

# Децентрализованная распределенная оптимизация

Булат Шелхонов  
shelkhonov.bv@phystech.edu

Проект по методам оптимизации

В данном проекте я рассматривал задачу децентрализованной оптимизации. Была рассмотрена статья по оптимизации в меняющейся со временем сети, в которой был предложен метод проективного градиентного спуска. Я провел эксперименты по скорости сходимости алгоритма Консенсус, а также сравнил децентрализованный градиентный спуск с классическим на двух датасетах.

## 1 Идея

Задача распределенной оптимизации решает оптимизационные проблемы с использованием распределенных агентов. Основная идея заключается в том, чтобы распределить процесс оптимизации между несколькими агентами, каждый из которых обладает своей локальной информацией и принимает решения в соответствии с ней. Преимущества распределенной оптимизации по сравнению с классической:

- ускорение на больших объемах данных - особенно полезно в современном мире для обучения LLM и больших нейросетей наподобие Midjourney, Stable Diffusion и так далее
- масштабируемость - можно без особых усилий расширить сеть и получить больше мощностей
- отказоустойчивость - отказ одного узла не останавливает процесс оптимизации

Децентрализованная оптимизация - одна из подзадач распределенной оптимизации. В ней нет центрального узла, который координирует вычисления. Обмен информацией между узлами может быть сильно ограничен.

### 1.1 Задача распределенной оптимизации

Сформулируем классическую задачу машинного обучения:

$$f(x) = \frac{1}{N} \sum_{i=1}^N f(x, z_i) \longrightarrow \min_{x \in \mathbb{R}^d} \quad (1)$$

Можно распределить данные на  $M$  узлов и получить следующую задачу:

$$f(x) = \frac{1}{M} \sum_{i=1}^M f_i(x) = \frac{1}{M} \sum_{i=1}^M \left[ \frac{1}{N_i} \sum_{j=1}^{N_i} f(x, z_{i,j}) \right] \longrightarrow \min_{x \in \mathbb{R}^d}, \quad (2)$$

где  $N_1 + \dots + N_M = N$ .

Можно выделить несколько типов распределения:

- кластерные вычисления: у каждого узла примерно одинаковая мощность, равномерно делим данные  $N_i \approx N_j$ , локальные функции в некоторой степени будут похожи между собой
- коллаборативные вычисления: делим данные, возможно неравномерно по количеству, но по природе
- федеративное обучение: вычислительные устройства - пользовательские: ноутбуки, планшеты, телефоны

Существуют 2 типа коммуникаций:

- централизованная архитектура: можно получить точное усреднение по всем устройствам
- децентрализованная архитектура: точное усреднение не предусмотрено

## 2 Оптимизация в меняющейся со временем сети

В моем проекте я рассмотрел метод из статьи *Projected Gradient Method for Decentralized Optimization over Time-Varying Networks*[1]. Также я ориентировался на статью *Decentralized convex optimization over time-varying graphs: a survey*[2], в которой рассматриваются state-of-the-art методы децентрализованной оптимизации.

Рассмотрим задачу:

Нужно оптимизировать сумму выпуклых функций:

$$f(x) = \sum_{i=1}^n f_i(x) \longrightarrow \min_{x \in \mathbb{R}^d} \quad (3)$$

Сеть представляет связный граф из  $n$  вершин, при этом сеть меняющаяся:

$$\{\mathcal{G}_k\}_{k=1}^{\infty} \\ \mathcal{G}_k = (V_k, E_k)$$

Задачу 3 можно переформулировать в следующем виде:

$$F(X) = \sum_{i=1}^n f_i(x_i) \longrightarrow \min_{x_1=\dots=x_n}, \quad (4)$$

где  $X \in \mathbb{R}^{d \times n}$ , а  $x_i$  -  $i$ -й столбец матрицы  $X$ .

Введем следующие определения:

**Определение 2.1** Пусть  $\mathbb{X}$  - пространство  $\mathbb{R}^d$  с  $l_2$ -нормой, либо  $\mathbb{R}^{d \times n}$  с нормой Фробениуса. Дифференцируемая функция  $f : \mathbb{X} \rightarrow \mathbb{R}$  является:

- выпуклой, если  $\forall x, y \in \mathbb{X} \ f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$ ,
- $\mu$ -сильно выпуклой, если  $\forall x, y \in \mathbb{X} \ f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2$ ,
- $L$ -гладкой, если

$$\forall x, y \in \mathbb{X} \ \|\nabla f(y) - \nabla f(x)\| \leq L \|x - y\|$$

, что эквивалентно

$$\forall x, y \in \mathbb{X} \ f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2$$

**Предположение 2.2** Функции должны удовлетворять следующим условиям:

- функция  $f(x)$   $\mu_f$ -сильно выпуклая и  $L_f$ -гладкая.
- функции  $f_i(x)$  дифференцируемы.

Обозначим  $K = \text{Span}(\mathbf{1})$  - пространство, где лежит решение задачи.

Предлагается следующий алгоритм:

Алгоритм представляет собой градиентный спуск с проекцией на пространство  $K$ . Так как алгоритм децентрализованный, то вычислить точное усреднение не представляется возможным. Из-за такого ограничения делается приближенная проекция.

