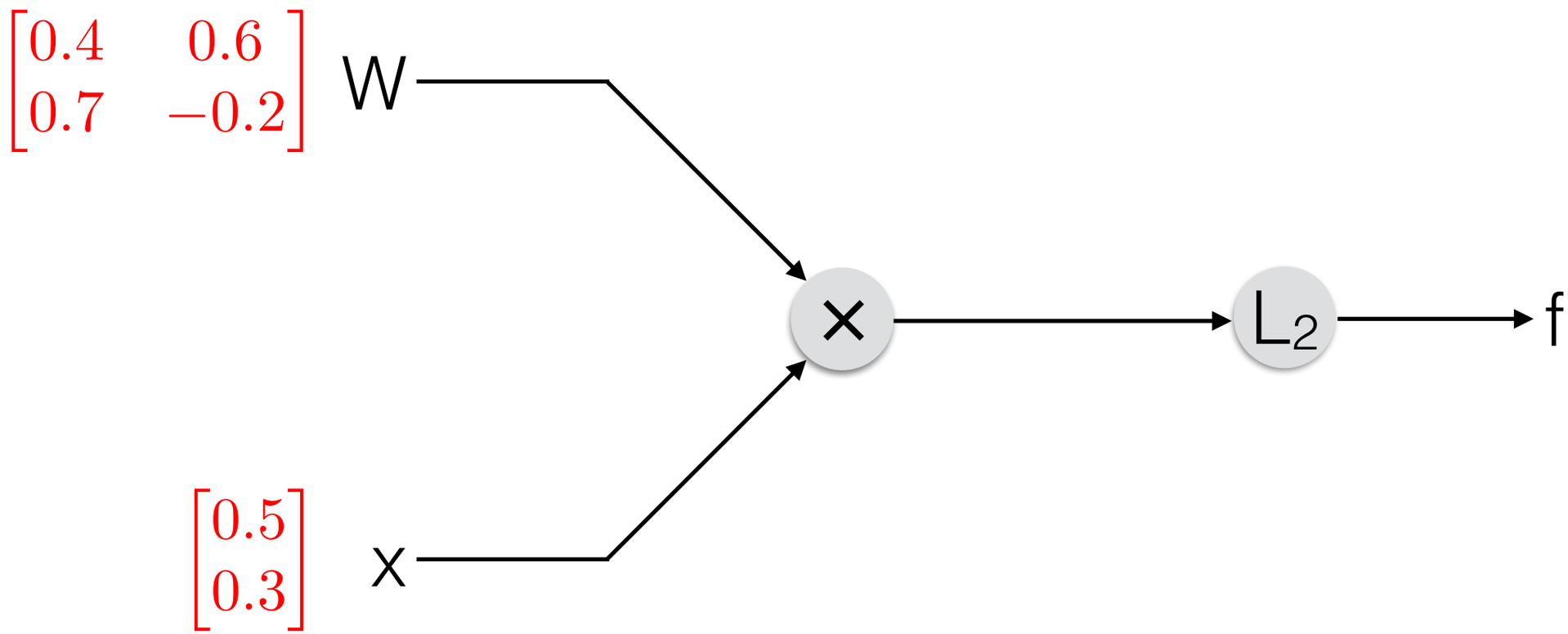
# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



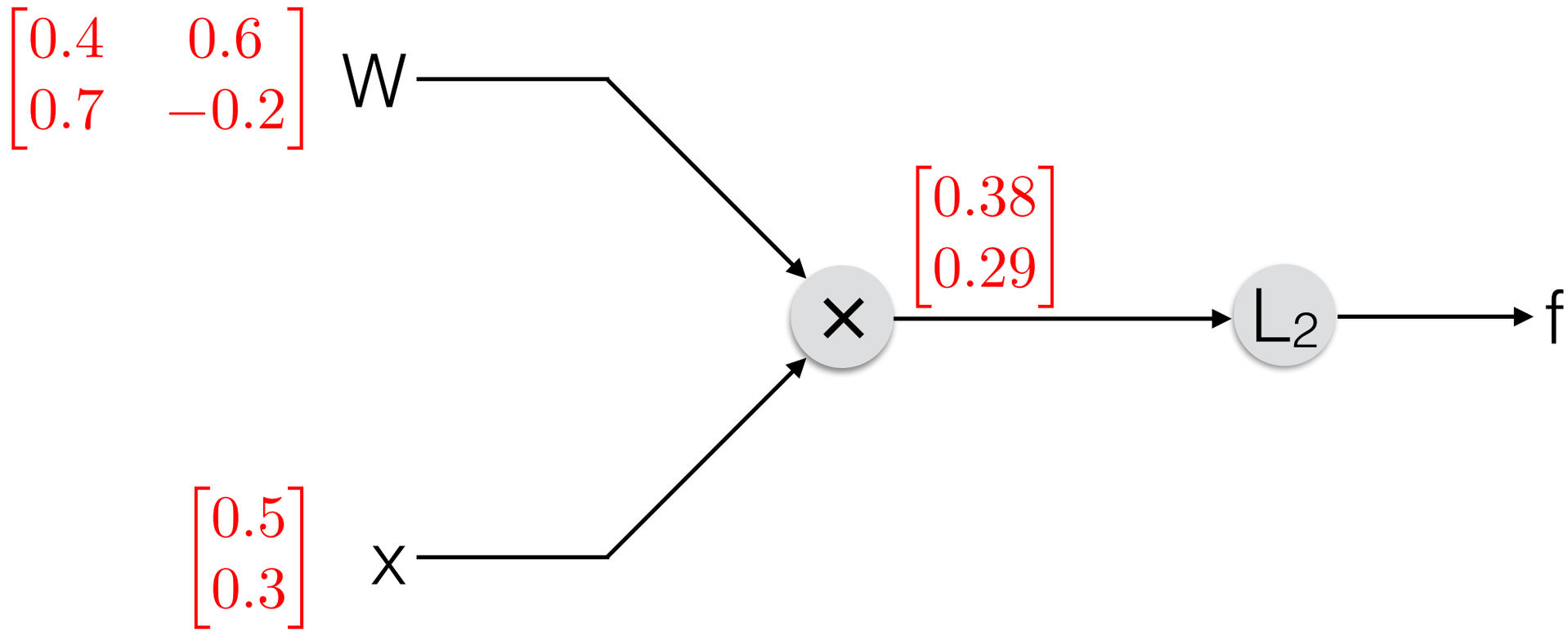
# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



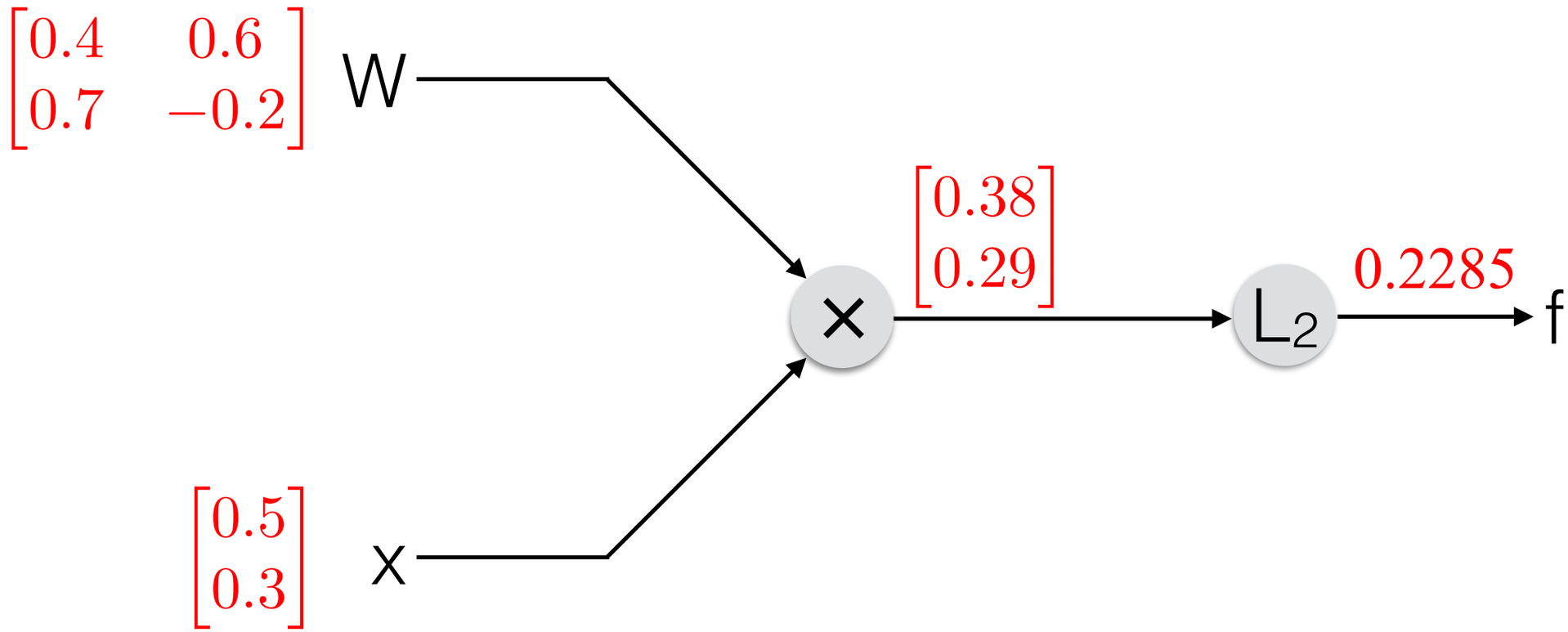
# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\frac{\partial f}{\partial W_{i,j}}, \frac{\partial f}{\partial x_i}$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\frac{\partial f}{\partial W_{i,j}}, \frac{\partial f}{\partial x_i}$

$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\boxed{\frac{\partial f}{\partial W_{i,j}}}, \frac{\partial f}{\partial x_i}$

$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\boxed{\frac{\partial f}{\partial W_{i,j}}}, \frac{\partial f}{\partial x_i}$

$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=i} x_j$$

$$\frac{\partial f}{\partial W_{i,j}} = \Sigma_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \Sigma_k (2q_k)(1_{k=i} x_j) = 2q_i x_j$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\frac{\partial f}{\partial W_{i,j}}, \boxed{\frac{\partial f}{\partial x_i}}$

$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

We want: $\frac{\partial f}{\partial W_{i,j}}, \boxed{\frac{\partial f}{\partial x_i}}$

$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

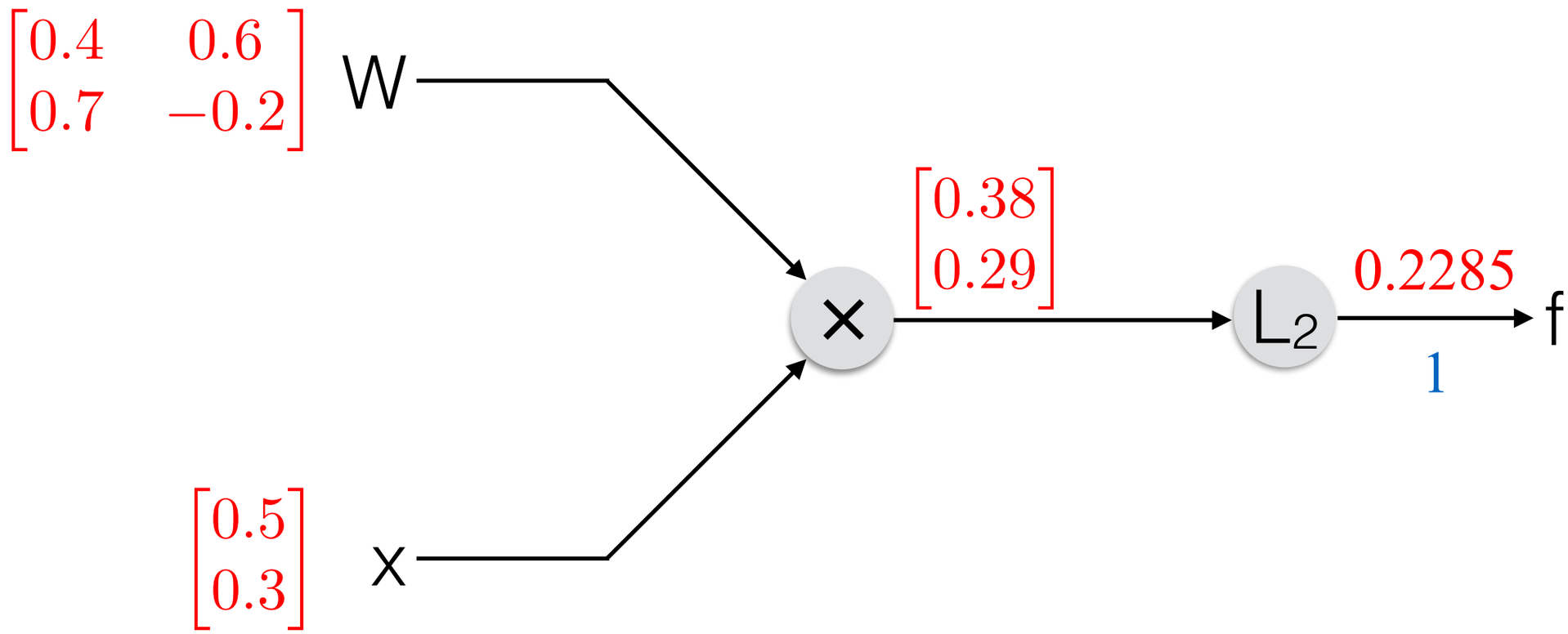
$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \Sigma_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \Sigma_k 2 q_k W_{k,i}$$

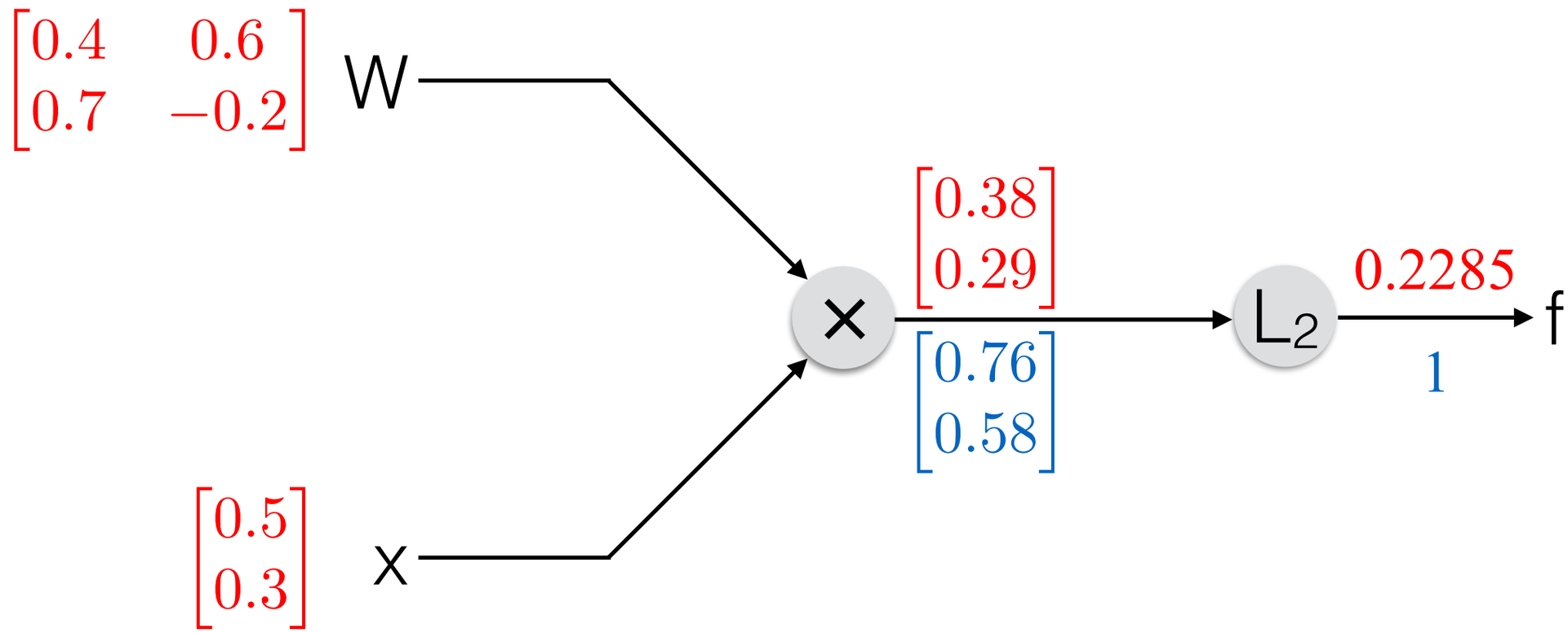
# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n}(W \cdot x)_i^2$

