Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Стоян Михов

Лекция 8: Невронни мрежи. Многослойни перцептрони. Пропагиране на градиента — Backpropagation

План на лекцията

1. **Формалности за курса (5 мин)**
2. Изкуствени невронни мрежи (10 мин)
3. Представимост на функции с невронни мрежи (10 мин)
4. Многослойни перцептрони (15 мин)
5. Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)
6. Пропагиране назад при логистична регресия (20 мин)

Формалности

- В Moodle е публикувано Домашно задание 1, което следва да бъде предадено до края на деня на 30.11.2021 г.

- Осмата лекция се базира на глави 2, 3 и 4 от втория учебник.

План на лекцията

1. Формалности за курса (5 мин)

2. **Изкуствени невронни мрежи (10 мин)**

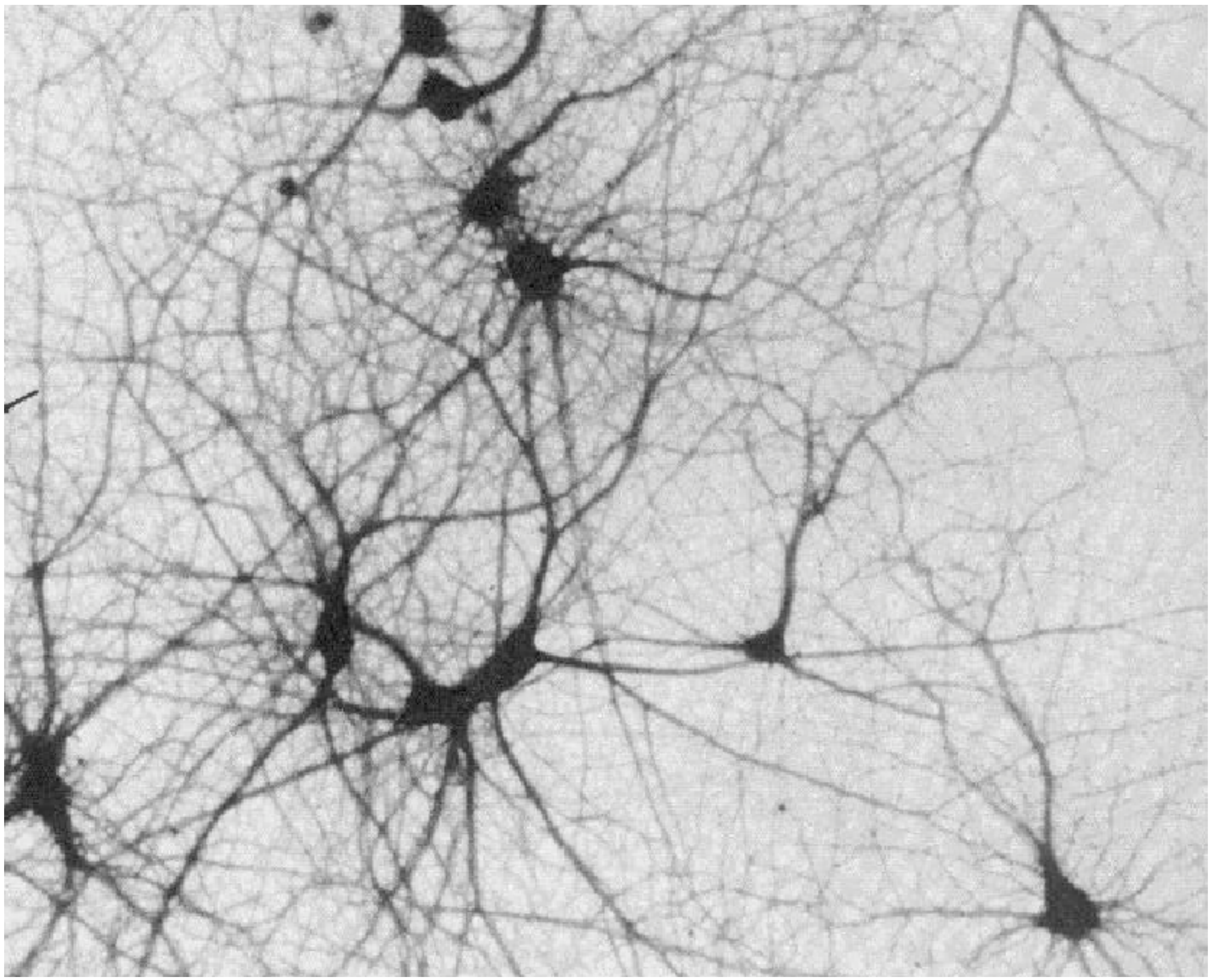
3. Представимост на функции с невронни мрежи (10 мин)

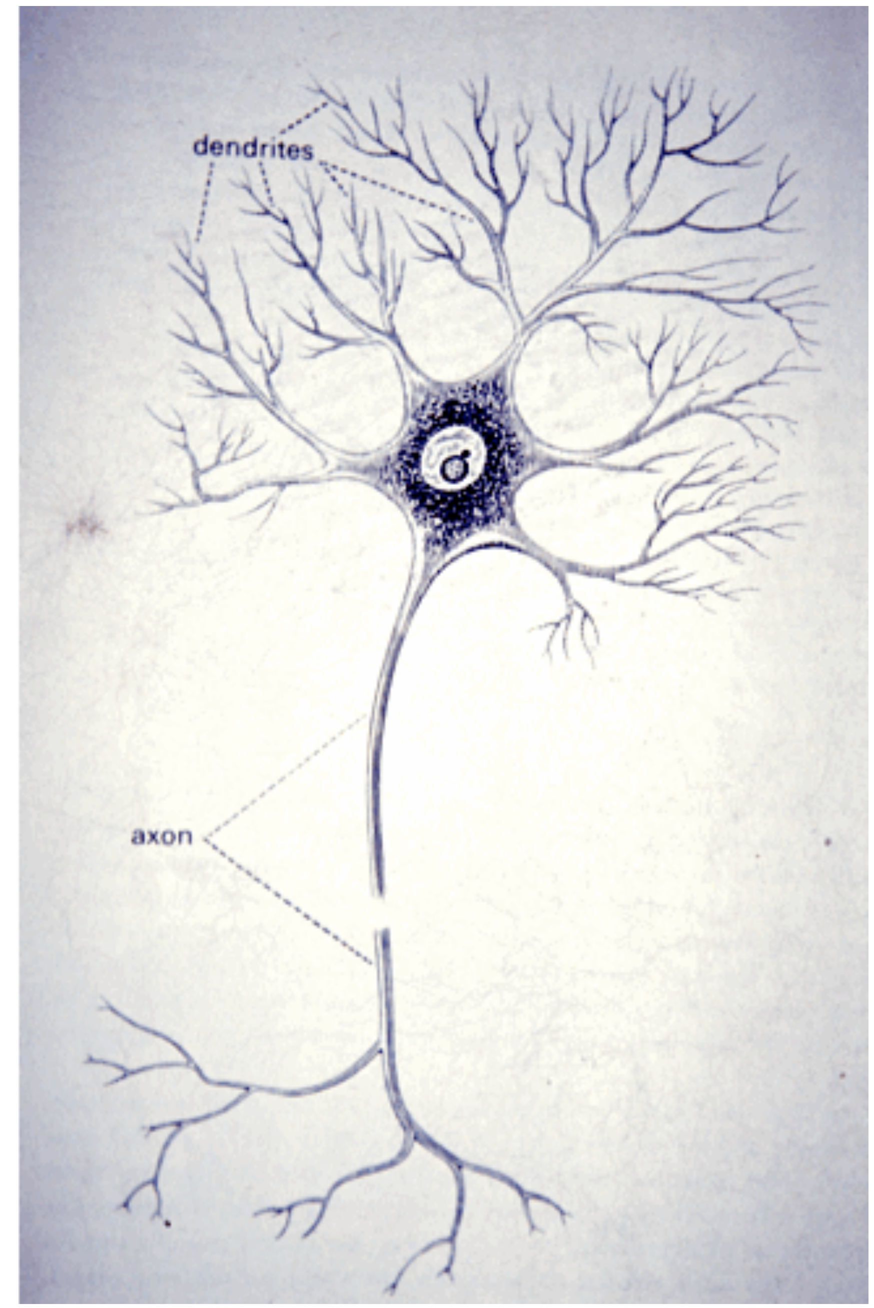
4. Многослойни перцептрони (15 мин)

5. Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)

6. Пропагиране назад при логистична регресия (20 мин)

Неврони





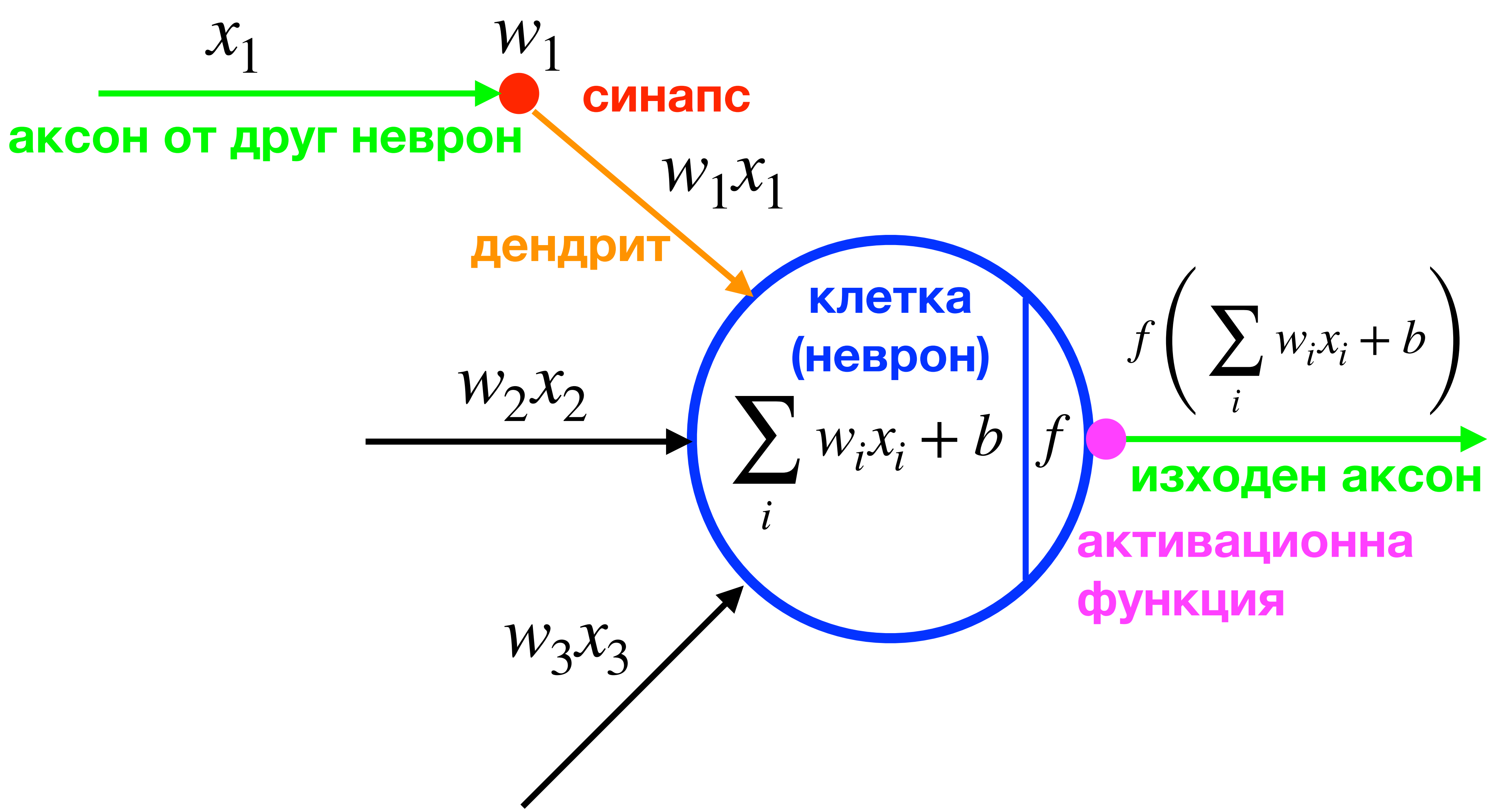
ANN или SNN

- Изкуствените невронни мрежи (ANN) са математически модели, които имат евентуално **далечна прилика** с функционирането на биологичните невронни мрежи. Те са в основата на съвременните методи на изкуствения интелект, чрез които в последните години са постигнати забележителни успехи (ChatGPT, Midjourney AlphaFold, …). Обучението се извършва изключително ефективно с градиентни методи.

- Импулсните невронни мрежи (SNN) обхващат модели, **директно имитиращи** невронната динамика на мозъка. В допълнение към невронното и синаптичното състояние, SNN включват концепцията за време в своя оперативен модел. Засега още не са известни ефективни методи за обучение на импулсни невронни мрежи, но се експериментира със “синаптичната пластичност”, което е вид клъстеризационен метод.

- В рамките на нашия курс ще разглеждаме само Изкуствените невронни мрежи (ANN).

