Уменшение размерности: обзор FE

И. Куралёнок

СПб, 2017

Как отличить extraction от датчиков?

Пример с мобильниками

Как понять каким образом человек передвигается?
Датчики: повесим специальный датчик
FE: соберем данные от пользователей и спомощью существующих датчиков предскажем.

Какие есть способы собрать что-то удобное?

- Без использования дополнительных данных (какая-то нормализация)
  - с точки зрения линейной алгебры (PCA/Kernel PCA)
  - с точки зрения разделения сигналов (ICA)
  - группировка (кластеризация)
  - какие-то еще принципы, специфичные для данных (например поделить голосовой сигнал)
- С дополнительными данными
  - Metric learning
  - Хитрая кластеризация с предпочтениями
  - Transfer learning
- Пускай само думает
  - Deep learning

Метод главных компонент

Из определения

Задача: превратить множество скоррелированных сигналов в множество линейно некоррелирующих с помощью поворота:

$$\begin{aligned} &\min \sum_{(i,j)} y_i^T y_j \\ &X = Q^T Y Q \\ &q_i^T q_j = 0, \forall i \neq j \\ &q_i^T q_i = 1, \forall i \end{aligned}$$

Мы знаем хотя бы одно решение такой задачи! При этом не все строки в Y одинаково полезны. Отсортируем их по модулю.

Метод главных компонент

Альтернативный способ: отсосы

Попробуем итеративно выбирать направление, несущее наибольшую “информацию” (на самом деле дисперсию):

$$\arg\max_{\|w\|_2=1} \sum_i (x_i^T w)^2$$
$$\arg\max_{\|w\|_2=1} w^T X^T X w$$
$$\arg\max_w \frac{w^T X^T X w}{w^T w}$$

Это Rayleigh Quotient, для которого известно решение – главный собственный вектор.

Как реализовать PCA?

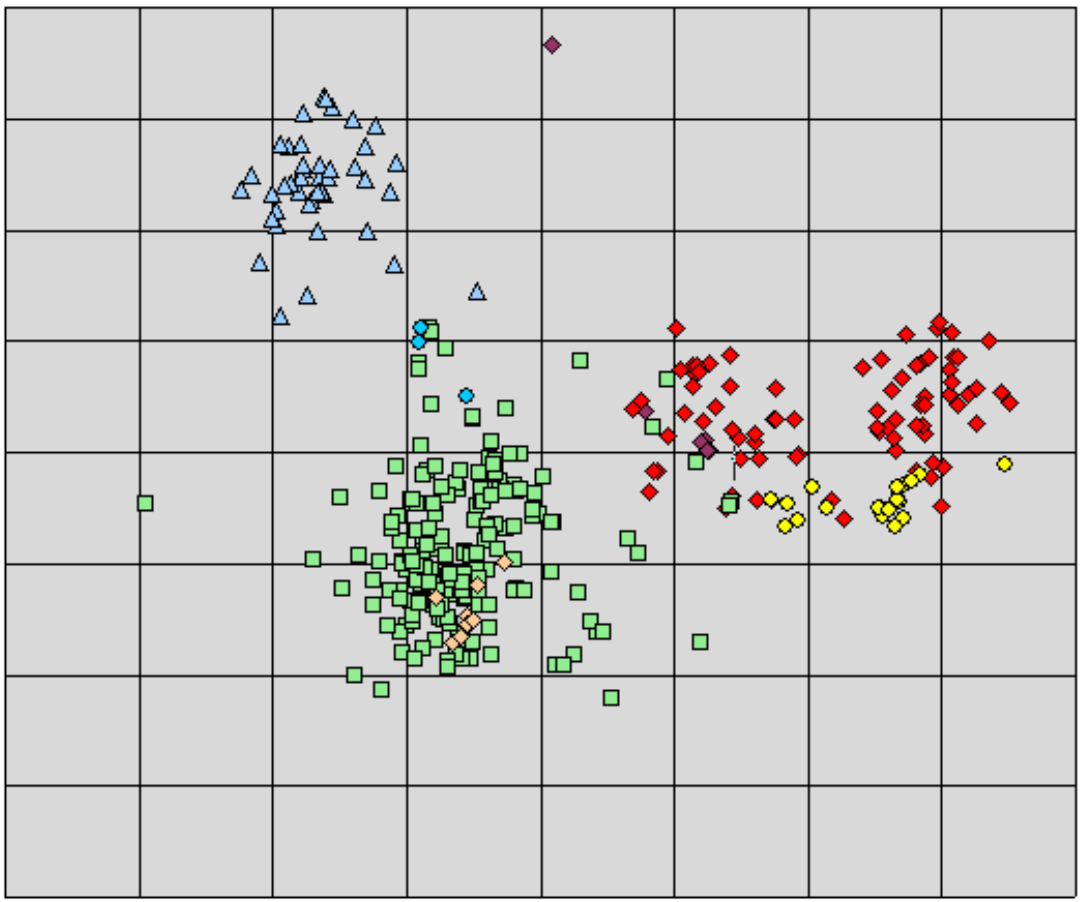
- Можно искать корни уравнения $det\|X - \lambda E\| = 0$, но мы так делать не будем
- Возведение в степень (power iteration)
- Тридиагонализация + (D & C или что-то более модное)
- Последовательное LQ (QR) разложение

