Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Волгоградский государственный технический университет»
Факультет электроники и вычислительной техники
Кафедра «САПР и ПК»

«Машинное обучение»

Методическое пособие

Волгоград, 2016

Оглавление

1. Основные понятия и термины

1.1. Машинное обучение

Теория машинного обучения зародилась практически одновременно с появлением первых компьютеров и на протяжении последних 70 лет является активно развивающейся дисциплиной. Ее постоянное развитие вызвано ростом возможностей современных вычислительных систем, еще более стремительным ростом объемов данных, доступных для анализа, а также постоянным расширением области применения методов машинного обучения на все более широкий класс задач обработки данных.

Определение машинного обучения, данное Томом М. Митчеллом:

> машина *обучается* на основе опыта E по отношению к некоторому классу задач T и меры качества P, если качество решения задач из класса T, измеренное на основе P, улучшается с приобретением опыта E.

Различают два типа обучения. Обучение *по прецедентам*, или *индуктивное* обучение, основано на выявлении общих закономерностей по частным эмпирическим данным. *Дедуктивное* обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний. Дедуктивное обучение принято относить к области экспертных систем, поэтому обычно под машинным обучением понимают обучение по прецедентам.

1.2. Обучение по прецедентам

Пусть задано множество *объектов* X, множество *допустимых ответов* Y, и существует *целевая функция* $y^*\colon X \to Y$, значения которой $y_i = y_i^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \ldots, x_\ell\} \subset X$. Пары «объект–ответ» (x_i, y_i) называются *прецедентами*. Совокупность пар $X^\ell = (x_i, y_i)_{i=1}^\ell$ называется *обучающей выборкой*.

Задача обучения по прецедентам заключается в том, чтобы по выборке X^ℓ восстановить зависимость y^*, то есть построить *решающую функцию* $a\colon X \to Y$, которая бы приближала целевую функцию $y^*(x)$, причем не только на обучающей выборке, но и на всем множестве X.

Решающая функция a должна допускать эффективную компьютерную реализацию, по этой причине ее также называют *алгоритмом*.

Объекты обладают различными характеристиками. Результат измерения некоторой характеристики называется *признаком* f объекта. Формально, признаком считается отображение $f\colon X \to D_f$, где D_f — множество допустимых значений признака. В зависимости от природы множества D_f признаки делятся на несколько типов:

- если $D_f = \{0, 1\}$, то f — бинарный признак;
- если D_f — конечное множество, то f — номинальный признак;
- если D_f — конечное упорядоченное множество, то f — порядковый признак;
- если $D_f = \mathbb{R}$, то f — количественный признак.

Если все признаки имеют одинаковый тип, то исходные данные называются *однородными*, в противном случае — *разнородными*. Совокупность всех признаков объекта называется *описанием* объекта.

Описание объекта обычно записывается в виде матрицы:

$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(\ell)} & x_2^{(\ell)} & \cdots & x_D^{(\ell)} \end{pmatrix}; \; Y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(\ell)} \end{pmatrix}; \; (X \mid Y) = \left(\begin{array}{cccc|c} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} & y^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(\ell)} & x_2^{(\ell)} & \cdots & x_D^{(\ell)} & y^{(\ell)} \end{array}\right).$$

Моделью алгоритмов называется параметрическое семейство отображений $A = \{g(x, \theta) \mid \theta \in \Theta\}$, где $g\colon X \times \Theta \to Y$ — некоторая фиксированная функция, Θ — множество допустимых значений параметра θ, называемое *пространством параметров*.

Процесс подбора оптимального параметра модели θ по обучающей выборке X^ℓ называют настройкой или *обучением* алгоритма $a \in A$.

Метод обучения — это отображение $\mu\colon (X \times Y)^\ell \to A$, которое произвольной конечной выборке $X^\ell = (x_i, y_i)_{i=1}^\ell$ ставит в соответствие некоторый алгоритм $a \in A$. Говорят также, что метод μ *строит* алгоритм a по выборке X^ℓ. Метод обучения должен допускать эффективную программную реализацию.

Видно, что задача обучения по прецедентам делится на два этапа:

этап *обучения* — метод μ по выборке X^ℓ строит алгоритм $a = \mu(X^\ell)$.

этап *применения* — алгоритм a для новых объектов x выдает ответы $y = a(x)$.

Этап обучения наиболее сложен. Как правило, он сводится к поиску параметров моделт, доставляющих оптимальное значение заданному функционалу качества.

Функционал качества алгоритма a на выборке X^ℓ:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(a, x_i), \tag{1}$$

где $\mathcal{L}(a, x)$ — *функция потерь* — неотрицательная функция, характеризующая величину ошибки алгоритма a на объекте x.

Если функция потерь принимает только значения 0 (ответ $a(x)$ корректен) и 1 (ответ $a(x)$ ошибочен), то она называется бинарной, а функционал Q называется *частотой ошибок* алгоритма a на выборке X^ℓ. Примером такой функции потерь может служить индикатор ошибки: $\mathcal{L}(a, x) = \big[a(x) \neq y^*(x)\big]$, где квадратные скобки обозначают перевод логического значения в число ($[\textit{ложь}] = 0$, $[\textit{истина}] = 1$).

Если функция потерь выражается как отклонение от правильного ответа $\mathcal{L}(a, x) = |a(x) - y^*(x)|$, то функционал Q называется *средней ошибкой* алгоритма a на выборке X^ℓ.

Если функция потерь выражается как квадрат отклонения от правильного ответа $\mathcal{L}(a, x) = \big(a(x) - y^*(x)\big)^2$, то функционал Q называется *средней квадратичной ошибкой* алгоритма a на выборке X^ℓ.

Классический метод обучения, называемый *минимизацией эмпирического риска*, заключается в том, чтобы найти в заданной модели A алгоритм a, доставляющий минимальное значение функционалу качества Q на заданной обучающей выборке X^ℓ:

$$\mu\left(X^\ell\right) = \arg\min_{a \in A} Q\left(a, X^\ell\right). \tag{2}$$

1.3. Примеры задач машинного обучения

Пример 1. Имеется база данных о клиентах туристического агентства с информацией о возрасте и доходе за месяц. Есть рекламный материал

двух видов: более дорогой и комфортный отдых и более дешевый, молодежный отдых. Соответственно, определены два класса клиентов: класс 1 — те клиенты, которые уже выбирали для себя более дорогой отдых, и класс 2 — те, что выбирали молодежный вариант. Необходимо определить, к какому классу принадлежит новый клиент и какой из двух видов рекламных материалов ему стоит отсылать.

Пример 2. Имеется база данных о доходах торговой сети и о товарах, продаваемых ею, за определенный период времени. О товарах известна их цена и объем продаж за период. Для повышения доходов торговой сети за следующий период времени, требуется определить: влияние цены товара на его спрос, товары, продажи которых сильно влияют на общий доход торговой сети, и товары, спрос на которые довольно низок и они мало влияют на доход торговой сети. Затем необходимо сделать прогноз на доход за следующий период времени, если произвести корректировку каталога товаров, продаваемых торговой сетью, и цен на них.

Пример 3. Имеется база данных о количестве жителей в домах города. Необходимо определить в каких местах оптимальнее всего размещать остановки общественного транспорта.

