Методы понижения размерности

Евгений Соколов
sokolov.evg@gmail.com

Large-Scale Machine Learning
10 Сентября 2015

Организационное

- На следующей неделе занятия не будет
- Сайт курса: `https://sites.google.com/site/lsmlfivt2015`

Содержание

Постановка задачи

Дано:

- Обучающая выборка $X^\ell = \{x_1, \dots, x_\ell\}$
- Ответы $y_1, \dots, y_\ell$
- Признаковые описания $x_i = (x_{i1}, \dots, x_{iD})$

Задача:

- Найти новые признаковые описания $z_i = (z_{i1}, \dots, z_{id})$
- $d \ll D$
- Новые признаки должны удовлетворять определенным условиям

Линейные модели

$$a(x) = \langle w, x \rangle + w_0$$
$$a(x) = \operatorname{sign}(\langle w, x \rangle + w_0)$$

- Часто используются на разреженных признаках (категориальные, bag-of-words)
- Быстрое вычисление прогноза
- Чтобы учитывать нелинейные зависимости, нужно добавлять признаки высоких порядок
- Альтернатива: перевести объекты в новое признаковое пространство с помощью нелинейных преобразований

Градиентный бустинг

$$a(x) = \sum_{n=1}^{N} \alpha_n b_n(x),$$

$b_n(x)$ — решающие деревья небольшой высоты $(2 - 10)$.

- Один из самых сильных методов машинного обучения
- Плохо работает на разреженных признаках
- Преобразование разреженных признаков в небольшое число плотных может существенно повысить качество

Метрические методы

$$a(x) = \arg\max_y \sum_{k=1}^{K} w(x_{(k)})[y_{(k)} = y]$$

Примеры применений:

- Поиск наиболее похожих изображений в базе
- Распознавание лиц
- Медицинская диагностика на основе близости генетических профилей
- Один из базовых алгоритмов при блендинге

Метрические методы

Особенности при больших объемах данных:

- В пространствах большой размерности евклидова метрика теряет смысл (проклятие размерности)
- Сложность поиска ближайшего соседа примитивным способом: $O(\ell d)$
- Есть методы быстрого поиска соседей: kd-tree, ball-tree и т.д., которые при малых d имеют сложность $O(\log \ell)$
- При $d \to \infty$ сложность этих методов становится линейной: $O(\ell d)$

Выход — понижение размерности.

Содержание

Locality-sensitive hashing

Идея: построить хэш-функцию, которая будет давать одинаковые значения на близких объектах, и разные — на далеких.

Зафиксируем метрику $\rho(x, y)$.

Определение. Семейство функций $\mathcal{F}$ называется (d_1, d_2, p_1, p_2)-чувствительным, если для всех $x, y \in X^L$ выполнено:

- Если $\rho(x, y) \leqslant d_1$, то $\mathbb{P}_{f \in \mathcal{F}}\left[f(x) = f(y)\right] \geqslant p_1$.
- Если $\rho(x, y) \geqslant d_2$, то $\mathbb{P}_{f \in \mathcal{F}}\left[f(x) = f(y)\right] \leqslant p_2$.

Здесь под вероятностью $\mathbb{P}_{f \in \mathcal{F}}$ понимается некоторое распределение на всех функциях семейства $\mathcal{F}$.

Мера Джаккарда

Пусть A и B — множества. Расстояние Джаккарда между ними:

$$\rho_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Семейство хэш-функций — MinHash:

1. Пусть все множества являются подмножествами универсального множества $U = \{u_1, \dots, u_n\}$
2. Выберем случайную перестановку π на U (чем меньше $\pi(u)$, тем больше важность u)
3. Хэш-функция: $f_\pi(A) = \min\{\pi(u) \mid u \in A\}$ (важность самого важного элемента в U)

Мера Джаккарда

Докажем, что MinHash — $(d_1, d_2, 1 - d_1, 1 - d_2)$-чувствительное семейство.

Разобьем элементы U на три группы:

1. $u \in A, u \in B$ — p штук
2. $u \in A, u \notin B$ или $u \notin A, u \in B$ — q штук
3. $u \notin A, u \notin B$

$$
\begin{aligned}
\mathbb{P}[f_\pi(A) = f_\pi(B)] = \mathbb{P}\left[\min_{u \in A} \pi(u) < \min_{u \in B} \pi(u)\right] = \\
= \frac{p}{p+q} = \frac{|A \cap B|}{|A \cup B|} = \\
= 1 - \rho_J(A, B)
\end{aligned}
$$

Другие метрики

Семейства хэш-функций известны для некоторых других метрик:

- Хэммингово расстояние:

$$\mathcal{F} = \{f_i(x) = x_i \,|\, i = 1, \dots, d\}.$$

- Косинусное расстояние:

$$\mathcal{F} = \{f_w(x) = \text{sign}\langle w, x\rangle \,|\, w \in \mathbb{R}^d, \|w\| = 1\}.$$

- Евклидово расстояние:

$$\mathcal{F} = \left\{ f_{w,b}(x) = \left\lfloor \frac{\langle w, x\rangle + b}{r} \right\rfloor \;\middle|\; w \in \mathbb{R}^d, b \in [0, r) \right\},$$

где $w \sim \mathcal{N}(0, 1)$.

