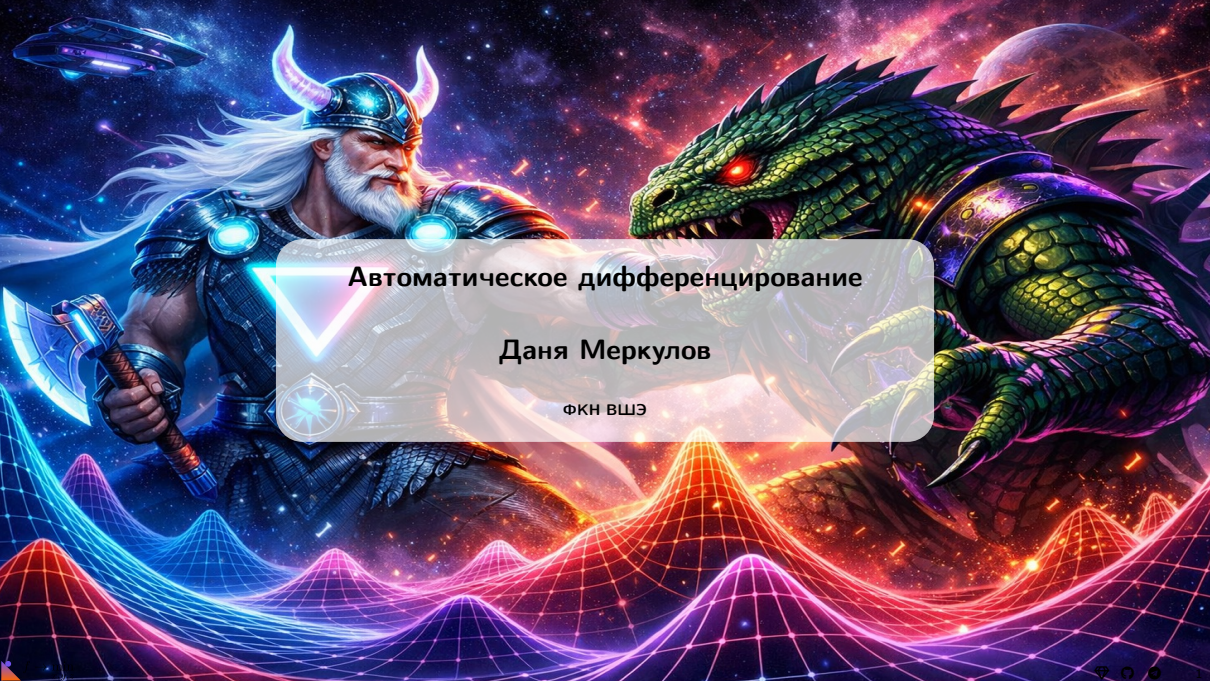


Автоматическое дифференцирование

Даня Меркулов

ФКН ВШЭ

Автоматическое дифференцирование

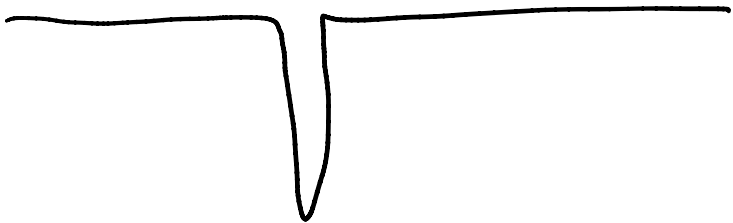
Figure 1: Когда вы поняли идею

Figure 2: Это не autograd

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).

Если при решении задачи вы используете только $L(w)$, но НЕ $\nabla_w L(w)$, $\nabla^2_w L(w)$, то ЭТО МЕДЛЕННО

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 $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_d}\right)^T$.

Задача

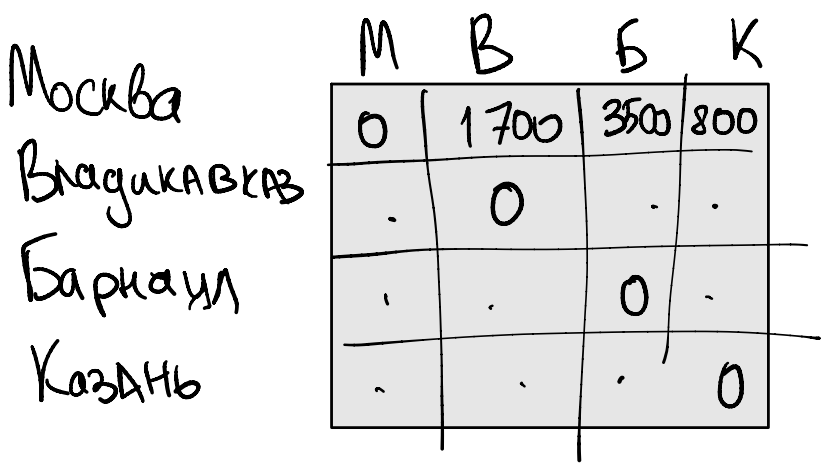
Предположим, что мы хотим решить следующую задачу:

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

- 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 $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_d}\right)^T$.
- Обычно методы первого порядка работают лучше в больших задачах, в то время как методы второго порядка требуют слишком много памяти.

Пример: многомерное шкалирование

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: многомерное шкалирование

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 $W_i \in \mathbb{R}^d,\ i = 1, \ldots, N$.

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

предсказанные расстояния

истинные расстояния

Пример: многомерное шкалирование

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 $W_i \in \mathbb{R}^d,\ i = 1, \dots, N$.

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

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 шкалирование

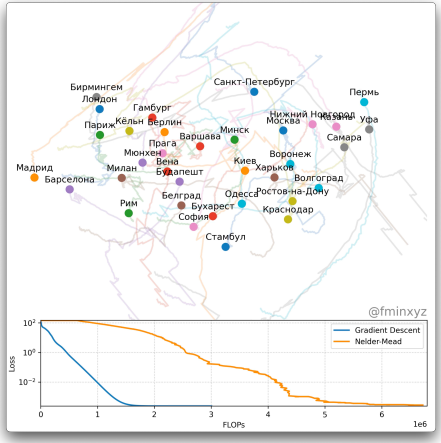


Figure 3: Ссылка на анимацию

Градиентный спуск без градиента: 2-точечный метод аппроксимации

Предположим, что мы хотим решить следующую задачу:

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

Градиентный спуск без градиента: 2-точечный метод аппроксимации

Предположим, что мы хотим решить следующую задачу:

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

с помощью алгоритма градиентного спуска (GD):

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

Градиентный спуск без градиента: 2-точечный метод аппроксимации

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?

Градиентный спуск без градиента: 2-точечный метод аппроксимации

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

Градиентный спуск без градиента: 2-точечный метод аппроксимации

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 - случайный вектор из сферически симметричного распределения.

вектор

вектор

скаляр

[1] рекомендуется хорошая презентация о безградиентных методах

Градиентный спуск без градиента: 2-точечный метод аппроксимации

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 - случайный вектор из сферически симметричного распределения.

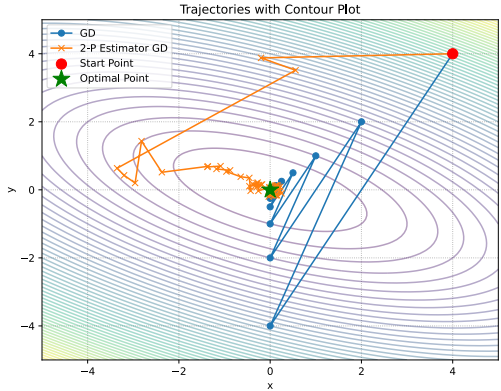


Figure 4: “Иллюстрация двухточечной оценки градиентного спуска”

[1] рекомендуется хорошая презентация о безградиентных методах

Градиентный спуск без градиента: многоточечный метод аппроксимации

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

Градиентный спуск без градиента: многоточечный метод аппроксимации

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

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 ♣

2·d вычислений знач.

