Отчет по применению алгоритма KNN над данными mnist784

Федоров Артем Максимович
3 курс

Октябрь 2023

1 Введение

Одними из классических алгоритмов, решающих задачи multiclass classification по отнесению объектов к одному из заранее определенный классов на основе их признаков, являются метрические алгоритмы, что строят свои оценки на основании расстояний между объектами, как это следует из названия. В таких задачах объекты рассматриваются в формате векторов — элементов своего эмбеддингового пространства — где каждый координата вектора ассоциируется с соответствующей характеристикой объекта. После чего в таком пространстве задается метрика, по которой в дальнейшем и строятся ответы модели.

Одним из наиболее используемых среди прочего алгоритмов данного класса является алгоритм классификации по ближайшим соседям KNN - "**K** - **N**earest **N**eighbours". Для известного множеству классов $Y = y_1, y_2, ..., y_q$ и по набору объектов $\mathcal{D}$ для которых известны классы KNN строит оценки классов для объектов на основе ближайших по метрике в пространстве эмбеддингов известных прецедентов.

> Одним из преимуществ моделей на базе KNN - непараметричность. Нам не требуется знать, из какого вероятностного распределения к нам пришли данные объекты. В то время как более сложные статистические модели, базирующиеся на вероятностных распределениях полученных объектов, имеют хорошую математическую обоснованность, их недостатком является - низкая объясняющая способность, интерпретируемость. Они способны с большой точностью отнести объект к классу, но не могут понятным образом объяснить почему. В свою же очередь KNN такого недостатка лишен

Данный отчет стремится оценить качество работы алгоритма KNN, различные способы его модернизации и интерпретируемость.

2 Сопутствующая работа

Пусть классифицируемые объекты нашей задачи могут быть описаны пространством $\mathcal{X} = \mathbb{R}^d$, где d — размерность векторов признаков объектов $x \in \mathcal{X} = \{x_{1i}, x_{2i}, ..., x_{di}\}^T$. Тогда для заранее известного множества классов $\mathcal{Y} = \{y_1, y_2, ..., y_q\}$ составим задачу классификации объектов из $\mathcal{X}$ объектами из $\mathcal{Y}$. В таком случае введем на множестве $[\mathcal{X} \times \mathcal{Y}]$ операцию отношения $(\cdot \sim \cdot)$ по следующему принципу: $x_i \; y_j \Longleftrightarrow x_i$ относится к y_i классу. Тогда поставим задачу метрического классификатора, основанного на алгоритме KNN, на основе заданного множества объектов обучающей выборки, для которых известны классы $\mathcal{D} = \{(x_i, y_j) | x_i \sim y_i\}; x_i \in \mathcal{X}, y_j \in \mathcal{Y}$ размерности n, определять для входных объектов из того же пространства $\mathcal{X}$ классы, к которым они относятся. Тогда критерием постановки в соответствие входному объекту x метки y_j будем считать решение следующей задачи дискретной оптимизации: $\hat{x} \sim y_j \Rightarrow j = arg \max\limits_{j \in \{1,...,q\}} \sum_{i=k}^{k} w_i [x_{m_i} \sim y_j]$, где $x_{m_i} \in \mathcal{D}$ получен из последовательности x_m отранжированной по возрастанию расстояния до исследуемого объекта, а w_i вес каждого i соседа от 1 до k

> В частности, мы можем указать каждый из весов w_i за единицу, тем самым констатировав, что каждый объект вносит одинаковый вклад в классификацию

Рассмотрим задачу на примере датасета (бенчмарка) "mnist784", состоящего из 70000 объектов — картинок цифр 28 на 28 пикселей, где значения кодируют степень закрашенности от белого до черного 1. Для такой задачи характерен выбор пространства $\mathcal{X} = \mathbb{R}^{(28 \times 28)} = \mathbb{R}^{784}$ и выбор $\mathcal{Y} = \{0, 1, 2, ..., 9\}$.

