

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Исследовательский проект на тему:

Распределенные алгоритмы с редукцией дисперсии для вычисления барицентров Васерштейна

Выполнил студент:

группы БПМИ202, 4 курса Аржанцев Андрей Иванович

Принял руководитель ВКР:

Двинских Дарина Михайловна

Старший научный сотрудник

Факультет компьютерных наук НИУ ВШЭ

Москва 2024

Содержание

Аннотация	3
1 Вступление	4
2 Обзор литературы	5
2.1 Обзор существующих алгоритмов для задач оптимизации	5
2.2 Обзор методов редукции дисперсии	7
2.3 Прочее	8
3 Постановка задачи	10
3.1 Изначальная формулировка	10
3.2 Переформулировка задачи	10
3.3 Невклидовая постановка задачи	13
4 Обзор итогового алгоритма	15
4.1 Итерации алгоритма	15
4.2 Семплирование	17
4.3 Реализация	20
4.4 Обоснование и скорость сходимости	22
4.5 Распределенный алгоритм	24
5 Эксперименты	24
5.1 Смесь гауссиан	24
5.2 MNIST	27
6 Результаты	28
Список литературы	29

лучше всё на русской



БВ

Аннотация

Данная работа посвящена задаче о барицентрах Вассерштейна (WB), которая является важным приложением задачи оптимального транспорта (OT). WB можно рассматривать как задачу оптимизации седловой точки, для решения которой разработано и изучено множество методов. В недавних статьях, относящихся к этой теме, были улучшены оценки сложности задачи, полученные с помощью таких методов оптимизации, как MirrorProx или Dual Extrapolation. Другие исследования в области задач оптимизации седловой точки демонстрируют, что алгоритмы могут быть улучшены за счет надлежащего использования методов ~~уменьшения дисперсии~~ ~~рекурсы~~ ~~уменьшения дисперсии~~. Основной целью работы является исследование неевклидовой постановки задачи WB и обобщение новейших алгоритмов для нее. Также рассматривается сравнение полученных и имеющихся методов.

Ключевые слова

Оптимальный транспорт, барицентры Вассерштейна, вариационные неравенства, редукция дисперсии, задача оптимизации седловой точки, дивергенция Брегмана, MirrorProx алгоритм
серловые задачи

1 Вступление

(хорошо бы что-то про
оптим. транспорт добавить:
Лонгин + Калгоровиц)

Барицентры Васерштейна - это часто используемый инструмент в задачах оптимального транспорта [1]. Например, он может быть использован для вероятностной многоуровневой кластеризации [9] или изучения представлений в различных задачах [17]. Задача о барицентрах Васерштейна предполагает поиск распределения вероятностей, которое служит оптимальным по отношению к заданным распределениям вероятностей в пространстве Васерштейна. Задача является NP сложной [15], поэтому для ее решения исследуются преимущественно итерационные методы и их различные варианты оптимизации.

К задаче о барицентрах Васерштейна можно подходить с различных точек зрения. Недавнее исследование [6] показывает, что рассмотрение проблемы WB как проблемы с седловой точкой и применение двойной экстраполяции привело к улучшениям по сравнению с предыдущими современными методами для WB, FastIBP [8] и ускоренного IBP [14]. Целью данной работы является изучение алгоритмов для решения задачи в неевклидовых условиях, используя идеи исследователей, занимающихся этой ~~задачей~~ проблемой в евклидовых пространствах.

Результатом этой работы стал алгоритм, использующий ~~методы~~ редукции дисперсии ~~для решения задачи~~, демонстрирующий улучшенную скорость сходимости по сравнению с некоторыми предыдущими версиями алгоритма. Помимо практической полезности результата, благодаря ускоренному алгоритму решения задачи WB, ожидается также получение ценных теоретических знаний, применимых к смежным задачам. Кроме того, будут представлены численные эксперименты с соответствующими наборами данных.

Для улучшения оценки сложности алгоритма, в дополнение к анализу существующих алгоритмов, требуется изучение методов оптимизации для аналогичных задач в целом, чтобы выявить соответствующие идеи, исправить их и адаптировать к конкретной задаче. Эти идеи в основном связаны с различными методами уменьшения дисперсии, а также скорость алгоритма может быть улучшена методами распределенной оптимизации.

Последующая часть статьи структурирована следующим образом. В разделе 2 будут рассмотрены связанные с ~~нашим~~ исследованием литература. В разделе 3 будет разобрана формулировка задачи и необходимые в дальнейшем выкладки по виду задачи. В разделе 4 будет описан и доказан представленный алгоритм. В разделе 5 будет рассказано о проведенных экспериментах и их результатах. В разделе 6 будут сформулированы итоговые результаты и вывод по всей проделанной работе.

2 Обзор литературы

2.1 Обзор существующих алгоритмов для задач оптимизации

В основном нас будут интересовать алгоритмы для решения вариационных неравенств. Эта категория задач включает в себя оптимизационные задачи о седловой точке, одной из которых как раз является задача о барицентрах Васерштейна. Подробно вывод переформулировок будет позже.

Как уже было сказано выше, почти все подобные алгоритмы являются итеративными и находят приближенное решение для заранее заданной точности. Самый известный, простой и популярный такой алгоритм - градиентный спуск, но здесь будет правильней упомянуть также более продвинутые методы.

Алгоритм **ExtraGradient** [12] - метод выпуклой оптимизации для вариационных неравенств, который подобен методу градиентного спуска, но позволяет находить седловые точки. Первоначально он был представлен для задач в евклидовом пространстве, но может быть распространен и на неевклидовую постановку задачи. Общая идея состоит в том, чтобы выполнять итерации с комбинацией шага градиента и проекции на соответствующее пространство. Одна итерация этого алгоритма для классической задачи минимизации некоторой функции f выглядит следующим образом

$$z_k = P_Q(z_k - \mu f'(z_k)),$$

$$z_{k+1} = P_Q(z_k - \mu f'(\hat{z}_k)),$$

где μ - длина шага, P_Q - оператор проекции на соответствующее пространство. Нахождение седловой точки с помощью этого алгоритма делается, благодаря одновременному применению подобных итераций к обеим переменным - в одном случае по градиенту функции, в другом в направлении обратному градиенту.

Алгоритм **MirrorProx** [11] - метод оптимизации для седловых точек, который тесно связан с теорией двойственности. Он использует двойственную функцию и двойственное пространство, полученные из исходной постановки задачи, из-за чего может быть применен в неевклидовой постановке. Одной из известных постановок задач, в которых алгоритм может быть применен, является Симплексная постановка, подразумевающая как раз работу в симплексных пространствах, подобными условиями обладает и задача о барицентрах Васерштейна. Одна итерация MirrorProx алгоритма для вариационных неравенств выглядит

последить, неочевидно

следующим образом:

$$\hat{x}^{k+1} = \operatorname{argmin}_{x \in X} (\tau \langle F(x^k), x - x^k \rangle + D_X(x, x^k))$$

и мы можем отсюда $\hat{x}^{k+1} = \operatorname{argmin}_{x \in X} (\tau \langle F(\hat{x}^{k+1}), x - x^k \rangle + D_X(x, x^k))$,

здесь D как раз дивергенция на пространстве X , заданная при помощи введенной на множестве зеркального отображения, которое при этом должно являться 1-строго выпуклой функцией. *Отметим, что первую операцию называют descent step (шаг спуска), ведь она является аналогом шага градиентного спуска на данном множестве относительно введенного зеркального отображения, а вторую - mirror step (зеркальный шаг), идейно в ней производится уточнение значение градиента, из-за перехода в другое пространство.*

Отметим, что наш алгоритм для решения задачи о барицентрах Васерштейна будет являться как раз вариацией алгоритма MirrorProx с добавлением в него техники *reduction* дисперсии.

Существует также множество других алгоритмов, некоторые из которых я в этом обзоре опишу очень кратко, тем не менее их также можно применять для конкретной задачи и добавлять в них техники редукции дисперсии, подобно тому, как это делается в нашей работе.

Алгоритм **Dual Extrapolation** [16]. Основная идея, заключается в одновременном уточнении прямых и двойственных оптимизируемых переменных. Сначала в двойственном пространстве мы делаем шаг, аналогичный шагу градиентного спуска, после чего экстраполируем полученное значение с помощью значений полученных, на прошлых итераций, и применяем полученную оценку для двойственной переменной в уточнении оценка для прямой переменной.

Алгоритм **Forward-Backward-Forward** [5]. В постановках данного алгоритма важно, чтобы оптимизируемый функционал можно было разделить на две части (например, для нашей задачи это условие выполняется). Далее для работы с уже двумя оптимизируемыми переменными мы делаем обычный градиентный шаг для второй переменной по значениям первой, после чего делаем обратный шаг с помощью проксимальной функции для пространства второй переменной, и наконец последняя операция состоит в еще одном шаге, который использует обе оценки, посчитанные на прошлых операций, сдвигаясь на разность значений градиентов в соответствующих точках.

2.2 Обзор методов редукции дисперсии

дисперсии

Методы редукция дисперсии в общем случае призваны уменьшать разброс в оценках γ для градиентов a [7]. Простым примером задачи, где эта техника может быть применима, является градиентный спуск. Пересчет всего градиента на каждой итерации является ресурсозатратным, в то время как стохастический градиентный спуск имеет маленькую скорость сходимости, как раз из-за большой дисперсии. Рассмотрим, как можно ~~пытаться~~ разрешить обе этих проблемы - трудозатратность пересчета оценки и увеличение дисперсии для упрощенных вариаций, как SGD.