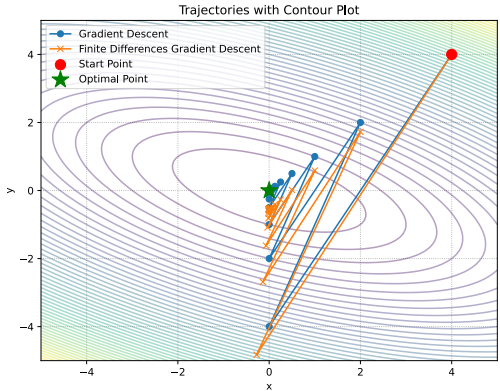


Figure 5: "Иллюстрация работы метода оценки градиента с помощью метода конечных разностей"

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:

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

i 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:

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

i 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:

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

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.

Теорема

Существует алгоритм для **точного** 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

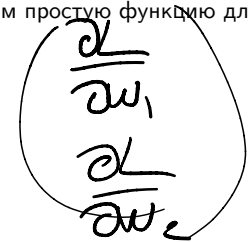
Чтобы глубже понять идею автоматического дифференцирования, рассмотрим простую функцию для вычисления производных:

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

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:

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

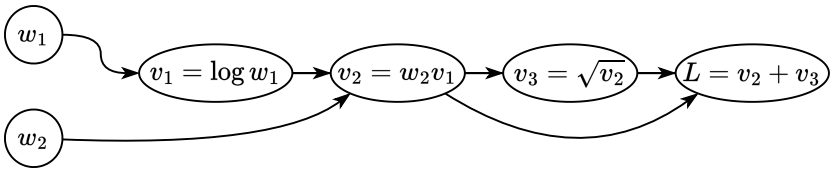


Figure 6: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

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:

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

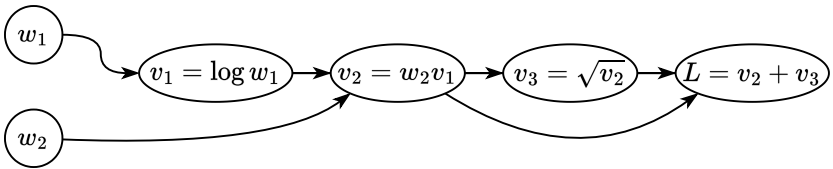


Figure 6: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

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

Прямой режим автоматического дифференцирования

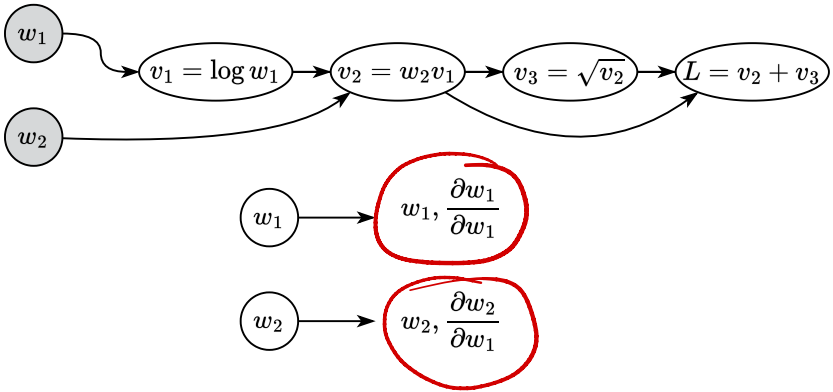


Figure 7: Иллюстрация прямого режима автоматического дифференцирования

Функция

$w_1 = w_1, w_2 = w_2$

Прямой режим автоматического дифференцирования

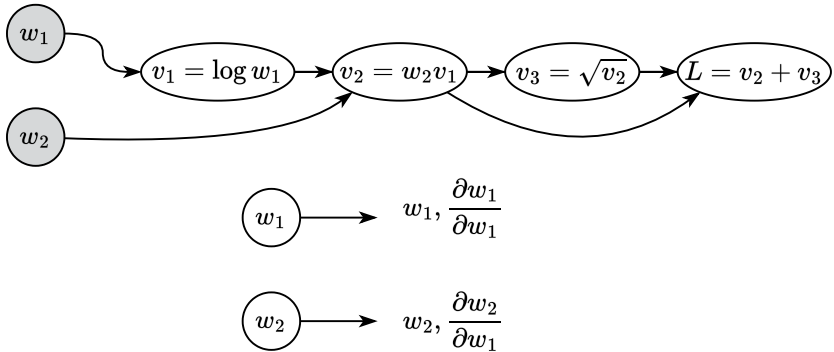


Figure 7: Иллюстрация прямого режима автоматического дифференцирования

Функция

$$w_1 = w_1, w_2 = w_2$$

Производная

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

Прямой режим автоматического дифференцирования

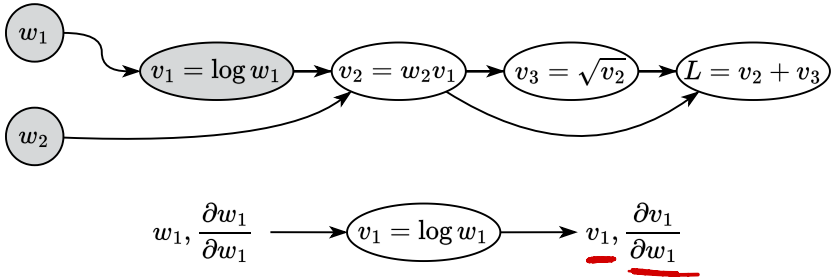


Figure 8: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

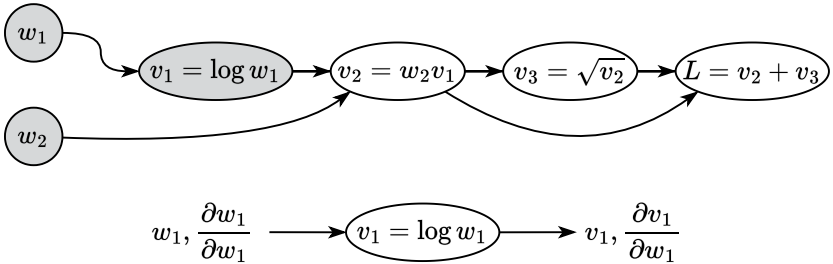


Figure 8: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_1 = \log w_1$

Прямой режим автоматического дифференцирования

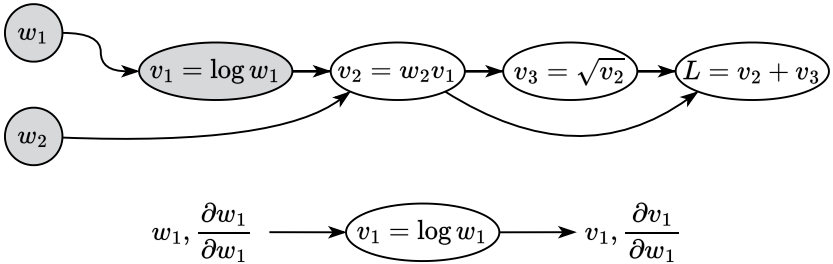


Figure 8: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_1 = \log w_1$

Производная

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

отрезок структурой графа

пришло с пред. шага

Прямой режим автоматического дифференцирования

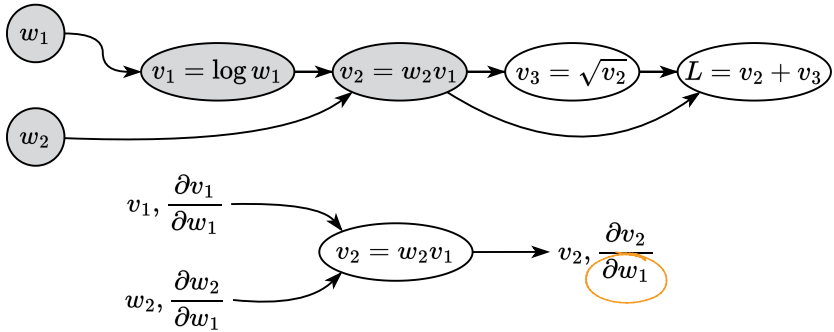


Figure 9: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

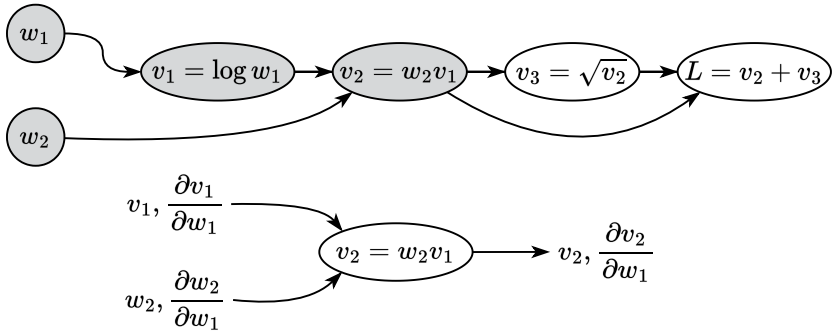


Figure 9: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_2 = w_2 v_1$

Прямой режим автоматического дифференцирования

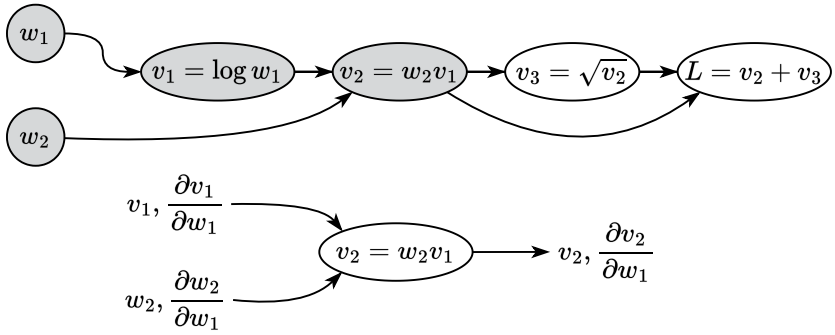


Figure 9: Иллюстрация прямого режима автоматического дифференцирования

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

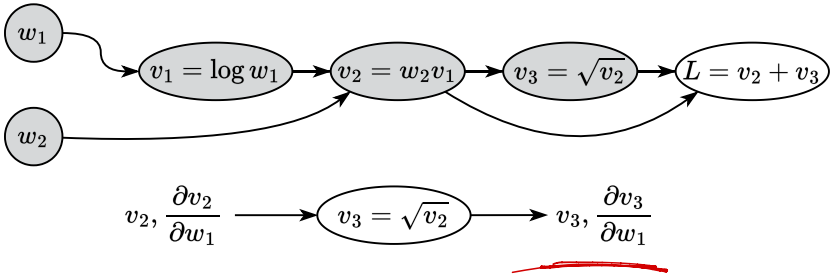


Figure 10: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

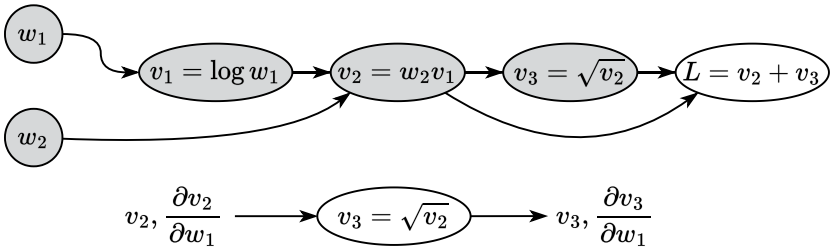


Figure 10: Иллюстрация прямого режима автоматического дифференцирования

Функция

$v_3 = \sqrt{v_2}$

Прямой режим автоматического дифференцирования

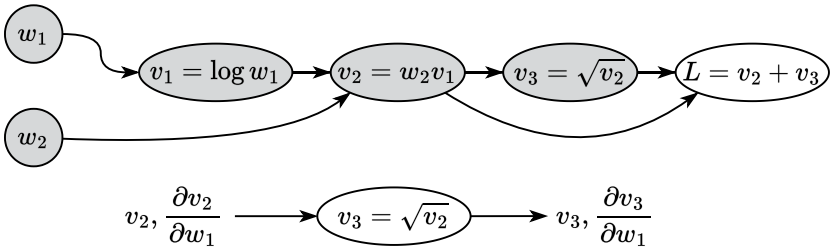


Figure 10: Иллюстрация прямого режима автоматического дифференцирования

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

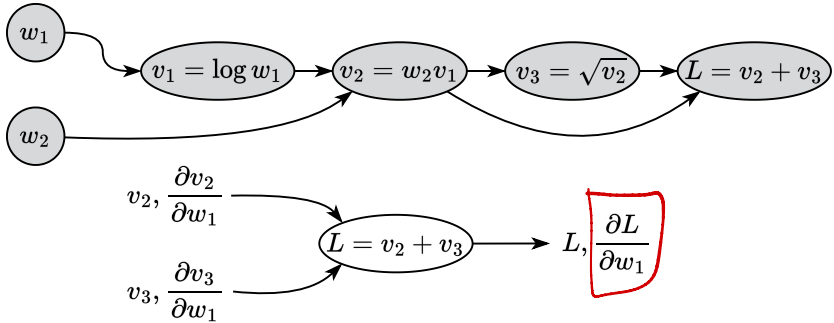


Figure 11: Иллюстрация прямого режима автоматического дифференцирования

Прямой режим автоматического дифференцирования

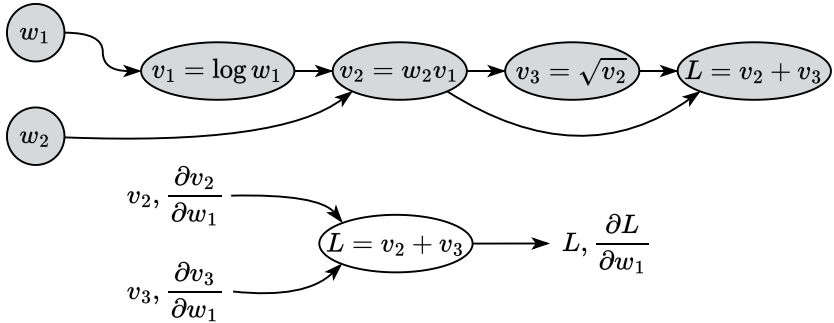


Figure 11: Иллюстрация прямого режима автоматического дифференцирования

Функция

$L = v_2 + v_3$

Прямой режим автоматического дифференцирования

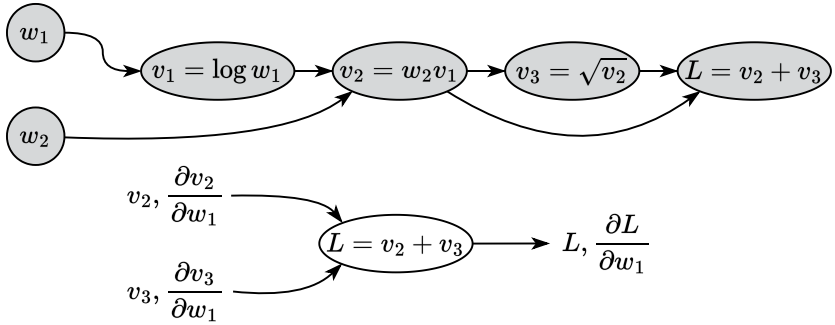


Figure 11: Иллюстрация прямого режима автоматического дифференцирования

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 $\dfrac{\partial L}{\partial w_2}$

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

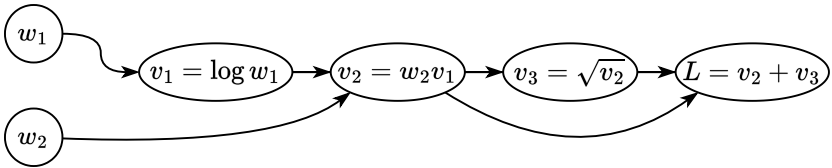


Figure 12: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Пример прямого режима автоматического дифференцирования