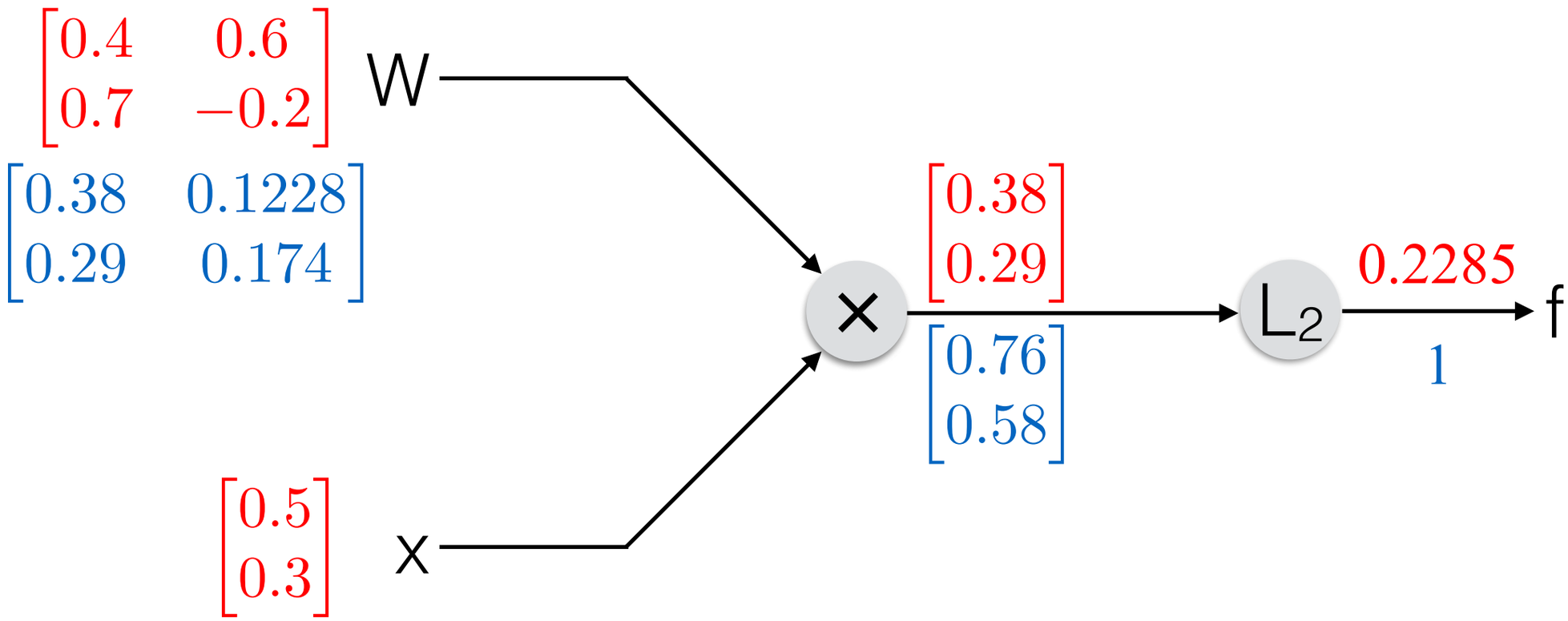



$$q = W \cdot x = \begin{pmatrix} W_{1,1x1} + ... + W_{1,nxn} \\ ... \\ W_{n,1x1} + ... + W_{n,nxn} \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + ... + q_n^2$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$

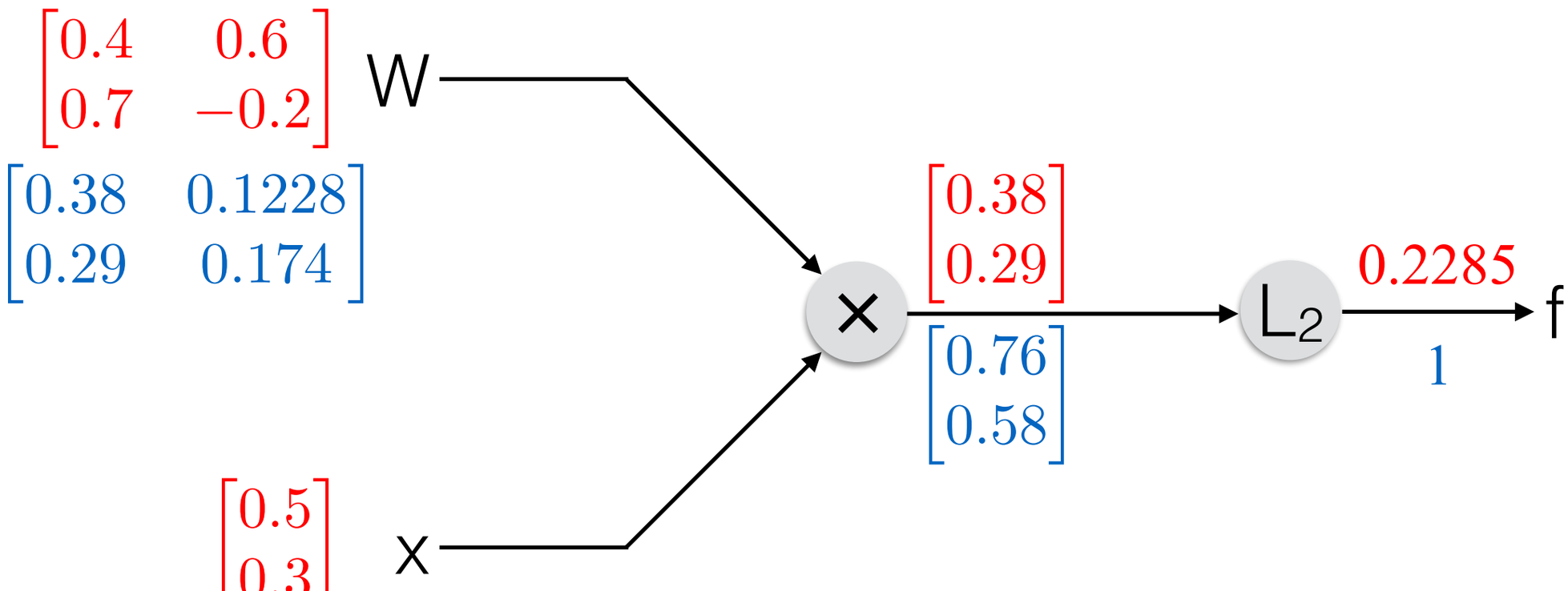



$$\frac{\partial q_k}{\partial W_{i,j}} = 1_{k=i} x_j$$

$$\frac{\partial f}{\partial W_{i,j}} = \Sigma_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \Sigma_k (2q_k)(1_{k=i} x_j) = 2q_i x_j$$

# Computational Graph

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \Sigma_{i=1}^{n} (W \cdot x)_i^2$



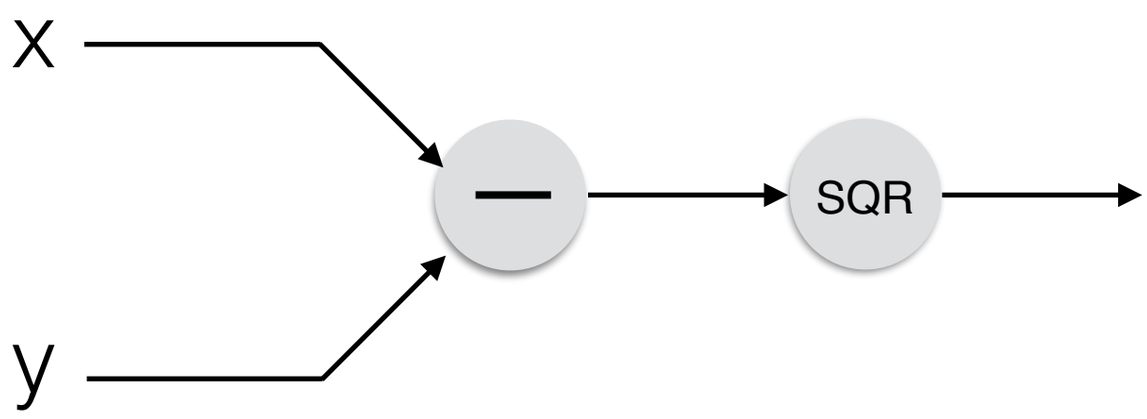

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \Sigma_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \Sigma_k 2 q_k W_{k,i}$$

# Layers with Computational Graph

**L2-loss**

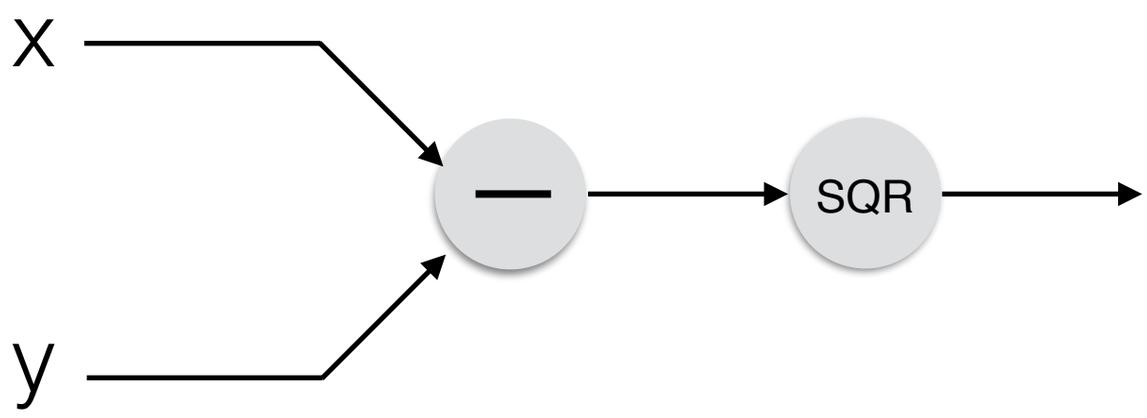
$$L_2(x, y) = (x - y)^2$$



# Layers with Computational Graph

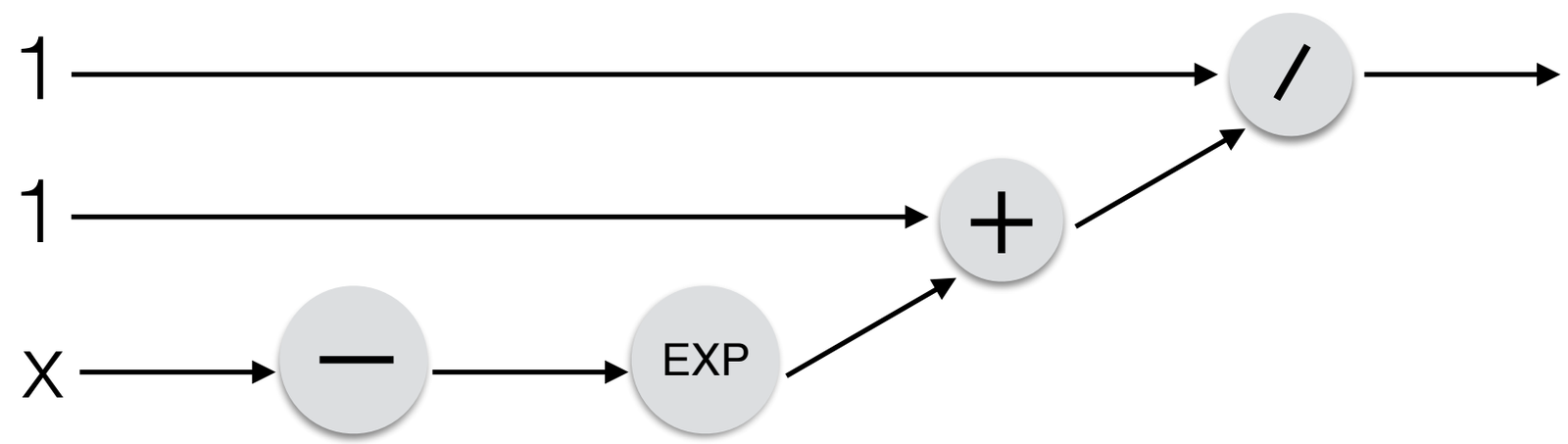
**L2-loss**

$$L_2(x, y) = (x - y)^2$$




**Sigmoid**

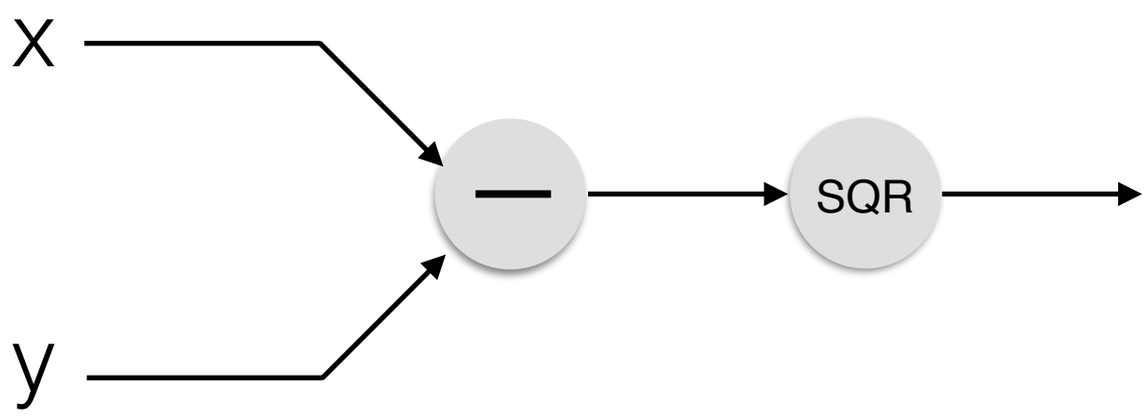
$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



