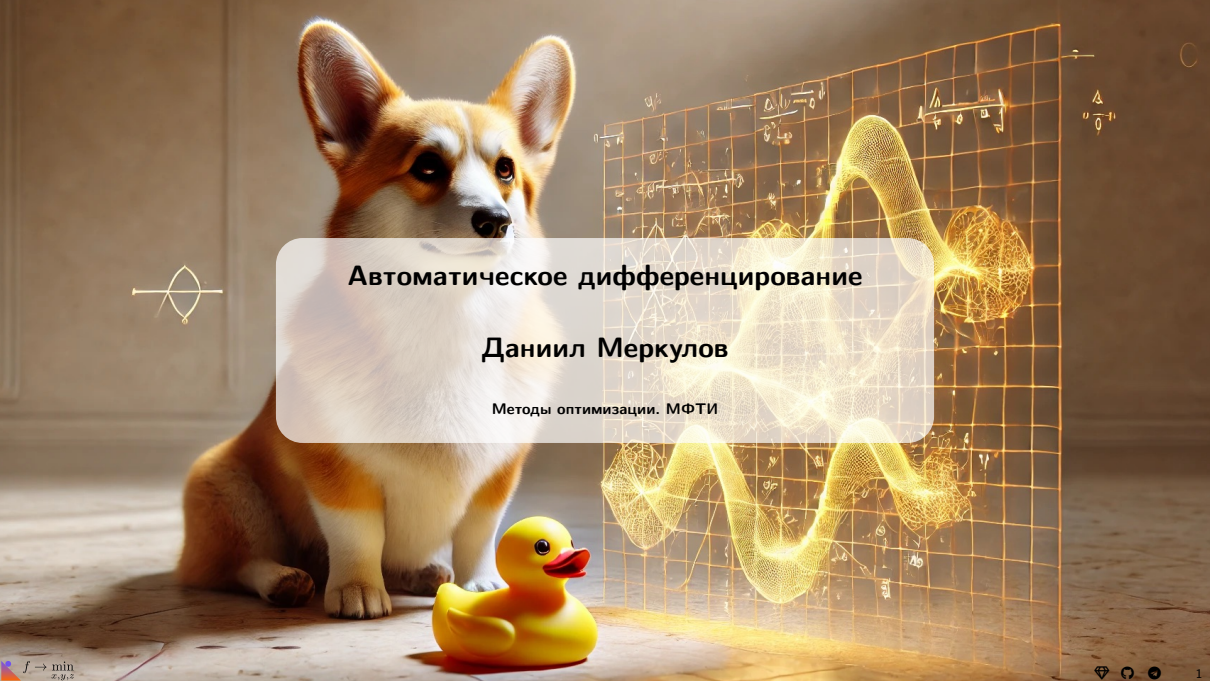

# Автоматическое дифференцирование

Даниил Меркулов

Методы оптимизации. МФТИ

# Повторим матричное дифференцирование

# Пример 1

**Example**

Найти гессиан $\nabla^2 f(x)$, если $f(x) = \langle x, Ax \rangle - b^T x + c$.

# Пример 1

**Example**

Найти гессиан $\nabla^2 f(x)$, если $f(x) = \langle x, Ax \rangle - b^T x + c$.

1. Распишем дифференциал $df$

$$\begin{aligned} df &= d\left(\langle Ax, x \rangle - \langle b, x \rangle + c\right) \\ &= \langle Ax, dx \rangle + \langle x, Adx \rangle - \langle b, dx \rangle \\ &= \langle Ax, dx \rangle + \langle A^T x, dx \rangle - \langle b, dx \rangle \\ &= \langle (A + A^T)x - b, dx \rangle \end{aligned}$$

Что означает, что градиент $\nabla f = (A + A^T)x - b$.

# Пример 1

**Example**

Найти гессиан $\nabla^2 f(x)$, если $f(x) = \langle x, Ax \rangle - b^T x + c$.