2. Технология применения

2.1. Вероятностная постановка задачи обучения

В задачах обучения по прецедентам элементы множества X — это не реальные объекты, а лишь доступные данные о них. Данные могут быть неточными, поскольку измерения значений признаков $f_j(x)$ и целевой зависимости $y^*(x)$ обычно выполняются с погрешностями. Данные могут быть неполными, поскольку измеряются не все мыслимые признаки, а лишь физически доступные для измерения. В результате одному и тому же описанию x могут соответствовать различные объекты и различные ответы. В таком случае $y^*(x)$, строго говоря, не является функцией. Устранить эту некорректность позволяет вероятностная постановка задачи.

Вместо существования неизвестной целевой зависимости $y^*(x)$ предположим существование неизвестного вероятностного распределения на множестве $X \times Y$ с плотностью $p(x, y)$, из которого случайно и независимо выбираются ℓ наблюдений $X^\ell = (x_i, y_i)_{i=1}^\ell$. Такие выборки называются простыми или случайными одинаково распределенными.

Вероятностная постановка задачи считается более общей, так как функциональную зависимость $y^*(x)$ можно представить в виде вероятностного распределения $p(x, y) = p(x)p(y|x)$, положив $p(y|x) = \dot{(}y - y^*(x))$, где $\dot{(}z)$ — дельта-функция.

2.1.1. Принцип максимума правдоподобия

При вероятностной постановке задачи вместо модели алгоритмов $g(x, \theta)$, аппроксимирующей неизвестную зависимость $y^*(x)$, задается модель совместной плотности распределения объектов и ответов $\varphi(x, y, \theta)$, аппроксимирующая неизвестную плотность $p(x, y)$. Затем определяется значение параметра θ, при котором выборка данных X^ℓ максимально правдоподобна, то есть наилучшим образом согласуется с моделью плотности.

Если наблюдения в выборке X^ℓ независимы, то совместная плотность распределения всех наблюдений равна произведению плотностей $p(x, y)$ в каждом наблюдении: $p(X^\ell) = p\big((x_1, y_1), \ldots, (x_\ell, y_\ell)\big) = p(x_1, y_1) \cdot \ldots \cdot p(x_\ell, y_\ell)$. Подставляя вместо $y^*(x)$ модель плотности $\varphi(x, y, \theta)$, получаем функцию правдоподобия:

$$L(\theta, X^\ell) = \prod_{i=1}^{\ell} \varphi\left(x_i, y_i, \theta\right). \tag{3}$$

Чем выше значение правдоподобия, тем лучше выборка согласуется с моделью. Значит, нужно искать значение параметра θ, при котором значение (3) максимально. В математической статистике это называется принципом максимума правдоподобия.

После того, как значение параметра θ найдено, искомый алгоритм $a_\theta(x)$ несложно строится по плотности $\varphi(x, y, \theta)$.

2.1.2. Связь максимизации правдоподобия с минимизацией эмпирического риска

Вместо максимизации L удобнее минимизировать функционал $-\ln L$, поскольку он аддитивен (имеет вид суммы) по объектам выборки:

$$-\ln L\left(\theta, X^{\ell}\right) = -\sum_{i=1}^{\ell} \ln \varphi\left(x_i, y_i, \theta\right) \to \min_{\theta}. \tag{4}$$

Этот функционал совпадает с функционалом эмпирического риска (1), если определить вероятностную функцию потерь $\mathcal{L}\left(a_\theta, x\right) = -\ell \ln \varphi(x, y, \theta)$. Такое определение потери вполне естественно — чем хуже пара (x_i, y_i) согласуется с моделью φ, тем меньше значение плотности $\varphi(x_i, y_i, \theta)$ и выше величина потери $\mathcal{L}(a_\theta, x)$.

Верно и обратное — для многих функций потерь возможно подобрать модель плотности $\varphi(x, y, \theta)$ таким образом, чтобы минимизация эмпирического риска была эквивалентна максимизации правдоподобия.

2.1.3. Проблема переобучения и понятие обобщающей способности

Если минимум функционала $Q(a, X^{\ell})$ достигается на алгоритме a, то это еще не гарантирует, что a будет хорошо приближать целевую зависимость на произвольной контрольной выборке $X^k = (x'_i, y'_i)_{i=1}^k$.

Когда качество работы алгоритма на новых объектах, не вошедших в состав обучения, оказывается существенно хуже, чем на обучающей выборке, говорят об эффекте переобучения или переподгонки. При решении практических задач с этим явлением приходится сталкиваться очень часто.

Легко представить себе метод, который минимизирует эмпирический риск до нуля, но при этом абсолютно не способен обучаться. Получив обучающую выборку X^{ℓ}, он запоминает ее и строит алгоритм, который сравнивает предъявляемый объект x с обучающими объектами x_i из X^{ℓ}. В случае совпадения $x = x_i$ алгоритм выдает правильный ответ y_i. Иначе выдается любой другой ответ. Эмпирический риск принимает наименьшее возможное значение, равное нулю. Однако этот алгоритм не способен восстановить зависимость вне материала обучения. Отсюда вывод: для успешного обучения необходимо не только запоминать, но и обобщать.

Обобщающая способность метода μ характеризуется величиной $Q\left(\mu\left(X^{\ell}\right), X^{k}\right)$ при условии, что выборки X^{ℓ} и X^{k} являются представительными. Для формализации понятия «представительная выборка» обычно принимается стандартное предположение, что выборки X^{ℓ} и X^{k} — простые, полученные из одного и того же неизвестного вероятностного распределения на множестве X.

2.2. Этапы разработки алгоритмов машинного обучения

1. Сбор данных. Существует бесчисленное множество способов сбора данных. Также можно использовать публично доступные данные.
2. Подготовка входных данных. Необходимо убедиться что они находятся в пригодном для использования виде.
3. Анализ входных данных. В данных можно увидеть некоторые закономерности. Однако, могут быть элементы, значительно отличающихся от общей массы. Возможно, в данных есть лишние и недостоверные прецеденты, которые необходимо удалить.
4. Обучение алгоритма. На этом этапе машина проходит обучение. Алгоритм обрабатывает данные, полученные на 1–3 этапах, и выдает информацию или знания. В случае использования обучения без учителя этот шаг пропускается.
5. Тестирование алгоритма. На этом шаге находит применение информация, полученная на предыдущем этапе. Во время тестирования алгоритм проходит проверку — насколько хорошо он справляется со своей задачей. В случае обучения с учителем есть некоторые известные значения, которые можно использовать для проверки. В случае обучения без учителя, придется использовать другие методики тестирования качества работы алгоритма. Если алгоритм не проходит проверку, то происходит возврат к шагу 4 для коррекции метода обучения. Если проблема кроется в данных, то происходит возврат к шагу 1.
6. Использование результата. На этом шаге на основе алгоритма разрабатывается работающая программа.

3. Задачи машинного обучения

Общая постановка задачи обучения по прецедентам:

дано конечное множество прецедентов, по каждому из которых измерены некоторые данные. Требуется по этим частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие не только этой конкретной выборке, но вообще всем прецедентам, в том числе тем, которые еще не наблюдались.

Для решения задачи обучения по прецедентам в первую очередь фиксируется модель восстанавливаемой зависимости. Затем вводится функционал качества, значение которого показывает, насколько хорошо модель описывает наблюдаемые данные. Алгоритм обучения ищет такой набор параметров модели, при котором функционал качества на заданной обучающей выборке принимает оптимальное значение. Процесс настройки модели по выборке данных в большинстве случаев сводится к применению численных методов оптимизации.

В основном все стандартные задачи можно разделить на три типа:

- обучение *с учителем*;
- обучение *без учителя*;
- *частичное* обучение.

3.1. Задачи обучения с учителем

В задачах обучения с учителем каждый прецедент представляет собой стандартную пару «объект–ответ». Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Функционал качества в таких задачах обычно определяется как средняя ошибка ответов, по всем объектам выборки.

