Сравнение методов оценки внутренней размерности данных (C-PCA и FisherS)

Авторы: Матвеев Артем Сергеевич
г. Москва

Аннотация – Современные данные почти всегда являются многомерными, поэтому встает задача снижения размерности этих данных для оптимизации вычислений и уменьшения объемов хранимых данных. Большая часть методов понижения размерности на входе заранее требуют размерность пространства, в которое мы отображаем эти данные. Такую размерность будем называть внутренней. В данной работе сравниваются два метода оценки этой внутренней размерности и показывается, что FisherS превосходит C-PCA как по скорости работы, так и по точности на обычных и зашумленных данных. Сравнения производятся на синтетических данных, с добавлением шума и без него.

Ключевые слова – внутренняя размерность, PCA, разделимость по Фишеру, минимальное покрытие множества.

I. ВВЕДЕНИЕ

Данное сочинение построено следующим образом: в разделе «I. Введение» описывается, как построена данная работа, в разделе «II. Определение внутренней размерности» дается строгая формализация данного понятия, в разделе «III. Обзор метода C-PCA» обозревается алгоритм оценки внутренней размерности, основанный на построении минимального покрытия данных и локальном применении метода главных компонент. В разделе «IV. Обзор метода FisherS» рассматривается алгоритм оценки внутренней размерности, основанный на специальном препроцессинге и разделимости по Фишеру. После изучения этих алгоритмов был реализован фреймворк на языке Python, где представлена реализация этих алгоритмов. API фреймворка описывается в разделе «V. Обзор самописного фрейморка estimators.py». В разделе «VI. Сравнение алгоритмов» расположена главная часть сочинения - это запуск данных методов на различных данных: синтетических без шума, синтетических с шумом. В разделе «VII. Заключение» делаются выводы по результатам сравнения. В разделе «VIII. Ссылки» представлен список используемой литературы.

II. ОПРЕДЕЛЕНИЯ ВНУТРЕННЕЙ РАЗМЕРНОСТИ

Перед тем, как производить обзор алгоритмов и сравнение, нужно понять, что вообще мы собираемся оценивать. Если говорить неформально, то внутренняя размерность - это число, равное эффективному количеству переменных, с помощью которых можно аппроксимировать наши данные с хорошей точностью. Несмотря на то, что термин «внутренняя размерность» используется в исследованиях, связанных с машинным обучением, ему не хватает уникального консенсусного определения. В данном сочинении мы будем использовать следующее определения для внутренней размерности. Внутренняя размерность - это n, если набор данных полностью лежит в пределах n-мерного многообразия без потери информации или с небольшой потерей. Напомним, что многообразием мы называем пространство, локально сходное с евклидовым, а размерность многообразия определяем по размерности евклидового пространства, с которым оно локально сходно.

III. ОБЗОР МЕТОДА C-PCA

A. Общие слова

Данный метод находит минимальное покрытие данных и рассматривает каждое подмножество как участок многообразия данных. Далее на каждом из выбранных подмножеств применяется процедура PCA, чтобы понять локальную структуру данных. Пересмотренный алгоритм PCA, который будет описан далее, может успешно фильтровать шумы в данных и приводит к стабильной и сходящейся оценке с разумным ростом размера региона (показывает эксперимент). Это преимущество по сравнению с обычным PCA, который очень чувствителен к шумам, выбросам и выбору подмножества данных, на котором применяется. Дальнейший анализ показывает, что пересмотренный PCA эффективен по расходованию процессорного времени и использует все данные для оценки внутренней размерности. Алгоритм также применим к постепенному обучению для последовательных данных, но сравнение методов мы будем производить только для случая постоянной выборки).

