Санкт-Петербургский государственный университет

Математико-механический факультет

Погорелов Петр Глебович

Научно-исследовательская работа

Ортогонализация набора данных с учетом применяемой решающей функции

Научный руководитель:
д. ф.-м. н., профессор Новиков Б.А.
Научный консультант:
к. ф.-м. н., доцент Кураленок И.Е.

Санкт-Петербург
2018

Ортогонализация Набора Данных с Учетом Применяемой Решающей Функции

Погорелов П.Г.[1]

Санкт-Петербургский Государственный Университет, Санкт-Петербург, Россия
peter.pogorelov@gmail.com

Аннотация В рамках настоящего исследования представлен результат работы над новым алгоритмом ортогонализации данных. Особенность данного алгоритма заключается в том, что в процессе декорреляции используется произвольная решающая функция. Это позволяет сформировать новый набор данных таким образом, что он оптимально подходит к конкретной модели классификации либо регрессии.

Keywords: машинное обучение · декорреляция набора данных · ортогонализация набора данных.

1 Введение

В процессе моделирования некоторой непрерывной $y \in R$ либо дискретной $y \in L = \{l_1, l_2, ..., l_k\}$ величины на основе реального (не синтетического) набора данных $X = (x_1, x_2, ...x_m)$, где x_i может быть как вещественным, так и бинарным фактором, очень часто можно наблюдать так называемую проблему мультиколлинеарности (или эндогенности факторов). Проблема мультиколлинеарности сводится к наличию линейной связи между величинами, которая выражается в существовании их линейной комбинации (т.е., один фактор можно выразить через набор других). В большинстве математических моделей присутствие линейных комбинаций факторов приводит к проблемам, связанным с статистическими характеристиками оценок их коэффициентов: эффективности, состоятельности, несмещенности. В частности, если аналитически выразить коэффициенты линейной регрессии $\hat{b} = (X^T X)^{-1} X^T y$, то можно обратить внимание, что элементы матрицы $(X^T X)^{-1}$ устремятся в бесконечность, если в данных будет присутствовать мультиколлинеарность. Это продиктовано тем, что ранг квадратной матрицы $X^T X$ будет меньше числа ее столбцов, что, следовательно выразится в равенстве определителя $det(X^T X)$, либо его стремлении к 0.

В рамках теории информации, проблему мультиколлинеарности можно интерпретировать как наличие взаимной информации в отдельных факторах друг о друге. Соответственно — один из подходов к устранению мультиколлинеарности — ортогонализация (или декорреляция) набора данных.

Для решения задачи ортогонализации данных существует множество подходов, которые можно разделить на две крупные категории, соответственно, линейные и нелинейные методы декорреляции. Также можно отдельно упомянуть класс методов декорреляции данных, называемый whitening-трансформация. Её суть заключается в том, чтобы сделать ковариационную матрицу декоррелированных факторов в новом базисе — единичной.

Цель настоящей работы — разработка алгоритма, позволяющего выполнить ортогонализацию (декорреляцию) набора данных с использованием произвольной решающей функции. Для достижения этой цели были поставлены задачи:

1. выполнить обзор наиболее популярных линейных и нелинейных алгоритмов ортогонализации данных,
2. формализовать новый алгоритм ортогонализации набора данных,
3. формализовать задачи, решаемые в рамках исследования, для которых может потребоваться выполнить ортогонализацию данных,
4. сравнить на разных наборах данных результат работы данных алгоритмов на произвольных моделях регрессии, классификации,
5. оценить его качество на произвольных моделях классификации.

В рамках последующих разделов представленного исследования будут рассмотрены все поставленные задачи, а в заключительной части сформулированы выводы касательно итогового результата работы.

Обзор существующих подходов

В настоящий момент существует множество алгоритмов ортогонализации (декорреляции) данных успешно применяемых в различных прикладных задачах, которые можно разделить на две крупные категории. Первая категория — линейные алгоритмы, вторая, соответственно — алгоритмы нелинейные. В первой категории особенно хочется выделить:

1. ZCA-Whitening
2. Principal Component Analysis
3. Independent Component Analysis [1]

Во второй категории наибольший интерес представляют следующие методы:

1. Isomap [2]
2. Multi-Dimensional Scaling [3]
3. Locally Linear Embedding [4]
4. t-SNE [5]

В следующих подразделах будет детально рассмотрена часть из представленных выше алгоритмов.

1.1 Principal Component Analysis

Метод главных компонент (PCA) можно назвать одним из важнейших прикладных инструментов математического моделирования, использующих аппарат линейной алгебры. Благодаря своей простоте и интуитивной интерпретации, метод главных компонент пользуется широкой популярностью специалистами различных областей, в круг задач которых входит аналитика над данными. Среди таких областей можно выделить: нейрофизиологию, биоинформатику, компьютерную графику, физику и др.

Данный непараметрический подход к декорреляции используется для извлечения полезной информации в зашумленных наборах данных. Благодаря тому факту, что метод главных компонент практически не содержит гиперпараметров (кроме количества выделяемых компонент), даже человек без математического образования может провести редукцию размерности комплексоного пространства данных с целью построения визуализации, подготовки входных параметров к некоторой математической модели, выявления скрытых закономерностей в данных.

Особенность метода главных компонент заклчается в подходе к выделению такого нового базиса матрицы факторов, что факторы в данном базисе становятся ортогональными (линейно независимыми). То есть, будет иметь место нулевая корреляция Пирсона между ними. Еще одна интересная особенность PCA заключается в том, что выделенные компоненты также будут отранжированы по степени выриации данных вдоль их проекции на компоненту. Иными словами — компоненты будут упорядочены по степени значимости (вклада) каждой из компонент в имеющийся набор данных.

Далее представлен алгоритм построения нового базиса для матрицы факторов, который используется в методе главных компонент,

1. сначала рассчитывается ковариационная матрица по набору данных,
2. для ковариационной матрицы рассчитываются соответствующие ей собственные вектора и собственные значения,
3. выполняется сортировка собственных векторов в порядке убывания соответсвующих им собственных чисел,
4. первые k собственных векторов рассматриваются как новый базис, выделяющий k независимых факторов,
5. с использованием линейного преобразования, построенного на предыдущем шаге, оригинальные данные преобразуются в k-мерные точки пространства нового базиса.

Таким образом, можно обозначить наиболее важные задачи, решаемые в рамках инструменратия, предлагаемого аппаратом метода главных компонент. Соответственно, первая - поиск линейно независимых подпространств внутри многомерного облака данных, на которые можно спроецировать данные с минимальными потерями. Вторая - предоставление возможности выполнить реконструкцию оригинальных измерений, то есть, опять же, потери должны быть минимальны. Суть метода заключается в том, чтобы итеративно находить такие проекции внутри облака данных, что вариация вдоль проекции будет максимальна, а проекции - ортогональны друг другу.

