Лекции по методам оценивания и выбора моделей

К. В. Воронцов

24 июня 2010 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по адресу `vokov@forecsys.ru`, либо высказанные в обсуждении страницы «Машинное обучение (курс лекций, К.В.Воронцов)» вики-ресурса `www.MachineLearning.ru`.

Содержание

1 Оценивание и выбор моделей

Практически в любой задаче классификации, регрессии или прогнозирования возникают вопросы: какие признаки использовать, а какие нет; нужно ли как-то преобразовывать исходные признаки; какую модель зависимости применить в данной задаче? Все эти вопросы имеют одну общую черту: как только ответы на них даны, становится предельно понятно, что делать дальше — настраивать параметры выбранной модели по обучающей выборке. Именно эти вопросы, связанные с выбором и конструированием моделей алгоритмов, и рассматриваются в данной главе.

Проблема *отбора признаков* (features selection) часто возникает из-за того, что на этапах постановки задачи и формирования данных ещё не ясно, какие признаки бесполезны или дублируют друг друга. Естественное стремление учесть как можно больше потенциально полезной информации приводит к появлению избыточных (шумовых) признаков. Однако если признак на самом деле не информативен, то есть не влияет на ответы, его включение в модель может только ухудшить её качество. Методы обучения должны отличать шумовые признаки и отбрасывать их.

По мере увеличения числа используемых признаков (сложности модели) средняя ошибка на обучающей выборке, как правило, монотонно убывает. При этом средняя ошибка на независимых контрольных данных сначала уменьшается, затем проходит через точку минимума и далее только возрастает. Это явление называют *переобучением*. В чрезмерно сложных моделях избыточные степени свободы «расходуются» не только на восстановление искомой зависимости, но и на подгонку под конкретные данные, фактически, на аппроксимацию погрешностей измерений и самой модели. Переобучение возникает в большинстве практических задач, независимо от предметной области. Отбор признаков позволяет находить модель оптимальной сложности, при которой переобучение минимально.

Отбор признаков сокращает стоимость сбора информации. В практических задачах затраты на измерение или вычисление отдельных признаков могут быть сопоставимы со стоимостью потерь от ошибочных прогнозов. Аккуратный учёт всех потерь позволяет отсеять не только шумовые, но и малоинформативные признаки.

Наконец, отбор признаков приводит к более простым и понятным моделям и повышает скорость выполнения алгоритмов.

Сложность задачи отбора признаков — в её переборном характере. Если число признаков равно n, то число непустых подмножеств составляет $2^n - 1$. Прямой перебор всех подмножеств приводит к комбинаторному взрыву и оказывается невозможным уже при n порядка 20 даже на самых современных компьютерах. Эффективные методы сокращения перебора рассматриваются в §1.2.

Синтез признаков (features extraction) — это второй подход к сокращению размерности. Он состоит в том, чтобы найти преобразование исходного пространства признаков в новое пространство существенно меньшей размерности. При этом не должно происходить потери информации, то есть описание любого объекта в исходном пространстве должно восстанавливаться с заданной точностью по его описанию в новом пространстве. Самый простой случай — линейное преобразование. Недостаток данного подхода в том, что новые признаки могут оказаться плохо интерпретируемыми. Методы синтеза признаков рассматриваются в §1.3.

Отбор признаков является специфическим случаем более общей задачи *выбора модели* (model selection) из некоторой совокупности моделей. На практике чаще всего

возникает задача выбора модели из небольшого числа моделей-претендентов. Это довольно простая задача, она рассматривается в 1.4.1. Гораздо более сложной является проблема *выбора структуры модели* (structure selection). Под структурой обычно понимается описание искомого алгоритма как суперпозиции некоторых элементарных функций. Вариантов построения суперпозиции ещё больше, чем вариантов отбора признаков. Поэтому проблема сокращения перебора стоит здесь ещё острее. Некоторые подходы к решению этой задачи изложены в §1.5.

В задачах отбора признаков, моделей и структур важную роль играют понятия *внутренних и внешних критериев*. С них мы и начнём в 1.1.1.

Основные обозначения. Пусть X — пространство объектов; Y — множество ответов; $y^*\colon X \to Y$ — целевая зависимость, 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 $a\colon X \to Y$, аппроксимирующий целевую зависимость y^* на всём множестве X.

Задан функционал средней ошибки алгоритма a на выборке X^ℓ:

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

где функция потерь $\mathscr{L}(a, x)$ характеризует величину ошибки алгоритма a на объекте x. Обычно полагают $\mathscr{L}(a, x) = [a(x) \neq y^*(x)]$ в случае классификации и $\mathscr{L}(a, x) = (a(x) - y^*(x))^2$ в случае регрессии. Другие варианты функции потерь обсуждались в разделе **??**.

Моделью алгоритмов называется параметрическое семейство отображений A, из которого выбирается искомый алгоритм $a(x)$.

Методом обучения называется отображение $\mu\colon X^\ell \mapsto a$, которое произвольной обучающей выборке X^ℓ ставит в соответствие некоторой алгоритм $a\colon X \to Y$ из заданной модели алгоритмов A.

Задача *выбора метода* состоит в том, чтобы в заданном множестве методов обучения M найти метод μ, выдающий алгоритмы с наилучшей обобщающей способностью. Перечислим некоторые частные постановки этой задачи.

1. Задача *выбора модели* (model selection). Имеется конечное множество альтернативных моделей $A_1, \ldots, A_T$, каждая со своим методом обучения, $M = \{\mu_1, \ldots, \mu_T\}$. Требуется найти модель, наиболее адекватную для данной выборки.

2. Задача *настройки гиперпараметра*. Имеется одна модель A, и один метод обучения μ_ω с параметром ω, который не может быть настроен по обучающей выборке. Например, в полиномиальной регрессии попытка оптимизировать степень полинома по обучающей выборке приведёт к выбору максимально возможной степени и переобучению. Такие параметры называют *управляющими*, *внешними* или *гиперпараметрами*. Требуется подобрать наилучшее значение гиперпараметра. В этом случае $M = \{\mu_\omega\colon \omega \in \Omega\}$, где Ω — множество допустимых значений гиперпараметра.

3. Задача *отбора признаков*. Имеется метод обучения $\mu_\mathscr{G}$, использующий только признаки из заданного подмножества $\mathscr{G} \subseteq \mathscr{F} = \{f_1, \ldots, f_n\}$. Требуется найти набор признаков, при котором алгоритм $a = \mu_\mathscr{G}(X^\ell)$ имеет наилучшую (или хотя бы достаточно хорошую) обобщающую способность. Здесь $M = \{\mu_\mathscr{G}\colon \mathscr{G} \subseteq \mathscr{F}\}$.

§1.1 Критерии выбора модели

Прежде, чем говорить о способах выбора метода обучения $\mu \in M$, необходимо сформулировать критерии выбора.

1.1.1 Внутренние и внешние критерии

Внутренний критерий — это функционал $Q_{\text{int}}(\mu, X^\ell)$, характеризующий качество метода μ по обучающей выборке X^ℓ. Каноническим примером внутреннего критерия является функционал *ошибки обучения* (training error):

$$Q_{\text{int}}(\mu, X^\ell) = Q\big(\mu(X^\ell), X^\ell\big).$$

Внутренние критерии используются для настройки параметров выбранной модели алгоритмов A. Например, метод *минимизации эмпирического риска* строит алгоритм, доставляющий минимальное значение внутреннему критерию:

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

Внутренние критерии нельзя использовать для выбора метода обучения, так как при этом будет поощряться переобучение.

Внешний критерий характеризует качество метода μ по тем данным, которые не использовались в процессе обучения. Внешний критерий проверяет, действительно ли полученный алгоритм хорошо работает в реальных условиях.

Идея применения внешних критериев для подбора оптимальной структуры модели была предложена А. Г. Ивахненко в конце 60-х в *методе группового учёта аргументов, МГУА* (group method of data handling, GMDH) [3]. Это один из успешных методов, с помощью которого были решены сотни прикладных задач. В частности, опыт МГУА показал, что имеет смысл использовать сразу несколько внешних критериев, характеризующих качество метода обучения с различных точек зрения.

Мы рассмотрим наиболее известные типы внешних критериев.

Будем полагать, что все критерии, будь то внешние или внутренние, требуется минимизировать. Чем меньше значение критерия $Q(\mu)$, тем выше качество метода μ.

1.1.2 Критерий средней ошибки на контрольных данных

Простейшим примером внешнего критерия является функционал средней ошибки на заданной контрольной выборке X^k, называемый *ошибкой обобщения* (generalization error). Разумеется, на объектах $x_i \in X^k$ также должны быть известны правильные ответы $y_i = y^*(x_i)$. Поэтому будем считать, что исходная *полная* выборка $X^L = X^\ell \cup X^k$ некоторым образом разбита на обучающую и контрольную части, $L = \ell + k$. Внешний критерий является функцией метода μ и полной выборки X^L:

$$Q_{\text{ext}}(\mu, X^L) = Q\left(\mu(X^\ell), X^k\right).$$

В МГУА его принято называть *критерием регулярности*, в англоязычной литературе — *ошибкой на отложенных данных* (hold-out error).

Выборки X^ℓ и X^k должны быть не только непересекающимися, но и независимыми. Критерий фактически останется внутренним, если контрольная выборка X^k

будет специально составлена из объектов, совпадающих или незначительно отличающихся от объектов обучения X^ℓ. На практике имеющуюся выборку данных X^L разбивают на обучение и контроль случайным образом. В контрольной выборке, как правило, оставляют от четверти до половины объектов.