Figure 1: Пример отображения элементов датасета

Определим особенности классификатора:

2.1 Выбор метрики

Если выбор $\mathcal{Y}$ не вызывает сомнений, то для $\mathcal{X}$ остается вольность в интерпретируемости метрики над векторами. Логичными кажутся две стандартные метрики: Евклидова метрика и косинусная метрика — выражаемые следующими выражениями:

$$d_{euclid}(x, y) = \sqrt{\sum_{k=1}^{784} (y_k - x_k)^2} = \sqrt{\sum_{k=1}^{784} y_k^2 + \sum_{k=1}^{784} x_k^2 - 2y^T x}$$
$$d_{cosine}(x, y) = 1 - \frac{y^T x}{\sqrt{\sum_{k=1}^{784} y_k^2} \sqrt{\sum_{k=1}^{784} x_k^2}}$$

Роль метрики в конечной задаче выполняет одну из решающих ролей, так как она определяет характер отношения "схожести" объектов. Если Евклидова метрика считает расстояние между объектами напрямую, как длина наименьшей кривой, соединяющей две точки, то косинусная метрика способна определить лишь отклонение одного вектора от другого, не указывая насколько именно далеко с точки зрения длины кривой, их соединяющей, они лежат. Тем самым одно из начальных предположений можно выдвинуть уже сейчас, что косинусная мера может быть более устойчивой при сравнении изображений одного класса, отличающихся мелкими деталями

2.2 Выбор весов

Отдельный вопрос состоит в том, чтобы определить, какую модель классификатора использовать. С точки зрения вероятностного подхода, метод KNN может выражаться в нахождении такого равномерно-наиболее мощного критерия для отвержения гипотезы, что x_i принадлежит классу y_j только для случая $w_i = 1, \forall i \in \{1, ..., k\}$. Однако очевидно, что при модернизации алгоритма тем, что мы будем учитывать вклад близких соседей сильнее чем дальних, может сильно улучшить стабильность работы алгоритма. Будем считать, что вес можно определить как $\frac{1}{d(x,y)+10^{-5}}$

2.3 Выбор стратегии поиска

Для поиска ближайших соседей среди обучающей выборки может быть использован любой алгоритм, способный ранжировать объекты выборки по удаленности от исследуемого по метрике в $\mathcal{X}$. Потому одной из основных идей реализации KNN является поиск наиболее оптимального для данной задачи метода. Рассмотрим 4 наиболее распространённых стратегий:

- Метод k-статистик
- Метод brute
- KD-tree
- Ball-tree

2.3.1 Метод k-статистик

Предполагает создание матрицы расстояний между всеми объектами обучающей выборки с последующим нахождением k — порядковой статистики по каждому исследуемому объекту с последующим сортированием такого ряда только по первым k объектам вариационного ряда. Тем самым мы находим матрице попарных расстояний и производить сортировку не всех, а только лишь нужных нам объектов

2.3.2 Метод brute

Более простым и наивным способом является метод brute-force. Точно так же создается матрица попарных расстояний, однако в ней уже напрямую идет перебор всех элементов, чтобы найти k самых близких, расположенный в порядке возрастания расстояния

2.3.3 Метод KD-tree

Известный метод построения дерева поиска, что разбивает пространство плоскостями, тем самым отделяя каждый объект от остальных в отдельных листьях. Должен быть быстрее предыдущих на стадии нахождения k — соседей, но медленнее на этапе построения дерева

2.3.4 Метод Ball-tree

Данный метод имеет точно такие же характеристики, что и kd-tree за тем лишь исключением, что он разбивает пространство шарами

3 Оценка быстродействия и эффективности алгоритмов

Важным параметром работы алгоритмов является его быстродействие. Посмотрим на скорость выполнения каждого из четырех стратегий KNN при $|\mathcal{D}| = 60000$ и размерности тестовой выборки в 10000 объектов, при условии поиска по 5 соседей. Действие классификатора подразделяется на две стадии — построение нужной структуры данных для работы (матрица попарных расстояний или же kd и ball деревья), результаты на каждом из которых представлены в таблицах 2 и 3, где отображено, как четыре метода KNN ведут себя по времени при взятии случайным образом 10, 20 и 100 признаков соответственно.

