МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра вычислительной математики
и прикладных информационных технологий

Применение персистентых гомологий для задачи классификации изображений

Выпускная квалификационная работа
Направление 01.03.02 Прикладная математика и информатика
Профиль Математическое моделирование и вычислительная математика

Зав. кафедрой _____ д.т.н., проф. Т. М. Леденева
Обучающийся _____ П. С. Снопов
Руководитель _____ д.т.н., проф. Т. М. Леденева

Воронеж 2021

Содержание

Введение

Топологический анализ данных – это новая область анализа данных, которая своими корнями уходит в алгебраическую топологию. Она возникла в результате прорывных работ Герберта Эдельсбруннера [6] и Гуннара Карлссона [7] по устойчивым гомологиям, которые впоследствии стали основными инструментами данной сферы.

В задачах анализа данных большое значение имеет информация о геометрической и топологической структуре исследуемых данных. Зачастую такая структура позволяет выявить некоторые закономерности между различными переменными. Помимо этого, существует также гипотеза о многообразии, которая утверждает, что реальные данные, представленные облаком точек из $\mathbb{R}^n$ лежат на некотором d-многообразии, где $d \leq n$.

Данная работа посвящена обзору современных инструментов топологического анализа данных, а также применению персистентных гомологий, которые являются основным инструментом данного подхода, к задаче классификации на примере стандартного для классификации датасета MNIST изображений рукописных цифр. Основная цель данной работы – построение алгоритма, который, на основе имеющейся изображений в выборке, будет способен определить, к какому классу относится новое изображение, т.е. изображение, не находящийся в исходной выборке.

Актуальность выбранной темы выражается в современным тенденциях прикладной математики: анализ данных и машинное обучение – одна из самых быстроразвивающихся областей, которая за последние десятилетия изменила окружающий нас мир. Так, на сервисе `arxiv.org` – сервисе по

свободному распространению и открытому доступу научных статей в области физики, математики, компьютерных и других вычислительных наук – ежедневно появляется порядка 150 публикаций каждый день [?].

С другой стороны, данная работа не в последнюю очередь – это работа по прикладной топологии. Эта область сейчас переживает свой расцвет: в последние годы появляется все больше прикладных работ, в которых используются устойчивые гомологии. Помимо этого, развивается и теоретический аппарат [?]. Одним из примеров таких работ служат работы, которые направлены на изучение т.н. латентного пространства некоторых видов нейросетей: VAE и GAN. Эти нейросети в ходе обучения пытаются воссоздать то самое многомерное многообразие, на котором лежат данные, например, изображения. Латентное пространство зачастую скрыто во многих методах анализа данных, и содержит всю необходимую информацию, поэтому изучение его свойств крайне важно. Топологический анализ данных, в силу своего топологического фундамента, может быть крайне полезен для такой задачи. Данная тема является популярной темой современных исследований в этой области.

В ходе данной работы был изучен теоретический материал по алгебраической и прикладной топологии, машинному обучению и тому, как связать, казалось бы, не связываемые на первый взгляд эти две области математики друг с другом. Была написана программа, которая для каждого изображения из датасета MNIST создает его векторное представление на основе топологических свойств изображения. Была выбрана модель машинного обучения, которая дает наилучший результат в поставленной задаче классификации. Таким образом, был получен алгоритм, который с высокой(более 90%) точностью способен определить рукописную цифру, оцифрованную в виде изображения, который в первую очередь эксплуатирует именно топологические особенности.

В ходе данной работы было также выявлено, что реализованный ал-

горитм является эффективным алгоритмом понижения размерности пространства признаков. Данный алгоритм сравнивался с другими моделями машинного обучения, которые были обучены обычным образом – на вход моделям подавалось плоское представление картинки. В результате сравнения было получено, что алгоритм классификации, который использовал топологические признаки, показывал более высокую точность при меньшем числе признаков, чем стандартные алгоритмы.

1 Элементы общей и алгебраической топологии

1.1. Некоторые сведения из топологии

Здесь будут приведены некоторые теоретические сведения из топологии, используемые в дальнейшем. Более подробно об этом можно прочитать в хороших книгах по топологии, например в [1, 2, 3].

Основным объектом в топологическом анализе данных можно считать симплициальный комплекс. В некотором смысле это обобщение графов на более старшие размерности. В частности, по любому графу можно построить (флаговый) комплекс, объявив каждую его клику симплексом.

Выпуклую оболочку набора $\{x_0, ..., x_n\} \subset \mathbb{R}^d$, причем такого, что векторы $x_1 - x_0, ..., x_n - x_0$ линейно независимы, называют n-мерным симплексом. Гранью n-симплекса называют k-симплекс, который получается как выпуклая оболочка некоторого подмножества вершин этого симплекса.

Так, k-симплекс можно рассматривать как k-мерное обобщение треугольника и тетраэдра. На рисунке 1.1 приведены примеры симплексов.

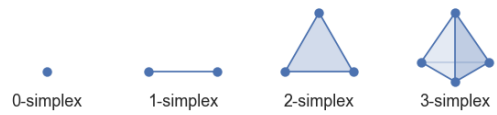


Рис. 1.1. Различные симплексы

Определение 1. *Топологический симплициальный комплекс K – это множество симплексов такое, что:*

1. *Для каждого симплекса из K его грани тоже лежат в K.*

2. *Пересечение любых двух симплексов $\sigma, \tau \in K$ либо пусто, либо является гранью и σ, и τ.*

На рисунке 1.2 представлен пример симплициального комплекса. Так как среди всех симплексов, входящих в представленный на рисунке комплекс, размерность не выше 3, то и размерность симплициального комплекса равна 3. По топологическому симплициальному комплексу можно

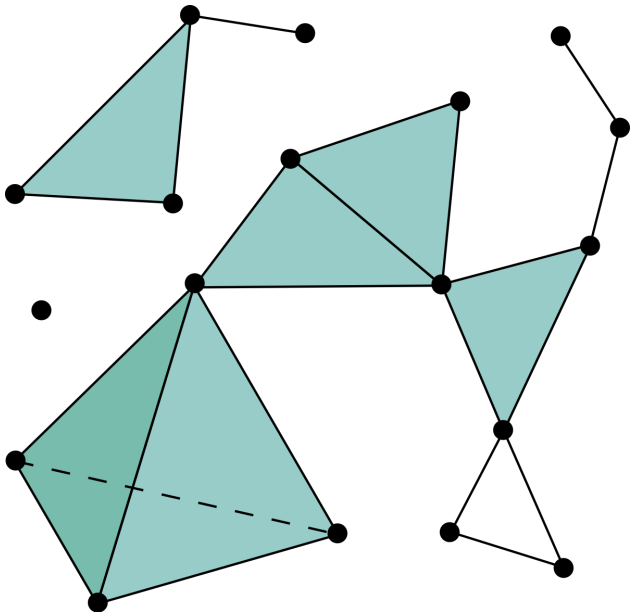


Рис. 1.2. Симплициальный комплекс

построить топологическое пространство – объединение его симплексов. Такие пространства тоже называют симплициальными комплексами. В таком пространстве топология индуцируется топологией на $\mathbb{R}^d$.

Имея топологическое пространство X, говорят, что набор его подмножеств $\{U_i : i \in J\}$ является *покрытием*, если $\bigcup\limits_{i \in J} U_i = X$. Покрытие называется *открытым*, если оно состоит из открытых множеств.

По покрытию пространства можно построить нерв покрытия (рис. 1.3) – симплициальный комплекс, соответствующий топологическому пространству и имеющий различные интересные топологические свойства. Пусть $[m] \coloneqq \{1, ..., m\}$ – m-элементное множество.

Определение 2. *Нерв покрытия $N(U)$, соответствующий топологическому пространству X, – это абстрактный симплициальный комплекс на $[m]$, такой что $\{k_1, ..., k_s\} \in N(U)$ если $U_{k_1} \cap ... \cap U_{k_s} \neq \emptyset$.*

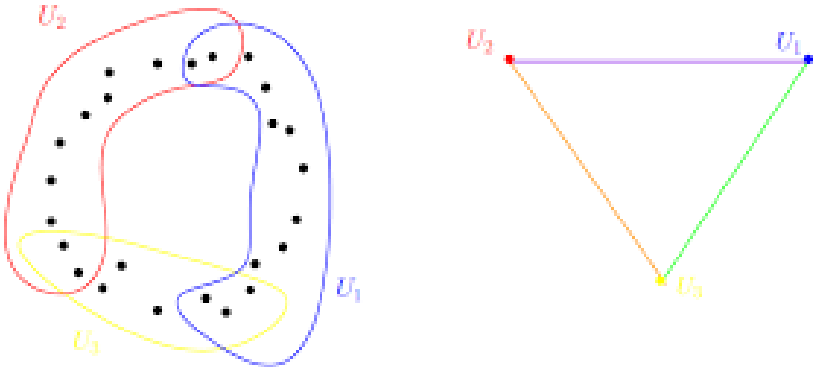


Рис. 1.3. Покрытие пространства и нерв покрытия

Напомним, что если имеется 2 непрерывных отображения $f, g : X \to Y$, то говорят, что f *гомотопно* g, если существует такое отображение $H : X \times I \to Y$, что $H(x,0) = f$ и $H(x,1) = g$. В таких случаях пишут, что $f \sim g$. Гомотопия является отношением эквивалентности на множестве непрерывных отображений $C(X,Y)$.