Формализовать метод главных компонент можно следующим образом. Пусть имеется некоторой набор данных $X \in \mathbb{R}^m$, где каждый из факторов является либо непрерывной случайной величиной $x_i \in R$, либо бинарной случайной величиной $x_i \in \{0, 1\}$. Ковариация случайного вектора x рассчитывается по формуле,

$$Cov(x_i, x_j) = \frac{1}{n} \sum_{t=1}^{n} (x_i(t) - E[x_i])(x_j(t) - E[x_j]).$$

В случае, если факторы ноль-центрированы ($E[x] = (0, 0, ..., 0)$), то представленное выше выражение можно упростить до следующей векторно - матричной формы: $Cov(X) = X^T X$. Далее, необходимо найти такое линейное преобразование $\hat{X} = PX$ что $Cov(\hat{X}) = \hat{\sum} = I$. То есть, ковариационная матрица набора данных из исходного пространства (X) в новом базисе P ($\hat{X} = PX$) становится единичной. Ковариацию набора данных в новом базисе можно выразить следующим образом, $\hat{\sum} = \hat{X}^T \hat{X} = (PX)^T (PX) = P^T X^T X P = P^T \sum P$. Очевидно, что если взять за P матрицу, построенную из собственных векторов исходной ковариационной матрицы $\sum$, то $\sum P = \Lambda P$, где Λ - диагональная матрица, на главной диагонали которой лежат собственные числа исходной ковариационной матрицы. Данное выражение напрямую вытекает из свойств собственных чисел и собственных векторов произвольной матрицы. То есть, если взять квадратную матрицу A для которой V - матрица собственных векторов, λ - вектор собственных чисел, то будет справедливо выражение выражение $AV = \lambda IV$. Таким образом, становится очевидной справедливость выражения $\hat{\sum} = P^T P \Lambda = I\Lambda$, в результате которого ковариационная матрица в новом базисе стала диагональной, и, следовательно, была выполнена декорреляция факторов, устранены линейные зависимости между ними. Теперь, если выполнить деление каждой из получившихся в результате рассмотренного преобразования компонент на квадратный корень соответствующего ей собственного числа, то данные будут сферизованы (ковариационная матрица станет единичной).

1.2 Independent Component Analysis

Метод независимых компонент (ICA) - еще один из линейных подходов к ортогонализации. Его принципиальное отличие от «классического» метода главных компонент заключается в том, в процессе «обучения» данного алгоритма в облаке данных ищутся такие проекции, где мера отличия от нормального распределения — максимальна. Метод анализа независимых компонент базируется на технике Projection Pursuit (гонка за проекцией) [6], суть которой которой заключается в том, чтобы найти некоторые «интересные» проекции в многомерном пространстве данных.

Формально, ICA можно рассматривать как частный случай реализации алгоритма Projection Pursuit. ICA позволяет (в прочем, наряду с PCA) выделять независимые компоненты (s) на основе имеющегося набора данных

(x) такие, что что $|s| \leq |p|$. То есть, число новых выделенных компонент может быть как меньше изначального числа представленных факторов, так и быть равным ему. Исходя из предположения, что некоторые измерения исходного пространства данных, не входящих в описываемое выделенными независимыми компонентами подпространство, заполнено белым шумом (gaussian noise), можно сформировать предпосылку о необходимости поиска такие проекции внутри исходного пространства, распределение данных вдоль которых максимально отличается от нормального. Соответственно, после того как все «не-гауссовские» проекции были найдены, можно считать, что все независимые компоненты были выделены.

Для того, чтобы интегрировать понятие «не-гауссовости» в метод анализа независимых компонент, необходимо формализовать некоторую меру, позволяющую определить, насколько сильно некоторая случайная величина y отклоняется от нормального распределения. Для простоты предлагается ввести предположение о ноль-центрировании y ($E[y] = 0$) и о её единичной дисперсии ($\sigma_y = 1$).

В качестве классической меры «не-гауссовости» часто рассматривают величину эмпирического коэффициента эксцесса (нормализованная версия момента четвертого порядка $E[y^4]$), который можно выразить следующим образом

$$kurt(y) = E[y^4] - 3(E[y^2])^2.$$

В случае, если величина y распределена нормально, коэффициент эксцесса будет равен нулю, в то время как для большинсва случайных величин из других распределений (не гауссовых) величина данного коэффициена будет отлична от нуля.

Еще одна важная мера «не-гауссовости» задается с помощью негэнтропии (negentropy). Для начала необходимо дать определение понятия энтропия. Энтропия представляет собой один из базовых концептов теории информации и может быть интерпретирована как объем информации, который может содержаться в случайной величине. Чем более «случайна» (непредсказуема) величина, тем выше её энтропия. Более формально, энтропия представляет собой оптимальную длинну строки, которая необходима, чтобы закодировать значения, принимаемые случайной величиной. В дискретном случае энтропию можно выразить следующим образом,

$$H(Y) = -\sum_i P_y(y_i) log(P_y(y_i)).$$

В случае, если величина y принадлежит к некоторому непрерывному распределению, то величину энтропии можно задать следующим образом,

$$H(y) = -\int f(y) log(f(y)) dy.$$

Пользуясь тем фактом, что случайная величина, порожденная нормальным распределением, обладает самой высокой энтропией среди всех случайных величин с тем же значением дисперсии, можно формализовать меру

«не-гауссовости» как значение отклонения энтропии некоторой случайной величины от энтропии нормального распределения. Эту меру принято называть негэнтропией, которая задается следующим образом,

$$J(y) = H_{gauss} - H(y).$$

Поскольку рассчет настоящего значения негэнтропии — весьма трудоемкая задача в смысле затрачиваемых вычислительных ресурсов, принято использовать различные апроксимации. В частности, классический метод, разработанный Jones с соавторами (1987) использует аппроксимацию негэнтропии с использованием моментов высших порядков,

$$J(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48} kurt\{y\}^2.$$

Важно отметить, что данную аппроксимацию нельзя назвать робастной в отношении коэффициента эксцесса, что затрудняет её применение в контексте обучения ICA модели.