B. Модифицированный PCA

Традиционный PCA может найти подпространство, на которое проекция данных имеет максимальную дисперсию. Пусть $X = [x_1, ..., x_N], x_i \in \mathbb{R}^D, \bar{x} = \frac{1}{N}\sum_{i=1}^{N}$. Рассмотрим матрицу ковариации для X:

$$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$$

Матрица C является неотрицательно-определенной матрицей, поэтому $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N \geq 0$ с $v_1, ..., v_N$ собственными векторами. Спектральное разложение

матрицы C это $C = GDG^T, D_{ii} = \lambda_i, G = [v_1, ..., v_N]$. Собственный вектор v_i это i-ый главный вектор, а $y_i = v_i * x, \forall x$ - это i-ая главная компонента. $var(y_i) = \lambda_i, cov(y_i, y_j) = 0$. Если набор данных распределены в линейном подпространстве, то для выбора первых d компонент нужно, чтобы они учитывали большую часть дисперсии, содержащейся в X. Для вычисления этого d будем ожидать выполнения одного из следующих критериев:

$$\frac{min_{i=1,...,d}(var(y_i))}{max_{j=d+1,...,N}(var(y_j))} > \alpha >> 1 \quad (1)$$

В наших исследования будем брать $\alpha = 10$

$$\frac{\sum_{i=1}^{d} var(y_i)}{\sum_{i=1}^{N} var(y_i)} > \beta, 0 < \beta < 1. \quad (2)$$

Теперь пусть к нашим данным добавился шум μ с $E(\mu) = 0$ и $var(\mu) = \sigma^2$. Тогда:

$$C' = var(X + \mu) = C + \sigma^2 I$$

Здесь видно, что главные вектора у матрицы C' совпадают с векторами матрицы C, а $\lambda_i' = \lambda_i + \sigma^2$. Если окажется, что σ относительно большая, то критерии (1) и (2) будут неэффективны. Эмпирическую дисперсию шума будем вычислять, как:

$$\sigma^2 = \frac{1}{N - r + 1} \sum_{i=r}^{N} var(y_i) \quad (3)$$

где r определяется, как:

$$\frac{\sum_{i=1}^{r-1} var(y_i)}{\sum_{i=1}^{N} var(y_i)} < P \text{ и } \frac{\sum_{i=1}^{r} var(y_i)}{\sum_{i=1}^{N} var(y_i)} > P$$

где P - это некоторая заранее выбранная константа, близкая к 1. В проводимых исследованиях будет браться $P = 0.95$. Тогда, новые критерии (1) и (2) будут применяться к собственным значениям вида: $var(y_i) = \lambda_i - \sigma^2$, где σ^2 получена из (3). В статье говорится, что предложенный модифицированный PCA устойчив не только к шуму, но и к выбросам.

C. Проблема покрытия множества

Дано множество X из N элементов. Дан набор F его подмножеств, $F = \{F_1, ..., F_N\}$. Задача минимального покрытия множества - эта задача поиска минимального по мощности подмножества F такого, что объединение этого подмножества множеств дает X. Преимущество поиска минимального покрытия заключается в том, что минимальность позволяет сохранить вычислительное время, а использование всего множества улучшить оценку. Задача поиска минимального покрытия является NP - полной, поэтому будем искать его приближенно. Ниже представлен алгоритм поиска, который ищет приближенное минимальное покрытие, используемый в данном методе. Это алгоритм находится под название «Algorithm 1». Данный алгоритм применим для неизменяемых данных, если же данные появляются постепенно, то в статье описано, как обрабатывать этот случай.

Algorithm 1 Минимальное покрытие множества

Require: Число k ближайших соседей
Ensure: минимальное покрытие $F = \{F_i, i = 1, ..., S\}$
  for i=1 to N do
    Для каждой точки множества определяем k самых близких и строим множество $F_i = \{i, i_1, .., i_k\}$. Также заполняем матрицу D, которая является $0-1$ матрицей смежности.
  end for
  for i=1 to N do
    Частотой x_i назовем величину $Q_i = \sum_{j=1}^{N} D_{ij}$
  end for
  for i=1 to N do
    if $Q_i, Q_{i1}, ..., Q_{ik} > 1$ then
      Удалить F_i из покрытия F и $Q_i = Q_i - 1, Q_{i1} = Q_{i1} - 1, ...$
    end if
  end for

Тогда основной алгоритм будет представлять из себя:

1. Шаг 1. На вход подается параметр k, считаем минимальное покрытие с помощью предыдущего алгоритма.
2. Шаг 2. Применяем алгоритм PCA, предложенный в секции III-B для подмножеств $F_i, i = 1, ..., S$. Тогда локальные оценки будут $\{d_i\}_{i=1}^{S}$.
3. Шаг 3. Общей оценкой для сравнения алгоритмов будет среднее этих локальных оценок.