# Layers with Computational Graph

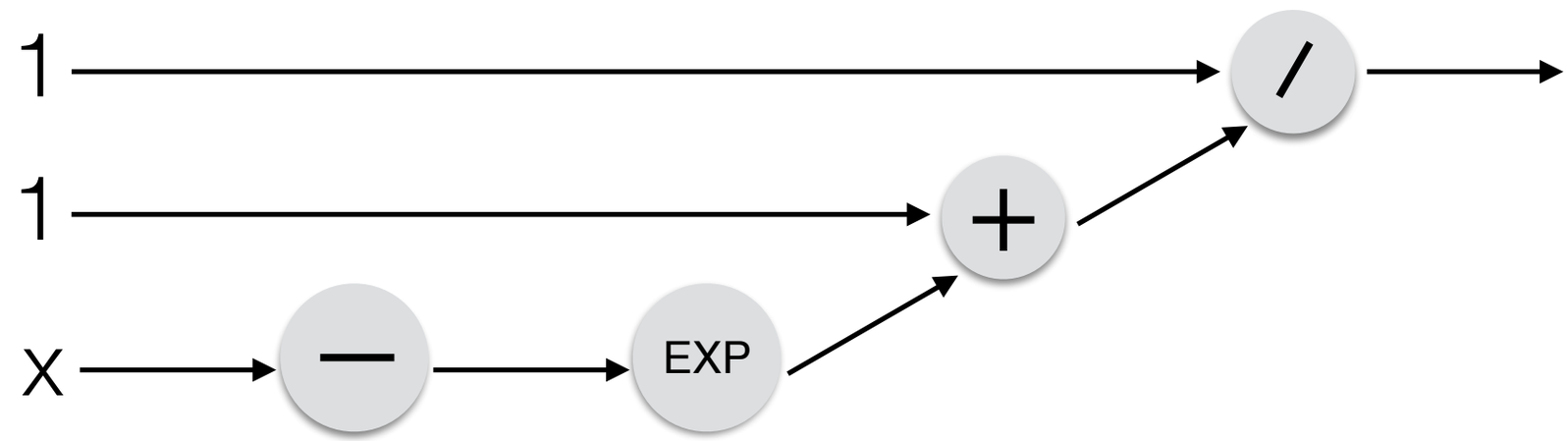
**L2-loss**

$$L_2(x, y) = (x - y)^2$$



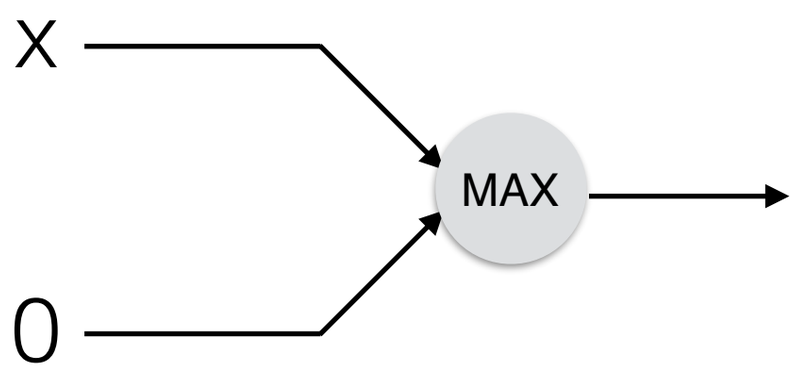

**Sigmoid**

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

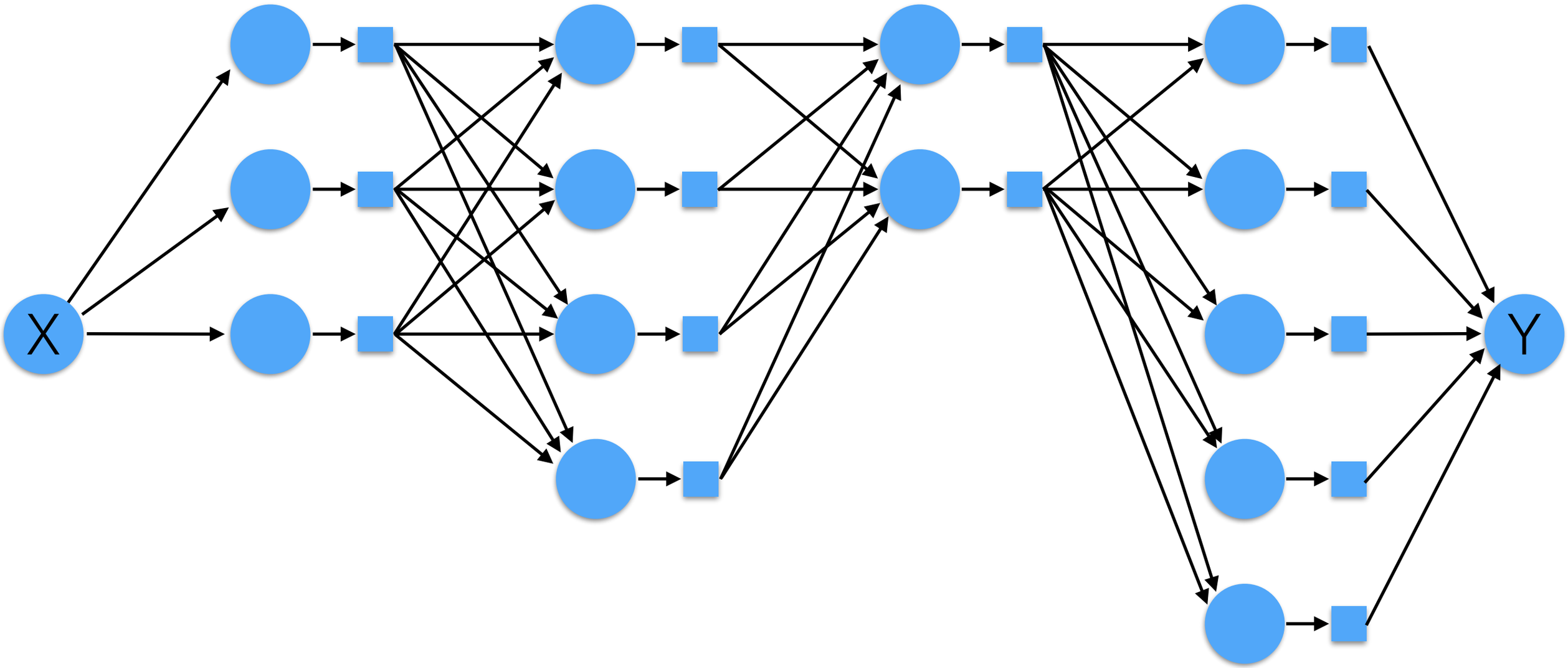



**ReLU**

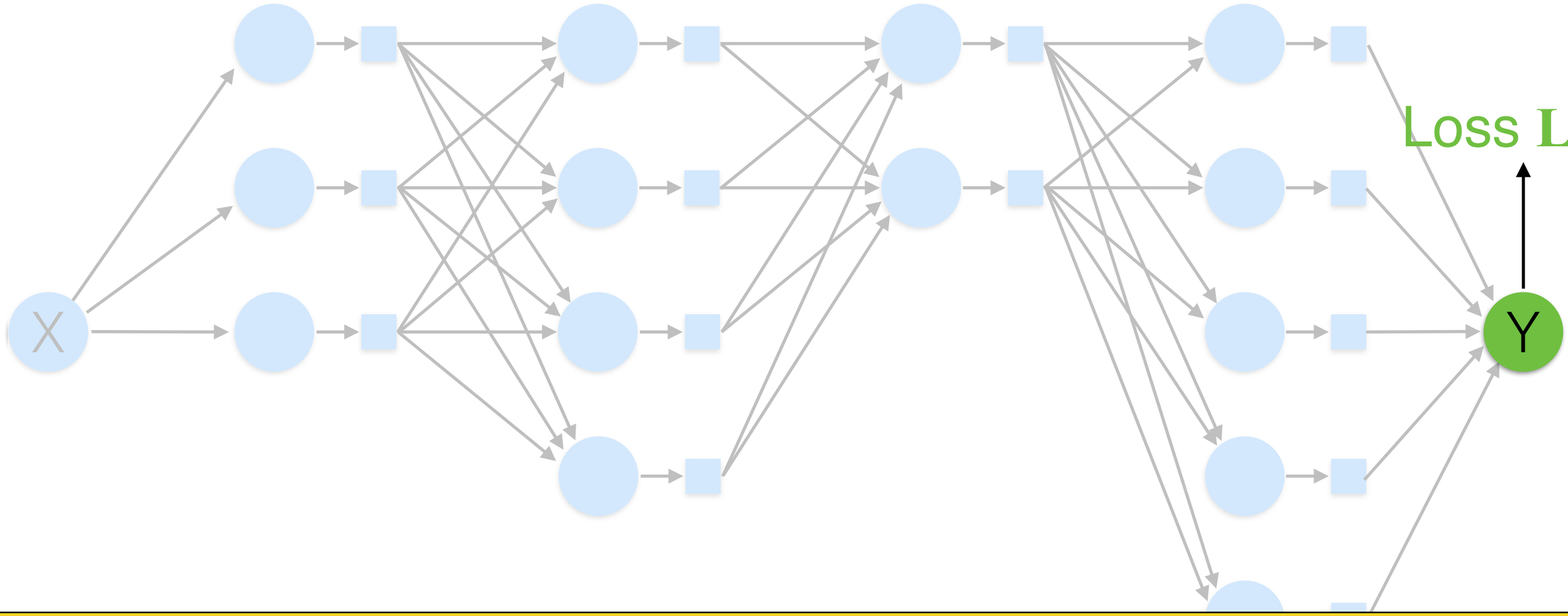
$$\text{ReLU}(x) = \max(x, 0)$$



# Neural Networks Example



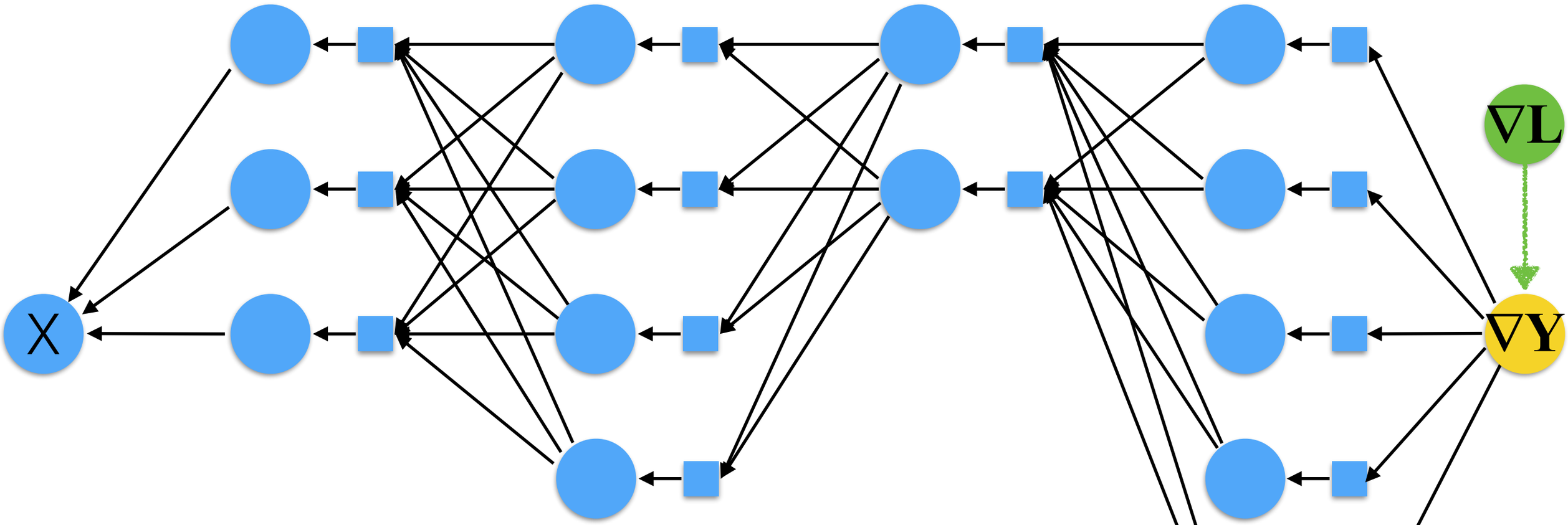
# Backward propagation




Compute a loss using an estimated $\mathbf{Y}$ and the ground truth $\mathbf{Y}^*$.

e.g., $\mathbf{L(W)} = \| \mathbf{f(X; W)} - \mathbf{Y}^* \|_2$

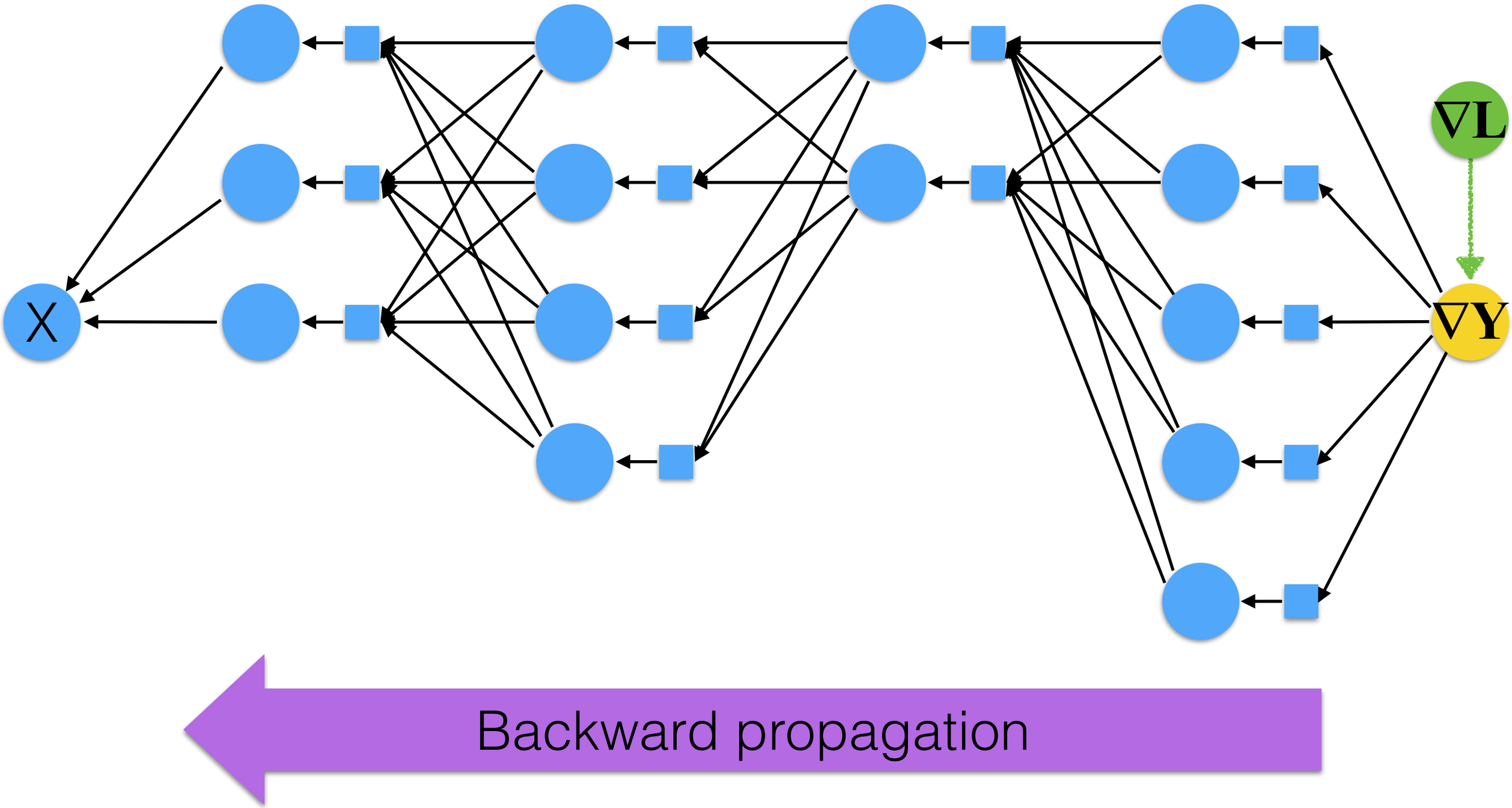
# Backward propagation




Compute gradients of $\mathbf{Y}$ using $\nabla\mathbf{L}$.

