Министерство образования и науки Российской Федерации

Калужский филиал

федерального государственного бюджетного образовательного

учреждения высшего образования

«Московский государственный технический университет

имени Н.Э. Баумана

(национальный исследовательский университет)»

(КФ МГТУ им. Н.Э. Баумана)

И.И. Кручинин
(к.т.н. доцент)

Вводная лекция
по курсу «Введение в машинное обучение»

Калуга - 2018

Краткое содержание:

Введение:

Машинное обучение (machine learning) – это область научного знания, имеющая дело с алгоритмами, "способными обучаться". Необходимость использования методов машинного обучения объясняется тем, что для многих сложных – "интеллектуальных" – задач (например, распознавание рукописного текста, речи и т. п.) очень сложно (или даже невозможно) разработать "явный" алгоритм их решения, однако часто можно научить компьютер обучиться решению этих задач. Одним из первых, кто использовал термин "машинное обучение", был изобретатель первой самообучающейся компьютерной программы игры в шашки А. Л. Самуэль в 1959 г.. Под обучением он понимал процесс, в результате которого компьютер способен показать поведение, которое в нее не было заложено "явно".

В настоящее время машинное обучение имеет многочисленные сферы приложения, такие, как компьютерное зрение, распознавание речи, компьютерная лингвистика и обработка естественных языков, медицинская диагностика, биоинформатика, техническая диагностика, финансовые приложения, поиск и рубрикация текстов, интеллектуальные игры, экспертные системы.

Различают дедуктивное и индуктивное обучение. В задачах дедуктивного обучения имеются знания, каким-либо образом формализованные. Основная задача индуктивного обучения заключается в восстановлении некоторой зависимости по эмпирическим данным. Индуктивное обучение подразделяется на обучение с учителем, обучение без учителя, обучение с подкреплением.

Основные понятия и определения

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

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

Решающая функция a должна допускать эффективную компьютерную реализацию; по этой причине будем называть её алгоритмом.

1.1.1 Объекты и признаки

Признак (feature) f объекта x — это результат измерения некоторой характеристики объекта. Формально признаком называется отображение $f: X \to D_f$, где D_f — , множество допустимых значений признака. В частности, любой алгоритм $a: X \to Y$ также можно рассматривать как признак.

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 = R, то f — количественный признак.

Если все признаки имеют одинаковый тип, $D_{f_1} = \cdots = D_{f_n}$, то исходные данные называются однородными, в противном случае — разнородными.

Пусть имеется набор признаков $f_1, \ldots, f_n$. Вектор $(f_1(x), \ldots, f_n(x))$ называют признаковым описанием объекта $x \in X$.В дальнейшем мы не будем различать объекты из X и их признаковые описания, полагая $X = D_{f_1} \times \ldots \times D_{f_n}$. Совокупность признаковых описаний всех объектов выборки X^ℓ, записанную в виде таблицы размера $\ell \times n$, называют матрицей объектов-признаков:

$$F = \|f_j(x_i)\|_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix}.$$

Матрица объектов-признаков является стандартным и наиболее распространённым способом представления исходных данных в прикладных задачах.

Ответы и типы задач

В зависимости от природы множества допустимых ответов Y задачи обучения по прецедентам делятся на следующие типы.

Если $Y = \{1,\ldots,M\}$, то это задача классификации (classification) на M непересекающихся классов. В этом случае всё множество объектов X разбивается на классы $K_y = \{x \in X : y*(x) = y\}$, и алгоритм $a(x)$ должен давать ответ на вопрос «какому классу принадлежит x?». В некоторых приложениях классы называют образами и говорят о задаче распознавания образов (pattern recognition).

Если $Y = \{0,1\}^M$, то это задача классификации на M пересекающихся классов. В простейшем случае эта задача сводится к решению M независимых задач классификации с двумя непересекающимися классами.

Если $Y = \mathrm{R}$, то это задача восстановления регрессии (regression estimation).

Задачи прогнозирования (forecasting) являются частными случаями классификации или восстановления регрессии, когда $x \in X$ — описание прошлого поведения объекта x, $y \in Y$ — описание некоторых характеристик его будущего поведения.

1.1.3 Модель алгоритмов и метод обучения

Опр. 1.1. 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 или пространством поиска (search space).

Пример 1.1. В задачах с n числовыми признаками $f_j : X \to \mathrm{R}$, $j = 1,\ldots,n$ широко используются линейные модели с вектором параметров $\theta = (\theta_1,\ldots,\theta_n) \in \Theta = \mathrm{R}^n$:

$$g(x,\theta) = \sum_{j=1}^{n} \theta_j f_j(x)$$

n — для задач восстановления регрессии, $Y = \mathrm{R}$;

$$g(x,\theta) = \operatorname{sign} \sum_{j=1} \theta_j f_j(x)$$

— для задач классификации, $Y = \{-1,+1\}$.

Признаками могут быть не только исходные измерения, но и функции от них. В частности, многомерные линейные модели могут использоваться даже в задачах с единственным числовым признаком.

Пример 1.2. Один из классических подходов к аппроксимации функций одной переменной по заданным точкам $(x_i,y_i) \in \mathrm{R}^2$, $i = 1,\ldots,\ell$ заключается в построении полиномиальной модели. Если ввести n признаков $f_j(x) = x^{j-1}$, то функция $g(x,\theta)$ из примера 1.1 будет определять полином степени $n - 1$ над исходным признаком x.

Процесс подбора оптимального параметра модели θ по обучающей выборке X^ℓ называют настройкой (fitting) или обучением (training, learning)[1] алгоритма $a \in A$.

Опр. 1.2. Метод обучения (learning algorithm) — 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.

Итак, в задачах обучения по прецедентам чётко различаются два этапа.

[1] Согласно английской терминологии алгоритм является обучаемым, учеником (learning machine), а выборка данных — обучающей, учителем (training sample).

На этапе обучения метод μ по выборке X^ℓ строит алгоритм $a = \mu(X^\ell)$.

На этапе применения алгоритм a для новых объектов x выдаёт ответы $y = a(x)$.

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

1.1.4 Функционал качества

Опр. 1.3. Функция потерь (loss function) — это неотрицательная функция $L(a,x)$, характеризующая величину ошибки алгоритма a на объекте x. Если $L(a,x) = 0$, то ответ $a(x)$ называется корректным.

Опр. 1.4. Функционал качества алгоритма a на выборке X^ℓ:

$$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}(a, x_i) \quad . \qquad (1.2)$$

Функционал Q называют также функционалом средних потерь или эмпирическим риском [4], так как он вычисляется по эмпирическим данным $(x_i, y_i)_{i=1}^{\ell}$.

Функция потерь, принимающая только значения 0 и 1, называется бинарной. В этом случае $L(a,x) = 1$ означает, что алгоритм a допускает ошибку на объекте x, а функционал Q называется частотой ошибок алгоритма a на выборке X^ℓ.

Наиболее часто используются следующие функции потерь, при $Y \subseteq R$:

$L(a,x) = [a(x) =6 y*(x)]$ — индикатор ошибки, обычно применяется в задачах классификации[2];

$L(a,x) = |a(x) - y*(x)|$ — отклонение от правильного ответа; функционал Q называется средней ошибкой алгоритма a на выборке X^ℓ;

$L(a,x) = (a(x) - y*(x))^2$ — квадратичная функция потерь; функционал Q называется средней квадратичной ошибкой алгоритма a на выборке X^ℓ; обычно применяется в задачах регрессии.

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

$$\mu(X^\ell) = \operatorname{argmin}_{a \in A} Q(a, X^\ell). \qquad (1.3)$$

Пример 1.3. В задаче восстановления регрессии ($Y = R$) с n числовыми признаками $f_j : X \to R$, $j = 1,\ldots,n$, и квадратичной функцией потерь метод минимизации эмпирического риска есть ничто иное, как метод наименьших квадратов:

$$\mu(X^\ell) = \operatorname*{argmin}_{\theta} \sum_{i=1}^{\ell} \big(g(x_i, \theta) - y_i\big)^2 .$$

1.1.5 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

[2] Квадратные скобки переводят логическое значение в число по правилу [ложь] = 0, [истина] = 1.

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 распределёнными (independent identically distributed, i.i.d.).

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 $p(y|x) = \delta(y - y*(x))$, где $\delta(z)$ — дельта-функция.

Принцип максимума правдоподобия. При вероятностной постановке задачи вместо модели алгоритмов $g(x,\theta)$, аппроксимирующей неизвестную зависимость $y*(x)$, задаётся модель совместной плотности распределения объектов и ответов $\phi(x,y,\theta)$, 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: $p(X^\ell) = p\big((x_1,y_1),\ldots,(x_\ell,y_\ell)\big) = p(x_1,y_1)\cdots p(x_\ell,y_\ell)$. Подставляя вместо $p(x,y)$ модель плотности $\phi(x,y,\theta)$, получаем функцию правдоподобия (likelihood):

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

Чем выше значение правдоподобия, тем лучше выборка согласуется с моделью. Значит, нужно искать значение параметра θ, при котором значение $L(\theta,X^\ell)$ максимально. В математической статистике это называется принципом максимума правдоподобия. Его формальные обоснования можно найти, например, в [13].

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

Связь максимизации правдоподобия с минимизацией эмпирического риска. Вместо максимизации L удобнее минимизировать функционал $-\ln L$, поскольку он аддитивен (имеет вид суммы) по объектам выборки:

$$-\ln L(\theta, X^\ell) = -\sum_{i=1}^{\ell} \ln \varphi(x_i, y_i, \theta) \to \min_\theta. \qquad (1.4)$$

Этот функционал совпадает с функционалом эмпирического риска (1.2), если определить вероятностную функцию потерь $\mathrm{L}(a_\theta,x) = -\ell\ln\phi(x,y,\theta)$. Такое определение потери вполне естественно — чем хуже пара (x_i,y_i) согласуется с моделью ϕ, тем меньше значение плотности $\phi(x_i,y_i,\theta)$ и выше величина потери $\mathrm{L}(a_\theta,x)$.