Композиция хэш-функций

- Одна хэш-функция, как правило, дает плохое качество
- Нужно строить композиции!

Два подхода:

- Хэш-функция возвращает вектор, а не число:

$$g(x) = (f_1(x), \dots, f_m(x)), \quad f_i \in \mathcal{F}$$

- Чтобы признать объекты x и z похожими, они должны пройти хотя бы один из тестов:

$$[g_1(x) = g_1(z)] + \dots + [g_L(x) = g_L(z)] \geqslant 1$$

Получаем $(d_1, d_2, 1 - (1 - p_1^m)^L, 1 - (1 - p_2^m)^L)$-чувствительный алгоритм.

Теоретические гарантии

Определение. Алгоритм решает задачу поиска c-ближайшего соседа, если для нового объекта u он с вероятностью $1 - \varepsilon$ возвращает объект выборки, удаленный от u не более чем в c раз сильнее, чем ближайший к u объект выборки.

Теорема. Существуют такие параметры L и m, что описанный алгоритм будет решать задачу поиска c-ближайшего соседа за $O(d\ell^r \log \ell)$.

Для многих функций расстояния: $r \approx 1/c$.

Содержание

Проблемы LSH

- Хэш-функции выбираются случайно, без учета распределения данных
- Могут потребоваться композиции из сотен хэш-функций, чтобы получить хорошее качество
- Хэш-функции известны лишь для небольшого числа метрик

Идея: обучать хэш-функции

Обучение хэшированию

Будем строить хэши $h(x) = (h_1(x), \dots, h_n(x))$, на основе которых будет определяться близость объектов:

$$\rho(x, z) \approx \rho(h(x), h(z)).$$

Требования к хэшам:

- Дисперсия обучающей выборки после хэширования должна быть как можно больше
- Каждая хэш-функция центрирована:
$$\sum_{i=1}^{\ell} h_k(x_i) = 0$$
- Хэши ортогональны:
$$\sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m$$

Обучение хэшированию

Будем искать хэши в виде

$$h_k(x) = \operatorname{sign}\langle w_k, x\rangle$$

Матрица хэшей:

$$H = (h_k(x_i))_{i,k} = \operatorname{sign} XW$$

Задача оптимизации:

$$\begin{cases} \operatorname{tr} H^T H \to \max\limits_{W} \\ h_k(x_i) \in \{-1, +1\}, \quad k = 1, \dots, n; i = 1, \dots, \ell \\ \sum\limits_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum\limits_{i=1}^{\ell} \sum\limits_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{cases}$$

Слишком сложно из-за бинарности хэшей.

Обучение хэшированию

Упростим:

$$h_k(x) = \langle w_k, x \rangle$$

Задача оптимизации:

$$\begin{cases} \operatorname{tr} W^T X^T X W \to \max\limits_W \\ W^T W = I \end{cases}$$

Похоже на PCA!

Решение: $w_1, \dots, w_k$ — собственные векторы $X^T X$, соответствующие наибольшим собственным значениям.

Spectral Hashing

В PCA Hashing мы искали хэши, исходя из максимизации дисперсии, хотя исходная задача — сохранение расстояний в пространстве хэшей.

Исправимся:

$$\begin{cases} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} v_{ij} \|h(x_i) - h(x_j)\|^2 = H^T (D - V) H \to \min\limits_{H} \\ h_k(x_i) \in \{-1, +1\}, \quad k = 1, \dots, n; i = 1, \dots, \ell \\ \sum_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{cases}$$

$v_{ij} = \exp(-\|x_i - x_j\|^2 / \sigma^2)$
$d_{ii} = \sum_{j=1}^{\ell} v_{ij}$

Задача NP-полная.

Spectral Hashing

Построим релаксацию:

$$\begin{cases} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} H^T (D - V) H \to \min_H \\ \sum_{i=1}^{\ell} h_k(x_i) = 0, \quad k = 1, \dots, n \\ \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} h_k(x_i) h_m(x_j) = 0, \quad k \neq m \end{cases}$$

Решение: собственные векторы $D - V$, соответствующие наименьшим собственным значениям

А как вычислять хэши для новых объектов?