Груба симулация на неврон — изкуствени неврони



При $f = \sigma$ получаваме логистичната регресия

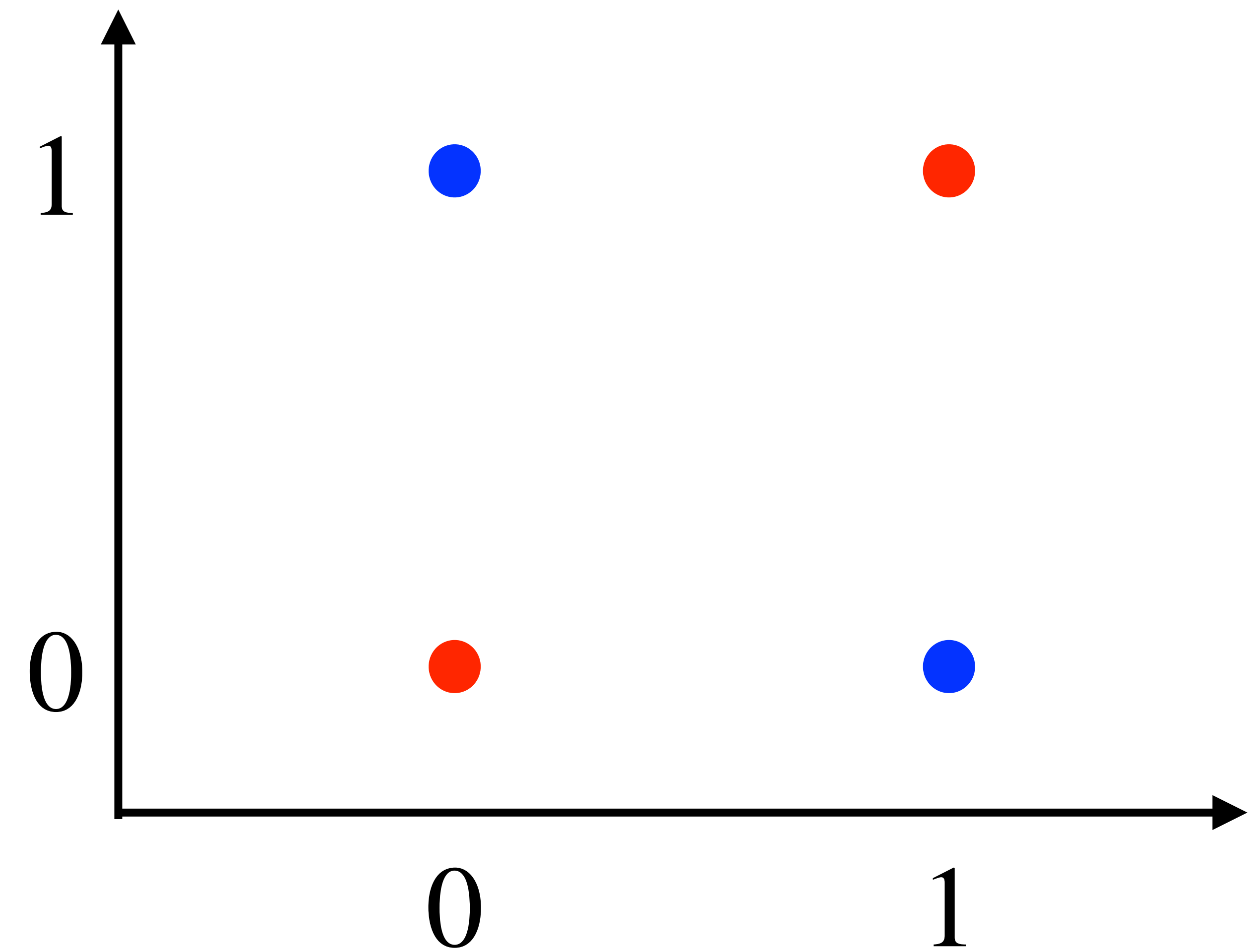
План на лекцията

1. Формалности за курса (5 мин)
2. Изкуствени невронни мрежи (10 мин)
3. **Представимост на функции с невронни мрежи (10 мин)**
4. Многослойни перцептрони (15 мин)
5. Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)
6. Пропагиране назад при логистична регресия (20 мин)

Ограничения на линейните модели — проблемът XOR

$$\left|\begin{array}{l} 0w_1 + 1w_2 + b \geq 0 \\ 1w_1 + 0w_2 + b \geq 0 \\ 1w_1 + 1w_2 + b < 0 \\ 0w_1 + 0w_2 + b < 0 \end{array}\right. \implies$$

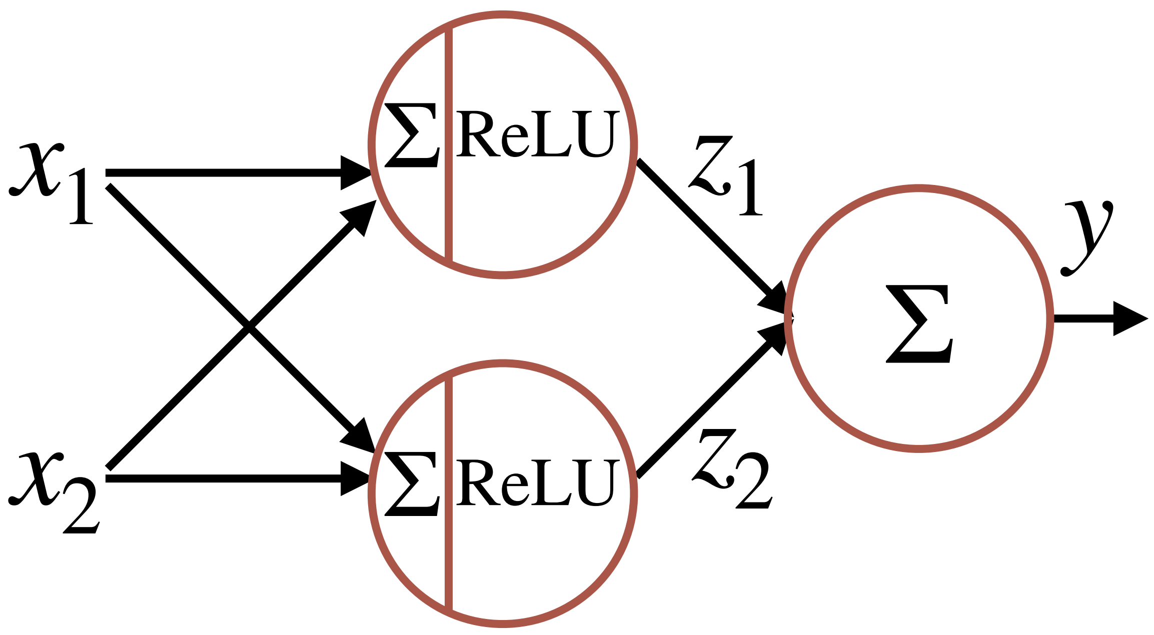
$$\implies \left|\begin{array}{l} w_1 + w_2 + 2b \geq 0 \\ w_1 + w_2 + 2b < 0 \end{array}\right.$$



Не съществува права, която да раздели тези наблюдения

Двуслойна невронна мрежа

- $W' = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, b' = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$
- $\mathbf{w} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, b = \frac{1}{2}$

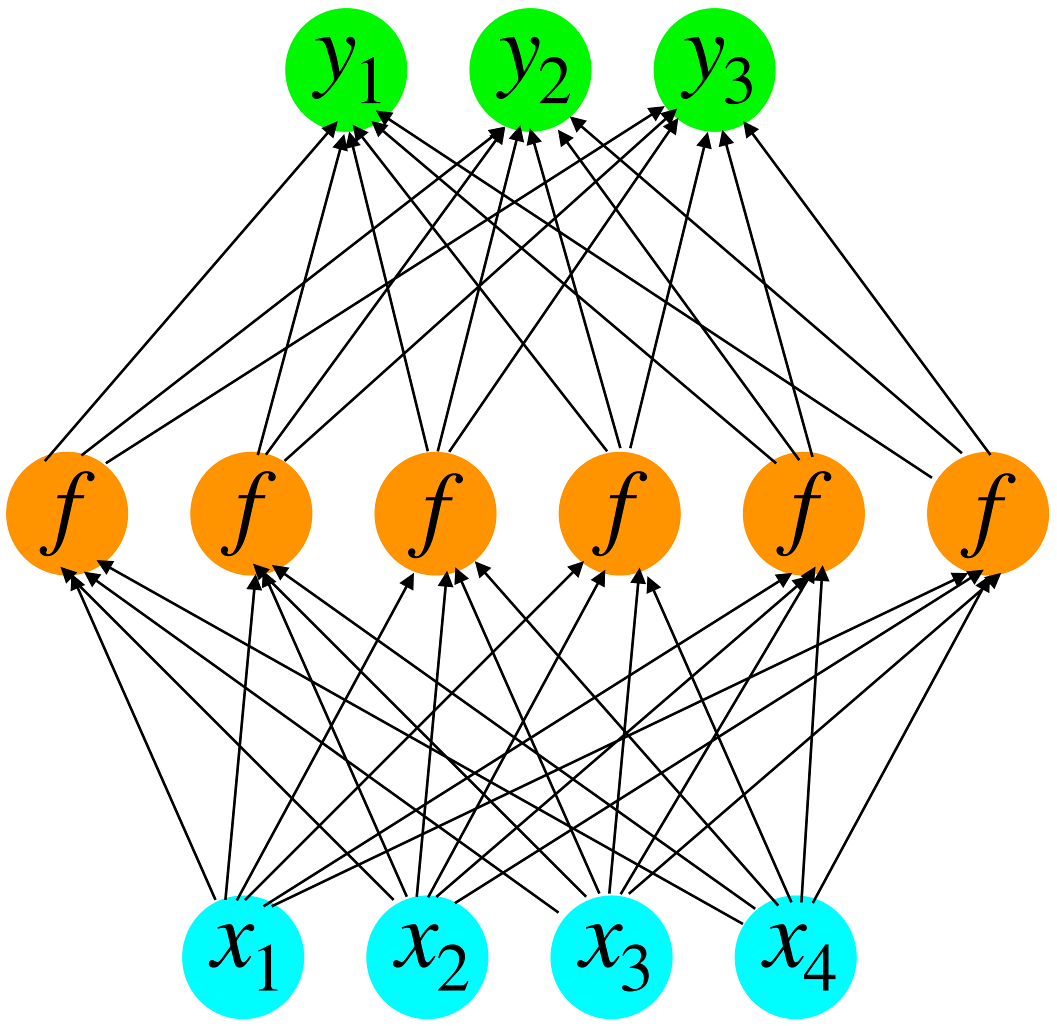


- $\mathbf{z} = \max(W'\mathbf{x} + \mathbf{b}', 0) = \mathrm{ReLU}(W'\mathbf{x} + \mathbf{b}')$
  $y = \mathbf{w}^\top \mathbf{z} + b$
- Така дефинираната функция разделя наблюденията.

Перцептрони

- Прост перцептрон — MLP0:
  $\mathbf{x} \in \mathbb{R}^{d_{in}}, W \in \mathbb{R}^{d_{out} \times d_{in}}, \mathbf{b} \in \mathbb{R}^{d_{out}}$
  $$\mathrm{NN}_{\mathrm{MLP0}}(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$$

- Еднослоен перцептрон (един скрит слой) — MLP1:
  $\mathbf{x} \in \mathbb{R}^{d_{in}}, W^{(1)} \in \mathbb{R}^{d_1 \times d_{in}}, \mathbf{b}^{(1)} \in \mathbb{R}^{d_1}$
  $W^{(2)} \in \mathbb{R}^{d_{out} \times d_1}, \mathbf{b}^{(2)} \in \mathbb{R}^{d_{out}}, f : \mathbb{R} \to \mathbb{R}$
  $\mathrm{NN}_{\mathrm{MLP1}}(\mathbf{x}) = W^{(2)} f(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$



План на лекцията

1. Формалности за курса (5 мин)
2. Изкуствени невронни мрежи (10 мин)
3. Представимост на функции с невронни мрежи (10 мин)
4. **Многослойни перцептрони (15 мин)**
5. Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)
6. Пропагиране назад при логистична регресия (20 мин)

Многослоен перцептрон

Multi Layer Perceptron — MLP

Двуслоен перцептрон (два скрити слоя) — MLP2:

