ML Snippets

s.pol

Оглавление

Предисловие

В данной книге описаны основные понятия, методы и подходы, широко используемые в современном DS и ML, встречавшиеся автору на собеседованиях и в ежедневной работе. Обычно, свободное владение этими понятиями необходимо для правильного понимания как основных, так и продвинутых методов ML и по умолчанию предполагается от специалиста в области DS.

Материал подается в виде сниппетов - коротких (по возможности) статей. Охвачены такие разделы, как теория вероятностей, классическая и байесовская статистика, некоторые вопросы мат. анализа, общие понятия DS и ML.

Книга представляет собой скорее расширенный глоссарий по основным понятиям ML, чем систематическое изложение какой-либо области. Освещение вопросов ни в коем случае не претендует на полноту и в некоторых случаях на математическую строгость.

Обозначения

DS	- наука о данных
ML	- машинное обучение
RV	- случайная величина
CDF	- функция распределения случайной величины
PDF	- плотность распределения случайной величины
CLT	- центральная предельная теорема
EX	- среднее случайной величины X
DX	- дисперсия случайной величины X
$X \sim Y$	- случайные величины X и Y одинаково распределены

Глава 1

Математика

В этой главе описаны основные математические понятия, необходимые для правильного понимания как основных, так и продвинутых методов ML. Охвачены: теория вероятностей, классическая и байесовская статистика, некоторые вопросы мат. анализа.

1.1 Случайная величина

Случайной величиной (RV) называется числовая функция X, определенная на некотором множестве элементарных исходов Ω (обычно подмножество $\mathbb{R}$ или $\mathbb{R}^n$),

$$X : \Omega \to \mathbb{R}.$$

С прикладной точки зрения на RV часто смотрят как на генераторы случайных чисел с заданным распределением.

Примеры:

- Рост людей, взятых из некоторой группы.
- Цвет фиксированного пикселя изображения, взятого из некоторого множества изображений.
- Некоторый признак из датасета ML задачи.

1.2 Распределение случайной величины

Если RV принимает дискретное множество значений $x_1, x_2, ...$, то она полностью определяется значениями их вероятностей: $p_k = \mathbb{P}(X = x_k)$.

Если множество значений RV не дискретно, то RV может быть описана своей функцией распределения (CDF, Cumulative distribution function): $F(x) = \mathbb{P}(X < x)$.

В большинстве прикладных случаев CDF оказывается дифференцируемой функцией. Производная от CDF называется плотностью распределения случайной величины (PDF, Probability density function): $f(x) = F'(x)$. Таким образом, по определению

$$\mathbb{P}(a < X < b) = \int_a^b f(x)dx.$$

1.3 Выборка

Выборкой объема n из генеральной совокупности X называется последовательность независимых и распределенных как X случайных величин:

$$X_1, X_2, ..., X_n, \quad X_k \sim X$$

На практике под выборкой понимают конкретные реализации величин X_k, то есть последовательность чисел $x_1, x_2, ..., x_n$.

1.4 Закон больших чисел

Закон больших чисел утверждает, что если $X_1, X_2, ..., X_n$ - выборка объема n из генеральной совокупности X, то ее среднее с ростом n становится все менее случайно и в конечном итоге стабилизируется к постоянной величине - среднему значению X:

$$\frac{X_1 + X_2 + ... + X_n}{n} \approx \mathbb{E}X, \quad n \to \infty.$$

1.5 Центральная предельная теорема

Центральная предельная теорема (CLT) является в некотором смысле уточнением закона больших чисел. В упрощенном варианте она утверждает, что если $X_1, X_2, ..., X_n$ - выборка объема n из генеральной совокупности X, то ее распределение ее среднего при больших n очень близко к нормальному,

$$\frac{X_1 + X_2 + ... + X_n}{n} \approx N(\mu, \sigma^2/n), \quad \mu = EX, \sigma^2 = DX, \quad n \to \infty.$$

Заметим, что если совокупность распределена нормально, $X \sim N(\mu, \sigma^2)$, то предыдущая формула обращается в точное равенство при любых n.

1.6 Приближенное вычисление интегралов

Пусть $x_1, x_2, ..., x_n$ - реализация некоторой выборки из генеральной совокупности X, распределенной с некоторой плотностью $f(x)$. Если $Y = g(X)$ - случайная величина, являющаяся функцией X, то ее среднее есть по определению

$$\mathbb{E}Y = \int g(x)f(x)dx.$$

при неизвестной плоности этот интеграл может быть приблизительно вычислен как

$$\mathbb{E}Y \approx \frac{1}{n}\sum_{i=1}^{n} g(x_i).$$

В частности,

$$\mathbb{E}X \approx \frac{1}{n}\sum_{i=1}^{n} x_i.$$

1.7 Статистики

Пусть $X_1, X_2, ..., X_n$ - выборка объема n. Статистикой называется произвольная RV, являющаяся функцией выборки:

$$T = T(X_1, X_2, ..., X_n).$$

Часто статистикой называют конкретное значение $T(x_1, x_2, ..., x_n)$, полученное на данной реализации $x_1, x_2, ..., x_n$ выборки.

Примеры:

- $\bar{X} = (X_1 + X_2 + ... + X_n)/n$ - выборочное среднее.
- $X_{(n)} = \max(X_1, X_2, ..., X_n)$ - максимальное значение в выборке.
- медиана, перцентили.

1.8 Bootstrap

Пусть имеется реализация некоторой выборки $x_1, x_2, ..., x_n$. Если по каким-либо причинам мы не можем больше сэмплировать из генеральной совокупности, то любая статистика выборки также представляет собой константу: $T(x_1, x_2, ..., x_n) = const$.

Бутстрепом называется процесс сэмплирования из имеющейся выборки $x_1, x_2, ..., x_n$ подвыборки длины n с возвращением. Проведя эту операцию k раз, можем, в частности, посчитать k значений интересующей нас статистики, то есть фактически исследовать ее возможные эмпирические значения.

Бутстреп позволяет получать эмпирические распределения интересующих нас величин, получать точечные и интервальные оценки, выравнивать количества несбалансированных выборок при A/B тестировании и многое другое.

При однократном выполнении данной процедуры мы используем в среднем только $1 - 1/e \approx 63\%$ всей выборки, что обычно используется в out-of-bag оценках различных алгоритмах, основанных на бутстрепе.