Заметно превосходство не библиотечного метода k-статистик на стадии построения матрицы расстояний. Он не идет в никакое сравнение с ни одним прочим методом, обгоняя их на порядки. Библиотечные алгоритмы показывают примерно одинаковое время, хотя алгоритмам kd-tree и ball-tree явно требуется больше времени на построение поисковых деревьев.

> Легко заметить, что значения скорости обработки алгоритмов сильно зависит от параметров, что участвуют в классификации. Это заметно на второй строке таблицы 2, соответствующей 20 признакам, где время работы трех библиотечных стратегий резко упало.

Однако прирост скорости на обработке обучающей выборки алгоритма k-статистик сильно отыгрывается прочими алгоритмами на этапе самой классификации. Здесь мы видим явное преимущество алгоритмов библиотеки. При этом заметна тенденция на то, что с увеличением количества признаков, разреженность пространства, в которое мы переводим объекты, увеличивается,

что выливается в стремительный рост сложности поисковых деревьев. Из-за чего они быстро начинают проигрывать методу brute. При этом наихудшие показатели у метода ball-tree, что, вероятнее всего, является следствием разряженности итогового пространства.

4 Оценка точности алгоритма

Выбор стратегии поиска ближайших соседей никак не влияет на качество самой классификации. Однако крайне важно понимать, как хорошо работает модель на заданных данных, чтобы иметь возможность предсказать, как она будет справляться с классификацией на других объектах. Оценим работу KNN на различных подмножествах обучающей выборки, чтобы получить оценку устойчивости модели при различных метриках: косинусной и евклидовой. Вместе с этим крайне важной составляющей точности модели KNN является параметр k, что указывает на количество объектов, что могут влиять на принимаемое решение по классификации

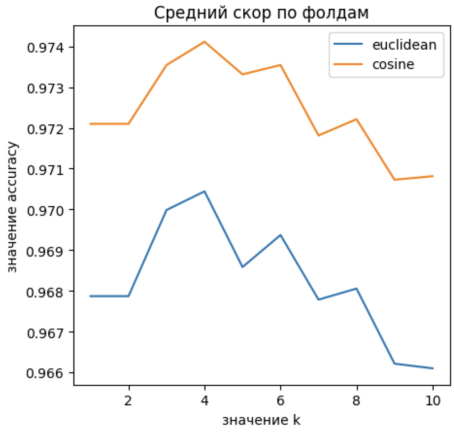


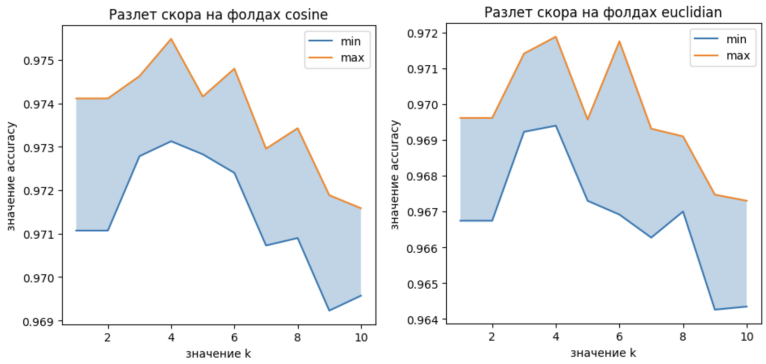
Figure 2: Средняя по фолдам accuracy

Воспользуемся CCV (кроссвалидацией) для определения точности алгоритма по оценке **accuracy** при k пробегающем значения от 1 до 10. Разобьем обучающую выборку на 3 части и будем поочередно каждую из частей считать валидационной, а прочие две обучающей выборкой. При этом важно понимать, как ведет себя ошибка для каждой метрике и значению k в среднем по трем фолдам, а так же максимальное значение accuracy и минимальное из трех фолдов

Как видно из графика 2, определяющего среднее значение accuracy на всех трех фолдах, лучше всего себя показывает именно алгоритм с метрикой cosine, при чем лучший результат добивается при 4 ближайших соседях. Если же посмотреть на графики 4