Свойства PCA

- Все хорошо, когда пространство похоже на нормальное распределение. Однако на практике это обычно не так:

Свойства PCA

- Все хорошо, когда пространство похоже на нормальное распределение. Однако на практике это обычно не так:



Свойства PCA

- Все хорошо, когда пространство похоже на нормальное распределение. Однако на практике это обычно не так:
- Мы рассмотрели линейную зависимость, а что если она нелинейна?

Свойства PCA

- Все хорошо, когда пространство похоже на нормальное распределение. Однако на практике это обычно не так:
- Мы рассмотрели линейную зависимость, а что если она нелинейна?

$\Rightarrow$ попробуем стандартный фокус с kernel

Kernel PCA

Хотим решить задачу PCA в каком-то другом пространстве, образованном из данного через преобразование Φ. Для этого нам нужно лишь узнать проекции на собственные вектора $\Phi(x)^T v_k$. Для этого воспользуемся таким:

$$\begin{aligned} \lambda v &= Cv \\ x^T v &= \tfrac{1}{\lambda} x^T Cv \\ v &= \sum a_i \Phi(x_i) \end{aligned}$$

Все это добро выполняется для точек из X. Представим, что kernel space имеет размерность N.

$$N\lambda K a = K^2 a$$

Решив это уравнение, получим разложение любого $\Phi(x)$ по V

$$\Phi(x)^T v_k = \frac{1}{\lambda_k} \sum_i a_i K(x, x_i)$$

Общая схема ICA

Представим сигнал как сумму независимых например так:

$$x = A(B(\alpha_i, \beta_i))_{i=1}^{n}$$

Теперь наша задача подобрать такие (A, α, β), которые будут наилучшим образом объяснять данные.

Общая схема ICA

Представим сигнал как сумму независимых например так:

$$x = A(B(\alpha_i, \beta_i))_{i=1}^{n}$$

Теперь наша задача подобрать такие (A, α, β), которые будут *наилучшим образом объяснять* данные.

Общая схема ICA

Представим сигнал как сумму независимых например так:

$$x = A(B(\alpha_i, \beta_i))_{i=1}^{n}$$

Теперь наша задача подобрать такие (A, α, β), которые будут *наилучшим образом объяснять* данные. Это можно делать по тем же принципам что и PCA:

- получше объяснить данные (ICA on LL/Infomax/KL/etc.);
- выделяя самые "жирные" направления по-одному (Projection Pursuit)

Общая модель и ее свойства

$$x = As, s = Wx$$

Генеративная модель с такими свойствами:

- так как подбираем одновременно и A и s, нужно зафиксировать свойства s (ожидание и scale)

Общая модель и ее свойства

$$x = As, s = Wx$$

Генеративная модель с такими свойствами:

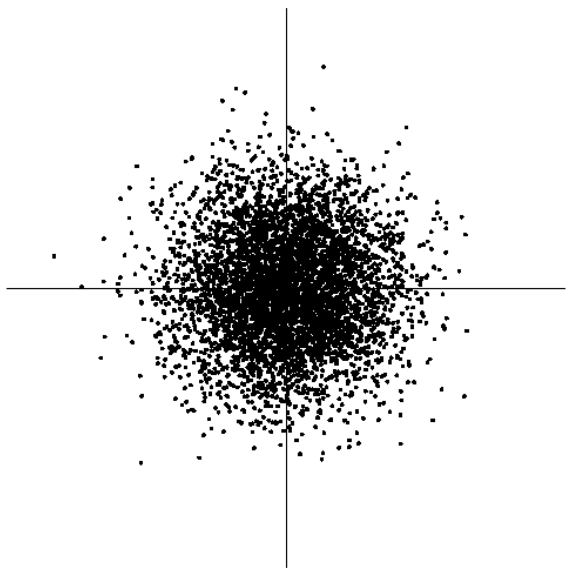
- так как подбираем одновременно и A и s, нужно зафиксировать свойства s (ожидание и scale)
- s не могут быть одинаково симметрично распределены (например нормально)