Два топологических пространства X,Y гомотопически эквивалентны ($X \sim Y$), если существует такая пара $(f,g) \in C(X,Y)^2$, такая, что $f \circ g \sim id_Y$ и $g \circ f \sim id_X$.

С понятием нерва покрытия связана очень важная лемма:

Лемма (о нерве)**.** *Если U – открытое покрытие, такое, что любое пересечение его подмножеств либо пусто, либо гомотопически эквивалентно точке, то нерв покрытия пространства X гомотопически эквивалентен самому пространству*

$$\forall I \in N(U) \left(\bigcap_{i \in I} U_i = \emptyset \vee \bigcap_{i \in I} U_i \sim pt \to N(U) \sim X \right).$$

.

Как всякая хорошая математическая структура, симплициальные комплексы образуют *категорию* **SCpx**. Морфизмами в такой категории являются симплициальные отображения $\varphi : M \to N$, т.е. такие отображения, что для каждого симплекса из M φ является линейным отображением на симплекс из N.

Имея симплициальный комплекс, можно найти его симплициальные гомологии. Такие гомологии можно рассматривать как функтор из категории симплициальных комплексов в категорию абелевых групп: каждому симплициальному комплексу сопоставляется некоторая абелева группа. Замечательный момент заключается в том, что гомеоморфные симплициальные комплексы имеют одинаковые группы гомологий, а потому теория гомологий является очень удобным инструментом для изучения симплициальных комплексов. На самом деле данные утверждения можно легко обобщить на общих случай топологических пространств.

Итак, пусть K – симплициальный комплекс и $k \geq 0$.

Группой k-цепей $C_k(K)$ симплициального комплекса K называют (свободную абелеву) группу, элементами которого являются формальные суммы k-симплексов K, то есть

$$c = \sum_i \varepsilon_i \sigma_i, \;\; \varepsilon_i \in \mathbb{Z},$$

где конечное число $\varepsilon_i \neq 0$, σ_i – k-симплекс. *Границей* k-цепи $\sum\limits_i \varepsilon_i \sigma_i$ называют $(k-1)$-цепь

$$\partial_k(\sigma) = \sum_{j=0}^{k} (-1)^j \sum_i \varepsilon_i \partial_j \sigma_i,$$

где $\partial_j \sigma_i = \partial_j [v_0, ..., v_k] = [v_0, ..., \hat{v_j}, ..., v_k]$ – $(k-1)$-симплекс, порожденный всеми вершинами, кроме вершины v_j. Гомоморфизм $\partial_k : C_k(K) \to C_{k-1}(K)$ называют *граничным оператором*. Он удовлетворяет следующему свойству:

$$\partial_{k-1} \circ \partial_k = 0.$$

То есть $\operatorname{im} \partial_{k+1} \leq \ker \partial_k \leq C_k(K)$. Последовательность $C_k(K)$ и ∂_k называется *цепным комплексом*

$$\ldots \xrightarrow{\partial_{k+2}} C_{k+1} \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \ldots \xrightarrow{\partial_1} C_0.$$

Цепной комплекс можно визуализировать следующим образом 1.4.

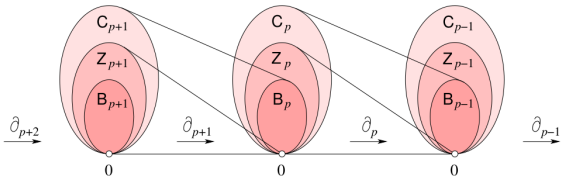


Рис. 1.4. Цепной комплекс

Определение 3. *k-ой группой гомологий симплициального комплекса K называют следующее фактор-пространство:*

$$H_k(K) = \ker \partial_k \,/ \operatorname{im} \partial_{k+1}.$$

Тогда k-ое число Бетти – размерность k-ой группы гомологий: $\beta_k(K) = \dim H_k(K)$.

Пространство	β_0	β_1	β_2
Pt	1	0	0
D^2	1	0	0
Треугольник	1	0	0
Граница треугольника	1	1	0
S^1	1	1	0
S^2	1	0	1
$\mathbb{T}^2 = S^1 \times S^1$	1	2	1

Таблица 1.1. Первые числа Бетти для некоторых пространств

При $k = 0$ число Бетти описывает количество компонент связности данного пространства. При $k = 1$ – количество циклов. При $k = 2$ число Бетти описывает количество "полостей".

1.2. Персистентные гомологии и их векторные представления

1.2.1 Персистентные гомологии

О персистентных гомологиях можно думать как об адаптации понятия гомологии к облаку точек.

Имея облако точек X, существует много способов построения симплициальных комплексов по нему. Например, задав некоторый $\tau > 0$, можно рассматривать не просто точки, но и их окрестности с радиусом τ. Тогда можно рассматривать следующую структуру: подмножество $\sigma \subset X$ будет симплексом, если пересечение окрестностей точек из σ будет непусто. Тогда симплициальным комплексом, построенным по облаку точек, будет совокупность таких симплексов. Такой симплициальный комплекс называется *комплексом Чеха* $\mathrm{Cech}_\tau(X)$.

На комплекс Чеха можно взглянуть с другой стороны: если данные действительно хорошо описывают некоторый топологический объект, который как бы стоит за этими данными, то, подобрав хороший радиус τ, объединение полученных окрестностей будет гомотопически эквивалентно этому объекту. А так как все шары выпуклы, то объединение окрестностей будет гомотопически эквивалентно нерву данного покрытия. А значит, по лемме о нерве, сам нерв будет гомотопически эквивалентен топологическому объекту, который описывается данными. И комплекс Чеха как раз и является нервом покрытия.

Но на практике оказывается, что комплекс Чеха очень сложно посчитать. Поэтому существует другой способ построения симплициального комплекса по облаку точек, более эффективный с точки зрения вы-

числений: зафиксировав τ, подмножество $\sigma \subset X$ будет симплексом, если $\forall i,j \in \sigma : \rho(i,j) < \tau$. Симплициальным комплексом будет совокупность таких симплексов. Полученный симплициальный комплекс называется *комплексом Вьеториса—Рипса*(или просто *комплексом Рипса*) $\mathrm{Rips}_\tau(X)$.

Алгоритм 1: Алгоритм построения комплекса Вьеториса—Рипса

Исходные параметры: Облако точек X, вещественное число $\alpha > 0$.

Результат: Симплициальный комплекс Вьеториса—Рипса

Для каждой точки x строим её α-окрестность $B_\alpha(x)$;

$i = 1$;

до тех пор, пока $i+1$ *окрестностей попарно имеют непустое пересечение*

выполнять

строим i-ый симплекс на соответствующих вершинах;

$i \leftarrow i + 1$;

конец

То есть, вместо рассмотрения пересечений шаров, как это было в случае комплекса Чеха, рассматривают попарные расстояния между точками. Очевидно, такой подход более эффективен с вычислительной точки зрения, более того, его можно обобщить на общий случай конечного метрического пространства, и даже, более общо, на случай произвольного конечного множества с симметрической функцией.

Комплексы Чеха и Рипса связаны между собой: для облака точек $X \subset \mathbb{R}^d$ они имеют одинаковый 1-остов. Более того, справедливо следующее соотношение:

$$\mathrm{Rips}_\tau(X) \subseteq \mathrm{Cech}_\tau(X) \subseteq \mathrm{Rips}_{2\tau}(X).$$

На рис. 1.5 изображены комплексы Чеха и Рипса.