1.1.3 Критерий скользящего контроля

Это критерий является обобщением предыдущего. Чтобы результат не зависел от способа разбиения, берут несколько различных разбиений исходной выборки X^L на обучение и контроль $X^L = X_n^\ell \cup X_n^k$, $n = 1, \ldots, N$, и среднюю ошибку на контроле усредняют по разбиениям. Этот функционал называется ошибкой *скользящего контроля* (cross-validation error, CV):

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

Обычно после выбора метода μ по критерию скользящего контроля окончательный алгоритм ещё раз обучают по полной выборке: $a = \mu(X^L)$. Но можно также выбрать в качестве решения лучший из алгоритмов, уже построенных по подвыборкам, что не требует дополнительного применения метода μ:

$$a = \arg\min_{n=1,\ldots,N} Q\big(\mu(X_n^\ell), X^L\big).$$

В зависимости от способа формирования разбиений различают несколько видов скользящего контроля.

Полный скользящий контроль (complete CV) строится по всем $N = C_L^k$ разбиениям. Это число становится слишком большим уже при $k > 2$, поэтому полный скользящий контроль используется либо в теоретических исследованиях, либо в тех редких случаях, когда для него удаётся вывести эффективную вычислительную формулу. Например, для метода k ближайших соседей такая формула получена в [9]. На практике чаще применяются другие разновидности скользящего контроля.

Контроль по отдельным объектам (leave-one-out CV) является частным случаем полного скользящего контроля при $k = 1$:

$$\mathrm{LOO}(\mu, X^L) = \frac{1}{L} \sum_{i=1}^{L} Q\left(\mu(X^L \setminus \{x_i\}), \{x_i\}\right).$$

Это, пожалуй, самый распространённый вариант скользящего контроля. Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.

Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится L раз. Некоторые методы обучения позволяют достаточно быстро перенастраивать внутренние параметры алгоритма при замене одного обучающего объекта другим. В этих случаях вычисление LOO удаётся заметно ускорить.

Контроль по q блокам (q-fold CV). Выборка случайным образом разбивается на q непересекающихся блоков одинаковой (или почти одинаковой) длины $l_1, \ldots, l_q$:

$$X^L = X_1^{l_1} \cup \cdots \cup X_q^{l_q}, \quad l_1 + \cdots + l_q = L.$$

Каждый блок по очереди становится контрольной подвыборкой, при этом обучение производится по остальным $q-1$ блокам. Критерий определяется как средняя по всем блокам ошибка на контроле:

$$Q_{\text{ext}}(\mu, X^L) = \frac{1}{q} \sum_{n=1}^{q} Q\big(\mu(X^L \setminus X_n^{l_n}), X_n^{l_n}\big).$$

Это компромисс между LOO и hold-out. С одной стороны, обучение производится только q раз вместо L. С другой стороны, длина обучающих подвыборок $L\frac{q-1}{q}$ (с точностью до округления) не сильно отличается от длины полной выборки L. Обычно выборку разбивают случайным образом на 10 или 20 блоков.

Контроль по случайным подвыборкам. Разбиения $n = 1, \ldots, N$ выбираются случайно, независимо и равновероятно из множества всех C_L^k разбиений. Обозначим $Q_n = Q\left(\mu(X_n^\ell), X_n^k\right)$. Если случайная величина $Q = Q\left(\mu(X^\ell), X^k\right)$ имеет непрерывное распределение, то вероятность события $Q > \max_n Q_n$ не превосходит $\frac{1}{N+1}$. Аналогично можно оценить и двусторонний *доверительный интервал*: вероятность того, что $Q \notin [\min_n Q_n, \max_n Q_n]$ не превосходит $\frac{2}{N+1}$. Таким образом, для получения верхней оценки с надёжностью 0.95 достаточно взять $N = 19$, а для двусторонней оценки $N = 39$ разбиений.

Бутстрэп (bootstrap) напоминает контроль по случайным подвыборкам. Отличия в том, что объекты выбираются с возвращением, длина обучающих подвыборок берётся равной длине полной выборки, при этом в подвыборке образуются повторы.

Проблема представительности подвыборок возникает во всех критериях, использующих случайные разбиения. Обучающие и контрольные подвыборки должны обладать теми же статистическими характеристиками, что и полная выборка X^L. В противном случае выбор модели и настройка её параметров будут плохо согласованы друг с другом.

В задачах классификации рекомендуется сохранять в каждой подвыборке те же пропорции распределения объектов по классам, что и на всей выборке. Этот приём называется *стратификацией* (stratification) выборки.

Кроме того, необходимо обеспечить равномерное распределение каждой подвыборки по всему пространству X. На практике поступают следующим образом. Полная выборка X^L упорядочивается по некоторому специально выделенному признаку, и каждый $\left[(i-\frac{1}{2})\frac{L}{k}\right]$-й объект, $i = 1, \ldots, k$, заносится в контрольную часть. Аналогично осуществляется распределение выборки по блокам в случае q-fold CV. Выделенный признак определяется исходя из особенностей задачи. Это может быть некоторый исходный признак $f_j(x_i)$, линейная комбинация нескольких исходных признаков (например, первая главная компонента, см. раздел **??**), целевой признак y_i, оценка уровня шума целевого признака y_i, расстояние от вектора признакового описания объекта x_i до центра масс выборки, и т. д.

1.1.4 Критерии непротиворечивости

Эта группа критериев основана на следующей идее, также идущей от МГУА. Если модель алгоритмов A и метод обучения μ подобраны правильно, то настройка параметров модели по различным представительным подвыборкам должна приводить к одинаковым или почти одинаковым алгоритмам. Говорят также об *устойчивости модели* относительно состава выборки. В МГУА это свойство называется *помехоустойчивостью моделирования*.

В простейшем случае критерий непротиворечивости определяется как средняя невязка ответов двух алгоритмов, построенных по двум случайным непересекающимся подвыборкам одинаковой или почти одинаковой длины, $X^\ell \cup X^k = X^L$:

$$Q_{\text{ext}}(\mu, X^L) = \frac{1}{L}\sum_{i=1}^{L} \big|a_1(x_i) - a_2(x_i)\big|, \quad a_1 = \mu(X^\ell), \quad a_2 = \mu(X^k).$$

В более сложных вариантах критерия выборка разбивается несколькими различными способами, как при скользящем контроле.

Различие двух алгоритмов не обязательно измерять как невязку их ответов на выборке. Если α_1, α_2 — векторы параметров алгоритмов a_1, a_2, то внешний критерий можно определить как расстояние между этими векторами в соответствующей метрике ρ:

$$Q_{\text{ext}}(\mu, X^L) = \rho(\alpha_1, \alpha_2).$$

Необходимым условием для применения этих критериев является избыточность исходных данных. Половина выборки должна быть достаточно представительной, чтобы по ней можно было построить алгоритм приемлемого качества. Поэтому критерии данного типа не рекомендуется применять в случае малых выборок.

1.1.5 Критерии регуляризации

Эти критерии вводятся в тех случаях, когда задача обучения алгоритма по выборке оказывается неустойчивой — многие алгоритмы доставляют внутреннему критерию значение, близкое к оптимальному, однако далеко не все из них обладают хорошей обобщающей способностью.

Идея регуляризации заключается в том, чтобы наложить ограничения на вектор параметров алгоритма, либо ввести штраф за выход вектора параметров из некоторой допустимой области. Критерии регуляризации не так универсальны, как предыдущие — их вид зависит от конкретной модели алгоритмов.

Например, в линейных моделях регрессии и классификации резкое увеличение нормы вектора параметров $\|\alpha\|$ в алгоритме $a = \mu(X^\ell)$, как правило, свидетельствует о переобучении. При этом модель становится неадекватной, появляются большие по модулю отрицательные и положительные коэффициенты, которые уже нельзя интерпретировать как степень важности соответствующего признака. Поэтому в качестве внешнего критерия регуляризации берут сумму внутреннего критерия и штрафного слагаемого:

$$Q_{\text{ext}}(\mu, X^\ell) = Q_{\text{int}}(\mu, X^\ell) + \tau\|\alpha\|,$$

где множитель τ называется *параметром регуляризации*. При $\tau \to 0$ решение неустойчиво; при $\tau \to \infty$, наоборот, вырождается в константу. Подбор τ позволяет найти компромиссное между двумя крайностями. С функционалами такого вида мы уже сталкивались в **??** и **??** при обсуждении проблемы мультиколлинеарности в линейном дискриминанте Фишера и многомерной линейной регрессии.

Преимущество этого критерия, по сравнению со скользящем контролем, в том, что нет необходимости многократно применять ресурсоёмкий метод обучения. Основная проблема — необходимость подбирать значение параметра регуляризации τ.

На практике можно упростить этот критерий и следить только за тем, чтобы норма $\|\alpha\|$ не выходила в область слишком больших значений. Однако и в этом случае необходимо задавать априорный параметр $\alpha_{\max}$:

$$Q_{\text{ext}}(\mu, X^L) = \|\alpha\| \leqslant \alpha_{\max}.$$

При увеличении размерности вектора параметров α функционалы регуляризации, как правило, начинают с некоторого момента резко расти, что свидетельствует о переобучении.