Общая модель и ее свойства

$$x = As, s = Wx$$

Генеративная модель с такими свойствами:

- так как подбираем одновременно и A и s, нужно зафиксировать свойства s (ожидание и scale)
- s не могут быть одинаково симметрично распределены (например нормально)



Общая модель и ее свойства

$$x = As, s = Wx$$

Генеративная модель с такими свойствами:

- так как подбираем одновременно и A и s, нужно зафиксировать свойства s (ожидание и scale)
- s не могут быть одинаково симметрично распределены (например нормально)
- исходные сигналы s никак не упорядочены

“Ненормальность” и независимость

Рассмотрим проекцию: $w^T x$, тогда:

$$w^T x = w^T A s = (A^T w)^T s = z^T s$$

Тогда минимизация нормальности проекции приведет ее к $z = (0, \ldots, 1, \ldots, 0)^T$, который соответствует одному из искомых независимых направлений.

$\Rightarrow$ осталось понять как искать ненормальных

Projection pursuit

Известно, что нормальное распределение имеет моменты старших порядков либо 0 либо 1. Поэтому можно поискать такие w, которые максимизируют отличия моментов от 0 или 1. После дисперсии там skewness и kurtosis. Которые отвечают за скрученность и остроконечность. Кубы никто не любит, так что обычно максимизируют kurtosis. Можно поставить такую задачу:

$$Kurt(\xi) = \frac{\mu(\xi_i - \mu(\xi))^4}{(\mu(\xi_i - \mu(\xi))^2)^2}$$

$$\arg\max_w Kurt(w^T x)$$

Ненормальность через Negentropy

Kurt неустойчив к выбросам, поэтому на практике хочется использовать что-то другое. Хотим померить количество информации, которую нам дает “ненормальность”:

$$J(X|M(X)) = H(\mathcal{N}(\hat{\mu}(X), \hat{\sigma}(X))) - H(M(X))$$

Так как нормальное распределение — самое непонятное, то $J > 0$. J тяжело считать напрямую, поэтому используют приближения:

$$J(y) \simeq \frac{1}{12}\mu(y^3)^2 + \frac{1}{48}Kurt(y)^2$$

классическое, но работает не очень, так как *Kurt*.

$$J(y) \simeq \sum_{i=1}^{p}(\mu(g_i(y^*)) - \mu(g_i(\nu))), \nu \sim \mathcal{N}(0,1), y^* = \Sigma^{-\frac{1}{2}}(y - \mu(y))$$

Метод FastICA

1. Нормализуем данные
2. Выберем такую f, которые позволят нам быстро сходиться и будут далеки от квадратов
3. Посчитаем от них производные g
4. Возьмем произвольный w
5. Минимизируем с помощью Ньютона

$$\hat{w} = \arg \min_{\|w\|=1} \left(\mathbb{E}\left(f\left(w^T x\right)\right) - \mathbb{E}\left(f\left(\nu\right)\right)\right)$$

Через variational Bayes

Итак мы хотим максимизировать KL:

$$\arg\max_{\Sigma,\mu} \int_{\gamma} KL(p(\gamma|X)\|p(\gamma|\mathcal{B}(\alpha,\beta))d\gamma$$

если развернуть все в likelihood, то получится так:

$$\arg\max_{\Sigma,\mu} \int_{\gamma} KL(p(X|\gamma)\|p(\gamma|\mathcal{B}(\alpha,\beta))d\gamma$$

Наша задача из этого выитегрировать γ. Фишка в том, что размерность векторов α, β, могут быть существенно меньше чем β

Что есть еще

Область очень горячая и есть еще много чего:

- Auto encoders
- t-SNE
- Locally Linear Embedding
- etc.

Locally Linear Embedding

Что мы сегодня узнали

- Чем мы занимаемся зависит от точки зрения: можно гордо делать датчики, а можно клепать костыли
- Если не использовать никакой дополнительной информации, то можно постараться нормировать модель тем или иным способом