Везде выше параметр τ был зафиксирован, но его можно изменять. Вслед за этим будут меняться и симплициальные комплексы, построенные с учетом τ. *Фильтрацией симплициального комплекса* K называют вложенное семейство подкомплексов $(K_\tau)_{\tau \in T}$, где $T \subseteq \mathbb{R}$, такое, что если $\tau < \tau'$, то $K_\tau \subseteq K_{\tau'}$. Пример такой фильтрации изображен на рис. 1.6, где

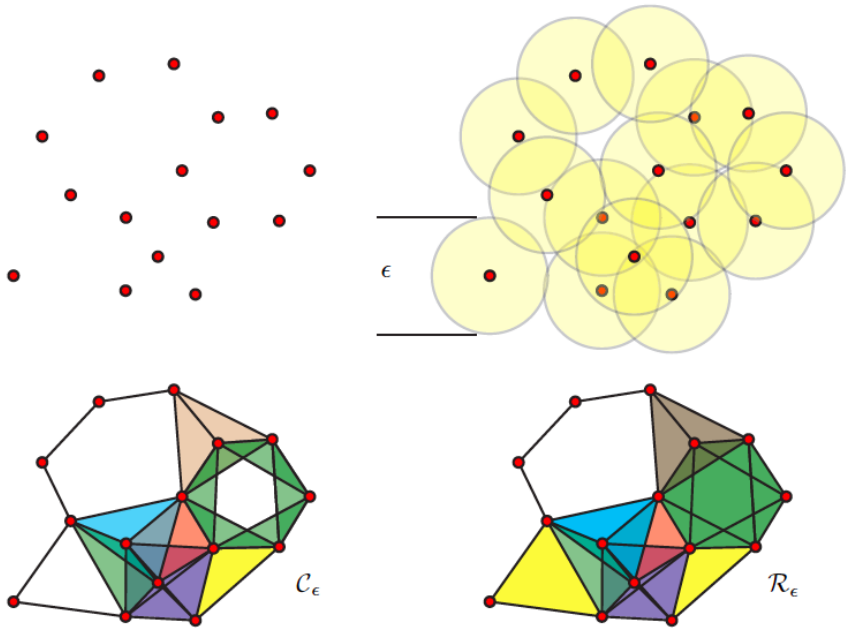


Рис. 1.5. Комплексы Чеха и Вьеториса—Рипса

изображения фильтрация симплициальных комплексов Рипса $\mathrm{Rips}_\tau(X)$.

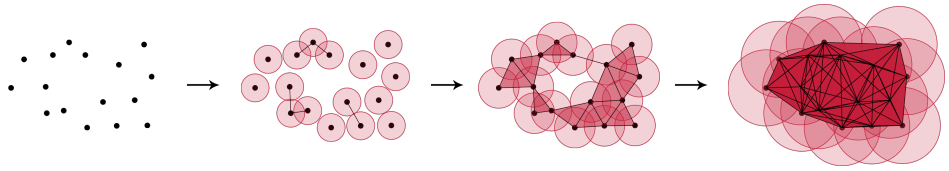


Рис. 1.6. Фильтрация Рипса

Имея фильтрацию $(K_\tau)_{\tau\in T}$, можно отслеживать изменение $H_k(K_\tau)$ с ростом τ: могут появляться новые компоненты связности, уже существующие могут объединяться в одну компоненту, могут появляться циклы и "пустоты соответствующие 1 и 2 группе гомологий.

Определение 4. *k-ыми устойчивыми гомологиями фильтрованного комплекса $(K_\tau)_{\tau\in T}$ называют проиндексированное семейство абелевых групп $H_n(T) = \{(H_n(K_\tau))_{\tau\in T}$ вместе с семейством гомоморфизмов $(H_n(K_\tau) \to H_n(K'_\tau))_{\tau\leq\tau'}\}$.*

Такое семейство абелевых групп вместе с морфизмами на самом деле

является примером персистентного $\mathbb{Z}$-модуля: *персистентным R-модулем* (A_*, x) называют последовательность R-модулей и гомоморфизмов между ними (над целочисленной временной шкалой $\mathbb{Z}_{\geq 0}$).

$$A_0 \xrightarrow{x} A_1 \xrightarrow{x} A_2 \xrightarrow{x} A_3 \xrightarrow{x} ...$$

Основная теорема про персистентные модули – это структурная теорема. По аналогии с обычными модулями, она формулируется в терминах *конечнопорожденного* модуля устойчивости, т.е. такого модуля (A_*, x), у которого существует конечный набор $a_1, ..., a_k \in A_*$ элементов, что любой элемент $a \in A_*$ можно выразить в виде линейной комбинации элементов вида $x^s a_r = x \circ ... \circ x(a_r)$. Для ее формулировки понадобится еще пару определений: если $0 \leq j < s$, то *интервальным модулем устойчивости* $I_{[j,s)}$ называют модуль устойчивости вида

$$0 \longrightarrow ... \longrightarrow 0 \longrightarrow \underset{j}{R} \xrightarrow{id_R} \underset{j+1}{R} \xrightarrow{id_R} ... \xrightarrow{id_R} \underset{s-1}{R} \xrightarrow{0} \underset{s}{0} \longrightarrow ...$$

Аналогично определяется бесконечный интервальный модуль $I_{[j,\infty)}$

$$0 \longrightarrow ... \longrightarrow 0 \longrightarrow \underset{j}{R} \xrightarrow{id_R} \underset{j+1}{R} \xrightarrow{id_R} ... \xrightarrow{id_R} R \xrightarrow{id_R} ...$$

Можно определить прямую сумму персистентных модулей: $(A_*, x) \oplus (B_*, x) = ((A_* \oplus B_*), x)$, где $(A_* \oplus B_*)_j = A_j \oplus B_j$, а x действует на каждом из слагаемых по отдельности.

Теорема (об интервальном разложении)**.** *Пусть R – произвольное поле, а (A_*, x) – конечнопорожденный персистентный R-модуль. Тогда (A_*, x) имеет единственное с точностью до перестановки слагаемых представление в виде прямой суммы конечного числа интервальный модулей:*

$$(A_*, x) \simeq \Big(\bigoplus_k I_{[j_k, s_k]}\Big) \oplus \Big(\bigoplus_k I_{[r_k, \infty)}\Big)$$

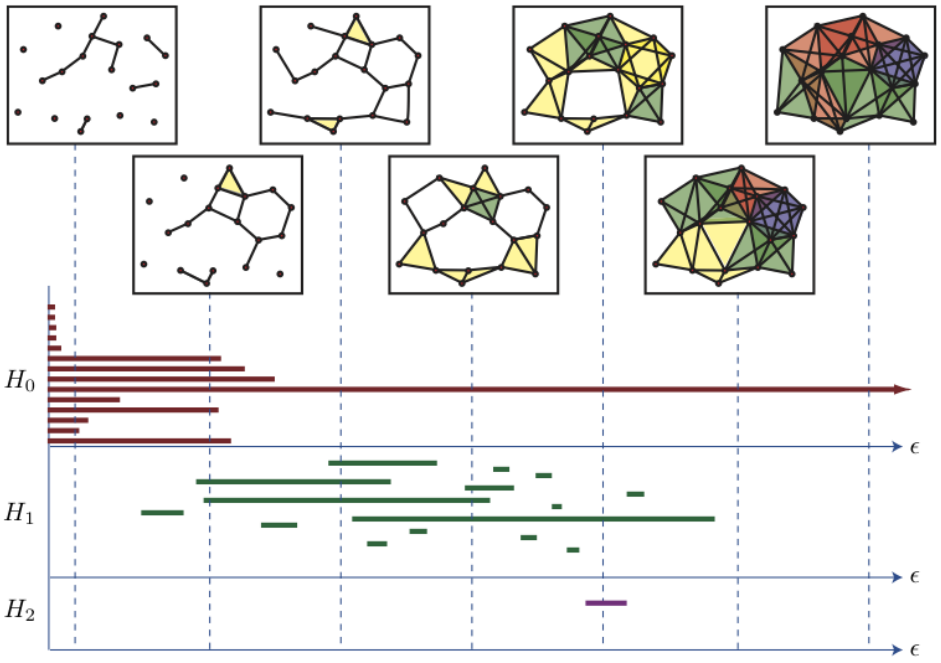


Рис. 1.7. Баркод

Так как персистентные гомологии – это типичный пример конечнопорожденного персистентного модуля, то к нему можно применить структурную теорему. Из теоремы следует, что любой конечнопорожденный персистентный модуль с коэффициентами в поле, а значит и персистентные гомологии, можно закодировать в виде баркода (рис. 1.7) – диаграммы, которая содержит интервалы, которые отвечают за время жизни свойств, которые как раз и характеризуются персистентными гомологиями.

Эту же информацию можно закодировать в виде диаграммы персистентности (рис. 1.8). Она строится следующим образом: каждый интервал баркода имеет начало t_{birth} и конец t_{death}. На персистентной диаграмме каждый интервал баркода изображается в виде точки с координатами (t_{birth}, t_{death}), и каждая такая точка соответствует одному из слагаемых $I_{[j_k,s_k]}$ и $I_{[r_k,\infty)}$ из разложения. Чем дальше точка на персистентной диаграмме от диагонали, тем она важнее – эта точка сигнализирует о наличии "n-мерной дырки".