1.9 Классический и байесовский подход

Основные отличия между классическим (частотным) и байесовским подходом отражены в следующей таблице:

	Частотный подход	Байесовский подход
Идеология	Объективная случайность существует	Любая случайность субъективна
Объект	Неслучайные параметры случайных величин	Распределения случайных величин
Метод	Метод максимального правдоподобия	Теорема Байеса
Результат	Точечные и интервальные оценки неизвестных параметров	Распределение неизвестных параметров
Применимость	Размер выборки велик по сравнению с кол-вом неизвестных параметров	Размер выборки любой

Иными словами, в байесовском подходе любая неизвестная величина считается случайной с некоторым распределением, которое затем может уточняться вогласно новым данным по теореме Байеса.

1.10 Классическая оценка параметров: MLE

Пусть имеется выборка значений $D = \{x_1, x_2, ..., x_n\}$ некоторой величины X, точное распределение которой нам неизвестно. Предполагаем, что плотность распределения X известна с точностью до параметра α (возможно, многомерного):

$$X \sim g(x, \alpha).$$

Величина

$$L(\alpha|D) = g(D|\alpha) = \prod_{i=1}^{n} g(x_i|\alpha)$$

называется ***правдоподобием*** и представляет собой плотность/вероятность получить именно такие данные D из распределения исходной величины X. Естественно пытаться определить неизвестный параметр α как тот, при котором эта вероятность максимальна. Правдоподобие обычно логарифмируют, что не влияет на

оптимальное значение параметра:

$$\alpha_{MLE} = \text{argmax} \ln L(\alpha|D).$$

1.11 Байесовская оценка параметров: MAP

Пусть имеется выборка значений $D = \{x_1, x_2, ..., x_n\}$ некоторой величины X, точное распределение которой нам неизвестно. Предполагаем, что плотность распределения X известна с точностью до параметра α (возможно, многомерного):

$$X \sim g(x, \alpha).$$

Согласно ***байесовской парадигме***, неизвестный параметр α трактуется не как число, а как случайная величина, имеющая некоторое распределение: $\alpha \sim f(\alpha)$. Это исходное распределение параметра может быть уточнено с учетом пришедших данных D по формуле Байеса:

$$f(\alpha|D) = \frac{g(D|\alpha)f(\alpha)}{h(D)}.$$

Здесь f - априорное распределение α, $g(D|*)$ - правдоподобие пришедших данных D, h - evidence - плотность/вероятность получения именно таких данных.

Процедуру уточнения распределения неизвестных параметров можно повторять по мере получения новых данных, используя полученное на предыдущем шаге апостериорное распределение как априорное для следующего.

MAP оценкой параметра α называется значение

$$\alpha_{MAP} = \text{argmax} \, f(\alpha|D).$$

Важно понимать, что байесовский подход дает оценку не плотности распределения исходных данных g напрямую, а оценку плотности распределения некоторого параметра распределения g.

Известными минусами байесовского подхода является необходимость откуда-то брать априорное распределение параметра $f(\alpha)$ и вычисление знаменателя $h(D)$, что является обычно вычислительно сложной задачей. Тем не менее, обычно выбор априорного распределения не оказывает сильного влияния на результат, а знаменатель $h(D)$, будучи не зависимой от α, не влияет на α_{MAP}.

1.12 Сравнение классической и байесовской оценок параметров

Пусть имеется выборка значений $D = \{x_1, x_2, ..., x_n\}$ некоторой величины X, точное распределение которой нам неизвестно. Предполагаем, что плотность распределения X известна с точностью до параметра α (возможно, многомерного):

$$X \sim g(x, \alpha).$$

Согласно ***классическому подходу***, величина α трактуется как фиксированное (хоть и неизвестное) число, которое находится исходя из желания максимизировать вероятность того, что исходные данные D пришли именно из этого распределения.

Согласно ***байесовскому подходу***, величина α трактуется как случайная с некоторым известным априорным распределением. Основной целью в этом случае выступает нахождение апостериорного распределения величины α, уточненного согласно пришедшим данным D по формуле Байеса.

Если интервал изменения неизвестного параметра α конечен, а о самом α мы не имеем никаких априорных знаний, естественно положить априорное распределение α равномерным. В этом случае результаты баейсовской и классической оценок совпадают: $\alpha_{MLE} = \alpha_{MAP}$.

1.13 Классический доверительный интервал

1.14 Байесовский доверительный интервал

1.15 Предсказательный интервал

1.16 Основные дискретные распределения

$https://medium.com/@srowen/common-probability-distributions-347e6b945ce4$

1.17 Основные непрерывные распределения

1.18 Метод Монте-Карло

1.19 Байесовская оптимизация

1.20 Матричные разложения

...может разделить главу на части...

1.21 К-Л дивергенция

Дивергенция Кульбака-Лейблера (относительная энтропия) - мера сходства двух распределений. Для RV P, Q вычисляется как

$$K(p,q) = -\sum_{i=0}^{n} p_i \log_2 \frac{q_i}{p_i}.$$

Определено для распределений $q_i = 0 \Rightarrow p_i = 0$. В байесовской интерпретации это информация, приобретенная при переходе от априорного распределения p к апостериорному распределению q.

К-Л дивергенция всегда неотрицательна и аддитивна в следующем смысле: если P_1, P_2 - независимые RV, Q_1, Q_2 - тоже, то для совместных плотностей распределения P и Q справедливо:

$$K(P, Q) = K(P_1, Q_1) + K(P_2, Q_2).$$

К-Л дивергенция может использоваться как метрика между распределениями, так как при $n \to \infty$ справедливо

$$K(P_n, Q) \to 0 \quad \Rightarrow \quad P_n \to Q$$

в смысле распределений.

1.22 Энтропия

1.23 Кросс-энтропия

1.24 Квантили

1.25 Точечные оценки

1.26 Интервальные оценки

1.27 Проверка гипотез ***

Пусть $x_1, x_2, ..., x_n$ - выборка из генеральной совокупности X, распределение которой заранее неизвестно. Требуется проверить некоторое утверждение о распределении генеральной совокупности X исходя из имеющейся выборки.

Общий подход к решению таких задач состоит в следующем:

1. Формулируется основная гипотеза H_0 о распределении X и некоторая альтернативная гипотеза H_1, которая может являться полным отрицанием H_0 (двусторонняя альтернатива), но не обязательно (односторонние и др. альтернативы).

