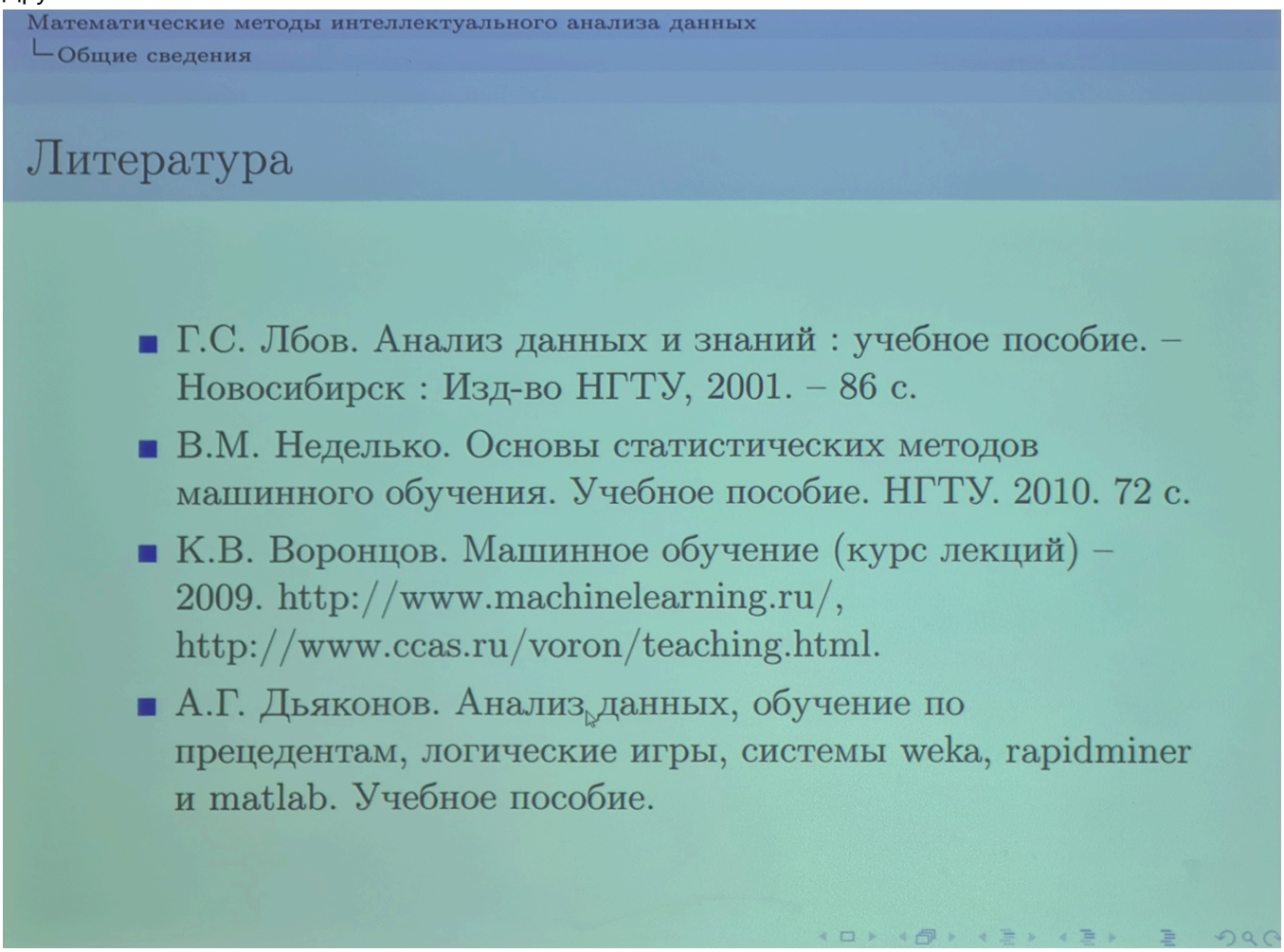
Инфо

Лектор - Неделько Виктор Михайлович

Семинарист - Денис Михайлович Михайлапов (Телега)

Хэндбук от Яндекса (*рекомендует не читать его сразу, а использовать как справочник*)

Другие книжки:



Kaggle - это мега-круто! Проводятся денежные конкурсы, публикуются решения задач, на рейтинг с Каггла работодатели обращают много внимания

Заметки с консультации

- Максимальная оценка = кол-во лаб. + 1, то есть за 3 сданных лабы можно получить оценку 4 и т.п.
- Механика - заходим, берём вопрос и тут же отвечаем (или почти тут же)
- Гистограммный классификатор - *не нужен (вопросов по нему не будет)*