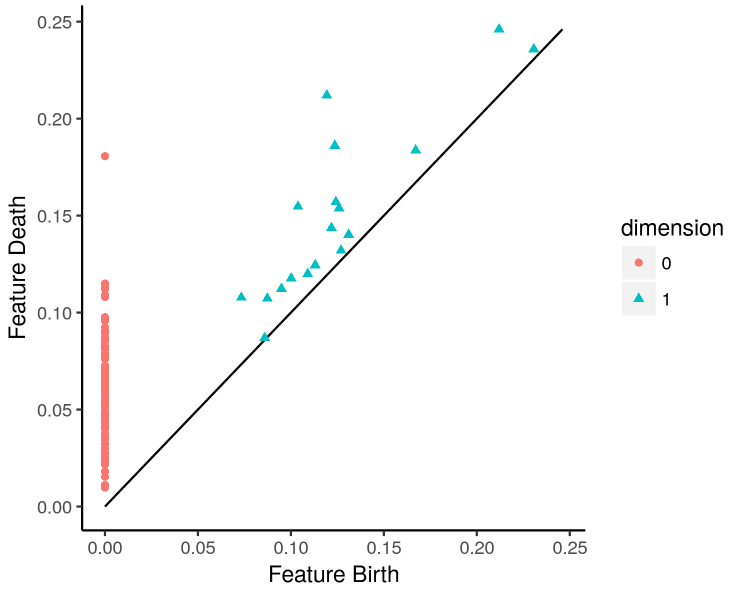


Рис. 1.8. Персистентная диаграмма

1.2.2 Векторные представления

1.3. Элементы машинного обучения

1.3.1 Введение в машинное обучение

Машинное обучение – раздел, стоящий на пересечении математики и компьютерных наук, который главным образом решает задачу восстановления некоторой зависимости, используя некоторые данные, которые представляют такую зависимость.

Машинное обучение можно разделить на категории: например, очень популярное разделение – это разделение по типу обучения:

- Обучение с учителем(Supervised Learning)
- Обучение без учителя(Unsupervised Learning)
- Обучение с подкреплением(Reinforcement Learning)

Остановимся поподробнее на обучении с учителем. Данный вид машинного обучения предполагает, что имеется некоторая выборка X – *мно-*

жество объектов, некоторая выборка Y – *множество ответов*, и необходимо восстановить *скрытую зависимость* $y : X \to Y$. Как правило, саму зависимость восстановить порой очень трудно, поэтому достаточно найти такой алгоритм $a : X \to Y$, который приближает y на X. При этом сами объекты задаются через *признаки* $f : X \to D$, где D – *пространство признаков*. Признаки – это какие-либо характеристики, которые описывают данный объект.

Основные задачи обучения с учителем:

- Регрессия
- Классификация

Фундаментальная разница между этими двумя типами задач следующая: задача *регрессии* – восстановление непрерывной функции, которая характеризует скрытую зависимость между входными переменными и выходными. Регрессия таким имеет возможность предсказывать численное значение этой скрытой непрерывной функции по новым значениям входных переменных. Задача *классификации* же в том, чтобы восстановить зависимость между входными параметрами, описывающие объект, и тем, к какому классу данный объект принадлежит. То есть в случае задачи классификации скрытая зависимость описывается некоторой пороговой функцией, где каждое значение области значения соответствует некоторому классу.

Важным шагом в решении задачи как регрессии, так и классификации, является выбор модели и настройка ее параметров. *Модель* – это параметрическое семейство функций

$$A := \{a(x) = g(x, \theta) | \theta \in \Theta\},$$

где Θ – множество значений параметра θ, а $g : X \times \Theta \to Y$ – некоторая фиксированная функция.

Решение задачи машинного обучения разделяется на два этапа:

1. Этап обучения

2. Этап применения

Этап обучения представляет из себя построение алгоритма выбора подходящей модели, т.е. имеется отображение $\mu : X \times Y \to A$, которое по каждой выборке (X,Y), которую обычно называют *обучающей выборкой*, ставит в соответствие алгоритм $a \coloneqq \mu(X, Y)$, который наилучшим образом описывает скрытую зависимость $y : X \to Y$. Для выбора модели необходимо сравнивать точность, которую они демонстрируют. Для этого обычно вводят т.н. *функцию потерь(loss function)* $L(a,x)$, которая измеряет величину ошибки алгоритма $a \in A$ на объекте $x \in X$. Например для задачи классификации рассматривают *индикатор ошибки(Accuracy)*:

$$L(a,y,x) = [a(x) \neq y(x)].$$

А для задач регрессии в качестве функции потерь рассматривают *квадратичную ошибку*

$$L(a,y,x) = (a(x) - y(x))^2,$$

или *абсолютное значение ошибки*

$$L(a,y,x) = |a(x) - y(x)|.$$

Чтобы оценить качество работы алгоритма a на всей выборке X обычно считают среднее арифметическое по значениям функции потерь для каждого объекта $x \in X$(иногда это называют *эмпирическим риском*):

$$Q(a,X,Y) = \frac{1}{|X|} \sum_{i=1}^{|X|} L(a,y_i,x_i).$$

И тогда для выбора модели решают задачу минимизации эмпирического риска:

$$\mu(X,Y) = \arg\min_{a \in A} Q(a,X,Y).$$

Этап применения – это следующий этап, который представляет собой т.н. задачу вывода. На этом этапе выбранный алгоритм a для новых объектов X^*, которые не были задействованы в этапе обучения, выдает ответы $a(X^*)$. Обычно такую выборку X^* называют *тестовой выборкой*. Одной из основных проблем, которые возникают на данном этапе – это проблема *переобучения*. Переобучение – это ситуация, которая возникает в следствие допущенной ошибки в ходе этапа обучения, и характеризуется тем, что величина ошибки на тестовой выборке сильно превышает величину ошибки на обучающей:

$$Q(\mu(X,Y),X^*,Y^*) >> Q(\mu(X,Y),X,Y).$$

Зачастую такую проблему можно исправить, если правильно подобрать параметры модели, которая использовалась для обучения.

Так как данная работа представляет собой решение задачи классификации, то остановимся поподробнее на соответствующих моделях машинного обучения.

1.3.2 Логистическая регрессия

Логистическая регрессия – один из самых простых и популярных алгоритмов для задачи классификации. Смысл данного метода заключается в том, чтобы построить разделяющую гиперплоскость в пространстве признаков, позволяющую отделить классы друг от друга.

В логистической регрессии строится линейный алгоритм классификации $a : X \to Y$, где X – пространство признаков, а Y – конечное множество номеров классов. Алгоритм имеет вид:

$$a(x,w) = sign(\sum_{j=1}^{n} w_j f_j(x) - w_0) = sign\langle x,w\rangle,$$

где w – вектор весов, w_0 – порог принятия решения, а $\langle\cdot,\cdot\rangle$ – скалярное произведение.

Задача обучения линейного классификатора заключается в том, чтобы по выборке настроить вектор весов. Для этого в логистической регрессии решается задача минимизации эмпирического риска с специальной функцией потерь вида:

$$Q(w) = \sum_{i=1}^{m} ln(1 + e^{-y_i \langle x_i, w \rangle}) \to \min_w$$

После нахождения решения w, становится возможным не только вычислять классификацию для произвольного объекта, но и оценивать апостериорные вероятности его принадлежности классам:

$$\mathbb{P}(y|x) = \sigma(y\langle x,w\rangle)$$

где $\sigma(z) = \frac{1}{1+e^{-z}}$ – сигмоидная функция.

1.3.3 Метод опорных векторов

Метод опорных векторов – также очень популярный метод машинного обучения, достаточно мощный и многогранный, применяемый в задачах классификации и регрессии.

Фундаментальная идея метода опорных векторов заключается в поиске гиперплоскости с наилучшим отступом – расстоянием между гиперплоскостью и опорными векторами – векторами, которые ближе всего находятся к разделяющей гиперплоскости.

Ищется решение задачи регрессии в линейном случае:

$$f(x) = \langle w,x\rangle - w_0.$$

Функция потерь принимает вид:

$$a(x_i) = |\langle w,x_i\rangle - w_0 - y_i|_\varepsilon$$

для каждого вектора (x_i,y_i).

В таком случае функционал потерь принимает вид:

$$Q_\varepsilon(a,X) = \sum_{i=1}^{l} |\langle w,x_i\rangle - w_0 - y_i|_\varepsilon + \tau\langle w,w\rangle^2 \to \min_{w,w_0}.$$

Последнее слагаемое удерживает коэффициенты w от бесконечного возрастания. Аналогично задаче классификации, решение зависит от скалярного произведения объектов, а не от самих объектов. Минимизация в данном случае эквивалентна задаче квадратичного программирования с ограничениями типа неравенств.

1.3.4 Градиентный бустинг

Градиентный бустинг — это метод машинного обучения, посвященный решению для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Обучение ансамбля проводится последовательно в отличие, например от бэггинга. На каждой итерации вычисляются отклонения предсказаний уже обученного ансамбля на обучающей выборке. Следующая модель, которая будет добавлена в ансамбль будет предсказывать эти отклонения. Таким образом, добавив предсказания нового дерева к предсказаниям обученного ансамбля мы можем уменьшить среднее отклонение модели, котрое является таргетом оптимизационной задачи. Новые деревья добавляются в ансамбль до тех пор, пока ошибка уменьшается, либо пока не выполняется одно из правил ранней остановки.

1.3.5 Случайный лес

Случайный лес – алгоритм машинного обчения, заключающийся в использовании ансамбля решающих деревьев. Алгоритм применяется как для задач классификации, так и для задач регрессии или кластеризации. Каждое решающее дерево само по себе дает не очень высокое качество классификации, но за счет их большого количества результат выходит хорошим.