Среди задач обучения с учителем можно выделить отдельные подтипы задач:

- задачи *классификации*. Они отличаются тем, что множество допустимых ответов конечно. Их называют метками классов, а классом называют множество всех объектов с данным значением метки.

Дано некоторое множество объектов X и конечное множество номеров классов Y. Известно, что существует неизвестная целевая зависимость $f^*\colon X \to Y$, значения которой известны только на объектах обучающей выборки. Задача заключается в построении алгоритма (*классификатора*), способного классифицировать произвольный объект $x \in X$.

В задаче классификации могут быть следующие типы входных данных:

1. признаковые описания (каждый объект описывается каким-либо набором признаков — характеристик);
2. матрица расстояний (для каждого объекта известно расстояние от него до всех других объектов);
3. временной ряд (последовательность измерений во времени).

Любые сложные входные данные, такие как графы, изображения, видеозаписи, запросы к БД и т. п. приводятся к перечисленным трем.

В качестве реальных примеров задачи классификации можно привести:

- распознавание образов (лиц). Классы — личности.
- Распознавание рукописного текста. Классы — символы.
- Определение спама в электронной почте. Классы — спам, не спам.
- Кредитный скоринг (в упрощенном варианте). Классы — платежеспособен, не платежеспособен.

- Задачи *восстановления регрессии* отличаются тем, что допустимым ответом является действительное число или числовой вектор.

  Дано некоторое множество объектов X и множество номеров классов Y. Причем $|Y| = |\mathbb{R}|$, где $\mathbb{R}$ — множество действительных чисел. Известно, что существует неизвестная целевая зависимость $f^*\colon X \to Y$, значения которой известны только на объектах обучающей выборки. Задача заключается в построении алгоритма, способного классифицировать произвольный объект $x \in X$.

Можно с некоторым допущением сказать, что задача восстановления регрессии это задача классификации с множеством классов $\mathbb{R}$. Очень многие задачи классификации несложным образом модифицируются под восстановление регрессии.

В качестве реальных примеров задачи восстановления регрессии можно привести:

- кредитный скоринг. Оценка максимальной суммы кредита.
- Продажи. Оценка объемов продаж.

- Задачи *прогнозирования* отличаются тем, что объектами являются отрезки временных рядов, обрывающиеся в тот момент, когда требуется сделать прогноз на будущее.

  Дано множество X, являющееся временным рядом (т. е. множество некоторых значений функции во времени). Задача заключается в нахождении значений функции за пределами данных, имеющихся в X.

  Задача прогнозирования может решаться методами решения задач классификации и восстановления регрессии и является самой популярной задачей машинного обучения.

  В качестве реальных примеров задачи прогнозирования можно привести:

  - сейсмопредсказания. Прогнозирование времени следующего землетрясения на определенной территории.
  - Изменение стоимости. Прогнозирование стоимости какого-либо продукта в определенный промежуток времени.
  - Нагрузка на call-центр. Прогнозирование количества телефонных звонков клиентов компании.

3.2. Задачи обучения без учителя

В задачах обучения без учителя ответы не задаются, и требуется искать зависимости между объектами.

Среди задач обучения без учителя можно выделить отдельные подтипы задач:

- задачи *кластеризации*. Их суть заключается в том, чтобы сгруппировать объекты в кластеры, используя данные о попарном сходстве объектов.

  > Дано некоторое множество объектов X. Задача заключается в построении алгоритма, способного относить произвольный объект $x \in X$ к некоторому кластеру. Кластером мы будем называть аналог класса в задаче классификации с той разницей, что кластеры изначально неизвестны.

  Кластеры, как правило, обладают следующими свойствами:

  1. непересекаемость — никакие два кластера не имеют общих элементов;
  2. конечность каждого кластера;
  3. связность — никакой кластер невозможно разбить на два непустых непересекающихся открытых подмножества.

  В качестве реальных примеров задачи кластеризации можно привести:

  – группировка текстов. Кластеры — тематики текстов.
  – Разделение людей по психотипу. Кластеры — психотипы.
  – Любая задача классификации без обучающей выборки.

- Задачи *поиска ассоциативных правил*. В них данные представляются в виде признаковых описаний, а найти требуется такие наборы признаков, и такие значения этих признаков, которые в описаниях встречаются особенно часто.

  > Дано некоторое множество объектов X. Задача заключается в построении алгоритма, способного находить взаимосвязи между элементами X.

  В качестве реальных примеров задачи поиска ассоциативных правил можно привести:

  – анализ рыночных корзин. Поиск наиболее типичных шаблонов покупок в супермаркетах.

– Анализ поведения пользователя. Отличающиеся от большинства элементы будут подозреваемыми во вторжении.

- Задачи *фильтрации выбросов* чем-то схожи с задачами поиска ассоциативных правил, только в них необходимо найти нетипичные объекты, имеющие признаки или комбинации признаков, встречающихся реже других;
- задачи *построения доверительной области* — области минимального объема с гладкой границей, содержащей заданную долю выборки;
- задачи *сокращения размерности*. Их суть заключается в том, чтобы по исходным признакам с помощью некоторых функций преобразования перейти к наименьшему числу новых признаков, не потеряв при этом никакой существенной информации об объектах.

  > Дано некоторое множество объектов X, представляющее собой декартово произведение $Y \times Y \times \ldots \times Y$. Задача заключается в сокращении размерности множества X с минимальной потерей качества данных.

  Практическая польза данной задачи очевидна. С данными больших размерностей трудно работать. Решение задачи позволяет повысить эффективность работы с ними.
- Задачи *заполнения пропущенных значений* — замена недостающих признаков у объектов их прогнозными значениями;
- В задачи *ранжирования* ответы надо получить сразу на множестве объектов, после чего отсортировать их по значениям ответов.

  > Дана пара «запрос–объект». Задача заключается в определении релевантности объекта запросу.

  Обычно, задача ранжирования решается статистически, но существуют и обучаемые методы.

  В качестве реальных примеров задачи ранжирования можно привести:

  – ранжирование поисковых запросов пользователя;

  – ранжирование в системах коллаборативной фильтрации (целевые рекомендации).

3.3. Другие виды обучения

Частичное обучение занимает промежуточное положение между обучением с учителем и без учителя. Каждый прецедент представляет собой пару «объект, ответ», но ответы известны только на части прецедентов.

Также есть и другие виды задач обучения, которые нельзя отнести к перечисленным трем типам задач:

- *трансдуктивное* обучение. Дана конечная обучающая выборка прецедентов. Требуется по этим частным данным сделать предсказания относительно других частных данных — тестовой выборки. В отличие от стандартной постановки, здесь не требуется выявлять общую закономерность, поскольку известно, что новых тестовых прецедентов не будет;
- обучение *с подкреплением*. Роль объектов играют пары «ситуация, принятое решение», ответами являются значения функционала качества, характеризующего правильность принятых решений. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени;
- *динамическое* обучение может быть как обучением с учителем, так и без учителя. Специфика в том, что прецеденты поступают потоком. Требуется немедленно принимать решение по каждому прецеденту и одновременно доучивать модель зависимости с учетом новых прецедентов. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени;
- *активное* обучение отличается тем, что обучаемый имеет возможность самостоятельно назначать следующий прецедент, который станет известен;
- *метаобучение* отличается тем, что прецедентами являются ранее решенные задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель — обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

Прикладные задачи классификации, регрессии, кластеризации и прогнозирования встречаются в самых разных областях человеческой деятельности, и их число постоянно растет.