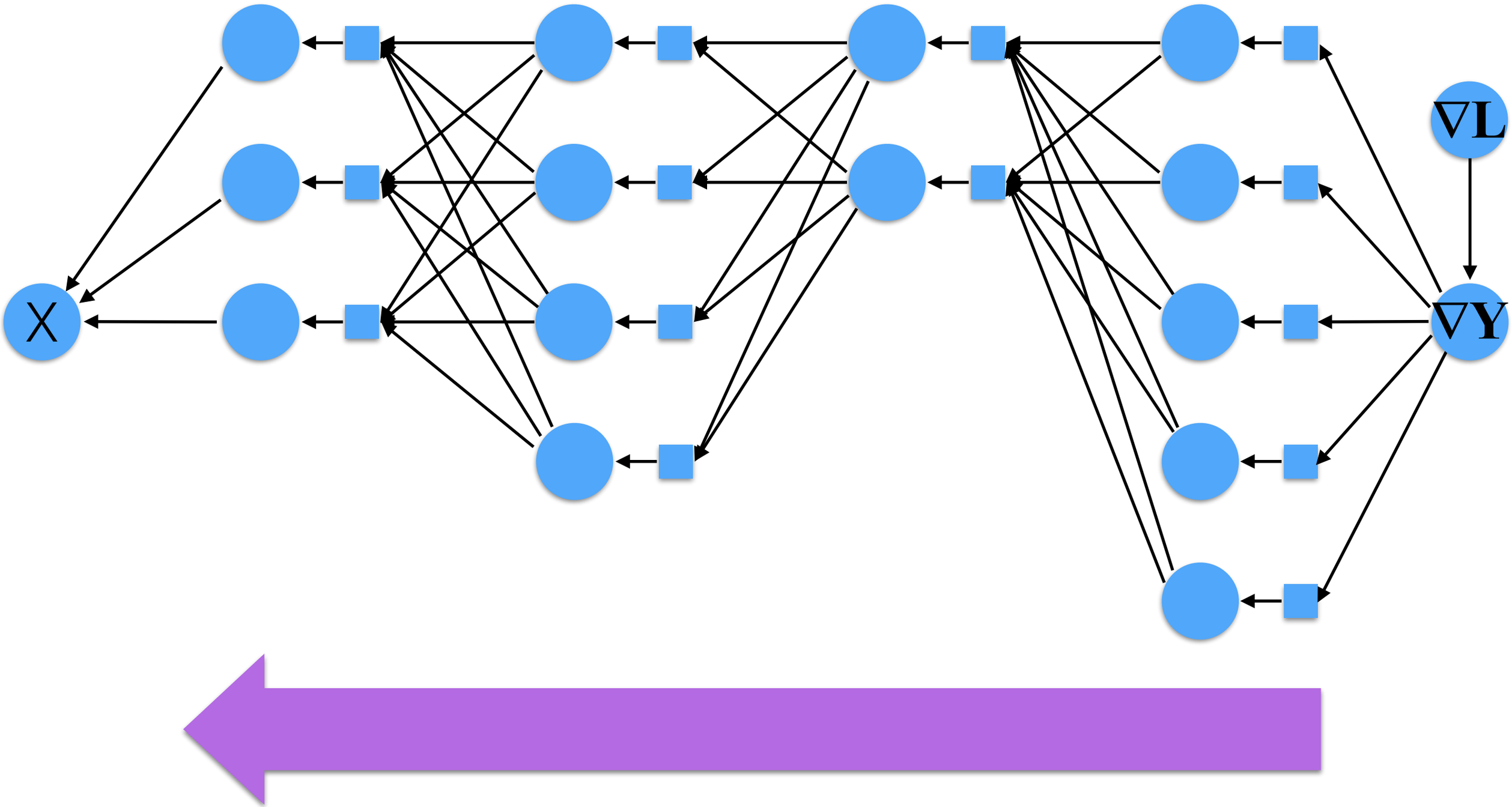
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X}$$

# Backward propagation

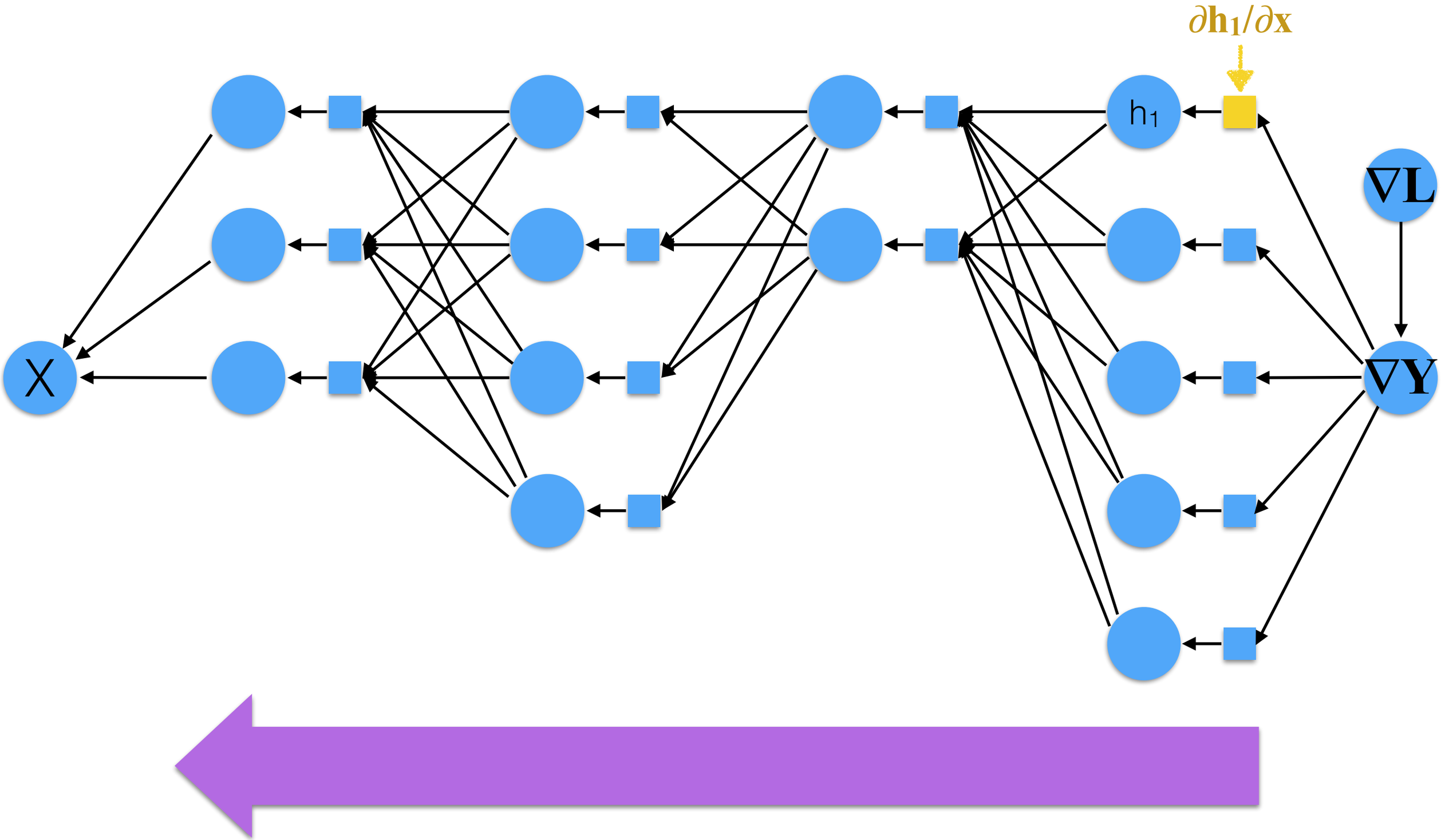



Backward propagation

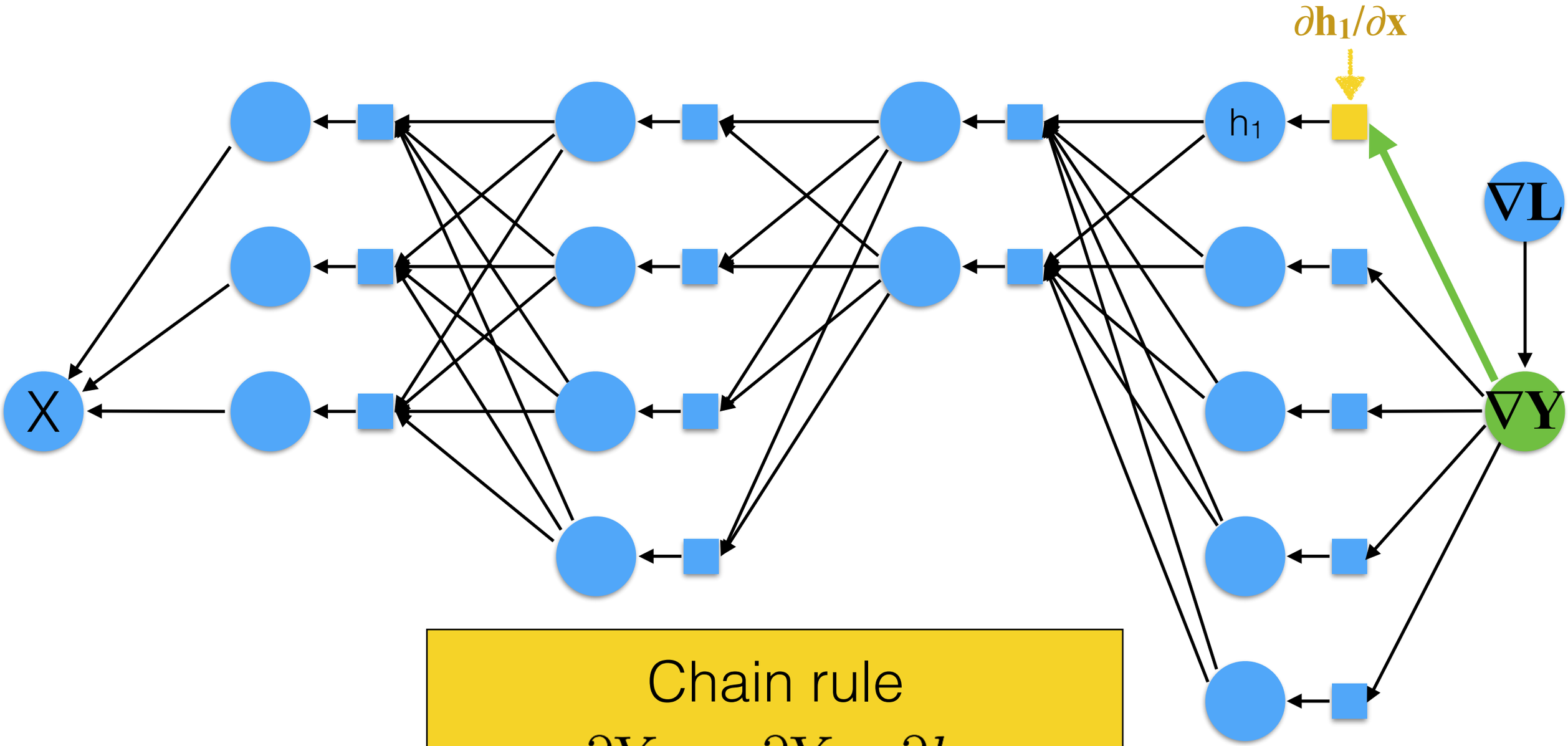
# Backward propagation



# Backward propagation



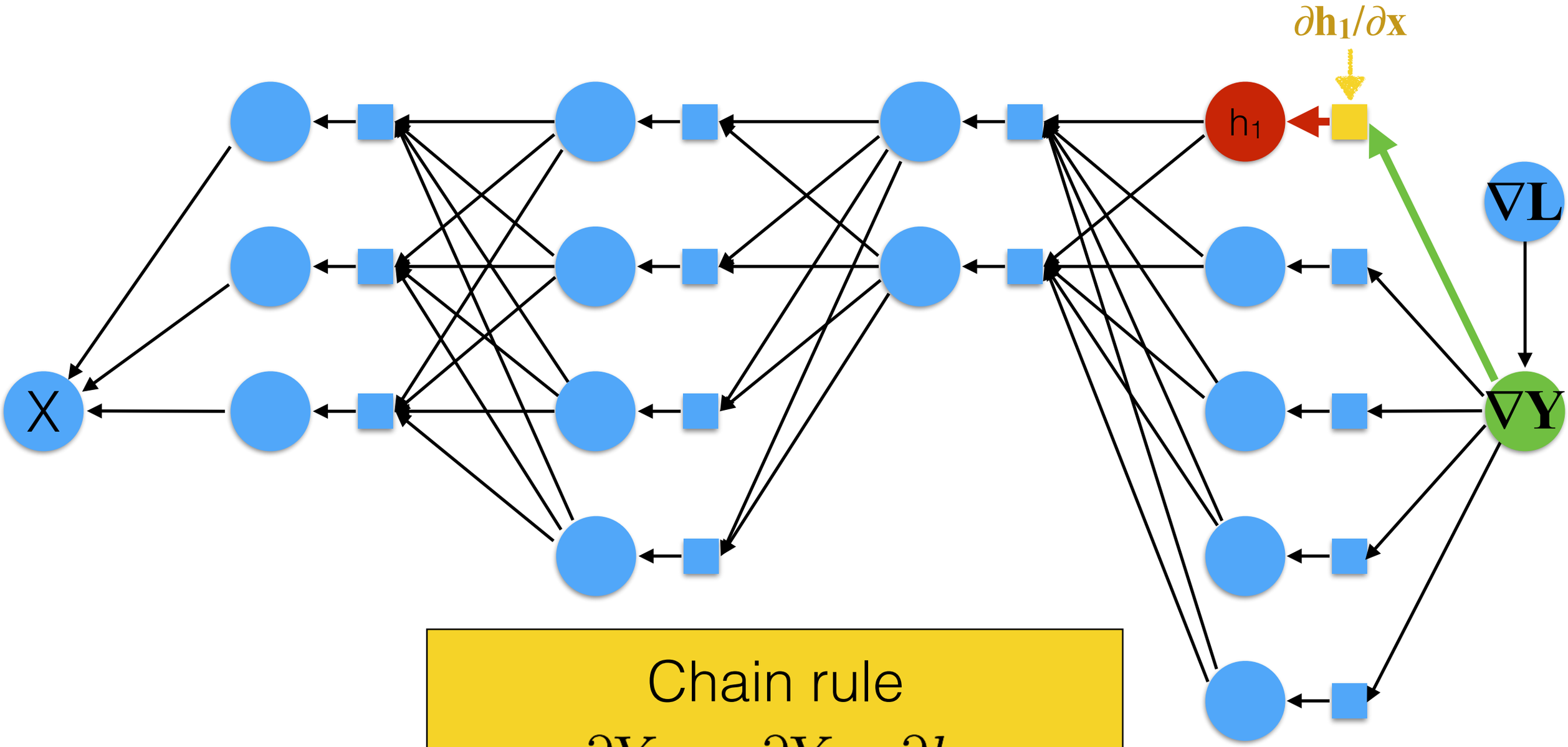
# Backward propagation




Chain rule

$$\frac{\partial Y}{\partial X} = \frac{\partial Y}{\partial h_1} \cdot \frac{\partial h_1}{\partial X}$$