Наиболее распространенный способ построения деревьев – бэггинг. Идея заключается в генерации случайной подвыборки с повторениями. Далее

строится решающее дерево, которое классифицирует образцы данной подвыборки, причем в ходе создания очередного узла дерева выбирается набор признаков, на основе которых производится разбиение, из которых выбирается наилучший(например, с использованием критерия Джини). Дерево строится до полного исчерпания подвыборки.

В задачах регрессии предсказывается значение путем усреднения ответов деревьев.

2 Классификация изображений датасета MNIST

2.1. Сравнительный анализ пакетов для вычисления устойчивых гомологий

В настоящее время существует ряд пакетов для вычисления устойчивых гомологий. Поэтому прежде чем приступать к задаче классификации, необходимо провести анализ и выбрать какой-то определенный пакет. Сравнительный анализ будет проводиться для пакетов с интерфейсом на Python, а конкретно Dionysus [17], Giotto-TDA [14], GUDHI [15], Ripser [19] – часть более обширной библиотеки Scikit-TDA [20] для топологического анализа данных.

Пакеты Dionysus и GUDHI существуют уже длительное время, и поэтому более надежны в работе, когда как Scikit-TDA и Giotto-TDA являются более новыми, но быстро развивающимися библиотеками. Сравнительный анализ этих пакетов представлен в таблице 2.1. Следующие алгоритмы используются для вычисления устойчивых гомологий:

- Стандартный алгоритм [12], заключающийся в приведении граничной матрицы фильтрации к ступенчатому виду.
- Twist algorithm [22], заключающийся в оптимизации стандартного алгоритма путем уменьшения размерности граничной матрицы фильтрации.
- Dual algorithm, заключающийся в вычислении устойчивых когомологий, что вычислительно более эффективно.

- Multifield algorithm [23], заключающийся в использовании других полей коэффециентов для более эффективного вычисления.
- Zigzag algorithm [24], также основанный на персистентных когомологиях, которые вычисляются более эффективно.

	Giotto-TDA	GUDHI	Ripser	Dionysus
Алгоритмы для вычисления устойчивых гомологий	standard, twist, dual	standard, dual, multifield	standard, twist, dual	standard, dual, zigzag
Коэффициенты	$\mathbb{F}_p$	$\mathbb{F}_p$	$\mathbb{F}_p$	$\mathbb{F}_p$ (dual); $\mathbb{F}_2$ (standard, zigzag)
Гомологии	Симплициальные, кубические	Симплициальные, кубические	Симплициальные	Симплициальные
Фильтрации	Виеторис-Рипс, Чех, α	α, Виеторис-Рипс, Чех, кубические, нерв, Witness, Делоне	Виеторис-Рипс	Виеторис-Рипс, α, Чех
Визуализация	Диаграммы устойчивости,	Диаграммы устойчивости, баркоды	Диаграмма устойчивости	Диаграммы устойчивости, баркоды

Таблица 2.1. Сравнительный анализ возможностей пакетов

Из таблицы видно, что библиотека GUDHI представляет максимально различные способы построения фильтраций, а также различные алгоритмы для вычисления устойчивых гомологий. Эта библиотека предоставляет также возможность вычислять кубические гомологии.

Проведем анализ пакетов на синтетических тестах. Для этого воспользуемся пакетов Tadasets [18], который, как и Ripser, является частью библиотеки Scikit-TDA [20], и предоставляет набор синтетических датасе-

тов, полезных для топологического анализа данных. Рассматриваемые синтетические датасеты – это не зашумленные облака точек в $\mathbb{R}^{26}$; количество точек $n = 2000$. Данные описывают следующие многообразия:

- Тор(расстояние от центра центра до центра внутренней окружности $c = 2$, радиус внутренней окружности $a = 1$).
- Швейцарский рулет(длина рулета $r = 4$).
- 2-Сфера(радиус $r = 3.14$).
- 3-Сфера(радиус $r = 3.14$).
- Букет 2 окружностей("знак бесконечности").

Во всех пакетах использовался стандартный алгоритм вычисления персистентных гомологий. Целью данного сравнения было нахождение пакета, который быстрее других вычисляет устойчивые гомологии, и для которого потребуются минимальные настройки. Все вычисления производились в среде Google Colab [21]. На основе данных сравнений можно сделать вывод, что библиотека GUDHI предоставляет наиболее эффективный способ вычисления устойчивых гомологий. В свою очередь, библиотека Dionysus тратила очень много времени на построение фильтрации, поэтому в вычислительном эксперименте ее результатов нет.

Таким образом, библиотека GUDHI представляет наибольшее разнообразие методов построения фильтраций, алгоритмов вычисления устойчивых гомологий, а также является эффективной с вычислительной точки зрения. Однако в дальнейшем будет использоваться Giotto-TDA, так как, несмотря на то, что в ряде тестов эта библиотека оказалась не самой эффективной с вычислительной точки зрения(например, на датасете "Швейцарский рулет" Giotto-TDA показала себя достаточно неэффективно, хотя все еще лучше, чем Ripser), эта библиотека предоставляет наиболее удобный интерфейс для использования, так как сам пакет сделан на основе

	Wall-time sec.				
	Тор	Швейцарский рулет	2-Сфера	3-Сфера	"Знак бесконечности"
Ripser	26.3	392	6.84	13.5	50.4
Gudhi	2.2	2.21	2.27	2.27	1.51
Giotto-TDA	14	264	4.14	7.04	29.1
	CPU time sec.				
	Тор	Швейцарский рулет	2-Сфера	3-Сфера	"Знак бесконечности"
Ripser	22.4	380	6.2	11.7	50.4
Gudhi	2.12	2.15	2.21	2.23	1.45
Giotto-TDA	13.8	263	4.07	6.72	28.9

Таблица 2.2. Тест скорости работы пакетов

пакета Scikit-Learn, одного из основных пакетов для машинного обучения на Python, а потому все методы имеют похожую структуру, какую имеют методы из пакета Scikit-Learn и других пакетов для научных вычислений на Python(например, NumPy).

2.2. Классификация изображений датасета MNIST

Теперь решим задачу классификации датасета MNIST. Датасет MNIST – это датасет, который состоит из 60000 изображений для обучения, а также 10000 изображений для тестирования, где каждое изображение является черно-белым изображением рукописной цифры от 0 до 9. Типичное изображение из данного датасета представлено на рис. 2.1. Данный датасет считается одним из стандартных датасетов для обучения различных методов распознавания изображений с помощью машинного обучения, причем в первую очередь для методов, основанных на нейронных сетях. Однако, в следствие нейросетевого бума, который случился в последнее десятилетие,

а в частности с появлением и развитием сверточных нейронных сетей, данный датасет слегка утратил свою актуальность – современные нейросети выдают почти 100-процентную точность, и для обучения нейросетей сейчас используют другие датасеты.

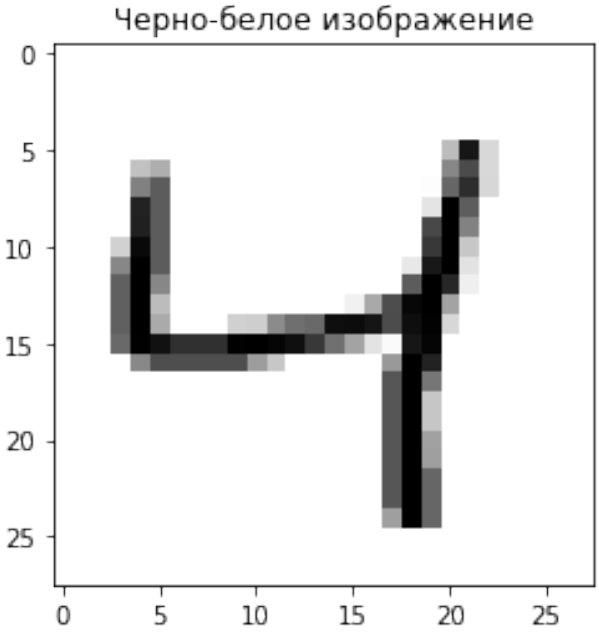


Рис. 2.1. Пример изображения из датасета MNIST

Однако при поиске новых подходов к классификации изображений все еще используют датасет MNIST. Поэтому в данной работе будет решена задачи классификации изображений именно с использованием данного датасета: топологический анализ данных, и в частности устойчивые гомологии – это новые инструменты для решения задачи классификации, в этой области все больше выходят современных исследований.

Опишем общий алгоритм 2, который лежит в основе решения задачи.