1.1.6 Критерии, основанные на оценках обобщающей способности

Теория Вапника-Червоненкиса даёт верхние оценки частоты ошибок на контрольной выборке, которые можно использовать в качестве внешнего критерия [1]:

$$Q(\mu(X^\ell), X^k) < Q(\mu(X^\ell), X^\ell) + \sqrt{\frac{h}{\ell}\left(\ln\frac{2\ell}{h} + 1\right) - \frac{\ln\eta}{\ell}}, \tag{1.1}$$

где h — *размерность Вапника-Червоненкиса* (*ёмкость*) модели алгоритмов; η — уровень значимости — вероятность, с которой данная оценка имеет право нарушаться. Функция потерь $\mathscr{L}(a, x)$ в функционале $Q(a, X^\ell)$, обязана быть двузначной и принимать только значения 0 или 1. Для линейной модели классификации ёмкость h совпадает с размерностью пространства параметров и с числом признаков n.

Информационный критерий Акаике является оценкой матожидания средней ошибки на независимых контрольных данных. Он выводится из предположений, что модель алгоритмов линейна, размерность вектора параметров равна n, и функционал Q соответствует принципу максимума правдоподобия:

$$Q(a, X^\ell) = -\frac{1}{\ell}\sum_{i=1}^{\ell} \ln p(x_i, a(x_i)).$$

Это означает, что задана *вероятностная функция потерь* $\mathscr{L}(a, x) = -\ln p(x, a(x))$, где $p(x, y)$ — плотность вероятностного распределения на множестве $X \times Y$, согласно которому и получена обучающая выборка $(x_i, y_i)_{i=1}^\ell$.

Информационный критерий Акаике (Akaike Information Criterion, AIC) [8]:

$$\text{AIC}(\mu, X^\ell) = Q(\mu(X^\ell), X^\ell) + \frac{2\hat{\sigma}^2}{\ell} n,$$

где $\hat{\sigma}^2$ — оценка дисперсии случайной величины $\xi(x) = y^*(x) - a^*(x)$, представляющей собой отклонение наилучшего в рамках используемой модели алгоритма a^* от неизвестной целевой функции y^*. В частности, для задач классификации можно воспользоваться оценкой $\hat{\sigma}^2 \leqslant \frac{1}{4}$.

На практике критерий AIC часто применяют и к нелинейным моделям, что не всегда хорошо обосновано, но во многих случаях приводит к выбору моделей вполне приемлемого качества.

Байесовский информационный критерий (Bayesian Information Criterion, BIC), так же, как и AIC, вытекает из принципа максимума правдоподобия. Линейность модели не предполагается, тем не менее, число параметров n всё равно возникает благодаря использованию аппроксимации Лапласа [8]:

$$\mathrm{BIC}(\mu, X^\ell) = \frac{\ell}{\hat{\sigma}^2}\left(Q(\mu(X^\ell), X^\ell) + \frac{\hat{\sigma}^2 \ln \ell}{\ell} n\right).$$

При $\ell \geqslant 8$ критерий BIC склонен сильнее штрафовать сложные модели, чем AIC.

Особенностью критерия BIC является то, что он не только позволяет выбрать лучшую модель, но и даёт оценку апостериорной вероятности каждой модели. Если выбор производился из T моделей $A_1, \ldots, A_T$, то вероятность p_t, что данные X^ℓ были порождены моделью A_t, даётся формулой Байеса:

$$p_t = \frac{\exp\left(-\frac{1}{2}\mathrm{BIC}_t\right)}{\sum_{s=1}^{T} \exp\left(-\frac{1}{2}\mathrm{BIC}_s\right)} \equiv \mathrm{SoftMax}_t(\mathrm{BIC}_1, \ldots, \mathrm{BIC}_T).$$

Обобщение. Все три критерия имеют схожий вид — это сумма внутреннего критерия и штрафного слагаемого, наказывающего чрезмерно сложные модели. В отличие от критериев регуляризации, они не содержат параметра τ. В то же время, между различными критериями нет согласия относительно того, какой множитель должен стоять перед штрафным слагаемым, в какую степень возводить параметр размерности модели, и как вообще определять понятие размерности. В результате возвращаемся к той же проблеме: имеется внешний критерий общего вида

$$Q_{\mathrm{ext}}(\mu, X^\ell) = Q_{\mathrm{int}}(\mu, X^\ell) + \tau\big(\dim \mu(X^\ell)\big)^\gamma,$$

в котором параметры τ и γ необходимо задать из априорных соображений.

1.1.7 Выбор метода по совокупности критериев

В МГУА рекомендуется использовать для выбора оптимальной модели несколько принципиально разных внешних критериев. Как правило, один внешний критерий отбирает несколько лучших методов, качество которых не отличается в пределах дисперсии критерия. Образуется дополнительная свобода выбора, которой разумно распорядиться с помощью второго внешнего критерия.

Дугой вариант — вычислять взвешенную сумму нескольких критериев, но это вызывает проблему выбора весовых коэффициентов. Агрегированный критерий зависит от них существенно, а из каких соображений их назначать — не ясно.

Более приемлемым представляется двухступенчатый отбор. Практическая рекомендация — отобрать некоторое количество лучших методов по критерию скользящего контроля; затем из них выбрать тот, для которого критерий регуляризации (либо критерий непротиворечивости) принимает наименьшее значение.

§1.2 Отбор информативных признаков

Будем считать, что объекты описываются набором признаков $\mathscr{F} = \{f_1, \ldots, f_n\}$. Каждый признак f_j — это отображение из X в некоторое множество D_j допустимых значений признака, в общем случае не обязательно числовое. Вектор $\big(f_1(x), \ldots, f_n(x)\big) \in D_1 \times \ldots \times D_n$ называется *признаковым описанием* объекта x.

Пусть $\mathscr{G} \subseteq \mathscr{F}$ — произвольное подмножество признаков.

Будем обозначать через $\mu_{\mathscr{G}}$ метод обучения, который строит алгоритмы, используя только признаки из подмножества $\mathscr{G}$. Будем предполагать, что метод $\mu_{\mathscr{G}}$ выбирает алгоритм из модели алгоритмов $A(\mathscr{G})$, использующей только признаки из $\mathscr{G}$. Число используемых признаков $|\mathscr{G}|$ будем называть *сложностью модели* $A(\mathscr{G})$.

Основное отличие внешних и внутренних критериев. По мере увеличения сложности модели $|\mathscr{G}|$ внутренний критерий $Q_{\text{int}}(\mathscr{G}) \equiv Q_{\text{int}}(\mu_{\mathscr{G}}, X^L)$, как правило, монотонно убывает. Внешний критерий $Q_{\text{ext}}(\mathscr{G}) \equiv Q_{\text{ext}}(\mu_{\mathscr{G}}, X^L)$ сначала убывает, затем проходит через точку минимума и далее только возрастает, Рис. **??**. Это типичное поведение критериев подтверждается как теоретически, так и экспериментально.

Полезно построить на одном графике кривые внутреннего критерия и нескольких внешних критериев. Иногда оказывается, что минимумы внешних критериев достигаются не только на различных моделях $\mathscr{G}$, но даже при различных значениях сложности $|\mathscr{G}|$. Это вполне естественно, так все критерии имеют некоторую дисперсию. График позволяет оценить уровень шума визуально и определить интервал допустимых значений сложности, в котором Q_{ext} незначимо отличается от минимума. Применение совокупности внешних критериев позволяет найти пересечение этих интервалов и с большей уверенностью определить оптимальную модель.

В разделах 1.2.1–1.2.8 будут рассмотрены наиболее общие способы отбора признаков, совместимые с произвольными методами обучения и с произвольными функционалами качества, и потому одинаково применимые как для классификации, так и для регрессии.

Во всех методах строится нижняя огибающая множества точек $\big(|\mathscr{G}|, Q_{\text{ext}}(\mathscr{G})\big)$. В случае полного перебора её минимум соответствует оптимальному набору $\mathscr{G}^*$ оптимальной сложности $j^* = |\mathscr{G}^*|$. Остальные алгоритмы решают задачу поиска оптимального набора признаков лишь приближённо.

1.2.1 Полный перебор

Пусть $Q(\mathscr{G})$ — заданный внешний критерий. Определим *нижнюю огибающую критерия* Q как наилучшее значение критерия для моделей сложности j:

$$Q(j) = \min_{\mathscr{G}:\, |\mathscr{G}|=j} Q(\mathscr{G}).$$

Алгоритм 1.1. FullSearch: выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ полным перебором

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметр d;

1: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
2: найти лучший набор сложности j:
$\mathscr{G}_j := \underset{\mathscr{G} \subseteq \mathscr{F}:\ |\mathscr{G}|=j}{\arg\min}\ Q(\mathscr{G})$;
3: запомнить, какую сложность имел самый лучший набор:
$j^* := \underset{s:\ s \leqslant j}{\arg\min}\, Q(\mathscr{G}_s)$;
4: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

Алгоритм 1.1 осуществляет полный перебор всевозможных наборов признаков $\mathscr{G}$ в порядке возрастания сложности. Для каждого значения сложности j строятся все возможные наборы $\mathscr{G}: |\mathscr{G}| = j$, и для каждого набора оценивается качество обучения с помощью заданного внешнего критерия Q. На шаге 2 выбирается лучший набор $\mathscr{G}_j$. На шаге 3 запоминается значение сложности j^*, при котором был получен наилучший набор. Если улучшить этот результат не удаётся на протяжении d итераций, то считается, что минимум внешнего критерия $Q(j)$ пройден, и алгоритм заканчивает работу, выдавая наилучший набор $\mathscr{G}_{j^*}$. Таким образом, некоторое сокращение полного перебора всё-таки происходит за счёт того, что слишком длинные наборы вообще не оцениваются. Число d является единственным параметром алгоритма. Правило останова, применённое на шаге 4, в дальнейшем будет использоваться и в более сложных алгоритмах.