1. Распишем дифференциал $df$

$$\begin{aligned} df &= d\left(\langle Ax, x \rangle - \langle b, x \rangle + c\right) \\ &= \langle Ax, dx \rangle + \langle x, Adx \rangle - \langle b, dx \rangle \\ &= \langle Ax, dx \rangle + \langle A^T x, dx \rangle - \langle b, dx \rangle \\ &= \langle (A + A^T)x - b, dx \rangle \end{aligned}$$

Что означает, что градиент $\nabla f = (A + A^T)x - b$.

# Пример 1

**Example**

Найти гессиан $\nabla^2 f(x)$, если $f(x) = \langle x, Ax \rangle - b^T x + c$.

1. Распишем дифференциал $df$

$$\begin{aligned} df &= d\left(\langle Ax, x\rangle - \langle b, x\rangle + c\right) \\ &= \langle Ax, dx\rangle + \langle x, Adx\rangle - \langle b, dx\rangle \\ &= \langle Ax, dx\rangle + \langle A^T x, dx\rangle - \langle b, dx\rangle \\ &= \langle (A + A^T)x - b, dx\rangle \end{aligned}$$

Что означает, что градиент $\nabla f = (A + A^T)x - b$.

2. Найдем второй дифференциал $d^2 f = d(df)$, полагая, что $dx = dx_1 = \text{const}$:

$$\begin{aligned} d^2 f &= d\left(\langle (A + A^T)x - b, dx_1\rangle\right) \\ &= \langle (A + A^T)dx, dx_1\rangle \\ &= \langle dx, (A + A^T)^T dx_1\rangle \\ &= \langle (A + A^T)dx_1, dx\rangle \end{aligned}$$

Таким образом, гессиан: $\nabla^2 f = (A + A^T)$.

# Пример 2

**Example**

Найти гессиан $\nabla^2 f(x)$, если $f(x) = \ln\langle x, Ax\rangle$.

# Пример 3

**Example**

Найти градиент $\nabla f(x)$ и гессиан $\nabla^2 f(x)$, если $f(x) = \ln\left(1 + \exp\langle a, x\rangle\right)$

# Пример 3

**Example**

Найти градиент $\nabla f(x)$ и гессиан $\nabla^2 f(x)$, если $f(x) = \ln(1 + \exp\langle a, x\rangle)$

1. Начнем с записи дифференциала $df$. Имеем:

$$f(x) = \ln(1 + \exp\langle a, x\rangle)$$

Используя правило дифференцирования сложной функции:

$$df = d(\ln(1 + \exp\langle a, x\rangle)) = \frac{d(1 + \exp\langle a, x\rangle)}{1 + \exp\langle a, x\rangle}$$

теперь посчитаем дифференциал экспоненты:

$$d(\exp\langle a, x\rangle) = \exp\langle a, x\rangle\langle a, dx\rangle$$

Подставляя в выражение выше, имеем:

$$df = \frac{\exp\langle a, x\rangle\langle a, dx\rangle}{1 + \exp\langle a, x\rangle}$$

# Пример 3

**Example**

Найти градиент $\nabla f(x)$ и гессиан $\nabla^2 f(x)$, если $f(x) = \ln(1 + \exp\langle a, x\rangle)$

1. Начнем с записи дифференциала $df$. Имеем:

$$f(x) = \ln(1 + \exp\langle a, x\rangle)$$

Используя правило дифференцирования сложной функции:

$$df = d(\ln(1 + \exp\langle a, x\rangle)) = \frac{d(1 + \exp\langle a, x\rangle)}{1 + \exp\langle a, x\rangle}$$

теперь посчитаем дифференциал экспоненты:

$$d(\exp\langle a, x\rangle) = \exp\langle a, x\rangle\langle a, dx\rangle$$

Подставляя в выражение выше, имеем:

$$df = \frac{\exp\langle a, x\rangle\langle a, dx\rangle}{1 + \exp\langle a, x\rangle}$$