2. Выбирается некоторая статистика T исходя из условия, что нулевая гипотеза H_0 верна тогда и только тогда, когда распределение T известно: $T \sim F(t|H_0)$ (нулевое распределение статистики).

3. Вычисляется значение t^* статистики T на имеющейся выборке. По известному распределению F можно судить, насколько вероятно получить значения t^*.

4. Выбирается уровень значимости α.

5. Вычисляется достигаемый уровень значимости p_{value} и сравнивается с α. Если $p_{value} > \alpha$, то нулевая гипотеза H_0 не может быть отвергнута, а если $p_{value} \leqslant \alpha$, гипотеза H_0 отвергается в пользу альтернативной H_1.

Пример: TODO

Обычно нулевая гипотеза H_0 означает, что "ничего интересного не происходит а альтернативная, напротив, говорит о том, что "что-то произошло".

1.28 Достигаемый уровень значимости, p_{value} ***

Пусть статистика T приняла на выборке $x_1, x_2, ..., x_n$ значение t^*. Так как нулевое распределение статистики известно, по значению t^* можно судить, насколько оно характерно для данного распределения. Именно, для случая правосторонней альтернативной определим значение

$$p_{value} = \mathbb{P}(T \geqslant t^* | H_0).$$

TODO картинка

Если бы H_0 была справедлива, то значение t^* вероятно оказалось бы около матожидания распределения T и, как следствие, p_{value} было бы велико. В противном случае, если t^* оказалось далеко правее среднего, p_{value} мало, и нулевую гипотезу следует отвергнуть в пользу правосторонней альтернативы H_1.

Число p_{value} называется достигаемым уровнем значимости. Оно сравнивается с заранее заданным уровнем значимости α, и если $p_{value} > \alpha$, то нулевая гипотеза H_0 не может быть отвергнута, а если $p_{value} \leqslant \alpha$, гипотеза H_0 отвергается в пользу альтернативной H_1.

Аналогичные рассуждения справедливы для случаев лево- и двусторонней альтернативы.

1.29 Множественная проверка гипотез

Пусть имеется ряд статистических гипотез, которые следует проверить в совокупности, то есть совокупная нулевая гипотеза состоит в том, что во все исходные нулевые гипотезы верны. Пусть α - уровень значимости для всех исходных гипотез. Опрерировать тем же уровнем значимости для совокупной гипотезы было бы

неверно, так как отвергание совокупной нулевой гипотезы означало бы, что хоть одна из исходных гипотез отвергается. Вероятность этого события

$$1 - (1 - \alpha)^m,$$

экспоненциально близка к 1 при больших m. Таким образом, вероятность совершить ошибку первого рода для множественной проверки гипотез всегда очень близка к 1, если использовать тот же уровень значимости, что и для исходных гипотез.

Пример: Некоторое лекарство исследуется на m возможных побочных эффектов. Чем больше m, тем выше вероятность, что хоть один побочный эффект проявится. Это не означает, что этот побочный эффект характерен для лекарства, он мог проявиться просто случайно.

1.30 Коррекции на множественную проверку гипотез

1.31 Ошибки I и II рода

1.32 Уровень значимости, α

Уровень значимости α - вероятность ошибки первого рода, то есть вероятность отвергнуть верную нулевую гипотезу.

1.33 Мощность статистического критерия

Пусть β - вероятности ошибки второго рода, то есть вероятности принять неверную нулевую гипотезу. Величина $1 - \beta$ называется мощностью статистического критерия. Таким образом, мощность статистического критерия - это вероятность отвергнуть неверную нулевую гипотезу.

1.34 Основные задачи статистики

...из лекций новосиба курсера...

1.35 Параметрические и непараметрические критерии, бутстреп

1.36 Проверка основных гипотез

1.37 Корреляция Пирсона

1.38 Корреляция Спирмена

1.39 Корреляция Метьюса

1.40 Корреляция Крамера

1.41 Z-тест Фишера

1.42 T-тест Стьюдента

1.43 Критерий Пирсона χ^2

1.44 Точный тест Фишера

1.45 Проклятье размерности

Глава 2

Анализ данных

Анализ и предобработка данных - первая задача, успешное решение которой зачастую определяет успех в решении любых задач ML. В этой главе описываются основные подходы....

2.1 Типы данных

2.2 Предобработка данных

2.3 Понижение размерности

Глава 3

Общие вопросы

В этой главе приводятся основные понятия ML и DS.

3.1 Машинное обучение

Машинное обучение (ML) - область искусственного интеллекта, изучающая самообучающиеся модели, то есть решаюшие поставленную задачу не по заранее запрограммированному алгоритму, а предварительно настраивая свое поведение согласно имеющимся данным.

Обычно методы ML содержат свободные параметры, подбор которых наилучшим (в смысле имеющихся данных и задачи) образом и составляет процесс обучения алгоритма. После обучения алгоритм можно использовать на новых данных, которые не были представлены алгоритму на стадии обучения.

3.2 Эмпирический риск

Будем считать, что обучающая выборка (x_1, y_1), (x_2, y_2), ... (x_n, y_n) приходит из некоторого распределения с плотностью $f(x, y)$. Пусть, далее, задано зависящее от параметра семейство алгоритмов $a = a(x, w)$ и некоторая функция потерь $L = L(y, a)$.

В этих условиях общая задача обучения с учителем может быть формализованна как задача минимизации риска, то есть средней ошибки

$$l(w) \equiv \mathbb{E}L = \int L(y, a(x, w)) f(x, y) dxdy,$$

$$w_{opt} = \operatorname{argmin} l(w).$$

Само распределение f обычно не известно, однако, при условии, что обучающая выборка пришла из этого распределения, интеграл можно приближенно заменить

средним и таким образом оценить среднюю ошибку через ***эмпирический риск***

$$r_{emp}(w) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i, w)) \approx l(w).$$

Таким образом, основную задачу обучения можно пытаться решать минимизацией эмпирического риска вместо полного.

3.3 Основные классы задач

3.3.1 Обучение с учителем

3.3.2 Обучение без учителя

3.3.3 Частичное обучение

3.3.4 Обучение с подкреплением

3.4 Обнаружение аномалий

3.5 Контроль качества

...оценка обобщающей способности...