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

Пример 1.4. Пусть задана модель $g(x,\theta)$. Примем дополнительное вероятностное предположение, что ошибки $\varepsilon(x,\theta) = g(x,\theta) - y*(x)$ имеют нормальное распределениес нулевым средним и дисперсией σ^2. Тогда модель

$\mathcal{N}(\varepsilon; 0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$

плотности имеет вид

$$\varphi(x, y, \theta) = p(x)\varphi(y \mid x, \theta) = p(x)\mathcal{N}\big(g(x, \theta) - y^*(x); 0, \sigma^2\big).$$

Отсюда следует, что вероятностная функция потерь совпадает с квадратичной с точностью до констант C_0 и C_1, не зависящих от параметра θ:

$$-\ln \varphi(x, y, \theta) = -\ln p(x)\mathcal{N}\big(g(x, \theta) - y^*(x); 0, \sigma^2\big) = C_0 + C_1\big(g(x, \theta) - y^*(x)\big)^2.$$

Таким образом, существуют два родственных подхода к формализации задачи обучения: первый основан на введении функции потерь, второй — на введении вероятностной модели порождения данных. Оба в итоге приводят к схожим (иногда даже в точности одинаковым) оптимизационным задачам. Обучение — это оптимизация.

1.1.6 Проблема переобучения и понятие обобщающей способности

Минимизацию эмпирического риска следует применять с известной долей осторожности. Если минимум функционала $Q(a,X^\ell)$ достигается на алгоритме a, то это ещё 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 (overtraining) или переподгонки (overfitting). 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 её и строит алгоритм, который сравнивает предъявляемый объект x с обучающими объектами x_i из X^ℓ. В случае совпадения $x = x_i$ алгоритм выдаёт правильный ответ y_i. Иначе выдаётся произвольный 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 (generalization ability) метода μ характеризуется величиной $Q(\mu(X^\ell),X^k)$ 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.

Опр. 1.5. Метод обучения μ называется состоятельным, если при заданных достаточно малых значениях ε и η справедливо неравенство

$$\mathrm{P}^{X^\ell, X^k}\big\{Q(\mu(X^\ell), X^k) > \varepsilon\big\} < \eta. \qquad (1.5)$$

Параметр ε называется точностью, параметр $(1 - \eta)$ — надёжностью.

Допустима также эквивалентная формулировка: для любых простых выборок X^ℓ и X^k оценка $Q(\mu(X^\ell),X^k)$ 6 ε справедлива с вероятностью не менее $1 - \eta$.

Получение оценок вида (1.5) является фундаментальной проблемой статистической теории обучения. Первые оценки были получены в конце 60-х годов В.Н.Вапником и А.Я.Червоненкисом [5, 6, 7]. В настоящее время статистическая теория развивается очень активно [34], однако для многих практически интересных случаев оценки обобщающей способности либо неизвестны, либо сильно завышены.

Эмпирические оценки обобщающей способности применяются в тех случаях, когда не удаётся воспользоваться теоретическими.

Пусть дана выборка $X^L = (x_i, y_i)_{i=1}^L$. Разобьём её N различными способами на две непересекающиеся подвыборки — обучающую X_n^ℓ длины ℓ и контрольную X_n^k длины $k = L-\ell$. Для

каждого разбиения n = 1,...,N построим алгоритм $a_n = \mu(X_n^\ell)$и вычислим значение$Q_n = Q(a_n, X_n^k)$. Среднее арифметическое значений Q_n по всем разбиениям называется оценкой скользящего контроля (cross-validation, CV):

$$\mathrm{CV}(\mu, X^L) = \frac{1}{N}\sum_{n=1}^{N} Q(\mu(X_n^\ell), X_n^k). \qquad (1.6)$$

Возможны различные варианты скользящего контроля, отличающиеся способами разбиения выборки X^L . В простейшем варианте разбиения генерируются случайным образом, число N берётся в диапазоне от 20 до 100. Стандартом «де факто» считается методика t×q-кратного скользящего контроля (t×q-fold cross-validation), когда выборка случайным образом разбивается на q блоков равной (или почти равной) длины, каждый блок по очереди становится контрольной выборкой, а объединение всех остальных блоков — обучающей. Выборка X^L по-разному t раз разбивается на q блоков. Итого получается $N = tq$ разбиений. Данная методика даёт более точные оценки за счёт того, что все объекты ровно по t раз встречаются в контроле.

Недостатками скользящего контроля являются: вычислительная неэффективность, высокая дисперсия, неполное использование имеющихся данных для обучения из-за сокращения длины обучающей выборки с L до ℓ.

Методология тестирования обучаемых алгоритмов

Пока ещё не создан универсальный метод обучения по прецедентам, способный решать любые практические задачи одинаково хорошо. Каждый метод имеет свои преимущества, недостатки и границы применимости. На практике приходится проводить численные эксперименты, чтобы понять, какой метод из имеющегося арсенала лучше подходит для конкретной задачи. Обычно для этого методы сравниваются по скользящему контролю.

Существует два типа экспериментальных исследований, отличающихся целями и методикой проведения.

Эксперименты на модельных данных. Их цель — выявление границ применимости метода обучения; построение примеров удачной и неудачной его работы; понимание, на что влияют параметры метода обучения. Модельные эксперименты часто используются на стадии отладки метода. Модельные выборки сначала генерируются в двумерном пространстве, чтобы работу метода можно было наглядно представить на плоских графиках. Затем исследуется работа метода на многомерных данных, при различном числе признаков. Генерация данных выполняется либо с помощью датчика случайных чисел по заданным вероятностным распределениям, либо детерминированным образом. Часто генерируется не одна модельная задача, а целая серия, параметризованная таким образом, чтобы среди задач оказались как заведомо «лёгкие», так и заведомо «трудные»; при такой организации эксперимента точнее выявляются границы применимости метода.

Эксперименты на реальных данных. Их цель — либо решение конкретной прикладной задачи, либо выявление «слабых мест» и границ применимости конкретного метода. В первом случае фиксируется задача, и к ней применяются многие методы, или, возможно, один и тот же метод при различных значениях параметров. Во втором случае фиксируется метод, и с его помощью решается большое число задач (обычно несколько десятков). Специально для проведения таких экспериментов создаются общедоступные репозитории реальных данных. Наиболее известный — репозиторий UCI (университета Ирвина, Калифорния), доступный по адресу http://archive.ics.uci.edu/ml. Он содержит около двух сотен задач, в основном классификации, из самых разных предметных областей.

Полигон алгоритмов классификации. В научных статьях по машинному обучению принято приводить результаты тестирования предложенного нового метода обучения в сравнении с другими методами на

представительном наборе задач. Сравнение должно производиться в равных условиях по одинаковой методике; если это скользящий контроль, то при одном и том же множестве разбиений. Несмотря на значительную стандартизацию таких экспериментов, результаты тестирования одних и тех же методов на одних и тех же задачах, полученные разными авторами, всё же могут существенно различаться. Проблема в том, что используются различные реализации методов обучения и методик тестирования, а проведённый кем-то ранее эксперимент практически невозможно воспроизвести во всех деталях. Для решения этой проблемы разработан Полигон алгоритмов классификации, доступный по адресу http://poligon.MachineLearning.ru. В этой системе реализована унифицированная расширенная методика тестирования и централизованное хранилище задач. Реализация алгоритмов классификации, наоборот, децентрализована. Любой пользователь Интернет может объявить свой компьютер вычислительным сервером Полигона, реализующим один или несколько методов классификации. Все результаты тестирования сохраняются как готовые отчёты в базе данных системы и могут быть в любой момент выданы по запросу без проведения трудоёмких вычислений заново.

Конкурсы по решению задач анализа данных. В последние годы компании, заинтересованные в решении прикладных задач анализа данных, всё чаще стали обращаться к такой форме привлечения научного сообщества, как открытые конкурсы с денежными премиями для победителя. В каждом таком конкурсе публикуется обучающая выборка с известными ответами, тестовая выборка, ответы на которой известны только организатору конкурса, и критерий, по которому алгоритмы претендентов сравниваются на данных тестовой выборки. Информацию о текущих конкурсах можно найти на сайтах http://www.kaggle.com, http://tunedit.org. Существуют также сайты, на которых можно тестировать различные алгоритмы на различных наборах данных: http://poligon.MachineLearning.ru, http://mlcomp.org.

1.2.7 Приёмы генерации модельных данных

Данный раздел носит справочный характер. В нём перечислены некоторые сведения, полезные при генерации модельных выборок данных.

Моделирование случайных данных. Следующие утверждения позволяют генерировать случайные выборки с заданными распределениями [10]. Будем предполагать, что имеется стандартный способ получать равномерно распределённые на отрезке [0,1] случайные величины.

Утв. 1. Если случайная величина r равномерно распределена на $[0,1]$, то случайная величина $\xi = [r < p]$ принимает значение 1 с вероятностью p и значение 0 с вероятностью $1 - p$.

Утв. 2. Если случайная величина r равномерно распределена на $[0,1]$, и задана возрастающая последовательность $F_0 = 0, F_1, \ldots, F_{k-1}, F_k = 1$, то дискретная случайная величина ξ, определяемая условием $F_{\xi-1} \leqslant r < F_\xi$, принимает значения $j = 1, \ldots, k$ с вероятностями $p_j = F_j - F_{j-1}$.

Утв. 3. Если случайная величина r равномерно распределена на $[0,1]$, и задана возрастающая на $\mathbb{R}$ функция $F(x)$, $0 \leqslant F(x) \leqslant 1$, то случайная величина $\xi = F^{-1}(r)$ имеет непрерывную функцию распределения $F(x)$.

Утв. 4. Если r_1, r_2 — две независимые случайные величины, равномерно распределённые на $[0,1]$, то *преобразование Бокса-Мюллера*

$$\xi_1 = \sqrt{-2\ln r_1}\sin 2\pi r_2;$$
$$\xi_2 = \sqrt{-2\ln r_1}\cos 2\pi r_2;$$

даёт две независимые нормальные случайные величины с нулевым матожиданием и единичной дисперсией: $\xi_1, \xi_2 \in \mathcal{N}(0,1)$.

Утв. 5. Если ξ — нормальная случайная величина из $\mathcal{N}(0,1)$, то случайная величина $\eta = \mu + \sigma\xi$ имеет нормальное распределение $\mathcal{N}(\mu, \sigma^2)$ с матожиданием μ и дисперсией σ^2.

Утв. 6. Пусть n-мерный вектор $x = (\xi_1, \ldots, \xi_n)$ составлен из независимых нормальных случайных величин $\xi_i \sim \mathcal{N}(0,1)$. Пусть V — невырожденная $n \times n$-матрица, $\mu \in \mathbb{R}^n$. Тогда вектор $x' = \mu + V^{\mathrm{T}}x$ имеет многомерное нормальное распределение $\mathcal{N}(\mu, \Sigma)$ с вектором матожидания μ и ковариационной матрицей $\Sigma = V^{\mathrm{T}}V$.