4. Основные методы машинного обучения

4.1. Задачи классификации

4.1.1. «Наивный» байесовский классификатор

Байесовский подход является классическим в теории распознавания образов и лежит в основе многих методов. Он опирается на теорему о том, что если плотности распределения классов известны, то алгоритм классификации, имеющий минимальную вероятность ошибок, можно выписать в явном виде.

Итак, пусть X — множество объектов, Y — конечное множество имен классов, множество $X \times Y$ является вероятностным пространством с плотностью распределения $p(x, y) = P(y)p(x \mid y)$. Вероятности появления объектов каждого из классов $P_y = P(y)$ называются априорными вероятностями классов. Плотности распределения $p_y(x) = p(x \mid y)$ называются функциями правдоподобия классов.

Рассмотрим произвольный алгоритм a: $X \to Y$. Он разбивает множество X на непересекающиеся области $A_y = \{x \in X \mid a(x) = y\}$, $y \in Y$. Вероятность того, что появится объект класса y и алгоритм a отнесет его к классу s, равна $P_y P(A_s \mid y)$. Каждой паре $(y, s) \in Y \times Y$ поставим в соответствие величину потери λ_{ys} при отнесении объекта класса y к классу s. Обычно полагают $\lambda_{yy} = 0$, и $\lambda_{ys} > 0$ при $y \neq s$. Соотношения потерь на разных классах, как правило, известны заранее.

Функционалом среднего риска называется ожидаемая величина потери при классификации объектов алгоритмом a:

$$R(a) = \sum_{y \in Y} \sum_{s \in Y} \lambda_{ys} P_y P(A_s \mid y).$$

Если величина потерь одинакова для ошибок любого рода, $\lambda_{ys} = [y \neq s]$, то средний риск $R(a)$ совпадает с вероятностью ошибки алгоритма a.

Если известны априорные вероятности P_y и функции правдоподобия $p_y(x)$, то минимум среднего риска $R(a)$ достигается алгоритмом

$$a(x) = \arg\min_{s \in Y} \sum_{y \in Y} \lambda_{ys} P_y p_y(x). \tag{5}$$

Часто можно полагать, что величина потери зависит только от истинной классификации объекта, но не от того, к какому классу он был ошибочно отнесен. Итак, если P_y и $p_y(x)$ известны, а $\lambda_{yy} = 0$ и $\lambda_{ys} \equiv \lambda_y$ для всех y, $s \in Y$, то минимум среднего риска достигается алгоритмом

$$a(x) = \arg\max_{y \in Y} \lambda_y P_y p_y(x). \tag{6}$$

Это выражение называют байесовским решающим правилом.

Допустим, что объекты $x \in X$ описываются n числовыми признаками f_j: $X \to \mathbb{R}$, $j = 1, \ldots, n$. Обозначим через $x = (\xi_1, \ldots, \xi_n)$ произвольный элемент пространства объектов $X = \mathbb{R}^n$, где $\xi_j = f_j(x)$.

Если предположить, что признаки $f_1(x), \ldots, f_n(x)$ являются независимыми случайными величинами, то функции правоподобия классов можно представить в виде

$$p_y(x) = p_{y1}(\xi_1) \cdot \ldots \cdot p_{yn}(\xi_n), \quad y \in Y, \tag{7}$$

где $p_{yj}(\xi_j)$ — плотность распределения значений j-го признака для класса y. Это условие крайне редко выполняется на практике, поэтому алгоритмы, использующие представление (7), называются наивными байесовскими.

Подставив эмпирические оценки одномерных плотностей $\hat{p}_{yj}(\xi)$ в (7) и затем в (6), получим алгоритм

$$a(x) = \arg\max_{y \in Y} \left(\ln \lambda_y \hat{P}_y + \sum_{j=1}^{n} \ln \hat{p}_{yj}(\xi_j) \right). \tag{8}$$

Основные его преимущества — простота реализации и низкие вычислительные затраты при обучении и классификации. В тех редких случаях, когда признаки (почти) независимы, наивный байесовский классификатор (почти) оптимален.

Основной его недостаток — низкое качество классификации. Он используется либо как эталон при экспериментальном сравнении алгоритмов, либо как элементарный «строительный блок» в алгоритмических композициях.

Для более качественной классификации необходимо делать более сложные оценки плотности распределения классов $p_y(x)$.

4.1.1.1. Непараметрический подход

Непараметрические методы классификации основаны на локальном оценивании плотностей распределения классов $p_y(x)$ в окрестности классифицируемого объекта $x \in X$. Для классификации объекта x применяется основная формула (6).

Локальное оценивание опирается на само определение плотности. Простейшие одномерные оценки могут оказаться полезными на практике, в частности, при построении «наивных» байесовских классификаторов (8).

Дискретный случай. Пусть X — конечное множество, причем $|X| \ll m$. Оценкой плотности служит гистограмма значений x_i, встретившихся в выборке $X^m = (x_i)_{i=1}^m$:

$$\hat{p}(x) = \frac{1}{m} \sum_{i=1}^{m} \big[x_i = x\big]. \tag{9}$$

Одномерный непрерывный случай. Пусть $X = \mathbb{R}$. Согласно определению плотности, $p(x) = \lim_{h \to 0} \frac{1}{2h} P[x-h, x+h]$, где $P[a,b]$ — вероятностная мера отрезка $[a,b]$. Соответственно, эмпирическая оценка плотности определяется как доля точек выборки, лежащих внутри отрезка $[x-h, x+h]$, где h — неотрицательный параметр, называемый шириной окна:

$$\hat{p}_h(x) = \frac{1}{2mh} \sum_{i=1}^{m} \left[|x - x_i| < h\right]. \tag{10}$$

Функция $\hat{p}_h(x)$ является кусочно-постоянной, что приводит к появлению широких зон неуверенности, в которых максимум (6) достигается одновременно для нескольких классов $y \in Y$. Проблема решается с помощью локальной непараметрической оценки Парзена-Розенблатта:

$$\hat{p}_h(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right), \tag{11}$$

где $K(z)$ — функция, называемая ядром, четная и нормированная $\int K(z)\,dz = 1$. Функция $\hat{p}_h(x)$ обладает той же степенью гладкости, что и

ядро $K(z)$, и, благодаря нормировке, действительно может интерпретироваться как плотность вероятности: $\int \hat{p}_h(x)\,dx = 1$ при любом h.

Многомерный непрерывный случай. Пусть объекты описываются n числовыми признаками f_j: $X \to \mathbb{R}$, $j = 1, \ldots, n$. Тогда непараметрическая оценка плотности в точке $x \in X$ записывается в следующем виде:

$$\hat{p}_h(x) = \frac{1}{m} \sum_{i=1}^{m} \prod_{j=1}^{n} \frac{1}{h_j} K\left(\frac{f_i(x) - f_j(x)}{h_j}\right). \tag{12}$$

Таким образом, в каждой точке x_i многомерная плотность представляется в виде произведения одномерных плотностей.

Произвольное метрическое пространство. Пусть на X задана функция расстояния $\rho(x, x')$. Одномерная оценка Парзена-Розенблатта (10) обобщается и на этот случай:

$$\hat{p}_h(x) = \frac{1}{mV(h)} \sum_{i=1}^{m} K\left(\frac{\rho(x, x_i)}{h}\right), \tag{13}$$

где $V(h)$ — нормирующий множитель, гарантирующий, что $\hat{p}_h(x)$ действительно является плотностью. Сходимость оценки (13) доказана при некоторых дополнительных ограничениях на ядро K и метрику ρ, причем скорость сходимости лишь немного хуже, чем в одномерном случае.