---

**Algorithm 1** Decentralized Projected GD

---

**Require:** Each node holds  $f_i(\cdot)$  and iteration number  $N$ .

- 1: Initialize  $X_0 = [x_0, \dots, x_0]$ , choose  $\varepsilon > 0$ .
  - 2: **for**  $k=0,1,\dots, N-1$  **do**
  - 3:    $Y_{k+1} = X_k - \gamma \nabla F(X_k)$ .
  - 4:    $X_{k+1} \approx \text{Proj}_K(Y_{k+1})$  with accuracy  $\varepsilon_1$ , i.e.  $\|X_{k+1} - \text{Proj}_K(Y_{k+1})\|^2 \leq \varepsilon_1$  and  $X_{k+1} - \text{Proj}_K(Y_{k+1}) \in K^\perp$ .
  - 5: **end for**
- 

## 2.1 Консенсус через Mixing матрицу

Консенсус можно искать с помощью mixing матрицы  $W \in \mathbb{R}^{n \times n}$ .

Так как сеть меняется, то имеем последовательность матриц  $\{W(k)\}_{k=1}^\infty$ . Обозначим  $W_b(k) := W(k)W(k-1) \dots W(k-b+1)$ .

**Предположение 2.3** *Mixing-матрица имеет следующие свойства:*

1.  $W(k)_{ij} = 0$ , если  $i \neq j$  и  $(i, j) \notin E$ .
2.  $W(k)\mathbf{1} = \mathbf{1}$  и  $\mathbf{1}^\top W(k) = \mathbf{1}^\top$ .
3. существует целое число  $B > 0$  такое, что  $\delta := \sup_{k \geq B-1} \delta(k) < 1$ , где  $\delta(k) = \sigma_{\max}\left[W_B(k) - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right]$ .

Первое условие говорит об отсутствии коммуникации, если нет связи между двумя вершинами. Второе, что если уже достигнут консенсус, то он должен остаться консенсусом. Третье, что многократное применение матриц  $W(k)$  приведет  $X$  к консенсусу.

Пример такой матрицы:

$$W_{ij} = \begin{cases} \frac{1}{\max(\deg i, \deg j)}, & (i, j) \in E \\ 1 - \sum_{l=1}^n W_{il}, & i = j \\ 0, & \text{else} \end{cases}$$

---

**Algorithm 2** Consensus

---

**Require:** Each node holds  $x_i^0$  and iteration number  $N$ .

- 1: **for**  $k=0,1,\dots, N-1$  **do**
  - 2:    $X_{k+1} = X_k W(k)$ .
  - 3: **end for**
- 

По сути, каждая вершина  $i$  будет вычислять свой  $x_i$  как линейную комбинацию векторов со своими соседями:  $W_{ii}x_i + \sum_{(i,j) \in E} W_{ij}x_j$ . И при многократном повторении этого действия сеть придет к консенсусу: на каждой вершину будут хранить приблизительно одинаковые данные.

## 2.2 Консенсус через Gossip матрицу

В статье [2] также рассказывается о эквивалентном поиске консенсуса, через двойственный метод. В нём используется последовательность gossip матриц  $\{\mathcal{L}^k\}_{k=0}^\infty$ .

**Предположение 2.4** *Эта последовательность удовлетворяет условиям:*

1.  $[\mathcal{L}^k]_{ij} = 0$ , если  $i \neq j$  и  $(i, j) \notin \mathcal{E}^k$
2.  $\text{Ker } \mathcal{L}^k \supseteq \text{Span}(\mathbf{1})$
3.  $\text{Im } \mathcal{L}^k \subseteq \{x \in \mathbb{R}^m : x_1 + \dots + x_m = 0\}$
4. Существует целое  $\tau > 0$  и  $\chi > 0$  такое, что для любого  $k \geq \tau-1$  и любого  $x \in \mathbb{R}^m$  и  $x_1 + \dots + x_m = 0$ , выполняется

$$\|\mathcal{L}_\tau^k x - x\|^2 \leq \left(1 - \frac{\tau}{\chi}\right) \|x\|^2$$

, здесь  $\mathcal{L}_\tau^k = \mathbf{I} - (\mathbf{I} - \mathcal{L}^k) \dots (\mathbf{I} - \mathcal{L}^{k-\tau+1})$

Получить gossip матрицу можно через mixing матрицу:  $L = \mathbf{I} - W$ . При этом матрица  $L$  удовлетворяет 2.4, тогда и только тогда, когда  $W$  удовлетворяет 2.3.

## 2.2.1 Вспомогательные формулы и утверждения

Альтернативный способ получить gossip матрицу - посчитать Лапласиан (матрицу Кирхгофа) графа:

$$[W]_{ij} = \begin{cases} \deg i, & i = j \\ -1, & (i, j) \in E \\ 0, & \text{else} \end{cases} \quad (5)$$

Свойства Лапласиана, которые используются в алгоритме:

- $W$  - симметричная положительная полуопределенная матрица
- если граф связный, тогда  $Wx = 0 \Leftrightarrow x_1 = \dots = x_n$ , то есть  $\text{Ker } W = \text{Span}(\mathbf{1})$ . Более того,  $\text{Ker } \sqrt{W} = \text{Span}(\mathbf{1})$