Утв. 7. Пусть на вероятностном пространстве X заданы k плотностей распределения $p_1(x),\ldots,p_k(x)$. Пусть дискретная случайная величина ξ принимает значения $1,\ldots,k$ с вероятностями $w_1,\ldots,w_k$. Тогда случайный элемент $x \in X$, полученный согласно распределению $p_\xi(x)$, подчиняется смеси распределений $p(x) = \sum_{j=1}^{k} w_j p_j(x)$.

На практике часто используют смеси многомерных нормальных распределений.

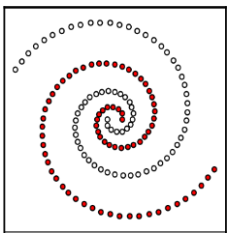

Рис. 1. Модельная выборка «спирали».

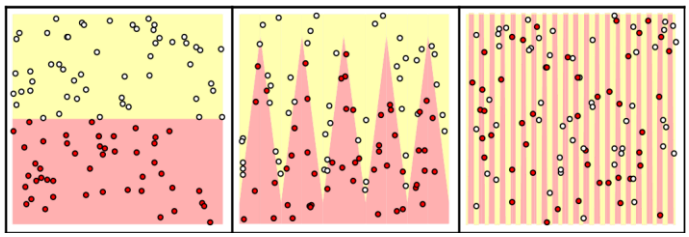

Рис. 2. Серия модельных выборок «пила».

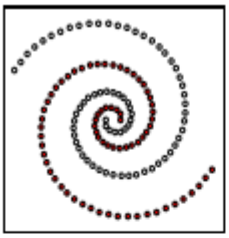

Рис. 1. Модельная выборка «спирали».

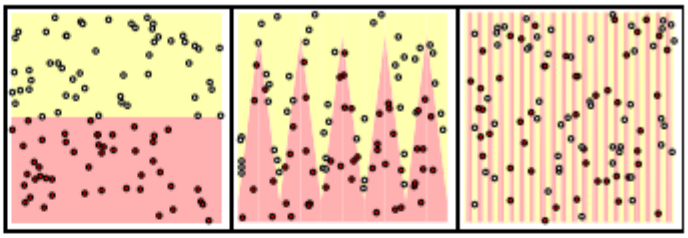

Рис. 2. Серия модельных выборок «пила».

Утв. 8. Предыдущий случай обобщается на континуальные смеси распределений. Пусть на вероятностном пространстве X задано параметрическое семейство плотностей распределения $p(x,t)$, где $t \in \mathbb{R}$ — параметр. Пусть значение $\tau \in \mathbb{R}$ взято из распределения с плотностью $w(t)$. Тогда случайный элемент $x \in X$, полученный согласно распределению $p(x,\tau)$, подчиняется распределению $p(x) = \int_{-\infty}^{+\infty} w(t)p(x,t)\,dt$. Этот метод, называемый *методом суперпозиций*, позволяет моделировать широкий класс вероятностных распределений, представимых интегралом указанного вида.

Утв. 9. Пусть в $\mathbb{R}^n$ задана прямоугольная область $\Pi = [a_1,b_1] \times \ldots \times [a_n,b_n]$ и произвольное подмножество $G \subset \Pi$. Пусть $r = (r_1,\ldots,r_n)$ — вектор из n независимых случайных величин r_i, равномерно распределённых на $[a_i,b_i]$. *Метод исключения* состоит в том, чтобы генерировать случайный вектор r до тех пор, пока не выполнится условие $r \in G$. Тогда результирующий вектор r равномерно распределён на G. Этот метод вычислительно неэффективен, если объём G много меньше объёма Π.

Неслучайные модельные данные позволяют наглядно продемонстрировать, в каких случаях одни методы работают лучше других.

Один из классических примеров — две спирали на Рис. 1. Эта выборка хорошо классифицируется методом ближайших соседей, но непреодолимо трудна для линейных разделяющих правил. Если витки спиралей расположить ближе друг к другу, задача станет трудна и для метода ближайших соседей. Некоторые кусочно-линейные разделители справляются с задачей и в этом случае.

Обычно при создании модельных данных, как случайных, так и неслучайных, вводится параметр, плавно изменяющий задачу от предельно простой до предельно трудной. Это позволяет исследовать границы применимости метода. На Рис. 2 показана серия модельных задач классификации с двумя классами, обладающая таким свойством относительно метода ближайших соседей и некоторых других алгоритмов.

Применение методов машинного обучения в различных сферах практической повседневной деятельности.

Задача классификации.

Классификатором или решающим правилом называется правило отнесения образа к одному из классов на основании его вектора признаков.

Пример 1. Иллюстрация понятий признаков и классификатора и идеи распознавания (классификации). Рассмотрим задачу диагностики печени по результатам инструментального исследования (рис.1.1).

Доброкачественные (левый рисунок – *класс* A) и злокачественные (правый рисунок – *класс* B) изменения дают разную картину. Предположим, что имеется несколько препаратов в базе данных, про которые известна их принадлежность к классам A и B (правильная классификация). Очевидно, что образцы отличаются интенсивностью точек изображения. В качестве вектора признаков выберем пару: среднее *значение* (μ) и среднеквадратичное отклонение (σ) интенсивности в изображении.

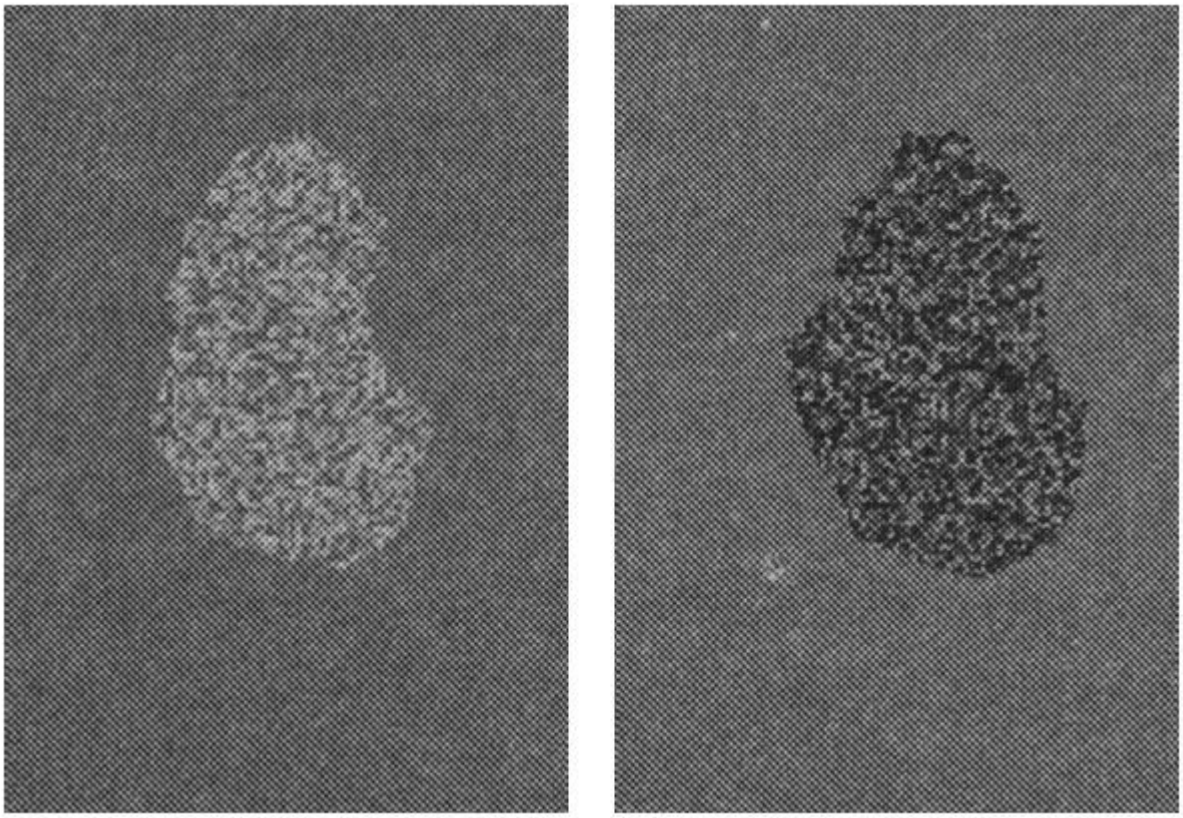

Рис. 1.1. Образы-прецеденты, соответствующие классу A (слева) и B (справа)

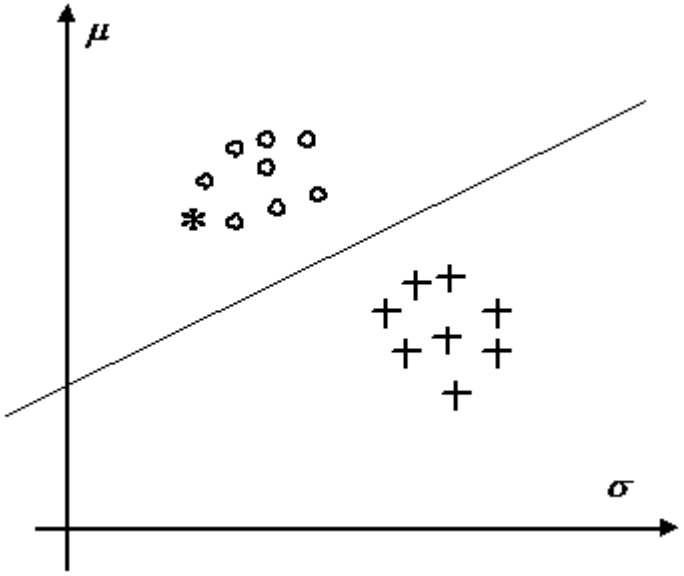

Рис. 1.2. Распределение векторов признаков прецедентах класса A (кружки) и класса B (крестики). Признаки - средние значения и средние отклонения яркости в образах. Прямая линия разделяет вектора из разных классов

На рис.1.2 представлены изображения этих образов в пространстве признаков. Точки, соответствующие прецедентам разных классов, разделяются *прямой* линией. Классификация неизвестного образа (соответствующая точка изображена звездочкой) состоит в проверке положения точки относительно этой разделяющей *прямой*.

Практическая разработка системы классификации осуществляется по следующей схеме (рис 1.3). В процессе разработки необходимо решить следующие вопросы.