Достоинства и недостатки. Алгоритм полного перебора наиболее прост для реализации и гарантирует, что будет найден наилучший набор. Однако его практическая применимость ограничена задачами с небольшим числом признаков n, так как время полного перебора растёт со скоростью 2^n. На современных компьютерах удаётся решать задачи с числом признаков не более 20–25. Примечательно, что данная рекомендация практически не изменилась за последние 30–40 лет: раньше речь шла о 15–20 признаках. Проблема «комбинаторного взрыва» в принципе не решается простым наращиванием вычислительных мощностей.

Алгоритм 1.1 является базовым для всех остальных эвристических алгоритмов, нацеленных на сокращение перебора. Ни один из них не гарантирует, что будет найдено наилучшее решение. Однако на практике они часто находят наборы, качество которых не сильно отличается от оптимального.

1.2.2 Последовательное добавление признаков

Алгоритм 1.3 добавляет к набору $\mathscr{G}$ по одному признаку, каждый раз выбирая тот признак, который приводит к наибольшему уменьшению внешнего критерия. Это простая стратегия жадного наискорейшего спуска. В литературе этот алгоритм получил название Add — столь же незатейливое, как и сам алгоритм.

Алгоритм 1.2. Add: выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ жадным добавлением

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметр d;

1: $\mathscr{G}_0 := \varnothing$;
2: **для всех** $j = 1, \ldots, n$, где j — сложность наборов:
3: найти признак, наиболее выгодный для добавления:
$f^* := \arg\min_{f \in \mathscr{F} \setminus \mathscr{G}_{j-1}} Q\big(\mathscr{G}_{j-1} \cup \{f\}\big)$;
4: добавить этот признак в набор:
$\mathscr{G}_j := \mathscr{G}_{j-1} \cup \{f^*\}$;
5: запомнить, какую сложность имел самый лучший набор:
$j^* := \arg\min_{s:\, s \leqslant j} Q(\mathscr{G}_s)$;
6: **если** $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

Достоинства и недостатки. Алгоритм Add сокращает трудоёмкость[1] перебора с $O(2^n)$ операций до $O(n^2)$, точнее $O(n(j^* + d))$. Другое его достоинство в том, что некоторые методы обучения позволяют быстро пересчитывать внутренние параметры алгоритма при добавлении одного признака. Например, таким свойством обладают линейные алгоритмы регрессии и классификации. Идея состоит в том, что при добавлении столбца пересчёт псевдообратной матрицы требует лишь $O(\ell^2)$ операций, а не $O(\ell^3)$, как при полной перенастройке.

Недостатки Add естественно вытекают из неоптимальности жадной стратегии. Алгоритм Add склонен включать в набор лишние признаки, причём если признак был единожды включён, он остаётся в наборе навсегда, хотя после него в набор могли попасть признаки, способные его заменить.

Последовательное удаление признаков. Наряду с Add существует двойственная жадная стратегия Del, последовательно удаляющая избыточные признаки. Итерации начинаются с $\mathscr{G}_0 := \mathscr{F}$, и далее из набора последовательно исключается по одному признаку так, чтобы значение критерия Q убывало как можно быстрее. Стратегия Del работает медленнее, так как на начальных итерациях приходится обучать алгоритм по всем или почти всем признакам. Её применяют в тех случаях, когда заранее известно, что информативных признаков гораздо больше, чем шумовых.

Пример 1.1. Рассмотрим задачу многомерной линейной регрессии. Пусть три линейно независимых вектора f_1, f_2, f_3 из $\mathbb{R}^\ell$ соответствуют трём различным признакам. Целевой вектор $y \in \mathbb{R}^\ell$ является линейной комбинацией f_1 и f_2, но по-отдельности слабо коррелирует с ними. Допустим, что y немного сильнее коррелирует с f_1, и существенно сильнее — с f_3. Тогда оптимальным набором признаков является $\{f_1, f_2\}$. В то же время, алгоритм жадного добавления Add построит следующую последовательность наборов: $\{f_3\}$, $\{f_3, f_1\}$, $\{f_3, f_1, f_2\}$. Ни один из них не совпадает с оптимальным набором $\{f_1, f_2\}$.

Этот контрпример не проходит, если после каждого добавления пробовать удалить наименее полезный признак, без которого модель становится только лучше.

[1] Здесь и далее трудоёмкость алгоритмов анализируется без учёта времени вычисления внешнего критерия $Q(\mathscr{G})$, которое может существенно зависеть от числа признаков и длины выборки.

Алгоритм 1.3. Add-Del: выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ поочерёдным добавлением и удалением

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметр d;

$\mathscr{G}_0 := \varnothing$; $Q(\mathscr{G}_0) := +\infty$; — инициализация, как в алгоритме Add;
$t := 0$; — счётчик числа итераций;
повторять
  начать последовательные добавления Add:
  $t_0 := t$;
  пока $|\mathscr{G}_t| < n$
    $t := t + 1$; — началась следующая итерация;
    $f^* := \arg\min_{f \in \mathscr{F} \setminus \mathscr{G}_{t-1}} Q\big(\mathscr{G}_{t-1} \cup \{f\}\big)$; — найти добавляемый признак;
    $\mathscr{G}_t := \mathscr{G}_{t-1} \cup \{f^*\}$; — добавить признак;
    $t^* := \arg\min_{s:\, t_0 \leqslant s \leqslant t} Q(\mathscr{G}_s)$; — итерация, на которой был получен лучший набор;
    если $t - t^* \geqslant d$ **то прервать цикл**;
  начать последовательные удаления Del:
  $t_0 := t$;
  пока $|\mathscr{G}_t| > 0$
    $t := t + 1$; — началась следующая итерация;
    $f^* := \arg\min_{f \in \mathscr{G}_{t-1}} Q\big(\mathscr{G}_{t-1} \setminus \{f\}\big)$; — найти удаляемый признак;
    $\mathscr{G}_t := \mathscr{G}_{t-1} \setminus \{f^*\}$; — удалить признак;
    $t^* := \arg\min_{s:\, t_0 \leqslant s \leqslant t} Q(\mathscr{G}_s)$; — итерация, на которой был получен лучший набор;
    если $t - t^* \geqslant d$ **то прервать цикл**;
пока значения критерия $Q(\mathscr{G}_{t^*})$ уменьшаются;

В данном примере на третьем шаге был бы удалён признак f_3, после чего остался бы оптимальный набор $\{f_1, f_2\}$. Эта идея обобщается в следующем алгоритме.

1.2.3 Поочерёдное добавление и удаление признаков

Алгоритм добавления-удаления признаков Add-Del, как следует из названия, совмещает в себе две жадные стратегии, действующие противонаправленно, и в результате получается не совсем жадный алгоритм.

Идея состоит в том, чтобы позволить алгоритму Add включить некоторое количество d избыточных признаков, в надежде на то, что полученный набор $\mathscr{G}$ будет содержать в себе оптимальный набор как подмножество. После этого запускается алгоритм Del, который пытается удалить избыточные признаки. Ему также позволяется удалить чуть больше, чем нужно, и после этого снова запускается Add. Процессы последовательных добавлений и удалений чередуются до тех пор, пока значение критерия $Q(\mathscr{G}_{t^*})$ в точках минимума функционала не перестанет уменьшаться, или пока состав признаков в оптимальном наборе $\mathscr{G}_{t^*}$ не стабилизируется. Отметим, что индекс t у набора $\mathscr{G}_t$ теперь обозначает не сложность модели, а номер итерации.

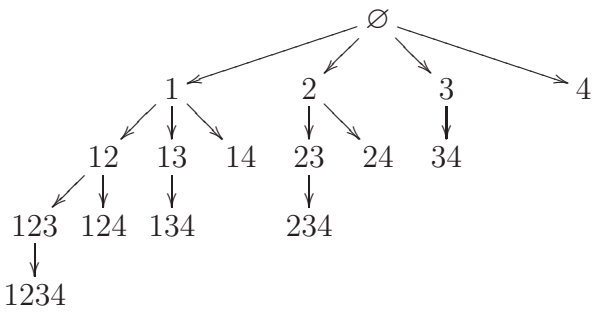

Рис. 1. Дерево полного перебора наборов признаков при $n = 4$. Для краткости наборы обозначены номерами составляющих их признаков.

Достоинства и недостатки. Алгоритм Add-Del работает дольше, чем Add и Del в отдельности, и также не гарантирует оптимальность. Однако на практике ему гораздо чаще удаётся найти лучшее решение, чем методам Add или Del. Число последовательных циклов удаления и добавления, как правило, не очень велико, и дополнительные затраты ресурсов окупаются заметным улучшением качества решения. К недостаткам можно отнести относительную сложность реализации. Многие методы обучения допускают эффективное добавление одного признака, однако чтобы тот же метод допускал также и эффективное удаление, приходится проявить изрядную изобретательность. В случае линейной регрессии такие методы были разработаны в 70-е годы.

Шаговая регрессия. В многомерной линейной регрессии алгоритм Add-Del называют *шаговой регрессией* (stepwise regression). Фёрнивалю удалось разработать эффективную схему вычислений, при которой вычисление значения функционала при добавлении и удалении признака требует лишь нескольких операций, то есть имеет трудоёмкость $O(1)$. Правда, это касается только внутреннего критерия.