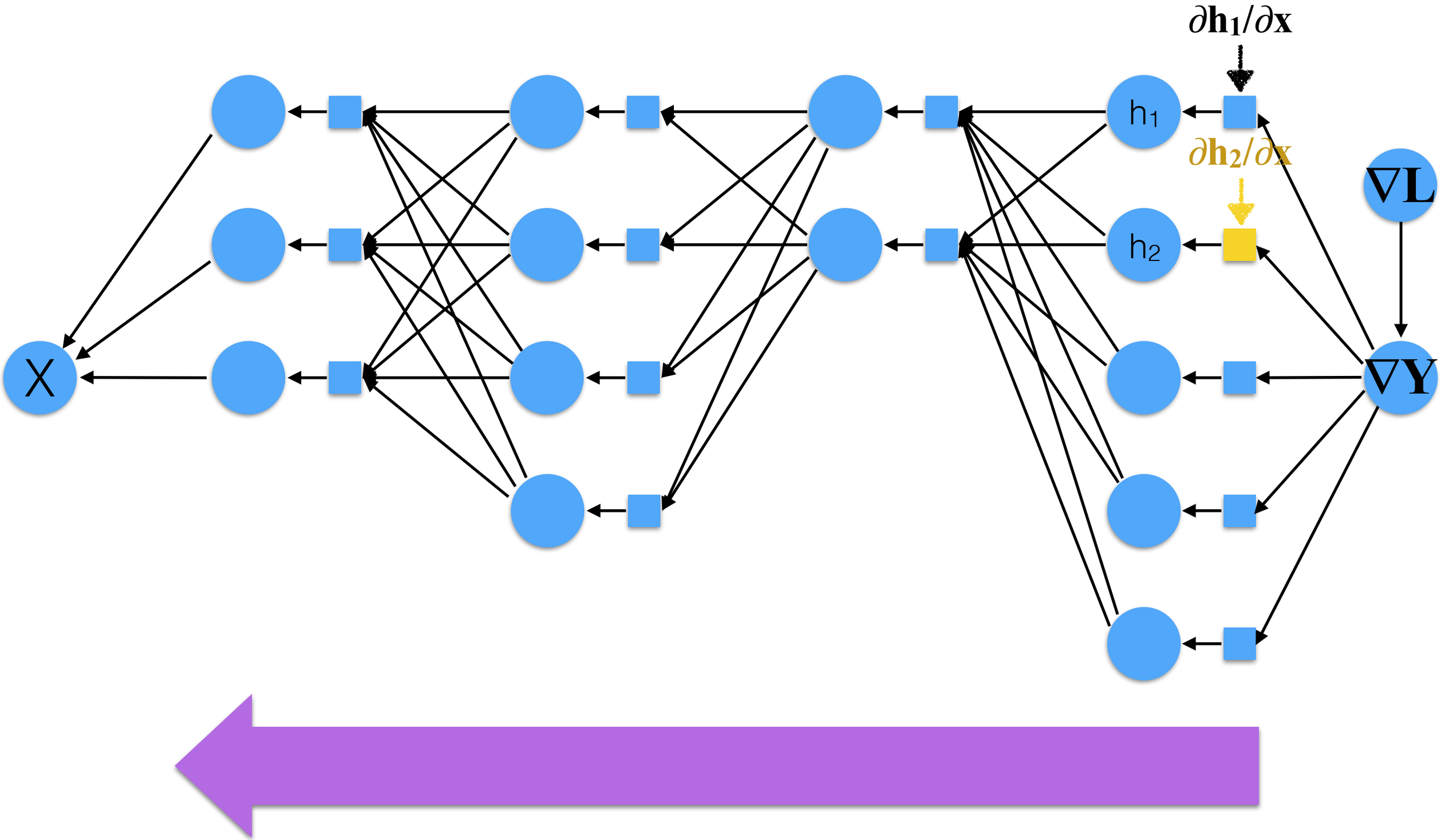
# Backward propagation



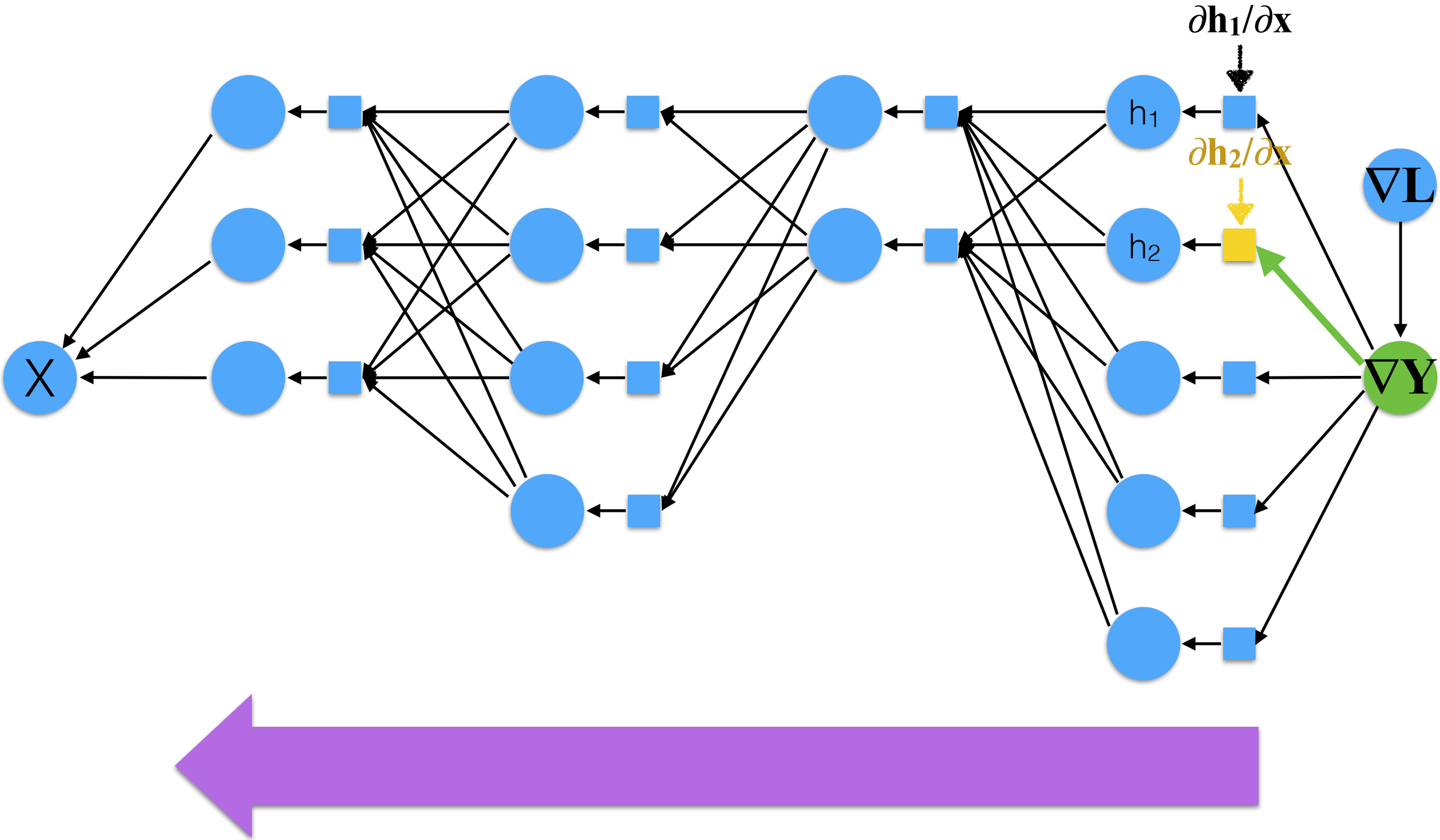

Chain rule

$$\frac{\partial Y}{\partial X} = \frac{\partial Y}{\partial h_1} \cdot \frac{\partial h_1}{\partial X}$$