D. Анализ вычислительной сложности

Часть поиска минимального покрытия:

1. подсчет матрицы попарных расстояний $O(N^2 * D)$, где D - изначальное число координат
2. поиск ближайших k точек $O(N^2 log(N))$
3. время поиска минимального покрытия $O(kN)$

Часть, где применяется PCA локально требует $O(k^2 N)$. Тогда общая сложность будет $O(N^2 log(N) + N^2 D)$ в моей реализации. В статье общая сложность была $O(k * N^2 + N^2 D)$, но я заменил часть поиска k максимальных элементов массива на сортировку, а потом выбор k последних, так как для данных, на которых производились вычисления, данный метод эффективнее.

E. Сходимость

В предложенной статье утверждалось, что преимущество данного метода заключается именно в модифицированном алгоритме PCA, благодаря чему обеспечивается сходимость с ростом размера области. Проверим этот факт сначала на выборках «M1_Sphere» и «Mbeta» без добавления шума, а потом с добавление шума.

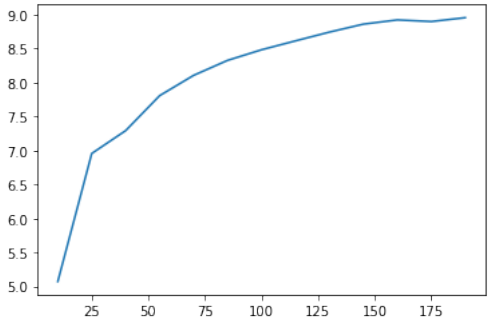

Рис. 1. Зависимость оценки внутренней размерности от числа соседей для выборки «M1_Sphere» из пакета skdims без добавления шума.

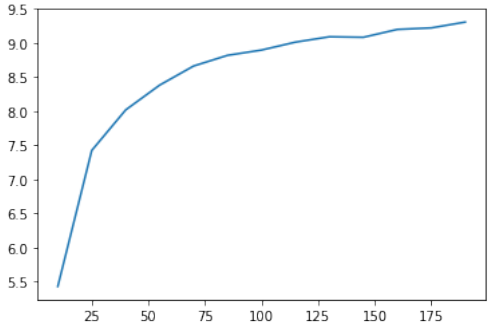

Рис. 2. Зависимость оценки внутренней размерности от числа соседей для выборки «Mbeta» из пакета skdims без добавления шума.

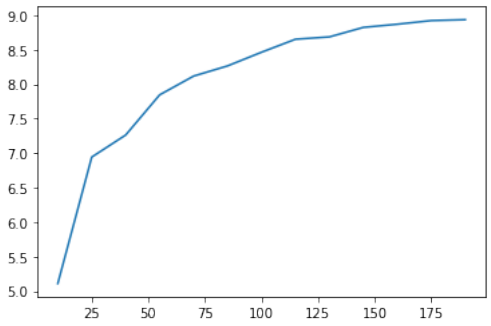

Рис. 3. Зависимость оценки внутренней размерности от числа соседей для выборки «M1_Sphere» из пакета skdims с добавления шума $\sigma = 0.05$.

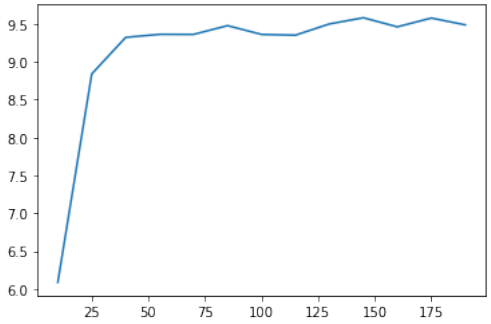

Рис. 4. Зависимость оценки внутренней размерности от числа соседей для выборки «Mbeta» из пакета skdims с добавления шума $\sigma = 0.05$.

Действительно, можем наблюдать факт сходимости к числам, не равным размерности пространсва, в котором расположена выборка.