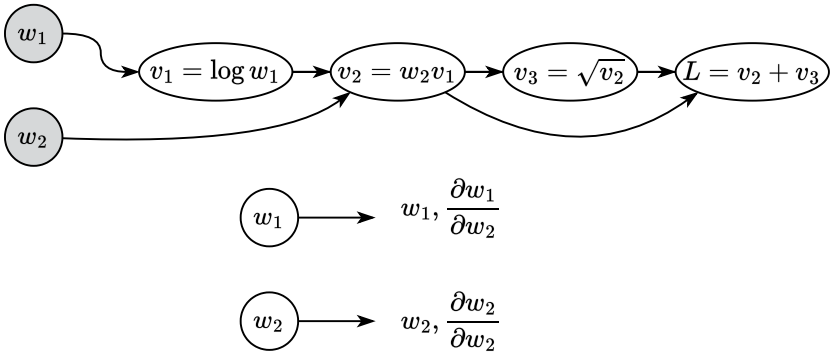


Figure 13: Иллюстрация прямого режима автоматического дифференцирования

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

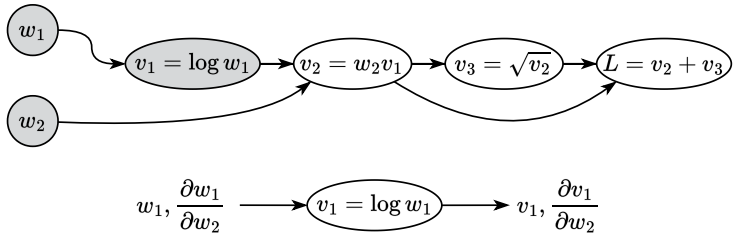


Figure 14: Иллюстрация прямого режима автоматического дифференцирования

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

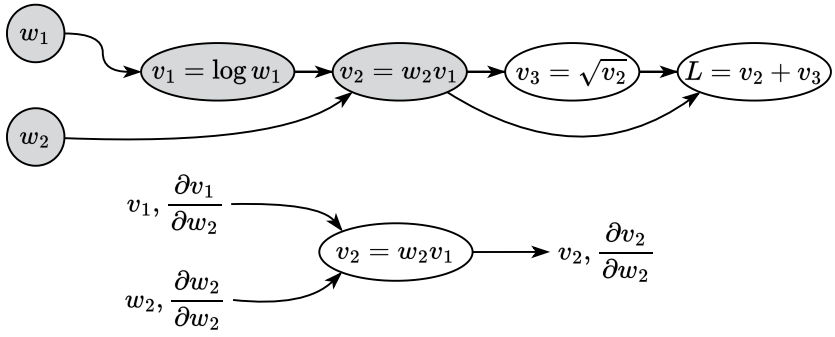


Figure 15: Иллюстрация прямого режима автоматического дифференцирования

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

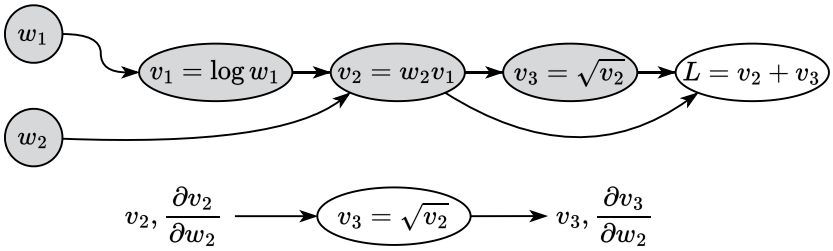


Figure 16: Иллюстрация прямого режима автоматического дифференцирования

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

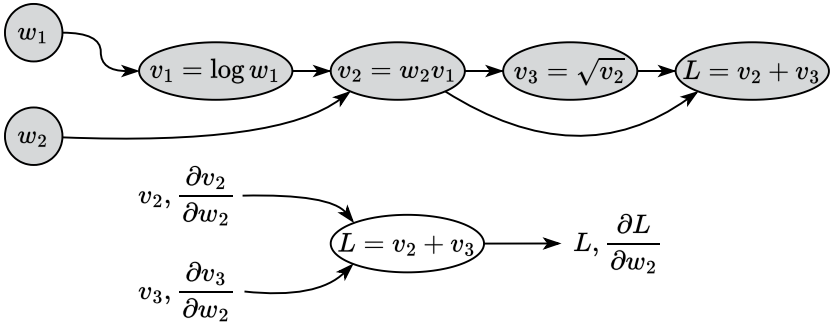


Figure 17: Иллюстрация прямого режима автоматического дифференцирования

Функция

$L = v_2 + v_3$

Производная

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

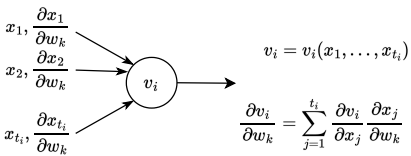
Алгоритм прямого режима автоматического дифференцирования

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. $\frac{\partial v_N}{\partial w_k}$. 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:

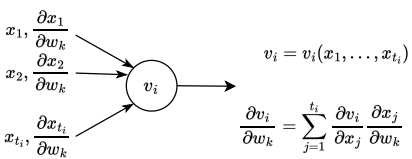
$$\overline{v_i} = \frac{\partial v_i}{\partial w_k}$$



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:

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

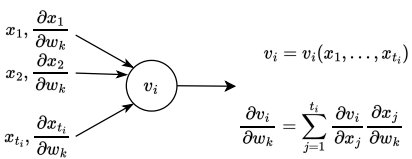


- Для $i = 1, \ldots, N$:

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:

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



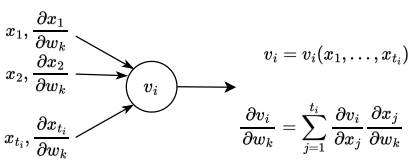
- Для $i = 1, \ldots, N$:
  - Вычислить v_i как функцию его предков $x_1, \ldots, x_{t_i}$:

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

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 $x_1, \ldots, x_{t_i}$:

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

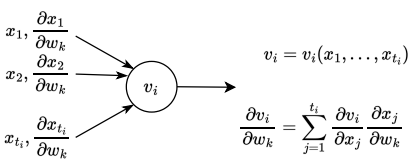
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции:

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

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 $x_1, \ldots, x_{t_i}$:

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

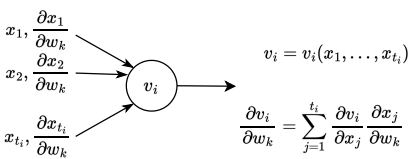
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции:

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

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:

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

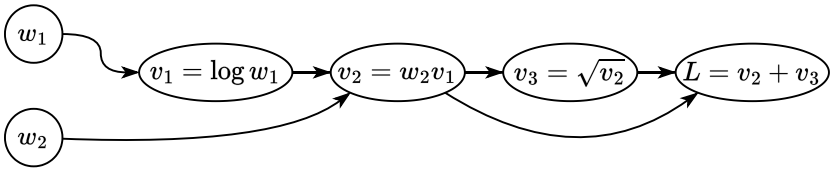


Figure 18: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

Обратный режим автоматического дифференцирования

Мы рассмотрим ту же функцию с вычислительным графом:

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

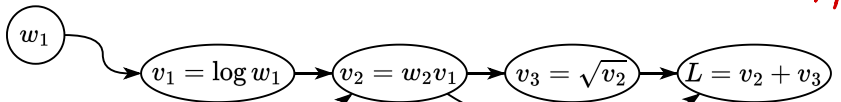


Figure 18: Иллюстрация вычислительного графа для функции $L(w_1, w_2)$

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

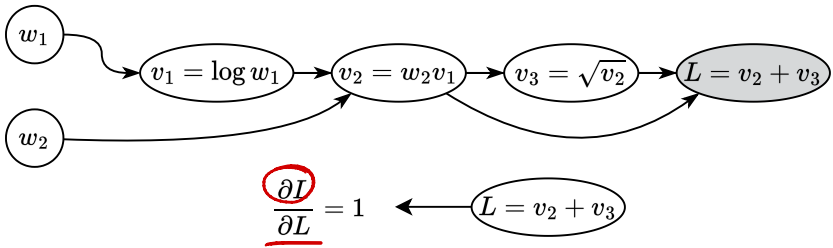


Figure 19: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

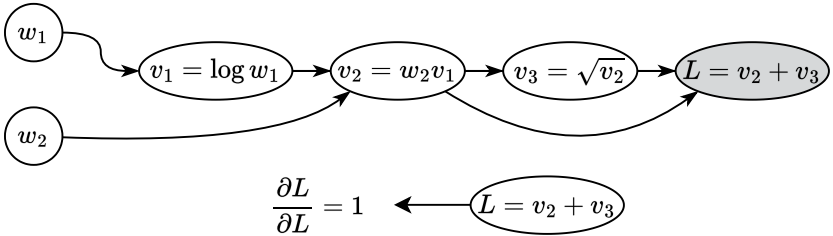


Figure 19: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

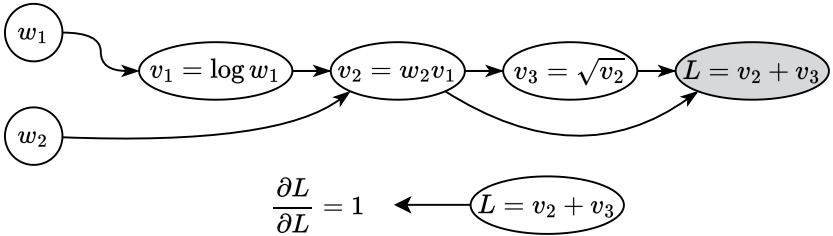


Figure 19: Иллюстрация обратного режима автоматического дифференцирования

Производные

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

Пример обратного режима автоматического дифференцирования

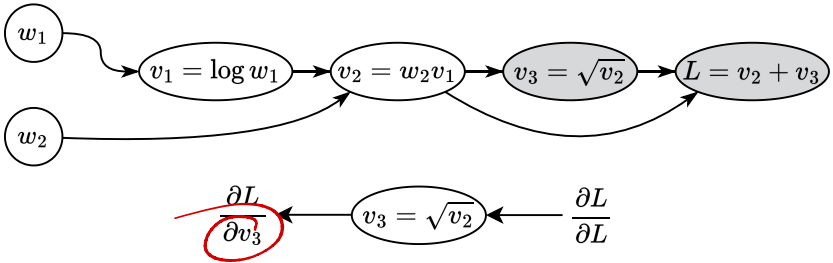


Figure 20: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

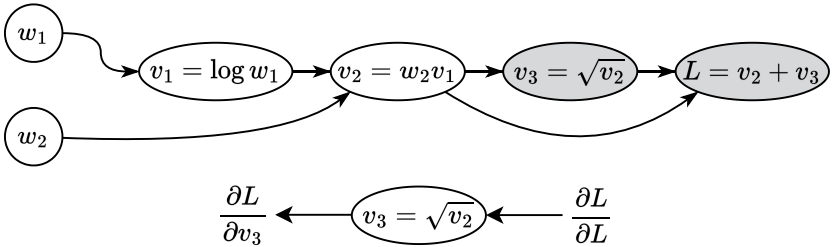


Figure 20: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

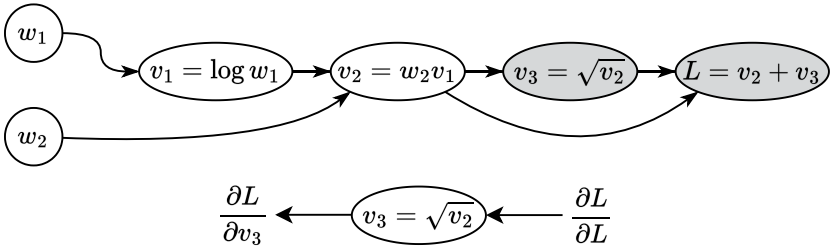


Figure 20: Иллюстрация обратного режима автоматического дифференцирования

Производные

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

Пример обратного режима автоматического дифференцирования

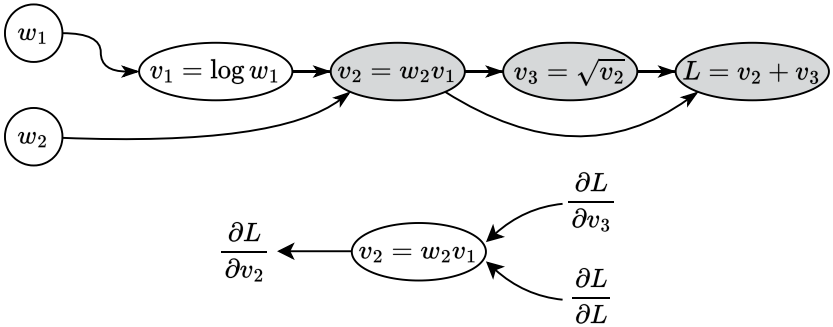


Figure 21: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

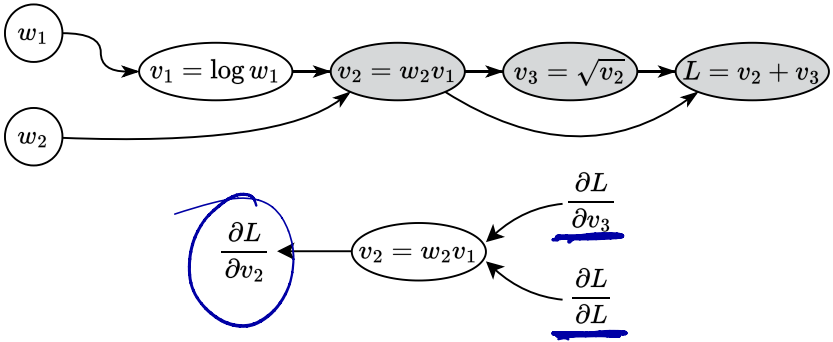


Figure 21: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

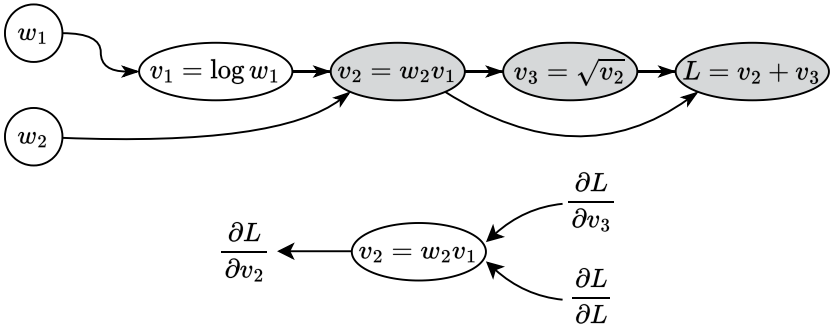


Figure 21: Иллюстрация обратного режима автоматического дифференцирования

Производные

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

Пример обратного режима автоматического дифференцирования

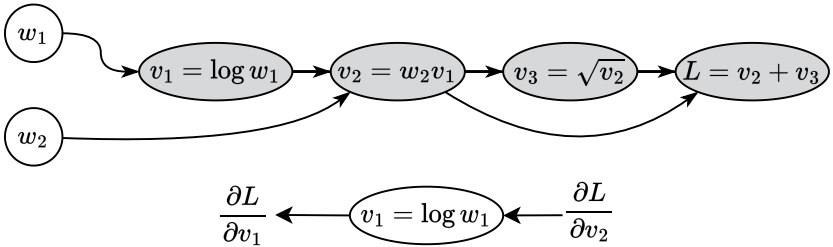


Figure 22: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

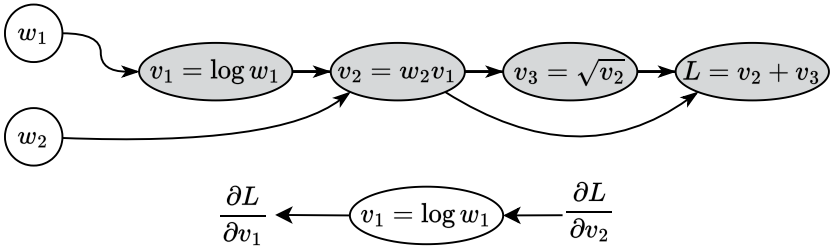


Figure 22: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

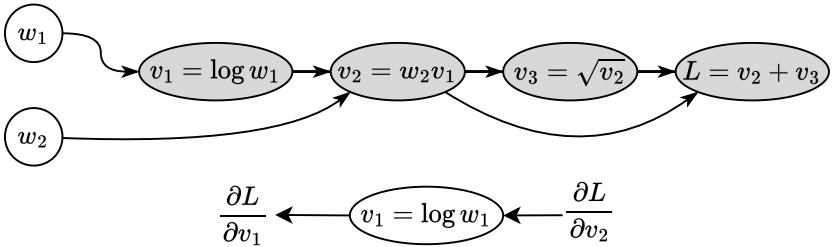


Figure 22: Иллюстрация обратного режима автоматического дифференцирования

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

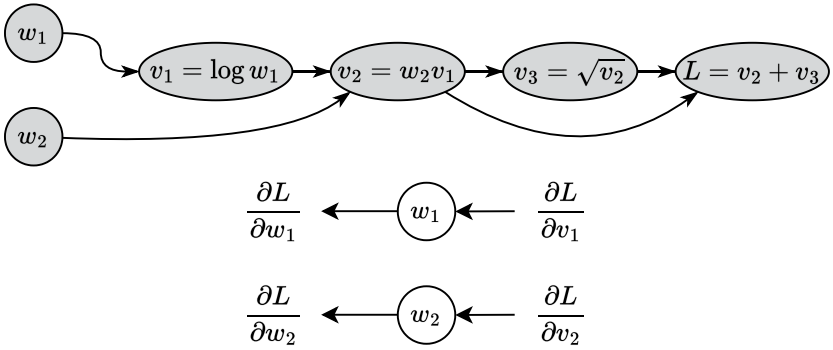


Figure 23: Иллюстрация обратного режима автоматического дифференцирования

Пример обратного режима автоматического дифференцирования

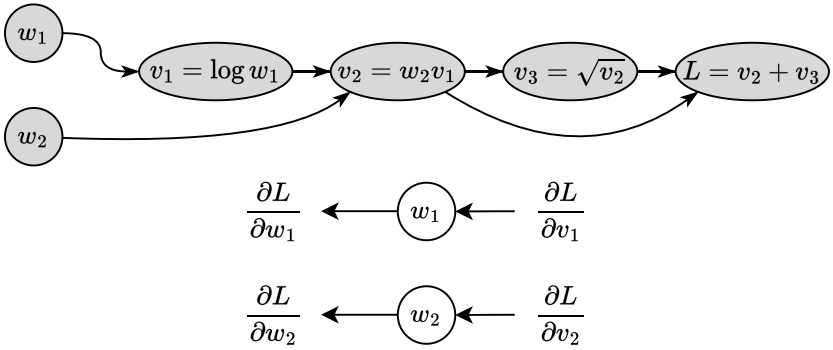


Figure 23: Иллюстрация обратного режима автоматического дифференцирования

Производные

Пример обратного режима автоматического дифференцирования

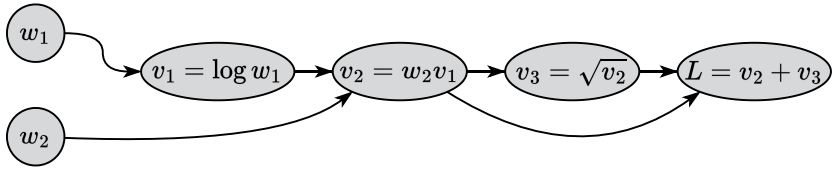


Figure 23: Иллюстрация обратного режима автоматического дифференцирования

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

1.5 B fp32 → 6ГБ

вам не хватит

5090 32 Гига

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?

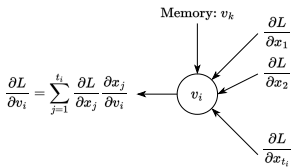