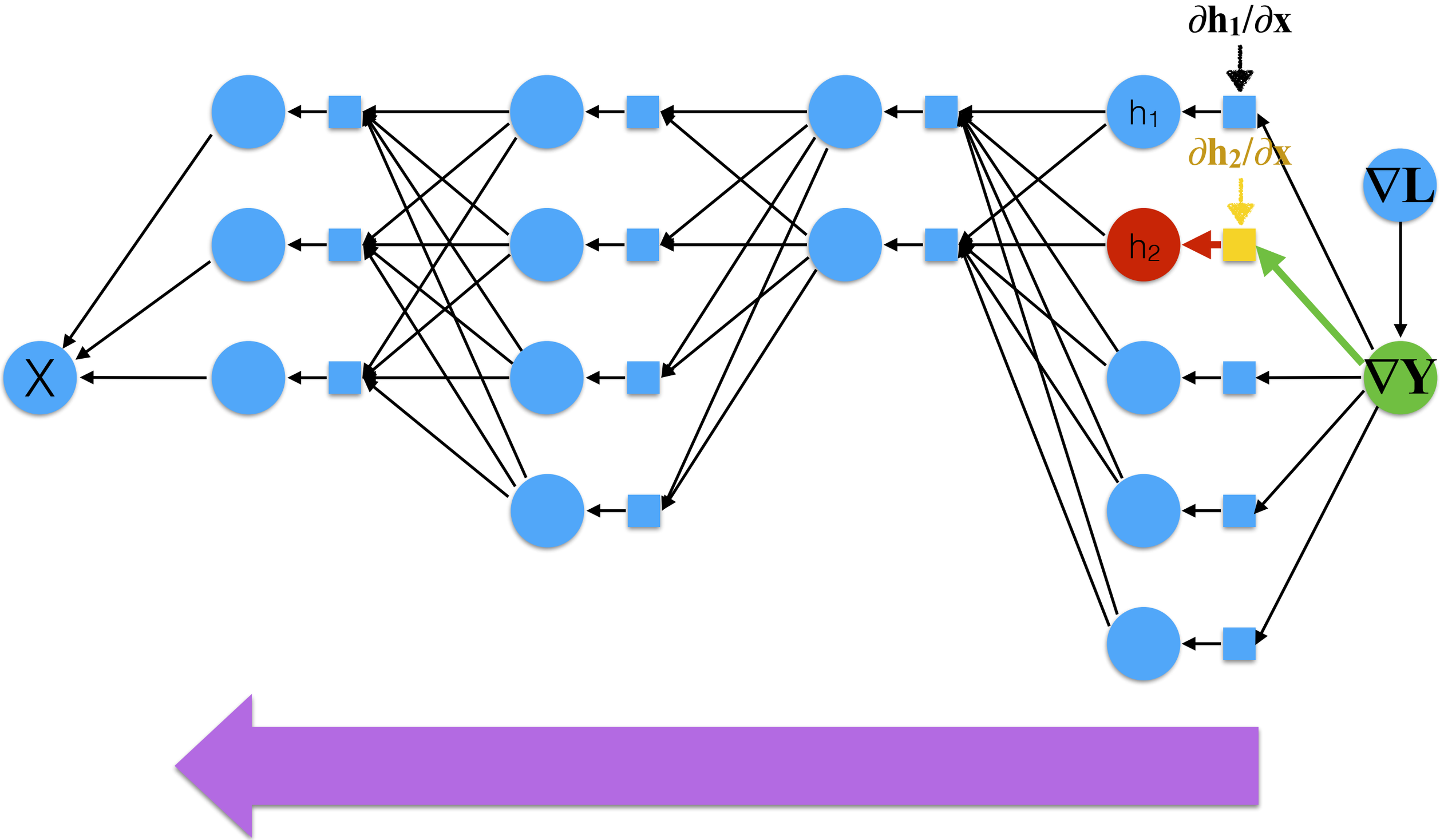
# Backward propagation



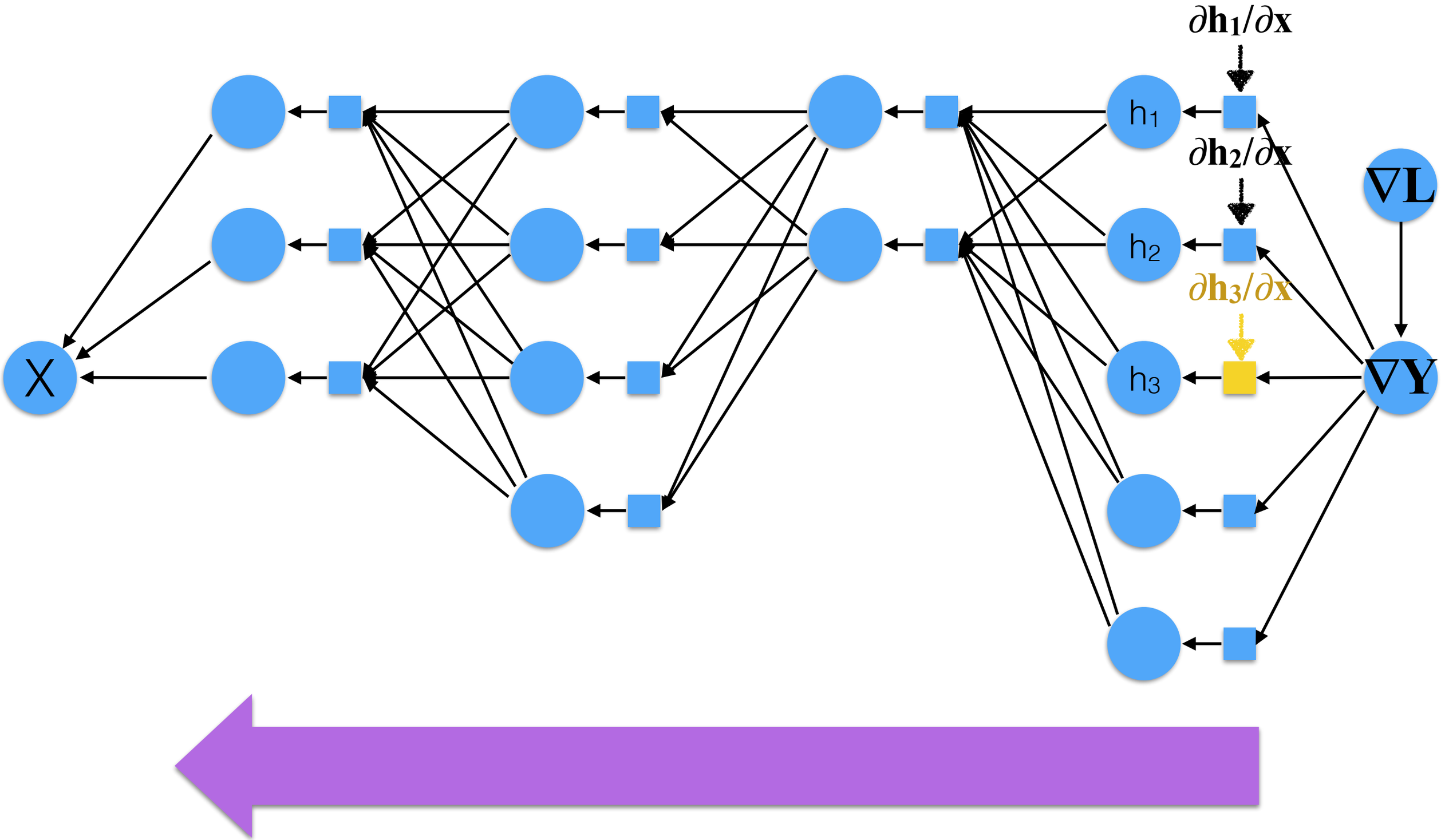
# Backward propagation



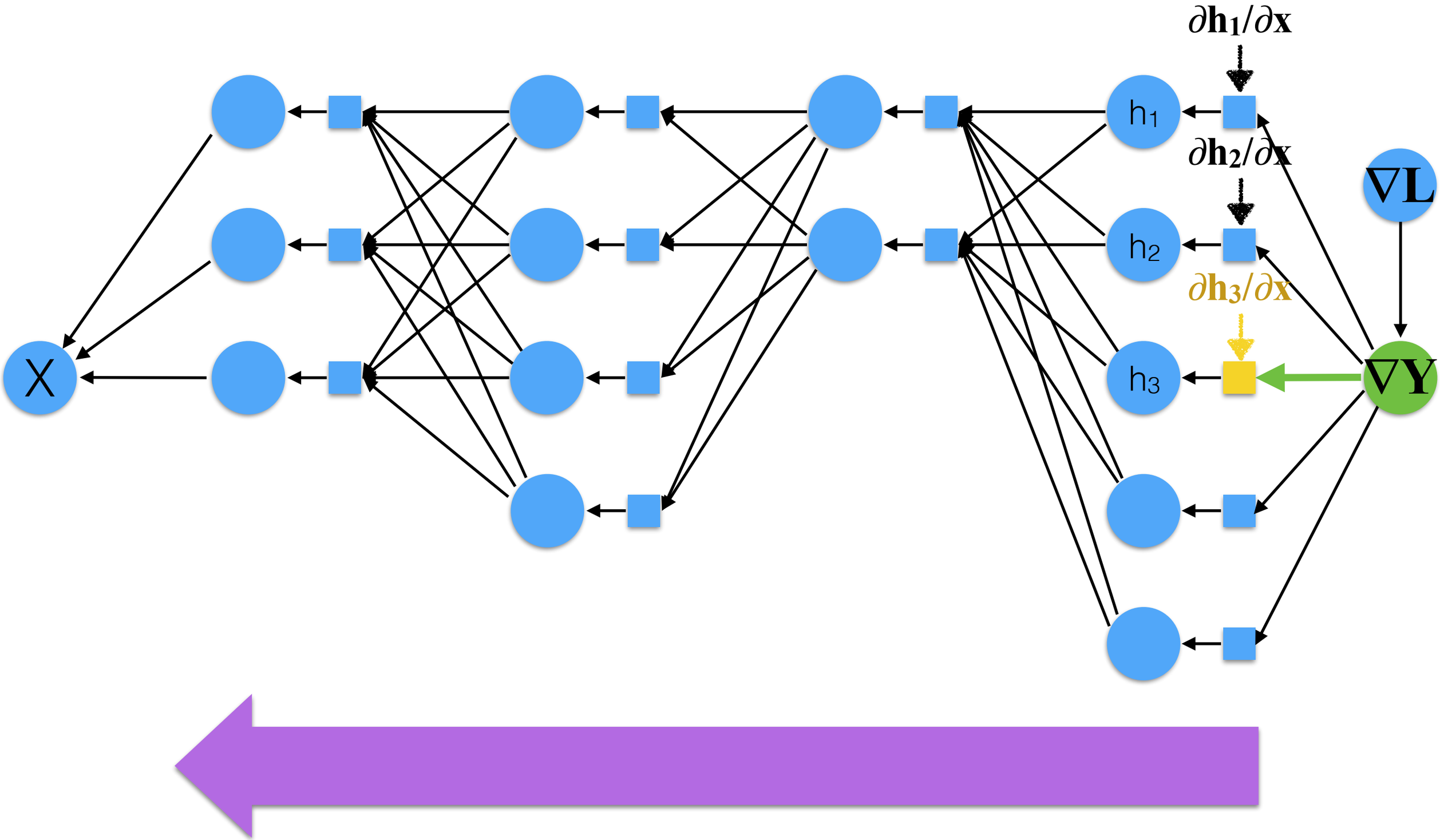
# Backward propagation



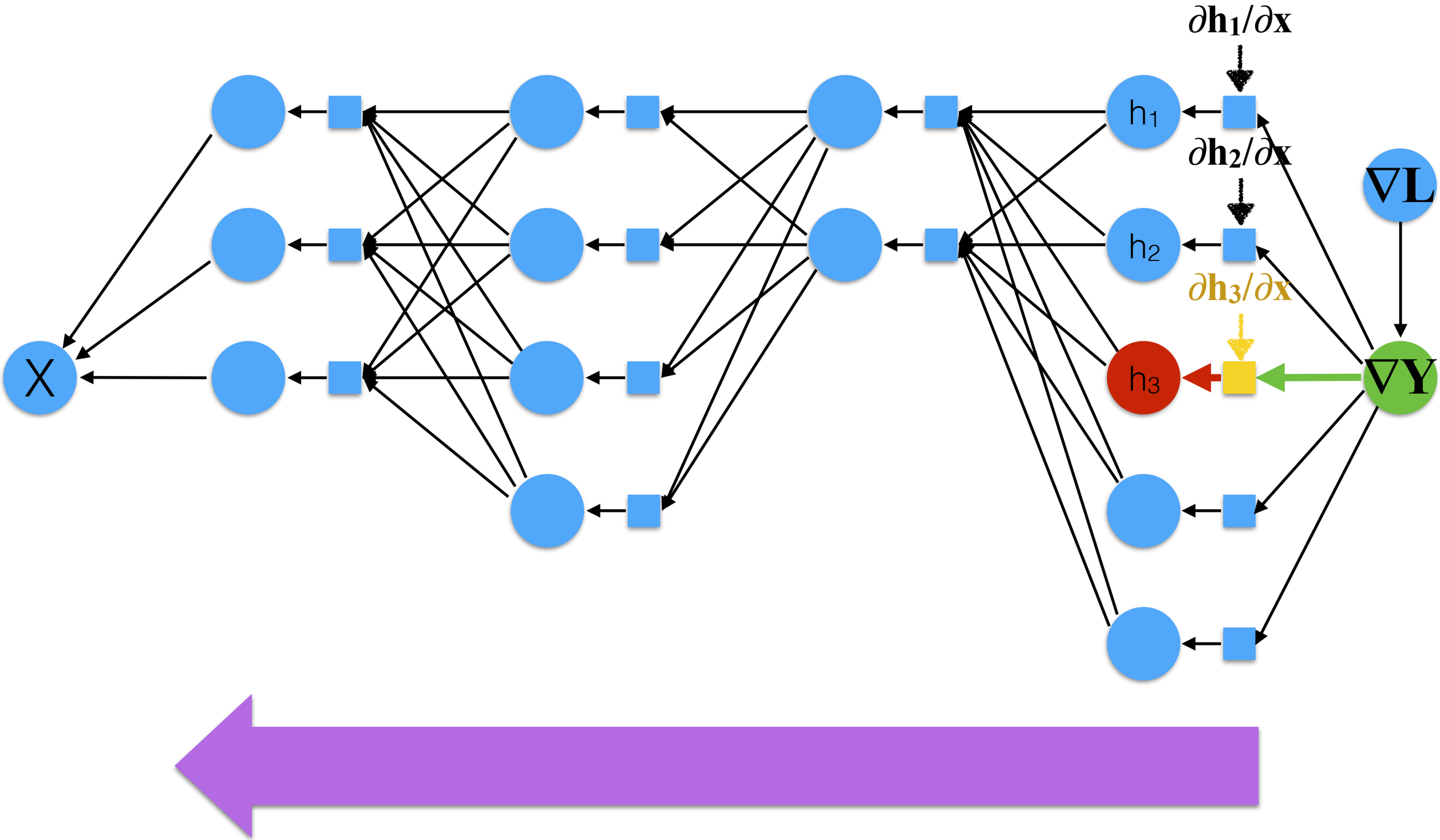
# Backward propagation



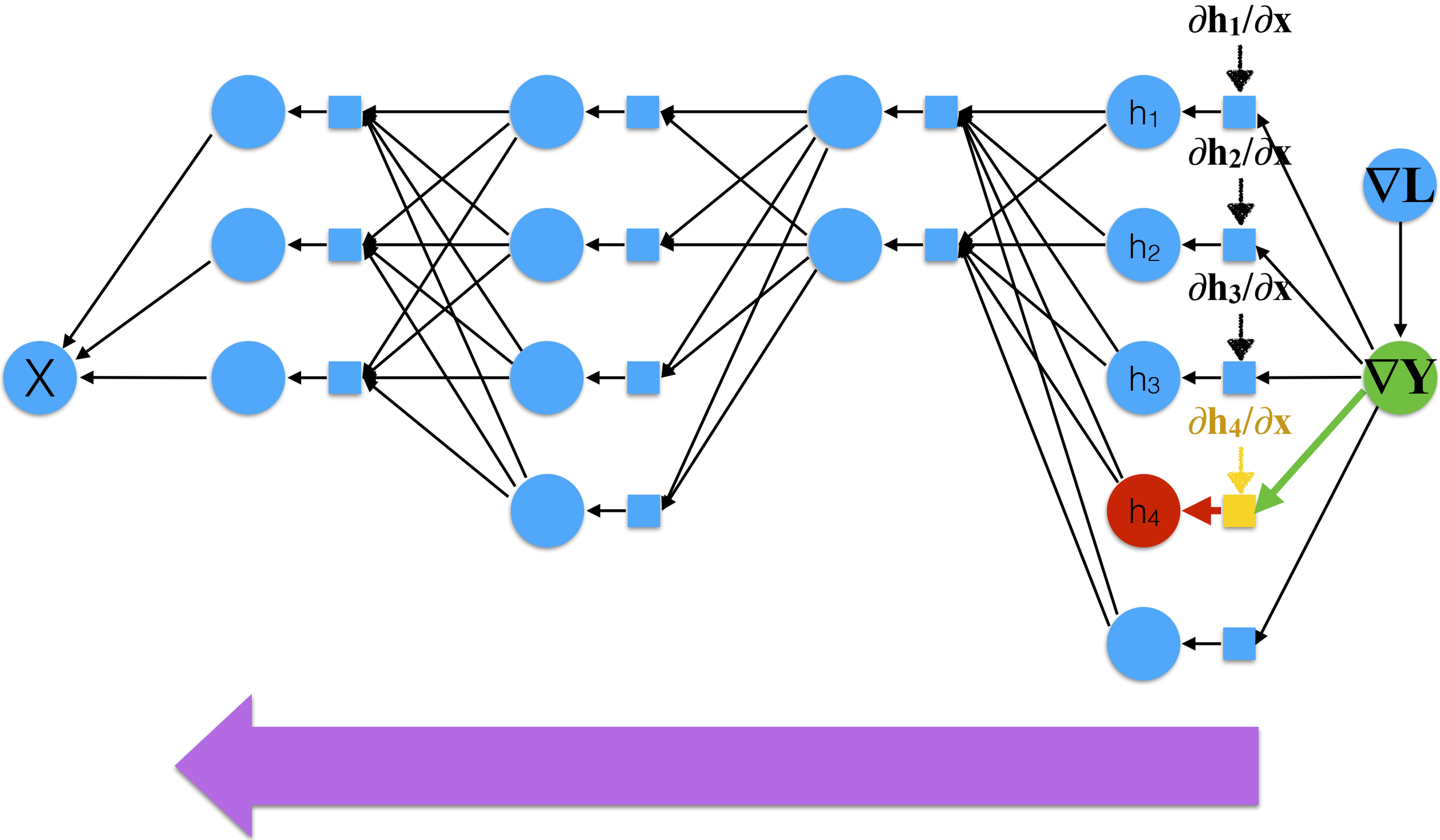
# Backward propagation



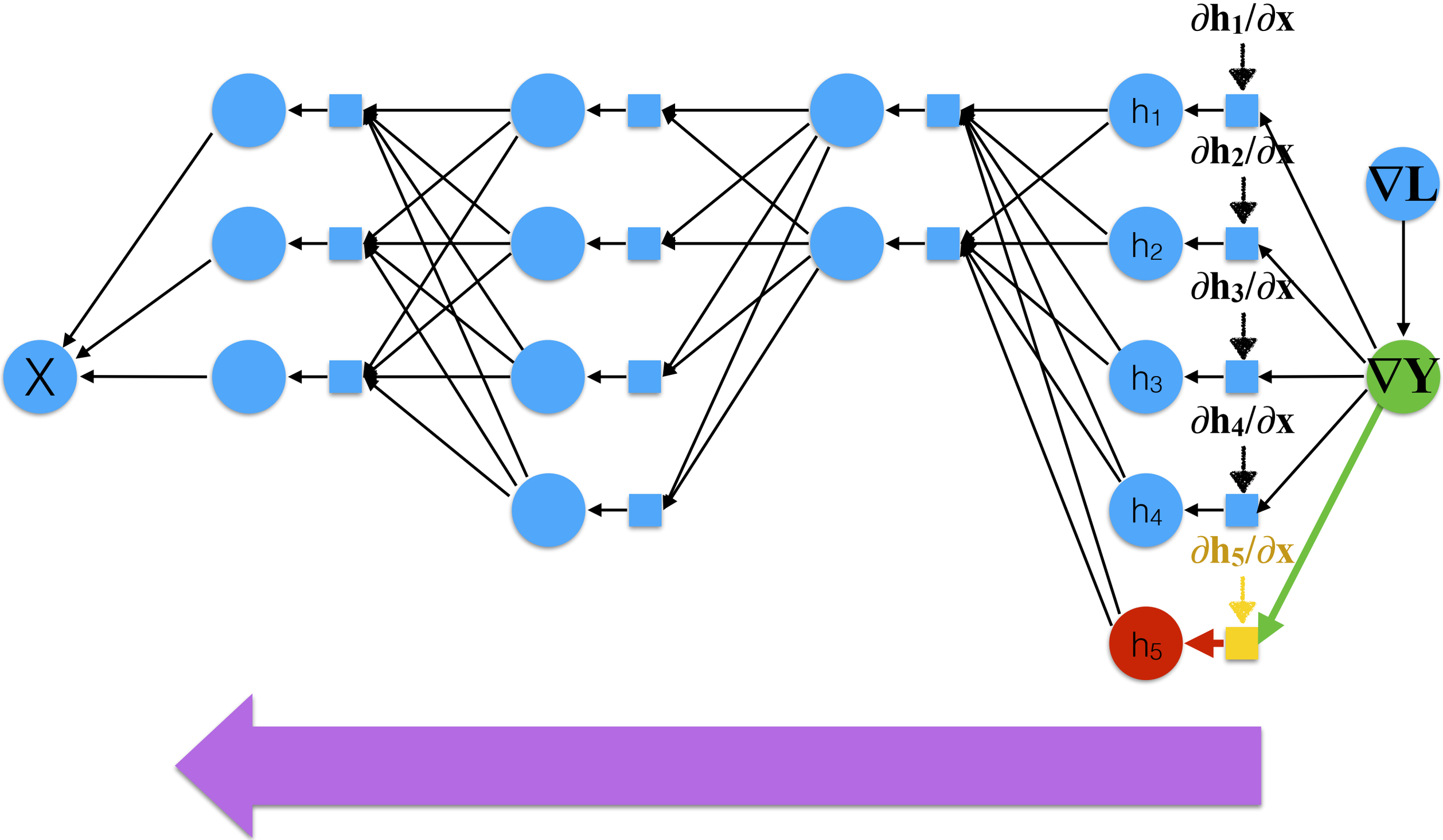
# Backward propagation



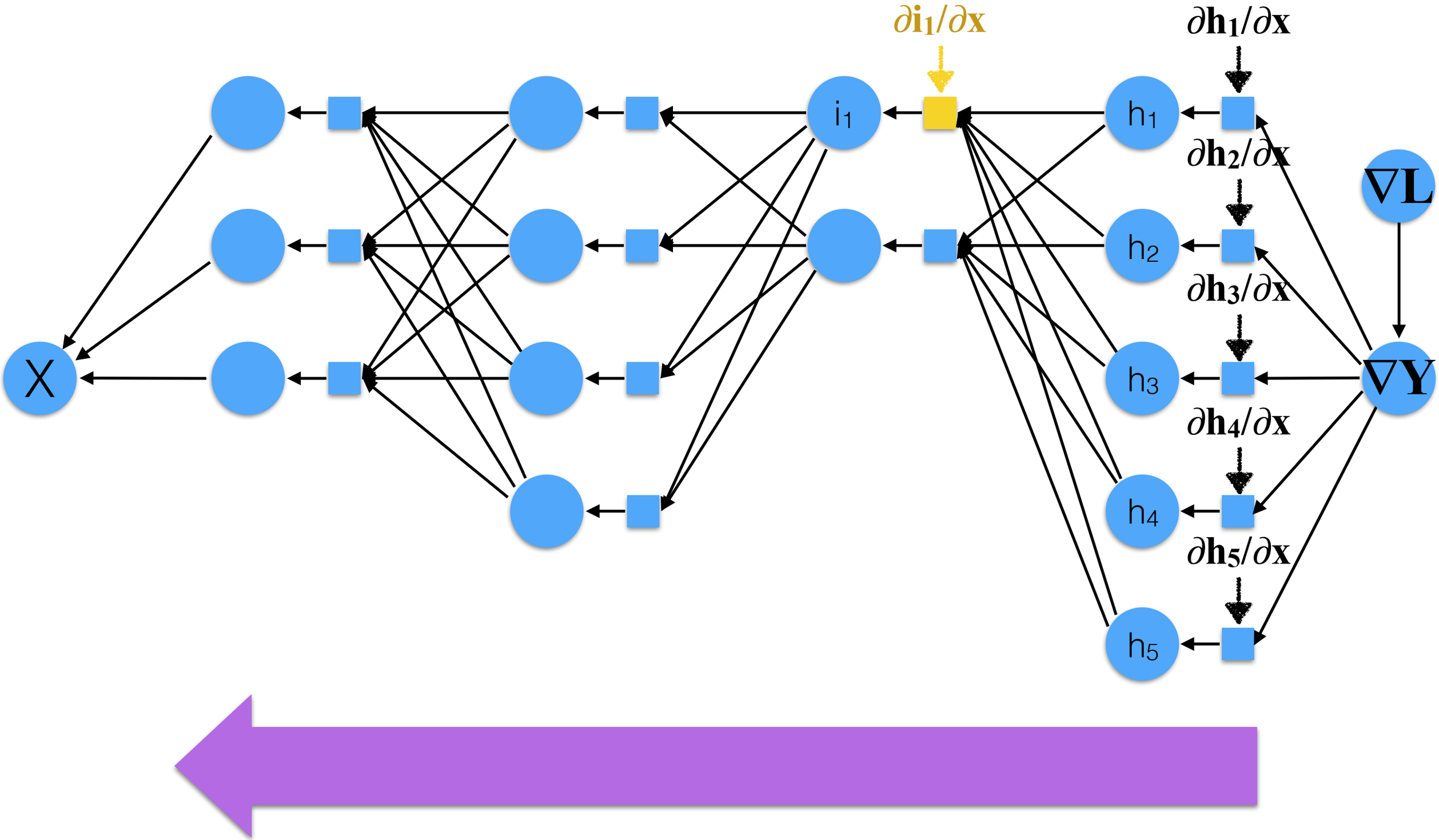
# Backward propagation



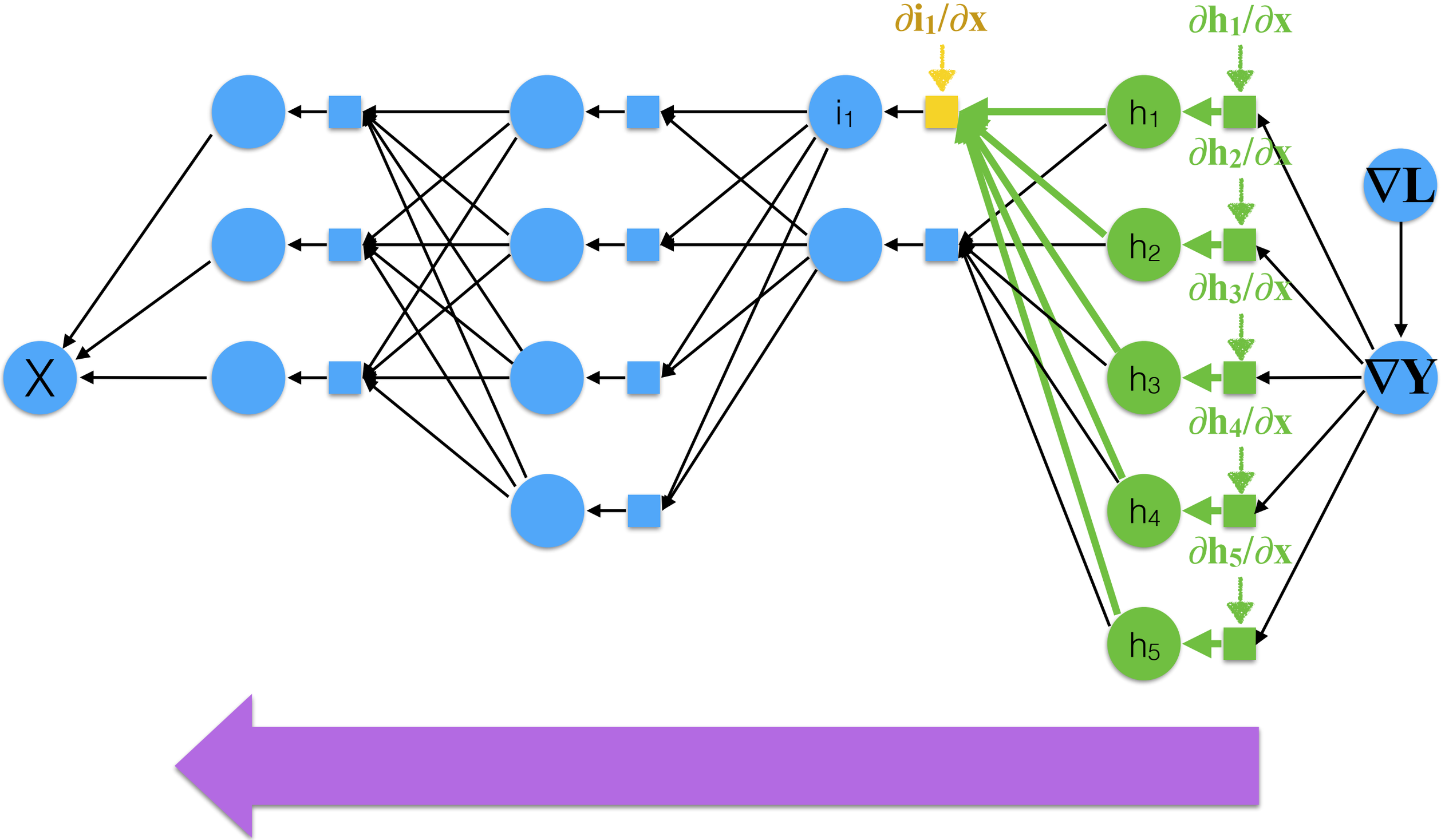
# Backward propagation



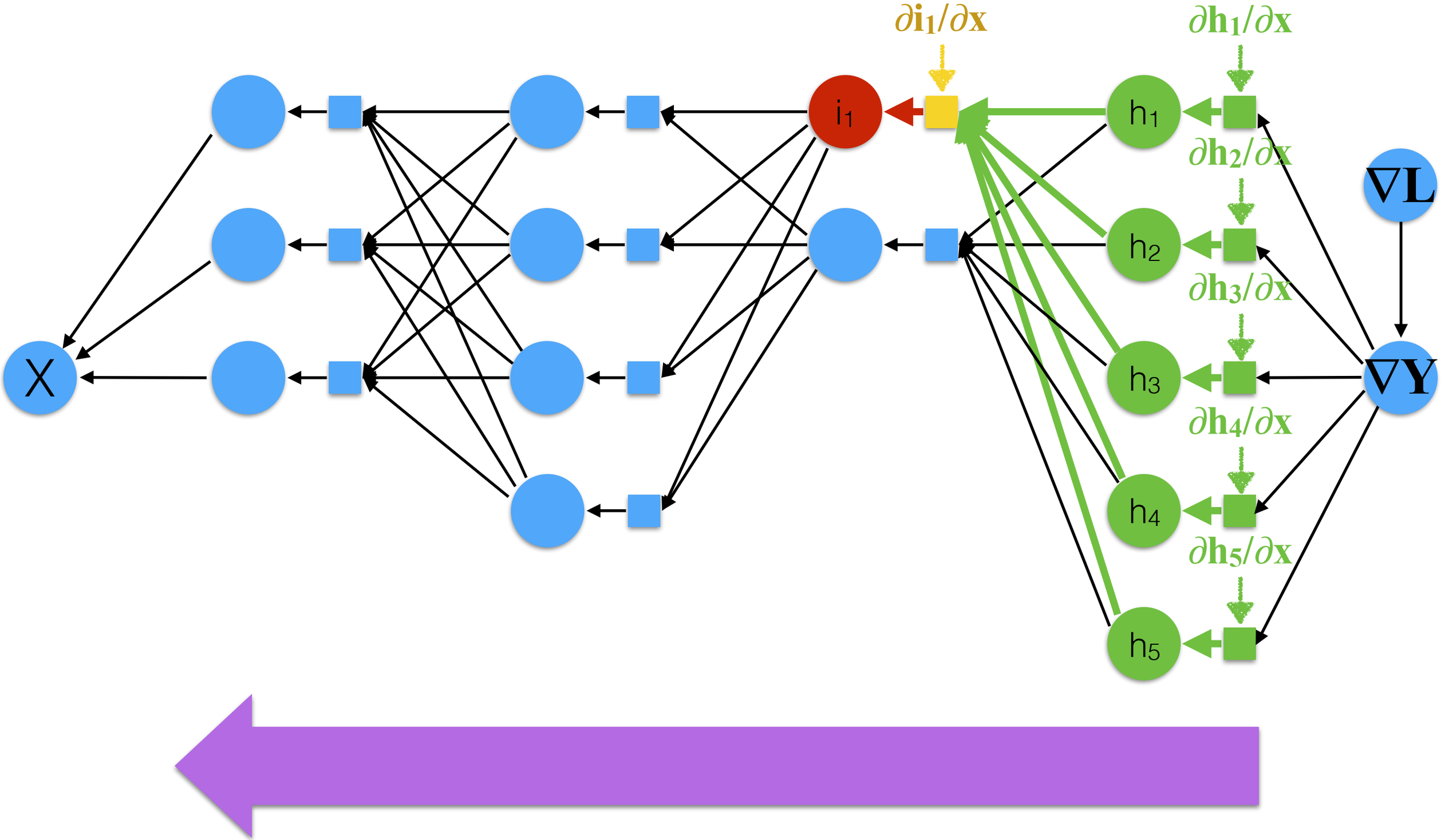
