Лекция 12
Обучение без учителя

Е. А. Соколов
ФКН ВШЭ

30 ноября 2017 г.

До сих пор мы изучали методы обучения с учителем — то есть методы, которые восстанавливают зависимость по объектам с известными ответами. Если задать семейство моделей и функционал ошибки, то обучение сводится к выбору лучшей модели с точки зрения этого функционала.

Также существует большой класс *обучения без учителя (unsupervised learning)*, в которых отсутствует целевая переменная, и требуется восстановить некую скрытую структуру в данных. Примером может служить визуализация — задача изображения многомерной выборки на двухмерной плоскости. Чтобы визуализация была осмысленной, при таком отображении нужно сохранить основные закономерности данных. Формализовать требование «сохранить основные закономерности» тяжело, и поэтому строго оценить качество решения данной задачи не представляется возможным.

Мы рассмотрим несколько типов задач обучения без учителя, обсудим методы их решения и подходы к измерению качества.

1 Кластеризация

Пусть дана выборка объектов $X = (x_i)_{i=1}^{\ell}$, $x_i \in \mathbb{X}$. В задаче кластеризации требуется выявить в данных K кластеров — таких областей, что объекты внутри одного кластера похожи друг на друга, а объекты из разных кластеров друг на друга не похожи. Более формально, требуется построить алгоритм $a : \mathbb{X} \to \{1, \dots, K\}$, определяющий для каждого объекта номер его кластера; число кластеров K может либо быть известно, либо являться параметром.

Кластеризовать можно много что: новости по сюжетам, пиксели на изображении по принадлежности объекту, музыку по жанрам, сообщения на форуме по темам, клиентов по типу поведения.

В нашей постановке задачи много неточностей — в частности, мы не указали, как измеряется сходство объектов. Как и раньше, начнём обсуждение задачи с метрик качества.

§1.1 Метрики качества кластеризации

Существует два подхода к измерению качества кластеризации: внутренний и внешний. Внутренний основан на некоторых свойствах выборки и кластеров, а внеш-

ний использует дополнительные данные — например, информацию об истинных кластерах.

Приведём несколько примеров внутренних метрик качества. Будем считать, что каждый кластер характеризуется своим *центром* c_k.

1. Внутрикластерное расстояние:

$$\sum_{k=1}^{K}\sum_{i=1}^{\ell}[a(x_i)=k]\rho(x_i,c_k), \tag{1.1}$$

   где $\rho(x,z)$ — некоторая функция расстояния. Данный функционал требуется минимизировать, поскольку в идеале все объекты кластера должны быть одинаковыми.

2. Межкластерное расстояние:

$$\sum_{i,j=1}^{\ell}[a(x_i)\neq a(x_j)]\rho(x_i,x_j).$$

   Данный функционал нужно максимизировать, поскольку объекты из разных кластеров должны быть как можно менее похожими друг на друга.

3. Индекс Данна (Dunn Index):

$$\frac{\min_{1\leqslant k<k'\leqslant K} d(k,k')}{\max_{1\leqslant k\leqslant K} d(k)},$$

   где $d(k,k')$ — расстояние между кластерами k и k' (например, евклидово расстояние между их центрами), а $d(k)$ — внутрикластерное расстояние для k-го кластера (например, сумма расстояний от всех объектов этого кластера до его центра). Данный индекс необходимо максимизировать.

Внешние метрики возможно использовать, если известно истинное распределение объектов по кластерам. В этом случае задачу кластеризации можно рассматривать как задачу многоклассовой классификации, и использовать любую метрику оттуда — F-меру с микро- или макро-усреднением.

§1.2 K-Means

Одним из наиболее популярных методов кластеризации является *K-Means*, который оптимизирует внутрикластерное расстояние (1.1), в котором используется квадрат евклидовой метрики.