Список экзаменационных вопросов

Нумерация соответствует той, что была на экзе в 24-м году

1. Метод kNN
2. Формула разделяющих кривых при нормальном распределении 2 классов и разных ковариационных матрицах (линейный дискриминант)
    - Тык (конкретно вот этот раздел)
3. Дискриминант Фишера
4. Формула логистической регрессии (сигмоид)
    - Тык
5. ...
6. Основная идея SVM
    - Тык
7. Kernel trick
    - Тык
8. ...
9. Напишите формулу критерия Gini (критерий ветвления деревьев)
    - Тык
    - Квадрат выводится из формулы среднего для вероятности ошибочного результата
10. Random forest - два основных принципа
    - Тык
11. Что такое бустинг? Как эта идея реализована в AdaBoost?
    - Тык
12. Основные параметры бустинга? Нарисовать зависимость точности модели от этих параметров
    - В целом о бустинге
    - Параметры: количество деревьев и их глубина (при этом глубину можно делать сильно меньше, чем в random forest)
    - Также сильно влияет количество зависимых переменных: чем больше зависимость, тем больше нужна выборка для той же точности
13. Почему бустинг лучше нейронных сетей на табличных данных?
    - Бустинг специально оптимизирован для решения задач с табличными данными, а потому решает их эффективнее. При этом нейросети в силу своей природы могут дать то же решение, что и бустинг, но с куда большими затратами по времени и ресурсам + с куда меньшей вероятностью, то есть их использование, конечно, возможно, но попросту неэффективно
    - Единственное исключение - табличные данные с несколькими разными целевыми переменными - для таких задач бустинг просто не приспособлен
    - *Ответ с консультации:*

        - Нейронкам очень сложно не учитывать какие-то из переменных
        - Построенные части в нейронке не будут фиксироваться
14. Как связаны бустинг и регрессия?
    - *Ответ с консультации (в моей интерпретации):* Бустинг может быть выражен через логистическую функцию, если мы настроим деревья так, что их результаты будут достаточно независимы
15. ROC-кривая
    - Тык
16. Precision-recall кривая
    - Тут есть ссылка на статью
17. Какую функцию ошибки лучше использовать для бинарной классификации
    - Вроде как logloss
18. Что такое переобучение и недообучение?
19. Напишите формулу дисперсии оценки точности классификации на выборке размера N
    - *Говорит, что это дисперсия Бернулли*
20. Теорема о несмещаемости кросс-валидации (формула в первую очередь)
    - Тык
21. В каком случае эмпирический риск (частота ошибок на обучающей выборке) является состоятельной оценкой риска (вероятности ошибки)? Запишите в мат. нотации
    - Смотри выводы к этому разделу
22. ...

Отсавшиеся вопросы (*их точные номера неизвестны*):

- Разложение bias
    - Тык
- Напишите формулу, подтверждающую гипотезу Наивного Байесовского классификатора
    - Тык
- Примеры выбросов, к которым SVM устойчив, а линейный дискриминант - нет
    - Вообще, для обоих методов лучше проводить фильтрацию, однако SVM в силу своей архитектуры (поиска векторов) будет куда слабее реагировать на редкие сильноудалённые точки в сравнении с линейным дискриминантом, который смотрит только на среднее и ковариации
    - ...

Лекция 1

Речь будет идти о "классическом" машинном обучении, то есть о методах, работающих с таблицами данных (изображения, речь, текст не являются табличными объектами и с ними работают нейронки), то есть набор чисел, которые связаны между собой

Классификация

Есть табличные данные с несколькими характеристиками, одна из которых - целевая. Необходимо будет используя остальные признаки, определить значение целевого

Метод прецедентов

Общая идея - смотрим на уже имеющиеся точки, чтобы сделать выводы о новой

Метод k-соседей

Смотрит на k ближайших значений по заданным параметрам и принимаем ключевое значение то же, что и бОльшего количества соседей

- Выбор k по большей части интуитивен, но в целом, зависит от выборки (очень желательно, чтобы k было намного меньше выборки)
- Второй вариант выбора k - это кросс-валидация - выбираем разные k и проверяем их корректность (надо делить выборку на тренировочную и тестовую часть)

Парзеновское окно

Смотрит на область, в которой размещаются точки с одинаковым ключевым признаком. Если новая точка располагается в нескольких областях, смотрим на то, каких точек больше

Ядровое сглаживание (kernel-based)

Вклад объектов в прогноз будет оцениваться при помощи фильтрующей функции расстояния (хороший вариант - гауссова функция расстояния)

Вот пример с использованием ядра Коши:

$$f(x) = \frac{1}{w}\sum_{i=1}^{N} y_i w_i$$
$$w = \sum_{i=1}^{N} w_i$$
$$w_i = \phi(\rho(x, x_i))$$
$$\phi(z) = \frac{1}{1 + (\frac{z}{r})^2}$$