Авторы публикации [1] предлагают использовать в качестве аппроксимации негэнтропии следующую функциональную форму, которая содержит некоторую функцию G, которая будет формализована далее

$$J(y) \propto (E[G(y)] - E[G(v)])^2,$$

где v представляет собой ноль-центрированную случайную величину из гауссовского распределения с единичной дисперсией (стандартизированную). Аналогичным образом подразумевается, что величина y тоже должна быть ноль-центрирована и иметь дисперсию равную единице. Очевидно, что $J(\cdot)$ представляет собой обобщение аппроксимации негэнтропии, базирующееся на использовании моментов, которая была рассмотрена ранее. Если взять такую функциональную форму G, которая не вызовет слишком резкий рост аппроксимированной величины негэнтропии J, то оценка негэнтропии будет считаться относительно устойчивой (робастой). В частности, авторы публикации предлагают использовать следующие функциональные формы G,

$$G_1(u) = \frac{1}{a_1} \log \cosh_{a1}(u), G_2(u) = -exp(-u^2/2),$$

где $1 \leq a_1 \leq 2$ - некоторая константа.

Один из самых популярных подходов для оценки параметров ICA модели — использование метода максимального правдоподобия, который имеет тесную связь с принципом infomax. Формализовать данный подход можно следующим образом. Пусть $W = (w_1, w_2, ..., w_k)^T = A^{-1}$, где A — линейное ортогональное преобразование, выделяющее независимые компоненты из набора данных $X = (x_1, x_2, ..., x_m)$. Лог-правдоподобие будет иметь вид,

$$L = \sum_{i=1}^{N} \sum_{j=1}^{k} \log f_j(w_j^T x(i)) + T \log(det\ W),$$

где f_i — плотность распределения независимой компоненты s_i (предполагается, что плотность известна), а $x(i)$ — наблюдения случайной величины x. Правый терм выражения, $log|detW|$ — напрямую следует из правила линейного преобразования случайных величин и их плотностей. Известно, что для любого случайного вектора x с плотностью p_x и для любой матрицы W плотность $y = Wx$ выражается как $p_x(Wx)|detW|$.

Далее предлагается формализовать алгоритм FastICA, который на текущий момент является самой популярной реализацией метода ICA для выделения независимых компонент в наборе данных. В своей базовой форме алгоритм можно выразить следующим образом,

1. выполняется предварительная инициализация вектора весов w, например, значения могут быть сгенерированы из стандартизированного нормального распределения,
2. далее рассчитывается «обновление» вектора весов $w^+ = E[xg(w^Tx)] - E[g'(w^Tx)]w$,
3. затем выполняется перерасчет вектора весов $w = w^+/\|w^+\|$,
4. шаги 2, 3 повторяются до тех пор, пока выражение не сошлось, то есть, пока $\|w\|_2 > \epsilon$, где ϵ - некоторая константа, например, $1e^-10$.

Далее предлагается рассмотреть вывод выражения для w^+. Пользуясь следствием теоремы Кунна-Таккера, можно утверждать что оптимальное значение для $E[G(w^Tx)]$ при заданном ограничении $E[(w^Tx)^2] = \|w\|^2 = 1$ достигается в точке, где выполняется следующее условие,

$$E[xg(w^Tx)] - \beta w = 0.$$

Данное выражение можно попытаться оптимизировать с помощью метода Ньютона. В частности, якобиан функции, используемый в контексте ограничения выше можно выразить следующим образом,

$$JF(w) = E[xx^Tg'(w^Tx)] - \beta I.$$

Далее, исходя из предположения, что данные были предварительно сферизованы (была применена некоторая whitening-трансформация, например, PCA, либо ZCA), можно ввести следующую аппроксимацию,

$$E[xx^Tg'(w^Tx)] \approx E[xx^T]E[g'(w^Tx)] = E[g'(w^Tx)]I.$$

Таким образом, аппроксимированная итерация метода Ньютона для оценки коэффициенов вектора w примет следующий вид,

$$w^+ = w - [E[xg(w^Tx)] - \beta w]/[E[g'(w^Tx)] - \beta].$$

1.3 Locally Linear Embedding

Локально-линейный эмбеддинг (Locality Linear Embedding или LLE) — представляет собой метод для поиска координат в пространстве с низкой

размерностью для набора данных, лежащего на некотором нелинейном многообразии, построенном на высоко-размерном пространстве. Идея метода заключается в том, чтобы индивидуально, для каждой отдельной точки, выполнять редукцию размерности. Предпосылка к использованию данного подхода заключается в том, что по утверждениям автора метода, любое нелинейное многообразие имеет локально-линейную структуру, аналогично интерпретации обыкновернной производной гладкой функции. Далее, после того, как размерность каждой из точек была редуцирована, авторы предлагают расположить точки в новом пространстве таким образом, чтобы некоторая мера «несоответствия» их локальной структуры была минимальна.

Сама процедура LLE преобразования выполняется в три шага. Сначала определяются ближайшие соседи для каждой из точек в представленном наборе данных. Затем рассчитываются веса для их линейной аппроксимации посредством их соседей. Заключительный этап алгоритма — поиск низкоразмерных координат, которые реконструируются полученными на прошлом шаге весами наиболее эффективным образом.

Формализовать алгоритм LLE можно следующим образом. Пусть имеется матрица $X = (x_1, x_2, ..., x_m)$, где каждый из m факторов описывается n наблюдениями. Задается размерность итогового низкоразмерного подпространства $q < m$ и число ближайших соседей k такое, что $k \geq q + 1$. Результатом работы алгоритма будет матрица $\hat{X} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_q,)$. Механизм работы LLE преобразования формально разбивается на следующие шаги,

1. для каждой точки X_i рассчитываются k ближайших соседей,
2. Рассчитывается матрица весов W, которая минимизирует остаточную сумму квадратов для реконструкции точки X_i по её соседям

$$RSS(W) = \sum_{i=1}^{n} \|X_i - \sum_{j \neq i} W_{ij} X_j\|^2,$$

   где $W_{ij} = 0$ для тех X_j, которые не являются соседними для X_i точками. Помимо всего прочего, для каждого i должно выполняться $\sum_j W_{ij} = 1$,
3. рассчитываются координаты $\hat{X}$, минимизирующие ошибку реконструкции точек X в низкоуровневом пространстве с использованием матрицы весов W, полученной на предыдущем шаге,

$$\Phi(\hat{X}) = \sum_{i=1}^{n} \|\hat{X}_i - \sum_{i \neq j} W_{ij} \hat{X}_j\|^2,$$

   где $\sum_i \hat{X}_{ij} = 0$ для каждой j, а $\hat{X}^T \hat{X} = I$.

1.4 t-Distributed Stochastic Neighbor Embedding

t-SNE представляет собой алгоритм нелинейного сжатия размерности, широко используемый в задачах исследования структуры многомерных данных. В большинстве случаев, данный алгоритм используется для проекции

многомерных данных на две (и более) оси координат с целью упрощения визуального анализа данных. На практике, преобразованные при помощи t-SNE факторы не используется в качестве входных данных для статистических моделей и моделей машинного обучения в силу высокой вычислительной сложности данного алгоритма. Фактически, сложность зависит от реализации, но во всех случаях она — полиномиальная в зависимости от числа наблюдений и количества факторов.