4.1.1.2. Параметрический подход

В параметрическом подходе предполагается, что плотность распределения выборки $X^m = \{x_1, \ldots, x_m\}$ известна с точностью до параметра, $p(x) = \varphi(x; \theta)$, где φ — фиксированная функция. Вектор параметров θ оценивается по выборке X^m с помощью принципа максимума правдоподобия.

Нормальный дискриминантный анализ — это специальный случай байесовской классификации, когда предполагается, что плотности всех классов $p_y(x)$, $y \in Y$ являются многомерными нормальными. Этот случай интересен и удобен тем, что задача оценивания параметров распределения по выборке решается аналитически.

Многомерное нормальное распределение. Пусть $X = \mathbb{R}^n$, то есть объекты описываются n числовыми признаками. Вероятностное распределение

с плотностью

$$N(x;\mu,\Sigma) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)\right), \quad (x \in \mathbb{R}^n),$$

называется n-мерным нормальным (гауссовским) распределением с математическим ожиданием (центром) $\mu \in \mathbb{R}^n$ и ковариационной матрицей $\Sigma \in \mathbb{R}^{n\times n}$. Предполагается, что матрица Σ симметричная, невырожденная, положительно определенная.

4.1.2. Метод парзеновского окна

Запишем парзеновскую оценку плотности (13) для каждого класса $y \in Y$:

$$\hat{p}_{y,h}(x) = \frac{1}{\ell_y V(h)} \sum_{i=1}^{\ell} [y_i = y] K\left(\frac{\rho(x,x_i)}{h}\right), \tag{14}$$

где K — ядро, h — ширина окна. Если нормирующий множитель $V(h)$ не зависит от y, то в байесовском классификаторе (6) его можно убрать из-под знака $\arg\max$ и вообще не вычислять. Подставим оценку плотности (14) и оценку априорной вероятности классов $\hat{P}_y = \ell_y/\ell$ в формулу (6):

$$a(x;X^\ell,h) = \arg\max_{y\in Y} \lambda_y \sum_{i=1}^{\ell} [y_i = y] K\left(\frac{\rho(x,x_i)}{h}\right). \tag{15}$$

Выборка X^ℓ сохраняется «как есть» и играет роль параметра алгоритма. Если метрика ρ фиксирована, то обучение парзеновского классификатора (14) сводится к подбору ширины окна h и вида ядра K.

4.1.3. Линейный дискриминант Фишера

Предположим, что ковариационные матрицы классов одинаковы и равны Σ. Оценим $\hat{\Sigma}$ по всем ℓ обучающим объектам. С учётом поправки на смещённость,

$$\hat{\Sigma} = \frac{1}{\ell - |Y|} \sum_{i=1}^{\ell} (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^{\mathrm{T}}.$$

Запишем подстановочный алгоритм:

$$\begin{aligned} a(x) &= \arg\max_{y\in Y}(\lambda_y P_y p_y(x)) \\ &= \arg\max_{y\in Y}\left(\ln(\lambda_y P_y) - \frac{1}{2}\hat{\mu}_y^{\mathrm{T}}\hat{\Sigma}^{-1}\hat{\mu}_y + x^{\mathrm{T}}\hat{\Sigma}^{-1}\hat{\mu}_y\right) \\ &= \arg\max_{y\in Y}\left(x^{\mathrm{T}}\alpha_y + \beta_y\right). \end{aligned} \tag{16}$$

Этот алгоритм называется линейным дискриминантом Фишера. Он неплохо работает, когда формы классов действительно близки к нормальным и не слишком сильно различаются. В этом случае линейное решающее правило близко к оптимальному байесовскому, но существенно более устойчиво, чем квадратичное, и часто обладает лучшей обобщающей способностью.

4.1.4. EM–алгоритм

Идея алгоритма заключается в следующем. Искусственно вводится вспомогательный вектор скрытых переменных G, обладающий двумя замечательными свойствами. С одной стороны, он может быть вычислен, если известны значения вектора параметров Θ. С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

EM-алгоритм состоит из итерационного повторения двух шагов. На E-шаге вычисляется ожидаемое значение вектора скрытых переменных G по текущему приближению вектора параметров Θ. На M-шаге решается задача максимизации правдоподобия и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ.

1: Вычислить начальное приближение вектора параметров Θ;
2: повторять
3: G `:= EStep(`Θ`);`
4: Θ `:= MStep(`Θ`,` G`);`
5: пока Θ и G не стабилизируются.

Область применения этого алгоритма чрезвычайно широка — дискриминантный анализ, кластеризация, восстановление пропусков в данных, обработка сигналов и изображений.

4.1.4.1. E-шаг

Обозначим через $p(x, \theta_j)$ плотность вероятности того, что объект x принадлежит j-му классу. По формуле условной вероятности

$$p(x, \theta_j) = p(x)P(\theta_j \,|\, x) = w_j p_j(x).$$

Введём обозначение $g_{ij} \equiv P(\theta_j \,|\, x_i)$. Это неизвестная апостериорная вероятность того, что обучающий объект x_i принадлежит j-му классу. Возьмём эти величины в качестве скрытых переменных. Обозначим $G = (g_{ij})_{m \times k} = (g_1, \ldots, g_j)$, где g_j — j-й столбец матрицы G. Каждый объект обязательно принадлежит какому-то классу, поэтому справедлива формула полной вероятности:

$$\sum_{j=1}^{k} g_{ij} = 1 \quad \text{для всех } i = 1, \ldots, \ell.$$

Зная параметры классов w_j, θ_j, легко вычислить g_{ij} по формуле Байеса:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \quad \text{для всех } i, j. \tag{17}$$

В этом и заключается E-шаг алгоритма EM.

4.1.4.2. M-шаг

M-шаг сводится к вычислению весов классов w_j как средних арифметических:

$$w_j = \frac{1}{m} \sum_{i=1}^{m} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} = \frac{1}{m} \sum_{i=1}^{m} g_{ij}, \quad j = 1, \ldots, k \tag{18}$$

и оцениванию параметров классов θ_j путём решения k независимых оптимизационных задач:

$$\theta_j = \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i; \theta), \quad j = 1, \ldots, k. \tag{19}$$

EM-алгоритм с фиксированным числом классов

Вход:

выборка $X^m = \{x_1, \ldots, x_m\}$;

k — число классов;

$\Theta = (w_j, \theta_j)_{j=1}^k$ — начальное приближение параметров;

δ — параметр критерия останова;

Выход:

$\Theta^* = (w_j, \theta_j)_{j=1}^k$ — оптимизированный вектор параметров.

повторять:

E-шаг:

для всех $i = 1, \ldots, m$, $j = 1, \ldots, k$

$$g_{ij}^0 = g_{ij}; \quad g_{ij} = \frac{w_j \varphi(x_i; \theta_j}{\sum_{s=1}^k w_s \varphi(x_i; \theta_s)};$$

M-шаг:

для всех $j = 1, \ldots, k$

$$\theta_j = \arg\max_\theta \sum_{i=1}^m g_{ij} \ln \varphi(x_i; \theta); \quad w_j = \frac{1}{m} \sum_{i=1}^m g_{ij};$$

пока $\max_{i,j} \left| g_{ij} - g_{ij}^0 \right| > \delta$;

вернуть $(w_j, \theta_j)_{j=1}^k$.