IV. ОБЗОР МЕТОДА FisherS

A. Основные понятия

Точка $x \in \mathbb{R}^n$ называется линейно отделимой от конечного множества $Y \subset R^n$, если существует линейная функция l, такая что $l(x) > l(y) \forall y \in Y$. Если для каждой точки существует такая линейная функция, то множество называется линейно разделимы. Давайте предположим, что набор данных X нормализован следующим образом:

1. центрирован
2. спроецирован на линейное подпространство, натянутое на первые k главных векторов, где k может быть большим
3. все дисперсии соответсвующих компонет нормированны к единицы
4. каждый вектор выборки нормализован к единичной длине (то есть спроецирован на единичную n-мерную сферу).

После такой нормализации вводим понятие точки, линейно разделенной по Фишеру. Точка $x \in X$ лейно разделена по Фишеру от остальных точек конечного множества с параметром α, если:

$$(x,y) \leq \alpha(x,x)$$

для $\forall y \in Y$, где $\alpha \in [0,1)$. Если данное неравенство выполняется для любой точки множества, то множество называется разделимым по Фишеру с параметром α. Введем новую величину $p_\alpha(y)$ - вероятность того, что точка y отделима от остальных точек на единичной сфере. Так как это вероятность не зависит от выбора точки в силу симметрии, то средняя вероятность по всем точкам будет совпадать с вероятностью в точке и она равна:

$$p_\alpha = \frac{(1-\alpha^2)^{\frac{n-1}{2}}}{\alpha\sqrt{2\pi n}} \tag{4}$$

Внутренняя размерность для параметра α вычисляется по формуле:

$$n_\alpha = \frac{W(\frac{-ln(1-\alpha^2)}{2\pi p_\alpha^2 \alpha^2 (1-\alpha^2)})}{-ln(1-\alpha^2)} \tag{5}$$

B. Анализ вычислительной сложности

1. применение PCA занимает $O(D^2 n + D^3)$
2. подсчет матрицы Грама $O(N^2 D)$

Это две самые тяжелые операции, то есть общая сложность будет $O(N^2 D + D^2 n + D^3)$.

Представленный выше алгоритм устойчив к шумам как показывают эксперименты.

Algorithm 2 Оценка ID для α

Require: Параметр α и данные
Ensure: n_α
1: Для матрицы данных X
2: Центрировать данные $X = X - E(X)$
3: Применить PCA: $[V,U,S] = PCA(X)$, где U - проекции на главные вектора V, S - объясненная дисперсия.
4: Выберем число компонент: $k = max\{i : S(1)/S(i) < C\}$
5: Для колонок U, u_i применить: $u_i = u_i/\sigma(u_i), i = 1...k$
6: Проецируем данные на единичную сферу: $u_i = u_i/|u_i|, i = 1...k$
7: Считаем матрицу Грама: $G = UU^T$
8: Нормализуем матрицу Грама: $G_{ji} = G_{ji}/G_{ii}$
9: Устанавливаем в нули диагональные элементы $G, G_{ii} = 0$
10: Для каждой строки матрицы G считаем число элементов, превосходящих α: v_j - равно этому числу.
11: Считаем эмпирические вероятности: $p_\alpha^j = v_j/N$
12: Считаем среднюю вероятность: p_α
13: По формуле (5) вычисляем n_α

V. ОБЗОР САМОПИСНОГО ФРЕЙМВОКРА ESTIMATORS.PY

На языке Python были реализованы два представленных выше алгоритма. Реализация находится здесь: `https://github.com/matfu-pixel/IdEstimation`. Здесь же находится и notebook с тестированием и сравнением алгоритмов на различных данных. API фреймворка похоже на API библиотеки skdim, представленной в качестве той, на которую следует опираться в анонсе конкурса. Пример вызова методов:

```
import estimators
...
cpca = estimators.cPCA()
cpca.fit(data)
print(cpca.dimension_)
fisherS = estimators.FisherS()
fisherS.fit(data)
print(fisherS.dimension_)
```