Заключение

Метод прецедентов считается самым простым, но и потому работает хорошо не всегда. Достаточно не плох он будет при малом количестве переменных, а вот при большом количестве переменных будет работать сильно хуже (будут возникать шумовые переменные, по которым разделение на классы будет затруднено)

Таким образом, метод k-соседей будет актуален в двух случаях:

- Мало характеристических переменных либо мы смогли отфильтровать нужные
- Как дополнение к какому-то комплексному методу или нейронкам (так, у текста все переменные после прохода через энкодер будут полезными)

Лекция 2

(Квази-)линейные методы классификации

"Квази" заключается в том, что мы решение нелинейной задачи сводим к линейной

Линейный/квадратичный дискриминант

Метод, основанный на матстате.

Мы предполагаем, что распределение выборки имеет нормальную форму. В таком случае нам надо найти параметры выборки (*рассматриваем 2 класса:* $y \in \{-1, 1\}$)

Напомним функцию нормального распределния:

$$\phi_y(x) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(x-\mu_y)^2}{2\sigma_y^2}}$$

Далее нам остаётся наложить друг на друга плотности распределения для каждого класса. У какого класса в заданной точке выше вероятность, тот и более вероятен. Область пересечения нескольких классов будет считаться ошибкой (то есть ошибку можно посчитать, взяв интеграл по этим областям)

Сама решающая функция Байеса будет выглядеть так:

$$f(x) = \arg\max_y \phi(x, y)$$
$$\phi(x, y) = \phi_y(x) P(y)$$

- $\phi_y(x)$ - функция распределния класса y (нормального распределения)

Если вероятность ошибки достаточно высока, то разумнее всего будет выдать пользователю информацию о том, какова вероятность отнесения к различным классам

Можно также записать решающую функцию через логарифм:

$$f(x) = \begin{cases} 1, l(x) \geq 0 \\ -1, l(x) < 0 \end{cases}$$
$$l(x) = \ln \phi_1(x) - \ln \phi_{-1}(x) + \ln \frac{P(1)}{P(-1)}$$

Подставив сюда функции нормально распределния с параметрами σ, μ, получим:

$$l(x) = \frac{(x-\mu_{-1})^2}{\sigma_{-1}^2} - \frac{(x-\mu_1)^2}{\sigma_1^2} + \ln \frac{\sigma_{-1}}{\sigma_1} + \ln \frac{P(1)}{P(-1)}$$

При $l(x) = 0$ это будет квадратное уравнение, имеющее 2 корня, либо 1 при $\sigma_1 = \sigma_{-1}$ либо 0 корней в вырожденных случаях

Случай N переменных

Также рассматриваем случай двух классов Для каждого класса y:

- μ_y - вектор матожиданий
- λ_y - ковариационная матрица

Решающая функция та же: $l(x) = \ln \phi_1(x) - \ln \phi_{-1}(x) + \ln \frac{P(1)}{P(-1)}$, а вот функция плотности будет такой:

$$\phi_y(x) = \frac{1}{\sqrt{2\pi}|\lambda_y|^{n/2}} e^{-\frac{1}{2}Q_y(x)}$$
$$Q_y(x) = (x-\mu_y)^T (\lambda_y)^{-1} (x-\mu_y)$$

Подставляем формулу плотности в $l(x)$ и получаем:

$$2l(x) = Q_{-1}(x) - Q_1(x) + \ln|\lambda_{-1}| - \ln|\lambda_1| + 2\ln P(1) - 2\ln P(-1)$$

Подставляем значения $Q_y(x)$ и после преобразований получаем:

$$2l(x) = x^T Ax + bx + c$$
$$A = (\lambda_{-1})^{-1} - (\lambda_1)^{-1}$$
$$b = 2\mu_1(\lambda_{-1})^{-1} - 2\mu_{-1}(\lambda_1)^{-1}$$
$$c = \mu_{-1}^T(\lambda_{-1})^{-1}\mu_{-1} - \mu_1^T(\lambda_1)^{-1}\mu_1 + \ln|\lambda_{-1}| - \ln|\lambda_1| + 2\ln P(1) - 2\ln P(-1)$$

Уравнение $l(x) = 0$ в общем случае даст нам плоскость второго порядка, которая при равенстве ковариационных матриц выродится в гиперплоскость

Примерные параметры

- N_y - число объектов класса y
- N - общее число объектов
- I_y - множество индексов объектов класса y