1. **Как выбрать вектора признаков**? *Задача генерации признаков* – это выбор тех признаков, которые с достаточной полнотой (в разумных пределах) описывают образ.
2. **Какие признаки наиболее существенны для разделения объектов разных классов**? *Задача селекции признаков* – отбор наиболее информативных признаков для классификации.
3. **Как построить классификатор**? *Задача построения классификатора* – выбор решающего правила, по которому на основании вектора признаков осуществляется отнесение объекта к тому или иному классу.
4. **Как оценить качество построенной системы классификации**? *Задача количественной оценки системы* (выбранные признаки + классификатор) с точки зрения правильности или ошибочности классификации.

Рис. 1.3. Основные элементы построения системы распознавания образов (классификации)

Задача построения классификатора

Пусть

- Ω – пространство образов,
- X – признаковое пространство,
- $g(\omega),\ \omega \in \Omega$ – индикаторная функция,
- M – множество признаков.

Тогда $g : \Omega \to M$.

Пусть также

- $X = \langle x(\omega_i), g(\omega_i) \rangle,\ i = 1, \ldots, N$ – множество прецедентов,
- $\widehat{g}(x)$ – решающее правило.

Тогда $\widehat{g} : X \to M$

Выбор решающего правила исходит из минимизации $d(g, \widehat{g}) \to \min$, где d – *метрика, мера* близости функций $g(\omega)$ и $\widehat{g}(x(\omega))$. Построение $\widehat{g}$ называют задачей обучения. $\widehat{g}$ – это ученик, процедура формирования – это учитель, прецеденты – это обучающая последовательность.

2. Качество обучения классификатора

Относительная доля несовпадений классификации с учителем для решающего правила есть: $K = \frac{m}{N}$, где $m = |\{\omega_i : g(\omega_i) \neq \widehat{g}(x(\omega_i)),\ i = 1, 2, \ldots, N\}|$. Надежность обучения классификатора – это *вероятность* получения решающего правила с заданным качеством.

Пусть $f(x, \alpha)$ – *класс* дискриминантных функций, где $\alpha \in A$ – *параметр*. Число степеней свободы при выборе конкретной функции в классе определяется количеством параметров в векторе α, т.е. размерностью A.

Например, для классов линейных и квадратичных функций имеем:

Линейная дискриминантная *функция*: $f(x, \alpha) = \sum_{i=1}^{n} \alpha_i x_i + \alpha_0$. В таком случае имеем $n + 1$ степень свободы.

Квадратичная дискриминантная *функция*: $f(x, \alpha) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij} x_i x_j + \sum_{i=1}^{n} \beta_i x_i + \beta_0$. В таком случае имеем $n^2 + n + 1$ степеней свободы.

С увеличением степеней свободы увеличивается способность классификатора по разделению.

2.3. Вероятностная модель

Пусть прецеденты – это результат реализации случайных величин. Рассмотрим *величину риска* (т.е. ошибки) связанной с классификацией. Определим понятия риск среднего и риска эмпирического.

Пусть на Ω заданы σ -алгебра и *мера* P. Пусть также

- x – вектор признаков,
- $\widetilde{f}$ – класс функций, из которых выбирается решающее правило,
- $f(x, \alpha)$ – решающее правило (результат классификации), которое принимает значение 0 или 1 при фиксированном векторе параметра,
- χ – характеристическая функция множества,
- A – множество параметров, описывающие различные функции в $\widetilde{f}$.

Тогда $\widehat{g} = f(x, \alpha)$, где $f \in \widetilde{f}$ и $f : X \times A \to M$, $y = g(\omega)$.

В данных обозначениях средний риск выглядит следующим образом:

$$K(\alpha) = \int_X \chi\{y \neq f(x, \alpha)\} dP.$$

Для случая двух классов, при $M = \{0, 1\}$, имеем:

$$K(\alpha) = \int_{\Omega} (y - f(x, \alpha))^2 dP$$

или

$$K(\alpha) = \int_{(X,M)} (y - f(x, \alpha))^2 dP(x, y),$$

где dP – это вероятностная *мера* на пространстве X.

2.4. Задача поиска наилучшего классификатора

Рассмотрим минимизацию функционала:

$$K(\alpha) \to \min_{\alpha \in A}$$

Задача же поиска наилучшего классификатора состоит в нахождении α^* такого, что

$$K(\alpha^*) = \min_{\alpha \in A} K(\alpha)$$

Если же минимума не существует, то надо найти α^* такое, что

$$\left| K(\alpha^*) - \inf_{\alpha \in A} K(\alpha) \right| < \delta.$$

Другими словами, необходимо решить задачу минимизации среднего риска.

Поскольку dP неизвестно, будем решать задачу минимизации *эмпирического риска*. Пусть l – число прецедентов. Тогда эмпирический риск задается выражением:

$$K_{эмп}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} |y - f(x, \alpha)| .$$

Таким образом, задача минимизации *эмпирического риска* выглядит так:

$$K_{эмп}(\alpha) \to \min_{\alpha \in A},$$

где случайные величины мы минимизируем по параметру α – любой возможный *параметр*.

В идеале надо получить взаимосвязанные оценки эмпирического и среднего риска.

Отметим, что чем меньше l, тем легче построить $f(x, \alpha)$ такую, что $K_{эмп}(\alpha)$ обращается в ноль, либо очень мало. Но при этом истинное *значение* $K(\alpha)$ может сильно отличаться от $K_{эмп}(\alpha)$. Необходимо выбрать $f(x, \alpha)$ такую, чтобы имела *место* равномерная *сходимость* по α выражения:

$$P\left\{\sup_{\alpha}|K_{эмп}(\alpha)-K(\alpha)|>\varepsilon\right\}\xrightarrow[l\to\infty]{}0.$$

Фактически это есть *сходимость* частот к математическому ожиданию.

В дальнейшем будем считать, что в зависимости от конкретного набора прецедентов можем получить любые α. Но необходимо, чтобы полученные эмпирическое решающее хорошо работало (отражало общие свойства) для всех образов. Поэтому в формуле присутствует равномерная *сходимость*.

2.5. Сходимость эмпирического риска к среднему. Случай конечного числа решающих правил.

Пусть

- $K(\alpha)$ – математическое ожидание ошибки классификатора $f(x,\alpha)$,
- A – событие – ошибка классификатора при решающем правиле $f(x,\alpha)$,
- $P(A)$ – вероятность,
- $v(A)$ – частота в l испытаниях.

Воспользуемся неравенством Бернштейна, тогда

$$P\{|v(A)-P(A)|>\varepsilon\}\le e^{-2e^2l}$$

есть оценка – соотношение между частотой и вероятностью при заданном количестве испытаний.

Пусть ξ_j – *случайная величина*. Тогда $E(\xi_j)=0$ – математическое ожидание ξ_j, $E\xi_j^2=\delta^2$ – *дисперсия*, причем $|\xi_j|\le L$. Обозначим $S_0=\xi_1+\xi_2+\ldots+\xi_n$. Тогда соответствующая оценка имеет вид:

$$P\left\{|S_n|>t\delta\sqrt{n}\right\}\le 2\cdot\exp\left\{-\frac{t^2}{2\cdot\left(1+\frac{a}{3}\right)}\right\},\ \text{где } a=\frac{L\cdot t}{\sqrt{n\delta}}.$$

$$l=\frac{\ln N-\ln\eta}{2e^2}\ \text{и}\ e=\sqrt{\frac{\ln N-\ln\eta}{2l}},$$

где l – необходимое количество прецедентов для обеспечения близости.

Теорема. *Пусть из множества, состоящего из N решающих правил, выбирается правило, частота ошибок которого на прецедентах составляет v. Тогда с вероятностью $1-\eta$ можно утверждать, что вероятность ошибочной классификации с помощью данного правила $f(x\alpha)$ составит величину, меньшую $v+e$, если длина обучающей последовательности не меньше $l=\frac{\ln N-\ln\eta}{2e^2}$, где $e=\sqrt{\frac{\ln N-\ln\eta}{2l}}$, η и e заданы и последовательность независима.*

Данная теорема справедлива для случая конечного числа решающих правил. Вапник и Червоненкис смогли обобщить эти оценки на случай бесконечного числа решающих правил.

1.1. Задача обучения с учителем

1.1.1. Постановка задачи обучения с учителем

Рассмотрим постановку задачи обучения с учителем (supervised learning). Пусть $\mathbb{X}$ - некоторое множество, элементы которого называются объектами или примерами, ситуациями, входами (samples); а $\mathbb{Y}$ - множество, элементы которого называются ответами или откликами, метками, выходами (responses). Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in \mathbb{X}$ предсказать $y \in \mathbb{Y}$. В частности, если зависимость детерминированная, то существует функция $f^* : \mathbb{X} \to \mathbb{Y}$. Зависимость известна только на объектах обучающей выборки

$$\{(x^{(i)}, y^{(i)}) : x^{(i)} \in \mathbb{X}, y^{(i)} \in \mathbb{Y}(i = 1, ..., N)\}.$$

Упорядоченная пара "объект-ответ" $(\mathrm{x}^{(i)}, y^{(i)}) \in \mathbb{X} \times \mathbb{Y}$ называется прецедентом.

Задача обучения с учителем заключается в восстановлении зависимости между входом и выходом по имеющейся обучающей выборке, т. е. необходимо построить функцию (решающее правило) $f : \mathbb{X} \to \mathbb{Y}$, по новым объектам $x \in \mathbb{X}$ предсказывающую ответ $f(x) \in \mathbb{Y}$:

$$y = f(x) \approx f^*(x).$$

Функция f при этом выбирается из некоторого множества возможных моделей $\mathbb{F}$. Процесс нахождения f называется обучением (learning), а также настройкой или подгонкой (fitting) модели.

Алгоритм построения функции $f \in \mathbb{F}$ по заданной обучающей выборке называется алгоритмом обучения. Некоторый класс алгоритмов называется методом обучения. Иногда термины "алгоритм" и "метод" используются как синонимы.

Алгоритмы обучения, конечно же, оперируют не с самими объектами, а их описаниями. Наиболее распространенным является признаковое описание. При таком подходе объект x представляется как вектор $x = (x_1, x_2, ..., x_d)$, где $x_j \in Q_j(j = 1, 2, ..., d)$. Таким образом,

$$\mathbb{X} = Q_1 \times Q_2 \times ... \times Q_d$$

Компонента x_j называется j-м признаком, или свойством (feature), или атрибутом объекта x. Если $Q_j = \mathbb{R}$, то j-й признак называется количественным или вещественным. Если Q_j конечно, то j-й признак называется номинальным, или категориальным, или фактором. Если при этом $\mid Q_j \mid = 2$, то признак называется бинарным. Если Q_j конечно и упорядочено, то признак называется порядковым. Множество $\mathbb{X}$ называется пространством признаков.