# Пример 3

**Example**

Найти градиент $\nabla f(x)$ и гессиан $\nabla^2 f(x)$, если $f(x) = \ln\left(1 + \exp\langle a, x\rangle\right)$

1. Начнем с записи дифференциала $df$. Имеем:

$$f(x) = \ln\left(1 + \exp\langle a, x\rangle\right)$$

Используя правило дифференцирования сложной функции:

$$df = d\left(\ln\left(1 + \exp\langle a, x\rangle\right)\right) = \frac{d\left(1 + \exp\langle a, x\rangle\right)}{1 + \exp\langle a, x\rangle}$$

теперь посчитаем дифференциал экспоненты:

$$d\left(\exp\langle a, x\rangle\right) = \exp\langle a, x\rangle\langle a, dx\rangle$$

Подставляя в выражение выше, имеем:

$$df = \frac{\exp\langle a, x\rangle\langle a, dx\rangle}{1 + \exp\langle a, x\rangle}$$

2. Для выражения $df$ в нужной форме, запишем:

$$df = \left\langle \frac{\exp\langle a, x\rangle}{1 + \exp\langle a, x\rangle} a, dx \right\rangle$$

Напомним, что функция сигмоиды определяется как:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

Таким образом, мы можем переписать дифференциал:

$$df = \langle \sigma(\langle a, x\rangle) a, dx\rangle$$

Следовательно, градиент:

$$\nabla f(x) = \sigma(\langle a, x\rangle) a$$

# Пример 3

**Example**

Найти градиент $\nabla f(x)$ и гессиан $\nabla^2 f(x)$, если $f(x) = \ln(1 + \exp\langle a, x\rangle)$

3. Теперь найдем гессиан с помозью второго дифференциала:

$$d(\nabla f(x)) = d\left(\sigma(\langle a, x\rangle)a\right)$$

Так как вектор $a$ константа, нам необходимо продифференцировать лишь сигмоиду:

$$d\left(\sigma(\langle a, x\rangle)\right) = \sigma(\langle a, x\rangle)(1 - \sigma(\langle a, x\rangle))\langle a, dx\rangle$$

То есть:

$$d(\nabla f(x)) = \sigma(\langle a, x\rangle)(1 - \sigma(\langle a, x\rangle))\langle a, dx\rangle a$$

Запишем гессиан:

$$\nabla^2 f(x) = \sigma(\langle a, x\rangle)(1 - \sigma(\langle a, x\rangle))aa^T$$

# Автоматическое дифференцирование

**@dpiponi@mathstodon.xyz**
@sigfpe

I think the first 40 years or so of automatic differentiation was largely people not using it because they didn't believe such an algorithm could possibly exist.

11:36 PM · Sep 17, 2019

9 26 159 13

Рис. 1: Когда понял идею

Рис. 2: Это не автоград

# Задача

Пусть есть задача оптимизации:

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

# Задача

Пусть есть задача оптимизации:

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

- Such problems typically arise in machine learning, when you need to find optimal hyperparameters $w$ of an ML model (i.e. train a neural network).

# Задача

Пусть есть задача оптимизации:

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

- Such problems typically arise in machine learning, when you need to find optimal hyperparameters $w$ of an ML model (i.e. train a neural network).
- You may use a lot of algorithms to approach this problem, but given the modern size of the problem, where $d$ could be dozens of billions it is very challenging to solve this problem without information about the gradients using zero-order optimization algorithms.

# Задача

Пусть есть задача оптимизации:

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

- Such problems typically arise in machine learning, when you need to find optimal hyperparameters $w$ of an ML model (i.e. train a neural network).
- You may use a lot of algorithms to approach this problem, but given the modern size of the problem, where $d$ could be dozens of billions it is very challenging to solve this problem without information about the gradients using zero-order optimization algorithms.
- That is why it would be beneficial to be able to calculate the gradient vector $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_d}\right)^T$.

