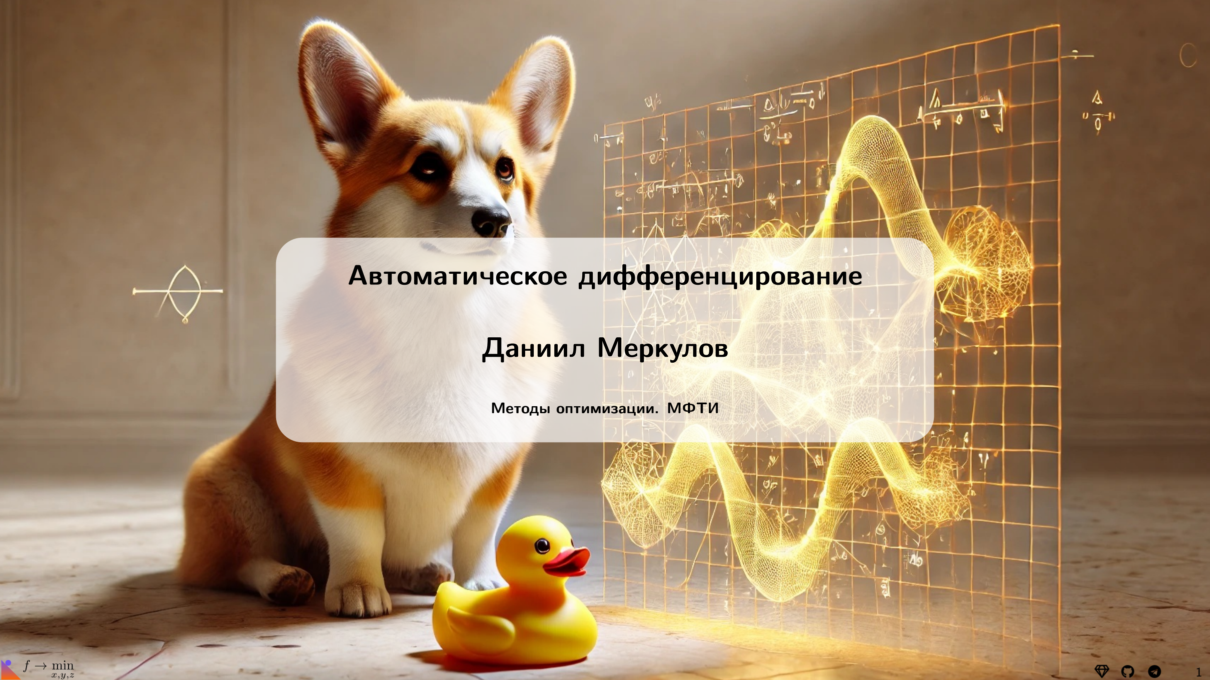


Автоматическое дифференцирование

Даниил Меркулов

Методы оптимизации. МФТИ

Матрично-векторное дифференцирование

Градиент

Пусть $f(x) : \mathbb{R}^n \to \mathbb{R}$, тогда вектор, который содержит все первые частные производные:

$$
\nabla f(x) = \dfrac{df}{dx} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}
$$

Градиент

Пусть $f(x) : \mathbb{R}^n \to \mathbb{R}$, тогда вектор, который содержит все первые частные производные:

$$\nabla f(x) = \dfrac{df}{dx} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

называется градиентом функции $f(x)$. Этот вектор указывает направление наискорейшего возрастания. Таким образом, вектор $-\nabla f(x)$ указывает направление наискорейшего убывания функции в точке. Кроме того, вектор градиента всегда ортогонален линии уровня в точке.

Example

Для функции $f(x, y) = x^2 + y^2$ градиент равен:

$$\nabla f(x, y) = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

Он указывает направление наискорейшего возрастания функции.

Question

Как связана норма градиента с крутизной функции?

Гессиан

Пусть $f(x) : \mathbb{R}^n \to \mathbb{R}$, тогда матрица, содержащая все вторые частные производные:

$$f''(x) = \nabla^2 f(x) = \dfrac{\partial^2 f}{\partial x_i \partial x_j} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Гессиан

Пусть $f(x) : \mathbb{R}^n \to \mathbb{R}$, тогда матрица, содержащая все вторые частные производные:

$$f''(x) = \nabla^2 f(x) = \dfrac{\partial^2 f}{\partial x_i \partial x_j} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Гессиан может быть тензором: $(f(x) : \mathbb{R}^n \to \mathbb{R}^m)$ Таким образом, это просто трехмерный тензор, каждый срез которого это гессиан соответствующей скалярной функции $(\nabla^2 f_1(x), \dots, \nabla^2 f_m(x))$.

Example

Для функции $f(x, y) = x^2 + y^2$ гессиан равен:

$$H_f(x, y) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Эта матрица содержит информацию о кривизне функции в разных направлениях.

Question

Как можно использовать гессиан для определения выпуклости или вогнутости функции?

Теорема Шварца

Пусть $f : \mathbb{R}^n \to \mathbb{R}$ - функция. Если смешанные частные производные $\frac{\partial^2 f}{\partial x_i \partial x_j}$ и $\frac{\partial^2 f}{\partial x_j \partial x_i}$ непрерывны на открытом множестве, содержащем точку a, то они равны в точке a. То есть,

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(a) = \frac{\partial^2 f}{\partial x_j \partial x_i}(a)$$

Теорема Шварца

Пусть $f : \mathbb{R}^n \to \mathbb{R}$ - функция. Если смешанные частные производные $\frac{\partial^2 f}{\partial x_i \partial x_j}$ и $\frac{\partial^2 f}{\partial x_j \partial x_i}$ непрерывны на открытом множестве, содержащем точку a, то они равны в точке a. То есть,

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(a) = \frac{\partial^2 f}{\partial x_j \partial x_i}(a)$$

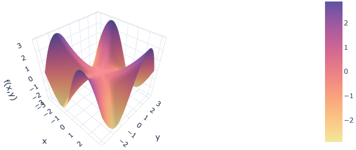
Согласно данной теореме, если смешанные частные производные непрерывны на открытом множестве, то гессиан симметричен. То есть,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i} \qquad \nabla^2 f(x) = \left(\nabla^2 f(x)\right)^T$$

Эта симметричность упрощает вычисления и анализ, связанные с гессианом в различных приложениях, особенно в оптимизации.

Контрпример Шварца

$$f(x,y) = \begin{cases} \frac{xy(x^2-y^2)}{x^2+y^2} & \text{для } (x,\, y) \neq (0,\, 0), \\ 0 & \text{для } (x, y) = (0, 0). \end{cases}$$



Можно проверить, что $\frac{\partial^2 f}{\partial x \partial y}(0,0) \neq \frac{\partial^2 f}{\partial y \partial x}(0,0)$, хотя смешанные частные производные существуют, и в каждой другой точке симметричность выполняется.

Якобиан

Обобщением понятия градиента на случай многомерной функции $f(x) : \mathbb{R}^n \to \mathbb{R}^m$ является следующая матрица:

$$
J_f = f'(x) = \frac{df}{dx^T} = \begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\
\frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n}
\end{pmatrix}
$$

Она содержит информацию о скорости изменения функции по отношению к ее входу.

Question

Можно ли связать эти три определения выше (градиент, якобиан, и гессиан) с помощью одного утверждения?

Example

Для функции

$$
f(x, y) = \begin{bmatrix} x + y \\ x - y \end{bmatrix},
$$

Якобиан равен:

$$
J_f(x, y) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
$$

Question

Как матрица Якоби связана с градиентом для скалярных функций?

Итог

$$
f(x) : X \to Y; \qquad \frac{\partial f(x)}{\partial x} \in G
$$

X	Y	G	Name
$\mathbb{R}$	$\mathbb{R}$	$\mathbb{R}$	$f'(x)$ (производная)
$\mathbb{R}^n$	$\mathbb{R}$	$\mathbb{R}^n$	$\dfrac{\partial f}{\partial x_i}$ (градиент)
$\mathbb{R}^n$	$\mathbb{R}^m$	$\mathbb{R}^{n \times m}$	$\dfrac{\partial f_i}{\partial x_j}$ (якобиан)
$\mathbb{R}^{m \times n}$	$\mathbb{R}$	$\mathbb{R}^{m \times n}$	$\dfrac{\partial f}{\partial x_{ij}}$

Аппроксимация Тейлора первого порядка

Аппроксимация Тейлора первого порядка, также известная как линейное приближение, строится вблизи некоторой точки x_0. Если $f : \mathbb{R}^n \to \mathbb{R}$ - дифференцируемая функция, то ее аппроксимация первого порядка задается следующим образом:

$$f_{x_0}^I(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$$

где:

- $f(x_0)$ - значение функции в точке x_0.

Аппроксимация Тейлора первого порядка

Аппроксимация Тейлора первого порядка, также известная как линейное приближение, строится вблизи некоторой точки x_0. Если $f : \mathbb{R}^n \to \mathbb{R}$ - дифференцируемая функция, то ее аппроксимация первого порядка задается следующим образом:

$$
f_{x_0}^I(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)
$$

где:

- $f(x_0)$ - значение функции в точке x_0.
- $\nabla f(x_0)$ - градиент функции в точке x_0.

Аппроксимация Тейлора первого порядка

Аппроксимация Тейлора первого порядка, также известная как линейное приближение, строится вблизи некоторой точки x_0. Если $f : \mathbb{R}^n \to \mathbb{R}$ - дифференцируемая функция, то ее аппроксимация первого порядка задается следующим образом:

$$f_{x_0}^I(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$$

где:

- $f(x_0)$ - значение функции в точке x_0.
- $\nabla f(x_0)$ - градиент функции в точке x_0.

Аппроксимация Тейлора первого порядка

Аппроксимация Тейлора первого порядка, также известная как линейное приближение, строится вблизи некоторой точки x_0. Если $f : \mathbb{R}^n \to \mathbb{R}$ - дифференцируемая функция, то ее аппроксимация первого порядка задается следующим образом:

$$f_{x_0}^I(x) = f(x_0) + \nabla f(x_0)^T (x - x_0)$$

где:

- $f(x_0)$ - значение функции в точке x_0.
- $\nabla f(x_0)$ - градиент функции в точке x_0.

Часто для упрощения теоретического анализа в некоторых методах заменяют функцию вблизи некоторой точки на её аппроксимацию

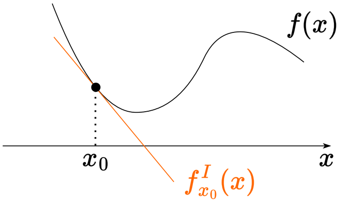


Рис. 1: Аппроксимация Тейлора первого порядка в окрестности точки x_0

Аппроксимация Тейлора второго порядка

Аппроксимация Тейлора второго порядка, также известная как квадратичное приближение, использует информацию о кривизне функции. Для дважды дифференцируемой функции $f : \mathbb{R}^n \to \mathbb{R}$, ее аппроксимация второго порядка, строящаяся вблизи некоторой точки x_0, задается следующим образом:

$$f_{x_0}^{II}(x) = f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0)$$

Где $\nabla^2 f(x_0)$ - гессиан функции f в точке x_0.

Аппроксимация Тейлора второго порядка

Аппроксимация Тейлора второго порядка, также известная как квадратичное приближение, использует информацию о кривизне функции. Для дважды дифференцируемой функции $f : \mathbb{R}^n \to \mathbb{R}$, ее аппроксимация второго порядка, строящаяся вблизи некоторой точки x_0, задается следующим образом:

$$f_{x_0}^{II}(x) = f(x_0) + \nabla f(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T\nabla^2 f(x_0)(x - x_0)$$

Где $\nabla^2 f(x_0)$ - гессиан функции f в точке x_0.
Когда линейного приближения функции не достаточно, можно рассмотреть замену $f(x)$ на $f_{x_0}^{II}(x)$ в окрестности точки x_0. В общем, приближения Тейлора дают нам способ локально аппроксимировать функции.
Аппроксимация первого порядка определяется градиентом функции в точке, т.е. нормалью к касательной гиперплоскости. А аппроксимация второго порядка представляет из себя параболу. Эти приближения особенно полезны в оптимизации и численных методах, потому что они предоставляют простой способ работы со сложными функциями.