В зависимости от того, какие значения может принимать ответ y, различают разные классы задач обучения с учителем. Если $\mathbb{Y} = \mathbb{R}$, то говорят о задаче восстановления регрессии. Решающее правило f при этом называют регрессией. Если $\mathbb{Y}$ конечно, например, $\mathbb{Y} = \{1, 2, ..., K\}$, то говорят о задаче классификации. Решающее правило f при этом называют классификатором. В

последнем случае y можно интерпретировать как номер класса, к которому принадлежит объект x. К задачам обучения относят также задачи ранжирования, прогнозирования и др.

Сделаем два замечания, касающиеся качества решения задачи.

Во-первых, найденное решающее правило должно обладать обобщающей способностью. Иными словами, построенный классификатор или функция регрессии должны отражать общую зависимость выхода от входа, основываясь лишь на известных данных о прецедентах обучающей выборки. При решении прикладных задач, как правило, говорить о восстановлении истинной взаимосвязи не приходится, и, следовательно, речь может идти лишь об отыскании некоторой аппроксимации. Таким образом, обученная модель должна выдавать в среднем достаточно точные предсказания на новых (не входящих в обучающую выборку) прецедентах.

Во-вторых, следует уделить внимание проблеме эффективной вычислимости функции f. Аналогичное требование предъявляется и к алгоритму обучения: настройка модели должна происходить за приемлемое время.

Пусть для задачи обучения с учителем определена функция потерь, или функцию штрафа, $L(y, y') = L(y, f(x))$, представляющая собой неотрицательную функцию от истинного значения выхода y и предсказанного с помощью модели значения $y' = f(x)$. Например, для задачи восстановления регрессии часто используют квадратичный штраф

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2$$

или абсолютный штраф:

$$L(y, f(x)) = | \, y - f(x) \, | \, .$$

Для задачи классификации можно взять ошибку предсказания

$$L(y, f(x)) = I(y \neq f(x)),$$

где $I(\cdot)$ - индикаторная функция:

$$I(\text{условие}) = \begin{cases} 1, & \text{условие выполнено,} \\ 0, & \text{условие не выполнено.} \end{cases}$$

Математическое ожидание функции потерь

$$R(f) = EL(Y, f(X))$$

называется средней ошибкой, или средним риском. В качестве решающего правила разумно взять функцию f^*, минимизирующую эту ошибку:

$$f^* = argmin_{f \in \mathbb{F}} R(f) = argmin_{f \in \mathbb{F}} EL(Y, f(X)), \qquad (1)$$

Часто закон распределения совместной случайной величины (X, Y) не известен, поэтому данный критерий не применим. Вместо среднего риска $R(f)$ рассмотрим эмпирический риск, или эмпирическую ошибку,

$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)})).$$

Критерий (1) заменим следующим:

$$f = agrmin_{f \in \mathbb{F}} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)})) \qquad (2)$$

где прецеденты $(x^{(i)}, y^{(i)}), (i = 1, 2, ..., N)$ составляют обучающую выборку.

В итоге задача свелась к отысканию функции f из допустимого множества $\mathbb{F}$, удовлетворяющей условию (2), при условии, что $\mathbb{F}$ и L фиксированы и известны. Это так называемый принцип минимизации эмпирического риска. Как правило, класс $\mathbb{F}$ параметризован, т. е. есть имеется его описание в вида $\mathbb{F} = \{f(x) = f(x, \theta) : \theta \in \Theta$, где Θ - некоторое известное множество. В процессе настройки модели алгоритмом обучения выбираются значения набора параметров Θ, обеспечивающих точное или приближенное выполнение условия (2), т. е. минимизации ошибки на прецедентах обучающей выборки. Однако данное условие не подходит для оценки обобщающей способности алгоритма. Более того, значения $\hat{R}(f)$ и $R(f)$ могут различаться значительно. Ситуация, когда $\hat{R}(f)$ мало, а $R(f)$ чересчур велико, называется переобучением.

На практике все имеющиеся данные разбивают на обучающую и тестовую выборки. Обучение производится с использованием обучающей выборки, а оценка качества предсказания на основе данных тестовой выборки.

Другим практическим методом оценки обобщающей способности решающего правила является метод q -кратного перекрестного (скользящего) контроля (CV - cross validation). Все имеющиеся данные разбиваются на q примерно равных (по числу прецедентов) частей. Далее, отделяя из выборки одну за другой каждую из этих частей, используют оставшиеся данные (составленные из $q-1$ частей) как обучающую выборку, а отделенную часть - как тестовую. Итоговая оценка ошибки определяется как средняя по всем разбиениям. Заметим, что само итоговое решающее правило строится по всей имеющейся обучающей выборке.

Часто используют значения $q = 5$ или 10 . Если $q = N-1$, то говорят о методе скльзящего контроля с одним отделяемым объектом (LOO - leave-one-out estimate).

1.1.2. Метод k ближайших соседей

Метод k ближайших соседей (k nearest-neighbor, k-NN) относится к наиболее простым и в то же время универсальным методам, используемым как для решения задач классификации, так и восстановления регрессии. В случае классификации новый объект классифицируется путем отнесения его к классу, являющемуся преобладающим среди k ближайших (в пространстве признаков) объектов из обучающей

выборки. Если $k = 1$, тоновый объект относится к тому же классу, что и ближайший объект из обучающей выборки.

Аналогичный способ используется и в задаче восстановления регрессии, с той лишь разницей, что в качестве ответа для объекта выступает среднее ответов k ближайших к нему объектов из обучающей выборки.

Опишем метод более формально. Пусть $N_k(x)$ - множество k ближайших к x объектов из обучающей выборки. Тогда для задачи классификации положим

$$f(x) = arg\min_y \mid \{i : y^{(i)} = y, x^{(i)} \in N_k(x)\} \mid,$$

а для задачи восстановления регрессии -

$$f(x) = \frac{1}{k} \sum_{x^{(i)} \in N_k(x)} y^{(i)}.$$

В некоторых случаях данные ответы учитываются с весами, обратно пропорциональными расстоянию до объекта. Это особенно полезно для решения задачи классификации с несбалансированными данными, т. е. когда число объектов, относящихся к разным классам, сильно различно.

Для определения ближайших соседей обычно используется евклидово расстояние

$$\rho(x, x') = \sqrt{\sum_{j=1}^{d} \mid x_j - x'_j \mid^2},$$

однако оно применимо только для признаков, описываемых количественными переменными. Если все переменные качественные, то можно использовать расстояние Хэмминга

$$\rho(x, x') = \sum_{j=1}^{d} I(x_j \neq x'_j).$$

В общем случае используют функцию

$$\rho(x, x') = \sum_{j=1}^{d} \alpha_j \rho_j(x_j, x'_j),$$

где α_j - неотрицательные параметры, $\rho_j(x, x') = (x_j - x'_j)^2$ для качественных переменных, $\rho_j(x, x') = I(x_j \neq x'_j)$ для качественных переменных. Заметим, что функция расстояния не обязательно должна быть метрикой и неравенство треугольника может быть не выполнено.

Для повышения точности модели также могут использоваться специальные алгоритмы обучения метрики расстояния (например, Large Margin Nearest Neighbour [8]).

Одним из основных параметров, влияющих на обобщающую способность алгоритма, является число "ближайших соседей" k. В целом, выбор определенного значения обусловлен характером данных

задачи. Большие значения k могут привести как к более точному описанию границы, разделяющей классы, так и переобучению. Обычно для выбора k применяют различные эвристики, в частности, метод перекрестного контроля.

1.1.3. Машина опорных векторов

Один из самых популярных методов машинного обучения – машина опорных векторов (SVM – Support Vector Machine) – является развитием идей, предложенных в 1960–1970 гг. В. Н. Вапником и А. Я. Червоненкисом. Окончательное очертание метод принял в 1995 г., когда было показано, как в этом методе можно эффективно использовать ядра [4].

Рассмотрим вначале случай двух линейно разделимых классов. Для удобства будем их кодировать числами 1, -1, т. е. $\mathbb{Y} = \{1, -1\}$.

Оптимальной разделяющей гиперплоскостью называется гиперплоскость, такая, что расстояние от нее до ближайшей точки из обучающей выборки (не важно из какого класса) максимально. Таким образом, оптимальная разделяющая гиперплоскость максимизирует зазор (отступ) - расстояние от нее до точек из обучающей выборки. Часто это приводит к хорошим результатам и на объектах тестовой выборки.

Математическая постановка задачи выглядит следующим образом:

Требуется найти

$$\max_{\beta,\beta_0,w} w,$$

при ограничениях

$$y^{(i)}(\beta x^{(i)} + \beta_0) \geq w \quad (i = 1, 2, ..., N),$$

$$\beta = 1,$$

где $\beta x^{(i)}$ означает скалярное произведение

$$\beta x^{(i)} = \sum_{j=1)^d \beta_j x_j^{(i)}},$$

а β – евклидову норму

$$\beta = \sqrt{\sum_{j=1)^d \beta_j{}^2}}.$$

Эквивалентная формулировка:

$$\max_{\beta,\beta_0} \beta^2,$$

при ограничениях

$$y^{(i)}(\beta x^{(i)} + \beta_0) \geq 1 \quad (i = 1, 2, ..., N).$$

Можно показать, что решение этой задачи имеет вид

$$\beta = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)}, \quad \text{где} \alpha_i \geq 0,$$

уравнение разделяющей гиперплоскости есть $\beta x + \beta_0 = 0$ и классификатор определяется правилом

$$f(x) = sign(\beta x + \beta_0).$$

Если $\alpha_i < 0$, то $x^{(i)}$ называется опорной точкой или опорным вектором. Легко видеть, что опорные точки лежат на границе разделяющей полосы

$$-1 \leq \beta x + \beta_0 \leq 1,$$

следовательно, $\beta_0 = y^{(i)} - \beta x^{(i)}$, где $x^{(i)}$ - произвольный опорный вектор.