Заметим, что в данном функционале имеется две степени свободы: центры кластеров c_k и распределение объектов по кластерам $a(x_i)$. Выберем для этих величин произвольные начальные приближения, а затем будем оптимизировать их по очереди:

1. Зафиксируем центры кластеров. В этом случае внутрикластерное расстояние будет минимальным, если каждый объект будет относиться к тому кластеру, чей центр является ближайшим:

$$a(x_i)=\arg\min_{1\leqslant k\leqslant K}\rho(x_i,c_k).$$

2. Зафиксируем распределение объектов по кластерам. В этом случае внутрикластерное расстояние с квадратом евклидовой метрики можно продифференцировать по центрам кластеров и вывести аналитические формулы для них:

$$c_k = \frac{1}{\sum_{i=1}^{\ell}[a(x_i) = k]} \sum_{i=1}^{\ell}[a(x_i) = k]x_i.$$

Повторяя эти шаги до сходимости, мы получим некоторое распределение объектов по кластерам. Новый объект относится к тому кластеру, чей центр является ближайшим.

Результат работы метода K-Means существенно зависит от начального приближения. Существует большое количество подходов к инициализации; одним из наиболее успешных считается k-means++.

§1.3 Графовые методы

Графовые методы кластеризации — это простейшие методы, которые основаны на построении графа близости. Его вершинами являются объекты, а выбор рёбер зависит от конкретного алгоритма. Например, рёбра могут быть проведены между объектами, расстояния между которыми меньше определённого порога. Кластерами же объявляются группы объектов, попадающих в одну компоненту связности.

Такие подходы очень простые, но при грамотном выборе функции расстояния (скажем, обученной под конкретную задачу) могут показывать очень хорошие результаты.

§1.4 Иерархическая кластеризация

Описанные выше методы кластеризации находят «плоскую» структуру кластеров. В некоторых задачах возникает потребность в построении иерархии кластеров, в которой верхним уровнем является один большой кластер, а нижним — ℓ кластеров, каждый из которых состоит из одного объекта. Например, при кластеризации новостей можно рассчитывать, что чем ниже мы спускаемся по иерархии, тем более тонкие различия между сюжетами будут выделяться.

Одним из подходов является восходящая кластеризация. Она начинается с нижнего уровня, на котором все объекты принадлежат к отдельным кластерам: $C^\ell = \{\{x_1\}, \ldots, \{x_\ell\}\}$. Каждый следующий уровень C^j получается путём объединения двух наиболее похожих кластеров с предыдущего уровня $C^{j+1} = \{X_1, \ldots, X_{j+1}\}$. Схожесть кластеров определяется с помощью некоторой функции $d(X_m, X_n)$ — например, это может быть расстояние между центрами кластеров.

2 Визуализация

Как уже упоминалось выше, задача *визуализации* состоит в отображении объектов в двух- или трёхмерное пространство с сохранением отношений между ними. Под сохранением отношений обычно понимают близость попарных расстояний в исходном и в новом пространствах.

Так, в методе многомерного шкалирования (multidimensional scaling, MDS) минимизируются квадраты отклонений между исходными и новыми попарными расстояниями:

$$\sum_{i \neq j}^{\ell} (\rho(x_i, x_j) - \rho(z_i, z_j))^2 \to \min_{z_1, \ldots, z_\ell},$$

где $x_i \in \mathbb{R}^D$ — исходные объекты, а $z_i \in \mathbb{R}^d$, $2 \leqslant d \leqslant 3$ — их низкоразмерные проекции. Обратим внимание на две особенности данного подхода:

- Исходные объекты не обязаны принадлежать евклидову пространству — достаточно лишь уметь вычислять расстояния между ними. Благодаря этому можно визуализировать даже сложные объекты вроде строк.

- Проекции объектов ищутся непосредственно, без какой-либо параметрической зависимости между ними и исходными представлениями объектов. Из-за этого затруднительно добавить к визуализации новые данные. Впрочем, это и не нужно — мы ведь хотим просто нарисовать объекты и посмотреть на них. Если же требуется отображать и новые, тестовые данные, то следует пользоваться методами понижения размерности, которые преобразуют объекты с помощью некоторой модели.

Одним из наиболее популярных методов визуализации на сегодняшний день является *t-distributed stochastic neighbor embedding (t-SNE)*, который исправляет несколько ключевых проблем многомерного шкалирования.