То мы увидим, что на самом что косинусная метрика еще и ведет себя более предсказуемо, стабильно. В среднем разница между максимальным и минимальным по фолдам значениями accuracy для косинусной метрики меньше, чем для евклидовой, особенно на значениях близких к 4. Что на самом деле ожидаемо, если учесть, что косинусная метрика не учитывает разницу по абсолютному значению между объектами

> Примером этому может служить картинка с одной и той же цифрой, но разной интенсивностью цвета. Интенсивность цвета выливается в увеличение элементов вектора, следовательно евклидова метрика будет



(a) (b)

Figure 3: (a) разлет скора на метрике cosine (b) разлет скора на метрике euclidian

> показывать отличные от нуля значения, в то время как косинусная останется одной и той же

Однако главным итогом этого эксперимента можно считать то, что мы увидели зависимость между увеличением числа рассматриваемых соседей и падением качества алгоритма. Это можно объяснить тем, что с увеличением числа рассматриваемых прецедентов, увеличивается и число тех объектов, что попасть в число прецедентов данного класса не должны были бы из-за своей удаленности. Вместе с тем в самом начале мы видим, рост как средней Accuracy, так и падение разницы между минимальным и максимальным значениями Accuracy по фолдам, что показывает, что 1 не является оптимальным значением.

5 Оценка использования весов

Из прошлого пункта вытекает, что существует существенное падение качества алгоритма при большом увеличении количества ближайших соседей. Попробуем применить взвешенных метод, чтобы избавиться от такого эффекта

Проведем эксперимент с взвешенным алгоритмом и с классическим на основе косинусной метрики при $k = \{1, 2, 10, 30, 100, 150, 300\}$, чтобы оценить динамику оценки accuracy от k

Как можно увидеть из графиков 4, на больших числа взвешенный алгоритм ведет себя лучше, при этом это никак не решило проблему с существенной разницей между максимальным и минимальным значением accuracy на фолдах

6 Оценка результатов

Посмотрим на сам датасет. Насколько он репрезентативен и насколько данные в нем распределены равномерно. Для этого вновь разобьем всю выборку в отношении 6:1. Обучим knn на данной выборке и посчитаем Accuracy. На таком же knn посчитаем среднее Accuracy по всем фолдам на CCV с количеством фолдов, равному 5.

> Если на валидации оценка точности алгоритма сильно разнится с усредненной оценкой по фолдам, то это может сигнализировать о том, что-либо данные в последовательности, полученной из начального обучающего множества, были распределены не равномерно, либо что алгоритм сам по себе не устойчив.

Возьмем модель knn со стратегией "kd_tree", значением $k = 4$ и косинусной метрикой. Тогда мы получим 0.9759 для CCV и 0.9752 для простого knn. Результаты очень похожи между собой, что говорит, что прецеденты были равномерно распределены, и мы имеем хорошо настроенный классификатор. Посмотрим на официальные лучшие результаты по миру в таблице 4. Как можно заметить, результаты классификации, что есть сейчас, достаточно велики, однако это не максимум, что можно получить от задачи.

Посмотрим на то, какие ошибки у нас повторяются наиболее часто. Для этого создадим матрицу ошибок 5 C_{ij}, где элемент с индексами i и j соответствует количеству наблюдений объектов, причисляемых к классу i, однако отнесенных к классу j. Исходя из данной таблицы можно определить, объекты какого класса часто путаются с объектами другого, при чем в общем случае эта матрица не симметрична.