# Задача

Пусть есть задача оптимизации:

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

- Such problems typically arise in machine learning, when you need to find optimal hyperparameters $w$ of an ML model (i.e. train a neural network).
- You may use a lot of algorithms to approach this problem, but given the modern size of the problem, where $d$ could be dozens of billions it is very challenging to solve this problem without information about the gradients using zero-order optimization algorithms.
- That is why it would be beneficial to be able to calculate the gradient vector $\nabla_w L = \left(\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_d}\right)^T$.
- Typically, first-order methods perform much better in huge-scale optimization, while second-order methods require too much memory.

# Пример: задача многомерного шкалирования

Suppose, we have a pairwise distance matrix for $N$ $d$-dimensional objects $D \in \mathbb{R}^{N \times N}$. Given this matrix, our goal is to recover the initial coordinates $W_i \in \mathbb{R}^d,\ i = 1, \dots, N$.

# Пример: задача многомерного шкалирования

Suppose, we have a pairwise distance matrix for $N$ $d$-dimensional objects $D \in \mathbb{R}^{N \times N}$. Given this matrix, our goal is to recover the initial coordinates $W_i \in \mathbb{R}^d,\ i = 1, \dots, N$.

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

# Пример: задача многомерного шкалирования

Suppose, we have a pairwise distance matrix for $N$ $d$-dimensional objects $D \in \mathbb{R}^{N \times N}$. Given this matrix, our goal is to recover the initial coordinates $W_i \in \mathbb{R}^d,\ i = 1, \dots, N$.

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

Link to a nice visualization ♣, where one can see, that gradient-free methods handle this problem much slower, especially in higher dimensions.

> **Question**
>
> Is it somehow connected with PCA?

# Пример: задача многомерного шкалирования

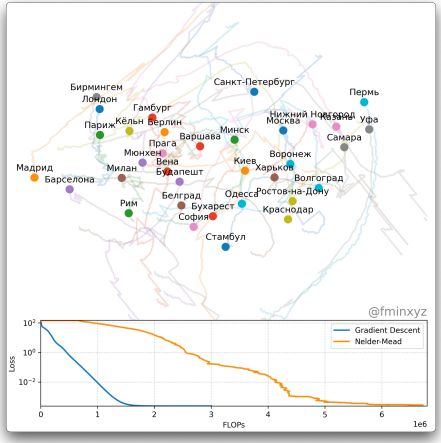


Рис. 3: Ссылка на анимацию

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

вместе с методом градиентного спуска (GD)

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

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

вместе с методом градиентного спуска (GD)

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

Можно ли заменить $\nabla_w L(w_k)$, используя, лишь информацию нулевого порядка о функции?

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

вместе с методом градиентного спуска (GD)

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

Можно ли заменить $\nabla_w L(w_k)$, используя, лишь информацию нулевого порядка о функции?
Да, но есть нюанс.

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

вместе с методом градиентного спуска (GD)

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

Можно ли заменить $\nabla_w L(w_k)$, используя, лишь информацию нулевого порядка о функции?
Да, но есть нюанс.
One can consider 2-point gradient estimator[a] $G$:

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

where $v$ is spherically symmetric.

[a] I suggest a nice presentation about gradient-free methods

# Пример: безградиентный градиентный спуск

Рассмотрим следующую задачу оптимизации

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

вместе с методом градиентного спуска (GD)

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

Можно ли заменить $\nabla_w L(w_k)$, используя, лишь информацию нулевого порядка о функции?
Да, но есть нюанс.
One can consider 2-point gradient estimator[a] $G$:

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

where $v$ is spherically symmetric.