3.6 Недообучение

3.7 Переобучение

3.8 Регуляризация

3.9 Отбор признаков

3.10 Параметры алгоритма

3.11 Подбора метапараметров

3.12 Основные типы алгоритмов

3.13 Многоклассовая классификация

В задачах многоклассовой классификации каждый объект x требуется отнести к двум или более классам $1, 2, ..., K$. Такого рода задачи могут решаться как алгоритмами, способными сразу работать с любым числов классов, так и сведением к нескольким задачам бинарной классификации.

Подходы к решению:

1. Использовать алгоритмы, способные сразу выдавать вектор оценок принадлежности объекта к каждому классу (нейросети, ближайшие соседи и пр.).

2. One-vs-All: Решить K задач по отделению i-го класса от всех остальных. В качестве оценки принадлежности объекта классу i следует выбрать ответ i-го алгоритма на этом объекте:

   $$p(i|x) = a_i(x).$$

   Плюс в небольшом числе обучаемых классификаторов. Минус в том, что обучающие выборки становятся несбалансированными.

3. One-vs-One: Решить $K(K-1)/2$ задач бинарной классификации для каждой пары классов. В качестве оценки принадлежности объекта классу i следует взять сумму оценок принадлежностей этому классу всех алгоритмов, отделяющих i-й класс от остальных:

   $$p(i|x) = \sum_{j \neq i} a_{ij}(x).$$

   Плюс в том, что обучающие выборки сохраняют сбалансированность. Минус в том, что число обучаемых классификаторов квадратично велико.

4. Error-Correcting Output Code (ECOC): Закодировать каждый класс вектором из нулей и единиц некоторого размера k ($k \geqslant \log_2 K$). Кодирующие векторы следует выбирать так, чтобы они образовывали коды, исправляющие ошибки. Обучаем k бинарных классификаторов. На этапе предсказания для объекта формируем k-мерный вектор и относим его к тому классу, расстояние Хэмминга до которого минимально.

3.14 Дисбаланс классов

...чем плохо... как бороться (over/undersampling/SMOTE)...

3.15 Ансамбли алгоритмов

3.16 Метрики и функции потерь

Метрика - величина, обычно диктуемая бизнесом, оптимизация (максимизация или минимизация) которой вполне очевидным образом свидетельствует об улучшении качества работы модели.

Функция потерь - величина, более удобная для оценки/оптимизации модели, уменьшение которой, вообще говоря, приводит к оптимизации метрики задачи.

Иными словами, улучшение метрики - конечная цель процесса обучения алгоритма, но достигается это зачастую оптимизацией именно некоторой функции потерь, с которой может быть удобнее работать. Метрики и функции потерь - близкие понятия, когда речь идет об оценки качества алгоритма, и их довольно часто смешивают.

Пример: Пусть в задаче бинарной классификации основной метрикой является accuracy - доля правильных ответов. Эта метрика не дифференцируема, поэтому ее оптимизация напрямую методами гладкой оптимизации невозможна. В качестве функции потерь выберем MSE - среднеквадратичную ошибку. Это уже гладкая функция своих аргументов,и ее минимизация скорее всего приведет к увеличению доли правильных ответов, то есть к конечной цели.

3.17 Метрики бинарной классификации

Пусть некоторый алгоритм a решает задачу бинарной классификации с классами 0 (негативный) и 1 (позитивный). Тестирование алгоритма a проводится на n объектах, ответы y на которых известны. Пусть TP и TN - числа правильно классифицированных позитивных и негативных объектов соответственно. Аналогично, FP и FN - числа неправильно классифицированных позитивных и негативных объектов соответственно.

О качестве алгоритма a можно судить по матрице ошибок:

	y=1	y=0
a=1	TP	FP
a=0	FN	TN

Для оценки качества работы алгоритмов бинарной классификации обычно используются описанные далее основные метрики. FP - ошибка первого рода, FN - ошибка второго рода.

3.17.1 Accuracy

Точность (accuracy) - доля правильных ответов,

$$accuracy = \frac{TP + TN}{n}.$$

Проста в использовании и интерпретации, но плоха для несбалансированных выборок. Кроме того, не дифференцируема и потому не может быть использована напрямую в качестве функции потерь для алгоритмов гладкой оптимизации.

3.17.2 Balanced Accuracy

Сбалансированная точность вычисляется по формуле

$$BA = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$

то есть является средней от полноты обоих классов.

Проста в использовании и интерпретации, хороша для несбалансированных выборок. Не дифференцируема и потому не может быть использована напрямую в качестве функции потерь для алгоритмов гладкой оптимизации.

3.17.3 Коэффициент Меттьюса

Коэффициент Меттьюса вычисляется по формуле

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \in [-1, 1].$$

Хорош для несбалансированных выборок, но не дифференцируем.

Cohen's Kappa

Каппа Коэна вычисляется по формуле

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

$$p_o = \frac{TP + TN}{n}, \quad p_e = \frac{TN + FP}{n}\frac{TN + FN}{n} + \frac{TP + FP}{n}\frac{TP + FN}{n}.$$

Проста в использовании, хороша для несбалансированных выборок. Не дифференцируема и потому не может быть использована напрямую в качестве функции потерь для алгоритмов гладкой оптимизации.

3.17.4 Precision

Точность (precision) - отношение числа правильно классифицированных позитивных объектов к общему количеству позитивно классифицированных,

$$precision = \frac{TP}{TP + FP}.$$

Чем ближе значение к 1, тем меньше ложных срабатываний (FP). Проста в использовании и интуитивна, то не использует информацию о негативно классифицированных объектах и, кроме того, не является дифференцируемой.

3.17.5 Полнота (recall)

Полнота (recall) - вычисляется как отношение

$$recall = \frac{TP}{TP + FN}.$$

Чем ближе значение к 1, тем меньше ложных пропусков (FN). Проста в использовании и интуитивна, то не использует TN, FP и, кроме того, не является дифференцируемой.

3.17.6 F1-мера

F1-мера - среднее гармоническое точности и полноты,

$$F = \frac{2PR}{P + R}.$$

F1-мера усредняет точность и полноту, является неплохом компромиссом между обеими метриками. Проста в использовании, но плохо интерпретируема и не является дифференцируемой.

3.17.7 F-мера

Обобщенная F-мера вычисляется как

$$F = (1 + \beta^2)\frac{PR}{\beta^2 P + R}.$$

F-мера усредняет точность и полноту, является неплохом компромиссом между обеими метриками, имеет настраиваемый параметр β. Проста в использовании, но плохо интерпретируема и не является дифференцируемой.