Посмотрим на самые большие числа в таблице — по ним видно, что самыми сложными для классификации цифрами являются 9 (что часто ошибочно относится к 4 или 7), 1 (часто относится классификатором к 7 или 9), 5 (путается с 3), и так далее. Объединяет все эти объекты то, что они по написанию слабо различаются с теми, с которыми их путает

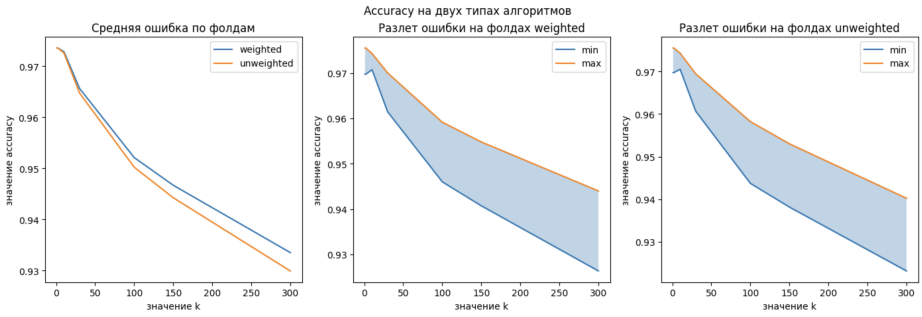


Figure 4: Оценки accuracy по среднему и по размаху на фолдах при взвешаном и классическом KNN

классификатор. Пример такого можно увидеть на 5

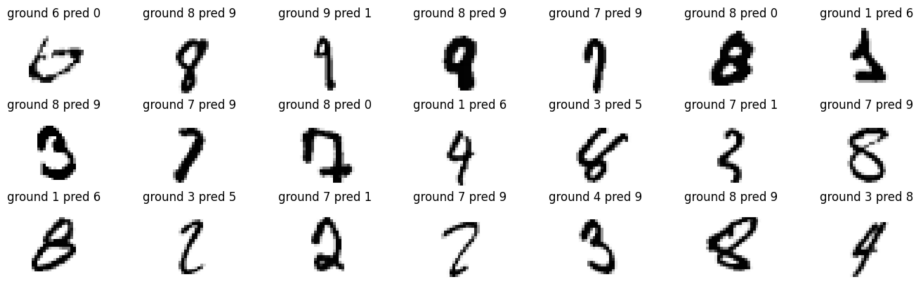


Figure 5: Примеры неверно классифицированных объектов

7 Аугментация данных

Чтобы правильно классифицировать проблемные объекты, нам требуется уметь выделять из них характерные для их класса признаки. Ситуация осложняется тем, что различные классы, они же цифры, имеют разные характерные признаки (черты). Поэтому для отличия 7 от 1 нужно учитывать разницу по горизонтали, в то время как, чтобы отличить 9 от 4, нужно уменьшить толщину линий на фото и сделать переход между черным и белым явнее.

Посмотрим на сам датасет, отобразив пространство признаков $\mathcal{X}$ и сам $\mathcal{D}$ выборку на плоскость, построив ембеддинги размерности $\mathbb{R}^2$. Воспользуемся методом **t-SNE** — T-distributed Stochastic Neighbor Embedding 6, где каждым цветом отобразим свою цифру. На данном отображении явно видно, почему наш метрический алгоритм хорошо подходит по данную задачу: точки одного цвета, ассоциируемые с цифрами одного класса, очень хорошо группируются в класстеры. Однако есть множество точек, что находятся в чужих кластерах. Именно такие точки очень сложно классифицировать правильно

Проведем аугментацию обучающей выборки: из первоначальной выборки изображений посредством морфологических преобразований получим новые датасеты, что в будущем будем использовать совместно с основной выборкой.

7.1 Виды аугментаций

Исходя из семантики задачи, нам требуется выделять особенные признаки классов, а так же добавить дополнительные, легко измененные, прецеденты каждого из классов, чтобы увеличить устойчивость модели

Используемые методы аугментации:

- Повороты — по и против часовой стрелки на 5, 10, 15 градусов соответственно
- Сдвиги — по каждой оси в обоих направлениях на 1, 2 и 3 пикселя

- Дисперсия (размытие) по Гауссу — размытие с коэффициентами 0.5, 1 и 1.5
- Эрозия с ядром 2×2 в одну итерацию
- Дилатация с ядром 2×2 в одну итерацию
- Открытие с ядром 2×2
- Закрытие с ядром 2×2

7.1.1 Оценка прироста качества алгоритма от каждой аугментации

Создадим дополнительные выборки для каждой из аугментаций и посмотрим, как будет вести себя модель KNN. Будем сращивать начальную и одну из аугментационных выборок и проверять среднее значение качества на фолдах при кросс-валидации **только** по начальным объектам. Сравним приросты качества классификации **взвешенным** алгоритмом KNN с использованием косинусной метрики, при количестве соседей $k = 4$ как среднее от Accuracy по всем фолдам и посчитаем матрицы ошибок, как суммы матриц ошибок по каждому из фолдов.