1.2.4 Поиск в глубину: метод ветвей и границ

Один из способов полного перебора 2^n наборов заключается в том, чтобы обойти дерево возможных наборов признаков, которое определяется следующим образом. Вершины дерева соответствуют наборам признаков. Корневая вершина соответствует пустому набору. Каждый дочерний набор образуется путём присоединения некоторого признака к родительскому набору. Чтобы избежать появления в дереве одинаковых наборов, отличающихся только порядком признаков, к дочерним наборам присоединяются только те признаки, номера которых превышают максимальный номер признака в родительском наборе. В результате на j-м уровне дерева образуются ровно C_n^j наборов, состоящих из j признаков. Пример дерева показан на Рис. 1.

Существуют две стратегии полного обхода дерева: *поиск в глубину* (depth-first search, DFS) и *поиск в ширину* (breadth-first search, BFS). Обе позволяют вводить различные эвристики для сокращения перебора. Сейчас остановимся на первой, а в разделе 1.2.5 рассмотрим вторую.

Для обхода дерева нет никакой необходимости строить его в явном виде. Идея состоит в том, чтобы использовать функцию `Нарастить`($\mathscr{G}$), которая по очереди присоединяет к $\mathscr{G}$ по одному признаку, и для каждого из полученных наборов $\mathscr{G} \cup \{f\}$

Алгоритм 1.4. Выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ сокращённым поиском в глубину

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметры d и $\varkappa$;

1: Инициализация массива лучших значений критерия:
$Q_j^{\min} := +\infty$ для всех $j = 1, \ldots, n$;
2: Упорядочить признаки по убыванию информативности;
3: `Нарастить`$(\varnothing)$;
4: **вернуть** $\mathscr{G}$, для которого $Q(\mathscr{G}) = \min\limits_{j=1,\ldots,n} Q_j^{\min}$;

5: **ПРОЦЕДУРА** `Нарастить` $(\mathscr{G})$;
6: **если** найдётся $j \leqslant |\mathscr{G}| - d$ такое, что $Q(\mathscr{G}) \geqslant \varkappa Q_j^{\min}$, **то**
7: **выход**;
8: $Q_{|\mathscr{G}|}^{\min} := \min\bigl\{Q_{|\mathscr{G}|}^{\min}, Q(\mathscr{G})\bigr\}$;
9: **для всех** $f_s \in \mathscr{F}$ таких, что $s > \max\{t \mid f_t \in \mathscr{G}\}$
`Нарастить`$(\mathscr{G} \cup \{f_s\})$;

сначала вычисляет значение критерия $Q(\mathscr{G} \cup \{f\})$, затем вызывает себя же рекурсивно: `Нарастить`$(\mathscr{G} \cup \{f\})$. Детали реализации показаны в Алгоритме 1.4.

Этот алгоритм позволяет легко перейти от полного перебора к сокращённому, заимствуя стандартные эвристики у метода ветвей и границ.

Первая эвристика состоит в том, чтобы оценивать перспективность ветви дерева и отказываться от её наращивания, если уже имеется лучшая ветвь. Набор $\mathscr{G}$ не наращивается, если значение критерия $Q(\mathscr{G})$ оказывается хуже, чем на самом лучшем из уже проверенных наборов меньшей мощности j. Формально это можно записать как одновременное выполнение двух условий:

$$Q(\mathscr{G}) \geqslant \varkappa Q_j^{\min};$$
$$|\mathscr{G}| \geqslant j + d;$$

где $Q_j^{\min}$ — значение критерия на самом лучшем наборе мощности j, $d \geqslant 0$ — целочисленный параметр, $\varkappa \geqslant 1$ — вещественный параметр. Чем меньше значения параметров d и $\varkappa$, тем сильнее сокращается перебор.

Вторая эвристика направлена на то, чтобы как можно раньше построить наиболее удачную ветвь дерева. Тогда значения $Q_j^{\min}$ будут близки к нижней огибающей $Q(j)$, и первая эвристика будет отсекать большинство ветвей уже в самом начале. Для этого признаки изначально ранжируются в порядке убывания их индивидуальной информативности. В качестве эвристической меры информативности признака f можно взять либо значение критерия на однопризнаковом наборе $Q(\{f\})$, либо корреляцию с целевым признаком y^*. Либо можно поступить так: сгенерировать некоторое количество случайных наборов $\mathscr{G}$ и оценить информативность каждого признака f как среднюю величину $Q(\mathscr{G})$ по всем $\mathscr{G}$, содержащим f.

Перечисленные приёмы в совокупности позволяют «оттянуть» наступление комбинаторного взрыва и находить оптимальные наборы при количестве признаков порядка 50–70.

Алгоритм 1.5. Выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ сокращённым поиском в ширину (итерационным многорядным алгоритмом МГУА)

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметры d, B;

первый ряд состоит из всех наборов длины 1:
$R_1 := \big\{\{f_1\}, \ldots, \{f_n\}\big\}$;
для всех $j = 1, \ldots, n$, где j — сложность наборов:
  отсортировать ряд $R_j = \big\{\mathscr{G}_j^1, \ldots, \mathscr{G}_j^{B_j}\big\}$ по возрастанию критерия:
  $Q\big(\mathscr{G}_j^1\big) \leqslant \ldots \leqslant Q\big(\mathscr{G}_j^{B_j}\big)$;
  если $B_j > B$ **то**
    оставить только B лучших наборов ряда:
    $R_j := \big\{\mathscr{G}_j^1, \ldots, \mathscr{G}_j^B\big\}$;
  запомнить, какую сложность имел самый лучший набор:
  $j^* := \arg\min\limits_{s:\, s \leqslant j} Q(\mathscr{G}_s^1)$;
  если $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}^1$;
  породить следующий ряд:
  $R_{j+1} := \big\{\mathscr{G} \cup \{f\} \;\big|\; \mathscr{G} \in R_j,\ f \in \mathscr{F} \setminus \mathscr{G}\big\}$;

1.2.5 Поиск в ширину: многорядный итерационный алгоритм МГУА

Алгоритм Add на каждой итерации выбирает только один признак, максимально уменьшающий функционал, и добавляет его в набор $\mathscr{G}$. Основной недостаток Add в том, что жадный выбор признака, как правило, оказывается неоптимальным после добавления следующих признаков.

Усовершенствуем этот алгоритм. На каждой j-й итерации будем строить не один набор, а множество из B_j наборов, называемое *j-м рядом*:

$$R_j = \{\mathscr{G}_j^1, \ldots, \mathscr{G}_j^{B_j}\}, \quad \mathscr{G}_j^b \subseteq \mathscr{F}, \quad |\mathscr{G}_j^b| = j, \quad b = 1, \ldots, B_j.$$

Для перехода от текущего ряда R_j к следующему R_{j+1} от каждого набора $\mathscr{G} \in R_j$ порождается $n - j$ новых наборов путём присоединения одного из признаков, не принадлежащих набору $\mathscr{G}$. Из порождённых $B_j(n-j)$ наборов в следующий ряд отбирается не более B наборов, лучших по внешнему критерию. Таким образом, на каждой итерации сложность всех моделей увеличивается на единицу.

Это и есть *многорядный итерационный алгоритм МГУА*. По сути, он представляет собой *сокращённый поиск в ширину* (breadth-first search, BFS) и воплощает *принцип неокончательных решений Габора*: принимая решения, следует оставлять максимальную свободу выбора для принятия последующих решений.

Число B является параметром алгоритма и называется *шириной поиска*. В частном случае при $B = 1$ снова получаем алгоритм Add. В некоторых случаях ряд может содержать и менее B наборов, например, на первой итерации, если $n < B$.

Трудоёмкость Алгоритма 1.5 составляет $O(Bn^2)$, точнее $O(Bn(j^* + d))$, что в B раз больше, чем у Add, но существенно меньше, чем при полном переборе.

Проблема дубликатов. Иногда в ходе итераций образуются совпадающие наборы, состоящие из одних и тех же признаков, которые добавлялись в разной последовательности. Проверка идентичности наборов может оказаться долгой операцией,

если делать её для всех пар наборов. Эффективное решение проблемы дубликатов состоит в том, чтобы после сортировки наборов на шаге 3 проверить на совпадение только пары соседних наборов, у которых совпадают значения и внутреннего, и внешнего критерия. В случае совпадения один из дублирующих наборов удаляется. Отметим, что это делается перед шагом 5, поэтому удаление дубликатов, как правило, не уменьшает число наборов, переходящих в следующий ряд.

Адаптивный отбор признаков. Для повышения эффективности алгоритма можно оценивать *информативность* отдельных признаков, и на шаге 8 добавлять к наборам j-го ряда только признаки с наибольшей информативностью. В качестве эвристической меры информативности $I_j(f)$ признака $f \in \mathscr{F}$ рекомендуется взять число вхождений данного признака в лучшие наборы j-го ряда:

$$I_j(f) = \sum_{b=1}^{B_j} [f \in \mathscr{G}_j^b].$$

1.2.6 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.

В генетических алгоритмах принята следующая терминология.

Наборы признаков $\mathscr{G} \subseteq \mathscr{F}$ называются *индивидами*. Каждый индивид взаимно однозначно кодируется бинарным вектором $\beta = (\beta_j)_{j=1}^n$, называемым *хромосомой*. В задаче отбора признаков естественно закодировать вхождение j-го признака в набор как единичку в j-м разряде хромосомы: $\beta_j = [f_j \in \mathscr{G}]$.

