Урок 6

Визуализация данных

6.1. Задача визуализации

6.1.1. Связь задачи визуализации и методов понижения размерности

Этот урок посвящён визуализации данных. Зачем визуализировать данные? Ответить на этот вопрос поможет пример: пусть имеются выборка, изображённая на рисунке 6.1, и требуется понизить её размерность до одного признака. Если не знать структуры данных, то можно долго и безуспешно пытаться спроецировать объекты на прямую, теряя при этом много информации. Однако если посмотреть на визуализацию данных, то становится понятно, что чтобы сохранить структуру, необходимо спроецировать данные на красную кривую, а для этого нужно использовать нелинейные методы понижения размерности.

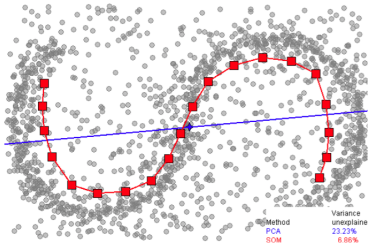

Рис. 6.1: Пример данных

В машинном обучении имеют дело с высокоразмерными выборками, то есть с выборками, в которых очень много признаков. Хочется понимать, как устроены данные, какие в них есть взаимосвязи, какие признаки важны. Для этого можно, как на рисунке 6.2, для каждой пары признаков спроецировать выборку на эту пару, а для каждого признака в отдельности построить гистограмму. Из этих графиков можно извлечь много информации: заметить, какие признаки лучше разделяют классы, или что даже какой-то один признак их хорошо разделяет.

Недостаток такого подхода — невозможность видеть всю выборку в целом. Хочется отобразить данные в двумерное или трёхмерное пространство, чтобы была видна их структура (как на рисунке 6.3): какие классы хорошо разделимы, какие — перемешаны между собой.

Итак, задача визуализации данных — это частный случай нелинейного понижения размерности, когда данные проецируются на плоскость или в трёхмерное пространство. При этом хочется спроецировать данные так, чтобы сохранить всю структуру и закономерности.

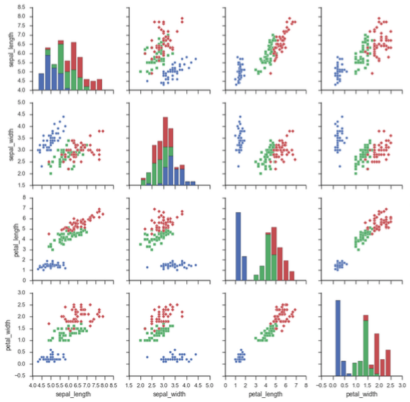

Рис. 6.2: Визуализация проекции данных на пары признаков

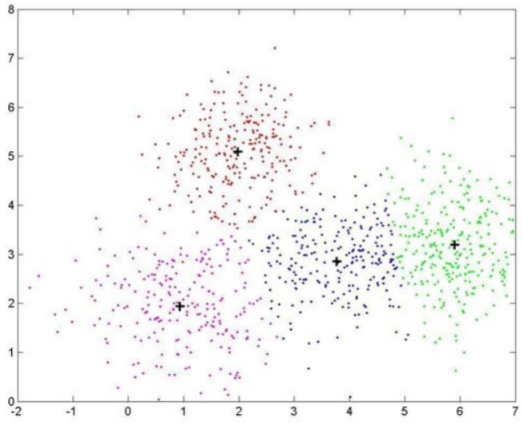

Рис. 6.3: Пример данных, отображённых в двумерное пространство

6.1.2. Поиск структуры в данных при помощи визуализации

Один из классических примеров, с помощью которого легко демонстрировать задачу визуализации — это набор данных MNIST (рисунок 6.4). Это изображения цифр, написанных от руки, причём все цифры очень разные.

Рис. 6.4: Набор данных MNIST

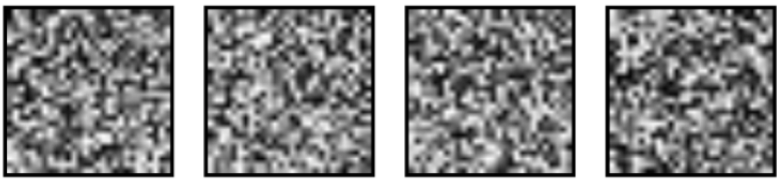

Рис. 6.5: Случайно сгенерированные изображения размера 28×28 пикселей

Требуется их разделить на 10 классов, то есть определить по изображению, какая цифра на нем нарисована. Каждая картинка в этом наборе данных имеет размер 28×28 пикселей, то есть всего у каждого объекта 784 признака. Но можно использовать намного меньше признаков, чтобы характеризовать каждую рукописную цифру. Это помогает понять следующий пример: если генерировать случайные картинки размером 28×28 пикселей, то результат будет похож на картинки, показанные на рисунке 6.5. На них нет ничего общего с изображениями цифр, и вероятность получить при случайной генерации что-то похожее на цифры очень мала. Это даёт понять, что внутренняя размерность данного набора данных сильно ниже.