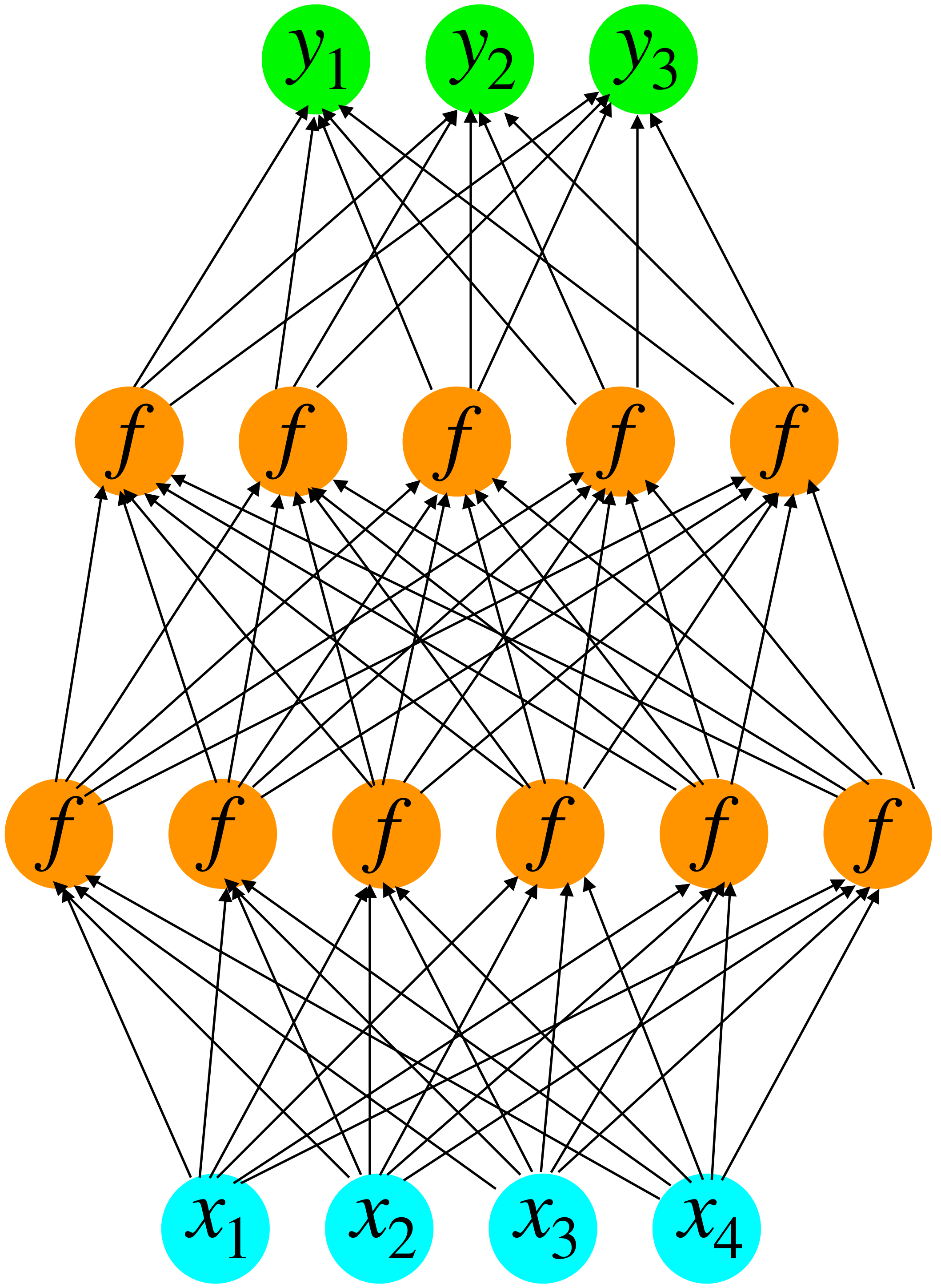
$\mathbf{x} \in \mathbb{R}^{d_{in}}, W^{(1)} \in \mathbb{R}^{d_1 \times d_{in}}, \mathbf{b}^{(1)} \in \mathbb{R}^{d_1}$

$W^{(2)} \in \mathbb{R}^{d_2 \times d_1}, \mathbf{b}^{(2)} \in \mathbb{R}^{d_2}, f^{(1)} : \mathbb{R} \to \mathbb{R}$

$W^{(3)} \in \mathbb{R}^{d_{out} \times d_2}, \mathbf{b}^{(3)} \in \mathbb{R}^{d_{out}}, f^{(2)} : \mathbb{R} \to \mathbb{R}$

$\mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) =$

$= W^{(3)} f^{(2)}(W^{(2)} f^{(1)}(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$



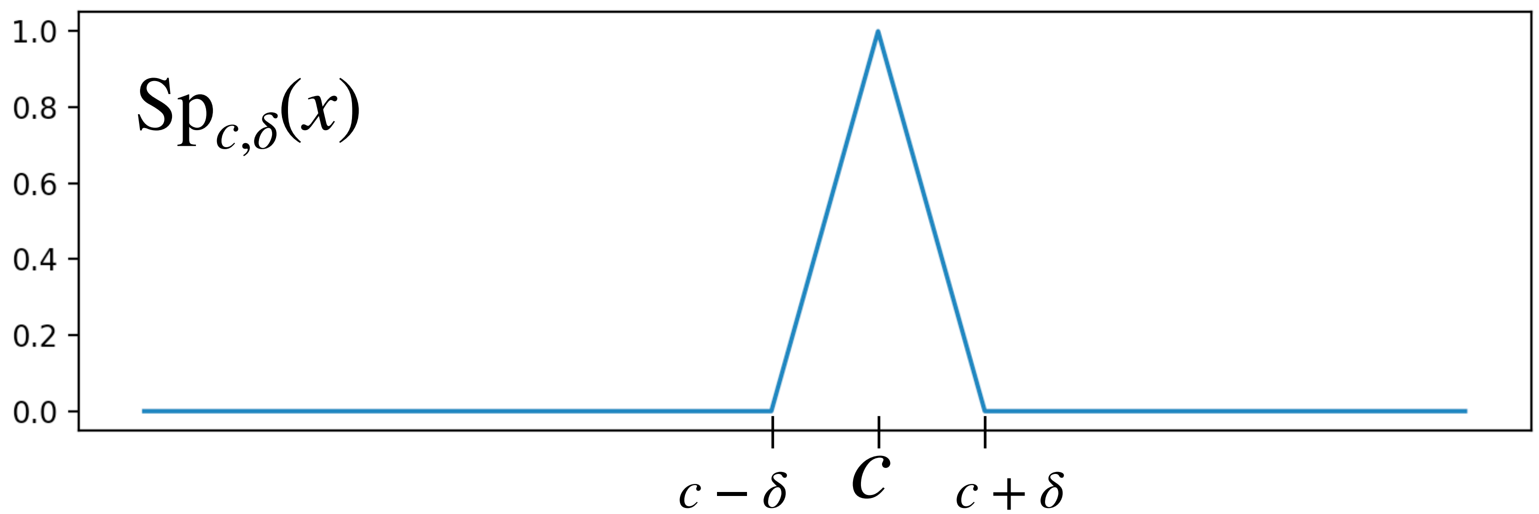
Теорема за представимост на Борелово измеримите функции

- Всяка Борелово измерима функция $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ дефинирана върху компактно множество може да се приближи произволно точно с многослоен перцептрон с поне един скрит слой (MLP1).

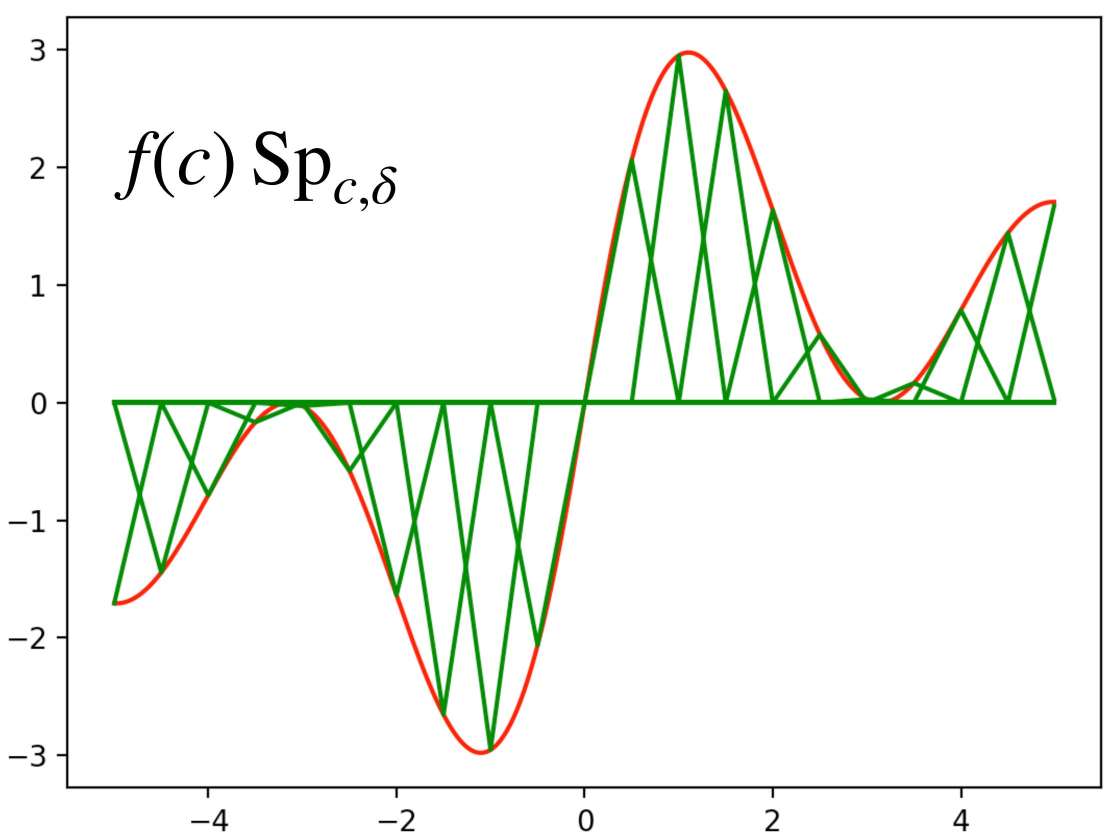
- Съществува многослойна невронна мрежа с даден размер, която не може да се приближи с невронна мрежа с по-малък брой скрити слоеве, освен ако броят на невроните в междинните слоеве не е експоненциално по-голям от броя им в първоначалната мрежа.

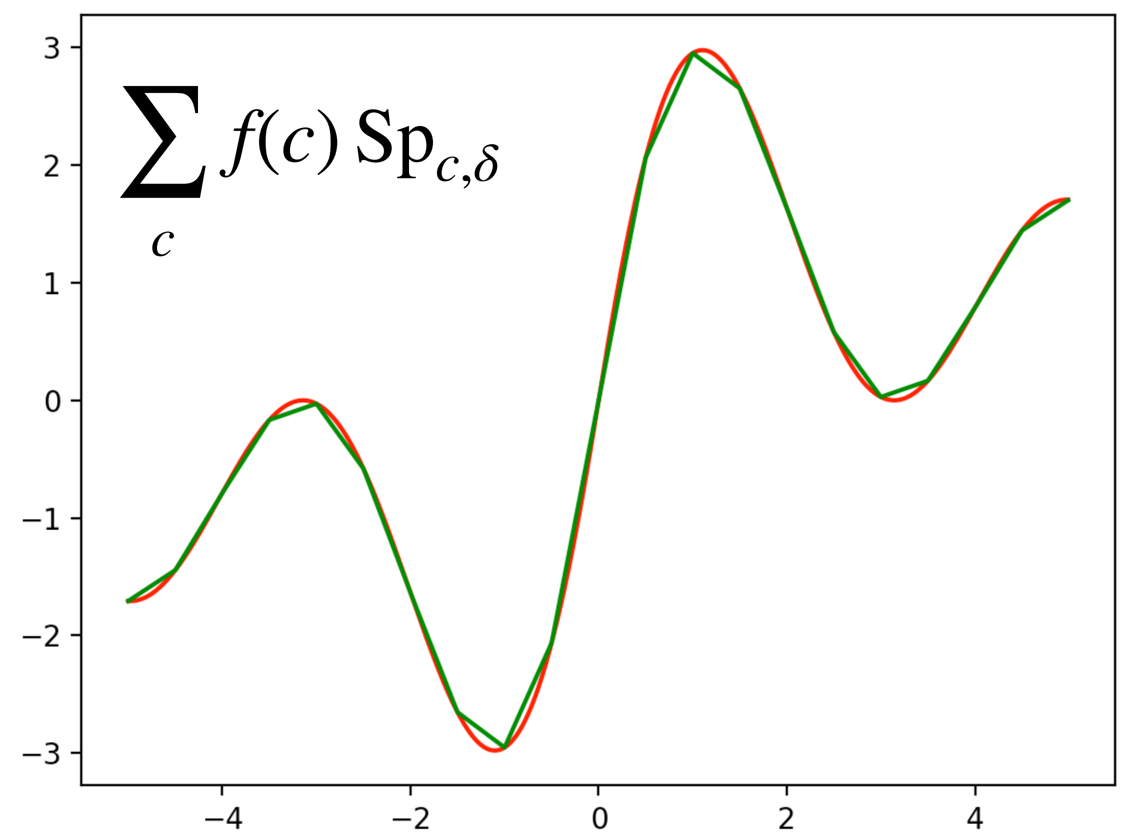
- Доказателство — например в курса “Теория на машинното обучение и някои нейни приложения в невронните мрежи“

Идея за доказателство



$$\mathrm{Sp}_{c,\delta}(x) := \frac{1}{\delta}(\mathrm{ReLU}(x-c+\delta) + \mathrm{ReLU}(-x+c+\delta) - \mathrm{ReLU}(-x+c) - \mathrm{ReLU}(x-c)) - 1$$





Ограничения

- Теоремите за представимост са резултати за съществуване.
    - Те не разглеждат въпроса как да се намери съответно представяне или как да се извърши обучение.
    - Те не дават насоки за необходимия брой параметрите на модела или каква архитектура е необходима.