классических

Одним из наивных методов можно назвать *mini – batching* - по сути обобщение стохастического градиентного спуска, где вместо одной координаты из градиента берется фиксированный достаточно маленький набор координат, обеспечивающий необходимое уменьшение дисперсии. Другим популярным методом является использование так называемого *momentum* - в этом методе градиент, посчитанный на предыдущем шаге учитывается с каким-то весом в оценке для градиента текущего шага. Оба этих метода довольно простые, тем не менее существует множество более подходящих для нашей задачи вариаций методов редукции дисперсии.

SAGA - метод редукции дисперсии, при котором вместо аппроксимации самого градиента используется специальная несмещенная оценка этого градиента:

$$g = \nabla f_i(x_i) - \nabla f_i(\hat{x}_i) + \frac{1}{n} \sum_{j=1}^n \nabla f_j(\hat{x}_j),$$

здесь \hat{x} - точка с уже посчитанным значением градиента. Для итеративных алгоритмов, использующего этот метод, значение x не сохраняется; вместо этого значение градиента в этой точке v сохраняется и обновляется одновременно, что приводит к следующему шагу алгоритма:

$$g_k = g_{k-1} + \frac{v_i - v}{n}$$

$$x_{k+1} = x_k - \nu(g_k + v_i - v),$$

где v на каждом шаге инициализируется предыдущим v_i , v_i - i -ая координата градиента в соответствующем x , и i выбирается случайным образом на каждом шаге.

это седловая точка!

Данный метод уже успешно применялся к задачам о седловой точке [ссылка]. В дополнение к стандартному методу были использованы *mini – batching* и повторная выборка для пересчитываемых координат.

SVRG - метод, который считается менее требовательным к памяти, чем SAGA, по-

скольку он сохраняет значения указанного x вместо значений градиента для этой точки. Недостатком этой функции является то, что внутри основного цикла требуется дополнительный цикл, а его длина является гиперпараметром, который следует настроить. В основном цикле x_0 инициализируется с помощью x из предыдущего шага, вычисляется и сохраняется полный градиент v в x_0 . Внутренний цикл выглядит следующим образом:

$$g_k = \nabla_i(x_k) - v_i + v$$

$$x_{k+1} = x_k - \varepsilon g_k,$$

где i выбирается случайным образом для каждого k .

Именно вариация данного метода и будет использоваться в нашем итоговом алгоритме, подобно тому как она применялась в статье [2] для неевклидовой постановки задачи о вариационном неравенстве, но также идея встречалась в других работах [3]. Правильное применение подобных техник редукции дисперсии подразумевает также подбор корректных оценок для градиента, которая будет немного отличаться от приведенных выше, где оценка есть по сути случайная проекция на одну из координат градиента. Далее в работе мы будем называть эти методы оценки градиента - семплирование (от англ. sample - выборка), ведь оно основано на выборе случайной оценки из изначально заданного распределения.

2.3 Прочее

Важной деталью постановки оптимизационной задачи является геометрическая структура множества, в котором находятся переменные. Структуры этого множества можно задать разными способами, основными из которых являются - евклидовая постановка и постановка по Брегману [4].

Евклидодовая постановка, пожалуй наиболее хорошо интерпретируемая, представляет собой оптимизационную задачу, в которой множество наделено евклидовой структурой, то есть используется расстояние $\|x - y\|_2$ и норма $\|x\|_2$. Для таких задач, например, будет работать обычный или стохастический градиентный спуск, или, например, стандартный вид *extragradient* алгоритма.

В постановке по Брегману геометрия множества задается специальной строго выпуклой функцией f , и соответствующей ей дивергенцией Брегмана. В каком-то смысле евклидовая постановка может быть также представлена в этом виде при выборе функции f , равной обычной евклидовой норме. Хорошими примерами задач, для которой постановка по Брег-

ману будет более оптимальной, являются как оптимизационные задачи на вероятностных распределениях, где структуру множества как раз логичней выбрать не стандартной. Алгоритмы для задач в постановке по Брегману чаще являются более сложными, в том числе сложность зависит от выбора функции f . В то же время анализ алгоритма в такой нестандартной постановке, часто приводит к улучшениям в скорости сходимости.

Наконец, закончим обзор литературы двумя статьями наиболее релевантными нашему исследованию, а именно недавней статье [6], предлагающей улучшенные алгоритмы для задачи о барицентрах Васерштейна, а также статье [2] о применении техник редукции дисперсия в общей постановке задач о вариационных неравенствах.

В первой статье ссылка предлагается два новых метода, сделанных на основе алгоритмов *MirrorProx* и *DualExterpolation*. Итоговая скорость сходимости предложенных методов является следующей

$$\text{Complexity}_{\text{MirrorProx}} = \tilde{O}\left(\frac{mn^2\sqrt{n}\|C\|_\infty}{\varepsilon}\right)$$
$$\text{Complexity}_{\text{DualExterpolation}} = \tilde{O}\left(\frac{mn^2\|C\|_\infty}{\varepsilon}\right)$$

Ценность данной работы, помимо наличия в ней алгоритмов для сравнения, заключается в упомянутых в ней переформулировках для задачи о барицентрах Васерштейна.

Вторая статья в то же время предлагает нам потенциальные подходы к имплементации техник редукции дисперсии. Как мы увидим позже из переформулировок задачи о барицентрах Васерштейна, предложенные в статье алгоритмы могут быть адаптированы под эту задачу.

Авторы статьи предлагают сразу несколько алгоритмов и их анализ, в том числе и вариацию алгоритма *MirrorProx*, которую мы и будем адаптировать для задач о барицентрах Васерштейна. Тем не менее остальные алгоритмы также достойны внимания и могут быть рассмотрены для будущего анализа, как, например, реализация метода *SVRG* для алгоритма *DualExterpolation* в евклидовой постановке, где авторам удалось избавиться от внутреннего цикла при семплировании, используя последовательно два разных семплирования на каждой итерации алгоритма.

Отметим также, что в статье упоминается реализация для билинейных задач, таких как линейная оптимизация с ограничениями, которые близки к задаче о барицентрах Васерштейна, благодаря чему многие аспекты в нашем итоговом алгоритме были позаимствованы из данной статьи и переиспользованы в анализе.

3 Постановка задачи

3.1 Изначальная формулировка

Наконец переходим к своей задачи, в этой части мы раскроем, что она вообще из себя представляет.

Предположим, что у нас есть два категориальных распределения p и q размерности n и матрица затрат C . Пусть U - набор матриц X , где $X \succeq 0$, $X\mathbf{1} = p$, $X^T\mathbf{1} = q$ ($\mathbf{1}$ равен $1 \times n$ вектор из единиц). Тогда можем определить такое расстояние между ними, которое как раз задает задачу об оптимальном транспорте.

$$W(p, q) = \min_{x \in U(p, q)} \langle C, X \rangle,$$

Барицентром Васерштейна набора категориальных распределений по сути называются оптимальные с точки зрения среднего расстояния точки из этого же пространства распределений. Более формально, пусть есть m различным категориальным распределениям $Q = \{q_1, q_2, \dots, q_m\}$, каждое из которых лежит в Δ_n , тогда барицентром Васерштейна, а также оптимизационную задачу, называют такое выражение:

$$WB(Q) = \operatorname{argmin}_{p \in \Delta(n)} \frac{1}{m} \sum_{q_i \in P} W(q_i, p)$$

Нетрудно заметить, например, что нахождение значение $WB(P)$ является выпуклой задачей оптимизации. Более детальный анализ данной постановки будет получен после некоторых ее переформулировок.

3.2 Переформулировка задачи

Сначала сведем задачу к более удобной для работы с методами оптимизации, а точнее покажем ее переформулировку в виде задачи об оптимизации седловой точки и в виде вариационного неравенства. Эти классы задач, как уже упоминалось в разделе с обзором литературы, широко изучены и имеют универсальные методы, которые и будут применяться для задачи барицентров Васерштейна.

Задача оптимизации седловой точки - это задача оптимизации, в общем виде записывающаяся, как

$$\min_{x \in X} \max_{y \in Y} F(x, y) + g_1(x) - g_2(y), \quad (1)$$

где $f(x)$ выпуклая по x и вогнутая по y , g_1 и g_2 выпуклые и являются по сути регуляризаторами.