3.17.8 ROC кривая

ROC кривая - характеристика качества алгоритмов бинарной классификации, дающих вероятностноподобный вывод, $a \in [0, 1]$. ROC кривая строится в координатах

$$FPR = \frac{FP}{FP + TN}, \quad TPR = \frac{TP}{TP + FN}.$$

Каждая точка кривой - значение (FPR, TPR), полученное для некоторого порога дискретизации алгоритма (см. 3.17.10).

Более простой способ построения ROC кривой состоит в следующем:

1. отрезки $[0, 1]$ по осям TPR и FPR разбиваются на $\#[y = 0]$ и $\#[y = 1]$ частей соответственно.

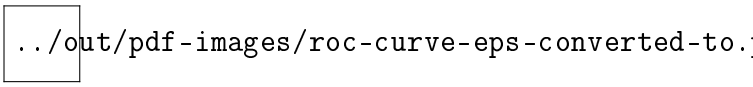
2. пары реальных ответов y_i упорядочиваются по убыванию соответствующих ответов алгоритма a_i.

3. проходя по получившемуся после сортировки массиву значений y_i, строим ROC кривую, начиная от начала координат и делая шаг вправо, если $y_i = 0$ и вверх, если $y_i = 1$. Важный момент: если рядом по порядку оказались несколько a_i с одинаковыми значениями, то соответствующий им участок ROC кривой будет не ступенчатым, а прямолинейным (см. пример ниже).

ROC кривая идеального алгорима проходит через точки $(0, 0)$, $(0, 1)$, $(1, 1)$; для случайного гадания - проходит вблизи прямой $FPR = TPR$. Наилучшим значением порога дискретизации алгоритма может считаться порог, соответствующий точке на ROC кривой, ближайшей к $(0, 1)$, либо точке, наиболее удаленной от прямой случайного гадания $TPR = FPR$.

Пример: для алгоритма, дающего вывод как в таблице ниже, график *ROC* кривой выглядит следующим образом

y	1	0	0	1	0	1	0
a	1.0	0.9	0.9	0.9	0.8	0.3	0.2



3.17.9 ROC-AUC

ROC-AUC - площадь под ROC кривой. Применяется к алгоритмам бинарной классификации, дающим вероятностноподобный вывод, $a \in [0, 1]$, позволяя оценить алгоритм "в целом без привязки к конкретному значению порога дискретизации алгоритма.

ROC-AUC принимает значения от 0 до 1. Значения близкие к 0.5 интерпретируются как самые худшие (случайное гадание), близкие к 1 - как хорошие. ROC-AUC более устойчива к дисбалансу классов, чем Accuracy, но не так хорошо, как

PR-AUC. ROC-AUC также не учитывает уверенность алгоритма в своих предсказаниях (насколько близко распределены предсказания к 0 и 1). Не является дифференцируемой.

ROC-AUC для примера 3.17.8 равна 2/3.

3.17.10 PR кривая

PR кривая - характеристика качества алгоритмов бинарной классификации, дающих вероятностноподобный вывод, $a \in [0, 1]$. PR кривая строится в координатах

$$recall = \frac{TP}{TP + FN}, \quad precision = \frac{TP}{TP + FP}.$$

Каждая точка кривой - значение $(recall, precision)$, полученное для некоторого порога дискретизации алгоритма.

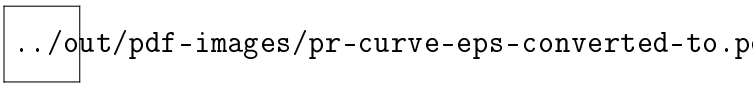
Способ построения PR кривой состоит в следующем:

1. вычисляются пороги h - всевозможные значения ответов алгоритма a.
2. ответы a_i дискретизируются для каждого значения порога и вычисляются значения $recall$ и $precision$. При этом ордината первой точки кривой, соответствующей порогу $h > 1$, не определена, так как знаменатель $precision$ обращается в ноль. В качестве ординаты берется ордината второй точки.
3. по полученным точкам строится график PR кривой.

PR кривая идеального алгорима проходит через точки $(0, 1)$, $(1, 1)$, $(1, \#[y = 1]/n)$; для случайного гадания - проходит вблизи прямой $precision = \#[y = 1]/n$.

Пример: для алгоритма, дающего вывод как в таблице ниже, график PR кривой выглядит следующим образом

y	1	0	0	1	0	1	0
a	1.0	0.9	0.9	0.9	0.8	0.3	0.2



3.17.11 PR-AUC

PR-AUC - площадь под PR кривой. Применяется к алгоритмам бинарной классификации, дающим вероятностноподобный вывод, $a \in [0, 1]$, позволяя оценить алгоритм "в целом без привязки к конкретному значению порога дискретизации алгоритма.

PR-AUC - площадь под PR кривой. Принимает значения от 0 до 1. Значения близкие к 1 интерпретируются как хорошие, близкие к $\#[y = 1]/n$ - как самые худшие (случайные гадания). PR-AUC более устойчива к дисбалансу классов, чем ROC-AUC, однако, не учитывает уверенность алгоритма в своих предсказаниях (насколько близко распределены предсказания к 0 и 1). Не является дифференцируемой.

PR-AUC для примера 3.17.10 равна $11/15 \approx 0.73$.

3.17.12 Бинарная кросс-энтропия (logloss)

Пусть y - истинная метка объекта (0 или 1), а a - ответы некоторого алгоритма (число из $[0, 1]$). Бинарная кросс-энтропия (logloss) вычисляется как

$$L(y, a) = -y \log_2 a - (1 - y) \log_2 (1 - a).$$

Слагаемые с нулевым множителем при логарифме (соответствующие $y = 0$ и $y = 1$) полагаются равными нулю.

Полная кросс-энтропия на множестве ответов определяется усреднением значений по всем объектам.