План на лекцията

1. Формалности за курса (5 мин)
2. Изкуствени невронни мрежи (10 мин)
3. Представимост на функции с невронни мрежи (10 мин)
4. Многослойни перцептрони (15 мин)
5. **Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)**
6. Пропагиране назад при логистична регресия (20 мин)

Защо да изучаваме автоматично диференциране, спускане по градиент, стохастичен градиент и т.н.

- Нали в модерните системи за дълбоко обучение тези функции вече са имплементирани за нас?
- Също, защо трябва да изучаваме компилатори, след като те вече са имплементирани за нас?

1. Да знаете какво става под повърхността винаги е полезно.
2. Автоматичното диференциране не винаги работи перфектно — разбирането на принципите е критично при дебъгване и подобряване на моделите.
3. При по-специални модели може да се наложи добавянето на нови модули. За разширяването на системите е съществено познаването на теорията.
4. Може да ви се наложи да участвате в разработването на нови системи за дълбоко обучение.

Числено намиране на градиент

- Нека $f : \mathbb{R}^n \to \mathbb{R}$. Тогава: $\dfrac{\partial f(x_1, \ldots, x_n)}{\partial x_i} = \lim\limits_{h \to 0} \dfrac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}$ т.е. $\dfrac{\partial f(\mathbf{x})}{\partial x_i} = \lim\limits_{h \to 0} \dfrac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$, където $\mathbf{e}_i$ е i-тия базисен вектор.

- Полагайки достатъчно малко h, примерно $h = 10^{-4}$, ние можем да намерим приближение на производната: $\dfrac{\partial f(\mathbf{x})}{\partial x_i} \approx \dfrac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$.

- По добра апроксимация на производната: $\dfrac{\partial f(\mathbf{x})}{\partial x_i} \approx \dfrac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}$.

- Градиента намираме, като апроксимираме производната в дадената точка по всяка от n-те координати.

- **Задача**: Докажете, че втората формула ни дава по-добро приближение на производната, като оцените порядъка на грешката.

- **Сложност**: Ако сложността на израза за f е от порядък $O(k)$ то сложността за численото намиране на градиента на f в точката $\mathbf{x}$ е от порядък $O(nk)$.

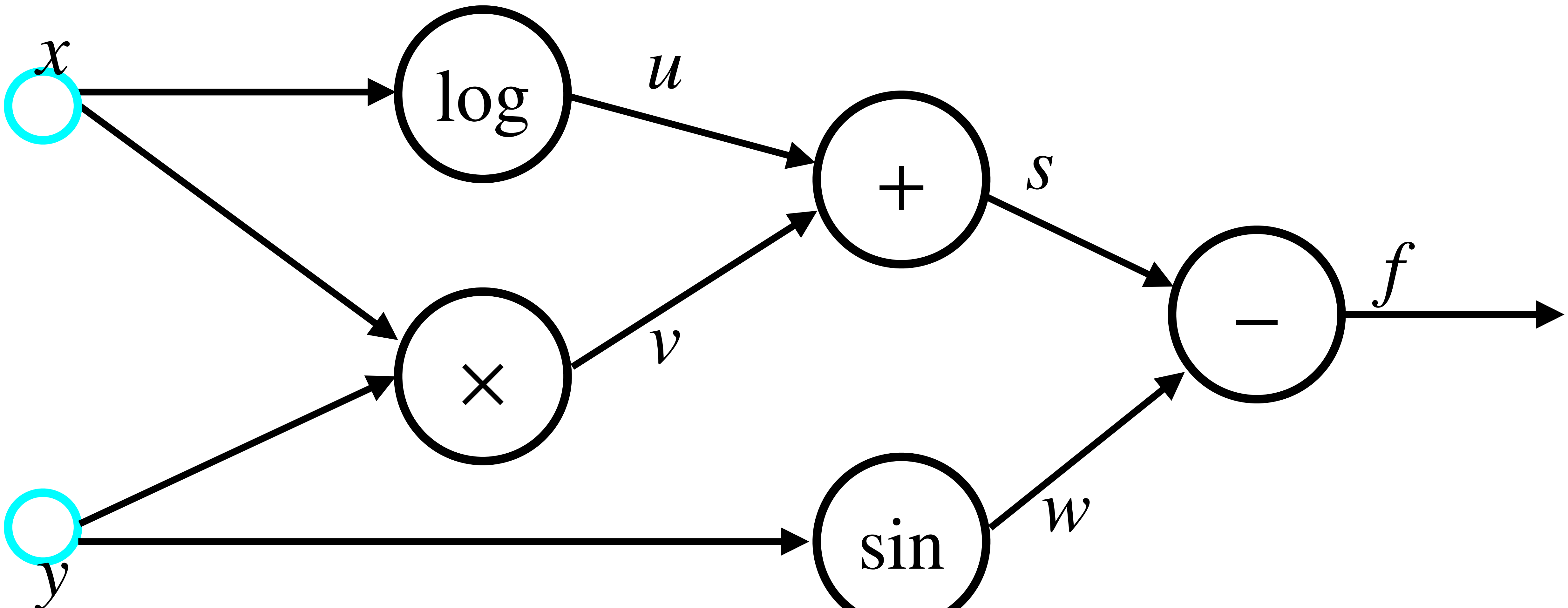
Аналитично намиране на градиент

- Нека $f : \mathbb{R}^n \to \mathbb{R}$. Нека сме намерили аналитични изрази за производните $\dfrac{\partial f(\mathbf{x})}{\partial x_i}$ за $i = 1,2,\ldots,n$.
- Производната по дадено направление намираме, като заместим в съответния израз за производната с дадената точка.
- Градиента намираме, като заместим в израза за производната по всяка от n-те координати с дадената точка.
- **Сложност**: Ако сложността на изразите за $\dfrac{\partial f(\mathbf{x})}{\partial x_i}$ е от порядък $O(k')$, то сложността за аналитичното намиране на градиента на f в точката $\mathbf{x}$ е от порядък $O(nk')$.
- **Задача**: Докажете, че съществуват изрази със сложност $O(k)$, за които сложността на израза за производната е от порядък $O(2^k)$.

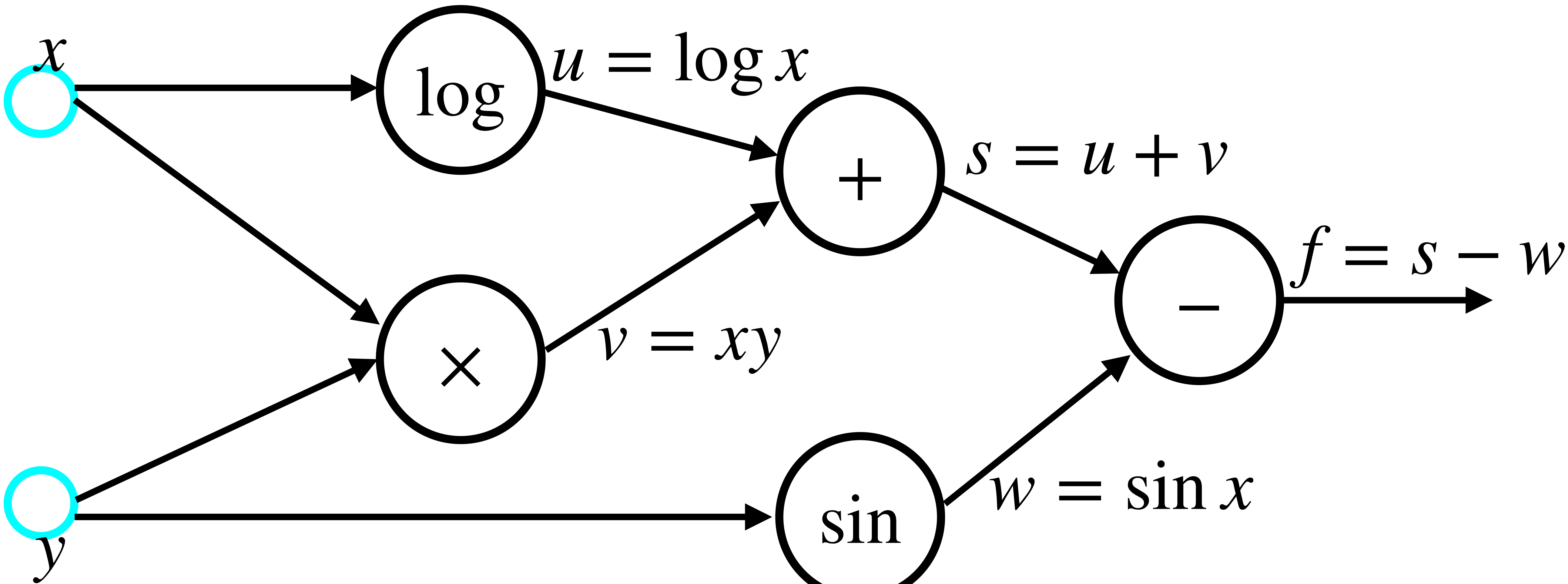
Намиране на градиент чрез пропагиране назад — **Backpropagation**

- Намира точните производни и градиент в дадена точка — не е числено приближение.
- Използва се аналитичен израз за производните само за локалните функции — избягва се намирането на изразите за производните на целевата функция.
- Преизползват се междинните резултати от изчисленията, с което се постига оптимална изчислителна сложност.
- Сложността за намиране на градиента е $O(k)$, за израз със сложност k.

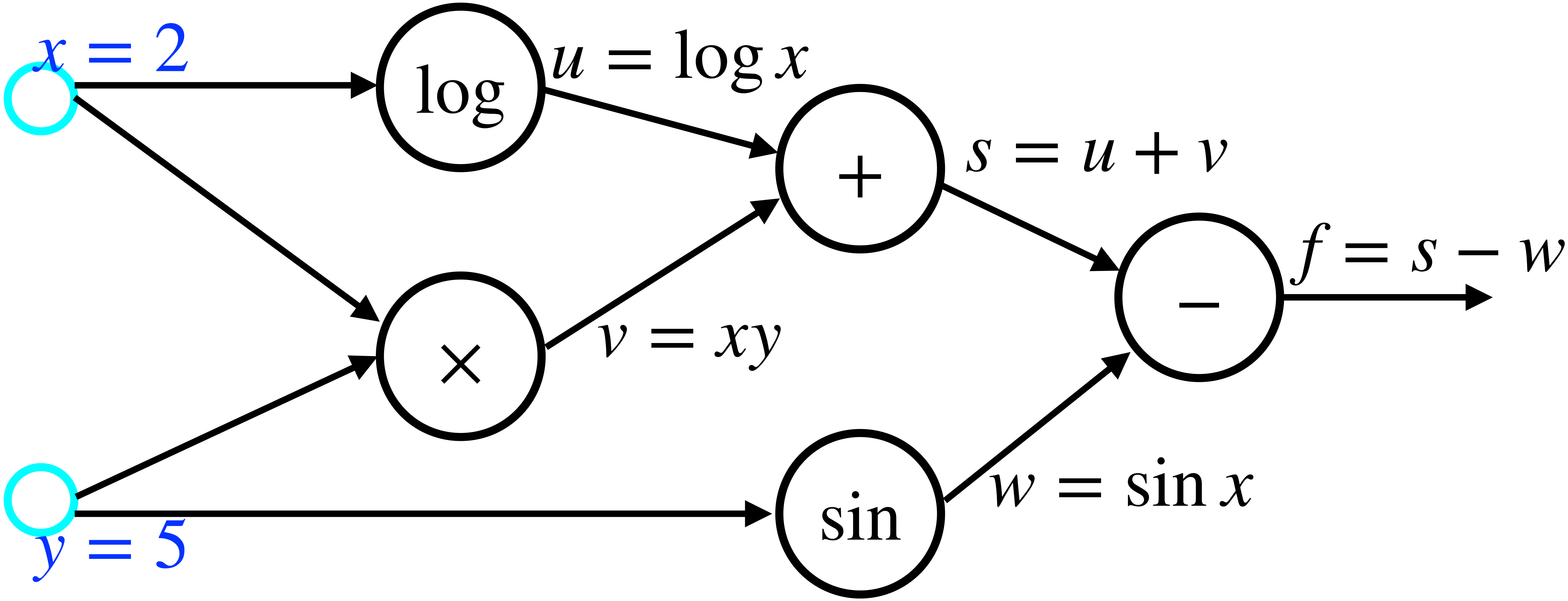
Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$



Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

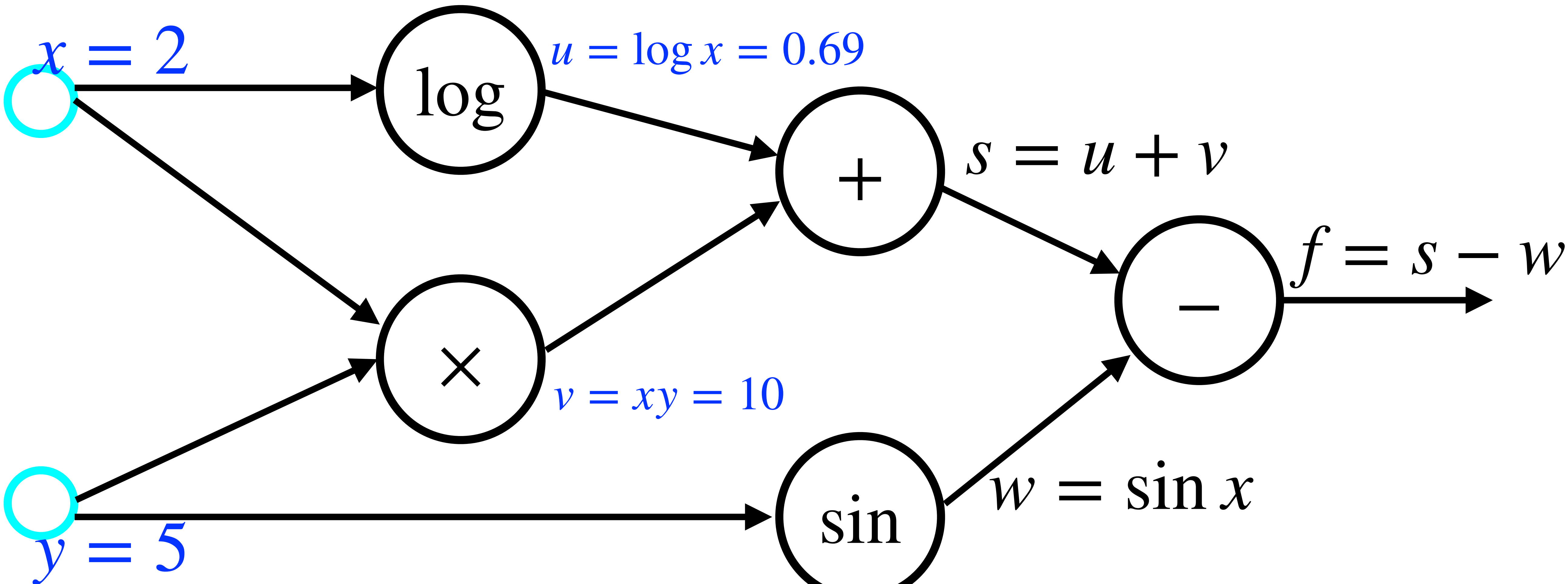


Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$



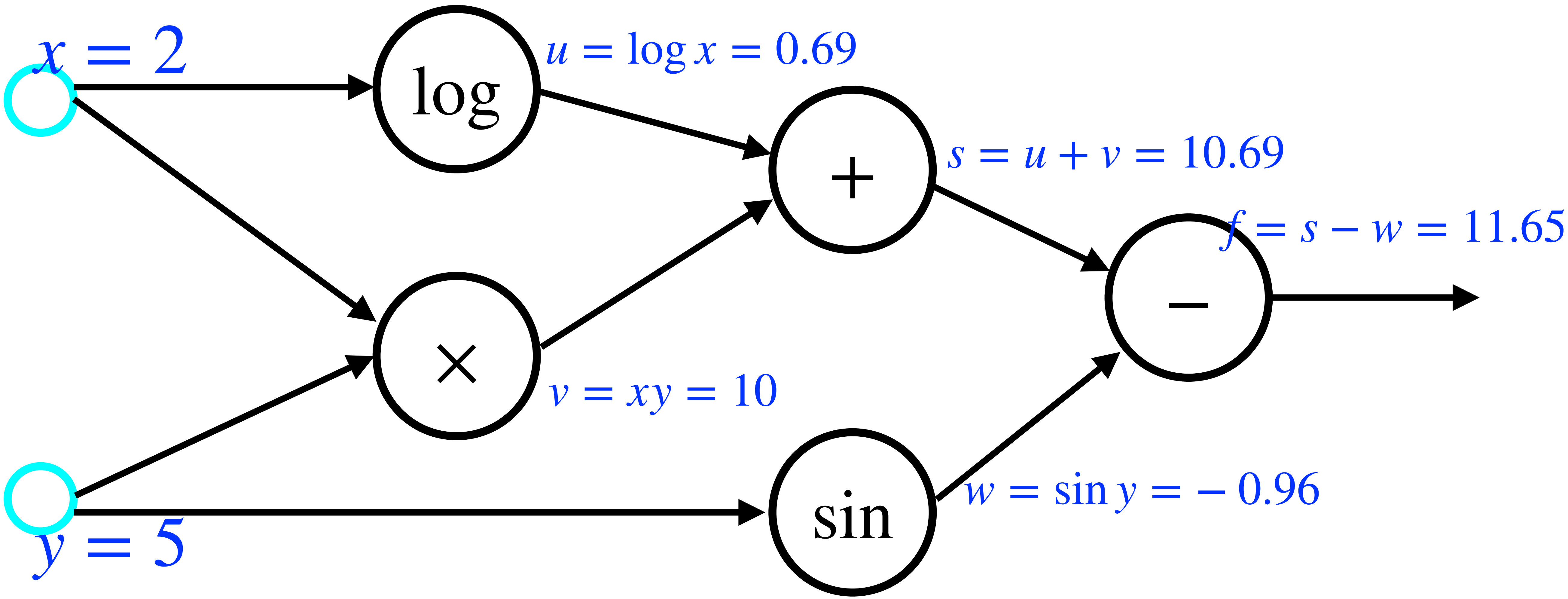
Пропагиране напред — Forward propagation

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$



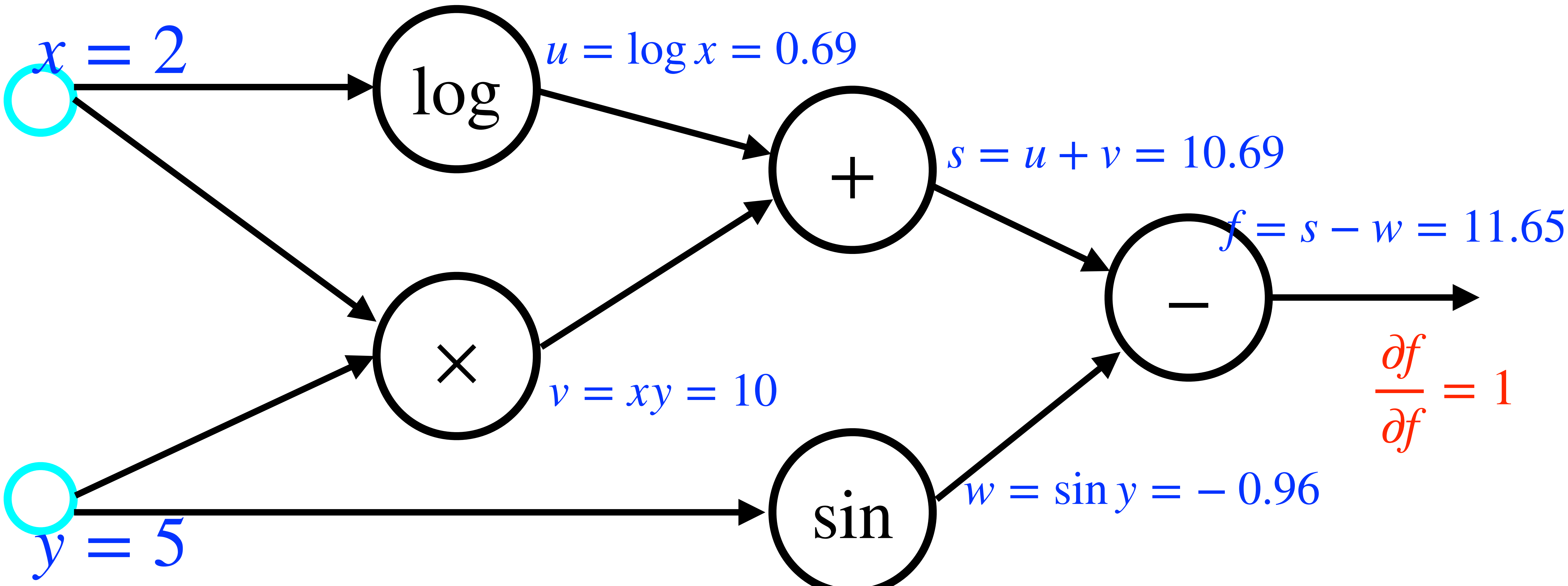
Пропагиране напред — Forward propagation

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$



Пропагиране напред — Forward propagation

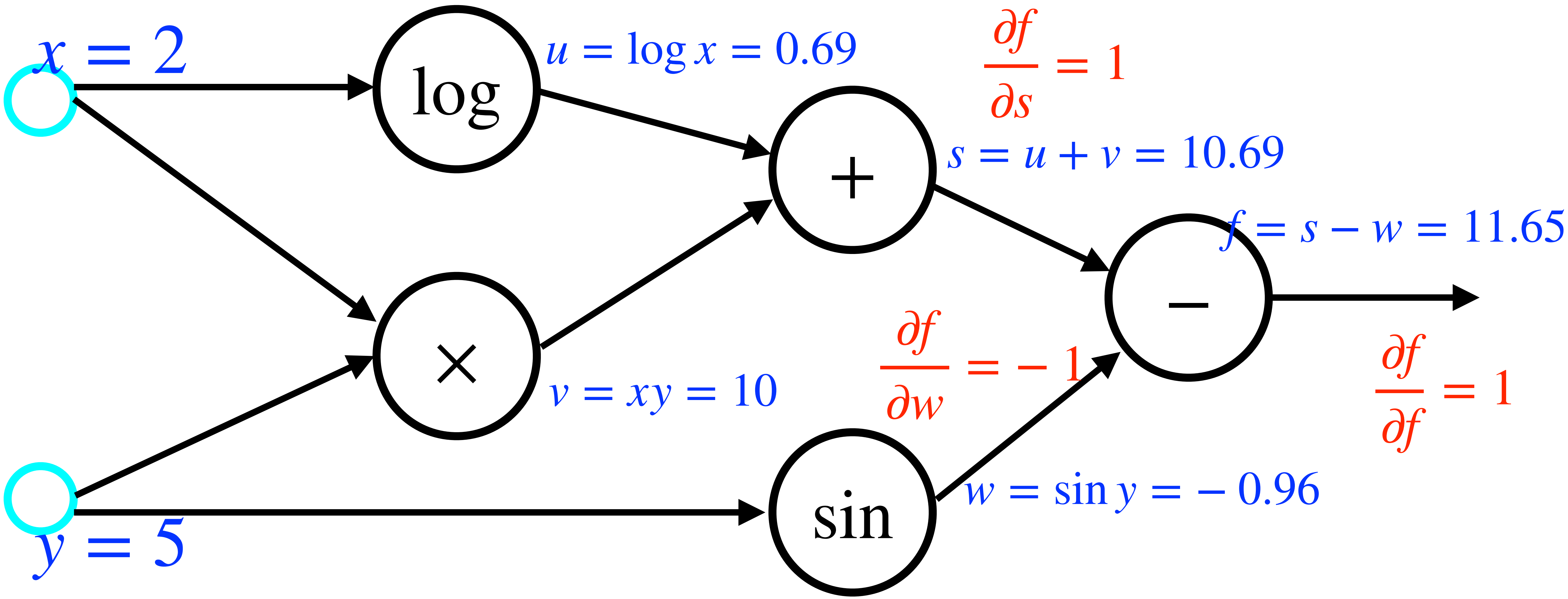
Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$



Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

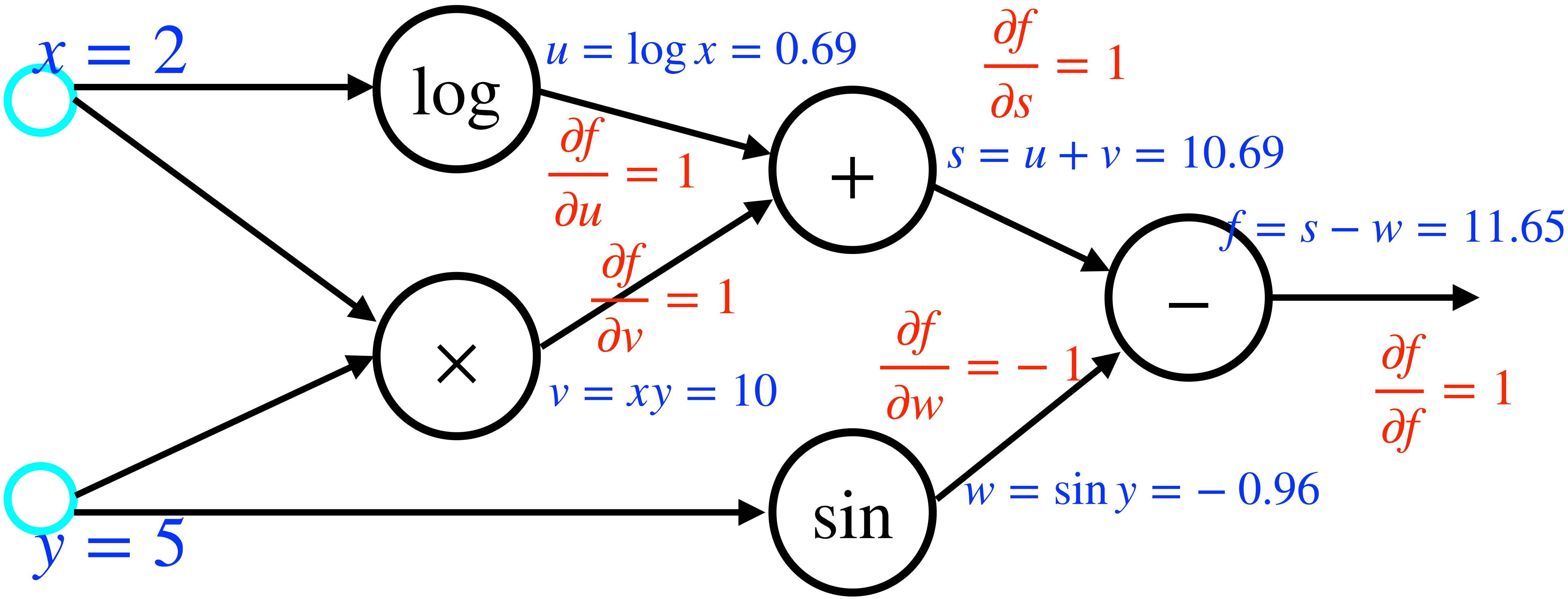


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial f}{\partial f}(-1) = -1, \quad \frac{\partial f}{\partial s} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial s} = \frac{\partial f}{\partial f}1 = 1$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