Тогда имеем:

$$\tilde{P}(y) = \frac{N^y}{N}$$
$$\tilde{\mu}_y = \frac{1}{N_y}\sum_{i \in I_y} x_i$$
$$\tilde{\lambda}_y = \frac{1}{N_y}\sum_{i \in I_y}(x_i - \tilde{\mu}_y)(x_i - \tilde{\mu}_y)^T$$

Для линейного и квадратичного правила нужно разное количество объектов в выборке: для линейного правила от 1000 объектов с 100 параметрами мы получим адекватный результат, тогда как для квадратичного правила этого уже не хватит

Дискриминант Фишера

Заключается в построении такой границы между классами (на самом деле только определении наклона границы), чтобы проецкии каждого класса на эту гарницу были максимальны

Формула легко находится во второй лекции и слишком тривиальна, чтобы переписывать её сюда

Преимущества метода в отсутствии необходимости делать вероятностные предположения и оценки всего по $n + 1$ параметру (то есть как линейный дискриминант, то есть не требователен к объёму выборки)

Однако метод неустойчив к выбросам (впрочем, их можно отфильтровать достаточно тривиально)

Логистическая регрессия

Функция условной вероятности для класса 1 может быть представлена так:

$$g(x) = \frac{1}{1 + e^{-l(x)}}$$

То есть в виде **сигмоида**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Подставив нормальные плотности при равзных матрицах ковариации, мы получаем, что

$$g(x) = \sigma(wx + w_0)$$

Здесь w и w_0 - переобозначенные b и c из метода дескриминанта

Метод логистической регрессии основан на оценивании условной вероятности моделью $\tilde{g}(x) = \sigma(wx + w_0)$, где параметры w, w_0 могут как подбираться вручную, так и максимизации критерия правдоподобия:

$$-K_V(w, w_0) = \sum_{i \in I_1} \ln \tilde{g}(x_i) + \sum_{i \in I_{-1}} \ln \tilde{g}(x_i)$$

Метод похож на линейный дискриминант, но ослабляет вероятностное предположение и куда более устойчив к выбросам

Наивный байесовский классификатор

Основывается на предположение о независимости переменных, что чаще всего оказывается неоправданно, однако метод, тем не менее, может дать достаточно точную оценку

Из формулы Байеса выводим такую функцию условной вероятности для класса 1:

$$p = P(y = 1)$$
$$g(x) = P(y = 1|x) = \frac{P(dx, y = 1)}{P(dx, y = 1) + P(dx, y = -1)} = \frac{1}{1 + \frac{1-p}{p} + \frac{P(dx, y=-1)}{P(dx, y=1)}} =$$
$$= \sigma(\ln p - \ln(1-p) + \ln \frac{P(dx, y = -1)}{P(dx, y = 1)}) =$$
$$= \sigma(\sigma^{-1}(p) + \ln \frac{P(dx, y = -1)}{P(dx, y = 1)})$$

Выразим правую часть:

$$\ln \frac{P(dx, y = -1)}{P(dx, y = 1)} = \sigma^{-1}(g(x)) - \sigma^{-1}(p)$$

$\sigma^{-1}(p)$ - контснта. Далее будем обозначать её как z_0 Из независимости переменных делаем следущий вывод:

$$\ln \frac{P(dx, y = -1)}{P(dx, y = 1)} = \ln(\prod \frac{P(dx_j, y = -1)}{P(dx_j, y = 1)}) = \sum \ln \frac{P(dx_j, y = -1)}{P(dx_j, y = 1)}$$

Теперь получаем:

$$\sigma^{-1}(g(x)) - z_0 = \sum(\sigma^{-1}(g_j(x_j)) - z_0)$$

Обозначим $z_j = \sigma^{-1}(g_j(x_j)) - z_0$ и тогда:

$$g(x) = \sigma(z_0 + \sum w_j z_j), w_j \equiv 1$$

Таким образом, задача свелась к **логистической регрессии**

Предположение о независиомсти переменных можно ослабить, предположив w_j свободными переменными и подобравив их каким-либо методом

Лекция 3

Метод опорных векторов

Похож по своей сути на дискриминант Фишера, однако здесь мы ищем разделяющую поверхность максимально отдалённую не от среднего обоих классов, а от ближайших к разделяющей поверхности точек (то есть от **крайних точек**)

В качестве решающей функции берётся пороговая функция классификации: $f(x) = sign(wx - w_0)$. Таким образом, задача сводится к нахождению оптимальных вектора w и скаляра w_0

В случае линейно разделимой выборки

Теперь зададим условия для поиска этих значений:

Условие нормировки (V - какая-то выборка переменных и класса из всех значений):

$$\min_{x_i, y_i \in V} y_i(wx_i - w_0) = 1$$

Минимум по этому критерию будет достигнут на каких-то граничных точках, при этом ширина разделяющей полосы окажется максимальной. Обозначим граничные точки как x_+, x_-, тогда условие примет вид:

$$(x_+w - w_0) = 1$$
$$-(x_-w - w_0) = 1$$

Ширина разделяющей полосы при этом составит $\frac{2}{|w|}$

Таким образом, максимизация разделяющей полосы будет эквивалентна минимизации нормы вектора (или его квадрата), а значит получаем такую оптимизационную задачу:

$$\begin{cases} w^2 \to \min_{w, w_0, \xi_i} \\ y_i(x_iw - w_0) \geq 1 \end{cases}$$

Получается тривиальная задача квадратичной оптимизации, для которой существует ряд алгоритмов, например, INCAS

Если выборка линейно неразделима

В этом случае не существует вектора w, который бы удовлетворял приятному и простому выведенному выше условию. Требуется его ослабить

В этом случае добавим константу C для регулировки силы влияния погрешности и меры неотрицательных ошибок ξ_i, которые также желательно получить минимальными:

$$\begin{cases} \frac{w^2}{2} + C\sum \xi_i \to \min \\ y_i(x_iw - w_0) \geq 1 - \xi_i \\ \xi_i > 0 \end{cases}$$

Далее задача будет решать теми же методами

Тут идёт какая-то слишком страшная муть про двойственную задачу

Kernel trick

Во многих случаях оказывается удобно перейти от исходного пространства переменных к какому-то новому, в котором точки будут уже линейно разделимы (**спрямляющему пространству**)

Скалярное произведение переменных в спрямляющем пространстве при этом будет какой-то новой функцией от этих же переменных. Эту функцию назовём **ядром**. Можно и вовсе не находить новое пространство, а сразу задать ядро. Этот приём называется **kernel trick**

Выводы

В итоге мы получаем разреженное решение известным методом квадратичного программирования. При этом:

- Возможно обобщение функцией ядра
- Можно работать с беспризнаковыми объектами

Однако при этом SVM сильно язвим для шума и смешанной выборки + использование kernel trick - уже не самая тривиальная задача

Лекция 4

Логические методы классификации

В целом будут сводиться к поиску закономерностей в выборке и выводе на их основе хорошего предиката, то есть такого, которые определяет верно наибольшее количество фактов принадлежности к классу. По своей сути закономерностями в данном случае называются области в плоскости, достаточно простые геометрически, чтобы их можно было описать предикатами.

Критерий хорошести предиката определяется при помощи двух выражений:

$$a = \frac{m}{M}$$
$$b = \frac{n - m}{n}$$

где:

- M - истинное число точек класса
- n - число точек, для которых предикат истинный
- m - число точек нужного класса, для которых предикат истинный

Можно комбинировать a, b Разными способами:

$$\frac{a}{b} \to \min$$
$$a - b \to \min$$
$$\sqrt{a} - \sqrt{b} \to \min$$
$$\dots$$

Далее немного мутно про методы оценки силы закономерности, выделю отсюда лишь формулу для силы отвергания нулевой гипотезы:

$$U(m) = \sum_{m'=m}^{M} \frac{C_n^{m'} C_{N-n}^{M-m'}}{C_N^M}$$

Чем она меньше, тем сильнее закономерность

Используя формулу энтропии и Стерлинга (*их и вывод я тут писать не буду*), выводим **информационный критерий**, который отражает количество информации в закономерности, представляющее собой разность между изначальной энтропией системы и её энтропией после выделения заданной области.

Решающие списки

В их основе лежит "жадный" алгоритм

- Для начала нужно как-то дискретно поделить пространство переменных
- Затем запустить на нём алгоритм поиска закономерностей по типу КОРА или ТЕМП
- В результате мы получаем список закономерностей со значениями их критериев качества
- Сортируем список по критерию
- Идём от головы списка и применяем закономерности к точке
- Первая истинная закономерность даёт нам искомый класс
    - Можно также взять несколько истинных закономерностей с самым высоким значением критерия и проводить между ними "голосование" (*что бы это ни значило... Ну я бы возможно случайно выбирал одну из закономерностей, взевшивая их вероятности значениями критерия*)

Решающие деревья

Суть метода заключается в построении иерархического разбиения пространства (непересекающихся подмножеств, образующих исходное множество)

Плюсы:

- Наглядность и понятность
- Варьирование сложности решений
- Автоматический выбор информативных переменных

Построения разбиения, максимально точно классифицирующего выборку - NP-полная задача, поэтому для её решения используются эвристические алгоритмы:

- Жадный (*Описывать его этапы подробно не вижу смысла, так как он невероятно банален и прост*)
- Рекурсивный алгоритм - предикат в узле выбирается с учётом ветвления на нижнем уровне
- Неограниченный - строит дерево, затем его оптимизирует, пока не кончится время