Вариационное неравенство есть следующая задача оптимизации

$$\text{find } z_{opt} \text{ such that } \forall z : \langle F(z), z - z_{opt} \rangle + g(z) - g(z_{opt}) \geq 0 \quad (2)$$

\nwarrow на русской

Переформулировка задачи о Барицентрах Васерштейна в виде задачи о седловой точке встречалась в статьях [6] и [10]. Для начала воспользуемся фактом из второй статьи, где показывается, что задача оптимального транспорта может быть переписана в таком виде:

$$\begin{aligned} W(p, q) &= \min_{X \in U(p, q)} \langle C, X_i \rangle \Leftrightarrow \\ &\Leftrightarrow W(p, q) = \min_{x \in \Delta_{n^2}} \max_{y \in [1, 1]^{2n}} (d^T x + 2\|d\|_\infty (y^T A x - b^T y)) \end{aligned} \quad (3)$$

нет от студента

Здесь d - вектор длины n^2 , который есть векторизованная матрица C , $b = \begin{bmatrix} p \\ q \end{bmatrix}$. Также здесь и далее Δ_k - симплекс размерности k . A - матрица встречаемости ребер соответствующего двудольного графа, В нашем случае это полный двудольный граф солями размера n , поэтому ее вид однозначный, например, для $n=3$ A имеет такой вид

$$\left[\begin{array}{cccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right] .$$

Теперь аналогично первой статье выведем отсюда искомую переформулировку задачи о Барицентрах Васерштейна через переформулировку задачи об оптимальном транспорте. Пусть даны распределения q_1, \dots, q_m , задача найти для них барицентр Васерштейна, формально можно записать, как следующая задача оптимизации:

$$p_{opt} = \arg \min_{p \in \Delta_n} \frac{1}{m} \sum_{i=1}^m W(p, q_i) .$$

Будем минимизировать значение, необходимый для этого p также найдется алгоритмом. Перепишем каждое слагаемое по формуле (3):

$$\min_{p \in \Delta_n} \frac{1}{m} \sum_{i=1}^m \min_{x_i \in \Delta_{n^2}} \max_{y_i \in [1, 1]^{2n}} (d^T x_i + 2\|d\|_\infty (y_i^T A x_i - b^T y_i)) . \quad (4)$$

← Наконец, мы можем привести это к виду (2), для этого введем пространства $X = \prod_{i=1}^m \Delta_{n^2} \times \Delta_n, Y = [-1, 1]^{2mn}$, которые представляют собой область значений x_1, \dots, x_m, p и y_1, \dots, y_m . Перепишем все оставшиеся члены формулы в матричном виде и получим следующую задачу

$$\min_{x \in X} \max_{y \in Y} \frac{1}{m} (D^T x + 2\|d\|_\infty (y^T \hat{A} x - B^T y)), \quad (5)$$

где

$$D = \begin{bmatrix} d^T & \dots & d^T & 0_n^T \end{bmatrix},$$

$$b = \begin{bmatrix} 0_n^T & q_1^T & 0_n^T & q_2^T \dots & 0_n^T & q_m^T \end{bmatrix},$$

$$\hat{A} = (A^m, M^m)$$

$$A^m = \begin{bmatrix} A & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & A & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & A \end{bmatrix}, M^m = \begin{bmatrix} M \\ M \\ \vdots \\ M \end{bmatrix},$$

$$M = \begin{bmatrix} -I_n & 0_{n \times n}^T \end{bmatrix}^T, \mathbf{0} = 0_{2n \times n^2}$$

Нетрудно убедиться, что это действительно задача о седловой точке, так как полученная функция $F(x, y) = \frac{1}{m} (D^T x + 2\|d\|_\infty (y^T \hat{A} x - B^T y))$ выпуклая по x и вогнутая по y . В то же время функции g_1 и g_2 здесь можно представить как тождественный ноль, что сильно упростит дальнейшие преобразования.

Теперь перейдем к переформулировке в виде вариационного неравенства. Сославшись на статью [13], где переформулировка как раз делается от задачи о седловой точке к вариационному неравенству. Оказывается, что определив

$$F(z) = \hat{F}(x, y) = [\nabla_x F(x, y), -\nabla_y F(x, y)]$$

$$g(z) = \hat{g}(x, y) = g_1(x) + g_2(y) ,$$

Вариационное неравенство относительно z имеет в точности те же решения, что и соответствующая данным $F(x, y), g_1(x), g_2(y)$ задача о седловой точке.

$$\nabla_x F(x, y) = \nabla_x \frac{1}{m} (D^T x + 2\|d\|_\infty (y^T \hat{A}x - B^T y)) = \frac{1}{m} (D + 2\|d\|_\infty \hat{A}^T y),$$

$$\nabla_y F(x, y) = \nabla_y \frac{1}{m} (D^T x + 2\|d\|_\infty (y^T \hat{A}x - B^T y)) = \frac{1}{m} 2\|d\|_\infty (\hat{A}x - B).$$

← Так как g в нашем случае тождественный ноль, получаем следующую переформулировку (сразу разделим x и p и будем делать так и далее):

find $x_{opt} \in X, y_{opt} \in Y$ such that $\forall x \in X, y \in Y :$

$$\langle [\frac{1}{m}(D + 2\|d\|_\infty (A^m)^T y), \frac{1}{m}2\|d\|_\infty (B - A^m x - mMp), \sum_{i=1}^m (y_i)_{1,\dots,n}], [x - x_{opt}, y - y_{opt}, p - p_{opt}] \rangle \geq 0 \quad (6)$$

$$[x - x_{opt}, y - y_{opt}, p - p_{opt}] \rangle \geq 0$$

← Алгоритмы для решения задачи о Барицентрах Васерштейна в формулировках (5) и (6) и будут рассматриваться далее для построения алгоритмов.

3.3 Неевклидовая постановка задачи

билингв

Для работы с задачей о барицентрах Васерштейна в неевклидовой постановке необходимо изначально обговорить свойства тех пространств, в которых мы работаем.

На множестве X мы предполагаем такую норму:

$$\|x\|_X = \sqrt{\sum_{i=1}^m \|x_i\|_1^2 + m\|p\|_1^2}.$$

Также мы наделяем X прокси-функцией и соответствующей ей дивергенцией Брегмана:

$$d_X(x) = \sum_{i=1}^m x_i \ln x_i + mp \ln p,$$

$$B_X(x^1, x^2) = \sum_{i=1}^m (x_i^1 \ln \frac{x_i^1}{x_i^2}) + mp^1 \ln \frac{p^1}{p^2},$$

Заметим, что $d_X(x)$ есть по сути суммарная отрицательная энтропия для x_i и p , а дивергенция по сути является дивергенцией Кульбака-Лейблера, которая помимо многое прочего часто используется как раз для симплексных постановок оптимизационных задач. Такую дивергенцию просто минимизировать, что пригодится нам в будущем для алгоритма. Для

нашей задачи также важно, чтобы $d_X(x)$ была 1-строго выпуклой, то есть

$$d_X(x + \varepsilon)f(x) + \nabla f(x)^T \varepsilon + \frac{1}{2}\|y - x\|_2^2.$$

Это гарантируется нам, из-за свойства дивергенции Кульбака-Лейблера, а именно

$$D(x^1, x^2) \geq \frac{1}{2}\|x^1 - x^2\|_2^2$$

*только КЛ
это не это в диверг.*

Для множества Y сделаем схожую постановку, но немного модифицируем ее, так как аналогичная идея с энтропией здесь не подошла бы, из-за того что значения y могут быть отрицательными вплоть до -1 ю

$$\|y\|_Y = \sqrt{\sum_{i=1}^m \|y_i\|_1^2}$$

$$d_Y(y) = \sum_{i=1}^m (y_i + 1) \ln(y_i + 1)$$

$$B_Y(y^1, y^2) = \sum_{i=1}^m (y_i^1 + 1) \ln \frac{y_i^1 + 1}{y_i^2 + 1}$$

Нетрудно проверить, что такая d_Y будет также 1-строго выпуклая, что гарантирует нам также это свойство на всем $Z = (X, Y)$. Отметим также, что альтернативной вариацией структуры множества Y может являться обычная евклидовая структура, подобно тому, как это делалось в статье [6].

Для множества $Z = (X, Y)$ будем рассматривать в качестве прокси-функции и дивергенции Брегмана такую линейную комбинацию

$$d_Z(X, Y) = \frac{1}{R_X^2} d_X(x) + \frac{1}{R_Y^2} d_Y(y)$$

$$B_Z(x^1, y^1, y^2) = \frac{1}{R_X^2} B_X(x^1, x^2) + \frac{1}{R_Y^2} B_Y(y^1, y^2),$$

где $R_X^2 = \sup_X d_X(x) - \inf_X d_X(x)$, $R_Y^2 = \sup_Y d_Y(y) - \inf_Y d_Y(y)$.

Значение энтропии на симплексе размерности k имеет минимум в равномерном распределении со значением $-\sqrt{k}$ и равен $\ln(k)$, а supremum равен 0. Отсюда получаем, что $R_X^2 = m \ln n^2 + m \ln n = 3m \ln n$. В дальнейшем будем также различать $R_X = 2m \ln n$ и $R_p = m \ln n$. Для y можем искать минимум и максимум по каждой координате отдельно. Минимум функции $f(y) = y \ln(y)$ на $[0, 1]$ есть $2 \ln(2)$ и $-\frac{1}{e}$. В итоге, $R_y^2 = 2mn(2 \ln(2) + \frac{1}{e})$.

4 Обзор итогового алгоритма

4.1 Итерации алгоритма

Наш алгоритм во многом является приложением к уже упомянутой статье о применении редукции дисперсии для вариационных неравенств. Он представляет из себя алгоритм MirrorProx с применением SVRG, то есть вместо обычной двухшаговой итерации алгоритма используются их адаптация с семплированием несмещенной оценки соответствующего функционала для уменьшения дисперсии на текущем шаге алгоритма. Аналогично статье мы будем делать следующие итерации во внутреннем цикле

$$\hat{z} = \arg \min_z (\tau \langle F(w), z \rangle + \alpha B_Z(z, z^k) + (1 - \alpha) B_Z(z, \hat{w}))$$

sample index ξ according to our sampling method

$$z^{k+1} = \arg \min_z (\tau \langle F(w) + F_\xi(\hat{z}) - F_\xi(\hat{w}), z \rangle + \alpha B_Z(z, z^k) + (1 - \alpha) B_Z(z, \hat{w})),$$

здесь w, \hat{w} - обновляемые вспомогательные переменные того же размера, что и z , необходимые для корректной и оптимальной работы SVRG. Позже разберем их обновления уже в алгоритме.