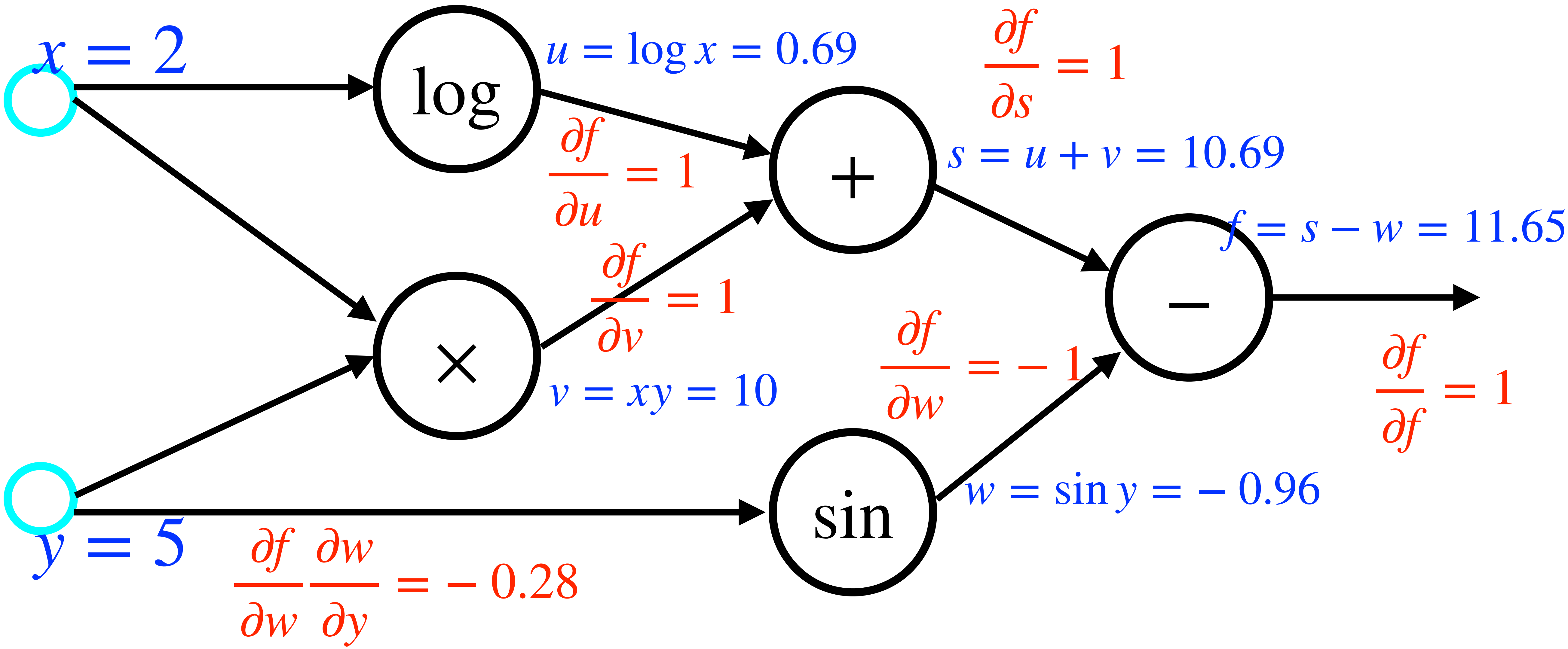


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial s}\frac{\partial s}{\partial u} = \frac{\partial f}{\partial s}1 = 1, \quad \frac{\partial f}{\partial v} = \frac{\partial f}{\partial s}\frac{\partial s}{\partial v} = \frac{\partial f}{\partial s}1 = 1$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

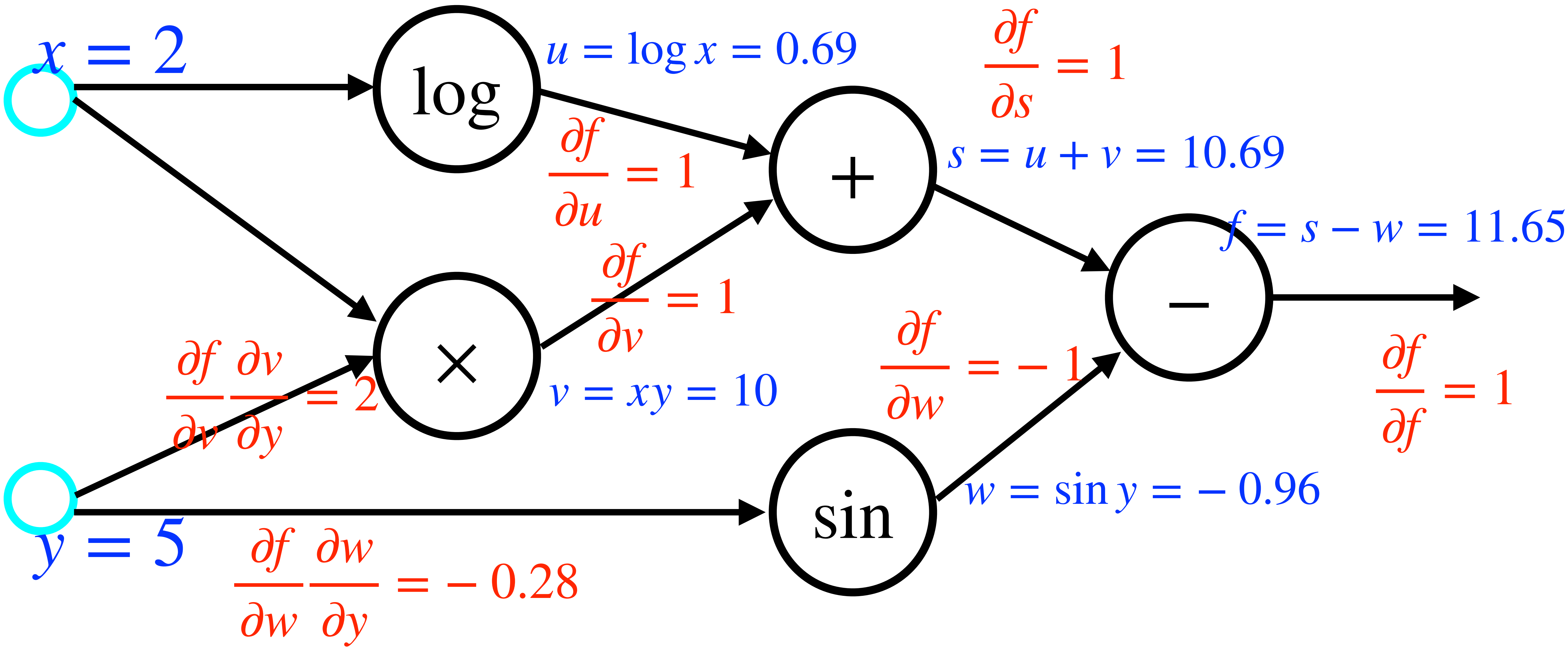


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial w}\frac{\partial w}{\partial y} = \frac{\partial f}{\partial w}\cos(y) = (-1)\cos(5) = -0.28$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

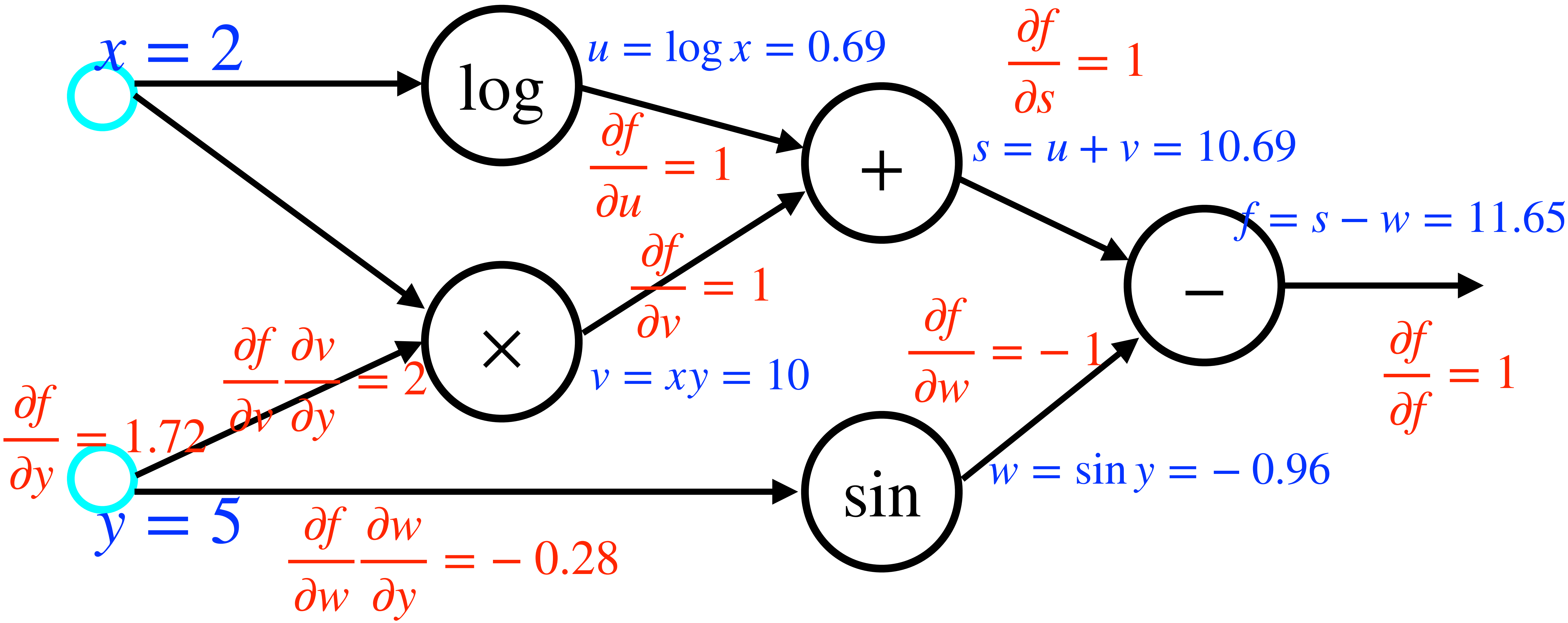


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial v}\frac{\partial v}{\partial y} = \frac{\partial f}{\partial v}x = 2$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