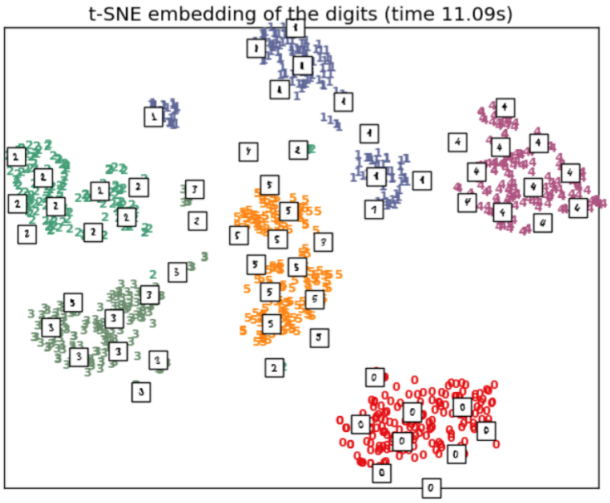

Рис. 6.6: Визуализация набора данных MNIST методом t-SNE

Если воспользоваться методом визуализации t-SNE (подробнее о нём будет сказано далее), то получится рисунок 6.6. Всего двух признаков достаточно, чтобы изобразить эти цифры так, что все классы будут идеально разделимы.

Ещё один пример взят из конкурса на сайте Kaggle. В этом конкурсе требовалось, используя описание и характеристики товара, определить, к какой из 9 категорий он относится. Всего признаков было 93. Если визуализировать этот набор данных, получится изображение, показанное на рисунке 6.7. При этом, если в исходном признаковом пространстве использовать для классификации сложные методы (случайный лес, градиентный бустинг), то видно, что в данных сохраняются те же закономерности, какие показаны на рисунке: хорошо отделимые на изображении классы разделяются хорошо, перемешанные классы — плохо.

6.2. Многомерное шкалирование

6.2.1. Постановка задачи

До этого речь уже шла о линейных методах понижения размерности, например, описывался метод случайных проекций:

$$z_{ij} = \sum_{k=1}^{D} w_{jk} x_{ik}$$

$$w_{jk} \sim \mathcal{N}(0, \frac{1}{d})$$

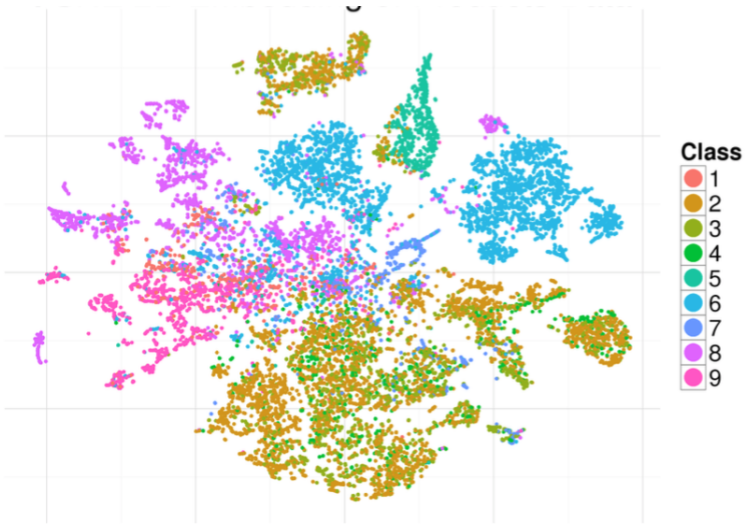

Рис. 6.7: Визуализация описаний товаров из конкурса на сайте Kaggle

Также обсуждался метод главных компонент, который выбирает веса более грамотно: он выражает их через сингулярные векторы матрицы «объекты-признаки».

Оба этих метода не могут определить нелинейные зависимости в данных. Найти их может, например, метод многомерного шкалирования (MDS). Для дальнейшего описания метода требуется ввести формальные обозначения:

- $x_1, ..., x_\ell$ — объекты в исходном пространстве;
- $\tilde{x}_1, ..., \tilde{x}_\ell$ — объекты в маломерном пространстве;
- $d_{ij} = \rho(x_i, x_j)$ — расстояния в исходном пространстве, ρ – метрика, необязательно евклидова;
- $\tilde{d}_{ij} = \|\tilde{x}_i - \tilde{x}_j\|$ — расстояния в маломерном пространстве, измеряются с использованием евклидовой метрики.