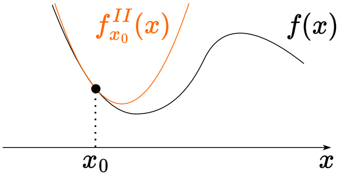


Рис. 2: Аппроксимация Тейлора второго порядка в окрестности точки x_0

Дифференциалы

Theorem

Пусть $x \in S$ - внутренняя точка множества S, и пусть $D : U \to V$ - линейный оператор. Мы говорим, что функция f дифференцируема в точке x с производной D, если для всех достаточно малых $h \in U$ выполняется следующее разложение:

$$f(x + h) = f(x) + D[h] + o(\|h\|)$$

Если для любого линейного оператора $D : U \to V$ функция f не дифференцируема в точке x с производной D, то мы говорим, что f не дифференцируема в точке x.

Дифференциалы

После получения дифференциальной записи df мы можем получить градиент, используя следующую формулу:

$$df(x) = \langle \nabla f(x), dx \rangle$$

Дифференциалы

После получения дифференциальной записи df мы можем получить градиент, используя следующую формулу:

$$df(x) = \langle \nabla f(x), dx \rangle$$

Далее, если у нас есть дифференциал в такой форме и мы хотим вычислить вторую производную матричной/векторной функции, мы рассматриваем "старый" dx как константу dx_1, затем вычисляем $d(df) = d^2 f(x)$

$$d^2 f(x) = \langle \nabla^2 f(x) dx_1, dx \rangle = \langle H_f(x) dx_1, dx \rangle$$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$
- $d(X^T) = (dX)^T$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X+Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$
- $d\left(\frac{X}{\phi}\right) = \frac{\phi dX - (d\phi)X}{\phi^2}$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$
- $d\left(\frac{X}{\phi}\right) = \frac{\phi dX - (d\phi)X}{\phi^2}$
- $d\left(\det X\right) = \det X \langle X^{-T}, dX\rangle$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X+Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$
- $d\left(\frac{X}{\phi}\right) = \frac{\phi dX - (d\phi)X}{\phi^2}$
- $d\left(\det X\right) = \det X \langle X^{-T}, dX\rangle$
- $d\left(\text{tr } X\right) = \langle I, dX\rangle$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X + Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$

- $d\left(\dfrac{X}{\phi}\right) = \dfrac{\phi dX - (d\phi)X}{\phi^2}$
- $d\left(\det X\right) = \det X\langle X^{-T}, dX\rangle$
- $d\left(\text{tr } X\right) = \langle I, dX\rangle$
- $df(g(x)) = \dfrac{df}{dg} \cdot dg(x)$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X+Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$
- $d\left(\dfrac{X}{\phi}\right) = \dfrac{\phi dX - (d\phi)X}{\phi^2}$
- $d\left(\det X\right) = \det X \langle X^{-T}, dX\rangle$
- $d\left(\text{tr } X\right) = \langle I, dX\rangle$
- $df(g(x)) = \dfrac{df}{dg} \cdot dg(x)$
- $H = (J(\nabla f))^T$

Свойства дифференциалов

Пусть A и B - постоянные матрицы, а X и Y - переменные (или матричные функции).

- $dA = 0$
- $d(\alpha X) = \alpha(dX)$
- $d(AXB) = A(dX)B$
- $d(X+Y) = dX + dY$
- $d(X^T) = (dX)^T$
- $d(XY) = (dX)Y + X(dY)$
- $d\langle X, Y\rangle = \langle dX, Y\rangle + \langle X, dY\rangle$
- $d\left(\dfrac{X}{\phi}\right) = \dfrac{\phi dX - (d\phi)X}{\phi^2}$
- $d\left(\det X\right) = \det X \langle X^{-T}, dX\rangle$
- $d\left(\text{tr } X\right) = \langle I, dX\rangle$
- $df(g(x)) = \dfrac{df}{dg} \cdot dg(x)$
- $H = (J(\nabla f))^T$
- $d(X^{-1}) = -X^{-1}(dX)X^{-1}$

Матричное дифференцирование. Пример 1

Example

Найти $df, \nabla f(x)$, если $f(x) = \langle x, Ax\rangle - b^T x + c$.

Матричное дифференцирование. Пример 2

Example

Найти $df, \nabla f(x)$, если $f(x) = \ln\langle x, Ax\rangle$.

Матричное дифференцирование. Пример 2

Example

Найти $df, \nabla f(x)$, если $f(x) = \ln\langle x, Ax\rangle$.

1. Заметим, что A должна быть положительно определенной, потому что $\langle x, Ax\rangle$ аргумент логарифма и для любого x формула должна быть положительной. Таким образом, $A \in \mathbb{S}^n_{++}$ Давайте сначала найдем дифференциал:

$$
df = d\left(\ln\langle x, Ax\rangle\right) = \frac{d\left(\langle x, Ax\rangle\right)}{\langle x, Ax\rangle} = \frac{\langle dx, Ax\rangle + \langle x, d(Ax)\rangle}{\langle x, Ax\rangle} =
$$

$$
= \frac{\langle Ax, dx\rangle + \langle x, Adx\rangle}{\langle x, Ax\rangle} = \frac{\langle Ax, dx\rangle + \langle A^T x, dx\rangle}{\langle x, Ax\rangle} = \frac{\langle (A + A^T)x, dx\rangle}{\langle x, Ax\rangle}
$$

Матричное дифференцирование. Пример 2

Example

Найти $df, \nabla f(x)$, если $f(x) = \ln\langle x, Ax\rangle$.

1. Заметим, что A должна быть положительно определенной, потому что $\langle x, Ax\rangle$ аргумент логарифма и для любого x формула должна быть положительной. Таким образом, $A \in \mathbb{S}^n_{++}$ Давайте сначала найдем дифференциал:

$$df = d\left(\ln\langle x, Ax\rangle\right) = \dfrac{d\left(\langle x, Ax\rangle\right)}{\langle x, Ax\rangle} = \dfrac{\langle dx, Ax\rangle + \langle x, d(Ax)\rangle}{\langle x, Ax\rangle} =$$
$$= \dfrac{\langle Ax, dx\rangle + \langle x, Adx\rangle}{\langle x, Ax\rangle} = \dfrac{\langle Ax, dx\rangle + \langle A^T x, dx\rangle}{\langle x, Ax\rangle} = \dfrac{\langle (A + A^T)x, dx\rangle}{\langle x, Ax\rangle}$$

2. Наша основная цель - получить форму $df = \langle \cdot, dx\rangle$

$$df = \left\langle \dfrac{2Ax}{\langle x, Ax\rangle}, dx\right\rangle$$

Таким образом, градиент равен $\nabla f(x) = \dfrac{2Ax}{\langle x, Ax\rangle}$

Матричное дифференцирование. Пример 3

Example

Найти $df, \nabla f(X)$, если $f(X) = \langle S, X \rangle - \log\det X$.

Автоматическое дифференцирование

Рис. 3: Когда понял идею

Рис. 4: Это не автоград

Задача

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

Задача

Предположим, что мы хотим решить следующую задачу:

$$
L(w) \to \min_{w \in \mathbb{R}^d}
$$

- Такие задачи обычно возникают в машинном обучении, когда нам нужно найти подходящие параметры w модели (например, обучить нейронную сеть).

Задача

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

- Такие задачи обычно возникают в машинном обучении, когда нам нужно найти подходящие параметры w модели (например, обучить нейронную сеть).
- Существуют разные методы решения этой задачи. Однако, размерность задач сегодня может достигать сотен миллиардов или даже триллионов переменных. Такие задачи очень тяжело решать без знания градиентов, то есть методами нулевого порядка.

Задача

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

- Такие задачи обычно возникают в машинном обучении, когда нам нужно найти подходящие параметры w модели (например, обучить нейронную сеть).
- Существуют разные методы решения этой задачи. Однако, размерность задач сегодня может достигать сотен миллиардов или даже триллионов переменных. Такие задачи очень тяжело решать без знания градиентов, то есть методами нулевого порядка.
- Поэтому было бы полезно уметь вычислять вектор градиента $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_d}\right)^T$.

Задача

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

- Такие задачи обычно возникают в машинном обучении, когда нам нужно найти подходящие параметры w модели (например, обучить нейронную сеть).
- Существуют разные методы решения этой задачи. Однако, размерность задач сегодня может достигать сотен миллиардов или даже триллионов переменных. Такие задачи очень тяжело решать без знания градиентов, то есть методами нулевого порядка.
- Поэтому было бы полезно уметь вычислять вектор градиента $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_d} \right)^T$.
- Обычно методы первого порядка работают лучше в больших задачах, в то время как методы второго порядка требуют слишком много памяти.

Пример: задача многомерного шкалирования

Предположим, что у нас есть матрица расстояний для N d-мерных объектов $D \in \mathbb{R}^{N \times N}$. Используя эту матрицу, мы хотим восстановить исходные координаты $W_i \in \mathbb{R}^d,\ i = 1, \ldots, N$.

Пример: задача многомерного шкалирования

Предположим, что у нас есть матрица расстояний для N d-мерных объектов $D \in \mathbb{R}^{N \times N}$. Используя эту матрицу, мы хотим восстановить исходные координаты $W_i \in \mathbb{R}^d, \; i = 1, \ldots, N$.

$$
L(W) = \sum_{i,j=1}^{N} \left(\|W_i - W_j\|_2^2 - D_{i,j} \right)^2 \to \min_{W \in \mathbb{R}^{N \times d}}
$$

Пример: задача многомерного шкалирования

Предположим, что у нас есть матрица расстояний для N d-мерных объектов $D \in \mathbb{R}^{N \times N}$. Используя эту матрицу, мы хотим восстановить исходные координаты $W_i \in \mathbb{R}^d$, $i = 1, \ldots, N$.

$$
L(W) = \sum_{i,j=1}^{N} \left(\|W_i - W_j\|_2^2 - D_{i,j}\right)^2 \to \min_{W \in \mathbb{R}^{N \times d}}
$$

Ссылка на визуализацию ♣, где можно увидеть, что безградиентные методы оптимизации решают эту задачу намного медленнее, особенно в пространствах большой размерности.

Question

Связано ли это с PCA?

Пример: многомерное масштабирование

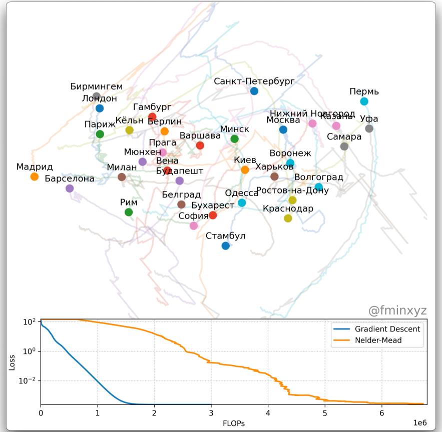


Рис. 5: Ссылка на анимацию

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$
L(w) \to \min_{w \in \mathbb{R}^d}
$$

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

с помощью алгоритма градиентного спуска (GD):

$$w_{k+1} = w_k - \alpha_k \nabla_w L(w_k)$$

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$
L(w) \to \min_{w \in \mathbb{R}^d}
$$

с помощью алгоритма градиентного спуска (GD):

$$
w_{k+1} = w_k - \alpha_k \nabla_w L(w_k)
$$

Можно ли заменить $\nabla_w L(w_k)$ используя только информацию нулевого порядка?

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$
L(w) \to \min_{w \in \mathbb{R}^d}
$$

с помощью алгоритма градиентного спуска (GD):

$$
w_{k+1} = w_k - \alpha_k \nabla_w L(w_k)
$$

Можно ли заменить $\nabla_w L(w_k)$ используя только информацию нулевого порядка?
Да, но за определенную цену.

[1] рекомендуется хорошая презентация о безградиентных методах

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

с помощью алгоритма градиентного спуска (GD):

$$w_{k+1} = w_k - \alpha_k \nabla_w L(w_k)$$

Можно ли заменить $\nabla_w L(w_k)$ используя только информацию нулевого порядка?
Да, но за определенную цену.
Рассмотрим двухточечную оценку градиента[1] G:

$$G = d \frac{L(w + \varepsilon v) - L(w - \varepsilon v)}{2\varepsilon} v,$$

где v сферически симметричен.

[1] рекомендуется хорошая презентация о безградиентных методах

Пример: градиентный спуск без градиента

Предположим, что мы хотим решить следующую задачу:

$$L(w) \to \min_{w \in \mathbb{R}^d}$$

с помощью алгоритма градиентного спуска (GD):

$$w_{k+1} = w_k - \alpha_k \nabla_w L(w_k)$$

Можно ли заменить $\nabla_w L(w_k)$ используя только информацию нулевого порядка?
Да, но за определенную цену.
Рассмотрим двухточечную оценку градиента[1] G:

$$G = d\frac{L(w + \varepsilon v) - L(w - \varepsilon v)}{2\varepsilon}v,$$

где v сферически симметричен.