В случае линейно не разделимых классов разрешим некоторым объектам из обучающей выборки "слегка" заходить за разделяющую гиперплоскость:

$$\max_{\beta, \beta_0, \zeta, w} w,$$

при ограничениях

$$y^{(i)}(x^{(i)}\beta + \beta_[0) \geq w(1 - \zeta_i), \quad \zeta_i \geq 0, \quad (i = 1, 2, ..., N),$$

$$\beta = 1,$$

$$\sum_{i=1}^{N} \zeta_i \leq \Xi,$$

где Ξ - некоторая константа (параметр метода). При $\Xi = 0$ получаем предыдущий случай (линейно разделимых классов). Значение ζ_i пропорционально величине, на которую $x^{(i)}$ заходит за границу разделяющей полосы. В частности, -й объект будет классифицирован неправильно тогда и только тогда, когда $\zeta_i > 1$. Чем меньше Ξ, тем меньше объектов классифицируется неправильно. С другой стороны, Ξ должно быть достаточно велико, чтобы задача была совместной.

Эквивалентная формулировка:

$$\max_{\beta, \beta_0, \zeta} \frac{1}{2}\beta + C \sum_{i=1}^{N} \zeta_i,$$

при ограничениях

$$y^{(i)}(\beta x^{(i)} + \beta_0) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, \quad (i = 1, 2, ..., N).$$

Здесь параметр $C \geq 0$ регулирует величину штрафа за то, что некоторые точки выходят за границу разделяющей полосы. Построенная задача является задачей квадратического программирования. Для ее решения можно использовать общие методы для решения таких задач, однако существуют весьма эффективные специальные методы.

Можно показать, что решение задачи единственно, если в обучающей выборке есть по крайней мере один представитель каждого из классов.

Дальнейшее усовершенствование метода - использование спрямляющих пространств. Пусть удается перейти от исходного пространства признаков $\mathbb{X}$ к новому пространству $\mathbb{H}$ (которое называется спрямляющим) с помощью некоторого отображения $h : \mathbb{X} \to \mathbb{H}$:

$$h(x) = (h_1(x), h_2(x), ..., h_M(x)),$$

где $h_m(x)$ - базисные функции ($(m = 1, 2, ..., M)$. Новый классификатор определяется теперь функцией

$$f(x) = sign(h(x)\beta + \beta_0).$$

Здесь $h(x)\beta + \beta_0 = 0$ - разделяющая поверхность. Если отображение подобрано удачно, в новом пространстве классы могут быть линейно разделимы или близки к таковым.

Оказывается, все вычисления при обучении и при расчете функции $f(x)$ можно организовать так, что $h(x)$ встречается только в выражениях вида

$$K(x, x') = h(x)h(x').$$

В частности уравнение разделяющей поверхности имеет вид

$$\left(\sum_{i=1}^{N} \alpha_i y^{(i)} h(x^{(i)}) \right) h(x) + \beta_0 = 0$$

что эквивалентно

$$\sum_{i=1}^{N} \alpha_i y^{(i)} h(x^{(i)}) h(x) + \beta_0 = 0 \quad \text{или} \quad \sum_{i=1}^{N} \alpha_i y^{(i)} K(x^{(i)}, x) + \beta_0 = 0.$$

Функция $K(x, x')$ называется ядром. На практике, таким образом, вместо того, чтобы испытывать различные функции h можно (так и поступают) подбирать наиболее подходящее ядро $K(x, x')$. Заметим, что не любая функция $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ представима в виде $K(x, x') = h(x)h(x')$, т.е. может быть использована в качестве ядра. Приведем перечень некоторых популярных функций-ядер:

линейное ядро: $K(x, x') = xx'$;

многочлен степени d : $K(x, x') = (\gamma_0 + \gamma xx')^d$;

радиальная функция: $K(x, x') = e^{-\gamma x - x'^2}$;

сигмоидальная ("нейронная") функция: $K(x, x') = tanh(\gamma_0 + \gamma xx')$.

Для того чтобы использовать метод опорных векторов для задачи классификации с числом классом $K > 2$, возможно использовать две стратегии:

"Каждый против каждого": построить $K(K-1)/2$ классификаторов на всех возможных подзадачах бинарной классификации. Новый объект классифицируется всеми построенными решающими правилами, затем выбирается преобладающий класс.

"Один против всех": обучить K моделей на задачах бинарной классификации вида "один класс против всех остальных". Класс нового объекта выбирается по максимальному значению отступа.

1.1.4. Деревья решений

Деревья решений [3] являются одним из наиболее наглядных и универсальных алгоритмов обучения. К достоинствам деревьев решений следует отнести:

Возможность производить обучение на исходных данных без их дополнительной предобработки (нормализация и т. п.);

Нечувствительность к монотонным преобразованиям данных;

Устойчивость к выбросам;

Возможность обрабатывать данные с пропущенными значения;

Поддержка работы с входными переменными разных (смешанных) типов;

Возможность интерпретации построенного дерева решений.

Кроме того, существуют эффективные алгоритмы их настройки (обучения), например, CART – Classification and Regression Trees [3] или See5/C5.0.

Основная идея деревьев решений состоит в рекурсивном разбиении пространства признаков с помощью разбиений (splits): гиперплоскостей, параллельных координатным гиперплоскостям (если признак – количественный), либо по категориям (если признак номинальный). В каждом из полученных в конце процедуры "ящиков" $R_1, R_2, ..., R_M$ функция аппроксимируется константой:

Для задачи классификации:

$$f(x) = arg\max_k \mid \{i : x^{(i)} \in R_m, y^{(i)} = k\} \mid$$

Для задачи восстановления регрессии:

$$f(x) = \frac{1}{N_m} \sum_{x^{(i)} \in R_m} y^{(i)}$$

В алгоритме CART разбиения имеют вид:

$x_j \leq c$, если j-й признак количественный;

$x_j \in L$, если j -й признак качественный, $L \subset Q_j$, где Q_j - набор значений, которые может принимать j-й признак.

Дерево строится рекурсивно с помощью следующей жадной процедуры, на каждом шаге максимально уменьшая значение функции, описывающей неоднородность данных, содержащихся в узле дерева. Пусть на текущем шаге имеется разбиение пространства признаков на области (ящики) $R_1, R_2, ..., R_M$

Выбираем область R_M.

Выбираем j и c (или L), так, чтобы добиться максимального уменьшения неоднородности, или загрязненности, (impurity) Im_m.

Строим разбиение и повторяем действия.

Разбиения проводятся до тех пор, пока в строящиеся вершины попадает достаточное количество точек обучающей выборки, или пока дерево не достигнет заданной глубины.

Используют различные способы измерить неоднородность, например,

Для задачи классификации:

$$Im_m = \frac{1}{N_m} \mid \{i : x^{(i)} \in R_m, y^{(i)} \neq f(x^{(i)})\} \mid,$$

Для задачи восстановления регрессии:

$$Im_m = \sum_{x^{(i)} \in R_m} \left(y^{(i)} - f(x^{(i)})\right)^2.$$

где N_m - количество точек из обучающей выборки, попавших в область R_m.

Другой важный параметр модели - это глубина дерева решений, регулирующая, "как сильно" будет разбито пространство признаков. Небольшое число разбиений может привести к тому, что построенная модель не будет учитывать некоторые особенности распределения признаков, описывающих те или иные классы, и таким образом приведет к уменьшению точности предсказания. Чрезмерное число разбиений может привести к переобучению модели и снижению её обобщающей способности.

Для того чтобы избежать переобучения, используется процедура отсечений (prunning) [3].

1.1.5. Случайный лес

Один из общих подходов в машинном обучении заключается в использовании композиции "слабых" решающих правил. Итоговое правило строится путем взвешенного голосования ансамбля базовых правил. Для построения базовых правил и вычисления весов в последнее время часто используются две идеи:

Баггинг (bagging – bootstrap aggregation): обучение базовых правил происходит на различных случайных подвыборках данных или/и на различных случайных частях признакового описания; при этом базовые правила строятся независимо друг от друга.

Бустинг (boosting): каждое следующее базовое правило строится с использованием информации об ошибках предыдущих правил, а именно, веса объектов обучающей выборки подстраиваются таким образом, чтобы новое правило точнее работало на тех объектах, на которых предыдущие правила чаще ошибались.

Эксперименты показывают, что, как правило, бустинг работает на больших обучающих выборках, тогда как баггинг – на малых.

Одной из реализаций идеи баггинга является случайный лес [2].

Случайный лес, а точнее – случайные леса (random forests), является одним из наиболее универсальных и эффективных алгоритмов обучения с учителем, применимым как для задач классификации, так и для задач восстановления регрессии. Идея метода [2] заключается в использовании ансамбля из M

деревьев решений (например, $M = 500$), которые обучаются независимо друг от друга. Итоговое решающее правило заключается в голосовании всех деревьев, входящих в состав ансамбля.

Для построения каждого дерева решений используется следующая процедура:

Генерация случайной подвыборки из обучающей выборки путем процедуры изъятия с возвращением (так называемая бутстрэп-выборка). Размер данной подвыборки обычно составляет 50–70% от размера всей обучающей выборки.

Построение дерева решений по данной подвыборке, причем в каждом новом узле дерева переменная для разбиения выбирается не из всех признаков, а из случайно выбранного их подмножества небольшой мощности ρ. Дерево строится до тех пор, пока не будет достигнут минимальный размер листа sz (количество объектов, попавших в него). Рекомендуемые значения: для задачи классификации $\rho = d/3, sz = 1$, для задачи восстановления регрессии $\rho = \sqrt{d}, sz = 3$.

Одной из модификаций метода случайных деревьев является алгоритм крайне случайных деревьев (extremely random forests), в котором на каждом этапе для выбора признака, по которому будет проводиться разбиение, используется вновь сгенерированная случайная бутстрэп-выборка.

Среди достоинств алгоритма случайных деревьев можно выделить высокое качество предсказания, способность эффективно обрабатывать данные с большим числом классов и признаков, внутреннюю оценку обобщающей способности модели. Легко построить параллельную высоко масштабируемую версию алгоритма. Кроме того, доказано, что данный алгоритм не переобучается (с ростом M). Также метод обладает всеми преимуществами деревьев решений, в том числе отсутствием необходимости предобработки входных данных, обработкой как вещественных, так и категориальных признаков, поддержкой работы с отсутствующими значениями.

1.1.6. Градиентный бустинг деревьев решений

Градиентный бустинг деревьев решений (Gradient Boosting Trees – GBT) [6 , 7] – другой универсальный алгоритм машинного обучения, основанный на использовании ансамбля деревьев решений. В отличие от случайного леса градиентный бустинг является развитием бустинг-идеи.