Ответ Обратите внимание, что для использования обратного режима AD вам нужно хранить все промежуточные вычисления из прямого прохода. Эта проблема может быть частично решена с помощью чекпоинтинга, при котором мы сохраняем только часть промежуточных значений, а остальные пересчитываем заново по мере необходимости. Это позволяет значительно уменьшить объём требуемой памяти при обучении больших моделей машинного обучения.

Но если вы карте, то хватит

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. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \dots, \frac{\partial v_N}{\partial w_d}\right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

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

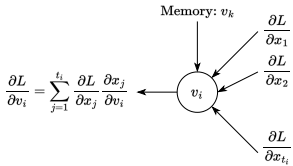


- **ПРЯМОЙ ПРОХОД**
  Для $i = 1, \dots, N$:

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. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \dots, \frac{\partial v_N}{\partial w_d}\right)^T$. Эта идея предполагает распространение градиента функции по промежуточным переменным от конца к началу, поэтому мы можем ввести обозначение:

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



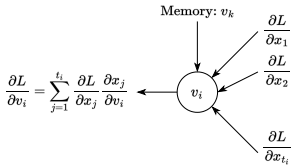
- **ПРЯМОЙ ПРОХОД**

  Для $i = 1, \dots, N$:
  - 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. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \dots, \frac{\partial v_N}{\partial w_d}\right)^T$. 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**

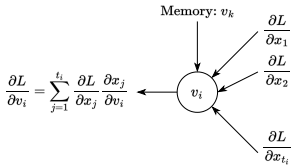
  Для $i = 1, \dots, N$:
  - Вычислить и сохранить значения v_i как функцию его предков
- **ОБРАТНЫЙ ПРОХОД**

  Для $i = N, \dots, 1$:

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. $\nabla_w v_N = \left(\frac{\partial v_N}{\partial w_1}, \dots, \frac{\partial v_N}{\partial w_d}\right)^T$. 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 $i = 1, \dots, N$:
  - Вычислить и сохранить значения v_i как функцию его предков

- **ОБРАТНЫЙ ПРОХОД**

  Для $i = N, \dots, 1$:
  - Вычислить производную $\overline{v_i}$ используя формулу производной сложной функции и информацию от всех потомков (выходов):

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

FORWARD

BACKWARD

LOSS = MSE.

LOSS.BACKWARD()

W.grad

Выбор режима AD

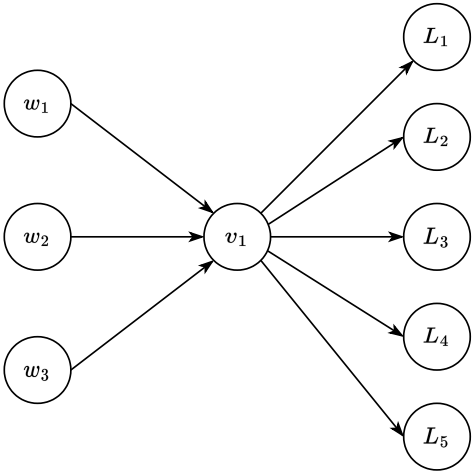


Figure 24: Какой режим вы бы выбрали для вычисления градиентов?

Question

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

Вход $x \in \mathbb{R}^3$

Выход $y \in \mathbb{R}^5$

$y = Ax$

Выбор режима AD

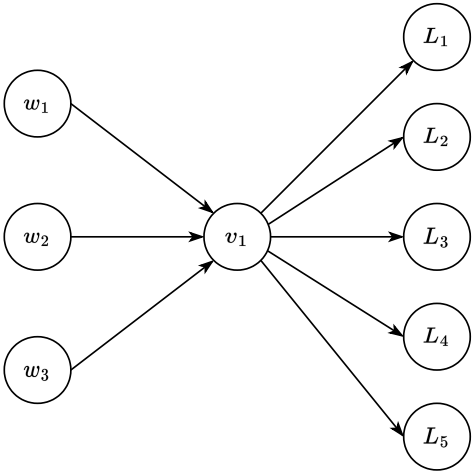


Figure 24: Какой режим вы бы выбрали для вычисления градиентов?

Question

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

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.

Выбор режима AD

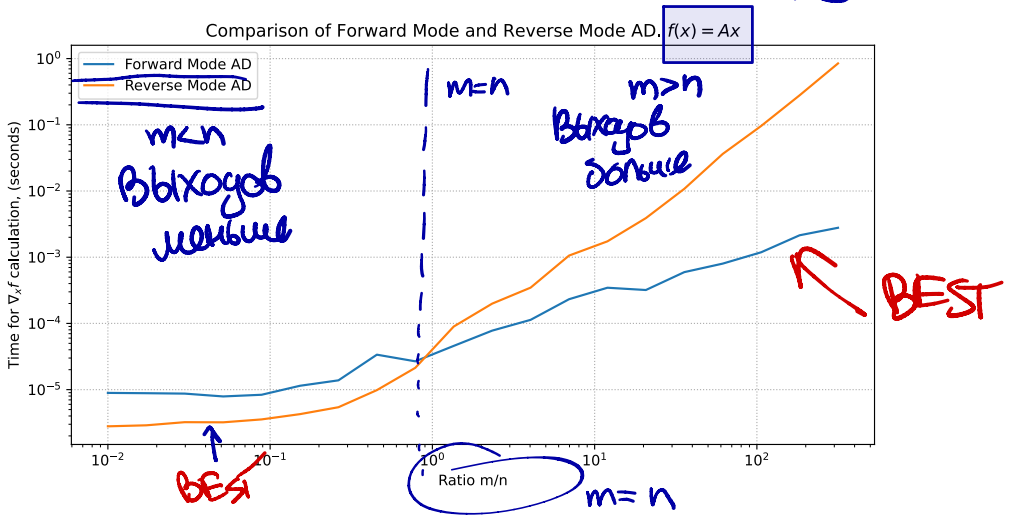


Figure 25: ♣ График иллюстрирует идею выбора между режимами автоматического дифференцирования. Размерность входа $n = 100$ фиксирована, измерено время вычисления якобиана в зависимости от соотношения размерностей выхода и входа для разных размерностей выхода m.