Над хромосомами, а значит, и над соответствующими индивидами, определены две операции — скрещивание и мутация.

Бинарная операция *скрещивания* (crossover) $\beta = \beta' \times \beta''$:

$$\beta_j = \begin{cases} \beta'_j, & \text{с вероятностью } 1/2; \\ \beta''_j, & \text{с вероятностью } 1/2; \end{cases}$$

Унарная операция *мутации* (mutation) $\beta = {\sim}\beta'$ зависит от параметра p_m, который называется *вероятностью мутации*:

$$\beta_j = \begin{cases} 1 - \beta'_j, & \text{с вероятностью } p_m; \\ \beta'_j, & \text{с вероятностью } 1 - p_m; \end{cases}$$

Генетический Алгоритм 1.6 отличается от Алгоритма МГУА 1.5, главным образом, правилом порождения следующей популяции $R_t := \{\mathscr{G}_t^1, \ldots, \mathscr{G}_t^{B_t}\}$ на шаге 8.

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;

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

Для этого выполняется некоторое число $B_t \ll B$ случайных скрещиваний. Ещё, в отличие от МГУА, популяции могут содержать наборы разной длины. Остальные различия чисто терминологические: в МГУА индивиды называются моделями, популяции — рядами, поколения — итерациями.

Эвристики для управления процессом эволюции. Генетические алгоритмы отличаются огромным разнообразием вариантов реализации. Перечислим лишь некоторые эвристики.

- Усовершенствование операции скрещивания. Увеличивается вероятность перехода признаков от более успешного родителя к потомку.
- Усовершенствование операции мутации. В процессе эволюции накапливаются оценки информативности признаков, например, это может быть число вхождений признака в лучшие наборы. Чем более информативен признак, тем выше вероятность его включения в набор во время мутации.
- Применение совокупности критериев качества. В следующее поколение отбираются индивиды, наилучшие с точки зрения сразу нескольких критериев.
- Скрещивание применяется только к лучшим индивидам (принцип элитаризма).
- Лучшие индивиды в неизменном виде переходят в следующее поколение (эта эвристика уже включена в Алгоритм 1.6 на шаге 8).
- При наступлении стагнации увеличивается вероятность мутаций.

- Параллельно выращивается несколько изолированных популяций (островная модель эволюции). Время от времени могут разрешаться скрещивания между индивидами из разных изолированных популяций.

Достоинства и недостатки. Достоинство генетического алгоритма — в его очевидности и богатых возможностях для введения различных эвристик. Экспериментирование с генетическими алгоритмами — очень увлекательный процесс, своеобразная «компьютерная игра для интеллектуалов». Этим в значительной мере и объясняется огромная популярность эволюционных алгоритмов.

Недостатком является относительно медленная сходимость. Несмотря на тысячи успешных практических применений, сходимость генетического алгоритма до сих пор остаётся открытой теоретической проблемой. Кроме того, хороший генетический алгоритм наряду с параметрами размера популяции B, максимального числа поколений T и вероятности мутации p_m имеет ещё с десяток-другой параметров, подбор которых является искусством и зависит от особенностей задачи.

1.2.7 Случайный поиск с адаптацией

Если упростить генетический алгоритм, отказавшись от скрещивания, то получится алгоритм *случайного поиска* (stochastic search). Это очень простой алгоритм: имея в t-м поколении популяцию из B наборов, слегка модифицируем каждый из них T раз случайным образом. Из полученных BT наборов отберём B лучших по заданному внешнему критерию Q, и сформируем из них $(t+1)$-е поколение. При $B = 1$ поиск лучшей модели ведётся локально в окрестности текущей модели; в этом случае алгоритм называется *локальным случайным поиском* (stochastic local search, SLS).

Недостаток случайного поиска — медленная сходимость. *Случайный поиск с адаптацией, СПА* [4, 5] направлен на ускорение сходимости. Идея адаптации состоит в том, чтобы, генерируя наборы признаков случайным образом, увеличивать вероятность включения в них тех признаков, которые чаще входят в наилучшие наборы. И, наоборот, признаки, входящие в наихудшие наборы, наказываются уменьшением вероятности их появления.

В Алгоритме 1.7 при каждом значении сложности j проводится T итераций. На каждой итерации используется своё распределение вероятностей признаков $p_1, \ldots, p_n$ согласно которому генерируется r наборов по j признаков в каждом. Все признаки, вошедшие в наихудший набор, наказываются уменьшением их вероятности на h. Все признаки, вошедшие в наилучший набор, поощряются увеличением вероятности на одну и ту же величину, так чтобы снова выполнялось условие нормировки $p_1 + \cdots + p_n = 1$. Числа T, r, и h являются параметрами алгоритма.

Трудоёмкость алгоритма составляет $O(Tr(j^* + d))$ операций.

Возможны различные варианты реализации этого алгоритма.

1. Если есть основания полагать, что информативность признаков зависит от длины наборов, то инициализацию вероятностей $p_s = 1/n$ имеет смысл внести в начало внутреннего цикла по t. При этом может возрасти число итераций T, требуемое для стабилизации распределения $\{p_s\}_{s=1}^n$.

2. Внешний цикл по j можно убрать, и вместо этого генерировать на шаге 4 наборы разной длины. При этом придётся увеличить r. С другой стороны, появится

Алгоритм 1.7. Выбор набора признаков $\mathscr{G} \subseteq \mathscr{F}$ случайным поиском с адаптацией

Вход: множество $\mathscr{F}$, выборка X^L, критерий Q, параметр d;
j_0 — минимальное число признаков в наборе;
T — число итераций;
r — число генерируемых наборов на каждой итерации;
h — скорость адаптации;

установить равные вероятности включения признаков:
$p_1 = \cdots = p_n := 1/n$;
для всех $j = j_0, \ldots, n$, где j — сложность наборов:
  для всех $t = 1, \ldots, T$, где t — номер итерации:
    сгенерировать r наборов признаков согласно распределению $\{p_1, \ldots, p_n\}$:
    $R_{jt} := \{\mathscr{G}^1_{jt}, \ldots, \mathscr{G}^r_{jt}\}, \quad |\mathscr{G}^1_{jt}| = \cdots = |\mathscr{G}^r_{jt}| = j$;
    $\mathscr{G}^{\min}_{jt} := \arg\min\limits_{\mathscr{G} \in R_{jt}} Q(\mathscr{G})$; — лучший из r наборов;
    $\mathscr{G}^{\max}_{jt} := \arg\max\limits_{\mathscr{G} \in R_{jt}} Q(\mathscr{G})$; — худший из r наборов;
    $H := 0$; — суммарное наказание худших признаков;
    все признаки $f_s \in \mathscr{G}^{\max}_{jt}$ наказать уменьшением вероятности p_s:
    $\Delta p_s := \min\{p_s, h\}; \quad p_s := p_s - \Delta p_s; \quad H := H + \Delta p_s$;
    все признаки $f_s \in \mathscr{G}^{\min}_{jt}$ поощрить увеличением вероятности p_s:
    $p_s := p_s + H/j$;
  найти лучший набор сложности j:
  $\mathscr{G}_j := \arg\min\limits_{t=1,\ldots,T} Q(\mathscr{G}^{\min}_{jt})$;
  запомнить, какую сложность имел самый лучший набор:
  $j^* := \arg\min\limits_{s:\, s \leqslant j} Q(\mathscr{G}_s)$;
  если $j - j^* \geqslant d$ **то вернуть** $\mathscr{G}_{j^*}$;

возможность уже на ранних стадиях грубо оценить нижнюю огибающую $Q\big(|\mathscr{G}|\big)$, которая будет постепенно уточняться. Чем ближе значение сложности $j = |\mathscr{G}|$ к точке минимума огибающей, тем выше должна становиться вероятность генерации наборов данной сложности j. Реализация этой идеи не показана в Алгоритме 1.7, чтобы не загромождать его техническими подробностями.

Рекомендации по выбору параметров r, T, h приводятся здесь согласно [5].

Число наборов r, генерируемых при каждом распределении вероятностей признаков, должно быть по возможности минимальным, но одновременно достаточным для того, чтобы лучший набор $\mathscr{G}^{\min}_{jt}$ был близок к оптимальному, а худший набор $\mathscr{G}^{\max}_{jt}$ — к самому неинформативному набору признаков. Будем исходить из требования, чтобы интервал $\big[Q(\mathscr{G}^{\min}_{jt}), Q(\mathscr{G}^{\max}_{jt})\big]$ покрывал почти весь диапазон значений критерия. Для произвольного $\mathscr{G}$, взятого из того же распределения, вероятность попадания $Q(\mathscr{G})$ вне этого интервала есть $\beta = \frac{2}{r+1}$. Следовательно, при $r = \frac{2}{\beta} - 1$ значение $Q(\mathscr{G})$ попадает в указанный интервал с вероятностью $1 - \beta$. При уровне значимости $\beta = 0.05$ и 0.10 получаем оценки, соответственно, $r = 39$ и 19.

Число итераций T, достаточное для сходимости процесса, рекомендуется брать порядка нескольких десятков, обычно от 10 до 50. Либо можно прекращать итерации

по эвристическому критерию останова «лучший набор $\mathscr{G}_{j^*}$ не изменялся за последние D итераций», где D — новый параметр алгоритма.