Для начала можно сформулировать некоторые предпосылки к использованию t-SNE, которые выделяют его на фоне линейных методов сжатия размерности данных, в том числе, метода главных компонент (PCA) и анализа независимых компонент (ICA). Следствием того, что PCA и ICA представляют собой линейные алгоритмы декорреляции, является их неспособность к интерпретации комплексных полиномиальных отношений между факторами. С другой стороны, t-SNE — базируется на вероятностных распределениях со случайным блужданием по графам ближайших соседей, с помощью чего данный алгоритм проводит анализ структуры набора данных, и, теоретически, способен выделять более сложные взаимосвязи, нежели линейные.

Самая главная проблема, связанная с применением линейных алгоритмов сжатия размерности заключается в том, что их механизм работы сводится к расположению «наименее близких» друг к другу точек как можно дальше друг от друга в пространстве, размерность которого меньше исходного. Однако следует обратить внимание, что для того, чтобы представить проекцию набора данных из пространства высокой размерности в нелинейном многообразии низкой размерности, необходимо сфокусироваться на анализе схожести точек, выступающих друг для друга ближайшими соседями. Линейные алгоритмы сжатия размерности пространства, фактически, не способны на это.

Особенность t-SNE в сравнении с другими алгоритмами сжатия нелинейных многобразий высокой размерности (в т.ч. LLE) заключается в том, что данный алгоритм способен восстанавливать не только локальную, но и глобальную структуру данных. Подходы к восстановлению локальной структуры ищут наиболее близкие друг к другу точки на многообразии и размещают их близко к друг другу в пространстве мешньшей размерности. Подходы к восстановлению глобальной структуры напротив, пытаются сохранить глобальную геометрию пространства на всех машстабах, то есть, располагают близкие друг к другу на многообразии точки близко друг к другу, а находящиеся далеко друг от друга — как можно дальше друг от друга.

Далее будут рассмотрены детали работы алгоритма t-SNE. Для начала, необходимо рассмотреть алгоритм-прародитель t-SNE — SNE или Stochastic Neighbor Embedding. Досновно его название можно перевести как "стохастический метод построения сжатого представления данных с использованием метода ближайших соседей". Данный алгоритм начинает свою работу с конвертации евклидовых расстояний между точками набора данных в условные вероятности, которые можно интерпретировать как меру схожести. Мера

схожести точки x_i и точки x_j выражается в виде условной вероятности $p(x_i|x_j)$ (далее используется обозначение $p_{i|j}$). Точка x_i «интерпретирует» точку x_j как своего соседа, если ее отклонение от x_i, которое выступает в качестве математического ожидания некоторой гауссианы, входит в допустимые рамки, ограниченные некоторой пропорцией (задающейся числом ближайших соседей).

Для точек, которые располагаются сравнительно близко друг к другу, величина $p_{i|j}$ — относительно велика, в то время как для точек, которые находятся далеко друг от друга - величина $p_{i|j}$ — стремится к нулю. Математически величину $p_{i|j}$ можно выразить следующим образом,

$$p_{i|j} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)},$$

где σ_i — среднеквадратичное отклонение гауссианы, центрированной в точке x_i.

Введем обозначение y, которое можно интерпретировать как проекцию точки x в пространстве меньшей размерности. В таком случае, можно ввести аналог вероятности $p_{i|j}$ и обозначить его как $q_{i|j}$,

$$q_{i|j} = \frac{exp(-\|y_i - yx_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2/2\sigma_i^2)}.$$

Проще говоря, величина $p_{i|j}$ отражает близость точек в пространстве высокой размерности, а $q_{i|j}$ — в пространстве низкой размерности. Интуитивно, для того, чтобы добиться близости точек в пространствах разной размерности, данные условные вероятности должны быть равны друг другу. Алгоритм SNE как раз базируется на этих предпосылках, пытаясь минимизировать разницу между условными вероятностями.

В алгоритме t-SNE для минимизации разницы между условными вероятностями используется мера из теории информации — дивергенция Кульбака-Лейблера (Kullback—Leibler Divergence). Она рассчитывается по формуле

$$KL(p\|q) = H(p) - H(p,q) = \sum_i p_i \log q_i.$$

Соответственно, в алгоритме t-SNE минимизируется следующий функционал,

$$KL(p\|q) = \sum_{i \neq j} \log \frac{p_{i|j}}{q_{i|j}} \to min_y.$$

Следует отметить, что t-SNE, как и LLE не подразумевают введение некоторого нелинейного преобразования на основе существующего набора данных, при помощи которого можно выполнить ортогонализацию факторов. Ортогонализация выполняется только по текущим данным, и в случае введения новой точки (наблюдения), все рассчеты потребуется производить заново.

Задачи классификации и регрессии

Задачи классификации и регрессии — одни из самых распространенных проблем, решаемых с помощью аппарата прикладной статистики либо, как на текущий момент принято выражаться, «машинного обучения». Задача классификации представляет собой подраздел группы методов машинного обучения с учителем, суть которой заключается в том, чтобы «обучить» алгоритм выполнять автоматическое определение категории объекта, к которой он относится с наиболее высокой вероятностью. Обучение выполняется с использованием размеченной выборки, то есть, некоторому объекту $x \in R^m$ ставится в соответствие бинарная $y \in B$, либо дискретная категория (метка) $y \in L$, где $L = \{l_1, l_2, ..., l_k\}$ — множество некоторых дискретных величин (категорий), а $B = \{0, 1\}$ — множество бинарных величин, принимающих только два значения. Соответственно, цель «обучения» — поиск некоторой функциональной формы $f \in F$ и оценка ее коэффицинтов с целью минимизации шанса совершить ошибку классификации $f(x_i) \neq y_i$. Алгоритмические подходы решения проблемы классификации можно выделить в две большие группы — модели бинарной классификации и методы множественной классификации. В дальнейших подразделах данной главы будет представлен их краткий обзор. Задача линейной регрессии аналогична проблеме классификации во многих аспектах, отличие заключается в том, что моделируемая величина — непрерывна ($y \in R$), в остальном постановка проблемы ничем принципиально не отличается от случая классификации.