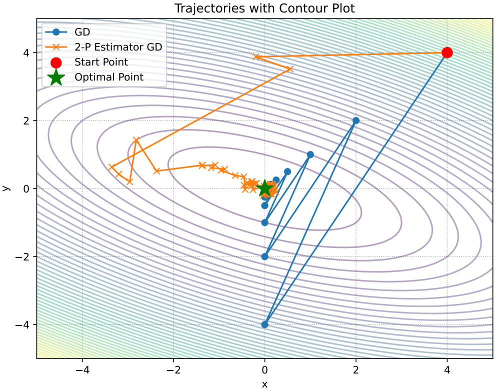


Рис. 6: “Иллюстрация двухточечной оценки градиентного спуска”

[1] рекомендуется хорошая презентация о безградиентных методах

Пример: конечные разности

$$w_{k+1} = w_k - \alpha_k G$$

Пример: конечные разности

$$
w_{k+1} = w_k - \alpha_k G
$$

Также рассмотрим идею конечных разностей:

$$
G = \sum_{i=1}^{d} \frac{L(w + \varepsilon e_i) - L(w - \varepsilon e_i)}{2\varepsilon} e_i
$$

Открыть в Colab ♣

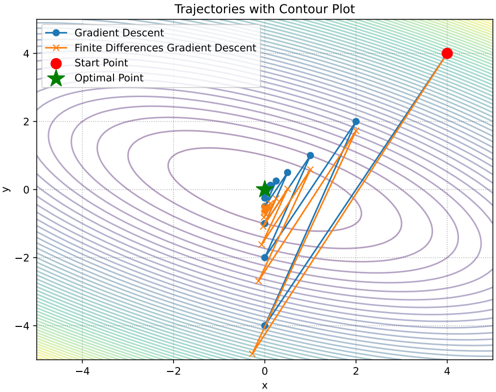


Рис. 7: "Иллюстрация работы метода оценки градиента с помощью метода конечных разностей"

Проклятие размерности для методов нулевого порядка [2]

$$\min_{x \in \mathbb{R}^n} f(x)$$

Проклятие размерности для методов нулевого порядка [2]

$$\min_{x \in \mathbb{R}^n} f(x)$$

GD: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ Zero order GD: $x_{k+1} = x_k - \alpha_k G$,

где G - оценка градиента 2-точечная или многоточечная.

[2] Оптимальные скорости для нулевого порядка выпуклой оптимизации: сила двух оценок функции

Проклятие размерности для методов нулевого порядка [2]

$$\min_{x\in\mathbb{R}^n} f(x)$$

GD: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ Zero order GD: $x_{k+1} = x_k - \alpha_k G,$

где G - оценка градиента 2-точечная или многоточечная.

	$f(x)$ - гладкая	$f(x)$ - гладкая и выпуклая	$f(x)$ - гладкая и сильно выпуклая
GD	$\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{1}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{1}{k}\right)$	$\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
GD нулевого порядка	$\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{n}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{n}{k}\right)$	$\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{nL}\right)^k\right)$

Для 2-точечных оценок, мы не можем сделать зависимость лучше, чем от $\sqrt{n}$!

[2] Оптимальные скорости для нулевого порядка выпуклой оптимизации: сила двух оценок функции

Конечные разности

Наивный подход к получению приблизительных значений градиентов - это подход **конечных разностей**. Для каждой координаты, можно вычислить приближенное значение частной производной:

$$
\frac{\partial L}{\partial w_k}(w) \approx \frac{L(w + \varepsilon e_k) - L(w)}{\varepsilon}, \quad e_k = (0, \dots, \underset{k}{1}, \dots, 0)
$$

[3] Linnainmaa S. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish), Univ. Helsinki, 1970.

Конечные разности

Наивный подход к получению приблизительных значений градиентов - это подход **конечных разностей**. Для каждой координаты, можно вычислить приближенное значение частной производной:

$$\frac{\partial L}{\partial w_k}(w) \approx \frac{L(w+\varepsilon e_k) - L(w)}{\varepsilon}, \quad e_k = (0, \ldots, \underset{k}{1}, \ldots, 0)$$

Question

Если время, необходимое для одного вычисления $L(w)$ равно T, то какое время необходимо для вычисления $\nabla_w L$ с этим подходом?

Конечные разности

Наивный подход к получению приблизительных значений градиентов - это подход **конечных разностей**. Для каждой координаты, можно вычислить приближенное значение частной производной:

$$\frac{\partial L}{\partial w_k}(w) \approx \frac{L(w + \varepsilon e_k) - L(w)}{\varepsilon}, \quad e_k = (0, \dots, \underset{k}{1}, \dots, 0)$$

Question

Если время, необходимое для одного вычисления $L(w)$ равно T, то какое время необходимо для вычисления $\nabla_w L$ с этим подходом?
Ответ $2dT$, что очень долго для больших задач. Кроме того, этот метод нестабилен, что означает, что нам придется выбирать между точностью и стабильностью.

Конечные разности

Наивный подход к получению приблизительных значений градиентов - это подход **конечных разностей**. Для каждой координаты, можно вычислить приближенное значение частной производной:

$$\frac{\partial L}{\partial w_k}(w) \approx \frac{L(w+\varepsilon e_k) - L(w)}{\varepsilon}, \quad e_k = (0, \ldots, \underset{k}{1}, \ldots, 0)$$

Question

Если время, необходимое для одного вычисления $L(w)$ равно T, то какое время необходимо для вычисления $\nabla_w L$ с этим подходом?

Ответ $2dT$, что очень долго для больших задач. Кроме того, этот метод нестабилен, что означает, что нам придется выбирать между точностью и стабильностью.

Теорема

Существует алгоритм для вычисления $\nabla_w L$ за $\mathcal{O}(T)$. [3]

[3]Linnainmaa S. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish), Univ. Helsinki, 1970.

Прямой режим автоматического дифференцирования

Чтобы глубже понять идею автоматического дифференцирования, рассмотрим простую функцию для вычисления производных:

$$L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}$$

Прямой режим автоматического дифференцирования

Чтобы глубже понять идею автоматического дифференцирования, рассмотрим простую функцию для вычисления производных:

$$
L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}
$$

Давайте нарисуем *вычислительный граф* этой функции:

$$
L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}
$$

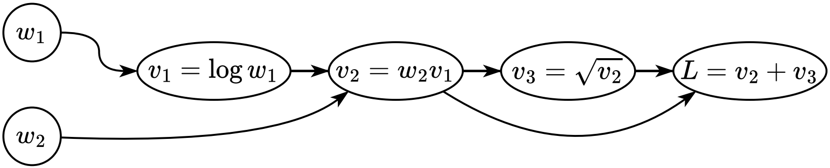


Рис. 8: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Прямой режим автоматического дифференцирования

Чтобы глубже понять идею автоматического дифференцирования, рассмотрим простую функцию для вычисления производных:

$$L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}$$

Давайте нарисуем *вычислительный граф* этой функции:

$$L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}$$

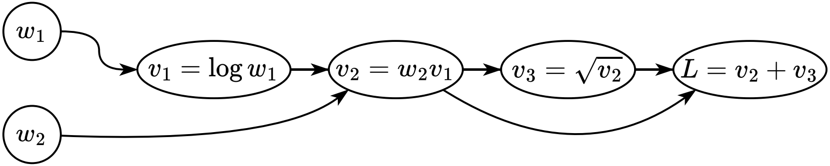


Рис. 8: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Давайте пойдем от начала графа к концу и вычислим производную $\dfrac{\partial L}{\partial w_1}$.

Прямой режим автоматического дифференцирования

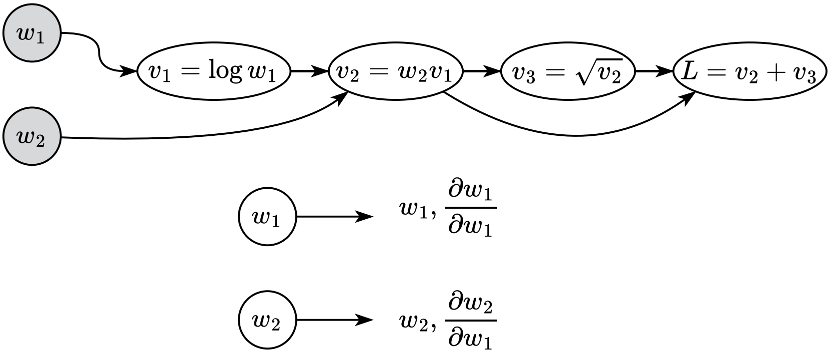


Рис. 9: Иллюстрация прямого режима автоматического дифференцирования

Функция

$w_1 = w_1, w_2 = w_2$

Прямой режим автоматического дифференцирования

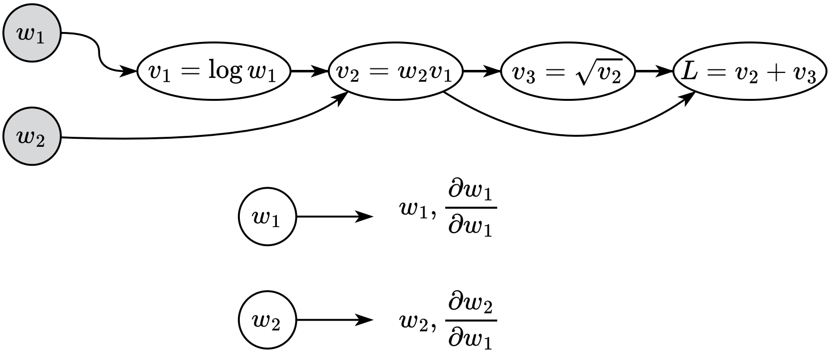


Рис. 9: Иллюстрация прямого режима автоматического дифференцирования

Функция

$w_1 = w_1, w_2 = w_2$

Производная

$$\frac{\partial w_1}{\partial w_1} = 1, \frac{\partial w_2}{\partial w_1} = 0$$

Прямой режим автоматического дифференцирования

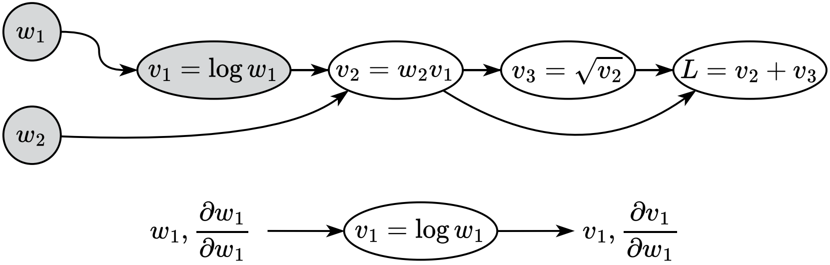


Рис. 10: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

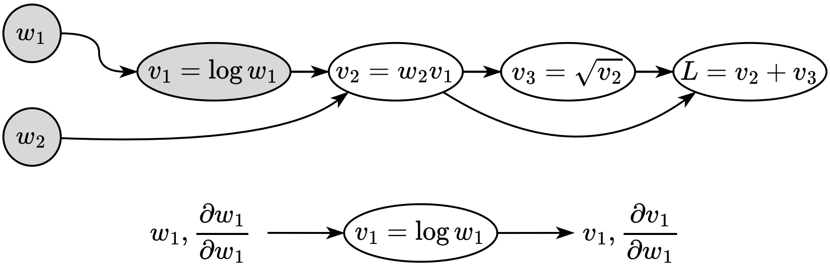


Рис. 10: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_1 = \log w_1$

Прямой режим автоматического дифференцирования

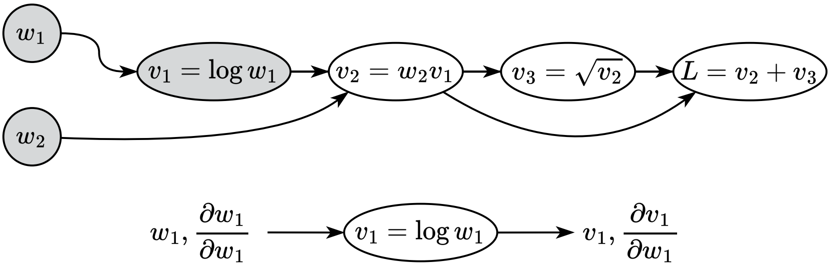


Рис. 10: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_1 = \log w_1$

Производная

$\frac{\partial v_1}{\partial w_1} = \frac{\partial v_1}{\partial w_1}\frac{\partial w_1}{\partial w_1} = \frac{1}{w_1} 1$