Пусть есть задача

$$f(x) \longrightarrow \min_{Ax=0} \quad (6)$$

Пусть  $y^*$  - решение двойственной задачи Лагранжа с минимальной нормой и  $R_y = \|y^*\|$ . Тогда 6 можно написать в виде задачи со штрафом:

$$f_A(x) = f(x) + \frac{R_y^2}{\varepsilon} \|Ax\|^2 \longrightarrow \min_{x \in \mathbb{R}^d} \quad (7)$$

**Лемма 2.5** Пусть  $x \in \mathbb{R}^d$  и  $f_A(x_N) - \min_x f_A(x) < \varepsilon$ .

Тогда

$$\begin{cases} f(x_N) - \min_{Ax=0} f(x) < \varepsilon \\ \|Ax_N\| < 2\varepsilon/R_y \end{cases}$$

## 2.2.2 Аппроксимация проекции

Основная часть алгоритма - приближение проекции  $Y$  на подпространство  $K$ .

Проекцию можно определить как решение следующей оптимизационной задачи:

$$\frac{1}{2} \|X - Y\|^2 \longrightarrow \min_{X \in K}$$

Пусть в данный момент сеть представляет собой граф  $\mathcal{G}$  с лапласианом  $W$ . Тогда задачу можно переписать так:

$$\frac{1}{2} \|X - Y\|^2 \longrightarrow \min_{X\sqrt{W}=0}$$

Более того, можно перенести ограничение  $X\sqrt{W} = 0$  как регуляризацию по утверждению 7:

$$\frac{1}{2} \|X - Y\|^2 + \frac{R^2}{\varepsilon_2} \|X\sqrt{W}\|^2 \longrightarrow \min_{X \in \mathbb{R}^{d \times n}} \quad (8)$$

Так как со временем сеть меняется, то мы работаем с последовательностью лапласианов  $\{W_k\}_{k=1}^\infty$ . Появляется последовательность  $H(X) = H_k(X)_{k=1}^\infty$ , где

$$H_k(X) = \frac{1}{2} \|X - Y\|^2 + \frac{R^2}{\varepsilon_2} \|X\sqrt{W}\|^2 \quad (9)$$

Распишем градиент  $H_k$ :

$$\nabla H_k(X) = X - Y + \frac{2R^2}{\varepsilon_2} XW$$

Теперь вспомним, что  $i$ -я колонка  $X$  и  $Y$  хранится на  $i$ -й вершине, соответственно,  $i$ -я колонка градиента будет в той же вершине:

$$\begin{aligned} [\nabla H_k(X)]_i &= [X]_i - [Y]_i + \frac{2R^2}{\varepsilon_2} [XW]_i \\ [XW]_i &= \deg i \cdot [X]_i - \sum_{j \neq i, (i,j) \in E_k} [X]_j \end{aligned} \quad (10)$$

Таким образом, получили подсчёт градиента по вершинам. Теперь можно считать проекции децентрализованно.

### 2.2.3 Примечание

На самом деле приведенные выше 2 способа коммуникации приводят к одному решению разными путями: многократном применении матрицы или минимизацией квадратичной функции.

## 3 Сложность алгоритма

Для упрощения анализа сложности алгоритмов добавим следующее предположение:

- все функции  $f_i$   $\mu_i$ -сильно выпуклые и  $L_i$ -гладкие

Определим константу  $r_0 = \|X_0 - \text{Proj}_K(X_0)\|$ .

**Теорема 3.1** После  $N = O\left(\frac{L_f}{\mu_f} \log\left(\frac{r_0^2}{\varepsilon}\right)\right)$  итераций алгоритм 1 с  $\varepsilon_1 = \frac{\mu_f^2}{13n^2 L_{\max}^2} \varepsilon$  возвращает  $X_N$  такой, что  $\|X_N - X^*\|^2 \leq \varepsilon$ .

**Теорема 3.2** Алгоритм 1 с  $\varepsilon_1 = \frac{\mu_f^2}{13n^2 L_{\max}^2}$  требует

$$N = O\left(\frac{L_f}{\mu_f} B \log\left(\frac{1}{\delta}\right)^{-1} \log\left(\frac{(\|\nabla F(X^*)\| + L_{\max}\|X_0 - X^*\|)n^2 L_{\max}^2}{\varepsilon \mu_f^2}\right) \log\left(\frac{r_0^2}{\varepsilon}\right)\right)$$

шагов, включая поиск проекции, чтобы получить такое  $X_N$ , что:

$$\|X_N - X^*\|^2 \leq \varepsilon$$

Эти теоремы говорят о том, что алгоритм сходится примерно экспоненциально относительно количества итераций.

## 4 Эксперименты

В данной части я провел эксперименты с алгоритмом поиска консенсуса 2. Также я реализовал логистическую регрессию по алгоритму 1 и сравнил ее с нераспределенной логистической регрессией на случайном датасете и на датасете MNIST.

### 4.1 Скорость сходимости Консенсуса

Для начала проверим, что алгоритм 2 действительно приводит нас к консенсусу. Для проверки я взял модель случайного графа Эрдеша-Реньи  $G(n, p)$ . Я рассматривал сходимость на 4-х случайных сетях: сеть могла быть постоянной и меняющейся со временем, а также разреженной и плотной. Для получения разреженного графа я использовал  $p = \frac{\log n}{n}$  (порог для связности графа), для плотного  $p = 0.3$ .