1.5 Задача классификации

Задача классификации — одна из наиболее популярных проблем, решаемых в контексте машинного обучения с учителем. Самый распространенный вид классификации — бинарная проблема, где моделируемая величина принимает только два уникальных значения значения (1 и 0, «ложь» и «истина», «да» и «нет» и так далее). Формализовать бинарную классификацию можно следующим образом. Пусть имеется некоторый набор данных (чаще всего употребляется термин «обучающая выборка») $X = \{x_i\}_{i=1}^n$, где n — объем выборки. В соответствие каждому индивидуальному объекту x_i поставлена метка $y \in \{0, 1\}$. Также предполагается, что каждый объект x_i, каждая метка y_i и каждое наблюдение (x_i, y_i) независимо сгенерированы из неизвестного распределения данных $P(x, y)$. Предположим, что существует некоторый генеративный процесс $P(y|x)$, который можно однозначно выразить в следующем виде,

$$y = f(x) + \epsilon,$$

где ϵ — некоторая случайная компонента (ошибка), а $f(\cdot)$ — функциональная зависимость между объектом выборки и соответствующей ему меткой. Соответственно, задача машинного обучения — подобрать функциональную форму $\hat{f} \in F$ аппроксимирующую реальную функциональную зависимость f с минимальной ошибкой.

При работе с моделями машинного обучения, предназначенными для категоризации наблюдений выборки только на два подмножества, возникает потребность в использовании некоторого мета-алгоритма, позволяющего расширить функциональность исходных моделей для включения дополнительного набора классов. На текущий момент существует множество подобных мета-алгоритмов, наиболее популярные из них разобраны в работе Daniely A. [7] с соавторами.

One versus Rest или OvR (один против всех) — наиболее очевидный способ реализации "мультиклассовости". Производится обучение m бинарных классификаторов f_i (где $i \in \{1, 2, ..., m\}$) разделяющих выборку между одним из классов $l_i \in L = \{l_1, l_2, ..., l_m\}$ и оставшимися $m - 1$ классами. На этапе предсказания, для каждого документа x выбирается такой класс $\hat{l_x}$, что степень уверенности соответствующей модели $f_{\hat{l_x}}$ — максимальна,

$$\hat{l_x} = argmax_{l \in L} f_l(x).$$

Под степенью уверенности классификатора можно понимать разные величины. Например, если речь идет о машине опорных векторов, то это – расстояние от классифицируемого объекта до разделяющей гиперплоскости. В логистической регрессии за данную величину можно принять предсказанную вероятность. В рамках данного исследования все модели бинарной классификации, функционал которых необходимо экстраполировать на множество классов, используют мета-алгоритм «один против всех».

1.6 Задача регрессии

Регрессионный анализ — это широкая область статистического моделирования, занимающаяся исследованием взаимосвязей между случайными величинами. Одна из задач, решаемая в рамках регрессионного анализа — моделирование так называемой целевой величины. То есть, подразумевается, что одна из величин фиксируется как целевая, а остальные рассматриваются как независимые (регрессоры, факторы), с помощью которых можно предсказать (смоделировать) целевую. Если удалось однозначно установить, что связь между целевой величиной и экзогенными факторами — линейная, то задача сводится к решению проблемы линейной регрессии. В таком случае имеет место следующая функциональная зависимость между целевой величиной и факторами,

$$y_i = \beta_0 + \sum_{j=1}^{m} \beta_j x_{i,j} + \epsilon,$$

где y_i — i-е наблюдение моделируемой величины, $x_{i,j}$ — j-я компонента i-го наблюдения экзогенной величины, β_j — некоторая константа, величину которой необходимо установить в рамках решения задачи регрессии, а ϵ — случайный шум (и не учтенные факторы). Напротив, если величины в модели связаны между собой не линейно, то могут потребоваться дополнительные

исследования, требующие глубокого понимания предметной области, чтобы установить эту связь.

Алгоритм подбора оптимального подмножества данных

1.7 Цель и задача алгоритма

Пусть имеется некоторое множество независимых случайных величин $X = \{x_i\}_m$, принадлежащих к разным вероятностным распределениям. Требуется сформировать упорядоченное подмножество $\tilde{X} = \{x_j \parallel h(x_j) > h(x_{j+1})\}_m$, отсортированное по величине кросс-энтропии между переменной x_j и ее предсказанным значением $\hat{x}_j$, полученным с помощью решающей функции $f \in F$, где F — множество решающих функций. Функция $h(\cdot) \in H$ используется для ранжирования величин, H — множество всех возможных подходов для ранжирования. В контексте данного исследования подразумевается использование кросс-энтропии между действительным значением величины и ее предсказанным значением в качестве меры ранжирования.

1.8 Описание алгоритма

Рассмотрим постановку задачи регрессии в контексте данного исследования. Под проблемой регрессии подразмевается моделирование условномого математического ожидания некоторой случайной величины (в дальнейшем предлагается использовать обозначение y) от набора других случайных величин, которые принято называть факторами (в дальнейшем предлагается использовать обозначение $X = (x_1, x_2, ...x_m)$. То есть, требуется установить некоторую функциональную форму $f(X)$, такую что $f(X) = E(y|X)$, где $f \in F$ некоторая функция-регрессор из множества решающих функций.

В рамках исследования предлагается свести задачу регрессии к задаче множественной классификации. Это продиктовано тем, что рассчет величины кросс-энтропии между непрерывными величинами требует применения некоторых численных методов для оценки плотности переменных, между которыми рассчитывается данная мера. В частности, сложность рассчета ядерного сглаживания составляет $O(kn)$ [8], в то время как рассчет эмпирического распределения дискретной случайной величины - $O(n)$. Так же важно отметить, что дихтомизация непрерывных факторов устраняет такие проблемы выборки, как выбросы. Для решения задачи дихтомизации был выбран метод разбиения по медиане, описанный в работе D. Rucker с соавторами [9]. В данной работе также рассматриваются как положительные, так и негативные эффекты, которые процесс дихтомизации выборки оказывает на качество классификации.

В результате дихтомизации данных множества непрерывных случайных величин $X_R = \{x_{Ri}\}_{m_R}$, формируется новое множество $X_D = \{x_{di}\}_{m_d}$, такое, что $x_{di} \in \{0, 1\}$, а $m_R < m_d$. В рамках данного исследования предлагается производить дихтомизацию величин методом бисекции по медиане на 32 части. Таким образом, число новых бинарных величин m_d будет равно $32m_R$. Код, выполняющий данную процедуру представлен в листинге на языке программирования Python. В данном случае переменная x_bins представляет собой массив новых, дихтомизированных величин из набора X.

```
x_bins = [np.sort(X)]
for i in range(int(iterations)):
    new_bins = list()
    for j in x_bins:
        index = j.shape[0] // 2
        new_bins += [j[:index], j[index:]]
    x_bins = new_bins
```

Далее к рассмотрению предлагается описание алгоритма подбора оптимального упорядоченного множества факторов. Пусть имеется набор вещественных факторов $X = (x_1, x_2, ..., x_m)$ и его дихтомизированное представление $X_D = (x_{D1}, x_{D2}, ..., x_{Dm})$, где $x_{Di} = (x_{di1}, x_{di2}, ..., x_{di32})$.