Критерии (в т.ч. Gini)

Можно пользоваться простыми критериями для каждого предиката в листе дерева, однако зачастую удобнее оказывается использовать общий критерий для всего дерева, имеющий следующий вид:

$$K = \sum_{t=1}^{T} N^t L(\nu_1^t, \nu_2^t, ..., \nu_k^t)$$

где:

- N^t - число объектов в листе t
- $\nu_1^t, \nu_2^t, ..., \nu_k^t$ - относительные частоты классов в листе t
- $L(...)$ может определяться по-разному:

- Число ошибок классификации - величина неверно классифицированных объектов, если объектам в листе приписан наиболее частый в ней класс
  - $$L_E(\nu_1^t, \nu_2^t, ..., \nu_k^t) = 1 - \max_{\omega}(\nu_\omega^t)$$
- Критерий Джини - величина ошибки, если класс будем приписывать случайно в зависимости от частот всех классов в листе
  - $$L_G(\nu_1^t, \nu_2^t, ..., \nu_k^t) = 1 - \sum_{\omega=1}^{k}(\nu_\omega^t)^2$$
- logloss - критерий максимального правдоподобия
  - $$L_G(\nu_1^t, \nu_2^t, ..., \nu_k^t) = -\sum_{\omega=1}^{k}\nu_\omega^t \ln \nu_\omega^t$$

Лекция 5

Композиция (ансамбли) решающих функций

Композиция - это буквально то, что мы и понимаем под композицией. То есть это будет функция (монотонная), которая принимает в качестве аргументов значения множества ршеающих функций и отображает его на множество решений

Линейная композиция:

$$\lambda(x) = sign(\sum \alpha_t \lambda_t(x))$$

Если $\alpha_t \geq 0$, то композиция называется **выпуклой**

В первую очередь речь здесь будет идти о методах сочтетания методов, уже изученных выше, как я понял

Независимые подвыборки (random forest)

Bagging - bootstrap aggregation - бутстрап-выборка - заключается в случайном копировании объектов из исходной выборки с возможными повторениями. Новая выборка по размеру должна быть как исходная. В итоге мы получим, что примерно 37% элементов исходной выборки попадут в новую

Случайный лес - заключается в построении усреднённого решающего дерева на основании ансамбля деревьев (леса), каждое дерево в котором тренируется на своей бутстрап-выборке. При этом для каждого дерева может браться случайный набор переменных. Сложность алгоритма при этом не увеличивается с ростом числа деревьев, а качество не может упасть, однако достаточно быстро выходит на плато (обычно больше сотни деревьев использовать смысла нет)

Бустинг

Здесь мы будем строить новые подвыборки, изменяя веса объектов исходной (и не только)

Основные принципы бустинга:

1. Каждый следующий базовый классификатор должен исправлять ошибки предыдущего

2. Мы не пытается исправить уже построенные классификаторы, а просто включаем их в композицию

AdaBoost

Итоговая решающая функция - линейная написанная выше

Изначально берутся какие то веса объектов $w^1 = (w_1^1, ..., w_n^1)$

- S - подвыборка
- λ_t - базовый классификатор

Построив базовый классификатор, можем посчитать его вес:

$$\alpha_t = \frac{1}{2} \ln \frac{W^+(S, w^t, \lambda_t)}{W^-(S, w^t, \lambda_t)}$$
$$W^+(S, w^t, \lambda_t) = \sum_{i=1}^{N} w_i^t I(y_i = \lambda(x_i))$$
$$W^-(S, w^t, \lambda_t) = \sum_{i=1}^{N} w_i^t I(y_i = -\lambda(x_i))$$

По-человечески: W^+, W^- - общие веса верно и неверно классифицированных объектов

Теперь строим новые веса объектов:

$$w_i^{t+1} = \frac{\overline{w_i^{t+1}}}{\sum_{j=1}^{N} \overline{w_j^{t+1}}}$$
$$\overline{w_i^{t+1}} = w_i^t e^{\alpha_t y_i \lambda_t(x_i)}$$

Вспомогательный конструкт надо просто запомнить, а первое - деление одного вспомогательного веса не сумму всех вспомогательных весов.