4.1.5. Метод ближайшего соседа

Пусть на множестве объектов X задана функция расстояния ρ: $X \times X \to [0, \infty)$. Существует целевая зависимость y^*: $X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y_i^*(x_i)$. Множество классов Y конечно. Требуется построить классификатор a: $X \to Y$, аппроксимирующий целевую зависимость $y^*(x)$ на всём множестве X.

Алгоритм ближайшего соседа относит классифицируемый объект $u \in X^\ell$ к тому классу, которому принадлежит ближайший обучающий объект:

$$w(i, u) = [i = 1]; \quad a(u; X^\ell) = y_u^{(1)}.$$

Этот алгоритм является, по всей видимости, самым простым классификатором.

Обучение алгоритма сводится к запоминанию выборки X^ℓ. Единственное достоинство этого алгоритма — простота реализации. Недостатков гораздо больше:

- Неустойчивость к погрешностям. Если среди обучающих объектов есть выброс — объект, находящийся в окружении объектов чужого класса, то не только он сам будет классифицирован неверно, но и те окружающие его объекты, для которых он окажется ближайшим.
- Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ.
- В результате — низкое качество классификации.

Чтобы сгладить влияние выбросов, будем относить объект u к тому классу, элементов которого окажется больше среди k ближайших соседей $x_u^{(i)}$, $i = 1, \ldots, k$:

$$w(i,u) = [i \leqslant k]; \quad a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y].$$

При $k = 1$ этот алгоритм совпадает с предыдущим, следовательно, неустойчив к шуму. При $k = \ell$, наоборот, он чрезмерно устойчив и вырождается в константу. Таким образом, крайние значения k нежелательны.

4.1.6. Метод опорных векторов

Метод опорных векторов основан на концепции гиперплоскостей, которые определяют границы гиперповерхностей. Разделяющая гиперплоскость — это гиперплоскость, которая отделяет группу объектов, имеющих различную классовую принадлежность. Метод опорных векторов обладает несколькими замечательными свойствами. Во-первых, обучение метода сводится к задаче квадратичного программирования, имеющей единственное решение, которое вычисляется достаточно эффективно даже на выборках в сотни тысяч объектов. Во-вторых, решение обладает свойством разреженности: положение оптимальной разделяющей гиперплоскости зависит лишь от небольшой доли обучающих объектов. Они называются опорными векторами; остальные объекты фактически не задействуются. На случай нелинейных разделяющих поверхностей метод обобщается введением функции ядра.

Рассмотрим задачу классификации на два непересекающихся класса, в которой объекты описываются n-мерными вещественными векторами: $X = \mathbb{R}^n$, $Y = \{-1, +1\}$. Будемстроить линейный пороговый классификатор:

$$a(x) = \operatorname{sign}\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = \operatorname{sign}\Big(\big\langle w, x\big\rangle - w_0\Big),$$

где $x = (x^1, \ldots, x^n)$ — признаковое описание объекта x; вектор $w = (w^1, \ldots, w^n) \in \mathbb{R}^n$ и скалярный порог $w_0 \in \mathbb{R}$ являются параметрами алгоритма. Уравнение $\big\langle w, x\big\rangle = w_0$ описывает гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^n$. Пусть выборка $X^\ell = (x_i, y_i)_{i=1}^\ell$ линейно разделима и существуют значения w, w_0, при которых функционал числа ошибок

$$Q(w, w_0) = \sum_{i=1}^{\ell} \Big[y_1\Big(\big\langle w, x\big\rangle - w_0\Big) \leqslant 0\Big]$$

принимает нулевое значение. Потребуем,чтобы разделяющая гиперплоскость максимально далеко отстояла от ближайших к ней точек обоих классов.

Параметры линейного порогового классификатора определены с точностью до нормировки: алгоритм $a(x)$ не изменится, если w и w_0 одновременно умножить на одну и ту же положительную константу. Удобно выбрать эту константу таким образом, чтобы выполнялось условие

$$\min_{i=1,\ldots,\ell} y_1\Big(\big\langle w, x_i\big\rangle - w_0\Big) = 1. \tag{20}$$

Множество точек $\Big\{x\colon\ -1 \leqslant \big\langle w, x_i\big\rangle - w_0 \leqslant 1\Big\}$ описывает полосу, разделяющую классы. Ни один из объектов обучающей выборки не попадает внутрь этой полосы. Границами полосы служат две параллельные гиперплоскости с вектором нормали w. Разделяющая гиперплоскость проходит ровно посередине между ними. Объекты, ближайшие к разделяющей гиперплоскости, лежат на границах полосы, и именно на них достигается минимум (20).

4.2. Задачи регрессии

Задачу обучения по прецедентам при $Y = \mathbb{R}$ принято называть задачей восстановления регрессии. Задано пространство объектов X и множество возможных ответов Y. Существует неизвестная целевая зависимость y^*: $X \to Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Требуется построить алгоритм, который в данной задаче принято называть «функцией регрессии» a: $X \to Y$, аппроксимирующий целевую зависимость y^*.

4.2.1. Метод наименьших квадратов

Пусть задана модель регрессии — параметрическое семейство функций $g(x, \alpha)$, где $\alpha \in \mathbb{R}^p$ — вектор параметров модели. Определим функционал качества аппроксимации целевой зависимости на выборке X^ℓ как сумму квадратов ошибок:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i)^2. \tag{21}$$

Обучение по методу наименьших квадратов состоит в том, чтобы найти вектор параметров α^*, при котором достигается минимум среднего квадрата ошибки на заданной обучающей выборке X^ℓ:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^p} Q(\alpha, X^\ell). \tag{22}$$

Стандартный способ решения этой оптимизационной задачи — воспользоваться необходимым условием минимума. Если функция $g(x, \alpha)$ достаточное число раз дифференцируема по α, то в точке минимума выполняется система p уравнений относительно p неизвестных:

$$\frac{\partial Q}{\partial \alpha}(\alpha, X^\ell) = 2 \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i) \frac{\partial g}{\partial \alpha}(x_i, \alpha) = 0. \tag{23}$$

4.2.2. Метод ядерного сглаживания

Возьмём самую простую модель регрессии, какая только возможна — константу $g(x, \alpha) = \alpha$, $\alpha \in \mathbb{R}$. Но при этом, чтобы не получить тривиального

решения, введём веса объектов $w_i(x)$, зависящие от того объекта x, в котором мы собираемся вычислять значение $a(x) = g(x, \alpha)$.

Чтобы вычислить значение $a(x) = \alpha$ для произвольного $x \in X$, воспользуемся методом наименьших квадратов (21):

$$Q(\alpha; X^\ell) = \sum_{i=1}^{\ell} w_i(x)(\alpha - y_i)^2 \to \min_{\alpha \in \mathbb{R}}.$$

Зададим веса w_i обучающих объектов так, чтобы они убывали по мере увеличения расстояния $\rho(x, x_i)$. Для этого введём невозрастающую, гладкую, ограниченную функцию K, называемую ядром:

$$w_i(x) = K\left(\frac{\rho(x, x_i)}{h}\right).$$

Параметр h называется шириной ядра. Чем меньше h, тем быстрее будут убывать веса $w_i(x)$ по мере удаления x_i от x.

Приравняв к нулю производную $\dfrac{\partial Q}{\partial \alpha} = 0$, получим формулу ядерного сглаживания:

$$\alpha_h(x; X^\ell) = \frac{\sum_{i=1}^{\ell} y_i w_i(x)}{\sum_{i=1}^{\ell} w_i(x)} = \frac{\sum_{i=1}^{\ell} y_i K\left(\frac{\rho(x,x_i)}{h}\right)}{\sum_{i=1}^{\ell} K\left(\frac{\rho(x,x_i)}{h}\right)}. \tag{24}$$

Ядерное сглаживание — это довольно простой метод с точки зрения реализации. Обучение алгоритма $a_h(x; X^\ell)$ сводится к запоминанию выборки, подбору ядра K и ширины окна h.