Algorithm 1 Поиск упорядоченного подмножества факторов

Data: $\tilde{X} \leftarrow \{\varnothing\}, I \leftarrow \{\varnothing\}$
Result: Упорядоченное по значимости множество индексов факторов (I); упорядоченное по значимости множество ортогонализованных факторов ($\tilde{X}$)
$\tilde{X} \leftarrow \tilde{X} \cup x_{Dr}; I \leftarrow \{r\}; \quad 1 \le r \le m$
while $\|\tilde{X}\| \le m$ **do**
  $h \leftarrow \{\varnothing\}$
  $\hat{X} \leftarrow \{\varnothing\}$
  for $x_D \in X_D \setminus \tilde{X}$ **do**
    $f \leftarrow arg\max_{f \in F} L_F(x_D|\tilde{X})$
    $\hat{x_D} \leftarrow x_D \oplus f(\tilde{X})$
    $h \leftarrow h \cup H(\hat{x_D}, x_D)$
    $\hat{X} \leftarrow \hat{X} \cup \hat{x_D}$
  end
  $\tilde{X} \leftarrow \tilde{X} \cup \hat{X}[arg\max h]$
  $I \leftarrow I \cup \{arg\max h\}$
end

Далее предлагается подробно рассмотреть представленный выше алгоритм. На этапе инициализации формируется пустое упорядоченное множество ортогонализованных факторов $\tilde{X} \leftarrow \{\varnothing\}$, куда добавляется один какой-нибудь случайный фактор x_{Dr}, $1 \le r \le m$. Аналогичным образом формируется пустое упорядоченное множество индексов, куда заносится индекс

r. Далее, до тех пор, пока выполняется условие $\|\tilde{X}\| \neq \|X_D\|$ ($\|\tilde{X}\| \leq m$), реализуются следующие шаги,

1. выполняется оценка коэффициентов решающей функции последовательно для каждого их элементов $x_D \in X_D$ на основе данных $\tilde{X}$, то есть, подбирается функциональная форма $\hat{f} \approx E[x_D|\tilde{X}]$. Рассчитывается величина $\hat{x_D}$, представляющая собой значение операции исключающего или между действительным значением величины x_D и ее оценкой $\hat{f}(\tilde{X})$ при заданном $\tilde{X}$,
2. рассчитывается величина кросс-энтропии $H(\cdot,\cdot)$ между результатом операции XOR и действительным значением, $h' = H(\hat{x_D}, x_D)$.
3. в множество $\tilde{X}$ добавляется величина $\hat{x_D}$ с максимальной величиной рассчитанной кросс-энтропии, а соответствующий ей индекс заносится в множество I.

Аналогичным образом, множество $\tilde{X}$ рассчитывается для всех начальных $r \in \{1, 2, ..., m\}$. Обозначим множество $\tilde{X}$, построенное на некоторой начальной r как $\tilde{X}_r$. Тогда, в результате последовательного выполнения алгоритма для всех уникальных значений параметра r будет сформировано множество $\tilde{X}_R = \{\tilde{X}_r\}_m$, элементы которого - различные подмножества ортогонализированных факторов. Таким же образом формируется множество $I_R = \{I_r\}_m$, элементами которого являются отранжированные по значимости индексы факторов, в зависимости от того какой фактор был выбран изначальным.

Для того, чтобы задать некоторое "глобальное"ранжирование для $\tilde{X}_R$ и I_R можно воспользоваться следующим алгоритмом.

Algorithm 2 Алгоритм ранжирования факторов по значимости

Data: $I_R = \{I_r\}_m, S = 0_m$
Result: Упорядоченное множество, у которого индекс элемента соответствует индексу фактора, а значение элемента - степени значимости фактора в наборе данных.

```
for I_r in I_R do
    n_features ← len(I_r)
    for i in I_r do
        S[i] ← n_features
        n_features ← n_features − 1
    end
end
```

Сравнение существующих подходов к ортогонализации данных в задаче регрессии

В рамках исследования было произвдеено сравнение ранее рассмотренных подходов для декорреляции данных на различных наборах данных с разными регрессионными моделями. В качестве метрики используется среднеквадратичная ошибка (MSE), рассчитываемая по следующей формуле:

$$MSE(y, \hat{y}) = \frac{1}{|y|}(y - \hat{y})^T(y - \hat{y}).$$

Было принято решение использовать следующие широко применяемые виды решающих функций в рамках тестирования алгоритмов ортогонализации данных,

1. Linear Regression (обычная регрессия, оцениваемая методом МНК),
2. Support Vector Regression (машина опорных векторов),
3. Random Forest Regression (случайный лес).

Перед рассчетом статистики для машины опорных векторов и случайного леса проводилась предварительная оптимизация гиперпараметров с использованием алгоритмов hyperopt и bayesian-optimization. Подробнее с их использованием можно ознакомиться на официальных страницах данных проектов. Для моделей линейной регрессии и машины опорных векторов были взяты реализации из пакета scikit-learn, соответственно, sklearn.linear. LinearRegression и sklearn.svm.SVR. Следует отметить, что в рамках программной реализации данных методов, по умолчанию, используются ралзичные методы регуляризации, которые были отключены для чистоты эксперимента. В качестве модели случайного леса использовалась реализация из пакета lightgbm.

Статистика представлена в виде кросс-валидации (по 8 уникальным разбиениям) по тренировочному и тестовому наборам и в виде one-fold валидации по отдельному, валидационному набору, который не пересекается с тренировочным и тестовым. Идея метода кросс-валидации заключается в том, чтобы итеративно разбивать имеющийся набор данных на два непересекающихся множества, где первое множество соответствует тренировочной выборке, а второе - тестовой, в соотношении 5 к 1. Далее выполняется «обучение» регрессионной модели на обучающием множестве данных и её валидация на тестовом множестве. Процедура выполняется ровно то число раз, которое соответсвует гиперпараметру алгоритма - количеству разбиений.

Для тестов использовались три различных набора данных, в том числе: Facebook Comment Volume Dataset, Parkinsons Telemonitoring Data Set, Energy Efficiency Data Set. Данные наборы данных достаточно популярны в рамках задач, которые включают в себя сравнение моделей машинного обучения. Это продиктовано тем, что эти наборы содержат разнообразные виды факторов (категриальные, бинарные, ординальные, а так же непрерывные), и их число сильно отличается. Помимо этого, следует отметить, что выборки очень сильно отличаются по объемам наблюдений.