Сначала разберем как функционалы от z выглядят в нашем случае, то есть в формулировке [6]. Для начале заметим, что мы можем каждый раз отдельно пересчитывать x, y, p , поэтому запишем каждую строчку отдельно для них. Обозначим $z = (x, y, p)$, $w = (u, v, s)$, $\hat{w} = (\hat{u}, \hat{v}, \hat{s})$

Сначала выведем для x . Здесь и далее под $\nabla_a F(x, y, p)$ подразумевается часть градиента по a , где производная не тождественный ноль (в данном случае отрезок градиента размера mn^2).

$$\begin{aligned} f(x) &= (\tau R_x^2 \nabla_x F(x, y, p)^T x + \alpha \text{KL}(x, x^k) + (1 - \alpha) \text{KL}(x, \hat{u})) = \\ &= \tau R_x^2 \nabla_x F(x, y, p)^T x + \sum_{i=1}^{mn^2} x_i \log \left(\frac{x_i}{(x_i^k)^\alpha \hat{u}_i^{1-\alpha}} \right) \end{aligned}$$

Вспоминая про условие, что x принадлежит симплексу, получаем Лагранжиан:

$$\mathcal{L}(x, \lambda) = \tau R_x^2 \nabla_x F(x, y, p)^T x + \sum_{i=1}^{mn^2} x_i \log \left(\frac{x_i}{(x_i^k)^\alpha \hat{u}_i^{1-\alpha}} \right) + \lambda \left(\sum_{i=1}^k x_i - 1 \right)$$

$$\frac{\partial \mathcal{L}}{\partial x_i} = \tau R_x^2 \nabla_x F(x, y, p) + \log \left(\frac{x_i}{(x_i^k)^{\alpha} \hat{u}_i^{1-\alpha}} \right) + 1 + \lambda = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \left(\sum_{i=1}^k x_i - 1 \right) = 0$$

Откуда получаем

$$x = (x^k)^{\alpha} \hat{u}^{1-\alpha} e^{-\tau R_x^2 \nabla_x F(x, y, p) - 1 - \lambda}$$

$$\left(\sum_{i=1}^k x_i - 1 \right) = 0 .$$

λ здесь по сути нормировочная константа. В итоге получаем формулу, которую проще всего записать через *softmax* ($\text{softmax}(x) = \frac{e^x}{\sum_{x_i \in x} e^{x_i}}$):

$$\hat{x} = -\tau R_x^2 \nabla_x F(x, y, p)^T x + \alpha \ln x^k + (1 - \alpha) \ln \hat{u}$$

$$x_{opt} = \text{softmax}(\hat{x})$$

Отметим, что название переменной \hat{x} здесь не случайно - эти переменные как раз отвечают за значение переменной в двойственном пространстве.

Аналогичным образом можно получить оптимальное значение для p .

$$f(p) = (\tau R_p^2 \nabla_p F(x, y, p)^T p + \alpha \text{KL}(p, p^k) + (1 - \alpha) \text{KL}(x, \hat{s})$$

$$\hat{p} = -\tau R_p^2 \nabla_p F(x, y, p)^T p + \alpha \ln p^k + (1 - \alpha) \ln \hat{s}$$

$$p_{opt} = \text{softmax}(\hat{p})$$

Для y получаем такой вывод без ограничений

$$f(y) = (\tau R_y^2 \nabla_y F(x, y, p)^T y + \alpha \text{KL}(y + 1, y^k + 1) + (1 - \alpha) \text{KL}(y + 1, \hat{v} + 1)) =$$

$$= \tau R_y^2 \nabla_y F(x, y, p)^T y + \sum_{i=1}^{2mn} y_i \log \left(\frac{y_i + 1}{(u_i^k + 1)^a (\hat{v} + 1)_i^{1-a}} \right)$$

$$\frac{\partial f}{\partial y_i} = -\tau R_y^2 \nabla_y F(x, y, p)^T y + \log \left(\frac{y_i + 1}{(y_i^k + 1)^a (\hat{v}_i + 1)^{1-a}} \right) + 1 = 0$$

$$y_{opt}^{temp} = (y^k + 1)^{\alpha} (\hat{v} + 1)^{1-\alpha} e^{\tau R_y^2 \nabla_y F(x, y, p) - 1} - 1$$

Вспомним, что $y \in [-1, 1]^{2nm}$, тогда в силу выпуклости оптимальной точкой будет

покоординатная проекция на это множество. Введем функцию $\text{proj}(y) = \max(\min(y, 1), -1)$, и применяя ее покоординатно, получаем:

$$y_{opt} = \text{proj}((y^k + 1)^\alpha (\hat{v} + 1)^{1-\alpha} e^{\tau R_y^2 \nabla_y F(x, y, p)} - 1)$$

Отметим также, что для евклидовой структуры на множестве Y мы бы имели шаг $y_{opt} = \text{proj}(\alpha y^k + (1 - \alpha)\hat{v} + \tau R_y^2 \nabla_y F(x, y, p))$. Для y в любой из постановок нам не потребуется использование какой-либо двойственной переменной, так как никакая двойственная задача здесь не решалась.

Осталось разобраться с операцией, которая идет после семплирования. Нетрудно заметить, что она имеет такой же вид, как и первая. Из-за этого для нее аналогичным образом можно вывести те же формулы, которые будут отличаться также только первым слагаемым - к нему добавится разность значений оценки градиента с помощью семплирования в соответствующих точках.

Теперь перейдем к способу семплирования, который применялся в данном алгоритме.

4.2 Семплирование *с генерацией*

Напомню, что эта операция необходима как раз для имплементации редукции дисперсии в алгоритм. Идея ровно та же, что и в каноничном SVRG, в конкретном случае мы хотим получить набор несмещенную оценку в конкретной точке нашей функции, которая представляет из себя конкатенацию градиентов со знаками, обладающей также некоторыми свойствами, необходимыми для улучшения сходимости. Здесь опять будем много ссылаться на статью [2], многие факты были подчеркнуты из нее и из ссылок в ней же.

Введем определение L-липшицева оценка функции F , потребуя от функции два условия. Первое условие - понятное необходимое требование, второе - оценка для мощности метода, именно оно отвечает за влияние техники редукции дисперсии.

$$E_\xi[F_\xi(x, y, p)] = F(x, y, p)$$

$$E_\xi[\|F_\xi(u) - F_\xi(v)\|_*^2] \leq L^2 \|u - v\|^2,$$

здесь $\|\cdot\|_*$ - двойственное расстояние.

Для примера возьмем наивным способом семплирования для некоторого вектора V является случайная проекция на одну из его координат, увенчанная в $\dim(V)$ раз. Понятно,

что такая оценка будет несмешенной, но константа L у нее может быть достаточно большой, как минимум из-за искусственного увеличения каждого семпла в $\dim(V)$ раз, что сразу увеличит константу в $\sqrt{\dim(V)}$ раз. Из плюсов такого способа - у нас фиксированное распределения на любом шаге алгоритма. Этого свойства мы вынуждены будем лишиться и будем выбирать каждый раз новое распределение, исходя из переменных, насчитанных к данной итерации, зато сильно уменьшим константу L .

Заметим, что оптимизировать все m значений x_i, y_i, p мы можем независимо друг от друга, так как в F все индексы на самом деле встречаются независимо друг от друга. Поэтому достаточно построить оценки для функций $\frac{1}{m}(d + 2\|d\|_\infty A^T y_i), \frac{1}{m}2\|d\|_\infty(b - Ax_i - Mp), 0_n$.

Вспомним, что в самом алгоритме от нас на самом деле требуется разность функций в конкретных двух точках, поэтому будем использовать эти точки для построения новых распределений и обновлять распределения по мере обновления значений этих точек. Начнем с x_i , градиент которого $\frac{1}{m}(d + 2\|d\|_\infty A^T y)$ и предположим у нас есть две точки u, v из того же пространства. Тогда будем семплировать точки следующим образом:

$$F_\xi^{X_k}(x_k) = \frac{1}{m}(d + 2\|d\|_\infty \frac{1}{r_i} A_{i,:}(y_k)_i), P(\xi = i) = r_i$$

$$r_i = \frac{|u_i - v_i|_1}{\sum_{j=1}^{n^2} |u_j - v_j|_1}$$

Нетрудно проверить эквивалентность несмешенности этой оценки и оценки $\frac{1}{r_i} A_{i,:}(y_k)_i$ для функции $A^T y_k$, более того благодаря линейности, эта оценка будет иметь константу такую же константу L с точностью до домножения на $\frac{1}{m}2\|d\|_\infty$. В статье эта оценка использовалась для схожей задачи и нам потребуется факт о значении константы липшицевости, которая в данном случае будет $\frac{1}{m}2\|d\|_\infty \|A\|_{max} = \frac{1}{m}2\|d\|_\infty$. Коротко опишу выкладки из доказательства этого факта:

$$\begin{aligned} E_{i \sim r}[(\frac{1}{r_i} A_{i,:} u_i - \frac{1}{r_i} A_{i,:} v_i)^2] &= \{(\|\cdot\|_1)_* = \|\cdot\|_\infty\} = E_{i \sim r}[\frac{1}{r_i^2} \|A_{i,:}(u_i - v_i)\|_\infty^2] = \\ &= E_{i \sim r}[\frac{1}{r_i^2} \|A_{i,:}\|_{max}^2 |u_i - v_i|^2] \leq \sum_{i=1}^{n^2} \|A\|_{max}^2 |u_i - v_i| \|u - v\|_1 = \|A\|_{max}^2 \|u - v\|_1^2 \end{aligned}$$

Метод семплирования для $\nabla_y F(x, y, p)$ будет похожим - отличие будет только в том, что здесь будет сумма двух умножений на матрицу: одно для x , другое для p

$$F_\xi^{Y_k}(y_k) = \frac{1}{m}2\|d\|_\infty(b - \frac{1}{r_i s_k} (A_{:,j}(x_k)_j - M_{:k}p)), P(\xi = j, k) = c_j s_k$$

$$c_j = \frac{|u_j - v_j|}{\sum_{j=1}^{j=2n} |u_j^y - v_j^y|_1}, s_k = \frac{|u_k^p - v_k^p|^2}{\sum_{k=1}^{k=n} |u_k^p - v_k^p|_1}$$

Благодаря тому, что для x_i и для p мы в неравенстве мы получим два независимых слагаемых, из-за выбора нормы, то и константа L в данном случае будет $\max(\|A\|_{max}, \|M\|_{max}) \frac{1}{m} 2\|d\|_\infty = \frac{1}{m} 2\|d\|_\infty$.