Прямой режим автоматического дифференцирования

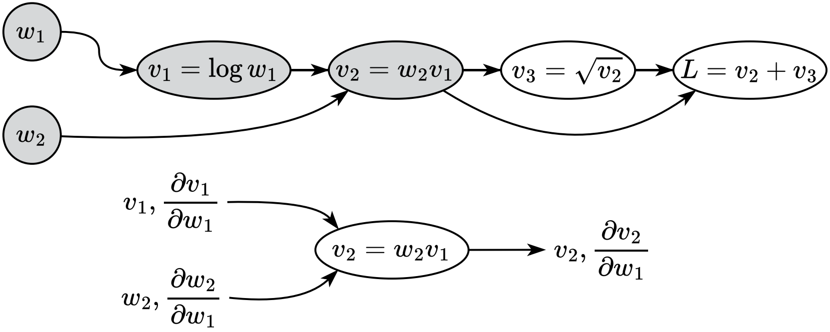


Рис. 11: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

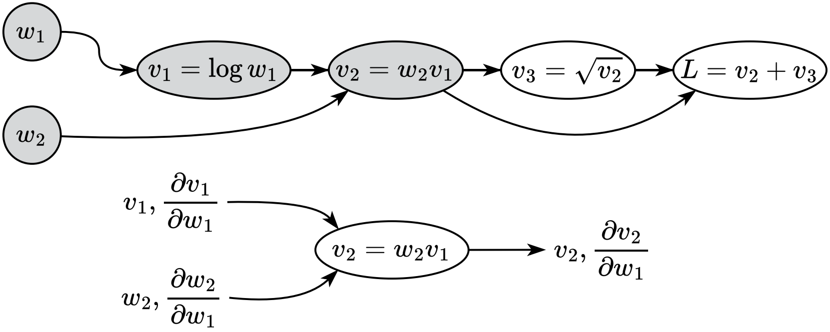


Рис. 11: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_2 = w_2 v_1$

Прямой режим автоматического дифференцирования

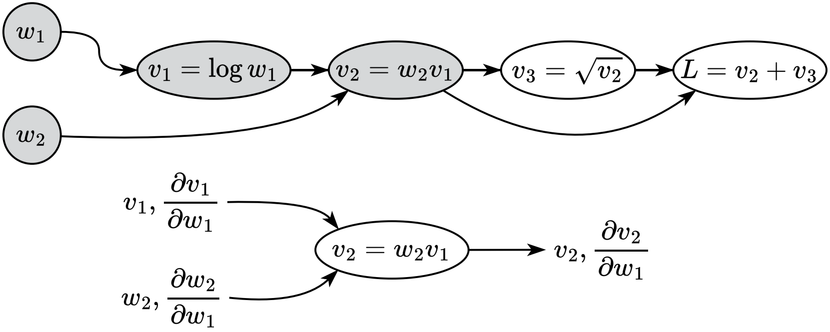


Рис. 11: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$v_2 = w_2 v_1$$

Производная

$$\frac{\partial v_2}{\partial w_1} = \frac{\partial v_2}{\partial v_1}\frac{\partial v_1}{\partial w_1} + \frac{\partial v_2}{\partial w_2}\frac{\partial w_2}{\partial w_1} = w_2\frac{\partial v_1}{\partial w_1} + v_1\frac{\partial w_2}{\partial w_1}$$

Прямой режим автоматического дифференцирования

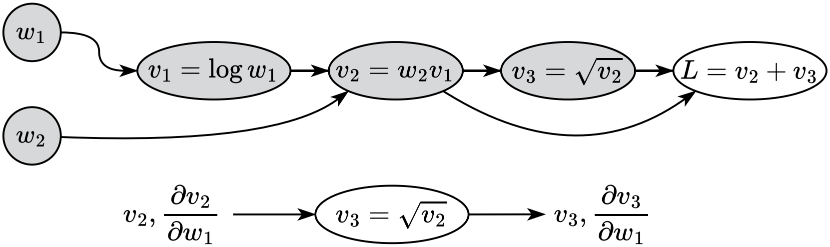


Рис. 12: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

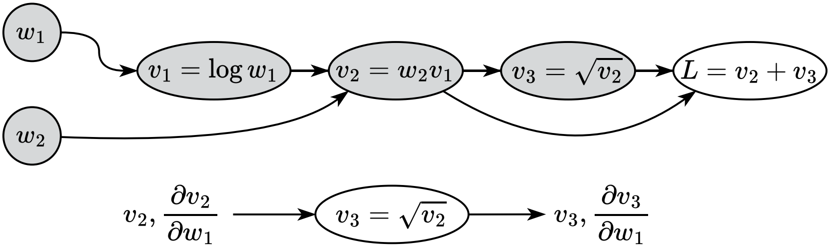


Рис. 12: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_3 = \sqrt{v_2}$

Прямой режим автоматического дифференцирования

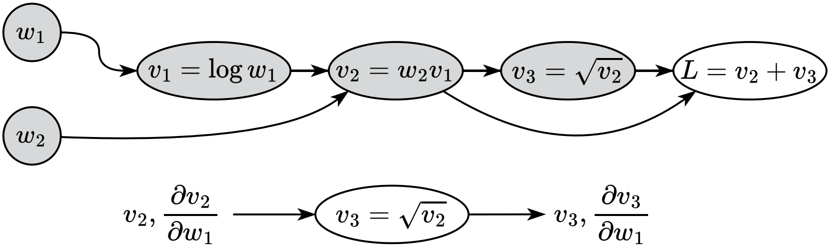


Рис. 12: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$v_3 = \sqrt{v_2}$$

Производная

$$\frac{\partial v_3}{\partial w_1} = \frac{\partial v_3}{\partial v_2}\frac{\partial v_2}{\partial w_1} = \frac{1}{2\sqrt{v_2}}\frac{\partial v_2}{\partial w_1}$$

Прямой режим автоматического дифференцирования

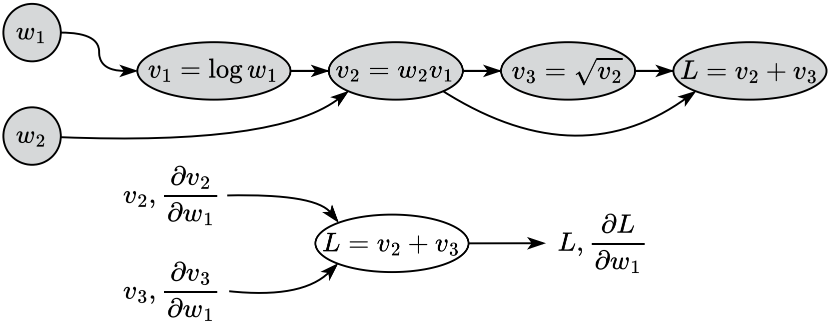


Рис. 13: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

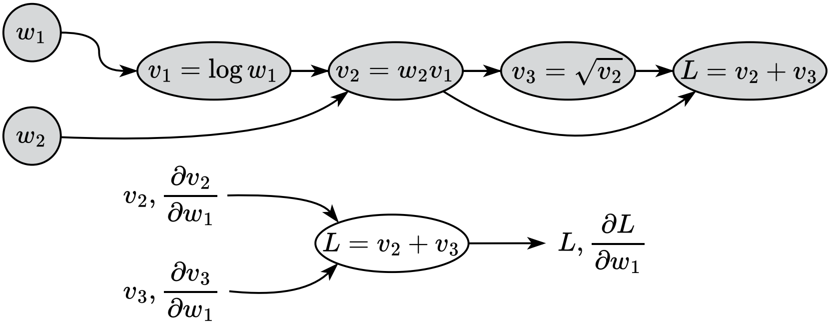


Рис. 13: Иллюстрация прямого режима автоматического дифференцирования

Функция

$L = v_2 + v_3$

Прямой режим автоматического дифференцирования

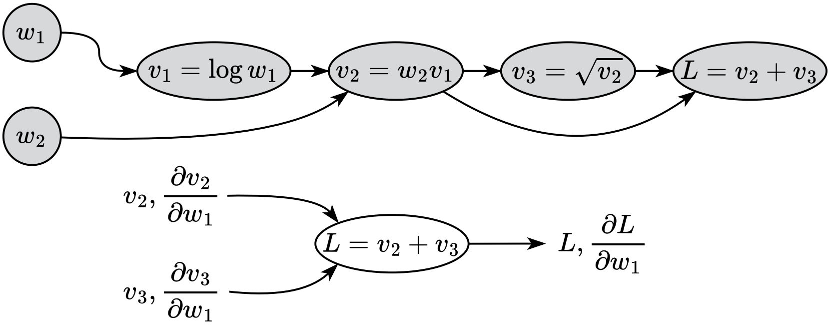


Рис. 13: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$L = v_2 + v_3$$

Производная

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial v_2}\frac{\partial v_2}{\partial w_1} + \frac{\partial L}{\partial v_3}\frac{\partial v_3}{\partial w_1} = 1\frac{\partial v_2}{\partial w_1} + 1\frac{\partial v_3}{\partial w_1}$$

Сделайте аналогичные вычисления для $\dfrac{\partial L}{\partial w_2}$

$$L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}$$

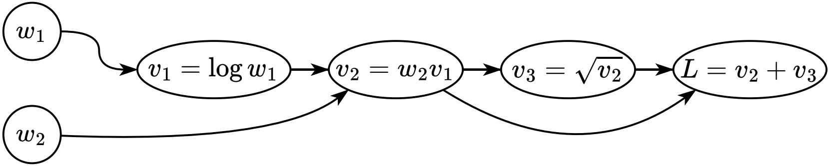


Рис. 14: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Пример прямого режима автоматического дифференцирования

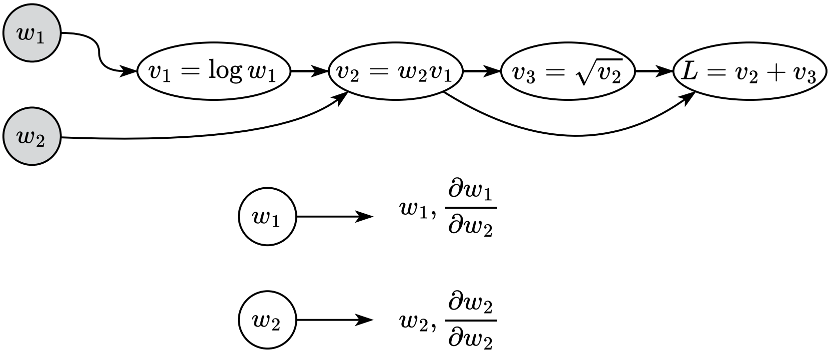


Рис. 15: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$w_1 = w_1, w_2 = w_2$$

Производная

$$\frac{\partial w_1}{\partial w_2} = 0, \frac{\partial w_2}{\partial w_2} = 1$$

Пример прямого режима автоматического дифференцирования

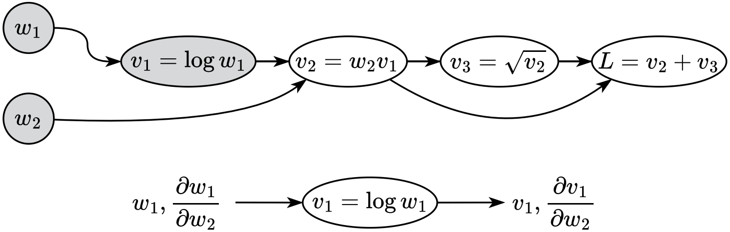


Рис. 16: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_1 = \log w_1$

Производная

$\frac{\partial v_1}{\partial w_2} = \frac{\partial v_1}{\partial w_1}\frac{\partial w_1}{\partial w_2} = \frac{1}{w_1} \cdot 0$

Пример прямого режима автоматического дифференцирования

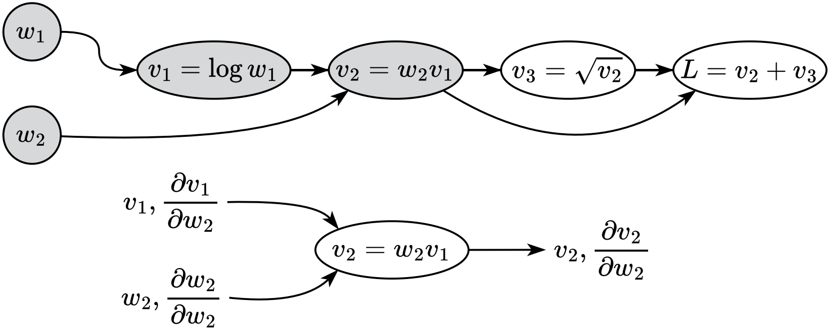


Рис. 17: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$v_2 = w_2 v_1$$

Производная

$$\frac{\partial v_2}{\partial w_2} = \frac{\partial v_2}{\partial v_1}\frac{\partial v_1}{\partial w_2} + \frac{\partial v_2}{\partial w_2}\frac{\partial w_2}{\partial w_2} = w_2 \frac{\partial v_1}{\partial w_2} + v_1 \frac{\partial w_2}{\partial w_2}$$