Бинарная кросс-энтропия имеет следующую вероятностную интерпретацию. Пусть метка i-му объекту назначается по схеме Бернулли, т.е. метка полагается равной $y_i = 1$ с вероятностью a_i и $y_i = 0$ с вероятностью $1 - a_i$. Тогда вероятность получить истинные ответы y_i равна

$$\prod_{i=1}^{n} a_i^{y_i} (1 - a_i)^{1-y_i}.$$

Логарифмируя, получаем правдоподобие, совпадающее с бинарной кросс-энтропией с точностью до знака. Таким образом, нахождение ответов a_i с позиции минимизации бинарной кросс-энтропии равносильно максимизации правдоподобия.

https://en.wikipedia.org/wiki/Loss$_f unctions_f or_c lassification$

3.18 Метрики многоклассовой классификации

3.18.1 Категориальная кросс-энтропия (logloss)

3.19 Индекс Джини

3.20 Метрики регрессии

3.20.1 Среднеквадратичная ошибка (MSE)

Пусть y - истинная метка объекта, а a - ответы некоторого алгоритма. Квадратичная ошибка вычисляется как

$$L(y, a) = (y - a)^2.$$

Полная среднеквадратичная ошибка на множестве ответов определяется усреднением значений по всем объектам. Наилучшим константным предсказанием для MSE является выборочное среднее:

$$a = \frac{1}{n} \sum_{i=1}^{n} y_i$$

MSE дифференцируема и проста в использовании, но плохо интрепретируема, так как дает ненормированный ни к чему результат, который трудно с чем-либо сравнить. Кроме того, MSE чувствительна к выбросам в выборке.

3.20.2 Среднеабсолютная ошибка (MAE)

Пусть y - истинная метка объекта, а a - ответы некоторого алгоритма. Абсолютная ошибка вычисляется как

$$L(y, a) = |y - a|.$$

Полная среднеквадратичная ошибка на множестве ответов определяется усреднением значений по всем объектам. Наилучшим константным предсказанием для MSE является выборочная медиана.

MAE проста в использовании, но недифференцируема и плохо интрепретируема, так как дает ненормированный ни к чему результат, который трудно с чем-либо сравнить. MAE менее чувствительна к выбросам в выборке, чем MSE.

3.20.3 Коэффициент детерминации (R^2)

Пусть y_i - истинные метки объектов x_i, а a_i - ответы некоторого алгоритма. Коэффициент детерминации R^2 вычисляется как

$$R^2(y, a) = 1 - \frac{\sum_{i=1}^{n}(y_i - a_i)^2}{\sum_{i=1}^{n}(y_i - \hat{y})^2}, \quad \hat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

R^2 показывает долю дисперсии y_i, объясняемую моделью. Является по сути линейной функцией от MSE, но более интерпретируема в силу нормировки к результату с константным прогнозом $\hat{y}$. Минусом является увеличение R^2 при увеличении числа признаков, что далеко не всегда свидетельствует о увеличении качества модели.

3.21 Метрики кластеризации

3.22 Разложение ошибки алгоритма

Будем считать, что обучающая выборка $X = (x_1, y_1), (x_2, y_2), \ldots (x_n, y_n) \in (\mathbb{X} \times \mathbb{Y})^n$ приходит из некоторого распределения с плотностью $f(x, y)$. Пусть также задан метод обучения $\mu : (\mathbb{X} \times \mathbb{Y})^n \to A$, который произвольной обучающей выборке ставит в соответствие алгоритм из некоторого семейства A. Будем использовать квадратичную функцию потерь, $L(y, a) = (y - a)^2$.

Мерой качества метода обучения μ естественно считать усредненный по всевозможным обучающим выборкам средний риск:

$$L(\mu) = \mathbb{E}_X \left[\mathbb{E}_{x,y} \left(y - \mu(X)(x)\right)^2\right]$$

Можно показать, что эта величина раскладывается в сумму трех:

$$L(\mu) = \underbrace{\mathbb{E}_{x,y}\left(y - a_*(x)\right)^2}_{\text{шум}} + \underbrace{\mathbb{E}_x\left(\mathbb{E}_X\mu(X)(x) - a_*(x)\right)^2}_{\text{смещение}} + \underbrace{\mathbb{E}_x\left[\mathbb{D}_X\mu(X)(x)\right]}_{\text{разброс}},$$

где $a_*(x) = \mathbb{E}_y\left[y|x\right]$ - наилучший из теоретически возможных алгоритм для среднеквадратичной функции потерь.

1. Шум (noise) - ошибка идеального алгоритма, невозможно как-либо уменьшить.

2. Смещение (bias) - отклонение среднего ответа обученного алгоритма от ответа идеального. Показывает, насколько хорошо можно приблизить идеальный алгоритм с помощью имеющихся данных. Смещение, как правило, маленькое у сложных семейств (деревьев) и велико у простых (линейных).

3. Дисперсия (variance) - разброс ответов обученных алгоритмов относительно их средних ответов. Показывает, насколько сильно может меняться ответ при варьировании выборки. Дисперсия, как правило, мала у простых алгоритмов и велика у сложных.

Таким образом, усложнением алгоритма можно добиться меньшего смещения, но при этом может существенно увеличиться разброс, что по сути является переобучением модели.

3.23 Кривые валидации

Кривые валидации - кривые, отражающие зависимость ошибки, либо некоторого функционала качества на выбранном датасете от некоторого параметра модели.

Обычно на одном графике строятся две кривые - одна для тренировочного датасета, другая для валидационного. Такие кривые позволяют понять, где качество модели на валидации проседает по сравнению с качеством на тренировке, давая переобучение модели.

Для последовательно оптимизируемых моделей (нейросетей, бустинга и т.п.) обычно строятся кривые валидации в виде зависимости качества/ошибки от итерации/эпохи обучения.

3.24 Кривые обучения

Кривые обучения - кривые, отражающие зависимость ошибки, либо какого-либо функционала качества на выбранном датасете от размера этого датасета.

Может оказаться, что сложность построения модели связанна не с недостатком имеющихся данных. Можно попробовать брать меньшие подвыборки и обучать модель на них, постепенно увеличивая их объем. Если начиная с некоторого момента

качество не будет существенно меняться, то для обучения модели вполне можно ограничиться этим меньшим объемом данных.

3.25 Метрические методы

3.26 Метод ближайших соседей

3.27 Линейные методы

3.28 Линейная регрессия

3.29 Логистическая регрессия

...отличие от линейной...

3.30 SVM

3.31 Ядра и спрямляющие пространства

3.32 Решаюшие деревья

Решающее дерево - бинарное дерево, во внутренних вершинах которого стоят предикаты $\beta : X \to \{0, 1\}$, а в листовых вершинах - элементы множества ответов Y (классы для задач классификации и числа для регрессии).

В качестве предикатов обычно используются простые линейные индикаторы $\beta_{jt}(x) = [x^j < t]$ (j-й признак меньше порога t).

Базовый алгоритм построения дерева решений (ID3 алгоритм):

1. Начинаем со всей обучающей выборки $U = x_{(1)}, ..., x_{(n)}$.