Выбор режима AD

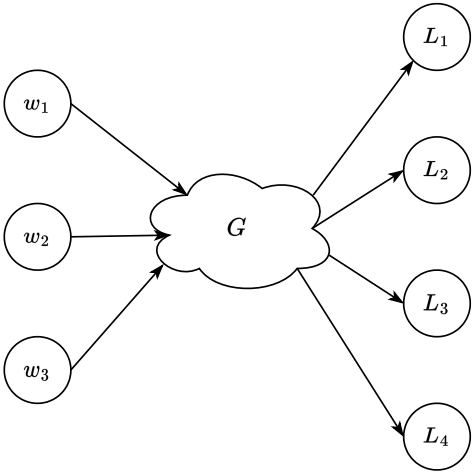


Figure 26: Какой режим вы бы выбрали для вычисления градиентов?

Question

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

Выбор режима AD

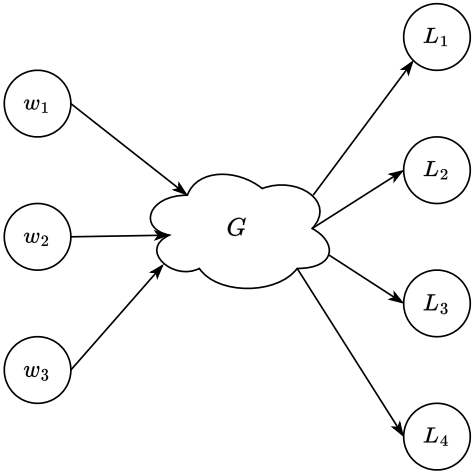


Figure 26: Какой режим вы бы выбрали для вычисления градиентов?

Question

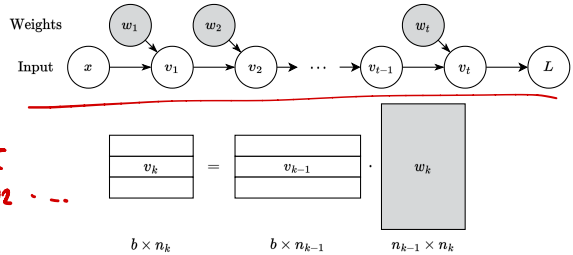
Какой из режимов AD вы бы выбрали (прямой/обратный) для следующего вычислительного графа арифметических операций? Предположим, что вам нужно вычислить якобиан $J = \left\{ \frac{\partial L_i}{\partial w_j} \right\}_{i,j}$. Обратите внимание, что G - это произвольный вычислительный граф

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.

Пример: алгоритм обратного AD для архитектуры прямого распространения

ПРЯМОЙ ПРОХОД

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



ОБРАТНЫЙ ПРОХОД

Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры прямого распространения

ПРЯМОЙ ПРОХОД

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

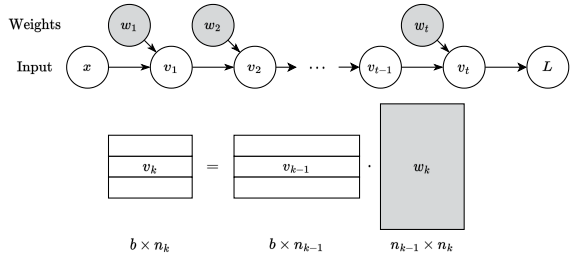


Figure 27: Архитектура прямого распространения нейронной сети

ОБРАТНЫЙ ПРОХОД

Пример: алгоритм обратного AD для архитектуры 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

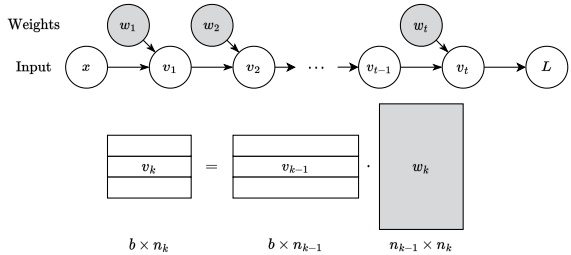


Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры 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

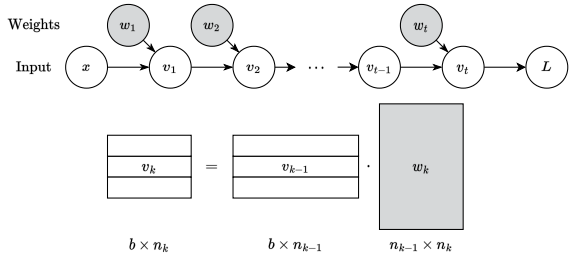


Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры 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

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

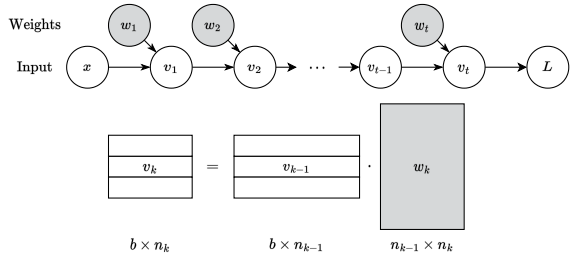


Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры 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 $k = t, t-1, \dots, 1$:

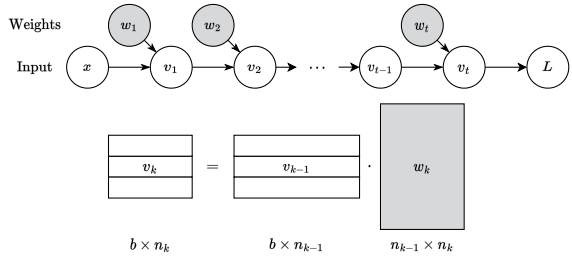


Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры 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 $k = t, t-1, \dots, 1$:
  - $\underset{b \times n_k}{\dfrac{\partial L}{\partial v_k}} = \underset{b \times n_{k+1}}{\dfrac{\partial L}{\partial v_{k+1}}} \underset{n_{k+1} \times n_k}{\dfrac{\partial v_{k+1}}{\partial v_k}}$

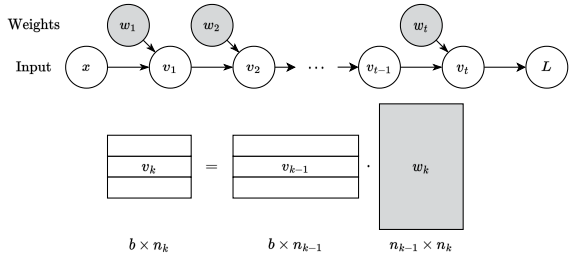


Figure 27: Архитектура прямого распространения нейронной сети

Пример: алгоритм обратного AD для архитектуры 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 $k = t, t-1, \dots, 1$:
  - $\underset{b \times n_k}{\dfrac{\partial L}{\partial v_k}} = \underset{b \times n_{k+1}}{\dfrac{\partial L}{\partial v_{k+1}}} \underset{n_{k+1} \times n_k}{\dfrac{\partial v_{k+1}}{\partial v_k}}$
  - $\underset{b \times n_{k-1} \cdot n_k}{\dfrac{\partial L}{\partial w_k}} = \underset{b \times n_{k+1}}{\dfrac{\partial L}{\partial v_{k+1}}} \cdot \underset{n_{k+1} \times n_{k-1} \cdot n_k}{\dfrac{\partial v_{k+1}}{\partial w_k}}$