> Использование взвешенного KNN здесь важно, ведь оно способно свести на нет шумовые совпадения оригинального объекта и видоизмененного из аугментации

Результаты представлены в таблице 1, где мы можем заметить убавление качества алгоритма на всех видах аугментаций. Такое поведение можно объяснить тем, что добавление аугментаций засоряет выборку, некоторые объекты, наиболее не похожие прочих представителей соответственного класса, становятся ближе к видоизмененным объектам уже чужого класса.

Вместе с тем мы видим, что в таблице повторяются одни и те же значения, что указывает на та, что классификатором явным образом отвергалось участие аугментированных объектов

7.1.2 Оценка прироста устойчивости алгоритма

Важным аспектом качественного классификатора является его устойчивость. Оценим качество работы алгоритма кроссвалидацией по выборке, что полученная объединением как начальной, так и аугментационной (каждой по отдельности), тем самым сымитировав возможное реальное использование модели. Если в среднем на видоизмененных объектах качество будет большим, значит модель можно считать устойчивой.

> Заметим, что в этот раз мы рассматриваем пары "$\mathcal{D}$ + аугментация" как все новое множество. В прошлом разделе аугментация рассматривалась как дополнительные объекты только в моменте обучения классификатора

Так и происходит: усредненное accuracy по фолдам равно 0.974, при чем для каждой из аугментаций, что сопоставим с наилучшим показателем модели и при этом является результатом построения устойчивого классификатора.

7.2 Построение улучшенного классификатора

Прошлые наблюдения показывают, что каждая из аугментаций по отдельности не способна привнести улучшения в существующую модель, однако способна улучшить ее устойчивость. Главной задачей теперь является построение качественного классификатора, базирующегося не только на начальной выборке, но и дополненной аугментациями.

Есть два возможных подхода к реализации такого алгоритма:

- Находить по k ближайших соседей из каждой из выборок и по ним уже строить классификацию
- Находить классификацию по каждому из объектов и применять голосование

7.2.1 Классификатор с использованием всех аугментаций

Построим такой классификатор, что для заданного объекта будет находить по k ближайших соседей по косинусной метрике среди начальной выборки и каждого из аугментационных множеств. Тогда мы получим $26k$ ближайших соседей из различных множеств, по которым уже строится классификация. Такое решение должно быть самым устойчивым к выбросам, однако не самым оптимальным с точки зрения эффективности вычислений и отбрасывания неэффективных элементов в аугментированной выборке.

> Посчитав среднее от Accuracy от кроссвалидации по трем фолдам, мы получим значение 0.9731, что очень близко к максимальному нашему значению в 0.9752

Table 1: Матрица качества аугментации

Поворот 5	Поворот -5	Поворот 10	Поворот -10	Поворот 15	Поворот 15
0.9698	0.9698	0.9698	0.9698	0.9698	0.9698
Сдвиг гор.1	Сдвиг гор.2	Сдвиг гор.3	Сдвиг гор.-1	Сдвиг гор.-2	Сдвиг гор.-3
0.9698	0.9698	0.9698	0.9698	0.9698	0.9698
Сдвиг верт.1	Сдвиг верт.2	Сдвиг верт.3	Сдвиг верт.-1	Сдвиг верт.-2	Сдвиг верт.-3
0.9698	0.9698	0.9698	0.9698	0.9698	0.9698
Гаус 0.5	Гаус 1	Гаус 1.5	Эрозия/Дилатация	Открытие	Закрытие
0.974	0.974	0.974	0.974	0.969	0.974

7.2.2 Классификатор, основанный на алгоритме голосования

На этот раз построим 26 моделей knn, обученных на каждой из выборок (начальной и каждой аугментации) по отдельности, и построим список из предсказаний, где каждый knn стоит в порядке увеличения средней Accuracy из таблицы 1. Тогда классификатор будет работать по принципу выбора самого часто встречаемого класса в массиве ответов.