2. Проверяем, не выполнился ли для текущей вершины некоторый критерий останова (максимальное число листьев, минимальное число объектов в вершине, все объекты одного класса и пр.). Если это так, то объявляем вершину листовой, помещая в нее в качестве предсказания наиболее многочисленный класс (для задачи классификации), либо среднее из ответов (для регрессии).

3. Если критерий останова не выполнился, вершина объявляется внутренней. В нее помещается выбираем предикат, максимизирующий функционал качества

$$(j, t) = \operatorname{argmax} Q(U, j, t), \quad Q(U, j, t) = H(U) - \frac{|U_l|}{|U|} H(U_l) - \frac{|U_r|}{|U|} H(U_r),$$

где U_l, U_r - подмножества выборки U, удовлетворяюшие и не удовлетворяющие предикату β_{jt} соответственно, H - некоторый критерий информативности (см. ниже). Информативность стремимся минимизировать, функционал качества, соответственно - максимизировать.

4. Строим левое и правое поддерево, рекурсивно повторяя шаги 2-4 для подвыборок U_l, U_r.

Приведенный алгоритм - жадный, так как на каждом шаге он старается максимизировать функционал качества, не заботясь о том, будет ли это оптимальным выбором в целом.

Пусть p_k - доля объектов класса k в выборке U (части обучающей выборки, попавшей в любую вершину). Примеры критериев информативности:

1. $H(U) = 1 - \max_k p_k$ - ошибка классификации,
2. $H(U) = \sum_{k=1}^{K} p_k(1 - p_k)$ - критерий Джини,
3. $H(U) = -\sum_{k=1}^{K} p_k \log p_k$ - энтропийный критерий.

Достоинства решающих деревьев:

- хорошая интерпретируемость,
- допускаются пропуски в данных,
- высокая скорость обучения и работы.

Недостатки:

- склонность к переобучению,
- низкая статистическая значимость глубоких деревьев (выборка все меньше дальше от корня),
- чувствительность к шуму и выбору критерия инормативности.

С переобучением решающих деревьев обычно борятся стрижкой (pruning), либо объединением древьев в ансамбли - случайный лес.

3.33 Случайный лес

Случайный лес - алгоритм, строящийся на основе беггинга над решающими деревьями. Каждое дерево обучается по выборке, полученной бутстрапированием обучающей выборки и, кроме того, в каждой вершине разбиение ищется по некоторому подмножеству признаков. Таким образом достигается низкая коррелированность базовых алгоритмов, а следовательно, уменьшение разброса итогового ансамбля.

Следует отметить, что случайное подмножество признаков выбирается свое для каждой вершины каждого дерева, а не одно на каждое дерево.

Каждое дерево в случайном лесе обучается по своей бутстрап-подвыборке, что дает возможность автоматически сделать out-of-bag оценку качества ансамбля, так как бутстрап в среднем использует только 63% объектов:

$$OOB = \frac{1}{n}\sum_{i=1}^{n} L(y_i, r(x_i)),$$

где $r(x_i)$ - усредненный ответ для всех деревьев, которые не видели объект x_i на этапе обучения, L - некоторая функция потерь.

3.34 Ансамбли алгоритмов

3.34.1 Средняя ошибка независимых алгоритмов

Ансамблем алгоритмов называется алгоритм, использующий в своей работе другие (базовые) алгоритмы. Простейший пример ансамбля в задачах классификации - комитет большинства:

$$a(x) = \text{mode}(a_1(x), ..., a_n(x)).$$

Простейший пример ансамбля в регрессии - среднее ответов некоторых алгоритмов:

$$a(x) = \frac{1}{n}\sum_{j=1}^{n} a_i(x).$$

Если алгоритмы a_i независимы как случайные величины (функции случайной величины x - объекта предсказания), имеют одинаковые средние и ограниченные одним числом d дисперсии, то ансамбль a будет иметь то же среднее и меньшую дисперсию:

$$\mathbb{E}a = \mathbb{E}a_j, \quad \mathbb{D}a \leqslant \frac{d}{n}.$$

Таким образом, независимые алгоритмы в среднем подавляют ошибки друг друга, уменьшая тем самым разброс значений.

3.34.2 Основные подходы

В общем виде ансамбль алгоритмов имеет вид

$$a(x) = b(a_1(x), a_2(x), ..., a_l(x)),$$

где a_i - базовые алгоритмы, а b - метаалгоритм. Ансамбли могут быть полезны

- статистически - ошибка ансамбля оказывается меньше, чем у любого базового алгоритма,

- вычислительно - могут строить более быстро/параллельно,
- функционально - могут приближатьболее сложные функциональные зависимости.

Одна из основных идей ансамблирования состоит в уменьшении зависимости между базовыми алгоритмами, что ведет к уменьшению разброса итогового ансамбля. Понизить корреляцию между алгоритмами можно за счет:

- варьирования обучающей выборки - беггинг,
- варьирования множества признаков - метод случайных подпространств,
- варьирования целевого вектора - ECOC (исправляющее ошибки кодирование) и др.,
- варьирование моделей - стекинг (построение метапризнаков),
- итеративное уменьшение ошибки - бустинг,
- варьирование в модели - обучение одного алгоритма с разными параметрами, рандомизация в алгоритма (случайный лес).

При ансамблировании алгоритмов следует учитывать возможное переобучение, связанное с тем, что мы в итоге можем настраивать результирующий ансамбль, используя информацию обо всей обучающей выборке. Для оценки качества ансамбля лучше сохранять некоторую валидирующую часть обучающей выборки, невидимую для всех базовых алгоритмах на этапе их обучения и ансамблирования.

3.34.3 Беггинг

Беггинг - **B**ootstrap **Agg**regat**ing**. Метод, согласно которому базовый алгоритм (или несколько алгоритмов) обучается не на исходной обучающей выборке, а на ее бутстрап-подвыборках. Результат усредняется.

Таким образом, каждый алгоритм использует только порядка 63% исходной выборки, что уменьшает корреляцию между базовыми алгоритмами и, как следствие, повышает качество ансамбля - их среднего.

В некоторых вараяциях беггинга семплируют подвыборку объема не всей обучающей выборки, а меньшего (с возвращением или без) - пэстинг.

3.34.4 Метод случайных подпространств

Метод случайных подпространств заключается в том, что базовый алгоритм (или несколько алгоритмов) обучается не на всем множестве исходных признаках, а многократно на его подмножествах. Результат усредняется.