Пример прямого режима автоматического дифференцирования

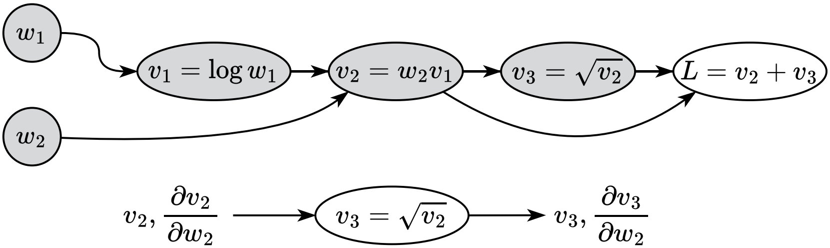


Рис. 18: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$v_3 = \sqrt{v_2}$$

Производная

$$\frac{\partial v_3}{\partial w_2} = \frac{\partial v_3}{\partial v_2}\frac{\partial v_2}{\partial w_2} = \frac{1}{2\sqrt{v_2}}\frac{\partial v_2}{\partial w_2}$$

Пример прямого режима автоматического дифференцирования

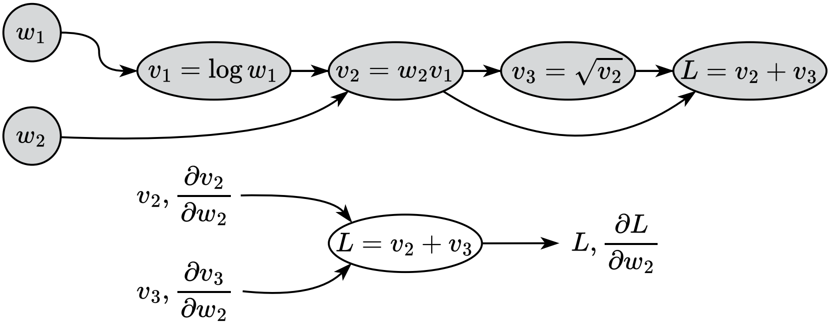


Рис. 19: Иллюстрация прямого режима автоматического дифференцирования

Функция

$L = v_2 + v_3$

Производная

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial v_2}\frac{\partial v_2}{\partial w_2} + \frac{\partial L}{\partial v_3}\frac{\partial v_3}{\partial w_2} = 1\frac{\partial v_2}{\partial w_2} + 1\frac{\partial v_3}{\partial w_2}$$

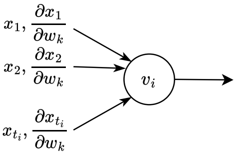
Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$
\overline{v_i} = \frac{\partial v_i}{\partial w_k}
$$



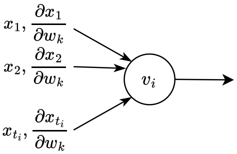
$$
v_i = v_i(x_1, \ldots, x_{t_i})
$$

$$
\frac{\partial v_i}{\partial w_k} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}
$$

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial v_i}{\partial w_k}$$



$$v_i = v_i(x_1, \ldots, x_{t_i})$$

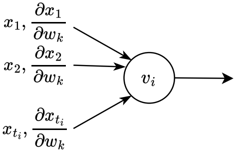
$$\frac{\partial v_i}{\partial w_k} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}$$

- Для $i = 1, \ldots, N$:

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial v_i}{\partial w_k}$$



$$v_i = v_i(x_1, \ldots, x_{t_i})$$

$$\frac{\partial v_i}{\partial w_k} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}$$

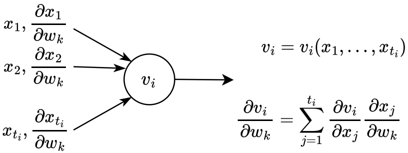
- Для $i = 1, \ldots, N$:
    - Вычислить v_i как функцию его предков $x_1, \ldots, x_{t_i}$:

$$v_i = v_i(x_1, \ldots, x_{t_i})$$

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial v_i}{\partial w_k}$$



- Для $i = 1, \ldots, N$:
  - Вычислить v_i как функцию его предков $x_1, \ldots, x_{t_i}$:

    $$v_i = v_i(x_1, \ldots, x_{t_i})$$

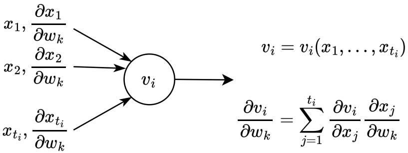
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции:

    $$\overline{v_i} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}$$

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial v_i}{\partial w_k}$$



- Для $i = 1, \ldots, N$:
  - Вычислить v_i как функцию его предков $x_1, \ldots, x_{t_i}$:

$$v_i = v_i(x_1, \ldots, x_{t_i})$$

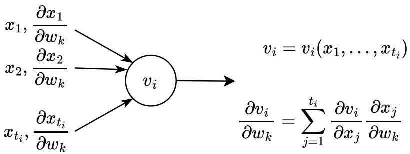
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции:

$$\overline{v_i} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}$$

Алгоритм прямого режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по некоторой входной переменной w_k, т.е. $\dfrac{\partial v_N}{\partial w_k}$. Эта идея предполагает распространение градиента по входной переменной от начала к концу, поэтому мы можем ввести обозначение:

$$
\overline{v_i} = \frac{\partial v_i}{\partial w_k}
$$



- Для $i = 1, \ldots, N$:
  - Вычислить v_i как функцию его предков $x_1, \ldots, x_{t_i}$:

$$
v_i = v_i(x_1, \ldots, x_{t_i})
$$

  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции:

$$
\overline{v_i} = \sum_{j=1}^{t_i} \frac{\partial v_i}{\partial x_j} \frac{\partial x_j}{\partial w_k}
$$

Обратите внимание, что этот подход не требует хранения всех промежуточных вычислений, но можно видеть, что для вычисления производной $\dfrac{\partial L}{\partial w_k}$ нам нужно $\mathcal{O}(T)$ операций. Это означает, что для всего градиента, нам нужно $d\mathcal{O}(T)$ операций, что то же самое, что и для конечных разностей, но теперь у нас нет проблем со стабильностью или неточностями(формулы выше точны).


There is another

Обратный режим автоматического дифференцирования

Мы рассмотрим ту же функцию с вычислительным графом:

$$
L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}
$$

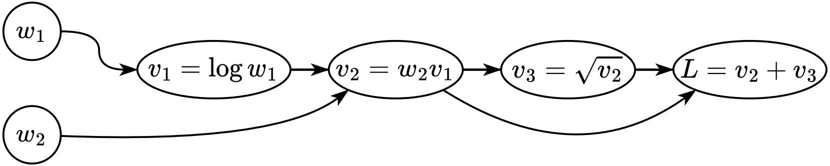


Рис. 20: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Обратный режим автоматического дифференцирования

Мы рассмотрим ту же функцию с вычислительным графом:

$$
L(w_1, w_2) = w_2 \log w_1 + \sqrt{w_2 \log w_1}
$$

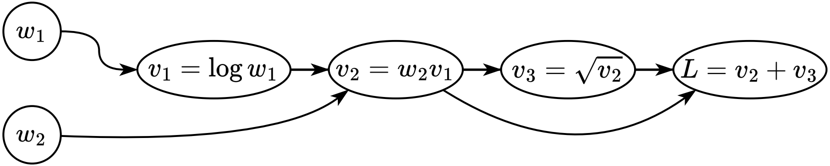


Рис. 20: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Предположим, что у нас есть некоторые значения параметров w_1, w_2 и мы уже выполнили прямой проход (т.е. вычисление значений всех промежуточных узлов вычислительного графа). Предположим также, что мы как-то сохранили все промежуточные значения v_i. Давайте пойдем от конца графа к началу и вычислим производные $\dfrac{\partial L}{\partial w_1}, \dfrac{\partial L}{\partial w_2}$:

Пример обратного режима автоматического дифференцирования

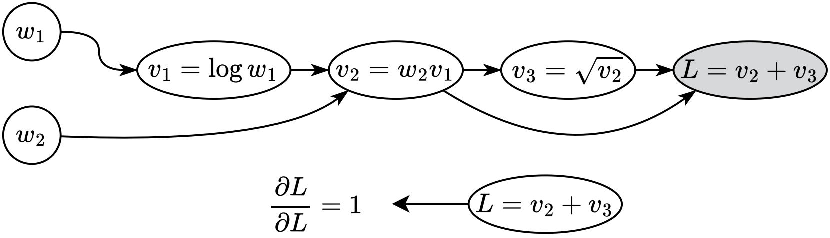


Рис. 21: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

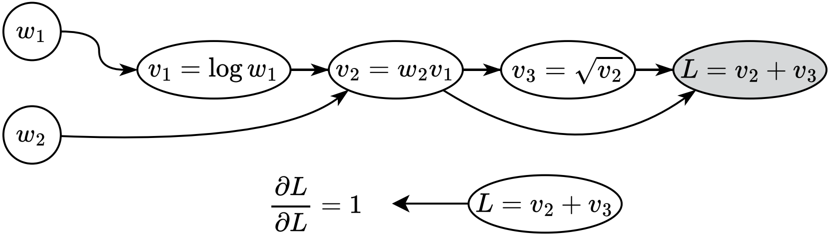


Рис. 21: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

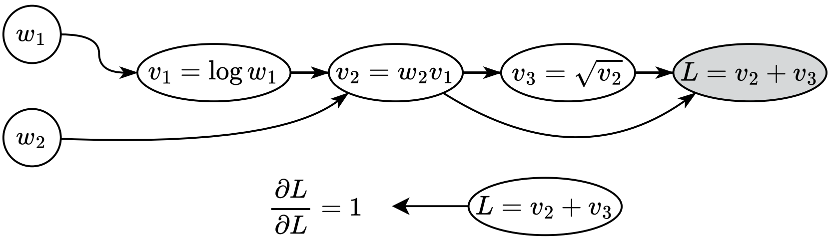


Рис. 21: Иллюстрация обратного режима автоматического дифференцирования

Производные

$$
\frac{\partial L}{\partial L} = 1
$$

Пример обратного режима автоматического дифференцирования

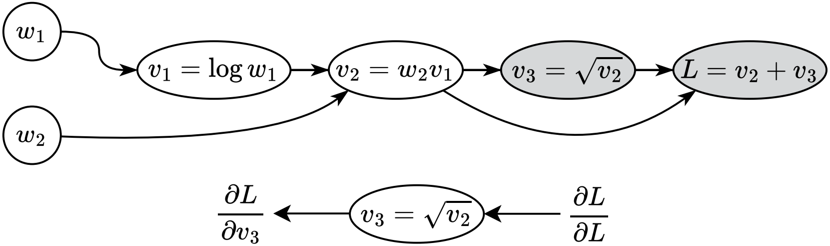


Рис. 22: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

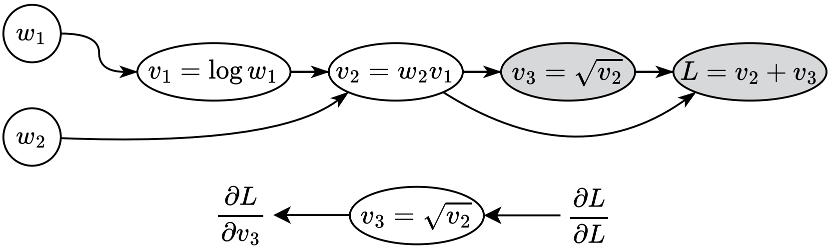


Рис. 22: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

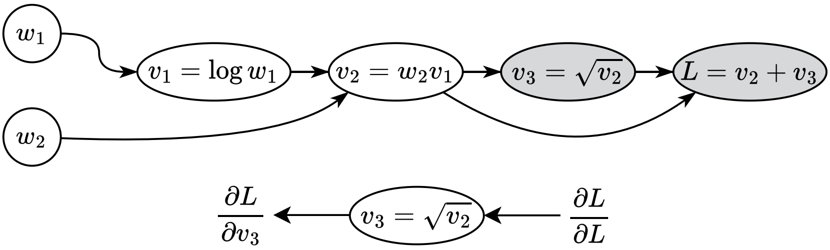


Рис. 22: Иллюстрация обратного режима автоматического дифференцирования

Производные

$$\frac{\partial L}{\partial v_3} = \frac{\partial L}{\partial L}\frac{\partial L}{\partial v_3} \quad = \frac{\partial L}{\partial L}1$$