В евклидовом сетапе для Y был бы аналогичный метод, только со второй нормой вместо первой и квадратами модулей. Теория показывает, что в этом случае L увеличилась бы в $\|A\|_{frob}$ раз по сравнению с Y с первой нормой.

Семплирование для $\nabla_p F(x, y, p)$ не нужно, так как F'_p не зависит от p , то есть любая оценка будет несмещенная с константой $L = 0$.

Как уже писалось раньше, все эти выкладки дают нам вывод о том, что при семплировании каждого x_i, y_i и p , мы получаем оценку $F_\xi(z)$ с константной липшицевости $\frac{1}{m} 2\|d\|_\infty$.

Наконец, в алгоритме эту часть можно реализовать сильно проще. Вместо того, чтобы пересчитывать каждую оценку, можно сразу перейти к следующему шагу, где оптимизируется функционал, использующий нашу оценку. Подробно этот шаг расписан в Алгоритмах 1 и 2, после чего применяется в Алгоритме 3.

Заметим, что алгоритмы корректные, исходя из объяснений выше. Все независимые от переменной значения в разности пропадут, поэтому в Алгоритмах нет слагаемых, содержащих вектора b и d из условия. Также для семплирования $\nabla_y F(x, y, p)$ матрица M оказывается равной $(I, 0)^T$, из-за чего запись для этого слагаемого упрощается.

Algorithm 1: Sampling step for X

Data:

q_i, \dots, q_m , incidence matrix A , two set of vectors (v_1, \dots, v_m) and $(v_1^{temp}, \dots, v_m^{temp})$

```

1 Oracle_u = (0)_{m × n^2}
2 for i ← 1 to m do
3     l1 = abs(v^{temp}[i] - v[i])
4     N = sum(l1)
5     Sample ind from [0, ..., n^2 - 1] with p = l1/N
6     Oracle_u[i] = A[ind] * N * sign(v^{temp}[i] - v[i])[ind]
7 end

```

Result: Oracle_u

Algorithm 2: Sampling step for Y

Data: q_i, \dots, q_m , incidence matrix A ,

two set of vectors (u_1, \dots, u_m) and $(u_1^{temp}, \dots, u_m^{temp})$, two vectors s and s^{temp}

```
1 Oraclev = (0)m × 2n
2 for i ← 1 to m do
3     lp, ly = abs(stemp - s), abs(vtemp - v)
4     Np, Ny = sum(l1), sum(l2)
5     Sample ind1 from [0, ..., n] with p = lp/Np
6     Sample ind2 from [0, ..., 2n] with p = ly/Ny
7     Oracleu[i] = A[:, ind2] * N * (vtemp - v)[ind2]/l2[ind2]
8     Oracleu[i][:n][ind1] += (stemp - s)[ind1]
9 end
```

Result: x,y,p

4.3 Реализация

В обычной реализации алгоритма с SVRG используют переменную w , чаще являющуюся усредненным значением ранее посчитанных значений переменной. Здесь же удобно ввести также \hat{w} - переменную, которая будет усреднением переменной в двойственном пространстве (в нашем случае это значение переменных до применения softmax). Необходимые гиперпараметры, такие как α, τ, K будут объяснены в разделе про скорость сходимости.

В результате моей работы получился Алгоритм 3. Опишем его идею здесь кратко словами, оставив детали для главы с анализом. На каждой итерации цикла происходит несколько операций (строки 8-14), подобных операциям MirrorProx. Отличие заключается в том, что дивергенцию в оптимизируемом в каждой операции функционале мы разделяем на линейную комбинацию дивергенций от прямой и двойственной переменных. Стока 11 записана для евкликовой структуры множества Y , для неевклидовой будем пользоваться другой формулой, выведенной выше. Семплирование для SVRG описано выше. Вторая итерация в алгоритме происходит неявно - как писалось выше, решение соответствующей оптимизационной задачи есть решение задачи из первой итерации, сдвинутой на соответствующее значение разности градиентов. Это и описано в строках 16-19. Пересчет переменных, являющийся по сути усреднением уже посчитанных значений, происходит в строках 20 – 24.

Реализация данного алгоритма была сделана с помощью python, с использованием библиотек для оптимальных матричных вычислений и работы с вероятностными пространствами, таких как numpy и scipy. По сравнению с описанным ниже Алгоритмом 3, его итого-

вся реализация получилась более корректной: использует векторные операции вместо циклов и матричную арифметику для разреженных матриц.

Algorithm 3: Mirror Prox with Variance Reduction

(MPVR) \leftarrow новая методика так съёмка

а на Mirror Prox как же MP.

Data: q_i, \dots, q_m , incidence matrix A , vectorized cost matrix d , $Rx = \sqrt{2m \ln(n)}$, $Rp = \sqrt{m \ln(n)}$, $Ry = \sqrt{mn}$, $\tau, K = 2, \alpha = \frac{1}{2}$

- 1 $x = u = u^{temp} = u^{next} = \mathbf{1}_{n^2}/(n^2)$
- 2 $y = v = v^{temp} = v^{next} = \mathbf{0}_{2n}$
- 3 $p = s^{temp} = s^{next} = \mathbf{1}_n/n$
- 4 $\hat{x} = \hat{u} = u^{\hat{temp}} = u^{\hat{next}} = \ln(x)$
- 5 $\hat{p} = s^{\hat{next}} = \ln(p)$
- 6 **while** $error < tol$ **do**
- 7 **for** $k \leftarrow 1$ **to** K **do**
- 8 **for** $i \leftarrow 1$ **to** m **do**
- 9 $u^{\hat{temp}}_i = -\tau Rx^2 \nabla_u F(u, v, s) + \alpha \hat{x}_i + (1 - \alpha) \hat{u}_i$
- 10 $u_i^{temp} = softmax(u_i^{\hat{temp}})$
- 11 $v_i^{temp} = proj(\tau Ry^2 (\nabla_v F(u, v, s))_i + \alpha y_i + (1 - \alpha) v_i)$
- 12 **end**
- 13 $\hat{p} = -\tau m \ln(n) \nabla_s F(u, v, s) + \alpha \hat{p} + (1 - \alpha) \hat{s}$
- 14 $s^{temp} = softmax(\hat{p})$
- 15 Calculate ($Oracle_u, Oracle_v$) according to algorithm 1 and 2
- 16 $\hat{x} = u^{\hat{temp}} - \tau 2m \ln(n) Oracle_u$
- 17 $x = softmax(\hat{x})$
- 18 $y = v^{\hat{temp}} - \tau mn Oracle_v$
- 19 $s = s^{temp}$
- 20 $u^{next} = (x + (k - 1) * u^{next})/k$
- 21 $v^{next} = (y + (k - 1) * v^{next})/k$
- 22 $s^{next} = (p + (k - 1) * s^{next})/k$
- 23 $u^{\hat{next}} = (\hat{x} + (k - 1) * u^{\hat{next}})/k$
- 24 $s^{\hat{next}} = (\hat{p} + (k - 1) * s^{\hat{next}})/k$
- 25 **end**
- 26 $u, v, s = u^{next}, v^{next}, s^{next}$
- 27 $\hat{u}, \hat{s} = u^{\hat{next}}, s^{\hat{next}}$
- 28 **end**

Result: x, y, p

4.4 Обоснование и скорость сходимости

В этой главе мы покажем корректность данного алгоритма, а также предложим доказательство полученной скорости сходимости. Во многом анализ будет ссылаться на уже несколько раз упомянутую статью [2].

Сначала нам потребуется мера сходимости, которую мы определим, как

$$Gap(w) = \max_{z \in Z} \langle \nabla F(z), w - z \rangle$$

где $F(z)$ мы определили в секции 3.2. Понятно, что сходимость алгоритма эквивалента сходимости этой функции к нулю, ведь в формулировке задачи как вариационного неравенства мы хотим найти такую w , что при любых значениях z функционал внутри определения $Gap(w)$ будет больше нуля. Отметим также, что работая с вероятностями мы будем оценивать $E[Gap(w)]$.