Spectral Hashing

Пусть объекты имеют распределение $p(x)$.

$$\begin{cases} \displaystyle\int \|h(x_1) - h(x_2)\|^2 W(x_1, x_2) p(x_1) p(x_2) dx_1 dx_2 \to \min_{h(x)} \\ \displaystyle\int h(x) p(x) dx = 0 \\ \displaystyle\int h(x) h^T(x) p(x) dx = I \end{cases}$$

Решения — собственные функции (дискретизованного) оператора Лапласа-Бельтрами

Содержание

Векторные представления слов

Хотим каждое слово представить как вещественный вектор:

$$w \to \vec{w} \in \mathbb{R}^d$$

Какие требования?

- Размерность d должна быть не очень велика
- Похожие слова должны иметь близкие векторы
- Арифметические операции над векторами должны иметь смысл

word2vec

- Будем обучать представления слов так, чтобы они хорошо предсказывали свой контекст
- Выборка состоит из текстов, каждый представляет собой последовательность слов $w_1, \dots, w_i, \dots, w_n$
- Контекст слова w_i: $c(w_i) = (w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k})$

Функционал для каждого текста:

$$\sum_{i=1}^{n} \sum_{\substack{j=-k \\ j \neq 0}}^{k} \log p(w_{i+j} \mid w_i) \to \max,$$

где вероятность вычисляется через soft-max:

$$p(w_i \mid w_j) = \frac{\exp(\langle \vec{w}_i, \vec{w}_j \rangle)}{\sum_w \exp(\langle \vec{w}, \vec{w}_j \rangle)}$$

(сумма в знаменателе — по всем словам из словаря)

Свойства представлений

- Косинусное расстояние хорошо отражает схожесть слов по тематике (в зависимости от корпуса)
- $\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}}$
- $\vec{\text{Moscow}} - \vec{\text{Russia}} + \vec{\text{England}} \approx \vec{\text{London}}$
- Перевод: $\vec{\text{one}} - \vec{\text{uno}} + \vec{\text{four}} \approx \vec{\text{quatro}}$
- Среднее представление по всем словам в тексте — хорошее признаковое описание

Содержание

Нейросети

Нейронная сеть принимает на вход вектор признаков и делает нелинейное преобразование на каждом слое:

$$
\begin{aligned}
v_1 &= v_1(x) = \sigma(W_1 x) \\
v_2 &= v_2(v_1) = \sigma(W_2 v_1) \\
&\ldots \\
v_n &= v_n(v_{n-1}) = \sigma(W_n v_{n-1})
\end{aligned}
$$

v_n — вектор ответов (как правило, одно число в регрессии и бинарной классификации)

Автокодировщики

Идея:

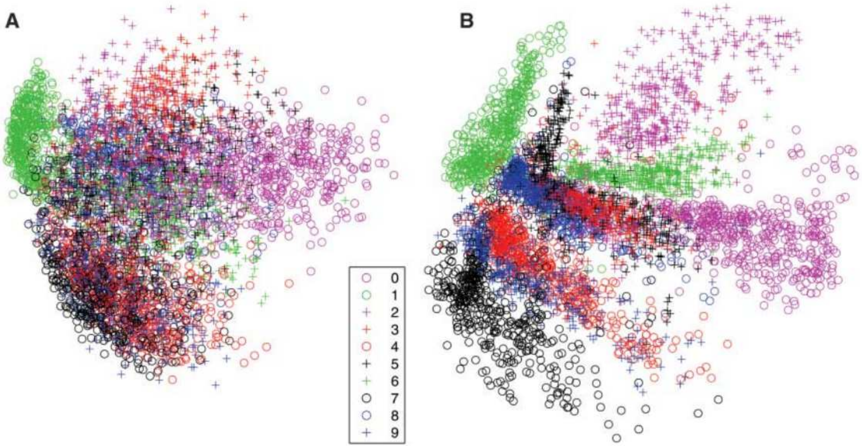
- Сделать слои симметричными, центральный слой самый маленький по числу нейронов
- Последний слой по размеру совпадает с первым
- Требовать, чтобы выход как можно сильнее совпадал со входом
- Тогда внутренний слой будет давать сжатое описание объекта, по которому хорошо восстанавливаются исходные признаки

Пример архитектуры для MNIST:
784-1000-500-250-30-250-500-1000-784

Примеры использования

- Генерация новых признаков, нелинейно зависящих от исходных
- Поиск близких объектов по сжатым признакам (kNN)
- Визуализация (при размере внутреннего слоя 2–3)

Примеры использования



MNIST. Слева PCA, справа автокодировщик

Обучение автокодировщиков

Проблема: обучение глубокой нейросети с помощью backpropagation затруднено.

Решение:

- Жадное обучение нейросети по слоям
- Обучение одного слоя — backpropagation или RBM

Автокодировщик — хороший способ инициализировать нейросеть для решения supervised-задачи.