Уменшение размерности: обзор

И. Куралёнок

СПб, 2017

Зачем бороться с размерностью?

1. Наглядность: сложно смотреть на 100-мерное пространство
2. Количество данных и надежно подбираемых параметров зависимы
3. Экономим стоимость использования:
   - вычислительные ресурсы на этапе обучения и/или использования;
   - стоимость отдельного измерения.
4. Убрать шум

Проклятье размерности

Что происходит с расстояниями когда размерность увеличивается?
Проведем простой опыт: будем равномерно выбирать точки из кубика $[0,1]^n \subset \mathbb{R}^n$.
Оказывается, что при увеличении n:

- Точки все ближе "жмутся" к краю
- Углы между точками выравниваются
- Окрестности все чаще упираются в границы
- Для того, чтобы пространство было плотным надо слишком много точек

$\Rightarrow$ Большая размерность — зло для kNN и не только для него!

Постановка задачи обучения

С построением фичей: повесим на клиента датчики
Наша цель повесить датчики правильно, зная какую информацию мы хотим получить.
Похоже на glass box

Без построения фичей: льется поток неведомых данных
Хотим выделить сигналы, имеющие отношение к искомому
Похоже на black box

Почему можно уменьшить размерность?

В конструктивной постановке

Датчики могут:

- быть разные, а информация одна и та же;
- быть неудачно поставленны оператором и работать не так, как задумано.

Почему можно уменьшить размерность?

В случае сырых данных

В случае сырых данных, мы по сути должны найти датчики, изучив их свойства по имеющимся данным. Поэтому все предыдущее верно и тут, а кроме этого:

- надо понять где причина, а где следствие;
- отделить свойства передачи данных от информации.

Как можно это делать

Feature selection (whitening)

Пример: проверка стат гипотез, что фича нужна

Feature extraction/construction (kostyl production)

Пример: метод главных компонент (PCA)

Embedded models (velosiped prouction)

Пример: LASSO

Как можно это делать

Feature selection (whitening)

Пример: проверка стат гипотез, что фича нужна
Девиз: меньше фишек — легче думать.

Feature extraction/construction (kostyl production)

Пример: метод главных компонент (PCA)

Embedded models (velosiped prouction)

Пример: LASSO

Как можно это делать

Feature selection (whitening)

Пример: проверка стат гипотез, что фича нужна
Девиз: меньше фишек — легче думать.

Feature extraction/construction (kostyl production)

Пример: метод главных компонент (PCA)
Девиз: Мурзик, ну еще капельку!

Embedded models (velosiped prouction)

Пример: LASSO

Как можно это делать

Feature selection (whitening)

Пример: проверка стат гипотез, что фича нужна
Девиз: меньше фишек — легче думать.

Feature extraction/construction (kostyl production)

Пример: метод главных компонент (PCA)
Девиз: Мурзик, ну еще капельку!

Embedded models (velosiped prouction)

Пример: LASSO
Девиз: само сдохнет.

Область применения

Feature selection

Зачем: оценка новых параметров, наглядность, экономия ресурсов, надежный подбор, прибрать шум.

Когда:

- шум/сигнал (dB) у некоторых совсем поганый
- оценка источника сигнала

Область применения

Feature extraction

Зачем: наглядность (быть аккуратными!), экономия ресурсов обучения и использования (предобработка), прибрать шум, гибкость в построении данных.

Когда:

- несколько факторов про одно, но все глядят в разные стороны из-за особенностей реализации;
- хотим обобщающие факторы;
- понимаем структуру данных;
- хочется потратить время.

Область применения

Embedded models

Зачем: экономия ресурсов использования, убрать шум

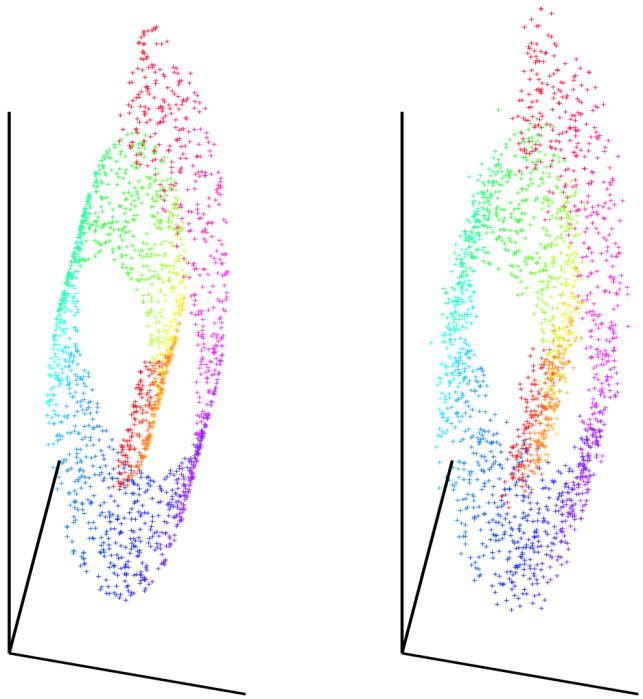
Когда:

- ничего не знаем про данные, зато понимаем как может быть устроено хорошее решение;
- не хотим думать про механизм приемки фичей.