В качестве итогового решения мы представляем значение

$$z_{final} = \frac{1}{mn} \sum_{i,j=1,1}^{i,j=K,S} [u_{ij}^{temp}, v_{ij}^{temp}, s_{ij}^{temp}]$$

где K, S - количество итераций во внешнем и внутреннем цикле соответственно, а переменные соответствует аналогичным в Алгоритме 3.

Следуя результату полученным в статье имеем следующее неравенство

$$E\left[\sum_{i,j=1,1}^{i,j=K,S} \max_{z \in Z} \langle \hat{F}_{ij} - F_{ij}, z \rangle\right] \leq \max_{z \in Z} D_z(z, z_0) + \sum_{i,j=1,1}^{i,j=K,S} 4\tau^2 L^2 E[\|z_{ij}^{temp} - w_i\|_2^2],$$

где L - константа липшицевости из оценки при семплировании, (напомним, что она получилась равной $\frac{1}{m}2\|d\|_\infty$), w_i - переменная соответствующая $[u, v, s]$, которые заданы в Алгоритме 3, $\hat{F}(w) = F(w) + F_\xi(\hat{z}) - F_\xi(\hat{w})$, то есть значение оценки градиента с применением показанного семплирования.

Также из теории мы имеем:

$$E[Gap(z_{final})] \leq \frac{1}{KS\tau} (E\left[\sum_{i,j=1,1}^{i,j=K,S} \max_{z \in Z} \langle \hat{F}_{ij} - F_{ij}, z \rangle\right] + \max_{z \in Z} (\alpha + K(1-\alpha)) D(z, z_0))$$

Наконец, мы можем оценить слагаемое, в котором мы используем вторые нормы $z_{ij}^{temp} - w_i$. Здесь также сошлемся на уже готовый результат.

Не забывайте быть отступа, проверяйте всегда

$$\sum_{i,j=1,1}^{i,j=K,S} E[\|z_{ij}^{temp} - w_i\|_2^2] \leq \max_{z \in Z} \frac{2(\alpha + K(1-\alpha))}{(1-\alpha)(1 - \frac{\tau^2 L^2}{1-\alpha})} D(z, z_0)$$

Исходя из всех имеющихся неравенств, можем вывести оценку

$$E[Gap(z_{final})] \leq \frac{1}{KS\tau} (\max_{z \in Z} D_z(z, z_0) + \max_{z \in Z} (\alpha + K(1-\alpha)) D(z, z_0)) + \\ + \max_{z \in Z} \frac{8\tau^2 L^2 (\alpha + K(1-\alpha))}{(1-\alpha)(1 - \frac{\tau^2 L^2}{1-\alpha})} D(z, z_0)) = \leq \frac{1}{KS\tau} (1 + \beta + \frac{8\gamma\beta}{1-\gamma}) \max_{z \in Z} D_z(z, z_0),$$

где $\beta = \alpha + K(1-\alpha)$, $\gamma = \frac{L^2\tau^2}{1-\alpha}$

Теперь выберем правильные гиперпараметры, а именно $\alpha = 1 - \frac{1}{K}$, $\tau = \frac{\sqrt{1-\alpha}}{\sqrt{17}L}$.

$$E[Gap(z_{final})] \leq \frac{1}{KS\tau} (2 + \frac{\frac{16(1-\alpha)}{17}}{1 - \frac{1-\alpha}{17}}) \max_{z \in Z} D(z, z_0) = \frac{3}{KS\tau} \max_{z \in Z} D(z, z_0) = O(\frac{L \max_{z \in Z} D(z, z_0)}{\sqrt{KS}}).$$

Теперь сошлемся на [6], где как раз рассматривается значение $L \max_{z \in Z} D(z, z_0)$. Исходя из выбора D по аналогии с литературой можем вывести оценку для данного выражения, соответствующую $LR_xR_y = O(\|C\|_\infty \sqrt{n})$. В итоге получаем скорость сходимости:

$$E[Gap(z_{final})] = O(\frac{\|C\|_\infty \sqrt{n}}{\sqrt{KS}})$$

Тогда для того, чтобы достичь точности в ε необходимо сделать количество итераций, равное $S = O(\frac{\|C\|_\infty \sqrt{n}}{\sqrt{K\varepsilon}})$.

Теперь поймем цену каждой итерации. Во-первых, мы должны посчитать градиент, что делается за $C_1 = 2mn^2$ операций, так как мы m раз должны произвести матрично-векторное умножение с разреженной матрицей, где $2n^2$ ненулевых элементов. Во-вторых, мы на каждом шаге мы оцениванием градиент m раз из матрицы по столбцам и по строкам, что дает еще $C_2 = Km(n^2 + 2n)$ операций. В итоге получаем

$$Complexity = O(\frac{(2mn^2 + Km(n^2 + 2n))\sqrt{n}\|C\|_\infty}{\sqrt{K\varepsilon}}) = O(\frac{mn^2\sqrt{n}\|C\|_\infty}{\varepsilon})$$

Для получения финальной асимптотике мы лишь взяли K , равный константе.

При евклидовой структуре множества Y аналогичный вывод дал бы асимптотику в $\|A\|_{Frob} = O(n)$ раз больше.

4.5 Распределенный алгоритм

В этой секции хотелось бы кратко отметить, что алгоритм можно сделать распределенным. Действительно, как минимум внутри каждой итерации мы независимо пересчитываем переменные для каждого x_i, y_i и p - эти вычисления можно распределить на несколько машин, что кратно ускорит вычисления.

Для экспериментов же мы решили не переписывать алгоритм в распределенный вид *один* для сравнения с такими же нераспределенными аналогами.

5 Эксперименты

проведенных

В этой секции будет рассказано о проведенных нами экспериментах с новым алгоритмом. С кодом вы можете познакомиться по данной [ссылке](#)

5.1 Смесь гауссиан

Первый эксперимент основан на задаче, где необходимо найти барицентр Васерштейна для смеси гауссиан. На вход алгоритму подается матрица с n случайно сгенерированными точками для каждого из m нормальных распределений и требуется на выходе получить точки из распределения, являющегося барицентром Васерштейна. Например, пример входных данных можно увидеть на рисунке 5.1

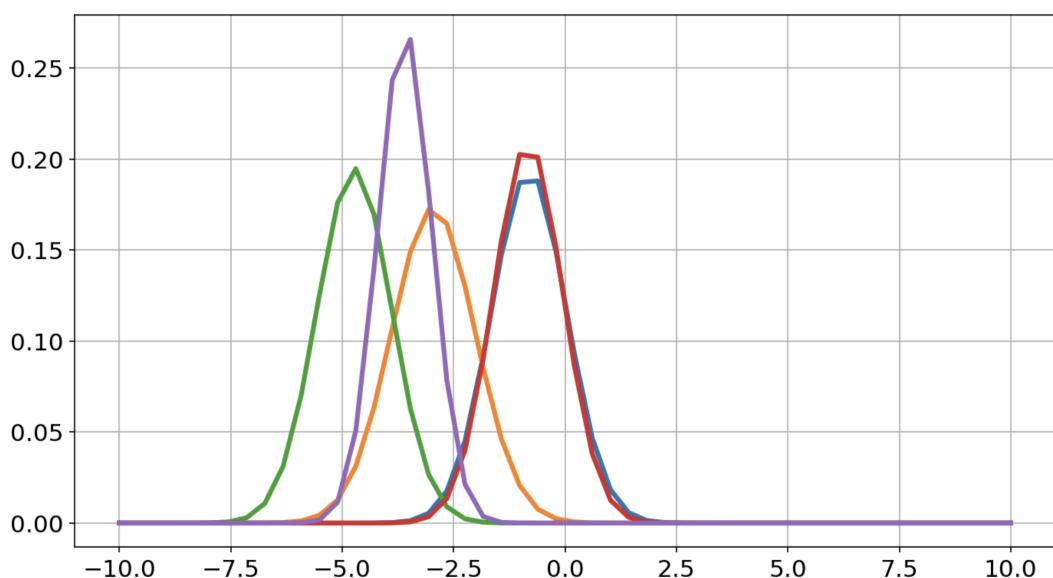


Рис. 5.1: Визуализация входных данных для задачи поиска барицентра Васерштейна на смеси гауссиан при $k=5$

Важной особенностью задачи про смесь гауссиан является то, что у задачи есть ана-

литическое решение для стандартного задания матрицы C : $C_{ij} = |i - j|^2 / (n - 1)^2$. В данном случае оптимальным решением является гауссиана с матожиданием равным среднему матожиданию всех m данных гауссиан и стандартным отклонением равным среднему стандартному отклонению всех m данных гауссиан.

Цель данного эксперимента: ~~была~~ проанализировать результат реализации алгоритма, то есть убедиться, что полученное в результате распределение действительно сходится к истинной гауссианы. Также мы сравнили результат работы алгоритма с алгоритмами приведенными в статье [2], чтобы визуально сравниться также и с ними.

