Уменьшение размерности

Feature extraction

И. Е. Кураленок

ikuralenok@gmail.com

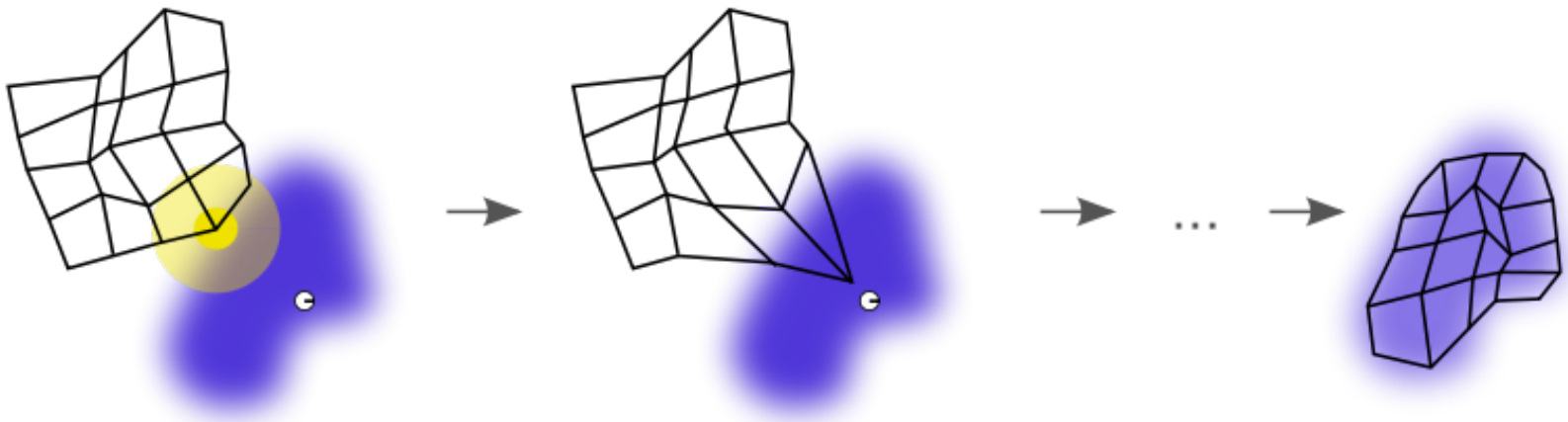
Самоорганизующиеся сети Кохоненна (SOM)

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
- Шестишранная 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)

0. Инициализируем точки, соответствующие узлам графа g_i в $\mathbb{R}^n$
1. Выберем случайную точку x^t в X
2. Находим ближайший g_{i_0}
3. "Двигаем" g_i в сторону точки x^t, по принципу "чем дальше, тем меньше":
$$g_i^{t+1} = g_i^t + q(G(i, i_0), t)s(t)(x_t - g_t)$$
4. Переходим в п. 1 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
- Выкинуть outlier'ов/сгладить движение от дистанции
- Бегать не в одном и том же порядке по множеству
- По минимуму использовать s

Кластерный анализ: определение

Определение (Wikipedia)

Кластерный анализ (англ. cluster analysis) *— задача разбиения заданной выборки объектов (ситуаций) на подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.*

КА: виды

- Иерархические методы (connectivity-based)
- Основанные на центройдах
- Генеративные модели (distribution-based)
- Ограниченные (constraint-based)

КА: иерархические методы

Цель:

$$C : \forall c \in X, c \in C_i, \exists c^* \in C_i : d(c, c^*) < \epsilon$$

Алгоритм (односвязный):

1. Выбираем случайную точку x и кладем ее C_i
2. Ищем соседей x, ближе чем ϵ и добавляем их в C_i
3. Если соседи x кончились, берем следующую точку из C_i и проделываем п. 2
4. Замкнув C_i, $i = i + 1$ и переходим к п. 1

Замечания: можно делать более чем односвязное замыкание, так как подобных делений много, результат сильно зависит от рандома, сложность $O(n^3)$.