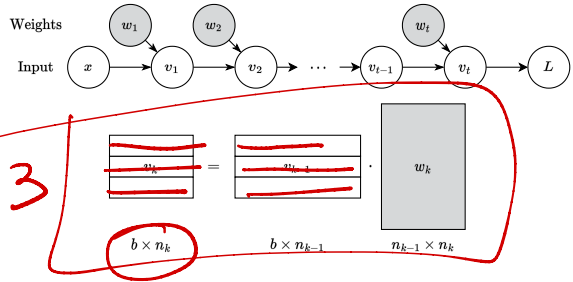
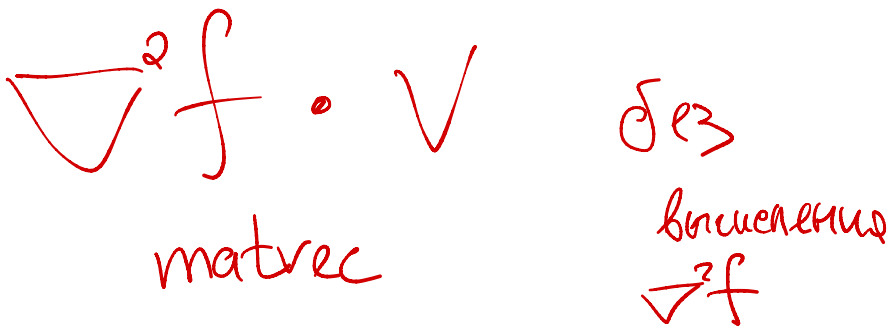


Figure 27: Архитектура прямого распространения нейронной сети

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

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

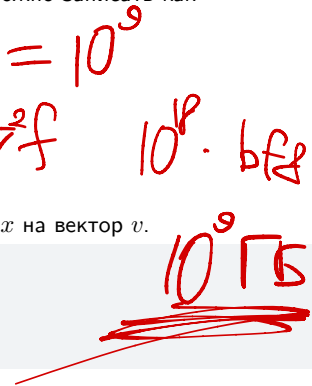
для любого вектора $v \in \mathbb{R}^n$. Мы можем использовать тождество

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

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

```
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]

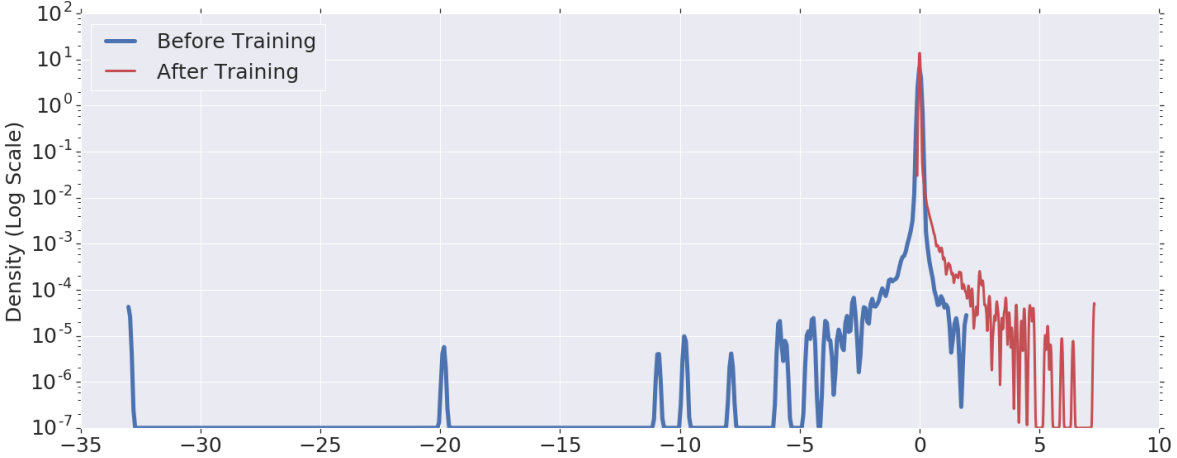


Figure 28: Большие по модулю отрицательные собственные значения гессиана исчезли после обучения ResNet-32

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

Идея Хатчинсона для оценки следа матрицы [5]

Этот пример иллюстрирует оценку следа гессиана нейронной сети с использованием метода Хатчинсона - алгоритма, который позволяет получить такую оценку из произведений матрицы на 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,

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

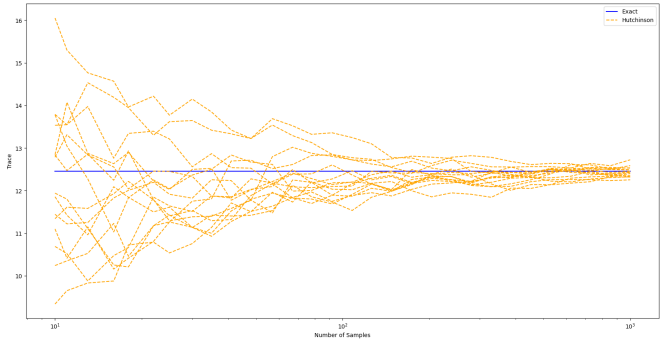


Figure 29: Источник

[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 без оптимизаций 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 без оптимизаций 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

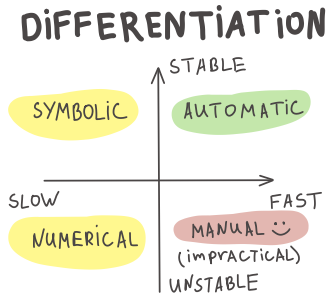


Figure 30: Различные подходы для взятия производных

Чем автоматическое дифференцирование (AD) не является:

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

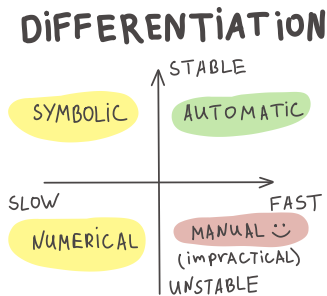


Figure 30: Различные подходы для взятия производных

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

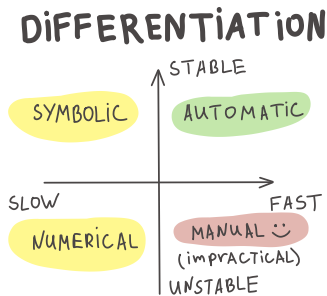


Figure 30: Различные подходы для взятия производных

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 - это не только backpropagation

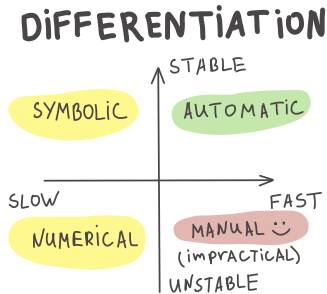


Figure 30: Различные подходы для взятия производных

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 - это не только backpropagation
- AD (обратный режим) является эффективным по времени и численно стабильным

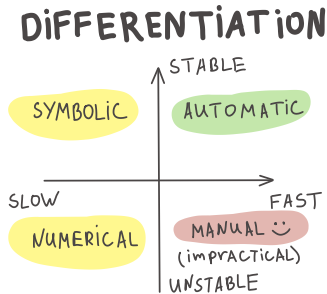


Figure 30: Различные подходы для взятия производных

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 - это не только backpropagation
- AD (обратный режим) является эффективным по времени и численно стабильным
- AD (обратный режим) не является эффективным по памяти (нужно хранить все промежуточные вычисления из прямого прохода)

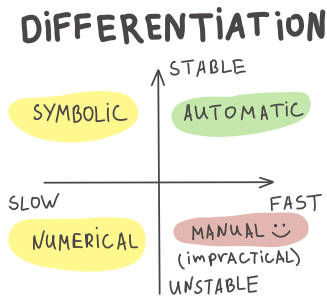


Figure 30: Различные подходы для взятия производных

Дополнительные материалы

- Рекомендую прочитать официальное руководство по Jax Autodiff. Open In Colab ♣

Дополнительные материалы

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

Дополнительные материалы

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

Дополнительные материалы

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

Итоги

Итоги

Определения

1. Формула для приближенного вычисления производной функции $f(x) : \mathbb{R}^n \to \mathbb{R}$ по k-ой координате с помощью метода конечных разностей.
2. Пусть $f = f(x_1(t), \dots, x_n(t))$. Формула для вычисления $\frac{\partial f}{\partial t}$ через $\frac{\partial x_i}{\partial t}$ (Forward chain rule).
3. Пусть L - функция, возвращающая скаляр, а v_k - функция, возвращающая вектор $x \in \mathbb{R}^t$. Формула для вычисления $\frac{\partial L}{\partial v_k}$ через $\frac{\partial L}{\partial x_i}$ (Backward chain rule).
4. Идея Хатчинсона для оценки следа матрицы с помощью matvec операций.

Теоремы

1. Автоматическое дифференцирование. Вычислительный граф. Forward/ Backward mode (в этом вопросе нет доказательств, но необходимо подробно описать алгоритмы).