Итак, в задача многомерного шкалирования требуется, чтобы попарные расстояния между объектами изменялись как можно меньше:

$$\sum_{i<j}^{\ell} (\|\tilde{x}_i - \tilde{x}_j\| - d_{ij})^2 \to \min_{\tilde{x}_1,...,\tilde{x}_\ell}.$$

Стоит обратить внимание, что при использовании данного метода не требуется знать признаковое описание объектов, достаточно лишь уметь вычислять расстояния между ними.

6.2.2. Оптимизация метода многомерного шкалирования

Эта задача сложнее тех, которые встречались до этого, потому что в этот раз суммирование производится не по всем объектам, а по всем парам объектов. Решить задачу можно с помощью метода оптимизации SMACOF, но он довольно сложный, поэтому здесь описан не будет.

Результат многомерного шкалирования набора данных MNIST показан на рисунке 6.8. Видно, что данные неплохо разделены, однако и объекты внутри одного класса, и сами классы располагаются довольно близко друг к другу. Это результат того, что данные спроецированы из 728-мерного пространства, а чем больше размерность пространства, тем более похожи расстояния между различными парами объектов (проклятье размерности).

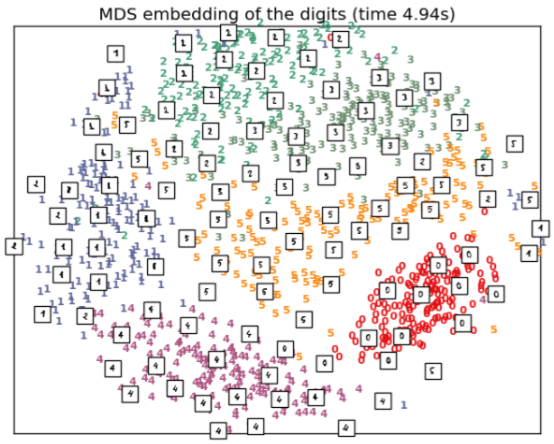

Рис. 6.8: Визуализация набора данных MNIST с помощью многомерного шкалирования

6.3. Метод t-SNE

6.3.1. Метод SNE

Решения задачи многомерного шкалирования, описанной ранее, получаются не очень качественными, потому что непросто сохранить попарные расстояния между объектами при радикальном уменьшении размерности пространства. Метод SNE (Stochastic Neighbor Embedding) пытается решить эту проблему: требуется не сохранение расстояний, а сохранение пропорций расстояний между объектами. Если

$$\rho(x_i, x_j) = \alpha\rho(x_i, x_k),$$

то

$$\rho(\tilde{x}_i, \tilde{x}_j) = \alpha\rho(\tilde{x}_i, \tilde{x}_k).$$

Чтобы формализовать эти понятия, необходимо ввести следующие условные вероятности:

$$p(x_j|x_i) = \frac{\exp\left(||x_i - x_j||^2/2\sigma^2\right)}{\sum\limits_{k\neq i}\exp\left(||x_i - x_k||^2/2\sigma^2\right)}$$

$$q(\tilde{x}_j|\tilde{x}_i) = \frac{\exp\left(||\tilde{x}_i - \tilde{x}_j||^2/2\sigma^2\right)}{\sum\limits_{k\neq i}\exp\left(||\tilde{x}_i - \tilde{x}_k||^2/2\sigma^2\right)}.$$

Это распределение учитывает только соотношения между расстояниями. Чтобы сравнить эти два распределения, необходимо использовать функционал, который находит различия между двумя вероятностными распределениями. Для этих целей хорошо подходит дивергенция Кульбака-Лейблера:

$$p(x_j|x_i)\log\frac{p(x_j|x_i)}{q(\tilde{x}_j|\tilde{x}_i)} \to \min_{\tilde{x}_1,\ldots,\tilde{x}_\ell}$$

Для нахождения описания объекта в маломерном признаковом пространстве необходимо минимизировать этот функционал (например, методом стохастического градиентного спуска).

6.3.2. Метод t-SNE

У метода SNE есть проблема: объекты в многомерном пространстве легче разместить рядом, чем в маломерном, а используемая евклидова метрика слишком сильно штрафует за несохранение пропорций. Метод t-SNE старается решить эту проблему, используя другой способ вычисления распределений в новом пространстве:

$$q(\tilde{x}_j|\tilde{x}_i) = \frac{\left(1 + \|\tilde{x}_i - \tilde{x}_j\|^2\right)^{-1}}{\sum\limits_{k\neq i}\left(1 + \|\tilde{x}_i - \tilde{x}_k\|^2\right)^{-1}}$$

При использовании этого метода разделение оказывается гораздо более чётким. Визуализация набора данных MNIST с помощью этого метода показана на рисунке 6.6.