- Веса правильно классифицированных объектов домножаем на $e^{-\alpha_t}$, а неверно - e^{α_t}

Чаще всего с бустингом используются решающие деревья. При этом играют роль глубина дерева и количество деревьев

Как правило, бустинг использует деревья меньшей глубины и при этом достигает лучшего качества

Количество независимых переменных прямо влияет на качество бустинга и необходимый размер выборки: чем больше независимых переменных, тем меньше необходимый объём выборки и наоборот

Ансамбли разных решающих функций

Сюда включаются как использование совсем разных методов, так и использование формально одного метода, но с разными фиксированными параметрами

Простое голосование

От простого подбора весов в линейной решающей функции отличие в том, что мы хотим, чтобы значения, близкие к 0 или 1 имели больший вес, чем значения вроде 0,5, поэтому используем

обратную логистическую функцию к результатам всех базовых функций, затем возвращая их к обычному виду, а далее уже применяем простое среднее

Стэкинг

Заключается в том, что мы добавляем результаты работы базовых функций как переменные (либо замещаем ими изначальные переменные), а дальше решаем задачу обычным образом

НЕОБХОДИМО ИСПОЛЬЗОВАТЬ КРОСС-ВАЛИДАЦИЮ

Лекция 6

Тут речь шла про нейронные сети, но материал настолько простой, что мне показался не заслуживающим упоминания. Интерес могут представлять последние слайды из презентации

В методичке это страница 95

Лекция 7

Оценивание качества решений

Качество принятого решения оценивается функцией потерь:

$$L : Y^2 \to [0, +\infty)$$

Риск - среднее функции потерь:

$$R(c, \lambda) = EL = \int_D L(y, \lambda(x)) P_c(dx, dy)$$

*Далее идут какие-то пространные рассуждения, из которых я разве что сделал вывод, что в нашем случае **риск - случайная величина***

При поиске функции оценки риска нам важны:

- Несмещаемость
- Состоятельность
- Эффективность

Каждое из этих свойств имеет пояснения, но они слишком душные

Контрольная выборка

Точечная оценка

Тривиально считаем среднее:

$$R^*(S^*, \lambda) = \frac{1}{N^*} \sum L(y^i, \lambda(x^i))$$

Если L - индикаторная функция, то точечная оценка приобретёт вид простой доли ошибочных ответов решающей функции

Доверительный интервал

Опуская душный матан, можно сказать, что это та область, в которой могут находиться резлуьтаты модели (откровенно говоря, этот ответ не говорит вообще ничего, так что лучше ещё раз о нём перечитать)

Байесовский подход и уравнивание

Байесовский подход даёт нам более точную оценку, чем простое среднее и при этом удобнее для пользователя, чем доверительный интервал. Основывается на формуле Байеса

Другой вариант - приравнять доверительный интервал и вероятность ошибки, что тоже даст нам неплохую оценку риска

Дальше шла ещё какая-то муть про случаи, когда функция потерь - не индикаторная функция, вывели пару "полезных" формул и устновили, что делить контрольную выборку на фолды бессмысленно

Выводы

- Простота
- Несмещённость (условная)
- Состоятельность
- Известен точный доверительный интервал
- Эффективность (условая)
- Нужна дополнительная выборка

Эмпирический риск

Формула - также просто среднее, но теперь не по новой выборке, а по обучающей. Является смещённой в силу "подгонки" решающей функции в ходе обучения

Оценки Вапника-Червоненкинса

Напишу только выводы. В их ходе использовалась ЦПТ и табличный интеграл Ф

Если будем брать частоту ошибок $\nu = \frac{N_e}{N}$, то вероятность того, что ν отличается от вероятности ошибки p на ϵ будет $< e^{-2\epsilon^2 N}$

Если у нас будет L решающих функций, то получим $\epsilon \approx \sqrt{\frac{\ln L}{2N}}$

Ввиду бесконечного количества решающих функций за L обычно берут количество классов эквивалентности решающих функций

*Вся эта херабора вроде как доказывает **состоятельность** эмпирического риска (а далее нам говорят, что практической пользы от этого мало... Ну заебись...)*

Затем идёт подраздел, который даже Неделько советует скипать (я проигнорирую его уточнеие, что "при первом прочтении"). Вывод из этого раздела - оценка такая будет неулучшаема, пока у нас нет новой информации

Выводы

- Простота вычисления
- Сильная смещённость
    - **В случае большой выборки и применении методов малой ёмкости (линейных или наивного Байесовского) смещение становится незначительным, поэтому эмпирический риск можно использовать как непосредственную оценку риска**
- Состоятельность при ограничении на сложность метода
- Малая дисперсия

В общем случае оценка эмпирического риска имеет смысл для верхней оценки качества и для контроля процесса обучения

Скользящий экзамен (leave-one-out, hold-out, кросс-валидация и т.п.)