Здесь представлены все основные методы, которые реализованы. Чтобы создать оценщик достаточно вызвать соответсвующий конструктор из пакета «estimators.py»: cPCA или FisherS. Для того, чтобы обучить оценщик на данных, достаточно вызвать метод fit() от него. Для CPCA у метода fit() есть следующие параметры:

1. $data$ - рассматриваемые данные
2. $n\_neighbors = 20$ - параметр, который указывает размер подмножеств, которые мы рассматриваем
3. $alpha = 10$ - для критерий (1)
4. $beta = 0.95$ - для критерия (2)
5. $P = 0.95$ - константа для оценки шума
6. noise = False - учитывается ли шум или нет

Для FisherS у метода fit() есть следующие параметры:

1. X - данные
2. $alpha = 0.8$ - константа из разделения по Фишеру
3. $C = 10$ - константа для нахождения главных векторов, на линейную оболочку которых осуществляется проекция.

VI. СРАВНЕНИЕ АЛГОРИТМОВ

Перед тем, как сравнивать алгоритмы, заметим, что в каждом из них есть по одному гиперпараметру и нужно понять, как его выбирать. Для алгоритма FisherS в статье представлен следующий критерий: в качестве оценки берем такой n_α, для которого $\alpha = 0.9\alpha_{max}$, где α_{max} максимальное значение alpha, для которого эмпирическое среднее $p_\alpha > 0$.

Для алгоритма C-PCA метода выбора оптимального числа соседей не было. В статье лишь говорилось, что при увеличении числа соседей достигалась сходимость оценки, причем с ростом области был рост и оценки внутренней размерности, поэтому было принято логичное решение брать максимум оценки по размерам областей.

По теоретическим сложностям трудно сказать, какой алгоритм работает быстрее, но из-за более простых операций, составляющих алгоритм, на практике FisherS показывает сильно лучшие результаты, чем C-PCA. Так как алгоритмы реализовывались на Python с использованием пакета numpy, такую большую разницу можно частично объяснить тем, что в FisherS было проще заменить все основные циклы на операции с массивами numpy, тогда как в C-PCA присутствует много циклов и работа со встроенными списками.

Далее воспользуемся пакетом «skdims» для того, чтобы провести сравнение на синтетических данных с шумом и без. Для подробного ознакомления с каждым из наборов данных можно обратится к данному пакету.

Среднюю ошибку будем считать по данной формуле в процентах:

$$Mean\%error = \frac{100}{\#\{M_i\}} \sum_{i=1}^{\#\{M_i\}} \frac{|n_{M_i}' - n_{Mi}|}{n_{Mi}}$$

Во всех представленных наборах данных число точке $N = 2500$.

Выборки без шума из «skdims»				
название	ID	D	FisherS	C-PCA
M1$_{Sphere}$	10	11	11.01	8.95
M2$_{Affine_3to5}$	3	5	2.67	2.0
M3$_{Nonlinear_4to6}$	4	6	2.83	3.55
M4$_{Nonlinear}$	4	8	5.92	5.05
M5b$_{Helix2d}$	2	3	2.69	2.0
M6$_{Nonlinear}$	6	36	8.56	10.32
M7$_{Roll}$	2	3	2.89	1.67
M8$_{Nonlinear}$	12	72	17.86	20.99
M9$_{Affine}$	20	20	18.65	14.92
M10a$_{Cubic}$	10	11	10.37	8.45
M10b$_{Cubic}$	17	18	16.98	13.63
M10c$_{Cubic}$	24	25	23.46	18.87
M10d$_{Cubic}$	70	71	69.42	48.66
M11$_{Moebius}$	2	3	1.98	2.0
M12$_{Norm}$	20	20	19.97	15.02
M13a$_{Scurve}$	2	3	2.42	1.0
Mbeta	10	40	5.34	9.32
Mn1$_{Nonlinear}$	18	72	16.65	18.0
Mn2$_{Nonlinear}$	24	96	22.18	24.0
Mp1$_{Paraboloid}$	3	12	0.92	2.0
Mean%error			35.20	47.00