# Backward propagation



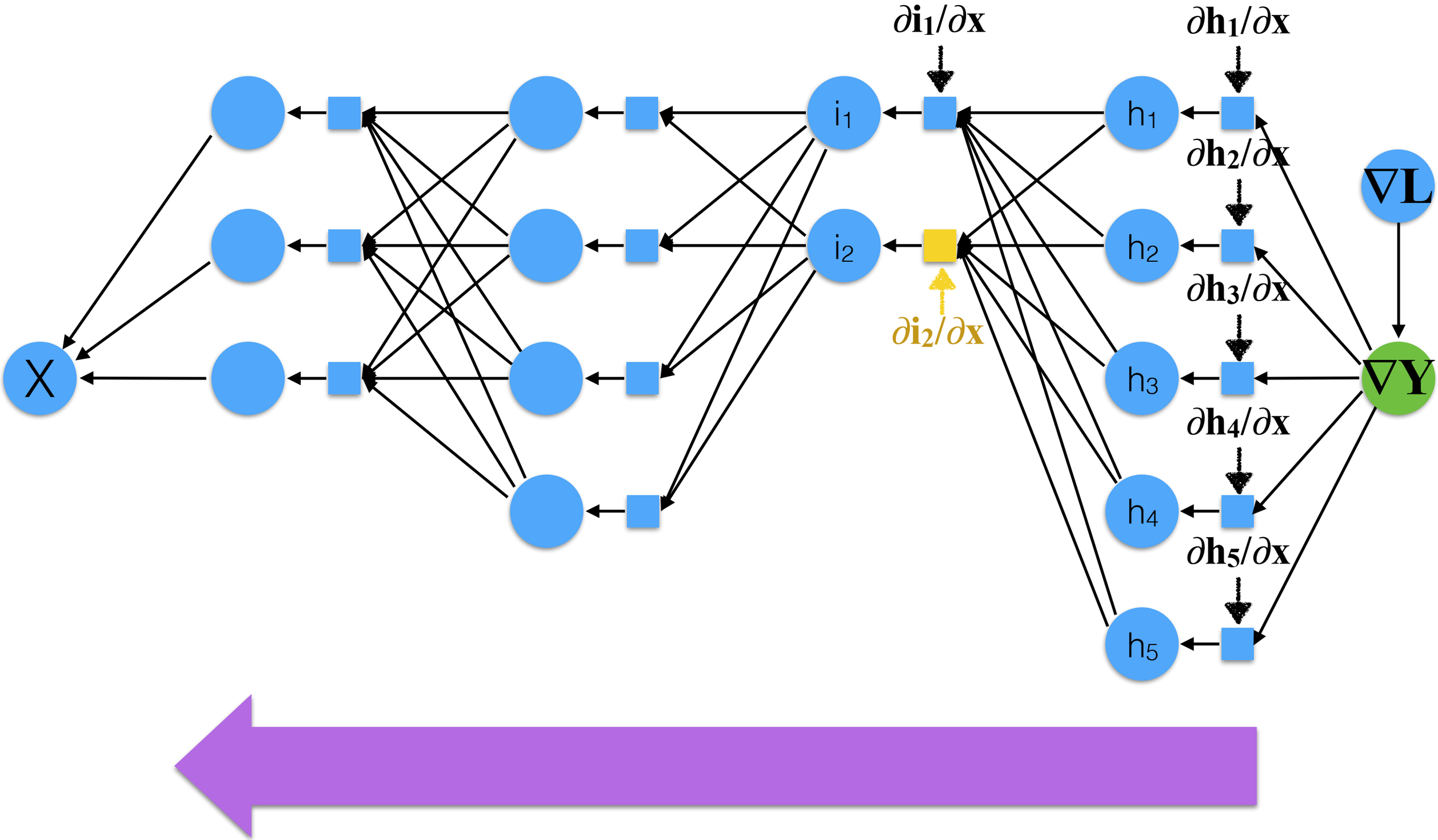
# Backward propagation



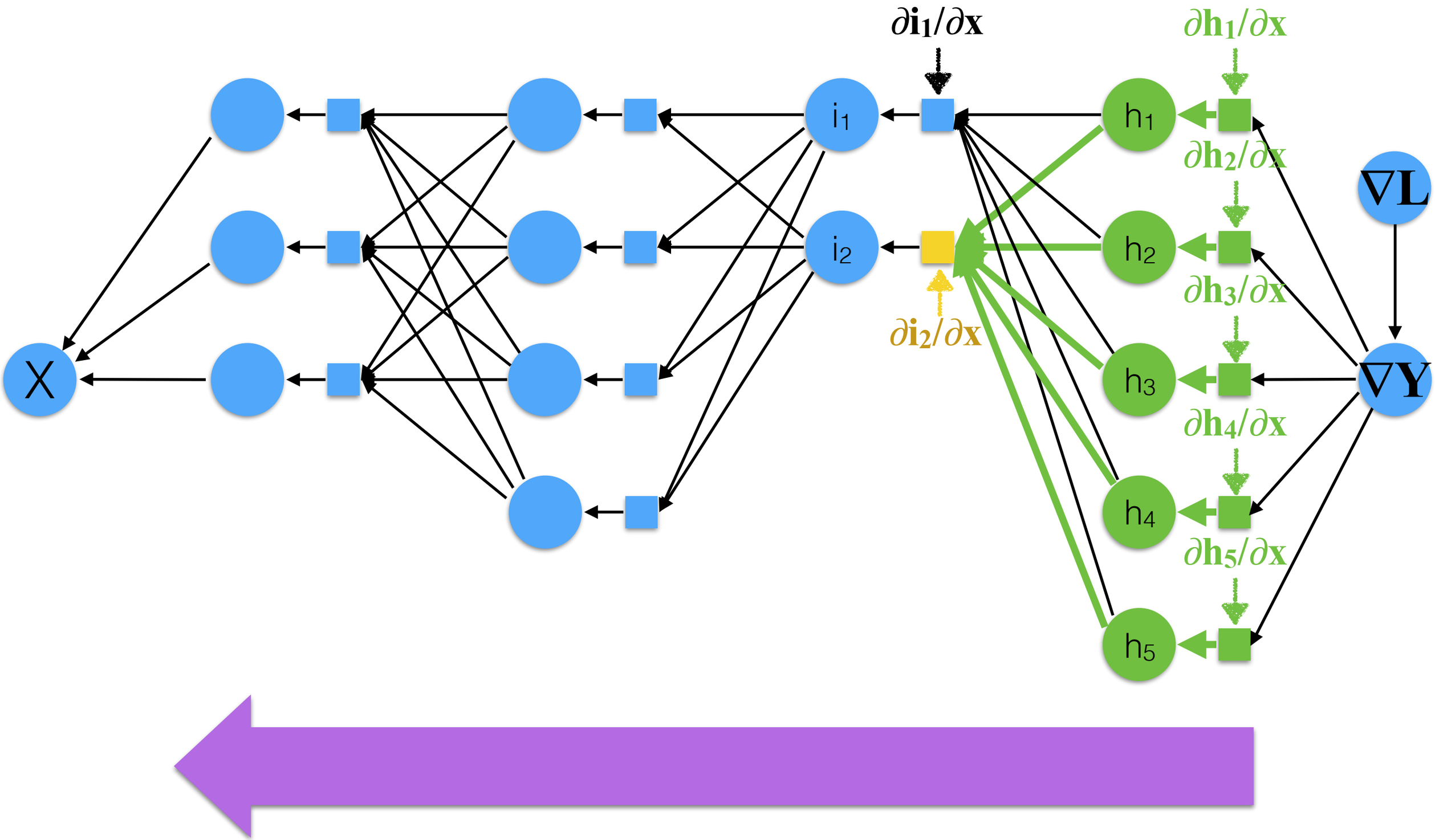
# Backward propagation



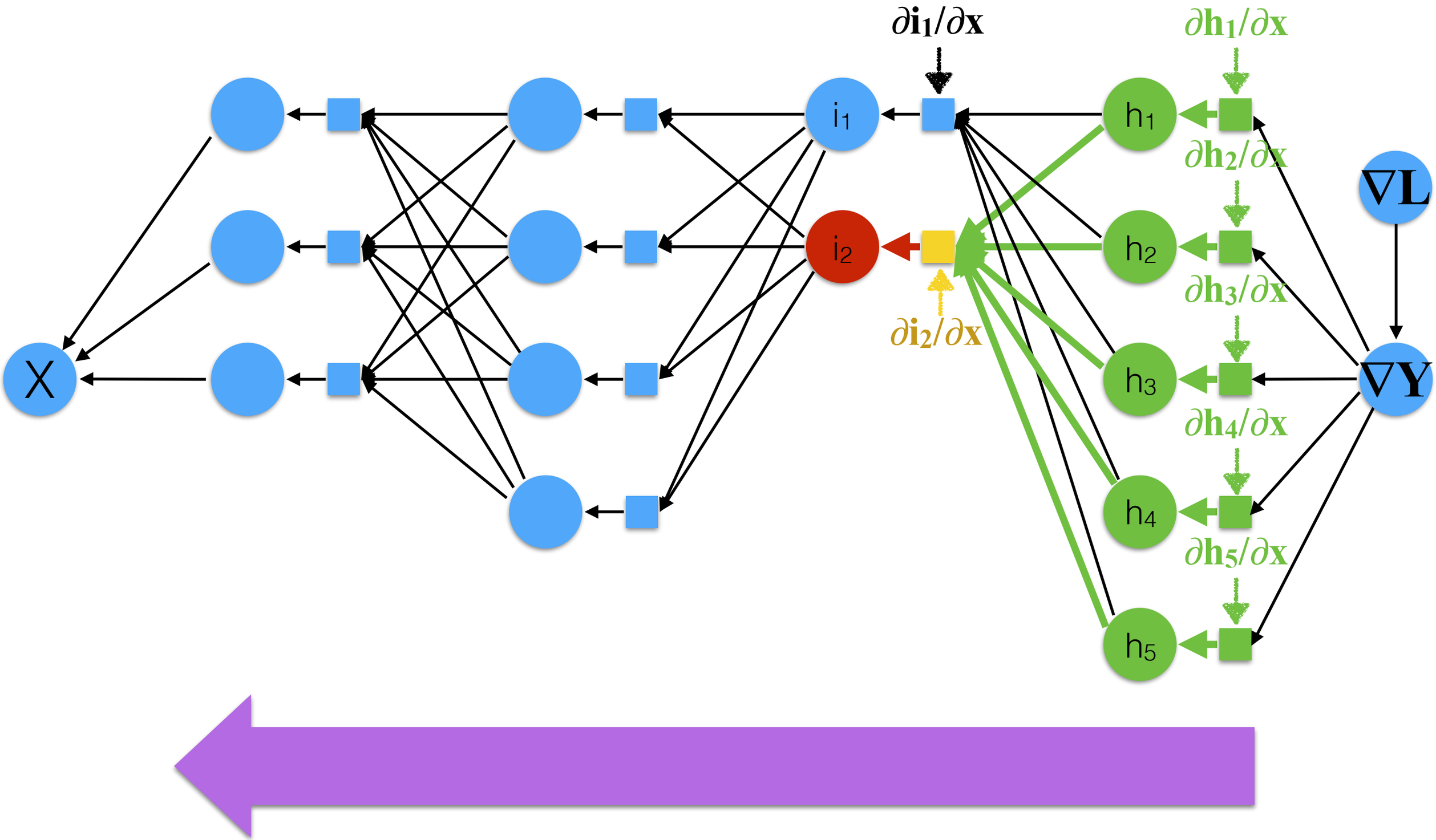
# Backward propagation



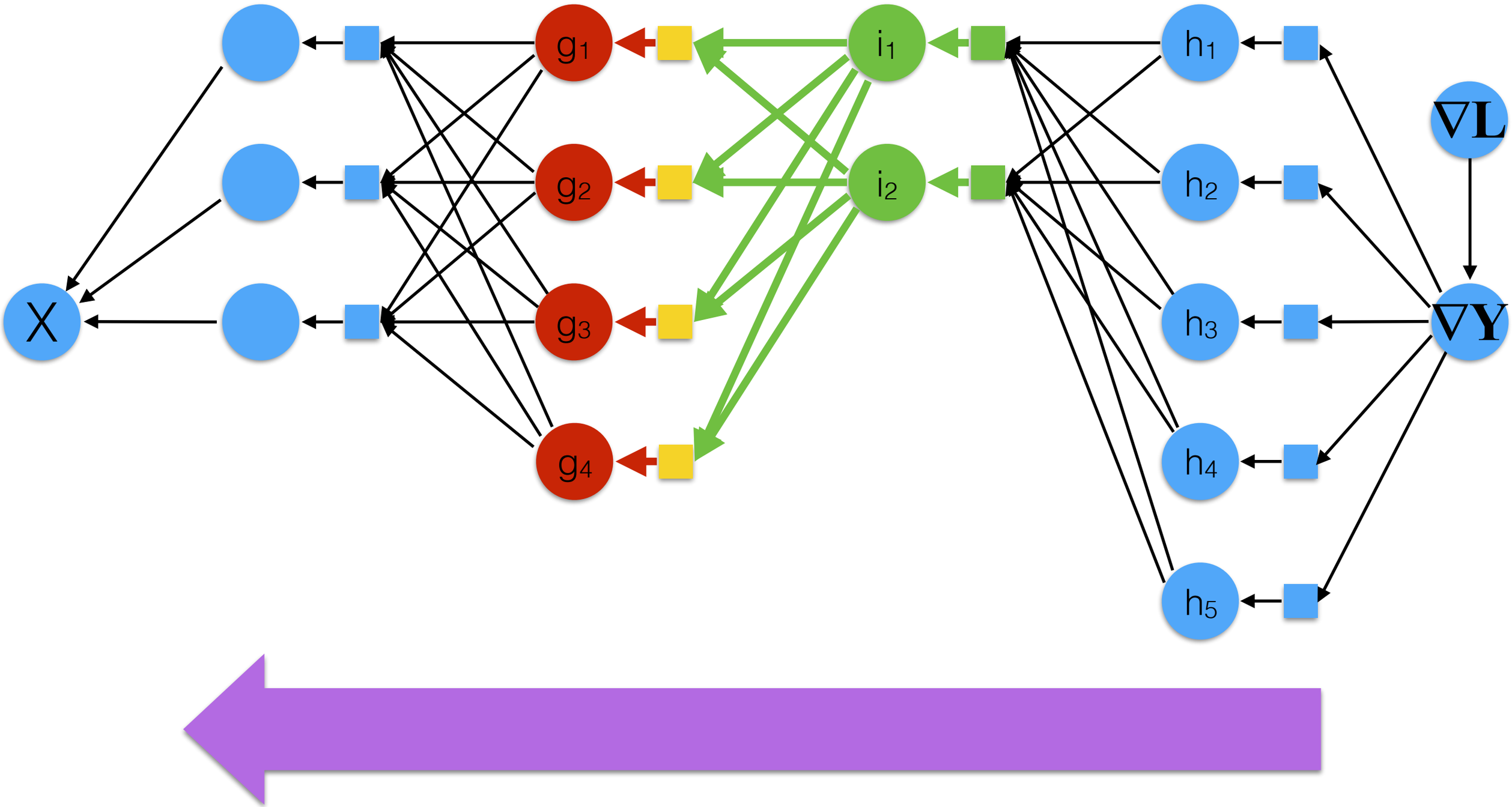
# Backward propagation



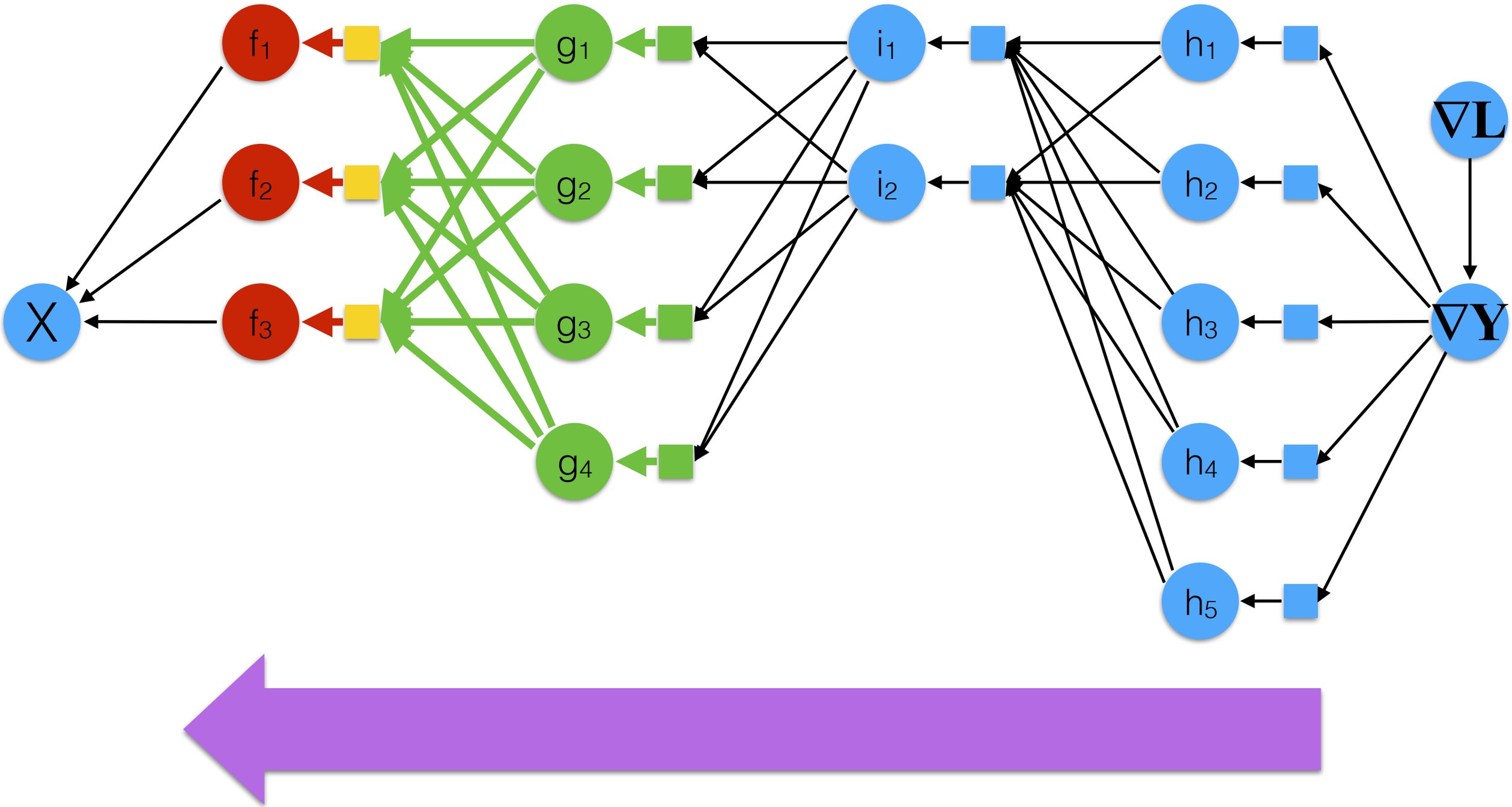
# Backward propagation



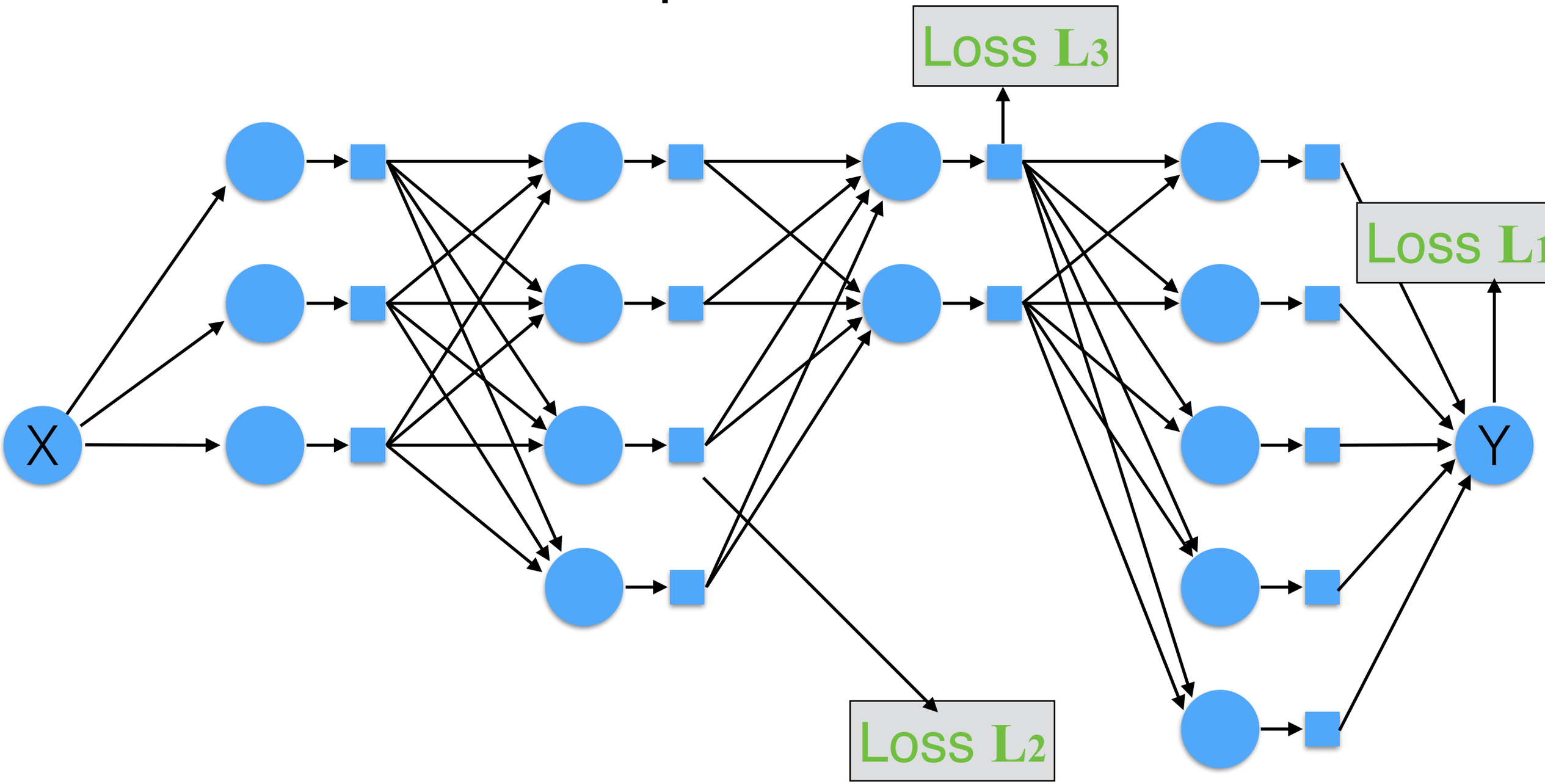
# Backward propagation



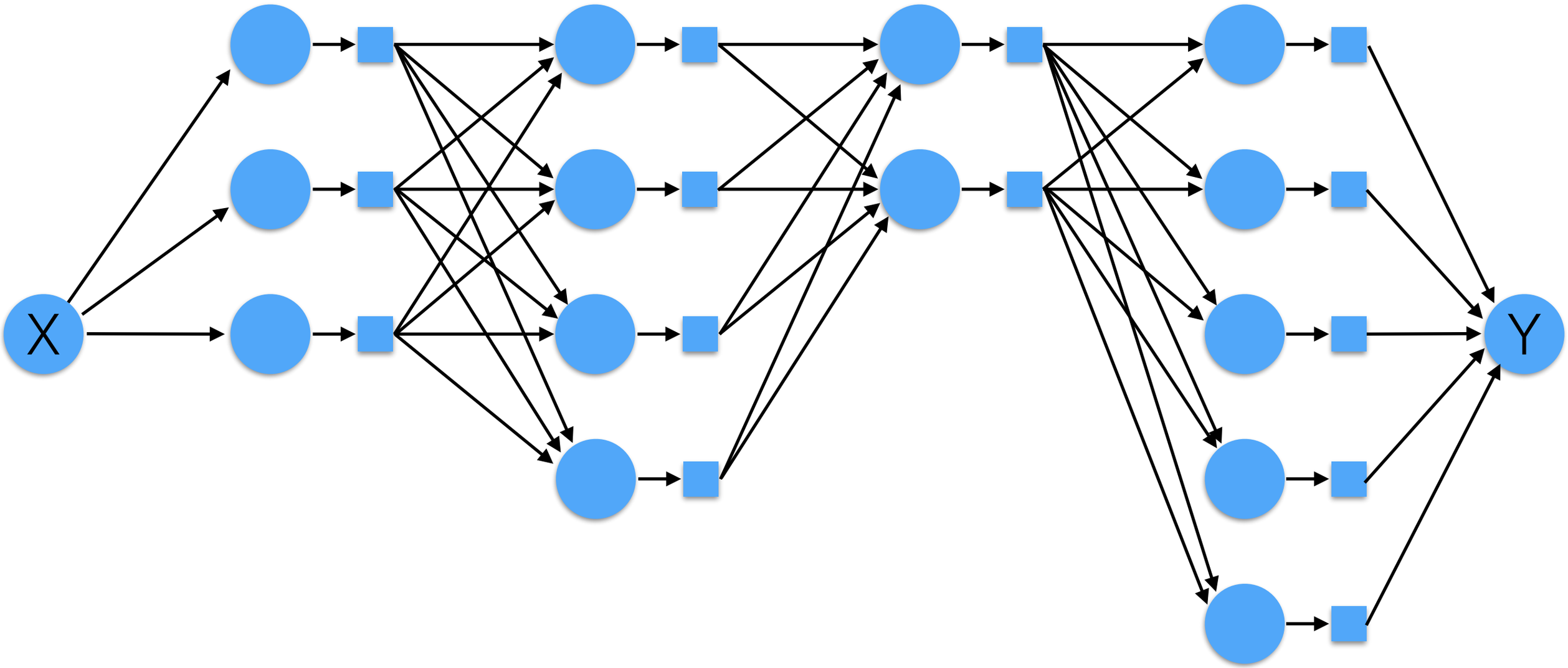
# Backward propagation



# Multiple losses



# Neural Networks Example




Backward propagation

# Summary

# Summary

- A choice of **loss function** matters.

# Summary

- A choice of **loss function** matters.
- Neural networks is essentially **wired neurons** (small functions).

# Summary

- A choice of **loss function** matters.
- Neural networks is essentially **wired neurons** (small functions).
- Weights can be learned using **backward propagation** (e.g. computational graph).

# Break Time

See you in 15 mins!

# Project Discussion

# Next week

- Convolutional Neural Networks
- Training Neural Networks

# Todo

- Midterm on week 7
- Assignment #1 & Project proposal due week 8
- Project Meeting with TA #1 by week 8

# Questions?