В качестве векторов я использовал случайные вектора  $x \in \mathbb{R}^{100}$ ,  $x_i \sim \mathcal{U}[-\frac{1}{\sqrt{100}}, \frac{1}{\sqrt{100}}]$ .

Как видно по графикам 1 и 2, консенсус очень быстро сходится.

Здесь считалось отклонение между значениями в узлах сети и истинным средним значений. Среднее отклонение =  $\frac{1}{n} \sum_{i=1}^n \|x_i - \bar{x}\|_2$ , максимальное отклонение =  $\max\{\|x_i - \bar{x}\|_2 \mid i = 1, \dots, n\}$ .

Примечательно, что у плотных графов увеличение количества вершин улучшает сходимость к среднему. Я объясняю это тем, что при большем количестве вершин в плотном графе соответственно больше соседей, тогда усредненное на вершине значение получается более близким к истинному среднему. Аналогично происходит в статистике - чем больше размер выборки, тем больше оценка среднего похожа на математическое ожидание.

Также на графиках видно, что максимальное отклонение уменьшается экспоненциально (ось Y на графиках - логарифмическая). Среднее отклонение тоже, кроме случая постоянной сети.

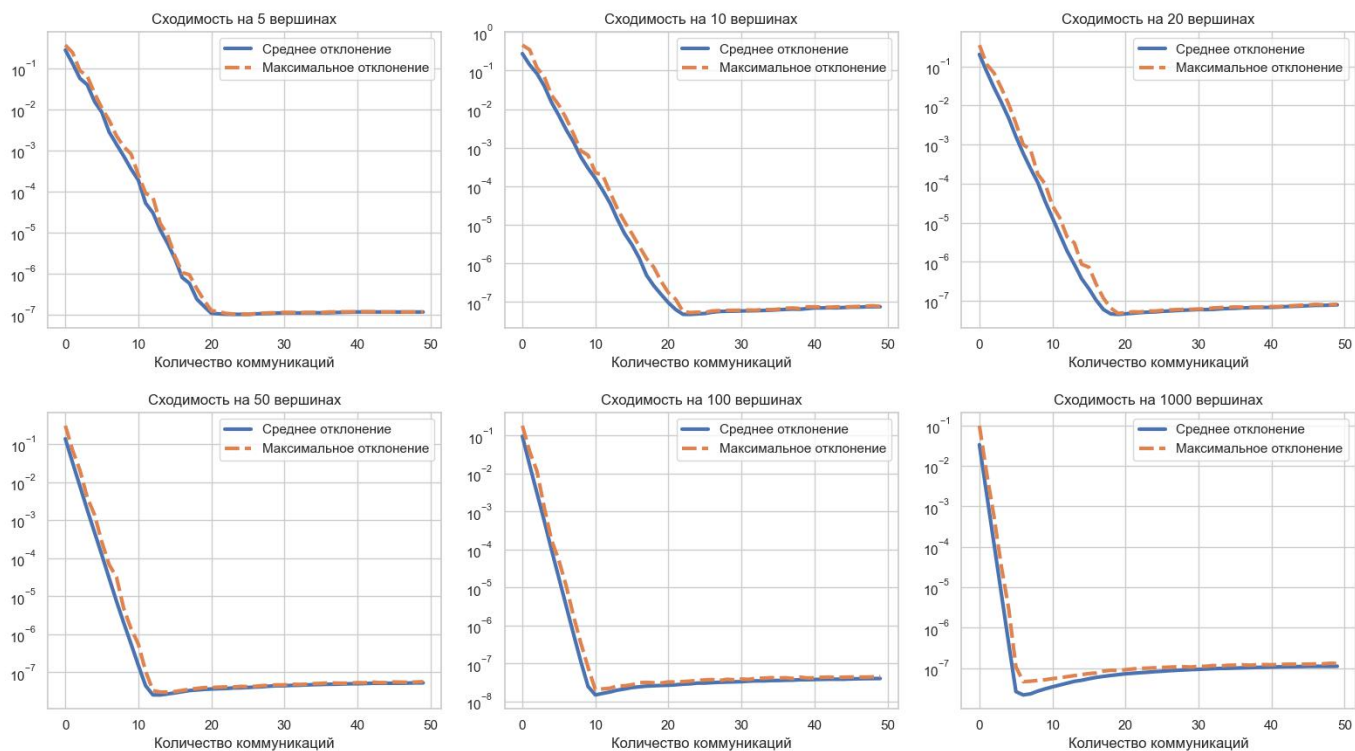


Figure 1: Сходимость консенсуса на плотной меняющейся случайной сети

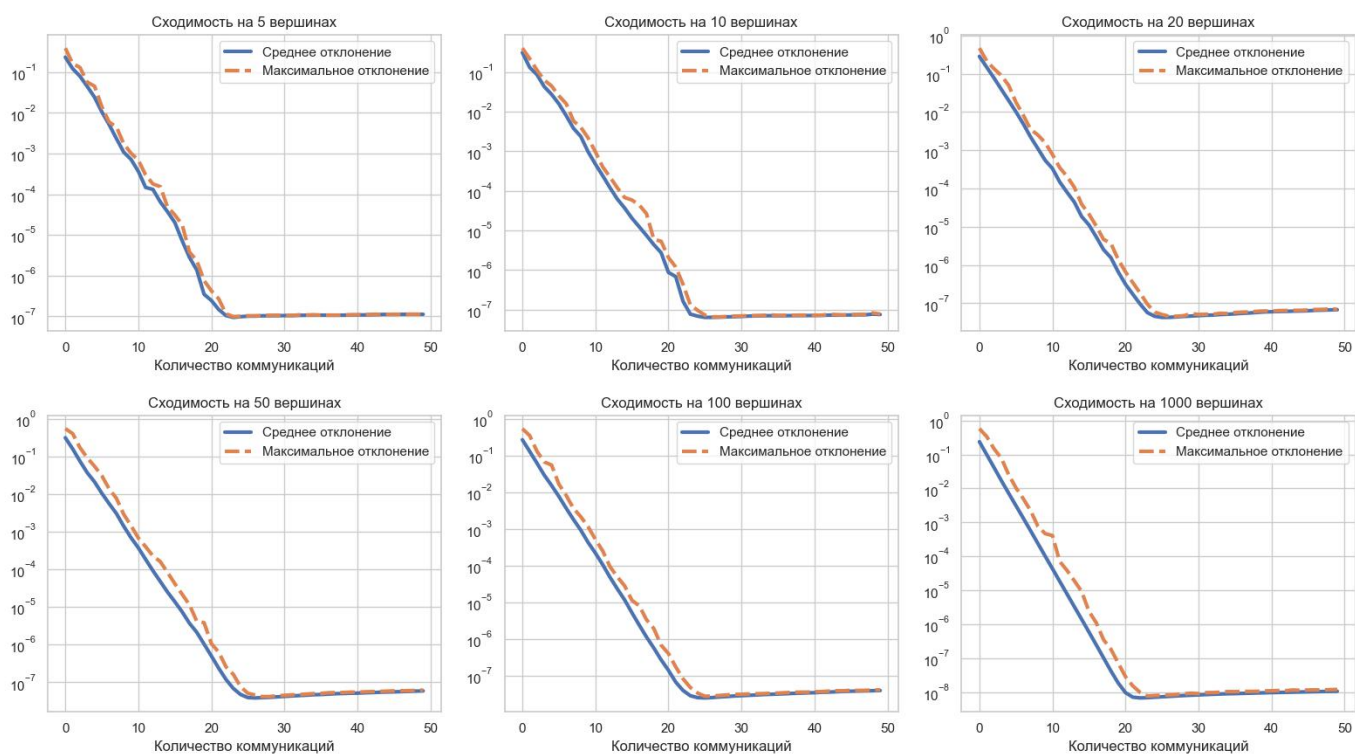


Figure 2: Сходимость консенсуса на разреженной меняющейся случайной сети

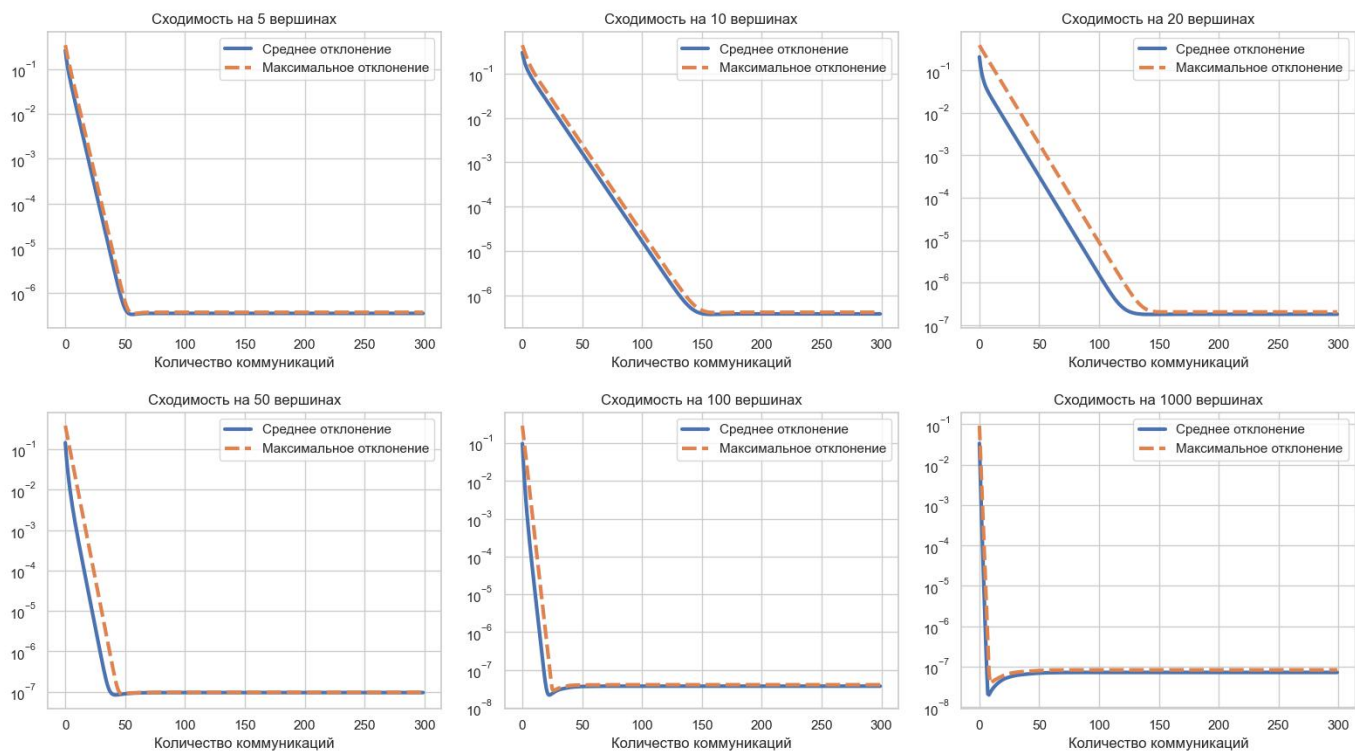


Figure 3: Сходимость консенсуса на плотной постоянной случайной сети