Заключается в том, что мы определённым образом делим выборку и считаем функцию потерь на её части, проверяя на другой

Leave-one-out - выкидываем один элемент, на остальной части учимся, на одном элементе проверяем, повторяем N раз и усредняем

K-fold кросс-валидация - делим всю выборку на K равных частей, тренируем на K-1 фолде, а на последнем тестируем, повторяем N/K раз и усредняем

Hold-out - как прошлое, но повторяем только один раз, то есть какой-то фолд выбирается тренировочным

Несмещённость кросс-валидации

Я хз, как эту дугу в латехе делать, так что будет скрин (стр. 168):

Теорема. Для оценки кроссвалидации имеет место

$$\mathsf{E}\breve{R}^K(S_N, Q) = \mathsf{E}R(S_{N-\frac{N}{K}}, Q).$$

Как частный случай

$$\mathsf{E}\breve{R}(S_N, Q) = \mathsf{E}R(S_{N-1}, Q).$$

Говоря человечески, матожидание риска k-fold кросс-валдиации будет равно матожиданию риска для выборки, из которой мы исключили число элементов, содержащееся в одном фолде

Доказывается достаточно тривиально:

- Рассмотрим hold-out - так как мы не использовали в оценке объекты из одного фолда, ожидаемая ошибка на них такая же, как на любых новых
- Матожидание суммы - это суммма матожиданий

Дисперсия не зависит от количества фолдов (практически)

Ещё некоторые методы скользящего экзамена

Случайная подвыборка - как hold-out, только для тестового фолда набираем элементы случайным образом и повторяем многократно

Повторные разбиения - добавляем к кросс-валидации перемешивание

Стратификация - разбиваем на фолды, сохраняя соотношение классов. На практике при кросс-валидации имеет не так много смысла

Bootstrap - используем для тренировки bootstrap-выборку, а для проверки - не попавшую в неё часть исходной выборки

Out-of-bag - после кросс-валидации мы не строим решение уже по всей выборке, а усредняем полученные решения с каждой итерации

Свойства

- Относительно просто вычисляется
- Несмещённый (пускай и не в том смысле)
- Состоятельный
- Большая дисперсия
- Среднеквадратичный разброс по фолдам практически никак не связан с дисперсией всей оценки

Выводы

- Наилучший способ оценки риска неизвестен
- На практике обычно используют скользящие методы, все разновидности которого ПРИМЕРНО равноценны
- Нет точных оценок доверительного интервала для риска по обучающей выборке
- Полезным может оказаться использование статистического моделирования

Лекция 8

Критерии качества

Критерий качества должен так или иначе учитывать нашу цель и цену ошибки. Объективно оценить второе порой бывает затруднительно, однако дать примерную оценку вполне возможно

Виды критериев:

- Целевой критерий - отражает реальные потери от ошибочно принятого решения. Мы его не выбираем
- Критерий для обучения - может использовать или нет целевую функцию потерь, может вообще не выражаться через потери, подвергается регуляризации

Критерии точности классификации

Матрица ошибок составляется так:

```
TP FP
TN FN
```

Подробнее тут (в т.ч. про precision-recall-кривую)

ROC-кривая

ROC-кривая - кривая ошибок

Самый большой практический смысл играет площадь под этой кривой (AUC) - чем она больше, тем лучше (идеальный вариант, это прямая по точкам (0,0), (0, 1), (1, 1))

Для обычной решающей функции AUC - это среднее между вероятностями правильного прогнозирования каждого из классов

Подробнее

Для непрерывной функции

Ну... Просто среднеквадратичное или среднемедианное отклонение считаем

Разложение ошибки

Bias-variance decomposition (разложение bias)

Разложение на смещение и разброс

Ну... В начале там идёт формула:

$$E(u-v)^2 = Du + (Eu - Ev)^2 + Dv$$

А дальше какая-то лютая дичь начинается... Надо найти материал на этот счёт нормальный. В методичке стр. 192

Могут характеризовать процесс обучения лишь косвенно

Ответ от GPT: Получаем «шум», смещение и разброс (в точке x).

1. Du — шум:
    - Это свойство данных: их разброс вокруг истинного значения. Даже если модель идеальная, данные всё равно могут содержать шум.
    - Например, при измерении температуры в одной точке термометр может показывать разные значения из-за погрешностей.
2. $(Eu - Ev)^2$ — смещение (bias):
    - Смещение измеряет, насколько среднее предсказание модели Ev отклоняется от истинного значения Eu.
    - Если модель систематически "не дооценивает" или "переоценивает" истинное значение, это проявляется в виде смещения.
    - Например, если модель всегда предсказывает значение на 5 ниже реального, это смещение.
3. Dv — разброс (varience):
    - Разброс показывает, насколько нестабильны предсказания модели v для одного и того же объекта x, если обучать модель на разных выборках.
    - Это отражает чувствительность модели к обучающим данным. Если модель слишком сложная (например, переобучилась), то её предсказания сильно зависят от конкретной

выборки.

Теория статистической устойчивости

А этого в известных билетах нет, так что идёт нахер - стр. 194