Пример обратного режима автоматического дифференцирования

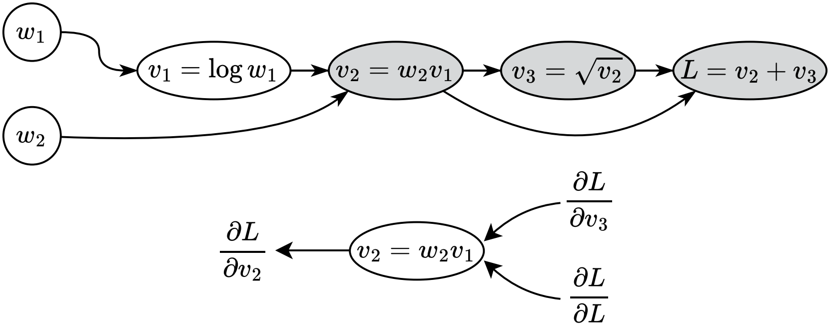


Рис. 23: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

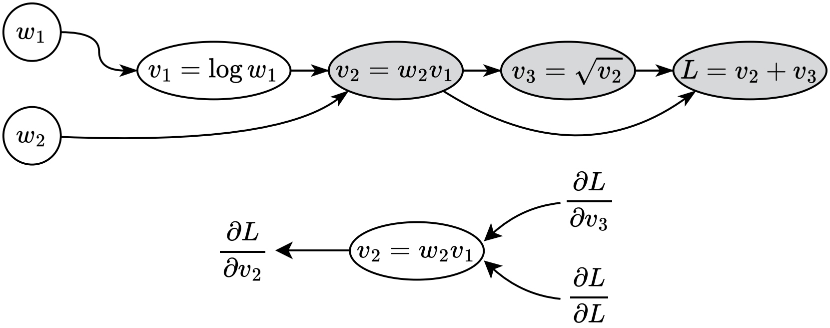


Рис. 23: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

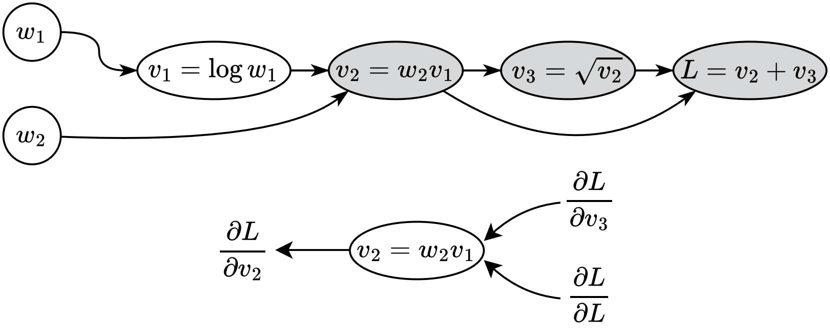


Рис. 23: Иллюстрация обратного режима автоматического дифференцирования

Производные

$$
\frac{\partial L}{\partial v_2} = \frac{\partial L}{\partial v_3}\frac{\partial v_3}{\partial v_2} + \frac{\partial L}{\partial L}\frac{\partial L}{\partial v_2} \quad = \frac{\partial L}{\partial v_3}\frac{1}{2\sqrt{v_2}} + \frac{\partial L}{\partial L}1
$$

Пример обратного режима автоматического дифференцирования

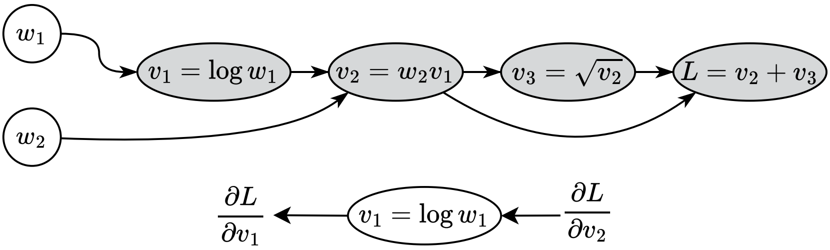


Рис. 24: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

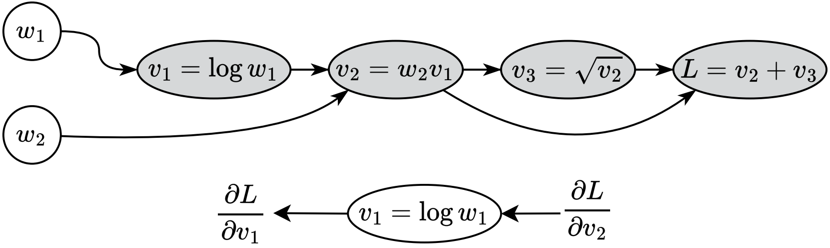


Рис. 24: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

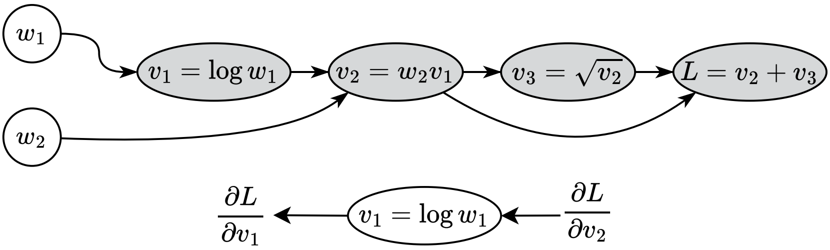


Рис. 24: Иллюстрация обратного режима автоматического дифференцирования

Производные

$$
\frac{\partial L}{\partial v_1} = \frac{\partial L}{\partial v_2}\frac{\partial v_2}{\partial v_1} \quad = \frac{\partial L}{\partial v_2} w_2
$$

Пример обратного режима автоматического дифференцирования

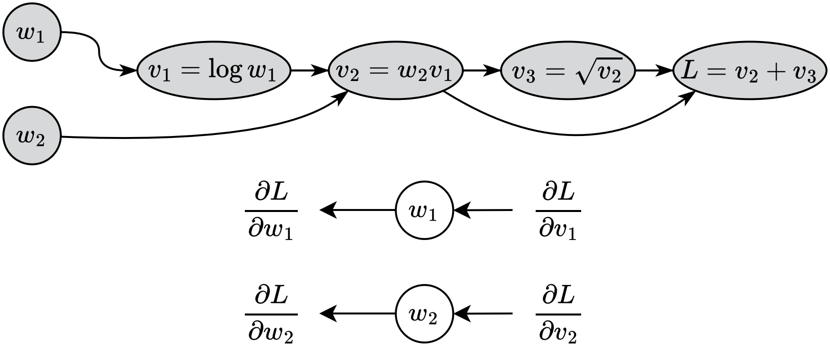


Рис. 25: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

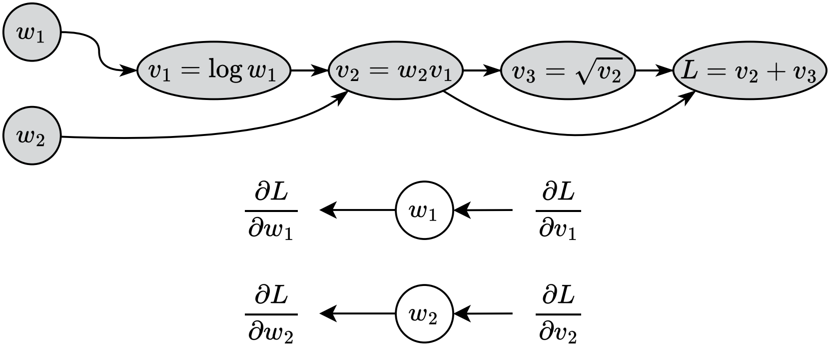


Рис. 25: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

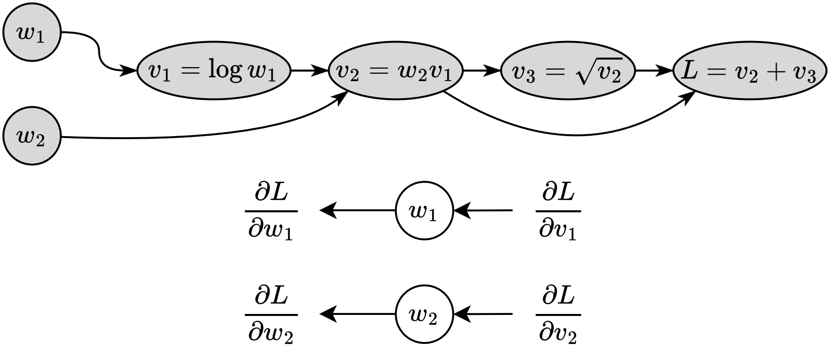


Рис. 25: Иллюстрация обратного режима автоматического дифференцирования

Производные

$$
\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial v_1}\frac{\partial v_1}{\partial w_1} = \frac{\partial L}{\partial v_1}\frac{1}{w_1} \qquad \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial v_2}\frac{\partial v_2}{\partial w_2} = \frac{\partial L}{\partial v_1} v_1
$$

Обратный режим автоматического дифференцирования

Question

Обратите внимание, что для того же количества вычислений, что и в прямом режиме, мы получаем полный вектор градиента $\nabla_w L$. Какова стоимость ускорения?

Обратный режим автоматического дифференцирования

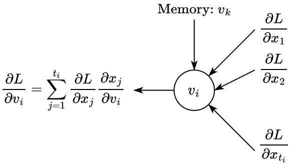
Question

Обратите внимание, что для того же количества вычислений, что и в прямом режиме, мы получаем полный вектор градиента $\nabla_w L$. Какова стоимость ускорения?
Ответ Обратите внимание, что для использования обратного режима AD вам нужно хранить все промежуточные вычисления из прямого прохода. Эта проблема может быть частично решена с помощью чекпоинтинга, при котором мы сохраняем только часть промежуточных значений, а остальные пересчитываем заново по мере необходимости. Это позволяет значительно уменьшить объём требуемой памяти при обучении больших моделей машинного обучения.

Алгоритм обратного режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по всем входным переменным w, т.е. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \ldots, \frac{\partial v_N}{\partial w_d} \right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial L}{\partial v_i} = \frac{\partial v_N}{\partial v_i}$$

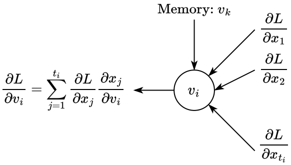


- **ПРЯМОЙ ПРОХОД**
  Для $i = 1, \ldots, N$:

Алгоритм обратного режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по всем входным переменным w, т.е. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \ldots, \frac{\partial v_N}{\partial w_d} \right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial L}{\partial v_i} = \frac{\partial v_N}{\partial v_i}$$



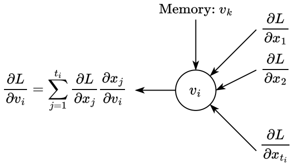
- **ПРЯМОЙ ПРОХОД**

  Для $i = 1, \ldots, N$:
  - Вычислить и сохранить значения v_i как функцию его предков

Алгоритм обратного режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по всем входным переменным w, т.е. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \ldots, \frac{\partial v_N}{\partial w_d}\right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial L}{\partial v_i} = \frac{\partial v_N}{\partial v_i}$$



- **ПРЯМОЙ ПРОХОД**

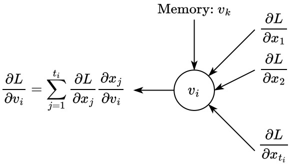
  Для $i = 1, \ldots, N$:
  - Вычислить и сохранить значения v_i как функцию его предков
- **ОБРАТНЫЙ ПРОХОД**

  Для $i = N, \ldots, 1$:

Алгоритм обратного режима автоматического дифференцирования

Предположим, что у нас есть вычислительный граф $v_i, i \in [1; N]$. Наша цель - вычислить производную выхода этого графа по всем входным переменным w, т.е. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \ldots, \frac{\partial v_N}{\partial w_d} \right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

$$\overline{v_i} = \frac{\partial L}{\partial v_i} = \frac{\partial v_N}{\partial v_i}$$



- **ПРЯМОЙ ПРОХОД**
  Для $i = 1, \ldots, N$:
  - Вычислить и сохранить значения v_i как функцию его предков
- **ОБРАТНЫЙ ПРОХОД**
  Для $i = N, \ldots, 1$:
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции и информацию от всех потомков (выходов):

$$\overline{v_i} = \frac{\partial L}{\partial v_i} = \sum_{j=1}^{t_i} \frac{\partial L}{\partial x_j} \frac{\partial x_j}{\partial v_i}$$

Choose your fighter

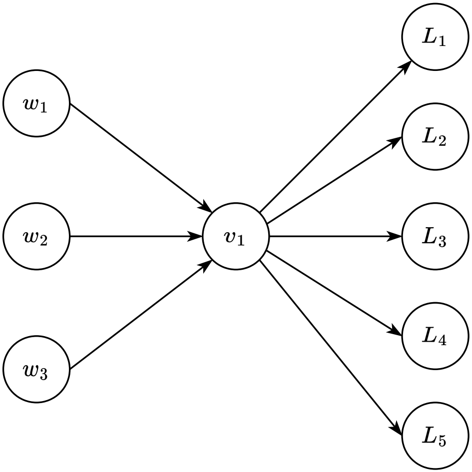


Рис. 26: Какой режим вы бы выбрали для вычисления градиентов?