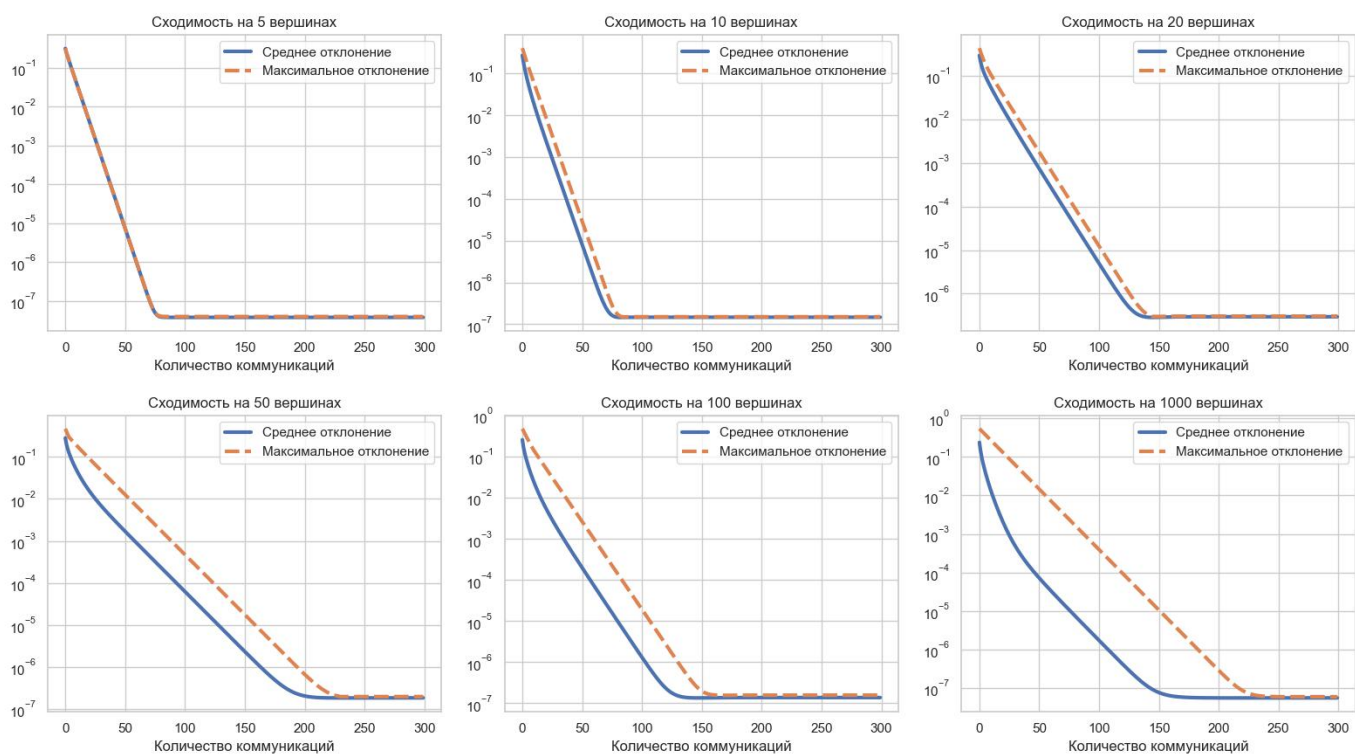


Figure 4: Сходимость консенсуса на разреженной постоянной случайной сети

## 4.2 Логистическая регрессия на сгенерированных данных

В данном разделе я сравниваю децентрализованную логистическую регрессию с обычной.

Информация об эксперименте:

- датасет: сгенерированный размера 5000 и с 100 признаками. Валидационная часть составляет 20% от датасета.



- оптимизатор: Adam,  $\text{learning\_rate} = 10^{-3}$  для децентрализованного алгоритма и  $10^{-2}$  для бейслайна.
- сеть: модель Эрдеша-Реньи  $G(n, \frac{\log n}{n})$ , где  $n = 100$ .
- функция потерь: бинарная кросс-энтропия с L2-регуляризацией с коэффициентом  $\frac{1}{20}$ .