В рамках исследования, для сравнения были отобраны следующие подходы к декорреляции данных, которые были подробно рассмотрены в предыдущих разделах работы,

1. Principal Component Analysis,
2. Independent Component Analysis,
3. t-Distributed Stochastic Neighbor Embedding,
4. Locality Linear Embedding.

Программные реализации данных алгоритмов были взяты в пакете scikit-learn. Важно отметить, что каждый из представленных алгоритмов подразумевает установку определенного, фиксированного числа независимых компонент. Из практических соображений было решено ограничиться 10-ю компонентами для PCA, ICA, LLE подходов на наборах данных Facebook Comment Volume Dataset, Parkinsons Telemonitoring Data Set. Напротив, в контексте набора данных Energy Efficency Data Set было решено ограничиться только четырьмя компонентами в связи с тем, что в этой выборке каждое наблюдение описывается только семью компонентами. Так же следует отметить, что по причине высокой вычислительной сложности алгоритма t-SNE, было решено ограничиться выделением только трех независимых компонент в каждом из наборов данных. Данное конкретное число - верхняя граница числа компонент для реализации t-SNE Барнса-Хута.

1.9 Кросс-валидация

Facebook Comment Volume Dataset

Regressor	Raw	PCA(10)	ICA(10)	T-SNE(3)	LLE(10)
LR	795.21	907.66	889.89	1218.02	1342.09
SVR	1728.86	185357.73	**1051.16**	**1407.01**	**1351.36**
RFR	772.62	910.07	901.54	1177.28	1278.63

Parkinsons Telemonitoring Data Set

Regressor	Raw	PCA(10)	ICA(10)	T-SNE(3)	LLE(10)
LR	6.81	83.11	86.59	105.74	113.98
SVR	17.60	94.73	88.57	288.82	114.03
RFR	8.00	8.42	21.56	93.43	86.16

Energy Efficiency Data Set

Regressor	Raw	PCA(4)	ICA(4)	T-SNE(3)	LLE(4)
LR	6.09	121.46	94.44	99.63	110.37
SVR	9.04	125.52	99.68	175.71	110.45
RFR	7.48	120.11	21.22	83.83	87.24

1.10 Оценка на валидационном наборе

Facebook Comment Volume Dataset

Regressor	Raw	PCA(10)	ICA(10)	T-SNE(3)	LLE(10)
LR	914.25	1464.96	1196.91	1623.08	**485.26**
SVR	275536.77	**95049.85**	**1338.91**	**1643.11**	**486.21**
RFR	977.35	1473.19	1153.20	1545.89	**469.00**

Parkinsons Telemonitoring Data Set

Regressor	Raw	PCA(10)	ICA(10)	T-SNE(3)	LLE(10)
LR	6.09	121.46	94.44	99.63	110.37
SVR	9.04	125.52	99.68	175.71	110.45
RFR	7.48	120.11	21.22	83.83	87.24

Energy Efficiency Data Set

Regressor	Raw	PCA(4)	ICA(4)	T-SNE(3)	LLE(4)
LR	7.74	103.87	97.03	65.73	78.76
SVR	17.12	152.16	519.39	103.26	77.51
RFR	1.34	107.50	109.79	14.29	34.39

Жирным шрифтом отмечены те значения метрики MSE, которые показали некоторый прирост, полученный при использовании методов ортогонализации относительно необработанных данных. Не трудно заметить, что в большинстве случаев использование независимых компонент вместо оригинального набора данных не дало какого либо прироста. Исключительная ситуация - с набором Facebook Comment Volume Dataset, где большинство используемых методов кроме PCA показало небольшой прирост в задаче построения машины опорных векторов.

Сравнение существующих подходов к ортогонализации данных в задаче классификации

В данном разделе представлено сравнение ранее рассмотренных методов ортогонализации данных на различных наборах данных с разными решающими функцями. Принципиальное отличие от предыдущего раздела заключается в том, что задача регрессии была сведена к задаче классификации путем дихтомизации объясняемой величины и факторов методом бисекции по медиане на 32 части. Соответственно, в результате дихтомизации все непрерывные величины были сконвертированы в дискретные, принимающие 32 уникальных значения. В качестве метрики для оценки качества классификации предлагается использовать точность (accuracy) в связи с тем, что ее достаточно легко интерпретировать. Точность рассчитывается по следующей формуле:

$$Accuracy(y, \hat{y}) = \frac{\|\{1|y_i = \hat{y_i}\}\|}{\|y\|},$$

где выражение $\|\{1|y_i = \hat{y_i}\}\|$ можно понимать как TP (True Positive) - число верно классифицированных объектов. Поскольку решается задача множественной классификации, где моделируемая величина принимает 32 уникальных значения ($y \in \{l_1, l_2, l_3, ...l_{32}\}$), необходимо формализовать некоторый подход для экстраполяции моделей бинарной классификации на случай нескольких классов. В рамках исследования предлагается использовать подход OvR (one versus rest), который заключается в построении m уникальных классификаторов для m отдельных классов. Каждый i-й классификатор решает такую бинарную проблему классификации, где $\tilde{y}_j = 1\ iif\ y_i = l_j\ else\ 0$. Некоторому объекту x ставится в соответствие класс l_i, если $\tilde{y}_i = 1$. В рамках эксперимента были использованы следующие модели классификации:

1. Logistic Regression (логистическая регрессия),
2. Support Vector Classifier (машина опорных векторов),
3. Random Forest Classifier (классификатор случайного леса).

Из используемых методов ортогонализации были исключены t-SNE, LLE. Это продиктовано тем, что после дихтомизации пространство факторов возрастает до $m * 32$ новых факторов. Проблема заключается в том, что данные алгоритмы крайне чувствительны к количеству используемых факторов, что чрезвычайно замедляет скорость их расчета и предъявляет высокие требования к объему оперативной памяти. В остальном, конфигурация эксперимента не отличается от той, что применялась при решении задачи построения регрессии.