Question

Какой из режимов AD вы бы выбрали (прямой/обратный) для следующего вычислительного графа арифметических операций? Предположим, что вам нужно вычислить якобиан $J = \left\{ \frac{\partial L_i}{\partial w_j} \right\}_{i,j}$

Choose your fighter

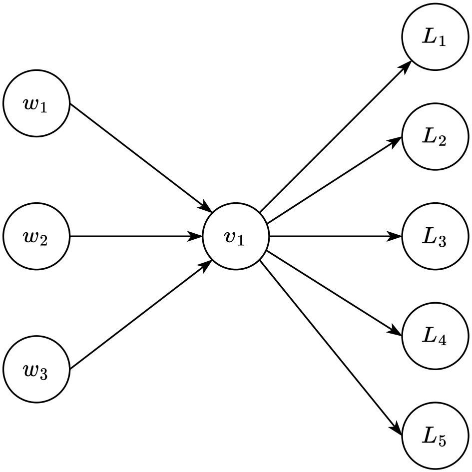


Рис. 26: Какой режим вы бы выбрали для вычисления градиентов?

Question

Какой из режимов AD вы бы выбрали (прямой/обратный) для следующего вычислительного графа арифметических операций? Предположим, что вам нужно вычислить якобиан $J = \left\{ \dfrac{\partial L_i}{\partial w_j} \right\}_{i,j}$

Ответ Обратите внимание, что время вычислений в обратном режиме пропорционально количеству выходов, тогда как время работы прямого режима пропорционально количеству входов. Поэтому было бы хорошей идеей рассмотреть прямой режим AD.

Choose your fighter

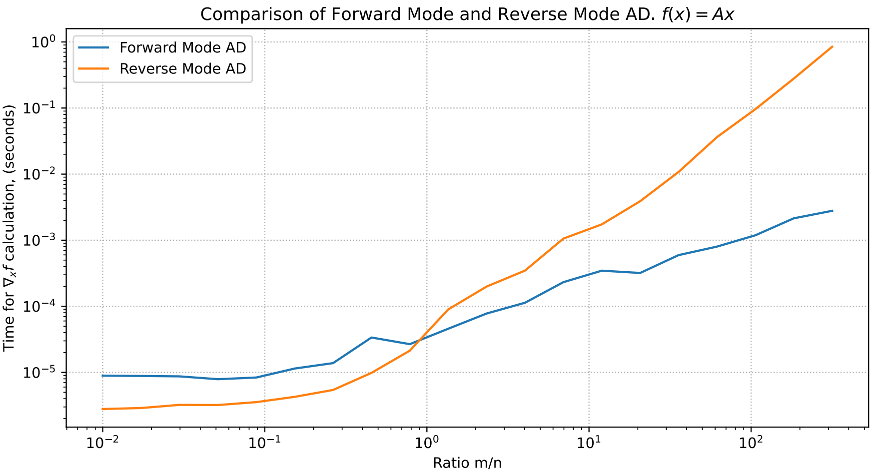


Рис. 27: ♣ График иллюстрирует идею выбора между режимами автоматического дифференцирования. Размерность входа $n = 100$ фиксирована, измерено время вычисления якобиана в зависимости от соотношения размерностей выхода и входа для разных размерностей выхода m.

Choose your fighter

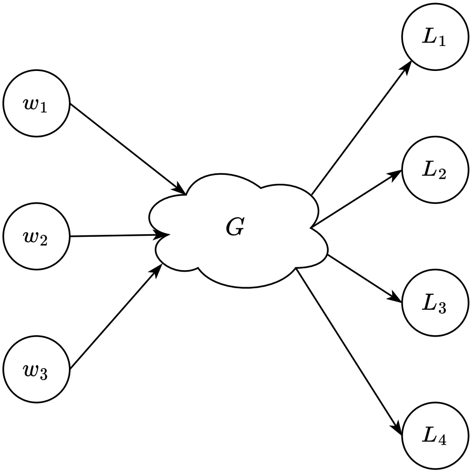


Рис. 28: Какой режим вы бы выбрали для вычисления градиентов?

Question

Какой из режимов AD вы бы выбрали (прямой/обратный) для следующего вычислительного графа арифметических операций? Предположим, что вам нужно вычислить якобиан $J = \left\{ \dfrac{\partial L_i}{\partial w_j} \right\}_{i,j}$. Обратите внимание, что G - это произвольный вычислительный граф

Choose your fighter

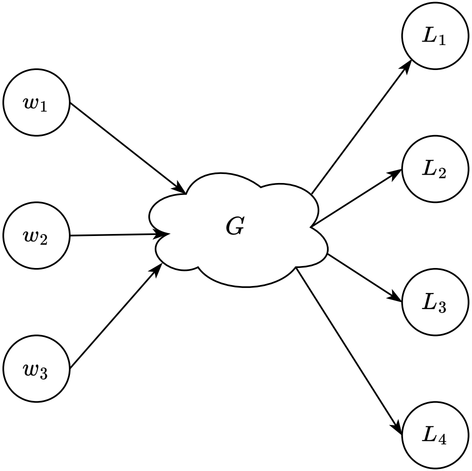


Рис. 28: Какой режим вы бы выбрали для вычисления градиентов?

Question

Какой из режимов AD вы бы выбрали (прямой/обратный) для следующего вычислительного графа арифметических операций? Предположим, что вам нужно вычислить якобиан $J = \left\{ \dfrac{\partial L_i}{\partial w_j} \right\}_{i,j}$. Обратите внимание, что G - это произвольный вычислительный граф

Ответ В общем случае невозможно ответить без некоторого знания о конкретной структуре графа G. Следует отметить, что существуют продвинутые подходы, смешивающие прямой и обратный режим AD в зависимости от конкретной структуры графа G.

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x

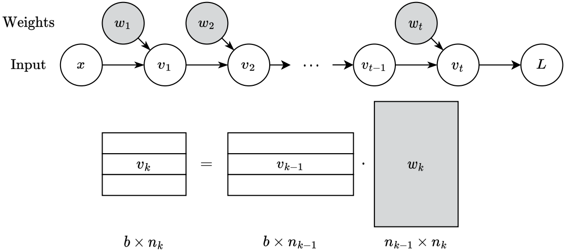


Рис. 29: Архитектура прямого распространения нейронной сети

ОБРАТНЫЙ ПРОХОД

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \dots, t-1, t$:

ОБРАТНЫЙ ПРОХОД

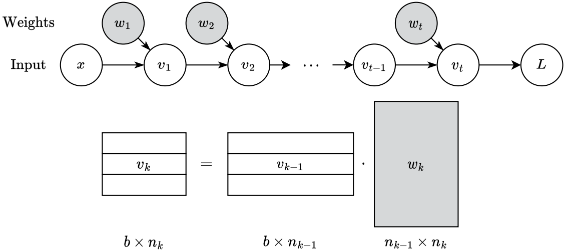


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \dots, t-1, t$:
  - $v_k = \sigma(v_{k-1} w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.

ОБРАТНЫЙ ПРОХОД

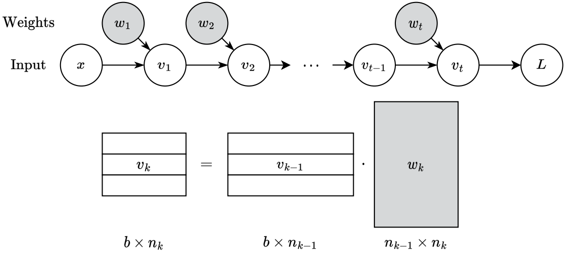


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \dots, t-1, t$:
  - $v_k = \sigma(v_{k-1} w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.
- $L = L(v_t)$ - вычислить функцию потерь.

ОБРАТНЫЙ ПРОХОД

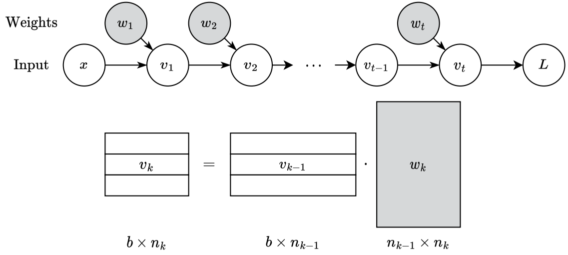


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \ldots, t-1, t$:
    - $v_k = \sigma(v_{k-1} w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.
- $L = L(v_t)$ - вычислить функцию потерь.

ОБРАТНЫЙ ПРОХОД

- $v_{t+1} = L, \dfrac{\partial L}{\partial L} = 1$

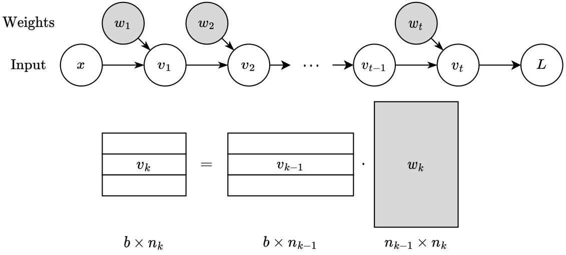


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \ldots, t-1, t$:
  - $v_k = \sigma(v_{k-1}w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.
- $L = L(v_t)$ - вычислить функцию потерь.

ОБРАТНЫЙ ПРОХОД

- $v_{t+1} = L, \dfrac{\partial L}{\partial L} = 1$
- Для $k = t, t-1, \ldots, 1$:

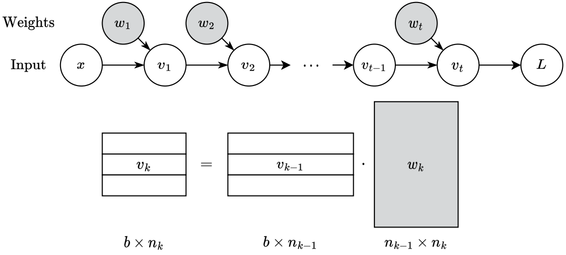


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \dots, t-1, t$:
  - $v_k = \sigma(v_{k-1} w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.
- $L = L(v_t)$ - вычислить функцию потерь.

ОБРАТНЫЙ ПРОХОД

- $v_{t+1} = L, \dfrac{\partial L}{\partial L} = 1$
- Для $k = t, t-1, \dots, 1$:
  - $$\underset{b \times n_k}{\frac{\partial L}{\partial v_k}} = \underset{b \times n_{k+1}}{\frac{\partial L}{\partial v_{k+1}}} \underset{n_{k+1} \times n_k}{\frac{\partial v_{k+1}}{\partial v_k}}$$

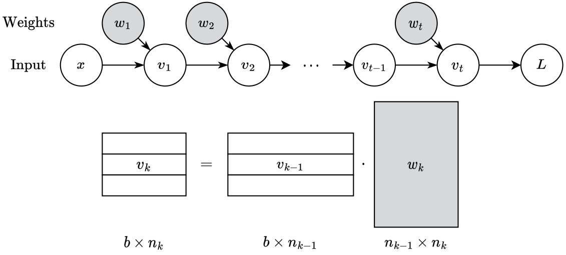


Рис. 29: Архитектура прямого распространения нейронной сети

Архитектура прямого распространения

ПРЯМОЙ ПРОХОД

- $v_0 = x$ на вход обычно подаётся батч данных x
- Для $k = 1, \dots, t-1, t$:
  - $v_k = \sigma(v_{k-1} w_k)$. Обратите внимание, что на практике, данные имеют размерность $x \in \mathbb{R}^{b \times d}$, где b - размер батча (для одного объекта из выборки $b = 1$). В то время как матрица весов w_k k слоя имеет размер $n_{k-1} \times n_k$, где n_k - размер внутреннего представления данных.
- $L = L(v_t)$ - вычислить функцию потерь.