Код для генерации датасета:

```
1 from sklearn.datasets import make_classification
2
3 X, y = make_classification(
4     5000, n_features=100, n_informative=95, n_redundant=5, random_state=1
5 )
6
7 X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=1)
```

Код для генерации mixing матрицы коммуникаций  $W$ :

```
1 import mlx.core as mx
2 import networkx as nx
3 import numpy as np
4
5 def generate_network(n, p):
6     g = nx.fast_gnp_random_graph(n, p)
7     while not nx.is_connected(g):
8         g = nx.fast_gnp_random_graph(n, p)
9     mixing_matrix = np.zeros([n, n])
10    adj = nx.adjacency_matrix(g)
11    deg = np.sum(adj, 1)
12    for u, v in g.edges:
13        max_deg = max(deg[u], deg[v])
14        mixing_matrix[u, v] = 1 / max_deg
15        mixing_matrix[v, u] = 1 / max_deg
16
17    for u in range(n):
18        mixing_matrix[u, u] = 1 - mixing_matrix[u].sum()
19
20    return mx.array(mixing_matrix)
```

Я построил основные метрики классификации на каждом шаге обучения на валидационной выборке. При этом я рассматривал модели с 0, 1, 10, 30 раундами коммуникаций в консенсусе.

На графиках 5 видно, что все алгоритмы показывают одинаковые результаты. При этом интересно, что если не вызывать алгоритм Консенсус (0 раундов коммуникаций), то качество не падает. Я объясняю это тем, что под данный датасет легко обучиться.



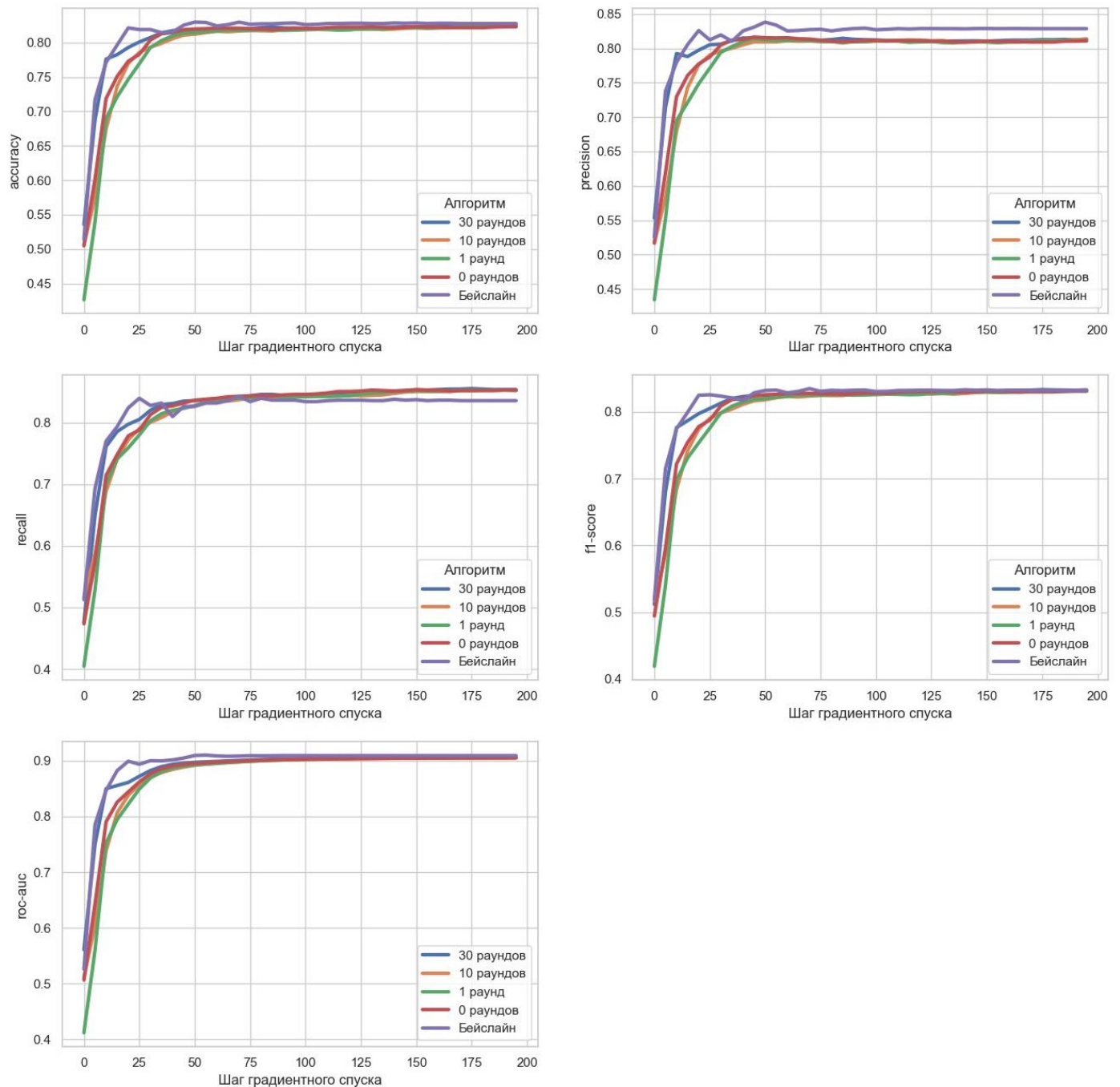


Figure 5: Сходимость консенсуса на разреженном постоянном случайном графе, размерность = 100

### 4.3 Логистическая регрессия на MNIST

Я взял подмножество датасета MNIST размера 20000 для обучения и 10000 для валидации.