Алгоритм минимизирует эмпирический риск жадным пошаговым алгоритмом, аналогичным методу градиентного спуска. Рассмотрим, например, задачу восстановления регрессии используют. Рассмотрим суммарный штраф на обучающей выборке как функцию от значений решающего правила f в точках $x^{(1)}, x^{(2)}, ..., x^{(N)}$:

$$L(f) = L\left(f(x^{(1)}), f(x^{(2)}), ..., f(x^{(N)}\right) = \sum_{i=1}^{N} L\left(y^{(i)}, f(x^{(i)})\right).$$

Тогда градиент функции $L(f)$ равен

$$gradL(f) = \left(\frac{\partial L\left(y^{(1)}, f(x^{(1)})\right)}{\partial f(x^{(1)})}, \frac{\partial L\left(y^{(2)}, f(x^{(2)})\right)}{\partial f(x^{(2)})}, ..., \frac{\partial L\left(y^{(N)}, f(x^{(N)})\right)}{\partial f(x^{(N)})}\right).$$

На предварительном этапе алгоритм строит оптимальную константную модель $f = g_0$. На m-й итерации конструируется дерево решений g_m (небольшой глубины), аппроксимирующее компоненты вектора антиградиента, вычисленного для текущей модели f. После этого значения в узлах построенного дерева g_m перевычисляются, так, чтобы минимизировать суммарную величину штрафа

$L(f + g_m)$. Далее осуществляем присваивание $f \leftarrow f + v \cdot g_m$, что и завершает m-ю итерацию. Здесь v - параметр регуляризации (shrinkage), призванный бороться с возможным переобучением. Он выбирается из интервала (0,1].

Для решения задачи восстановления регрессии часто используются следующие штрафные функции:

квадратичный штраф

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2,$$

абсолютный штраф

$$L(y, f(x)) = \mid y - f(x) \mid .$$

или функция Хьюбера

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2 = \begin{cases} \frac{1}{2}(y - f(x))^2, \text{при } |y - f(x)| \leq \delta, \\ \delta(|y - f(x)| - \frac{\delta}{2})^2, \text{ при} |y - f(x)| > \delta. \end{cases}$$

Заметим, что функция Хьюбера и квадратичный штраф дифференцируемы всюду, тогда как абсолютный штраф - везде, кроме точек, в которых $y = f(x)$.

Для задачи классификации с K классами метод остается прежним, только вместо одной функции f конструируют сразу K функций $f_k \quad (k = 1, 2, ..., K)$. В качестве штрафа можно использовать кросс-энтропию

$$L(y, f_1(x), y, f_2(x), ..., y, f_K(x) = -\log_2 \rho_y(x),$$

где

$$\rho_y(x) = \frac{e^{f_y(x)}}{\sum_{k=1}^{K} e^{f_y(x)}} \quad (y = 1, 2, ..., K)$$

- есть оценка вероятности того, что $f(x) = y$. Итоговый классификатор определяется как

$$f(x) = agrmax_y \rho_y(x).$$

Более подробное описание алгоритма см. в [6 , 7].

Другим популярным методом, использующим идею бустинга, является алгоритм AdaBoost и его модификации.

1.2. Кластеризация

В задачах обучения без учителя (unsupervised learning) у объектов не известны выходы, и требуется найти некоторые закономерности в данных. К задачам обучения без учителя относят задачи кластеризации, понижения размерности, визуализации и др. Здесь рассматривается только кластеризация.

Задача кластеризации - это задача разбиения заданного набора объектов на кластеры, т. е. группы близких по своему признаковому описанию объектов. "Похожие" друг на друга объекты должны входить в один кластер, "не похожие" объекты должны попасть в разные кластеры.

Близость ("похожесть") объектов измеряется на основе функции расстояния $\rho(x, x') : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$.

1.2.1. Метод центров тяжести

Рассмотрим один из алгоритмов, решающих задачу кластеризации - метод центров тяжестей (K -means). На вход алгоритма поступает набор данных

$$x^{(1)}, x^{(2)}, ..., x^{(N)}, \text{где} x^{(i)} \in \mathbb{X} = Q_1 \times Q_2 \times ... \times Q_d \quad (i = 1, 2, ..., N)$$

и натуральное число K - количество кластеров, на которые нужно разбить данные. Алгоритм реализует пошаговую процедуру минимизации

$$\min_{C, m_k} \sum_{i=1}^{N} \rho(x^{(i)}, m_k,$$

где m_k - центр тяжести объектов, относящихся k -му кластеру:

$$m_k = \frac{\sum_{C(i)=k} x^{(i)}}{| \{i : C)i) = k\} |} \quad (k = 1, 2, ..., K).$$

При этом не гарантируется нахождение глобального минимума. На предварительном этапе строится некоторое разбиение входных данных на K групп (например, случайно). Пусть $C(i)$ - номер группы, к которой принадлежит -й объект. В ходе работы алгоритма значения $C(i)$ обновляются. В конце работы алгоритма значения $C(i)$ будут соответствовать разбиению данных на кластеры. Каждая итерация представляет собой последовательность следующих шагов:

Вычисляем центр тяжести $m_k(k = 1, 2, ..., K)$ объектов в каждой группе.

Для каждого объекта $x^{(i)}$ находим k, для которого расстояние от $x^{(i)}$ до m_k, т. е. $\rho(x^{(i)}, m_k)$, минимально. Обновляем функцию $C(i)$, положив $C(i) = k(i = 1, 2, ..., N)$.

Итерации завершаются, когда наступает стабилизация значений $C(i)$, либо по достижении максимального значения числа итераций.

1.2.2. Метод медиан

Метод центров тяжестей работает с явными описаниями объектов $x^{(i)}$ Модификацией этого метода является метод медиан, или метод срединных точек, (K -medians, или K -medoids). На вход этого алгоритма подается число кластеров K и матрица расстояний $D = (d_{ii'})$, где $d_{ii'} = \rho(x^{(i)}, x^{(i')})(i, i' = 1, 2, ..., N)$. Заметим, что сама функция $\rho(x, x')$ может быть не известна.

Алгоритм ничем не отличается от предыдущего, но вместо центра тяжестей, для каждой группы будем находить медиану, или срединную точку, $m_k = x_{i_k^*}$ где

$$i_k^* = arg \min_{C(i)=k} \sum_{C(i')=k} d_{ii'}^2.$$

Каждая итерация представляет собой последовательность следующих шагов:

Вычисляем медиану $m_k = x_{i_k^*}(k = 1, 2, ..., K)$ объектов в каждой группе.

Для каждого объекта $x^{(i)}$ находим k, для которого d_{i,i_k^*} минимально. Обновляем функцию $C(i)$ положив $C(i) = k(i = 1, 2, ..., N)$.

Как правило, результаты работы алгоритмов центров тяжестей и медиан (если они оба применимы к данным) близки.

Известные классические алгоритмы Machine Learning

Кооперативная коэволюция

Общая идея применения генетических алгоритмов для глобальной оптимизации функционала качества $Q(a)$ в некотором сложно устроенном пространстве $a \in A$ заключается в следующем. Формируется популяция индивидов — конечное множество $\Pi(t) \subset A$. Над ними производятся генетические операции, порождающее новое поколение индивидов. Обычно это бинарная операция рекомбинации (скрещивания) и унарная — мутации. Из большого числа порождённых индивидов в следующее поколение $\Pi(t + 1)$ отбираются только наиболее адаптивные — наилучшие с точки зрения функционала качества $Q(a)$. Эволюционный процесс смены поколений останавливается, когда качество лучшего индивида перестаёт улучшаться.

Для построения композиции обучаемых алгоритмов хорошо подходит специфическая модель эволюции, называемая кооперативной коэволюцией. Как всякий эволюционный алгоритм, он представляет собой итерационный процесс смены поколений.

Метод получил название CCEL — Cooperative Coevolution Ensemble Learner. Он имеет большое число параметров и большую свободу выбора различных эвристик. Инициализация(N_0) — процедура, создающая N_0 индивидов. Каждый индивид формируется случайным образом. Для этого задаются два параметра — вероятность включения объекта q_X и вероятность включения признака q_G. Селекция(Π,N) — генетическая операция, отбирающая N наиболее адаптивных индивидов популяции Π. В методе CCEL она используется дважды. На шаге 5 — для переноса лучших индивидов (элиты) из популяции родителей в популяцию потомков. На шаге 11 — для естественного отбора, при котором из N_1 потомков только N_0 лучших переносятся в следующее поколение. Рекомбинация(Π,N_1) — генетическая операция, порождающая N_1 новых индивидов путём попарного скрещивания индивидов популяции Π. Родительские пары выбираются случайным образом, возможно, с поощрением наиболее адаптивных индивидов. Для формирования потомка в методе CCEL используется так называемый равномерный

кроссинговер (uniform crossover) — каждый бит потомка равновероятно выбирается у одного из родителей. Мутация(П) — генетическая операция, производящая случайные изменения в индивидах популяции П. В качестве параметра алгоритма задаётся вероятность инверсии бита в бинарном коде индивида.

Алгоритм 1.7. Коэволюционное обучение алгоритмической композиции CCEL

Вход:
$T_{\max}$ — максимальное число поколений;
N_0 — размер основной популяции;
N_1 — размер промежуточной популяции;
N_2 — размер элиты, переходящей в следующее поколение без изменений;
Выход:
композиция $a = F(b_1, \ldots, b_p)$;

начальное число популяций: $p(1) := 1$;
создать популяцию: $\Pi_1(1) := \texttt{Инициализация}(N_0)$;
для всех поколений $t := 1, \ldots, T_{\max}$
  для всех популяций $\Pi_j(t)$, $j := 1, \ldots, p(t)$
    породить промежуточную популяцию:
      $\Pi'_j := \texttt{Рекомбинация}(\Pi_j(t), N_1)$;
      $\Pi''_j := \texttt{Мутация}(\Pi'_j) \cup \texttt{Селекция}(\Pi_j(t), N_2)$;
    для всех $v_j \in \Pi''_j$
      оценить адаптивность индивида $\varphi(v_j)$;
      $Q_t := \min\{Q_t, \varphi(v_j)\}$;
      запомнить лучшего индивида в популяции:
        $v^*_j := \arg\min_{v \in \Pi''_j} \varphi(v)$; $b^*_j := \mu(v^*_j)$;
    отобрать лучших индивидов в следующее поколение:
      $\Pi_j(t+1) := \texttt{Селекция}(\Pi''_j, N_0)$;
    если $\texttt{Вклад}(\Pi_j) < \delta$ **то**
      удалить популяцию Π_j; $p(t+1) := p(t) - 1$;
  если $\texttt{Стагнация}(Q, t)$ **то**
    $p(t+1) := p(t) + 1$; добавить $\Pi_{p(t+1)}(t+1) := \texttt{Инициализация}(N_0)$;
  если $\texttt{Останов}(Q, t)$ **то**
    выход;
вернуть композицию $F\left(b^*_1, \ldots, b^*_{p(t)}\right)$, на которой достигается $\min_t Q_t$.