Эффективная размерность

Постановка задачи

Бывают такие ситуации, когда размерность пространства задачи может быть явно меньше чем количество факторов.



Эффективная размерность

"Определение"

Definition (Эффективная размерность)

Будем называть размерность $n' < n$ эффективной размерностью задачи, если существует такое преобразование пространства задачи в $\mathbb{R}^{n'}$, что сохраняет интересные нам свойства

Jonson-Lindestrauss lemma I

Theorem

Для заданного $0 < \epsilon < 1$, *множества точек* $X \supset \mathbb{R}^{n \times m}$, *числа* $k > \frac{\log m}{\epsilon^2}$, *существует линейное преобразование* $f : \mathbb{R}^n \to \mathbb{R}^k$ *такое, что:*

$$(1 - \epsilon)\|u - v\|_2^2 \leq \|f(u) - f(v)\|_2^2 \leq (1 + \epsilon)\|u - v\|_2^2$$

для любой пары $(u, v) \in X^2$.

Jonson-Lindestrauss lemma II

К сожалению лемма довольно tight:

Theorem

Существует множество точек X, для которого потребуется $k = O(-\frac{\log m}{\epsilon^2 \log \epsilon})$.

Но мы помним:

- ограничения только на линейные преобразования;
- есть предположение о полной информации, содержащейся в X.

Как узнать эффективную размерность?

- Спектральная энтропия erank
- Сонаправленность соседей
- Измерение концентрации примеров вблизи точки

Идея: отобразить многомерное пространство в граф, зафиксированной структуры так, чтобы как можно лучше сохранить взаимные расстояния.

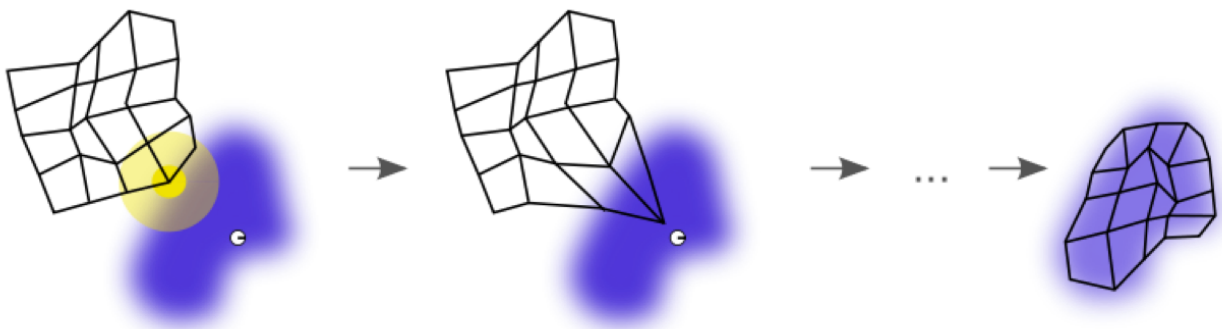
Графы бывают разные:

- Простая 2/3-х мерная сеть
- Шестигранная сетка
- Односвязная “нить”.

SOM

процесс обучения

Можно представить так: кидаем платок и расправляем его.



SOM

алгоритм

Дано: функция расстояния по графу (G), множество точек $(X \in \mathbb{R}^n)$, дискаунт за расстояние (q), дискаунт за итерацию (s)

1. Инициализируем точки, соответствующие узлам графа g_i в $\mathbb{R}^n$
2. Выберем случайную точку x^t в X
3. Находим ближайший g_{i_0}
4. "Двигаем" g_i в сторону точки x^t, по принципу "чем дальше, тем меньше":

$$g_i^{t+1} = g_i^t + q(G(i, i_0), t)s(t)(x_t - g_t)$$

5. Переходим в п.2 пока не сошлось.

SOM

свойства

Что можно с их помощью делать:

- Смотреть на данные “глазами”
- Если из области следует какая-то структура, то можно подобрать ее распределение таким образом
- Аля PCA по 1D кривулинам, полученным с помощью SOM

Проблемы:

- Результат существенно зависит от подбора начальных $g_i \Rightarrow$ зачастую не воспроизводим
- Никаких гарантий сходимости при $s(t) = 1$
- Никаких хороших математических свойств.

Но, it just works!

SOM

рекомендации SOMоводов

- Выбирать начальные значения не по рандому, а по PCA2
- Выбросить выбросы/сгладить движение от дистанции
- Бегать не в одном и том же порядке по множеству
- По минимуму использовать s.