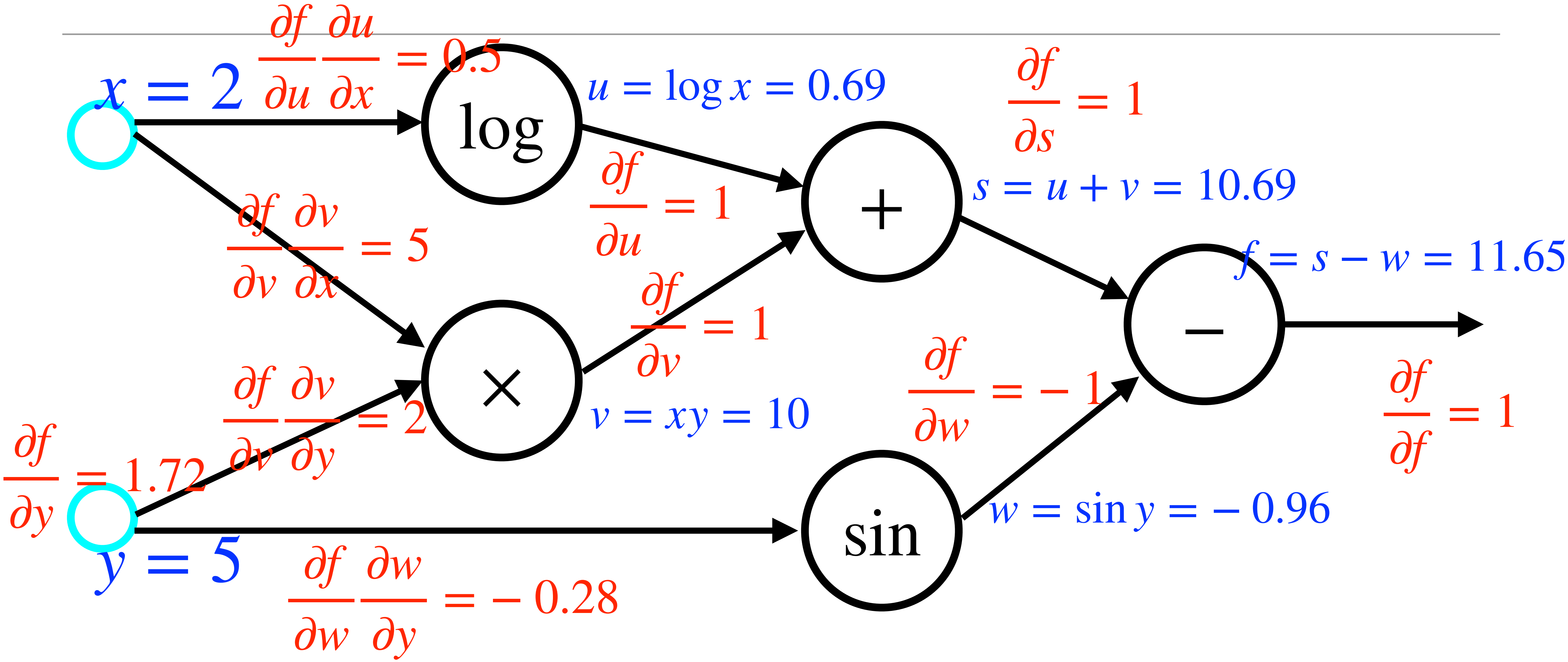


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial v}\frac{\partial v}{\partial y} + \frac{\partial f}{\partial w}\frac{\partial w}{\partial y} = 2 - 0.28 = 1.72$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

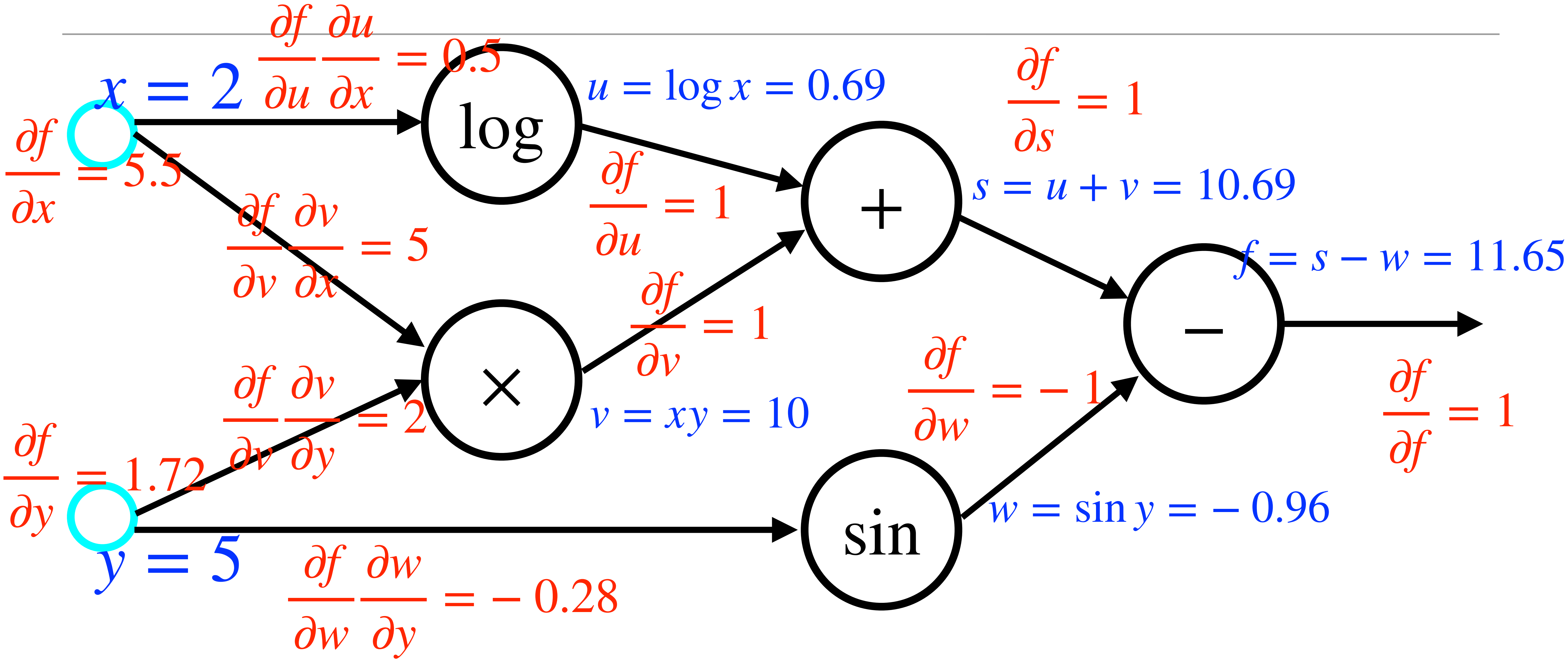


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial u}\frac{\partial u}{\partial x} = \frac{\partial f}{\partial u}\frac{1}{x} = 0.5, \frac{\partial f}{\partial v}\frac{\partial v}{\partial x} = \frac{\partial f}{\partial v}y = 5$$

Пример: $f(x, y) = (\log(x) + xy) - \sin(y)$

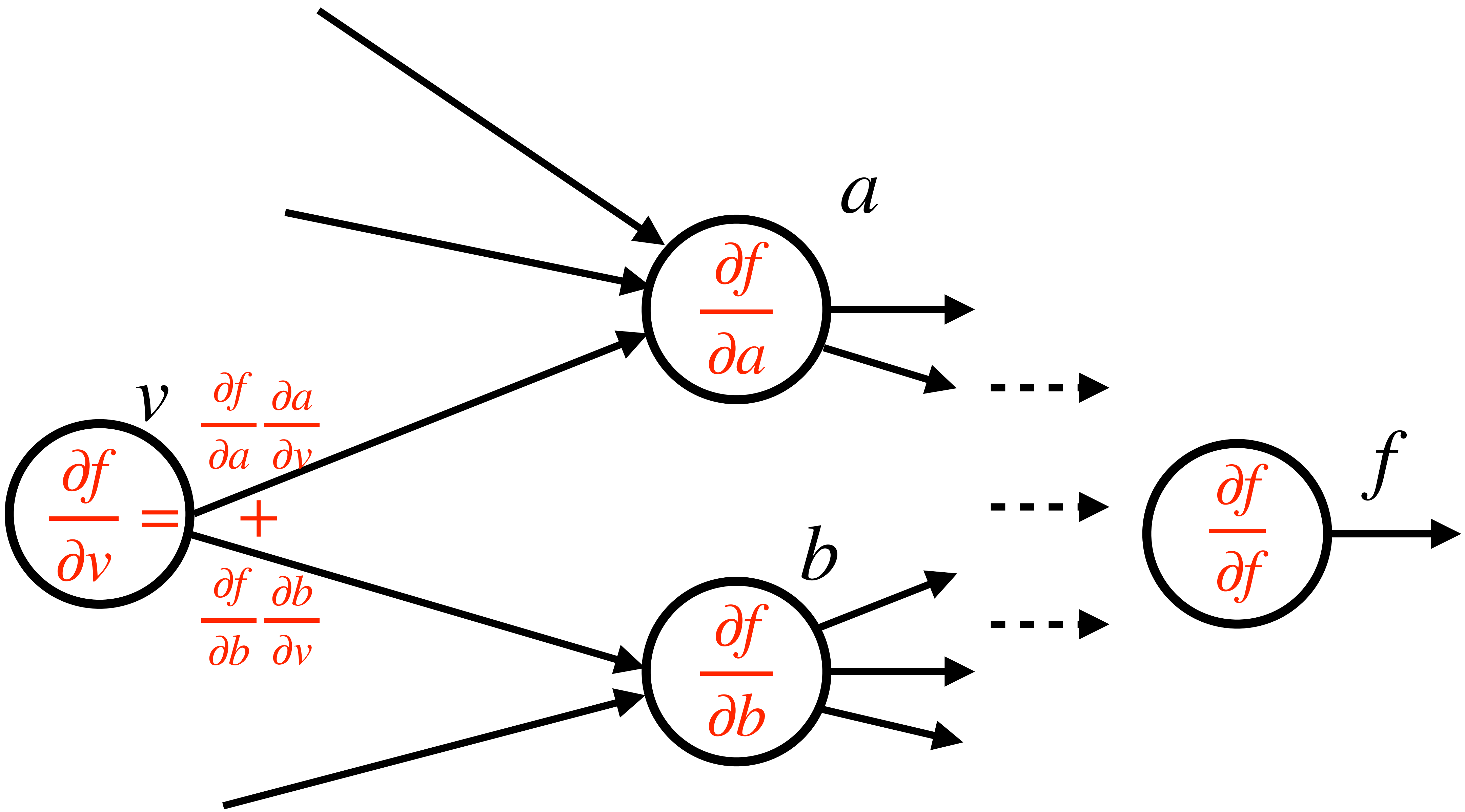


Пропагиране напред — Forward propagation

Пропагиране назад— Back propagation

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial f}{\partial v}\frac{\partial v}{\partial x} = 5 + 0.5 = 5.5$$

Основен инвариант



Формализация на Backpropagation — изчислителен граф

- Даден е ацикличен граф $G = (V, E), E \subset V \times V$.

- За всеки връх $v \in V$ (непосредствените) предшественици означаваме с $P_G(v) = [p \in V \mid (p, v) \in E]$. Ще предполагаме, че $P_G(v)$ е списък.

- Листата на графа ще означаваме с $L(G) = [v \mid P_G(v) = \emptyset]$.

- Крайни върхове на графа ще означаваме с $T(G) = [v \mid \neg \exists u \in V : (v, u) \in E]$. Ще предполагаме, че в графа съществува единствен краен връх: $|T(G)| = 1$

Формализация на Backpropagation — изчислителен граф

- За всеки връх $v \in V \setminus L(G)$ ще предполагаме, че е дефинирана функция за изчисляване на стойността: $\mathrm{calc}(G, v) : \mathbb{R}^{|P_G(v)|} \to \mathbb{R}$.

- За всеки връх $v \in V \setminus L(G)$ и всеки негов предшественик $u \in P_G(v)$ ще предполагаме, че е дефинирана функция за изчисляване на частната производна на функцията $\mathrm{calc}(G, v)$ спрямо аргумента u, която бележим с $\mathrm{deriv}(G, v, u) : \mathbb{R}^{|P_G(v)|} \to \mathbb{R}$.

Backpropagation алгоритъм

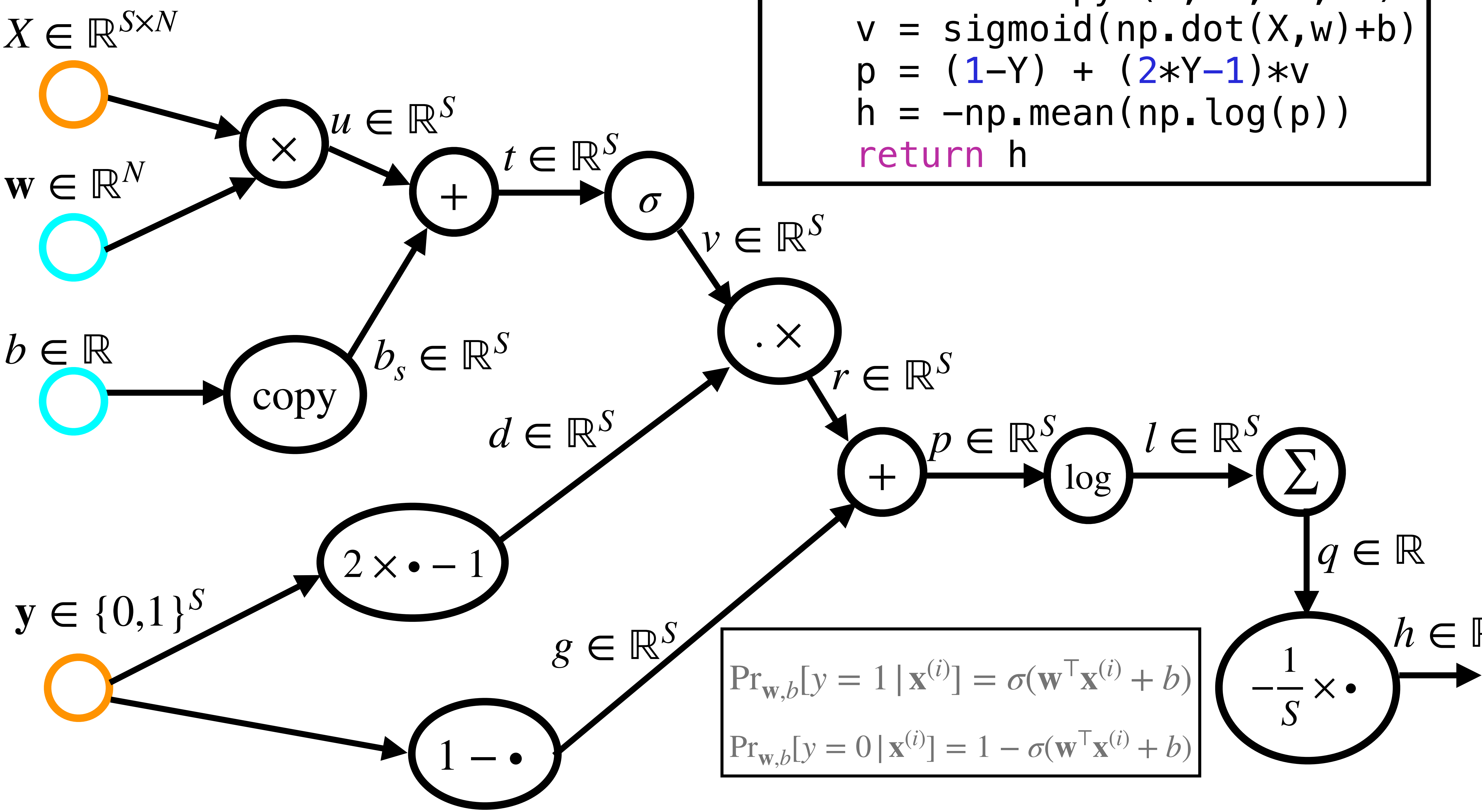
```
Forward(G):
for v in topologicalSort(G) do
   if v in Leafs(G) then read(F(v))
   else
      [p1,p2,...,pk] <- Predecessors(G)(v)
      F(v) <- calc(G,v)(F(p1),…,F(pk))
return F

Backward(G,F):
for v in topologicalSort(G) do B(v) <- 0
for v in reverseTopologicalSort(G) do
   if v in Top(G) then B(v) <- 1
   [p1,p2,...,pk] <- Predecessors(G)(v)
   for p in Predecessors(G)(v) do
      B(p) += B(v) * deriv(G,v,p)(F(p1),…,F(pk))
return B
```