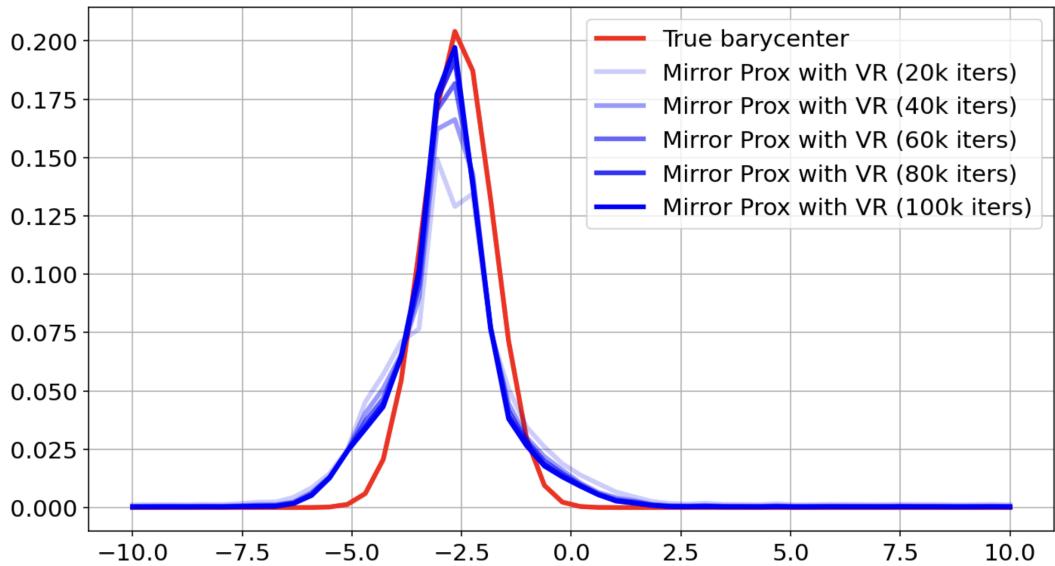


Рис. 5.2: Результат работы нашего алгоритма при разном количестве итераций

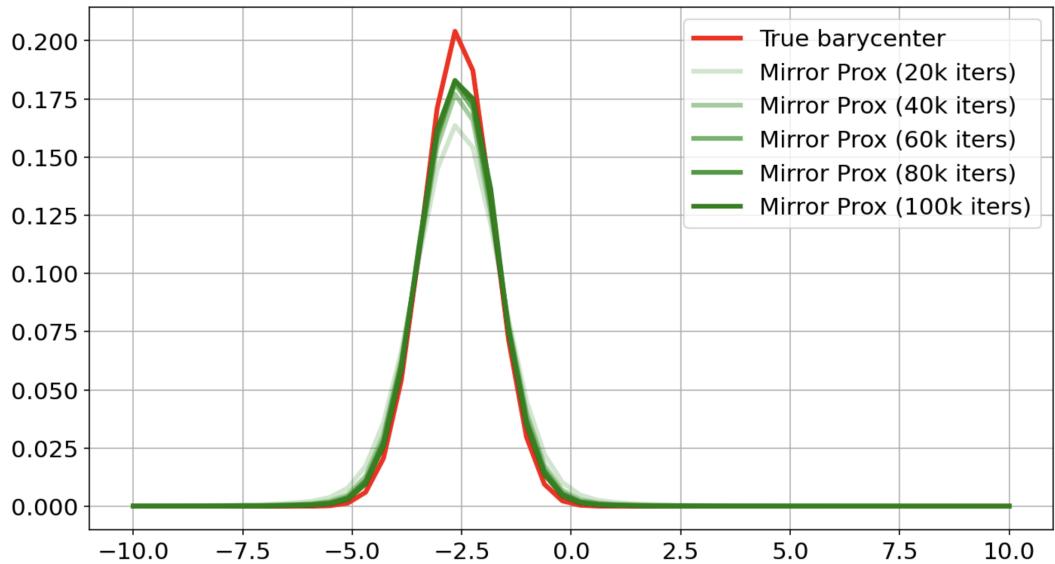


Рис. 5.3: Результат работы алгоритма из статьи [2] при разном количестве итераций

Как можно видеть на рисунке 5.2 алгоритм действительно сходится к настоящему барицентру. В то же время сравнивая результат с уже существующим и на данный момент

оптимальным алгоритмом (см рисунок 5.3) можно отметить, что сходится наш алгоритм медленнее.

Мы также проверили, что будет при увеличении количества итераций внутреннего цикла, отвечающего за сэмплирование. Напомним, что из анализа получалось, что достаточно брать константу, чтобы удовлетворять асимптотике.

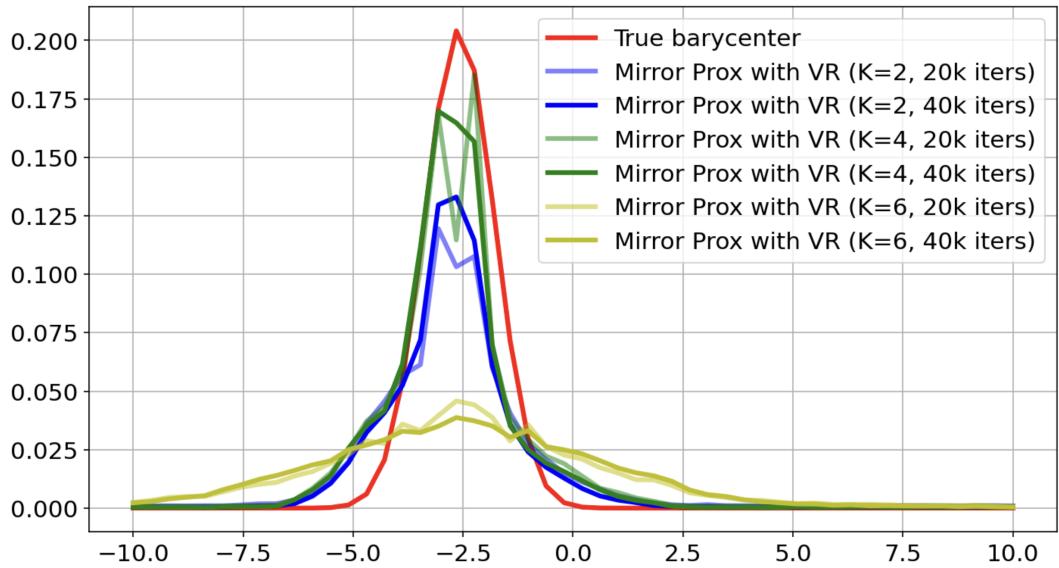


Рис. 5.4: Результат работы нашего алгоритма для разных K

какого? Ссылка

*Алг 10,
научные
справки*

На рисунке 5.4 можно видеть результат работы алгоритма для алгоритма для $K \in [2, 4, 6]$. Нетрудно заметить, что наилучший результат показал $K = 4$, в то же время для $K = 6$, несмотря на большее количество итераций и, следовательно, более долгое время работы, алгоритм сходится медленнее. Это может быть понятно из асимптотики, поскольку в нее добавится лишняя константа, то есть для достижения той же точности потребуется и увеличение количества итераций во внешнем цикле.

Также посмотрим на скорость сходимости алгоритма. Будем считать ошибкой функциональное оптимальное расстояние, которое определяется по формуле

$$gap(p) = \frac{1}{m} \sum_{i=1}^{i=m} W(p, q_i) - \sum_{i=1}^{i=m} W(p_{true}, q_i),$$

где W - расстояние в терминах оптимального транспорта, считающееся в python с помощью функции из специальной библиотеки *ot*. p_{true} - истинное значение барицентра, а p - значение полученное с помощью нашего алгоритма.

К сожалению, представленную выше хорошую скорость сходимости не удалось показать на практике. На рисунке 5.5 можно видеть, что упомянутая выше вариация MirrorProx без использования техники редукции дисперсии сходится быстрее.