Алгоритм 2: Общий алгоритм решения задачи классификации

Исходные параметры: Набор изображений рукописных цифр

Результат: Алгоритм, определяющий рукописную цифру

для каждого *изображения из выборки* **выполнять**

Построить фильтрации;

Найти персистентные гомологии, построить диаграмму персистентости;

Получить топологические признаки из диаграммы персистентности;

конец

На наборе полученных признаков обучить модель машинного обучения;

Как видно из алгоритма, основным этапом является получение топологических признаков. На их основе и будет обучаться модель машинного обучения. Признаки получаются из диаграммы устойчивости различными методами, поэтому для одного изображения можно получить сразу несколько признаков по одной диаграмме. С другой стороны, персистентные гомологии, а значит и диаграмму устойчивости, можно считать для различных фильтраций одного и того же изображения. Таким образом на основе одного изображения можно сразу получить большое количество признаков, строя по ней различные фильтрации и различными способами векторизуя диаграмму. В данной работе использовался подход, изображенный на рис. 2.2

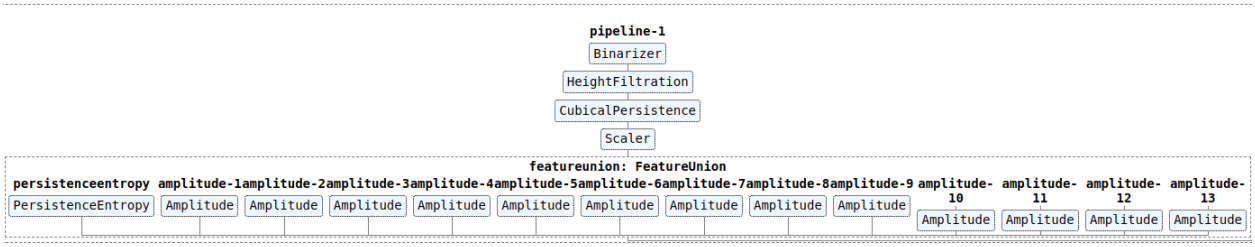


Рис. 2.2. Алгоритм получения признаков для Height Filtration

Опишем подробнее алгоритм получения признаков. Первым шагом является бинаризация изображения с заранее выбранным пороговым значением(в данной работе значение порога равнялось 0.4). Это нужно для того, чтобы далее воспользоваться специальными методами фильтрации,

которые работают именно с бинарным изображением. Бинарным изображением будет называть отображение $B : \mathbb{R}^d \to \{0, 1\}$.

Далее по бинарному изображению B строятся специальные фильтрации. По большому счету фильтрации можно воспринимать как трансформации бинарного изображения обратно в черно-белое, пропущенное через специальный фильтр. Так как в результате получается черно-белое изображение, то можно считать устойчивые гомологии сразу для самой картинки, но обычно построение черно-белого изображения через бинаризацию и фильтрацию наиболее сильно подчеркивает различные топологические особенности. В данном алгоритме использовалась т.н. фильтрация по высоте, радиальная фильтрация, а также фильтрация по плотности(рис. 2.3).

Фильтрация по высоте(Height filtration) – это отображение $H : I \to \mathbb{R}$, где I – это бинарное изображение в общем случае размерности d(в нашем случае размерность равна 2), которое каждому пикселю изображения p сопоставляет расстояние до гиперплоскости, которая определена вектором v длины 1, заданным заранее:

$$H : p \mapsto \begin{cases} \langle p,v \rangle, & \text{если } B(p) = 1, \\ H_\infty, & \text{иначе,} \end{cases}$$

где H_∞ – это значение фильтрации H самого дальнего пикселя изображения до гиперплоскости.

Радиальная фильтрация(Radial filtration) – это отображение $R : I \to \mathbb{R}$, которое каждому пикселю изображения сопоставляет расстояние до выбранного заранее центра c:

$$R : p \mapsto \begin{cases} \|c - p\|_2, & \text{если } B(p) = 1, \\ R_\infty, & \text{иначе,} \end{cases}$$

где R_∞ – это расстояние самого дальнего пикселя до центра. То есть фильтрация строится исходя из значения некоторой радиальной функции от пикселя p, у которого $B(p) = 1$.

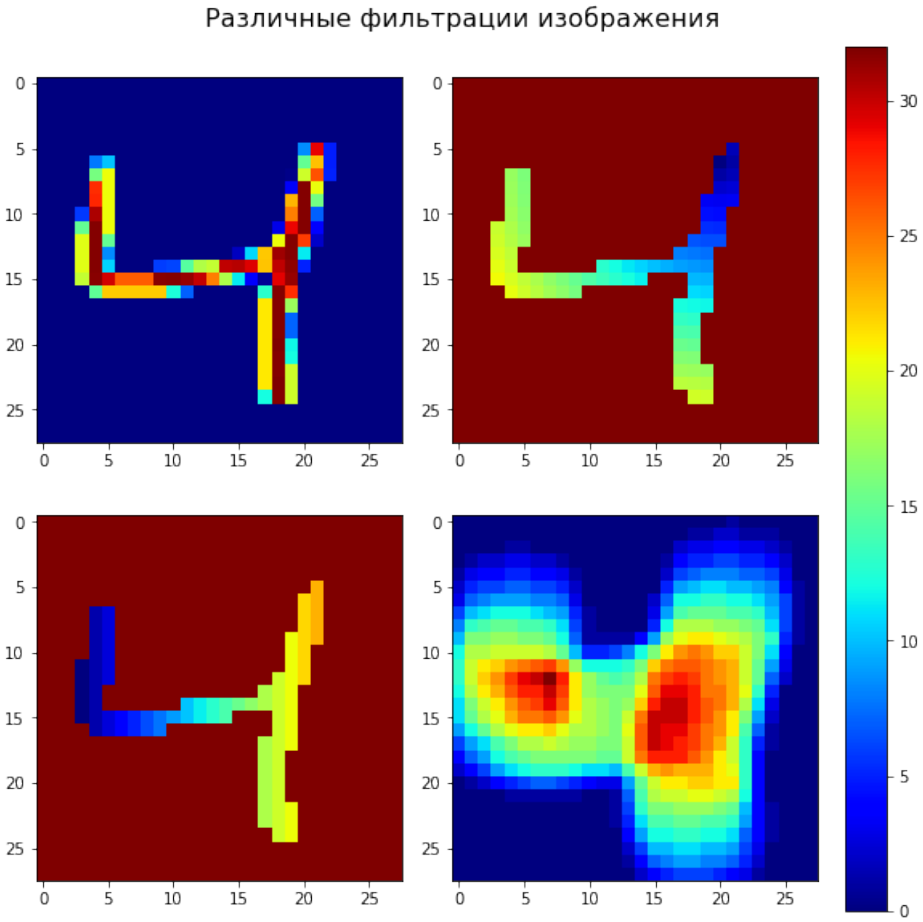


Рис. 2.3. Черно-белые изображения, полученные с помощью различных фильтраций. Для наглядности, была использована цветная гамма для представления черно-белых значений. В верхнем левом углу – оригинальное изображение; в верхней правой – радиальная фильтрация. В нижнем левой углу – фильтрация по высоте; в нижнем правом углу – фильтрация по плотности

Фильтрация по плотности(Density filtration) – это отображение $D : I \to \mathbb{R}$, которое каждому пикселю изображения сопоставляет число пикселей p, таких, что $B(p) = 1$, находящихся на растоянии не больше задаваемого r:

$$D_r(p) := |\{v \in I | B(v) = 1 \wedge \|p - v\| \leq r\}|,$$

где $\|\cdot\|$ – любая норма в $\mathbb{R}^2$. В данном алгоритме использовалась стандартная евклидова метрика.

Так как вектор направления для фильтрации по высоте и центр для радиальной фильтрации задается заранее, то можно формировать различные фильтрации при различных значениях вектора направления и центра. В данном алгоритме фильтрация по высоте строилась с 8 различными значениями для вектора направления, радиальная фильтрация строилась с 9 различными значениями для центра, а фильтрация по плотности строилась с 3 различными значениями для радиуса. Таким образом, для одного изображения получалось 20 фильтрации.

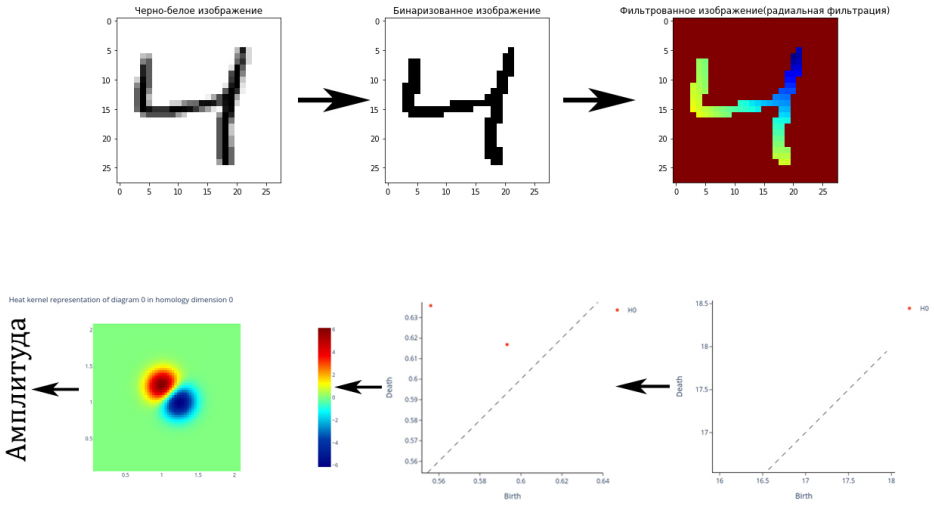
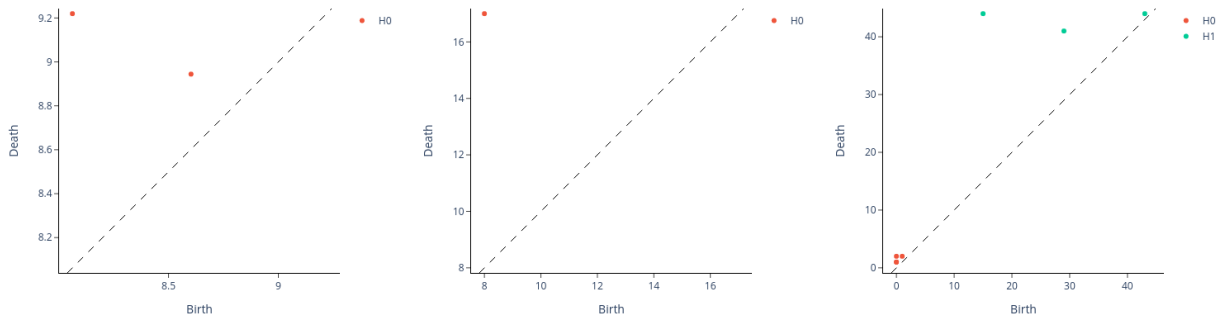