В этом эксперименте параметры такие же как в предыдущем, кроме `learning_rate` - здесь я везде поставил  $10^{-3}$ .

На реальном датасете появилась значительная разница между 0 раундами коммуникации и  $> 0$  раундами: алгоритм с 0 раундами дошёл до ассигасу около 0.78, все остальные достигли почти 0.9 ассигасу.

При этом 1 раунд коммуникации всё равно даёт такое же качество, что и алгоритмы с большим числом раундов. В итоге, всё быстро сошлось к такому же результату, как и бейслайн.

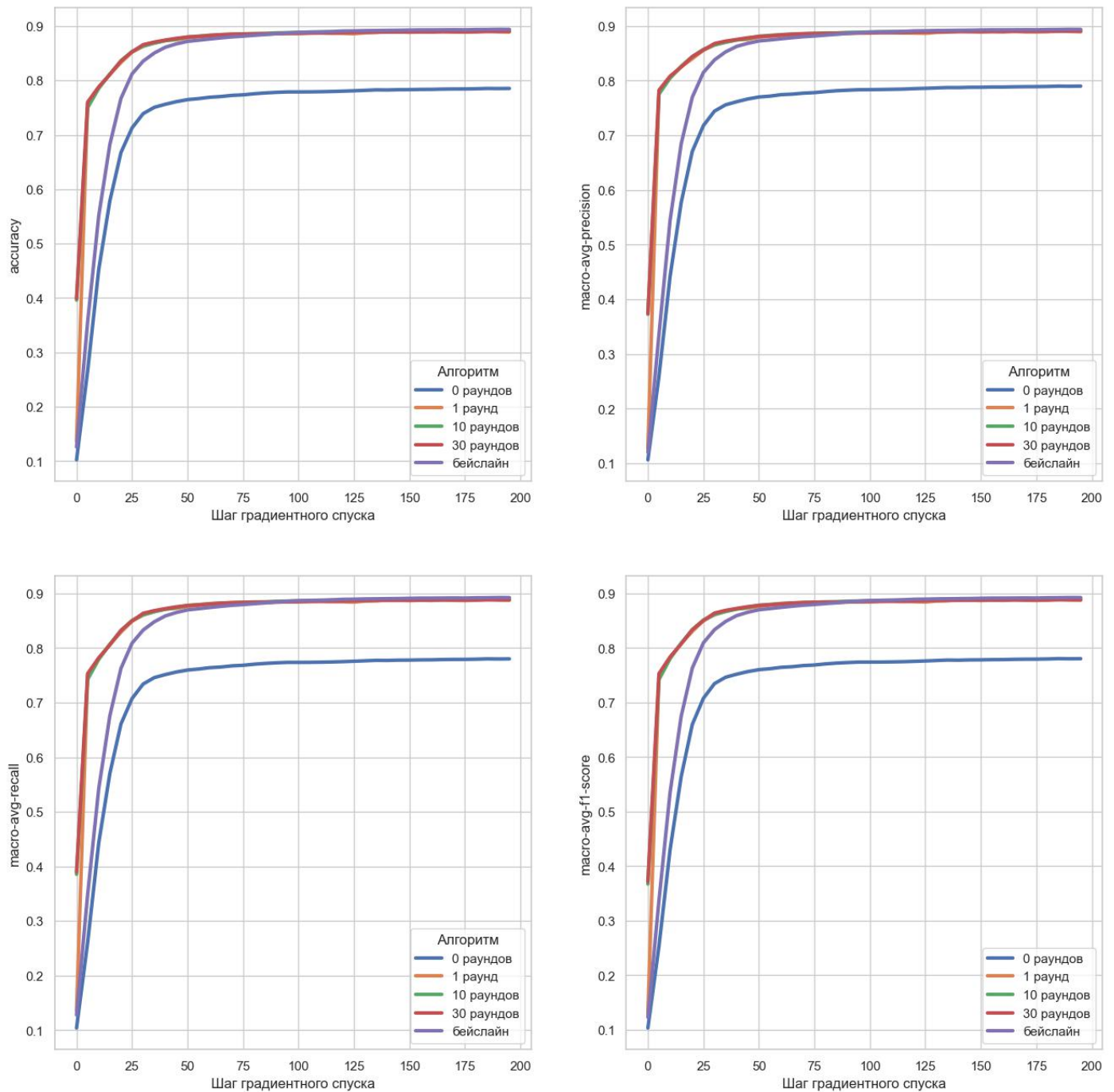


Figure 6: Результаты на датасете MNIST

## 5 Вывод

Данный алгоритм позволяет децентрализованно оптимизировать функцию без значительных потерь в качестве и скорости сходимости по сравнению с централизованным. Эксперименты показали, что для достижения сопоставимого с централизованным алгоритмом качества не обязательно проводить больше число раундов коммуникаций, если сеть похожа на модель Эрдеша-Реньи.

## References

- [1] Alexander Rogozin and Alexander Gasnikov. Projected gradient method for decentralized optimization over time-varying networks, 2020.
- [2] Alexander Rogozin, Alexander Gasnikov, Aleksander Beznosikov, and Dmitry Kovalev. *Decentralized Convex Optimization over Time-Varying Graphs*, page 1–17. Springer International Publishing, 2023.