Для начала заметим, что нам не так важно точное сохранение расстояний после проецирования — достаточно лишь сохранить пропорции. Например, если $\rho(x_1, x_2) = \alpha\rho(x_1, x_3)$, то в новом пространстве достаточно выполнения такого же равенства, чтобы соотношения между этими тремя объектами были сохранены: $\rho(z_1, z_2) = \alpha\rho(z_1, z_3)$. Будем использовать нормальную плотность для измерения сходства объектов в исходном пространстве:

$$\rho(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right).$$

Отнормируем эти близости так, чтобы получить вектор распределений расстояний от объекта x_j до всех остальных объектов:

$$p(i \,|\, j) = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_j^2)}{\sum_{k \neq j} \exp(-\|x_k - x_j\|^2 / 2\sigma_j^2)}$$

Данные величины не являются симметричными, что может добавить нам дополнительных сложностей при дальнейшей работе. Симметризуем их:

$$p_{ij} = \frac{p(i \,|\, j) + p(j \,|\, i)}{2\ell}.$$

Благодаря данному способу симметризации невозможно ситуация, в которой для некоторого отдалённого объекта x_i все близости p_{ij} будут близки к нулю — можно показать, что всегда выполнено $\sum_j p_{ij} > \frac{1}{2\ell}$.

Перейдём теперь к измерению сходства в новом низкоразмерном пространстве. Известно, что в пространствах высокой размерности можно разместить объекты так, что их попарные расстояния будут близки — а вот сохранить это свойство в низкоразмерном пространстве вряд ли возможно. Поэтому будем измерять сходства между объектами с помощью распределения Коши, которое имеет тяжёлые хвосты и не так сильно штрафует за увеличение расстояний между объектами:

$$q_{ij} = \frac{(1 + \|z_i - z_j\|^2)^{-1}}{\sum_{k \neq m}(1 + \|z_k - z_m\|^2)^{-1}}$$

Теперь мы умеем измерять расстояния между объектами как в исходном, так и в новом пространствах, и осталось лишь задать функционал ошибки проецирования. Будем измерять ошибку с помощью дивергенции Кульбака-Лейблера, которая часто используется для измерения расстояний между распределениями:

$$\mathrm{KL}(p \,||\, q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \to \min_{z_1, \ldots, z_\ell}$$

Решать данную задачу оптимизации можно, как всегда, с помощью стохастического градиентного спуска.

3 Обучение представлений

Ещё одной разновидностью задач обучения без учителя является *обучение представлений (representation learning)*, которое состоит в построении некоторых числовых представлений исходных объектов с сохранением свойств этих объектов.

Мы уже сталкивались с этой областью — выходы одного из последних слоёв свёрточной сети являются представлениями изображения, и их можно использовать как признаки при решении той или иной задачи. В данном разделе мы разберём *word2vec*, способ обучения представлений для слов.

В лингвистике существует дистрибутивная гипотеза, согласно которой слова, встречающиеся в похожих контекстах, имеют похожие смыслы. Будем строить представления для слов, опираясь на эту гипотезу: чем в более похожих контекстах встречаются два слова, тем ближе должны быть соответствующие им векторы.

Итак, мы хотим для каждого слова w из словаря W найти вектор $\vec{w} \in \mathbb{R}^d$. Пусть дан некоторый текст $x = (w_1 \ldots w_n)$. Контекстом слова w_j будем называть слова, находящиеся от него на расстоянии не более K — то есть слова $w_{j-K}, \ldots, w_{j-1}, w_{j+1}, \ldots, w_{j+K}$. Определим через векторы слов вероятность встретить слово w_i в контексте слова w_j:

$$p(w_i \,|\, w_j) = \frac{\exp(\langle \vec{w}_i, \vec{w}_j \rangle)}{\sum_{w \in W} \exp(\langle \vec{w}, \vec{w}_j \rangle)}$$

Тогда для выборки текстов $X = \{x_1, \ldots, x_\ell\}$, где текст x_i имеет длину n_i, можно определить правдоподобие и максимизировать его:

$$\sum_{i=1}^{\ell} \sum_{j=1}^{n_i} \sum_{\substack{k=-K \\ k \neq 0}}^{K} \log p(\vec{w}_{j+k} \,|\, \vec{w}_j) \to \max_{\{\vec{w}\}_{w \in W}}$$

Данный функционал можно оптимизировать стохастическим градиентным спуском. В результате обучения мы получим представления для слов, которые, как показывает практика, будут обладать многими интересными свойствами — и, в том числе, близкие по смыслу слова будут иметь близкие векторы.