План на лекцията

1. Формалности за курса (5 мин)
2. Изкуствени невронни мрежи (10 мин)
3. Представимост на функции с невронни мрежи (10 мин)
4. Многослойни перцептрони (15 мин)
5. Намиране на градиент чрез пропагиране назад — Backpropagation (20 мин)
6. **Пропагиране назад при логистична регресия (20 мин)**

Логистична регресия — векторен запис

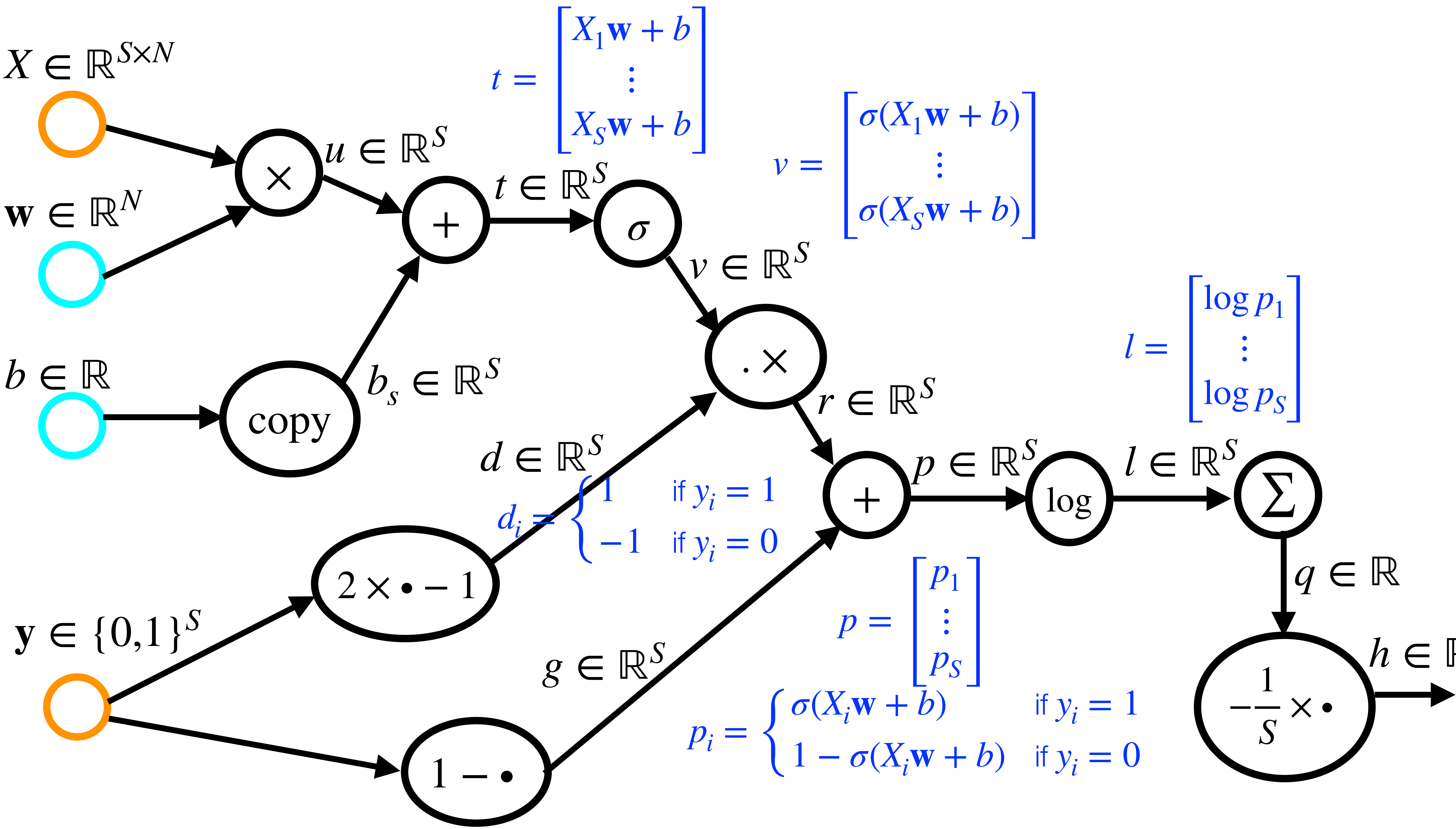
```python
def crossEntropyS(X, Y, w, b):
    v = sigmoid(np.dot(X,w)+b)
    p = (1-Y) + (2*Y-1)*v
    h = -np.mean(np.log(p))
    return h
```



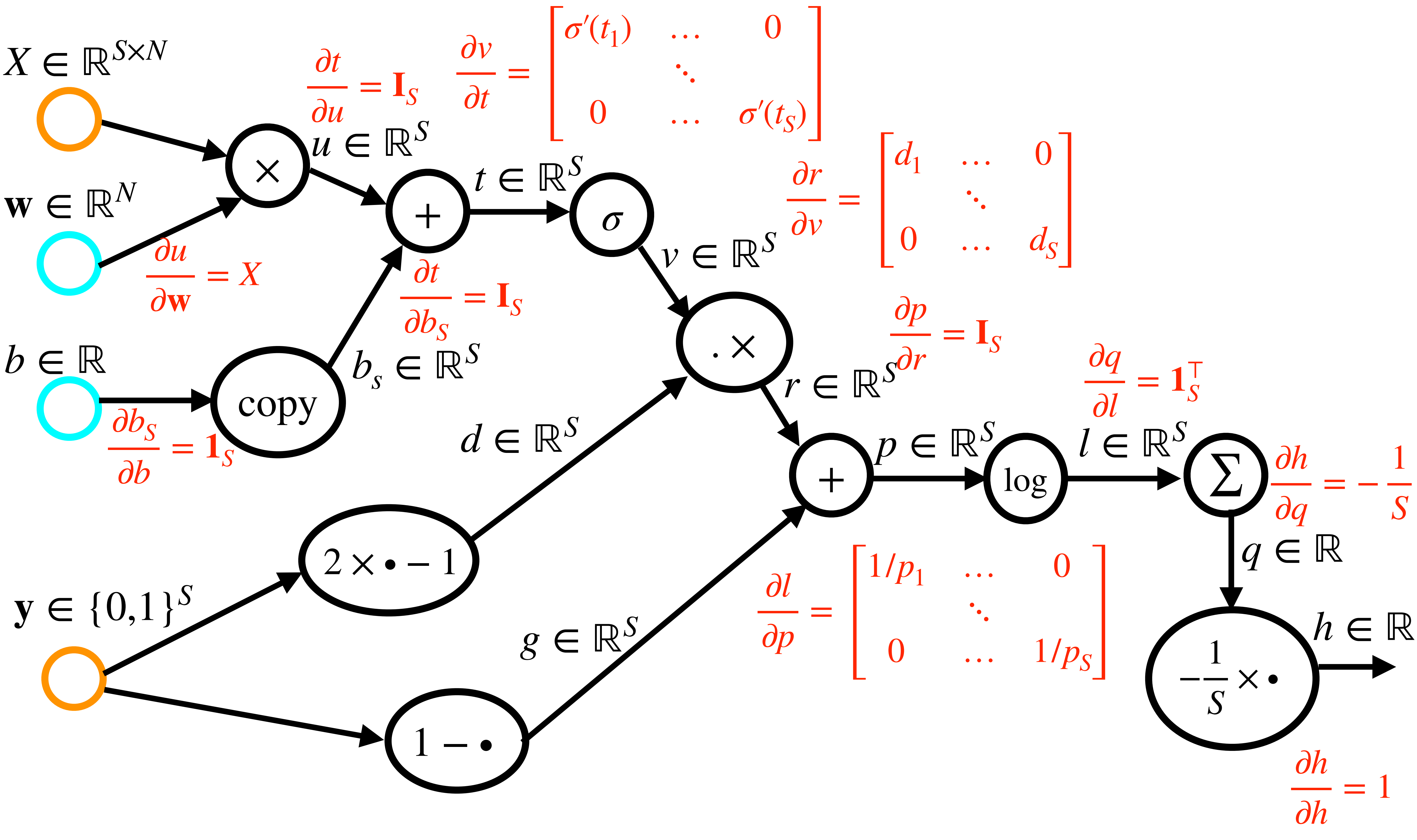
$$\Pr_{\mathbf{w},b}[y = 1 \,|\, \mathbf{x}^{(i)}] = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$$

$$\Pr_{\mathbf{w},b}[y = 0 \,|\, \mathbf{x}^{(i)}] = 1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$$

Логистична регресия — векторен запис



Логистична регресия — векторен запис



$$\frac{\partial h}{\partial t} = \frac{\partial h}{\partial q}\frac{\partial q}{\partial l}\frac{\partial l}{\partial p}\frac{\partial p}{\partial r}\frac{\partial r}{\partial v}\frac{\partial v}{\partial t}$$

$$= -\frac{1}{S}\mathbf{1}_S^\top \begin{bmatrix} 1/p_1 \dots & 0 \\ \ddots & \\ 0 \dots & 1/p_S \end{bmatrix} \mathbf{I}_S \begin{bmatrix} d_1 \dots & 0 \\ \ddots & \\ 0 \dots & d_S \end{bmatrix} \begin{bmatrix} \sigma'(t_1) \dots & 0 \\ \ddots & \\ 0 \dots & \sigma'(t_S) \end{bmatrix} =$$

$$= \left[-\frac{y_1 - \sigma(t_1)}{S} \dots -\frac{y_S - \sigma(t_S)}{S}\right] = \left[-\frac{y_1 - \sigma(X_1\mathbf{w} + b)}{S} \dots -\frac{y_S - \sigma(X_S\mathbf{w} + b)}{S}\right]$$

$$\frac{\partial h}{\partial b} = \frac{\partial h}{\partial t}\frac{\partial t}{\partial b_S}\frac{\partial b_S}{\partial b} = \left[-\frac{y_1 - \sigma(X_1\mathbf{w} + b)}{S} \quad \dots \quad -\frac{y_S - \sigma(X_S\mathbf{w} + b)}{S}\right] \mathbf{I}_S \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = -\frac{1}{S}\sum_{i=1}^{S}\big(y_i - \sigma(X_i\mathbf{w} + b)\big)$$

$$\frac{\partial h}{\partial w} = \frac{\partial h}{\partial t}\frac{\partial t}{\partial u}\frac{\partial u}{\partial w} = \left[-\frac{y_1 - \sigma(X_1\mathbf{w} + b)}{S} \dots -\frac{y_S - \sigma(X_S\mathbf{w} + b)}{S}\right] \mathbf{I}_S X =$$

$$= -\frac{1}{S}\sum_{i=1}^{S}(y_i - \sigma(X_i\mathbf{w} + b))X_i$$

Заключение

- Изкуствените невронни мрежи са сравнително универсален модел за апроксимация на сложни функции.
- Чрез дълбоки (многослойни) архитектури значително се разширява изразителната им способност.
- Обучението на дълбоките невронни мрежи се извършва предимно с градиентни методи.
- Ефективен и автоматичен метод за намиране на градиентите на сложни функции е метода известен като Backpropagation
- Чрез Backpropagation градиентите на сложни функции се изчисляват автоматично, ефективно и точно.
- Този метод се ползва на практика във всички софтуерни системи за дълбоко обучение — pytorch, tensorflow, CNTK, …