Параметр h, задающий степень адаптации, выбирается так, чтобы вероятность включения признака не могла обнулиться. Пусть $p_{\min}$ — достаточно малая вероятность, скажем, $p_{\min} = \frac{1}{10n}$. Тогда можно положить $h = \frac{1-p_{\min}}{Rn}$. При $h = 0$ алгоритм СПА совпадает с неадаптивным случайным поиском.

Достоинства и недостатки. Достоинством алгоритма является простота реализации и существенно меньшее, по сравнению с генетикой, число параметров. Для выбора параметров СПА имеются довольно чёткие рекомендации. Как показывает практика, СПА сходится гораздо быстрее, чем случайный поиск без адаптации, и во многих экспериментах находит решения лучшего качества, чем шаговый алгоритм Add-Del [5].

Чтобы СПА действительно приводил к наилучшему набору, должна выполняться следующая гипотеза: *признаки, входящие в наилучший набор, часто встречаются в подмножествах с близким значением критерия.* Как проверять эту гипотезу на практике — не ясно; гораздо легче попробовать применить сам алгоритм, сравнив его с другими методами отбора признаков, и выбрать лучший. Сходимость СПА, как и генетического алгоритма, в общем случае не доказана. Некоторые соображения о сходимости можно найти в [6]. На практике подходящие значения параметров всё-таки приходится подбирать экспериментально для каждой конкретной задачи.

1.2.8 Кластеризация признаков

Ещё одна идея отбора признаков восходит к *методу корреляционных плеяд* Терентьева [7] и состоит в том, чтобы найти и исключить схожие признаки с помощью кластеризации. Методы кластеризации, рассмотренные в **??**, позволяют разбить выборку объектов на кластеры, состоящие из схожих объектов, и выделить в каждой группе по одному наиболее типичному представителю. То же самое можно проделать и с признаками, если определить функцию расстояния между признаками, например, через коэффициент корреляции. Метод корреляционных плеяд известен довольно давно и широко применялся на практике задолго до того, как были придуманы более совершенные методы отбора признаков. Было накоплено множество как удачных, так и неудачных примеров его применения. Основные его недостатки заключаются в следующем.

Во-первых, могут существовать кластеры, целиком состоящие из неинформативных признаков. Взяв по одному типичному представителю от каждого кластера, всё равно придётся решать задачу отбора признаков, хотя объём перебора при этом сильно сократится.

Во-вторых, информации о попарном сходстве между признаками в общем случае не достаточно для выделения оптимального набора признаков. В частности, кластеризация не решает проблему мультиколлинеарности, поскольку набор попарно некоррелированных признаков вполне может оказаться линейно зависимым.

Предварительную кластеризацию признаков имеет смысл применять для сокращения перебора в других методах отбора признаков. Например, чтобы запретить слишком похожим признакам входить в один и тот же набор.

Метрики на признаках. Для применения кластеризационных методов отбора признаков необходимо ввести метрику на множестве признаков $\mathscr{F}$. Пусть $f(x)$, $g(x)$ — два произвольных признака из $\mathscr{F}$. Векторы значений признаков на элементах выборки X^ℓ обозначим через $f = (f_1, \ldots, f_\ell) = \big(f(x_1), \ldots, f(x_\ell)\big)$ и $g = (g_1, \ldots, g_\ell) = \big(g(x_1), \ldots, f(g_\ell)\big)$. Рассмотрим несколько вариантов определения функции расстояния $\rho(f, g)$.

Для количественных признаков чаще всего применяется метрика на основе коэффициента линейной корреляции $r(f, g)$:

$$\rho(f, g) = 1 - |r(f, g)|, \quad r(f, g) = \sum_{i=1}^{\ell} f'_i g'_i,$$

где f' и g' — нормированные и центрированные признаки f и g. Это расстояние равно нулю тогда и только тогда, когда признаки связаны линейной зависимостью.

Для порядковых признаков более естественной является метрика Кенделла-Кемени. Она определяется как доля пар объектов x_i, x_j с различными порядковыми отношениями между значениями признаков f и g:

$$\rho(f, g) = \frac{1}{2C_\ell^2} \sum_{i=1}^{\ell-1} \sum_{j=i}^{\ell} \big|\operatorname{sign}(f_i - f_j) - \operatorname{sign}(g_i - g_j)\big|.$$

Это расстояние равно нулю тогда и только тогда, когда признаки связаны монотонной зависимостью, то есть существует монотонная функция M такая, что $f_j = M(g_j)$ для всех $j = 1, \ldots, \ell$.

Для номинальных признаков с одинаковыми множествами допустимых значений $D_f = D_g$ можно использовать метрику Хэмминга, которая обращается в нуль тогда и только тогда, когда векторы f и g совпадают:

$$\rho(f, g) = \sum_{i=1}^{\ell} \big[f_i \neq g_i\big].$$

Для номинальных признаков с различными множествами значений приходится искать соответствие $\sigma\colon D_f \to D_g$, при котором хэммингово расстояние минимально (без ограничения общности предполагается, что $|D_f| \geqslant |D_g|$):

$$\rho(f, g) = \min_{\sigma} \sum_{i=1}^{\ell} \big[\sigma(f_i) \neq g_i\big].$$

Если необходимо найти метрику между разнотипными признаками, измеренными в разных шкалах, то они сначала приводятся к одной общей шкале [2].

1.2.9 Отбор признаков методами математического программирования

Методы математического программирования используются для отбора признаков, главным образом, в линейных моделях регрессии и классификации. Они отличаются от предыдущих методов тем, что в них нет явного перебора признаков. При более глубоком рассмотрении отличие оказывается поверхностным — перебор

осуществляется внутри стандартных процедур математического программирования при поиске активных ограничений, удовлетворяющих условиям Куна-Таккера.

Для простоты ограничимся линейной многомерной регрессией, хотя основная идея переносится и на задачи классификации. Пусть алгоритм имеет вид

$$a(x,\alpha) = \sum_{j=1}^{n} \alpha_j f_j(x),$$

где $\alpha = (\alpha_1, \ldots, \alpha_n)$ — вектор параметров, настраиваемый по внутреннему критерию

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} \big(a(x_i,\alpha) - y_i\big)^2 \to \min_{\alpha\in\mathbb{R}^n}.$$

Метод лассо основан на применении специальной разновидности внешнего критерия регуляризации, см. 1.1.5. Чтобы коэффициенты α_j не принимали слишком большие по модулю значения, квадратичный функционал $Q(\alpha, X^\ell)$ минимизируется при дополнительном ограничении типа неравенства

$$\sum_{j=1}^{n} |\alpha_j| \leqslant \varkappa.$$

Благодаря тому, что в неравенстве стоят модули, а не квадраты, решение этой задачи приобретает одно замечательное свойство: чем меньше $\varkappa$, тем больше коэффициентов α_j принимают нулевое значение. Образно говоря, параметр $\varkappa$ зажимает вектор коэффициентов, лишая его степеней свободы. Отсюда и название — метод лассо [10, 8]. Более подробно этот метод рассматривается в разделе **??**.

Линейная монотонная регрессия основана на предположении, что «хорошие» признаки $f_j(x)$ входят в модель с положительными коэффициентами α_j. Фактически, это тоже разновидность регуляризации, но она подходит не для всех задач. Должно быть априори известно, что «чем больше значение признака f_j, тем больше отклик y». Квадратичный функционал $Q(\alpha, X^\ell)$ минимизируется при дополнительных ограничениях

$$\alpha_j \geqslant 0; \quad j = 1, \ldots, n.$$

Применение стандартных методов квадратичного программирования для решения данной задачи автоматически приводит к обнулению некоторых коэффициентов, следовательно, к отбору признаков. Обнуляются те α_j, для которых ограничение становится активным. Более подробно этот метод разбирается в разделе **??**.

§1.3 Синтез информативных признаков

Синтез признаков (features extraction), также как и отбор, решает задачу сокращения размерности пространства. При отборе часть признаков полностью игнорируется. При синтезе все n исходных признаков участвуют в построении меньшего числа m новых признаков:

$$g_j(x) = g_j\big(f_1(x), \ldots, f_n(x)\big), \quad j = 1, \ldots, m.$$

Синтез имеет два достоинства перед отбором. Во-первых, отсутствует комбинаторный перебор вариантов. Во-вторых, вся исходная информация учитывается в полном объёме. Впрочем, второе достоинство становится недостатком в тех задачах, где присутствуют заведомо неинформативные шумовые признаки.

Недостатком синтеза является то, что новые признаки могут оказаться неинтерпретируемыми.

Общий вывод таков: использовать отбор или синтез — определяется особенностями задачи. В некоторых случаях приходится пользоваться и тем, и другим.

Существует несколько подходов к синтезу признаков.

Первый подход — использование априорной информации. Во многих задачах эксперты могут указать, например, на то, что более информативным показателем является не сам признак, а его отношение (сумма, разность) с другим признаком. В общем случае может быть известно некоторое количество «полезных» функциональных преобразований над исходными признаками.

Второй подход — использование «оптимальных» в некотором смысле линейных преобразований признаков. Это приводит к таким методам, как анализ главных компонент и анализ независимых компонент.

Третий подход — использование произвольных нелинейных преобразований признаков. Частным случаем являются *обобщённые линейные модели* (generalized liner models, GLM) и метод *возвратной настройки* (backfitting), рассмотренные в разделе **??**. В более общем случае строятся произвольные суперпозиции из произвольных нелинейных функций с параметрами. Собственно, это уже не синтез признаков, а подбор структуры модели. Некоторые подходы к решению этой сложной задачи будут рассмотрены в разделе §1.5.