1.11 Кросс-валидация

Facebook Comment Volume Dataset

Regressor	Raw	PCA(10)	ICA(10)
LR	0.688	0.551	0.551
SVC	0.688	0.551	0.652
RFC	0.681	0.551	0.660

Parkinsons Telemonitoring Data Set

Regressor	Raw	PCA(10)	ICA(10)
LR	0.857	0.502	0.478
SVC	0.859	0.505	0.505
RFC	0.885	0.598	0.673

Energy Efficiency Data Set

Regressor	Raw	PCA(4)	ICA(4)
LR	0.743	0.730	0.539
SVC	0.742	0.733	0.722
RFC	0.759	**0.774**	**0.761**

1.12 Оценка на валидационном наборе

Facebook Comment Volume Dataset

Regressor	Raw	PCA(10)	ICA(10)
LR	0.680	0.550	0.553
SVC	0.680	0.550	0.649
RFC	0.674	0.550	0.661

Parkinsons Telemonitoring Data Set

Regressor	Raw	PCA(10)	ICA(10)
LR	0.841	0.464	0.464
SVC	0.846	0.469	0.488
RFC	0.877	0.591	0.656

Energy Efficiency Data Set

Regressor	Raw	PCA(4)	ICA(4)
LR	0.727	0.714	0.571
SVC	0.740	0.714	**0.766**
RFC	0.727	**0.766**	**0.792**

Жирным шрифтом отмечены те значения метрики *Accuracy*, которые показали некоторый прирост относительно необработанных данных благодаря использованию методов ортогонализации. В большинстве случаев, использование независимых компонент не отразилось положительно на результатах классификации. Однако стоит отметить, что некоторый прирост стабильно наблюдался в наборе данных Energy Efficiency Data Set для модели случайного леса. В этом случае и PCA и ICA показали стабильный прирост точности как на кросс-валидации так и на one-fold валидации с использованием обособленного валидационного набора данных.

Апробация алгоритма подбора оптимального подмножества данных

В данном разделе представлены результаты апробации алгоритма подбора оптимального подмножества на ранее рассматриваемых наборах данных, на которых проводилось тестирование алгоритмов ортогонализации, в том числе:

1. Facebook Comment Volume Dataset,
2. Parkinsons Telemonitoring Data Set,
3. Energy Efficiency Data Set.

Тестирование подхода реализовывалось на задаче регрессии, сведенной к задаче классификации путем дихтомизации объясняемой величины модели и факторов методом бисекции по медиане на 32 части. Для статистики были взяты следующие модели классификации:

1. Logistic Regression (логистическая регрессия),
2. Support Vector Classifier (машина опорных векторов),
3. Random Forest Classifier (случайный лес).

В качестве метрики качества, как и в предыдущем разделе использовалась точность (accuracy), которая рассчитывается по следующей формуле: $Accuracy(y, \hat{y}) = \frac{\|\{1|y_i=\hat{y_i}\}\|}{\|y\|}$, где выражение $\|\{1|y_i = \hat{y_i}\}\|$ можно интерпретировать как TP (True Positive) - число верно классифицированных объектов.

Далее приведена статистика, полученная с использованием кросс - валидации по 8 уникальным подразбиениям с усредненным значением метрики

Accuracy, где индекс набора данных соответствует его порядковому номеру в списке, который можно найти в начале данного раздела.

Regressor	$dataset_1$	$dataset_2$	$dataset_3$
LR	0.677	0.828	0.760
SVC	0.678	0.832	0.759
RFC	0.572	0.409	0.542

Можно обратить внимание, что в сравнении с подходами PCA и ICA, ортогонализация набора данных положительно сказалать на точности предсказаний моделей логистической регрессии и машины опорных векторов. При этом, напротив, точность классификации очень сильно падает в классификаторе случайного леса. Соответственно, основываясь на имеющихся данных, можно сделать предположение о том, что данный алгоритм эффективен на множестве линейных моделей классификации и показывает низкую точность на других подходах к классификации. Данное предположение будет верифицировано или опровергнуто в рамках последующих исследований в данной области.

Заключение

В рамках настоящего исследования, проведенного в целях разработки алгоритма, позволяющего выполнить ортогонализацию (декорреляцию) набора данных с использованием произвольной решающей функции, был выполнен анализ современных подходов к решению задачи ортогонализации данных, а так же их сравнение в рамках задачи регрессии и классификации. Для анализа использовались следущие наборы данных, широко применяемые в экспериментах, связанных с оценкой качества различных моделей машинного обучения,

1. Facebook Comment Volume Dataset,
2. Parkinsons Telemonitoring Data Set,
3. Energy Efficiency Data Set.

Для построения регрессии были задействованы следующие подходы,

1. Linear Regression (обычная регрессия, оцениваемая методом МНК),
2. Support Vector Regression (машина опорных векторов),
3. Random Forest Regression (случайный лес).

Для решения задачи классификации были использованы модели,

1. Logistic Regression (логистическая регрессия),
2. Support Vector Classifier (машина опорных векторов),
3. Random Forest Classifier (случайный лес).

Сравнительный анализ показал, что в большинстве случаев ни один из использовавшихся алгоритмов ортогонализации данных не продемонстрировал прироста релевантной данной задаче метрики в сравнении с исходным, сырым набором данных.

Был предложен новый алгоритм для декорреляции данных и проведена его апробация на аналогичных выборках данных в рамках решения задачи классификации. Результаты показали, что новый подход эффективно выделяется на фоне остальных алгоритмов при использовании линейных моделей классификации, но испытывает деградацию в точности классификации в рамках решения поставленной задачи с использованием метода случайного леса. Было выдвинуто предположение о том, что алгоритм эффективен только при его использовании в паре с линейными моделями. В ходе дальнейших исследований планируется проверить данную гипотезу, а так же продолжить работу над доработкой нового метода декорреляции набора данных.

Список литературы

1. Hyvarinen Aapo. Fast and robust fixed-point algorithms for independent component analysis // IEEE transactions on Neural Networks. 1999. Т. 10, № 3. С. 626–634.
2. Tenenbaum Joshua B, De Silva Vin, Langford John C. A global geometric framework for nonlinear dimensionality reduction // science. 2000. Т. 290, № 5500. С. 2319–2323.
3. Kruskal Joseph B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis // Psychometrika. 1964. Т. 29, № 1. С. 1–27.
4. Roweis Sam T, Saul Lawrence K. Nonlinear dimensionality reduction by locally linear embedding // science. 2000. Т. 290, № 5500. С. 2323–2326.
5. Maaten Laurens van der, Hinton Geoffrey. Visualizing data using t-SNE // Journal of machine learning research. 2008. Т. 9, № Nov. С. 2579–2605.
6. Friedman Jerome H, Tukey John W. A projection pursuit algorithm for exploratory data analysis // IEEE Transactions on computers. 1974. Т. 100, № 9. С. 881–890.
7. Daniely Amit, Sabato Sivan, Shwartz Shai S. Multiclass learning approaches: A theoretical comparison with implications // Advances in Neural Information Processing Systems. 2012. С. 485–493.
8. Raykar Vikas C, Duraiswami Ramani, Zhao Linda H. Fast computation of kernel estimators // Journal of Computational and Graphical Statistics. 2010. Т. 19, № 1. С. 205–220.
9. On the practice of dichotomization of quantitative variables. / Robert C MacCallum, Shaobo Zhang, Kristopher J Preacher [и др.] // Psychological methods. 2002. Т. 7, № 1. с. 19.