[a] I suggest a nice presentation about gradient-free methods

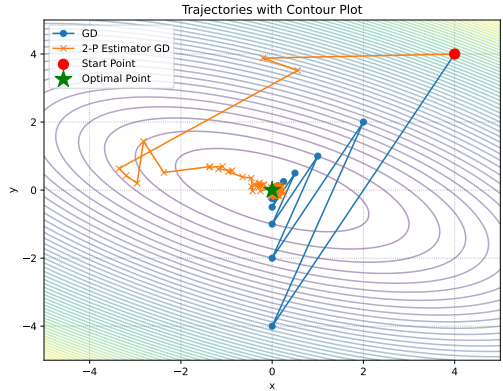


Рис. 4: ``Illustration of two-point estimator of Gradient Descent''

# Пример: конечно-разностный градиентный спуск

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

# Пример: конечно-разностный градиентный спуск

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

One can also consider the idea of finite differences:

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

Open In Colab ♣

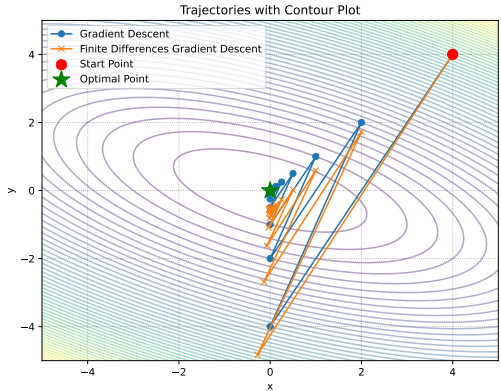


Рис. 5: ``Illustration of finite differences estimator of Gradient Descent''

# Проклятие размерности методов нулевого порядка

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

# Проклятие размерности методов нулевого порядка

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

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

where $G$ is a 2-point or multi-point estimator of the gradient.

# Проклятие размерности методов нулевого порядка

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

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

where $G$ is a 2-point or multi-point estimator of the gradient.

| | $f(x)$ - smooth | $f(x)$ - smooth and convex | $f(x)$ - smooth and strongly convex |
|---|---|---|---|
| GD | $\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{1}{k}\right)$ | $f(x_k) - f^* \approx \mathcal{O}\left(\frac{1}{k}\right)$ | $\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$ |
| Zero order GD | $\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{n}{k}\right)$ | $f(x_k) - f^* \approx \mathcal{O}\left(\frac{n}{k}\right)$ | $\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{nL}\right)^k\right)$ |

# Метод конечных разностей

The naive approach to get approximate values of gradients is **Finite differences** approach. For each coordinate, one can calculate the partial derivative approximation:

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

[1] 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.

# Метод конечных разностей

The naive approach to get approximate values of gradients is **Finite differences** approach. For each coordinate, one can calculate the partial derivative approximation:

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

**Question**

If the time needed for one calculation of $L(w)$ is $T$, what is the time needed for calculating $\nabla_w L$ with this approach?

# Метод конечных разностей

The naive approach to get approximate values of gradients is **Finite differences** approach. For each coordinate, one can calculate the partial derivative approximation:

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

**i** Question

If the time needed for one calculation of $L(w)$ is $T$, what is the time needed for calculating $\nabla_w L$ with this approach?

**Answer** $2dT$, which is extremely long for the huge scale optimization. Moreover, this exact scheme is unstable, which means that you will have to choose between accuracy and stability.

# Метод конечных разностей

The naive approach to get approximate values of gradients is **Finite differences** approach. For each coordinate, one can calculate the partial derivative approximation:

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

**Question**

If the time needed for one calculation of $L(w)$ is $T$, what is the time needed for calculating $\nabla_w L$ with this approach?

**Answer** $2dT$, which is extremely long for the huge scale optimization. Moreover, this exact scheme is unstable, which means that you will have to choose between accuracy and stability.

**Theorem**

There is an algorithm to compute $\nabla_w L$ in $\mathcal{O}(T)$ operations. [1]

[1] 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.