КА: иерархические методы, плотность

Цель:

$$C : \forall c \in X, \exists C^* \subseteq C_i : |C^*| > m, \forall c \in C^* d(c, c^*) < \epsilon$$

Алгоритм (DBSCAN):

1. Выбираем случайную точку x и кладем ее в S
2. Для каждой точки из S:
   - 2.1 Ищем соседей в радиусе ϵ
   - 2.2 Если точка еще не в каком-либо кластере, кладем точку в C_i
   - 2.3 Если таких соседей больше m, и добавляем соседей в S

Замечания: сложность $O(n^2)$, повязаны на поиск ближайших соседей в радиусе.

KA: центроиды

Цель:

$$\bar{x}_c = \frac{1}{|c|} \sum_{x \in c} x$$
$$\min_C \sum_{c \in C, |C|=k} \sum_{x \in c} \|x - \bar{x}_c\|$$

Такая задачка *NP*–полная.
Приблизительный алгоритм (*k*–means/Lloyd's):

инициализация: выберем k штук $\bar{x}_c$;

assignment step: для каждой точки $x \in X$ ищем ближайший $\bar{x}_c$ и относим ее к соответствующему кластеру c;

update step: пересчитываем $\bar{x}_c$;

Переходим к assignment step, пока не сойдется.

КА: k–means, свойства

- Никаких гарантий сходимости для k–means.
- Сильная зависимость от начальных параметров
- Вместо центройдов, можно использовать медианы
- Один из самых простых случаев EM $\Rightarrow$ можно попробовать EM гауссовой смеси напрямую

КА: центройды, FOREL

1. Выбираем случайную точку x, зовем ее “центром тяжести” R
2. Добавляем все точки из ϵ-окрестности R в C_i:
3. Пересчитываем $R = \arg\min_{x_0 \in C_i} \sum_{x \in C_i} \|x - x_0\|$

Свойства:

- Быстрый и тупой
- Не сходится
- Результат опять зависит от рандома

КА: центройды, quality threshold

1. Для каждой точки строим ϵ-окрестности.
2. Сортируем полученные множества по уменьшению мощности.
3. Объявляем первое множество в списке кластером, если количество элементов в нем $> N$, иначе выходим.
4. Отсеиваем из окрестностей содержание нового кластера
5. Переходим в п. 2

Свойства:

- Медленный, хотя возможны оптимизации
- Детерминированный
- Контролируемое качество кластеризации

КА: генеративные модели

Цель:

$$p(x|c) = f(x, \lambda_c)$$
$$\max_{c,\lambda_c} \prod_c \prod_{x \in c} p(x|c)$$

Решается EM.

Свойства зависят от того, насколько хорошо подобрана модель пространства.

КА: что еще

- Спектральная кластеризация $L = E - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$
- Constraint-based clustering

КА: оценка

Внутренняя оценка

- Davies-Bouldin index $\frac{1}{n}\sum_1^n \max_{i\neq j}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$
- Dunn index $\min_{i\neq j}\frac{\|c_i-c_j\|}{\max_k \sigma_k}$

Внешняя/косвенная оценка

- Rand measure $\frac{tp+tn}{tp+fp+tn+fn}$
- F-мера $F_\beta = \frac{(1+\beta^2)pr}{\beta^2 p+r}$
- Jaccard index $J(A,B) = \frac{A\cap B}{A\cup B}$
- KL-divergence
- etc.

Johnson–Lindenstrauss lemma

Можно ли уменьшить размерность, при этом сохранив расстояния?

Теорема (Johnson–Lindenstrauss)

Для любого $0 < \epsilon < 1$, $X \supset \mathbb{R}^n : |X| = m$, $k > k_0 = O\left(\frac{\log m}{\epsilon^2}\right)$,

найдется липшецева функция $f : \mathbb{R}^n \to \mathbb{R}^k$, *такая что:*

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

, *для* $\forall (u, v) \in X^2$.

NB: граница достаточно точная, так как можно построить множество из m точек, которое потребует размерности

$$O\left(\frac{\log m}{\epsilon^2 \log \epsilon^{-1}}\right)$$