Выбор ядра K мало влияет на точность аппроксимации, но определяющим образом влияет на степень гладкости функции $a_h(x)$.

Выбор ширины окна h решающим образом влияет на качество восстановления зависимости. При слишком узком окне ($h \to 0$) функция $a_h(x)$ стремится пройти через все точки выборки, реагируя на шум и претерпевая резкие скачки. При слишком широком окне функция чрезмерно сглаживается и в пределе $h \to \infty$ вырождается в константу.

4.3. Задачи кластеризации

4.3.1. Метод k-средних

Наиболее простой, но в то же время достаточно неточный метод кластеризации в классической реализации. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k. Действие алгоритма таково, что он стремится минимизировать среднеквадратичное отклонение на точках каждого кластера. Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Проблемы алгоритма k-средних:

- необходимо заранее знать количество кластеров;
- алгоритм очень чувствителен к выбору начальных центров кластеров. Классический вариант подразумевает случайный выбор кластеров.

Алгоритм:

сформировать начальное приближение центров всех кластеров $y \in Y$:
μ_y — наиболее удаленные друг от друга объекты выборки;

повторять

отнести каждый объект к ближайшему центру:

$$y_i = \arg\min_{y \in Y} \rho(x_i, \mu_y), \quad i = 1, \ldots, \ell;$$

вычислить новое положение центров:

$$\mu_{yj} = \frac{\sum_{i=1}^{\ell} [y_i = y] f_j(x_i)}{\sum_{i=1}^{\ell} [y_i = y]}, \quad y \in Y, j = 1, \ldots, n;$$

пока y_i не перестанут изменяться.

4.3.2. Иерархический метод

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху–вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева — дендрограммы.

К недостатку иерархических алгоритмов можно отнести систему полных разбиений, которая может являться излишней в контексте решаемой задачи.

Алгоритм восходящей иерархической кластеризации:

сначала все кластеры одноэлементные:

$$t = 1; \quad C_t = \big\{\{x_1\}, \ldots, \{x_\ell\}\big\}; \quad R(\{x_i\}, \{x_j\}) = \rho(x_i, x_j);$$

для всех $t = 2, \ldots, \ell$ (t — номер итерации):

найти в C_{t-1} два ближайших кластера:

$$(U, V) = \arg\min_{U \neq V} R(U, V); R_t = R(U, V);$$

слить их в один кластер:

$$W = U \cup V; C_t = C_{t-1} \cup \{W\} \backslash \{U, V\};$$

для всех $S \in C_t$

вычислить $R(W, S)$.

4.3.3. Выделение связных компонент

В алгоритме выделения связных компонент задается входной параметр R и в графе удаляются все ребра, для которых «расстояния» больше R. Соеди-

ненными остаются только наиболее близкие пары объектов. Смысл алгоритма заключается в том, чтобы подобрать такое значение R, лежащее в диапазон всех «расстояний», при котором граф «развалится» на несколько связных компонент. Полученные компоненты и есть кластеры.

Для подбора параметра R обычно строится гистограмма распределений попарных расстояний. В задачах с хорошо выраженной кластерной структурой данных на гистограмме будет два пика — один соответствует внутрикластерным расстояниям, второй — межкластерным расстояния. Параметр R подбирается из зоны минимума между этими пиками. При этом управлять количеством кластеров при помощи порога расстояния довольно затруднительно.

5. Направление развития

5.1. Глубинное обучение

Методы глубинного обучения являются попыткой реинкарнации нейронных сетей, с конца 80-ых гг. прошлого века переживающих кризис. Причинами кризиса традиционных нейронных сетей стали:

- критическая зависимость качества настройки весов сети от выбора начального приближения и, как следствие, проблемы с воспроизводимостью «успешных» результатов, публиковавшихся в научных журналах;
- большая подверженность переобучению вкупе со слабыми возможностями контроля обобщающей способности сети;
- большое количество локальных минимумов функционала качества, большинство из которых оказывались плохими.

С другой стороны, неоспоримой сильной стороной нейронных сетей явилось открытие метода обратного распространения ошибки, позволявшего отслеживать влияние внутренних слоев сети на качество прогноза скрытых переменных объектов обучающей выборки.

Во второй половине 00-ых гг. стало активно развиваться направление, получившее название глубинного обучения. В его основе лежат нейроные сети, претерпевшие значительные изменения:

- Глубиннное обучение строит не дискриминативные, а порождающие модели, в которых моделируется общее распределение $p(X;T;\vec{w})$, в отличие от дискриминативных моделей, позволяющее, например, генерировать новые объекты.

- В наиболее распространенной постановке все переменные объектов предполагаются бинарными. Это облегчает моделирование зависимостей между переменными объекта.

- Каждый слой сети сначала обучается независимо, проходя процедуру предобучения. Это позволяет «нащупать» хорошее начальное приближение для последующего запуска алгоритма обратного распространения ошибки. Каждый слой, в зависимости от выбранной модели, представляет собой ограниченную машину Больцмана или сверточную сеть.

- Для обучения используются сотни тысяч и миллионы объектов. Такие гигантские выборки позволяют настраивать сети с десятками тысяч параметров, без риска переобучения. Обученные таким образом сети, не просто позволяют моделировать сложные объекты, но и генерируют в процессе обучения информативные признаковые описания, которые могут быть использованы другими, более простыми алгоритмами машинного обучения в качестве наблюдаемых переменных объекта.

5.2. Непараметрические байесовские методы

Традиционно, методы непараметрической статистики определялись как раздел статистики, в которой число параметров, описывающих данные не фиксировано, а растет с ростом числа объектов. Для примера можно рассмотреть задачу определения числа кластеров в постоянно растущей выборке объектов. Данная задача тем более актуальна, что общепринятых методов определения, из скольки же кластеров состоит даже зафиксированная выборка, на сегодняшний день не существует. Чем больше объектов поступает в наше распоряжение, тем с большим разрешением мы можем находить в них структуру, выделяя кластеры схожих между собой объектов. В случае достаточно неоднородной выборки число кластеров должно постепенно увеличиваться по мере поступления новых объектов. В непараметрическом случае, нам необходимо задать распре-

деление над всевозможными разбиениями произвольного количества объектов. Такое распределение задается с помощью случайных процессов. В данном случае, это процесс Дирихле. С его помощью, удается не только рассчитать для любого разбиения произвольного числа объектов на кластеры его априорную вероятность, но и учесть характеристики объектов, чтобы перейти к апостериорному распределению на всевозможные разбиения. Как это часто бывает при применении байесовских методов, апостериорное распределение имеет острый пик, который соответствует устойчивому разбиению выборки объектов на некоторое число кластеров.

5.3. Обучение с подкреплением

Обучение с подкреплением предназначено для обучения агентов в условиях неопределенности, порождаемой как неполнотой информации об окружающей обстановке, так и возможными действиями других агентов. В зависимости от текущего состояния среды и действий агентов рассчитывается функция выгоды, которую получит агент в следующий момент времени.

Важным достоинством алгоритмов обучения с подкреплением является возможность обучения агента «с нуля» за счет балансируемого сочетания режимов «исследование–использование» и выучивания стратегий, позволяющих жертвовать малым сейчас ради получения большей выгоды в дальнейшем. Алгоритмы обучения с подкреплением нашли широкое применение не только в таких традиционных областях как робототехника, но и, например, на фондовых рынках.