Бэггинг. Метод бэггинга (bagging, bootstrap aggregation) был предложен Л. Брейманом в 1996 году]. Из исходной обучающей выборки длины ℓ формируются различные обучающие подвыборки той же длины ℓ. Эффективность бэггинга объясняется двумя обстоятельствами. Во-первых, благодаря различности базовых алгоритмов, их ошибки взаимно компенсируются при голосовании. Во-вторых, объекты-выбросы могут не попадать в некоторые обучающие подвыборки. Тогда алгоритм, построенный по подвыборке, может оказаться даже точнее алгоритма, построенного по полной выборке. Бэггинг особенно эффективен на малых выборках, когда исключение даже небольшой доли обучающих объектов приводит к построению существенно различных базовых алгоритмов. В случае сверхбольших избыточных выборок приходится строить подвыборки меньшей длины $\ell_0 \ll \ell$, при этом возникает задача подбора оптимального значения ℓ_0.

Алгоритм 1.6. Бэггинг и RSM

Параметры:

ℓ' — длина обучающих подвыборок;
n' — длина признакового подописания;
ε_1 — порог качества базовых алгоритмов на обучении;
ε_2 — порог качества базовых алгоритмов на контроле;

1: **для всех** $t = 1, \ldots, T$, пока не выполнен критерий останова
2: $U :=$ случайное подмножество X^ℓ длины ℓ';
3: $\mathscr{G} :=$ случайное подмножество $\mathscr{F}$ длины n';
4: $b_t := \mu(\mathscr{G}, U)$;
5: **если** $Q(b_t, U) > \varepsilon_1$ или $Q(b_t, X^\ell \setminus U) > \varepsilon_2$ **то**
6: не включать b_t в композицию;

Рассмотрим задачу классификации на два класса, $Y = \{-1,+1\}$. Допустим, что решающее правило фиксировано, $C(b) = \text{sign}(b)$, базовые алгоритмы возвращают ответы −1, 0, +1. Ответ $b_t(x) = 0$ означает, что базовый алгоритм b_t отказывается от классификации объекта x, и ответ $b_t(x)$ не учитывается. При добавлении в композицию слагаемого $\alpha_t b_t(x)$ оптимизируется только базовый алгоритм b_t и коэффициент при нём α_t, а все предыдущие слагаемые $\alpha_1 b_1(x),\ldots,\alpha_{t-1} b_{t-1}(x)$ полагаются фиксированными. Пороговая функция потерь в функционале Q_t аппроксимируется (заменяется) непрерывно дифференцируемой оценкой сверху. Логарифмическая функция связана с принципом максимума правдоподобия и применяется в нейронных сетях и логистической регрессии. Кусочно-линейная аппроксимация связана с принципом максимизации зазора между классами и применяется в методе опорных векторов. Так возник новый вариант бустинга с экспоненциальной аппроксимацией.

Алгоритм 1.1. AdaBoost — построение линейной комбинации классификаторов

Вход:

X^ℓ, Y^ℓ — обучающая выборка; T — максимальное число базовых алгоритмов;

Выход:

базовые алгоритмы и их веса $\alpha_t b_t$, $t = 1, \ldots, T$;

1: инициализация весов объектов: $w_i := 1/\ell$, $i = 1, \ldots, \ell$;
2: **для всех** $t = 1, \ldots, T$, пока не выполнен критерий останова
3: $b_t := \arg\min_b N(b; W^\ell)$;
4: $\alpha_t := \frac{1}{2} \ln \frac{1 - N(b_t; W^\ell)}{N(b_t; W^\ell)}$;
5: пересчёт весов объектов: $w_i := w_i \exp\bigl(-\alpha_t y_i b_t(x_i)\bigr)$, $i = 1, \ldots, \ell$;
6: нормировка весов объектов: $w_0 := \sum_{j=1}^{\ell} w_j$; $w_i := w_i / w_0$, $i = 1, \ldots, \ell$;

Генетический алгоритм

Генетический алгоритм осуществляет поиск наилучшего набора признаков по принципам дарвиновской эволюции. Первое поколение наборов генерируется случайным образом. К этим наборам применяются операции скрещивания и мутации для порождения большого числа новых наборов. Затем производится «искусственный отбор» или селекция: во второе поколение отбираются только B наборов, лучших по заданному внешнему критерию Q. Ко второму поколению также применяются операции скрещивания, мутации и селекции, и порождается третье поколение. Эволюционный процесс переходит от поколения к поколению до тех пор, пока не наступит стагнация, то есть качество лучшего набора в поколении перестанет улучшаться.

Алгоритм 1.6. Выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ генетическим алгоритмом

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметр d;
B — размер популяции;
T — число поколений;
p_m — вероятность мутации;

инициализировать случайную популяцию из B наборов:
$B_1 := B$; $R_1 := \{\mathscr{G}_1^1, \ldots, \mathscr{G}_1^{B_1}\}$;
для всех $t = 1, \ldots, T$, где t — номер поколения:
  отсортировать индивидов в поколении R_t по возрастанию критерия:
  $Q(\mathscr{G}_t^1) \leqslant \ldots \leqslant Q(\mathscr{G}_t^{B_t})$;
  если $B_t > B$ **то**
    операция селекции: оставить только B лучших индивидов в поколении:
    $R_t := \{\mathscr{G}_t^1, \ldots, \mathscr{G}_t^B\}$;
  запомнить, какую сложность имел самый лучший набор:
  $t^* := \arg\min_{s:\, s \leqslant t} Q(\mathscr{G}_s^1)$;
  если $t - t^* \geqslant d$ **то вернуть** $\mathscr{G}_{t^*}^1$;
  породить следующее поколение с помощью операций скрещивания и мутации:
  $R_{t+1} := \{\sim(\mathscr{G}' \times \mathscr{G}'') \mid \mathscr{G}', \mathscr{G}'' \in R_t\} \cup R_t$;

Список литературы

[1] Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности. — М.: Финансы и статистика, 1989.

[2] Айвазян С. А., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: исследование зависимостей. — М.: Финансы и статистика, 1985.

[3] Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970. — 320 pp.

[4] Вапник В. Н. Восстановление зависимостей по эмпирическим данным. — М.: Наука, 1979.

[5] Вапник В. Н., Червоненкис А. Я. О равномерной сходимости частот появления событий к их вероятностям // ДАН СССР. — 1968. — Т. 181, № 4. — С. 781–784.

[6] Вапник В. Н., Червоненкис А. Я. О равномерной сходимости частот появления событий к их вероятностям // Теория вероятностей и ее применения. — 1971. — Т. 16, № 2. — С. 264–280.

[7] Вапник В. Н., Червоненкис А. Я. Теория распознавания образов. — М.: Наука, 1974.

[8] Головко В. А. Нейронные сети: обучение, организация и применение. — М.: ИПРЖР, 2001.

[9] Епанечников В. А. Непараметрическая оценка многомерной плотности вероятности // Теория вероятностей и её применения. — 1969. — Т. 14, № 1. — С. 156–161.

[10] Ермаков С. М., Михайлов Г. А. Курс статистического моделирования. — М.: Наука, 1976.

[11] Мазуров Вл. Д. О построении комитета системы выпуклых неравенств. — Кибернетика, 1967. — №2.

[12] Мазуров Вл. Д. О некоторых математических конструкциях для распознавания образов // Нелинейное программирование и приложения. — Свердловск, 1979.

[13] Мазуров Вл. Д., Казанцев В. С., Белецкий Н. Г., Мезенцев С. В., Сачков Н. О. Пакет КВАЗАР прикладных программ распознавания образов. — Свердловск, 1979.

[14] Нильсон Н. Обучающие машины. — М., 1968.

[15] Черников С. Н. Свёртывание конечных систем линейных неравенств. - ДАН УССР, 1969. - № 1.

[16] Яненко Н. Н. Проблемы математической технологии. — Тезисы доклада меж»тнародной конференции «Структура и организация пакетов программ». — Тбилиси, 1976.

[17] Журавлёв Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации. — Проблемы кибернетики. — 1978. — № 3.

[18] Айзерман М. А., Браверманн Э. М., Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. — М., 1970.

[19] Горелик А. Л., Скрипкин В. А. Построение систем распознавания. — М., 1974.

[20] Бонгард М. М. Проблема узнавания. — М., 1967.

[21] Загоруйко Н. Г. Методы распознавания и их применение. - м., 1972.

[22] Schapire R. E. The strength of weak learnability // Machine Learning.— 1990.— Vol. 5.— Pp. 197–227. http://citeseer.ist.psu.edu/schapire90strength.html.

[23] Skurichina M., Duin R. P. W. Limited bagging, boosting and the random subspace method for linear classifiers // Pattern Analysis & Applications.— 2002.— no. 5.— Pp. 121–135. http://citeseer.ist.psu.edu/skurichina02limited.html.

[24] Tresp V. Committee machines // Handbook for Neural Network Signal Processing / Ed. by Y. H. Hu, J.-N. Hwang. — CRC Press, 2001. http://citeseer.ist.psu.edu/tresp01committee.html.

[25] Tresp V., Taniguchi M. Combining estimators using non-constant weighting functions // Advances in Neural Information Processing Systems / Ed. by G. Tesauro, D. Touretzky, T. Leen. — Vol. 7. — The MIT Press, 1995.— Pp. 419–426. http://citeseer.ist.psu.edu/tresp95combining.html.

[26] Vapnik V. Statistical Learning Theory. — Wiley, New York, 1998.

[27] Vapnik V. The nature of statistical learning theory. — 2 edition. — Springer-Verlag, New York, 2000.

[28] Waterhouse S. R., Robinson A. J. Classification using hierarchical mixtures of experts // Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing IV. — Long Beach, CA: IEEE Press, 1994.— Pp. 177–186. http://citeseer.ist.psu.edu/waterhouse94classification.html.

[29] Wolpert D. H. Stacked generalization // Neural Networks.— 1992.— no. 5.— Pp. 241–259. http://citeseer.ist.psu.edu/wolpert92stacked.html.

[30]