> Условие ранжирования моделей knn в порядке уменьшения качества позволяет выбирать ответ на основе нашего доверия к их ответу. Из таблицы качества каждой из моделей по отдельности можно получить оценку того, какая из аугментаций привносит больше в конечное качество классификатора, а какая меньше (делается это из среднего качества на фолдах: больше значит лучше).

Построив алгоритм прогоним его на той же тестовой выборке и получим, что нам удалось повысить качество модели до 0.9816 по Accuracy, что составляет всего 1.8% ошибок, тем самым показав, что алгоритм голосования работает лучше на данном датасете.

8 Итоги

В данном отчете мы анализировали построенный алгоритм KNN на датасете "mnist784", на котором нам удалось выявить наиболее удачные параметры для стратегии поиска ближайших соседей, метрики, что дает лучшие результаты, а так же смогли улучшить начальную выборку аугментациями, что позволило нам добиться процента ошибок на тесте всего в 1.8%

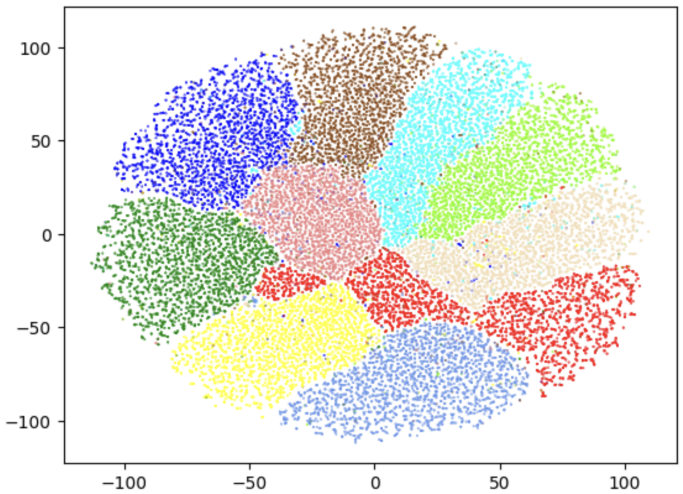


Figure 6: Отображение пространства векторов $\mathcal{Z}$ на плоскость

Table 2: Время построения расстояний в секундах

Кол-во параметров	Метод k-статистик	Метод brute	KD-tree	Ball-tree
10	10^{-5}	0.0018	0.0799	0.128
20	$9.29\ 10^{-6}$	0.0023	0.365	0.146
100	$1.12\ 10^{-5}$	0.025	1.798	1.085

Table 3: Время поиска k ближайших соседей в секундах

Кол-во параметров	Метод k-статистик	Метод brute	KD-tree	Ball-tree
10	23.932	13.183	4.534	3.824
20	52.082	15.63	10.188	39.934
100	228.291	18.578	189.85	200.445

Table 4: Показатели метрических классификаторов на базе KNN

Модель	процент ошибок	Accuracy
Наш KNN	2.48%	0.9752
K-NN with non-linear deformation (IDM)	0.54%	0.9946
K-NN, Tangent Distance	1.1%	0.989
K-NN, shape context matching	0.63%	0,9937

Table 5: Матрица ошибок на оригинальном датасете

	0	1	2	3	4	5	6	7	8	9
0	977	0	8	0	2	4	3	2	7	7
1	1	1129	0	1	1	0	3	10	1	7
2	0	3	1009	3	0	0	0	4	2	2
3	0	1	1	976	0	9	0	0	9	5
4	0	0	1	1	946	1	1	1	3	7
5	0	0	0	12	0	863	3	0	3	3
6	1	2	0	0	6	7	948	0	5	1
7	1	0	8	4	2	1	0	998	4	4
8	0	0	5	9	0	4	0	0	936	3
9	0	0	0	4	25	3	0	13	4	970