1.3.1 Анализ главных компонент

В *методе главных компонент* (principal component analysis, PCA) ставится задача найти минимальное число новых признаков, по которым исходные признаки можно было бы восстановить линейным преобразованием, возможно, с незначительными погрешностями. PCA относится к методам *обучения без учителя* (unsupervised learning), поскольку преобразование строится по матрице «объекты–признаки» F, без учёта целевого вектора y. Такой подход успешно решает проблему мультиколлинеарности (взаимной зависимости признаков), но не позволяет избавиться от шумовых признаков, не связанных с целевой функцией. В результате шумовые признаки «растворяются» в новых признаках.

Метод главных компонент подробно рассматривается в разделе **??**.

1.3.2 Анализ независимых компонент

§1.4 Выбор модели

1.4.1 Выбор метода обучения

Задача *выбора модели* (model selection) состоит в следующем. Имеется T различных методов обучения $\mu_1, \ldots, \mu_T$. Все T методов применяются к обучающей выборке X^ℓ, в результате получается T алгоритмов $a_t = \mu_t(X^\ell)$. Возникает вопрос: какой из алгоритмов выбрать?

В типичных случаях эта задача возникает, когда априорные предпочтения для использования какой-то определённой модели алгоритмов отсутствуют, и хочется выбрать ту модель, которая обладает лучшей обобщающей способностью. Другой случай — имеется фиксированная модель и метод её настройки μ, но этот метод зависит от некоторых параметров, которые не могут быть оптимизированы по обучающей выборке и должны быть назначены из априорных соображений. Например, в полиномиальной регрессии таким параметром является степень полинома. Похожая ситуация наблюдается и в задачах отбора признаков, когда метод настройки μ фиксирован, но заранее неизвестно, какое взять подмножество признаков $\mathscr{G} \subseteq \mathscr{F}$.

К задачам выбора модели в полной мере применима методология внутренних и внешних критериев. Выбирать лучшую модель по внутреннему критерию некорректно по тем же соображениям, которые ранее излагались в 1.1.1. Поэтому выбор модели производится по внешнему критерию. Наиболее широко распространён *выбор модели по скользящему контролю* (cross-validated model selection):

$$\mathrm{CV}(t, X^L) = \frac{1}{N} \sum_{n=1}^{N} Q\left(\mu_t(X_n^\ell), X_n^k\right), \quad t = 1, \ldots, T;$$
$$t^* = \arg\min_{t=1,\ldots,T} \mathrm{CV}(t, X^L);$$

где X^L — полная выборка, которая делится N различными способами на обучающую часть X_n^ℓ и контрольную часть X_n^k.

1.4.2 Переобучение при выборе метода

§1.5 Оптимизация структуры модели

Общая задача информационного моделирования заключается в том, чтобы по имеющейся обучающей выборке определить не только параметры модели, но и саму модель. Под *моделью* или *структурой* алгоритма $a(x)$ будем понимать описание отображения $a(x)$ как суперпозиции элементарных функций F_J, каждая из которых принимает в качестве аргументов либо другие элементарные функции, либо признаки, либо константы, и может зависеть от вектора параметров β_J:

$$a(x) = F_0\big(a_1(x), \ldots, a_{n_0}(x); \beta_0\big);$$
$$a_J(x) = F_J\big(a_{J,1}(x), \ldots, a_{J,n_J}(x); \beta_J\big);$$

где через J обозначается список целочисленных индексов, однозначно определяющий положение функции F_J в суперпозиции (пустой список обозначается нулём). Способ формирования этих индексов очевиден из Рис. 2. Число элементов списка $|J|$ будем называть *уровнем* функции F_J в суперпозиции. Множество всех списочных индексов J, встречающихся в описании суперпозиции, включая нулевой, обозначим через $\mathcal{J}$. Совокупность $\beta = (\beta_J)_{J \in \mathcal{J}}$ образует вектор параметров алгоритма $a(x)$.

1.5.1 Снова МГУА

1.5.2 Снова генетический алгоритм

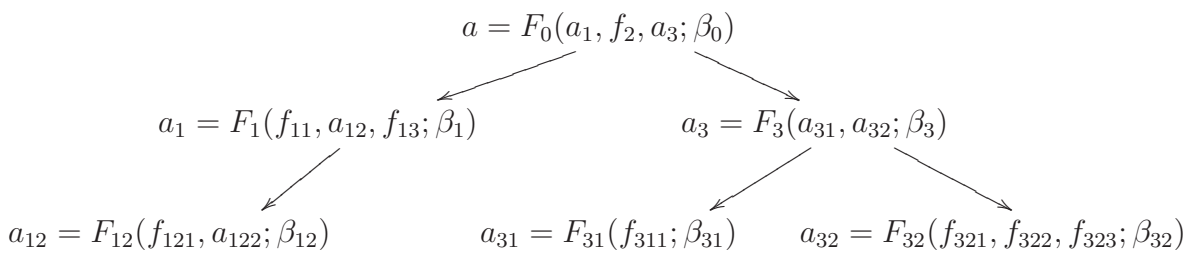

Рис. 2. Пример структуры алгоритма как суперпозиции элементарных функций.

Резюме

1. *Задача выбора модели* состоит в том, чтобы выбрать адекватную структуру восстанавливаемой зависимости. Чем сложнее структура, то есть чем больше параметров вводится в модель, или чем больше признаков задействуется, тем меньшую ошибку можно обеспечить на обучающей выборке. Однако избыточно точная настройка может приводить к увеличению ошибки на новых данных; это явление называется *переобучением*.

2. *Внутренние критерии* применяются для обучения (подгонки) параметров модели по обучающей выборке. Внутренние критерии не следует использовать для выбора модели, так как при этом поощряются чрезмерно сложные модели, склонные к переобучению.

3. *Внешние критерии*. Критерии *скользящего контроля* и *непротиворечивости* оценивают качество моделей по тем данным, которые не использовались для обучения. Критерии *регуляризации* накладывают ограничения на параметры модели и следят за её устойчивостью. Критерии *Вапника-Червоненкиса*, *Акаике* (AIC), *байесовский* (BIC) основаны на теоретических оценках обобщающей способности и также не требуют дополнительных данных. При выборе модели целесообразно использовать сразу несколько внешних критериев.

4. *Задача отбора признаков* заключается в том, чтобы найти набор признаков, оптимальный с точки зрения выбранных внешних критериев. В общем случае эта задача является NP-полной и требует перебора всех 2^n наборов признаков, причём для каждого набора приходится решать задачу обучения параметров модели. Как правило, это становится невозможным уже при n порядка 20. Все методы отбора признаков являются более или менее удачными эвристиками, нацеленными на сокращение перебора.

5. *Жадные методы* добавляют или удаляют по одному признаку, каждый раз выбирая тот признак, при котором достигается наибольшее улучшение внешнего критерия. Жадные стратегии могут «застревать» в локальных оптимумах и часто не находят хорошее решение.

6. *Поиск в глубину* или метод ветвей и границ усовершенствует стратегию жадного добавления, прорабатывая на каждом шаге несколько альтернативных вариантов добавления признака. Различные эвристики позволяют оценивать перспективность вариантов и регулировать количество рассматриваемых альтернатив, добиваясь компромисса между временем поиска и качеством решения.

7. *Поиск в ширину* или многорядный итерационный алгоритм МГУА (метода группового учёта аргументов). Это ещё одна «полу-жадная» стратегия добавления. На каждом шаге рассматривается ряд наборов, к каждому из них добавляется по одному признаку всеми возможными способами, и из этого огромного количества наборов отбираются лучшие по внешнему критерию; таким образом оказывается сформированным следующий ряд, и процесс повторяется.

8. *Генетический алгоритм* очень похож на МГУА, только «ряд» называется «поколением», и каждое последующее поколение порождается операциями скрещивания и мутации наборов, а не операцией добавления одного признака.

9. *Случайный поиск с адаптацией* основан на генерации случайных наборов признаков. Каждый набор оценивается по внешнему критерию, и если он оказывается достаточно хорошим, то все входящие в него признаки поощряются увеличением вероятности их появления в следующих наборах. Соответственно, признаки, образующие худшие наборы, наказываются уменьшением вероятности.

Список литературы

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

[2] *Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С.* Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985.

[3] *Ивахненко А. Г., Юрачковский Ю. П.* Моделирование сложных систем по экспериментальным данным. — М.: Радио и связь, 1987.

[4] *Лбов Г. С.* Выбор эффективной системы зависимых признаков // *Вычислительные системы.* — 1965. — Т. 19. — С. 21–34.

[5] *Лбов Г. С.* Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.

[6] *Меерков С. М.* Свойство замедления в задаче поиска глобального экстремума функций // *Автоматика и телемеханика.* — 1972. — № 12. — С. 129–139.

[7] *Терентьев П. В.* Метод корреляционных плеяд // *Вест. Ленингр. ун-та.* — 1959. — № 9. — С. 137–141.

[8] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. — Springer, 2001.

[9] *Mullin M., Sukthankar R.* Complete cross-validation for nearest neighbor classifiers // Proceedings of International Conference on Machine Learning. — 2000. — Pp. 639–646.
http://citeseer.ist.psu.edu/309025.html.

[10] *Tibshirani R. J.* Regression shrinkage and selection via the lasso // *Journal of the Royal Statistical Society. Series B (Methodological).* — 1996. — Vol. 58, no. 1. — Pp. 267–288.
http://citeseer.ist.psu.edu/tibshirani94regression.html.