Выборки с шумом $\sigma = 0.05$ из «skdims»				
название	ID	D	FisherS	C-PCA
M1$_{Sphere}$	10	11	10.98	9.0
M2$_{Affine_3to5}$	3	5	2.67	2.0
M3$_{Nonlinear_4to6}$	4	6	2.83	3.58
M4$_{Nonlinear}$	4	8	5.92	5.05
M5b$_{Helix2d}$	2	3	2.69	2.0
M6$_{Nonlinear}$	6	36	8.57	10.35
M7$_{Roll}$	2	3	2.89	1.67
M8$_{Nonlinear}$	12	72	17.95	21.0
M9$_{Affine}$	20	20	18.68	14.94
M10a$_{Cubic}$	10	11	10.35	8.39
M10b$_{Cubic}$	17	18	17.01	13.57
M10c$_{Cubic}$	24	25	23.51	18.89
M10d$_{Cubic}$	70	71	69.42	48.71
M11$_{Moebius}$	2	3	1.98	2.0
M12$_{Norm}$	20	20	19.94	14.99
M13a$_{Scurve}$	2	3	2.42	1.0
Mbeta	10	40	5.37	9.56
Mn1$_{Nonlinear}$	18	72	16.57	18.0
Mn2$_{Nonlinear}$	24	96	22.28	24.0
Mp1$_{Paraboloid}$	3	12	0.92	2.54
Mean%error			35.18	46.73

Выборки с шумом $\sigma = 0.5$ из «skdims»				
название	ID	D	FisherS	C-PCA
M1$_{Sphere}$	10	11	10.99	8.68
M2$_{Affine_3to5}$	3	5	2.67	3.0
M3$_{Nonlinear_4to6}$	4	6	2.97	4.19
M4$_{Nonlinear}$	4	8	6.7	5.32
M5b$_{Helix2d}$	2	3	2.69	2.0
M6$_{Nonlinear}$	6	36	9.07	15.29
M7$_{Roll}$	2	3	2.89	1.63
M8$_{Nonlinear}$	12	72	18.37	28.91
M9$_{Affine}$	20	20	18.8	14.96
M10a$_{Cubic}$	10	11	10.64	8.21
M10b$_{Cubic}$	17	18	17.49	14.0
M10c$_{Cubic}$	24	25	25.3	18.69
M10d$_{Cubic}$	70	71	70.34	48.62
M11$_{Moebius}$	2	3	2.3	2.0
M12$_{Norm}$	20	20	20.12	14.99
M13a$_{Scurve}$	2	3	2.44	2.0
Mbeta	10	40	15.45	24.79
Mn1$_{Nonlinear}$	18	72	18.73	37.32
Mn2$_{Nonlinear}$	24	96	25.46	49.63
Mp1$_{Paraboloid}$	3	12	0.92	7.9
Mean%error			31.74	35.55

VII. ЗАКЛЮЧЕНИЕ

Еще раз вспомним основную цель данной работы - сравнение двух методов оценки внутренней размерности. На основе проделанных экспериментах можем сделать вывод, что FisherS не только эффективнее в плане времени, затрачиваемого на оценку, но еще и точнее эту оценку дает. Как мы можем видеть из экспериментов, есть наборы, а именно 3D-параболоид, 6D-параболоид, многообразие, полученное из некоторой гладкой функции плотности, на которых C-PCA дает очень точные оценки, в отличии от FisherS, но в большинстве случаем FisherS лучше предсказывает внутреннюю структуру данных.

VIII. ССЫЛКИ

[1] Methods based on PCA (Fan, M. et al. (2010). Intrinsic dimension estimation of data by principal component analysis)

[2] Fisher separability (Albergante, L., et al. (2019), Estimating the effective dimension of large biological datasets using Fisher separability analysis., 2019 International Joint Conference on Neural Networks, IEEE.)

[3] A. N. Gorban, V. A. Makarov, and I. Y. Tyukin, "The unreasonable effectiveness of small neural ensembles in high-dimensional brain," Physics of Life Reviews, Oct 2018. [Online]. Available: `http://dx.doi.org/10.1016/j.plrev.2018.09.005`

[4] K. Fukunaga, D.R. Olsen, An algorithm for finding intrinsic dimensionality of data, IEEE Transactions on Computers 20 (1971), 176-183.