ОБРАТНЫЙ ПРОХОД

- $v_{t+1} = L, \dfrac{\partial L}{\partial L} = 1$
- Для $k = t, t-1, \dots, 1$:
  - $\underset{b \times n_k}{\dfrac{\partial L}{\partial v_k}} = \underset{b \times n_{k+1}}{\dfrac{\partial L}{\partial v_{k+1}}} \underset{n_{k+1} \times n_k}{\dfrac{\partial v_{k+1}}{\partial v_k}}$
  - $\underset{b \times n_{k-1} \cdot n_k}{\dfrac{\partial L}{\partial w_k}} = \underset{b \times n_{k+1}}{\dfrac{\partial L}{\partial v_{k+1}}} \cdot \underset{n_{k+1} \times n_{k-1} \cdot n_k}{\dfrac{\partial v_{k+1}}{\partial w_k}}$

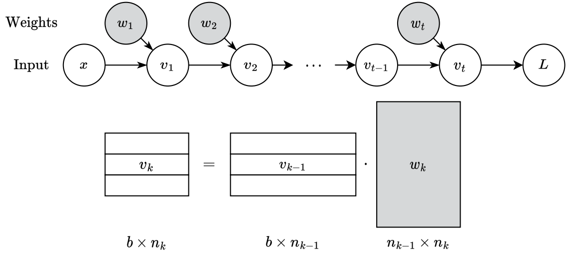


Рис. 29: Архитектура прямого распространения нейронной сети

Произведение Гессиана на вектор без вычисления самого Гессиана

Когда вам нужна некоторая информация о кривизне функции, обычно вам нужно работать с гессианом. Однако, это трудно делать, когда размерность задачи велика. Для скалярной функции $f : \mathbb{R}^n \to \mathbb{R}$, гессиан в точке $x \in \mathbb{R}^n$ записывается как $\nabla^2 f(x)$. Тогда произведение вектора на гессиан можно записать как

Произведение Гессиана на вектор без вычисления самого Гессиана

Когда вам нужна некоторая информация о кривизне функции, обычно вам нужно работать с гессианом. Однако, это трудно делать, когда размерность задачи велика. Для скалярной функции $f : \mathbb{R}^n \to \mathbb{R}$, гессиан в точке $x \in \mathbb{R}^n$ записывается как $\nabla^2 f(x)$. Тогда произведение вектора на гессиан можно записать как

$$
v \mapsto \nabla^2 f(x) \cdot v
$$

Произведение Гессиана на вектор без вычисления самого Гессиана

Когда вам нужна некоторая информация о кривизне функции, обычно вам нужно работать с гессианом. Однако, это трудно делать, когда размерность задачи велика. Для скалярной функции $f : \mathbb{R}^n \to \mathbb{R}$, гессиан в точке $x \in \mathbb{R}^n$ записывается как $\nabla^2 f(x)$. Тогда произведение вектора на гессиан можно записать как

$$
v \mapsto \nabla^2 f(x) \cdot v
$$

для любого вектора $v \in \mathbb{R}^n$. Мы можем использовать тождество

$$
\nabla^2 f(x) v = \nabla [x \mapsto \nabla f(x)^T \cdot v] = \nabla g(x),
$$

где $g(x) = \nabla f(x)^T \cdot v$ - новая функция, которая скалярно умножает градиент f в x на вектор v.

Произведение Гессиана на вектор без вычисления самого Гессиана

Когда вам нужна некоторая информация о кривизне функции, обычно вам нужно работать с гессианом. Однако, это трудно делать, когда размерность задачи велика. Для скалярной функции $f : \mathbb{R}^n \to \mathbb{R}$, гессиан в точке $x \in \mathbb{R}^n$ записывается как $\nabla^2 f(x)$. Тогда произведение вектора на гессиан можно записать как

$$v \mapsto \nabla^2 f(x) \cdot v$$

для любого вектора $v \in \mathbb{R}^n$. Мы можем использовать тождество

$$\nabla^2 f(x) v = \nabla [x \mapsto \nabla f(x)^T \cdot v] = \nabla g(x),$$

где $g(x) = \nabla f(x)^T \cdot v$ - новая функция, которая скалярно умножает градиент f в x на вектор v.

```python
import jax.numpy as jnp

def hvp(f, x, v):
    return grad(lambda x: jnp.vdot(grad(f)(x), v))(x)
```

Динамика обучения нейронной сети через спектр Гессиана и hvp [4]

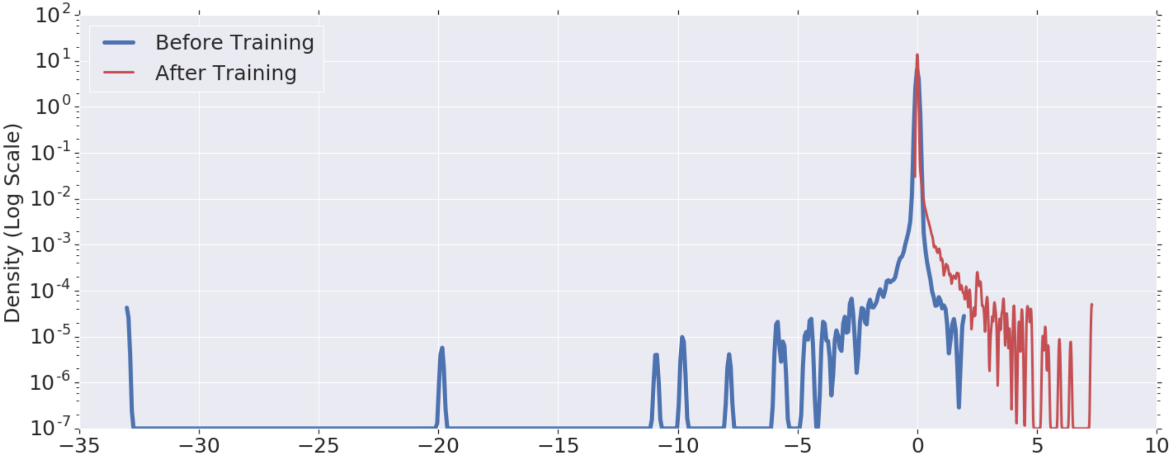


Рис. 30: Большие по модулю отрицательные собственные значения гессиана исчезли после обучения ResNet-32

[4] Некоторые исследования в оптимизации нейронных сетей через спектр собственных значений Гессиана

Идея Хадчинсона для оценки следа матрицы [5]

Метод Хатчинсона позволяет оценить след гессиана с помощью операций вычисления умножения гессиана на произвольный вектор:

Пусть $X \in \mathbb{R}^{d \times d}$ и $v \in \mathbb{R}^d$ - случайный вектор такой, что $\mathbb{E}[vv^T] = I$. Тогда,

$$
\mathrm{Tr}(X) = \mathbb{E}[v^T X v] = \frac{1}{V} \sum_{i=1}^{V} v_i^T X v_i.
$$

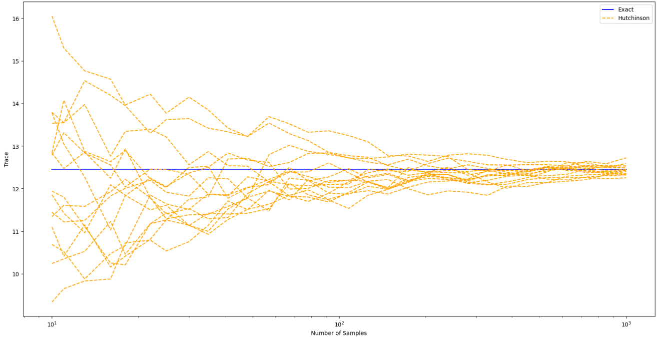


Рис. 31: Источник

[5] A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines - M.F. Hutchinson, 1990

Чекпоинтинг

Анимация вышеуказанных подходов

Пример использования контрольных точек градиента

[6] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Чекпоинтинг

Анимация вышеуказанных подходов

Пример использования контрольных точек градиента

В качестве примера рассмотрим обучение **GPT-2**[6]:

- Активации в простом режиме могут занимать гораздо больше памяти: для последовательности длиной 1K и размера батча 32, 60 GB нужно для хранения всех промежуточных активаций.

[6] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Чекпоинтинг

Анимация вышеуказанных подходов

Пример использования контрольных точек градиента

В качестве примера рассмотрим обучение **GPT-2**[6]:

- Активации в простом режиме могут занимать гораздо больше памяти: для последовательности длиной 1K и размера батча 32, 60 GB нужно для хранения всех промежуточных активаций.
- Чекпоинтинг может снизить потребление до 8 GB, пересчитывая их (33% дополнительных вычислений)

[6] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Чем автоматическое дифференцирование (AD) не является:

- AD не является методом конечных разностей

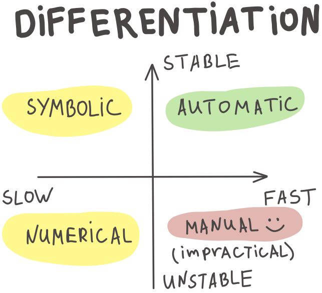


Рис. 32: Различные подходы для взятия производных

Чем автоматическое дифференцирование (AD) не является:

- AD не является методом конечных разностей
- AD не является символьным вычислением производных

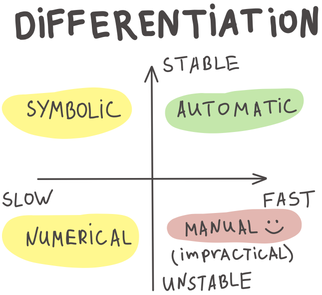


Рис. 32: Различные подходы для взятия производных

Чем автоматическое дифференцирование (AD) не является:

- AD не является методом конечных разностей
- AD не является символьным вычислением производных
- AD не является только правилом вычисления производной сложной функции

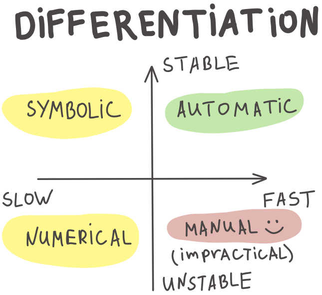


Рис. 32: Различные подходы для взятия производных

Чем автоматическое дифференцирование (AD) не является:

- AD не является методом конечных разностей
- AD не является символьным вычислением производных
- AD не является только правилом вычисления производной сложной функции
- AD (обратный режим) является времяэффективным и численно стабильным

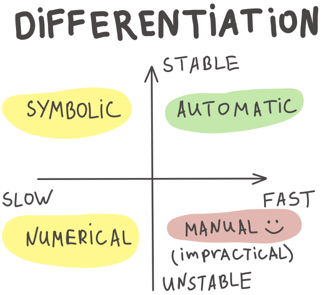


Рис. 32: Различные подходы для взятия производных

Чем автоматическое дифференцирование (AD) не является:

- AD не является методом конечных разностей
- AD не является символьным вычислением производных
- AD не является только правилом вычисления производной сложной функции
- AD (обратный режим) является времяэффективным и численно стабильным
- AD (обратный режим) не является эффективным по памяти (нужно хранить все промежуточные вычисления из прямого прохода)

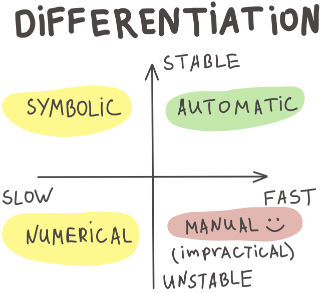


Рис. 32: Различные подходы для взятия производных

Дополнительные материалы

- Рекомендую прочитать официальную книгу по Jax Autodiff. Open In Colab ♣

Дополнительные материалы

- Рекомендую прочитать официальную книгу по Jax Autodiff. Open In Colab ♣
- Распространение градиента через линейные наименьшие квадраты [семинар]

Дополнительные материалы

- Рекомендую прочитать официальную книгу по Jax Autodiff. Open In Colab ♣
- Распространение градиента через линейные наименьшие квадраты [семинар]
- Распространение градиента через SVD [семинар]

Дополнительные материалы

- Рекомендую прочитать официальную книгу по Jax Autodiff. Open In Colab ♣
- Распространение градиента через линейные наименьшие квадраты [семинар]
- Распространение градиента через SVD [семинар]
- Контрольные точки активаций [семинар]