Этот подход может уменьшить корреляцию между базовыми алгоритмами и, как следствие, повысить качество ансамбля - их среднего.

3.34.5 Метод случайных патчей

Метод случайных подпространств и беггинг часто используется в связке и, таким образом, каждый базовый алгоритм обучается на своей подвыборке исходной обучающей выбокри и своем множестве признаков. На схожей идее построен алгоритм случайного леса с тем исключением, что там подмножество признаков не фиксируется для каждого дерева, а выбирается свое в каждой вершине каждого дерева.

3.34.6 Кросс-валидированный комитет

Разбиваем выборку на l фолдов и обучаем l алгоритмов каждый на (l-1) фолде. Результаты усредняются/голосование.

В этом случае для постороения результирующего ансамбля фактически используется вся выборка, поэтому желательно заранее оставлять валидационное множество объектов, чтобы избежать переобучения.

3.34.7 Блендинг

Постеший вариант стекинга. Подход предлагает смотреть на ответы базовых алгоритмов как на новые признаки, на которых можно обучать результирующий метаалгоритм.

Обучающая выборка разбивается на две части. На первой обучают базовые алгоритмы, затем используют их на второй части, получая их ответы как метапризнаки. На них обучается метаалгоритм. Для валидации/предсказания следует к выборке применить базовые алгоритмы, а затем к их результатам полученный метаалгоритм.

Недостатком блендинга является необходимость разделения исходной обучающей выборки на две части (или даже три, если планируется валидация). Бороться с этим можно, обучив несколько блендингов: базовые алгоритмы обучаются для разных разбиений обучающей выборки на две подвыборки. Результаты усредняют, либо конкатенируют и отправляют на обучение меаалгоритма. На валидации/тесте применяют обученные батчи базовых алгоритмов, результаты усредняют/конкатенируют и применяют метаалгоритм.

3.34.8 Стекинг

Подход стекинга предлагает смотреть на ответы базовых алгоритмов как на новые признаки, на которых можно обучать результирующий метаалгоритм.

Пусть имеется m некоторых базовых алгоритмов. Алгоритм стекинга:

1. Обучающая выборка разбивается на l фолдов.

2. Для каждого фолда (l-1) оставшихся объединяются в один и на нам тренируются все m базовых алгоритмов. На оставшемся фолде делаются предсказания.

3. После использования таким образом всех фолдов получится датасет того же размера по строкам, что и исходный тренировочный, но с m колонками - новыми метапризнаками.

4. На этом датасете из метапризнаков обучается метаалгоритм. При желании можно добавить в датасет некоторые старые или совершенно новые признаки.

Для выполнения предсказаний все базовые алгоритмы обучают на всей обучающей выборке (единовременно). Затем на валидации берут их ответы и на них окончательно берут ответ обученного ранее метаалгоритма.

3.34.9 Бустинг

Основная идея бустинга заключается в последовательном построении метаалгоритма как суммы базовых алгоритмы таким образом, что каждое последующее слагаемое исправляет ошибки предыдущих частичных сумм.

Пусть решается задача регрессии с функцией ошибки $L(y, a)$. Пусть уже имеется некоторый алгоритм a. Новый алгоритм b настраивается так, чтобы

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i) + b(x_i, w)) \to \min_w,$$

где w - искомый параметр, определающий уточняющий алгоритм b.

Эту идею можно применить последовательно:

1. Начать с $a_0(x) \equiv 0$

2. Для $k = 1, 2, ...$

$$(w_*, \eta_*) = \operatorname*{argmin}_{w,\eta} \frac{1}{n}\sum_{i=1}^{n} L(y_i, a_{k-1}(x_i) + \eta b(x_i, w)),$$

$$a_k(x) = a_{k-1}(x) + \eta_* b(w_*, x).$$

Здесь η - темп обучения - дополнительный параметр.

Первым вариантом бустинга был AdaBoost, использовавший в кажестве базовых алгоритмов решающие пни. На сегодняшний день его вытеснил алгоритм градиентного бустинга.

3.34.10 Градиентный бустинг

Основная задача бустинга

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i) + b(x_i, w)) \to \min_w$$

довольно трудно поддается решению напрямую. Желательно вместо такой постановки было бы иметь отдельную задачу на обучение добавляемого алгоритма b на выборке x_i с некоторыми ответами. Проблема в том, что нельзя просто взять $b_i = y_i - a(x_i)$, так как это эквивалентно исходной задаче только для простейшей квадратичной функции ошибки $L(y, a) = (y - a)^2/2$.

Для понимания идеи градиентного бустинга полезно переписать основную задачу в виде

$$F(b_1, ...b_n) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i) + b_i) \to \min_b .$$

Это не отнюдь означает, что после такой минимизации такое w, что $b(x_i, w) = b_i$, вообще найдется. Это просто наводящее соображение для того, на какой выборке лучше обучать алгоритм b.

Раскладывая F в ряд Тейлора около 0, с точностью до членов высшего порядка будем иметь

$$F(b_1, ...b_n) = const + \frac{1}{n}\sum_{i=1}^{n} L'(y_i, a(x_i))b_i + ...,$$

где производная берется по второму аргументу. Таким образом, для минимизации F наиболее выгодно выбрать $b_i = -L'(y_i, a(x_i))$ с точностью до некоторой мультипликативной постоянной, которую можно подобрать позже. Получается, что поправочный алгоритм b лучше всего обучать на выборке $(x_i, -L'(y_i, a(x_i)))_{i=1}^n$.

Для квадратичной функции ошибки $L(y, a) = (y-a)^2/2$ имеем выборку $(x_i, y_i - a(x_i))_{i=1}^n$, то есть новый алгоритм настраивается на ошибках предыдущего, что вполне согласуется с интуицией.

Замечания:

- После получения поправочного алгоритма b часто минимизируют величину

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i) + \eta b(x_i)) \to \min_\eta$$

- Для уменьшения эффекта переобучения сам поправочный алгоритм b добавляют к общей сумме с некоторым множителем - темпом обучения:

$$a_{t+1}(x) = a_t(x) + \eta b_t(x).$$

- Применение беггинга - обучение каждого нового поправочного алгоритма на своей подвыборке обучающей выборки.

3.35 Байесовские методы

Глава 4

Нейросети

В данной главе приводится обзор основных понятий и методов, связанных с нейросетями.