5.4. Анализ больших объемов данных

Традиционные методы машинного обучения не всегда применимы для анализа выборок такого размера, поскольку в них зачастую неявно предполагается, что вся выборка помещается в памяти компьютера, или же они имеют недостаточно высокие показатели скорости роста вычислительной сложности в зависимости от размера выборки). Для преодоления этих ограничений часто используются приемы из следующих категорий:

- Распараллеливание. Независимые части алгоритма могут выполняться па-

раллельными обработчиками и в произвольном порядке. В некоторых случаях параллельной реализации классичесского алгоритма может быть достаточно для конкретной задачи.

- Аппроксимация. Известно, что многие сложные задачи могут быть решены приближенно с достаточно большой точностью, достаточной для данного эксперимента.

- Стохастичность. При наличии большого числа независимых объектов в выборке, многие необходимые статистики могут быть оценены по случайной подвыборке, при этом сохраняются теоретические гарантии оптимальности и сходимости алгоритма.

В последнее время стали также набирать популярность так называемые потоковые алгоритмы, способные обучаться инкрементально в режиме реального времени на постоянно поступающих данных без необходимости хранить их где-либо в памяти. Спрос на них возникает, как правило, в приложениях, где данные поступают в таких количествах и с такой скоростью, что нет никакой возможности сохранять их, по крайней мере, надолго. С такими задачами анализа данных сталкиваются, например, исследователи в ЦЕРНе, где данные генерируются со скоростью более 700 мегабайт в секунду.

6. Библиотеки

Microsoft Azure Machine Learning — инфраструктура облачных вычислений от компании Microsoft, представляющая собой обновленную версию MS Azure. В контексте машинного обучения инструмент представляет собой платформу, которая позволяет вне зависимости от качества данных строить решения прямо «на облаке» и с легкостью создавать на их основе BI-приложения.

RapidMiner — один из самых известных инструментов анализа данных. ПО разрабатывалось в целях машинного обучения, так что на сегодня многие пользователи хорошо знакомы с его преимуществами. Среди них — удобный графический интерфейс с функцией «перетаскивания» потоков данных.

Apache Mahout — проект фонда Apache и часть экосистемы Apache Hadoop, предназначенная для реализации распределенных или предусматривающих возможность масштабирования алгоритмов машинного обучения. ПО с открытым

кодом главным образом ориентировано на алгоритмы совместной фильтрации, кластеризации и классификации.

Caffe представляет собой библиотеку на языке C++, реализующую алгоритмы глубокого обучения, которая была разработана с упором на такие важные характеристики, как поддержание уровня чистоты данных, их читабельность и скорость обработки.

Открытый код, поддержка Python и интеграция с MATLAB, а также высокая скорость работы позволили Caffe найти себе широкое применение, в том числе, и в коммерческой среде.

OpenCV — библиотека с открытым кодом, реализованная на C/C++ и перенесенная на Python, Java и MATLAB. OpenCV предлагает широкий инструментарий для работы с визуализацией — в том числе, и в целях машинного обучения, причем библиотека работает с большинством операционных систем, включая мобильные iOS и Android.

Torch — научный вычислительный фреймвок, имеющий широкую поддержку алгоритмов машинного обучения, использующих для работы GPU. Он прост в использовании и довольно эффективен благодаря скриптовому языку LuaJIT и реализации на C/CUDA.

Weka — библиотека алгоритмов машинного обучения для решения задач интеллектуального анализа данных. Система позволяет непосредственно применять алгоритмы к выборкам данных, а также вызывать алгоритмы из программ на языке Java.

Theano — это библиотека на Python и оптимизирующий компилятор, которые позволяют определять, оптимизировать и вычислять математические выражения эффективно используя многомерные массивы. Библиотека тесно интегрирована с NumPy, поддерживает использование GPU, динамически генерирует код на C.

TensorFlow — система с открытым исходным кодом от Google. TensorFlow способна создавать и обучать нейронные сети. Среди преимуществ системы TensorFlow — простое масштабирование и возможность использовать ее в качестве инструмента анализа сложных данных. Система предоставляет унифицированный интерфейс для различных методов машинного обучения, позволяющий строить модели в виде графа и включающий в себя имплементацию методов оптимизации этих моделей.

Scikit-learn — отлично документированная библиотека на Python, в которой реализовано большое количество алгоритмов машинного обучения. Помимо этого библиотека содержит различные инструменты начальной и конечной обработки данных и визуализации.

7. Литература, курсы, конференции

1. machinelearning.ru — Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных.
2. habrahabr.ru/hub/machine_learning — Хаб «Машинное обучение».
3. datareview.info — информационно-образовательный портал, посвященный вопросам анализа и обработки данных.
4. yury.name/internet — Юрий Лифшиц – курс «Алгоритмы для Интернета».
5. intuit.ru/studies/courses/13844/1241/info — Курс «Машинное обучение» от Школы Анализа Данных (Яндекс).
6. uic.unn.ru/∼zny/ml — Н. Ю. Золотых – курс «Машинное обучение».
7. mmro.ru — конференции «Математические методы распознавания образов» и «Интеллектуализация обработки информации».
8. machinelearning.org/icml.html — International Conference on Machine Learning.
9. nips.cc — конференция Neural Information Processing Systems.
10. learningtheory.org — конференция Computational Learning Theory.

Список использованной литературы

[1] Воронцов, К. В. Математические методы обучения по прецедентам (теория обучения машин). Курс лекций [Электронный ресурс] — Режим доступа: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf

[2] MachineLearning.ru — Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных [Электронный ресурс] — Режим доступа: http://www.machinelearning.ru/

[3] Обзор алгоритмов кластеризации данных [Электронный ресурс] // Хабрахабр — Режим доступа: https://habrahabr.ru/post/101338/

[4] Введение в машинное обучение с помощью Python и Scikit-Learn [Электронный ресурс] // Хабрахабр — Режим доступа: https://habrahabr.ru/company/mlclass/blog/247751/

[5] Google открыла для всех библиотеку машинного обучения TensorFlow [Электронный ресурс] // Хабрахабр — Режим доступа: https://geektimes.ru/post/265516/

[6] Машинное обучение для самых маленьких [Электронный ресурс] // SavePearlHarbor – Ещё одна копия хабора — Режим доступа: http://savepearlharbor.com/?p=189178

[7] Кластеризация: алгоритмы k-means и c-means [Электронный ресурс] // Хабрахабр — Режим доступа: https://habrahabr.ru/post/67078/

[8] ТОП-5 инструментов для машинного обучения // DataReview.info — Режим доступа: http://datareview.info/article/top-5-instrumentov-dlya-mashinnogo-obucheniya/

[9] Золотых, Н. Ю. Машинное обучение и анализ данных [Электронный ресурс] — Режим доступа: http://www.uic.unn.ru/~zny/ml/Lectures/Old/ml_pres_2015a.pdf

[10] Воронцов, К. В. Методы кластеризации [Электронный ресурс] — Режим доступа:
http://www.machinelearning.ru/wiki/images/2/28/Voron-ML-Clustering-slides.pdf

[11] Ветров, Д. П. Машинное обучение – состояние и перспективы [Электронный ресурс] — Режим доступа:
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.402.8676

[12] Классификация [Электронный ресурс] // MachineLearning.ru — Режим доступа:
http://www.machinelearning.ru/wiki/index.php?title=Классификация