Рис. 2.4. Пример работы алгоритма

Далее для каждой из полученных фильтраций вычислялись устойчивые гомологии 0 и 1 размерности, а по ним строились диаграммы перси-

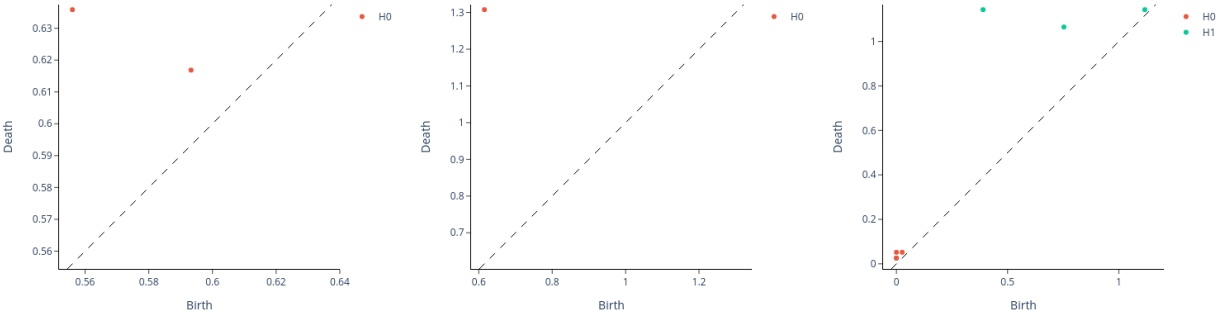
стентности, которые далее были приведены к одному и тому же масштабу.



(a) Radial Filtration (b) Height Filtration (c) Density Filtration

Рис. 2.5. Диаграммы устойчивости для различных фильтраций

На рис. 2.4 представлен пример работы алгоритма для одного бинарного изображения из датасета. На рис. 2.5 представлены диаграммы устойчивости для различных фильтраций. На рис. 2.6 представлены диаграммы устойчивости, приведенные к одному и тому же масштабу. Видно, что изменились значения шкал рождения/смерти у диаграмм.

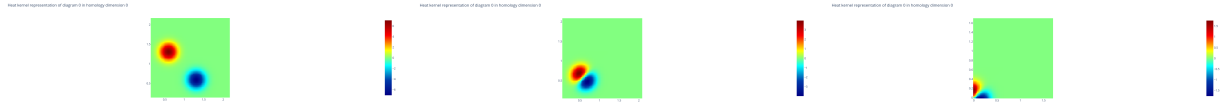


(a) Radial Filtration (b) Height Filtration (c) Density Filtration

Рис. 2.6. Диаграммы устойчивости для различных фильтраций, приведенные к одному масштабу

В свою же очередь, для каждой диаграммы вычислялись 14 признаков: персистентная энтропия и 13 амплитуд для различных векторных представлений. На рис. 2.7 представлен пример таких представлений для различных фильтраций. Таким образом, для одной картинки получалось

$20 \times 2 \times 14 = 560$ признаков. Однако вероятно, что некоторые из таких признаков будут коррелировать, а поэтому необходимо будет произвести отбор признаков.



(a) Height filtration (b) Radial filtration (c) Density filtration

Рис. 2.7. Heat Kernel для различных фильтраций для одного и того же изображения

После получения векторного представления для изображений из выборки, дальнейшим этапом является обучение модели машинного обучения. В данной работе использовались следующие модели:

- Метод опорных векторов
- Случайный лес
- Логистическая регрессия
- LightGBM
- CatBoost
- XGBoost

Первым шагом было запустить эти модели без настраивания параметров и отбора признаков, и посмотреть, как хорошо они справляются с задачей, т.е. таким образом получить начальное значение для каждой из модели, чтобы в дальнейшем значения качества моделей увеличивать как за счет настраивания параметров модели, так и за счет отбора признаков. Полученшиеся результаты отображены в табл. 2.3

Несмотря на очевидное переобучение – значения на тренировочной выборке куда лучше, чем на тестовой – все модели проявили себя очень

Название модели	Значение на тренировочной выборке	Значение на тестовой выборке
Логистическая регрессия	0.989	0.903
Метод опорных векторов	1.0	0.893
Случайный лес	1.0	0.89
LightGBM	1.0	0.9
XGBoost	1.0	0.883
CatBoost	1.0	0.893

Таблица 2.3. Значения базовых моделей на тренировочной и тестовой выборках

хорошо и показали неплохие результаты(тут стоит отметить, что современные сверточные нейронные сети достигают 99.95% точности). Попробуем для каждой модели сделать поиск по сетке, чтобы улучшить результаты моделей. Результат поиска по сетке представлен в табл. 2.4. Можно заметить, что для некоторых моделей проблема переобучения исчезла в результате подбора параметров.

Название модели	Значение на тренировочной выборке	Значение на тестовой выборке
Логистическая регрессия	0.911	0.917
Метод опорных векторов	0.903	0.893
Случайный лес	0.88	0.87
LightGBM	1.0	0.9
XGBoost	1.0	0.883
CatBoost	1.0	0.893

Таблица 2.4. Значения наилучших моделей, полученных в результате подбора параметров поиском по сетке, на тренировочной и тестовой выборках

Как было сказано ранее, скорее всего многие признаки коррелируют между собой. Поэтому дальнейший шаг – это отбор признаков. Сперва убедимся, что уменьшение признаков может привести к росту качества модели. Для этого запустим случайный лес на выборке. Случайный лес

может быть использован в качестве отбора признаков – есть возможность подсчитать важность признаков для этой модели. Те признаки, которые оказались наиболее важны для случайного леса, вероятно будут и наиболее важными и для других моделей. Поэтому с помощью случайного леса упорядочим все признаки по убыванию их важности. Далее в цикле будем запускать модели на выборке, но с разным числом признаков – так как они упорядочены, то будем брать первые $50n$ признаков, где $1 \leq n \leq 11$. Запускать модели будем с поиском по сетке, чтобы использовать наилучшие параметры для данной выборке. График зависимости качества построенных моделей от числа признаков представлен на рис. 2.8.

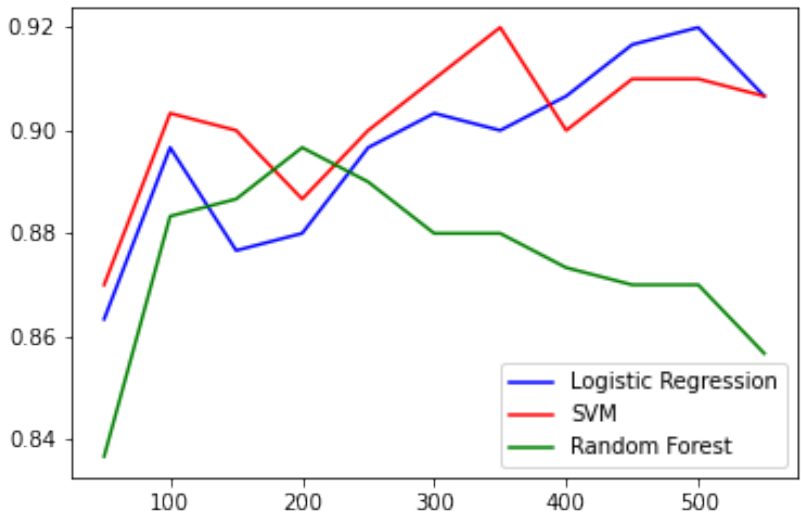


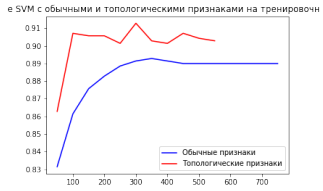
Рис. 2.8. График зависимости качества построенных моделей от числа признаков

На графике видно, что, уменьшив количество признаков, все модели улучшают свои показатели качества.