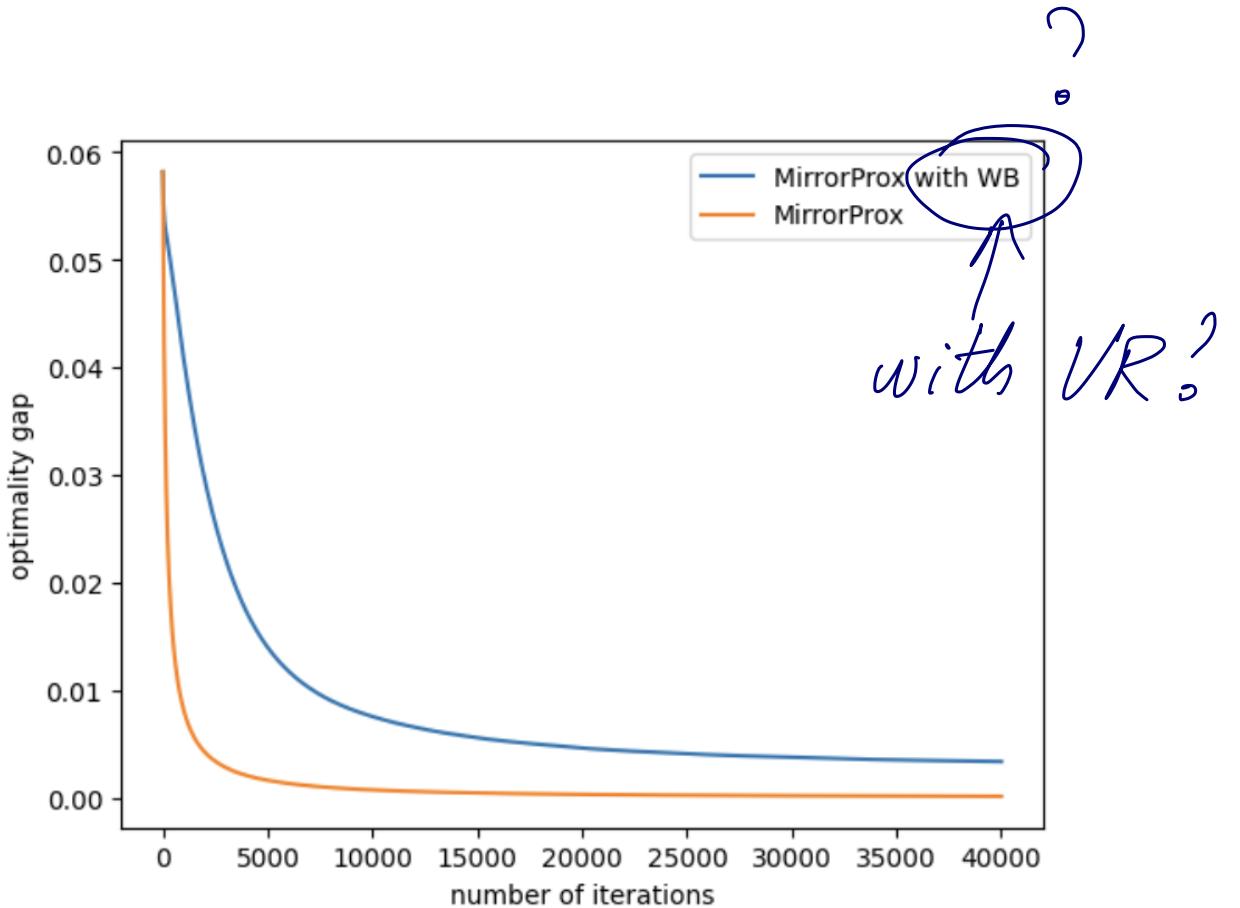


Рис. 5.5: График зависимости $gap(p)$ от количества итераций

5.2 MNIST

Вторая задача, на которой мы проверяли алгоритм - приближение изображений из набора данных *MNIST*. Постановка задача такая - берем к изображений из набора с иллюстрацией одной и той же цифры, каждое изображение растягивается в вектор и интерпретируется как некоторое распределение. Барицентр этих распределений в итоге должен выдавать усредненное в терминах оптимального транспорта изображение с очертанием той же цифры.

Ниже приведены примеры полученных изображений (рисунок 5.6). Можно видеть, что хоть и отдаленно, но эти изображения соответствуют изначальным цифрам. Совсем не очень получилось только для цифры 4. Частично это можно оправдать тем, что задача на самом деле громоздкая, так как размерность n здесь порядка сотен, а как мы знаем в асимптотике встречается целый n^2 . Из-за этого количество итераций пришлось понизить до 1000.

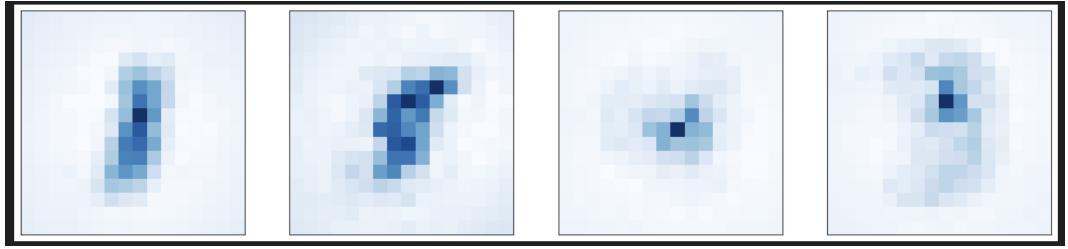


Рис. 5.6: Иллюстрация для найденных барицентров, полученных из распределений, соответствующим набору изображений из набора данных MNSIT для цифр 1, 5, 4, 3

6 Результаты

В этой секции подведем некоторый итог проделанной работы. Основным результатом работы является полученный нами алгоритм, представленный в секции 4.3. Как и было запланировано, в нем используется техника редукции дисперсии, а именно вариация метода *SVRG* со специально выбранным методом семплирования. Сам алгоритм, являющийся вариацией метода *MirrorProx*, был успешно адаптирован для задачи о барицентрах Васерштейна и подробно описан в тексте работы, также может быть рассмотрен с точки зрения распределенных методов оптимизации.

Задача и подходы к ее решению были широко и подробно рассмотрены с теоретической стороны. Была рассмотрена как сама задача с различных сторон и необходимых для нас переформулировок, так и был проведен анализ полученного алгоритма. В результате данного анализа была также получена скорость сходимости предложенного метода, которая не уступает адаптации *MirrorProx* без техник редукции дисперсии, в то же время опережая другие методы, предлагавшиеся для задачи ранее. В частности, для обеспечения необходимой скорости сходимости были подобраны оптимальные для конкретной постановки задачи гиперпараметры.

Также были проведены несколько экспериментов на нескольких наборах данных. Была показана корректность алгоритма на примере смеси гауссиан и набора данных MNIST. Дополнительно был проведен анализ некоторых параметров алгоритма. На экспериментах мы видим, что алгоритм может успешно применяться, в то же время виден потенциал для улучшений результатов его работы.

Дальнейшая работа, являющаяся логическим продолжением текущей, может включать в себя внедрение редукции дисперсии в новые вариации уже изученных методов оптимизации, таких как Dual Extrapolation или extragradient, для применения на задаче о барицентрах Васерштейна. Сами методы редукции дисперсии потенциально могут быть улучшены для конкретной задачи, например, при выборе оптимального метода семплирования.

Список литературы

- [1] Martial Aguech и Guillaume Carlier. “Barycenters in the Wasserstein Space”. B: *SIAM Journal on Mathematical Analysis* (January 2011). URL: https://www.researchgate.net/publication/220132573_Barycenters_in_the_Wasserstein_Space
- [2] Ahmet Alacaoglu и Yura Malitsky. “Stochastic Variance Reduction for Variational Inequality Methods”. B: (June, 2022). URL: <https://arxiv.org/pdf/2102.08352.pdf>
- [3] P. Balamurugan и Francis Bach. “Stochastic Variance Reduction Methods for Saddle-Point Problems”. B: (November, 2016). URL: <https://arxiv.org/pdf/1605.06398.pdf>
- [4] Sébastien Bubeck. “Theory of Convex Optimization for Machine Learning”. B: (May, 2014). URL: <https://arxiv.org/pdf/1405.4980v1.pdf>
- [5] Xiao Ding и Deren Han. “A modification of the forward-backward splitting method for maximal monotone mappings”. B: (2013). URL: <https://www.aimscolloquium.org/article/doi/10.3934/naco.2013.3.295>
- [6] Darina Dvinskikh и Daniil Tiapkin. “Improved Complexity Bounds in Wasserstein Barycenter Problem”. B: (October, 2020). URL: <https://proceedings.mlr.press/v130/dvinskikh21a/dvinskikh21a.pdf>
- [7] Robert M. Gower, Mark Schmidt, Francis Bach и Peter Richter. “Variance-Reduced Methods for Machine Learning”. B: (October, 2020). URL: <https://arxiv.org/pdf/2010.00892.pdf>
- [8] Sergey Guminov, Pavel Dvurechensky и Alexander Gasnikov. “Accelerated Alternating Minimization”. B: (June, 2019). URL: <https://arxiv.org/pdf/1906.03622v1.pdf>
- [9] Nhat Ho, Viet Huynh, Dinh Phung и Michael I. Jordan. “Probabilistic Multilevel Clustering via Composite Transportation Distance”. B: (October 30, 2018). URL: <https://arxiv.org/pdf/1810.11911.pdf>
- [10] Arun Jambulapati, Aaron Sidford и Kevin Tian. “A Direct $O(1/\epsilon)$ Iteration Parallel Algorithm for Optimal Transport”. B: (June, 2019). URL: <https://arxiv.org/pdf/1906.00618.pdf>
- [11] Anatoli Juditsky, Arkadi Nemirovski и Claire Tauvel. “Solving variational inequalities with Stochastic Mirror-Prox”. B: (October, 2018). URL: <https://cs.uwaterloo.ca/~y328yu/classics/extragrad.pdf>
- [12] G. Korpelevich. “The extragradient method for finding saddle points and other problems”. B: (1976). URL: <https://cs.uwaterloo.ca/~y328yu/classics/extragrad.pdf>

- [13] Dmitry Kovalev, Aleksandr Beznosikov, Abdurakhmon Sadiev, Michael Persiianov, Peter Richta и Alexander Gasnikov. “Algorithms for Decentralized Stochastic Variational Inequalities”. B: *NeurIPS 2022* (October, 2018). URL: https://papers.nips.cc/paper_files/paper/2022/file/c959bb2cb164d37569a17fa67494d69a-Paper-Conference.pdf.
- [14] Tianyi Lin, Nhat Ho, Xi Chen, Marco Cuturi и Michael I. Jordan. “Fixed-Support Wasserstein Barycenters: Computational Hardness and Fast Algorithm”. B: (June 2022). URL: <https://arxiv.org/pdf/2002.04783.pdf>.
- [15] Jason M. Altschuler и Enric Boix-Adser‘a. “Wasserstein barycenters are NP-hard to compute”. B: (December, 2020). URL: <https://arxiv.org/pdf/2101.01100>.
- [16] Yu. Nesterov. “Dual extrapolation and its applications for solving variational inequalities and related problems”. B: (August, 2003). URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4de9a7269ab4125f0f2240a136ef3c34c5527a73>.
- [17] Sidak Pal Singh, Andreas Hug, Aymeric Dieuleveut и Martin Jaggi. “Context Mover’s Distance Barycenters: Optimal Transport of Contexts for Building Representations”. B: (февр. 2020). URL: <https://arxiv.org/pdf/1808.09663>