Далее можно сравнить качество моделей, которые используют топологические признаки, с качеством моделей, которые в качестве признаков будут получать просто векторное представление изображения. То есть для таких моделей признаками будут сами значения пикселей изображения.

Так как модели с топологическими признаками показали наилучший результат именно при уменьшении количества признаков, то также отсортируем признаки для новых моделей по убыванию важности для случайного леса, и будем брать первые $50n$ признаков, каждый раз обучая модель с поиском по сетке, чтобы подобрать наилучшие параметры. Так как теперь признаков будет $28 \times 28 = 784$, то $1 \leq n \leq 15$. На графике 2.9 представлены сравнения различных моделей, но с разными признаками.

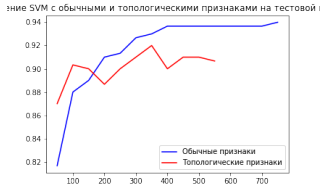
(a) Логистическая регрессия на тренировочной выборке



(b) Метод опорных векторов на тренировочной выборке

(c) Случайный лес на тренировочной выборке

(d) Логистическая регрессия на тестовой выборке



(e) Метод опорных векторов на тестовой выборке

(f) Случайный лес на тестовой выборке

Рис. 2.9. Сравнение различных моделей, которые используют различные признаки

Видно, что модели с топологическими признаками при малом количестве признаков достигает лучшей точности. Отсюда можно сделать вывод, что топологические признаки содержат в себе наиболее важную, релевантную информацию о структуре данных в меньшем числе признаков. Поэтому такой подход можно рассматривать как подход для уменьшения размерности данных.

Видно, что метод опорных векторов с подобранными параметрами

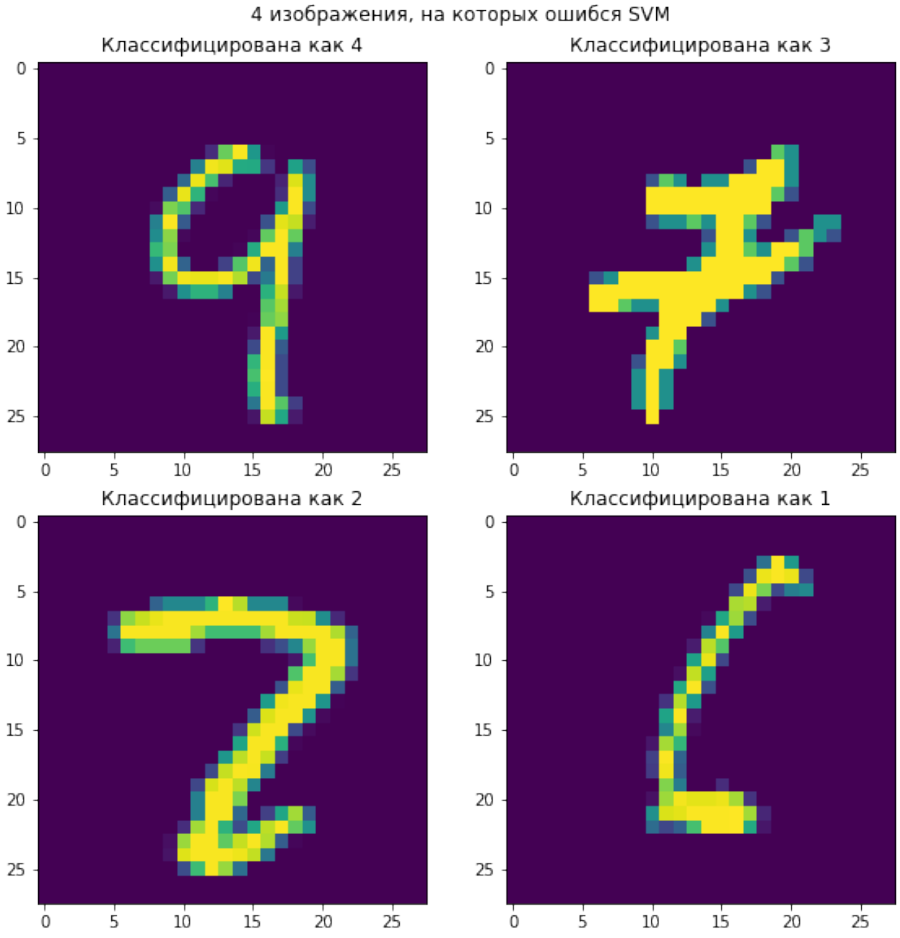


Рис. 2.10. Первые несколько изображений, на которых ошибся классификатор

на 350 лучших признаках, отобранных с помощью случайного леса, показывает чуть ли не самый высокий результат. Интересно посмотреть, на каких изображений эта модель ошиблась. На рис. 2.10 представлены как раз такие 4 изображения. Видно, что цифры, на которых ошибся классификатор, действительно похожи на те, которые классификатор предписал данным изображениях.

Заключение

полстраницы В ходе данной

Литература

[1] Виро О.Я., Иванов О.А., Нецветаев Н.Ю., Харламов В.М. (2018). Элементарная топология. 978-5-4439-2680-3.

[2] Вик Дж. У. (2005). Теория гомологий. Введение в алгебраическую топологию. 5-94057-086-0.

[3] Хатчер А. (2011). Алгебраическая топология. 978-5-94057-748-5.

[4] Munkres, James R. (2000). Topology. 0-13-181629-2.

[5] Фоменко А.Т. (2001). Математика и миф сквозь призму геометрии. URL: `http://dfgm.math.msu.su/files/fomenko/myth-vved.php`

[6] Edelsbrunner, H., Letscher, D., and Zomorodian, A. (2002). Topological persistence and simplification. Discrete Comput. Geom., 28:511–533.

[7] Zomorodian, A. and Carlsson, G. (2005). Computing persistent homology. Discrete Comput. Geom., 33(2):249–274.

[8] Carlsson, G. (2009). Topology and data. AMS Bulletin, 46(2):255–308.

[9] Brittany T. Fasy, Jisu Kim, Fabrizio Lecci, Clement Maria, David L. Millman, and Vincent Rouvreau. Introduction to the R package TDA. URL: `https://cran.r-project.org/web/packages/TDA/vignettes/article.pdf`

[10] F. Chazal An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists / F.Chazal, B. Michel. URL : https://arxiv.org/pdf/1710.04019.pdf

[11] H. Edelsbrunner Computational Topology An Introduction / H. Edelsbrunner, J. Harer ; AMS : Providence, 2009. – 241 c.

[12] A. Zomorodian Computing persistent homology / A. Zomorodian, G. Carlsson // Discrete Comput. Geom. – 2005. – Vol. 33, № 2. – P. 249 – 274. URL : https://geometry.stanford.edu/papers/zc-cph-05/zc-cph-05.pdf

[13] N. Otter A roadmap for the computation of persistent homology / N. Otter, M.A. Porter, U. Tillmann, P. Grindrod, H. A Harrington // EPJ Data Sci. – 2017. – Vol. 6, №17. URL : https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-017-0109-5

[14] Giotto-tda – библиотека для топологического анализа данных на Python. – URL : https://github.com/giotto-ai/giotto-tda

[15] GUDHI – библиотека для топологического анализа данных на C++ с интерфейсом для Python. – URL : https://gudhi.inria.fr/

[16] F. Chazal The Structure and Stability of Persistence Modules / F. Chazal, V. de Silva, M. Glisse, S. Outdot. URL : https://arxiv.org/pdf/1207.3674.pdf

[17] Dionysus 2 – библиотека для вычислений устойчивых гомологий, написанная на C++ и имеющая интерфейс на Python. URL : https://www.mrzv.org/software/dionysus/

[18] Tadasets – библиотека, содержащая синтетические датасеты для топологического анализа данных. URL : https://github.com/scikit-tda/tadasets

[19] Интерфейс для Ripser на Python. URL : https://ripser.scikit-tda.org/en/latest/

[20] Scikit-TDA – библиотека для топологического анализа данных, содержащая Ripser и Tadasets. URL : https://scikit-tda.org/

[21] Google Colab – Сервис Google, предоставляющий возможность запускать код, написанный на Python, в браузере, обладающий интерфейсом Jupyter Notebook, специально созданный для задач машинного обучения, анализа данных, а также образования. URL : https://colab.research.google.com/

[22] C. Chen Persistent Homology Computation with a Twist / C. Chen, M. Kerber // EuroCG – 2011. URL : https://eurocg11.inf.ethz.ch/abstracts/22.pdf

[23] J. Boissonnat Computing Persistent Homology with Various Coefficient Fields in a Single Pass / J.Boissonnat, C. Maria. URL : https://arxiv.org/abs/2001.02960.

[24] C. Maria Computing Persistent Cohomology / C. Maria, S. Oudot. URL: https://arxiv.org/abs/1608.06039.